Assignment 3
Introduction to Data Science

Magnus Lindberg Christensen (fwz302)

March 5, 2024

In the following assignment I will work with the *Bayes Statistics* as well as *K-means clustering*.

Exercise 1

In the following exercise I will give an example of how bayesian statistics can be used in my study degree.

My particular study degree is a MSc in IT & Cognition, and it's heavily concerned with computational cognitive science, language processing, as well as vision and image processing.

An example of using bayesian statistics could be with learning the meaning of words. For instance, what meaning do a particular subject give to words based on the amount of examples seen.

$$p(\theta \mid D) = \frac{p(D \mid \theta)\pi(\theta)}{p(D)} \propto p(D \mid \theta)\pi(\theta) \tag{1}$$

Taking our basis in Bayes' theorem (Figure 1), the posterior, likelihood, and prior would look like the following:

- Posterior / $p(\theta \mid D)$:
  This would be the actual meaning (θ) that a subject would assign to a given word based on the data D
- Likelihood / $p(D \mid \theta)$:
  The likelihood utilises the data, and expresses the probability of the data (e.g. words) looking like it does, based on the meaning the potential meaning of the word. In other words, the likelihood of a word D taking on the meaning θ
- Prior / $p(\theta)$:
  The prior illustrates our existing knowledge. To our example this means the probability of a particular word meaning θ before having seen any new information, data/words D. This prior can be the posterior from previous iterations as well.

The example is inspired by (Xu & Tenenbaum, 2007)

Exercise 2

In the following exercise, I will justify each of the four steps in the ELBO derivation. The derivation consists of four different steps.

1. The first expression states that the probability under the distribution x with the parameters θ is equal to the expected value of the same probability distribution under another distribution $(z|x)$ with parameters ϕ. This is a valid statement as the probability $p_\theta(x)$ can be represented by the sum rule for the continuous case of the two distributions, which is $\int p(x, z)$. Expanding this joint distribution by the product rule this leads to the following $\int p(x|z)p(z)$. By convention this is as well the expectation for $p(x)$. Implementing the logarithmic function, we get the rewritten expression with the expectation $\mathbb{E}$.

2. In the second expression the inner of the expectation are being rewritten. We have our original distribution $p_\theta(x)$, and as we know from the first step, this can be written as $p_\theta(x, z) = p_\theta(x|z)p_\theta(z) = p_\theta(z|x)p_\theta(x)$. It's now possible to refactor the expression to the following $p_\theta(x) = \frac{p_\theta(x,z)}{p_\theta(z|x)}$. Hereby, we have now shown the derivation of the second line.

3. Since this is a work of variational bayes, we want to minimize the KL-divergence. Therefore, we need to incorporate the variational distribution $q(z|x)$. This can be done by multiply the existing expression with a factor of $\frac{q_\phi(z|x)}{q_\phi(z|x)}$, which basically a equivalent to by multiplying the original expression with 1, and therefore doesn't change the distribution. This can then be split accordingly to get the desired expression.

4. We now have two separate fractions inside the equation, and each of these is an independent random variable. It is obvious that the right expression $\frac{q_\phi(z|x)}{p_\theta(z|x)}$ is the KL divergence on the form $D_{KL}(q_\phi(z \mid x) \mid\mid p_\theta(z \mid x))$, which as mentioned earlier is what we want to minimize, while the other is the evidence lower bound. We can seperate each of these expressions by applying the logarithmic product rule which states that $log[X \cdot Y] = log[X] + log[Y]$. Hereafter, we can utilise the rule for adding together two independent random variables under expectations $\mathbb{E}[X+Y] = \mathbb{E}[X]+\mathbb{E}[Y]$. To our example the provides the following and final equation:

$$\mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)} \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

Hereby, we have derived the expression shown in the assignment.

Exercise 3

In the following exercise I will consider the pyMC3 model for Bayesian linear regression and present and correct the errors that it contains. The identified errors can be seen in the list below.

1. *Slope*
   The first correction can be found in the prior a, which acts as the slope. From this particular example we don't have any information on the data, and therefore it shouldn't affect the posterior in large degrees. At the moment mu (mean) and sigma (standard deviation) are large (mu:100, sigma:100). Due to the lack of knowledge, it would be better to reduce the influence of the prior, for instance by assuming it takes on the form as a standard normal distribution with $mu = 0$ and $sigma = 1$. It is obvious that this choice is a result from a cost benefit analysis. Having a larger variance would indicate that we are more open towards new data. Yet, much of our decisions should be based on the likelihood, and though the larger sigma might be informative, it might as well just be diffused, which we can handle by this. Also, having a larger variance, might be inefficient due to the small impact it would have on the data, which could increase the computation time significantly, and we would therefore still want a somewhat impact prior. This leads to the following corrections

   ```
   a = pm.Normal("slope", mu=0, sigma=1)
   ```

2. *Intercept*
   The second prior b contains the same issues as the before mentioned. Therefore, its the same reasoning that holds, which leads to the following corrections

   ```
   b = pm.Normal("intercept", mu=0, sigma=1)
   ```

3. *Standard Deviation*
   As the code emphasizes the standard deviation should be in the form of a HalfNormal, which means that it takes on the same dimensions as the Normal distributions, in our case $mu = 0$ and $sigma = 1$, but everything at $x < 0$ will be 0. Hereby we ensure that only positive values are contained in the distribution. Yet, currently it can take on both positive and negative values. This leads to the following corrections

   ```
   s = pm.HalfNormal("sigma", sigma=1)
   ```

4. *Likelihood*
   In Bayesian Statistics the likelihood is what contains information of the

data, and it is here we produce and model the data based on the distributions of different parameters. In the code the observed data y is present but not used. By not using this, the Monte Carlo Markov Chain will contain a lot diverges and potential faulty results. To accommodate this, the correct data should be provided by utilising the *observed* argument.

```
likelihood = pm.Normal("y", mu=a*x+b, sigma=s, observed=y)
```

Exercise 4

In the following exercise I will cluster a particular set of data using the K Means algorithm.

Description of software used

On a broad level the software used was **Visual Studio Code** with an **Jupyter Notebook** integration. The algorithm was developed from scratch using the general purpose programming language, **Python**, utilising the libraries **numpy** and **pandas**. More specifically how the algorithm works, it is a python class with four functions; *init, distance, fit* and *predict*.

The init function initializes the training data and centroids to None, while specifying the particular use of k.

The fit function takes an argument being the data to train the model on. We then initialise the first iteration centroids to be the first two examples of the data. The algorithm is then run while the converged flag isn't true. During each iteration of the while loop we go through all the points in the data, calculates their distance between the current centroids, and label the centroid they belong to as the closest one. Hereafter, we calculate the new centroids. This is done for each centroid, averaging all its belonging data. This value is then the new centroid which will follow into the next iteration. The algorithm converges when the newly calculated centroids are the same as the previous ones.

The predictions function takes as well some data. Each point in this data is then compared to the converged centroids and labeled as before. This list of labels with the same length as the input data is then returned and can be compared.

Cluster center values

The converged values of the two cluster centers can be seen in Figure 1

```
Centroid 0:
[5.70726496e+00 4.93012821e+01 7.92408120e+02 3.85595940e+03
 3.38821368e+03 1.35652778e+03 2.91737179e+02 1.29989316e+02
 6.86111111e+01 3.81880342e+01 1.87692308e+01 4.13461538e+00
 4.42307692e-01]

Centroid 1:
[2.19924812e+00 1.40018797e+01 1.73727444e+02 1.40094549e+03
 3.18759962e+03 2.62043985e+03 1.00147368e+03 6.31413534e+02
 4.95295113e+02 2.95238722e+02 1.45689850e+02 2.91466165e+01
 2.82330827e+00]
```

Figure 1: The values of the $k = 2$ centroids derived from the K Means Clustering algorithm

Display of the two constructed plots

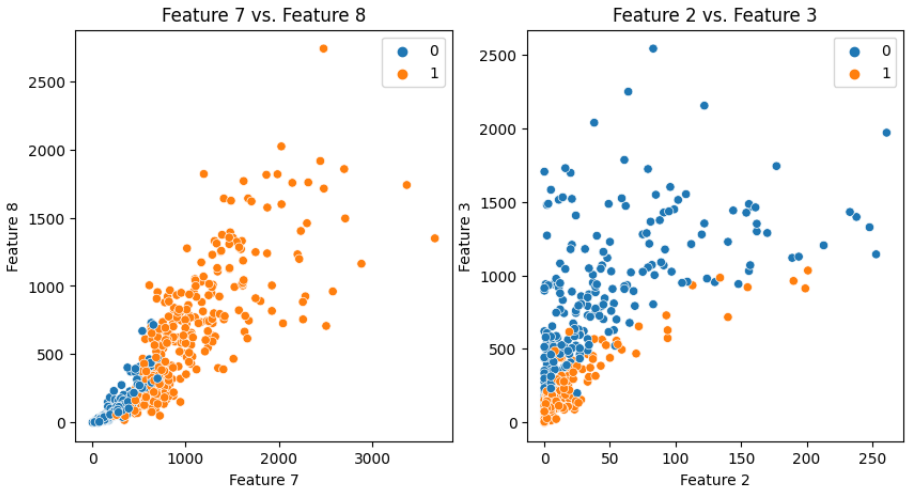

Figure 2: Plot showing different features against each other, depicted with the plot that they belong to.

From the two plots in Figure 2, we can make some observations. It seems that the features have a different range of values. Feature 7, Feature 8 and Feature 3 are obivously more spread than Feature 2. This indicates, that there might be an imbalance in the dataset, and that an approach for normalization should be constructed. Yet, it might also not be a problem.

Another observation is that in the first plot (Feature 7 vs. Feature 8), it appears that most of the data belongs to centroid 0 when the value of Feature 7 is $\approx<$ 850. Hereafter, most of the data belongs to centroid 1. On the other

hand, when Feature 8 has a value $\approx> 900$, all data belongs to centroid 1.

The same observations can be made for the second plot. Here we find that when Feature 3 is $\approx> 1000$ all data belongs to centroid 0.

Generally for both of the plots the density of the data is the highest within the bottom left parts of the plots. This might indicate that the ones that aren't can be potential outliers (e.g. the ones in the upper right corner of the plots), as few data points has high values in both compared features.

Exercise 5

In the following exercise I will run the K Means clustering algorithm on the image *dog.jpg* and use the *elbow method* to decide on the optimal number of clusters. Hereafter, I will compress det image and show the two side by side.

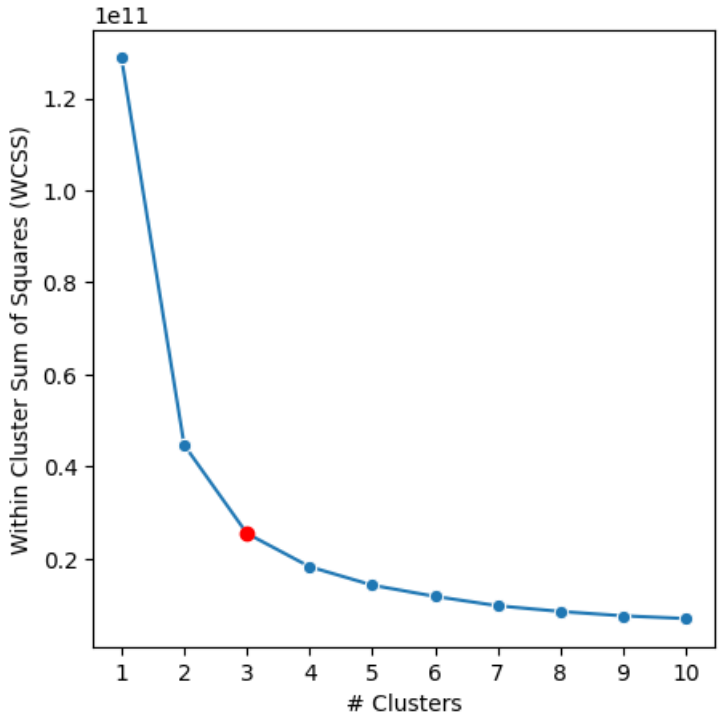

Figure 3: The results from the elbow method. $1 - 10$ clusters was tested

In Figure 3 you can see the results from conducting the elbow method on the data. Looking at the graph we can observe that there is a decrease in the within cluster sum of squares with an increase in clusters. This means that the data within a cluster becomes more compact the more clusters there are. Eventually, this pattern flattens out.

The optimal number of clusters as well as its corresponding inertia value can be seen in Figure 4

```
Optimal # Clusters: 3
Inertia value: 25430780968.224243
```

Figure 4: The optimal # of clusters (k) and its corresponding inertia value

Comments on the method

Their might be several reasons as to why the elbow method might not be a good method for estimating k. Yet, I will only focus on the following one.

(1) The method is highly subjective, as it is being decided based on a graphical representation. Even though this might be clear and easy in some cases, you can imagine situations where the optimal k wouldn't be as easy to spot, and therefore induce errors into you model from a subjective intuition of the representation. Therefore, the plot should be clearly outlined with enough space as well, to make sure that one can easily notice the "elbow" in the pattern.

K Means algorithm and image segmentation

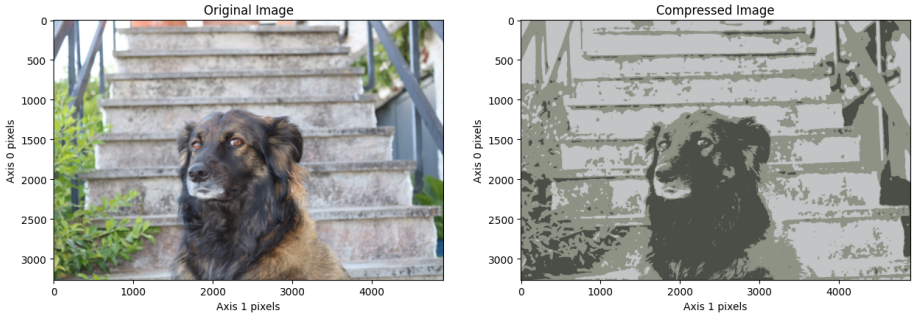

Figure 5: The original image and the compressed image as a result of running K Means algorithm with optimal number of clusters

In Figure 5 we can see the image *dog.jpg* in its original form and compressed form side by side.

The compressed image is derived by running the K Means algorithm with the optimal number of clusters, $k = 3$. In this exercise the meaning of the k value is the amount of colors being shown in the compressed image. In the example of $k = 3$, we have 3 clusters, which means the image are being divided into three different color values. This is the reason that the compressed image only is composed from three different colors, yet these can appear in several areas of the image. Increasing k would lead to more colors, and hereby also more distinct regions of the image, which again can appear several places.

Assignment Statements

Gemini version 1.5 has been used to cross validate my understanding of the notation throughout the second and third exercises

References

Xu, F., & Tenenbaum, J. B. (2007). Word learning as bayesian inference. *Psychological review*, *114*(2), 245.