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

Problem Set 4 - Problem 4

Dan McQuillan *Handed In: April 8, 2014*

1. **Solution to problem 4**

   (a) I again used the following four topologies when training the neural nets within the classifier with varying results:

   (1) A neural network with 1 hidden layer consisting of 10 nodes.

   (2) A neural network with 2 hidden layers with 10 nodes in the first layer and 5 nodes in the second layer.

   (3) A neural network with 4 hidden layers with 5 nodes in the first layer, 10 nodes in the second layer, 15 nodes in the third layer and 20 nodes in the fourth layer

   (4) A neural network with 8 hidden layers with 5 nodes in the first layer, 10 nodes in the second layer, 15 nodes in the third layer, 20 nodes in the fourth layer, 15 nodes in the fifth layer, 10 nodes in the sixth layer, 5 nodes in the seventh layer and 2 nodes in the eighth layer.

   (b) Determine experimentally whether the bagging technique helps avoid over fitting in this case

   My expectation for this experiment was that I would obtain a higher accuracy with the bagging classifier than the neural network classifier. Therefore, to prove this expectation I used the same network topology for bagging as I did for problem 3a (as described above). The results of the training and testing compared to the results of the neural network are below:

Neural net #	Bagging Accuracy	Neural Network Accuracy
1	14.018691588785046%	67.28971962616822%
2	14.018691588785046%	67.99065420560747%
3	14.330218068535826%	57.00934579439252%
4	14.25233644859813%	51.518691588785046%

   Since I did not see a higher accuracy after training, it leads me to believe that I chose bad parameters for the neural network which could be the cause of the drop in accuracy. I do however think that the bagging technique helped to reduce over fitting. I believe this because in the results of the training it showed that the bagging technique consistently produced an accuracy of $\sim 14\%$ whereas the variance of the neural network accuracy was much larger. Therefore the bagging technique did help alleviate over fitting.

1 Source code:

1.1 MainP4.java

```
1  package hw4.weka;
2
3  import java.io.BufferedReader;
4  import java.io.File;
5  import java.io.FileReader;
6  import java.io.IOException;
7  import java.util.Date;
8  import java.util.Random;
9
10 import weka.classifiers.evaluation.Evaluation;
11 import weka.classifiers.evaluation.output.prediction.XML;
12 import weka.classifiers.functions.MultilayerPerceptron;
13 import weka.classifiers.meta.Bagging;
14 import weka.core.Instances;
15
16 public class MainP4 {
17
18   public static void main(String[] args) {
19     try {
20
21       File file = new File("glass.arff");
22       BufferedReader bufferedReader = new BufferedReader(new FileReader(file))
             ;
23
24
25       Instances instances = new Instances(bufferedReader);
26
27       bufferedReader.close();
28
29       final int numInstances = instances.numAttributes();
30       instances.setClassIndex(numInstances - 1);
31
32       Evaluation evaluation = new Evaluation(instances);
33       final Integer folds = 10;
34
35       StringBuilder applicationOutput = new StringBuilder();
36
37       applicationOutput.append("Begin problem 3 part a\n");
38       applicationOutput.append("folds: " + folds + "\n\n");
39
40
41       StringBuffer internalStringBuffer = new StringBuffer();
42       XML internalOutput = new XML();
43       internalOutput.setBuffer(internalStringBuffer);
44       internalOutput.setHeader(instances);
45       internalOutput.setOutputDistribution(true);
46
47 //       percep.setGUI(true);
48 //       percep.buildClassifier(instances);
```

```
49
50          MultilayerPerceptron percep = new MultilayerPerceptron();
51          percep.setHiddenLayers("10");
52          Bagging bagging = new Bagging();
53          bagging.setClassifier(percep);
54          evaluation.crossValidateModel(bagging, instances, folds, new Random( new
                Date().getTime() ), internalOutput );
55          applicationOutput.append("Accuracy: " + evaluation.pctCorrect() + "%\n")
                ;
56
57          MultilayerPerceptron percep2 = new MultilayerPerceptron();
58          percep2.setHiddenLayers("10,5");
59          Bagging bagging2 = new Bagging();
60          bagging2.setClassifier(percep);
61          evaluation.crossValidateModel(bagging2, instances, folds, new Random(
                new Date().getTime() ), internalOutput );
62          applicationOutput.append("Accuracy: " + evaluation.pctCorrect() + "%\n")
                ;
63
64          MultilayerPerceptron percep3 = new MultilayerPerceptron();
65          percep3.setHiddenLayers("5,10,15,20");
66          Bagging bagging3 = new Bagging();
67          bagging3.setClassifier(percep);
68          evaluation.crossValidateModel(bagging3, instances, folds, new Random(
                new Date().getTime() ), internalOutput );
69          applicationOutput.append("Accuracy: " + evaluation.pctCorrect() + "%\n")
                ;
70
71          MultilayerPerceptron percep4 = new MultilayerPerceptron();
72          percep4.setHiddenLayers("5,10,15,20,15,10,5,2");
73          Bagging bagging4 = new Bagging();
74          bagging4.setClassifier(percep);
75          evaluation.crossValidateModel(bagging4, instances, folds, new Random(
                new Date().getTime() ), internalOutput );
76          applicationOutput.append("Accuracy: " + evaluation.pctCorrect() + "%\n")
                ;
77
78          System.out.println("complete.");
79
80          applicationOutput.append("Begin problem 4 part a\n");
81          applicationOutput.append("folds: " + folds + "\n\n");
82
83
84          Bagging percepTest = new Bagging();
85  //          percep.setHiddenLayers("10");
86  //          percep.setGUI(true);
87          percepTest.buildClassifier(instances);
88
89          evaluation.crossValidateModel(percepTest, instances, folds, new Random(
                new Date().getTime() ), internalOutput );
90          applicationOutput.append("Accuracy: " + evaluation.pctCorrect() + "%\n")
                ;
91
92          System.out.println(applicationOutput.toString());
```

```
93
94          System.out.println("complete.");
95
96        } catch (IOException e) {
97          System.err.println( e );
98        } catch (Exception e) {
99          e.printStackTrace();
100       }
101     }
102
103 }
```