PAGER: Parameterless, Accurate, Generic, Efficient kNN-based Regression

by

Aditya Desai, Himanshu singh, Vikram Pudi

Report No: IIIT/TR/2009/157

Centre for Data Engineering
International Institute of Information Technology
Hyderabad - 500 032, INDIA
October 2009

PAGER: Parameterless, Accurate, Generic, Efficient *k*NN-based Regression

Aditya Desai, Himanshu Singh, Vikram Pudi

Center for Data Engineering, International Institute of Information Technology, Hyderabad, India.
{aditya_desai,himanshusingh}@students.iiit.ac.in,vikram@iiit.ac.in

1 Introduction

The problem of regression is to estimate the value of a dependent numeric variable based on the values of one or more independent numeric variables. Regression algorithms can be used for prediction (including forecasting of time-series data), inference, hypothesis testing, and modeling of causal relationships. Although this problem has been studied extensively in statistics, it has not received much attention from the data mining community [5, 11].

Statistical approaches try to model the relationship between the dependent and independent variables as a *closed form* function. It is the user's responsibility to make an intelligent guess about the form of this function. This is done by studying the application domain, and may involve trial-and-error methods. Once the function's form is fixed, its parameters (or coefficients) are estimated so as to give a "best fit" to the available data. Typically, the function also has an error term that is used to compensate for unexplained variation in the dependent variable.

The above nature of statistical approaches requires that regression problems in each specific application domain are separately studied and solved optimally for that domain. For example, [6] applies non-linear regression models for longitudinal data with errors that follow a skew-elliptical distribution. The main objective of the study was to predict normal versus abnormal pregnancy outcomes from beta human chorionic gonadotropin data.

Another problem with the statistical approaches is *outlier sensitivity*. Outliers (extreme cases) can seriously bias the results by pulling or pushing the regression curve in a particular direction, leading to biased regression coefficients. Often, excluding just a single extreme case can yield a completely different set of results. In this paper we present a new regression algorithm **PAGER** – **P**arameterless, **A**ccurate, **G**eneric, **E**fficient *k*NN-based **R**egression. PAGER has the following desirable features:

1. **Parameterless:** We first design a version of PAGER that takes two input parameters, and then show how these parameters can be automatically set, thereby resulting in a parameterless algorithm. This removes the burden from the user of having to set parameter values – a process that typically involves repeated trial-and-error for every application domain and dataset.

2. **Accurate:** Our experimental study in Section 3 shows that PAGER provides more accurate estimates than its competitors on several datasets. Among the algorithms we included for comparison are the best available algorithms from the Weka toolkit [15].
3. **Generic:** Our approach is generic as it is based on piecewise linear nature for continuous functions which is valid for most real-life datasets. Hence, it will work "out-of-the-box" and doesn't require to be tinkered with for every application domain and dataset.
4. **Efficient:** PAGER is based on the nearest neighbour (k-NN) approach and is thereby equally efficient, provided there are indexes available for easily finding the k nearest neighbours.
5. **Simple:** The design of PAGER is simple, as it is based on the k-NN approach. This makes it easy to implement, maintain, embed and modify as the situation demands.
6. **Outlier Resilient:** The output of PAGER for a particular input record R is dependent only on the nearest neighbours of R and is therefore insensitive to far-away ouliers.

The remainder of the report is organized as follows: In Section 2 we present the PAGER algorithm. Then, in Section 3 we experimentally evaluate our algorithm and show the results. Finally, in Section 4, we summarize the conclusions of our study and identify future work.

2 The PAGER Algorithm

In this section we present the PAGER (Parameterless, Accurate, Generic, Efficient kNN-based Regression) algorithm. Being derived from nearest neighbour methods, PAGER is also simple and outlier-resilient. These desirable features make PAGER a very attractive alternative to existing approaches. In the remainder of this section, we use the notation shown in Table 1. First in Section 2.1, we present a simple version of PAGER that requires two input parameters, and then improve it by presenting two improved variations in Sections 2.2 and 2.3.

k	The number of closest neighbours used for prediction.
E_{th}	The error threshold
D	The Training Data
A_i	Denotes a feature
X	The feature vectors space. X= $(A_1, \ldots, A_n)$.
T	A tuple in the X-space. T= $(t_1, \ldots, t_n)$
y	The response variable.
w_i	The weight assigned to feature A_i.
e_i	The overall mean error of prediction in the A_i, y plane.

Table 1. Notation

2.1 Fixed Parameter PAGER

This version of PAGER takes two fixed parameters as input: (1) k, and (2) E_{th}. In addition, as mentioned in the problem definition (Section **??**), the input contains the training data D and the test tuple $T = (t_1, \ldots, t_n)$ in X-space whose value of the response variable y is to be estimated.

The algorithm then operates as follows:

First, determine the 2 nearest neighbours of T in X-space, say T_1 and T_2. Then, for each feature value x_i, $i = 1, \ldots, n$, consider the space composed of (x_i, y), and let L_i be the line that passes through T_1 and T_2 in the plane of (A_i, y). Using L_i, we can estimate the value of y for different values of x_i.

Using each L_i computed above, estimate the value of y for T. We get n values of y (say $y_1, \ldots, y_n$) that are not necessarily equal. The algorithm finally outputs a normalized weighted average of these y_i:

$$y = \frac{w_1 \times y_1 + \ldots + w_n \times y_n}{w_1 + \ldots + w_n} \tag{1}$$

In order to calculate the weights, the following procedure is used:

Using each L_i, estimate the value of y for the k nearest neighbours of T in D. Since the actual values of y for these points is known, we compute e_i, using L_i in estimating their y.

The lower the value of e_i, more closely A_i is related to y, and hence such closely related variables should get higher weights. In order to consider only dimensions that are closely related to y, we set $w_i = 0$ for dimensions for which $e_i > E_{th} \times min(e_i)$, where E_{th} is the error threshold input to the algorithm. For all other dimensions the weights are assigned as $w_i = max(e_i)/e_i$.

2.2 Parameterless PAGER

We observed that selecting the correct k is a crucial issue as noisy neighbours contribute to incorrect error estimates. These incorrect error estimates would further contribute to inaccurate weights for dimensions and hence wrong predicted values. Moreover, there is no value of k that is *globally correct* for all test data points. The correct value depends on the specific location of the input test point T and is perhaps dependent on the density of training points around T.

In order to determine the correct value of k for a given test point T, we use the following procedure which takes 2 optional parameters as input. These two parameters correspond to the lower (min) and upper bound (max) of k. It is not necessary to set these parameters accurately, and they are used only for improving run-time efficiency.

This procedure iterates over different values of k, from min to max and executes the algorithm in the previous section for each value of k to estimate the value of y for T. While running this algorithm, it additionally computes the error in prediction for all the $k - 2$ closest neighbours starting from the third closest neighbour to the k^{th} closest neighbour, ignoring the two neighbours which are

used for constructing the L_i. The mean error of estimation ($\overline{E_k}$) for a given value of k is defined using the formula below:

$$\overline{E_k} = \frac{(E_3 + \ldots + E_k)}{k-2}$$

where each E_i is the error in prediction of the i^{th}-closest neighbour using equation 1. The error value $\overline{E_k}$ is compared with the error values generated in previous iterations and the k for which the lowest error value was reported is chosen. The value of y for this k is the output predicted value. In case the user doesn't input max, we can detect the value of k after which mean error value starts increasing steadily and then use this k for further calculations. The rationale is that as k increases, initially, error would reduce upto a point and then steadily increase due to increasing noisy or unrelated far away neighbours. On the other hand, if the user doesn't input min, we can set min to 1.

The algorithm of the previous section also had E_{th} as one of the parameter. Setting an accurate value for E_{th} is not critical – it is only for improving computational efficiency. This is because E_{th} is only used to remove dimensions with high error. This is not necessary as those dimensions with high error tend to have a very small weight and hence do not affect the predicted values much.

By removing both k and E_{th} parameters, the resulting algorithm becomes parameterless.

2.3 Weighted Distance PAGER

The algorithms in the previous two subsections rely on euclidean distance metric for finding the k nearest neighbours.

For two tuples $T_1 = (t_{11}, \ldots, t_{1n})$ and $T_2 = (t_{21}, \ldots, t_{2n})$, the distance is computed as $\sum_{i=1}^{n}(t_{1i} - t_{2i})^2$.

We also note that in the previous sections, weights were assigned to different dimensions according to the accuracy of prediction in these dimensions. Higher the weights assigned, higher was the accuracy of prediction in those dimensions and vice versa.

The idea in this PAGER variation is to use the weights of dimensions in the calculation of distance – and thereby use a weighted euclidean distance. The nearest neighbour computation of the previous sections is done using this distance metric. The result of this change is that the distance along more important (high-weight) dimensions will be magnified *i.e.* points that are far away along these dimensions will move even farther and thereby will not be considered as part of the neighbourhood.

The modified distance metric is computed as $\sum_{i=1}^{n}(w_i * (t_{1i} - t_{2i}))^2$, where $w_1, \ldots, w_n$ are the weights derived for the best k in the previous iteration.

Specifically, the weighted distance PAGER runs as follows:

The parameterless PAGER of the previous subsection is iterated more than once where the distance function used in the first iteration is the regular euclidean distance, but in the succeeding iterations a weighted euclidean distance

is used where the weights are the same as that obtained for the best k in the previous iteration. The rest of the steps followed in that iteration are same as for parameterless PAGER. Weighted PAGER terminates when the least mean error of estimation for neighbours is greater than or equal to that obtained in the previous iterations.

2.4 Efficiency Considerations

The algorithms presented in the previous two subsections repeat the basic algorithm (of Section 2.1) for several iterations. This may seem computationally extravagant. However, this is not a problem because the value of k that produces effective accurate results is typically very small (less than 10). This means that the basic algorithm can be iterated over several times and still maintain interactive response times.

2.5 Illustration of PAGER

In this section we illustrate with an example as to how given an input tuple, the task of estimation of the dependant variable takes place. Given an input training dataset and an input test tuple, the task is to find the corresponding value of the dependant variable. Let the input tuple be T=(24.1,32.1,29.3,?) with the fourth attribute to be estimated. Let the training dataset contain 10 points. The points and their euclidean distance from T are depicted in a tabular form in Table 2. Let the independant variable dimensions be considered as (A_0, A_1, A_2) and the dependant variable y.

Id	Tuple	Distance
T0	23.7, 32.1, 28.9, 18.7	0.379
T1	24.0, 32.9, 29.2, 18.4	0.508
T2	24.5, 32.1, 28.6, 18.0	0.582
T3	23.4, 32.5, 29.8, 17.4	0.601
T4	23.2, 32.4, 29.7, 19.0	0.616
T5	23.2, 31.8, 29.7, 18.3	0.616
T6	24.1, 32.9, 29.8, 18.8	0.630
T7	23.9, 31.4, 29.9, 18.9	0.645
T8	23.6, 32.7, 29.9, 19.1	0.656
T9	25.0, 31.3, 29.2, 19.1	0.711

Table 2. Illustrative Dataset

Lines constructed using first two neighbours (T_0, T_1) in each dimension
Let a line be constructed in the plane of A_0 and y for the two closest neighbours T_0 : (23.7, 32.1, 28.9, 18.7), T_1 : (24.0, 32.9, 29.2, 18.4). Thus the line to be

constructed should pass through the points (23.7, 18.7) and (24.0, 18.4). Thus the equation of the line is $y = \alpha \times A_0 + \beta$, where $\alpha = (18.7 - 18.4)/(23.7 - 24.0) = -1.0$ and $\beta = (18.7 - \alpha \times 23.7) = (18.7 + 1 \times 23.7) = 42.4$.

Similar lines are constructed passing through T_0 and T_1 in the plane of A_1 and y and similarly, in the plane of A_2 and y. The (α,β) values of the lines in these planes are $(-0.375, 30.7375)$ and $(-1.0, 47.6)$ respectively.

Using these equations, the y values of neighbours can be estimated by knowing the values of their independant variable. As the values of y are already known for the training points, the errors are also computed in these dimensions. Thus, the values of error are computed from neighbour T_2 onwards (as the 2 neighbours before this are used for line construction), using the formulas derived for lines in (A_0, y), (A_1, y) and (A_2, y) planes. Let the error for neighbour T_i for dimension A_j be e_{ij}.

Computing the correct number of neighbours We start our computation from the minimum number of neighbours which in this case was set to 5. The first two neighbours are ignored for the very fact that the line has been constructed using these two neighbours themselves and hence the error will always be 0 for these points.

Thus the errors (e_{20}, e_{21}, e_{22}) are computed as said above for neighbour (T_2) and similar computation is done for neighbours (T_3), (T_4). After this step, mean error in each dimension is computed. The mean error in dimensions A_0, A_1, A_2 for number of neighbours 5 is given as:

$$\text{Error}(A_0) = (e_{20} + e_{30} + e_{40})/3$$
$$\text{Error}(A_1) = (e_{21} + e_{31} + e_{41})/3$$
$$\text{Error}(A_2) = (e_{22} + e_{32} + e_{42})/3$$

Now, a low mean error in any dimension indicates that the data along that dimension is highly linear and hence the predictions along that dimension are more accurate. Thus we set a weight for each dimension to be inversely proportional to the mean error in that dimension. The mean error for number of neighbours(k) "5" are: (2.600, 1.492, 1.370) in dimensions (A_0, A_1, A_2) respectively. Here the constant of proportionality can be taken to be the maximum of the error and hence the weights are (w_1, w_2, w_3) = (2.6/2.6,2.6/1.492,2.6/1.37) = (1.0, 1.741, 1.897). Those dimensions with errors greater than $E_{th} \times min(Error(A_i))$ are set to zero weights.

Using the equations of lines in each dimension, we estimate the values of the response variable for the given input tuple in the (A_0, y), (A_1, y) and (A_2, y) planes. Let the the predicted values be y_0 in (A_0, y) plane, y_1 in (A_1, y) plane, and so on. Knowing the mean error values in different dimensions, it is intuitive to assign weights in such a manner that the value of the dependant variable for the input tuple should be more inclined towards dimensions which are stable rather than other unstable dimensions. Higher errors indicate lower stability and

lower errors indicate higher stability. Using this concept, the value of dependant variable is estimated as $(y_1 \times w_1 + y_2 \times w_2 + y_3 \times w_3)/(w_1 + w_2 + w_3)$ where w_1, w_2 and w_3 are computed previously as (1.0, 1.741, 1.897) and the value of y_i is computed using the line equations as done previously.

The same line equations are again used to predict dependant variable values of neighbours and the values of errors are recorded. For number of neighbours(k) 5, errors in prediction for T_2, T_3, T_4 are computed – the mean error is 0.5407 and the predicted value is 18.4501. Similarly, errors are computed until a maximum number of neighbours which can be set to a sufficiently large value which in this case we set to 9. The predicted values and mean errors for neighbours, on choosing k as 6, 7, 8 and 9 are (18.4862, 0.6365), (18.4754, 0.4228), (18.5662, 0.4216) and (18.5514, 0.4027) respectively.

It has been observed that smaller the mean error values for the neighbours, more correct are the weight estimates and hence more correct are the predicted values. This intuition is seen to be accurate in this case as the predicted value 18.5514 is closest to the actual value 18.50 when the mean error of the neighbours is minimum at 0.4027. The values of weights obtained for the best number of neighbours are used for similarity computation in the next iteration.

The parameterless nature of PAGER comes from the counter-balancing forces exerted by the threshold values. Consider, the error-threshold parameter whose value need not be accurate as those dimensions with higher error typically contribute little by means of smaller weights and hence setting the error-threshold value is not a serious issue. Two other parameters that are generally set are minimum number of neighbours and the maximum number of neighbours. Setting the minimum to a small value and the maximum to a high value can be a reasonable choice due to an exhaustive search in between. The mean error along the dimensions, the stability of dimensions and the weightings in this case counterbalance each other to provide accurate estimates for the value of predicted variables thus making the entire process virtually parameterless.

3 Experimental Study

In this section, we evaluate the proposed PAGER algorithm. We describe the experimental setting and performance metrics in Section 3.1 and the experimental results in Section 3.3. The comparitive results are in Table 4.

3.1 Experimental Setting

We compare our algorithm against twelve other algorithms. One of these is the weighted k-NN based approach [4, 8] described in Section **??**. The remaining eleven are available in the Weka toolkit [15] namely Additive Regression, Gaussian Regression, Isotonic Regression, Least Median Square Regression (LMS), Linear Regression, Multi Layer Perceptron based Regression (MLP), Pace Regression, RBF Regression, Simple Linear Regression, SMO Regression, and SVM Regression.

The version of PAGER used is the Weighted Distance PAGER.

We used four datasets in our experimental study, namely CPU (dataset available with Weka), Housing [2], Concrete [16] and Body-Fat [1]. All the results and comparison have been done using the *leave one out* comparison technique which is a specific case of n-folds cross validation, where the n is set to the number of instances of the dataset. The metrics used for comparison are RMSE (Root Mean Square Error) and ABME (Absolute Mean Error). The k-NN technique of [4, 8] was run on number-of-neighbours=10. The algorithms in Weka were run with the parameters as can be found in Table 3. The parameters of Weka are not mentioned here due to lack of space and are available in [15].

Algorithm	Parameter settings
Additive Regression	(-S 1.0 -I 10 -W weka.classifiers.trees.DecisionStump)
Gaussian Regression	(-L 1.0 -N 0 -K weka.classifiers.functions. supportVector.RBFKernel -C 250007 -G 1.0)
Isotonic Regression	(IsotonicRegression)
Least Median Square Regression [18]	(LeastMedSq -S 4 -G 0)
Linear Regression	(LinearRegression -S 0 -R 1.0E-8)
Multi Layer Perceptron [8]	(MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a)
Pace Regression [4][5]	(PaceRegression -E eb)
RBF Network	(RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1)
Simple Linear Regression	(SimpleLinearRegression)
SMOreg Regression [6][7]	(SMOreg -S 0.001 -C 1.0 -T 0.001 -P 1.0E-12 -N 0 -K weka.classifiers.functions. supportVector.PolyKernel -C 250007 -E 1.0)
SVMReg Regression [6][7][15]	(SVMreg -C 1.0 -N 0 -I weka.classifiers.functions. supportVector.RegSMOImproved -L 0.001 -W 1 -P 1.0E-12 -T 0.001 -V -K weka.classifiers.functions. supportVector.PolyKernel -C 250007 -E 1.0)

Table 3. Experimental Settings

3.2 Dataset Description

In this section, we describe each dataset used for experiments.

CPU Dataset The CPU dataset was obtained directly from Weka and has 6 independant variables and 1 dependant variable. The dataset has 209 tuples. The task of the predictor is to estimate the relative performance given its machine cycle time in nanoseconds (integer), minimum main memory in kilobytes (integer), maximum main memory in kilobytes (integer), cache memory in kilobytes (integer), minimum channels in units (integer), and maximum channels in units

(integer).

Housing Dataset The Housing dataset [2] was obtained from the UCI data repository and has 13 independant variables and 1 dependant variable. The dataset has 506 tuples. The Housing Dataset concerns housing values in suburbs of Boston, which is the variable value to be predicted. The independant variables are per capita crime rate by town, proportion of residential land zones for lots over 25,000 sq.ft., proportion of non-retail business acres per town, Charles River dummy variable (= 1 if tract bounds river; 0 otherwise), nitric oxides concentration (parts per 10 million), average number of rooms per dwelling, proportion of owner-occupied units built prior to 1940, weighted distances to five Boston employment centres, index of accessibility to radial highways, full-value property-tax rate per $10,000, pupil-teacher ratio by town, B: 1000 (Bk - 0.63)$\hat{2}$ where Bk is the proportion of blacks by town, % lower status of the population, Median value of owner-occupied homes in $1000's.

Concrete Dataset The Concrete dataset [16] was obtained from the UCI data repository [2] and has 8 independant variables and 1 dependant variable. The dataset has 1030 tuples. The UCI dataset site reported that "the concrete compressive strength is a highly nonlinear function of age and ingredients". This dataset is therefore particularly challenging and has not been studied before for regression analysis, to the best of our knowledge. The task is to estimate the concrete compressive strength(Mpa) of the mixture given quantative attributes Cement (Kg/m^3), Blast Furnace Slag (Kg/m^3), Fly-Ash (Kg/m^3), Water (Kg/m^3), Superplasticizer (Kg/m^3), Coarse Aggregate (Kg/m^3), Fine Aggregate(Kg/m^3), Age.

BodyFat Dataset The BodyFat dataset was obtained from the CMU data repository [1] and has 14 independant variables and 1 dependant variable. The dataset has 252 tuples. The task is to determine the percentage of bodyfat in an individual given Age (years), Weight (lbs), Height (inches), Neck circumference (cm), Chest circumference(cm), Abdomen 2 circumference (cm), Hip circumference (cm), Thigh circumference (cm), Knee circumference (cm), Ankle circumference (cm), Biceps (extended) circumference (cm), Forearm circumference (cm), and Wrist circumference (cm).

3.3 Results

In this section, we report the experimental results obtained for each dataset.

3.4 Discussion of Results

From the experimental results it is evident that on the CPU dataset PAGER outperforms all other algorithms except the regression algorithm based on Multilayer Perceptron. The reason for this is analyzed in detail below. On the other

Regression	CPU Dataset		Housing Dataset		Concrete Dataset		Bodyfat Dataset	
Algorithm	ABME	RMSE	ABME	RMSE	ABME	RMSE	ABME	RMSE
PAGER	*9.28*	*31.04*	***2.10***	***3.47***	***5.87***	***7.71***	***0.38***	***0.49***
Additive	25.46	59.06	3.35	4.86	6.67	8.45	0.53	0.64
Gaussian	15.08	81.57	2.58	3.89	6.07	7.82	0.42	0.56
Isotonic	23.93	51.98	3.81	5.32	10.84	13.52	0.55	0.69
LMS	33.60	107.55	3.36	5.40	9.28	16.55	0.43	0.55
Linear	34.61	55.22	3.37	4.84	8.29	10.46	0.42	0.54
MLP	**6.28**	**16.70**	3.10	4.64	6.58	8.61	0.50	0.70
Pace	34.83	56.12	3.36	4.82	8.32	10.52	0.41	0.53
RBF	52.25	119.28	6.05	8.19	13.43	16.67	0.61	0.77
Simple Linear	43.13	70.46	4.52	6.23	11.87	14.50	0.50	0.62
SMO	20.70	64.22	3.25	5.09	8.23	10.97	0.43	0.56
SVM	20.71	64.24	3.24	5.08	8.23	10.97	0.43	0.56
kNN	18.92	74.83	2.97	4.63	6.55	8.57	0.45	0.58

Table 4. Experimental Results on CPU, Housing, Concrete and Bodyfat Dataset

three datasets *i.e.* the Housing Dataset, BodyFat Dataset and the Concrete Dataset, PAGER outperforms all other algorithms. It is particularly noteworthy that our algorithm performs very well on the Concrete Dataset which was claimed to be a challenging, highly non-linear function of its attributes. The success of our algorithm is due to our very valid assumption that the data variation may not be linear throughout but is usually linear in a very small neighbourhood of the given input tuple. This assumption is true for majority of the real life datasets as variations of the dependent variable based on variations in the values of independent variables typically show a smooth transition.

A positive point in this algorithm that is evident through the illustrated example is its simplicity and generic nature. The parameterless nature of this code makes it easy to apply it to any domain even if sufficient domain knowledge is not available. However, it should be noted that our algorithm works only for numeric data with no missing values.

Even though our algorithm has performed well on the datasets as compared to the other algorithm one aspect that deserves attention in future is the neighbourhood selection mechanism. The current weighting of dimensions depends on the actual and estimated values of the response variable of neighbours. Table 5 shows that the best number of neighbours that can be selected varies very sharply from 5 to as high as 98 which is the reason why a static neighbour selection is highly infeasible. Below we present an analysis of our result for the CPU dataset where we were outperformed by the Multi-layer Perceptron approach, and state some of the improvements that can take place. In Table 5, it can be observed that for the 100th tuple the number of neighbours that we selected was 98 where as the best value was 12. This can be due to the fact that one of the closer neighbours had a very noisy dependant variable value while the succeeding neighbours had comparatively smaller errors in prediction. Thus, the mean

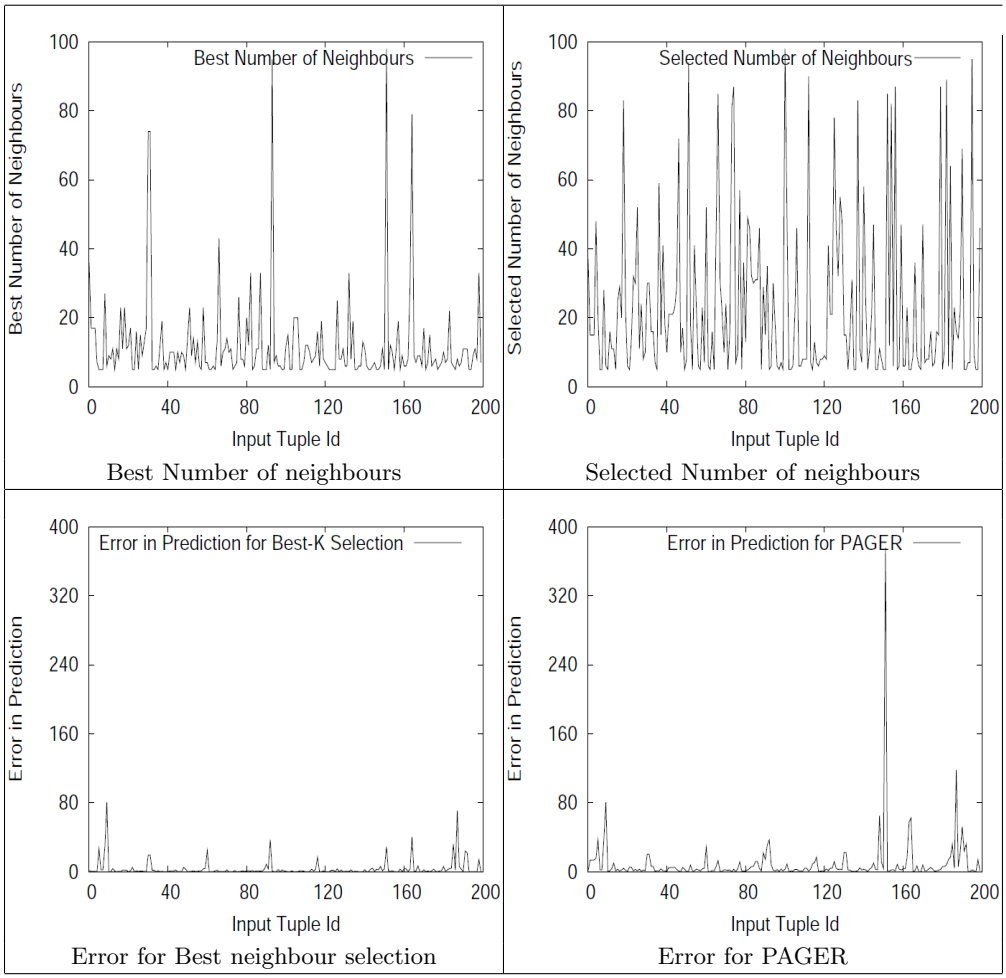

Table 5. Results on CPU Dataset

absolute error decreased as the number of neighbours increased. This decrease in mean error can be attributed to the fact that the error contributed by the noisy neighbour was supressed by the increasing number of neighbours and hence the algorithm chose the number of neighbours as 98. Another case that can be observed is that of tuple 151 where the best number of neighbours is 98, while the number of neighbours chosen is 5, due to which there is an extremely large error in prediction. These two cases show the importance the neighbour selection procedure lends to the algorithms. Other similar cases exist as can be clearly seen from the figure. Despite the fact that incorrect number of neighbours are chosen in some cases, it can be observed in Table 5 that difference between the

"mean absolute error of PAGER" v/s "mean absolute error if the algorithm in 4.1 is applied with the best number of neighbours", is small for most of the input tuples thus demonstrating the accuracy and parameterless nature of PAGER.

Even though the comparitive results show that PAGER outperforms or is comparable to other algorithms as cited in the experimental results there is much scope for improvement especially in the neighbour selection domain.

4 Conclusions

In this paper we have presented and evaluated PAGER, a new algorithm for regression based on nearest neighbour methods. Evaluation was done against 12 competing algorithms on 4 standard real-life datasets. Although simple, it outperformed all competing algorithms on all datasets but one. Unlike most other algorithms, PAGER can be used "out-of-the-box" without having to extensively tune or tweak it for each application domain and dataset. In fact, if the parameterless version is used, it requires absolutely no tuning.

Future work includes determinining high quality neighbours and the correct number of neighbours. This requires developing techniques such as neighbour selection algorithms, noisy neighbour elimination among others. Another future direction is to construct a curve or the closest fitting line from k neighbours instead of a line which is presently constructed from the two neighbours. It is also of interest to design algorithms that work when the independent variables are categorical, or come from a mixture of categorical and numeric domains.

References

1. The body fat dataset, 1985. http://lib.stat.cmu.edu/datasets/bodyfat.
2. A. Asuncion and D. Newman. UCI machine learning repository, 2007.
3. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression trees*. Wadsworth Inc., 1984.
4. E. Fix and J. L. H. Jr. Discriminatory analysis, non-parameteric discrimination: Consistency properties. Technical Report 21-49-004(4), USAF school of aviation medicine, Randolf field, Texas, 1951.
5. K.-M. Jung. Multivariate least-trimmed squares regression estimator. *Computational Statistics and Data Analysis (CSDA)*, 48(2)::307–316, 2005.
6. R. D. la Cruz. Bayesian non-linear regression models with skew-elliptical errors: Applications to the classification of longitudinal profiles. *Computational Statistics and Data Analysis (CSDA)*, 53(2)::436–449, 2008.
7. J. Mielniczuk and J. Tyrcha. Consistency of multilayer perceptron regression estimators. *Neural Networks*, 53(2)::1019–1022, 1993.
8. P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. Wiley, 1987.
9. S. Shevade, S. Keerthi, C. Bhattacharyya, and K. Murthy. Improvements to smo algorithm for svm regression. Technical Report CD-99-16, Control Division Dept of Mechanical and Production Engineering, National University of Singapore, 1999.

10. A. J. Smola and B. Scholkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2 Technical Report Series, 1998.
11. Q. F. Stout. Unimodal regression via prefix isotonic regression. *Computational Statistics and Data Analysis (CSDA)*, 53(2)::307–316, 2008.
12. Y. Wang. *A new approach to fitting linear models in high dimensional spaces*. PhD thesis, Department of Computer Science, University of Waikato, New Zealand, 2000.
13. Y. Wang and I. H. Witten. Modeling for optimal probability prediction, 2002.
14. M. Ware. Implementation of multilayer perceptron backpropagation, 2005. http://weka.sourceforge.net/doc/weka/classifiers/functions/MultilayerPerceptron.html.
15. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2 edition, 2005.
16. I.-C. Yeh. Modeling of strength of high performance concrete using artificial neural networks. *Cement and Concrete Research*, 28, No. 12:1797–1808, 1998.