Modern Optimization Methods for Big Data Problems
ASSIGNMENT 3: Classification with Smoothed Hinge Loss

DEADLINE: Sunday, March 12, 2016 @ 22:00pm

February 19, 2017

Note: The deadline was extended by 1 week when compared with the original plan.

Instructions: Submit your solution electronically via Learn. Your submission should be a single Julia notebook, i.e., a single .ipynb file. Use a style similar to how the labs are written. Include explanatory text where needed. This is important and will be marked as well. Use any Julia kernel that works for you to produce the file. Start the notebook with a markdown cell containing the text

```
## Assignment 3 -- NAME SURNAME -- UUN
```

Clearly mark the start of each of the 5 problems by inserting a markdown cell with the text

```
## Problem X
```

where $X = 1, 2, \ldots, 5$.

This assignment is one third out of 50% towards your final mark in the course. You may discuss the problems with others if you wish, but you may not copy any code nor solution. Only verbal discussion is allowed.

Background. In this assignment we shall consider the problem of *binary classification*. Consider a collection of data belonging to one of two categories: -1 and 1. For instance, the dataset can be a collection of emails, with 1 denoting spam and -1 denoting ordinary emails. Formally, let us denote the data-label pairs $(A_i, y_i) \in \mathbf{R}^d \times \{-1, 1\}$. As in the lecture, we assume (A_i, y_i) follow some unknown distribution D. The goal of (linear) binary classification is to find a vector $w \in \mathbf{R}^d$ and use it to predict label y_i of a randomly drawn example A_i via the prediction rule

$$y_i \leftarrow \text{sign}(A_i^\top w),$$

where sign is the (slightly modified) sign function: $\text{sign}(t) = 1$ for $t \geq 0$ and $\text{sign}(t) = -1$ for $t < 0$. Hence, if $y_i = 1$, we wish $A_i^\top w$ to be positive, and if $y_i = -1$, we wish $A_i^\top w$ to be negative.

We shall find the classifier w by solving the $L2$-regularized Empirical Risk Minimization (ERM) problem

$$\min_{w \in \mathbf{R}^d} P(w) = \frac{1}{n} \sum_{i=1}^{n} \phi_i(A_i^\top w) + \frac{\lambda}{2} \|w\|_2^2, \tag{ERM}$$

where $\lambda > 0$ is a regularization constant, and $\phi_i : \mathbf{R} \to \mathbf{R}$ is a loss function associated with example i. The most natural loss function in the context of binary classification is the *binary loss*:

$$\phi_i^b(t) = 1 - \text{sign}(y_i t).$$

Indeed, the binary loss is equal to 1 if we predict the wrong label and equal to 0 if we predict the correct label (make sure you understand why). However, this loss function is hard to work with: it is not continuous, nor convex. For this reason, in practice one often uses a different loss function. One popular example is the *hinge loss*:

$$\phi_i^h(t) = \max\{0, 1 - y_i t\}.$$

Hinge loss is both continuous and convex. It can be seen as a convex "approximation" of the binary loss (think why!). However, the hinge loss is not smooth, and hence we can't apply the dfSDCA method to solve ERM. Hence, we shall further replace the hinge loss by a smooth approximation thereof, called *smoothed hinge loss*:

$$\phi_i^{sh}(t) = \begin{cases} 0 & y_i t > 1, \\ 0.5 - y_i t & y_i t < 0, \\ 0.5(1 - y_i t)^2 & \text{otherwise.} \end{cases}$$

It can be shown that this function is convex and 1-smooth. In this assignment we consider (ERM) with smoothed hinge loss.

1. [5 marks] Write function `GenerateData(n,d,theta,a_{+},a_{-},sigma)` which will output matrix $A = [A_1, \ldots, A_n] \in \mathbf{R}^{d \times n}$ and vector $y \in \mathbf{R}^n$ as follows. For each $i = 1, 2, \ldots, n$, A_i is chosen as a random Gaussian vector in $\mathbf{R}^d$ with covariance matrix $\sigma^2 I$ (where I is the $d \times d$ identity matrix) and mean $a \in \mathbf{R}^d$ chosen as follows: $a = a_+$ with probability θ and $a = a_-$ with probability $1 - \theta$. Finally, we set $y_i = +1$ if $a = a_+$ and $y_i = -1$ if $a = a_-$. *Note:* Equivalently, you may think of the j^{th} element of A_i to be a univariate Gaussian random variable with mean a_j (the j^{th} element of a) and variance σ^2.

2. [5 marks] Write function `SmoothedHingeLoss(A,y)` whose input is the matrix $A = [A_1, \ldots, A_n]$ and vector $y = (y_1, \ldots, y_n)^\top$, and whose output are two functions: ii) `SHLgrad(i,w)`, outputting the derivative of $t \to \phi_i^{sh}(t)$ evaluated at $A_i^\top w$, and ii) `P(w)`, evaluating the objective value of (ERM).

3. [5 marks] Write function `SGD(h,N,w0)` which implements standard stochastic gradient descent (SGD) method with constant stepsize h, applied to $L2$-regularized ERM problem with smoothed hinge loss, started from `w0`$=w^0$, and running for N iterations. Standard SGD is the method

$$w^{k+1} = w^k - h^k \left(\nabla \phi_i^{sh}(A_i^\top w^k) A_i + \lambda w^k \right),$$

   where i is chosen uniformly at random, and $h^k > 0$ is a stepsize. The function should use the function `SHLgrad(i,w)` inside.

   Run this method on a random problem with $d = 2$, $n = 10^3$, $\lambda = 1/n$ and $N = 10n$. Use $w^0 = 0$ and experiment with stepsize selection. Report briefly on your findings.

4. [5 marks] Write function `dfSDCA(N,alpha0)` which implements the dfSDCA method, where `alpha0` $= \alpha^0 = (\alpha_1^0, \ldots, \alpha_n^0) \in \mathbf{R}^n$ is the starting "dual" iterate, and N is the number of iterations. Use serial uniform sampling, i.e., $\text{Prob}(\hat{S} = \{i\}) = 1/n$ for all i.

   Compare `dfSDCA` and `SGD` (with a selection of stepsizes) on a number of randomly generated datasets (using the function developed in Problem 1) with $d = 2$, $n = 10^3$, $\lambda = 1/n$ and $N = 10n$. Use $\alpha_i^0 = 0$ for all i. Use a carefully chosen plot to do this comparison. It's up to you to decide what to plot, but the conclusions you make based on the results must make sense. Briefly describe your findings.

5. *[5 marks] Implement a minibatch version of dfSDCA with τ-nice sampling; call the function `mdfSDCA`. Compare it with `dfSDCA`: it is up to you how you make this comparison, but it should be fair and clear, and it should make sense. After this, try to come up with *any method* which also samples τ examples in each iteration, as `mdfSDCA`, but which works better than `mdfSDCA`. You can use any trick or tricks you wish to make the new method work better (e.g., importance sampling of sets of size τ, more agressive stepsizes, changing some other details of the method, and so on). This second part of the problem is open ended. Try to be creative.