Applied Intelligence 21, 57–79, 2004

Instance-Based Regression by Partitioning Feature Projections

İLHAN UYSAL AND H. ALTAY GÜVENIR
Department of Computer Engineering, Bilkent University, 06800 Ankara, Turkey
uilhan@cs.bilkent.edu.tr
guvenir@cs.bilkent.edu.tr

Abstract. A new instance-based learning method is presented for regression problems with high-dimensional data. As an instance-based approach, the conventional method, KNN, is very popular for classification. Although KNN performs well on classification tasks, it does not perform as well on regression problems. We have developed a new instance-based method, called Regression by Partitioning Feature Projections (RPFP) which is designed to meet the requirement for a lazy method that achieves high levels of accuracy on regression problems. RPFP gives better performance than well-known eager approaches found in machine learning and statistics such as MARS, rule-based regression, and regression tree induction systems. The most important property of RPFP is that it is a projection-based approach that can handle interactions. We show that it outperforms existing eager or lazy approaches on many domains when there are many missing values in the training data.

Keywords: machine learning, regression, feature projections

1. Introduction

Predicting values of numeric or continuous attributes is known as *regression* in statistics. Predicting real values is an important challenge for machine learning because many problems encountered in real life involve regression.

In machine learning, much research has been performed for classification, where the single predicted feature is nominal or discrete. Regression differs from classification in that the output or predicted feature in regression problems is continuous. Even regression is one of the earliest data analysis approaches studied in statistics, machine learning community also study on this problem since large number of real-life problems can be modelled as regression problems. Various names are used for this problem in the literature, such as function prediction, real value prediction, function approximation and continuous class learning. We will prefer its historical name, *regression*, in this article.

The term *eager* is used for learning systems that induce rigorous models during training. These models can be used to make predictions for instances drawn from the same domain. Induced models allow interpretation of the underlying data. Decision trees and decision rules are such models. On the other hand, *lazy* approaches do not construct models and delay processing to the prediction phase. In fact, the model is usually the normalized training data itself [1].

An important drawback of lazy learners is that they are not suitable for interpretation, since data itself is not a compact description when compared with other models such as trees or rules. Hence, the major task of these methods is prediction. A second limitation is that they generally have to store the whole data set in memory, resulting in a high space complexity.

However, *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 data by constructing global models. Another advantage is that learning in lazy approaches is very simple and fast, since it only involves storing the instances. Finally lazy approaches do not have to

construct a new model when a new instance is added to the data.

The most significant problem with lazy approaches is the one posed by irrelevant features. Some feature selection and feature weighting algorithms have been developed in the literature for this purpose [2]. However, these algorithms have also the common characteristic that they ignore the fact that some features may be relevant only in context. 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, and discussed in Section 4.4.

This paper describes a new instance-based 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 [3–5]. These projections can be generalized into feature intervals. 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 is their short classification time. The concept representation in the form of feature intervals can be transformed into decision rules easily. They are also robust to irrelevant features and missing values. However, the main shortcoming of feature projection based methods is that they ignore the interactions between features.

The RPFP method described in this paper is adaptive and robust to irrelevant features. It is not a simple first-order projection-based technique; it uses projections and also handles interactions between input variables.

Conceptually, consider a geographical surface plotted on a 2-dimensional plane. For a given geographical location, we try to catch a rectangle by cutting each dimension one by one. This operation is done carefully such that, on the remaining rectangle we should see the significant properties of the place, where the given location (query) falls in, by looking at the location from every direction separately. This technique enables interactions to be handled. The technique described in this paper consider the hyperspace instead of a restricted 2D-space, where the data set is placed on it and forms an hyper-rectangle by employing some computational and statistical measures and algorithms.

On the other hand, RPFP can cope with the curse of dimensionality problem, making it suitable for high-dimensional data. It does not require any normalization of feature values and successfully handles the cases with missing feature values. RPFP should be designed as a lazy, non-parametric, non-linear, and adaptive induction method that is based on feature projections in implementation.

The next section gives a short overview on related regression methods. In Section 3, RPFP algorithms are described. Important properties of the RPFP method are described in detail in Section 4. Section 5 gives a short summary of the theoretical comparisons of RPFP with many important regression algorithms. Detailed empirical comparisons are given in Section 6. The last section concludes the paper with some directions for future work.

2. Regression Overview

If the parametric form of the function to be approximated is known *a-priori*, the best approach is to estimate the parameters of this function. If the function is linear for example, the linear least squares regression can produce accurate results in the following form.

$$\hat{f}(\mathbf{x}_q) = \sum_{j=1}^{p} \beta_j \cdot x_{qj} + \beta_0 \tag{1}$$

here, p is the number of features, $\mathbf{x}_q$ is the query point, $\mathbf{x}_{qj}$ is the jth feature value of the query, β_j is the jth parameter of the function and $\hat{f}(\mathbf{x}_q)$ is the estimated value of the function for the query point $\mathbf{x}_q$.

However, the assumption that the approximated function is linear is a very strong one and causes large bias error, especially for many real domains. Many modern techniques have been developed, where no assumption is made about the form of the approximated function in order to achieve much better results. Tree and rule induction algorithms of machine learning are examples of such non-parametric approaches.

Additive regression models and feature projection based classification methods of machine learning such as Classification with Feature Projections (CFP) [3] improve the linear parametric form of the (1) by replacing the parameters in this equation with non-parametric

functions of the following form.

$$\hat{f}(\mathbf{x}_q) = \sum_{j=1}^{p} \hat{g}_j(\mathbf{x}_{qj}) \tag{2}$$

where $\hat{g}_j$ is the estimation for feature j.

With this form, the assumption that the approximated function is parametric is removed. However, we assume that the input features or variables additively form the approximated function. It is shown that for classification tasks of many real world domains, for example that of the datasets used for classification in the UCI repository [6], additive forms achieves high accuracy [3, 7]. Even though regression and classification are similar problems, one predicting a continuous target and the other predicting a categorical one, their characteristics are different, and they are investigated independently in the literature.

There are many approximation techniques that can cope with interaction effects. KNN and partitioning approaches such as rule-based regression [8, 9], tree-based regression [10, 11] and MARS [12] are such techniques. Among projection-based methods, only projection pursuit regression, PPR [13], handles interactions with the following model.

$$\hat{f}(\mathbf{x}_q) = \sum_{m=1}^{M} f_m \left(\sum_{j=1}^{p} \beta_{mj} \cdot \mathbf{x}_{qj} \right) \tag{3}$$

where M is the number of projections, β_{mj} is the jth parameter of the mth projection axis and f_m is the smooth or approximation function for the mth projection axis. Here the instances are not projected to feature dimensions. Instead, they are projected to projection axes. The whole model is constructed with M successive steps, and at each step of the model construction process, a new projection is found which is a linear equation. However, if there are both interactions and additive (main) effects in a domain, most models that handle interactions, including PPR, may loose some information by not evaluating main effects using individual features.

$$\hat{f}(\mathbf{x}_q) = \sum_{R' \in \{R_s, \mathbf{X}\}} \sum_{j=1}^{p} \hat{g}_{j,R'}(\mathbf{x}_q) I(j) \tag{4}$$

where R' is either the whole instance space $\mathbf{X}$ or the region obtained after s partitioning steps, R_s; and $I(j)$ is an indicator function whose value is either 0 or 1, according to the feature j.

RPFP incorporates interactions by partitioning the instance space. However, this partitioning does not produce disjoint regions, as C4.5 [14] does for classification and CART [10] for regression. Instead, these are overlapping regions similar to MARS, DART and KNN. Query instances are always close to the center of these regions, which is the case in most lazy approaches. If some features do not have interactions with others, which is the situation in most cases, RPFP incorporates the main effects of these features as much as possible by using the whole instance space, with more crowded instances as additive methods. It decreases the effects of the curse of dimensionality, which is a problem for almost all other approximation techniques except projection-based approaches. On the other hand, if a feature has interactions with others, the region formed after partitioning, R_s, is used for the contribution of that feature for prediction.

3. Regression by Partitioning Feature Projections

In this section we describe the new regression method called Regression by Partitioning Feature Projections (RPFP). RPFP incorporates some advantages of eager approaches, while eliminating most limitations of both eager and lazy methods. Like other instance-based methods it is local, memory-based and lazy.

3.1. RPFP Algorithm

In statistics, for the prediction of an outcome (target) the predictors (attributes or features) are employed in different ways in different methods according to their assumptions. For example, linearity is one major assumption in multivariate linear regression: It is assumed that each predictor involves separately and linearly on the outcome. Separate effects of predictors on the target are called as "main effect". On the other hand if some features collectively make a different effect on the target it is called as "interaction". RPFP is a projection-based approach that can handle interactions. However, if main effects are larger than interaction effects in a domain, or some features have only main effects, which is probably the case for most real world regression problems, the functional form of RPFP, given below in (4), enables those effects to be incorporated into the solution properly.

An important property of RPFP is that a different approximation is done for each feature by using the projections of the training instances on each

feature dimension separately. These approximations may be different for each feature and for each query point. A partitioning strategy is employed and some portion of the data is removed from the instance space at each step. The same approximations are repeated for a sequence of partitioning steps, until a partition containing a number of instances greater than a predefined parameter k, is reached. Figures 2 to 5 illustrate the RPFP method on a sample data set.

The procedure described above is applied for all query instances. This produces different regions and different contribution of features for each query in the instance space, thus providing context-sensitive solutions.

3.1.1. Training. Training involves simply storing the training set as their projections to the features. This is done by associating a copy of the target value with each feature dimension, then sorting the instances for each feature dimension according to their feature values. If there are missing values for a feature, the corresponding instances are placed at the end of the list for that feature dimension. Missing values of features are not included in the computations.

3.1.2. Approximation Using Feature Projections. Approximation at feature projections is the first stage in the prediction phase of the 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 [11]. By determining a different linear equation for each different query value at feature dimensions, we can form the function $\hat{g}_{j,R'}(\mathbf{x}_q)$ in (4), even though it is complex.

Given the linear equation to be approximated in the following form (6), the classical approach is to approximate coefficients of this equation using the least squares error criterion in (6).

$$\hat{y}_{qf} = \beta_{0f} + \beta_{1f} x_{qf} \tag{5}$$

$$E_f = \sum_{i=1}^{n} (y_i - \hat{y}_{if})^2 \tag{6}$$

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

RPFP employs the weighted linear least squares approximation for the feature predictions. Similar to the standard linear least squares approach, the parameters of (5), β_{0f} and β_{1f} for each feature are found by employing a weight function to the least squares error, in order to determine the weighted linear least squares approximation.

$$E_f = \sum_{i=1}^{n} w_{if} (y_i - \hat{y}_{if})^2 \tag{7}$$

where

$$w_{if} = \frac{1}{(x_{if} - x_{qf})^2} \tag{8}$$

The weight measure (8) implies that each instance is weighted according to the inverse square of its distance from the query. The weighted linear least squares approximation is not appropriate for categorical features. Since there is no ordering between most categorical features, extracting a linear relation is not useful. On the other hand, if the categorical values have an ordering, weighted linear least squares approximation shall be employed. By taking the derivatives of (9) to minimize the error E_f, the parameters β_0 and β_1 for weighted linear least squares approximation are found.

$$E_f = \sum_{i=1}^{n} w_{if} (y_i - \beta_{0f} - \beta_{1f} x_{if})^2 \tag{9}$$

From $\frac{\partial E}{\partial \beta_{0f}} = 0$

$$\beta_{0f} \left(\sum_{i=1}^{n} w_{if} \right) + \beta_{1f} \left(\sum_{i=1}^{n} x_{if} w_{if} \right) = \sum_{i=1}^{n} y_i w_{if} \tag{10}$$

From $\frac{\partial E}{\partial \beta_{1f}} = 0$

$$\beta_{0f} \left(\sum_{i=1}^{n} x_{if} w_{if} \right) + \beta_{1f} \left(\sum_{i=1}^{n} x_{if}^2 w_{if} \right) = \sum_{i=1}^{n} x_{if} y_i w_{if} \tag{11}$$

By solving the above equations, β_{0f} and β_{1f} are found as follows.

$$\beta_{0f} = \frac{\sum_{i=1}^{n} y_i w_{if} - \beta_{1f} \sum_{i=1}^{n} x_{if} w_{if}}{\sum_{i=1}^{n} w_{if}} \tag{12}$$

$$\beta_{1f} = \frac{SP_f}{SSx_f} \tag{13}$$

where

$$SP_f = \sum_{i=1}^{n} x_{if} y_i w_{if} - \frac{\left(\sum_{i=1}^{n} x_{if} w_{if}\right)\left(\sum_{i=1}^{n} y_i w_{if}\right)}{\sum_{i=1}^{n} w_{if}} \tag{14}$$

and

$$SSx_f = \sum_{i=1}^{n} x_{if}^2 w_{if} - \frac{\left(\sum_{i=1}^{n} x_{if} w_{if}\right)^2}{\sum_{i=1}^{n} w_{if}} \tag{15}$$

If all the instances have the same linear value for a particular feature dimension, the slope of the equation will be infinity. This situation can be determined by examining the value of SS_{xf} in (15). If $SS_{xf} = 0$, we can not employ the weighted linear least squares approximation. In this case, we employ an averaging procedure instead of linear regression. In RPFP, means of the target values are used as an approximation for such cases as given in (16). The same approximation is used for categorical features. If the value of a categorical feature does not match the feature value of the query instance, the contribution of that feature in the final prediction is excluded.

$$\hat{y}_{qf} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{16}$$

3.1.3. Local Weights. Some regions on a feature dimension may produce better approximations than others. If the region that the query point falls in is smooth, a high weight is given to that feature in the final prediction. In this way, the effects of irrelevant features, and that of the irrelevant regions on feature dimensions are eliminated. This gives 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 by V_f as defined in (18). By subtracting it from V_{all}, the variance of the target values of all instances, defined in (17), we find the explained variance according to the region the query instance falls in and by normalizing it with the variance of the training set, we obtain a measure, called *prediction index* (PI) (20). We use the squared PI as the *local weight* (LW) for each feature (21).

$$V_{\text{all}} = \frac{\sum_{i=1}^{n} (y_i - \bar{y})^2}{n} \tag{17}$$

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

$$V_f = \frac{\sum_{i=1}^{n} w'_{if} (y_i - \beta_0 - \beta_1 x_{if})^2}{\sum_{i=1}^{n} w'_{if}} \tag{18}$$

where w'_{if} is defined in (19).

$$w'_{if} = \frac{1}{1 + (x_{if} - x_{qf})^2} \tag{19}$$

$$PI_f = \frac{V_{\text{all}} - V_f}{V_{\text{all}}} \tag{20}$$

$$LW_f = \begin{cases} PI_f^2 & \text{if } PI_f > 0 \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

To compute local weights for categorical features, a refinement is required. By replacing (18) by (22) for such features, we can use the same procedure as for continuous features to compute local weights. Note that w'_{if} in (19) will be 1 if all of the categorical values are the same.

$$V_f = \frac{\sum_{i=1}^{N_c} w'_i (y_i - \hat{y}_{qf})^2}{\sum_{i=1}^{N_c} w'_{if}} \tag{22}$$

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 enables us dealing with interactions among features. If there are no interactions among some features, we use the results obtained and recorded for these features before the partitioning of the instance space. Partitioning is an iterative procedure applied to each query instance, where the remaining final region may differ for each query instance.

Partitioning is an iterative procedure that goes on, as far as, there are more than k instances in the final region. 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, according to the relevancy of features for the prediction of the query. This causes longer edges for less relevant features, and much shorter edges for relevant ones (see Figs. 2 to 5).

In the first step of partitioning, the predictions and local weights of the features are found and recorded. The feature having the highest local weight is used first. Partitioning is done on this feature dimension. The instances farthest from the query value on this feature dimension are marked. The number of these instances are determined by using the local weight of that feature, then they are removed on all feature dimensions. If the feature selected for partitioning is nominal, then all the instances having different nominal values on that feature are also removed. After shrinking the marked instances on all feature projections, partitioning continues by selecting a new feature at each step.

The partitioning algorithm applies a strategy to select the right feature for partitioning. For example, if the feature selected in the first step has the highest local weight again in the second step, then the feature having the second highest local weight is selected. In this way, we can pass possible ridges in the data set and we give chance to different features even if they have less local weight. This selection may lead to an increase in their local weights in forthcoming steps. 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 the features may have been selected in previous steps, a counter is associated with each feature in order to give opportunity to different features at each step. Different strategies can be developed further to increase the efficiency of the partitioning.

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 Fig. 1. The partitioning

```
[1]   n' ← n; S_max ← log n; s ← 0; D' ← D
[2]   for f = 1 to p
[3]      priority(f) ← S_max
[4]   end for

[5]   while n' > k and s < S_max
[6]      s ← s + 1
[7]      for f = 1 to p
[8]         if x_qf is known then
[9]            compute and record LW_f(s) and ŷ_qf(s) on D'
[10]        end if
[11]     end for
[12]     MaxF ← any f where x_qf is known and LW_f(s) > 0
[13]     for f = 1 to p
[14]        if LW_f(s) > 0 and x_qf is known then
[15]           if priority(f) > priority(Maxf) then MaxF ← f end if
[16]           if priority(f) = priority(Maxf) then
[17]              if LW_f(s) > LW_MaxF(s) then MaxF ← f end if
[18]           end if
[19]        end if
[20]     end for
[21]     if MaxF is continuous then
[22]        priority(MaxF) ← priority(MaxF) − 1
[23]     end if
[24]     if MaxF is nominal then
[25]        priority(MaxF) ← 0
[26]     end if
[27]     D' ← partition(D', MaxF)
[28]     n' ← size of D'
[29]  end while
[30]  S ← s
```

Figure 1. Partitioning algorithm.

is repeated for all query instances by using a copy of the feature projections of the data obtained in the training phase.

At line 30, in Fig. 1, the number of steps for the partitioning is recorded to be used in the final prediction phase. At line 27, a partitioning of the remaining training set, D', is employed along the feature dimension, *MaxF*, selected for partitioning. The *partition* call at line 27 refers to a simple procedure which is not described in Fig. 1. It simply removes the instances located far to the query value on feature *MaxF*.

At any particular step of partitioning, we must determine the number of instances, n', that will remain after partitioning according to the local weight of the selected feature. If we use local weight directly as a criterion for removing instances, since it takes values between 0 and 1, all the instances may remain or all of them may be removed for extreme values (e.g. for 0 or 1). The solution to this problem is found by windowing the local weight to a narrower interval. Its size is determined by a windowing constant, c_w, that takes values between 0 and 0.5, giving a local weight interval, $[0.5 \mp c_w]$. Local weights are transformed to this interval. Thus, for $c_w = 0.3$, the value we have used in experiments, the largest local weight becomes $LW_{\max} = 0.8$ and the smallest one becomes $LW_{\min} = 0.2$ after this transformation. The equation used to determine the number of instances that will remain is given below (23).

$$n_a = (n_b - m_f)(LW_{\max} - (LW_{\max} - LW_{\min})LW_f) + m_f \tag{23}$$

where n_a and n_b are number of instances after and before partitioning respectively, and m_f is the number of missing values at dimension f.

Once we determine the number of instances that will remain, we keep that many instances by removing the farthest instances from the query. The instances having missing values for that feature are excluded from this marking process, and they are preserved as their different feature values may be useful in the final prediction. However, we exclude missing values in the computations.

An example training set and its partitioning on features x and y is illustrated in Figs. 2–5. In this example, we suppose k is 5, that is partitioning goes on as far as $k \geq 5$.

3.1.5. The Prediction. After the partitioning, we obtain a final region, with the query instance in the center of it. We compare local weights obtained for a feature for the instance space before partitioning.

This comparison is performed for each feature separately. If the local weight of a feature on the initial projections of the instances is larger than the projections, main effects of those features are regarded in the final prediction. Otherwise, we use the feature predictions and local weights computed on the final region.

If a 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 local weights. The prediction algorithm is shown in Fig. 6.

3.2. RPFP-N Algorithm

We have developed the RPFP-N algorithm by modifying RPFP in order to use it for domains having noisy target values. Instance-based algorithms are robust to noisy or extreme input feature (predictor) values since the query instances will be far from these instances and their effect will be very small. However, if the target values of training instances are noisy, this situation must be handled.

We have modified the RPFP algorithm, by changing only the feature prediction phase, described in Section 3.1.2 in order to cope with noisy target values. RPFP-N employs an averaging procedure, instead of weighted linear least squares approximation for feature prediction. This is *distance weighted median* measure and its algorithm is described in Fig. 7, which is used for both categorical and continuous features. For categorical features, the instances which are in the same category with the feature value of the query instance are used for computation of both feature prediction and local weights. In the algorithm, Eq. (19) is used as the weight function for feature prediction.

After determining the prediction for a feature; in order to determine its local weight, Eq. (22) is employed in Eqs. (20) and (21), for both categorical and continues features.

4. Properties of RPFP

In this section, we discuss important properties and problems for regression algorithms and evaluate RPFP according to them.

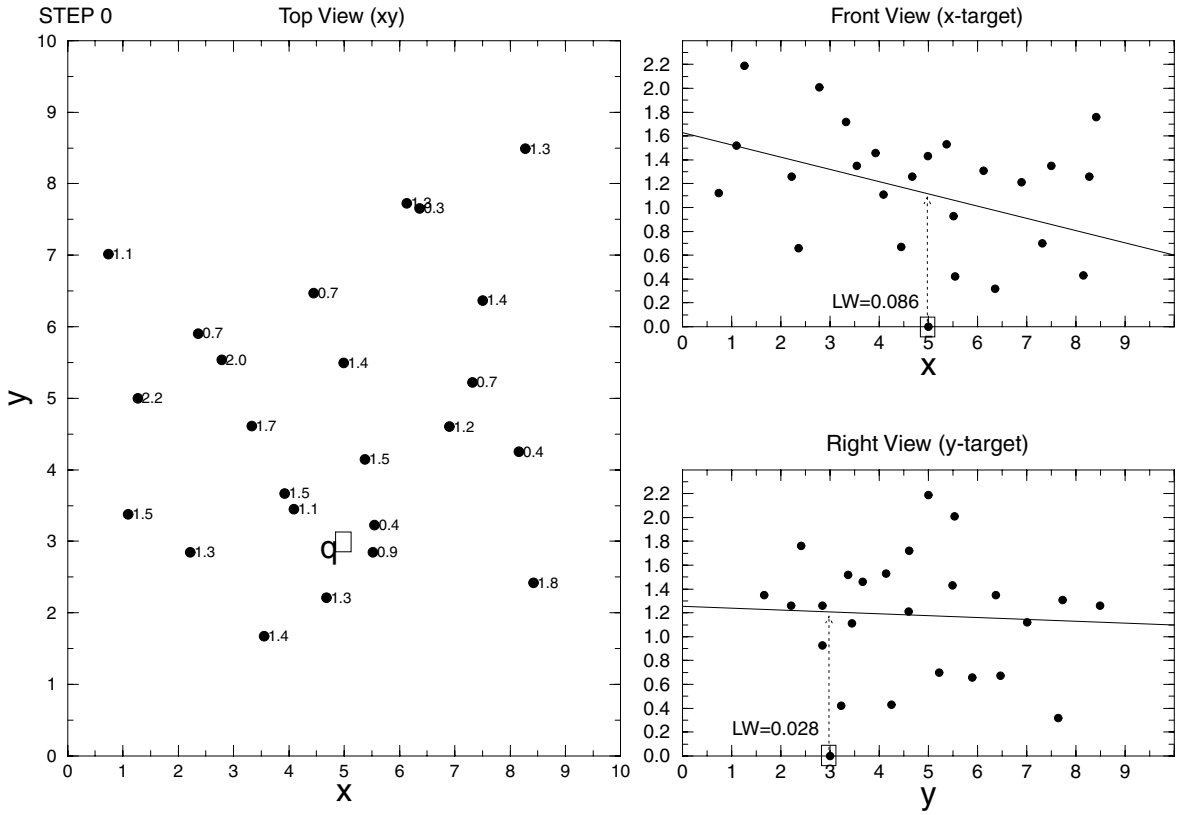

Figure 2. Data set before partitioning. Step 0.

4.1. Curse of Dimensionality

The curse of dimensionality is a problem for nearly all learning and prediction methods that do not apply strong assumptions about the domain. There are some models that handle this situation with assumptions. For example in additive models it is assumed that features separately contribute to the solutions, as in (2). Another solution to this problem comes with projection pursuit regression, which projects the instance space to a lower dimensional space (generally one or two dimensional). However, this approach assumes 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 the target in prediction problems can be seen as a sub-category of additive models by comparing (1) and (2); and it is a strong assumption that is employed in classical linear regression and linear discriminant analysis, which induce parametric models.

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 $1/4$th of the instances. One more feature will decrease this ratio to $1/8$, and so on exponentially. Adding new features will cause instance spaces to become even more sparse. 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 yet contain sufficient training data; thus, using large regions for approximation causes large bias errors. The following expression, described by Friedman (1996), explains the problem for KNN [15].

$$\frac{size(R_k)}{size(R_0)} = \left(\frac{k}{n}\right)^{1/p} \tag{24}$$

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

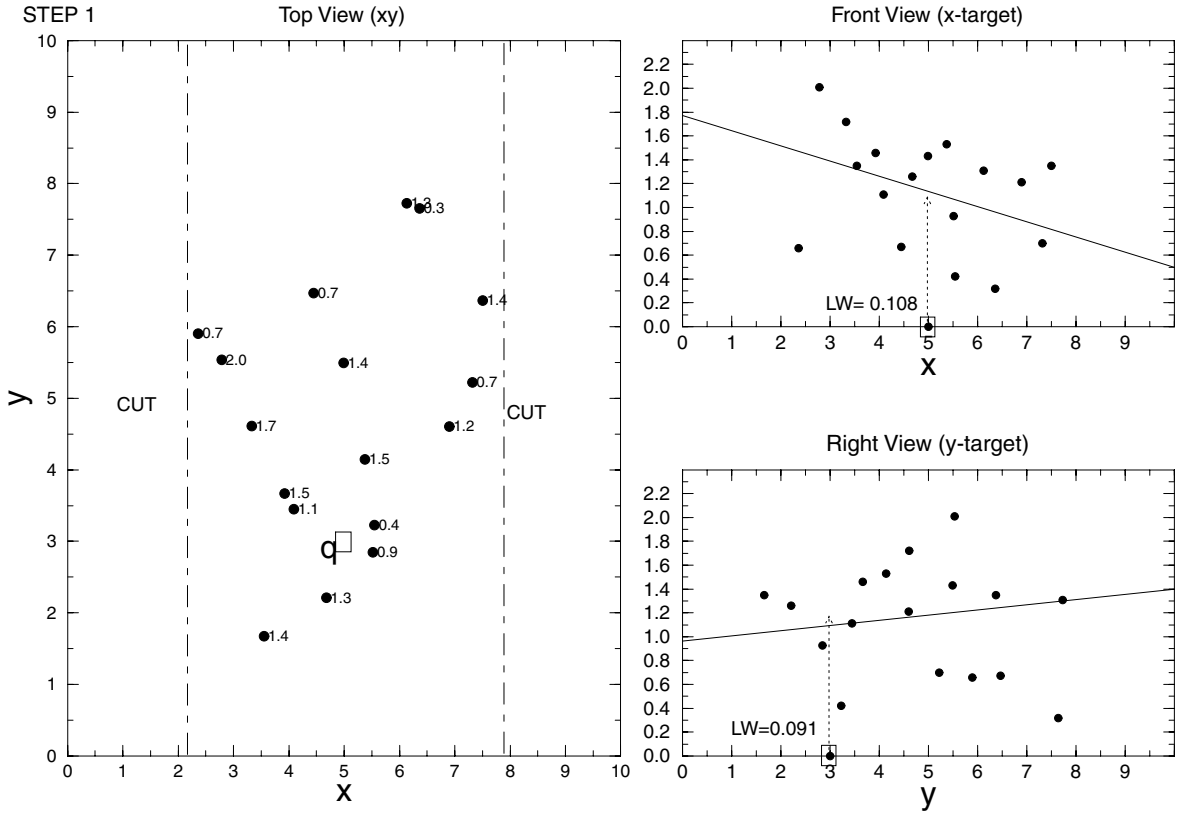

Figure 3. Step 1.

dimensions the size of the region will be close to R_0 even for $k = 1$.

The strong assumptions made in prediction tasks cause large bias errors in most domains. This is also what the curse of dimensionality causes in other non-parametric learning methods. Therefore, generally the choice is whether to put up with strong assumptions or with the curse of dimensionality. This problem was discussed in detail in the literature [11] and it is a critical issue for lazy and instance based approaches such as KNN [2], especially for regression problems.

RPFP is a member of the class 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 uses measures to decrease the effect of the 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 affect their contributions, since the feature predictions are determined on a single dimension. For remaining features, the effect of the curse of dimensionality is not severe. Either the partitioning algorithm does not allow irrelevant features to affect partitioning (if their local weights are 0), or their effects 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. *Bias-Variance Trade-off*

Following the considerations presented by Friedman [15], two important error types collectively affect the success of learning approaches according to the underlying problem to which they are applied. They are *bias* and *variance* errors, caused by under-fitting and over-fitting respectively in the learning application. A decrease in one of these errors, generally causes an

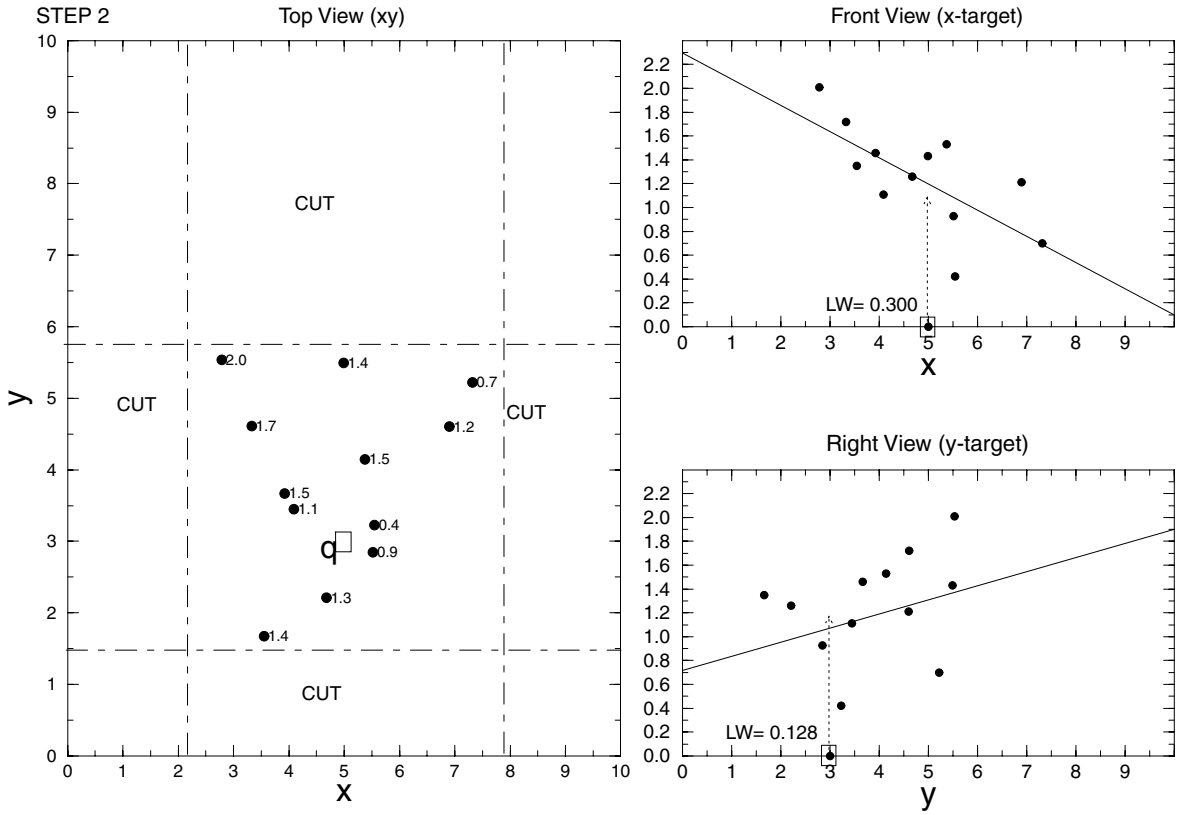

Figure 4. Step 2.

increase on the other. However the behavior of interaction between bias and variance differs according to the algorithm and the problem to which it is applied. To illustrate these error components with an example, large k values in the application of KNN algorithm may cause large bias error, while small k values may cause large variance error.

Many factors affect these error components including the curse of dimensionality, model complexity, model flexibility, local vs. global approximations, assumptions of the learning approach, noise, missing attribute values, the number of features and the number of observations in applications. 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 the reasons for bias error. The effect of these issues on RPFP will be discussed in the following sections.

An important result presented by Friedman is that for classification tasks the major component of the error is formed by variance [15]. In contrast, bias error is more important for regression tasks. This is the main reason for the success of the simple nearest neighbor approach, which outperforms some sophisticated methods for many classification tasks, even though the curse of dimensionality problem causes large bias. However, this is not the situation for regression, and the effect of bias error is much greater, unless the underlying domain includes a small number of features or a large number of observations.

In learning problems, this trade-off is unavoidable and RPFP employs many techniques to decrease bias while sacrificing variance error. Its method for handling bias error caused by the curse of dimensionality is described in the previous section. In addition, it makes weaker assumptions than 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 over-fitting, which causes an increase in the variance error.

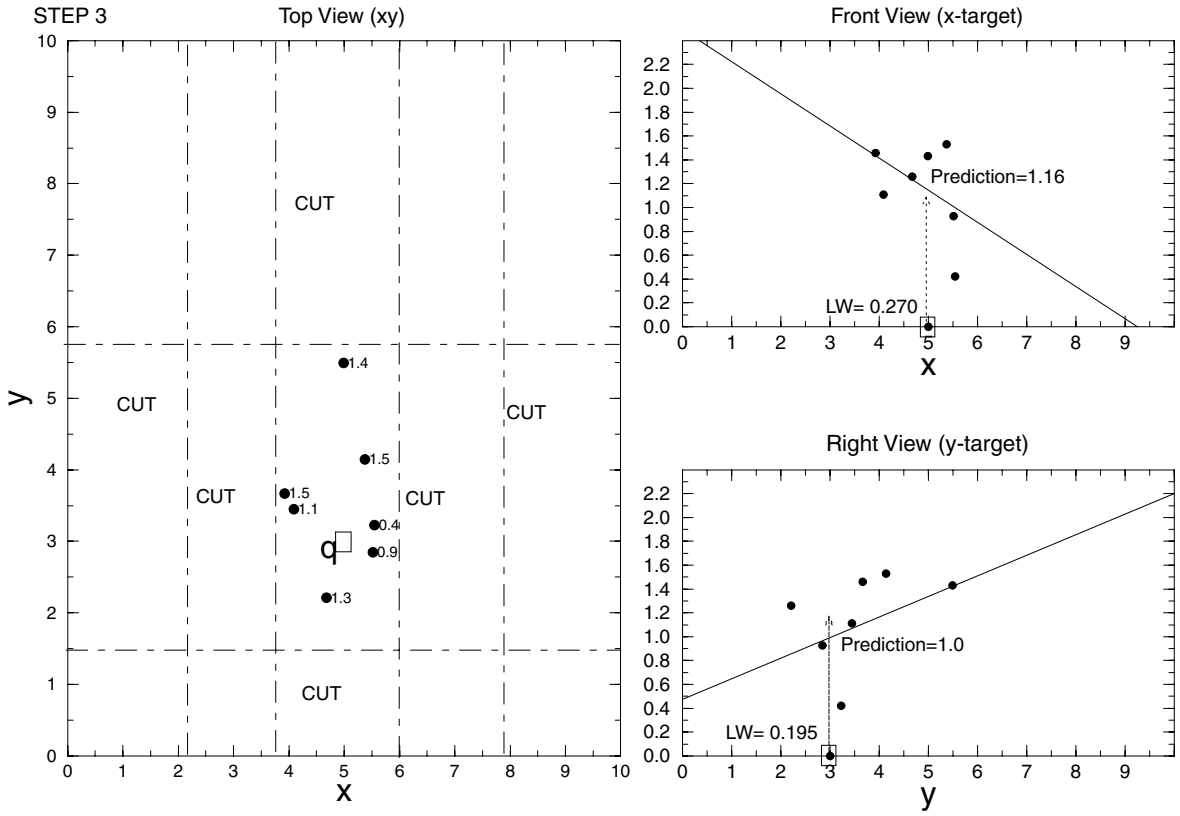

Figure 5. Step 3. We suppose there is interaction between x and y, as their local weights after partitioning process (step 3) are larger than their initial local weights (step 0). Final prediction $= 1.16 \times 0.270 + 1.0 \times 0.195 = 1.09$.

```
[1]   Prediction ← 0; WeightSum ← 0
[2]   S (# of partitioning steps), LW and ŷ_q are determined by partitioning algorithm

[3]   For f = 1 to p
[4]     if x_qf is known then
[5]       if LW_f(0) > LW_f(S) then (local weights before and after partitioning are compared)
[6]         Prediction ← Prediction + ŷ_qf(0)
[7]         WeightSum ← WeightSum + LW_f(0)
[8]       end if
[9]       else
[10]        Prediction ← Prediction + ŷ_qf(S)
[11]        WeightSum ← WeightSum + LW_f(S)
[12]      end else
[13]    end if
[14]  end for
[15]  Prediction ← Prediction/WeightSum
```

Figure 6. Prediction algorithm.

[1] $sum \leftarrow 0$
[2] $weight \leftarrow \sum_{i=1}^{n'} w'_{if}$
[3] sort instances according to their target values
[3] **while** $sum < weight/2$ take a new instance in the sorted order
[4] $feature\ prediction \leftarrow y_i$
[5] $sum \leftarrow sum + w'_{if}$
[6] end **while**
[7] $\hat{y}_{qf} \leftarrow feature\ prediction$ using equation 19

Figure 7. Weighted median algorithm.

For the bias-variance dilemma, we voted for the variance error in order to decrease the bias error, which is more important for regression problems. On the other hand, the local weight applied to feature predictions, prevent in some extent this error component since if there are noisy or extreme instances around the query in one dimension, its local weight will be low, and its effect will be prevented or decreased.

Empirical results show that RPFP yields lower error rates than KNN on regression tasks (as we will show), and decrease the variance error which is major part (error component) for regression.

4.3. Irrelevant Features and Dimensionality

An 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 (20).

4.4. Context-Sensitive Learning

RPFP is an adaptive or context-sensitive method in the sense that in different locations of the instance space the contributions of the features are different. This property is enabled by two characteristics of RPFP. One of them is its 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 the query. The other one is the use of local weights. Features may take different local weights according to the location of the query. In addition to this, the local weights of features will differ because different instances will be the neighbors at different feature dimensions.

Different sets of neighbors in different dimensions, on the other hand, reduce possible over-fitting by establishing an implicit boosting. Boosting methods in the literature produce multiple versions of data by making bootstrap replicates of the learning set and using these as new learning sets. A similar aprroacgh, bagging predictors [16], for example, is a method for generating multiple versions of a predictor and using these to get an aggregated predictor. RPFP has the properties of both bagging and boosting implicitly. It is similar to bagging since predictions for different features are made seperately and they are aggregated to reach the final result; and similar to boosting since the same data is used again for different features on different distances from the query, leading to a different sampling for each feature.

Nearly all eager approaches, to some extent, are context-sensitive, which is an advantage over KNN.

4.5. Missing Feature Values

It is very likely that some feature values may be unknown in the application domain. In relational databases, the most suitable type of databases for most current learning techniques, the problem occurs frequently because all the records of a table must have the same fields, even if values are non-existent for most records [17].

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

4.6. Interactions

If some input features have inter-relationships such that the effect of any feature on the target is dependent on

one or more different input features, those relations are called interactions. Appropriate handling of interactions may have an important impact on accuracy in some data sets. By making predictions in local query regions, RPFP can handle interactions properly.

Some research on classification methods and real data sets shows that generally the main effects of the features are sufficient to determine the target values [3, 8]. If some features have only main effects on targets, RPFP makes predictions for those features by using the whole instance space (instead of the local region determined by partitioning) since a large number of training instances allows better approximations. Another limitation of some partitioning methods, such as regression tree induction, is that the partitioning always occurs with many variables and this allows handling of only high-order interactions. This problem makes it difficult to approximate even some simple forms such as linear functions [12].

Dealing with interactions is particularly important in regression problems since bias error is more important to achieve better prediction accuracy. Refer to the earlier discussions about bias-variance trade off and comparison between RPFP and RPFP-A in the following sections, where empirical experiments are presented. The bias error for additive methods comes from the assumption that we can predict an outcome by evaluating features seperately. This can be illustrated by a simple example. Suppose there are two features representing edges of a rectangle; an additive method may well approximate the peripheral of the rectangle, however it is difficult to make a similar success for the prediction of the area of it. The assumption here, leading to bias error, is that the data has an additive nature as for the prediction of the peripheral. Handling interactions properly will be useful to decrease bias error for regression problems.

4.7. Complexity Analysis

Since RPFP is a lazy approach, and stores all instances in 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 $m \times n$. In the training phase, the computational complexity of projecting instances to input features, which requires a sort operation on each feature, is $O(m \times n \times \log n)$. 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 weights is in fact $O(n)$, which is also the total computation complexity at the 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 $n + n/2 + n/4 + \cdots \approx 2n$. If we compute the complexity for all features we obtain a complexity proportional to $O(m \times n)$, 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 for 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 show that the running time of RPFP is proportional to KNN.

5. Comparison of Regression Methods

In the previous sections we have discussed the main features and limitations of RPFP. In this section, we summarize its properties in comparison with other important approaches in the literature. The approaches considered in the literature are instance-based regression, KNN [18], locally weighted regression, LOESS [19], rule-based regression, RULE [8], projection pursuit regression, PPR [13], partitioning algorithms that induce decision trees, CART [10], DART [11] and multivariate adaptive regression splines, MARS [12]. Properties of RPFP and the other seven approaches are summarized in Table 1. It can be seen from Table 1 that all partitioning methods (RULE, CART, DART, MARS) except RPFP have similar properties. A detailed overview and comparison of these regression techniques is given in [1].

The shortfall of RPFP is it does not perform well for smooth and simple mathematical functions, since it applies heuristic approach for nonlinear and non-parametric prediction. The reason for this shartfall for RPFP is it works on feature dimensions seperately and it applies a partitioning on tha data that may prevent it to catch the smooth global properties. Even though RPFP apllies some mathematical calculations on the local data on seperate feature dimensions, it is a heuristic approach when the global data is under consideration. For example for a globally linear function least squares

Table 1. Properties of Regression Algorithms. The (√) 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	LOESS	PPR	RULE	CART	DART	MARS
Adaptive	√		√		√	√	√	√
Continuous	√	√	√	√			√	√
Categ.&Num.F.	√	√			√	√	√	√
Dimensionality	√			√				
Incremental	√	√	√					
Interactions	√	√	√	√	√	√	√	√
Interpretable				√	√	√	√	√
Irrelevant F.	√				√	√	√	√
Local	√	√	√		√	√	√	√
Memory Cost				√	√	√	√	√
Missing Val.	√							
Noise	RPFP-N							
Normalization	√				√	√	√	√
Regions Overlap	√	√	√	√			√	√
Partitioning	√				√	√	√	√
Testing Cost				√	√	√	√	√
Train Cost	√	√	√					

linear regression and neural network models are better choises. Multivariate adaptive regression splines, is also an appropriate method for simple linear, quadratic and cubic mathematical functions. Projection pursuit regression and a similar approach recently developed, Locally Weighted Projection Regression (LWPR) [20] are shown to be succesful on mathematical functions. LWPR puts an additional property to PPR such that, projections are done on local instances instead of the global instance space. It is additionally adaptive, local and incremental when compared to PPR.

Another shartfall of RPFP method is that the projections are done on feature dimensions, and the region left after partitioning is a smooth hyper-rectangle. This shorfall can be compared to Projection Pursuit Regression the Locally Weighted Projection Regression, projections are done to different projection directions, which is appropriate for oblique data. However, this shortfall is not so severe as regression tree induction algorithms - they also produce such hyper-rectangles, since local weight of RPFP apllied to instances make closer instances more dominant on predictions and the boundaries of the rectangles are not dominant on predictions.

Dimentionality reduction property of Locally Weighted Linear Regresson and Projection Pursuit Regression are also absent in RPFP, as this property is useful in decreasing the computation time in the testing phase. However these methods works on data sets having only continuous features, and handling missing values is a disadvantage when compared to RPFP. Noise is an other problem for such methods developed for mathematical functional prediction.

6. Empirical Evaluations

In this section empirical results of RPFP and other important regression methods are presented. Though the main purpose is to measure the performance of RPFP and to compare it with contemporary regression algorithms, another goal is to present a comparison of those methods on a large number of real domains, since it is difficult to find such comparisons in the literature.

The algorithms are carefully selected according to certain criteria. All of them can handle high dimensional domains and accept both categorical and continuous input features. Because it accepts only continuous features, we did not include LOESS for example, even though it is a lazy approach like RPFP. The algorithms use different approaches, such as regression tree induction, instance-based learning and rule-based learning. Most of them have been recently developed

and outperform earlier algorithms using the same approach. Finally, all of them are publically available.

The algorithms chosen are KNN (instance-based), which is the most significant one since it belongs to the same category as RPFP, RULE (rule-based learning), DART (regression tree induction), and MARS (spline-based, partitioning regression).

In the next section, we describe the evaluation methodology which is commonly used to measure accuracy of regression methods. Later, algorithms and real datasets are described and empirical results are presented, including accuracy performance, robustness of the algorithms to irrelevant features, missing values and noise.

6.1. Performance Measure

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) [8, 9]. It is the mean of absolute error found for all test examples.

$$MAD = \frac{\sum_{i=1}^{T} |y_i - \hat{y}_i|}{T} \tag{25}$$

where T is the number of test instances.

However in order to get similar performance values for all datasets a modified version of MAD, namely relative error (RE), is used in the experiments. Relative error is the true mean absolute distance normalized by the mean absolute distance from the median.

$$RE = \frac{MAD}{\frac{1}{T}\sum_{i=1}^{T} |y_i - median(y)|} \tag{26}$$

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

6.2. Algorithms Used in Comparisons

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

6.2.1. RPFP. K is the parameter of RPFP[1] that defines the minimum number of instances allowed for a region determined by the partitioning algorithm; it is set to 10. RPFP-N is also used for artificial noisy domains extracted from real datasets, to measure robustness to noise.

6.2.2. KNN. The weighted KNN[2] algorithm described by Mitchell [18] is used. It performs better than simple KNN which 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. Normalization of test and train input feature values is applied in order to obtain values in the range between 0 and 1. For matching nominal values the difference is measured as 0, and for different nominal values on a single dimension 1 is assigned.

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

6.2.3. RULE. The latest rule-based regression implementation, written by Weiss and Indurkhya [9] is used in our experiments. The program is available in the data mining software kit (DMSK), attached to Weiss and Indurkhya's book.

6.2.4. DART. DART is the latest regression tree induction program developed by Friedman [11]. It avoids the limitations of disjoint partitioning, used for other tree-based regression methods, by producing overlapping regions with increased training cost. In the experiments, the parameter *maximum dimension* (features) is increased from 100 to 200, in order to enable experiments for irrelevant features.

6.2.5. MARS. The latest shared version of MARS, mars3.6, is used in experiments, which is developed by Friedman [12]. The highest possible interaction level is enabled and linear spline approximation is set, which generally produces better results than cubic splines for the real datasets used in the experiments.

6.3. Real Datasets

It is possible to obtain a large number of real world datasets for classification, however this is not easy for regression. For this reason, datasets used in the experiments were collected mainly from three sources [6, 21, 22].[3] Properties of the datasets are shown in Table 2. In order to save space, they are coded with two letters (e.g., AB for Abalone).

Table 2. Datasets.

Dataset	Code	Instances	Features ($C + N$)	Missing values	Target feature
Abalone [6]	AB	4177	8 (7 + 1)	None	Rings
Airport [21]	AI	135	4 (4 + 0)	None	Tons of mail
Auto [6]	AU	398	7 (6 + 1)	6	Gas consumption
Baseball [21]	BA	337	16 (16 + 0)	None	Salary
Buying [22]	BU	100	39 (39 + 0)	27	Husbands buy video
College [22]	CL	236	20 (20 + 0)	381	Competitiveness
Country [22]	CO	122	20 (20 + 0)	34	Population
Electric [22]	EL	240	12 (10 + 2)	58	Serum 58
Fat [21]	FA	252	17 (17 + 0)	None	Body height
Fishcatch [21]	FI	164	7 (6 + 1)	87	Fish weight
Flare2 [6]	FL	1066	10 (0 + 10)	None	Flare production
Fruitfly [21]	FR	125	4 (3 + 1)	None	Sleep time
Gss2 [22]	GS	1500	43 (43 + 0)	2918	Income in 1991
Homerun [21]	HO	163	19 (19 + 0)	None	Run race score
Normtemp [21]	NO	130	2 (2 + 0)	None	Heart rate
Northridge [23]	NR	2929	10 (10 + 0)	None	Earthquake magnit.
Plastic [22]	PL	1650	2 (2 + 0)	None	Pressure
Poverty [21]	PO	97	6 (5 + 1)	6	Death rate
Read [22]	RE	681	25 (24 + 1)	1097	Reader satisfaction
Schools [22]	SC	62	19 (19 + 0)	1	Reading score
Servo [6]	SE	167	4 (0 + 4)	None	Rise time of a servo
Stock [22]	ST	950	9 (9 + 0)	None	Stock price
Television [21]	TE	40	4 (4 + 0)	None	People per TV
Usnews [21]	UN	1269	31 (31 + 0)	7624	Rate of Ph.D.'s
Village [24]	VL	766	32 (29 + 3)	3986	Number of Sheep

C: Continuous, N: Nominal.

6.4. Accuracy

The relative errors of algorithms on 25 real data sets are shown in Table 3. The best results, smallest relative errors, are shown in bold type. RPFP achieves best results in 9 of these datasets. MARS and DART achieve best results in 7 and 4 of these datasets respectively.

As can be seen, when the average relative errors of the algorithms on real data sets are compared, RPFP outperforms the other algorithms and achieves the smallest mean relative error (0.778). Another important result extracted from Table 3 is the distribution of relative errors for different datasets. We have computed the variance of errors for each algorithm on all datasets. These variance values show that the performance of RPFP (0.109) is not affected much by different domains. This shows the domain independence characteristic of RPFP, important for large databases today, where data from many domains are collected together. The diversity of domains in databases is one of the reasons that increase the need for automatic knowledge discovery tools and inductive learning algorithms.

On the other hand, on some data sets, we see that RPFP do not produce better results as some other algorithms. Some reasons that may prevent RPFP' success may be listed as follows: Locality—on some data sets global algorithms may produce better results, if there is a relation between instances on the global space and there interactions on the data globally. Bias-variance trade-off: RPFP is tuned for regression problems and it may not produce better results if the data is biased, in other words variance error or error coming from noise and over-fitting is important, especially for data appropriate for classification algorithms and

Table 3. Relative errors of algorithms. Best results are shown in bold type.

Dataset	RPFP	KNN	RULE	MARS	DART	StdDev
AB	0.675	**0.661**	0.899	0.683	0.678	0.101
AI	**0.473**	0.612	0.744	0.720	0.546	0.115
AU	0.334	**0.321**	0.451	0.333	**0.321**	**0.056**
BA	0.664	**0.441**	0.668	0.497	0.525	0.102
BU	**0.792**	0.951	0.944	0.883	0.858	**0.066**
CL	0.692	0.764	0.290	1.854	**0.261**	0.646
CO	**1.301**	1.642	6.307	5.110	1.845	2.300
EL	**1.009**	1.194	1.528	1.066	1.095	0.207
FA	0.667	0.785	0.820	**0.305**	0.638	0.204
FI	0.243	0.582	0.258	**0.190**	0.284	0.155
FL	**1.218**	2.307	1.792	1.556	1.695	0.397
FR	1.056	1.201	1.558	**1.012**	1.077	0.222
GS	0.566	0.654	**0.218**	0.359	0.342	0.177
HO	0.868	0.907	0.890	**0.769**	0.986	0.078
NO	**0.962**	1.232	1.250	1.012	1.112	0.128
NR	0.947	1.034	1.217	0.928	**0.873**	0.134
PL	0.415	0.475	0.477	**0.404**	0.432	**0.034**
PO	0.703	0.796	0.916	1.251	**0.677**	0.233
RE	**1.008**	1.062	1.352	1.045	1.194	0.142
SC	0.319	0.388	0.341	**0.223**	0.350	**0.062**
SE	0.527	0.619	**0.229**	0.441	0.337	0.153
ST	0.729	**0.599**	0.906	0.781	0.754	0.110
TE	**1.659**	1.895	4.195	7.203	2.690	2.281
UN	0.666	0.480	0.550	**0.412**	0.444	0.101
VL	**0.970**	1.017	1.267	1.138	1.131	0.116
Mean	**0.778**	0.904	1.203	1.207	0.846	
Variance	**0.109**	0.231	1.775	2.482	0.327	

pruning methods. In the light of above discussion; DART (improved version of CART—classification tree algorithm) and RULE (a partitioning algorithm that produce hyper-rectangles on global space and make approximations on the whole partitions) algorithms, as being global partitioning algorithms, and it is known that their classification counter parts of these algorithms produce better results on the classification data, achieve better performance on CL data (CL data has ordered 8 integer target values). An other shortcoming of RPFP on CL data may be because it makes local linear approximation on local regions and target values may not always enable linear approximations well, especially if they have categorical nature as CL target values.

In fact it is difficult to measure the error components (error coming from properties of the algorithms and variance and bias components of the error) of algorithms on real data sets is difficult, we considered to artificially modify the real data sets to test the behaviour of algorithms against noise, irrelevant features and missing feature values. The final column of Table 3 shows the standard deviation of the results for each dataset, and it is used in the following sections. It is used to determine a small number of datasets, to be used for further comparisons of algorithms for noise, irrelevant features and missing values. We have determined a subset of datasets that have similar results for the comparison of algorithms for increasing missing values, irrelevant features and noise. Selected datasets, that have small standard deviations, are indicated in the last column with bold type. In only one of these selected datasets does RPFP performs best.

6.5. Robustness to Irrelevant Features

The performance of five algorithms on the selected datasets (AU, BU, PL and SC), when new irrelevant features are added, is shown in Fig. 8. From the graphs it is seen that the performance of RPFP is not affected by irrelevant features except in PL dataset. RULE and MARS are also robust to irrelevant features. Note that, in only one of these data sets (BU) does RPFP perform best initially. It is affected by irrelevant features in PL probably because it is a low dimensional data set, initially having only two input features. Most advantages of RPFP are generally manifested in higher dimensions.

These graphs show that RPFP is not affected much by irrelevant features. This is the major drawback of KNN, the other lazy algorithm in these comparisons, and this is apparent in the graphs. This robustness of RPFP is achieved by the local weight assigned to each feature and by making computations on each feature separately.

A comparison of algorithms on all datasets where 30 irrelevant features are added to each of them is shown in Table 4. RPFP outperforms other algorithms for the robustness to irrelevant features according to this table.

6.6. Robustness to Missing Values

With current relational databases, the issue of missing values is a common problem for most domains. RPFP handles missing values naturally by simply ignoring

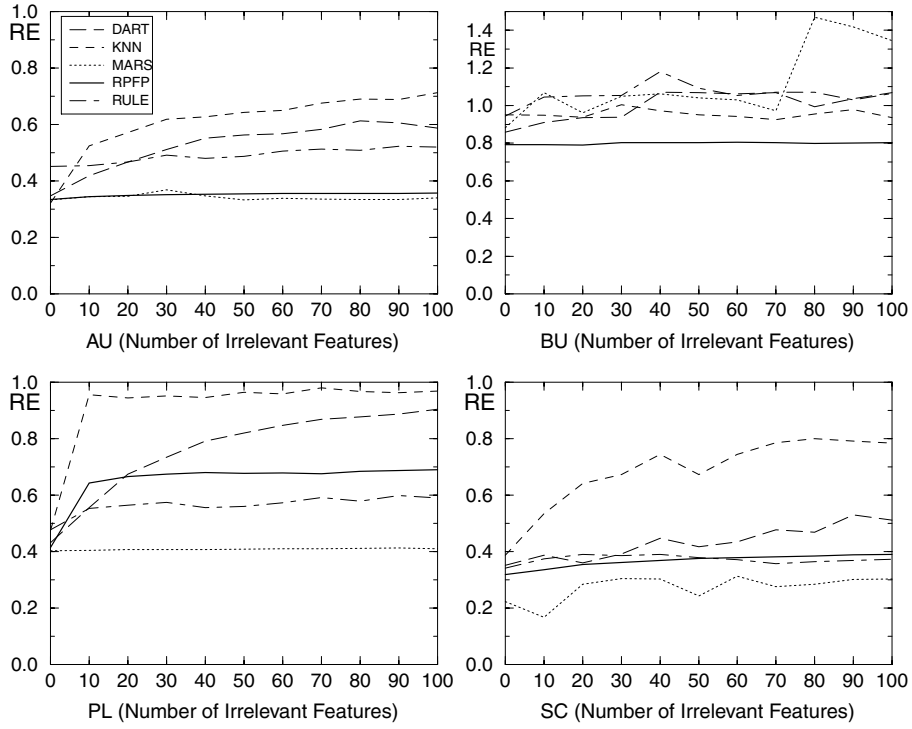

Figure 8. Relative errors of algorithms with increasing irrelevant features.

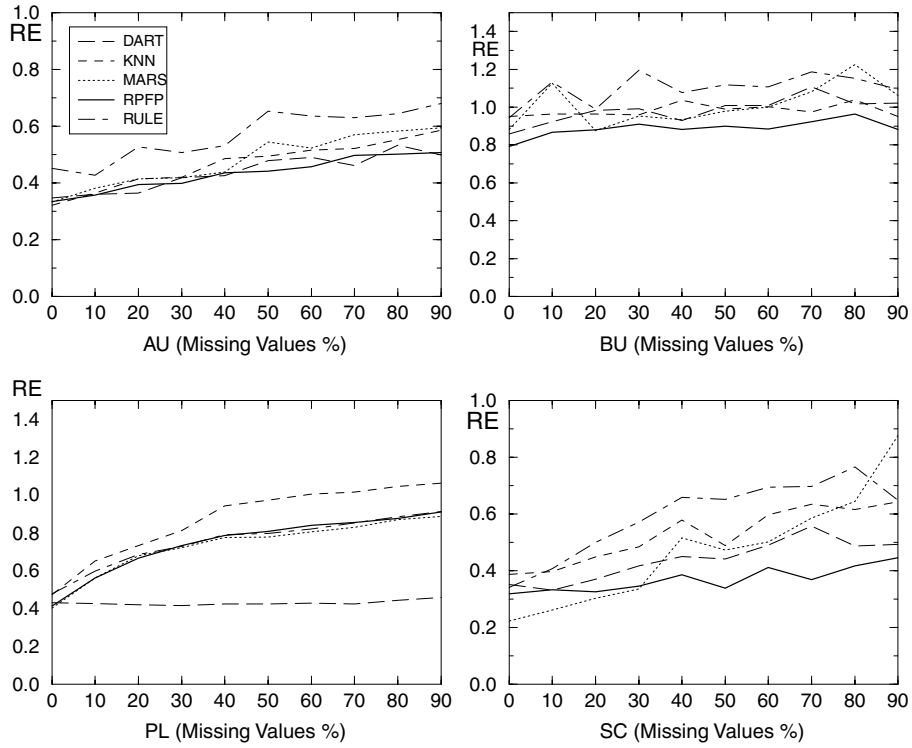

Figure 9. Relative errors of algorithms with increasing missing values.

Table 4. Relative errors of algorithms, where 30 irrelevant features are added to real datasets. Where no result is available due to a singular variance/covariance matrix, it is shown with (*). Best results are shown in bold type.

Dataset	RPFP	KNN	RULE	DART	MARS
AB	0.704	0.906	0.899	*	**0.682**
AI	**0.500**	1.539	0.744	0.658	0.682
AU	**0.351**	0.618	0.451	0.511	0.369
BA	0.670	0.723	0.668	0.641	**0.573**
BU	**0.802**	1.005	0.944	0.938	1.049
CL	0.716	1.320	**0.290**	0.306	2.195
CO	**1.330**	3.027	6.307	1.662	4.126
EL	**1.018**	1.076	1.528	1.236	1.134
FA	0.698	1.058	0.820	0.877	**0.249**
FI	0.295	0.985	0.258	0.350	**0.208**
FL	**1.038**	1.537	1.792	1.490	1.629
FR	**0.959**	1.075	1.558	1.430	1.777
GS	0.568	0.893	**0.218**	0.573	0.404
HO	0.876	0.974	0.890	1.165	**0.847**
NO	**1.024**	1.071	1.250	1.157	1.370
NR	0.979	1.149	1.217	*	**0.916**
PL	0.674	0.952	0.477	0.734	**0.407**
PO	**0.775**	0.934	0.916	1.013	1.005
RE	**1.033**	1.060	1.352	1.311	1.042
SC	0.362	0.673	0.341	0.391	**0.305**
SE	0.589	1.021	**0.229**	0.650	0.798
ST	0.782	1.151	0.906	**0.756**	0.818
TE	**1.617**	2.455	4.195	2.709	5.614
UN	0.671	0.856	0.550	0.906	**0.394**
VL	**0.972**	1.111	1.267	1.307	1.257
Mean	**0.800**	1.167	1.203	0.990	1.194
Variance	**0.090**	0.278	1.775	0.287	1.517

Table 5. Relative errors of algorithms, where 20% of values in the datasets are removed. Where no result is available due to a singular variance/covariance matrix, it is shown with (*). Best results are shown in bold type.

Dataset	RPFP	KNN	RULE	DART	MARS
AB	0.739	0.750	0.962	**0.688**	0.748
AI	**0.532**	0.726	0.676	0.546	0.798
AU	0.393	0.414	0.526	**0.363**	0.414
BA	0.817	**0.560**	0.783	0.565	0.709
BU	0.881	0.964	0.989	0.983	**0.877**
CL	0.796	0.942	**0.400**	0.435	0.801
CO	**1.439**	1.856	3.698	2.377	3.733
EL	**1.029**	1.097	1.537	1.191	1.074
FA	0.767	0.849	0.949	0.735	**0.731**
FI	**0.273**	0.584	0.336	0.320	0.348
FL	**1.377**	1.851	1.751	1.421	1.557
FR	1.033	1.711	1.557	1.347	**1.012**
GS	0.702	0.743	**0.497**	0.536	0.595
HO	0.889	0.911	1.040	0.974	**0.836**
NO	**0.986**	1.229	1.363	1.222	0.989
NR	**0.940**	1.072	1.272	*	0.972
PL	0.668	0.733	0.686	**0.420**	0.679
PO	**0.682**	0.976	1.189	0.792	1.026
RE	**1.007**	1.059	1.364	1.229	1.048
SC	0.327	0.449	0.500	0.370	**0.303**
SE	0.938	0.921	0.849	**0.495**	0.733
ST	0.777	0.744	0.904	**0.707**	0.930
TE	**1.810**	4.398	3.645	2.512	16.503
UN	0.669	0.559	0.620	0.844	**0.497**
VL	**1.014**	1.056	1.410	*	1.090
Mean	**0.859**	1.086	1.180	0.916	1.560
Variance	**0.115**	0.624	0.714	0.344	10.10

them, and using all other values provided. A comparison of RPFP and other algorithms on selected data sets for increasing percentages of missing values is shown in Fig. 9. As the values are removed from the data, information loss and decrease in performance are evident. However, the decrease in performance is smaller in RPFP than in other algorithms, where the missing values are filled with means or most frequent nominal value. The error rate of RPFP becomes relatively minimal in all selected datasets, when the percentage of missing values reaches 90%, except for the low dimensional data set PL. According to these results, DART also performs well in terms of robustness to missing values.

A comparison of algorithms on all datasets, where 20% of the values of real datasets are removed, is shown in Table 5. According to these results RPFP outperforms other algorithms in terms of robustness to missing values.

6.7. Robustness to Noise

It is apparent from the graphs in Fig. 10 that RPFP-N outperforms the other algorithms for most of the selected datasets. An interesting result is that RPFP also achieves better than other algorithms in most datasets. However, all algorithms except RPFP-N reach

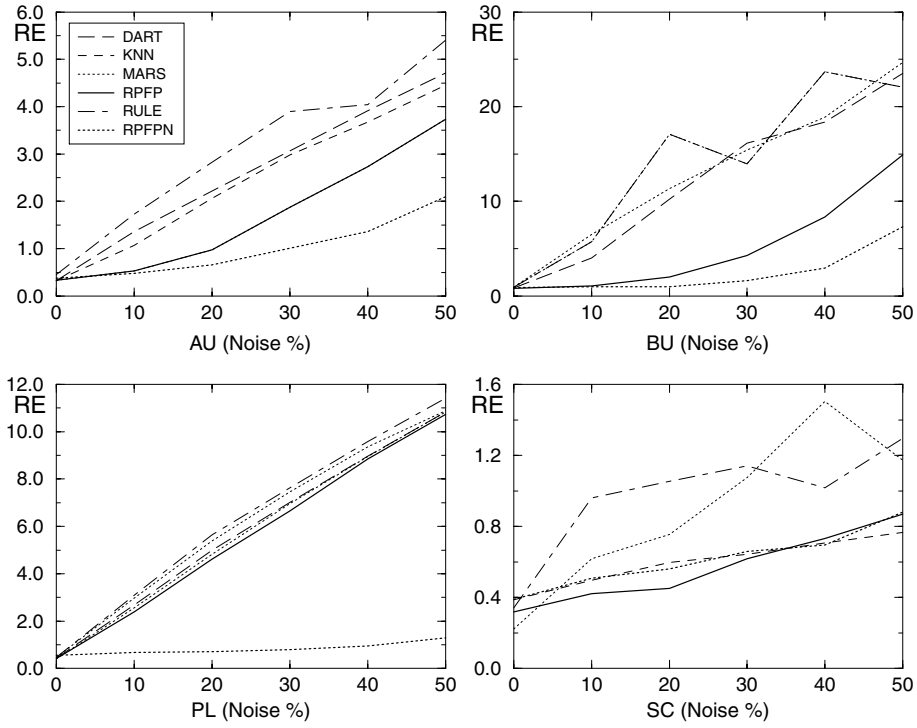

Figure 10. Relative errors of algorithms with increasing target noise.

unacceptable error rates with a small increase in the ratio of noise.

The main disadvantage of complex models is large variance errors because of overfitting. This difficulty becomes more apparent when there are noisy cases in the data. RPFP is generally more robust to noise when compared to other methods even though it produces complex models on specific local regions. One reason for this is that noisy instances on feature dimensions cause small local weight for that feature. Another reason may be the implicit boosting effect of RPFP as described in Section 4.4 before which causes small variance error.

6.8. *Interactions*

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

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

The first experiment is done with a simple artificial dataset having two interacting features and 100 instances formed as shown in Fig. 11. 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 RPFP-A, whose relative error is 1.35.

Another experiment is conducted with real data sets, and the results are shown in Table 6. The results show that RPFP significantly outperforms RPFP-A, which demonstrates the ability of RPFP to handle interactions.

6.9. *Computation Times*

Since the computation times of lazy and eager approaches differ significantly for training and predic-

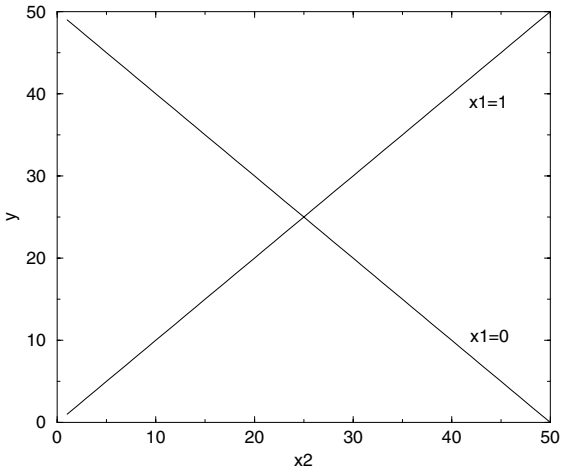

Figure 11. Artificial data set. $x1$ and $x2$ are input features.

tion phases, training times of eager approaches and prediction or test times of lazy approaches are given in Table 7. Generally test times of eager approaches and training times of lazy approaches are close to zero. The time durations are measured on a Pentium450 personal computer running Linux operating system.

The results justify the theoretical considerations in determining the computational complexity of RPFP such that it is proportional to the linear prediction complexity of KNN. On the average, prediction time of RPFP is 2.5 times higher than of KNN. This is more apparent for largest datasets (AB, GS, NR). In general computation performances of algorithms differ for different datasets.

Table 6. Comparison of RPFP with its additive version RPFP-A. Best results are shown in bold type.

D	RPFP	RPFP-A	D	RPFP	RPFP-A	D	RPFP	RPFP-A
AB	**0.675**	0.815	FA	**0.667**	0.855	PL	**0.415**	0.819
AI	**0.473**	0.500	FI	**0.243**	0.334	PO	**0.703**	0.783
AU	**0.334**	0.430	FL	**1.218**	1.487	RE	1.008	**1.000**
BA	**0.664**	0.752	FR	1.056	**1.041**	SC	**0.319**	0.337
BU	**0.792**	0.896	GS	**0.566**	0.667	SE	**0.527**	0.944
CL	**0.692**	0.773	HO	**0.868**	0.939	ST	**0.729**	0.992
CO	**1.301**	1.354	NO	0.962	**0.958**	TE	1.659	**1.629**
EL	**1.009**	1.019	NR	**0.947**	0.956	UN	**0.666**	0.718
VL	**0.970**	0.988						

Table 7. Time durations of algorithms for real data sets in milliseconds.

Dataset	RPFP Test	KNN Test	RULE Train	DART Train	MARS Train	RPFP/KNNR Ratio
AB	40081.2	17217.1	6593.3	458503.0	7629.1	2.3
AI	7.5	3.0	248.4	57.8	153.5	2.5
AU	124.1	41.8	407.4	1772.2	573.9	3.0
BA	261.9	50.1	429.8	3022.1	912.4	5.2
BU	51.6	49.2	284.9	667.7	738.6	1.0
CL	150.9	35.6	464.0	708.6	1039.9	4.2
CO	32.0	31.7	396.5	459.4	484.8	1.0
EL	141.1	19.5	389.1	933.2	385.6	7.2
FA	167.0	30.3	403.9	1654.4	755.8	5.5
FI	18.1	161.3	278.5	200.5	226.7	0.1
FL	1198.8	775.9	408.7	901.4	543.8	1.5
FR	9.0	44.0	234.5	42.8	99.9	0.2
GS	14241.0	6435.2	1236.6	23845.8	8797.0	2.2
HO	92.8	140.7	266.3	835.8	616.2	0.7
NO	6.0	1.7	236.8	18.0	68.1	3.5
NR	24027.0	9346.7	7006.4	81984.0	4207.3	2.6
PL	1536.4	1415.5	503.3	9343.1	670.7	1.1
PO	6.1	2.0	250.3	41.1	121.6	3.1
RE	2717.0	674.6	625.2	35541.5	2260.1	4.0
SC	7.5	2.0	251.2	78.2	283.8	3.8
SE	6.9	4.0	221.6	78.2	109.0	1.7
ST	1173.8	759.8	845.6	16203.2	1839.2	1.5
TE	0.1	0.0	235.4	3.1	25.5	*
UN	6459.7	3858.9	4834.2	153959.0	7287.0	1.7
VL	2113.9	1229.5	1101.0	107661.0	3082.9	1.7
Mean						2.5

7. Conclusion

In this paper we have presented a new regression method called RPFP. It is an instance-based, nonparametric, nonlinear, context-sensitive, and local supervised learning method based on feature projections. Regression is one of the oldest techniques in the literature, for which many researchers from different disciplines have developed great deal of methods. We have selected appropriate eager and lazy methods in the literature in for comparisons, in terms of accuracy, robustness to irrelevent features, missing values and noise, and ability to handle interactions. We also compared the properties of RPFP with different approaches.

RPFP achieves high levels of accuracy especially when compared to the most common lazy approach,

KNN. Its performance is also very good when compared to important eager approaches from both machine learning and statistics. The feature projection based construction of the method enable it to handles missing values naturally, by leaving sparse fields empty, and on the other hand, RPFP handles interactions by partitioning the data as opposed to other feature projection based and additive approaches in the literature. Additional to its success on data sets having missing values RPFP algorithm presented in the paper can be modified in order to achieve high accuracies in very noisy domains, as illustrated with RPFP-N. The concept of local weight is employed for increasing the accuracy to deal with irrelevant features, establishing adaptivity and as a key element to direct partitioning. As a summary, RPFP is a new instance-based, nonlinear, context-sensitive, and local approach to regression problems; and many experiments show its performance for different domains. The main drawback of RPFP is the lack of interpretation and its high prediction time. When compared to KNN, RPFP is 2.5 times slower, as we experimented with data sets having different number of dimensions and sizes.

Further research is required to address the limitations of RPFP so that interpretation can be enabled by determining the relative importance of features, and interactions between them. Different partitioning strategies on selecting appropriate features at each partitioning step and on reduction of the instance space can be further developed for improving the efficiency of partitioning. 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 are also important issues to be addressed by further research. It is also possible to employ and test the effectiveness of RPFP for different problems such as time-series prediction.

Notes

1. Implementation of RPFP in *C* code is available from the authors upon request.
2. K is set to 10 for all experiments, and our implementation of KNN is available from the authors upon request.
3. All of the datasets are available from the authors upon request.

References

1. İ. Uysal and H.A. Güvenir, "An overview of regression techniques for knowledge discovery," *Knowledge Engineering Review*, Cambridge University Press, vol. 14, pp. 1–22, 1999.
2. D. Wettschereck, D.W. Aha, and T. Mohri, T. "A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms," *AI Review*, vol. 11, pp. 273–314, 1997.
3. H.A. Güvenir and İ. Sirin, "Classification by feature partitioning," *Machine Learning*, vol. 23, pp. 47–67, 1996.
4. H.A. Güvenir and H.G. Koç, "Concept representation with overlapping feature intervals," *Cybernetics and Systems: An International Journal*, vol. 29, pp. 263–282, 1998.
5. H.A. Güvenir, G. Demiroz, and H. İlter. "Learning differential diagnosis of erythemato squamous diseases using voting feature intervals," Artificial Intelligence in Medicine, vol. 13, pp. 147–165, 1998.
6. C. Blake, E. Keogh, and C.J. Merz, "UCI repository of machine learning databases," [http://www.ics.uci.edu/mlearn/MLRepository.html], University of California, Department of Information and Computer Science, Irvine, CA, 1998.
7. R.C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Machine Learning*, vol. 1, pp. 63–91, 1993.
8. S. Weiss and N. Indurkhya, "Rule-based machine learning methods for functional prediction," *Journal of Artificial Intelligence Research*, vol. 3, pp. 383–403, 1995.
9. S. Weiss and N. Indurkhya, *Predictive Data Mining: A Practical Guide*, Morgan Kaufmann: San Francisco, 1998.
10. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*, Wadsworth: Belmont, CA, 1984.
11. J.H. Friedman, "Local learning based on recursive covering," [ftp://stat.stan- ford.edu/pub/friedman/dart.ps.Z].
12. J.H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, pp. 1–141, 1991.
13. J.H. Friedman and W. Stuetzle, "Projection pursuit regression," *J. Amer. Statist. Assoc.*, vol. 76, pp. 817–823, 1981.
14. J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann: California, 1993.
15. J.H. Friedman, "On bias, variance, 0/1-loss and the curse of dimensionality," *Data Mining and Knowledge Discovery*, vol. 1, pp. 55–77, 1997.
16. L. Breiman, L. "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.
17. C.J. Matheus, P.K. Chan, and G. Piatetsky-Shapiro, "Systems for knowledge discovery in databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, pp. 903–913, 1993.
18. T.M. Mitchell, *Machine Learning*, McGraw Hill: New York, 1997.
19. C.G. Atkenson, A.W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, nos. 1/5, pp. 11–73, 1997.
20. S. Vijayakumar and S. Schaal, "LWPR: An $O(n)$ algorithm for incremental real time learning in high dimensional space," in *Proc. of Seventeenth International Conference on Machine Learning (ICML2000) Stanford*, California, 2000, pp. 1079–1086.
21. R.H. Lock and T. Arnold, "Datasets and stories: Introduction and guidelines," *Journal of Statistics Education* [Online], vol. 1, [http://www.amstat.org/publi cations/jse/v1n1/datasets.html], 1993.
22. SPSS, "Sample data sets," [ftp://ftp.spss.com/pub/spss/sample/datasets/], 1999.

23. D. Draper and G. Michailides, "UCLA statistics case studies," http://www.stat.ucla.edu/cases/northridge/index.phtml.
24. İ. Uysal and H.A. Güvenir, "Function approximation repository" http://funapp.cs.bilkent.edu.tr, 2000.
25. P. Domingos, "Occam's two razors, the sharp and the blunt," in *Proc. KDD'98*, 1998.
26. G. Webb and M. Kuzmycz, "Further experimental evidence against the utility of Occam's razor," *Journal of Artificial Intelligence Research*, vol. 4, pp. 397–417, 1996.

Ilhan Uysal is currently a Ph.D. candidate in the Department of Computer Engineering at Bilkent University, Ankara, Turkey. He received his M.S. degree from the the same department in 2000. His reserach interests include machine learning, knowledge discovery, data mining, function approximations and time-series prediction.

H. Altay Güvenir is currently a professor in the Department of Computer Engineering at Bilkent University, Ankara, Turkey. He received his Ph.D. in Computer Engineering and Science at Case Western Reserve University in 1987. He received his B.S. and M.S. degrees from the Istanbul Technical University in 1979 and 1981, respectively, all in electrical engineering. Dr. Güvenir's research interests include machine learning, knowledge discovery, data mining, function approximations, time-series prediction and cost sensitive classification. He has (co-)authored over 80 articles in journals, conference proceedings and edited books.