Machine Learning 2

Midterm Review

Tran Hai Nam - 11219279 - DSEB

1 Dimensional Reduction

Problem 1

What are the steps of PCA? Give your reasons for each step if possible.

Solution.

Steps of PCA involves:

1. **Standardize the data**: If the features in the dataset have different scales, it is important to standardize them (subtract the mean and divide by the standard deviation) to ensure that each feature contributes equally to the analysis.
2. **Compute the covariance matrix**: The covariance matrix measures the relationships between the different features in the dataset. It provides information about how the features vary together.
3. **Calculate the eigenvectors and eigenvalues**: The eigenvectors and eigenvalues of the covariance matrix represent the principal components. The eigenvectors are the directions of maximum variance, while the corresponding eigenvalues indicate the amount of variance explained by each principal component.
4. **Select the principal components**: To reduce the dimensionality, you can select a subset of the principal components based on their corresponding eigenvalues
5. **Transform the data:** Multiply the standardized data by the selected principal components to obtain the transformed dataset with reduced dimensionality. Each data point is projected onto the new principal component axes.

Problem 2

How to choose the number of dimensions to reduce when building PCA?

Solution. There are ways to choose number of dimensions when building PCA:

1. **Variance Explained**: Analyze the eigenvalues (variance of each principal component). Aim for a cumulative percentage of explained variance that meets your needs. Often, 80-90
2. **Elbow Method**: Plot the scree plot (eigenvalues vs. component number). Look for the "elbow" where the explained variance starts decreasing rapidly. Components after the elbow contribute less significant variance.

Problem 3

What are the limitatations of PCA?

Solution.

1. **Assumes Linear Relationships**: PCA works best when the data has a linear relationship between variables. If the relationships are non-linear, PCA might not capture the underlying structure effectively.
2. **Loss of Information**: PCA discards information by focusing on the most significant components. There's a trade-off between dimensionality reduction and the information loss that comes with it.
3. **Outlier Sensitivity**: Outliers can significantly impact the principal components identified by PCA. Extreme values can distort the principal components.
4. **Limited Interpretability**: The resulting principal components are linear combinations of original features. They might be less interpretable than the original data points.

Problem 4

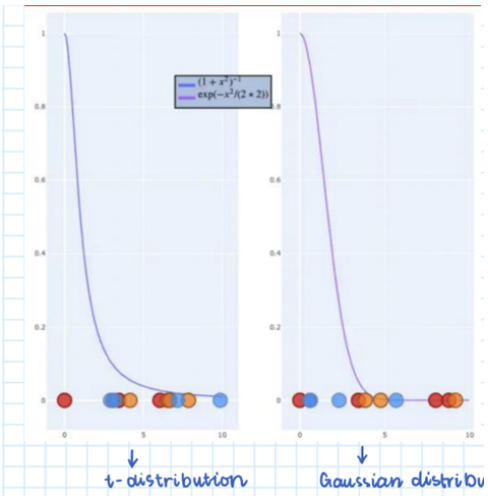
Why do we have to use t-distribution in t-SNE?

Solution.

In t-SNE (t-distributed Stochastic Neighbor Embedding), we use the t-distribution instead of the Gaussian distribution for a specific reason:

Crowding Problem Reduction:

. In low-dimensional space, t-SNE aims to maintain relative similarities between data points. The t-distribution (also known as the Cauchy distribution) has a heavier tail than the Gaussian distribution. This compensates for the original imbalance and allows t-SNE to map large distances more effectively, preventing points from appearing too close together.



Problem 5

Reconstruct PCA.

Solution.

Given dataset $X = \{x_1, x_2, ..., x_n\}$ with mean $\mu = 0$.

We assume there exist a low-dimension compressed representation of a data point x_n given by:

$$z_n = B^T x_n$$

where we compressed x_n of D dimensions to z_n to M dimensions (M ¡ D), and the projection matrix is:

$$B = [b_1, b_2, .., b_M] \in^{D \times M}$$

Note that z also has zero mean: $\mathbf{E}_z[z] = E_x[B^T x] = B^T \mathbf{E}_x[x] = 0$

We need to find a matrix B that retains as much information as possible when compressing data by projecting it onto the subspace spanned by the columns $b_1, b_2, .., b_M$ of B. Retaining most information after data compression is equivalent to capturing the largest amount of variance in the low-dimensional code.

We maximize the variance of the low-dimensional code using a sequential approach. First, we consider the case when X is projected onto a single vector $b \in^D$ (basically, this is when the projection matrix B is just a vector and we want to keep only one most important feature). Our aim is to maximize the variance of the projected data, i.e:

$$\text{maximize } V = \frac{1}{N} \sum_{n=1}^{N} z_n^2$$
$$\text{where } z_n = b^T x_n$$

Substituting this into the expression of V:

$$\begin{aligned} V &= \frac{1}{N} \sum_{n=1}^{N} (b^T x_n)^2 \\ &= \frac{1}{N} \sum_{n=1}^{N} b^T x_n x_n^T b \\ &= b^T (\frac{1}{N} \sum_{n=1}^{N} x_n x_n^T) b \\ &= b^T S b \end{aligned}$$

where S is the covariance matrix of the original dataset.

Here, we observe that increasing the magnitude of b will increase V, thus making this optimization problem impossible to solve. Therefore, we restrict all solutions to $\|b\|^2 = 1$ which results in a constrained optimization problem:

$$\text{find argmax}_b \; b^T S b$$
$$\text{subject to} \|b\|^2 = 1.$$

The Lagrangian function for the problem is:

$$L(B, \lambda) = b^T S b + \lambda(1 - b^T b)$$

Note that b is a single vector then λ is just a number (not a vector) because we only have one condition. Take the partial derivative of L with respect to b and λ to 0 :

$$\frac{\partial L}{\partial b} = 2b^T S + 2\lambda b^T = 0 \iff S.b = \lambda b$$
$$\frac{\partial L}{\partial b} = 1 - b^T b \iff b^T b = 1$$

At this point, we can see that b and λ are a part of eigenvector and eigenvalue S. Then we can write the variance V as:

$$V = b^T Sb = b^T \lambda b = \lambda$$

To maximize the variance of the low-dimensional code, we choose the basis vector associated with the largest eigenvalue principal component of the data covariance matrix. This eigenvector is called the first principal component.

Therefore, if we want to find M axes to project data, we can choose M eigenvectors $b_1, b_2, ..., b_M$ of the data variance matrix that associates with M largest eigenvalues, each eigenvector corresponding to a new axis of projection. The projection matrix B is formed by those eigenvectors, i.e: $B = [b_1, b_2, ..., b_M]$.

Finally, the projection of dataset $X \in^{D \times N}$ onto $B \in^{D \times M}$ given by $\widetilde{X} = B^T X$ will result, in new representation of data, i. e $\widetilde{X} \in^{M \times N}$ (M features and N samples).

2 Clustering

Problem 1

What are steps in K-means?

Solution.

Steps in Kmeans:

1. **Initialize centroids**: Select k random data points as initial centroids, which represent the center of each cluster.
2. **Assign data points to clusters**: Calculate the distance between each data point and all centroids. Assign each data point to the cluster with the nearest centroid.
3. **Recompute centroids**: Calculate the average of all data points belonging to each cluster. These new averages become the new centroids.
4. **Repeat steps 2 and 3** : Continue assigning data points to clusters based on the new centroids and recomputing the centroids until there are no significant changes in the assignments (convergence).

Problem 2

Given pairs of points x,y with coordinates as follows:
$(0.5; 0.5), (1; 0.5), (1; 1.5), (1.5; 1), (2.6; 2), (3; 2), (2.4; 2.5), (2.5; 3)$

- Apply k-means algorithm with centroid number equal to 2.
- Initialize the centroid center with random coordinates, run the k-means algorithm for 3 iterations and return the coordinates of 2 centroids.

Solution.

Let $X = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8]^T = \begin{bmatrix} 0.5 & 0.5 \\ 1 & 0.5 \\ 1 & 1.5 \\ 1.5 & 1 \\ 2.6 & 2 \\ 3 & 2 \\ 2.4 & 2.5 \\ 2.5 & 3 \end{bmatrix}$

Random intialize centroids for 2 clusters :

- $c_1(0, 0)$
- $c_2(2, 2)$

Iteration 1:

Distance from each point to centroids:

x	$d(x, c_1)$	$d(x, c_2)$	cluster
x_1	0.71	2.12	c_1
x_2	1.12	1.8	c_1
x_3	1.8	1.12	c_2
x_4	1.8	1.12	c_2
x_5	3.28	0.6	c_2
x_6	3.61	1	c_2
x_7	3.47	0.64	c_2
x_8	3.91	1.12	c_2

We see that after first iteration, cluster 1 include to point x_1 and x_2, while the remaining belong to cluster 2.

Recalculate the coordinates of the centers for new groups based on the coordinates of the objects in the group:

$$c_1 = (\frac{0.5+1}{2}, \frac{0.5+0.5}{2}) = (0.75, 0.5)$$
$$c_2 = (\frac{1+1.5+2.6+3+2.4+2.5}{6}, \frac{1+1+2+2+2.5+3}{6}) = (\frac{8}{3}, 2)$$

Iteration 2:

Distance from each point to centroids:

x	$d(x, c_1)$	$d(x, c_2)$	cluster
x_1	0.25	2.24	c_1
x_2	0.25	1.9	c_1
x_3	1.03	1.27	c_1
x_4	0.9	1.2	c_1
x_5	2.38	0.43	c_2
x_6	2.7	0.83	c_2
x_7	2.59	0.55	c_2
x_8	3.05	1.05	c_2

Recalculate the center for the new group:

$$c_1 = (\frac{0.5+1+1+1.5}{4}, \frac{0.5+0.5+1+1}{4}) = (1, \frac{7}{8})$$
$$c_2 = (\frac{2.6+3+2.4+2.5}{4}, \frac{2+2+2.5+3}{4}) = (\frac{21}{8}, \frac{19}{8})$$

Iteration 3:

Distance from each point to centroids:

x	$d(x, c_1)$	$d(x, c_2)$	cluster
x_1	0.25	2.24	c_1
x_2	0.25	1.9	c_1
x_3	1.03	1.27	c_1
x_4	0.9	1.2	c_1
x_5	2.38	0.43	c_2
x_6	2.7	0.83	c_2
x_7	2.59	0.55	c_2
x_8	3.05	1.05	c_2

Objects in clusters do not change with last iteration, so centroids do not change.

Final centroids coordinates : $c_1 = (1, \frac{7}{8}), c_2 = (\frac{21}{8}, \frac{19}{8})$

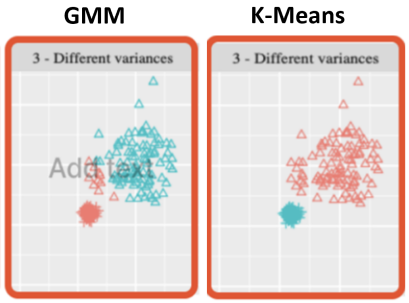
Problem 3

Draw 3 different cases where the data set when clustering does not work/is not good using k-means. State the reason.Suggest a suitable algorithm for that data set and explain why it is suitable

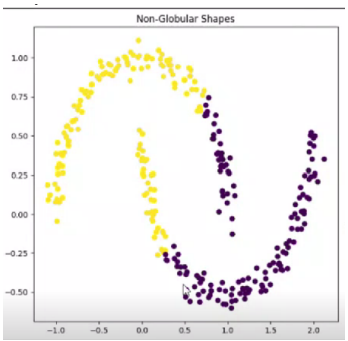
Solution.

3 different cases where the data set when clustering does not work/is not good using k-means:

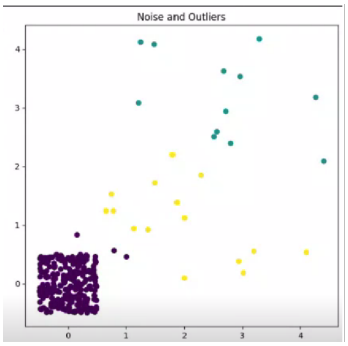
1. **Dataset with difference variances**: Since k-means uses distance to centroids, points in high-variance regions get assigned greater distances, leading to skewed clusters.
   Better Alogrithm: Gaussian Mixture Models (GMM) - GMM assumes data points come from a mixture of Gaussian distributions (bell curves). It can handle clusters with different variances by fitting individual Gaussians to each cluster.



2. **Dataset are not in spherical cluster shape** : K-means works best with spherical clusters, where points are evenly distributed around a central point. Since K-means minimizes the sum of squared distances to centroids. This measurement might not accurately reflect the actual shape of non-spherical clusters.
   Better Alogrithm: DBSCAN. DBSCAN doesn't rely on pre-defined shapes. It focuses on identifying dense regions (clusters) of points separated by areas with low density (noise).



3. **Datasets contains noise and outliers** : K-means is sensitive to outliers (distant data points) and noise (random data points). Outliers can distort the centroids, leading to poorly defined clusters.
   Better Algorithm: DBSCAN - DBSCAN is robust to outliers. It can identify core points (densely surrounded) and separate them from outliers, which are typically isolated points with few neighbors.



Problem 4

GMM algorithm

1. Write Loss function for the algorithm
2. The word loss function indicates the approach of the GMM algorithm to optimize this loss (E-M in slides). Explain the meaning of E-step and M-step (using math formula)

Solution.

Objective function for GMM is log-likelihood function:

$$I(\theta) = \sum_{n=1}^{N} log(\sum_{k=1}^{K} \pi_k N(x_n|\mu_k, \Sigma_k)$$

Taking partial derrivatives and setting to zero, we have:

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N}$$

where $\gamma(z_{nk})$ is the responsibility of component k for data point.
We see that three model parameters $\{\mu_k, \Sigma_k, \pi_k\}$ depend on $\gamma(z_{nk})$
$\Longrightarrow$ iterative scheme can be used.

EM Steps:
First we randomly initialize the parameters $\{\mu_k, \Sigma_k, \pi_k\}$, then iterative the following 2 steps (Repeat until convergence):

- **E-step** (Expectation Step): Calculate responsibilities using current parameters:

$$\gamma(z_{nk}) = \frac{p(z_{nk} = 1)p(x_n|z_{nk} = 1)}{\sum_{j=1}^{K} p(z_{nj} = 1)p(x_n|z_{nj} = 1)}$$

- **M-step** (Maximization Step): Re-estimate parameters using these $\gamma(z_{nk})$:

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$
$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T$$
$$\pi_k = \frac{N_k}{N}$$

Problem 5

List the steps of the GMM algorithm

Solution.

Steps of the GMM

1. Decide number of Distributions
2. Initialize random mean and variance for each distribution
3. **E Step**: Generate distribution based on mean and variance coming from previous step
4. **M Step**:
   - Calculate the likelihood of each data point belong to each distribution above generated.
   - And update the mean and variance, in order to maximize the likelihood for each data point.
5. Repeat until convergence.