Homework 4: Unsupervised Learning

Print to PDF

Contents

> ⚠ **Warning**
>
> The submission of the homeworks has **NO** deadline. You can submit them whenever you want, on Virtuale. You are only required to upload it on Virtuale **BEFORE** your exam session, since the Homeworks will be a central part of the oral exam.
>
> You are asked to submit the homework as one of the two, following modalities:
>
> - A PDF (or Word) document, containing screenshoots of code snippets, screeshots of the results generated by your code, and a brief comment on the obtained results.
> - A Python Notebook (i.e. a `.ipynb` file), with cells containing the code required to solve the indicated exercises, alternated with a brief comment on the obtained results in the form of a markdown cell. We remark that the code **SHOULD NOT** be runned during the exam, but the student is asked to enter the exam with all the programs **already executed**, with the results clearly visible on the screen.
>
> Joining the oral exam with a non-executed code OR without a PDF file with the obtained results visible on that, will cause the student to be rejected.

Exercise 1: Compute the SVD of a Matrix & Verify the Reconstruction

1. Construct any non-square matrix $A \in \mathbb{R}^{m \times n}$ (e.g. $m = 10$, $n = 6$) with random or structured entries.
2. Compute its SVD:

$$A = U\Sigma V^T.$$

3. Verify numerically that:

$$A \approx U\Sigma V^T, \qquad \|A - U\Sigma V^T\|_F \approx 0.$$

4. Print and plot the singular values $\sigma_1, \ldots, \sigma_{\min(m,n)}$.
5. Comment on why singular values appear in descending order, why small singular values correspond to "less important" directions, and why floating-point arithmetic makes exact zeros rare.

Exercise 2: Best Rank-k Approximation

1. Implement a function:

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

2. For several values of k, compute:

$$E(k) = \|A - A_k\|_F.$$

3. Plot the approximation error $E(k)$ vs k.
4. Explain why SVD gives the *optimal* rank-k approximation.

Exercise 3: Image Compression with SVD

Using the `cameraman` image (as in the notes):

1. Load the image as a matrix $X \in \mathbb{R}^{512 \times 512}$.
2. Compute its SVD:

$$X = U\Sigma V^T.$$

3. For $k \in \{5, 20, 50, 100, 200\}$:
   - Compute the rank-k approximation X_k,
   - Plot each reconstructed image,
   - Compute the compression factor:

$$c_k = 1 - \frac{k(m+n+1)}{mn}.$$

   - Plot reconstruction error vs k,
   - Plot compression factor vs k.
4. Comment on:
   - How visual quality improves with k,
   - Why most of the "energy" is contained in the first singular values,
   - The trade-off between compression and fidelity,
   - The connection between SVD and optimal low-rank approximation.

Exercise 5: PCA + Clustering on a Real Dataset

Use the MNIST Kaggle dataset: https://www.kaggle.com/datasets/animatronbot/mnist-digit-recognizer.

To reduce computational burden, **filter only digits 3 and 4** exactly as in the teaching notes.

Do as follows:

1. Load the CSV file into memory (you may use NumPy or pandas).
2. Split into training and test sets.
3. Center the training data:

$$X_{c,\text{train}} = X_{\text{train}} - c(X_{\text{train}}).$$

4. Compute the reduced SVD of the centered training data.
5. Choose $k = 2$ principal components.
6. Project:
   - $Z_{\text{train}} = X_{c,\text{train}} V_2$,
   - $Z_{\text{test}} = X_{c,\text{test}} V_2$.

7. Plot:
    - Scatterplot of Z_{train} colored by labels (3 vs 4),
    - Scatterplot of Z_{test} overlaid in the same plot.
8. Repeat with digits 5 vs 8, or 1 vs 7 and discuss on the different results.
9. Finally, try $k = 3$ and use a 3D scatterplot.

Exercise 6: Classification After PCA: Linear Classifier and Centroid Classifier

In this exercise, you will build two simple classifiers **after projecting the data with PCA**:

1. **A linear classifier on 2D PCA features**,
2. **A PCA-centroid classifier** that assigns each test point to the closest class centroid in the reduced space.

You will perform all experiments on the same Kaggle dataset used previously: https://www.kaggle.com/datasets/animatronbot/mnist-digit-recognizer, filtering only digits 3 and 4, as you did in the previosu exercise.

The aim of this exercise is to connect dimensionality reduction with classification and illustrates how PCA can be used not only for visualization, but also as a preprocessing tool for machine learning pipelines.

Step 1: PCA Projection

Use exactly the same PCA workflow as in Exercise 6:

1. Load and split digits {3,4} into `train` and `test`.
2. Center the training data:

$$X_{c,\text{train}} = X_{\text{train}} - c(X_{\text{train}}).$$

3. Compute the reduced SVD:

$$X_{c,\text{train}} = U\Sigma V^T.$$

4. Take the first two principal directions:

$$P = V_2 \in \mathbb{R}^{d\times 2}.$$

5. Project:

$$Z_{\text{train}} = X_{c,\text{train}} P, \qquad Z_{\text{test}} = (X_{\text{test}} - c(X_{\text{train}}))\, P.$$

You will now use these 2D representations Z_{train} and Z_{test} for classification.

Step 2: Fit a linear classifier

1. Use the logistic regression model:

   $$f_\Theta(z) = \sigma(\Theta^T z), \qquad z \in \mathbb{R}^2,$$

   with sigmoid and binary cross-entropy loss, and train the model using SGD.

2. Once trained:
   1. Plot the PCA-reduced training data in 2D,
   2. Plot the decision boundary defined by:

   $$\Theta^T z = 0 \quad \Rightarrow \quad \text{boundary line.}$$

3. Compute, on the test set:
   - Accuracy,
   - Precision and Recall,
   - Confusion matrix.

Step 3: PCA-Centroid Classifier (Nearest-Centroid in PCA Space)

This classifier does **not** use a linear model. It relies purely on geometry in the reduced PCA space.

1. For each class $c \in \{3, 4\}$, compute the centroid in PCA space:

   $$\mu_c = \frac{1}{|\mathcal{D}_c|} \sum_{z_i \in \mathcal{D}_c} z_i.$$

   Plot both centroids μ_3 and μ_4 in the PCA plane.

2. Given a test sample $x \in \mathbb{R}^d$:
   1. Project it:

$$z = (x - c(X_{\text{train}})) P.$$

2. Compute distances:

$$d_3 = \|z - \mu_3\|_2, \qquad d_4 = \|z - \mu_4\|_2.$$

3. Assign:

$$\hat{y} = \arg \min_{c \in \{3,4\}} d_c.$$

This is the **nearest-centroid classifier in PCA space**.

3. Compute, on the test set:
   - Accuracy,
   - Precision and Recall,
   - Confusion matrix.

   Plot the test points with colors given by classification results.

Step 4: Compare accuracy and error patterns

1. Compare linear classifier on PCA features vs centroid classifier in PCA space.
2. On the PCA scatterplot, plot:
   - the linear boundary,
   - the two centroids,
   - misclassified test samples highlighted in red.
3. Repeat by changing the digits in $\{(1, 7), (5, 8), (2, 3)\}$ and observe how class similarity influences:
   - the cluster shape,
   - centroid separation,
   - decision boundaries.