Qualitative Data Mining and Its Applications

Ivan Bratko and Dorian Šuc

Faculty of Computer and Information Science, University of Ljubljana, Slovenia

In machine learning from numerical data, usually the target concept is a numerical function that facilitates quantitative prediction. In contrast to this, we consider qualitative data mining which aims at finding qualitative patterns, or qualitative relationships in numerical data. We present one approach to qualitative data mining, in which the target concepts are expressed as qualitative decision trees. We review some case studies in qualitative data mining, and discuss typical application scenarios that involve the learning of qualitative trees.

Keywords: data mining, machine learning, qualitative reasoning, numerical regression, qualitative induction.

1. Introduction

Consider the mining of numerical data. Usually, in machine learning involved in this, the target construct is a numerical function of the form $y = f(x_1, x_2, \ldots)$ where y is a distinguished variable, usually called the class variable (or dependent variable), and $x_1, x_2, \ldots$ are attributes (or independent variables). Examples of this kind of numerical learning are regression trees (CART, Breiman et al. 84), Retis (Karalič 92), M5 (Quinlan 1992).

In contrast to this, in this paper we consider *qualitative* data mining which aims at finding *qualitative* patterns, or qualitative relationships in numerical data.

An obvious motivation for qualitative data mining comes from the fact that for some tasks, qualitative models are more suitable than classical quantitative, numerical models. Examples of such tasks are diagnosis, generating explanation of system's behaviour, and the design from first principles. In the fields of Qualitative Physics and Qualitative Reasoning (Weld and de Kleer 1990), techniques have been developed that enable a kind of commonsense reasoning with qualitative models, based on the abstraction of numerical values into qualitative values, and real functions into qualitative constraints. This kind of reasoning enables the solution of certain types of problems without resorting to numerical computation, and without the use of a quantitative model of the system in question. When a problem can be solved at the qualitative level of abstraction, there is no need for building a quantitative model — often a demanding or unrealistic task. Building qualitative models for complex systems should be easier. However, even this simpler task is known to be demanding and time consuming, and tools to support this would be welcome. In particular, machine learning methods aiming at inducing qualitative models from (possibly numerical) data would be very useful in this respect.

While there are many machine learning or data mining tools that support the building of numerical models from data, there are few tools to support the building of qualitative models from data.

In this paper we present one approach to qualitative data mining, realized in the program QUIN in which the target concepts are expressed by so-called *qualitative decision trees*. We review some case studies in qualitative data mining, and discuss typical application scenarios that involve the learning of qualitative trees.

2. The QUIN Approach to Qualitative Data Mining

2.1. Representation

QUIN (Qualitative Induction) is a learning program that looks for qualitative patterns in numerical data (Šuc 2001; Šuc and Bratko 2001). These patterns are then combined into a so-called *qualitative tree*. Induction of qualitative trees is similar to the well-known 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 monotonic 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 X. In general, QCFs can have more than one argument. For example, $Z = M^{+,-}(X, Y)$ says that Z monotonically increases in X and decreases in Y.

Monotonic constraints can be combined into if-then rules to express piece-wise monotonic functional relationships. For example:

$$\text{if } X < 0 \text{ then } Y = M^{-}(X) \text{ else } Y = M^{+}(X)$$

Nested if-then expressions can be represented as trees, called *qualitative trees* (Šuc 2001). Qualitative trees are similar to regression trees (Breiman et al. 1984). Both regression and qualitative trees describe how a numerical variable (called *class*) depends on other variables (called *attributes*). The difference between the two types of trees only occurs in the leaves. A leaf of a regression tree specifies a numerical regression function that tells how the class variable numerically depends on the attributes within the scope of the leaf. On the other hand, a leaf in a qualitative tree only specifies the relation between the class and the attributes *qualitatively*, in terms of monotonic qualitative constraints.

QUIN takes as input a set of numerical examples and 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. As a simple example consider a data set with three variables X, Y and Z where data triples (X, Y, Z) correspond to the function $Z = X^2 - Y^2$, possibly with some Gaussian 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 that can be represented by the following nested if-then-else expression:

```
if $X < 0$ then
    if $Y < 0$ then $Z = M^{-,+}(X, Y)$
              else $Z = M^{-,-}(X, Y)$
else
    if $Y < 0$ then $Z = M^{+,+}(X, Y)$
              else $Z = M^{+,-}(X, Y)$
```

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 X and Y. QUIN can tolerate some noise in the data and would induce in this example the same tree (with thresholds for X and Y slightly distorted) even when moderate noise is added to the data.

2.2. Outline of Algorithm

QUIN constructs a tree in a top-down greedy fashion, similarly to decision tree induction algorithms. At each internal node of the tree, QUIN considers all possible splits, that is conditions of the form $X < T$ for all the attribute variables X and effectively all possible thresholds T with respect to X. Each such condition partitions the training data into two subsets. QUIN finds the "best" QCF for each subset according to an error-cost measure for QCFs. Then the best split is selected according to the MDL (minimum description length) principle, which minimizes the error-cost and the encoding complexity of QCFs. The error-cost of a QCF with respect to an example set S is defined so that it takes into account the consistency and "predictive strength" of the QCF with respect to S (the more unambiguous qualitative predictions the QCF can make in S, the better). Technical details of all this can be found in (Šuc 2001) or (Šuc and Bratko 2001) where QUIN's performance on noisy data is also studied.

3. Case Studies

In this section, some applications of QUIN will be reviewed in which some unexpected uses of qualitative data mining were observed. One is the so-called behavioural cloning where the skill of a human operator controlling a dynamic system is reconstructed from the operator's control traces. Another one is qualitative reverse engineering where a device is qualitatively reconstructed from its behaviour data (Šuc and Bratko 2002). Finally, QUIN has also been used in the current work on the so-called Q^2 learning (Qualitatively faithful Quantitative learning) in which numerical regression is carried out in such a way that the qualitative properties in the data are preserved. In an application aiming at speeding up an industrial car simulator, Q^2 learning significantly outperformed other state-of-the-art regression techniques.

3.1. Behavioural Cloning

Controlling a complex dynamic system, such as an aircraft or a crane, requires operator's skill acquired through experience. This skill is tacit and hard to access through introspection. Therefore some questions of interest are: How to understand such tacit human skills? How to design automatic controllers based on human skill, that is how to transfer the operators' skill into a controller?

Given the difficulties of skill transfer through introspection (studied by Bratko and Urbančič 1999), an alternative approach to skill reconstruction is to use the *manifestation* of the skill that is available in the form of traces of the operator's actions. One idea is to use these traces as examples and extract operational models of the skill by machine learning techniques. This is known as *behavioural cloning* (Michie 1993; Michie et al. 1990). In general there are two goals of behavioural cloning:

- To generate *good performance* clones, that is those that can reliably carry out the control task.
- To generate *meaningful* clones, that is those that would help to understand the operator's skill by making the skill symbolically explicit.

Here we illustrate how qualitative machine learning can help the understanding of operators' control strategies, and generating explanation of how and why these strategies work.

In one experiment, a simulated container crane (Fig. 1) was controlled by several human operators and qualitative descriptions of their control strategies were induced with QUIN from the operators' control traces. Each control trace is a sequence of dynamic states of the crane system and operator's actions performed in these states. A control trace in this domain typically consisted of 500 to 1000 states taken at sample time points. In the case of the container crane, a dynamic state of the system consists of six variables:

X, position of the trolley
DX, trolley velocity
Fi, angle of rope with respect to vertical
DFi, angular velocity
L, length of rope
DL, rope length velocity

The available actions are: force $F1$ (to push the trolley left or right), and force $F2$ (to pull the rope). In the task executed by the operator, in the initial state the trolley's position was $X = 0$. The trolley's goal position was $X = 60$. The requirement was that when the load was at the goal position, the swinging of the load was within very small tolerance around the vertical. The task was to be executed in as short a time as possible.

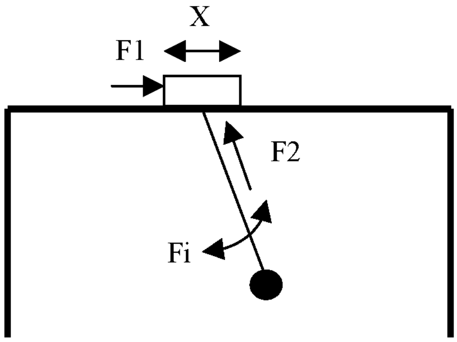

Fig. 1. Container crane with horizontal (F1) and vertical (F2) control forces; the task is to move the load to a goal position while controlling the swinging of the rope at the goal.

To qualitatively analyse a control trace with QUIN, we have to choose one of the state variables or an action variable as the class variable, and a subset of the remaining state variables as the attributes. Let us consider just the horizontal movement of the trolley and the swinging of the load, and choose DX as the class variable and X, Fi and DFi as the attributes. With this selection of variables, we use QUIN to find qualitative properties of the relation $DX = f(X, Fi, DFi)$.

Below we give as an example two qualitative trees induced by QUIN from qualitative traces of two of our human operators that we here refer to as S and L. We chose these two operators because their control performances were rather different, and we were interested in finding the differences in their control strategies. Operator S performed very cautiously, never causing a large swing of the load. To avoid large swing, he could not afford large accelerations of the trolley. The qualitative tree (written as an if-then expression) induced from a trace of operator S, is:

```
if $X < 20.7$
   then $DX = M^{+}(X)$
   else
   if $X < 60.1$ then $DX = M^{-}(X)$
                 else $DX = M^{+}(Fi)$
```

This tree offers a nice explanation of operator S's control strategy. In the initial stage, when X is small, the desired velocity DX increases with X. From about one third of the total distance to the goal, velocity DX starts decreasing with X. Obviously, this aims at zero velocity when the goal $X = 60$ is reached. The right-most leaf of the qualitative tree above ($DX = M^{+}(Fi)$) mentions the angle Fi. This reveals that this operator makes an effort at controlling the swing, although only at the very final stage.

Operator L was much more adventurous than S. His skill allowed him to afford vigorous accelerations in the X direction, causing large swing of the load. However, he had the skill of reducing the swing later. This enabled him to perform much more efficiently in terms of execution time, achieving about 20 or 30% shorter finishing times than S. A qualitative tree induced by QUIN from a trace of operator L is as follows:

```
if $X < 29.3$
   then $DX = M^{+,+,-}(X, Fi, DFi)$
   else
   if $DFi < -0.02$ then $DX = M - (X)$
                    else $DX = M^{-,+}(X, Fi)$
```

Although the overall structure of this tree is similar to that of S's tree, L's qualitative strategy is considerably more sophisticated. Obviously, L was paying attention to the load swing already at the initial stages of the task and making active effort at reducing the swing.

3.2. Qualitative Reverse Engineering

Accumulated engineering design knowledge in a company often takes the form of a library of designs and corresponding simulation models. Typically such libraries contain numerous versions of models (designs) where comparative advantages and drawbacks of alternative models are not comprehensively documented.

Re-use of such design libraries is made difficult specially because the intuitions behind designs and their improvements are not explained in the documentation. Although there may be complete mathematical models and working simulation programs included in the library, the user of the library is impeded by lack of understanding of how does the designed system work. For example, how does a controller of a dynamic system achieve the goal of control.

The case study in (Šuc and Bratko 2002) considers the task of recovering the underlying ideas of designs by aid of qualitative machine learning. We assume a model in an engineering library is complete so that it can be executed on a simulator. The simulated system can thus be observed as a black box, but the internal structure of the system is obscure to the user because it is too complex to be understood without some intuitive explanation. To help the user to develop some intuitive understanding of how the black box works, QUIN was applied to hopefully induce some meaningful relations among the system's variables.

The particular task of this case study was to reverse engineer the design of a crane controller that has been in regular industrial use (Valašek 1996). In this case study we compared two machine learning methods that both seem relevant to this task: (a) the usual regression tree learning, and in particular its variant called model trees, implemented in the M5 system (Quinlan 1992), and (b) induction of qualitative trees implemented in QUIN.

The problem of reverse engineering is similar to the problem of human operator's skill reconstruction. To assess the success of reverse engineering of a controller we use the same criteria as are normally used in the cloning of operators' skill: (1) How successfully the induced clone performs the control task, and (2) how useful the clone is as an explanation of the control strategy implemented by the controller?

The experiments in this case study in reverse engineering of controllers showed advantages of the qualitative learning approach. The induced qualitative trees help to explain intuitively the main idea behind the design of this crane controller. When the qualitative trees were transformed (through optimisation) into a numerical controller that could actually be used to perform the control task, the so obtained controller performed equally well as the original industrial controller and was simpler than the original controller. In a similar case study in qualitative reverse engineering within the Clockwork European research project, a semi active car shock absorber was reversely engineered in a similar way. The resulting controller outperformed the original design according to the performance criteria normally used in the design of shock absorption controllers.

3.3. Q^2 Learning

In (Šuc et al. 2003) QUIN was applied 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, as one would expect from a qualitative model. More surprisingly, however, it was also shown 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 significantly more accurate than those provided by state-of-the-art numerical modelling methods.

Thus the main message of this case study is that a combination of methods for qualitative and quantitative system identification methods has good chances to attain significant improvements over numerical system identification techniques, including techniques of numerical machine learning methods, such as regression trees, model trees, and locally weighted regression. 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 was in (Šuc et al. 2003) carried out in two stages. First, induce with QUIN qualitative constraints from system's behaviour data. Second, induce a numerical regression function that both respects the qualitative constraints and fits well the training data numerically (called Q2Q transformation, Qualitative to Quantitative transformation). This approach was named Q^2 learning, which stands for Qualitatively faithful Quantitative learning.

4. Summary and Conclusions

In this paper we reviewed some ideas of qualitative data mining or qualitative machine learning. We also reviewed case studies in which qualitative induction with QUIN was applied. These case studies demonstrate, as one would expect, that induced qualitative patterns are useful to facilitate the user's understanding of the domain of application, and to enhance the domain knowledge. For example, to better understand the tacit operator's skill or the differences between different operators, or reconstruct the intuition behind an engineering design.

However, less expected, we found another very common application scenario of qualitative learning, when we are not just interested in qualitative relations, but in a concrete quantitative solution (e.g. making quantitative predictions, synthesising an actual, numerical controller for a dynamic system, numerical system identification, or qualitatively faithful quantitative learning).

This type of application gives rise to the general scheme in which we complete the abstraction-concretion loop shown in Fig. 2. In our approach, roughly the qualitative solution was

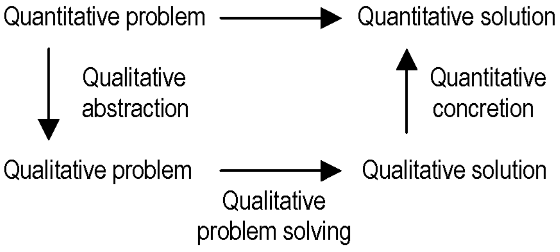

Fig. 2. Solving quantitative problems by means of qualitative abstraction.

obtained through qualitative machine learning (with program QUIN), and quantitative concretion was accomplished as a solution of an optimisation problem in the Q2Q transformation (Qualitative-to-Quantitative transformation).

5. Acknowledgements

The work reviewed in this paper was partly supported by the European Fifth Framework project Clockwork, and the Slovenian Ministry of Education, Science and Sport.

References

[1] BRATKO, I., URBANČIČ, T., Control skill, machine learning and hand-crafting in controller design, in: *Machine Intelligence 15* (eds. K. Furukawa, D. Michie, S. Muggleton), Oxford University Press, 1999.

[2] BREIMAN, L., FRIEDMAN, J.H., OLSHEN, R.A., STONE, C.J., *Classification and Regression Trees*. Monterey, CA: Wadsford, 1984.

[3] KARALIČ, A., Employing linear regression in regression tree leaves. *Proc. ECAI'92 (10th European Conference on Artificial Intelligence)*, pp. 440–441. Vienna 1992. Wiley & Sons.

[4] MICHIE, D., Knowledge, learning and machine intelligence, in: L.S. Sterling (ed.) *Intelligent Systems*, Plenum Press, 1993.

[5] MICHIE, D., BAIN, M., HAYES-MICHIE, J., Cognitive models from subcognitive skills, in: Grimble, M., McGhee, J., Mowforth, P. (eds.) *Knowledge-Based Systems in Industrial Control*, Stevenage: Peter Peregrinus, 1990.

[6] QUINLAN, J. R., Learning with continuous classes, *Proc. 5th Australian Joint Conf. Artificial Intelligence*, pp. 343–348. Singapore: World Scientific, 1992.

[7] ŠUC, D., *Machine Reconstruction of Human Control Strategies*. PhD Thesis, University of Ljubljana, Faculty of Computer and Information Sc., Ljubljana 2001.

[8] ŠUC, D., BRATKO, I., Induction of qualitative trees, *Proc. Machine learning: ECML 2001:* (Lecture notes in artificial intelligence, Lecture notes in computer science, 2167) pp. 442–453, Berlin: Springer, 2001.

[9] ŠUC, D., BRATKO, I., Qualitative reverse engineering, *Proc. ICML'02 (Int. Conf. on Machine Learning)*, Sydney, Australia, 2002.

[10] ŠUC, D., VLADUŠIČ, D., BRATKO, I., Qualitatively faithful quantitative prediction, Proc. IJCAI'2003, Acapulco, pp. 1052–1057, San Francisco: Morgan Kaufmann, 2003.

[11] VALAŠEK, M., Position and velocity control of gantry crane, *Proc. Mechatronic* 96, pp. I–203–208, Guimaraes, 1996.

[12] WELD. D. S., DE KLEER, J. (eds.), *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufmann, San Mateo, California.

Received: June, 2003
Accepted: September, 2003

Contact address:

Ivan Bratko, Dorian Šuc
Faculty of Computer and Information Science
University of Ljubljana
Tržaška 25
SI-1001 Ljubljana
Slovenia
Phone: (+386) 1 4768-267
Fax: (+386) 1 4768-386
e-mail: ivan.bratko@fri.uni-lj.si
dorian.suc@fri.uni-lj.si
Web: http://ai.fri.uni-lj.si/dorian/

IVAN BRATKO is professor of computer science at the Faculty of Computer and Information Science, Ljubljana University, Slovenia. He heads the AI laboratory at the University. He has conducted research in machine learning, knowledge-based systems, qualitative modelling, intelligent robotics, heuristic programming and computer chess. His main interests in machine learning have been in learning from noisy data, combining learning and qualitative reasoning, and various applications of machine learning and inductive logic programming, including medicine, ecological modelling and control of dynamic systems. Ivan Bratko is the author of widely adopted text PROLOG Programming for Artificial Intelligence (third edition: Pearson Education / Addison-Wesley 2001). He co-edited (with R.S. Michalsky and M. Kubat) Machine Learning and Data Mining: Methods and Applications, Wiley, 1998, and co-authored (with I. Mozetic and N. Lavrac) KARDIO: a Study in Deep and Qualitative Knowledge for Expert Systems (MIT Press, 1989).

DORIAN ŠUC is a teaching assistant at the Faculty of Computer and Information Science, University of Ljubljana. He completed his M.Sc. in 1998 and received his Ph.D. in computer science and informatics from the University of Ljubljana in 2001. For his doctoral dissertation he received the 2001 ECCAI Artificial Intelligence Dissertation Award, sponsored by the European Coordinating Committee for Artificial Intelligence. His research interests include machine learning, human skill reconstruction and behavioural cloning, reinforcement learning, qualitative modelling and induction of qualitative models as a general method to discover qualitative relations in data.