CSC 411/2515 Projects

Ethan Fetaya, James Lucas and Emad Andrews

1 General Guidelines

The idea of the final project is to give you some experience trying to do a piece of original research in machine learning and writing up your results in a paper style format. What we expect to see is an idea/task that you describe clearly, relate to existing work, implement and test on a dataset. To do this you will need to write code, run it on some data, make some figures, read a few background papers, collect some references, and write a few pages describing your task, the algorithm(s) you used and the results you obtained.

Students can work on projects individually or in pairs (*but the expectation will be proportionally higher*). The grade will depend on the ideas, how well you present them in the report, how clearly you position your work relative to existing literature, how illuminating your experiments are, and well-supported your conclusions are.

You can use word or whatever you like to format your report, but ideally it will be in the format required for NIPS, which can be found here `https://nips.cc/Conferences/2016/PaperInformation/StyleFiles`.

The idea is that this project report should be a manageable amount of work, but that if you want to turn your project into a paper, everything in the project report will need to be done anyways. If you feel that your project won't fit into this rubric, please talk to me. There are many ways to make contributions to a field!

Your project must implement one or more machine learning algorithms and apply them to some data. Your project may be a comparison of several existing algorithms, or it may propose a new algorithm in which case you still must compare it to at least one other approach. Being a graduate student, you are free to pick a project of your own design. Regardless of which way you select a project, you cannot use the excuse that you got a "bad project" to explain doing a poor job on it. So select wisely! Some example project approaches might be,

- Explore an existing algorithm/regularization scheme/etc. Show ample empirical evidence to support some new or existing claims.
- Propose a new machine learning approach based on existing work and show empirical evidence that this is viable.
- Apply an existing machine learning algorithm to a unique domain showing that the ML techniques used are competitive with existing approaches (or offer some other benefits, e.g. interpretability or run time).

You are free to use any third-party ideas or code that you wish as long as it is publicly available. You must properly provide references to any work that is not your own in the write-up. The project is not intended to be a stressful exercise; instead it is a chance for you to experiment, to think, to play and to hopefully have fun! Start with simple methods that work more or less out of the box and go from there.

2 Specific Requirements

Length: 4 to 8 pages, not including appendices. Don't be afraid to keep the text short and to the point, and to include large illustrative figures.

1. **Abstract (5 points):** that summarizes the main idea of the project and its contributions.
    - Should be understandable to anyone in the course.
    - You don't need to say everything you did, just what the main idea was and one or two takeaways.
2. **Introduction (5 points):** that states the problem being addressed and why we might want to solve it.
3. **Figure or diagram (10 points):** that shows the overall model or idea. The idea is to make your paper more accessible, especially to readers who are starting by skimming your paper.
    - For the project, taking a picture of a hand-drawn diagram is fine, as long as it's legible.
    - For camera-ready diagrams, we recommend using Tikz, a LaTeX package.
    - Try to be clear whether arrows indicate computational flow, or conditional dependencies, or both.
4. **Formal description (20 points:** of the model / loss function / conjecture / problem domain. Include at least one of:
    - An algorithm box.
    - Equations describing your model.
    - A theorem or formally stated conjecture.
    - A formal description of a problem domain.

    Differentiate your work Highlight how your model is different from other approaches, or what the main relevant considerations are for the domain. This can be done by comparing it to an existing model, perhaps by using another diagram or in words. E.g. if you are proposing a new algorithm that only changes one line in an existing algorithm, highlight that one line, or do a side-by-side comparison.
5. **Related work (20 points):** section and bibliography.
    - If your project builds on previous work, clearly distinguish what they did from what your new contribution is.

- Include a 1-2 sentence summary of other closely related papers. You might not know about all related papers (or have time to carefully read all related papers), and that's OK for this project. A rough guide is that you should be able to
- find 3-4 closely related papers, and another 3-4 papers that all those papers cite as foundational work. These foundational papers are often cited in the introduction.
- Using bibtex is annoying at first, but Google Scholar can give you the bibtex entries, and it will save you time in the long run.

6. **Comparison or demonstration (20 points):** Include at least one of:
   - A demonstration of a theorem or conjecture. For example, an example or counter-example.
   - A comparison of data generated by your model to a baseline model. Qualititative evaluation is OK for the project.
   - An experiment demonstrating a property that your model has that a baseline model does not. Experiments should also include a description of how you prepared your datasets, how you trained your model, and any tricks you used to get it to work.
   - If doing a review, include a table comparing the properties of the different approaches.
7. **Limitations (15 points):** of your approach.
   - Describe some settings in which we'd expect your approach to perform poorly, or where all existing models fail.
   - Try to guess or explain why these limitations are the way they are.
   - Give some examples of possible extensions, ways to address these limitations, or open problems.
8. **Conclusions (5 points):**
   - State the results achieved in relation to the problem described in the introduction.
   - Repeat the main takeaways from your paper.

2.1 Project proposal

You must turn in a brief project proposal. Your project proposal should describe the idea behind your self-defined project. You should also briefly describe software you will need to write, and papers (2-3) you plan to read. Please also say if you will have a partner, and if so, who it will be.

Include your email address on your proposal. We need this to contact you and arrange meetings to discuss your proposal. The proposal is due Nov. 4, by email to csc411-20179-instrs@cs.toronto.edu.

2.2 Project submission

Your final submission must include:

- Your research paper
- We encourage you to attach your code. Although this is not required it will be helpful to better judge your work.

3 Deadlines and Submission Instructions

Make sure that you don't miss any of the following deadlines:

- The **project proposal** should be submitted electronically via email to csc411-20179-instrs@cs.toronto.edu. The submission deadline is noon on **November 4th**.
- The **project report** should be submitted via MarkUs by noon on **December 20th**. Name your submission `Project-*your-student-id*.pdf`. A penalty of 25% will be applied per day that the submission is late.

4 Friendly Advice

(For this project and for doing research in general)

- **Be honest!** You are not being marked on how good the results are. It doesn't matter if your method is better or worse than the ones you compare to. What matters is that you clearly describe the problem, your method, what you did, and what the results were.
- **Be careful!** Don't do foolish things like test on your training data, set parameters by cheating, compare unfairly against other methods, include plots with unlabeled axes, use undefined symbols in equations, etc. Do sensible crosschecks like running your algorithms several times, leaving out small parts of your data, adding a few noisy points, etc. to make sure everything still works reasonably well. Make lots of pictures along the way.