

# Momentum-based Optimization Techniques in Machine Learning

Pham Viet Hung



M U E G Y E T E M 1 7 8 2

# Content

- ▶ Overview
  - ▶ Estimation/Learning vs. Pure Optimization
  - ▶ Gradient-Based Optimization
- ▶ Momentum-based methods
  - ▶ Vanila SGD
  - ▶ SGD with Polyak Momentum
  - ▶ SGD with Nesterov Momentum

But before digging into the topic...

<https://www.youtube.com/watch?v=dQ4jUFmp7hQ>

# Overview

## Estimation/Learning vs. Pure Optimization

- ▶ **Stats:** Estimation; **CS:** Learning — Meaning: using data to estimate an unknown quantity (Wasserman, 2004)
- ▶ In ML/DL, optimization is a tool to achieve the best possible  $P$  (some performance metric) defined w.r.t the test set; hard to track it  $\implies$  optimize  $P$  indirectly through a cost function  $J(\theta)$ .
- ▶ In pure optimization, care about only  $J(\theta)$
- ▶ Cost Function:  $J(\theta) = \mathbb{E}_{(x,y) \sim \hat{p}_{\text{data}}} [L(f(x; \theta), y)]$ ;  $\hat{p}_{\text{data}}$ : empirical distro
- ▶ Empirical Risk Minimization (ERM):

$$\mathbb{E}_{(x,y) \sim \hat{p}_{\text{data}}} [L(f(x; \theta), y)] = \frac{1}{n} \sum_{i=1}^n L(f(x^{(i)}; \theta), y^{(i)}) \quad (1)$$

## Gradient-based Optimization

- ▶ **First-order optimization algorithms:** Using variants of gradient-only algorithm. Mostly some variants of gradient descent algorithms.
- ▶ **Second-order optimization algorithms:** Algorithms that make use of Hessian matrix (a.k.a second derivatives). *Problem:* Poor conditioning of the Hessian matrix.
- ▶ The above-mentioned optimization techniques are mostly used in convex optimization. In Deep Learning, mostly as a subroutine of some Deep Learning algorithms.
- ▶ Usually variants of Stochastic Gradient Descent optimization (usually with minibatch) is used in Machine Learning (ML) and Deep Learning (DL).

# Momentum-based methods

## Vanila SGD (1)

- ▶ SGD (with minibatch) and its flavours are the most used optimization methods for ML problem.
- ▶ The step length/learning rate is defined as fixed in SGD, but in practice we need a gradually decreasing values  $\implies$  denote it as  $\epsilon_k$
- ▶ The two sufficient conditions to guarantee convergence of SGD:

$$\sum_{k=1}^{\infty} \epsilon_k = \infty \quad (2)$$

$$\sum_{k=1}^{\infty} \epsilon_k^2 \leq \infty \quad (3)$$



## Vanila SGD (2)

---

**Algorithm 8.1** Stochastic gradient descent (SGD) update at training iteration  $k$

---

**Require:** Learning rate  $\epsilon_k$

**Require:** Initial parameter  $\theta$

**while** stopping criterion not met **do**

    Sample a minibatch of  $m$  examples from the training set  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  with corresponding targets  $\mathbf{y}^{(i)}$ .

    Compute gradient estimate:  $\hat{\mathbf{g}} \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$ .

    Apply update:  $\theta \leftarrow \theta - \epsilon_k \hat{\mathbf{g}}$ .

**end while**

---

**Figure:** Algorithm of the Vanila SGD (Goodfellow et al., 2016)

# SGD with Polyak Momentum (1)

- ▶ Sometimes learning with SGD is slow.
- ▶ Polyak (1964) proposes a method to accelerate learning in a high curvature, small and consistent/noisy gradient environment.
- ▶ The key driver is the **velocity** variable  $v$

## SGD with Polyak Momentum (2)

---

**Algorithm 8.2** Stochastic gradient descent (SGD) with momentum

---

**Require:** Learning rate  $\epsilon$ , momentum parameter  $\alpha$

**Require:** Initial parameter  $\theta$ , initial velocity  $v$

**while** stopping criterion not met **do**

    Sample a minibatch of  $m$  examples from the training set  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  with corresponding targets  $\mathbf{y}^{(i)}$ .

    Compute gradient estimate:  $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$ .

    Compute velocity update:  $\mathbf{v} \leftarrow \alpha \mathbf{v} - \epsilon \mathbf{g}$ .

    Apply update:  $\theta \leftarrow \theta + \mathbf{v}$ .

**end while**

---

**Figure:** Algorithm of the SGD w/ Polyak Momentum (Goodfellow et al., 2016)

# SGD with Nesterov Momentum (1)

- ▶ Based on Nesterov's (1983, 2004) accelerated method, Sutskever (2013) in his PhD thesis proposed a variant of momentum-based SGD optimization technique
- ▶ Difference between Nesterov and Polyak momentum is the evaluation point of the gradient.
- ▶ With Nesterov momentum, the evaluation of the the gradient occurs after the current velovity is applied.

## SGD with Nesterov Momentum (2)

---

**Algorithm 8.3** Stochastic gradient descent (SGD) with Nesterov momentum

---

**Require:** Learning rate  $\epsilon$ , momentum parameter  $\alpha$

**Require:** Initial parameter  $\theta$ , initial velocity  $v$

**while** stopping criterion not met **do**

    Sample a minibatch of  $m$  examples from the training set  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  with corresponding labels  $\mathbf{y}^{(i)}$ .

    Apply interim update:  $\tilde{\theta} \leftarrow \theta + \alpha v$ .

    Compute gradient (at interim point):  $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\theta}} \sum_i L(f(\mathbf{x}^{(i)}; \tilde{\theta}), \mathbf{y}^{(i)})$ .

    Compute velocity update:  $v \leftarrow \alpha v - \epsilon \mathbf{g}$ .

    Apply update:  $\theta \leftarrow \theta + v$ .





**end while**

---





**Figure:** Algorithm of the SGD w/ Nesterov Momentum (Goodfellow et al., 2016)

Thank you very much for your attention!

## References 1

-  Wasserman, L. (2004). *All of statistics: a concise course in statistical inference* (Vol. 26). New York: Springer.
-  Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press.
-  Melville, J. (2016). *Nesterov Accelerated Gradient and Momentum*.  
[https://jlmelville.github.io/mize/nesterov.html#Sutskever\\_Nesterov\\_Momentum](https://jlmelville.github.io/mize/nesterov.html#Sutskever_Nesterov_Momentum)
-  Li, F-F. et al. (2022). *CS231n: Deep Learning for Computer Vision*. Stanford University. <https://cs231n.github.io/neural-networks-3/>

## References 2

-  Polyak, B. T. (1964). *Some methods of speeding up the convergence of iteration methods*. USSR Computational Mathematics and Mathematical Physics, 4(5), 1–17.
-  Nesterov, Y. (1983). *A method of solving a convex programming problem with convergence rate  $O(1/k^2)$* . Soviet Mathematics Doklady, 27(2):372–376
-  Nesterov, Y. (2004). *Introductory lectures on convex optimization: A basic course*. Kluwer Academic Publishers
-  Sutskever, I. (2013). *Training Recurrent Neural Networks*. University of Toronto.  
[https://tspace.library.utoronto.ca/bitstream/1807/36012/6/Ilya\\_Sutskever\\_201306\\_PhD\\_thesis.pdf](https://tspace.library.utoronto.ca/bitstream/1807/36012/6/Ilya_Sutskever_201306_PhD_thesis.pdf)