Qualitatively Faithful Quantitative Prediction

Dorian Sue, Daniel Vladusic, Ivan Bratko
Faculty of Computer and Information Science, University of Ljubljana
Trzaska 25, 1000 Ljubljana, Slovenia
{dorian.sue, ivan.bratko, daniel.vladusic}@fri.uni-lj.si

Abstract

In this paper we describe a case study in which we applied an approach to qualitative machine learning to induce, from system's behaviour data, a qualitative model of a complex, industrially relevant mechanical system (a car wheel suspension system). The induced qualitative model enables nice causal interpretation of the relations in the modelled system. Moreover, we also show that the qualitative model can be used to guide the *quantitative* modelling process leading to numerical predictions that may be considerably more accurate than those obtained by state-of-the-art numerical modelling methods. This idea of combining qualitative and quantitative machine learning for system identification is in this paper carried out in two stages: (1) induction of qualitative constraints from system's behaviour data, and (2) induction of a numerical regression function that both respects the qualitative constraints and fits the training data numerically. We call this approach Q^2 learning, which stands for Qualitatively faithful Quantitative learning.

1 Introduction

It is generally accepted that qualitative models are easier to understand and reason about than quantitative models. Qualitative models thus provide a better basis for the explanation of phenomena in a modelled system than numerical models. In this paper we describe a case study in which we applied an approach to qualitative machine learning to induce, from system's behaviour data, a qualitative model of a complex, industrially relevant mechanical system (a car wheel suspension system). The induced qualitative model enables nice causal interpretation of the relations among the variables in the system. This is precisely as one would expect from a qualitative model. More surprisingly, however, we also show in this case study that the qualitative model can be used to guide the *quantitative* modelling process that may lead to numerical predictions that are considerably more accurate than those provided by state-of-the-art numerical modelling methods.

Thus the main message of this paper is that a combination of methods for qualitative and quantitative system identification has good chances to attain significant improvements over numerical system identification techniques, including techniques of numerical machine learning methods, such as regression trees [Breiman *et ai,* 1984] and model trees [Quinlan, 1992]. The potential improvements are in two respects: first, the predictions are qualitatively consistent with the properties of the modelled system, and in addition they are also numerically more accurate.

This idea of combining qualitative and quantitative machine learning for system identification is in this paper carried out in two stages. First, induce qualitative constraints from system's behaviour data (training data) with program QUIN (overviewed in Section 3). Second, induce a numerical regression function that both respects the qualitative constraints and fits well the training data numerically (called Qualitative to Quantitative transformation or Q2Q for short, described in Section 4). We call this approach Q^2 learning, which stands for Qualitatively faithful Quantitative learning. To underline the importance of qualitative fidelity, we illustrate in Section 2 some problems that numerical learners typically have in respect of qualitative consistency. In Section 5 we present the case study in applying Q^2 to the chosen problem of modelling car suspension.

There are several approaches to learning qualitative models from numerical data that may support alternative approaches to Q^2 learning. These include the program QMN [Dzeroski and Todorovski, 1995], QSI [Say and Kuru, 1996], SQUID [Kay *et al,* 2000], and QOPH [Coghill *et al,* 2002].

2 Qualitative difficulties of numerical learning

Consider a simple container of cylindrical shape and a drain at the bottom. If we fill the container with water, the water drains out. Water level monotonically decreases, until it reaches zero. Suppose we fill the container with water, and measure initial outflow $\Phi_0 = \Phi(t_0)$ and the time behaviour of water level $h(t)$. Since this is a rather simple behaviour, one would naturally expect that machine learning techniques should be able to fairly well predict time behaviour of water level if enough learning examples are given. Quite surprisingly, even in such simple cases, the usual numerical predictors can give strange and qualitatively unacceptable predictions. We illustrate the problems with a simple experiment, using well-known techniques for numerical prediction: model trees and locally weighted regression.

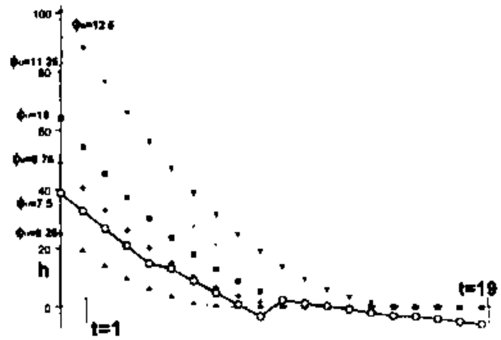

Figure 1: M5 predictions of water outflow: empty circles are M5 predictions of level $h(t)$ on a test set with Φ_0 = 7.5. Other dots are the the learning examples. Note that M5 predicts that water level *increases* at time $t=10$.

In our experiment we used container outflow simulation data to evaluate how different numerical predictors learn the time behaviour of water level. The outflow from a container is $\Phi(t)=c\sqrt{h(t)}$, where *c* is a parameter depending on the area of the drain. For the simulation we used Euler integration $h(t + \Delta t) = h(t) - \Phi(t)\Delta t$, where $\Delta t=0.1$ seconds and $c=1.25$. We used six example sets, generated with different initial water levels and initial outflows $\Phi_0=5 + ic$, $i=1,..,6$. An example set has 20 examples $(t, h(t), \Phi_0)$ corresponding to 19 seconds of simulation.

We used the Weka [Witten and Frank, 2000J implementations of locally weighted regression lAtkeson *et al,* 1997] (LWR, for short), and M5 regression and model trees [Quinlan, 1992] to learn the time behaviour of level given the initial outflow, i.e. $h(t)=f(t, \Phi_0)$. One example set was used as a test set and other five sets (100 examples) for learning. When the test set was the set with $\Phi_0=7.5$, M5 with the default parameters induced a model tree with 9 leaves. Figure 1 shows the learning examples and the M5 prediction of level $h(t)$ on a test set. Note that M5 predicts that water level *increases* at time $t=10$. The same happens if we change the pruning parameter. This is of course qualitatively unacceptable as water level can never increase. Also, there are no learning examples where water level increases.

One might think that this is an isolated weird case or M5 bug. But it is not. LWR makes a similar *qualitative error* on the test set with Φ_0 = 11.25, when it predicts that water level increases at $t=5$. Of course, LWR predictions depend on its parameter k, i.e. the number of neighbors used, but often the appropriate k that doesn't give qualitative errors on one container, gives qualitative errors when learning from similar data but with different area of the drain. Often, the errors are even more obvious if we make predictions at the edges of the space covered by learning examples, i.e. using as test set Φ_0 = 6.25 or Φ_0 = 12.5. As one would expect, regression trees make similar qualitative errors.

We believe that other numerical predictors make similar *qualitative errors,* at least in more complex domains. This might be quite acceptable in applications where we just want to minimize *numerical* prediction errors. But often it is also important to respect qualitative relations that are either given in advance or hidden in the data. Ignoring them results in clearly unrealistic predictions that a domain expert would not approve. The idea of this paper is to use such qualitative relations, either given or induced from data, not only to avoid qualitatively incorrect predictions, but also to improve the accuracy of numerical prediction.

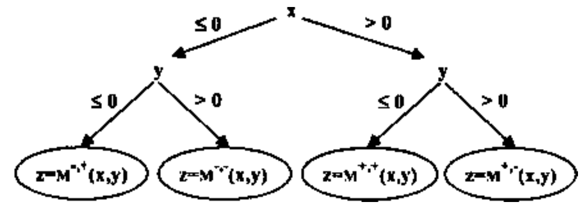

Figure 2: A qualitative tree induced from a set of examples for the function $Z = X^2 - Y^2$. The rightmost leaf, applying when attributes A' and *Y* are positive, says that *Z* is strictly increasing in its dependence on A' and strictly decreasing in its dependence on}'.

3 Qualitative data mining with QUIN

QU1N (Qualitative Induction) is a learning program that looks for qualitative patterns in numerical data [Sue, 2001; Sue and Bratko, 2001; 2002]. Induction of the so-called qualitative trees is similar to induction of decision trees. The difference is that in decision trees the leaves are labelled with class values, whereas in qualitative trees the leaves are labelled with what we call qualitatively constrained functions.

Qualitatively constrained functions (QCFs for short) are a kind of monotonicity constraints that are widely used in the field of qualitative reasoning. A simple example of QCF is: $Y = M^+(X)$. This says that *Y* is a monotonically increasing function of A. In general, QCFs can have more than one argument. For example, $Z = M^{+,-}(X, Y)$ says that *Z* monotonically increases in A and decreases in *Y.* We say that Z is positively related to X and negatively related to Y If both A and *Y* increase, then according to this constraint, *Z* may increase, decrease or stay unchanged. In such a case, a QCF cannot make an unambiguous prediction of the qualitative change in *Z.*

QUIN takes as input a set of numerical examples and looks for qualitative patterns in the data. More precisely, QUIN looks for regions in the data space where monotonicity constraints hold. Such a set of qualitative patterns are represented in terms of a qualitative tree. As in decision trees, the internal nodes in a qualitative tree specify conditions that split the attribute space into subspaces. In a qualitative tree, however, each leaf specifies a QCF that holds among the input data that fall into that leaf. For example, consider a set of data points *(Xy,Z)* where $Z = X^2 - Y^2$ possibly with some noise added. When QUIN is asked to find in these data qualitative constraints on *Z* as a function of X and *Y,* QUIN generates the qualitative tree shown in Figure 2. This tree partitions the data space into four regions that correspond to the four leaves of the tree. A different QCF applies in each of the leaves. The tree describes how *Z* qualitatively depends on A and *Y.*

QUIN constructs a tree in the top-down greedy fashion, similarly to decision tree induction algorithms. A more elaborate description of the QUIN algorithm and its evaluation on a set of artificial domains is given elsewhere [Sue, 2001; Sue and Bratko, 2001]. QUIN has been applied to qualitative reconstruction of human control strategies [Sue, 2001], and to reverse engineer a complex industrial controller for gantry cranes [Sue and Bratko, 2002].

4 Q2Q transformation

In this section we describe the qualitative-to-quantitative transformation (Q2Q for short). Given a set of numerical data and a qualitative tree, Q2Q attempts to find a regression function that fits the data well numerically, and also respects the qualitative constraints in the tree. We say that such a regression function is *qualitatively consistent.* In fact, Q2Q finds a qualitatively consistent regression function with good fit for each leaf in the tree separately. The overall regression function is then obtained by gluing together the regression functions for the leaves.

The Q2Q procedure is as follows. First, we partition the learning examples according to the leaves of the qualitative tree. These subsets are then used for learning qualitatively consistent functions of the corresponding leaves. Let us focus on learning a qualitatively consistent function for some particular leaf. Suppose we have a leaf with the qualitative constraint $y=M^{+}(x)$. We then have to find some monotonically increasing function that fits the data well. One straightforward solution, used in Q2Q, is to divide the range of variable x with a number of equidistant points (i.e. $\{x_1, x_2, ..., x_n\}$) in which we learn from the given data the function values *y.* The result is a set of pairs $\{(x_1, y_1), ..., (x_n, y_n)\}$ that defines a piece-wise linear function which can be easily checked for compliance with the given qualitative constraint. This procedure can be generalized to qualitative constraints of any dimension.

In our implementation the function values *yi* were determined with a standard version of locally weighted regression (LWR) [Atkeson *et ai,* 1997], which used Gaussian weighting function. Therefore, the two parameters of the transformation were the number of equidistant points per dimension $(N \in \{3,4,5,6\})$ and the kernel size of the Gaussian weighting function $(K \in \{0.3, 0.4, .., 0.8\})$. All possible combinations of these two parameters (4*6=24) define the space of all candidate piece-wise linear functions for each leaf. These sets of candidate functions are exhaustively searched by Q2Q to find functions that satisfy given qualitative constraints. For each leaf, Q2Q selects among these qualitatively consistent piece-wise linear functions one that has best fit with the data that fall into this leaf. Quality of the fit is measured with root mean squared error, *RMSE* for short. It is theoretically possible that none of the candidate functions is qualitatively consistent with the QCF in the leaf. In such a case the Q2Q procedure would fail to find a qualitatively consistent regression function. Although in our experiments this never happened, we are working on an improved version of Q2Q that is guaranteed to find a qualitatively consistent regression function.

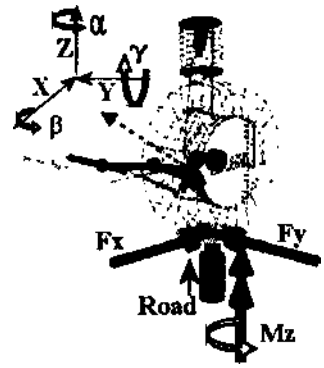

Figure 3: Intec wheel model: wheel position is given by x, y and z coordinates of the wheel center, and rotation angles about axes x, y and z, called enforced wheel-spin angle γ, camber β and toe angle a, respectively. These are affected by horizontal forces F_x and F_y, elevation of the road *R* and rotational moment M_z that act upon the tyre.

5 Intec case study

5.1 Intec wheel model

In this section we present an application of Q^2 learning to the modelling of car wheel suspension system. This is a complex mechanical system of industrial relevance. The model and simulation software used in this experiment were provided by a German car simulation company Intec. The main role of the application in this paper is to provide a controlled experiment to assess the potentials of Q^2 learning on a modelling problem of industrial complexity. However, although the target model was already known and developing such a model was not an issue of practical relevance, initially this case study was nevertheless motivated by a practical objective. Namely, the complexity of Intec's model is so high that it cannot be simulated on the present simulation platform in real time. Therefore the practical objective of the application of Q^2 learning was to speed up the wheel simulation. The goal thus was to obtain a simplified wheel model that would still be sufficiently accurate and at the same time significantly simpler than the original model to allow real-time simulation. Indeed, the simplified model obtained with Q^2 is computationally trivial compared with the original model.

The Intec wheel model (shown in Figure 3) is a multi-body model of a front wheel suspension built in compliance with the physical model assuming no car-body movement and no wheel-spin. In fact, the suspension system is modelled as if the car-body is fixed. Wheel position is given by x, y and z coordinates of the wheel center. Because the flexible joints in multi-body suspension system that links the wheel to the car-body allow several levels of displacements, rotation angles about axes x, y and z are also measured. These are called enforced wheel-spin angle γ, camber β and toe angle α, respectively.

The multi-body simulation software Simpack [Intec, 2002] was used to set up the model and to generate simulation traces. During simulation, a number of elements are acting upon the tyre: two horizontal forces F_x and F_y, vertical movement (measured as elevation of the road *R)* and rota-

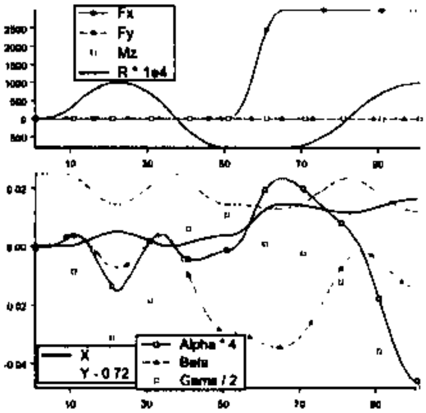

Figure 4: A typical simulation trace of the Intec wheel model: the input variables are on the upper graph, the output variables (except z that changes the same as road R) are on the lower graph, x-axis is time in steps $dt=0.7$ seconds. Note the complex behaviour of the output variables resulting from changes in F_x and road R.

tional moment M_z. For example, F_x is acting upon the tyre when braking, F_y when driving through corners (centripetal force) and rotational moment M_z when parking the car.

During the simulation, input and output variables are logged to a file called simulation trace. We used traces of wheel simulation with different trajectories of input variables. Each trace lasted for 70 seconds, and was sampled with $dt=0.7$ seconds. In this way a trace gives 100 examples, each example contains 10 values, corresponding to the values of four input and six output variables at a given time. Figure 4 shows a typical simulation trace. It should be noted that all these traces correspond to very slow changes of input variables, and as a result the traces are illustrative mainly of the kinematics of the mechanism, but not also of its dynamics.

5.2 Details of experiments

The experiments reported in this paper were done using a black-box approach. We did not use any knowledge of the model. The simulation traces were provided by our partners from Czech Technical University in the European project Clockwork.

In all the experiments we used 7 traces for learning with the same road profile as in the trace of Figure 4. In the first learning trace all other three input variables were zero. In the next three traces two of the other three input variables were zero and one other variable (F_x,F_y or M_z) was changing. Figure 4 shows one such trace. The remaining three traces were similar, but the trajectory of the changing variable was different, i.e. it first increased, stayed unchanged for 20 seconds, and than slowly decreased to zero. Each trace gives 100 examples, giving altogether 700 learning examples with 10 variables.

The task was to learn each of the six output variables as a function of input variables. In this way we have six learning problems, where an output variable is the class and the input variables are the attributes. For example, angle a was learned as $\alpha=f(R,F_x,F_y,M_z)$.

The prediction accuracy was tested on 7 test traces. All the test traces have the same road profile as the traces used for learning, but different profiles of other three input variables, i.e. F_x, F_y and M_z. In the first trace all of the three input variables change similar as F_x in the trace in Figure 4. This trace was recommended as the most critical test trace by the domain expert, who considered it far more difficult (all 4 input variables change) than other 6 test traces where one or two input variables were always zero.

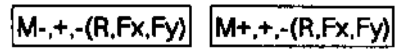

Figure 5: Induced qualitative tree for x wheel center position.

5.3 Inducing a qualitative wheel model with QUIN

As described above, QUIN was used to induce a qualitative tree for each of the six output variables, where the input variables were the attributes. All of the induced qualitative trees had over 99 % consistency on the learning set of examples. We say that a QCF is consistent with an example if the QCF's qualitative prediction of the dependent variable does not contradict the direction of change in the example. The level of consistency of a qualitative tree with the examples is the percentage of the examples with which the tree is consistent. Consistency of 99% indicates that the induced qualitative model fits the data nearly perfectly.

The simplest qualitative tree was induced for translation in the z-axis. This tree only has one leaf with QCF $z=M^{+}(R)$. This tree has a simple and obvious explanation. It says that z changes in the direction of the road change. If road increases then z increases, i.e. the wheel center moves upwards.

Qualitative trees for translations in x and y axes are a bit more complicated. Since they have similar explanations we will present just the qualitative tree for x translations, given in Figure 5. Note that x is measured in the opposite direction as usual, i.e. positive x means wheel center moving in the direction of car driving backwards. Both leaves of the tree have the same qualitative dependence on F_x and F_y, but differ in qualitative dependence on road R. The qualitative tree says that x is positively related to force F_x that acts in the direction of x. Obviously, wheel center position x changes (wheel moves backward or forward) in the direction of force in x direction. Second, x is negatively related to force F_y. This means that if we push the wheels together (we apply force in the y direction), the wheels will move forward (x position decreasing). This is not so obvious, but can be understood if we consider the usual mechanics of wheel suspension. The qualitative dependence on road R is a bit more complicated. The qualitative tree of Figure 5 says that x is negatively related to R when $R \leq 0.001$. Otherwise x is positively related to R, When R increases from its minimum to its maximum, x will first decrease and than increase, i.e. the wheel center will first move forward and than backward.

Rotations about axes x, y and z are measured by enforced wheel-spin γ, camber β and toe angle α, respectively. For enforced wheel-spin γ, QUIN induced a simple one-leaf tree that says $\gamma=M^{-,-,+}(R,F_x,F_y)$. Note that γ changes in the

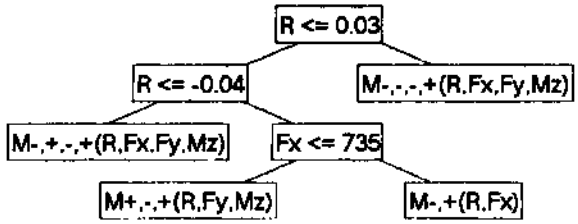

Figure 6: Induced qualitative tree for toe angle α.

direction of the tyre rotation when driving forward. Consider for example the dependence of γ on force F_x that is positive during braking. Since γ is negatively related to F_x, increasing F_x causes γ to decrease, i.e. during braking enforced wheel spin angle changes in the direction of the tyre rotation. For camber angle β QU1N induced a qualitative tree that is similar to qualitative trees for x and y translations.

The toe angle α, i.e. the rotation about z-axis is effected by all input variables and is the most complicated. The induced tree is given in Figure 6. We will omit explanation of this qualitative tree, since it requires understanding of the flexible nature of the multi-body suspension system that links the wheel to the car-body.

These qualitative trees give a good explanation of wheel suspension system behaviour. Moreover, they provide a qualitative model of wheel suspension system that enables qualitative simulation. In this way, they enable to predict all possible qualitative changes of output variables over an arbitrary time interval given qualitative changes of all or some input variables. This qualitative model also enables to improve numerical predictions.

5.4 Qualitative correctness of numerical predictors

In this section we illustrate why Q^2 learning may have an advantage over the usual numerical predictors. Figure 7 shows α predicted with M5 model tree, LWR and LWR with optimized parameters, on the most critical test trace, where all the input variables are changing simultaneously. The figure shows that both M5 and LWR sometimes make large errors. Moreover these errors are not only numerical, but also *qualitative.* Consider for example the LWR predictions at the beginning of the trace. Here the predicted α is decreasing, but the correct α is increasing. This error could be avoided by considering the induced qualitative tree for α given in Figure 6. Since at the beginning of the test trace road *R* is near zero and increasing, and all other input variables are zero, the second leftmost leaf of the qualitative tree applies. Its QCF $\alpha = M^{+,-,+}(R, F_y, M_z)$ requires increasing a since road *R* is increasing.

As can be observed in Figure 7, M5 and LWR often make qualitative errors. Q^2 predictions are qualitatively correct. The use of (induced) qualitative model enables Q^2 to better generalize in the areas sparsely covered by the training examples, resulting in better numerical accuracy.

5.5 Numerical accuracy of the induced models

Here we compare the numerical accuracy of the Weka implementations of LWR, M5 model trees and Q^2 learning. All the methods learned from 7 learning traces (also used for learning of qualitative trees) and were tested against 7 test traces described in Section 5.2. The test results are divided in two groups. The first group consists of results on a single test trace. This trace was recommended as the most critical test trace by the domain expert, who considered it far more difficult (all 4 input variables are changing simultaneously) than the 6 test traces in the second group.

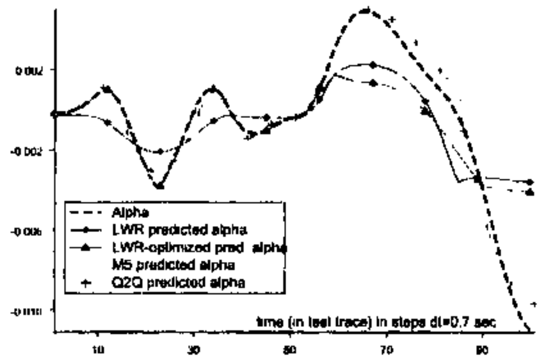

Figure 7: LWR, LWR with optimized parameters, M5 and Q^2 predictions of α on the most critical test trace. With each method, a at time step t_i (on x-axis) was predicted according to the values of input variables at time t_i in the test trace.

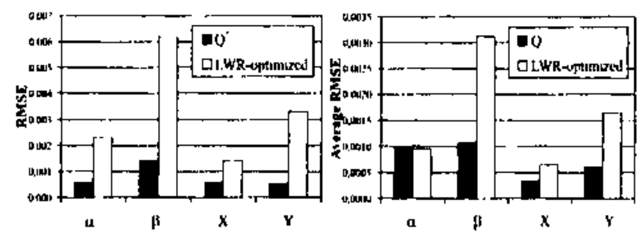

Figure 8: Comparing accuracy (measured with *RMSE)* of Q^2 and LWR with optimized parameters: the left graph compares *RMSE* on the most difficult test trace and the right graph shows *RMSE* on the remaining test traces.

Learning of qualitatively consistent functions in the leaves was performed as described in Section 4. The best fitting regression functions were then taken and glued together into the overall induced numerical model. We compared the accuracy of our Q^2 method with LWR and M5. The parameters of LWR were optimized for each output variable $(\alpha, \beta,$ ...) according to the *RMSE* criterion, through internal cross-validation on the training set. When experimenting with M5, we noticed that it was grossly inferior both in terms of qualitative acceptability as well as numerical error. Attempts at optimizing M5's parameters did not help noticeably.

Figure 8 gives the prediction accuracy for variables α, β, x and y. The predictions of the remaining variables γ and z were not much affected by qualitative constraints. The results on the most difficult test trace (left graph in Figure 8) show that even our simple Q2Q method improves the numerical prediction on all the variables (compared to LWR). Results on the second test trace group are given on the right graph in Figure 8. As these traces were more similar to the learning traces, the improvement of Q^2 over LWR is smaller.

6 Discussion

In this paper we introduced a new approach to machine learning in numerical domains, which we call Q^2 learning (qualitatively faithful quantitative learning). This combines the induction of qualitative properties from numerical data and numerical regression that respects the induced qualitative properties. We showed by an experimental case study that Q^2 learning may lead to the following advantages compared to the usual numerical learning:

(1) Induced models tend to be *qualitatively* consistent with the data and therefore have better chances to correspond to the qualitative mechanisms in the domain of modeling. For example, if the amount of water in a container is decreasing, the level of water cannot be increasing. This is important with respect to the interpretation of induced models and explanation of phenomena in the domain based on these models.

(2) Qualitative consistency of induced models with learning data also affects the accuracy of the model's *numerical* predictions: numerical accuracy may be considerably improved. This is illustrated by the experimental results.

In respect of numerical prediction accuracy, in our case study Q^2 overall outperformed all competing numerical learners. Among these, locally weighted regression (LWR) with optimized parameters (through internal cross validation on the training set) performed best in terms of mean squared error. However its performance may sharply degrade under more difficult circumstances. Consider LWR-optimized performance on the the most difficult test set (Figure 7). It achieves excellent accuracy on the first part of this trace which is similar to data in the training sets. LWR-optimized accuracy there is actually better than that of Q^2. However, problems begin for LWR in the second part of this trace where the input variables start to deviate considerably from the training data, and LWR's predictive error increases sharply. In this part of the trace, Q^2 manages to largely preserve qualitative consistency with the true results, and maintains the numerical accuracy at a comparable level as in the area densely populated by training examples.

LWR-optimized was the best among standard numerical learners, and therefore our presentation of experimental results largely concentrated on comparison between Q^2 and LWR. The performance of M5 was grossly inferior both in terms of qualitative acceptability as well as numerical error. Optimizing M5's parameters did not help noticeably.

It should be noted that qualitatively faithful regression as carried out by the Q2Q program is actually inferior to LWR as a regression method. Struggling to satisfy qualitative consistency, Q2Q is limited to piece-wise linear regression with a small number of linear segments. This numerical inferiority of Q2Q usually turns out to be more than compensated by preserving qualitative consistency.

In this paper, qualitative constraints for Q2Q were induced from training data with QUIN. Alternatively, such constraints can be defined directly by a domain expert. In such a case, the Q^2 learning can be viewed as an approach that enables the use of expert's qualitative knowledge in system identification.

Among the limitations of our realization of Q^2, the primitive numerical regression method in Q2Q should be noted. This method allows sharp changes in variable values (discontinuities in the variables' derivatives) at the borders between leaves of a qualitative tree. Future work should include a method for smoothing such discontinuities.

Acknowledgments

The work reported in this paper was partially supported by the European Fifth Framework project Clockwork and the Slovenian Ministry of Education, Science and Sport. We thank A. Eichberger of Intec, for providing the wheel suspension model and the Simpack simulation software for this study, and for helpful notes on the model. M.Valasck and RSteinbauer from Czech Technical University also helped in the Intec case study.

References

[Atkesonc/tf/., 1997] C.G Atkeson, A.W Moore, and SA. Schaal. Locally weighted learning. *Artificial Intelligence Review,* 11:11-73,1997.

[Breiman *et al,* 1984] L. Breiman, J.H. Friedman, R.A. 01-shen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, California, 1984.

[Coghill?/tf/.,2002] G.M. Coghill, Garrett S.M., and R.D. King. Learning qualitative models in the presence of noise. In *Proceedings of the QR'02 Workshop on Qualitative Reasoning,* 2002. Sitges, Spain.

[Dzeroski and Todorovski, 1995] S. Dzeroski and L. Todorovski. Discovering dynamics: from inductive logic programming to machine discovery. *./. Intell. Information Syst.,* 4:89-108, 1995.

[Intec, 2002] INTEC GmbH. Simpack software. 2002. www.simpack.com.

[Kay *etai,* 2000] H. Kay, B. Rinner, and B. Kuipcrs. Semi-quantitative system identification. *Artificial Intelligence,* 119:103-140,2000.

[Quinlan, 1992] J.R. Quinlan. Leraning with continuous classes. In *Proc. 5th Australian Joint Conf. on Artificial Intelligence,* pages 343-348, 1992. World Scientific.

[Say andKuru, 1996] A.C.C. Say and S. Kuru. Qualitative system identification: deriving structure from behavior. *Artificial Intelligence,* 83:75-141, 1996.

[Sue and Bratko, 2001] D. Sue and I. Bratko. Induction of qualitative trees. In L. De Raedt and R Flach, editors, *Proc. 12th European Conference on Machine Learning,* pages 442-453. Springer, 2001. Freiburg, Germany.

[Sue and Bratko, 2002] D. Sue and I. Bratko. Qualitative reverse engineering. In *Proc. 19th Int. Conf on Machine Learning,* pages 610-617. Morgan Kaufmann, 2002.

LSuc, 2001] D. Sue. *Machine reconstruction of human control strategies.* PhD thesis, Faculty of Computer and Information Sc, University of Ljubljana, Slovenia, 2001.

[Witten and Frank, 2000] Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations,* chapter 8, pages 265-320. Morgan Kaufmann, San Francisco, 2000.