GMDL212, HW #6

Oren Freifeld and Guy Erez
The Department of Computer Science, Ben-Gurion University

Release Date: 09/06/2021
Submission Deadline: 23/6/2021 , 23:59

Abstract

This assignment focuses on dimensionality reduction using PCA and Autoencoders.

Contents

Version Log

- 1.00, 26/5/2021. Initial release.

In this assignment you will implement two dimensionality reduction schemes we saw in class:

1. Principal Component Analysis (PCA)
2. Autoencoder

The data used in the following exercises is a small subset of the MNIST dataset. Tou can download it from the kaggle "Digit Recognizer" challenge. For your convince, the relevant file ("train.csv") is attached to this assignment.

In this assignment you are provided with a utilities file, to help you with data loading, visualisation, training and embedding extraction. More than that you are provided with two files, one for each section containing an initial structure for your solution. This files are there to help, if you fill it is slowing you down don't use it.

1 PCA

Computer Exercise 1 *Implement the PCA procedure in file* `pca.py`. *You can follow this steps:*

1. *Subtract the mean*
2. *Calculate the correlation matrix of the data*
3. *Compute the eigenvectors with the largest eigenvalues. Those eigenvectors will be the projection matrix.*
4. *Project the data using the projection matrix.*

Useful functions:
`sklearn.preprocessing.StandardScaler`
`scipy.linalg.eigh`
`numpy.matmul`.

Remark 1 *You can implement a general PCA and receive as input the desired components. Or, you can implement the version we will use for this exercise, taking the first two principal components.*

Computer Exercise 2 *Use your implementation for PCA to project the provided data from* $\mathbb{R}^{784}$ *to* $\mathbb{R}^2$ *and save a visualisation of the results. You can follow this steps:*

1. *Load the data using* `utils.load_data("train.csv")`.
2. *Use the function you just implemented to get a two dimensional representation of the data.*
3. *Use* `utils.scatter_plot(coordinates,labels,k)` *to visualize the results.*
4. *Save the figure.* ◊

2 Autoencoder

Computer Exercise 3 *Implement the autoencoder class in file* `autoencoder.py`. *You can follow this steps:*

1. *Fill in the* `init` *and* `forward` *functions (use a simple FC net, with as many layers you wish, feel free to browse the web if you fill lost)*
2. *Don't forget to use activation functions between the layers.*
3. *Use the provided boilerplate code to get a dataloader for the data.*
4. *Use* `utils.train(num_epochs,dataloader,model,criterion,optimizer)` *to train the model.*
5. *Observe the loss function to ensure it decreases.* ◊

Computer Exercise 4 *Use your implementation for an autoencoder to project the provided data from* $\mathbb{R}^{784}$ *to* $\mathbb{R}^2$ *and save a visualisation of the results. You can follow this steps:*

1. *Use* `get_embedding(model,dataloader)` *with the trained model and the provided dataloader.*
2. *Use* `utils.scatter_plot(coordinates,labels,k)` *to visualize the results.*
3. *Save the figure.* ◊

Computer Exercise 5 *Fill in the class* `linear_autoencoder` *in file* `autoencoder.py` *with the exact same implementation as in class* `autoencoder` *but* ***without*** *the activation functions. Repeat computer exercise 4 with the linear model.* ◊

3 Comparison

Problem 1 *Look at the three figures you saved in the previous exercises. Speculate the similarities and differences between the three.* ◊

4 Submission Instructions

Provide a .zip file containing the following files:

- The provided files (containing your solutions) (`autoencoder.py,pca.py,utils.py`)
- A PDF file with all three figures and your answer to problem 1 .

Don't forget to follow the general HW instructions from the course Moodle.