BU-CE-0011
Instance-based Regression by Partitioning Feature Projections [*]

İlhan Uysal H.Altay Güvenir
Department of Computer Engineering,
Bilkent University,
06533 Ankara, Turkey
{uilhan,guvenir}@cs.bilkent.edu.tr

Abstract

This paper presents a new instance-based method, called Regression by Partitioning Feature Projections (RPFP) to fill the gap in the literature for lazy methods that achieves higher accuracies for regression problems. RPFP also presents some additional advantages and even better performance when compared to other lazy approaches such as k-Nearest Neighbor Regression and Locally Weighted Regression. RPFP makes predictions on each feature dimension separately, and combines these predictions to find a prediction for a given query instance. Even with its additive nature, RPFP can properly handle strong dependencies between features as shown by its comparison with the additive algorithm, Regression on Feature Projections (RFP). Besides those benefits, RPFP enjoys some other properties; e.g., it is robust to missing values, irrelevant features and the curse of dimensionality.

Keywords: Regression, Function Approximation, Feature Projections.

1 Introduction

Predicting values of numeric or continuous attributes is known as *regression* in statistics. Predicting real values is an important research field in machine learning. Even though, much research is concentrated on classification in machine learning literature, recently the focus of machine learning community has moved strongly towards regression, since a large number of real-life problems can be modeled as regression problems. Various names are used for this problem in the literature, such as functional prediction, real value prediction, function approximation and continuous class learning. We will prefer its historical name, *regression*, in the paper.

The term *eager* is used for learning systems that construct rigorous models of the domain during training. By constructing models, two types of knowledge, prediction and concept descriptions that enable interpretation can be addressed. By using induced models of many eager methods, interpretation of the underlying data can be done. On the other hand, *lazy* approaches do not construct models and delay processing to the prediction phase (Aha, 1997).

[*]This research is supported, in part, by TUBITAK (Scientific and Technical Research Council of Turkey) under Grant 198E015.

An important problem, common to many lazy learning algorithms, is that they are not suitable for interpretation by humans; since they do not produce compact descriptions especially when compared to algorithms that induce models such as trees or rules. Hence, the major task of these methods is prediction.

Nevertheless, *lazy* approaches are very popular in the literature, because of some important properties. They make predictions according to the local position of query instances. They can form complex decision boundaries in the instance space even when relatively little information is available, since they do not generalize the training data by constructing global models. Another advantage is that learning in lazy approaches is very simple and fast, since training generally involves just storing the instances. Finally they do not have to construct a new model, when a new instance is added to the data.

Besides these common characteristics of lazy approaches, however, the most significant problem with them is the one posed by irrelevant features (Domingos, 1997). Some feature selection and feature weighting algorithms have been developed in the literature for this purpose (Wettscherect, Aha & Mohri, 1997). A good property that must be enabled in a lazy algorithm is that some features may be important or relevant only in some regions of the instance space. This characteristic is known as *context-sensitivity* or *adaptivity* in the literature.

This paper describes a new lazy regression method based-on feature projections called Regression by Partitioning Feature Projections (RPFP). Feature projection based approaches store the training instances as their projected values on each feature dimension separately (Güvenir & Sirin, 1996; Güvenir, Demiroz & İlter, 1998; Uysal & Güvenir, 1999b). In predicting the target value of a query instance, each feature makes a separate prediction using only the value of the query instance for that feature, then all the feature predictions are combined to make the final prediction.

Feature projection based techniques have been applied to many classification problems successfully. The main advantage of feature projection based classification methods are their short classification time. They are also robust to irrelevant features and missing feature values. On the other hand, the main shortcoming of feature projection based methods is that they ignore the interactions or dependencies between features.

The RPFP method described in this paper is adaptive and robust to irrelevant features. Further, it can cope with the curse of dimensionality problem, therefore it is suitable for high-dimensional data, and can handle interactions between features. It does not require any normalization of feature values and handles successfully the cases having missing feature values. RPFP can be designed as a lazy, non-parametric, non-linear, and adaptive induction method based on feature projections.

The next section gives a short overview on related regression methods. In Section 3, RPFP is described. Important properties of the RPFP method are described in detail in Section 4. Section 5 gives the complexity analysis of RPFP. Section 6 compares RPFP with KNN and LWR. Results of empirical evaluations are presented in Section 7. Section 8 concludes the paper with some directions for the future work.

2 Lazy Methods for Regression

The literature does not provide as many lazy approaches for regression as the case for eager approaches. The two well known lazy approaches are K-Nearest Neighbor (KNN) regression, and Locally Weighted Regression (LWR). They are similar to each other. The most important similarity is that both of them use nearest neighbor instances for a given query location, and make the predictions by using these instances. The major difference is in the way they compute their predictions. KNN makes local approximations by using neighboring instances, on the other hand LWR constructs a local model using these instances, and makes approximation on the extracted linear or nonlinear local models. Many improvements and implementations of them are developed so far. They are described in the following sections briefly, since these methods are similar to RPFP in the literature of both statistics and machine learning.

2.1 Instance-Based Regression

Instance-based learning (IBL) algorithms are very popular since they are simple for computation during the training of instances (Aha, Kibler & Albert, 1991; Dasarathy, 1991). In most applications, training is done simply by storing the instances. This section describes the application of this technique for regression (Kibler, Aha & Albert, 1989).

In instance-based regression, each instance is usually represented as a set of attribute value pairs, where the values are either nominal or numeric, and the value to be predicted is continuous. The problem to be solved is, for a given query instance, to predict its target value as a function of similar other instances whose target values are known. The *nearest neighbor* is the most popular instance-based algorithm. The target values of the most similar neighbors are used in this task. Here the similarity is the negation of the Euclidean distance between instances. Formally, if we let real numbers R be a numeric domain, and $\mathbf{X}$ be an instance space with p attributes, then we can describe the approximation function, F, for predicting the target value y_t of instance x_t as follows:

$$F(x_{t1}, \ldots, x_{tp}) = \bar{y}_t \qquad \text{where } \bar{y}_t \in R. \tag{1}$$

```
[1]  $\forall \mathbf{x}_i \in$ Training Set
[2]      $normalize(\mathbf{x}_i)$
[3]  $\forall \mathbf{x}_t \in$ Test Set
[4]      $normalize(\mathbf{x}_t)$
[5]      $\forall \mathbf{x}_i \{\mathbf{x}_i \neq \mathbf{x}_t\}$: Calculate $Similarity(\mathbf{x}_t, \mathbf{x}_i)$
[6]      Let $Similars$ be set of $N$ most similar instances to $\mathbf{x}_t$ in Training Set
[7]      Let $Sum = \sum_{\mathbf{x}_i \in Similars} Similarity(\mathbf{x}_t, \mathbf{x}_i)$
[8]      Then $\bar{y}_t = \sum_{\mathbf{x}_i \in Similars} \frac{Similarity(\mathbf{x}_t,\mathbf{x}_i)}{Sum} F(\mathbf{x}_i)$
```

Figure 1: The Proximity Algorithm

There are a variety of instance-based algorithms in the literature. Here, the simplest one, called *proximity* algorithm is described in Figure 1. The proximity algorithm simply saves all training instances in the training set. The normalization algorithm maps each attribute value into the continuous range $(0 - 1)$. The estimate $\bar{y}_t$ for test instance $\mathbf{x}_t$ is defined in terms of a weighted similarity function of $\mathbf{x}_t$'s nearest neighbors in the training set. The similarity of two normalized instances is described by Equation 2.

$$Similarity(\mathbf{x}_t, \mathbf{x}_i) = \sum_{j=1}^{p} Sim(x_{tj}, x_{ij}) \tag{2}$$

where $Sim(x, y) = 1.0 - |x - y|$, and t_i.

The assumption in this approach is that the function is locally linear. For sufficiently large sample sizes this technique yields a good approximation for continuous functions. Another important property of instance-based regression is its incremental learning behavior. By default, the instance-based regression assumes that all the features are equally relevant. However, the prediction accuracy of this technique can be improved by assigning weights to the attributes. In order to reduce the storage requirements for large training sets, averaging techniques for the instances can be employed (Aha, Kibler & Albert, 1991).

2.2 Locally Weighted Regression

Locally weighted regression (LWR) is similar to the nearest neighbor approach described in the previous section, especially for three properties. First, the training phases of both algorithms include just storing the training data, and the main work is done during prediction. Such methods are also known as *lazy learning* methods. Second, they predict query instances by strong influence of the nearby or similar training instances. Third, they represent instances as real-valued points in p-dimensional Euclidean space. The main difference between IBL and LWR is that, while the former predicts instances by averaging the nearby instances, the latter makes predictions by forming an averaging model at the location of query instance. This local model is generally a linear or nonlinear parametric function. After a prediction for query instance is done, this model is deleted, and for every new query a new local model is formed according to the location of query instance. In such local models, neighboring instances of the query have large weights on the model, and distant instances have less or no weights. For a detailed overview of the locally weighted methods see (Atkenson, Moore & Schaal, 1997).

2.2.1 Nonlinear Local Models

Nonlinear local models can be constructed by modifying global parametric models. A general global model can be trained to minimize the following training criterion:

$$C = \sum_i L(f(\mathbf{x}_i, \boldsymbol{\beta}), y_i) \tag{3}$$

where y_i is the response value corresponding to the input vectors $\mathbf{x}_i$, and $\boldsymbol{\beta}$ is the parameter vector for the nonlinear model $\bar{y}_i = f(x_i, \beta)$ and L is the general loss function for predicting y_i. If this model is a neural net, then the $\boldsymbol{\beta}$ will be a vector of the synaptic weights. If we use least squares for the loss function L, the training criterion will be as follows:

$$C = \sum_i (f(\mathbf{x}_i, \boldsymbol{\beta}) - y_i)^2 \tag{4}$$

In order for making nearby points to the query to have more influence in the regression, a weighting factor can be added to the criterion.

$$C(q) = \sum_i [L(f(\mathbf{x}_i, \boldsymbol{\beta}), y_i))K(d(\mathbf{x}_i, \mathbf{q}))] \tag{5}$$

where K is the weighting or *kernel* function and $d(\mathbf{x}_i, \mathbf{q})$ is the distance between the data point $\mathbf{x}_i$ and the query $\mathbf{q}$. Using this training criterion, f becomes a local model, and it can have different set of parameters for each query point.

2.2.2 Linear Local Models

The well-known linear global model for regression is the simple regression, where the least squares approximation is used as training criterion. Such linear models can be expressed as:

$$\mathbf{x}_i\boldsymbol{\beta} = y_i \tag{6}$$

where $\boldsymbol{\beta}$ is the parameter vector. Whole training data can be defined with the following matrix equation.

$$\mathbf{X}\boldsymbol{\beta} = \mathbf{y} \tag{7}$$

where $\mathbf{X}$ is the training matrix whose ith row is $\mathbf{x}_i$ and $\mathbf{y}$ is a vector whose ith element is y_i. Estimating the parameters $\boldsymbol{\beta}$ using least squares criterion minimizes the following criterion:

$$C = \sum_i (\mathbf{x}_i\boldsymbol{\beta} - y_i)^2 \tag{8}$$

We can use this global linear parametric model, where all the training instances have equal weights; for locally weighted regression, by assigning nearby instances to the query point higher weights. This can be done by using the following weighted training criterion:

$$C = \sum_i [(f(\mathbf{x}_i, \boldsymbol{\beta}) - y_i)^2 K(d(\mathbf{x}_i, \mathbf{q}))]. \tag{9}$$

Various distance (d) and weighting (K) functions for local models are described in (Atkenson, Moore & Schaal, 1997). Different linear and nonlinear locally weighted regression models can be estimated with those functions.

3 Regression by Partitioning Feature Projections

In this section we describe the new regression method called Regression by Partitioning Feature Projections (RPFP). It is an instance-based method where most properties are similar to other instance-based methods such that it is a local, memory-based, lazy and distance-based approach. Furthermore, RPFP incorporates some advantages of eager approaches, while eliminating many limitations of both eager and lazy methods.

In Section 2, we described KNN and LWR. Both KNN and LWR use instances close to the location of query in the instance space. RPFP also uses similar instances, that is nearest instances to the query location. However, the space formed by the nearest instances are spherical in previous approaches, whereas it is a rectangular region for RPFP. This difference brings some advantages to RPFP. These advantages are related with the effect of irrelevant features, and the difference in the effects of all the features in different locations of the instance space. These benefits of using rectangular regions formed by nearest neighbors are described in Section 4.

On the other hand, RPFP is similar to LWR, in that, the effect of closest instances are larger than the others. However RPFP does not restrict the number of instances to K closest instances as in KNN and LWR.

One of the main differences of RPFP from KNN and LWR is that, the predictions are made separately for each feature, as done in the additive eager models. However RPFP handles the dependencies between features by making predictions in local regions of instance space. If the underlying phenomenon is additive itself, which is a frequent case for most real data sets, RPFP is suitable for those problems because of its additive nature.

3.1 RPFP Algorithm

In this section we give an intuitive description of RPFP algorithm, described in detail in the following sections. In RPFP, training involves associating each feature value with its corresponding target value, and a sorting process for each dimension on their feature values. The prediction is done for each feature dimension separately, either on the whole data or a partitioned portion of it. The final phase includes combining these results to form a single prediction for a given test instance. An overview of RPFP algorithm is shown in figure 2.

```
TRAINING:
[1]   Associate target values for each sorted feature value in all dimensions.

PREDICTION:
[2]   Find local weights and feature predictions for each feature dimension for
      given test instance.
[3]   Partition the instance space around the query location.
[4]   Find local weights and feature predictions for each feature dimension for the
      given test instance in the partitioned region.
[5]   Combine feature predictions to find final prediction, by selecting one of the
      results for each feature, obtained either in step 2 (on the whole data)
      or step 4 (on partitioned region), according to the local weight values.
```

Figure 2: An overview of RPFP algorithm

3.1.1 Training

Training involves simply storing the training set as their projections to the features as described above. If there are missing feature values, they are placed at the end of the feature dimensions by setting their feature values to infinity. These instances are ignored when prediction is done for that feature.

3.1.2 Approximation using Feature Projections

Approximation at feature projections is the first stage in the prediction phase of RPFP algorithm. Since the location of the query instance is known, the approximation is done according to this location. At each feature dimension, a separate approximation is obtained by using the value of the query instance for that feature.

Taylor's theorem states that if a region is local enough, any continuous function can be well approximated by a low order polynomial within it (Friedman, 1996). Given the linear equation to be approximated in the following form (10), the classical approach is to approximate coefficients of this equation using the least squares error criterion in (11).

$$\hat{y}_{qf} = \beta_{0f} + \beta_{1f} x_{qf} \tag{10}$$

$$E_f = \sum_{i=1}^{n} (y_i - \hat{y}_{if})^2 \tag{11}$$

where n is the number of training instances, $\hat{y}_{qf}$ is the approximation for query at feature f, y_i is the actual target value, and $\hat{y}_{if}$ is the target estimate for instance i.

RPFP employs the weighted linear least squares approximation for the feature predictions. Similar to the standard linear least squares approach, the parameters of (10), β_{0f} and β_{1f} are computed for each feature by employing a weight function to the least squares error, in order to determine weighted linear least squares approximation.

$$E_f = \sum_{i=1}^{n} w_{if} (y_i - \hat{y}_{if})^2 \tag{12}$$

where

$$w_{if} = \frac{1}{(x_{if} - x_{qf})^2} \tag{13}$$

By taking the derivative of (14) to minimize the error E_f, the parameters β_0 and β_1 for weighted linear least squares approximation are obtained.

$$E_f = \sum_{i=1}^{n} w_{if} (y_i - \beta_{0f} - \beta_{1f} x_{if})^2 \tag{14}$$

The weighted linear least squares approximation is not appropriate for two cases. One of them is for categorical features. Since there is no ordering between most categorical features, extracting a linear relation is not meaningful. On the other hand, if a categorical feature has an ordering between categorical values, weighted linear least squares approximation is employed.

Another case is for linear features. If all the instances have the same linear value for a particular feature dimension, the slope of the equation will be infinity.

In these two cases RPFP employs an averaging procedure instead of linear regression. In such cases, mean values of the target values are used as an approximation for such cases as given in (15). For categorical features, if the value of a feature does not match the feature value of the query instance, the contribution of that feature in the final prediction is excluded.

$$\hat{y}_{qf} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{15}$$

3.1.3 Local Weight

Some regions on a feature dimension may produce better approximations when compared to others. In order to obtain a degree of prediction ability of a region on feature dimension, we employ a measure in the prediction algorithm. If the region that query point falls in is smooth on a feature dimension, we give a high weight to that feature in the final prediction. In this way the effects of irrelevant features, as well as the irrelevant regions on a feature dimension, are eliminated. This establishes an adaptive or context sensitive nature, where at different locations in the instance space, the contribution of features on the final approximation differs.

In order to measure the degree of smoothness for continuous features we compute the distance weighted mean squared residuals. Residuals are differences between target values of the instances and their predicted values found by weighted linear least squares approximation for the feature value of each instance. We denote this measure with V_f as given in (17). By subtracting V_f from the variance of the target values of all instances, V_{all}, we find the explained variance according to the region that the query instance falls in. By normalizing it with the variance of training set we obtain a measure, called *prediction index* (PI) (19). We use the squared PI as the *local weight* (LW) for each feature (20).

$$V_{all} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n} \tag{16}$$

where $\bar{y}$ is the mean of target values of training set.

$$V_f = \frac{\sum_{i=1}^{n} w_i^{'}(y_i - \beta_0 - \beta_1 x_{if})^2}{\sum_{i=1}^{n} w_i^{'}} \tag{17}$$

where $w_i^{'}$ is defined in (18). For an overview of weight functions for regression problems see (Atkenson, Moore & Schaal, 1997).

$$w_{if}^{'} = \frac{1}{1 + (x_{if} - x_{qf})^2} \tag{18}$$

$$PI_f = \frac{V_{all} - V_f}{V_{all}} \tag{19}$$

$$LW_f = \begin{cases} PI_f^2 & \text{if } PI_f > 0 \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

In computing the local weight for categorical features, a refinement is required. We replace (21) with (17). Then we use the same procedure used for continuous features for the computation of local weight. Note that $w_{if}^{'}$ in (18) will be 1 for all the instances in the same category.

$$V_f = \frac{\sum_{i=1}^{N_c} w_i^{'}(y_i - \hat{y}_{qf})^2}{\sum_{i=1}^{N_c} w_i^{'}} \tag{21}$$

where N_c is the number of instances having the same categorical value, and $\hat{y}_{qf}$ is the average of their target values.

3.1.4 Partitioning Algorithm

Partitioning the set of training instances enables us to deal with dependencies among features. If a feature does not have a dependency with any other, we simply ignore the partitioning procedure for that feature and use the approximation for all projected instances. On the other hand, according to the local weight values obtained after partitioning, if an improvement is obtained, then prediction on partitioned region is used for that feature. Partitioning is an iterative procedure applied for each query instance, and the final region after partitioning may be different for each of them.

Partitioning improves the context-sensitive nature of RPFP such that, the edges of the final region, a hyper-rectangle, are not equal in length for each query. The length of edges are determined according to the relevancy of features for the prediction of the query. This causes longer edges for irrelevant features, and shorter edges for relevant ones. The region is formed by the partitioning algorithm that will be described in this section, by using an iterative procedure that continues until a small number of instances, say k, are left. K is taken as 10 by default in the experiments.

In the first step of partitioning, the predictions and local weights of the features are found and recorded. For the partitioning of the data, the feature with the highest local weight is used. Partitioning is done again on this feature dimension. The farthest instances to the query on this feature dimension are marked. The number of such instances are determined by using local weight of that feature. Large number of instances are removed from the instance space if local weight is high. Then, these are removed from all feature dimensions separately. If the feature selected for partitioning is nominal, simply all the instances having different nominal values are removed. After shrinking the marked instances from other feature dimensions, partitioning continues by selecting a new feature.

The partitioning algorithm applies a strategy, in order to select the right feature for partitioning. For example, if the feature selected in the first step has the highest local weight again, for the query in the second step, then the feature having the second highest local weight is selected. By this way, we can pass over possible ridges in the data set. Selecting a feature with small local weight may increase their local weights in forthcoming steps significantly. However, at a particular step, the features with zero local weights are not used for partitioning for that step, unless all local weights in a particular step are zero. This strategy decreases the effect of irrelevant features, especially in high-dimensional domains. Since all of the features may have been selected before partitioning ends, a counter is associated with each feature in order to give chance to different features each time.

A different strategy is applied for nominal features. If a nominal feature is selected for partitioning once, it is never used again for partitioning. The partitioning algorithm of RPFP is shown in Figure 3. The partitioning is repeated for all query instances from the beginning, by using a copy of the feature projections of the data obtained in the training phase.

At line 21, in Figure 3, number of steps for the partitioning is recorded to be used in the final prediction phase. At line 19, a partitioning of the remaining training set, D', is employed along the feature dimension, $MaxF$, selected for partitioning.

3.1.5 Prediction

After completing the partitioning and obtaining a final region, where the query instance is in the center of it, we compare local weights obtained for each feature for this region and for the whole region before partitioning. This comparison is performed for each feature separately. If the local weight of a feature for the initial projections is larger than that of final region, we use the initial computations for that feature. Otherwise, we use the computations on the final region for that feature in the final prediction.

If a query value for a feature is missing, that feature is not used in the final prediction. Finally a prediction is done for a query instance by computing the weighted average of feature predictions, where weights are the computed local weights. Prediction algorithm is shown in Figure 4.

4 Properties of RPFP

In this section, we describe important properties and problems for regression algorithms and we present theoretical evaluations of RPFP according to these properties. These properties show the behavior of RPFP in handling the curse of dimensionality problem,

```
[1]  $n^{'} \leftarrow n$; $S_{max} \leftarrow \log n$; $s \leftarrow 0$; $D^{'} \leftarrow D$
[2]  For $f = 1$ to $p$
[3]      $priority(f) \leftarrow S_{max}$

[4]  While $n^{'} > k$ and $s < S_{max}$
[5]      $s \leftarrow s + 1$
[6]      For $f = 1$ to $p$
[7]          if $x_{qf}$ is known then
[8]              compute and record $LW_f(s)$ and $\hat{y}_{qf}(s)$ on $D^{'}$
[9]      $MaxF \leftarrow$ any $f$ where $x_{qf}$ is known and $LW_f(s) > 0$
[10]     For $f = 1$ to $p$
[11]         if $LW_f(s) > 0$ and $x_{qf}$ is known then
[12]             if $priority(f) > priority(Maxf)$ then $MaxF \leftarrow f$
[13]             if $priority(f) = priority(Maxf)$ then
[14]                 if $LW_f(s) > LW_{MaxF}(s)$ then $MaxF \leftarrow f$
[15]     if $MaxF$ is continuous then
[16]         $priority(MaxF) \leftarrow priority(MaxF) - 1$
[17]     if $MaxF$ is nominal then
[18]         $priority(MaxF) \leftarrow 0$
[19]     $D^{'} \leftarrow partition(D^{'}, MaxF)$
[20]     $n^{'} \leftarrow$ size of $D^{'}$
[21] $S \leftarrow s$
[22] Return $D^{'}$
```

Figure 3: Partitioning Algorithm

its additive nature, its ability in handling dependencies between features, its vote for either bias or variance error, its complex nature in making predictions, its robustness to irrelevant features, its ability in handling large number of dimensions, its context sensitive nature, and finally its robustness to missing feature values.

4.1 Curse of Dimensionality

The curse of dimensionality is a problem for nearly all learning and prediction methods that do not make strong assumptions about the domains. There are some models that handles this situation with assumptions. Assuming that features separately contribute to the solutions is the assumption used in additive models. Another solution to this problem is proposed by the projection pursuit regression. The instance space is projected to a lower dimensional space (generally one or two dimensional). However, this approach also makes an assumption such that, the information in data can be evaluated by using only the projection of data to some projection axes. Assuming linearity between input features and target in the prediction problems can be seen as a sub-category of additive models and it is a strong assumption that is employed in classical linear regression and linear discriminant analysis, which are parametric models.

The strong assumptions made in prediction tasks introduce large bias errors in most domains. This is also what the curse of dimensionality problem causes in other non-

```
[1]  $Prediction \leftarrow 0$; $WeightSum \leftarrow 0$
[2]  $S$, $LW$ and $\hat{y}_{\mathbf{q}}$ are determined during partitioning process.

[3]  For $f = 1$ to $p$
[4]      if $x_{qf}$ is known then
[5]          if $LW_f(0) > LW_f(S)$ then
[6]              $Prediction \leftarrow Prediction + \hat{y}_{qf}(0)$
[7]              $WeightSum \leftarrow WeightSum + LW_f(0)$
[8]          else
[9]              $Prediction \leftarrow Prediction + \hat{y}_{qf}(S)$
[10]             $WeightSum \leftarrow WeightSum + LW_f(S)$
[11] $Prediction \leftarrow Prediction / WeightSum$
```

Figure 4: Prediction Algorithm

parametric learning methods. Therefore, generally the choice is whether to put up with strong assumptions or with the curse of dimensionality.

Most modern techniques for regression in the literature are developed in order to obtain better accuracies by eliminating assumptions employed in classical, generally linear and parametric methods. Hence, developing some measures to decrease the effect of curse of dimensionality is important for modern techniques in order to achieve higher accuracies.

The problem can be illustrated with a simple example. Consider a one dimensional input space, where all instances are uniformly distributed and feature values range from 0 to 1. In this situation half of the feature dimension contains half of the instances. If we add one more feature with the same properties to the instance space, using half of each feature dimension will include 1/4th of the instances. One more feature will decrease this ratio to 1/8, and so on exponentially. Adding new features will cause more sparse instance spaces. In order to keep the same ratio for the number of instances in a region we have to increase the volume of the region exponentially. This is because in high dimensional spaces it is impossible to construct regions that have small size simultaneously in all directions and containing sufficient training data; thus, using large regions for approximation causes large bias errors (Friedman, 1996). The following expression, described by Friedman (1996), explains the problem for KNN.

$$\frac{size(R_k)}{size(R_0)} = \left(\frac{k}{n}\right)^{1/p} \tag{22}$$

where k is the number of training instances in region R_k and R_0 is the instance space.

Thus, in high dimensions the size of the region will be close to R_0 even for $k = 1$. Curse of dimensionality is a much more important problem for KNN, when compared to eager methods. Nearly all eager learning approaches (e.g., Rule-based learning, tree-based learning and MARS) have some measures to decrease the effect of the curse of dimensionality. The most important one is to properly select the features to be included in the model, and decrease the number of dimensions. This is also the reason for the success of eager approaches against irrelevant features.

Another measure is the adaptive nature of partitioning eager approaches. For example, in a uniformly distributed space, after a normalization process, KNN always forms regions with a sphere shape, having the same diameter in all dimensions. However this volume is a hyper-rectangle for most eager approaches rather than a sphere or hypercube, since the edge lengths are determined according to the position of a region in the instance space. Intuitively, important features have smaller edges when compared to unimportant features at the location of region.

Some solutions similar to these measures is available for KNN, by using external feature selection and feature weighting algorithms before applying KNN (Wettscherect, Aha & Mohri, 1997). Feature selection techniques can eliminate irrelevant features and feature weighting can produce elliptic regions instead of spherical ones. Another desired property is that, the shape of this elliptic region must change according to the location of the query in the instance space.

On the other hand, the problem of curse of dimensionality is much more important for KNN when the task is regression instead of classification. There is an important empirical evidence that KNN can achieve high accuracies in many domains for classification. However this is not the situation for regression (Friedman, 1996). This property of KNN will be discussed in the following sections.

RPFP is a member of instance-based approaches, that are local, memory-based, lazy, non-parametric and do not depend on strong assumptions such as those described above. However, RPFP has some measures to decrease the effect of curse of dimensionality.

In the final prediction phase of RPFP, a subset of features are used in additive form, only for their main effects on the target. The curse of dimensionality does not effect their contributions, since the feature predictions are determined on a single dimension. For remaining features, the effect of curse of dimensionality is not severe. The partitioning algorithm either does not allow irrelevant features to effect partitioning (if their local weights are 0), or their effect are small since a dynamic partitioning occurs according to their local weights. The partitioning strategy of RPFP forms adaptive regions. According to the position of each query instance, the edge lengths of these regions for each feature dimension may change. For remaining features, predictions are done on these regions.

4.2 Classification versus Regression for Additive Models

The success of additive classification algorithms is pointed out by many different additive approaches in the machine learning literature. Holte (1993), Güvenir et.al. (1993,1998) and Frank et.al. (1998) show significant performance of their additive classification algorithms, on real data sets. However, regression implementations of those methods, do not show such significant success, when compared to regression methods that handle dependencies between features. An analogy by Frank et.al.(1998) shows that naive Bayes method is more successful in classification, when compared to its success in regression. In this paper, we show that RFP (Uysal & Güvenir 1999b), the additive version of RPFP is not as successful in performance when compared to RPFP, where we involve a partitioning procedure in order to handle interactions. However, RPFP does not exclude some advantages of additive methods, such as handling missing feature values naturally, and especially when some features additively effect the response in a given data set.

4.3 Bias-variance Trade-off

Following the considerations presented by Friedman (1997), two important error types collectively effect the success of learning approaches according to the underlying problem they are applied. They are *bias* and *variance* errors, caused by under-fitting and over-fitting respectively on the learning application. A decrease in one of those errors, generally causes an increase on the other. However the behavior of interaction between bias and variance differs according to the algorithm and problem the algorithms are applied. If we illustrate these error components with an example, large k values in the application of KNN algorithm may cause large bias error, on the other hand, small k values may cause large variance error.

Many factors are effective for these error components. The curse of dimensionality, model complexity, model flexibility, local vs. global approximations, assumptions of the learning approach, noise, missing attribute values, number of features and number of observations in applications are some of those. For example large number of features, small number of training instances, many missing values, large local approximation regions, strong assumptions and simple models are among reasons of bias error. The effect of these issues on RPFP will be discussed in the following sections.

An important result presented by Friedman (1997) is that for classification tasks the major component of the error is formed by variance, on the other hand, for regression problems the bias error becomes important. This is shown as the main reason for the success of the simple nearest neighbor approach such that it over-performs some sophisticated methods for many classification tasks even the curse of dimensionality problem of KNN causes strong bias. The success of additive approaches for classification is also reported by Guvenir et.al.(1996,1998).However, this is not the situation for regression, and the effect of bias error is much more important unless the underlying domain includes small number of features or large number of observations.

In learning problems, this trade-off is unavoidable and RPFP casts its vote for variance by employing many arguments to decrease the bias error. The way for handling bias error caused by the curse of dimensionality is described in the previous section. Besides, it does not make strong assumptions as non-parametric methods. It develops flexible, adaptive and locally weighted approximations in small local projections at each feature dimension for each query instance. All these things may increase the over-fitting, which causes an increase on the variance error. However empirical results show that RPFP is more successful than KNN, which justifies these ideas about the behavior of classification and regression for the bias-variance trade-off.

4.4 Model Complexity and Occam's Razor

William of Occam's Razor principle states that "Entities should not be multiplied beyond necessity" (Domingos, 1998). This idea has been accepted theoretically in machine learning with two different interpretations (Domingos, 1998). One of them is, "Given two models with the same accuracy, the simpler one must be selected because the simplicity is desirable in itself". The other interpretation states that, "Given two models with the same training-set error, the simpler one should be preferred because it is likely to have lower prediction error". The well known example for the second interpretation is the pruning applied in decision tree learning.

The second interpretation has been found inconvenient by many researchers recently and some theoretical and empirical work are published supporting this idea (Webb & Kuzmycz, 1996). An overview is given by Domingos (1998). It is also possible that complex models can produce better accuracies than simpler ones.

RPFP is flexible and complex, such that for different query locations in the instance space, producing infinite number of different local approximation functions on many different domains is possible. If we consider many different feature dimensions having such approximations, RPFP becomes more flexible and complex as the number of dimensions increase. The performance results of RPFP on real data sets vindicate those resent worries about the second interpretation of the Occam's Razor principle.

4.5 Irrelevant Features and Dimensionality

The sensitivity to irrelevant features is the most important problem for lazy methods. On the other hand, eager approaches in general (especially recursive partitioning methods) are successful in eliminating the effects of irrelevant features. For example, in regression tree induction, the partitioning starts from the most significant feature and continues recursively by processing less relevant ones. It is very likely that some irrelevant features will not be used even in constructing a regression tree.

The reason that irrelevant features cause problems in lazy learners comes from the distance measure used in those methods. In nearest neighbor approaches for example, nearest instances are determined according to a distance measure in p dimensional space, which is generally the Euclidean distance. In the computation of distances all features are given equal importance, including irrelevant ones. This may cause important instances for a query to go away from the query.

Irrelevant features do not cause much difficulty for RPFP, since distances are computed for each feature separately. Another important advantage of RPFP is that it is very likely for those features to take lower local weights, since the distribution of target values of nearest instances at any query location will be very close to the distribution of the whole target values in the training set (19). RPFP is capable of incorporating all features according to their relevancy on the query instance. If the irrelevant features or the relevance of features changes according to the locations of the instance space, this is handled by RPFP, due to its adaptive nature.

However large dimensionality may cause problems on some eager approaches, in addition to the curse of dimensionality problem. With regression trees, for example, after a small number of steps in the tree construction process, the number of instances at tree nodes may be exhausted before many relevant features get a chance. This second problem of high dimensionality is also resolved in RPFP, since all the features are used in the final prediction phase.

4.6 Context-sensitive Learning

RPFP is an adaptive or context-sensitive method in the sense that in different locations of the instance space the contribution of the features are different. This property is enabled with two characteristics of RPFP. One of them is the partitioning algorithm. The region formed around the query instance is determined adaptively; different features have different lengths of edges in the final region according to the location of query. The other one is in the local weights. Features may take different local weights according to the location of the query. On the other hand, the local weights of features will be different since different instances will be the neighbors at different feature dimensions. The difference in the neighbors will reduce possible over-fitting, similar to sampling approaches such as boosting (Breiman, 1996), which brings an advantage to RPFP. Nearly all eager approaches, to some extent, are context-sensitive, which is an advantage over KNN.

4.7 Missing Feature Values

It is very likely that some feature values may be unknown in the application domain. In relational databases, the most suitable form of databases for the most current learning techniques, the problem occurs frequently because all the records of a table must have the same fields, even if values are inexistent for most records (Matheus, Chan & Piatetsky-Shapiro, 1993). For example, in a hospital database including many fields for many laboratory tests and medical procedures, only a few of these fields will be filled in for any patient.

The natural and the best solution for handling missing values is leaving those places empty and not to distort the information in the data. Additive models or feature projection based methods handle missing values in that way, since each feature is evaluated separately. However, their limitation is that they assume all features to have independent effects on the target.

RPFP deals with missing values similar to additive or previous feature projection based models, and also resolve the interactions between features by applying a partitioning process. RPFP achieves this by applying approximations on feature projections using only known values, and in partitioning, for a selected feature dimension along with the partitioning occurs, by keeping missing valued instances of that feature.

5 Complexity Analysis

Since RPFP is a lazy approach, and stores all instances in the memory, a space proportional to the whole training data is required. Given a data set with n instances and m features this space is proportional to $m.n$. Again, for the training phase, the computational complexity of projecting instances to input features, which requires a sort operation on each feature, is $O(m.n.\log n)$. The computation of variance ($O(n)$) of target values for all training data is also computed in the training phase, and it does not change the above complexity.

Taking a copy of projections for a feature requires a complexity of $O(n)$. The computation complexity of local approximation in the first step of partitioning is again $O(n)$. The complexity of computing local weight is also $O(n)$, which also the total computation complexity at first partitioning step. The partitioning at each step removes, on the average, half of the instances. For the whole partitioning process the total computation for a single feature will be proportional to $2n$ since $n + n/2 + n/4 + \ldots \approx 2n$. If we compute the complexity for all features we obtain a complexity proportional to $O(m.n)$, which is equal to the complexity of KNN. If we consider situations for nominal features, this complexity does not change much. Prediction time is even shorter for nominal features then linear features. In the worst case where a nominal feature has two values, it requires on the average the same complexity. The test times of the algorithms, run on the real datasets also shows that the running test time of RPFP is proportional to KNN.

6 Comparisons of Regression Methods

In the previous sections we have described some properties and limitations of RPFP, and made some comparisons with the other lazy approaches KNN (Mitchell, 1997) and LWR (Atkenson, Moore & Schaal, 1997). A comparison of RPFP, KNN and LWR is summarized in Table 1 according to important properties. A detailed overview and comparison of many other regression techniques is also given in Uysal & Guvenir (1999a).

Table 1: Properties of Regression Algorithms. The ($\surd$) is used for cases if the corresponding algorithm handles a problem or it is advantageous in a property when compared to others.

Properties	RPFP	KNN	LWR
Adaptive (context sensitive)	✓		✓
Continuous at the boundaries of regions	✓	✓	✓
Nominal feature types can be evaluated	✓	✓	
All features can be evaluated	✓		
Incremental	✓	✓	✓
Handle strong dependencies between features	✓	✓	✓
Robust to irrelevant features	✓		
Local learning method	✓	✓	
Memory cost is large			
Robust to missing values	✓		
Normalization is not required	✓		
Regions formed around queries overlap	✓	✓	✓
Testing cost is low			
Training cost is low	✓	✓	✓

7 Empirical Evaluations

In this section empirical results of RPFP and other lazy regression methods are presented. The accuracy performance of the regression methods is based on the prediction error of the algorithms. Since the target values are continuous, the absolute difference between the prediction and the true target value in the test example is used. One common measure is *mean absolute distance* (MAD) (Weiss & Indurkhya, 1995; Weiss & Indurkhya, 1998). It is the mean of absolute error found for all test examples.

$$MAD = \frac{\sum_{t=1}^{T} |y_t - \hat{y}_t|}{T} \tag{23}$$

where T is the number of test instances.

However in order to get similar performance values for all datasets a modified version of MAD, relative error (RE) is used in the experiments (Weiss & Indurkhya, 1995; Weiss & Indurkhya, 1998). Relative error is the true mean absolute distance normalized by the mean absolute distance from the mean target value. Mean target value is computed using the training data.

$$RE = \frac{MAD}{\frac{1}{T}\sum_{i=1}^{T} |y_i - \bar{y}|} \tag{24}$$

Performance results in the experiments are reported as the average of relative errors measured by applying 10-fold cross-validation on datasets.

7.1 Algorithms Used in Comparisons

In this section the properties of algorithms used in experiments are briefly described.

7.1.1 RPFP

K is the parameter of RPFP[1] that defines the minimum number of instances allowed for a region determined by the partitioning algorithm; and it is set to 10.

7.1.2 KNN

The weighted KNN[2] algorithm described by Mitchell (1997) is used. It performs better than simple KNN that employs simple averaging. The instances close to the query have larger weights, and these weights are determined by inverse squared distance. The distance measure used is the Euclidean distance. A normalization on test and train input feature values is applied in order to obtain values range between 0 and 1. In computing Euclidean distance, for matching nominal values the difference is measured as 0, and for different nominal values on a single dimension 1 is assigned.

Missing values are filled with mean values of the feature if it is continuous, or filled with the most frequent categorical value, if feature is nominal.

7.1.3 LWR

The Weka program (Witten & Frank, 1999) is used to experiment Locally Weighted Regression method. Number of nearest instances is determined to be 10 as done for KNN and RPFP method.

7.1.4 RFP

In order to experiment whether RPFP handle interactions well, and performs better than an additive regression approach, Regression of Feature Projection (RFP) is used in the experiments (Guvenir & Uysal, 2000b). RFP is also a feature projection-based regression approach, where a partitioning on the data is not enabled. We can consider it as an additive regression method.

7.2 Real Datasets

The datasets used in the experiments are provided by Witten et.al. (1997). These data sets are also used as benchmark datasets by Frank et.al. (1998). Characteristics of the data sets (30 different domains) used in experiments are given in Table 2.

7.3 Accuracy

In order to measure accuracy performance of algorithms, relative errors on the real data sets are computed. We employed 10-fold cross-validation, and computed the average relative error of 10 different test instance sets for each data set. The relative errors of algorithms on real data sets are shown in Table 3. The best results, smallest relative errors, are typed in bold font. In order to compare algorithms, a pairwise comparison is done with respect to RPFP. The results that are significantly below that of RPFP are marked by (#) sign. On the other hand if the relative error of any algorithm is significantly better than that of RPFP, the result is marked by (*).

The pairwise comparisons are summarized in Table 4. According to the comparisons, RPFP is better than all three algorithms. The comparisons between LWR, KNN and

[1]Implementations of RPFP in C code is available from authors upon request.

[2]K is set to 10 for all experiments, and implementation of KNN is available from authors upon request.

Table 2: Datasets used for experiments.

Data sets	Abbr.	Instances	Missing values (%)	Numeric attributes	Nominal attributes
auto-mpg	AU	398	0.2	4	3
basketball	BK	96	0.0	4	0
bodyfat	BD	25	0.0	14	0
bolts	BL	40	0.0	7	0
breast tumor	BT	286	0.3	1	8
cholesterol	CL	303	0.1	6	7
cleveland	CV	303	0.1	6	7
cloud	CU	108	0.0	4	2
cpu	CH	209	0.0	6	1
echoMonths	EM	131	7.5	6	3
elusage	EU	55	0.0	1	1
fishcatch	FC	158	6.9	5	2
fruitfly	FF	125	0.0	2	2
housing	HO	506	0.0	12	1
hungarian	HG	294	19.4	6	7
longley	LO	16	0.0	6	0
lowbwt	LW	189	0.0	2	7
mbagrade	MB	61	0.0	1	1
pbc	PB	418	15.6	10	8
pharynx	PH	195	0.1	1	10
pollution	PO	60	0.0	15	0
pwLinear	PW	200	0.0	10	0
quake	QU	2178	0.0	3	0
schlvote	SV	38	0.4	4	1
sensory	SN	576	0.0	0	11
servo	SE	167	0.0	0	4
sleep	SL	57	2.0	7	0
strike	ST	625	0.0	5	1
veteran	VE	137	0.0	3	4
vineyard	VY	52	0.0	3	0

RFP can also be seen in the table, and these results do not show significant differences among them. Accordingly, LWR is better than RFP and worse than KNN, and RFP is better than KNN. These results also show that RPFP is much better than RFP, which is an additive regression method. This result shows the ability of RPFP in handling interactions between features.

In order to obtain a summary of the relative errors of the algorithms, average values on all the data sets are computed and shown in Table 5. The average errors shows that, RPFP has the smallest value, 0.74. In order to make a better comparison of the averages, and to prevent the effect of high values in the results (e.g. the error of LWR on PO data set, 3.89), a new average is computed using some of data sets, where all of the algorithms have results below 1, and where all algorithms accomplish a useful performance. These averages also do not change the relative performance comparisons. Consequently, the average error of RPFP is smaller than all others, and standard deviation values show that RPFP, KNN and RFP are robust to domain differences, which show the domain independent characteristic of RPFP.

The robustness of RPFP to missing values, irrelevant features and noise were also determined with an earlier study by using real data sets of Bilkent Function Approximation

Table 3: Relative Errors of Algorithms. Best results are typed with bold font.The results that are significantly below that of RPFP are marked by (#) sign. On the other hand if the relative error of any algorithm is significantly better than that of RPFP, the result is marked by (*).

Data Sets	RPFP	LWR	KNN	RFP
AU	0.34	0.48 #	**0.32**	0.43 #
BK	**0.88**	0.98 #	0.96 #	0.96 #
BD	**0.06**	0.16 #	0.31 #	0.41 #
BL	0.39	**0.18** *	0.53 #	0.36 *
BT	**0.96**	1.61 #	1.21 #	0.98
CL	**0.99**	3.31 #	1.09 #	1.04 #
CV	0.77	1.35 #	**0.62** *	0.90 #
CU	0.51	**0.44** *	0.58 #	0.50
CH	0.36	**0.26** *	0.27 *	0.51 #
EM	0.75	0.82 #	**0.69** *	0.81 #
EU	**0.50**	0.59 #	0.54 #	**0.50**
FC	0.34	**0.31** *	0.63 #	0.53 #
FF	**1.00**	1.26 #	1.34 #	1.05 #
HO	0.60	0.66 #	**0.59**	0.78 #
HG	0.62	1.08 #	**0.52** *	1.22 #
LO	0.26	**0.18** *	0.42 #	0.29 #
LW	2.00	**0.93** *	1.84 *	1.35 *
MB	1.26	**0.88** *	1.37 #	1.12 *
PB	1.03	2.17 #	1.01	**0.93** *
PH	0.84	0.93 #	**0.82**	0.93 #
PO	0.73	3.89 #	**0.67** *	0.77 *
PW	0.58	0.57	**0.48** *	0.74 #
QU	**0.99**	1.14 #	1.12 #	1.00
SV	**0.68**	1.21 #	0.83 #	0.82 #
SN	1.03	1.44 #	1.04	**1.02**
SE	0.45	**0.28** *	0.32 *	0.66 #
SL	0.96	0.99	**0.87** *	0.88 *
ST	0.86	0.78 *	0.94 #	0.86
VE	**0.77**	1.17 #	0.93 #	0.85 #
VY	0.69	**0.56** *	0.68	0.75 #

Repository (Guvenir & Uysal, 2000a). These experiments also shows that RPFP show better performance on real data sets, when compared to eager algorithms of machine learning literature, and statistics (Uysal, 2000).

7.4 Interactions

RPFP handles interactions in a similar way other eager partitioning approaches work, by partitioning the instance space. The best way to show how partitioning in RPFP handles interactions and generally increase accuracy for datasets having interactions is to compare it with its additive version. All other algorithms compared in the previous sections have this property. The following experiment show that RPFP also has this property as other eager partitioning algorithms.

The additive version of RPFP is obtained by excluding the partitioning from RPFP algorithm and simply by combining the feature predictions and obtaining the final prediction after the first step.

Table 4: Pairwise comparison of RPFP with LWR, KNN and RFP. In 18 of the data sets, RPFP is better than LWR and RFP.

	RPFP	LWR	KNN	RFP
RPFP	-	18	15	18
LWR	10	-	11	14
KNN	9	16	-	12
RFP	6	13	14	-

Table 5: Average relative errors of algorithms and their standard deviations. Standard deviation values show that RPFP, KNN and RFP are robust to difference domains

	RPFP	LWR	KNN	RFP
Average	0.74	1.02	0.78	0.80
Stand. Dev.	0.37	0.85	0.37	0.26
Average (< 1)	0.56	0.72	0.59	0.65
Stand.Dev.(< 1)	0.25	0.84	0.22	0.21

The experiment is done with a simple artificial dataset having two interacting features and 100 instances formed as shown in Figure 5. Here $x1$ and $x2$ are the input features and y is the target. The feature$x1$ takes binary values and $x2$ and y take continuous values from 0 to 50. The relative error of RPFP on this data set is 0.31, which is much smaller than that of additive RPFP, whose relative error is 1.35.

8 Conclusions

In this paper we have presented a new regression method RPFP. It is an instance-based, non-parametric, nonlinear, context-sensitive, and local supervised learning method based on feature projections. It achieves higher accuracy especially when compared to the well-known lazy approaches, KNN and LWR. The main drawback of RPFP is the lack of interpretation and its high prediction time, which is smaller than LWR, comparable to that of KNN and worse than that of RFP.

Further research for RPFP can be directed to overcome its limitations, such that interpretation can be enabled by determining relative importance of features, and interactions

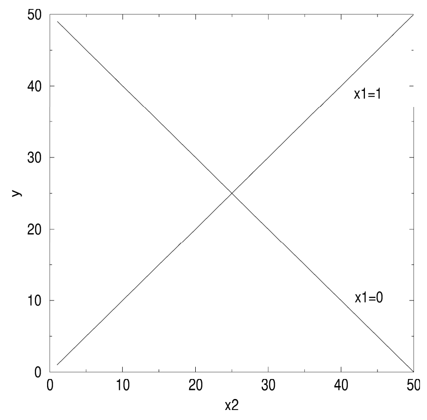

Figure 5: Artificial data set. $x1$ and $x2$ are input features.

between them. Another study can be done to measure performance of RPFP, when it is modified for classification tasks. Incorporating domain knowledge for stand-alone applications where explicit domain knowledge is available and incorporating misclassification cost for domains where a cost function is available can also be considered for further research.

References

[1] Aha, D. (1997). Special Issue on Lazy Learning, *Artificial Intelligence Review*, 11, No.1-5.

[2] Aha, D., Kibler, D., and Albert, M. (1991). Instance-based Learning Algorithms, *Machine Learning*, 6, 37-66.

[3] Atkenson, G. C., Moore, A. W., & Schaal, S. (1997). Locally Weighted Learning, *Artificial Intelligence Review*, 11, No.1-5, 11-73.

[4] Breiman, L. (1996). Bagging Predictors, *Machine Learning*, 24, 123-140.

[5] Dasarathy, B. V. (Ed.), Nearest Neighbor(NN) Norms: NN Pattern Classification Techniques, *CA:IEEE Computer Society Press*, Los Alamitos, 1991.

[6] Domingos, P. (1997). Context-sensitive Feature Selection for Lazy Learners, *Artificial Intelligence Review (Special Issue on Lazy Learning)*, 11, 227-253.

[7] Domingos, P. (1998). Occam's Two Razors, The Sharp and the Blunt, *Proceedings of KDD'98.*

[8] Frank E., Trigg L., Holmes G. & Witten I.H. (1998). Naive Bayes for Regression, *Working Paper 98/15*, Department of Computer Science, University of Waikato; April.

[9] Friedman, J. H. (1996). Local Learning Based on Recursive Covering, [ftp://stat.stanford.edu /pub/friedman/dart.ps.Z].

[10] Friedman, J. H. (1997). On Bias, Variance, 0/1-loss and the Curse of Dimensionality, *Data Mining and Knowledge Discovery*, 1, 55.

[11] Güvenir, H.A. & Sirin, İ. (1996). Classification by Feature Partitioning, *Machine Learning*, 23, 47-67.

[12] Güvenir, H.A. & Demiroz, G., İlter N. (1998). Learning Differential Diagnosis of Erythemato Squamous Diseases using Voting Feature Intervals, *Artificial Intelligence in Medicine*, 13, 147-165.

[13] Güvenir, H.A. & Uysal, İ. (2000a). Bilkent Function Approximation Repository, [funapp.cs.bilkent. edu.tr].

[14] Güvenir, H.A. & Uysal, İ. (2000b). Regression on Feature Projections, *Knowledge-Based Systems*, 13, No.4, 207-214.

[15] Holte, R. C. (1993) Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, *Machine Learning*, 11, 63-91.

[16] Kibler, D., Aha D. W., Albert, M. K. (1989). Instance-based Prediction of Real-valued Attributes, *Comput. Intell.*, 5, 51-57.

[17] Matheus, C. J., Chan, P. K., & Piatetsky-Shapiro, G. (1993). Systems for Knowledge Discovery in Databases, *IEEE Transactions on Knowledge and Data Engineering*, 5, No.6, 903-913.

[18] Mitchell, T.M. (1997). Machine Learning, *McGraw Hill.*

[19] Uysal, İ. & Güvenir, H.A. (1999a). An Overview of Regression Techniques for Knowledge Discovery, *Knowledge Engineering Review*, Cambridge University Press, 14, No.4, 1-22.

[20] Uysal, İ. & Güvenir, H.A. (1999b). Regression by Feature Projections, *Proceedings of Third European Conference on Principles and Practice of Knowledge Discovery in Databases*, Springer-Verlag, LNAI 1704, Prague.

[21] Uysal, İ. (2000). Instance-based Regression by Partitioning Feature Projections, *Technical Report BU-CE-0009*, Bilkent University Computer Engineering Department.

[22] Webb, G. & Kuzmycz, M. (1996). Further Experimental Evidence Against the Utility of Occam's Razor. *Journal of Artificial Intelligence Research*, 4, 397-417.

[23] Weiss, S. & Indurkhya, N. (1995). Rule-based Machine Learning Methods for Functional Prediction, *Journal of Artificial Intelligence Research*, 3, 383-403.

[24] Weiss, S. & Indurkhya, N. (1998). Predictive Data Mining: A Practical Guide, *Morgan Kaufmann*, San Francisco.

[25] Wettscherect, D., Aha, D.W., & Mohri, T. (1997). A Review and Empirical Evaluation of Feature-Weighting Methods for a Class of Lazy Learning Algorithms, *AI Review*, 11, No.1-5, 273-314.

[26] Witten, I.H., Frank, E. (1997). Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, *Morgan Kaufmann.*