AMSC808N/CMSC828V Maria Cameron

Project 1. Due Thursday, Oct. 8

The goals of this project are

- to acquire some practice of working with real data;
- to explore various optimization methods for solving classification problems and understand how their performance is affected by their settings.

Dataset: USA 2012 and 2016 election data by counties downloaded from here: J. Jia and A. Benson, 2020. For now, we will ignore the graph structure of the dataset. All these data and all my codes are in `Project1.zip`.

I suggest to play with the following classification problem. Assign labels -1 and 1 to counties with majorities of votes given to democrats and republicans respectively. Out of available data for each county (# of Votes, Median Income, Migration Rate, Birth Rate, Death Rate, Bachelor Rate, Unemployment Rate) select any three (so that you can readily visualize the data) that distinguish pro-democratic and pro-republican counties the most. I selected Median Income, log # of Votes, and Bachelor Rate. If you find a better set of three data, please use your choice.

Classification problem: *find an optimal plane* $w^\top x + b$, $w \in \mathbb{R}^3$, $b \in \mathbb{R}$, *dividing pro-democratic and pro-republican counties.* The sense of the word *optimal* will be specified below. If you come up with a more meaningful classification problem, you are welcome to play with it instead.

Choice of programming language. Use a high-level programming language, e.g., Matlab or Python, with which you are the most comfortable. My codes are in Matlab simply because I program a lot in Matlab and do not program in Python (for no good reason). If you go with Python, it shouldn't be difficult to rewrite my codes in it.

What to submit. Please submit one report per working group of 2–3 students with figures and comments. Every group member should link her/his codes to the report pdf. These can be e.g. Dropbox links or GitHub links, etc.

1. **SVN.** First consider a subset of data taken from all 58 CA counties. (In `Project1main.m`, uncomment line 22 and line 27 and comment out lines 40–59.) The initial guess for the dividing plane is obtained by running inexact (due to the use of CG) Newton's method. (Subsampled Inexact Newton's method becomes a regular Inexact Newton's method as $58 < 64$ (64 is my default batch size – see `SINewton.m`). Run the code `Project1main.m` and you obtain `w`. Its first three components make up w and the last one is b. See the next item for more details on it.)

   - Set up the SVN constrained minimization problem with soft margins (Eqs. (52)–(54) in `2-OptProb&Methods4Classification.pdf`) and solve it using the *active-set method* (my routine `ASM` should work just fine on this set). Compare the dividing planes produced by minimizing the loss function and solving the SVN problem with soft margins. Which one looks more reasonable?

- Increase the dataset by adding counties from other states (e.g. OR, WA, ...). Comment on the performance of the SVN. Do your observations motivate you to switch to unconstrained minimization of a reasonable loss function?

2. **SG.** Take the dataset consisting of all pro-democratic counties and an equal number of randomly chosen pro-republican counties. (In `Project1main.m`, lines 22–27 must be commented out and lines 40–59 must be uncommented.) This dataset contains 972 counties. Denote the $n \times 3$ data matrix by X (`XX` in `Project1main.m`), the label vector by y (`label` in `Project1main.m`). Define $Y := (y \cdot 1_{1\times n}) \odot [X, 1_{n\times 1}] \in \mathbb{R}^{n\times 4}$ and $\mathsf{w} := [w; b] \in \mathbb{R}^4$. Then $Y\mathsf{w} \equiv y \odot (X\mathsf{w} + b)$. Consider the following *loss function*:

$$f(\mathsf{w}) := \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-Y_i \mathsf{w})) + \frac{\lambda}{2}\|\mathsf{w}\|^2 \tag{1}$$

where Y_i is the ith row of Y and λ is Tikhonov regularization parameter. The term $\frac{\lambda}{2}\|\mathsf{w}\|^2$ makes the loss function strongly convex. I set $\lambda = 0.01$. Note that $\log(1 + exp(-x))$ is nearly 0 for positive x and grows approximately linearly as $-x$ increases.

Implement the SG algorithm. Experiment with various batch sizes and stepsize decreasing strategies. For each batch size and each stepsize decreasing strategy, accumulate statistics (e.g. do 1000 runs for each) and plot average function value vs iteration number and vs runtime. Comment on how the batch size affects the performance.

Experiment with various values of the parameter λ. Comment on how this parameter affects the results.

3. **Subsampled Inexact Newton.** Experiment with various batch sizes for subsampled inexact Newton's method (you can use my routine `SINewton` or write your own). Plot average function value vs iteration number and vs runtimes. Compare these plots with those for SG. Comment on how these two approaches (SG and Newton) compare.

4. **Stochastic L-BFGS.** Program stochastic L-BFGS method. Experiment with choosing stepsize, batch sizes for the gradient and the pairs $(\mathbf{s}, \mathbf{y})$, and with the frequency of updating the pairs. Set $m = 5$. Compare its performance with stochastic Newton and SG. My codes `LBFGS.m` and `finddirection.m` implementing the regular L-BFGS are added on ELMS to Files/Codes.