MAD Exam 2019

MAD 2018
Department of Computer Science
University of Copenhagen

Exam number: 46

Version 1
Due: January 20th, 23:59

Contents

1. Exercise

(a)

To solve this, we need to realise that a pdf always has the following property: It's integral over the entire function is equal to 1. Since we know that

$$f(x) = 0, \text{ if } 0 > x, 5 < x$$

We can compute the definite integral between 0 and5, $\int_0^5 f(x)dx$ to find the integral. This means that we have to solve the following equation:

$$\int_0^5 c(5-x)dx = 1$$

First we will extend it to make it easier to work with

$$\int_0^5 5c - xcdx = 1$$

We know that definite integrals are solved as follows.

$$\int_a^b f(x)dx = F(b) - F(a)$$

First we find $F(x)$:

$$\int 5c - xcdx = 5xc - \frac{x^2c}{2}$$

Now we solve $F(5) - F(0)$:

$$(5 * 5c - \frac{5^2c}{2}) - (5 * 0c - \frac{0^2c}{2}) = 1$$

This can be simplified to:

$$12.5c = 1$$

We can then realise that for this to be true, $c = \frac{2}{25}$

(b)

First we will find the pdf of V. We know that to find the pdf, we have to compute the derivative of the cdf. To do this on a piecewise function, we compute the derivative of each piece:

$$f(v) = \frac{d}{dv}F(v) = \begin{cases} \frac{d}{dv}0 & if \quad v < 0 \\ \frac{d}{dv}v^5 & if \quad 0 \le v \le 1 \\ \frac{d}{dv}1 & if \quad v > 1 \end{cases} = \begin{cases} 0 & if \quad v < 0 \\ 5v^4 & if \quad 0 \le v \le 1 \\ 0 & if \quad v > 1 \end{cases}$$

To find the expectation, $\mathbb{E}V$ we have to use the following formula, since V is a continuous random variable. We know that $\int f(v)dv = F(v)$:

$$\mathbb{E}V = \int vf(v) = \begin{cases} \int 0vdv & if \quad v < 0 \\ \int 5v^5dv & if \quad 0 \leq v \leq 1 \\ \int 0vdv & if \quad v > 1 \end{cases} = \begin{cases} 0 & if \quad v < 0 \\ \frac{5v^6}{6} & if \quad 0 \leq v \leq 1 \\ 0 & if \quad v > 1 \end{cases}$$

(c)

To prove this, we first have to set an $\epsilon\varepsilon > 0$. We know the following:

$$\mathbb{P}(|Y_n - Y| \geq \varepsilon) \longrightarrow 0$$

$$\mathbb{P}(|Z_n - Z| \geq \varepsilon) \longrightarrow 0$$

And we want to prove that:

$$\mathbb{P}(|(Y_n - Y) + (Z_n - Z)| \geq \varepsilon) \longrightarrow 0$$

We start off by writing the following, which we know is true, since the value of ε is arbitrarily small:

$$\mathbb{P}\Big(|Y_n - Y| \geq \frac{\varepsilon}{2}\Big) \longrightarrow 0$$

$$\mathbb{P}\Big(|Z_n - Z| \geq \frac{\varepsilon}{2}\Big) \longrightarrow 0$$

We can now write the following:

$$\mathbb{P}\Big(|Y_n - Y| \geq \frac{\varepsilon}{2}\Big) + \mathbb{P}\Big(|Z_n - Z| \geq \frac{\varepsilon}{2}\Big) \longrightarrow 0$$

We can then write the following, and also see that it still approaches 0:

$$\mathbb{P}\Big(|Y_n - Y| \geq \frac{\varepsilon}{2}\Big) + \mathbb{P}\Big(|Z_n - Z| \geq \frac{\varepsilon}{2}\Big) \geq \mathbb{P}\Big(|Z_n - Z| \geq \frac{\varepsilon}{2} \vee |Y_n - Y| \geq \frac{\varepsilon}{2}\Big)$$

$$\mathbb{P}\Big(|Z_n - Z| \geq \frac{\varepsilon}{2} \vee |Y_n - Y| \geq \frac{\varepsilon}{2}\Big) \longrightarrow 0$$

We can see that this probability *must* be greater than $\mathbb{P}(|(Y_n - Y) + (Z_n - Z)| \geq \varepsilon)$, since for this to be true at least one of the events in the previous probability have to happen. Because of this, we can write the following:

$$\mathbb{P}(|(Y_n - Y) + (Z_n - Z)| \geq \varepsilon) \leq \mathbb{P}\Big(|Z_n - Z| \geq \frac{\varepsilon}{2} \vee |Y_n - Y| \geq \frac{\varepsilon}{2}\Big)$$

We can now say that since

$$\mathbb{P}\Big(|Z_n - Z| \geq \frac{\varepsilon}{2} \vee |Y_n - Y| \geq \frac{\varepsilon}{2}\Big) \longrightarrow 0$$

and

$$\mathbb{P}(|(Y_n - Y) + (Z_n - Z)| \geq \varepsilon) \leq \mathbb{P}\Big(|Z_n - Z| \geq \frac{\varepsilon}{2} \vee |Y_n - Y| \geq \frac{\varepsilon}{2}\Big)$$

The following must be true:

$$\mathbb{P}(|(Y_n - Y) + (Z_n - Z)| \geq \varepsilon) \longrightarrow 0$$

This is exactly what we wanted to prove, since:

$$Y_n + Z_n \xrightarrow{\mathbb{P}} Y + Z = \mathbb{P}(|(Y_n - Y) + (Z_n - Z)| \geq \varepsilon) \longrightarrow 0$$

2. Exercise

(a)

First, we will find the likelihood, which is the same as the *joint pdf*, which is defined as thus:

$$f_\theta^{joint}(x_1, ..., x_n) = \prod_{i=1}^{n} f_\theta(x_n)$$

We can then figure out that the likelihood is:

$$f_\beta^{joint}(x_1, ..., x_n) = \begin{cases} \prod_{i=1}^{n} f_\beta(x_n) & \text{if } 0 \leq x \leq \beta \\ 0 & otherwise. \end{cases}$$

$$f_\beta^{joint}(x_1, ..., x_n) = \begin{cases} \prod_{i=1}^{n} \frac{2}{\beta^2} \cdot (\beta - x_n) & \text{if } 0 \leq x \leq \beta \\ 0 & otherwise. \end{cases}$$

We then have to figure out the maximum likelihood estimator. This is done using the following formula:

$$\hat{\theta}_n^{ML}(x_1, ..., x_n) = \underset{\theta}{argmax} \prod_{i=1}^{n} f_\theta(x_n)$$

We insert the previous formula with the given values for X:

$$\hat{\beta}_n^{ML}(x_1, ..., x_n) = \underset{\beta}{argmax} \prod_{i=1}^{n} f_\beta(x_n) = \underset{\beta}{argmax} \left(\frac{2}{\beta^2} \cdot (\beta - 3) \cdot \frac{2}{\beta^2} \cdot (\beta - 4) \right)$$

First we rewrite it to make our lives easier:

$$\frac{4(\beta - 4)(\beta - 3)}{\beta^4} = \frac{4}{\beta^2} - \frac{28}{\beta^3} + \frac{48}{\beta^4}$$

Now we have to differentiate it to find the local extrema:

$$\frac{d}{d\beta}\frac{4}{\beta^2}-\frac{28}{\beta^3}+\frac{48}{\beta^4}=\frac{-8\beta^2+84\beta-192}{\beta^5}$$

Then we have to set this to 0 and solve:

$$\frac{-8\beta^2+84\beta-192}{\beta^5}=0$$

$$-8\beta^2+84\beta-192=0$$

And find that there are two solutions:

$$\beta_1=\frac{21}{4}-\frac{\sqrt{57}}{4}$$

$$\beta_2=\frac{21}{4}+\frac{\sqrt{57}}{4}$$

Then, to find whether they are maxima or minima, we compute derive again, and solve with $\beta=\beta_1$ and $\beta=\beta_2$:

$$-16\left(\frac{21}{4}-\frac{\sqrt{57}}{4}\right)+84=-16\sqrt{57}$$

$$-16\left(\frac{21}{4}+\frac{\sqrt{57}}{4}\right)+84=16\sqrt{57}$$

Since $f''(\beta_1)<0$, we find that the $\underset{\beta}{argmax}$ is at:

$$\beta=\frac{21}{4}+\frac{\sqrt{57}}{4}$$

We find that the approximate value of this is ≈ 7.137, and verify that it satisfies $x\leq\beta$

$$x_1=3<x_2=4\leq\beta\approx 7.137$$

(b)

So, the parameters we are given in this exercise actually require us to go through extra, unnecessary steps, to perform a two-tailed hypothesis test, when a one-tailed test could have been enough. The reasoning for this is that we do not want to look at whether the machine predicts *differently* from 0, we want to look at whether the machine predicts at a rate *better than* 0.5. The smarter choice of H_0 and H_1 would be $H_0:\theta\leq 0.5$ and $H_1:\theta>0.5$, since we don't care if the machine predicts at a rate worse than 0.5 (This is arguable, if the machine predicts at $\theta<0.5$, does it not also predict better if we reverse it? For this test to be good, we assume that this is not the case). With these H_0 and H_1 values, we would also set a different rejection region: $\mathcal{R}=[0,\alpha]$, where α is the significance level.
To perform this statistical test, we have to go through the six steps:

- A model:
  The model is given as $X \sim Bin(n, \theta)$, with $n = 20$
- A hypothesis:
  A hypothesis is given as well: $H_0 : \theta = 0.5, H_1 : \theta \neq 0.5$
- A test statistic and distribution:
  We use a binomial test to find the probability that a $\theta = 0.5$ is correct when $n = 20$ and $X = 13$, given a binomial distribution:

  $$T = \sum_{k=X}^{n} \left(\binom{n}{k} \cdot \theta^k (1-\theta)^{n-k} \right)$$

- A significance level:
  The significance level is also given as 0.05
- A rejection region:
  A rejection region is given as:

  $$\mathcal{R} = \{0, ..., \alpha\} \cup \{20 - \alpha, ..., 20\}$$

  So we have to find an α. We do this by solving the following:

  $$\underset{a}{argmax} \left(\sum_{k=0}^{a} \left(\binom{n}{k} \theta^k (1-\theta)^{n-k} \right) \leq 0.025 \right) = 5$$

  We thus find that $\mathcal{R} = \{0, ..., 5\} \cup \{15, ..., 20\}$
- Compute test statistic:
  We insert the values into the test statistic to see whether the given data is within the rejection region:

  $$T = \sum_{k=13}^{n=20} \left(\binom{20}{k} \cdot 0.5^k (1-0.5)^{20-k} \right) = \frac{34495}{262144} \approx 0.1316$$

  We can see that this is not within the rejection region:

  $$T \approx 0.1316 \notin \mathcal{R} = \{0, ..., 0.05\} \cup \{19.95, ..., 20\}$$

  Thus, we can attribute the higher than average number of correct guesses to chance, not the predictions.

3. Exercise

(a)

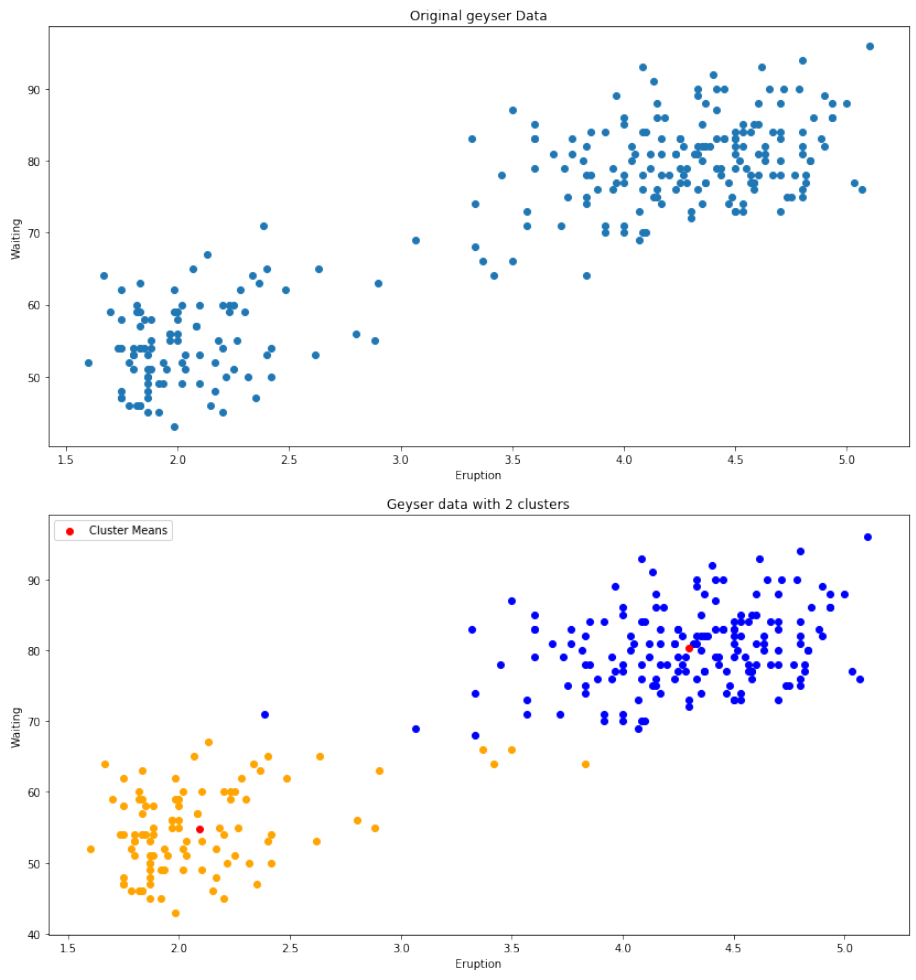
This exercise has been completed in the attached *K_Means.ipynb* Jupyter Notebook. The most relevant bits of source code can be seen here as well:

```
data = np.loadtxt("old_faithful.csv", delimiter=',', skiprows=1)

meanData = KMeans(n_clusters=2, max_iter=30, seed=0)
meanData.fit(data)

for i in range(data.shape[0]):
    if (meanData.cluster_assignments[i] == 0):
        plt.scatter(data[i,0],data[i,1], color='blue')
    else:
        plt.scatter(data[i,0],data[i,1], color='orange')
```

The 2 cluster means, calculated using the Jupyter Notebook, look as follows:

$$[4.29793023, 80.28488372]$$
$$[2.09433, 54.75]$$

And the resulting images, with, with each cluster in their own colour, and the cluster means plotted as singular red dots:



(b)

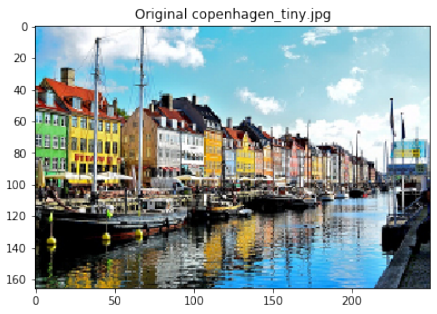
This exercise has been completed in the attached *K_Means.ipynb* Jupyter Notebook. The most relevant bits of source code can be seen here as well, though all comments are left out. The final version is also different, as the function has to accommodate larger images. The function remains unchanged for this image though:

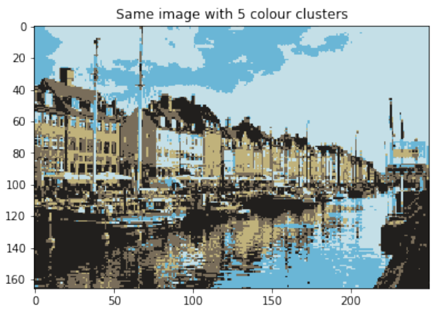
```
def makeNewImg(image, n_clusters, max_iter, seed):
    imgx   = image.shape[0]
    imgy   = image.shape[1]
    imgRGB = image.shape[2]
    model  = KMeans(n_clusters, max_iter, seed)
    image  = image.reshape((imgx*imgy),imgRGB)
    model.fit(image)

    retval = np.zeros(((imgx*imgy),imgRGB))

    for i in range(n_clusters):
        for j in range(imgRGB):
            model.cluster_means[i,j] =
                model.cluster_means[i,j]/255

    for i in range(retval.shape[0]):
        retval[i] =
            model.cluster_means[model.cluster_assignments[i]]

    retval = retval.reshape((imgx,imgy,imgRGB))
    return retval, model.cluster_means
```

The 5 cluster means, calculated using the Jupyter Notebook, look as follows. It should be noted that they are all RGB values, but Python wants RGB values in one of two forms; floats between 0 and 1, or integers between 0 and 255. Here they are floats between 0 and 1:

$$[0.41773344, 0.71691352, 0.84028094]$$
$$[0.13106363, 0.12181295, 0.11628755]$$
$$[0.47558986, 0.42848816, 0.35511877]$$
$$[0.76865056, 0.87663444, 0.90978927]$$
$$[0.75330927, 0.69978601, 0.48412478]$$

The assigned cluster for each pixel has been calculated, and the new image has been plotted:





(c)

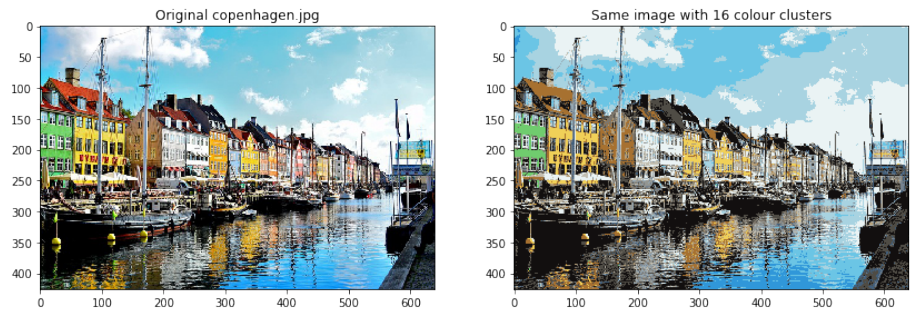
This exercise has been completed in the attached *K_Means.ipynb* Jupyter Notebook. The most relevant bits of source code can be seen here as well, though all comments are left out. It should be noted that the code for this is the exact same as in the previous subexercise, except the "model.fit(image)" line has been exchanged for the following. The full source code is in the notebook:

```
if (imgx*imgy > 50000):
        modLen = 5000

        temp = np.arange(0,(imgx*imgy),1).reshape((imgx*imgy))
        smallImgInd = np.random.choice(temp, modLen, False)

        smallImg = image[smallImgInd]

        model.fit(smallImg)
        model.cluster_assignments =
            model.assign_to_clusters(image, model.cluster_means)
    else:
        model.fit(image)
```

The 16 cluster means, calculated using the Jupyter Notebook, look as follows. It should be noted that the cluster means have been calculated by choosing 5000 random pixels in the image, and fitting the model to those instead of the entire image. Doing it this way should only have a minimal effect on the result, if 5000 pixels are enough to find a representative of the image. It should also be noted that they are all RGB values, but Python wants RGB values in one of two forms; floats between 0 and 1, or integers between 0 and 255. Here they are floats between 0 and 1:

$$
\begin{gathered}
[0.49441064, 0.46930358, 0.42724814] \\
[0.31073644, 0.29808085, 0.27156238] \\
[0.66100118, 0.82807700, 0.88582735] \\
[0.44086687, 0.80361197, 0.43168215] \\
[0.76419118, 0.74735294, 0.69117647] \\
[0.07393128, 0.06195302, 0.06012036] \\
[0.65701033, 0.44427577, 0.18943707] \\
[0.58076367, 0.58930857, 0.57473684] \\
[0.94262126, 0.81946042, 0.34577621] \\
[0.25930767, 0.37516340, 0.48351489] \\
[0.19096334, 0.59283887, 0.85473146] \\
[0.59541250, 0.66392897, 0.70810211] \\
[0.42336683, 0.77392124, 0.89866355] \\
[0.45891803, 0.51344200, 0.56130404] \\
[0.71649763, 0.62609872, 0.47396890] \\
[0.92123446, 0.95420905, 0.95794388]
\end{gathered}
$$

The assigned cluster for each pixel has been calculated, and the new image has been plotted:



4. Exercise

(a)