Deep Learning

08 Training DNNs - I

Dr. Konda Reddy Mopuri
Dept. of Artificial Intelligence
IIT Hyderabad
Jan-May 2024

Issues with SGD

- DNNs are trained via SGD: $w_{t+1} = w_t - \eta \cdot \nabla_w J(w)$

Issues with SGD

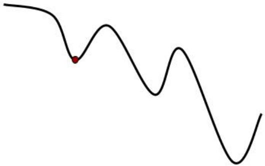
- DNNs are trained via SGD: $w_{t+1} = w_t - \eta \cdot \nabla_w J(w)$
- Loss is a high dimensional function

Issues with SGD

- DNNs are trained via SGD: $w_{t+1} = w_t - \eta \cdot \nabla_w J(w)$
- Loss is a high dimensional function
  - May have local minima

Issues with SGD

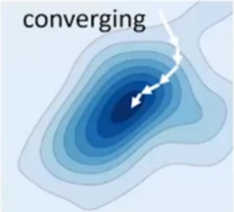
- DNNs are trained via SGD: $w_{t+1} = w_t - \eta \cdot \nabla_w J(w)$
- Loss is a high dimensional function
  - May have local minima
  - May have saddle points



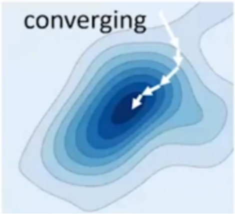
Stuck at a local minimum

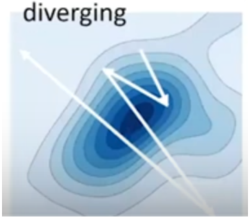
Stuck at a saddle point

Convergence of Gradient Descent



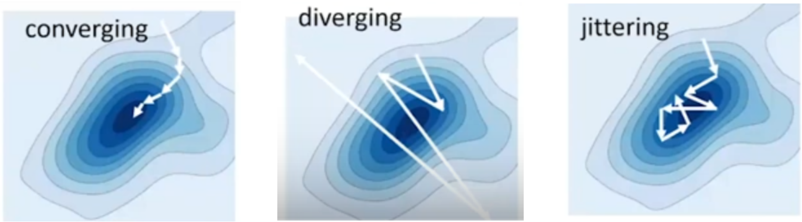
Convergence of Gradient Descent





Convergence of Gradient Descent

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad



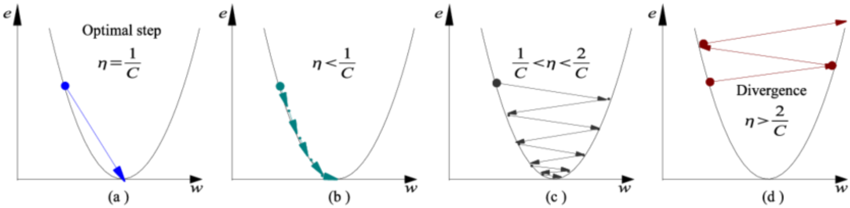
Convergence of Gradient Descent

- When does it diverge?

Convergence of Gradient Descent

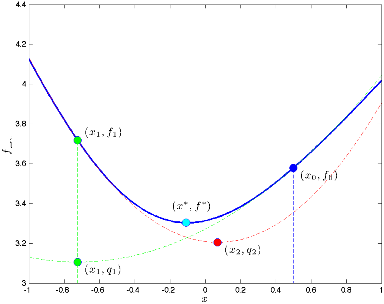
- When does it diverge?
- How to ensure smooth convergence? (Conditions for convergence)

Convergence for Quadratic functions



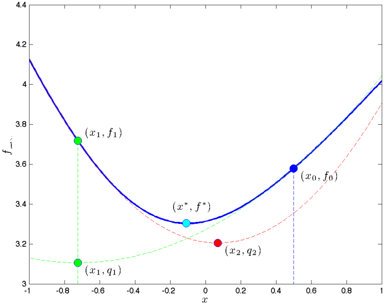
In case of generic and convex functions

- Perform a quadratic approximation



In case of generic and convex functions

- Perform a quadratic approximation
- $\eta_{opt} = \frac{1}{f''}$ (Newton's Method)



Multivariate functions

- $f(\mathbf{x}) = \mathbf{\frac{1}{2}} \mathbf{x}^{\mathbf{T}} \mathbf{A} \mathbf{x} + \mathbf{x}^{\mathbf{T}} \mathbf{b} + \mathbf{c}$

Multivariate functions

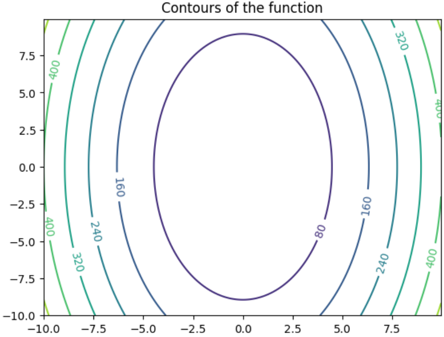
- $f(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{2}}\mathbf{x}^\mathbf{T}\mathbf{A}\mathbf{x} + \mathbf{x}^\mathbf{T}\mathbf{b} + \mathbf{c}$
- For convex functions, A is positive definite

Multivariate functions

- $f(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{2}}\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x} + \mathbf{x}^{\mathbf{T}}\mathbf{b} + \mathbf{c}$
- For convex functions, A is positive definite
- (For simplicity) If A is diagonal ($+$ve entries for convex f), then f is sum of multiple quadratic functions

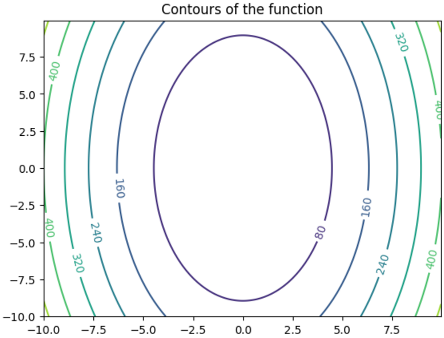
Multivariate functions

- Optimization gets decoupled (each component can be optimized independently)



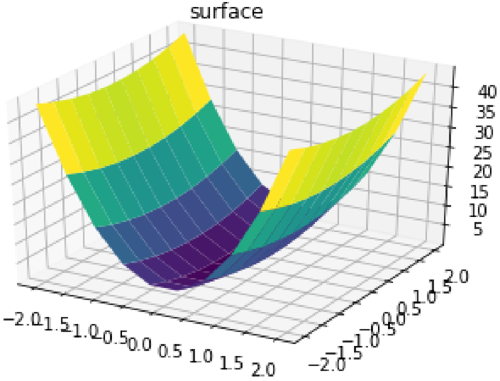
Multivariate functions

- Optimization gets decoupled (each component can be optimized independently)
- Optimal Learning rate is different for different components



Issues with SGD

- DNNs are trained via SGD: $w_{t+1} = w_t - \eta \cdot \nabla_w J(w)$
- Loss is a high dimensional function
  - May vary swiftly in one direction and slowly in the other



Issues with SGD

- Learning rate must be smaller than the twice the smallest optimal learning rate $\eta < 2 \cdot \eta_{min}$

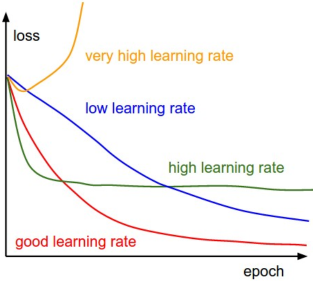
Issues with SGD

- Learning rate must be smaller than the twice the smallest optimal learning rate $\eta < 2 \cdot \eta_{min}$
- Else, it may diverge

Issues with SGD

- Learning rate must be smaller than the twice the smallest optimal learning rate $\eta < 2 \cdot \eta_{min}$
- Else, it may diverge
- This makes the convergence slow (and oscillate in some directions)

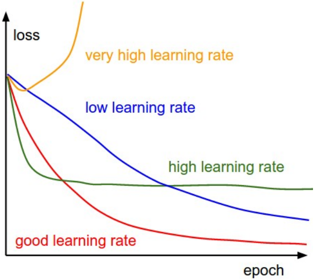
Learning rate (lr)



- What lr to use?

Figure credits: CS231n-Standford

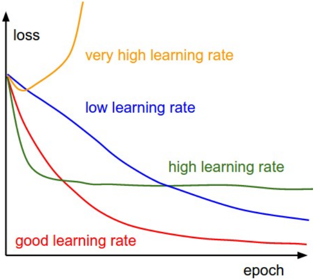
Learning rate (lr)



- What lr to use?
- Different lr at different stages of the training!

Figure credits: CS231n-Standford

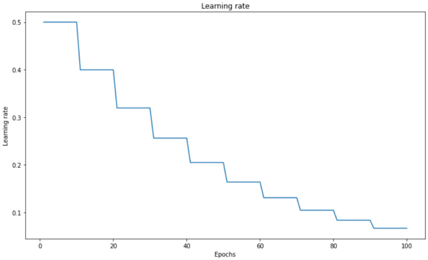
Learning rate (lr)



- What lr to use?
- Different lr at different stages of the training!
- Start with high lr and reduce it with time

Figure credits: CS231n-Standford

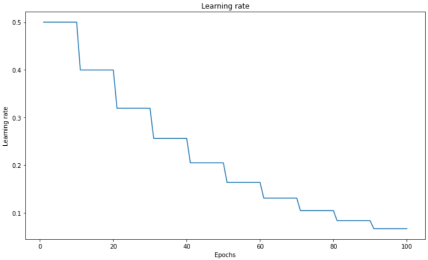
Learning Rate decay: Step



1. Reduce the lr after regular intervals

Figure credits: Katherine Li

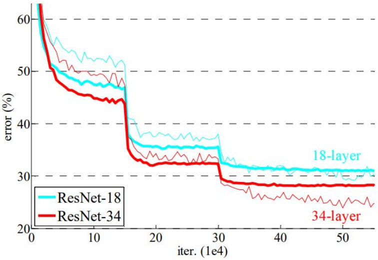
Learning Rate decay: Step



1. Reduce the lr after regular intervals
2. E.g. after every 30 epochs, $\eta* = 0.1 \cdot \eta$

Figure credits: Katherine Li

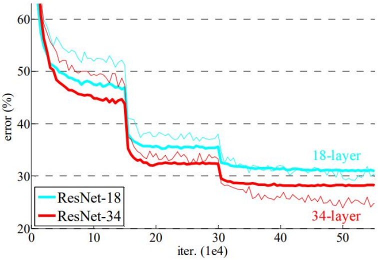
Learning Rate decay: Step



1. Characteristic loss curve: different phases for ''stage'

Figure credits: Kaiming He et al. 2015, ResNets

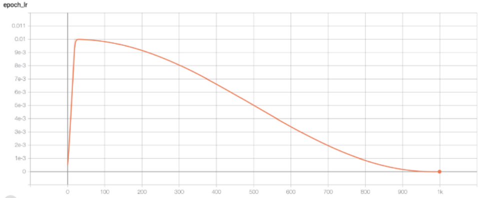
Learning Rate decay: Step



1. Characteristic loss curve: different phases for ''stage'
2. Issues: annoying hyper-params (when to reduce, by how much, etc.)

Figure credits: Kaiming He et al. 2015, ResNets

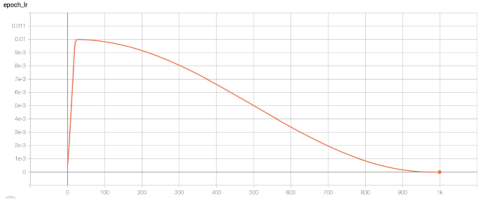
Learning Rate decay: Cosine



1. Reduces the lr continuously
   $\eta_t = \frac{1}{2}\eta_0(1 + cos(t\pi/T))$

Figure credits: Sebastian Correa and Medium.com

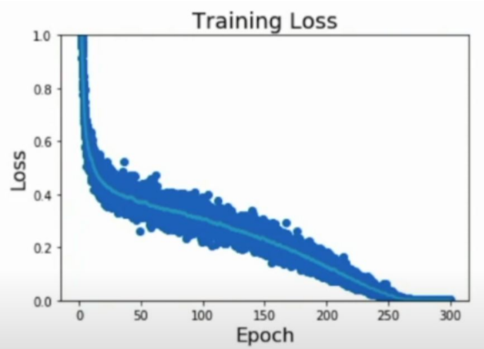
Learning Rate decay: Cosine



1. Reduces the lr continuously $\eta_t = \frac{1}{2}\eta_0(1 + cos(t\pi/T))$
2. Less number of hyper-parameters

Figure credits: Sebastian Correa and Medium.com

Learning Rate decay: Cosine

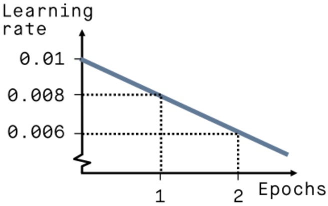


1. Training longer tends to work, but initial lr is still a tricky one

Figure credits: Dr Justin Johnson, U Michigan

Learning Rate decay: Linear

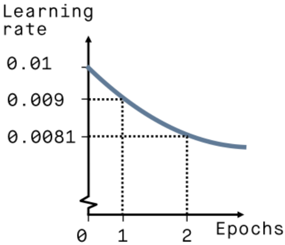
భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad



1. $\eta_t = \eta_0(1 - t/T)$

Figure credits: peltarion.com

Learning Rate decay: Exponential



1. $\eta_t = \eta_0 \cdot (1 - \alpha/100)^t$

Figure credits: peltarion.com

Learning Rate decay: Constant lr

1. No change in the learning rate
   $\eta_t = \eta_0$

Learning Rate decay: Constant lr

1. No change in the learning rate
   $\eta_t = \eta_0$
2. Works for prototyping of ideas (other schedules may be better for squeezing in those 1-2% of gains in the performance)

Issues with SGD

- SGD leads to jitter along the deep dimension and slow progress along the shallow one

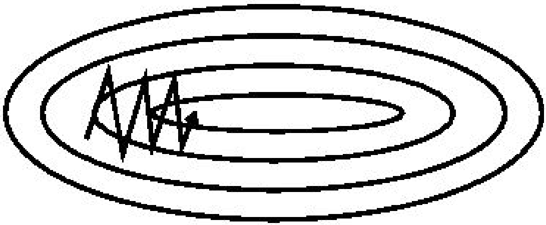


Figure credits: Sebastian Ruder

SGD+Momentum

SGD

$$w_{t+1} = w_t - \eta \cdot \nabla_w J(w)$$

SGD+Momentum

$$v_0 = 0$$
$$v_{t+1} = \rho \cdot v_t + \nabla_w J(w)$$
$$w_{t+1} = w_t - \eta \cdot v_{t+1}$$

I Sutskever et al., ICML 2013

SGD+Momentum

SGD

$$w_{t+1} = w_t - \eta \cdot \nabla_w J(w)$$

SGD+Momentum

$$v_0 = 0$$
$$v_{t+1} = \rho \cdot v_t + \nabla_w J(w)$$
$$w_{t+1} = w_t - \eta \cdot v_{t+1}$$

- Aggregates velocity: exponential moving average over gradients

I Sutskever et al., ICML 2013

SGD+Momentum

SGD

$$w_{t+1} = w_t - \eta \cdot \nabla_w J(w)$$

SGD+Momentum

$$v_0 = 0$$
$$v_{t+1} = \rho \cdot v_t + \nabla_w J(w)$$
$$w_{t+1} = w_t - \eta \cdot v_{t+1}$$

- Aggregates velocity: exponential moving average over gradients
- ρ is the friction (typically set to 0.9 or 0.99)

I Sutskever et al., ICML 2013

SGD+Momentum

SGD

$$w_{t+1} = w_t - \eta \cdot \nabla_w J(w)$$

for i in range(num_iters):
$\rightarrow$dw = grad(J, W, x, y)
$\rightarrow w- = \eta \cdot dw$

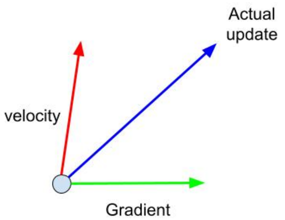
SGD+Momentum

$$v_0 = 0$$
$$v_{t+1} = \rho \cdot v_t + \nabla_w J(w)$$
$$w_{t+1} = w_t - \eta \cdot v_{t+1}$$

$v_0 = 0$
for i in range(num_iters):
$\rightarrow$dw = grad(J, W, x, y)
$\rightarrow v = \rho \cdot v + dw$
$\rightarrow w- = \eta \cdot v$

I Sutskever et al., ICML 2013

SGD+Momentum

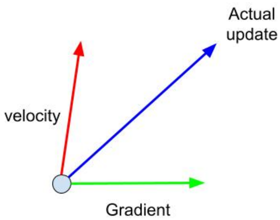


Momentum Update

1. How can momentum help?

I Sutskever et al., ICML 2013

SGD+Momentum

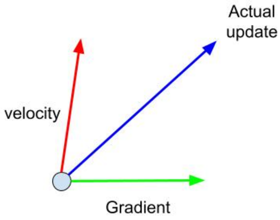


Momentum Update

1. How can momentum help?
   - Optimization proceeds even at the local minimum or saddle point (because of the accumulated velocity)

I Sutskever et al., ICML 2013

SGD+Momentum

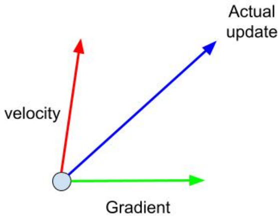


Momentum Update

1. How can momentum help?
   - Optimization proceeds even at the local minimum or saddle point (because of the accumulated velocity)
   - Jitter is reduced in ravine like loss surfaces

I Sutskever et al., ICML 2013

SGD+Momentum



Momentum Update

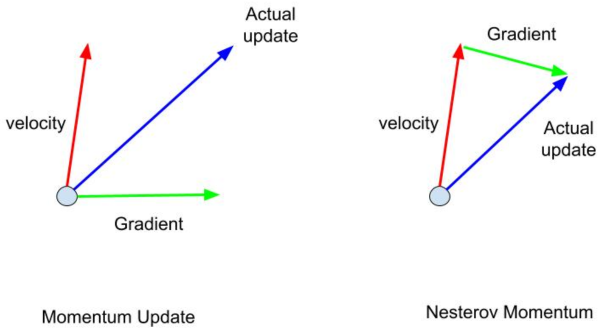
1. How can momentum help?
    - Optimization proceeds even at the local minimum or saddle point (because of the accumulated velocity)
    - Jitter is reduced in ravine like loss surfaces
    - Updates are more smoothed out (less noisy because of the exponential averaging)

I Sutskever et al., ICML 2013

Nesterov Momentum

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

1. Look ahead with the velocity, then take a step in the gradient's direction



I Sutskever et al., ICML 2013

Nesterov Momentum

$v_0 = 0$

`for i in range(num_iters):`

$\rightarrow$`dw = grad(`$J, W + \rho \cdot v, x, y$`)`

$\rightarrow v = \rho \cdot v + dw$

$\rightarrow w- = \eta \cdot v$

NAG allows to change velocity in a faster and more responsive way (particularly for large values of ρ)

I Sutskever et al., ICML 2013

Ada Grad

1. Goal: Adaptive (or, per-parameter) learning rates are introduced

Duchi et al. 2011, JMLR

Ada Grad

1. Goal: Adaptive (or, per-parameter) learning rates are introduced
2. Parameter-wise scaling of the learning rate by the aggregated gradient

Duchi et al. 2011, JMLR

Ada Grad

```
 grad_sq = 0
for i in range(max_iters):
→ dw = grad(J,w,x,y)
→grad_sq += dw ⊙ dw
→ w− = η · dw/(sqrt(grad_sq) + ε)
```

Duchi et al. 2011, JMLR

Ada Grad


```
 grad_sq = 0
for i in range(max_iters):
→ dw = grad(J,w,x,y)
→grad_sq += dw ⊙ dw
→ w- = η · dw/(sqrt(grad_sq) + ϵ)
```

- Smaller (larger) updates for parameters associated with frequently (infrequently) occurring features

Duchi et al. 2011, JMLR

Ada Grad

```
 grad_sq = 0
for i in range(max_iters):
→ dw = grad(J,w,x,y)
→grad_sq += dw ⊙ dw
```
$\rightarrow w -= \eta \cdot dw / (\texttt{sqrt(grad\_sq)} + \epsilon)$

- Smaller (larger) updates for parameters associated with frequently (infrequently) occurring features
- well-suited for dealing with sparse data

Duchi et al. 2011, JMLR

RMS Prop

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

1. If Ada Grad is run for too long
   - the gradients accumulate to a big value
   - $\rightarrow$ update becomes too small (or, learning rate is reduced continuously)

RMS Prop

1. If Ada Grad is run for too long
   - the gradients accumulate to a big value
   - $\rightarrow$ update becomes too small (or, learning rate is reduced continuously)
2. RMS prop (a leaky version of Ada Grad) addresses this using a friction coefficient (ρ)

RMS Prop

```
grad_sq = 0
for i in range(max_iters):
→ dw = grad(J,w,x,y)
→grad_sq = ρ· grad_sq + (1 − ρ)· dw ⊙ dw
→ w− = η · dw/(sqrt(grad_sq) + ϵ)
```

Adam

1. Inculcates both the good things: momentum and the adaptive learning rates
   Adam = RMSProp + Momentum

Adam

1. Inculcates both the good things: momentum and the adaptive learning rates
   Adam = RMSProp + Momentum
2. $m1 = 0$
   $m2 = 0$
   `for i in range(max_iters):`
   $\rightarrow$ `dw = grad(J,w,x,y)`
   $\rightarrow\ m1 = \beta_1 \cdot m1 + (1 - \beta_1) \cdot dw$
   $\rightarrow\ m2 = \beta_2 \cdot m2 + (1 - \beta_2) \cdot dw^2$
   $\rightarrow\ w- = \eta \cdot m1/(\texttt{sqrt}(m2) + \epsilon)$

Adam

1. $m1 = 0$

   $m2 = 0$

   `for i in range(max_iters):`

   $\rightarrow$ `dw = grad(J,w,x,y)`

   $\rightarrow m1 = \beta_1 \cdot m1 + (1 - \beta_1) \cdot dw$

   $\rightarrow m2 = \beta_2 \cdot m2 + (1 - \beta_2) \cdot dw^2$

   $\rightarrow w- = \eta \cdot m1/(\texttt{sqrt}(m2) + \epsilon)$

Adam

1. $m1 = 0$
   $m2 = 0$
   `for i in range(max_iters):`
   $\rightarrow$ `dw = grad(J,w,x,y)`
   $\rightarrow$ $m1 = \beta_1 \cdot m1 + (1 - \beta_1) \cdot dw$
   $\rightarrow$ $m2 = \beta_2 \cdot m2 + (1 - \beta_2) \cdot dw^2$
   $\rightarrow$ $w- = \eta \cdot m1/(\texttt{sqrt}(m2) + \epsilon)$
2. Bias correction is performed (since the estimates start from 0)

Adam

1. $m1 = 0$
   $m2 = 0$
   `for i in range(max_iters):`
   $\rightarrow$ `dw = grad(J,w,x,y)`
   $\rightarrow m1 = \beta_1 \cdot m1 + (1 - \beta_1) \cdot dw$
   $\rightarrow m2 = \beta_2 \cdot m2 + (1 - \beta_2) \cdot dw^2$
   $\rightarrow w- = \eta \cdot m1 / (\texttt{sqrt}(m2) + \epsilon)$
2. Bias correction is performed (since the estimates start from 0)
3. Adam works well in practice (mostly with a fixed set of values for the hyper-params)

Many more variants exist

1. AMSGrad
2. Nadam
3. AdaMax
4. AdaDelta