ELSEVIER

Knowledge-Based Systems 13 (2000) 207–214

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

Regression on feature projections

H. Altay Guvenir*, I. Uysal

[a]*Department of Computer Engineering, Bilkent University, 06533 Ankara, Turkey*

Received 23 August 1999; revised 21 March 2000; accepted 3 April 2000

Abstract

This paper describes a machine learning method, called Regression on Feature Projections (RFP), for predicting a real-valued target feature, given the values of multiple predictive features. In RFP training is based on simply storing the projections of the training instances on each feature separately. Prediction of the target value for a query point is obtained through two averaging procedures executed sequentially. The first averaging process is to find the individual predictions of features by using the K-Nearest Neighbor (KNN) algorithm. The second averaging process combines the predictions of all features. During the first averaging step, each feature is associated with a weight in order to determine the prediction ability of the feature at the local query point. The weights, found for each local query point, are used in the second prediction step and enforce the method to have an adaptive or context-sensitive nature. We have compared RFP with KNN and the rule based-regression algorithms. Results on real data sets show that RFP achieves better or comparable accuracy and is faster than both KNN and Rule-based regression algorithms.

Keywords: Regression; Function approximation; Feature projections

1. Introduction

Prediction has been the most common problem researched in machine learning and data mining. Here we describe a method for predicting a real-valued or continuous target feature, given the values of multiple predictive features. Predicting a continuous feature is generally known as *regression* among related fields such as machine learning, statistics, pattern recognition as well as knowledge discovery in databases (KDD) and data mining. Regression differs from classification in that the predicted target variable t in regression problems is *continuous*, whereas in classification t is strictly categorical. From this perspective, classification can be considered as a subcategory of regression.

There are two different approaches for regression in the literature: *Eager* and *lazy* learning. The term *eager* refers to the learning systems that construct models that represent knowledge using the training data. After training, they make predictions by using this model, which is a compact representation of the data. In *lazy* learning, on the other hand, all processing is delayed to prediction phase.

We describe a lazy learning method Regression by Feature Projections (RFP) to predict a real-valued target, where the instances are stored as their projections on each feature dimension. In the RFP method, we use the KNN algorithm on each individual feature dimension to find their own prediction, independent of the predictions of other features. Then we find the precision of those features at the local position of query instance. We define the precision as a local weight that brings an adaptive or context-sensitive nature to the method. By adaptive, we mean that the contribution of each feature changes according to the local position of the query instance. The final prediction is made by combining individual feature predictions and using their local weights.

The traditional approach for prediction of a continuous target is the classical linear least-squares regression [22]. The model constructed for regression in this traditional approach is a linear equation. By estimating the parameters of this equation with a computationally simple process on the training set, a model is constructed. However, the linearity assumption between input features and target introduces a large bias error for most domains. That is why most studies are directed to new nonlinear and, in most cases, non-parametric techniques for the regression problem. Among eager regression learning systems, CART [5], RETIS [16], M5 [21], DART/HYESS [10], and Stacked Regressions [6] induce regression trees, FORS [4] uses inductive logic programming for regression

* Corresponding author. Tel.: +90-312-290-1252, fax: +90-312-266-4126.

E-mail address: guvenir@bilkent.edu.tr (H. Altay Guvenir).

0950-7051/00/$ - see front matter
PII: S0950-7051(00)00060-5

and Rule [25] induces regression rules. On the other hand, projection pursuit regression [9], neural network models and MARS [8] produce mathematical models. Among lazy learning methods, locally weighted regression [2] produces local parametric functions according to the query instances, and KNN [1,18,20] algorithm is the most popular nonparametric instance-based approach for the lazy learning of a real-valued target, to which we have compared our nonparametric lazy learning algorithm based on feature projections. The KNN algorithm is also known as *kernel regression* in statistical pattern recognition literature.

RFP eliminates most of the problems of other eager or lazy methods, such as the curse of dimensionality, dealing with missing feature values, information loss because of disjoint partitioning of data, irrelevant features, computational complexity of test or training, missing local information at query locations and the need for normalization with a trade off not dealing with interactions, which is in fact not met with most real-world data sets.

The theoretical and empirical results show that, in prediction, the RFP method is much faster than its natural competitor KNN, and achieves a comparable accuracy. For most data mining or knowledge discovery applications, where very large databases are in concern, this is thought of a solution because of low computational complexity, and eliminating the problems listed above, some of which take too much time for large data.

In Sections 2 and 3, we review the KNN and Rule-based algorithms for regression, respectively. Section 4 gives a detailed description of the RFP algorithm. Section 5 is devoted to the theoretical and empirical evaluation of RFP and its comparisons with KNN and the Rule algorithms. Finally, in Section 6, a conclusion and future work are presented.

2. K-Nearest Neighbor algorithm

The main advantage of lazy learning approaches is that, they make use of local information in the instance space, as the prediction is made according to exact location of each individual query instance. Another advantage of the lazy methods is their short training time, because training involves only the storage of training examples. The most commonly used instance-based or lazy method for both classification and regression problems are the K-Nearest Neighbor (KNN) algorithm. In this section we will review the way it is used for regression.

The main assumption behind the KNN algorithm is that the closest instances to the query point have similar target values to the query. Hence, the KNN algorithm first finds the closest instances to the query point in the instance space according to a distance measure, then outputs the average of the target values of those instances as the prediction for that query instance. Generally, the Eucledian distance metric is used to measure the similarity between two points in the instance space. Therefore, the *similarity* between a query point q_i and a sample point s_j in an N-dimensional instance space is computed as

$$\mathrm{Sim}(q_i, s_j) = 1 - \frac{\sqrt{\sum_{f=1}^{N} \delta(q,s,f)^2}}{\sqrt{N}} \tag{1}$$

$$\delta(d,s,f) = \begin{cases} 1 & \text{if } q_{i,f} \text{or } s_{j,f} \text{ is unknown (missing)} \\ (q_{i,f} - s_{i,f}) & \text{if } f \text{ is linear} \\ 0 & \text{if } f \text{ is nominal and } q_{i,f} = s_{j,f} \\ 1 & \text{if } f \text{ is nominal and } q_{i,f} \neq s_{j,f} \end{cases} \tag{2}$$

where $q_{i,f}$ and $s_{j,f}$ refer to the value of feature f for instances q_i and s_j, respectively. Feature values are assumed to be in the same range, e.g. [0,1], for all features. This is achieved by a simple normalization preprocessing of both training and query instances. If $q_i = s_j$ in Eq. (1), then $\mathrm{Sim}(q_i, s_j) = 1$. The lowest value for $\mathrm{Sim}(q_i,s_j)$ is 0.

The target value of the query point q_i is predicted as the similarity proportional average of the target values $t(s_j)$ of the nearest k neighbors of q_i among all the training instances

$$\hat{t}(q_i) = \frac{\sum_{i=1}^{k} t(s_j) \cdot \mathrm{Sim}(q_i, s_j)}{\sum_{i=1}^{k} \mathrm{Sim}(q_i, s_j)} \tag{3}$$

Therefore, the KNN algorithm assigns larger weights to the closer neighbors. As the prediction of the target value of a query instance requires to measure its distance to all training instances, which might be a very huge set, the prediction in KNN is very costly.

3. Rule-based regression

Inducing rules from a given training set is a well-studied topic in machine learning. Weiss and Indurkhya employed rule induction for regression problem and reported significant results [25]. In this section, we will first review the rule-based classification algorithm [24], Swap-1, that learns decision rules in disjunctive normal form (DNF), and later on describe its adaptation for regression.

The main advantage of inducing rules in DNF is its explanatory capability. It is comparable to decision trees as they can be converted into DNF models as well. Unlike decision trees, DNF rules need not be mutually exclusive. In decision trees, for each instance, there is exactly one rule, a path from root to a leaf that is satisfied. Because of this restriction, decision tree induction algorithms may not produce compact models. On the other hand, as the regions for rules need not be disjoint, several rules may be satisfied for a single query instance. The common solution to this

1. Generate a set of Pseudo-classes using the P-Class algorithm
2. Generate a covering rule-set for the transformed classification problem using a rule induction method such as Swap-1
3. Initialize the current rule set to be the covering rule set and save it.
4. If the current rule set can be pruned, iteratively do the following:
   4.1. Prune the current rule set.
   4.2. Optimize the pruned rule set and save it.
   4.3. Make this pruned rule set the new current rule set.
5. Use test instances or cross-validation to pick the best of the saved rule sets.

Fig. 1. Overview of learning regression rules.

problem is to assign priorities or ordering to the rules according to their extraction order. The first rule, according to this ordering that satisfies the query instance, determine the class of a query. Such ordered rule-sets have been referred to as *decision lists*.

While constructing a rule, the Swap-1 algorithm searches all the conjunctive components it has already formed, and swaps them with all possible components it will build. This search also involves deletion of some components from the rule. If no improvement is established from these swaps and deletions, then the best component is added to the rule. In order to find the best component to be added, the predictive value of a component, as the percentage of correct decisions, is evaluated. If the predictive values of the candidates are equal, maximum instance coverage is used as the second criterion. These swappings and additions end when the rule reaches 100% prediction accuracy.

After forming a new rule for the model, all instances that the rule covers are removed from the training instance set, and remaining instances are considered for the following steps. When a class is covered, the remaining classes are considered, in turn. This process iterates until the instance set becomes empty, that is all instances are covered. After the formation of the rule set, if removal of any rule does not change the performance on training set, such rules are removed from the model. Further, in order to reach an optimum rule set, an optimization procedure is used [24].

The rule induction algorithms for classification, such as Swap-1, can also be applied to the regression problems. As these algorithms are designed for the prediction of nominal attributes, by a preprocessing procedure, the numeric attribute in regression to be predicted is transformed to a nominal one. Weiss and Indurkhya used the P-class algorithm for this transformation [25]. This transformation is in fact a one-dimensional clustering of training instances on response variable t, in order to form classes. The purpose is to make t values within one class most similar, and across classes most dissimilar. The assignment of these values to classes is carried out in a way that, the distance between each t_i and its class mean must be minimum.

The P-Class algorithm sorts all the target values of the training samples first. Then, it assigns approximately equal number of contiguous target values to each class. Finally it moves a target value to one of the neighboring class if it reduces its distance to the mean of that class.

This procedure is a variation of *K-MEANS* clustering algorithm [7,17]. Given the number of clusters initially, on randomly decomposed clusters, the *K-MEANS* algorithm swaps the instances between the clusters if it increases a clustering measure or criterion that employs inter and intra-cluster distances. Given the number of classes, P-Class is a quick and precise procedure. However, no idea is stated in the literature about an efficient way to determine the number of classes.

After the formation of (pseudo) classes and application of a rule induction algorithm to these classes such as Swap-1, in order to produce optimum set of regression rules, a pruning and an optimization procedure can be applied to these rules [24,25]. The overview of the overall procedure for induction of regression rules is shown in Fig. 1.

The naive way to predict the response for a query instance is to assign the average of responses. The average may be a median or mean of that class. However different approaches also can be considered by applying a parametric or non-parametric model for that specific class. For example, the nearest-neighbor approach is used for this purpose, and significant improvements of this combination against the naive approach have been reported [25].

4. Regression on feature projections

In this section we introduce a lazy regression method based-on feature projections, called Regression on Feature Projections (RFP). The training samples are stored as their projections on every feature. Given a query instance, the RFP algorithm first makes a separate approximation of the target value for each feature, independently of the other features. This approximation is made by using the nearest instances to the query point on that feature dimension. The nearest instances may differ at each feature dimension. The final prediction is computed by the weighted combination of predictions made by each feature. The performance results obtained for classification algorithms based on feature projections [13,14] encouraged us to develop RFP. The next subsections describe the training and prediction phases

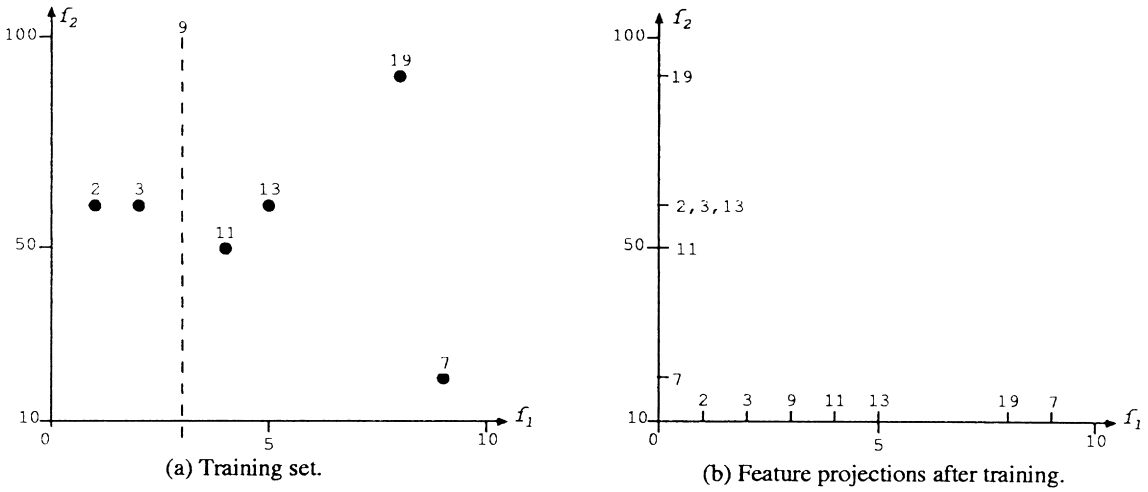

Fig. 2. Training in RFP.

of the RFP algorithm, along with the averaging and local weighting on feature projections.

4.1. Training

Training in RFP involves simply storing the training set as projections to the features. This is carried out by associating a copy of the target value with each projection, then sorting the instances for each feature dimension according to their feature values. If there are missing values of features, the training instance is not stored only for these features.

Training and prediction in RFP will be illustrated through an example. Let our example domain consist of two features, f_1 and f_2. Our training set contains seven samples; {1,60,2}, {2,60,3}, {4,50,11}, {5,60,13}, {8,90,19}, {9,20,7), {3,?,9}. Here the first two elements represent the values of f_1 and f_2, respectively, and the last element represents the target value. The value of f_2 in the last sample instance is unknown. The training set is shown in Fig. 2a. After training with this set, instances are stored by their feature projections as in Fig. 2b.

4.2. Prediction

In order to predict the target value of a query instance q_i, the RFP algorithm first projects the query instance on each feature. Then, for each feature f a prediction $\hat{t}(f, q_{i,f})$ for the target value $t(q_i)$ is made using only the value of $q_{i,f}$ and its nearest K neighboring training instances on the projections on the feature f. Here, $\hat{t}(f, q_{i,f})$ stands for the expected value of the target if $f = q_{i,f}$, independent of the other features. As the instances are sorted according to feature values in the training phase, the nearest neighbors can be found using a

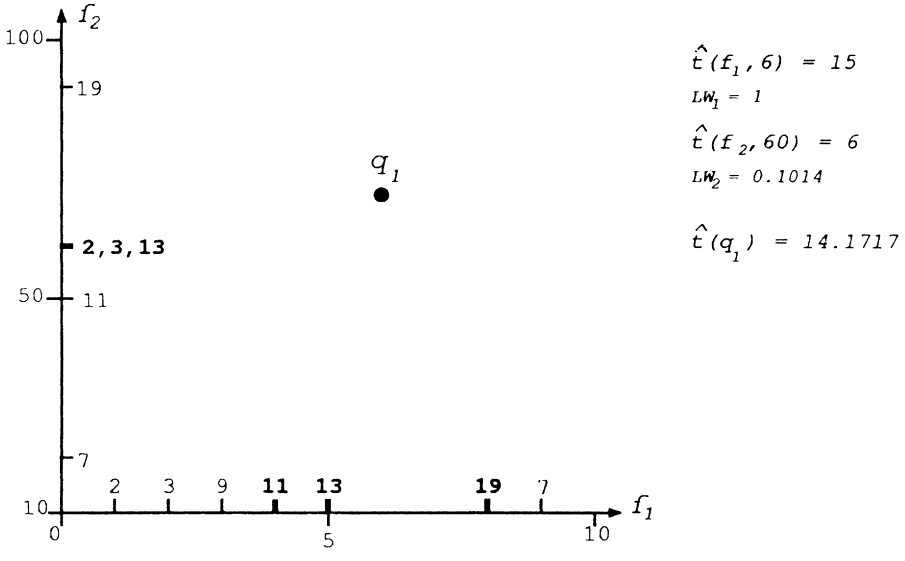

Fig. 3. Prediction in RFP.

binary search. If the feature f is categorical, then the average of the target values of these K nearest neighbors on f is used as the prediction on that feature. For a continuous feature, the linear least-squares approximation, given in Eq. (4), is used to make the prediction, $\hat{t}(f, q_{j,f})$. The linear least-squares algorithm finds a linear equation that minimizes the sum of the squared errors of the training instances [19]. Formally, the linear equation is in the following form:

$$\hat{t}(f, q_{i,f}) = \beta_0 + \beta_1 \cdot q_{i,f}. \tag{4}$$

Here, β_0 and β_1 are the coefficients of the linear equation. The error of this linear equation, formed for q_i on feature f, is computed as

$$\text{Error}(i,f) = \sum_{j=1}^{K} [t(s_j) - \hat{t}(f, s_{j,f})]^2 \tag{5}$$

where K is the number of neighbors considered, s_j is the jth nearest instance to q_i on f, and $t(s_j)$ is actual target value of the training sample s_j.

After constructing a linear equation, using the linear least-squares algorithm, the prediction for a particular feature is carried out by simply substituting the feature value of the query instance to this equation. If all K nearest training instances are found at the same feature value as $q_{i,f}$ then the linear least-squares approach will fail to make a plausible prediction, as, in that case, the slope β_1 will be infinite. In such cases simple averaging, as in categorical features, is used.

Continuing on the previous example, let us assume the query instance q_1 has the value 60 for f_1 and 7 for f_2, as shown in Fig. 3. Let us also assume the value of K is 3. RFP prediction algorithm locates the three nearest neighbors of the query point on each dimension, separately. These neighbors are shown in boldface in the figure. The training instance projections found on f_1 dimension, with target values 11, 13 and 19 form a linear regression equation as $\hat{t}(f_1, q_{1,1}) = 3 + 2 \cdot q_{1,1}$, substituting the value of $q_{1,1} = 6$, the prediction for f_1 is obtained as $\hat{t}(f_1, 6) = 15$. For the feature f_2, all three training instances are found at the value of 60. The RFP prediction algorithm does not construct a linear regression equation for f_2, as β_1 coefficient of such an equation would have to be ∞, and therefore the predicted value would be ∞ as well. In this case the RFP prediction algorithm finds the average of the target values of these three instances and outputs this value as the prediction for f_2. Therefore, $\hat{t}(f_2, 60) = \text{avg}(2, 3, 13) = 6$. After obtaining $\hat{t}(f, q_{i,f})$ for all features, the next step is to combine these individual predictions to make the final prediction.

4.3. Local weight

Some regions on a feature dimension may produce better approximations than others. In order to obtain the degree of fitting for a feature prediction, we employ a locality measure in the prediction algorithm. If the region where the query point falls in is smooth, we give a high weight to that feature in the final prediction. The locality measure allows the RFP prediction algorithm both eliminate the effect of irrelevant features, as well as the irrelevant regions of a feature dimension. It 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, the RFP prediction algorithm computes the distance-weighted mean of squared differences of the target values of the K nearest neighbors of q_i and their estimated values. We denote this measure with $V_{i,k,f}$ shown in Eq. (7). By subtracting it from the variance of the target values of all training instances, V_{all}, we find the explained variance for that region around q_i, and by normalizing it with the variance of training set we obtain a measure, called *prediction index* ($\text{PI}_{i,f}$) for the feature f (9). We use the square of $\text{PI}_{i,f}$ as the *local weight* ($\text{LW}_{i,f}$) of feature f (10)

$$V_{\text{all}} = \frac{\sum_{j=1}^{M} (t(s_j) - \bar{t})^2}{M} \tag{6}$$

$$V_{i,k,f} = \frac{\sum_{j=1}^{k} w_{f,i,j} (t(s_j) - \hat{t}(s_{j,f})^2}{\sum_{j=1}^{k} w_{f,i,j}} \tag{7}$$

where M is the number of training instances, $\hat{t}$ is the mean of target values in all training set, $\hat{t}(s_{j,f})$ is the estimation of the feature f for jth nearest training instance and $w_{f,i,j}$ is defined in Eq. (8)

$$w_{f,i,j} \frac{1}{(q_{i,f} - s_{j,f})^2 + \epsilon} \tag{8}$$

where ϵ is a positive real number close to zero,[1] used to avoid ∞ values for $w_{f,i,j}$.

$$\text{PI}_{i,f} = \frac{V_{\text{all}} - V_{i,k,f}}{V_{\text{all}}} \tag{9}$$

$$\text{LW}_{i,f} = \begin{cases} \text{PI}_{i,f}^2 & \text{if } \text{PI}_{i,f} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

For the example given above $M = 8$, and $V_{\text{all}} = 36.1875$. The local weight values for q_1 are computed as $\text{LW}_{1,1} = 1$ and $\text{LW}_{1,2} = 0.1014$.

4.4. Final prediction

The RFP prediction algorithm computes the final prediction for a query instance q_i as the weighted average of the predictions $\hat{t}(f, q_{i,f})$ found for each feature dimension, using the local weights. Fig. 4 summarizes the prediction phase as

[1] $\epsilon = 10^{-8}$ is used in the experiments.

Training:

Store training instances as their projections on each feature

Sort the projections of each feature

Compute V_{all}

end.

Prediction(q_i, K)**:**

Initialize $Sum_P = 0$, $Sum_{LW} = 0$;

For each feature f

If $q_{i,f}$ is known

Find K nearest neighbors of q_i on f

Compute feature prediction $\hat{t}(f, q_{i,f})$

Compute local weight $LW_{i,f}$

$Sum_P = Sum_P + LW_{i,f} \cdot \hat{t}(f, q_{i,f})$

$Sum_{LW} = Sum_{LW} + LW_f$

end of If

end of For

return (Sum_P / Sum_{LW})

end.

Fig. 4. Training and prediction algorithms of RFP.

well as the training. Formally

$$\hat{t}(q_i) = \frac{\sum_{f=1}^{N} \mathrm{LW}_{i,f} \cdot \hat{t}(f, q_{i,\ f})}{\sum_{f=1}^{N} \mathrm{LW}_{i,f}} \tag{11}$$

For the example given above, the final prediction for the target value of the query instance is $\hat{t}(q_i) = 14.1717$.

The RFP prediction algorithm does not make predictions for features whose vales are unknown in the query instance. The final prediction is obtained as the weighted average of the predictions by only features whose value is known in the query instance. Therefore, the RFP algorithm does not require any unnatural modification of the data to cope with missing feature values.

5. Theoretical and empirical evaluation

RFP inherits most properties of other lazy approaches. Two most important benefits of lazy learning approaches are their very small training complexity and the utilization of local information in the instance space. RFP benefits from these properties with an additional property of having small prediction time requirement. The method also deals with both types of input features, categorical and continuous, and handles irrelevant features in a very natural way. The single drawback of the method is its inability for dealing with interactions or relations among input features that lead to a decrease in prediction accuracy. However, we have seen that generally the real world datasets do not contain such interactions between features [13,15]. On the other hand, especially for large datasets with large number of input features and instances, the RFP method can be considered as a reliable solution, as it can reduce the effect of irrelevant features by assigning them lower weights. Important theoretical and empirical properties and results on real datasets obtained from two different sources [3,23] (also available at http://funapp.cs.bilkent.edu.tr [11]) are described in the following sections.

5.1. Complexity analysis of RFP

As the method uses a lazy approach, without a summarization with preprocessing phase, a memory space proportional to the whole training data is required. Given a data set with M training instances and N features the space requirement is proportional to $M{\cdot}N$. Again, for the training phase, the computational complexity of sorting projections on all features is $O(N{\cdot}M{\cdot}\log M)$. The computation of variance ($O(M)$) of target values for all training data is also computed in the training phase, and it does not change the above complexity, $O(N{\cdot}M{\cdot}\log M)$.

The complexity of finding the nearest instance on a sorted data for a feature, using binary search, is $O(\log M)$. Again on the sorted data, finding remaining $K-1$ instances

Table 1
Data sets used in the experiments

Dataset	Instances	Features (linear + nominal)	Missing values	Target feature
Abalone	4177	8 (7 + 1)	None	Rings
Auto-mpg	398	7 (6 + 1)	6	Gas consumption
Buying	100	39 (39 + 0)	27	Husbands buying video tape
Country	122	20 (20 + 0)	34	Population
Cpu	209	7 (1 + 6)	None	Relative CPU performance
Electric	240	12 (10 + 2)	58	Serum Cholesterol 58
Flare	1066	10 (0 + 10)	None	Flares production
Housing	506	13 (12 + 1)	None	House prices
Read	681	25 (24 + 1)	1097	Readership satisfaction
Servo	167	4 (0 + 4)	None	Rise time of a servomechanism

Table 2
10-Fold cross-validation performance comparison of RFP, KNN and Rule-based Regression on ten real-world data sets. For RFP and KNN, $K = 10$. Train and test times are in milliseconds; a 0 indicates the time is below 0.1. Best values are shown in boldface

Data set	RFP			KNN			Rule		
	RE	Train	Test	RE	Train	Test	RE	Train	Test
Abalone	0.748	130.8	**41.7**	**0.646**	**8.9**	4509.6	0.899	16 332.2	463.6
auto-mpg	0.426	7	**1**	**0.343**	**0**	31.8	0.450	249.6	20.3
buying	**0.911**	9	**1**	0.968	**0**	4	0.945	132.2	14
country	**1.439**	5	**0.4**	1.788	**0**	4	6.306	122.6	14.9
cpu	0.766	3	**0**	1.288	**0**	8	**0.677**	123	15.9
Electric	**1.032**	6	**1**	1.083	**0**	13.4	1.528	166.4	17.5
flare	**1.368**	28.4	31.3	1.579	**2**	288.6	1.792	315.7	**20.7**
Housing	0.798	16.4	**3.6**	0.651	**1**	63.8	**0.640**	643.1	32
read	**1.011**	**44.4**	**15.1**	1.042	**2**	169.6	1.352	732.2	37.9
servo	0.822	1	**0**	0.639	**0**	4	**0.229**	73.7	13

requires a complexity of O(K). Finding the feature prediction using K nearest neighbors requires a time proportional to O(K). As the variance of all the data is computed once in the training phase, the complexity of computing local weight is also O(K). After finding predictions for each feature dimension, the complexity of taking weighted average of all feature predictions is O(N). The overall complexity of prediction of a query instance is O(N·(log M + K)). Note that assuming $M \gg K$, this complexity is O(N·log M). The test times of the algorithms, run on the datasets shown in Table 1, are given in Table 2.

5.2. Prediction accuracy

In order to evaluate the prediction performance of a regression method, we used relative error (*RE*) computed by the following formula:

$$\mathrm{RE} = \frac{\mathrm{MAD}}{\frac{1}{Q}\sum_{i=1}^{Q} |t(q_i) - \bar{t}|} \tag{12}$$

where Q is the number of query instances, $\bar{t}$ is the median of the target values of training instances and MAD (Mean Absolute Distance) is defined below.

$$\mathrm{MAD} = \frac{1}{Q}\sum_{i=1}^{Q} |t(q_i) - \hat{t}(q_i)| \tag{13}$$

In order to compare the RFP algorithm with KNN and Rules learning algorithms, we used abalone, auto-mpg, buying, country, cpu, electric, flare, housing, read and servo real world datasets for function approximation (available at http://funapp.cs.bilkent.edu.tr [11]). The information about the number of instances, number and type of features and presence of missing values are summarized in Table 1.

We have measured the error rate RE, using 10-fold cross-validation. We have compared the results for RFP with the results of KNN and Rule-based regression [26], for $K = 10$.

From the results given in Table 2, we can easily conclude that, RFP has the shortest test time (except flare), while KNN has the shortest training time on all data sets. Average RE for RFP over the ten data sets is 0.932. On the other hand, the rule-based regression algorithm achieved 1.489 average RE, and KNN had 1.003 average RE. Therefore, we can conclude that the RFP algorithm achieves better performance on relative error and prediction time then KNN and Rule-based regression.

5.3. Preprocessing

In most data mining applications, preprocessing of the data takes more time than running the machine learning algorithms; however less research has been carried out on it. Dealing with missing feature values, determining and applying a normalization procedure and applying sampling to the data in order to decrease running time of the algorithm are some of such preprocessing tasks. As RFP handles missing values in a natural way, simply ignoring these missing feature values, no such preprocessing is required. Most other learning algorithms require these missing values to be filled with some data values [12]. Another advantage of the RFP is that it does not require any normalization on the data, because the distance measures used involves only a single feature, and predictions are carried out separately for each feature dimension. Finally the efficient computation time of the method eliminates most sampling tasks needed to speed up data mining applications.

5.4. Curse of dimensionality

For very large dimensions and with moderate number of training instances, other methods, except projection pursuit regression [9], suffer from sparsity. In other words, as the dimension increases, much more data are required to make better approximation. This problem is known as *curse of dimensionality*. As RFP makes approximations on each feature dimension separately, the density of instances at any local region of feature projections does not change as the dimension increase. Therefore RFP is suitable for data sets with large number of features, probably including some irrelevant ones.

5.5. Locality of information

Recursive partitioning regression methods, such as regression tree induction systems, partition the instance space into disjoint local regions. This partitioning causes some problems. One of them is the approximations carried out on the boundaries of regions will not be continuous. More intuitively, if a single value is predicted for each region, the prediction will not change at any location within the boundaries of a region. Another problem with tree induction methods is that, slight changes on the parent regions (parent nodes of the tree) may produce quite different regions at the leaf nodes of the tree. For such problems different modifications to tree induction methods are carried out by producing overlapping regions with increased computation [8,10]. The RFP algorithm handles local information like other lazy approaches, as an approximation is carried out according to the local position of each query instance.

6. Conclusion

We have described a regression method called RFP, based on feature projections, which achieves fast computation time, by preserving a comparable or better accuracy with other popular regression algorithms. The method inherits most of the properties of lazy regression methods and has some additional benefits. Besides fast prediction time, it handles some of the problems generally resolved with an additional preprocessing. These results encourage us to present this method as a data mining solution for high dimensional databases with very large sizes, by an additional advantage of eliminating curse of dimensionality problem. Future works can be directed towards new methods, which inherit the advantages of RFP, and also deals with interactions, in order to reach much better prediction performance. Further new methods can be developed for regression that make generalizations on feature projections, in order to enable the interpretation of data.

References

[1] D. Aha, D. Kibler, M. Albert, Instance-based learning algorithms, Machine Learning 6 (1991) 37–66.
[2] G.C. Atkenson, A.W. Moore, S. Schaal, Locally weighted learning [http://funapp.cs.bilkent.edu.tr], 1996.
[3] C. Blake, E. Keogh, C.J. Merz, UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html] Irvine, CA, University of California, Department of Information and Computer Science, 1998
[4] I. Bratko, A. Karalic, First order regression, Machine Learning 26 (1997) 147–176.
[5] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.
[6] M.L. Breiman, L. Breiman, Stacked regressions, Machine Learning 24 (1996) 49–64.
[7] R. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[8] J.H. Friedman, Multivariate adaptive regression splines, The Annals of Statistics 19 (1) (1991) 1–141.
[9] J.H. Friedman, W. Stuetzle, Projection pursuit regression, Journal of American Statistical Association 76 (1981) 817–823.
[10] J.H. Friedman, Local learning based on recursive covering, Department of Statistics, Stanford University [ftp://stat.stanford.edu/pub/friedman/dart.ps.Z], 1996.
[11] H.A. Guvenir, Bilkent FunApp repository [http://funapp.cs.bilkent.edu.tr], Bilkent University, Department of Computer Engineering, Ankara, 2000.
[12] R. Greiner, A.J. Grove, A. Kogan, Knowing what doesn't matter: exploiting the omission of irrelevant data, Artificial Intelligence 97 (1997) 345–380.
[13] H.A. Guvenir, İ. Sirin, Classification by feature partitioning, Machine Learning 23 (1996) 47–67.
[14] H.A. Guvenir, G. Demiroz, N. Ilter, Learning differential diagnosis of erythemato squamous diseases using voting feature intervals, Artificial Intelligence in Medicine 13 (1998) 147–165.
[15] R.C. Holte, Very simple classification rules perform well on most commonly used datasets, Machine Learning 11 (1993) 63–91.
[16] A. Karalic, Employing linear regression in regression tree leaves, in: B. Newmann (Ed.), Proceedings of ECAI'92 Vienna, Austria, 1992, pp. 440–441.
[17] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data—An Introduction to Cluster Analysis, Wiley Series in Probability and Mathematical Statistics 1990.
[18] D. Kibler, D.W. Aha, M.K. Albert, Instance-based prediction of real-valued attributes, Computational Intelligence 5 (1989) 51–57.
[19] J.H. Mathews, Numerical Methods for Computer Science, Engineering and Mathematics, Prentice-Hall, Englewood Cliffs, NJ, 1987.
[20] T.M. Mitchell, Machine Learning, McGraw Hill, New York, 1997.
[21] J.R. Quinlan, Learning with continuous classes, in: Adams and Sterling (Eds.), Proceedings, AI'92, 1992, pp. 343–348.
[22] J. Rawlings, Applied Regression Analysis, A Research Tool, Wadsworth, Belmont, CA, 1988.
[23] SPSS Sample Data Sets [ftp://ftp.spss.com/pub/spss/sample/datasets/], 1999.
[24] S. Weiss, N. Indurkhya, Optimized rule induction, IEEE Expert 8 (6) (1993) 61–69.
[25] S. Weiss, N. Indurkhya, Rule-based machine learning methods for functional prediction, Journal of Artificial Intelligence Research 3 (1995) 383–403.
[26] S. Weiss, N. Indurkhya, Predictive Data Mining: A Practical Guide, Morgan Kaufmann, San Francisco, 1998.