Homework 3: Deep Learning

Out May 17; Due May 24, beginning of exercise session*

Kristian Kersting, Alejandro Molina, Jinseok Nam
{kersting, molina, nam}@cs.tu-darmstadt.de

upload link: `https://www.dropbox.com/request/4B0KMkPqUZMHv4uoOIKU`

1. In this homework, you will use a neural network to classify MNIST!. Probably the most famous dataset in the deep learning community[1].

MNIST is a collection of images of handwritten digits. The task here is to classify an image into the correct digits.

You can use the code you have implemented in the previous homework. You are to train a neural network on the train datasets (X_train, Y_train) and then use the test datasets (X_test, Y_test) to make predictions.

X datasets contain the image data, the Ys are the labels.

Implement an NN with a 1-hidden layer (you get to decide on the size) and a 1-output layer of size 10 (each output represents a class).

To train this network, you have to use one-hot-encoding. That is, you convert your numerical label into a vector of 10 dimensions that is all zeros, except for one value that contains a one in the position of the value of the label.

$0 \rightarrow [1,0,0,0,0,0,0,0,0,0]$
$1 \rightarrow [0,1,0,0,0,0,0,0,0,0]$
$\vdots$
$8 \rightarrow [0,0,0,0,0,0,0,0,1,0]$
$9 \rightarrow [0,0,0,0,0,0,0,0,0,1]$

Use sigmoid activation functions and square error loss function 1.

$$\mathcal{L}(\theta; \mathbf{x}, \mathbf{y}) = \frac{1}{M} \sum_{n=1}^{M} \frac{1}{2} \| f_{NN}(\mathbf{x}_n) - \mathbf{y}_n \|^2 \tag{1}$$

The predicted class is obtained by finding the output neuron with the highest value.

*We will discuss the solutions in the exercise session. It is my suggestions that you try to address at least 50% of the exercise questions. Simply try hard to solve them. This way, you will get familiar with the technical terms and with the underlying ideas of the lecture.

[1] http://yann.lecun.com/exdb/mnist/

Use the following code to obtain the training and test datasets as well as to compute the classification reports:

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_mldata
from sklearn.preprocessing import StandardScaler
from sklearn.utils import check_random_state

mnist = fetch_mldata('MNIST original')
X = mnist.data.astype('float64')
y = mnist.target
random_state = check_random_state(0)

X_train, X_test, y_train, y_test = train_test_split(X, y,
    train_size=1000, test_size=300, random_state=random_state)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

expected = y_test
predicted = y_test # here, use a NN to make predictions

print("Classification report:")
print(classification_report(expected, predicted))
print("Confusion matrix:")
print(confusion_matrix(expected, predicted))
```