

# Stochastic Gradient Descent

- Batches and Epochs
- Learning Rate and Momentum
- Nesterov Momentum
- ADAM optimizer

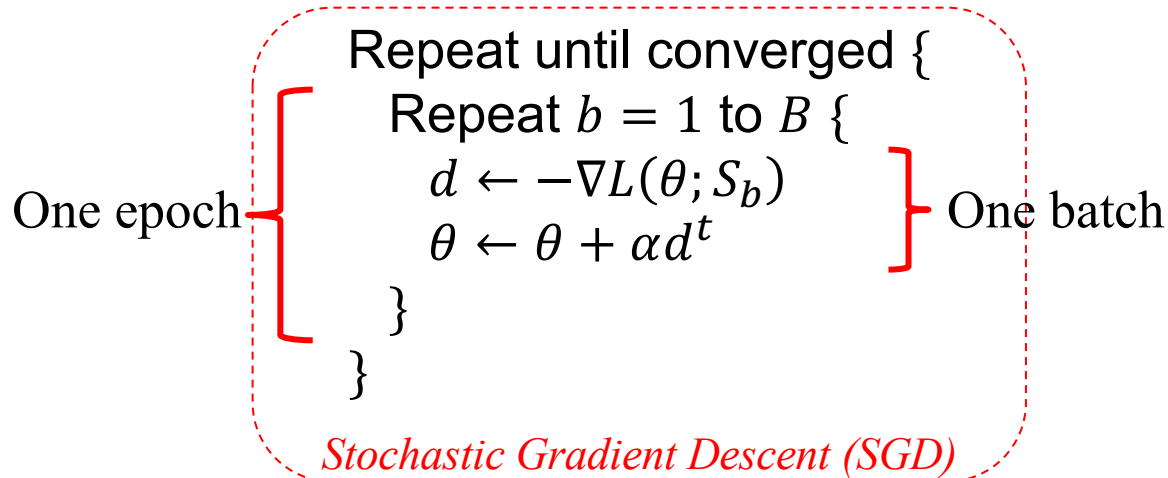
# Batches, Epochs, and Stochastic Gradient Descent (SGD)

- Partition training set into randomized batches

$$S = \{1, \dots, K\} = \bigcup_{b=1}^B S_b \quad \begin{aligned} K_b &= |S_b| \\ &= (\# \text{ of sampler per batch}) \end{aligned}$$

- For each batch you compute a separate gradient

$\nabla L(\theta; S_b) \Leftarrow$  Gradient for  $b^{\text{th}}$  batch of training data



# Theoretical Analysis of SGD

- Assume simple sampling (sampling with replacement)

$$g_k = \nabla L_k(\theta) = (\text{gradient from } k^{\text{th}} \text{ training sample})$$

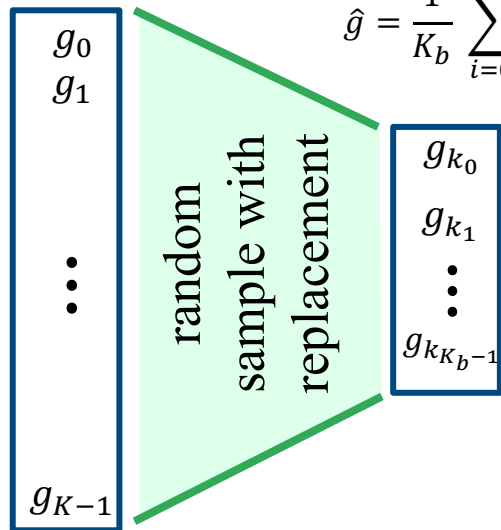
- Each sample,  $g_{k_i}$ , is i.i.d. with distribution  $p(g) = \frac{\text{histogram}(g)}{K}$

True gradient:

$$g = \frac{1}{K} \sum_{k=0}^{K-1} g_k$$

Batch gradient:

$$\hat{g} = \frac{1}{K_b} \sum_{i=0}^{K_b-1} g_{k_i}$$



• Then

$$\hat{g} = g + \frac{w}{\sqrt{K_b}}$$

Batch Gradient      True Gradient      Noise

where

$$E[w] = 0$$

$$\text{Var}[w] \approx \frac{1}{K} \sum_{k=0}^{K-1} (g_k - g)(g_k - g)^t$$

# Effect of Batch Size on SGD

$$\hat{g} = g + \frac{w}{\sqrt{K_b}}$$

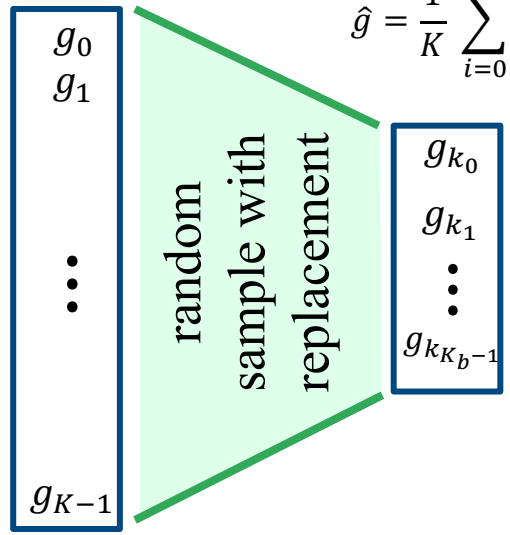
Batch Gradient = True Gradient + Noise

True gradient:

$$g = \frac{1}{K} \sum_{k=0}^{K-1} g_k$$

Batch gradient:

$$\hat{g} = \frac{1}{K} \sum_{i=0}^{K_b-1} g_{k_i}$$

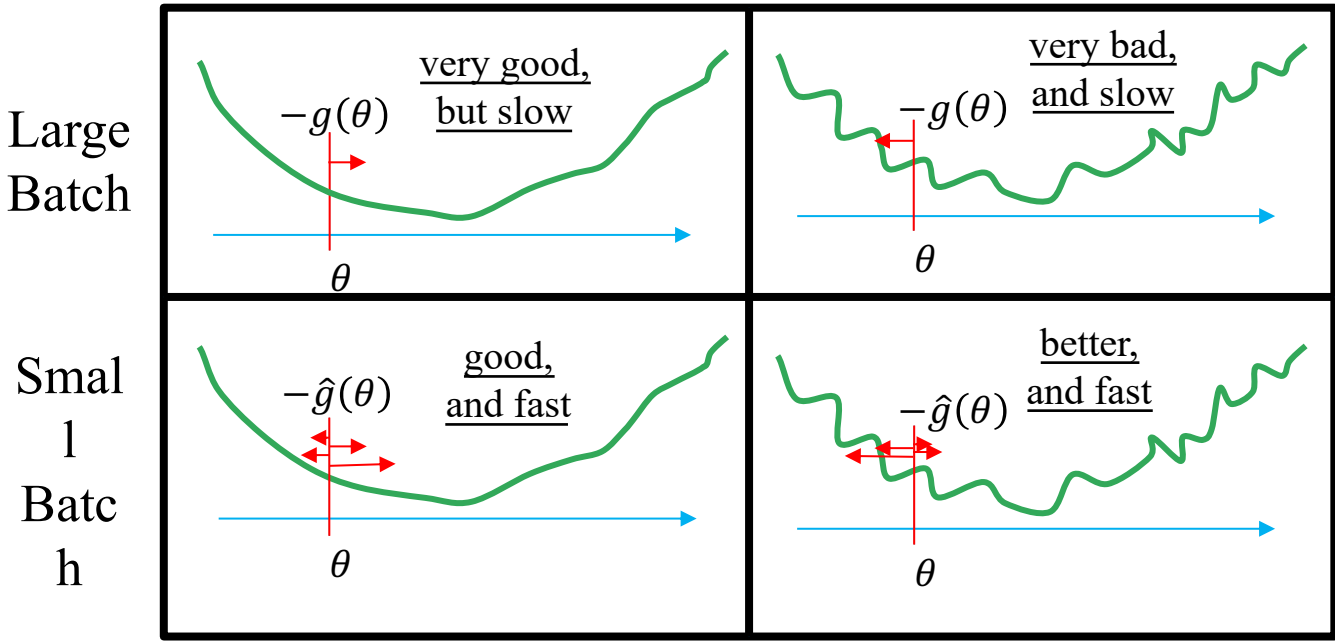


- Then we have that:
  - As  $K_b \rightarrow \infty$  (batch size goes up)  $\Rightarrow$ 
    - Noise decreases and computation increases
  - As  $K_b \rightarrow 0$  (batch size goes down)  $\Rightarrow$ 
    - Noise increases and computation decreases

# Effect of Gradient Noise: Exploration

$$\hat{g} = g + \frac{W}{\sqrt{K_b}}$$

Batch Gradient
True Gradient
Noise



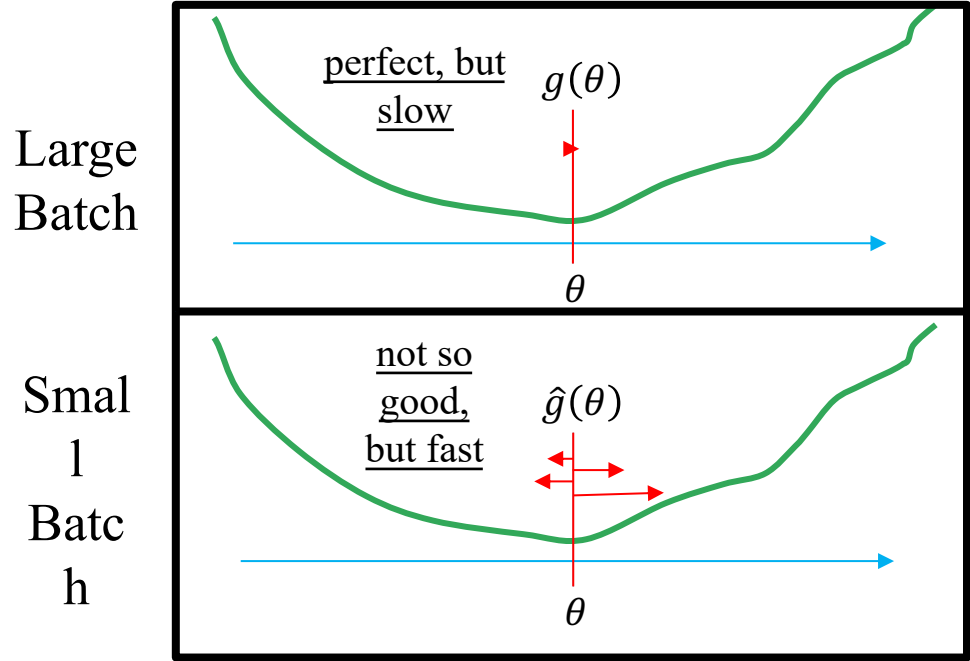
Smooth Function

Bumpy Function

# Effect of Gradient Noise: Exploitation

$$\hat{g} = g + \frac{W}{\sqrt{K_b}}$$

Batch Gradient      True Gradient      Noise



# SGD Issues and Tradeoffs

## ▪ Why SGD works?

- The gradient for a small batch is much faster to compute and almost as good as the full gradient.
- If  $K = 10,000$  and  $K_b = |S_b| = 32$ , then one iteration of SGD is approximately  $\frac{10,000}{32} \approx 312$  times faster than GD.

## ▪ Batch size

- Larger batches: less “noise” in gradient  $\Rightarrow$ 
  - *Worse*: slower updates; less exploration.
  - *Better*: better local convergence.
- Smaller batches: more “noise” in gradient  $\Rightarrow$ 
  - *Worse*: hunts around local minimum.
  - *Better*: faster updates; better exploration.

## ▪ Patch size:

- Many algorithms train on image “patches”
- Apocryphal: Smaller patches speed training. Not true!!!!
- However, smaller patches might fit better into GPU cache

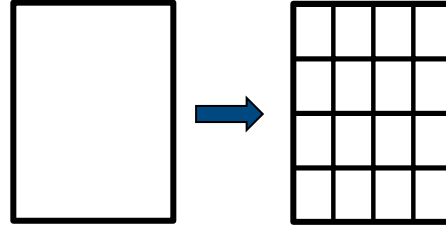
## ▪ Step size $\alpha$

- Too large  $\Rightarrow$  hunts around local minimum
- Too small  $\Rightarrow$  slow convergence

# Training with Patches

## ▪ Concept:

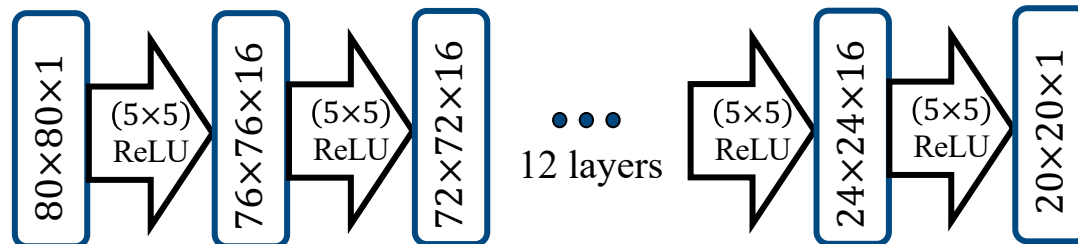
- Break training images into patches
- Often used in training denoising, deblurring, or reconstruction algorithms



- Typically, may be  $N \times N$  where  $N = 80$  patches (DNCNN)
- Typically, use a stride of  $N_s = N/2$  so that patches overlap

## ▪ Patch size issues:

- Apocryphal:
  - Smaller patches increase amount of training data. Not true!!
  - Smaller patches speed training. Not true!!!!
- Advantages:
  - Smaller patches might fit better into GPU cache
- Disadvantages:
  - Valid region tends towards 0 for deep CNNs



# Momentum

- SGD with momentum

- $\alpha$  is step size, and  $\gamma$  is momentum typically with  $\gamma = 0.9$

```
init  $v \leftarrow 0$ 
Repeat until converged {
  Repeat  $b = 1$  to  $B$  {
     $d \leftarrow -\nabla L(\theta; S_b)$ 
     $v \leftarrow \gamma v + \alpha(1 - \gamma)d$ 
     $\theta \leftarrow \theta + v^t$ 
  }
}
```

*momentum  
term*

*Stochastic Gradient Descent (SGD)  
with momentum*

- Interpretation

- $\theta$  is like position
- $v$  is like velocity
- Friction =  $1 - \gamma$

# Momentum

## ■ SGD with momentum

- $\alpha$  is step size, and  $\gamma$  is momentum typically with  $\gamma = 0.9$

init  $v \leftarrow 0$

Repeat until converged {

Repeat  $b = 1$  to  $B$  {

$d \leftarrow -\nabla L(\theta; S_b)$

$v \leftarrow \gamma v + \alpha(1 - \gamma)d$

$\theta \leftarrow \theta + v^t$

}

}

*Stochastic Gradient Descent (SGD)  
with momentum*

*for each epoch*

*for each batch*

*momentum*

*term*

### – Interpretation

- $\theta$  is like position
- $v$  is like velocity
- Friction =  $1 - \gamma$

# Interpretation of Momentum

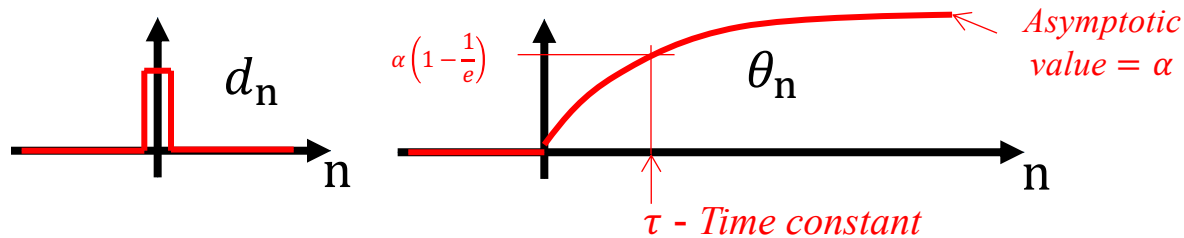
- Special case of impulsive input: If  $d_n = \delta_n$

```
init  $v \leftarrow 0$ ;  $\theta_{-1} \leftarrow 0$ 
Repeat  $n = 0$  to  $N - 1$  {
   $v \leftarrow \gamma v + \alpha(1 - \gamma)\delta_n$ 
   $\theta_n \leftarrow \theta_{n-1} + v^t$ 
}
```

*Momentum*

–Then

$$\theta_n = \begin{cases} \alpha(1 - \gamma^{n+1}) & n \geq 0 \\ 0 & n < 0 \end{cases}$$

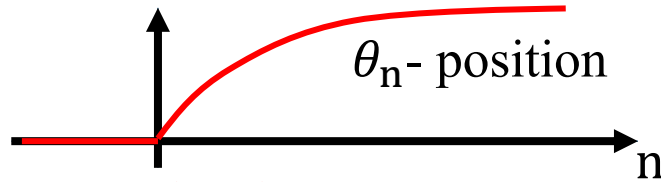
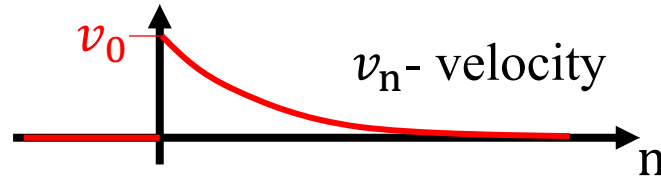


$$\tau = -1 - \frac{1}{\log \gamma} \quad \gamma = \exp\left\{-\frac{1}{\tau + 1}\right\}$$

# Intuition



*Friction =  $1 - \gamma$*



*Time constant  
=  $-1 - 1/\log \gamma$*

# Nesterov Momentum\*

“I skate to where the puck is going to be, not where it has been.” - Wayne Gretzky

- SGD with momentum
  - $\alpha$  is step size, and  $\gamma$  is momentum typically with  $\gamma \approx 0.9$

```
init  $v \leftarrow 0$ 
Repeat until converged {
  Repeat  $b = 1$  to  $B$  {
     $d \leftarrow -\nabla L(\theta + \gamma v^t; S_b)$ 
     $v \leftarrow \gamma v + \alpha d$ 
     $\theta \leftarrow \theta + v^t$ 
  }
}
```

*Nesterov  
gradient*

*Stochastic Gradient Descent (SGD)  
with Nesterov momentum*

- Intuition:
  - Even if  $d_n = 0$ , we have that  $\theta_{n+1} = \theta_n + \gamma v^t$  because of momentum
  - So compute the gradient at  $\theta_n + \gamma v^t$

\*Yu. E. Nesterov, “A method of solving a convex programming problem with convergence rate  $O(1/k^2)$ ”, Doklady ANSSSR (translated as Soviet.Math.Docl.), vol. 269, no. 3, pp. 543– 547.

# Preconditioned Gradient Descent

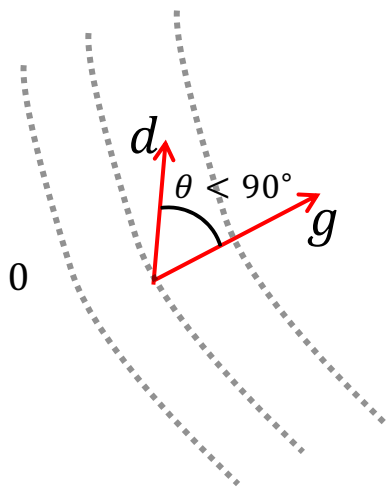
- Pick any positive definite matrix,  $M$ .
- Then Preconditioned Gradient Descent is

Repeat until converged {  
 $g \leftarrow -[\nabla L(\theta)]^t$   
 $d \leftarrow Mg$   
 $\theta \leftarrow \theta + \alpha d$   
}

- Notice that then

$$\langle d, g \rangle = \langle Mg, g \rangle = (Mg)^t g = g^t Mg > 0$$

- This means that the angle between  $d$  and  $g$  is  $< 90^\circ$
- So we see that PGD always goes down hill!



- How to pick  $M$ ?

- $M = B^t B$
- $M = \text{diag}(a_0, \dots, a_{N-1})$  where  $a_i > 0$

# ADAM (Adaptive Moment Estimation)\*

- SGD with ADAM optimization = Momentum + Preconditioning

```
init  $v \leftarrow 0$ ;  $r \leftarrow 0$ ;  
init  $t \leftarrow 0$   
Repeat until converged {  
  Repeat  $b = 1$  to  $B$  {  
     $t \leftarrow t + 1$   
     $d \leftarrow -\nabla L(\theta; S_b)$   
     $v \leftarrow \beta_1 v + (1 - \beta_1) d$   
     $r \leftarrow \beta_2 r + (1 - \beta_2) d^2$   
     $\hat{v} \leftarrow v / (1 - \beta_1^t)$   
     $\hat{r} \leftarrow r / (1 - \beta_2^t)$   
     $\theta \leftarrow \theta + \alpha (\sqrt{\hat{r}} + \epsilon)^{-1} \hat{v}$   
  }  
}
```

*ADAM Optimization*

- Typical parameters:  $\alpha = 0.001$ ;  $\beta_1 = 0.9$ ;  $\beta_2 = 0.999$ ;  $\epsilon = 10^{-8}$

\*Diederik P. Kingma and Jimmy Ba, “Adam: A Method for Stochastic Optimization”, The 3rd International Conference for Learning Representations (ICLR), San Diego, 2015.

# ADAM (Adaptive Moment Estimation)\*

- SGD with ADAM optimization = Momentum + Preconditioning

```
init  $v \leftarrow 0$ ;  $r \leftarrow 0$ ;  
init  $t \leftarrow 0$   
Repeat until converged {  
  Repeat  $b = 1$  to  $B$  {  
     $t \leftarrow t + 1$   
     $d \leftarrow -\nabla L(\theta; S_b)$   
     $v \leftarrow \beta_1 v + (1 - \beta_1) d$   
     $r \leftarrow \beta_2 r + (1 - \beta_2) d^2$   
     $\hat{v} \leftarrow v / (1 - \beta_1^t)$   
     $\hat{r} \leftarrow r / (1 - \beta_2^t)$   
     $\theta \leftarrow \theta + \alpha (\sqrt{\hat{r}} + \epsilon)^{-1} \hat{v}$   
  }  
}
```

*ADAM Optimization*

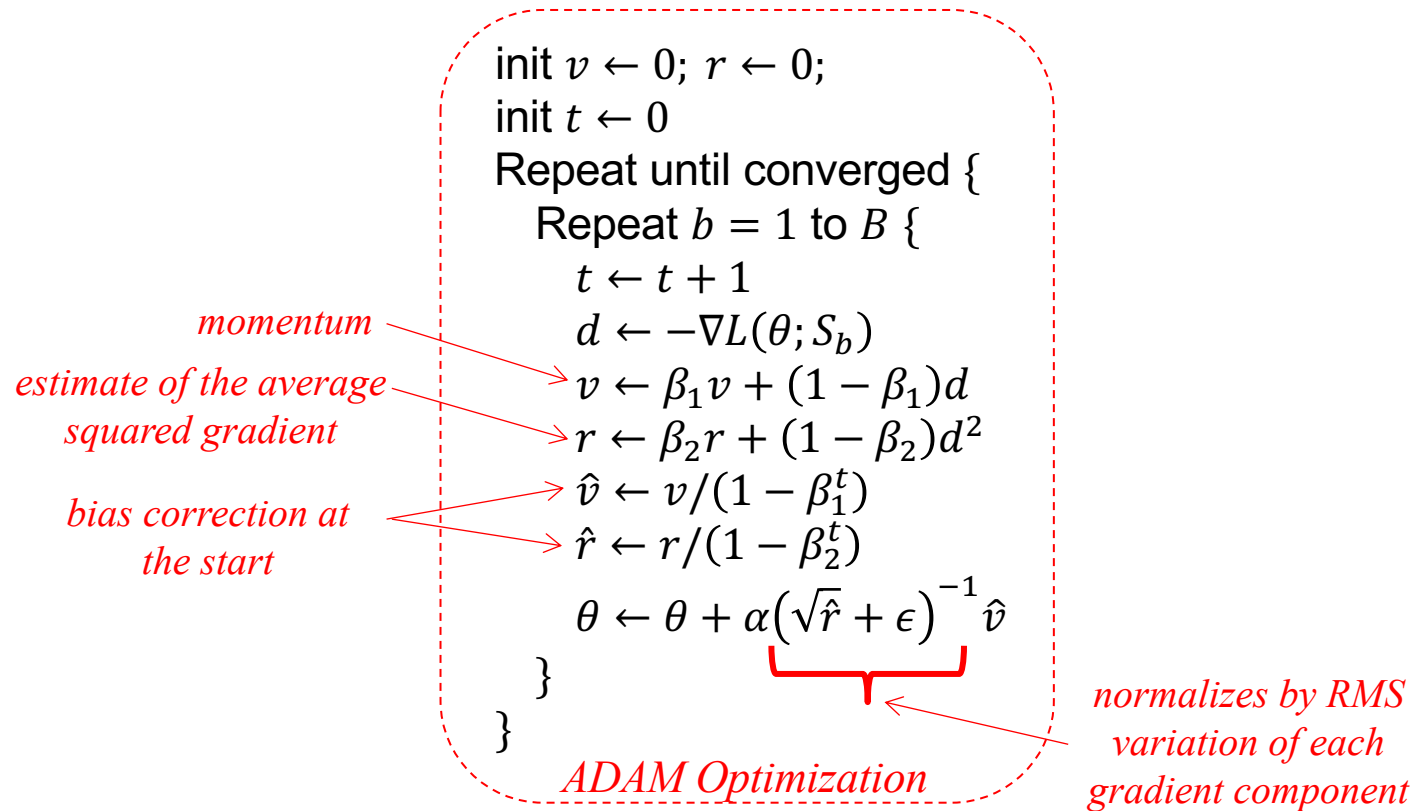
*These are all  
interpreted as element-  
wise operations on  
vectors*

- Typical parameters:  $\alpha = 0.001$ ;  $\beta_1 = 0.9$ ;  $\beta_2 = 0.999$ ;  $\epsilon = 10^{-8}$

\*[Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization", The 3rd International Conference for Learning Representations \(ICLR\), San Diego, 2015.](#)

# ADAM (Adaptive Moment Estimation)\*

- SGD with ADAM optimization = Momentum + Preconditioning



- Typical parameters:  $\alpha = 0.001$ ;  $\beta_1 = 0.9$ ;  $\beta_2 = 0.999$ ;  $\epsilon = 10^{-8}$

\*Diederik P. Kingma and Jimmy Ba, “Adam: A Method for Stochastic Optimization”, The 3rd International Conference for Learning Representations (ICLR), San Diego, 2015.