```java
package edu.gmu.classifier.naivebayes;

import java.io.File;
import java.io.FilenameFilter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import edu.gmu.classifier.io.DataLoader;
import edu.gmu.classifier.io.TrainingExample;
import edu.gmu.classifier.neuralnet.util.NeuralNetUtils;

public class Homework7
{
    public static void main( String[] args ) throws IOException
    {
        // list the test data files
        File dataDirectory = new File( "/home/ulman/CSI873/midterm/data" );
        String[] testDataFiles = dataDirectory.list( new FilenameFilter( )
        {
            @Override
            public boolean accept( File dir, String name )
            {
                return name.startsWith( "test" );
            }
```

```
29          } );
30
31          // sort the testDataFiles
32          Arrays.sort( testDataFiles );
33
34          // list the training data files
35          String[] trainingDataFiles = dataDirectory.list
  ( new FilenameFilter( )
36          {
37              @Override
38              public boolean accept( File dir, String
  name )
39              {
40                  return name.startsWith( "train" );
41              }
42          } );
43
44          // sort the trainingDataFiles
45          Arrays.sort( trainingDataFiles );
46
47          // load all test data examples
48          List<TrainingExample> testDataList = new
  ArrayList<TrainingExample>( );
49          for ( String fileName : testDataFiles )
50          {
51              testDataList.addAll( DataLoader.loadFile( new
  File( dataDirectory, fileName ) ) );
52          }
53
54          // load all training data examples
55          List<TrainingExample> trainingDataListAll = new
  ArrayList<TrainingExample>( );
```

```
56          final Map<Integer, List<TrainingExample>>
  trainingDataMap = new HashMap<Integer,
  List<TrainingExample>>( );
57          for ( String fileName : trainingDataFiles )
58          {
59              List<TrainingExample> list =
  DataLoader.loadFile( new File( dataDirectory,
  fileName ) );
60              int digit = list.get( 0 ).getDigit( );
61              trainingDataMap.put( digit, list );
62              trainingDataListAll.addAll( list );
63          }
64
65          // create a data structure to store the training
  frequencies
66          Map<P, Double> p0map = new HashMap<P, Double>( );
67
68          // loop over digits and input indices,
  calculating the conditional probabilities:
69          // p( index i = 0 | digit = d )
70          for ( int d = 0; d < 10; d++ )
71          {
72              List<TrainingExample> examplesForDigit =
  trainingDataMap.get( d );
73
74              for ( int i = 0; i < DataLoader.INPUT_SIZE; i
  ++ )
75              {
76                  P key = new P( i, d );
77                  Double p = calculateTrainingProbability
  ( examplesForDigit, d, i );
78                  p0map.put( key, p );
```

```
 79              }
 80          }
 81
 82          double trainErrorRate = calculateErrorRate
   ( p0map, trainingDataListAll );
 83          double testErrorRate = calculateErrorRate( p0map,
   testDataList );
 84
 85          System.out.printf( "Training Error Rate: %.3f
   Testing ErrorRate: %.3f%n", trainErrorRate,
   testErrorRate );
 86      }
 87
 88      /**
 89       * Calculate the conditional probability p( index i =
   0 | digit = d )
 90       *
 91       * @param examplesForDigit the set of training digits
   for digit d
 92       * @param digit
 93       * @param inputIndex
 94       * @return the probability p( index i = 0 | digit =
   d )
 95       */
 96      public static double calculateTrainingProbability
   ( List<TrainingExample> examplesForDigit, int digit, int
   inputIndex )
 97      {
 98          int count = 0;
 99
100          for ( TrainingExample example :
   examplesForDigit )
```

```
101         {
102             if ( example.getInputs( )[inputIndex] ==
    0.0 ) count++;
103         }
104
105         if ( count == 0 )
106         {
107             System.out.printf( "Zero probability warning:
    index: %d digit: %d%n", inputIndex, digit );
108         }
109
110         return (double) count / ( double )
    examplesForDigit.size( );
111     }
112
113     public static double calculateErrorRate( Map<P,
    Double> p0map, List<TrainingExample> dataList )
114     {
115         int correctCount = 0;
116
117         for ( TrainingExample data : dataList )
118         {
119             double[] likelihoods =
    calculateOutputLikelihoods( p0map, data );
120             int predictedDigit =
    NeuralNetUtils.getLargestIndex( likelihoods );
121
122             if ( predictedDigit == data.getDigit( ) )
123             {
124                 correctCount++;
125             }
126         }
```

```
127
128         return 1.0 - ( double ) correctCount / (double)
    dataList.size( );
129     }
130
131     /**
132      * Use the pre-calculated conditional probabilities
    stored in p0map to calculate the probability
133      * that the TrainingExample represents each of the
    ten possible digits.
134      *
135      * @param p0map a map of pre-computed conditional
    probabilities p( index i = 0 | digit = d )
136      * @param data the training data element
137      * @return the probability that data represents each
    of the possible ten digits
138      */
139     public static double[] calculateOutputLikelihoods
    ( Map<P, Double> p0map, TrainingExample data )
140     {
141         // allocate an array to return the 10 calculated
    probabilities for each digit
142         double[] likelihoods = new double[10];
143         // get the input data for the training example
144         double[] input = data.getInputs( );
145
146         // apply the naive bayes classifier using the
    pre-calculated conditional probabilities
147         // the result is one likelihood value for each
    digit
148         for ( int d = 0; d < 10; d++ )
149         {
```

```
150             // in our case, all the digits have the same
    number of values, so p(v) is always 0.1
151             double likelihood = 0.1;
152 
153             for ( int i = 0 ; i < DataLoader.INPUT_SIZE;
    i++ )
154             {
155                 // get the pre-computed conditional
    probability for the current digit and input index
156                 double p = p0map.get( new P( i, d ) );
157                 // the pre-computed probabilities are for
    input = 0, subtract one minus the probability
158                 // if the input value at the index is a 1
159                 likelihood *= input[i] == 0 ? p : 1-p;
160             }
161 
162             likelihoods[d] = likelihood;
163         }
164 
165         return likelihoods;
166     }
167}
168 
```