Genetic Programming – A Tool for Flexible Rule Extraction

R. König, U. Johansson, L. Niklasson

Abstract— **Although data mining is performed to support decision making, many of the most powerful techniques, like neural networks and ensembles, produce opaque models. This lack of interpretability is an obvious disadvantage, since decision makers normally require some sort of explanation before taking action. To achieve comprehensibility, accuracy is often sacrificed by the use of simpler, transparent models, such as decision trees. Another alternative is rule extraction; i.e. to transform the opaque model into a comprehensible model, keeping acceptable accuracy. We have previously suggested a rule extraction algorithm named G-REX, which is based on genetic programming. One key property of G-REX, due to the use of genetic programming, is the possibility to use different representation languages. In this study we apply G-REX to estimation tasks. More specifically, three representation languages are evaluated using eight publicly available data sets. The quality of the extracted rules is compared to two standard techniques producing comprehensible models; multiple linear regression and the decision tree algorithm C&RT. The results show that G-REX outperforms the standard techniques, but that the choice of representation language is important.**

I. Introduction

In the data-mining domain, one main goal is to produce predictive models able to accurately estimate future values of the target variable. The most accurate techniques often produce very complex models that must be considered opaque. An opaque model, for example a trained neural network or an ensemble, is incomprehensible for human decision makers. Experience from the field of Expert Systems has, however, shown that an explanation capability is a vital function provided by symbolic AI systems. In particular the ability to generate even limited explanations is absolutely crucial for the user acceptance of such systems; see e.g. [1]. Since the purpose of most data mining systems is to support decision making, the need for explanation facilities in these systems is apparent. Nevertheless many systems (especially those using neural network techniques but also ensemble methods like boosting) are normally regarded as black boxes; i.e. they are opaque to the user.

This work was supported by the Information Fusion Research Program (University of Skövde, Sweden) in partnership with the Swedish Knowledge Foundation under grant 2003/0104 (URL: http://www.infofusion.se).

R. König is with the School of Business and Informatics, University of Borås, Sweden. SE-501 90 Borås, Sweden. (phone: +46 (0)33 – 4355945; (email: rikard.konig@hb.se).

U. Johansson is with the School of Business and Informatics, University of Borås, Sweden. (email: ulf.johansson@hb.se).

L. Niklasson is with the School of Humanities and Informatics, University of Skövde, Sweden. (email: lars.niklasson@his.se).

II. Background

A. Predictive modeling

The purpose of a predictive model is to allow the data miner to predict an unknown (often future) value of a specific variable; the target variable. The predictive model is created using available historical, data. Most often a predictive model is created using directed data mining; i.e. a top-down approach where a mapping from a vector input to a scalar output is learnt from samples. The training data thus consists of pairs of measurements, each containing an input vector $\boldsymbol{x}(i)$ with a corresponding target value $y(i)$. The predictive model is an estimation of the function $y=f(\boldsymbol{x},\theta)$ used to predict a value y given an input vector of measured values $\boldsymbol{x}$ and a set of estimated parameters θ for the model.

A special case is time series prediction, where the samples are ordered by time and where the input vector often includes previous values of the target variable.

For predictive modeling, the primary goal is to achieve high accuracy, i.e. a low error between the predicted value and the real value, when the model is applied to novel data. In order to improve accuracy, ensembles are often used. First, several models are created, possibly using different techniques, and then these models are combined into an ensemble. The motivation for using ensembles in general is obvious; they are more robust, i.e. their applicability span over a larger set of problems. Bishop [2] shows why an ensemble of ANNs, in general, has higher accuracy than the mean accuracy of its members.

To achieve comprehensibility, accuracy is often sacrificed by the use of simpler models like decision trees; a trade-off termed the accuracy vs. comprehensibility trade-off. Several researchers have tried to bridge the gap by introducing techniques for converting opaque models to transparent models, without sacrificing the performance. Most significant are the many attempts to extract rules from trained neural networks; for a survey see [3]. It should be noted that rule extraction could be used in two slightly different ways. If the extracted representation is used for predicting this would normally result in at least a small loss of accuracy. An alternative is to employ the opaque model for the actual predictions and use the extracted representation only as an explanation facility.

B. G-REX

We have previously [4] [5] suggested a method for rule extraction based on genetic programming (GP). The method named G-REX (Genetic Rule EXtraction), is very general since the representation language is determined by the function and terminal sets, while the fitness function controls

the accuracy vs. comprehensibility trade-off. G-REX is a black-box method; i.e. it directly creates a function describing the output in terms of the input by using training examples generated by the opaque model. The easiest way to understand the process is to view black-box rule extraction as an instance of predictive modeling where the original input patterns are used with the corresponding prediction from the opaque model as the target variable.

G-REX uses GP when searching for the best solution. More specifically, a pool of candidate rules is continuously evaluated using a fitness function. The evolution of the rules uses the standard genetic operations as described in [6]. The initial population is created randomly following the ramped half and half strategy. Rules are chosen for reproduction using roulette wheel selection with reselection. The genetic operators (crossover and mutation) are applied to the selected rules in a standard fashion, and it is at all times made sure that the resulting programs are still syntactically correct. The exact parameters like crossover and mutation rates, number of individuals in the population and number of generations are normally found from initial experimentation. After several generations, the fittest program is chosen as the extracted rule.

As mentioned above, when G-REX is applied to a specific problem, fitness function, function set and terminal set must be chosen. The function and terminal sets determine the representation language, while the fitness function captures what should be optimized in the extracted representation. Different representations are more or less suitable for a specific problem, which makes it important to be able to adjust the representation language for the problem at hand. As G-REX is based on genetic programming, it can handle almost arbitrary functions and terminals, but they must of course be appropriate for the current problem. G-REX has a large set of predefined functions including arithmetic operators, conditional operators, relational operators, and Boolean operators. The terminals usually represent the input variables and random constants. Functions and terminals are combined randomly, but the possible combinations are restricted by a predefined syntax. G-REX uses a syntactical language similar to the Backus-Naur form, (BNF) making it straightforward to tailor the representation for each problem.

The fitness function normally includes components measuring fidelity and comprehensibility. The fidelity component could, for classification tasks, for instance, increase the fitness value by one for each training sample classified in the same way as the opaque model. The comprehensibility component is normally a penalty term; typically proportional to the length of the program.

We have previously used G-REX in different domains and evaluated it extensively. G-REX has extracted rules in several representation languages; e.g. Boolean rules, decision trees, fuzzy rules and regression trees. For details see e.g. [4], [5] and [7].

Previous studies have focused on classification tasks, and only a few experiments have been performed on estimation tasks. In the estimation study mentioned, G-REX used a representation language described using BNF in Table 1. Furthermore, the use of G-REX for estimation was only tested in the market response modeling domain. With this in mind, the overall purpose of this study is to evaluate G-REX using two new representation languages for estimation.

III. METHOD

This study evaluates if GP, and more specifically G-REX, can be used to extract accurate comprehensible rules from an ensemble of Artificial Neural Networks (ANN). The extracted rules are evaluated against the tree algorithm C&RT, and multiple linear regression. The following sub-sections will describe the experimentation.

A. Ensemble creation

The ensembles used in the experiments consist of 20 fully-connected ANNs, each having one hidden layer. Each ANN is trained on an individual subset of the training set. The subset is created by randomly picking 80% of the training instances without replacement. The number of input variables is also reduced by selecting only 80% of the features randomly. The purpose of this is, of course, to increase the ensemble diversity. Each ANN is initiated with random weights, and with a random number of hidden units which is calculated using (1), where rand is a random number between 0 and 1.

$$\#hidden = floor(rand * 20) \quad (1)$$

The training algorithm used to train the ANNs is MatLab's *traingdx*, which is a gradient descent backpropagation algorithm, using a momentum term and adaptive learning rate. A validation set is used to decide when to stop the training. The validation sets are created by removing 20% of the training instances and assigning them to the validation set.

B. Representation language

As described above, one of G-REX main strengths is that it can use many different representation languages. To evaluate the importance of representation language, the experiments are performed evaluating three different representation languages. The representation languages used by G-REX are presented using BNF in the tables below.

TABLE 1
BNF NORMAL REPRESENTATION (NORM)

Rule ::=	<if –statement>
<if –statement>::=	If <condition> then < expression > else < expression >
<condition>::=	<continuous variable> <function> <constant> | <continuous variable> <function> <continuous variable> | <categorical variable> = <class>
<function>::=	< | >
< expression >::=	<if-statement> | <constant>
<constant>::=	Double in the range of associated <continuous variable>
<class>::=	Class of associated <categorical variable>

NORM is the simplest representation language for a regression tree and uses the same set of functions and terminals as C&RT. The only difference is that C&RT's syntax only allows comparisons between a variable and a constant value in a condition, while NORM also allows comparisons between two variables. NORM is the original representation language used for G-REX estimation in [5]. The if-statements can be seen as a series of conditions dividing the instances of a data set into subgroups depending on the value of some input variables. Each subgroup is associated with the value of a certain terminal. For the NORM BNF the terminal is always a constant which means that all instances (known and novel) belonging to a certain terminal will be predicted to have the same value. The conditions of the if-statements and the terminal constants are of course optimized, using GP, on the training data.

TABLE 2
BNF LINEAR REGRESSION TERMINALS (LINREG)

Rule ::=	<if –statement>
<if –statement>::=	if <condition> then < expression > else < expression >
<condition>::=	<continuous variable> <function> <constant> | <continuous variable> <function> <continuous variable> | <categorical variable> = <class>
<function>::=	< | >
<expression >::=	<if-statement> | <linear regression>
<linear regression>::=	<continuous variable> * <constant> + <constant> Calculated using the least square method on the training instances reaching the node:
<constant>::=	Double in the range of associated <continuous variable>
<class>::=	Class of associated <categorical variable>

The LINREG representation extends the NORM BNF by allowing arithmetic operators and variables as terminals. The BNF ensures that each terminal is a part of a linear regression.

Using this grammar, the representation language becomes more powerful, since it can predict different values for each of the instances it is associated with. The predicted value p_i for instance i is calculated using equation (2) below; where z_i is the value of one of the input variables. K and M are constants found by the linear regression.

$$p_i = K * z_i + M \quad (2)$$

When a linear regression expression is created, it is not done randomly, instead the best linear regression for the selected input variable are calculated by the least square method [8]. The calculations are done on the training instances associated with the linear regression terminal, when the rule is first created. During evolution, mutation and crossover are only applied to the if-statements and its conditions, which will affect how the instances are associated with the terminals but not the linear regressions, which will remain the same. Mutation can introduce new linear regression terminals during the evolution.

The FREE representation presented below is similar to LINREG with two exceptions. First, each subgroup of instances can be associated with a combination of several linear regressions, making it possible to base the prediction on several input variables. Secondly the linear regressions are created randomly and are affected by the genetic operations in normal fashion. The linear regressions are combined by addition, in the same way as in multiple linear regressions. This is in essence a way to evolve different multiple linear regressions for different parts of the data sets.

TABLE 3
BNF FREE REPRESENTATION (FREE)

Rule ::=	<if –statement>
<if -statement>::=	if <condition> then < expression > else < expression >
<condition>::=	<continuous variable> <function> <constant> | <continuous variable> <function> <continuous variable> | <categorical variable> = <class>
<function>::=	< | >
< expression >::=	<if-statement> |<+ -statement>| <* -statement>
<+ -statement>::=	<term> + <term>
<* -statement>::=	<continuous variable> * <constant>
<term>::=	<+ -statement>|<* -statement>| <constant>
<constant>::=	Double in the range of associated <continuous variable>
<class>::=	Class of associated <categorical variable>

C. Fitness function

For estimation tasks, G-REX uses a fitness function based on the total absolute error *ae* made by the rule. To produce short comprehensible rules, G-REX also uses a punishment for longer rules. The length punishment is calculated as the number of tokens (element in the BNF interior representation) in the rule times a factor, here 0.01. In previous studies, where G-REX has been used to extract regression rules, equation 3 has been used to calculate the fitness of rule *r*.

$$fitness_r = -ae_r - \#tokens_r * factor \quad (3)$$

ae is negated as a higher fitness value usually corresponds to a fitter individual. The factor is used to balance accuracy versus comprehensibility, but it will have different effect depending on the magnitude of the error. Before G-REX is applied to a new data set, trial experiments have to be performed to decide an appropriate magnitude for the factor. To remove the need of trial experiments a constant *C* is introduced into the fitness function, see equation 4. *C* is used to normalize the *ae* in a way that makes the balancing of the length of the rule independent of the magnitude of the error. The normalization ensures that a certain factor will have the same affect on the rules for all data sets.

$$fitness_r = 1 - \frac{ae_r}{C} - \#tokens_r * factor \quad (4)$$

The constant *C* is initially set to the sum of the target variable and is then updated after each generation according to Equation 5.

$$C = \begin{cases} C * 1.5 & if \ \ fitness_{best} > 0.9 \\ C & if \ \ 0.1 \geq \ \ fitness_{best} \geq 0.9 \\ C/1.5 & if \ \ fitness_{best} < 0.1 \end{cases} \quad (5)$$

D. Evaluation

To evaluate the benefit of using rule extraction, G-REX is evaluated against algorithms that produce comprehensible models. In this study, G-REX is evaluated against multi linear regression (MREG) [9] and C&RT [10] a common decision tree algorithm. Both techniques produce transparent models, normally regarded as comprehensible.

E. Error measures

When comparing different estimation models over several data sets, it is important to use an appropriate error measure. Two often used error measures are the root mean square error (RMSE) and the coefficient of determination, R^2, which is strange since it is well known that both measures are unsuitable for estimation tasks. R^2 takes no account for the bias which means that even if a model has a perfect R^2 its prediction can still differ greatly from the actual value [11].

RMSE is a bad choice for several reasons; it is poorly protected from outliers, has low reliability and low construct validity [12]. Another reason is that when comparing models over several data sets the error metric has to be relative. This is true for both estimations tasks [13] and for time series prediction [11]. A relative measure is a measure which is relative to some reference. There are many choices for reference but usually some simple straightforward model like the mean is used. A relative measure is also important since values of the target variable can differ greatly in magnitude between data sets, which of course will affect the magnitude of the error. The reference also accounts for the difficulty of the data set which also is of great concern in a comparison. It is not always worse to achieve a higher error on a difficult task then a slightly lower error on a simple task.

In [12], several error measures were evaluated according to reliability, construct validity, sensitivity to small changes, protection against outliers, and their relationship to decision making. Geometric Mean of the Relative Absolute Error (GMRAE) was presented as one of the strongest measures. GMRAE is based on Realtive Absolute Value (RAE) which is defined in Equation 6. $F_{m,s}$ is the prediction (forecast) made by the model *m* on instance *s*, A_s is the actual (real) value. RAE is relative to the random walk (RW) with zero drift, i.e. the most recent known actual value. For series *s* RW predicts A_{s-1}.

$$RAE_s = \frac{|F_{m,s} - A_s|}{|F_{rw,s} - A_s|} \quad (6)$$

Random walk is often used as a reference point since it is straightforward and easily interpreted. In addition, random walk most often outperforms the mean prediction for time series. To handle extreme values, the RAE is trimmed using Winsorizing according to equation (7).

$$WRAE_s = \begin{cases} 0.01 & if \ RAE_s < 0.01 \\ RAE_s & if \ 0.01 \leq RAE_s \leq 10 \\ 10 & if \ RAE > 10 \end{cases} \quad (7)$$

GMRAE summarizes the WRAE using the geometric mean. The arithmetic mean is not suitable as any arithmetic average will be dominated by its large terms. In other words, good small errors should be counted but would essentially be ignored. It might be reasonable to expect that a large error should be balanced by a sufficiently small error which is the case for the geometric average but not for the arithmetic averages [14].

$$GMRAE = [\prod_{s=1}^{s} WRAE_s]^{1/s} \quad (8)$$

This study focus on estimation problems where the RW cannot be applied. But the GMRAE can still be used by simply exchanging the reference to the RW in equation 6 with the mean of the currently known actual values.

F. Data sets

The study is performed on eight data sets which are presented below. All data sets are publicly available at the UCI repository. Table 4 shows the general properties of the data sets. #In is the number of instances in the data set, #co is the number of continuous input variables and #ca is the number of categorical input variables. The mean value (Mean), the standard deviation (Std), the maximum value (Max), the minimum value (Min) and the number of outliers (#Outl.) are all calculated for the target variable.

- **Auto price (AUT).**
  The data set consists of car specifications from which the price of the cars should be predicted.
- **Diabetes_Numeric (DIA)**
  The objective is to investigate the dependence of the level of serum C-peptide on the various other factors in order to understand the patterns of residual insulin secretion. The response measurement is the logarithm of C-peptide concentration (pmol/ml) at the diagnosis, and the predictor measurements age and base deficit, a measure of acidity.
- **Boston Housing (HOU).**
  The inputs variables is statistics related to the livings standard in the different suburbs. The target variable is the median value of owner-occupied homes.
- **Machine CPU (MAC).**
  The problem concerns predicting the relative CPU performance based on six CPU properties.
- **Pharynx (PHA).**
  The data set consist of patients with squamous carcinoma of 3 sites in the mouth and throat (in the oropharynx). The objective of the study was to compare the two treatment policies with respect to patient survival time.
- **Sleep (SLE).**
  Includes brain and body weight, life span, gestation time, predation and danger indices for 62 mammals. The target is the total time the mammal spends sleeping.
- **Veteran's Admin. Lung Cancer Trial (VET).**
  The data set contains data about veterans with lung cancer. The input variables consist of information about the patients, the type of cancer and the treatment. The target variable is the patient's survival time.
- **Wisconsin (WIS).**
  Each record in the data set represents follow-up data for a breast cancer study. The target is the recurrence time of the cancer and the input variables is measures for the cell nucleus.

TABLE 4
PROPERTIES OF DATA SETS

	#In	#Co	#Ca	Mean	Std	Max	Min	#O
AUT	159	14	1	11446	5878	35056	5118	7
DIA	43	2	0	4.8	0.72	6.6	3	2
HOU	506	13	1	22.5	9.2	50	5	31
MAC	209	6	0	105.6	161	1150	6	9
PHA	195	2	10	555.5	422	1823	0	13
SLE	62	7	0	8.5	5.8	20	0	0
VET	137	3	4	121.6	158	999	1	5
WIS	194	32	0	46.9	34.5	125	1	9

The experiments were performed using standard 10-fold cross validation. The data sets were stratified to ensure that each fold was representative.

IV. RESULTS

The results presented in Table 5 are average GMRAE over the ten folds of each data set. ENS is the result for the ensemble, which is the target for G-REX. The bold results are the best result for each data set excluding the ensemble. The results are relative to the mean prediction, so a value less than one means that the results are better than the mean prediction.

TABLE 5
GMRAE ON INDIVIDUAL DATA SETS

	ENS	NORM	FREE	LINREG	MREG	C&RT
AUT	0.235	0.351	0.341	**0.262**	0.317	0.303
DIA	0.871	1.054	3.186	**0.818**	0.942	0.971
HOU	0.370	0.852	0.734	**0.488**	0.506	0.501
MAC	0.250	0.499	0.403	**0.289**	0.383	0.320
PHA	0.672	0.952	0.946	0.725	**0.672**	1.049
SLE	0.520	0.856	0.777	0.665	1.104	**0.238**
VET	0.856	0.848	0.992	**0.772**	0.947	1.034
WIS	0.835	0.961	0.886	**0.902**	0.903	1.008
MEAN	0.576	0.796	1.033	**0.615**	0.722	0.678

G-REX with LINREG representation achieves the overall best result and perfoms best on 6 of 8 data sets. The difference between the ensemble result and LINREG is only 0.04, which has to be regarded as a very successful result for the rule extraction. LINREG is also the only technique that performs better than the mean prediction on all data sets. C&RT is the second best technique, mainly due to the very good result on the SLE data set. On three of the data sets, C&RT performs more or less equal to the mean prediction. FREE is the worst performing technique but with the exception of the DIA data set, it achieves similar to MREG and C&RT. DIA is the smallest of the data sets and seems to be the most difficult data set for G-REX.

A. Comprehensibility

Table 6 shows the average length of the rules produced by each technique. The length is calculated as the number of tokens in the rule.

TABLE 6
RULE LENGTH

	NORM	FREE	REG	MREG	C&RT
AUT	13.5	11.8	26.5	61	30.5
DIA	9.5	15.8	35.1	9	11
HOU	8.0	12.2	24.5	57	32
MAC	8.0	11.1	38.0	25	21.5
PHA	8.0	10.6	24.5	49	1.0
SLE	7.0	10.1	44.5	29	12.5
VET	10.5	11.1	32.9	29	1.0
WIS	11.0	18.5	38.0	129	1.5
MEAN	9.4	12.7	33.0	48.5	11.2

All rules produced by the different techniques have a reasonable size and could probably be regarded as comprehensible. C&RT produces extremely short rules for the data sets PHA,VET and WIS by guessing one value for all instances. Note that these data sets are the ones on which C&RT has the worst accuracy, see Tables 4 and 5.

B. Rules

In this sub-section typical rules for each technique are presented. Rules are only presented for the AUT data set as this probably is the data set that is easiest to comprehend. Each rule is close to the average length (over all data sets) for respective technique. AUT contains data of different properties of cars and the target is to predict the price of a certain car.

FIGURE 1 TYPICAL
NORM RULE FOR AUT

```
if cityMpg < 41.38
 |T: if weight < 2601.03
 |    |T: 7621.43
 |    |F: 16857.99
 |F: 7022.83
```

The rule has a length of 11 and predicts one of three values. CityMpg stand for miles per gallon in city traffic and weight is the curb weight of the car.

FIGURE 2
TYPICAL FREE RULE FOR AUT

```
if highwayMpg > 27.25
 |T:  horsepower * 105.65
 |F:  horsepower * 2.91 +
      engineSize * 113.29
```

Figure 2 shows a rule with length 14 produced by G-REX using the FREE representation. Depending on the highway mileage, the rule predicts a value as a product of one or two variables.

FIGURE 3
TYPICAL LINREG RULE FOR AUT

```
if losses > 184.34
 |T: weight*5.44+1913.26
 |F: if engineSize < 106.32
 |    |T: weight*4.77-2467.55
 |    |F: if losses < 106.32
 |    |    |T: weight*13.03-21334.75
 |    |    |F: weight*11.01-16536.69
```

The LINREG rule is slightly more complex and uses four linear regressions to predict the price. The total length is and 32. All regression uses the weight of the car as regressor and only differ in the values of the constants.

FIGURE 4
TYPICAL MREG RULE FOR AUT

```
-57183 +
-131.56 * symboling +
 10.889 * losses +
  131.4 * wheelBase +
-95.736 * length +
  828.3 * width +
 16.846 * height +
 6.1278 * weight +
 47.855 * engineSize +
-1687.5 * bore +
-1815.7 * stroke +
 103.57 * compressionRatio +
  15.65 * horsepower +
0.82669 * peakRpm +
-47.336 * cityMpg +
 28.109 * highwayMpg +
```

In the implementation used in this study, MREG always uses all input variables to make the prediction. This makes the rule complex and difficult to understand. On the other hand, the rule clearly shows the impact of each variable. For the AUT data set MREG produces rules with length 61.

FIGURE 5
TYPICAL C&RT RULE FOR AUT

```
if weight < 2664
|T: if weight < 2291.5
    |T: 7127.9
    |F: if losses < 180
        |T:9852.2
        |F:16677.5
|F: if weight < 3395.5
    |T: if width < 68.6
        |T: if losses < 162.5
            |T: if horsepower < 158
                |T: 14764.2
                |F: 18535
            |F: 18874.6
        |F: 20264.2
    |F: 30147
```

The C&RT rule in Figure 5 has a length of 36. The average length of the C&RT rule are 21 excluding PHA,VET and WIS.

V. Conclusions

All rules in the experiments can be regarded as comprehensible. The results show that G-REX outperformed MREG and C&RT on the data sets used in the study. The results also show that it is important to choose an appropriate representation. This is not a problem for G-REX, since it can use any representation language. In addition, it is easy to perform initial experiments in order to choose the most suitable.

Both LINREG and FREE have the same terminal and function set, which means that they search the same space of possible solutions. The fact that LINREG performs significantly better then FREE is probably due to the fact that LINREGs search is guided by the least square when the constants of the linear regressions are selected. In theory FREE should be able to achieve a performance similar to LINREG, but that would probably require a longer evolution and a larger population.

LINREG's advantage over MREG could either be explained by the fact that the ensemble aids LINREG in a favorable way, or that LINREG has a slightly more powerful representation, i.e. it can use different MREG for different parts of the data set. The same reasoning applies to C&RT, but both techniques are limited to their representation to calculate a solution, while G-REX can use almost arbitrary representations.

References

[1] R. Davis, B. Buchanan and E. Shortliffe, "Production Rules as a Representation for a Knowledge-Based Consultation Program", *Artificial Intelligence*, vol. 8, 1977.

[2] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[3] R. Andrew, J. Diedrich and A. B. Ticlke, "A survey and qritique of techniques for extracting rules from trained artificial neural networks.", *Knowledge-Based Systems*, vol. 8. 1995.

[4] U. Johansson, L Niklasson and R. König, "Rule Extraction from Trained Neural Networks using Genetic Programming" in *Supp. Proc. 13th International Conference on Artificial Neural Networks*, Istanbul, 2003, pp. 13-16.

[5] U. Johansson, L Niklasson and R. König, "The truth is in There - Rule Extraction from Opaque Models using Genetic Programming." In *Proc. 17th Florida Artificial Intelligence Research Symposium,* 2004, pp. 658-662.

[6] J. R. Koza, *Genetic Programming*. MIT Press, 2000.

[7] U. Johansson, C. Sönströd, R. König and L. Niklasson, "Neural Networks and Rule Extraction for Prediction and Explanation in the Marketing Domain." In *Proc. The International Joint Conference on Neural Networks*, Portland, 2003.

[8] M. H. Dunham, *Data Mining Introduction and Advanced Topics.* Prentice Hall, 2003.

[9] S. Chatterjee, A. S. Hadi, "Influential Observations, High Leverage Points, and Outliers in Linear Regression." *Statistical Science*, pp. 379-416, 1986.

[10] L. Breiman, *Classification and Regression Trees.* CRC Press, 1984.

[11] S. J. Armstrong, *Principles of forecasting.* Kluwer Academic Publishers, 2001.

[12] S. J. Armstrong and F. Collopy "Error measures for generalizing about forecasting methods: Emperical comparisons" *International Journal of Forecasting*, Vol. 8, 1992, pp. 69-80.

[13] X. R. Li and Z. Zhao "Relative Error Measures for Evaluation of Estimations algorithms" in *Proc. 7th International Conference on Information Fusion,* 2005, pp. 211-218.

[14] X. R. Li and Z. Zhao "Measures of Performance for evaluation of estimators and Filters", in *Proc. SPIE Conference on Signal and Data Processing.* 2001