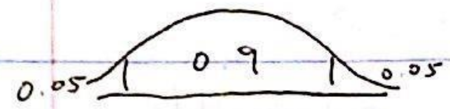
Homework 5.1

1) a) $\bar{x} = 447$ $\quad s^2 = 376.36$ $\quad n = 300$

$\alpha = 10\%$

$z = 1.645$



$ME = 1.645 \sqrt{\frac{376.36}{300}} = 1.8425$

$\boxed{(445.158 \,,\ 448.842) \text{ seconds}}$

b) $z = 2.58$

$ME = 2.58 \sqrt{\frac{376.36}{300}} = 2.885$

$\boxed{(444.1,\ 449.9) \text{ seconds}}$

2) a) $L(\theta) = \prod_{i=1}^{n} f(x_i | \theta) = \prod_{i=1}^{n} \lambda e^{-\lambda x_i}$

$LL(\theta) = \sum_{i=1}^{n} \log(\cancel{\#} \lambda e^{-\lambda x_i})$

$= \sum_{i=1}^{n} \log \lambda - \lambda X_i$

$= n \log \lambda - \lambda \sum_{i=1}^{n} X_i$ $\qquad Y = \sum_{i=1}^{n} X_i$

$= n \log \lambda - \lambda Y$

$$\frac{\partial LL(\lambda)}{\partial \lambda} = \frac{n}{\lambda} - Y = 0$$

$$\boxed{\lambda_{MLE} = \frac{n}{Y}} = \frac{n}{\sum_{i=1}^{n} X_i} = \frac{1}{\bar{X}}$$

b) Yes it is biased because the PDF for the exponential distribution is convex, so $E[f(x)] \geq f(E[x])$ and the bias is not 0.

c) $\lim_{n\to\infty} P(|\lambda_{MLE} - \lambda| < \varepsilon) = 1$ for $\varepsilon > 0$

$= \lim_{n\to\infty} P(|\frac{1}{\bar{x}} - \lambda| < \varepsilon) = 1$

Yes it is consistent because as n increases $\bar{X}$ approaches $\frac{1}{\lambda}$.

Jensen's

$f(x) = \frac{1}{x}$ $\quad E[\frac{1}{x}] \geq \frac{1}{E[x]}$

$\frac{1}{x}$ is not a line, so

$$E[\frac{1}{x}] > \frac{1}{E[x]} = \frac{1}{\frac{1}{\lambda}} = \lambda$$

$$E[\frac{1}{x}] = \frac{1}{E[x]} = \lambda_{MLE} > \lambda$$

3) If the value of X_i is known, the values for all other $X_{i \leq K}$ are known as well. The X ~~values~~ random variables are not independent, so it is problematic for the model learned by the Naive Bayesian Classifier, because it assumes the input features are independent.

4. No Laplace Estimators
   Class 0: tested 2, correctly classified 2
   Class 1: tested 2, correctly classified 2
   Overall: tested 4, correctly classified 4
   Accuracy: 1.000000

   With Laplace Estimators
   Class 0: tested 2, correctly classified 2
   Class 1: tested 2, correctly classified 2
   Overall: tested 4, correctly classified 4
   Accuracy: 1.000000

5. No Laplace Estimators
   Class 0: tested 48, correctly classified 52
   Class 1: tested 76, correctly classified 83
   Overall: tested 124, correctly classified 135 Accuracy: 0.918519

   With Laplace Estimators
   Class 0: tested 48, correctly classified 52
   Class 1: tested 76, correctly classified 83
   Overall: tested 124, correctly classified 135
   Accuracy: 0.918519

6. No Laplace Estimators
   Class 0: tested 10, correctly classified 15
   Class 1: tested 135, correctly classified 172
   Overall: tested 145, correctly classified 187 Accuracy: 0.775401

   With Laplace Estimators
   Class 0: tested 10, correctly classified 15
   Class 1: tested 130, correctly classified 172
   Overall: tested 140, correctly classified 187 Accuracy: 0.748663

7. Laplace Estimators did not change the accuracy of the vote data, but decreased the accuracy for the heart data. In general, Laplace Estimators improve accuracy when test data produce unrepresentative MLE estimates, such as an MLE estimate of 0 for an event that can actually occur. Otherwise, Laplace Estimators are not better than MLEs for classification accuracy.

```
/* CS109 HW5.1: Naive Bayes
* ------------------------
* This file contains starter code for your Naive Bayes classifier. Your
  job
* is to read in the data file and implement the Naive Bayes 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.
 */
import java.util.*; import
java.io.*;

public class NaiveBayes
{
      String fileName;
      Scanner input;
      Table[] table; //table for Y and each Xi, taken from training data
int[] classVar; //instances of Y=0 and Y=1 in training data
      int[][] accuCount; //cc[y][0] is number of successes for Y=y
                                    //cc[y][1] is number of tests Y=y

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

            readTrainingInput();

            System.out.println("No Laplace Estimators");
calcResults();
            reportResults();

            System.out.println("\nWith Laplace Estimators");
            toggleLaplace();
calcResults();
            reportResults();

//          sanity check for P(X=1 | Y=1)
//          for(int i = 0; i < table.length; i++)
//          {
//                double py1 = 1-
(double)classVar[0]/(classVar[0]+classVar[1]);
//                System.out.println(i+1 + "  " + table[i].getMLE(1,
1)/py1);
//
//          }
      }
	public void toggleLaplace()
	{
//		System.out.println(classVar[0] + "\t" + classVar[1]);
		for(int i = 0; i < table.length; i++)
		{
			table[i].toggleLaplace();
		}
		//update the number of Y=0 and Y=1
		classVar[0] += 2;//limited to binary variables
		classVar[1] += 2;
//		System.out.println(classVar[0] + "\t" + classVar[1]);

	}

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

	public void calcResults()
	{
		try {
			input = new Scanner(new File("src/PC-datasets/" + 
fileName + "-test-PC.txt"));
		} catch (FileNotFoundException e) {
e.printStackTrace();
		}
		int vectorLength = input.nextInt();
	int numVectors = input.nextInt();
		accuCount = new int[2][2];//limited to binary variables
		int[] x = new int[vectorLength]; //input vector

		for(int i = 0; i < numVectors; i++)
		{
			for(int j = 0; j < vectorLength-1; j++)
			{
				x[j] = input.nextInt();
			}
			x[vectorLength-1] = 
Integer.parseInt(input.next().substring(0, 1));
//		System.out.print(i + ":  ");
		int yhat = calcYhat(x);
//		System.out.println("Yhat = " + yhat);
		int y = input.nextInt();
```

```
            accuCount[y][1]++;
            if(y == yhat)
            {
//                System.out.println("Accurate");
                  accuCount[y][0]++;
            }
            else
            {
//                 System.out.println("Inaccurate");
            }
         }




//       for(int i = 0; i < table.length; i++)
//           table[i].toggleLaplace();
//       System.out.println("with  laplace:  Yhat  =  "  +
      }  calcYhat(x));

      public int calcYhat (int[] x)
      {
            double py0 = (double)classVar[0]/(classVar[0]+classVar[1]);
//P(Y=0)
            double py1 = 1 - py0; //P(Y=1)

//          double p0 = Math.log(py0);
//          for(int i = 0; i < table.length; i++)
//          {
//                p0 += Math.log(table[i].getMLE(x[i], 0)) -
Math.log(py0);
//          }
//          double p1 = Math.log(py1);
//          for(int i = 0; i < table.length; i++)
//          {
//                 p1 += Math.log(table[i].getMLE(x[i], 1)) -
Math.log(py1);
//          }

            //P(X, Y=0)
            double p0 = py0;
            for(int i = 0; i < table.length; i++)
            {
                  p0 *= table[i].getMLE(x[i], 0) / py0;
            }
            //P(X, Y=1)
            double p1 = py1;
            for(int i = 0; i < table.length; i++)
            {
                  p1 *= table[i].getMLE(x[i], 1) / py1;
            }
```

```
//          System.out.printf("P(X, Y=1)=%.10f\tP(X, Y=0)=%.10f\n", p1,
p0);
            if(p1 < p0)
            {
                  return 0;
            }
            return 1;
      }

      public void readTrainingInput()
      {
            try {
                  input = new Scanner(new File("src/PC-datasets/" +
fileName + "-train-PC.txt"));
            } catch (FileNotFoundException e) {
e.printStackTrace();
            }
            classVar = new int[2];
int vectorLength = input.nextInt();
int numVectors = input.nextInt();
table = new Table[vectorLength];
                  for(int i = 0; i < table.length;
            i++) {
                  table[i] = new Table(2, 2);//limited to binary variables
            }
            for(int i = 0; i < numVectors; i++)
            {
                  int[] x = new int[vectorLength];
                  for(int j = 0; j < x.length-1; j++)
                  {
                        x[j] = input.nextInt();
                  }
                  //last x-value has colon attached to it
                  x[vectorLength-1] =
Integer.parseInt(input.next().substring(0, 1));
                  int y = input.nextInt();
classVar[y]++;
                  for(int j = 0; j < x.length; j++)
                  {
                        table[j].add(x[j], y);
                  }
            }

      }

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

```
}


public class Table
{
      int[][] table;
      boolean laplace;

      public Table(int i, int j)
      {
            table = new int[i][j];
            laplace = false;
      }

      public void toggleLaplace()
      {
            int d = 1;
if(laplace)
            {
                  d = -1;
            }
            for(int i = 0; i < table.length; i++)
            {
                  for(int j = 0; j < table[i].length; j++)
                  {
                        table[i][j] += d;
                  }
            }
            laplace = !laplace;
      }

      public void add(int i, int j)
      {
            table[i][j]++;
      }

      public int get(int i, int j)
      {
            return table[i][j];
      }

      public double getMLE(int i, int j)
      {
            return (double)table[i][j] / sum();
      }

      public void print()
      {
            for(int i = 0; i < table.length; i++)
            {
                  for(int j = 0; j < table[i].length; j++)
```

```
                {
                        System.out.print(table[i][j] + "   ");
                }
                System.out.println();
        }
    }

    private int sum()
    {
        int res = 0;
        for(int i = 0; i < table.length; i++)
        {
            for(int j = 0; j < table[i].length; j++)
            {
                res += table[i][j];
            }
        }
        return res;
    }
}
```