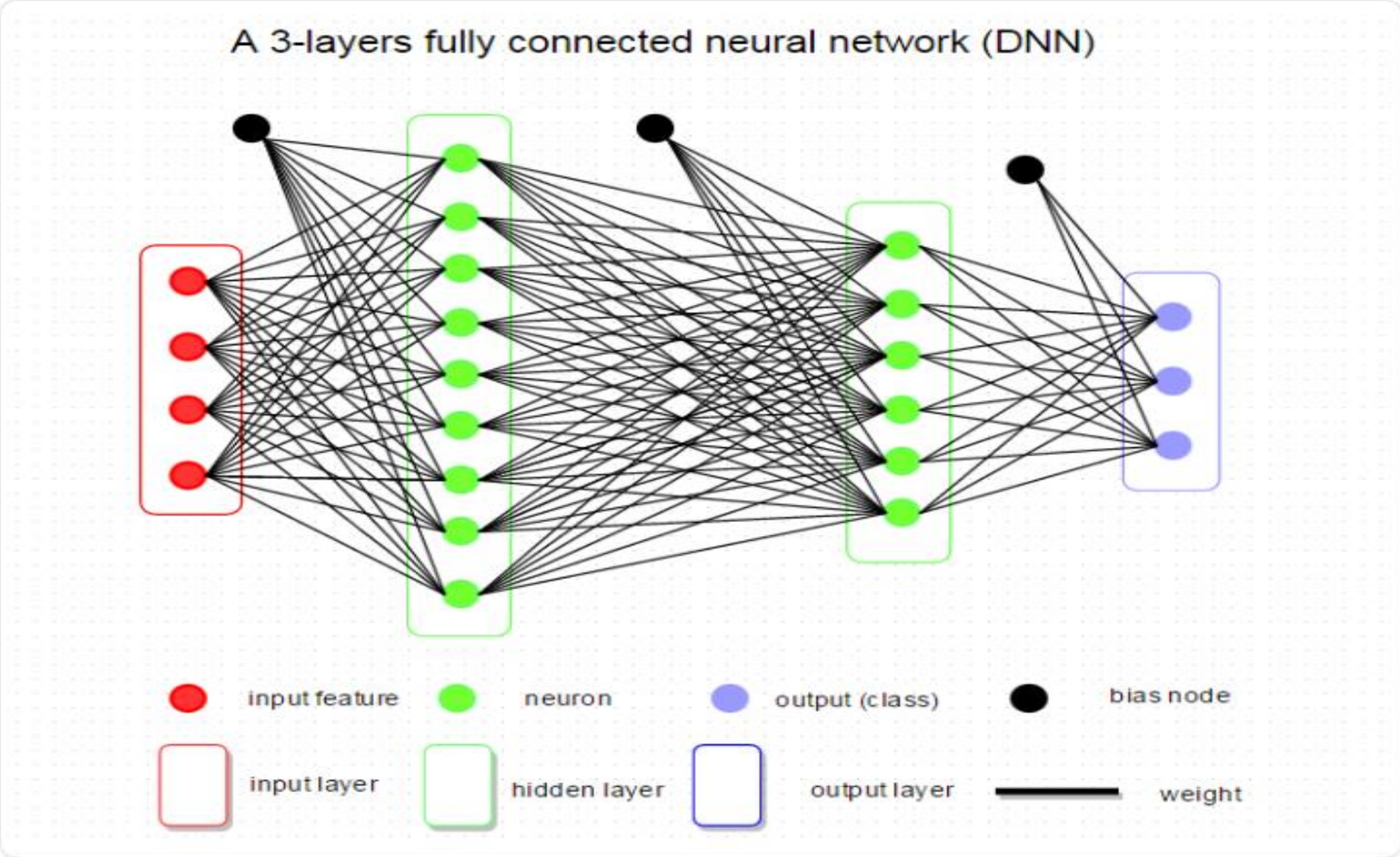
인공지능(AI)

06주차.심층신경망(Deep Neural Network) 구성에 대한 이해

학습목표

1. 딥러닝의 학습 알고리즘을 설명할 수 있다.
2. Gradient와 Gradient Descent에 대해 설명할 수 있다.
3. Loss 함수와 Loss 함수의 종류에 대해 설명할 수 있다.
4. Optimizer와 Learning Rate에 대해 설명할 수 있다.
5. Activation Function에 대해 설명 할 수 있다.

비유 내용



- 학습(learning)은 최적의 가중치(weight)들의 조합을 찾는 과정

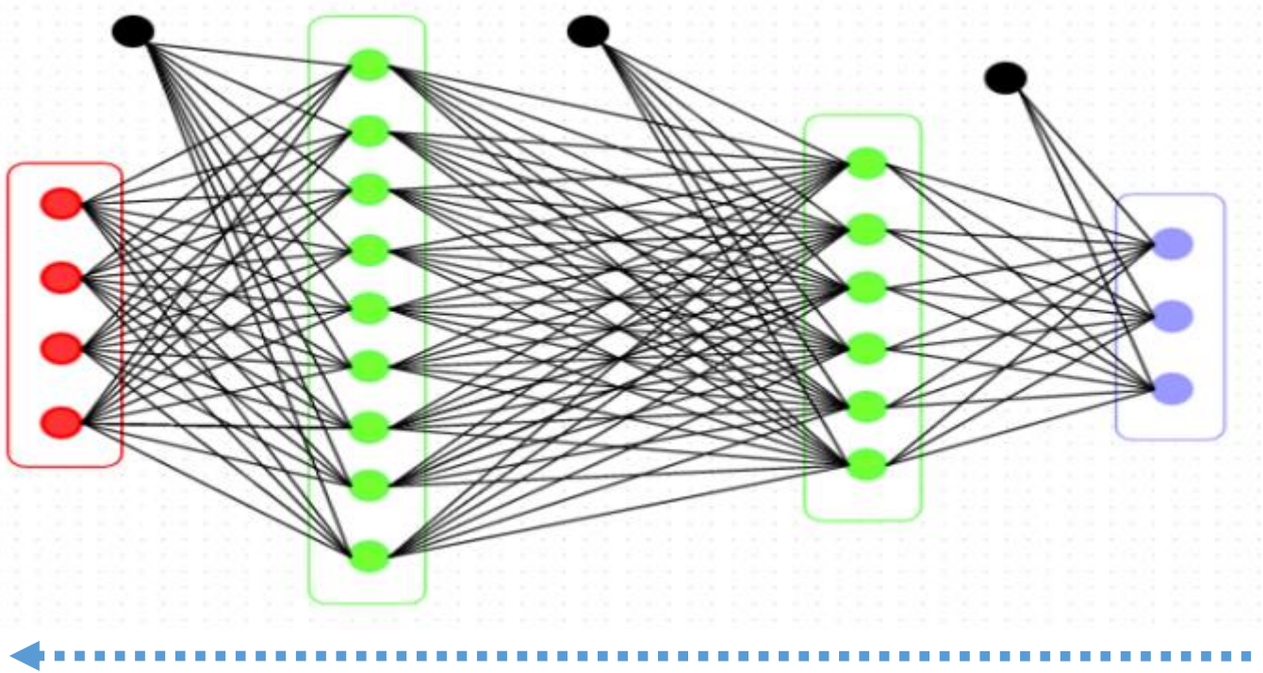
- 노드(뉴런) 수가 많아지고 레이어가 깊어질 수록 학습해야 할 weight수는 지수적으로 증가

- DNN(Deep Neural Network)에서 수 많은 weight들의 최적의 조합을 찾는 알고리즘은 무엇일까?

출처 : http://www.parallelr.com/r-deep-neural-network-from-scratch/

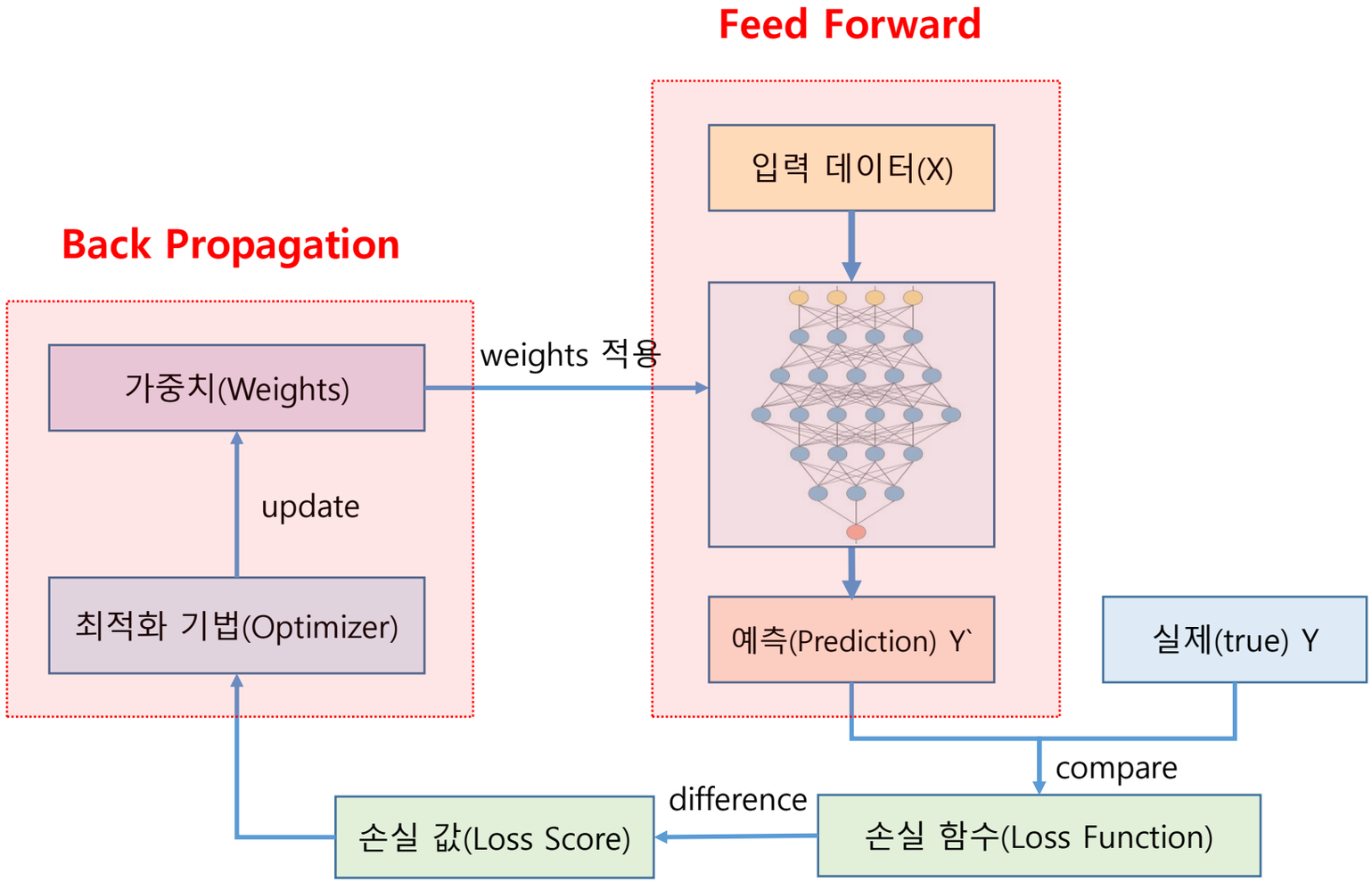
Feed Forward

input 값과 weight / bias를 이용한 예측 값 계산



Back Propagation(오차 역전파)

Gradient Descent 알고리즘을 이용한 weight 계산 및 업데이트

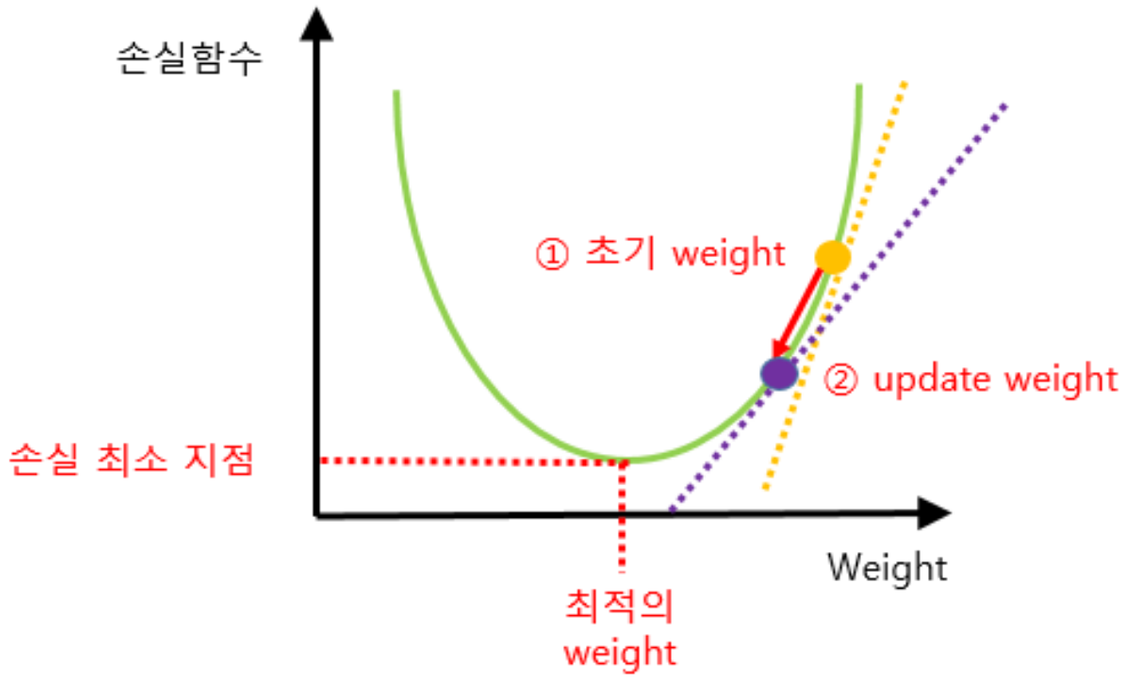

Feed Forward
입력 데이터(X)
Back Propagation
가중치(Weights)
weights 적용
update
최적화 기법(Optimizer)
예측(Prediction) Y`
실제(true) Y
compare
difference
손실 값(Loss Score)
손실 함수(Loss Function)

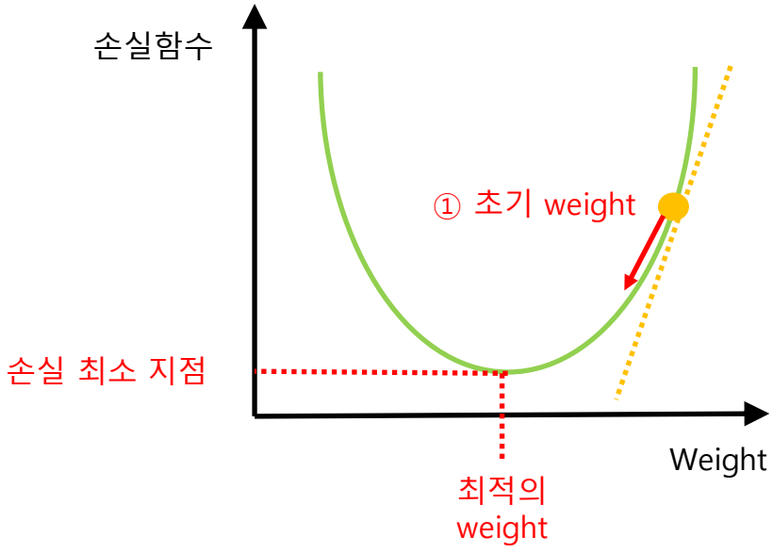
학습(learning)

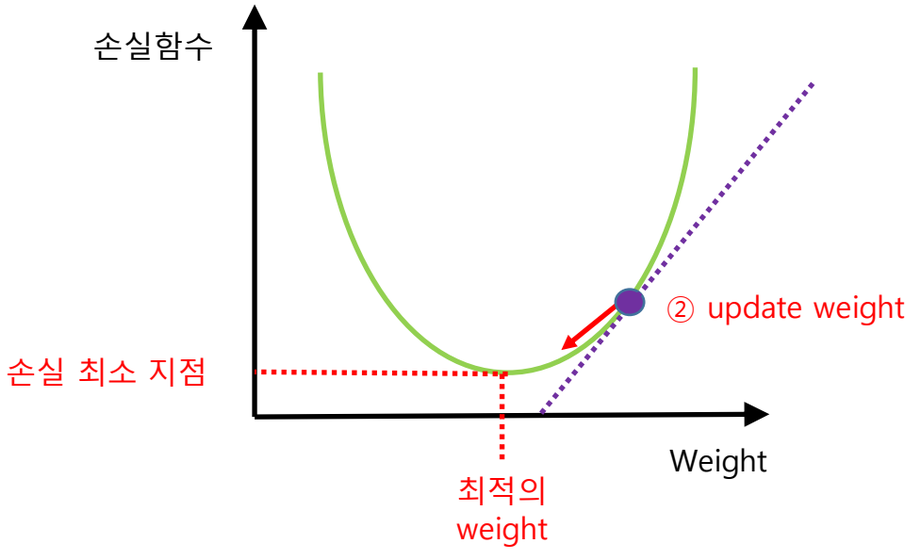
최적의 weight의 조합을 찾는 과정, 손실 값(실제 값과 예측 값의 차이)이 최소가 되는 weight를 찾는 과정

Gradient Descent

Weight와 손실함수(loss function)의 관계를 이용하여 최적의 weight를 찾는 과정
특정 지점 w에서 기울기가 가장 가파르게 하강하는 곳을 따라 learing rate만큼 이동하면서 손실 값 최소 지점으로 이동해 가면서 최적의 weight를 찾는 알고리즘







목표 : 최적의 weight 값 계산

(손실 최소 지점, 접선의 기울기(미분 값)가 0)

방법 : ① 초기 weight 값에서 접선의 기울기 계산

기울기가 0이 아니면 화살표의 크기 만큼 이동

② 이동한 위치에서 접선의 기울기 계산

기울기가 0이 아니면 화살표의 크기 만큼 이동

…

기울기가 0이 되는 weight을 찾기 까지 계속 반복

대상 : NN(Neural Network) 전체의 weight 값에 대해 수행

딥러닝 프레임워크 Optimizer

접선의 기울기 → Gradient

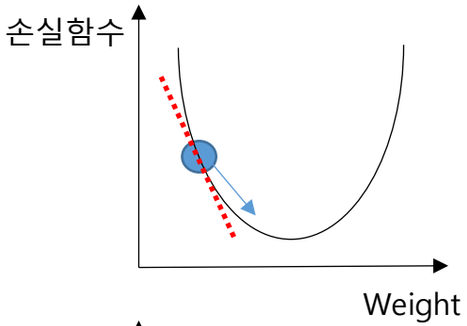
화살표의 크기 → Learning Rate

Gradient

손실 함수의 weight들에 대한 모든 기울기로 미분(편미분)한 값의 집합(벡터)
가중치 매개변수의 값을 변화시켰을 때 손실 함수의 변화량

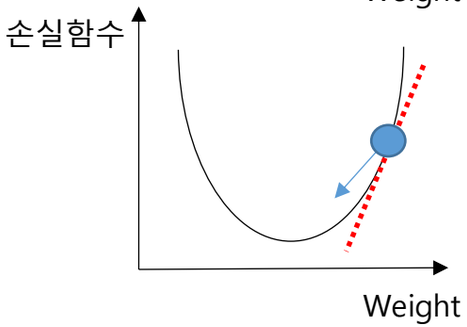
Gradient가 음수(-)이면

weight를 양의 방향으로 이동



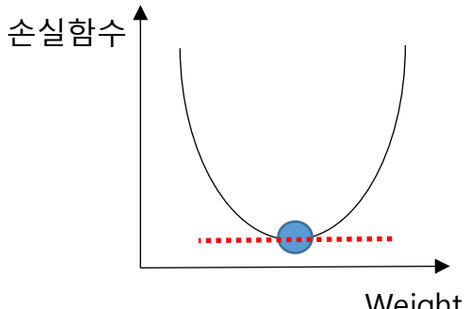
Gradient가 양수(+)이면

weight를 음의 방향으로 이동



Gradient 0이면

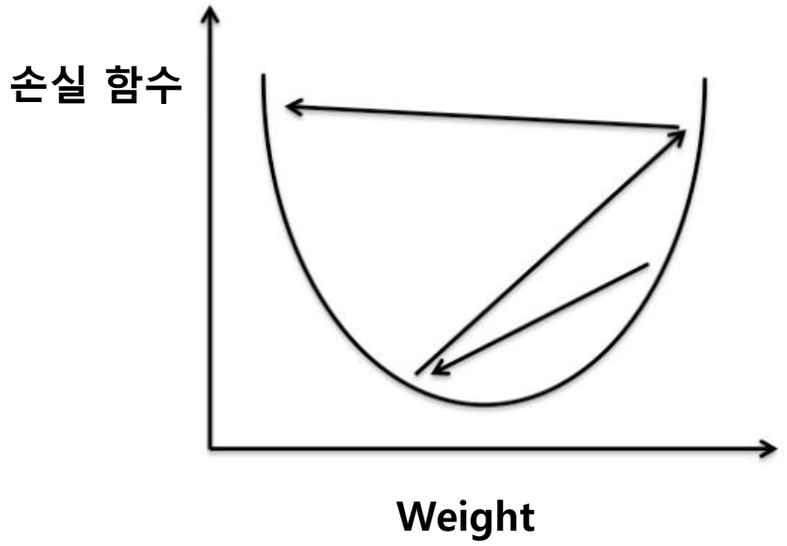
weight 갱신 종료(학습 종료)



Learning Rate

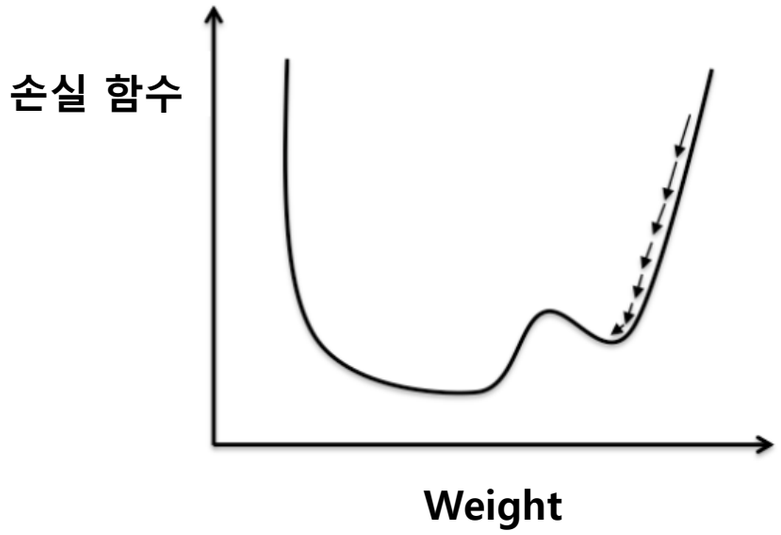
Gradient에 의해 결정된 이동방향에 따라 얼마만큼 이동 크기를 결정하기 위한 Hyper Parameter

learning rate 가 너무 큰 경우



Overshooting 발생

learning rate 가 너무 작은 경우



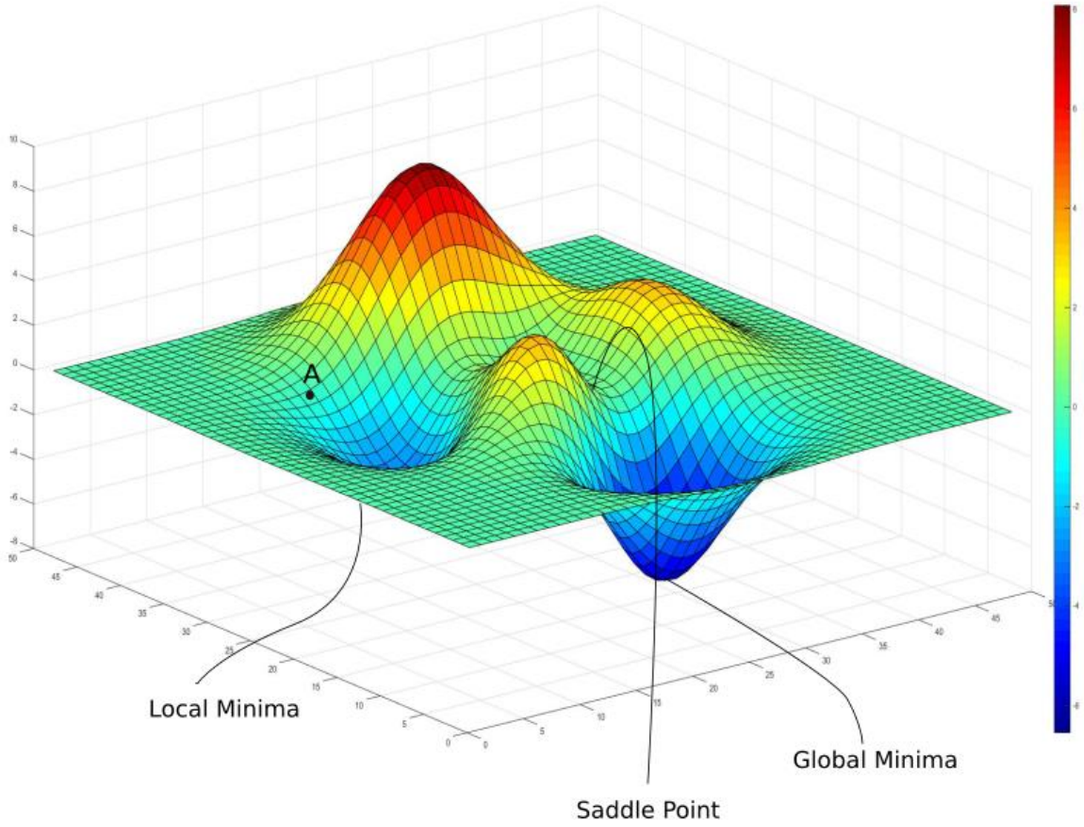
학습속도 저하, local minima

Learing Rate는 일반적으로 0.01 ~ 0.001 사이의 값을 가장 많이 사용

실제 손실함수와 Weight 간의 관계를 표현한 그래프

최종 목표 지점은 Global Minima이며,

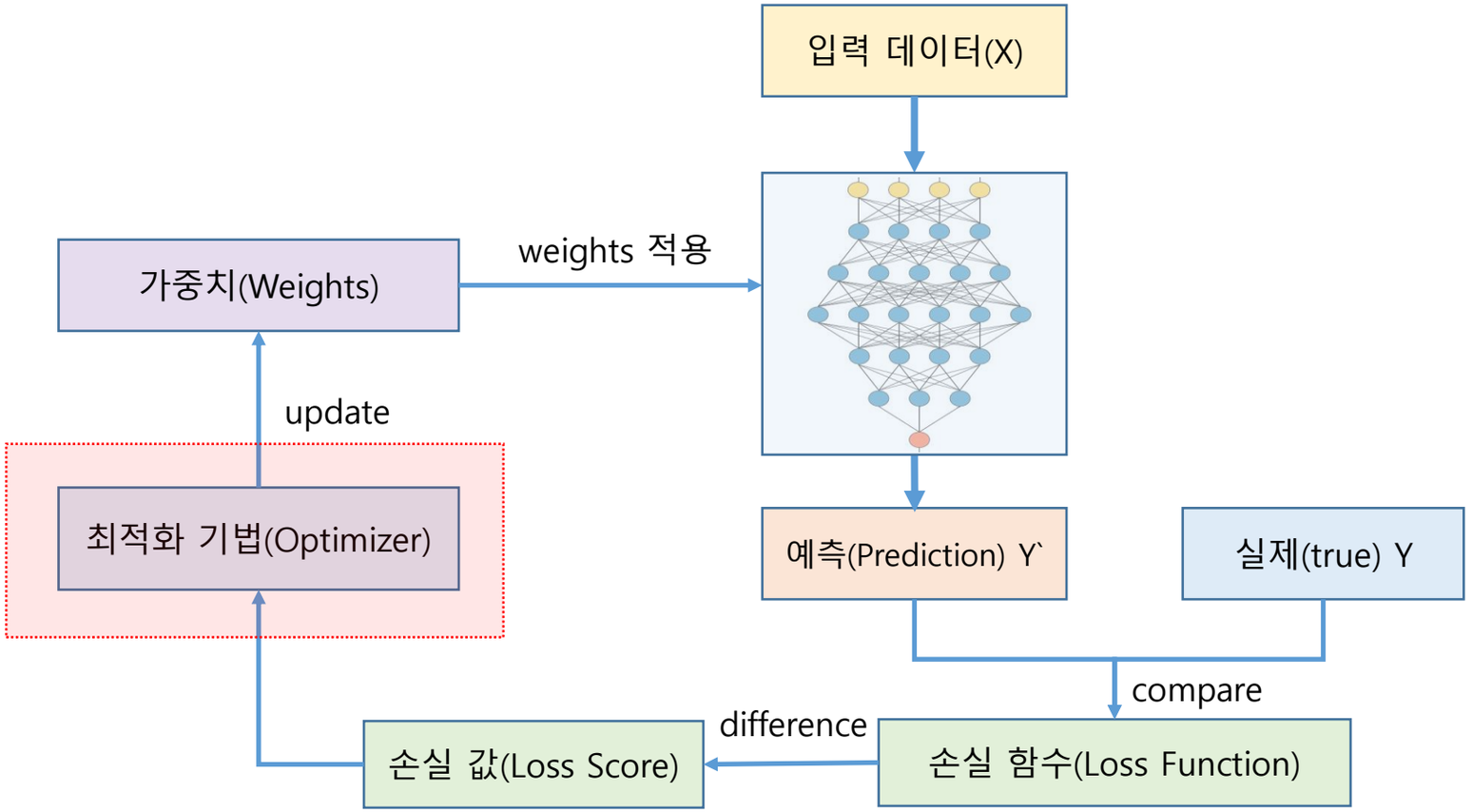
Saddle Point와 Local Minima에서 학습이 종료되는 함정(trap)에 빠지지 않도록 주의 필요



출처 : https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

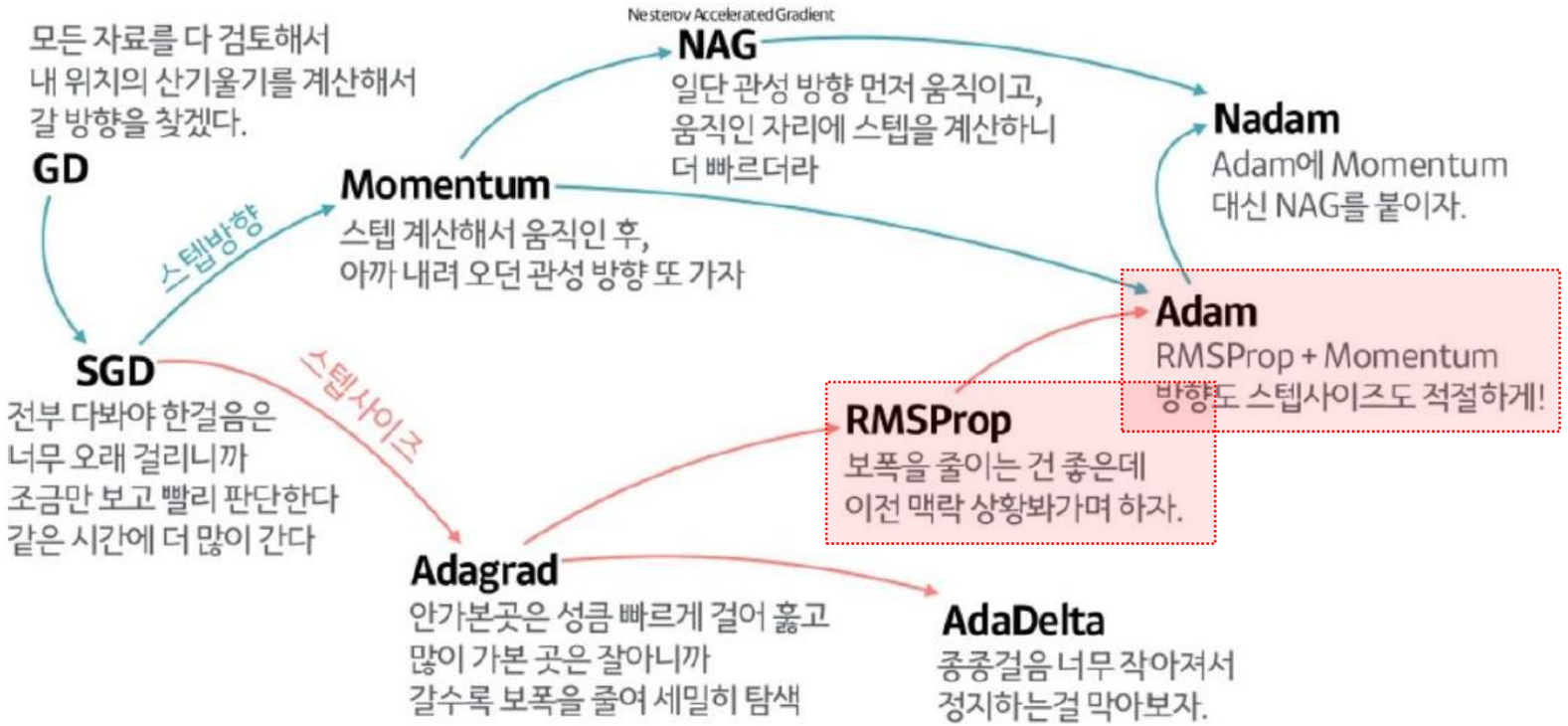
Optimizer

최적의 weight를 찾아 업데이트하기 위한 알고리즘
Gradient 계산, learning rate 만큼 이동하여 weight 계산, weight 업데이트 수행 등



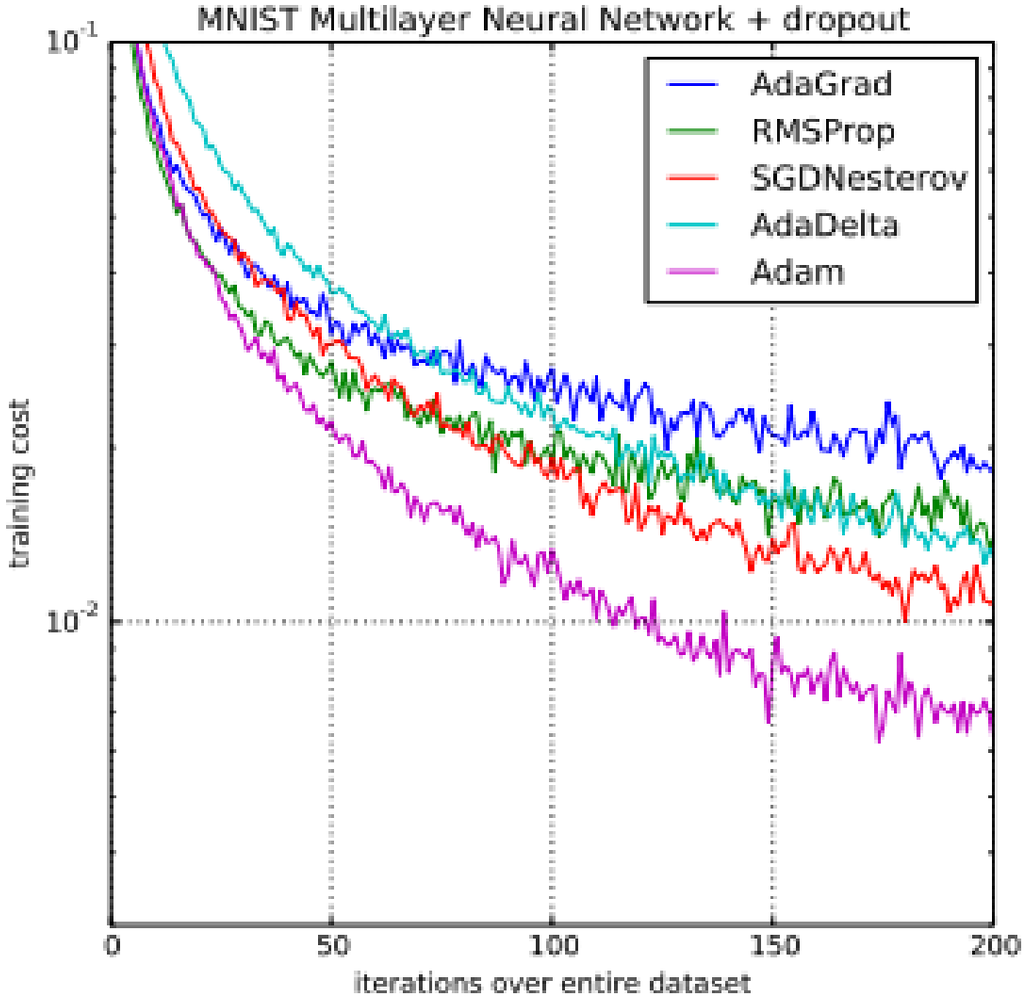
* SGD
* Adagrad
* RMSprop
* Adam
  …

여러 Optimizer 중에서 일반적으로 RMSProp이나 Adam Optimizer를 많이 사용



출처 : https://www.slideshare.net/yongho/ss-79607172

Optimizer의 학습 속도 비교



출처 : https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

RMSprop Optimizer

```
from keras.optimizers import RMSprop
optimizer = RMSprop(lr=0.001)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

Test Accuracy : 97.93

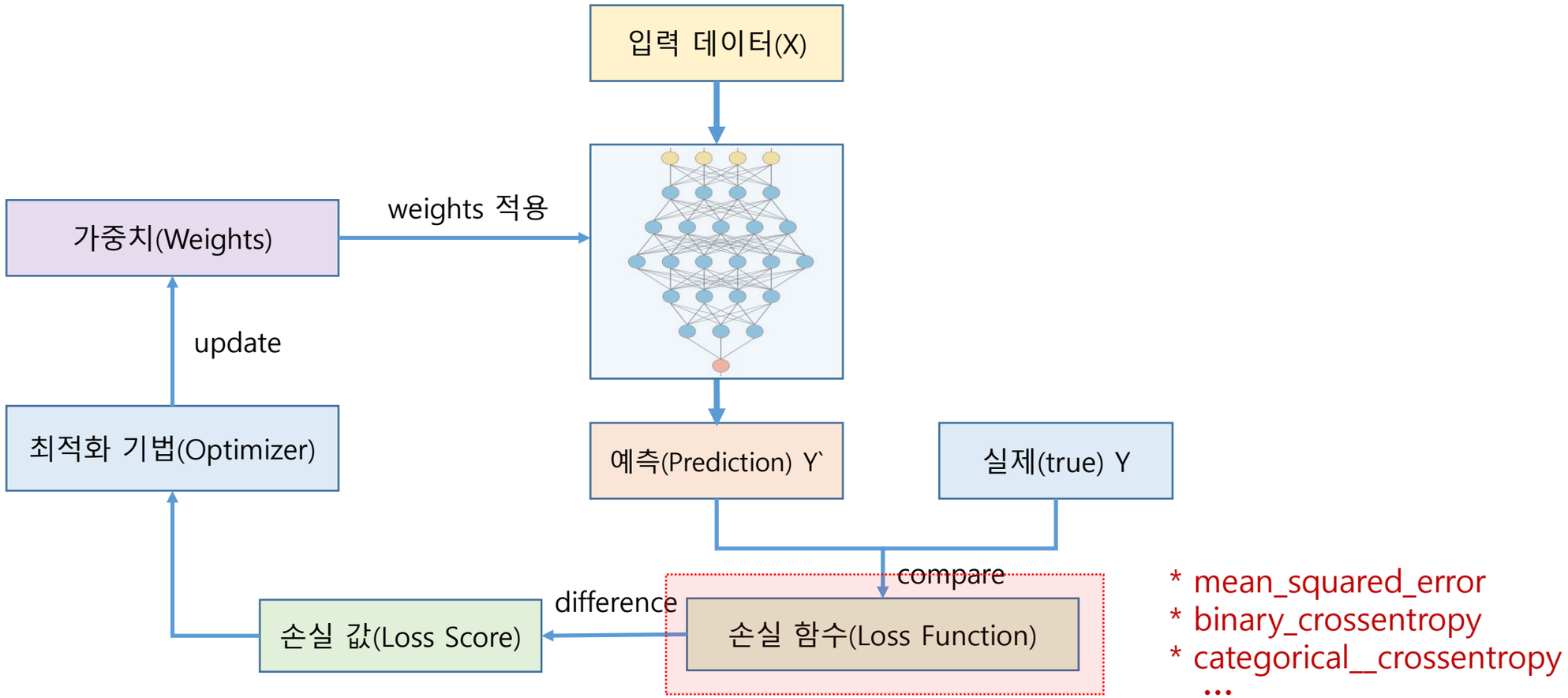
Adam Optimizer

```
from keras.optimizers import Adam
optimizer = Adam(lr=0.001)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

Test Accuracy : 98.12

손실 함수(loss function)

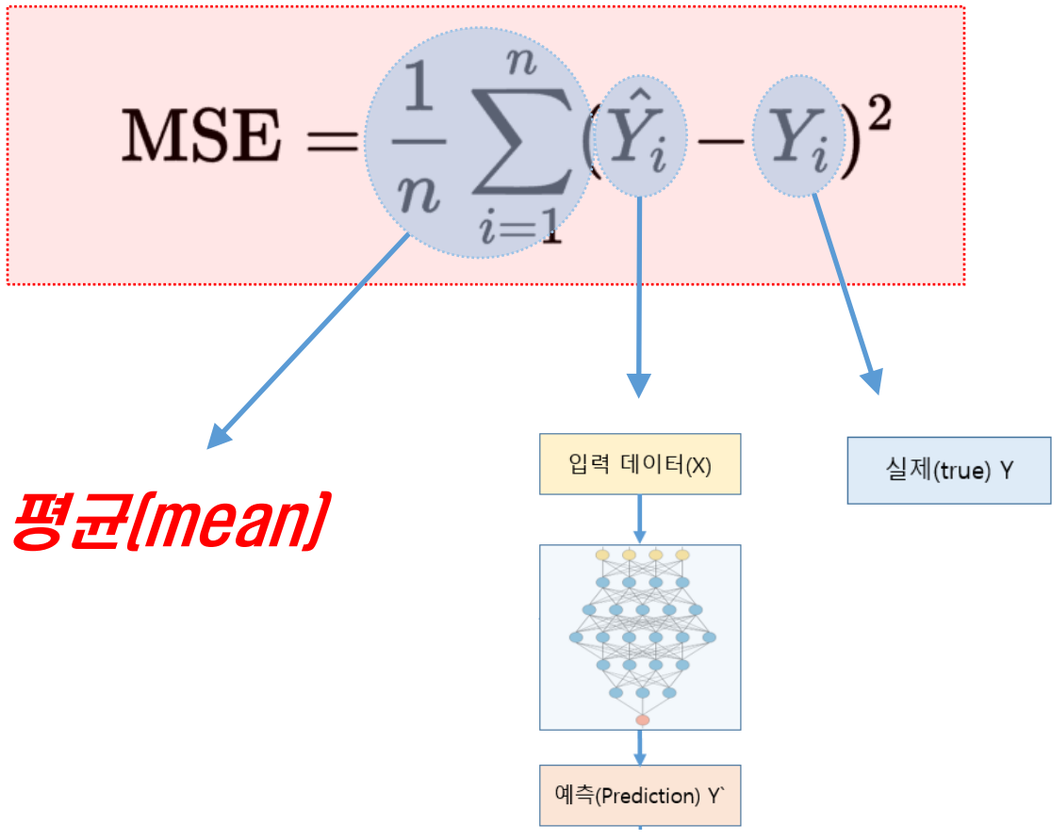
실제 값과 예측 값을 비교하여 차이를 계산하기 위한 함수



평균 제곱 오차(Mean Squared Error, MSE)

예측 값에서 실제 값을 뺀 차이 값을 제곱한 후 평균을 구하여 loss를 계산

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$




```
model.compile(optimizer=optimizer, loss='mse', metrics=['accuracy'])
```

교차 엔트로피(Cross Entropy)

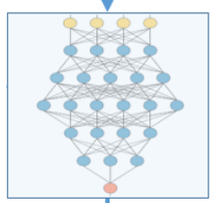
예측 값에 자연로그(nature log)를 취해서 실제 값과 곱한 후 모두 더해서 loss를 계산

$$E = -\sum_{k} t_k \log y_k$$

실제(true) Y

자연 log

입력 데이터(X)



예측(Prediction) Y`

binary cross entropy

label이 두 종류인 경우 (0 또는 1을 예측하는 경우)

```
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
```

categorical cross entropy

label이 두 종류 이상 인 경우(0 ~ 9, 0 ~ 99, 0 ~ N 등 두 종류 이상의 level을 예측하는 경우

```
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

RMSProp Optimizer

In [81]:
```
# from keras.optimizers import RMSprop
# optimizer = RMSprop(lr=0.001)
```

Adam Optimizer

In [82]:
```
from keras.optimizers import Adam
optimizer = Adam(lr=0.00
```

In [83]:
```
model.compile(optimizer=
              loss='cate
              metrics=['
```

optimizer.ipynb 실습

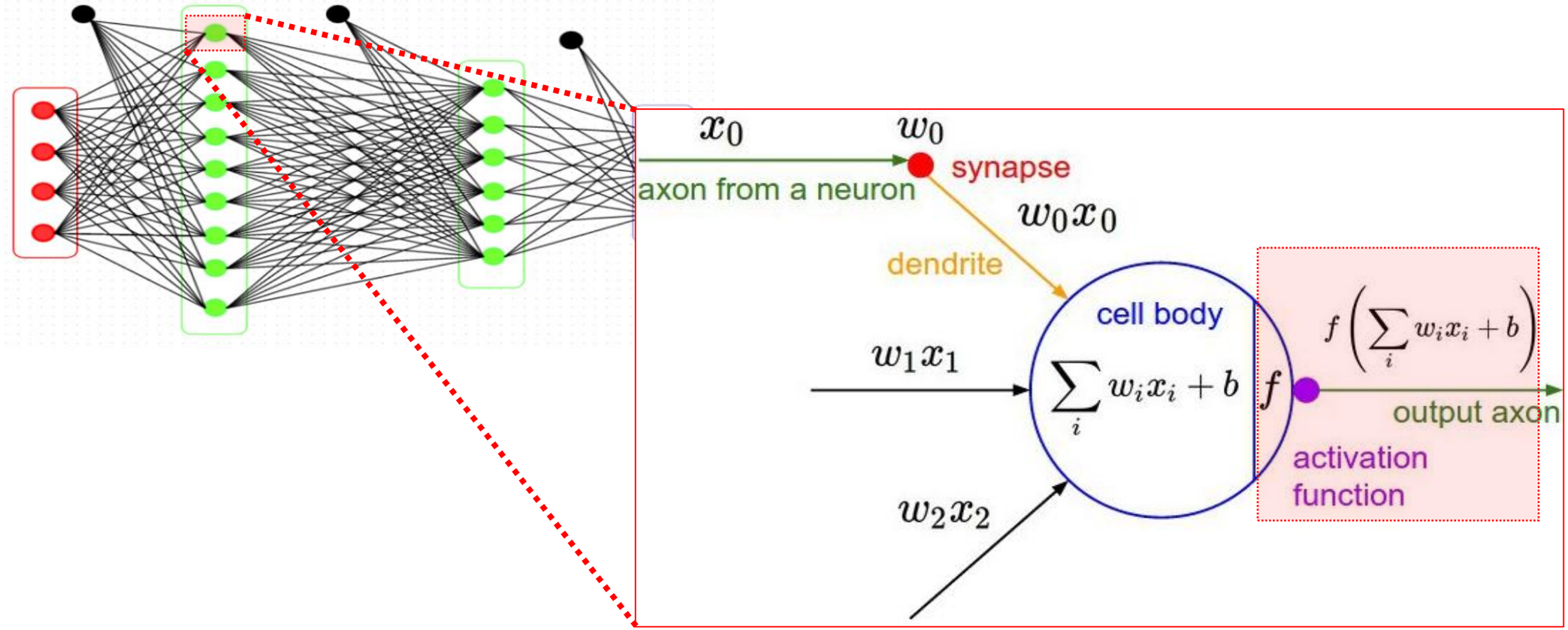
In [84]:
```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_11 (Dense)             (None, 512)               401920
_________________________________________________________________
dense_12 (Dense)             (None, 10)                5130
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

활성화 함수(Activation Function)

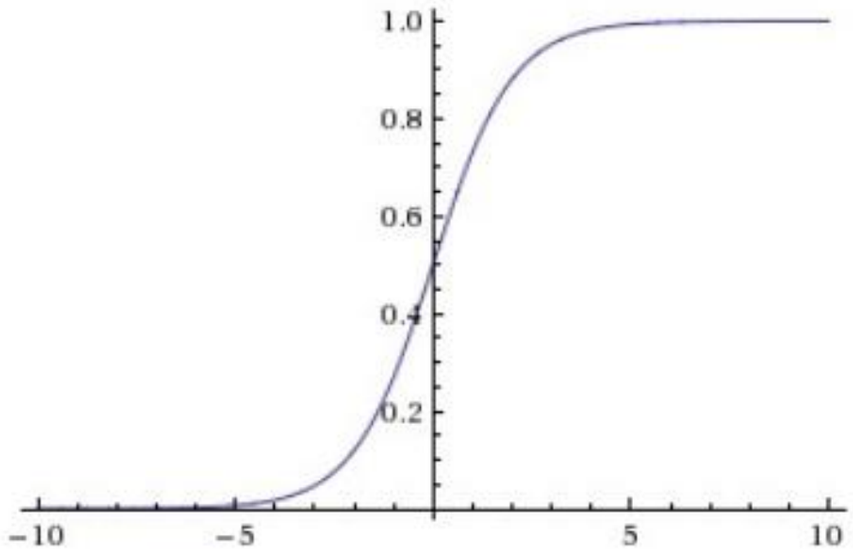
단일 노드(node)에 여러 신호가 들어오면, 다음 레이어(layer)에 보낼 신호의 강도를 결정

NN(Neural Network)에 비선형성(non linearity) 추가하는 역할



시그모이드(Sigmoid)

어떠한 값이 입력되더라도 0 ~ 1 사이의 값 만을 출력하는 활성화 함수
그레이언트 소실(gradient vanishing) 문제 발생 가능

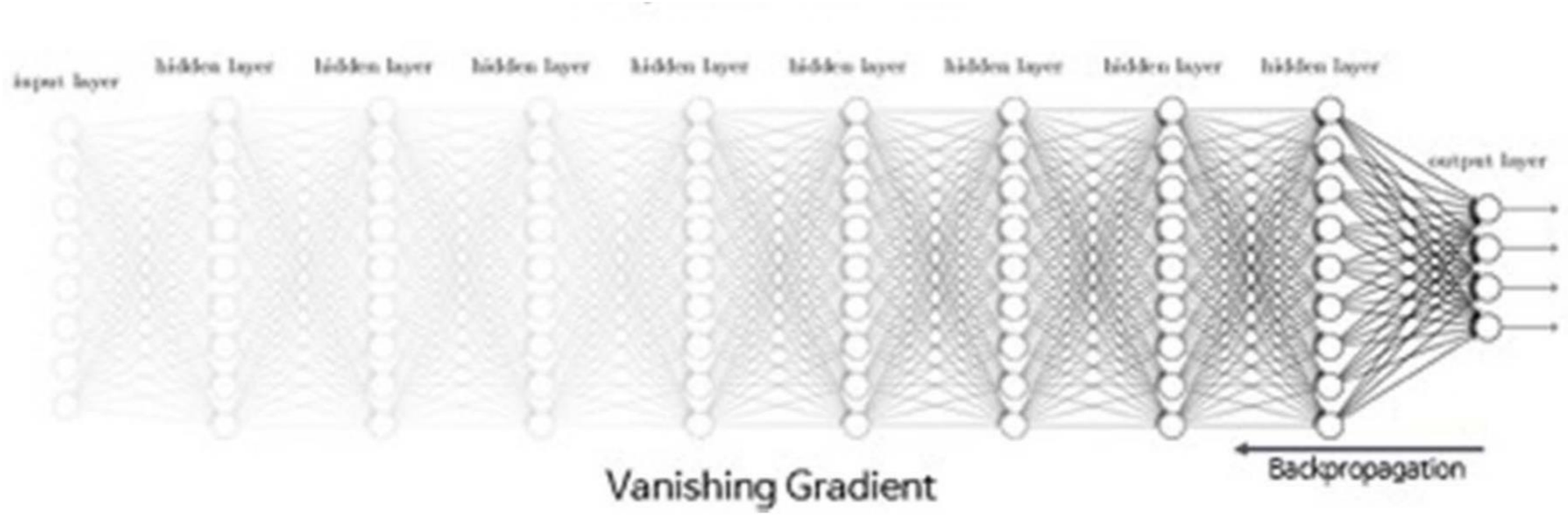


$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

```
model.add(layers.Dense(512, activation='sigmoid', input_shape=(28 * 28,)))
```

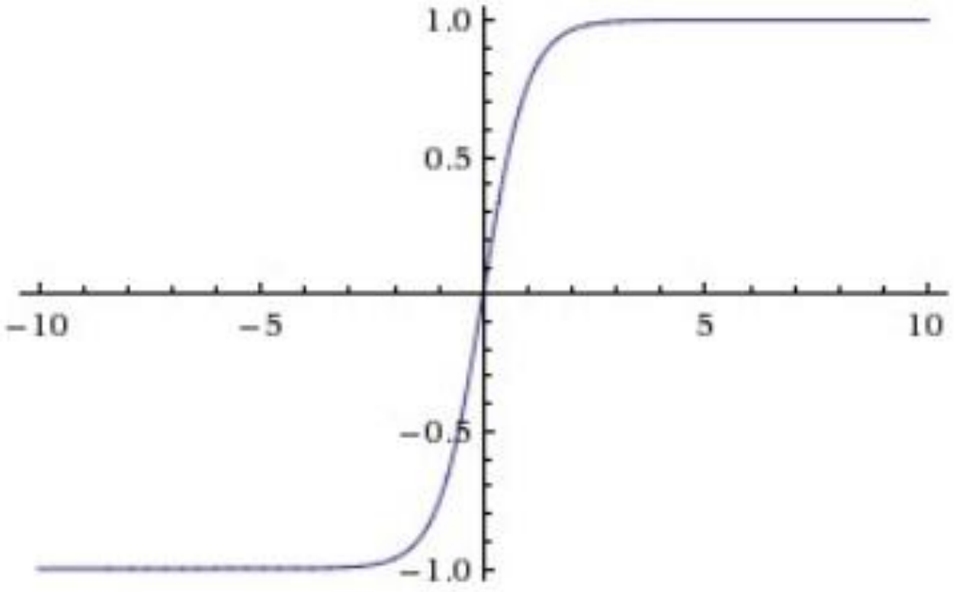
Gradient vashing

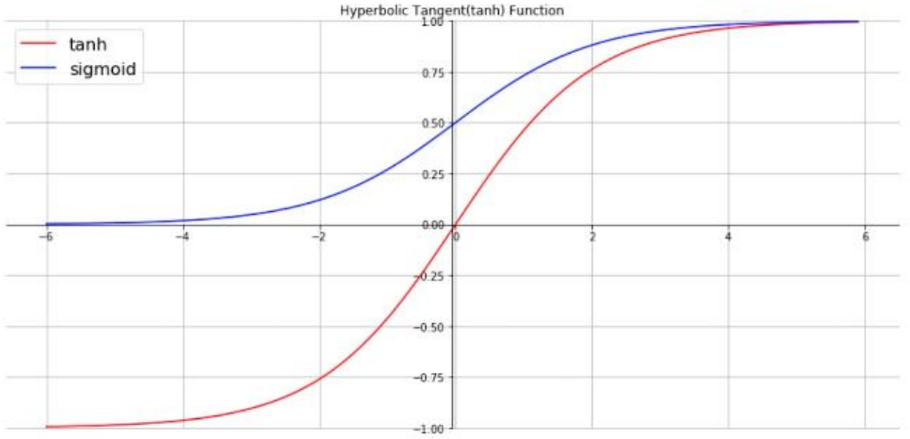
Backpropagation을 수행하기 위해 여러 단계의 layer를 거치게 되면,
소수점 이하의 작은 값을 여러 번 곱하게 됨으로써 gradient가 점점 작아져 0에 가깝게 수렴하게 되면서
더 이상 학습이 진행되지 않는 상태

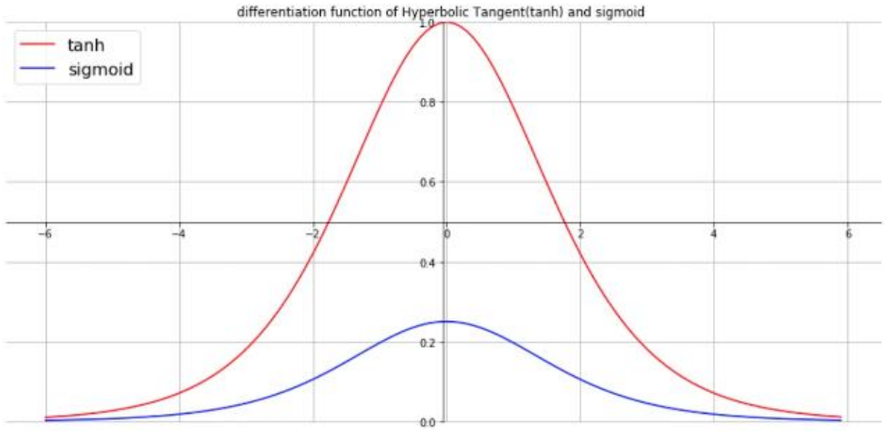


tanh(Hyperbolic tan)

-1 ~ 1 사이의 값을 출력
sigmoid 보다 gradient vanishing 발생확률 ¼ 정도 낮음



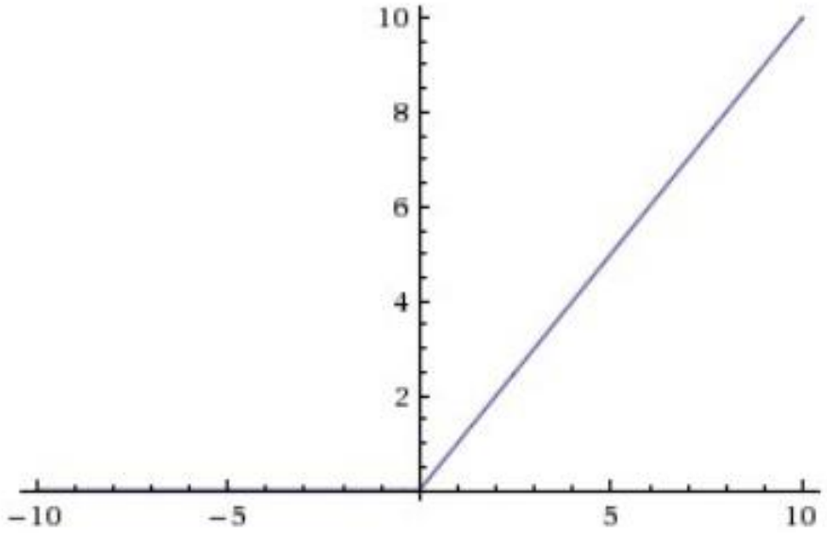





```
model.add(layers.Dense(512, activation='tanh', input_shape=(28 * 28,)))
```

ReLU(Rectified Linear Unit)

양수가 입력되면 양수 그대로를 출력하고, 음수가 입력되면 0으로 출력하는 활성화 함수
음수가 입력되면 0을 출력하기 때문에 한번 음수가 입력되면 해당 노드는 더 이상 학습이 진행되지 않음

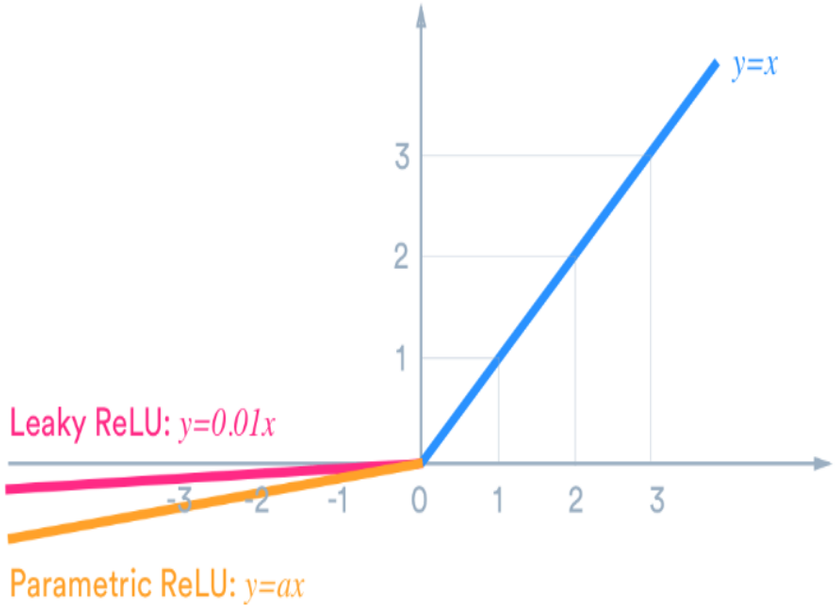


$$f(x) = max(0, x)$$

```
model.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
```

Leaky ReLU(Rectified Linear Unit)

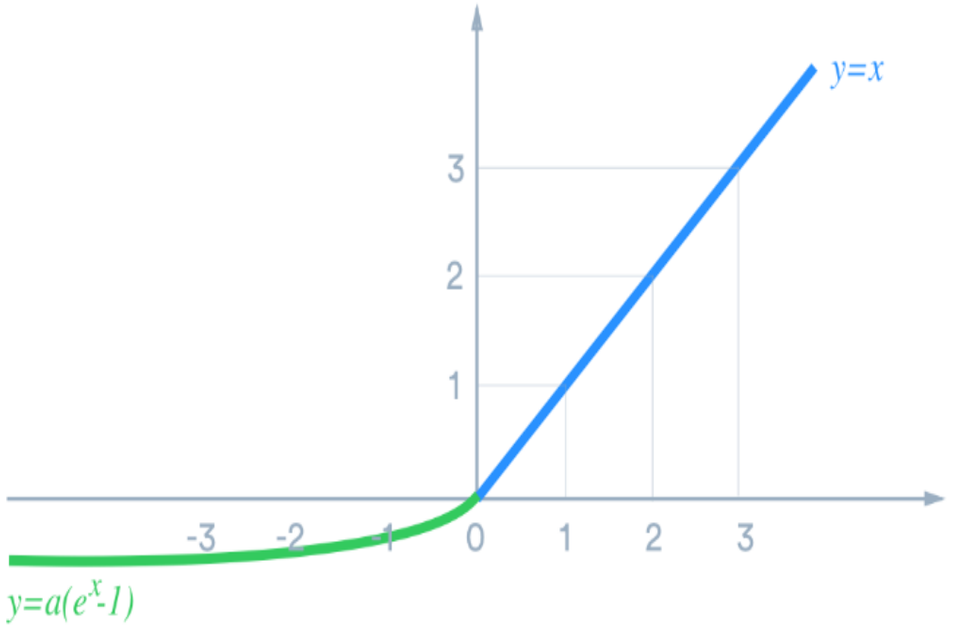
음수 값이 입력되면 0을 출력하여 학습이 진행되지 않는 ReLU의 문제점을 개선하기 위한 활성화 함수
음수가 입력되면 0으로 출력되지 않고 1보다 작은 값을 곱한 음수 값으로 출력




```
from keras.layers.advanced_activations import LeakyReLU
activation = LeakyReLU(alpha=.001)
model.add(layers.Dense(512, activation=activation, input_shape=(28 * 28,)))
```

ELU(Exponential ReLU)

음수가 입력되면 Exponential 적용한 음수 값으로 출력하는 활성화 함수
Exponential 계산 비용 필요




```
from keras.layers.advanced_activations import ELU
activation = ELU(alpha=.001)
model.add(layers.Dense(512, activation=activation, input_shape=(28 * 28,)))
```

Activation - Leaky Relu

In [67]:
```
# from keras.layers.advanced_activations import LeakyReLU

# model = models.Sequential()
# lrelu = LeakyReLU(alpha=.001)
```

Activation - ELU

activation.ipynb 실습

In [68]:
```
from keras.layers.adva

model = models.Sequenti
activation = ELU(alpha=.001)
```

In [69]:
```
model.add(layers.Dense(512, activation=activation, input_shape=(28 * 28,)))
model.add(layers.Dense(10, activation='softmax'))
```

형성평가

번호	문제	보기	정답	해설
1	DNN(Deep Neural Network)에서 Back Propergation를 통해 예측 값을 계산하고 Feed Forward를 통해 weight를 계산하고 학습한다.		X	DNN(Deep Neural Network)에서 Feed Forward 프로세스를 통해 예측 값을 계산하고 Back Propergation을 통해 weight를 계산하고 학습한다.

형성평가

번호	문제	보기	정답	해설
2	다음 중 Gradient에 관한 설명으로 올바르지 않은 것은?	① Gradient는 손실 함수 weight들에 대한 기울기로 미분(편미분)한 값의 집합을 말한다. ② Gradient가 음수(-)이면 weight를 양의 방향으로 이동시킨다. ③ Gradient가 양수(+)이면 weight를 음의 방향으로 이동시킨다. ④ Gradient가 0이면 weight 갱신을 종료시키지 않고 학습을 계속 진행한다.	④	- Gradient는 손실 함수 weight들에 대한 기울기로 미분(편미분)한 값의 집합을 말한다. - Gradient가 음수(-)이면 weight를 양의 방향으로 이동시킨다. - Gradient가 양수(+)이면 weight를 음의 방향으로 이동시킨다. - Gradient가 0이면 weight 갱신을 멈추고 학습을 종료한다.

형성평가

번호	문제	보기	정답	해설
3	아래에서 설명하는 활성화 함수(Activation Function은 무엇인가? - 양수가 입력되면 양수 그대로를 출력 - 음수가 입력되면 0으로 출력 - 음수가 입력되면 해당 노드는 더 이상 학습이 진행되지 않음		ReLU	활성화 함수인 ReLU는 양수가 입력되면 양수 그대로를 출력하고, 음수가 입력되면 0으로 출력 음수가 입력되면 0을 출력하기 때문에 한번 음수가 입력되면 해당 노드는 더 이상 학습이 진행되지 않는다.

학습정리

- 유닛 1

Back Propagation(오차 역전파)는 DNN의 학습 메커니즘으로 Gradient Descent 알고리즘을 이용한 weight 계산 및 업데이트 과정으로 수행된다.

Gradient Descent는 Weight와 손실함수(loss function)의 관계를 이용하여 최적의 weight를 찾는 과정으로 특정 지점 w에서 기울기가 가장 가파르게 하강하는 곳을 따라 learing rate만큼 이동하면서 손실 값 최소 지점으로 이동해 가면서 최적의 weight를 찾는 알고리즘이다.

Gradient는 손실 함수의 weight들에 대한 모든 기울기로 미분(편미분)한 값의 집합(벡터)로 가중치 매개변수의 값을 변화시켰을 때 손실 함수의 변화량을 말한다.

Gradient가 음수(-)이면 weight를 양의 방향으로 이동시키고, Gradient가 양수(+)이면 weight를 음의 방향으로 이동시킨다.

Gradient 0이면 weight 갱신을 멈추고 학습을 종료시킨다.

Learning Rate는 Gradient에 의해 결정된 이동방향에 따라 얼마만큼 이동 크기를 결정하기 위한 Hyper Parameter이다.

- 유닛 2

Optimizer는 최적의 weight를 찾아 업데이트하기 위한 알고리즘으로 Gradient 계산, learning rate 만큼 이동하여 weight 계산, weight 업데이트 등을 수행한다.

여러 Optimizer 중에서 일반적으로 RMSProp이나 Adam Optimizer를 많이 사용한다.

손실 함수(loss function)는 실제 값과 예측 값을 비교하여 차이를 계산하기 위한 함수이다.

평균 제곱 오차(Mean Squared Error, MSE)는 예측 값에서 실제 값을 뺀 차이 값을 제곱한 후 평균을 구하여 loss를 계산한다.

교차 엔트로피(Cross Entropy)는 예측 값에 자연로그(nature log)를 취해서 실제 값과 곱한 후 모두 더해서 loss를 계산한다.

학습정리

- 유닛 3

활성화 함수(Activation Function)는 단일 노드(node)에 여러 신호가 들어오면, 다음 레이어(layer)에 보낼 신호의 강도를 결정하며, NN(Neural Network)에 비선형성(non linearity) 추가하는 역할을 한다.

시그모이드(Sigmoid)는 어떠한 값이 입력되더라도 0 ~ 1 사이의 값 만을 출력하는 활성화 함수로 그레이언트 소실(gradient vanishing) 문제가 발생할 수 있다.

Gradient vashing은 Backpropagation을 수행하기 위해 여러 단계의 layer를 거치게 되면, 소수점 이하의 작은 값을 여러 번 곱하게 됨으로써 gradient가 점점 작아져 0에 가깝게 수렴하게 되어 더 이상 학습이 진행되지 않게 되는 상태를 말한다.

tanh(Hyperbolic tan)는 -1 ~ 1 사이의 값을 출력하는 활성화 함수이다.

ReLU(Rectified Linear Unit)는 양수가 입력되면 양수 그대로를 출력하고, 음수가 입력되면 0으로 출력하는 활성화 함수로 음수가 입력되면 0을 출력하기 때문에 한번 음수가 입력되면 해당 노드는 더 이상 학습이 진행되지 않을 수 있다.

Leaky ReLU(Rectified Linear Unit)는 음수 값이 입력되면 0을 출력하여 학습이 진행되지 않는 ReLU의 문제점을 개선하기 위한 활성화 함수로 음수가 입력되더라도 0으로 출력되지 않고 1보다 작은 값을 곱한 음수 값으로 출력되도록 한다.

ELU(Exponential ReLU)는 음수가 입력되면 Exponential 적용한 음수 값으로 출력하는 활성화 함수로 Exponential 계산 비용이 필요하다.

추가 학습요소