HW #4

Uziel Rivera-Lopez

03/14/2023

Question 1

(a) Feature selection involves us selecting subets of the most relevant features from our original dataset. This gives us easier to interpret features and imapcts, reduction in computation time, and avoiding overfitting the data. Yet, with only selecting a subset of the features, we will lose information that could be helpful, and, because of it's simplicity, we cannot capture complex relationships with dataset. Feature extraction involves creating a new set of features from the original set. It gives us capture complex relationships, and a better representation of our dataset. However, itcan lead hard interpretations, computing is increased, and we are introduce risk of giving our model irrelevant information. If we want the model more easier and reduce a high dimensional dataset, while also reducing computation time, we should use feature selection. If we want to capture complex relationships, and have a better representation of our dataset, we should use feature extraction.

(b) Both Forward and Backward selection use Feature Selection, but in their own ways. Forward selection is adding features that improve a model and continues iteratively, until we hit a limit that doesn't signficantly improve the model. The working principle goes like this:

   (a) You have to start with an empty model, no features.
   (b) Evaluate the performance of the model with each feature.
   (c) Add the feature that gives the best performance.
   (d) Repeat b and c until we have the best performing model, and no further imrpovement is seen.

   This gives us the least chance of removing/leaving out any important features, computing is more efficient if we have a high number of features, and it's also a good starting point as to what makes our modeel perform well. However, because we are evaluating the performance of the model with each feature, we are creating a specific order that the features must go in to have the best performance. The other issue is that when we stop adding based on the performance it can be subjective and suboptimal.

Backward selection is the opposite we are removing features each time, and determing which contributes the least in performance. The working principle goes like this:

(a) You start with all features.

(b) Evaluate the performance of the model with each removed feature.

(c) Remove the feature that gives the least negative impact.

(d) Repeat b and c until we have the best performing model, and no further imrpovement is seen.

This gives us the least possible chance of missing important features, and finding any repeative features. Yet, if we do this early on, we remove features that may not seem important at first, but are important later. Also if our dataset is high in dimension, it can be computationally expensive. In terms of which one to use, if we want to deal witha high dimension dataset, and not mind having redundant features, we should go with Forward. But if we don't want to repeatitive features, and only want to work with the important features but will have to deal with the computing expense, we should go with Backward.

(c) The calculation would be $\binom{20}{10}$ which is 184,756

Question 2

(a) PCA is a unsupervised method that reduces the dimensions of a complex dataset, while retaining important patterns and trends. It involves standardizing the data, calculating the covariance matrix, determing the eigenvectors and eigenvalues, and then projecting the data onto the new space.

(b) The principle components are the eigenvectors that give us the data in a new space, makes the data easier to interpret, and gives us the important information of the data. Usually we don't have an optimal number of PC to use, but when determining the projections we base it of the first two PC, and see if we can capture the most variance. Another thing is that we can use the elbow method to determine the number of PC to use, since PCs have a limit as to how much the contributes to the variance. Using PC, we reduce the computing cost, improve interpretations, and avoid overfitting. Yet, we are reducing the dimensions so we lose information, and have the possibility of suboptimal selection.

(c) Judging from Z_1 graphs, the first option shows much better separation and a clear visual of the data, the projection also seems to be more optimal than the rest. The other options are way too condensed. They are not as optimal as the first option.

Question 3

$$\mu = \begin{bmatrix}2\\3\end{bmatrix} \Sigma = \begin{bmatrix}1 & 0.5\\0.5 & 2\end{bmatrix} W = \begin{bmatrix}1\\2\end{bmatrix}$$

$$W^T \cdot \Sigma \cdot W = \begin{bmatrix}1 & 2\end{bmatrix} \cdot \begin{bmatrix}1 & 0.5\\0.5 & 2\end{bmatrix} \cdot \begin{bmatrix}1\\2\end{bmatrix} = [11] = \Sigma_{W^T}$$

$$W^T \cdot \mu = \begin{bmatrix}1 & 2\end{bmatrix} \cdot \begin{bmatrix}2\\3\end{bmatrix} = [8] = \mu_{W^T}$$

So the new distribution would be $N([8], [11])$

Question 4

(a) If our covariance matrix is full rank it means there are non-zero eigenvalues. It shows that our eigenvectors are linear independent from each other. We also don't lose information when we are extracting the features, which in turn also gives us a unqiue solution to our problem. This gives us a more clear interpretation of our data, and a better representation.

(b) We meglect the later eigenvectors because they don't contribute much to the variance. We would consider the less eigenvectors if we want them for data visualization, or to see any patterns that are not captured by the first few eigenvectors. But we should not consider if they cause a much higher noise, limited to resources, and they also don't contribute much to the variance.

(c) Taking the sum of the first two eigenvalues and dividing by the sum of all the eigenvalues, we get 54%. It tells us that only with the first two, we can capture 54% of the variance. With the whole set of values given to us we capture only 78% of the variance. Because we want to ideal capture about 80-90% of the variance, we would need to use the first the eigenvalues given to us plus a 1 or 2 more, since the values are decreasing I can only assume it will be getting much more smaller.

Question 5

$$Cov(x) = \begin{bmatrix}0.15 & 1 & 2 & 0.08\\0.23 & 0.1 & 0.43 & 0.32\\0.19 & 0.6 & 0.45 & 0.07\\0.3 & 0.4 & 0.5 & 0.07\end{bmatrix} V = \begin{bmatrix}0.41 & -0.14 & -0.13 & 0.27\\0.08 & 0.2 & -0.03 & 0.19\\0.03 & -0.07 & -0.21 & 0.11\\0 & 0 & 0 & -0.06\end{bmatrix}$$

$$V \cdot V^T = \begin{bmatrix}0.41 & -0.14 & -0.13 & 0.27\\0.08 & 0.2 & -0.03 & 0.19\\0.03 & -0.07 & -0.21 & 0.11\\0 & 0 & 0 & -0.06\end{bmatrix} \cdot \begin{bmatrix}0.41 & 0.08 & 0.03 & 0\\-0.14 & 0.2 & -0.07 & 0\\-0.13 & -0.03 & -0.21 & 0\\0.27 & 0.19 & 0.11 & -0.06\end{bmatrix} =$$

$$\begin{bmatrix} 0.2775 & 0.06 & 0.791 - 0.0162 & \\ 0.06 & 0.0834 & 0.0156 & -0.0114 \\ 0.0791 & 0.0156 & 0.062 & -0.0066 \\ -0.0162 & -0.0114 & -0.0066 & 0.0036 \end{bmatrix}$$

To find the noise we would take the sum of the diagonal of the matrix. $\Psi = diag(\Sigma - V \cdot V^T)$

$$\Psi = \Sigma - diag(V \cdot V^T) = diag(\begin{bmatrix} -0.1275 & 0.94 & 1.9209 & 0.0962 \\ 0.17 & 0.0166 & 0.4144 & 0.3314 \\ 0.1109 & 0.5844 & 0.388 & 0.0766 \\ 0.3162 & 0.4114 & 0.5066 & 0.0664 \end{bmatrix}) = \begin{bmatrix} 0.1275 & 0.94 & 1.9209 & 0.0962 \\ 0.17 & 0.0166 & 0.4144 & 0.3314 \\ 0.1109 & 0.5844 & 0.388 & 0.0766 \\ 0.3162 & 0.4114 & 0.5066 & 0.0664 \end{bmatrix}$$

Question 6

(a) Centers of the clusters are:
c1 = 30, c2 = 45, c3 = 4
Data points are:
a = 21, b = 35, c = 10, d = 28, e = 41
Now we determine the distance for each cluster and assign them to the closest cluster. Note we are doing data - center of cluster for each data point, and assigning the cluster in the cluster section. Updating the vectors based on how many data points are in each cluster, we get:

Data	c1	c2	c3	Cluster
a	9	24	17	c1
b	5	10	31	c1
c	20	35	6	c3
d	2	17	24	c1
e	11	4	37	c2

c1 = (21 + 35 + 28)/3 = 28
c2 = 41
c3 = 10
This is iteration one, with two it's the same concept, but with our new centers. Now we update the reference vectors again, and we get:

Data	c1	c2	c3	Cluster
a	7	20	11	c1
b	7	6	25	c2
c	18	31	0	c3
d	0	13	18	c1
e	13	0	31	c2

c1 = (21 + 28)/2 = 24.5
c2 = (35+41)/2 = 38
c3 = 10

(b) One could be that we have a preferance towards a local cetner within the cluster, which if we don't place them right initially we can get a suboptimal solution, which leads on to the next point. If we have outliers in terms of cluster shape, when we are updating the reference vectors, we can possibly get located far from the center. This leads to the initial centroids being influenced by the outliers, and we get inaccurate results. To fix this, we can randomly initialize the center of the clusters, and allow for resets to then get an average of the results. We can also use K-means++ algorithm, mentioned online, where we can get the initial centroid, place it randomly, and then place the rest based a maxium square distance from the previous centroid.