MACHINE LEARNING APPROACHES FOR BIOMATERIAL MODELLING

by

DIPTEE MEHTA

(Under the Direction of Khaled Rasheed)

ABSTRACT

In this thesis, we explore the use of different machine learning approaches for predicting the amount of protein adsorption on biomaterial surfaces. The models built using machine learning approaches would be a valuable tool for biomaterial modeling. Feature selection is very important for this problem domain. Several filter based and wrapper based feature selection methods were tried. The wrapper based approach with a genetic algorithm gives the best results. Machine learning schemes like model trees, regression trees, radial basis function networks, neural networks, and locally weighted regression were applied in this problem domain. The neural networks with backpropagation give the best results in terms of performance on the original dataset with average values for the target attribute. The experiments suggest that finding the right feature subset is more important than finding the right machine learning scheme for this problem domain. The uncertainty handling capability of the machine learning schemes were also tested. It was found that the uncertainty handling should be incorporated in the feature selection process as well.

INDEX WORDS: biomaterial, model trees, regression trees, neural networks, radial basis function networks, genetic algorithm.

MACHINE LEARNING APPROACHES FOR BIOMATERIAL MODELLING

by

DIPTEE MEHTA

B.Tech, SNDT University, India, 2001

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2004

MACHINE LEARNING APPROACHES FOR BIOMATERIAL MODELLING

by

DIPTEE MEHTA

Major Professor:	Khaled Rasheed
Committee:	Liming Cai
	Don Potter

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2004

DEDICATION

To my parents

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Khaled Rasheed for his constant guidance, patience and advice and for encouraging me throughout my research. Dr. Rasheed has been very generous with his time. I would like to thank Dr. Don Potter and Dr. Liming Cai for being a part of my committee. This thesis used datasets obtained from the laboratory of Professor Joachim Kohn at the New Jersey Center for Biomaterials, Rutgers University. The assistance of Professor Kohn's laboratory is highly acknowledged. I am very thankful to Jack Smith as well for his help and guidance. I would also like to thank my entire family and my friends for their support. A special thanks to my brother Dinesh and my husband Amit for their constant encouragement.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1 MACHINE LEARNING

A popular definition of learning is "Any change in a system that allows it to perform better the second time on repetition of the same task or on another task drawn from the same population" (Simon, H 1983). According to this definition, the most important feature of a learning system is its improvement in performance or skill refinement. In living systems this is achieved through experience. Machine Learning is the study of computer algorithms that improve performance automatically through experience (Mitchell, 1997). One way to improve performance is to improve its prediction capability.

At the heart of performance is prediction. An algorithm that—when presented with data that exemplifies a task—improves its ability to predict key elements of the task can be said to have *learned*. Machine learning can be thought of as a search through a space of possible hypotheses for one that fits the given data. The input to a machine learning algorithm takes the form of concepts, instances and attributes. The thing to be learned is called a concept description, the information that the algorithm is given takes the form of a set of instances. Thus each instance is an independent example of the concept to be learned. Each instance is characterized by the values of attributes or features that measure different aspects of the instance.

Based on the training data provided, machine learning can be broadly of two types: **supervised** and **unsupervised learning** (Witten I.; Eibe F., 2000). One of the common forms of supervised learning is learning from labeled examples. After defining classes, a number of

positive and negative examples of objects belonging to these classes are provided to the learning system. The system is then to find out common properties of the different classes, and what separates them, in order to make correct classification for other new unseen objects. This is called classification learning. If the learning system has to predict a numeric quantity based on the given examples and their numerical outcomes, it is called numeric prediction. Both classification learning and numeric prediction are types of supervised learning. In the case of **unsupervised learning**, no classes are defined a priori. Thus, the system itself must find some way of clustering the objects into classes, and also find descriptions for these classes. This type of learning is also called association learning. This thesis concentrates on supervised learning.

1.2 BIOMATERIAL MODELLING

A biomaterial (Hench; Polak, 2002) is defined as "A biocompatible material that is used to construct artificial organs, rehabilitation devices, or prostheses and replace natural body tissues."

Biomaterials are employed in plastic and reconstructive surgery, used to make the tools needed to examine the human body, or to cure the deficiency of an organ. Hence, biomaterials must be biologically compatible with the organism and need to interoperate well with the body tissue. Dental crowns and contact lenses that we see in our everyday life, for example use biomaterials.

Polymers are very widely used as biomaterials. A polymer is defined as a compound formed by the joining of smaller, usually repeating, units linked by covalent bonds. In implant applications, typically involving tissue regeneration, adsorption of protein to these polymer surfaces is critical in cell attachment and hence a major determinant of the suitability of the material (Magnani, A 2002). There are several techniques for measuring protein adsorption to polymer surfaces. However conducting these experiments is very laborious and time consuming.

In addition these methodologies do not provide details concerning the molecular-scale properties of biomaterials relevant to protein/surface interactions. Thus a semi empirical model for selecting the materials for detailed studies would be very useful and helpful.

Applying machine learning techniques to predict the amount of protein adsorption on polymer surfaces would then lead to making a selection of materials which could further be analyzed and tested in detail. The polyarylate library, that forms the training dataset for the machine learning algorithms, has been developed by Fiordeliso et al. (Fiordeliso, 1995). Each polyarylate is described with the help of 109 molecular-level descriptors of polymer chain structure. The entire list of these descriptors is included in the Appendix and their detailed description is beyond the scope of this thesis. The interested reader is referred to the Molecular Operating Environment software (MOE, 20003) and the Dragon software (Dragon, 2003).

1.3 RELATED WORK

The previous work in biomaterial modeling has been done mostly from a domain point of view by conducting experiments *in vitro*. Such experiments are very expensive. However there have been a few approaches that attempt to develop a surrogate model using techniques that are different from the traditional techniques from the chemical and biological domain.

Previously, a tool called Logical Analysis of Data (LAD) was used to develop a surrogate model of cellular growth for biomaterials (Abramson S. et al. 2002). LAD is a knowledge extraction method for data analysis based on combinatorics, optimization and Boolean logic. LAD was initially created for the classification of binary data (Hammer, 1986; Crama, Hammer et al. 1988), but was later extended to datasets having positive and negative observations (Boros, Hammer et al. 1997) depending on continuous variables. It is based on the concept of finding positive and negative patterns in the dataset. A positive pattern is a conjunction of conditions on

the values of the attributes, which is satisfied (covered) by a sufficiently large proportion of positive instances in the dataset. At least one condition of the conjunction corresponding to a positive pattern must be violated by all the negative instances in the dataset. The LAD model consists of two sets of patterns, forming a positive and negative theory respectively. If a new observation is covered by a pattern in the positive theory and none in the negative theory, it is classified as positive; and if the observation is covered by a pattern in the negative theory and none in the positive theory, it is classified as a negative example. The entire dataset was labeled as weak and strong observations based on the target attribute value. The predictive power of the model was tested using the cross validation technique.

The idea of applying machine learning techniques for biomaterial modeling has been previously done by Jack et al (Smith J., 2004). He had used decision trees to screen the most relevant attributes from all the attributes. The target attribute was discretised using equal frequency binning and then a decision tree was used to identify the most important attributes. The experiment was repeated several times by varying the target attribute value for each instance in the dataset within a normal distribution defined by the experimental mean and standard deviation. From the results of these experiments, the three attributes occurring the most number of times in the top portion of the tree were selected as the attributes to be used by a backpropagation neural network.

In this thesis, we have explored the use of other feature selection methods as well. Feature selection methods like Relief, Correlation based feature selection, Principal Component Analysis, decision trees and genetic algorithm based approach were tried. We have also explored the use of different machine learning schemes like model trees, regression trees, radial basis function networks, locally weighted regression and neural networks. Different numbers of

attributes were used with the above mentioned machine learning schemes and the results were compared.

In our approach, we propose the use of genetic algorithms for feature selection in this problem domain. The idea of using genetic algorithms for feature selection was introduced by Siedlecki and Sklanski (Siedlecki and Sklanski, 1989). In their approach, the individuals of the population were represented by bit vectors. If the i^{th} bit of this vector equals 1, then the i^{th} feature is used to participate in classification; if the bit is a 0, then the corresponding feature does not participate. Each resulting subset of features was evaluated according to its classification accuracy on a set of testing data using a nearest-neighbor classifier.

1.4 GOALS OF THIS THESIS

This thesis aims at applying machine learning techniques to predict the amount of protein adsorption to biomaterial surfaces based on their molecular and chemical properties. Each biomaterial is described with the help of 109 descriptors and the amount of different proteins adsorbed by it. The number of training examples is very small. Feature subset selection can dramatically improve the performance of a machine learning algorithm. Thus coming up with the optimal number and the best feature subset for a machine learning algorithm is very important.

Also, the amount of the protein adsorption of the training examples is based on the outcomes of several experiments done *in vitro* and hence this problem domain has uncertainty in the values of its target attributes. Any machine learning algorithm needs to avoid over-fitting or under-fitting of the data and should be evaluated on the basis of its ability to predict unseen data not involved in the training process.

Specifically, the objectives of this thesis were:

1) Analyzing the data and coming up with the best feature subset using different methods.

2) Finding the machine learning algorithms that would be useful in the problem domain of biomaterial modeling.
3) Implementing the proposed machine learning algorithms.
4) Measuring the degree of robustness of the proposed machine learning algorithms to the uncertainties in the data.
5) Evaluating the predictive accuracy of the proposed machine learning models on the basis of their ability to predict unseen data not involved in the training process.

CHAPTER 2

FEATURE SELECTION

2.1 INTRODUCTION

For a prediction problem, the number of independent pieces of information that go into the estimate of a parameter is called the degrees-of-freedom (df). Degrees-of-freedom is defined as the number of independent components (number of instances) minus the number of parameters estimated (usually at least equal to the number of attributes or features). When the number of features in a dataset is larger than the number of observations or instances, this dataset is said to lack degrees-of-freedom and thus is over-fit. Thus degrees-of-freedom is the number of pieces of useful information in the dataset.

In the biomaterials dataset, each polymer is described with the help of 109 attributes and the number of polymers in the Fibrinogen dataset is 45 and in the NMA-NFF dataset is 93. Thus in this case, we have negative degrees-of-freedom (45-109 and 93-109). Thus feature subset selection is very important for this problem. By doing feature selection, we can identify the most salient features for learning and the machine learning algorithm could then focus on those aspects of the data most useful for analysis and prediction. The benefits of feature selection for learning include a reduction in the amount of data needed to achieve learning, improved predictive accuracy, learned knowledge that is more compact and easily understood, and reduced execution time.

Feature subset selection algorithms fall into two broad categories: One is to make an independent assessment of the feature subset based on the general characteristics of the data and the other is to assess the subset using the machine learning algorithm that will be used on the data. The former approach is called a *filter approach* because the irrelevant attributes get filtered out and the filtered data is then presented to the machine learning algorithm. Figure 1 depicts the way in which filter based approaches work.

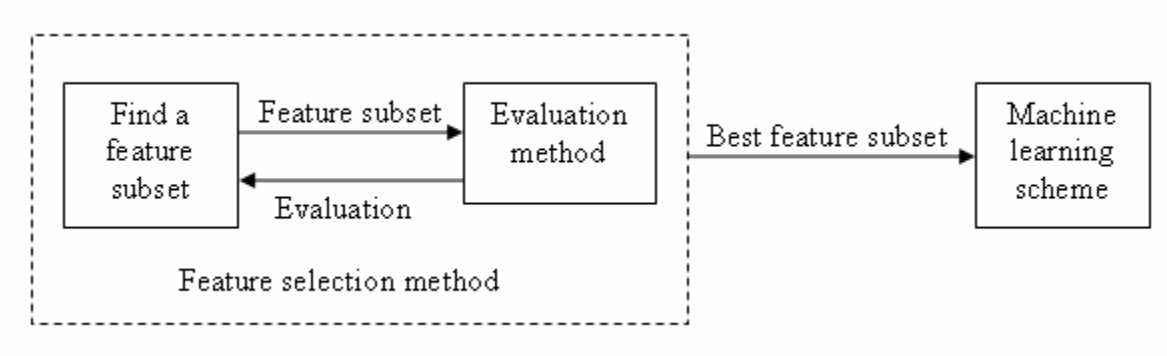

Figure 1: The filter based approach of feature selection

The second approach is called a *wrapper approach*, because the machine learning algorithm is involved and wrapped in the feature selection process. Figure 2 depicts the way in which wrapper based approaches work.

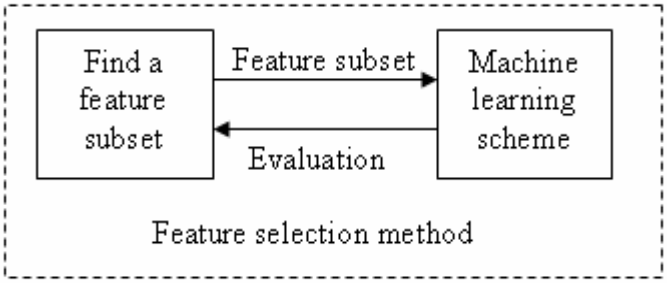

Figure 2: The wrapper based approach of feature selection

2.2 GENETIC ALGORITHM AS THE WRAPPER APPROACH

Feature subset selection involves searching the space of all possible feature subsets to find the subset that gives the optimal performance for a machine learning algorithm. The number of

possible subsets of attributes increases exponentially with the number of attributes, making exhaustive search impractical even for problems with a small number of features. Genetic algorithms, developed by Holland, are search algorithms based on the mechanics of natural selection and genetics (Goldberg, 1989) and have proven to give substantial improvement over several random and local search methods in many domains. Thus they would be ideal for feature subset selection for biomaterial modeling.

Genetic algorithms

Genetic algorithms are a class of algorithms that imitate natural selection and genetics. A genetic algorithm has five main components: representation of a solution (or individual), genetic operators (i.e. crossover and mutation operators), a parent selection strategy, a fitness function and a replacement strategy.

Genetic algorithms start by generating a population of solutions, also called individuals. Each individual is represented by a genetic string composed of genes and is associated with a fitness value, which represents the quality of the solution or the individual. The fitter (or better) individuals in a population are encouraged to reproduce to create a fitter offspring in the next population by using genetic operators like mutation and crossover on the individuals of the population.

The *crossover* operator takes two selected individuals and combines them about a crossover point thereby creating two new individuals. The new individual generated gets some portion of its genes from each of its parent.

The *mutation* operator randomly modifies the genes of an individual introducing further randomness into the population and leading to exploring new areas of the search space of all possible solutions.

There are two flavors of genetic algorithms based on the replacement strategy: steady state genetic algorithms and generational genetic algorithms. In steady state genetic algorithms, the newly created individual is replaced with an individual from the old population and in generational genetic algorithms a new population is formed with the newly created individuals. Figure 3 shows the typical structure of a genetic algorithm.

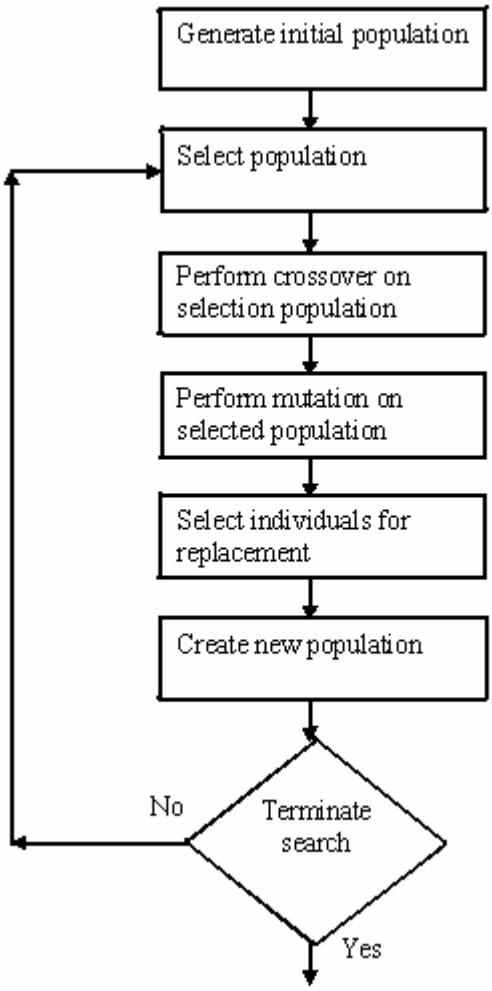

Figure 3: The basic structure of a genetic algorithm

2.3 THE FILTER APPROACH

This approach is also called unsupervised feature selection as the feature selection process is done independently of the machine learning algorithm to be applied. Some of the filter

approaches only give the ranking of the attributes and the optimal number of attributes to be used for learning is left to the user. In such cases, different values for the number of attributes were tried. The following filter approaches for feature selection were tried in the problem of biomaterial modeling:

2.3.1 CORRELATION BASED FEATURE SELECTION

It is a filter approach for feature selection that uses a correlation based heuristic to evaluate the merit of features. It gives the optimal feature subset using a search algorithm. The hypothesis on which feature selection is based is as follows (Hall, M 1999): "Good feature subsets contain features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other."

The following equation formalizes the heuristic:

$$G_s = \frac{kr_{ci}}{\sqrt{k + k(k-1)r_{ii}}}$$

Where k is the number of features in the subset, r_{ci} is the mean feature correlation with the class and r_{ii} is the average feature inter-correlation and G_s is the goodness of a feature subset. Figure 4 gives a diagrammatic representation of this feature selection method:

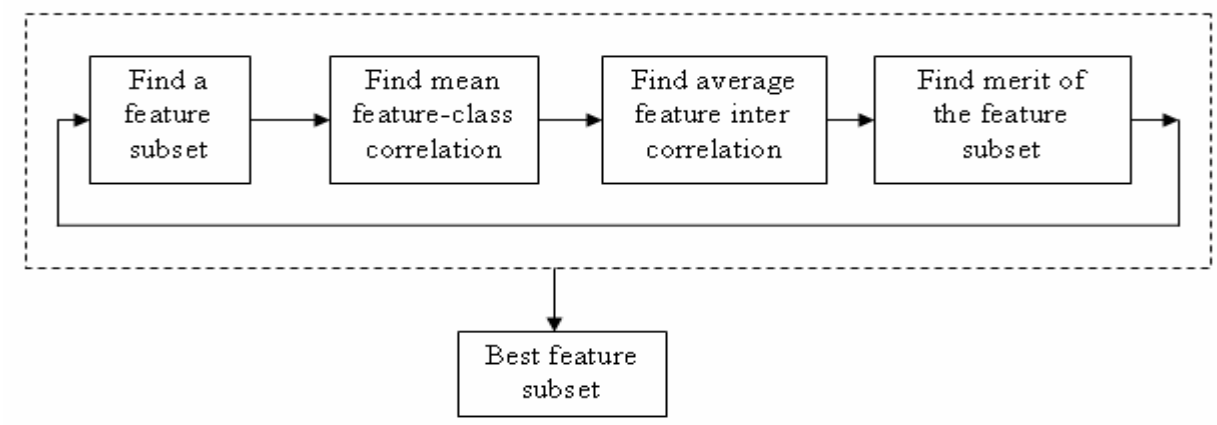

Figure 4: Correlation based feature selection method.

Thus in this method irrelevant features will be filtered out as they will be poor predictors of the class and redundant features will be filtered out as they will be highly correlated with some other features. In this method a way to measure the feature-feature correlation and feature-target class is needed and we need to decide on some method for searching the space of feature subsets.

Searching the space of all feature subsets in an exhaustive way is impractical for most of the problems as the number of possible feature subsets increases exponentially with the number of attributes. Various heuristic search strategies such as hill climbing and Best First search (Rich and Knight, 1991) are often applied to search the feature subset space in a reasonable amount of time. Two forms of hill climbing, forward selection and backward elimination can, be used. In forward selection, we start with no attributes and add them one at a time and in backward elimination, we start with the full set and delete attributes one at a time. Whenever an attribute is added or removed, the resulting feature subset is evaluated using the above discussed measure. In forward selection, the effect of adding each possible attribute to the subset is evaluated and the one with the best evaluation is chosen and the search continues. The same thing is done analogously in backward elimination. However if no attribute addition or deletion produces an improvement, the search ends. In best first search we start with an empty set of features and generate all possible single feature expansions. The subset with highest evaluation is chosen and is expanded in the same manner. If expanding a feature subset does not result in an improvement, it revisits the best unexpanded feature subset and continues the search from there. To prevent it from doing an exhaustive search, it is common to limit the number of subsets expanded that result in no improvement.

Different information based measures of association were tried for finding feature-feature and feature-class correlations. Best results were achieved by using the gain ratio for feature-class correlations and symmetrical uncertainty coefficients for feature inter-correlations. All the continuous features were converted to nominal by binning before proceeding with the correlation calculations. If the target value in the dataset was a continuous value, it was also converted into a nominal value by binning.

The information gain, Gain(S, A) of an attribute A relative to a collection of examples S is defined as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Where Values (A) is the set of all possible values for attribute A and S_v is the subset of S for which attribute A has value v. Here the entropy is the purity or impurity of a collection of samples and is defined as

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Where p_i is the proportion of S belonging to class i.

Information gain is biased in favor of attributes that can take a large number of possible values. The gain ratio (Quinlan, J, 1986) is a measure that tries to compensate for this bias. The gain ratio, Gain Ratio(S, A) of an attribute A relative to a collection of examples S is defined as

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInformation(S, A)}$$

Here Split Information is the term that decreases the value of Gain Ratio for attributes with many uniformly distributed values and is defined as

$$SplitInformation(S, A) = \sum_{i=1}^{c} \frac{S_i}{S} \log_2 \frac{S_i}{S}$$

Where S_1 through S_c are the c subsets of examples resulting from partitioning S by the c-valued attribute.

Symmetry is desirable property for a measure of feature-feature inter-correlation. For a nominal feature Y, a probabilistic model can be formed by estimating the probabilities of the different values of Y from the training data. If this model is used to predict the value of Y for some other instance, then the entropy of this model and hence the attribute is

$$H(Y) = \sum_{y} p(y) \log_2 (p(y))$$

If the observed values of Y in the training data are partitioned according to the values of a second feature X, the entropy of Y after observing X is given by

$$H(Y \mid X) = \sum_{x} p(x) \sum_{y} p(y \mid x) \log_2 (p(y \mid x))$$

The amount by which the entropy of Y decreases reflects additional information about Y provided by X. This is again biased for features having greater number of possible values. Symmetrical uncertainty coefficient (Press W, 1998) is a measure that tries to compensate for the bias in information gain and normalizes its values in between 0 and 1 and is defined as follows

$$SymmetricalUncert = 2.0 \times [H(Y) - H(Y \mid X)] / [H(Y) + H(X)]$$

Both gain ratio and symmetrical uncertainty coefficient lie between 0 and 1. A value of 0 for gain ratio indicates that the particular feature is a poor predictor of the class and a value of 1 indicates that it is a very good predictor of the class. Similarly a value of 0 for the symmetrical uncertainty coefficient indicates that the two features have no correlation and a value of 1 indicates that the knowledge of one completely predicts the other.

2.3.2 RELIEF-F FEATURE SELECTION

RELIEF is a feature selection method proposed by Kira and Rendell (Kira and Rendell, 1992). It produces a ranking of all the attributes. The basic idea is to measure the relevance of features in the neighborhoods around target samples.

The original algorithm of RELIEF(Kira and Rendell, 1992) is the following

Set all weights W[A]= 0.0

For i = 1 to m do

Begin

Randomly select an instance R;

Find nearest hit H and nearest miss M

For A = 1 to all_attributes do

W[A] = W[A] – diff (A, R, H)/m + diff (A, R, M)/m

End

Where nearest hit is a nearest neighbor from the same class, nearest miss is a nearest neighbor from the other class and diff(Attribute, Instance1, Instance2) calculates the difference between the values of Attribute for two instances.

For discrete attributes,

diff(Attribute, Instance1, Instance2) = 0 if Val(Attribute, Instance1) is equal to Val(Attribute, Instance2)

diff(Attribute, Instance1, Instance2) = 1 if Val(Attribute, Instance1) is not equal to Val (Attribute, Instance2)

For continuous attributes,

diff(Attribute,Instance1,Instance2)=[Val(Attribute,Instance1)-Val(Attribute,Instance2)]/ RangeOfAttribute

For continuous attributes, we divide the difference in the values by the range so that the difference value is normalized to the interval [0, 1]. We then divide the value of difference by the value of m, so that all the weights are in the interval of [-1, +1]. If the training set is small, instead of using only m instances, the same procedure can be repeated for all the instances. The larger the value of m, the more reliable the approximation we get.

The original RELIEF algorithm described above can deal with both discrete and continuous attributes however is limited to only two class problems.

The algorithm was extended by Kononenko (Kononenko, 1994) to deal with multi-class and continuous prediction problems. The extended algorithm called RELIEF-F is more robust to incomplete and noisy data. Instead of finding a single nearest hit and a single nearest miss, RELIEF-F finds k nearest hits and k nearest misses and averages their contribution to W[A]. Thus relevant features start accumulating high positive weights and less relevant features retain their weight close to zero.

In continuous prediction problems, the notion of nearest hits or nearest misses cannot be used, as the value to be predicted is not divided into discrete classes, but is a continuous value. Thus the probability that the predicted values of two instances are different is used. This probability is modeled with the relative distance between the predicted values of two instances.

The modified RELIEF-F algorithm (Sikonja & Konenko, 1997) is as follows:

Set all N_{dC} ,$N_{dA}[A]$,$N_{dC\&dA}[A]$, W[A] to 0

For i = 1 to m do

Begin

Randomly select instance R_i

Select k instances I_j nearest to R_i.

For j=1 to k do

Begin

$N_{dC} = N_{dC} + |f(R_i) - f(I_j)| .d(i,j)$

For A = 1 to allAttributes do

Begin

$N_{dA}[A] = N_{dA}[A] + diff(A, R_i, I_j) .d(i, j)$

$N_{dC\&dA}[A] = N_{dC\&dA}[A] + |f(R_i) - f(I_j)|.diff(A,R_i,I_j).d(i,j)$

End

End

End

For A=1 to allAttributes do

$W[A] = N_{dC\&dA}[A] / N - (N_{dA}[A\} - N_{dC\&dA}[A]) / (m - N_{dC})$

In the above algorithm, N_{dC} stands for the probability that two instances have different prediction or different class, N_{dA} stands for the probability that two instances have different values for attribute A, $N_{dC\&dA}$ stands for the probability that two instances have different prediction and have different values for the attribute A, f(Ij) is the prediction or the target value for instance j and d(i,j) is the distance between two instances whose indexes are i and j.

2.3.3 PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis can be used to transform the original large number of features into a smaller number of features. Principal Component Analysis is a way of identifying patterns in data and expressing the data in such a way as to highlight their similarities and differences.

Once these patterns are found in the data we can compress the data without much loss of information.

The main steps in performing principal component analysis are as follows:

a) After getting the data, calculate the mean value for each of the features. We then have to subtract this mean from the attribute values for all the instances. This produces a dataset whose mean is zero.

b) The next step is to calculate the covariance matrix. If we have n features, we get an n × n covariance matrix. The $(i,j)^{th}$ element of this matrix denotes the covariance between feature i and feature j. Covariance is a measure of how much the two features vary from the mean with respect to each other. It is calculated using the following formula:

$$\operatorname{cov}(X,Y) = \sum_{i=1}^{n} \frac{(X_i - X_{mean})(Y_i - Y_{mean})}{n-1}$$

Where X_i is the value of the attribute X for instance i, Y_i is the value of the attribute Y for instance i, X_{mean} is the mean of attribute values of X for all the instances, Y_{mean} is the mean of attribute values of Y for all the instances and n is the number of instances in the dataset.

The exact value of the covariance is not as important as its sign (i.e. positive or negative). If the value is positive, then the values for both the features increase together. If the value is negative, then as the value for one feature increases, the other decreases. If the value is zero, it indicates that the two features are independent of each other. Since cov(X,Y) is same as cov(Y,X), the covariance matrix is symmetric about the main diagonal.

c) We then have to calculate the eigenvectors and eigenvalues of the covariance matrix. Eigenvectors can be found only for square matrices and are always orthogonal to each other. Thus by this process of taking the eigenvectors, we are able to express the data in terms of these orthogonal eigenvectors.

d) We now have to transform the data using the eigenvectors and eigenvalues found in the previous section. After finding the eigenvectors we have to order them by eigenvalues highest to lowest, which gives us the components in order of significance. By ignoring the components of less significance, we get the final dataset with smaller dimensions than the original dataset.

e) The final step is transforming the dataset using the components selected in the earlier step. This is done as follows:

$$FinalDataset = RowFeatureVector \times Dataset$$

Where RowFeatureVector is a matrix formed with the selected eigenvectors placed in the rows, Dataset is the mean-adjusted dataset transposed (each row holding values for a feature) and the FinalDataset is the new transformed dataset with reduced number of features.

2.3.4 DECISION TREES

Machine learning schemes can also be used for feature selection. Thus decision trees were first applied to the full dataset and only the attributes occurring in the top portion of the tree were selected for use by other machine learning schemes. In this scheme the user has to determine how many attributes are to be selected from the top portion of the decision tree.

Decision tree learning is a method for approximating discrete valued target functions, in which the learned function is represented by a decision tree. To classify an unseen instance, the tree is followed down to a leaf and the classification at that leaf is assigned to the unseen instance. Decision tree algorithms follow a top-down, greedy search through a space of possible decision trees. They are based on the divide-and-conquer strategy. Nodes in a decision tree involve testing a particular attribute. Leaf nodes give a classification that applies to all instances that reach the leaf. A sample decision tree for the Fibrinogen dataset is shown in Figure 5. The continuous target value is converted into a discrete value represented by values from the set

{One, Two, Three, Four, Five, Six}.The internal nodes represent the attributes and the leaf nodes represent the target classification.

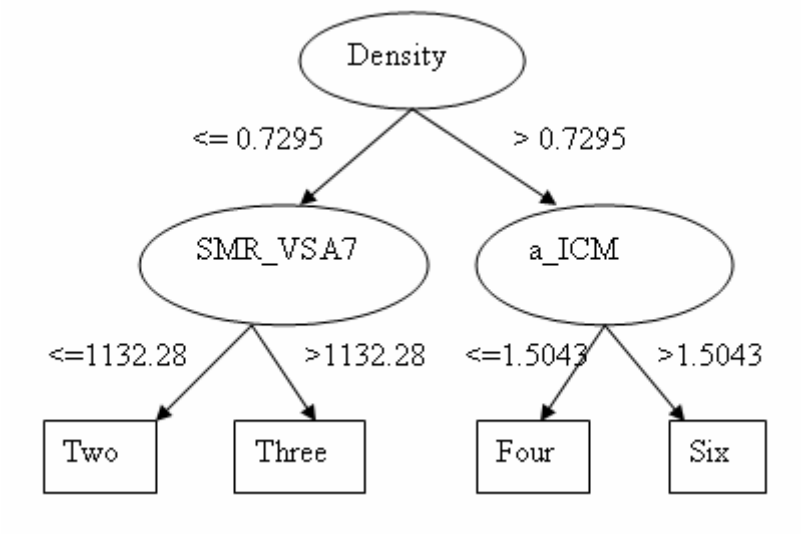

Figure 5: A sample decision tree for the Fibrinogen dataset.

The algorithm for constructing a decision tree can be expressed recursively (Quinlan, 1986). We first select an attribute to place at the root node and make one branch for each of its possible values. This spits up the example set into subsets, one for every value of the attribute. We then repeat the process recursively for each branch, using only those instances that actually reach the branch. When all instances at a node have the same classification, we stop developing that part of the tree and create a leaf node.

The most important part is selecting which attribute to test at each node in the tree. The definition of gain ratio, mentioned in section 2.3.1, is used for this purpose.

CHAPTER 3

MACHINE LEARNING ALGORITHMS

Several machine learning algorithms were tried. Two machine learning schemes, in which the learned representation was a tree structure, were explored: model trees and regression trees. Both these schemes are useful for continuous prediction problems. Locally weighted regression which is a type of instance based learning or lazy learning was also tried. Backpropagation neural networks, which are capable of learning complex decision surfaces, were also explored. Radial basis function networks, which are a blend of neural networks and instance based learning, were also explored.

3.1 MODEL TREES

Model trees are decision trees with linear regression functions at the leaf nodes instead of terminal class values. Thus they offer a useful approach for predicting numeric quantities. Model trees are particularly useful when most of the attributes are numeric. The M5 algorithm (Quinlan, J, 1992) proposed by J. R. Quinlan for model trees, is described below. When a model tree is used to predict the target value for an unseen instance, the tree is followed down to a leaf in the same way as in a decision tree, using the instance's attribute values to make routing decisions at each node. The leaf contains a linear model based on some of the attribute values, and this is evaluated for the test instance to yield a predicted value. However instead of using this value directly, a smoothing process is used to compensate for the sharp discontinuities that will

inevitably occur between adjacent linear models at the leaves of the pruned tree, particularly for some models constructed from a smaller number of training examples.

Smoothing can be accomplished by producing linear models for each internal node, as well as for the leaves, during the time the tree is built. A linear model is created at each node of the tree, using standard regression techniques, however using only the attributes that are tested in the sub-tree below this node. Then, once the leaf model has been used to obtain the raw predicted value for a test instance, that value is filtered along the path back to the root, and is combined with the value predicted by the linear model for that node.

To combine the values at all nodes, the following smoothing calculation (Frank et al, 1998) is used

$$p' = \frac{np + kq}{n + k}$$

where p' is the prediction passed up to next higher node, p is the prediction passed to this node from below, q is the value predicted by the model at this node, n is the number of training instances that reach the node below, and k is a smoothing constant. Quinlan proposes a value of 15 to be used for the smoothing constant (Quinlan, 1992). Experiments show that smoothing substantially increases the accuracy of predictions. Model trees can have multivariate linear models at the leaves and therefore resemble piecewise linear functions.

Model trees, like the decision trees are constructed by the divide-and-conquer approach. The set of training examples, T is either associated with a leaf, or some test is chosen that splits T into subsets corresponding to the test outcomes and the same process is applied recursively to the subsets.

The first step in building the tree is computing the standard deviation of the target values of the training examples in T. This is used as a measure of the error at that node. We then

calculate the expected reduction in this error as a result of testing an attribute at that node. The attribute which maximizes the expected error reduction is chosen for splitting at that node.

The expected error reduction, also called standard deviation reduction (SDR), is calculated by,

$$SDR = sd(T) - \sum_i \frac{|T_i|}{|T|} \times sd(T_i)$$

Where T_1, T_2,.. are the sets that result from splitting the node according to the chosen attribute.

The splitting process terminates when the target values of the instances that reach a node vary only slightly, or when just a few instances reach the node.

3.2 REGRESSION TREES

Regression trees may be considered as a variant of decision trees, designed to approximate real-valued functions instead of being used for classification tasks. They were proposed by Breiman (Breiman , 1984).

The inner nodes of regression trees are marked with tests as in decision trees. The difference is, that the leaves of regression trees may be marked with arbitrary real values, whereas in decision trees the leaves may only be marked with elements of a finite set of nominal values. Thus the leaves in a regression tree do not contain a nominal value, but contain a numeric value which is the average of all the training set values that reach that leaf. The term regression means computing an expression that predicts a numeric quantity. Thus the decision trees with averaged numeric values at the leaves are called regression trees.

Regression trees are built the same way as model trees, however the linear regression step at the leaves is omitted and the target value at the leaf is taken to be the average target value of the training examples that reach this leaf.

3.3 NEURAL NETWORKS

An artificial neural network (ANN) is a common method used for predicting numeric quantities. The backpropagation algorithm (Rumelhart, 1986) is the most commonly used Artificial Neural Network learning technique.

The multilayer networks learned by the Backpropagation algorithm are capable of expressing a rich variety of non linear decision surfaces. The sigmoid unit forms the basic unit of multilayer networks.

A sigmoid unit is a type of threshold unit that has a smooth threshold function, rather than a step function. This unit takes as input the weighted sum, S, of the values coming from the units connected to it. The function inside sigmoid units calculates the following value, given a real-valued input S:

$$\sigma(S) = \frac{1}{1 + e^{-S}}$$

The advantage of the sigmoid transfer function is that it is differentiable. This makes it suitable for use in multi-layer networks. Multilayer networks are formed by connecting several sigmoid units together. The attribute values of the training examples are applied to the input layer and the output layer gives the output of the whole network. In addition to these input and output layers, there are usually one or two *hidden layers.* Figure 6 shows two neural networks, one which has a single hidden layer and the other which has two hidden layers.

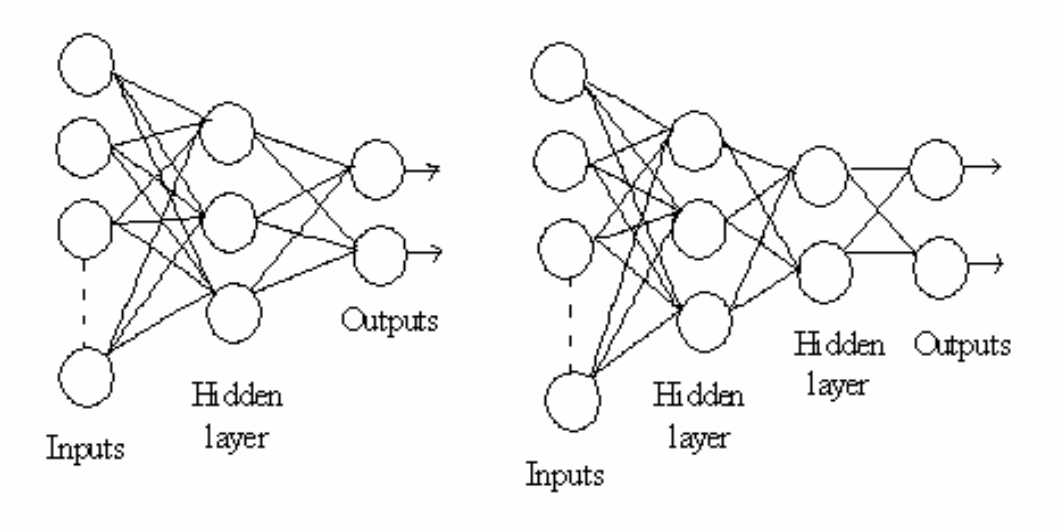

Figure 6: Neural networks with one and two hidden layers.

The Backpropagation algorithm

The task of the algorithm is to learn a multilayer ANN to correctly categorize the unseen examples. The training occurs in a supervised style. The basic idea is to present the input vector to the network; calculate in the forward direction the output of each layer and the final output of the network. For the output layer the desired values are known and therefore the weights can be adjusted as for a single layer network; in the case of the Backpropagation algorithm according to the gradient decent rule. To calculate the weight changes in the hidden layer the error in the output layer is back-propagated to these layers according to the connecting weights. This process is repeated for each sample in the training set. One cycle through the training set is called an epoch. The number of epochs needed to train the network depends on various factors, especially on the complexity of the problem at hand.

Thus for a network with a single hidden layer, the weight updates in the neural network takes place in two phases:

- *Weight updates from hidden to the output layer*

This weight update takes place by minimizing the squared error between the network outputs values and the target values for these outputs. A particular output node has no effect on any error made by other output nodes, it is only responsible for its own errors.

- *Weight updates from the input to the hidden layer*

  Hidden nodes themselves do not make errors; they contribute to the errors of the output nodes. The derivative of the network's error with respect to a hidden node's output is therefore the sum of that hidden node's contributions to the errors of all the output nodes.

The above mentioned weight updates are done as follows (Smith, 1993):

$$a_{i,j}^{(n)} = a_{i,j}^{(n-1)} + \Delta a_{i,j}^{(n)}$$

Where $a_{i,j}^{(n)}$ is the weight from node i to node j in the n^{th} epoch, $a_{i,j}^{(n-1)}$ is the value of the same weight in the $(n-1)^{th}$ epoch and $\Delta_{ai,j}^{(n)}$ is the weight change in the n^{th} epoch.

The weight change is calculated as follows:

$$\Delta a_{i,j}^{(n)} = \eta \frac{\delta E}{\delta a_{i,j}} + \mu \Delta a_{i,j}^{(n-1)}$$

Where E is the squared difference between the actual values and the predicted values for all the training instances, η is learning rate, $\Delta_{ai,j}^{(n-1)}$ is the weight change in the $(n-1)^{th}$ epoch, and μ is the momentum.

3.4 LOCALLY WEIGHTED REGRESSION

Locally weighted regression is a type of instance based learning. Learning in these algorithms consists of simply storing the presented training data. When a new query instance is encountered, a set of similar related instances is retrieved from memory and used to classify the new query instance. These methods are also known as lazy learning methods, because they delay the processing until a query instance has to be classified.

Locally weighted regression attempts to approximate a real valued target function over a local region surrounding a query instance. The method is called "locally weighted regression" as it tries to approximate the function based only on data near the query point *i.e. "local"*, it weighs the contribution of each training example from the query point by its distance *i.e. "weighted*" and it is trying to approximate real-valued function, *i.e. "regression"* (Mitchell, 1997).

Given a new query instance, the general approach in locally weighted regression is to find the training examples in the neighborhood of the query instance based on some distance measure and then constructing an approximation that fits this neighborhood surrounding the query instance. While constructing an approximation, it weighs the training instances according to their distance to the query instance. Thus, training instances close to the test instance receive a high weight and those far away a low one. The approximation is then used to calculate the target value for the query instance.

In order to use locally weighted regression, it is necessary to decide upon a distance based weighing scheme for the training instances. A common choice for finding the distance between instances is the Euclidean distance. There are several choices for the weighing function also. The requirements of a good weighing function are (Atkeson, 1996): The maximum value of the weighing function should be at zero distance and the function should decay smoothly as the distance increases. Some of the commonly used weighing schemes are as follows:

- Linear: This weighing function is as follows:

  $K(d) = Max(1 - d, 0)$

  Here d is the distance between two instances and the Max function returns the larger of its two arguments.

- Inverse: This weighing function is as follows:

$$K(d) = \frac{1}{1+d^p}$$

Here p decides how local the regression will be and d is the distance between two instances.

- Gaussian: The weighing function is as follows:

$$K(d) = \exp(-d^2)$$

Here d is the distance between two instances.

In this thesis, we are using the linear weighing scheme and all the instances in the dataset are used to predict the target value for a given query instance.

3.5 RADIAL BASIS FUNCTION NETWORKS

Radial basis function networks (RBF networks) represent an interesting blend of instance-based and neural network learning algorithms (Broomhead and Lowe, 1988). In this approach, the learned hypothesis is a function that is a linear combination of a set of m fixed functions, often called basis functions (Mitchell, 1997). It is of the following form:

$$f'(x) = w_0 + \sum_{i=1}^{m} w_u K_u[d(x_u, x)]$$

The flexibility of f', its ability to fit many different functions, derives from the freedom to choose different values for the weights and the appropriate kernel functions.

The basis function is defined so that it decreases as the distance $d(x_u, x)$ increases. Thus even though f'(x) is a global approximation to f(x), the contribution from each of the basis functions is localized to a region near the point x_u. It is common to choose each of the basis functions to be a Gaussian function centered at point x_u with some variance σ_u^2.

Radial basis function networks have been traditionally associated with radial functions in a single-layer network as shown in Figure 7. Each attribute of the instances in the training data

feed forward to the m basis functions whose outputs are linearly combined with weights to form the network output f(x).

Given a set of training instances of the target function, RBF networks are trained as follows: First the number m of hidden units or basis functions is determined and then each hidden unit is defined by choosing the values of x_u and σ_u^2. Next the network is trained and the weights are determined to maximize the fit of the network to training data by minimizing the squared error between the actual and predicted values of the target attribute.

The number of basis functions can be less than the number of training examples. Clustering is applied to instances to identify groups of similar instances and a basis function can be added centered at each cluster. The target attribute value of the instances does not enter into the calculation of number of hidden units and in choosing the kernel centers. They are only used to calculate the output layer weights.

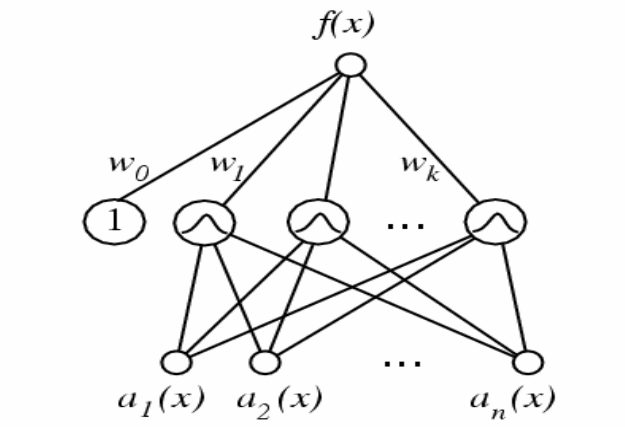

Figure 7: A traditional single layer radial basis function network. (Mitchell, 1997)

CHAPTER 4

IMPLEMENTATION

4.1 GENETIC ALGORITHM AS THE WRAPPER APPROACH

A genetic algorithm was written in JAVA whose details are given below:

Representation

Each individual in a genetic algorithm is a solution to the problem. We are using the genetic algorithm to find the best feature subset for a given machine learning scheme and for a given number of features. Thus each individual is a subset of features and is represented by an array of floating point numbers. The length of the array is equal to the total number of features i.e. 109. Each number in this array can take a value between 0 and 1. If we are using the genetic algorithm to find the best feature subset containing say 10 features, the 10 largest numbers from the array are selected and the features corresponding to these will form the feature subset. If we are using the genetic algorithm to find the best feature subset containing 20 features, the features corresponding to the 20 largest numbers are used. Thus the features corresponding to the largest numbers in the array form the feature subset. Different values for the number of features were tried to find the best feature subset giving the most optimal performance.

Parent selection strategy

The binary tournament selection strategy is used for this purpose. In this method, two individuals are first randomly selected from the population and the fitter of the two is selected as the first

parent. The same procedure is repeated for selecting the second parent, taking care that both parents selected are distinct individuals in the population.

Crossover operators

Two crossover operators are used. At every iteration, heuristic crossover was used with a probability of 20% and whole arithmetic recombination otherwise.

- *Heuristic crossover*

  This operator (Wright, A, 1991) is a bit greedy. It takes the fitness of the parents into account and tries to generate the new individual in the vicinity of the better individual. Suppose $\overline{X}$ and $\overline{Y}$ represent the two floating point arrays of two individuals and $\overline{Y}$ is better than $\overline{X}$. Then the child is generated as follows:

  $$\overline{Child} = \overline{Y} + r(\overline{Y} - \overline{X})$$

  Here r is a uniform random variable between 0 and 1 and $\overline{Child}$ is the floating point array of the child. If the values of the floating point array of the child go out of bounds, they are set to the lower bound (0) or the upper bound (1), whichever they are violating.

- *Whole arithmetic recombination*

  This operator (Schwefel, 1981) creates the child by taking the weighted sum of the two parents for each gene. Suppose $\overline{X}$ and $\overline{Y}$ represent the two floating point arrays of two individuals. Then the child is generated as follows:

  $$\overline{Child} = r\overline{X} + (1 - r)\overline{Y}$$

  Here $\overline{Child}$ is the floating point array of the child. Commonly, r is a uniform random variable between 0 and 1. However with this approach there is no way to get beyond the maximum and minimum values present in the initial population. That's why we are taking r as a uniform random variable between -1 and 2. By doing so we are no longer restricted by

the convex hull. If the values of the floating point array of the child go out of bounds, they are set to the lower bound (0) or the upper bound (1), whichever they are violating.

Mutation operators

A mutation operation is applied to every child generated by the crossover operator. A random number between 0 and length of the floating point array is selected. The floating point number at this position undergoes mutation. An adaptive non-uniform mutation (Michalewicz, 1994) was used. This mutation operator is adaptive in the sense that, it is uniform in the first iteration and the change is zero in the last iteration. Thus as the number of iterations increase, it shrinks the range from which values are drawn.

Suppose x_i is the floating point number that we wish to mutate. It is then replaced by x_{new} which is calculated as follows:

$$x_{new} = x_i + (u_i - x_i) * r(1 - t/T)$$

Where r is a uniform random variable between 0 and 1, t is the current iteration, T is the maximum number of iterations, and u_i is the upper limit of the range of values that x_i can take. In our case, it is equal to 1.

Replacement strategy

A steady state genetic algorithm was used. In order to maintain diversity, similarity based replacement was used. However the most similar individual to the individual being newly introduced could be the fittest individual. Thus, crowding based heuristic was used and the most similar individual from the worst one-fourth of the population was replaced (De Jong, 1975).

Fitness evaluation function

We are using a genetic algorithm as a wrapper approach for feature selection. In the wrapper approach, the usefulness of a feature subset is evaluated with the help of a machine learning

scheme that will be used for learning. Each individual in the population is a feature subset. Thus the fitness of the individual is calculated by evaluating the machine learning scheme using only those features. To evaluate a machine learning scheme, five tenfold cross-validation runs with different folds are performed and the correlation coefficient (defined in Section 5.1) is averaged over all runs. Thus, the higher the correlation coefficient, the fitter the individual is.

Initialization

As each individual is an array of floating point numbers between 0 and 1, the initial population of 500 individuals is created by randomly generating floating point numbers in the range 0 to 1.

Stopping Criterion

The genetic algorithm is allowed to run for 25000 iterations.

4.2 RELIEF-F FEATURE SELECTION

An implementation of the Relief-F feature selection is available as part of the WEKA package (Witten I.H., Eibe F., 2000) provided by the department of Computer Science at University of Waikato, New Zealand. The WEKA package is open source software issued under the GNU General Public License and is written in JAVA. Both equal influence nearest neighbors and weighted nearest neighbors were tried. The number of nearest neighbors had to be specified as a user defined parameter. Experiments showed that the best results were obtained using a value of ten for this parameter. A ranking of all the attributes in the dataset was obtained. The number of features to be used from this ranking was also to be decided. Different values for the number of features for both the above mentioned options were tried.

4.3 CORRELATION BASED FEATURE SELECTION

An implementation of the Correlation based feature selection is available as part of the WEKA package. The search method for searching the feature subset had to be specified as a user defined

parameter. Experiments were done using forward selection and best first search as the search methods and results were obtained. As opposed to RELIEF-F feature selection that produces a ranking of all the attributes, this method produces the best feature subset according to its internal criteria.

4.4 PRINCIPAL COMPONENT ANALYSIS

An implementation of Principal Component Analysis is available as part of the WEKA package. As opposed to other feature selection methods that produce a ranking of the attributes or a feature subset, this method produces new attributes that are linear combinations of the existing attributes. Thus, this method transforms the original feature space into a new reduced feature space. This reduced feature space was then used by the different machine learning schemes and results were obtained.

4.5 DECISION TREE

The See5 program (Quinlan, 1993) was used as the decision tree learning software. In biomaterial modeling, we are trying to predict the amount of protein adsorption, which is a continuous value. Thus it was necessary to convert the target attribute of the training examples into a discrete value. Hence the range of the target attribute was divided into six bins. We tried two methods for setting the range of these bins.

- Equal-interval-binning

  In this method, all the bins had equal range. This method often distributes the instances unevenly.

- Equal-frequency-binning

  In this method, all the bins have almost equal number of instances in them. However the range for each of the bin would be different.

After discretising the target attribute, the program was run to get a decision tree. The winnowing option was used which discards the irrelevant attributes before constructing the decision tree. After the decision tree is created, global pruning was applied that looks at the tree as a whole and prunes relatively weaker sub trees. The attributes occurring in the globally pruned decision tree were then used as the result of this feature selection method.

4.6 MACHINE LEARNING SCHEMES

The machine learning algorithms used in this thesis are: Regression trees, model trees, radial basis function networks, neural networks with backpropagation and locally weighted regression. The implementations for these machine learning schemes are available in the WEKA package. All the machine learning schemes available in WEKA are written in JAVA. The model trees were used with the smoothing option. The number of hidden nodes for the neural networks was found on the basis of the number of inputs applied to it. The linear weighing function was used in locally weighted regression. In RBF networks, the number of kernel functions was decided using clustering and a kernel function was added at the center of each cluster.

CHAPTER 5

EVALUATION AND RESULTS

5.1 EVALUATION OF A MACHINE LEARNING SCHEME

Performance evaluation of a machine learning scheme is a very important issue for comparing the performance of different machine learning schemes as well as in the evaluation of an individual of the population in the genetic algorithm based approach of feature selection. A common way of evaluating a machine learning scheme is to predict the error rate of the learning technique given a fixed sample of data. If the performance is tested on the same dataset that was used for training, an over optimistic performance measure will be obtained. Hence the performance of the learning scheme should be tested on a dataset that was not used in the training process. Since we had limited data, testing was done using tenfold cross validation. The data is divided randomly into approximately ten equal parts. Each part is held out in turn and the training is done using the remaining nine-tenths, then the error rate is calculated on the holdout set. The ten error estimates are averaged to yield an overall error estimate. A single tenfold cross-validation might not be enough to get a reliable error estimate. Different tenfold cross-validation experiments with the same learning scheme and dataset often produce different results, because of the effect of random variation in choosing the folds themselves. In this thesis, the tenfold cross-validation procedure is repeated five times and the results are averaged. In order to compare the performance of the different machine learning schemes, it is necessary that the

results be obtained on the same partitions of the datasets. Thus the same folds in all the cross-validations are used by all the learning schemes and for all the feature subsets.

Different measures of the error estimate like mean absolute error; mean squared error etc. could be used. We have selected to use the correlation coefficient. It is defined as a measure of the degree to which the actual and the predicted values of the target attribute tend to move together. The coefficient has a value between plus and minus 1, which indicates the strength and direction of association. A value of 0 means that there is no correlation between the actual and the predicted values, a value of +1 means that the results are highly correlated and a value of -1 means that the results are negatively correlated. The correlation coefficient is calculated as follows (Witten I.H., Eibe F., 2000):

$$CorrelationCoefficient = \frac{S_{PA}}{S_P S_A}$$

Where

$$S_{PA} = \frac{\sum_i (p_i - p_{mean})(a_i - a_{mean})}{n-1} \quad , S_P = \frac{\sum_i (p_i - p_{mean})^2}{n-1} \text{ and } S_A = \frac{\sum_i (a_i - a_{mean})^2}{n-1}$$

Here $p_1, p_2, \ldots p_n$ are the predicted values and $a_1, a_2, \ldots a_n$ are the actual values of the n instances in the test dataset.

The correlation coefficient measure was chosen in particular, because we would be using the machine learning models to identify the biomaterials with high protein adsorption, which would be experimented on and further tested *in vitro*. Thus we are more interested in the way the predicted and actual values associate with each other rather than the mean absolute difference between them.

5.2 FEATURE SELECTION RESULTS

From the entire dataset, machine learning approaches were used to predict the amount of fibrinogen adsorption and NMA-NFF adsorption. The fibrinogen protein was selected because fibrinogen adsorption plays a very important role in the performance of biomaterials and the NMA-NFF protein was selected because the amount of training data available for this protein was larger than other proteins. Both filter based and wrapper based approaches for feature selection were tried. For the filter approach, the feature subset obtained by the feature selection method was used by all the machine learning schemes and the results thus obtained were compared. In the wrapper approach, the machine learning scheme was an integral part of the feature selection process itself. To obtain a fair comparison of the results, the same folds were used by all the machine learning schemes.

We begin by comparing the feature selection methods that produce a ranking of the features to the genetic algorithm based approach. RELIEF-F and decision trees are such feature selection methods. RELIEF-F was tried with both the options: equal influence nearest neighbors and weighted nearest neighbors and the number of nearest neighbors used was set to 10. With decision trees, it was consistently found that equal interval binning gave better results than equal frequency binning and hence the attributes obtained by the equal interval binning were tried with different machine learning schemes. The genetic algorithm based approach was repeated three times and the average correlation coefficient of the three runs was used in the comparison. Different numbers of features were tried for each of the above mentioned feature selection methods and for each machine learning scheme and the results are depicted below. The X axis denotes the number of features used and the Y axis denotes the correlation coefficient obtained by the machine learning scheme involved.

A) Results with the Fibrinogen dataset.

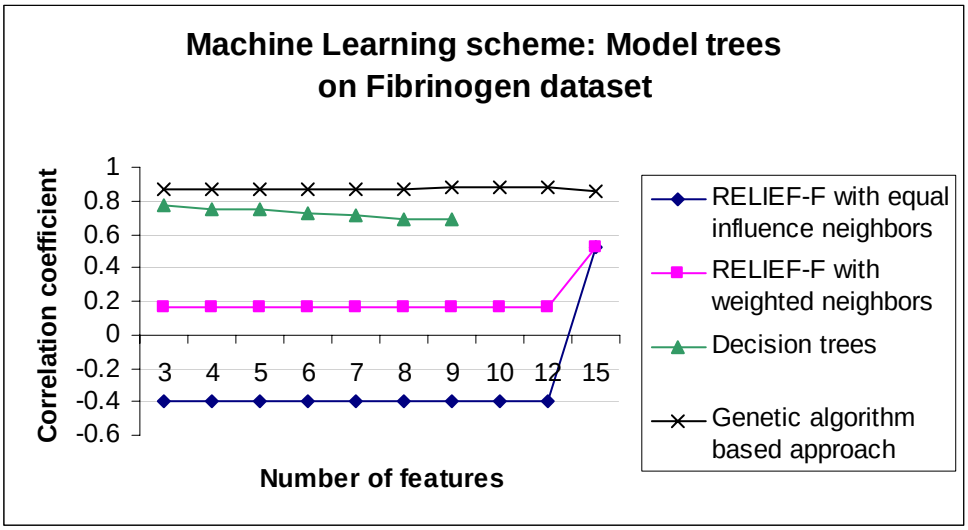

Figure 8: Performance comparison of feature selection methods with the model trees on the Fibrinogen dataset.

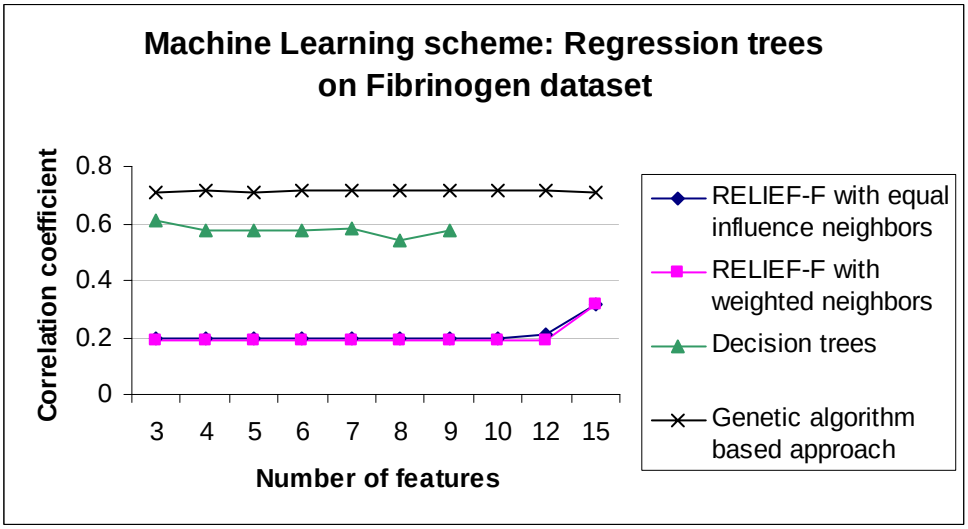

Figure 9: Performance comparison of feature selection methods with regression trees on the Fibrinogen dataset

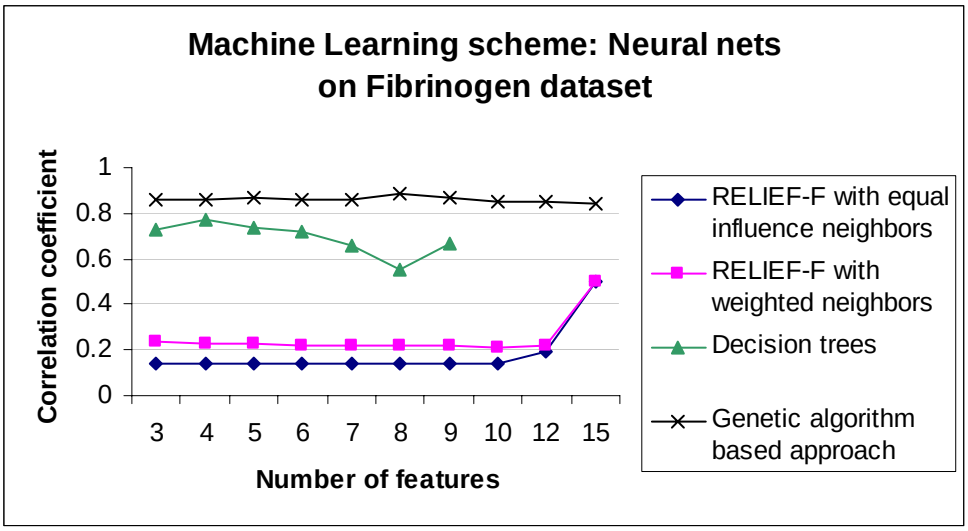

Figure 10: Performance comparison of feature selection methods with neural networks with backpropagation on the Fibrinogen dataset

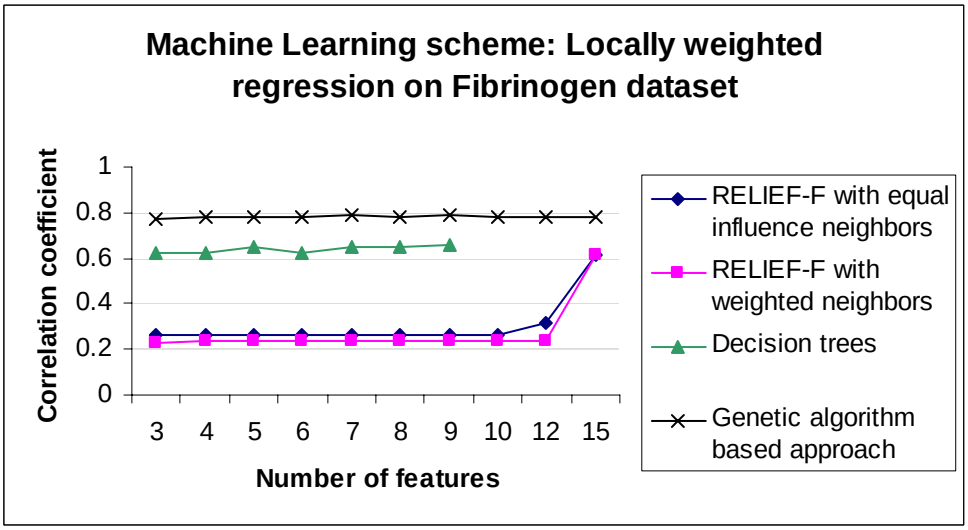

Figure 11: Performance comparison of feature selection methods with locally weighted regression (LWR) on the Fibrinogen dataset

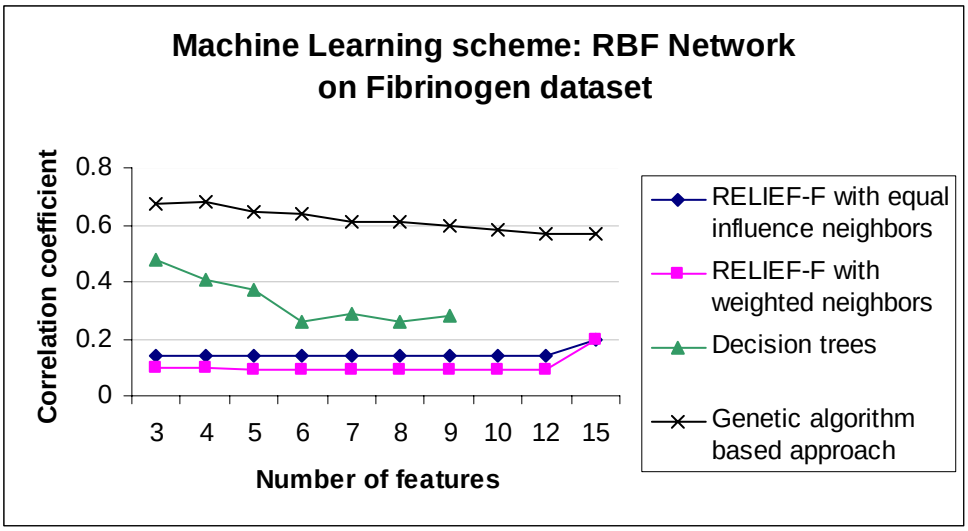

Figure 12: Performance comparison of feature selection methods with RBF networks on the Fibrinogen dataset

B) Results with the NMA-NFF dataset.

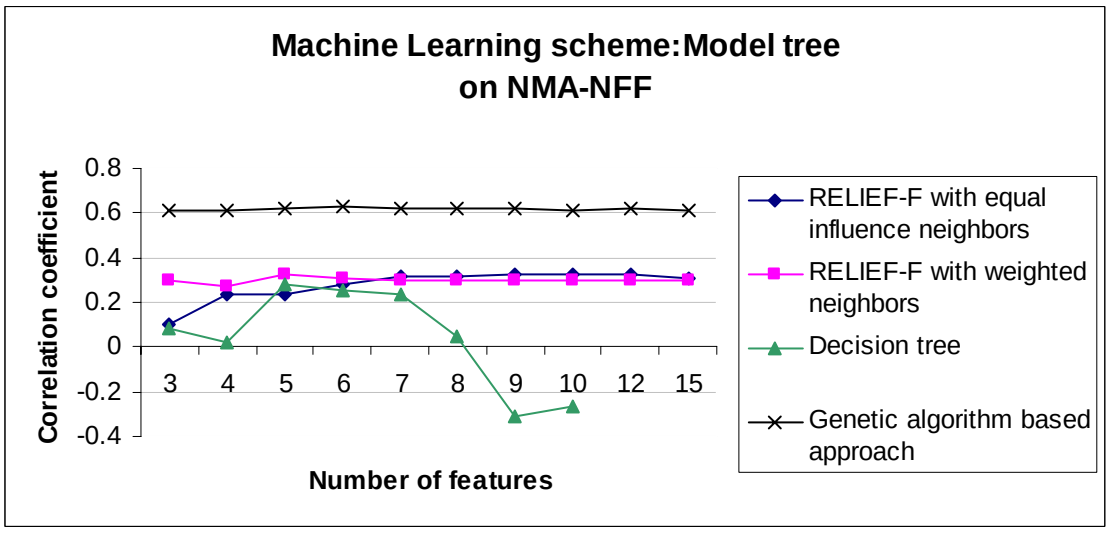

Figure 13: Performance comparison of feature selection methods with model trees on the NMA-NFF dataset

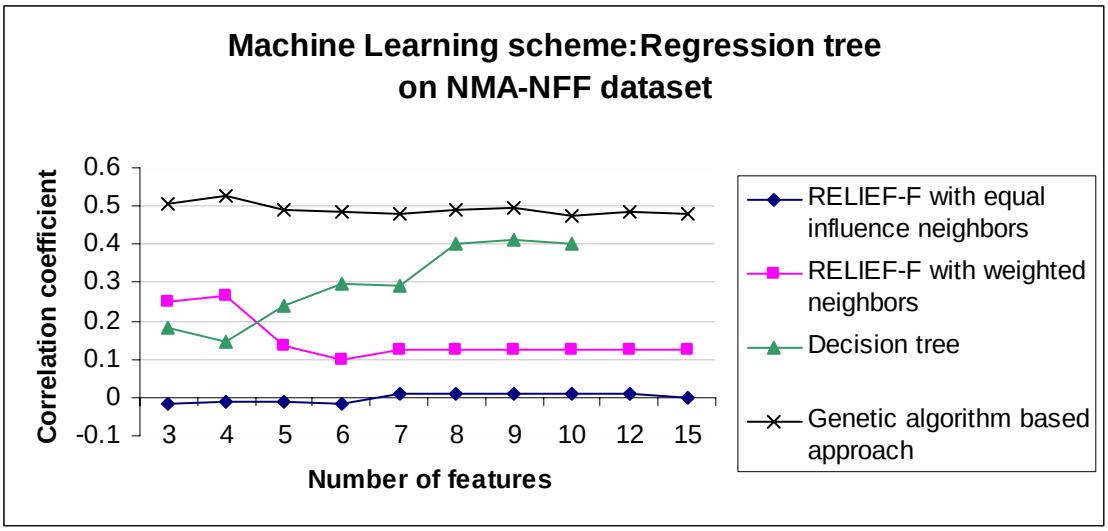

Figure 14: Performance comparison of feature selection methods with regression trees on the NMA-NFF dataset

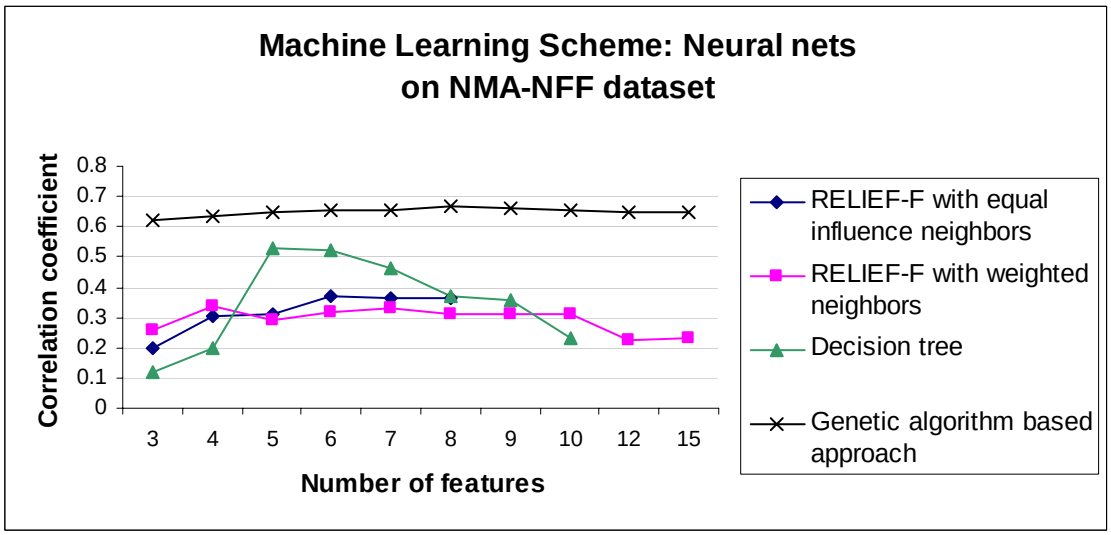

Figure 15: Performance comparison of feature selection methods with neural networks with backpropagation on the NMA-NFF dataset

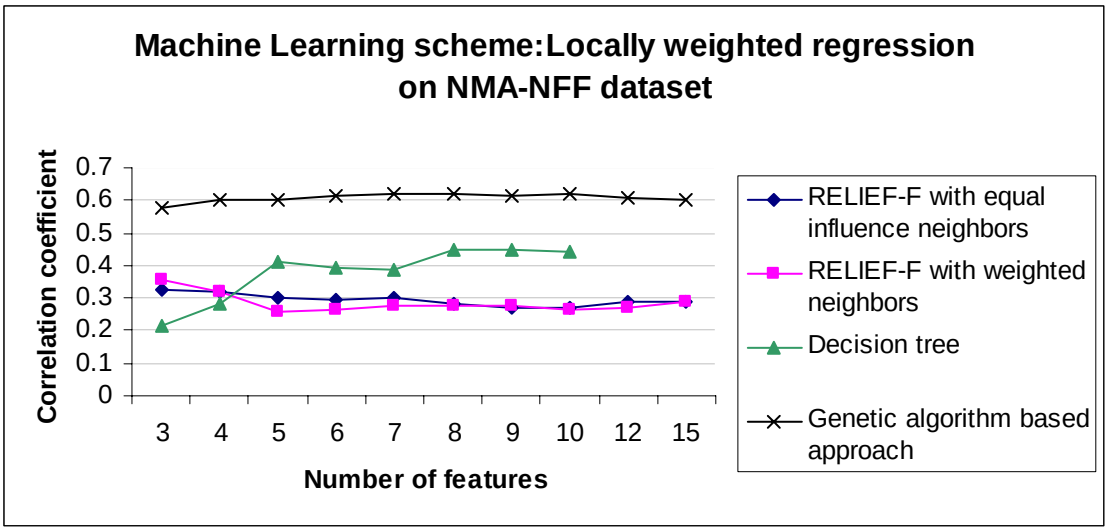

Figure 16: Performance comparison of feature selection methods with locally weighted regression (LWR) on the NMA-NFF dataset

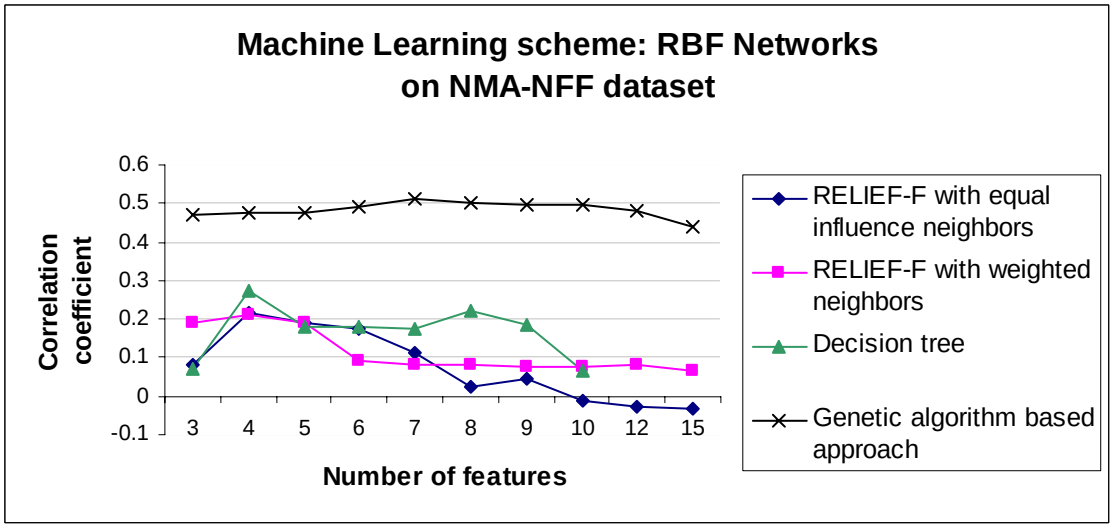

Figure 17: Performance comparison of feature selection methods with RBF networks on the NMA-NFF dataset

Thus, we can see that amongst RELIEF-F, decision trees and genetic algorithm based approaches, with all the machine learning schemes, with both the datasets, the genetic algorithm based approach gives the best results.

We will next be comparing the feature selection methods that produce a feature subset (as opposed to ranking of the attributes) with the genetic algorithm based approach. Correlation

based feature selection and Principal Component Analysis are such feature selection methods. Both best first search and forward selection were tried with correlation based feature selection. For a given machine learning scheme, the best result obtained by the genetic algorithm was used for the comparison. The results obtained are shown in Figures 18 and 19. The X axis denotes the various machine learning schemes used and the Y axis denotes the correlation coefficient obtained by the machine learning scheme involved. It should be noted that the results with CFS with both the best first search and forward search are very similar in the two figures.

A) Results with the Fibrinogen dataset.

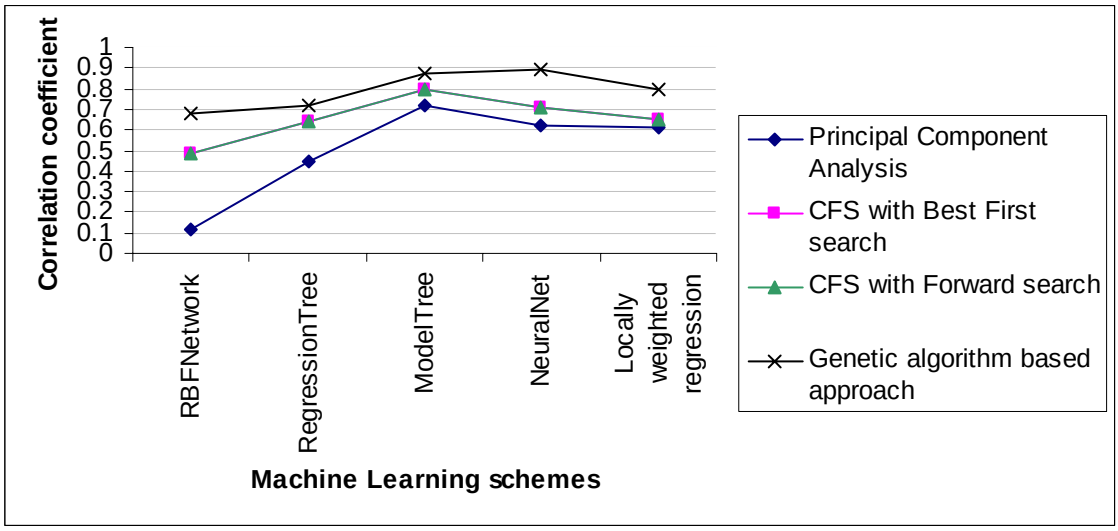

Figure 18: Performance of feature selection methods that give a feature subset on Fibrinogen dataset.

B) Results with the NMA-NFF dataset.

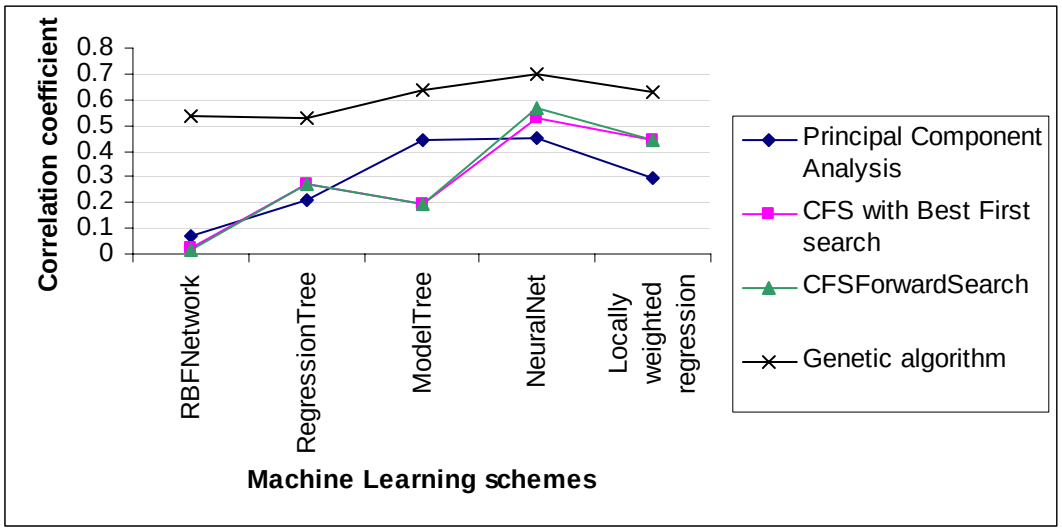

Figure 19: Performance of feature selection methods with that give a feature subset on NMA-NFF dataset.

Thus, we can see that amongst Principal Component Analysis, correlation based feature selection and the genetic algorithm approach, for all the machine learning schemes, with both the datasets, the genetic algorithm approach gives the best results.

From the above two comparisons, it can be deduced that the wrapper based approach with genetic algorithm was the most appropriate feature selection method in this problem domain. For each machine learning scheme, the best feature subset found by the genetic algorithm did not necessarily have the same number of features. For example, for the Fibrinogen dataset, the best feature subset for model trees had nine attributes and for regression trees had six attributes. For all the machine learning schemes, for different numbers of features, the average of the correlation coefficient from all the three genetic algorithm runs was calculated. For each machine learning scheme, the largest of these values and the corresponding number of features were noted. Shown below are the best feature subsets obtained by the genetic algorithm in the different runs for each machine learning scheme for that optimal number of features. It should

also be noted that the best results for a given machine learning scheme were obtained in one of the three runs for the same optimal number of features. The feature subsets are represented by the indices of the attributes, the entire set of which is listed in the Appendix.

A) Results with the Fibrinogen dataset.

Table 1: Best feature subset with 9 attributes for model trees in the three GA runs on Fibrinogen dataset.

Run1 *CorrelationCoefficient=0.8635*	*Run2* *CorrelationCoefficient=0.8714*	*Run3* *CorrelationCoefficient=0.8610*
1,15,20,48,74,77,78,79,86	1,13,15,48,63,78,82,83,86	15,21,48,50,53,63,83,90,91

Table 2: Best feature subset with 6 attributes for regression trees in the three GA runs on Fibrinogen dataset

Run1 *CorrelationCoefficient=0.7157*	*Run2* *CorrelationCoefficient=0.7143*	*Run3* *CorrelationCoefficient=0.7145*
52,63,77,80,91,108	52,59,63,77,91,108	19,52,63,77,78,91

Table 3: Best feature subset with 8 attributes for neural networks in the three GA runs on Fibrinogen dataset

Run1 *CorrelationCoefficient=0.8818*	*Run2* *CorrelationCoefficient=0.8975*	*Run3* *CorrelationCoefficient=0.8712*
8,14,48,55,58,72,96,100	14,18,27,72,94,96,98,106	4,14,48,55,72,82,94,106

Table 4: Best feature subset with 7 attributes for locally weighted regression (LWR) in the three GA runs on Fibrinogen dataset.

Run1 *CorrelationCoefficient=0.7848*	*Run2* *CorrelationCoefficient=0.7957*	*Run3* *CorrelationCoefficient=0.7907*
0,48,51,61,66,80,85	0,51,66,85,90,100,101	0,31,48,51,56,66,85

Table 5: Best feature subset with 4 attributes for RBF networks in the three GA runs on Fibrinogen dataset.

Run1 *CorrelationCoefficient=0.6812*	*Run2* *CorrelationCoefficient=0.6812*	*Run3* *CorrelationCoefficient=0.6798*
14,93,94,101	15,93,94,101	15,44,49,93

B) Results with the NMA-NFF dataset.

Table 6: Best feature subset with 5 attributes for model trees in the three GA runs on NMA-NFF dataset.

Run1	*Run2*	*Run3*
CorrelationCoefficient=0.6399	*CorrelationCoefficient=0.6384*	*CorrelationCoefficient=0.6191*
15,17,50,94,104	8,17,94,102,104	53,71,94,102,104

Table 7: Best feature subset with 4 attributes for regression trees in the three GA runs on NMA-NFF dataset

Run1	*Run2*	*Run3*
CorrelationCoefficient=0.5312	*CorrelationCoefficient=0.5197*	*CorrelationCoefficient=0.4954*
8,13,17,85	6,20,24,85	11,20,17,85

Table 8: Best feature subset with 8 attributes for neural networks in the three GA runs on NMA-NFF dataset

Run1	*Run2*	*Run3*
CorrelationCoefficient=0.6979	*CorrelationCoefficient=0.6563*	*CorrelationCoefficient=0.6451*
5,15,18,27,33,48,87,104	5,22,32,33,65,87,93,94,104	5,19,33,62,68,87,88,104

Table 9: Best feature subset with 9 attributes for locally weighted regression (LWR) in the three GA runs on NMA-NFF dataset.

Run1	*Run2*	*Run3*
CorrelationCoefficient=0.6121	*CorrelationCoefficient=0.6295*	*CorrelationCoefficient=0.6074*
18,19,30,48,58,78,99,101,104	15,18,19,49,78,92,99,101,104	1,7,8,18,19,61,78,99,101

Table 10: Best feature subset with 8 attributes for RBF networks in the three GA runs on NMA-NFF dataset.

Run1	*Run2*	*Run3*
CorrelationCoefficient=0.5366	*CorrelationCoefficient=0.4829*	*CorrelationCoefficient=0.4801*
20,29,45,60,96,101,103,107	20,68,82,95,97,101,103,107	11,17,38,60,95,101,103,107

It can be seen from the above results that the optimal number and the best feature subset found by the genetic algorithm were heavily dependent on the underlying machine learning

scheme. It was also found that the best individual found by the genetic algorithm in all the three runs had some common repeating features.

5.3 MACHINE LEARNING SCHEME RESULTS

In order to compare the performance of different machine learning schemes, feature selection using the genetic algorithm based approach is done and the best feature subset found for a given machine learning scheme is used. The performance of the machine learning scheme is evaluated using this feature subset by performing ten-fold cross validation five times. To make a fair comparison, the same folds are used for all the machine learning schemes. The performance comparison of all the machine learning schemes is shown in Figures 20 and 21.

A) Results with the Fibrinogen dataset.

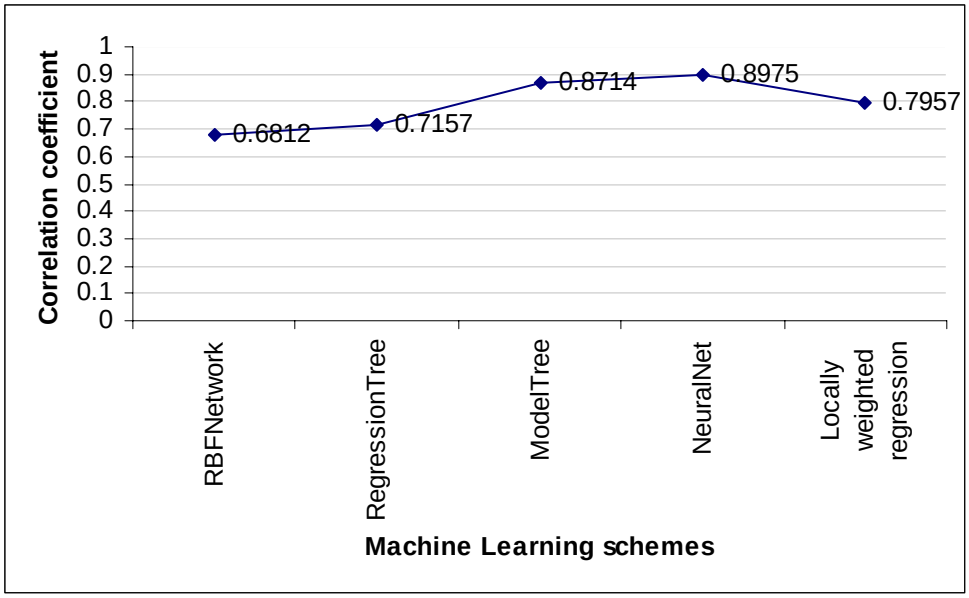

Figure 20: Performance comparison of different machine learning schemes on Fibrinogen dataset

B) Results with the NMA-NFF dataset.

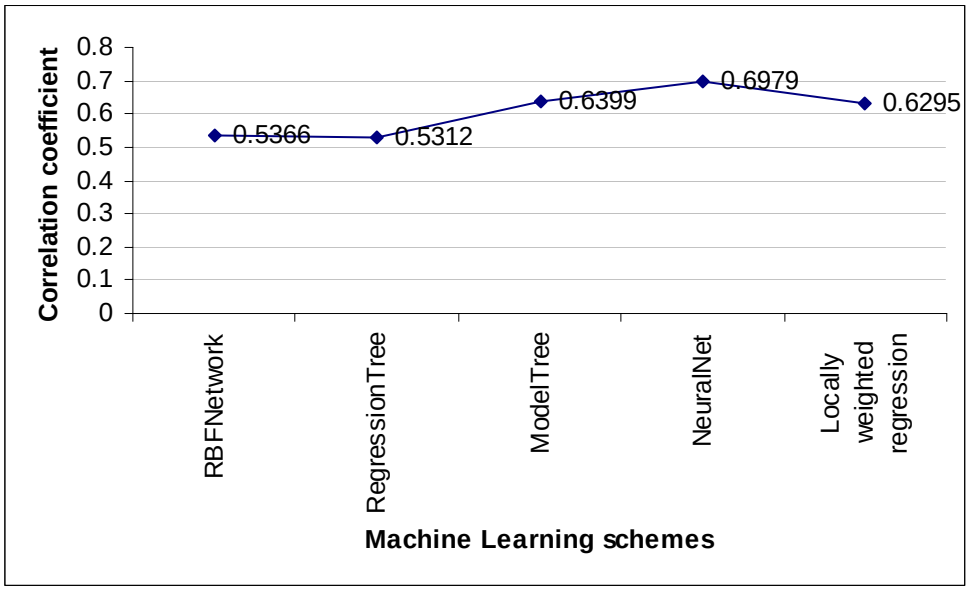

Figure 21: Performance comparison of different machine learning schemes on NMA-NFF dataset

Thus we can see that among all the five machine learning schemes compared, on both the datasets, the backpropagation neural networks give the highest correlation coefficient and hence the best results. Model trees also give good results, but not as good as backpropagation neural networks. The next best results obtained by locally weighted regression. We also notice that the difference in performance of all the machine learning methods is moderate compared to the difference in the performance of all the feature selection methods.

5.4 UNCERTAINTY MEASUREMENT RESULTS

The dataset used by the machine learning schemes consisted of several polymers. Each polymer was described with the help of 109 descriptors and the average and standard deviation values for amount of protein adsorption obtained by several experiments done *in vitro*. Since the target attribute value for each example was obtained experimentally, there is much uncertainty in these values. Thus, all the machine learning results should be robust to this uncertainty in the data. Usually in the medical and biological domains, people tend to ignore the uncertainty in the data.

Smith et al. (Smith, 2004) have done an effort to incorporate the uncertainty handling into the selection of features with decision trees for neural networks. In this thesis we are not dealing with the uncertainty and are only attempting to quantify the effect of this uncertainty in the data on the machine learning results. In order to accomplish this, the following approach was used: Several datasets were generated by varying the value of the target attribute measured for each polymer in a random fashion, within a normal distribution defined by the experimental mean and standard deviation. The machine learning schemes were tried with all these datasets to find the sensitivity of each machine learning scheme to the uncertainty in the measurements of the data.

All the machine learning schemes were tried with the thirty datasets created. The optimal number and the actual features to be used for each of the machine learning schemes was found by using the best feature subset found by the genetic algorithm for it respectively. The results obtained for all the thirty experiments for all the machine learning schemes are shown in Figures 22 and 23. The X axis denotes the different machine learning schemes and the Y axis denotes the average correlation coefficient and its standard deviation obtained from the thirty experiments.

A) Results with the Fibrinogen dataset.

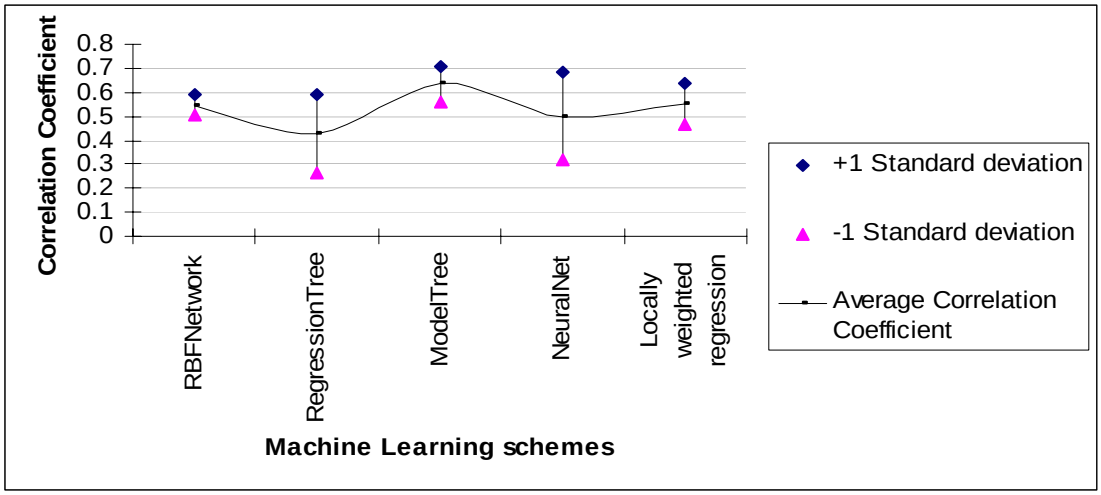

Figure 22: Comparison of the different machine learning schemes in terms of uncertainty handling on Fibrinogen dataset.

B) Results with the NMA-NFF dataset.

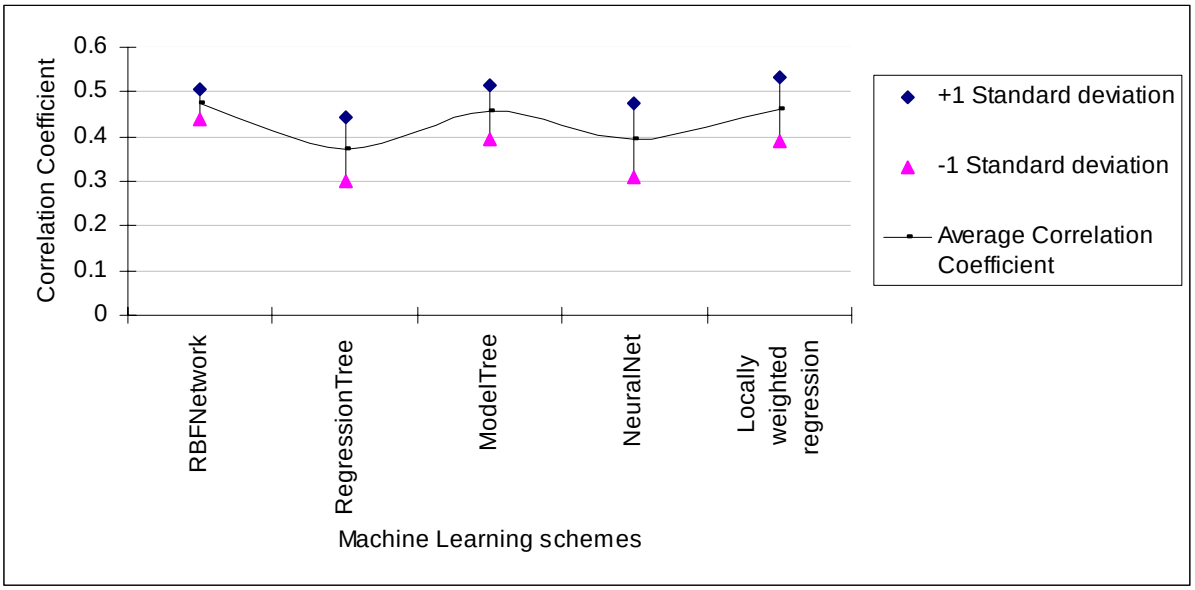

Figure 23: Performance comparison of the different machine learning schemes in terms of uncertainty handling.

We can see that the average correlation coefficient obtained from the thirty runs for both the datasets and for each machine learning method is much less than that obtained with the original dataset using average values for the target attribute. The same feature subset with which very good results were obtained with the original dataset with the average values was used for all the thirty experiments and the performance has deteriorated. This suggests that the feature subset used for these thirty experiments for all the machine learning schemes is also very sensitive to the uncertainty in the data. Thus uncertainty handling is a very important issue in this problem domain and needs to be incorporated in the initial feature selection phases as well.

This also suggests that the standard deviation that was used to create these thirty datasets is very high. We therefore recommend conducting more experiments *in vitro* to reduce the standard deviation.

CHAPTER 6

CONCLUSIONS

In this thesis, we explore the application of machine learning techniques to predict the amount of protein adsorption on biomaterial surfaces. This would be a very valuable tool for biomaterial modeling. Since the number of attributes is very large, feature selection was very important for this problem domain. Different filter based and wrapper based approaches for feature selection were compared. The filter based approaches used were RELIEF-F, correlation based feature selection, principal component analysis and decision trees. The wrapper based approach was developed by writing a genetic algorithm in JAVA. The machine learning schemes explored in this thesis were model trees, regression trees, neural networks with backpropagation, locally weighted regression, and radial basis function networks. The predictive accuracy and the uncertainty handling capabilities of these machine learning schemes was compared.

Among all the feature selection methods tested in this thesis, the wrapper based approach is found to be the best method for this problem domain. The results obtained by the wrapper approach are consistently better than all other feature selection methods explored in this thesis. The number and the actual features selected by this approach are very much dependent on the underlying machine learning scheme. Every example in the original dataset for both Fibrinogen and NMA-NFF has its target attribute value as an average of several experiments done *in vitro*. With this original dataset, neural networks with backpropagation give the best results. The model trees and locally weighted regression also give good results, though not as good as neural

networks with backpropagation. The difference in performance of the machine learning schemes is not much compared to the difference in performance of the feature selection methods. With the right feature subset, most of the machine learning schemes give quite good results. Thus it appears that using the right feature selection method is a more important issue than using the right machine learning scheme in this problem domain.

We have explored the effect of uncertainty only on the best feature subset found for a given machine learning scheme. From the uncertainty measurement results, we see that it should be incorporated in the feature selection process as well. The best feature subset obtained for the original dataset with average values for the target attribute does not necessarily give every good results on the thirty datasets that were randomly created for uncertainty measurement. Thus this work could be extended by exploring the effect of uncertainty during the feature selection process as well. The features occurring frequently could be included in the optimal feature subset for a given machine learning scheme. In this thesis we are only attempting to quantify the effect of uncertainty in the data on the performance of machine learning schemes. This work could also be extended by finding ways to actually deal with the uncertainties in the data and come up with models that are robust to this uncertainty.

The number of training instances in these domains is very small which clearly reduces the accuracy of the models. Therefore we need to explore methods for improving the performance despite the scarcity of training instances. One way of doing this could be using both labeled and unlabelled instances for building the models. The labeled instances could be used by a machine learning scheme to come up with a model, which in turn would be used to probabilistically predict values for the unlabelled instances. Those newly labeled instances could then further take part in the process of building the model. This iterative process could continue until there is no

significant change in the labels of the originally unlabelled instances. This approach can be viewed as a variation of the Expectation Maximization (EM) algorithm (Mitchell,1997).

REFERENCES

[1] Abramson, S., Alexe, G., Hammer, P., Knight, D., Kohn, J. (2002), "*Using logical analysis of data to find physio-mechanical data patterns which predict cellular outcomes*".

[2] Atkeson, C., Moore, A., and Schaal, S. (1996), "*Locally weighted learning*", AI Review, Volume 11, pp. 11-73, Kluwer publications.

[3] Boros, E., Hammer, P., Ibaraki, T., Kogan, A. (1997), "*A logical analysis of numerical data*" Math program 79, pp. 163-190.

[4] Breiman, L., Friedman, J.H., Olshen, R.A. and Stone C.J (1984), "*Classification and regression trees*", Wadsworth publications.

[5] Broomhead, D.S., Lowe, D. (1988), "*Multivariable functional interpolation and adaptive networks*", Complex Systems, Volume 2, pp. 321-355.

[6] Chemical Computing Group Inc., *MOE (The Molecular Operating Environment), v. 2003.02*, Montreal, Canada H3A 2R7: 2003.

[7] Crama, Y., Hammer, P., Ibaraki, T. (1988), "*Cause-effect relationships and partially defined Boolean functions*", Ann Oper Res, Volume 16, pp. 299-326.

[8] Dragon (Web Version) http://www.disat.unimib.it/chm/dragon.htm R. Todeschini, V. Consonni, A. Mauri, M. Pavan v. 3.0, Milano, Italy: 2003.

[9] Fiordeliso, J., Bron, S., Kohn, J. (1995), "*Biomaterials in Solution, as Interfaces and as Solids*" VSP Publisher: Utrecht, The Netherlands, pp. 695-708.

[10] Frank, E., Wang, Y., Inglis, S., Homes, G., Witten, I. (1998), "*Using model trees for classification*", Machine Learning, Volume 32 , Issue 1, pp. 63 - 76.

[11] Goldberg, D. E. (1989), "*Genetic Algorithms in Search, Optimization, and Machine Learning*". Addison-Wesley, Reading, MA.

[12] Hall, M.A.(1999), "*Correlation-based Feature Selection for Machine Learning*", Ph.D. thesis, Department of Computer Science, Waikato University, New Zealand.

[13] Hench, L. and J. Polak (2002). "Third-Generation Biomedical Materials." *Science* 295: 1014-1017.

[14] Jong, D. (1975), "*An analysis of the behavior of a class of genetic adaptive systems*", PhD Thesis, University of Michigan.

[15] Kira K. ,Rendell L.A (1992), "*A practical approach to feature selection*" In D. Sleeman and P. Edwards editors, Proceedings of the International Conference on Machine Learning, pp. 249-256, Morgan Kaufman.

[16] Kononenko, I. (1994), "*Estimating attributes: analysis and extensions of Relief*.", In De Raedt, L. and Bergadano, F., Machine Learning: ECML-94, pp. 171-182. Springer Verlag.

[17] Magnani, A., Peluso, G., Maragarucci, S., Chittur, K. K. (2002), "*Integrated Biomaterials Science*", Kluwer Academic/Plenum: New York, pp. 669-689.

[18] Michalewicz, Z., Logan, T. D., and Swaminathan, S. (1994), "*Evolutionary Operators for continuous convex Parameter Spaces*" In Proceeding of 3rd Annual Conference on Evolutionary Programming eds. A.V. Sebald and L.G. Fogel, River Edge, NJ, World Scientific Publishing, pp. 98-108.

[19] Mitchell, T. (1997), "*Machine Learning*", McGraw Hill.

[20] Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T. (1988), "*Numerical Recipes in C*" Cambridge University Press, Cambridge.

[21] Quinlan, J. R. (1986), "*Induction of decision trees*". Machine Learning, Volume 1, pp. 81-106.

[22] Quinlan J. R. (1992), "*Learning with continuous classes*", Proceeding Australian Joint Conference on Artificial Intelligence, World Scientific, Singapore, pp. 343-348

[23] Quinlan, J. R. (1993), "*C4.5: Programs for machine learning*", Morgan Kaufmann.

[24] Rich, E., Knight, K. (1991), "*Artificial Intelligence.*" McGraw-Hill.

[25] Rumelhart D. E., Hinton, G. E., & Williams, R. J. (1986), "*Learning internal representations by error propagation*" in "Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1, pp. 318–362, Bradford Books/MIT Press" Cambridge, MA.

[26] Schwefel, H. P.(1981) , "*Numerical Optimization for Computer Models*", Chichester, John Wiley & Sons, UK.

[27] Siedlecki, W., Sklansky, J. (1989), "*A note on genetic algorithms for large-scale feature selection*" Pattern Recognition Letters, Volume 10, pp. 335–347.

[28] Sikonja, M. R., Kononenko, I. (1997) "*An adaptation of Relief for attribute estimation on regression.*" In D.Fisher (ed.): Machine Learning, Proceedings of 14th International Conference on Machine Learning ICML'97*,* Nashville, TN.

[29] Simon, H.(1983), "*Why Should Machines Learn?*", Machine Learning: An Artificial Intelligence Approach, Vol. 1, Michalski, R.S., Carbonell, J.G., Mitchell, T.M., (Eds.), Tioga, Palo Alto/CA.

[30] Smith, J. R., Knight, D., Kohn, J., Rasheed, K., Weber, N., Kholodovych, V. ,Welsh, W. J. (2004) ," *Using surrogate modeling in the prediction of fibrinogen adsorption onto polymer surfaces*".

[31] Smith M. (1993), "*Neural Networks for Statistical Modeling*" Van Nostrand Reinhold, New York.

[32] Witten, I. H., Eibe, F. (2000), "Data *Mining: Practical machine learning tools with Java implementations*", Morgan Kaufmann, San Francisco.

[33] Wright, A. H. (1991), "*Genetic Algorithms For Real Parameter Optimization*", Foundations of Genetic Algorithms, Morgan Kaufman, pp. 205-218.

APPENDIX

The dataset used in this thesis was obtained from "New Jersey Centre for Biomaterials" at Rutgers University (www.njbiomaterials.org). The Fibrinogen dataset consisted of 45 instances described by 109 attributes and the NMA-NFF dataset consisted of 93 instances described by 109 attributes. Given below is the complete list of these 109 attributes:

Table 11: List of all the attributes.

0) Pendent	55) PEOE_VSA_FPPOS
1) Diacid	56) PEOE_VSA_HYD
2) No of secondary C	57) PEOE_VSA_NEG
3) No of tertiary C	58) PEOE_VSA_PNEG
4) No of subaromatic C	59) PEOE_VSA_POL
5) No of primary alcohols	60) PEOE_VSA_POS
6) No of ethers	61) Q_VSA_POS
7) No of acceptor atoms	62) Kier1
8) Insaturation index	63) Kier2
9) Hydrophilic factor	64) Kier3
10) Aromatic ratio	65) KierA1
11) Ghose-Crippen molar refractivity	66) KierA2
12) Fragment based polar surface area	67) KierA3
13) MoriguchiOctanol-waterPartitionCoefficientLogP	68) KierFlex
14) Tg	69) apol
15) AWCA	70) bpol
16) FI	71) mr
17) Diameter	72) a_acc
18) Petitjean	73) a_hyd
19) PetitjeanSC	74) vsa_acc
20) Radius	75) vsa_hyd
21) weinerPol	76) Slogp
22) a_count	77) SlogP_VSA2
23) a_IC	78) SlogP_VSA4
24) a_ICM	79) SlogP_VSA5
25) a_nH	80) SlogP_VSA9
26) b_ar	81) SMR
27) b_count	82) SMR_VSA0
28) b_rotN	83) SMR_VSA3
29) b_rotR	84) SMR_VSA5
30) b_single	85) SMR_VSA6
31) chi0v	86) SMR_VSA7

32) chi0v_C	87) TPSA
33) chi1v	88) density
34) chi1v_C	89) vdw_area
35) weight	90) vdw_vol
36) a_heavy	91) logP(o/w)
37) a_nC	92) noEthersInBackbone
38) b_heavy	93) noEthersInPendantChain
39) chi0	94) eth/TFI
40) chi0_C	95) RBN
41) chi1	96) BEHe8
42) chi1_C	97) ARR
43) Zagreb	98) MAVdWV
44) PEOE_PC+	99) VEv2
45) PEOE_PC-	100) BEHm8
46) PEOE_VSA+0	101) MATS3e
47) PEOE_VSA+1	102) nBM
48) PEOE_VSA+2	103) BELm2
49) PEOE_VSA-0	104) ATS8v
50) PEOE_VSA_FHYD	105) GATS8M
51) PEOE_VSA_FNEG	106) X5A
52) PEOE_VSA_FPNEG	107) TIC2
53) PEOE_VSA_FPOL	108) Mv
54) PEOE_VSA_FPOS	