

Stochastic Gradient Descent (2)

Problem Formulation

Consider minimizing an average of functions,

$$\min_x \frac{1}{n} \sum_{i=1}^n f_i(x)$$

where f is convex and differentiable, $\text{dom}(f) = \mathbb{R}^d$.

Review

Stochastic Gradient Descent (SGD)

- Initialize $x^{(0)} \in \mathbb{R}^d$
- Repeat:

$$x^{(k)} = x^{(k-1)} - t \cdot \nabla f_{i_k} \left(x^{(k-1)} \right), \quad k = 1, 2, 3, \dots$$

where t is the step size, $i_k \in \{1, \dots, n\}$ is an index randomly selected at iteration k .

Mini-batch Stochastic Gradient Descent (Mini-batch)

- Initialize $x^{(0)} \in \mathbb{R}^d$
- Repeat:

$$x^{(k)} = x^{(k-1)} - t \cdot \frac{1}{b} \sum_{i \in I_k} \nabla f_i \left(x^{(k-1)} \right), \quad k = 1, 2, 3, \dots$$

where $I_k \subseteq \{1, \dots, n\}$ is a subset of size $|I_k| = b \ll n$ randomly selected at iteration k .

Stochastic Average Gradient (SAG)^{[1](#)}

- Initialize $x^{(0)}$ and $g_i^{(0)} = \nabla f_i \left(x^{(0)} \right), i = 1, \dots, n$
- At steps $k = 1, 2, 3, \dots$, pick random $i_k \in \{1, \dots, n\}$, then let

$$g_{i_k}^{(k)} = \nabla f_{i_k} \left(x^{(k-1)} \right)$$

Set all other $g_i^{(k)} = g_i^{(k-1)}, i \neq i_k$.

- Update

$$x^{(k)} = x^{(k-1)} - t \cdot \frac{1}{n} \sum_{i=1}^n g_i^{(k)}$$

SAGA²

- Initialize $x^{(0)}$ and $g_i^{(0)} = \nabla f_i \left(x^{(0)} \right), i = 1, \dots, n$
- At steps $k = 1, 2, 3, \dots$, pick random $i_k \in \{1, \dots, n\}$, then let

$$g_{i_k}^{(k)} = \nabla f_{i_k} \left(x^{(k-1)} \right)$$

Set all other $g_i^{(k)} = g_i^{(k-1)}, i \neq i_k$.

- Update

$$x^{(k)} = x^{(k-1)} - t \cdot \left(g_{i_k}^{(k)} - g_{i_k}^{(k-1)} + \frac{1}{n} \sum_{i=1}^n g_i^{(k-1)} \right)$$

Stochastic Variance Reduced Gradient (SVRG)³

- Initialize $\hat{x}^{(0)} \in \mathbb{R}^d$
- At step $k = 1, 2, 3, \dots$,

- Set:

$$\hat{x} = \hat{x}^{(k-1)}, x_0 = \hat{x}$$

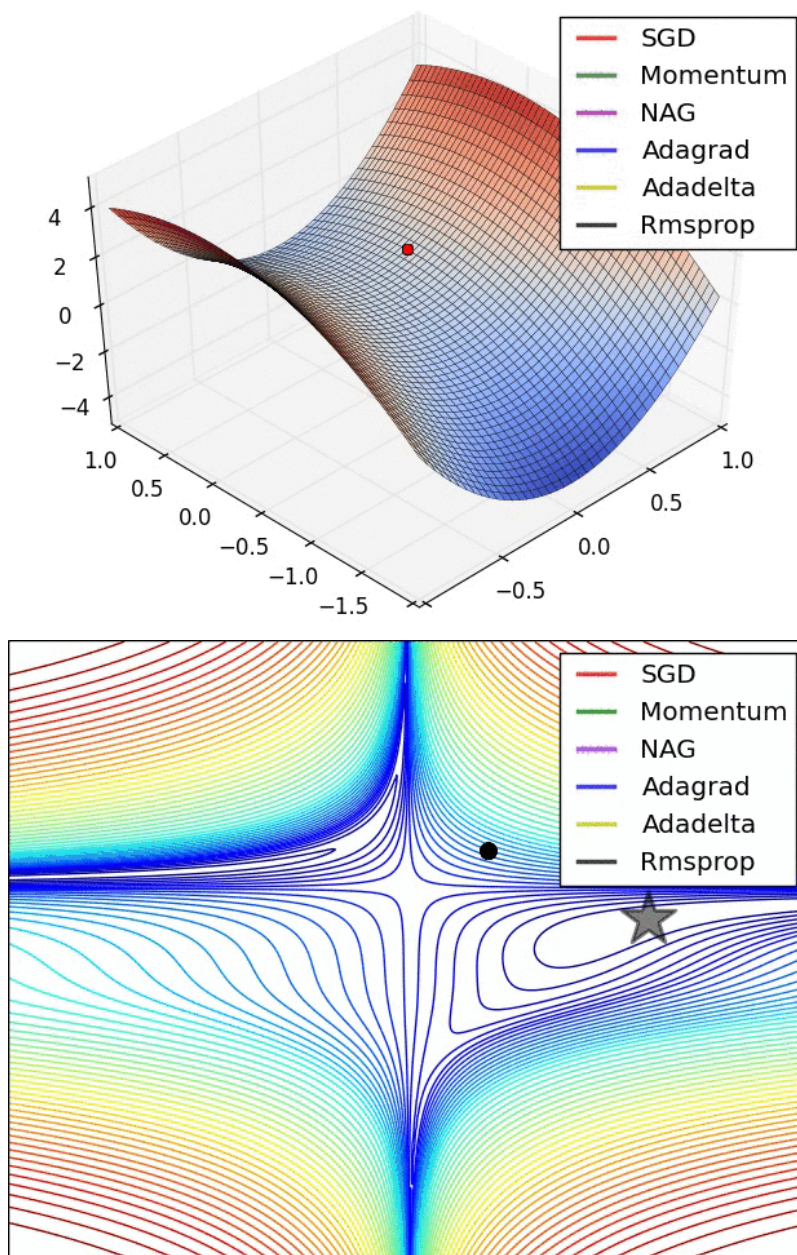
- For $j = 1, \dots, m$, randomly pick $i_j \in \{1, \dots, n\}$ and update weight

$$x_j = x_{j-1} - t(\nabla f_{i_j}(x_{t-1}) - \nabla f_{i_j}(\hat{x}) + \frac{1}{n} \sum_{i=1}^n \nabla f_i(\hat{x}))$$

- Option 1: Set $\hat{x}^{(k)} = x_m$.
- Option 2: Set $\hat{x}^{(k)} = x_j$ for randomly chosen $j \in \{1, \dots, m\}$.

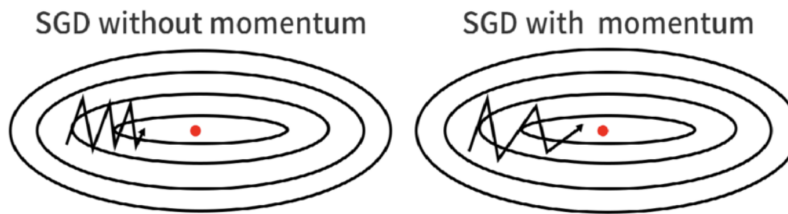
Other Improvements on Stochastic Gradient Descent

In this lecture, we are going to introduce more improvements on stochastic gradient descent. Here are some visualizations about the method we are going to introduce.



Momentum

A **momentum term**⁴ is usually added to SGD to improve the speed of learning and dampens oscillations.



- Initialize $x^{(0)}, v^{(0)} \in \mathbb{R}^d$
- Repeat:

$$v^{(k)} = \gamma v^{(k-1)} + t \nabla f_{i_k} (x^{(k-1)})$$

$$x^{(k)} = x^{(k-1)} - v^{(k)}, \quad k = 1, 2, 3, \dots$$

where decay factor $0 < \gamma < 1$, t is the step size, $i_k \in \{1, \dots, n\}$ is an index randomly selected at iteration k .

- If $\gamma = 0$, it is SGD.
- The update of SGD is usually unstable.
- Momentum works by adding a fraction γ of the update vector of the past time step to the current update vector.

Nesterov accelerated gradient (NAG)

Inspired by the **Nesterov method**⁵ for optimizing convex functions, Ilya Sutskever⁶ suggested a new form of momentum that often works better.

- Initialize $x^{(0)}, v^{(0)} \in \mathbb{R}^d$
- Repeat:

$$v^{(k)} = \gamma v^{(k-1)} + t \nabla f_{i_k} (x^{(k-1)} - \gamma v^{(k-1)})$$

$$x^{(k)} = x^{(k-1)} - v^{(k)}, \quad k = 1, 2, 3, \dots$$

where decay factor $\gamma < 1$, step size t is chosen to be fixed and small, $i_k \in \{1, \dots, n\}$ is an index randomly selected at iteration k .

Adaptive Gradient Algorithm (Adagrad)

Adagrad⁷ uses a different learning rate for every $x_j^{(k)}$ at every time step k . Here, $x_j^{(k)}$ is the j -th component of $x^{(k)}$, $j = 1, 2, \dots, d$.

- Initialize $x^{(0)} \in \mathbb{R}^d$
- Repeat:

$$x_j^{(k)} = x_j^{(k-1)} - \frac{t}{\sqrt{G_{j,j}^{(k-1)} + \epsilon}} \nabla_{x_j} f_{i_k}(x^{(k-1)}), \quad k = 1, 2, 3, \dots$$

where $G^{(k-1)} \in \mathbb{R}^{d \times d}$ is a diagonal matrix where the j -th diagonal element is the sum of the square of the gradient with respect to x_j up to time step k

$$G_{j,j}^{(k-1)} = \sum_{l=1}^k \left\| \nabla_{x_j} f_{i_l}(x^{(l-1)}) \right\|^2$$

and ϵ is added to prevent division by zero. Step size t is chosen to be fixed and small, $i_k \in \{1, \dots, n\}$ is an index randomly selected at iteration k .

- Adagrad **adapts the learning rate to the parameters**, performing smaller updates for parameters associated with frequently occurring features, and larger updates for parameters associated with infrequent features.
- It is well-suited for dealing with **sparse data** and greatly improves the robustness of SGD.
- Adagrad's main **weakness** is its accumulation of the squared gradients in the denominator. Since the accumulated sum keeps growing during training, **the learning rate will shrink and eventually become infinitesimally small**.

Adadelta

Adadelta⁸ is also proposed to solve the weakness of Adagrad.

Denote $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$. The running average at step k depends on the previous weighted average and the current gradient.

$$\mathbb{E}[g^2]^{(k)} = \gamma \mathbb{E}[g^2]^{(k-1)} + (1 - \gamma)(g^{(k)})^2$$

- The notation \mathbb{E} in this algorithm is rather vague, however it is exactly what has been written in the original work. It is in fact an exponential average.

Replace the diagonal matrix in Adagrad with this decaying average yields

$$\begin{aligned}\Delta x^{(k)} &= x^{(k)} - x^{(k-1)} \\ &= -\frac{t}{\sqrt{\mathbb{E}[g^2]^{(k)} + \epsilon}} g^{(k)} \\ &= -\frac{t}{RMS[g]^{(k)}} g^{(k)}\end{aligned}$$

where RMS stands for the root mean squared error. However, the authors noticed that the units in this update do not match, in which the update should have the same hypothetical units as the parameter.

To realize this, they first define another exponentially decaying average, the squared parameter updates:

$$\mathbb{E}[\Delta x^2]^{(k)} = \gamma \mathbb{E}[\Delta x^2]^{(k-1)} + (1 - \gamma)(\Delta x^{(k)})^2$$

The root mean squared error of parameter updates is

$$RMS[\Delta x]^{(k)} = \sqrt{\mathbb{E}[\Delta x^2]^{(k)} + \epsilon}$$

Since $RMS[\Delta x]^{(k)}$ is unknown, it is approximated with $RMS[\Delta x]^{(k-1)}$. Replacing the learning rate of the previous update with $RMS[\Delta x]^{(k-1)}$ gives the Adadelta update rule:

- Initialize $x^{(0)} \in \mathbb{R}^d$
- Repeat:

$$\begin{aligned}\Delta x^{(k)} &= -\frac{RMS[\Delta x]^{(k-1)}}{RMS[g]^{(k)}} g^{(k)} \\ x^{(k+1)} &= x^{(k)} + \Delta x^{(k)}\end{aligned}$$

- Since the step size has been eliminated from the update rule, we do not even need to set a default value.

RMSprop

RMSprop is another algorithm that deals with the weakness of Adagrad. Instead of summing up all the squared gradients, it calculates the weighted mean of the gradients.

- Initialize $x^{(0)}, v^{(0)} \in \mathbb{R}^d$
- Repeat:

$$v_j^{(k)} = \gamma v_j^{(k-1)} + (1 - \gamma) \left\| \nabla_{x_j} f_{i_k}(x^{(k-1)}) \right\|^2$$

$$x_j^{(k)} = x_j^{(k-1)} - \frac{t}{\sqrt{v_j^{(k)}} + \epsilon} \nabla_{x_j} f_{i_k}(x^{(k-1)})$$

$$j = 1, 2, \dots, d, \quad k = 1, 2, 3 \dots$$

- RMSprop in fact is identical to **the first update rule in the deduction of Adadelta**.
- RMSprop is not a published work, it's from **Hinton's lecture note**.
- Hinton suggests γ to be set to 0.9, while a good default value for the learning rate t is 0.001.

Adaptive Moment Estimation (Adam)

Adam² is another first-order gradient-based optimization method that computes adaptive learning rates for each parameter.

- Initialize $x^{(0)} \in \mathbb{R}^d, m^{(0)}, v^{(0)} \in \mathbb{R}$
- Repeat:

$$g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$$

$$m^{(k)} = \beta_1 m^{(k-1)} + (1 - \beta_1) g^{(k)}$$

$$v^{(k)} = \beta_2 v^{(k-1)} + (1 - \beta_2) \|g^{(k)}\|^2$$

$$\hat{m}^{(k)} = \frac{m^{(k)}}{1 - \beta_1^k}$$

$$\hat{v}^{(k)} = \frac{v^{(k)}}{1 - \beta_2^k}$$

$$x^{(k)} = x^{(k-1)} - t \frac{\hat{m}^{(k)}}{\sqrt{\hat{v}^{(k)}} + \epsilon}$$

where $\beta_1, \beta_2 \in [0, 1)$. β_1^k and β_2^k denote β_1 and β_2 to power k . Step size t is chosen to be fixed and small, $i_k \in \{1, \dots, n\}$ is an index randomly selected at iteration k .

- Adam has **quite a lot merits**: easy to implement, computationally efficient, little memory requirements, suitable for problems with large data/parameters, suitable for problems with noisy/sparse gradients...
- Widely used in machine learning.
- **Good default values** are: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

AdaMax

AdaMax is a variant of Adam based on the infinity norm. The $v^{(k)}$ factor in the Adam update rule scales the gradient inversely proportionally to the L^2 norm of the past and current gradient.

$$v^{(k)} = \beta_2 v^{(k-1)} + (1 - \beta_2) \|g^{(k)}\|_2$$

This can be generalized L^p norm.

$$v^{(k)} = \beta_2^p v^{(k-1)} + (1 - \beta_2^p) \|g^{(k)}\|_p$$

For large p , such variants become numerically unstable. However, let $p \rightarrow \infty$, the algorithm becomes surprisingly simple and stable.

- Initialize $x^{(0)} \in \mathbb{R}^d, m^{(0)}, v^{(0)} \in \mathbb{R}$
- Repeat:

$$\begin{aligned} g^{(k)} &= \nabla f_{i_k}(x^{(k-1)}) \\ m^{(k)} &= \beta_1 m^{(k-1)} + (1 - \beta_1) g^{(k)} \\ u^{(k)} &= \max(\beta_2 u^{(k-1)}, \|g^{(k)}\|_\infty) \\ x^{(k)} &= x^{(k-1)} - \frac{t m^{(k)}}{(1 - \beta_1^{(k)}) u^{(k)}} \end{aligned}$$

where β_1^k and β_2^k denote β_1 and β_2 to power k . Step size t is chosen to be fixed and small, $i_k \in \{1, \dots, n\}$ is an index randomly selected at iteration k .

- AdaMax is proposed in the same work as Adam.
- **Good default values** are: $\alpha = 0.002$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Nesterov-accelerated Adaptive Moment Estimation (Nadam)

Nadam¹⁰ is a combination of Adam and NAG.

Recall the update rule of NAG:

$$\begin{aligned}g^{(k)} &= \nabla f_{i_k}(x^{(k)} - \gamma m^{(k-1)}) \\m^{(k)} &= \gamma m^{(k-1)} + tg^{(k)} \\x^{(k+1)} &= x^{(k)} - m^{(k)}\end{aligned}$$

Dozat (author of Nadam paper) rewrote this rule into:

$$\begin{aligned}g^{(k)} &= \nabla f_{i_k}(x^{(k)}) \\m^{(k)} &= \gamma m^{(k-1)} + tg^{(k)} \\x^{(k+1)} &= x^{(k)} - (\gamma m^{(k)} + tg^{(k)})\end{aligned}$$

Recall the Adam update rule, here $\hat{v}^{(k)}$ remains the same:

$$\begin{aligned}m^{(k)} &= \beta_1 m^{(k-1)} + (1 - \beta_1)g^{(k)} \\\hat{m}^{(k)} &= \frac{m^{(k)}}{1 - \beta_1^k} \\x^{(k+1)} &= x^{(k)} - \frac{t}{\sqrt{\hat{v}^{(k)} + \epsilon}} \hat{m}^{(k)}\end{aligned}$$

Expand the third equation with the definitions of $\hat{m}^{(k)}$ and $m^{(k)}$.

$$\begin{aligned}x^{(k+1)} &= x^{(k)} - \frac{t}{\sqrt{\hat{v}^{(k)} + \epsilon}} \left(\frac{\beta_1 m^{(k-1)}}{1 - \beta_1^k} + \frac{(1 - \beta_1)g^{(k)}}{1 - \beta_1^k} \right) \\x^{(k+1)} &= x^{(k)} - \frac{t}{\sqrt{\hat{v}^{(k)} + \epsilon}} \left(\beta_1 \hat{m}^{(k-1)} + \frac{(1 - \beta_1)g^{(k)}}{1 - \beta_1^k} \right)\end{aligned}$$

Note that for simplicity, we ignore that the denominator is $1 - \beta_1^k$ and not $1 - \beta_1^{k-1}$. The Nadam update rule is:

- Initialize $x^{(0)} \in \mathbb{R}^d, m^{(0)}, v^{(0)} \in \mathbb{R}$
- Repeat:

$$\begin{aligned}
g^{(k)} &= \nabla f_{i_k}(x^{(k-1)}) \\
m^{(k)} &= \beta_1 m^{(k-1)} + (1 - \beta_1) g^{(k)} \\
v^{(k)} &= \beta_2 v^{(k-1)} + (1 - \beta_2) \|g^{(k)}\|^2 \\
\hat{m}^{(k)} &= \frac{m^{(k)}}{1 - \beta_1^k} \\
\hat{v}^{(k)} &= \frac{v^{(k)}}{1 - \beta_2^k} \\
x^{(k+1)} &= x^{(k)} - \frac{t}{\sqrt{\hat{v}^{(k)}} + \epsilon} \left(\beta_1 \hat{m}^{(k-1)} + \frac{(1 - \beta_1) g^{(k)}}{1 - \beta_1^k} \right)
\end{aligned}$$

where β_1^k and β_2^k denote β_1 and β_2 to power k . Step size t is chosen to be fixed and small, $i_k \in \{1, \dots, n\}$ is an index randomly selected at iteration k .

1. <https://arxiv.org/pdf/1309.2388.pdf> ↵
2. <https://papers.nips.cc/paper/5258-saga-a-fast-incremental-gradient-method-with-support-for-non-strongly-convex-composite-objectives.pdf> ↵
3. <https://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction.pdf> ↵
4. <https://www.sciencedirect.com/science/article/pii/S0893608098001166?via%3Dihub> ↵
5. <http://mpawankumar.info/teaching/cdt-big-data/nesterov83.pdf> ↵
6. https://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf ↵
7. <https://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf> ↵
8. <https://arxiv.org/pdf/1212.5701.pdf> ↵
9. <https://arxiv.org/pdf/1412.6980.pdf> ↵
10. <https://openreview.net/pdf/OM0jvwB8Jlp57ZJjtNEZ.pdf> ↵