Modern Optimization Methods for Big Data Problems
ASSIGNMENT 2: Randomized Coordinate Descent

DEADLINE: Saturday, February 26, 2016 @ 22:00pm

February 5, 2017

Note: The deadline was extended by 1 week (from Feb 19, as originally planned, to Feb 26). This is because we will cover the material you need to know to solve the problems on Wednesday 8.2 and on Monday 13.2).

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 2 -- NAME SURNAME -- UUN
```

Clearly mark the start of each of the 4 problems by inserting a markdown cell with the text

```
## Problem X
```

where $X = 1, 2, 3, 4$.

This assignment is worth 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. Consider the problem

$$\min_{x\in\mathbf{R}^n} f(x), \qquad \text{where} \qquad f(x) = \left(\sum_{j=1}^{d} \phi_j^*(b_j^T x)\right) + \gamma\|x\|_2^2, \tag{*}$$

and where $b_j \in \mathbf{R}^n$ for all $j = 1, 2, \ldots, d$ and $\gamma > 0$. Assume that $n \gg d$. By $\phi_j^* : \mathbf{R} \to \mathbf{R}$ we denote the Fenchel (convex) conjugate of the function $\phi_j : \mathbf{R} \to \mathbf{R}$ defined by

$$\phi_j(t) = |t| + \frac{\alpha_j}{2}t^2,$$

where $\alpha_j > 0$. That is (recall the definition from the ERM lecture), $\phi_j^*(s) = \sup_{t\in\mathbf{R}} st - \phi_j(t)$. It can be shown by direct computation that ϕ_j^* can be written explicitly as

$$\phi_j^*(s) = \begin{cases} \frac{(s+1)^2}{2\alpha_j} & \text{if} \quad s < -1 \\ 0 & \text{if} \quad s \in (-1, 1) \\ \frac{(s-1)^2}{2\alpha_j} & \text{if} \quad s > 1 \end{cases}.$$

It can also be easily verified that ϕ_j^* is $1/\alpha_j$-smooth.

1. [10 marks] Let $B = [b_1, \ldots, b_d] \in \mathbf{R}^{n\times d}$. Write function `ComputeESO(B,alpha,gamma,tau)` with output being a vector $v \in \mathbf{R}^n$ such that $(f, \hat{S}) \sim ESO(v)$, where the first three input parameters define f (`B` is a matrix, as above, and `alpha` is a vector), and $\hat{S}$ is the τ-nice sampling. Justify your code verbally, referring to results from class.

(a) First make sure the code works correctly for $\tau = 1$.

(b) Then come up with a way to handle the $\tau > 1$ case.

(c) Then makes sure you have as tight (i.e., small) parameters v as possible (recall that we have given multiple formulae) for v (some give better results than others).

2. [5 marks] Test your code from part (a) numerically for $\tau = 1$ and $\tau = 2$. That is, numerically test that the ESO inequality holds. Recall that the ESO inequality involves an expectation. If you can, directly evaluate the expectation (if you can't, approximate the expectation through Monte Carlo simulation). Explain verbally what you are doing - justify your test code. Call the code `ESO_Test(B,alpha,gamma,tau`. It's enough to test your code on a single random instance of (*), i.e., for a single combination of B, α, γ. Choose

$$d = 10, \quad n = 1000, \quad \gamma = 1/d, \quad b_j \sim N(0, I), \quad \text{and} \quad \alpha_j = n/j \quad (\text{for all } j).$$

The notation $N(0, I)$ refers to standard Gaussian vector (i.e., a random normal vector with mean 0 and identity covariance matrix).

3. [5 marks] Let

$$\Omega(\tau) = \max_i \frac{v_i}{p_i \lambda}$$

be the leading constant in the complexity of NSync, when applied to (*), assuming f is λ-strongly convex, and when the τ-nice sampling is used.

(a) Write function `PlotSpeedup` (you should decide on the input and output), which plots the function

$$Speedup(\tau) \stackrel{\text{def}}{=} \frac{\Omega(\tau)}{\Omega(1)}.$$

This is the theoretical speedup obtained by "parallelization" (i.e., by using NSync with τ-nice sampling as opposed to NSync with 1-nice sampling).

(b) Test your function on an array of hand-crafted problems of form (*). You can test on random problems, sparse problems – generate the date in any way you like. Explain your findings.

*(c) Can you find data (i.e., B, α and λ) for which the speedup will be nearly linear for all $\tau \in [1, n]$?

4. [5 marks] Write function

`NSyncMinibatch(B,alpha,gamma,tau,x0,k)`

which applies the NSync algorithm with τ-nice sampling to (*) initiated from the starting point `x_0`, producing iterates $x^0, x^1, x^2, \ldots, x^k$, stops at iteration T and outputs x^k and $f(x^k)$.

If you can't do the τ-nice version of NSync, do the 1-nice version (this is worth less marks).

(a) Test the function on random problems with the same parameters as specified in problem 2. Use $x^0 = (10, 10, \ldots, 10) \in \mathbf{R}^n$ and several choices of τ. Display meaningful plots, using meaningful values of k.

(b) Explain in words what your results show.

(c) What is the optimal solution of (*)? Explain through experiments and also theoretically.