UPMC SORBONNE UNIVERSITÉS

LIP6

FIRST-ORDER OPTIMIZATION ALGORITHMS

Tuesday 24th October, 2017

Micael Carvalho

What's the problem ?

$$\arg\min_{\theta} L(\phi(\theta, X))$$

$L(\theta) = L(\phi(\theta, X))$, for future notations

$L(\cdot)$ is a differentiable loss function [☺];
$\theta \in \mathbb{R}^d$, d may be huge (1 billion or even more) [☹×2].

Content

1. Gradient Descent;
2. Gradient Descent + Momentum;
3. Nesterov Accelerated Gradient;
4. Adagrad;
5. RMSProp;
6. Adadelta;
7. Adam.

RevolutionAry poNDerOus Method (RANDOM)

Randomly guess the parameters.

Very easy to understand and implement [☺];
Usually, has the prob. $\frac{1}{(2^{32})^d}$ of finding the global minimum [☹].

RevolutionAry poNDerOus Method (RANDOM)

Randomly guess the parameters.

Very easy to understand and implement [☺];
Usually, has the prob. $\frac{1}{(2^{32})^d}$ of finding the global minimum [☹].

Fortunately, for $d = 5$, this is only*:

$$\frac{n}{1461501637330902918203684832716283019655932542976}$$

$*$ with n global minima.

Gradient Descent (GD)

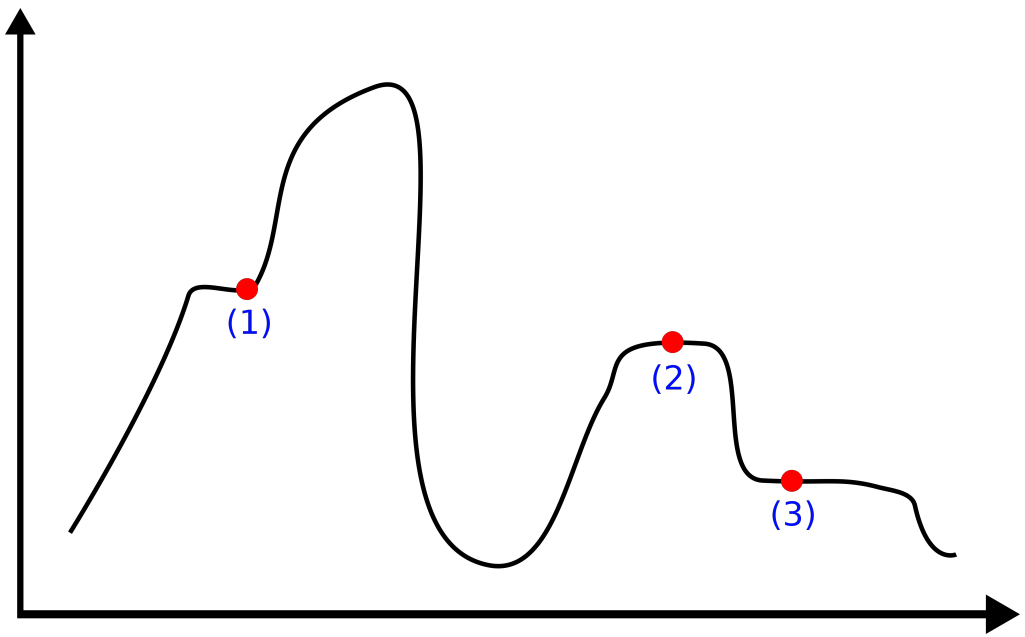
Uses the gradient's information to decide the direction of the step.

$$\Delta\theta_t = \lambda \frac{\partial L(\theta_t)}{\partial \theta_t} = \lambda \nabla_{\theta_t} L(\theta_t)$$

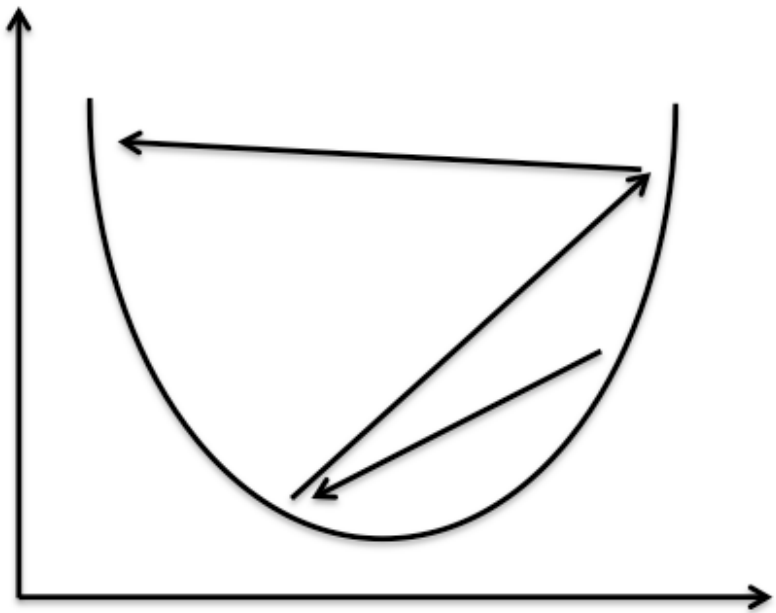
$$\theta_{t+1} = \theta_t - \Delta\theta_t$$

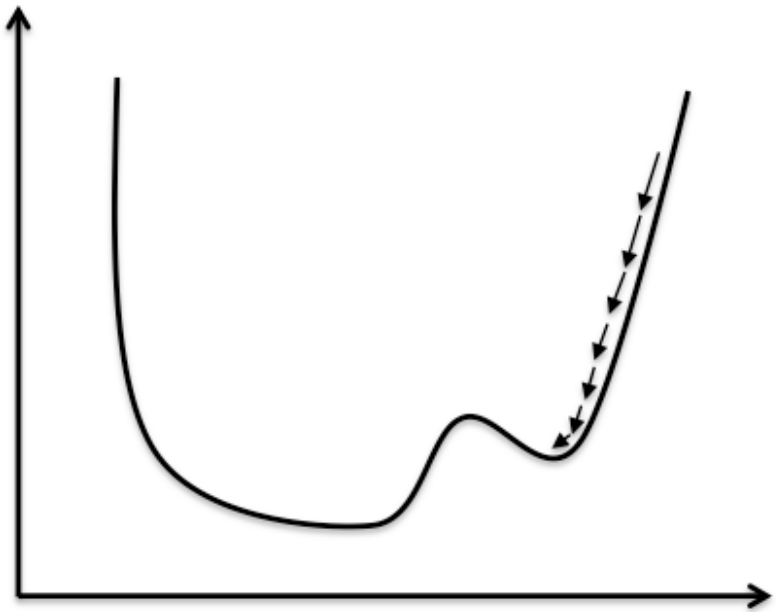
$\frac{\partial L(\theta_t)}{\partial \theta_t}$ is a vector of derivatives, also known as **gradient** (∇_{θ_t}).

[☺] Works well for high-dimensionality optimizations;
[☹] Easily trapped by saddle points, ravines etc (1-3);
[☹] Not adaptable to the parameters' frequency (3);
[☹] Hard to choose the learning rate λ;



[☺] Works well for high-dimensionality optimizations;
[☹] Easily trapped by saddle points, ravines etc (1-3);
[☹] Not adaptable to the parameters' frequency (3);
[☹] Hard to choose the learning rate λ;





Gradient Descent with Momentum (GDM)

Learning representations by back-propagating errors

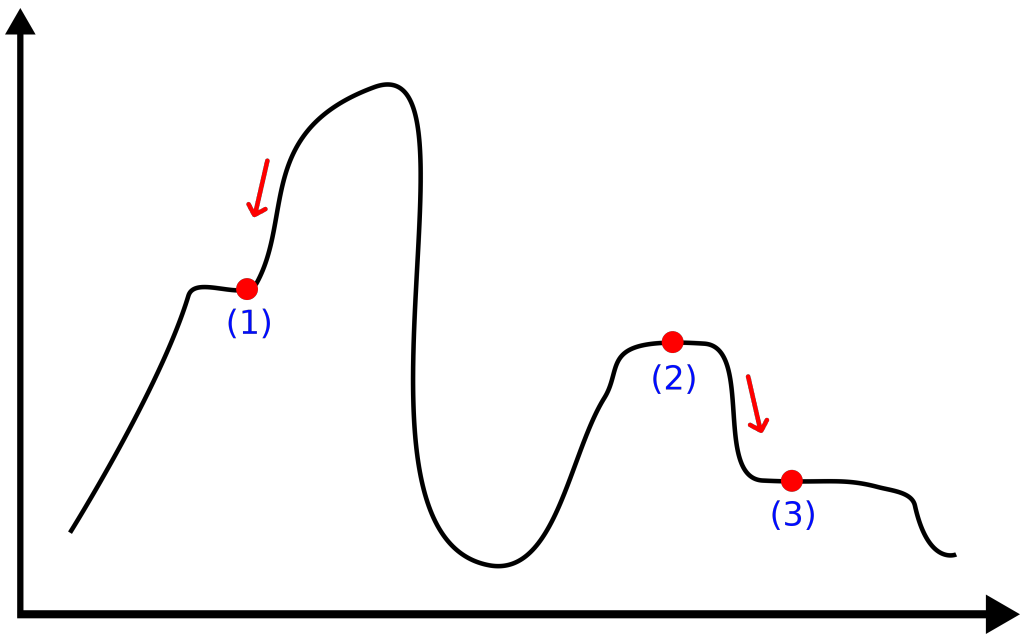
David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams
Nature, 1986

$$\text{GD: } \Delta\theta_t = \lambda \nabla_{\theta_t} L(\theta_t)$$

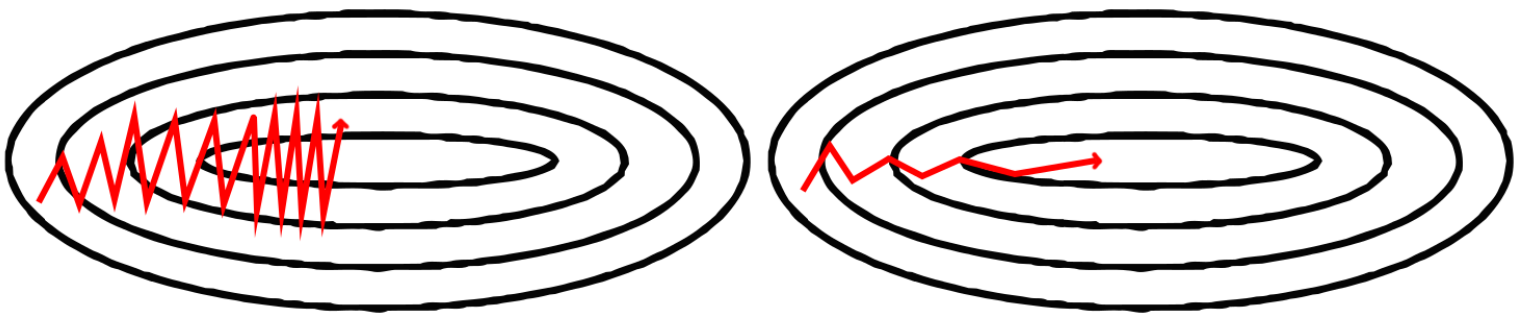
$$\text{GDM: } \Delta\theta_t = \gamma \Delta\theta_{t-1} + \lambda \nabla_{\theta_t} L(\theta_t)$$

γ is the momentum and its value is usually 0.9.

[☺] Not easily trapped by saddle points, ravines etc (1,3);
[☺] Reduced oscillation around local minima and acceleration;
[😐] A little adaptable to the parameters' frequency;
[☹] Hard to choose the learning rate λ;



[☺] Not easily trapped by saddle points, ravines etc (1,3);
[☺] Reduced oscillation around local minima and acceleration;
[😐] A little adaptable to the parameters' frequency;
[☹] Hard to choose the learning rate λ;



Nesterov Accelerated Gradient (NAG)

On the importance of initialization and momentum in deep learning

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton
ICML, 2015
Inspired by Yurii Nesterov's theorems (1983, 2003)

$$\text{GDM: } \Delta\theta_t = \gamma\Delta\theta_{t-1} + \lambda\nabla_{\theta_t} L(\theta_t)$$

$$\text{NAG: } \Delta\theta_t = \gamma\Delta\theta_{t-1} + \lambda\nabla_{\theta_t} L(\theta_t - \gamma\Delta\theta_{t-1})$$

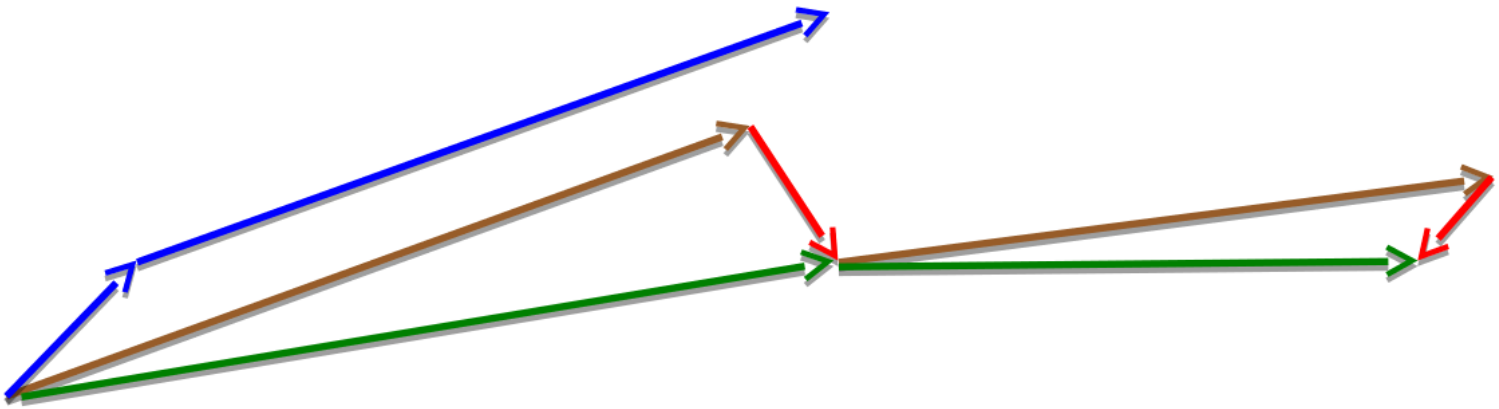
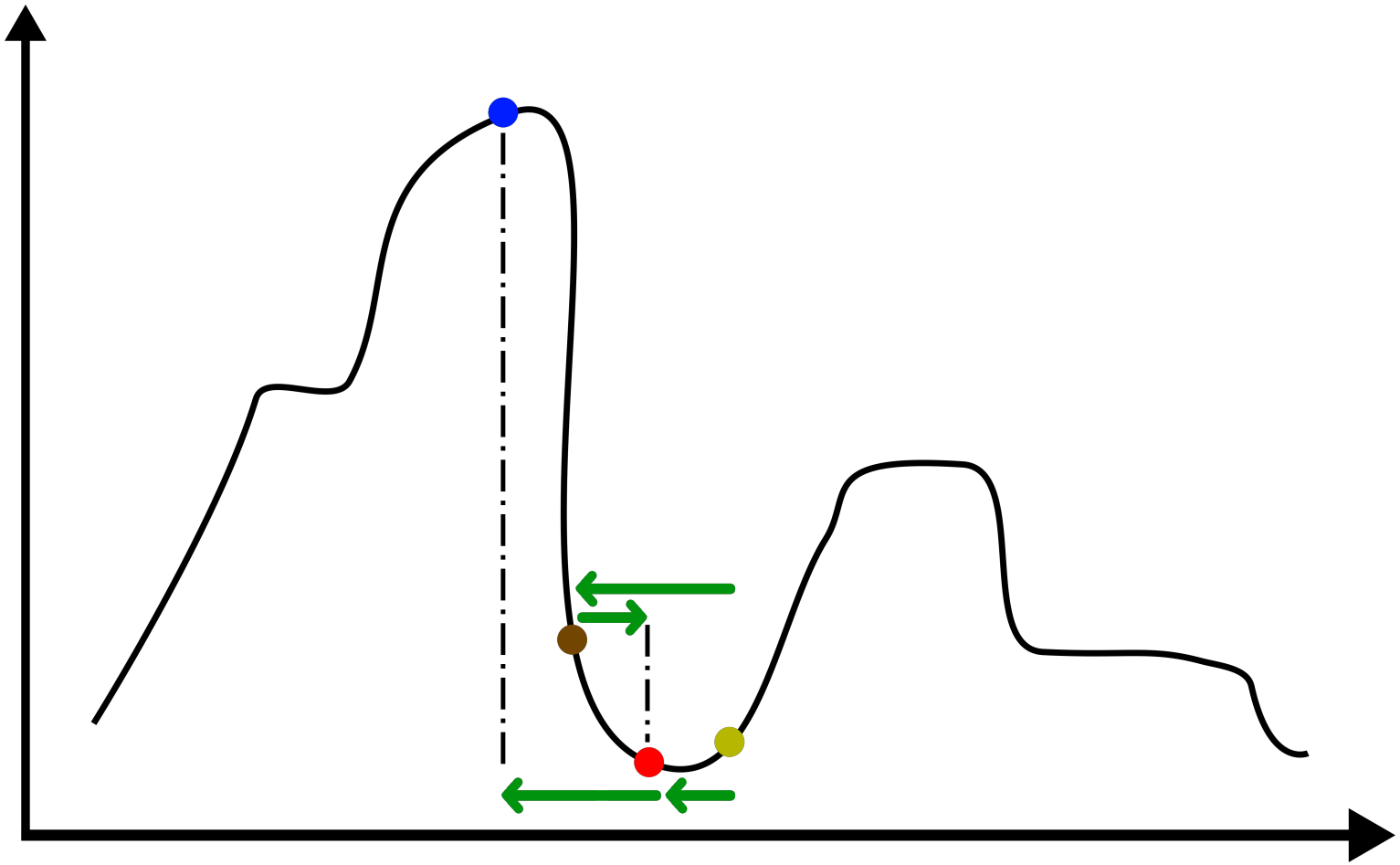


Image from: Geoffrey Hinton's lecture 6c for CSC321

Nesterov Accelerated Gradient (NAG)



Adagrad

Adaptive Subgradient Methods for Online Learning and Stochastic Optimization

John Duchi, Elad Hazan, and Yoram Singer
JMLR, 2011

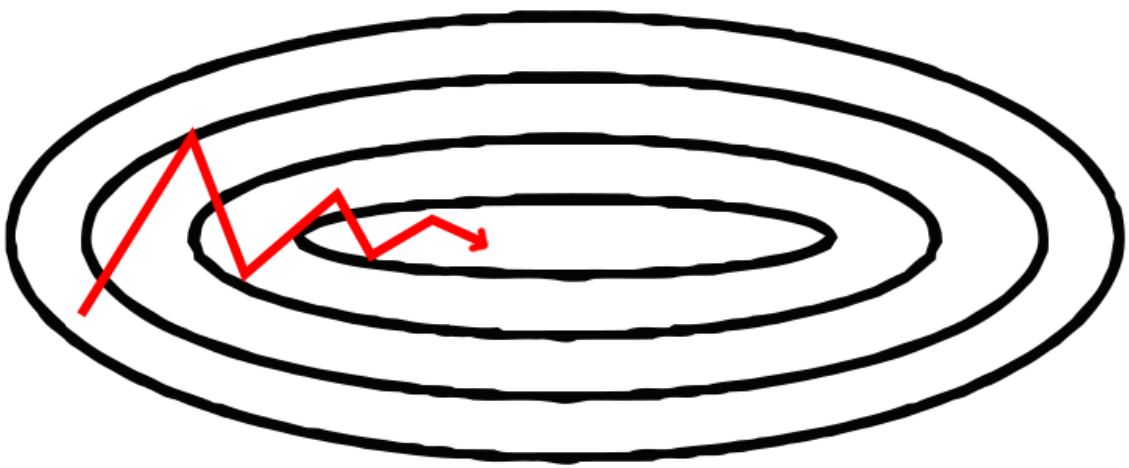
New notation: $g_t = \nabla_{\theta_t} L(\theta_t)$

Adagrad: $\Delta\theta_t = \frac{\lambda}{\sqrt{G_t + \epsilon}} g_t$

G_t is a diagonal matrix; the element $G_{t,i,i}$ is the sum of the squares of $g_{:,i}$, and $\epsilon \approx 1e-8$ is a term to avoid division by zero;

Adagrad

[☺] Adaptable to the parameters' frequency;
[☹] Learning rate is monotonically decreasing during training;
[☹] G is monotonically increasing during training.



RMSProp

RMSProp (Root Mean Square Propagation)

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky
not published ; proposed in the lecture 6e of Hinton's Coursera class

$$\text{Adagrad: } \theta_{t+1} = \theta_t - \frac{\lambda}{\sqrt{G_t + \epsilon}} \odot g_t$$

$$\text{RMSProp: } \theta_{t+1} = \theta_t - \frac{\lambda}{\sqrt{E[g^2]_t + \epsilon}} \odot g_t$$

Where $E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$
And can be seen as a decaying G.

[☺] Learning rate is not monotonically decreasing during training;

Adadelta found a problem in the paradise

ADADELTA: An Adaptive Learning Rate Method

Matthew D. Zeiler
arXiv 2012

$$\text{GD/GDM: } \Delta\theta_t = \lambda g_{t,i} \propto \frac{1}{\text{units of } \theta_t}$$

$$\text{Adagrad/RMSProp: } \Delta\theta_t = \frac{\lambda}{\sqrt{E[g^2]_t + \epsilon}} \odot g_t \propto \text{unitless}$$

$$\text{Reminder: } \theta_{t+1} = \theta_t - \Delta\theta_t$$

But the units for θ_t and $\Delta\theta_t$ **should match**. For GD/GDM, units of $\Delta\theta_t$ are related to the gradient, not the parameters:

$$\text{units of } \Delta\theta_t \propto \frac{\partial L}{\partial \theta_t} \propto \frac{1}{\text{units of } \theta_t}$$

Adadelta's approximation

Newton's method: $\Delta\theta_t = H_t^{-1} g_t \approx \frac{\frac{\partial L}{\partial \theta_t}}{\frac{\partial^2 L}{\partial \theta_t^2}} \propto \frac{\frac{1}{\text{units of } \theta_t}}{\frac{1}{\text{units of } \theta_t^2}} \propto \text{units of } \theta_t$

Manipulation: $\Delta\theta_t \approx \frac{\frac{\partial L}{\partial \theta_t}}{\frac{\partial^2 L}{\partial \theta_t^2}} \Rightarrow \frac{1}{\frac{\partial^2 L}{\partial \theta_t^2}} = \frac{\Delta\theta_t}{\frac{\partial L}{\partial \theta_t}} = diag(H)^{-1}$

$$\Delta\theta_t = \frac{\sqrt{E[\Delta\theta^2]_{t-1} + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} \odot g_t \approx diag(H)_t^{-1} g_t$$

Where $E[g^2]_t$ is the decaying average of the gradients and $E[\Delta\theta^2]_{t-1}$ is the decaying average of previous $\Delta\theta$'s.

- [☺] Robust to the choice of hyperparameters;
- [😐] Assumes L(·) is unitless;
- [☹] No learning rate (e.g. for fine-tuning).

Adaptative Momentum Estimation (Adam)

UPMC SORBONNE UNIVERSITÉS

Adam: A Method for Stochastic Optimization

Diederik P. Kingma, and Jimmy Lei Ba (equal contrib.)
ICLR 2015

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Adaptative Momentum Estimation (Adam)

Adam: A Method for Stochastic Optimization

Diederik P. Kingma, and Jimmy Lei Ba (equal contrib.)
ICLR 2015

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad , \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Adaptative Momentum Estimation (Adam)

Adam: A Method for Stochastic Optimization

Diederik P. Kingma, and Jimmy Lei Ba (equal contrib.)
ICLR 2015

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad , \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\Delta\theta_t = \frac{\lambda}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t = \lambda\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$$

Recommended values
$\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and $\lambda = 10^{-3}$

Adagrad, RMSProp and Adam

$$\text{Adagrad: } \Delta\theta_t = \frac{\lambda}{\sqrt{G_t+\epsilon}} \odot g_t = \lambda \frac{g_t}{\sqrt{G_t+\epsilon}}$$

$$\text{RMSProp: } \Delta\theta_t = \frac{\lambda}{\sqrt{E[g^2]_t+\epsilon}} \odot g_t = \lambda \frac{g_t}{\sqrt{E[g^2]_t+\epsilon}}$$

$$\text{Adam: } \Delta\theta_t = \frac{\lambda}{\sqrt{\hat{v}_t}+\epsilon} \hat{m}_t = \lambda \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \text{ (from } g_t\text{)} \quad , \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \text{ (from } g_t^2\text{)}$$

Adagrad, Nesterov and Adam

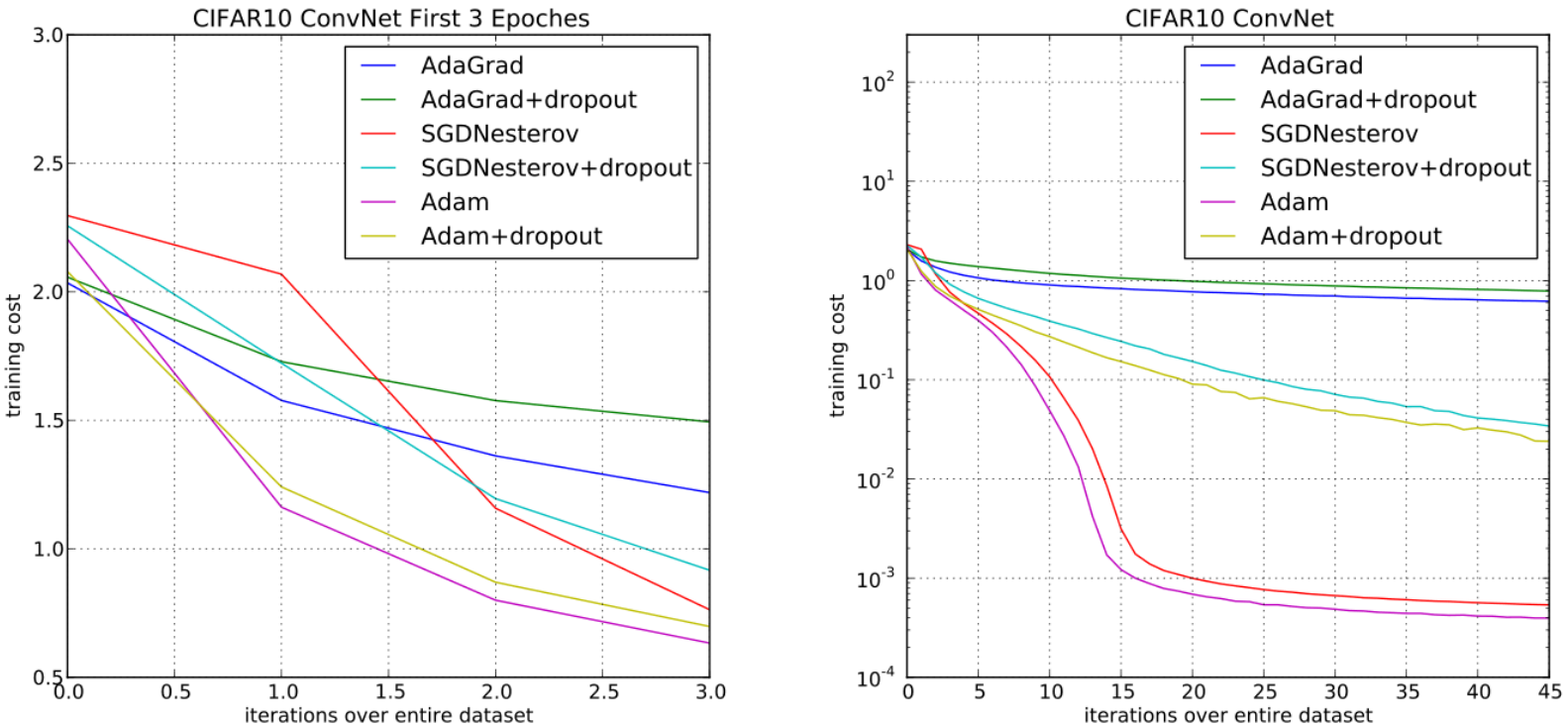


Image from: Diederik et al. 2015

Summary

GD: $\Delta\theta_t = \lambda g_t$

GDM: $\Delta\theta_t = \gamma\Delta\theta_{t-1} + \lambda g_t$

NAG: $\Delta\theta_t = \gamma\Delta\theta_{t-1} + \lambda\nabla_{\theta_t} L(\theta_t - \gamma\Delta\theta_{t-1})$

Adagrad: $\Delta\theta_t = \frac{\lambda}{\sqrt{G_t+\epsilon}} \odot g_t$

RMSProp: $\Delta\theta_t = \frac{\lambda}{\sqrt{E[g^2]_t+\epsilon}} \odot g_t$

Adadelta: $\Delta\theta_t = \frac{\sqrt{E[\Delta\theta^2]_{t-1}}}{\sqrt{E[g^2]_t+\epsilon}} \odot g_t$

Adam: $\Delta\theta_t = \frac{\lambda}{\sqrt{\ hatv_t+\epsilon}} \hat{m}_t$

Thank you !

Micael Carvalho (micael.carvalho@lip6.fr)

Laboratoire d'Informatique de Paris 6; Sorbonne Universités, UPMC, Paris, France

Inspired by the blog post of Sebastian Ruder: http://ruder.io/optimizing-gradient-descent/