UNIVERSITÄT HEIDELBERG
Zukunft. Seit 1386.
IWR Interdisciplinary Center for Scientific Computing

Deep Vision
Summer Semester 19
Prof. Björn Ommer
ommer@uni-heidelberg.de

Exercise sheet 4: Condensing Information

Due on *20/05/2019*, 11:00.
Johannes Haux (johannes.haux@iwr.uni-heidelberg.de).

DISCLAIMER: You are allowed to use `numpy` and `torch` functions for all exercises, as well as code entirely written by you but no other libraries.

Task 1: PCA **(4P)**

When working with data it is important to understand its inherent dynamics. One common tool to do so is the Principal Component Analysis (PCA), which you already saw in the lecture. The idea is simple: Which directions in the high dimensional space my data live in show the highest variance?
In this exercise you will work with MNIST images, visualize their most prominent components and use those to plot a two dimensional distribution of the images.
All code can be found in the accompanying file `ex1.py`.

1. Using the function `load_data` load the train split of MNIST and plot 10 examples with their label as title. Also report some statistics about the data like `min`, `max`, `mean`, `shape`, `dtype`.

2. Convert all images into plain vectors and process them to be centered around 0 in the range of `[-1, 1]`. In the end you should have two arrays of images and labels.

3. Now do the PCA: For that compute the covariance on the array of MNIST vectors. Next compute their eigenvectors and their corresponding eigenvalues. Sort the eigenvectors by their eigenvalues to find the most important ones.

4. Reshape the 10 most important Principal Components to the shape of `[28, 28]` and plot the resulting images. Explain what you are seeing.

5. Project the MNIST vectors onto the 2 most important Principal Components (using the dot product) and plot the resulting points. To get a better overview you can also choose a subset of the points. Color each dot corresponding to its class. Interpret the plot. What can it tell us about the MNIST dataset?

please turn over

Deep Vision
Summer Semester 19
Prof. Björn Ommer
ommer@uni-heidelberg.de

Task 2: Multilayer Perceptron **(4P)**

The Multilayer perceptron was introduced at the beginning of the lectures. It is one the simplest neural networks. We will use it to get to know `pytorch` and how to use automatic backpropagation to train your own models.
All code accompanying this exercise can be found in the file `ex2.py`.

1. Using all skeleton function provided for this exercise, build a 5 layer Neural Network, which accepts MNIST vectors (those from exercise 1) as input and has hidden layers of size 100. The Network should output a vector containing 10 units, one for each MNIST class. All hidden layers should use ReLU activations.

2. Implement a function, which can report the accuracy of a batch of predictions in percent.

3. Train the network for at least 5 epochs and the validate on the test split of the data after each epoch. Do so using the ADAM optimizer supplied by `torch` and set the learning rate to 1×10^{-4}. As a signal to the model we will use the Cross Entropy loss also supplied by `pytorch`.

4. How well does your model perform on the test data?

Task 3: PCA on Perceptron features **(2P)**

Now that we have a trained Network let us take a look at its intermediate or hidden layers.

1. For all MNIST images of the test split, or if you have limited computational resource for a subset of at least 1000 images, store the activation features from after each ReLU call. You should have 4 arrays of shape `[10000, 100]`.

2. For each of the arrays repeat the steps from exercise 1 to find the Principal Components of the resulting vectors and project the activation features onto the two most important PCs.

3. Make a scatter plot of the resulting projected points for the features from the first, second, third and fourth layer. What can you observe?

4. Compare the plots to the scatter plot from exercise 1. What has changed, what is similar?

UNIVERSITÄT HEIDELBERG
Zukunft. Seit 1386.

IWR
Interdisciplinary Center for Scientific Computing

Deep Vision

Summer Semester 19
Prof. Björn Ommer
ommer@uni-heidelberg.de

Note: Submit exactly one ZIP file and one PDF file via Moodle before the deadline. The ZIP file should contain your executable code. Make sure that it runs on different operating systems and use relative paths. Non-trivial sections of your code should be explained with short comments, and variables should have self-explanatory names. The PDF file should contain your written code, all figures, explanations and answers to questions. Make sure that plots have informative axis labels, legends and captions.