Homework 1

Part A: due by 11:59pm Feb 10 on Blackboard

Part B: due by 11:59pm Feb 13 on erdos

A. Mathematics:

1. Consider a Bayesian classification problem where we wish to determine if a recently-registered user of our Fordham network is a student, a faculty member, or a dangerous hacker. We use four features - x_1 (VisitFrequency), x_2 (LoginLocation), x_3 (LoginDuration), x_4 SoftwareUsed . Each feature takes on one of five values, shown below:

VisitFrequency	Never	Monthly	Daily		
LoginLocation	OnCampus	InCity	USA	OutsideUSA	In State
LoginDuration	FewMinutes	FewDays	FewHours		
SoftwareUsed	PythonShell	Tableau	Hadoop	top	

We classify each user as one of three classes: y^i=Hacker, y^i=Student, or y^i=Faculty. Based on a large training set, we wish to estimate all joint probability **likelihoods**, e.g.,
P(x_1=Monthly,x_2=InCity,x_3=FewDays,x_4=top | y=Hacker),
P(x_1=Daily,x_2=USA,x_3=FewHours,x_4=Tableau | y=Hacker).

a) Assuming the features **are not** independent, how many total parameters need to be estimated, accounting for classifying students, faculty, and hackers?

b) Assuming the features **are** independent, how many total parameters need to be estimated, accounting for classifying students, faculty, and hackers?

Now assume we only classify subjects using the first two features, and we replace the discrete feature values with numbers:

VisitFrequency	0 (Never)	1 (Monthly)	2 (Daily)		
LoginLocation	0 (OnCampus)	1 (InCity)	2 (USA)	3 (OutsideUSA)	4 (In State)

We use a joint Gaussian likelihood for the probability of the two features for each class y: P(x_1,x_2|y) , and we also will estimate a prior probability for each of the three user classes.

~~c) Assuming x_1 and x_2 are not independent, how many parameters need to be learned to compute the likelihood P(x_1, x_2 | y)~~

d) Assuming x_1 and x_2 are independent, how many parameters need to be learned to compute the posterior probability P(y | x_1, x_2)

2. I have written a classifier to determine if my dog is sick or healthy. I record the sounds my dog makes once a minute for six minutes, obtaining six sound measurements s_1 , s_2 , … , s_6 . At each time, there is a likelihood my dog will make the sounds: bark, no-sound, pant, whimper:

S	P(s|y=sick)	P(s|y=healthy)
Bark	0.15	0.4
No-sound	0.4	0.3
Pant	0.25	0.1

We compute P(y | s_1, …, s_6) = P(y) $\prod_j P(s_j|y)$

Note, P(y=healthy)=0.8

Provide the y=sick and y=healthy likelihoods and the max-posterior classification for each of the following sound recordings

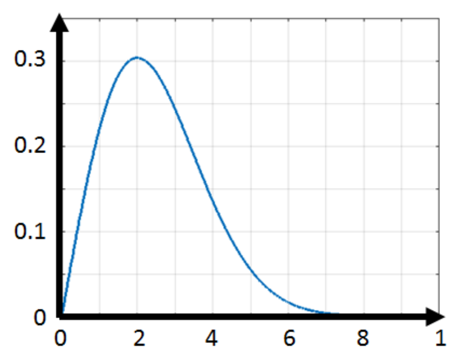
a) s_1 = No-sound , s_2 = Bark , s_3 = No-sound, s_4 = Bark, s_5 = Bark, s_6 = Pant

b) s_1 = Bark, s_2 = Bark , s_3 = Whimper, s_4 = No-sound, s_5 = Bark, s_6 = No-sound

c) s_1 = Whimper, s_2 = Whimper, s_3 = Bark, s_4 = No-sound, s_5 = Bark, s_6 = No-sound

3. In lecture, we used the Gaussian probability function to express the likelihood of light-intensity conditioned on the weather being Cloudy, Eclipse, or Non-cloudy (clear skies). However, the Gaussian function allows for both positive **and negative** light intensities. Let us instead consider the Rayleigh probability as the likelihood for P(light|weather):

$$P(light|w) = \frac{light}{{\sigma_w}^2} e^{-light^2/(2{\sigma_w}^2)}$$



This function **only** is defined for light ≥ 0 . It has one parameter, σ_w, determined by weather.

a) Assuming $\sigma_w = \cancel{3}$ **2**, compute:
  - $P(light = 1|\sigma_w = 2)$
  - $P(light = 3|\sigma_w = 2)$

b) We have 200 measurements of light-intensities $light^1, light^2, \cdots, light^{200}$ during a snow storm and we wish to estimate the corresponding parameter σ_{snow}.

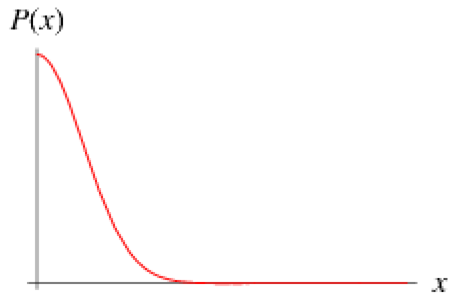
Derive the maximum likelihood estimate estimate for $\boldsymbol{\sigma_{snow}}$. Start with

$$P(D|\theta) = \prod_i P(light^i|\sigma_{snow})$$

Show at least three mathematical steps to get your estimate of $\boldsymbol{\sigma_{snow}}$.

c) Let us assume we have a prior probability on σ values defined by the "half-normal distribution" P(σ):

$$P(\sigma) = \frac{\sqrt{2}}{3\sqrt{\pi}} e^{-\frac{\sigma^2}{18}} \qquad \sigma \geq 0$$



This function **only** is defined for $\sigma \geq 0$.

Technical note: For simplicity I use a very specific version of the half-normal distribution here with no extra hyper-parameters. In this case, we are computing the value of σ, not using it to control the shape of the half-normal distribution.

Derive the maximum a posteriori estimate for σ_{snow}. Show at least three mathematical steps to get your estimate of σ_{snow}

4. You have developed a program that determines from a user's movie-watching history whether s/he is an adult or a teenager. We know 25% of users are teenagers. If user X is a teenager, the program will say so with probability 90%. If user X is an adult, the program will say s/he is an adult with probability 60%.

Assume that the program says user X is a teenager. What is the probability s/he is actually an adult?

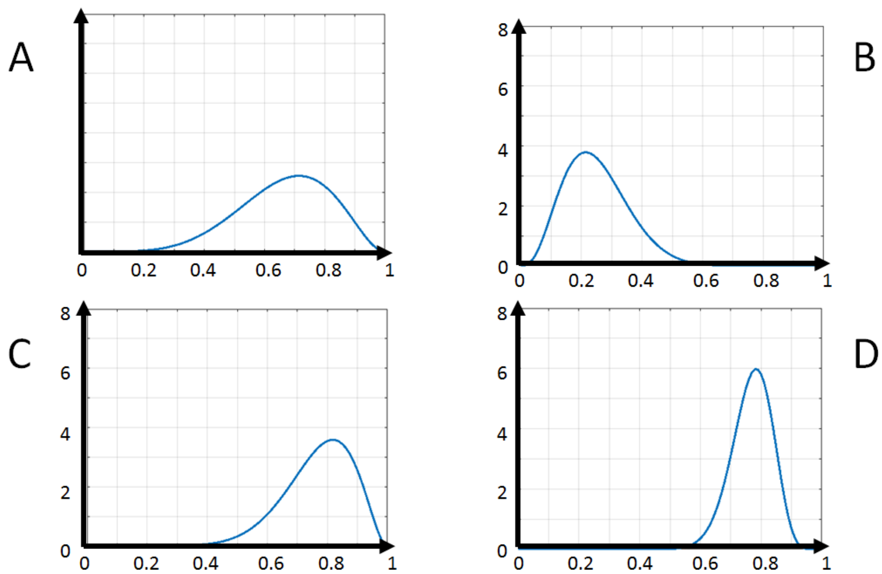
~~5. A "positive definite matrix" is any matrix A such that every vector x yields a matrix product:~~ $\cancel{x^T A x \geq 0}$~~. Given the matrix~~ $\cancel{A = \begin{bmatrix} 1 & -4 & 0 \\ 1 & 0 & -1 \\ 2 & 0 & 1 \end{bmatrix}}$~~, find a vector that produces a negative output~~ $\cancel{x^T A x}$~~.~~

6. In class we discussed a family of distributions with input between 0 and 1; this family was from the Beta distribution $(x|\alpha, \beta)$:

$$P(x|\alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$$

Where $0 \leq x \leq 1$

a) What α, β combination will produce each of the probabilities below. (At least three of the options below are valid)
Option I : $\alpha = 10, \beta = 3$
Option II : $\alpha = 6, \beta = 3$
Option III : $\alpha = 5, \beta = 5$
Option IV : $\alpha = 30, \beta = 9$
Option V: $\alpha = 1, \beta = 500$



b) The Beta distribution $P(x|\alpha = 20, \beta = 5)$ reaches a maximum probability P_{max} when $x = x_{max}$. How can we define a distribution that also reaches maximum probability when $x = x_{max}$, but the maximum probability is now larger (e.g., max probability is now $3P_{max}$, where P_{max} was defined in the past sentence).

B. Programming

<u>Detailed submission instructions:</u> Code must be left in your `private/CIS5800/` directory. Include all function definitions and your answers to questions 1 and 6 (as comments) in the file `hw1.py` . For this homework, we will require several numpy array inputs.

Now, on to the programming:
In class, we discussed classification using Max Likelihood and using Max Posterior. For this assignment, you will create the Max Posterior/Bayes classifier to label online social network users based on their posting history. Specifically, you will determine if each user is based in Cairo, Frankfurt, Philadelphia, or Seoul. You will make this determination based on a single feature x – the time the user most frequently posts online.

Accessing our data
The file hw1data.mat is available on our website (and on erdos using `cp ~dleeds/MLpublic/hw1data.mat .`) Load this file into your Python session to get access to the trainData and testData numpy arrays . For each array, each row is one example data point. The first column represents the user class – 0 for Cairo, 1 for Frankfurt, 2 Philadelphia, and 3 Seoul – and the second column represents the corresponding postingTime (most common posting time) for the example data point (user).

Note postingTime will be determined based on the current time in New York City. Also, time will be recorded on the 24-hour clock, where 0 is midnight, 430 is 4:30am, and 1750 is 5:50pm.

Programming assignments:
1. Inspect the distribution of the `postingTime` feature for each class and determine if it follows a Gaussian or a Uniform distribution. (Note, uniform was shown earlier in Lecture 1.) Record this result as a comment in hw1.py .

You can inspect the distribution of values in a list/vector of numbers `vector` through a histogram.

```
import matplotlib.pyplot as plt
plt.hist(vector)
plt.show()
```

Regardless of our results from question 1, we will assume all distributions really are Gaussian for the rest of this assignment.

2. Write a function called `learnParams` that takes in a data set and returns the learned mean and standard deviation for each class. Specifically, the function will be called as:

```
params=learnMean(Data)
```

where `Data` is a numpy array with shape (N,2) where N is the number of data points and `params` is a numpy array with shape (M,2) where there are M classes, `params[i,0]` is the mean for class i and `params[i,1]` is the standard deviation of class i.

learnParams(np.array([[0,200],[1,1500],[0,300],[1,1700],
[0,400],[1,1300]])
would return `np.array([[300,100],[1500,200]])`

3. Write a function called `learnPriors` that takes in a data set and returns the prior probability of each class. Specifically, the function will be called as:
   priors=learnPriors(Data)
where `Data` is a numpy array with shape (N,2) where N is the number of data points and `priors` is a numpy array with shape (M) where there are M classes, `priors[i]` is the estimated prior probability for class i .

```
learnPriors(np.array([[0,200],[1,1500],[0,300],[1,1700],
[0,400],[1,1300]])
```
would return `np.array([0.5,0.5])`

4. Write a function called `labelBayes` that takes in posting times for multiple users as well as the learned parameters for the likelihoods and prior, and return the most probably class for each user. Specifically, the function will be called as:
```
labelsOut = labelBayes(postTimes,paramsL,priors)
```
where `postTimes` is a numpy array of shape (K) containing post times for K users, `paramsL` is a numpy array with shape (M,2) matching the description of the output for `learnParams` and `priors` is a numpy array with shape (M) matching the description of the output for `learnPriors` ; `labelsOut` is a numpy array with shape (K) containing the most probable label for each user, where `labelsOut[j]` corresponds to `postTimes[j]` . Labels are computed using the Gaussian Bayes classifier!

```
labelBayes(np.array([430,2110,845]),
        np.array([[300,100],[1500,250]]),np.array([0.2,0.8]))
```
would return `np.array([0,1,1])`

5. Write a function called `evaluateBayes` that takes in classifier parameters for likelihoods and priors, and a set of labels and feature values, and returns the percent of input data correctly classified. Specifically, the function will be called as:
```
accuracy = evaluateBayes(paramsL,priors,testData)
```
where `paramsL` is a numpy array with shape (M,2) matching the description of the output for `learnParams` and `priors` is a numpy array with shape (M) matching the description of the output for `learnPriors` , `testData` is a numpy array with shape (J,2) where `testData[j,0]` contains the label of data point j and `testData[j,1]` contains the feature value (posting time) for data point j ; `accuracy` is a number between 0 and 1 indicating the accuracy of the Gaussian Bayes classifier using the specified parameters on the specified input data set.

```
evaluateBayes(np.array([[300,100],[1500,200]]),
              np.array([0.2,0.8]),
              np.array([0,430],[1,2110],[0,845]))
```

would return 0.6666

6. Our definition for time-of-day for user posting is not truly linear or continuous. 859 (8:59am) is followed by 900 (9:00am), skipping the integers 860, 861, through 899. 2359 (11:59pm) is followed by 0 (midnight) – it is much closer in time to midnight than it is to 2030 (8:30pm), while the integer 2359 is much closer to 2030 than it is to 0.

Rewrite either `learnParams` (from question 2) or `labelBayes` (from question 4) to more-naturally reflect the circular nature of the clock, and to account for skips in integers for each hour. Call this function `learnParamsClock` or `labelBayesClock`.

Explain the reasoning of your approach in a comment in your function.

There are many reasonable ways to answer this question!