Maxwell's Scalable semi-supervised learning contributions

October 18, 2022

1 Codebase

1. Created codebase from very little starting code in order to handle a wide range of applications
2. Creating datasets(create_datap̈y):
   - added code to create datasets for FashionMNISt and CIFAR in addition to MNIST
   - added ability to choose which layer to pick from for data from the CIFAR dataset after going through a ResNet model adapted for the CIFAR dataset
   - added easy ability to query a dataset with a subset size and quickly retreive said set of subsets with specified size
   - added functionality with 20newsgroups dataset, including removing stopwords and turning each article into vectors with word frequencies, which then used cosine similarity for distance
3. Benchmarking datasets:
   - benchmarked MNISt, FASHIONMNIST, CIFAR datasets across:
     - k values 1 through 15
     - sigma values 2 to 15
     - CG method values 20 to 100
     - sizes 500 to 2000
4. Testing inverse versus CG method (matrixtestp̈y)
   - created custom script that took in a config file with lists for sizes, sigma values, k-nn values, types of inverses, etc and would then save the product of all of these lists w.r.t. accuracy, variance in accuracy, time, and variance in time in a .pkl file.

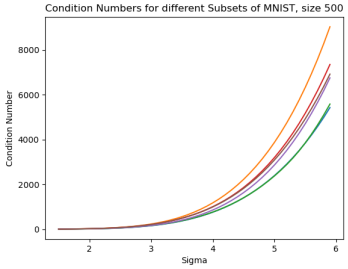
- This is saved as data[label("accuracy", "Time", etc)][inverse_type][size_index, sigma_index, num_CG_iterations_index, k_index,]
- also have functionality for more toy problems like a guassian or cork dataset
- specific function implementations for Knn transform, CG Method, as well as using sparse matrix class to speed up algorithms

5. plotting results (create_plots and Algorithm)
    - Created custom plotting functions. We have (size, sigma, and k value) as possible independent variables, with a dependent variable of time. Allowed for any of the 3 to be on the x axis, with the other 2 either being fixed, or the best values in terms of accuracy being chosen for plotting.
    - also can toggle on and off logarithmic scale and confidence intervals
    - created custom plotting for interval calculation for Algorithm 4 (created intervals using line segments since there were no great methods that did exactly what I wanted)
    - created custom plots for the graph of g_u with derivatives as well as a visualization of descent for a specific instance.
    - verified that condition numbers followed an exponential distribution over multiple subsets
6. Running algorithm 4
    - Created first working version of algorithm 4
    - able to debug algorithm enough to get it to work even for small values of sigma(tinkering with learning rate, doing min, other things specified in the proofs/algorithms section)
    - eventually able to get the algorithhm working with only 5 iterations, in line with other paper resuts for CG method

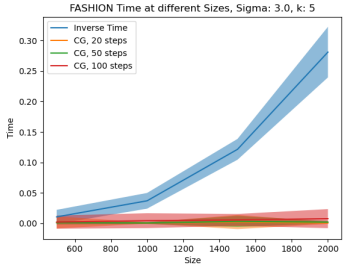
For all code/images, I've shared a folder here. I would definitely encourage looking over all the code, since I expect there is more than you may at first think.

2 Proofs/Algorithms

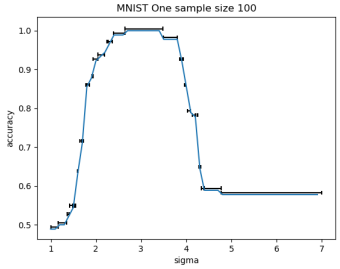
1. Fixed proof of derivative for $\frac{\partial g_u}{\partial \sigma}$ so it would work in practice
2. Had the novel idea to try and use Min(Newton's, GD) to get the algorithm to work in practice (as of now, one of the main ideas in the paper)
3. implemented early stopping in practice to save time when calculating intervals



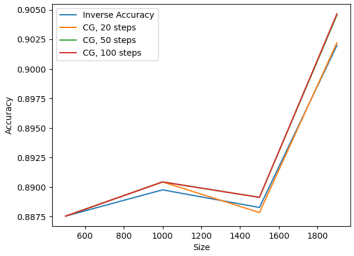
(a) Condition Number Graph



(b) TimeScale Graph w. conf intervals for FashionMNIST



(a) interval data for CG method with 5 iterations



(b) WORDS dataset accuracys (unused)

4. found that we have to make sure that we reach stopping points by checking to make sure g_u is small before stopping the algorithm

5. created updated proof of Algorithm 4 in the approximation case:

    - created proof of CG method in terms of graph. Can be used for regular graphs or to find better bounds for k nearest neighbor graphs
    - proof of convergence for Newton's method
    - 3 separate proofs of convergence for GD (regular GD, strongly convex GD, nesterov GD)
    - proof of convergence of min(Newton's, Gradient Descent) if assumptions from both hold
    - proof of bound for kappa number of given graph family

3 Overviews

1. Anchor point algorithm overview (main results, implementation, complexity analysis)

2. Push relabel algorithm overview (main results, complexity analysis)

3. Sparsifiers

    - Resistance Sparsifiers (main results, complexity)
    - Tree embeddings (main results, complexity analysis)

4. CG Method:

    - General overview
    - preconditioned method, and how to use this method with graph sparsifiers
    - equivalence with label propogation and bounds in terms of kappa number κ

5. Label Propogation/Quadratic criterion (provably correct)

    - Main papers with results
    - different extensions/ versions and why each may be useful

Overall, I've definitely put a lot of time into the project, specifically the codebase and proofs to try and make things work. I've also been succesful in finding new results on my own/continuing the project forward whether that be through experiments or proofs as shown by the examples I've presented. I know that the people that review reccomendations really care about how strong the recommender thinks that the candidate is with respect to other candidates, so I would be super grateful if you could say something about my work being

exceptional or going above and beyond compartively in one area (this may be in converting purely theoretical algorithms to more concrete algorithms, or transfering theory to practice then reproving the theory or something along those lines). Thanks so much for everything so far, and I'm excited to continue with the paper to completion :)