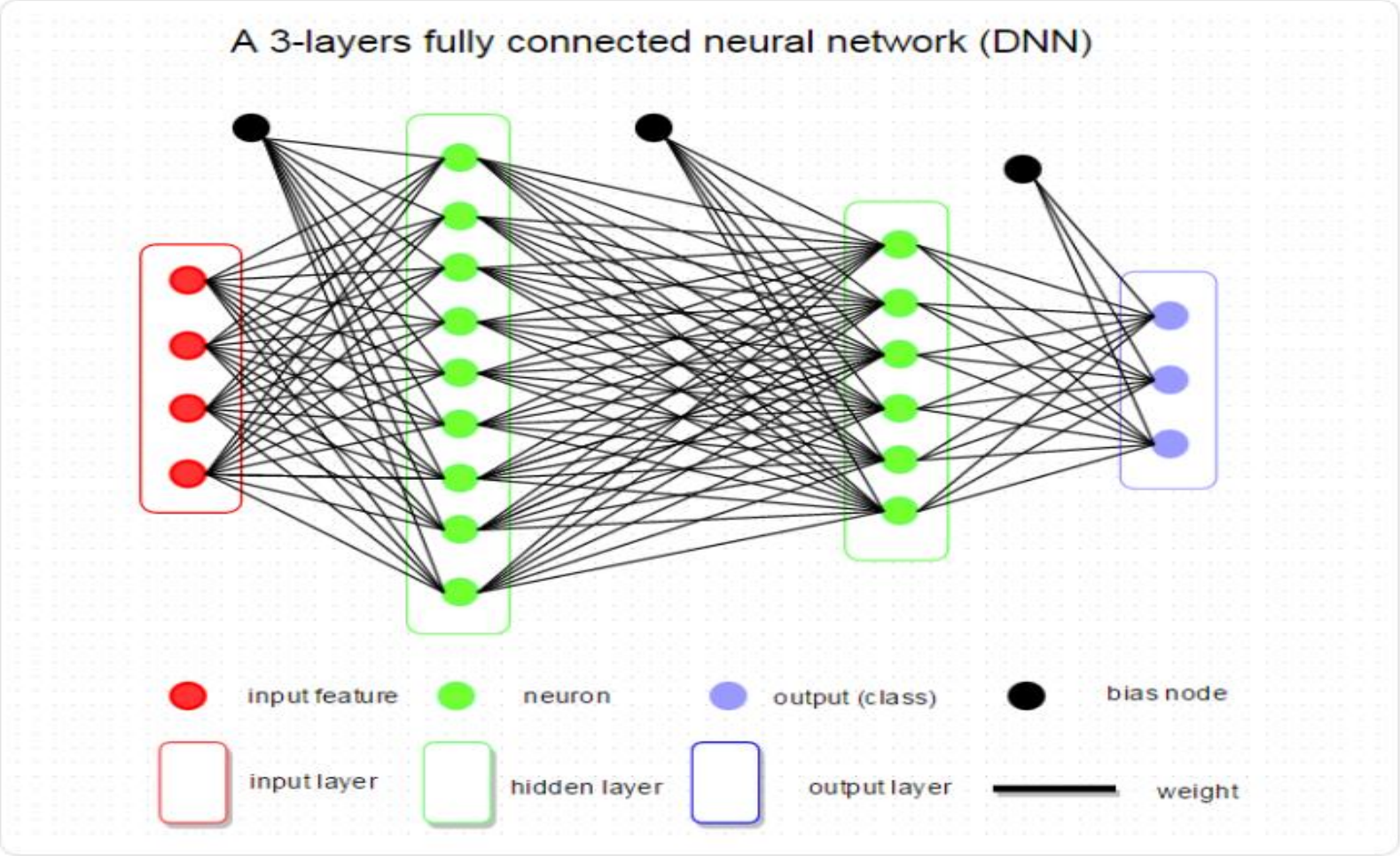
SUN SOFTWARE MAESTRO

Deep Neural Network

Deep Neural Network

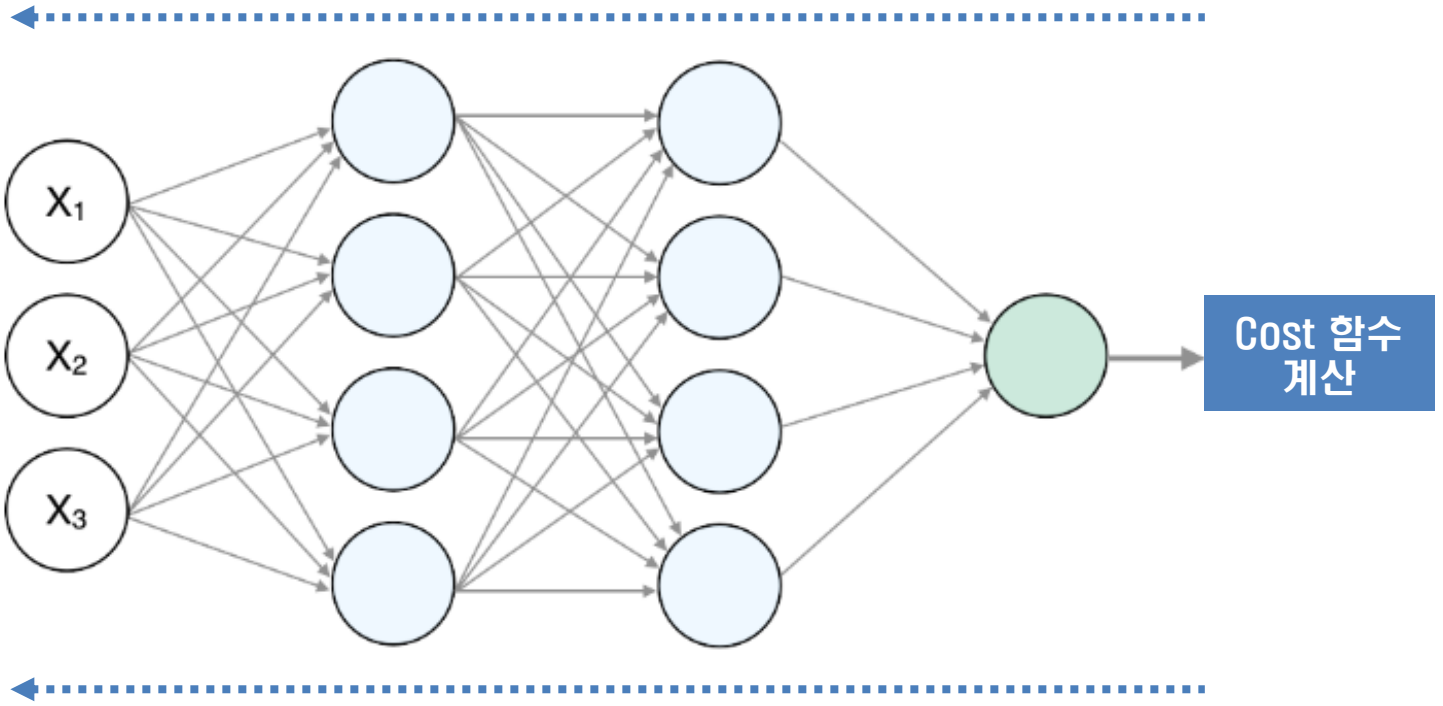


출처 : http://www.parallelr.com/r-deep-neural-network-from-scratch/

Deep Neural Network – Feed Forward & Back Propagation

Feed Forward

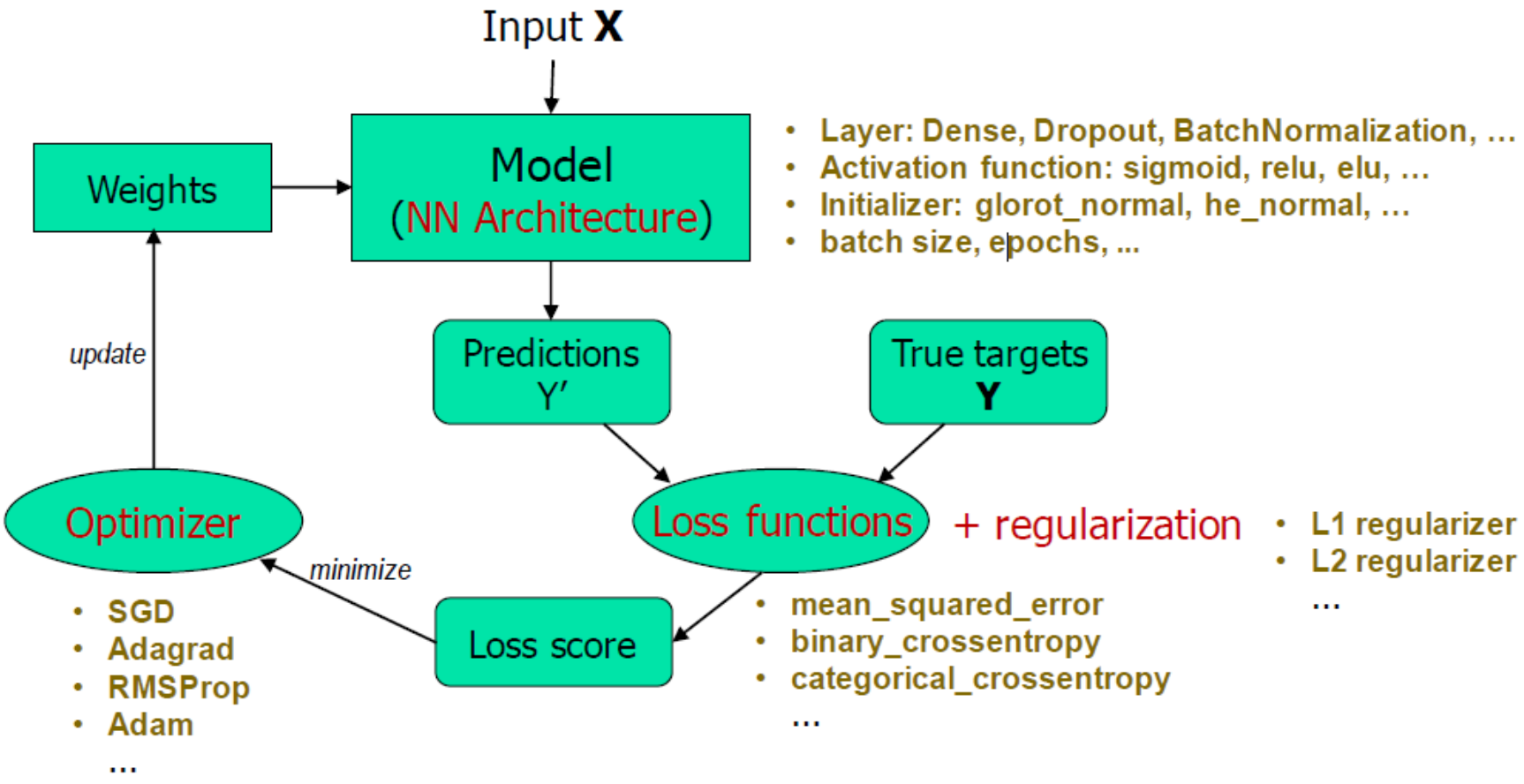
input 값과 weight를 이용한 예측값 계산



Gradient Descent를 이용한 weight 업데이트

Back Propagation

Deep Neural Network – Overview



Deep Neural Network – Cost Function

- 평균 제곱 오차(Mean Squared Error, MSE)

$$E = \frac{1}{2}\sum_{k}(y_k - t_k)^2$$

(y: predict, t: target, k: dimension)

```
hypothesis = x_train * W + b
cost = tf.reduce_mean(tf.square(hypothesis - y_train))
```

Deep Neural Network – Cost Function

- 교차 엔트로피(Cross Entropy)

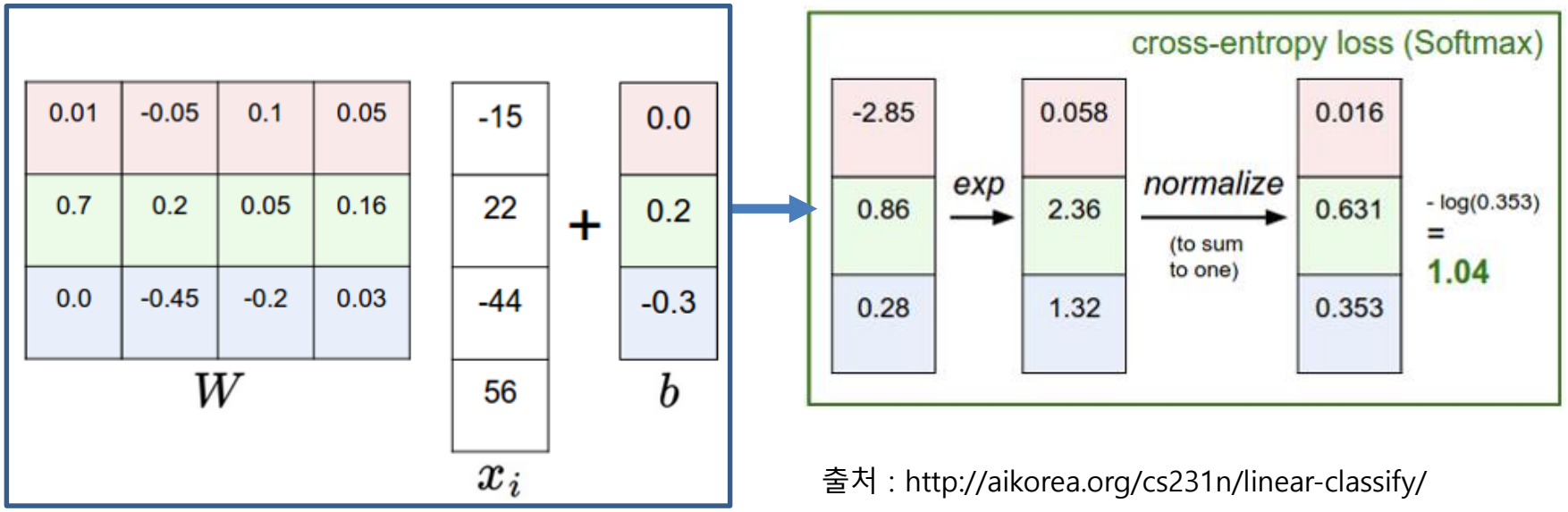
$$E = -\sum_{k} t_k \log y_k$$

(y: predict, t: target, k: dimension, log: natural log)

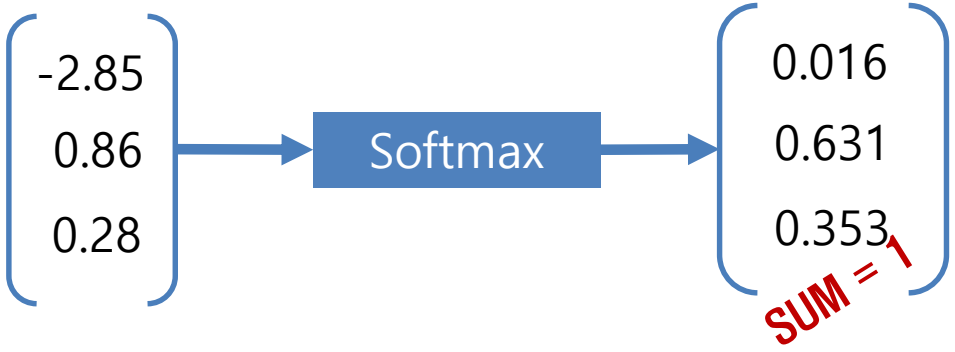
```
hypothesis = tf.matmul(X, W) + b

cost = -tf.reduce_sum(Y * tf.log(hypothesis))
```

Deep Neural Network – Softmax

Cross entropy & Softmax



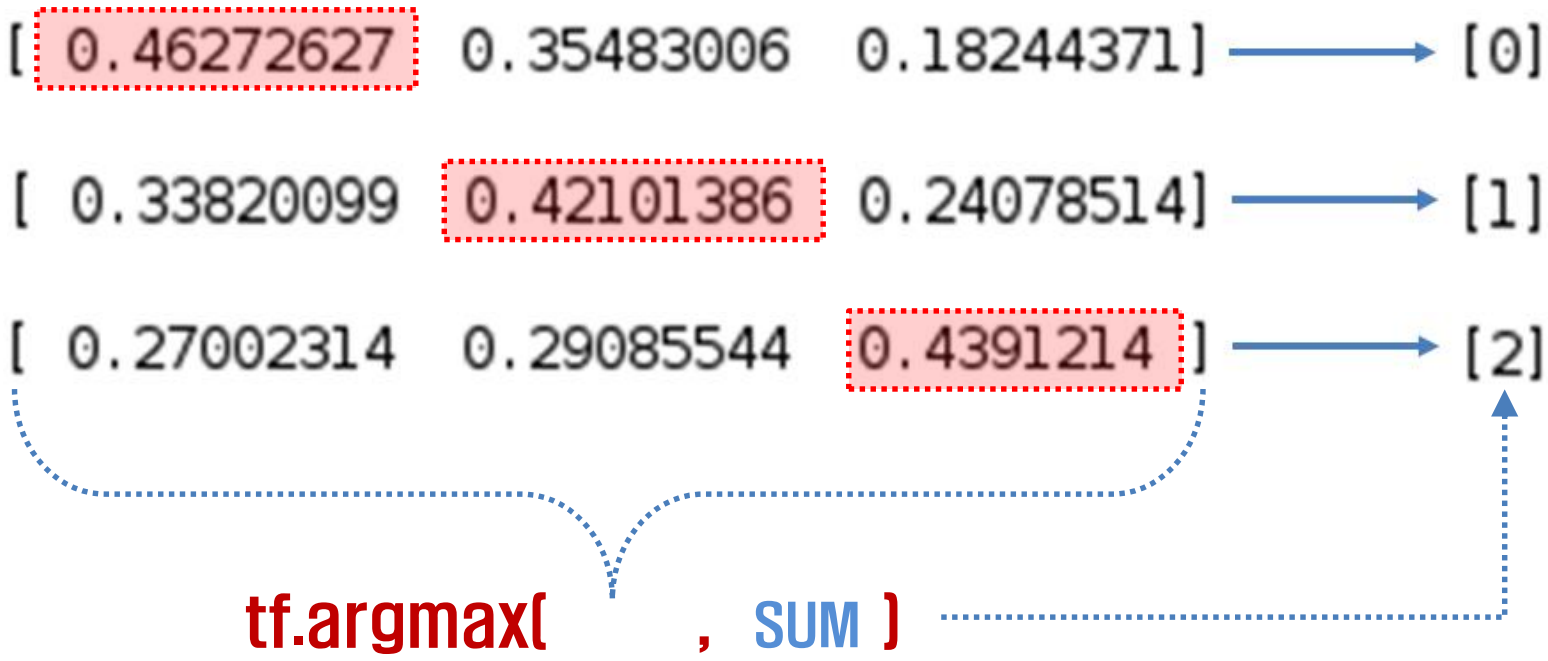
출처 : http://aikorea.org/cs231n/linear-classify/



Deep Neural Network – Softmax

- 소프트맥스(Softmax)

  소프트맥스 함수의 출력은 0에서 1사이의 실수이며,
  출력의 총합은 1 (소프트맥스 함수의 출력을 확률로 해석 가능)



Deep Neural Network – Softmax

- Cross entropy & Softmax

Computes softmax cross entropy between logits and labels. (deprecated)

```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
        logits=hypothesis, labels=Y))
```

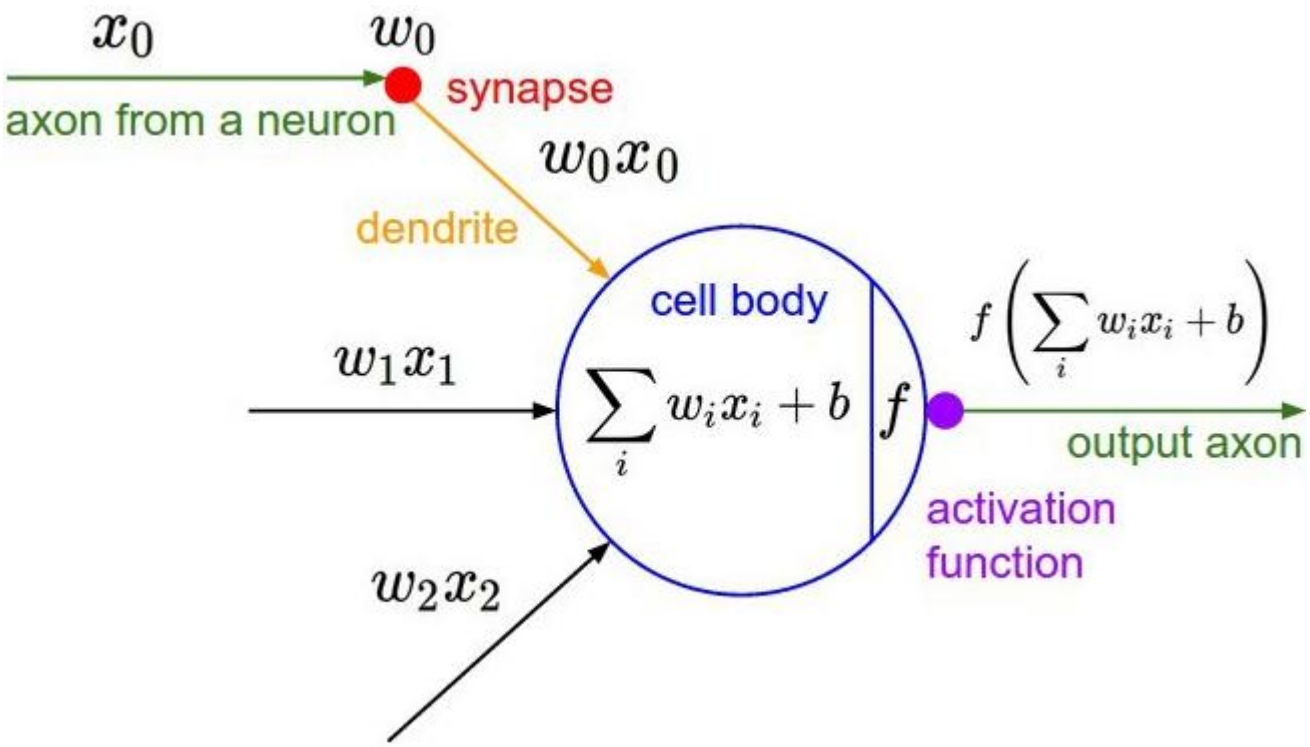
```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(
        logits=hypothesis, labels=Y))
```

실습 : DNN_softmax.py

Deep Neural Network – Activation Function

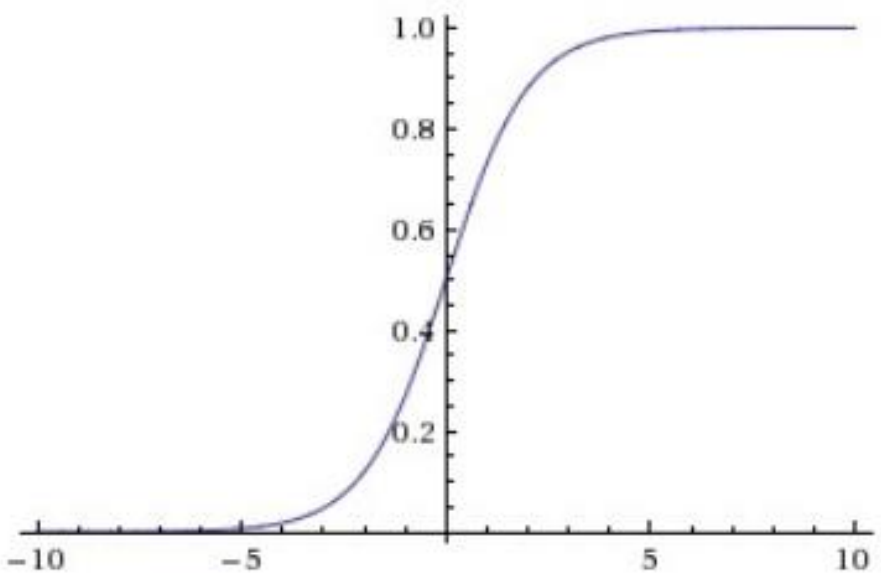
- Activation Function

  네트워크에 **비선형성(nonlinearity)**을 추가하기 위해 사용



Deep Neural Network – Activation Function

- 시그모이드(sigmoid)



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

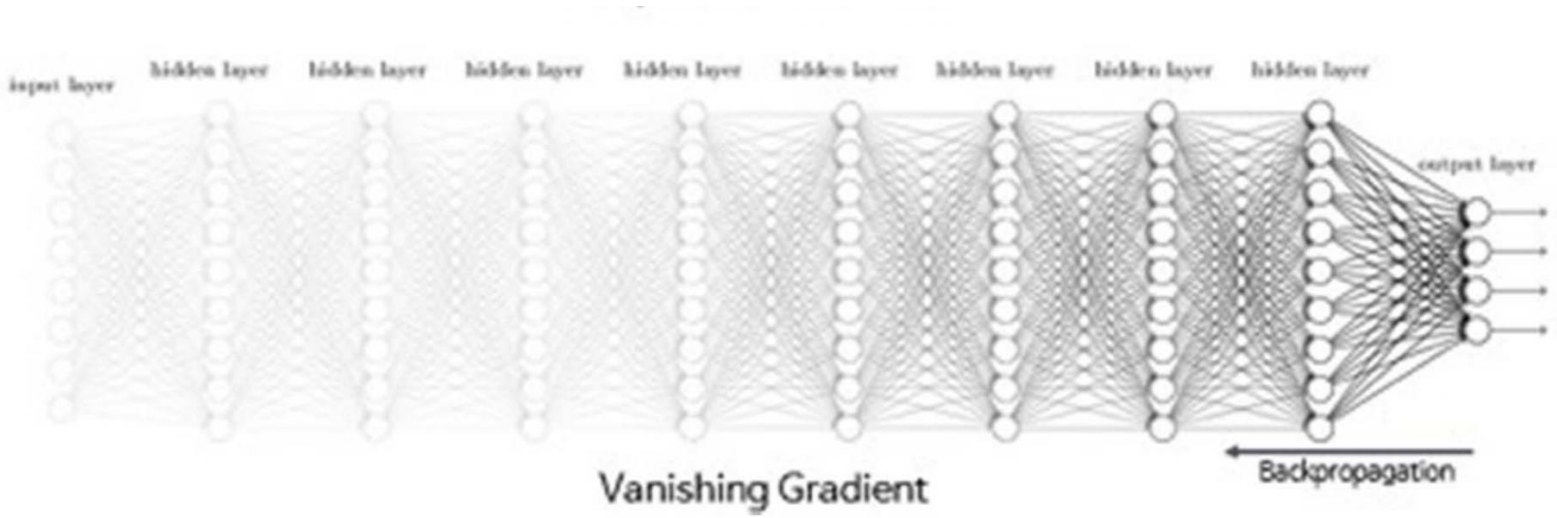
출력값이 0 ~ 1 사이의 값

Gradient vanishing 문제 발생

Deep Neural Network – Activation Function

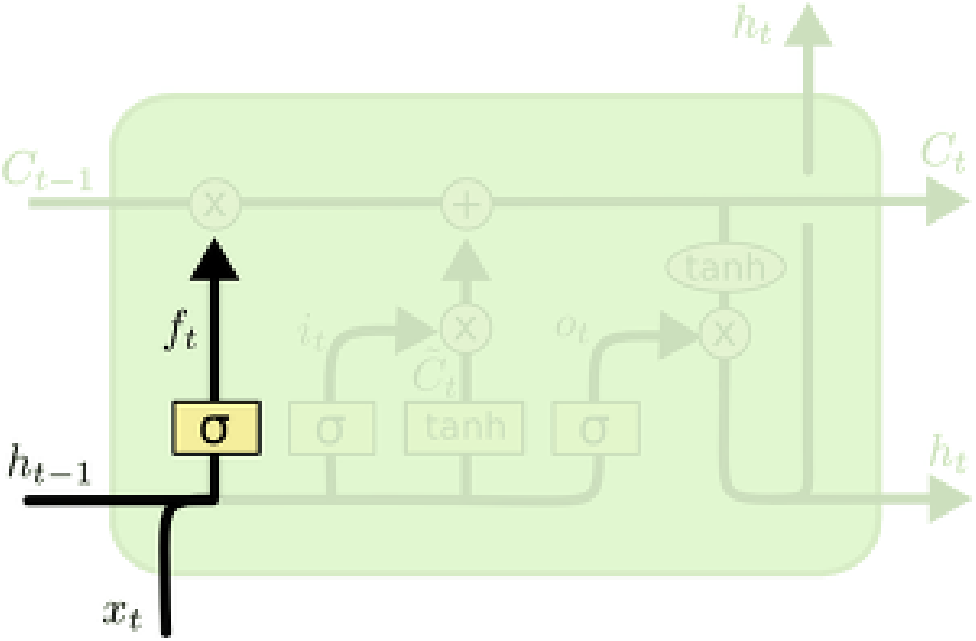
- Gradient vashing

  Backpropagation을 수행하기 위해 여러 단계의 layer를 거치게 되면,
  소수점 이하의 작은 gradient를 여러 번 곱하게 됨
  gradient가 점점 작아져 0에 가깝게 수렴하게 되면서 더 이상 학습이 진행
  되지 않는 상태가 됨



Deep Neural Network – Activation Function

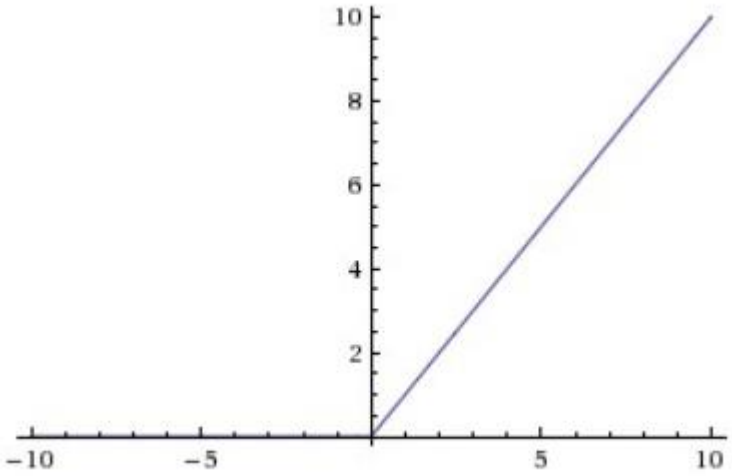
- 시그모이드(sigmoid) 사용 예



cell state 정보 중 어떤 정보를 유지하고 삭제(forget)할 지 결정
1에 가까울 수록 많은 정보 반영, 0에 가까울 수록 적은 정보의 반영

Deep Neural Network – Activation Function

- ReLU, Rectified Linear Unit



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

Gradient vanishing 문제 해소 가능

음수가 입력되면 0을 출력하기 때문에
한번 음수가 입력되면 해당 노드는 더 이상 학습이 진행되지 않음

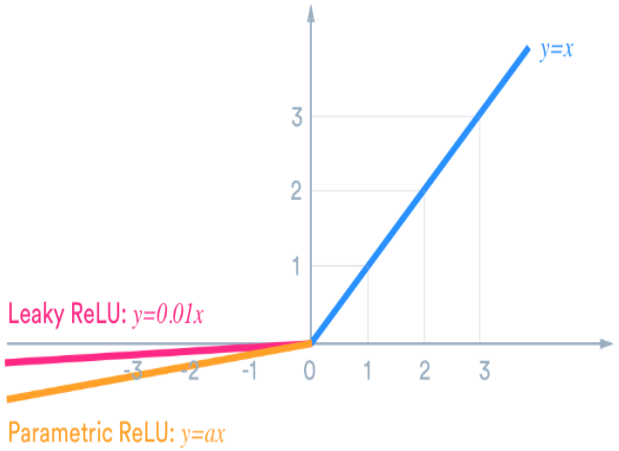
실습 : DNN_deep_hidden_layer3.py

실습 : DNN_deep_hidden_layer5.py

Deep Neural Network – Activation Function

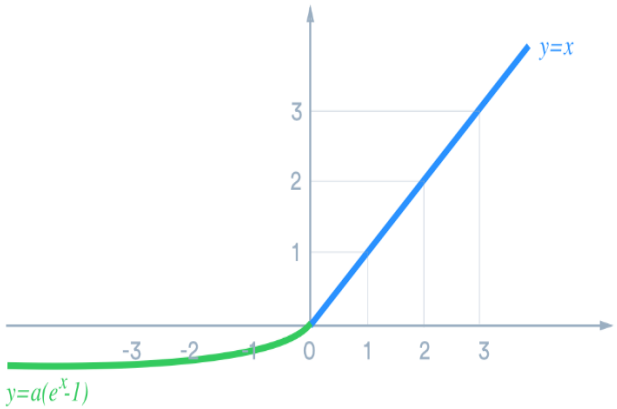
- Leaky ReLU & Exponential LU & tanh

Leaky ReLU



tf.nn.leaky_relu()

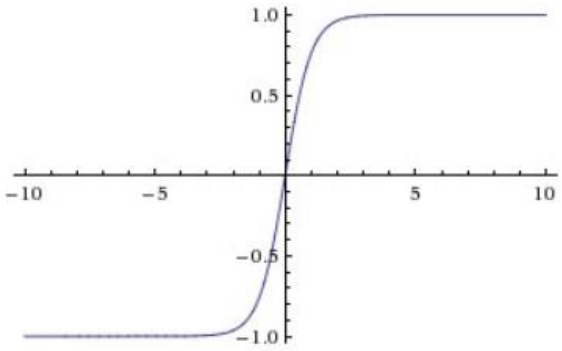
Exponential LU



tf.nn.elu

exp() 계산 비용 소요

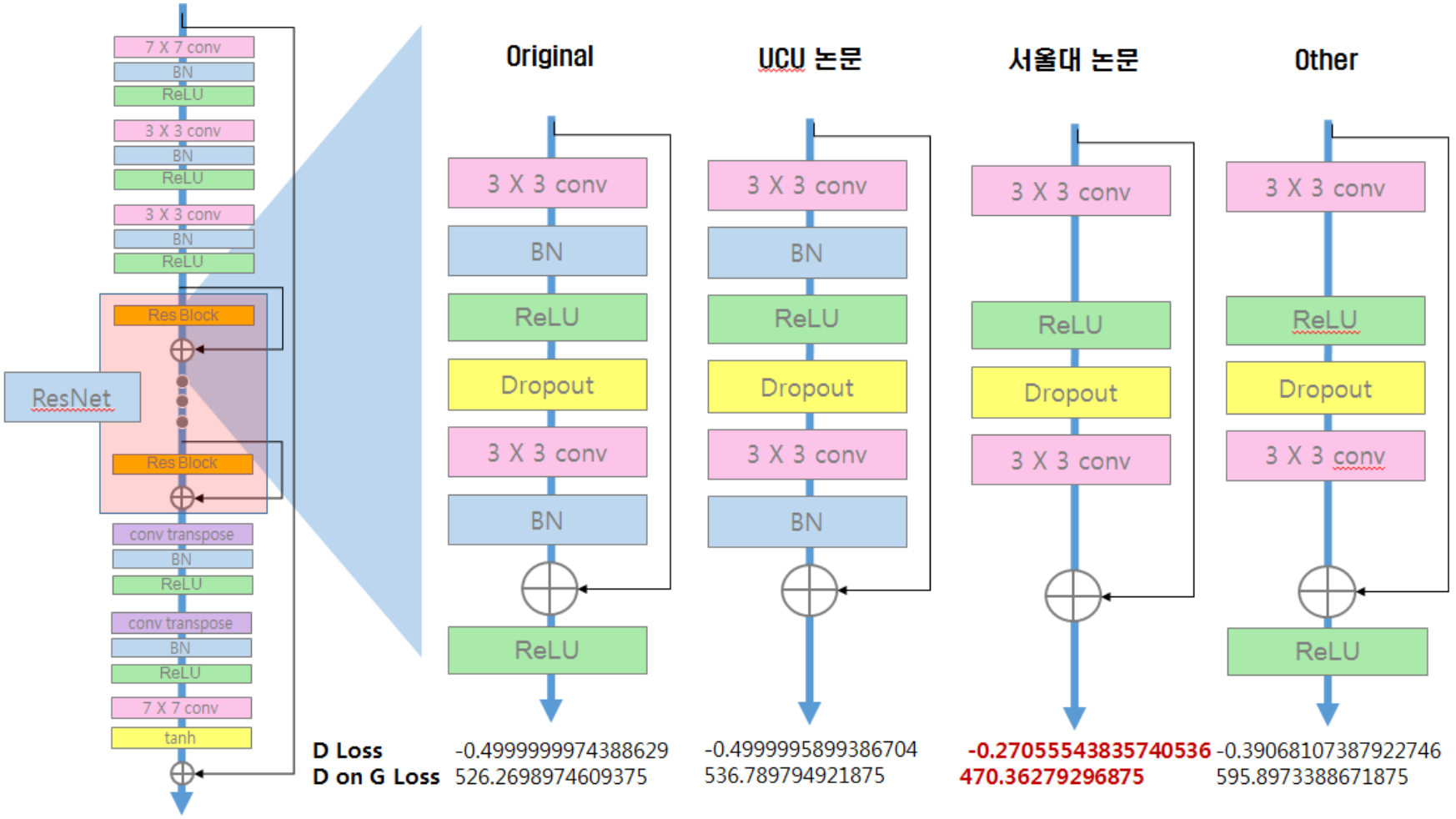
tanh



tf.nn.tanh()

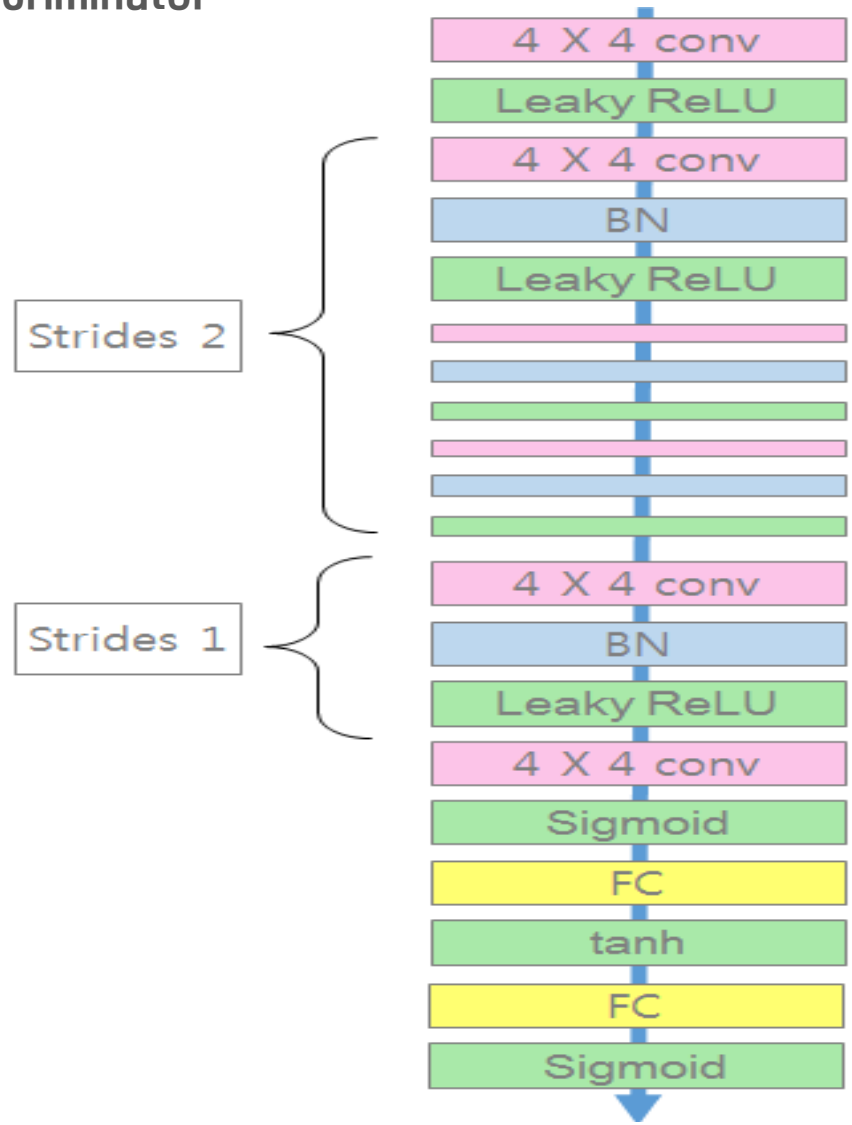
Deep Neural Network – Activation Function 예시

GAN-Generator

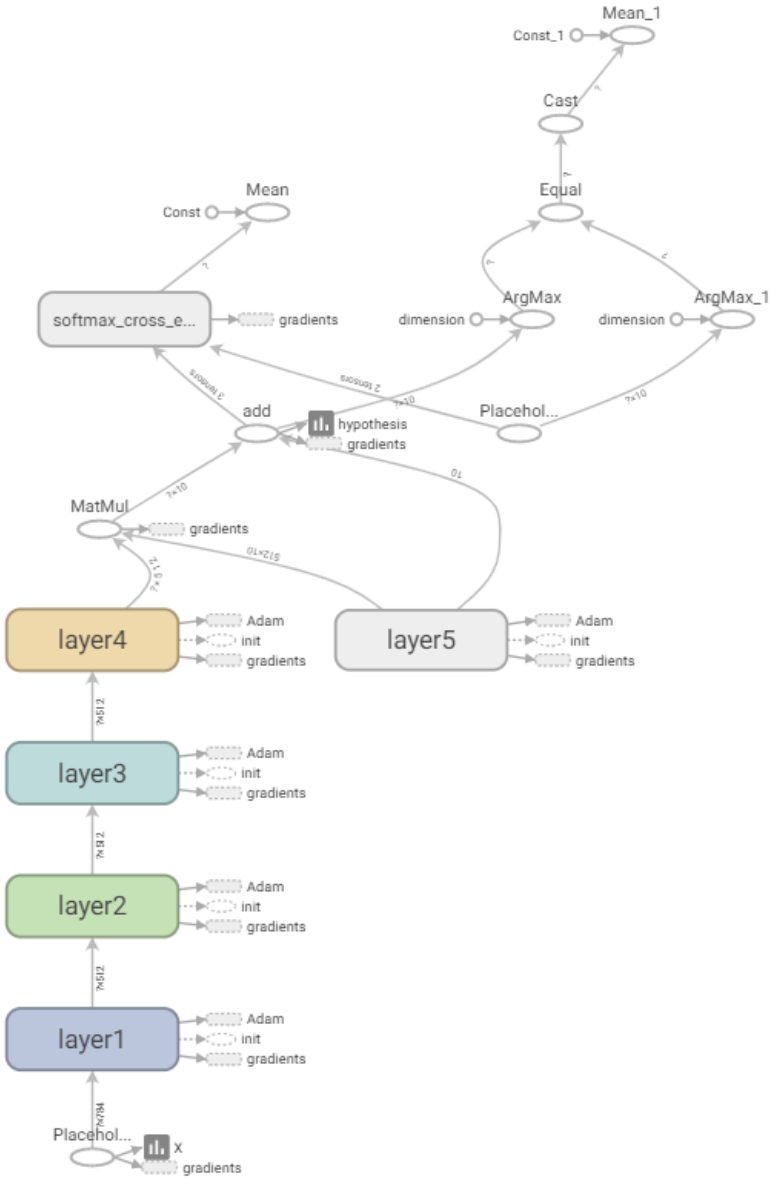


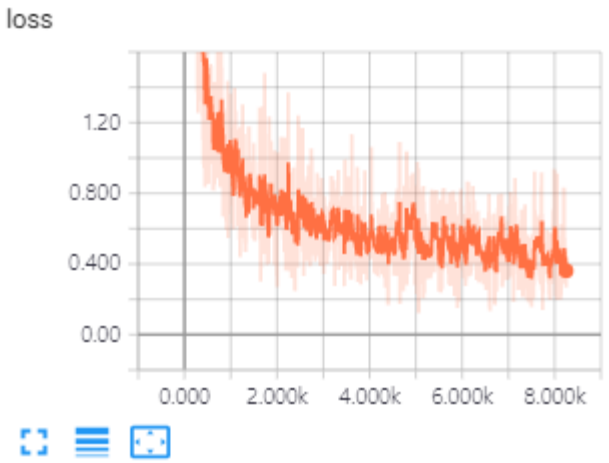
Deep Neural Network – Activation Function 예시

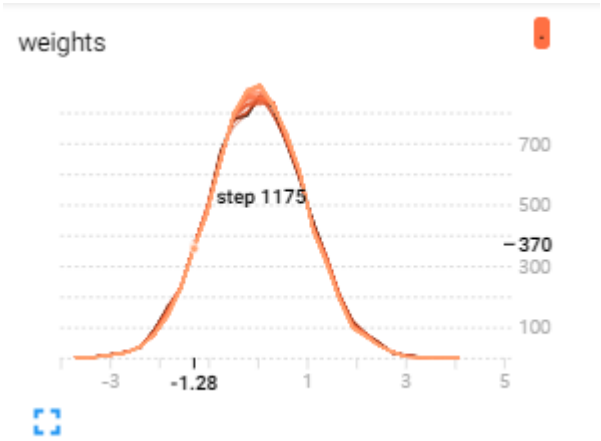
- GAN-Discriminator



Deep Neural Network – Activation Function





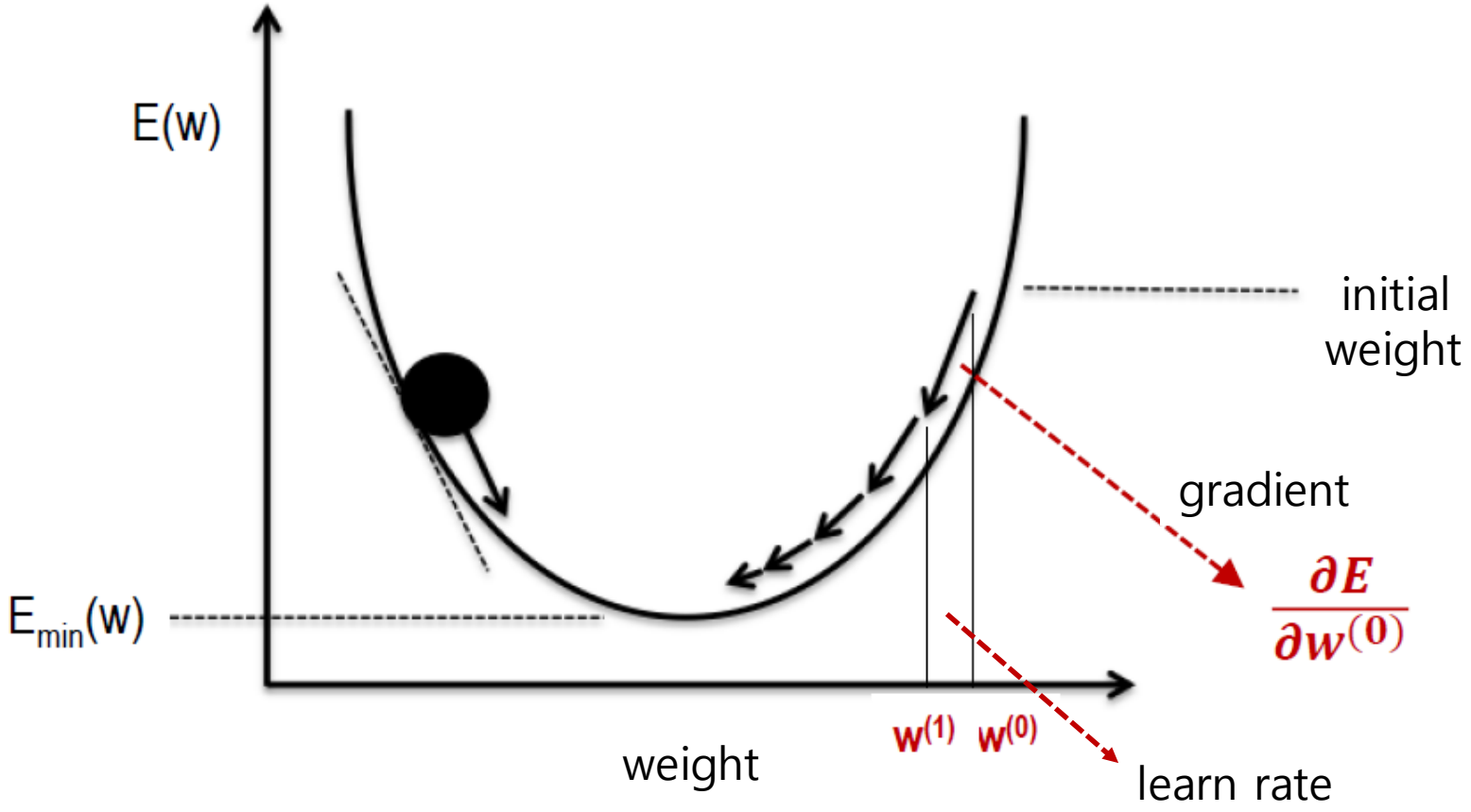


tensorboard --logdir=d:/temp/tensorboardlogs

실습 : DNN_deep_hidden_layer5_tensorboard.py

Deep Neural Network – Gradient Descent

- **학습** : 손실 함수 E(w)가 최소가 되는 weight W를 찾는 과정



현재 지점의 w에서 기울기가 가장 가파르게 하강하는 곳을 따라 learing rate만큼 이동

Deep Neural Network – Gradient Descent

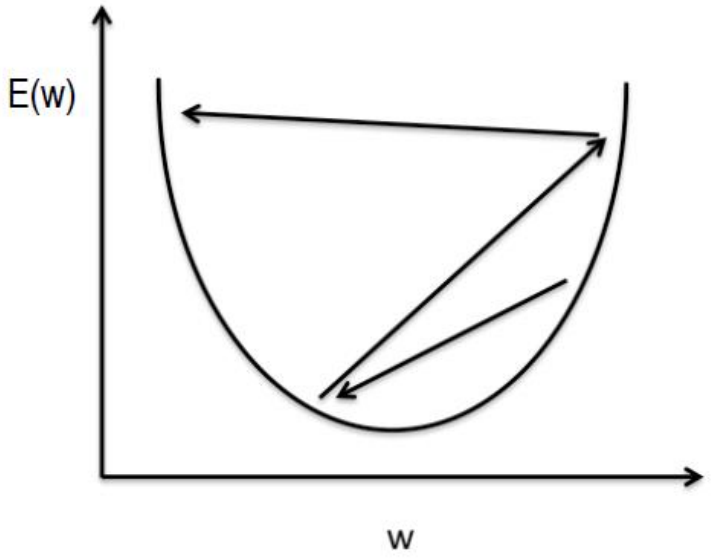
- Gradient : 손실 함수 E(w)의 weight 대한 모든 편미분 값의 벡터
  가중치 매개변수의 값을 변화시켰을 때 손실 함수의 변화량

- Gradient가 음수이면 weight를 양의 방향으로
- Gradient가 양수이면 weight를 음의 방향으로
- Gradient 0이면 weight 갱신 종료(학습 종료)

- 최적화(Optimization) : 최적의 weight를 찾기 위한 해법

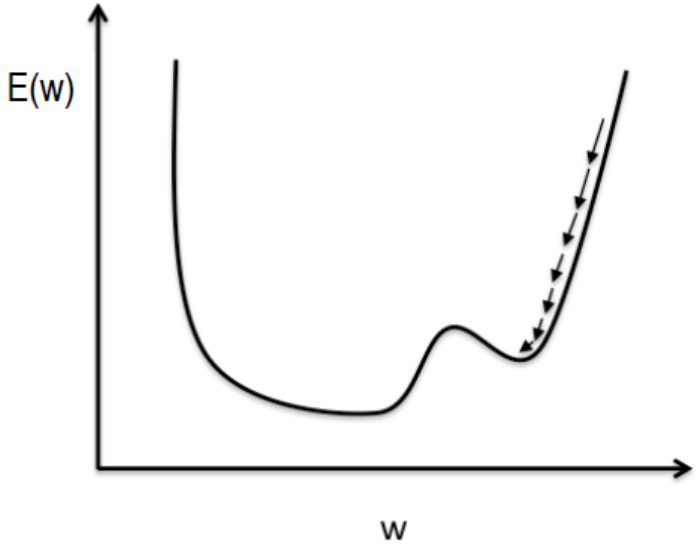
Deep Neural Network – Learning Rate

learning rate 가 너무 큰 경우



Overshooting 발생

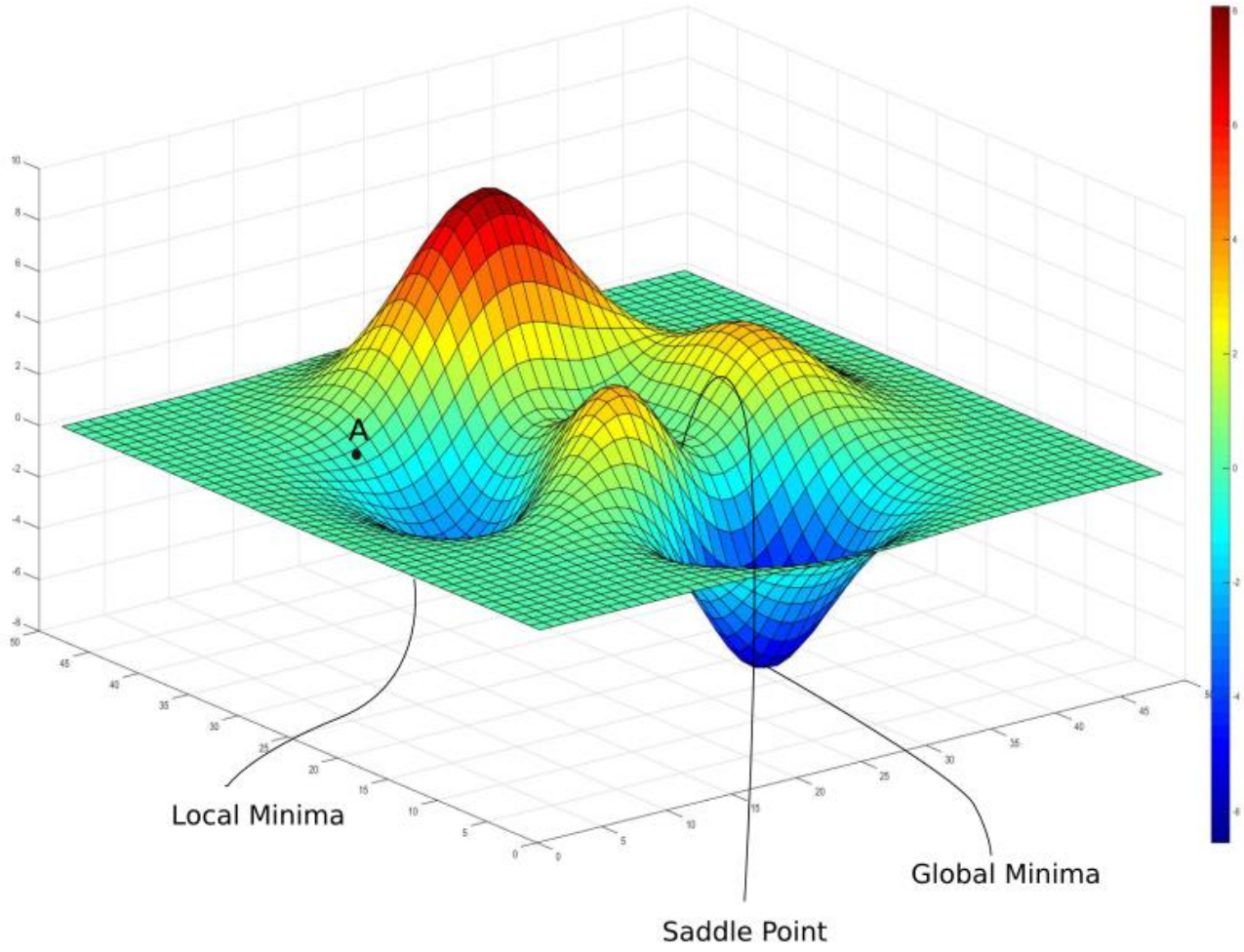
learning rate 가 너무 작은 경우



학습속도 저하, local minima

learing rate는 일반적으로 0.01 ~ 0.001 사이의 값을 가장 많이 사용

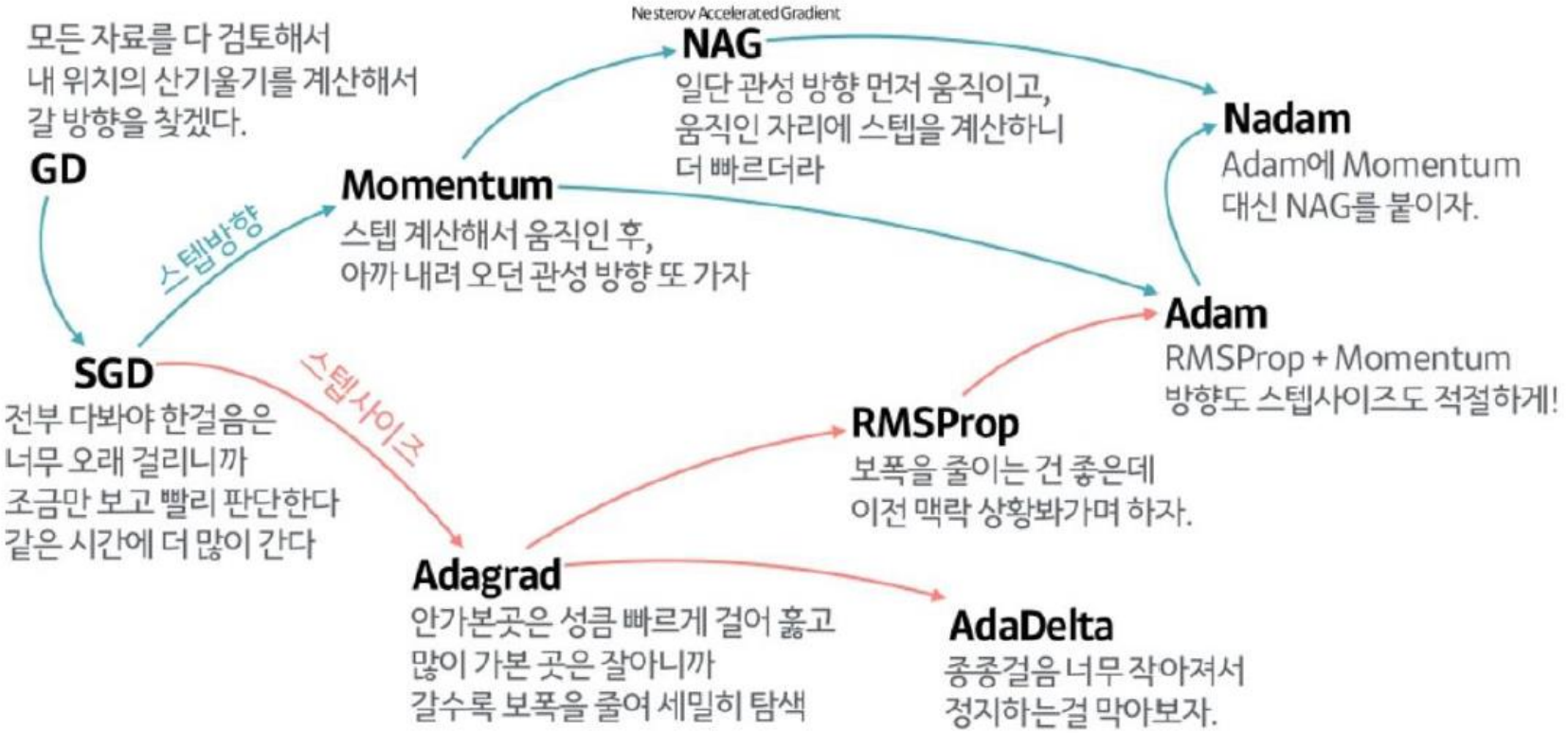
Deep Neural Network – Gradient Descent



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

Deep Neural Network – Optimizer

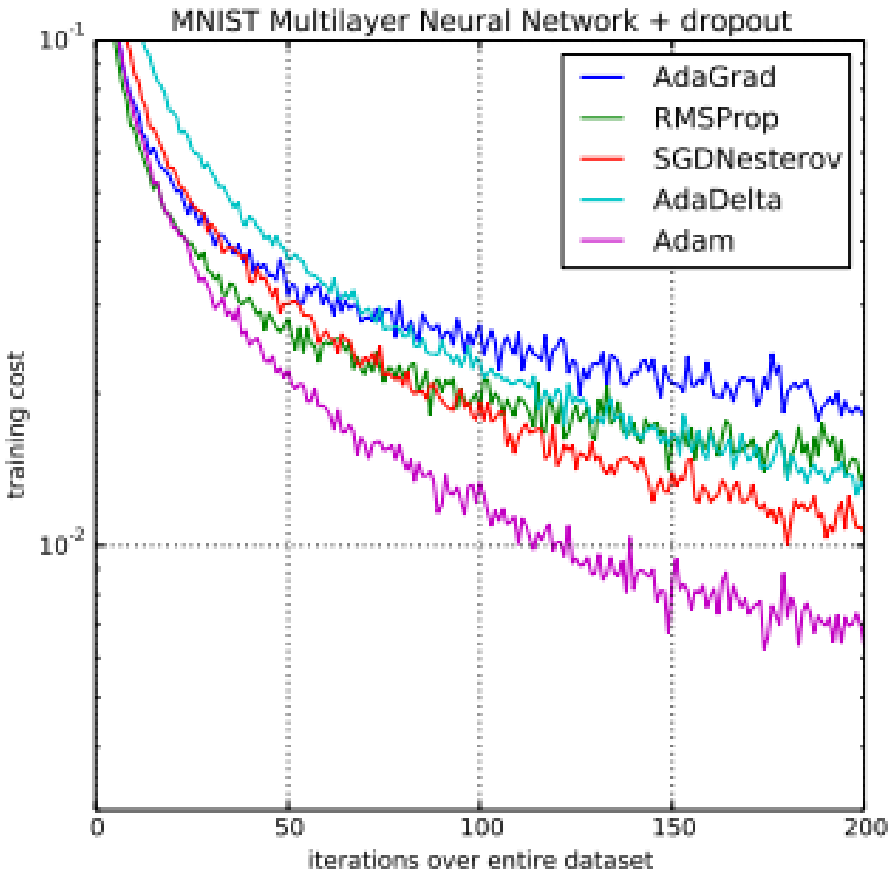
- Optimizer : 최적의 weight를 찾아 업데이트하기 위한 알고리즘



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

Deep Neural Network – Optimizer

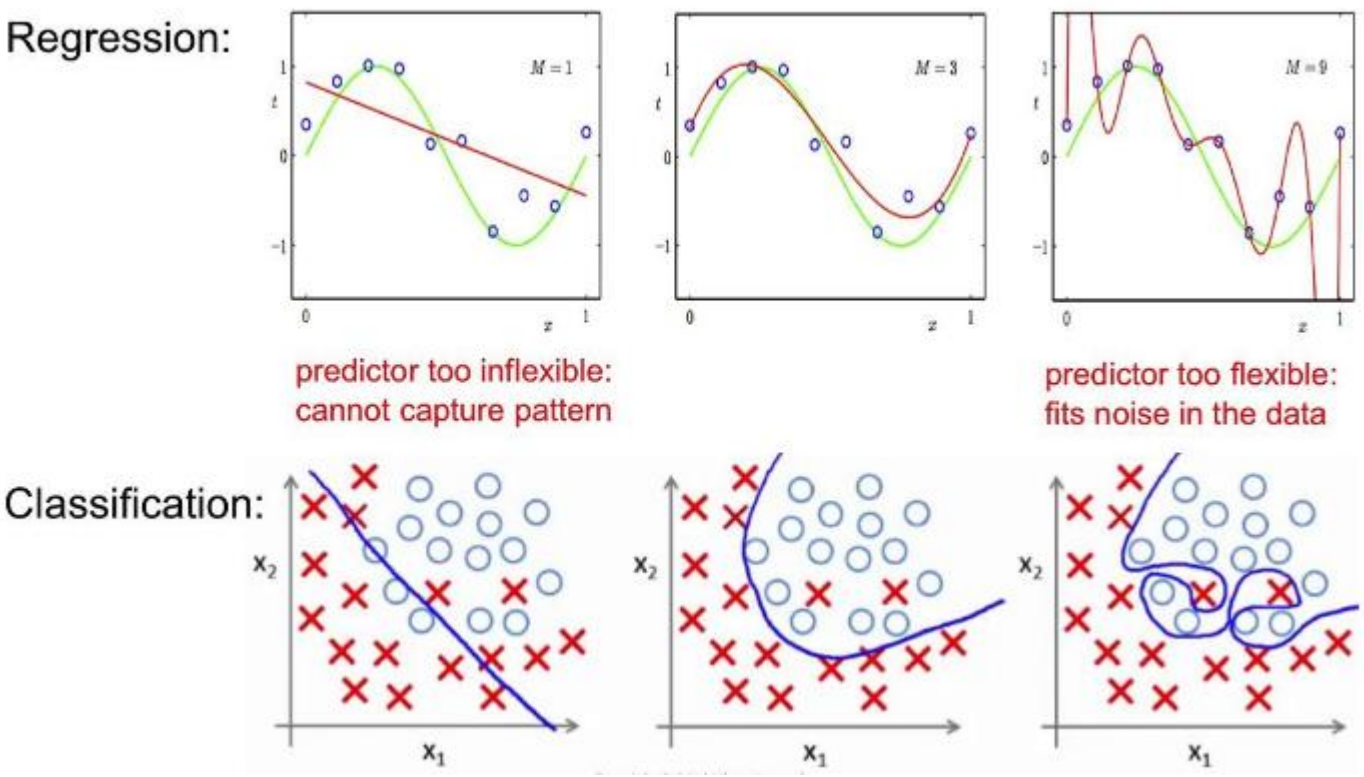
- Optimizer 학습 속도 비교



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

Deep Neural Network – Overfitting

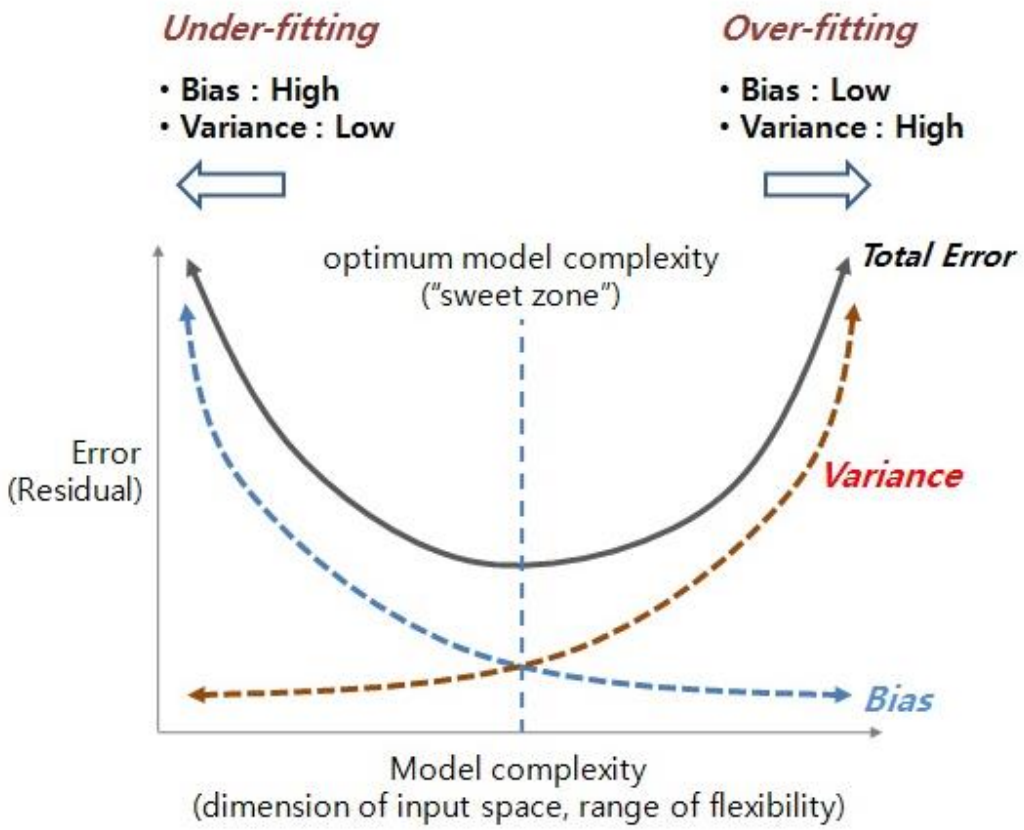
- Overfitting과 Underfitting



출처 : https://brunch.co.kr/@gimmesilver/44

Deep Neural Network – Overfitting

- Overfitting과 Underfitting



출처 : https://rfriend.tistory.com/tag/bias-variance%20tradeoff

Deep Neural Network – initial weight

- Xavier 초기값

  각 층의 활성화 값들을 광범위하게 분포시킬 목적으로 가중치의 적절한 분포를 가지기 위한 초기값에 대한 연구 논문인 사비에르 글로로트(Xavier Glorot)와 오슈아 벤지오(Yoshua Bengio)의 논문

  Weight의 초기값으로 선행 계층의뉴런이 n개라면 초기 값의 표준편차가 **sqrt(1/n)인 정규분포**를 사용

  tf.contrib.layers.xavier_initializer()

- He 초기값

  ReLU함수에서 사용하는 초기값으로 카이밍 히(Kaiming He)가 발견

  초기값의 표준편차가 **sqrt(2/n)인 정규분포**를 사용

  tf.contrib.layers.variance_scaling_initializer()

  tf.variance_scaling_initializer()

실습 : DNN_initializer.py

Deep Neural Network – weight decay

- Weight Decay : 학습과정에서 가중치가 클 경우 패널티를 부과하여 overfitting 억제

- L2 정규화(regularization)

  모든 가중치 각각의 손실 함수에 $1/2\lambda W^2$ 를 더하면 가중치는 $1/2\lambda W^2$ 만큼 감소 효과

  람다(λ)는 정규화의 강도를 조절하는 hyper parameter

  람다(λ)를 크게 설정할수록 큰 가중치에 대한 패널티가 커짐

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2$$

- L1 정규화(regularization)

  W^2 대신 W의 절대값 사용

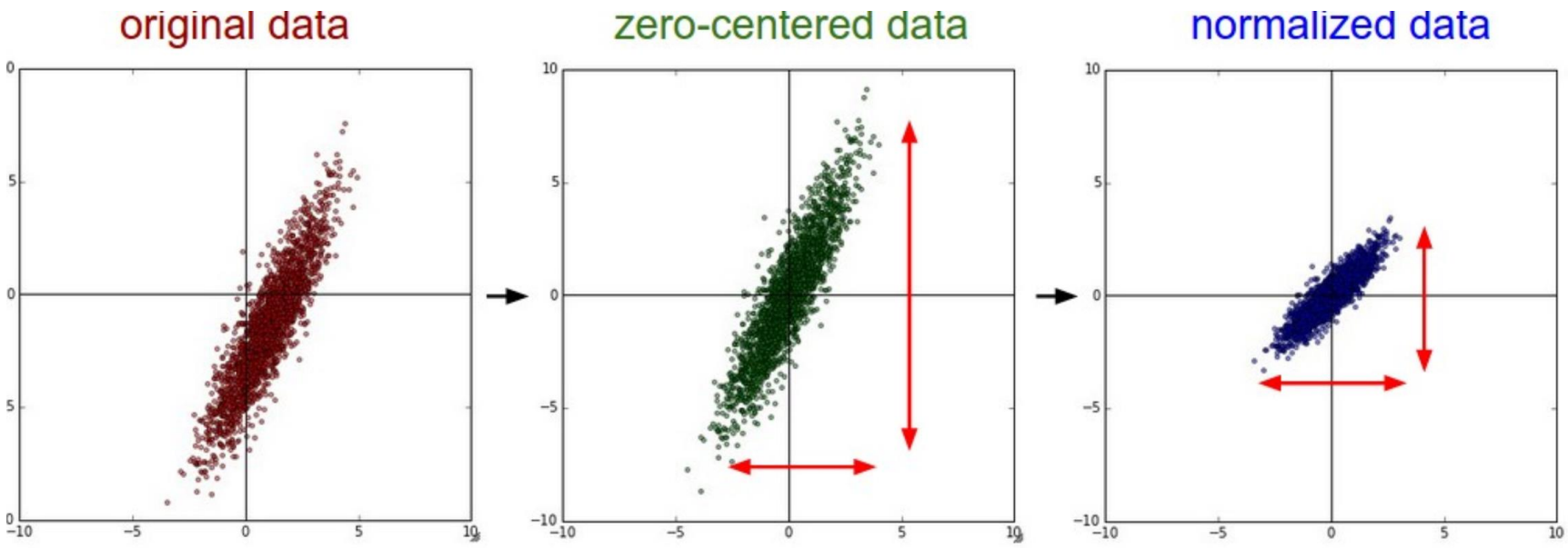
실습 : DNN_regularization.py

Deep Neural Network – Batch Normalizaion

- 정규화(Normalization)

분포가 **평균 0, 분산 1**

```
X -= np.mean(X)
X /= np.std(X, axis = 0)
```

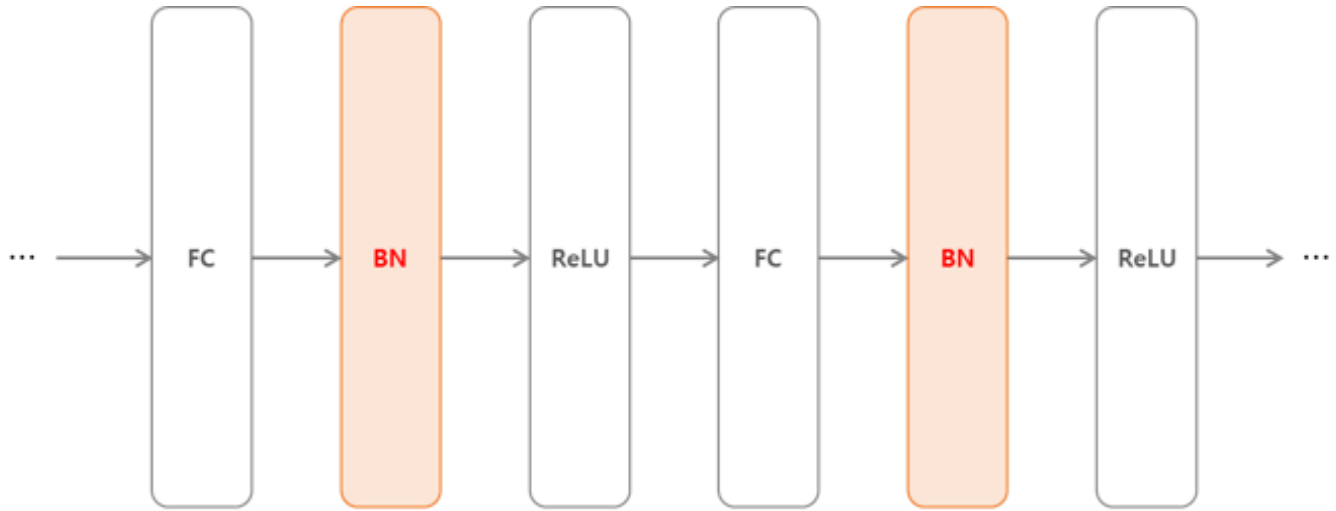


출처 : http://aikorea.org/cs231n/neural-networks-2-kr/

Deep Neural Network – Batch Normalizaion

- 배치 정규화(Batch Normalization)

  각 Layer의 출력값의 분포가 **평균 0, 분산 1**이 되도록 mini batch단위로 정규화를 통해 학습 속도 개선, overfitting 억제



[BN을 사용한 신경망 예]

출처 : https://excelsior-cjh.tistory.com/178

실습 : DNN_batch_normalization.py

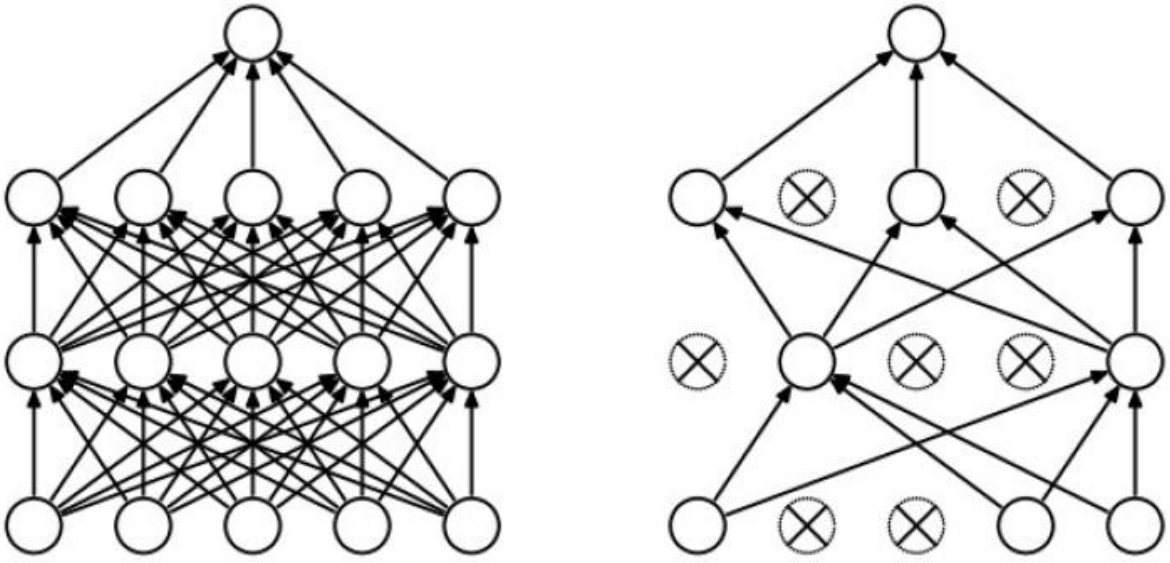
Deep Neural Network – dropout

- dropout

신경망 모델이 복잡해지면 weight decay만으로 overfitting 해결이 어려움

학습과정에서 은닉층의 뉴런을 random하게 제거하거나

연결 가중치가 높은 링크를 확률적으로 제거



(a) Standard Neural Net

(b) After applying dropout.

출처 : https://medium.com/@amarbudhiraja

실습 : DNN_batch_normalization.py

Deep Neural Network – hyperparameter

- 은닉층(hidden layer)과 은익 노드(hidden node)의 개수
- 활성화 함수(Activation Function)
- Regularization 기법
- 최적화 기법(Optimizer)
- 가중치 초기화 방식
- learning rate와 mini-batchsize

Deep Neural Network – hyperparameter

- Manual Search

  설계자의 직관이나 경험에 기반, 임의 값을 대입하여 결과를 확인한 후 탐색 알고리즘을 적용하여 그 결과가 움직이는 방향으로 추정해 보고, 이런 과정을 반복하여 최적의 결과를 도출, 탐색 알고리즘에 따라 결과 차이

- Grid Search

  선험적인 지식을 활용하여 문제를 분석, hyperparameter의 범위를 선정, 범위 안에서 일정한 간격으로 값을 설정, 그 값들을 차례로 실험하여 최적의 값을 찾은 후 best로 추정이 되는 점을 기준으로 세분화하여 최적값을 찾는 방법, 결과를 판정하기 위한 validation set이 필요

Deep Neural Network – hyperparameter

- Random search

  Grid search와 마찬가지로 선험적인 지식을 이용하여 hyperparameter의 범위를 정함, 범위내에서 무작위로 최적값을 찾는 작업을 진행, 정해진 시간 안에 결과를 내야 하는 경우, Random search를 할 때 더 좋은 결과 도출 가능

- Bayesian optimization

  실험 결과를 바탕으로 통계적인 모델을 만들고, 그것을 바탕으로 다음 탐색을 해야 할 방향을 효과적으로 정함, Bayesian optimization 방법을 사용하면 Random search나 Grid search를 사용하는 것에 비해 좀 더 짧은 시간에 최적 값을 찾아내는 경향을 보임