Optimization for Machine Learning (Final Exam)

Assignment date: Nov 24
Due date: Dec 12 (noon)

1. (6 points) Consider the following finite sum optimization problem

$$\frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - w^\top x_i y_i)^2 + \frac{\lambda}{2}\|w\|_2^2,$$

where $x_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$.

- (1 points) Derive the dual formulation, as in the SDCA procedure.
  Solution. We note that $f_i(u) = (1 - uy_i)_+^2$, and $f_i^*(-\alpha_i) = -\alpha_i y_i + \alpha_i^2/4$, subject to $\alpha_i y_i \geq 0$. The dual is
  $$\frac{1}{n}\sum_{i=1}^{n} - \left[-\alpha_i y_i + 0.25\alpha_i^2\right] - \frac{1}{2\lambda n^2}\left\|\sum_{i=1}^{n} x_i \alpha_i\right\|_2^2 \qquad (\alpha_i y_i \geq 0).$$
  □
- (1 point) Write down the formula for closed form solution for SDCA (Option I of Algorithm 11.1).
  Solution.
  $$\Delta\alpha_i y_i = \max\left(-\alpha_i^{(t-1)} y_i, \frac{1 - (w^{(t-1)})^\top x_i y_i}{0.5 + \|x_i\|_2^2/(\lambda n)}\right).$$
  □
- (1 point) Write down the dual free SDCA update rule for $\Delta\alpha_i$ in Algorithm 14.3.
  Solution.
  $$\Delta\alpha_i = \eta(2\max(0, 1 - (w^{(t-1)})^\top x_i y_i) y_i - \alpha_i^{(t-1)}).$$
  □
- (1 point) Write down the SGD update rule.
  Solution.
  $$w^{(t)} = w^{(t-1)} + 2\eta \max(0, 1 - (w^{(t-1)})^\top x_i y_i) x_i y_i.$$
  □
- (2 points) Implement the three methods (SDCA, dual-free SDCA, SGD) in prob1() of prog-template.py, and plot the convergence curves (wrt primal-suboptimality) until SDCA converges (error $< 10^{-10}$).

2. (6 points) Consider the minimax problem:

$$\min_x \max_{y \in C}\left[x^\top A y + b^\top y + \frac{1}{2}\|x\|_2^2\right], \qquad C = \{y : y_j \geq 0\}.$$

- (2 points) Write down the optimal solution of x as a function of y. Write the optimization problem in terms of y by eliminating x. Explain the derivations.

  Solution. Using minimax theorem, we have

$$\begin{aligned}&\min_x \max_{y \in C} \left[x^\top Ay + b^\top y + \frac{1}{2}\|x\|_2^2\right]\\ =&\max_{y \in C} \min_x \left[x^\top Ay + b^\top y + \frac{1}{2}\|x\|_2^2\right] = \max_{y \in C}\left[b^\top y - \frac{1}{2}\|Ay\|_2^2\right].\end{aligned}$$

  Here the solution of x is given by $x = Ay$. □

- (1 points) Write down the GDA update rule for this problem with learning rate η. **Solution.** GDA:

$$x \leftarrow x - \eta(x + Ay), \quad y \leftarrow \max(0, y + \eta(b + A^\top x)).$$

  □

- (3 points) Implement GDA, extra gradient, optimistic GDA in prob2() of prog-temp.py, and plot the convergence curves (wrt gradient norm of x and y; 2-norm of $x - Ay$; $by - \frac{1}{2}\|Ay\|_2^2$) for 100 iterations.

3. (6 points) Consider zero-th order optimization.

- (2 points) In Theorem 18.6, if we further assume that $f(x)$ is λ strongly convex concave, and take $\eta_t = (\lambda t)^{-1}$. Derive the corresponding convergence result.

  Solution. The result follows that of SGD for nonsmooth and strongly convex problem. It is

$$\mathbb{E}f(\bar{x}_T) \le f(x) + cdG^2\frac{\ln T + 1}{2\lambda T} + \sigma G.$$

  □

- (2 points) the optimization problem

$$\min_\theta \mathbb{E}_{x \sim \pi(x|\theta)} f(x),$$

  where $x \in \mathbb{R}^d$. Assume we want to solve this problem using policy gradient, with $\theta = (\mu, \rho)$, where μ is d-dimensional vector, and $\rho \in \mathbb{R}$. Both are part of model parameters. Consider distribution $\pi(x|\theta) = N(\mu, e^{-\rho}I)$. Derive the policy gradient update rule for θ including both (μ, ρ).

  Solution. We have

$$\ln \pi(x|\theta) = -\frac{e^\rho\|x-\mu\|_2^2}{2} + \frac{d\rho}{2} - \frac{d}{2}\ln(2\pi).$$

  It follows that

$$\nabla_\mu \ln \pi(x|\theta) = e^\rho(x - \mu), \qquad \nabla_\rho \ln \pi(x|\theta) = 0.5\left[d - e^\rho\|x-\mu\|_2^2\right].$$

  We can use update rule as follows. Draw $z \sim N(\mu, e^{-\rho}I)$:

$$\begin{aligned}\mu \leftarrow&\mu - \eta[f(z) - f(\mu)]e^\rho(z - \mu)\\ \rho \leftarrow&\rho - 0.5\eta[f(z) - f(\mu)]\left[d - e^\rho\|z-\mu\|_2^2\right].\end{aligned}$$

  □

- (2 points) Consider the zero-th order optimization problem over discrete set $x \in \{0,1\}^d$. Implement policy gradients in Example 18.10 and Example 18.11 to solve the objective function

$$\min_{x \in \{0,1\}^d} f(x), \qquad f(x) = \left[\frac{1}{2}x^\top Ax - b^\top x + c\right]$$

  on prob3() of prog-template.py, plot convergence curves (wrt $f(x)$) and report your x_*, θ_* (refer to the Example, $p(x_i = 1) = \theta_i$).

4. (6 points) Consider the setting of decentralized computing, where we are given m nodes. A vector $x = [x_1, \dots, x_m]$ has m components, and each node contains a local component x_i of the vector, with local objective function $f_i(x_i) + g_i(x_i)$. At any time step, in addition to local algebraic operations, we can perform the following function calls simultaneously on all nodes:

   - (gradient computation) call grad(x): each node computes the local gradient $\nabla f_i(x)$.
   - (proximal mapping) call prox(η, z): each node computes the local proximal mapping
   $$\arg\min_{u_i}[0.5\|u_i - z_i\|_2^2 + \eta g_i(u_i)].$$
   - (communication) call communicate(z) $= [(z_{i-1} + z_{i+1})/2]_{i=1,\dots,m}$: each node sends its local vector z_i over the network, then it receives vectors from the neighboring nodes $i-1$ and $i+1$ via the network, and computes the average $(z_{i-1} + z_{i+1})/2$ (where $z_0 = z_m$ and $z_{m+1} = z_1$).

   If we have a variable $w = [w_1, \dots, w_m]$, with w_i stored on node i, then node $j \neq i$ cannot access the information w_i on node i directly, except through calling communicate(). We want to use the above function calls to jointly optimize the following objective function:
   $$\sum_{i=1}^{m}[f_i(x_i) + g_i(z_i)] \qquad x_1 = x_2 = \cdots = x_m = z_1 = \cdots = z_m,$$
   by rewriting the above problem as
   $$f(x) + g(z) \qquad \begin{bmatrix} 0 \\ I \end{bmatrix} x - \begin{bmatrix} B \\ I \end{bmatrix} z = 0,$$
   with $[Bz]_i = z_i - (z_{i-1} + z_{i+1})/2$, $f(x) = \sum_i f_i(x_i)$, $g(z) = \sum_i g_i(z_i)$.

   - (3 points) Write down an algorithm for decentralized optimization using linearized ADMM. Try to combine redundant communications so that no more than two communication calls are needed for each gradient computation.

     Solution. The algorithm is given as follows.

     Algorithm 1: Stochastic Accelerated Linearized ADMM

     Input: $\phi(\cdot)$, A, B, c, $\{\eta_t\}$, β, ρ, α_0, w_0, z_0

     Output: w_T, z_T, α_T

     for $t = 1, 2, \dots, T$ **do**
     - Let $\tilde{z}_t = z_{t-1} + \eta_t \bar{\beta}_{t-1} + \eta_t[\alpha_{t-1} + \rho(w_{t-1} - z_{t-1})]$
     - Let $z_t = \arg\min_z[0.5\|z - \tilde{z}_t\|_2^2 + \eta_t g(z)]$
     - Let $w_t = w_{t-1} - \eta_t \nabla f(w_{t-1}) - \eta_t[\alpha_{t-1} + \rho(w_{t-1} - z_t)]$
     - Let $\bar{z}_t = B^2 z_t$
     - Let $\beta_t = \beta_{t-1} - \rho \bar{z}_t$
     - Let $\bar{\beta}_t = \beta_t - \rho \bar{z}_t$
     - Let $\alpha_t = \alpha_{t-1} + \rho[w_t - z_t]$

     Return: w_T, z_T, α_T

     □

   - (3 points) Implement and plot convergence curves (wrt primal-suboptimality) with different parameters (eta and rho in ADMM) to solve the objective function in prob4() of prog-template.py.