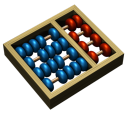

University of Campinas
Institute of Computing

Class Development Plan: MO810/MC959

Topics in Artificial Intelligence
(Introduction to Deep Learning)

Instructor: Joao Meidanis

Tuesdays and Thursdays, 7-9pm
First Semester, 2018

www.ic.unicamp.br/~meidanis/courses/mo810/2018s1

Contents

1 Overview

This is an introductory course on Deep Learning, aimed at graduate students. Undergraduate students in their last years of college can also attend. Graduate students must register for MO810, while undergrads should take MC959. The prerequisites are:

- Proficiency in programming. We'll use Python for most of the course. You may get by with R or even Octave (free replacement for Matlab), but I haven't tried these languages except for very simple examples. For the more complex examples, it is much more convenient to use a framework such as TensorFlow. There is room in the schedule for a couple of classes on Python and TensorFlow.
- Basic computational performance analysis, complexity concepts.
- Introductory calculus, some linear algebra.
- Some graph theory, probability, and information theory.

The course will have both a theoretical component and a practical component. Each class will have a theoretical part and a practical part. In the theoretical side, we'll cover deep forward networks, back-propagation, regularization, convolutional layers, recurrent networks, and learning with no training data. Guides for this part include the Deep Learning book by Goodfellow et al. [GBC16], and an entry in Karpathy's blog [Kar16].

For the practical part, we'll start with very basic computational neural networks that can learn simple Boolean functions, and then proceed to more complex networks that can learn handwriting/images, sequences, and strategy games. Guides for this part include Nielsen's online book [Nie15], and an entry in Karpathy's blog [Kar15].

Grading will be based on in-class short tests, one per lecture, and on student projects.

We won't have access to powerful computers or GPUs. Therefore, our emphasis in the course will be on efficiency, i.e., trying to find the smallest net that does the work, or as much of the work as possible. Ideally, the code each one of us produces or tests will run in our own laptops, desktops, and other computers available to us.

2 Projects

Part of the grade will come from student projects. Students will choose a problem that they are interested in and design, train, and test a network that solves this problem. We list below a few suggestions, but your are free to choose your own problem. Students can form groups of 2 to 4 members, but in that case each member must have a definite task within the group, declared in advance, and each person will be graded separately according to their own performance.

Students will have to keep a diary, which is a written record of what they've done, to be able to communicate the results and reproduce them later on. There should be entries every day. Plan to keep yourself busy with the course for 2 to 2.5 hours a day, and use the final part of this time to summarize in the diary what you did on that day. Diaries can be requested by the instructor at any time and have to be handed out with the Project Report. Diaries are individual artifacts, so even if you are working in a group, you are responsible for keeping your own diary.

There will be checkpoints for the projects and a final presentation and handout. The checkpoints are as follows:

- April 3rd (One month after classes begin): problem chosen and data obtained.
- May 3rd (Two months after classes begin): model and baseline defined.
- June 14th: Project Report, Diaries, Code, and Video presentation (4 min) due.
- June 14th to 28th: Fine tuning of Project Deliverables. Your reports, presentations, diaries, etc. will be checked and may receive suggestions to be applied in order to merit a better grade.
- Remember: we won't have big computers, or GPUs — we'll need to be efficient! One of our goals will always be to find the smallest net that does the job.

In summary, students will have one month to select the problem and gather the data; another month to define the model (i.e., network architecture), and baseline (i.e., the best results in the literature for this task); and another $\sim$ 40 days to train their network, fine tune the hyperparameters of

the model, write a report, and produce a video (4 minutes maximum) summarizing the experience. The code must be made available as well. The rest of the term will be used for us all to collectively review the results of every group and suggest improvements.

Project Suggestions

- Email filter (spam killer)
- Computational biology: genes associated with a certain disease
- Handwriting (MNIST, etc.)
- Images (CIFAR, etc.)
- strategy games (checkers. etc.)
- online games (Pong, Dinousaur, etc.)
- predictions in the stock market
- writing poetry
- natural language translation

3 Office hours

By appointment only.

4 Program

The following table contains the courses's program, regarding both the theoretical and practical parts. While the items faithfully reflect the program, the schedule is only approximate. Small changes are possible, and will be posted in the course's site:

`www.ic.unicamp.br/~meidanis/courses/mo810/2018s1`

This schedule does not include holidays and other relevant dates. For a more complete schedule, visit the course's home page above.

Date	Theory	Practice
Feb. 27	Course outline	
Mar. 01	Introduction	
Mar. 06-08	Linear Algebra	Simple examples
Mar. 13-15	Probability and Information	R, Octave
Mar. 20-22	Numerical Computation	Python
Mar. 27	Machine Learning	TensorFlow
Apr. 03	Projects: topic, data	
Apr. 05	Machine Learning	TensorFlow
Apr. 10-12	Deep Forward Nets	IRIS
Apr. 17-19	Regularization	MNIST
Apr. 24-26	Optimization	Tic-Tac-Toe
May 03	Projects: model, baseline	
May 08-10	Convolutional Nets	To be announced
May 15-17	Recurrent Nets	DNA classification
May 22-24	Learning Without Data	Learning text
May 29	Practical Methodology	Master Mind
June 14	Projects Due	
June 14-28	Project Reviews	
July 10	**Final Exam** (undergraduates only)	

5 Grading

Grading for both graduate and undergraduate students will be based on the **score** S obtained by the student during the term. For gradute students, this score will then be converted to a letter grade G using the table shown below. For undergraduates, the score, possibly averaged with the result from the final exame E, will be sent to the university administration as the final grade G.

The score S will be determined by the student's performance in the following activities:

- in-class tests (score: T)

- project (score: P)

The mean of T and P will be the student's score S, that is, $S = (T+S)/2$.

Test scores In-class tests will be applied on the last 15 minutes of each class, covering the previous' class topics. Typically, there will be one of two questions per test. Each in-class test will be graded on a scale of 0 to 10. Only the k best grades are kept, typically 75% to 80% of the total. This way, a student can skip a few tests without significant impact on their grade. The average of the best k tests will be the test score T.

Project scores Projects will be graded taking into account the following aspects:

- problem relevance and difficulty
- student diary
- final report, after suggestions (clarity, enough information to reproduce the results, organization, conciseness, etc.)
- code (style, comments, efficiency, etc.)
- video presentation (clarity, objectivity, respect of time limit, etc.)

Projects will be graded on a 0 to 10 scale, to yield a project score P.

Score to letter grade table To be used for graduate students.

Score	**Grade**
$8.5 \leq S \leq 10$	A
$7.0 \leq S < 8.5$	B
$5.0 \leq S < 7.0$	C
$0.0 \leq S < 5.0$	D

Final exam Undergraduate students who end up with a score S below 5.0 may take a final exam, and their grade E on this final exam will be combined with S to form the final grade:

$$G = (S + E)/2.$$

References

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[Kar15] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. Blog Entry, May 2015. karpathy.github.io/2015/05/21/rnn-effectiveness/.

[Kar16] Andrej Karpathy. Deep reinforcement learning: Pong from pixels. Blog Entry, May 2016. karpathy.github.io/2016/05/31/rl/.

[Nie15] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com.