
Lecture3 Optimization

K u g g l e D e e p L e a r n i n g S t u d y w e e k 1

김효은

INDEX



Random Search



Follow the slope



Gradient Descent



Stochastic Gradient Descent



1. Random Search

우리가 이제까지 배운 것:

1. 예측하는 모델을 만드는 것 ($Wx+b$ 이용하는 등 ...)
2. 이 모델이 좋은 지 나쁜 지 판단하는 Loss function (지우가 바로 직전에 설명해준 😊)
ex) SVM의 hinge loss, multinomial Logistic Regression의 softmax

-> 이제부터는 데이터로부터 계산된 Loss를 바탕으로 더 좋은 모델이 되도록 뭔가를 할 거예요!

1. Random Search



산 전체 = loss function

산의 높이 = loss

내 모델이 낮은 Loss를 갖기를 원해!

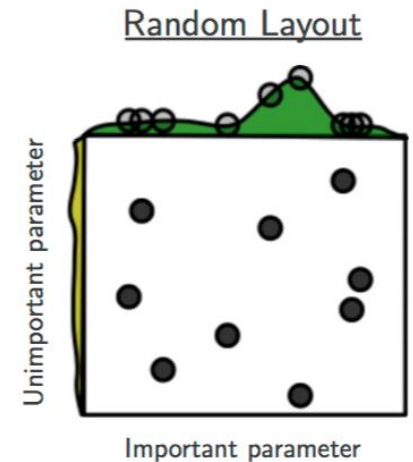
= 산의 높이가 낮아지는 곳으로 가길 원해!

Q. 어떻게 하면 산의 낮은 곳으로
이동할 수 있을까?

1. Random Search

방법1. 그냥 랜덤하게 방향 정하기 (random search)

```
bestloss = float("inf") # Python assigns the highest possible float
for num in xrange(1000):
    W = np.random.randn(10, 3073) * 0.0001 # generate random parameters
    loss = L(X_train, Y_train, W) # get the loss over the entire training set
    if loss < bestloss: # keep track of the best solution
        bestloss = loss
        bestW = W
print 'in attempt %d the loss was %f, best %f' % (num, loss, bestloss)
```



W를 random으로 정해 -> loss를 구해 -> loss가 낮아졌으면 그 값으로 W를 바꿔
-> loss가 낮아지지 않았으면 pass

2. Follow the slope

~~방법1. 그냥 랜덤하게 방향 정하기 (이것도 해보고, 저것도 해보고 ...)~~

방법2. 기울기를 따라 이동하기 (gradient descent)

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

기울기를 따라 이동하려면, 먼저 기울기(gradient)를 계산해야 한다.

(gradient를 어떻게 weight update에 사용하는지는 뒤에 나와요)

* 여기서 gradient란 편도함수의 집합이라고 생각하면 쉬워요.

2. Follow the slope

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

gradient dW:

[?,
?,
?,
?,
?,
?,
?,
?,
?,...]

먼저 gradient 구하는 것 부터 시작합니다~~~!

(정확히는 loss에 대한 W의 gradient)

2. Follow the slope

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

W + h (first dim):

[0.34 + 0.0001,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25322

gradient dW:

[?,
?,
?,
?,
?,
?,
?,
?,
?,...]

편미분 값을 위해 w1의 값만 조금 증가

2. Follow the slope

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

W + h (first dim):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25322

gradient dW:

[-2.5,
?,
?,

$$(1.25322 - 1.25347)/0.0001 = -2.5$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

?,
?,...]

2. Follow the slope

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

W + h (second dim):

[0.34,
-1.11 + 0.0001,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25353

gradient dW:

[-2.5,
0.6,
?,
?,
?,...]

$$(1.25353 - 1.25347) / 0.0001 = 0.6$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

?,...]

11

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

0.55,...]
loss 1.25347

[0.34,
-1.11,
0.78 + 0.0001,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

[illegible]

[-2.5,
0.6,
0,
?,
0

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

...

2. Follow the slope

근데 이거 언제 다해? ...

우리가 gradient를 구한 방식은 수치 미분(numerical differentiation)

(변화량을 아주 작은 값으로 두고 계산하는 것: 우리는 0.0001로 두고 계산했다)

수치 미분(numerical differentiation) approximate하고, 느려

-> 해석 미분(analytical differentiation) 사용해서 gradient 구하는 걸로 바꾸자!

(이것을 사용하면 나중에 나오는 backpropagation을 사용할 수 있게 된다!)

2. Follow the slope

수치미분의 단점으로 gradient를 analytical differentiation으로 구한다고 하자.

그렇다면 이제 gradient를 이동하는데 어떻게 활용할 것인가?

잠깐, remind!

- *우리가 하고자 하는 것은 Loss를 작게 만드는 parameter(W)를 찾는 것
- * Loss를 작게 만들기 위해 W를 어떻게 이동시켜야 하는지 그 정보를 찾는 것!


3. Gradient descent

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

$dW = \dots$
(some function
data and W)



gradient dW:

[-2.5,
0.6,
0,
0.2,
0.7,
-0.5,
1.1,
1.3,
-2.1,...]

Gradient가 의미하는 바?


3. Gradient descent

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

$dW = \dots$
(some function
data and W)



gradient dW:

[-2.5,
0.6,
0,
0.2,
0.7,
-0.5,
1.1,
1.3,
-2.1,...]

Gradient가 의미하는 바?

W1을 1만큼 증가하면
loss는 -2.5 증가
= 2.5 감소


3. Gradient descent

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

$dW = \dots$
(some function
data and W)



gradient dW:

[-2.5,
0.6,
0,
0.2,
0.7,
-0.5,
1.1,
1.3,
-2.1,...]

Gradient가 의미하는 바?

W1을 1만큼 증가하면
Loss는 -2.5 증가
= 2.5 감소

W2를 1만큼 증가하면
Loss는 0.6 증가

3. Gradient descent

current W:

Gradient가 음수(-)일 때 w가 증가(+)하면 Loss 감소

Gradient가 양수(+)일 때 w가 감소(-)하면 Loss 감소

$dW = \dots$
(some function
data and W)

-> gradient의 반대 방향으로 이동하면 Loss 감소!

-> gradient로부터 이동할 방향에 대한 정보를 얻자!

0.33,...]

loss 1.25347

gradient dW:

[-2.5,
0.6,
0,
0.2,
0.7,
-0.5,
1.1,
1.3,
-2.1,...]

Gradient가 의미하는 바?

W1을 1만큼 증가하면

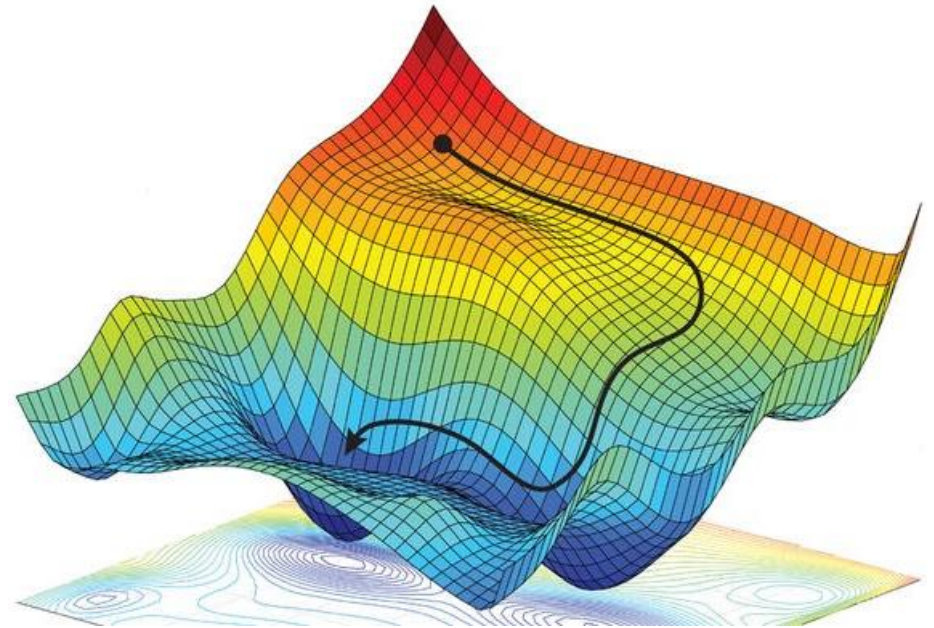
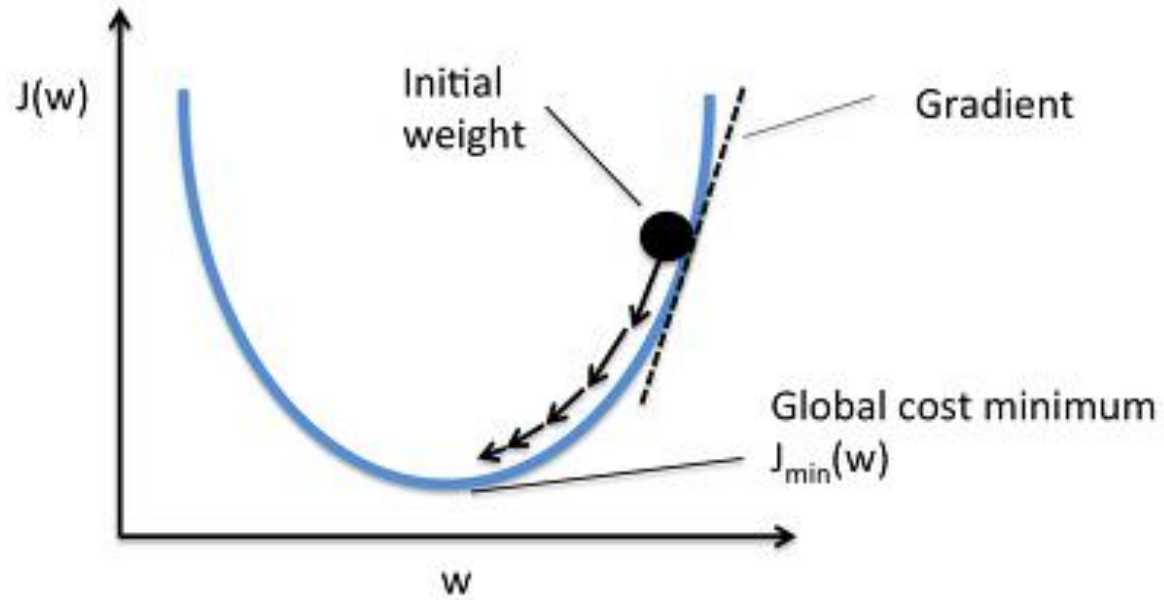
Loss는 -2.5 증가

= 2.5 감소

W2를 1만큼 증가하면

Loss는 0.6 증가

3. Gradient descent



3. Gradient descent

```
# Vanilla Gradient Descent
```

```
while True:
```

```
    weights_grad = evaluate_gradient(loss_fun, data, weights)
```

```
    weights += - step_size * weights_grad # perform parameter update
```

$\text{weights} = \text{weights} - \text{step_size} * \text{weight_grad}$



업데이트 된 weight

$= \text{weights} + \text{step_size} * (-\text{weight_grad})$



업데이트 이전 weights



Gradient의 반대방향으로 이동

3. Gradient descent

```
# Vanilla Gradient Descent
```

```
while True:
```

```
    weights_grad = evaluate_gradient(loss_fun, data, weights)
```

```
    weights += - step_size * weights_grad # perform parameter update
```

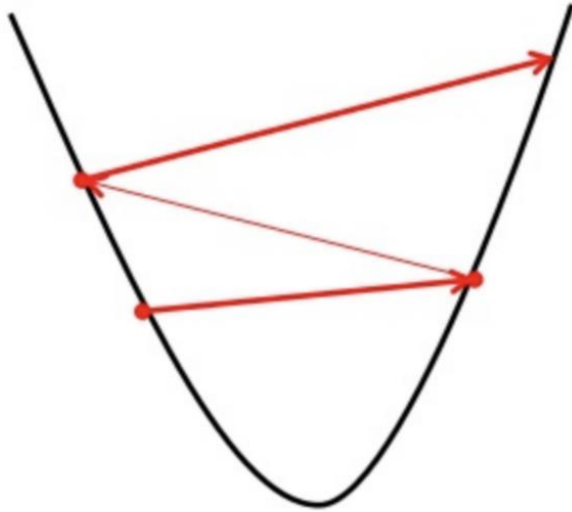
$\text{weights} = \text{weights} - \text{step_size} * \text{weight_grad}$

$= \text{weights} + \text{step_size} * (-\text{weight_grad})$

업데이트 된 weight 업데이트 이전 weights Gradient의 반대방향으로 이동

3. Gradient descent

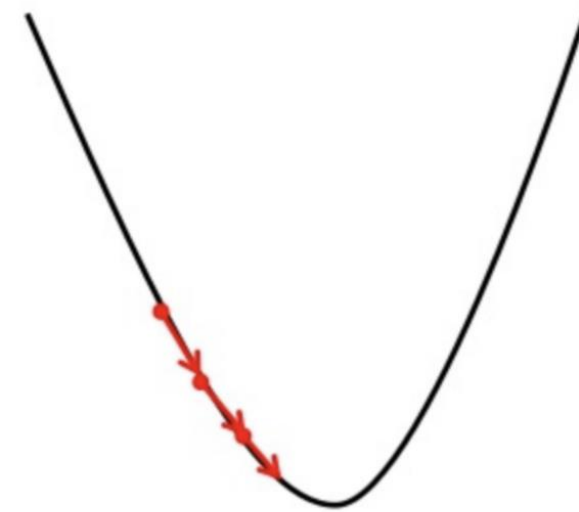
Big learning rate



Overshooting ☹

오히려 Loss가 증가하는 방향으로 update가 될 수도

Small learning rate



Slow ☹

너무 느리게 진행된다

3. Gradient descent

Big learning rate

Small learning rate

1. 언제나 적절한 learning rate는 존재하지 않는다!

2. 적절한 learning rate를 설정해주는 것은 중요하다.

3. Gradient descent 외 다른 optimizer도 다수 존재한다. Ex) Adam, Momentum, ...

Overshooting ☹

오히려 Loss가 증가하는 방향으로 update가 될 수도

Slow ☹

너무 느리게 진행된다

4. Stochastic Gradient Descent

Gradient를 구하려면 먼저 Loss를 구해야 한다!

Loss를 구하려면 각 데이터에서 prediction을 구하고, 실제 정답과 비교해야 한다.

근데, 데이터가 엄청 많으면?

Loss 구하는데 시간이 너무 오래 걸린다.

-> 학습하는 데 시간 너무 오래 걸린다 ☹

4. Stochastic Gradient Descent

Gradient를 구하려면 먼저 Loss를 구해야 한다!

Loss를 구하려면 각 데이터에서 prediction을 구하고, 실제 정답과 비교해야 한다.

전체 데이터가 아니라 일부 데이터(mini batch)로만 Loss를 구하자!

근데, 데이터가 엄청-> stochastic gradient descent

Loss 구하는데 시간이 너무 오래 걸린다.

-> 학습하는 데 시간 너무 오래 걸린다 ☹

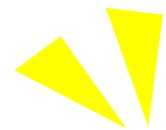
4. Stochastic Gradient Descent

Q. Stochastic은 무슨 의미로 붙었나요?

A. 여기서 stochastic은 '확률적인' 이라는 뜻으로,

전체 데이터의 loss를 sample 데이터에서 구한 loss로 추정하고, Sample 데이터로 gradient를 추정하기 때문

4. Stochastic Gradient Descent



$$L(W) = \frac{1}{N} \sum_{i=1}^N L_i(x_i, y_i, W) + \lambda R(W)$$

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^N \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

Full sum expensive
when N is large!

Approximate sum
using a **minibatch** of
examples
32 / 64 / 128 common

```
# Vanilla Minibatch Gradient Descent
```

```
while True: 전체 데이터에서 일부 데이터만 뽑는다
```

```
    data_batch = sample_training_data(data, 256) # sample 256 examples
```

```
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
```

```
    weights += - step_size * weights_grad # perform parameter update
```

4. Stochastic Gradient Descent

Algorithm 8.1 Stochastic gradient descent (SGD) update

Require: Learning rate schedule $\epsilon_1, \epsilon_2, \dots$

Require: Initial parameter θ

$k \leftarrow 1$

while stopping criterion not met **do**

 Sample a minibatch of m examples from the training set $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$.

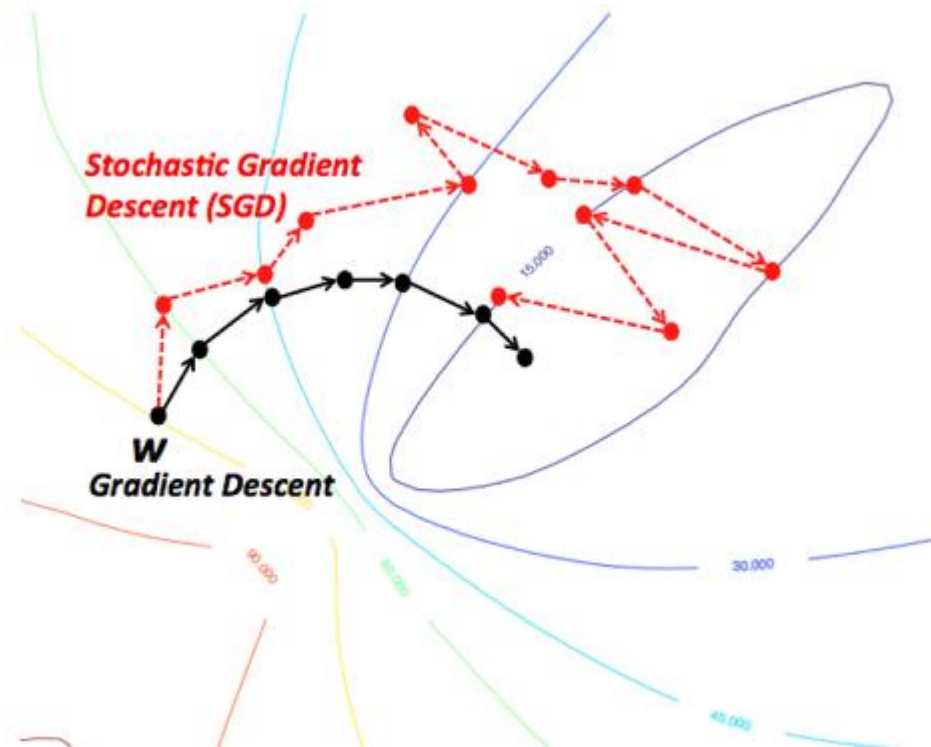
 Compute gradient estimate: $\hat{\mathbf{g}} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

 Apply update: $\theta \leftarrow \theta - \epsilon_k \hat{\mathbf{g}}$

$k \leftarrow k + 1$

end while

4. Stochastic Gradient Descent



Gradient Descent

- 모든 데이터를 계산한다.
- 확실한데 너무 느리다

Stochastic Gradient Descent

- 일부 데이터만 계산한다.
- 조금 헤메지만 빠르다

4. Stochastic Gradient Descent

1. Hyperparameter vs parameter

- Hyperparameter : 모델 학습 전에 설정되어야 하는 변수 (우리가 설정!) ex. learning_rate
- Parameter : 모델이 학습되며 업데이트 되는 변수 ex. 회귀분석에서는 회귀계수, $Wx+b$ 에서 W 와 b

2. mini-batch

<https://light-tree.tistory.com/133>

————— 감 사 합 니 다 ! —————

발표 들어주셔서 감사합니다

—————