Final Project details and Formatting Instructions

Nate Veldt (write your names here instead)

Abstract

You should include a short abstract that covers what your project is about, a little bit about what makes your project new compared to existing results (even if it's just updated proofs of a new survey and summary of previous results), and a summary of what you show.

1 Instructions regarding the use of Generative AI tools

Tools such as ChatGPT or Co-Pilot are allowed, under the following conditions:

- You carefully verify the output of any such tool to learn from it and check its correctness
- You include details in your report (can be in the appendix) about how they were used (e.g., what prompts), and what quality control measures you used to verify correctness

2 Project Instructions

You will collaborate with a team of 2-4 people on a project covering some topic on computational methods for data science[1]. You are not required to prove a brand new result or design a brand new algorithm, but you are certainly welcome to pursue these goals.

Your project should of course do *something* that can be considered "new", that goes beyond, for example:

- Downloading someone's code and re-running a set of existing experiments they have already set up
- Copying down a set of results from an existing set of notes

In particular, when I go to the sources you use, I have to see that there is a clear and meaningful difference. This can take many different forms such as:

- Providing your own implementation of someone's algorithm
- Running someone else's algorithm and implementations on new/different datasets
- Synthesizing/summarizing results/ideas/code from multiple different sources
- Comparing several different algorithms for the same task
- Re-writing an existing proof in your own words and translating it into notation that we used in this class.
- Re-writing someone else's proof but including theoretical details they skipped.
- Proving a result that was stated but whose proof was skipped because of space, or wasn't shown because it mostly follows some previous result that was proven elsewhere.

[1] You may work on a team of 1 if you want to go with one of the default projects. In general, I'd encourage people to work in teams.

Preprint. Under review.

3 Key Components of the Final Project

There are three key components that will go into your project:

1. **Survey and summary of previous work**: you should provide background on the topic your are studying, why it is relevant, how it is applied, and what are major results on this topic.
2. **Technical content**: formal details including notation, models, objective functions, algorithms, and (if applicable) proofs of key technical results
3. **Coding and experiments**: implementation of algorithms, and numerical experiments on real and/or synthetic datasets. These can be displayed in figures and/or tables.

The first two components are strictly required, though for the second you do not necessarily have to prove anything if you want your project to instead be more focused on implementations. Although I should see some math when you are explaining your topic and how it is formalized, you may choose to *not* include full rigorous proofs of any theoretical results, but only if you have a very strong coding/experiments component. Alternatively, you may choose to not include a coding/experiments section if you have a very strong emphasis on theory. But you are encouraged to include all three components in your final write-up in some way.

4 Guidelines and Instructions for the Proposal

By April 22 at the latest, one team member from your group should submit a proposal for what topic you will be addressing for your final project. You can simply send this to Dr. Veldt via email. It should include the following:

- Who your team is.
- What is the problem/topic you'd like to study
- An overview of what you will include in terms of survey (e.g., you have found xyz papers that you will start with)
- An overview of what you will include or try to accomplish in terms of theory
- An overview of what you will include or try to accomplish in terms of code and experiments.

You do not have to have a complete idea of what you will do, but try to be as specific as you can. I will then provide some feedback on whether you need to make any adjustments (in terms of what you will do) in order for this to be an acceptable project.

5 Guidelines and Instructions on Write-up

- **You should use the .tex file for this set of instructions as a template for your final write-up**.
- Include references using BibTex. You should have at bare minimum 5 references, but ideally you would have quite a few more than that. This tex tile comes with a bib file with example references, that you cite like this [**?**].
- This project should represent your original writing and presentation of thetopic . You must cite all sources appropriately and use only original figures.
- You are limited to 6 pages. Do not adjust margins or font size to be under the limit. You will likely be over the limit at some point in a first draft. When you are at that point, go through and decide what to shorten and cut to make the writing clearer.
- You may include extra details in an appendix, if you wish. This may or may not be read carefully.
- If your project involves code, you are required to also upload a .zip file with your code including a readme for how to reproduce all of your experiments. If you use datasets that are too big to include in the .zip file, include instructions for how to access the dataset.
- All members of the project team should upload the final pdf on canvas by the end of the day on December 10.

Table 1: Sample table title

	Part	
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

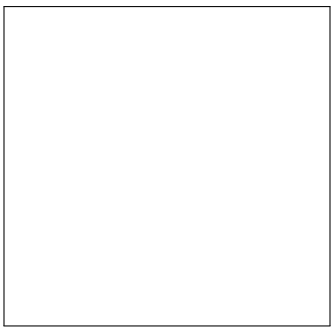

Figure 1: Sample figure caption.

6 Other Writing Tips

- You are encouraged to use some figures of some sort. This may be to illustrate some technical concept, or be a plot of experimental results. Here's an example for how to include a figure: see Figure 1. You can use other ways to include figures in Latex as well. Any figure you include should be your original figure. *Do not simply download someone else's figure and include it in your project.*
- State any results that you cover (if you choose to cover theoretical results) using nice theorem environments. Like this:

  Lemma 1. *This is a lemma.*

  Theorem 1. *This is a theorem.*
- Tables are good for displaying results as well. Here's an example of a table, in Table 1.
- Be careful to explain all your terminology and notation well. Do your best to follow the terminology and notation we have used in class, when possible.
- All team members should be broadly aware of what's going on in every aspect of the project. However, you can certainly consider assigning people to be primarily in charge of different aspects, e.g., person 1 does a literature review and main draft of the intro and background, person 2 works on providing some theoretical results, team members 3 and 4 implement some algorithms and try some experiments.
- All team members should be a part of the writing process in some way. At very least, everyone should be a part of the final proofreading and checking. If you don't all understand the write up of a proof, something is probably unclear and should be re-written. Same with all other aspects of the paper.

7 How will this be graded?

The project will be graded out of 25 points. Here are some questions to make you aware of what I will be checking for when I am grading.

- What is the (broad) topic being addressed here and why is it important? What are some examples of applications?
- What is the (specific) set of results that are covered in the project write-up? What theory was presented? What experiments were shown?

- What's new or different about this project as compared with other papers I can go read?
- You do not need to exhaustively survey previous literature, but I'll be checking to see if you have given a decent overview of previous work that includes at least a few relevant papers.
- If included, are proofs correct? Are they clear?
- Is the notation consistent and clearly defined?
- If the project is primarily about summarizing and surveying previous papers and results, does the paper include a nice overview and summary of several different papers (as opposed to just re-proving everything from one paper)?
- Is the paper readable? Is the writing clear? Are there grammatical/spelling/notation errors? Just a few, or many of them?
- Are experimental results (if they exist) explained clearly and with helpful figures and tables?
- Does this project adequately address the goals set forth in the proposal? (It may not look exactly the same, but it should be commensurate. E.g., if you were unable to do thing 1, swap it out with a result about thing 2).
- Is the paper within the 6 page limit?

I may or may not pull up your code and check it. I may also check it for example if something seems strange or suspicious in your plots, or if I simply want to try to reproduce a plot. The primary thing I will be checking for is reproducibility.

8 Sample paper outline

Here's an example of an outline that you can use when writing up your results, but you do not need to follow it. It is only a suggestion.

1. Introduction: What is the problem, applications, and what are the key results included in this project.
2. Preliminaries (1 section): Technical background, explanation of terminology and notation that will be used.
3. Theoretical Results (1-3 sections) : Prove new results that you've come up with, or organize and present existing results.
4. Experimental Results (1 section): Describe implementation details, experimental setups, datasets, and experimental results
5. Related work (1 section): If there's anything else that's related to the project that is worth mentioning.
6. Conclusion and Discussion (1 section): Short summary of results, discussion about limitations, surprising findings, possible future work, etc.

9 Default Projects

Everyone has the option to just do one of the "default" projects—the first is on solving least squares problems and the second is on symmetric positive definite systems. Both focus on using a variety of different techniques we have seen in class. Students may work in teams of 1-4. If there are more people on the team, there should be a commensurate increase in what you do (e.g., implement more methods to compare, run more experiments, etc.)

You are allowed to use built-in methods for matrix factorizations (SVD, Cholesky, QR) and basic things like computing norms. Otherwise you should implement all aspects of your algorithm (which should amount to matrix and vector products mostly). You can use more sophisticated build-in methods as a point of comparison to see how your code stacks up against state-of-the art, but the expectation for this project is that you are mainly getting practice implementing the method from the ground up. You must turn in your code with your project as a zip file, with a clear README on how to reproduce your results.

You can add to this and be creative (e.g., other datasets, other methods not explicitly mentioned here) if you wish.

9.1 Default project 1: Least squares

Recall that for least squares the goal is to solve

$$\min \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

which (for full rank matrices with more rows that columns) is equivalent (in terms of exact arithmetic) to solving the normal equations:

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$$

where $\mathbf{A}^T\mathbf{A}$ is a symmetric positive definite matrix. Here are some approaches you could take to solve this:

- Direct methods: SVD, QR, Cholesky factorization for $\mathbf{A}^T\mathbf{A}$
- Conjugate gradient applied to the normal equations
- Standard gradient descent or SGC (applied directly to the least squares objective or to the normal equation)

Note that some of these methods may be better ideas than others. The idea is for you to explore options and get the feel for how to (1) implement this techniques and (2) see in practice how performance varies. You should choose a few different algorithms and compare their performance on some or all of the least squares problems here:

- https://sparse.tamu.edu/HB/well1850
- https://sparse.tamu.edu/HB/well1033
- https://sparse.tamu.edu/HB/ash958
- https://sparse.tamu.edu/HB/ash85
- https://sparse.tamu.edu/HB/ash608
- https://sparse.tamu.edu/HB/illc1850
- https://sparse.tamu.edu/HB/ash331
- https://sparse.tamu.edu/HB/illc1033
- https://sparse.tamu.edu/HB/abb313

(You may need to generate different possible right hand side vectors **b** for these). It is up to you to choose which algorithms to implement and run, and in some cases you may have to make choices about hyperparameters (e.g., different step lengths for GD and SGD). Just make sure to:

- Try both iterative and direct methods
- Try at least 3 different approaches (and possibly more especially if you are working as a team)
- Carefully explain what techniques you applied and any important design choices (e.g., did you set steplengths in a certain way? Did you apply some strategy directly to the least squares objective or to the normal equations?)

Then write up details on what methods you tried and how they did in terms of runtime. You should display results in Tables and Figures (Figures may be helpful especially when comparing residual/error against the number of iterations and/or runtime).

You should make sure you are using methods that at least in exact arithmetic able to find the optimal solution. You should also confirm that you're getting roughly the same numerical solution, even if there are slight deviations (some of which may be due to numerical stability differences between algorithms).

Comment on your findings and any takeaways you had (e.g., with respect to speed, differences in solutions, unexpected things arising during implementations, etc.)

9.2 Default Project 2: SPD linear systems

For this part, you will generate synthetic symmetric matrices as follows:

- Set size n (e.g., 200-1000)
- Put a 1 in each diagonal entry
- For each off diagonal entry, put a number uniformly chosen at random from $[-1, 1]$ (make sure to maintain symmetry $A_{ij} = A_{ji}$)
- Choose some parameter δ (between 0.01 and 0.2) and set A_{ij} (and A_{ji}) to zero if its entry is larger than δ in absolute value
- Generate a random vector $\mathbf{b}$ (e.g., from a random Gaussian distribution)

Then, compare the following approaches for solving the system $\mathbf{Ax} = \mathbf{b}$:

- Cholesky factorization plus triangular system solves
- Conjugate gradient
- Gradient descent (with optimally chosen steplength α_k computed in each iteration, or some fixed steplength α)
- Stochastic gradient descent

See how they perform as the sparsity parameter δ changes (note that when δ gets large enough, the matrix will not be positive definite and some methods will not necessarily converge, but you can still run them and see how they do). You can also see how things change as n varies.

Compare different methods by generating some nice figures (e.g., runtime vs. plots). Here are some good points of comparison:

- Runtime of direct method (Cholesky) vs. iterative methods (especially CG) as sparsity parameter δ changes, for some fixed n
- Runtime vs. residual for gradient descent vs. conjugate gradient

Think of some other comparisons you could make as well. Feel free to get creative (e.g., try other types of symmetric positive definite matrices from other sources). Write up what methods you tried and any relevant parameter settings. Comment on your findings and any takeaways you had.

10 Sample ideas for more open ended projects

You could study another form of matrix factorization or dimensionality reduction technique, providing a summary, some implementations, and experimental results on a few interesting and relevant datasets, possibly comparing it against other simpler factorizations such at SVD

1. Nonnegative matrix factorization
2. Sparse PCA
3. CUR decomposition
4. Johnson-Lindenstrauss Lemma
5. Nonlinear dimensionality-reduction

You could pick an algorithm we studied or a close relative and provide your own new implementation of it, running it on some datasets Other ideas:

- You could run your own eigenfaces experiment
- Write a report and provide some experiments on some variant of gradient descent we didn't get to (e.g., gradient methods with momentum)

A Appendix

You may include some proofs or extra experimental results in the appendix, but I will not necessarily look at it. You will be graded on the six pages before the references.