Learning Predictive Qualitative Models with Padé

Jure Žabkar, Martin Možina, Ivan Bratko and Janez Demšar
University of Ljubljana, Faculty of Computer and Information Science, Tržaška 25, Ljubljana
E-mail: jure.zabkar|martin.mozinalivan.bratkoljanez.demsar}@fri.uni-lj.si

Keywords: qualitative modelling, machine learning

Received: April 18, 2010

Qualitative models are similar to regression models, except that instead of numerical predictions they provide insight into how a change of a certain input variable affects the output within a context of other inputs. Although people usually reason qualitatively, machine learning has mostly ignored this type of model. We present a new approach to learning qualitative models from numerical data. We describe Padé, a suite of methods for estimating partial derivatives of unknown sampled target functions. We show how to build qualitative models using standard machine learning algorithms by replacing the output variable with signs of computed derivatives. Experiments show that the developed methods are quite accurate, scalable to high number of dimensions and robust with regard to noise.

Povzetek: Predstavljena je nova metoda za učenje iz kvalitativnih podatkov, imenovana Padé. Temelji na ocenjevanju parcialnih odvodov neznane vzorčene ciljne funkcije.

1 Introduction

Qualitative models describe quantitative relations in qualitative terms, for instance, *the more it rains and the longer I stay in the rain, the wetter I will get (unless I have an umbrella)*. Although seemingly inferior to the more accurate numerical models, there are many reasons why qualitative models are interesting for artificial intelligence.

One of the goals of artificial intelligence, according to one of its founding fathers Alan Turing, is to mimic the natural, human intelligence. In everyday life we intuitively use qualitative models, not numerical equations. For instance, the complete, realistic equation for behaviour of a child swing would be extremely complicated, yet a five year child knows how to "operate" the swing, and can describe her actions *qualitatively*, *e.g.* when to lean forward and backward to regulate the amplitude.

Induced qualitative models can offer more insight into the domain than numerical ones. The standard approach to regression modelling is fitting the data to a polynomial or another chosen function template. Although such models are sometimes considered symbolic, they do not offer any useful insight. For instance, the true and insightful numerical symbolic model for swinging of a simple pendulum would be a sine function like the one we get by solving the corresponding differential equations. Such solutions are difficult to induce from data using the current regression modelling tools. In contrast, qualitative model can provide a simple, but correct conceptual description: the pendulum swings back and forth. Given enough data, we can discover that the amplitude of the pendulum eventually decreases until the pendulum stops. Given data on multiple pendulums, we find out that longer strings yield longer periods, and that changing the weight has no effect. While these descriptions are insufficient for computing any actual periods, they often provide all the insight we need. For instance, a practical task may be to make the period of a pendulum match that of another one. The guidance provided by the qualitative model – *increase the length if the period is too long* and vice versa – would suffice to accomplish the task. Even when the final goal is to have a quantitative model, the qualitative one can be helpful in its construction [12].

Finally, such models can also be more applicable than numerical ones. For a simple example from economics, consider the law of demand: "the higher the price, the shorter the queue (other things left unchanged, less people are willing to buy things for a higher price)." Any numerical description of this relation would fail to give exact predictions since it would include variables which are not measurable with sufficient precision. Qualitative models, on the other hand, deliver what they promise, that is, correct qualitative predictions. The above simple qualitative rule is routinely, although not necessarily consciously, used to control the market prices.

The field of machine learning, which developed many methods for induction of (numerical) regression models, showed surprisingly little interest in learning of qualitative models from data. We will describe a suite of new machine learning algorithms with a common name Padé (an acronym for "partial derivative", and the name of a famous French mathematician). Padé first computes partial derivatives with respect to all independent attributes for all examples appearing in the data. Then it discards the quantitative information, the magnitude, which is difficult to estimate precisely, and only keeps the signs of derivatives, which represent qualitative relations between the independent and dependent attributes. Afterwards, we can use standard ma-

chine learning algorithms to induce predictive qualitative models, or venture into exploratory analysis and visualisation techniques.

We will continue the introduction with a formal definition of the problem and an overview of the related work. The following section describes the algorithms for computation of partial derivatives from the data. In the section on experiments we test the algorithms on artificial data sets specifically constructed to explore particular properties of the algorithms. We conclude with discussion of the experimental results and some remarks.

1.1 Problem definition

We define *qualitative partial derivative* of function $f(x_1, \ldots, x_n)$ with respect to attribute x_i as the sign of partial derivative,

$$\frac{\partial_Q f}{\partial_Q x_i} = \mathrm{sgn}\frac{\partial f}{\partial x_i} \tag{1}$$

The qualitative derivative can be increasing ($+$), decreasing ($-$) or steady ($\circ$). We will write the fact that a function is increasing, decreasing or steady with respect to x_i as $f = Q(+x_i)$, $f = Q(-x_i)$ and $f = Q(\circ x_i)$, respectively.

Qualitative models are models which describe how the qualitative behaviour of the function with respect to one attribute depends upon values of other attributes. For instance, function $f(x_1, x_2) = x_1 x_2$ increases with x_1 if x_2 is positive, and decreases with x_1 if x_2 is negative. If x_2 is zero, x_1 has no effect on the value of the function. This model can be written down in form of the following three rules:

$$\begin{aligned}&\text{if } x_2 > 0 \text{ then } f = Q(+x_1)\\ &\text{if } x_2 < 0 \text{ then } f = Q(-x_1)\\ &\text{if } x_2 = 0 \text{ then } f = Q(\circ x_1).\end{aligned}$$

Function arguments can also be discrete, as in the following qualitative relations between the price of a product and its type and size:

$$\begin{aligned}&\text{if ProductType = car}\\ &\qquad\text{then Price} = Q(+\mathrm{ProductSize})\\ &\text{if ProductType = computer}\\ &\qquad\text{then Price} = Q(-\mathrm{ProductSize})\end{aligned}$$

The task of qualitative modelling is to construct such models. In our case, we will induce them from data given as a set of learning examples. Each example is described by values of discrete or continuous attributes and with a continuous outcome. The outcome represents the value of some unknown function. The task is to describe the qualitative behaviour with respect to one or more attributes, conditioned by values of these or other attributes.

The method proposed in this paper solves the problem in two steps. First, we compute qualitative partial derivatives for each example. This translates modelling the function's behaviour into the training of classifiers which predict the qualitative derivative in different parts of attribute space. For instance, the above rules can be acquired by running the CN2 rule learning algorithm on examples labelled by qualitative partial derivatives.

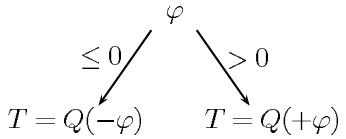
Separate models can be built for each attribute with respect to which we observe the function's behaviour. Alternatively, one can also build a classifier which predicts all qualitative derivatives at once.

$$T = Q(+l)$$

(a) Relation between the period and the length of the rope.

$$T = Q(\circ m)$$

(b) Relation between the period and the mass of the bob.



(c) Relation between the period and the initial amplitude.

Figure 1: Qualitative models describing the relations between the period T and the experimentally controlled variables.

1.2 Introductory example

Consider a set of experiments with a simple pendulum. The task is to learn qualitative relations between the period of the pendulum's first swing (T), the length of the rope, l, the mass of the bob, m, and the initial angle of displacement, φ. A sample of data collected in such an experiment is shown in Table 1 (first four columns).[1]

We can then use Padé to compute the qualitative relations for each measurement: $\partial_Q T/\partial_Q m$, $\partial_Q T/\partial_Q l$, $\partial_Q T/\partial_Q \varphi$, and append them to the original data (Table 1, last three columns). Finally, an algorithm for induction of classification trees is used to construct a qualitative tree for each qualitative relation, where examples are represented by original attributes and the partial derivative (*e.g.* one of the last three columns from Table 1) plays the role of the class. The resulting trees are shown in Fig. 1. Two of them have only a single leaf: the period always increases with the length of the rope ($T = Q(+l)$) and does not depend on the mass of the bob ($T = Q(\circ m)$). The tree describing the relation between T and φ says that for negative angles, T decreases with increasing angle while for positive an-

[1] A part of this experiment was actually performed using a Nao robot.

m	l	φ	T	$\partial_Q T/\partial_Q m$	$\partial_Q T/\partial_Q l$	$\partial_Q T/\partial_Q \varphi$
3.61	0.69	37.23	1.70	○	+	+
5.49	0.71	46.52	1.74	○	+	+
9.19	0.84	-48.91	1.91	○	+	−
7.17	0.33	33.89	1.17	○	+	+
6.81	0.50	65.93	1.51	○	+	+
4.64	0.69	-78.89	1.7	○	+	−

Table 1: A sample of data collected by experimenting with a simple pendulum.

gles, T increases when φ increases. We can reinterpret this as $T = Q(+|\varphi|)$.

1.3 Related work

Mathematical foundations of qualitative reasoning were established by the work of Kalagnanam and Simon [6, 7, 5], but building on the much older work of Samuelson [9] in economics, as well as the work on qualitative stability in ecology [4, 8].

Many algorithms have, in one way or another, tackled the problem of qualitative model induction from observation data. Algorithms QUIN and epQUIN [11, 10, 2] learn qualitative trees similar to those in Figure 1, except for a somewhat different definition of the relation in the leaf. Qualitative relations in QUIN are not based on partial derivatives, as in Padé, but on qualitative constraints. A constraint $z = M^{+-}(x, y)$ would state that for every pair of examples in which x increases and y decreases (or stays the same), the function value z increases. This also implies that the function value depends on no other attributes than x and y. QUIN constructs such trees by computing the qualitative change vectors between all pairs of examples in the data and then recursively splitting the space into regions which share common qualitative properties, such as the one given above. Although Padé combined with a tree learning algorithm can produce a similar tree as QUIN, the two methods are fundamentally different. Besides Padé being merely a preprocessor which can be used with any other machine learning algorithm or visualization technique, the crucial difference is that Padé considers individual examples while QUIN operates on pairs. As one of the consequences, Padé can compute qualitative (or numerical) derivatives for a particular point in the attribute space, while QUIN observes properties of regions of space.

Gerçeker and Say [3] fit polynomials to numerical data and use them to induce qualitative models. LYQUID is designed for modelling dynamic systems, where the data consists of traces sampled in time. We believe that this system could be adapted to also work in static systems.

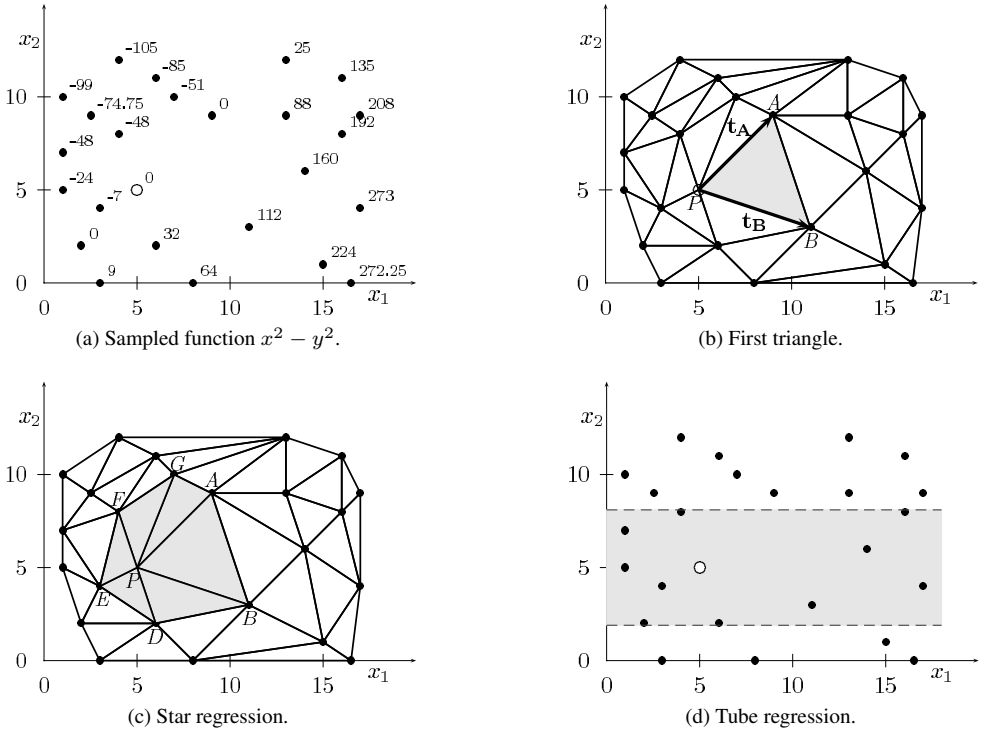
Padé differs from past methods in being, to our knowledge, the only algorithm for computing (partial) derivatives on point cloud data. An important difference between these algorithms and Padé is also that Padé is essentially a preprocessor while other algorithms induce a model. Padé merely augments the learning examples with additional labels, which can later be used by appropriate algorithms for induction of classification or regression models, or for visualization. This results in a number of Padé's advantages. For instance, to our knowledge, most other algorithms for learning qualitative models only handle numerical attributes, except for QDE learners which already take qualitative behaviours as input. One variant of Padé can use discrete attributes while computing the derivative, while with others we can use them later, when machine learning algorithms are applied to Padé's output.

The major contribution of this work, besides the idea of transforming the problem of qualitative modelling to standard induction of classifiers, are methods for computing partial derivatives of an unknown sampled function. In this respect it is related to numerical analysis for estimation of partial derivatives. Numerical analysis methods for computing partial derivatives are only useful for a function which is *known* in the sense that we can compute its value at any values of arguments which the algorithm requires. These methods are not appropriate for learning from data, where the function is sampled only in a limited number of points.

2 Algorithms

Let f be a continuous function of n arguments, $y = f(x_1, x_2, \ldots, x_n)$. The function is sampled in N points; the point's coordinates together with a function value represent a learning example. In machine learning terminology, each example is described by a list of *attributes*, $(a_1, a_2, \ldots a_n)$ and the outcome $f(a_1, a_2, \ldots, a_n)$. The basic task of Padé is to compute a partial derivative at each point (learning example) P in direction x_i.

Fig. 2a shows an example of such data for a function $f(x, y) = x^2 - y^2$. Each point represents a learning example, and the numbers beside the examples give the function values. We will compute the derivative w.r.t. x_1 at $P = (5, 5)$ marked by a hollow symbol. Other points used in computation will be marked as A, B, C and so on. We will treat these points as elements of affine space, and use $\mathbf{t_A} = A - P$, $\mathbf{t_B} = B - P$,... to denote the vectors from P to the corresponding points. We will also extend the definition of f to these vectors, *i.e* $f(\mathbf{t_A}) = f(A - P) := f(A) - f(P)$ and so forth. These linear transformations simplify the computation by setting up a coordinate system



(a) Sampled function $x^2 - y^2$. (b) First triangle.

(c) Star regression. (d) Tube regression.

Figure 2: An artificial, sampled function (a) and an illustration of Padé's methods (b-d).

in which the point P lies in the centre, $\mathbf{t_P} = 0$ and the corresponding function value $f(\mathbf{t_P})$ equals 0.

Formally, the partial derivative with respect to x_i at point P is defined as

$$\frac{\partial f}{\partial x_i}(P) = \lim_{h \to 0} \frac{f(P + h\mathbf{x_i}) - f(P)}{h}, \qquad (2)$$

where $\mathbf{x_i}$ is the i-th base vector.

This definition cannot be used directly on data since it involves infinitesimally small h and, furthermore, since we cannot compute the function value at arbitrary points. According to Taylor's theorem, the function can be treated as approximately linear in small neighbourhoods of P,

$$f(x_1, \ldots x_n) = b_0 + \sum_{i=1}^{n} b_i x_i + \epsilon, \qquad (3)$$

where ϵ represents the remainder in Taylor expansion (the function's non-linearity within the neighbourhood) and also any noise in the data.

The derivative with respect to x_i equals b_i in (3). The task is then to define a suitable neighbourhood and estimate the coefficient b_i accordingly. We will present three different ways for solving this problem. The first method determines the linear function f by simple linear interpolation over the simplex while the other two methods use linear regression.

2.1 First Triangle method

First triangle method models the function's behaviour by dividing the attribute space into simplices (our two-dimensional illustrations show them as triangles) by using the standard Delaunay triangulation [1] as shown in Fig. 2b. Let us assume that there is no noise in the data and that the sample is sufficiently dense so the function is approximately linear within each triangle.

Since the number of unknown coefficients b_i in (3) equals the number of vertices of the simplex, the coefficients can be found analytically by setting $\epsilon = 0$. Performing the calculation in vector space instead of in the affine space of points also eliminates the free term b_0 and point P.

Let $\mathbf{t_1}, \ldots, \mathbf{t_n}$ be the vectors from P to the vertices of the simplex which lies in direction x_i. We look for $b_1, \ldots, b_n$ which satisfy

$$[b_1 \ldots b_n][\mathbf{t_1} \ldots \mathbf{t_n}] = [f(\mathbf{t_1}) \ldots f(\mathbf{t_n})] \qquad (4)$$

(note that $\mathbf{t_i}$ are n-dimensional vectors) and thus

$$[b_1 \ldots b_n] = [f(\mathbf{t_1}) \ldots f(\mathbf{t_n})][\mathbf{t_1} \ldots \mathbf{t_n}]^{-1} \qquad (5)$$

For our two-dimensional example (Fig. 2b), we interpolate over the triangle PAB and compute the coefficients as

$$[b_1, b_2] = [f(\mathbf{t_A}), f(\mathbf{t_B})][\mathbf{t_A}, \mathbf{t_B}]^{-1},$$

which equals

$$[b_1, b_2] = [f(A) - f(P), f(B) - f(P)][A - P, B - P]^{-1}.$$

2.2 Star Regression

Star Regression is based on similar assumptions as the First triangle method, but improves its noise resistance by assuming the function's linearity across the entire star (the set of simplices surrounding a point) around the point P instead of just across a single simplex.

We can no longer use interpolation as in the First triangle, as it would result in a system with more equations than unknowns and would usually have no solution. We therefore allow non-zero error terms ϵ and translate the problem into computation of univariate linear regression over the vertices in the star.

If $\mathbf{t_1},\ldots,\mathbf{t_n}$ are the vectors from P to the vertices of the star, we compute b_i, which equals the derivative, as

$$b_i = \frac{\sum_j t_{ji} f(\mathbf{t_j})}{\sum_j t_{ji}^2}, \tag{6}$$

where t_{ji} represents the i-th component of the vector corresponding to the j-th point in the star, $\mathbf{t_j}$.

In our illustration (Fig. 2c), we compute the univariate linear regression over examples A, B, C, D, E and F, and use the first coefficient as derivative.

2.3 Tube Regression

Tube Regression adds even more noise resilience. Instead of triangulating, it considers a certain number of examples in a (hyper)tube passing through point P in direction parallel to the axis of differentiation (Fig. 2d; the tube is represented by the shaded area). We now assume that the function is approximately linear within short parts of the tube and again estimate the derivative from the corresponding coefficient computed by the univariate regression, this time over the examples in the tube.

Since the tube can also contain examples that lie quite far away from P, we weight the examples by their distances from P along the tube (that is, ignoring all dimensions but x_i). The weight of the j-th example in the tube equals

$$w_j = e^{-t_{ji}^2/\sigma^2}, \tag{7}$$

where t_{ji} is the i-th component of vector $\mathbf{t_j}$ (that is, the distance between P and the j-th example in direction parallel to the axis of differentiation, x_i). Parameter σ is chosen so that the farthest example in the tube has a user-set negligible weight. As a rule of thumb, we use tubes with 30 examples, with the farthest (*e.g.* the right-most point in the tube in Fig. 2d) having a weight of $w_{30} = 0.001$.

We then use the standard weighted univariate linear regression to compute the coefficient of the linear term b_i,

$$b_i = \frac{\sum_j w_j t_{ji} f(\mathbf{t_j})}{\sum_j w_j t_{ji}^2}. \tag{8}$$

The Tube regression is computed from a larger set of examples, so we can use the t-test to estimate the significance of the derivative. Significance together with the sign of b_i can be used to define qualitative derivatives in the following way: if the significance is above the user-specified threshold (*e.g.* $p \leq 0.7$), the qualitative derivative equals the sign of b_i; if significance is below the threshold we define the qualitative derivative to be *steady*, disregarding the sign of b_i.

2.4 Time complexity

Let N be the number of examples and n the number attributes (function arguments, dimensions).

First triangle and *Star regression* methods are based on Delaunay triangulation. The time complexity of its computation is difficult to assess without making any strong assumptions about the data. The state of the art qhull library needs, roughly, $O(2^n N \log N)$ to compute the triangulation (a detailed analysis can be found in [1]).

For each data point, the First triangle algorithm needs to find the triangle lying in the desired direction, which requires computing the determinant of a n-dimensional matrix for every triangle in the star. The time complexity is $O(Nn^3t)$, where t is the maximal number of triangles in any star. The value of t is again difficult to estimate, but it usually rises exponentially with the number of dimensions, which makes the time complexity $O(Nn^3 2^n)$. The total time complexity, including the triangulation, is thus $O(N2^n(\log N + n^3))$

Star regression computes univariate regression at every data point and has a time complexity of $O(Ns)$, where s is the number of points in the star. As s generally rises exponentially with the number of dimensions, the theoretical time complexity of this part of Star regression is $O(N2^n)$. The total complexity is dominated by that of triangulation and thus equals $O(2^n N \log N)$.

Tube regression finds the nearest neighbours of each of N examples, which takes $O(nN^2)$, followed by linear regression over the k examples in the tube. The total time complexity is $O(nN^2 + k) \approx O(nN^2)$.

Since these theoretical time complexities do not offer much insight into the algorithms' actual running times, we conducted a set of experiments with different number of examples and attributes. The goal function was random since it does not affect the time complexity. All experiments were run on a 2 GHz laptop with 2 GB of RAM. Results (Table 2) indicate that the time complexity of triangulation-based methods is indeed exponential in number of attributes and log linear in number of examples, while the Tube regression is linear in number of attributes and quadratic in number of examples. The exponential time complexity prevents the use of triangulation-based methods with more than four attributes.

3 Experiments

We evaluated Padé on a set of artificial data sets to observe the correctness of derivatives, its scalability with respect to the number of attributes, and its treatment of noise and of

	1000			2000			5000			10000		
	FT	SR	TR	FT	SR	TR	FT	SR	TR	FT	SR	TR
2	3	1	10	5	1	41	14	6	259	42	29	1159
4	242	5	22	449	19	86	1082	114	532	2280	473	2444
6	33049	1387	32	–	3908	134	–	–	857	–	–	3924
8	–	–	45	–	–	176	–	–	1163	–	–	5143
10	–	–	60	–	–	232	–	–	1627	–	–	6694

Table 2: Running times (in seconds) of First triangle (FT), Star regression (SR) and Tube regression (TR) methods for calculation of derivatives w.r.t. each attribute on data sets with 1000, 2000, 5000 and 10000 examples and 2, 4, 6, 8, 10 attributes. Symbol – denotes that the program ran out of memory.

discrete attributes. The accuracy is measured by comparing the predicted qualitative behaviour with the analytically derived true relation.

3.1 Accuracy

We observed the accuracy of Padé on a few mathematical functions. We estimated partial derivatives using Padé and compared them with the analytically obtained correct answers, except for the functions in Fig. 3d and Fig. 3e for which we computed numerical approximations of partial derivatives by Mathematica [14]. Note that this procedure does not require cross-validation or a similar form of data sampling since the known ground truth (the correct derivatives) is not used in the induction process.

Functions $f(x, y) = x^2 - y^2$ and $f(x, y) = xy$, with x and y sampled from $[-10, 10]$ were used as simple examples of functions which are continuous and differentiable in the whole interval. The heavily oscillating $f(x, y) = \sin x \sin y$, with x and y from $[-7, 7]$ represents a function whose qualitative behaviour changes frequently, so the partial derivatives are more difficult to compute and model. Functions $Im(\arcsin(x + iy)^4)$ and $Im(\text{arctanh}(x + iy)^3)$ in $[-2, 2]\times[-2, 2]$ are two examples of discontinuous functions. All functions are visualized in Fig. 3.

We computed the derivatives and trained the classifiers on ten random samples of 1000 points. Average proportions of correctly calculated qualitative derivatives are shown in Table 3. First triangle and Tube regression perform equally well, except for the Tube regression's failure on $f(x, y) = \sin x \sin y$. A visual exploration of predicted derivatives using a scatter plot clearly shows that this is due to the tube being too long and thus covering multiple periods of the function. Star regression's performance lags behind those of the other two algorithms.

To estimate the dependence of classification accuracy on data set size we conducted these same experiments on samples of 100 to 2000 data points. We found out that learning curves tend to flatten out at around 500 examples (Fig. 4). The general order of methods w.r.t their accuracy remains the same for all sample sizes, except for the last two functions, which are discontinuous and where the First Triangle method seems to suffer the least at very small samples.

3.2 Scalability to high number of dimensions

We checked the scalability of Padé to high dimensional spaces with an experiment with function $x^2 - y^2$, in which we added 98 attributes with random values from $[-10, 10]$ to the data. We use the Tube regression to calculate the derivatives since First triangle and Star regression cannot handle such high dimensional data due to their use of triangulation. We analysed the results by inducing classification trees with the computed qualitative derivatives as classes. Trees for derivatives by x and y agree well with the correct results (Fig. 5).

3.3 Robustness to noise

We sampled the function $f(x, y) = x^2 - y^2$ in 1000 points with x and y from $[-10, 10]$, and introduced uniform random noise of up to ± 20 to the function value. Since the First triangle and Star regression methods suppose no or little noise, we again tested only the Tube regression.

Induced models (Fig. 6) are correct and the split thresholds are surprisingly accurate given the huge relative amount of noise at around $x = 0$ and $y = 0$.

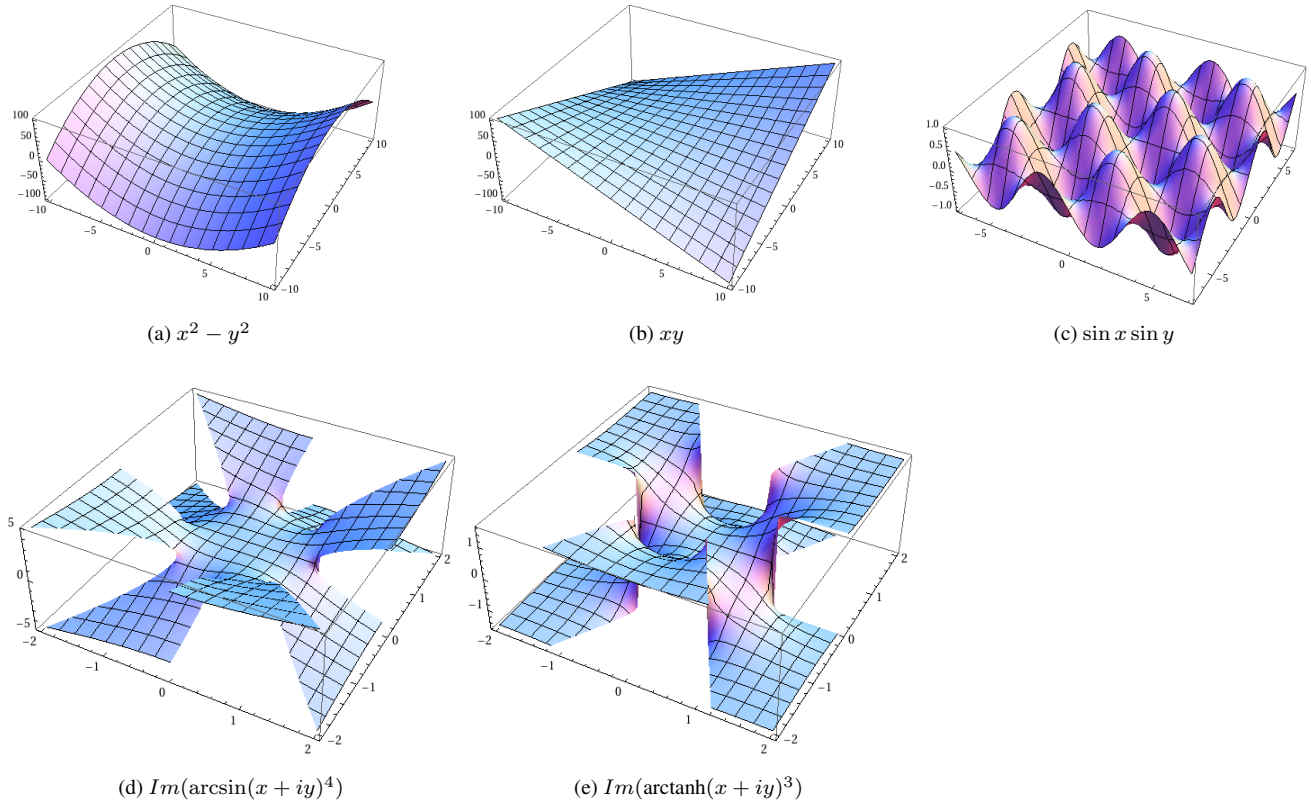
3.4 Handling of discrete attributes

We explored Padé's handling of discrete attributes on a function defined as

$$f(x, s) = \begin{cases} x/10 & ; s = 1 \\ 10x & ; s = 0 \end{cases}$$

Besides the continuous attribute x and boolean attribute s, the data set also included an attribute r with random values and no influence on f. Variables x and r were from the same definition range, $[-10, 10]$. The function was sampled in 400 points.

Tube Regression, whose results we used to construct a classification tree, found the correct solution (Fig. 7). Other methods failed to recognize the role of s, which they were given as a continuous attribute. Using dummy variables, like in statistical regression methods, does not work for triangulation-based Padé's methods.



(a) $x^2 - y^2$ (b) xy (c) $\sin x \sin y$

(d) $Im(\arcsin(x+iy)^4)$ (e) $Im(\text{arctanh}(x+iy)^3)$

Figure 3: Functions used in experiments.

$f(x,y)$	First Triangle		Star Regression		Tube Regression	
	$\partial_Q f/\partial_Q x$	$\partial_Q f/\partial_Q y$	$\partial_Q f/\partial_Q x$	$\partial_Q f/\partial_Q y$	$\partial_Q f/\partial_Q x$	$\partial_Q f/\partial_Q y$
$x^2 - y^2$	98%	98%	98%	97%	95%	95%
xy	99%	99%	92%	92%	99%	99%
$\sin x \sin y$	89%	89%	73%	72%	53%	53%
$Im(\arcsin(x+iy)^4)$	93%	93%	87%	86%	93%	93%
$Im(\text{arctanh}(x+iy)^3)$	91%	93%	76%	79%	86%	90%

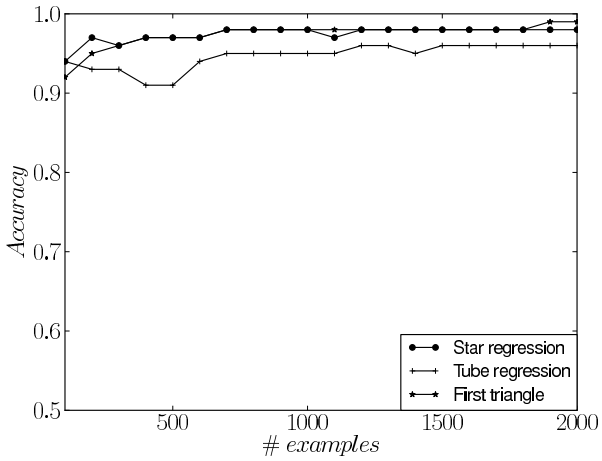
Table 3: The comparison of accuracies of Padé's methods on artificial datasets.

4 Conclusion

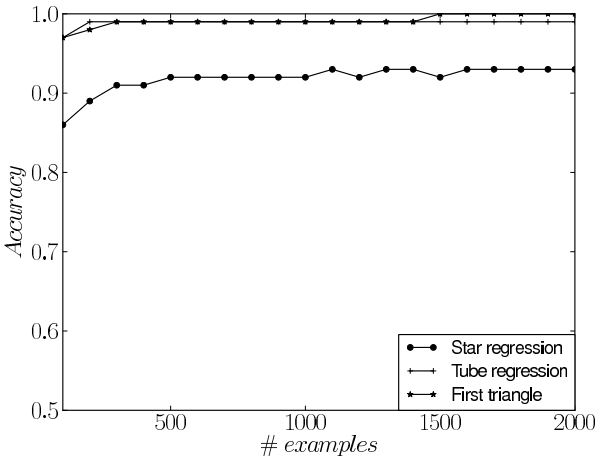
We described a new approach to induction of qualitative models whose advantage over (rare) existing similar algorithms is that it translates the problem into a standard supervised learning problem, which is one of the most researched fields in machine learning. The proposed translation requires computation of qualitative partial derivatives, which we defined simply as signs of ordinary partial derivatives. The biggest problem – and with that the core of this paper – is computation of partial derivatives of the function which is being modelled. Standard methods from numerical analysis cannot be applied here since they require a known function whereas in our case the function value is known only in a finite number of sampled examples.

We proposed three methods for this task. Two are based on triangulation and suppose either no noise or a small amount of noise. Besides this, the two methods are not likely to be useful in real-world scenarios which often contain more attributes than triangulation can handle. Experiments with the time complexity of the methods clearly show that the triangulation-based methods are unable to handle more than 6 attributes. Nevertheless, in absence of noise these two methods provide very good accuracy for functions with complex qualitative behaviour and low number of arguments, such as $f(x,y) = \sin x \sin y$. Tube regression on the other hand offers robustness to noise, scales well to high dimensional spaces and can also handle discrete function arguments with proper definition of metrics.

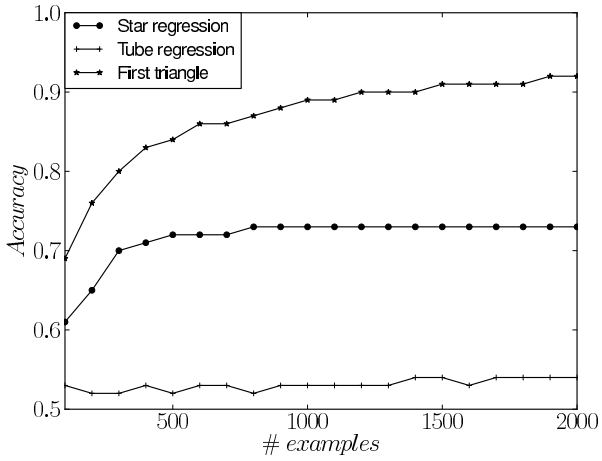
In general, the triangulation-based methods may be mostly of theoretical interest, while the Tube regression has all the features required of a practically useful machine



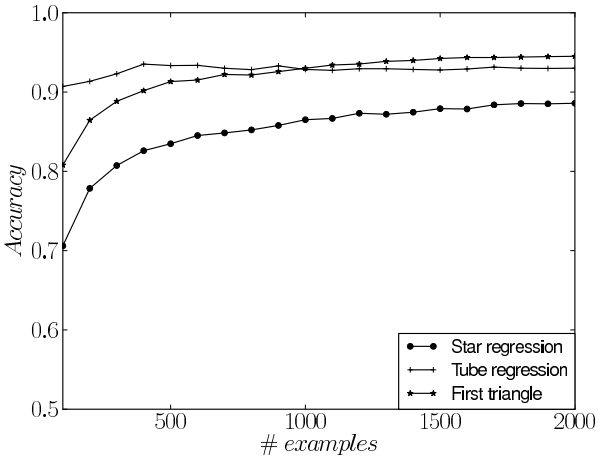
(a) $f(x, y) = x^2 - y^2$



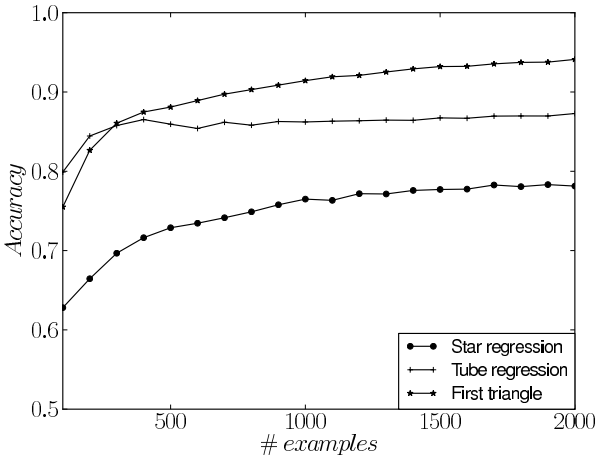
(b) $f(x, y) = xy$



(c) $f(x, y) = \sin x \sin y$

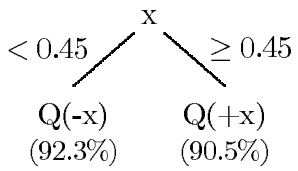


(d) $f(x, y) = Im(\arcsin(x + iy)^4)$

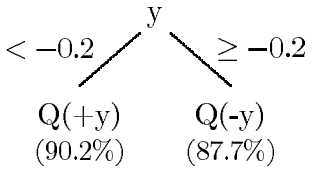


(e) $Im(\text{arctanh}(x + iy)^3)$

Figure 4: Accuracy of $\partial_Q f/\partial_Q x$ over different sizes of data sets. Results for $\partial_Q f/\partial_Q y$ are similar.

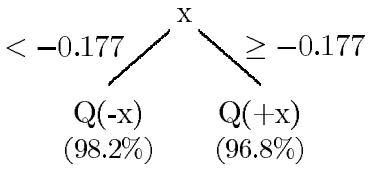


(a) Derivative w.r.t. x.

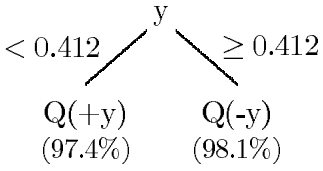


(b) Derivative w.r.t. to y.

Figure 5: Qualitative models of function $x^2 - y^2$ with 98 additional random attributes.



(a) Derivative w.r.t. x.



(b) Derivative w.r.t. to y.

Figure 6: Qualitative models of function $x^2 - y^2$ with added random uniform noise.

learning method.

We have put the methods at a practical test within European project XPERO (IST-29427). Our goal was to provide a robot with an algorithm for autonomous learning. We found qualitative models most suitable for this task. For example, a particular case was to discover the relation between the area of the ball in the image from robot's camera, and the robot's angle and distance from the ball [13]. The robot learnt that the area of the ball is increasing with decreasing distance and decreasing with increasing angle (the robot turning away from the ball, so it gradually vanishes from the robot's field of view).

Since the field of learning qualitative models from data is rather unexplored, the paper opens more new interesting questions than it answers. Pioneers of qualitative modelling who constructed the models manually were able to describe real phenomena using simpler models, not unlike the classification trees and rules presented here. Is this generally the case? Do simple learning algorithms like tree induction, suffice, or will actual problems require more sophisticated algorithm, such as, for instance, support vector machines?

This paper follows the mathematical definition of partial derivative which is essentially univariate. Partial derivatives are linear and do not interact: the effect of changing two quantities at the same time equals the sum of effects of changing each of them separately. The exception to this rule are certain kinds of singularities. Does this happen in practice, especially in qualitative descriptions of problems? Can it happen, for instance, that two economic measures used separately decrease the inflation while using both together would increase it? Is treating each attribute separately indeed appropriate? We leave these questions open for further research.

Acknowledgements

This work was supported by the Slovenian research agency ARRS (J2-2194, P2-0209) and by the European project XMEDIA under EC grant number IST-FP6-026978.

References

[1] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[2] Ivan Bratko and Dorian Šuc. Learning qualitative models. *AI Magazine*, 24(4):107–119, 2003.

[3] R. K. Gerçeker and A. Say. Using polynomial approximations to discover qualitative models. In *Proc. of the 20th International Workshop on Qualitative Reasoning*, Hanover, New Hampshire, 2006.

[4] Clark Jeffries. Qualitative stability and digraphs in model ecosystems. *Ecology*, 55(6):1415–1419, 1974.

[5] Jayant Kalagnanam and Herbert A. Simon. Directions for qualitative reasoning. *Computational Intelligence*, 8(2):308–315, 1992.

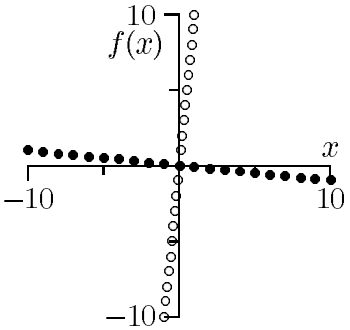
[6] Jayant Kalagnanam, Herbert A. Simon, and Yumi Iwasaki. The mathematical bases for qualitative reasoning. *IEEE Intelligent Systems*, 6(2):11–19, 1991.

[7] Jayant Ramarao Kalagnanam. *Qualitative analysis of system behaviour*. PhD thesis, Pittsburgh, PA, USA, 1992.

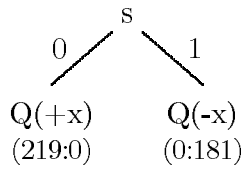
[8] Robert M. May. Qualitative stability in model ecosystems. *Ecology*, 54(3):638–641, 1973.

[9] Paul A. Samuelson. *Foundations of Economic Analysis*. Harvard University Press; Enlarged edition, 1983.

[10] Dorian Šuc. *Machine Reconstruction of Human Control Strategies*, volume 99 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands, 2003.



(a) Modelled function; filled and hollow circles represent examples with $s = 1$ and $s = 0$, respectively.



(b) Qualitative tree based on Tube Regression.

Figure 7: The function used in the experiment with discrete attributes and the corresponding qualitative model.

[11] Dorian Šuc and Ivan Bratko. Induction of qualitative trees. In L. De Raedt and P. Flach, editors, *Proceedings of the 12th European Conference on Machine Learning*, pages 442–453. Springer, 2001. Freiburg, Germany.

[12] Dorian Šuc, Daniel Vladušič, and Ivan Bratko. Qualitatively faithful quantitative prediction. *Artificial Intelligence*, 158(2):189–214, 2004.

[13] Jure Žabkar, Ivan Bratko, and Janez Demšar. Learning qualitative models through partial derivatives by Padé. In *Proceedings of the 21th International Workshop on Qualitative Reasoning*, Aberystwyth, U.K., 2007.

[14] Wolfram Research, Inc. Mathematica, version 7.0. Wolfram Research, Champaign, Illinois, 2008.