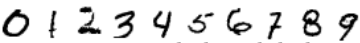



The objective of this lab is for you to program PCA and LDA in Matlab and apply them to some datasets. The TA will first demonstrate the results that PCA and LDA give on the MNIST dataset, and then you will program them, replicate those results, and further explore the datasets with PCA and LDA. You can use the textbook, lecture notes and your own notes.

## I Datasets

You will use the MNIST dataset of handwritten digits . PCA will need the instances  $\mathbf{x} \in \mathbb{R}^D$  (where  $D = 784$ ), while LDA will need both the instances and their labels  $y_n \in \{0, \dots, 9\}$ . You will need to plot instances as grayscale images of  $28 \times 28$ , as seen in previous labs; and “eigendigits” as color images. You will also need to plot reduced-dimension instances  $\mathbf{z}_n \in \mathbb{R}^L$  (where  $L$  is 1D, 2D or 3D) as scatterplots; color them differently for each class, so we can tell them apart.

Additionally, you will apply PCA and LDA to the rotated-7 MNIST dataset . Here, each digit ‘7’ should be considered as a class containing all its rotated versions.

## II Implementing and using PCA

**Important:** when developing and testing your code, use toy examples for which you know the true solution ahead of time (e.g. generate points along a line in 3D and add noise to them, then reduce to 1D or 2D with PCA). Once your code works well there, try it on more difficult datasets.

- Assume a matrix  $\mathbf{X}$  of  $D \times N$  (instances = columns).
- Start by computing the mean  $\boldsymbol{\mu}$  and covariance  $\boldsymbol{\Sigma}$  of the data. Program this using loops, then check the result with the Matlab functions `mean` and `cov`.
- Program PCA by computing the eigendecomposition of the covariance matrix  $\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$  and setting  $\mathbf{W} = \mathbf{U}_{1:L}$ .
- Program how to project a point  $\mathbf{x} \in \mathbb{R}^D$  onto the  $L$  principal component subspace (where  $1 \leq L \leq D$ ), and how to reconstruct a vector  $\mathbf{z} \in \mathbb{R}^L$  into the original, data space. This is given by the PCA projection mapping  $\mathbf{z} = \mathbf{F}(\mathbf{x}) = \mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu})$  and the reconstruction mapping  $\mathbf{x}' = \mathbf{f}(\mathbf{z}) = \mathbf{W}\mathbf{z} + \boldsymbol{\mu}$ , respectively.
- Verify that the covariance matrix in the projected space (that is,  $\text{cov}\{\mathbf{z}_1, \dots, \mathbf{z}_N\}$ ) equals  $\mathbf{W}^T\boldsymbol{\Sigma}\mathbf{W}$ , that it is diagonal, and that the sum of its diagonal elements equals  $\lambda_1 + \dots + \lambda_L$ .
- Plot the following figures:
  1. The eigenvalues  $\lambda_1, \dots, \lambda_D$  and the proportion of explained variance  $\frac{\lambda_1 + \dots + \lambda_L}{\lambda_1 + \dots + \lambda_D} \in [0, 1]$  as a function of the number of dimensions used  $L$  (as in the textbook fig. 6.4).
  2. The mean  $\boldsymbol{\mu}$ , as a grayscale image.
  3. The MNIST dataset projected onto 2D (as in the textbook fig. 6.5). Use different colors/markers for different digit classes, so we can recognize them.
  4. The MNIST dataset projected onto 3D, colored as in the 2D plot.
  5. The eigenvectors  $\mathbf{u}_1, \dots, \mathbf{u}_L \in \mathbb{R}^D$ , as color images (“eigendigits”).
  6. A vector  $\mathbf{x}$  and its reconstruction  $\mathbf{x}' = \mathbf{W}(\mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu})) + \boldsymbol{\mu}$ , both as grayscale images.
  7. Vectors of the form  $\boldsymbol{\mu} \pm \alpha \mathbf{u}_l$  for  $\alpha > 0$  (where  $1 \leq l \leq D$ ), as grayscale images. This shows what the  $l$ th principal component subspace corresponds to in data space. It is equivalent to reconstructing vectors  $\mathbf{z} \in \mathbb{R}^L$  that move along the  $l$ th PC axis.
- Explore the algorithm in different settings:
  - Compute PCA on the entire MNIST dataset (all digits), then visualize it and reconstruct digits.

- Compute PCA on only the digits 1s, then visualize it and reconstruct digits (1s, 2s, etc.). The projection on the first two PCs shows a clear structure, what does it correspond to? Why does the mean  $\mu$  look the way it does?

The following Matlab functions will be useful (among others): `mean cov eig sort find linspace`.

### III Implementing and using LDA

- Assume a matrix  $\mathbf{X}$  of  $D \times N$  (instances = columns) and a vector  $\mathbf{y}$  of  $1 \times N$  (class labels).
- Program how to compute the within-class and between-class scatter matrices  $\mathbf{S}_W$  and  $\mathbf{S}_B$ .
- Program LDA by computing the eigendecomposition of  $\mathbf{S}_W^{-1}\mathbf{S}_B = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$  and setting  $\mathbf{W} = \mathbf{U}_{1:L}$ .
- Program how to project a point  $\mathbf{x} \in \mathbb{R}^D$  onto the LDA subspace of dimension  $L$  (where  $1 \leq L \leq K - 1$ ). This is given by the LDA projection mapping  $\mathbf{z} = \mathbf{F}(\mathbf{x}) = \mathbf{W}^T \mathbf{x}$ .
- Plot the following figures:
  1. The eigenvalues  $\lambda_1, \dots, \lambda_D$  and the proportion of explained variance  $\frac{\lambda_1 + \dots + \lambda_L}{\lambda_1 + \dots + \lambda_D} \in [0, 1]$  as a function of the number of dimensions used  $L$  (as in the textbook fig. 6.4).
  2. The mean of each class  $\mu_k$ , as a grayscale image.
  3. The MNIST dataset projected onto 2D (as in the textbook fig. 6.12). Use different colors/markers for different digit classes, so we can recognize them.
  4. The MNIST dataset projected onto 3D, colored as in the 2D plot.
  5. The eigenvectors  $\mathbf{u}_1, \dots, \mathbf{u}_L \in \mathbb{R}^D$ , as color images (“Fisherdigits”).
- Explore the algorithm in different settings:
  - Use datasets with different numbers of classes (different digits).

Questions to consider:

- How does the result of LDA differ from that of PCA? In particular, observe the 2D projections and the eigendigits and Fisherdigits.
- How many eigenvalues are nonzero in LDA (and how many in PCA)? Why? Remember that LDA applies if  $\mathbf{S}_W$  is invertible and  $L \leq K - 1$ .

Practical advice:

- Machine learning algorithms can have a high time or space complexity, so to get a result in a few seconds you may need to run them on small datasets. You can do this by selecting a random sample of a given dataset.
- Machine learning algorithms often are randomized. Likewise, toy datasets are usually generated randomly. To make sure you can generate the exact dataset multiple times and run an algorithm and get the same result every time, fix the seed of the pseudorandom number generator. In Matlab: `rng(1778)`; where 1778 is the seed. You can also save a toy dataset for later use.
- Matlab tips:
  - To suppress extra line feeds: `format compact`.
  - To get more decimals: `format long`.
  - To compare two matrices or vectors (by finding the largest difference): `max(abs(A(:)-B(:)))`.
  - To avoid distorted plots: `daspect([1 1 1])`.
  - To plot grayscale images with values in  $[0, 1]$ : `colormap(gray(256)); imagesc(I,[0 1])`;  
To plot images with negative and positive values: `colormap(parula(256)); imagesc(I)`;