Optimization for Machine Learning
机器学习中的优化方法

陈 程

华东师范大学 软件工程学院

chchen@sei.ecnu.edu.cn

Outline

Stochastic optimization

Stochastic optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) \triangleq \underbrace{\mathbb{E}_\xi[f(\mathbf{x};\xi)]}_{\text{expectation setting}},$$

where the random variable $\xi \sim \mathcal{D}$.

Stochastic gradient descent:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t, \xi_t).$$

Stochastic variance reduced gradient (SVRG)

The finite-sum setting is a special case of the expectation setting:

$$F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}).$$

Stochastic variance reduced gradient (SVRG):

$$\underbrace{\nabla f_i(\mathbf{x}_t) - \nabla f_i(\tilde{\mathbf{x}})}_{\to \mathbf{0} \text{ if } \mathbf{x}_t \approx \tilde{\mathbf{x}}} + \underbrace{\nabla F(\tilde{\mathbf{x}})}_{\to \mathbf{0} \text{ if } \tilde{\mathbf{x}} \approx \mathbf{x}^*}$$

where $\tilde{\mathbf{x}}$ is a history point updated every $O(\kappa)$ rounds.

Iteration complexity

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}).$$

	iteration complexity	per-iteration	total
batch GD	$\kappa \log(1/\epsilon)$	n	$n\kappa \log(1/\epsilon)$
SGD	$1/\epsilon$	1	$1/\epsilon$
SVRG	$\log(1/\epsilon)$	$n+\kappa$	$(n+\kappa)\log(1/\epsilon)$

Table: Convergence rate for the strongly convex case

Stochastic nonconvex optimization

Stochastic nonconvex optimization:

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) \triangleq \mathbb{E}_\xi[f(\mathbf{x};\xi)],$$

where $f(\mathbf{x};\xi)$ is L-smooth and potentially nonconvex.

Our goal is to find a first-order stationary point $\mathbf{x}$ such that

$$\mathbb{E}[\|\nabla F(\mathbf{x})\|_2] \leq \epsilon.$$

Assumption:

$$\mathbb{E}_\xi[\|f(\mathbf{x},\xi) - F(\mathbf{x})\|_2^2] \leq \sigma^2.$$

SGD for nonconvex optimization

Stochastic gradient descent:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t, \xi_t).$$

- Return $\bar{\mathbf{x}}$ chosen uniformly at random from $\{\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}\}$.
- If we choose

$$\eta = \eta_t = \frac{1}{L} \min\left\{\frac{\epsilon^2}{2\sigma^2}, 1\right\} \text{ and } t = \frac{4(F(\mathbf{x}_0) - F(\mathbf{x}^*))}{\epsilon^2 \eta},$$

then

$$\mathbb{E}[\|\nabla F(\bar{\mathbf{x}})\|_2] \leq \epsilon.$$

Stochastic recursive gradient

Stochastic recursive gradient estimates:

$$\mathbf{g}_t = \nabla f_i(\mathbf{x}_t) - \nabla f_i(\mathbf{x}_{t-1}) + \mathbf{g}_{t-1}$$

where i is randomly sampled from $\{1, \ldots, n\}$.

comparison to SVRG (use a fixed snapshot point for the entire epoch)

$$\nabla f_i(\mathbf{x}_t) - \nabla f_i(\tilde{\mathbf{x}}) + \nabla F(\tilde{\mathbf{x}})$$

- Unlike SVRG, $\mathbf{g}_t$ is NOT an unbiased estimator of $\nabla F(\mathbf{x}_t)$.
- We have $\mathbb{E}_t[\mathbf{g}_t - \nabla F(\mathbf{x}_t)] = \mathbf{g}_{t-1} - \nabla F(\mathbf{x}_{t-1})$.
- If we average out all randomness, we have $\mathbb{E}[\mathbf{g}_t] = \mathbb{E}[\nabla F(\mathbf{x}_t)]$.

StochAstic Recursive grAdient algoritHm (SARAH)

Algorithm 1 SARAH

Input: $\mathbf{x}_0$, η, m, S
$\tilde{\mathbf{x}}^{(0)} = \mathbf{x}_0$
for $s = 0, \ldots, S-1$
    $\mathbf{g}_0 = \nabla f(\tilde{\mathbf{x}}^{(s)})$
    $\mathbf{x}_0 = \tilde{\mathbf{x}} = \tilde{\mathbf{x}}^{(s)}$
    for $t = 0, \ldots, m-1$
        draw i_t from $\{1, \ldots, n\}$ uniformly
        $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \mathbf{g}_t$
        $\mathbf{g}_{t+1} = \nabla f_{i_t}(\mathbf{x}_{t+1}) - \nabla f_{i_t}(\mathbf{x}_t) + \mathbf{g}_t$
    end for
    $\tilde{\mathbf{x}}^{(s+1)} = \mathbf{x}_t$ for randomly chosen $t \in \{0, \ldots, m-1\}$
end for
Output: $\tilde{\mathbf{x}}^{(S)}$

Convergence rates for finite-sum setting

Method	Complexity
GD	$n\kappa \log(1/\epsilon)$
SGD	$1/\epsilon$
SVRG	$(n+\kappa)\log(1/\epsilon)$
SARAH [1]	$(n+\kappa)\log(1/\epsilon)$

Table: Convergence rate for the strongly convex case

Method	Complexity
GD	n/ϵ
SGD	$1/\epsilon^2$
SVRG	$(n+\sqrt{n}/\epsilon)$
SARAH [1]	$(n+1/\epsilon)\log(1/\epsilon)$

Table: Convergence rate for the smooth and convex case

Convergence Rates for Finite-sum Setting

Method	Complexity
GD	n/ϵ^2
SGD	$1/\epsilon^4$
SVRG [2]	$(n + n^{2/3}/\epsilon^2)$
SARAH [3]	$(n + \sqrt{n}/\epsilon^2)$

Table: Convergence rate for the smooth and nonconvex case

Outline

Momentum SGD

Momentum variant of SGD (Polyak, 1964):

$$
\begin{aligned}
&\text{pick a stochastic gradient } \mathbf{g}_t \\
&\mathbf{m}_t = \beta \mathbf{m}_{t-1} + (1-\beta)\mathbf{g}_t \quad \text{(momentum term)} \\
&\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{m}_t
\end{aligned}
$$

- is the stochastic variant of heavy-ball method
- key element of deep learning optimizers

Adagrad

Adagrad is an adaptive variant of SGD

$$
\begin{aligned}
&\text{pick a stochastic gradient } \mathbf{g}_t \\
&\mathbf{r}_t = \mathbf{r}_{t-1} + \mathbf{g}_t \odot \mathbf{g}_t \\
&\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{\eta_t}{\delta + \sqrt{\mathbf{r}_t}} \odot \mathbf{g}_t
\end{aligned}
$$

- chooses an adaptive, coordinate-wise learning rate
- variants: Adadelta, Adam, RMSprop,...

RMSprop

RMSprop is a moving average variant of AdaGrad

$$
\begin{aligned}
&\text{pick a stochastic gradient } \mathbf{g}_t \\
&\mathbf{r}_t = \beta \mathbf{r}_{t-1} + (1-\beta)\mathbf{g}_t \odot \mathbf{g}_t \\
&\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{\eta_t}{\delta + \sqrt{\mathbf{r}_t}} \odot \mathbf{g}_t
\end{aligned}
$$

- faster forgetting of older weights

Adam

Adam is a momentum variant of RMSprop

$$
\begin{aligned}
&\text{pick a stochastic gradient } \mathbf{g}_t \\
&\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t \quad \text{(momentum term)} \\
&\mathbf{r}_t = \beta_2 \mathbf{r}_{t-1} + (1-\beta_2)\mathbf{g}_t \odot \mathbf{g}_t \\
&\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{\eta_t}{\delta + \sqrt{\mathbf{r}_t}} \odot \mathbf{m}_t
\end{aligned}
$$

- strong performance in practice, e.g. for self-attention networks
- may not converge in some special cases, see [4]

Outline

Minimax optimization

Minimax optimization:

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x}, \mathbf{y})$$

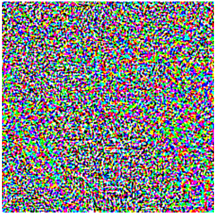
Applications:

- Adversarial learning
- Generative Adversarial Network (GAN)
- Two-player games

Examples: adversarial learning

“panda”
57.7% confidence

$+ .007 \times$



noise

$=$

“gibbon”
99.3 % confidence

Examples: adversarial learning

In supervised learning, we consider

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \triangleq \frac{1}{n}\sum_{i=1}^{n} l(\mathbf{x}; \mathbf{a}_i, b_i) + \lambda R(\mathbf{x}).$$

In adversarial training, we use a perturbed $\mathbf{y}_i$ for each data $\mathbf{a}_i$.

It leads to the following minimax optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} \max_{\mathbf{y}_i\in\mathcal{Y}_i, i=1,\ldots,n} \tilde{f}(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_n) \triangleq \frac{1}{n}\sum_{i=1}^{n} l(\mathbf{x}; \mathbf{y}_i, b_i) + \lambda R(\mathbf{x}),$$

where $\mathcal{Y}_i = \{\mathbf{y} : \|\mathbf{y} - \mathbf{a}_i\| \leq \delta\}$ for some small $\delta > 0$.

Examples: generative adversarial network (GAN)

Given n data samples $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^d$ from an unknown distribution, GAN aims to generate additional samples with the same distribution as the observed samples.

We formulate the minimax optimization problem

$$\min_{\mathbf{w}\in\mathcal{W}} \max_{\boldsymbol{\theta}\in\Theta} \ \frac{1}{n}\sum_{i=1}^{n} \log D(\boldsymbol{\theta}, \mathbf{a}_i) + \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\big[\log(1 - D(\boldsymbol{\theta}, G(\mathbf{w}, \mathbf{z})))\big].$$

1. $D(\boldsymbol{\theta}, \cdot)$ is the discriminator that tries to separate the generated data $G(\mathbf{w}; \mathbf{z})$ from the real data samples $\mathbf{a}_i$
2. $G(\mathbf{w}, \cdot)$ is the generator that tries to make $D(\boldsymbol{\theta}, \cdot)$ cannot separate the distributions of $G(\mathbf{w}; \mathbf{z})$ and $\mathbf{a}_i$

Examples: two-player games

Consider the payoff matrix of rock-paper-scissor:

$$
\begin{array}{cccc}
 & \text{rock} & \text{paper} & \text{scissor} \\
\text{rock} & 0 & 1 & -1 \\
\text{paper} & -1 & 0 & 1 \\
\text{scissor} & 1 & -1 & 0
\end{array} = \mathbf{A}
$$

The two-player rock-paper-scissor games aim to optimize:

$$\min_{\mathbf{x} \in \mathcal{X}} \max_{\mathbf{y} \in \mathcal{Y}} \mathbf{x}^\top \mathbf{A} \mathbf{y}$$

- Pure strategy: $\mathcal{X} = \mathcal{Y} = \{e_1, e_2, e_3\}$, not a convex set
- Mixed strategy: $\mathcal{X} = \mathcal{Y} = \Delta$, simplex over 3 dimension

Properties of minimax optimization

In general, we have

$$\max_{\mathbf{y}\in\mathcal{Y}} \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x},\mathbf{y}) \le \min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y})$$

Von Neumann's Minimax Theorem. If both $\mathcal{X}$ and $\mathcal{Y}$ are compact convex sets, and $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is convex-concave, then

$$\max_{\mathbf{y}\in\mathcal{Y}} \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x},\mathbf{y}) = \min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y})$$

Convex-concave optimization

We measure the optimality of $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ in terms of the **duality gap**:

$$gap \triangleq \max_{\mathbf{y}\in\mathcal{Y}} f(\hat{\mathbf{x}}, \mathbf{y}) - \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}, \hat{\mathbf{y}}) \geq 0$$

Review:

- f is L-Lipschitz if $|f(\mathbf{z}_1) - f(\mathbf{z}_2)| \leq L \|\mathbf{z}_1 - \mathbf{z}_2\|_2$.
- f is ℓ-smooth if $\|\nabla f(\mathbf{z}_1) - \nabla f(\mathbf{z}_2)\|_2 \leq \ell \|\mathbf{z}_1 - \mathbf{z}_2\|_2$.

Gradient descent ascent (GDA)

Projected gradient descent ascent:

$$
\begin{aligned}
\tilde{\mathbf{x}}_{t+1} &= \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t) \\
\tilde{\mathbf{y}}_{t+1} &= \mathbf{y}_t + \eta \nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t) \\
\mathbf{x}_{t+1} &= \mathcal{P}_{\mathcal{X}}(\tilde{\mathbf{x}}_{t+1}) \\
\mathbf{y}_{t+1} &= \mathcal{P}_{\mathcal{Y}}(\tilde{\mathbf{y}}_{t+1})
\end{aligned}
$$

Convergence rates of GDA

If f is L-Lipschitz and convex-concave, let the diameter of $\mathcal{X}$ and $\mathcal{Y}$ be R. Then for fixed t with learning rate $\eta = \frac{R}{L\sqrt{t}}$, we have

$$\max_{\mathbf{y}\in\mathcal{Y}} f\left(\frac{1}{t}\sum_{k=1}^{t}\mathbf{x}_k, \mathbf{y}\right) - \min_{\mathbf{x}\in\mathcal{X}} f\left(\mathbf{x}, \frac{1}{t}\sum_{k=1}^{t}\mathbf{y}_k\right) \leq \frac{2LR}{\sqrt{t}}.$$

If f is ℓ-smooth and convex-concave, let the diameter of $\mathcal{X}$ and $\mathcal{Y}$ be R. Then for fixed t with $\eta = \frac{R}{L\sqrt{t}}$ where $L = 2\ell R + \|\nabla f(\mathbf{x}_0, \mathbf{y}_0)\|_2$, we have

$$\max_{\mathbf{y}\in\mathcal{Y}} f\left(\frac{1}{t}\sum_{k=1}^{t}\mathbf{x}_k, \mathbf{y}\right) - \min_{\mathbf{x}\in\mathcal{X}} f\left(\mathbf{x}, \frac{1}{t}\sum_{k=1}^{t}\mathbf{y}_k\right) \leq \frac{2LR}{\sqrt{t}}.$$

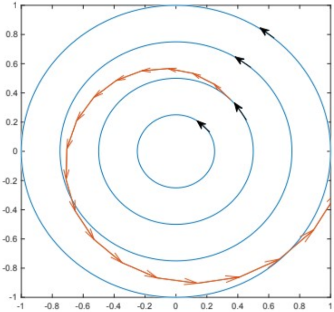
Slower than minimization problem!

GDA does not have last iterate guarantees

Consider following problem:

$$\min_{x \in [-1,1]} \max_{y \in [-1,1]} xy$$

- The optimal point is $(0, 0)$.
- GDA will diverge for unconstrained case or hit the boundary for constrained case.



References

[1] “SARAH: A novel method for machine learning problems using stochastic recursive gradient,” L. Nguyen, J. Liu, K. Scheinberg, M. Takac, ICML 2017.

[2] “Stochastic variance reduction for nonconvex optimization,” S. Reddi, A. Hefny, S. Sra, B. Poczos, A. Smola, ICML 2016.

[3] “Finite-Sum Smooth Optimization with SARAH,” L. Nguyen, M. Dijk, D. Phan, P. Nguyen, T. Weng, J. Kalagnanam, Computational Optimization and Applications.

[4] “On the Convergence of Adam and Beyond.”, S. Reddi, S. Kale, S. Kumar, ICLR 2018.