CS 6220: DATA-SPARSE MATRIX COMPUTATIONS

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. While 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 due April 1 by the end of the day and final reports are due TBD (sometime May 18 – 25)

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 defined data-sparsity rather broadly, so it is perfectly reasonable to propose a project on a related/relevant topic. For example, there may be algorithms related or relevant to your research and you could use this project as a means to explore them.

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:

- Implementation: This component may consist of the implementation and testing/validation of algorithms relevant to your project. For example, a project on rank-structured matrices could include an implementation of the factorization 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?
- Application: Here, you should include the formulation and discussion of applications for which the algorithms discussed in the project are relevant/useful. It may also include actually testing your implementation on real or model problems for a given application. Though 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 PDEs that may be formulated in a manner amenable to the use of rank-structured factorizations. Such formulations may be discussed, and simple model problems solved.
- Theory: This portion of the project may look like a literature review. You should read a couple of papers (or more, depending on the group size), and be able to understand, summarize, and interpret the theoretical foundations of the methods you are examining. 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 three deliverables for the project: a proposal, a presentation, 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 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, 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 and outline 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 presentation: To complement the written components of the project, there will be presentations given during the last week of classes. This will take the form of something akin to a poster blitz (you may be familiar with this increasingly popular component of academic conferences). Each individual will give a brief overview of their project. Slides will be collected the

day before the presentations to facilitate keeping to the schedule. These presentations happen significantly prior to the final report due date, so results may be preliminary — that is okay. The key here is to express what you are doing and a few relevant/interesting points about the implementation, application, or theory.

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 or algorithmic goal and a corresponding algorithm and explore from there. Many examples may be found in the various references available on the course website.

- Rank-structured matrices: We discussed one specific rank-structured matrix algorithm in class, you could take this or a related algorithm and develop an implementation, read and understand the theoretical underpinnings, and apply it to an application like the one we described in class (boundary integral equations). However, you could also consider other applications such as Gaussian processes, sparse matrices arising from PDE discretization, or volumetric integral formulations. The specifics of the rank structured algorithms that work best may vary with the application, and any of these could be the starting point for a project. With more people, you could compare and contrast a wider variety of algorithms that fall into this class.
- Randomized algorithms: We have discussed only a 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.
- Sparse recovery and compressing sensing is a broad area with a lot of potential model applications. Furthermore, many of the algorithms exhibit phase transitions. Too few samples and you cannot recover the desired result, but as soon as you pass the threshold exact recovery of the solution becomes possible with high probability. You can recreate this phenomenon with your own implementations of these algorithms and problem setups, this would certainly also require a good understanding of the relevant theory. This project would also give you the opportunity to familiarize yourself with good convex optimization tools such as TFOCS or CVX as they may be required for the implementation.
- A canonical model problem for low-rank plus sparse approximations is the separation of slowly varying backgrounds from rapidly varying objects in movies. You could explore this problem; see how well you can make it work and when it fails. This would again require the use of optimization packages in the implementation.