Machine Learning Practical 2020/21: Coursework 1

Released: Monday 19 October 2020

Submission due: 16:00 Friday 30 October 2020

1 Introduction

The aim of this coursework is to explore the classification of images of handwritten digits using neural networks. The first part of this coursework will concern the identification and discussion of a *fundamental problem* in machine learning, as shown in Figure 1. Following this preliminary discussion, you will further investigate this problem in *wider* and *deeper* neural networks, study it in terms of network width and depth. The second part involves implementing different methods to combat the problem identified in Task 1 and then comparing these methods empirically and theoretically. In the final part, you will briefly discuss the main strengths and weakness of any one related work to the methods examined in Task 2.

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 `mlp2020-21/coursework_1` of the MLP github), and Section 4 describes how the coursework is structured into three tasks. The main deliverable of this coursework is a report, discussed in section 8, using a template that is available on the github. Section 9 discusses the details of carrying out and submitting the coursework, and the marking scheme is discussed in Section 10.

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 9.2 – 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 `mlp2020-21/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 `mlp2020-21/coursework_1`. To create a local working copy of this branch in your local repository you need to do the following.

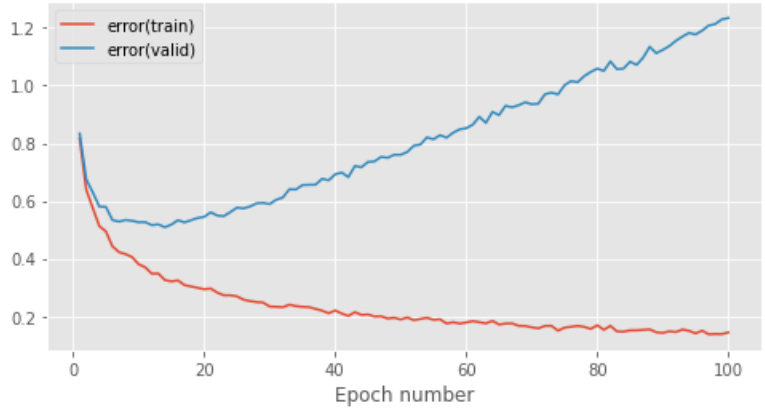
1. Make sure all modified files on the branch you are currently have been committed (see notes/getting-started-in-a-lab.md if you are unsure how to do this).
2. Fetch changes to the upstream `origin` repository by running
   `git fetch origin`
3. Checkout a new local branch from the fetched branch using
   `git checkout -b coursework_1 origin/mlp2020-21/coursework_1`

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

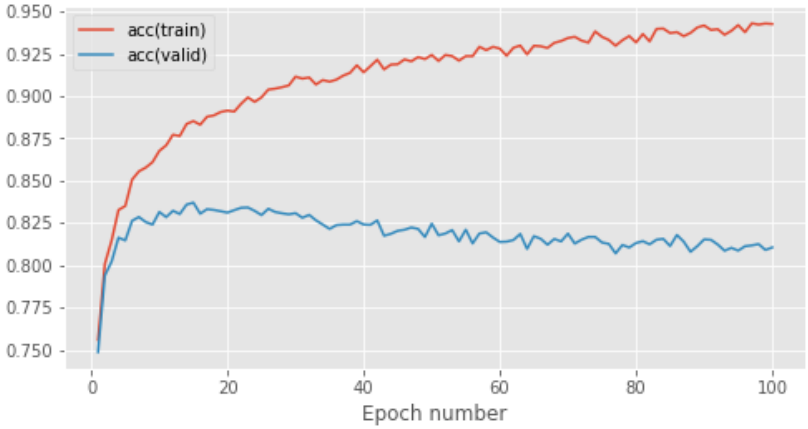
This branch includes the following additions to your setup:

- 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.
- In order to further improve performance and mitigate the problem identified in neural networks, you will also need to implement a new class in the `mlp.layers` module:
  `DropoutLayer`
  and also two weight penalty tecniques in the `mlp.penalties` module:
  `L1Penalty and L2Penalty`.
- `DropoutandPenalty_tests.ipynb` Jupyter notebook
  to be used for testing the implementations of `DropoutLayer`, `L1Penalty and L2Penalty` classes. 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.

  Please note that passing these preliminary tests *does not* necessarily mean your classes are absolutely bug-free. If you get unexpected curves during model training, re-check your implementation of the classes.
- 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.



(a) Error curve on the training and validation set of EMNIST dataset.



(b) Accuracy curve on the training and validation set of EMNIST dataset.

Figure 1: Error and Accuracy curves for a baseline model on EMNIST Dataset.

4 Tasks

The coursework is structured into 3 tasks, the first two are supported by experiments on `EMNIST` dataset.

1. Identification of a fundamental problem in machine learning as shown in Fig 1 and setting up a **baseline** system on `EMNIST` by a valid hyper-parameter search.
2. A research investigation and analysis into whether using Dropout and/or Weight Penalty (`L1Penalty` and `L2Penalty`) addresses the problem found in training machine learning models (Fig 1). How do these two approaches improve/degrade the model's performance?
3. A brief literature review of any one work as discussed in Section 7.

5 Task 1: Problem identification

Figure 1 shows the training and validation error curves in Figure 1a and also training and validation accuracies in Figure 1b for a model with 1 hidden layer[1] and a ReLU activation function trained on the EMNIST dataset by using cross-entropy error function. This curve can be re-produced by running the model settings defined in the `Coursework1.ipynb` notebook in the github repository. First identify and discuss the problem shown by the curves in Figure 1 and briefly discuss potential solutions in this section for overcoming this problem.

[1] *A hidden layer is AffineLayer+ReLULayer together.*

Varying number of hidden units. Initially you will train various 1-hidden layer networks by using either 32, 64 and 128 ReLU hidden units per layer on `EMNIST` using stochastic gradient descent (SGD) without any regularization. Make sure you use an appropriate learning rate and train each network for 100 epochs. Visualise and discuss how increasing number of hidden units affects the validation performance and whether it worsens or mitigates the problem.

(10 Marks)

Varying number of layers. Here you will train various neural networks by using either 1, 2, 3 hidden layers with 128 ReLU hidden units per layer on `EMNIST` using stochastic gradient descent (SGD) without any regularization. Make sure you use an appropriate learning rate and train each network for 100 epochs. Visualise and discuss how increasing number of layers affects the validation performance and whether it worsens or mitigates the problem.

(10 Marks)

6 Task 2: Mitigating the problem with Dropout and Weight Penalty

Definition and Motivation. Here you will explain

- `Dropout Layer`, `L1Penalty` and `L2Penalty` including their formulations and implementation details (do not copy/paste your code here),
- how/why/to what extent each one can alleviate the problem above,
- how they differ from each other in theory.

These explanations must be **in your own words**.

(10 Marks)

Implementing Dropout and Weight Penalty. Here you will implement `DropoutLayer`, `L1Penalty` and `L2Penalty` and test their correctness. Here are the steps to follow:

1. Implement the Dropout class in the `DropoutLayer` of the `mlp.layers` module. You need to implement `fprop` and `bprop` methods for this class. Please note that the sample distribution to be used for Dropout implementation is a uniform distribution, U(0,1) to pass the unit tests.
2. Implement the L1Penalty and L2Penalty class in the `L1Penalty` and `L2Penalty` of the `mlp.penalties` module. You need to implement `__call__` and `grad` methods for this class. After defining these functions, they can be provided as a parameter, `weights_penalty, biases_penalty` in the `AffineLayer` class while creating the multi-layer neural network.
3. Verify the correctness of your implementation using the supplied unit tests in `DropoutandPenalty_tests.ipynb`
4. Automatically create test outputs `xxxxxxx_regularization_test_pack.npy`, by running the provided program `scripts/generate_regularization_layer_test_outputs.py` which uses your code for the previously mentioned layers to run your fprop, bprop, `__call__` and `grad` methods where necessary 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_regularization_layer_test_outputs.py --student_id sxxxxxxx` replacing the student id with yours. A file called `xxxxxxx_regularization_test_pack.npy` will be generated under data which you need to submit with your report.

(20 Marks)

EMNIST Experiments. In this section you should modify your baseline network to one that uses one or a combination of `DropoutLayer` with either `L1Penalty` or `L2Penalty` and train a model. For the experiments, your baseline network should contain 3 hidden layers and 128 hidden units with ReLU activation function. Your main aim is to i) investigate whether/how each of these functions addresses the above mentioned problem, ii) study the generalization performance of your network when used with one of these functions or a combination of them, iii) discover the best possible network configuration, when the only available options to choose from are Dropout and Weight Penalty functions and the hyper-parameters (learning rate, Dropout Probability and `penalty coefficient` for the Weight Penalty functions).

The Dropout probability is a float value in the range (0,1), *e.g.* 0.5, chosen manually. Penalty coefficient is also a manually selected float value, *e.g.* 0.001, usually in the range of $0.1 - 0.00001$. **For model selection, you should use validation performance to pick the best model and finally report test performance of the best model.**

Ensure that you thoroughly describe how these functions affect performance when used together and separately with different hyperparameters 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)

7 Task 3: Literature Review

In this section, you will explore one related work in the research area of regularization methods. You should discuss the summary of the paper, strengths and limitations of the research work in less than **500 words**. Note that this review must be **in your own words**.

Below is a list of papers that you are recommended to consider for selecting one paper. If for any particular reason, you do not wish to choose from the given list, you are free to discuss any other paper which relates this coursework and is a published work in AI/ML/CV/NLP conferences such as ICML, NeurIPS, ICLR, AAAI, IJCAI, CVPR, ECCV, ICCV, EMNLP or ACL. This is not an exhaustive list of conferences but largely covers most of them.

- Dropout: a simple way to prevent neural networks from overfitting, Srivastava et al. [2014].
- Maxout Networks, Goodfellow et al. [2013].
- Understanding deep learning requires rethinking generalization, Zhang et al. [2016].

(10 Marks)

8 Report

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

The report template is divided into sections which corresponds to each task in the coursework specs, especially from Section 2 to Section 5 of the report. The Abstract, Introduction and Conclusion sections *will not be graded* for the final marks but we highly encourage you to attempt them in the report as we will provide feedback on them. Please note that these sections will be graded in the later courseworks, hence it is beneficial to take the feedback on coursework 1 into serious account.

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 `mlp2020` (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.

9 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 submission system on Learn 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.

9.1 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.

9.2 Submission

Your coursework submission should be done online on the Learn course webpage.

Your submission should include one zip file `sxxxxxxx.zip` that should contain

- Your test outputs `xxxxxxx_regularization_test_pack.npy`. which can be generated by implementing the previously mentioned classes, going into `scripts/` and running `python generate_regularization_layer_test_outputs.py --student_id sxxxxxxx` replacing the student id with yours. A file called `xxxxxxx_regularization_test_pack.npy` will be generated under data which you need to submit with your report and the code.
- your completed report as a PDF file renamed as `sxxxxxxx_report.pdf`, using the provided template
- your local version of the `mlp` code including any changes you made to the modules (`.py` files) and the `Coursework_1.ipynb` notebook.

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

You can use this command on Linux machines to zip all the files together -

```
zip sxxxxxxx.zip mlp/ Coursework_1.ipynb sxxxxxxx_report.pdf
xxxxxxx_regularization_test_pack.npy
```

Replace `sxxxxxxx` with your student id.

Once you have successfully created the .zip file, you need to login to your Learn `Machine Learning Practical (2020-2021)[YR]` webpage and submit the file.

- Migrate to the section **Coursework** on the left column on the course page.
- Click on Coursework 1.
- A page will appear where you will need to browse and upload your .zip file that you created previously in `Attach Files` and then click **Submit**.

You can amend an existing submission by attaching a different .zip file using the Attach Files option and then Submit again.

Note that we will only mark the last uploaded coursework in case you amend your files. Thus it is your responsibility to make sure that correct files are uploaded.

10 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

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.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR, 2013.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. 2016.