NNRMLR Algorithm

A Review on a combined method of Nearest Neighbour Regression and Multiple Learning Regression

Sarvadnya Dalesh Desai

Electrical Department, Indian Institute of Technology, Bombay

210040138@iitb.ac.in

***Abstract*—This document is a review report on the NNRMLR Algorithm. [1] presented at the IIAI International Conference on Advanced Applied Informatics (IIAIAAI) held during 20-22 Sept. 2012. The paper presents a novel way of predicting continuous valued variables using multiple continuous valued features which are positively correlated with the target feature.**

I. Introduction

Linear regression is on of the most prevalent algorithms for continuous value prediction. This algorithm can be assess at a great depth analytical in view of its closed form analytical solution rather than having a numerical approximation.

K-Nearest Neighbours is a one of the most widely used classification algorithms. However, with a slight modification we can use it as a regression algorithm popularly known as Nearest Neighbour Regression.

We structure this review paper as follows, we start by introducing the two algorithms separately and showcase the results they have provided us with(results are showcased at the end). Then we move to the combined algorithm and scope for variations in the proposed vanilla algorithm. We have tried out a few of the variations our self but it was not possible to try and analyze all the variations on a single data set. Furthermore, testing a variation on a single data set may not provide use well-tested results. Lastly we shall put forth a few concluding remarks regarding the reviewing procedure followed.

II. Multiple Linear Regression

We have used the vanilla algorithm of the purpose of this project. We can find out the closed form solution for the weights and biases for predicting the target variable values using the following formula.

$$\theta = (X^T X)^{-1} X^T y \tag{1}$$

Where X is the data matrix with features as its columns and the data entries as its rows. 'y' is the target variable vector to the corresponding data entries. θ is a vector containing all the feature weights and biases.

The Multiple Linear Regression Algorithm works on a few assumptions:

1) **Linear Relationship:** There exists a linear relationship between the predictor variable and the target variable.
2) **No Multicollinearity:** Two predictor variables are in no way highly correlated with each other. If they are they can be reduced to a single feature without loss of performance for the algorithm.
3) **Independence:** The data entries are independent of each other and in no way are 2 instances of the same occurrence. If so they can undesirably skew the weights in their favour.
4) **Homoscedasticity:** The residuals have constant variance at every point in the linear model. This properly implies that all the random variables(the entries in the data set) have the same variance. We can say this generally holds true as all the data samples are nearly drawn from the same distribution. Generally we use a suitable spatial transform in case this assumption is violated.
5) **Multivariate Normality:** The residuals of the model are normally distributed. This can be visualize using a qq-plot.
6) **Target-Feature Correlation:** The target feature is assumed to be correlated with the predictor feature with a correlation value of 1. Weak correlation features add up to an erroneous prediction.

The last point is precisely why the chosen data set and target variable is justified. We wish to have a high correlation between the target variable(Total number of bedrooms) and the predictor features. In case of a failure of this we end up with a model with prediction error of as high as 1 or 1.5 error fraction on an average.

The target variable in this example doesn't hold much importance as compared to other entries in the data set such as property price. However, it obeys a greater number of the assumptions made above and thus is chosen for the purpose of this implementation.

III. K-Nearest Neighbour Regression

A. Overview of the Algorithm

K-Nearest Neighbour Regression is an modified algorithm of its widespread clustering counterpart. The algorithm goes as follows:

1) Find the euclidean distance between the all training features and test sample features.
2) Set a fixed number of neighbours with the least euclidean distance.
3) The prediction value is the average of the target values of these K-neighbour test samples we used.

4) Finally find the performance of this algorithm using MSE(Mean-Squared Error) and R2 Score as a standard metric.

The formula for the euclidean distance in the vanilla algorithm is as follows:

$$d_0(a_i, b_j) = \sqrt{\sum_{l=1}^{p}(a_{il} - b_{jl})^2} \tag{2}$$

Where there are 'p' prediction features, 'a' is an entry in the training set and 'b' is an entry in the test set.

A few of the underlying assumptions for this algorithm are:

1) **Distance Based :** The K-NN algorithm is a distance based algorithm which will fail in case the target variable value varies largely between near points and comparatively less between distant points.
2) **No Multicollinearity:** This is similar to the Multiple linear regression case. In case of very high correlation between prediction features we can merge dimensions. This will greatly improve the computation time of the training process.
3) **Discriminating Model:** A discriminating model is one where the target values are known and based on those we predict the target value of the test case whose actual target value is known too. A Discriminanting model is a supervised way of learning the data set.
4) **Computational Complexity:**The K-NN algorithm is and O(n^2) complexity algorithm as every computation of euclidean distance is to be done for each training set entry with respect to a corresponding test entry.

B. Implementation challenges and Solutions:

One of the implementation challenges of this algorithm was the large training time. However much couldn't be done here due to the fundamental limitations of the algorithm. Each cycle took about 13-15 mins to run, and there was no definite comment on which variation will run the fastest as the speed were largely dependent on the resources allocated by the cloud server of google Collaboratory. Better Methods may surely exist but this was the best ones we had at our disposal.

Another challenge at hand was normalization. We had values like population and total rooms as two of our prediction features. Now the values that these two quantities have are too distant in terms of numbers. How to bring them closer such that we can have dimensions of comparable size. Well we shall demonstrate 2 of the popular methods, both of which are implemented in our project.

$$d_0(a_i, b_j) = \sqrt{\sum_{l=1}^{p}(\frac{a_{il} - b_{jl}}{x_{l,avg}})^2} \tag{3}$$

$$d_0(a_i, b_j) = \sqrt{\sum_{l=1}^{p}(\frac{a_{il} - b_{jl}}{x_{l,max} - x_{l,min}})^2} \tag{4}$$

Where 'x_l' is the prediction feature from the full data set, 'p' is the number of prediction features, 'a' is an entry from the training set and 'b' from the test set.

IV. The combined NNRMLR Algorithm

The idea behind this algorithm is to have the best of both worlds. MLR algorithm tries ti derive an affine transformation between the predictor features and the target features. The K-NN algorithm is a localization algorithm which localizes the target variable value as an average of the target values of its K nearest neighbours in the orthogonal prediction feature space. The closed form for the algorithm is given by:

$$d_0(a_i, b_j) = \sqrt{\sum_{l=1}^{p}|\beta_l|(a_{il} - b_{jl})^2} \tag{5}$$

Where the data set has 'p' features, β is a the parameter predicted in the MLR algorithm, 'a' is an entry from the training set and 'b' an entry from the test set.

Another variation of the proposed algorithm was using acceleration coefficients, $'\gamma'$. The form of the euclidean distance is modified as follows:

$$d_0(a_i, b_j) = \sqrt{\sum_{l=1}^{p}|\beta_l|^{\gamma}(a_{il} - b_{jl})^2} \tag{6}$$

Where γ is the acceleration coefficient. According to the paper introducing NNRMLR algorithm [1], the value of $'\gamma'$ providing the best performance is between 2 and 3.

We thought of another variation to this algorithm. How about adding normalization along with the existing parameters from MLR. How shall this method fair? We tried out the following as the new euclidean distance:

$$d_0(a_i, b_j) = \sqrt{\sum_{l=1}^{p}|\beta_l|^{\gamma}(\frac{a_{il} - b_{jl}}{x_{l,max} - x_{l,min})^2}} \tag{7}$$

Where we used 'γ' as 2. We got better results compared to the MLR algorithm however the results were not as good as the initial NNRMLR algorithm. Surprisingly using acceleration coefficient didn't give us better performance compared to the NNRMLR vanilla algorithm.

V. Scope for future Experiments

We tried out a total of 5-6 variations in this project, however that doesn't restrict us from trying out the endless possibilities we can explore with this algorithm. A few of the questions that remain unanswered are:

1) How will this algorithm perform if the training set to test set ratio is modified?
2) How will this algorithm perform in case of less data. Does this algorithm a lot of training data to work well?
3) How will this algorithm perform in different data sets. What if the correlation between the prediction features and the target features is lower compared to this data set. Does this provide a relative lower error?
4) Trying out more acceleration coefficients. (In view of a large training time this couldn't be extensively tested.)

These are a few questions that can be answered in a following avenue. A few of them can be analytically be though

of too. MLR and K-NN both require loads of correlated data to train, NNRMLR which is a combination of the two should be no exception right?

VI. Obtained Results

A. Error Analysis of Applied Algorithms

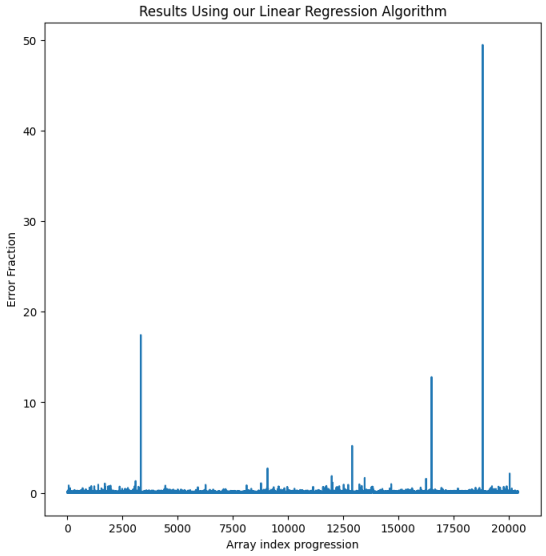

Fig. 1. Error Fraction v.s. data index plot for MLR

Mean Absolute Error:(MAE) 40.5438
Mean Squared Error:(MSE) 5847.3943
Root Mean Squared Error(RMSE): 76.4683

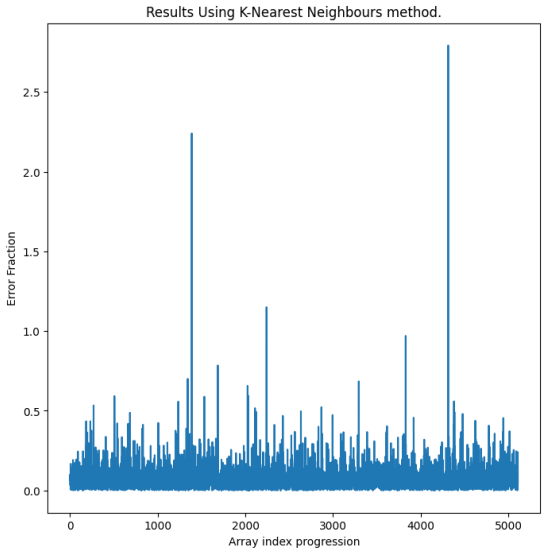

Fig. 2. Error Fraction v.s. data index plot for K-NN(avg. normalization)

Mean Absolute Error:(MAE) 33.0322
Mean Squared Error:(MSE) 3398.5705
Root Mean Squared Error(RMSE): 58.2973
R2 Score: 0.9810

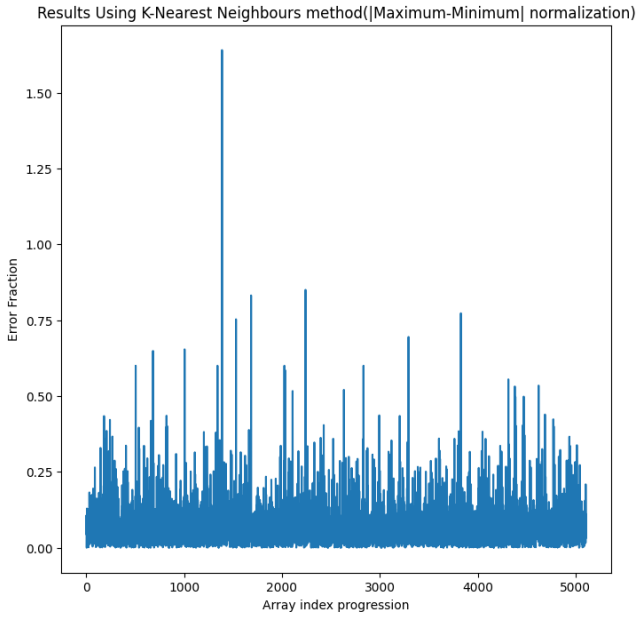

Fig. 3. Error Fraction v.s. data index plot for K-NN(max-min normalization)

Mean Absolute Error:(MAE) 32.7021
Mean Squared Error:(MSE) 3330.0284
Root Mean Squared Error(RMSE): 57.7064
R2 Score: 0.9814

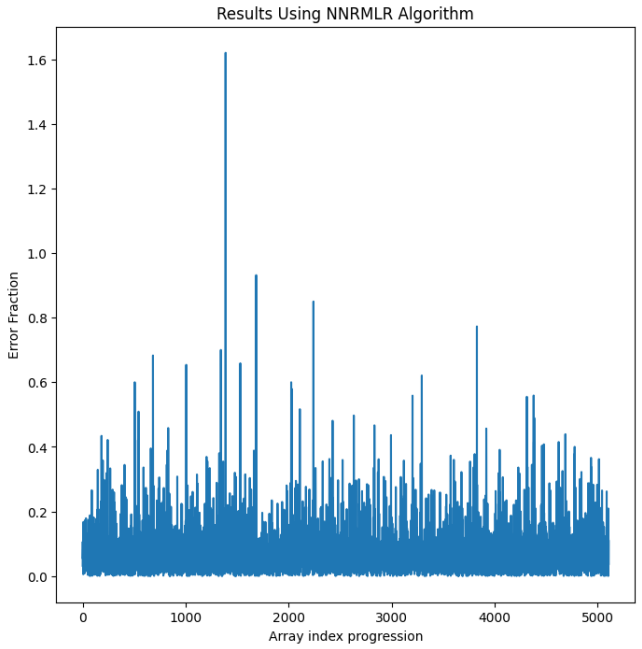

Fig. 4. Error Fraction v.s. data index plot for NNRMLR

Mean Absolute Error:(MAE) 33.0177
Mean Squared Error:(MSE) 3311.3960
Root Mean Squared Error(RMSE): 57.5447
R2 Score: 0.9815

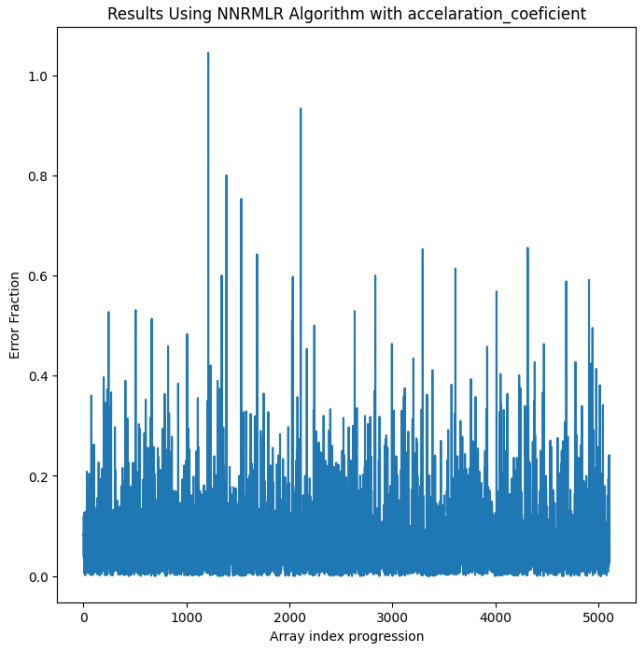

Fig. 5. Error Fraction v.s. data index plot for NNRMLR γ=2

Mean Absolute Error:(MAE) 32.9789
Mean Squared Error:(MSE) 4006.9372
Root Mean Squared Error(RMSE): 63.3004
R2 Score: 0.9776

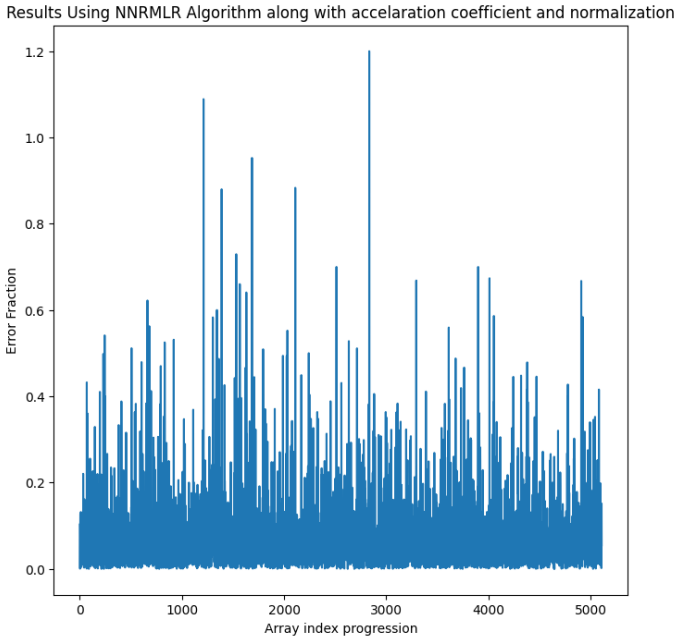

Fig. 6. Error Fraction v.s. data index plot for NNRMLR with Normalization

Mean Absolute Error:(MAE) 34.5562
Mean Squared Error:(MSE) 5562.0226
Root Mean Squared Error(RMSE): 74.5790
R2 Score: 0.9689

B. Scatter Plot Analysis of Prediction Features and Target

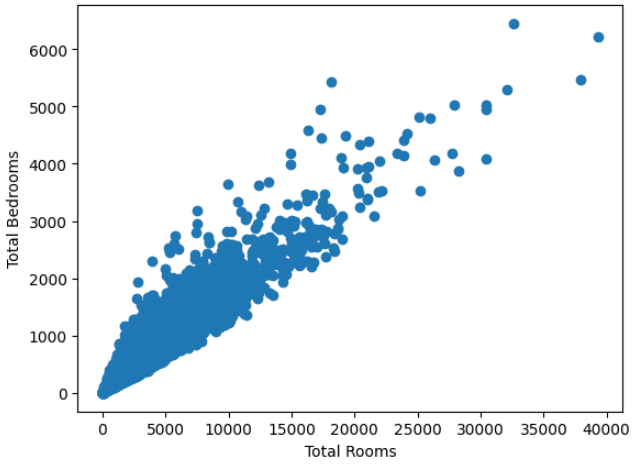

Fig. 7. Total Bedrooms vs Total Rooms

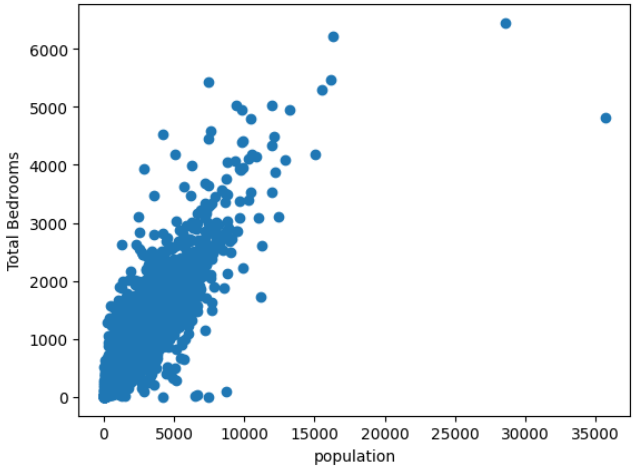

Fig. 8. Total Bedrooms vs Population

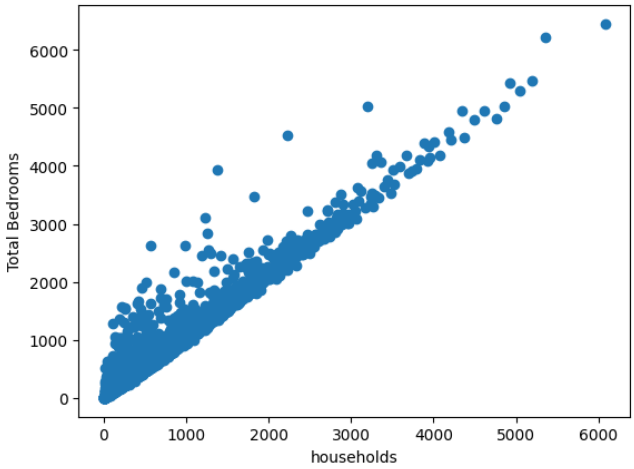

Fig. 9. Total Bedrooms vs Household

VII. Concluding Remarks

One of the prominent observations is that localization within the larger dataset has surely helped us. This can be observed using the error metrics calculated for the K-NN normalized prediction algorithm and the MLR algorithm. The NNRMLR algorithm has surely improved the metrics marginally however this can be extensively studied in further detail over more number of data sets.

Additional efforts can be put in to derive metrics which capture data such as number of incorrect predictions over a certain error threshold and see a comparative study between 2 variations of this algorithm. This might probably help us show that the acceleration coefficient and normalization indeed help the algorithm perform better. Another approach that could be used is modifying the train-test ratio to reduce over fitting.(This is assumed to have occurred as accelerated parameters performed worse than the mere weights).

References

[1] H. Hirose, Y. Soejima and K. Hirose, "NNRMLR: A Combined Method of Nearest Neighbor Regression and Multiple Linear Regression," 2012 IIAI International Conference on Advanced Applied Informatics, Fukuoka, Japan, 2012, pp. 351-356, doi: 10.1109/IIAI-AAI.2012.76.