19COC102 – AY: 2019-20 – Semester 2

Advanced Artificial Intelligence Systems - Coursework

Deep learning pipeline with Jupyter Notebook

Date set: 3rd February 2020

Due date: 16th March 2020

Method of submission: electronic via Learn.

Weight: 30% of the module assessment.

Formats of submission: Report in PDF (from Latex source) of 5 (five) pages. Zip archive with executed Jupiter Notebook.

IMPORTANT NOTE ON PLAGIARISM/COLLUSION

The university uses an automated software that compares submitted coursework with:

1) All coursework submissions by all students (e.g. it finds if two or more courseworks are too similar)

2) Other materials online including webpages, tutorials, scientific papers, and more.

After submission, the module leader receives a report with similarity percentages and automated and precise identification of similar parts. Some similarity is expected, e.g., in code where import of libraries are required and predefined functions. However, if similarities are found in the original implementation, structure of the code, the module leader is obliged to file a **case of suspected academic misconduct**. The university will then issue a letter inviting the student to provide a defence in front of an academic panel. If found guilty of academic misconduct, actions may vary from an official warning to the reduction of the mark to 1%, and in the worst case to the termination of studies.

How to avoid plagiarism/collusion:

1) Do not share your code or any part of it

2) Do not share any part of your report

3) Do not ask to see someone's else code, even if you are late or struggling to complete. Consider emailing the module leader or consider a mitigating circumstance claim if prevented to progress by external factors or illness.

4) Do not work closely together ending up writing the same code line by line

5) Use citations correctly, both in the report and in the code when copying and pasting lines from an external source.

EXAMPLE:
Case of potential plagiarism:

```
# MNIST dataset
train_dataset = torchvision.datasets.MNIST(root='../../data',
                                           train=True,
                                           transform=transforms.ToTensor(),
                                           download=True)

test_dataset = torchvision.datasets.MNIST(root='../../data',
                                          train=False,
                                          transform=transforms.ToTensor())

# Data loader
train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                           batch_size=batch_size,
                                           shuffle=True)

test_loader = torch.utils.data.DataLoader(dataset=test_dataset,
                                          batch_size=batch_size,
                                          shuffle=False)
```

Correct way to do it:

```
# MNIST dataset
# download and dataloader code as from source [1]
train_dataset = torchvision.datasets.MNIST(root='../../data',
                                           train=True,
                                           transform=transforms.ToTensor(),
                                           download=True)

test_dataset = torchvision.datasets.MNIST(root='../../data',
                                          train=False,
                                          transform=transforms.ToTensor())

# Data loader
train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                           batch_size=batch_size,
                                           shuffle=True)

test_loader = torch.utils.data.DataLoader(dataset=test_dataset,
                                          batch_size=batch_size,
                                          shuffle=False)
### end of source [1]

#source [1] https://github.com/yunjey/pytorch-tutorial/blob/master/tutorials/01-
basics/feedforward_neural_network/main.py#L18-L36
```

Note 1: We know that code can often look very similar if not identical for certain parts, e.g. the dataLoader or the specification of a network architecture. Therefore, if you cite the sources correctly, use comments, and you do not copy the from other course mates, you are not at risk.

Note 2: While some similarity is expected in the first blocks of the Jupyter Notebook, we do not expect significant similarities in the presentation of the results and in particular in the report.

Overview

This coursework is designed to help you build proficiency in coding and visualising results for machine learning and AI systems using the latest technology. It will provide you with hands-on experience with the entire pipeline of training deep neural networks.

Main objectives

- Create a python code in Jupyter notebook to perform training of at least two deep neural network architectures on a simple classification task.
- Use a deep learning framework such as Pytorch or Tensorflow.
- Describe the structure of the code and illustrate the results of training with graphs and analysis and draw conclusions in a written report

Motivations

Deep learning tools, datasets and algorithms are rapidly developing, offering increasingly great opportunities to implement and manage complex machine learning projects. Such a moving technological landscape also implies that the knowledge to use such tools is not static, and requires continuous updates, reading documentation, and self-learning abilities to re-use and understand new procedures. Therefore, one of the aims of this coursework (as part of an advanced AI part C module) is also to test your capability of conducting independent research online, finding tutorials and examples, and using them with your computer science background to demonstrate knowledge, implementation capabilities, and presentation of results according to your understanding of the problem.

Detailed description

1) Suggested preparatory steps

    a) Get familiar with Jupyter notebook and the system requirements to run it.

    b) If you want to run Jupyter notebook locally, you need to use the Anaconda environment to create a Python environment (we recommend 3.6) and install required packages you want to use in your code.

    c) If you use Google Colab, Anaconda is not required.

    d) Get familiar with useful libraries such as numpy and matplotlib.

    e) Learn how to download/use a dataset, e.g. MNIST or CIFAR-10 with a DataLoader. If you choose any other dataset, specify the source and describe it in the text.

    f) Get familiar with Pytorch/Tensorflow foundations, e.g. how to use tensors and how to build a deep neural network.

2) Core assignment

    a) Experiment training with convolutional networks in a Notebook script to classify your dataset: **use at least two (2) different neural architectures and two (2) different configuration parameters for each**.

b) Note that while there are available pretrained models, you are supposed to write your own architecture and train it. There are plenty of examples online of different architectures: choose a few and implement them in your code. For example, you can choose a single fully connected layer and compare it with one or more complex CNN architectures with different number of layers, different filter sizes, etc.

c) The comparison of results may include loss graphs vs epochs with training, validation and test errors, considerations on overfitting, training speed in relation to learning rates and different optimisers during epochs, etc.

3) Reporting. Present your setups and the results of your experiments within clear subsections, clear explanations, graphs and tables if appropriate, as detailed in the next point.

4) Submit the following:

a) Report part A: A description of the tools you used and the methods, including sources (correctly cited) and a comprehensive description of the changes you made (2 pages).

b) Report part B: A critical report of the results, graphs and tables, and what works well, what works less well and why (3 pages).

c) Submit a Jupyter Notebook with your experiments (executed, showing outputs and graphs). Submit the file as a ZIP archive.

d) Formatting instruction: use the Latex template at https://www.overleaf.com/read/gkphfttddwfc . The page limits are strict and should hold all the content, i.e. no extra pages for titles, figures or references will be accepted. Only upload a PDF version of your report. Also do not adjust fonts or margins from the template. Remember to write your name and ID at the start of your report.

In your report, give more emphasis to your original contribution in the implementation, which is how you build different architectures, the parameters and methods you use for the training, and how you conduct the evaluation and comparison.

Marking scheme

20% for the report part A: this is assessed for completeness, the clarity of the explanations, correct citations of sources, reproducibility of the approach (how well someone can read and reproduce your work).
55% for the report part B: this is assessed for clarity of results in relation to your described method, clarity of graphs and tables, soundness of the methods and insight derived from the comparison. The performance achieved (i.e. the accuracy of your model) will not be used as a marking criterion.
25% for the Jupyter Notebook: this is assessed for sufficient commenting, division of code in clear blocks representing each step of the algorithm, clarity of the implementation and plots

Questions and answers

Q: Can you provide examples of what are different architectures?

A: There are many examples of architectures online. Even a simple one or two layer fully connected architecture can be used and compared to more complex, e.g., convolutional, architectures.

Q: Can you provide examples of what are different parameter settings?

A: Parameter settings might include the learning rate, the type of optimiser, e.g RMSProp, ADAM, the number of epochs, the size of layers in the neural architecture, etc. The important aspect here is to show that you can test different such parameter settings and draw conclusions on what works best.

Q: Will the performance in the training be used as a marking criterion?

A: No. The soundness of the work will be assessed in relation to whether your deep learning pipeline is correct and regardless of the reported performance. Note: it is expected that your code is bug free, and bugs will be penalised. Also if performance of a given approach was weaker than you expected, explanation is required.