Active Learning of Qualitative Models with Padé

Jure Žabkar, Tadej Janež, Martin Možina and **Ivan Bratko**

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

Abstract

This paper presents a new approach to learning qualitative models, namely *active learning of qualitative models*. We combine a traditional approach of learning qualitative models from numerical data with the active learning paradigm. In general, active learning is useful when labeling learning examples is expensive or difficult. Selecting a representative subset of learning examples is thus extremely important in order to learn a good model. While learning qualitative models by approximating partial derivatives of output variable w.r.t. the selected input variable, we can use the *ceteris paribus* assumption to choose the next learning example for which we demand a label. In other words, the next most useful example is the one that changes the value in the direction of differentiation, other variables being equal. We propose a new method, Active Padé, for active learning of qualitative models. Preliminary experiments on an artificial data set show that our algorithm can learn quite accurate models with a small subset of labeled examples, even on noisy domains.

1 Introduction

Learning qualitative models from numerical data is becoming an increasingly more important area of qualitative modeling. It is useful in many situations. For example, it can be used to learn simple physical laws about concepts such as *centripetal force*, *gravitation* and *pendulum* from given learning examples randomly sampled from the attribute space (Zabkar et al. 2010). Furthermore, such models are usually simpler and easier to understand by humans. An example of such a model is the qualitative model describing the relation between the period T of a pendulum, its length l and the gravitational acceleration g : $T = Q(+l, -g)$. It tells us that the period increases with l and decreases with g.

All the previously mentioned qualitative models describing simple physical laws were built using a large number of training examples (usually thousands of examples). This is fine if obtaining the examples is fast and cheap, however, that may not always be the case. Imagine that we have an autonomous robot trying to induce a qualitative model describing the simple physical law of a *pendulum* we mentioned before. It is given a set of actions that allow it to change the following attributes: the length of the rope l, the mass of the bob m and the initial angle φ. The robot has no prior knowledge regarding the relations between the given attributes and starts collecting examples by performing experiments and measuring the results (the period T). For each triplet of values (l, m, φ) for which it wants to know the value of T, it has to set up a new experiment with the desired values of l, m and φ and measure the result. This process can quickly become very time consuming and inefficient.

This leads us to propose a new approach to qualitative learning, namely *active learning of qualitative models*. *Active learning* is a sub-field of machine learning built on the assumption that, if the learning algorithm is allowed to choose the data from which it learns, it will perform better with less labeled training instances (Settles 2009). The active learner may ask queries in the form of unlabeled instances to be labeled by an oracle (e.g. a human supervisor) or perform the experiment by itself (e.g. an autonomous robot) to obtain the label. Returning to the autonomous robot example, this would mean that by learning *actively*, the robot would learn better models with less labeled training examples. The robot has access to practically infinite number of unlabeled examples by sampling the attributes l, m and φ from their corresponding domains. The labeling process involves the robot performing the experiment for the given triplet of values (l, m, φ) and measuring the period T. By choosing the most appropriate unlabeled examples (triplets (l, m, φ)), the robot will quickly learn the correct qualitative model.

In this paper we use the same definition of qualitative relations as the one described in (Zabkar et al. 2010). This definition is based on partial derivatives and states the following: a function $f(x_1, \ldots, x_n)$ is in positive (negative) qualitative relation with x_i over a region $\mathcal{R}$ if the partial derivative of f with respect to x_i is positive (negative) over the entire $\mathcal{R}$:

$$f = Q_{\mathcal{R}}(+x_i) \quad \equiv \quad \forall \mathbf{x}_0 \in \mathcal{R} : \frac{\partial f}{\partial x_i}(\mathbf{x}_0) > 0 \qquad (1)$$

and

$$f = Q_{\mathcal{R}}(-x_i) \quad \equiv \quad \forall \mathbf{x}_0 \in \mathcal{R} : \frac{\partial f}{\partial x_i}(\mathbf{x}_0) < 0. \qquad (2)$$

Let $f = f(x, y)$ and $\mathcal{R} = \mathbb{R}^2$. Two examples of such qualitative relations would be:

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

and

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

For the sake of clarity, we usually omit specifying the region when it is obvious from the context.

2 Related Work

Our paper aims to bring together two areas of AI, the learning of qualitative models from numerical data and active learning. Herein we only mention the work that is most relevant to our approach.

There are several approaches to learning qualitative models from data. Most of that work is concerned with learning QDE (Qualitative Differential Equations) models. Examples of such approach are programs GENMODEL (Coiera 1989; Hau and Coiera 1997), MISQ (Ramachandran, Mooney, and Kuipers 1994; Richards, Kuipers, and Kraan 1992), QSI (Say and Kuru 1996), QMN (Dzeroski and Todorovski 1995), LAGRANGE (Dzeroski and Todorovski 1993), LAGRAMGE (Todorovski 2003) and methods using ILP (Inductive Logic Programming) to induce such models (Bratko, Muggleton, and Varšek 1991; Coghill, Garrett, and King 2002; Coghill, Srinivasan, and King 2008). In this paper we only consider learning qualitative models of static systems, for which Padé (Zabkar et al. 2010) and QUIN (Suc and Bratko 2001; Suc 2003; Bratko and Suc 2003) are the most appropriate algorithms. The main difference between the two 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 every possible qualitative monotonicity constraint. Another important difference between the two is that QUIN is limited to building classification trees, whereas Padé can be used with any general-purpose machine learning algorithm.

The proposed Active Padé method belongs to the area of *pool-based active learning*, in which label queries are selected from a large pool of unlabeled examples (Settles 2009). There are several approaches tackling this problem, including uncertainty sampling (Lewis and Gale 1994), query-by-committee (Seung, Opper, and Sompolinsky 1992), variance reduction (Cohn, Atlas, and Ladner 1994), estimated error reduction (Roy and McCallum 2001) and expected model change (Settles, Craven, and Ray 2008). Our approach is different from the aforementioned ones in that, it does not compute a certain measure (e.g., uncertainty, error reduction) for a single unlabeled example, but instead computes Pearson's χ^2 in each leaf of the qualitative tree and chooses the one with the lowest value. In the region described by the leaf, the method demands the most appropriate pair of unlabeled examples from this region according to the assumptions of Padé.

3 Padé Parallel Pairs

Padé is a set of methods for qualitative learning proposed by (Zabkar et al. 2010), which estimate partial derivatives of the target function from training data and use them to induce qualitative models of the target function. All the methods are based on linear regression within local neighborhoods to compute the partial derivatives. Padé differs from other approaches, because it uses a two-step approach to induction of qualitative models. In the first step Padé estimates a partial derivative at each point covered by a learning example. Then it replaces the label of example by the sign of the corresponding partial derivative. In the second step, a general-purpose machine learning algorithm is used to generalize from this relabeled data, resulting in a qualitative model describing the function's behavior over the entire domain. In this paper, we will look at one particular method, *parallel pairs*, and use it as a basis for developing a new active learning method on top of it.

Let $(\mathbf{x}, y)$ denote a learning example, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ represents the values of all attributes and y represents the value of the unknown sampled function, $y = f(\mathbf{x})$. Without loss of generality, let us assume that we are trying to estimate the partial derivative of f w.r.t. x_1, $\partial f/\partial x_1$, at point $\mathbf{x}_0$.

The *parallel pairs* method defines the neighborhood of $\mathbf{x}_0$, $\mathcal{N}(\mathbf{x}_0)$, as the κ nearest examples of $\mathbf{x}_0$. The values of all attributes are normalized with the *linear scaling transform* described in (Pyle 1999) prior to computing the neighborhoods $\mathcal{N}(\cdot)$. The points in $\mathcal{N}(\mathbf{x}_0)$ thus fall into a small hyper-sphere around $\mathbf{x}_0$ and we can assume that $x_{si} \approx x_{0i},\ \forall i \in \{1, \ldots, n\}$, for each example $(\mathbf{x}_s, y_s)$ in $\mathcal{N}(\mathbf{x}_0)$.

According to Taylor's theorem, a differentiable function is approximately linear in the neighborhood of $\mathbf{x}_0$:

$$f(\mathbf{x}_s) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0) \cdot (\mathbf{x}_s - \mathbf{x}_0) + R_2, \qquad (3)$$

where $\nabla f(\mathbf{x}_0)$ represents the vector of partial derivatives of f at point $\mathbf{x}_0$ that we are trying to find and R_2 represents the remainder term. We can solve this task using linear regression by rephrasing (3) as a linear model:

$$y_s = \beta_0 + \boldsymbol{\beta}^T(\mathbf{x}_s - \mathbf{x}_0) + \epsilon_s, \quad \forall (\mathbf{x}_s, y_s) \in \mathcal{N}(\mathbf{x}_0), \quad (4)$$

where the task is to find $\boldsymbol{\beta}$ (and β_0) with the minimal sum of squared errors ϵ_s over $\mathcal{N}(\mathbf{x}_0)$.

Let us now consider a pair of examples $(\mathbf{x}_s, y_s)$, $(\mathbf{x}_t, y_t)$ from $\mathcal{N}(\mathbf{x}_0)$, which are approximately aligned with the x_1 axis, i.e., $\|\mathbf{x}_s - \mathbf{x}_t\| \approx |x_{s1} - x_{t1}|$. If we substitute $\mathbf{x}_s$ and y_s with $\mathbf{x}_t$ and y_t in (4), respectively, and subtract it from the same equation, we get:

$$y_s - y_t = \boldsymbol{\beta}^T(\mathbf{x}_s - \mathbf{x}_t) + (\epsilon_s - \epsilon_t). \qquad (5)$$

Since we selected $\mathbf{x}_s$ and $\mathbf{x}_t$ such that $|x_{s1} - x_{t1}| \gg |x_{si} - x_{ti}|$ for all $i > 1$ and if we also assume that partial derivatives with regard to different arguments (and hence coef. β_i) are comparable in size[1], we get that $|\beta_1||x_{s1} - x_{t1}| \gg |\beta_i||x_{si} - x_{ti}|$ for all $i > 1$. We can thus omit all dimensions but the first one from the scalar product in (5):

$$y_s - y_t \simeq \beta_1(x_{s1} - x_{t1}) + (\epsilon_s - \epsilon_t) \qquad (6)$$

[1]Implications of this assumption and robustness of the *parallel pairs* method in case when this assumption is violated are described in (Zabkar et al. 2010).

or

$$y_{(s,t)} \simeq \beta_1 x_{(s,t)1} + \epsilon_{s,t}, \quad (7)$$

where $y_{(s,t)} = y_s - y_t$ and $x_{(s,t)1} = x_{s1} - x_{t1}$. The difference $y_s - y_t$ is therefore linear with the difference in the attribute values x_{s1} and x_{t1}. Coefficient β_1 approximates the derivative $\partial f/\partial x_1(\mathbf{x_0})$.

To compute the derivative using (6), we compute the alignment α of each pair of examples $(\mathbf{x}_s, y_s)$, $(\mathbf{x}_t, y_t)$ from $\mathcal{N}(\mathbf{x}_0)$ with the x_1 axis using a scalar product with the base vector $\mathbf{e}_1$:

$$\alpha_{(s,t)} = \left| \frac{(\mathbf{x}_s - \mathbf{x}_t)^T \mathbf{e}_1}{\|\mathbf{x}_s - \mathbf{x}_t\| \|\mathbf{e}_1\|} \right| = \frac{|x_{s1} - x_{t1}|}{\|\mathbf{x}_s - \mathbf{x}_t\|}. \quad (8)$$

We select the k best aligned pairs from κ points in $\mathcal{N}(\mathbf{x}_0)$ and assign them weights w corresponding to the alignment, i.e., the bigger the alignment the bigger the weight:

$$w_{(s,t)} = e^{-(1-\alpha_{(s,t)})^2/\sigma^2}, \quad (9)$$

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

The derivative $\partial f/\partial x_1(\mathbf{x}_0) = \beta_1$ is then computed using univariate locally weighted linear regression:

$$\beta_1 = \frac{\sum_{(\mathbf{x}_s,\mathbf{x}_t)\in\mathcal{N}(\mathbf{x}_0)} w_{(s,t)}(x_{s1} - x_{t1})(y_s - y_t)}{\sum_{(\mathbf{x}_s,\mathbf{x}_t)\in\mathcal{N}(\mathbf{x}_0)} w_{(s,t)}(x_{s1} - x_{t1})^2}. \quad (10)$$

4 Active Learning with Padé

In the previous section we saw how the *Padé parallel pairs* method uses pairs of examples aligned with the axis of derivation in the local neighborhood of the given point to approximate the partial derivative in this point. It assumes that all the training points are given and fixed and tries to find k best aligned pairs in each local neighborhood of the points. However, if we put this method in the context of active learning, we are given the option of choosing for which unlabeled examples we want to make label queries.

The intuition for the new active learning method is therefore the following: make label queries for the pairs of examples that are *close* to each other and *well aligned* with the axis of derivation. Then use the *parallel pairs* method to compute the partial derivatives in these points.

4.1 Outline of the algorithm

We developed a new active learning method named Active Padé that makes label queries about unlabeled examples in the aforementioned way. In this section we will present the outline of the proposed method.

Let $\mathcal{U}$ be the set of currently unlabeled examples, $\mathcal{L}$ the set of currently labeled examples and $\mathcal{Q}$ the set of currently qualitatively labeled examples. Let i be the iteration counter. Without loss of generality, let us assume that we are computing the partial derivative $\partial f/\partial x_1$. The Active Padé algorithm is the following:

1 $i \leftarrow 0$
2 **while** $i < init\_iter$ **do**
3 choose a *good* pair of examples $\mathbf{x_s}, \mathbf{x_t}$ among all examples in $\mathcal{U}$ and make queries for their respective labels y_s, y_t
4 remove the newly labeled examples $(\mathbf{x_s}, y_s)$ and $(\mathbf{x_t}, y_t)$ from $\mathcal{U}$ and add them to $\mathcal{L}$
5 compute the partial derivative $\partial f/\partial x_1$ along the pair of points $(\mathbf{x_s}, y_s)$ and $(\mathbf{x_t}, y_t)$
6 relabel both examples with the sign of $\partial f/\partial x_1$ and add them to set $\mathcal{Q}$
7 use $\mathcal{Q}$ to induce the classification tree model $\mathcal{T}_i$ in the current iteration
8 $i \leftarrow i + 1$
9 **end while**
10 **while** $i < max\_iter$ **do**
11 compute the *reliability* of each leaf of $\mathcal{T}_{i-1}$ (class. tree from the previous iteration) using Pearson's χ^2 test statistic
12 select all unlabeled examples $\mathcal{U}_\ell$ that fall into the most *unreliable* leaf ℓ
13 choose a *good* pair of examples $\mathbf{x_s}, \mathbf{x_t}$ among the examples in $\mathcal{U}_\ell$ and make queries for their respective labels y_s, y_t
14 remove the newly labeled examples $(\mathbf{x_s}, y_s)$ and $(\mathbf{x_t}, y_t)$ from $\mathcal{U}$ and add them to $\mathcal{L}$
15 use the *Padé parallel pairs* method on set $\mathcal{L}$ to obtain the current set of qualitatively labeled examples $\mathcal{Q}$
16 use $\mathcal{Q}$ to induce the classification tree model $\mathcal{T}_i$ in the current iteration
17 $i \leftarrow i + 1$
18 **end while**

The presented algorithm consists of two parts, the *initialization loop* (first while loop) and the *main loop* (second while loop). We present the details in sections 4.2 and 4.3, respectively.

4.2 Initialization loop

The first loop of Active Padé algorithm (lines 2-9) is called the *initialization loop*, because it serves as a preparation stage for the later *main loop*. Its length is controlled by the $init\_iter$ parameter.

The *initialization loop* differs from the *main loop* in that it does not use the classification tree model $\mathcal{T}_{i-1}$ from the previous iteration as an input in deciding for which unlabeled examples to make label queries in the next step. Instead, it chooses *good* pairs of examples from the whole pool of unlabeled examples.

The example choosing process described in line 3 starts by creating a buffer $\mathcal{B}$ with unlabeled example pairs and their corresponding alignments with the x_1 axis. Instead of computing the alignment of all pairs of examples in $\mathcal{U}$, we only consider pairs of examples that are *close* to each other. We enforce this by iterating over all remaining unlabeled examples $\mathbf{x}_u$ and only computing the alignment of pairs $\mathbf{x}_s$, $\mathbf{x}_t$ that lie in $\mathcal{N}(\mathbf{x}_u)$. The size of $\mathcal{N}(\mathbf{x}_u)$ is controlled by the parameter κ_{init}. Only computing the alignments of pairs that are *close* to each other gives us two advantages. The first one is that this reduces the amount of alignment computations from $\mathcal{O}(|\mathcal{U}|^2)$ to $\mathcal{O}(\kappa_{init}\,|\mathcal{U}|)$. The second one is that this keeps the pairs in accordance with the locality principle inherent in the computation of partial derivatives with the *parallel pairs* method.

The alignment α of each example pair $\mathbf{x}_s, \mathbf{x}_t$ in $\mathcal{B}$ is computed as in (8). The weights w of each pair are computed as follows:

$$w_{(s,t)} = (1 - \alpha_{(s,t)})^{-2} / w_{max}, \tag{11}$$

where w_{max} is set so that the biggest weight equals 1.0. Note that this time we use a different formula to compute the weights. The formula from (9) is more lenient with pairs with lower alignment, while the formula from (11) penalizes misaligned pairs more severely. We want this type of weights to increase the probability of selecting a well aligned example pair.

Next we normalize the weights of all example pairs in $\mathcal{B}$ so that their sum equals 1. We would like to make a weighted random choice of the next example pair from $\mathcal{B}$ for which to make the label query.

Let us define a discrete random variable B that takes example pairs from $\mathcal{B}$ as its values and define $P(B = \mathbf{x}_s, \mathbf{x}_t) = w_{(s,t)}$. Using the *Inversion by sequential search* algorithm described in (Devroye 1986), we make a random sample of size 1 from B. The obtained value of B is the pair of examples $\mathbf{x}_s, \mathbf{x}_t \in \mathcal{U}$ for which we make label queries in the current iteration of the Active Padé method to obtain their respective function values y_s, y_t. Examples $(\mathbf{x}_s, y_s), (\mathbf{x}_t, y_t)$ are then removed from $\mathcal{U}$ and added to $\mathcal{L}$ (line 4).

In line 5 we compute the partial derivative $\partial f / \partial x_1$ along the newly labeled pair of points $(\mathbf{x}_s, y_s), (\mathbf{x}_t, y_t)$. Because the labeled examples in the *initialization loop* occur in pairs that are few and far between, we set the neighborhood of each example to only include itself and the other example of the pair, i.e., $\mathcal{N}(\mathbf{x}_s) = \mathcal{N}(\mathbf{x}_t) = \{(\mathbf{x}_s, y_s), (\mathbf{x}_t, y_t)\}$. This simplifies the computation of β_1 in (10) to:

$$\beta_1 = \frac{(x_{s1} - x_{t1})(y_s - y_t)}{(x_{s1} - x_{t1})^2}. \tag{12}$$

The computed β_1 is the approximation of both $\partial f / \partial x_1(\mathbf{x}_s)$ and $\partial f / \partial x_1(\mathbf{x}_t)$.

In the next step (line 6), we relabel both examples $(\mathbf{x}_s, y_s)$, $(\mathbf{x}_t, y_t)$ to $(\mathbf{x}_s, q)$, $(\mathbf{x}_t, q)$, where $q \in \{+, -\}$ denotes the sign of the computed partial derivative β_1, and add them to the set $\mathcal{Q}$.

Then we use $\mathcal{Q}$ as the training set for the *C4.5* algorithm (Quinlan 1993) to induce the classification tree model $\mathcal{T}_i$ in the current iteration (line 7). Note that $\mathcal{T}_i$ is induced from scratch, i.e., independently from all the previously induced classification trees.

Lastly, we increment the iteration counter i by 1 (line 8).

4.3 Main loop

The second while loop of the algorithm (lines 10-18) is called the *main loop*. As mentioned in the previous section, its main difference with the *initialization loop* is that it uses the classification tree $\mathcal{T}_{i-1}$ trained in the previous iteration as an input in deciding for which unlabeled examples to make label queries in the current step.

We start by computing the *reliability* of each leaf ℓ of $\mathcal{T}_{i-1}$ using Pearson's χ^2 test statistic (line 11). Let $\mathcal{Q}_\ell$ be the set of qualitatively labeled examples that correspond to leaf ℓ. Let O_+ and O_- be the frequencies of examples in $\mathcal{Q}_\ell$ that have labels $+$ and $-$, respectively. Let us define $E_+ = E_- = |\mathcal{Q}_\ell|/2$. The value of Pearson's χ^2 test statistic X^2 for ℓ can then be computed as:

$$X^2 = \frac{(O_+ - E_+)^2}{E_+} + \frac{(O_- - E_-)^2}{E_-}, \tag{13}$$

where O_+ and O_- represent the observed frequencies of examples with labels $+$ and $-$, respectively. Labels E_+ and E_- represent the expected frequencies of examples with labels $+$ and $-$, respectively. The value of X^2 asymptotically approaches the χ^2 distribution.

Pearson's χ^2 is, among other things, used to test the goodness of fit, which establishes whether or not an observed frequency distribution differs from an expected distribution. In our case, we set the expected frequency distribution to be $<|\mathcal{Q}_\ell|/2, |\mathcal{Q}_\ell|/2>$, i.e., we expect half of the examples from ℓ to have label $+$ and half of them to have label $-$.

The null hypothesis of this test states that the frequency distribution of examples observed in a sample (leaf ℓ) is consistent with the given expected (theoretical) distribution ($<|\mathcal{Q}_\ell|/2, |\mathcal{Q}_\ell|/2>$). A higher value of X^2 from (13) therefore corresponds to a stronger rejection of the null hypothesis.

This gives us a criterion for evaluating the *reliability* of leaves from $\mathcal{T}_{i-1}$. We define the *reliability* ρ of leaf ℓ as:

$$\rho_\ell = X^2, \tag{14}$$

where X^2 represents the value of Pearson's χ^2 test statistic as computed in (13). Leaves with a higher value of ρ will more strongly reject the null hypothesis, while leaves with a small value of ρ may not be able to reject it at all.

If we now look back at how we defined the expected frequency distribution, we can interpret the null hypothesis in a new way. We set the expected frequency distribution to $<|\mathcal{Q}_\ell|/2, |\mathcal{Q}_\ell|/2>$, which corresponds to the maximally unreliable leaf, i.e., a leaf with half of examples belonging to one class and the other half to the other class. Thus we can interpret the null hypothesis as stating that the frequency distribution of examples observed in a leaf is consistent with the expected distribution of the maximally unreliable leaf. Therefore, the leaves that have a higher value of ρ more substantially differ from the maximally unreliable leaf, while the leaves with a small value of ρ may not differ from it at all.

In line 12 we select the most unreliable leaf ℓ as the one with the lowest ρ value. Then we create a set $\mathcal{U}_\ell$ with all the remaining unlabeled examples that correspond to the selected leaf ℓ.

The selection of a *good* example pair in line 13 is akin to the one in line 3. The only difference is that now we do not consider all the examples from $\mathcal{U}$, but rather only focus on the ones in $\mathcal{U}_\ell$.

After selecting a *good* pair of examples $\mathbf{x}_s, \mathbf{x}_t$ from $\mathcal{U}_\ell$, we query for their respective labels y_s, y_t, remove them from $\mathcal{U}$ and add them to $\mathcal{L}$ (line 14).

The next step (line 15) uses the *Padé parallel pairs* method described in section 3 to compute the partial derivatives $\partial f / \partial x_1$ in the points from $\mathcal{L}$. The *parallel pairs*

method uses the parameter κ to control the size of the neighborhoods for computing partial derivatives. As described in (Zabkar et al. 2010), the size of the neighborhoods should reflect the density of examples and the amplitude of noise. While we expect the latter not to change, the density of examples increases as the number of labeled examples in $\mathcal{L}$ increases.

Therefore, instead of setting κ to a predefined value, we introduce a new parameter η, which regulates κ in relation to the current size of $\mathcal{L}$. We set $\kappa = \lceil \eta |\mathcal{L}| \rceil$, which means that we linearly increase κ as the number of labeled examples in $\mathcal{L}$ increases. To avoid senseless neighborhoods of size 1, we further enforce that $\kappa \geq 2$.

After computing the partial derivatives, we relabel all examples $(\mathbf{x}, y)$ from $\mathcal{L}$ to $(\mathbf{x}, q)$, where q denotes the sign of the computed partial derivative β_1 from (10), and add them to the set $\mathcal{Q}$.

Identically to the lines 7-8 of the *initialization loop*, we use $\mathcal{Q}$ to induce the classification tree model $\mathcal{T}_i$ in the current iteration and increment the iteration counter i by 1 in lines 16-17.

5 Experiments

To estimate the empirical performance of the proposed algorithm, we conducted a set of experiments on an artificially constructed problem. We used the function $f(x, y) = x^2 - y^2$, which is a standard test function used in (Zabkar et al. 2010) and (Suc 2003).

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 behavior with respect to y is similar, we only observed the results for $\partial f/\partial x$.

To create the data set, we sampled the function $f(x, y) = x^2 - y^2$ in 2000 points chosen uniformly at random from the range $[-10, 10] \times [-10, 10]$. The data set was then split into a training set of 1000 points and a testing set of 1000 points.

Since we are only interested in the qualitative behavior of the given function, i.e., the sign of the partial derivative $\partial f/\partial x$ (*qualitative derivative*), we relabeled all the testing points with the analytically computed qualitative derivatives in the given points. Therefore, all the points with $x > 0$ were relabeled with a $+$ and all points with $x < 0$ were relabeled with a $-$. We define the accuracy of the learned models as the proportion of testing examples with correctly predicted qualitative derivative.

All the points from the training set had their function value removed and were given to the pool of unlabeled examples.

We compared Active Padé with different parameter settings against choosing examples at *random*, which served us as the baseline for our experiments. For obtaining the accuracy of *random* choosing, we repeated the following three steps every time the set of labeled training examples increased by five:

1 use the *Padé parallel pairs* method to compute the qualitative derivatives for the current set of labeled examples $\mathcal{L}$ producing the current set of qualitatively labeled examples $\mathcal{Q}$

2 use $\mathcal{Q}$ to train the classification tree $\mathcal{T}$ at the current iteration

3 use $\mathcal{T}$ to predict the qualitative derivatives in the points from the testing set to get the final accuracy

In the *parallel pairs* method, we set $\kappa = k = 20$, as this parameter settings were shown to perform well in an extensive set of experiments conducted in (Zabkar et al. 2010).

For Active Padé, we set $\eta = 0.2$ and $k = \kappa$, i.e., k and κ equaled 20% of the size of the data set of currently labeled examples. The $init\_iter$ parameter, which controls the length of the *initialization loop*, was set to 5 and 10. We also tried other values of the $init\_iter$ parameter, but omitted them from the following figures for better clarity of the results, since they did not differ a lot from values of 5 and 10.

Throughout all the methods, we used the classification tree induction algorithm C4.5 as reimplemented in the Orange data mining software (Demsar et al. 2004) with the following custom parameter settings: $minSubset = 5$, $maxDepth = 3$ and $maxMajority = 0.9$. All other parameters were left at their default values.

For evaluating the performance of the methods, we constructed learning curves that depict how the accuracy of the models change with the increased number of labeled examples. This is a standard way of evaluating active learning methods also used in (Roy and McCallum 2001), (Guo and Greiner 2007) and (Settles, Craven, and Ray 2008).

To obtain the learning curve for each method, we measured the accuracy of the current model every fifth time after adding a new example to the labeled set. All the learning curves presented in the following figures represent averages of 10 trials.

Contrary to other evaluations of active learning methods, in our case, the methods were initially not given any labeled examples.

Each figure also contains a solid gray line showing the maximum possible accuracy achieved by using *Padé parallel pairs* method with $\kappa = k = 20$ on the wholly labeled training data set (and using a classification tree afterwards to build the final model).

5.1 Accuracy

We measured the accuracy of Active Padé with several settings of parameters on data without noise. The comparison is shown in Fig. 1.

Active Padé performed quite well, reaching the accuracies above 90% after only 15 labeled examples, almost doubling the accuracy of *random* choosing after the same amount of labeled examples. They retained this gap until *random* choosing reached the same level of accuracy after 30 labeled examples.

We can also observe a small performance hit in all variants of Active Padé. For the method with $init\_iter = 10$, this happens after switching from the *initial loop* to the *main loop* after 20 labeled examples. Note that on each iteration of our method, the algorithm makes label queries for two new unlabeled examples. Thus, after completing 10 iterations, the method has collected 20 labeled examples. However, for the method with $init\_iter = 5$, this phenomenon

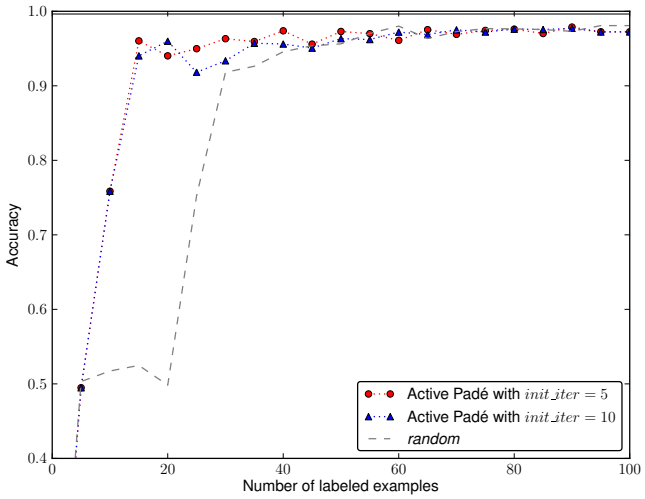

Figure 1: Average test set accuracy for different values of the $init\_iter$ parameter of the Active Padé method and *random* choosing.

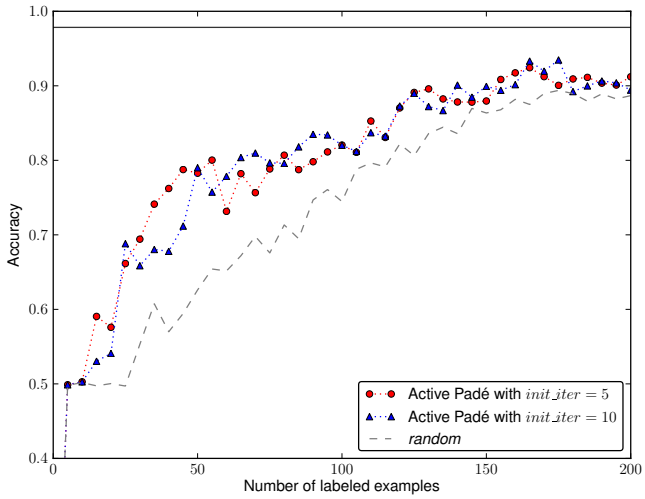

Figure 2: Average test set accuracy for different values of the $init\_iter$ parameter of the Active Padé method and *random* choosing after adding 5 irrelevant attributes.

does not happen after 10 labeled examples, as we would have expected, but rather after collecting 15 labeled examples. As of yet, we cannot explain this small drop in accuracy. Also, this drop in accuracy is almost negligible. All in all, Active Padé with $init\_iter = 5$ performed best on this data set. All the methods were evened out after about 50 labeled examples.

5.2 The effect of Irrelevant Attributes

Many real-world data sets include a large number of attributes that do not effect the function value. The following experiment shows how these irrelevant attributes effect the evaluated active learning methods.

We sampled the function $f(x, y) = x^2 - y^2$ as described above, with the difference that this time we added 5 random attributes sampled from the same interval as x and y, $[-10, 10]$, to the domain. Figure 2 shows the learning curves for this case.

The first thing we noticed is that the methods needed a lot more labeled examples to reach the same level of accuracy as before. Both Active Padé methods needed more than 150 labeled examples to reach accuracies above 90% as opposed to only needing 15 examples to reach the same level of accuracy before (Fig. 1). The *random* choosing did not reach accuracy above 90% after 200 labeled examples.

The observed behavior is due to the fact that finding pairs of unlabeled examples parallel to the axis of partial derivation becomes increasingly harder after inflating the dimensionality of the attribute space by adding more irrelevant attributes to the data set. Similarly, adding more dimensions to the attribute space also effects the *random* choosing. Instead of uniformly sampling 2 dimensions (namely x and y), it now samples 7 dimensions. Thus the probability of choosing good example pairs parallel to the axis of partial derivation also decreases.

Secondly, Active Padé again remained ahead of *random* choosing. Both were very good at the beginning and reached accuracies of 78.3% and 79% after only 50 labeled examples. Meanwhile, *random* choosing needed more than twice as many examples to reach the same level of accuracy. The gap between our method and *random* choosing eventually became smaller and practically vanished after 200 labeled examples. Also worth noting is that the methods never reached the maximum possible accuracy of 97.8%.

5.3 Coping with Noise

Noise in the function value is also a common thing in real-world data. In this experiment we simulated such situation by adding a certain amount 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 range $[-100, 100]$. We added Gaussian noise with a mean of 0 and standard deviation of 10, i.e., distributed as $\mathcal{N}(0, 10^2)$, to the function value. The results are shown on Fig. 3.

The first thing to notice is that applying random noise to the function value did not have dramatic effects on the performance of any method. In fact, the performance of *random* choosing did not change at all from the one on Fig. 1, while Active Padé is affected to various degrees depending on the parameters' settings.

Again, Active Padé performed better than *random* choosing at the beginning. The one with $init\_iter = 5$ only needed 20 labeled examples to reach accuracy above 90%, while *random* choosing reached the same level of accuracy after 30 labeled examples. After 40 labeled examples, all the methods were practically evened out.

Similarly to the first experiment (Fig. 1), Active Padé experienced a drop in accuracy after 20 labeled examples. The drop occurred when Active Padé was already in the *main loop*, meaning that the neighborhood size (κ) increased from

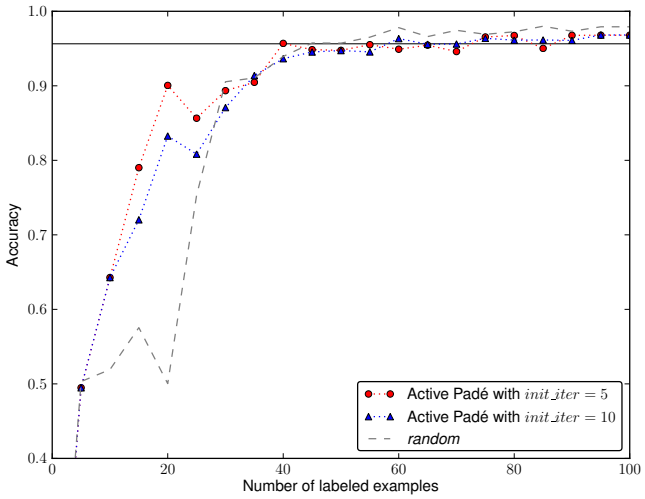

Figure 3: Average test set accuracy for different values of the $init\_iter$ parameter of the Active Padé method and *random* choosing after adding random noise distributed as $\mathcal{N}(0, 10^2)$ to the function value.

4 to 5 (recall that $\eta = 0.2$) labeled examples. This apparently effects the computation of partial derivatives with the *parallel pairs* method and subsequently causes a drop in accuracy of the final qualitative model.

In this experiment, Active Padé with $init\_iter = 5$ outperformed the one with $init\_iter = 10$. This shows that, in the case of noise in the example labels, it is better to switch to the *main loop* as soon as possible. The reason for this lies in the way examples are qualitatively relabeled in the *initialization loop* and in the *main loop*. In the first case, the examples always retain the same qualitative label and newly added example pairs do not effect the previously computed labels. However, in the second case, qualitative labels for all the labeled examples are recomputed on each step. Thus, by adding more example pairs to the labeled data set, we help to improve the accuracy of the computed qualitative labels on the whole data set.

6 Conclusion and Future Work

We have proposed a new approach to learning qualitative models, namely *active learning of qualitative models*, by combining a traditional approach to learning qualitative models from numerical data with the active learning paradigm. Building on the *Padé parallel pairs* method, we developed a new method for active learning of qualitative models called Active Padé. It uses Pearson's χ^2 test statistic as a heuristic for deciding which subspace of the unlabeled pool of examples to explore next.

For empirical evaluation of the proposed method, we conducted a set of experiments on an artificially generated data set $f(x, y) = x^2 - y^2$. We compared our algorithm with various parameter settings to *random* choosing of unlabeled examples. When applied to data without noise and without irrelevant attributes, Active Padé significantly outperformed *random* choosing, needing only half as many labeled examples to reach the accuracies above 90%.

To make the learning task a bit harder, we added five irrelevant (random) attributes to the data set and observed the results. This time all the methods needed a lot more learning examples to reach the accuracy of the Padé parallel pairs method learned on the wholly labeled training data set. The lead of Active Padé was bigger at the beginning and slowly dissolved as the number of labeled examples increased.

In the last experiment, we studied the effect of random Gaussian noise applied to the function value. Interestingly, it did not effect the *random* choosing at all, while the performance of Active Padé slightly decreased. Due to the different nature of qualitative relabeling of examples in the *initialization loop* and the *main loop*, it is better to set the $init\_iter$ parameter to a lower value so that Active Padé switches to the *main loop* sooner.

All in all, the results are promising and we believe our approach is also applicable to other, more complex domains, which include both irrelevant attributes and noise in the class value. We intend to prove that by performing a comprehensive set of experiments on real-word data sets in the future.

References

Bratko, I., and Suc, D. 2003. Learning Qualitative Models. *AI Magazine* 24(4):107.

Bratko, I.; Muggleton, S.; and Varšek, A. 1991. Learning Qualitative Models of Dynamic Systems. In *Proceedings of the First International Workshop on Inductive Logic Programming ILP-91*, 207–224.

Coghill, G. M.; Garrett, S. M.; and King, R. D. 2002. Learning Qualitative Models in the Presence of Noise. In *Proceedings of the 16th International Workshop on Qualitative Reasoning QR-02*.

Coghill, G.; Srinivasan, A.; and King, R. 2008. Qualitative System Identification from Imperfect Data. *Journal of Artificial Intelligence Research* 32(1):825–877.

Cohn, D.; Atlas, L.; and Ladner, R. 1994. Improving Generalization with Active Learning. *Machine Learning* 15(2):201–221.

Coiera, E. 1989. Learning Qualitative Models from Example Behaviours. In *Proceedings of the Third Workshop on Qualitative Physics*.

Demsar, J.; Zupan, B.; Leban, G.; and Curk, T. 2004. Orange: From Experimental Machine Learning to Interactive Data Mining. In *Knowledge Discovery in Databases: PKDD 2004*, 537–539. Springer.

Devroye, L. 1986. *Non-Uniform Random Variate Generation*. Springer-Verlag. chapter 3, 85.

Dzeroski, S., and Todorovski, L. 1993. Discovering Dynamics. In *Proceedings of the 10th International Conference on Machine Learning*, 97–103.

Dzeroski, S., and Todorovski, L. 1995. Discovering Dynamics: From Inductive Logic Programming to Machine Discovery. *Journal of Intelligent Information Systems* 4(1):89–108.

Guo, Y., and Greiner, R. 2007. Optimistic Active Learning using Mutual Information. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 823–829.

Hau, D. T., and Coiera, E. W. 1997. Learning Qualitative Models of Dynamic Systems. *Machine Learning* 26(2):177–211.

Lewis, D. D., and Gale, W. A. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, 3–12. Springer-Verlag New York.

Pyle, D. 1999. *Data Preparation for Data Mining*. Morgan Kaufmann. chapter 7.1.7, 251.

Quinlan, J. 1993. *C 4.5: Programs for Machine Learning*. Morgan Kaufmann.

Ramachandran, S.; Mooney, R. J.; and Kuipers, B. J. 1994. Learning Qualitative Models for Systems with Multiple Operating Regions. In *Proceedings of the 8th International Workshop on Qualitative Reasoning about Physical Systems*, 212–223.

Richards, B. L.; Kuipers, B. J.; and Kraan, I. 1992. Automatic Abduction of Qualitative Models. In *Proceedings of the National Conference on Artificial Intelligence*, 723–723.

Roy, N., and McCallum, A. 2001. Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In *Proceedings of the 18th International Conference on Machine Learning*, 441–448. Morgan Kaufmann, San Francisco, CA.

Say, A. C., and Kuru, S. 1996. Qualitative System Identification: Deriving Structure from Behavior. *Artificial Intelligence* 83(1):75–141.

Settles, B.; Craven, M.; and Ray, S. 2008. Multiple-Instance Active Learning. In *Advances in Neural Information Processing Systems (NIPS)*, 1289–1296. MIT Press.

Settles, B. 2009. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Seung, H. S.; Opper, M.; and Sompolinsky, H. 1992. Query by Committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 287–294. ACM Press New York, NY, USA.

Suc, D., and Bratko, I. 2001. Induction of Qualitative Trees. In *Proceedings of the 12th European Conference on Machine Learning ECML-2001*, 442–453. Springer-Verlag.

Suc, D. 2003. *Machine Reconstruction of Human Control Strategies*. IOS Press.

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

Zabkar, J.; Mozina, M.; Bratko, I.; and Demsar, J. 2010. Learning Qualitative Models from Numerical Data. *Preprint submitted to Artificial Intelligence (2010-03-16), available for download at http://www.ailab.si/jure/.*