Variations on k-Nearest Neighbors Algorithms

Logan Ladd logan.ladd@ecat1.montana.edu
Gianforte School of Computing
Montana State University
Bozeman, MT 59717-3880, USA

Asher Worley asher.worley@ecat1.montana.edu
Gianforte School of Computing
Montana State University
Bozeman, MT 59717-3880, USA

Editor: Logan Ladd

Abstract

This paper describes and analyzes the performance of five different models based on the k-nearest neighbor algorithm on a set of six different data sets, four of which contained continuous data, and three pf which required regression models. Our goal is to determine the best algorithm of those tested, as well as to demonstrate the power of non-parametric models by reducing the data. The models were tested using a ten-fold cross-validation method, with 10% of the data used for tuning purposes. The data was then evaluated using mean error and mean squared error for the regression models, and using mean standard error and accuracy for the categorical models. Our results found that, for the categorical models, the unmodified k-nearest neighbors model performed best with one exception, and that for the regression models there was little difference between the models, with one exception. The evidence provided is not enough to make any conclusions and further experimentation is necessary, though the trend found in these expiriments implies that, for categorical models, using the base k-nearest neighbor model is most likely best, as long as it is computationally practical, and for the regression models, using the most computationally efficient model is most likely best.

1. Introduction

The base k-nearest neighbor model is a simple model that can be surprisingly accurate when applied correctly. The model works by comparing how similar a test point is to other points nearby usually using a Euclidean metric space. This model can be very good, assuming the feature variables are each have equal weight in determining the correct classification or value of a point. However, therein lies the problem. Generally speaking, it can be difficult to pare down the dimensionality of the data since it isn't always possible to know which features are important. This is important to do, however, in order to avoid the dreaded curse of dimensionality, a phenomenon what describes the rarefaction of space as the dimensionality of a space increases. The sparsity that is introduced in the process causes the ranking process in the k-nearest neighbor model to change. Thus, it can be vital to know which features are important in the data.

Furthermore, another issue arises when creating a k-nearest neighbor model, namely that it can be computationally taxing when using large data sets. One way to address this is to pare down the number of points used in the model, only taking into consideration the points that are important for classification purposes. The process of paring down the points is a topic with great amounts of research. Our goal is to examine four of the many possible methods and test which works best on our data.

We hypothesise that a basic k-nearest neighbor model will generally perform better than any of the pared models examined.

The paring models that we intend to use are as follows: "*edited k-nearest neighbor*", "*condensed nearest neighbor*", "*k-means clustering*", and "*k-metoids clustering*".

1.1 Data Sets

We used six data sets in our experiment. The data sets had between 209 and 4177 data points, with a maximum of 17 missing attribute values. Of these six data sets, three required a regression model.

Data Set	# of Features	# of Missing Attribute Values	Model Type Needed
abalone[7]	4177	0	Categorical
glass[3]	214	0	Categorical
house-votes-84[6]	435	17	Categorical
segmentation[4]	2310	0	Regression
machine[2]	209	0	Regression
forestfires[1]	512	0	Regression

We first preprocessed the data by randomizing the order in which the data appears. Then we separated the data in order to perform a ten-fold cross-validation. We also created similarity matrices in order to be able to analyze the categorical featured.

1.2 Models

1.2.1 k-Nearest Neighbors

The k-nearest neighbors model works by taking the closest training points in space to a test point and determines the category of value that the test point should be based on these points. There are a few different ways to change the behavior of this model. Changing the number of neighbors that are being evaluated, k, and changing the weight that these points have are two of these ways. Another way would be to change the metric space. Most often, a Euclidean space is used, however abstract metric spaces may be used as well. In fact, some data sets will not function within a k-nearest neighbors model with a Euclidean metric space.[5] However, for our applications a simple Euclidean metric space was used. Specifically, the following distance function was used:

$$d(\vec{x},\vec{y}) = \sqrt{\sum_{j=0}^{i-1} x_j - y_j}$$

Where $\vec{x}, \vec{y} \in \mathbb{R}^i$.

1.2.2 Edited k-Nearest Neighbors

The edited k-nearest neighbors works just like the k-nearest neighbors model, except with fewer points. Of interest is the method of choosing which training points need to be edited out. There are many different methods currently being used, however the method we used was a implemented a stepwise backward elimination technique with the following steps: For each $x_i \in X$, where X is the dataset, classify x_i using the rest of the data. If the classification is incorrect, remove that point from the data set. Repeat this process with the entire dataset. This ends up clearing all the noise from the model that doesn't fit within the model. This can be advantageous if the data is particularly noisy, however it can also be dangerous if the data set is small, since most of the data would be considered noise by this process.

1.2.3 Condensed Nearest Neighbors

The condensed nearest neighbor model is similar to the edited k-nearest neighbor model, with the difference that, while the edited k-nearest neighbor model implemented a stepwise backward elimination technique, the condensed nearest neighbor model uses a stepwise forward selection process. The process works as follows: Start by initializing a set Z by selection a random testing point from the dataset X. Then, choose another random point from the dataset, x. Then find the closest point to in the set Z to the point x, $x^{'}$. If $x^{'}$ is classified differently then x, add x to Z, otherwise discard the point. Repeat the process until there are no more elements of X that have not been tested. This model is meant to remove irrelevant and outlying data points from the dataset and make the k-nearest neighbors model become more efficient.

1.2.4 k-Means Clustering

The k-means clustering model is similar to the condensed nearest neighbor model in that it attempts to minimize the effect of irrelevant and outlying data points. The way it attempts to achieve this is very different however. The model is created as follows: First, choose the number of clusters to be evaluated, k. Represent the center point of the cluster with a point and randomize those points positions within space. Then, add all the points from the data set. For each point, assign it to the closest cluster. Once all of the points have been assigned to a cluster, find the centroid of the cluster by finding the mean of all points assigned to a cluster, and move the point that represents that cluster to that centroid. Then unassign the data points and repeat the process until the centroids no longer move. In order to classify any points in the test set, find the closest centroid and find the most common classification of points belonging to that centroid, and assign that classification to the test point.

The k-means clustering model assumes that the data naturally clusters together based on classifications. Of note is that the model doesn't evaluate the classification of each cluster until testing is done. This means that the model would ignore outlying points and noise.

Also of note is that the model may not converge to a universal minima, but rather may converge to a local minima. This implies that it would be best to make several iterations of this model and test to see which is most efficient.

1.2.5 k-Medoids Clustering

The k-medoids clustering model is very similar to the k-means clustering and makes the same assumptions, namely that the data is clustered via classification. The main difference is that instead of finding the means of the clusters, we find the most central data point and use that as the center of the cluster. This method is known as partitioning around medoids. The process is as follows: Plot out all data points. Then, randomly assign k points as medoids of the clusters. From there, for each medoid, find the cost of that medoid. Namely, find σ, where:

$$\sigma = \sum_{i=0}^{j-1} D(k, x_i)$$

Where D is the distance function of the metric space, j is the number of data points assigned to the cluster, k is the medoid of the cluster, and x_i is the i'th data point in the medoid. Once this cost is found, compare against the cost associated with assuming other points within the cluster is the mediod. Choose the minimum cost, and recalculate the clusters. Repeat this process until the medoids do not change.

This model has many of the same advantages of the k-means clustering model, and is test data is evaluated in the same way. One of the major disadvantages of this model is that it's computation time is $O(n^2)$. The implication is that, while the end result of the model may reduce the number of points in the data set, it also isn't efficient with large data sets.

1.3 Categorical vs. Regression

The above model descriptions assume a categorical classification, however in reality some data sets require a regression classification. The main difference between the two classification types is that, while categorical classifiers output a discrete value, regression classifiers output a continuous value. In order to deal with regression classifiers, it is important to note that the k-nearest neighbors algorithms do not assume linearity, so it isn't enough to simply weight data points in a linear way. The method we used was to use a Gaussian kernal. The kernal is defined as follows:

$$f(x) = \frac{1}{\sqrt{(2\pi)}} e^{\frac{-||u||}{2}}$$

Where u is the vector from the k'th point to the point being tested. So, $f(x)k$, where k is the value of the point being evaluated, is the output value of the model. The implication of this method is that edited k-nearest neighbor and condensed k-nearest neighbor models will not function on data which requires regression classification. For the simple k-nearest neighbors model, the value is found by averaging the values found by $f(x)k$ for each k. For the k-means and k-medoids clustering models, the value is found using the nearest centroid or medoid as the point of origin.

1.4 Hyperparameter Tuning

The main hyperparameter for tuning categorical models is k. The standard is to choose $k = \sqrt{n}$, where n is the number of data points in the data set. We started with this as a basis, and we tested everywhere from -20% $+k$ to 20% $+k$ of this basis in 10% intervals.

For regression models, we needed to tune the number of clusters. We chose to start with the number of points remaining when the data was run through the edited k-nearest neighbors algorithm. We assigned V to this number. We then tested everywhere from -20% $+V$ to 20% $+V$ of this basis in 10% intervals. For both of these hyperparameters, we chose the values that resulted in the best results.

1.5 Results

After performing the experiments, we found that each model performed very differently than expected.

Data Set	KNN	ENN	CNN	K-Means	K-Medoids
Glass Accuracy	78.4	77.7	43.5	7.9	67.9
Glass MSE	11.2	10.9	954.3	7733.3	12.4
Image Segmentation Accuracy	83.8	84.3	56.3	14.3	82.4
Image Segmentation MSE	2.1	2.0	5.9	56.4	2.1
House-Votes-84 Accuracy	30.9	33.9	31.3	0.1	0.1
House-Votes-84 MSE	238.6	363.2	236.9	6334.5	6334.5
Forest Fires ME	4238.2	N/A	N/A	4234.7	4229.6
Forest Fires MSE	12.7	N/A	N/A	12.7	12.6
Abalone ME	14.2	N/A	N/A	14.2	14.2
Abalone MSE	199.8	N/A	N/A	7056.9	7056.9
Machine ME	0.5	N/A	N/A	0.5	0.5
Machine MSE	2.1	N/A	N/A	2.1	2.1

We found that, for the data sets that required categorical classification, the clustering models generally performed worse than the other models. We found that, among the other models, there is no discernible pattern as to which works best. For the data sets that required regression classification, the differences were, generally, extremely minimal.

Of note is that the clustering models performed worst on the House Votes data set. The accuracy and the mean squared difference was about 30 times worse than the nearest neighbors models. The implication is that the data is not clustered at all. However, even the nearest neighbors models performed very poorly, reaching less than 35% accuracy. When considering what the data represents, this implies that there is significant variation in the way politicians vote, and, based on the data, they rarely vote on party lines.

Also of note is the difference between the accuracy of the k-means model and the k-medoids model on the Glass data set. The difference is striking, considering that the k-means model had a 7.9% accuracy compared to the 67.9% accuracy achieved by the k-medoids model. It is possible that this difference is a result of the random initialization process of these models, and that the k-medoids model found a deep local minima, whereas the k-medoids model found a shallow local minima. More testing is required to make any confident statements.

2. Summary

The variations present in the data imply an erratic nature to which method works best. It is likely that each model is more useful for some data sets, but less useful for others. Unfortunately, this class of models appears to have so much variation that it is difficult to say anything about them with any kind of certainty. It is interesting to note that the basic k-nearest neighbors model didn't perform the best among the other models in both the Image Segmentation and the House Votes data sets. However, in the Glass data set, it did perform the best. Thus, nothing can be said, except perhaps that the performance is purely based on the data set itself.

Considering that the data sets that required a regression classification method all performed generally the same across the models, with one exception, that exception being the Abalone data set, which performed differently under the basic k-nearest neighbors model than the clustering models. According to the mean squared difference metric, the k-nearest neighbor model performed approximately 35 times better. Also of note though, is that by the mean difference metric the models performed the same. This implies that way the data was distributed was unique, and that this data should be considered an outlier.

Based on the data, there is not enough evidence to reject the hypothesis. However, the data was also erratic enough that the hypothesis cannot yet be accepted. Ultimately, more testing must be done to make any kind of solid conclusion.

References

[1] Paulo Cortez and Anibal Morais. *UCI Forest Fires.* 2007.

[2] Phillip Ein-Dor and Jacob Feldmesser. *UCI Relative CPU Performance Data.* 1987.

[3] B. German. *UCI Glass Identification Data Set.* 1987. URL: `https://archive.ics.uci.edu/ml/datasets/Glass+Identification`.

[4] Vision Group. *UCI Image Segmentation Data.* 1990.

[5] Rodrigo Paredes and Gonzalo Navarro. *UCI Practical Construction of k Nearest Neighbor Graphs in Metric Spaces.* URL: `https://www.dcc.uchile.cl/TR/2005/TR_DCC-2005-006.pdf`.

[6] *UCI Congressional Voting records.* 1984. URL: `https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records`.

[7] Dr. Sam Waugh. *UCI Abalone Data.* 1995.