CO3091 Computational Intelligence and Software Engineering

Coursework 2 – Machine Learning

Author: Mohamed Mubaarak
User: mm698 // 149025232

Deadline 23rd November 2017, 23:59

Methods Used

1: findNearestNeighbours(Instance instance)

This method takes a parameter of type instance and outputs an array of instances that are nearest to the provided instance. It does this by using a bubble sort algorithm which compares any given instance at an index in the Training Data to the instance for which we wish to determine the nearest neighbour for. The method for ranking the best/nearest neighbours is determined by the Euclidean distance formula and method (explained further down). If it happens to be that the number of instances in the Training Data is less than K, the method assigns all the values available in the Training Data to the Instance array and returns them.Finally, there is a for loop that iterates to the length of K via a getK method and so using this, returns the number of nearest neighbours up to the index equal to K in the array.

2: normaliseNumericInputAttributesTrainingData()

This method functions to normalise all the available Input attributes in the Training Data(The definition of Normalisation and its function will be discussed in method 3). This is done by traversing the individual instances that are stored in the training data and running the normaliseNumericInputAttributes method with the instance as a parameter. Finally, the normalised values are then set to the current index (i) in the loop and stored in the Training Data.

3: normaliseNumericInputAttributes(Instance Instance)

The main function of this method is to implement the normalisation formula in order to normalise any given instance. It is important to normalise values because diffferent input attributes may use different ranges. These variations in the ranges of input attributes will have an impact on determining the Euclidean Distance (explained further in method 5). Normalisation sets all the values of the input attributes on a scale from 0 to 1. This method uses one for loop to to traverse the input attributes available in a given instance (provided as a parameter to the method). From here we evaluate whether the attribute is either an class index (output attribute) or not. We ignore output attributes as this method is only concerned with the input attributes, and the location of this output attribute is accounted for as the for loop iterates through every possible value available in each index of the instance. In the case that we do meet an input attribute while traversing the instance, the formula for normalisation is applied to the input attribute with requires the calculation of Minimum and Maximum arrays (determined in method 4).

4: determineMinMaxAttributeValues()

The role of this method is to determine and allocate the minimum and maximum possible values for numerical input attributes of the Training Data. These are the values that will be used in the normalisation formula and methods (methods 2 and 3). The method begins by creating two arrays which are each equal in size to the number of attributes available in a given instance (at an index) in the Training Data and fills each of these arrays with 0. Next we check if the iterator (in the for loop) is equal to 0 which will allow you to instantiate the assignment of the min, max arrays by using the iterators I and J by assigning the min array to corresponding instance I at the index J.

Finally the method compares the value of a given instance in the training data at index J, and compares it to the value at the same index position J in the min or max array and assigns the array based on whether it is greater than or smaller than the value of the instance in the training data.

5: euclideanDistance(Instance instance1, Instance Instance2)

This method is used to determine the distance between input attributes of two instances. It does this by firstly iterating through the number of values present in the first instance and checking that it is a numerical input attribute and that it is not the output attribute, **i.e. we are only after numerical input attributes at this specific point**. Next, the method subtracts the value at a given index (i) in the second instance from a value at the same index in the first instance and squares the result using Math.pow. This done for each numerical input attribute and their results are summed together and stored in a variable named “result” which self increments as it iterates through all in the input attributes of instance(s). This in essence, outlines the formula for the determining the Euclidean distance:

$$\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}.$$

However, there arises a problem, what if we come across a non-numerical input attribute such as Language as given in the “Desharnais_edited_nomissing.arff” example Training Data? In this case, the method has an alternative case whereby comparisons can be made between categorical or “Nominal” data which goes as following (as stated in the 16th Lecture slides :

- $X_1 = Y_1$ == We add 0^2 to the total result
- $X_1 \neq Y_1$ == We add 1^2 to the total result

6: determinePredictedOutput(Instance[] nearestNeighbours)

The function of this method is to determine the average of outputs gathered from an array of instances passed as a parameter to the method. This is done by iterating through the nearestNeighbours array and the adding each class value in the instance at a given index to a variable (namely totalOutput in the method). The class value is the numerical value of the output attribute for any given instance. The totalOutput variable self increments with each iteration of the for loop thus, generating a rolling total of the sum of outputs. Next, the sum total is divided by K to produce a value for the average of the nearest neighbour which is the predicted output. The method also accounts for any occassions where the value assigned to K may be greater than the size of the nearestNeighbours array, and in this occasion, is still able to determine the predicted output based on all of instances available in the nearestNeighbours array.

Output Results from WEKA

=== Run information ===

Scheme: weka.classifiers.lazy.MyKnn -K 1
Relation: desharnais.csv-weka.filters.unsupervised.attribute.Remove-R1,5
Instances: 77
Attributes: 10
TeamExp
ManagerExp
YearEnd
Transactions
Entities
PointsNonAdjust
Adjustment
PointsAjust
Language
Effort
Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

MyKnn classifier using k=1.
Trained on 77 examples.

Time taken to build model: 0.86 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient	0.4708
Mean absolute error	2700.3377
Root mean squared error	4077.8434
Relative absolute error	88.7807 %
Root relative squared error	97.0147 %
Total Number of Instances	77