Artificial Intelligence 175 (2011) 1604–1619

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Learning qualitative models from numerical data

Jure Žabkar [*], Martin Možina, Ivan Bratko, Janez Demšar

Faculty of Computer and Information Science, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia

A R T I C L E I N F O

Article history:
Received 16 March 2010
Received in revised form 23 February 2011
Accepted 23 February 2011
Available online 2 March 2011

Keywords:
Qualitative modelling
Regression
Partial derivatives
Monotone models

A B S T R A C T

Qualitative models describe relations between the observed quantities in qualitative terms. In predictive modelling, a qualitative model tells whether the output increases or decreases with the input. We describe Padé, a new method for qualitative learning which estimates partial derivatives of the target function from training data and uses them to induce qualitative models of the target function. We formulated three methods for computation of derivatives, all based on using linear regression on local neighbourhoods. The methods were empirically tested on artificial and real-world data. We also provide a case study which shows how the developed methods can be used in practice.

1. Introduction

People most often reason qualitatively. For example, playing with a simple pendulum, a five year old child discovers that the period of the pendulum increases if he uses a longer rope. Although most of us are later taught a more accurate numerical model describing the same behaviour, $T = 2\pi\sqrt{l/g}$, we keep relying on the more "operational" qualitative relation in everyday life. Still, despite Turing's proposition that artificial intelligence should *mimic human intelligence*, not much work has been done so far in trying to learn such models from data.

We can formally describe the relation between the period of a pendulum T, its length l and the gravitational acceleration g as $T = Q(+l, -g)$, meaning that the period increases with l and decreases with g. Different definitions of qualitative relations are discussed in related work. We will base ours on partial derivatives: a function f is in positive (negative) qualitative relation with x over a region $\mathcal{R}$ if the partial derivative of f with respect to x is positive (negative) over the entire $\mathcal{R}$,

$$f = Q_{\mathcal{R}}(+x) \quad \equiv \quad \forall x_0 \in \mathcal{R}\colon \frac{\partial f}{\partial x}(x_0) > 0 \tag{1}$$

and

$$f = Q_{\mathcal{R}}(-x) \quad \equiv \quad \forall x_0 \in \mathcal{R}\colon \frac{\partial f}{\partial x}(x_0) < 0. \tag{2}$$

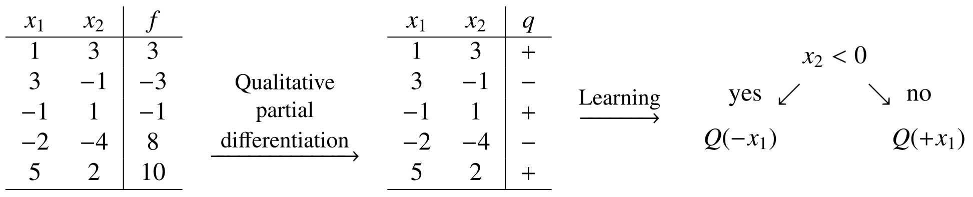
Qualitative predictive models describe qualitative relations between input variables and a continuous output, for instance

$$\begin{aligned} &\text{if } z > 0 \wedge x > 0 \quad \text{then } f = Q(+x),\\ &\text{if } z < 0 \vee x < 0 \quad \text{then } f = Q(-x). \end{aligned}$$

For sake of clarity, we omitted specifying the region since it is obvious from the context.

[*] Corresponding author. Tel.: +386 1 4768 154; fax: +386 1 4768 386.
E-mail addresses: jure.zabkar@fri.uni-lj.si (J. Žabkar), martin.mozina@fri.uni-lj.si (M. Možina), ivan.bratko@fri.uni-lj.si (I. Bratko), janez.demsar@fri.uni-lj.si (J. Demšar).

0004-3702/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2011.02.004



x_1	x_2	f
1	3	3
3	−1	−3
−1	1	−1
−2	−4	8
5	2	10

x_1	x_2	q
1	3	+
3	−1	−
−1	1	+
−2	−4	−
5	2	+

Fig. 1. General outline of the two-step method on a simple example. The target function, $f(x_1, x_2) = x_1 x_2$, is sampled in five points. The first step relabels the examples by replacing the value of the target function by the sign of the (estimated) derivative of f w.r.t. x_1. The second step induces a classification model from this data.

The paper includes three major contributions. The first one is the idea of transforming the problem of learning qualitative models to that of learning ordinary predictive models, described in the next section. This is followed by a new method called Padé[1] for computing partial derivatives from data typical for machine learning. We show three ways of computing the derivatives, all based on variations of local linear regression. Finally, we provide an extensive experimental evaluation of the proposed setup.

2. General outline of the method

We propose a new, two-step approach to induction of qualitative models from data, presented in Fig. 1. Input data describes a sampled function given as a set of examples $(\mathbf{x}, f)$, where $\mathbf{x}$ are attribute values and f is the function value.

In the first step we estimate the partial derivative at each point covered by a learning example. We replace the value of the output f for each example with the sign of the corresponding derivative (q). Each relabelled example, $(\mathbf{x}, q)$, describes the qualitative behaviour of the function at a single point.

In the second step, a general-purpose machine learning algorithm is used to generalise from this relabelled data, resulting in a qualitative model describing the function's (qualitative) behaviour in the entire domain.

Such models describe the relation between the output and a single input variable in dependence of other attributes. In Fig. 1 we modelled how the effect of x_1 on y depends upon the values of x_1 and x_2: the model tells that f decreases with increasing x_1 if x_2 is negative, and increases with increasing x_1 if x_2 is positive. In real life, we can model the economic conditions (*e.g.* interest rates, public debt, taxation, inflation and unemployment) at which an increase of interest rates will increase/decrease the unemployment. To describe the influence of multiple inputs (*e.g.* the effect of interest rates *and* of inflation and taxation on unemployment) we would induce a separate model for each independent attribute.

In this paper we will use the C4.5 tree inducer for construction of models, so the resulting models will be qualitative trees; these are, essentially, classification trees predicting qualitative behaviour. While we decided for the trees because of their interpretability, any other classification model, for instance naive Bayesian classifier, random forest or SVM could be used to induce qualitative models.

3. Related work

The beginnings of the field of qualitative reasoning go back to early work done outside AI. Jeffries [1] and May [2] introduced qualitative stability in ecology, whereas Samuelson [3] discussed the use of qualitative reasoning in economics. Examples of qualitative reasoning in economics also include the early work of Simon and Ando [4,5]. The papers by Forbus [6], de Kleer and Brown [7], and Kuipers [8] describe approaches that became the foundations of much of qualitative reasoning work in AI. Kalagnanam et al. [9–11] contributed to the mathematical foundations of qualitative reasoning.

There are a number of approaches to qualitative system identification, also known as learning qualitative models from data. Most of this work is concerned with the learning of QDE (Qualitative Differential Equations) models. One approach is to translate a numerical system's behaviour in time into a qualitative representation, and then check which QDE constraints (or QSIM-type constraints [8]) are satisfied by the qualitative behaviour. The resulting constraints constitute a qualitative model. Examples of this approach are the programs GENMODEL [12,13], MISQ [14,15] and QSI [16]. A similar approach can be carried out with a general purpose ILP system (Inductive Logic Programming) to induce a model from qualitative behaviours, which enjoys the advantages of the power and flexibility of ILP [17–19]. Džeroski and Todorovski developed several approaches to discovering dynamics. QMN [20] generates models in the form of qualitative differential equations. LAGRANGE [21] and LAGRAMGE [22] can describe the observed behaviour in the form of differential equations while system PADLES [23] extends the approach to learning partial differential equations. SQUID [24] finds models in terms of semi-quantitative differential equations. Gerçeker and Say [25] fit polynomials to numerical data and use them to induce qualitative models. We believe that their algorithm LYQUID can also be used in static systems although they experiment only with dynamic systems.

[1] An acronym for "partial derivative", and the name of a famous French mathematician.

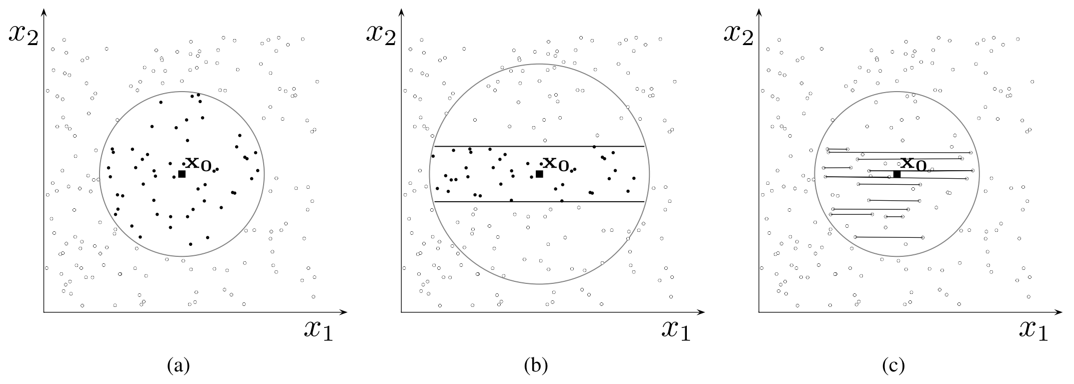


Fig. 2. The neighbourhoods for locally weighted regression (a), τ-regression (b) and parallel pairs (c).

Qualitative models of dynamic systems are not necessarily based on QDE. For example, the KARDIO model of the heart [26] uses symbolic descriptions in logic instead. QuMas [27,28] learned such models from example qualitative behaviours in time using an approach similar to ILP.

The QUIN algorithm [29–31] induces qualitative trees which are similar to classification trees but have different leaf nodes. The leaves of a qualitative tree contain qualitative constraints (*e.g.* $z = M^{+-}(x, y)$, meaning that z increases with x, decreases with y and depends on no other variables). QUIN constructs such trees by computing the qualitative change vectors between all pairs of points in the data and then recursively splitting the space into regions which share common qualitative properties, such as the one given above. Despite similarities, Padé and QUIN are significantly different in that Padé acts as a preprocessor of numerical data and can be used in combination with any attribute-value learning system. Padé computes qualitative partial derivatives in all example data points, and these derivatives become class values for the subsequent learning. For example, Padé combined with a decision tree learner would produce qualitative trees similar to those induced by QUIN. However, Padé combined with a rule learner, say, produces a model in the form of "qualitative rules". The main algorithmic difference between the two systems is that Padé computes quantitative and qualitative partial derivatives in every example point, whereas QUIN computes the degree of consistency of a subregion of numerical data with (in principle) every possible qualitative monotonicity constraint.

Padé is, to our knowledge, the only algorithm for computing (partial) derivatives on point cloud data. Another important difference between Padé and above mentioned algorithms 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 visualisation. This results in a number of Padé's advantages. For instance, most other algorithms for learning qualitative models only handle numerical attributes. Since Padé can be combined with any machine learning method, the learner can also use discrete attributes.

Besides symbolic methods, some interesting approaches to qualitative modelling are inspired by biological systems, *e.g.* [32,33].

Outside artificial intelligence, there are several numerical methods for computation of derivatives at a given point. Such numerical derivatives could be used for modelling qualitative behaviour. Unfortunately, numerical differentiation computes the function value at points chosen by the algorithm, while we are limited to a given data sample.

4. Computation of partial derivatives

We will denote a learning example as $(\mathbf{x}, y)$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and y is the value of the unknown sampled function, $y = f(\mathbf{x})$.

We will introduce three methods for estimation of partial derivative of f at point $\mathbf{x_0}$. The simplest one assumes that the function is linear in a small hyper-sphere around $\mathbf{x_0}$ (Fig. 2(a)). It computes *locally weighted linear regression* on examples lying in the hyper-sphere and considers the computed coefficients as partial derivatives. The second method, τ*-regression*, computes a single partial derivative at a time. To avoid the influence of other arguments of the function, it considers only those points in the sphere which lie in a hyper-tube along the axis of differentiation (Fig. 2(b)). The derivative can then be computed with weighted univariate regression. Finally, the *parallel pairs* method replaces the single hyper-tube with a set of pairs aligned with the axis of differentiation (Fig. 2(c)), which allows it to focus on the direction of differentiation without decreasing the number of examples considered in the computation.

4.1. Locally weighted regression

Let $\mathcal{N}(\mathbf{x_0})$ be a set of examples $(\mathbf{x_m}, y_m)$ such that $x_{mi} \approx x_{0i}$ for all i (Fig. 2(a)). According to Taylor's theorem, a differentiable function is approximately linear in a neighbourhood of $\mathbf{x_0}$,

$$f(\mathbf{x_m}) = f(\mathbf{x_0}) + \nabla f(\mathbf{x_0}) \cdot (\mathbf{x_m} - \mathbf{x_0}) + R_2. \tag{3}$$

Our task is to find the vector of partial derivatives, $\nabla f(\mathbf{x_0})$. We can solve this as a linear regression problem by rephrasing (3) as a linear model

$$y_m = \beta_0 + \boldsymbol{\beta}^{\mathrm{T}}(\mathbf{x_m} - \mathbf{x_0}) + \epsilon_m, \quad (\mathbf{x_m}, y_m) \in \mathcal{N}(\mathbf{x_0}), \tag{4}$$

where the task is to find $\boldsymbol{\beta}$ (and β_0) with the minimal sum of squared errors ϵ_m over $\mathcal{N}(\mathbf{x_0})$. The error term ϵ_m covers the remainder of the Taylor expansion, R_2, as well as noise in the data.

The size of the neighbourhood $\mathcal{N}(\mathbf{x_0})$ should reflect the density of examples and the amplitude of noise. Instead of setting a predefined radius (*e.g.* $\|\mathbf{x_m} - \mathbf{x_0}\| < \delta$), we consider a neighbourhood of k nearest examples and weigh the points according to their distance from $\mathbf{x_0}$,

$$w_m = e^{-\|\mathbf{x_m}-\mathbf{x_0}\|^2/\sigma^2}. \tag{5}$$

The parameter σ^2 is fitted so that the farthest example has a negligible weight of 0.001.

This transforms the problem into locally weighted regression (LWR) [34], where the regression coefficients represent partial derivatives,

$$\begin{bmatrix} \beta_0 \\ \boldsymbol{\beta} \end{bmatrix} = \left(X^{\mathrm{T}} W X\right)^{-1} X^{\mathrm{T}} W Y, \tag{6}$$

where

$$X = \begin{bmatrix} 1 & x_{11} & \dots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{k1} & \dots & x_{kn} \end{bmatrix}, \qquad W = \begin{bmatrix} w_1 & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & w_k \end{bmatrix}, \qquad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix}. \tag{7}$$

The computed $\boldsymbol{\beta}$ estimates the vector of partial derivatives $\nabla f(\mathbf{x_0})$.

As usual in linear regression, the inverse in (6) can be replaced by pseudo-inverse to increase the stability of the method.

Fig. 3(a) shows a simple example for computation of derivative of $f(x_1, x_2) = x_1^2 - x_2^2$ at point $(10, 10)$ on a neighbourhood of $k = 5$ examples.

4.2. τ-regression

The τ-regression algorithm differs from LWR in the shape of the neighbourhood of the reference point. It starts with κ examples in a hyper-sphere, which is generally larger than that for LWR, but then keeps only k examples that lie closest to the axis of differentiation (Fig. 2(b)). Let us assume without loss of generality that we compute the derivative with regard to the first argument x_1. The neighbourhood $\mathcal{N}(\mathbf{x_0})$ thus contains k examples with the smallest distance $\|\mathbf{x_m} - \mathbf{x_0}\|_{\backslash 1}$ chosen from the κ examples with the smallest distance $\|\mathbf{x_m} - \mathbf{x_0}\|$, where $\|\cdot\|_{\backslash i}$ represents the distance computed over all dimensions except the i-th.

With a suitable selection of κ and k, we can assume $|x_{m1} - x_{01}| \gg |x_{mi} - x_{0i}|$ for all $i > 1$ for most examples $\mathbf{x_m}$. If we also assume that partial derivatives with regard to different arguments are comparable in size, we get $\partial f/\partial x_1(x_{m1} - x_{01}) \gg \partial f/\partial x_i(x_{mi} - x_{0i})$ for $i > 1$. We can thus omit all dimensions but the first from the scalar product in (3):

$$f(\mathbf{x_m}) \approx f(\mathbf{x_0}) + \frac{\partial f}{\partial x_1}(\mathbf{x_0})(x_{m1} - x_{01}) + R_2 \tag{8}$$

for $(\mathbf{x_m}, y_m) \in \mathcal{N}(\mathbf{x_0})$. We again set up a linear model,

$$y_m = \beta_0 + \beta_1(x_{m1} - x_{01}) + \epsilon_m, \tag{9}$$

where β_1 approximates the derivative $\frac{\partial f}{\partial x_1}(\mathbf{x_0})$. The task is to find the value of β_1 which minimises error over $\mathcal{N}(\mathbf{x_0})$.

Examples in $\mathcal{N}(\mathbf{x_0})$ are weighted according to their distance from $\mathbf{x_0}$ along the axis of differentiation,

$$w_m = e^{-(x_{m1}-x_{01})^2/\sigma^2}, \tag{10}$$

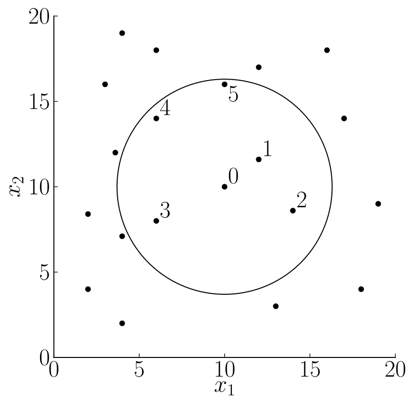
where σ is again set so that the farthest example has a weight of 0.001.

The described linear model can be solved by weighted univariate linear regression over the neighbourhood $\mathcal{N}(\mathbf{x_0})$,

$$\frac{\partial f}{\partial x_1}(\mathbf{x_0}) = \beta_1 = \frac{\sum w_m x_{m1} y_m - \sum w_m x_{m1} \sum w_m y_m / \sum w_m}{\sum w_m x_{m1}^2 - (\sum w_m x_{m1})^2 / \sum w_m}, \tag{11}$$

where $\sum$ stands for $\sum_{\mathbf{x_m} \in \mathcal{N}(\mathbf{x_0})}$.

Fig. 3(b) shows a computation of derivative of $f(x_1, x_2) = x_1^2 - x_2^2$ at point $(10, 10)$ for $\kappa = 7$ and $k = 5$.

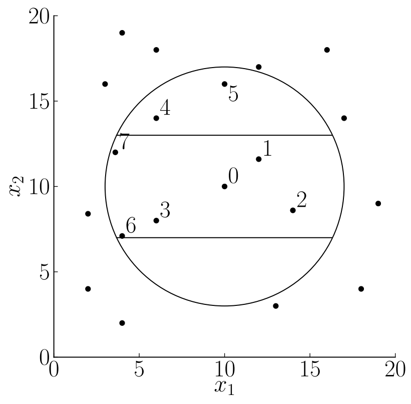


m	x_1	x_2	y	$\lVert\mathbf{x_m} - \mathbf{x_0}\rVert$	w
1	12.0	11.6	9.44	6.56	0.284
2	14.0	8.6	122.04	17.96	0.032
3	6.0	8.0	-28.0	20.0	0.022
4	6.0	14.0	-160.0	32.0	0.002
5	10.0	16.0	-156.0	36.0	0.001

$$y = 40.82 + 20.44x_1 - 23.85x_2$$

$$\frac{\partial y}{\partial x_1}(10, 10) = 20.44$$

(a) Locally weighted regression

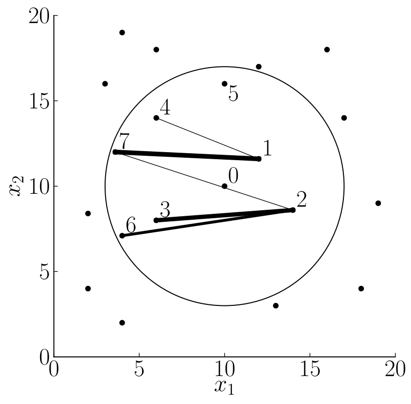


m	x_1	x_2	y	$\lvert x_{m1} - x_{01}\rvert$	w
1	12.0	11.6	9.44	2.0	0.509
2	14.0	8.6	122.04	4.0	0.067
3	6.0	8.0	-28.0	4.0	0.067
6	4.0	7.1	-34.41	6.0	0.002
7	3.6	12.0	-131.04	6.4	0.001

$$y = -126.49 + 12.42x_1$$

$$\frac{\partial y}{\partial x_1}(10, 10) = 12.42$$

(b) τ-regression



(m, j)	$x_{(m,j)1} =$ $x_{m1} - x_{j1}$	$y_{(m,j)1} =$ $y_m - y_j$	$\alpha_{m,j}$	w
(7,1)	-8.4	-140.48	2.7	0.893
(3,2)	-8.0	-150.04	4.3	0.756
(6,2)	-10.0	-156.45	8.5	0.332
(7,2)	-10.4	-253.08	18.1	0.008
(4,1)	-6.0	-169.44	21.8	0.001

$$y_{(m,j)1} = -49.05 + 11.52x_{(m,j)1}$$

$$\frac{\partial y}{\partial x_1}(10, 10) = 11.52$$

(c) Parallel pairs

Fig. 3. Computation of partial derivatives of $y = x_1^2 - x_2^2$ at point $(10, 10)$.

4.3. Parallel pairs

Consider two examples $(\mathbf{x_m}, y_m)$ and $(\mathbf{x_j}, y_j)$ which are close to $\mathbf{x_0}$ and aligned with the x_1 axis, $\lVert\mathbf{x_m} - \mathbf{x_j}\rVert \approx |x_{m1} - x_{j1}|$. Under these premises we can suppose that both examples correspond to the same linear model (9) with the same coefficients β_0 and β_1. Subtracting (9) for y_m and y_j gives

$$y_m - y_j = \beta_1(x_{m1} - x_{j1}) + (\epsilon_m - \epsilon_j) \tag{12}$$

or

$$y_{(m,j)} = \beta_1 x_{(m,j)1} + \epsilon_{(m,j)}, \tag{13}$$

where $y_{(m,j)} = y_m - y_j$ and $x_{(m,j)1} = x_{m1} - x_{j1}$. The difference $y_m - y_j$ is therefore linear with the difference in the attribute values x_{m1} and x_{j1}.

Coefficient β_1 again approximates the derivative $\frac{\partial f}{\partial x_1}(\mathbf{x_0})$. Note that the model has no intercept term, β_0.

To compute the derivative using (12) we take the spherical neighbourhood like the one from the first method, LWR. For each pair we compute its alignment with the x_1 axis using a scalar product with the base vector $\mathbf{e_1}$,

$$\alpha_{(m,j)} = \left| \frac{(\mathbf{x_m} - \mathbf{x_j})^{\mathrm{T}} \mathbf{e_1}}{\|\mathbf{x_m} - \mathbf{x_j}\| \|\mathbf{e_1}\|} \right| = \frac{|x_{m1} - x_{j1}|}{\|\mathbf{x_m} - \mathbf{x_j}\|}. \tag{14}$$

We select the k best aligned pairs from κ points in the hyper-sphere around $\mathbf{x_0}$ (Fig. 2(c)) and assign them weights corresponding to the alignment,

$$w_{(m,j)} = e^{-(1-\alpha_{(m,j)})^2/\sigma^2}, \tag{15}$$

with σ^2 set so that the smallest weight equals 0.001.

The derivative is again computed using univariate linear regression,

$$\frac{\partial f}{\partial x_1}(\mathbf{x_0}) = \beta_1 = \frac{\sum_{(\mathbf{x_m},\mathbf{x_j}) \in \mathcal{N}(\mathbf{x_0})} w_{(m,j)}(x_{m1} - x_{j1})(y_m - y_j)}{\sum_{(\mathbf{x_m},\mathbf{x_j}) \in \mathcal{N}(\mathbf{x_0})} w_{(m,j)}(x_{m1} - x_{j1})^2}. \tag{16}$$

Fig. 3(c) illustrates the computation with $k = 5$ pairs from $\kappa = 7$ points. Note that the regression formula now uses $x_{(m,j)1}$ as independent and $y_{(m,j)1}$ as dependent variable.

4.4. Time complexity

Let the data set contain N points. Described methods compute partial derivative at a single point by first finding the k nearest neighbours with distances computed from values of n attributes, which requires $O(N(n + \ln k))$. The costliest remaining step of the LWR method is computation of the inverse of a $k \times k$ matrix, $X^T W X$ which takes $O(k^3)$ using brute force, or $O(k^{2.376})$ using the Coppersmith–Winograd algorithm. τ-regression and parallel pairs use univariate linear regression so their running times are $O(k)$ and $O(\kappa)$, respectively. Finding the nearest neighbours usually dominates over the time complexity of all other steps. The time complexity of computing the derivatives at all points in the data set thus rises quadratically with the number of points, $O(N^2(n + \ln k))$.

The time complexity of the second step of the schema, building the qualitative model from examples labelled by qualitative derivatives, depends upon the used machine learning algorithm and its settings.

5. Experiments

We first performed an extensive set of experiments on artificially constructed problems. These data sets are used to test Padé with respect to accuracy, the effect of irrelevant attributes and the effect of noise in the data.

To assess the accuracy of induced models, we compute the derivatives and the model from the entire data set. We then check whether the predictions of the model match the analytically computed partial derivatives. We define the accuracy of Padé as the proportion of examples with correctly predicted qualitative partial derivatives. Note that this procedure does not require separate training and testing data set since the correct answer with which the prediction is compared is not used in induction of the model. Where not stated otherwise, experimental results represent averages of ten trials.

Another set of experiments was performed on real-world data. Since the ground truth (the correct partial derivative) on such data sets is not necessarily known, we cannot compute the predictive accuracy of the model. In such cases, stability is often used as a measure of quality of a machine learning method [35]. Stability measures the influence that variation of the input has on the output of a system. The motivation for such analysis is to design robust systems that will not be affected by noise and randomness in sampling.

In experiments with the parallel pairs method we reduced the number of arguments to be fitted by setting $\kappa = k$. Since the first batch of experiments on artificial data sets suggested that parameter settings do not significantly affect the performance of methods, we use constant values in other experiments: we use $k = 20$ for LWR and parallel pairs, and $\kappa = 100$ and $k = 20$ for τ-regression, except for real-world data sets where we decreased κ to 50 due to a smaller number of examples.

For LWR, we used ridge regression to compute the ill-posed inverse in (6). We used C4.5 as reimplemented in Orange [36] for induction of qualitative models. We preferred this algorithm over potentially more accurate methods, like SVM, since one of our goals was to induce comprehensible models. Besides that, most artificial data sets are simple enough to require only a single-node tree or a tree with two leaves only. C4.5 was run with default settings except where noted otherwise.

Table 1
Results of experiments with $f(x, y) = x^2 - y^2$.

	LWR	Pairs	τ-regression			
			$\kappa = 30$	50	70	100
(a) Accuracy of qualitative derivatives						
$k = 10$	0.991	0.986	0.948	0.968	0.971	0.980
20	0.993	0.992	0.929	0.960	0.972	0.981
30	0.992	0.992	0.909	0.953	0.969	0.978
40	0.993	0.993		0.950	0.964	0.974
50	0.994	0.993		0.935	0.961	0.972
(b) Accuracy of qualitative models induced by C40.5						
$k = 10$	0.997	0.993	0.990	0.993	0.990	0.991
20	0.996	0.995	0.983	0.991	0.986	0.991
30	0.996	0.994	0.983	0.987	0.988	0.993
40	0.995	0.995		0.978	0.978	0.990
50	0.998	0.995		0.977	0.987	0.991

Table 2
Results of experiments with $f(x, y) = x^3 - y$.

	LWR	Pairs	τ-regression			
			$\kappa = 30$	50	70	100
(a) Accuracy of qualitative derivatives						
$k = 10$	0.727	0.690	0.597	0.626	0.639	0.647
20	0.725	0.792	0.576	0.593	0.619	0.646
30	0.751	0.879	0.545	0.571	0.609	0.618
40	0.740	0.919		0.556	0.588	0.613
50	0.725	0.966		0.541	0.558	0.612
(b) Accuracy of qualitative models induced by C40.5						
$k = 10$	10.00	0.898	0.737	0.880	0.890	0.864
20	10.00	0.930	0.745	0.743	0.811	0.860
30	0.978	0.954	0.752	0.599	0.813	0.777
40	0.971	0.964		0.780	0.732	0.700
50	0.956	0.986		0.906	0.805	0.760

5.1. Accuracy on artificial data sets

We performed experiments with three mathematical functions: $f(x, y) = x^2 - y^2$, $f(x, y) = x^3 - y$, $f(x, y) = \sin x \sin y$. We sampled them uniform randomly in 1000 points in the range $[-10, 10] \times [-10, 10]$.

Function $f(x, y) = x^2 - y^2$ is a standard test function often used in [30]. Its partial derivative w.r.t. x is $\partial f/\partial x = 2x$, so $f = Q(+x)$ if $x > 0$ and $f = Q(-x)$ if $x < 0$. Since the function's behaviour with respect to y is similar, we observed only results for $\partial f/\partial x$. The accuracy of all methods is close to 100% (Table 1). Changing the values of parameters has no major effect except for τ regression, where short ($\kappa = 30$ and $\kappa = 50$) and wide ($k \geqslant 10$) tubes give better accuracy while for very long tubes ($\kappa = 100$) the accuracy decreases with k. The latter can indicate that longer tubes reach across the boundary between the positive and negative values of x. Induced tree models have the same high accuracy.

Function $f(x, y) = x^3 - y$ is globally monotone, increasing in x and decreasing in y in the whole region. The function is interesting because its much stronger dependency on x can obscure the role of y. All methods have a 100% accuracy with regard to x. Prediction of function's behaviour w.r.t. y proves to be difficult: the accuracy of τ-regression is 50–60% and the accuracy of LWR is just over 70% (Table 2). Parallel pairs seem undisturbed by the influence of x and estimate the sign of $\partial f/\partial y$ with accuracy of more than 95% with proper parameter settings.

An interesting observation here is that the accuracy of induced qualitative tree models highly exceeds that of point-wise partial derivatives. For instance, qualitative models for derivatives by LWR reach 95–100% despite the low, 70% accuracy of estimates of the derivative. When generalising from labels denoting qualitative derivatives, incorrect labels are scattered randomly enough that C4.5 recognises them as noise and induces a tree with a single node.

Function $f(x, y) = \sin x \sin y$ has partial derivatives $\partial f/\partial x = \cos x \sin y$ and $\partial f/\partial y = \cos y \sin x$, which change their signs multiple times in the observed region. The accuracy of all methods is mostly between 80 and 90%, degrading with larger neighbourhoods (Table 3). However, the accuracy of C4.5 barely exceeds 50% which we would get by making random predictions. Rather than a limitation of Padé, this shows the (expected) inability of C4.5 to learn this chessboard-like concept.

Since qualitative modelling has been traditionally interested in examples from physics, we also tested Padé's ability to rediscover three simple physical laws from computer generated data. Each domain is described by an equation which was used to obtained 1000 uniform random samples.

Table 3
Results of experiments with $f(x, y) = \sin x \sin y$.

	LWR	Pairs	τ-regression			
			$\kappa = 30$	50	70	100
(a) Accuracy of qualitative derivatives						
$k = 10$	0.882	0.858	0.863	0.885	0.890	0.886
20	0.870	0.841	0.820	0.865	0.880	0.882
30	0.862	0.823	0.769	0.837	0.861	0.877
40	0.844	0.796		0.799	0.839	0.853
50	0.814	0.754		0.717	0.810	0.845
(b) Accuracy of qualitative models induced by C40.5						
$k = 10$	0.509	0.519	0.516	0.512	0.510	0.509
20	0.515	0.531	0.509	0.518	0.522	0.510
30	0.515	0.523	0.510	0.513	0.514	0.515
40	0.521	0.555		0.511	0.507	0.511
50	0.516	0.583		0.507	0.508	0.515

Table 4
Accuracy of computed partial derivatives for domains from physics.

(a) Centripetal force			
Variable	m	v	r
LWR	1.00	1.00	1.00
τ-regression	0.86	0.97	0.94
Parallel pairs	1.00	1.00	0.98
(b) Gravitational force			
Variable	m_1	m_2	r
LWR	0.96	0.96	1.00
τ-regression	0.88	0.87	0.99
Parallel pairs	0.96	0.96	1.00
(c) Pendulum			
Variable	l	φ	
LWR	1.00	0.98	
τ-regression	1.00	0.83	
Parallel pairs	1.00	0.97	

5.1.1. Centripetal force

The centripetal force F on an object of mass m moving at a speed v along a circular path with radius r is

$$F = \frac{mv^2}{r}.$$

We prepared a data set for this target concept with m sampled randomly from $[0.1, 1]$, r from $[0.1, 2]$ and v from $[1, 10]$.

Apart from the failure of τ-regression on $\partial F/\partial m$, which we cannot explain, the proportion of correctly computed qualitative partial derivatives (Table 4(a)) is around 100%. C4.5 correctly induced single-node trees predicting $F = Q(+m)$, $F = Q(+v)$ and $F = Q(-r)$.

5.1.2. Gravitation

Newton's law of universal gravitation states that a point mass attracts another point mass by a force pointing along the line intersecting both points with the force equal to

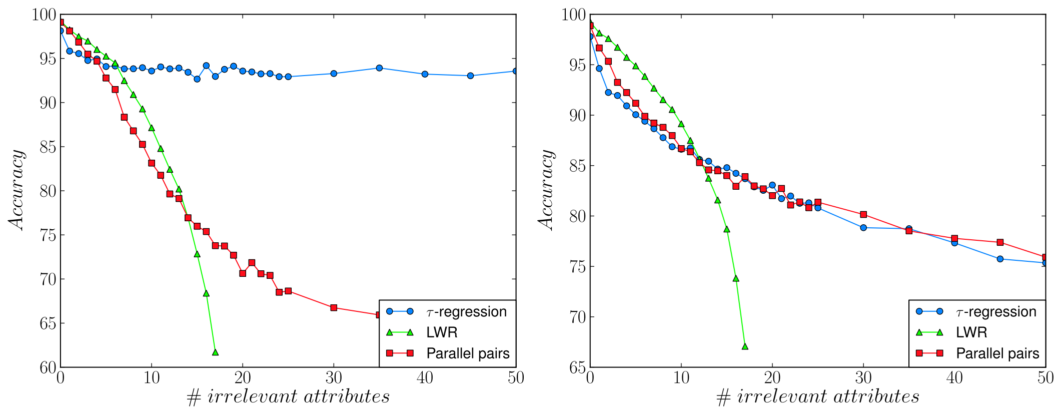
$$F = G\frac{m_1 m_2}{r^2},$$

where F is gravitational force, G is the gravitational constant, m_1 and m_2 are the points' masses, and r is the distance between the two point masses. We prepared a data set where m_1 and m_2 were sampled randomly from $[0.1, 1]$ and r from $[0.1, 2]$.

Padé again reached almost 100% accuracy (Table 4(b)) and C4.5 induced correct models $F = Q(+m_1)$, $F = Q(+m_2)$ and $F = Q(-r)$.

$$\begin{tikzcd}
 & \varphi \arrow[dl, "\leq -1.06"'] \arrow[dr, ">-1.06"] & \\
T = Q(-\varphi) & & T = Q(+\varphi)
\end{tikzcd}$$

Fig. 4. The qualitative model describing the relation between T and φ.



(a) Data set $x^2 - y^2$.

(b) Data set xy.

Fig. 5. Accuracy for different number of additional irrelevant attributes.

5.1.3. Pendulum

The period T of the first swing of a pendulum w.r.t. its length l, the mass of the bob m and the initial angle φ_0 is[2]

$$T = 2\pi\sqrt{\frac{l}{g}}\left(1 + \frac{1}{16}\varphi_0^2 + \frac{11}{3072}\varphi_0^4 + \frac{173}{737\,280}\varphi_0^6 + \cdots\right).$$

The mass was again chosen randomly from $[0.1, 1]$, the length was from $[0.1, 2]$, the angle was between -180 and $180°$, and $g = 9.81$.

Table 4(c) shows the accuracies for dependency of the period on the length and the angle. All methods achieved 100% accuracy for the length, *i.e.* for each learning example the qualitative derivative equals the ground truth.

The qualitative model describing the relation between T and φ depends on the value of φ. C4.5 induced a qualitative tree shown in Fig. 4 from data labelled by LWR. The models based on τ-regression and parallel pairs are the same – the only difference is in the split threshold at the root node which is $-2.7°$ for τ-regression and $-2.28°$ for parallel pairs. The tree states that for negative angles the period T decreases with φ while for positive angles it increases with φ. In other words, the period increases with the absolute value of φ.

5.2. Effect of irrelevant attributes

Many real-world data sets include large number of attributes with no effect on the function value. The following experiment shows the robustness of Padé in such cases.

We consider data sets obtained by sampling $f(x, y) = x^2 - y^2$ and $f(x, y) = xy$ as described above and observe the correctness of partial derivatives computed by Padé when 0–50 random attributes from the same interval as x and y, $[-10, 10]$, are added to the domain. The graphs in Fig. 5 show how the accuracy drops with the increasing number of added irrelevant attributes for each method.

We observe a steep drop in the accuracy of LWR which we can ascribe to the fact that LWR is multivariate and is solving an ill-conditioned system as the number of attributes approaches k. Parallel pairs and τ-regression avoid this problem by computing the derivative by a single attribute at a time.

Similar results of the latter two algorithms on $f(x, y) = xy$ may reflect the fact that both deal with the same problem, that is, that the gradient of the function is not aligned with the axis of differentiation; the problem gets worse with increasing number of additional random variables. The algorithms behave quite differently for $f(x, y) = x^2 - y^2$. Here, the hyper-plane at $x = 0$ represents the boundary between positive and negative derivatives w.r.t. x. τ-regression uses a larger neighbourhood than parallel pairs, which is generally a disadvantage, yet in this case approximately the same number of

[2] http://en.wikipedia.org/wiki/Pendulum.

Table 5
Effect of noise on the accuracy of computed qualitative partial derivatives for $f(x, y) = x^2 - y^2$ with random noise from $N(0, \sigma)$.

	$\sigma = 0$	$\sigma = 10$	$\sigma = 30$	$\sigma = 50$
(a) Correctness of computed derivatives				
LWR	0.993	0.962	0.878	0.795
τ-regression	0.981	0.945	0.848	0.760
parallel pairs	0.992	0.924	0.771	0.680
(b) Correctness of qualitative models induced by C4.5				
LWR	0.996	0.978	0.956	0.922
τ-regression	0.991	0.976	0.949	0.917
parallel pairs	0.995	0.966	0.949	0.901

examples from both sides of the boundary is added so the qualitative predictions still come out correct. Parallel pairs, on the other hand, choose pairs according to their angles with the axis of differentiation. These become random as the number of random variables increases, so the derivative is computed in a random direction.

It is difficult to estimate whether this distinction between the two methods also affects their performance in real-world problems. We have performed the same experiment on a number of other function and found that no method consistently outperforms the other.

5.3. *Coping with noise*

In this experiment we add various amounts of noise to the function value. The target function is $f(x, y) = x^2 - y^2$ defined on $[-10, 10] \times [-10, 10]$ which puts f in $[-100, 100]$. We added Gaussian noise with a mean of 0 and standard deviation 0, 10, 30, and 50, *i.e.* from no noise to the noise in the range comparable to the signal itself. We measured the accuracy of derivatives and models induced by C4.5. Since the data contains noise, we set the C4.5's parameter m (minimal number of examples in a leaf) to 10% of the examples of our data set, $m = 100$. We repeated the experiment with each amount of noise 100 times and computed the average accuracies.

The results are shown in Table 5. Padé is quite robust despite the huge amount of noise. As in other experiments on artificial data sets, we again observed that C4.5 is able to learn almost perfect models despite the drop in correctness of derivatives at higher noise levels.

5.4. *Stability on real-world data sets*

In measurements of stability we ran Padé on six UCI ML repository [37] data sets (servo, imports-85, galaxy, prostate, housing, auto-mpg) and the four artificial data sets (xy, $x^2 - y^2$, $x^3 - y$, $\sin x \sin y$) which were already used in previous experiments. For each data set we created 100 bootstrap samples and for each sample we computed qualitative partial derivatives of the class w.r.t. all continuous attributes for each example. We defined stability w.r.t. attribute x_i for example $\mathbf{x_m}$, $\beta_{\mathbf{x_m}}(x_i)$, as the proportion of the prevailing derivatives ($Q_{\mathbf{x_m}}(+x_i)$ or $Q_{\mathbf{x_m}}(-x_i)$) among all positive and negative derivatives predicted for $\mathbf{x_m}$,

$$\beta_{\mathbf{x_m}}(x_i) = \frac{max(\#Q_{\mathbf{x_m}}(+x_i), \#Q_{\mathbf{x_m}}(-x_i))}{\#Q_{\mathbf{x_m}}(+x_i) + \#Q_{\mathbf{x_m}}(-x_i)}. \tag{17}$$

The total stability with regard to attribute x_i is the average stability across all examples in the data set. We prefer β close to 1, which happens when most of the qualitative predictions for this derivative are the same. The lowest possible β is 0.5.

The settings for all methods were $k = 20$ and $\kappa = 50$ for τ-regression. The lower value of κ was used due to a smaller number of examples. Results in Table 6 show the average stability of the partial derivative of the outcome w.r.t. to each attribute.

Altogether there are 46 continuous attributes in all domains. We ranked performances of methods for each attribute, where rank 1 was assigned to the best method and rank 3 to the worst one. Averaging over all attributes showed that the parallel pairs method is the most stable with a rank of 1.80, slightly behind is the τ-regression method with 1.84, while the LWR method achieved a considerably worse rank of 2.35.

5.5. *Comparison between Padé and QUIN*

We conclude the experiments with comparison between Padé and QUIN. The two algorithms can be used for similar purposes, yet it is difficult to bring them to the same ground. While the output of QUIN are tree models, Padé computes partial derivatives, which can be used for constructing trees or other kinds of models or visualisations. For sake of comparison, we observed trees constructed by C4.5 on data relabelled by Padé, as elsewhere in the paper. Padé's aim is to find a partial derivative w.r.t. a *given variable* at a given point; QUIN identifies areas with qualitatively invariant behaviour of the function w.r.t. to *some variables* not specified in advance. For objectivity, we followed the definition from [38] stating that absence of

Table 6
Stability on real and artificial data sets. The numbers correspond to stabilities of LWR, τ-regression and parallel pairs, respectively.

Data set/attribute	LWR	τ	PP
Auto			
displacement	0.76	0.80	0.75
horsepower	0.71	0.84	0.75
weight	0.77	0.81	0.78
acceleration	0.75	0.76	0.77
Imports-85			
normalized-losses	0.68	0.76	0.72
wheel-base	0.66	0.85	0.89
length	0.72	0.85	0.82
width	0.67	0.82	0.94
curb-weight	0.88	0.88	0.96
height	0.72	0.79	0.75
engine-size	0.71	0.80	0.91
bore	0.67	0.78	0.85
compression-ratio	0.68	0.81	0.79
stroke	0.65	0.80	0.81
horsepower	0.64	0.89	0.93
peak-rpm	0.66	0.83	0.75
highway-mpg	0.66	0.87	0.84
city-mpg	0.65	0.87	0.88
Housing			
crim	0.71	0.80	0.78
indus	0.68	0.85	0.81
nox	0.70	0.80	0.85
rm	0.87	0.88	0.92
age	0.82	0.79	0.79
dis	0.71	0.81	0.81
tax	0.70	0.88	0.84
ptratio	0.67	0.85	0.84
b	0.68	0.76	0.75
lstat	0.71	0.91	0.93

Data set/attribute	LWR	τ	PP
Galaxy			
east.west	0.93	0.90	0.88
north.south	0.93	0.90	0.80
radial.position	0.93	0.90	0.83
Prostate			
lcavol	0.92	0.82	0.93
lweight	0.78	0.80	0.80
age	0.69	0.79	0.79
lbph	0.69	0.78	0.78
lcp	0.71	0.77	0.71
Servo			
pgain	0.95	0.94	1.00
vgain	0.88	0.72	0.85
xy			
x	0.99	0.95	0.99
y	0.99	0.94	0.98
$x^3 - y$			
x	1.00	1.00	1.00
y	0.77	0.75	0.78
$x^2 - y^2$			
x	0.99	0.95	0.99
y	0.99	0.96	0.99
$\sin x \sin y$			
x	0.91	0.88	0.86
y	0.91	0.89	0.86

a variable from a leaf in the QUIN's tree means that the derivative with regard to that variable is zero, although in practise it is often considered to be insignificant or undefined. Padé does not predict zero derivatives. Instead of comparing accuracy of the algorithms we therefore show the confusion matrices in which the results of Padé have one column less.

We report on results obtained with τ-regression; performance of parallel pairs is similar. We used the same parameters as elsewhere in the paper ($\kappa = 100$, $k = 20$). QUIN was also run with default parameters.

We performed the comparison on three artificial functions: $x^2 - y^2$, $x^3 - y$ and xy, under three different conditions: noiseless-data including only the relevant variables x and y, data with Gaussian noise with $\sigma = 10$ as described in Section 5.3, and noiseless data with three additional random variables unrelated to the output. We constructed 1000 training and 1000 testing instances for each domain. We repeated each experiment 10 times and report the averages.

Results are given in Table 7. For instance, on noiseless data there were 500 points with positive derivative of $f(x, y) = x^2 - y^2$ w.r.t. x, out of which QUIN (correctly) predicted the positive derivative in 482 points and a zero derivative at 17 (the missing example is due to rounding error).

Both algorithms achieve good results on noiseless and noisy data, except for the QUIN's overlooking of the effect of y in $f(x, y) = x^3 - y$. Padé makes a correct prediction in 71% cases for this problem. It is however impossible to tell what would happen if QUIN could be forced into predicting either increase or decrease or if Padé was allowed to abstain, as QUIN is.

QUIN occasionally gives quite different results for x and y on symmetric functions $f(x, y) = x^2 - y^2$ and $f(x, y) = xy$, which cannot be ascribed to the algorithm and can only be explained as implementational issue.

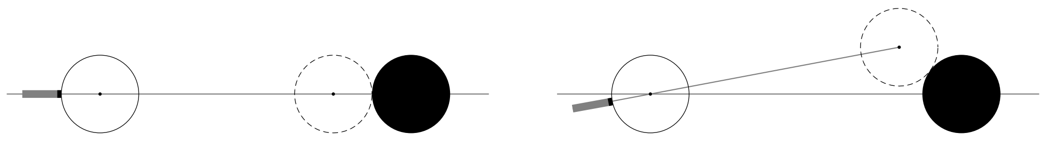
The performance of the two algorithms significantly diverges in the presence of irrelevant variables. Padé is affected only on $f(x, y) = x^3 - y$, where the inferior influence of y is additionally obscured by random variables. QUIN's results degrade significantly. This can be explained by the different nature of the two algorithms: Padé's τ-regression and parallel pairs are univariate methods, and QUIN is multivariate and models the effect of all attributes at once. Padé observes the values of other attributes only in the definition of the neighbourhood, while in QUIN they are also a part of the target concept.

6. Case study: billiards

To show a practical use of the proposed methods, we analyse a problem from billiards. Billiards is a common name for table games played with a cue and a set of balls, such as snooker or pool and their variants. The goals of the games vary, but the main idea is common to all: the player uses the cue to stroke the cue ball aiming at another ball to achieve the

Table 7
Confusion matrices for QUIN and Padé on test data sets.

$f(x,y)$	$\frac{\partial f}{\partial x}$	QUIN $+ : \circ : -$	Padé $+ : \circ : -$	$\frac{\partial f}{\partial y}$	QUIN $+ : \circ : -$	Padé $+ : \circ : -$
(a) Noiseless data with only the relevant attributes						
$x^2 - y^2$	$+$	482 : 17 : 0	498 : .. : 2	$+$	500 : 0 : 0	493 : .. : 7
	$-$	0 : 62 : 437	2 : .. : 497	$-$	23 : 0 : 476	2 : .. : 497
$x^3 - y$	$+$	1000 : 0 : 0	1000 : .. : 0	$+$	0 : 0 : 0	0 : .. : 0
	$-$	0 : 0 : 0	0 : .. : 0	$-$	0 : 1000 : 0	282 : .. : 717
xy	$+$	495 : 0 : 4	497 : .. : 1	$+$	500 : 0 : 0	495 : .. : 4
	$-$	1 : 0 : 498	3 : .. : 496	$-$	9 : 0 : 490	4 : .. : 495
(b) Data with Gaussian noise added to the function value, $\sigma = 10$						
$x^2 - y^2$	$+$	497 : 2 : 0	481 : .. : 18	$+$	494 : 0 : 0	483 : .. : 10
	$-$	30 : 28 : 440	10 : .. : 489	$-$	25 : 0 : 479	28 : .. : 476
$x^3 - y$	$+$	1000 : 0 : 0	1000 : .. : 0	$+$	0 : 0 : 0	0 : .. : 0
	$-$	0 : 0 : 0	0 : .. : 0	$-$	0 : 1000 : 0	290 : .. : 709
xy	$+$	496 : 0 : 8	484 : .. : 21	$+$	171 : 320 : 8	469 : .. : 30
	$-$	0 : 0 : 494	20 : .. : 474	$-$	0 : 0 : 499	5 : .. : 494
(c) Data with three additional irrelevant variables						
$x^2 - y^2$	$+$	103 : 0 : 399	487 : .. : 15	$+$	175 : 0 : 321	490 : .. : 6
	$-$	106 : 0 : 390	4 : .. : 492	$-$	178 : 0 : 324	4 : .. : 498
$x^3 - y$	$+$	1000 : 0 : 0	1000 : .. : 0	$+$	0 : 0 : 0	0 : .. : 0
	$-$	0 : 0 : 0	0 : .. : 0	$-$	0 : 1000 : 0	339 : .. : 660
xy	$+$	0 : 343 : 159	499 : .. : 3	$+$	502 : 0 : 0	494 : .. : 8
	$-$	0 : 342 : 154	5 : .. : 491	$-$	108 : 0 : 388	9 : .. : 487



(a) A stroke with azimuth 0. The cue ball hits the black one with a *full hit*.

(b) A stroke with azimuth > 0 causes a so called *thin hit*.

Fig. 6. Strokes with different fullness of hit (azimuth).

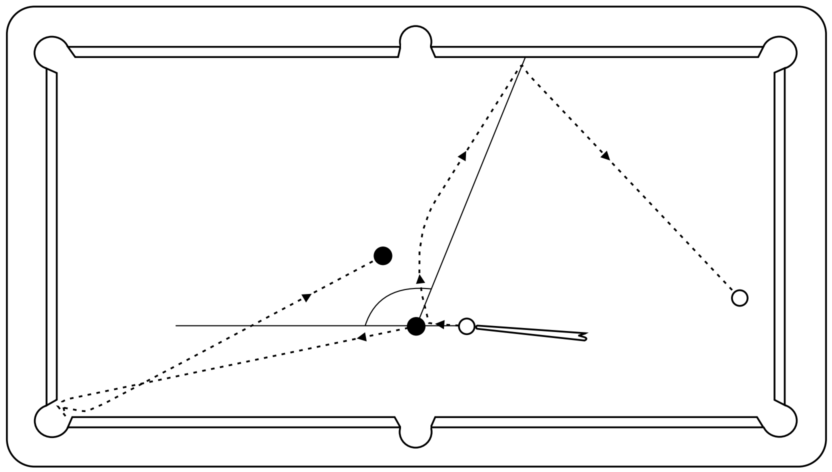


Fig. 7. Example trace from the simulator.

desired effect. The friction between the table and the balls, the spin of the cue ball and the collision of the balls combine into a very complex physical system [39]. However, an amateur player can learn the basic principles of how to stroke the cue ball without knowing much about the physics behind it.

Our goal is to induce a qualitative model describing the relation between the azimuth (degree of fullness of hit; see Fig. 6) and the reflection angle of the cue ball after the collision with another ball. We define the reflection angle as the

Table 8
The observed variables in the billiards case study.

Name	Range	Description
Azimuth	[−15, 15] deg	The horizontal angle of the cue
Elevation	[0, 30] deg	The vertical angle of the cue
Velocity	[3, 5] m/s	Velocity of the cue
Follow	[−0.5*r*, 0.5*r*]	Forward/backward spin caused by hitting the ball above/below the centre
Angle	[−180, 180) deg	The reflection angle of the cue ball

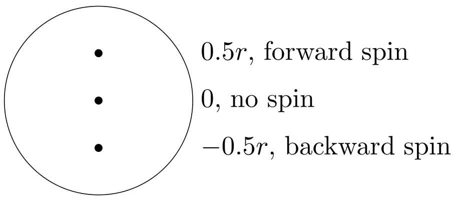


Fig. 8. Spinning the cue ball with a follow (above centre) or a draw (below centre). Dots represent the point at which the cue ball is hit with the cue.

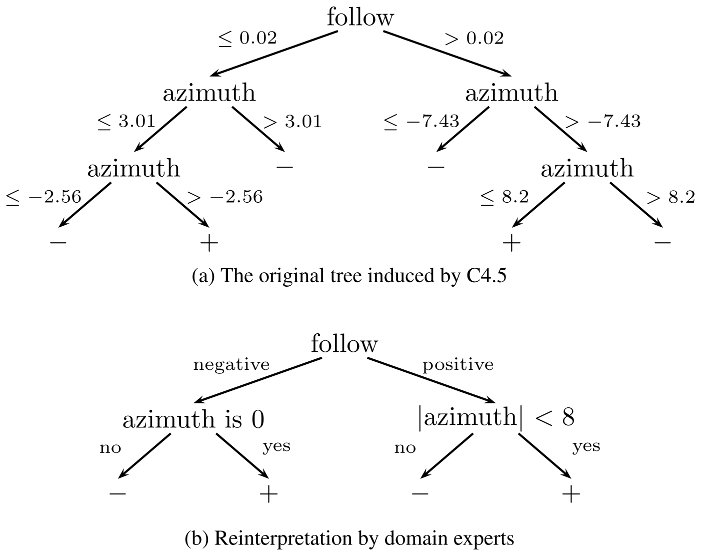


Fig. 9. The qualitative model describing the relation between the reflection angle and azimuth.

angle between the stroke direction and the line connecting the positions of the cue ball's collision with the black ball and its next collision, either with the cushion or with the black ball again (Fig. 7).

We model the relation between the azimuth and the angle as it depends on four attributes (Table 8): the horizontal and the vertical angle of the cue, called *azimuth* and *elevation*, respectively, as well as the *follow* (forward/backward spin; see Fig. 8) of the cue ball and the *velocity* of the stroke.

We used the billiards simulator [40] to collect a sample of 5000 randomly chosen scenarios. We chose to use τ-regression with parameters as usual, $\kappa = 100$, $k = 20$. We induced a qualitative tree using C4.5 and asked the experts for their interpretation of the model.

The tree induced by C4.5 is shown in Fig. 9(a). The experts found the model much easier to understand than a set of equations and the knowledge easier to transfer to the actual game. They were also able to recast the original tree into a simpler model (Fig. 9(b)). As such, it can serve as a conceptualisation of the domain useful for beginners.

The tree first splits on attribute *follow*. For negative values (the cue ball is hit below the centre), the backward spin results in inversely proportional relation between the reflection angle and the *azimuth*, *i.e.*, increasing the *azimuth* decreases the *angle* and vice versa. The exception at the *azimuth* of zero is the artefact of the coordinate system: the reflection angle at this *azimuth* crosses the line of discontinuity at ±180°, so an increase from, say, 179 to 181° is recorded as a decrease from 179 to −179°.

Positive values of *follow* give the cue ball an extra amount of forward spin, which adds to the spin that the cue ball acquires due to rolling. In this case, *azimuth* has a different effect on the black ball. Having a forward spin, the cue ball has a tendency to roll forward after the collision. For *azimuth* approximately between −8 and 8°, increasing (decreasing) the

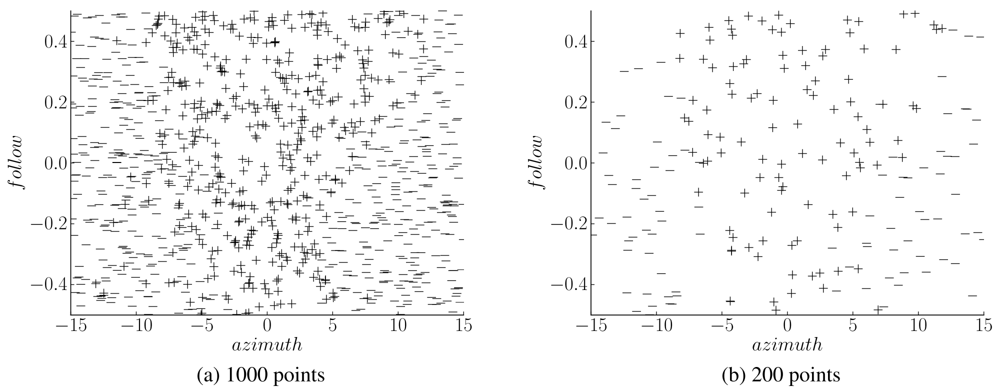


Fig. 10. The plots of *follow–azimuth* for different sample sizes.

azimuth also increases (decreases) the reflection angle. For greater absolute values of *azimuth* the cue ball only partially hits the black ball which causes the reflection angle to become negatively related with the *azimuth*.

Our model also correctly recognised the two important attributes, *follow* and *azimuth*. The other two, *elevation* and *velocity* indeed have no effect on reflection angle in our context, according to the expert opinion.

We have consequently tested the obtained qualitative relations in the simulator by simulating small changes in azimuth while leaving the values of other attributes constant and observing the reflection angle. We found all rules correct apart from some small deviations regarding the numerical values of the splits in the qualitative tree.

Finally, we have used smaller random samples (from 100 to 1000 examples with a step of 100) from this domain to assess the practical usefulness of Padé on sparse data. We first excluded the tree learning algorithm from the experiment by plotting scatter plots in which examples are marked by their respective qualitative derivatives. Scatter plots of *follow–azimuth* show results computed on two random samples with 1000 (Fig. 10(a)) and 200 examples (Fig. 10(b)), which is the smallest sample for which Padé worked reasonably well. However, when we tried to induce qualitative trees from this data, we observed that they become unreliable for samples with less than around 1000 examples.

7. Discussion and conclusion

The three methods for computation of derivatives differ in their theoretical background, which leads to differences in their performance.

LWR is a multivariate method, computing all derivatives at once, while the other two are univariate. This mostly puts LWR at a disadvantage. LWR is more sensitive to the size of data set: if the number of attributes is comparable to (or even greater) than the number of points on which the regression is computed (in our case, this is the number of examples in the neighbourhood, k, and not the total size of the data set), the LWR is solving an ill-conditioned system. This problem clearly surfaces in our experiments with irrelevant attributes. The other two methods take one attribute at a time and do not run into this problem. When the number of attributes is small, LWR however gives satisfactory results.

LWR also fared considerably worse on real-world data sets where we evaluated the methods indirectly by testing their stability. The lower stability of LWR reflects its higher complexity. Based on multivariate regression, the number of parameters that LWR optimises equals the number of attributes. The higher complexity of the LWR method allows it to fit the data better, *i.e.* it has lower bias at the cost of higher variance. Therefore, LWR can in theory better estimate derivatives when compared to the other two, but it can overfit on data with many attributes. This property is also manifested in our experiments, where LWR performed considerably worse on domains with many attributes (housing and imports-85), while it scored best on the five domains which only have two attributes.

Being univariate, the τ-regression and the parallel pairs methods have to select examples lying in the direction of differentiation to exclude the influence of other attributes on function value. To gather the same number of examples as the LWR, they expand the neighbourhood. The τ-regression does so by extending the neighbourhood in direction of the axis of differentiation and the parallel pairs method differentiates along lines parallel to the actual axis of differentiation. The accuracy of computed derivatives should profit from excluding other attributes and suffer from extending the neighbourhood. The effect of this is visible on target functions $x^3 - y$ and $\sin x \sin y$. The former tests how well the method copes with effects of x when differentiating by y; LWR fails here, while parallel pairs work well. The latter concept requires a small neighbourhood, so LWR fares better here, although the difference in performance is not as large as for the former concept.

The difference between the τ-regression and parallel pairs is subtle. The τ-regression computes the derivative inside a somewhat longer tube, whose length is governed by the parameter κ, which we typically set to 50–100. It assumes that the function is linear in a longer region, and the error comes mostly from violating this assumption. Parallel pairs use a smaller vicinity; in most experiments we set κ and k to 20. The function is thus required to be linear only close to $\mathbf{x_0}$. The price to pay is that the derivative is smoothed over a hyper-disc perpendicular to the axis of differentiation instead of being computed at $\mathbf{x_0}$, as in τ-regression. This may cause problems if the derivative strongly depends upon other arguments of the function. We were however unable to experimentally find any practical consequences of this theoretical difference between the τ-regression and parallel pairs.

As a general rule, LWR should be preferred for data sets with a small number of attributes (e.g. up to 5) and a sparse sample. On the other hand, τ-regression should be used with high dimensional data and a dense sample. Parallel pairs are most appropriate when we assume that the attributes have very different influence on the class variable.

Experiments demonstrate that while computation of partial derivatives is often difficult and prone to errors, even quite unreliable estimates usually result in correct qualitative models. Even a modest learning algorithm such as C4.5 is able to reduce the noise in the computed derivatives and build models which almost perfectly match the target function.

Apart from ignoring the noisy labels, the C4.5 task was rather trivial for most artificial domains as it required inducing a tree with one or two leaves. It was put to a more serious test in a case study from billiards. It has induced a correct model which also demonstrates the basic advantage of qualitative modelling over standard regression modelling: the model we induced presents the general principles which can be used by a player, while a regression model would describe complex numerical relationships which would probably be difficult to comprehend and certainly impossible to use at the billiards table.

We also experimented with other machine learning algorithms (rule learning, naive Bayesian classifier) and got similar results. Since C4.5 proved adequate for our needs, we have not systematically tested and published the behaviour of Padé in combination with any other algorithm.

While all presented models are rather small and simple, real-world qualitative models induced by experts, such as Samuelson [3], Jeffries [1] and May [2], do not get any more complex than this neither. It may be that either the world is simpler than expected when described qualitatively, or that humans are used to reason using simple models which skip over the details, which are included only in specific models for particular contexts.

In summary, we introduced a new approach to learning qualitative models which differs from existing approaches by its trick of translating the learning problem into a classification problem and then applying the general-purpose learning methods to solve it. To focus the paper, we mostly explored the first step which involves the estimation of partial derivatives. The second step opens a number of other interesting research problems not explicitly discussed here. One is exploration of other, more powerful learning algorithms. Is our above assumption of the simplicity of the world in qualitative terms correct, or would using, say, support vector machines, result in much better models? Another interesting question is how to combine models for derivatives with respect to individual arguments into a single model describing the function's behaviour with respect to all arguments. One approach may be to use multi-label learning methods to model all derivatives at once. We leave these questions open for further research in the area.

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. We also thank Tadej Janež, Lan Žagar and Jure Žbontar for expert help in billiards case study.

References

[1] C. Jeffries, Qualitative stability and digraphs in model ecosystems, Ecology 55 (6) (1974) 1415–1419.
[2] R.M. May, Qualitative stability in model ecosystems, Ecology 54 (3) (1973) 638–641.
[3] P.A. Samuelson, Foundations of Economic Analysis, enlarged edition, Harvard University Press, 1983.
[4] H.A. Simon, On the definition of the causal relation, The Journal of Philosophy 49 (1952) 517–528;
Reprinted in H.A. Simon, Models of Discovery, D. Reidel, Boston, 1977.
[5] H.A. Simon, A. Ando, Aggregation of variables in dynamic systems, Econometrica 29 (1961) 111–138.
[6] K. Forbus, Qualitative process theory, Artificial Intelligence 24 (1984) 85–168.
[7] J. de Kleer, J.S. Brown, A qualitative physics based on confluences, Artificial Intelligence 24 (1984) 7–83.
[8] B. Kuipers, Qualitative simulation, Artificial Intelligence 29 (1986) 289–338.
[9] J. Kalagnanam, H.A. Simon, Y. Iwasaki, The mathematical bases for qualitative reasoning, IEEE Intelligent Systems 6 (2) (1991) 11–19.
[10] J. Kalagnanam, Qualitative analysis of system behaviour, Ph.D. thesis, Pittsburgh, PA, USA, 1992.
[11] J. Kalagnanam, H.A. Simon, Directions for qualitative reasoning, Computational Intelligence 8 (2) (1992) 308–315.
[12] E. Coiera, Generating qualitative models from example behaviours, Tech. rep. 8901, University of New South Wales, 1989.
[13] D. Hau, E. Coiera, Learning qualitative models of dynamic systems, Machine Learning Journal 26 (1997) 177–211.
[14] S. Ramachandran, R. Mooney, B. Kuipers, Learning qualitative models for systems with multiple operating regions, in: Working Papers of the 8th International Workshop on Qualitative Reasoning about Physical Systems, Japan, 1994.
[15] B. Richards, I. Kraan, B. Kuipers, Automatic abduction of qualitative models, in: Proceedings of the National Conference on Artificial Intelligence, AAAI/MIT Press, 1992.
[16] A. Say, S. Kuru, Qualitative system identification: deriving structure from behavior, Artificial Intelligence 83 (1996) 75–141.
[17] I. Bratko, S. Muggleton, A. Varšek, Learning qualitative models of dynamic systems, in: Proceeding of the 1st International Workshop on Inductive Logic Programming, ILP 91, 1991, pp. 207–224.
[18] G.M. Coghill, S.M. Garrett, R.D. King, Learning qualitative models in the presence of noise, in: Proceedings of the QR'02 Workshop on Qualitative Reasoning Workshop, 2002.
[19] G.M. Coghill, A. Srinivasan, R.D. King, Qualitative system identification from imperfect data, Journal of Artificial Intelligence Research 32 (1) (2008) 825–877.

[20] S. Džeroski, L. Todorovski, Discovering dynamics: from inductive logic programming to machine discovery, Journal of Intelligent Information Systems 4 (1995) 89–108.
[21] S. Džeroski, L. Todorovski, Discovering dynamics, in: International Conference on Machine Learning, 1993, pp. 97–103.
[22] L. Todorovski, Using domain knowledge for automated modeling of dynamic systems with equation discovery, Ph.D. thesis, University of Ljubljana, Ljubljana, Slovenia, 2003.
[23] L. Todorovski, S. Džeroski, A. Srinivasan, J. Whiteley, D. Gavaghan, Discovering the structure of partial differential equations from example behavior, in: Proceedings of the Seventeenth International Conference on Machine Learning, 2000.
[24] H. Kay, B. Rinner, B. Kuipers, Semi-quantitative system identification, Artificial Intelligence 119 (2000) 103–140.
[25] R.K. Gerçeker, A. Say, Using polynomial approximations to discover qualitative models, in: Proceedings of the 20th International Workshop on Qualitative Reasoning, Hanover, New Hampshire, 2006.
[26] I. Bratko, I. Mozetič, N. Lavrač, KARDIO: a Study in Deep and Qualitative Knowledge for Expert Systems, MIT Press, Cambridge, MA, USA, 1989.
[27] I. Mozetič, Learning of qualitative models, in: EWSL, 1987, pp. 201–217.
[28] I. Mozetič, The role of abstractions in learning qualitative models, in: Proceedings of the Fourth International Workshop on Machine Learning, Morgan Kaufmann, 1987.
[29] D. Šuc, I. Bratko, Induction of qualitative trees, in: L. De Raedt, P. Flach (Eds.), Proceedings of the 12th European Conference on Machine Learning, Springer, Freiburg, Germany, 2001, pp. 442–453.
[30] D. Šuc, Machine Reconstruction of Human Control Strategies, Frontiers in Artificial Intelligence and Applications, vol. 99, IOS Press, Amsterdam, The Netherlands, 2003.
[31] I. Bratko, D. Šuc, Learning qualitative models, AI Magazine 24 (4) (2003) 107–119.
[32] A. Varsek, Qualitative model evolution, in: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, 1991, pp. 1311–1316.
[33] W. Pang, G. Coghill, An immune-inspired approach to qualitative system identification of biological pathways, Natural Computing (2010) 1–19, doi:10.1007/s11047-010-9212-2.
[34] C. Atkeson, A. Moore, S. Schaal, Locally weighted learning, Artificial Intelligence Review 11 (1997) 11–73.
[35] S. Ben-David, U. von Luxburg, D. Pal, A sober look at clustering stability, in: 19th Annual Conference on Learning Theory, Springer, Berlin, Germany, 2006, pp. 5–19.
[36] J. Demšar, B. Zupan, G. Leban, T. Curk, Orange: from experimental machine learning to interactive data mining, in: Proceedings of PKDD, 2004, pp. 537–539.
[37] A. Asuncion, D.J. Newman, UCI machine learning repository, 2007.
[38] D. Šuc, D. Vladušič, I. Bratko, Qualitatively faithful quantitative prediction, Artificial Intelligence 158 (2) (2004) 189–214.
[39] D.G. Alciatore, The Illustrated Principles of Pool and Billiards, first edition, Sterling, 2004.
[40] D. Papavasiliou, Billiards manual, Tech. rep., 2009, http://www.nongnu.org/billiards/.