Section 06: Stochastic Gradient Descent

1. Gradient Descent

We will now examine gradient descent algorithm and study the effect of learning rate α on the convergence of the algorithm. Recall from lecture that Gradient Descent takes on the form of $x_{k+1} = x_k - \alpha \nabla f(x_k)$, with an initialization x_0.

(a) Assume that $f\colon \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, and additionally,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \text{ for any } x, y,$$

i.e., ∇f is Lipschitz continuous with constant $L \geq 0$.
Show that: Gradient descent with fixed step size $\eta \leq \frac{1}{L}$ satisfies

$$f(x_k) - f(x^*) \leq \frac{\|x_0 - x^*\|^2}{2\eta k}, \qquad k \in \mathbb{N}.$$

I.e., gradient descent has convergence rate $O\left(\frac{1}{k}\right)$.

Hints:

(i) ∇f is Lipschitz continuous with constant $L \geq 0 \implies f(y) \leq f(x) + \nabla f(x)(y - x) + \frac{L}{2}\|y - x\|^2$ for all x, y.

(ii) f is convex $\implies f(x) \leq f(x^*) + \nabla f(x)(x - x^*)$, where x^* is the local minima that the gradient descent algorithm is converging to.

(iii) $2\eta \nabla f(x)(x - x^*) - \eta^2 \|\nabla f(x)\|^2 = \|x - x^*\|^2 - \|x - \eta \nabla f(x) - x^*\|^2$.

2. Stochastic Gradient Descent

Consider minimizing an average of functions:

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell_i(w),$$

where w is a $d-$dimensional vector (or the feature dimension is d). The minimization of the negative of a log-likelihood function can serve as an example. Recall that the (full) gradient descent step is given by

$$w^{(t+1)} = w^{(t)} - \eta \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_i\big(w^{(t)}\big).$$

The computational cost of a single step here is $\mathcal{O}(dn)$. To reduce cost, one idea is to just use a subset of all samples to approximate the full gradient. Specifically, consider revising the gradient descent step as follows:

$$w^{(t+1)} = w^{(t)} - \eta \cdot \nabla \ell_{I_t}(w^{(t)}),$$

where I_t is chosen randomly within $\{1, 2, ..., n\}$ with equal probability. This is called **stochastic gradient descent (SGD)**, and the computational cost of a single step now reduces to $\mathcal{O}(d)$.

(a) The following two results provide intuitions or foundations for why SGD works.

- $\mathbb{E}_{I_t}\left[\nabla \ell_{I_t}(w^{(t)})\right] = \frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i\big(w^{(t)}\big)$, which is the full gradient. Hence the estimate of gradient is unbiased.
- Let $\ell(w) = \frac{1}{n}\sum_i \ell_i(w)$ and $w^* = \arg\min_w \ell(w)$. Assume $\|w^{(1)} - w^*\|_2^2 \le R$ and $\sup_w \max_i \|\nabla \ell_i(w)\|_2^2 \le G$. Then

$$\mathbb{E}[\ell(\bar{w}) - \ell(w^*)] \le \sqrt{\frac{RG}{T}},$$

where $\bar{w} \coloneqq \frac{1}{T}\sum_{t=1}^{\top} w^{(t)}$. Therefore, the expected error over T iterations is $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$. (The proof of this result is provided in the solution part for reference.)

(b) What disadvantages can SGD have? How can we balance between the noise in updates and computational cost?

3. Extensions of SGD

(a) Gradient descent requires the full gradient when updating while (standard) SGD utilizes the gradient of one sample when updating. **Mini-batching** is somewhere between the two extremes. That is, we choose a random subset $I_t \subseteq \{1, ..., n\}$ with size $|I_t| = b \ll n$ in the stochastic gradient descent step:

$$w^{(t+1)} = w^{(t)} - \eta \cdot \frac{1}{b}\sum_{i_t \in I_t} \nabla \ell_{i_t}(w^{(t)}).$$

With mini-batching, we have the following results:

- $\mathbb{E}_{I_t}\left[\frac{1}{b}\sum_{i_t \in I_t} \nabla \ell_{i_t}(w^{(t)})\right] = \frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(w^{(t)})$: we still have an unbiased estimate of the full gradient.
- Compared to standard SGD, variance of the gradient estimate is reduced approximately by $\frac{1}{b}$.
- Computational cost for each step now becomes $\mathcal{O}(db)$.

Remark: By matrix computations (computing b gradients at a time) and parallelization, we can denoise the estimated gradients without increasing much computational cost (for batch size b that is not large).

(b) How should we choose the batch size?

(c) Are there other extensions or variants of the basic stochastic gradient descent algorithm?