K-Nearest Neighbor and Regressor

Tarney, Brandon
Johns Hopkins University Engineering Professionals CS 605.649

Table of Contents

Abstract

The purpose of this assignment was to implement k-nearest neighbor and regressor algorithms. The effect of varying the k value is explored as is the impact of reducing the training data set. Additionally, the impact of the type of data analyzed is analyzed. Finally, conclusions are drawn from the performance of the existing models and improvements are considered.

All code, input data, and results can be found here:
https://github.com/1amBulletproof/machine_learning_p3_k_nearest_neighbor

Keywords: k-nearest neighbor, k-nearest regressor, radial basis function, Machine Learning

K-Nearest Neighbor and Regressor

Experiment Summary

Problem Statement

Introduction to nonparametric machine learning algorithms through experimentation with k-nearest neighbor algorithms.

Hypothesis

This experiment will examine the affect of varying k values on the k-nearest neighbor algorithms. The expectation is a k value of one will perform very well. Additionally, increasing the value of k is expected to increase the performance of models with ample training data and examples of each class but decrease performance with either or both of those cases are not true (ultimately at some arbitrarily large k value all k-nearest neighbor algorithms will likely decrease in performance due to the incorporation of dissimilar points, however). Finally, the condensed k-nearest neighbor should perform nearly as well as the k-nearest neighbor algorithm despite using far fewer samples.

Background

K-Nearest Neighbor

Implementing k-nearest neighbor algorithm is fairly straight forward, especially if you have implemented distance-based algorithms prior. First of all, it is a nonparametric algorithm, meaning it is lazy and does not do an actual model training before evaluation/test time. At query time, effectively, the closest k points to the given point is selected and then these results are combined to form your classification or regression result. For regression data the results are combined via averaging. For classification data, the results are combined via majority vote. (Alpaydin, E. 2010)

Condensed K-Nearest Neighbor (Regressor)

Condensed k-nearest neighbor is identical to k-nearest neighbor with one important exception: it has a a training step. The training step performed for condensed k-nearest neighbor is similar to stepwise forward selection. Starting with an empty set of input points, you being to accumulate points which are necessary to properly classic each point in the training data. The key here is any point which is *not* required to properly classify a point is *not* included in the final set of data. The result is condensed k-nearest neighbor typically uses a fraction of the original sample data for test which can greatly speed-up the evaluation step with a minimal cost to performance. (Blank, DATE)

Experiment Details

Experimental Approach

This experiment covers the entire process of creating and applying a machine learning algorithm, not simply implementing the algorithm in isolation. Overall, this experiment can be broken down into several key steps:

Input Data and Data Pre-Processing

Four data sets were used for this experiment, all of which can be found in the references section. There was a wide variance in these data sets so all of it was normalized as well, though the

algorithms were run on on both normalized and original data. Finally, sometime meaningless features were removed from data sets and the syntax was standardized to be comma-separated.

Developing the K-Nearest Neighbor algorithm

The k-nearest neighbor (KNN) algorithm is nonparametric and therefore fairly simple to implement. The training step simply sets the training data and does nothing else. The evaluation step relies on a distance calculation to determine the KNN points. In this case, Euclidean distance was chosen due to its simple and robust performance between two vectors. Another important design detail of KNN is the method of determining the classification or regression value of the set of chosen k points. For this experiment, a majority vote was used to determine classification and an average was used for regression data.

Developing the Condensed K-Nearest Neighbor algorithm

Condensed k-nearest neighbor (CKNN) algorithm involves a training step which is effectively feature selection via KNN! Therefore, this implementation is a subclass of the KNN model which allows it to reuse the evaluation step, which is identical. Furthermore, the necessary implementation of a distance function and majority vote function is available via inheritance. The data for training is considered in random order so the resulting training data is not identical on every run.

Developing the five-fold cross-validation

In order to properly validate model results across an entire data set five-fold cross-validation (5-FOLD) was implemented. 5-FOLD divides a data set into five equally sized groups, creates a training data set from four of the five, and creates a test data set from the final group. The process of creating training and test data sets is repeated five times so that each group is tested once. While 5-FOLD alone works well for regression data sets, it can result in poor training or testing data sets for classification problems where equal representation of each class is necessary in each group. Therefore, in this experiment 5-FOLD was modified to create groups with equal class representation as well.

Running and Validation

While there are many files included in this experiment, there is only which runs the entire experiment: “run_model_with_five_fold_cross_validation.py”. This script is created in python 3 and will allow the user to run a model (select CKNN or KNN) with a given k number on the assumed data groups in files data_0, data_1, data_2, data_3, data_4. “split_data.py” will allow a user to split their data randomly or with equal class representation in order to support the final experiment. All of the source code is in folder “src”, all of the data is in folder “data” with sub-directories for preprocessed data and split data (into five-fold groups). Finally, all of the results are in the “results” folder where they are grouped by data set. Important to note, any reference to “normal” in the results or data indicates the data used was normalized. The lack of any such signifier in the file name indicates it was *not* normalized.

Modifications and Tweaks

The main two modifications to the models are the k value and decision algorithm. Both can be varied via command line input. Additionally, there is normalized and original data available for every data set. Normalized data may perform better, assuming that the largest values do not have

an overwhelming correlation to the result. Hamming distance as an alternative to Euclidean distance was experimented-with as well, but the results were worse than Euclidean distance.

Results, Analysis and Conclusions

Overall, the CKNN and KNN performed well for classification data sets when there was ample data with many inputs for each class. When the data had few examples for a given class, a small overall sample size, or especially when there were many zero values the result was less positive. As can be seen in figures 1-4 the k value was increased from one to seven for every data set. The ecoli data set, which had easily the best performance of the four data sets, showed improved performance as the k value increased. This was expected and part of the hypothesis because incorporating more *good* examples into your decision algorithm is possible in this data set. Interestingly, the normalization process did not appear to meaningfully increase performance, but once again, this makes some sense since the original data was fairly even. The other classification data set, image segmentation, showed likewise good performance with a small k value. As the k value increased the condensed model performance dropped drastically, likely because there are so few training samples in the condensed model that increasing the k value will incorporate uncorrelated examples into the decision algorithm.

The regression data sets proved to be much more challenging for the KNN. The computer hardware data set showed worse performance as the k value increased likely due to the small overall data set size and poor distribution of values. The forest fires data set performed as expected with more input data: performance increased with the value of k. It is important to note the value of average mean squared error used for these data sets is completely relative and therefore should not be compared directly between normalized and unnormalized data.

The CKNN typically produced a training data set of nearly one fourth the size of the given training data (i.e. what was used for KNN). This is impressive and resulted in much better compute times, at typically not much performance lost. With small k values, the CKNN performance was typically within 10% of the KNN. As is shown in the figures below, when the k value increased the CKNN performance decreased rapidly due to a lack of correlated inputs to add to the decision algorithm. This is due to the somewhat greedy nature of the local optimum search performed by CKNN which effectively seeks to find one and only one best point to represent an entire vornoi area. This means when you incorporate more than one value in the decision algorithm, there will be *more* error. This can be mitigated via the use of distance-based weights, but the error will undoubtedly still increase.

While 5-FOLD could result in longer run times for algorithms which take a long time to train on the given data but it does not add much runtime to this experiment thanks to the *lazy* nature of k-nearest networks which effectively do all calculations at query time.

Summary

K-nearest neighbor algorithms are simple to implement, do not require model training, and perform well in certain situations. The best k value will depend on the variance, distritbution, correlation, and abundance of your input data. A k value of one is often a good starting point and will typically result in performance within nearly 10% of the maximum performance and it can actually be the optimal value. A condensed nearest neighbor algorithm can perform nearly as well as a KNN at a fraction of the computational cost, however, caution should be taken to ensure a small k value is used.

References

Alpaydin, E. (2010). Introduction to machine learning (3rd ed.). Cambridge, MA: The MIT Press. ISBN-10: 0262028182; ISBN-13: 978-0262028189.

Cortez, P. Morais, A. (2007). *UCI Machine Learning Repository: Forest Fires Data Set.* [online] Archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/Forest+Fires [Accessed 4 Oct. 2018].

Hart, P. (1968). *The Condensed Nearest Neighbor Rule*. IEE Transactions on Information Theory 18. 515-516. Doi:10.1109/TIT.1968.1054155

Nakai, K. (1996). *UCI Machine Learning Repository: Protein Localization Sites Data Set.* [online] Archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/Ecoli [Accessed 4 Oct. 2018].

Ein-Dor, P. Fedlmesser, J. (1987). *UCI Machine Learning Repository: Relative CPU Performance Data Set.* [online] Archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/Computer+Hardware [Accessed 4 Oct. 2018].

Vision Group. (1990). *UCI Machine Learning Repository: Image Segmentation Data Set.* [online] Archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/Image+Segmentation [Accessed 4 Oct. 2018].

Tables & Graphs

Figure 1: E.coli data set analysis

Resulting 5-fold cross validation average accuracy				
k value	knn	knn_normal	condensed_knn	condensed_knn normal
1	80	80	72	68
3	84	83.5	78	77
5	85	85	81	79
7	87	86	82	76

knn train data size: 269

condensed knn train data size avg: 70

Figure 2: image segmentation *data set analysis*

Resulting 5-fold cross validation average accuracy				
k value	knn	knn_normal	condensed_knn	condensed_knn normal
1	76	86	74	81
3	77	86.5	64	67
5	78	86.5	49	57
7	79	87	40	40

knn train data size: 168

condensed knn train data size avg: 50

Figure 3: computer hardware data set analysis

	Resulting 5-fold cross validation average mean squared error	
k value	knn	knn_normal
1	3180	0.0045
3	5293	0.0047
5	6858	0.0058
7	8076	0.006
Train data size:		164

Figure 4: forest fires data set analysis

	Resulting 5-fold cross validation average mean squared error	
k value	knn	knn_normal
1	11075	0.011
3	5750	0.0055
5	4766	0.0047
7	4702	0.0042
train data size:		412