Stochastic Gradient Descent

Zou Lexiao

Harbin Institute of Technology, Shenzhen

January 2023

Table of Contents

Motivation

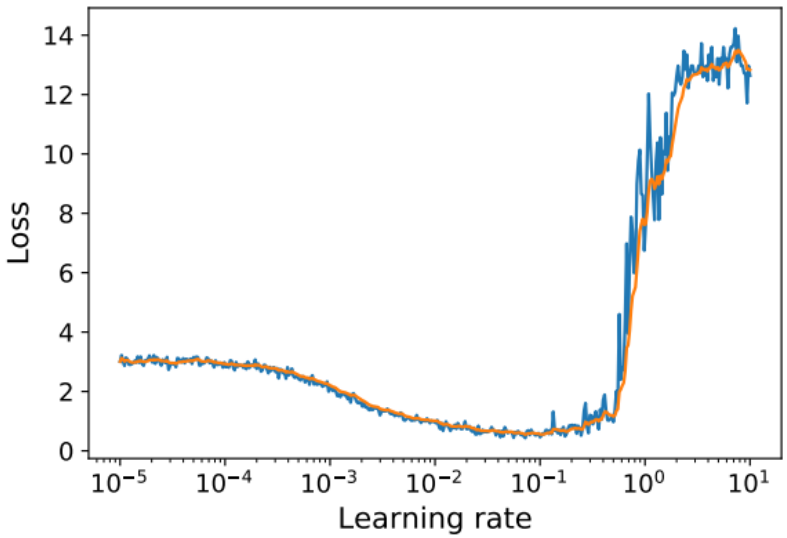


Figure 8.17: Loss vs learning rate (horizontal axis). Training loss vs learning rate for a small MLP fit to FashionMNIST using vanilla SGD. (Raw loss in blue, EWMA smoothed version in orange). Generated by lrschedule_tf.ipynb.

One heuristic

- Start with a small learning rate and gradually increase it, evaluating performance using a small number of minibatches.
- In practice, it is better to pick a rate that is slightly smaller than (i.e., to the left of) the one with the lowest loss, to ensure stability.

Learning rate schedule

- Definition:
  Learning rate schedule: Adjust the step size over time.
- Sufficient Condition for convergence:
  Robbins-Monro conditions[1]:

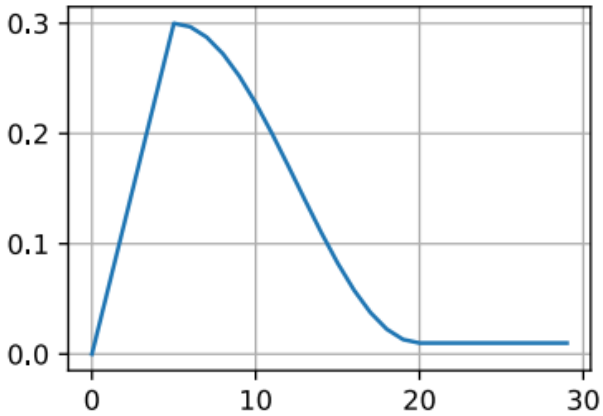
$$\eta_t \to 0, \frac{\sum_{t=1}^{\infty} \eta_t^2}{\sum_{t=1}^{\infty} \eta_t} \to 0$$

- Examples
  - piecewise constant $\eta_t = \eta_i$ if $t_i \leq t \leq t_{i+1}$
    - Step decay
    - Reduce-on-plateau
  - exponential decay $\eta_t = \eta_0 e^{-\lambda t}$
  - polynomial decay $\eta_t = \eta_0 (\beta t + 1)^{-\alpha}$
    - square-root schedule: $\alpha = 0.5, \beta = 1$

[1] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

Learning rate schedule: In deep learning community

Learning rate warmup(one-cycle learning rate schedule):

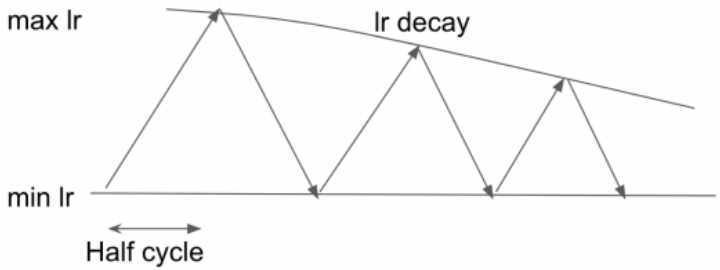


(a)

- Notice:
  To ensure convergence to a point, we must reduce the learning rate to 0.

Learning rate schedule: In deep learning community

Cyclical learning rate:



- Motivation: To escape local minima.
- Methods: The minimum and maximum learning rates can be found based on the initial "dry run" described above, and the half-cycle can be chosen based on how many restarts you want to do with your training budget.
- Related works: **stochastic gradient descent with warm restarts(SGDR)**
  storing all the checkpoints visited after each cool down, and using all of them as members of a **model ensemble**.

Table of Contents

Iterate averaging

- Motivation:
  The parameter estimates produced by SGD can be very unstable over time.
- Methods:

$$\overline{\boldsymbol{\theta}}_t = \frac{1}{t}\sum_{i=1}^{t}\boldsymbol{\theta}_i = \frac{1}{t}\boldsymbol{\theta}_t + \frac{t-1}{t}\overline{\boldsymbol{\theta}}_{t-1}$$

- Advantages:
  - Achieves the best possible asymptotic convergence rate among SGD algorithms, matching that of variants using second-order information[2].
  - Statistical benefits: in the case of linear regression, this method is equivalent to ℓ_2 regularization (i.e., ridge regression)
- Related work: **Stochastic Weight Averaging (SWA)**[3]
  SWA exploits the flatness in objectives used to train deep neural networks, to find solutions which provide better generalization.

[2] Boris Polyak and Anatoli Juditsky. "Acceleration of Stochastic Approximation by Averaging". In: *SIAM Journal on Control and Optimization* 30 (July 1992), pp. 838–855. DOI: 10.1137/0330046.

[3] Pavel Izmailov et al. "Averaging Weights Leads to Wider Optima and Better Generalization". In: *arXiv e-prints*, arXiv:1803.05407 (Mar. 2018), arXiv:1803.05407. arXiv: 1803.05407 [cs.LG].

Table of Contents

stochastic variance reduced gradient (SVRG)

Ever so often (e.g., once per epoch), we compute the full gradient at a "snapshot" of the model parameters $\tilde{\boldsymbol{\theta}}$; the corresponding "exact" gradient is therefore $\nabla \mathcal{L}(\tilde{\boldsymbol{\theta}})$. At step t, we compute the usual stochastic gradient at the current parameters, $\nabla \mathcal{L}_t(\boldsymbol{\theta}_t)$, but also at the snapshot parameters, $\nabla \mathcal{L}_t(\tilde{\boldsymbol{\theta}})$, which we use as a baseline. We can then use the following improved gradient estimate

$$\boldsymbol{g}_t = \nabla \mathcal{L}_t(\boldsymbol{\theta}_t) - \nabla \mathcal{L}_t(\tilde{\boldsymbol{\theta}}) + \nabla \mathcal{L}(\tilde{\boldsymbol{\theta}})$$

stochastic averaged gradient accelerated (SAGA)

We first initialize by computing $\boldsymbol{g}_n^{\text{local}} = \nabla \mathcal{L}_n(\boldsymbol{\theta}_0)$ for all n, and the average, $\boldsymbol{g}^{\text{avg}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{g}_n^{\text{local}}$. Then, at iteration t, we use the gradient estimate

$$\boldsymbol{g}_t = \nabla \mathcal{L}_n(\boldsymbol{\theta}_t) - \boldsymbol{g}_n^{\text{local}} + \boldsymbol{g}^{\text{avg}}$$

where $n \sim \text{Unif}\{1, \ldots, N\}$ is the example index sampled at iteration t. We then update $\boldsymbol{g}_n^{\text{local}} = \nabla \mathcal{L}_n(\boldsymbol{\theta}_t)$ and $\boldsymbol{g}^{\text{avg}}$ by replacing the old $\boldsymbol{g}_n^{\text{local}}$ by its new value.

- Unlike SVRG, it only requires one full batch gradient computation, at the start of the algorithm. While the downside is the extra memory cost.
- It is recommended for use in the sklearn logistic regression code when N is large and $\boldsymbol{x}$ is sparse.

Application to deep learning

For example, the use of batch normalization , data augmentation and dropout all break the assumptions of SVRG, since the loss will differ randomly in ways that depend not just on the parameters and the data index n.

Table of Contents

Preconditioned SGD

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \mathbf{M}_t^{-1} \boldsymbol{g}_t$$

- The noise in the gradient estimates make it difficult to reliably estimate the Hessian.
- Most practitioners use a diagonal preconditioner

AdaGrad

$$\theta_{t+1,d} = \theta_{t,d} - \eta_t \frac{1}{\sqrt{s_{t,d} + \epsilon}} g_{t,d}$$

where $d = 1 : D$ indexes the dimensions of the parameter vector, and

$$s_{t,d} = \sum_{i=1}^{t} g_{i,d}^2$$

is the sum of the squared gradients and $\epsilon > 0$ is a small term to avoid dividing by zero.

RMSprop and AdaDelta

Motivation: the denominator get larger over time. However, it might hurt performance as the denominator gets large too fast.

RMSprop

$$s_{t+1,d} = \beta s_{t,d} + (1-\beta) g_{t,d}^2$$

- We apply the EWMA of the past squared gradients rather than their sum.
- In practice we usually use $\beta \sim 0.9$, which puts more weight on recent examples.

RMSprop and AdaDelta

AdaDelta

$$\Delta\boldsymbol{\theta}_t = -\eta_t \frac{\sqrt{\boldsymbol{\delta}_{t-1} + \epsilon}}{\sqrt{\boldsymbol{s}_t + \epsilon}} \boldsymbol{g}_t$$

where

$$\boldsymbol{\delta}_t = \beta \boldsymbol{\delta}_{t-1} + (1-\beta)\left(\Delta\boldsymbol{\theta}_t\right)^2$$

- Eliminate the need to tune learning rate η_t, since the "units" of the numerator and denominator cancel, so we are just elementwise-multiplying the gradient by a scalar.[4]

However, since these adaptive learning rates need not decrease with time (unless we choose η_t to explicitly do so), these methods are not guaranteed to converge to a solution.

[4] Matthew Zeiler. "ADADELTA: An adaptive learning rate method". In: 1212 (Dec 2012).

Adam

Adam=RMSProp + momentum

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\,\boldsymbol{g}_t$$
$$\boldsymbol{s}_t = \beta_2 \boldsymbol{s}_{t-1} + (1-\beta_2)\,\boldsymbol{g}_t^2$$

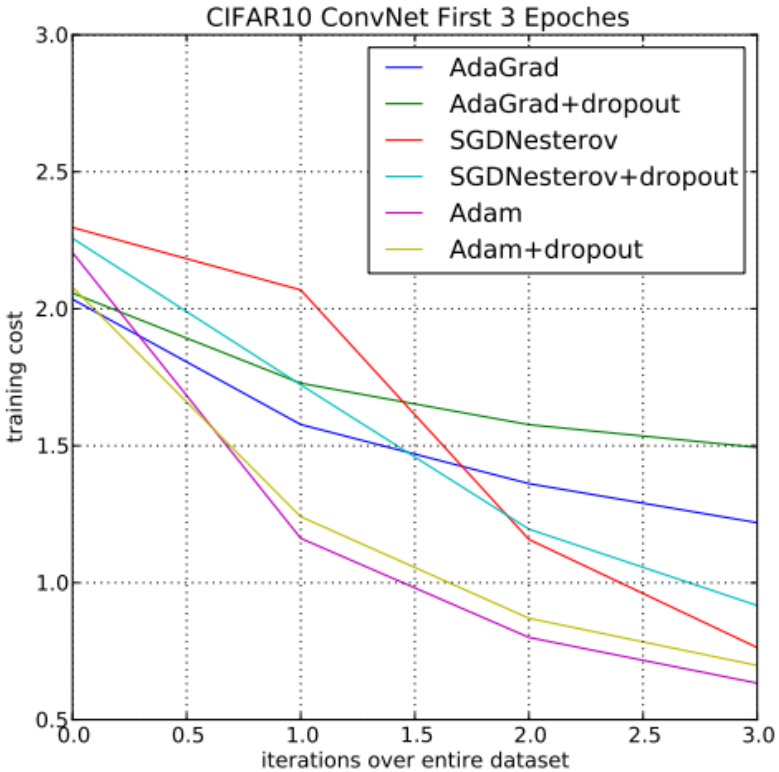
We then perform the update

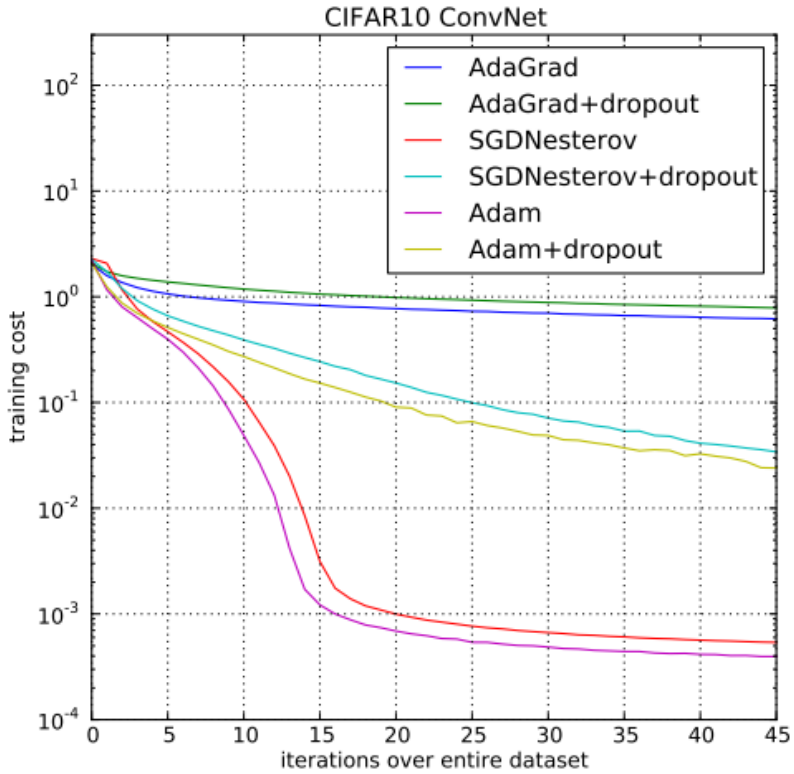
$$\Delta\boldsymbol{\theta}_t = -\eta_t \frac{1}{\sqrt{\boldsymbol{s}_t} + \epsilon}\boldsymbol{m}_t$$

If we initialize with $\boldsymbol{m}_0 = \boldsymbol{s}_0 = \mathbf{0}$, then initial estimates will be biased towards small values. The authors therefore recommend using the bias-corrected moments, which increase the values early in the optimization process. These estimates are given by

$$\hat{\boldsymbol{m}}_t = \boldsymbol{m}_t / \left(1-\beta_1^t\right)$$
$$\hat{\boldsymbol{s}}_t = \boldsymbol{s}_t / \left(1-\beta_2^t\right)$$

Adam


CIFAR10 ConvNet First 3 Epoches
AdaGrad
AdaGrad+dropout
SGDNesterov
SGDNesterov+dropout
Adam
Adam+dropout
training cost
iterations over entire dataset
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0


CIFAR10 ConvNet
AdaGrad
AdaGrad+dropout
SGDNesterov
SGDNesterov+dropout
Adam
Adam+dropout
training cost
iterations over entire dataset
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
5
10
15
20
25
30
35
40
45

Issues with adaptive learning rates

- Although it is called adaptive learning rate methods, it still needs to set the base learning rate η_0
- Since the EWMA methods are typically used in the stochastic setting where the gradient estimates are noisy, their learning rate adaptation can result in non-convergence even on convex problems.

Non-diagonal preconditioning matrices

Motivation: Although the methods we have discussed above can adapt the learning rate of each parameter, they do not solve the more fundamental problem of ill-conditioning due to correlation of the parameters, and hence do not always provide as much of a speed boost over vanilla SGD as one may hope.[5]

[5] Rohan Anil et al. "Scalable Second Order Optimization for Deep Learning". In: (Mar. 2021).

References

[1] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

[2] Boris Polyak and Anatoli Juditsky. "Acceleration of Stochastic Approximation by Averaging". In: *SIAM Journal on Control and Optimization* 30 (July 1992), pp. 838–855. DOI: 10.1137/0330046.

[3] Pavel Izmailov et al. "Averaging Weights Leads to Wider Optima and Better Generalization". In: *arXiv e-prints*, arXiv:1803.05407 (Mar. 2018), arXiv:1803.05407. arXiv: 1803.05407 [cs.LG].

[4] Matthew Zeiler. "ADADELTA: An adaptive learning rate method". In: 1212 (Dec. 2012).

[5] Rohan Anil et al. "Scalable Second Order Optimization for Deep Learning". In: (Mar. 2021).