SI221 Practical assignment #7: Principal Component Analysis and Boosting - 04/11/2020

Instructions

Read all the instructions below carefully before you start working on the assignment.

- Please submit the source code and a pdf report of your work. You can use the Jupyter notebook instead, by submitting the ipynb file. Each file must have as title: *TP_PCA_Boosting_Student1_Student2*.
- Late assignments will not be corrected.
- You must do this assignment in groups of 2. Please submit no more than one submission per group.
- You must implement all algorithms from scratch.
- Code that does not work will not be considered.
- Send your work to: `nazareth@telecom-paris.fr` and `colombo.pierre@gmail.com`

Practical assignment objective

- To implement the PCA from scratch and apply it for dimensionality reduction and image denoising.
- To implement the AdaBoost algorithm from scratch in order to learn Boosting technique.

Working with PCA[1]

Large data sets are increasingly common and are often difficult to interpret. In order to interpret such data sets, methods are required to drastically reduce their dimensionality in an interpretable way, such that most of the information in the data is preserved. Many techniques have been developed for this purpose, but principal component analysis (PCA) is one of the oldest and most widely used.

[1]Reference paper: "Principal component analysis: A review and recent developments" by I. T. Jolliffe and J. Cadima, 2016.

Principal component analysis (PCA) is a technique for reducing the dimensionality of such data sets, increasing interpretability but at the same time minimizing the reconstruction error. Thus, PCA seeks the linear projection of the data in a space of reduced size by minimizing the mean square error between the input data and its recovered version.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be n vectors in $\mathbb{R}^d$ that represent n samples described by d features. Let's reduce the dimensionality of the input vectors from the original space of d variables to a new space of p variables, such that $p < d$. Using PCA, we aim to find an orthogonal matrix $\mathbf{P} \in \mathbf{R}^{d \times p}$ so that

$$\underset{\mathbf{P} \in \mathbb{R}^{d \times p}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \mathbf{P}\mathbf{P}^{\mathrm{T}} \mathbf{x}_i \right\|^2 . \tag{1}$$

PCA consists of the following steps (via Eigendecomposition):

1. A common practice is to center the data before applying PCA, i.e.: compute the sample mean $\mu = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$ and then $\bar{\mathbf{x}}_i = \mathbf{x}_i - \mu$ in order to compute the mean-subtracted feature matrix $\bar{\mathbf{X}} = [\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_n]^{\mathrm{T}} \in \mathbb{R}^{n \times d}$.

2. Compute the eigendecomposition of $\bar{\mathbf{X}}^{\mathrm{T}} \bar{\mathbf{X}}$ to identify the principal components[2].

3. Form a projection matrix $\mathbf{P} \in \mathbb{R}^{d \times p}$ containing p eigenvectors associated to the p largest eigenvalues (computed in the previous step).

4. Transform the mean-subtracted data to obtain lower-dimensional data through: $\mathbf{Y} = \bar{\mathbf{X}}\mathbf{P} \in \mathbb{R}^{n \times p}$. Observe that the columns of $\mathbf{Y}$ represent the transformed features, which is given by the a linear combination of the input features.

5. Recover the input data by the following transformation: $\hat{\mathbf{X}} = \mathbf{Y}\mathbf{P}^{\mathrm{T}} + \mu \in \mathbb{R}^{n \times d}$.

Reminders on AdaBoost

In this homework we aim at applying the AdaBoost algorithm for a two-class classification problem. The algorithm sequentially applies a weak classification to modified versions of the data. By increasing the weights of the missclassified observations, each weak learner focuses on the error of the previous one. The predictions are aggregated through a weighted majority vote. We work with the Hastie (10.2) dataset.

[2]Remark that the matrix $\bar{\mathbf{X}}^{\mathrm{T}} \bar{\mathbf{X}}$ is proportional to the sample covariance matrix of the data set. In case $n < d$, check the book *"Understanding Machine Learning: From Theory to Algorithms", by Shai Shalev-Shwartz and Shai Ben-David.*

A reminder of the AdaBoost algorithm can be found in algorithm 1.

Algorithm 1: AdaBoost Algorithm

Result: Write here the result

Weight Initialization:

$$for \quad i \in [1, N], \quad \omega_i = \frac{1}{N}$$

while m ¡ M **do**

Fit a classifier G_m a classifier on the training data with weight w_i ;

Compute the error:

$$\text{err}_m = \frac{\sum_{i=1}^{N} \omega_i 1_{y_i \neq G_m(x_i)}}{\sum_{i=1}^{N} \omega_i}$$

Compute

$$\alpha_m = \log[\frac{1 - \text{err}_m}{\text{err}_m}]$$

Update the weights

$$\omega_i = \omega_i \times exp(\alpha_m \times 1_{y_i \neq G_m(x_i)}) \quad i = 1, \cdot N$$

end

Output

$$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$$

Getting started!

1 PCA for dimensionality reduction and image denoising

In this part we will implement the PCA from scratch and apply it for dimensionality reduction and image denoising[3]. Let's use the well-known MNIST data set that is commonly used for training various image processing systems and can be easily loaded from *keras.datasets*[4].

Short description: The MNIST data set is a large data set of handwritten digits (from 0 to 9), containing 60000 gray-scale images for training and 10000 for testing, each image has 28x28 pixels with range of possible values from 0 to 255.

Data Preparation: For simplicity we will consider only the test data set and the first 2000 samples (in case of limited available memory). Re-scale the images to $[0, 1]$ dividing them by 255. Vectorize each image $\mathbf{x}_i \in \mathbb{R}^d$ and form a matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^{\mathrm{T}} \in \mathbb{R}^{n \times d}$. Remark that we will have $d = 784$ and $n = 2000$.

1. Compute the eigendecomposition of the sample covariance matrix and use the eigenvalues to calculate the percentage of variance explained (given by the eigenvalues). Plot the cumulative sum of these percentages[5] *versus* the number of components. Comment. *Hint: Use cumsum from Numpy to calculate the cumulative sum.*

2. Apply the PCA via Eigendecomposition to reduce the dimensionality of the images for each $p \in \{50, 250, 500\}$. Compute the normalized reconstruction error in terms of the Frobenius norm, i.e. $e_p = \|\mathbf{X} - \hat{\mathbf{X}}_p|\,|_{\mathrm{F}} / \|\mathbf{X}|\,|_{\mathrm{F}}$, where $\mathbf{X}$ denotes the input matrix, and $\hat{\mathbf{X}}_p$ denotes the recovered matrix associated to each p. Visualize some recovered images and compare them with their corresponding original images. Comment on what happens when we reduce the number of components p.

3. Now we will apply the PCA for image denoising. Considering the same input matrix, let's add some Gaussian noise[6] with zero mean and variance $\sigma^2 = 0.25$. Visualize the corrupted images and compare them with their corresponding original images, you should obtain a similar figure to Fig. 1. Plot the cumulative explained variance *versus* the number of components, as in the first item. Compare it with the one obtained in the noiseless case and comment.

4. Generate the noisy data, as in the previous item, for each $\sigma^2 \in \{0.15, 0.25, 0.50\}$. Apply the PCA via Eigendecomposition for each σ^2 and fixing $p = 250$. Visualize

[3] Noise reduction or denoising is the process of removal of noise from any signal or data input.

[4] Please do not use the data set available through Sklearn (too low resolution!).

[5] Also, shortly known as *cumulative explained variance*.

[6] Make sure that the range of possible values of the generated noisy data will remain between 0 and 1, as well as the input data.

some recovered images and compare them with their corresponding noisy images. Compute the normalized reconstruction error in terms of the Frobenius norm, obtained for all values of σ^2, with respect to the original images. Comment.

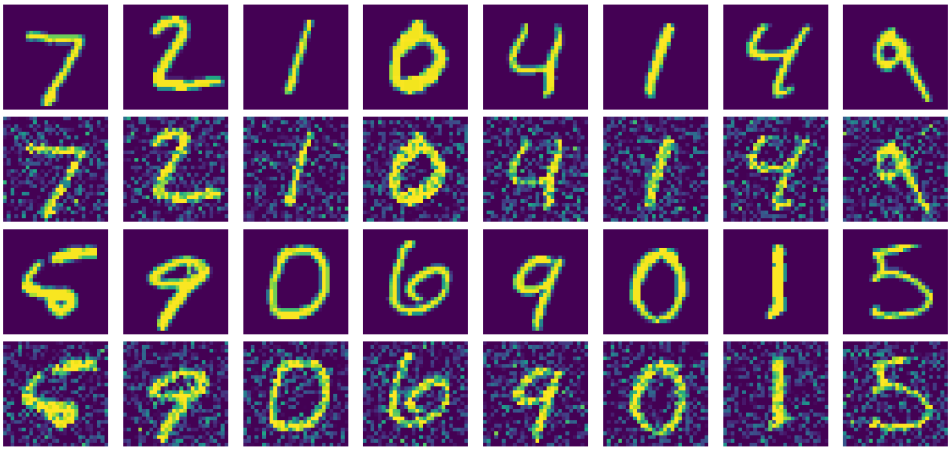

Figure 1: Noisy data generation.

2 AdaBoost: Building A Strong Learner from Weak Learners

In this second exercise you will implement an AdaBoost Classifier. Fill the notebook provided. You should obtain a figure similar Figure 2

Comment the figure.

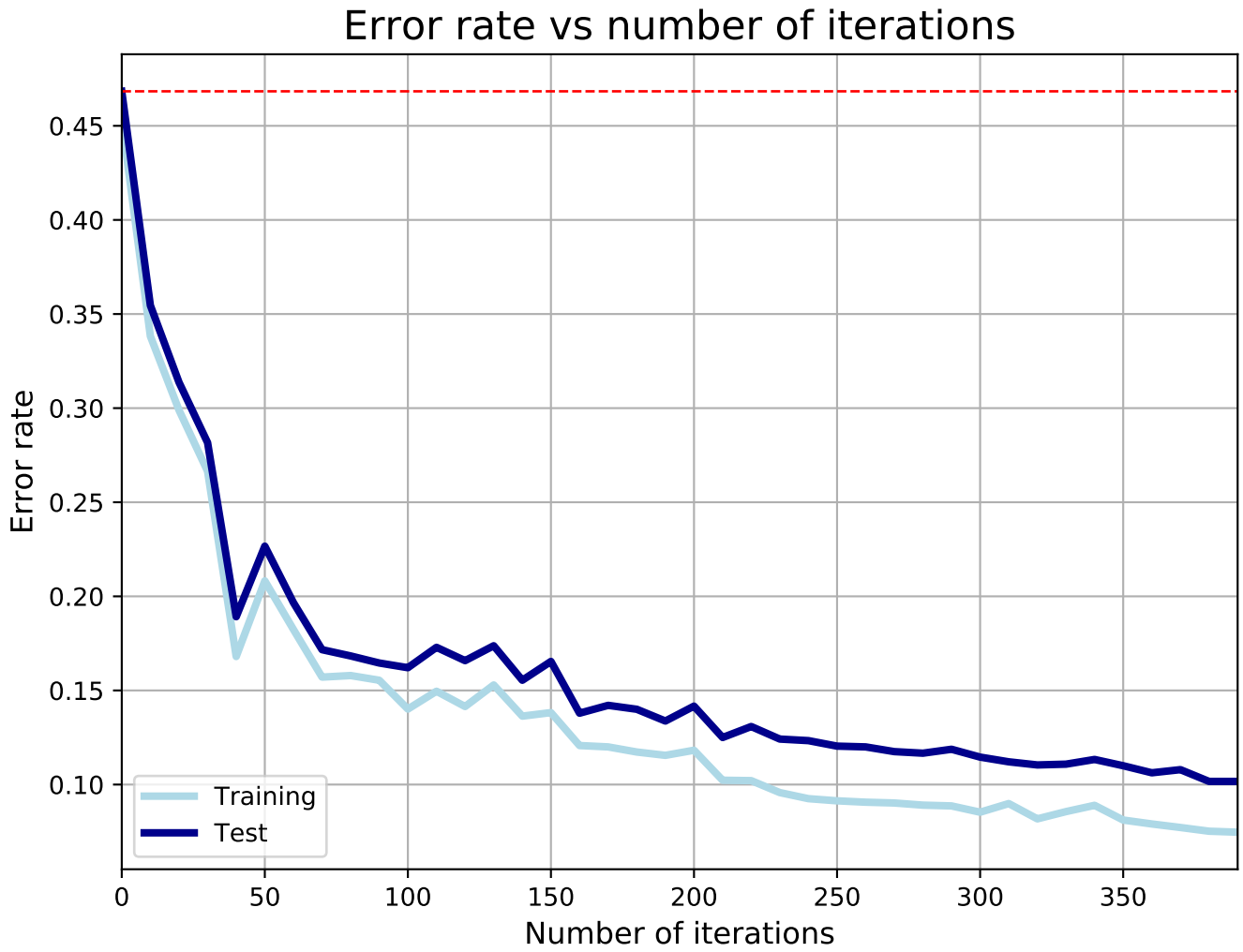

Figure 2: Errors of the AdaBoost Classifier when the number of iteration Increases.