Journal of Machine Learning Research 2 (2019) 1-10 Submitted 10/11; Published N/A

K-Nearest Neighbors and Other Related Algorithms

Norhan Abbas
Gianforte School of Computing
Montana State University
Bozeman, MT 59715, USA

Robert Lewis
Gianforte School of Computing
Montana State University
Bozeman, MT 59715, USA

Anthony Nardiello
Gianforte School of Computing
Montana State University
Bozeman, MT 59715, USA

Abstract

This paper describes various related algorithms including K-Nearest Neighbor, Edited Nearest Neighbor, Condensed Nearest Neighbor, K-Means Clustering, and Partitioning around K-Medoids. The algorithms are all implemented and when appropriate tested on regression and classification data. The results were recorded in the form of 0/1 loss (accuracy) for the classification problems and mean squared error (MSE) and subsequently root mean squared error (RMSE) for the regression problems. The hypothesis being tested follows the 'no free lunch' principle, wherein there is no single algorithm best suited for all regression and classification problems (or any problem applicable to a machine learning solution), thus the relative ranking of performance from each algorithm will differ from set to set. Each of these algorithms had several hyper-parameters that were tuned and results were recorded for the best performances of all algorithms on a given file under the hyper-parameters. This allows for a comparison of each algorithm on every data set given the best performance found for any given algorithm. The results were compared, and although general trends were found for the performances of all algorithms across any given data set, there was indeed a disparity between which algorithms were best suited for any given data set demonstrating the 'no free lunch' principle.

Keywords: Nearest Neighbor, Classification, Regression

1. Introduction

Suppose the following hypothetical situation: there are cats and dogs, each of which represents a particular machine learning algorithm. There are also people, who represent the data set to be used with each algorithms. Some people prefer cats, while others prefer dogs, and even still others prefer some other type of animal. Throughout all this though, there is no single animal who can be the best for every single person. As we move from animal to animal, we will see the overall ranking change, from group of people to group of people. It's this exact hypothetical situation which helps demonstrate the principle of No Free Lunch.

The 'No Free Lunch Theorem' states that for "any elevated performance over one class of problems is offset by performance over another class"Wolpert et al. (1997). This theorem has been proven mathematically via geometric interpretation of what it means for one algorithm to be well suited for a problem. Although this is a known principle, the algorithms tested here coupled with a quantification of their performance present a good opportunity to demonstrate this principle. The methods used herein represented neither a complete geometric interpretation of the algorithms nor its performance across any given data sets. Yet, they did give the sense that some algorithms could have elevated performance over others for certain problems. Ever algorithm coupled with its different hyper parameters effectively creates a Voronoi space under which regression and classification occur. The process of creating each Voronoi space is different for every algorithm, and by the 'no free lunch' theorem, there should be no optimal process for creating these Voronoi spaces for all possible data sets such that classification and regression perform completely optimally for new data.

2. Algorithm Implementation

The following five different algorithms had been implemented to perform classification or regression on some real-world data sets. They are instance-based learning algorithms, accordingly, they are trained to classify and/or predict using only specific data points.

- **K-Nearest Neighbor (KNN)**
  Implementing this algorithm, only the features, which were believed to be correlated to each other, were considered. By the same token, each instance was regarded as a unique data point due to the considered features. Then, Euclidean distance function was applied to predict the nearest neighbors for each data point. For illustration, comparing the relative distance between each point and the other data points could predict the number of similar instances (Zhenyun Deng (2016)).

- **Edited Nearest Neighbor (ENN)**
  Using KNN, Edited Nearest Neighbor filtered some noise by reducing the number of instances the data set has by removing the data points that differed from their neighboring instances (Grochowski Marek (2004)). Each instance was checked with KNN, if KNN did not manage to correctly classify that instance, it was then removed.

- **Condensed Nearest Neighbor (CNN)**
  Using KNN, CNN also reduced the data set by learning how to be able to classify each instance (Abdul Qayyum (2017)). First, a random data point was added to KNN classifier, so it could be compared to the next one. If a match occurred between between a candidate data point and its nearest neighbor in this new set, the candidate point was not added to the data set. A new data set was built up piecewise following this process. Accordingly, the resultant set of data points representing a reduced feature vector space could predict the class of different data points.

- **Clustering**

  - **K-Means**
    Clustering using K-Means aggregated instances in a cluster due to the similar features they had. First, ENN was implemented to reduce the number of data points that the data set had. So that, clustering for classification could be performed on a smaller data set since most of the instances left would be sharing similar attributes with their neighbors. For regression data sets, number of clusters was provided, which was 25% of the total number of data points.

  - **Partitioning Around Medoids (PAM)**
    This clustering method created a sequence of instances, which could be considered as the centers of the different clusters within the data set with each Medoid being a real point in the data set. The main goal of this algorithm was minimizing the average dissimilarity of the data points for the Euclidian Distance to their closest neighbors, such that the quality of clustering could be improved. Chih-Tang Chang (2010)

3. Experimental Setup

Six data sets from https://archive.ics.uci.edu/ml/datasets.php were used for training, testing, and validating the implementation of the five algorithms: Abalone Waugh (1995), Car Evaluation Bohanec (1997), Image Segmentation Brodley (1990), Computer Hardware Phillip Ein-Dor (1987), Forest Fires Paulo Cortez (2007), and Wine Quality P. Cortez (2009).

Euclidean distance was used to calculate the nearest neighbors for each algorithm. For numerical features, getting the Euclidean distance was trivial, but a different method had to be used to determine the distance between categorical features. In the implementation of Euclidean distance used, the distance between categorical features was 0 if the two features were held the same value. For different valued categorical features, the distance was assigned to be equal to the total number of features in the feature vector. This is not a perfect representation of distance between any two given categorical features, but it does incur a penalty for two categorical features having different values. This effectively weights categorical features differently than their numerical counterparts, and would represent a warping of the feature space. However this does capture dissimilarity between different valued categorical features.

After the folds had been created, the data set was partitioned into two parts: a training set, and a test set. Each of the 10 folds was held out once as a test set, and the union of the other nine folds was used as a training set for the algorithm, with the algorithm being run to train and test on each of these 10 partition combinations.

After an experiment had been run like this, and the results recorded, some of the features of the data set were then scrambled across a column, with each column corresponding to specific attribute values across the feature-class vectors.

Classification was performed on the first three data sets, Abalone, Car Evaluation, and Image Segmentation, where supervised and unsupervised techniques were used. Regression was performed on the second three data sets, Computer Hardware, Forest Fires, and Wine Quality, where only quarter of the data set were considered.

4. Discussion of Algorithm Performance

While theoretical discussion of algorithms and their ability to complete tasks could be an invigorating task to ponder, implementation often yielded results that could force someone to reconsider their understanding of the material. Such was the nature of the implementation for the five algorithm, as each algorithm proved to contain hidden challenges that needed to be rectified in order for our models to be built. These challenges proved to be far greater than what was originally envisioned, and as a result it implied the necessity of discussing the proposed solution to the most pressing of these issues.

Analysis began with the first algorithm, which was the K-Nearest Neighbor algorithm implementation. The K-Nearest Neighbor started off working exactly as theorized, which meant that as the value of K increases, the algorithm took longer to work. A similar principle happened with both Edited Nearest Neighbor and Condensed Nearest Neighbor, both of these had increased the time that the algorithm needed to run on their unique data set manipulation.

In addition, the time complexity of creating a model had grown exponentially with the clustering algorithms, K-Means Clustering (K-Means), and Partitioning around the Medoids (PAM). During initial runs of these algorithm implementations, models took around 45 minutes to get built and could test a single fold of 10-fold cross validation on the smaller data sets. The other three algorithms were actually capable of performing cross 10-fold validation across all data sets in four seconds. The number of CPU cycles needed to build optimal models for these data sets would absolutely be enormous. There was no realistic way to run the algorithms as was intended due to time constraints and lack of computing power. The decision was made to limit how long each algorithm could optimize the clusters across a single fold in 10-fold cross validation (20 minutes for K-Means, and 45 minutes for PAM). This decision was reached out of a necessity, but the understanding was that our results would definitely be effected by this cut-off as the models created did not fully represent the final models. Understanding why this happened would have been beneficial in early stages, such that some preparations could have been made to reduce the time in a more streamlined manner. However, it was possible to do plenty of testing using different hyper parameters for K-Means and PAM as the number of clusters and medoids used for each problem was fixed. Yet, with a single model generated it was possible to test many different ‘k’ values as quickly as the K-Nearest Neighbor algorithm might.

As the tables show, the ranking of relative algorithm performance differed from set to set. Several hyper parameters were tested for every algorithm, and the hyper parameters that yielded the best performance are shown. It is important to note that this process of testing did not yield a complete search of the hyper parameter space, thus it is possible that each algorithm may have better performance given different hyper parameters. It was observed that most algorithms tended to converge towards a peak for the performance metrics. The random nature of cross-fold validation and in the case of ENN creating a validation set would sometimes yield different performance results under the same hyper-parameters but nonetheless there were clear and strong trends between algorithm performance and hyper-parameters.

5. Summary

To summarize, the ‘No Free Lunch theorem’ continues to prove time and again why it holds up when analyzed across different data set/algorithm pairings. Certain algorithms performed certain tasks better than others, however no single algorithm proved to be the best at every task that it was given. Likewise the relative ranking of algorithm performance differed from data set to data set. Even across similar types of problems (ie regression and classification), the algorithms’ behavior was entirely dependent on the influence of the data set. As a result, the hypothesis was in fact proven correct; no single algorithm was shown to be the best for all types of problems. Understanding how and why the ‘No Free Lunch’ theorem holds true for algorithm design has proven to be invaluable, as is analogous in the real world, the right tool should be chosen for the right job.

6. Results (Begin on next Page)

Table 1: Machine Data Set Regression Results

Algorithm name	Data Set	Hyper-Parameters	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	Standard Deviations Away
K-Means	machine_data.csv	K=2	2125.123156	46.099058	0.297461256
K-Means	machine_data.csv	K=5	2833.456813	53.230225	0.343476203
K-Means	machine_data.csv	K=8	2836.123456	53.255267	0.343637793
KNN	machine_data.csv	K=2	1224.961617	34.999452	0.22583934
KNN	machine_data.csv	K=3	2578.224875	50.776224	0.327641384
KNN	machine_data.csv	K=6	2728.301779	52.233148	0.337042415
PAM Regress	machine_data.csv	K=2	2724.925548	52.200819	0.336833808
PAM Regress	machine_data.csv	K=3	3192.792185	56.504798	0.364605891
PAM Regress	machine_data.csv	K=4	3869.375941	62.204308	0.401382856

Table 2: Forest Fire Regression Results

Algorithm name	Data Set	Hyper-Parameters	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	Standard Deviations Away
K-Means	forestfires.csv	K=30	4174.986352	64.614134	1.01505465384357
K-Means	forestfires.csv	K=22	4176.542345	64.626174	1.01524378868981
K-Means	forestfires.csv	K=28	4178.456212	64.640979	1.01547637570543
KNN	forestfires.csv	K=4	3682.563545	60.684129	0.953316305948243
KNN	forestfires.csv	K=5	3706.201716	60.878582	0.956371053952864
KNN	forestfires.csv	K=3	3875.457522	62.253173	0.977965168660842
PAM Regress	forestfires.csv	K=45	4185.986305	64.699199	1.01639097062018
PAM Regress	forestfires.csv	K=43	4187.302436	64.709369	1.01655074163179
PAM Regress	forestfires.csv	K=44	4189.678148	64.727723	1.01683907635096

Table 3: White Wine Regression Results

Algorithm name	Data Set	Hyper-Parameters	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	Standard Deviations Away
K-Means	winequality-white.csv	K=28	0.98752	0.993742	0.807512563
K-Means	winequality-white.csv	K=25	1.02587	1.012853	0.823042108
K-Means	winequality-white.csv	K=12	1.3225	1.15	0.934487547
KNN	winequality-white.csv	K=2	1.502817	1.225894	0.996159148
KNN	winequality-white.csv	K=37	1.548284	1.244301	1.011116019
KNN	winequality-white.csv	K=31	1.54983	1.244922	1.011620706
PAM Regress	winequality-white.csv	K=22	1.068711	1.033785	0.840051319
PAM Regress	winequality-white.csv	K=23	1.069411	1.034123	0.840326388
PAM Regress	winequality-white.csv	K=24	1.07029	1.034548	0.840671669

Table 4: Red Wine Regression Results

Algorithm name	Data Set	Hyper-Parameters	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	Standard Deviations Away
K-Means	winequality-red.csv	K=14	1.44	1.2	1.126054268
K-Means	winequality-red.csv	K=6	1.512923	1.230009	1.154214398
K-Means	winequality-red.csv	K=7	1.7225	1.31244	1.231565999
KNN	winequality-red.csv	K=2	1.322235	1.149885	1.079027218
KNN	winequality-red.csv	K=3	1.426008	1.194156	1.120570167
KNN	winequality-red.csv	K=12	1.475221	1.214587	1.139742172
PAM Regress	winequality-red.csv	K=5	1.030892	1.015329	0.95276251
PAM Regress	winequality-red.csv	K=6	1.03588	1.017782	0.955064712
PAM Regress	winequality-red.csv	K=15	1.039776	1.019694	0.956859052

Table 5: Abalone Classification Results

Algorithm Name	Data Set	Hyper Parameters	0/1 Loss (Accuracy)
CNN	abalone_data.csv	K=22	0.266227
CNN	abalone_data.csv	K=15	0.262864
CNN	abalone_data.csv	K=18	0.261692
CNN	abalone_data.csv	K=21	0.261669
CNN	abalone_data.csv	K=17	0.259757
ENN	abalone_data.csv	validationSetFraction=0.059321—K=13	0.263823
ENN	abalone_data.csv	validationSetFraction=0.087138—K=22	0.26215
ENN	abalone_data.csv	validationSetFraction=0.191671—K=7	0.260951
ENN	abalone_data.csv	validationSetFraction=0.260488—K=14	0.260709
ENN	abalone_data.csv	validationSetFraction=0.237849—K=8	0.259536
K-means	abalone_data.csv	K=6	0.219061
K-means	abalone_data.csv	K=4	0.217614
K-means	abalone_data.csv	K=11	0.203506
K-means	abalone_data.csv	K=21	0.19918
K-means	abalone_data.csv	K=3	0.1945
KNN	abalone_data.csv	K=19	0.271725
KNN	abalone_data.csv	K=22	0.270538
KNN	abalone_data.csv	K=21	0.26694
KNN	abalone_data.csv	K=24	0.265015
KNN	abalone_data.csv	K=20	0.264305
PAM	abalone_data.csv	K=8	0.233185
PAM	abalone_data.csv	K=18	0.232233
PAM	abalone_data.csv	K=17	0.231034
PAM	abalone_data.csv	K=10	0.230555
PAM	abalone_data.csv	K=20	0.229841
PAM	abalone_data.csv	K=19	0.228637

Table 6: Car Classification Results

Algorithm	Data Set	Hyper Parameters	0/1 Loss (Accuracy)
CNN	car_data.csv	K=1	0.925928
CNN	car_data.csv	K=10	0.860566
CNN	car_data.csv	K=14	0.859383
CNN	car_data.csv	K=14	0.845487
CNN	car_data.csv	K=13	0.844885
ENN	car_data.csv	validationSetFraction=0.250748—K=1	0.884837
ENN	car_data.csv	validationSetFraction=0.445615—K=13	0.827527
ENN	car_data.csv	validationSetFraction=0.421136—K=14	0.821196
ENN	car_data.csv	validationSetFraction=0.353614—K=16	0.821162
ENN	car_data.csv	validationSetFraction=0.313243—K=21	0.819404
K-means	car_data.csv	K=18	0.847231
K-means	car_data.csv	K=7	0.84143
K-means	car_data.csv	K=7	0.84143
K-means	car_data.csv	K=22	0.814236
K-means	car_data.csv	K=22	0.814236
KNN	car_data.csv	K=1	0.951979
KNN	car_data.csv	K=3	0.935176
KNN	car_data.csv	K=19	0.934632
KNN	car_data.csv	K=18	0.931718
KNN	car_data.csv	K=2	0.931126
PAM	car_data.csv	K=8	0.842536
PAM	car_data.csv	K=9	0.8385
PAM	car_data.csv	K=12	0.837922
PAM	car_data.csv	K=11	0.837918
PAM	car_data.csv	K=7	0.835583

Table 7: Image Segmentation Classification Results

Algorithm	Data Set	Hyper Parameters	0/1 Loss (Accuracy)
CNN	segmentation_data.csv	K=1	0.82381
CNN	segmentation_data.csv	K=2	0.638095
CNN	segmentation_data.csv	K=3	0.628571
CNN	segmentation_data.csv	K=5	0.595238
CNN	segmentation_data.csv	K=4	0.566667
ENN	segmentation_data.csv	validationSetFraction=0.208549—K=1	0.77619
ENN	segmentation_data.csv	validationSetFraction=0.206086—K=5	0.761905
ENN	segmentation_data.csv	validationSetFraction=0.134956—K=3	0.752381
ENN	segmentation_data.csv	validationSetFraction=0.296886—K=7	0.742857
ENN	segmentation_data.csv	validationSetFraction=0.301114—K=6	0.738095
K-means	segmentation_data.csv	K=18	0.833333
K-means	segmentation_data.csv	K=12	0.761905
K-means	segmentation_data.csv	K=3	0.761905
K-means	segmentation_data.csv	K=19	0.757143
K-means	segmentation_data.csv	K=25	0.70562
KNN	segmentation_data.csv	K=1	0.842857
KNN	segmentation_data.csv	K=2	0.804762
KNN	segmentation_data.csv	K=8	0.8
KNN	segmentation_data.csv	K=7	0.795238
KNN	segmentation_data.csv	K=9	0.795238
PAM	segmentation_data.csv	K=1	0.804762
PAM	segmentation_data.csv	K=9	0.752381
PAM	segmentation_data.csv	K=12	0.747619
PAM	segmentation_data.csv	K=6	0.742857
PAM	segmentation_data.csv	K=7	0.742857

References

Naufal M Saad Mahboob Iqbal Mohd Faris Abdullah Waqas Rasheed Tuan AB Rashid Abdullah Mohd Yaqoob Bin Jafaar Abdul Qayyum, Aamir Saeed Malik. Cnn. https://www.tandfonline.com/doi/full/10.1080/01431161.2017.1296206, 2017.

Marko Bohanec. Car evaluation database. https://archive.ics.uci.edu/ml/datasets/Car+Evaluation, 1997.

Carla Brodley. Image segmentation database. https://archive.ics.uci.edu/ml/datasets/Image+Segmentation, 1990.

M.D. Jeng Chih-Tang Chang, Jim Z.C. Lai. Fast agglomerative clustering using information ofk-nearest neighbors. https://www.sciencedirect.com/science/article/pii/S0031320310003171/pdfft?md5=78324e4ab521016a9e1dbb7243468888&pid=1-s2.0-S0031320310003171-main.pdf, 2010.

Jankowski Norbert Grochowski Marek. Enn. https://link.springer.com/content/pdf/10.1007%2F978-3-540-24844-6_87.pdf, 2004.

F. Almeida T. Matos J. Reis. P. Cortez, A. Cerdeira. Wine quality. https://archive.ics.uci.edu/ml/datasets/Wine+Quality, 2009.

Aníbal Morais Paulo Cortez. Forest fires database. https://archive.ics.uci.edu/ml/datasets/Forest+Fires, 2007.

Jacob Feldmesser Phillip Ein-Dor. Computer hardware database. https://archive.ics.uci.edu/ml/datasets/Computer+Hardware, 1987.

Sam Waugh. Abalone data set. https://archive.ics.uci.edu/ml/datasets/Abalone, 1995.

David H Wolpert, William G Macready, et al. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Debo Cheng Ming Zong Shichao Zhang Zhenyun Deng, Xiaoshu Zhu. Knn. https://www.sciencedirect.com/science/article/pii/S0925231216001132, 2016.