06-12412 Introduction to Neural Computation[*]
2017/2018 Coursework Description

Released November 15, 2017

1 Introduction

The module is assessed by 80% examination and 20% continuous assessment. This document specifies the continuous assessment component. The objective of this coursework is for you to gain practical experience in designing, implementing, training and optimizing a neural network to carry out a specific real-world task. This year the task is to recognise objects in images. The coursework is designed as a Kaggle competition where you will work together in groups, each group submitting a solution. The allocation of students to groups will be specified on Canvas.

All files must be submitted via Canvas by **4 pm on Friday 8th of December 2017**[1] by the group leader. If you miss this deadline, your mark will be reduced by 5% (out of 100%) for each School working day (or part thereof) your submission is late. Feedback with marks will be returned within three weeks of the hand-in deadline.

[*]Note: this coursework is for 06-12412 Introduction to Neural Computation only. Students who are taking 06-20416 Neural Computation are encouraged to form teams and participate in the competition, but their work will not be assessed.

[1]Due to School regulations about out of semester working times, we will not be able to set a deadline in January as earlier stated.

2 What you need to do

1. Create an account on `kaggle.com` and set up a team together with your group members. Inform us about your Kaggle account details by sending an email to `P.Nguyen@cs.bham.ac.uk` with the subject NC2017 containing your Kaggle user name, and the Kaggle team name.

2. Join the Kaggle competition using the following URL:

   `https://www.kaggle.com/t/f81bc60dd7e048dfa558a9f8de85dfaa`

3. Follow the instructions on the Kaggle website for instructions on how to download the training data set, format the submission files etc.

4. Design a suitable neural network that can do object recognition from the training data set. Hint: Backpropagation and some form of stochastic gradient descent will probably be most appropriate.

5. You should implement the neural network using Python, but you can use any library for neural networks (e.g. Tensorflow is installed on the lab machines in the School of Computer Science). You can implement everything from scratch, or use any other code you find on the internet as long as you properly reference the sources in your report and source code. Hint: Writing your own code often proves easier than figuring out how to get someone else's simulator to do what you need, and you will probably learn more, but there are limited credit/marks available for the implementation aspect of this exercise.

6. Experiment systematically with the neural network details that you think are most likely to improve the generalization performance you achieve. Hints: Consider how you are going to avoid under-fitting and over-fitting of the training data, and what parameters are going to have the biggest effect on that. It is usually better to aim to optimize two or three things well, rather than a large number of things not very well.

7. Regularly submit your results on the test data set to Kaggle to see how well your network performs compared with the others.

8. Write one group report explaining what you did and what you found. A reasonable length for the report would be between 3000 and 4000 words

excluding the reference list, plus as many diagrams, tables and graphs as you think appropriate. You should include the following sections:

- Introduction - Discuss the data sets involved, the machine learning task, and what you aimed to achieve. [5%]
- Design - Describe and justify the neural network you designed for the task, and the factors you decided to experiment with. [15%]
- Implementation - Describe how you implemented your neural network and the associated performance analysis mechanisms. Explain why you chose to do it that way. Remember to cite any sources you used. Include comments in the source code that explains how the code works. [20%]
- Experiments - Describe the experiments you carried out to optimise your network's generalization performance, and present the results you obtained. Explain in detail how you used the training and testing data sets. The results should be presented in a statistically rigorous manner. [45%]
- Conclusions - Summarize your key findings, including which factors proved most crucial, and what was the best generalization performance you achieved. [10%]
- Description of contribution - Describe each group members' contribution to the overall project. [5%]

3 Files to submit

The group leader should submit the following two files on Canvas, where `gn` should be replaced by the group number:

- `gn-coursework.ipynb`
  Source code and report in the form of an executable Jupyter notebook. The Jupyter notebook should have the section headings as described above.

- `gn-coursework.pdf`
  A PDF version of the Jupyter notebook file.

4 Assessment

The groups will be marked by their report and source code as indicated by the percentages above. If there is evidence of significant differences in contributions among group members, then individual marks may be adjusted.

Keep in mind that this coursework corresponds to only 20% of a 10 credit module. Spend an appropriate amount of time on it!

5 Help and Advice

If you get stuck, feel free to ask for advice during Per Kristian's office hours which are at 12-1 pm every Tuesday during Autumn Term in Room 207 in the School of Computer Science. You can also write questions and comments in the Discussion forum on Kaggle.

It can be challenging to find suitable time and place for all group members to meet. It is therefore highly recommended that you make use of online collaborative tools. In particular, you could set up a discussion channel for the project on Slack[2], and keep the code in a git repository, e.g. on the School of Computer Science's git server[3], or with some external provider such as Bitbucket or GitHub. You can also arrange virtual meetings via Google Hangout.

[2]`https://slack.com`
[3]`https://git-teaching.cs.bham.ac.uk/`