Contents

1

Probability and training algorithms

1.1 Stochastic gradient descent method and convergence theory

The next optimization problem is the most common case in machine learning.

Problem 1.

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1.1}$$

where

$$f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x). \tag{1.2}$$

One version of stochastic gradient descent (SGD) algorithm is:

Algorithm 1 SGD

Input: initialization x_0, learning rate η_t.
For: t = 0,1,2,...
Randomly pick $i_t \in \{1, 2, \cdots, N\}$ independently with probability $\frac{1}{N}$

$$x_{t+1} = x_t - \eta_t \nabla f_{i_t}(x_t). \tag{1.3}$$

1.1.1 Convergence of SGD

Theorem 1. *Assume that each $f_i(x)$ is λ-strongly convex and $\|\nabla f_i(x)\| \le M$ for some $M > 0$. If we take $\eta_t = \frac{a}{\lambda(t+1)}$ with sufficiently large a such that*

$$\|x_0 - x^*\|^2 \le \frac{a^2 M^2}{(a-1)\lambda^2} \tag{1.4}$$

then

$$\mathbb{E}e_t^2 \leq \frac{a^2M^2}{(a-1)\lambda^2(t+1)}, \quad t \geq 1, \tag{1.5}$$

where $e_t = \|x_t - x^*\|$.

Proof. The L^2 error of SGD can be written as

$$\begin{aligned}
\mathbb{E}\|x_{t+1} - x^*\|^2 &\leq \mathbb{E}\|x_t - \eta_t \nabla f_{i_t}(x_t) - x^*\|^2 \\
&\leq \mathbb{E}\|x_t - x^*\|^2 - 2\eta_t \mathbb{E}(\nabla f_{i_t}(x_t) \cdot (x_t - x^*)) + \eta_t^2 \mathbb{E}\|\nabla f_{i_t}(x_t)\|^2 \\
&\leq \mathbb{E}\|x_t - x^*\|^2 - 2\eta_t \mathbb{E}(\nabla f(x_t) \cdot (x_t - x^*)) + \eta_t^2 M^2 \\
&\leq \mathbb{E}\|x_t - x^*\|^2 - \eta_t \lambda \mathbb{E}\|x_t - x^*\|^2 + \eta_t^2 M^2 \\
&= (1 - \eta_t \lambda)\mathbb{E}\|x_t - x^*\|^2 + \eta_t^2 M^2
\end{aligned} \tag{1.6}$$

The third line comes from the fact that

$$\begin{aligned}
\mathbb{E}(\nabla f_{i_t}(x_t) \cdot (x_t - x^*)) &= \mathbb{E}_{i_1 i_2 \cdots i_t}(\nabla f_{i_t}(x_t) \cdot (x_t - x^*)) \\
&= \mathbb{E}_{i_1 i_2 \cdots i_{t-1}} \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x_t) \cdot (x_t - x^*) \\
&= \mathbb{E}_{i_1 i_2 \cdots i_{t-1}} \nabla f(x_t) \cdot (x_t - x^*) \\
&= \mathbb{E}\nabla f(x_t) \cdot (x_t - x^*),
\end{aligned} \tag{1.7}$$

and

$$\mathbb{E}\|\nabla f_{i_t}(x_t)\|^2 \leq \mathbb{E}M^2 = M^2. \tag{1.8}$$

Note when $t = 0$, we have

$$\mathbb{E}e_0^2 = \|x_0 - x^*\|^2 \leq \frac{a^2M^2}{(a-1)\lambda}, \tag{1.9}$$

based on the assumption.

In the case of SDG, by the inductive hypothesis,

$$\begin{aligned}
\mathbb{E}e_{t+1}^2 &\leq (1 - \eta_t\lambda)\mathbb{E}e_t^2 + \eta_t^2 M^2 \\
&\leq (1 - \frac{a}{t+1})\frac{a^2M^2}{(a-1)\lambda^2(t+1)} + \frac{a^2M^2}{\lambda^2(t+1)^2} \\
&\leq \frac{a^2M^2}{(a-1)\lambda^2}\frac{1}{(t+1)^2}(t+1-a+a-1) \\
&= \frac{a^2M^2}{(a-1)\lambda^2}\frac{t}{(t+1)^2} \\
&\leq \frac{a^2M^2}{(a-1)\lambda^2(t+2)}. \quad \left(\frac{t}{(t+1)^2} \leq \frac{1}{t+2}\right)
\end{aligned} \tag{1.10}$$

□

1.1.2 SGD with mini-batch

Firstly, we will introduce a natural extended version of the SGD discussed above with introducing mini-bath.

Algorithm 2 SGD with mini-batch

Input: initialization x_0, learning rate η_t.
For: t = 0,1,2,...

$$\text{Randomly pick } B_t \subset \{1, 2, \cdots, N\} \text{ independently with probability } \frac{1}{\binom{N}{m}} \text{ and } \#B_t = m.$$

$$x_{t+1} = x_t - \eta_t g_t(x_t). \tag{1.11}$$

where

$$g_t(x_t) = \frac{1}{m} \sum_{i \in B_t} \nabla f_i(x_t)$$

Now we introduce the SGD algorithm with mini-batch without replacement which is the most commonly used version of SGD in machine learning.

Algorithm 3 Shuffle SGD with mini-batch

Input: learning rate η_k, mini-batch size m, parameter initialization x_0 and denote $M = \lceil \frac{N}{m} \rceil$.
For Epoch $k = 1, 2, \ldots$

$$\text{Randomly pick } B_i \subset \{1, 2, \cdots, N\} \text{ without replacement with } \#B_i = m \text{ for } i = 1, 2, \cdots, t.$$

For mini-batch $t = 1 : M$
Compute the gradient on B_t:
Update x:

$$x \leftarrow x - \eta_k g_t(x),$$

where

$$g_t(x) = \frac{1}{m} \sum_{i \in B_t} \nabla f_i(x)$$

EndFor
EndFor

To " randomly pick $B_i \subset \{1, 2, \cdots, N\}$ without replacement with $\#B_i = m$ for $i = 1, 2, \cdots, t$", we usually just randomly shuffle the index set first and then consec-

utively pick every m elements in the shuffled index set. That is the reason why we would like to call the algorithm as shuffled SGD while this is the mostly used version of SGD in machine learning.

Remark 1. Let recall a general machine learning loss function

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(h(X_i;\theta), Y_i), \tag{1.12}$$

where $\{(X_i, Y_i)\}_{i=1}^N$ correspond to these data pairs. For example, $\ell(\cdot,\cdot)$ takes cross-entropy and $h(x;\theta) = \boldsymbol{p}(x;\theta)$ as we discussed in Logistic regression section.

Thus, we have the following corresponding relation

$$\begin{aligned} f(x) &\leftrightsquigarrow L(\theta) \\ f_i(x) &\leftrightsquigarrow \ell(h(X_i;\theta), Y_i). \end{aligned}$$