CS 240 : Lab 10
PCA, LDA, and Clustering

TAs : Kartik Gokhale, Drashthi Doshi

Instructions

- This lab will be **graded**. The weightage of each question is provided in this PDF.
- Please read the problem statement and the submission guidelines carefully.
- All code fragments need to be written within the `TODO` blocks in the given Python files. Do not change any other part of the code.
- **Do not** add any **additional** *print* statements to the final submission since the submission will be evaluated automatically.
- For any doubts or questions, please contact either the TA assigned to your lab group or one of the 2 TAs involved in making the lab.
- The deadline for this lab is **Sunday, 24 March, 5 PM**.
- The submissions will be checked for plagiarism, and any form of cheating will be appropriately penalized.

The submissions will be on Gradescope. You need to upload the following Python files: `q1.py`, `q2.py` and `q3.py`. In Gradescope, you can directly submit these Python files using the upload option (you can either drag and drop or upload by browsing). No need to create a tar or zip file.

1 Principal Component Analysis [35 marks]

The manifold hypothesis posits that many high-dimensional data sets that occur in the real world actually lie along low-dimensional latent manifolds inside that high-dimensional space. In this question, we will explore Principal Component Analysis (PCA), a technique by which you hope to capture the data in a vector space having dimension much lower than the dimension of its embedding space, without losing much information contained in the data. As the name suggests, the central idea is to identify the principal "components" of a given set of data samples.

1.1 PCA and Dimensionality Reduction

The principal components of a collection of points in a real coordinate space are a sequence of p unit vectors, where the i-th vector is the direction of a line that best fits the data while being orthogonal to the first $i-1$ vectors. Let us mathematically formalise the concept of "best-fitting" the data.

Consider a dataset $\mathcal{D}$ consisting of N samples of points $(x_1, x_2, \ldots, x_N)$ where each $x_i \in \mathbb{R}^d$. The goal of PCA is to identify $k(k < d)$ unit vectors $(p_1, p_2, \ldots, p_k)$, such that the loss defined as

$$\text{MSE}_k = \frac{1}{N} \sum_{i=1}^{N} \|x_i - \hat{x}_i\|^2 \tag{1}$$

where $\hat{x}_i$ is the projection of x_i onto the subspace formed by the unit vectors $p_1, p_2, \ldots, p_k$, is minimised. These basis vectors at which the loss is minimised are called the principal components. We can use the above loss to sort the basis vectors according to their "relevance". For a given set of basis vectors, let the value of the loss defined above be L_1. Now, suppose you remove a vector from the basis, say p_i and consider the loss with respect to the new subspace formed by the remaining vectors (with p_i removed), say L_2. Then, the relevance of basis vector p_i is $L_2 - L_1$

1.1.1 Implementing PCA

We will be implementing PCA on the mnist dataset. In the function pca, you will be given the MNIST dataset images, comprising images of handwritten digits, as well as their labels. For the points of the target class, you must find a subspace of dimensionality k that optimises the loss as stated above. The principal components must be sorted in descending order of their relevance (as defined above) where ties can be resolved arbitrarily. Refer to the code files for more details

1.1.2 Projection

Given the basis vectors that you determined above, find the projection of each of the data points $(\hat{x}_i)$ of the target class onto these k basis vectors. Now, you will have a k-dimensional representation of each of the data points, where $k < d$. Thus, you have compressed these images.

1.2 Tasks

You need to complete the following classes and functions inside `q1.py`:

- Implement function `pca` as described above. [25 marks]
- Implement function `projection` as described above. [10 marks]

2 Linear Discriminant Analysis [25 marks]

Linear discriminant analysis (LDA), normal discriminant analysis (NDA), or discriminant function analysis is a generalization of Fisher's linear discriminant, a method used in statistics and other fields, to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.

It differs from PCA as can be seen in the following diagram

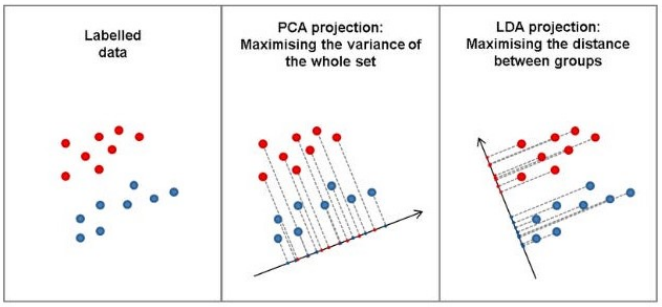

Figure 1: LDA and PCA

LDA explicitly attempts to model the difference between the classes of data. PCA, in contrast, does not take into account any difference in class

In this question, you will implement LDA on the MNIST dataset. In the function fit, you are given the MNIST dataset and must find the Linear Discriminant array of size $[d, 1]$ where d is the dimension of each point. Then, much like you did in PCA, you must return the features projected onto 1-dimensional space

2.1 2 Class LDA

When the number of classes in the dataset is 2. The goal of performing LDA is to find a projection vector that maximizes the below function.

$$\max_{v} \quad \frac{(v^\top m_1 - v^\top m_2)^2}{s_1^2 + s_2^2} \tag{2}$$

$$s_1^2 = \sum_{x \in C_1} (v^\top x - v^\top m_1)^2, \; s_2^2 = \sum_{x \in C_2} (v^\top x - v^\top m_2)^2$$

where m_1, m_2 are the mean values of the first and second classes respectively. And s_1^2, s_2^2 are the variances of the two classes. The goal of PCA is to maximize the distance of the projected means of the two classes while minimizing their variance.

$$\begin{aligned}(v^\top m_1 - v^\top m_2)^2 &= (v^\top (m_1 - m_2))^2 \\ &= v^\top (m_1 - m_2) \cdot (m_1 - m_2)^\top v \\ &= v^\top S_b v\end{aligned}$$

$$S_b = (m_1 - m_2)(m_1 - m_2)^\top$$

$$
\begin{aligned}
s_1^2 &= \sum_{x \in C_1} (v^\top x - v^\top m_1)^2 \\
&= \sum_{x \in C_1} v^\top (x - m_1)(x - m_1)^\top v \\
&= v^\top \left(\sum_{x_i \in C_j} (x_i - m_1)(x_1 - m_1)^\top \right) v \\
&= v^\top S_1 v,
\end{aligned}
$$

The same holds for s_2^2

$$
S_i = \sum_{x \in C_i} (x - m_i)(x - m_i)^\top \tag{3}
$$

$$
S_w = S_1 + S_2 = \sum_{x_i \in C_1} (x_i - m_1)(x_i - m_1)^\top + \sum_{x_i \in C_2} (x_i - m_2)(x_i - m_2)^\top \tag{4}
$$

Therefore optimizing the given objective function is the same as

$$
\max_v \quad \frac{v^\top S_b v}{v^\top S_w v} \tag{5}
$$

Maximizing the above objective is the same as finding the eigenvector corresponding to the largest eigenvalue of the matrix $S_w^{-1} S_b$

2.2 Multi Class LDA

For a classification problem with more than 2 classes the above 2 Class LDA can be extended to a $K = \{1, 2, ..., k\}$ Class LDA as follows: The objective remains as in Eq 5 with S_w as defined in Eq 4 and

$$
S_b = \sum_{i \in K} n_i (m - m_i)(m - m_i)^\top \tag{6}
$$

where n_i is the number of samples in class i.

2.3 Implementation

The goal is to implement multi-class LDA on the MNIST dataset. Construct the matrices S_w and S_b. Find and return the required eigenvector(s). Project the feature matrix onto the required eigenspace.

2.4 Tasks

You have the following tasks to complete in `q2.py`:

- Complete the function `fit` as described above. [15 marks]
- Complete the function `transform` as described above. [10 marks]

3 k-means Clustering [40 marks]

One of the most popular type of analysis under unsupervised learning is Cluster analysis. One way of clustering data is the method of k-means. Let us formally state the problem.

3.1 Setup of k-means

Given a set of observations $(x_1, x_2, ..., x_n)$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into $k(\leq n)$ sets $S = \{S_1, S_2, ..., S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg\min_{S,\{\boldsymbol{\mu}_i\}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \min_{S} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i \tag{7}$$

where $\boldsymbol{\mu}_i$ is the representative point associated with cluster S_i. Our goal is to learn these representative points for each of the k clusters and determine which cluster each point belongs to. As discussed in class, this problem is NP-hard and hence we employ a heuristic method called the k-means clustering which converges to a local minima.

We will be following the following two step algorithm to find the clusters

- Assign each point to the cluster corresponding to the representative point it is closest to.
- Update each representative point to the centroid of all the points assigned to it.

This is guaranteed to terminate (do not forget a suitable termination condition) and converge to a local optima. However, the local optima converged to depends on the initialisation. The fit function updates the centroids as per the algorithm stated.

3.1.1 Initialisation

We will be using the k-means++ method for initialising the clusters. The intuition behind this approach is that spreading out the k initial cluster centers is a good thing: the first cluster center is chosen uniformly at random from the data points that are being clustered, after which each subsequent cluster center is chosen from the remaining data points with probability proportional to its squared distance from the point's closest existing cluster center. Initialisation initialises the k-centroids.

3.1.2 Evaluation

Given the centroids and the data, the evaluation simply assigns each point to a cluster and returns the centroids in order of the cluster index and the cluster index associated with each point.

3.2 Tasks

You need to complete the following classes and functions inside `q3a.py`:

- Implement function `initialise` as described above. [15 marks]
- Implement function `fit` as described above. [15 marks]
- Implement function `evaluate` as described above. [10 marks]