COMP 307 Assignment 1

Daniel Braithwaite

April 6, 2016

1 Nearest Neighbor Method

1.1 Class Labels For Test Data

The K1 classifier has an accuracy of 90.7% on the test data set. **See Appendix A for class label data**

1.2 Comparing K1 And K3

With this training and test data using both K1 and K3 classifiers I find that the K3 classifier yields slightly better results with a accuracy of 93% compared to the K1 classifier that has 90.7%. This extra accuracy is at little cost to efficiency. One might argue that we are having to inspect more neighbors, increasing the time to compute the classification. But this extra time is negligible when comparing K1 and K3 as the algorithm for computing the neighbors is order $O(mnlog(n))$, where m is the number of test instances and n is the number of training instances. However with large amounts of training and testing data increasing K could have a noticeable effect on performance.
Inspecting where the classifier was failing identified some interesting features of the data. The classifier never wrongly classifiys an instance as 'Iris Setosa' nor does it ever assign an instance of 'Iris Setosa' the wrong class. It only mixes up the two classes 'Iris-versicolor' and 'Iris-virginica'. This indicates that the region containing all the instances belonging to 'Iris Setosa' is disjoint from the regions containing 'Iris-versicolor' and 'Iris-virginica'. Not only this but it also shows that the regions which contain instances of class 'Iris-versicolor' or 'Iris-virginica' intersect.

1.3 Nearest Neighbor Classification Discussion

1. **(+) Simple:** The algorithm is very simple and is easy to implement
2. **(+) Works On Nonlinear Data:** The data doesn't have to be linearly separable for this classification method to work.
3. **(-) Need To Find K:** We need to find for which value of K this classification method works optimally for a given problem.
4. **(-) Slow For Large Data:** The way the I implemented the algorithm for classifying test instances involves sorting all n elements of the training data, then taking the first k elements. This is to be repeated for each of the m instances to classify. We therefor get the order cost of

this algorithm to be $O(m(nlog(n)+n)) = O(mnlog(n))$. If a KD-Tree was used to store the training data then we could reduce this cost to $O(m(log(n)+n) = O(mn)$ which is a considerable improvement but on large training and testing data is still costly. One side effect of using the KD-Tree is that it would take longer to train the classifier as the instances have to be inserted into the tree

5. **(-) The Data Can Be Skewed:** In your training data maybe there is 100 data points and we know that each can belong to one of three classes. If we had 95% of the data points belonging to class 1 then our data would be skewed to be giving a classification of class 1.

1.4 K-Fold Cross Validation

The general idea for K-fold cross validation is as follows

1. Split the input data into k equal chunks
2. For each chunk
   (a) Treat the current chunk as test data and the rest of the chunks as training data
   (b) Train the classifier with the training data and run it over the testing data
3. Average the results from each run to produce an estimation of accuracy

So for the current example we would split our iris data into 5 equal chunks and for each chunk train our nearest neighbor classifier with the remaining data and test it on our current chunk. Lastly we would be able to average the accuracy from the 5 different runs to get our final estimation of how accurate the classifier is.

1.5 Clustering Problem

The clustering method to use would be K-Means Clustering. The algorithm for this is as follows

1. Place K random centroids.
2. For each data point assign it to the closest centroid
3. For each centroid set the new location to the mean location of all points assigned to it

4. Return to step 2 until convergence

For when we are wanting to create 3 clusters, we set K = 3 i.e. creating 3 centroids.
Using this algorithm on the iris training data we get mostly inaccurate results. We would expect to get 3 groups of size 25 as this is how our data is split up, how ever with this method we rarely get close to that
Another possible method of choosing the initial centroid locations would be to randomly select 3 data points and use them. This alternative method while seeming to provided more consistently accurate results also sometimes provides poor groupings.

2 Decision Tree Learning

2.1 Decision Tree Results

Over full test data the decision tree has an accuracy of 81% which is slightly worse than the accuracy of the baseline classifier which has 85% accuracy.
Out of the 27 test instances there are 5 incorrectly classified. The miss classifications where evenly split so don't favor ether of the classes.

Over split test data The average accuracy over the 10 trials is 78% giving us a 3% difference from the accuracy over the full test data. Not enough to be significant.

2.2 Pruning Discussion

2.2.1 Pruning Techniques

Reduced error pruning involves iterating through the tree (starting at the leaves) and replacing each node with the base classifier (most popular class). If the accuracy of the decision tree is unchanged then the modified tree is kept.

2.2.2 Reducing Accuracy On Training Data

When generating the decision tree the only information the algorithm has is that of the training data. Therefor the algorithm generates a tree that models the data used to train it. The process of pruning the tree is to make it more general and less specific to the training data which is why after pruning the tree could perform worse on the data used to train it.

2.2.3 Improving Accuracy On Testing Data

Before pruning the decision tree is tailored to the training data. By pruning and making the decision tree more general it might have a better chance of correctly classifying a problem instance it hasn't seen before.

2.3 Impurity Measure Discussion

We have list of instances L (each instance has a class in the set $N_1, ..., N_k$) and a function f such that $f(L)$ gives us the impurity of the set L. For f to be valid the following 3 conditions must be met

1. $f(L) = 0$ if and only if all instances belong to one class (i.e. L is pure)
2. f is at max if all instances equally distributed between k classes
3. f must be continuous

Define $f(L)$ as

$$f(L) = P(N_1) * ... * P(N_k)$$

Assume that $k > 2$ then take the first condition. Assume for our list L there is some $i < k$ such that no instance in L has class N_i and every other class has at least one instance belonging to it, then we have, where n_j = number of instance belonging to N_j

$$\frac{n_1 * ... * n_i * ... * n_k}{(n_1 + ... + n_k)^k} = \frac{n_1 * ... * 0 * ... * n_k}{(n_1 + ... + n_k)^k} = \frac{0}{(n_1 + ... + n_k)^k} = 0$$

So our function gives output 0 even though L isn't pure. So f doesn't meet all the conditions set out above so isn't a good measure when $k > 2$.

3 Perceptron Learning

3.1 Perceptron Performance

The perceptron classifier converges on a solution in under 100 epochs with the number of features set to 75

3.2 Data Discussion

Using the training data to evaluate the peformance of the perceptron is not a great mesure because the peceptron could be trained to work perfectly on the the training data but when given examples it hasnt seen before it cant classify them correctly which we cant check for if we are using the training data to validate the peformance of the perceptron.
I created some extra data to test the peformance of the perceptron classifier the results of which where interesting. Overall the data contained 10 instances split evenly between positive and negative examples. The accuracy of the classifier on this data was 70%. Upon further inspection the classifier correctly identified all the positive examples and just had trouble with negative examples.
An explanation for why the classifier has trouble with negative examples is because the space of negative instances is much larger than the space of positive instances. And if the negative instances in the test data are very different from the ones used while testing then this could cause the inaccuracy.

4 Appendix A (Nearest Neighbor Output)

Actual Class	Sepal Length	Sepal Width	Petal Length	Petal Width	Predicted Class
Iris-setosa	5.0cm	3.0cm	1.6cm	0.2cm	Iris-setosa
Iris-setosa	5.0cm	3.4cm	1.6cm	0.4cm	Iris-setosa
Iris-setosa	5.2cm	3.5cm	1.5cm	0.2cm	Iris-setosa
Iris-setosa	5.2cm	3.4cm	1.4cm	0.2cm	Iris-setosa
Iris-setosa	4.7cm	3.2cm	1.6cm	0.2cm	Iris-setosa
Iris-setosa	4.8cm	3.1cm	1.6cm	0.2cm	Iris-setosa
Iris-setosa	5.4cm	3.4cm	1.5cm	0.4cm	Iris-setosa
Iris-setosa	5.2cm	4.1cm	1.5cm	0.1cm	Iris-setosa
Iris-setosa	5.5cm	4.2cm	1.4cm	0.2cm	Iris-setosa
Iris-setosa	4.9cm	3.1cm	1.5cm	0.1cm	Iris-setosa
Iris-setosa	5.0cm	3.2cm	1.2cm	0.2cm	Iris-setosa
Iris-setosa	5.5cm	3.5cm	1.3cm	0.2cm	Iris-setosa
Iris-setosa	4.9cm	3.1cm	1.5cm	0.1cm	Iris-setosa
Iris-setosa	4.4cm	3.0cm	1.3cm	0.2cm	Iris-setosa
Iris-setosa	5.1cm	3.4cm	1.5cm	0.2cm	Iris-setosa
Iris-setosa	5.0cm	3.5cm	1.3cm	0.3cm	Iris-setosa
Iris-setosa	4.5cm	2.3cm	1.3cm	0.3cm	Iris-setosa
Iris-setosa	4.4cm	3.2cm	1.3cm	0.2cm	Iris-setosa
Iris-setosa	5.0cm	3.5cm	1.6cm	0.6cm	Iris-setosa
Iris-setosa	5.1cm	3.8cm	1.9cm	0.4cm	Iris-setosa
Iris-setosa	4.8cm	3.0cm	1.4cm	0.3cm	Iris-setosa
Iris-setosa	5.1cm	3.8cm	1.6cm	0.2cm	Iris-setosa
Iris-setosa	4.6cm	3.2cm	1.4cm	0.2cm	Iris-setosa
Iris-setosa	5.3cm	3.7cm	1.5cm	0.2cm	Iris-setosa
Iris-setosa	5.0cm	3.3cm	1.4cm	0.2cm	Iris-setosa
Iris-versicolor	6.6cm	3.0cm	4.4cm	1.4cm	Iris-versicolor
Iris-versicolor	6.8cm	2.8cm	4.8cm	1.4cm	Iris-versicolor
Iris-versicolor	6.7cm	3.0cm	5.0cm	1.7cm	**Iris-virginica**
Iris-versicolor	6.0cm	2.9cm	4.5cm	1.5cm	Iris-versicolor
Iris-versicolor	5.7cm	2.6cm	3.5cm	1.0cm	Iris-versicolor
Iris-versicolor	5.5cm	2.4cm	3.8cm	1.1cm	Iris-versicolor
Iris-versicolor	5.5cm	2.4cm	3.7cm	1.0cm	Iris-versicolor
Iris-versicolor	5.8cm	2.7cm	3.9cm	1.2cm	Iris-versicolor
Iris-versicolor	6.0cm	2.7cm	5.1cm	1.6cm	**Iris-virginica**
Iris-versicolor	5.4cm	3.0cm	4.5cm	1.5cm	Iris-versicolor
Iris-versicolor	6.0cm	3.4cm	4.5cm	1.6cm	Iris-versicolor
Iris-versicolor	6.7cm	3.1cm	4.7cm	1.5cm	Iris-versicolor
Iris-versicolor	6.3cm	2.3cm	4.4cm	1.3cm	Iris-versicolor
Iris-versicolor	5.6cm	3.0cm	4.1cm	1.3cm	Iris-versicolor
Iris-versicolor	5.5cm	2.5cm	4.0cm	1.3cm	Iris-versicolor
Iris-versicolor	5.5cm	2.6cm	4.4cm	1.2cm	Iris-versicolor
Iris-versicolor	6.1cm	3.0cm	4.6cm	1.4cm	Iris-versicolor
Iris-versicolor	5.8cm	2.6cm	4.0cm	1.2cm	Iris-versicolor
Iris-versicolor	5.0cm	2.3cm	3.3cm	1.0cm	Iris-versicolor
Iris-versicolor	5.6cm	2.7cm	4.2cm	1.3cm	Iris-versicolor

Iris-versicolor	5.7cm	3.0cm	4.2cm	1.2cm	Iris-versicolor
Iris-versicolor	5.7cm	2.9cm	4.2cm	1.3cm	Iris-versicolor
Iris-versicolor	6.2cm	2.9cm	4.3cm	1.3cm	Iris-versicolor
Iris-versicolor	5.1cm	2.5cm	3.0cm	1.1cm	Iris-versicolor
Iris-versicolor	5.7cm	2.8cm	4.1cm	1.3cm	Iris-versicolor
Iris-virginica	7.2cm	3.2cm	6.0cm	1.8cm	Iris-virginica
Iris-virginica	6.2cm	2.8cm	4.8cm	1.8cm	Iris-virginica
Iris-virginica	6.1cm	3.0cm	4.9cm	1.8cm	**Iris-versicolor**
Iris-virginica	6.4cm	2.8cm	5.6cm	2.1cm	Iris-virginica
Iris-virginica	7.2cm	3.0cm	5.8cm	1.6cm	Iris-virginica
Iris-virginica	7.4cm	2.8cm	6.1cm	1.9cm	Iris-virginica
Iris-virginica	7.9cm	3.8cm	6.4cm	2.0cm	Iris-virginica
Iris-virginica	6.4cm	2.8cm	5.6cm	2.2cm	Iris-virginica
Iris-virginica	6.3cm	2.8cm	5.1cm	1.5cm	**Iris-versicolor**
Iris-virginica	6.1cm	2.6cm	5.6cm	1.4cm	**Iris-versicolor**
Iris-virginica	7.7cm	3.0cm	6.1cm	2.3cm	Iris-virginica
Iris-virginica	6.3cm	3.4cm	5.6cm	2.4cm	Iris-virginica
Iris-virginica	6.4cm	3.1cm	5.5cm	1.8cm	Iris-virginica
Iris-virginica	6.0cm	3.0cm	4.8cm	1.8cm	**Iris-versicolor**
Iris-virginica	6.9cm	3.1cm	5.4cm	2.1cm	Iris-virginica
Iris-virginica	6.7cm	3.1cm	5.6cm	2.4cm	Iris-virginica
Iris-virginica	6.9cm	3.1cm	5.1cm	2.3cm	Iris-virginica
Iris-virginica	5.8cm	2.7cm	5.1cm	1.9cm	Iris-virginica
Iris-virginica	6.8cm	3.2cm	5.9cm	2.3cm	Iris-virginica
Iris-virginica	6.7cm	3.3cm	5.7cm	2.5cm	Iris-virginica
Iris-virginica	6.7cm	3.0cm	5.2cm	2.3cm	Iris-virginica
Iris-virginica	6.3cm	2.5cm	5.0cm	1.9cm	Iris-virginica
Iris-virginica	6.5cm	3.0cm	5.2cm	2.0cm	Iris-virginica
Iris-virginica	6.2cm	3.4cm	5.4cm	2.3cm	Iris-virginica
Iris-virginica	5.9cm	3.0cm	5.1cm	1.8cm	**Iris-versicolor**