An Overview of Gradient Descent Optimization Algorithms

by Sebastian Ruder

Group 7

Myung-Hwan Song

Index

Gradient descent variants

- **Batch gradient descent**

  Use whole dataset to compute gradient

- **Stochastic gradient descent**

  Use one data sample to compute gradient

- **Mini-batch gradient descent**

  Use few number of samples(mini-batch) to compute gradient

Challenges

- **Choosing step size (learning rate)**

  Annealing, scheduling

- **Same step size applies to all parameters**

  Adaptive learning rate strategy

- **Avoiding local minima / saddle point**

  Avoiding saddle point is more critical!

Momentum

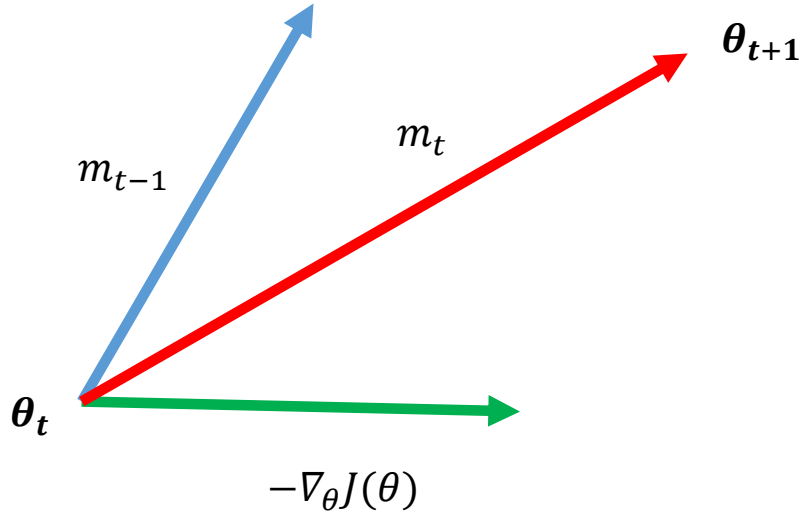
- **Update Rule**

$$m_t = \gamma m_{t-1} + \eta \nabla_\theta J(\theta)$$

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

- **What it does**

Use momentum term(m) to compute next move
We can expect higher training speed



Nesterov's Accelerated Gradient (NAG)

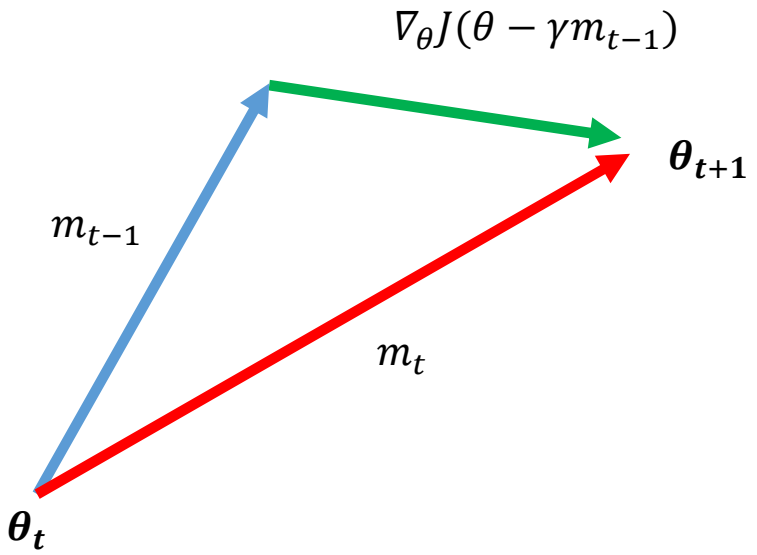
- **Update Rule**

$$m_t = \gamma m_{t-1} + \eta \nabla_\theta J(\theta - \gamma m_{t-1})$$

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

- **What it does**

Compute gradient direction
after moving to past momentum direction



Adaptive Learning Rate

- **Recall the second challenge**

  Same learning rate applies to all parameters

- **When dataset is sparse this can be a problem**

  Some parameters will converge very slowly

- **If we apply different step size to different parameter**

  -> Adaptive learning rate

Adagrad

Duchi et al., (2010)

- **Update Rule**

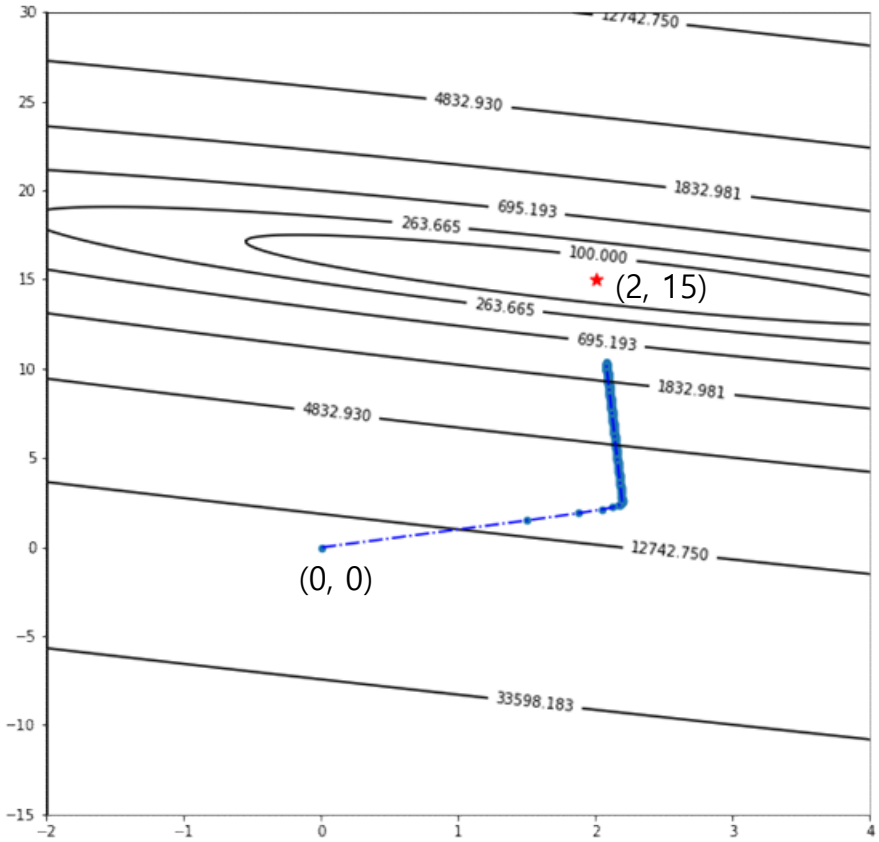
$$G_{t,i} = G_{t-1,i} + \left(\nabla_{\theta_t} j(\theta_{t,i})\right)^2$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,i} + \epsilon}} \nabla_{\theta_t} J(\theta_{t,i})$$

- **What it does**

  Moved small in the past → far away from the optimal

  Moved a lot in the past → maybe near the optimal



RMSprop

Geoffery Hinton et al., 2012

- **Update Rule**

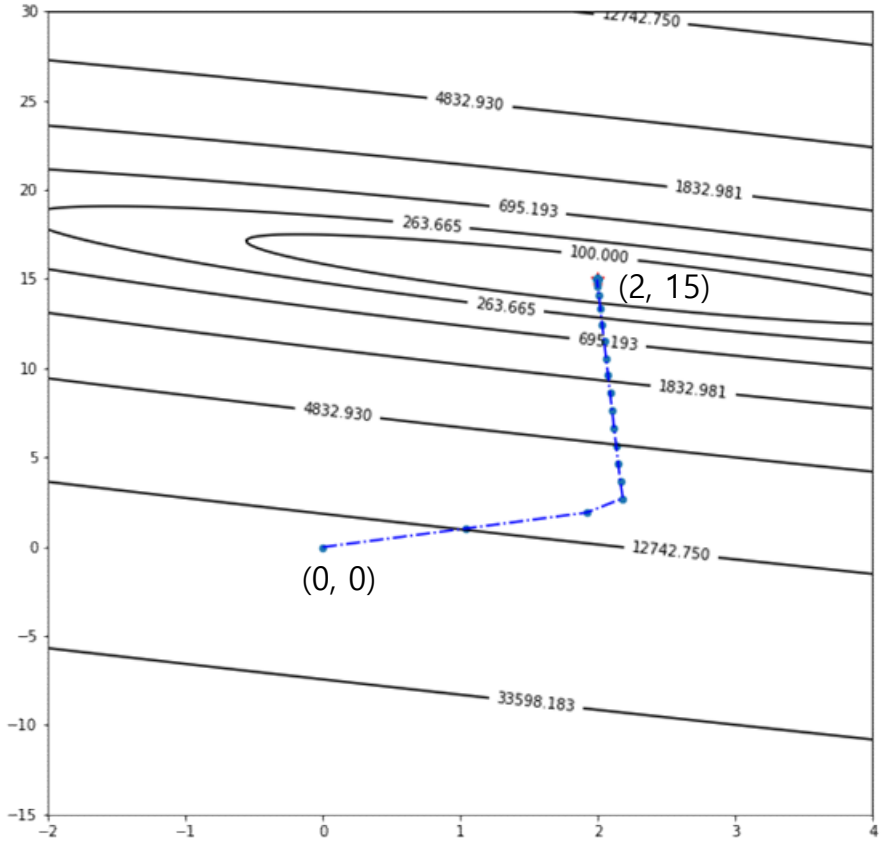
$$G_{t,i} = \gamma G_{t-1,i} + (1-\gamma)\left(\nabla_{\theta_t} j(\theta_{t,i})\right)^2$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,i}+\epsilon}} \nabla_{\theta_t} J(\theta_{t,i})$$

- **What it does**

  Moving average of squared gradient descent

  Therefore, G can get smaller!



Adadelta

Matthew D. Zeiler (2012)

- **Update Rule**

$$G_{t,i} = \gamma G_{t-1,i} + (1-\gamma)\left(\nabla_{\theta_t} j(\theta_{t,i})\right)^2$$ ← Moving average of squared gradient

$$s_t = \gamma s_{t-1} + (1-\gamma)\Delta_{\theta_t}^2$$ ← **Moving average of squared delta**

$$\Delta_{\theta_t} = -\frac{\sqrt{s_{t-1}+\epsilon}}{\sqrt{G_{t,ii}+\epsilon}}\nabla_{\theta_t} J(\theta_{t,i})$$ ← Delta decides how much you update the parameter

$$\theta_{t+1} = \theta_t + \Delta_{\theta_t}$$

Adadelta – approximate Hessian!

Matthew D. Zeiler (2012)

- **Why using squared delta?**
- **In Newton Method…**

  $(g = \nabla j, H = \nabla^2 J)$

  assume J has no unit

$$\Delta_\theta \propto H^{-1} g \propto \frac{\frac{\partial J}{\partial \theta}}{\frac{\partial^2 J}{\partial \theta^2}} \propto \partial\theta \propto unit(\theta)$$

← Right relation

- **However, in Gradient Method…**

$$\Delta_\theta \propto g \propto \frac{\partial J}{\partial \theta} \propto \frac{1}{\partial \theta} \propto \frac{1}{unit(\theta)}$$

← Wrong relation!

- **Therefore…**

$$\Delta_\theta = \frac{\frac{\partial J}{\partial \theta}}{\frac{\partial^2 J}{\partial \theta^2}} \Rightarrow H^{-1} = \frac{1}{\frac{\partial^2 J}{\partial \theta^2}} = \frac{\boldsymbol{\Delta_\theta}}{\boldsymbol{\frac{\partial J}{\partial \theta}}}$$

$$\theta_{t+1} = \theta_t + -\frac{\sqrt{\boldsymbol{s_{t-1} + \epsilon}}}{\sqrt{\boldsymbol{G_{t,ii} + \epsilon}}} \nabla_{\theta_t} J(\theta_{t,i})$$

Adam

Kingma & Ba, 2015

- **Update Rule**

$$g_t = \nabla_\theta J(\theta)$$

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

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

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$ ← Bias correction Term

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Adamax

Kingma & Ba, 2015

- **Update Rule**

$$g_t = \nabla_\theta J(\theta)$$

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

$$u_t = \max(\beta_2 u_{t-1}, |g_t|)$$ ← Max norm of gradient

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$ ← No need to correct bias

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{u_t}$$

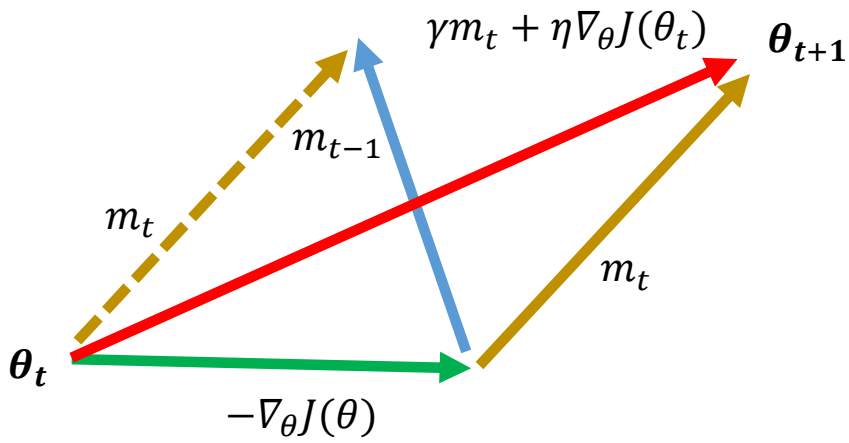
Nadam

Timothy Dozat (2016)

- **Modified NAG**

$$m_t = \gamma m_{t-1} + \eta \nabla_\theta J(\theta_t)$$

$$\theta_{t+1} = \theta_t - (\gamma m_t + \eta \nabla_\theta J(\theta_t))$$



- **What it does**

  Use **current momentum** and current gradient to decide direction

Nadam

Timothy Dozat (2016)

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} = \theta_{t-1} - \eta \frac{1}{\sqrt{\hat{v}_t} + \epsilon} (\beta_1 \hat{\boldsymbol{m}}_{\boldsymbol{t-1}} + \frac{(1-\beta_1)\nabla_\theta J(\theta_t)}{1-\beta_1^t})$$

Change momentum term with modified NAG

$$\theta_t = \theta_{t-1} - \eta \frac{1}{\sqrt{\hat{v}_t} + \epsilon} (\beta_1 \hat{\boldsymbol{m}}_{\boldsymbol{t}} + \frac{(1-\beta_1)\nabla_\theta J(\theta_t)}{1-\beta_1^t})$$

Which Optimizer to Use?

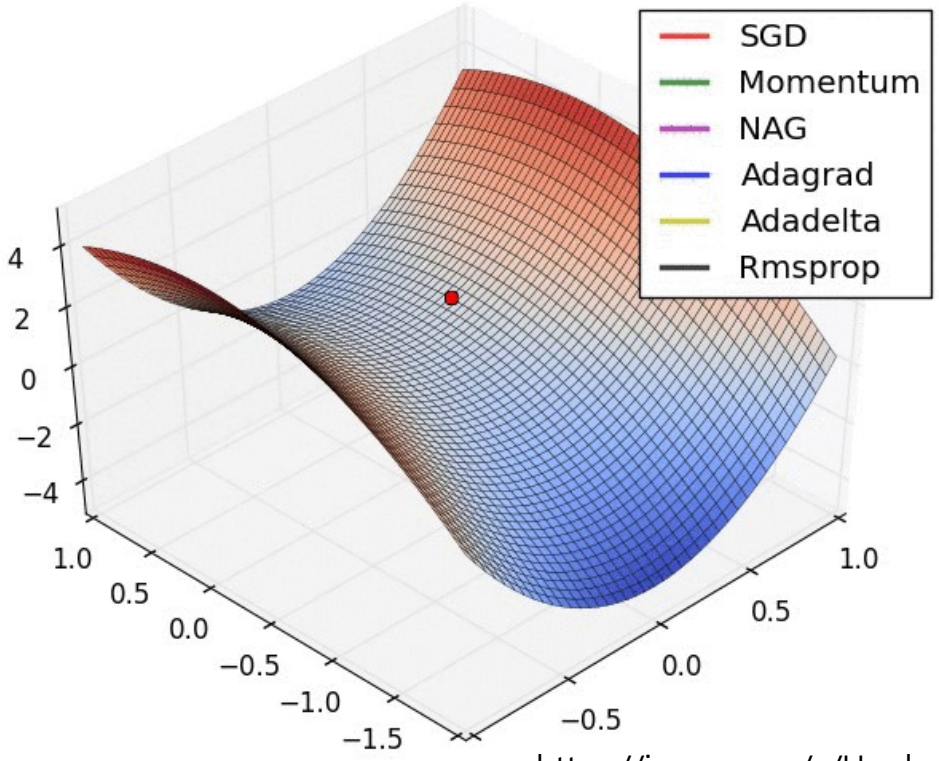
- **Not surprisingly, no exact answer**
- **But generally…**

  Momentum strategy -> high training speed

  If dataset is sparse -> adaptive learning rate

- **Adam?**

  Kind of default algorithm to use, however…



https://imgur.com/a/Hqolp

Reference

- Sebastian Ruder, An overview of gradient descent optimization algorithms, arXiv:1609.04747v2 [cs.LG] 15 Jun 2017
- Diederik P. Kingma et al. Adam: A method for stochastic optimization, arXiv:1412.6980v9 [cs.LG] 30 Jan 2017 Published
- Ilya Loshchilov & Frank Hutter, FixingWeight Decay Regularization in Adam, arXiv:1711.05101v1 [cs.LG] 14 Nov 2017 FixingWeight
- Matthew D. Zeiler, Adadelta: An Adaptive Learning Rate Method, arXiv:1212.5701v1 [cs.LG] 22 Dec 2012 ABSTRACT
- Timothy Dozat, Incorporating Nesterov Momentum Into Adam, ICLR 2016
- http://ruder.io/optimizing-gradient-descent/
- http://cs231n.github.io/optimization-1/

Thank you!