Regression by Feature Projections*

İlhan Uysal H.Altay Güvenir

Department of Computer Engineering and Information Sciences,
Bilkent University, 06533 Ankara, Turkey
{uilhan,guvenir}@cs.bilkent.edu.tr

Abstract. This paper describes a machine learning method, called *Regression by Feature Projections* (RFP), for predicting a real-valued target feature. 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 approximation procedures executed sequentially. The first approximation process is to find the individual predictions of features by using the K-nearest neighbor algorithm (KNN). The second approximation process combines the predictions of all features. During the first approximation 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 step and enforce the method to have an adaptive or context-sensitive nature. We have compared RFP with the KNN algorithm. Results on real data sets show that RFP is much faster than KNN, yet its prediction accuracy is comparable with the KNN algorithm.

1 Introduction

We will describe a method in the paper for predicting a continuous target feature. 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. There are two different approaches for regression in the literature: *eager* and *lazy* learning. The term *eager* is used for the learning systems that construct models that represent knowledge using the training data. After training, predictions are made 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 called *Regression by Feature Projections* (RFP), to predict a continuous target, where the instances are stored as their projections on each feature dimension. In RFP method, we use the KNN algorithm together with linear least squares approximation to find the prediction at each feature dimension. Then we find the precision of those features at the local

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

position of query instance. We define the precision as a local weight which 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 obtained by combining individual feature predictions and using their local weights.

RFP eliminates some problems met in real data sets. Those are missing feature values, irrelevant features and normalization of data. The major limitation of the algorithm is its assumption that contribution of each feature to the final prediction is independent of other features.

The empirical results show that RFP method is much faster than its natural competitor KNN, and achieves a comparable accuracy. The description of the weighted KNN regression algorithm we used for comparisons is given in [5]. For most data mining or knowledge discovery applications, where very large databases are in concern, this is thought of a solution because of its small computational complexity, and elimination of the above problems with real data sets.

In Section 2 and Section 3 description of RFP and its evaluation are given respectively. Finally in Section 4, conclusions and future works are presented.

2 Regression by Feature Projections

In this section we introduce a lazy regression method based-on feature projections, called *Regression by Feature Projections* (RFP). The main property of the algorithm is that, a different approximation is done for each feature, where the training data is projected to every feature. This approximation is done by using the nearest instances to the query point, where these instances may differ at each feature dimension, independent of other features. The final prediction is found with the weighted combination of feature predictions.

2.1 Training

As we have described above, training involves simply storing the training set as projections to the features. This is done by associating a copy of 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, they are simply ignored on the corresponding features.

2.2 Approximation at Feature Projections

In the first approximation step of the prediction algorithm, we employ the KNN algorithm at each feature dimension. Since the instances are sorted according to feature values in the training, the nearest neighbors can be found by a binary search. Then all K nearest neighbors are found by comparing the sorted feature values of neighboring instances. After determining the K nearest instances, a prediction is made for that feature using the feature and target values of these

instances. We apply linear least squares approximation, given by Equation (1) to find the predicted target value at a particular feature dimension. Linear least squares algorithm is described in [4], which minimizes the sum of squared errors of instances (2).

$$\bar{y}_i = \beta_0 + \beta_1 x_{i1} \tag{1}$$

$$Error = \sum_{i=1}^{n} (y_i - \bar{y}_i)^2 \tag{2}$$

where n is the number of instances and y_i is the actual target value.

After constructing a linear equation by using the linear least squares algorithm, the prediction at a particular feature projection is done by simply substituting the feature value of the query instance to this equation.

2.3 Local Weight

Some regions at any feature dimension may produce better approximation than others. In order to obtain a measure for estimation at a particular feature, we employ a weighting measure in the prediction algorithm. If the region that query point fall is smooth, we give a high weight to that feature in the final prediction. By this way we both eliminate the effect of irrelevant features, as well as the irrelevant regions of a feature dimension. This establishes an adaptive, or context-sensitive nature, where at different locations in the instance space, the contribution of features on the final approximation differs. Since we employ linear least squares approximation, the smoothness is determined according to the constructed linear equation.

In order to measure the degree of smoothness, we compute the distance weighted mean of squared differences of the target values of the nearest neighbors and their estimated values found by using linear equation. We denote this measure with V_f shown in Equation (4). By subtracting it from the variance of the target values of all instances, V_{all}, we find the explained variance for that region, and by normalizing it with the variance of training set we obtain a measure, called *prediction index* (PI)given in Equation (6). We use the squared PI as the *local weight* (LW) for each feature (7).

$$V_{all} = \frac{\sum_{i=1}^{n} (y_i - \bar{y})^2}{n} \tag{3}$$

$$V_f = \frac{\sum_{i=1}^{k} w_i (y_i - \bar{y}_i)^2}{\sum_{i=1}^{k} w_i} \tag{4}$$

where n is the number of instances, $\bar{y}$ is the mean of target values of training set, $\bar{y}_i$ is the estimation of the feature for ith instance and w_i is defined in Equation (5).

$$w_i = \frac{1}{\epsilon + (x_{if} - x_{qf})^2} \tag{5}$$

where ϵ is a positive real number close to zero.

```
Training:
  Store each feature value with target separately
  Sort each input feature dimension according to feature values

Prediction(q,k):
/* q: query instance   k: number of neighbors */
  Let Sum_weight = 0 and prediction = 0
  for each feature f
     if feature value of q is not missing
         Find k nearest neighbors
          /* apply binary search to find the nearest neighbor */
         Find linear least squares estimate, Pf
         Find local weight, LW
         Sum_weight = Sum_weight + LW
         prediction = prediction + LW * Pf
  prediction = prediction / Sum_weight
  return (prediction)
```

Fig. 1. Training and Prediction Algorithms

$$PI_f = \frac{V_{all} - V_f}{V_{all}} \tag{6}$$

$$LW_f = \begin{cases} PI_f^2 \text{ if } PI_f > 0 \\ 0 \quad \text{otherwise} \end{cases} \tag{7}$$

2.4 Prediction

We find the final approximation, by merging the predictions found for each feature dimension. This is obtained by averaging these results where the local weights are also employed. Figure 1 summarizes the prediction phase as well as the training.

If there are missing values of a query instance, this situation is refined by the prediction algorithm, by simply ignoring the prediction on the feature dimension whose value of query is missing. Finally a prediction is done by giving higher weights to the feature predictions, whose local regions at the query location are smooth.

3 Empirical Evaluation

RFP inherits most properties of other lazy approaches. Two most important benefits of lazy learning are very small training complexity and handling local information in the instance space. RFP benefits these properties with an additional property of having small prediction time. The method also deals with both types of input features, categorical and continuous, and handles irrelevant

features. The single drawback of the method is its inability for dealing with interactions or relations among input features which lead to a decrease in prediction accuracy. However, we have observed that generally the real world datasets do not contain such interactions between features [1, 2, 3]. 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, since it can eliminate the irrelevant features by assigning them lower weights.

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

$$RE = \frac{MSE}{\frac{1}{t}\sum_{i=1}^{t}(y_t - \bar{y})^2} \tag{8}$$

where t is the number of test cases, $\bar{y}$ is the median of the target values of training instances and mean squared error(MSE) is defined below.

$$MSE = \frac{1}{t}\sum_{i=1}^{t}(y_t - prediction_t)^2 \tag{9}$$

Most of the real world datasets are selected from the collection of datasets provided by the machine learning group at University of California at Irvine. The information about the number of instances, features and missing values and the target features of datasets[2] we used for experiments are summarized in Table 1.

Dataset	Instances	Features	Miss.Val.	Target Feature
abalone	4177	8	None	Rings (Age=Rings+1.5)
cpu	209	9	None	Relative CPU Performance
housing	506	13	None	Housing Values in Boston
villages1	887	32	Many	Animal Resources
villages2	766	32	Many	Agriculture Area

Table 1. Datasets

We have measured the error rate RE, using 10-fold cross-validation. We have compared the results for RFP with the results of distance-weighted-KNN (WKNN), for K values of 5 and 10.

From the results given in Table 2, we can easily conclude that, RFP achieves a comparable prediction performance with K-nearest neighbor algorithm. For villages datasets, where there are many features and missing values, it achieves lower prediction error.

The computational complexity of RFP is $O(m\log(n))$. It is better than the complexity of KNN, $O(mn)$, which is apparent in empirical results, especially for large data sets.

[2] The official villages dataset includes data about villages around the same region. It can be obtained from the authors.

K	Data:	abalone	cpu	housing	villages1	villages2
5	RFP	0.56	0.30	0.60	0.94	0.90
10	RFP	0.57	0.25	0.60	0.95	0.90
10	TEST TIME(ms)	768	24	94	430	529
5	WKNN	0.51	0.52	0.39	1.46	1.33
10	WKNN	0.47	0.52	0.39	1.14	1.14
10	TEST TIME(ms)	8047	17.2	143	635	853

Table 2. RE Rates of RFP

4 Conclusions

We have described a regression method called RFP, based on feature projections, which achieves a fast computation time, by preserving a comparable accuracy with the most popular lazy method, KNN. The method inherits most of the properties of lazy regression methods and have some additional benefits. It handles missing values appropriately simply by ignoring them and handles both nominal and continuous feature values. Besides them, it does not require a normalization of the data which is an important process required for KNN algorithm. Finally RFP is appropriate method for data sets having irrelevant features, since it employs a weighting for them. The performance results and fast computation time of RFP encourage us to present this method as a data mining solution for high dimensional databases with very large sizes. On the other hand the major limitation of RFP is its assumption that the features are independent. Future works can be directed towards new methods which inherit the advantages of RFP and deals with interactions, in order to reach much better prediction performance. Also new methods can be developed for regression that make generalizations on feature projections, in order to enable the interpretation of data.

References

1. Güvenir, H. A. and Sirin, İ.,Classification by Feature Partitioning, *Machine Learning*, 23:47-67, 1996.
2. Güvenir, H. A. and Demiroz, G., Ilter N., Learning Differential Diagnosis of Erythemato Squamous Diseases using Voting Feature Intervals, *Artificial Intelligence in Medicine*, 13:147-165, 1998.
3. Holte, R. C., Very Simple Classification Rules Perform Well on Most Commonly Used Data Sets, *Machine Learning*, 11:63-91, 1993.
4. Mathews,J.H.,Numerical Methods for Computer Science, Engineering and Mathematics *Prentice-Hall*, 1987.
5. Mitchell,T.M.,Machine Learning, *McGraw Hill*, 1997.