CS440 : Introduction to Artificial Intelligence **Spring 2014**

Problem Set 4 - Problem 1

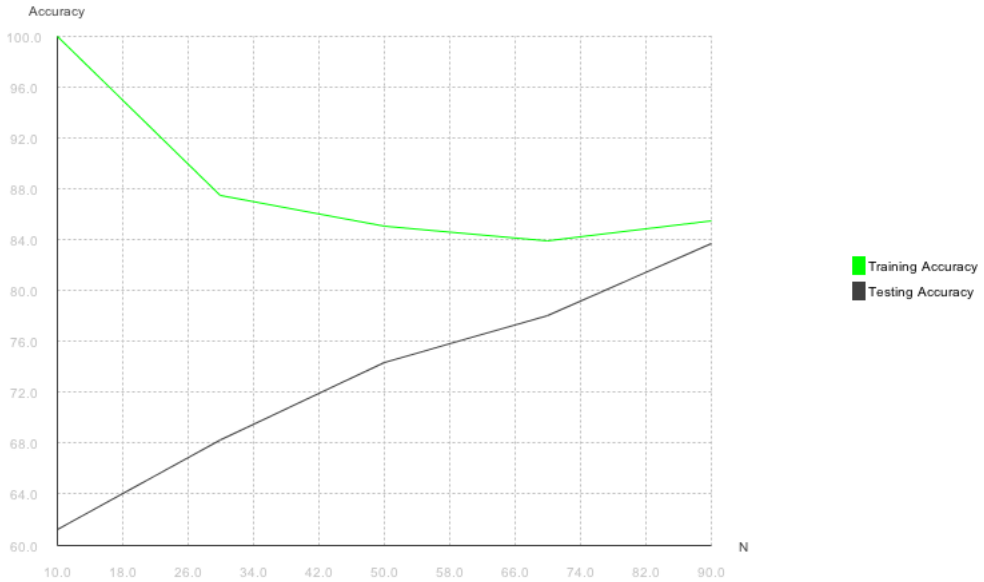
Dan McQuillan *Handed In: April 8, 2014*

1. **Problem 1 - General Classification**

   (a) (i) For each $N = \{10, 30, 50, 70, 90\}$ report the accuracy of the classifier over the training data and the accuracy over the testing data.

N	Training Accuracy	Testing Accuracy
10	100.0	61.21495327102804
30	87.5	68.22429906542057
50	85.04672897196262	74.29906542056075
70	83.89261744966443	78.03738317757009
90	85.41666666666667	83.64485981308411

   (ii) Plot accuracy on training data vs. N and accuracy on testing data vs. N on the same plot



   (iii) Explain the results

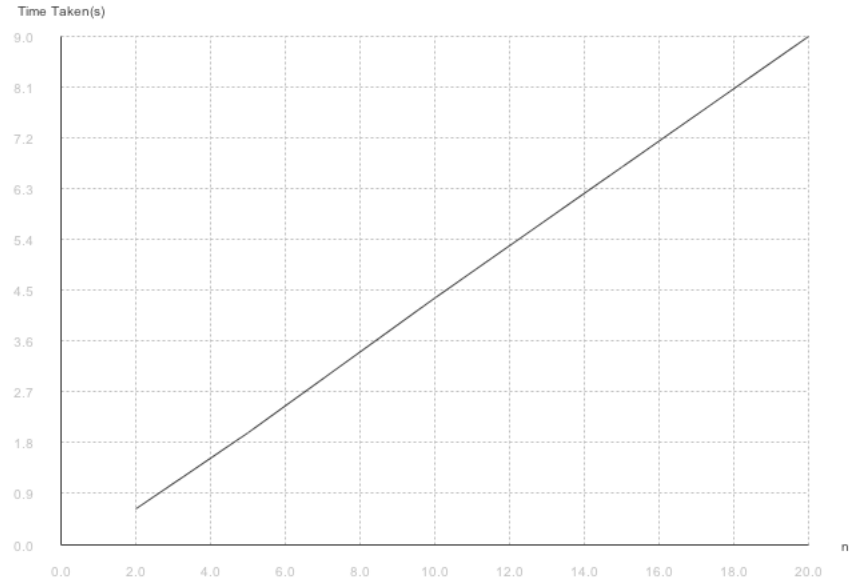
   The result is as expected since with each value of N the amount of training examples is increasing. Therefore, with each next value of N the accuracy will also increase since the bias is increasing. However, since the number of epochs in the weka implementation is constrained to 500 the classifier may not always converge which explains why the accuracy of the classifier drops when more samples are used.

(b)

N	Time Taken	Percent Correct	Percent Incorrect
2	$0.621s$	63.08411214953271	36.91588785046729
5	$1.947s$	60.2803738317757	39.7196261682243
10	$4.276s$	66.35514018691589	33.64485981308411
15	$6.571s$	68.22429906542057	31.77570093457944
20	$9.0s$	66.82242990654206	33.177570093457945

(i) Plot the accuracy vs. n

(ii) Plot the time taken by cross validation vs. n



(iii) Discuss the advantages of using n-fold cross validation as opposed to the testing method we used in part a

The advantage of using n-fold cross validation is simply that the learning of which partitions of data to be used is also learned by k-fold cross validation. Also this not only constrains the data but also randomizes the partitions of the selected data to the optimal partitions for the classifier.

(iv) Discuss the advantages and disadvantages of using a large n

The advantages of using a large n is you will get a set of partitions with smaller size and therefore the resultant classifier will use a larger set of data when it has finished and will eliminate a lot of the invalid priori. However the disadvantage comes with the processing time of the cross validation as shown in the graph from part b.ii.

1 Source Code:

1.1 ClassifierWrapper.java

```
1  package hw4.weka;
2
3  public class ClassifierWrapper {
4
5    private Double trainingAccuracy;
6    private Double testingAccuracy;
7    private final MultilayerPerceptronClassifier classifier;
8
9    public ClassifierWrapper( final Double trainingAccuracy, final Double
       testingAccuracy, final MultilayerPerceptronClassifier classifier ) {
10     this.trainingAccuracy = trainingAccuracy;
11     this.testingAccuracy = testingAccuracy;
12     this.classifier = classifier;
13   }
14
15   public Double getTrainingAccuracy() {
16     return trainingAccuracy;
17   }
18
19   public Double getTestingAccuracy() {
20     return testingAccuracy;
21   }
22
23   public MultilayerPerceptronClassifier getClassifier() {
24     return classifier;
25   }
26
27   public void setTrainingAccuracy(Double trainingAccuracy) {
28     this.trainingAccuracy = trainingAccuracy;
29   }
30
31   public void setTestingAccuracy(Double testingAccuracy) {
32     this.testingAccuracy = testingAccuracy;
33   }
```

```
34
35    @Override
36    public String toString() {
37      StringBuilder stringBuilder = new StringBuilder();
38      stringBuilder.append("Training Accuracy: " + trainingAccuracy );
39      stringBuilder.append("\n");
40      stringBuilder.append("Testing Accuracy: " + testingAccuracy );
41      stringBuilder.append("\n");
42
43      return stringBuilder.toString();
44    }
45
46  }
```

1.2 MultilayerPerceptronClassifier.java

```
1  package hw4.weka;
2
3  import weka.classifiers.functions.MultilayerPerceptron;
4  import weka.core.Instances;
5
6  public class MultilayerPerceptronClassifier {
7
8    private Instances instances = null;
9
10   private Double N = 0.0;
11
12   private MultilayerPerceptron percep;
13
14   public MultilayerPerceptronClassifier( final Instances instances, final
         Double N ) {
15     this.instances = instances;
16     this.N = N;
17   }
18
19   public Double classify() throws Exception {
20     int numInstances = instances.numInstances();
21
22     instances.setClassIndex(instances.numAttributes() - 1);
23
24     percep = new MultilayerPerceptron();
25     Instances trainingData = new Instances(instances, 0, (int) (numInstances *
           (N / 100.0 )) );
26
27     percep.buildClassifier(trainingData);
28
29     return test(instances, N );
30
31   }
32
33   public Double test(Instances testingInstances, Double N ) throws Exception {
34     int numInstances = testingInstances.numInstances();
```

```
35
36      Instances testingData = new Instances(
37          instances,
38          0,
39          (int) (numInstances * (N  / 100.0 ))
40        );
41
42      double correct = 0.0;
43      double incorrect = 0.0;
44
45      for( int i = 0; i < testingData.numInstances(); i++) {
46        double assignedClass = percep.classifyInstance(testingData.instance(i));
47        double originalClass = testingData.instance(i).classValue();
48
49        if( assignedClass == originalClass ) {
50          correct++;
51        } else {
52          incorrect++;
53        }
54      }
55
56      Double accuracy = 100.0 * correct / (correct + incorrect);
57
58      return accuracy;
59    }
60  }
```

1.3 Pair.java

```
1  package hw4.weka;
2
3  public class Pair<T, V> {
4
5    T left;
6    V right;
7
8    public T getLeft() {
9      return left;
10   }
11   public void setLeft(T left) {
12     this.left = left;
13   }
14   public V getRight() {
15     return right;
16   }
17   public void setRight(V right) {
18     this.right = right;
19   }
20
21
22 }
```

1.4 Tuple3.java

```
1  package hw4.weka;
2
3  public class Tuple3<T, V, W> {
4
5    T first;
6    V second;
7    W third;
8
9    public T getFirst() {
10     return first;
11   }
12   public void setFirst(T first) {
13     this.first = first;
14   }
15   public V getSecond() {
16     return second;
17   }
18   public void setSecond(V second) {
19     this.second = second;
20   }
21   public W getThird() {
22     return third;
23   }
24   public void setThird(W third) {
25     this.third = third;
26   }
27 }
```

1.5 Main.java

```
1  package hw4.weka;
2
3  import java.awt.Color;
4  import java.io.BufferedReader;
5  import java.io.BufferedWriter;
6  import java.io.File;
7  import java.io.FileReader;
8  import java.io.FileWriter;
9  import java.io.IOException;
10 import java.util.ArrayList;
11 import java.util.Collections;
12 import java.util.Date;
13 import java.util.HashMap;
14 import java.util.List;
15 import java.util.Map;
16 import java.util.Random;
17
18 import javax.swing.JFrame;
19
20 import org.math.plot.Plot2DPanel;
```

```
21
22 import weka.classifiers.evaluation.Evaluation;
23 import weka.classifiers.evaluation.output.prediction.XML;
24 import weka.classifiers.functions.MultilayerPerceptron;
25 import weka.core.Instances;
26
27 public class Main {
28
29   public static void main(String[] args) {
30
31     try {
32       File file = new File("data/glass.arff");
33       BufferedReader bufferedReader = new BufferedReader(new FileReader(file))
             ;
34
35       Instances instances = new Instances(bufferedReader);
36
37       bufferedReader.close();
38
39       StringBuilder applicationOutput = new StringBuilder();
40       applicationOutput.append("**************************\n").append("
             **************************\n").append("**************************\n"
             )
41             .append("Begin Part A\n")
42             .append("**************************\n").append("
                 **************************\n").append("
                 **************************\n");
43       // Part a
44       double[] nValues = new double[] {
45           10.0,
46           30.0,
47           50.0,
48           70.0,
49           90.0
50       };
51
52       Map<Double, ClassifierWrapper> classificationResults = new HashMap<
             Double, ClassifierWrapper>();
53       double accuracyArray[] = new double[5];
54       double trainingAccuracyArray[] = new double[5];
55       int i = 0;
56       for( Double N : nValues ) {
57
58         ClassifierWrapper wrapper = new ClassifierWrapper(
59             null,
60             null,
61             new MultilayerPerceptronClassifier(instances, N)
62           );
63
64         wrapper.setTrainingAccuracy( wrapper.getClassifier().classify() );
65         wrapper.setTestingAccuracy( wrapper.getClassifier().test( instances,
               100.0 ) );
66         classificationResults.put( N, wrapper );
67
```

```
68          trainingAccuracyArray[i] = wrapper.getTrainingAccuracy();
69          accuracyArray[i++] = wrapper.getTestingAccuracy();
70
71          applicationOutput.append( wrapper.toString() ).append("\n");
72        }
73
74 //        Plot2DPanel plot = new Plot2DPanel();
75 //        plot.addLinePlot("Training Accuracy", Color.GREEN, nValues,
      trainingAccuracyArray);
76 //        plot.addLinePlot("Testing Accuracy", Color.darkGray, nValues,
      accuracyArray);
77 //        plot.setLegendOrientation("NORTH");
78 //        JFrame frame = new JFrame("Problem 1a Results");
79 //        plot.setAxisLabels(new String[] { "N", "Accuracy" });
80 //        frame.setContentPane(plot);
81 //        frame.setVisible(true);
82 //        frame.setSize(800, 600);
83
84        applicationOutput.append("**************************\n").append("
            **************************\n").append("**************************\n"
            )
85                .append("End Part A\n")
86                .append("**************************\n").append("
                    **************************\n").append("
                    **************************\n\n");
87
88        applicationOutput.append("**************************\n").append("
            **************************\n").append("**************************\n"
            )
89                .append("Begin Part B\n")
90                .append("**************************\n").append("
                    **************************\n").append("
                    **************************\n");
91
92        // Part b
93        int[] foldsArray = new int[] {
94            2,
95            5,
96            10,
97            15,
98            20
99        };
100
101       double[] foldsDoubleArray = new double[] {
102           2,
103           5,
104           10,
105           15,
106           20
107       };
108
109       Instances trainInstances = instances;
110       Map<Integer, Tuple3<StringBuffer, Evaluation, Double>> results = new
              HashMap<Integer, Tuple3<StringBuffer, Evaluation, Double>>();
```

```
111         for( final Integer folds : foldsArray ) {
112           Tuple3<StringBuffer, Evaluation, Double> tuple3 = new Tuple3<
                StringBuffer, Evaluation, Double>();
113
114           StringBuffer internalStringBuffer = new StringBuffer();
115           XML internalOutput = new XML();
116           internalOutput.setBuffer(internalStringBuffer);
117           internalOutput.setHeader(trainInstances);
118           internalOutput.setOutputDistribution(true);
119
120           Evaluation evaluation = new Evaluation(trainInstances);
121
122           MultilayerPerceptron percep = new MultilayerPerceptron();
123           Date beforeCrossValidate = new Date();
124           evaluation.crossValidateModel(percep, trainInstances, folds, new
                Random( new Date().getTime() ), internalOutput );
125           Date afterCrossValidate = new Date();
126
127           Double timeTakenCrossValidation =  ((double)( afterCrossValidate.
                getTime() - beforeCrossValidate.getTime())) / 1000.0;
128
129           tuple3.setFirst( internalStringBuffer );
130           tuple3.setSecond( evaluation );
131           tuple3.setThird( timeTakenCrossValidation );
132
133           results.put(folds, tuple3);
134         }
135         List<Integer> keySet = new ArrayList<Integer>();
136         for( Integer key : results.keySet() ) {
137           keySet.add(key);
138         }
139         Collections.sort(keySet);
140         accuracyArray = new double[5];
141         double timeTakenArray[] = new double[5];
142         i = 0;
143         for( final Integer key : keySet ) {
144           Tuple3<StringBuffer, Evaluation, Double> result = results.get(key);
145           File outputFile = new File("result/cross-validation-output-" + key + "
                .xml");
146           if( outputFile.exists() ) {
147             outputFile.delete();
148           }
149
150           BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(
                outputFile));
151           bufferedWriter.write(result.getFirst().toString());
152           bufferedWriter.close();
153
154           timeTakenArray[i] = result.getThird();
155           accuracyArray[i++] = result.getSecond().pctCorrect();
156
157           applicationOutput.append("*********************************\nCross
                Validation Result\n*********************************\n");
158           applicationOutput.append("Results for N = ").append(key).append("\n");
```

```
159           applicationOutput.append("Time taken: " + result.getThird() + "s\n");
160           applicationOutput.append("Percent Correct: " + result.getSecond().
                  pctCorrect() + "\n");
161           applicationOutput.append("Percent Incorrect: " + result.getSecond().
                  pctIncorrect() + "\n");
162           applicationOutput.append("Confusion Matrix: \n" + result.getSecond().
                  confusionMatrix() + "\n\n");
163 //          applicationOutput.append("-------------\nXML Data\n-------------\n\n")
        .append( result.getFirst().toString() ).append("\n\n");
164         }
165
166         Plot2DPanel plot2 = new Plot2DPanel();
167         plot2.addLinePlot("n vs. Accuracy", Color.DARK_GRAY, foldsDoubleArray,
                accuracyArray);
168         JFrame frame2 = new JFrame("Problem 1b Results");
169         plot2.setAxisLabels(new String[] { "N", "Accuracy" });
170         frame2.setContentPane(plot2);
171         frame2.setVisible(true);
172         frame2.setSize(800, 600);
173
174
175         Plot2DPanel plot3 = new Plot2DPanel();
176         plot3.addLinePlot("n vs. Time Taken by Cross Validation", Color.
                DARK_GRAY, foldsDoubleArray, timeTakenArray);
177         JFrame frame3 = new JFrame("Problem 1b Results");
178         plot3.setAxisLabels(new String[] { "n", "Time Taken(s)" });
179         frame3.setContentPane(plot3);
180         frame3.setVisible(true);
181         frame3.setSize(800, 600);
182
183         applicationOutput.append("**************************\n").append("
                **************************\n").append("**************************\n"
                )
184                 .append("End Part B\n")
185                 .append("**************************\n").append("
                        **************************\n").append("
                        **************************\n");
186
187         System.out.println(applicationOutput.toString());
188       } catch (IOException e) {
189         System.err.println( e );
190       } catch (Exception e) {
191         e.printStackTrace();
192       }
193
194     }
195
196 }
```