CS250/EE387 Winter 2018: Final project guidelines

February 14, 2018

Overview

The final project should be in teams, ideally of size 2-3 (although solo projects are allowed, and very ambitious projects with four members are also okay). There are three forms the project can take, which are described in more detail below. They are:

- **Reading/Report**. You read some stuff and write a report.
- **I've had it with all this theory**. Investigate some instances of coding theory in the real world and teach me how they work.
- **Original work**. Original research involving error correcting codes, or a new implementation of something related to error correcting codes.

All types of projects will involve a write-up (10-15 pages as a target length, maximum of 20 pages unless you have a very very good reason). Please review the possibilities below, find teammates, and let me know what you plan to work on by **Thursday March 1.**

What to hand in (by email to marykw@stanford.edu) by March 1.

- Who you plan to work with (each group should only submit one thing)
- Which type of project you want to do
- If the project is a reading/report type, on a topic listed below, say with topic. If there are many papers listed and you want to focus on a subset, or you want to focus on something tangentially related to one of these topics, say that too.
- If the project is not a topic suggested below, please include a few paragraphs describing the project or topic you have in mind.

I am happy to meet or answer questions beforehand to discuss your options or possible project ideas.

Timeline

- Project Proposals due Thursday March 1.
- I will confirm that your proposal looks good by Saturday March 3. If you get me your project proposal earlier, I can probably confirm that it looks okay earlier.
- Project Reports are due 3/22, by email to marykw@stanford.edu.

Grading

The project counts for two homework assignments. The report will be assessed on the following criteria. First, all reports will be assessed on the technical content: is it sound, and does it convey a good understanding of the material. For Reading/Report projects or "I've had it with all this theory" projects, you will also be assessed on how well you synthesize all the materials out there into a coherent and engaging summary. For original work, you will be assessed as one would assess a normal academic paper, including motivation and clarity.

Reading/Report projects

A project of this type will involve reading up on a topic that we didn't have time to cover in class. You should read a few papers (and you are also welcome/encouraged to look at other resources like lecture notes or textbooks), and synthesize this information into a coherent literature review that covers some of the main technical ideas in the area. Your intended audience should be someone who has just finished this class. **Your goal is not to convince the reader that you did some work, it is to teach the reader something.**

A few potential projects are listed below, with some starting points[1], below. However, part of the project is to decide what you think is interesting: don't just regurgitate the sources listed. Rather, do some research, find good sources, and come up with a nice story to tell. Your report should be 10-20 pages long, demonstrate a technical understanding of the chosen topic, and be interesting to the reader.[2]

You are welcome to choose whatever topic you like, with instructor approval. A few are listed below. Some of these topics we might touch on briefly in class, and you can choose to go deeper for a project.

- Polar Codes
  - Arikan: Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. https://arxiv.org/abs/0807.3917
  - Guruswami and Xia: Polar Codes: Speed of polarization and polynomial gap to capacity https://arxiv.org/abs/1304.4321
- Expander Codes
  - Sipser and Spielman: Expander codes http://ldpccodes.com/papers/expandersIT.pdf
  - Zemor: On expander codes http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=910593 (The only online copy I can find is behind a pay-wall; you should be able to access it from any Stanford computer. If you are having trouble, please ask and I will send a pdf).
- Multiplicity Codes
  - You can choose either Local decoding of multiplicity codes, or list decoding, or both.
  - Local decoding:
    - Kopparty, Saraf, Yekhanin: High-rate codes with sublinear time decoding http://www.math.rutgers.edu/~sk1233/highratelocal.pdf
  - List decoding: (two papers with very similar results but different approaches)
    - Kopparty: List-decoding multiplicity codes http://www.math.rutgers.edu/~sk1233/part2.pdf
    - Guruswami and Wang: Optimal rate list-decoding via derivative codes https://arxiv.org/abs/1106.3951

[1] For more recent topics, these starting points are specific papers; for more classical topics, I've just included the buzzwords, and most textbooks in the library will include some discussion of these topics

[2] I (Mary) am interested in any core material you can possibly come up with; this requirement is to synthesize it in an interesting way and to give a good presentation.

- Algebraic Geometry Codes **Warning: this topic requires a fair amount of algebraic maturity**
  - Find a textbook or tutorial that speaks to you. I like Stichtenoth's *Algebraic Function Fields and Codes*.
- Hardness of maximum-likelihood decoding of Reed-Solomon Codes
  - Guruswami and Vardy: Maximum Likelihood Decoding of Reed-Solomon Codes is NP-hard https://arxiv.org/abs/cs/0405005
  - Cheng and Wan: On the list and bounded distance decodability of Reed-Solomon Codes http://www.cs.ou.edu/~qcheng/paper/ld.pdf
  - Gandikota, Ghazi, Grigorescu: NP-Hardness of Reed-Solomon Decoding and the Prouhet-Tarry-Escott Problem https://www.cs.purdue.edu/homes/egrigore/papers/RS_BDD_hard.pdf
- Fast implementations of the Guruswami-Sudan algorithm
  - Alekhnovich: Linear diophantine equations over polynomials and soft decoding of Reed-Solomon codes. http://ieeexplore.ieee.org/document/1181968/
  - Chowdhury, Jeannerod, Neiger, Schost, Villard: Faster Algorithms for Multivariate Interpolation with Multiplicities and Simultaneous Polynomial Approximations. https://arxiv.org/pdf/1402.0643v2.pdf, and the references in Table 1 of that paper.
- More group testing
  - Porat and Rothschild: Explicit Non-Adaptive Combinatorial Group Testing Schemes https://arxiv.org/abs/0712.3876
  - Indyk, Ngo, Rudra: Efficiently Decodable Non-Adaptive Group Testing https://www.cse.buffalo.edu/~hungngo/papers/soda10.pdf
- Coding theory in compressed sensing
  - Gilbert, Li, Porat, Strauss: For-all sparse recovery in near-optimal time https://arxiv.org/pdf/1402.1726v1.pdf
  - Ngo, Porat, Rudra: Efficiently Decodable Compressed Sensing by List-Recoverable Codes and Recursion http://www.cse.buffalo.edu/faculty/atri/papers/coding/npr12.pdf
- List-decoding and pseudorandomness.
  - Vadhan: Pseudorandomness. (Shorter survey: http://people.seas.harvard.edu/ salil/research/unified.pdf, Longer book form: http://people.seas.harvard.edu/~salil/pseudorandomness/) These have lots in them and it might be good to focus on just one relationship (eg, list decoding and extractors).
  - Ta-Shma and Zuckerman: Extractor codes https://www.cs.utexas.edu/~diz/pubs/extractor-codes.ps
- Locally testable codes in complexity theory: the PCP theorem
  - Irit Dinur: The PCP theorem by gap amplification https://eccc.weizmann.ac.il/eccc-reports/2005/TR05-046/
- Combinatorics of Reed-Muller codes
  - Kaufman, Lovett and Porat: Weight Distribution and List-Decoding Size of Reed-Muller Codes http://conference.iiis.tsinghua.edu.cn/ICS2010/content/paper/Paper_33.pdf

  - Bhowmick and Lovett: List decoding Reed-Muller codes over small fields https://arxiv.org/abs/1407.3433

- Concatenated codes on the GV bound and with efficient list decoding to capacity
  - Thommesen: The existence of binary linear concatenated codes with Reed-Solomon outer codes which asymptotically meet the Gilbert-Varshamov bound. http://ieeexplore.ieee.org/document/1056765/
  - Guruswami and Rudra: Concatenated Codes Achieve List-Decoding Capacity https://eccc.weizmann.ac.il/eccc-reports/2008/TR08-054/index.html

- Explicit codes near the GV bound
  - Ta-Shma: Explicit, almost optimal, epsilon-balanced codes. https://eccc.weizmann.ac.il/report/2017/041/

This is not an exhaustive list! If you have another topic in mind (either which you stumbled on independently, or which we discussed in class too briefly), please ask me about it! I can help you find some starting resources if you would like.

I've had it with all this theory

A project of this type will involve you going out and finding some example of error correcting codes (ideally, one of the families we've talked about in class) in the real world (for example, in storage, satellite communication, etc), and reading up on it to figure out how it works in practice. Your report should highlight:

- What are the main coding-theoretic ideas behind this implementation?
- Why was this coding scheme chosen for this application instead of another one? What are the important desiderata and how are they acheived?
- What are the real-world issues that complicate matters? How does this implementation deal with them?
- Interesting historical trivia about the development of this application is encouraged.

Your goal is to present not only the technical details (although you should demonstrate a good understanding of this) but also an interesting (if technical) story.[3]

Original work

A project of this type will be either a research project or a new implementation. For example, perhaps you have found a way to use error correcting codes in your own research. Or perhaps you are already working on something directly related to error correcting codes. Or, perhaps you realize that sage could really use a better way to do XYZ which is related to error correcting codes, or that what the world really needs is a tool to which implements Reed-Solomon error correction via aesthetically pleasing 3D printing[4] and want to implement this yourself.

In this case, your report should contain your results (or a way to access your results, if they are open-source code or a 3D-printed Reed-Solomon-related piece of artwork.)

If you have done original research, your report should be formatted like a research paper, including background, motivation, your results, and demonstrations (proofs or empirical work) that your results are sound.

[3] same as the previous footnote

[4] Okay, I don't really know what this means, but if you figure it out it could probably be cool.

If you have done an original implementation, your report should outline the motivation for why your implementation is/will be useful, any relevant background, details of the implementation, as well as any tricky issues you ran into when developing it and how you got around them. Your report should also include examples of your implementation at work (empirical results about the speed/accuracy, or anything else that demonstrates that you have effectively solved this problem).

Note: If you set out to do original work, but encounter an issue[5] and do not manage to succeed by the end of the quarter (research is unpredictable; that's how it goes), then write up what you tried, what you learned, where you got stuck, and what you are thinking about possible ways forward.

[5] An issue means that the problem was harder than your anticipated. It does not mean that you spent more time watching youtube videos than you anticipated and were not able to spend enough time on your project.