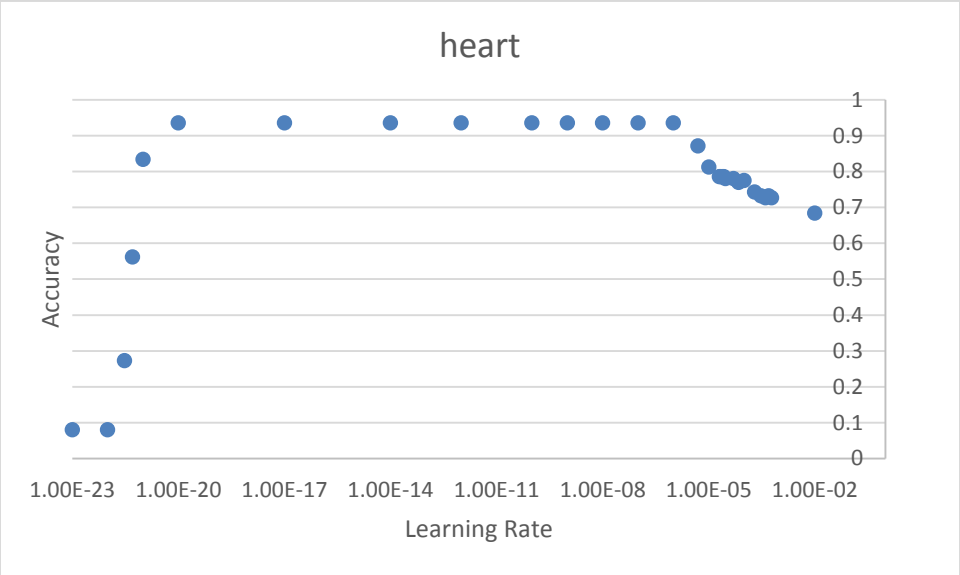
1. simple
   Class 0: tested 2, correctly classified 2
   Class 1: tested 2, correctly classified 2
   Overall: tested 4, correctly classified 4
   Accuracy: 1.000000

2. vote
   Class 0: tested 52, correctly classified 51
   Class 1: tested 83, correctly classified 83
   Overall: tested 135, correctly classified 134
   Accuracy: 0.992593

3. heart
   learning rate: .00000001
   Class 0: tested 15, correctly classified 7
   Class 1: tested 172, correctly classified 168
   Overall: tested 187, correctly classified 175
   Accuracy: 0.935829



4. Learning rates that are too small do not reach logistic functions that are representative models of the data, because each step size is not large enough to improve the beta values by sufficient amounts. Learning rates that are too large result in large step sizes that are not contoured to the function in gradient descent and may overshoot the optimal beta values.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

/* CS109 HW5.2: Logistic Regression
 * --------------------------------
 * This file contains starter code for your logistic regression
classifier.
 * Your job is to read in the data file and implement the logistic
regression
 * algorithm.
 *
 * Note: This starter code is written without the Stanford libraries.
If you
 * took CS106A and want that style of starter code, download the other
Java
 * starter code.
 */

public class LogisticRegression {
      String fileName = "simple";
      Scanner read;
      int vectorDim;
      int[][] vect; //[numVectors][vectorDimensions+1]
                        //X0 defined as 1, use one-indexing
      int[] y;
      double[] beta;
      int[][] ycount = new int[2][2];//y=0, tested, correct
                                                    //y=1, tested,
correct


      public void run()
      {
            read = new Scanner(System.in);
            System.out.print("Select a file (simple, vote, heart): ");
            fileName = read.next();

            readTrainingInput();
```

```
            learnBeta();

            //sanity check
//          for(int i = 0; i < 48; i+=2)
//               System.out.println(i + "   " + beta[i]);


            calcResults();
            printResults();
      }

      public void calcResults()
      {
            try {
                  read = new Scanner(new File("src/PC-datasets/" +
fileName + "-test-PC.txt"));
            } catch (FileNotFoundException e) {
                  e.printStackTrace();
            }
            vectorDim = read.nextInt();
            int numVectors = read.nextInt();

            for(int c = 0; c < numVectors; c++)
            {
                  int[] x = new int[vectorDim+1];
                  x[0] = 1;
                  for(int j = 1; j < vectorDim; j++)
                  {
                        x[j] = read.nextInt();
                  }
                  x[vectorDim] =
Integer.parseInt(read.next().substring(0, 1));
                  int y = read.nextInt();


                  double z = 0.0;
                  for(int j = 0; j <= vectorDim; j++)
                        z += beta[j] * x[j];
                  double p = 1/(1+Math.exp(-z));
```

```
                int yhat = 0;
                if(p > 0.5)
                      yhat = 1;
//              System.out.println(y + "     " + yhat);
                ycount[y][0]++;
                if(y == yhat)
                      ycount[y][1]++;

          }
     }

     public void printResults()
     {
          System.out.printf("Class 0: tested %d, correctly classified
%d\n", ycount[0][0], ycount[0][1]);
          System.out.printf("Class 1: tested %d, correctly classified
%d\n", ycount[1][0], ycount[1][1]);
          System.out.printf("Overall: tested %d, correctly classified
%d\n", ycount[0][0]+ycount[1][0], ycount[0][1]+ycount[1][1]);
          System.out.printf("Accuracy: %f",
(double)(ycount[0][1]+ycount[1][1])/(ycount[0][0]+ycount[1][0]));
     }

     public void learnBeta()
     {
          beta = new double[vectorDim+1];
          int epoch = 10000;
          double learnRate = 0.0001;
          if(fileName.equals("heart"))
          {
                read = new Scanner(System.in);
                System.out.print("Enter a learning rate: ");
                learnRate = read.nextDouble();
          }
          for(int e = 0; e < epoch; e++)
          {
                double[] gradient = new double[vectorDim+1];
                double[] z = new double[y.length];
                for(int i = 0; i < z.length; i++)
```

```
                {
                      z[i] = 0.0;
                      for(int j = 0; j <= vectorDim; j++)
                            z[i] += beta[j] * vect[i][j];
                }
                for(int k = 0; k <= vectorDim; k++)
                {
                      for(int i = 0; i < y.length; i++)
                      {
                            gradient[k] += vect[i][k] * (y[i] -
1/(1+Math.exp(-z[i])));
                      }
                }
                for(int k = 0; k <= vectorDim; k++)
                {
                      beta[k] += learnRate * gradient[k];
                }
          }
     }

//    public double z(int i)
//    {
//          double res = 0.0;
//          for(int j = 0; j <= vectorDim; j++)
//          {
////              System.out.println(i+ "   " + j);
//                res += beta[j] * vect[i][j];
//          }
//          return res;
//    }

     public void readTrainingInput()
     {
          try {
                read = new Scanner(new File("src/PC-datasets/" +
fileName + "-train-PC.txt"));
          } catch (FileNotFoundException e) {
                e.printStackTrace();
          }
```

```
            vectorDim = read.nextInt();
            int numVectors = read.nextInt();
            vect = new int[numVectors][vectorDim+1];
            y = new int[numVectors];
            for(int i = 0; i < numVectors; i++)
            {
                  vect[i][0] = 1;
                  for(int j = 1; j < vectorDim; j++)
                  {
                        vect[i][j] = read.nextInt();
                  }
                  //last x-value has colon attached to it
                  vect[i][vectorDim] =
Integer.parseInt(read.next().substring(0, 1));
                  y[i] = read.nextInt();
            }
      }

//    public double z(double beta)
//    {
//          for(int )
//    }

      public static void main(String[] args) {
            //TODO: Fill this out!
            LogisticRegression l = new LogisticRegression();
            l.run();
      }

}
```