CS281 Final Project Requirements

Objective

The objective of this final project is to explore new research in machine learning. The ideal outcome would be a paper that could be submitted to one of the top machine learning conferences, such as NIPS, ICML, UAI, AISTATS, or KDD.

Types of Papers A strong paper along these lines is one that develops a new or improved algorithm (runs faster, scales better, makes better predictions) that learns to generalize from experience, broadly defined. Such a paper would demonstrate theoretical and/or empirical improvements over the state of the art. Papers tend to fall in to one of three categories:

- **Modeling Papers** Modeling papers introduce a new probabilistic model that captures important characteristics of data that had previously been unexplored. A successful modeling paper will provide statistical properties of the proposed model, derive tractable inference algorithms for the model, and justify its utility theoretically—as unifying or providing insight into existing models—or empirically—compared with exisiting models.
- **Inference Papers** Inference papers develop new approaches for performing inference and learning in exisiting models. They might also focus on inference in specific settings, such as in parallel or distributed systems or systems with constrained resources (such as limited limited memory, real-time performance requirements, or streaming data).
- **Application Papers** Application papers are grounded in a particular problem domain. They often involve few to moderate model or inference innovations but generate impact by solving an important problem.

These options are ordered roughly in order of decreasing risk for a final project. Modeling and inference papers often require innovative ideas about machine learning that may be difficult to come by in a single semester; it may be helpful, therefore, to initially focus on a specific problem domain that you find important and exciting. Consider what the fundamental task is that needs to be solved and think about how it might map onto material from class. Catalogue the types of data that are available and consider how these might be exploited. What are the features that will help your algorithm make decisions or predictions? Don't be afraid to make assumptions that help establish an abstraction. Prefer abstractions and assumptions that may generalize beyond the immediate task at hand.

Creating Evaluations The next step is critical: define *quantitative metrics* and *honest baselines* for success on held-out test data. Classification accuracy? Area under the ROC curve? Predictive log probability? Rand index? If you are focused on resource constraints, then perhaps your metrics will be curves that measure prediction as a function of, e.g., memory usage or CPU time. If possible, consider several possible metrics so that it will be easy for the eventual readers of your paper to map your algorithm onto their priorities. Once you have laid out the task, data and metrics, try to apply the dumbest, simplest possible algorithm first. *Do not* immediately try your new fancy idea. If you have a classification problem, try logistic regression first. Try things from the literature that seem like they would be applicable. Establish baselines for comparison that are honest attempts at doing well on the problem.

If you have chosen to do a more applications-oriented project, you may learn that logistic regression is difficult to beat. What does that mean? Maybe you need to focus on extracting features

rather than a fancy classifier. Applying simple things first will help you understand where the frontier of the problem lies and help you determine whether your abstraction actually provides the information that you require for the task. In the end, it may be difficult to make a methodological contribution. If you have taken a problem-driven approach, however, then you have still done useful research by improving our understanding of how machine learning algorithms behave when applied to new problems. For two strong examples of papers that are important, but fundamentally about empirical evaluation, I suggest looking at Jarrett et al. (2009) and Coates et al. (2011).

If you have chosen to do a more methodological project, such as developing better inference algorithms, you will want to test your algorithm in a variety of settings: How does your approach compare to your honest baselines in resource constrained settings? How does it compare for datasets with large numbers of observations? Large numbers of dimensions? Large numbers of observations and dimensions? How does it compare for different kinds of noise of model misspecification, such as the wrong number of latent variables or unexpected correlations in the noise? You do not have to try all of these settings, of course, but you should do careful testing on a variety of synthetic and real data sets. You may find that your proposed approaches do not outperform your honest baselines. That is okay, as long as you can explain why your baselines were hard to beat.

Evaluations are also an iterative process. It is not sufficient for you to find that your algorithm outperforms baselines on several data sets. You might be able to explain why: design additional experiments that show that your approach works well when you think it should, and other experiments to show that it fails when you think it should. Running experiments can require significant computational skills so that your algorithms complete in a reasonable amount of time and consume reasonable resources. You are expected to code up your baselines honestly; you may not use a poorly thought-out implementation to claim that a baseline is not viable. **A significant portion of your project grade will depend on your ability to define quantitative metrics, identify honest baselines, and discuss the outcomes of your experiments. Carefully designed experiments and good science are paramount.**

Collaboration

You may work on this project alone or with a partner, your choice. Larger groups will be considered with permission. With a partner, you will be expected to tackle a more difficult problem. If you are working in a group, your proposal should include what aspects each person will be working on, and each individual should submit a short statement of work (no more than 250 words) about their contribution as well as their final write-up. Unless there are extreme extenuating circumstances, everyone in the group will receive the same grade.

Deliverables

There are four separate components of the final project. Each of these is due as a PDF file uploaded to Canvas by 5pm of the day specified. The proposal and status report may be submitted up to a week late with a 50% penalty. There will be no extensions given for the poster and final report.

Project Proposal (13 October 2017) Write a 1-2 page document that describes the plan for your project. This does not need to be a comprehensive document and I expect that it will be speculative. Your focus should be on identifying the questions you wish to answer about your data or your method and specifying clearly what "success" will mean. Specifically, your project proposal should include

- *Problem Statement/Objective*: This should clearly state what problem you are trying to solve. If you have developed a new model, explain what models this work will build on and how it resolves deficiencies. If it is a new algorithm for inference, explain the regimes for which you think it will be well-suited. If you are developing a new theoretical contribution, discuss the theorems you will prove. For problem-driven papers, discuss the data and the unique challenges that make this interesting.

- *Approach*: Tell us about the code that you will be writing or the theorems that you will be proving. Non-theory CS281 projects should have a significant implementation component in which you will be coding up your own inference algorithms; describe what these algorithms will be. (For example, "I will implement a collapsed Gibbs sampler for...")

- *Evaluation*: Identify relevant work and algorithms you intend to implement as baselines. (Note: it is okay for a project to fail to beat the baselines, but you must be able to explain why.)

- *Collaboration Plan*: If you're working in a team, include who you'll be working with and how you intend to split up the work.

- *Double-dipping*: You may use work that you are completing as part of another final project or your research for your CS281 final project, as long as (a) your proposed work meets the technical requirements for a CS281 project (that is, significant implementation of your own algorithms) and (b) you have discussed double-dipping with your other course instructor/advisor and have their approval. You must explicitly state "I have discussed combining my CS281 final project and my [research/course number of other project] with [name of instructor/advisor] and have his/her approval."

Abstract and Status Report Building from your proposal, write a three to four-page document on the status of your project. This document should: (a) contain a paragraph that is a draft of the abstract for the final paper (b) reflect any changes you have to your approach since your proposal, (c) include results for at least some baselines or, for more theoretical projects, any proof attempts so far.

Poster Presentation We will have a class poster session, where you will present a conference-style poster. SEAS will pay for the poster printing (more details about this to come). You will also submit the poster as a PDF.

Final Report Use the ICML, UAI, or KDD style templates to write a paper of up to eight pages (not including references). This paper should have a typical conference style, with abstract, introduction, etc. You should clearly state what problem you are trying to solve, introduce and explain your approach, and review the relevant literature. It should explain in detail the experiments that were run, show their results and discuss conclusions that can be drawn.

References

Adam Coates, Honglak Lee, and Andrew Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best

multi-stage architecture for object recognition? In *Proc. International Conference on Computer Vision (ICCV'09)*. IEEE, 2009.