Optimization for Deep Learning (2023 Fall)
Final Exam

Attention:

- Please submit your exam paper (either in LaTeX format or as a picture of your handwritten paper) to `pkudlopt@163.com` by 11:59 pm on January 14. No late submissions will be accepted under any circumstances.
- While preparing your solution, feel free to consult our slides, notes, papers, or any online materials you find helpful. However, you are not allowed to discuss these exam problems with your classmates.
- If you find the description of any problem confusing, please contact the instructor or teaching assistant immediately.

1 Stochastic gradient descent (20 points)

Consider the following stochastic optimization problem:

$$\min_{x\in\mathbb{R}^d} \quad f(x) := \mathbb{E}_{\xi\sim\mathcal{D}}[F(x;\xi)] \tag{1}$$

where $\mathcal{D}$ denotes the data distribution and $\xi \sim \mathcal{D}$ denotes the random data sample. Throughout this problem, we assume $f(x)$ is L-smooth, $F(x;\xi)$ is differentiable in terms of x for any $x \in \mathbb{R}^d$, and the stochastic gradient $\nabla F(x;\xi)$ is unbiased and has bounded variance σ^2, i.e.,

$$\mathbb{E}_{\xi\sim\mathcal{D}}[\nabla F(x;\xi)] = \nabla f(x), \qquad \mathbb{E}_{\xi\sim\mathcal{D}}\|\nabla F(x;\xi) - \nabla f(x)\|^2 \le \sigma^2$$

for any given $x \in \mathbb{R}^d$. Please answer the following questions.

- Write down the stochastic gradient descent (SGD) algorithm to solve problem (1).
- Establish the convergence rate for SGD when $f(x)$ is non-convex.
- Establish the convergence rate for SGD when $f(x)$ is μ-strongly convex.

2 Variance reduction (25 points)

Consider the following problem with finite data samples:

$$\min_{x\in\mathbb{R}^d} \quad f(x) := \frac{1}{M}\sum_{i=1}^{M} F_i(x). \tag{2}$$

SAGA is an effective variance-reduced stochastic algorithm to solve the above problem. Its recursion is listed in Algorithm 1. Please prove that SAGA can converge exponentially fast to the global solution $x^\star$ to problem (2) if each $F_i(x)$ is L-smooth and μ-strongly convex.

Algorithm 1: SAGA Algorithm

Initialization: Initialize x_0 arbitrarily; let $\alpha_j^{(0)} = x_0$ and $u_0 = \sum_{j=1}^{M} \nabla F_j(\alpha_j^{(0)})/M$

for $k = 0, 1, \cdots, T-1$ **do**

- Sample an uniformly random index $i_k \in \{1, \cdots, M\}$;
- Update $g_k = \nabla F_{i_k}(x_k) - \nabla F_{i_k}(\alpha_{i_k}^{(k)}) + u_k$;
- Update $x_{k+1} = x_k - \gamma g_k$;
- Update $\alpha_{i_k}^{(k+1)} = x_k$ and update $\alpha_j^{(k+1)} = \alpha_j^{(k)}$ if $j \neq i_k$;
- Update $u_{k+1} = u_k + (1/M)\big(\nabla F_{i_k}(x_k) - \nabla F_{i_k}(\alpha_{i_k}^{(k)})\big)$;

Return iterates $x_0, x_1, \cdots, x_T$;

3 Dynamic average consensus (25 points)

Consider the following dynamic average consensus recursion:

$$\mathbf{x}_{k+1} = W\left(\mathbf{x}_k + \mathbf{r}_{k+1} - \mathbf{r}_k\right), \quad \forall\ k = 0, 1, 2, \cdots \tag{3}$$

where $\mathbf{r}_k \in \mathbb{R}^n$ is a dynamic input vector observed at iteration k, and $W \in \mathbb{R}^{n\times n}$ is a mixing matrix satisfying

$$W\mathbf{1}_n = \mathbf{1}_n, \quad \mathbf{1}_n^\top W = \mathbf{1}_n^\top, \quad \|W - (1/n)\mathbf{1}_n\mathbf{1}_n^\top\| \le \rho \in (0, 1),$$

where $\mathbf{1}_n = [1; 1; \cdots; 1] \in \mathbb{R}^n$. Moreover, we initialize $\mathbf{x}_0 = \mathbf{r}_0$. Dynamic average consensus can enable each entry of $\mathbf{x}_k$ to track the average of the dynamic input $\mathbf{r}_k$. To see it, please answer the following questions.

- Let $\bar{x}_k = (1/n)\mathbf{1}_n^\top\mathbf{x}_k \in \mathbb{R}$ and $\bar{r}_k = (1/n)\mathbf{1}_n^\top\mathbf{r}_k \in \mathbb{R}$. Prove $\bar{x}_k = \bar{r}_k$ for $k = 1, 2, \cdots$.
- Suppose $\|\mathbf{r}_{k+1} - \mathbf{r}_k\| \le \Delta$, examine the convergence property of $\|\mathbf{x}_k - \bar{r}_k\mathbf{1}_n\|$.
- Suppose $\|\mathbf{r}_{k+1} - \mathbf{r}_k\| \le \alpha^k$ for some $\alpha \in (0, 1)$, examine the convergence property of $\|\mathbf{x}_k - \bar{r}_k\mathbf{1}_n\|$.

4 Improve the performance of FedAvg (30 points)

Suppose N computing nodes collaborate to solve the following distributed **deterministic** optimization problem:

$$\min_{x\in\mathbb{R}^d} \quad f(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(x), \tag{4}$$

where each local cost function $f_i(x)$ is kept by node i. Each node can only access its own local cost function $f_i(x)$ and gradient $\nabla f_i(x)$. The well-known FedAvg algorithm to solve problem (4) in the federated learning setting is given by (each node conducts the following update in parallel)

$$\begin{aligned}
\psi_i^{(k)} &= x_i^{(k)} - \gamma \nabla f(x_i^{(k)}) \\
x_i^{(k+1)} &= \begin{cases} (1/N)\sum_{j=1}^{N} \psi_j^{(k)} & \text{if} \quad \mod (k,\tau) = 0 \\ \psi_i^{(k)} & \text{otherwise} \end{cases}
\end{aligned}$$

where $\tau > 0$ is the communication period and γ is the learning rate. However, FedAvg cannot converge exactly to the solution to problem (4) with constant learning rate. For example, the convergence performance of FedAvg for distributed linear regression problem with constant learning rate is shown in Fig. 1. It is observed that FedAvg oscillates around 5×10^{-3} and cannot converge to a better solution.

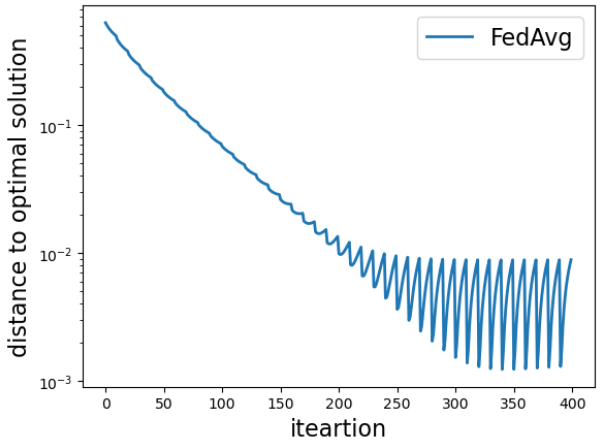

Figure 1: The convergence performance of FedAvg with constant learning rate for distributed linear regression problem.

In this problem, we will develop new algorithms that can correct the bias that FedAvg suffers and make it converge to the **exact solution** rather than the neighborhood around the solution. To this end, we introduce the following notations

$$\begin{aligned}
\mathbf{x} &= [x_1^\top; x_2^\top; \cdots; x_N^\top] \in \mathbb{R}^{N\times d} \\
\mathbf{g} &= [\left(\nabla f_1(x_1)\right)^\top; \left(\nabla f_2(x_2)\right)^\top; \cdots; \left(\nabla f_N(x_N)\right)^\top] \in \mathbb{R}^{N\times d}
\end{aligned}$$

and rewrite FedAvg into a more compact recursion

$$\mathbf{x}^{(k+1)} = W^{(k)}(\mathbf{x}^{(k)} - \gamma \mathbf{g}^{(k)}) \tag{5}$$

with

$$W^{(k)} = \begin{cases} (1/N)\mathbf{1}_N \mathbf{1}_N^\top & \text{if} \quad \text{mod } (k, \tau) = 0 \\ I_N & \text{otherwise} \end{cases}$$

It is observed from recursion (5) that **FedAvg can be viewed as a decentralized gradient descent with a time-varying mixing matrix!** With this critical observation, we can utilize various advanced techniques in decentralized optimization, which can remove the influence of data heterogeneity suffered by decentralized gradient descent, to help improve the performance of FedAvg.

- Utilize the knowledge you learn from decentralized optimization to develop new algorithms that can make FedAvg converge exactly to the solution even with constant learning rate. Write down your new algorithm recursions.

- Read and run the code provided in "Prob4.ipynb" to check the performance of FedAvg. Implement your cool algorithm in "Prob4.ipynb" and compare it with FedAvg. Ensure that your algorithm exhibits exponential convergence to an accuracy superior to 10^{-10} even with a constant learning rate when tackling the distributed linear regression problem. Copy the resulting comparison figure into your solution for this exam paper.