JMLR: Workshop and Conference Proceedings 1:1–5, 2016 Submitted 10/9/2016; Published 10/9/2016

Introduction to Machine Learning - EN 605.449.81 - Lab 3

Max Henry MHENRY22@JHU.EDU

Abstract

In this lab, I implemented the k-nearest neighbor algorithm and its variants and a radial basis function network. Variants include a condensed k-nearest neighbor algorithm, a k-means influenced k-nearest neighbor, and a k-means influenced radial basis function network. In general, k-nearest neighbor algorithms outperformed their radial basis network counterparts on most classification and regression data sets tested. This suggests more tuning is needed to get the best performance from the radial basis function network.

Keywords: k-means, k-nearest neighbor, radial basis function network

1. Problem Statement

Given two classification data sets and two regression data sets, we were tasked with implementing k-nearest neighbor, condensed k-nearest neighbor, k-means-fed nearest neighbor, radial basis function network, and k-means-fed radial basis function network. Since these algorithms all have tunable parameters, thought was given to have to properly adjust those parameters to obtain the best performance. I hypothesize that the radial basis network will outperform the k-nearest neighbor algorithms for all data sets due to it being a universal approximater.

2. Algorithms

2.1. k-nearest Neighbor

k-nearest neighbor is a non-parametric algorithm that determines the closest k training points to a query point to classify or approximate the query point's value. A majority voting scheme can be used to determine a query point's class label or a simple average scheme can be used to determine a query point's function output. Distance determines a query points nearest neighbor's and Euclidean distance is used in these experiments but other metrics are available. k-nearest neighbor is considered a lazy learner because no learner is done until a query point is fed into the system. This means during the training phase training data is merely saved and not searched until the prediction stage. In the naive implementation, all training points need to be checked for distance from the query point, however there is a way to condense the training set so the same performance may be achieved with a subset of the original training set. In this scheme, the training set is iterated through, distances are calculated to each training instance in the subset. If the given training instance is already classified correctly by its neighbor in the subset then the given training instance is thrown out. Another way to simplify the training set is to use k-means to find prototypes for the given training set and these prototypes are used in the prediction stages. Both of these training set optimizations are used in these experiments.

2.2. Radial Basis Function Network

The radial basis function network is a simple three layer neural network. The first layer is the feature layer, one node for each feature. The second layer is comprised of n nodes whose outputs are the outputs of radial basis functions and n is the number of training examples (in most cases some k less than n is used instead), called hidden nodes. The implementer has a choice of using different radial basis functions for each hidden node. In addition, each radial basis function needs to define a center to anchor the function and a spread parameter, which defines how wide a particular basis functions influence is. In these experiments, a Gaussian radial basis function is used for each node but the spreads are allowed to vary. The spreads were determined by calculating two times the average distance between the the center of the basis function and all other basis functions. To determine the center of these basis functions, two approaches are followed in these experiments. The first approach is to naively use each training point as a center and the second approach is to cluster the training data and use the prototypes return from k-means as the centers. Finally, the last layer of the network is the output layer. The output of the layer is determined by the dot product of a vector of weights, one for each hidden node, by the vector outputs from the hidden nodes. To learn these weights, batch update gradient descent is used. An error function of squared error is used in these experiments. In the case of classification, an activation function of $\frac{1}{1+e^{-u}}$ is used where u is the dot product of the weight vector and the hidden node outputs.

3. Experimental Approach

3.1. Data Cleaning

All categorical values in the data sets were translated to integer values. In addition, the data was normalized for one run of the radial basis network but results were discouraging so the normalized data was not used in further testing. In the case of a multi-class problem, the original data set was broken down into 0/1 class problems.

3.2. Experiments

Four data sets were used in these experiments, two regression (fire and computer hardware) and two classification (image and ecoli). As mentioned, k-nearest neighbor and radial basis function networks have a few tunable parameters. In order to tune these parameters, a number of experiments were run starting with tuning the k of the original k-nearest neighbor. Next the k of k-means was determined, using the best k from the k-nearest neighbor experiment for each data set. Separately, condensed k-nearest neighbor was run and the radial basis function networks were trained, one a randomly chosen 10% subset of the training set to serve as the center of the hidden nodes and one with a condensed training set based on the prototypes returned by k-means.

In all experiments, the best values of tunable parameters were chosen via 10-fold cross validation. The k‘s

4. Results

Table 1: Best k's

Data sets	k of K-means	k of k-nearest Neighbor
CPU	18	1
Fire	6	96
Ecoli (cp)	45	6
Ecoli (im)	46	6
Ecoli (imS)	5	6
Ecoli (imL)	5	6
Ecoli (imU)	5	11
Ecoli (om)	5	1
Ecoli (omL)	15	1
Ecoli (pp)	49	11
Image (brickface)	5	1
Image (sky)	41	1
Image (foliage)	3	1
Image (cement)	19	1
Image (window)	35	1
Image (path)	21	6
Image (grass)	19	1

Table 2: k-NN Results

Data sets	Original k-NN	Condensed k-NN	K-means-fed
CPU	42394.00	N/A	43192.289856
Fire	210733.9978	N/A	209949.129985
Ecoli (cp)	0.960606	0.775758 (k = 1)	0.951515
Ecoli (im)	0.915152	0.787879 (k = 1)	0.863636
Ecoli (imU)	0.930303	0.893939 (k = 1)	0.893939
Ecoli (om)	1.00	1.00	1.000000
Ecoli (pp)	0.954545	0.845455	0.857576
Image (brickface)	0.923810	0.857143	0.857143
Image (sky)	1.00	0.857143	0.990476
Image (foliage)	0.914286	0.857143 (k = 16)	0.861905
Image (cement)	0.952381	0.857143	0.861905
Image (window)	0.90	0.857143 (k = 46)	0.861905
Image (path)	0.990476	0.857143	0.861905
Image (grass)	0.995238	0.857143	0.876190

5. Conclusions

As can be seen in the tables above, k-nearest neighbor outperforms the radial basis function network on every data set by a large margin. A major reason this might be the case is the

Table 3: RBN Results

Data sets	Original Radial Basis Function Network	K-means-fed Radial Basis Function Network
CPU	495830.815545	497864.672605
Fire	209911.459624	219498.854013
Ecoli (cp)	0.438095	0.457143
Ecoli (im)	0.533333	0.515152
Ecoli (imU)	0.600000	0.006061
Ecoli (om)	0.457576	0.106061
Ecoli (pp)	0.484848	0.357576
Image (brickface)	0.438095	0.457143
Image (sky)	0.428571	0.519048
Image (foliage)	0.552381	0.366667
Image (cement)	0.428571	0.314286
Image (window)	0.561905	0.419048
Image (path)	0.390476	0.714286
Image (grass)	0.480952	0.552381

calculated spread for each Gaussian basis function was too small, too large, or overlapped with neighboring basis functions too much. In future work, it would be interested to try different calculations for the spread and observe their effect.

6. Summary

In this lab, I implemented and tested k-nearest neighbor and radial basis function network algorithms and some of their variants. Testing was done using regression and classification data sets and revealed the surprising result that the k-nearest neighbor algorithms always outperformed their radial basis network counterpart.

7. References

[1] Beyer, Kevin, et al. "When is nearest neighbor meaningful?." International conference on database theory. Springer Berlin Heidelberg, 1999.

[2] Gowda, K. Chidananda, and G. Krishna. "The condensed nearest neighbor rule using the concept of mutual nearest neighborhood." IEEE Transactions on Information Theory 25.4 (1979): 488-490.

[3] Chen, Sheng, Bernard Mulgrew, and Peter M. Grant. "A clustering technique for digital communications channel equalization using radial basis function networks." IEEE Transactions on neural networks 4.4 (1993): 570-590.

[4] Du, Ke-Lin, and M. N. S. Swamy. "Radial basis function networks." Neural Networks and Statistical Learning. Springer London, 2014. 299-335.

[5] UCI machine learning repository: Ecoli data set. (1996, September 1). Retrieved October 9, 2016, from https://archive.ics.uci.edu/ml/datasets/Ecoli

[6] UCI machine learning repository: Image segmentation data set. (1990, November 1). Retrieved October 9, 2016, from https://archive.ics.uci.edu/ml/datasets/Image+Segmentation

[7] UCI machine learning repository: Computer hardware data set. (1987, October 1). Retrieved October 9, 2016, from https://archive.ics.uci.edu/ml/datasets/Computer+Hardware

[8] UCI machine learning repository: Forest Fires data set. (2008, February 29). Retrieved October 9, 2016, from https://archive.ics.uci.edu/ml/datasets/Forest+Fires