Learning Qualitative Models through Partial Derivatives by Padé

Jure Žabkar and **Ivan Bratko** and **Janez Demšar**
AI Lab, Faculty of Computer and Information Science,
University of Ljubljana, SI-1000 Ljubljana, Slovenia

Abstract

Padé is a new method for learning qualitative models from observation data by computing partial derivatives from the data. Padé estimates partial derivatives of a target function from the learning data by splitting the attribute space into triangles or stars from Delaunay triangulation, or into tubes, and computing the linear interpolation or regression within these regions. Generalization is then accomplished by any attribute-value learning method. The methods for estimating partial derivatives differ regarding their resistance to noise, ability to handle noisy and missing values, computation speed and other properties. The experiments show these methods to be quite accurate, fast and robust. Being well integrated into our general machine learning and data mining suite Orange, Padé should also prove useful in practice.

Introduction

One of the goals of attribute-based machine learning is to explain the roles of individual attributes. An efficient way of achieving this in regression problems is to observe the change of function value corresponding to changes in individual attribute values. In mathematics, this is called partial derivative and has been — from its invention by Newton and Leibniz on — a most fundamental tool for describing relations in physics. In this paper we develop a machine learning method that computes partial derivatives and combines well established principles from mathematics and physics with the robustness and flexibility of typical machine learning algorithms.

In qualitative modeling, the task is often limited to only predicting the sign of the derivative and not its magnitude. In this paper we propose a set of methods with a common name Padé (an acronym for "partial derivative", and the name of a famous French mathematician). The methods assess qualitative or quantitative partial derivatives for points in the attribute space. We can then use machine learning algorithms to induce a predictive model, or venture into exploratory analysis and manually discover relations in the data.

Our goal was not only to design a fast, robust and conceptually clean algorithm with a small number of parameters, but also make it well integrated into our general ML platform Orange (Zupan, Leban, & Demšar 2004) with its substantial arsenal of machine learning and data mining techniques.

Learning method

We assume the following learning problem: the input data is a set of variable-value vectors, each consisting of attribute values and a class-value. The attributes normally correspond to independent variables in our problem space, and the class corresponds to a dependent variable. The task is to find a qualitative model that explains this data. The model may be a formal structure, such as a set of qualitative rules or a tree, or visual, for instance with a scatter plot. The model obtained from the data should enable predictions of the dependent variable value when given the values of the variables.

The class variable is continuous, and the attributes may be either continuous or discrete. In the context of learning models of physical systems, typically at least some of the attributes are real-valued. In our case, a qualitative model will consist of a set of qualitative proportionality constraints that will appear in if-then rules or decision trees. For example, let y be a function of x: $y = x^2$. The learning data would consist of a sample of pairs of values (x, y) where x is the attribute (independent variable) and y is the class (dependent variable). A correct qualitative model induced from this data would be:

$$\text{if } x > 0 \text{ then } y = Q(+x)$$
$$\text{if } x < 0 \text{ then } y = Q(-x)$$

The constraint $y = Q(+x)$ is read as y is qualitatively proportional to x. Roughly, this means that y increases with x. More precisely, in Padé this means

$$\frac{\partial y}{\partial x} > 0.$$

The notation $y = Q(-x)$ means that y is inversely qualitatively proportional to x (i.e. the partial derivative of y w.r.t. x is negative).

As another example that involves a discrete variable, we may state the qualitative relations between the price of a product, and the product's type and size:

$$\text{if } ProductType = car \text{ then } Price = Q(+ProductSize)$$
$$\text{if } ProductType = computer \text{ then } Price = Q(-ProductSize)$$

We will also be using an abbreviated notation when referring to several qualitative proportionalities. For example,

two constraints $z = Q(+x)$ and $z = Q(-y)$ will be abbreviated to $z = Q(+x, -y)$.

Using Padé for learning of qualitative models of this kind consists of three stages:

1. For all the given data points, use Padé to assess numerically the partial derivative of the class variable w.r.t. a chosen continuous attribute.
2. Perform a qualitative abstraction: Convert the computed numerical approximations of partial derivatives at all the data points into their signs. These signs are then used as discrete class values in the next step.
3. Use any attribute-value learning method (such as if-then rule learning, or decision tree learning) to produce a classifier that maps the points in the attribute space into qualitative proportionality constraints. This classifier is our qualitative model induced from the data.

Padé is not a single algorithm but a suite of methods performing this task: approximation of partial derivatives of a sampled unknown function f. The input for all the methods is a set of examples described by a list of attribute values and the value of a continuous dependent variable. An example of a function with two attributes is depicted in Fig. 1(a): each point represents a learning example and is assigned a continuous value. Our task is to compute a partial derivative at each given point. In our illustrations, we shall show the computation at point (5, 5), which is marked with a hollow symbol.

In the following paragraphs, we describe four novel methods of numerically estimating partial derivatives, which we use in Padé.

First Triangle method was the initial venture point of development of Padé. It models the function's behavior by dividing the attribute space into simplices (we shall refer to them as "triangles") by using the standard Delaunay triangulation as shown in Fig 1(b). If the samples are sufficiently dense the function's behavior within triangles is approximately linear.

Being able to compute the value of function in point $f(P + dx)$ with a simple interpolation between points P, A and B, First triangle method can apply the textbook definition of the partial derivative:

$$\frac{\partial f}{\partial x} = \frac{f(P + dx) - f(P)}{dx}.$$

Star Regression is based on the First triangle method, but improves its noise resistance by assuming the function's linearity across the entire star (a topological term for the set of triangles surrounding a point) instead of just a single triangle. It finds the value of the partial derivative that minimizes the square error, which translates into computing the univariate linear regression across the points of the star.

In the case shown in Fig. 1(c), the derivative would be computed as the coefficient of the linear regression on points A, B, C, D, E and F.

Triangles' Path method copes with more noise by smoothing the function more. To keep the computation focused, we do not simply widen the star but instead follow the triangles in the direction in which we compute the derivative (1(d)). The partial derivative is then again computed by minimizing the square error for the points lying on that path. In each shaded triangle, we choose an arbitrary point and assign it a function value by linear interpolation between triangle's vertices.

This method was actually never implemented as described here, but only as a simplified version of another, more complicated algorithm Qing, which includes many other improvements that will be published elsewhere.

Tube Regression is an approximation of the Triangles' Path method. It avoids computing the triangulation altogether, but instead considers a certain number of examples nearest to the axis in the direction in which we compute the derivative. These examples lie in a (hyper)tube which approximates (or, better, *mimics*) the Triangles' path (Fig. 1(e)).

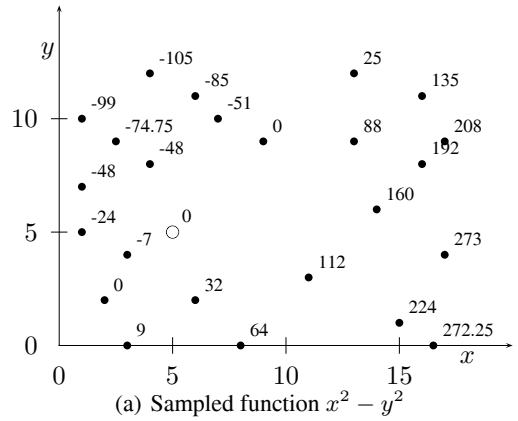
The tube can also contain points that lie quite far from the point P. To observe the local behavior of the function, Tube regression weights examples by their distances from P along the tube (that is, ignoring all dimensions but x). The method is thus similar to computing 1-dimensional LWR within the tube and taking the coefficients as partial derivatives.

All described methods are implemented as preprocessors, which get a sampled function, described by values of arguments and the function value, and return the corresponding numerical or qualitative partial derivative at each point. These derivative data can then be modeled with regression or classification trees (in the latter case one can model derivatives for each attribute separately or all attributes together) or by any other appropriate machine learning algorithm. It is usually even more interesting and useful to observe the data by visualizing it in scatter plots or other visualizations.

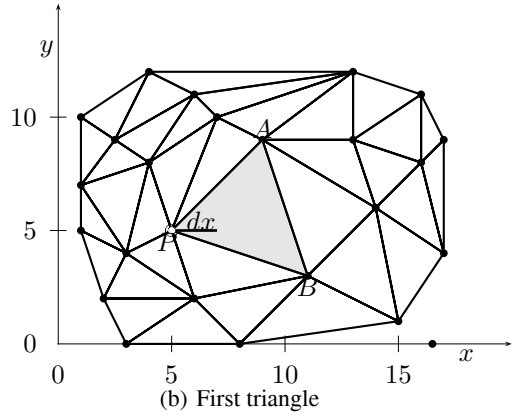
Experiments

The described methods were implemented within data mining and machine learning framework Orange (Zupan, Leban, & Demšar 2004), so they can be used with its huge arsenal of machine learning and visualization techniques.

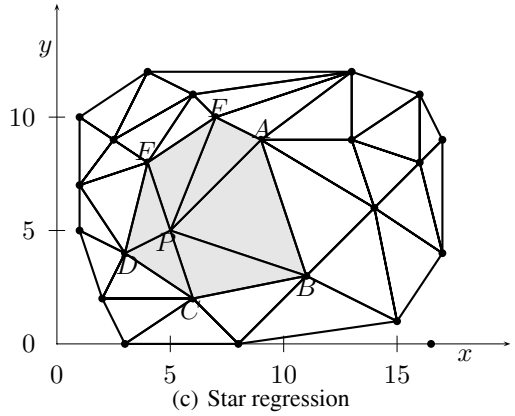
We will illustrate the interesting qualities and shortcomings of the algorithms with several experiments. We commence with inverted pendulum: we use a simple visualization that reveals the qualitative behavior of the function and also helps choosing a suitable modeling algorithm. We then show a simple artificial domain where the correct model depends on using a discrete attribute. We continue with another artificial example, $\sin(x)\sin(y)$ over a few periods, where the visualization turns out to be the only sensible "model". We then investigate Padé's ability to cope with noise, and conclude with an example with data from a 6th Framework European research project XPERO.



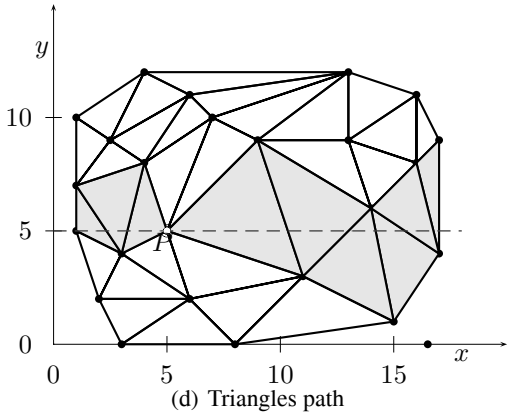
(a) Sampled function $x^2 - y^2$



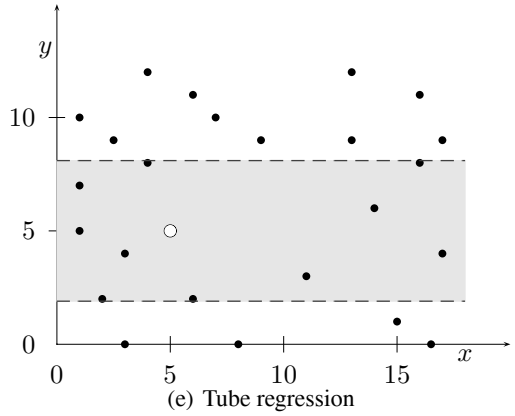
(b) First triangle



(c) Star regression



(d) Triangles path



(e) Tube regression

Figure 1: Illustration of Padé's methods

Inverted pendulum

The inverted pendulum is a well known dynamic domain often used in evaluation of algorithms for learning and control. The physical model is determined by equations for $\ddot{x}$ and $\ddot{\varphi}$.

$$\ddot{x} = \frac{4F + 2lm\dot{\varphi}^2 \sin\varphi - 1.5mg \sin 2\varphi}{4M + 4m - 3m\cos^2\varphi}$$

$$\ddot{\varphi} = \frac{(M+m)g\sin\varphi - F\cos\varphi - \frac{1}{2}ml\dot{\varphi}^2 \sin\varphi\cos\varphi}{\frac{1}{6}(4M + 4m - 3m\cos^2\varphi)l}$$

The variables x and φ are the horizontal position of the cart and the angle of the pole w.r.t. vertical axis. F is the horizontal force applied to the cart. The parameters M, m and l are the mass of the cart, the mass of the pole and the length of the pole. We here demonstrate our algorithm on the problem of modeling $\ddot{x}$, which is more difficult than the modeling of $\ddot{\varphi}$. We generated a set of 1000 examples by random sampling with parameters set to: $F = 0$ (free movement), $M = 1$ (mass of cart), $m = 0.1$ (mass of pole), $l = 1$ (length of pole). Independent variables are $\varphi \in [-\pi/2, \pi/2]$ and $\dot{\varphi} \in [-10, 10]$.

We approached the problem with the First triangle method because the data consists of only continuous attributes without any noisy or missing values. Its results can be nicely visualized with a scatter plot, which shows that $\ddot{x}$ is negative for smaller values of φ and $\dot{\varphi}$, and positive for the larger (Fig. 2a). The boundary between the two areas suggests the unsuitability of classification trees for modeling the domain. We instead used Orange's implementation of naive Bayesian classifier which uses LOESS for estimating the conditional probabilities for continuous attributes. Its visualization with the nomogram (Fig. 2b) shows that it can fit the boundary perfectly (see (Jakulin *et al.* 2005) for a detailed explanation of nomograms).

We obtained similar results with other Padé's methods, though they somewhat distorted the ellipse.

Discrete Attributes

We checked the Tube Regression's handling of discrete attributes with a function nastily defined as

$$\text{IF } s = 1 \text{ THEN } f = -x/10 \text{ ELSE } f = 10x.$$

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 using C4.5 (Quinlan 1993) included in Orange, found the correct solution (Fig. 3). We also tried other Padé's methods, which, as expected, mostly failed to recognize the role of s (which they were given as a continuous attribute). This confirms that replacing discrete attributes with dummy variables, like in statistical regression methods, will not work with triangulation-based Padé's methods.

Visualization

There are domains in which most machine learning algorithms fail to produce any meaningful results without a strong help from the expert. In such cases, using a good visualization is a much better choice than blindly inducing a model. Padé works with many visualization algorithms, from a simple distribution graph or scatter plot to state-of-the-art methods of intelligent visualization (Leban *et al.* 2006).

To illustrate such a domain, we generated a data set of 10000 sampled points for function $\sin(x)\sin(y)$, $x, y \in [-3\pi, 3\pi]$ (Fig. 4(a)). Such periodic functions of two variables are quite common in the real world. We computed derivatives by x; results for y are analogous.

The obvious candidate for this data is the First triangle method: there is no noise and all attributes are continuous. Knowing the complexity of the modeled function in advance, we can expect the noise reducing methods to almost certainly "oversimplify" the data.

First triangle (Fig. 4(b)) performed perfectly. The edges are perfectly sharp, which is due to the very high density of the samples. We checked that the algorithm still performs very well with 500-1000 samples, except for the edges which then evidently follow the individual sample points.

Star regression (Fig. 4(c)) exhibits some smearing at the corners, yet its results are still excellent and useful. Our suspicions that Tube regression (Fig. 4(d)) is unable to model this data were proved correct: it merged the left-most and the right-most two columns, and performed miserably in between.

Figures 4(e) and 4(f) visualize numerical derivatives.

Noise

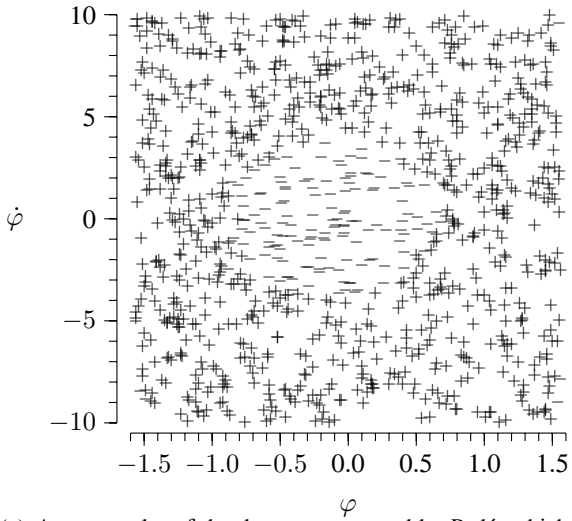
As an example of a very noisy function, we sampled the function $f(x, y) = x^2 - y^2$ on interval $[-100, 100] \times [-100, 100]$ to which we added uniform random noise of up to ± 2000. The data set consisted of 1000 random samples. Fig. 5 shows the intersection of the "noised" surface $x^2 - y^2$ with the plane $y = 0$ to illustrate the magnitude of the added noise around the point where the qualitative behavior of the function changes.

With such extreme noise, the method of choice is Tube regression. The assessed qualitative derivatives were used to induce a decision tree (Fig. 6). The induced models are correct and the split thresholds are quite accurate given the huge relative noise at around $x = 0$ and $y = 0$.

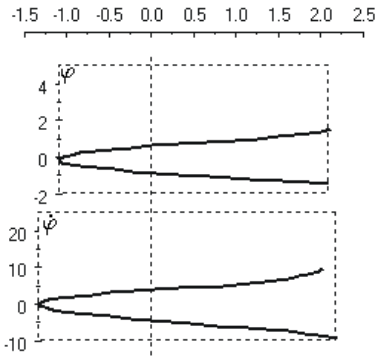
XPERO robot

For the final example, we used Padé on data from the ongoing European project XPERO (IST-29427). A simulated robot with a camera observes a ball. The task in this particular case was to discover the relation between the area of the ball in the picture, and the robot's angle and distance from the ball.

Figure 7 shows the corresponding trees. Padé performed perfectly regarding qualitative proportionality between distance and area: whenever the ball is visible (that is, the angle is approximately between -28.9 and +27.8 degrees), the area



(a) A scatter plot of the data set generated by Padé, which visualizes qualitative behaviour of $\ddot{x}$ with regards to φ.



(b) A naive Bayesian nomogram from Padé's data that models the qualitative behavior depicted in the scatter plot on the left.

Figure 2: A visualization and a qualitative model for inverted pendulum.

decreases with distance. Otherwise, is does not depend upon the distance (denoted by Q()).

The tree for the correspondence between the angle and the area correctly discovered – although with rather loose threshold values – that the area increases when the robot is turning towards the ball (that is, when the negative angle increases and when the positive angle decreases). When the entire ball is visible, the angle plays no role; the prevalent class with a slim majority is Q(-angle), yet the tree continues to split further and further. These further splits make no sense and, admittedly, the tree in Figure 7 was manually pruned at the middle leaf. For the angles in between, the area thus sometimes increase and sometimes decrease with the angle, which is the artifact of particular learning traces.

Discussion

All Padé's methods are fairly easy to understand and implement. They, however, differ in many important aspects.

Noise Handling

First triangle's beauty is in its pure use of concepts from topology and analysis. Its results on noiseless data are as good as the density of samples permit, while with increasing the noise level they soon degrade to useless. Noise canceling algorithms from topology are being added as a part of the Qing algorithm mentioned earlier.

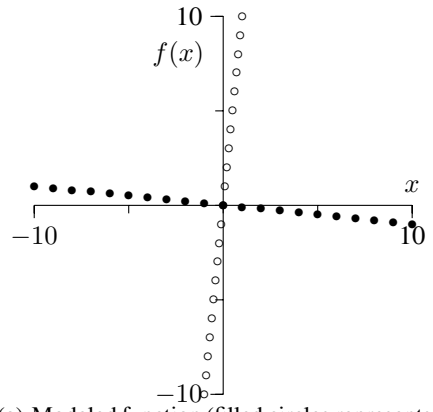
Tube regression, on the other side of the spectrum, is highly noise resistant, which will, as usual, also make it smear fine details in noiseless data. The actual degree of smoothing is in principle regulated with two arguments. The width of the tube should balance between having enough examples for a reliable estimation of the coefficient on one side and not observing the examples where the values of other attributes could significantly affect the function value (too much) on the other. However, if the tube is symmetrically covered by the examples (this is probably true except on the boundaries of the covered attribute space) and if the function which we model is negatively symmetrical with respect to other attributes' values in the part of the space covered by the tube,[1] impacts of other attributes can be expected to cancel out. Wide tubes therefore should not (and empirically do not) cause too much of a problem.

There is a similar balancing along dimension x: if the kernel function for the weight is too wide, the derivative will not be local enough, while a narrow kernel will not be resistant to noise. This dilemma is the same as in LWR, with the only difference that while LWR computes a function value, we here observe the regression coefficient.

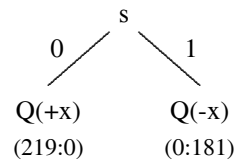
We experimentally observed that the method's parameters do not have considerable impact on the results and fixed the width of the tube to 30 points. Examples are weighted using a Gaussian kernel fitted so that the point farthest along the tube has a negligible coefficient of 0.001. The method is thus effectively without user-definable parameters.

Star Regression's resistance to noise is in between those of the First Triangle and the Tube Regression. We would also expect the Triangles' Path to be close to that.

[1] Formally, $f(\mathbf{x}+\mathbf{y})-f(\mathbf{x}) \approx f(\mathbf{x})-f(\mathbf{x}-\mathbf{y})$, where $\mathbf{x}$ is a point on the axis and $\mathbf{y}$ is a vector perpendicular to the axis and smaller than the tube's diameter. Linear functions, for instance, have this property, and most other functions we model are also locally linear enough.

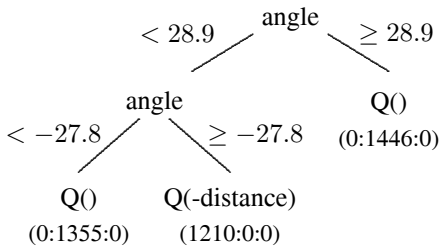


(a) Modeled function (filled circles represents points with $s = 1$, hollow circles represent points where $s = 0$)

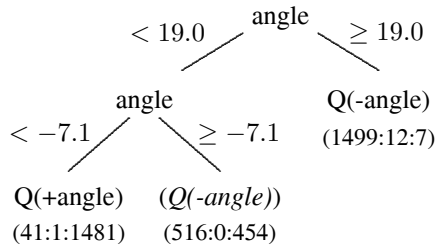


(b) Qualitative tree with Tube Regression and C4.5

Figure 3: Experiment with discrete attributes, function IF $s = 1$ THEN $f = -x/10$ ELSE $f = 10x$.



(a) Change of area w.r.t. distance



(b) Change of area w.r.t. angle (pruned)

Figure 7: Padé's modeling of XPERO data. Q() signifies negligible changes. Numbers in the leaves represent the number of examples with Q(-), Q() and Q(+), respectively.

Discrete Attributes and Unknown Values

The methods based on triangulation cannot handle unknown attribute values. Discrete attributes can be considered only if they are converted to dummy continuous variables, yet this gives awkward triangulations and meaningless results. Although not used in computation of derivatives, they can still be used in further processing of the data provided by Padé.

Tube regression can handle unknown values of attributes (except for the attribute on which we compute the derivative). This is, however, done through implicit imputation in distance computation procedure. Discrete values seem to pose no problems, as shown in experiments.

Total Derivatives

It may sometimes be interesting to observe the behavior of a function in a particular given direction not orthogonal to the coordinate axes. The adaptations of Padé's methods for that purpose are obvious. In First Triangle we align dx with the given direction, and in Tube regression we do the same with the tube. For Star Regression we can rotate the star in a similar fashion or, differently from the above tricks, compute multiple regression instead of univariate and treat the coefficients as a gradient. We then get the total derivative by multiplying the gradient with a (normalized) direction vector. None of these methods were implemented and evaluated yet.

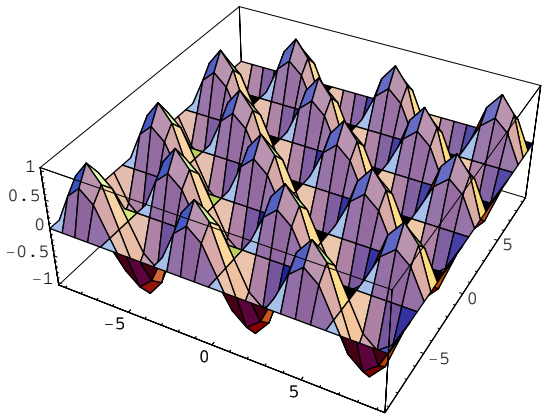
Time Complexity

For First triangle method, the most time consuming step is finding the triangle lying in the desired direction, which requires computing the determinant of a d-dimensional matrix (where d is the number of attributes). Such a triangle needs to be found for every point in space, for every attribute by which we compute the derivative. The running time strongly depends on the number of triangles that surround each point, which usually rises exponentially with the number of dimensions. In practice, the method is fast on low dimensional data and gets slower when the number of dimensions increases.

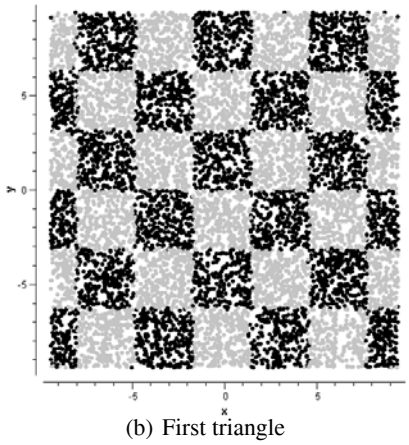
Tube regression's time complexity is linear in the number of dimensions and quadratic in the number of examples. It is consistently the slowest of all methods, except, possibly the Triangles' path, whose run time we have not measured.

Star regression always outran all other methods.

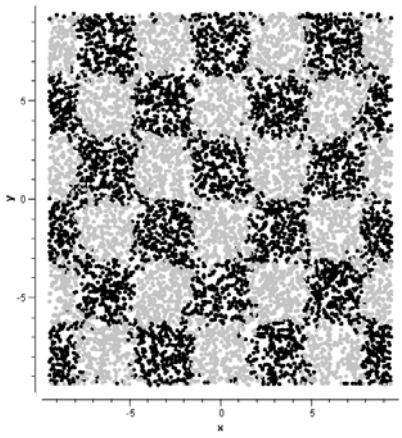
Table 1 sums up the running times of First triangle, Star regression and Tube regression for all experiments performed in the previous section.



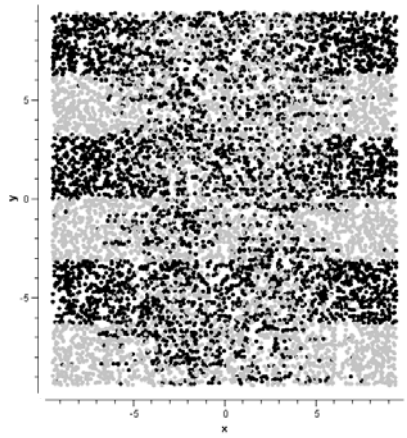
(a) Function $\sin x \sin y$



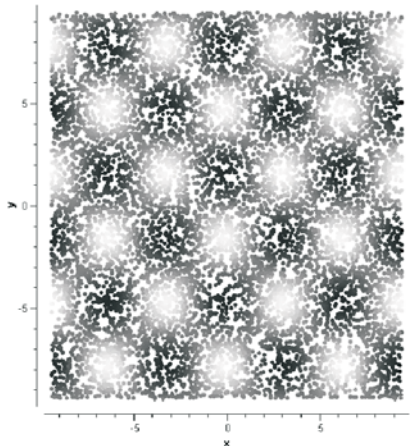
(b) First triangle



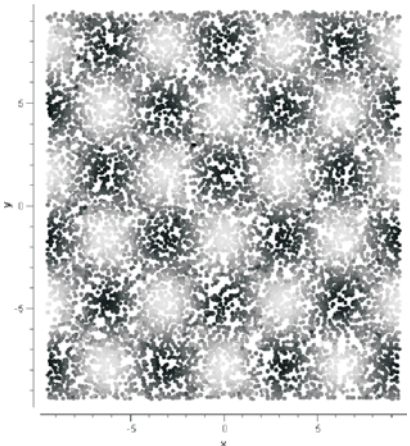
(c) Star regression



(d) Tube regression



(e) Numerical derivatives with First triangle



(f) Numerical derivatives with Star regression.

Figure 4: Function $\sin(x)\sin(y)$, its qualitative behavior and numeric derivatives by x of . Brighter and darker colors represent positive and negative derivatives. Color scales for graphs (e), (f) and other graphs are not comparable.

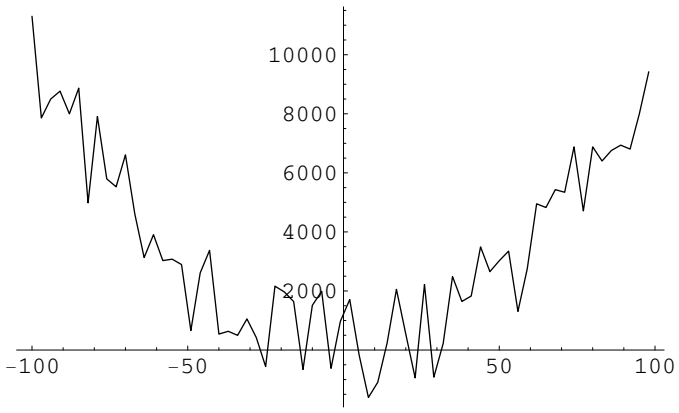


Figure 5: The intersection of the surface $x^2 - y^2$ with the plane $y = 0$ to illustrate the added noise.

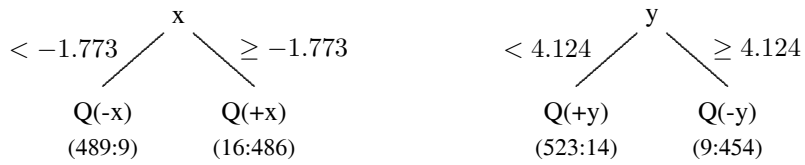


Figure 6: Qualitative models of function $x^2 - y^2$ with added random uniform noise. The data set contains 1000 randomly sampled examples.

	#attr	examples	Triangle	Star	Tube
pendulum	2	1,000	2	< 1	4
discrete	2	400	1	< 1	1
sine	3	10,000	35	8	496
noise	2	1,000	2	< 1	4
XPERO	2	4,011	15	3	80

Table 1: Run times (in seconds of CPU on a 2 GHz laptop) of First triangle, Star regression and Tube regression in the experiments.

Related Work

Many algorithms have, in one way or another, tackled the problem of qualitative model induction from observation data. Recently, Gerçeker and Say (Gerçeker & Say 2006) proposed algorithm LYQUID which fits polynomials to numerical data and use them to induce qualitative models. Other systems include QMN (Džeroski & Todorovski 1995), LAGRANGE (Džeroski & Todorovski 1993) and LAGRAMGE (Todorovski 2003). Other approaches that mostly induce models in the form of QDEs include GENMODEL for the induction of QSIM-type models (Hau & Coiera 1997), and SQUID (Kay, Rinner, & Kuipers 2000) which focuses on trends and extreme points in numerical data and use envelopes that bound the trajectories of variables.

An important difference between these algorithms and Padé is that Padé is essentially a preprocessor while other algorithms produce a model. Padé outputs a data set which can later be used by appropriate algorithms for induction of classification or regression models, or for visualization. So Padé in the context of learning qualitative models is of interest mainly in combination with other ML systems. To our knowledge, most other algorithms for learning qualitative models only handle numerical attributes, except QDE learners that take qualitative behaviors as input. In Padé, Tube regression can also use discrete attributes, whereas other methods are limited to continuous attributes. However, discrete attributes can be used by machine learning algorithms applied to Padé's output, which means that the final model can include discrete attributes.

We shall compare our work in more detail with the well-known algorithms QUIN and epQUIN (Šuc & Bratko 2001; Šuc 2003; Bratko & Šuc 2003). Examining the differences between Padé and QUIN will also be helpful for better understanding of the design of Padé itself.

The common property of Padé and QUIN (and, for that sake, any other algorithm for estimation of derivatives from sampled functions) is that they observe the local behavior of the function by summing up the information from sampled points in the vicinity of the point of interest. QUIN does this by comparing the attributes and class value at each pair of near data points, and constructs a vector of qualitative changes. These vectors are used to determine how well the learning data in various regions comply with possible qualitative constraints. epQUIN differs from QUIN by considering every pair of examples, not only near neighbors, but

weighting the evidence by the distance. The results are used to induce a qualitative tree, that is a decision tree with the qualitative constraints that fit well the corresponding data in the leaves.

The most obvious difference between Padé and QUIN is that Padé computes numerical derivatives, which can be (and in most of our experiments indeed were) later used qualitatively. QUIN, on the other hand, sums up the qualitative changes. While Padé estimates the magnitude of change, QUIN estimates the probabilities of various changes. These probabilities can be rather unreliable since they may be computed from small subsets of examples only.

There is an important difference in the definitions of qualitative proportionality constraints in Padé (denoted by Q), and monotonic qualitative constraints in QUIN (denoted by M). Padé's Q-constraints correspond to qualitative partial derivatives. QUIN's M-constraints, on the other hand, have a different definition (see (Šuc, Vladušič, & Bratko 2004)) illustrated by the following example. The M constraint $z = M^{+,-}(x, y)$ means: for all the points (x_1, y_1, z_1) and (x_2, y_2, z_2) in the region in which the constraint holds, we have: if $x_2 > x_1$ and $y_2 < y_1$ then $z_2 > z_1$. According to the continuous reification theorem (Šuc, Vladušič, & Bratko 2004), if x, y and z are continuous variables then if $z = M^{+,-}(x, y)$ holds then e.g. $z = M^{+}(x)$ cannot hold. This is obviously different from the Q-constraints. The difference comes from the fact that Padé only considers changes along the independent variables (which corresponds to partial derivatives), whereas QUIN considers changes in any direction (e.g. changes in both arguments x and y). This leads to a less apparent, yet crucial difference in the definition of vicinity in both systems.

A practical difference between the methods is that QUIN is implemented as a tree learning algorithm, while Padé is a data preprocessor which can be used with any learning or visualization algorithm. This is further simplified by Orange's versatile graphical interface for connecting various methods.

We noticed that QUIN is considerably slower than learning with Padé's and a typical chosen ML method, even when Padé is run with its slowest method - Tube Regression. It is though difficult to tell whether the difference comes from the algorithms themselves or only from their implementations.

Conclusion

We presented a novel method for learning qualitative models based on estimating partial derivatives from data. We developed an algorithm for estimation of partial derivatives from a sampled continuous function. The basic version of the algorithm, First Triangle, is based on splitting the attribute space into regions defined by Delaunay triangulation and the reasonable assumption that the function sample density is high enough to exhibit sufficiently linear behavior within the regions. The method is beautifully simple, but unfortunately unable to cope with any significant noise. To amend this, we developed several modifications of the method – Star Regression, Triangles' Path and Tube Regression. The methods are parameter-free, except for the threshold defining the negligible change if the numerical derivatives are transformed into qualitative changes. The only potential parameters would occur in Tube Regression, but since modifying them has no significant impact on the results, we – preferring simplicity over "tweakability" – froze the parameters and hid them from the user.

In experiments on a few artificial and semi-artificial data sets the algorithms behaved according to expectations, so we believe that they will also be useful in practice.

Padé has been implemented inside the general machine learning and data mining environment Orange, which can be freely downloaded (either as sources or in binary format for Windows or Linux) at http://www.ailab.si/orange.

Acknowledgment

This work was supported by the Slovenian research agency ARRS, and by the European project XPERO: Learning by Experimentation (IST-29427).

References

Bratko, I., and Šuc, D. 2003. Learning qualitative models. *AI Magazine* 24(4):107–119.

Džeroski, S., and Todorovski, L. 1995. Discovering dynamics: from inductive logic programming to machine discovery. *Journal of Intelligent Information Systems* 4:89–108.

Džeroski, S., and Todorovski, L. 1993. Discovering dynamics. In *International Conference on Machine Learning*, 97–103.

Gerçeker, R. K., and Say, A. C. C. 2006. Using polynomial approximations to discover qualitative models. In *In Proceedings of the 20th International Workshop on Qualitative Reasoning*.

Hau, D., and Coiera, E. 1997. Learning qualitative models of dynamic systems. *Machine Learning Journal* 26:177–211.

Jakulin, A.; Možina, M.; Demšar, J.; Bratko, I.; and Zupan, B. 2005. Nomograms for visualing support vector machines. In *KDD-2005: proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Kay, H.; Rinner, B.; and Kuipers, B. 2000. Semi-quantitative system identification. *Artificial Intelligence* 119:103–140.

Leban, G.; Zupan, B.; Vidmar, G.; and Bratko, I. 2006. Vizrank: Data visualization guided by machine learning. *Data Mining and Knowledge Discovery,* In press.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

Todorovski, L. 2003. *Using Domain Knowledge for Automated Modeling of Dynamic Systems with Equation Discovery*. Ph.D. Dissertation, University of Ljubljana, Ljubljana, Slovenia.

Šuc, D., and Bratko, I. 2001. Induction of qualitative trees. In De Raedt, L., and Flach, P., eds., *Proceedings of the 12th European Conference on Machine Learning*, 442–453. Springer. Freiburg, Germany.

Šuc, D.; Vladušič, D.; and Bratko, I. 2004. Qualitatively faithful quantitative prediction. *Artificial Intelligence* 158(2):189–214.

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

Zupan, B.; Leban, G.; and Demšar, J. 2004. *Orange: Widgets and visual programming, A White Paper*. Faculty of Computer and Information Science, Ljubljana, Slovenia.