CS 6501 Natural Language Processing

Optimization for Deep Learning

Yangfeng Ji

October 17, 2018

Department of Computer Science
University of Virginia

UNIVERSITY of VIRGINIA | ENGINEERING

Overview

1. Learning via Optimization
2. Stochastic Gradient Descent
3. Adaptive Learning Rates
4. Other Tricks

Learning via Optimization

Expected Loss

For a distribution of $\mathscr{D}$ over $(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ denotes the input and y is the corresponding output/label, the ideal prediction function is the one that minimize the expected loss $E(f) = \int_{\mathscr{D}} L(f(\boldsymbol{x}), y)$

$$f^* = \arg\min_f E(f) \tag{1}$$

where $L(\cdot, \cdot)$ is the loss function.

Empirical Loss

Instead of minimizing the expected loss in Equation 1, which is also impossible, we can minimize the empirical loss $E_n(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}^{(i)}), y^{(i)})$,

$$f_n = \arg\min_f E_n(f) \tag{2}$$

where $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n} \sim \mathcal{D}$ is the training set.

Question

How far from f_n to f^*

Question

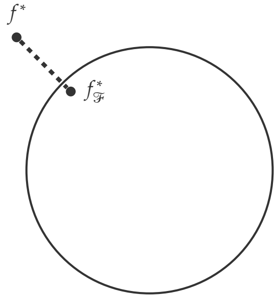
How far from f_n to f^*

Three steps in machine learning

1. collect data
2. design a model
3. optimize an objective function

Approximation Error

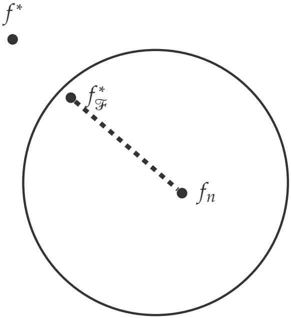
$$
\begin{aligned}
f^* &= \arg\min_{f} E(f) \qquad (3)\\
f^*_{\mathcal{F}} &= \arg\min_{f \in \mathcal{F}} E(f) \qquad (4)
\end{aligned}
$$



[Kearns et al., 1994, Agnostic ML]

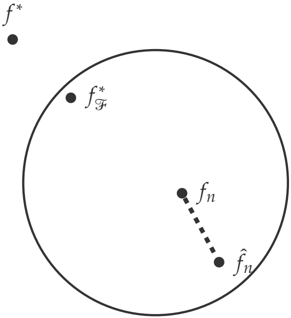
Estimation Error

$$
\begin{aligned}
f_{\mathcal{F}}^{*} &= \arg\min_{f \in \mathcal{F}} E(f) && (5)\\
f_n &= \arg\min_{f \in \mathcal{F}} E_n(f) && (6)
\end{aligned}
$$



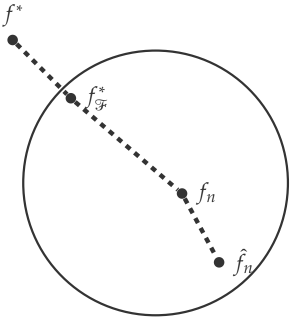
Optimization Error

$$f_n = \arg\min_{f \in \mathcal{F}} E_n(f) \tag{7}$$



Error Decomposition

$$\underbrace{E[E(f^*_{\mathcal{F}}) - E(f^*)]}_{\text{approximation error}} + \underbrace{E[E(f_n) - E(f^*_{\mathcal{F}})]}_{\text{estimation error}} + \underbrace{E[E(\hat{f}_n) - E(f_n)]}_{\text{optimization error}}$$

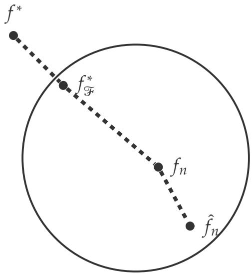


[Bottou, 2012, Sec. 18.3.1]

For a given machine learning problem,

- there is no way to know the oracle function f^* and $f^*_{\mathcal{F}}$, and
- it is difficult to get f_n, especially when $\mathcal{F}$ is a collection of deep neural networks

But it is useful to think about the decomposition.



For example, in the context of neural network learning,

- to reduce the approximation error is the motivation to design your neural network model carefully;
- to reduce the estimation error is the reason we should have enough data;
- to reduce the optimization error is why we need to know the optimization algorithms.

Stochastic Gradient Descent

Given a training set $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$, the empirical loss is defined as

$$\ell(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}) \tag{8}$$

where $L(\cdot, \cdot)$ is the loss function for a single example and $\boldsymbol{\theta}$ denotes the parameters in f.

SGD

To learn the parameter $\boldsymbol{\theta}$, we can compute the gradient with respect to one training example and then use stochastic gradient descent as

$$\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} - \eta \cdot \boldsymbol{g}^{(t-1)} \tag{9}$$

where $\boldsymbol{g}^{(t-1)} = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(t-1)})$ is the gradient of the single-example loss L.

Learning Rate

The usual conditions on the learning rates are

$$\sum_{t=1}^{\infty} \eta_t = \infty \tag{10}$$

$$\sum_{t=1}^{\infty} \eta_t^2 \leq \infty \tag{11}$$

A simplest function that satisfies these conditions is $\eta_t = \frac{1}{t}$.

[Bottou, 1998]

SGD with Momentum

Given the loss function $L(\boldsymbol{\theta})$ to be minimized, SGD with momentum is given by

$$\boldsymbol{v}^{(t)} = \mu \boldsymbol{v}^{(t-1)} + \boldsymbol{g}^{(t-1)} \quad (12)$$

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta \boldsymbol{v}^{(t)} \quad (13)$$

where η is still the learning rate and $\mu \in [0, 1]$ is the momentum coefficient. Usually, $\mu = 0.99$ or 0.999.

Intuitive Explanation

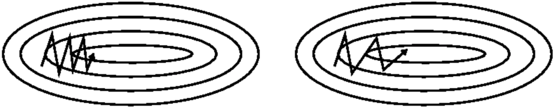
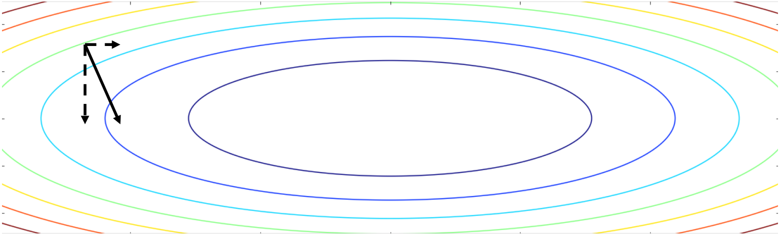


Figure: The effect of momentum in SGD. Left: SGD without momentum. Right: SGD with momentum. (Credit: Genevieve B. Orr)

$$y = x_1^2 + 10x_2^2 \tag{14}$$

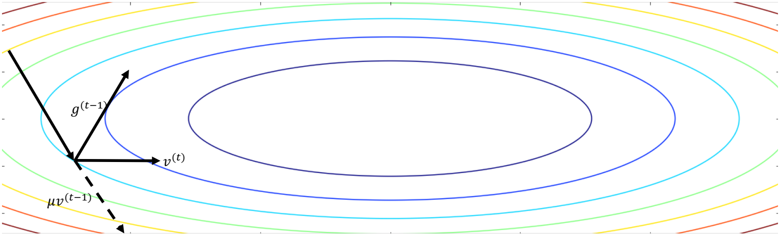
$$\frac{\partial y}{\partial x_1} = 2x_1 \tag{15}$$

$$\frac{\partial y}{\partial x_2} = 20x_2 \tag{16}$$



$$v^{(t)} = \mu v^{(t-1)} + g^{(t-1)} \tag{17}$$

$$\theta^{(t)} = \theta^{(t-1)} - \eta v^{(t)} \tag{18}$$



Adaptive Learning Rates

Basic Idea

For neural networks, the motivation of picking a different learning rate for each θ_k (the k-th component of parameter θ) is not new [LeCun et al., 2012] (the article was originally published in 1998).

- The basic idea is to make sure that all θ_k's converge roughly at the same speed.
- Depending on the curvature of the error surface, some θ_k's may require a small learning rate in order to avoid divergence, while others may require a large learning rate in order to converge fast.

AdaGrad

The basic idea of **AdaGrad** is to modify the learning rate η for $\boldsymbol{\theta}_k$ by using the history of $\partial_{\boldsymbol{\theta}_k} L$

$$\boldsymbol{\theta}_k^{(t)} = \boldsymbol{\theta}_k^{(t-1)} - \frac{\eta_0}{\sqrt{G_{k,k}^{(t-1)} + \epsilon}} \boldsymbol{g}_k^{(t-1)} \tag{19}$$

where $\boldsymbol{g}_k^{(t-1)} = [\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(t-1)})]_k$ is the k-th component of $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(t-1)})$, $G_{k,k}^{(t-1)} = \sum_{i=1}^{t-1} (\boldsymbol{g}_k^{(i)})^2$, η_0 is the initial learning rate and ϵ is a smoothing parameter usually with order 10^{-6}.

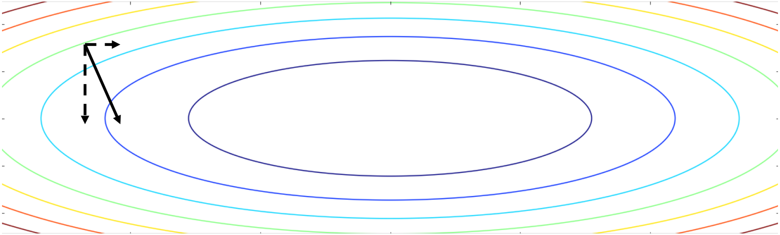
AdaGrad (II)

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \frac{\eta_0}{\sqrt{G_{k,k}^{(t-1)} + \epsilon}} g_k^{(t-1)} \tag{20}$$

The accumulation of squared gradients from the beginning of training can result in a a premature and excessive decrease in the effective learning rate.

AdaGrad (III)

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \frac{\eta_0}{\sqrt{G_{k,k}^{(t-1)} + \epsilon}} g_k^{(t-1)} \tag{21}$$



RMSProp

RMSProp uses a moving average over the past

$$r_k^{(t)} = \rho r_k^{(t-1)} + (1-\rho)[g_k^{(t-1)}]^2 \tag{22}$$

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \frac{\eta_0}{\sqrt{r_k^{(t)} + \epsilon}} g_k^{(t-1)} \tag{23}$$

[Hinton et al., 2012]

Adam

$$
\begin{aligned}
\boldsymbol{v}_k^{(t)} &= \mu \boldsymbol{v}_k^{(t-1)} + (1-\mu)\boldsymbol{g}_k^{(t-1)} && (24) \\
\boldsymbol{r}_k^{(t)} &= \rho \boldsymbol{r}_k^{(t-1)} + (1-\rho)[\boldsymbol{g}_k^{(t-1)}]^2 && (25) \\
\hat{\boldsymbol{v}}_k^{(t)} &= \frac{\boldsymbol{v}_k^{(t)}}{1-\mu^t} && (26) \\
\hat{\boldsymbol{r}}_k^{(t)} &= \frac{\boldsymbol{r}_k^{(t)}}{1-\rho^t} && (27) \\
\boldsymbol{\theta}_k^{(t)} &= \boldsymbol{\theta}_k^{(t-1)} - \eta_0 \frac{\hat{\boldsymbol{v}}_k^{(t)}}{\sqrt{\hat{\boldsymbol{r}}_k^{(t)} + \epsilon}} && (28)
\end{aligned}
$$

The default values of μ and ρ are 0.9 and 0.999 respectively.

Adam (II)

Recent theoretical work [Reddi et al., 2018] shows that, for any constant $\mu, \rho \in [0, 1)$, if $\mu < \sqrt{\rho}$, there is a stochastic convex optimization problem for which Adam does not converge to the optimal solution.

How to Choose a Optimization Algorithm?

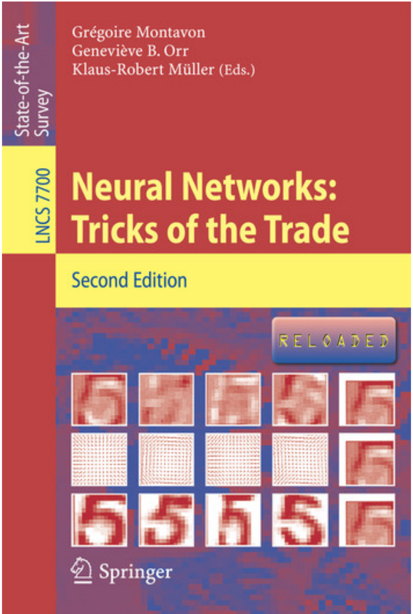
- User's familiarity with the algorithms [Goodfellow et al., 2016]
- Start from SGD first (My opinion)

Other Tricks

Other Tricks

1. Shuffle training examples in each epoch
2. Normalize inputs
3. Initialization

Further Reference



Summary

1. Learning via Optimization
2. Stochastic Gradient Descent
3. Adaptive Learning Rates
4. Other Tricks

Reference

Bottou, L. (1998).
Online learning and stochastic approximations.
On-line learning in neural networks, 17(9):142.

Bottou, L. (2012).
Stochastic gradient descent tricks.
In *Neural networks: Tricks of the trade*, pages 421–436. Springer.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016).
Deep Learning, volume 1.
MIT press Cambridge.

Hinton, G., Srivastava, N., and Swersky, K. (2012).
Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

Kearns, M. J., Schapire, R. E., and Sellie, L. M. (1994).
Toward efficient agnostic learning.
Machine Learning, 17(2-3):115–141.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012).
Efficient backprop.
In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Reddi, S. J., Kale, S., and Kumar, S. (2018).
On the convergence of adam and beyond.
In *ICLR*.