On the importance of initialization and momentum in deep learning

수학과 오서영

Overview

DNN and RNN : almost impossible to train using SGD with momentum.
-> SGD with momentum + well-designed random initialization + slowly increasing for momentum parameter

→ It can train to levels of performance with only Hessian-Free optimization
→ Initialization and Momentum are crucial

DNN and RNN from random initializations -> fail
→ Carefully tuned momentum methods suffice for dealing with the curvature issues in DNN and RNN without the need for sophisticated second-order methods.

Introduction

DNN (Deep neural networks)
- Greedy layer-wise training (2006)
- auxiliary objective and fine tune the entire network with standard optimization (SGD) (2007)
- truncated-Newton method called Hessian-free optimization (2010)
    -> training DNN from certain random initializations without the use of pre-training / achieve low errors

RNN (Recurrent neural networks)
- Have a layer for each time-step with parameter sharing across the layers
    -> harder to train than DNN
- HF method of Martens : effectively train RNNs on artificial problems that exhibit very long-range dependencies (2010)

Introduction

Contributions

- Study effectiveness of SGD when combined with well-chosen initialization schemes and various forms of momentum-based acceleration

- Definite performance gap between plain SGD and HF on certain deep and temporal learning problems
    -> this gap can be eliminated by careful use of classical momentum methods

- Show how certain carefully designed schedules for the constant of momentum μ
    -> inspired by various theoretical convergence-rate theorems.
    -> produce results that even surpass those by Martens on certain deep-autoencoder training tasks

Momentum and Nesterov's Accelerated Gradient

1. Classical Momentum (CM)

- Technique for accelerating gradient descent that accumulates a velocity vector in directions of persistent reduction in the objective across iterations.

$$\begin{aligned} v_{t+1} &= \mu v_t - \varepsilon \underline{\nabla f(\theta_t)} \longrightarrow \text{Gradient at } \theta_t \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned}$$

f : Objective function to be minimized

$\varepsilon > 0$: learning rate

$\mu \in [0, 1]$: momentum coefficient

Momentum and Nesterov’s Accelerated Gradient

Since directions d of low-curvature have slower local chance in their rate of reduction
-> They will tend to persist across iterations

CM can considerably accelerate convergence to a local minimum requiring fewer iterations than steepest descent to reach the same level of accuracy

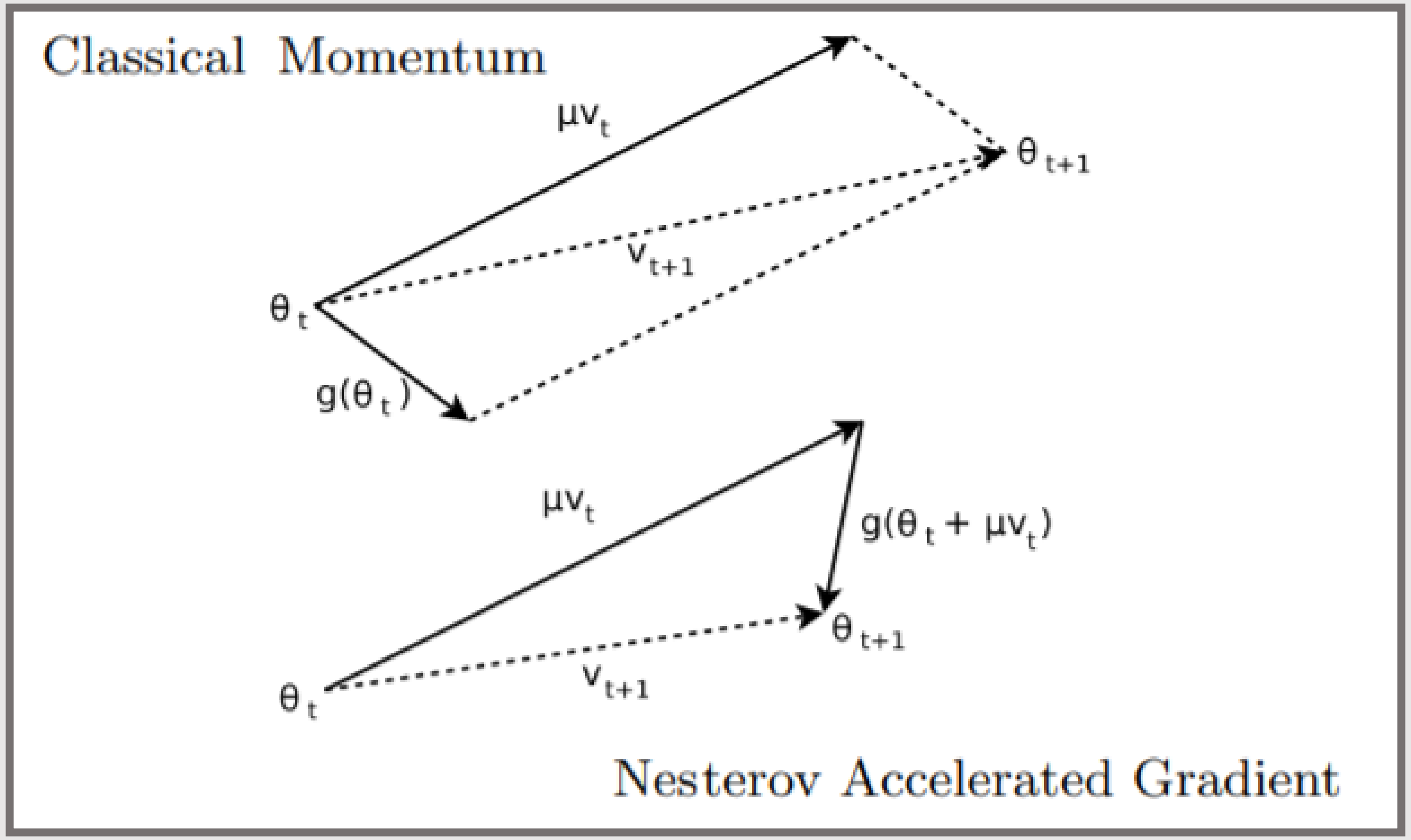
Momentum and Nesterov's Accelerated Gradient

2. Nesterov's Accelerated Gradient (NAG)

- Like momentum, NAG is a first-order optimization method with better convergence rate guarantee than GD in certain situations.

$$\begin{aligned} v_{t+1} &= \mu v_t - \varepsilon \nabla \underline{f(\theta_t + \mu v_t)} \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned}$$

Momentum and Nesterov's Accelerated Gradient

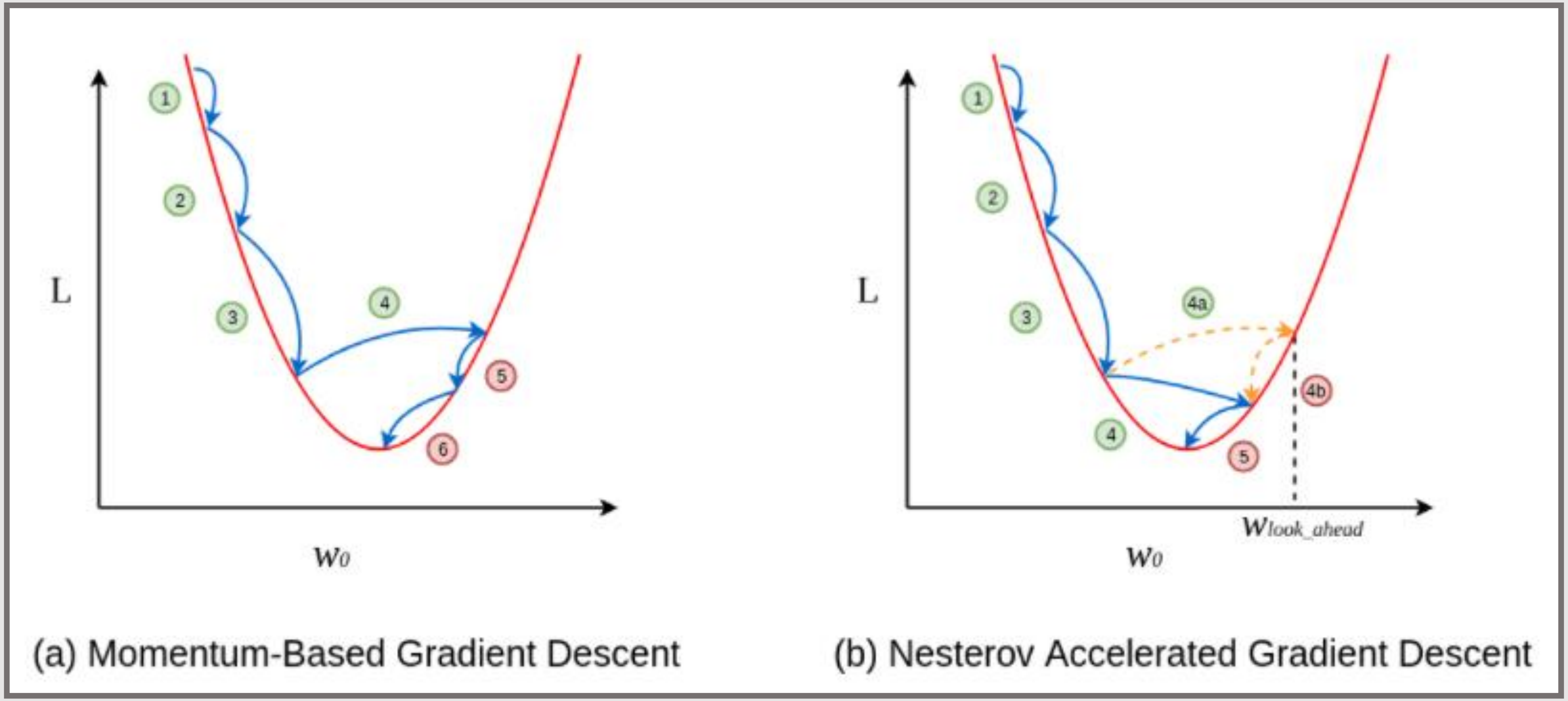


$$\begin{aligned} v_{t+1} &= \mu v_t - \varepsilon \nabla f(\theta_t) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned}$$

$$\begin{aligned} v_{t+1} &= \mu v_t - \varepsilon \nabla f(\theta_t + \mu v_t) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned}$$

CM	NAG

Momentum and Nesterov's Accelerated Gradient



(a) Momentum-Based Gradient Descent

(b) Nesterov Accelerated Gradient Descent

The relationship between CM and NAG

Both CM and NAG compute the new velocity by applying a gradient-based correction to the previous velocity vector and then add the velocity to θ_t

CM : computes the gradient update from the current position θ_t
NAG : first performs a partial update to θ_t , computing $\underline{\theta_t + \mu v_t}$

Similar to θ_{t+1}

-> NAG changes v in a quicker and more responsive way, letting it behave more stably than CM Especially for higher values of μ

$$\begin{aligned} v_{t+1} &= \mu v_t - \varepsilon \nabla f(\theta_t) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned}$$

CM

$$\begin{aligned} v_{t+1} &= \mu v_t - \varepsilon \nabla f(\theta_t + \mu v_t) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned}$$

NAG

The relationship between CM and NAG

Where the addition of μv_t

- Results in an immediate undesirable increase in the objective f.
The gradient correction to the velocity v_t is computed at $\theta_t + \mu v_t$
If μv_t is indeed a poor update, then $\nabla f(\theta_t + \mu v_t)$ will point back towards θ_t
More strongly than $\nabla f(\theta_t)$ does -> provide a larger and more timely correction to v_t than CM

$$\begin{aligned} v_{t+1} &= \mu v_t - \varepsilon \nabla f(\theta_t) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned}$$

CM

$$\begin{aligned} v_{t+1} &= \mu v_t - \varepsilon \nabla f(\theta_t + \mu v_t) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned}$$

NAG

Results

NAG can achieve results that are comparable with some of the best HF results for training deep autoencoders.

Training Technique
Reducing μ and moving to this fine convergence regime (take place along the high-curvature directions) too early may make it difficult for the optimization to make significant progress along the low-curvature directions, since without the benefit of momentum-based acceleration, first-order methods are notoriously bad at this.

Sparse Initialization technique (SI)

- Each random unit is connected to 15 randomly chosen units in the previous layer, whose weights are drawn from a unit Gaussian, and the biases are set to zero.

- The total amount of input to each unit will not depend on the size of the previous layer and hence they will not as easily saturate.

- Because the inputs to each unit are not all randomly weighted blends of the outputs of many 100s or 1000s of units in the previous layer, they will tend to be qualitatively more "diverse" in their response to inputs.