Machine Learning Practical 2019/20: Coursework 1

Released: Monday 14 October 2019

Submission due: 16:00 Friday 25 October 2019

1 Introduction

The aim of this coursework is to explore the classification of images of handwritten digits using neural networks. The first part of the coursework will concern the implementation of different activation functions (discussed in lecture 4) in the MLP framework. The second part will involve exploring different multi-layer neural network architectures.

The coursework will use an extended version of the MNIST database, the EMNIST Balanced dataset, described in Section 2. Section 3 describes the additional code provided for the coursework (in branch `mlp2019-20/coursework_1` of the MLP github), and Section 4 describes how the coursework is structured into four tasks. The main deliverable of this coursework is a report, discussed in section 5, using a template that is available on the github. Section 6 discusses the details of carrying out and submitting the coursework, and the marking scheme is discussed in Section 7.

You will need to submit your completed report as a PDF file and your local version of the `mlp` code including any changes you made to the provided (`.py` files). The detailed submission instructions are given in Section 6.2 – please follow these instructions carefully.

2 EMNIST dataset

In this coursework we shall use the EMNIST (Extended MNIST) Balanced dataset [Cohen et al., 2017], https://www.nist.gov/itl/iad/image-group/emnist-dataset. EMNIST extends MNIST by including images of handwritten letters (upper and lower case) as well as handwritten digits. Both EMNIST and MNIST are extracted from the same underlying dataset, referred to as NIST Special Database 19. Both use the same conversion process resulting in centred images of dimension 28×28.

There are 62 potential classes for EMNIST (10 digits, 26 lower case letters, and 26 upper case letters). However, we shall use a reduced label set of 47 different labels. This is because (following the data conversion process) there are 15 letters for which it is confusing to discriminate between upper-case and lower-case versions. In the 47 label set, upper- and lower-case labels are merged for the following letters:
`C, I, J, K, L, M, O, P, S, U, V, W, X, Y, Z`.

The training set for Balanced EMNIST has about twice the number of examples as the MNIST training set, thus you should expect the run-time of your experiments to be about twice as long. The expected accuracy rates are lower for EMNIST than for MNIST (as EMNIST has more classes, and more confusable examples), and differences in accuracy between different systems should be larger. Cohen et al. [2017] present some baseline results for EMNIST.

You do *not* need to directly download the EMNIST database from the nist.gov website, as it is part of the `coursework_1` branch in the `mlpractical` Github repository, discussed in Section 3 below.

3 Github branch `mlp2019-20/coursework_1`

You should run all of the experiments for the coursework inside the Conda environment you set up for the labs. The code for the coursework is available on the course Github repository on a branch `mlp2019-20/coursework_1`. To create a local working copy of this branch in your local repository you need to do the following.

1. Make sure all modified files on the branch you are currently have been committed (see notes/getting-started-in-a-lab.md if you are unsure how to do this).
2. Fetch changes to the upstream `origin` repository by running
   `git fetch origin`
3. Checkout a new local branch from the fetched branch using
   `git checkout -b coursework_1 origin/mlp2019-20/coursework_1`

You will now have a new branch in your local repository with all the code necessary for the coursework in it.

This branch includes the following additions to your setup:

- A new `EMNISTDataProvider` class in the `mlp.data_providers` module. This class makes some changes to the `MNISTDataProvider` class, linking to the `EMNIST Balanced` data, and setting the number of classes to 47.
- Training, validation, and test sets for the `EMNIST Balanced` dataset that you will use in this coursework.
- Four new classes in the `mlp.layers` module:
  `LeakyReLULayer, ParametricReLULayer, RandomReLULayer, ELULayer`.
  These are skeleton classes that you will need fully implement as part of the coursework (see Sections 4.1).
- One Jupyter notebook:
  `ActivationFunction_tests.ipynb`
  to be used for testing the implementations of`LeakyReLULayer, ParametricReLULayer, RandomReLULayer` and `ELULayer` (Sections 4.1). The tests serve as a safeguard to prevent experimentation with faulty code which might lead to wrong conclusions. Tests in general are a *vital* ingredient for good software development, and especially important for building correct and efficient deep learning systems.
- A directory called `report` which contains the LaTeX template and style files for your report. You should copy all these files into the directory which will contain your report.

4 Tasks

The coursework is structured into 4 tasks, all supported by experiments on `EMNIST`

1. Implementation of the activation functions Leaky ReLU, Parametric ReLU, Random ReLU and ELU.
2. Setting up a **baseline** system on `EMNIST`.
3. Exploration of the implemented activation functions.
4. A research investigation into whether non-linearities are necessary for fitting deep networks.

4.1 Part 1: Implementing Activation Functions

In the first part of the assignment you will implement three further activation functions, each of which is related to ReLU. Each of these units defines an activation function for which $f(x) = x$ when $x > 0$, as for ReLU, but avoid having a zero gradient when $x < 0$.

Leaky ReLU, Random ReLU and Parametric ReLU all share the same form of:

$$f(x) = \left\{ \begin{array}{cc} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{array} \right\} \tag{1}$$

However, they differ in how the α variable is selected.

- In **Leaky ReLU** α is a single manually selected scalar value, usually in the range of $0.001 - 0.2$.
- In **Random ReLU** α is a vector of uniformly drawn scalar coefficients, usually ranging from $0.001 - 0.2$, each unit to be activated has its own unique α. The α vector is uniformly drawn at every single forward propagation phase.
- In **Parametric ReLU** the α is a single learnable parameter usually initialized uniformly in the range $0.001 - 0.2$ and learned using backpropagation.
- **ELU** Clevert et al. [2015] has the following form:

$$f(x) = \left\{ \begin{array}{cc} x & \text{if } x >= 0 \\ a(e^x - 1) & \text{otherwise} \end{array} \right\} \tag{2}$$

  where α is a single manually selected positive scalar value.

Here are the steps to follow:

1. Implement each of these activations function as classes `ParametricReLULayer`, `LeakyReLULayer`, `RandomReLULayer` and `ELULayer`. You need to implement `fprop` and `bprop` methods for each class.
2. Verify the correctness of your implementation using the supplied unit tests in `Activation_Tests.ipynb`.
3. Automatically create test outputs `sxxxxxxx_activation_test_pack.npy`, by running the provided program `scripts/generate_activation_layer_test_outputs.py` which uses your code for the previously mentioned layers to run your fprop and bprop and where necessary the `grads_wrt_params` methods for each layer on a unique test vector generated using your student ID number. To do this part simply go to the scripts folder `scripts/` and then run `python generate_activation_layer_test_outputs.py --student_id sxxxxxxx` replacing the student id with yours. A file called `sxxxxxxx_activation_test_pack.npz` will be generated under data which you need to submit with your report.

For Part 1 of the coursework you need to submit the test files `sxxxxxxx_activation_test_pack.npz` (where `sXXXXXXX` is replaced with your student number) created in step 3 above.

In your report you should thoroughly define and analyse each activation function.

(30 Marks)

4.2 Baseline

Initially you need to establish a baseline system on `EMNIST` using stochastic gradient descent (SGD) and without explicit regularization. Carry out experiments using 32, 64 and 128 ReLU hidden units per layer, and investigate using from 1-3 hidden layers. Make sure you use an appropriate learning rate. Train for 100 epochs. For the initial experiments, compare systems using the test set accuracy. For model selection, you are free to use early stopping or the best validation model.

(10 Marks)

4.3 Part 2: EMNIST Experiments

For the experiments going forward you should use a standard architecture to compare the different algorithms that you explore: it is recommended that you use 3 hidden layers and 128 hidden units.

In Part 2 of the coursework you will experiment with `ParametricReLULayer`, `LeakyReLULayer`, `RandomReLULayer` and `ELULayer` in multi-layer networks trained on EMNIST.

Part 2A: Comparing activation functions. You should modify your baseline network to one that uses one of each of the above activation functions and train a model for Your main aim is to compare the generalization performance of each of these activation functions as well as discover the best possible network configuration, when the only available options to choose from are the activation functions and their hyper-parameters. Ensure that you thoroughly describe the relationships between each of the activation functions in your report, ideally both at the theoretical and empirical level. **Note that the expected amount of work in this part is not a brute-force exploration of all possible variations of network configurations and hyperparameters but a carefully designed set of experiments that provides meaningful analysis and insights.**

(40 Marks)

Part 2B: Linear vs Non-Linear activations. In this section you will need to set up an experiment to explore whether non-linear activation functions are necessary for fitting deep neural networks on large datasets. A linear activation function is effectively the output of a fully connected layer without an activation applied to it. Design, implement and run an experiment that can showcase whether non-linear activation functions are helpful and/or necessary for successful training. Furthermore, provide theoretical and practical intuitions as to why this is the case.

(20 Marks)

5 Report

Your coursework will be primarily assessed based on your submitted report.

The directory `coursework_1/report` contains a template for your report (`mlp-cw1-template.tex`); the generated pdf file (`mlp-cw1-template.pdf`) is also provided, and you should read this file carefully as it contains some useful information about the required structure and content. The template is written in LaTeX, and we strongly recommend that you write your own report using LaTeX, using the supplied document style `mlp2019` (as in the template).

You should copy the files in the `report` directory to the directory containing the LaTeX file of your report, as `pdflatex` will need to access these files when building the pdf document from the LaTeX source file.

Your report should be in a 2-column format, based on the document format used for the ICML conference. The report should be a **maximum of 5 pages long**, not including references. We will not read or assess any parts of the report beyond this limit.

Ideally, all figures should be included in your report file as vector graphics files rather than raster files as this will make sure all detail in the plot is visible. Matplotlib supports saving high quality figures in a wide range of common image formats using the `savefig` function. **You should use `savefig` rather than copying the screen-resolution raster images outputted in the notebook.** An example of using `savefig` to save a figure as a PDF file (which can be included as graphics in LaTeX compiled with `pdflatex` is given below.

```
import matplotlib.pyplot as plt
import numpy as np
# Generate some example data to plot
x = np.linspace(0., 1., 100)
y1 = np.sin(2. * np.pi * x)
y2 = np.cos(2. * np.pi * x)
fig_size = (6, 3)  # Set figure size in inches (width, height)
fig = plt.figure(figsize=fig_size)  # Create a new figure object
ax = fig.add_subplot(1, 1, 1)  # Add a single axes to the figure
# Plot lines giving each a label for the legend and setting line width to 2
ax.plot(x, y1, linewidth=2, label='$y = \sin(2\pi x)$')
ax.plot(x, y2, linewidth=2, label='$y = \cos(2\pi x)$')
# Set the axes labels. Can use LaTeX in labels within $...$ delimiters.
ax.set_xlabel('$x$', fontsize=12)
ax.set_ylabel('$y$', fontsize=12)
ax.grid('on')  # Turn axes grid on
ax.legend(loc='best', fontsize=11)  # Add a legend
fig.tight_layout()  # This minimises whitespace around the axes.
fig.savefig('file-name.pdf') # Save figure to current directory in PDF format
```

If you make use of any any books, articles, web pages or other resources you should appropriately cite these in your report. You do not need to cite material from the course lecture slides or lab notebooks.

To create a pdf file `mlp-cw1-template.pdf` from a LaTeX source file (`mlp-cw1-template.tex`), you can run the following in a terminal:

```
pdflatex mlp-cw1-template
bibtex mlp-cw1-template
pdflatex mlp-cw1-template
pdflatex mlp-cw1-template
```

(Yes, you have to run pdflatex multiple times, in order for latex to construct the internal document references.)

An alternative, simpler approach uses the `latexmk` program:

```
latexmk -pdf mlp-cw1-template
```

Another alternative is to use an online LaTeX authoring environment such as https://overleaf.com – note that all staff and students have free access to Overleaf Pro - see https://www.ed.ac.uk/information-services/computing/desktop-personal/software/main-software-deals/other-software/overleaf.

It is worth learning how to use LaTeX effectively, as it is particularly powerful for mathematical and academic writing. There are many tutorials on the web.

6 Mechanics

Marks: This assignment will be assessed out of 100 marks and forms 10% of your final grade for the course.

Academic conduct: Assessed work is subject to University regulations on academic conduct:
http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

Submission: You can submit more than once up until the submission deadline. All submissions are timestamped automatically. Identically named files will overwrite earlier submitted versions, so we will mark the latest submission that comes in before the deadline.

If you submit anything before the deadline, you may not resubmit after the deadline. (This policy allows us to begin marking submissions immediately after the deadline, without having to worry that some may need to be re-marked).

If you do not submit anything before the deadline, you may submit *exactly once* after the deadline, and a late penalty will be applied to this submission unless you have received an approved extension. Please be aware that late submissions may receive lower priority for marking, and marks may not be returned within the same timeframe as for on-time submissions.

Warning: Unfortunately the `submit` command will technically allow you to submit late even if you submitted before the deadline (i.e. it does not enforce the above policy). Don't do this! We will mark the version that we retrieve just after the deadline.

Extension requests: For additional information about late penalties and extension requests, see the School web page below. **Do not email any course staff directly about extension requests**; you must follow the instructions on the web page.

http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/late-coursework-extension-requests

Late submission penalty: Following the University guidelines, late coursework submitted without an authorised extension will be recorded as late and the following penalties will apply: 5 percentage points will be deducted for every calendar day or part thereof it is late, up to a maximum of 7 calendar days. After this time a mark of zero will be recorded.

6.1 Backing up your work

It is **strongly recommended** you use some method for backing up your work. Those working in their AFS homespace on DICE will have their work automatically backed up as part of the routine backup of all user homespaces. If you are working on a personal computer you should have your own backup method in place (e.g. saving additional copies to an external drive, syncing to a cloud service or pushing commits to your local Git repository to a private repository on Github). **Loss of work through failure to back up does not constitute a good reason for late submission**.

You may *additionally* wish to keep your coursework under version control in your local Git repository on the `coursework_1` branch.

If you make regular commits of your work on the coursework this will allow you to better keep track of the changes you have made and if necessary revert to previous versions of files and/or restore accidentally deleted work. This is not however required and you should note that keeping your work under version control is a distinct issue from backing up to guard against hard drive failure. If you are working on a personal computer you should still keep an additional back up of your work as described above.

6.2 Submission

Your coursework submission should be done electronically using the `submit` command available on DICE machines.

Your submission should include

- Your test outputs `sxxxxxxx_activation_test_pack.npz`. Which can be generated by implementing the previously mentioned classes, going into `scripts/` and running `python generate_activation_layer_test_outputs.py --student_id sxxxxxxx` replacing the student id with yours. A file called `sxxxxxxx_activation_test_pack.npz` will be generated under data which you need to submit with your report.
- your completed report as a PDF file, using the provided template
- your local version of the `mlp` code including any changes you made to the modules (`.py` files)

Please do not submit anything else (e.g. log files).

You should copy all of the files to a single directory, `coursework1`, e.g.

```
mkdir coursework1
cp reports/coursework1.pdf coursework1
```

and then submit this directory using

```
submit mlp cw1 coursework1
```

Please submit the directory, not a zip file, not a tar file.

The `submit` command will prompt you with the details of the submission including the name of the files / directories you are submitting and the name of the course and exercise you are submitting for and ask you to check if these details are correct. You should check these carefully and reply `y` to submit if you are sure the files are correct and `n` otherwise.

You can amend an existing submission by rerunning the `submit` command any time up to the deadline. It is therefore a good idea (particularly if this is your first time using the DICE submit mechanism) to do an initial run of the `submit` command early on and then rerun the command if you make any further updates to your submission rather than leaving submission to the last minute.

7 Marking Guidelines

This document (Section 4 in particular) and the template report (`mlp-cw1-template.pdf`) provide a description of what you are expected to do in this assignment, and how the report should be written and structured.

Assignments will be marked using the scale defined by the **University Common Marking Scheme**:

Numeric mark	Equivalent letter grade	Approximate meaning
< 40	F	fail
40-49	D	poor
50-59	C	acceptable
60-69	B	good
70-79	A3	very good/distinction
80-100	A1, A2	excellent/outstanding/high distinction

Please note the University specifications for marks above 70:

A1 90-100 Often faultless. The work is well beyond what is expected for the level of study.

A2 80-89 A truly professional piece of scholarship, often with an absence of errors.
As 'A3' but shows (depending upon the item of assessment): significant personal insight / creativity / originality and / or extra depth and academic maturity in the elements of assessment.

A3 70-79
Knowledge: Comprehensive range of up-to-date material handled in a professional way.
Understanding/handling of key concepts: Shows a command of the subject and current theory.
Focus on the subject: Clear and analytical; fully explores the subject.
Critical analysis and discussion: Shows evidence of serious thought in critically evaluating and integrating the evidenced and ideas. Deals confidently with the complexities and subtleties of the arguments. Shows elements of personal insight / creativity / originality.
Structure: Clear and coherent showing logical, ordered thought.
Presentation: Clear and professional with few, relatively minor flaws. Accurate referencing. Figures and tables well constructed and accurate. Good standard of spelling and grammar.

And finally... this assignment is worth 10% of the total marks for the course, and the next assignment is worth 40%. This is not because the second assignment is four times bigger or harder than this one (although it will be more challenging). The reason that this assignment is worth 10% is so that people get an opportunity to learn from their errors in doing the assignment, without it having a very big impact on their overall grade for the module.

References

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017. URL https://arxiv.org/abs/1702.05373.