Project Guidelines AMATH 515 2024

February 6, 2024

Overview You will work in groups of 1-4 on a small project to independently extend your knowledge and skills beyond what we've covered in 515. The project is meant to be flexible and to give you a chance to explore, learn and write about something that you're interested in.

Deliverable The final product should be a report, typeset as a PDF, length of about 2-15 pages. The report should include a description of the problem, explanation of your work (methods, implementations, experiments), and a clear presentation of results using figures and tables. Code should be included in an appendix. Grading is based on overall cohesion cohesion between the question, methods and results (does everything make sense and fit together); mathematical content (how much you've done); and quality of presentation (figures & tables). The plan is for this to take somewhere around 1.5 times the workload of a homework assignment for each team member.

Optional Project Fair If there is interest, we'll have an optional and casual "project fair" during finals week with snacks for people to present and show off what they've worked on, so everyone can learn from each other. No worries if you have travel plans, etc.

1 Project Ideas

1.1 Operator Splitting

Operator splitting methods like the alternating direction method of multipliers (ADMM) provide an easy and flexible framework to solve a general class of optimization problems, and are applicable whenever you can perform a dual decomposition and compute proximal operators. These methods allow you to arbitrarily combine any number of terms in an optimization problem and give (relatively) fast convergence for relatively little effort. Implement your choice of operator splitting algorithm, and test it on a handful of problems of your choice. Some good examples include total variation image denoising/inpainting, and matrix completion with side information. Besides ADMM, another interesting algorithm to study is Chambolle-Pock, which dualizes one of the pieces and computes proximal operations in the dual. A good path here is to compare the performance of two or more splitting methods across decisions inherent in these algorithms.

1.2 Optimal Transport

Optimal transport (OT) is about the "best" way of transforming one distribution into another while minimizing a cost function.[1] While optimal transport extends to relatively general spaces and has a very interesting theory for probability measures supported on $\mathbb{R}^n$, there are still a lot of interesting problems in the discrete setting. Implement some algorithms to (approximately) solve the discrete optimal transport problem. Some choices include applying linear programming, Sinkhorn iterations (for the entropically regularized version) and operator splitting methods.

1.3 Deep Learning

Modern deep learning hinges on variants of stochastic gradient descent. Give an overview of some of the algorithms (ADAM, RMSProp, Adagrad,momentum, etc.) and stepsizing schemes (learning rate decay, cosine annealing, superconvergence, hypergradient learning rate adaptation, etc.) and perform an empirical comparison of their performance on some test problems of your choice. Obviously, no comparison here can be anywhere close to exhaustive due to the overabundance of deep learning optimization papers, so just go through a handful of ideas that you find interesting. If you want to do this with realistic (somewhat large) networks, you'll probably need access to some GPUs. This will take a bit more work (but should still be doable!) to do as a project if you don't have any previous experience in deep learning.

[1] See the first four chapters of this: https://arxiv.org/pdf/1803.00567.pdf for a nice introduction.

1.4 Variance Reduction for Stochastic Optimization

While stochastic gradient descent methods generally converge at a slow sublinear rate in terms of iteration count due to the need for learning rate decay, applying variance reduction methods like SVRG and SAGA can speed up algorithms to have a linear convergence rate by taking advantage of the finite sum structure. These methods are state of the art on certain very large scale convex optimization problems because of their fast convergence, and extremely low cost per iteration. Give an overview of some of the algorithms, and compare their convergence rate on some test problems of your choice.

1.5 Implementing interior point methods

Interior point methods basically allow you to use Newton's method to solve nonsmooth and constrained problems. They require converting optimality conditions into systems of nonlinear equations, using barrier functions for constraints, and a homotopy to drive the barrier to converge to the indicator. They work amazingly well. A good path here is to implement interior point methods for least squares with general inequality constraints and then compare to something simple like projected gradient descent for a box-constrained least squares problem to see if you can get that Newton behavior. If you are more ambitious, you can also use dual representations to implement an interior point method for the Lasso problem.

2 Propose your own project

Please feel completely free to propose your own project that you're interested in and we'll be happy to support you. The only requirements are (1) that it has to have something to do with or use optimization or root finding (both broadly defined) and (2) that you have something to implement, that is, it is not a purely theoretical work. Basically, if you like it and we don't know about it, we want to see what it does on an example. Here are some example projects along these lines:

1. Give an overview and implementation with some examples of any optimization algorithms we didn't have time to cover in class in detail, including quasi-newton methods like BFGS and L-BFGS, Gauss-Newton/Levenberg-Marquardt methods, penalty methods, Fletcher's dog-leg method, the dual bunny-hop,etc.

2. We've generally assumed that we have analytic expressions for gradients, or computed gradients using e.g. finite differences. In practice, people often algorithmically determine gradients using tools like automatic differentiation (AD) or the adjoint state method. Another good project would be to cover one of these approaches and implement an example.

3. How do we perform optimization in infinite dimensional function spaces? Write about something you find interesting in the calculus of variations. Make sure you have an implementable example to show the motivation and the performance of the method.

4. Cast a problem that you've seen in your own research or are generally interested in as an optimization problem. Use a good method to solve a simple instance and explain the performance and impact on the original question.