Assignment 3 - Jesús Rabanal Álvarez

March 4, 2024

[1]:
```
from sklearn.cluster import KMeans
import numpy as np
from matplotlib.image import imread
import matplotlib.pyplot as plt
import seaborn as sns
```

BAYESIAN EXERCISES

EXERCISE 1:
Provide a small and concise example of how bayesian statistics can be applied in your study degree. You should provide references to your answer if you don't come up with an example on your own.

Bayesian Statistics played a profound role in the development of Natural Language Processing in the 90s.(Cohen, 2022)
A great example of the use of Bayesian Statistics in NLP is sentiment analysis, for example, as used in **movie reviews or tweets**.

$$P\left(w_k|c_i\right) = \frac{\text{count}\left(w_k, c_i\right) + 1}{\sum_{w \in V} \text{count}\left(w, c_i\right) + |V|}$$

- **The prior:** $P(c_i)$ representing the likelihood of a document being in class c_i, without considering any specific words.
- **The likelihood function:** where $P\left(w_k|c_i\right)$ represents the probability of observing word w_k given that the documents belongs to the class c_i.
- Then the numerator $\text{count}\left(w_k, c_i\right)+1$ denotes the number of occurrences of word w_k in documents belonging to the class c_i, with laplace smoothing.
- Finally the denominator $\sum_{w \in V} \text{count}\left(w, c_i\right) + |V|$ is the laplace smoothed sum of the counts of all the words in the vocabulary that appear in the documents of the class c_i, plus the size of the vocabulary multiplied by the Laplace smoothing parameter.
- **The posterior:** $P(w_1, w_2, ..., w_n) = \sum_i P(w_1, w_2, ..., w_n|c_i) \times P(c_i)$ shows us the probability of the document belonging to class c_i given the words $w_1, w_2, ..., w_n$ present in the document.

EXERCISE 2:
Consider the following derivation of the ELBO, a quantity used in variational Bayes inference. For each of the 4 lines in the derivation, explain its justification (hint: remember the "three power tools of statistics").

1. $\log p_\theta(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right]$ This line states that the log likelihood of observing data x is equal to the expectation of the log likelihood under the variational distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$.

2. $= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right]$ This lines uses the definition of joint and conditional probability to express $p(x)$ in terms of joint and conditional probabilities.

3. $= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right]$ The third line applies the property of logarithms where the difference of logarithms is the logarithm of the quotient. This is done in order to obtain the last step, or line 4.

4. $= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]\right]}_{=\mathcal{L}_{\theta,\phi}(\mathbf{x})} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right]}_{=D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))}$ This last line, following the third line splits the expectation in two terms. The first represents ELBO and the second one represents Kullback-Leibler divergence between the variational distribution and the true posterior.

EXERCISE 3

Consider the following pyMC3 model for Bayesian linear regression. There are 4 lines with errors. Identify them, correct the error, and explain why it was an error (i.e., not just what was wrong, but WHY it was wrong).

[ ]:
```
#Define the model
with pm.Model() as model:

    x = pm.Data('x', x_data)
    y = pm.Data('y_obs', y_data)

    a = pm.Normal('slope', mu=0, sigma=1)  #set mu to 0 and sigma to 1
    b = pm.Normal('intercept', mu=0, sigma=1) #set mu to 0 and sigma to 1
    s = pm.HalfNormal('sigma', sigma=0.001) #change function from pm.Normal()␣
↪to pm.HalfNormal()

    mu = pm.Deterministic('mu', a*x+b) #add definition of mu as a deterministic␣
↪variable to the model

    likelihood = pm.Normal('y', mu=mu, sigma=s, observed=y) #add the observed␣
↪data to the model

    step = pm.NUTS()

    trace = pm.sample(1000, tune= 1000, init=None, step=step, cores=2)
```

1. Setting a mu so high can make the model be overtly biased towards higher values of the slope and intercept. (2 mistakes)

2. A standard deviation that large allows for a vast amount of plausible values, while it might be indicative of uncertainty, it will definitely make the prior less informative and therefore resulting in not regularizing the model properly. (2 mistakes)

3. Thirdly, the function used to determinate s can't be a normal since the standard deviation used in the likelihood can't be negative, it therefore must be a HalfNormal. (1 mistake)

4. The observed data is presently doing nothing, it must be included in the definition for the likelihood so the function has data to propagate from. Also (1 mistake)

5. The mu used in the likelihood function should be defined using the pm.Deterministic() to properly represent the deterministic relationship between the data and the parameters. (1 mistake)

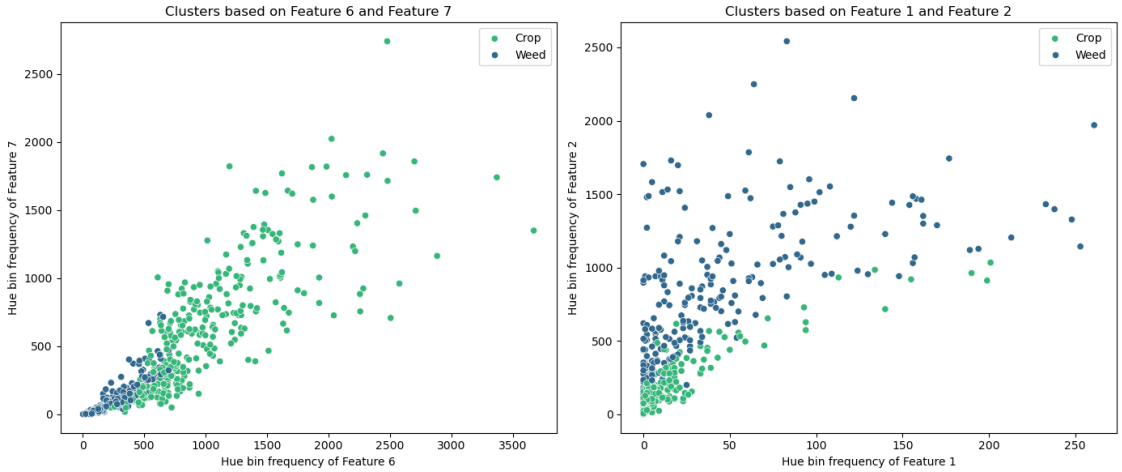
CLUSTERING EXERCISES

EXERCISE 4

- Perform 2-means clustering of the input data in IDSWeedCropTrain.csv. For the submission, initialize the cluster centers with the first two data points in IDSWeedCropTrain.csv (that is not a recommended initialization technique, but makes it easier to correct the assignment). [Visualization] Assuming we can get a rough idea about our data clusters by taking a look at 2 of the 13 features, make two scatter plots of your test split clusters by using:
- Feature 6 and Feature 7 (e.g. you can get feature 6 by using the 6th column XTest[:, 6]).
- Feature 1 and Feature 2
- Compare the difference in the scatter plots and summarize your observations on using the different selected features. (Hints: you can use sns.scatterplot() for visualization)

[3]:
```
dataTrain = np.loadtxt('IDSWeedCropTrain.csv', delimiter=',')
dataTest = np.loadtxt('IDSWeedCropTest.csv', delimiter=',')

XTrain = dataTrain[:, :-1]
YTrain = dataTrain[:, -1]
XTest = dataTest[:, :-1]
YTest = dataTest[:, -1]

#get the first two rows of the training data
startingPoint = np.vstack((XTrain[0,], XTrain[1,]))

#create instance of KMeans with 2 clusters
kmeans = KMeans(n_clusters=2, n_init=1, init=startingPoint, algorithm='lloyd')
kmeans.fit(XTrain)

#predict the labels of the test data
test_labels = kmeans.predict(XTest)

#plot the clusters
plt.figure(figsize=(14, 6))

#plot the clusters based on Feature 6 and Feature 7
plt.subplot(1, 2, 1)
sns.scatterplot(x=XTest[:, 6], y=XTest[:, 7], hue=test_labels,␣
 ↪palette='viridis')
plt.title('Clusters based on Feature 6 and Feature 7')
plt.xlabel('Hue bin frequency of Feature 6')
plt.ylabel('Hue bin frequency of Feature 7')
```

```
plt.legend(['Crop', 'Weed'], loc='upper right')

#plot the clusters based on Feature 1 and Feature 2
plt.subplot(1, 2, 2)
sns.scatterplot(x=XTest[:, 1], y=XTest[:, 2], hue=test_labels,␣
 ↪palette='viridis')
plt.title('Clusters based on Feature 1 and Feature 2')
plt.xlabel('Hue bin frequency of Feature 1')
plt.ylabel('Hue bin frequency of Feature 2')
plt.legend(['Crop', 'Weed'], loc='upper right')

plt.tight_layout()
plt.show()
```



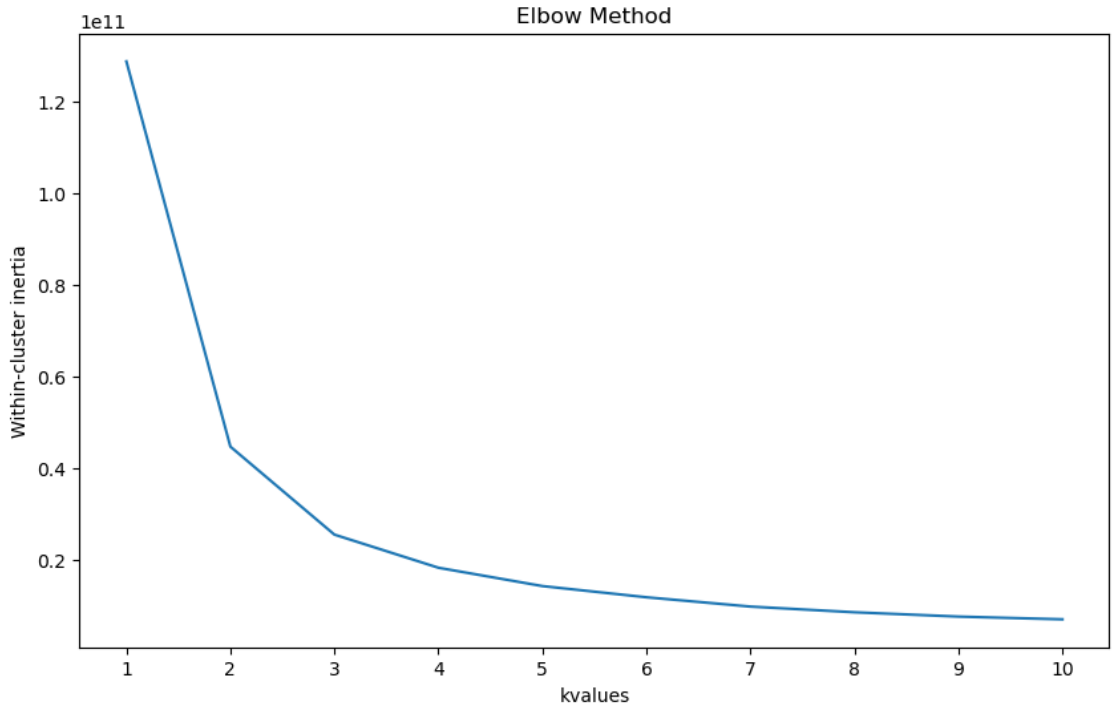
In the plots above, we applied Lloyd's algorithm to cluster information from provided CSV files. Initially, we initialized two cluster centers using the first two data points from the training dataset. Then, the algorithm was executed with these two seeds. Providing seed centroids to K-means facilitates calculations, adds more control over the results, and aids in better representing the data. This is necessary because K-means can be highly sensitive to the initial placement of centroids. However, it's crucial to note that the quality of the seeds must be good, as poor initialization can result in suboptimal clustering outcomes. In the first scatterplot (features 6 and 7), we can see how the lower frequencies of both features were assigned to cluster Weed, and the higher frequencies were assigned to the Crop cluster. In the second scatterplot (features 1 and 2), we can see a different tendency here, that higher values from feature 2 were assigned generally to cluster Weed, and lower values of the same feature were assigned to the Crop cluster in this case.

EXERCISE 5

1. Open and read image into np.array. Use the elbow method to choose the number of k clusters (kvaries between 1 and 10).
2. Now, with the best observed k value implement k-means clustering on the image and display original and compressed image.

[4]:
```
#read the image and reshape it
dog_img = imread('dog.jpg')
dog_2d = dog_img.reshape(dog_img.shape[0] * dog_img.shape[1], dog_img.shape[2])

#initialize the starting points
startingPoint_list = np.unique(dog_2d, axis = 0)

#initialize the list to store the inertia values
kvalues = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
kinertia = []

#loop through the kvalues and fit the KMeans model
for i, k in enumerate(kvalues):
    i+=1
    kmeans = KMeans(n_clusters=k, algorithm='lloyd', n_init=1,␣
 ↪init=startingPoint_list[:i]).fit(dog_2d)
    inertia = kmeans.inertia_
    kinertia.append(inertia)
```

[5]:
```
#plot the elbow method
plt.figure(figsize=(10, 6))
plt.plot(kvalues, kinertia, '-')
plt.title("Elbow Method")
plt.xlabel("kvalues")
plt.xticks(kvalues)
plt.ylabel("Within-cluster inertia")
plt.show()
print(f'Inertia Value for k = 3: {kinertia[2]:.2f}')
```




```
Inertia Value for k = 3: 25445823965.33
```

In graph above we calculated the optimal kvalue for kmeans, where we plot the inertia of each cluster agaisnt the number of clusters (kvalues). Using the elbow method then we identify the elbow then at kvalue 3. In this case using the elbow is okay because we are working with very few kvalues, but with bigger amounts of kvalues it can be difficult to indentify the elbow in the plot.

[6]:
```
#Create a KMeans instance with 3 clusters
kmeans = KMeans(n_clusters=3, algorithm='lloyd', n_init=1,␣
 ↪init=startingPoint_list[:3]).fit(dog_2d)

#Predict the labels of the dog_2d data
dog_labels = kmeans.predict(dog_2d)
dog_clustered = kmeans.cluster_centers_

#Create a compressed image
dog_compressed = np.empty_like(dog_2d)

#Replace each pixel with its corresponding cluster center
for i in range(len(dog_2d)):
    dog_compressed[i] = dog_clustered[dog_labels[i]]

#Reshape the compressed image
dog_size = dog_img.shape
dog_compressed = dog_compressed.reshape(dog_size)
```

```
#Plot the original and the compressed image
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.imshow(dog_img)
plt.title('Unclustered Dog')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(dog_compressed)
plt.title('Clustered Dog')
plt.axis('off')
plt.show()
```

In this exercise picking k = 3 means choosing how many shades of grey we are using to segment the image. In this case 3 shades of grey as it can be seen in the clustered dog.

References:

Cohen, S. (2022). Bayesian analysis in natural language processing. Springer Nature.
GitHub Copilot v1.171.0 was used as assistance while developing the code for this assignment.