Zequn Yu
CSE 151

PA – 1

1. Table:

	Training Error	Test Error	Validation Error
K = 1	0.0	0.094	0.082
K = 3	0.0435	0.092	0.098
K = 5	0.0565	0.098	0.095
K =9	0.0685	0.101	0.104
K = 15	0.0925	0.114	0.108

Based on the table, when K = 1, validation error is lowest. So 1NN classifier performs the best on validation data. The test error is 0.094.

2.

	Training Error	Test Error	Validation Error
K = 1	0.0	0.335	0.335
K = 3	0.1695	0.322	0.315
K = 5	0.195	0.31	0.312
K = 9	0.2285	0.309	0.309
K = 15	0.257	0.317	0.306

Based on the table, when k = 15, validation error is lowest. So 15NN classifier performs the best on validation data. The test error is 0.257. My program will run faster on projected data with less accuracy.

---------- Code ----------

```
/*
 * Author: Zequn Yu
 * ID: A14712777
 *
 * KNN, K = 1, 5, 9, 15
 */

import java.io.*;
import java.util.*;
import java.lang.Math;

public class KNN {

	// list of list to store each number in one line
	static LinkedList<double[]> trainList;
	static LinkedList<double[]> testList1;
	static LinkedList<double[]> testList2;
	static LinkedList<double[]> testList3;
```

```
    public static void main(String args[]) throws
IOException{

          // get training error

          // read data from pa1train.txt as train points
          //File trainFile = new
File("/Users/Jason/Desktop/151pa1/151pa1/src/pa1train.txt");
          File trainFile = new
File("/Users/Jason/Desktop/151pa1/151pa1/src/projection.txt");
          BufferedReader rtrain = new BufferedReader(new
FileReader(trainFile));

          // read data from pa1train.txt as test points
          File testTrainFile1 = new
File("/Users/Jason/Desktop/151pa1/151pa1/src/pa1train.txt");
          File testTrainFile2 = new
File("/Users/Jason/Desktop/151pa1/151pa1/src/pa1test.txt");
          File testTrainFile3 = new
File("/Users/Jason/Desktop/151pa1/151pa1/src/pa1validate.txt");
          BufferedReader rtest1 = new BufferedReader(new
FileReader(testTrainFile1));
          BufferedReader rtest2 = new BufferedReader(new
FileReader(testTrainFile2));
          BufferedReader rtest3 = new BufferedReader(new
FileReader(testTrainFile3));

          // collect each number in one line
          // for train data
          //eg. tl[0] = {...}, tl[1] = {...}, tl[2] = {...}
          trainList = collectData(rtrain);
          // for test data
          testList1 = collectData(rtest1);
          testList2 = collectData(rtest2);
          testList3 = collectData(rtest3);

          // for part 2
          LinkedList<double[]> pTr;
          LinkedList<double[]> pVa;
          LinkedList<double[]> pTe;
       double [][] Proj = new double[trainList.size()][];
       for (int i = 0; i < trainList.size(); i++) {
          Proj[i] = trainList.get(i);
       }
          pTr = project(testList1, Proj);
```

```
pVa = project(testList2, Proj);
pTe = project(testList3, Proj);

// set the index of case
// set pq for KNN
//PriorityQueue<Data> KNN;
int err = 0;
int setSize = testList3.size();
int K;
double result;

//---- part 2 -------//
K = 1;
System.out.println("When K = : " + K);
err = getKNN(K, pTr, pVa);
result = (double)err / (double)setSize;
System.out.println("Training errors: " + result);

K = 3;
System.out.println("When K = : " + K);
err = getKNN(K, pTr, pVa);;
result = (double)err / (double)setSize;
System.out.println("Training errors: " + result);

K = 5;
System.out.println("When K = : " + K);
err = getKNN(K, pTr, pVa);
result = (double)err / (double)setSize;
System.out.println("Training errors: " + result);

K = 9;
System.out.println("When K = : " + K);
err = getKNN(K, pTr, pVa);
result = (double)err / (double)setSize;
System.out.println("Training errors: " + result);

K = 15;
System.out.println("When K = : " + K);
err = getKNN(K, pTr, pVa);
result = (double)err / (double)setSize;
System.out.println("Training errors: " + result);

//------ part 1 --------//


/*
```

```
// for case 1
K = 1;
err = getKNN(K, trainList, testList);
System.out.println("When K = : " + K);
//System.out.println("Error: " + err);
//setSize = trainList.size();
//System.out.println("Size: " + setSize);
result = (double)err / (double)setSize;
System.out.println("Training errors: " + result);


// test case: for case 3
K = 3;
// get the error number
err = getKNN(K, trainList, testList);
System.out.println("When K = : " + K);
//System.out.println("Error: " + err);
//setSize = trainList.size();
//System.out.println("Size: " + setSize);
result = (double)err / (double)setSize;
System.out.println("Training errors: " + result);


// for case 5
K = 5;
err = getKNN(K, trainList, testList);
System.out.println("When K = : " + K);
//System.out.println("Error: " + err);
//setSize = trainList.size();
//System.out.println("Size: " + setSize);
result = (double)err / (double)setSize;
System.out.println("Training errors: " + result);


// for case 9
K = 9;
err = getKNN(K, trainList, testList);
System.out.println("When K = : " + K);
//System.out.println("Error: " + err);
//setSize = trainList.size();
//System.out.println("Size: " + setSize);
result = (double)err / (double)setSize;
System.out.println("Training errors: " + result);


// for case 15
K = 15;
```

```
            err = getKNN(K, trainList, testList);
            System.out.println("When K = : " + K);
            //System.out.println("Error: " + err);
            //setSize = trainList.size();
            //System.out.println("Size: " + setSize);
            result = (double)err / (double)setSize;
            System.out.println("Training errors: " + result);

            */
      }

    private static LinkedList<double[]>
project(LinkedList<double[]> testList2, double[][] p2) {
      LinkedList<double[]> p = new LinkedList<double[]>();
      // for each row
         for (double[] row : testList2) {
             // set label
            double label = row[784];
             // save data
            double[] data = new double[row.length-1];
             for( int i = 0; i < row.length - 1; i++){
                 data[i] = row[i];
             }
             double[] tmpv = new double[21];
             tmpv[20] = label;
             // project element in two matrix
             for (int i = 0; i < tmpv.length - 1; i++) {
                 for (int j = 0; j < data.length - 1; j++) {
                 tmpv[i] += data[j] * p2[j][i];
                 }
             }
             p.add(tmpv);
         }

         return p;
    }


      // function for collecting actual data from .txt file
      public static LinkedList<double[]>
collectData(BufferedReader br) throws IOException{
            // set a return value
            LinkedList<double[]> dataList = new
LinkedList<double[]>();

            // read one line each time
            String st;
```

```
            while( (st = br.readLine()) != null){
                  // set a array to store each number in line
splited by whitespace
                  String snum[] = st.split("\\s");
                  // get length of snum
                  int numOfNum = snum.length;
                  // set a array to store each number in integer
                  double[] inum = new double[numOfNum];
                  // transfer String to integer
                  for( int i = 0; i < numOfNum; i++){
                        // transfer
                        inum[i] = Double.parseDouble(snum[i]);
                  }
                  // this line is done, collect all integer data
                  dataList.add(inum);
            }
            // return dataList
            return dataList;
      }// collectData done

      // class for pair(distance, label)
      public static class Data implements Comparable<Data>{
            public double getLabel() {
                  return label;
            }

            public void setLabel(double label) {
                  this.label = label;
            }

            public void setDistance(double distance) {
                  this.distance = distance;
            }

            public double getDistance() {
                  return distance;
            }

            private double distance;
            private double label;

            public Data( double distance, double label){
                  this.distance = distance;
                  this.label = label;
            }

            @Override
```

```
            public int compareTo(Data o) {
                  // TODO Auto-generated method stub
                  return 0;
            }

      }

      static class DataComparator implements
Comparator<Data>{
         public int compare(Data d1, Data d2) {
             if (d1.distance < d2.distance)
                 return 1;
             else if (d1.distance > d2.distance)
                 return -1;
             return 0;
             }
     }


      // function to get KNN
      public static int getKNN( int K, LinkedList<double[]>
p, LinkedList<double[]> p2){
            // set return value
           PriorityQueue<Data> KNN = new PriorityQueue<Data>(K,
new DataComparator());
            //PriorityQueue<Integer> dis = new
PriorityQueue<Integer>();
            int errNum = 0;

            // calculate the distance
            // for each array element in bList
            for( double btemp[] : p2){
                  // set b label
                  double blabel = btemp[btemp.length - 1];
                  // for each array element in aList
                  for( double atemp[] : p){
                        // calculate distance and add into KNN
                        //Integer distance = calDistance(atemp,
btemp);
                        //dis.add(distance);
                        //Data tmp = calDistance(atemp, btemp);
                        double distance = calDistance(atemp,
btemp);
                        double label = atemp[atemp.length - 1];
                        // if pq < K, add them
                        Data data = new Data(distance, label);
                        KNN.add(data);
```

```
                    // else throw biggest distance
                    if ( KNN.size() > K){
                        KNN.poll();
                    }
                }
                // all lines of btemp done.
                // get the target label
                // because the number of label is fixed
                Integer labelTable[] = { 0, 0, 0, 0, 0, 0, 0,
0, 0, 0 };
                // loop for KNN
                while(KNN.isEmpty() != true){
                    int tmpLabel = (int)KNN.poll().label;
                    labelTable[tmpLabel]++;
                }
                // check which label is greatest
                int retLabel = 0;
                for(int i = 0; i < labelTable.length; i++ ){
                    if(labelTable[i] >
labelTable[retLabel] ){
                        retLabel = i;
                    }
                }
                // check error
                if( retLabel != blabel){
                    errNum++;
                }
            }// for loop ends
            // return
            return errNum;
        }

        public static double calDistance(double[] atemp,
double[] btemp) {
            // set return value
            double distance = 0.0;
            //int label = 0;

            for( int i = 0; i < atemp.length - 2; i++ ){
                //distance += Math.pow((atemp[i] - btemp[i]),
2);
                distance += (atemp[i] - btemp[i]) * (atemp[i]
- btemp[i]);
            }
            // return var
            //label = atemp[atemp.length - 1];
            return distance;
```

```
            //Data ret = new Data(distance, label);
            //return ret;
            //PriorityQueue<Data> ret = new PriorityQueue<>();
            //Data var = null;
            //var.setDistance(distance);
            //var.setLabel(label);
            //ret.add(var);
            //return ret;
        }

    }
```