1a: Answer the following True/False questions, providing one sentence of explanation for each one

1a(i): We compute principal components by finding eigenvectors of the dataset's feature matrix X.

1a(ii): To select the number of components K to use, we can find the value of K that minimizes reconstruction error on the training set. This choice will help us manage accuracy and model complexity.

1a(iii): If we already have fit a PCA model with K=10 components, fitting a PCA model with K=9 components for the same dataset is easy.

Answer

1a(i): False. We center the data as $\tilde{X}$. Then, we compute SVD or the eigenvectors and eigenvalues of $\tilde{X}\ \tilde{X}^T$.
1a(ii): False. We select K components such that it preserves major information(such as 90% or 95% in L_2-norm) or by memory constaints or optimize the evaluation metric.
1a(iii):True. In this case, one can find just exclude the component with the smallest singular value, which can be easily calculated given the data.

1b: Suppose we had a dataset for a medical application, with many measurements describing the patient's height, weight, income, daily food consumption, and many other attributes.

1b(i): Would the PCA components learned be the same if you used feet or centimeters to measure height? Why or why not?

1b(ii): Before applying PCA to this dataset, what (if any) preprocessing would you recommend and why?

Answer

1b(i): No. PCA is variant when rescaling. Unless we rescale all data by the same ratio, it usually leads to a different result.
1b(ii): In this case, we should normalize the data before applying the PCA analysis.

1C:

Answer:

No. The problem is we should transform X' into the same space of X. This means we should use the same values and pre-process that are used in X to transform X'.
In particular, we should use the means from X, not from X':
$m = \frac{1}{N} \quad x_n$

2a: Answer the following True/False questions, providing one sentence of explanation for each one

2a(i): We always get the same clustering of dataset X when applying K-means with K=1, no matter how we initialize the cluster centroids μ

2a(ii): We always get the same clustering of dataset X when applying K-means with K=2, no matter how we initialize the cluster centroids μ

2a(iii): The only way to find the cluster centroids μ that minimize the K-means cost function (minimize sum of distances to nearest centroid) is to apply the K-means coordinate descent algorithm, alternatively updating assignments and cluster centroid locations.

2a(iv): The K-means cost function requires computing the Euclidean distance from each example xn to its nearest cluster centroid μk. Because the Euclidean distance requires a square root operation (e.g. np.sqrt or np.pow(___, 0.5)), no implementation of the K-means coordinate descent algorithm can be correct unless a "sqrt" operation is performed when computing distances between examples and clusters.

Answer:

2a(i): True. There is only the whole set.
2a(ii): False. This will depends on the initialization because the existence of local min of cost function.
2a(iii):False. One example is that we can use probalistic methods like GMM and do MLE.
2a(iv): False. Since the Euclidean Distance d is non-negative, minimizing d is the same as minimizing d^2. So the "sqrt" is not necessary.

2b:

One can calculate the distance d_n of a point x_n from its nearest cluster point(centroid). Then we pick the x_k such that it has the max of d_n (the point that is the most away from any centroid) and take it as our newest initial cluster point. Now, we can update and find optimal cluster points as normal.

2c:

init='k-means++'
n_init=20

2d:

2d(i): We first compute the distance of x_n from K centroids(O(F) each centroid for K times, O(FK) total). Then we compare the distace (O(K)). So the total is O(FK)+O(K)= O(FK) for each x_n. Therefore, we need to compute $\mathbf{O(FKN)}$ for all x_n.

2d(ii): We put x_n into its cluster and accumulate the sum of all x_n in the clusters. Therefore for each x_n, there is O(F) for addition, which makes O(FN) operations. To get the centroids, we need to count number of points O(N) in total and divide sum by the number of points in each cluster O(K). Since K≤N, O(FN)+O(N)+O(K)=$\mathbf{O(FN)}$.

In [ ]: