ADAM : A METHOD FOR STOCHASTIC OPTIMIZATION

Kingma, Jimmy

2021.05.25
임진혁

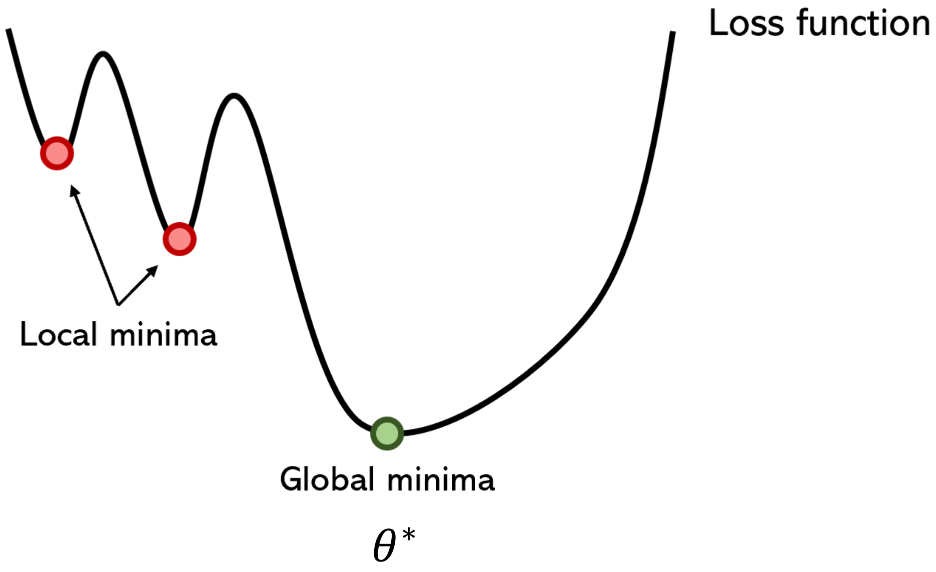
Index

eeplearning@unist.ac.kr<임진혁>

(1) Optimizer

- To solve optimization problem

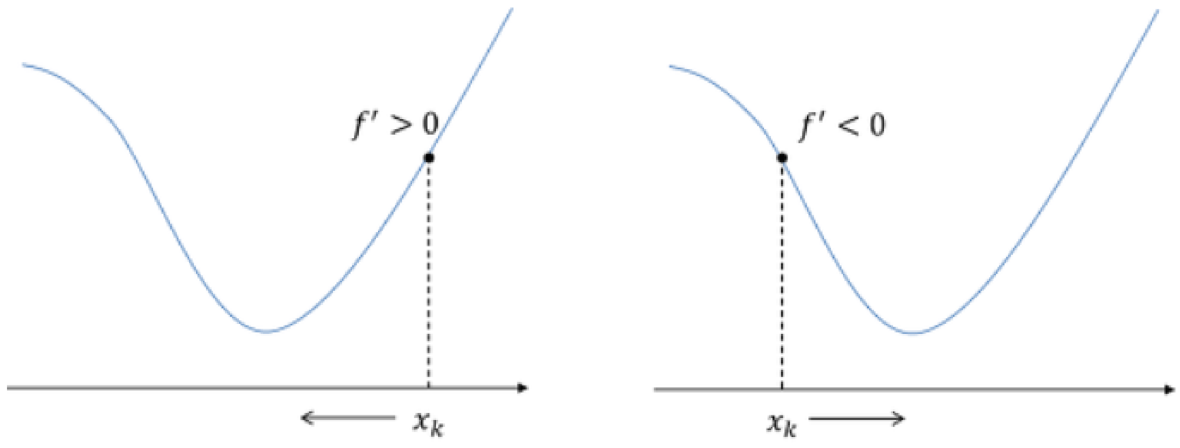
- Find optimal parameter which is make minimum loss



1 Adam Introduction

(1) Optimizer

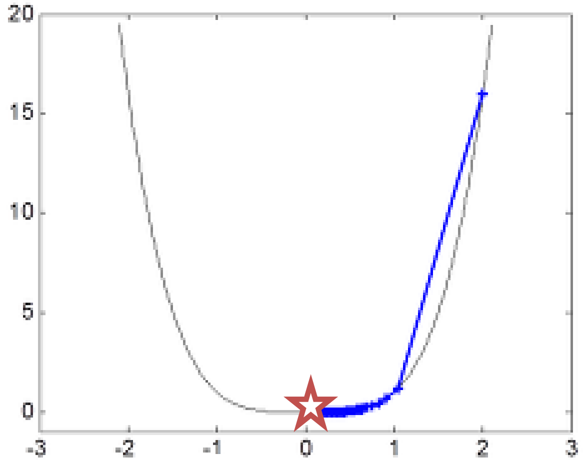
Direction towards low Loss



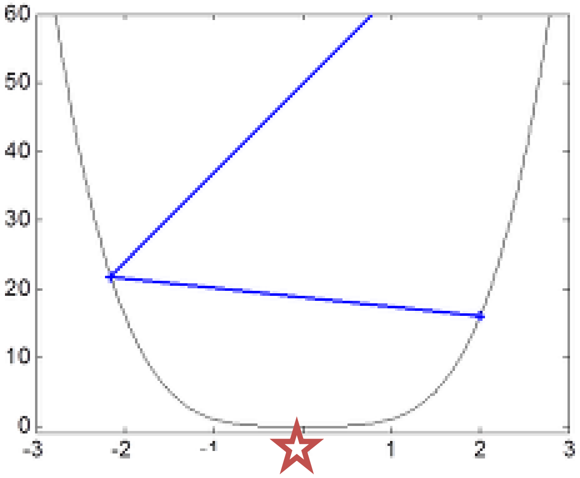
$$\theta_{t+1} = \theta_t - \alpha \cdot g_t$$

(1) Optimizer

Move as much as Step size



α = 0.03



α = 0.13

$$\theta_{t+1} = \theta_t - \alpha \cdot g_t$$

eeplearning@unist.ac.kr<임진혁>

(2) Gradient descent

- Batch gradient descent: Entire training sample

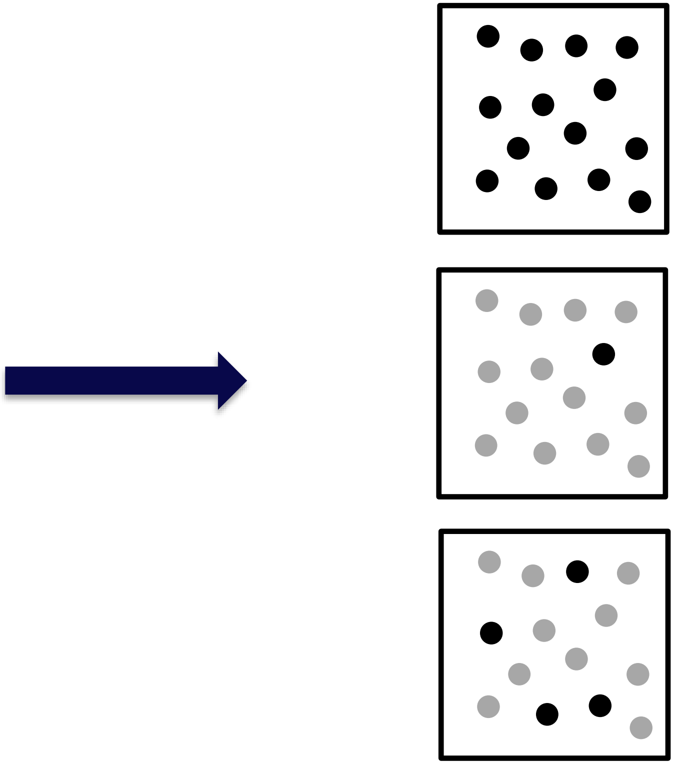
$$\theta_t = \theta_{t-1} - \alpha \nabla_\theta(\theta)$$

- Stochastic gradient descent: Each training sample

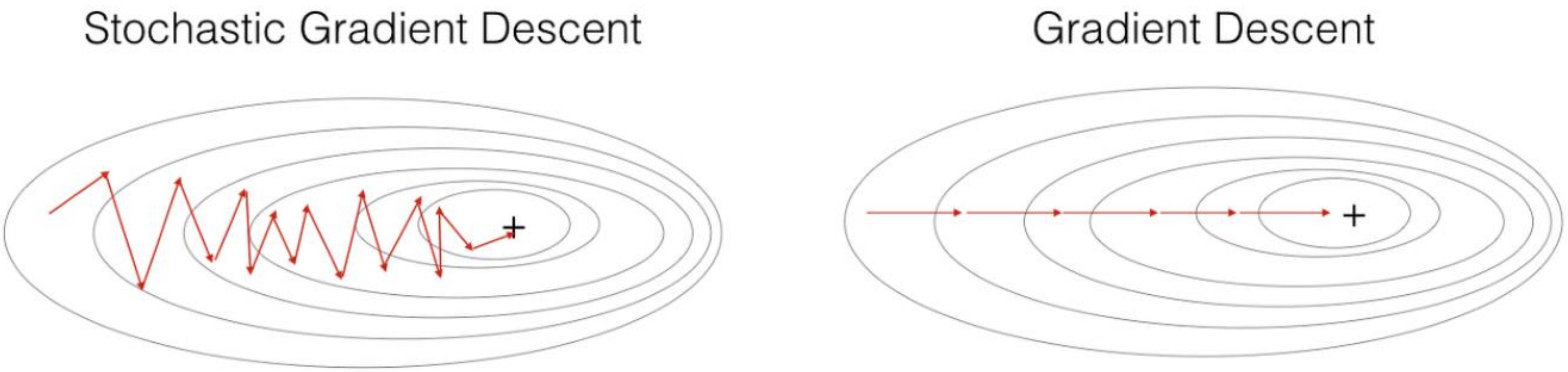
$$\theta_t = \theta_{t-1} - \alpha \nabla_\theta(\theta; x^{(i)}; y^{(i)})$$

- Mini-batch gradient descent: n training sample

$$\theta_t = \theta_{t-1} - \alpha \nabla_\theta(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$



Stochastic Optimization Problem



https://jypng.tistory.com/26#footnote_26_5

eeplearning@unist.ac.kr<임진혁>

Introduction

Stochastic Optimization Problem

- Jitter
- Saddle point
- Mini batch noise

https://jypng.tistory.com/26#footnote_26_5

eeplearning@unist.ac.kr<임진혁>

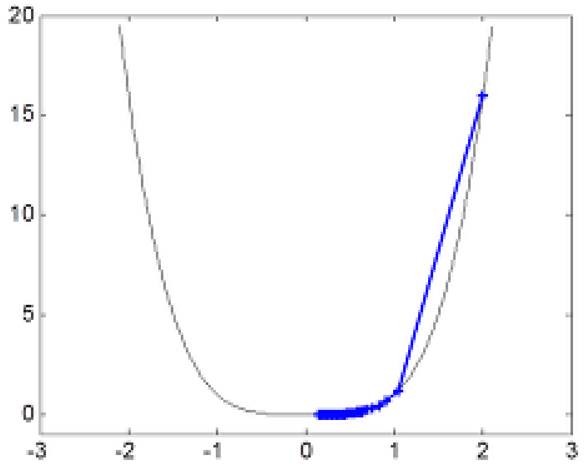
(2) Gradient descent

$$\theta_t = \boxed{\theta_{t-1}} - \boxed{\alpha} \cdot \boxed{\nabla_\theta f(\theta)}$$

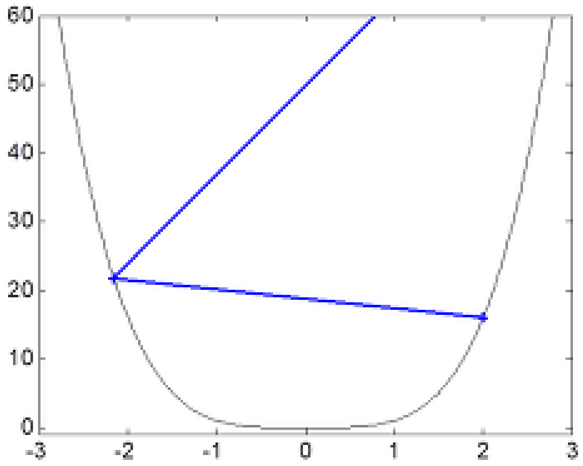
Previous parameter

Stepsize

Gradient



$\alpha = 0.03$



$\alpha = 0.13$

Problem: Parameters are updated only by depending on gradient

eeplearning@unist.ac.kr<임진혁>

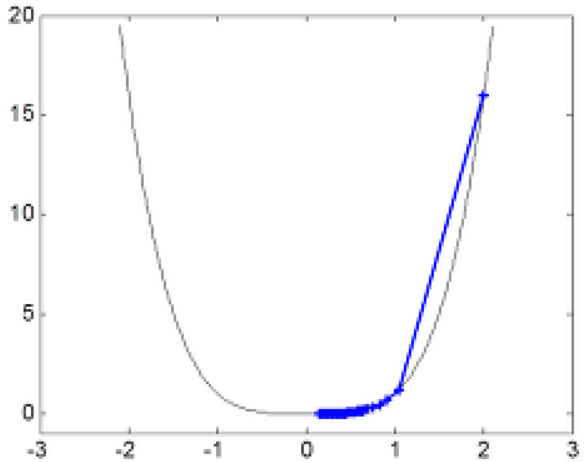
(2) Gradient descent

$$\theta_t = \boxed{\theta_{t-1}} - \boxed{\alpha} \cdot \boxed{\nabla_\theta f(\theta)}$$

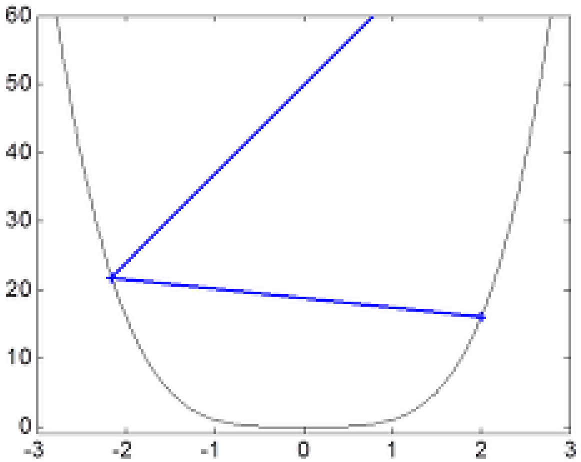
Previous parameter

Stepsize

Gradient



$\alpha = 0.03$



$\alpha = 0.13$

⇒ **Key Point: Not rely on current gradient and adjust step size**

Related works

(1) Momentum

$$v_t = \underbrace{\boxed{\gamma v_{t-1}}}_{\text{Momentum}} + \alpha \nabla_\theta J(\theta) \qquad (\gamma < 1)$$

$$\theta_t = \theta_{t-1} - v_t$$

$$v_t = \alpha \nabla_\theta J(\theta)_t + \boxed{\gamma} \alpha \nabla_\theta J(\theta)_{t-1} + \boxed{\gamma^2} \alpha \nabla_\theta J(\theta)_{t-2} + \cdots$$

- To Solve the prior problem
- Consider the previous parameter direction
- Focus on the latest gradients

(2) Adagrad

$$g_{t,i} = \nabla_\theta J(\theta_i)$$

- The gradient of the objective function to θ_i at t

$$G_t = G_{t-1} + (\nabla_\theta J(\theta_t))^2$$

- Sum of the squares of the gradients

$$\theta_{t+1} = \theta_t - \boxed{\frac{\alpha}{\sqrt{G_t + \varepsilon}}} \cdot \nabla_\theta J(\theta_t)$$

- How can control the stepsize
- Update parameters differently (element-wise product)
- However, the stepsize can be reduced

(3) RMSProp

$$G_t = \gamma G_{t-1} + \boxed{(1 - \gamma)}(\nabla_\theta J(\theta_t))^2$$

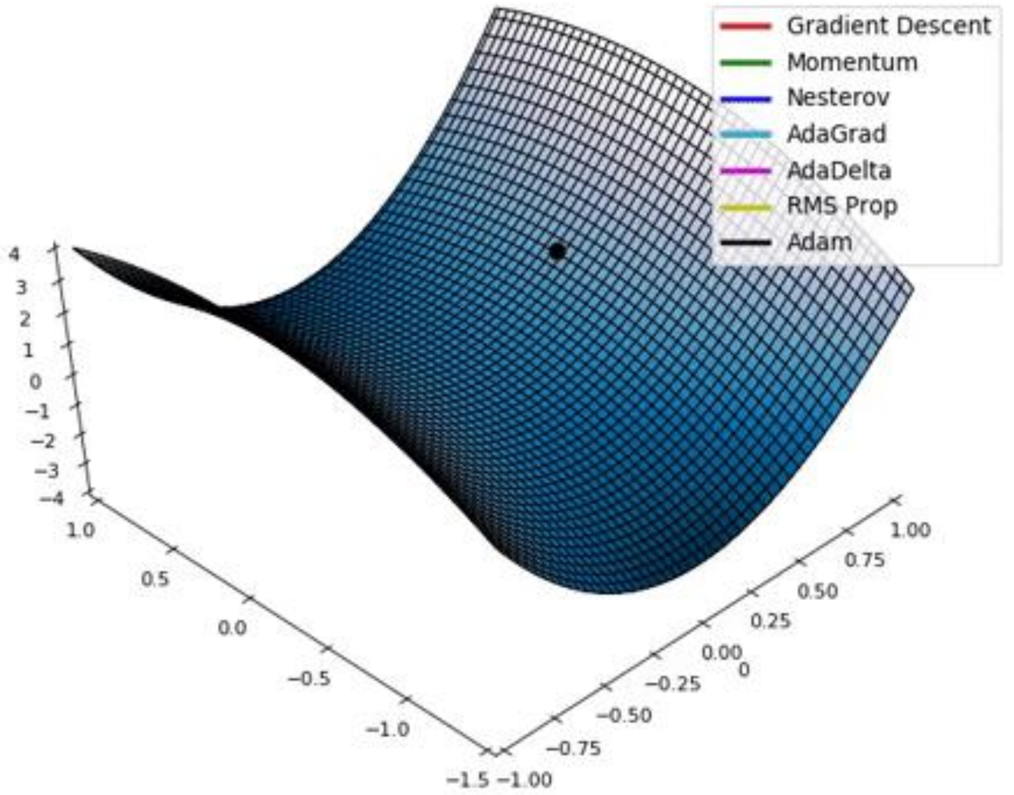
$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{G_t + \varepsilon}} \cdot \nabla_\theta J(\theta_t)$$

- To solve the problem of reducing stepsize
- Exponentially decaying average of squared gradients
- Well-suited for dealing with non-stationary data

Proposed Method: ADAM

Adam: "Efficient" optimization method

- High-dimensional parameter
- Little memory requirement
- Naturally step size annealing
- Invariant to re-scaling of gradient



https://jypng.tistory.com/26#footnote_link_26_12

Proposed Method: ADAM

Adam: Adaptive Moment estimation

- Adaptive
- Moment
- Estimation

$$g_t = \nabla f_t(\theta_{t-1})$$
$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\boxed{\theta = \theta_{t-1} - \alpha \frac{m_t}{v_t + \epsilon}}$$

3 Adam Proposed Method: ADAM

Adam: Adaptive Moment estimation

- Adaptive
  - individual adaptive step size

- Moment
  - 1^{st} moment(Mean) : m
  - 2^{nd} mement(Variance) : v

- Estimation
  - Decay Rates $\beta 1$, $\beta 2$ (exponential moving average)

$$g_t = \nabla f_t(\theta_{t-1})$$
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\boxed{\theta = \theta_{t-1} - \alpha \frac{m_t}{v_t + \epsilon}}$$

Proposed Method: ADAM

Adam: Momentum + RMSprop

Momentum

$$v_t = \gamma v_{t-1} + \alpha g_t$$
$$\theta_t = \theta_{t-1} - v_t$$

Sparse gradient , Saddle point, Local minima with momentum

RMSprop

$$v_t = \gamma v_{t-1} + (1-\gamma) g_t^2$$
$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{v_t + \epsilon}} g_t$$

Adaptive learning rate methods : Less Update , Large Stepsize

Proposed Method: ADAM

Adam: Momentum + RMSprop

1st moment (Mean)

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

2nd moment (Variance)

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

$$\boxed{\theta = \theta_{t-1} - \alpha \frac{m_t}{\sqrt{v_t + \epsilon}}}$$

Momentum

$$v_t = \gamma v_{t-1} + \alpha g_t$$

$$\theta_t = \theta_{t-1} - v_t$$

RMSprop

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

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{v_t + \epsilon}} g_t$$

3 Adam Proposed Method: ADAM

Adam: Momentum + RMSprop

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

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

$$\boxed{\theta = \theta_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon}}$$

```
first_moment = 0
second_moment = 0
while True:
  dx = compute_gradient(x)
  first_moment = beta1 * first_moment  + (1 - beta1) * dx
  second_moment = beta2 * second_moment + (1 - beta2) * dx * dx
  x -= learning_rate * first_moment / (np.sqrt(second_moment) + 1e-7))
```

Momentum

AdaGrad / RMSProp

Sort of like RMSProp with momentum

Proposed Method: ADAM

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\boxed{\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)}$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

- Adaptive
  - individual adaptive step size (*a*)

- Moment
  - 1st moment(Mean) : *m*
  - 2nd mement(Variance) : *v*

- Estimation
  - Decay Rates *β1, β2*

Proposed Method: ADAM

0. Initialization

- Each moment estimate vector is initialized to zero.

1. Loop (each time *t*)
   - (1) Calculate current timestep's gradient

   - (2) Update estimate vector using gradient & decay parameter

   - (3) Update the weight parameter

3 Adam Proposed Method: ADAM

0. Initialization

- Each moment estimate vector is initialized to zero.

$$m_0 = 0$$

$$v_0 = 0$$

```
first_moment = 0
second_moment = 0
```

1. Loop (each time *t*)

```
for t in range(num_iterations):
```

- (1) Calculate current timestep's gradient

$$g_t = \nabla f_t(\theta_{t-1})$$

```
dx = compute_gradient(x)
```

- (2) Update estimate vector using gradient & decay parameter

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

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

```
first_moment = beta1 * first_moment  + (1 - beta1) * dx
second_moment = beta2 * second_moment + (1 - beta2) * dx * dx
```

- (3) Update the weight parameter until convergence

$$\theta = \theta_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon}$$

```
x -= learning_rate * first_moment / (np.sqrt(second_moment) + 1e-7))
```

Proposed Method: ADAM

BIAS CORRECTION : Why divide each moment by (1-Beta)

To approximate well $E[estimate]$ => $E[gradint]$

If *Beta* is close to 1, *each moment estimation values* biased to zero.

So, Use **bias corrected** moment values to update weights

$$\hat{m}_t = m_t / (1 - \beta_1^t)$$

$$\hat{v}_t = v_t / (1 - \beta_2^t)$$

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

Proposed Method: ADAM

BIAS CORRECTION : Why divide each moment by (1-Beta)

To approximate well $E[estimate]$ => $E[gradint]$

$$\mathbb{E}\left[v_t\right] = \mathbb{E}\left[(1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i}\cdot g_i^2\right]$$

$$= \mathbb{E}\left[g_t^2\right]\cdot(1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i}+\zeta$$

$$= \mathbb{E}\left[g_t^2\right]\cdot(1-\beta_2^t)+\zeta$$

Proposed Method: ADAM

BIAS CORRECTION : Why divide each moment by (1-Beta)

To approximate well $E[estimate]$ => $E[gradint]$

$$\boxed{m_1 \leftarrow \beta_1 m_0 + (1-\beta_1) g_1}$$

$$\widehat{m_1} \leftarrow \frac{m_1}{1-\beta_1^1} = \frac{\beta_1 m_0}{1-\beta_1^1} + \frac{(1-\beta_1) g_1}{1-\beta_1^1}$$

$$= 0 + g_1 (\because m_0 = 0)$$

Case of "Bias Correction not applied"

Proposed Method: ADAM

BIAS CORRECTION : Why divide each moment by (1-Beta)

To approximate well $E[estimate]$ => $E[gradint]$

$$m_1 \leftarrow \beta_1 m_0 + (1-\beta_1)g_1$$

$$\widehat{m_1} \leftarrow \frac{m_1}{1-\beta_1^1} = \frac{\beta_1 m_0}{1-\beta_1^1} + \frac{(1-\beta_1)g_1}{1-\beta_1^1}$$
$$= 0 + g_1 (\because m_0 = 0)$$

Case of "Bias Correction applied"

Update Rule

It is important for Adam to choose the step size effectively

The step size has two upper bounds

$$\Delta_t = \alpha \cdot \widehat{m}_t / \sqrt{\hat{v}_t}$$

The first case(sparsity case)

The amount of update change should be made larger by increasing the step size

The second case(general case)

The amount of update change is made small by reducing the step size.

Proposed Method: ADAM

Update Rule

It is important for Adam to choose the step size effectively
The step size has two upper bounds

$$\Delta_t = \alpha \cdot \widehat{m}_t / \sqrt{\hat{v}_t}$$

(i) Frist Case

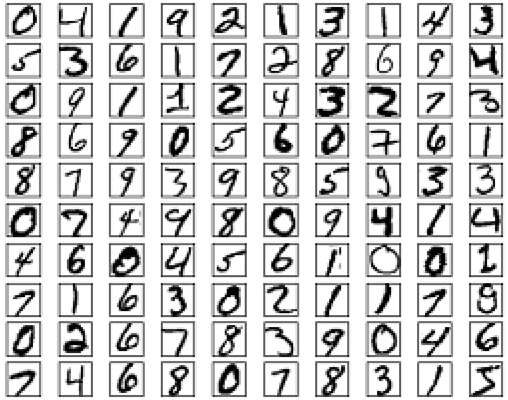
$$(1-\beta_1) > \sqrt{1-\beta_2} \quad ==> \quad |\Delta_t| \leq \alpha \cdot (1-\beta_1) / \sqrt{1-\beta_2}$$

(i) Second Case

$$(1-\beta_1) \langle= \sqrt{1-\beta_2} \quad ==> \quad |\Delta_t| \leq \alpha$$

4 Adam Experiment

Experiment(1): Logistic Regression



MNIST

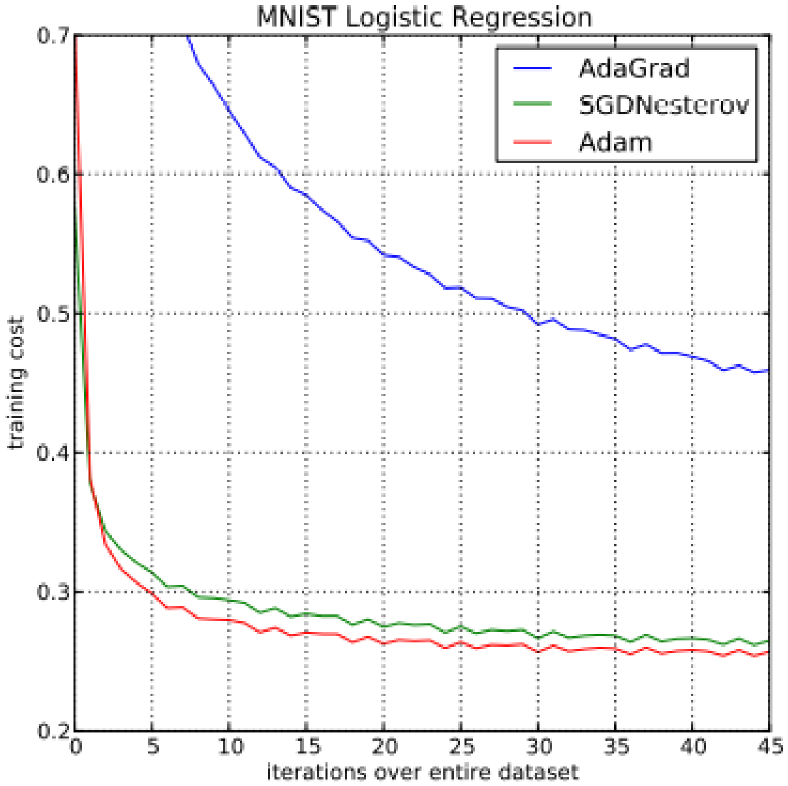
	id	imdb_id	original_title	director	production	genre	cast	budget	revenue	runtime	release_year	v
0	135397	tt0369610	Jurassic World	Colin Trevorrow	Universal Studios	Action	Chris Pratt	150000000	1513528810	124	2015	
1	76341	tt1392190	Mad Max Fury Road	George Miller	Village Roadshow Pictures	Action	Tom Hardy	150000000	378436354	120	2015	
2	262500	tt2908446	Insurgent	Robert Schwentke	Summit Entertainment	Adventure	Shailene Woodley	110000000	295238201	119	2015	
3	140607	tt2488496	Star Wars The Force Awakens	JJ Abrams	Lucasfilm	Action	Harrison Ford	200000000	2068178225	136	2015	
4	168259	tt2820852	Furious	James Wan	Universal Pictures	Action	Vin Diesel	190000000	1506249360	137	2015	

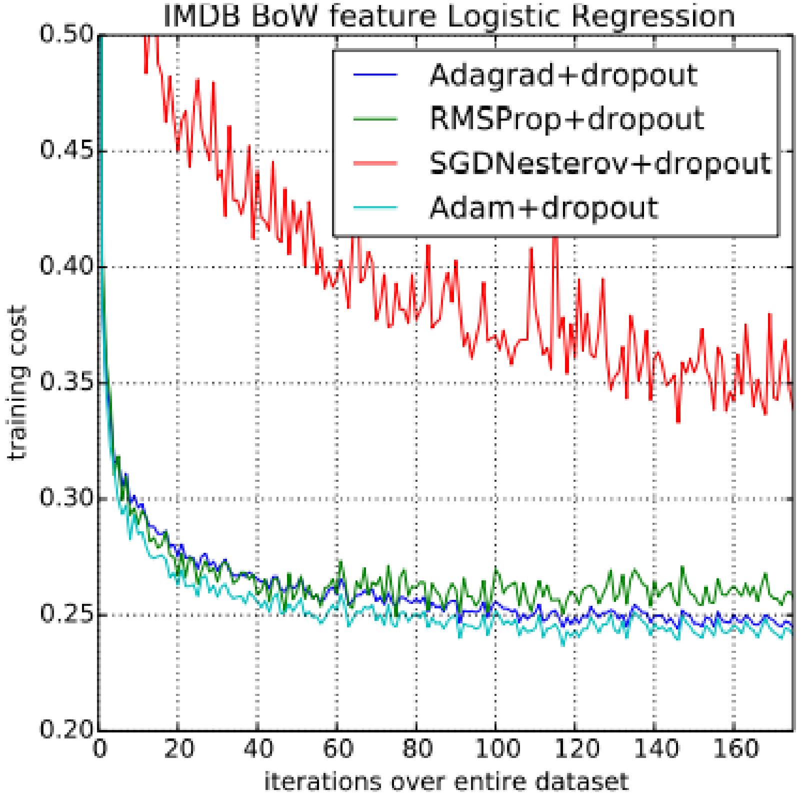
IMDB dataset

(for sparse feature problem)

*Use same parameter initialization
*Hyper-parameters(learning rate, momentum) are searched over dense grid
*Results are reported using the best parameter

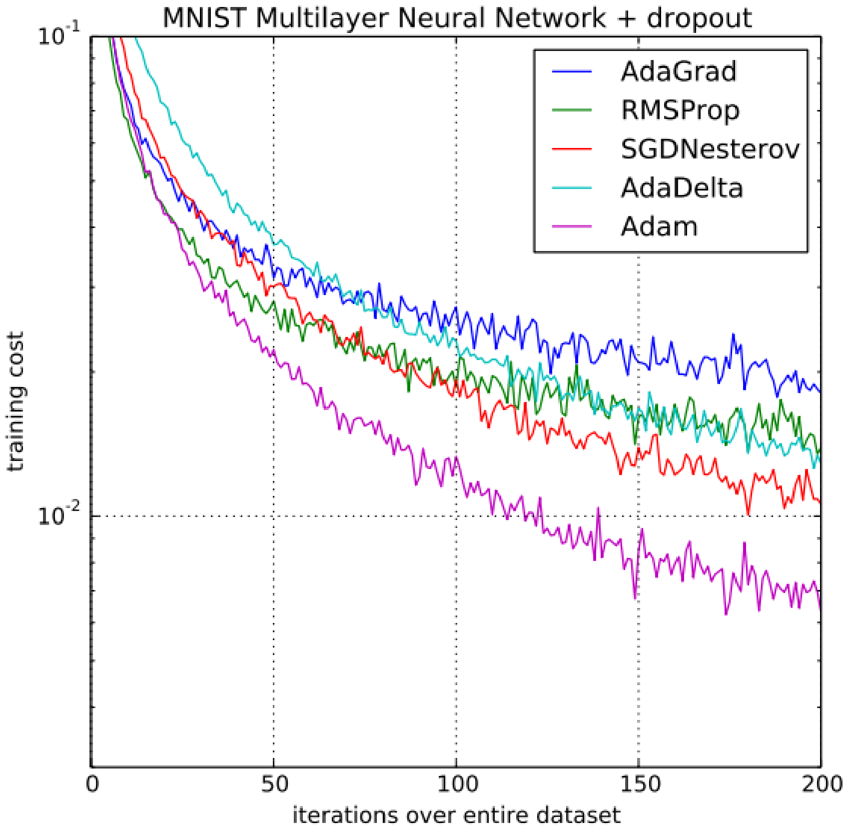
Experiment(1): Logistic Regression





*Use same parameter initialization
*Hyper-parameters(learning rate, momentum) are searched over dense grid
*Results are reported using the best parameter

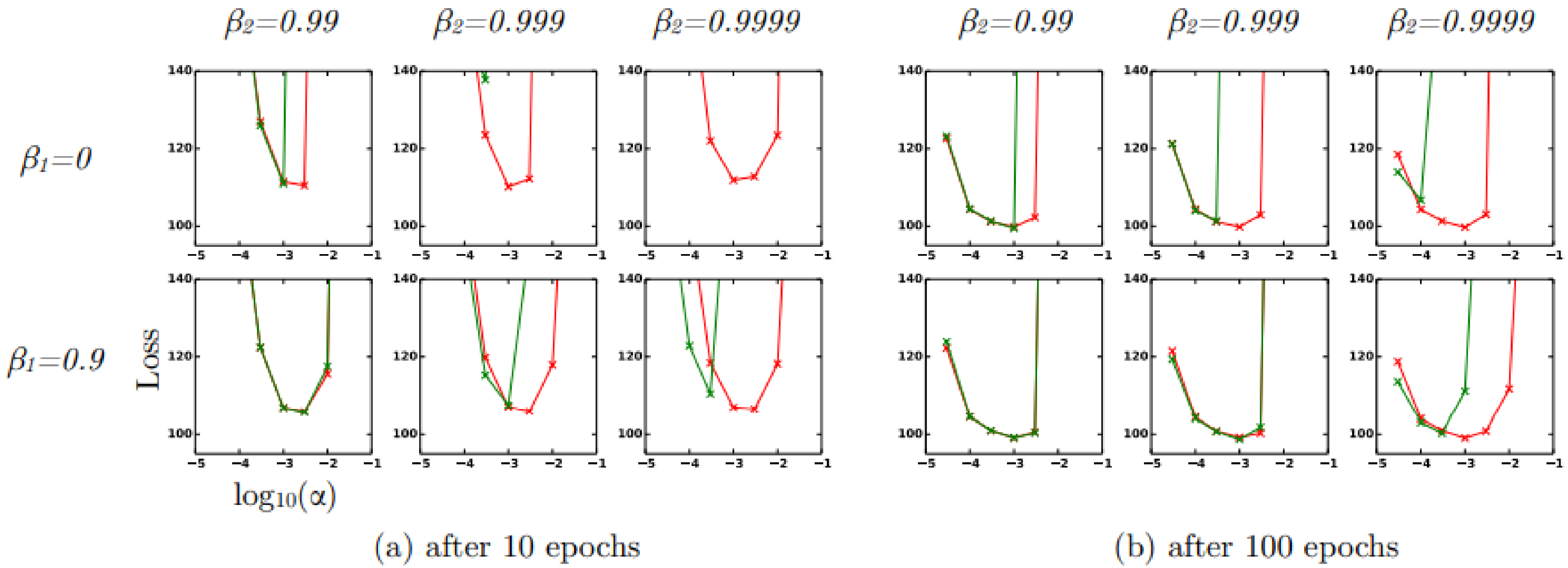
Experiment(2): MULTI-LAYER NEURAL NETWORKS



- two fully connected hidden layer with 1000 hiden units (Relu activation, mini-batch 128 size)
- Cross-Entropy Loss
- Adam most Fast

*Use same parameter initialization
*Hyper-parameters(learning rate, momentum) are searched over dense grid
*Results are reported using the best parameter

Experiment(3): BIAS-CORRECTION TERM



no bias correction terms

bias correction terms

*Use same parameter initialization
*Hyper-parameters(learning rate, momentum) are searched over dense grid
*Results are reported using the best parameter

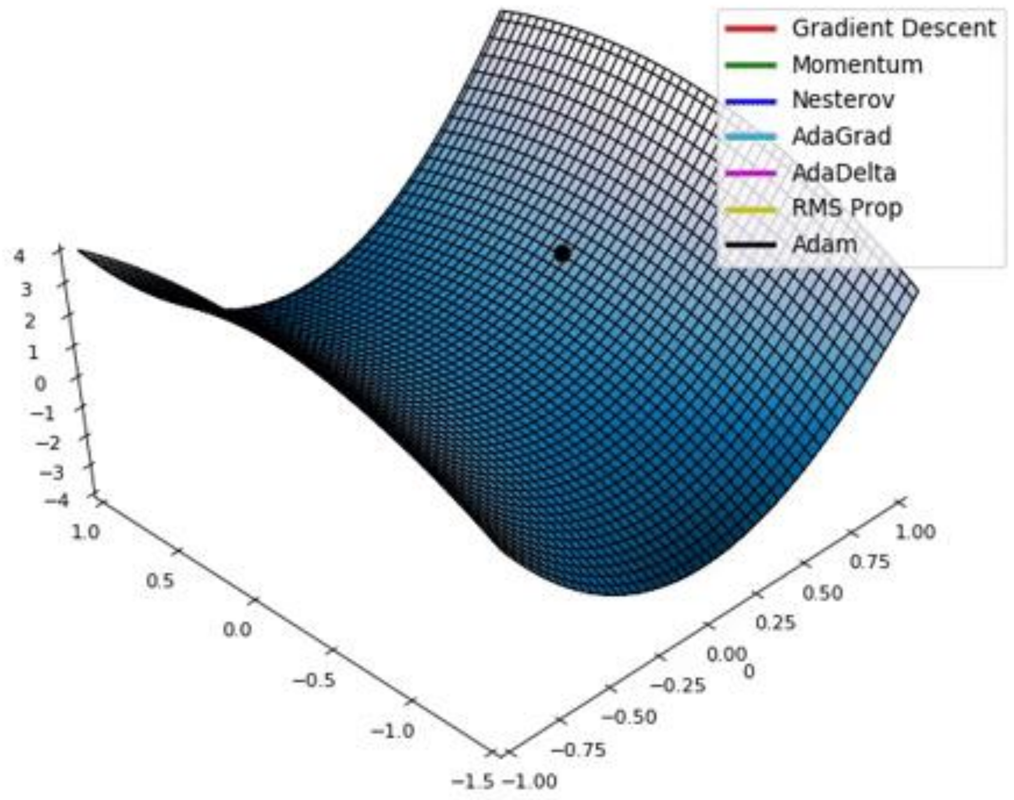
5 Adam Conclusion

- Simple and efficient optimization algorithm

- How AdaGrad(Momentum) handles sparse gradients (different step size for each parameter)

+

- How RMSProp reflects the past slope less than the present



Reference

- P.Kingma, D., Jimmy, Ba. (2015). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. ICLR

- "ADAM Review". https://ropiens.tistory.com/90. (2021.05.13)

Thank you