(Invited and) Accepted Paper at EUFIT'96, Aachen, September 2-5, 1996

ACTIVE LEARNING OF SOFT RULES FOR SYSTEM MODELLING

Eibe Frank
Department of Computer Science
University of Waikato, Hamilton, New Zealand
T.: +64-7856-2889; email: eibe@cs.waikato.ac.nz

Klaus-Peter Huber
Inst. f. Computer Design and Fault Tolerance (Prof. D. Schmid)
University of Karlsruhe, 76128 Karlsruhe, Germany
T.: +49-721-6084257; email: kphuber@ira.uka.de

Abstract - Using rule learning algorithms to model systems has gained considerable interest in the past. The underlying idea of active learning is to let the learning algorithm influence the selection of training examples. The presented method estimates the utility of new experiments based on the knowledge represented by the existing rulebase. An extended rule format allows to deal with uncertainty. Experiments with different artificial system functions show that the presented method improves the model quality respectively decreases the number of experiments needed to reach a specific level of performance.

1. Introduction

In some cases it is to difficult to describe the behaviour of a technical system analytically with mathematical means like differential equations. But, since a lot of manufacturing processes produce huge amounts of numerical data, a *model* of the underlying *system function* can be derived empirically by analyzing this data. The aim is to build an easy to interpret model which describes the system function as accurately as possible given a certain amount of data. For this purpose different tools for *data analysis* can be used. Here, we concentrate on *rule learning* methods since the generated *rule base* is easy to interpret by a human operator.

Most approaches are based on the assumption that the algorithm has no control over what data it receives from the environment. Therefore the quality of the rule model heavily depends on the given data. For that reason the basic idea of *active learning* is - instead of just providing data by observing the system - to let the learning algorithm actively experiment with the system (figure 1). If e. g. the system under concern is a simulation model the learning algorithm can iteratively define the data points to be evaluated by simulation. The learning algorithm selects *experiments* which are expected to provide the most information about the system's behaviour. This improves the quality of the rule base respectively decreases the number of data points required, which is especially important when experiments with the system are costly, time consuming or dangerous.

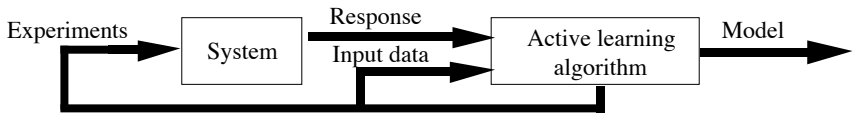

Figure 1: System modelling by active learning

Most existing approaches for active experiment selection are based on neural networks (see e.g. [KRV95][PAK95]) or statistical models (see [CGJ95]). Both kinds of methods have proven to show good performance on function approximation and classification tasks but lack a way to interpret the resulting model. One approach for active learning of easy to interpret rules is presented in [GRO88]. The method is based on different heuristics that aim to maximize the volume that is covered by the rules. Its main weakness is that derived rules are reviewed insufficiently so that errors in the rule base may persist. The approach described in [NIQ92] only delivers regions of interest where experiments could be performed. The concrete experiments have to be provided by other means, e.g. a human operator or a data base.

In this paper we present a new method for active learning of rules. The central idea is to estimate the *utility* of new experiments based on the knowledge represented by the already existing rule base. The rules' form is extended in a way that allows uncertainty in the model to be expressed more flexibly. The method allows to embed arbitrary *passive learning* algorithms that are able to extract a rule base from given data.

2. Experiment Generation

A passive rule learning algorithm needs *examples* of the system's behaviour to construct a model from. These examples consist of a fixed number of *input parameters* and a corresponding *output class*, assuming that there is only one output parameter. In an application for fault diagnosis input parameter could represent sensor information and each output class could represent a specific fault. If the output is continuous it has to be mapped on a fixed

number of classes before the learning method can be applied. If there is more than one output parameter a separate model has to be built for each of them. In the following we will describe how an arbitrary passive rule learning algorithm can be extended to automatically generate useful examples by experiments (see figure 2).

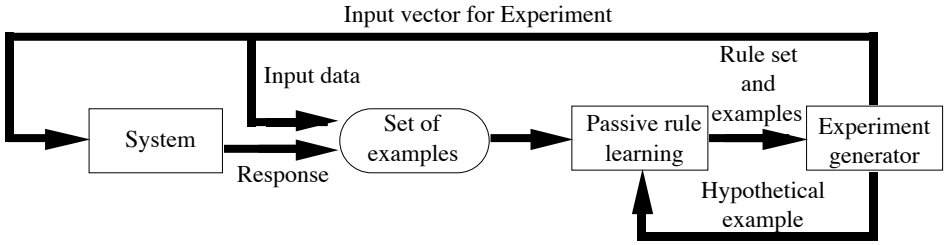

Figure 2: A framework for active rule learning

The passive learning algorithm generates a model using the examples gathered so far. These examples and the model are then passed to an *experiment generator* which chooses a new experiment. The result of that experiment leads to a new model. Choosing an experiment which maximizes the expected *quality* of that model would be the best solution. But in general it is not known what the resulting model will look like since that depends on the experiment's resulting class. For that reason hypothetical models have to be evaluated. Since in general the experiment may produce any output class there are as many *hypothetical models* as there are output classes. They can be generated by simply adding a corresponding *hypothetical example* to the set of examples and then the passive learning method processes each of these hypothetical sets. In the next section we will show a way to measure the quality of these hypothetical models and a way to determine their probabilities based on the existing model. Given that, we can define the experiment's expected *utility* as the expected quality over all hypothetical models.

Using the procedure described above it is possible to compute the expected utility of one specific experiment. Since the aim is to maximize the expected utility it is necessary to evaluate as many experiments as possible to get closest to the maximum. These candidates for a new experiment are generated according to the known distribution of the input vectors. This distribution depends on the system under concern since not all input vectors may be equally important.

To reduce the number of candidates efficiently the following heuristic can be used. It is based on the assumption that in general more useful experiments are different from experiments already executed before. So for each candidate the Euclidean Distance to the nearest of the previously executed experiments is computed. After that the candidates are sorted according to that distance. One can then remove a certain percentage of the candidates with the smallest distances before computing their expected utilities (figure 3). As will be demonstrated later this preselection of candidates for a new experiment is very efficient.

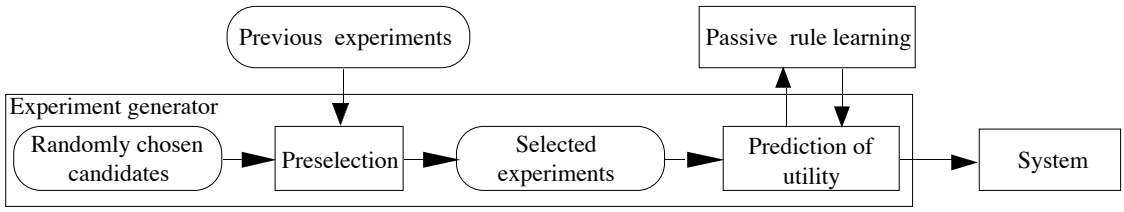

Figure 3: Selection of an experiment

3. Rules and Density Functions

The type of model regarded in this paper consists of an extended form of rules, so-called soft rules [ZAD94]. Those rules can be constructed from data examples very efficiently (e.g. with the Rectangular Basis Network approach [HUB95] or with MaxRect [FRA94]). Each rule can be written as:

$$\text{IF } x_1 \in [a_1, b_1] \subset (\alpha_1, \beta_1) \wedge x_2 \in [a_2, b_2] \subset (\alpha_2, \beta_2) \wedge \ldots \wedge x_n \in [a_n, b_n] \subset (\alpha_n, \beta_n) \text{ THEN CLASS IS } cl_j$$

where $x_1,\ldots,x_n$ are numerical input parameters and cl_j is one of the output classes. The rule includes a more specific part (*core region*) where the certainty of class membership is high (= 1) and a more general part (*support region*) where it is lower. In high dimensional input space one rule is represented by two hyper-rectangles (a two-

dimensional example is shown in figure 4a). The core region is the minimum rectangle which covers the examples (black dots) the rule is responsible for while the support region is extended until it reaches an example (white dot) of a conflicting class.

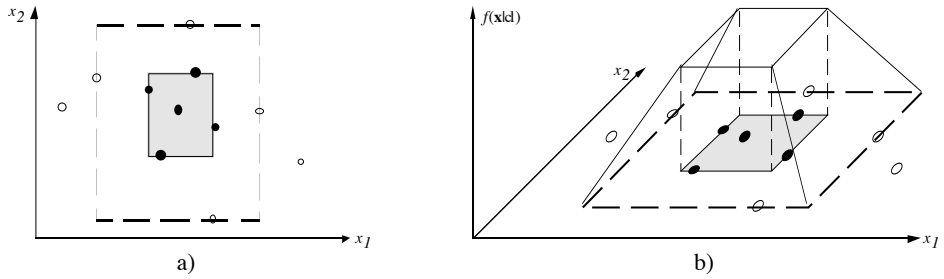

Figure 4: A soft rule with core and support regions (a) and the corresponding density function (b)

To get the expected utility $U(X)$ of an experiment X it is necessary to measure the quality $Q(cl_j,X)$ of its possible results and to derive their probabilities $p(cl_j|X)$. The whole process is depicted in figure 5. Our approach for deriving the probabilities is based on the computation of a trapezoidal density function corresponding to each rule. This includes the assumption that the credibility for class membership degrades linearly from the border of the core to the border of the support region (for an example see figure 4b).

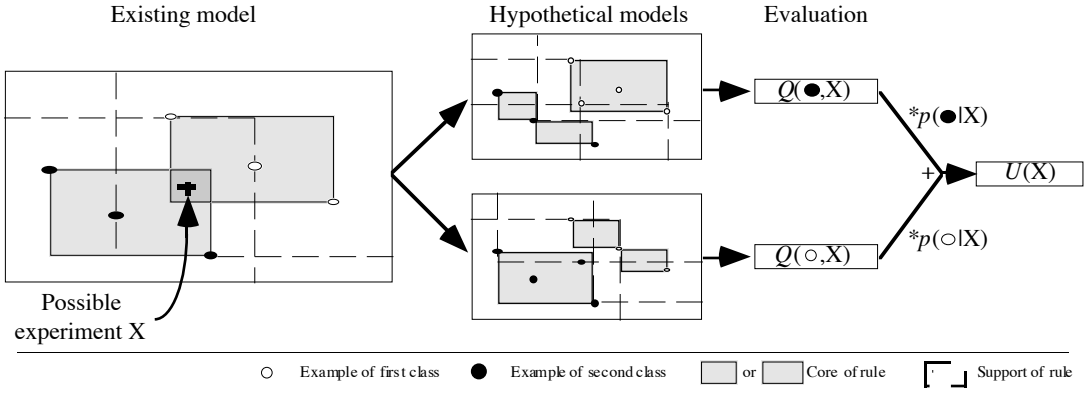

Figure 5: Computing the expected utility of an experiment

Let k_{il} be the number of examples of class cl_l the rule i is responsible for. Furthermore let n_l be the overall number of examples belonging to that class. We then define the height of the rule's trapezoidal density function by demanding that the following condition is met by its volume:

$$\int f_i(\mathbf{x}|cl_l)d\mathbf{x} = \frac{k_{il}}{n_l}$$

A density function of the class conditional probability distribution can then be obtained by simply summing up the density functions belonging to the corresponding rules. It is easy to show that the sum is indeed a density function of a probability distribution:

$$\int p(\mathbf{x}|cl_l)d\mathbf{x} = \int \sum_i f_i(\mathbf{x}|cl_l)d\mathbf{x} = \sum_i \int f_i(\mathbf{x}|cl_l)d\mathbf{x} = \sum_i \frac{k_{il}}{n_l} = \frac{n_l}{n_l} = 1$$

Using Bayes' rule it is then possible to compute the probability that - according to the model and the assumptions - a given experiment will result in the system's output belonging to a specific class:

$$p(cl_l|\mathbf{x}) = \frac{p(\mathbf{x}|cl_l) * p(cl_l)}{\sum_l p(\mathbf{x}|cl_l) * p(cl_l)} = \frac{p(\mathbf{x}|cl_l) * p(cl_l)}{p(\mathbf{x})}$$

Given a way to derive a hypothetical model's probability the problem remains how to define its quality. In general the quality is defined as the model's accuracy when used for predicting class memberships. But when all examples are used for building the model there is no additional data available to test it. So heuristics have to be used for measuring its quality.

Two observations concerning the accuracy of a model are important for choosing efficient heuristics. In the beginning of the learning process it is important that the model gives an as complete as possible but possibly insecure description of the system's behaviour (*coarse modelling*). This leads to the biggest increase in classification performance. A criterion for the coarse modelling process should therefore prefer models which cover all relevant regions of the input space with rules. This property of a model can be measured by the *cross entropy* between the given density of the input vectors and the derived one ($p(\mathbf{x})$). It is a measure for the similarity between two densities.

After a certain amount of training cycles all important regions of the input space are covered with suitable rules. Then it becomes more important to fine tune the model (*fine modelling*). This can be done by reducing the uncertainty in the model (overlapping rules). A criterion for the fine tuning process should therefore prefer models with minimal uncertainty. The *conditional entropy* measures this property based on the probabilities of class membership.

It remains the problem of switching from coarse modelling to fine modelling at the appropriate time. There is no general solution for this problem since it depends on the complexity of the system's behaviour. The heuristic chosen when obtaining the experimental results described in the next section gives priority to the coarse modelling criterion. Only when no experiment can be found for which its value is expected to increase the fine modelling criterion is used.

4. Some Results

Many experiments have been conducted to compare active learning against passive learning. In the following we will present exemplary results of learning two artificial system functions in two-dimensional space (figure 6). The output values were partitioned into three classes. The rule learning algorithm MaxRect was applied to generate the rules. It generalizes an example to a core rectangle by including examples of the same class with increasing Euclidean Distance. After all examples are covered the support rectangles are generated by extending the cores. More details can be found in [FRA94].

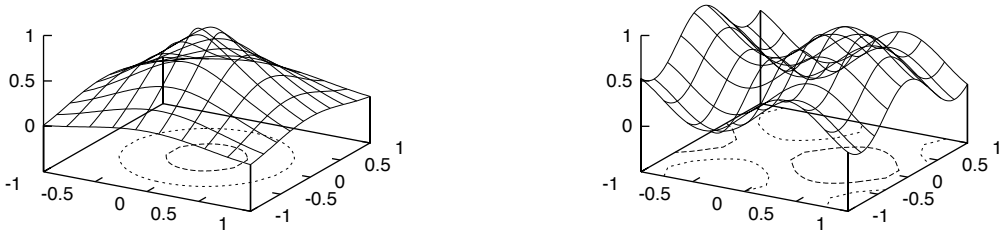

Figure 6: Two artificial system functions - borders of classes are shown as dotted lines

Three different learning modes were used to obtain the results shown in figure 7. They show the error in classification of thousand randomly selected test examples relative to the number of experiments performed. The first mode applied (solid line) was pure passive learning where the experiments were selected at random. For the second mode (dashed line) 100 candidates for an experiment were selected randomly. After that the preselection described above was used to choose the experiment "farest" from the experiments already performed. Finally for the third mode (dotted line) the preselection was used to select 10 candidates out of 100 choosen at random. After that the expected utility was computed for the rest of the experiments subsequently choosing the one with the highest score for execution. The graphs show the average over results obtained with three different random number streams.

In both cases active learning performs much better then passive learning. On average around twice as many experiments are needed by passive learning for reaching the same error rate. For the less complex first function an error rate under 10 percent is reached with less than 100 experiments. The pure preselection is slightly more efficient than passive learning but selection according to the expected utility leads to the best result. For the more complex second function twice as many experiments are necessary to reach an error rate below 10 percent. In this case pure preselection outperforms passive learning and is even better than selection according to the ex-

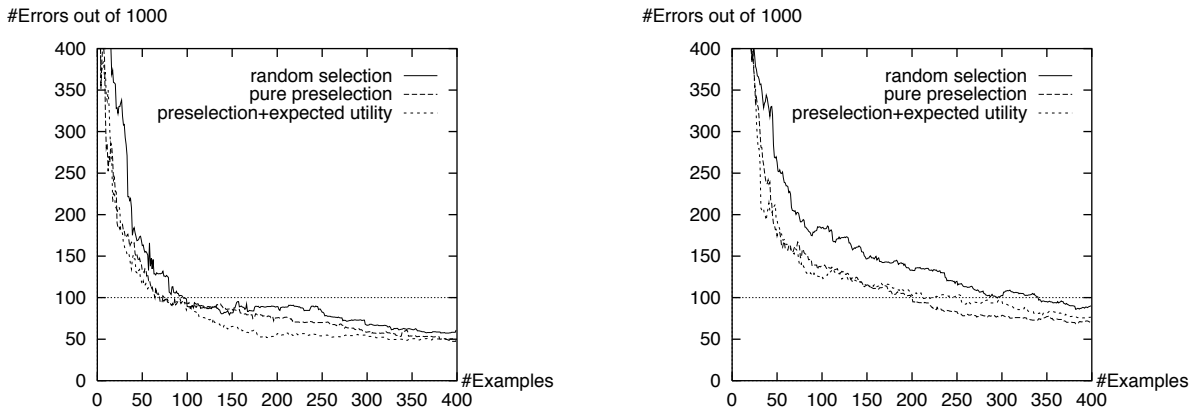

Figure 7: Classification performance relative to the number of experiments

pected utility. Two hypotheses are based on these observations. First, preselection seems to improve the quality of the model. Secondly, selection according to the expected utility seems to improve the results of preselection if the function can be described accurately with few rules. Additional experiments with other system functions and also other learning algorithms (e.g. C4.5 [QUI93]) corroborated these hypotheses (for more details see [FRA95]).

5. Conclusions

The presented method can be used to automatically derive a rule model describing the behaviour of a given technical system by generating new experiments. It selects an experiment using a preselection method and by maximizing its expected utility using the current rule model. Experimental results show that the preselection is very efficient. Independent of the system it improves the quality of the model compared to random selection. Additional selection by using the expected utility seems only to be useful if the system's behaviour can be described accurately with few rules.

The presented framework also allows to include arbitrary passive rule learning algorithms. Therefore active learning can be used to improve the process of modelling systems with automatically generated rules.

Acknowledgements

Thanks go to Prof. Dr. D. Schmid for his support and the opportunity to work on this interesting project. Thanks also to Michael Berthold for many fruitful discussions and helpful remarks.

References

[CGJ95] David A. Cohn, Zoubin Ghahramani, Michael I. Jordan: Active Learning with Statistical Models, Advances in Neural Information Processing Systems 7, pp. 705-712, Morgan Kaufmann, 1995.

[FRA94] E. Frank: Entwicklung eines induktiven Rechteck-Lernverfahrens zur Regelextraktion, Studienarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, 1994.

[FRA95] E. Frank: Systemmodellierung durch aktives Lernen von Regeln, Diplomarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, 1995.

[GRO88] K.P.Gross: Incremental Multiple Concept Learning Using Experiments, Proceedings of the Fifth Internatinal Workshop on Machine Learning, Michigan, Ann Arbor, pp. 22-28, 1988.

[HUB95] Klaus-Peter Huber und Michael R. Berthold: Building Precise Classifiers with Automatic Rule Extraction, IEEE International Conference on Neural Networks, vol. 3, pp. 1263-1268, 1995.

[KRV95] Anders Krogh, Jeders Vedelsby: Neural Network Ensembles, Cross Validation and Active Learning, Advances in Neural Information Processing Systems 7, pp. 231-238, 1995.

[NIQ92] Y. Niquil: Guiding Example Acquisition by Generating Scenarios, Proceedings of the Ninth International Workshop on Machine Learning, pp. 348-354, 1992.

[PAK95] Gerhard Paass, Jörg Kindermann: Bayesian Query Construction for Neural Network Models, Advances in Neural Information Processing Systems 7, pp. 443-450, Morgan Kaufmann, 1995.

[QUI93] J. Ross Quinlan: C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.

[ZAD94] Lotfi A. Zadeh: Soft Computing and Fuzzy Logic, IEEE Software, pp. 48-56,nov. 1994.