CLUEKR : CLUstering Based Efficient *k*NN Regression

Harshit Dubey[1] and Vikram Pudi[2]

[1] International Institute of Information Technology, Hyderabad, Andhra Pradesh, India 500032
harshit.dubeyug08@students.iiit.ac.in
[2] International Institute of Information Technology, Hyderabad, Andhra Pradesh, India 500032
vikram@iiit.ac.in

Abstract. K-Nearest Neighbor based regression algorithm assigns a value to the query instance based on the values of its neighborhood instances. Although kNN has proved to be a ubiquitous classification/regression tool with good scalability but it suffers from some drawbacks. One of its biggest drawback is that, it is a lazy learner i.e. it uses all the training data at runtime. In this paper, we propose a novel, efficient and accurate, clustering based kNN regression algorithm CLUEKR having the advantage of low computational complexity. Instead of searching for nearest neighbors directly in the entire dataset, we first hierarchically cluster the data and then find the cluster in which the query point should lie. Our empirical experiments with several real world datasets show that our algorithm reduces the search space for kNN significantly and is yet accurate.

Keywords: Regression, Efficient, Accurate, K-Nearest Neighbor, Clusters, Hierarchy, Likelihood.

1 Introduction

The problem of regression is to estimate the value of a dependent variable based on the values of one or more independent variables, e.g., predicting price increase based on demand or money supply based on inflation rate etc. Regression analysis helps to understand how the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed. Regression algorithms can be used for prediction (including forecasting of time-series data), inference, hypothesis-testing and modeling of causal relationships.

In statistics, Regression is considered as collection of statistical function-fitting techniques. These techniques are classified according to the form of the function being fit to the data. Regression analysis has been studied extensively in statistics, but there have been only a few studies from the data mining perspective. Majority of this study resulted into algorithms that fall under the following

J. Pei et al. (Eds.): PAKDD 2013, Part I, LNAI 7818, pp. 450–458, 2013.

broad categories - Linear Regression [12], Nearest Neighbor Algorithms [5], Decision Trees [3], Support Vector Machines [4], Neural Networks [7] and Logistic Regression [8]. However most of these algorithms were originally developed for classification purpose, but have been later modified for regression.

In the recent past, a lot of research centered at nearest neighbor methodology has been performed. Instead of being computationally expensive kNN algorithm is very simple to understand, accurate, requires only a few parameters to be tuned and is robust with regard to the search space. Also kNN classifier can be updated at a very little cost as new training instances with known classes are presented. A strong point of kNN is that, for all data distributions, its probability of error is bounded above by twice the Bayes probability of error[10]. However one of the major drawbacks of kNN is that, it is a lazy learner i.e. it uses all the training data at runtime. However for majority of the datasets, we perform as accurate as kNN. In this paper, we propose a novel, efficient and accurate, clustering based kNN regression algorithm CLUEKR, which instead of searching for nearest neighbors directly in the entire dataset, first find the cluster in which the query point has maximum likelihood of occurrence. We first hierarchically cluster the data in the pre-processing step, then a recursive search starting from root node of the hierarchy is performed. For current search node in the hierarchy, we select a cluster among its child, in which the query point has maximum likelihood of occurrence and then a recursive search is applied to it. Finally we find the k nearest neighbors of query points in the obtained cluster and return the weighted mean of their response variable as result.

The organization of rest of the paper is as follows. In section 2, we throw light on related, and recent, work in the literature. Section 3 deals with problem formulation. We explain the modified algorithm in Section 4. In Section 5, experimental results are presented together with a thorough comparison with the state-of-the-art algorithms. Finally, in Section 6, conclusions are drawn.

2 Related Work

Traditional Statistical Approaches follow a methodology that requires the form of the curve to be specified in advance. This requires regression problems in each special application domain be studied and solved optimally for that domain. Another problem with these approaches is outlier (extreme cases) sensitivity. The most common statistical regression approach is linear regression which assumes the entire data to follow a linear relationship between the response and feature variables. But this assumption is unlikely to hold on variety of application.

Segmented or piecewise regression [11] is a method in regression analysis in which the independent variable is partitioned into intervals and a separate line segment is fit to each interval. It is is essentially a wedding of hierarchical clustering and standard regression theory. It can also be performed on multivariate data by partitioning the various independent variables. Segmented regression is useful when the independent variables, clustered into different groups, exhibit different relationships between the variables in these regions. The boundaries between the segments are breakpoints.

Most of the Regression approaches in data mining falls under these categories - Nearest Neighbor, Regression trees, Neural Networks, and Support Vector Machines. Regression trees are a variation of decision trees in which the predicted outcome is a real number. Neural Networks and SVM techniques are quite complex and an in-depth analysis of results obtained is not possible.

One of the oldest, accurate and simplest method for pattern classification and regression is K-Nearest-Neighbor (kNN) [5]. kNN algorithms have been identified as one of the top ten most influential data mining algorithms [14] for their ability of producing simple but powerful classifiers. It has been studied at length over the past few decades and is widely applied in many fields. Despite its simplicity, the kNN rule often yields competitive results. However one of the major drawbacks of kNN is that, it is a lazy learner i.e. it uses all the training data at the runtime.

A recent work on prototype reduction, called Weighted Distance Nearest Neighbor (WDNN) [9] is based on retaining the informative instances and learning their weights for classification. The algorithm assigns a non negative weight to each training instance tuple at the training phase and only the training instances with positive weight are retained (as the prototypes) in the test phase. However the algorithm is specifically designed for classification purpose and cannot be used for regression.

In another recent work [13], a Parameterless, Accurate, Generic, Efficient NN-Based Regression algorithm PAGER is proposed, which is based on the assumption that value of the dependent variable varies smoothly with the variation in values of independent variable. For each dimension in the search space, the authors construct a 1-dimensional predictor as a line passing through two closest neighbor of the query point. For each of the predictor obtained, they determine the mean error occurred if the predictor was used in prediction of the k-nearest neighbors. A weight inversely proportional to the mean error is assigned to each predictor. Finally a weighted sum of the value output by individual predictors for the query instance is assigned to it.

Saket and others propose a kNN based regression algorithm BINER : BINary search based Efficient Regression [2], which instead of directly predicting the value of response variable recursively narrows down the range in which the response variable lies. In the pre-processing step training data is sorted based on the value of response variable and is hierarchically structured. At each level in the hierarchy, data is divided into three parts, one containing elements from first to middle, other contains elements from middle to last and the third contains elements from middle of first portion to middle of second portion. The algorithm then finds the portion in which the query point has maximum likelihood to lie and finally kNN algorithm is applied to that portion.

Qi Yu and others in one of their recent work [15] proposed a methodology named Optimally Pruned K-Nearest Neighbors (OP-kNNs) which builds a one hidden-layer feedforward neural network using K-Nearest Neighbors as kernels to perform regression. The approach performed better compared to state-of-the-art methods while remaining fast.

3 Problem Formulation

In this section, we present the problem of regression and notation used to model the dataset.

The problem of regression is to estimate the value of a dependent variable (known as response variable) based on the values of one or more independent variables (known as feature variables). We model the tuple as $\{X, y\}$ where X is an ordered set of attribute values like $\{x_1, x_2, \ldots, x_d\}$ and y is the numeric variable to be predicted. Here x_i is the value of the i^{th} attribute and there are d attributes overall corresponding to a d-dimensional space.

Formally, the problem has the following inputs:

- An ordered set of feature variables Q i.e. $\{q_1, q_2, \ldots, q_d\}$
- A set of n tuples called the training dataset, $D, = \{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$.

The output is an estimated value of y for the given query Q. Mathematically, it can be represented as

$$y = f(X, D, parameters), \tag{1}$$

where *parameters* are the arguments which the function $f()$ takes. These are generally set by user and are learned by trial and error method.

4 The CLUEKR Algorithm

We describe our proposed algorithm in this section. Our algorithm proceeds in two steps :

- It first find the cluster in the hierarchy, in which the query point has maximum likelihood of occurrence.
- kNN is applied to points present in the cluster, and weighted mean of the k nearest neighbors of query point in the cluster is quoted as output.

Size of the obtained cluster is less compared to the size of entire dataset, in this way our algorithm reduce the search space for K-Nearest Neighbor algorithm. Now we explain in detail about the clustering phase (pre-processing step) first and later throw light on the actual algorithm.

4.1 Pre-processing

In the pre-processing step, data is hierarchically clustered and mean value for each of the cluster is calculated and stored. Each node in this hierarchy consist of clusters containing data point, which are recursively divided into three child clusters as we move down the hierarchy. We make use of the fact that similar instances have similar value of response variable (basic analogy on which kNN works), while selecting the cluster center. Each cluster node is sorted based on

the value of response variable, then $n/4$ and $3n/4$ ranked instance are selected as the center for two of the child clusters. For the third cluster, mean of the other two cluster's center is taken as center. All the points present in the cluster node are divided into child cluster, based on to which child cluster center the point is closest.

We aim to divide each cluster nodes is such a fashion, that each child node contains half the data present in the parent node. However at each level we have added an extra child cluster to make the division of points smooth, this cluster also contain points belonging to other cluster that lies at its boundary with other cluster. This will help to properly classify the query instance that lie at the boundary of clusters. Boundary points to third cluster are defined as, points whose distance ratio from other cluster center to third cluster center is between 0.9 to 1.0.

We recursively divide the cluster nodes, till we get a cluster node which contain less that $2 * k$ points. The limiting size of $2 * k$ was chosen in order to keep a margin for selection of k nearest neighbors.

4.2 Actual Algorithm

Pseudo code for the actual algorithm is provided in Algo. 1. The recursive search starts from the root node and goes down the hierarchy. In order to find the cluster among the child nodes (for current node in the search) in which the query point has maximum likelihood of occurrence, distance of query point from mean value of all the child cluster of current node is calculated (lines 1-3). If the two closest distances comes out to be similar ($Confidance$ returns $false$) then we can't say with confidence, that which child cluster to pick and hence kNN algorithm is applied to the current cluster (line 5), else recursive search continues on the child cluster which is closest to the Query point (lines 7-8) i.e distance from mean of that cluster is least.

We say that two distances, d_i and d_j are similar if $min(d_i/d_j, d_j/d_i)$ is greater than 0.90. The value of 0.90 was selected by experimentations and it works well on most of the datasets as shown in the experimental section.

Algorithm 1. CLUEKR : Pseudo code

Input: *Query instance* Q, *Pointer to Root node in hierarchy* R, *Parameter* k
Output: *Value of Response Variable for Query instance* Q

1: **for** *each node* j *in childs of* R **do**
2: $dist[j] = distance(Q, mean(j))$
3: **end for**
4: **if** $Confidance(dist)$ **then**
5: $ChildPtr = argmin(dist)$
6: **return** $CLUEKR(Q, ChildPtr,\ k)$
7: **else**
8: **return** $\mathrm{k}NN(Q,\ Data\ of\ R,\ k)$
9: **end if**

4.3 Complexity Analysis

We perform complexity analysis for both pre-processing step and actual algorithm in this section.

- In the pre-processing step, hierarchy consisting of cluster node is constructed, for which each cluster node is divided into child clusters. Assuming that we have data points sorted based on the value of response variable in the cluster node to be split, we can select the centers for child cluster directly and then data can be redistributed among the clusters in $O(n_s)$, where n_s is the size of cluster node to be spitted. If the cluster node is transversed in sorted order of response variable to redistribute data in child cluster, then child cluster will also contain data, sorted based on response variable. So if the root node has sorted data based on response variable, all other nodes in the hierarchy will also follow the same ordering of data. As data is distributed from parent to child cluster, total amount of data present in child cluster is equal to data present in the parent cluster (ignoring the extra boundary points present in third cluster). Total time complexity of pre-processing step comes out to $O(n*log(n))+O(h*n)$, where $n*log(n)$ is the cost involved in sorting root node and $h*n$ is the cost of splitting cluster nodes to construct a hierarchy of height h. This also involve the cost of calculating mean of each cluster node, as it can be calculated during the transversal of cluster node for distribution of data.
- At runtime our algorithm first performs a search for the cluster in the hierarchy,in which the query point has maximum likelihood of occurrence and then kNN is applied to the obtained cluster. Runtime complexity of our algorithm is $O(h + n_c)$, where h is the height of the hierarchy and n_c is the size of the cluster obtained. As n_c is supposed to be less compared to n, our algorithm is faster at runtime compared to kNN.

5 Experimental Study

5.1 Performance Model

In this section, we demonstrate our experimental settings. The experiments were obtained on a wide variety of real life datasets obtained from UCI data repository [1] and Weka Datasets [6].A short description of all the datasets used is provided in Table 1. We have compared our performance against the following approaches: K Nearest Neighbor, Isotonic , Linear Regression(Linear Reg.), Least Mean Square (LMS) algorithm, Radial Basis Function Network (RBF Network), Regression Tree (RepTree) and Decision Stump(Dec Stump). Most of the algorithms are available as part of the Weka toolkit. All the results have been obtained using 10-fold cross validation technique.

We have used two metrics for quantifying our results, namely, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). MAE is mean of the absolute errors (actual output - predicted output). RMSE is square root of mean of squared errors. We have used euclidean distance matrix to calculate the distances in our algorithm.

Table 1. Dataset Description

Dataset	#Instances	#Attributes	Dataset	#Instances	#Attributes
Autompg	392	8	Bank	8192	9
Bodyfat	252	15	Concrete	1030	9
Cpu	209	7	Forestfire	517	11
Flow	103	8	Housing	507	14
Space	3107	7	Slump	103	8
Synfriedman	500	6	Synfriedman1	500	6

Table 2. Comparison of results of CLUEKR with other standard approaches

Dataset	CLUEKR		*k*NN		Isotonic		Linear Reg.		LMS		RBF		Rep Tree		Dec Stump	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Autompg	**2.36**	**3.24**	**2.40**	**3.59**	3.16	4.3	2.56	3.4	**2.50**	**2.59**	3.90	5.07	**2.30**	**3.31**	4.2	5.18
Bank	**0.02**	**0.03**	**0.02**	**0.03**	0.03	0.46	**0.02**	**0.03**	0.03	0.05	0.04	0.06	**0.02**	**0.03**	0.04	0.05
Bodyfat	0.51	0.65	0.49	0.62	0.52	0.67	**0.43**	**0.56**	**0.45**	**0.58**	0.61	0.77	0.52	0.67	0.63	0.8
Concrete	7.20	9.92	**5.71**	**8.14**	10.81	13.45	8.30	10.45	9.52	17.53	13.38	16.56	**5.43**	**7.38**	11.54	14.46
Cpu	**21.45**	**70.76**	**18.79**	**74.37**	**23.12**	**50.95**	36.05	66.24	33.97	108.32	51.87	126.35	32.34	93.21	71.56	126.40
Flow	**11.27**	**14.78**	11.89	16.42	11.55	14.18	**10.99**	**13.26**	13.28	18.56	14.76	17.42	12.11	15.57	12.48	15.41
Forestfire	21.83	69.04	21.75	68.13	19.18	63.5	19.92	64.28	**12.88**	**64.91**	**18.86**	**63.86**	19.24	64.56	18.93	64.68
Housing	**2.96**	**4.63**	**2.97**	**4.63**	3.80	5.32	3.39	4.91	3.42	5.55	6.13	8.42	3.18	4.84	5.61	7.5
Slump	**5.56**	**7.75**	5.89	9.83	**5.86**	**7.52**	6.67	7.82	6.68	10.56	6.98	8.73	6.14	8.19	7.05	8.86
Space	**0.10**	**0.17**	**0.10**	**0.17**	0.12	0.16	0.11	0.16	0.11	0.15	0.14	0.19	**0.10**	**0.14**	0.13	0.18
Synf.	**2.10**	**2.64**	**1.96**	**2.45**	3.69	4.44	2.25	2.83	2.24	2.82	3.86	4.81	2.69	3.45	3.73	4.63
Synf. 1	**1.92**	**2.48**	**1.75**	**2.16**	3.59	4.32	2.76	4.65	2.45	4.71	3.92	4.91	2.57	3.24	3.69	4.4

Table 3. Comparison of dataset size with size of the cluster obtained

Dataset	Dataset Size	Cluster Size	%Ratio	Dataset	Dataset Size	Cluster Size	%Ratio
Autompg	392	76	20	Bank	8192	2975	36
Bodyfat	252	50	20	Concrete	1030	191	18
Cpu	209	53	25	Forestfire	517	156	30
Flow	103	35	33	Housing	507	105	20
Space	3107	262	9	Slump	103	34	33
Synf.	500	143	28	Synf. 1	500	136	27

5.2 Results and Discussion

Table 2 compares the results obtained for our algorithm with other existing state of art approaches, top two results are highlighted in bold. As can be seen from the results, for majority of the datasets, we perform as accurate as kNN, however for some of the datasets, our algorithm also outperform kNN. Also our algorithm more than often outperforms other existing state-of-the art algorithms. However for concrete dataset in which the response variable is highly non-linear function of its attributes, we do not perform as good as for other datasets. Table 3 compares average size of cluster obtained by our algorithm with original dataset size.Ratio column in the table shows the percentage ratio of the cluster size obtained by our algorithm to the original dataset size. It is clear from the data, that our algorithm reduces the search space for kNN significantly, however the degree of reduction varies depending on the type of dataset.

6 Conclusion

In this paper, we have proposed a novel clustering based k nearest neighbor regression algorithm which is efficient and accurate. Our work is based on reducing the search space for nearest neighbors for any given point. We hierarchically cluster the data in pre-processing step and then search is performed to find the cluster in the hierarchy, in which the query point has the maximum likelihood of occurrence. We have also evaluated our approach against the existing state-of-the-art regression algorithms. As shown in the experimental section, our approaches reduces the search space for kNN significantly and is yet accurate.

References

1. Newman, D., Asuncion, A.: UCI machine learning repository (2007)
2. Bharambe, S., Dubey, H., Pudi, V.: BINER: BINary Search Based Efficient Regression. In: Perner, P. (ed.) MLDM 2012. LNCS, vol. 7376, pp. 76–85. Springer, Heidelberg (2012)
3. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey (1984)
4. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20, 273–297 (1995), doi:10.1007/BF00994018
5. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, New York (2001)
6. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. 11, 10–18 (2009)
7. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall (1999)
8. Hosmer, D., Lemeshow, S.: Applied Logistic Regression. Wiley Series in Probability and Statistics: Texts and References Section. John Wiley & Sons (2000)
9. Jahromi, M.Z., Parvinnia, E., John, R.: A method of learning weighted similarity function to improve the performance of nearest neighbor. Inf. Sci. 179, 2964–2973 (2009)

10. Loizou, G., Maybank, S.J.: The nearest neighbor and the bayes error rates. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9(2), 254–262 (1987)
11. McZgee, V.E., Carleton, W.T.: Piecewise regression. Journal of the American Statistical Association 65(331), 1109–1124 (1970)
12. Montgomery, D., Peck, E., Vining, G.: Introduction to linear regression analysis. Wiley series in probability and statistics: Texts, references, and pocketbooks section. Wiley (2001)
13. Singh, H., Desai, A., Pudi, V.: PAGER: Parameterless, accurate, generic, efficient kNN-based regression. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010, Part II. LNCS, vol. 6262, pp. 168–176. Springer, Heidelberg (2010)
14. Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.-H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. Knowl. Inf. Syst. 14, 1–37 (2007)
15. Yu, Q., Miche, Y., Sorjamaa, A., Guillen, A., Lendasse, A., Séverin, E.: Op-knn: method and applications. Adv. Artif. Neu. Sys. 1, 1:1–1:6 (2010)