Homework 3

Kevin Chao

November 1, 2019

Question 1:

a. Let X be the time taken by the TA helping a student, where X is an exponential distribution. If there are 4 students, then let the time taken by each student be X_1, X_2, X_3, X_4. Each time taken is independent and thus the sum is a gamma distribution

$$Y = \sum_{i=1}^{4} X_i \sim \Gamma(4, 0)$$

Since the mean of an individual time is 8 minutes, $\theta = \frac{1}{8}$ since $E(X_i)$ for all i is 8. And thus, $E(Y) = \frac{4}{\theta} = 32$

b. Standard deviation is typically $\sqrt{Var[X]}$, and if $\theta = \frac{1}{8}$, then the $Var[X_i]$ for all i is $\frac{1}{\frac{1}{8^2}} = 64$. The Var[Y] = 256. The standard deviation would then be $\sqrt{Var[Y]} = \sqrt{256} = 16$.

c. $P(X_i \leq 10) = \int_0^{10} X_i = .71350$
$\prod_{i=1}^{4} = (.71350)^4 = .25916$

d. One important detail to note is that this problem in specific is an exponential distribution. With that, that means that this distribution is memoryless. With that that means that

$$Pr(X > s + 10 | X > 10) = Pr(X > s)$$

This means that the probability of s is the same as if time started at 0 even after 10 minutes, which means that neither the mean nor standard deviation changes. Which means

$$\mu = 8$$
$$\sigma = 8$$

Question 2:

a.

$$\begin{aligned}
z =& \frac{18-13}{2} \\
=& 2.5 \\
P(z \leq 2.5) =& .99379 \\
P(z > 2.5) =& 1 - P(z \leq 2.5) \\
=& .00621
\end{aligned}$$

b. Given that $\mu = 13$ seconds and $\sigma = 2$ seconds:

$$\begin{aligned}
z(10) =& \frac{10-13}{2} \\
=& -1.5 \\
P(z \leq -1.5) =& \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-1.5} e^{-\frac{t^2}{2}} \\
=& .06681 \\
z(16) =& \frac{16-13}{2} \\
=& 1.5 \\
P(z \leq 1.5) =& \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{1.5} e^{-\frac{t^2}{2}} \\
=& .93319 \\
P(-1.5 \leq z \leq 1.5) =& P(z \leq 1.5) - P(z \leq -1.5) \\
=& .93319 - .06681 \\
=& .86438
\end{aligned}$$

c. Essentially, the question wants $P(Z \geq z) = .01$, so it can then be thought of as

$$P(Z \leq z) = 1 - .01 = .99$$

Since (a) provides that 2.5 is .99379, it must be lower than 2.5 and bigger than 1.5. Testing this using python, here is the code below to help

```
f = lambda x: np.exp(-(x**2)/2)
print(integrate.quad(f,np.NINF,2.33)[0]/sqrt(2*pi))
```

I used z=2.25 to lower the potential limit since z=1.5 is only .93319 . After a few more testings, z=2.33 is the closest approximation to the second decimal i could find where $P(Z \leq z) = .99$. Now, given z and σ , we can find the mean:

$$
\begin{aligned}
2.33 &= \frac{15-\mu}{2} \\
4.66 &= 15-\mu \\
\mu &= 10.34
\end{aligned}
$$

Question 3:

a. Let's organize our data first. This problem is an exponential distribution, and it is given that $(X|Y=0) \sim exp(1)$ and $(X|Y=1) \sim exp(50)$. For this particular problem, $P(Y=0) = .5$ and $P(Y=1) = .5$. This particular problem wants to maximize the probability that the prediction is correct, or in other words *minimize the probability of error*. First, must find $P(Y=0|X=x)$ and $P(Y=1|X=x)$.

$$
\begin{aligned}
P(Y=0|X=x) =& f_x(X|Y=0) \\
=& (1)e^{-1x} \\
P(Y=1|X=x) =& f_x(X|Y=1) \\
=& \frac{1}{50}e^{\frac{-x}{50}}
\end{aligned}
$$

And the factory manufactures correctly if:

$$P(Y=0|X=x) \geq P(Y=1|X=x)$$

Plugging in values for the inequality, the steps to solve c (including using natural log on both side) looks like this:

$$
\begin{aligned}
P(Y=0)(f_x(X|Y=0) \geq& P(Y=1)f_x(X|Y=1) \\
.5 * e^{-x} \geq& .5 * \frac{1}{50}e^{\frac{-x}{50}} \\
e^{-x} \geq& \frac{1}{50}e^{\frac{-x}{50}} \\
e^{\frac{-49x}{50}} \geq& \frac{1}{50} \\
\frac{-49x}{50} \geq& ln(\frac{1}{50}) \\
x \leq& -\frac{50}{49} * ln\frac{1}{50} \\
x \leq& 3.99186
\end{aligned}
$$

And thus, c as a threshold is = 3.99186 when P(Y=0) = .5

b. This part is similar to (a) except that the Probability of P(Y=0) = .99.

$$
\begin{aligned}
.99 * e^{-x} &\geq .01 * \frac{1}{50} e^{\frac{-x}{50}} \\
99 * e^{-x} &\geq 1 * 50 e^{\frac{-x}{50}} \\
e^{\frac{-49x}{50}} &\geq 99 * 50 \\
\frac{-49x}{50} &\geq ln(\frac{1}{4950}) \\
x &\leq \frac{-50}{49} * ln(\frac{1}{4950}) \\
x &\leq 8.68076
\end{aligned}
$$

Thus, c as a threshold is = 8.68076 when P(Y=0) =.99

c. This part is similar to (b) except that now there are weighted loss factors. If it wrongly decides that a defective (Y=1) is 'correct' (z=0), then it is 500 times more of a loss than when (Y=0,z=1). It'll look similar to (b) but with these weights taking place.

$$
\begin{aligned}
1 * .99 * e^{-x} &\geq .01 * 500 * .02 e^{\frac{-x}{50}} \\
.99 * e^{-x} &\geq .1 e^{\frac{-x}{50}} \\
99 * e^{-x} &\geq 10 e^{\frac{-x}{50}} \\
e^{\frac{-49x}{50}} &\geq \frac{10}{99} \\
\frac{-49x}{50} &\geq ln(\frac{10}{99}) \\
x &\leq \frac{-50}{49} * ln(\frac{10}{99}) x \leq \quad 2.33932
\end{aligned}
$$

Thus, c as a threshold is =2.33932 when Probability of P(Y=0) =.99 and the loss when L(1,0) = 500 and L(0,1) = 1

Question 4:

a. Let's write the conditional probability density function for $f_{X|Y}(x_i|1)$:

$$
f_{X|Y}(x_i|1) = \prod_{j=1}^{M} \frac{1}{\sqrt{2\pi\sigma_{1j}^2}} e^{-\frac{(x_{ij} - \mu_{1j})^2}{2\sigma_{1j}^2}}
$$

And now lets natural log both side.

$$ln(f_{X|Y}(x_i|1)) = \prod_{j=1}^{M} ln(\frac{1}{\sqrt{2\pi\sigma_{1j}^2}}e^{-\frac{(x_{ij}-\mu_{1j})^2}{2\sigma_{1j}^2}})$$

Since for numerical robustness, we must remove the exponential function (the base of natural logarithm e). To do that, first we must apply the product rule of logarithm, which states $log_a(xy) = log_a(x) + log_a(y)$. As a result of product ruling logarithm, it no longer multiplies for each different j, but rather it finds the sum for each different j. Applying these, the conditional probability is now:

$$ln(f_{X|Y}(x_i|1)) = \sum_{j=1}^{M} ln(\frac{1}{\sqrt{2\pi\sigma_{1j}^2}}) + ln(e^{-\frac{(x_{ij}-\mu_{1j})^2}{2\sigma_{1j}^2}})$$

And now you apply the power rule of logarithm.

$$ln(f_{X|Y}(x_i|1)) = \sum_{j=1}^{M} ln(\frac{1}{\sqrt{2\pi\sigma_{1j}^2}}) - \frac{(x_{ij}-\mu_{1j})^2}{2\sigma_{1j}^2}$$

This is the derivation for $ln(f_{X|Y}(x_i|1))$, and $ln(f_{X|Y}(x_i|0))$ is similar, but with the means and variance being 0j instead of 1j:

$$ln(f_{X|Y}(x_i|0)) = \sum_{j=1}^{M} ln(\frac{1}{\sqrt{2\pi\sigma_{0j}^2}}) - \frac{(x_{ij}-\mu_{0j})^2}{2\sigma_{0j}^2}$$

b. So first, the classifier is known as this:

$$ln(p_y(1)) + ln(f_{X|Y}(x_i|1)) > ln(p_y(0)) + ln(f_{X|Y}(x_i|0))$$

In this particular part, it is given that $p_y(1) = p_y(0) = \frac{1}{2}$, so the classifier can be reduced to:

$$ln(f_{X|Y}(x_i|1)) > ln(f_{X|Y}(x_i|0))$$

Also in this particular party, there is unit variance, which makes the derivation in (a) look like this:

$$ln(f_{X|Y}(x_i|1)) = \sum_{j=1}^{M} ln(\frac{1}{\sqrt{2\pi}}) - \frac{(x_{ij}-\mu_{1j})^2}{2}$$

The code to compute the log conditional densities is below:

```
x = -np.log(math.sqrt(2*math.pi))
y1Days, y0Days = np.array([]), y0Days = np.array([])
for i in range(0,len(testFeat)):
    yIs1, yIs0 = np.array([]), np.array([])
    for j in range(0,M):
        xij = testFeat[i][j]
        mu0j, mu1j =  muhat[0][j], muhat[1][j]
        numerator0 = (xij - mu0j)**2
        numerator1 = (xij - mu1j)**2
        total0 = x - (numerator0/2)
        total1 = x - (numerator1/2)
        yIs0 = np.append(yIs0, total0)
        yIs1 = np.append(yIs1, total1)
    y0Days = np.append(y0Days, np.sum(yIs0))
    y1Days = np.append(y1Days, np.sum(yIs1))
yHat = y1Days > y0Days
```

yHat is the array of classification on the test days. With this, here are the following information:

Accuracy = 58.3333%, False Alarms = 267, Missed Detections = 3

c. The variance parameters σ_{1j}^2 and σ_{0j}^2 can be found on python using numpy.var(), which finds the variance on the empirical distribution. The classifier should look like:

$$ln(f_{X|Y}(x_i|1)) > ln(f_{X|Y}(x_i|0))$$

With that, the code looks like this:

```
var = np.zeros((2,M))
var[0] = np.var(trainFeat0,axis=0)
var[1] = np.var(trainFeat1,axis=0)
y1Days, y0Days = np.array([]), np.array([])
for i in range(0,len(testFeat)):
    yIs1, yIs0 = np.array([]), np.array([])
    for j in range(0,M):
        x0 = -np.log(math.sqrt(2*math.pi*var[0][j]))
        x1 = -np.log(math.sqrt(2*math.pi*var[1][j]))
        xij = testFeat[i][j]
        mu0j, mu1j =  muhat[0][j], muhat[1][j]
        numerator0 = (xij - mu0j)**2
        numerator1 = (xij - mu1j)**2
        total0 = x0 - (numerator0/(2*var[0][j]))
        total1 = x1 - (numerator1/(2*var[1][j]))
        yIs0 = np.append(yIs0, total0)
```

```
        yIs1 = np.append(yIs1, total1)
    y0Days = np.append(y0Days, np.sum(yIs0))
    y1Days = np.append(y1Days, np.sum(yIs1))
yHat = y1Days > y0Days
```

With that, here is the classification information:

Accuracy = 73.9198%, False Alarms = 167, Missed Detections = 2

d. The probability of days have changed, based on the training examples as a fraction (i.e. # of Ozone Days/Total Days in Training Data). The code from part (c) still applies, but after that code, the following takes place:

```
#p_{Y}(1) and p_{Y}(0) of training examples
p0 = np.log((len(trainFeat0)/numTrain))
p1 = np.log((len(trainFeat1)/numTrain))
for i in range(0,len(testFeat)):
    y0Days[i] += p0
    y1Days[i] += p1
yHat = y1Days > y0Days
```

In this code, each log conditional density adds the respective probabilities ($ln(1167/1200)$ when Y=0 and $ln(33/1200)$ when Y=1) for each day. The classification information is:

Accuracy = .750000%, False Alarms = 158, Missed Detections = 4