Volume 29 Number 1 May 2005 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

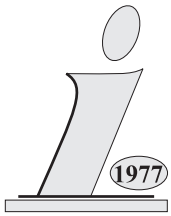
Special Issue:

Computational Intelligence in Data mining

Guest Editors:

Janos Abonyi

Ajith Abraham



The Slovene Society Informatika, Ljubljana, Slovenia

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Special Issue on Computational Intelligence in Data mining

In our society the amount of data doubles almost every year. Hence, there is an urgent need for a new generation of computationally intelligent techniques and tools to assist humans in extracting useful information (knowledge) from the rapidly growing volume of data.

When we attempt to solve real-world problems, like extracting knowledge from large amount of data, we realize that they are typically ill-defined systems, difficult to model and with large-scale solution spaces. In these cases, precise models are impractical, too expensive, or non-existent. Furthermore, the relevant available information is usually in the form of empirical prior knowledge and input–output data representing instances of the system's behavior. Therefore, we need an approximate reasoning system capable of handling such imperfect information. While Bezdek [2] defines such approaches within a frame called computational intelligence, Zadeh [3] explains the same using the soft computing paradigm. According to Zadeh "... in contrast to traditional, hard computing, soft computing is tolerant of imprecision, uncertainty, and partial truth." In this context Fuzzy Logic (FL), Probabilistic Reasoning (PR), Neural Networks (NNs), and Evolutionary Algorithms (EAs) are considered as main components of CI. Each of these technologies provide us with complementary **reasoning** and **searching** methods to solve complex, real-world problems. What is important to note is that soft computing is not a melange. Rather, it is a partnership in which each of the partners contributes a distinct methodology for addressing problems in its domain. In this perspective, the principal constituent methodologies in CI are complementary rather than competitive [4].

This special issue deals with the importance of computational intelligence (CI) paradigms in data mining and knowledge discovery.

The first paper is aimed to give a comprehensive view about the links between computational intelligence and data mining. Further, a case study is also given in which the extracted knowledge is represented by fuzzy rule-based expert systems obtained by soft computing based data mining algorithms. It is recognized that both model performance and interpretability are of major importance, and effort is required to keep the resulting rule bases small and comprehensible. Therefore, CI technique based data mining algorithms have been developed for feature selection, feature extraction, model optimization and model reduction (rule base simplification). The results illustrate that that CI based tools can be applied in a synergistic manner though the nine steps of knowledge discovery.

The remaining papers were selected from the papers presented at the **4th International Conference on Intelligent Systems Design and Application (ISDA'04 at August 26-28, 2004)** on the basis of fundamental ideas/concepts rather than the thoroughness of techniques deployed.

The second paper of this special issue by **Górriz, Segura, Puntonet and Salmerón** presents a survey of forecasting preprocessing techniques. Authors have illustrated that these methods could play a major role in the final model accuracy.

Many computational intelligence method can be used for clustering purposes beside the classical techniques. **Liu, Özyer, Alhajj and Barker** in the third paper propose a new clustering algorithm integrating multi-objective genetic algorithm and validity analysis.

In the fourth paper, **Podgorelec, Kokol, Heričko and Rozman** present a method for data classification which is based on constructing decision graphs with the help of several agents. They present a two-leveled evolutionary algorithm for the induction of decision graphs and describe the principle of classification based on the decision graphs.

Kotsiantis and Pintelas in the fifth paper combine a simple Bayesian classification method with Logitboost, which is a bias reduction technique. Logitboost requires a regression algorithm for base learner. For this reason, they slightly modify simple Bayesian classifier in order to be able to run as a regression method. They performed a large-scale comparison with other state-of-the-art algorithms and ensembles on 27 standard benchmark datasets and the empirical results looks very interesting.

In the sixth paper, **Wang and Garibaldi** propose a method useful in cancer diagnosis called Simulated Annealing Fuzzy Clustering. This technique can solve two major problems that exist in simpler methods: it does not need the number of clusters in advance and is able to avoid sub-optimal solutions in some sense.

Ivancsy and Vajk in the seventh paper focus on mining frequent patterns in large transactional databases. The contribution of their new method is to count the short patterns in a very fast way, using a specific index structure.

An important research field within pattern recognition is the analysis of time series data. In the eighth paper, **Toshniwal and Joshi** introduce a novel approach for performing similarity search in time series data. Their technique is based on the intuition that similar time sequences will have similar variations in their slopes.

Chong, Abraham and Paprzycki in the ninths paper propose a hybrid model involving decision trees and neural networks for classification of the type of injury severity of various traffic accidents. They also consider neural networks trained using hybrid learning approaches, support vector machines, and decision trees.

Ahvenlampi and Kortela in the last paper propose a hybrid system for controllability of quality in continuous digesters in which Self-Organizing Maps and Gustafson-Kessel fuzzy clustering algorithm are used.

The editors wish to thank Professor Matjaz Gams(Editor-in-Chief of Informatica)for providing the opportunity to edit this special issue on Computational Intelligence in Data Mining. We would also like to thank

the referees who have critically evaluated the papers within the short stipulated time. Finally we hope the reader will share our joy and find this special issue very useful.

Janos Abonyi and Ajith Abraham

Computational Intelligence in Data Mining

Janos Abonyi and Balazs Feil
University of Veszprem, Department of Process Engineering,
P.O. Box 158, H-8201 Veszprem, Hungary, abonyij@fmt.vein.hu
www.fmt.vein.hu/softcomp

Ajith Abraham
School of Computer Science and Engineering,
Chung-Ang University, Seoul, S. Korea, ajith.abraham@ieee.org
http://ajith.softcomputing.net

Keywords: KDD, Computational Intelligence, Soft Computing, Fuzzy Classifier System, Rule Base Reduction, Visualization

Received: December 20, 2004

This paper is aimed to give a comprehensive view about the links between computational intelligence and data mining. Further, a case study is also given in which the extracted knowledge is represented by fuzzy rule-based expert systems obtained by soft computing based data mining algorithms. It is recognized that both model performance and interpretability are of major importance, and effort is required to keep the resulting rule bases small and comprehensible. Therefore, CI technique based data mining algorithms have been developed for feature selection, feature extraction, model optimization and model reduction (rule base simplification). Application of these techniques is illustrated using the Wine data classification problem. The results illustrate that that CI based tools can be applied in a synergistic manner though the nine steps of knowledge discovery.

Povzetek: Rudarjenje podatkov je podano v povezavi z računsko inteligenco.

1 Introduction

In our society the amount of data doubles almost every year. Hence, there is an urgent need for a new generation of computationally intelligent techniques and tools to assist humans in extracting useful information (knowledge) from the rapidly growing volume of data.

Historically the notion of finding useful patterns in data has been given a variety of names including data mining, knowledge extraction, information discovery, and data pattern processing. The term data mining has been mostly used by statisticians, data analysts, and the management information systems (MIS) communities.

The term knowledge discovery in databases (KDD) refers to the overall process of discovering knowledge from data, while data mining refers to a particular step of this process. Data mining is the application of specific algorithms for extracting patterns from data [1]. The additional steps in the KDD process, such as data selection, data cleaning, incorporating appropriate prior knowledge, and proper interpretation of the results are essential to ensure that useful knowledge is derived form the data.

KDD has evolved from the intersection of research fields such as machine learning, pattern recognition, databases, statistics, artificial intelligence, and more recently it gets new inspiration from computational intelligence.

When we attempt to solve real-world problems, like extracting knowledge from large amount of data, we realize that they are typically ill-defined systems, difficult to model and with large-scale solution spaces. In these cases, precise models are impractical, too expensive, or non-existent. Furthermore, the relevant available information is usually in the form of empirical prior knowledge and input–output data representing instances of the system's behavior. Therefore, we need an approximate reasoning system capable of handling such imperfect information. While Bezdek [2] defines such approaches within a frame called computational intelligence, Zadeh [3] explains the same using the soft computing paradigm. According to Zadeh "... in contrast to traditional, hard computing, soft computing is tolerant of imprecision, uncertainty, and partial truth." In this context Fuzzy Logic (FL), Probabilistic Reasoning (PR), Neural Networks (NNs), and Evolutionary Algorithms (EAs) are considered as main components of CI. Each of these technologies provide us with complementary **reasoning** and **searching** methods to solve complex, real-world problems. What is important to note is that soft computing is not a melange. Rather, it is a partnership in which each of the partners contributes a distinct methodology for addressing problems in its domain. In this perspective, the principal constituent methodologies in CI are complementary rather than competitive [4].

The aim of this paper is to illustrate how these elements

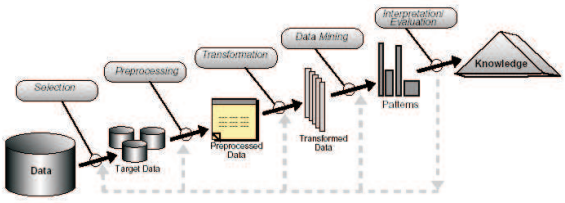


Figure 1: Steps of the knowledge discovery process.

of CI could be used in data mining. This special issue is focused on some of the theoretical developments and advances in this field.

Steps of Knowledge Discovery

Brachman and Anand [5] give a practical view of the KDD process emphasizing the interactive nature of the process. Here we broadly outline some of its basic steps depicted in Fig. 1 taken from [6], and we show the connections of these steps to CI based models and algorithms.

1. *Developing and understanding the application domain, the relevant prior knowledge, and identifying the goal of the KDD process.* The transparency of fuzzy systems allows the user to effectively combine different types of information, namely linguistic knowledge, first-principle knowledge and information from data. An example for the incorporation of prior knowledge into data-driven identification of dynamic fuzzy models of the Takagi-Sugeno type can be found in [7] where the prior information enters to the model through constraints defined on the model parameters. In [8] and [9] a different approach has been developed which uses block-oriented fuzzy models.

2. *Creating target data set.*

3. *Data cleaning and preprocessing*: basic operations such as the removal of noise, handling missing data fields.

4. *Data reduction and projection*: finding useful features to represent the data depending the goal of the task. Using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration or to find invariant representation of data. Neural networks [10], cluster analysis [11], Markov blanket modeling [12], decision trees [13], evolutionary computing [14] and neuro-fuzzy systems are often used for this purpose.

5. *Matching the goals of the KDD process to a particular* **data mining method**: Although the boundaries between prediction and description are not sharp, the distinction is useful for understanding the overall discovery goal. The goals of knowledge discovery are achieved via the following data mining methods:

   - **Clustering**: Identification of a finite set of categories or clusters to describe the data. Closely related to clustering is the method of probability density estimation. Clustering quantizes the available input-output data to get a set of prototypes and use the obtained prototypes (signatures, templates, etc., and many writers refer to as codebook) and use the prototypes as model parameters.
   - **Summation**: finding a compact description for subset of data, e.g. the derivation of summary for association of rules and the use of multivariate visualization techniques.
   - **Dependency modeling**: finding a model which describes significant dependencies between variables (e.g. learning of belief networks).
   - **Regression**: learning a function which maps a data item to a real-valued prediction variable and the discovery of functional relationships between variables.
   - **Classification**: learning a function that maps (classifies) a data item into one of several predefined classes.
   - **Change and Deviation Detection:** Discovering the most significant changes in the data from previously measured or normative values.

6. *Choosing the* **data mining algorithm(s)**: selecting algorithms for searching for patterns in the data. This includes deciding which model and parameters may be appropriate and matching a particular algorithm with the overall criteria of the KDD process (e.g. the end-user may be more interested in understanding the model than its predictive capabilities.) One can identify three primary components in any data mining algorithm: model representation, model evaluation, and search.

   - **Model representation**: the language is used to describe the discoverable patterns. If the representation is too limited, then no amount of training time or examples will produce an accurate model for the data. Note that more powerful representation of models increases the danger of overfitting the training data resulting in reduced prediction accuracy on unseen data. It is important that data analysts fully comprehend the representational assumptions which may be inherent in a particular method.

     For instance, rule-based expert systems are often applied to classification problems in fault detection, biology, medicine etc. Among the wide range of CI techniques, fuzzy logic improves classification and decision support systems by allowing the use of overlapping class definitions and improves the interpretability of the

results by providing more insight into the classifier structure and decision making process [15]. In Section 2 a detailed discussion about the use of fuzzy techniques for knowledge representation in classifier systems will be given.

- **Model evaluation criteria**: qualitative statements or fit functions of how well a particular pattern (a model and its parameters) meet the goals of the KDD process. For example, predictive models can often judged by the empirical prediction accuracy on some test set. Descriptive models can be evaluated evaluated along the dimensions of predictive accuracy, novelty, utility, and understandability of the fitted model.

  Traditionally, algorithms to obtain classifiers have focused either on accuracy or interpretability. Recently some approaches to combining these properties have been reported; fuzzy clustering is proposed to derive transparent models in [16], linguistic constraints are applied to fuzzy modeling in [15] and rule extraction from neural networks is described in [17]. Hence, to obtain compact and interpretable fuzzy models, reduction algorithms have to be used that will be overviewed in Section 3.

- **Search method**: consists of two components: parameter search and model search. Once the model representation and the model evaluation criteria are fixed, then the data mining problem has been reduced to purely an optimization task: find the parameters/models for the selected family which optimize the evaluation criteria given observed data and fixed model representation. Model search occurs as a loop over the parameter search method [18].

  The automatic determination of fuzzy classification rules from data has been approached by several different techniques: neuro-fuzzy methods [19], genetic-algorithm based rule selection [20], hybrid combination of genetic algorithm and neural learning [21] and fuzzy clustering in combination with GA-optimization [22] [23]. For high-dimensional classification problems, the initialization step of the identification procedure of the fuzzy model becomes very significant. Several CI based tools developed for this purpose will be presented in Section 4.

7. *Data mining*: searching for patterns of interest in a particular representation form or a set of such representations: classification rules or trees, regression. Some of the CI models lend themselves to transform into other model structure that allows information transfer between different models. For example, in [24] a decision tree was mapped into a feedforward neural network. A variation of this method is given in [25] where the decision tree was used for the input domains discretization only. This approach was extended with a model pruning method in [26]. Another example is that as radial basis functions (RBF) are functionally equivalent to fuzzy inference systems [27, 28], tools developed for the identification of RBFs can also be used to design fuzzy models.

8. *Interpreting mined patterns*, possibly return to any of the steps 1-7 described above for further iteration. This step can also involve the visualization of the extracted patterns/models, or visualization of the data given the extracted models. Self-Organizing Map (SOM) as a special clustering tool that provides a compact representation of the data distribution, hence it has been widely applied in the visualization of high-dimensional data [29]. In Section 5 the theory and in Section 6 the application of SOM will be presented.

9. *Consolidating discovered knowledge*: incorporating this knowledge into another system for further action, or simply documenting and reporting it.

The remainder of this article is organized as follows. In the remaining sections, tools for visualization, knowledge representation, classifier identification and reduction are discussed. The proposed approaches are experimentally evaluated for the three-class Wine classification problem. Finally, conclusions are given in Section 7.

2 Effective Model Representation by Fuzzy Systems

2.1 Classifier Systems

The identification of a classifier system means the construction of a model that predicts whether a given pattern, $\boldsymbol{x}_k = [x_{1,k}, \ldots, x_{n,k},]$, in which $y_k = \{c_1, \ldots, c_C\}$ class should be classified. The classic approach for this problem with C classes is based on Bayes' rule. The probability of making an error when classifying an example $\boldsymbol{x}$ is minimized by Bayes' decision rule of assigning it to the class with the largest posterior probability:

$$\boldsymbol{x} \text{ is assigned to } c_i \Longleftrightarrow p(c_i|\boldsymbol{x}) \geq p(c_j|\boldsymbol{x})\, \forall j \neq i \quad (1)$$

The *a posteriori* probability of each class given a pattern $\boldsymbol{x}$ can be calculated based on the $p(\boldsymbol{x}|c_i)$ class conditional distribution, which models the density of the data belonging to the c_i class, and the $P(c_i)$ class prior, which represents the probability that an arbitrary example out of data belongs to class c_i

$$p(c_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|c_i)P(c_i)}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|c_i)P(c_i)}{\sum_{j=1}^{C} p(\boldsymbol{x}|c_j)P(c_j)} \quad (2)$$

As (1) can be rewritten using the numerator of (2) we would have an optimal classifier if we would perfectly estimate the class priors and the class conditional densities. Of

course in practice one needs to find approximate estimates of these quantities on a finite set of training data $\{\boldsymbol{x}_k, y_k\}$, $k = 1, \ldots, N$. Priors $P(c_i)$ are often estimated on the basis of the training set as the proportion of samples of class c_i or using prior knowledge. The $p(c_i|\boldsymbol{x})$ class conditional densities can be modeled with non-parametric methods like histograms, nearest-neighbors or parametric methods such as mixture models.

2.2 Fuzzy Rules for Providing Interpretability of Classifiers

The classical fuzzy rule-based classifier consists of fuzzy rules that each describe one of the C classes. The rule antecedent defines the operating region of the rule in the n-dimensional feature space and the rule consequent is a crisp (non-fuzzy) class label from the $\{c_1, \ldots, c_C\}$ set:

$$r_i : \textbf{If } x_1 \text{ is } A_{i,1}(x_{1,k}) \textbf{ and } \ldots x_n \text{ is } A_{i,n}(x_{n,k}) \textbf{ then } \hat{y} = c_i, \quad [w_i] \tag{3}$$

where $A_{i,1}, \ldots, A_{i,n}$ are the antecedent fuzzy sets and w_i is a certainty factor that represents the desired impact of the rule. The value of w_i is usually chosen by the designer of the fuzzy system according to his or her belief in the accuracy of the rule. When such knowledge is not available, $w_i = 1, \forall\, i$ is used.

The **and** connective is modeled by the product operator allowing for interaction between the propositions in the antecedent. Hence, the degree of activation of the ith rule is calculated as:

$$\beta_i(\boldsymbol{x}_k) = w_i \prod_{j=1}^{n} A_{i,j}(x_{j,k}) \tag{4}$$

The output of the classical fuzzy classifier is determined by the *winner takes all* strategy, i.e. the output is the class related to the consequent of the rule that has the highest degree of activation:

$$\hat{y}_k = c_i^* \ , \ i^* = \arg\max_{1 \le i \le C} \ \beta_i(\boldsymbol{x}_k) \tag{5}$$

The fuzzy classifier defined by the previous equations is in fact a quadratic Bayes classifier when $\beta_i(\boldsymbol{x}_k) = p(\boldsymbol{x}|c_i)P(c_i)$.

As the number of the rules in the above representation is equal to the number of the classes, the application of this classical fuzzy classifier is restricted. In the [30], a new rule-structure has been derived to avoid this problem, where the $p(c_i|\boldsymbol{x})$ posteriori densities are modeled by $R > C$ mixture of models

$$p(c_i|\boldsymbol{x}) = \sum_{l=1}^{R} p(r_l|\boldsymbol{x})P(c_i|r_l) \tag{6}$$

This idea results in fuzzy rulebase where the consequent of rule defines the probability of the given rule represents the $c_1, \ldots, c_C$ classes:

$$r_i : \textbf{If } x_1 \text{ is } A_{i,1}(x_{1,k}) \textbf{ and } \ldots x_n \text{ is } A_{i,n}(x_{n,k}) \textbf{ then } \hat{y}_k = c_1 \text{ with } P(c_1|r_i) \ \ldots, \ \hat{y}_k = c_C \text{ with } P(c_C|r_i) \ [w_i] \tag{7}$$

The aim of the remaining part of the paper is to review some techniques for the identification of the fuzzy classifier presented above. In addition, methods for reduction of the model will be described.

3 Model Evaluation Criteria and Rule Base Reduction

Traditionally, algorithms to obtain best classifiers have been based either on accuracy or interpretability. Recently some approaches to combining these properties have been reported; fuzzy clustering is proposed to derive transparent models in [16], linguistic constraints are applied to fuzzy modeling in [15] and rule extraction from neural networks is described in [17].

3.1 Similarity-driven rule base simplification

The similarity-driven rule base simplification method [31] uses a similarity measure to quantify the redundancy among the fuzzy sets in the rule base. A similarity measure based on the set-theoretic operations of intersection and union is applied:

$$S(A_{i,j}, A_{l,j}) = \frac{|A_{i,j} \cap A_{l,j}|}{|A_{i,j} \cup A_{l,j}|} \tag{8}$$

where $|.|$ denotes the cardinality of a set, and the $\cap$ and $\cup$ operators represent the intersection and union of fuzzy sets, respectively. S is a symmetric measure in [0,1]. If $S(A_{i,j}, A_{l,j}) = 1$, then the two membership functions $A_{i,j}$ and $A_{l,j}$ are equal. $S(A_{i,j}, A_{l,j})$ becomes 0 when the membership functions are non-overlapping. The complete rule base simplification algorithm is given in [31].

Similar fuzzy sets are merged when their similarity exceeds a user defined threshold $\theta \in [0, 1]$ (θ=0.5 is applied). Merging reduces the number of different fuzzy sets (linguistic terms) used in the model and thereby increases the transparency. The similarity measure is also used to detect "don't care" terms, i.e., fuzzy sets in which all elements of a domain have a membership close to one. If all the fuzzy sets for a feature are similar to the universal set, or if merging led to only one membership function for a feature, then this feature is eliminated from the model. The method is illustrated in Fig. 2

3.2 Multi-Objective Function for GA based Identification

To improve the classification capability of the rule base, genetic algorithm (GA) optimization method can be applied [32] where the cost function is based on the model

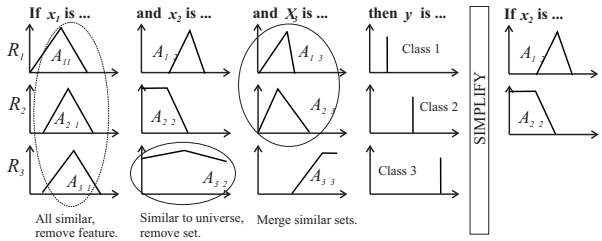


Figure 2: Similarity-driven simplification.

accuracy measured in terms of the number of misclassifications. Also other model properties can be optimized by applying multi-objective functions. For example in [33] to reduce the model complexity, the misclassification rate is combined with a similarity measure in the GA objective function. Similarity is rewarded during the iterative process, that is, the GA tries to emphasize the redundancy in the model. This redundancy is then used to remove unnecessary fuzzy sets in the next iteration. In the final step, fine tuning is combined with a penalized similarity among fuzzy sets to obtain a distinguishable term set for linguistic interpretation.

The GAs is subject to minimize the following multi-objective function:

$$J = (1 + \lambda S^*) \cdot Error\,, \tag{9}$$

where $S^* \in [0, 1]$ is the average of the maximum pairwise similarity that is present in each input, i.e., S^* is an aggregated similarity measure for the total model. The weighting function $\lambda \in [-1, 1]$ determines whether similarity is rewarded ($\lambda < 0$) or penalized ($\lambda > 0$).

3.3 Other Reduction Algorithms

The application of orthogonal transforms for reducing the number of rules has received much attention in recent literature [34]. These methods evaluate the output contribution of the rules to obtain an importance ordering. For modeling purpose Orthogonal Least Squares (OLS) is the most appropriate tool [35]. Evaluating only the approximation capabilities of the rules, the OLS method often assigns high importance to a set of redundant or correlated rules. To avoid this, in [36] some extension for the OLS method was proposed.

Using too many input variables may result in difficulties in the interpretability capabilities of the obtained classifier. Hence, selection of the relevant features is usually necessary. Others have focused on reducing the antecedent by similarity analysis of the fuzzy sets [33], however this method is not very suitable for feature selection. Hence, for this purpose, Fischer interclass separability method which is based on statistical properties of the data [37] has been modified in [38].

4 CI based Search Methods for the Identification of Fuzzy Classifiers

Fixed membership functions are often used to partition the feature space [20]. Membership functions derived from the data, however, explain the data-patterns in a better way. The automatic determination of fuzzy classification rules from data has been approached by several different techniques: neuro-fuzzy methods [19], genetic-algorithm based rule selection [20] and fuzzy clustering in combination with GA-optimization [22]. For high-dimensional classification problems, the initialization step of the identification procedure of the fuzzy model becomes very significant. Common initializations methods such as grid-type partitioning [20] and *rule generation on extrema* initialization [39], result in complex and non-interpretable initial models and the rule-base simplification and reduction step become computationally demanding.

4.1 Identification by Fuzzy Clustering

To obtain compact initial fuzzy models fuzzy clustering algorithms [22] or similar but less complex covariance based initialization techniques [38] were put forward, where the data is partitioned by ellipsoidal regions (multivariable membership functions). Normal fuzzy sets can then be obtained by an orthogonal projection of the multivariable membership functions onto the input-output domains. The projection of the ellipsoids results in hyperboxes in the product space. The information loss at this step makes the model suboptimal resulting in a much worse performance than the initial model defined by multivariable membership functions. However, gaining linguistic interpretability is the main advantage derived from this step. To avoid the erroneous projection step multivariate membership functions [40] or clustering algorithms providing axis-parallel clusters can be used [30]

4.2 Other Initialization Algorithms

For the effective initialization of fuzzy classifiers crisp decision tree-based initialization technique is proposed in [41]. DT-based classifiers perform a rectangular partitioning of the input space, while fuzzy models generate non-axis parallel decision boundaries [42]. Hence, the main advantage of rule-based fuzzy classifiers over crisp-DTs is the greater flexibility of the decision boundaries. Therefore fuzzy classifiers can be more parsimonious than DTs and one may conclude that the fuzzy classifiers, based on the transformation of DTs only [43], [44] will usually be more complex than necessary. This suggests that the simple transformation of a DT into a fuzzy model may be successfully followed by model reduction steps to reduce the complexity and improve the interpretability. The next section proposes rule-base optimization and simplification steps for this purpose.

5 Clustering by SOM for Visualization

The Self-Organizing Map (SOM) algorithm performs a topology preserving mapping from high dimensional space onto map units so that relative distances between data points are preserved. The map units, or neurons, form usually a two dimensional regular lattice. Each neuron i of the SOM is represented by an l-dimensional weight, or model vector $\boldsymbol{m}_i = [m_{i,1}, \ldots, m_{i,l}]^T$. These weight vectors of the SOM form a codebook. The neurons of the map are connected to adjacent neurons by a neighborhood relation, which dictates the topology of the map. The number of the neurons determines the granularity of the mapping, which affects the accuracy and the generalization capability of the SOM.

SOM is a vector quantizer, where the weights play the role of the codebook vectors. This means, each weight vector represents a local neighborhood of the space, also called Voronoi cell. The response of a SOM to an input $\boldsymbol{x}$ is determined by the reference vector (weight) $\boldsymbol{m}_i^0$ which produces the best match of the input

$$i^0 = \arg\min_i \ \|\boldsymbol{m}_i - \boldsymbol{x}\| \tag{10}$$

where i^0 represents the index of the Best Matching Unit (BMU).

During the iterative training, the SOM forms an elastic net that folds onto "cloud" formed by the data. The net tends for approximate the probability density of the data: the codebook vectors tend to drift there where the data are dense, while there are only a few codebook vectors where the data are sparse. The training of SOM can be accomplished generally with a competitive learning rule as

$$\boldsymbol{m}_i^{(k+1)} = \boldsymbol{m}_i^{(k)} + \eta \Lambda_{i^0,i}(\boldsymbol{x} - \boldsymbol{m}_i^{(k)}) \tag{11}$$

where $\Lambda_{i^0,i}$ is a spatial neighborhood function and η is the learning rate. Usually, the neighborhood function is

$$\Lambda_{i^0,i} = \exp\left(\frac{\|\boldsymbol{r}_i - \boldsymbol{r}_i^0\|^2}{2\sigma^{2(k)}}\right) \tag{12}$$

where $\|\boldsymbol{r}_i - \boldsymbol{r}_i^0\|$ represents the Euclidean distance in the output space between the i-th vector and the winner.

6 Case study: Wine Classification by CI techniques

6.1 Wine Data

The Wine data [1] contains the chemical analysis of 178 wines grown in the same region in Italy but derived from three different cultivars. The problem is to distinguish the three different types based on 13 continuous attributes derived from chemical analysis. : Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Non-flavanoid phenols, Proanthocyaninsm color intensity, Hue, OD280/OD315 of diluted wines and Proline (Fig. 3).

6.2 Fuzzy Classifier Identified by GA

An initial classifier with three rules was constructed by the covariance–based model initialization technique proposed in [38] using all samples resulting in 90.5% correct, 1.7% undecided and 7.9% misclassifications for the three wine classes. Improved classifiers are developed based on the GA based optimization technique discussed in Section 3.2. Based on the similarity analysis of the optimized fuzzy sets, some features have been removed from individual rules, while the interclass separability method have been used to omit some features in all the rules. The achieved membership functions are shown in Fig. 4, while the obtained rules are shown in Table 1.

6.3 Fuzzy Classifier Identified by Fuzzy Clustering

A fuzzy classifier, that utilizes all the 13 information profile data about the wine, has been identified by the clustering algorithm proposed in [30], where the obtained classifier is formulated by rules given by (7). Fuzzy models with three and four rules were identified. The three rule-model gave only 2 misclassification (98.9%). When a cluster was

[1] The Wine data is available from the University of California, Irvine, via anonymous ftp *ftp.ics.uci.edu/pub/machine-learning-databases.*

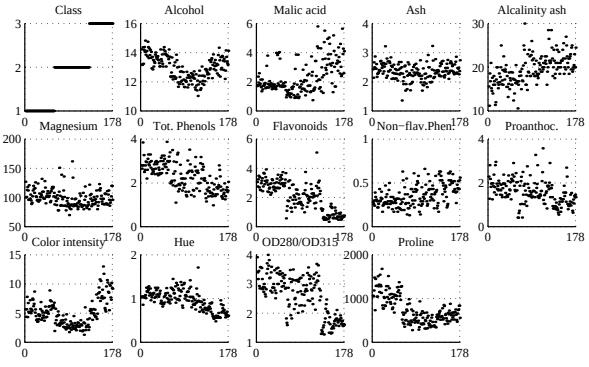


Figure 3: Wine data: 3 classes and 13 attributes.

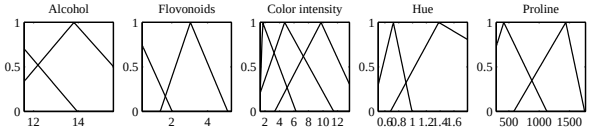


Figure 4: The fuzzy sets of the optimized three rule classifier for the Wine data.

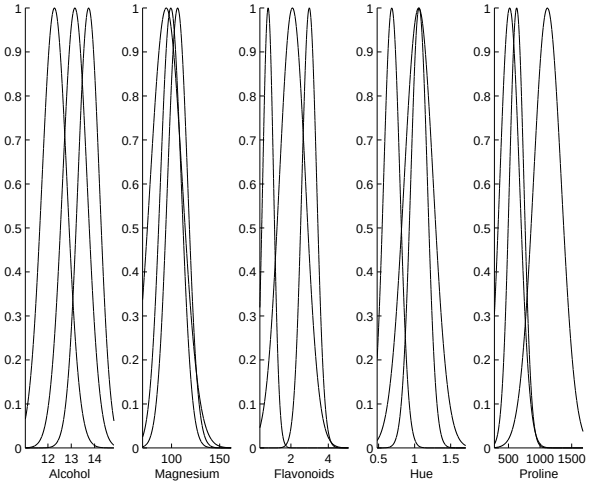


Figure 5: Membership functions obtained by fuzzy clustering.

added to improve the performance of this model, the obtained classifier gave only 1 misclassification (99.4%).

The classification power of the identified models is compared with fuzzy models with the same number of rules obtained by Gath-Geva clustering, as Gath-Geva clustering can be considered the unsupervised version of the proposed clustering algorithm. The Gath-Geva identified fuzzy model gives 8 (95.5%) misclassification when the fuzzy model has three rules and 6 (96.6%) misclassification with four rules. These results indicate that the proposed clustering method effectively utilizes the class labels.

The interclass separability based model reduction technique is applied to remove redundancy and simplify the obtained fuzzy models and five features were selected. The clustering has been applied again to identify a model based on the selected five attributes. This compact model with three, four and five rules gives four, two and zero misclassification, respectively. The resulted membership functions and the selected features are shown in Fig. 5.

6.4 Visualization by SOM

The SOM presented in Section 5. has been utilized to visualize the Wine data. SOM can be effectively used for correlation hunting, which procedure is useful for detecting the redundant features. It is interesting to note that the rules given in Table 1 can easily validated by the map of the variables given in Fig. 6

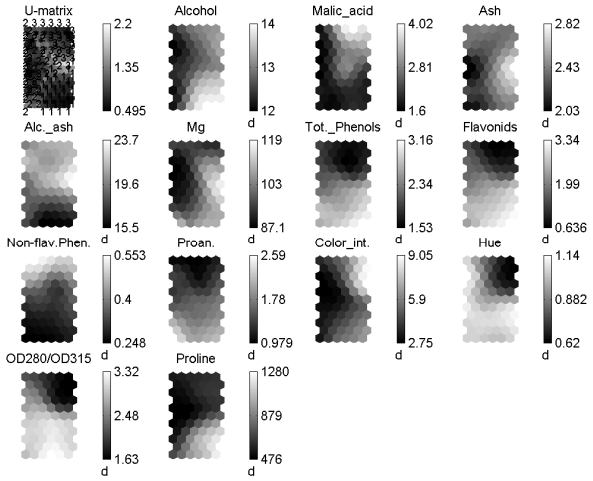


Figure 6: Self-Organizing Map of the Wine data

6.5 Discussion

The Wine data is widely applied for comparing the capabilities of different data mining tools. Corcoran and Sen [45] applied all the 178 samples for learning 60 non-fuzzy if-then rules in a real-coded genetic based-machine learning approach. They used a population of 1500 individuals and applied 300 generations, with full replacement, to come up with the following result for ten independent trials: best classification rate 100%, average classification rate 99.5% and worst classification rate 98.3% which is 3 misclassifications. Ishibuchi et al. [20] applied all the 178 samples designing a fuzzy classifier with 60 fuzzy rules by means of an integer-coded genetic algorithm and grid partitioning. Their population contained 100 individuals and they applied 1000 generations, with full replacement, to come up with the following result for ten independent trials: best classification rate 99.4% (1 misclassifications), average classification rate 98.5% and worst classification rate 97.8% (4 misclassifications). In both approaches the final rule base contains 60 rules. The main difference is the number of model evaluations that was necessary to come to the final result.

As can be seen from Table 2, because of the simplicity of the proposed clustering algorithm, the proposed approach is attractive in comparison with other iterative and optimization schemes that involves extensive intermediate optimization to generate fuzzy classifiers.

The results are summarized in Table 2. As it is shown, the performance of the obtained classifiers are comparable to those in [45] and [20], but use far less rules (3-5 compared to 60) and less features.

Comparing the fuzzy sets in Fig. 5 with the data in Fig. 3 shows that the obtained rules are highly interpretable. For example, the Flavonoids are divided in Low, Medium and High, which is clearly visible in the data. This knowledge can be easily validated by analyzing the SOM of the data given in Fig. 6.

7 Conclusion

The design of rule base classifiers is approached by combining a wide range of CI tools developed for knowledge representation (fuzzy rules), feature selection (class separability criterion), model initialization (clustering and decision tree), model reduction (orthogonal methods) and tuning (genetic algorithm). It has been shown that these tools can be applied in a synergistic manner though the nine steps of knowledge discovery.

References

[1] U. Fayyad, G. Piatestku-Shapio, P. Smyth, R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996.

[2] J. Bezdek, Computational intelligence definedŰby everyone!, in: Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, O. Kaynak et al. (Eds.), Springer Verlag, Germany, 1996.

[3] L. Zadeh, Roles of soft computing and fuzzy logic in the conception, design and deployment of information/intelligent systems, in: Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, O. Kaynak et al. (Eds.), Springer Verlag, Germany, pp. 1-9, 1998.

[4] A. Abraham, Intelligent systems: Architectures and perspectives, Recent Advances in Intelligent Paradigms and Applications, Abraham A., Jain L. and Kacprzyk J. (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, ISBN 3790815381, Chapter 1, pp. 1-35 .

[5] R. Brachman, T. Anand, The process of knowledge discovery in databases, in: Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1994, pp. 37–58.

[6] U. Fayyad, G. Piatestku-Shapio, P. Smyth, Knowledge discovery and data mining: Towards a unifying framework, in: Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1994.

[7] J. Abonyi, R. Babuska, H. Verbruggen, F. Szeifert, Using a priori knowledge in fuzzy model identification, International Journal of Systems Science 31 (2000) 657–667.

[8] J. Abonyi, L. Nagy, F. Szeifert, Hybrid fuzzy convolution modelling and identification of chemical process systems, International Journal of Systems Science 31 (2000) 457–466.

[9] J. Abonyi, A. Bodizs, L. Nagy, F. Szeifert, Hybrid fuzzy convolution model and its application in predictive control, Chemical Engineering Research and Design 78 (2000) 597–604.

[10] J. Mao, K. Jain, Artificial neural networks for feature extraction and multivariate data projection, IEEE Trans. on Neural Networks 6(2) (1995) 296–317.

[11] S. Abe, R. Thawonmas, Y. Kobayashi, Feature selection by analyzing regions approximated by ellipsoids, IEEE Trans. on Systems, Man, and Cybernetics, Part. C 28(2) (1998) 282–287.

[12] C. A. I. Tsamardinos, A. Statnikov, Time and sample efficient discovery of markov blankets and direct causal relations, in: 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, pages 673-678, USA, 2003.

[13] A. A. S. Chebrolu, J. Thomas, Feature deduction and ensemble design of intrusion detection systems, Computers and Security <http://dx.doi.org/10.1016/j.cose.2004.09.008> .

[14] J. D. Y. Chen, B. Yang, A. Abraham, Time series forecasting using flexible neural tree model, Information Sciences <http://dx.doi.org/10.1016/j.ins.2004.10.005> .

[15] J. V. de Oliveira, Semantic constraints for membership function optimization, IEEE Trans. FS 19 (1999) 128–138.

[16] M. Setnes, R. Babuška, Fuzzy relational classifier trained by fuzzy clustering, IEEE Trans. on Systems, Man, and Cybernetics, Part. B 29 (1999) 619–625.

[17] R. Setiono, Generating concise and accurate classification rules for breast cancer diagnosis, Artificial Intelligence in Medicine 18 (2000) 205–219.

[18] A. Abraham, Meta-learning evolutionary artificial neural networks, Neurocomputing, Elsevier Science, Netherlands, Vol. 56c, pp. 1-38 .

[19] D. Nauck, R. Kruse, Obtaining interpretable fuzzy classification rules from medical data, Artificial Intelligence in Medicine 16 (1999) 149–169.

[20] H. Ishibuchi, T. Nakashima, T. Murata, Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems, IEEE Trans. SMC–B 29 (1999) 601–618.

[21] A. Abraham, Evonf: A framework for optimization of fuzzy inference systems using neural network learning and evolutionary computation, in: The 17th IEEE International Symposium on Intelligent Control, ISIC'02, IEEE Press, ISBN 0780376218, pp. 327-332, Canada, 2002.

[22] M. Setnes, J. Roubos, Rule-based modeling: Precision and transparency, IEEE Trans. FS. in press.

[23] A. Abraham, i-miner: A web usage mining framework using hierarchical intelligent systems, in: The IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'03, IEEE Press, ISBN 0780378113, pp. 1129-1134, USA, 2003.

[24] L. Sethi., Entropy nets: From decision trees to neural networks, Proc. IEEE 78 (1990) 1605–1613.

[25] I. Ivanova, M. Kubat, Initialization of neural networks by means of decision trees, Knowledge-Based Systems 8 (1995) 333–344.

[26] R. Setiono, W. Leow, On mapping decision trees and neural networks, Knowledge Based Systems 13 (1999) 95–99.

[27] J.-S. Jang, C.-T. Sun, Functional equivalence between radial basis function networks and fuzzy inference systems, IEEE Trans. NN 4 (1993) 156–159.

[28] L. T. Kóczy, D. Tikk, T. D. Gedeon, On functional equivalence of certain fuzzy controllers and rbf type approximation schemes, International Journal of Fuzzy Systems .

[29] T. Kohonen, The self-organizing map, Proceedings of the IEEE 78(9) (1990) 1464–1480.

[30] J. Abonyi, F. Szeifert, Supervised fuzzy clustering for the identification of fuzzy classifiers, Pattern Recognition Letters 24(14) (2003) 2195–2207.

[31] M. Setnes, R. Babuška, U. Kaymak, H. van Nauta Lemke, Similarity measures in fuzzy rule base simplification, IEEE Trans. SMC-B 28 (1998) 376–386.

[32] M. Setnes, J. Roubos, Transparent fuzzy modeling using fuzzy clustering and GA's, in: In NAFIPS, New York, USA, 1999, pp. 198–202.

[33] J. Roubos, M. Setnes, Compact fuzzy models through complexity reduction and evolutionary optimization, in: Proc. of IEEE international conference on fuzzy systems, San Antonio, USA, 2000, pp. 762–767.

[34] Y. Yam, P. Baranyi, C. Yang, Reduction of fuzzy rule base via singular value decomposition, IEEE Trans. Fuzzy Systems 7(2) (1999) 120–132.

[35] J. Yen, L. Wang., Simplifying fuzzy rule-based models using orthogonal transformation methods, IEEE Trans. SMC-B 29 (1999) 13–24.

[36] M. Setnes, R. Babuška, Rule base reduction: Some comments on the use of orthogonal transforms, IEEE Trans. on Systems, Man, and Cybernetics, Part. B 31 (2001) 199–206.

[37] K. Cios, W. Pedrycz, R. Swiniarski, Data Mining Methods for Knowledge Discovery, Kluwer Academic Press, Boston, 1998.

[38] J. Roubos, M. Setnes, J. Abonyi, Learning fuzzy classification rules from labeled data, International Journal of Information Sciences 150(1-2) (2003) 612–621.

[39] Y. Jin, Fuzzy modeling of high-dimensional systems, IEEE Trans. FS 8 (2000) 212–221.

[40] J. Abonyi, R. Babuška, F. Szeifert, Fuzzy modeling with multidimensional membership functions: Grey-box identification and control design, IEEE Trans. on Systems, Man, and Cybernetics, Part. B (2002) 612–621.

[41] J. Abonyi, H. Roubos, F. Szeifert, Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision tree initialization, International Journal of Approximate Reasoning Jan (2003) 1–21.

[42] F. Hoppner, F. Klawonn, R. Kruse, T. Runkler, Fuzzy Cluster Analysis - Methods for Classification, Data Analysis and Image Recognition, John Wiley and Sons, 1999.

[43] O. Nelles, M. Fischer, Local linear model trees (LOLIMOT) for nonlinear system identification of a cooling blast, in: European Congress on Intelligent Techniques and Soft Computing (EUFIT), Aachen, Germany, 1996.

[44] J.-S. Jang, Structure determination in fuzzy modeling: A fuzzy cart approach, in: Proc. of IEEE international conference on fuzzy systems, Orlando, USA, 1994.

[45] A. Corcoran, S. Sen, Using real-valued genetic algorithms to evolve rule sets for classification, in: IEEE-CEC, Orlando, USA, 1994, pp. 120–124.

Table 1: Three rule fuzzy classifier (L=low, M=medium , H=high).

	1 Alc	2 Mal	3 Ash	4 aAsh	5 Mag	6 Tot	7 Fla	8 nFlav	9 Pro	10 Col	11 Hue	12 OD2	13 Pro	Class
R_1	H	-	-	-	-	-	H	-	-	M	L	-	L	1
R_2	L	-	-	-	-	-	-	-	-	L	L	-	H	2
R_3	H	-	-	-	-	-	L	-	-	H	H	-	H	3

Table 2: Classification rates on the Wine data for ten independent runs.

Method	Best result	Aver result	Worst result	Rules	Model eval
Corcoran and Sen [45]	100%	99.5%	98.3%	60	150000
Ishibuchi et al. [20]	99.4%	98.5%	97.8%	60	6000
Cluster + GA	99.4 %	varying schemes	98.3%	3	4000-8000
Gath-Geva clustering	95.5 %	95.5 %	95.5 %	3	1
Sup. cluster (13 features)	98.9 %	98.9 %	98.9 %	3	1
Sup. cluster (5 features)	100 %	100 %	100 %	5	2

A Survey of Forecasting Preprocessing Techniques using RNs

J.M. Górriz and J.C. Segura-Luna
Dpt. Signal Theory Daniel Saucedo s/n E-18071
gorriz@ugr.es web: hal.ugr.es

C.G. Puntonet and M. Salmerón
Dpt. Architecture and Computer Tech. Daniel Saucedo E-18071
carlos@atc.ugr.es web: atc.ugr.es

Keywords: Regularization Networks, Independent Component Analysis, Principal Component Analysis

Received: October 28, 2004

In this paper we make a survey of various preprocessing techniques including the statistical method for volatile time series forecasting using Regularization Networks (RNs). These methods improve the performance of Regularization Networks i.e. using Independent Component Analysis (ICA) algorithms and filtering as preprocessing tools. The preprocessed data is introduced into a Regularized Artificial Neural Network (ANN) based on radial basis functions (RBFs) and the prediction results are compared with the ones we get without these preprocessing tools, with the high computational effort method based on multidimensional regularization networks (MRN) and with the Principal Component Analysis (PCA) technique.

Povzetek: Predstavljene so razne metode predprocesiranja podatkov za analizo časovnih vrst.

1 Introduction

In the history of research of the forecasting problem one can extract various relevant periods such as the following mentioned: a possible solution to this problem was described by Box and Jenkins [1], who developed a time-series forecasting analysis technique based on linear systems. Basically the procedure consisted of suppressing the non-seasonality of the series, performing parameter analysis, which measures time-series correlation, and selecting the model that best fits the data set (a specific order ARIMA model). But in real systems, non-linear and stochastic phenomena crop up, and then time series dynamics cannot be described exactly using classical models. ANNs have improved results in forecasting by detecting the non-linear nature of the data. ANNs based on RBFs allow a better forecasting adjustment; they implement local approximations to non-linear functions, minimizing the mean square error to achieve the adjustment of neural parameters. For example, Platt's algorithm [2], Resource Allocating Network (RAN), consisted of neural network size control, reducing the computational time cost associated with computing the optimum weights in perceptron networks.

Matrix decomposition techniques have been used as an improvement on Platt's model [3]. For example, Singular Value Decomposition (SVD) with pivoting QR decomposition selects the most relevant data in the input space avoiding non-relevant information processing (NAPA-PRED "Neural model with Automatic Parameter Adjustment for PREDiction"). NAPA-PRED also includes neural pruning [4]. An improved version of this algorithm can be found in [5] based on Support Vector Machine philosophy.

The next step was to include exogenous information in these models. There are some choices in order to do that; we can use the forecasting model used in [6] which gives good results but with computational time and complexity cost; Principal Component Analysis (PCA) is a well-established tool in Finance. It was already proved [3] that prediction results can be improved using the PCA technique. This method linear transform the observed signal into principal components which are uncorrelated (features), giving projections of the data in the direction of the maximum variance [7]. PCA algorithms use only second order statistical information; Finally, in [8] we can discover interesting structure in finance using the new signal-processing tool Independent Component Analysis (ICA). ICA finds statistically independent components using higher order statistical information for blind source separation ([9], [10]). This new technique may use Entropy (Bell and Sejnowski 1995, [11]), Contrast functions based on Information Theory (Comon 1994, [12]), Mutual Information (Amari, Cichocki y Yang 1996, [13]) or geometric considerations in data distribution spaces (Carlos G. Puntonet 1994 [14], [15]), etc. Forecasting and analyzing financial time series using ICA can contributes to a better understanding and prediction of financial markets ([6],[8]).

There exist numerous forecasting applications in time series forecasting, as analyzed in [16]: signal statistical preprocessing and communications, industrial control processing, econometrics, meteorology, physics, biology, medicine, oceanography, seismology, astronomy and psychology. We organize the essay as follows. In section 2 we

describe the neural model used and the certain conditions to achieve a good confidence interval in prediction. In sections 3,4 and 5 we describe in detail three methods for time series preprocessing showing some results and finally in section we describe a brand new experimental framework comparing the previous discussed methods stating some conclusions.

2 Regularization networks based on RBFs

Because of their inherent non-linear processing and learning capabilities, ANNs (Artificial Neural Networks) have been proposed to solve prediction problems. An excellent survey of NN forecasting applications is to be found in [17]. There it is claimed that neural nets often offer better performance, especially for difficult time series than are hard to deal with classical models such as ARIMA models [1]. One of the simplest, but also most powerful, ANN models is the Radial Basis Function (RBF) network model. This consists of locally-receptive activation functions (or neurons) implemented by means of gaussian functions [18]. In mathematical terms, we have

$$o(\mathbf{x}) = \sum_{i=1}^{N} o_i(\mathbf{x}) = \sum_{i=1}^{N} h_i \exp\left\{-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{\sigma_i^2}\right\} \quad (1)$$

where N denotes the number of nodes (RBFs) used; $o_i(\mathbf{x})$ gives the output computed by the i-th RBF for the input vector $\mathbf{x}$ as an exponential transformation over the norm that measures the distance between $\mathbf{x}$ and the RBF center $\mathbf{c}_i$, whereas σ_i denotes the *radius* that controls the locality degree of the corresponding i-th gaussian response. The global output $o(\mathbf{x})$ of the neural network is, as can be seen, a linear aggregate or combination of the individual outputs, weighted by the real coefficients h_i.

In most RBF network applications, the coefficients h_i are determined after setting up the location and radius for each of the N nodes. The locations can be set, as in [18], by a *clustering* algorithm, such as the *K-means algorithm* [19], and the radius is usually set after taking into account considerations on RBFs close to the one being configured. The adjustment of the linear expansion coefficients can be done using recursive methods for linear least squares problems [20, 21] or the new method based on Regularization-VC Theory presented in [5] characterized by a suitable regularization term which enforces flatness in the input space, so that the actual risk functional over a training data set is minimized, and determined by the previously set parameter values [22]. Recursive specification allows for real-time implementations, but the questions arises of whether or not we are using a simplified-enough neural network, and this is a question we will try to address using matrix techniques over the data processed by the neural net.

In the particular context of the RBF networks, the mapping of a time series prediction problem to the network is performed setting up the input as past values (consecutive ones, in a first approximation) of the time series. The output is viewed as a prediction for the future value that we want to estimate, and the computed error between the desired and network- estimated value is used to adjust parameters in the network.

2.1 Regularization Theory (RT)

RT appeared in the methods for solving *ill posed problems* [23]. In RT we minimize a expression similar to the one in Support Vector Machines scenario (SVM). However, the search criterium is enforcing smoothness (instead of flatness) for the function in input space (instead of feature space). Thus we get:

$$R_{reg}[f] = R_{emp}[f] + \frac{\lambda}{2}||\hat{P}f||^2. \quad (2)$$

where $\hat{P}$ denotes a regularization operator in the sense of [23], mapping from the Hilbert Space H of functions to a dot product Space D such as $\langle f, g\rangle \quad \forall f, g \in H$ is well defined. Applying Fréchett's differential[1] to equation 2 and the concept of Greent's function of $\hat{P}^*\hat{P}$:

$$\hat{P}^*\hat{P} \cdot G(x_i, x_j) = \delta(x_i - x_j). \quad (3)$$

(here δ denotes the Diract's δ, that is $\langle f, \delta(x_i)\rangle = f(x_i)$), we get [22]:

$$f(x) = \lambda \sum_{i=1}^{\ell} [y_i - f(x_i)]_\epsilon \cdot G(x, x_i). \quad (4)$$

The correspondence between SVM and RN is proved if and only if the Greent's function G is an "admissible" kernel in the terms of Mercert's theorem [24],i.e. we can write G as:

$$G(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j)\rangle \quad (5)$$

$$with \quad \Phi : x_i \rightarrow (\hat{P}G)(x_i, .). \quad (6)$$

Prove: Minimizing $||\mathbf{P}f||^2$ can be expressed as:

$$||\mathbf{P}f||^2 = \int dx (\mathbf{P}f)^2 = \int dx f(x) \mathbf{P}^*\mathbf{P}f(x) \quad (7)$$

we can expand f in terms of green's function associated to $\mathbf{P}$, thus we get:

$$\begin{aligned} &||\mathbf{P}f||^2 = \textstyle\sum_{i,j}^{N} h_i h_j \int dx G(x, x_i) \mathbf{P}^*\mathbf{P} G(x, x_j) \\ &= \textstyle\sum_{i,j}^{N} h_i h_j \int dx G(x, x_i) \delta(x - x_j) \\ &= \textstyle\sum_{i,j}^{N} h_i h_j G(x_j, x_i) \end{aligned} \quad (8)$$

then only if G is a Mercer Kernel it correspond to a dot product in some feature space. Then minimizing 2 is equivalent to SVM minimization†.

[1] Generalized differentiation of a function: $dR[f] = \left[\frac{d}{d\rho} R[f + \rho h]\right]$, where $h \in H$.

A similar prove of this connection can be found in [25]. Hence given a regularization operator, we can find an admissible kernel such that SV machine using it will enforce flatness in feature space and minimize the equation 2. Moreover, given a SV kernel we can find a regularization operator such that the SVM can be seen as a RN.

2.2 On-line Endogenous Learning Machine Using Regularization Operators

In this section we show on-line RN based on "Resource Allocating Network" algorithms (RAN) [2] [2] which consist of a network using RBFs, a strategy for allocating new units (RBFs), using two part novelty condition [2]; input space selection and neural pruning using matrix decompositions such as SVD and QR with pivoting [4]; and a learning rule based on SRM as discussed in the previous sections. The pseudo-code of the new on-line algorithm is presented in [26]. Our network has 1 layer as is stated in equation 1. In terms of RBFs the latter equation can be expressed as:

$$f(x) = \sum_{i=1}^{N(t)} h_i \cdot \exp\left(-\frac{||x(t) - x_i(t)||^2}{2\sigma_i^2(t)}\right) + b. \quad (9)$$

where $N(t)$ is the number of neurons, $x_i(t)$ is the center of neurons and $\sigma_i(t)$ the radius of neurons, at time "t".

In order to minimize equation 2 we propose a regularization operator based on SVM philosophy. We enforce flatness in feature space, as described in [26], using the regularization operator $||\hat{P}f||^2 \equiv ||\omega||^2$, thus we get:

$$R_{reg}[f] = R_{emp}[f] + \frac{\lambda}{2} \sum_{i,j=1}^{N(t)} h_i h_j k(x_i, x_j). \quad (10)$$

We assume that $R_{emp} = (y - f(x))^2$ we minimize equation 10 adjusting the centers and radius (gradient descend method $\Delta\chi = -\eta\frac{\partial R[f]}{\partial \chi}$, with simulated annealing [27]):

$$\begin{array}{c} \Delta x_i = -2\frac{\eta}{\sigma_i}(x - x_i)h_i(f(x) - y)k(x, x_i) \\ +\alpha \sum_{i,j=1}^{N(t)} h_i h_j k(x_i, x_j)(x_i - x_j). \end{array} \quad (11)$$

and

$$\Delta h_i = \tilde{\alpha}(t) f(x_i) - \eta(f(x) - y)k(x, x_i). \quad (12)$$

where $\alpha(t), \tilde{\alpha}(t)$ are scalar-valued "adaptation gain", related to a similar gain used in the stochastic approximation processes, as in these methods, it should decrease in time. The second summand in equation 11 can be evaluated in several regions inspired by the so called "divide-and-conquer" principle and used in unsupervised learning, i.e. competitive learning in self organizing maps [28] or in SVMs experts [29]. This is necessary because of volatile nature of time series, i.e. stock returns, switch their dynamics among different regions, leading to gradual changes in the dependency between the input and output variables [26]. Thus the super-index in the latter equation is redefined as:

$$N_c(t) = \{s_i(t) : ||x(t) - x_i(t)|| \leq \rho\}. \quad (13)$$

that is the set of neurons close to the current input.

[2]The principal feature of these algorithms is sequential adaptation of neural resources.

3 RNs and PCA

3.1 Introduction

PCA is probably the oldest and most popular technique in multivariate data analysis. It transforms the data space into a feature space, in such a way, that the new data space is represented by a reduced number of "effective" features. Its main advantages lie in the low computational effort and the algebraic procedure.

Given a $n \times N$ data set $\mathbf{x}$,where N is the sample size, PCA tries to find a linear transformation $\tilde{\mathbf{x}} = \mathbf{W}^T\mathbf{x}$ into a new orthogonal basis $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_m\} \quad m \leq n$ such that:

$$Cov(\tilde{\mathbf{x}}) = E\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\} = \mathbf{W}^T Cov(\mathbf{x})\mathbf{W} = \mathbf{\Lambda} \quad (14)$$

where $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix. Hence PCA, decorrelates the vector $\mathbf{x}$ as all off-diagonal elements in the covariance matrix of the transformed vector vanish. In addition to the transformation presented in equation 14 (Karhunen-Loeve transformation when $m = n$) the variances of the transformed vectors $\tilde{\mathbf{x}}$ can be normalized to one using:

$$\tilde{\mathbf{x}} = \mathbf{W}_z^T\mathbf{x} \quad (15)$$

with the sphering matrix $\mathbf{W}_z = \left[\frac{\mathbf{w_1}}{\sqrt{\lambda_1}}, \ldots, \frac{\mathbf{w_m}}{\sqrt{\lambda_m}}\right]$ It has been shown that prediction results can be improved using this technique in [4].

In this Section we give an overview of the basic ideas underlying Principal Component Analysis (PCA) and its application to improve forecasting results using the algorithm presented in Section 2. The improvement consist on including exogenous information as is shown [3] and extracting results from this technique to complete the different methods of inclusion extra information.

The purpose of this Section is twofold. It should serve as a self-contained introduction to PCA and its relation with ANNs (Section 3.2). On the other hand, in Section 3.3, we discuss the use of this tool with the algorithm presented in Section 2 to get better results in prediction. To this end we follow the method proposed in [3] and see the disadvantages of using it.

3.2 Basis PCA and Applications

There are numerous forecasting applications in which interesting relations between variables are studied. The nature of this dependence among observations of a time series is of considerable practical interest and researchers have

to analyze this dependence to find out which variables are most relevant in practical problems. Thus, our objective is to obtain a forecast function of a time series from current and past values of relevant exogenous variables.

Some tools have been developed in physics and engineering areas which allow this kind of analysis. For example, Factor Analysis (FA) [30] is useful to extract relevant combinations (i.e factors) from the set of original variables. The extracted factors, obtained from an input linear model, are rotated to find out interesting structures in data. The procedure is based on correlation matrix between variables [3].

FA has strong restrictions on the nature of data (linear models), thus PCA is more useful to our application due to the low computational effort and the algebraic procedure as it is shown in the next Section. Reducing input space dimension (feature space) using PCA is of vital importance when working with large data set or with "on line" applications (i.e time series forecasting). The key idea in PCA, as we say latter, is *transforming* the set of correlated input space variables into a lower dimension set of new uncorrelated *features*. This is an advantage in physics and engineering fields where theret's a high computational speed demand in on-line systems (such as sequential time series forecasting).

In addition to these traditional applications in physics and engineering, this technique has been applied to economy, psychology, and social sciences in general. However, owing to different reasons [31], PCA has not been established in these fields as good as the others. In some fields PCA became popular , i.e. statistics or data mining using intelligent computational techniques [32]. Obviously, the new research in neural networks and statistical learning theory will bring applications in which PCA will be applied to reduce dimensionality or real-time series analysis.

3.2.1 PCA Operation

Let $\mathbf{x} \in \mathcal{R}^n$ representing a stochastic process. The target in PCA [33] is to find a unitary vector basis (norm equal to 1)

$$\{\mathbf{u}_j : j = 1, 2, \ldots, r\} \ , \tag{16}$$

where $r < n$, and with projections of this kind:

$$\mathbf{u}_j^T \cdot \mathbf{x} \tag{17}$$

have *maximum expected variance*, w.r.t all possible configurations (16). In other words, the first vector belonging to this basis, $\mathbf{u}_1$, must have the following property: $\mathbf{u}_1 \cdot \mathbf{x}$, considered as a random variable (since $\mathbf{x}$ is a random variable), has maximum variance between all possible linear combinations of the components of $\mathbf{x}$. At the same time, $\mathbf{u}_2$ is such that $\mathbf{u}_2 \cdot \mathbf{x}$ has maximum variance between all possible orthogonal directions to $\mathbf{u}_1$, and so on. The next unitary vectors are selected from the set of vector $\{\mathbf{w}\}$ satisfying:

$$\mathbf{w}^T \cdot \mathbf{u}_k = 0, k = 1, \ldots, j-1 \ \ and \ \ \mathbf{w}^T \cdot \mathbf{w} = 1 \ , \tag{18}$$

and then from this set $\{\mathbf{w}\}$, we choose them using

$$\mathbf{u}_j = \arg\max_{\mathbf{w}} E\left(Var[\mathbf{w}^T \cdot \mathbf{x}]\right) \ , \tag{19}$$

where $\mathbf{w}$ verifies (18).

Let a vector $\mathbf{x} \in \mathcal{R}^n$, the set of orthogonal projections with maximum variances $\mathbf{u}_j^T \cdot \mathbf{x}$ are given by:

$$\max_{\mathbf{u}_j} E\left(Var[\mathbf{u}_j^T \cdot \mathbf{x}]\right) = \lambda_j \ , \tag{20}$$

where λ_j is the j-th *eigenvalue* of the covariance matrix

$$\mathbf{R} \equiv E\left[(\mathbf{x} - \mu_{\mathbf{x}}) \cdot (\mathbf{x} - \mu_{\mathbf{x}})^T\right] \ , \tag{21}$$

that is a $n \times n$ square matrix. In the equation (21), $\mu_{\mathbf{x}}$ represents the stationary stochastic process mean which can be calculated using the set of samples $\mathbf{x}$. the eigenvalues λ_j can be calculated according the EIGD of matrix (21), which definition and properties are shown in [34].

Furthermore, it can be proved that the "principal components" from which we can get the maximum variances, are the eigenvectors of the covariance matrix $\mathbf{R}$. Note that $\mathbf{R}$ is semi-definite positive matrix thus all eigenvalues are positive real numbers including 0 $[0, +\infty)$ and their eigenvectors $\mathbf{u}_j$ satisfying:

$$\mathbf{R} \cdot \mathbf{u}_j = \lambda_j \cdot \mathbf{u}_j \ , \tag{22}$$

where λ_j denotes the associated eigenvalue, can be merged to compose an orthogonal matrix.

Hence we can estimate the covariance matrix $\mathbf{R}$ as:

$$\hat{\mathbf{R}} \equiv 1/(N-1) \cdot \mathbf{X} \cdot \mathbf{X}^T \ , \tag{23}$$

where

$$\mathbf{X} = [\mathbf{x}_1 - \bar{\mathbf{x}}, \mathbf{x}_2 - \bar{\mathbf{x}}, \ldots, \mathbf{x}_N - \bar{\mathbf{x}}] \tag{24}$$

is a $n \times N$ matrix including the set of N samples (or n-dimensional vectors) $\mathbf{x}_i$, with mean equal to $\bar{\mathbf{x}}$. Once the estimation of $\hat{\mathbf{R}}$ has been got, we can use EIGD, to obtain the following matrix:

$$\mathbf{U} = [\mathbf{u}_1 \mathbf{u}_2 \ldots \mathbf{u}_n] \tag{25}$$

including all eigenvectors in the columns, and the diagonal matrix Λ with the corresponding eigenvalues in the main diagonal.

Obviously if we choose the set of orthogonal and unitary vectors given by the equation (25), then theret's an unique

[3] Factor analysis is a statistical approach that can be used to analyze interrelationships among a large number of variables and to explain these variables in terms of their common underlying dimensions ("factors"). The statistical approach involving finding a way of condensing the information contained in a number of original variables into a smaller set of dimensions (factors) with a minimum loss of information. It has been used in disciplines as diverse as chemistry, sociology, economics or psychology.

correspondence between the input space matrix $\mathbf{X}$ in equation (24) and the $n \times N$ matrix $\mathbf{X}' = \mathbf{U}^T \cdot \mathbf{X}$. This transformation is invertible since $\mathbf{U}^{-1} = \mathbf{U}$, i.e $\mathbf{U}$ is orthogonal. All the process is based on a simple linear transformation into a new orthogonal basis such that the covariance matrix in the new system is diagonal.

3.3 Time Series prediction with PCA

In this Section we show how Principal Component Analysis (PCA) technique can be hybridized with the algorithm presented in section 2 to improve prediction results, including exogenous information.

3.3.1 Data compression and reducing dimensionality

As we mentioned latter, PCA is a useful tool in the preprocessing step in data analysis, i.e. those techniques based on artificial neural networks considered in this work. In this way, there can be a PCA layer that compresses raw data from the sample set.

The basic idea is consider a large set of input variables and transforms it to a new set of variables containing without loss of much information [7]. This is possible due to that very often, multivariate data contains redundant information of 2^{nd} order.

On the other hand, the more dimension reduction we want to achieve the more fraction of original loss variance, in other words, we can lose much relevant information. Thus, under this conditions, the inclusion of exogenous information would contaminate prediction capacity of any system (neural or not). Hence, if PCA extract only the first r factors (in terms of variance) such that:

$$r = \min\{k : \sum_{i=1}^{k} \lambda_i \geq \rho \cdot tr(\hat{\mathbf{R}})\} \ , \qquad (26)$$

only a fraction of ρ of the exogenous data overall variance will be kept. In the equation (26), $tr(\hat{\mathbf{R}})$ denotes *trace* of the estimated covariance matrix $\hat{\mathbf{R}}$ for the set of data (that is, adding its diagonal elements or individual variance components). Given that $\sum_{i=1}^{n} \lambda_i$ is the overall variance and PCA projection variances are given by the eigenvalues λ_i, if we consider the complete set of eigenvalues we would have the complete variance, so this way, if we select a subset of eigenvalues $r < n$, we would hold a fraction ρ (at least) of the overall variance of the exogenous data.

In this method of data compression using the maximum variance principle, PCA is basically regarded as a standard statistical technique. In neural networks research areas, the term *unsupervised Hebbian learning* [35, 36, 37] is usually used to refer this powerful tool in data analysis, such as *discriminant analysis* is used as a theoretical foundation to justify neural architectures (i.e. multilayer perceptrons or linear architectures) for classification [38].

The previous discussion explains the concept of *dimension reduction*, projecting onto r more relevant unitary vectors (that is, those ones that hold the bigger fraction of variance of data) is the way of develop this reduction since multiplying by $\mathbf{U}$ gives a $r \times N$ matrix. In addition, the column vectors in $\mathbf{U}$ can be seen as *feature vectors*, containing the principal characteristics of the data set; the projection onto $\mathbf{u}_j$ must be understood, in that case, such as a measure of certain characteristic in data samples. The transformation onto this new feature space is suitable way to analyse raw data, thus, in this Section, we will use this technique in the preprocessing step, before neural stages.

3.3.2 Improving neural input space

Moreover, PCA can be used to include exogenous information [3], in other words, we can increase *input space dimension* using variables related to the original series. The principal advantage of using PCA over straightforward inclusion is that PCA can reduce input space dimensionality without loss of much information (in terms of variance) included is such variables.

As we said latter, to reduce input space dimensionality using PCA, a fraction of information, i.e variance, must be rejected. In some cases, it can be a decision with unforeseeable consequences, thus, a conservative policy should be followed, i.e. using PCA variables such as "extra" variables to improve the prediction results. This is the key idea in this Section.

This rule based on "catalytic variables" is appropriate in a practical point of view. In fact in [7], hybridized with filtering techniques, is a success. In the following example, we show how this technique is applied to stock series obtaining noticeable improvements.

3.4 Results

In the following Section we show an example in which we applied hybrid models based on PCA and ANN originally discussed in [3]. We intend to forecast (with horizon equal to 1) stock series (indexes) of different Spanish banks and other companies during the same period.

3.4.1 Description and data set

We have specifically focussed on the IBEX35 index of Spanish stock, which we consider the most representative sample of Spanish stock movements. We have chosen seven relevant indexes such as *Banesto*, *Bankinter*, *BBVA*, *Pastor*, *Popular*, *SCH* y *Zaragozano*, and we build a matrix including 1672 consecutive observations (closing prices). A representation of theses indexes can be found in the figure 1.

Itt's clear, from the latter figure, that there is a wide range of indexes closing prices. Thus, it means the need for a logarithmic transformation (to make variance steady) and later suitable differentiation of the data (to remove the residual non-stationary behavior). Once the proper transformations are achieved we obtain the results shown in figure 2.

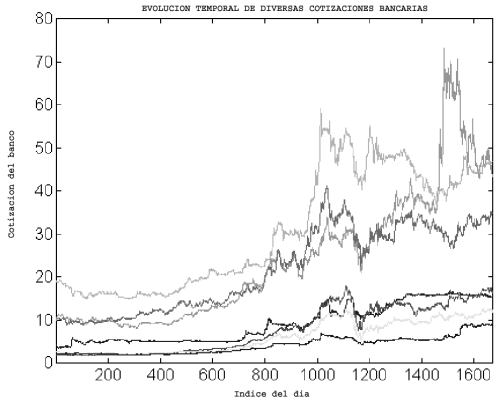


Figure 1: Closing prices evolution for selected indexes.

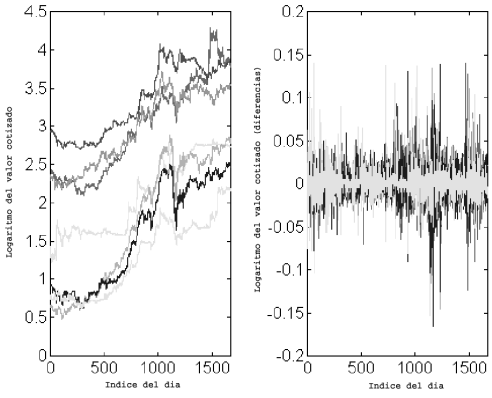


Figure 2: Closing prices evolution for selected indexes, after logarithmic transformation (on the left) and differentiation (on the right).

Table 1: Eigenvalues λ_i and variance percentages of PCs.

Index	*Eigenvalue*	*Pct. variance*	*Pct. overall*
1	0.9302	33.03	33.03
2	0.6670	23.69	56.72
3	0.3303	11.73	68.45
4	0.2694	9.57	78.02
5	0.2198	7.80	85.82
6	0.2134	7.58	93.40
7	0.1858	6.60	100.00

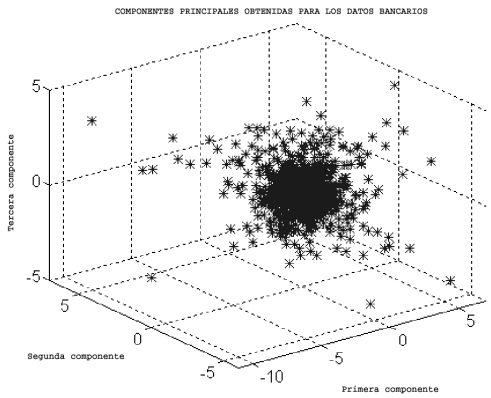


Figure 3: 3D schematic representation of the three first principal components.

After these basic transformations, the new set of series can be processed using the algorithm introduced in Section 2. The object of the method is to train our neural network based on RBFs with the set of transformed series, to predict (with horizon equal to 1) the first of the selected stocks (strictly speaking, we predict the transformed value that must be inverted in the final step) using its own endogenous information (the number of lags were fixed to 2) and the more relevant principal components (thus we don't use the other series directly).

Using a training set consisting of 1000 samples, we computed the first 3 principal components. The matrix used consist of the complete set of series (every stock). As we mentioned in Section 3.3.2, we determinate the number of inputs to improve the prediction result of interest using the principal components as additional data input. We remark that the 3 components represents about 70% of the overall variance (table 1). In a three dimensional space we can plot the components as is shown in figure 3.

Finally, the last 10 samples of the complete set (1000) were used to compare prediction results with and without exogenous inputs. In addition, in this example we included the well-known sphering or y-score transformation [33](as the variance along all principal components equals one):

$$\mathbf{w}_i = \sqrt{\lambda_i}^{-1} \cdot \mathbf{u}_i \ , \tag{27}$$

instead of using the original eigenvector $\mathbf{u}_i$. This trans-

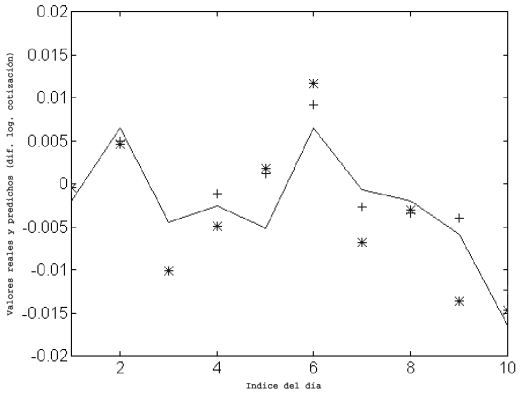


Figure 4: Prediction results with and without exogenous information using NAPA-PRED. The solid line is the real time series, asterisks (*) are results using NAPA-PRED, and crosses (+) are results using NAPA-PRED+PCA

Table 2: NRMSE for last 10 points (normalized round mean square error).

Mode	*NRMSE Error*
With PCA	0.7192
Without PCA	0.5189

formation is very common in the field of artificial neural networks and in many ICA-algorithms (whitening) in a preprocessing step as it completely removes all correlations up to the 2^{nd} order.

The reason for using just the last 10 sample points lies in the fact that economic series extremely volatile and PCA can only extract up to second order relations (in Section 4 we use a better tool to develop it). So the extra information extracted using this technique forces redefining the preprocessing step in a few iterations. This is related to the fact that we choose the principal components instead of the original series (5 variables); this choice would increase even more the input space dimension reducing the efficiency of the model.

3.4.2 Prediction Results

We compare prediction results obtained using the algorithm in section 2, rejecting the regularization term with and without the method proposed in this Section as is shown in figure 4. Prediction results improve and it is due to fraction of exogenous information included. In table 2 we show that (after transformations are inverted) the results for these 10 point are improved in a percentage around 7%.

3.5 Conclusions

From the results in the previous Section, itt's clear that the originally proposed method in [3], improves algorithms efficiency. This increase is based on selecting a suitable input space using a statistic method (PCA) and to date, it's the only way to develop it.

However the reader can notice the problems of this method. These problems are mentioned in the introduction of the chapter and are about the order of statistics used. In addition the increasing dimensionality ("curse of dimensionality") can cause serious problems (we were using a 5 dimensional input space) damaging the quality of the result. Neural networks are very sensitive to this problem because of the number of neurons to ensure universal approximation conditions [39] grows exponentially with the input space dimension unlike multilayer perceptrons, i.e. they are global approximations of nonlinear transformations, so they have a natural capacity of generalization [22, Sec. 7.9] with limited data set.

4 RNs and ICA

In this Section we describe a method for volatile time series forecasting using Independent Component Analysis (ICA) algorithms (see [40]) and Savitzky-Golay filtering as preprocessing tools. The preprocessed data will be introduce in a based radial basis functions (RBF) Artificial Neural Network (ANN) and the prediction result will be compared with the one we get without these preprocessing tools. This method is a generalization of the classical Principal Component Analysis (PCA) method for exogenous information inclusion (see Section 3)

4.1 Basic ICA

ICA has been used as a solution of the blind source separation problem [10] denoting the process of taking a set of measured signal in a vector, $\mathbf{x}$, and extracting from them a new set of statistically independent components (ICs) in a vector $\mathbf{y}$. In the basic ICA each component of the vector $\mathbf{x}$ is a linear instantaneous mixture of independent source signals in a vector $\mathbf{s}$ with some unknown deterministic mixing coefficients:

$$x_i = \sum_{i=1}^{N} a_{ij} s_j \tag{28}$$

Due to the nature of the mixing model we are able to estimate the original sources $\tilde{s}_i$ and the unmixing weights b_{ij} applying i.e. ICA algorithms based on higher order statistics such as cumulants.

$$\tilde{s}_i = \sum_{i=1}^{N} b_{ij} x_j \tag{29}$$

Using vector-matrix notation and defining a time series vector $\mathbf{x} = (x_1, \ldots, x_n)^T$, $\mathbf{s}$, $\tilde{\mathbf{s}}$ and the matrix $\mathbf{A} = \{a_{ij}\}$ and $\mathbf{B} = \{b_{ij}\}$ we can write the overall process as:

$$\tilde{\mathbf{s}} = \mathbf{B}\mathbf{x} = \mathbf{B}\mathbf{A}\mathbf{s} = \mathbf{G}\mathbf{s} \tag{30}$$

where we define $\mathbf{G}$ as the overall transfer matrix. The estimated original sources will be, under some conditions included in Darmois-Skitovich theorem (chapter 1 in [41]), a permuted and scaled version of the original ones. Thus, in general, it is only possible to find $\mathbf{G}$ such that $\mathbf{G} = \mathbf{PD}$ where $\mathbf{P}$ is a permutation matrix and $\mathbf{D}$ is a diagonal scaling matrix.

This model (equation (28)) can be applied to the stock series where there are some underlying factors like seasonal variations or economic events that affect the stock time series simultaneously and can be assumed to be quite independent [42].

4.2 Preprocessing Time Series with ICA+Filtering

The main goal, in the preprocessing step, is to find non-volatile time series including exogenous information i.e. financial time series, easier to predict using ANNs based on RBFs. This is due to smoothed nature of the *kernel* functions used in regression over multidimensional domains [43]. We propose the following Preprocessing Steps

- After whitening the set of time series $\{x_i\}_{i=1}^n$(subtracting the mean of each time series and removing the second order statistic effect or covariance matrix diagonalization process—see Section 3 for further details)

- We apply an ICA algorithm to estimate the original sources s_i and the mixing matrix $\mathbf{A}$ in equation (28). Each IC has information of the stock set weighted by the components of the mixing matrix. In particular, we use an equivariant robust ICA algorithm based in cumulants (see [41] and [6]) however another choices can be taken instead, i.e. in [40]. The unmixing matrix is calculated according the following iteration:

$$\mathbf{B}^{(n+1)} = \mathbf{B}^{(n)} + \mu^{(n)}(\mathbf{C}_{\mathbf{s},\mathbf{s}}^{1,\beta}\mathbf{S}_{\mathbf{s}}^{\beta} - \mathbf{I})\mathbf{B}^{(n)} \qquad (31)$$

  where $\mathbf{I}$ is the identity matrix, $\mathbf{C}_{\mathbf{s},\mathbf{s}}^{1,\beta}$ is the $\beta+1$ order cumulant of the sources (we chose $\beta = 3$ in simulations) , $\mathbf{S}_{\mathbf{s}}^{\beta} = diag(sign(diag(\mathbf{C}_{\mathbf{s},\mathbf{s}}^{1,\beta})))$ and $\mu^{(n)}$ is the step size.

  Once convergence, which is related to cross-cumulants [4] absolute value, is reached, we estimate the mixing matrix inverting $\mathbf{B}$.

  Generally, the ICs obtained from the stock returns reveal the following aspects [8]:

  1. Only a few ICs contribute to most of the movements in the stock return.
  2. Large amplitude transients in th dominant ICs contribute to the major level changes. The non-dominant components do not contribute significantly to level changes.
  3. Small amplitude ICs contribute to the change in levels over short time scales, but over the whole period, there is little change in levels.

- Filtering.

  1. We neglect non-relevant components in the mixing matrix $\mathbf{A}$ according to their absolute value. We consider the rows $\mathbf{A}_i$ in matrix $\mathbf{A}$ as vectors and calculate the mean Frobenius norm [5] of each one. Only the components bigger than mean Frobenius norm will be considered. This is the principal preprocessing step using PCA tool but in this case this is not enough.

$$\tilde{\mathbf{A}} = \mathbf{Z} \cdot \mathbf{A} \qquad (32)$$

     where $\{\mathbf{Z}\}_{ij} = [\{\mathbf{A}\}_{ij} > \frac{||\mathbf{A}_i||_{Fr}}{n}]$

  2. We apply a low band pass filter to the ICs. We choose the well-adapted for data smoothing Savitsky-Golay smoothing filter [44] for two reasons: a)ours is a real-time application for which we must process a continuous data stream and wish to output filtered values at the same rate we receive raw data and b) the quantity of data to be processed is so large that we just can afford only a very small number of floating operations on each data point thus computational cost in frequency domain for high dimensional data is avoided even the modest-sized FFT (see in [40]). This filter is also called Least-Squares [45] or DISPO [46]. These filters derive from a particular formulation of the data smoothing problem in the time domain and their goal is to find filter coefficients c_n in the expression:

$$\bar{s}_i = \sum_{n=-n_L}^{n_R} c_n s_{i+n} \qquad (33)$$

     where $\{s_{i+n}\}$ represent the values for the ICs in a window of length $n_L + n_R + 1$ centered on i and $\tilde{s}_i$ is the filter output (the smoothed ICs), preserving higher moments [47].

     For each point s_i we least-squares fit a m order polynomial for all n_L+n_R+1 points in the moving window and then set $\tilde{s}_i$ to the value of that polynomial at position i. As shown in [47] there are a set of coefficients for which equation (33) accomplishes the process of polynomial least-squares fitting inside a moving window:

[4] Fourth order cumulant between each pair of sources must equals zero. This is the essential condition of statistical independence as is shown in chapter 3 in [6].

[5] Given $x \in \mathcal{R}^n$, its Frobenius norm is $||\mathbf{x}||_{Fr} \equiv \sqrt{\sum_{i=1}^n x_i^2}$

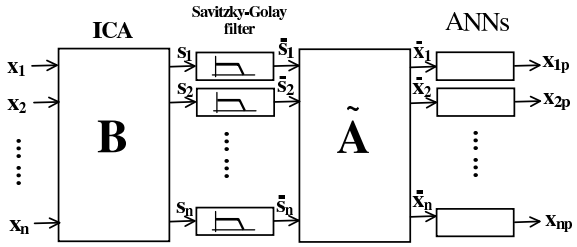


Figure 5: Schematic representation of prediction and filtering process.

$$c_n = \{(\mathbf{M}^T \cdot \mathbf{M})^{-1}(\mathbf{M}^T \cdot \mathbf{e}_n)\}_0 = \\ = \sum_{j=0}^{m}\{(\mathbf{M}^T \cdot \mathbf{M})^{-1}\}_{0j} \cdot n^j$$

where $\{\mathbf{M}\}_{ij} = i^j$, $i = -n_L, \ldots, n_R$, $j = 0, \ldots, m$, and $\mathbf{e}_n$ is the unit vector with $-n_L < n < n_R$. Note that equation (34) implies that we need only one row of the inverse matrix (numerically we can get this by LU decomposition [47], with only a single backsubstitution).

– Reconstructing the original series using the smoothed ICs and filtered $\tilde{\mathbf{A}}$ matrix we obtain a less high frequency variance version of the series including exogenous influence of the exogenous ones. We can write using equations 42 and 41.

$$\mathbf{x} = \tilde{\mathbf{A}} \cdot \bar{\mathbf{s}} \tag{34}$$

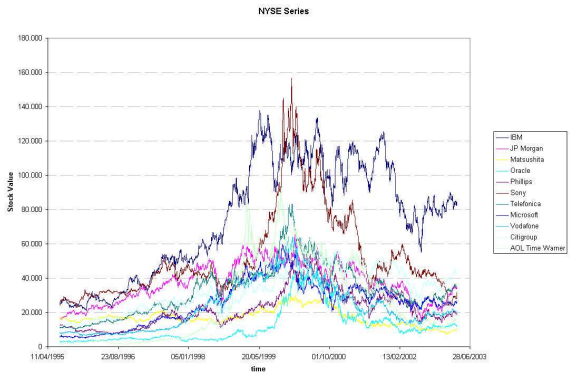


Figure 6: Set of stock series.

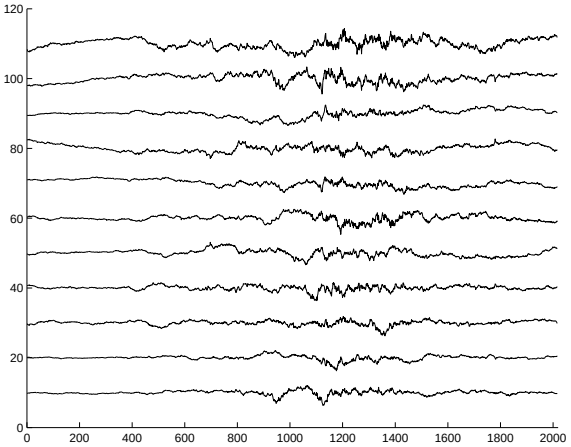


Figure 7: Set of ICs of the stock series.

4.3 Time Series Forecasting Model

We use an ANN based on RBFs to forecast a series x_i from the Stock Exchange building a forecasting function $\mathbf{P}$ with the help of the algorithm presented in Section 2, for one of the set of signals $\{x_1, \ldots, x_n\}$. As shown in Section 2 the individual forecasting function can be expressed in terms of RBFs as [48]:

$$\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{N} f_i(\mathbf{x}) = \sum_{i=1}^{N} h_i \exp\{\frac{||\mathbf{x} - \mathbf{c}_i||^2}{\mathbf{r}_i^2}\} \tag{35}$$

where $\mathbf{x}$ is a p-dimensional vector input at time $\mathbf{t}$, $\mathbf{N}$ is the number of neurons (RBFs) , $\mathbf{f}_i$ is the output for each neuron i-th , $\mathbf{c}_i$ is the centers of i-th neuron which controls the situation of local space of this cell and $\mathbf{r}_i$ is the radius of the i-th neuron. The overall output is a linear combination of the individual output for each neuron with the weight of $\mathbf{h}_i$. Thus we are using a method for moving beyond the linearity where the core idea is to augment/replace the vector input $\mathbf{x}$ with additional variables, which are transformations of $\mathbf{x}$, and then use linear models in this new space of derived input features. RBFs are one of the most popular *kernel* methods for regression over the domain $\mathcal{R}^n$ and consist on fitting a different but simple model at each query point $\mathbf{c}_i$ using those observations close to this target point in order to get a smoothed function. This localization is achieved via a weighting function or *kernel* $\mathbf{f}_i$.

The preprocessing step suggested in Section 4.2 is necessary due to the dynamics of the series (the algorithm presented Section 2 is sensitive to this preprocessed series) and it will be shown that results improve noticeably [40]. Thus we use as input series the smoothed ones obtained from equation (45).

4.4 Simulations

In the current simulation we have worked with an index of a Spanish bank (Bankinter) and other companies (such as exogenous variables) during the same period to investigate the effectiveness of ICA techniques for financial time series (figure 6). We have specifically focussed on the dowjones from american stock, which we consider the most representative sample of the american stock movements, using closing prices series.

We considered the closing prices of Bankinter for prediction and 10 indexes of intenational companies (IBM, JP Morgan, Matsushita, Oracle, Phillips, Sony, Microsoft, Vodaphone, Citigroup and Warner). Each time series includes

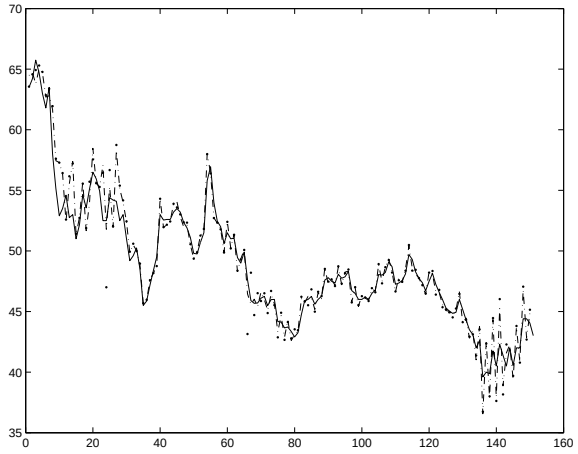


Figure 8: Real Series(line),predicted Series with ICA+SG(dash-dotted),predicted Series without preprocessing (dotted). The stock selected was Bakinter from IBEX35

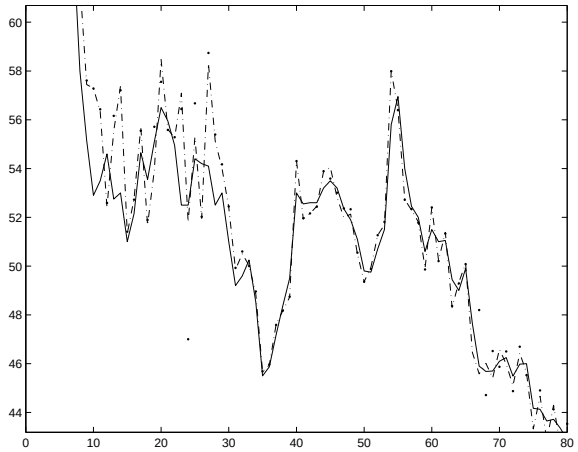


Figure 9: Zoom on figure 8.

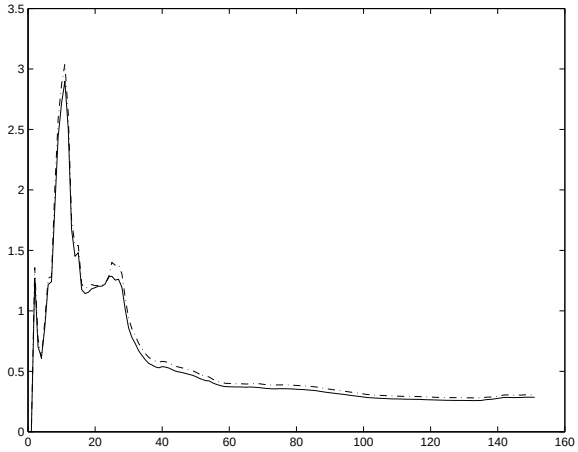


Figure 10: NRMSE evolution for ANN method and ANN+ICA method for Bankinter Series; theret's a noticeable improvement even under volatile conditions.

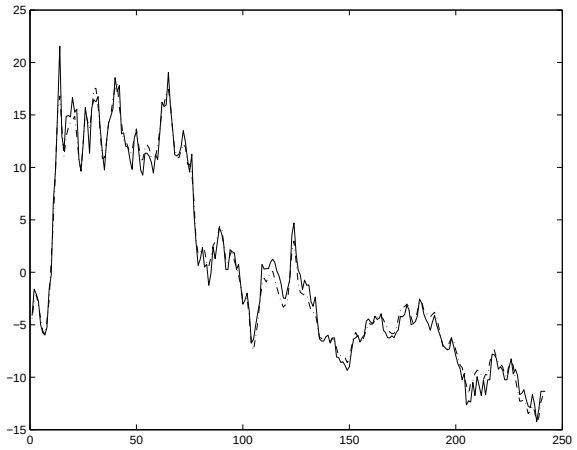


Figure 11: Real series from ICA reconstruction (scaled old version)(line) and preprocessed real series (dotted line). Selected Stock: Bankinter

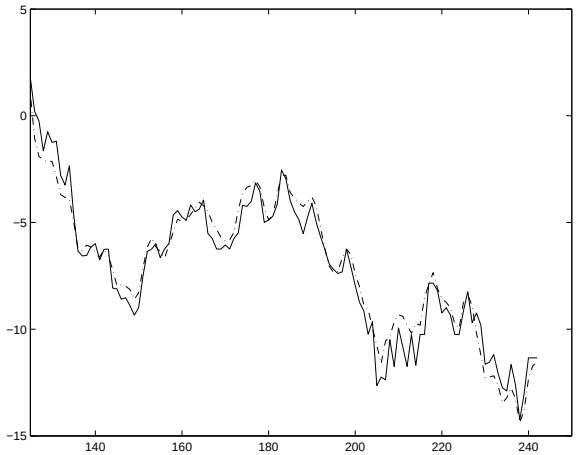


Figure 12: Zoom on figure 11.

2000 points corresponding to selling days (quoting days).

We performed ICA on the Stock returns using the ICA algorithm presented in Section 4.2 assuming that the number of stocks equals the number of sources supplied to the mixing model. This algorithm whiten the raw data as the first step. The ICs are shown in the figure 7. These ICs represents independent and different underlying factors like seasonal variations or economic events that affect the stock time series simultaneously. Via the rows of $\mathbf{A}$ we can reconstruct the original signals with the help of these ICs i.e. Bankinter stock after we preprocess the raw data:

– Frobenius Filtering: the original mixing matrix[6]:

$$\mathbf{A} = \begin{pmatrix} \ddots & \vdots & \vdots & \vdots & \vdots \\ \dots & 0.33 & -0.23 & 0.17 & 0.04 \\ \dots & \mathbf{-0.28} & \mathbf{1.95} & \mathbf{-0.33} & \mathbf{4.70} \\ \dots & 0.33 & -0.19 & 0.05 & -0.23 \\ \dots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{36}$$

is transformed to:

$$\tilde{\mathbf{A}} = \begin{pmatrix} \ddots & \vdots & \vdots & \vdots & \vdots \\ \dots & 0.33 & -0.23 & 0.17 & 0.04 \\ \dots & \mathbf{0} & \mathbf{1.95} & \mathbf{0} & \mathbf{4.70} \\ \dots & 0.33 & -0.19 & 0.05 & -0.23 \\ \dots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{37}$$

thus we neglect the influence of two ICs on the original 5^{th} stock. Thus only a few ICs contribute to most of the movements in the stock returns and each IC contributes to a level change depending its amplitude transient [8].

– We compute a polynomial fit in the ICs using the library supported by *MatLab* and the reconstruction of the selected stock (see figure 11) to supply the ANN.

In figures 8 and 9 we show the results (prediction for 150 samples) we obtained using our ANN with the latter ICA method. We can say that prediction is better with the preprocessing step avoiding the disturbing peaks or convergence problems in prediction. As is shown in figure 10, the NRMSE is always lower using the techniques we discussed in Section 4.3.

Finally, with these models we avoid the "curse of dimensionality" or difficulties associated with the feasibility of density estimation in many dimensions presented in AR or ANNs models (i.e RAN networks without input space control, see Section 2) with high number of inputs as shown in figure 14 and the delay problem presented in non relevant

[6]We show and select the relevant part of the row corresponding to the Bankinter Stock.

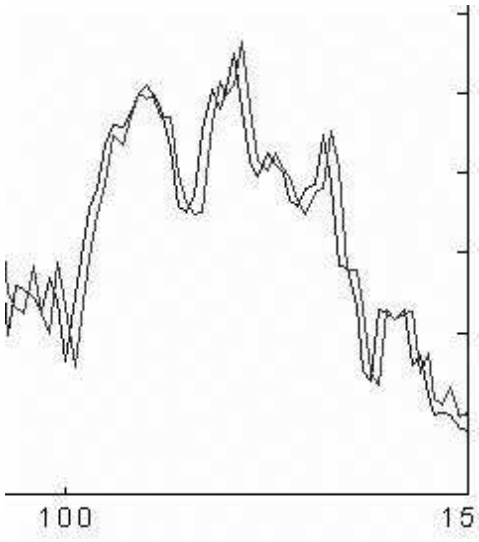


Figure 13: Delay problem in ANNs

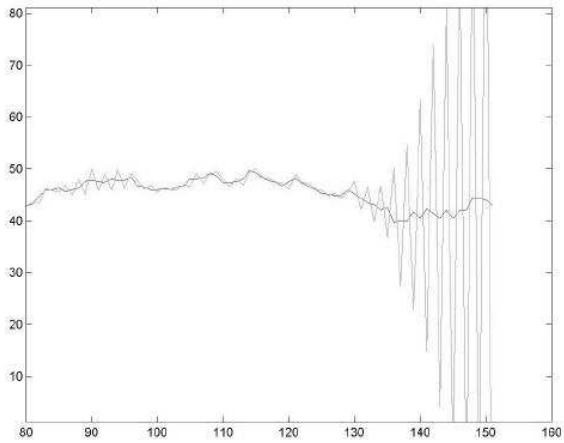


Figure 14: curse of dimensionality

time periods of prediction (bad capacity of generalization in figure 13). In addition, we improve the model presented in Section 3 in two ways:

– This method includes more relevant information (higher order statistics) in which PCA is the first step in the process.

– The way of including extra information avoids "curse of dimensionality" in any case.

4.5 Conclusions

In this Section we showed that prediction results can be improved with the help of techniques like ICA. ICA decompose a set of 11 returns from the stock into independent components which fall in two categories[40]:

1. Large components responsible of the major changes in level prices and

2. Small fluctuations responsible of undesirable fluctuations in time.

Smoothing this components and neglecting the non-relevant ones we can reconstruct a new version of the Stock easier to predict. Moreover we describe a new filtering method to volatile time series that are supplied to ANNs in real-time applications.

5 Multidimensional Regularization Networks

In this Section we propose the simplest way of including extra information(MRNs). We assume a linear model in some feature space on which the set of transformed input series will be fitted minimizing the empirical risk ("Feature Space Learning"). Various techniques can be applied to minimize this functional, i.e. we propose a genetic algorithm (GA) based on *neighbor philosophy* to speed up the convergence. This model, using the capacity of generalization of INAPA-PRED algorithm, is suitable for small sample size improving forecasting results under this condition.

5.1 Forecasting Model

The new prediction model is shown in figure 15. We consider a data set consisting of some correlated signals from the Stock Exchange and seek to build a forecasting function $\mathbf{P}$, for one of the sets of signals $\{series_1, \ldots, series_S\}$, which allows exogenous data from the other series to be included. If we consider just one series (see Section 2) the individual forecasting function can be expressed in terms of RBFs as in [48]:

$$\mathbf{F}(\mathbf{x}) = \sum_{i=1}^{N} f_i(\mathbf{x}) = \sum_{i=1}^{N} h_i \cdot \exp\left\{\frac{||\mathbf{x} - \mathbf{c}_i||^2}{\mathbf{r}_i^2}\right\} \quad (38)$$

where $\mathbf{x}$ is a p-dimensional vector input at time $\mathbf{t}$, $\mathbf{N}$ is the number of neurons (RBFs) , $\mathbf{f}_i$ is the output for each $i-th$ neuron, $\mathbf{c}_i$ is the centers of the $i-th$ neuron which controls the situation of local space of this cell and $\mathbf{r}_i$ is the radius of the i-th neuron. The overall output is a linear combination of the individual outputs for each neuron with a weight of $\mathbf{h}_i$. Thus we are using a method for moving beyond linearity in which the key idea is to augment/replace the vector input $\mathbf{x}$ with additional variables, which are transformations of $\mathbf{x}$, and then use linear models in this new space of derived input features. RBFs are one of the most popular *kernel* methods for regression over the domain $\mathcal{R}^n$ and consist of fitting a different but simple model at each query point $\mathbf{c}_i$ using the observations close to this target point in order to get a smoothed function (see previous Sections). This localization is achieved via a weighting function or *kernel* $\mathbf{f}_i$.

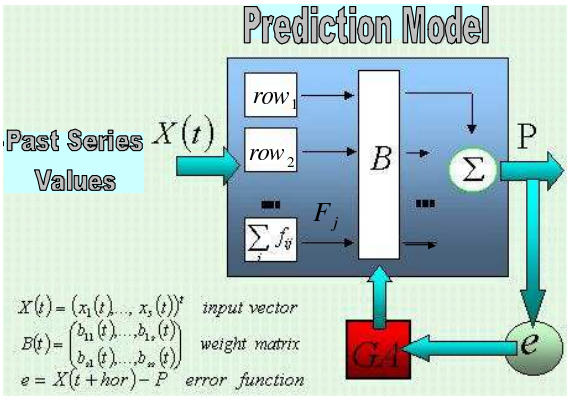


Figure 15: Schematic representation of MRN with adaptive radius, centers and input space ANNs (CPM CrossOver Prediction Model). [5]

We apply/extend this regularization concept (see Section 2, to extra series, see figure15, including a row of neurons (equation (38)) for each series, and weight these values by a factor $\mathbf{b}_{ij}$. Finally, the overall smoothed function for the stock $\mathbf{j}$ is defined as:

$$\mathbf{P}_j(\mathbf{x}) = \sum_{i=1}^{S} \mathbf{b}_{ij} F_i(x_i, j) \quad (39)$$

where $\mathbf{F}_i$ is the smoothed function of each series, $\mathbf{S}$ is the number of input series and $\mathbf{b}_{ij}$ are the weights for j-stock forecasting. Obviously one of these weight factors must be relevant in this linear fit ($\mathbf{b}_{jj} \sim 1$, or auto weight factor).

Matrix notation can be used to include the set of forecasts in an S-dimensional vector $\mathbf{P}$ ($\mathbf{B}$ in figure15):

$$\mathbf{P}(\mathbf{x}) = diag(\mathbf{B} \cdot \mathbf{F}(\mathbf{x})) \quad (40)$$

where $\mathbf{F} = (\mathbf{F}_1, \ldots, \mathbf{F}_S)$ is an $S \times S$ matrix with $F_i \in \mathcal{R}^S$ and B is an $S \times S$ weight matrix. The operator $diag$ extracts the main diagonal. Because the number of neurons and the input space dimension increases in prediction function (equation (40)), we must control them (*parsimony*) to reduce curse of dimensionality effect and overfitting.

To check this model, we choose a set of values for the weight factors as functions of correlation factors between the series, and thus equation (39) can be expressed (replacing $\mathbf{P_j}$ with $\mathbf{P}$) as:

$$\mathbf{P}(\mathbf{x}) = (1 - \sum_{i \neq j}^{S} \rho_i)\mathbf{F}_j + \sum_{i \neq j}^{S} \rho_i \mathbf{F}_i \quad (41)$$

where $\mathbf{P}$ is the forecasting function for the desired stock j and $\boldsymbol{\rho}_i$ is the correlation factor with the exogenous series $\mathbf{i}$.

We can include equation (41) in the Generalized Additive models for regression proposed in supervised learning [43]:

$$\mathbf{E}\{Y|\mathbf{X}_1, \ldots, \mathbf{X}_n\} = \alpha + \mathbf{f}_1(\mathbf{X}_1) + \ldots + \mathbf{f}_n(\mathbf{X}_n) \quad (42)$$

where $\mathbf{X}_i s$ usually represent predictors and $\mathbf{Y}$ represents the system output; $\mathbf{f}_j s$ are unspecific smooth ("nonparametric") functions. Thus we can fit this model by minimizing the mean square error function or by other methods presented in [43] (in the next Section we use a GA, a well known optimization tool, to minimize the mean square error).

5.2 Forecasting Model and Genetic Algorithms

MRN uses a GA for $\mathbf{b}_i$ parameter fitting. A GA can be modelled by means of a *time inhomogeneous Markov* chain [49] obtaining interesting properties related to weak and strong ergodicity, convergence and the distribution probability of the process (see [50]). In the latter reference, a canonical GA is constituted by operations of parameter encoding, population initialization, crossover , mutation, mate selection, population replacement, fitness scaling, etc. proving that with these simple operators a GA does not converge to a population containing only optimal members. However, there are GAs that converge to the optimum, *The Elitist GA* [51] and those which introduce *Reduction Operators* [52].

We have borrowed the notation mainly from [53] where the model for GAs is a inhomogeneous Markov chain model on probability distributions ($\mathbf{S}$) over the set of all possible populations of a fixed finite size. Let $\mathbf{C}$ the set of all possible creatures in a given world (vectors of dimension equal to the number of extra series) and a function $f : \mathbf{C} \rightarrow R^+$. The task of GAs is to find an element $c \in \mathbf{C}$ for which $f(c)$ is maximal. We encode creatures into genes and chromosomes or individuals as strings of length ℓ of binary digits (size of Alphabet *A* is $a = 2$) using one-complement representation; other encoding methods, also possible i.e [54], [55],[56] or [57], where the value of each parameter is a gene and an individual is encoded by a string of real numbers instead of binary ones.

In the Initial Population Generation step (choosing randomly $p \in \wp_N$, where $\wp_N$ is the set of populations, i.e the set of N-tuples of creatures containing $a^{L \equiv N \cdot \ell}$ elements) we assume that creatures lie in a bounded region $[0, 1]$ (at the edge of this region we can reconstruct the model without exogenous data). After the initial population p has been generated, the fitness of each chromosome $\mathbf{c}_i$ is determined via the function:

$$f(c_i) = \frac{1}{\mathbf{e}(\mathbf{c}_i)} \tag{43}$$

where $\mathbf{e}$ is an error function (i.e square error sum in a set of neural outputs, adjusting the convergence problem in the optimal solution by adding a positive constant to the denominator)

The next step in canonical GA is to define the Selection Operator. New generations for mating are selected depending on their fitness function values using *roulette wheel selection*. Let $p = (c_1, \ldots, c_N) \in \wp_N$, $n \in \mathcal{N}$ and f the fitness function acting in each component of p. Scaled fitness selection of p is a lottery for every position $1 \leq i \leq N$ in population p such that creature c_j is selected with probability:

$$\frac{f_n(p, j)}{\sum_{i=1}^{N} f_n(p, i)} \tag{44}$$

thus proportional fitness selection can be described by column stochastic matrices $\mathbf{F_n}$, $n \in \mathcal{N}$, with components:

$$\langle q, \mathbf{F_n} p \rangle = \prod_{i=1}^{N} \frac{n(q_i) f_n(p, q_i)}{\sum_{j=1}^{N} f_n(p, j)} \tag{45}$$

where $p, q \in \wp_N$ so $p_i, q_i \in \mathbf{C}$, $\langle \ldots \rangle$ denotes the standard inner product, and $n(d_i)$ the number of occurrences of q_i in p.

Once the two individuals have been selected, an elementary crossover operator $\mathbf{C}(K, P_c)$ is applied (setting the crossover rate at a value, i.e. $P_c \rightarrow 0$, which implies children similar to parent individuals) that is given (assuming N even) by:

$$\mathbf{C}(K, P_c) = \prod_{i=1}^{N/2} ((1 - P_c)\mathcal{I} + P_c \mathbf{C}(2i - 1, 2i, k_i)) \tag{46}$$

where $\mathbf{C}(2i - 1, 2i, k_i)$ denotes elementary crossover operation of c_i, c_j creatures at position $1 \leq k \leq \ell$ and $\mathcal{I}$ the identity matrix, to generate two offspring (see [50] for details of the crossover operator).

The Mutation Operator $\mathbf{M_{P_m}}$ is applied (with probability $\mathbf{P}_m$) independently at each bit in a population $p \in \wp_N$, to avoid premature convergence (see [54] for further discussion). The multi-bit mutation operator with change probability following a *simulated annealing* law with respect to the position $1 \leq i \leq L$ in $p \in \wp_N$:

Table 3: Pseudo-code of GA.

```
Initialize Population
i=0
while not stop do
  do N/2 times
    Select two mates from p_i
    Generate two offspring using
crossover operator
    Mutate the two children
    Include children in new generation
p_new
  end do
  Build population p̂_i = p_i ∪ p_new
  Apply Reduction Operators
(Elitist Strategies) to get p_{i+1}
  i=i+1
end
```

$$P_m(i) = \mu \cdot \exp\left(\frac{-mod\{\frac{i-1}{N}\}}{\emptyset}\right) \quad (47)$$

where $\emptyset$ is a normalization constant and μ the change probability at the beginning of each creature p_i in population p; can be described as a positive stochastic matrix in the form:

$$\langle q, \mathbf{M}_{\mathbf{P_m}} p\rangle = \mu^{\Delta(p,q)} \exp\left(-\sum_{dif(i)}^{\Delta(p,q)} \frac{mod\{\frac{i-1}{N}\}}{\emptyset}\right) \cdot \prod_{equ(i)}^{L-\Delta(p,q)} \left[1 - \mu \cdot \exp\left(\frac{-mod\{\frac{i-1}{N}\}}{\emptyset}\right)\right] \quad (48)$$

where $\Delta(p,q)$ is the Hamming distance between p and $q \in \wp_N$, $dif(i)$ resp. $equ(i)$ is the set of indexes where p and q are different resp. equal. Following from equation (48) and checking how the matrices act on populations we can write:

$$\mathbf{M}_{\mathbf{P_m}} = \prod_{\lambda=1}^{N} \left([1 - P_m(\lambda)]\,\mathbf{1} + P_m(\lambda)\hat{\mathbf{m}}^1(\lambda)\right) \quad (49)$$

where $\hat{\mathbf{m}}^1(\lambda) = \mathbf{1} \otimes \mathbf{1} \ldots \otimes \overbrace{\hat{m}^1}^{\lambda} \otimes \ldots \otimes \mathbf{1}$ is a linear operator on $V_\wp$, the free vector space over A^L and $\hat{m}^1$ is the linear 1-bit mutation operator on V_1, the free vector space over A. The latter operator is defined acting on Alphabet as:

$$\langle \hat{a}(\tau'), \hat{m}^1 \hat{a}(\tau)\rangle = (a-1)^{-1}, \quad 0 \le \tau' \ne \tau \le a-1 \quad (50)$$

i.e. probability of change a letter in the Alphabet once mutation occurs with probability equal to $L\mu$.

The spectrum of $\mathbf{M}_{\mathbf{P_m}}$ can be evaluated according to the following expression:

$$sp(\mathbf{M}_{\mathbf{P_m}}) = \left\{\left(1 - \frac{\mu(\lambda)}{a-1}\right)^{\lambda}; \quad \lambda \in [0, L]\right\} \quad (51)$$

where $\mu(\lambda) = \exp\left(\frac{-mod\{\frac{\lambda-1}{N}\}}{\emptyset}\right)$.

The operator presented in equation (49) has similar properties to the Constant Multiple-bit mutation operator $\mathbf{M}_{\boldsymbol{\mu}}$. $\mathbf{M}_{\boldsymbol{\mu}}$ is a contracting map in the sense presented in [53]. It is easy to prove that $\mathbf{M}_{\mathbf{P_m}}$ is a another contracting map, using the Corollary B.1 in [6] and the eigenvalues of this operator(equation (51)).

We can also compare the coefficients of ergodicity:

$$\tau_r(\mathbf{M}_{\mathbf{P_m}}) < \tau_r(\mathbf{M}_{\boldsymbol{\mu}}) \quad (52)$$

where $\tau_r(\mathbf{X}) = max\{\|\mathbf{X}v\|_r : v \in \mathcal{R}^n, \; v \perp e \; and \; \|v\|_r = 1\}$.

Mutation is more likely at the beginning of the string of binary digits ("small neighborhood philosophy"). In order to improve the speed convergence of the algorithm we have included mechanisms such as elitist strategy (reduction operator [58]) in which the best individual in the current generation always survives into the next (a further discussion about reduction operator, $\mathbf{P}_R$, can be found in [59]).

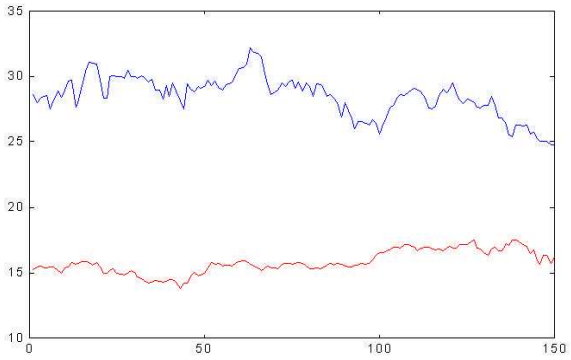


Figure 16: Set of data series. Top: Real Series ACS;Bottom: Real Series BBVA.

Finally the GA is modelled, at each step, as the stochastic matrix product acting on probability distributions over populations:

$$\mathbf{S}_{\mathbf{P^n_m},\mathbf{P^n_c}} = \mathbf{P}_R^n \cdot \mathbf{F_n} \cdot \mathbf{C}^{\mathbf{k}}_{\mathbf{P^n_c}} \cdot \mathbf{M}_{\mathbf{P^n_m}} \quad (53)$$

The GA used in forecasting function (equation (39)) has absolute error value start criterion (i.e $error > uga = 1.5$). Once it starts, it uses the values (or individual) found to be optimal (elite) the last time, and applies local search (using the selected mutation and crossover operators) around this elite individual. Thus we perform an efficient search around an individual (set of $\mathbf{b}_i s$) in which one parameter is more relevant than the others.

The computational time depends on the encoding length, number of individuals and genes. Because of the probabilistic nature of the GA- based method, the proposed method almost converges to a global optimal solution on average. In our simulation nonconvergent case was found. Table 3 shows the GA-pseudocode and in [5] the iterative procedure implemented for the overall prediction system including GA is shown.

5.3 Simulations and Conclusions.

With the aim of assessing the performance of the MRN we have worked with indexes of different Spanish banks and other companies during the same period. We have specifically focussed on the IBEX35 index of Spanish stock, which we consider the most representative sample of Spanish stock movements. We used *MatLab* to implement MRN on a Pentium III at 850MHz.

We started by considering the most simplest case, which consists of two time series corresponding to the companies ACS ($series_1$) and BBVA ($series_2$). The first one is the target of the forecasting process; the second one is introduced as external information. The period under study

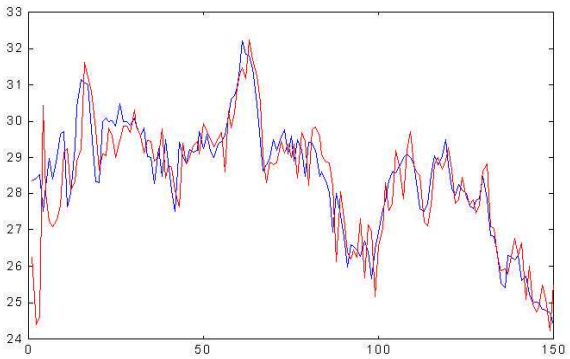


Figure 17: Real Series and Predicted ACS Series with MRN.

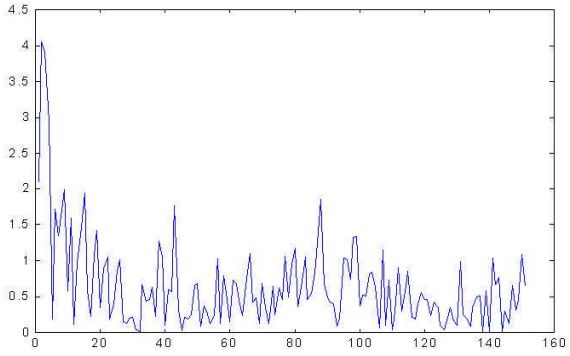


Figure 18: Absolute Error Value with MRN.

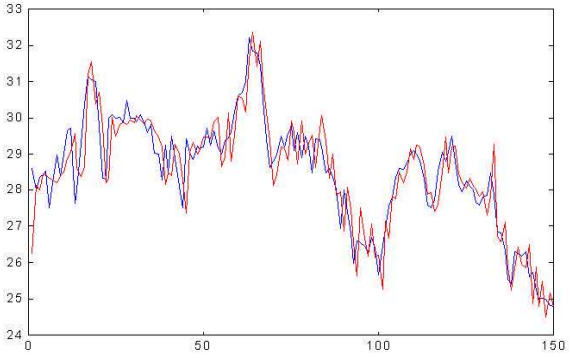


Figure 19: Real Series and Predicted ACS Series with MRN+GA.

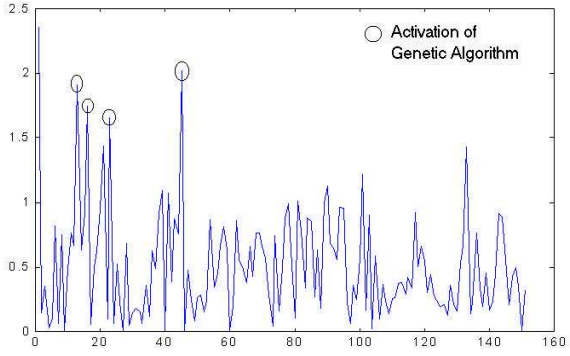


Figure 20: Absolute Error Value with MRN + GA.

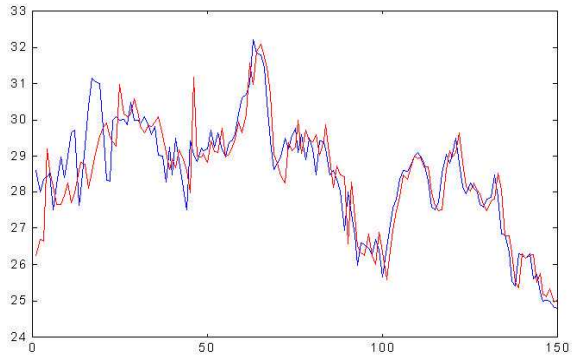


Figure 21: Real Series and Predicted ACS Series without exogenous data.

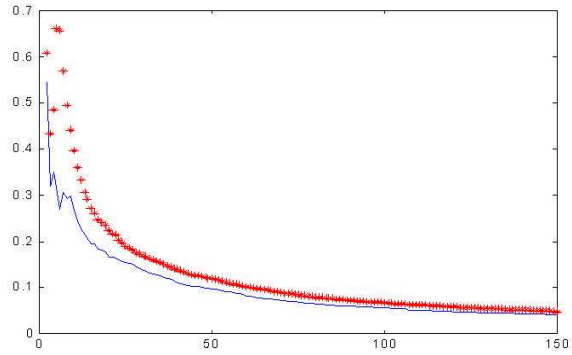


Figure 22: NRMSE evolution for MRN(dot) MRN + GA(line).

covers the year 2000. Each time series includes 200 points corresponding to selling days (quoting days).

We highlight two parameters in the simulation process. The horizon of the forecasting process (*hor*) was set at 1; the weight function of the forecasting function was a correlation function between the two time series for the $series_2$ (in particular we chose its square) and the difference to one for the series 1. We took a forecasting window (**W**) of 10 lags, and the maximum lag number was set at double the value of W, and thus we built a 10×20 Toeplitz matrix. We started at time point $t_o = 50$. Figures 17,18 19 and 20 show the forecasting results from lag 50 to lag 200 corresponding to $series_1$.

Note the instability of the system in the very first iterations until it reaches an acceptable convergence. The most interesting feature of the result is shown in table 4; from this table it is easy to deduce that if we move one of the two series horizontally the correlation between them dramatically decreases. This proves that we avoid the delay problem (trivial prediction) shown by certain networks (see figure 21), in periods where the information introduced to the system is non-relevant. This is due to the increase of information ($series_2$) associated with an increase in neuron resources. At the end of the process we used 20 neurons for net 1 and 21 for net 2. Although forecasting function is acceptable we would expect a better performance with

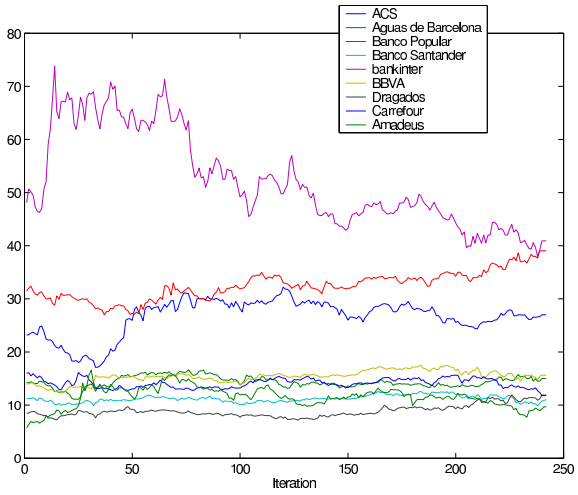


Figure 23: Set of series for complete simulation.

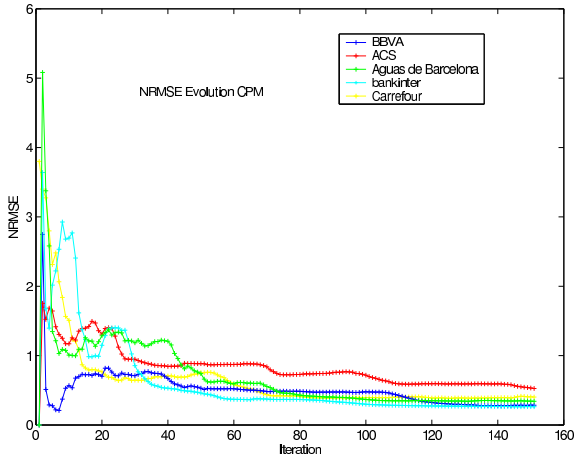


Figure 24: NRMSE evolution for selected stock indexes.

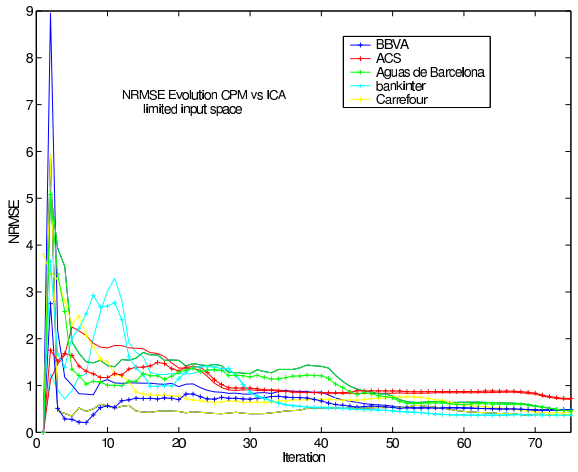


Figure 25: NRMSE evolution using ICA method and MRN.

Table 4: Correlation coefficients between real signal and the predicted signal for different lags.

delay	ρ	delay	ρ
0	0.89	0	0.89
+1	0.79	−1	0.88
+2	0.73	−2	0.88
+3	0.68	−3	0.82
+4	0.63	−4	0.76
+5	0.59	−5	0.71
+6	0.55	−6	0.66
+7	0.49	−7	0.63
+8	0.45	−8	0.61
+9	0.45	−9	0.58
+10	0.44	−10	0.51

Table 5: Dynamics and values of the weights for the GA.

b_{series}	T_1	T_2	T_3	T_4
b_1	0.8924	0.8846	0.8723	0.8760
b_2	0.2770	0.2359	0.2860	0.2634

bigger data set.

The next step consists of using the complete algorithm including the GA. A population of 40 individuals (N_{ind}) was used, with a 2×1 dimension; we used this small number because we had a bounded searching space and we were using a single PC. The genetic algorithm was run four times before reaching the convergence (when the error increase by 1.5 points, see error plot in figure 20 to see the effect of GA) ; the individuals were codified with 34 bits (17 bits for each parameter). In this case convergence is defined in terms of the adjustment function; other authors use other parameters of the GA, like the absence of change in the individuals after a certain number of generations, etc. We observed a considerable improvement in the forecasting results and noted disappearance of the delay problem, as is shown in table 4. This table represents the correlation between the real function and the neural function for different lags. The correlation function presents a maximum at lag 0.

The number of neurons at the end of the process is the same as in the latter case, because we have only modified the weight of each series during the forecasting process. The dynamics and values of the weights are shown in table 5.

Error behaviour is shown in figures. Note:

- We can bound the error by means of a suitable selection of the parameters $\mathbf{b_i}$, when the dynamics of the series is coherent (avoiding large fluctuations in the stock).
- The algorithm converges faster, as is shown at the very

beginning of the graph.

– The forecasting results are better using GA, as is shown in figure 20, where the evolution of the normalized round mean square error is plotted.

Finally we carried out a simulation with 9 indexes and computed the prediction function for 5 series, obtaining similar results, as presented in figure 24. In the complete model we limited the input space dimension to 3 for extra series and 5 for target series (figure 22). NRMSE depends on each series (data set) and target series (evolution). In figure 25 we also compare the ICA method versus MRN for limited data set (70 iterations). NRMSE of indexes increases at the beginning of the process using the ICA method and converges to CPM NRMSE values when the data set increases (estimators approach higher order statistics). This effect is also observed when the dynamics of the series change suddenly due to a new independent event.

Due to the symmetric character of our forecasting model, it is sufficient to implement it in parallel programming software (such as PVM —*Parallel Virtual Machine*—) or MPI — *Message-Passing Interface*— [60]) to build a more general forecasting model for the complete set of series. We would spawn the same number for offspring processes and banks; these process would run forecasting vectors, which would be weighted by a square matrix with dimension equal to the number of series $\mathbf{B}$. The "master" process would have the results of the forecasting process for the calculus of the error vector, in order to update the neuron resources. Thus we would take advantage of the computational cost of a forecasting function to calculate the rest of the series (see [60]).

5.3.1 Conclusions

This new forecasting model for time-series is characterized by:

– The enclosing of external information. We avoid preprocessing and data contamination applying by ICA and PCA for limited data sets or sudden new shocks. These techniques can be included in MRN under better conditions. Series are introduced into the net directly.

– The forecasting results are improved using hybrid techniques like GA.

– The possibility of implementing in parallel programming languages (i.e. PVM — see [60]); and the improved performance and lower computational time achieved using a parallel neural network.

6 Comparison among methods

Consider the set of series in Figure 6, using 1000 point as training samples. We apply the latter models to forecast Sony index in 70 future points using the other indexes as endogenous variables. Following the methodology described in the previous sections we get:

Table 6: Eigenvalues λ_i and variance percentages of PCs.

Index	*Eigenvalue*	*Pct. variance*	*Pct. overall*
1	0.0031	35.1642	35.1642
2	0.0016	17.9489	53.1131
3	0.0010	11.7188	64.8319
4	0.0006	6.9857	71.8176
5	0.0005	5.7787	77.5963
6	0.0004	4.8978	82.4941
7	0.0004	4.4184	86.9125
8	0.0004	4.2492	91.1617
9	0.0003	3.7238	94.8855
10	0.0003	3.2031	98.0886
11	0.0002	1.9113	100.00

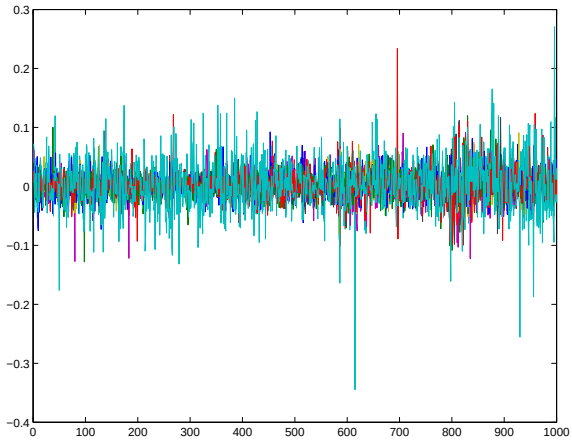


Figure 26: Closing prices evolution for selected indexes (Figure 6), after logarithmic transformation and differentiation.

– PCA operation: in this case results using PCA are not so good as we expected. The volatile nature of the series and the lack of information from the 3 principal components used contributes to this failure. The 3 components only hold a fraction equal to 64% of variance as is shown in Table 6, however if we increase the number of components used in prediction we get worse results owing to "curse of dimensionality".

– MRN operation: MRN get good prediction results comparing with PCA method, however in the last iterations PCA and MRN methods are of similar NRMSE. The main disadvantages of MRN are computational demand and the need for bounding input space dimension.

– ICA operation: ICA is the best method using large sample size as is shown in Figure 28. ICA reveals some underlying structure in the data since we used HOS to estimate ICs that usually fall into two cate-

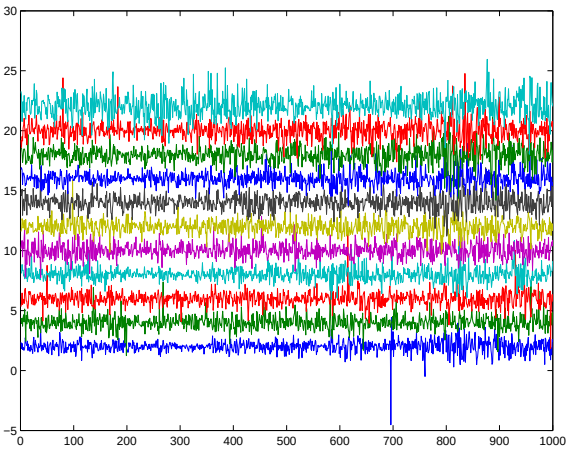


Figure 27: ICA ICs for the set of indexes.

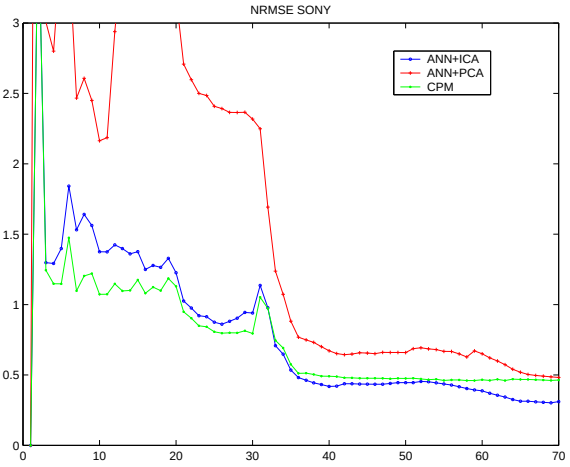


Figure 28: NRMSE evolution of SONY index for the 70 forecasted points using exogenous methods.

gories as we mentioned in section 4(see Figure 27): infrequent but large shocks (responsible for the major changes in the stock prices) and frequent but rather small fluctuations (contributing only little to the overall level of the stocks).

Acknowledgement

We want to thank SESIBON and HIWIRE grants from Spanish Government for funding.

References

[1] G. Box, G. Jenkins, and G. Reinsel, *Time Series Analysis: Forecasting and Control*, 3rd ed. Prentice-Hall. Englewood Cliffs, New Jersey, U.S.A., 1994.

[2] J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, pp. 213–225, 1991.

[3] M. Salmerón, J. Ortega, C. Puntonet, and F. Pelayo, *Time Series Prediction with Hybrid Neuronal, Statistical and Matrix Methods (in Spanish)*. Department of Computer Architecture and Computer Technology. University of Granada, Spain, 2001.

[4] M. Salmerón, J. Ortega, C. G. Puntonet, and A. Prieto, "Improved ran sequential prediction using orthogonal techniques," *Neurocomputing*, vol. 41, pp. 153–172, 2001.

[5] J. M. Górriz, C. G. Puntonet, M. Salmerón, and J. González, "New model for time-series forecasting using rbfś and exogenous data," *Neural Computation and Applications, Vol 13 Issue 2. Jun. 2004.*, 2003.

[6] J. M. Górriz, "Algoritmos híbridos para la modelización de series temporales con técnicas ar-ica," Ph.D. dissertation, University of Cádiz , Departamento de Ing. de Sistemas y Aut. Tec. Electrónica y Electrónica. http://wwwlib.umi.com/cr/uca/main, 2003.

[7] T. Masters, *Neural, Novel and Hybrid Algorithms for Time Series Prediction*. John Wiley & Sons. New York, U.S.A., 1995.

[8] A. Back and A. Weigend, "Discovering structure in finance using independent component analysis," *Computational Finance*, 1997.

[9] A. Back and T. Trappenberg, "Selecting inputs for modelling using normalized higher order statistics and independent component analysis," *IEEE Transactions on Neural Networks*, vol. 12, 2001.

[10] A. Hyvarinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Networks*, vol. 13, pp. 411–430, 2000.

[11] A. Bell and T. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, pp. 1129–1159, 1995.

[12] P. Comon, "Independent component analysis, a new concept?" *Signal Processing*, vol. 3, pp. 287–314, 1994.

[13] S. Amari, A. Cichocki, and H. Yang, "A new learning algorithm for blind source separation," *Advances in Neural Information Processing Systems. MIT Press*, vol. 8, pp. 757–763, 1996.

[14] C. G. Puntonet, *Nuevos algoritmos de Separación de fuentes en medios Lineales (in Spanish)*. Department of Computer Architecture and Computer Technology. University of Granada, Spain, 1994.

[15] A. Mansour, N. Ohnishi, and C. Puntonet, "Blind multiuser separation of instantaneous mixture algorithm based on geometrical concepts," *Signal Processing*, vol. 82, pp. 1155–1175, 2002.

[16] D. Pollock, *A Handbook of Time Series Analysis, Signal Processing and Dynamics*. Academic Press. New York, U.S.A., 1999.

[17] G. Zhang, B. Patuwo, and M. Hu, "Forecasting with artificial neural networks: the state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[18] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 284–294, 1989.

[19] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. L. Cam and J. Neyman, Eds., vol. 1. Berkeley, California, U.S.A.: University of California Press, 1967, pp. 281–297.

[20] C. Lawson and R. Hanson, *Solving Least Squares Problems*. SIAM Publications. Philadelphia, U.S.A., 1995.

[21] G. Zelniker and F. Taylor, *Advanced Digital Signal Processing: Theory and Applications*. Marcel Dekker. New York, U.S.A., 1994.

[22] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company. New York, U.S.A., 1994.

[23] A. Tikhonov and V. Arsenin, *Solutions of Ill-Posed Problems*. Winston. Washington D.C., U.S.A., 1977.

[24] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philos. Trans. Roy. Soc. London*, vol. A, pp. 415–446, 1909.

[25] A. J. Smola, B. Schölkopf, and K.-R. Müller, "The connection between regularization operators and support vector kernels," *Neural Networks*, vol. 11, pp. 637–649, 1998.

[26] J. M. Górriz, C. G., Puntonet, and M. Salmerón, "On line algorithm for time series prediction based on support vector machine philosophy," *LNCS*, vol. 3037, pp. 50–57, 2004.

[27] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, pp. 1087–1092, 1953.

[28] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, 1990.

[29] L. Cao, "Support vector machines experts for time series forecasting," *Neurocomputing*, pp. 321–339, 2003.

[30] A. Comrey, *A First Course in Factor Analysis*. Academic Press. New York, U.S.A., 1973.

[31] J. Ramsay and B. Silverman, *Functional Data Analysis*. Springer-Verlag. New York, U.S.A., 1997.

[32] R. J. M. II, "Intelligence: Computational versus artificial," *IEEE Transactions on Neural Networks*, vol. 4, pp. 737–739, 1993.

[33] J. Jackson, *A User's Guide to Principal Components*. John Wiley & Sons. New York, U.S.A., 1991.

[34] G. Golub and C. V. Loan, *Matrix Computations*, 3rd ed. The Johns Hopkins University Press. Baltimore, Maryland, U.S.A., 1996.

[35] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation*. Addison-Wesley. Redwood City, California, U.S.A., 1991.

[36] K. Diamantaras and S. Kung, *Principal Component Neural Networks: Theory and Applications*. John Wiley & Sons. New York, U.S.A., 1996.

[37] E. Oja, "Principal components, minor components, and linear neural networks," *Neural Networks*, vol. 5, pp. 927–935, 1992.

[38] W. Sarle, "Neural networks and statistical models," in *Proceedings of the 19th Anual SAS Users Group International Conference*, Cary, NC, U.S.A., 1994, pp. 1538–1550.

[39] J. Park and I. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, pp. 246–257, 1991.

[40] J. M. Górriz, C. G. Puntonet, M. Salmerón, and J. Ortega, "New method for filtered ica signals applied to volatile time series," *7th International Work Conference on Artificial and Natural Neural Networks IWANN 2003. Lecture Notes in Computer Science Vol 2687 / 2003, Springer 433-440*, 2003.

[41] S. Cruces, *An unified view of BSS algorithms(in Spanish)*. University of Vigo, Spain, 1999.

[42] K. Kiviluoto and E. Oja, "Independent component analysis for parallel financial time series," *Proc. in ICONIP98*, vol. 1, pp. 895–898, 1998.

[43] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of Statistical Learning*. Springer, 2000.

[44] A. Savitzky and M. Golay, *Analytical Chemestry*, vol. 36, pp. 1627–1639, 1964.

[45] R. Hamming, *Digital Filters*, 2^a ed. Prentice Hall, 1983.

[46] H. Ziegler, *Applied Spectroscopy*, vol. 35, pp. 88–92, 1981.

[47] W. Press, S. Teukolsky, W. Vertterling, and B. Flannery, *Numerical Recipes in C++*, 2^a ed. Cambridge University Press, 2002.

[48] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281–294, 1989.

[49] O. Haggstrom, *Finite Markov Chains and Algorithmic Applications.* Cambridge University, 1998.

[50] L. Schmitt, C. Nehaniv, and R. Fujii, "Linear analysis of genetic algorithms," *Theoretical Computer Science*, vol. 200, pp. 101–134, 1998.

[51] J. Suzuki, "A markov chain analysis on simple genetic algorithms," *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 25, pp. 655–659, 1995.

[52] A. Eiben, E. Aarts, and K. V. Hee, "Global convergence of genetic algorithms: a markov chain analysis," *Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, vol. 496, pp. 4–12, 1991.

[53] L. Schmitt, "Theory of genetic algorithms," *Theoretical Computer Science*, vol. 259, pp. 1–61, 2001.

[54] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.

[55] T. S. S. Matwin and K. Haigh, "Genetic algorithms approach to a negotiation support system," *IEEE Trans. Syst. , Man. Cybern*, vol. 21, pp. 102–114, 1991.

[56] S. Chen and Y. Wu, "Genetic algorithm optimization for blind channel identification with higher order cumulant fitting," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 259–264, 1997.

[57] L. Chao and W. Sethares, "Non linear parameter estimation via the genetic algorithm," *IEEE Transactions on Signal Processing*, vol. 42, pp. 927–935, 1994.

[58] J. Lozano, P. Larranaga, M. Grana, and F. Albizuri, "Genetic algorithms: Bridging the convergence gap," *Theoretical Computer Science*, vol. 229, pp. 11–22, 1999.

[59] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Transactions on Neural Networks*, vol. 5, pp. 96–101, 1994.

[60] J. M. Górriz, C. G. Puntonet, M. Salmerón, and R. Martin-Clemente, "Parallelization of time series forecasting model," *12 th IEEE Conference on Parallel, Distributed and Network based Processing (Euromicro) PDP 2004 pp 103-112. A Coruña, Spain.*, 2004.

Integrating Multi-Objective Genetic Algorithm and Validity Analysis for Locating and Ranking Alternative Clustering

Yimin Liu, Tansel Özyer, Reda Alhajj and Ken Barker
Advanced Database Systems and Applications Lab
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada
{liuyi, ozyer, alhajj, barker}@cpsc.ucalgary.ca

Keywords: multi-objective genetic algorithm, clustering, k-means, validity analysis

Received: March 15, 2004

Clustering algorithms in general need the number of clusters as a priori, which is mostly hard for domain experts to estimate. In this paper, we use Niched Pareto k-means Genetic Algorithm (GA) for clustering. After running the multi-objective GA, we get the pareto-optimal front that gives the optimal number of clusters as a solution set. We analyze the clustering results using several cluster validity techniques proposed in the literature, namely Silhoutte, C index, Dunn's index, DB index, SD index and S-Dbw index. This gives an idea about ranking the optimal number of clusters for each validity index. We demonstrate the applicability and effectiveness of the proposed clustering approach by conducting experiments using two datasets: Iris and the well-known Ruspini dataset.

Povzetek: "[Click here and Enter short Abstract in Slovene language]"

1 Introduction

Data mining methods and techniques have been successfully applied to different areas including bioinformatics. They are designed for extracting previously unknown significant relationships and regularities out of huge heaps of details in large data collections [11].

Classification is one of the well-known mining techniques. It has two main aspects: discrimination and clustering. In discrimination analysis, also known as supervised clustering, observations are known to belong to pre-specified classes. The task is to allocate predictors for the new coming instances to be able to classify them correctly. In contrast to classification, in clustering, also known as unsupervised clustering, classes are unknown *a prior*; the task is to determine classes from the data instances. Clustering is used to describe methods to group unlabeled data. By clustering, we aim to discover gene/samples groups that enable us to discover, for example, the functional role or the existence of a regulatory novel gene among the members in a group. The literature shows that increasing attention is devoted to the development of new clustering techniques [12]. Existing clustering techniques mostly used for gene expression data clustering can be classified into traditional clustering algorithms including hierarchical clustering [20], partitioning [22], and recently emerging clustering techniques such as graph-based [19] and model-based [21, 23] approaches.

As described in the literature, some of the existing clustering techniques have been successfully employed in analyzing gene expression data. These include hierarchical clustering, partitional clustering, graph-based clustering, and model-based clustering. In general, existing clustering techniques require pre-specification of the number of clusters, which is not an easy task to predict *a prior* even for experts. Thus, the problem handled in this paper may be identified as follows: Given a set of data instances, we mainly concentrate on microarray data, it is required to develop an approach that produces different alternative solutions, and then conduct validity analysis on the resulting solutions to rank them.

Different assumptions and terminologies were considered for the components of the clustering process and the context in which clustering is used. There exist fuzzy clustering techniques as well as hard clustering techniques. Hard clustering assigns a label l_i to each object x_i, identifying its class label. The set of all labels for the object set is $\{ l_1 , l_2 ,\ldots, l_n \}$, where $l_i \in \{ l_1 , l_2 ,\ldots, l_k \}$, and k is the number of clusters. In fuzzy clustering, an object may belong to more than one cluster, but with different degree of membership; an object x_i is assigned to cluster j based on the value of the corresponding function f_{ij}. The membership of an object may not be precisely defined; there is likelihood that each object may or may not be member of some clusters. The presence of noise in the data set may be quite high.

Our approach presented in this paper has been designed to smoothly handle the clustering of different data sets. In the existing approaches, the number of clusters is mostly given a-priori. This motivated us to consider the idea of proposing multi-objective k-means

genetic algorithm (MOKGA) approach in order to present to the user several alternatives without taking the weight values into account. Otherwise, the user will have several trials weighting with different values until a satisfactory result is obtained. We evaluate the obtained candidate optimal number of clusters by applying the cluster validity techniques, namely Silhoutte, C index, Dunn's index, DB index, SD index and S-Dbw index. Finally, the proposed approach has been tested using the Iris and Ruspini datasets.

As *K*-Means clustering is concerned, it is a commonly used algorithm for partition clustering [22]. It is a widely used technique and has been utilized to analyze gene expression data. The purpose of *K*-Means clustering is the optimization of an objective function that is described by the equation:

$$E = \sum_{i=1}^{c} \sum_{x \in C_i} d(x, m_i) \quad (1)$$

where m_i is the center of cluster C_i, and $d(x, m_i)$ is the Euclidean distance between a point x and m_i. It can be seen that the criterion function attempts to minimize the distance between each point and the center of its cluster. The algorithm begins by randomly initializing a set of C cluster centers, then assigns each object of the dataset to the cluster whose center is the nearest, and re-computes the centers. This process is repeated until the total error criterion converges.

The rest of the paper is as follows. Section 2 is an overview of the multi-objective approach. Section 3 describes the proposed system: namely, clustering and cluster validity analysis. Section 4 includes the experimental results. Section 5 is the conclusions.

2 Multi-objective Genetic Algorithms

A multi-objective optimization problem has *n* decision variables, *k* objective functions, and *m* constraints. Objective functions and constraints are functions of the decision variables. The optimization goal may be described as follows:

$$\begin{aligned} &\max imize \setminus \min imize \quad y = f(x) = (f_1(x), f_2(x), ..., f_k(x)) \\ &subject \quad to \quad e(x) = (e_1(x), e_2(x), ..., e_m(x)) \le 0 \\ &where \qquad x = ((x_1, x_2, ..., x_n) \in X \\ &\qquad\quad y = ((y_1, y_2, ..., y_n) \in Y \end{aligned} \quad (2)$$

where x is the decision vector, y is the objective vector, X denotes the decision space, and Y is called the objective space. The constraints $e(x) \ge 0$ determine the set of feasible solutions [14].

Solutions to a multi-objective optimization method are mathematically expressed in terms of non-dominated or superior points. In a minimization problem, a vector $x^{(1)}$ is partially less than another vector $x^{(2)}$, denoted $x^{(1)} \prec x^{(2)}$, when no value of $x^{(2)}$ is less than $x^{(1)}$ and at least one value of $x^{(2)}$ is strictly greater than $x^{(1)}$. If $x^{(1)}$ is partially less than $x^{(2)}$, we say that $x^{(1)}$ dominates $x^{(2)}$ or the solution $x^{(2)}$ is inferior to $x^{(1)}$. Any vector which is not dominated by any other vectors is said to be non-dominated or non-inferior. The optimal solutions to a multi-objective optimization problem are non-dominated solutions [13].

A common difficulty with the multi-objective optimization is the conflict between the objective functions. None of the feasible solutions allows optimal solutions for all the objectives. Pareto-optimal is the solution, which offers the least objective conflict. In traditional multi-objective optimization, multiple objectives are combined to form one objective function. One of the traditional methods being used is weighting each objective and scalarizing the result. At the end of each run, pareto-optimal front may be obtained, which actually represents one single point.

3 The Proposed Approach

In this paper, we propose a new clustering approach, namely Multi-Objective Genetic K-means algorithm (MOKGA), which is a general purpose approach for clustering other datasets after modifying the fitness functions and changing the proximity values as distance or non-decreasing similarity function according to the requirements of the dataset to be clustered.

Concerning our approach, after running the multi-objective k-means genetic algorithm, we get the pareto-optimal front that gives the optimal number of clusters as a solution set. Then, the system analyzes the clustering results found under six of the cluster validity techniques proposed in the literature, namely Silhoutte, *C* index, Dunn's index, SD index, DB index and S_Dbw index.

3.1 A. Multi-Objective Genetic K-Means Algorithm

The Multi-Objective Genetic K-means Algorithm (MOKGA) is basically the combination of the Fast Genetic K-means Algorithm (FGKA) [1] and Niched Pareto Genetic Algorithm [2].

As presented in the flowchart shown in Figure 1, MOKGA uses a list of parameters to drive the evaluation procedure as in the other genetic types of algorithms: including population size (number of chromosomes), t_dom (number of comparison set) representing the assumed non-dominated set, crossover, mutation probability and the number of iterations that the execution of the algorithm needs to obtain the result.

Sub-goals can be defined as fitness functions; and instead of scalarizing them to find the goal as the overall fitness function with the user defined weight values, we expect the system to find the set of best solutions, i.e., the pareto-optimal front. By using the specified formulas, at each generation, each chromosome in the population is evaluated and assigned a value for each fitness function.

The coding of our individual population is a chromosome of length *n*. Each allele in the chromosome takes a value from the set {1, 2, …, *K*}, and represents a pattern. The value indicates the cluster that the corresponding pattern belongs to. Each chromosome exhibits a solution set in the population. If the chromosome has *k* clusters, then each gene a_n (n=1 to N) takes different values from [1..*k*].

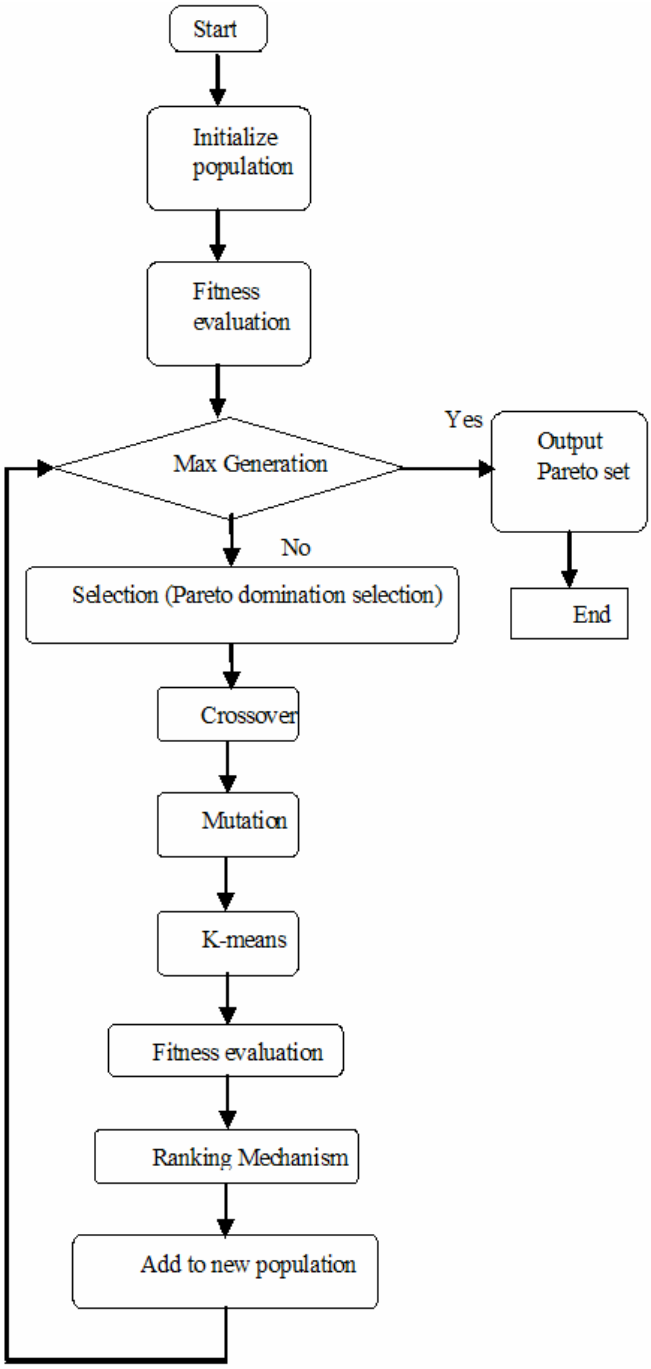


Figure 1: Flow chart: the process of the Multi-Objective Genetic K-means Algorithm

Initially, *current generation* is assigned to zero. Each chromosome takes *number of clusters* parameter within the range 1 to maximum number of clusters given by the user. A population with the specified number of chromosomes is created randomly by using the method described in [5]: Data points are randomly assigned to each cluster at the beginning; then the rest of the points are randomly assigned to clusters. By using this method, we can avoid generating illegal strings, which means some clusters do not have any pattern in the string.

Using the current population, the next population is generated and generation number is incremented by 1. During the next generation, the current population performs the pareto domination tournament to get rid of the worst solutions from the population, crossover, mutation and k-means operator [1] to reorganize each object's assigned cluster number. Finally, we will have twice the number of individuals after the pareto domination tournament. We apply the ranking mechanism used in [15] to satisfy the elitism and diversity preservation. By using this method the number of individuals is halved.

The first step in the construction of the next generation is the selection using pareto domination tournaments: In this step, two candidate items picked among (*population size- t_{dom}*) individuals participate in the pareto domination tournament against the t_{dom} individuals for the survival of each in the population. In the selection part, t_{dom} individuals are randomly picked from the population. With two randomly selected chromosome candidates in (*population size- t_{dom}*) individuals, each of the candidates is compared against each individual in the comparison set, t_{dom}. If one candidate has a larger total within-cluster variation fitness value and a larger number of cluster values than of all of the chromosomes in the comparison set, this means it is dominated by the comparison set already and will be deleted from the population permanently. Otherwise, it resides in the population.

After the pareto domination tournament, one-point crossover operator is applied on randomly chosen two chromosomes. The crossover operation is carried out on the population with the crossover p_c. After the crossover, assigned cluster number for each gene is renumbered beginning from a_1 to a_n. For example, if two chromosomes having 3 clusters and 5 clusters, respectively, need to have a crossover at the third location:

Number of clusters=3: 1 2 3 3 3
Number of clusters=5: 1 4 3 2 5

We will get 1 2 3 2 5 and 1 4 3 3 3; and then they are renumbered to get the new number of clusters parameters:

Number of clusters=4: 1 2 3 2 4 (for 1 2 3 2 5)
Number of clusters=3: 1 2 3 3 3 (for 1 4 3 3 3)

The reason for choosing one-point crossover is because it produced better results compared to multi-point after some initial experiments.

The mutation operator on the current population is employed after the crossover. During the mutation, we replace each gene value a_n by a_n' with respect to the probability distribution; for n=1, ..., N simultaneously. a_n' is a cluster number randomly selected from {1, …, K} with the probability distribution $\{p_1, p_2, ..., p_K\}$ defined using the following formula:

$$p_i = \frac{1.5 * d_{\max}(X_n) - d(X_n, c_k)}{\sum_{k=1}^{K} 1.5 * d_{\max}(X_n) - d(X_n, c_k)} \quad (3)$$

where $i \in [1..k]$ and $d(X_n, C_k)$ denotes Euclidean distance between pattern X_n and the centroid C_k of the k-th cluster. $d_{\max}(X_n) = \max_k\{d(X_n, C_k)\}$, p_i represents what

the probability interval of mutating gene is assigned to cluster *i* (e.g., Roulette Wheel).

Finally, k-means operator is applied. It is used to reanalyze each chromosome gene's assigned cluster value; it calculates the cluster centre for each cluster; and then it re-assigns each gene to the cluster that is the closest one to the instance in the gene. Hence, *k*-means operator is used to speed up the convergence process by replacing a_n by a_n' for n=1, ..., N simultaneously, where a_n' is the closest to object X_n in Euclidean distance.

After all the operators are applied, we have twice the number of individuals, after having the pareto dominated tournament. We can not give an exact number as equal to the number of initial population size because at each generation randomly picked candidates are picked for the survival test leading to deletion of one or both, in case dominated. To halve the number of individuals, having the number of individuals we had, the ranking mechanism proposed in [15] is employed. So, the individuals obtained after crossover, mutation and k-means operator are ranked, and we pick the best individuals among them to place in the population for the next generation.

Our approach picks the first *l* individuals considering the elitism and diversity among *2l* individuals. Pareto fronts are ranked. Basically, we find the pareto-optimal front and remove the individuals of the pareto-optimal front from *2l* set and place it in the population to be run in the next generation. In the remaining set, again we get the first pareto-optimal front and we put it in the population and so on. Since we try to get the first *l* individuals, the last pareto-optimal front may have more individuals required to complete the number of individuals to *l*, we handle the diversity automatically. We rank them and reduce the objective dimension into one. Then, we sum the normalized value of the objective functions of each individual. We sort them in increasing order and find each individual's total difference from its individual pairs, the one with the closest smaller summed values and the one with the closest greater summed values. After sorting the individuals in terms of each one's total difference in decreasing order, we keep placing from the top as many individuals as we need to complete the number of population to *l*. The reason for doing this is to take the crowding factor into account automatically, so that individuals occurring closer to others are unlikely to be picked. Solutions far apart from the others will be considered for the necessity of diversity. Further details are given in [15]. This method was also suggested as a solution for the elitism and diversity for improvement in NSGA-II.

Finally, if the maximum number of generations is reached, or the prespecified threshold is satisfied then exit; otherwise the next generation is performed.

During our clustering process, we defined two objective functions: minimizing the number of clusters and minimizing the partitioning error. To partition the *N* pattern points into *K* clusters one goal is to minimize the Total Within-Cluster Variation (*TWCV*), which is specified as:

$$TWCV = \sum_{n=1}^{N}\sum_{d=1}^{D} X_{nd}^2 - \sum_{k=1}^{K}\frac{1}{Z_k}\sum_{d=1}^{D} SF_{kd}^2 \qquad (4)$$

where $X_1, X_2, .., X_N$ are the N objects, X_{nd} denotes feature d of pattern X_n (n = 1 to N). SF_{kd} is the sum of the d-th features of all the patterns in cluster k (G_k) and Z_k denote the number of patterns in cluster k (G_k) and SF_{kd} is:

$$SF_{kd} = \sum_{\overrightarrow{X_n} \in G_k} X_{nd}, \quad (d = 1, 2, ...D). \qquad (5)$$

And the other objective function is to minimize the *number of clusters* parameter. Under the lights of these two objective functions, after running the algorithm, we aim at obtaining the first pareto optimal front having the best partition with the least number of clusters as optimal solution set.

3.2 Cluster Validity Techniques

Clustering is an unsupervised task and after clustering the data, partitioning into subgroups, we need to check its validity. The criteria widely accepted by the clustering algorithms are the compactness of the cluster and their well-separateness. Those criteria should be validated and optimal clusters should be found, so the correct input parameters must be given to the satisfaction of optimal clusters. Basically, the number of clusters is given as a priori. However, pareto-optimal solution set for the clustering results is obtained in our approach, MOKGA. We believe that these are the good clustering outcomes, and we use the cluster validity index to decide and see the overall picture of those validity index value changes for each *number of clusters* parameter value in the solution set. In our system, we considered six cluster validity techniques widely used for the validation task. These are Dunn index [4], Davies-Bouldin index [3], Silhouette index [5], C index [6], SD index [8] and S_Dbw index [9]. Based on the validated results, the optimal number of clusters can be determined.

The SD validity index definition is based on the concepts of average scattering for clusters and total separation between clusters. The average scattering for clusters is defined as:

$$Scatt(n_c) = \frac{1}{n_c}\sum_{i=1}^{n_c} \left\| \sigma(v_i) \right\| \Big/ \left\| \sigma(x) \right\| \qquad (6)$$

where $\sigma(v_i)$ is the average standard deviation (average of the Euclidian distance between all the points) of cluster centers; and $\sigma(x)$ is the average standard deviation of all the data points. The total separation between clusters is defined as:

$$Dis(n_c) = \frac{D_{\max}}{D_{\min}} \sum_{k=1}^{n_c} \left(\sum_{i=1}^{n_c} \left\| v_k - v_z \right\| \right)^{-1} \qquad (8)$$

where, D_{max} = max($\|v_i - v_j\|$) $\forall i, j \in \{1, 2,3,\ldots, n_c\}$ is the maximum distance between cluster centers and D_{min}= min($\|v_i - v_j\|$) $\forall i, j \in \{1, 2,\ldots, n_c\}$ is the minimum distance between cluster centers.

The SD index is calculated using the following equation:

$$SD(n_c) = \alpha \times Scat(n_c) + Dis(n_c) \qquad (9)$$

where α is a weighting factor.

In the above equation, Scat(n_c) indicates the average compactness of clusters. A small value for this term indicates compact clusters. *Dis*(n_c) indicates the total separation between the *n* clusters. Since the two terms of *SD* have different ranges, a weighting factor is needed to incorporate both terms in a balanced way. The number of clusters that minimizes the index is an optimal value.

S_Dbw is formalized based on the clusters' compactness (intra-cluster variance) and the density (Inter-cluster Density) between clusters. Inter-cluster density is defined as follows:

$$Dens\_bw(n_c) = \frac{1}{n_c \times (n_c - 1)} \sum_{i=1}^{n_c} \left(\sum_{\substack{j=1 \\ i \neq j}}^{n_c} \frac{density(u_{ij})}{\max\{density(v_i), density(v_j)\}} \right) \quad (10)$$

where v_i and v_j are centers of clusters c_i and c_j; and u_{ij} is the middle point of the line segment defined by the clusters' centers v_i and v_j. The term *density(u)* is given by following equation:

$$density(u) = \sum_{l=1}^{n_{ij}} f(x_l, u) \quad (11)$$

where n_{ij} is the number of tuples that belong to the cluster c_i and c_j, i.e., $x_l \in c_i$, and $c_j \in S$. Function $f(x,u)$ is defined as:

$$f(x,u) = \begin{cases} 0, & if \quad d(x,u) > stedev \\ 1, & otherwise \end{cases} \quad (12)$$

where *stedev* is the average standard deviation of clusters.

Inter-cluster Density (ID) evaluates the average density in the region among clusters in relation to the density of the clusters. Intra-cluster variance measures the average scattering of clusters (*Scat*(n_c)) and has already been defined in the *SD* index part.

The *S_Dbw* is calculated using the following equation:

$$S\_Dbw(n_c) = Scat(n_c) + Dens\_bw(n_c) \quad (13)$$

the definition of *S_Dbw* considers both compactness and separation. The number of clusters that minimizes the index is an optimal value.

The Dunn index is calculated using the following equation :

$$D_{n_c} = \min_{i=1,\ldots,n_c} \left\{ \min_{j=i+1,\ldots,n_c} \left[\frac{\frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} d(x,y)}{\max_{k=1,\ldots,n_c} \left(\frac{\sum_{x \in C_k} d(x,c_k)}{|C_k|} \right)^2} \right] \right\} \quad (14)$$

where c_i represents the *i*-cluster of a certain partition, d(x,y) is the distance between data points x and y, where x belongs to cluster *i* and y belongs to cluster *j* , d(x, c_k) is the distance of data point x to the cluster centre that it belongs to, $|C_k|$ is the number of data points in cluster K.

The main goal of the measure is to maximize the intercluster distances and minimize the intracluster distances. Therefore, the number of clusters that maximizes D is taken as the optimal number of clusters.

The DB index is calculated using the following equation:

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{i \neq j} \left\{ \frac{S_n(Q_i) + S_n(Q_j)}{S(Q_i, Q_j)} \right\} \quad (15)$$

where *n* is the number of clusters, S_n is the average distance of all objects from the cluster to their cluster center, $S(Q_i, Q_j)$ denotes the distance between centres of clusters.

The Davies-Bouldin index is a function of the ratio of the sum of within-cluster scatter to between cluster separation. When it has a small value it exhibits a good clustering.

The following formula is used to calculate the Silhouette index:

$$S(i) = \frac{(b(i) - a(i))}{\max\{a(i), b(i)\}} \quad (16)$$

where $a(i)$ is the average dissimilarity of *i*-object to all other objects in the same cluster, Euclidian distance is used to calculate the dissimilarity; and $b(i)$ is the average dissimilarity of *i*-object to all objects in the closest cluster.

The formula indicates that the silhouette value is in the interval [–1, 1]:

- Silhouette value is close to 1: means that the sample is assigned to a very appropriate cluster.
- Silhouette value is about 0: means that that the sample lies equally far away from both clusters, it can be assigned to another closest cluster as well.
- Silhouette value is close to –1: means that the sample is "misclassified".

The partition with the largest overall average silhouette means the best clustering. So, the number of clusters with the maximum overall average silhouette width is taken as the optimal number of clusters. This index is defined as follows:

$$C = \frac{S - S_{\min}}{S_{\max} - S_{\min}} \quad (17)$$

where *S* is the sum of distances over all pairs of patterns from the same cluster, *L* is the number of pairs for calculating S_{min} and S_{max}, S_{min} is the sum of the *l* smallest distances if all pairs of patterns are considered, and S_{max} is the sum of the *l* largest distances out of all pairs. It can be seen that a small value of *C* indicates a good clustering.

4 Experiments

We conducted our experiments on Intel® 4, 2.00 GHz CPU, 512 MB RAM running Windows XP Dell PC. The proposed MOKGA approach and the utilized cluster validity algorithms have been implemented using Microsoft Visual Studio 6.0 C++. We used two data sets in the evaluation process. The first data set is the Iris dataset [17]. It contains 150 instances each having 4 attributes; it has three clusters each has 50 instances. The Iris dataset is a famous dataset widely used in pattern recognition and clustering. One cluster is linearly separable from the other two and the latter two are not exactly linearly separable from each other. The second data set is the Ruspini dataset with 75 instances with 2 attributes and integer coordinates: $0 < X < 120$, $0 < Y <$

160, which might be naturally grouped into 4 sets [16]. The Ruspini dataset is popular for illustrating clustering techniques.

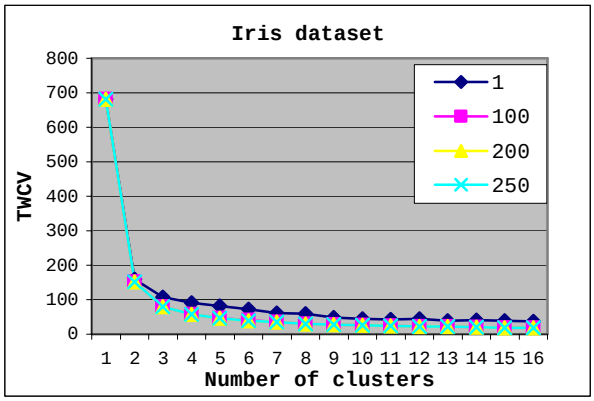


Figure 2: Pareto-fronts for IRIS dataset

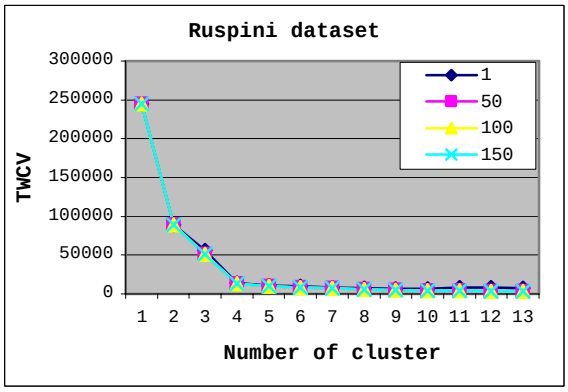


Figure 3: Pareto-fronts for RUSPINI dataset.

Table 1: IRIS DATASET TWCV FOR K=6

Iteration	TWCV
1	72.60164
50	40.4213
100	39.9218
150	39.6762
250	39.5762

Table 2: RUSPINI DATASET TWCV FOR K=12

Iteration	TWCV
1	8331.376
50	3555.116
100	3524.366
150	3513.254

We have run the proposed genetic algorithm based approach ten times with the following parameters: population size=100, t_dom (number of comparison set=10) and crossover= 0.8 and mutation=0.01 and we used 250, and 150 as the maximum number of generations for the Iris and Ruspini datasets, respectively. Finally, we picked the range [2, 20] for finding the optimal number of clusters for both experiments.

After running the algorithm for the Iris and Ruspini datasets, the changes in the pareto-optimal front are displayed in Figure 2, and Figure 3, respectively, for different generations; demonstrating how the system converges to an optimal pareto-optimal front. As the actual change in the value of TWVC is not reflected in the curves in Figure 2 and Figure 3, some key TWVC values are reported in Table 1 and Table 2, respectively.

After we get the pareto optimal front, we tested and analyzed the obtained results for the two data sets using the six indexes: Dunn index, Davies-Bouldin index, Silhouette index, C index, SD index and S_Dbw index. The results are reported in Figures 4-7. Finally, we compared our results with the corresponding results reported in [16, 17].

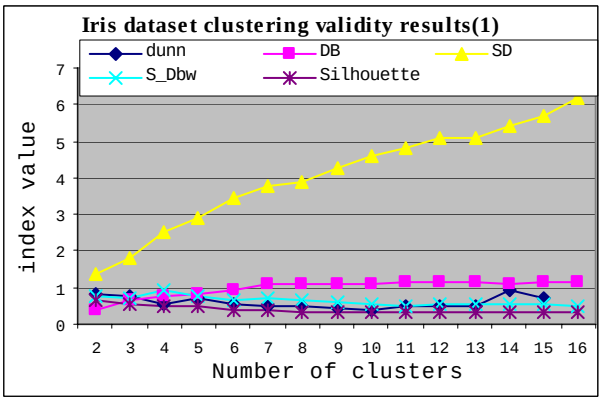


Figure 4: Validity results of five indexes for the IRIS dataset

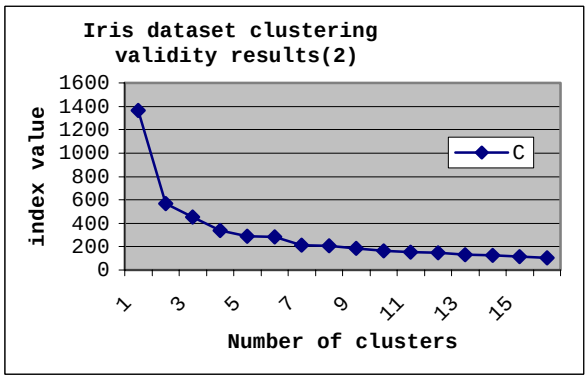


Figure 5: C-index validity results for the IRIS dataset

According to [17], the optimal number of clusters found for the Iris data is 3, which ranked the second for all the indexes except S-Dbw (see Figure 4 and Figure 5).

During the process, pareto-optimal front changes were given in Figure 2 and Figure 3. The reason for getting a stabilized front for both datasets is the k-means operator and the mutation which was also used in [1]. In the work described in [1], they used 517 instances with 19 attributes and their study found the stabilization in 20 generations. However, objective functions do not converge in the first 20 generations; the partitioning error

keeps reducing slightly because of the k-means and mutation. We make sure it converged by having objective functions without a change between the current and the previous one. In our multi-objective genetic algorithm based approach, we got the same outcome as in [1]. In other words, our method stabilizes in the first 50 generations but it does not converge until around 200 generations for Iris; and close to 150 generations for Ruspini. Because of the lack of space, we show the change for one cluster *k* (finally converging ones).

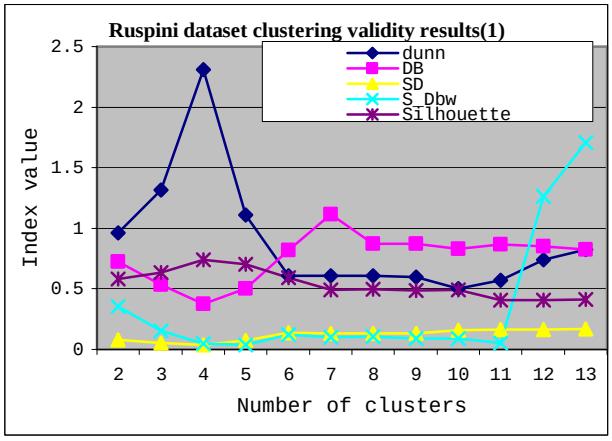


Figure 6: Validity results of five indexes for the RUSPINI dataset

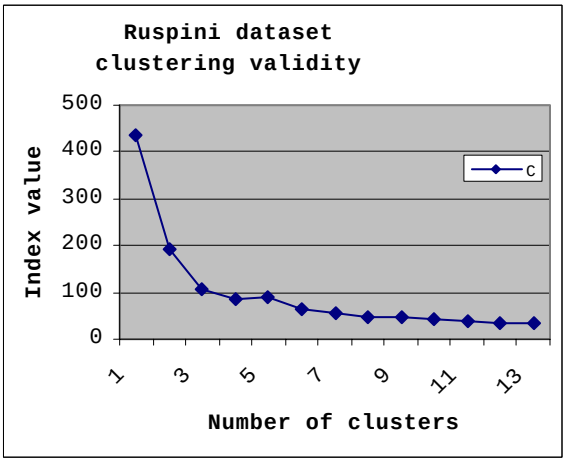


Figure 7: C-index validity results for the RUSPINI dataset

For the Ruspini dataset, it is naturally grouped into 4 clusters as reported in [16]. Not only we have 4 in our pareto optimal front as it can be easily seen from the curves plotted in Figure 6 and Figure 7, but also we got this value as the best in all the cluster validity analysis indexes except C-index. However, 4 can be considered as the best for the C-index as well if it is deemed slight changes after 4 has converged. This finding is consistent with the results found before. Actually, C index is likely to be data dependent and the behavior of the index may change when different data structures were used.

5 Conclusions

In this paper, we proposed a multi-objective genetic algorithm called MOKGA to handle the clustering problem. It is the combination of the niched pareto optimal and fast k-means genetic algorithm. In other words, in MOKGA both crossover and mutation operators are used for the evolutionary process, in addition to the *K*-means operator used to make the evolutionary process faster. For the selection, Niched Pareto tournament selection method is used. Additionally, a multiple Pareto-optimal front layer ranking method is proposed to maintain relative consistence population size in the genetic process. In the experiments, it is also verified that this method can help in leading to the global optimal solution set. In the MOKGA process, the distance (Euclidean distance) between the current generation's Pareto optimal front and the previous generation is calculated and counted compared with the threshold, which can be used to decide when to terminate the genetic process. This way, we overcome the difficulty of determining the weight of each objective function taking part in the fitness. Otherwise, the user would have been expected to do many trials with different weighting of objectives as in traditional genetic algorithms. By using MOKGA, we aim at finding the pareto-optimal front to help the user to see many alternative solutions at once. Then, cluster validity index values are evaluated for each pareto-optimal front value, which is considered the optimal *number of clusters* value.

References

[1] Y. Lu, et al, "FGKA: A Fast Genetic K-means Clustering Algorithm," *Proceedings of ACM Symposium on Applied Computing*, pp.162-163, Nicosia, Cyprus, 2004.

[2] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," *Proceedings of IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Computation*, Vol.1, pp.82-87, Piscataway, NJ., 1994.

[3] D.L. Davies and D.W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, No.1, pp.224-227, 1979.

[4] J. Dunn, Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, Vol.4, pp.95-104, 1974.

[5] P.J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,"

Journal of Comp App. Math, Vol.20, pp.53-65, 1987.

[6] L. Hubert and J. Schultz, "Quadratic assignment as a general data-analysis strategy," *British Journal of Mathematical and Statistical Psychologie*, Vol.29, pp.190-241, 1976.

[7] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 1998.

[8] M. Halkidi, M. Vazirgiannis and I. Batistakis, "Quality scheme assessment in the clustering process," *Proceedings of PKDD*, Lyon, France, 2000.

[9] M. Halkidi, M. Vazirgiannis, Clustering "Validity Assessment: Finding the optimal partitioning of a data set," *Proceedings of IEEE ICDM*, California, Nov. 2001.

[10] Gene Expression Data of the Genomic Resources, University of Stanford (Available at: http://genome-www.stanford.edu/ serum/data.html), accessed June 2004.

[11] J. Grabmeier, et al, "Techniques of Cluster Algorithms in Data Mining," *Data Mining and Knowledge Discovery*, Vol.6, pp.303–360, 2003.

[12] A. K. Jain, et al, "Data Clustering: A Review," *ACM Surveys*, Vol.31, No.3, 1999.

[13] K. Tamura, et al, "Necessary and Sufficient Conditions for Local and Global Non-Dominated Solutions in Decision Problems with Multi-objectives," *Journal of Optimization Theory and Applications*, Vol.27, 509-523, 1979.

[14] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," *Doctoral thesis ETH NO. 13398, Zurich: Swiss Federal Institute of Technology (ETH)*, Aachen, Germany: Shaker Verlag, 1999.

[15] K. Deb et al., "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," *Proceedings of the Parallel Problem Solving from Nature. Springer Lecture Notes in Computer Science No. 1917*, Paris, France, 2000.

[16] L. Kaufman and P.L. Rouseeauw, *Finding group in data: An introduction to cluster analysis*, John Wiley & Sons. New York, p.100, 1990.

[17] H.P. Friedman and J. Rubin, "On some invariant criteria for grouping data," *Journal of the American Statistical Association*, 62:1159-1178, 1967.

[18] Microarray Data Analysis: Direct Gene Sample Correlations, Gene Network Science, Inc. (c). 2001.

[19] M. Neef, D. Thierens, & H. Arciszewski, A Case Study of a Multi-objective Elitist Recombinative Genetic Algorithm with Coevolutionary Sharing. In Angeline, P. (Ed.), *Proc. of the International Congress on Evolutionary Computation*, pp.796-803. Priscatawy. 1999.

[20] W. Shannon, R. Culverhouse J. Duncan. Analyzing microarray data using cluster analysis, *Pharmacogenomics*, Vol.4, No.1, pp.41-52, 2003.

[21] T. Kohonen, Self-organizing Maps, Springer-Verlag, 1997.

[22] P.J. Waddell, H. Kishino, Cluster Inference Methods and Graphical Models Evaluated on NCI60 Microarray Gene Expression Data, *Genome Informatics*, Vol.11, pp.129-140, 2000.

Complexity-driven Evolution of Decision Graphs for Classification of Medical Data

Vili Podgorelec
Institute of Informatics, University of Maribor, FERI,
Smetanova 17, SI-2000 Maribor,
Slovenia
vili.podgorelec@uni-mb.si

Keywords: data mining, classification, meta agents, automatic programming, complexity, medical data

Received: October 27, 2004

In the paper we study the possibility of constructing decision graphs with the help of several meta agents. Decision graphs are an extension of the well known decision trees and introduce the possibility of program nodes and cycles in a classification model. A two-leveled evolutionary algorithm for the induction of decision graphs is presented and the principle of classification based on the decision graphs is described. Several agents are used to construct the decision graphs; they are constructed and evolved with the help of automatic programming and evaluated with a universal complexity measure. The developed model is applied to a medical dataset for the classification of patients with mitral valve prolapse syndrome.

Povzetek: Obravnavana je konstrukcija odločitvenih grafov s pomočjo metaagentov in njihova uporaba za klasifikacijo medicinskih podatkov.

1 Introduction

Making the right decision is becoming the key factor for the successful achievement of our goals in all areas of our work. The ways of finding the right decision are as many as the number of people who have to make them. Nevertheless, the basic idea is the same for many of them: a decision is usually made as a combination of experiences from solving similar cases, the results of recent researches and personal judgment. The number of solved cases and new researches is increasing rapidly. It could be expected that newly made decisions will become better and more reliable but for the individuals and groups who have to make decisions it is actually becoming more and more complicated, because they simply can not process the huge amounts of data anymore. And there the need for a good decision support technique arises. It should be able to process those huge amounts of data and to help experts to make their decisions easier and more reliably. For this purpose it is equally or even more important as suggesting the possible decision, to provide also an explanation of how and why the suggested decision was chosen. In this manner an expert can decide whether the suggested solution is appropriate or not.

As in many other areas, decisions play an important role also in medicine, especially in medical diagnostic processes. Decision support systems helping physicians are becoming a very important part in medical decision making, particularly in those situations where decision must be made effectively and reliably. Since conceptual simple decision making models with the possibility of automatic learning should be considered for performing such tasks, decision trees are a very suitable candidate. They have been already successfully used for many decision making purposes [Pod02].

1.1 Scope and contributions of the paper

The paper will introduce the concept of complexity-driven evolution of meta agents in classification. First the related research will be presented and the most important basic concepts defined. Then the paper will focus on agents learning with genetic programming. The concept of decision graphs will be introduced, which are an extension of decision trees. The process of agent evolution is presented with the emphasis on the complexity-based fitness function that is used to evaluate individual solutions. In the final section the application of our approach to the problem of mitral valve prolapse is shown. In the concluding section the advantages and the drawbacks of the method are presented and some future plans outlined.

The main contributions of the paper are:

- agent-based construction of decision graphs, and
- complexity-driven evolution of meta agents.

2 Related research

In today's world there is a pressing need to automate common administrative tasks in order to lower costs, minimize time spent and increase productivity. To help accommodate far-reaching lifestyle changes, new tools are needed to support imperatives of this rapidly changing environment - intelligent agents. Intelligent

agents are software programs that have the ability to act autonomously on their user's behalf, learn from experience and collaborate with other agents to achieve a common goal [Woo95]. They help their users with routine computer tasks, while still accommodating individual habits.

2.1 Intelligent agents

Intelligent agents are software programs that can identify repetitive patterns of behavior, similarities between events or things, and changes in patterns over time.

A formal specification of agents must include the representation of the following aspects:

- knowledge – the beliefs that an agent has (the information about the environment),
- engine – the actions that an agent performs and the effects of these actions,
- adapters – "ears, eyes and hands" of an agent that allow it to communicate with the environment and also with other agents over time and take the appropriate actions, and
- the goals that an agent will try to achieve.

The salient feature of agents is their adaptability and capacity for learning otherwise they are just ordinary programs. Therefore we can define an intelligent agent as an entity that owns the following properties [Woo95]:

- *autonomy*: agents encapsulate some state, and make decisions on the basis of that state without the direct human intervention;
- *reactivity*: agents are situated in certain environment, and are able to perceive changes in that environment (through the implemented sensors). They are also able to respond to changes in timely fashion;
- *pro-activity*: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative;
- *social ability*: agents interact with other agents (and possibly humans) via some kind of agent-communication language, and typically have the ability to engage in social activities (such as cooperative problem solving or negotiation) in order to achieve their goals.

Agents are usually developed to provide expertise in a specific area and can, through cooperative work, jointly accomplish larger and more complex tasks. Autonomous agents have the ability to handle user-defined tasks independent of the user and often without the user's guidance or presence. Learning agents have the ability to learn user's habits through observation, user feedback or training. Reasoning capability is very important for agents to operate in a decision-making capacity in complex, changing environment.

2.2 Mobile agents

Agents can be mobile by moving from machine to machine (form site to site). A mobile agent (MA) has a long-term memory (a suitcase) that includes all sites, which the MA visited, actions performed on each sites and the results of these actions. When it enters a new site a MA receives information about the site that might be useful for further decision-making (a briefing), such as local file system and databases. The suitcase and the briefing provide all the data needs of the agent [Bha96]. The agent server controls the migration of a MA to a new system by controlling a security and authentication of the MA, briefing the agent as it enters the system and executing the agent code [Bha96]. The MA uses hop instruction to move from one site to another.

Agent based systems are becoming one of the most important computer technologies, holding out many promises for solving real-world problems. An agent-based system may contain a single agent but the greatest potential lies in the application of multi-agent systems.

2.3 Multi-agent systems

While problems are often too complex to be solved by one intelligent agent the development is tending toward using multi-agent systems (MAS). MAS are agent groups where each component of intelligent behavior is delegated to a separate agent. Several agents that exist at the same time, share common resources and communicate with each other [Fer99]. MAS simplify problem solving by dividing the necessary knowledge into subunits to which an independent intelligent agent is associated and therefore every independent agent has the ability to solve a specific problem. The problem also has to bee discrete so that it can be divided into independent sub-problems. Individual agents are than assigned to solve a specific sub-problem. A partial global plan has to be created for tasks definitions.

2.4 Meta agents

Meta agents are a way to help agents observe the environment, evaluate alternatives and prescribe and schedule actions. In addition, strategies can be formulated and implemented not only within an agent but also among a group of agents. For any given problem, various strategies may be available. The main role of the meta agent is to sift through these strategies and make intelligent decisions. Eventually each agent will consult the meta agent prior to performing analyses; the agent will state the desired analyses and the utility of performing them. Subsequently, the meta agent will determine the feasibility of performing the actions by considering timing requirements and resource constraints. In addition, extra resources may be requested from the resource manager.

There are numerous applications where an agent needs to reason about the beliefs of another agent, as well as about the actions that other agents may take. Eiter [Eit99a] presents the concept of an agent program, and a language within which the operating principles of an agent can be declaratively encoded on top of imperative data structures is defined. In [Dix00] a certain belief data structures that an agent needs to maintain are introduced.

That extends the framework [Eit99b] so as to allow agents to perform meta reasoning.

In another work Stone discussed about the meta agent decisions on a strategy [Sto97]. It determines what behaviors of the agents would achieve this strategy and accordingly triggers those behaviors. By triggering only the behaviors appropriate to the current strategy, the meta agent also reduces behavior-behavior interactions. This is also in accordance with the layered architecture proposed in the paper. Higher level layers implement more abstract behaviors by selecting and activating the appropriate behaviors from the next level. The higher levels provide the strategic reasoning while the lower level provides reactivity to the system. This leads to the two most important questions the meta agent needs to answer:

- How to choose an appropriate strategy?
- How to characterize agent behaviors?

To allow coexistence of multiple agents Lakshmikumar proposed framework of the development of a reusable meta agent that manages these agents [Lak01]. The goals of this meta agent are:

- perform reasoning (to reason about agent autonomy): this is going to decide how much cooperation there needs to be between the agents;
- planning: plan actions;
- individual agent modeler: maintain individual agent state information;
- other agent modeler: maintain information on other agents and attempt to predict future behavior;
- conflict manager: classify conflicts and resolve them.

2.5 Use of agent systems in medicine

Agent systems and MAS are wide spread in all areas of user applications such as telecommunications, Internet, health care, tutoring systems, management systems, ecology, etc. They have also been useful in medicine [And00]. We will briefly highlight a few of these projects:

G. Lanzura, L. et al. [Lan99] at the University of Pavia, illustrated a methodology facilitating the development of interoperable intelligent software agents for medical applications and proposed a generic computational model for implementing them. That model may be specialized in order to support all the different information and knowledge related requirements of a Hospital Information System.

L.M. Camarinha-Matos and W. Vieira [Cam99] proposed an inexpensive support system for elderly people staying alone at home, allowing care and health centres to remotely observe and help them. It is based on the Internet and uses the multi-agent systems paradigm that includes both stationary and mobile agents.

M. Gnoth and I. Münich [Gno99] described the ChariTime project for the distributed scheduling of diagnostic and therapeutic appointments, based on multi-agent systems.

A. Boucher et.al. [Bou98] presented a multi-agent model for the analysis of living cells. The system is used particularly to study cell migration, for example the migration of tumor cells in response to treatment with antineoplastic drugs.

R. Freitas Jr. [Fre99] described medical nanorobots with tiny sensors and medical devices that will be capable of performing delicate, fine-grained operations within the human body.

An agent-based tutoring system for students of medicine was evolved at University of Southern California by Ganeshan et.al.[Gan00]

An agent-based approach to facilitate cooperative medical diagnosis was evolved at University College Galway in Ireland [Mul98]. It is achieved through monitoring patient record construction and by highlighting relevant diagnosis information.

3 Learning and intelligent agents

The machine learning community has paid increasing attention to problems of delayed reinforcement learning [Jaa94, Mca95]. These problems usually involve an agent that has to make a sequence of decisions, or actions, in an environment that provides feedback about those decisions. The basic loop followed in sequential decision making tasks such as these includes evaluating the current state, taking an action, and computing the new state. This loop is repeated until the system either reaches a goal state or recognizes that it will never terminate.

Research in multiple agent planning and control has been limited largely to the area of distributed artificial intelligence [Sto96] and artificial life [Dor96]. In distributed AI (DAI), several agents cooperate to achieve some goal or accomplish some task. The task is usually one of sufficient complexity that no single agent can accomplish the task alone. Because the agents cooperate, research in distributed AI has focused primarily on developing efficient procedures for communicating between the agents to enable the agents to develop the cooperative plans.

Although artificial life research does explore issues related to both cooperation and competition, its primary focus is on the emergence of intelligent behavior in a population of agents. For example, one area of application that has received considerable attention is the evolution of foraging behavior among artificial organisms (e.g., artificial ants) in the presence of predators. Also, migration patterns of artificial birds have been evolved. In none of these cases has behavior of individual agents been the focus of the research.

Recently, work has begun to appear that focuses on learning in MAS. Stone and Veloso provide a taxonomy of MAS by focusing on attributes such as agent homogeneity, communication, deliberative versus reactive control, and number of agents [Sto96]. Problems in MAS are distinct from problems in DAI and distributed computing, from which the field was derived, in that DAI and distributed computing focus on information processing and MAS focus on behavior development and behavior management. In addition, problems in MAS are distinct from problems in artificial life in that MAS still focus on individual behaviors and

artificial life focuses on population dynamics. So far, most work in learning and MAS has focused on multiple agents' learning complementary behaviors in a coordinated environment to accomplish some task, such as team game playing [Tam96], combinatorial optimization [Dor96], and obstacle avoidance [Gre91].

3.1 Learning agents with GP

Genetic programming (GP) and its variants have been applied to multi-agent learning. For instance, Koza used GP to evolve sets of seemingly simple rules that exhibit an emergent behavior. The goal was to genetically breed a common computer program, when simultaneously executed by all the individuals in a group of independent agent, i.e., the homogeneous breeding, that causes the emergence of beneficial and interesting higher-level collective behavior [Koz92].

Haynes proposed an approach to the construction of cooperation strategies based on GP for a group of agents [Hay95]. He experimented in the predator-prey domain, i.e., the pursuit game, and showed that the GP paradigm could be effectively used to generate apparently complex cooperation strategies without any deep domain knowledge.

Iba has applied GP-based multi-agent learning to the Tile World and proposed a co-evolutionary breeding scheme [Iba96]. Experimental results have shown the superiority of the co-evolutionary breeding over the two strategies, i.e., the homogeneous strategy and the heterogeneous strategy. In the co-evolutionary strategy, some individuals were expected to perform specialized tasks for different agents with generations.

4 Constructing decision graphs

Decision graph is an extension of a very well known decision tree representation [Qui93, Bre84, Pod02]. Similar to decision trees a decision graph contains attribute and decision nodes, where attribute nodes contain some kind of test of attributes' values and decision nodes serves to predict the solution (Figure 1, Figure 2). However, the decision graph principle is more flexible and more general than a decision tree. Since it contains also cycles, additional internal variables (different from attributes) can be added that help to process a temporal information, i.e., input can be represented in a time-series manner, which makes the decision graphs especially appropriate to deal with signals and continuous data. Decision trees are of course not able to process those kind of data.

A node in a decision graph contains a kind of transition rule that tells what edge to follow in a decision making process, based on the test of attributes' values and/or the state of internal variables. Transition rules can be very simple (as in decision trees) or more complex (each node contains a program). Since we decided to construct a decision graph with the help of evolving agents, the rules can not be too complex in order to maintain a simplicity of each of the participating agents. Therefore the rules are simple `if..then` statements, where the condition is a single attribute test or a single internal variable test. When composing two nodes (as a consequence of the JOIN agent) those simple statements are combined in a composed `if..then` statement. An example of a composed transition rule is presented on Figure 2.

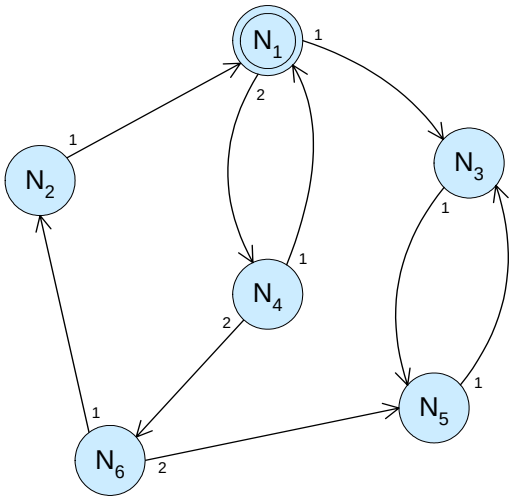


Figure 1. A simple decision graph. Every node contains a transition rule (Figure 2) that serves both as a test and/or as a decision class prediction. All edges from a single node are numbered, the numbers determine the next node based on the transition rule.

```
if (A3 > split) then
   if (INV2 < TRESHOLD2) then
      moveTo(3)
   else
      decision(CLASS1)
   endif
else
   moveTo(1)
endif
inc(INV2)
moveTo(1)
```

Figure 2. A composed transition rule. A_x is attribute `x`, `split` is a testing split, INV_x is internal variable, $THRESHOLD_x$ is a testing value for an internal variable, `moveTo(x)` indicates the transition to node `x`, $decision(CLASS_x)$ indicates the prediction of the decision class `x`, and $inc(INV_x)$ indicates the increase of the internal variable `x` by 1.

4.1 Two-leveled evolution process

The outline of the decision graph construction algorithm can be best described as a two-leveled evolution process. At the lower level decision graphs are being evolved. The evolution at this level starts by constructing an initial population of random decision graphs. For this purpose a random amount of decision nodes is created (a transition rule is initialized) which are then randomly connected with edges. In the continuation of the evolution at this

lower level in each generation participating agents modify the decision graphs. The quality of a decision graph is evaluated based on the accuracy of classification of training objects.

Naturally, because the construction of initial decision graphs is random, the classification accuracy in the early stages of evolution is low. Therefore, the quality of participating agents is essential in order to improve the predicting capabilities of the decision graphs. In this manner, the participating agents should improve to produce good results. To achieve the quality improvement in agents, they are also being evolved – and that is the second, the higher level of our global process. Each participating agent is evolved independently from the others by automatic programming approach with the *proGenesys* system. The quality of agents is not evaluated explicitly, but rather an universal complexity measure α is used that implicitly drives the agents to higher complexities.

In the Figure 3 a pseudo-code procedure for the two-leveled evolution process is presented.

```
initialize_agents()
repeat
   evolve_next_generations_of_agents()
   initialize_decision_graph()
   repeat
      apply_agents_to_modify_decision_graph()
      evaluate_decision_graph()
   until (num_generations > MAX_GENERATIONS)
   remember_the_best_decision_graph()
until (solution does not improve)
```

Figure 3. A pseudo code of the two-leveled evolution process of decision graphs construction. The inner `repeat..until` loop represents the lower level of the evolution process that changes decision graphs and the outer `repeat..until` loop represents the higher level of the evolution process that changes the agents.

4.2 Participating agents

Seven different agents are used in the process of decision graph evolution. We named those agents as: ADD, DELETE, MUTATE, JOIN, DISJOIN, PROTECT, and UNPROTECT. Each agent has its own function and works on the evolving decision graph independently from the other agents, according to its own procedure, defined by the outcome of genetic programming process in the current generation. In this way the decision graph is modified by those agents in order to become as accurate in classifying the training objects as possible. The functions of the participating agents are the following:

1. each ADD agent adds with certain probability: 1) a node (creates new transition rule for the new node), or 2) an edge (renumbering the existing connections);
2. each DELETE agent deletes with certain probability: 1) an edge (renumbering the remaining connections), or 2) a node (renumbering the connections of the connected nodes);
3. each MUTATE agent changes transition rule in a node with certain probability: 1) an attribute, 2) a split value, 3) an internal variable, 4) a threshold value for internal variables, 5) transition value, or 6) predicted decision;
4. each JOIN agent merges two selected nodes with certain probability, adjusting the transition rule and renumbering the new connections;
5. each DISJOIN agent separates a composed node into two connected nodes with certain probability, distributing the existing edges to either one new node or another;
6. each PROTECT agent protects with certain probability: 1) a node, and/or 2) an edge either against deletion, mutation, joining and/or disjoining;
7. each UNPROTECT agent unprotects with certain probability a protected: 1) node, and/or 2) edge;

5 Evolution of agents

All the agents are evolved with the use of automatic programming, a genetic programming technique for evolving programs in an arbitrary programming language, described with a context-free grammar. For this purpose we have used our evolutionary program generation tool called *proGenesys* [Pod99].

5.1 The kernel of *proGenesys*

The aim of the *proGenesys* tool is automatic generation of program code. We used genetic programming as the underlying principle of program generation. Generation of initial programs and basic evolutionary processes are quite similar, the most important difference represents the evaluation function that we used to determine the fitness of each individual. Since our intentions are to generate optimal programs performing some very complex task, we don't exactly evaluate evolved programs but rather use an universal complexity measure, namely the software complexity metrics α that is described later in the paper.

5.1.1 Generation of initial population

The first phase of genetic process is the generation of an initial population. Enough individuals have to be constructed to fulfill the whole population. Since later evolution depends quite a lot on initial population (especially its diversity), a great care was taken to implement a method for the construction of an individual.

For the construction of randomly generated programs, slightly modified Backus-Naur form (BNF) of programming language is used with some meta-symbols added, defining probabilities of transitions into specific branches of BNF structure, setting maximum recursion level of non-terminals extension, limiting the complexity of different program blocks, like expressions, etc. It is

important that the construction can start from within any BNF production, since it is also used later when mutation operator is applied. In this manner any BNF substructure can be generated when needed, like programming sentence, expression, etc.

An individual is internally represented as a syntax or derivation tree of a generated program (Figure 4). The syntax tree contains not only a program code but also a complete information on how this code was constructed from the starting symbol of the programming language's BNF. There are two types of nodes in a syntax tree. Internal nodes represent non-terminal symbols of BNF and show how each production was expanded to form the program. External or leaf nodes represent terminal symbols of BNF which actually construct the resulting program code. In this way program code can be extracted easily and syntax information is preserved throughout the evolution, which makes it easier to develop appropriate genetic operators.

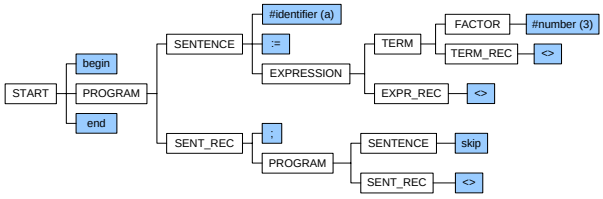


Figure 4. An example of a generated individual – a syntax tree.

5.1.2 Selection and fitness function

For the selection scheme we used a slightly modified exponential ranking selection method. After the evaluation of all individuals, they are sorted accordingly to their fitness score. Then we replace existing individuals from the worst to the best by creating new ones with crossover from two selected individuals, that still exist from the old population. When all the individuals are replaced, the new population is generated (there is still mutation to be applied).

For effective selection we have to define an adequate evaluation function, that determines the fitness score of each individual. This is the point where our approach differs the most from the other genetic programming applications. We don't try to exactly evaluate evolved programs, but rather use an universal complexity measure - our software complexity metric α. In this way individuals are evolved to very complex programs which are eventually evaluated through their performance upon decision graphs by measuring the effectiveness of the decision graphs in classifying the training objects.

5.1.3 Crossover

As the two individuals are selected from within the current population, a new solution is constructed by applying the genetic operator of crossover (Figure 5) and the constructed individual is placed in a growing new population. In order to perform a crossover operation, an appropriate crossover point has to be determined. For this purpose, a set of non-terminal symbols, contained in both selected individuals (parents), is computed and one of those symbols is chosen randomly. Then it is looked for such a node in both parent trees and offspring is created by concatenating a branch (a subtree) from the chosen node in second parent to the chosen node in first parent (see Figure 5). In this way the syntax correctness is preserved, since there is only a possible extension of BNF production replaced with another and the whole is still a correct program (considering that both parents were correct, what is actually the case, since the program generation algorithm guarantees only correct programs to be generated).

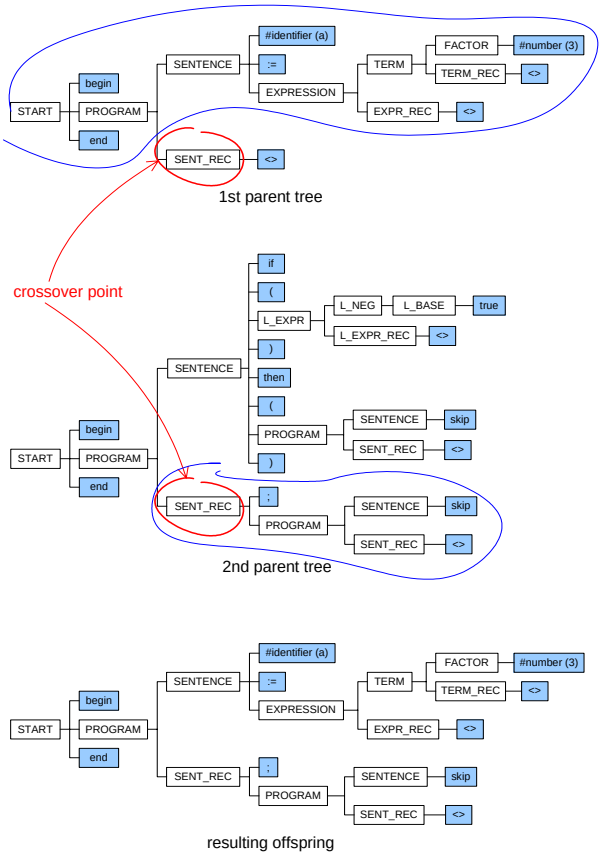


Figure 5. An example of a crossover. Circled parts of both syntax trees are combined into one offspring program.

5.1.4 Mutation

After a new individual is constructed by crossover, a genetic operator of mutation is applied (Figure 6) with certain probability. Mutation serves as a random change of an existing.

First the mutation point is randomly chosen in a given syntax tree. According to the selected node there are two possible situations. First, if an internal node was selected, representing a non-terminal symbol of BNF. In this case the existing extension of BNF production is replaced with a newly generated derivation. That is why

we mentioned the importance of a program generation algorithm to start with any BNF production, since here we send the chosen non-terminal symbol (from mutation point) and expect the algorithm to generate an adequate portion of a program code that is concatenated to the selected tree node instead of the existing part.

Second, if a special kind of external or leaf node was selected, representing a categorized terminal symbol of BNF. Such a categorized terminal is a number (category `#Number`) for example. In this case a new terminal is constructed so that it fits into specific category (for example, a new number is randomly chosen, replacing the existing one).

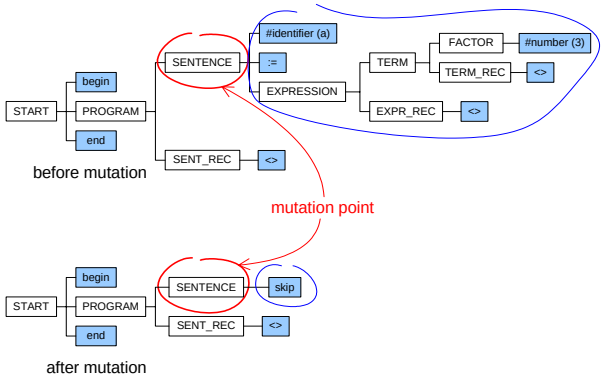


Figure 6. An example of a mutation. Circled part of first tree is mutated into the circled part of second tree.

5.2 Software complexity measure α

Many quantities have been proposed as measures of complexity. Gell-Man [Gel95] suggests there have to be many different measures to capture all our intuitive ideas about what is meant by complexity. Some of the quantities are computational complexity, information content, algorithmic information content, the length of a concise description of a set of the entity's regularities, logical depth, etc. (in contemplating various phenomena we frequently have to distinguish between effective complexity and logical depth - for example some very complex behavior patterns can be generated from very simple formula like Mandelbrot's fractal set, energy levels of atomic nuclei, the unified quantum theory, etc. - that means that they have little effective complexity and great logical depth). A more concrete measure of complexity, based on the generalization of the entropy, is correlation [Sch93], which can be relatively easy to calculate for a special kind of systems, namely the systems which can be represented as strings of symbols.

Computer programs are conventionally analyzed using the computational complexity or measured using complexity metrics. Another way to asses complexity is to use fractal metrics [Kok96, Kok99] or entropy based measure [Har89]. However, we can regard computer programs from the viewpoint of "complexity as a discipline" and according to that apply various possible complexity measures presented above. The fact that a computer program is a string of symbols, introduces an elegant method to asses the complexity – namely to calculate long range correlations between symbols, an approach which has been successfully used in the DNA decoding [Bul94] and on human writings [Sch93].

Our fractal measure α is based on char method, which is an extension of the method originally proposed by Schenkel [Sch93] for human writings. Like a human writing, a computer program can be seen as a string of symbols: letters, digits and some delimiting symbols – empty spaces are ignored. Using code table, where each of these symbols is represented by a binary sequence, the program is transformed into Brownian motion model (0's → step down, 1's → step up, see Figure 7), a base for the calculation of regression function *F(l)*:

$$F^2(l) \equiv \overline{[\Delta y(l)]^2} - \overline{\Delta y(l)}^2$$

$$\Delta y(l) = y(l_0 + l) - y(l)$$

where *Δy(l)* is relative difference between two points in Brownian motion model (Figure 7). The coefficient α is then calculated with the least squares method as the linear representation of the points on a double logarithmic scale [*ln (l), ln (F(l))*] and represents the complexity of a computer program (Figure 8).

According to above definition regression points [l_0, $F(l_0)$] are calculated from Brownian motion as follows:

$$F(l_0) = \sqrt{\frac{\sum_{l=1}^{S-l_0} \left(y(l+l_0) - y(l)\right)^2}{S-l} - \left(\frac{\sum_{l=1}^{S-l_0} y(l+l_0) - y(l)}{S-l}\right)^2}, \forall l_0 \in [1..\frac{S}{2}]$$

where *S* is the number of points in Brownian motion plot.

After the regression points are calculated, the coefficient α is then calculated with the least squares line as the linear representation of the points on a double logarithmic scale [*ln (l), ln (F(l))*]. As line crosses the axis at [*0, 0*] the line equation

$$y = a + bx$$

becomes simpler

$$y = bx$$

where *b* actually represents α.

From the method of least squares, *b* (or α in our case) is calculated as

$$b = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

Regarding the regression points [*ln (l), ln (F(l))*] the α is thus calculated as

$$\alpha = \frac{\frac{S}{2}\sum_{i=1}^{\frac{S}{2}} \ln(i)\cdot\ln(F(i)) - \left(\sum_{i=1}^{\frac{S}{2}} \ln(i)\right)\left(\sum_{i=1}^{\frac{S}{2}} F(i)\right)}{\frac{S}{2}\sum_{i=1}^{\frac{S}{2}} (\ln(i))^2 - \left(\sum_{i=1}^{\frac{S}{2}} \ln(i)\right)^2}$$

To better understand the process of calculating α the Figure 7 shows how the relative difference *Δy(l)* between two points in a Brownian motion model is calculated, and the Figure 8 shows the graphical representation of α on a double logarithmic scale of regression points [*ln (l), ln (F(l))*].

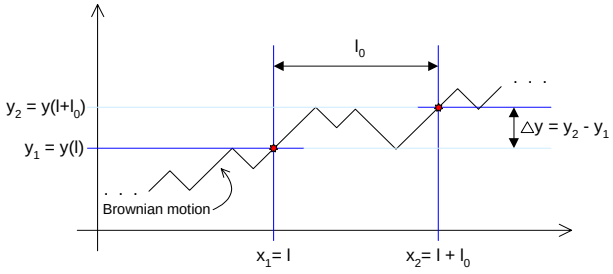


Figure 7. Calculation of regression curve points [*l, F(l)*] from a Brownian model plot.

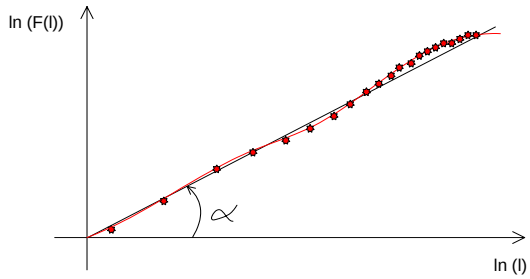


Figure 8. Representation of α on the regression curve points [*ln(l), ln(F(l))*] plot.

5.3 Programs for agents

In order to evolve the agents in accordance with our goal – to improve the classification capabilities of a decision graph – agents should be run by appropriate programs. The more obvious way to achieve this goal would be to define the programs for all the agents and let them work on the decision graphs. In this manner no second (higher) level of evolution would be necessary and the system would work faster. But of course, defining the agents is not a trivial task, and also we do not know what are the optimal programs for all the agents. Therefore, we decided to evolve the agents from the scratch with the use of the described *proGenesys* system.

For this purpose a language has to be defined first (a set of terminals, non-terminals and BNF productions) for each agent. We decided to keep the languages as simple as possible and therefore each language contains only few function calls to pre-defined functions (like `join()` for JOIN agent, `deleteNode()` or `deleteEdge()` for DELETE agent, etc.), simple condition statements (`if..then..else`, etc.) and simple expressions. A program for each agent is then interpreted to modify the decision graph in the lower level of evolution. An example of such a program for DELETE agent is presented in Figure 9.

```
if (random_condition == true) then
   node = selectNode()
   deleteNode(node)
else
   edge = selectEdge()
   deleteEdge(edge)
endif
```

Figure 9. An example of a simple program for the DELETE agent.

6 Application of the method and results

First we tested the performance of proposed classification method for training data and test data by the well-known iris data set [Fis36], which is not very complex. The iris data consists of 150 objects described by 4 continuous attributes and has three possible outcomes: Iris-setosa, Iris-versicolor, Iris-virginica. We selected 117 objects for training and the remaining 33 objects for testing. The average results over 5 runs are presented in Table 1.

Table 1. The results (accuracy) of iris data classification by evolved decision graphs (average over 5 runs).

	training data	test data
accuracy	96.58	93.94

An evolved decision graph for the classification of iris data is presented in Figure 10. It consists of 4 nodes, of which two contains classification of decision class. There is one cycle and two internal variables are used: one is a dummy and the other (`INV1`) actually plays an important role in classification process. It is interesting that there is only one decision statement for each class, what means that a classification for each class is made exactly once.

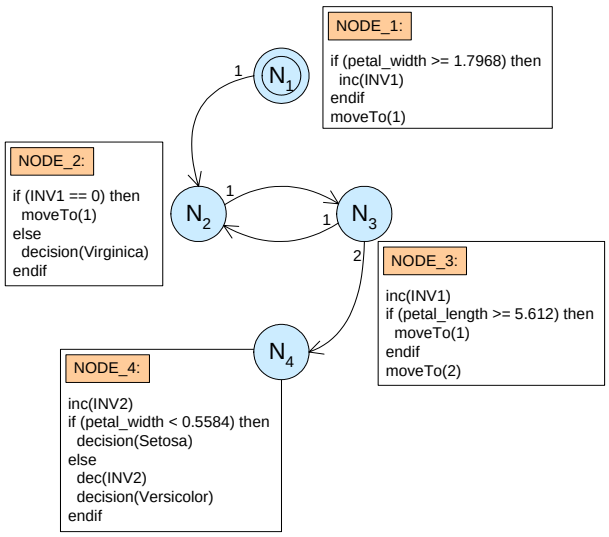


Figure 10. Evolved decision graph for the classification of iris dataset.

6.1 Mitral valve prolapse dataset

Because of the good results obtained for the iris data set, we decided to test a real-world medical problem of classifying mitral valve prolapse syndrome. Prolapse is defined as the displacement of a bodily part from its normal position. The term mitral valve prolapse (MVP) [And91, Dev89, Mar76], therefore, implies that the mitral leaflets are displaced relative to some structure, generally taken to be the mitral annulus. The silent prolapse is the prolapse which can not be heard with the auscultation diagnosis and is especially hard to diagnose. The implications of the MVP are the following: disturbed normal laminar blood flow, turbulence of the blood flow, injury of the chordae tendinae, the possibility of thrombus' composition, bacterial endocarditis and finally hemodynamic changes defined as mitral insufficiency and mitral regurgitation.

MVP is one of the most prevalent cardiac conditions, which may affect up to five to ten percent of normal population and one of the most controversial one. The commonest cause is probably myxomatous change in the connective tissue of the valvar liflets that makes them excessively pliable and allows them to prolapse into the left atrium during ventricular systole. The clinical manifestations of the Syndrome are multiple. The great majority of patients are asymptomatic. Other patients, however may present atypical chest-pain or supraventricular tachyarrhythmyas. Rarely, patients develop significant mitral regurgitation and, as with any valvar lesions, bacterial andocarditis is a risk.

Uncertainty persists about how it should be diagnosed and about its clinical importance. Historically, MVP was first recognized by auscultation of mid systolic "click" and late systolic murmur, and its presence is still usually suggested by auscultatory findings. However, the recognition of the variability of the auscultatory findings and of the high level of skill needed to perform such an examination has prompted a search for reliable laboratory methods of diagnosis. M-mod echocardiography and 2D echocardiography have played an important part in the diagnosis of mitral valve prolapse because of the comprehensive information they provide about the structure and function of the mitral valve.

Medical experts propose [And91, Mar76] that echocardiography enables properly trained experts armed with proper criteria to evaluate MVP almost 100%. Unfortunately however, there are some problems concerned with the use of echocardiography. The first problem is that current MVP evaluation criteria are not strict enough [Kok94]. The second problem is the incidence of the MVP in the general population and the unavailability of the expensive ECHO - machines to general practitioners. According to above problems we have decided to develop a decision support system enabling the general practitioner to evaluate the MVP using conventional methods and to identify potential patients from the general population.

6.2 Classification results and discussion

Using the Monte Carlo sampling method 900 children and adolescents representing the whole population under eighteen years of life have been selected. All of them were born in Maribor region and all were white. Routinely they were called for an echocardiography no matter of prior findings. From 900 selected 631 volunteers were successfully examined.

They all passed an examination of their health state in a form of a carefully prepared protocol specially made for the Syndrome of MVP. The protocol consisted of general data, mothers health, fathers health, pregnancy, delivery, post-natal period, injuries of chest or any other kind, chronic diseases, sports, physical examination, subjective difficulties like headaches, chest-pain, palpitation, perspiring, dizziness etc., auscultation, phonocardiography, ECG and finally ECHO. In that manner, 103 parameters were gathered that can possibly indicate the presence of MVP.

All 631 patient records were randomly divided into a training and a testing set. The average results over 10 runs, as obtained with the described evolutionary method, are presented in Table 2.

For the sake of comparison, we induced a traditional decision tree with C4.5 algorithm [Qui93]. It scored the following results for the test data set: accuracy 90.00%, sensitivity 63.64%, and specificity 91.60% (see Table 2).

Table 2. The results of MVP classification by evolved decision graphs (average over 10 runs) and C4.5.

	training data		test data	
	graph	C4.5	graph	C4.5
accuracy	92.21	94.21	86.92	90.00
sensitivity	94.83	54.24	72.73	63.64
specificity	91.87	96.3	88.24	91.60

Regarding the accuracy of classification our results are a bit worse than those obtained by classical decision tree induction method, both on the training and the test data. Also the specificity (percentage of correctly classified negatives, i.e. patients with no prolapse in our case) is better with C4.5 decision tree. On the other hand, the sensitivity (percentage of correctly classified positives, i.e. patients with prolapse our case) is better with our method. Because the number of positives is much smaller than negatives, it could be concluded that the decision graph produced by our method is more appropriate for classifying unbalanced data sets, which is very common in medicine.

Furthermore, decision trees induction methods like C4.5 are able to generate tree-like structures with their limited capabilities. Contrary our approach generates graphs, which have in medical environment a lot of advantages, like:

- classifying the cycle: diagnosis → treatment → outcome,
- revealing the relations between diagnosis, treatment and outcome,
- classification of temporal data like EEG, ECG, EMG, etc.

On the other hand, the use of our system (at least in current stage) is not as easy to use as C4.5 for example. There is some "overhead" needed to set up the method for a new classification task (like adaptation of programs induction, evolution of agents). Furthermore, the amount of computational resources is much higher than in a classical decision tree induction method, several runs are needed to evolve the proper agents and decision graphs. Finally, one further drawback of our method is the interpretability of the mined knowledge; because the nodes in our decision graphs are more complex, it is more difficult to interpret the decision graph model than the decision tree. However, the results are not a black box (as in the case of neural networks for example) and still allow an expert to validate them.

Regarding both the advantages and the drawbacks of our method, it can be concluded, that it is appropriate for difficult, unbalanced datasets, where even the smallest improvement in results is worth the higher effort in achieving this improvement. This is certainly the case in medicine, where human health and welfare is in question.

7 Conclusion

In the paper a new approach to the classification of medical data based on the meta agents system for the construction of decision graphs is presented. We applied it to the prediction of MVP, but being a general-purpose classification model it can be used for different kinds of classification tasks. Some reasons in favor of using a complexity-driven evolution of agents have been stated. The whole two-leveled evolution process of decision graphs construction is described and also the kernel of the *proGenesys* tool for automatic evolution of agent programs is described. Results of MVP classification by constructed decision graphs are compared with those obtained by traditionally constructed decision trees.

There are two essential contributions of the paper. The first one is the flexible decision graph approach to the classification, that is an extension of the well-known decision tree classifier. The second one is the complexity-driven evolution of agents, that can replace the explicit evaluation of individuals in the process of programs evolution. In this manner, the need for the definition of an appropriate fitness function is avoided.

An obvious drawback of our approach, when applied to a real-world problem, is somewhat lower classification accuracy (when compared to the decision trees), and especially the lower interpretability of the mined knowledge. Additionally, the proposed classification approach is more difficult to use than the majority of the known ones. On the other hand, there are important advantages, like good classification of unbalanced data sets, flexibility of the knowledge model, novelty of the complexity-driven approach to the evolution of agents, as stated in the application section of this paper.

In future we plan to reduce the effort needed to set up the described method for a new classification task. Another aspect that we want to explore is to transfer the implementation onto a grid system; greater computational power of a grid would increase the possibilities of evolution process – in this manner we hope to further improve the overall effectiveness and quality of the obtained results. If resources allow, we want to further develop the proposed concept.

References

[And91] Anderson HR et al, *Clinicians Illustrated Dictionary of Cardiology*, Science Press, London, 1991.

[And00] Andonyadis CG, *A Hybrid Architecture for Web-Based Personal Healthcare Support Agents*, PHD thesis, George Washington University, 2000.

[Bha96] Bharat K, Cardelli L, Migratory Applications, *Mobile Object Systems Toward the Programmable Internet: Second International Workshop*, pp. 131-148, Springer, 1997.

[Bou98] Boucher A, Doisy A, Ronot X, Garbay C, A society of goal-oriented agents for the analysis of living cells, *Artificial Intelligence in Medicine,* 4(1-2), pp. 183-199, 1998.

[Bre84] Breiman L, Friedman JH, Olsen RA, Stone CJ, *Classification and regression trees*, Wadsworth, USA, 1984.

[Bul94] Buldyrev SV et al., Fractals in Biology and Medicine: From DNA to the Heartbeat, *Fractals in Science* (Bundle A, Havlin S, eds.), Springer Verlag, 1994.

[Cam99]Camarinha-Matos LM, Vieira W, Intelligent mobile agents in elderly care**,** *Robotics and Autonomous Systems*, 27(1-2), pp. 59-75, 1999.

[Chu95] Chu-Caroll J, Carberry S, Communicating for Conflict Resolution in Multi-agent Collaborative Planning, in *Proc. of the International Conference on Multi-Agent Systems ICMAS'95*, 1995.

[Dec87] Decker KS, Distributed Problem Solving: A Survey, *IEEE Transactions on Systems, Man, and Cybernetics*, 17, pp. 729-740, 1987.

[Dev89] Devereoux R, Diagnosis and Prognosis of Mitral Valve Prolaps, *The New England Journal of Medicine*, 320(16), pp. 1077-1079, 1989.

[Dix00] Dix J, Subrahmanian VS, Pick G, Meta-agent programs**,** *The Journal of Logic Programming,* 46(1-2), pp. 1-60 , 2000.

[Dor96] Dorigo M, Maniezzo V, Colorni A, The Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man and Cybernetics*, 26(1), pp. 1-13, 1996.

[Eit99a] Eiter T, Subrahmanian VS, Rogers TJ, Heterogeneous active agents, III: polynomially Implementable Agents, *Artificial Intelligence*, 117(1), pp. 107-167, 2000.

[Eit99b] Eiter T, Subrahmanian VS, Pick G, Heterogeneous active agents, I: semantics, *Artificial Intelligence*, 108(1-2), pp. 179–255, 1999.

[Fer99] Ferber J, *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*, Addison Wesley Longman, 1999.

[Fis36] Fisher RA, The use of Multiple Measurements in Taxonomic Problems, *Annals Eugenics*, 7, pp. 179-188, 1936.

[Fre99] Freitas Jr. R, *Nanomedicine V1: Basic Capabilities*, Landes Bioscience Publishers, 1999.

[Gan00] Ganeshan R, Johnson WL, Shaw E, Wood BP, Intelligent tutoring systems, *Lecture notes in computer science*, Springer–Verlag, 2000.

[Gel95] Gell-Man M, What is complexity?, *Complexity*, 1(1), pp. 16-19, 1995.

[Gno99] Gnoth M, Münich I, *ChariTime Systementwurf*, Humboldt University Berlin, Dep. of Computer Science Working Paper, 1999.

[Gre91] Grefenstette J, Lamarkian Learning in Multi-agent Environments, *Proceedings of the Fourth International Conference of Genetic Algorithms*, pp. 303-310, Morgan Kaufmann, 1991.

[Har89] Harrison W, An Entropy-Based Measure of Software Complexity, *IEEE Transactions on Software*, 18(11), pp. 1025-1029, 1989.

[Hay95] Haynes T, Waiwright R, Sen S, Evolving a Team, *Working Notes of the AAAI-95 Symposium on Genetic Programming*, AAAI Press, 1995.

[Iba96] Iba H, Emergent Cooperation for Multiple Agents using Genetic Programming, *Parallel Problem Solving from Nature IV PPSN96*, 1996.

[Iba97] Iba H, Nozoe T, Ueda K, Evolving Communicating Agents based on Genetic Programming, *Proc. of the IEEE International Conference on Evolutionary Computation ICEC97*, 1997.

[Jaa94] Jaakkola T, Jordan M, Singh S, On the Convergence of Stochastic Iterative Dynamic Programming Algorithms, *Neural Computation*, 1994.

[Kok94] Kokol P, et al., Decision Trees and Automatic Learning and Their Use in Cardiology, *Journal of Medical Systems*, 19(4), 1994.

[Kok96] Kokol P, Brest J, Zumer V, Software Complexity - An Alternative View, *SIGPLAN*, 31(2), pp. 35-41, 1996.

[Kok99] Kokol P, Podgorelec V, Zorman M, Pighin M, Alpha - a generic software complexity metric, *Proc. of the ESCOM - SCOPE '99*, pp. 397-405, 1999.

[Koz92] Koza J, *Genetic Programming - On the Programming of Computers by means of Natural Selection*, MIT Press, 1992.

[Koz94] Koza J, *Genetic Programming II - Automatic Discovery of Reusable Programs*, MIT Press, Cambridge MA, 1994.

[Lak01] Lakshmikumar A, Meta-agents, http://zen.ece.ohiou.edu/~robocup/papers/HTML/SSST/node8.html, 2001.

[Lan99] Lanzola G, Gatti L, Falasconi S, Stefanelli M, A framework for building cooperative software agents in medical applications, *Artificial Intelligence in Medicine*, 16(3), pp. 223-249, 1999.

[Les95] Lesser VR, Multiagent Systems: An Emerging Subdiscipline of AI, *ACM Computing Surveys*, 27, pp. 340-342, 1995.

[Mar76] Markiewicz W, et al, Mitral valve Prolaps in One Hundred Presumably Young Females, *Circulation*, 53(3), pp. 464-473, 1976.

[Mca95] McCallum R, Instance-based Utile Distinction for Reinforcement Learning with Hidden State, *Proc. of the Twelfth International Conference on Machine Learning*, 1995.

[Mul98] Mulvihill C, Patel A, O'Meara T, Intelligent agents for collaborative diagnosis, *Proceedings of Medinfo98*, 9(1), pp. 232-236, 1998.

[Pod99] Podgorelec V, proGenesys - Program Generation Tool Based on Genetic Systems, *Proc. of the International Conference on Artificial Intelligence IC-AI'99*, pp. 299-302, CSREA Press, 1999.

[Pod02] Podgorelec V, Kokol P, Stiglic B, Rozman I, Decision trees: an overview and their use in medicine, *Journal of Medical Systems*, 26(5), pp. 445-463, 2002.

[Qui93] Quinlan JR, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo CA, 1993.

[Sch93] Schenkel A, Zhang J, Zhang Y, Long range correlations in human writings, *Fractals*, 1(1), pp. 47-55, 1993.

[Sto96] Stone P, Veloso M, Multiagent Systems: A Survey from a Machine Learning Perspective, *IEEE Transactions on Knowledge and Data Engineering*, 1996.

[Sto97] Stone P, Veloso M, Using decision tree confidence factors for multiagent control, *RoboCup-97: The First Robot World Cup*, 1997.

[Tam96] Tambe M, Teamwork in Real-world, Dynamic Environments, *1996 International Conference on Multiagent Systems*, AAAI Press, 1996.

[Woo95] Wooldridge M, Jennings NR, Intelligent Agents: Theory and Practice, *Knowledge Engineering Review*, Cambridge University Press, 10(2), 1995.

Logitboost of Simple Bayesian Classifier

S. B. Kotsiantis and P. E. Pintelas
{sotos, pintelas}@math.upatras.gr
Educational Software Development Laboratory
Department of Mathematics
University of Patras, Hellas

Keywords: supervised machine learning, predictive data mining

Received: November 30, 2004

The ensembles of simple Bayesian classifiers have traditionally not been a focus of research. The reason is that simple Bayes is an extremely stable learning algorithm and most ensemble techniques such as bagging is mainly variance reduction techniques, thus not being able to benefit from its integration. However, simple Bayes can be effectively used in ensemble techniques, which perform also bias reduction, such as Logitboost. However, Logitboost requires a regression algorithm for base learner. For this reason, we slightly modify simple Bayesian classifier in order to be able to run as a regression method. Finally, we performed a large-scale comparison on 27 standard benchmark datasets with other state-of-the-art algorithms and ensembles using the simple Bayesian algorithm as base learner and the proposed technique was more accurate in most cases.

Povzetek: Preprosti Bayesov klasifikator je uporabljen v varianti Logiboost algoritma.

1 Introduction

The assumption of independence of simple Bayesian classifier is clearly almost always wrong. However, a large-scale comparison of simple Bayesian classifier with state-of-the-art algorithms for decision tree induction and instance-based learning on standard benchmark datasets found that simple Bayesian classifier sometimes is superior to each of the other learning schemes even on datasets with substantial feature dependencies [5]. An explanation why simple Bayesian method remains competitive, even though it provides very poor estimates of the true underlying probabilities can be found in [9].

Although simple Bayesian method remains competitive, the ensembles of simple Bayesian classifiers have not been a focus of research. The explanation is that simple Bayes is a very stable learning algorithm and most ensemble techniques such as bagging is mainly variance reduction techniques, thus not being able to benefit from its combination.

In this study, we combine simple Bayesian method with Logitboost [10], which is a bias reduction technique. As it is well known, Logitboost requires a regression algorithm for base learner. For this reason, we slightly modify simple Bayesian classifier in order to be able to run as a regression method. Finally, we performed a large-scale comparison with other state-of-the-art algorithms and ensembles on 27 standard benchmark datasets and the proposed technique was more accurate in most cases.

Description of some of the attempts that have been tried to improve the performance of simple Bayesian classifier using ensembles techniques is given in section 2. Section 3 discusses the proposed method. Experiment results in a number of data sets are presented in section 4, while brief summary with further research topics are given in Section 5.

2 Ensembles of simple Bayesian classifiers

Lately in the area of ML the concept of combining classifiers is proposed as a new direction for the improvement of the performance of individual classifiers. In [1], the researchers built an ensemble of simple Bayes classifiers using bagging [3] and boosting procedures [7]. They concluded that bagging did not manage to improve the results of simple Bayes classifier. The researchers also reported that there was a problem with boosting which was the robustness to noise. This is expected because noisy examples tend to be misclassified, and the weight will be increased for these examples.

In [20], the authors showed that boosting improves the accuracy of the simple Bayesian classifier in 9 out of the tested 14 data sets. However, they concluded that the mean relative error reduction of boosting over the simple Bayesian classifier in the 14 data sets was only 1%, indicating very marginal improvement due to boosting.

Other authors [13] also made use of Adaboost, with the difference that they used a discretization method and they removed redundant features in each iteration of Adaboost using a filter feature selection method. That algorithm has more significant mean relative error reduction over the simple Bayesian classifier in the tested data sets. The main reason is that the embedding feature selection technique makes the simple Bayes slightly unstable and as a result more suitable for Adaboost.

Other researchers presented Naive Bayes tree learner, called NBTree [11] that combines Naive Bayesian classification and decision tree learning. It uses a tree structure to split the instance space into sub-spaces defined by the path of the tree. A Naive Bayesian classifier is then generated in each sub-space. Each leaf of the Naive Bayesian tree contains a local Naive Bayesian classifier. As in many other learning algorithms that are based on tree structure, NBTree suffers from the small disjunct problem. To tackle this problem, other researchers [24] applied lazy learning techniques to Bayesian tree in-duction and presented the resulting lazy Bayesian rule learning algorithm LBR. LBR constructs a Bayesian rule specifically for an input test example and uses this rule to predict the class label of the example.

Another way that has been examined for generation of ensemble of simple Bayesian classifiers is by using different feature subsets randomly and taking a vote of the predictions of each classifier that uses different feature subset [21].

Melville and Mooney [14] present another meta-learner (DECORATE, Diverse En-semble Creation by Oppositional Relabeling of Artificial Training Examples) that uses a learner to build a diverse committee. This is accomplished by adding different randomly constructed examples to the training set when building new committee members. These artificially constructed examples are given category labels that disagree with the current decision of the committee, thereby directly increasing diversity when a new classifier is trained on the augmented data and added to the committee.

Finally, AODE classification algorithm [22] averages all models from a restricted class of one-dependence classifiers, the class of all such classifiers that have all other attributes depend on a common attribute and the class. The authors' experiments suggest that the resulting classifiers have substantially lower bias than Naive Bayes at the cost of a very small increase in variance.

3 Presented Algorithm

As we have also mentioned, Naive Bayes can be effectively used in ensemble techniques that perform bias reduction, such as Logitboost. However, Logitboost requires a regression algorithm for base learner. For this reason, we slightly modify simple Bayesian classifier so as to be able to run as a regression method.

Naive Bayes classifier is the simplest form of Bayesian network [5] since it captures the assumption that every feature is independent from the rest of the features, given the state of the class feature. Naive Bayes classifiers operate on data sets where each example x consists of feature values $<a_1, a_2 \ldots a_i>$ and the target function f(x) can take on any value from a pre-defined finite set $V=(v_1, v_2 \ldots v_j)$. Classifying unseen examples involves calculating the most probable target value $\mathbf{v}_{max}$ and is defined as:

$$v_{\max} = \max_{v_j \in V} P(v_j | a_1, a_2, ..., a_i).$$

Using Bayes theorem v_{max} can be rewritten as:

$$v_{\max} = \max_{v_j \in V} P(a_1, a_2, ..., a_i | v_j) P(v_j).$$

Under the assumption that features values are conditionally independent given the target value. The formula used by the Naive Bayes classifier is:

$$v_{\max} = \max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

where V is the target output of the classifier and $P(a_i|v_j)$ and $P(v_i)$ can be calculated based on their frequency in the training data.

Thus, Naive Bayes assigns a probability to every possible value in the target range. The resulting distribution is then condensed into a single prediction. In categorical problems, the optimal prediction under zero-one loss is the most likely value—the mode of the underlying distribution. However, in numeric problems the optimal prediction is either the mean or the median, depending on the loss function. These two statistics are far more sensitive to the underlying distribution than the most likely value: they almost always change when the underlying distribution changes, even by a small amount. For this reason NB is not as stable in regression as in classification problems. Some researchers have previously applied Naive Bayes to regression problems [6].

Generally, mapping regression into classification is a kind of pre-processing technique that enables us to use classification algorithms on regression problems. The use of the algorithm involves two main steps. First there is the creation of a data set with discrete classes. This step involves looking at the original continuous class values and dividing them into a series of intervals. Each of these intervals will be a discrete class. Every example whose output variable value lies within an interval will be assigned the respective discrete class. The second step consists on reversing the discretisation process after the learning phase takes place. This will enable us to make numeric predictions from our learned model.

One of the most well known techniques for discritization of numerical attributes is the Equal Width intervals (EW). This strategy creates a set of N intervals with the same number of elements. To better illustrate this strategy we show how they group the set of values {1,3,6,7,8,9.5,10,11} assuming that we want to partition them into three intervals (N=3). Using equal width we get [1 .. 4.33], {4.33 .. 7.66] and [7.66 .. 11] containing the values {1,3}, {6,7} and {8,9.5,10,11}.

For the proposed algorithm, we discretized the target value into a set of 10 equal-width intervals, and applied Naive Bayes for classification to the discretized data. At test time, the predicted value is the weighted average of each bin's value, using the classifiers probabilistic class memberships as weights.

It must be mentioned that the Logitboost algorithm [10] is based on the observation that Adaboost [7] is in essence fitting an additive logistic regression model to the training data. An additive model is an approximation to a function F(x) of the form:

$$F(x)=\sum_{m=1}^{M} c_m f_m(x)$$

where the c_m are constants to be determined and f_m are basis functions.

If we assume that F(x) is the mapping that we seek to fit as our strong aggregate hypothesis, and f(x) are our weak hypotheses, then it can be shown that the two-class Adaboost algorithm is fitting such a model by minimizing the criterion:

$$J(F)=E(e^{-yF(x)})$$

where *y* is the true class label in {-1,1}. Logitboost minimises this criterion by using Newton-like steps to fit an additive logistic regression model to directly optimise the binomial log-likelihood:

$$-\log(1+e^{-2yF(x)})$$

Finally, the proposed algorithm (LogitBoostNB) is summarized in (Figure 1), where p_j are the class probabilities returned by NB, N is the number of the examples of the dataset and J is the number of classes of the dataset.

Step 1: Initialization

- Start with weights $w_{i,j}=1/N$, i=1,…,N, j=1,…,J, $F_j(x)=0$ and $p_j=1/J \ \forall j$
- Discretize the target value into a set of 10 equal-width intervals

Step 2: LogitBoost iterations:

for m=1,2,...,10 repeat:

A. Fitting the NB learner

For j=1,...,J

- Compute working responses and weights for the jth class

$$z_{i.j}=\frac{y^*_{i,j}-p_j(x_i)}{p_j(x_i)(1-p_j(x_i))}$$

where y* the current response

$$w_{i.j}=p_j(x_i)(1-p_j(x_i))$$

- Fit the function $f_{mj}(x)$ by a weighted least squares regression of z_{ij} to x_i with weights w_{ij}

B. Set $f_{mj}(x)\leftarrow\frac{J-1}{J}\left(f_{mj}(x)-\frac{1}{J}\sum_{k=1}^{J} f_{mk}(x)\right)$ and

$F_j(x)\leftarrow F_j(x)+f_{mj}(x)$

C. Set $p_j(x)=\frac{e^{F_j(x)}}{\sum_{k=1}^{J} e^{F_k(x)}}$, enforcing the condition

$\sum_{k=1}^{J} F_k(x)=0$

Step 3: Output the classifier $\text{argmax}_j F_j(x)$

Figure 1: The proposed algorithm

In the following section, we present the experiments. It must be mentioned that the comparisons of the proposed algorithm are separated in three phases: a) with the other attempts that have tried to improve the accuracy of the simple Bayes algorithm, b) with other state-of-the-art algorithms and c) with other well-known ensembles.

4 Comparisons and Results

For the purpose of our study, we used 27 well-known datasets from many domains mainly from the UCI repository [2]. These data sets were hand selected so as to come from real-world problems and to vary in characteristics. Thus, we have used data sets from the domains of: pattern recognition (iris, zoo), image recognition (ionosphere, sonar), medical diagnosis (breast-cancer, breast-w, colic, diabetes, heart-c, heart-h, heart-statlog, hepatitis, lymphotherapy, primary-tumor) commodity trading (autos, credit-g) computer games (monk1, monk2, monk3), various control applications (balance) and prediction of student dropout (student) [12].

In Table 1, there is a brief description of these data sets.

Table 1: Description of the data sets

Datasets	Instances	Categ. features	Numer. features	Classes
autos	205	10	15	6
badge	294	4	7	2
balanceScale	625	0	4	3
breast-cancer	286	9	0	2
breast-w	699	0	9	2
colic	368	15	7	2
credit-g	1000	13	7	2
diabetes	768	0	8	2
haberman	306	0	3	2
heart-c	303	7	6	5
heart-h	294	7	6	5
heart-statlog	270	0	13	2
hepatitis	155	13	6	2
ionosphere	351	34	0	2
iris	150	0	4	3
labor	57	8	8	2
lymph/rapy	148	15	3	4
monk1	124	6	0	2
monk2	169	6	0	2
monk3	122	6	0	2
relation	2201	3	0	2
sonar	208	0	60	2
student	344	11	0	2
vechicle	846	0	18	4
vote	435	16	0	2
wine	178	0	13	3
zoo	101	16	1	7

In order to calculate the classifiers' accuracy, the whole training set was divided into ten mutually exclusive and equal-sized subsets and for each subset the classifier was trained on the union of all of the other subsets. Then, cross validation was run 10 times for each algorithm and the average value of the 10-cross validations was calculated. It must be mentioned that we used for the most of the algorithms the free available source code by the book [23].

In Table 2, we represent with "*v*" that the proposed LogitBoostNB algorithm *looses* from the specific algorithm. That is, the specific algorithm performed statistically better than LogitBoostNB according to t-test with $p<0.05$. Furthermore, in Table 2, "*" indicates that LogitBoostNB performed statistically better than the specific classifier according to t-test with $p<0.05$. In all the other cases, there is no significant statistical difference between the results (*Draws*). In the last rows of the Table 2 one can see the aggregated results in the form (*a/b/c*). In this notation "*a*" means that the proposed algorithm is significantly less accurate than the compared algorithm in *a* out of 27 datasets, "*c*" means that the proposed algorithm is significantly more accurate than the compared algorithm in *c* out of 27 datasets, while in the remaining cases (*b*), there is no significant statistical difference between the results. We also present the average accuracy of all the examined algorithms in all tested datasets.

The proposed algorithm is significantly more accurate than simple Bayes (NB) in 6 out of the 27 datasets, while it has not significantly higher error rates than simple Bayes in none dataset (see Table 2). Moreover, the proposed algorithm is significantly more accurate than AODE [22] in 2 out of the 27 datasets, while it has significantly higher error rates than AODE in one dataset.

We also compared the proposed algorithm with a Bayesian network classifier that is an extension of the simple Bayesian classifier. Several algorithms have been proposed in the last decade for inductive learning of Bayesian networks. Our experiments are based on the Bayesian scoring approach first used in K2 [4]. K2 proceeds by initially assuming that a node has no parents, and then adding incrementally that parent whose addition most increases the probability of the resulting network. Parents are added greedily to a node until the addition of no one parent can increase the structure probability. The proposed algorithm is significantly more accurate than Bayesian Network (K2) in 6 out of the 27 datasets, while it has not significantly higher error rates than simple Bayes in none dataset.

Table 2: Comparing the proposed algorithm with other Bayesian classifiers

Datasets	Logit-boostNB	NB	Bayesian Network (K2)	AODE
Autos	75.40	57.41*	67.26*	74.76
Badges	99.80	99.66	100.00	100.00
balance-scale	91.19	90.53	71.56*	69.96*
breast-cancer	66.67	72.7	72.59	73.05v
breast-w	96.18	96.07	97.20	97.05
Colic	80.74	78.7	80.98	82.45
credit-g	72.73	75.16	74.97	75.83
diabetes	74.11	75.75	75.25	75.70
haberman	71.48	75.06	71.57	71.57
heart-c	78.15	83.34	83.34	82.87
Heart-h	79.55	83.95	84.57	84.33
heart-statlog	79.93	83.59	82.56	82.70
hepatitis	84.67	83.81	84.18	85.36
ionosphere	92.71	82.17*	89.54	91.09
iris	94.87	95.53	93.20	93.07
labor	93.23	93.57	90.60	88.43
lymphography	84.65	83.13	85.64	86.86
monk1	85.33	73.38*	73.46*	82.32
monk2	59.92	56.83	56.78	59.62
monk3	91.87	93.45	93.45	93.21
relation	77.99	77.85	77.86	78.21
sonar	84.41	67.71*	76.71*	77.05*
students	83.04	85.70	85.76	86.08
vehicle	70.91	44.68*	61.05*	70.32
vote	95.20	90.02*	90.23*	94.28
wine	98.14	97.46	98.65	98.21
zoo	96.91	94.97	94.37	94.66
Average accuracy	83.70	81.19	81.98	83.30
W/D/L		0/21/6	0/21/6	1/23/2

In Table 3, one can see the comparisons of the proposed algorithm with other ensembles' techniques that have tried to improve the classification accuracy of the simple Bayes algorithm. Three well-known ensemble techniques were used for the comparison: Adaboost NB [19], Bagging NB [1], Decorate NB [14] with 25 iterations.

The proposed algorithm has significantly lower error rates in 6 out of the 27 datasets than Bagging NB, while it is significantly less accurate in one dataset. Furthermore, the proposed algorithm is significantly more accurate than Boosting NB in 3 out of the 27 datasets. In none dataset, the proposed algorithm has significantly higher error rate. Moreover, the proposed algorithm is significantly more accurate than Decorate NB in 7 out of the 27 datasets while, the proposed algorithm has significantly higher error rates in 2 datasets.

Table 3: Comparing the proposed algorithm with well known ensembles of NB

Datasets	Logit-boostNB	Adaboost NB	Bagging NB	DECORATE NB
autos	75.40	57.12 *	57.12 *	57.82 *
badges	99.80	99.66	99.69	96.73 *
balance-scale	91.19	91.68	90.29	90.55
breast-cancer	66.67	68.68	73.12v	72.97v
breast-w	96.18	95.55	96.04	95.97
colic	80.74	77.62	78.73	78.05
credit-g	72.73	75.14	75.20	74.72
diabetes	74.11	75.86	75.64	75.33
haberman	71.48	73.91	74.90	74.83
heart-c	78.15	82.97	83.37	83.51v
Heart-h	79.55	84.81	84.13	83.98
heart-statlog	79.93	82.59	83.59	83.74
hepatitis	84.67	84.62	84.13	82.99
ionosphere	92.71	91.06	82.00*	83.08*
iris	94.87	94.80	95.53	94.87
labor	93.23	89.60	93.73	92.87
lymphography	84.65	83.76	81.27	82.98
monk1	85.33	72.68*	73.22*	75.90*
monk2	59.92	56.83	56.56	57.08
monk3	91.87	90.90	93.37	93.29
relation	77.99	77.86	77.88	78.31
sonar	84.41	80.77	68.21*	67.65*
students	83.04	85.18	85.73	85.09
vehicle	70.91	44.68*	45.58*	46.77*
vote	95.20	95.01	90.02*	89.93*
wine	98.14	96.18	97.36	96.51
zoo	96.91	97.23	95.07	94.68
Average accuracy	83.70	81.73	81.17	81.12
W/D/L		0/24/3	1/20/6	2/18/7

Table 4: Comparing the proposed algorithm with other well known ensembles of NB

Datasets	Logit-boostNB	BoostFSNB	NBTree	LBR
autos	75.40	76.33	77.18	74.04
badges	99.80	100.00	100.00	100.00
balance-scale	91.19	81.98*	75.83*	72.17*
breast-cancer	66.67	72.07	70.99	72.35
Breast-w	96.18	96.05	95.97	97.23
colic	80.74	82.35	81.88	82.33
credit-g	72.73	71.36	74.07	74.90
Diabetes	74.11	75.10	75.18	75.38
haberman	71.48	73.08	71.97	71.57
heart-c	78.15	82.15	80.60	83.54
heart-h	79.55	83.61	81.33	84.54
heart-statlog	79.93	82.26	80.59	82.59
Hepatitis	84.67	84.81	81.36	84.97
ionosphere	92.71	91.86	89.49	89.92
iris	94.87	93.47	93.53	93.20
labor	93.23	88.03	91.70	87.50
lymphography	84.65	82.99	80.89	85.45
monk1	85.33	69.67*	91.78	94.91v
monk2	59.92	61.43	63.72	60.40
monk3	91.87	92.88	92.94	93.45
relation	77.99	77.23	78.00	78.31
sonar	84.41	77.95*	77.16	76.47*
students	83.04	86.20	84.62	85.38
vehicle	70.91	62.14*	71.03	69.43
vote	95.20	95.26	95.03	94.11
wine	98.14	96.84	96.57	98.71
zoo	96.91	95.75	94.55	93.21
Average accuracy	83.70	82.70	83.26	83.56
W/D/L		0/23/4	0/26/1	1/26/2

In Table 4, one can see the comparisons of the proposed algorithm with other more sophisticated ensembles of the simple Bayes algorithm. The proposed algorithm is significantly more precise than BoostFSNB algorithm [13] in 4 datasets, whilst it has not significantly higher error rates in any dataset. In addition, the proposed algorithm is significantly more accurate than NBTree [11] algorithm in 1 out of the 27 datasets, whereas it has significantly higher error rates in none dataset.

Furthermore, the proposed algorithm is significantly more precise than LBR [24] algorithm in 2 out of the 27 datasets, while it has significantly higher error rates in one dataset.

In brief, we managed to improve the performance of the simple Bayes Classifier obtaining better accuracy than other well known methods that have tried to improve the performance of the simple Bayes algorithm.

Finally, we compare the performance of the proposed technique with bagging decision trees and boosting decision trees that have been proved to be very successful for many machine-learning problems [18]. Similarly with the proposed algorithm, Quinlan [18] used 10 iterations for bagging and boosting C4.5 algorithm [17]. We also compare the proposed algorithm with Logitboost Decision Stump, which was the base learner used by the authors who proposed Logitboost [10]. In the last rows of the Table 5 one can see the aggregated results.

The proposed algorithm is significantly more accurate than boosting C4.5 algorithm with 10 classifiers in 3 out of the 27 datasets. In only 2 datasets, the proposed algorithm has significantly higher error rates. In addition, the proposed algorithm is significantly more accurate than bagging C4.5 algorithm with 10 classifiers in 4 out of the 27 datasets, while in 3 datasets, the proposed algorithm has significantly higher error rates using in any case less time for training. Moreover, the proposed algorithm is significantly more accurate than Logitboost

DS algorithm (using 25 classifiers) in 3 out of the 27 datasets, while in 1 dataset, the proposed algorithm has significantly higher error rate.

Table 5: Comparing the proposed algorithm with well known ensembles

Datasets	Logit-boostNB	Boost C4.5	Bagging C4.5	Logit-boost DS
autos	75.40	85.46	82.24 v	79.22
badges	99.80	100.00	100.00	100.00
balance-scale	91.19	78.35*	82.04*	87.34*
breast-cancer	66.67	66.89	72.71v	71.42
Breast-w	96.18	95.55	95.17	95.63
colic	80.74	81.63	85.34v	82.75
credit-g	72.73	70.75	73.89	71.68
Diabetes	74.11	71.69	75.65	74.54
haberman	71.48	71.12	72.78	73.61
heart-c	78.15	78.79	78.88	81.59
heart-h	79.55	78.68	79.93	81.44
heart-statlog	79.93	78.59	80.59	82.22
Hepatitis	84.67	82.38	80.73*	81.58
ionosphere	92.71	93.05	92.17	90.83
iris	94.87	94.33	94.67	94.93
labor	93.23	87.17	82.60	92.33
lymphography	84.65	80.87*	77.25*	82.36
monk1	85.33	94.10v	82.10	71.63*
monk2	59.92	60.82	59.80	55.60
monk3	91.87	90.01	92.38	93.37
relation	77.99	78.86	78.10	77.83
sonar	84.41	79.13*	78.51*	77.17*
students	83.04	81.70	86.20	86.73v
vehicle	70.91	75.59v	74.48	70.73
vote	95.20	95.51	96.27	95.49
wine	98.14	96.45	95.16	97.86
zoo	96.91	95.18	93.21	95.06
Average accuracy	83.70	83.06	83.07	83.15
W/D/L		2/22/3	3/20/4	1/23/3

To sum up, the proposed technique has better performance than all the tested algorithms.

5 Conclusion

Ideally, we would like to be able to identify or design the single best learning algorithm to be used in all situations. However, both experimental results [15] and theoretical work [16] indicate that this is not possible. The simple Bayes classifier has much broader applicability than previously thought. Besides its high classification accuracy, it also has advantages in terms of simplicity, learning speed, classification speed and storage space.

In this work, we managed to improve the performance of the simple Bayesian Classifier. We combined simple Bayesian method with Logitboost [10]. However, as it is well known, Logitboost requires a regression algorithm for base learner. For this reason, we slightly modified simple Bayesian classifier in order to run as a regression method. We performed a large-scale comparison with other attempts that have tried to improve the accuracy of the simple Bayes algorithm as well as other state-of-the-art algorithms and ensembles on 27 standard benchmark datasets and the proposed technique had better accuracy in most cases.

In future research it would be interesting to find a more sophisticated algorithm for choosing the number of intervals, for the application of Naive Bayes to the discretized data. How many intervals should be generated? Depending on the application, the trend of the error of the class mean or median for a variable number of classes can be observed. Too few intervals would imply an easier classification problem, but put an unacceptable limit on the potential performance; too many intervals might make the classification problem too difficult.

References

[1] E. Bauer, and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Machine Learning, 36 (1999): 105–139.

[2] C. L. Blake and C. J. Merz, UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998).

[3] L. Breiman, Bagging Predictors. Machine Learning, 24 (1996): 123-140.

[4] G.F. Cooper and E. Herskovits (1992). A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning, Vol. 9, pp 309-347. Kluwer Academic Publishers, Boston.

[5] P. Domingos and M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, Machine Learning, 29(1997): 103-130.

[6] E. Frank, Trigg L., Holmes G. and Witten I.H. (2000), Technical Note: Naive Bayes for regression, Machine Learning, 41(1) 5-26, October.

[7] Y. Freund and R. E. Schapire, Experiments with a New Boosting Algorithm, In Proceedings of ICML'96, 148-156.

[8] J. H. Friedman, On bias, variance, 0/1-loss and curse-of-dimensionality. Data Mining and Knowledge Discovery, 1(1997): 55-77.

[9] N. Friedman, D. Geiger and M. Goldszmidt, Bayesian network classifiers. Machine Learning, 29(1997): 131-163.

[10] J. Friedman, T. Hastie and R. Tibshirani, Additive Logistic Regression: a Statistical View of Boosting, The Annals of Statistics, 28: 337-374, 2000.

[11] R. Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. Proceedings of the second International Conference on Knowledge Discovery and Data Mining. Menlo Park, CA: The AAAI Press. 1996. pp. 202-207

[12] S. Kotsiantis, C. Pierrakeas and P. Pintelas, Preventing student dropout in distance learning systems using machine learning techniques, Lecture Notes in Artificial Intelligence, Springer-Verlag Vol 2774, pp 267-274.

[13] S. Kotsiantis, P. Pintelas, Increasing the Classification Accuracy of Simple Bayesian Classifier, Lecture Notes in

Artificial Intelligence, Springer-Verlag Vol 3192, pp. 198-207.
[14] P. Melville, and R. Mooney (2003), Constructing Diverse Classifier Ensembles using Artificial Training Examples, Proceedings of the IJCAI-2003, pp.505-510, Acapulco, Mexico, August 2003
[15] D. Michie, D. Spiegelhalter and C. Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood, 1994.
[16] T. Mitchell, Machine Learning, McGraw Hill, 1997.
[17] J.Quinlan, C4.5: Programs for machine learning. Morgan Kaufmann, San Francisco, 1993.
[18] J. R. Quinlan, Bagging, boosting, and C4.5. In Proceedings of the Thirteenth National Conference on Artificial Intelligence (1996), 725–730.
[19] G. Ridgeway, D. Madigan and T. Richardson, Interpretable boosted Naive Bayes classification. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, Menlo Park (1998): 101-104.
[20] K. Ting and Z. Zheng,. Improving the Performance of Boosting for Naive Bayesian Classification, N. Zhong and L. Zhou (Eds.): PAKDD'99, LNAI 1574, pp. 296-305, 1999.
[21] A. Tsymbal, S. Puuronen and D. Patterson, Feature Selection for Ensembles of Simple Bayesian Classifiers, In Proceedings of ISMIS (2002): 592-600, Lyon, June 27-29.
[22] G. Webb, J. Boughton & Z. Wang (2004). Not So Naive Bayes, To be published in Machine learning journal.
[23] I. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Mateo, 2000.
[24] Z. Zheng, and G.I. Webb, Lazy learning of Bayesian rules. Machine Learning, 41 (2000): 53-84

Simulated Annealing Fuzzy Clustering in Cancer Diagnosis

Xiao-Ying Wang and Jonathan M. Garibaldi
Automated Scheduling, Optimisation and Planning (ASAP) Research Group
Department of Computer Science & Information Technology
The University of Nottingham, Jubilee Campus, Wollaton Road, United Kingdom
{xyw, jmg}@cs.nott.ac.uk

Keywords: Fourier Transform Infrared spectroscopy, Hierarchical Cluster Analysis, Fuzzy C-Means, Simulated Annealing Fuzzy Clustering, Xie-Beni validity measure.

Received: November 10, 2004

Classification is an important research area in cancer diagnosis. Fuzzy C-means (FCM) is one of the most widely used fuzzy clustering algorithms in real world applications. However there are two major limitations that exist in this method. The first is that a predefined number of clusters must be given in advance. The second is that the FCM technique can get stuck in sub-optimal solutions. In order to overcome these two limitations, Bandyopadhyay proposed a Variable String Length Simulated Annealing (VFC-SA) algorithm. Nevertheless, when this algorithm was implemented, it was found that sub-optimal solutions were still obtained in certain circumstances. In this paper, we propose an alternative fuzzy clustering algorithm, Simulated Annealing Fuzzy Clustering (SAFC), that improves and extends the ideas present in VFC-SA. The data from seven oral cancer patients tissue samples, obtained through Fourier Transform Infrared Spectroscopy (FTIR), were clustered using FCM, VFC-SA and the proposed SAFC algorithm. Experimental results are provided and comparisons are made to illustrate that the SAFC algorithm is able to find better clusters than the other two methods.

Povzetek: Opisana je nova variacija algoritma FMC za klasifikacijo s pomočjo mehkega grupiranja.

1 Introduction

Cancer has become one of the major causes of mortality around the world and research into its diagnosis and treatment has become an important issue for the scientific community. In Britain, more than one in three people will be diagnosed with cancer during their lifetime and one in four will die from cancer. Accurate diagnostic techniques could enable various cancers to be detected in their infancy and, consequently, the corresponding treatments could be undertaken earlier. In recent years, FTIR has been increasingly applied to the study of biomedical conditions and could become a very powerful tool for determination and monitoring of chemical composition within biological systems [1]. It has also been used as a diagnostic tool for various human cancers and other diseases [2-5]. This technology works by measuring the wavelengths at which different functional groups of chemical samples absorb infrared radiation (IR) and the intensities of these absorptions. The quantity of absorption depends on the chemical bonds and the structure of the molecule and, hence, small changes in molecular structure can significantly affect the absorption intensity. Since chemical functional groups absorb light at specific wavelengths, the resultant FTIR spectrum can be likened to a molecular "fingerprint". If the characteristic spectrum of an abnormal and normal tissue component is known (in a "fingerprint library"), it may be possible to compare each obtained spectrum to these reference spectra and, hence, accurate diagnosis may be achieved. An instance of FTIR spectra from a non-biochemical application in which example spectra for standard and unknown paint samples are compared is shown in Figure 1 [6]. In the context of cancer diagnosis, the FTIR technique detects molecular differences within the cell rather than morphological changes of the cell and hence may lead to earlier detection of cell abnormalities.

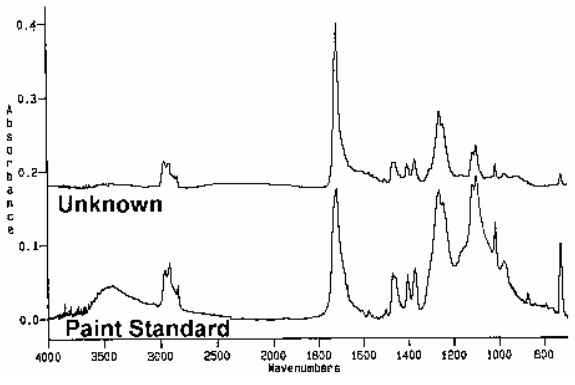


Figure 1. FTIR spectra for paint analysis.

Some advantages of FTIR analysis compared to conventional cytological clinical analysis might be:

1) *It has the potential for fully automatic measurement and analysis.*
2) *It is very sensitive; very small samples are adequate.*
3) *It is potentially much quicker and so cheaper for large scale screening procedures.*
4) *It has the potential to detect changes in cellular composition prior to such changes being detectable by other means.*

In previous clinical work [7], Chalmers *et al.* reported on the analysis of sets of FTIR spectra taken from oral cancer tissue samples. In general, the experiments analyzed the tissue samples in two parallel processes. In the first process, the samples were scanned by FTIR spectroscopy, various pre-processing techniques (such as mean-centering, variance scaling and first derivative) were performed on the FTIR spectral data empirically. The data was then classified by hierarchical cluster analysis after principal component analysis. In the second process, the samples were stained with a chemical solution and then examined through conventional cytology to group the samples into different functional groups. The results from these two processes were then compared. The clustering results showed that accurate clustering could only be achieved by manually applying pre-processing techniques that varied according to the particular sample characteristics and clustering algorithms. However, the pre-processing procedures needed extra time, software tools and significant human expertise. If a clustering technique could be developed which could obtain clustering results as good or even better than conventional clinical analysis without the necessity for pre-processing procedures, it would make the diagnosis more efficient and enable automation.

In previous research work, hierarchical clustering analysis (HCA) and the fuzzy c-means (FCM) algorithm have been used to classify non pre-processed FTIR oral cancer data [8]. The results showed that the FCM method performed significantly better than HCA. However, there are two major limitations of FCM which may affect the use of the technique as a practical diagnostic tool. Firstly, before performing the algorithm, an assumption of the number of clusters has to be made in advance. In real medical diagnosis, of course this number would not be known. Secondly, it is a non-convex method [9] so may often lead to local minima solutions, and hence misdiagnosis could occur. In order to avoid these limitations, a simulated annealing based FCM algorithm (SAFC) was introduced by the authors in [10]. It was developed by modifying and extending Bandyopadhyay's Variable String Length Simulated Annealing (VFC-SA) algorithm which was used for the classification of remote sensing satellite images [11].

In this paper, we describe the SAFC algorithm in further detail and give additional analysis on the experimental results. In Section 2, the background techniques and some related work are introduced. The original VFC-SA and our extended SAFC algorithm are described in Section 3. In Section 4, we provide the results of our experimentation in which the FCM, VFC-SA and SAFC algorithms were applied to seven sets of oral cancer FTIR data. The classification results are discussed in Section 5 and conclusions are drawn.

2 Background

2.1 FCM algorithm

The FCM algorithm, also known as Fuzzy ISODATA, is one of the most frequently used methods in pattern recognition. It is based on minimisation of the objective function (1) to achieve good classifications.

$$J(U,V) = \sum_{i=1}^{n}\sum_{j=1}^{c}(\mu_{ij})^m \| x_i - v_j \|^2 \qquad \textbf{(1)}$$

J(U,V) is a squared error clustering criterion, and solutions of minimisation of (1) are least-squared error stationary points of *J(U,V)*. The expression, $X = \{x_1, x_2, ...x_n\}$ is a collection of data, where *n* is the number of data points. $V = \{v_1, v_2, ...v_c\}$ is a set of corresponding cluster centres in the data set ***X***, where *c* is the number of clusters. μ_{ij} is the membership degree of data x_i to the cluster centre v_j. Meanwhile, μ_{ij} has to satisfy the following conditions:

$$\mu_{ij} \in [0,1], \qquad \forall i = 1,...n, \forall j = 1,...c \qquad \textbf{(2)}$$

$$\sum_{j=1}^{c}\mu_{ij} = 1, \qquad \forall i = 1,...n \qquad \textbf{(3)}$$

Where $U = (\mu_{ij})_{n*c}$ is a fuzzy partition matrix, $\| x_i - v_j \|$ represents the Euclidean distance between x_i and v_j, parameter *m* is the "fuzziness index" and is used to control the fuzziness of membership of each datum in the range $m \in [1, \infty]$. In this experimentation the value of $m = 2.0$ was chosen. Although there is no theoretical basis for the optimal selection of *m*, this has been chosen because the value has been commonly applied within the literature. The FCM algorithm is described in, for example, [12] and can be

performed by the following steps:

1) Initialize the cluster centres $V = \{v_1, v_2, ...v_c\}$, or initialize the membership matrix μ_{ij} with random value and make sure it satisfies conditions **(2)** and **(3)** and then calculate the centres.

2) Calculate the fuzzy membership μ_{ij} using

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} (\frac{d_{ij}}{d_{ik}})^{\frac{2}{m-1}}} \quad \textbf{(4)}$$

where $d_{ij} = \| x_i - v_j \|, \forall i = 1,...,n, \forall j = 1,...c$

3) Compute the fuzzy centres v_j using

$$v_j = \frac{\sum_{i=1}^{n} (\mu_{ij})^m x_i}{\sum_{i=1}^{n} (\mu_{ij})^m}, \forall j = 1,...,c \quad \textbf{(5)}$$

4) Repeat steps 2) and 3) until the minimum *J* value is achieved.

5) Finally, defuzzification is necessary to assign each data point to a specific cluster (i.e. by setting a data point to a cluster for which the degree of the membership is maximal).

2.2 Simulated Annealing algorithm and related works

The first simulated annealing algorithm was proposed by Metropolis et al. in 1953 [13]. It was motivated by simulating the physical process of annealing solids. The process can be described as follows. Firstly, a solid is heated from a high temperature and then cooled slowly so that the system at any time is approximately in thermodynamic equilibrium. At equilibrium, there may be many configurations with each one corresponding to a specific energy level. The chance of accepting a change from the current configuration to a new configuration is related to the difference in energy between the two states. Kirkpatrick et al. were the first to introduce simulated annealing to optimisation problems in 1982 [14]. Since then, simulated annealing has been widely used in combinatorial optimisation problems and has achieved good results on a variety of problem instances.

We use E_n and E_c represent the new energy and current energy respectively. E_n is always accepted if it satisfies $E_n < E_c$, but if $E_n \geq E_c$ the new energy level is only accepted with a probability as specified by $\exp(-(E_n - E_c)/T)$, where T is the current temperature. Hence, worse solutions are accepted based on the change in solution quality which allows the search to avoid becoming trapped at local minima. The temperature is then decreased gradually and the annealing process is repeated until no more improvement is reached or any termination criteria have been met.

Al-Sultan [15,16] and Kein and Dubes [17] have developed algorithms based on simulated annealing to find the global minimum solution using Fuzzy C-Means and other crisp (non-fuzzy) clustering methods. These were applied, for example, to determine the best clustering criterion for the multi-sensor fusion problem. However, the number of clusters has to be declared in advance for both of these techniques.

Although simulated annealing is used in the experimentation described here, other search algorithms have been used by other authors. Tseng and Yang proposed a genetic algorithm based clustering algorithm, in which the genetic algorithm was used to group the small clusters into successively larger clusters. A heuristic strategy [18] is then used to find a 'good' clustering (see below). Maulik and Bandyopadhyay developed a fuzzy clustering method which combined a genetic algorithm and FCM clustering to automatically segment satellite images obtained by remote sensing [19].

2.3 Xie-Beni validity index

Clustering validity is a concept that is used to evaluate the quality of clustering results. If the number of clusters is not known prior to commencing an algorithm, the clustering validity index may be used to find the optimal number of clusters [20]. This can be achieved by evaluating all of the possible clusters with the validity index and then the optimal number of clusters can be determined by selecting the minimum value of the index.

Many clusters validation indices have been developed in the past. In the context of fuzzy methods, some of them only use the membership values of a fuzzy cluster of the data, such as the partition coefficient [21] and partition entropy [22]. The advantage of this type of index is that it is easy to compute but it is only useful for the small number of well-separated clusters. Furthermore, it also lacks direct connection to the geometrical properties of the data. In order to overcome this problem Xie and Beni defined a validity index which measures the compactness and separation of clusters [23]. In this paper, the Xie-Beni index has been chosen as the cluster validity measure because it has been shown to be able to detect the correct number of clusters in several experiments [24]. Xie-Beni validity is the combination of two functions. The first calculates the compactness of data in the same cluster and the second computes the separateness of data in different clusters. Let S represent the overall validity index, π be the

compactness and s be the separation of the fuzzy c-partition of the data set. The Xie-Beni validity can now be expressed as:

$$S = \frac{\pi}{s} \qquad \textbf{(6)}$$

where

$$\pi = \frac{\sum_{j=1}^{c}\sum_{i=1}^{n} \mu_{ij}^{\ 2} \| x_i - v_j \|^2}{n},$$

and

$$s = (d_{\min})^2.$$

$d_{\min}$ is the minimum distance between cluster centres, given by $d_{\min} = \min_{ij} \| v_i - v_j \|$.

Smaller values of π indicate that the clusters are more compact and larger values of s indicate the clusters are well separated. Thus a smaller S reflects that the clusters have greater separation from each other and are more compact.

3 VFC-SA and SAFC

Recently, Bandyopadhyay proposed a Variable String Length Simulated Annealing (VFC-SA) algorithm [11]. It has the advantage that, by using simulated annealing, the algorithm can escape local optima and, therefore, may be able to find globally optimal solutions. The Xie-Beni index was used as the cluster validity index to evaluate the quality of the solutions. Hence this VFC-SA algorithm can generally avoid the limitations which exist in the standard FCM algorithm. However when we implemented this proposed algorithm, it was found that sub-optimal solutions could be obtained in certain circumstances. In order to overcome this limitation, we extended the original VFC-SA algorithm to produce the Simulated Annealing Fuzzy Clustering (SAFC) algorithm. In this section, we will describe the original VFC-SA and the extended SAFC algorithm in detail.

3.1 VFC-SA algorithm

In this algorithm, all of the cluster centres were encoded using a variable length string to which simulated annealing was applied. At a given temperature, the new state (string encoding) was accepted with a probability:

$$1/(1+\exp(-(E_n - E_c)/T))$$

The Xie-Beni index was used to compute the evaluation of a cluster. The initial state of the VFC-SA was generated by randomly choosing c points from the data sets where c is a integer within the range $[c_{\min}, c_{\max}]$. The values $c_{\min} = 2$ and $c_{\max} = \sqrt{n}$ (where n is the number of data points) was used following the suggestion proposed by Bezkek in [25]. The initial temperature T was set to a high temperature $T_{\max}$, a neighbour of the solution was produced by randomly flipping one bit within the string (describing the cluster centres) and then the energy of the new solution was calculated. The new solution was kept if it satisfied the simulated annealing acceptance requirement. This process was repeated for a certain number of iterations, k, at the given temperature. A cooling rate, r, where $0 < r < 1$, decreased the current temperature $T = rT$ and was repeated until the T reached the termination criteria temperature $T_{\min}$, at which point the current solution was returned. The whole VFC-SA algorithm process is summarised in the following steps:

Set parameters $T_{\max}, T_{\min}, c, k, r$.

Initialised the string by randomly choosing c data points from the data set to be cluster centres.

Compute the corresponding membership values using equation **(4)**

Calculate the initial energy E_c using XB index from equation **(6).**

Set the current temperature $T = T_{\max}$.

while $T \geq T_{\min}$

For $i = 1$ to k

Perturb a current centre in the string.

Compute the corresponding membership values using equation **(4)**.

Compute the corresponding centres with the equation **(5)**.

Calculate the new energy E_n from the new string.

If $E_n < E_c$ or $E_n > E_c$ with accept probability > a random number between [0, 1], accept the new string and set it as current string.

Else, reject it.

End for

$T = rT$.

End while.

Return the current string as the final solution.

The process of perturbing a current cluster centre comprised three functions. They are: perturbing an existing centre (*Perturb Centre*), splitting an existing

centre (*Split Centre*) and deleting an existing centre (*Delete Centre*). At each iteration, one of the three functions was randomly chosen. When splitting or deleting a centre, the cluster sizes were used to select a centre. The size, C_j, of a cluster, j, can be expressed by (where c is the number of clusters):

$$| C_j | = \sum_{i=1}^{n} \mu_{ij} , \quad \forall j = 1,...c \qquad \textbf{(7)}$$

The three functions are described below.

a) *Perturb Centre*

A random centre in the string is selected. This centre position is then modified through addition of the change rate $cr[d] = r \cdot pr \cdot v[d]$, where v is the current chosen centre and $d = 1,...,N$, where N is the number of dimensions. r is a random number between [-1, 1] and pr is the perturbation rate which was set through initial experimentation as 0.007 as this gave the best trade-off between the quality of the solutions produced and time taken to achieve them. Let $v_{current}[d]$ and $v_{new}[d]$ represent the current and new centre respectively, and *Perturb Centre* can then be expressed as:

$$v_{new}[d] = v_{current}[d] + cr[d] .$$

b) *Split Centre*

The centre of the biggest cluster is chosen by using equation **(7)**. This centre is then replaced by two new centres which are created by the following procedure. A reference point with a membership value less than but closest 0.5 to the selected centre is identified. Then the distance between this reference point and the current chosen centre is calculated using:

$$dist[d] = | v_{current}[d] - w_{reference}[d] |$$

Finally, the two new centres are then obtained by:

$$v_{new}[d] = v_{current}[d] \pm dist[d]$$

c) *Delete Centre*

As opposed to *Split Centre*, the smallest cluster is identified and its centre deleted from the string encoding.

3.2 SAFC algorithm

When the original VFC-SA algorithm was implemented by the authors on a wider set of test cases than originally used by Bandyopadhyay [11], it was found to suffer from several difficulties. In order to overcome these difficulties, four extensions to the algorithm were developed. In addition, some details were not explicit in the original algorithm. In this Section, the focus is placed on the extensions to VFC-SA in order to describe the proposed SAFC algorithm.

The first extension is in the initialisation of the string. Instead of the original initialisation in which random data points were chosen as initial cluster centres, the FCM clustering algorithm was applied using the random integer $c \in [c_{\min}, c_{\max}]$ as the number of clusters. The cluster centres obtained from the FCM clustering are then utilised as the initial cluster centres for SAFC. This is because re-initialization is a source of computational inefficiency. Using the clustering results from previous results leads to a better initialization.

The second extension is in *Perturb Centre*. The method of choosing a centre in the VFC-SA algorithm is to randomly select a centre from the current string. However, this means that even a 'good' centre can be altered. In contrast, if the weakest (smallest) centre is chosen, the situation in which an already good (large) centre is destabilized is avoided. Ultimately, this can lead to a quicker and more productive search as the poorer regions of a solution can be concentrated upon.

The third extension is in *Split Centre*. If the boundary between the biggest cluster and the other clusters is not obvious (not very marked), then a suitable approach is to choose a reference point with a membership degree that is less than but closest to 0.5. That is to say there are some data points whose membership degree to the chosen centre is close to 0.5. There is another situation that can also occur in the process of splitting centre; the biggest cluster is separate and distinct from the other clusters. For example, let there be two clusters in a set of data points which are separated, with a clear boundary between them. v_1 and v_2 are the corresponding cluster centres at a specific time in the search as shown in Figure 2 (shown in two-dimensions). The biggest cluster is chosen, say v_1. Then a data point whose membership degree is closest to but less than 0.5 can only be chosen from the data points that belong to v_2 (where the data points have membership degrees less than 0.5 to v_1). So, for example, the data point w_1 (which is closest to v_1) is chosen as the reference data point. The new centres will then move to v_{new1} and v_{new2}. Obviously these centres are far from the ideal solution. Although the new centres would be changed by the *Perturb Centre* function afterwards, it will inevitably take a longer time to 'repair' the solutions. In the modified approach, two new centres are created within the biggest cluster. The same dataset as in Figure 2 is used to illustrate this process. A data point is chosen, w_1, that is closest the mean value of the membership degree above 0.5. Then two new centres v_{new1} and v_{new2} are created according the distance between v_1 and w_1. This is shown in Figure 3. Obviously the new centres are better than the ones in Figure 2 and therefore better solutions are likely to be found in same time (number of iterations).

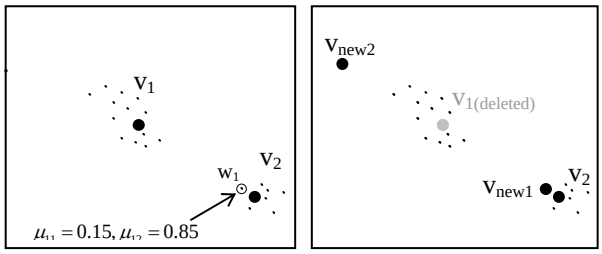


Figure 2. An illustration of *Split Centre* from the original algorithm with distinct clusters (where μ_{11} and μ_{12} represent the membership degree of w_1 to the centres v_1 and v_2 respectively)

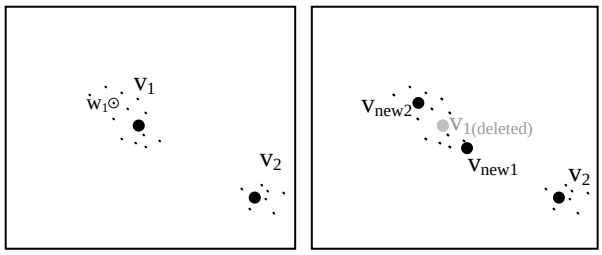


Figure 3. The new *Split Centre* applied to the same data set as Figure 2, above, (where w_1 is now the data point that is closest to the mean value of the membership degree above 0.5)

The fourth extension is in the final step of the algorithm (return the current solution as the final solution). In the SAFC algorithm, the best centre positions (with the best XB index value) that have been encountered are stored throughout the search. At the end of the search, rather than returning the current solution, the best solution seen throughout the whole duration of the search is returned.

Aside from these four extensions, we also ensure that the number of clusters never violates the criteria whereby the number of clusters C should be within the range of $[c_{\min}, c_{\max}]$. Therefore when splitting a centre, if the number of clusters has reached $c_{\max}$ then the operation is disallowed. Dually, when deleting a centre, the operation is not allowed if the number of clusters in the current solution is $c_{\min}$.

4 Experiments and Results

In this section, the clinical data used are firstly introduced and then the FCM, VFC-SA and SAFC algorithms are applied to seven sets of oral cancer FTIR data in order to compare the results.

4.1 Clinical data background

In these experiments, all the algorithms are applied to FTIR spectral data sets obtained from oral cancer patients. These data have been provided by Leeds Royal Infirmary, U.K. and Derby Royal Infirmary, U.K. and Derby City General Hospital, UK. All of the FTIR spectra data have been produced by a Nicolet 730 FTIR spectrometer (Nicolet Instruments, Inc., Madison, USA), which is interfaced to a NicPlan IR-microscope fitted with a liquid-nitrogen cooled narrow-band mercury-cadmium-telluride (MCT) detector. Transmission spectra were recorded either 4cm^{-1} or 8cm^{-1} spectral resolution, typically co-adding 512 or 1024 scans per spectrum. The FTIR microscope was operated using an $32\times$ objective lens. Background single-beam spectra were recorded through a blank BaF2 window. A Nicolet Nexus FTIR spectrometer interfaced to a Continuum IR microscope fitted with a narrow-band MCT detector, sited at the University of Nottingham, was used to record the conventional Globar-sourced spectra.

Multivariate data analysis on pre-processed spectra was undertaken using Infometrix Pirouette, version 3, multivariate analysis software (Infometrix, Inc., Woodinville, WA, USA). In this study, the data analysis was limited to those that lie within the spectral range 900-1800 cm^{-1}

The tissue samples, with nominal thickness $5\mu m$, were mounted on $0.5mm$ thickness BaF_2 windows for FTIR investigations. Parallel sections were stained conventionally to facilitate identifying regions for particular interest. Some of the sections used for infrared examinations were also stained after they had been studied spectroscopically.

In this study, the FCM, VFC-SA and SAFC algorithms were implemented in MATLAB (version 6.5.0, release 13.0.1).

All the FTIR spectra were taken from three oral cancer patients, which contain a mixture of tumour (neoplasm), stroma (connective tissue), ‘early keratinisation’ and ‘necrotic’. The seven data sets have been taken from three different patients. The number of data points within each of these data sets is: 15, 18, 11, 31, 30, 15 and 42. Figure 4 (a) shows a $4\times$ magnification visual image from one of the hematoxylin and Eosin stained oral tissue sections, which has been taken from the first patient. There are two types of cells (stroma and tumour) in this section with their regions clearly identifiable by their light and dark coloured stains respectively. Figure 4(b) shows a $32\times$ magnified visual image from a portion of a parallel, unstained section; the superimposed dashed white line separates the visually different morphologies. Five single point spectra were recorded from each of the three distinct regions using an aperture of $10\mu m \times 10\mu m$. The locations of these are marked by ‘+’ on Figure 4.(b) and numbered as 1-5 for the upper tumour region, 6-10 for the central stroma layer, and 11-15 for the lower tumour region. The fifteen FTIR transmission spectra from these positions are recorded as data set 1, and corresponding FTIR spectra are shown in Figure 5.

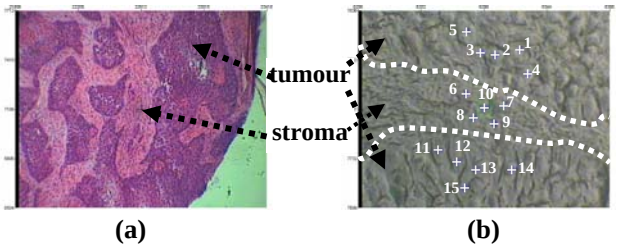


Figure 4. Tissue samples from data set 1 (a) stained (b) unstained

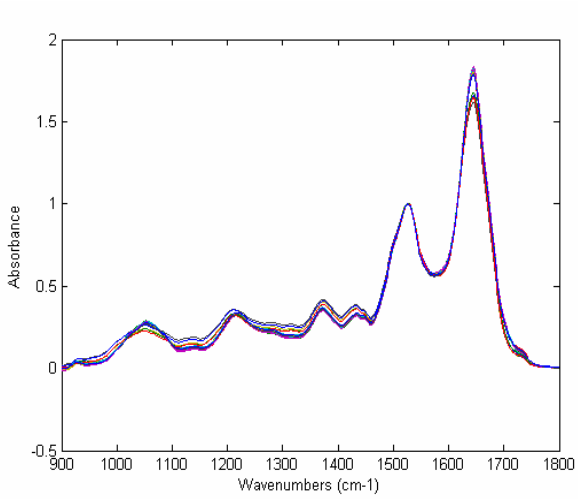


Figure 5. FTIR spectra from data set 1

4.2 Evaluation of FCM, VFC-SA and SAFC

In these experiments, the number of different types of cells in each tissue section from clinical analysis was considered as the number of clusters to be referenced. They were also used as the parameter for FCM. The Xie-Beni index value has been utilised throughout to evaluate the quality of the classification for these three algorithms. The parameters for VFC-SA and SAFC are: $T_{\min} = 1e-5$, $k = 40$, $r = 0.9$. $T_{\max}$ was set as 3 in all cases. That is because the maximum temperature has a direct impact on how much worse the XB index value of a solution can be accepted at the beginning. If the $T_{\max}$ value is set too high, this may result in the earlier stages of the search being less productive because simulated annealing will accept almost all of the solutions and, therefore, will behave like random search. It was empirically determined that when the initial temperature was 3, the percentage of worse solutions that were accepted was around 60%. In 1996, Rayward-Smith et al discussed starting temperatures for simulated annealing search procedures and concluded that a starting temperature that results in 60% of worse solutions being accepted yields a good balance between the usefulness of the initial search and overall search time (i.e. high enough to allow some worse solutions, but low enough to avoid conducting a random walk through the search space and wasting search time) [26]. Therefore, the initial temperature was chosen based on this observation.

Solutions for the seven FTIR data sets were generated by using the FCM, VFC-SA and SAFC algorithms. Each data set was allowed 10 runs on each method. As mentioned at the beginning of this Section, the number of clusters was predetermined for FCM through clinical analysis. The outputs of FCM (centres and membership degrees) were then used to compute the corresponding XB index value. VFC-SA and SAFC automatically found the number of clusters by choosing the solution with the smallest XB index value. Table 1 shows the average XB index values obtained after 10 runs of each algorithm (best average is shown in bold).

Dataset	Average XB Index Value		
	FCM	VFC-SA	SAFC
1	0.048036	0.047837	**0.047729**
2	0.078896	0.078880	**0.078076**
3	0.291699	0.282852	**0.077935**
4	0.416011	0.046125	**0.046108**
5	0.295937	0.251705	**0.212153**
6	0.071460	0.070533	**0.070512**
7	0.140328	0.149508	**0.135858**

Table 1. Average of the XB index values obtained when using the FCM, VFC-SA and SAFC algorithms.

In Table 1, it can be seen that in all of these seven data sets, the average XB values of the solutions found by SAFC are smaller than both VFC-SA and FCM. This means that the clusters obtained by SAFC have, on average, better XB index values than the other two approaches. Put another way, it may also indicate that SAFC is able to escape sub-optimal solutions better than the other two methods.

In the data sets 1, 2, 4 and 6, the average of XB index values in SAFC is only slightly smaller than that obtained using VFC-SA. Nevertheless, when the Mann-Whitney test (with $p<0.01$) [27] was conducted on the results of these two algorithms, the XB index for SAFC was found to be statistically significantly lower than that for VFC-SA for all data sets

The number of clusters obtained by VFC-SA and SAFC for each dataset is presented in Table 2. The brackets indicate the number of runs for which that particular cluster number was returned. For example on dataset 5, the VFC-SA algorithm found 2 clusters in 5 runs and 3 clusters in the other 5 runs. The number of clusters identified by clinical analysis is also shown for comparative purposes.

Dataset	Number of Clusters in Solution		
	Clinical	VFC-SA	SAFC
1	2	2(10)	2(10)
2	2	2(10)	2(10)
3	2	2(10)	3(10)
4	3	2(10)	2(10)
5	2	2(5), 3(5)	3(10)
6	2	2(10)	2(10)
7	3	3(9), 4(1)	3(10)

Table 2. Comparison of the number of clusters achieved by clinical analysis, VFC-SA and the SAFC methods.

In Table 2, it can be observed that in data sets 3, 4, 5 and 7, either one or both of the VFC-SA and SAFC obtain solutions with a differing number of clusters than provided by clinical analysis. In fact, with data sets 5 and 7, VFC-SA even produced a variable number of clusters within the 10 runs. Returning to the XB validity index values of Table 1, it was shown that all the average XB index values obtained by SAFC are better.

It can be observed that the corresponding XB average index values for SAFC for data sets 3, 4 and 5 produced much smaller values than FCM. These three data sets are also the data sets which SAFC obtained a different number of clusters to clinical analysis. In data set 3, the average XB index value in SAFC is much smaller than in VFC-SA. This is because the number of clusters obtained from these two algorithms is different (see Table 2). Obviously a different number of clusters lead to a different cluster structure, and so there can be a big difference in the validity index. In data sets 5 and 7, the differences of XB index values are noticeable, though not as big as data set 3. This is because in these two data sets, some runs of VFC-SA obtained the same number of clusters as SAFC.

In order to examine the results further, the data has been plotted using the first and second principal components in two dimensions. These have been extracted using the principal component analysis (PCA) technique [28, 29]. The data has been plotted in this way because, although the FTIR spectra are limited to within $900cm^{-1} - 1800cm^{-1}$, there are still 901 absorbance values corresponding to each wavenumber for each datum. The first and second principal components are the components that have the most variance in the original data. Therefore, although the data is multidimensional, the principal components can be plotted to give an approximate visualization of the solutions that have been achieved. Figures 6, 7, 8 and 9 show the results for data sets 3, 4, 5 and 7 respectively using SAFC (the data in each cluster is depicted using different markers and each cluster centre is presented by a star). The first and second principal components in data sets 3, 4, 5 and 7 contain 89.76, 93.57, 79.28 and 82.64 percent of the variances in the original data, respectively.

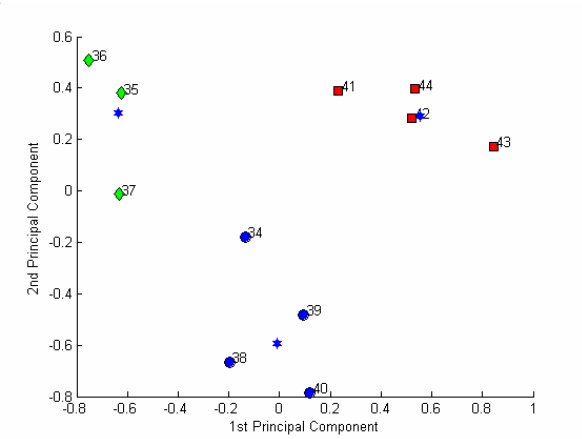


Figure 6. SAFC Cluster result in PCA for data set 3.

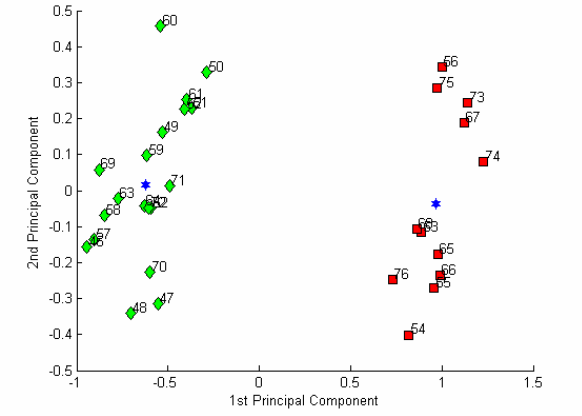


Figure 7. SAFC Cluster result in PCA for data set 4.

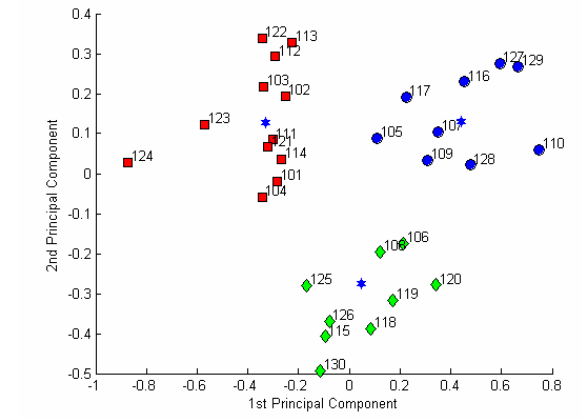


Figure 8. SAFC Cluster result in PCA for data set 5.

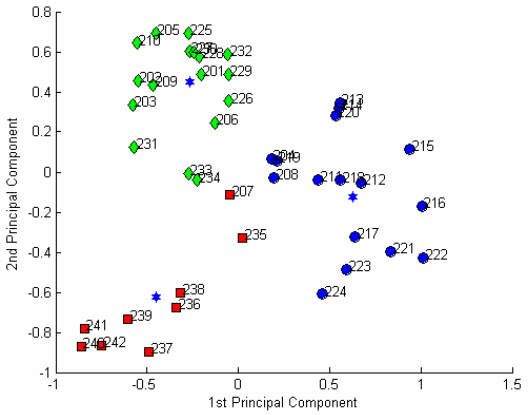


Figure 9. SAFC Cluster result in PCA for data set 7.

There are three possible explanations for this phenomenon. Firstly, the clinical analysis *may not* be correct – this could potentially be caused by the different types of cells in the tissue sample not being noticed by the clinical observers or the cells within each sample could have been mixed with others. Secondly, it could be that although a smaller XB validity index value was obtained, indicating a 'better' solution in technical terms, the Xie Beni validity index is not accurately capturing the real validity of the clusters. Put another way, although the SAFC finds the better solution in terms of Xie-Beni validity index, this is not actually the best set of clusters in practice. A third possibility is that the FTIR spectroscopic data has not extracted the required information necessary in order to permit a correct determination of cluster numbers – i.e. there is a methodological problem with the technique itself. None of these explanations of the difference between SAFC and VFC-SA algorithms detracts from the fact that the SAFC produces better solutions in that it consistently finds better (statistically lower) values of the objective function (Xie-Beni validity index).

5 Conclusion

In this paper, a new SAFC method has been proposed which has been extended from the original VFC-SA algorithm in four ways. The newly proposed algorithm's performance has been evaluated on seven oral cancer FTIR data and compared to clinical analysis, FCM and VFC-SA. The XB validity index was used as the evaluation method to measure the quality of the clusters produced. The experimental results have shown that the SAFC algorithm can escape the sub-optimal solutions obtained in the other two approaches and hence produce better clusters. On the other hand, the number of clusters obtained by SAFC in some data sets are not in agreement with those provided through clinical analysis. This can be explained in three ways. Firstly, the number of cluster from clinical analysis may not correct; secondly, the XB validity index may not suitable to apply on these clinical data; and thirdly, the FTIR technique has not (for these data sets) captured sufficient information to permit correct classification. However, more results and information are needed before any definitive conclusion can be made in this case. Nevertheless, this SAFC algorithm is a further step towards the automatic classification of data for real medical applications. The further development of this algorithm is ongoing research area.

In the future, we are also trying to obtain a wider source of sample data for which the number of classifications is known from a number of clinical domains such as cervical cancer smear test screening and lymphnodes disease. Establishing the techniques necessary to develop clinically useful automated diagnosis tools across a range of medical domains is the ultimate goal of this research.

Acknowledgement

The authors are grateful to John Chalmers et al. for providing the FTIR spectral data used in this study, and for making available their internal report on the clinical study carried out at Derby General City Hospital. The authors would like to express their kind thanks to Sanghamitra Bandyopadhyay for permission to use and extend the VFC-SA algorithm. Moreover, the authors also like to express their thanks to the anonymous referees for their valuable comments.

References

[1] H. Mantsch, R. N. McElhaney, "Application of IR spectroscopy to biology and medicine," *J Molec Struc*, vol. 217, pp. 347-362, 1990.

[2] P. T. T. Wong, B. Rigas, "Infrared spectra of microtome sections of human colon tissues," *Applied Spectroscopy*, vol. 44, pp. 1715-1718, 1990.

[3] B. Rigas, S. Morgello, I.S. Goldan, P. T. T. Wong, "Human colorectal cancers display abnormal Fourier –transform infrared spectra," in *Proc. The National Academy of Science of the USA*, vol. 87, 1990, pp. 8140-8144.

[4] P. T. T. Wong, S. M. Goldstein, R. C. Grekin, T. A. Godwin, C. Pivik, B. Rigas, "Distinct infrared spectroscopic patterns of human basal cell carcinoma of the skin," *Cancer Research*, vol. 53, no. 4, pp. 762-765, 1993.

[5] B. J. Morris, C. Lee, B. N. Nightingale, E. Molodysky, L. J. Morris, R. Appio, "Fourier transform infrared spectroscopy of dysplastic, papillomavirus-positive cervicovaginal lavage

speciens," *Gynecological Oncology*, vol. 56., no. 2, pp. 245-249, 1995.

[6] *Handbook of Analytical Method for Materials,* Materials Evaluation and Engineering, Inc, United Stated of America, 2001, pp. 15.

[7] R. Allibone, J. M. Chalmers, M. A. Chesters, S. Fisher, A. Hitchcock, M. Pearson, F. J. M. Rutten, I. Symonds, M. Tobin, "FT-IR microscopy of oral and cervical tissue samples", internal report, Derby City General Hospital, England, 2002, unpublished.

[8] X-Y. Wang, J.M. Garibaldi, and T. Ozen, "Application of The Fuzzy C-Means Clustering Method on the Analysis of non Pre-processed FTIR Data for Cancer Diagnosis", in the Proceedings of the 8th Australian and New Zealand Conference on Intelligent Information Systems, pp. 233-238, Sydney, Australia, December 10-12, 2003.

[9] S. Z. Selim, "Using nonconvex programming techniques in cluster analysis", *Jt Meet. Ops Res. Soc. Am. And Inst. Mgmt Sci*., Houston, Texas, 1981.

[10] X-Y. Wang, G. Whitwell and J.M Garibaldi, "The Application of a Simulated Annealing Fuzzy Clustering Algorithm for Cancer Diagnosis", in the Proceedings of IEEE 4th International Conference on Intelligent Systems Design and Application, pp. 467-472, Budapest, Hungary, August 26-28, 2004.

[11] S. Bandyopadhyay, "Simulated Annealing for Fuzzy Clustering: Variable Representation, Evolution of the Number of Clusters and Remote Sensing Application", Machine Intelligence Unit, Indian Statistical Institute, 2003, unpublished, private communication.

[12] J. Bezdek, Pattern Recognition With Fuzzy Objective Function Algorithms. New York: Plenum, 1981.

[13] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines", *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087-1092, 1953.

[14] S. Kirkpatrick, C. D. Gelatt Jr. and M. P. Vecchi, "Optimisation by Simulated Annealing", *IBM Research Report*, RC 9355, 1982.

[15] K. S. Al-Sultan and S. Z. Selim, "A Global Algorithm for the Fuzzy Clustering Problem", *Pattern Recognition*, vol. 26, no. 9, pp. 1357-1361, 1993.

[16] S. Z. Selim and K. Alsultan, "A Simulated Annealing Algorithm for the Clustering Problem", *Pattern Recognition*, vol. 24, no. 10, pp. 1003-1008, 1991.

[17] R. W. Klein and R. C. Dubes, "Experiments in Projection and Clustering by Simulated Annealing", *Pattern Recognition*, vol. 22. no. 2. pp. 213-220, 1989.

[18] L. Y. Tseng, S. B. Yang, "A genetic approach to the automatic clustering problem", *Patten Recognition*, vol. 34. pp. 415-424, 2001.

[19] U. Maulik and S. Bandyopadhyay, "Fuzzy Partitioning Using a Real-coded Variable Length Genetic Algorithm for Pixel Classification", *IEEE Trans. On Geosciences and Remote Sensing*, vol. 41, no. 5, 5 May 2003.

[20] M. Ramze Rezaee, B. P. F. Leieveldt, J. H. C. Reiber, A New Cluster Validity Index for the Fuzzy C-Means, *Pattern Recognition Letters*, Vol. 19, Elsevier, pp. 237-246, 1998.

[21] J. C.Bezdek, Cluster Validity with Fuzzy Sets. *J. Cybernet*, Vol. 3, No. 3, pp. 58-72, 1974.

[22] J. C. Bezdek, Mathematical Models for Systematics and Taxonomy. In Estabrook, G. (Ed.), Proceeding of 8th Internat Conference Numerical Taxonomy. Freeman, San Francisco, CA, pp. 143-166, 1975.

[23] X. L. Xie and G. Beni, " A validity measure for fuzzy clustering", *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 13, pp. 841-847, 1991.

[24] N. R. Pal, J. C. Bezdek, "On cluster validity for the fuzzy c-means model", *IEEE Trans. Fuzzy Systs.*, Vol. 3, pp. 370-379, 1995.

[25] J. C. Bezdek, *Pattern Recognition* in Handbook of Fuzzy Computation. IOP Publishing Ltdl, Boston, Ny 1998(Chapter F6).

[26] V. J. Rayward-Smith, I. H. Osman, C. R. Reeves and G. D. Smith, Modern Heuristic Search Methods. John Wiley & Sons, 1996.

[27] W. J. Conover, Practical Nonparametric Statistics. John Wiley & Sons, 1999.

[28] D. R. Causton, A Biologist's Advanced mathematics. London: Allen & Unwin, 1987.

[29] I. T. Jolliffe, Principal Component Analysis. New York: Springer-Verlag, 1986.

Fast Discovery of Frequent Itemsets: a Cubic Structure-Based Approach

Renata Ivancsy and Istvan Vajk
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
and HAS-BUTE Control Research Group
H-1111, Goldmann Gy. ter 3, Budapest, Hungary
{renata.ivancsy,vajk}@aut.bme.hu

Keywords: frequent itemset mining, Apriori algorithm, FP-growth algorithm

Received: November 20, 2004

Mining frequent patterns in large transactional databases is a highly researched area in the field of data mining. The different existing frequent pattern discovering algorithms suffer from various problems regarding the computational and I/O cost, and memory requirements when mining large amount of data. In this paper a novel approach is introduced for solving the aforementioned issues. The contribution of the new method is to count the short patterns in a very fast way, using a specific index structure. The suggested algorithm is partially based on the apriori hypothesis and exploits the benefit of a new index table-based cubic structure to count the occurrences of the candidates. Experimental results show the advantageous execution time behavior of the proposed algorithm, especially when mining datasets having huge number of short patterns. Its memory requirement, which is independent from the number of processed transactions, is another benefit of the new method.

Povzetek: Predstavljena je nova, hitrejša metoda iskanja krajših vzorcev v velikih transakcijskih bazah.

1 Introduction

The task of association rule mining is to find hidden, previously unknown and potentially useful information in large amount of data. Since it was first introduced by Agrawal et al [1] the problem of discovering frequent patterns has received a great deal of attention. The problem is widely known as market basket analysis, however, several other applications exist which are searching for frequent recurring itemsets.

In general the process of association rule mining consists of two main steps. The first one is to discover the frequent itemsets in the dataset. The second one is to create rules from the itemsets found during the first step. Most of the existing algorithm's aim is to find the frequent itemsets, i.e. the frequent patterns in the transactions because of two reasons. The first reason is the much higher computational complexity of the frequent pattern discovery task than that of the rule generation. The frequent itemsets are discovered from the original database, which can be terabytes in size; meanwhile the rules are generated from the relatively small number of itemsets found by the first step. The second reason is that the approach of discovering frequent patterns is utilized in wide range of applications, for example for mining sequential patterns, episodes, partial periodicity and many other important data mining tasks.

The different types of frequent itemset mining algorithms suit to datasets having different characteristics. However all of them has problems either with the computational cost or the I/O activity or the memory requirement. The "candidate generate and test" algorithms, such as the Apriori algorithm [2], suffer from the problem spending much of their time to discard the infrequent candidates on each level. Another problem can be the high I/O cost which is inseparable from the level-wise approach. In case of the Apriori algorithm the database is accessed as many times as the size of the maximal frequent itemset is. Several algorithms were developed based on the Apriori method in order to enhance its performance. One of them is the DHP (Dynamic Hash and Prune) [3] algorithm which uses hash tables to collect support information about the potentially $(i+1)$-itemsets when discovering the i-itemsets. In this way the cost of generating and testing the candidates on the $(i+1)^{th}$ level is reduced. Another enhancement of the Apriori algorithm is the DIC (Dynamic Itemset Counting) [4] algorithm. It defines checkpoints in the database and scans it continuously. When a checkpoint is reached, new candidates are generated from those itemsets which are proved frequent and those are discarded which are proved infrequent since the last pass of the same checkpoint. In this way the number of the database scans can be reduced. There are several algorithms contributed [5, 6, 7, 8] to improve the performance of the Apriori algorithm that use different type of approaches. An analysis of the best known algorithms can be found in [9].

The FP-growth (Frequent Pattern-growth) [10] algorithm differs basically from the level-wise algorithms, that use a "candidate generate and test" approach. It does not

use candidates at all, but it compresses the database into the memory in a form of a so-called FP-tree using a pruning technique. The patterns are discovered using a recursive pattern growth method by creating and processing conditional FP-trees. The drawback of the algorithm is its huge memory requirement which is dependent on the minimum support threshold and on the number and length of the transactions. [11] suggest a variant of the FP-growth algorithm, such that the memory cost of building conditional FP-trees are minimized due to building a so-called COFI-tree for each frequent item. Another memory resident algorithm is the H-mine algorithm [12] which represents the transactions as a list of elements in the memory. The traversal of the lists is helped with some header tables. A further memory-based frequent itemset counting algorithm was introduced in [13]. One advantage of the memory resident algorithms is that the number of the database accesses is independent from the size of the maximal frequent itemset. Unfortunately, the size of the memory is a function of the number of transactions.

The method proposed in this paper belongs to the Apriori-like algorithms, thus it uses candidates, but it has the advantage counting and testing them quickly using an index structure. Its other advantage is the relatively small memory requirement that is dependent on the minimum support threshold and on the item number.

The organization of the paper is as follows. [1] Section 2 defines the association rule mining problem. In Section 3 two of the most common association rule mining algorithms are described in detail, a level-wise "candidate generate and test" method, and a memory-based algorithm. The execution behavior of the presented algorithms is analyzed in Section 4. After drawing the conclusion of the experiments a new method is suggested and described in detail in Section 5. Some experimental results are shown in this section as well. Conclusion can be found in Section 6.

2 Problem statement

Frequent pattern mining is one of the most fundamental data mining tasks. It is used besides several applications mainly in association rule mining algorithms. This section formally introduces the problem of association rule mining and defines the most important terms in this field.

The association rule mining problem is defined as follows. Let $I = \{i_1, i_2, \ldots, i_n\}$ be the complete set of items appearing in the transactions, where n denotes the maximum number of items. An itemset is a non-empty subset of I, and if the length of the itemset is k, then it is called k-itemset. A transaction T is a set of items such that $T \subseteq I$. Each transaction in the database has an identifier, called TID. A transaction T contains an itemset X if and only if $X \subseteq T$. The support of the itemset X, denoted as $\sigma(X)$, is defined as the percentage of the transactions in the database which contain X.

An association rule is an implication of the form $X \rightarrow Y$ where both X and Y are itemsets, and there exists no item which appears both in X and in Y, formally, $X \subset I$, $Y \subset I$ and $X \cap Y = \emptyset$. An association rule has two properties: the support and the confidence. The support of the rule $X \rightarrow Y$ equals to the support of the itemset XY. The confidence, denoted with c, is the percentage of the transactions in the database containing X that also contain Y. This is taken as a conditional probability, $P(Y|X)$.

In order to reduce the search space and to discover only those rules which can be interesting for the user, two thresholds are introduced, the minimum support and the minimum confidence thresholds. An itemset is frequent if its support exceeds a user-defined minimum support threshold, σ_{min}. However the support and the minimum support threshold are defined as percentage, the algorithms convert them to an integer value (sup) using the number of transactions N. In this way calculating the support of the itemset is only counting its occurrences in the transactions, and it can be easily compared to the integer minimum occurrence threshold ($minsup$). The rules are created only from frequent itemsets. The rule is only a valid rule if its confidence exceeds the minimum confidenc threshold ($minconf$).

3 Basic Algorithms

The frequent itemset mining algorithms can be classified regarding several aspects. One of the most distinctive features of the methods is whether they use candidates. Another aspect of the classification can be the number of the database scans because of the high cost of the I/O activity. Regarding these aspects two basic algorithms are explained in this paper whose approaches are fundamentally different. The first algorithm, introduced in Subsection 3.1, is the Apriori algorithm that is a basic level-wise method and uses candidates to discover the frequent itemsets. The other algorithm, presented in Subsection 3.2, is the FP-growth algorithm which is a two-phase method and does not use any candidates to generate the patterns. The two algorithms are selected for presentation because their importance in the data mining field. Before introducing the algorithms in detail some assumptions are needed to explain. These assumptions without loss of generality make easier to handle the problem. Firstly it is assumed that the items are presented with continuous integers. The other supposition is that the items are in lexicographic order both in the transactions and in the candidates. If it is not the case the conversions can be done during a preprocessing step.

3.1 Apriori algorithm

The most commonly known, and the first presented association rule mining algorithm is the Apriori algorithm introduced by Agrawal et al in [2]. Since its introduction several other algorithms were presented which are based on it.

[1] Short version of this paper is presented in [14].

The main idea of the algorithm is based on the a priori hypothesis, namely, an itemset can only be frequent if all its subsets are also frequent. In other words, if an itemset is not frequent, no superset of it can be frequent. Exploiting this knowledge makes possible to reduce the search space efficiently when discovering the frequent itemsets, because using this knowledge the number of the candidates can be reduced. The Apriori algorithm is a level-wise method, which means that it discovers the k-itemsets during the k^{th} database scan.

The algorithm works as follows. During the first database scan the items in the transactions are counted and the infrequent ones are discarded. In this way the frequent 1-itemsets are found. From these frequent items two candidates are generated by creating all the combination of them by keeping the lexicographic order. Formally, the items x and y form a candidate (x, y) when $x \leq y$. During the second database scan the support of the 2-candidates are counted. After a database reading the counters of the candidates are checked whether they are over the minimum support threshold. If a value of a counter exceeds the threshold, the candidate belonging to it becomes frequent, otherwise it is filtered out. The 3-candidates are generated from the frequent 2-itemsets regarding the following rule. Let be given two itemsets (i_1, i_2) and (i_3, i_4) where $i_1 < i_2$ and $i_3 < i_4$ as mentioned earlier. The two itemsets can form a 3-candidate if $i_1 = i_3$ and $(i_2, i4)$ is also frequent. Fulfilling the second condition means that the a priori hypothesis is fulfilled. The resulting 3-candidate is the following: (i_1, i_2, i_4). In general two k-itemsets are joined by keeping the lexicographic order to form a $(k+1)$-itemset if the first $k-1$ items of them are in common and all the $(k-1)$-subsets of the resulting candidate are frequent as well. The algorithm terminates if no candidates can be generated or no frequent itemsets are found. The pseudo code of the algorithm is depicted in Table 1 and Table 2.

```
procedure Apriori(minsup)
L_1 = find frequent 1-itemsets
for (k = 2;L_{k-1} ! = null;k++)
   C_k = AprioriGen(L_{k-1})
   for each transaction t do
      C_t = subset(C_k,t)
      for each candidate c in C_t do
         c.counter++
      for each c in C_k do
         if c.counter >= minsup then
            L_k.Add(c)
return C_k
```

Table 1: Pseudo code of the Apriori algorithm

3.2 FP-growth algorithm

One of the algorithms which do not use any candidates to discover the frequent patterns is the FP-growth (Frequent Pattern Growth) algorithm proposed in [10]. The other main difference to the Apriori algorithm is the number of the database readings. While the Apriori is a level-wise algorithm the FP-growth is a two-phase method. It reads the database only twice and stores the database in a form of a tree in the main memory.

```
procedure Apriori(minsup)
for each itemset l_1 in L_{k-1}do
   for each itemset l_2 in L_{k-1} do
      if l_1[1] = l_2[1]
         and l_1[2] = l_2[2]
         and ...and l_1[k-2] = l_2[k-2]
         and l_1[k-1] < l_2[k-1]
      then
         c = l_1 join l_2
         if c has infrequent subset
         then DELETE c
         else C_k.Add(c)
return C_k
```

Table 2: Pseudo code of the AprioriGen procedure

The algorithm works as follows. During the first database scan the number of occurrences of each item is determined and the infrequent ones are discarded. Then the frequent items are ordered descending their support. During the second database scan the transactions are read and the frequent items of them are inserted into a so-called FP-tree structure. In this way the database is pruned and is compressed into the memory. The aim of using the FP-tree is to store the transactions in such a way that discovering the patterns can be achieved efficiently.

Each node in the tree contains an item, a counter to count the support, and links to the child nodes, to the parent nodes and to the siblings of the node. The rule for constructing the FP-tree is as follows. When reading a transaction its infrequent items are omitted and the frequent ones are ordered regarding their support. The transaction is then inserted into the tree. If the tree is empty the transaction is inserted as the only branch in the tree. If it is not empty, while the first k items of the transaction fit the prefix of one of the branches of the tree, a counter is incremented in each referred node in the tree. From the $(k+1)^{th}$ item, a new branch is created as a child of the node, which corresponds to the k^{th} item in the transaction, and the further items in the transactions are inserted as this new branch with a support counter set to one. A header belongs to the FP-tree which contains the sorted 1-frequent items, their supports and a pointer to the first occurrence of the given item in the tree. The other occurrences of the given item in the tree are linked together sequentially as a list.

The FP-tree is processed recursively by creating several so-called conditional FP-trees. This is the recursive pattern growth method of the algorithm. When a conditional FP-tree contains exactly one branch the frequent itemsets are generated from it by creating all the combinations of each

items. When traversing the whole FP-tree, all the frequent itemsets are discovered. The pseudo code of the FP-growth algorithm is depicted in Table 3.

```
procedure FPGrowth(Tree, α)
if Tree contains a sigle path P then
   for each β = comb. of nodes in P do
      pattern = β ∪ α
      sup = min(sup of the nodes in β)
else
   for each a_i in the header of Tree do
      generatepattern = β ∪ α
      sup = a_i.support
      construct β's conditional pattern base
      FPTree = construct β's
                  conditional FP-tree
      if FPTree != 0 then
         FPGrowth(FPTree,β)
```

Table 3: Pseudo code of the FP-growth algorithm

4 Comparison of the Algorithms

The experimental results presented in this paper are performed on semantic datasets generated by the dataset generator downloaded from the IBM website. The datasets generated with this program accomplish the conditions introduced in [2]. The algorithms were implemented in $C\#$. The simulations were executed on a Pentium 4 CPU, 2.40 GHz, and 1GB of RAM computer on .NET Framework v1.1. The naming conventions of the datasets are shown in Table 4. The number of the items that can occur in the transactions is 1000.

Parameter	Meaning
T	Average length of the transactions
I	Average size of maximal frequent itemsets
D	Number of transactions
K	Thousand

Table 4: Meaning of the parameters in the names of the datasets

In order to find a more effective algorithm to solve the frequent itemset mining problem in a given range of the parameters the behavior of the most representatives algorithms should be investigated. After detecting their drawbacks a novel method can be developed which aim is to avoid the disadvantages found by the algorithms examined. The objectives of the investigation are the execution time behavior and the memory requirements of each methods.

A major aspect of the examination is which parameters of the dataset affect the behavior of the algorithms significantly. The two main parameters of the datasets are the number of items that can appear in the transactions, denoted with n, and the number of transactions, denoted with T.

Fig. 1 shows the execution times of the two algorithms as a function of the number of transactions. It can be easily concluded that the execution time dependency of the Apriori algorithm on the number of transactions is linear, and that of the FP-growth algorithm is rather a polynomial of two degree. The memory requirement of the two algorithms is depicted in Fig. 2 as a function of the number of transactions. It is obvious, that the memory requirement of the Apriori algorithm does not depend on the number of transactions. The reason for that can be found in the "candidate generate and test" approach. The number of the candidates does not depend on the number of transactions; it depends only on the item number and on the minimum support threshold.

The memory requirement of the FP-growth algorithm increases significantly with the growth of the number of transactions. The reason for this can be found when examining the sizes of the trees which are generated by the algorithm. If the algorithm mines two datasets with the same statistical properties but the one contains an order of magnitude more transactions than the other, the first FP-tree built by the FP-growth algorithm contains an order of magnitude more nodes in the former case than in the latter. However the rules that have been found are nearly the same. From this fact we can draw the conclusion that several redundant nodes are in the FP-tree when increasing the number of the transactions. The claim is laid to modify the algorithm so that the created tree does not contain as redundant nodes as in the original case. The function between the number of transactions and the size of the first generated tree is linear, which is shown in Fig. 3 by different minimum support thresholds.

The advantage of the FP-growth algorithm is the quick mining process which does not use candidates. Its drawback is, however, that the memory requirement of the algorithm is huge, especially by lower minimum support threshold. The main problem of the "candidate generate and test" methods is the computational cost when filtering out the infrequent itemsets. Fig. 4 shows the execution time of the Apriori algorithm by itemset levels when using T20I7D200K dataset. When investigating the execution times by itemset levels the fact is proved that the algorithm uses most of its time to discover the small frequent itemsets. In general it uses more than 70% of its execution time to discover the 4-frequent itemsets, and more than 50% of this time is used to find the 2-frequent itemsets. Its reason is the huge number of candidates in the first four levels. The candidate numbers in each single level are depicted in Fig. 5. It can be seen well that the number of the candidates in the second level is two orders of magnitude higher than in the further levels, however the number of the frequent itemsets, depicted in Fig. 6, are about the same.

The Apriori algorithm stores the candidates in a hash tree

in order to quick find those candidates, which are to be checked whether they are contained by a certain transaction. The benefit of using a hash-tree is to reduce the number of candidates to be checked when processing a transaction. During a database scan each transaction is processed and its subsets are checked whether a counter belongs to it in the hash tree or not. This method is faster than finding the candidates by linear search, but in case of huge candidate number, using a hash tree is inefficient. The number of the database accesses of the Apriori algorithm equals to the size of the maximal frequent itemset. It accesses the database k times even than when only one k-frequent itemset exists. If the dataset is huge, the multiple database scans can be one of the drawbacks of the Apriori algorithm.

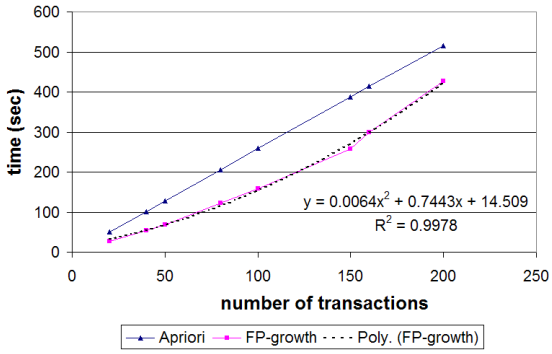


Figure 1: Execution time of the two algorithms as a function of the number of transactions by 0.9% minimum support threshold

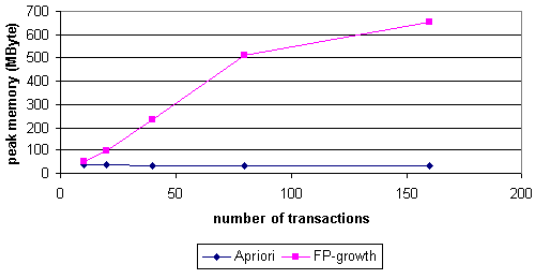


Figure 2: Peak memory of the two algorithms as a function of the number of transactions by 0.9% minimum support threshold

5 Cubic algorithm

The previous section describes the advantages and the disadvantages of the Apriori and the FP-growth algorithms. The main motivation of the novel method, called Cubic, is to enhance the aforementioned algorithms both regarding the execution time behavior and the memory requirement.

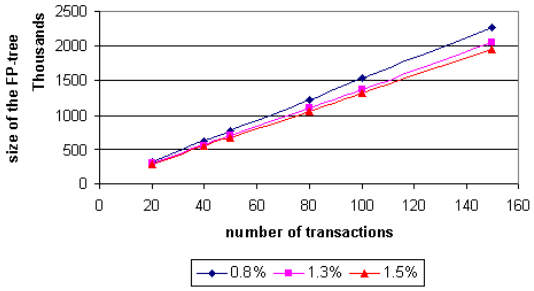


Figure 3: Sizes of the first generated FP-tree as a function of the number of transactions by 0.8%, 1.3% and 1.5% minimum support thresholds

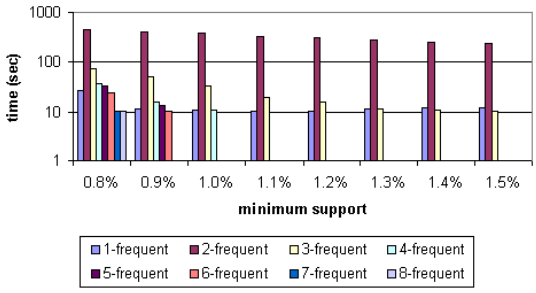


Figure 4: 7 Execution time on each level of the Apriori algorithm when using T20I7D200K dataset

The aim was to develop an algorithm whose memory usage is significantly lower than that of the FP-growth algorithm, and its execution time is smaller than the execution times of both of the algorithms described earlier. The new method is based on the Apriori algorithm, its aim is to enhance the discovering of the small patterns. Thus the novel method is faster than the introduced algorithms especially in those cases when the characteristics of the dataset shows much more small patterns than long ones.

5.1 Description of the algorithm

The Cubic algorithm is a novel method to find the frequent 4-itemsets quickly. It discovers the 4-itemsets in only two database scans. During the first disk access the support of the one and two itemsets are counted using an upper triangular matrix M. If n stands for the cardinality of the items in the database, then the size of M equals to $\frac{n(n+1)}{2}$. The diagonal elements of the matrix contains counters for the items, and the other cells are counters for the item pairs. The support counting can be achieved by a direct indexing method using the matrix and in this manner it is the fastest way.

The three and four frequent itemsets are counted during a further database scan. For efficient counting the support of the candidates their counters are stored in an index table-

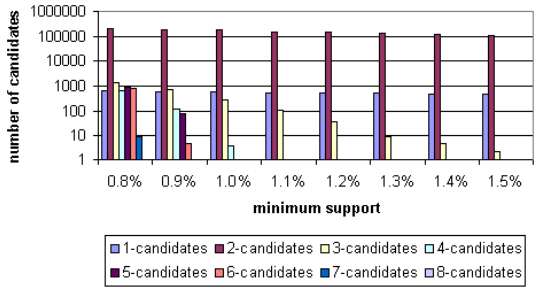


Figure 5: Sizes of the candidates in each level when using T20I7D200K dataset

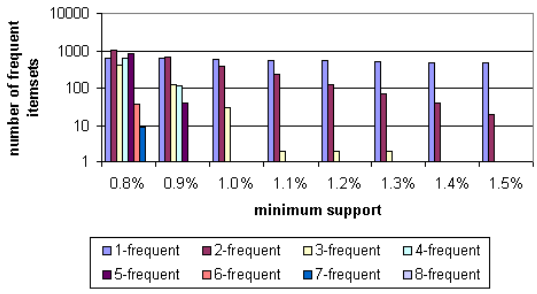


Figure 6: Sizes of the frequent itemsets in each level when using T20I7D200K dataset

based cubic structure. This is built when traversing the matrix M. A cube is created for those rows of the matrix whose value of the diagonal element is over the minimum support threshold. It means, that one cube is created in order to store the 3 and 4-candidates which belong to the frequent 2-itemsets beginning with the same item. In this manner the first item of a candidate selects the appropriate cube and the further items addresses the cells in the cube.

The matrix M is processed by rows. The i^{th} row is only processed, if the value of the i^{th} diagonal element in M is greater than the minimum support threshold. In this case a new index table is created with size of n, and the values in the i^{th} row are checked whether they are over the threshold or not. If $M[i,j] > minsup$, $(i < j)$, the j^{th} element in the index structure is set to the number which will later index the cube. If all the elements of the i^{th} row are checked, a cube is created. The size of the cube is the number of the frequent item pairs in the i^{th} row. A reverse index is created as well, in order to easy converting the index value, which addresses the cells in the cube, to the original item when traversing the cubes. This is used by the counter checking process. During the second database scan every 3 and 4-subsets of the transactions are created, which has at least one 2-frequent subset, and the appropriate element in the cube is incremented. The cube is selected by the first item of the subset. The other items address the counters in the cube using the index structure belonging to the selected cube.

The Apriori hypothesis is used only partially because of the following reason. The Apriori assumption is exploited when the algorithm creates different cubes for the itemsets having different first item. However it is not used when the edges of the cube are created. If the value of $M[i,j]$ is greater than the minimum support threshold, the item j is added to the index table of the cube independently whether the elements $M[j,s]$, $(i < s < n)$ are greater than the minimum support or not, where s denotes those items which satisfy the $M[i,s] > minsup$ condition. The reason for this is that the storage space for the cube is rather compact, and there would not be any benefit discarding these items. In addition it would take more time to discard the item than to count its support. The main parts of the algorithm are depicted in Table 5 and Table 6.

The Cubic algorithm discovers the 4-frequent itemsets. The further itemsets can be found in different ways. One of the possibilities is a level-wise approach, which simply invokes the Apriori algorithm. This is the easiest way and often a very quick solution because the Apriori algorithm finds the itemsets with cardinality greater than five relatively quick. This algorithm is called Cubic-Apriori.

Another way is to call the FP-growth algorithm after discovering the frequent 4-itemsets. The FP-tree should be created by leaving out those transactions, which do not contain frequent 4-itemsets. So the basic idea of the suggested Cubic FP-growth algorithm is that there is no need to build a much larger tree, if the rules are contained also in a smaller. In this case the FP-tree must be generated only from those transactions, which contains at least one 4-frequent itemset. In this way the profit is the smaller tree generated by the FP-growth algorithm, thus, in general, the execution time is enhanced as well. In addition only one additional database scan is needed in this case than in case of using the original FP-growth algorithm.

5.2 Simulation results

In Fig. 7 the execution time of the four algorithms is analyzed when using T20I7D200K dataset. It is clear, that the Cubic method continued by the Apriori algorithm, called Cubic Apriori algorithm, is the fastest of all the four methods. The execution time of the Cubic FP-growth method is always smaller than that of the Apriori algorithm but it is not always smaller, than the execution time of the FP-growth algorithm. The reason for that is illustrated in Fig. 8. The sizes of the first generated FP-trees are depicted in it in cases of the FP-growth and of the Cubic FP-growth algorithms when using T20I7D200K dataset as a function of the minimum support threshold. Apparently the sizes of the tree in case of small minimum support thresholds are near to each other, moreover by minimum support threshold of 0.5% they are about the same. It means that the Cubic FP-growth algorithm has to accomplish about the same recursive pattern growth process as the FP-growth algorithm does, but before this, the Cubic FP-growth al-

```
procedure FillCubes()
for each transaction t do
  for (i=0; i < t.count; i++)
    if ixStruct[t[i]] = null then
      continue
    for (j=i + 1; j < t.count; j++)
      if M[t[i], t[j]] < minsup then
        continue
      ix1 = IxStruct[t[i]][t[j]]
      for (k = j + 1; k < t.count; k++)
        ix2 = IxStruct[t[i]][t[k]]
        if ix2! = −1 then
          CubeL[t[i]][ix1, ix2, 0]++
            for (l = k + 1; l < t.count; l++)
              ix3 = IxStruct[t[i]][t[l]] + 1
              if ix3! = 0 then
                CubeL[t[i]][ix1, ix2, ix3]++
```

Table 5: Pseudo code of the candidate counting procedure of the Cubic algorithm

```
procedure CheckCubes()
rI=reverseIndexTable.Clone()
for (i=0;i < cubeL.count;i++)
  if cubeL[i]!=null then
    for (j=0;j < rI[i].count;j++)
      for (k=j + 1;k < rI[i].count;k++)
        if cubeL[i][j; k; 0] >= minSup
        then
          item2 = rI[i][j]
          item3 = rI[i][k]
          L3.Add(i, item2, item3)
          for (l=k + 1;l < rI[i].count;l++)
            if cubeL[i][j, k, l] > minSup
            then
              item2 = rI[i][j]
              item3 = rI[i][k]
              item4 = rI[i][l − 1]
              L4.Add(i, item2, item3, item4)
```

Table 6: Pseudo code of the candidate checking procedure of the Cubic algorithm

gorithm has also to mine the 4-frequent itemsets using the Cubic method. In this case filtering the transactions by using the results of the Cubic algorithm causes no significant profit regarding the number of nodes in the tree. The saving in the node number is rather by minimum support threshold higher than 0.7%. In Fig. 9 the peak memory sizes are illustrated as a function of the number of transactions when the average size of the maximal frequent items is 7 and the average size of the transactions is 20. The minimum support threshold is set to 0.9%. It is shown, that the memory requirement of the Cubic Apriori algorithm does not depend on the number of transactions.

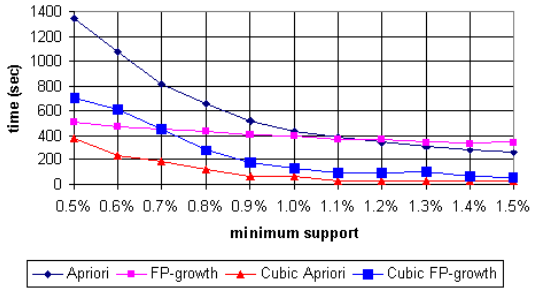


Figure 7: Execution time of the four algorithms when using T20I7D200K

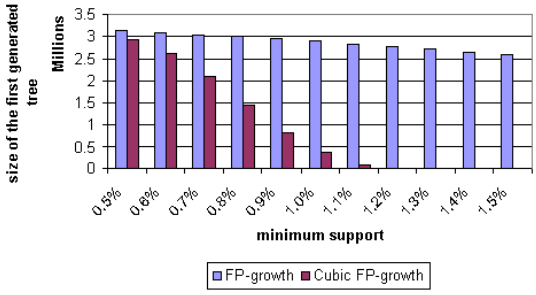


Figure 8: Sizes of the first generated tree of the FP-growth and of the Cubic FP-growth algorithm when using T20I7D200K

6 Conclusion

This paper is concerned with the problem of efficiently discovering frequent itemsets in transactional databases. The algorithms dealing with this type of data mining problem can be divided into several classes regarding their behavior. The two most representative classes are the one which contains the level-wise methods and the class that contains the two-phase methods. Two basic algorithms of these classes were explained in detail. After investigating the execution time behavior and the memory requirement of the Apriori and the FP-growth algorithm the advantages and disadvantages of them were illustrated. The main drawback of the Apriori algorithm is its relatively slow candidate testing method using the hash-tree data structure in case of small candidates, when the number of these candidates is high. The memory requirement dependency on the number of transactions is proved as the major problem of the FP-growth algorithm.

A novel method, the Cubic algorithm is presented in order to enhance the Apriori algorithm by finding the short frequent patterns quickly, using an index table-based cubic structure. The algorithm exploits the benefits of direct indexing over the hash tree-based searching. Experimental

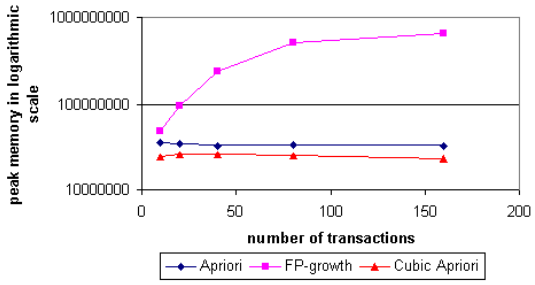


Figure 9: Peak memory of the algorithms as a function of the number of transactions by 0.9% minimum support threshold

results show the time saving when replacing the first four steps of the Apriori algorithm with the novel method. In this way, the Cubic Apriori algorithm is even faster than the FP-growth algorithm, and the memory requirement of the novel method does not depend on the number of transactions.

Using the Cubic algorithm the performance of the FP-growth algorithm can be enhanced as well. When a preprocessing step is inserted before the FP-growth algorithm, namely discovering the frequent 4-itemsets using the Cubic algorithm, the size of the FP-tree can be reduced. In this case the memory requirement is reduced.

Acknowledgement

This work has been supported by the fund of the Hungarian Academy of Sciences for control research and the Hungarian National Research Fund (grant number: T042741)

References

[1] R. Agrawal, T. Imielinski and A.Swami (1993) Mining association rules between sets of items of large databases, *Proc. of the ACM SIGMOD Intl'l Conf. On Management of Data*,Washington D.C.,USA, pp. 207–216.

[2] R. Agrawal and R. Srikant (1994) Fast algorithms for mining association rules, *Proc. 20th Very Large Databases Conference*, Santiago, Chile, pp. 487–499.

[3] J. S. Park, M. Chen, and P. S. Yu (1995) An effective hash based algorithm for mining association rules, *Proc. of the 1995 ACM Int. Conf. on Management of Data*, San Jose, California, USA, pp. 175–186.

[4] S. Brin, R. Motawani, J.D. Ullman and S. Tsur (1997) Dynamic Item set counting and implication rules for market basket data, *Proc of the ACM SIGMOD Intl'l Conf. On Management of Data*, Tucson, Arizona, USA, pp. 255–264.

[5] M. J. Zaki (2000) Scalable algorithms for association mining, *IEEE Transaction on Knowledge and Data Engineering. Vol 12. No 3. May/June 2000*, pp. 372–390.

[6] V. S. Ananthanarayana, D. K. Subramanian and M. N. Murty (2000) Scalable, distributed and dynamic mining of association rules *Proceedings of the 7th International Conference on High Performance Computing - HiPC 2000*, Bangalore, India, pp. 559–566.

[7] R. J. Bayardo (1998) Efficiently mining long patterns from databases *Proceedings of the ACM SIGMOD international conference on management of data*, Seatle, WA, pp 85–93.

[8] P. Shenoy, J.R. Haritsa, S. Sundarshan, G. Bhalotia, M. Bawa and D. Shah (2000) Turbo-charging vertical mining of large databases *Proceedings of the ACM SIGMOD*, Dallas, TX, pp. 22–33.

[9] R Ivancsy, F. Kovacs and I. Vajk (2004) An Analysis of Association Rule Mining Algorithms, *In CD-ROM Proc. of Fourth International ICSC Symposium on Engineering of Intelligent Systems (EIS 2004)*, Island of Madeira, Portugal.

[10] J.Han, J. Pei and Y. Yin (2000) Mining frequent patterns without candidate generation, *Proc. of the 2000 ACM-SIGMOD Int'l Conf. On Management of Data*, Dallas, Texas, USA, pp. 1–12.

[11] M. El-Hajj and O. R. Zaïane (2003) Non Recursive Generation of Frequent K-itemsets from Frequent Pattern Tree Representations, *Proc. of 5th International Conference on Data Warehousing and Knowledge Discovery (DaWak'2003)*, Prague, Czech Republic

[12] J. Pei, J. Han, H. Lu et al (2001) H-Mine: Hyper-structure mining of frequent patterns in large databases, *In Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01)*, San Jose, California, pp. 441–448.

[13] Q. Zou, W. Chu, D. Johnson and H. Chiu (2002) Pattern decomposition algorithm for data mining of frequent patterns *Journal of Knowledge and Information System, Volume 4, Issue 4* pp. 466–482.

[14] R. Ivancsy and I. Vajk (2004) Fast Discovery of Frequent Patterns in Market Basket Data *In. Proc. of 4th International Conference on Intelligent Systems Design and Applications (ISDAŠ04)*, Budapest, Hungary, pp. 575–580.

Similarity Search in Time Series Data Using Time Weighted Slopes

Durga Toshniwal and R. C. Joshi
Department of Electronics and Computer Engineering
Indian Institute of Technology
Roorkee, India
E-mail: durgadec@iitr.ernet.in, joshifcc@iitr.ernet.in

Keywords: Data Mining, Knowledge Discovery, Time Series Data, Similarity Search, Trend Analysis, Variation in Slopes

Received: November 30, 2004

Similarity search in time series data is an area of active interest in the data mining community. In this paper we introduce a novel approach for performing similarity search in time series data. This technique is based on the intuition that similar time sequences will have similar variations in their slopes and consequently in their time weighted slopes. The proposed technique is capable of handling variable length queries and also works irrespective of different baselines and scaling factors.

Povzetek: Opisana je nova metoda rudarjenja podatkov časovnih vrst z iskanjem podobnosti.

1 Introduction

A large portion of scientific and business data stored on computers is comprised of time series data. Some typical examples include stock indices, biomedical data, retail data and atmospheric data. During the past few years, there has been an explosion of research in the area of time series data mining. This includes attempts to model time series data, to design languages to query such data, and to develop access structures to efficiently process queries on such data. The problem of similarity search in time series data is important and non-trivial.

To perform similarity search on time series data, indexing methods that are capable of supporting efficient retrieval and matching of time series data are required. Most of the indexing methods available today for multi-dimensional data such as the R-tree [1] and the R*-tree [2] degrade performance at dimensionalities greater than 8-10 [3] and eventually perform almost like sequential scanning algorithms at high dimensionalities. Thus, to utilize multi-dimensional indexing techniques, it is essential to first perform dimension reduction on time series data. This helps to map the high-dimensional data to a lower dimension space. Then some distance measure such as the Euclidean Distance may be used to calculate the distance and hence the similarity between any two time sequences.

Most of the approaches developed so far for performing similarity search in time series data are based on dimension reduction. Dimension reduction can be performed by several ways. Some commonly used methods for performing dimension reduction include Discrete Fourier Transform (DFT) [4, 5, 6, 7], Discrete Wavelet Transform (DWT) [8, 9, 10, 11, 12], Singular Value Decomposition (SVD) [13] and Piecewise Aggregate Approximation (PAA) [14].

The most frequently used method for dimension reduction is based on the DFT. The DFT is quite suited for naturally occurring sinusoidal signals but it is ill-suited for representing signals having discontinuities.

The Haar wavelet transform is the most commonly used wavelet transform for dimension reduction. But the basis function for Haar is not smooth. Thus the Haar wavelet transform approximates any signal by a ladder like structure. Hence the Haar wavelet transform is not likely to approximate a smooth function using only a few coefficients. So the number of coefficients to be added must be high. Finding wavelets having more continuous derivatives is still an active area of research.

The SVD technique is a data dependent dimension reduction technique. It uses the KL transform for performing dimension reduction. The given data is used to compute basis vectors. So whenever the database is updated, the basis vectors need to be recomputed. The recomputation time may become infeasible for practical purposes especially when the database is very large.

The PAA performs dimension reduction by dividing the time sequences into equal length segments. The corresponding feature sequence comprises of mean values of each segment. But the means representing each segment give only a rough approximation of each time sequence.

In this paper, we introduce a new approach for similarity search in time series databases. We assume that a time series comprises of samples of a single measured variable against time. The proposed approach is based on the observation that similar time sequences will have similar variations in their slopes and hence time weighted slopes. By time weighted slopes we mean that the slope

is assigned a weight depending upon its location along the time axis. The technique being proposed involves some data pre-processing that enables it to handle variable length queries. It is also capable of handling global scaling and shrinking of the data and works irrespective of vertical shifts that may exist between the given time sequences. Further, it does not rely on any kind of dimension reduction.

2 Related Work

In this section we discuss some key approaches for performing similarity search in time series data.

Agrawal et al. [4] used the Discrete Fourier Transform to perform dimension reduction. The DFT was used to map the time sequences to the frequency domain and the index so built was called the F-index. For most sequences of practical interest, the low frequency coefficients are strong. Thus the first few Fourier coefficients are used to represent the time sequence in frequency domain. These coefficients were indexed using the R*-tree [2] for fast retrieval. The basis for this indexing technique is Parseval's theorem. The Parseval's theorem guarantees that the distance between two sequences in the frequency domain is the same as the distance between them in the time domain. For a range query the F-index returns a set of sequences that are at a Euclidean Distance $\in$ from the query sequence.

The F-index may raise false alarms but does not introduce false dismissals. The actual matches are obtained in a post-processing step wherein the distance between the sequences are calculated in the time domain and those sequences which are within $\in$ distance are retained and the others are dismissed. The F-index typically handles 'whole matching' queries.

Faloutsos et al. generalized the F-index method in [15] and called it the ST-index. In this technique, subsequence queries are handled by mapping data sequences into a small set of multidimensional rectangles in feature space. These rectangles are indexed using spatial access methods like the R*-tree [2].

A sliding window is used to extract features from the data sequence resulting in a trail in the feature space. These trails are divided into sub-trails which can be represented by their Minimum Bounding Rectangles (MBR). Thus, in place of storing all the points in a trail, only a few MBRs are stored. When a query is presented to the database, all the MBRs intersecting the query region are retrieved. This guarantees no false dismissals but also raises some false alarms as sub-trails that do not intersect the query region but their MBRs are also retrieved.

Chan et al. [8] have proposed to use the DWT in place of DFT for performing dimension reduction in time series data. Unlike the DFT which misses the time localization of sequences, the DWT allows time as well as frequency localization concurrently. The DWT thus bears more information of signals in contrast to DFT in which only frequencies are considered. The approach in [8] employed the Haar Wavelet Transform for mapping high-dimensional time series data to lower dimensions.

A data dependent indexing scheme was proposed in [13] and is known as the SVD method for dimension reduction. The database consists of n-dimensional points. We map them on a k-dimensional subspace, where $k < n$, maximizing the variations in the chosen dimensions. An important drawback of this approach is the deterioration of performance upon incremental update of the index. Therefore the new projection matrix should be calculated and the index tree has to be reorganized periodically to keep up the search performance.

In PAA [14], each time sequence say of length k is segmented into m equal length segments such that m is a multiple of k. If that is not the case, then the sequence is padded with zeros in order to perform the segmentation. The averages of segments together form the new feature vector for the sequence. The correct selection of m is very important because if m is very large, the approximation becomes very rough but if m is very small, the performance deteriorates.

Mostly similarity search methods utilize the Euclidean distance model for calculating the similarity between the query and candidate sequence. According to this model, if the Euclidean Distance $D\ (X, Y)$ between two time sequences X and Y of length n is less than a threshold $\in$, then the two sequences are said to be similar. The Euclidean Distance is given as:

$$D\ (X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad (1)$$

A major shortcoming in the Euclidean distance model is that it is not able to handle vertical shifts existing between the time sequences under comparison.

Toshniwal et al. [16] have used the cumulative variation of slopes for computing the similarity in the given time series data. In this technique the data is first pre-processed. Next, each of the time sequence is divided into same number of small, equi-width strips as shown in Figure 1. The cumulative variation in slopes is then computed as the parameter $S\ (Q, C)$ where Q and C are the two time sequences under comparison. Mathematically:

$$S(Q, C) = \sqrt{\sum (S_{qj} - S_{cj})^2} \quad (2)$$

S_{cj} and S_{qj} are the slopes for the j^{th} strip in the candidate time sequence C and the query time sequence Q respectively.

Ideally, for exactly similar time sequences, the parameter S *(Q, C)* would be zero. Practically, the smaller the value of *S (Q, C),* the more is the similarity between the time sequences under comparison. For range queries and nearest-neighbour queries we may choose to have *S (Q, C)* $\leq$ / where / specifies some degree of tolerance allowed while performing similarity search in the time-series database.

In this paper, we present an approach for similarity search in time series data which is an improvement over [16]. The technique proposed in this paper is also based on the concept that similar sequences will have similar variations in their slopes. In [16], the cumulative difference between the slopes of the query and the candidate time sequences has been computed as given by (2). In the present study, weights have been given to the locations of the slopes along the time axis. In [16] the square of the differences in the slopes has been used for computing the parameter for similarity *S (Q, C)*. However, in this paper, the sign of the slopes have also been accounted for while computing the cumulative variation in slopes by using a cubic function as in (4).

3 Proposed Approach

The cumulative variation in time weighted slopes has been used in this paper for performing similarity search in time series data. Here, we assume that a time series consists of a sequence of real numbers which represent the values of a measured parameter at equal intervals of time. Let the time series database consist of p time sequences designated by $X_1, X_2 \ldots X_p$. Each time sequence X_i in turn can be represented as $< (t_{i1}, y_{i1}), (t_{i2}, y_{i2}) \ldots (t_{in}, y_{in}) >$ where n is the number of samples in the time sequence.

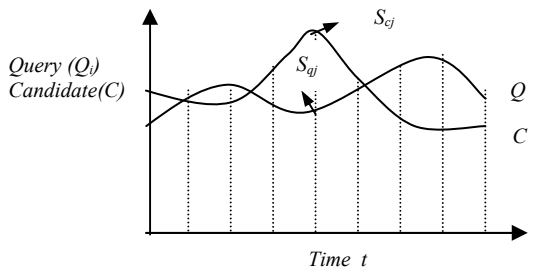


Figure 1: Query and candidate time sequences divided into strips.

In the proposed approach, each of the candidates X_i in the time series database is first scaled along the time axis so that their time axes become equal to some desired time t_d. The selection of t_d is done by the user and may depend on the domain of application of the data. In our technique, scaling along the time axis is done to equalize the time durations of candidates and query. This helps to compare variable length time sequences. For example, a 5-year sales pattern of a Product A can be compared to a 10-year sales pattern of Product B. Another example where scaling can play a crucial role is the comparison of the growth of a tumour for the past 10-months versus the growth of the tumour for past 10-days. In order to avoid any distortions that may arise due to time scaling, the values along the y-axis for each X_i are also scaled proportionately. Thus each transformed X_i denoted by X_i' may be represented as $< (t_{i1}', y_{i1}'), (t_{i2}', y_{i2}') \ldots (t_{in}', y_{in}') >$ where:

$$t_{ij}' = t_{ij}' * (t_d / t_{in}) \quad \text{and} \quad y_{ij} = y_{ij} * (t_d / t_{in}) \quad (3)$$

This is followed by dividing each of the candidate time sequences in the database into same number of small, equi-width strips along the time-axis as shown in Figure 1. Thus each candidate time sequence is divided into say m number of strips.

The same procedure is repeated for any query Q for similarity search. Or in other words, the query is first time scaled to t_d and then scaled proportionately along the y-axis. The resulting sequence is divided into m number of small, equi-width strips. The strips have different heights but same widths along the time-axis as shown in Figure 1.

Finally, we compute the cumulative variation in time weighted slopes between any two sequences Q and C as:

$$WS(Q, C) = \left| \sqrt[3]{\sum_{i=1}^{m} (S_{qj} - S_{cj})^3 * t_j / t_d} \right| \quad (4)$$

where S_{cj} and S_{qj} are the slopes for the j^{th} strip (Fig. 1) in the candidate time sequence C and the query time sequence Q respectively :

$$S_{cj} = \{ y_{ic''(j+1)} - y_{icj''} \} / \Delta t \quad (5)$$

$$\text{and } S_{qj} = \{ y_{q''(j+1)} - y_{qj''} \} / \Delta t \quad (6)$$

We assume in (5) and (6) that the starting and ending coordinates for the j^{th} strip of the candidate are given by (t'_{icj}, y_{icj}'') and $(t'_{ic(j+1)}, y_{ic''(j+1)})$. Similarly, the starting and ending coordinates for the j^{th} strip of the query time sequence are given by (t'_{qj}, y_{qj}'') and $(t'_{q(j+1)}, y_{q''(j+1)})$. And Δt is the width of each of the strips and is a constant. The choice of Δt may be user specified or domain specific.

The important thing to note about the selection of Δt is that its value should be optimally selected so that it is neither too small (because that may lead to excessive computations) nor too large (loss of details). The number of equi width strips in the query as well as the candidate

time sequences is equal (Fig. 1). As the width of the strips in the query as well as the candidate time sequences are equal, Δt is given as:

$$\Delta t = t'_{ic(j+1)} - t'_{icj}$$
$$Or \quad \Delta t = t'_{q(j+1)} - t'_{qj} \tag{7}$$

The weight associated with the location of the strip along the time axis is given by the factor t_j / t_d . As the number of the strips in the query as well as the candidate sequences is equal, t_j is given as:

$$t_j = t'_{ic(j+1)} = t'_{q(j+1)} \tag{8}$$

Ideally for two exactly similar time sequences, the value of the parameter *WS (Q, C)* must be zero. Practically, the smaller the value of *WS (Q, C),* the more is the similarity between the time sequences under comparison. For range queries and nearest-neighbour queries we may choose to have *WS (Q, C)* $\leq \varepsilon$ where ε specifies some degree of tolerance allowed while performing similarity search in the time-series database.

The cube root of time weighted variations in slopes has been specially chosen to account for the positive or negative sign of the differences between the slopes of the query as well as the candidate time sequences at corresponding time locations while calculating the cumulative variation in slopes. We feel that the inclusion of the sign plays a key role while computing the cumulative variation in slopes between the query as well the candidate time sequences.

The overall strategy thus involves the following steps:
Step 1: Scaling of data along the time-axis to allow variable length queries.
Step 2: Scaling the values of y-ordinates proportionately to avoid any possibility of data distortions arising from step 1.
Step 3: Dividing each time sequences into same number of small, equi-width strips.
Step 4: Computing the parameter *WS (Q, C)* for cumulative variations in time weighted slopes of the two time sequences under comparison. Ideally, it should be zero.

4 Experimental Results

We have evaluated the performance of the proposed technique by considering synthetically generated sample time sequences as the test data. The test data has been designed specially for this purpose so as to include a variety of curves and reverse curves to demonstrate the effectiveness of the proposed approach.

The first set of sample data considered are shown in Figure 2. It comprises of *A1, A2, A3* and *A4*. The dataset has been scaled both along the x-axis and correspondingly along the y-axis taking $t_d = 5$ and $\Delta t =$ 0.385 (taken randomly) and the results are shown in Figure 3. Table 1 summarizes the results obtained by computing the parameter *WS (Q, C), S (Q, C)* and *D (Q, C)* taking *A1t* as the query and the others as the candidates. To graphically illustrate the similarity between *A1t, A2t, A3t* and *A4t*, we have shifted *A1t, A2t* and *A4t* vertically so that all of the time sequences have the same initial y-values. This is shown in Figure 4. It is clear from Figure 4 and also from the parameter *WS (Q, C)* computed in Table 1 that *A1t* is most similar to *A2t* and is most dissimilar to *A4t*. Or in other words, *A1* is most similar to *A2* and is very dissimilar to *A4*. In this case, the results of the Euclidean Distance computations and *S (Q, C)* also give the same results as can be seen from Table 1.

Next we have considered a set of reverse curves - *A1R, A2R, A3R* and *A4R* as the sample data as shown in Figure 5. The dataset has been scaled both along the x-axis and correspondingly along the y-axis taking $t_d = 5$ and the results are shown in Figure 6. Table 2 shows the results obtained by computing the parameter *WS (Q, C)* and *D (Q, C)* taking *A1Rt* as the query and the others as the candidates. To bring out the similarity between *A1Rt, A2Rt, A3Rt* and *A4Rt*, we have shifted *A1Rt, A3Rt* and *A4Rt* vertically so that all of the time sequences have the same initial y-values. This is shown in Figure 7. It is clear from Figure 7 and also from the parameter *WS (Q, C)* computed in Table 2 that *A1Rt* is most similar to *A3Rt* and is most dissimilar to *A4Rt*. Or in other words, *A1R* is most similar to *A3R* and is very dissimilar to *A4R*. As seen from Table 2, the results of the parameter *S (Q, C)* also indicate the same order of similarity for this dataset. But the Euclidean Distance computations do not give correct results.

The dataset considered next comprises of *B1, B2, B3* and *B4* as shown in Figure 8. The dataset has been scaled taking $t_d = 5$ and the results are shown in Figure 9. Table 3 shows the results obtained by taking *B1t* as the query and the others as the candidates. To show graphically the similarity between *B1t, B2t, B3t* and *B4t*, we have shifted *B2t, B3t* and *B4t* vertically so that all of time sequences have the same initial y-values. This is shown in Figure 10. It can be clearly seen from Figure 10 and also from the parameter *WS (Q, C)* computed in Table 3 that *B1t* is most similar to *B2t* and is most dissimilar to *B4t*. Or in other words, *B1* and *B2* are very similar to each other whereas *B1* is most dissimilar to *B4*. As seen from Table 3, in this case the results of the Euclidean Distance computations as well as the parameter *S (Q, C)* do not provide appropriate similarity comparisions.

The next dataset for similarity search comprises of reverse time sequences *B1R, B2R, B3R* and *B4R* as shown in Figure 11. After scaling, the resulting time sequences are shown in Figure 12 and are denoted by *B1Rt, B2Rt, B3Rt* and *B4Rt.* To graphically show the similarity, we have shifted vertically, *B1Rt, B2Rt* and *B3Rt* so that all of them lie at the same initial y-value as that of *B4Rt* and shown it in Figure 13. The *WS (Q, C), S (Q, C)* and *D (Q, C)* computations are shown in Table 4

taking *B1R* as the query and all others as the candidates. It can be seen clearly from Table 4 and Figure 13 that both the *WS* as well as the *S* parameters indicate that *B1Rt* is most similar to *B3Rt* and is very dissimilar to *B4Rt*. Or in other words, *B1R* is very similar to *B3R* and is dissimilar to *B4R*. The Euclidean Distance is not able to assess the similarity correctly in this case.

The dataset studied next consists of the time sequences *C1, C2, C3* and *C4* and is shown in Figure 14. The pre-processed data *C1t, C2t, C3t* and *C4t* are shown in Figure 15. The results of the computations of *WS (Q, C), S (Q, C)* and *D (Q, C)* are summarized in Table 5. To graphically show the similarity, we have shifted vertically, C*1t, C2t* and C*3t* so that all of them lie at the same initial y-value as that of C*4t* and shown it in Figure 16.

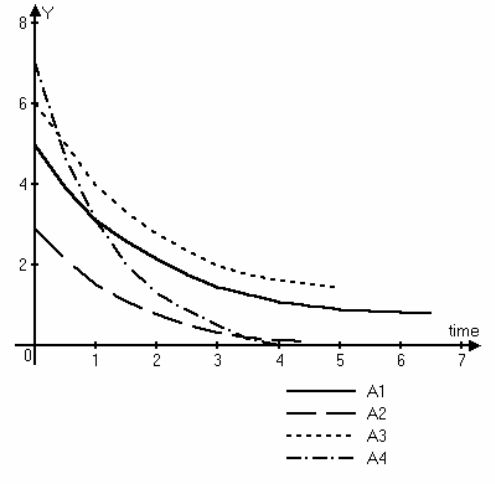


Figure 2: Time series dataset *A*.

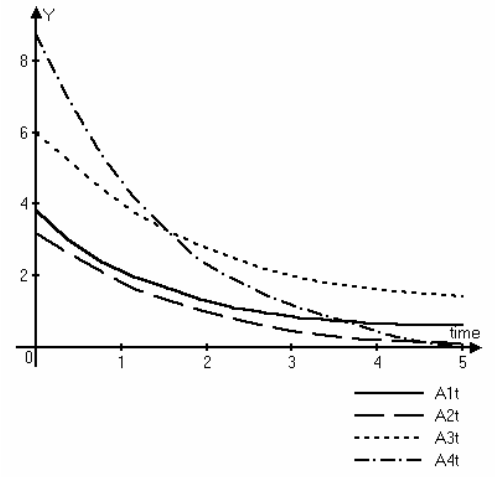


Figure 3: Scaled sequences designated by *A1t, A2t, A3t* and *A4t.*

TABLE 1
PARAMETER *WS (Q, C), S (Q, C)* VERSUS EUCLIDEAN DISTANCE *D (Q, C)*

Sequence Pairs	**Parameter *WS***	**Euclidean Distance *D***	**Parameter *S***
A1t, A2t	0.274	1.60	0.809
A1t, A3t	0.611	5.53	1.241
A1t, A4t	2.058	7.66	4.853

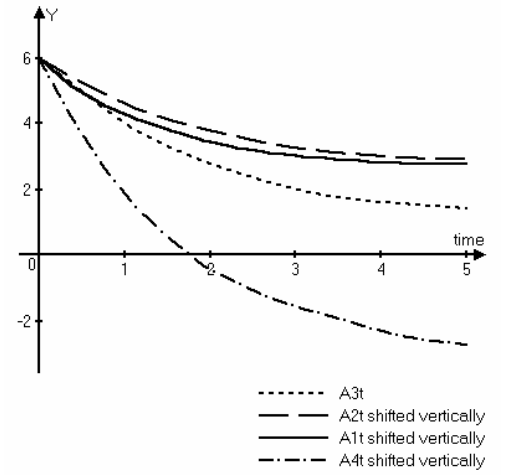


Figure 4: The sequences *A1t, A2t* and *A4t* shifted vertically.

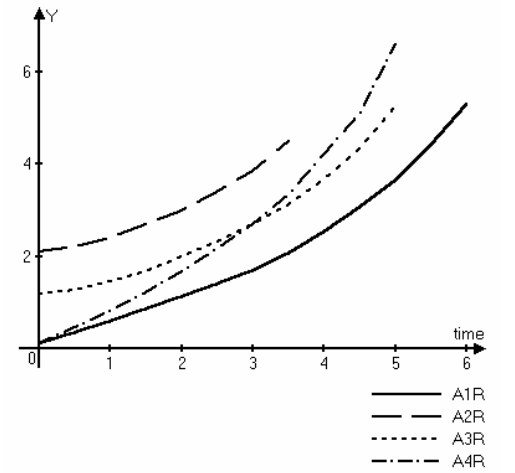


Figure 5: Time series dataset *AR*.

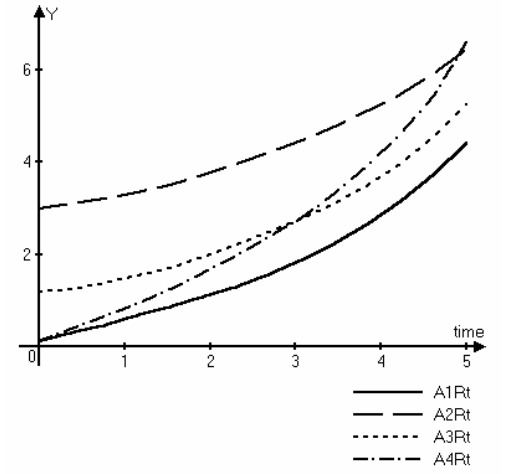


Figure 6: Scaled sequences designated by *A1Rt, A2Rt, A3Rt and A4Rt.*

TABLE 2
PARAMETER *WS (Q, C), S (Q, C)* VERSUS EUCLIDEAN DISTANCE *D (Q, C)*

Sequence Pairs	Parameter *WS*	Euclidean Distance *D*	Parameter *S*
A1Rt, A2Rt	0.566	9.59	0.813
A1Rt, A3Rt	0.151	3.30	0.432
A1Rt, A4Rt	1.359	3.87	1.830

.

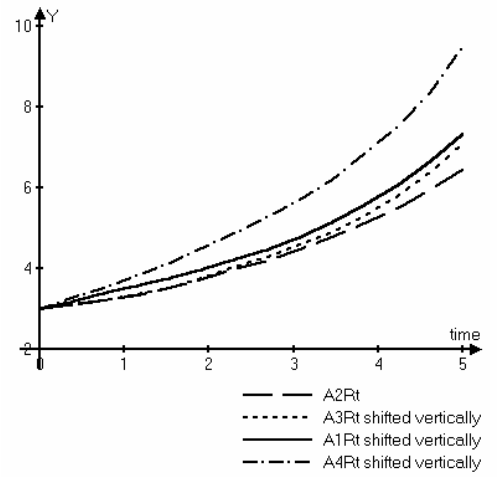


Figure 7: The sequences *A1Rt, A3Rt* and *A4Rt* shifted vertically.

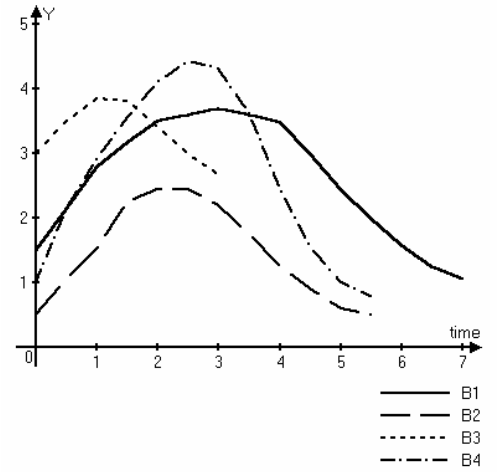
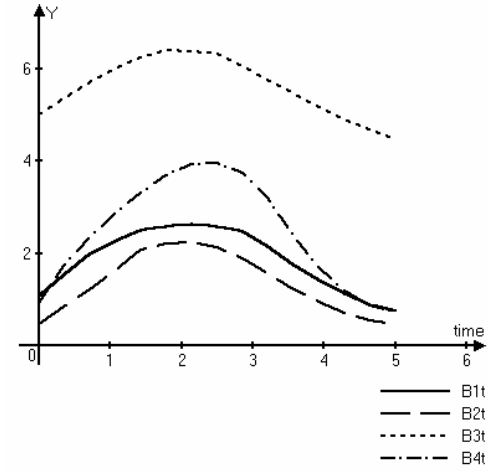


Figure 8: Time series dataset *B*.

It can be concluded from Table 5 and Figure 16 that the query *C1t* is similar to the candidates *C3t, C2t* and *C4t* in that order. While the parameter *S (Q, C)* also indicate the same results, but the Euclidean Distance model gives results which are incorrect. Thus the sequence *C1* is most similar to *C2* and least similar to *C4.*

Finally, we have considered the reverse dataset *CR* as shown in Figure 17. The pre-processed dataset is shown in Figure 18. The results have been computed in Table 6. To graphically show the similarity, we have shifted vertically, C*1Rt, C2Rt* and *C3Rt* so that all of them lie at the same initial y-value as that of C*4Rt* and shown it in Figure 19. It is clear from the parameters *WS (Q, C)* and *S (Q, C)* that *C1R* is most similar to *C3R* and least similar to *C4R*.



. Figure 9: Scaled sequences designated by *B1t, B2t, B3t* and *B4t*.

TABLE 3
PARAMETER *WS (Q, C), S (Q, C)* VERSUS EUCLIDEAN DISTANCE *D (Q, C)*

Sequence Pairs	Parameter *WS*	Euclidean Distance *D*	Parameter *S*
B1t, B2t	0.398	2.06	1.017
B1t, B3t	0.466	14.51	0.886
B1t, B4t	1.243	3.03	2.531

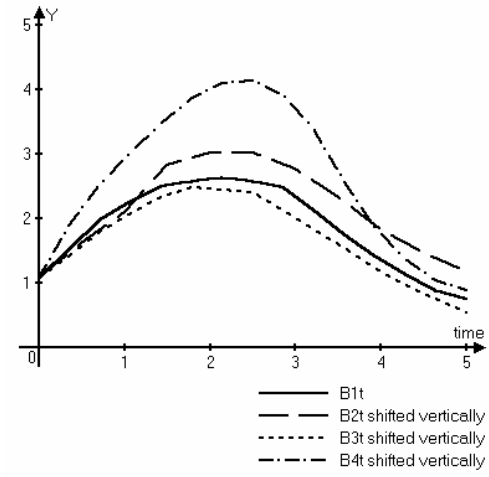


Figure 10: The sequences *B2t, B3t* and *B4t* shifted vertically.

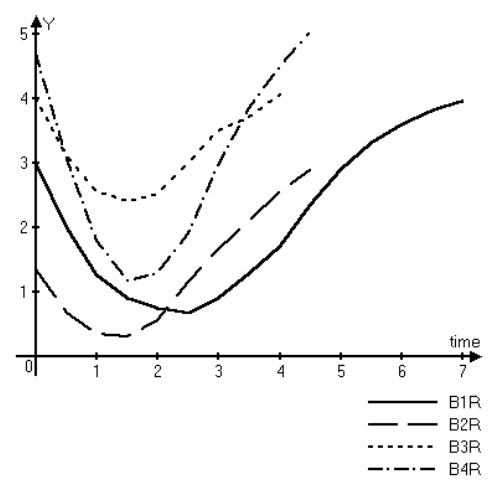


Figure 11: Time series dataset *BR*.

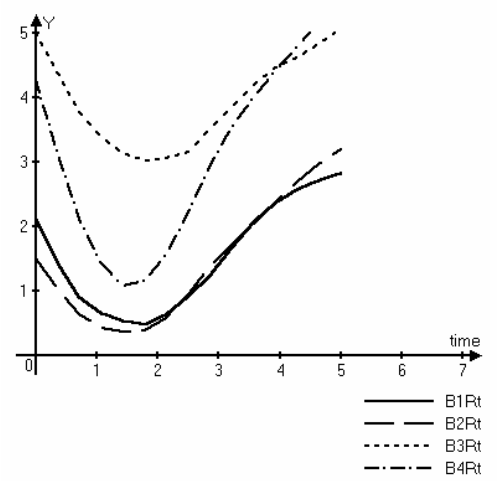


Figure 12: Scaled sequences designated by *B1Rt, B2Rt, B3Rt* and *B4Rt*.

TABLE 4
PARAMETER *WS (Q, C), S (Q, C)* VERSUS EUCLIDEAN DISTANCE *D (Q, C)*

Sequence Pairs	**Parameter *WS***	**Euclidean Distance *D***	**Parameter *S***
B1Rt, B2Rt	0.535	1.00	1.132
B1Rt, B3Rt	0.445	9.43	1.027
B1Rt, B4Rt	1.009	6.65	3.027

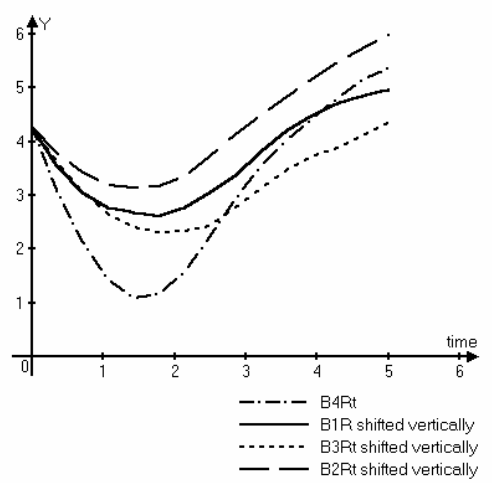


Figure 13: The sequences *B1Rt, B2Rt* and *B3Rt* shifted vertically.

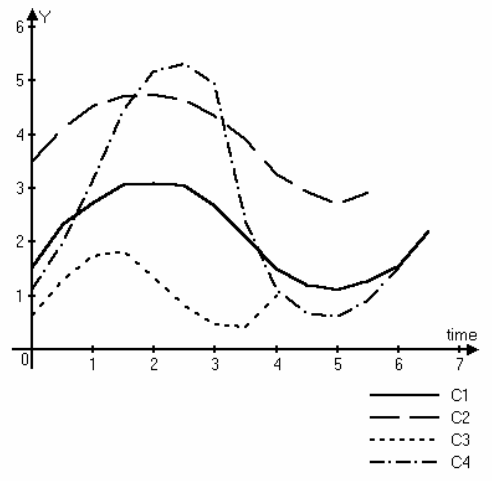


Figure 14: Time series dataset *C*.

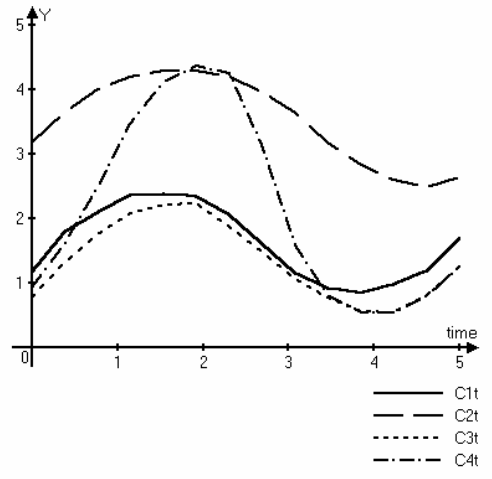


Figure 15: Scaled sequences designated by *C1t, C2t, C3t* and *C4t*.

TABLE 5
PARAMETER *WS (Q, C), S (Q, C)* VERSUS EUCLIDEAN DISTANCE *D (Q, C)*

Sequence Pairs	Parameter *WS*	Euclidean Distance *D*	Parameter *S*
C1t, C2t	1.346	7.22	2.046
C1t, C3t	0.354	1.17	0.876
C1t, C4t	2.571	4.07	4.784

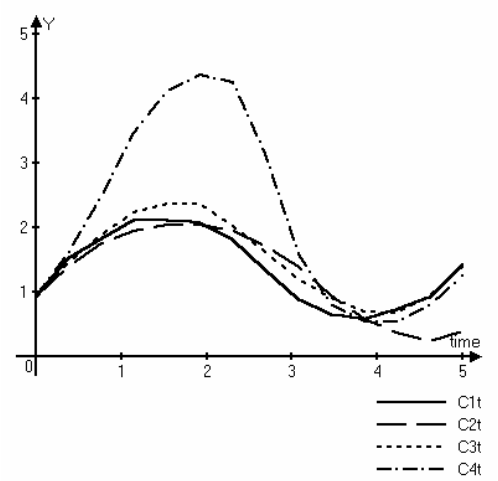


Figure 16: The sequences *C1R, C2R* and C*3R* shifted vertically.

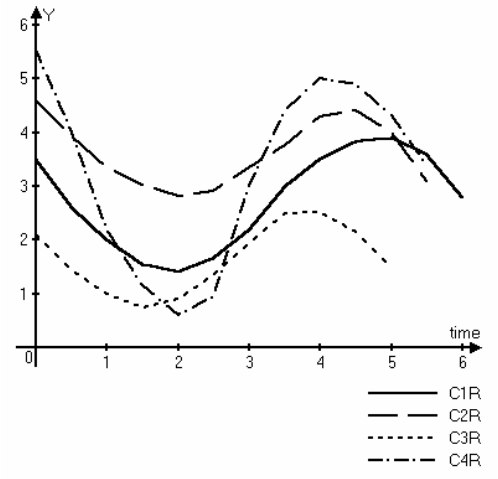


Figure 17: Time series dataset *CR*.

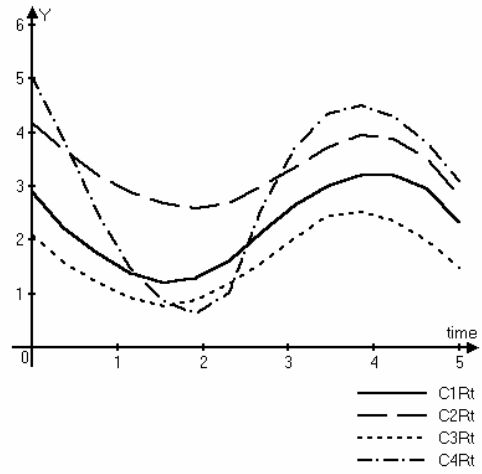


Figure 18: Scaled sequences designated by *C1Rt, C2Rt, C3Rt* and *C4Rt*.

TABLE 6
PARAMETER *WS (Q, C), S (Q, C)* VERSUS EUCLIDEAN DISTANCE *D (Q, C)*

Sequence Pairs	Parameter *WS*	Euclidean Distance *D*	Parameter *S*
C1Rt, C2Rt	0.816	4.04	1.231
C1Rt, C3Rt	0.372	2.46	0.977
C1Rt, C4Rt	1.544	3.99	4.812

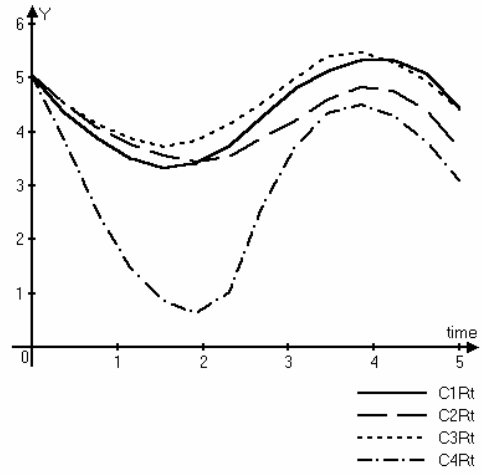


Figure 19: The sequences *C1Rt, C2Rt* and C*3Rt* shifted vertically.

5 Case Study

The case study undertaken in this paper consists of similarity analysis of retail sales data (in millions of dollars) collected on a monthly basis over a period of 11 years (from 01/1992 to 12/2002) for chain retail stores in USA [17]. The length of each time sequence in the retail sales time series database thus consists of 132 datapoints (for each item under sales). We considered sales data of several types of retail businesses as listed in Table 7.

The results of computing the parameter for cumulative variation in time weighted slopes denoted by *WS (Q, C)* given by (4) as compared to the Euclidean Distance given by (1) are shown in Table 8. The Sales at Health and Personal Care Stores has been taken as the query and all others have been taken as the candidate time sequences.

Some of the most similar sequences as evaluated using (4) are shown in Figure 20. It can be concluded from Table 8 that the sales at Health and Personal Care Stores recorded on a monthly basis for a period of 11 years is found to be most similar to the sales at Pharmacies and Drug stores recorded during the same period of time and is found to be the most dissimilar to the sales at New Car Dealers collected during the same period of time.

The results of computing the parameter for cumulative variation in time weighted slopes denoted by *S (Q, C)* given by (2) as compared to the Euclidean Distance given by (1) are shown in Table 9.

TABLE 7
BUSINESSES FOR WHICH RETAIL TIME SERIES DATA HAS BEEN CONSIDERED

S. No.	Description
1	Health and Personal Care Stores (Query)
2	Pharmacies and Drug stores
3	Furniture Stores
4	Jewellery stores
5	Sporting goods, Hobby and Music Stores
6	Household Appliances Stores
7	Men's Clothing Stores
8	Women's Clothing Stores
9	Shoe Stores
10	New Car Dealers
11	Used Car Dealers

TABLE 8
PARAMETER *WS (Q, C)* VERSUS *D (Q, C)*

S. No.	Description	Parameter *WS*	Euclidean Distance *D* (* 10^3)
1	Health and Personal Care Stores (Query)	0	0
2	Pharmacies and Drug stores	362.27	19.86
3	Used Car Dealers	1947.97	79.48
4	Women's Clothing Stores	2409.69	95.92
5	Shoe Stores	3086.38	105.35
6	Men's Clothing Stores	3090.50	115.27
7	Furniture Stores	3185.23	52.05
8	Household Appliances Stores	3226.95	114.76
9	Jewellery Stores	3264.25	104.55
10	Sporting goods, Hobby and Music Stores	5341.22	61.52
11	New Car Dealers	7938.09	390.38

The Sales at Health and Personal Care Stores has again been taken as the query and all others have been taken as the candidate time sequences. Some of the most similar sequences evaluated using (2) are shown in Figure 21.

It can be concluded from Table 9 that the sales at Health and Personal Care Stores recorded on a monthly basis for a period of 11 years is found to be most similar to the sales at Pharmacies and Drug stores and the least similar to the sales at New Car Dealers collected during the same period of time. From Table 9 it can be seen that the order of some of the candidate time sequences has changed.

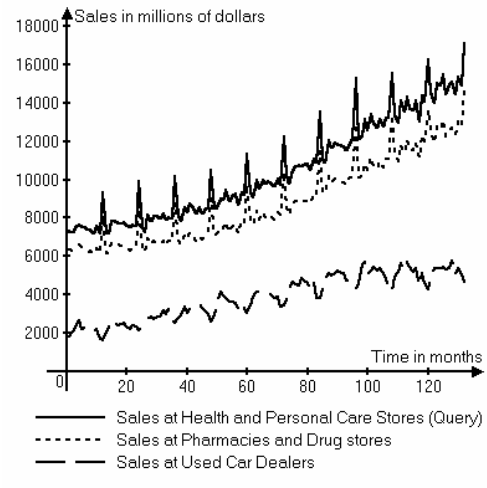


Figure 20: The most similar sequences as indicated by Table 8.

TABLE 9
PARAMETER *S (Q, C)* VERSUS *D (Q, C)*

S. No.	Description	Parameter *S* (* 10^2)	Euclidean Distance *D* (* 10^3)
1	Health and Personal Care Stores (Query)	0	0
2	Pharmacies and Drug stores	17.42	19.86
3	Women's Clothing Stores	57.53	95.92
4	Furniture Stores	71.27	52.05
5	Jewellery Stores	79.03	104.55
6	Shoe Stores	83.32	105.35
7	Men's Clothing Stores	88.49	115.27
8	Household Appliances Stores	106.22	114.76
9	Used Cars Dealers	126.40	79.48
10	Sporting goods, Hobby and Music Stores	144.22	61.52
11	New Car Dealers	426.39	390.38

Figure 21: The most similar sequences as indicated by Table 9.

6 Conclusions and Future Work

In this paper, a simple approach for performing similarity search in time series data has been proposed. The given time sequences are pre-processed and brought to the same time range. The y-values are also proportionately scaled to avoid any data distortions that may arise due to scaling along the time axis. The computation of the parameter for cumulative variations in time weighted slopes is done on the pre-processed data. It has been verified by the help of test data that the proposed technique can handle vertical shifts in the time sequence data, global scaling or shrinking of the data as well as variable length queries. No dimension reduction is required in this technique. Euclidean distance model has also been used to compare the test data considered. A case study on retail sales data from stores in USA has been undertaken.

In this approach we have assumed that a time series comprises of samples of a single measured variable against time. In future work, we intend to broaden its scope so that it can handle multivariable time sequences. We also intend to develop alternate parameters using the concept of slopes and time weights for assessing similarity in time series data which may be used individually or in conjunction with each other.

References

[1] A. Guttman (1984) "R-trees: A dynamic index structure for spatial searching," in *Proceedings of the ACM SIGMOD Conference,* pp. 47-57.

[2] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger (1990) "The R*-tree: An efficient and robust access method for points and rectangles," in *Proceedings of the ACM SIGMOD Conference,* pp. 322-331.

[3] K.V. Kanth, D. Agrawal, and A. Singh (1998) "Dimensionality reduction for similarity searching in dynamic databases," in *Proceedings of the ACM SIGMOD Conference.*, pp. 166-176.

[4] R. Agrawal, C. Faloutsos, and A. Swami (1993) "Efficient similarity search in sequence databases," in *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, pp. 69-84.

[5] K. Chu and M. Wong (1999) "Fast time-series searching with scaling and shifting," in *Proceedings of the 18th ACM Symposium on Principles of Database Systems,* pp. 237-248.

[6] C.Faloutsos, H. Jagadish, A. Mendelzon, and T. Milo (1997) "A signature technique for similarity based queries," in *Proceedings of the International Conference on Compression and Complexity of Sequences,* Positano-Salerno, Italy.

[7] D. Refiei (1999) "On similarity based queries for time series data," in *Proceedings of the 15th IEEE International Conference on Data Engineering*, pp. 410-417.

[8] K. Chan and A. W. Fu (1999) "Efficient time series matching by wavelets," in *Proceedings of the 15th IEEE International Conference on Data Engineering,* pp. 126-133.

[9] Y. Wu, D. Agrawal, and A. El Abbadi (2000) "A comparison of DFT and DWT based similarity search in time series databases," in *Proceedings of 9th ACM International Conference on Information and Knowledge Management,* pp. 488-495.

[10] T. Kahveei and A. Singh (2001) "Variable length queries for time series data," in *Proceedings of 17th International Conference on Data Engineering*, pp. 273-282.

[11] Z. Struzik and A. Siebes (1999) "The haar wavelet transform in the time series similarity paradigm," in *Proceedings of Conference on Principles of Data Mining and Knowledge Discovery*, pp. 12-22.

[12] C. Wang and X. S. Wang (2000) "Supporting content based searches on time series via approximation," in *Proceedings of International Conference on Scientific and Statistical Database Management,* pp. 69-81.

[13] F. Korn, H. Jagadish, and C. Faloutsos (1997) "Efficiently supporting ad hoc queries in large datasets of time sequences," in *Proceedings of ACM SIGMOD International Conference on Management of Data,* pp. 289-300.

[14] Byoung-Kee Yi and C. Faloutsos (2000) "Fast time sequence indexing for arbitrary Lp norms," *The VLDB Journal,* pp. 385-394.

[15] C.Faloutsos, M. Ranganathan, and Y. Mano Lopoulos (1999) "Fast subsequence matching in time-series databases," in *Proceedings of ACM SIGMOD International Conference on Management of Data,* pp. 419-429.

[16] Durga Toshniwal and R. C. Joshi (2004) " A new approach for similarity search in time series databases based on slopes" , *4th IEEE International Conference on Intelligent Systems, Design and Applications*, Budapest, Hungary, pp. 719- 724.

[17] Economic Time Series Page, http://www.economagic.com

Traffic Accident Analysis Using Machine Learning Paradigms

Miao Chong[1], Ajith Abraham[2] and Marcin Paprzycki[1, 3]
[1]Computer Science Department, Oklahoma State University, USA, marcin@cs.okstate.edu
[2]School of Computer Science and Engineering, Chung-Ang University, Korea, ajith.abraham@ieee.org
[3]Computer Science, SWPS, Warszawa, Poland

Keywords: traffic accident, data mining, machine learning, hybrid system, decision trees, support vector machine

Received: December 20, 2004

Engineers and researchers in the automobile industry have tried to design and build safer automobiles, but traffic accidents are unavoidable. Patterns involved in dangerous crashes could be detected if we develop accurate prediction models capable of automatic classification of type of injury severity of various traffic accidents. These behavioral and roadway accident patterns can be useful to develop traffic safety control policies. We believe that to obtain the greatest possible accident reduction effects with limited budgetary resources, it is important that measures be based on scientific and objective surveys of the causes of accidents and severity of injuries. This paper summarizes the performance of four machine learning paradigms applied to modeling the severity of injury that occurred during traffic accidents. We considered neural networks trained using hybrid learning approaches, support vector machines, decision trees and a concurrent hybrid model involving decision trees and neural networks. Experiment results reveal that among the machine learning paradigms considered the hybrid decision tree-neural network approach outperformed the individual approaches.

Povzetek: Štirje pristopi strojnega učenja so uporabljeni za preiskovanje zakonitosti poškodb v prometnih nesrečah.

1 Introduction

The costs of fatalities and injuries due to traffic accidents have a great impact on the society. In recent years, researchers have paid increasing attention to determining factors that significantly affect severity of driver injuries caused by traffic accidents [29][30]. There are several approaches that researchers have employed to study this problem. These include neural network, nesting logic formulation, log-linear model, fuzzy ART maps and so on.

Applying data mining techniques to model traffic accident data records can help to understand the characteristics of drivers' behaviour, roadway condition and weather condition that were causally connected with different injury severity. This can help decision makers to formulate better traffic safety control policies. Roh et al. [22] illustrated how statistical methods based on directed graphs, constructed over data for the recent period, may be useful in modelling traffic fatalities by comparing models specified using directed graphs to a model, based on out-of-sample forecasts, originally developed by Peltzman [23]. The directed graphs model outperformed Peltzman's model in root mean squared forecast error.

Ossenbruggen et al. [24] used a logistic regression model to identify statistically significant factors that predict the probabilities of crashes and injury crashes aiming at using these models to perform a risk assessment of a given region. These models were functions of factors that describe a site by its land use activity, roadside design, use of traffic control devices and traffic exposure. Their study illustrated that village sites are less hazardous than residential and shopping sites. Abdalla et al. [25] studied the relationship between casualty frequencies and the distance of the accidents from the zones of residence. As might have been anticipated, the casualty frequencies were higher nearer to the zones of residence, possibly due to higher exposure. The study revealed that the casualty rates amongst residents from areas classified as relatively deprived were significantly higher than those from relatively affluent areas.

Miaou et al. [26] studied the statistical properties of four regression models: two conventional linear regression models and two Poisson regression models in terms of their ability to model vehicle accidents and highway geometric design relationships. Roadway and truck accident data from the Highway Safety Information System (HSIS) have been employed to illustrate the use and the limitations of these models. It was demonstrated that the conventional linear regression models lack the distributional property to describe adequately random, discrete, nonnegative, and typically sporadic vehicle accident events on the road. The Poisson regression models, on the other hand, possess most of the desirable statistical properties in developing the relationships.

Abdelwahab et al. studied the 1997 accident data for the Central Florida area [2]. The analysis focused on vehicle accidents that occurred at signalized intersections. The injury severity was divided into three

classes: no injury, possible injury and disabling injury. They compared the performance of Multi-layered Perceptron (MLP) and Fuzzy ARTMAP, and found that the MLP classification accuracy is higher than the Fuzzy ARTMAP. Levenberg-Marquardt algorithm was used for the MLP training and achieved 65.6 and 60.4 percent classification accuracy for the training and testing phases, respectively. The Fuzzy ARTMAP achieved a classification accuracy of 56.1 percent.

Yang et al. used neural network approach to detect safer driving patterns that have less chances of causing death and injury when a car crash occurs [17]. They performed the Cramer's V Coefficient test [18] to identify significant variables that cause injury to reduce the dimensions of the data. Then, they applied data transformation method with a frequency-based scheme to transform categorical codes into numerical values. They used the Critical Analysis Reporting Environment (CARE) system, which was developed at the University of Alabama, using a Backpropagation (BP) neural network. They used the 1997 Alabama interstate alcohol-related data, and further studied the weights on the trained network to obtain a set of controllable cause variables that are likely causing the injury during a crash. The target variable in their study had two classes: injury and non-injury, in which injury class included fatalities. They found that by controlling a single variable (such as the driving speed, or the light conditions) they potentially could reduce fatalities and injuries by up to 40%.

Sohn et al. applied data fusion, ensemble and clustering to improve the accuracy of individual classifiers for two categories of severity (bodily injury and property damage) of road traffic accidents [15]. The individual classifiers used were neural network and decision tree. They applied a clustering algorithm to the dataset to divide it into subsets, and then used each subset of data to train the classifiers. They found that classification based on clustering works better if the variation in observations is relatively large as in Korean road traffic accident data.

Mussone et al. used neural networks to analyze vehicle accident that occurred at intersections in Milan, Italy [12]. They chose feed-forward MLP using BP learning. The model had 10 input nodes for eight variables (day or night, traffic flows circulating in the intersection, number of virtual conflict points, number of real conflict points, type of intersection, accident type, road surface condition, and weather conditions). The output node was called an accident index and was calculated as the ratio between the number of accidents for a given intersection and the number of accidents at the most dangerous intersection. Results showed that the highest accident index for running over of pedestrian occurs at non-signalized intersections at nighttime.

Dia et al. used real-world data for developing a multi-layered MLP neural network freeway incident detection model [5]. They compared the performance of the neural network model and the incident detection model in operation on Melbourne's freeways. Results showed that neural network model could provide faster and more reliable incident detection over the model that was in operation. They also found that failure to provide speed data at a station could significantly deteriorate model performance within that section of the freeway.

Shankar et al. applied a nested logic formulation for estimating accident severity likelihood conditioned on the occurrence of an accident [14]. They found that there is a greater probability of evident injury or disabling injury/fatality relative to no evident injury if at least one driver did not use a restraint system at the time of the accident.

Kim et al. developed a log-linear model to clarify the role of driver characteristics and behaviors in the causal sequence leading to more severe injuries. They found that alcohol or drug use and lack of seat belt use greatly increase the odds of more severe crashes and injuries [8].

Abdel-Aty et al. used the Fatality Analysis Reporting System (FARS) crash databases covering the period of 1975-2000 to analyze the effect of the increasing number of Light Truck Vehicle (LTV) registrations on fatal angle collision trends in the US [1]. They investigated the number of annual fatalities that resulted from angle collisions as well as collision configuration (car-car, car-LTV, LTV-car, and LTV-LTV). Time series modeling results showed that fatalities in angle collisions will increase in the next 10 years, and that they are affected by the expected overall increase of the percentage of LTVs in traffic.

Bedard et al. applied a multivariate logistic regression to determine the independent contribution of driver, crash, and vehicle characteristics to drivers' fatality risk [3]. They found that increasing seatbelt use, reducing speed, and reducing the number and severity of driver-side impacts might prevent fatalities. Evanco conducted a multivariate population-based statistical analysis to determine the relationship between fatalities and accident notification times [6]. The analysis demonstrated that accident notification time is an important determinant of the number of fatalities for accidents on rural roadways.

Ossiander et al. used Poisson regression to analyze the association between the fatal crash rate (fatal crashes per vehicle mile traveled) and the speed limit increase [13]. They found that the speed limit increase was associated with a higher fatal crash rate and more deaths on freeways in Washington State.

Finally, researchers studied the relationship between drivers' age, gender, vehicle mass, impact speed or driving speed measure with fatalities and the results of their work can be found in [4, 9, 10, 11, 16].

This paper investigates application of neural networks, decision trees and a hybrid combination of decision tree and neural network to build models that could predict injury severity. The remaining parts of the paper are organized as follows. In Section 2, more details about the problem and the pre-processing of data to be used are presented, followed, in Section 3, by a short description the different machine learning paradigms used. Performance analysis is presented in Section 4 and finally some discussions and conclusions are given towards the end.

2 Accident Data Set

A. Description of the Dataset

This study used data from the National Automotive Sampling System (NASS) General Estimates System (GES) [21]. The GES datasets are intended to be a nationally representative probability samples from the annual estimated 6.4 million accident reports in the United States. The initial dataset for the study contained traffic accident records from 1995 to 2000, a total number of 417,670 cases. According to the variable definitions for the GES dataset, this dataset has drivers' records only and does not include passengers' information. The total set includes labels of year, month, region, primary sampling unit, the number describing the police jurisdiction, case number, person number, vehicle number, vehicle make and model; inputs of drivers' age, gender, alcohol usage, restraint system, eject, vehicle body type, vehicle age, vehicle role, initial point of impact, manner of collision, rollover, roadway surface condition, light condition, travel speed, speed limit and the output injury severity. The injury severity has five classes: *no injury*, *possible injury*, *non-incapacitating injury*, *incapacitating injury*, and *fatal injury*. In the original dataset, 70.18% of the cases have output of no injury, 16.07% of the cases have output of possible injury, 9.48% of the cases have output of non-incapacitating injury, 4.02% of the cases have output of incapacitating injury, and 0.25% of the cases have fatal injury.

Our task was to develop machine learning based intelligent models that could accurately classify the severity of injuries (5 categories). This can in turn lead to greater understanding of the relationship between the factors of driver, vehicle, roadway, and environment and driver injury severity. Accurate results of such data analysis could provide crucial information for the road accident prevention policy. The records in the dataset are input/output pairs with each record have an associated output. The output variable, the injury severity, is categorical and (as described above) has five classes. A supervised learning algorithm will try to map an input vector to the desired output class.

B. Data Preparation

When the input and output variables are considered there are no conflicts between the attributes since each variable represents its own characteristics. Variables are already categorized and represented by numbers. The manner in which the collision occurred has 7 categories: non-collision, rear-end, head-on, rear-to-rear, angle, sideswipe same direction, and sideswipe opposite direction. For these 7 categories the distribution of the fatal injury is as follows: 0.56% for non collision, 0.08% for rear-end collision, 1.54% for head-on collision, 0.00% for rear-to-rear collision, 0.20% for angle collision, 0.08% for sideswipe same direction collision, 0.49% for sideswipe opposite direction collision. Since *head-on* collision has the highest percent of fatal injury; therefore, the dataset was narrowed down to head-on collision only. Head-on collision has a total of 10,386 records, where 160 records show the result as a *fatal injury*; all of these 160 records have the initial point of impact categorized as *front*.

The initial point of impact has 9 categories: no damage/non-collision, front, right side, left side, back, front right corner, front left corner, back right corner, back left corner. The head-on collision with *front impact* has 10,251 records; this is 98.70% of the 10,386 head-on collision records. We have therefore decided to focus on front impact only and removed the remaining 135 records. Travel speed and speed limit were not used in the model because in the dataset there are too many records with unknown value. Specifically, for 67.68% of records the travel speed during accident and local speed limit were unknown. This means that the remaining input variables were: drivers' age, gender, alcohol usage, restraint system, eject, vehicle body type, vehicle role, vehicle age, rollover, road surface condition, light condition. Table 1 summarizes the driver injury severity distribution for head-on collision and front impact point dataset. From Table 1, it is immediately evident that the alcohol usage and not using seat belt, ejection of driver, driver's age (>65), vehicle rollover, and lighting condition can be associated with higher percentages of fatal injury, incapacitating injury and non-incapacitating injury.

There are only single vehicles with ages 37, 41, 46 and 56 years reported in the dataset and therefore these four records were deleted from the dataset (since they were clear outliers). After the preprocessing was completed, the final dataset used for modeling had 10,247 records. There were 5,171 (50.46%) records with no injury, 2138 (20.86%) records with possible injury, 1721 (16.80%) records with non-incapacitating injury, 1057 (10.32%) records with incapacitating injury, and 160 (1.56%) records with fatal injury. We have separated each output class and used one-against-all approach. This approach selects one output class to be the positive class, and all the other classes are combined to be the negative class. We set the output value of the positive class to 1, and the (combined) negative classes to 0. We divided the datasets randomly into 60%, 20%, and 20% for training, cross-validation, and testing respectively.

To make sure that our data preparation is valid, we have checked the correctness of attribute selection. There are several attribute selection techniques to find a minimum set of attributes so that the resulting probability distribution of the data classes is as close as possible to the original distribution of all attributes. To determine the best and worst attributes, we used the chi-squared (χ^2) test to determine the dependence of input and output variables. The χ^2 test indicated that all the variables are significant (p-value < 0.05).

3. Machine Learning Paradigms

A. Artificial Neural Networks Using Hybrid Learning

A Multilayer Perceptron (MLP) is a feed forward neural network with one or more hidden layers.

Table 1: Driver injury severity distribution

Factor	No Injury	Pos injury	Non-incapacitating	Incapacitating	Fatal	Total
Age						
0 (24&under)	1629(52.80%)	608(19.71%)	505(16.37%)	307(9.95%)	36(1.17%)	3085
1 (25-64)	3171(49.88%)	1362(21.43%)	1075(16.91%)	654(10.29%)	95(1.49%)	6357
2 (65+)	373(46.11%)	168(20.77%)	143(17.68%)	96(11.87%)	29(3.58%)	809
Gender						
0 (Female)	1749(41.95%)	1072(25.71%)	778(18.66%)	507(12.16%)	63(1.51%)	4169
1 (Male)	3424(56.30%)	1066(17.53%)	945(15.54%)	550(9.04%)	97(1.59%)	6082
Eject						
0 (No Eject)	5171(50.55%)	2137(20.89%)	1719(16.80%)	1047(10.23%)	156(1.52%)	10230
1 (Eject)	2(9.52%)	1(4.76%)	4(19.05%)	10(47.62%)	4(19.05%)	21
Alcohol						
0 (No Alcohol)	4997(51.35%)	2067(21.24%)	1600(16.44%)	935(9.61%)	133(1.37%)	9732
1 (Alcohol)	176(33.91%)	71(13.68%)	123(23.70%)	122(23.51%)	27(5.20%)	519
Restraining System						
0 (Not Used)	337(27.44%)	193(15.72%)	336(27.36%)	283(23.05%)	79(6.43%)	1228
1 (Used)	4836(53.60%)	1945(21.56%)	1387(15.37%)	774(8.58%)	81(0.90%)	9023
Body Type						
0 (cars)	3408(47.49%)	1600(22.30%)	1272(17.73%)	780(10.87%)	116(1.62%)	7176
1 (SUV &Van)	747(56.59%)	259(19.62%)	189(14.32%)	111(8.41%)	14(1.06%)	1320
2 (Truck)	1018(58.01%)	279(15.90%)	262(14.93%)	166(9.46%)	30(1.71%)	1755
Vehicle Role						
1 (Striking)	4742(49.86%)	2011(21.15%)	1636(17.20%)	970(10.20%)	151(1.59%)	9510
2 (Struck)	261(72.70%)	54(15.04%)	29(8.08%)	15(4.18%)	0(0%)	359
3 (Both)	170(44.50%)	73(19.11%)	58(15.18%)	72(18.85%)	9(2.36%)	382
Rollover						
0 (No-rollover)	5069(50.78%)	2123(20.85%)	1699(16.69%)	1037(10.19%)	152(1.49%)	10180
1 (Rollover)	4(5.63%)	15(21.13%)	24(33.80%)	20(28.17%)	8(11.27%)	71
Road Surface Condition						
0 (Dry)	3467(49.97%)	1404(20.24%)	1190(17.15%)	750(10.81%)	127(1.83%)	6938
1 (Slippery)	1706(51.49%)	734(22.16%)	533 (16.09%)	307(9.27%)	33(1.00%)	3313
Light Condition						
0 (Daylight)	3613(51.18%)	1487(21.06%)	1174(16.63%)	688(9.75%)	98(1.39%)	7060
1(Partial dark)	1139(52.71%)	465(21.52%)	348(16.10%)	186(8.61%)	23(1.06%)	2161
2 (Dark)	421(40.87%)	186(18.06%)	201(19.51%)	183(17.77%)	39(3.79%)	1030

The network consists of an input layer of source neurons, at least one hidden layer of computational neurons, and an output layer of computational neurons. The input layer accepts input signals and redistributes these signals to all neurons in the hidden layer. The output layer accepts a stimulus pattern from the hidden layer and establishes the output pattern of the entire network. The MLP neural networks training phase works as follows: given a collection of training data $\{x_1(p), d_1(p)\}$, …, $\{x_i(p), d_i(p)\}$, …, $\{x_n(p), d_n(p)\}$, the objective is to obtain a set of weights that makes almost all the tuples in the training data classified correctly, or in other words, is to map $\{x_1(p)$ to $d_1(p)\}$, …, $\{x_i(p)$ to $d_i(p)\}$, and eventually $\{x_n(p)$ to $d_n(p)\}$. The algorithm starts with initializing all the weights (w) and threshold (θ) levels of the network to small random numbers. Then calculate the actual output of the neurons in the hidden layer as:

$$y_i(p) = f[\sum_{(i=1 \text{ to } n)} x_i(p) * w_{ij}(p) - \theta_j],$$

where n is the number of inputs of neuron j in the hidden layer. Next calculate the actual outputs of the neurons in the output layer as:

$$y_k(p) = f[\sum_{(j=1 \text{ to } m)} x_{jk}(p) * w_{jk}(p) - \theta_k],$$

where m is the number of inputs of neuron k in the output layer. The weight training is to update the weights using the Backpropagation (BP) learning method with the error function:

E (w) = $\sum_{(p=1 \text{ to } PT)} \sum_{(i=1 \text{ to } l)} [d_i(p) - y_i(p)]^2$,

where

E (w) = error function to be minimized,

w = weight vector,

PT = number of training patterns,

l = number of output neurons,

$d_i(p)$ = desired output of neuron I when pattern p is introduced to the MLP, and

$y_i(p)$ = actual output of the neuron I when pattern p is introduced to the MLP. The objective of weight training is to change the weight vector w so that the error function is minimized. By minimizing the error function, the actual output is driven closer to the desired output.

Empirical research [19] has shown that the BP used for training neural networks has the following problems:

- BP often gets trapped in a local minimum mainly because of the random initialization of weights.
- BP usually generalizes quite well to detect the global features of the input but after prolonged training the network will start to recognize individual input/output pair rather than settling for weights that generally describe the mapping for the whole training set.

The second popular training algorithm for neural networks is Scaled Conjugate Gradient Algorithm (SCGA). Moller [20] introduced it as a way of avoiding the complicated line search procedure of conventional conjugate gradient algorithm (CGA). According to the SCGA, the Hessian matrix is approximated by

$$E''(w_k)p_k = \frac{E'(w_k + \sigma_k p_k) - E'(w_k)}{\sigma_k} + \lambda_k p_k$$

where E' and E'' are the first and second derivative information of global error function $E(w_k)$. The other terms p_k, σ_k and λ_k represent the weights, search direction, parameter controlling the change in weight for the second derivative approximation and parameter for regulating the indefiniteness of the Hessian. In order to obtain a good, quadratic, approximation of E, a mechanism to raise and lower λ_k is needed when the Hessian is positive definite. Detailed step-by-step description can be found in [20].

In order to minimize the above-mentioned problems resulting from the BP training, we used a combination of BP and SCG for training.

B. Decision Trees

Decision trees are well-known algorithm for classification problems. The Classification and Regression Trees (CART) model consists of a hierarchy of univariate binary decisions. Each internal node in the tree specifies a binary test on a single variable, branch represents an outcome of the test, each leaf node represent class labels or class distribution. CART operates by choosing the best variable for splitting the data into two groups at the root node, partitioning the data into two disjoint branches in such a way that the class labels in each branch are as homogeneous as possible, and then splitting is recursively applied to each branch, and so forth.

If a dataset T contains examples from n classes, gini index, $gini(T)$ is defined as: $gini\ (T) = 1 - \sum_{j=1 \text{ to } n} p_j\text{^}2$, where p_j is the relative frequency of class j in T [31]. If dataset T is split into two subsets T_1 and T_2 with sizes N_1 and N_2, the gini index of the split data contains examples from n classes, the gini index $gini(T)$ is defined as:

$$gini_{\text{split}}(T) = N_1/N\ gini(T_1) + N_2/N\ gini(T_2).$$

CART exhaustively searches for univariate splits. The attribute provides the smallest $gini_{\text{split}}(T)$ is chosen to split the node. CART recursively expands the tree from a root node, and then gradually prunes back the large tree. The advantage of a decision tree is the extraction of classification rules from trees that is very straightforward. More precisely, a decision tree can represent the knowledge in the form of if-then rules; one rule is created for each path from the root to a leaf node.

C. Support Vector Machines

Support Vector Machine (SVM) is based on statistical learning theory [28] . SVMs have been successfully applied to a number of applications ranging from handwriting recognition, intrusion detection in computer networks, and text categorization to image classification, breast cancer diagnosis and prognosis and bioinformatics. SVM involves two key techniques, one is the mathematical programming and the other is kernel functions. Here, parameters are found by solving a quadratic programming problem with linear equality and inequality constraints; rather than by solving a non-convex, unconstrained optimization problem. SVMs are kernel-based learning algorithms in which only a fraction of the training examples are used in the solution (these are called the support vectors), and where the objective of learning is to maximize a margin around the decision surface. The flexibility of kernel functions allows the SVM to search a wide variety of hypothesis spaces. The basic idea of applying SVMs to pattern classification can be stated briefly as: first map the input vectors into one feature space (possible with a higher dimension), either linearly or nonlinearly, whichever is relevant to the selection of the kernel function; then within the feature space, seek an optimized linear division, i.e. construct a hyperplane which separates two classes.

For a set of n training examples (x_i, y_i), where $x_i \in \boldsymbol{R}^d$ and $y_i \in \{-1, +1\}$, suppose there is a hyperplane, which separates the positive from the negative examples. The points x which lie on the hyperplane (H_0) satisfy $w \cdot x + b = 0$, the algorithm finds this hyperplane (H_0) and other two hyperplanes (H_1, H_2) parallel and equidistant to H_0,

H_1: $w \cdot x_i + b = 1$, H_2: $w \cdot x_i + b = -1$,

H_1 and H_2 are parallel and no training points fall between them. Support vector algorithm looks for the separating hyperplane and maximizes the distance between H_1 and H_2. So there will be some positive examples on H_1 and some negative examples on H_2. These examples are called support vectors. The distance between H_1 and H_2 is $2/\|w\|$, in order to maximize the distance, we should minimize $\|w\| = w^T w$, subject to constraints $y_i (w \cdot x_i + b) >= 1, \forall_i$

Introducing Lagrangian multipliers $\alpha_1, \alpha_2, \ldots, \alpha n>=0$, the learning task becomes

$L (w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1 \text{ to } n} \alpha_l [y_i (w \cdot x_i + b) - 1]$

The above equation is for two classes that are linearly separable. When the two classes are non-linearly separable, SVM can transform the data points to another high dimensional space. Detailed description to the theory of SVMs for pattern recognition can be found in [32].

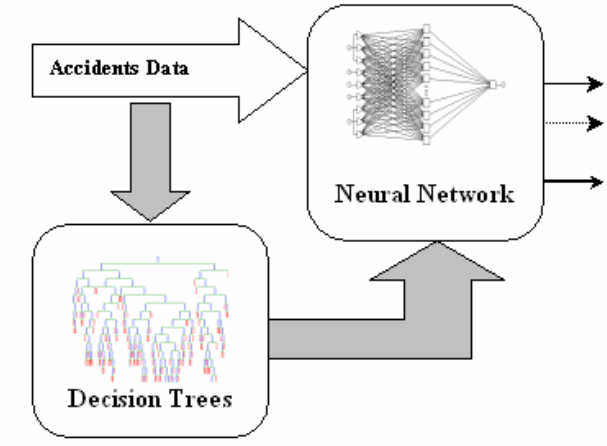


Fig. 1. Hybrid concurrent decision tree-ANN model for accident data

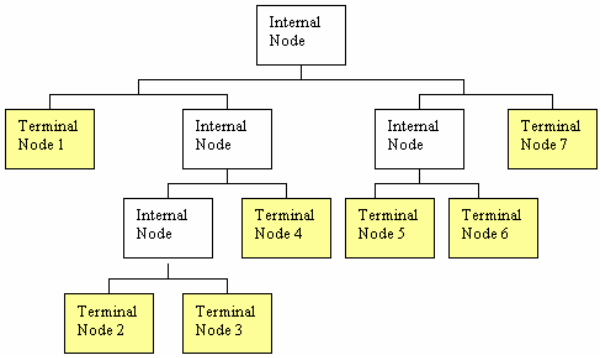


Fig. 2. Decision tree structure

D. Hybrid Decision Tree-ANN (DTANN)

A hybrid intelligent system uses the approach of integrating different learning or decision-making models. Each learning model works in a different manner and exploits different set of features. Integrating different learning models gives better performance than the individual learning or decision-making models by reducing their individual limitations and exploiting their different mechanisms. In a hierarchical hybrid intelligent system each layer provides some new information to the higher level [33]. The overall functioning of the system depends on the correct functionality of all the layers. Figure 1 illustrates the hybrid decision tree-ANN (DTANN) model for predicting drivers' injury severity. We used a concurrent hybrid model where traffic accidents data are fed to the decision tree to generate the node information. Terminal nodes were numbered left to right starting with 1. All the data set records were assigned to one of the terminal nodes, which represented the particular class or subset. The training data together with the node information were supplied for training the ANN. Figure 2 illustrates a decision tree structure with the node numbering. For the hybrid decision tree–ANN, we used the same hybrid learning algorithms and parameters setting as we used for ANN (except for the number of hidden neurons). Experiments were performed with different number of hidden neurons and models were selected with the highest classification accuracy for the output class.

4. Performance Analysis

A. Neural Networks

In the case of neural network based modeling, the hyperbolic activation function was used in the hidden layer and the logistic activation function in the output layer. Models were trained with BP (100 epochs, learning rate 0.01) and SCGA (500 epochs) to minimize the Mean Squared Error (MSE). For each output class, we experimented with different number of hidden neurons, and report the model with highest classification accuracy for the class. From the experiment results, for the no injury class the best model had 65 hidden neurons, and achieved training and testing performance of 63.86% and 60.45% respectively. For the possible injury class, the best model had 65 hidden neurons achieving it's training and testing performance of 59.34% and 57.58% respectively. For the non-incapacitating injury class, the best model had 75 hidden neurons achieving training and testing performance of 58.71% and 56.8% respectively. For the incapacitating injury class, the best model had 60 hidden neurons achieving training and testing performance of 63.40% and 63.36% respectively. Finally, for the fatal injury class, the best model had 45 hidden neurons achieving training and testing performance of 78.61% and 78.17% respectively. These results are the summary of multiple experiments (for variable no of hidden neurons and for a number of attempts with random initial weight distributions resulting in almost exact performance of the trained network) and are presented in Table 2.

B. Decision Trees

We have experimented with a number of setups of decision tree parameters and report the best results obtained for our dataset. We trained each class with Gini goodness of fit measure, the prior class probabilities parameter was set to equal, the stopping option for pruning was misclassification error, the minimum *n* per node was set to 5, the fraction of objects was 0.05, the

maximum number of nodes was 1000, the maximum number of levels in the tree was 32, the number of surrogates was 5, we used 10 fold cross-validation, and generated comprehensive results. The cross-validation testing ensured that the patterns found will hold up when applied to new data.

Table 2. Neural network performance

Table 2. Neural network performance														
No Injury			**Possible Injury**			**Non-incapacitating**			**Incapacitating**			**Fatal Injury**		
# neurons	Accuracy %		# neurons	Accuracy %		# neurons	Accuracy %		# neurons	Accuracy %		# neurons	Accuracy %	
	Train	Test		Train	Test		Train	Test		Train	Test		Train	Test
60	63.57	59.67	65	59.34	**57.58**	60	57.88	55.25	60	63.4	**63.36**	45	77.26	**75.17**
65	63.86	**60.45**	70	59.56	55.15	65	57.69	54.66	65	62.23	61.32	57	74.78	70.65
70	63.93	60.25	75	58.88	57.29	75	58.71	**56.80**	75	61.06	61.52	65	69.81	69.73
75	64.38	57.43	80	58.39	56.22	80	57.78	54.13	84	63.23	58.41	75	60.19	59.62
80	63.64	58.89	95	60.07	55.93	85	57.83	55.59	90	59.32	59.08	80	74.33	71.77

Table 3: Performance of SVM using radial basis function kernel

	g=0.0001 c=42.8758	g=0.001 c=4.6594	g=0.5 c=0.5	g=1.2 c=0.5	g=1.5 c=2	g=2 c=10	g=0.00001 c=100	g=0.0001 c=100	g=0.001 c=100
No injury									
Class 0	59.76	59.80	57.95	57.65	53.62	54.12	57.34	59.76	60.46
Class 1	60.14	60.14	60.82	55.63	55.73	55.53	62.88	60.14	60.14
Possible injury									
Class 0	100.00	100.00	100.00	99.88	95.33	95.58	100.00	100.00	100.00
Class 1	0.00	0.00	0.00	0.00	3.67	3.42	0.00	0.00	0.00
Non-incapacitating									
Class 0	100.00	100.00	100.00	100.00	97.43	97.49	100.00	100.00	100.00
Class 1	0.00	0.00	0.00	0.00	3.21	2.92	0.00	0.00	0.00
Incapacitating									
Class 0	100.00	100.00	100.00	99.89	98.06	98.11	100.00	100.00	100.00
Class 1	0.00	0.00	0.00	0.00	2.83	2.83	0.00	0.00	0.00
Fatal Injury									
Class 0	100.00	100.00	100.00	100.00	99.95	99.95	100.00	100.00	100.00
Class 1	0.00	0.00	0.00	0.00	3.33	3.33	0.00	0.00	0.00

Table 4. Decision tree performance

Injury Class	Accuracy (%)
No Injury	67.54
Possible Injury	64.40
Non-incapacitating Injury	60.37
Incapacitating Injury	71.38
Fatal Injury	89.46

The performance for no injury, possible injury, non-incapacitating injury, incapacitating injury and fatal injury models was 67.54%, 64.39%, 60.37%, 71.38%, and 89.46% respectively. Empirical results including classification matrix are illustrated in Table 4. The developed decision trees are depicted in Figures 3-7. Each of these trees has a completely different structure and number of nodes and leaves. Note, that information stored in leaves of exactly these decision trees has been used in developing the hybrid decision tree – neural network model.

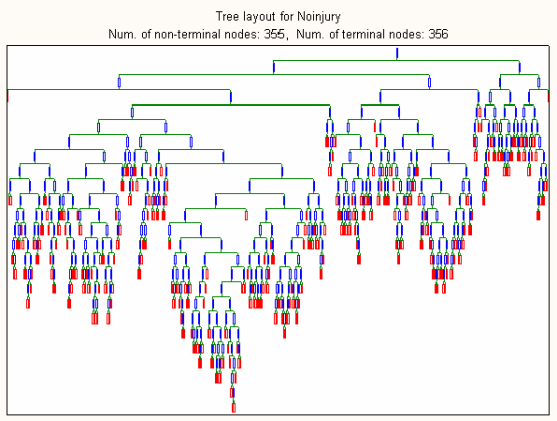


Fig. 3: No injury tree structure

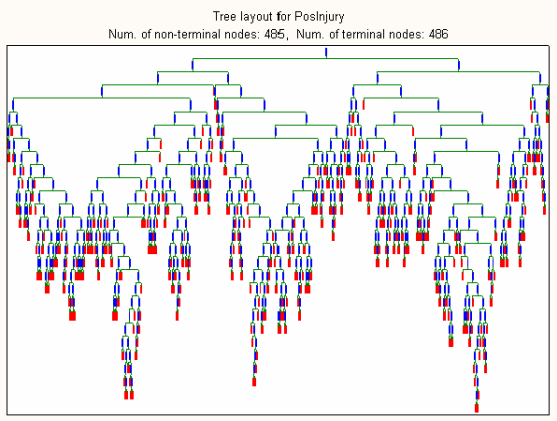


Fig. 4: Possible injury tree structure

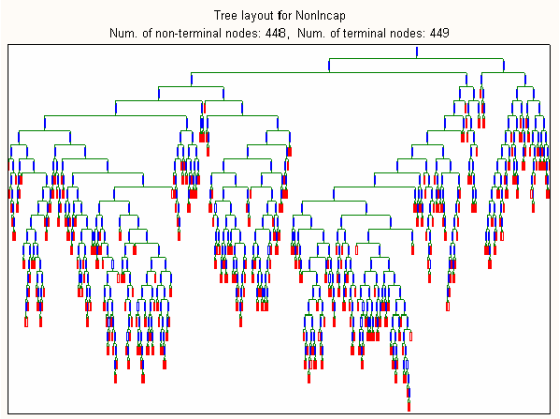


Fig. 5: Non-incapacitating injury tree structure

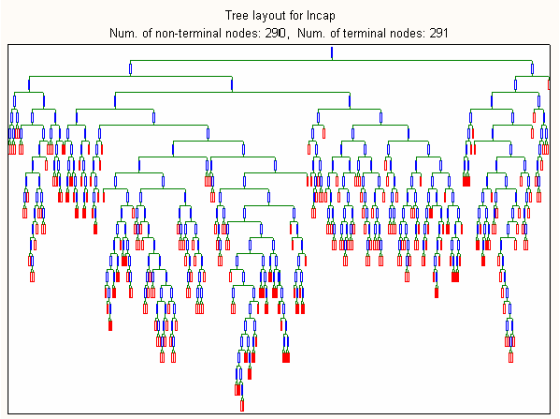


Fig. 6: Incapacitating injury tree structure

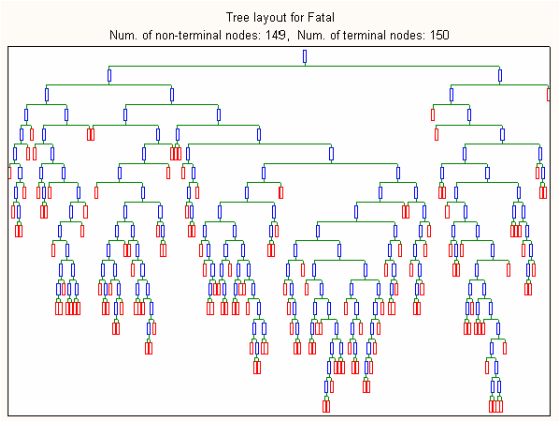


Fig. 7: Fatal injury tree structure

C. Support Vector Machines

In our experiments we used the SVM^{light} [27] and selected the polynomial and radial basis function kernels. For an unknown reason, the polynomial kernel was not successful and hence we only focused on the radial basis function (RBF) kernels. Table 3 illustrates the SVM performance for the different parameter settings and the obtained accuracies for each class.

D. Hybrid DT-ANN Approach

In the case of the hybrid approach, for the no injury class the best model had 70 hidden neurons, with training and testing performance of 83.02% and 65.12% respectively.

For the possible injury class, the best model had 98 hidden neurons with training and testing performance of 74.93% and 63.10% respectively. For the non-incapacitating injury class, the best model had 109 hidden neurons with training and testing performance of 71.88% and 62.24% respectively. For the incapacitating injury class, the best model had 102 hidden neurons, with training and testing performance of 77.95% and 72.63% respectively. Finally, for the fatal injury class, the best model had 76 hidden neurons with training and testing performance of 91.53% and 90.00% respectively. These are the best models out of multiple experiments varying various parameters of the ANN and the decision tree. Empirical results are presented in Table 5 and the final comparison between ANN, DT and DTANN is graphically illustrated in Figure 8. For all the output classes, the hybrid DTANN outperformed the ANN. For non-incapacitating injury, incapacitating injury, and fatal injury classes, the hybrid DTANN outperformed both ANN and DT.

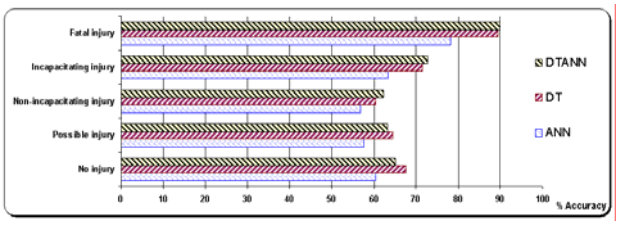


Fig. 8. Performance comparison of the different learning paradigms

Table 5. Test performance of DTANN

Injury type	% Accuracy
No injury	65.12
Possible injury	63.10
Non-incapacitating injury	62.24
Incapacitating injury	72.63
Fatal injury	90.00

5. Concluding Remarks

In this paper, we analyzed the GES automobile accident data from 1995 to 2000 and investigated the performance of neural network, decision tree, support vector machines and a hybrid decision tree – neural network based approaches to predicting drivers' injury severity in head-on front impact point collisions. The classification accuracy obtained in our experiments reveals that, for the non-incapacitating injury, the incapacitating injury, and the fatal injury classes, the hybrid approach performed better than neural network, decision trees and support vector machines. For the no injury and the possible injury classes, the hybrid approach performed better than neural network. The no injury and the possible injury classes could be best modeled directly by decision trees.

Past research focused mainly on distinguishing between no-injury and injury (including fatality) classes. We extended the research to possible injury, non-incapacitating injury, incapacitating injury, and fatal injury classes. Our experiments showed that the model for fatal and non-fatal injury performed better than other classes. The ability of predicting fatal and non-fatal injury is very important since drivers' fatality has the highest cost to society economically and socially.

It is well known that one of the very important factors causing different injury level is the actual speed that the vehicle was going when the accident happened. Unfortunately, our dataset doesn't provide enough information on the actual speed since speed for 67.68% of the data records' was unknown. If the speed was available, it is extremely likely that it could have helped to improve the performance of models studied in this paper.

6. References

[1] Abdel-Aty, M., and Abdelwahab, H., Analysis and Prediction of Traffic Fatalities Resulting From Angle Collisions Including the Effect of Vehicles' Configuration and Compatibility. *Accident Analysis and Prevention*, 2003.

[2] Abdelwahab, H. T. and Abdel-Aty, M. A., Development of Artificial Neural Network Models to Predict Driver Injury Severity in Traffic Accidents at Signalized Intersections. *Transportation Research Record 1746*, Paper No. 01-2234.

[3] Bedard, M., Guyatt, G. H., Stones, M. J., & Hireds, J. P., The Independent Contribution of Driver, Crash, and Vehicle Characteristics to Driver Fatalities. *Accident analysis and Prevention*, Vol. 34, pp. 717-727, 2002.

[4] Buzeman, D. G., Viano, D. C., & Lovsund, P., Car Occupant Safety in Frontal Crashes: A Parameter Study of Vehicle Mass, Impact Speed, and Inherent Vehicle Protection. *Accident Analysis and Prevention*, Vol. 30, No. 6, pp. 713-722, 1998.

[5] Dia, H., & Rose, G., Development and Evaluation of Neural Network Freeway Incident Detection Models Using Field Data. *Transportation Research C*, Vol. 5, No. 5, 1997, pp. 313-331.

[6] Evanco, W. M., The Potential Impact of Rural Mayday Systems on Vehicular Crash Fatalities. Accident Analysis and Prevention, Vol. 31, 1999, pp. 455-462.

[7] Hand, D., Mannila, H., & Smyth, P., Principles of Data Mining. The MIT Press, 2001.

[8] Kim, K., Nitz, L., Richardson, J., & Li, L., Personal and Behavioral Predictors of Automobile Crash and

Injury Severity. *Accident Analysis and Prevention*, Vol. 27, No. 4, 1995, pp. 469-481.

[9] Kweon, Y. J., & Kockelman, D. M., Overall Injury Risk to Different Drivers: Combining Exposure, Frequency, and Severity Models. Accident Analysis and Prevention, Vol. 35, 2003, pp. 441-450.

[10] Martin, P. G., Crandall, J. R., & Pilkey, W. D., Injury Trends of Passenger Car Drivers In the USA. Accident Analysis and Prevention, Vol. 32, 2000, pp. 541-557.

[11] Mayhew, D. R., Ferguson, S. A., Desmond, K. J., & Simpson, G. M., Trends In Fatal Crashes Involving Female Drivers, 1975-1998. Accident Analysis and Prevention, Vol. 35, 2003, pp. 407-415.

[12] Mussone, L., Ferrari, A., & Oneta, M., An analysis of urban collisions using an artificial intelligence model. *Accident Analysis and Prevention*, Vol. 31, 1999, pp. 705-718.

[13] Ossiander, E. M., & Cummings, P., Freeway speed limits and Traffic Fatalities in Washington State. Accident Analysis and Prevention, Vol. 34, 2002, pp. 13-18.

[14] Shankar, V., Mannering, F., & Barfield, W., Statistical Analysis of Accident Severity on Rural Freeways. *Accident Analysis and Prevention*, Vol. 28, No. 3, 1996, pp.391-401.

[15] Sohn, S. Y., & Lee, S. H., Data Fusion, Ensemble and Clustering to Improve the Classification Accuracy for the Severity of Road Traffic Accidents in Korea. *Safety Science*, Vol. 4, issue1, February 2003, pp. 1-14.

[16] Tavris, D. R., Kuhn, E. M, & Layde, P. M., Age and Gender Patterns In Motor Vehicle Crash injuries: Improtance of Type of Crash and Occupant Role. Accident Analysis and Prevention, Vol. 33, 2001, pp. 167-172.

[17] Yang, W.T., Chen, H. C., & Brown, D. B., Detecting Safer Driving Patterns By A Neural Network Approach. *ANNIE '99 for the Proceedings of Smart Engineering System Design Neural Network, Evolutionary Programming, Complex Systems and Data Mining*, Vol. 9, pp 839-844, Nov. 1999.

[18] Zembowicz, R. and Zytkow, J. M., 1996. From Contingency Tables to Various Forms of Knowledge in Database. Advances in knowledge Discovery and Data Mining, editors, Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. AAAI Press/The MIT Press, pp.329-349.

[19] Abraham, A., Meta-Learning Evolutionary Artificial Neural Networks, Neurocomputing Journal, Elsevier Science, Netherlands, Vol. 56c, pp. 1-38, 2004.

[20] Moller, A.F., A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning, Neural Networks, Volume (6), pp. 525-533, 1993.

[21] National Center for Statistics and Analysis http://www-nrd.nhtsa.dot.gov/departments/nrd-30/ncsa/NASS.html

[22] Roh J.W., Bessler D.A. and Gilbert R.F., Traffic fatalities, Peltzman's model, and directed graphs, Accident Analysis & Prevention, Volume 31, Issues 1-2, pp. 55-61, 1998.

[23] Peltzman, S., The effects of automobile safety regulation. Journal of Political Economy 83, pp. 677–725, 1975.

[24] Ossenbruggen, P.J., Pendharkar, J. and Ivan, J., Roadway safety in rural and small urbanized areas. Accid. Anal. Prev. 33 4, pp. 485–498, 2001.

[25] Abdalla, I.M., Robert, R., Derek, B. and McGuicagan, D.R.D., An investigation into the relationships between area social characteristics and road accident casualties. Accid. Anal. Prev. 29 5, pp. 583–593, 1997.

[26] Miaou, S.P. and Harry, L., Modeling vehicle accidents and highway geometric design relationships. Accid. Anal. Prev. 25 6, pp. 689–709, 1993.

[27] SVM^{light}. http://www.cs.cornell.edu/People/tj/svm_light/. Access date: May, 2003.

[28] Vapnik, V. N., The Nature of Statistical Learning Theory. Springer, 1995.

[29] Chong M., Abraham A., Paprzycki M., Traffic Accident Data Mining Using Machine Learning Paradigms, Fourth International Conference on Intelligent Systems Design and Applications (ISDA'04), Hungary, ISBN 9637154302, pp. 415-420, 2004.

[30] Chong M., Abraham A., Paprzycki M., Traffic Accident Analysis Using Decision Trees and Neural Networks, IADIS International Conference on Applied Computing, Portugal, IADIS Press, Nuno Guimarães and Pedro Isaías (Eds.), ISBN: 9729894736, Volume 2, pp. 39-42, 2004.

[31] Eui-Hong (Sam) Han, Shashi Shekhar, Vipin Kumar, M. Ganesh, Jaideep Srivastava, Search Framework for Mining Classification Decision Trees, 1996. umn.edu/dept/users/kumar/dmclass.ps

[32] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, 2000.

[33] Abraham, Intelligent Systems: Architectures and Perspectives, Recent Advances in Intelligent Paradigms and Applications, Abraham A., Jain L. and Kacprzyk J. (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, Chapter 1, pp. 1-35, 2002.

Clustering Algorithms in Process Monitoring and Control Application to Continuous Digesters

Timo Ahvenlampi and Urpo Kortela
Department of Process and Environmental Engineering,
Systems Engineering Laboratory, P.O. Box 4300,
FIN-90014 University of Oulu, Finland.
e-mail: timo.ahvenlampi@oulu.fi

Keywords: fuzzy clustering, SOM, fault diagnosis, pulping

Received: November 30, 2004

In this study, the controlability of the Kappa number in two cooking applications is investigated. The Kappa number is one of the quality measures in the pulp cooking process and usually the only on-line measurement. It is a measure of the residual lignin content in the pulp. The cooking of the pulp mainly takes place in the digester, where the significant part of the lignin is removed from the chips. The control of the Kappa number is mainly carried out by temperature before the cooking zone, therefore it is important to get some indication of the quality (Kappa number) beforehand. The Kappa number is predicted before the cooking zone in two different cooking applications with the main variables affecting the Kappa number using a clustering and fault diagnosis system (SOM and fuzzy clustering). The clustering and fault diagnosis system is used also for a monitoring of the input variables. The data is collected from industrial conventional and Downflow Lo-Solids continuous cooking digesters. Good results were achieved using the clustering and fault diagnosis system.

Povzetek: Grupiranje je uporabljeno na dveh kuharskih problemih za nadzorno aplikacijo.

1 Introduction

Industrial processes generate a lot of information for operators. The operators have many measurements to observe and control at the same time. This can be helped by combining the knowledge. Clustering (see e.g. [1] and [2]) is one of the methods for combination, because the information is saved in databases and it is available. Industrial processes are usually highly non-linear and it is very difficult or impossible to make accurate models with conventional modeling techniques. These kinds of systems can be called complex systems. Pulp and paper processes are examples of this kind of systems. Due to the non-linearities and insufficient measurements for physical modeling, neural networks ([3] and [4]) and fuzzy methods ([5], [6], [7],[8], [9], [10], [11], [12] and [13]) have been applied for the modeling and clustering purposes of the industrial systems.

The processes studied are continuous cooking applications. Most of the kraft pulp is produced in the continuous digesters [14]. In a typical chemical pulping process, the pre-treated and penetrated wood chips are fed into the impregnation vessel and pulp digester where lignin is removed from the chips with the aid of chemical reactions. Thus the wood fibres are separated from each other. The kraft pulping process has been widely investigated during recent years (see e.g. [15], [16] and [17]) and the optimal cooking conditions at the single chip scale are well known. The usual problem, however, is that the optimal conditions at the digester scale cannot be ensured. Reasons for this are the large dimensions of the process equipment, inadequate measurements and a residence time of several hours.

The quality of the pulping is characterized e.g. by the pulp's strength, viscosity, yield and Kappa number. The Kappa number indicates the residual lignin content of the pulp in the blow line. The control of the Kappa number is a very important part of the continuous cooking process. A steady blow line Kappa number enables an optimized chemical consumption in the subsequent parts of the fibre line. The quality of the pulp has a major effect on the final paper quality. [18]

The Kappa number is one of the most important quality indicators in the cooking process. Therefore, the control of the Kappa number is important. In conventional cooking, the main control actions are performed in the top of the digester, but the on-line measurement of the Kappa number is in the bottom of the digester. The residence time between these points is about four hours. It is obvious that with a prediction of the Kappa number in the top of the digester, more information is achieved and control actions can be executed earlier than without any prediction. The prediction can give new information of the change in the process state and the direction of a change earlier. In the Downflow Lo-Solids cooking process, the cooking zone begins at the middle of the digester and the cooking temperature is the main variable for Kappa number control.

The main active variables for the Kappa number are temperature, alkali concentration, cooking (residence) time and the wood species. The temperature is controlled prior to the cooking zone. The alkali (white liquor) is added into the several parts of the process, depending on the application. The alkali is impregnated into the chips in the impregnation vessel, before the cooking operation occurs in the digester. The air is removed from the chips before the impregnation vessel in order to ensure the impregnation of the chips with the alkali. The lignin is partly removed in the impregnation vessel, due to the high temperature and alkali addition in the feed of the impregnation vessel. The main lignin removal takes place in the cooking zone, where the temperature is significantly higher than in the impregnation vessel. The pulp is washed in the counter-current washing zone.

The Kappa number is modeled or predicted in several studies, e.g. [19],[20],[21],[22] and [23]. A neural network trained with a back propagation learning rule was used in Dayal [19]. In Musavi *et al.* [20] a radial basis function neural network model was constructed. In Musavi *et al.* [21] a neuro-fuzzy system is utilized in the Kappa number prediction. Gustafson's Kappa number model [23] is used in the real-time Kappa number modeling in the conventional cooking process in [24] and in the Downflow Lo-Solids cooking process in [25].

In this study, the Kappa number is predicted in the two cooking applications before the cooking zone using the main variables affecting the Kappa number. The inputs before the cooking zone (BCZ) and the output (blow line Kappa number) of the model are presented in the Table I.

Table I. Variables of the system.

Variable	Unit
Alkali concentration BCZ	g/l (Na_2O, EA)
Temperature BCZ	K
Production rate BCZ	adt/d
Kappa number BCZ	
Blow line Kappa number	

The prediction model for the Kappa number is constructed by a combination of the SOM [4] and fuzzy clustering [10]. The system is used for prediction and monitoring purposes. SOM is the first clustering tool and also a fault diagnosis system. SOM has been used for monitoring and prediction purposes in [26]. The quantization error is calculated, and if the error is notable, information is given that the prediction can be faulty. This signal is given with color codes. The colors of the traffic light are used as in Ahvenlampi *et al*. [27]. If the system is in a good process state, the signal is green. A slight deviation from the normal process state is indicated using a yellow color, and very significant changes are colored with red. This color code is a very useful tool for the operators. The final prediction model is done with a fuzzy clustering model. In this study, the Gustafson-Kessel [28] fuzzy clustering model, which is a modification of the fuzzy c-means [29] algorithm, is used. The inputs are the main active variables of the Kappa number: the effective alkali, the temperature and the residence time of the chips in the cooking, which is, in our case, the production rate. Also, the Kappa number before the cooking zone was used as an input to the system. The Kappa number was modeled using Gustafson's [23] Kappa number model. The input variables are the same as in Gustafson's Kappa number model. The results for the conventional cooking process are presented for the first time in [30]. In this study, the results for the Downflow Lo-Solids cooking process are also presented. Good results were achieved in both processes using the clustering and fault diagnosis system.

The structure of the paper is an following. The methods used are presented in chapter 2. Results are considered in chapter 3 and discussion and conclusions are displayed in chapters 4 and 5.

2 Methods used

In this chapter, methods used are presented. Empirical and experimental methods were applied. Gustafson's [23] Kappa number model is an empirical model for delignification. Clustering methods, such as fuzzy clustering (see e.g. [9] and [10]) and SOM [4] are also presented. The clustering and fault diagnosis system is formulated using the combination of SOM and the fuzzy clustering model.

2.1 Gustafson's Kappa number model

Gustafson *et al*. [23] have derived a mathematical model consisting of a series of differential equations describing the combined diffusion and kinetics within a wood chip during the kraft pulping process. The model development has been based on the studies of several researchers (see, e.g. [23], [31] and [15]). The results are compared with data from cooks, in which the pulping rates were kinetically controlled, and in which the pulping rates were partially mass transfer controlled.

The lignin removal in the impregnation vessel can be calculated using Gustafson's Kappa number model for the initial phase. The rate equation for the initial phase delignification. is:

$$\frac{dL}{dt} = k_{il} e^{(17.5-8760/T)} L \qquad (1)$$

where L is the lignin content at time t,
k_{il} is a species specific constant and
T is temperature.

The species specific parameter k_{il} in the rate equation in the initial phase is 1. The initial phase seems to be independent of the OH^- concentration. This does not mean one can proceed through this phase without alkali, but only indicates that alkali concentration does not influence the rate.

2.2 Clustering methods

Fuzzy clustering methods can be used in modeling, identification and pattern recognition [29]. In this chapter, sev-

eral objective functions used for Takagi-Sugeno[6] model identification, usually minimized by fuzzy clustering methods, are presented. SOM [4] is also presented in this chapter. Classified data in c clusters is arranged in a vector $Z = \{z_1, z_2, ..., z_N\}$. In this study, the consequent parameters for Sugeno models are estimated using weighted least squares.

2.2.1 Fuzzy c-means

Fuzzy c-means is a widely used algorithm for fuzzy identification. The FCM cost function is usually formulated as [29]:

$$J(Z;U,C) = \sum_{i=1}^{c}\sum_{k=1}^{N} (\mu_{ik})^m D_{ik}^2 \quad (2)$$

where $C = \{c_1, ..., c_c\}$. $\{c_1, ..., c_c\}$ are the cluster centers (prototypes) to be determined, $U = [\mu_{ik}]$ is a fuzzy partition matrix [29] and

$$D_{ik}^2 = (z_k - c_i)^T B (z_k - c_i) \quad (3)$$

is a distance (norm) defined by matrix B (usually the identity matrix), and m is a weighting exponent which determines the fuzziness of the resulting clusters.

2.2.2 Gustafson-Kessel algorithm

Gustafson-Kessel algorithm [28] (Appendix A) is the extension most used by the FCM for identification [9]. In this method, norm can be different with every cluster, and the method has the advantage of looking for variable size hyper ellipsoids. The new distance to be used in (2) becomes:

$$D_{ikBi}^2 = (z_k - c_i)^T B_i (z_k - c_i) \quad (4)$$

In this way, quasi-linear behaviors of the existing operating regimes are detected quite correctly. Improved covariance estimation for Gustafson-Kessel algorithm has been introduced in [32].

2.2.3 Number of the clusters

The decision of the number of the clusters is perhaps the most critical point in fuzzy clustering. Many methods have been introduced for the selection of the clusters, see e.g. [9] and [10].

In this study, fuzzy hypervolume [33] is used in deciding of the clusters. Fuzzy hypervolume is calculated using equation (5)

$$F_{hv} = \sum_{i=1}^{c} [\det (F_i)]^{1/2} \quad (5)$$

where F_i is a fuzzy covariance matrix.

2.2.4 SOM

The SOM [4] (Appendix B) is an unsupervised artificial neural network. The network is normally a two-dimensional mapping / projection of the data group. The visualization of the map is easier with a two-dimensional map. In the training of the SOM network, data points are sequentially introduced to the SOM. In each iteration, the SOM neuron which is closest to the input unit is selected by the equation (6). This unit is the Best Matching Unit (BMU) or winner.

$$\|z - c_c\| = \min_i \{\|z - c_i\|\} \quad (6)$$

The weight vectors are updated using the following formula. Only the weight vectors which are inside the neighborhood radius, are updated.

$$c_i (t+1) = c_i (t) + h_{ci} (t) [z (t) - c_i (t)] \quad (7)$$

2.3 Clustering and fault diagnosis system

The clustering and fault diagnosis system is formulated with the combination of SOM and the fuzzy clustering algorithm. The SOM is used as a first clustering method [34] and a fault diagnosis tool in the system. The SOM is trained with the normal operation data which is normalized. The inputs to the system are the temperature, alkali concentration, production rate and Kappa number before the cooking zone. The output is the Kappa number at the blow line of the digester. The SOM codebook matrix (50 times 40 matrix) is used as input data for fuzzy clustering identification. When the clustering and fault diagnosis system is formulated, the validation data is put through the SOM network and the best matching unit is found. The best matching units are used with the fuzzy clustering model. The quantization errors are used in the coloring of the trends of the measured inputs and the predicted output. The value of the error is used in the color-coding. In the normal process state, the color code is green. Yellow color is used for slight deviations from the normal operation and major changes are colored with a red color code. The structure of the clustering and fault diagnosis system is illustrated in the Fig. 1.

3 Case studies

In this study, clustering and fault diagnosis system is used for monitoring and prediction purposes in conventional and Downflow Lo-Solids continuous cooking digesters. The inputs to the system are monitored and the Kappa number in the blow line is predicted. The clustering and fault diagnosis system is a combination of SOM and fuzzy clustering methods. The modeling data (about 30 000 data points for both applications) was normal operation data from the industrial continuous digesters. The outliers and faulty measurements are filtered out from the data. The inputs are

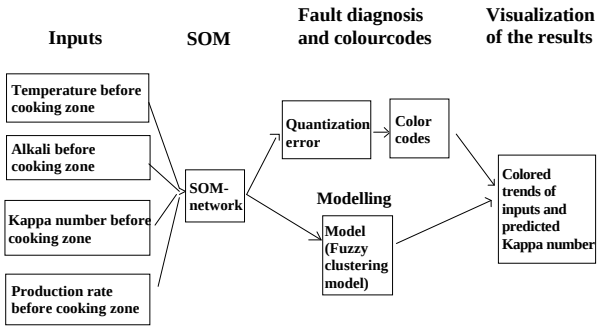


Figure 1: Clustering and fault diagnosis system

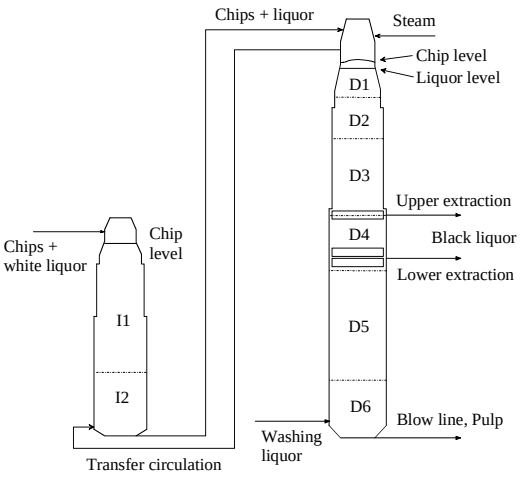


Figure 2: Impregnation vessel and continuous Kamyr digester in conventional cooking process

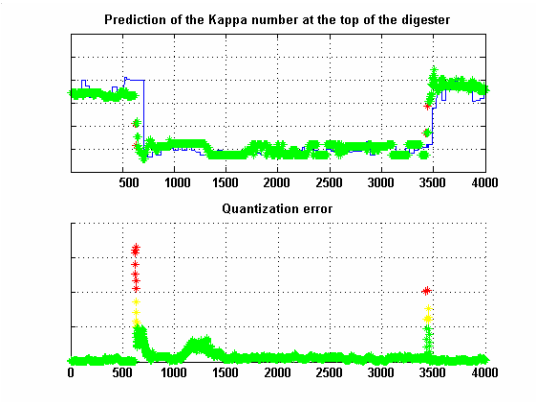


Figure 3: The coloring of the grade changes in the validation period 1.

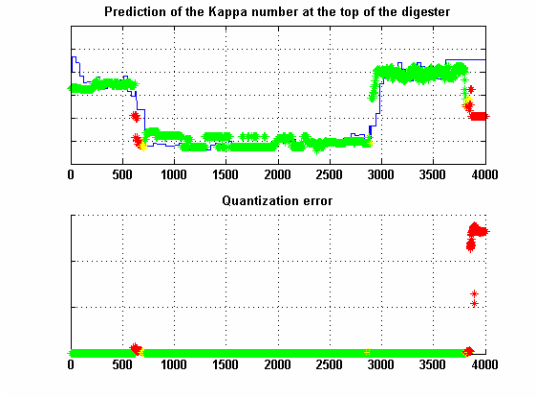


Figure 4: The coloring of the grade changes and shutdown in the validation period 2.

temperature, alkali, Kappa number and production rate before the cooking zone. The output is the predicted Kappa number at the bottom of the digester. The system is validated with the data from the same industrial digester, but from the different time periods.

3.1 Case 1

Case one is a conventional Kamyr process consisting of an impregnation vessel and a steam/liquor phase digester (Fig. 2). The process has been simplified by removing almost all of the original liquor circulations, thus only the upper and lower extraction screens in the end part of the cooking zone are used. A characteristic of this process is the grade changes between softwood and hardwood performed almost every other day. The active alkali concentrations of the white liquor, the digester feed circulation liquor and the two black liquor circulations from the end of the cooking zone are measured. The sulphide concentration of the white liquor is also measured and it is assumed to stay constant during the cooking. Before the latest simplifications of the process, alkali measurements were taken from the extraction screens in the upper part of the digester's cooking zone. These measurements have been utilized in the alkali profile. Temperatures are measured from the various parts of the digester.

The size of the SOM network structure was 50 times 40. The SOM codebook vector (2000 neurons) was an input data for the fuzzy clustering model. The fuzzy clustering method used was the Gustafson-Kessel algorithm. The fuzzy clustering model was divided into 4 local models (clusters) according to the fuzzy hypervolume [33]. The premise membership functions (bell-typed) are projected from the clusters and the local models are obtained using weighted least squares. The fine tuning of the parameters is performed by gradient descent algorithm, see e.g. [8].

The fault diagnosis phase uses different size quantization errors to indicate the deviations from the normal operation points. Thus, this information is used in the coloring of the Kappa number prediction trend with the colors (green, yellow, red). In the Figs. 3 and 4, the situations where the errors deviate and the trends have changing colors are

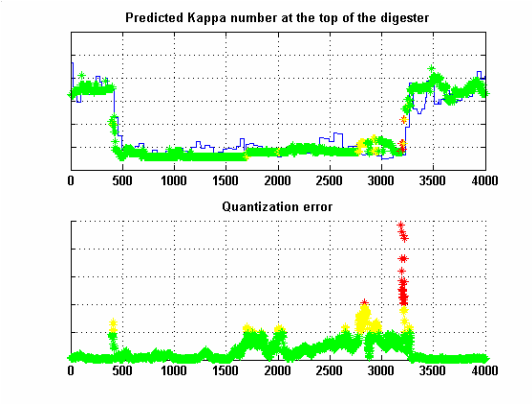


Figure 5: Predicted Kappa number and the quantization error in the validation period 3.

shown. In these process states, the deviations are caused by the grade changes and shutdown. In Fig. 3, there are grade changes at the points of 700 and 3500. In Fig. 4, the grade changes are at the points of about 600 and 2900. The shutdown can be seen in Fig. 4 at the point of 3800.

In Figs. 5 and 6, a faulty process state where the system is not normal can be observed. There are grade changes at the points of about 450 and 3250. A slight deviation can be seen at the point of about 2750, and it can be observed from both Figs. 5 and 6 as a yellow and red trend color. The same kind of example is illustrated in Figs. 7 and 8, where the grade changes are at the points of about 500 and 3750. The operational failure is at the point of 2450, which has been identified by the clustering and fault diagnosis system. In these figures the only significant deviations are colored. Thus, the system is not too sensitive to small deviations. The error size can be used as a tuning factor to the system. The color changing value can be small, if every deviation is desired to be shown, and if only notable disturbances are needed to be shown, the tuning factor can be big.

In Figs. 3-8, the validation results of the clustering and fault diagnosis system in the conventional cooking process are shown. The time period in the figures is one minute. It is the same time period as for the history database in this industrial plant. As it can be seen from the figures, the combined clustering model is accurate and it is able to observe the changes in the process. The clustering and fault diagnosis system can be used in fault diagnosis and for Kappa number prediction purposes.

3.2 Case 2

Case two is a Downflow Lo-Solids [35] cooking process (Fig. 9). The chips are impregnated in the impregnation vessel (I1-I2) and in the first zone (D1) of the digester. Between upper extraction and cooking circulation there is a counter-current washing zone (D2). In this zone, black liquor is displaced with cooking circulation liquor which

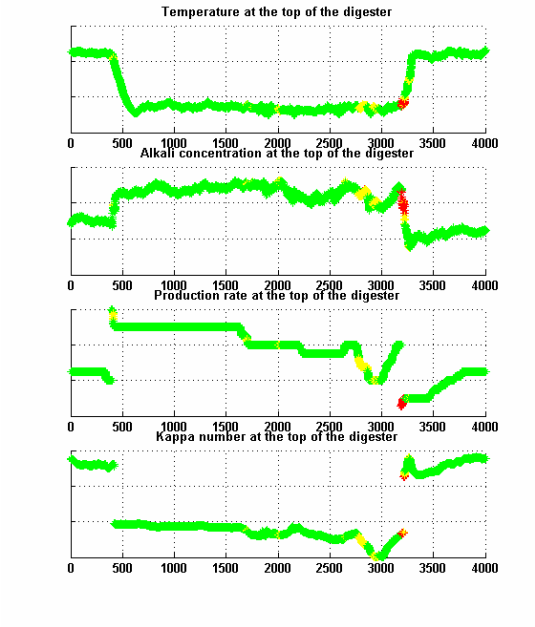


Figure 6: Inputs to the system in validation period 3

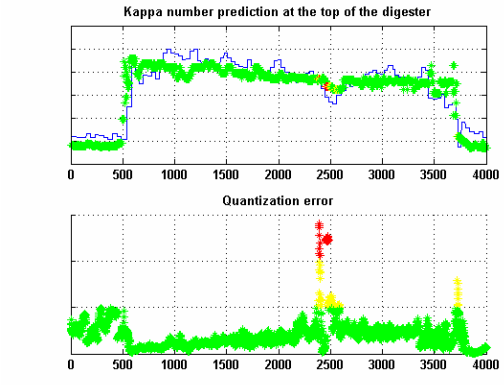


Figure 7: Prediction in validation period 4.

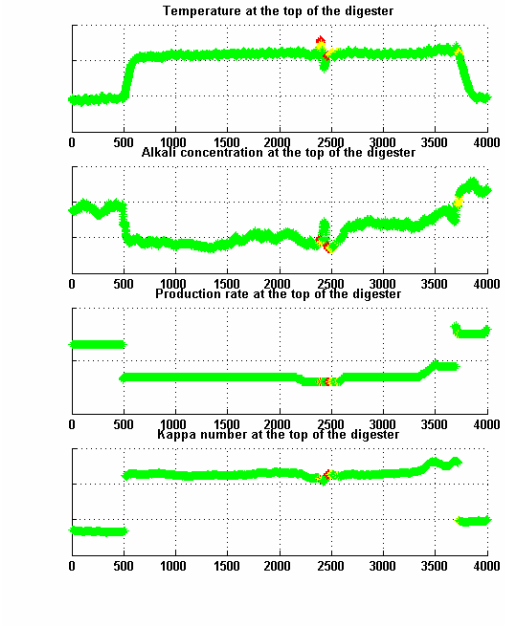


Figure 8: Inputs in the validation period 4.

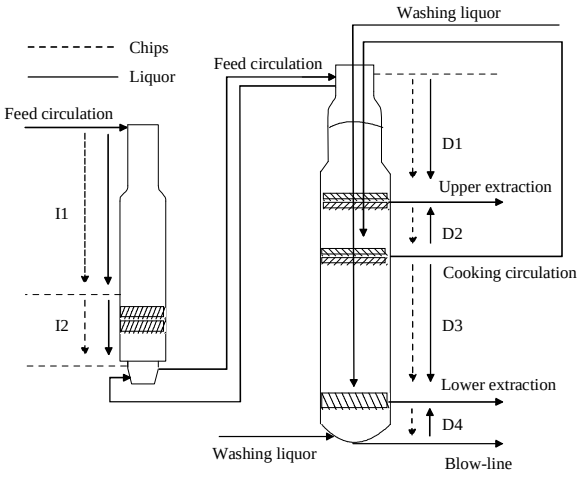


Figure 9: Main flows and flow directions of chips and liquor in impregnation vessel and digester in Downflow Lo-Solids cooking

temperature and alkali concentration are high. The lignin is mainly removed in the comparatively long co-current cooking zone (D3). At the bottom of the digester there is a short washing zone. Softwood chips mainly consist of pine chips with a small amount of spruce chips. Hardwood chips consist mainly of birch chips with a small addition of aspen chips.

The effective alkali concentrations of the white liquor, digester feed circulation liquor, two black liquor extractions and cooking circulation are measured. The white liquor is added to the impregnation vessel's feed circulation, to the digester's feed circulation and to the cooking circulation. The sulphide concentration of the white liquor is measured, and it is assumed to stay constant during the cooking. Temperatures are measured from the liquor circulations and from the heating steam at the top of the digester. A temperature profile from the top of the digester to the cooking circulation is constructed emphasizing the measured temperatures suitably. The temperature profile from the cooking circulation to the blow line is based on the temperature of cooking circulation.

In Downflow Lo-Solids cooking, the Kappa number control is mainly performed by the cooking zone temperature in the middle of the digester (before D3).

The Kappa number prediction is shown in Fig. 10 and the inputs to the system in Fig. 11. The prediction is quite accurate in both species (hardwood at 0-1750 and softwood at 1750-3500). A grade change has occurred at 1750, where the system indicates a disturbance. Another faulty period is between 1850-2100, where the system has turned to a yellow signal.

In Figs. 12 and 13, the results from validation period 2

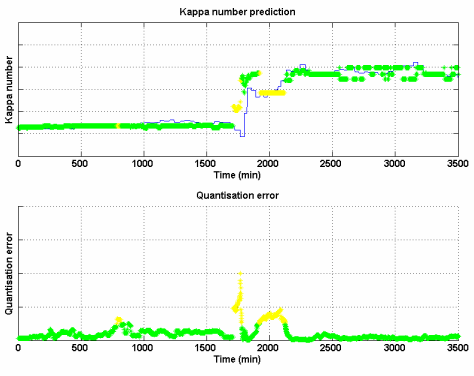


Figure 10: Kappa number prediction in the Downflow Lo-Solids cooking process. (Validation period 1)

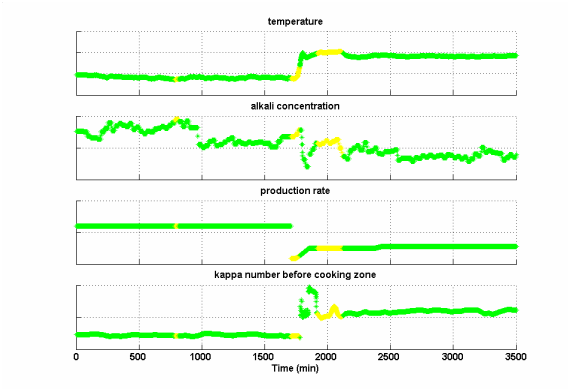


Figure 11: The input variables to the system in the Downflow Lo-Solids cooking process. (Validation period 1).

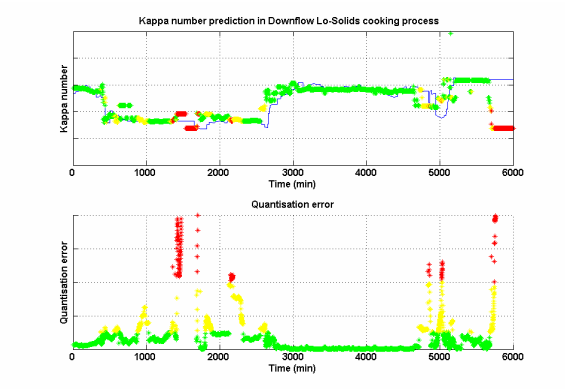


Figure 12: Kappa number prediction in the Downflow Lo-Solids cooking process. (Validation period 2)

are presented. There are grades changes at the points of about 400 and 2700. Shutdown has occurred at the point 5700. There is a faulty process state in both species. In the hardwood, the faulty state is at the period 1300-1600 and in the softwood case, the period is 4700-5100 and after the shutdown at 5700-6000.

4 Discussion

The sampling interval of the on-line Kappa number measurements is about half an hour. Hence, it is useful to also get continuous information about quality properties. The control of the Kappa number is mainly carried out with the cooking temperature, therefore it is important to get an indication of the quality (Kappa number) before the cooking zone in order to execute necessary control actions soon enough.

In this study, a clustering and fault diagnosis system for the monitoring of the process and prediction of the Kappa number in the blow line of the digester is constructed and validated. The system is implemented with a combination of SOM and the fuzzy clustering model.

As shown in Figs. 3-13, the results of the fault diagnosis and clustering system are very accurate. The proposed method is suitable for the optimization and fault diagnosis of the kraft cooking process. In the case of major process changes, the adjustment and verification of the model parameters into the optimal form is quite easy.

Fault diagnosis is carried out using the quantization errors in a coloring of the trends of the input measurements and predicted Kappa numbers. In Figs. 3-13, only significant deviations are colored. Thus, the system is not too sensitive to small deviations. The error size can be used as a tuning factor for the system. The color changing value can be small, if every deviation is desired to be shown, and if only major disturbances are needed to be shown, the tuning factor can be bigger. Color can be used to observe failures in the input measurements or deviation from good operation points. Yellow and red colors indicate also that the

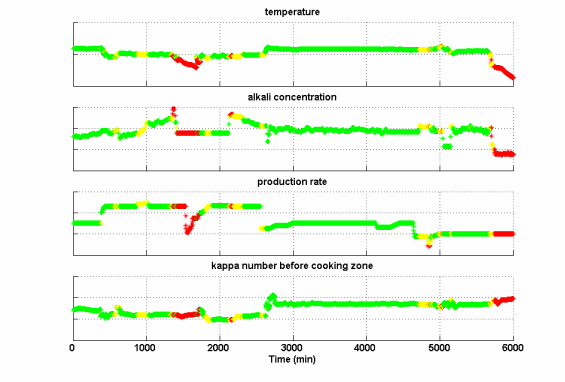


Figure 13: The input variables to the system in Downflow Lo-Solids cooking process. (Validation period 2).

prediction may be inaccurate.

The method has been tested with the conventional and Downflow Lo-Solids continuous cooking digesters and the possibility to implement the system into an automation system is considered. The clustering and fault diagnosis system will be used also as a fault diagnosis and redundant system for Gustafson's Kappa number model.

5 Conclusions

The applicability of SOM and fuzzy clustering approach for the controlability of the Kappa number was considered. The results of the usability of the combined clustering and fault diagnosis system in the monitoring of the conventional and Downflow Lo-Solids continuous cooking processes and the prediction of the Kappa number with the system are shown.

Acknowledgement

This research study was funded by Stora Enso Oyj, Metso Automation Oy, Metsä-Botnia Oy, Andritz Oy and the Na-

tional Technology Agency (Tekes). The authors would like to thank the partners for the special knowledge and process data provided.

Appendix A

Process of Gustafson-Kessel algorithm:

Step 1: Compute the cluster centres:

$$c_i^{(l)} = \frac{\sum \left(\mu_{ik}^{(l-1)}\right)^m z_k}{\sum \left(\mu_{ik}^{(l-1)}\right)^m}, 1 \leq i \leq C$$

Step 2: Compute fuzzy covariance matrix:

$$F_i = \frac{\sum_{k=1}^{N} \left(\mu_{ik}^{(l-1)}\right)^m \left(z_k - c_i^{(l)}\right)\left(z_k - c_i^{(l)}\right)^T}{\sum_{k=1}^{N} \left(\mu_{ik}^{(l-1)}\right)^m}, \quad 1 \leq i \leq C$$

Step 3: Compute the distances:

$$B_i = \rho_i \det\left(F_i\right)^{1/n} F_i^{-1}, 1 \leq i \leq C$$

$$D_{ikBi}^2 = \left(z_k - c_i\right)^T B_i \left(z_k - c_i\right), 1 \leq i \leq C, 1 \leq k \leq N$$

Step 4: Update the partition matrix:

$$\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{C} \left(D_{ikBi}/D_{jkBi}\right)^{2/(m-1)}}$$

iterate until $\left\|U^{(l)} - U^{(l-1)}\right\| < \varepsilon$.

Appendix B

The training of the SOM network is as following:

Step 1: Give initial values for neighborhood radius $h_{ci}\left(t\right)$ and learning rate $\alpha(t)$

Step 2: Choose the steps K

Step 3: Choose one vector z from the learning data Z

Step 4: Find c, BMU (best matching unit) from the initialized network, which distance is closest to the input vector z. Euclidian distance is used.

$$\left\|z - c_c\right\| = \min\left\{\left\|z - c_i\right\|\right\}$$

Step 5: The updating of the weight vectors. Only the weight vectors which are inside the neighborhood radius are updated.

$$c_i\left(t+1\right) = c_i\left(t\right) + \alpha(t) h_{ci}\left(t\right)\left[z\left(t\right) - c_i\left(t\right)\right]$$

Step 6: Set $t = t + 1$. If $t = K$, stop. Otherwise go to step 3.

References

[1] A. K. Jain, R. C. Dubes (1988) *Algorithms for clustering data,* Prentice Hall, Inc., New Yersey.

[2] R. O. Duda, P. E. Hart, D. G. Stork (2001) *Pattern classification, second edition,* John Wiley & Sons, Inc., New York.

[3] C. M. Bishop (1997) *Neural networks for pattern recognition,* Oxford University Press Inc., New York.

[4] T. Kohonen (1997) *Self-organizing maps, second edition,* Springer-Verlag, Heidelberg.

[5] E. Mamdani (1977) Application of fuzzy logic to approximate reasoning using linguistic systems, *IEEE Transactions on Computers* 26(12), pp. 1182-1191.

[6] T. Takagi and M. Sugeno (1985) Fuzzy identification of systems and its applications to modeling and control, *IEEE Transactions on Systems, Man, and Cybernetics* 15(1), pp. 116-132.

[7] M. Sugeno and T. Yasukawa (1993) A fuzzy-logic-based approach to qualitative modeling, *IEEE Transactions on fuzzy systems* 1, pp. 7-31.

[8] E. Kim, M. Park, S. Ji and M. Park (1997) A new approach to fuzzy modeling, *IEEE Transactions on Fuzzy Systems* 5(3), pp. 328-337.

[9] R. Babuska (1998) *Fuzzy modeling for control*, Kluwer Academic Publishers, Boston.

[10] J. C. Bezdek, J. Keller, R. Krisnapuram and N. R. Pal (1999) *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer Academic Publishers, Massaschusetts.

[11] J. Abonyi (2003) *Fuzzy model identification for control,* Birkhäuser, Boston.

[12] M. Sato, Y. Sato and L. C. Jain (1997) *Fuzzy clustering models and applications,* Physica-Verlag, Heidelberg.

[13] F. Hoppner, F Klawonn, R. Kruse, T. Runkler (2000) *Fuzzy cluster analysis: methods for classification, data analysis and image recognition,* John Wiley & sons, ltd. West Sussex, England.

[14] J. Gullichsen (2000) Chemical engineering principles of fiber line operations, In: *Chemical pulping*. J. Gullichsen and C-J. Fogelholm, Eds., Papermaking Science and Technology, Book 6A. Fapet Oy, Jyväskyä, Finland, 2000, pp. 244-327.

[15] N. Agarwal, R. Gustafson (1997) A Contribution to the modeling of kraft pulping, *Can. Chem. Eng. J.* 75(2), pp. 8-15.

[16] K. Walkush, R. G. Gustafson (2002) Application of pulping models to investigate the performance of commercial continuous digesters, *Tappi J*. 1(5), pp. 13-19.

[17] S. Bhartiya, P. Dufour, F. J. III Doyle (2003) Fundamental thermal-hydraulic pulp digester model with grade transition, *AIChE J*. 49(2), pp. 411-425.

[18] K. Leiviskä (1999) Process control in fiber line, In: *Process Control*, Leiviskä K., Ed., Papermaking Science and Technology, Book 14, Fapet Oy, Jyväskylä, Finland.

[19] B.S. Dayal, J.F. MacGregor, P.A. Taylor, R. Kildaw and S. Marcikic (1994) Application of feedforward neural networks and partial least squares regression for modelling kappa number in a continuous Kamyr digester, *Pulp & Paper Canada* 95(1), pp. 26-32.

[20] M.T. Musavi, D.R. Coughlin and M Qiao (1995) Prediction of wood pulp K\# with radial basis function neural networks, *IEEE International Symposium on Circuits and Systems*, 1995. ISCAS '95, Vol. 3, 30 Apr-3 May 1995, pp. 1716 -1719.

[21] M.T. Musavi, C. Domnisoru, G. Smith, D.R. Coughlin and A.L Gould (1999) A neuro-fuzzy system for prediction of pulp digester K-number, *Int. Joint Conf. on Neural Networks*, volume 6, pp. 4253-4258.

[22] K.E. Vroom (1957) The H-Factor, a means of expressing cooking times and temperatures as a single variable, *Pulp. Pap. Mag. Can.* 58, 3, pp. 228-231.

[23] R. G. Gustafson, C. A. Sleicher, W. T. McKean and B. A. Finlayson (1983) Theoretical model of the kraft pulping process. Ind. and Eng. Chem., Process Design and Development 22(1), pp. 87-96.

[24] R. Rantanen, T. Ahvenlampi and U. Kortela (2003) Kappa number profile in continuous cooking – applying Gustafson's model to softwood and hardwood pulping process, *The 4 Biennial Johan Gullichsen Colloquium*, September 10, 2003, Espoo, Finland, pp. 83-92.

[25] R. Rantanen, E. Similä and T. Ahvenlampi (2005) Modeling of Kappa number in Downflow Lo-Solids cooking using Gustafson's model, *Pulp & Paper Canada*. Accepted

[26] J. Abonyi, S. Nemeth, C. Vincze and P. Arva (2003) Process Analysis and Product Quality Estimation by Self-Organizing Maps with an Application to Polyethylene Production, *Computers in industry* 52, pp. 221-234.

[27] T. Ahvenlampi, T. Hietanen, J. Hiltunen, U. Kortela and M. Tervaskanto (2001) Uusien työkalujen kehittäminen metsäteollisuuden osaprosessien hallintaan, *Automation seminar '01*, September 4-6, 2001, Helsinki, Finland, pp. 315-321. In Finnish.

[28] E.E. Gustafson and W.C. Kessel (1979) Fuzzy clustering with a fuzzy covariance matrix, *Proceedings of the IEEE CDC*, San Diego, California, pp. 761-766.

[29] J.C. Bezdek (1981) *Pattern recognition with fuzzy objective function algorithms*, Plenum Press, New York.

[30] T. Ahvenlampi and U. Kortela (2004) Clustering and fault diagnosis approach in controllability of Kappa number, *IEEE 4th international conference on Intelligent Systems Design and Applications (ISDA 2004),* August 26-28, Budapest, Hungary, pp.193-198.

[31] Q. Pu, W. McKean, R. Gustafson (1991) Kinetic model of softwood kraft pulping and simulation of RDH process, *Proceedings of 45th Appita Annual General Conference*, Australia, pp. 187-194.

[32] R. Babuska, P.J. van der Veen, U. Kaymak (2002) Improved covariance estimation for Gustafson-Kessel clustering, *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems*, Vol 2. pp. 1081-1085.

[33] I. Gath and A. B. Geva (1989) Unsupervised Optimal Fuzzy Clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11(7), 773-781.

[34] J. Vesanto and E. Alhoniemi (2000) Clustering of the self-organizing map, *IEEE Transactions on neural networks*, 11(3), pp. 586-600.

[35] B. Marcoccia, R. Laakso and G. Mcclain (1996), Lo-SolidsTM Pulping: principles and applications. *Tappi J*. 79(6), pp. 179-188.

A Multi-agent Approach to Manage a Network of Mobile Agent Servers

Peter Braun
Faculty of Information and Communication Technologies
Swinburne University of Technology
Hawthorn, Victoria 3122, Australia
pbraun@it.swin.edu.au

Jan Eismann, Christian Erfurth, Arndt Döhler, Wilhelm R. and Rossak
Computer Science Department
Friedrich Schiller University Jena
D-07740 Jena, Germany
{eismann, cen, arndt.doehler, rossak}@informatik.uni-jena.de

Keywords: Mobile agents

Received: February 19, 2004

In this paper we present an approach to construct and evolve a network of mobile agent servers. It can be seen as a service that is indispensable for mobile agents to move through the network automatically. Without such a service the programmer of a mobile agent must code the agent's itinerary into its business logic. Our approach has a two-level structure, where agent servers within a subnetwork are combined into a domain, and domains can be connected to each other using a client/server or peer-to-peer technique. Our approach is multi-agent based, that is several stationary and mobile agents communicate to each other to build and evolve the logical network. Main characteristics of our approach are its robustness in failure situations and its high performance, which is shown by results of a first evaluation.

Povzetek: Predstavljena je dvonivojska mreža mobilnih agentov s povečano trdoživostjo.

1 Introduction

Mobile agents are small software entities that can roam the Internet in order to fulfill a user-given task. They allow for task processing that is dynamically distributed over the network by searching for agent servers which offer appropriate services in a network of interconnected platforms [8]. In the last years mobile agents have been a very fast growing area of research and development. While most research was done in the area of code security [22], control algorithms [1], and mobile agent coordination [20], we consider mobile agents to be foremost a promising design paradigm for the architecture of distributed systems.

Many mobile agent toolkits have been developed, for example Aglets [15] by IBM, Concordia [17] by Mitsubishi, and Grasshopper [2] by IKV++. All these toolkits can be considered to have almost product status. Nevertheless, with regard to specific basic system services most mobile agent toolkits are still in their infancy. As an example, many standard services found in middleware components for distributed applications, as for example CORBA [12], are missing even in the above mentioned toolkits. Our aim is to introduce standardized services for distributed agent server networks and to evaluate these services in the Tracy [8] mobile agent toolkit.

1.1 Logical Agent Server Networks

The basic concept we employ is that of a *logical agent server network*. An agent server is the environment on a single computer system that allows the receipt and execution of mobile agents. We define a logical network as an undirected graph in which vertices represent agent servers and an edge exists between a pair of vertices if there is the possibility to transmit mobile agents between the corresponding servers. Not all agent servers must be able to exchange mobile agents due to different transmission protocols, firewalls or private subnetworks that are only reachable via a gateway server. A logical network is a necessary prerequisite for a mobile agent to move through the network automatically. On each server it can ask through a stationary agent, or a service, for the neighboring agent servers and decide to which it will migrate to next.

Without such a network service the agent's programmer has the obligation to code the agent's itinerary into its business logic. While this is sufficient in some applications and in small networks, it is not reasonable to define an agent's route in a world wide network, for example. In such an environment mobile agents must be able to find their itinerary on their own. They must be in a position to react on unreliable network connections and unreliable agent servers and, therefore, modify their itinerary on the fly.

In its minimal variant, a logical network is equal to the

minimal spanning tree guaranteeing that each agent server can be reached. The drawback of this solution is the on average higher number of edges on a path between two nodes, which results in a higher number of migration steps and therefore in a higher execution time of the agent. A smaller number of edges on a path between two nodes results from redundant edges. In a network where all agent servers are connected with each other, called a clique, the maximal number of edges on each path equals one. Both solutions lack any structure that could support the agent in optimizing its itinerary while traversing.

A logical agent server network is the foundation for more sophisticated services of this kind. They all need information about neighboring agent servers or possibly all agent servers currently reachable. One of the research topics is to develop algorithms to plan (semi-)optimal routes for mobile agents with regard to application capabilities (data, user-level services) offered on agent servers and network quality information.Currently we are developing a network performance measuring component for the Tracy toolkit. This component is to measure transmission time to other agent servers periodically and providing this information to mobile agents for planning their optimal route through the network. It uses information of the logical agent server network to find agent servers to which the network quality should be tested. On an even higher level of abstraction the logical agent server network is used to propagate information about applications offered on one agent server to other servers in the network. A mobile agent can use this information to plan its route according to the task that it should fulfill.

Currently, in all of the above mentioned mobile agent toolkits, it is only possible to manage a single *stand-alone* agent server by using some kind of console or graphical user interface. In a logical agent server network, the administrator can obtain an immediate view of all agent servers. In the Tracy toolkit there is already an approach to use this information in combination with its graphical user interface which can be dynamically connected to other running agent servers to administrate them, e.g. to start and stop agents or for checking the status of the server.

1.2 Similar Approaches

As far as we are aware, there is only one agent toolkit, Grasshopper, that offers a service that is distantly related to a logical agent server network. The *region registry* is a central component within a single domain to hold a list of all agent servers. It is not only used to store information about other agent servers, but also information about all agents that are currently residing on all agent servers within a region. Thus, the region registry can be described as a kind of agent tracking service which is also a good approach to find other agents to communicate to. However, multiple region registries cannot be connected, with the consequence that mobile agents cannot find agent servers beyond their region (local domain).

The Grasshopper approach is a good solution either for local area networks or for a small number of agent servers spread over a large region. It offers more than a logical agent server network insofar as it also provides information about the agents in the local domain.

Other agent toolkits, for example Jade [3], provide another approach. A Jade platform contains of several so-called *containers*. One agent server, named the *main container*, maintains a directory of all containers, agents, and services that are provided within a platform. When a new Jade container is started, the administrator has to decide, whether this agent server shall work as main container. Otherwise, the address of the main container has to be defined to let the new agent server register with it afterwards.

1.3 Our Solution

In this paper we present our approach to constructing logical agent server networks. The basic architecture of our approach to mobile agent networks consists of domains, which are limited to subnetworks. All agent servers within a single domain enlist at a central server, which is called *domain manager*. Domains can be connected to each other so that mobile agents can also reach agent servers in other domains. Connecting and disconnecting of agent servers to the network works fully automatic and dynamic.

One characteristic of our approach is its robustness in failure situations. For example we can guarantee that at any time there exists a domain manager for each domain. If a domain manager crashes (because its host agent server crashes) all remaining agent servers vote for becoming the new domain manager. If the original domain manager is relaunched, it can reclaim this role.

Our approach is multi-agent based. The domain management service is completely implemented using stationary and mobile agents. Is does not depend on any specific mobile agent toolkit. Although we have implemented our approach on top of the mobile agent toolkit Tracy it is designed to be portable to any other toolkit with minimal effort. For the rest of this paper, to ground our argumentation and our examples, we will use Tracy as a reference system.

Tracy is a general-purpose mobile agent toolkit. It was designed as an extendable toolkit that consists of a kernel and optional plugins. The kernel only provides those services that are similar in all agent toolkits, for example to execute agents and control agents' life-cycle. All high-level services like agent communication, agent migration, partial solutions of the agent security problem, etc. are implemented as additional plugins. We refer to [7] for more information about Tracy's architecture. A detailed introduction to use and program Tracy is given in [8].

1.4 Related Work

A logical agent server network can be seen as an overlay network and the inceptions of creating and maintaining overlay networks go back to the beginnings of the Inter-

net. The Internet was originally conceived as a peer-to-peer system, in which every network node was a peer and the overall number of nodes was very small. In the following, the Internet has grown enormously, making it impossible to maintain the huge number of nodes as in a peer-to-peer system. Overlay structures were conceived as indispensable means to keep the number of nodes manageable.

A well known example for a overlay structure of a peer-to-peer system is the Domain Name System (DNS) [18, 19], which divides the domain name space in a hierarchical manner and forms a distributed database system. The DNS as a whole works and scales very well, but it was designed for a static network with only few dynamics and only with immobile nodes.

In the late 1990s the upcoming file-sharing systems revive the use of the peer-to-peer paradigm. Napster [21] is a peer-to-peer system with a centralized search facility. This solution scales well in practice because of centralizing search while distributing download. However, centralized solutions lead to a single point of failure and cannot scale completely.

Gnutella [11], another peer-to-peer file-sharing protocol, came originally without any network structure and was a robust, fully decentralized approach. The search algorithm was a simple broadcast with hop limit mechanism (horizon) and led to high network load, so it scales not with the possible number of requests. A second drawback of this solution was the lack of known entrance points into the network at the first time of join Gnutella.

Present Gnutella clones and successors uses additional mechanisms to prevent the drawbacks of the original solution. eDonkey [9] and KaZaA [14] for example use high potential nodes as super-nodes which are connected to each other and thus form a backbone network. A super-node manages a couple of clients, their connections and search requests and forwards the requests to other super-nodes eventually. This approach joins central and decentral features to a hybrid network structure which may scale well, if the number of peers and super-nodes retains well balanced.

The Tracy domain service provides likewise a hybrid overlay structure. It consists of local centralized, quick-reacting domains, that can intercept high network dynamics. Domains can be loosely coupled to each other and are combined in a global hierarchical structure. Agent migration happens fully decentral as in a common peer-to-peer manner. This structure guarantees robustness against breakdowns and scalability in a wide range.

1.5 Structure of this Paper

The rest of this paper is structured as follows: In the following section we will present basic concepts of our approach. Sec. 3 contains a detailed description of our stationary agent which is used to build up the network management service. Sec. 4 focuses on the concept of priorities by which we model different types of agent servers within the network. In Sec. 5 we describe possible failure situations and their solution within our approach. Sec. 6 contains a concise description on how mobile agents can employ the network management service. Sec. 7 provides first results of performance measurements. Finally, the last section gives a summary and an outlook to further development.

2 Basic Concepts

In this section we introduce the basic concepts of our approach. We start by describing the topology of our logical network. Then we argue the usage of stationary and mobile agents for our approach.

2.1 Topology

The topology or architecture of the created logical network can be best described as an interconnection of several domains, see Fig. 1. One domain consists of several agent servers that are connected in one subnetwork and that are able to exchange mobile agents. This restriction derives from a feature of the Tracy mobile agent toolkit that allows mobile agents to be sent using several transmission protocols, like TCP, UDP, or SSL. Agent servers in one domain must, therefore, at least share one transmission protocol to communicate successfully.

Note, that in Tracy each computer system can host more than one agent server. Thus, the number of agent servers per domain is not limited to the number of computers per subnetwork (which equals 255). Usually, there is only one domain per subnetwork. However, several domains can be used, for example if there are agent servers without a common transmission protocol. Another way to split agent servers within a subnetwork into two domains is described later in Sec. 3.2.

In each domain there exists the so-called *domain manager node* which is an agent server, that is responsible to manage all other agent servers in this domain, which are called *domain nodes*. Every domain node has its unique domain manager node, and vice-versa the domain manager node knows all domain nodes that are currently active in its domain. If an agent server starts or stops, it has to register and deregister with the domain manager node. Compare Sec. 3 for more details.

A logical agent server network finally consists of several domains which are connected with each other via their domain manager nodes. Thus, no domain node has a direct connection to any agent server in another domain. There are several ways to connect domain manager nodes to each other. One is to use a central unique so-called *master node* to receive names of other domain manager nodes. We describe this and other techniques later in Sec. 3.2. As indicated in Fig. 1 there is a hierarchy of nodes in our network model: At the lowest level there are domain nodes that represent usual agent servers. A more specialized type of node is a domain manager, which is responsible to manage all nodes in a single domain. At the highest level there

is the master node which is responsible to manage domain managers. In the notion of object-oriented analysis we can state, that a master node is a domain manager node, and a domain manager node is a domain node.

2.2 Multi-Agent based

Our approach to constructing agent server networks is implemented as a multi-agent solution, where one corresponding agent–the domain information agent–exists on each agent server. This domain information agent can play the role of domain node, domain manager, and even master, as is necessary.

It is quite obvious to use agents for this task, as the whole solution is provided as a service in a mobile agent system. Another solution would have been to implement a new protocol on top of TCP or to use Java RMI. By utilizing agents we preserved the independence of the agent toolkit and offer a clear communication interface for other agents that want to utilize the provided information.

The domain information agent is a single stationary agent on each agent server. In the Tracy toolkit only stationary agents are permitted to access the local file system, open network connections, etc. in contrast to mobile agents which will usually not have these rights. This domain information agent employs mobile agents to connect and disconnect agent servers, to inspect network connections, and to inform agent servers in failure situations. The only exception is that we do not use mobile agents in the very first step, that is to find the domain manager node within the subnetwork. This is done by a multicast, see Sec. 3.2.

The reasons to employ mobile agents are the following: In Tracy, being a purely mobile agent toolkit, we are forced to use mobile agents to send messages between agent servers. Tracy does not offer the ability for agents to communicate to remote agents by sending messages (messages being different from agents). On the other hand, exchanging information between remote agents can be obtained by a very small mobile agent. Fortunately, there is no penalty for using mobile agents for those tasks in the Tracy system, because migration is performed very fast [6, 4]. Remote communication using Java Remote Method Invocation would not be faster. In addition, it would have lead to a complete new communication channel in addition to the core agent server, introducing more dependencies and basically redundant services.

There is one scenario that illustrates nicely the main advantage of mobile agents, that is moving code close to the data instead of moving the data to the code. When a domain manager node is in need of other domain manager nodes to be connected to, the master node can be searched for. It holds a list of other domain manager nodes which can be filtered according to some *neighbor* relation. As this data base can be very large in size, it is obviously a better strategy to choose suitable neighboring agent servers directly at the master node. The standard solution to offer a specific interface lacks the flexibility to define the term *neighbor*. Furthermore, to transmit the whole data base to the domain manager node would cause unnecessary network traffic. We decided to utilize mobile agents to find neighbors implementing the notion of a *neighbor* inside the agent's decision capability. This decision algorithm can be modified by the user and is therefore adaptable to different requirements.

3 The Domain Information Agent

This section gives a detailed description of the main component of our domain manager service, which is the stationary *domain information agent* (DIA). The DIA functions according to the roles assigned to it. Here, we will explain how a DIA determines its role during launching and present different techniques to connect domain information agents to each other.

3.1 Constructing Domains

As described in the last section each agent server holds a specific role in the logical agent server network: domain node, domain manager node, and master node. In our solution we provide only one type of agent, the domain information agent, which has to take care of each of theses roles. This method is obvious due to the necessity of changing roles dynamically in failure situations between a domain node and a domain manager node. For the sake of simplicity, we do not introduce new names for the DIA's roles. Thus, if an agent server is currently a domain manager node, the respective domain information agent has to provide the functionality of this higher level role, too.

The determination of the DIA's role is done semi-automatically when launching the agent. First, the agent is in the role of a domain node, assuming that there is a DIA on another remote agent server which holds already the role of the domain manager node. To find this domain manager node and later register with it, the new agent sends out a UDP multicast message to all computer systems in the same subnetwork. If there is a domain manager node in this subnetwork, it receives the multicast message and answers with a single UDP package containing its URL. This URL can be used to address migrations to. In the second step, the new agent now checks if both agent servers can exchange mobile agents by using the same transmission protocol. In the third step, the new agent sends a mobile agent to the domain manager node to register the new domain node over there. The mobile agent returns to indicate that the registration process was successful. This process works fully automatic and due to the usage of UDP messages and very small mobile agents, the whole registration process concludes in less than 40 ms on average in a 100 Mbit/s network (see Sec. 7 for more performance results).

If no domain manager agent has answered the UDP multicast message or both agent servers do not offer at least one equal transmission protocol, the new agent passes into

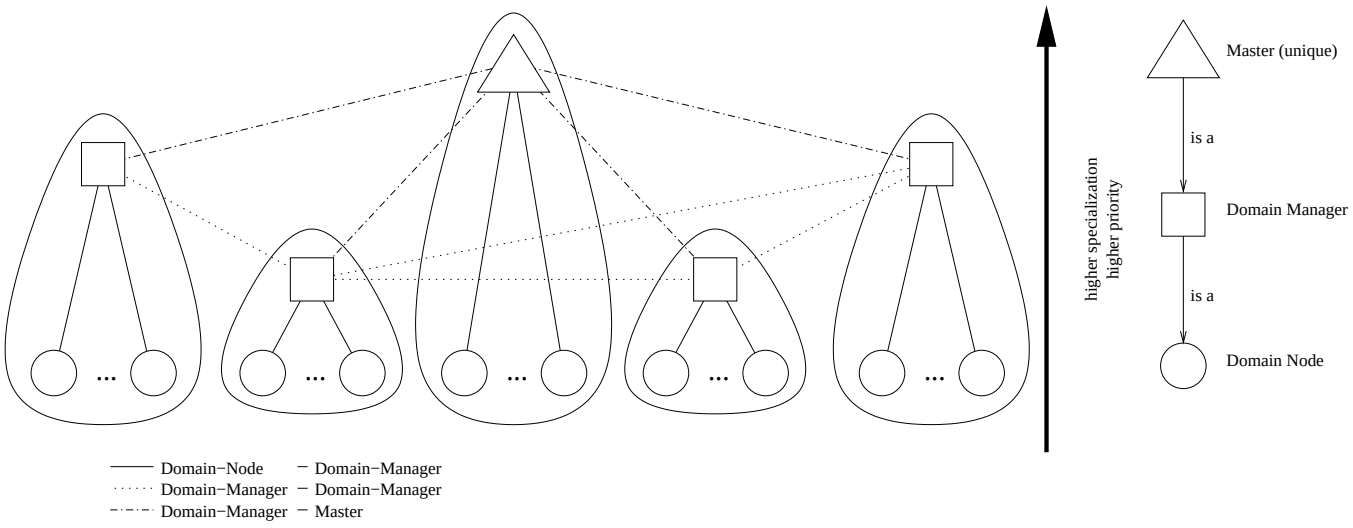


Figure 1: Topology of our logical agent server network. An edge between a pair of vertices indicates that the corresponding agent servers know each other.

the role of a domain manager node itself. As we mentioned briefly in the last section, it can occur that two domains exist in one subnetwork at the same time for the following three reasons: First, if the new agent sends out the UDP multicast message to another UDP port than the domain manager agent receives messages on, the registration process will not start. Second, both agent servers do not speak at least one equal transmission protocol. Third, UDP is an unreliable communication channel. The multicast message as well as the answer package may get lost with the effect that the registration process will fail. Actually, in our implementation, the UDP multicast is resent three times to compensate UDP's unreliability.

As a consequence of a failure in the registration process, a new domain is created within a subnetwork. Agent servers within this subnetwork could now be registered at two separate domain manager nodes. In the current implementation the choice of domain manager nodes is driven by the first-come-first-serves principle, that is the domain manager node that answers the UDP multicast first, is chosen. When the new agent is in the role of the domain manager node it has to connect to other domain manager nodes, see Sec. 3.2 for more information. The only drawback of having two domains in one subnetwork results from slightly increased migration times to agent servers in the other domain inside the same subnetwork. The agent must search for the agent server it wants to visit by first migrating via two domain manager nodes instead of reading the information locally at its current agent server.

The highest level role a domain manager agent may assume is that of the master node. Whereas the above described process works fully automatic, the decision on the master node is done manually by the administrator of the agent server that should become master node. A master keeps its role over its whole life-time. No other domain node or domain manager node may become master node. The master node is one specific node whose name is known to some/all domain manager nodes. From this master they can obtain information about other domain manager nodes to connect to. A master nodes skips the search for a domain manager node and accepts its role immediately. After that, it behaves identically to a domain manager node within its domain.

When shutting down an agent server, we have to distinguish two cases. If the agent server was in the role of a domain node, the domain information agent has to deregister from the domain manager node. If the agent server was in the role of a domain manager node (but not in the role of a master node), it has to hand over this role to another agent server in this domain. The simplest idea is that the current domain manager node randomly selects from the list of all registered domain nodes one agent server that will become the next domain manager node. With one mobile agent that visits all registered agent server within this domain, the resignation of the old and the taking over of the new domain manager node is reported to all domain information agents. In the current implementation the whole selection process is not implemented this way, but is driven by the priorities concept that we explain in Sec. 4. If the master node is shut down, no other domain manager will take over this role. Thus, if the master node is used by domain managers to connect to other domain managers, a master node failure will cause severe problems to maintain the inter-domain structure of the network.

3.2 Connecting Domain Managers

We are now going to present our approach to connect separated domains to one single network. This process is also designed to work fully automatic and allows for dynamic

adaptation of connections. Therefore, our solution does not enforce the construction of a specific network topology at this level of connection. It depends on the connection strategy which topology is going to emerge, for example to either build up a fully connected graph out of all domain manager nodes or to connect only neighboring nodes according to some kind of distance metric.

In the first approach we use the master node to obtain information about other domain manager nodes. After the domain information agent starts in the role of a domain manager it sends a mobile agent to the master node to filter the remote data base of known domain manager nodes. As we mentioned above, the usage of the mobile code approach in this context allows for effective and very flexible remote filtering. The definition of the notion *neighbor* is encapsulated within the mobile agent and thus can easily be adopted to local requirements at the domain manager node. The selection process also influences the degree of connectivity of the resulting graph.

If the mobile agent can obtain at least one neighboring domain manager node it is guaranteed that no islands will exist. The drawback of this approach is the client/server like architecture which contradicts the mobile agent philosophy to some extent. The master node is a bottleneck and single point of failure that should usually not exist in a mobile agent network.

We also support another approach to connect domain manager nodes, one which is comparable to the connection strategies implemented in peer-to-peer systems, like Gnutella [13] or FreeNet [16]. A fresh domain manager node must connect at least to one other domain manager node. This address can be defined by the administrator, or, including our master-approach, obtained by the master node.

During life-time, a domain manager node searches for other domain managers using mobile agents traversing the network. It selects other domain managers according to its distance metric. When shutting down, it saves the current list of neighbor in the local file system to reconnect to them at the next time being launched.

4 The Concept of Priorities

With the simple concepts introduced so far, some problems arise in realistic application situations. As can be deduced from the definition of roles within the logical network, the life-time and the quality of each type is different. We assume that a domain manager node has a longer life-time and a higher reliability than a domain node, which can be a mobile device using a wireless connection. The master's life-time and reliability is assumed to be even higher than that of a domain manager node. However, this idea cannot be found in the approach presented so far.

One shortcoming would result from the selection process which starts when a domain manager node is shutting down. Instead of choosing an arbitrary node this selection process should prevent that a short-living node or unreliable (e.g. on a mobile host) becomes domain manager for a foreseeable short time only. The other drawback is strongly related to this. If the domain manager node is restarted again, it should be able to take over the role of a domain manager node from the present one. Two agent servers starting accidentally at the same time would thus cause a collision problem that should be prevented.

We introduce now a concept of priorities to influence the role of an agent server within the logical network. The priority of a domain information agent is modeled as a value between -128 and +127. This priority is defined by the administrator before the domain information agent is started. It cannot be changed during the agent's life-time, at the earliest when the agent is restarted. The priority value should result from the reliability and long-liveness of this agent server. The higher the value is the more important is the role that this agent server may assume within the network (see also Fig. 1). The default value of an agent server equals 0.

With the concept of priorities, the launching process of a domain information agent changes slightly. When a new domain information agent receives the UDP packages containing the URL and priority information of found domain manager nodes (remember that several domain managers might exist in a single subnetwork), it now compares the priorities of theses nodes with its own. If its own priority is higher, the new node becomes domain manager. In the other case, it tries to register with one of these nodes starting with the one with highest priority.

If a new node becomes domain manager, a process of changing roles is started: A mobile agent is started to visit all domain manager nodes and notifies each to release its role and to fall back to the role of a domain node. Each node is informed about the new domain manager node, so that no new registration process is necessary.

When a domain manager node is shutting down, it selects the next domain manager from the list of all known domain nodes according to their priority. To inform the new domain manager and all connected domain nodes about the new situation, the same process starts as mentioned above.

To prevent that two agent servers starting at the same time become domain manager nodes, the priorities and the agent servers' names can be used. When receiving the UDP package containing URL and priority of another domain manager node, the new node can determine which node is going to take the role of a domain manager node by comparing the priorities of both nodes. If both priorities are equal, the first node according to lexical sorting of their URLs is selected.

5 Recovery from Failure Situations

The handling of failure situations is a very important issue in distributed systems. Typical challenges of mobile agent

systems are especially a consequence of the wide area network character of agent networks. In a mobile agent network it might happen very often that servers go down or are unavailable. As a consequence, agents must search for an other host offering the same services. For the management of a logical agent server network it is, therefore, very important to handle those situations appropriately. The technique to find out that neighboring servers are not available anymore is implemented by the use of mobile agents that are sent to the other server. Those so-called *ping agents* are sent from nodes to managers and from managers to other managers. Currently we do not distinguish between server and network failures and handle both situations identically as a server failure. The following failure situations are detected:

1. **Failure of a domain manager**. Each domain node periodically sends out ping agents to its domain manager. If the migration process fails for any reason, it is assumed that the domain manager is not available any more and that a new domain manager must be defined. The first node that has noticed this situation becomes a domain manager for a limited time, not considering any priority information. All other nodes within this domain will notice this failure situation within a pre-defined period of time and will find this temporary manager node to register with. This period of time equals the maximal time between sending two ping agents (see below). The temporary manager node collects all priority information and can be certain to select the node with the highest priority as the next domain manager. The exchange of roles is performed in the same way as described above in Sec. 3.1. Without the temporary manager node it might happen that exchanging roles between nodes takes place as often as nodes are in the domain.

2. **Failure of a neighboring domain manager**. If a domain manager detects that a neighboring domain manager is not alive anymore, it removes it from the list of known managers. Depending on the the strategy of finding other neighbors it might immediately request new neighboring managers from the master node. If the failed manager has been the only connection between two separate subnetworks it is possible that they become disconnected. Therefore it is important to specify a good minimal degree of connection for the network.

3. **Failure of a node**. If a domain manager detects that a node is not alive anymore, it is removed from the list of known domain nodes. If the node is restarted, it can register with the manager again.

If any of these failure situations can be attributed to a network failure instead of a node failure, it might happen that the alleged crashed node will be available again after a short period of time. With our approach we can achieve the expected behavior, that is to restore the old state automatically. For example, if the connection between all nodes and the manager node is broken (but the manager node is still alive), a new domain manager will be elected as described above. The disconnected manager notices stepwise that each node is not available any more and removes it from the list of known domain nodes. After the last node has been removed, the domain manager checks if there is another domain manager reachable by using an UDP multicast. If it finds another domain manager the process of comparing priorities and possibly exchanging roles starts. Otherwise, it stays as a domain manager, but tries to find another domain manager until a new domain node has registered.

A very important parameter which influences the quality of service of the logical agent network is the time between two ping agents (ping interval). If it is too long, the network will react sluggish and might hand out outdated information about the network. If the ping interval is too short, network traffic will increase.

If the master node concept is used to connect to other domain managers, a master node failure would cause severe problems to maintain the inter-domain structure of the network. Currently, we have not implemented any approach to handle this failure situation.

6 Usage

After we have illustrated how the network structure is constructed, we will now explain how the information is provided to mobile agents. It is again the domain information agent (DIA) that provides information about known domain nodes and domain managers. A mobile agent can gain this information either by exchanging messages with the DIA, or by observing the blackboard. A description of all types of messages, including the content of the reply message can be found in Tab. 1. All capitalized words are String constants defined in class `DomainInformationAgent`. The following example (Fig. 2) shows an agent that sends a message to the DIA to retrieve all domain nodes. The answer message contains a list of URLs in the message body, where entries are separated using three " `%` " characters. To parse this list of URLs, class `StringArray` in package `de.unijena.tracy.util` can be used. See the Tracy JavaDoc documentation for more information.

The second way to obtain information about the network structure it to observe specific entries on the blackboard. The blackboard is a hierarchically structured container for information to provide information that should be seen by all agents on a single agent server. The DIA publishes information about known domain managers and domain nodes on the blackboard in directories `System/domainmanager/managers` resp. `System/domainmanager/nodes`. Both directories contain sub-directories for each host.

```
import de.unijena.tracy.util.*;
import de.unijena.tracy.agent.*;
import de.unijena.tracy.agentsystem.*;
import de.unijena.tracy.comm.*;
import de.unijena.tracy.domainservice.*;

public class MessageExampleAgent extends MobileAgent{
  public void startAgent() {
    try{
      sendMessage(DomainInformationAgent.DOMAIN_AGENT_NAME,
                  DomainInformationAgent.GET_NODES,
                  null);
    } catch (MessageQueueException e){
      // DomainInformationAgent does not exist
    }
  }
   public void handleMessage(Message msg){
    if (msg.getTyp().equals(DomainInformationAgent.GET_ANSWER)){
      if (msg.getContent() != null){
        String[] servers =
                 StringArray.stringToStringArray(msg.getContent());
      }
    }
  }
  public void systemFailure(){
  }
}
```

Figure 2: An example of an agent that asks the domain information agent for the names of all domain nodes.

Request subject	Reply subject	Reply parameter	Description
GET_NODES	GET_ANSWER	List of URLs	Get a list of all nodes.
GET_MANAGERS	GET_ANSWER	List of URLs	Get a list of all managers.
IS_DM	IS_DM_ANSWER	DM_TRUE/DM_FALSE	Ask for the role.

Table 1: Messages to obtain information from the domain information agent.

Each host directory contains entries with the agent server name. For example, if agent server with name `swiss.uni-jena.de/piz-gloria` is a domain node, the blackboard contains an entry with name `System/domainmanager/nodes/swiss.uni-jena.de/piz-gloria`. Using the blackboard has the advantage to be able to observe single directories or entries and become immediately informed in case of any modification. Doing this, it is for example very easy to notice that new nodes have enrolled on this domain manager. See [5] for more information on how to access and observe the blackboard.

7 Quality of our Approach

The quality of an approach for managing logical agent server networks is influenced by the quality of the resulting network and the performance of the whole service in failure situations. The first issue will be a topic of further work, where we want to evaluate the resulting network quality with regard to different connection strategies and distance metrics. This research is mostly done on an application level, for example to evaluate the time for an agent to route its way through the network considering the information provided by the logical network.

For the moment, we have evaluated our approach with regard to the performance of the pure management services. Our results show on the one hand that our approach causes no measurable overhead for the whole network traffic and on the other hand that the network can react very fast in failure situations and can achieve a stable state within a very short period of time.

All our experiments were conducted in a cluster of 8 computer systems running Linux on a processor with 800 MHz and in a network with 100 MBit/s bandwidth. The UDP time-out is set to 100 ms. The ping interval is set to 1000 ms.

Our results are summarized in Tab. 2. Each experiment was repeated 10 times and given results are mean values. Times do not include start-up times of the agent servers. In experiment 1 a new agent server registers at an existing domain manager. The time consists of sending the UDP multicast message to find the domain manager, checking transmission protocols, and register with the domain manager by using a mobile agent. In experiment 2 a new agent server is started and there is no other domain manager in the subnetwork. Thus, it becomes a new domain manager after it has waited for three UDP time-out periods (which in sum are 300 ms). No mobile agent is used in this experiment. In experiment 3 the current domain manager is shut down and delegates the role of the domain manager to a known agent server. The mobile agent is used to inform all nodes of the new domain manager and to delegate the role to the new domain manager. In experiment 4 the current domain manager is manually stopped to indicate a failure situation. The measured time is needed for the other seven agent servers to notice this situation (after at last 1000 ms) and to vote for becoming new domain manager, including the whole process of changing roles using a mobile agent. During this process 24 migrations are performed in parallel to connect nodes to the temporarily domain manager, followed by 8 migrations to change roles. Finally, in experiment 5 a new domain manager registers at two other domain managers, located in other LANs in Jena, using the master approach. The master is located at Weimar university (network bandwidth 34 Mbit/s). The mobile agent is used to search for the neighboring domain managers and to register over there.

8 Using Domain Manager Functionalities to Propagate Services

A possible application of the domain manager concept is to use it as a basis to design an information service for agents. *Information service* in this context means to support agents to search actively for those services they need to fulfill their user-defined task. It is the agent that locates available services, maps them to its needs, and autonomously charts a best possible route through the network to reach them.

In general, services are provided in the network by an agency or an agent within an agency. They are distributed over the nodes in the available network. The idea is to improve the autonomy of agents in a way that is transparent to the end user. The proposed information service would make it possible for the end user to simply state *what* he or she wants the agent to do, instead of *how*. This means that the end user could avoid to program a dedicated route into the agent's code, a route that is based on a most likely incomplete and possibly outdated human perception of the network. The human owner of a mobile agent could leave it open where fitting information and processing capability is collected and utilized. The agent and its supporting infrastructure will take care of the rest.

Of course this means that each agent, and the network

Experiment	Description	Time [ms]
1	Register a single node with an existing domain manager	40
2	Start a new domain manager	442
3	Shut-down a domain manager	558
4	7 server vote to become new domain manager	1542
5	Register a new domain manager at two neighboring managers	655

Table 2: Inspected scenarios for the performance evaluation. Given times are mean value of 10 experiments each.

of agencies they use, must provide a couple of new capabilities: Advertise and describe available services; match services to the user's orders; locate the nodes where those services are available; find a possibly optimal route through the network to travel to all indicated nodes/agencies and actually trigger service execution before returning home with the desired result. We are currently designing and implementing support for the tasks that tackle advertising of services and location of the respective nodes. Matching is simply based on a fixed set of keywords and, thus, kept very simple. Routing is open to a variety of solutions, including graph-based algorithms or methods from artificial intelligence. In this short overview we will focus on the most basic problems, that is to advertise and locate services in a most likely dynamic network of inter-linked domains.

Domain nodes already offer various services to mobile agents, either themselves or through agents they currently host. These services are well known to all agents that currently reside at the domain node. However, to make these services known to all agents in the domain, each domain node has to publish a service list. This can in our environment be done by a relatively simple extension of the domain manager concept: Each node's service list is transmitted to the domain manager node using the already necessary exchange of mobile agents during the registration process. Additional information about services available on the domain node is simply attached to the registration agent. Ping agents are used to hold this information up to date. (Remember: Ping agents are used to check if neighboring servers are still reachable. Such agents are exchanged between the domain manager and the domain nodes on a regular basis - see section 5).

The set of transmitted service lists can be seen as a service map of the local domain that is placed at the respective domain manager node. Agents are now able to use this map to match services, chart a route, etc. To avoid the migration to the service map, that is the domain manager node, this map can be mirrored at every domain node, again by simply using ping agents. If there are connections to other domains available - this is already possible on the basis of links between domain manager nodes - a summary of a domain specific service map can be propagated to other domains.

We improve the service map a little bit more by collecting additional technical information, e. g. characteristics like bandwidth, latency etc., and adding them to the service map. This would help the agent to plan the trip more exactly by taking into account also quality and reliability of available connections between nodes. To achieve this task a net sensing module quantifies the line characteristics in certain intervals by performing measurement experiments. The extracted data is collected and stored on the local domain node as well as sent to other domain nodes. This can be done very effectively during such an experiment by using the packages that have to be sent over the network for measurement purposes anyway. The actuality of the line characteristics depends on the frequency of the measurement experiments. The more dynamic the network is, the higher the frequency should be. In addition, forecast modules may be used to calculate the next expected value for extracted characteristics.

At this point a complete map with services and line characteristics of the local domain is available. This map is provided to mobile agents on every node in the domain that hosts an agency. By now such an agent is not only able to chart a service-oriented route that allows it to support its task, but is also able to estimate migration times and determine an optimized migration strategy.

As a possible alternative to the specified map-building mechanism the whole process of creating the enhanced service map could be based solely on the infrastructure of net sensing modules. This would avoid the (mis-)use of register or ping agents, as proposed so far. As already done with line data, the service map could be transmitted to other domain nodes using the bandwidth experiment packages. This would minimize any coupling between the domain manager and the service propagation subsystems and provide a quick and reliable basic infrastructure for service propagation. Therefore, currently the implementation of net sensing modules is our highest priority. For more detailed information regarding the use of the Tracy domain service in this context we refer to [10].

9 Conclusions and Further Work

In this paper we introduced the concept of a logical agent server network, which is a necessary service for mobile agents to roam a network automatically. We have presented our approach which is multi-agent based. On each agent server a stationary domain information agent is responsible to provide information about neighboring agent servers to mobile agents. The stationary agent uses several kinds of mobile agents to communicate to other domain information agents on remote agent servers. We introduced the

priorities concept to model different kinds of agent server nodes and to influence to process of finding domain managers. We have implemented our approach successfully in the Tracy toolkit and we presented results of first performance measurements.

Currently, we are developing a network performance measurement tool which uses our logical agent server network to locate agent servers to test latency and bandwidth. This information is provided to mobile agents, which use them to determine a fast route through the network.

In a next step we will evaluate the quality of the evolved network. Therefore, we will take an application level view and compare different connection strategies for domain managers and their influence on the performance of mobile agents traversing the network.

References

[1] Joachim Baumann. *Mobile Agents: Control Algorithms*, volume 1658 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[2] Christoph Bäumer, Markus Breugst, Sang Choy, and Thomas Magedanz. Grasshopper — A universal agent platform based on OMG MASIF and FIPA standards. In Ahmed Karmouch and Roger Impey, editors, *Mobile Agents for Telecommunication Applications, Proceedings of the First International Workshop (MATA 1999), Ottawa (Canada), October 1999*, pages 1–18. World Scientific Pub., 1999.

[3] Fabio Bellifimine, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. Jade – A White Paper. *EXP in search of innovation*, 3(3):6–19, 2003.

[4] Peter Braun, Jan Eismann, and Wilhelm R. Rossak. Various Performance Experiments with the Mobile Agent System Tracy. Technical Report 11/01, Friedrich-Schiller-Universität Jena, Institut für Informatik, 2001.

[5] Peter Braun, Christian Erfurth, and Wilhelm R. Rossak. An Introduction to the Tracy Mobile Agent System. Technical Report Math/Inf/00/24, Friedrich-Schiller-Universität Jena, Institut für Informatik, September 2000.

[6] Peter Braun, Christian Erfurth, and Wilhelm R. Rossak. Performance Evaluation of Various Migration Strategies for Mobile Agents. In Ulrich Killat and Winfried Lamersdorf, editors, *Fachtagung Kommunikation in verteilten Systemen (KiVS 2001), Hamburg (Germany), February 2001*, Informatik aktuell, pages 315–324. Springer-Verlag, 2001.

[7] Peter Braun, Ingo Müller, Sven Geisenhainer, Volkmar Schau, and Wilhelm R. Rossak. A service-oriented software architecture for mobile agent toolkits. In *11th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2004) May 2004, Brno (Czech Republic)*, pages 550–556. IEEE Computer Society Press, 2004.

[8] Peter Braun and Wilhelm R. Rossak. *Mobile Agents – Basic Concept, Mobility Models, and the Tracy Toolkit*. Morgan Kaufmann Publishers, December 2004.

[9] www.edonkey2000.com.

[10] Christian Erfurth, Arndt Döhler, and Wilhelm R. Rossak. A First Look at the Performance of Autonomous Mobile Agents in Dynamic Networks. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9, Big Island, Hawaii (USA), January 2004*. IEEE Computer Society, 2004.

[11] www.gnutella.com.

[12] Object Management Group. The Common Object Request Broker Architecture, Rev. 2.2, February 1998.

[13] Gene Kan. Gnutella. In Oram [21], pages 94–122.

[14] www.kazaa.com.

[15] Danny B. Lange and Mitsuru Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.

[16] Adam Langley. Freenet. In Oram [21], pages 123–132.

[17] Mitsubishi Electric ITA. *Mobile Agent Computing – A White Paper*, 1998.

[18] Paul Mockapetris. Domain names - concepts and facilities, 1987. Internet Engineering Task Force, RFC 1034.

[19] Paul Mockapetris. Domain names - implementation and specification, 1987. Internet Engineering Task Force, RFC 1035.

[20] Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors. *Coordination of Internet Agents: Models, Technologies, and Applications*. Springer-Verlag, 2001.

[21] Andy Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[22] Giovanni Vigna, editor. *Mobile Agents and Securtiy*, volume 1419 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of S**love**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 219 385
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica LaTeX format and figures in `.eps` format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than ten years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

Informatica WWW:

large**http://ai.ijs.si/informatica/**

Referees:

Witold Abramowicz, David Abramson, Adel Adi, Kenneth Aizawa, Suad Alagić, Mohamad Alam, Dia Ali, Alan Aliu, Richard Amoroso, John Anderson, Hans-Jurgen Appelrath, Iván Araujo, Vladimir Bajič, Michel Barbeau, Grzegorz Bartoszewicz, Catriel Beeri, Daniel Beech, Fevzi Belli, Simon Beloglavec, Sondes Bennasri, Francesco Bergadano, Istvan Berkeley, Azer Bestavros, Andraž Bežek, Balaji Bharadwaj, Ralph Bisland, Jacek Blazewicz, Laszlo Boeszoermenyi, Damjan Bojadžijev, Jeff Bone, Ivan Bratko, Pavel Brazdil, Bostjan Brumen, Jerzy Brzezinski, Marian Bubak, Davide Bugali, Troy Bull, Sabin Corneliu Buraga, Leslie Burkholder, Frada Burstein, Wojciech Buszkowski, Rajkumar Bvyya, Giacomo Cabri, Netiva Caftori, Particia Carando, Robert Cattral, Jason Ceddia, Ryszard Choras, Wojciech Cellary, Wojciech Chybowski, Andrzej Ciepielewski, Vic Ciesielski, Mel Ó Cinnéide, David Cliff, Maria Cobb, Jean-Pierre Corriveau, Travis Craig, Noel Craske, Matthew Crocker, Tadeusz Czachorski, Milan Češka, Honghua Dai, Bart de Decker, Deborah Dent, Andrej Dobnikar, Sait Dogru, Peter Dolog, Georg Dorfner, Ludoslaw Drelichowski, Matija Drobnič, Maciej Drozdowski, Marek Druzdzel, Marjan Družovec, Jozo Dujmović, Pavol Ďuriš, Amnon Eden, Johann Eder, Hesham El-Rewini, Darrell Ferguson, Warren Fergusson, David Flater, Pierre Flener, Wojciech Fliegner, Vladimir A. Fomichov, Terrence Forgarty, Hans Fraaije, Stan Franklin, Violetta Galant, Hugo de Garis, Eugeniusz Gatnar, Grant Gayed, James Geller, Michael Georgiopolus, Michael Gertz, Jan Goliński, Janusz Gorski, Georg Gottlob, David Green, Herbert Groiss, Jozsef Gyorkos, Marten Haglind, Abdelwahab Hamou-Lhadj, Inman Harvey, Jaak Henno, Marjan Hericko, Henry Hexmoor, Elke Hochmueller, Jack Hodges, Doug Howe, Rod Howell, Tomáš Hruška, Don Huch, Simone Fischer-Huebner, Zbigniew Huzar, Alexey Ippa, Hannu Jaakkola, Sushil Jajodia, Ryszard Jakubowski, Piotr Jedrzejowicz, A. Milton Jenkins, Eric Johnson, Polina Jordanova, Djani Juričič, Marko Juvancic, Sabhash Kak, Li-Shan Kang, Ivan Kapustøk, Orlando Karam, Roland Kaschek, Jacek Kierzenka, Jan Kniat, Stavros Kokkotos, Fabio Kon, Kevin Korb, Gilad Koren, Andrej Krajnc, Henryk Krawczyk, Ben Kroese, Zbyszko Krolikowski, Benjamin Kuipers, Matjaž Kukar, Aarre Laakso, Sofiane Labidi, Les Labuschagne, Ivan Lah, Phil Laplante, Bud Lawson, Herbert Leitold, Ulrike Leopold-Wildburger, Timothy C. Lethbridge, Joseph Y-T. Leung, Barry Levine, Xuefeng Li, Alexander Linkevich, Raymond Lister, Doug Locke, Peter Lockeman, Vincenzo Loia, Matija Lokar, Jason Lowder, Kim Teng Lua, Ann Macintosh, Bernardo Magnini, Andrzej Małachowski, Peter Marcer, Andrzej Marciniak, Witold Marciszewski, Vladimir Marik, Jacek Martinek, Tomasz Maruszewski, Florian Matthes, Daniel Memmi, Timothy Menzies, Dieter Merkl, Zbigniew Michalewicz, Armin R. Mikler, Gautam Mitra, Roland Mittermeir, Madhav Moganti, Reinhard Moller, Tadeusz Morzy, Daniel Mossé, John Mueller, Jari Multisilta, Hari Narayanan, Jerzy Nawrocki, Rance Necaise, Elzbieta Niedzielska, Marian Niedq'zwiedziński, Jaroslav Nieplocha, Oscar Nierstrasz, Roumen Nikolov, Mark Nissen, Jerzy Nogieć, Stefano Nolfi, Franc Novak, Antoni Nowakowski, Adam Nowicki, Tadeusz Nowicki, Daniel Olejar, Hubert Österle, Wojciech Olejniczak, Jerzy Olszewski, Cherry Owen, Mieczyslaw Owoc, Tadeusz Pankowski, Jens Penberg, William C. Perkins, Warren Persons, Mitja Peruš, Fred Petry, Stephen Pike, Niki Pissinou, Aleksander Pivk, Ullin Place, Peter Planinšec, Gabika Polčicová, Gustav Pomberger, James Pomykalski, Tomas E. Potok, Dimithu Prasanna, Gary Preckshot, Dejan Rakovič, Cveta Razdevšek Pučko, Ke Qiu, Michael Quinn, Gerald Quirchmayer, Vojislav D. Radonjic, Luc de Raedt, Ewaryst Rafajlowicz, Sita Ramakrishnan, Kai Rannenberg, Wolf Rauch, Peter Rechenberg, Felix Redmill, James Edward Ries, David Robertson, Marko Robnik, Colette Rolland, Wilhelm Rossak, Ingrid Russel, A.S.M. Sajeev, Kimmo Salmenjoki, Pierangela Samarati, Bo Sanden, P. G. Sarang, Vivek Sarin, Iztok Savnik, Ichiro Satoh, Walter Schempp, Wolfgang Schreiner, Guenter Schmidt, Heinz Schmidt, Dennis Sewer, Zhongzhi Shi, Mária Smolárová, Carine Souveyet, William Spears, Hartmut Stadtler, Stanislaw Stanek, Olivero Stock, Janusz Stokłosa, Przemysław Stpiczyński, Andrej Stritar, Maciej Stroinski, Leon Strous, Ron Sun, Tomasz Szmuc, Zdzislaw Szyjewski, Jure Šilc, Metod Škarja, Jiři Šlechta, Chew Lim Tan, Zahir Tari, Jurij Tasič, Gheorge Tecuci, Piotr Teczynski, Stephanie Teufel, Ken Tindell, A Min Tjoa, Drago Torkar, Vladimir Tosic, Wieslaw Traczyk, Denis Trček, Roman Trobec, Marek Tudruj, Andrej Ule, Amjad Umar, Andrzej Urbanski, Marko Uršič, Tadeusz Usowicz, Romana Vajde Horvat, Elisabeth Valentine, Kanonkluk Vanapipat, Alexander P. Vazhenin, Jan Verschuren, Zygmunt Vetulani, Olivier de Vel, Didier Vojtisek, Valentino Vranić, Jozef Vyskoc, Eugene Wallingford, Matthew Warren, John Weckert, Michael Weiss, Tatjana Welzer, Lee White, Gerhard Widmer, Stefan Wrobel, Stanislaw Wrycza, Tatyana Yakhno, Janusz Zalewski, Damir Zazula, Yanchun Zhang, Ales Zivkovic, Zonling Zhou, Robert Zorc, Anton P. Železnikar

Informatica

An International Journal of Computing and Informatics

Archive of abstracts may be accessed at USA: http://, Europe: http://ai.ijs.si/informatica, Asia: http://www.comp.nus.edu.sg/ liuh/Informatica/index.html.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2005 (Volume 29) is
– 60 Euro (80 USD) for institutions,
– 30 Euro (40 USD) for individuals, and
– 15 Euro (20 USD) for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

LaTeX Tech. Support: Borut Žnidar.
Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrežje 11, Ljubljana, Slovenia.
Printed by Biro M, d.o.o., Žibertova 1, 1000 Ljubljana, Slovenia.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. Drago Torkar, Jožef Stefan Institute: Tel (+386) 1 4773 900, Fax (+386) 1 219 385, or send checks or VISA card number or use the bank account number 900–27620–5159/4 Nova Ljubljanska Banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)
ACM Slovenia (Dunja Mladenič)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, ACM Digital Library, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Computer Literature Index, Cur. Cont. & Comp. & Math. Sear., Current Mathematical Publications, Cybernetica Newsletter, DBLP Computer Science Bibliography, Engineering Index, INSPEC, Linguistics and Language Behaviour Abstracts, Mathematical Reviews, MathSci, Sociological Abstracts, Uncover, Zentralblatt für Mathematik

The issuing of the Informatica journal is financially supported by the Ministry of Education, Science and Sport, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 29 Number 1 May 2005 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Informatica **29** (2005) Number 1, pp. 1–121