Arda Carkot BATI
21302578

# CS-464 Machine Learning

# HW#3

1) For two class case we defined within class scatter matrices as:

$S_W = S_1 + S_2$, where for example $S_1 = \sum_{x \in w_1} (x - \mu_1)(x - \mu_1)^T$

For the multiple class case, let there be N classes. We can define the new within class scatter matrix as:

$$S_W = S_1 + S_2 + S_3 + \cdots + S_N$$

From our lecture notes and current argument, the between class scatter matrix can be written as: → given in the lecture notes

$$S_B = \sum_{i=1}^{N} M_i (\mu_i - \mu)(\mu_i - \mu)^T \quad \text{where } M_i = \#\text{ data points in the class } i$$

We have defined the Fisher criterion as:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$ in our lecture notes. We have to maximize the criterion, $\arg\max_w \frac{w^T S_B w}{w^T S_W w}$. In the two class case we found one projection w that maximized this function. For the N class case we can find at most N-1 projection vectors $w_1$ to $w_{N-1}$.

From the lecture notes: $S_W = \sum_{i=1}^{c} S_i$ where $S_i$ is defined above, $\mu_i = \frac{1}{M_i} \sum_{x \in w_i} x$

For the two class case we solved the eigen value problem $S_W^{-1} S_B w - Jw = 0$ to find the w projection in our lecture notes.

For the multiple class case, some expressions can be used →

the solutions ($w_1$ to $w_{N-1}$) will be the eigen values of this expression.

we will write $S_W^{-1} S_B w = \lambda w$, finding the eigen values by

$|S_W^{-1} S_B - \lambda I| = 0$, eigen vectors by $S_W^{-1} S_B w_i^* = \lambda_i w_i^*$

for each $\lambda_i$ eigen value a $w_i^*$ eigen vector will be found.

2) a) I have applied the K-means algorithm 1000 seperate times (with 1000 random starting points.) I did this to prevent k-means getting stuck in Local minima. The inner loop of k-means algorithm repeated 200 times each time.

Among the 1000 seperate trials the best k-means performance was 0.725 accuracy. For this accuracy the confusion matrix was as follows Confusion M = $\begin{bmatrix} 100 & 0 \\ 55 & 45 \end{bmatrix}$ $\left( \begin{bmatrix} 1,1 & 1,2 \\ 2,1 & 2,2 \end{bmatrix} \right)$

$(i,j) \Rightarrow i$ = true label
$j$ = predicted label

The accuracy and confusion matrix of the algorithm change slightly during each run. I don't want to do more than 1000 trials so I will leave it as it is.

For the best accuracy solution the cluster plot was drawn under the results section of Q2)A). Class 1 is represented by red circles and Class 2 by green circles. The centroid of class 1 is the blue circle, centroid of class 2 is the blue x.

Discussion: K-means doesn't perform very well The real classes have different sizes, different densities. Also their centroids are very close to each other. K-means doesn't effectively split these classes

2) a) continued : because 2-D feature dimension counts the algorithm

As the data distribution is symmetric overall, k-means tries to split it into two with a linear line, and makes mistakes.

2) b) $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$

$= \langle \Phi(x_i)^T \Phi(x_j) \rangle$

$$J(D) = \sum_{i=1}^{N} \sum_{k=1}^{K} \delta_{ik} \cdot \| \Phi(x_i) - m_k \|^2$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} \delta_{ik} \left( K(x_i, x_j) - 2 \sum_{i=1}^{N} \delta_{ik} K(x_i, x_j) + \frac{\sum_{i=1}^{N} \sum_{l=1}^{N} \delta_{ik} \delta_{il} K(x_i, x_j)}{\sum_{i=1}^{N} \sum_{l=1}^{N} \delta_{ik} \delta_{il}} \right)$$

2) c) Polynomial Kernel can be written as

$k(x, x') = (x^T x')^2$,

$$k\left\langle \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \right\rangle = (x_1 x_2 + y_1 y_2)^2$$

$$= (x_1 x_2)^2 + 2 x_1 y_1 x_2 y_2 + (y_1 y_2)^2$$

$$= \begin{pmatrix} x_1^2 & \sqrt{2} x_1 y_1 & y_1^2 \end{pmatrix} \begin{pmatrix} x_2^2 \\ \sqrt{2} x_2 y_2 \\ y_2^2 \end{pmatrix}$$

So for the kernel k-means, I will use the following transformation

$x_k = x^2$, $y_k = \sqrt{2} xy$, $z_k = y^2$. I have to adjust my data from 2D to 3D, and also select the starting points in 3D random. I applied the kernel k-means algorithm 15 times with 15 random starting points. In each run the algorithm repeated the main loop only 2 times. I calculated the accuracy for each of the 15

iterations. All of them gave 100% accuracy. I reported the Confusion matrix for one of them $CM = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}$, same in all of them.

I also plotted the the distribution. The coloring and centroid representation is the same as Q2 part A.

Kernel-k means performs perfectly with this distribution. There are no errors, and even with different starting points the algorithm finds the correct distribution in only 2 iterations. It is definetly better than k-means for this distribution. Clustering the data in 3D is much easier than in 2D for this particular distribution

3) a) The principle components were found by pca(..) function of MATLAB. This function returns the PCA coefficients as well as eigen values of each component. The PCA components are already given in descending order by the function.

I plotted the resulting eigen values for each 400 PC's. The plot reveals that after the 100th PCA component, the rest has very low eigen values. Therefore, I would choose somewhere between 100 to 150 of the top PC's. This way I would be able to represent the data with very minimal error, and not use too many components.

3) b) The Top 5 and Bottom 5 principal component's were drawn in two seperate plots. For the top 5 components, shapes similar to digits can be observed. Bottom 5 components include mostly grey pixels with no particular shape. During reconstruction the top 5 will be much more successfull, as they capture the digit characteristics much better.