Homework 4

Nikolaj Takata Mücke

17-03-2022

Due date: 7th of April, 2022
Submission via email: nikolaj.mucke@cwi.nl
Submissions should consist of a *short* `.pdf` report of your findings, and an offline, compilable copy of the jupyter notebook.

In this homework, we are dealing with convolutional neural networks (CNNs). Firstly, we will revisit classification of handwritten digits. Then we will move on to a slightly more complicated classification task in order to show that CNNs can outperform dense neural networks (DNNs). Lastly, we will look at time series forecasting.

The exercises in this homework is formulated quite loosely. That means you have a lot of freedom to explorer various methods. As long as you argue for your choices we strongly encourage you to experiment!

Due to potential long training times, it is okay not to perform long hyperparameter tuning sessions. The most important thing is to show that you understand what you are doing and can achieve slight improvements.

Since we are now working with more complicated datasets, pre-processing and/or data transformations plays a bigger role than in previous homeworks. Remember to take this into consideration and discuss your choices on this matter. See https://pytorch.org/tutorials/beginner/basics/data_tutorial.html for details on torch `datasets` and `dataloaders` if you find it necessary.

Your report will also be judged based general best practices in machine learning. So remember everything you've learned from the past weeks!

Ex. 1 (Classification of handwritten digits)

As in Homework 1, we aim to classify handwritten digits based on the MNIST dataset. In homework 1, SLP, logistic regression and dense neural networks were used. The goal of this exercise is to show that convolutional neural networks can perform better than the previously used methods.

You are allowed to reuse as much code as you want from Homework 1.

A) Setup and train a convolutional neural network for the above mentioned classification problem. Discuss considerations and choices you make along the way (pooling, stride, kernel size, etc.).

B) Perform hyperparameter tuning of your choice. Choose which hyperparameters you want to optimize and choose which hyperparameter optimization method to use. Discuss your choices and performance improvement.

C) Compare results with the results from Homework 1. If you get improved performance discuss why. Otherwise, discuss why you don't get improved performance.

Ex. 2 (Classification of RGB images)

We now focus on a slightly more complicated dataset: CIFAR-10. A brief description of the dataset can be found here: https://www.cs.toronto.edu/~kriz/cifar.html.

Redo Ex. 1 but with the CIFAR-10 dataset. You only have to discuss the changes you make compared to Ex. 1. Furthermore, you must also implement a dense neural network in order to compare performance.

Is the increase in performance greater than for the MNIST dataset? If so, why?

The dataset can be downloaded with the following code:

```
from torchvision.datasets import CIFAR10
from torchvision.transforms import ToTensor

dataset = CIFAR10(root='data/', download=True, transform=ToTensor())
test_dataset = CIFAR10(root='data/', train=False, transform=ToTensor())
```

Data transformation can be applied with the `transform` input. Replace `ToTensor()` with a data transformation pipeline.

Ex. 3 (Forecasting of stock prices)

The aim of this exercise is to predict stock prices using a causal convolutional neural network.

As above, perform hyperparameter tuning and discuss your choices and results throughout the exercise.

Note: Stock prices are inherently difficult to predict and the price history is not the only features that influence the price. Therefore, don't expect as good results as in previous exercises. If this was easy and straightforward, many more people would be rich!

To download the stock data, use the code below.

A) Choose a stock, e.g. Apple (ticker=AAPL) or Microsoft (ticker=MSFT). Train on stock closing prices from 1990-12-01 to 2018-12-31 and test your network on closing prices from 2019-01-01 to 2019-04-01. Your CCNN should predict one day ahead.

   Try with varying sizes of the receptive field and compare results.

B) Using the same data, make multistep predictions for the entire test period. That is, you can only use data from the training period and should apply the CCNN repeatedly on your predictions.

   How is the long-term prediction accuracy?

C) Add more stocks to the training data and use that to predict the same stock price as above. E.g. use Apple, Microsoft, IBM, etc. to predict the price of Apple stock price. You can choose how many and which stocks to add.

   Discuss how you incorporate the extra information in your CCNN.

   Does the extra stock prices add any predictive power? Does adding 3 or 4 stocks result in significantly better predictions than adding 1?

```
!pip install yfinance

import yfinance as yf
```

```
import numpy as np

def get_stock_data(ticker, start_date, end_date, receptive_field):

    # Download data as a pandas dataframe
    # Note: we only need the closing price!
    stock_data_df = yf.download(ticker,  start = start_date, end = end_date)
    stock_data_df = stock_data_df['Close']

    # Get number of data points
    stock_data_len = stock_data_df['Close'].count()

    # Impute values for NaN entries
    stock_data_df = stock_data_df.fillna(method='ffill')

    # Get data as numpy array
    data = stock_data_df.iloc[:, 0:1].values

    # Prepare data according to the receptive field
    features = []
    labels = []
    for i in range(receptive_field, stock_data_len):
        features.append(data[i-receptive_field:i, 0])
        labels.append(data[i, 0:1])

    # Transform data to numpy array
    features = np.array(features)
    labels = np.array(labels)

    # Reashape data into the right shape for Conv1d
    features = np.reshape(features, (features.shape[0], 1, features.shape[1]))

    return features, labels

# Ticker for the stock to be predicted
ticker = 'AAPL'

# Set start and end date
start_date = '1990-12-01'
end_date = '2018-12-31'

features, labels = get_stock_data(ticker,
                                  start_date,
                                  end_date,
                                  receptive_field=60)
```