CS 3220: COMPUTATIONAL MATHEMATICS FOR COMPUTER SCIENCE

The project: As part of this course you will be required to complete a course project on a topic of your choosing (some examples will be added here in the near future and given in class). This may be done individually or in groups of up to three students (flexible), though the scope should be scaled accordingly. The project should contain a mix of implementation, application examples, and theoretical discussion. However, depending on the specific topic the relative composition of the project may vary as appropriate.

Proposals are due on November 2 by the end of the day and final reports are due December 21 at 5 pm ET.

Proposal and final report submissions will be done via the CMS and only one submission is required per group. Parts of the code that are explicitly referred to in the report should be included in an appendix and a complete copy of the code should be submitted as a separate compressed file.

General requirements: The goal of this project is to give you the opportunity to further explore one of the topics covered in the course. However, the scope of the project is not strictly limited to the areas we discussed in class. We have discussed a rather broad range of topics, so it is perfectly reasonable to propose a project on a related/relevant topic. For example, there may be algorithms/problems related to another class you are interested in learning more about. Irrespective of the topic, the project should include (or at the very least touch on) all three components of the class: linear algebra, statistics, and optimization.

As noted above, the project has three primary content requirements: implementation, application, and theoretical discussion. The relative composition of those three points may vary, but all must be present in some manner. More specifically:

- **Application:** Here, you should include the formulation and discussion of the applications you are interested in and why the algorithms/methods you are considering are relevant/useful. This should also include actually testing your implementation on real or model problems for a given application. There is a wide variety of complexity in the practical formulation of various application specific problems, so this may not always be practical. For example, there are various modern machine learning techniques that leverage the tools we covered in class. However, their complexity may make the formulation of a completely realistic example prohibitive and you may instead consider a simple model problem. Similarly, for a data science driven project you may use common "representative" data sets to test and asses the validity of your methodology.
- **Implementation:** This component may consist of the implementation and testing/validation of algorithms relevant to your project. For example, a project using matrix factorizations could include an implementation one or more of the factorizations we discussed in class (or a related one). However, it is also important to test, experiment with, and validate your implementation. The resulting discussion may address questions such as: does the algorithm behave as expected/theorized? What are its limits? Does your implementation achieve the desired computational scaling? How have you designed experiments to test the validity of your algorithm?
- **Theory:** This portion of the project may look like a light literature review. You should look into the theoretical properties of the algorithms you are using and discuss them at a high level. This ties into other portions of the project as well. For example, you can run experiments and see if you observe worst case behavior or satisfy the relevant bounds. How specific/detailed the existing theory is will vary with your topic, for the sake of this project the more important point is to be able to understand and interpret what does exist.

Practically, there are two deliverables for the project: a proposal and a final report. All of these items (outlined individually below) should be clearly written and cleanly presented — this will factor into your grade on the project. As a brief administrative note, The Cornell Code of Academic Integrity applies to the project. While the applications, algorithms, and theory you discuss may not be original, the project itself is original work and should appropriately cite the relevant references as needed. For some projects, the use of open source toolboxes may be a necessary part of the implementation and their use should be documented and referenced accordingly.

The proposal: This is a brief, (up to) two-page summary of what topic you would like to tackle for the project. Deciding and scoping what exactly you want to do is very much part of the project itself. I am happy to chat during office hours or by appointment about ideas you may have and provide relevant references. The proposal should provide an overview of the topic you are choosing, the application area that motivates it, and an outline of your plan to address all three required aspects of the project. Upon submission, I will provide feedback on the proposals and make suggestions on the topic, scope, and design of your project.

The final report: The final project report should summarize your investigation. This means discussing and presenting the algorithms and applications, summarizing and interpreting the relevant theoretical results, and presenting your experimental exploration. The specific format and flow of the document is up to you, but it does need to address the required components of the project in a clear and professional manner. I can provide feedback on early drafts with sufficient lead time (*e.g.* a week before the due date).

Example projects: A few specific examples are given below, but generally one way to pick a project is to pick an application of interest to you and explore from there considering the types of algorithms that are relevant for that problem, their properties, and how effective they are at providing the desired solution. Many examples may be found in the various references available on the course website.

- Randomized algorithms: We have discussed only a (very) small subset of recently developed randomized algorithms. A project could consist of picking a small number of these with a similar goal and an application (*e.g.* a problem where low rank factorizations are relevant) that fits with their objective. It is then possible to implement and test the algorithms for model problems from the application, and also probe them to find their failure modes. In addition, you could compare and contrast some of the rank structured algorithms with the same goal and explore the oft discussed enhancements to some of these algorithms. For example, using iteration when singular values decay slowly or observing potential conditioning issues with one-pass algorithms.
- Building simple neural networks: It is probably not an understatement to say that neural networks have garnered significant interest in solving application level problems. While typically one may use a variety of rapidly evolving libraries and code bases to use neural networks to solve a given application problem, the tools from this class enable to you develop your own simple code to train and test neural networks. Therefore, you could explore the uses of neural networks in applications and then build your own and apply them to simple model problems (it may be computationally prohibitive to apply your own to a "real" application, but you can get a sense for the performance, behavior, etc. using simple model problems.
- Kernel methods in machine learning: One broad class of methods to solve some modern machine learning problems are known as kernel methods, for this project you could explore how these methods work and the underlying algorithmic questions they raise. For example, one specific instance of this is so-called kernelized spectral clustering, a methodology well-developed over the past 20 years and one that conveniently necessitates solving an eigenvalue/eigenvector.
- Regression problems: Given a proposed generative model for some real-world data we are often interested in fitting parameters of that model to a specific data set. We saw one instance of this in class when studying least squares problems—there are many more examples of this. As part of this project you can explore fitting models to data (defined broadly) and the linear algebra, statistics, and optimization that goes into such a process. There is a lot to explore here, there are linear and non-linear methods, interpretations of their effectiveness based on noise models, and a plethora of model regularization choices to consider.