Lecture 8.
Training Neural Networks Ⅲ

Review

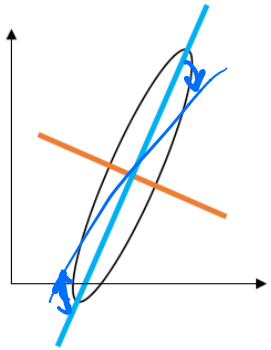
Data Normalization.

왜?

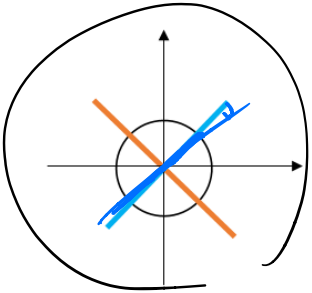
어떻게?

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$ Data → [0, 1]

Before normalization



After normalization



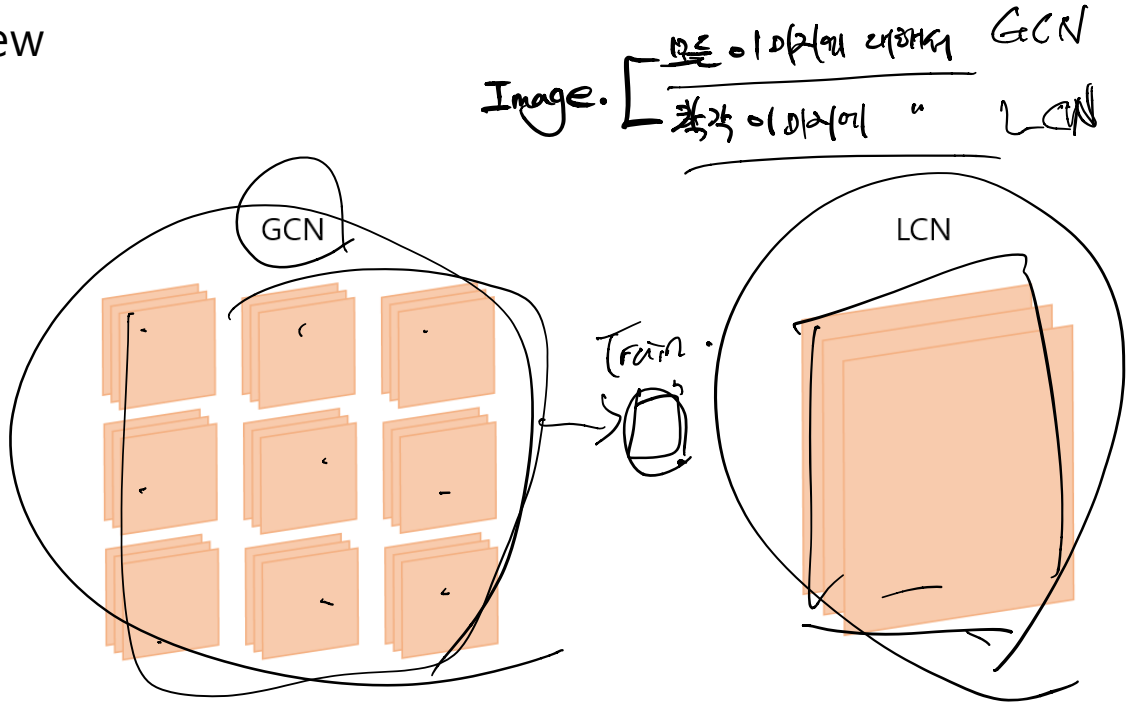
$$x_{new} = \frac{x - \mu}{\sigma}$$

or $x \sim N(0, 1)$.

zero-centered

Sigmoid. / ReLU

Review



Review

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

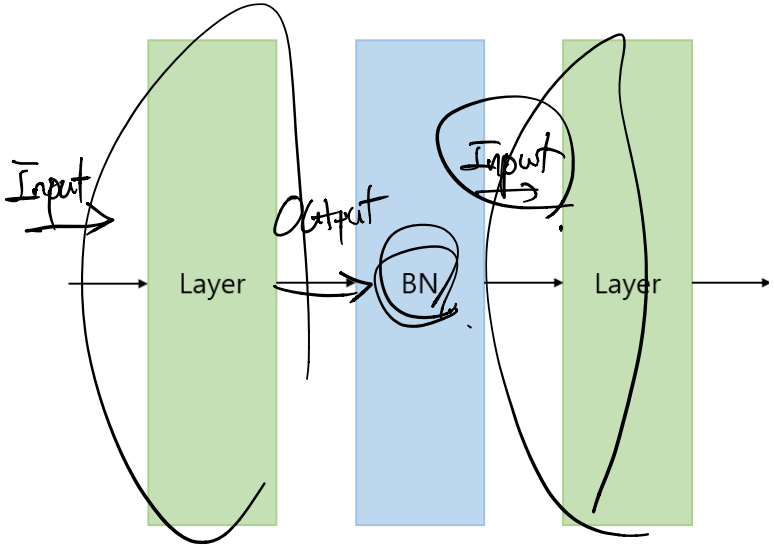
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

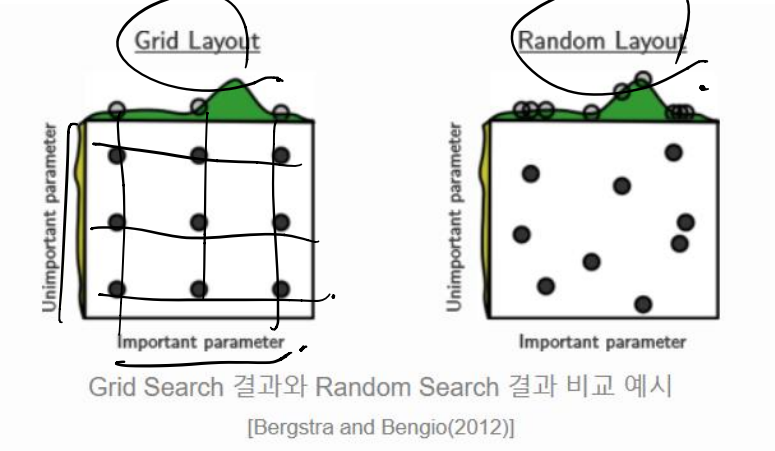
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

γ, β.



Review



Grid Search 결과와 Random Search 결과 비교 예시

[Bergstra and Bengio(2012)]

Review

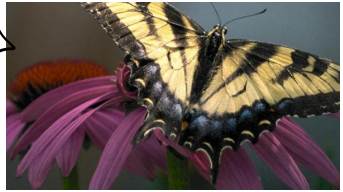
Image augmentation

Crop

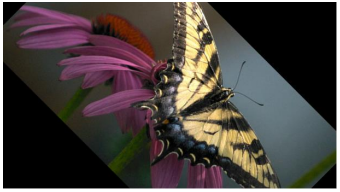
Flip

Noise

Scale



Rotate



Translation

Today's Contents

1. Change Optimization Process
   - Gradient based method - Optimizer
   - Regularization - Dropout (+Ensemble)

Gradient Descent Optimization Method

1. Gradient Descent
2. Stochastic Gradient Descent (SGD)
3. Momentum
4. Nesterov Accelerated Gradient (NAG)
5. Adagrad
6. RMSProp
7. AdaDelta
8. Adam

Gradient Descent (GD)

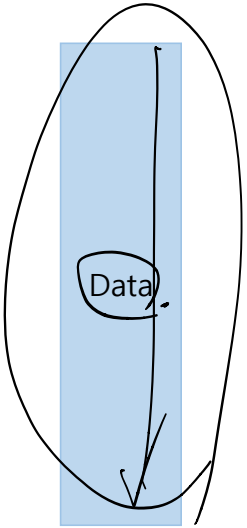
$$w_{new} = w - \eta \times \frac{\partial L}{\partial w}$$

w_{old}

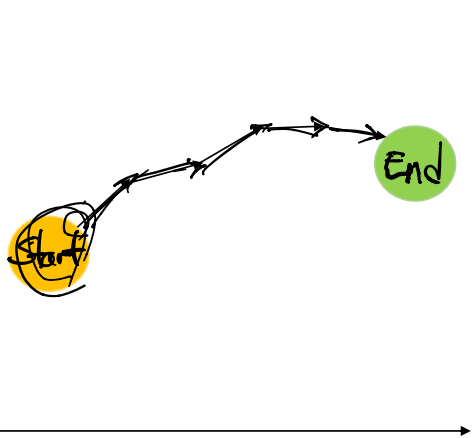
→ gradient

g

learning rate



거의 정확하다!
시간이 너무 오래

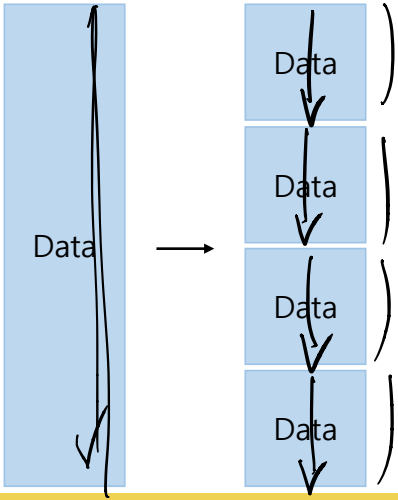


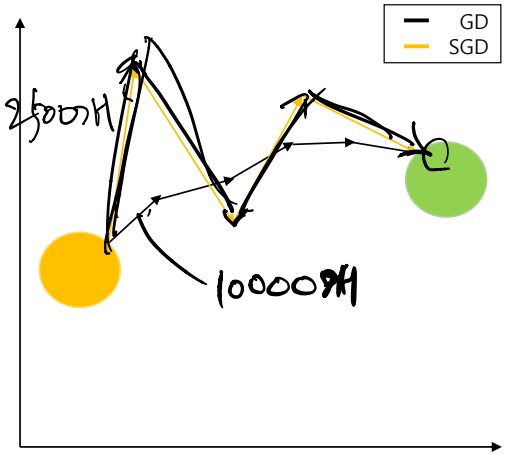
batch_size = 32

Stochastic Gradient Descent (SGD)

$$w_{new} = w - \eta \times \frac{\partial L}{\partial w}$$

mini_batch

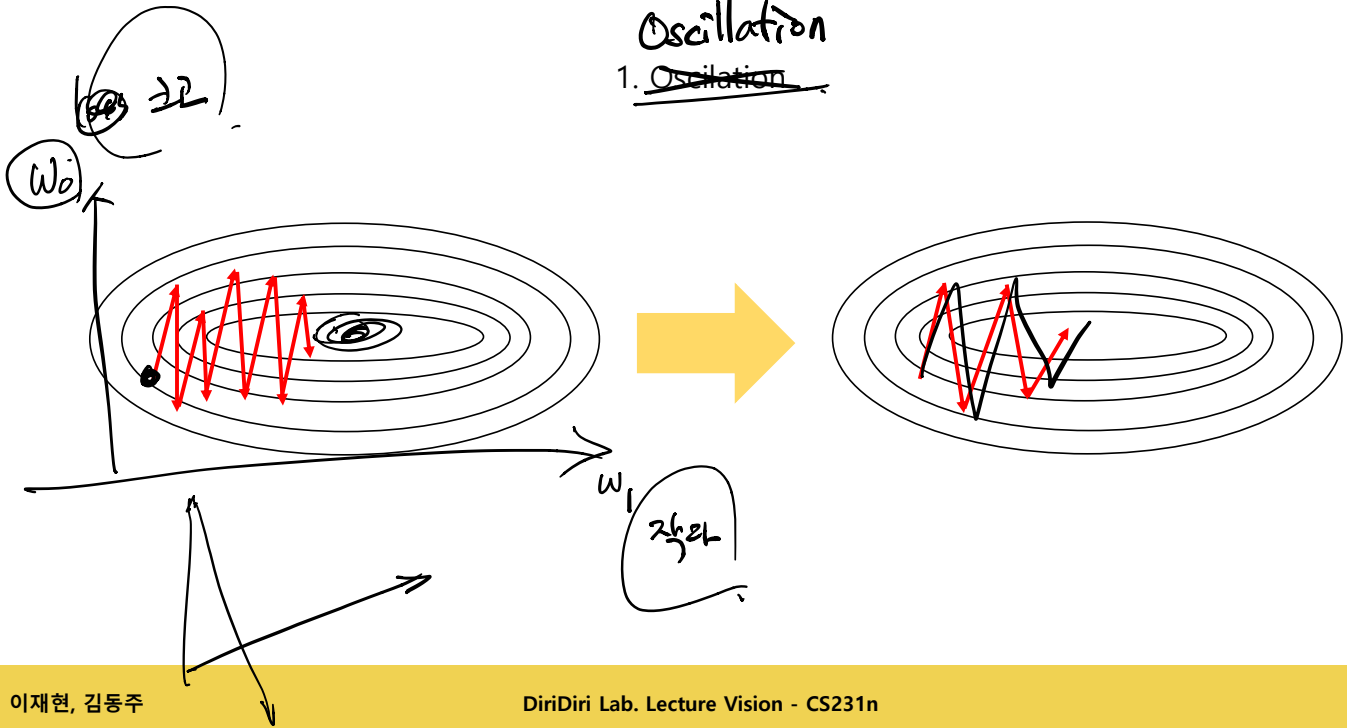




Two Problems of SGD

Oscillation

1. ~~Oscilation~~



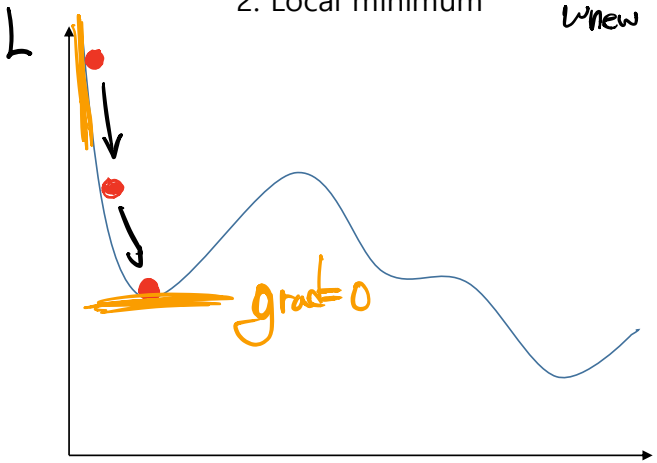
Two Problems of SGD

$w_{new} = w - \eta \times g$

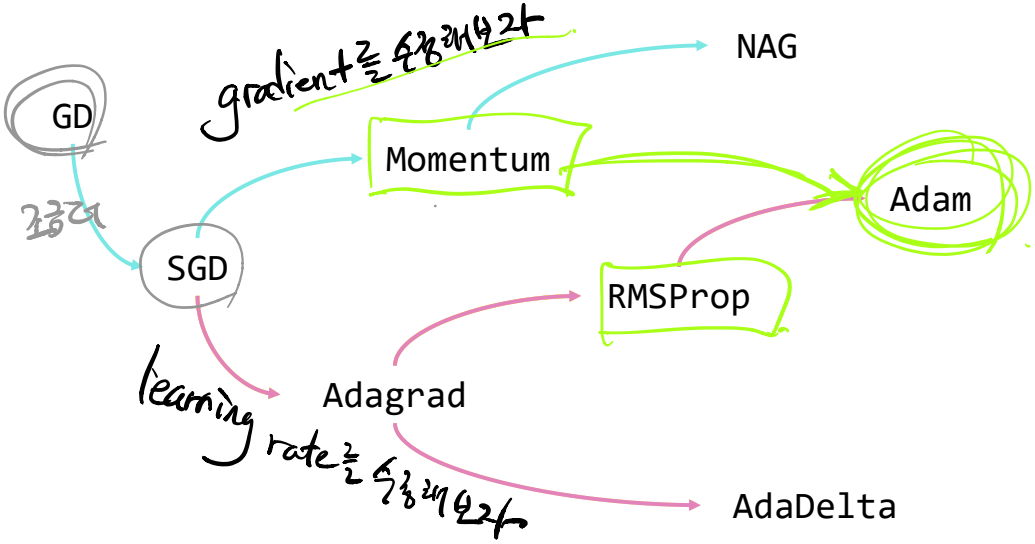
$w_{new} = w_{old}$

$\eta = 0$

2. Local minimum



Optimizer 발전 과정



Mometum

Idea : gradient update에 관성을 더하자!

v_t = 속도 벡터

$$w_{new} = w - v_t$$

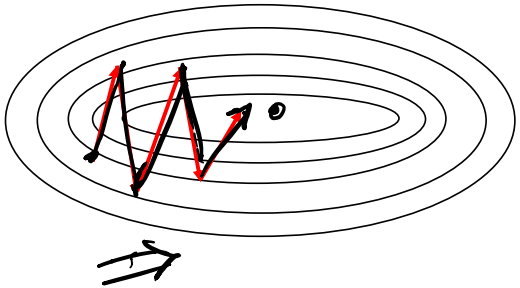
$$v_t = \gamma v_{t-1} + \eta \frac{\partial L}{\partial w}$$

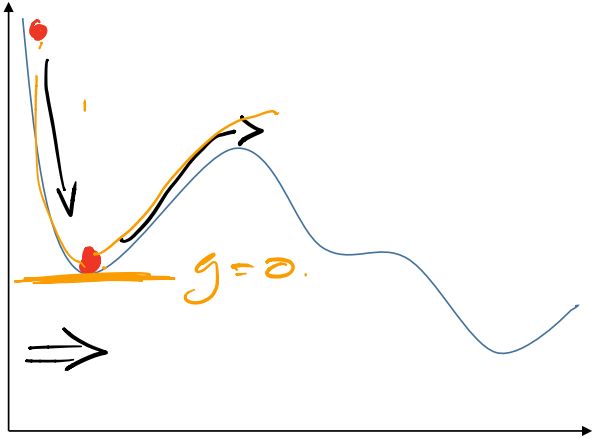
γv_{t-1} : friction 이전의 속도 마찰을

γ : 보통은 0.9

Momentum

$$v_t = \gamma \cdot v_{t-1} + \eta \times g$$

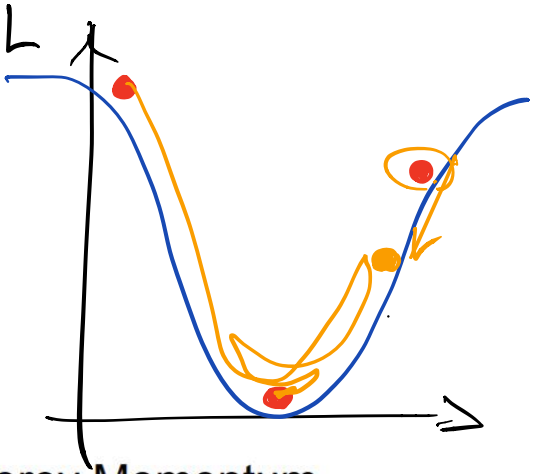




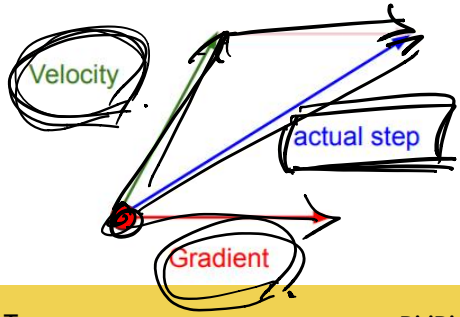
Nestrov Accelerated Gradient (NAG)

$$w_{new} = w - v_t$$

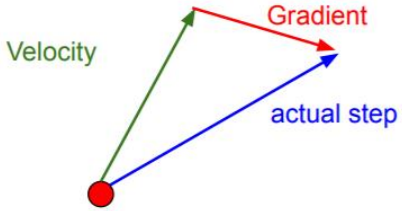
$$v_t = \gamma v_{t-1} + \eta \frac{\partial L}{\partial w}\Big|_{w=w-\gamma v_{t-1}}$$



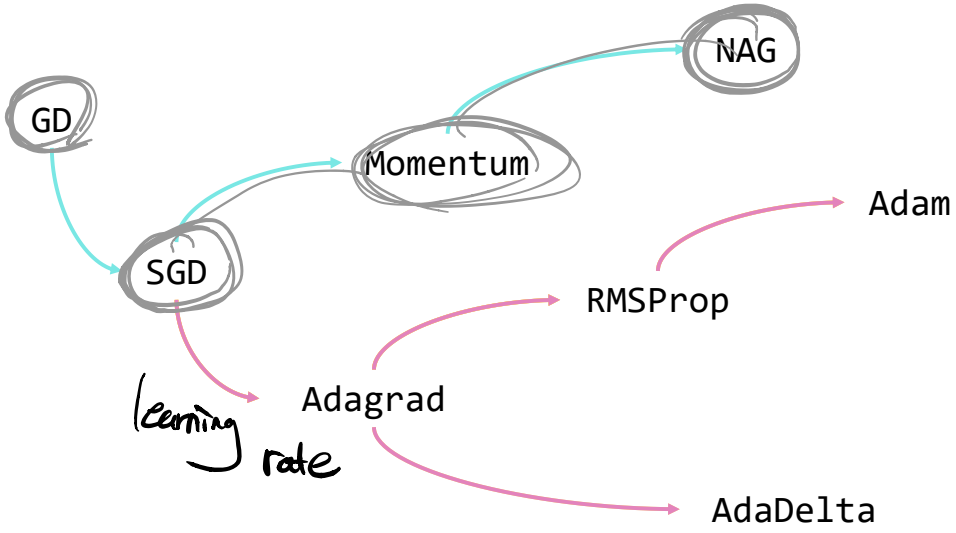
Momentum update:



Nesterov Momentum



Optimizer 발전 과정



Adagrad

$$\eta' = \frac{\eta}{\sqrt{G_t}}$$

Idea : learning rate를 gradient에 따라 다르게 탐색하자!

w_0 w_1

-5 0.1

-20 0.05

$$w_{new} = w - \frac{\eta}{\sqrt{G_t + \varepsilon}} \frac{\partial L}{\partial w}$$

ε : 10^{-8}

$$G_t = G_{t-1} + \left(\frac{\partial L}{\partial w}\right)^2$$

양수 → 여태까지 gradient 양을 scale을 조절

G: [425, 0.0125]

20 0.1

$w_0 \Rightarrow \frac{0.01}{20} = 0.0005$

$w_1 \Rightarrow \frac{0.01}{0.1} = 0.1$

Problem of Adagrad

$G_t \to \infty$ $\eta' \to 0$

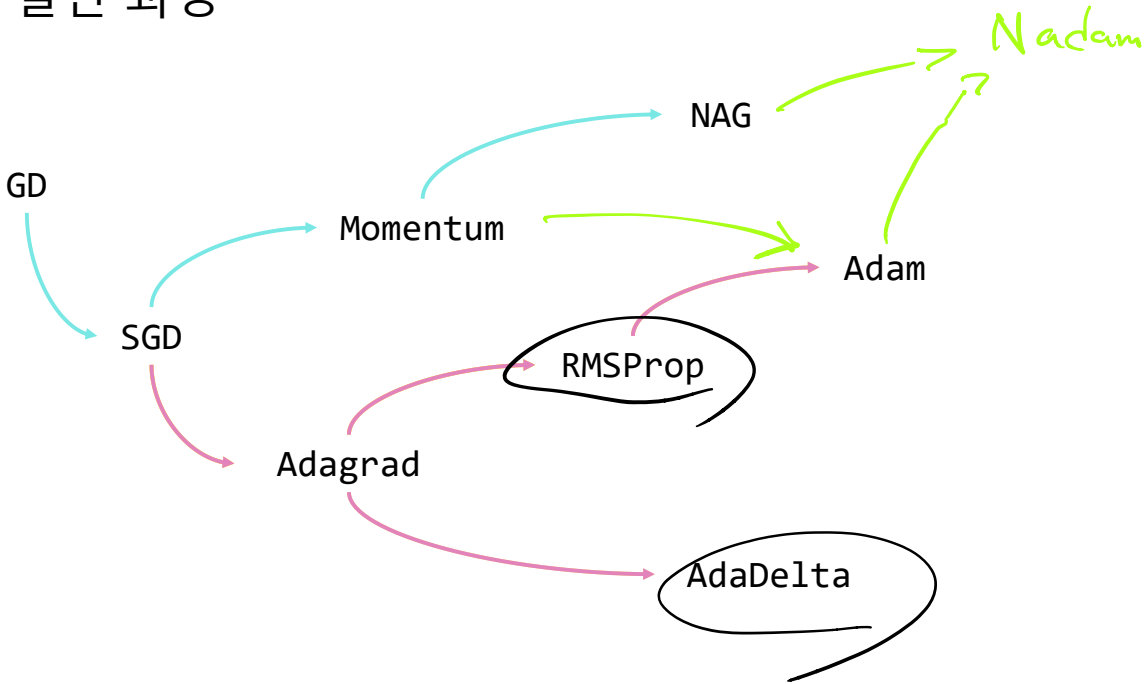
$$\eta' = \frac{\eta}{\sqrt{G_t + \varepsilon}} \to \infty$$

G_t가 무한정 커지면...?

➔ learning rate η가 점점 0에 가까워진다!

➔ 학습이 안 된다..!

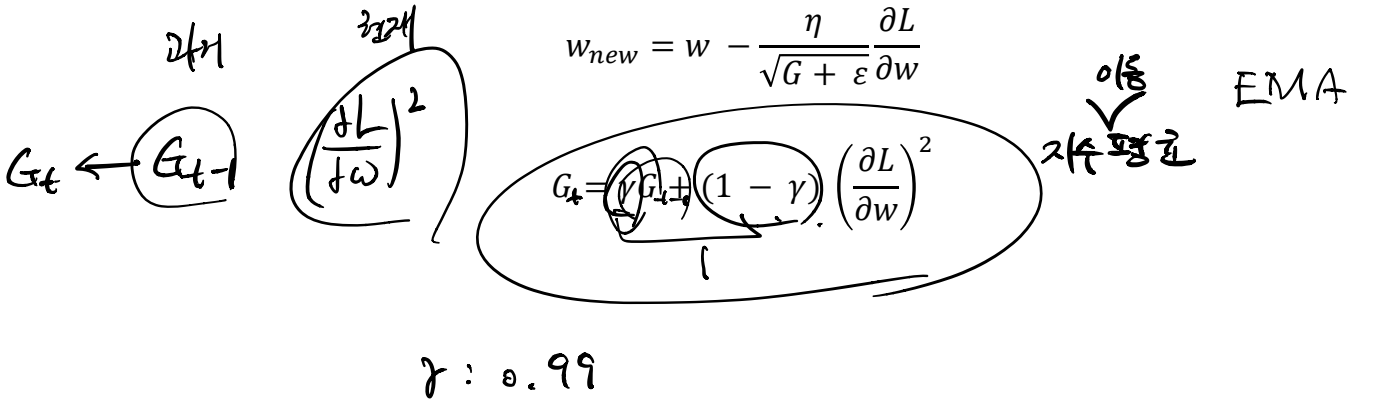
Optimizer 발전 과정



RMSProp

Adagrad

Idea : gradient에 따라 learning rate 바꾸는 건 좋은데, 무한정 커지는 걸 방지하자!



AdaDelta

$$\Delta w \propto \frac{\partial L}{\partial w} \propto \frac{1}{u(w)}$$

Idea : gradient에 따라 learning rate 바꾸는 건 좋은데, 무한정 커지는 걸 방지하자!

$$w_{new} = w - \Delta w$$

$$\Delta w = \frac{\sqrt{s + \varepsilon}}{\sqrt{G + \varepsilon}} \frac{\partial L}{\partial w}$$

$$\Delta w \propto \frac{\frac{\partial L}{\partial w}}{\frac{\partial^2 L}{\partial w^2}} \propto u(w)$$

$$G = \gamma G + (1 - \gamma) \left(\frac{\partial L}{\partial w}\right)^2$$

$$s = \gamma s + (1 - \gamma) \Delta w^2$$

Adam → Momentum + RMSProp.

Idea : RMSProp에 Momentum을 합친다면..?

$$w_{new} = w - \frac{\eta}{\sqrt{\hat{v}_t + \varepsilon}} \hat{m}_t \rightarrow g = \frac{\partial L}{\partial w}$$

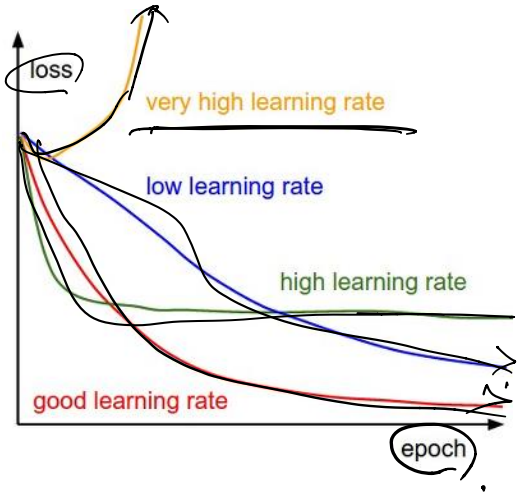
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

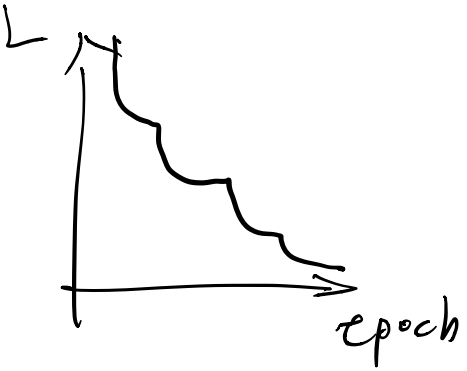
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\frac{\partial L}{\partial w}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\left(\frac{\partial L}{\partial w}\right)^2$$

$\Rightarrow 0$

Learning rate decay





Optimizers in Keras

```
keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```

```
keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=None, decay=0.0)
```

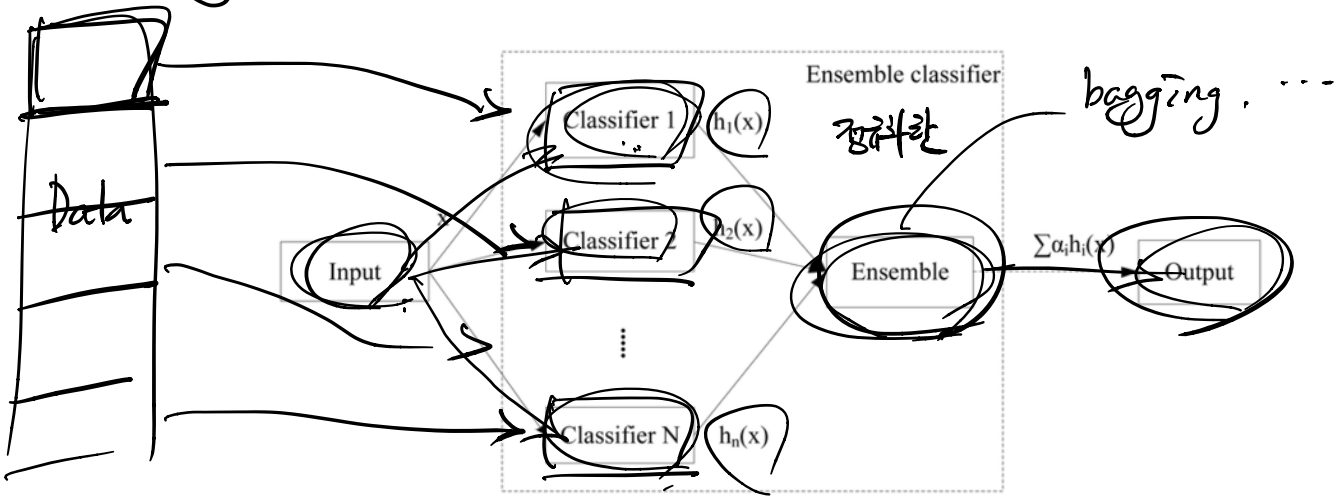
```
keras.optimizers.Adagrad(lr=0.01, epsilon=None, decay=0.0)
```

```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
```

Regularization

Model Ensemble

overfitting Idea : 모델 하나만 학습시킬 게 아니라, 여러 개를 학습시켜서 합하자!

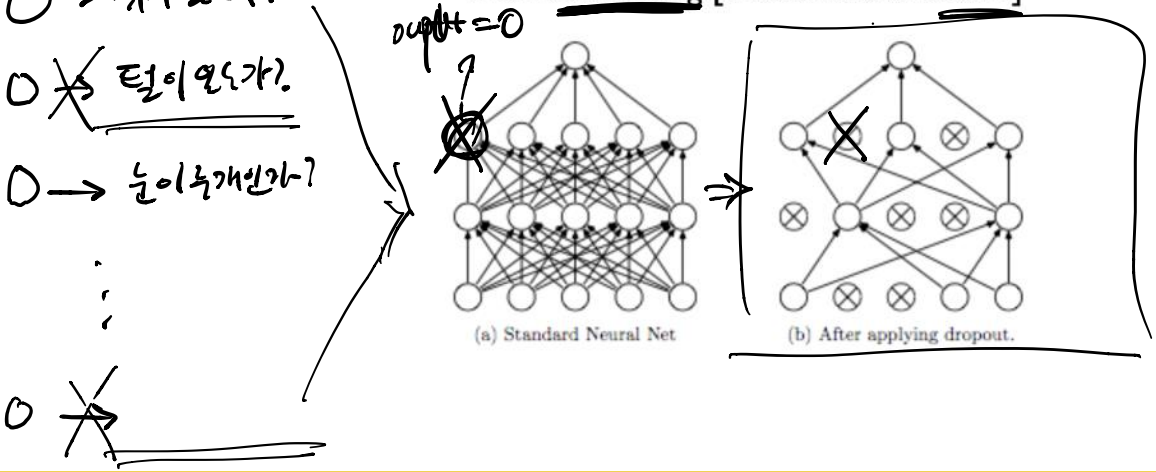


Dropout

$p = 0.8$

모델 앙상블

Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Srivastava et al. 2014]

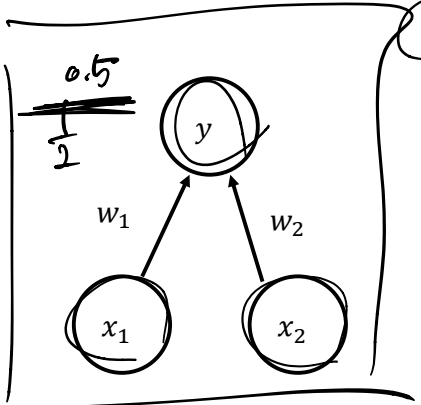


Dropout for test data..?

output ← Input

$y = f(x, z)$ → random mask.

$$y = f(x) = E_z[f(x, z)] = \int p(z) f(x, z) dz$$



$y = w_1 x_1 + w_2 x_2$.

Test time: $E[y] = w_1 x_1 + w_2 x_2$

During Training: $E[y] = \frac{1}{4}(w_1 x_1 + w_2 x_2) + \frac{1}{4}(w_1 x_1 + 0) + \frac{1}{4}(0 + w_2 x_2) + \frac{1}{4}(0 + 0)$

$= \frac{1}{2}(w_1 x_1 + w_2 x_2)$

if keep

Dropout in Keras

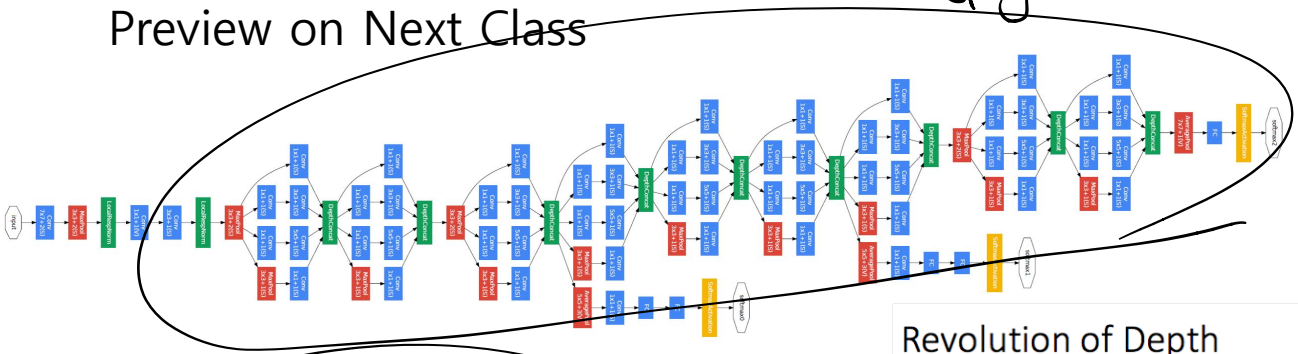
```
model = Sequential([
    tf.keras.layers.Flatten(),
    ...
    tf.keras.layers.Dropout(0.2)
])
```

tf.keras.

```
tf.keras.layers.Dropout(rate, noise_shape=None, seed=None)
```

Preview on Next Class

GoogLeNet



ResNet