ChatGPT와 함께하는 딥러닝 5일 완성

딥러닝 핵심 개념과 Fashion-MNIST 분류 실습

2일차

서울대학교 평생교육원
Extension College Seoul National University

Topics

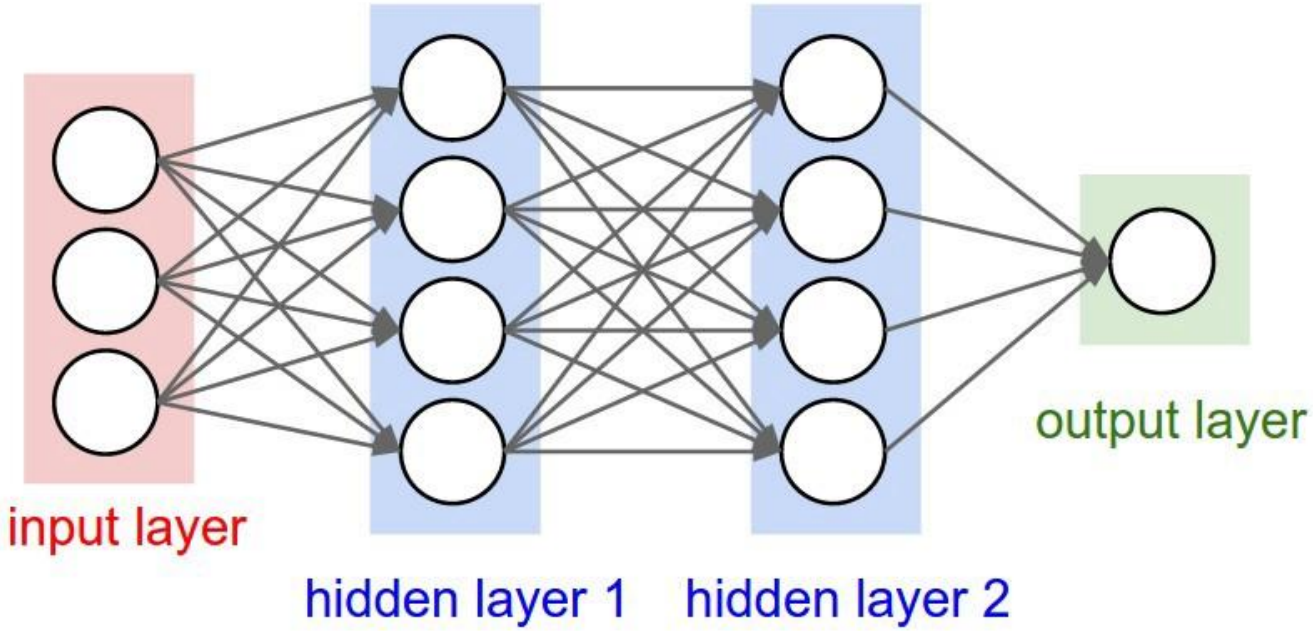
- 활성화 함수, 손실 함수, 옵티마이저 이해
- 드롭아웃, 학습률 등 성능 조절
- 텐서보드로 학습 과정 확인, 체크포인트 저장 설정
- Fashion-MNIST 이미지 분류 실습
- ChatGPT로 코드 개선 연습

Components of Multilayer Perceptron 다층 신경망의 구성요소

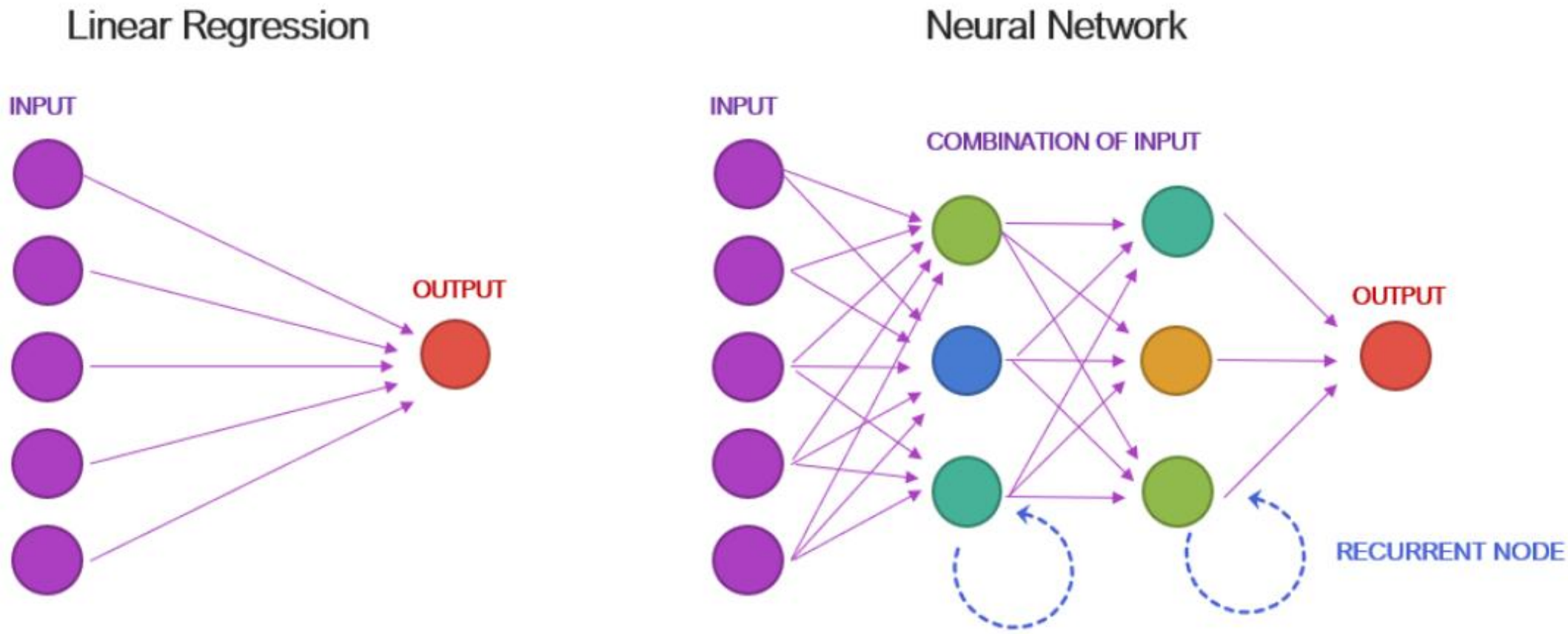
- An input layer, x 입력층
- An arbitrary amount of hidden layers 은닉층
- An output layer, $\hat{y}$ 출력층

입력 레이어는 일반적으로 카운팅하는 데서 빠짐

The first layer of your data is the input layer. Then, all the layers between the input layer and the output layer are the hidden layers. **첫번째 레이어는 입력층, 입력층과 출력층 사이의 모든 레이어는 은닉층이라고 함**

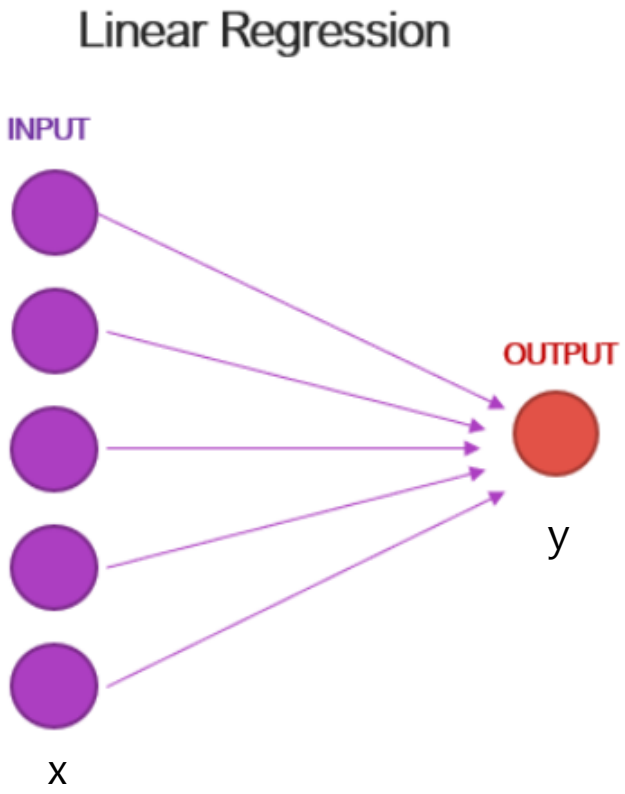


Multilayer Perceptron 다층 신경망



대출예측문제를 위한 회귀분석

대출여부를 y로 놓고, 나머지를 변수를 x1, x2, x3… 로 설정



독립변수 종속변수

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
2	LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
5	LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360	1	Urban	Y
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
7	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
8	LP001013	Male	Yes	0	Not Graduate	No	2333	1516	95	360	1	Urban	Y
9	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N
10	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y
11	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurban	N
12	LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
13	LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
14	LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y
15	LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N
16	LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y
17	LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y
18	LP001034	Male	No	1	Not Graduate	No	3596	0	100	240		Urban	Y
19	LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N
20	LP001038	Male	Yes	0	Not Graduate	No	4887	0	133	360	1	Rural	N

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_p x_p$$

대출예측모델을 위한 예측모델

복잡한 계산을 거쳐서 모든 데이터를 가장 잘 표현할수 있는 예측식의 기울기와 y절편을 구함

X값이 하나인 경우 기울기와 y절편을 구하는 공식

기울기

$$\beta_1 = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{m}(x_i - \bar{x})^2}$$

y절편

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

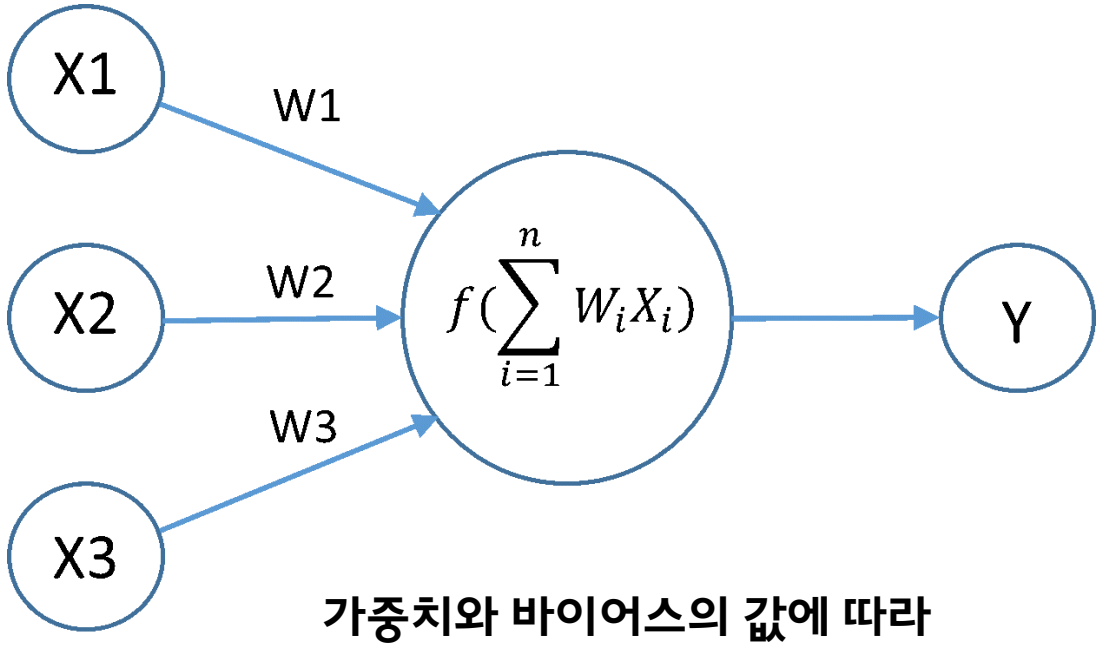
X값이 11개인 경우 구한 기울기와 y절편을 이용해서 만든 예측식

Y = 0.2409165928576860 2 - 0.029044 x1 + 0.107329 x2 + 0.007897 x3 + 0.076671 x4 + 0.018043 x5 - 0.000002 x6 - 0.000010 x7 -0.000093 x8 - 0.000105 x9 + 0.435916 x10 + 0.069534 x11

새로운 대출지원자가 있을 때 이 모델을 이용하여 그 사람의 정보를 가지고 대출을 해줄지 말지를 결정할 수 있음

Weighs and Bias 가중치와 바이어스

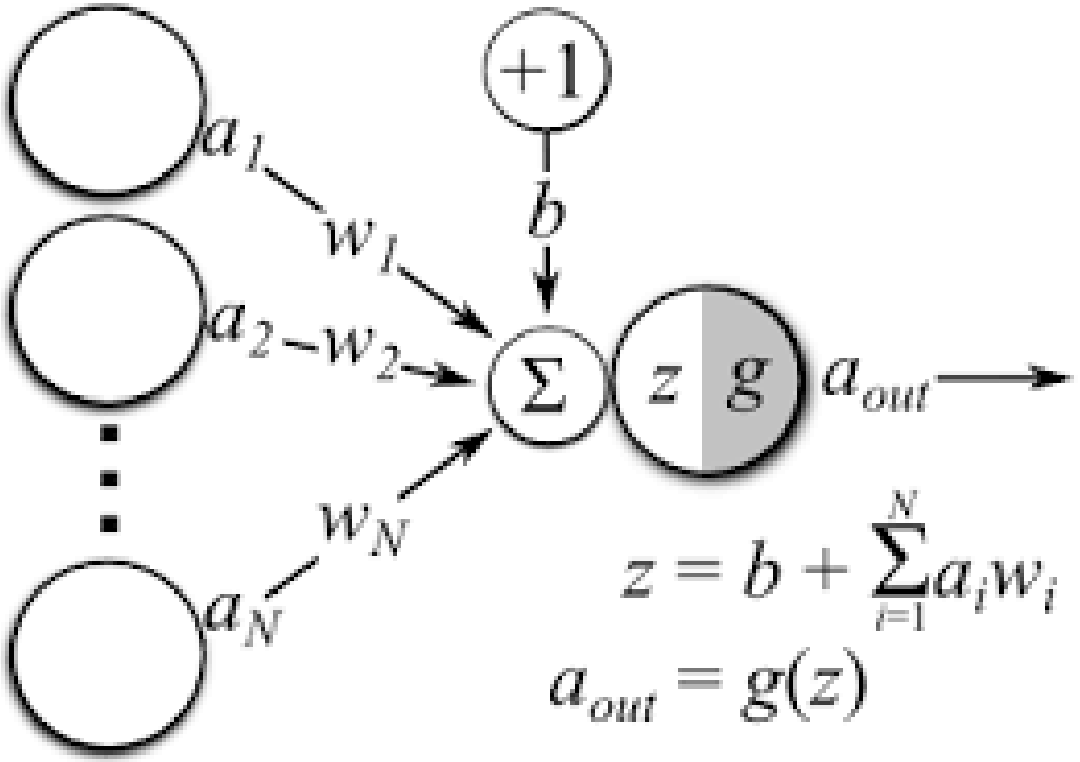
- A set of weights and biases between each layer, W and b. 레이어 사이에 가중치와 바이어스가 있음
- A choice of activation function for each hidden layer, σ. 각 히든 레이어에 엑티베이션 함수가 있음



가중치와 바이어스의 값에 따라 예측의 정확도가 결정됨

The circles are neurons or nodes, with their functions on the data and the lines/edges connecting them are the weights/information being passed along. 동그라미는 함수를 가진 노드들. 노드를 연결시키는 선은 가중치 임!

Output Calculations 출력계산



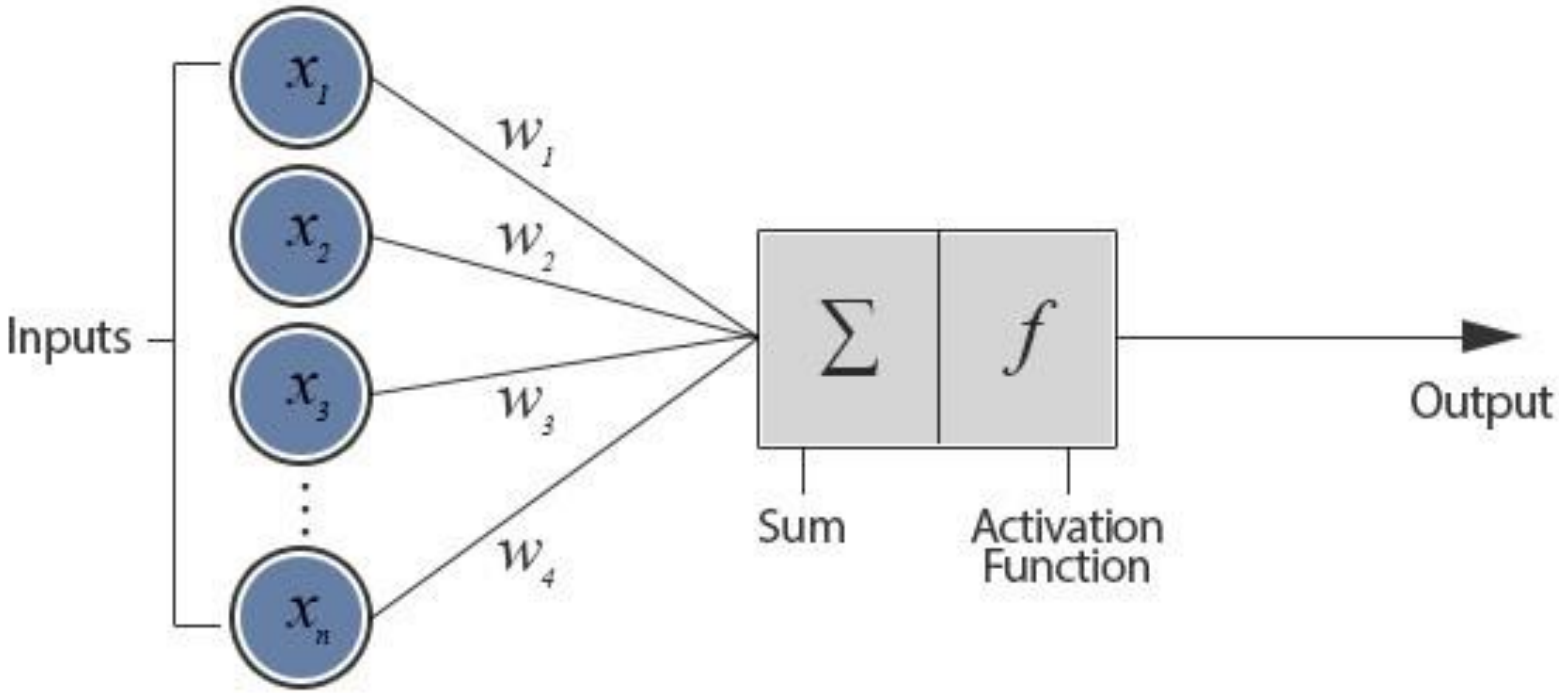
- Receive a set of inputs, perform progressively complex calculations on them, and give output to solve real world problems like classification. 입력을 받아 계산한 후 출력을 함. 분류문제에 적합

Forward Pass:

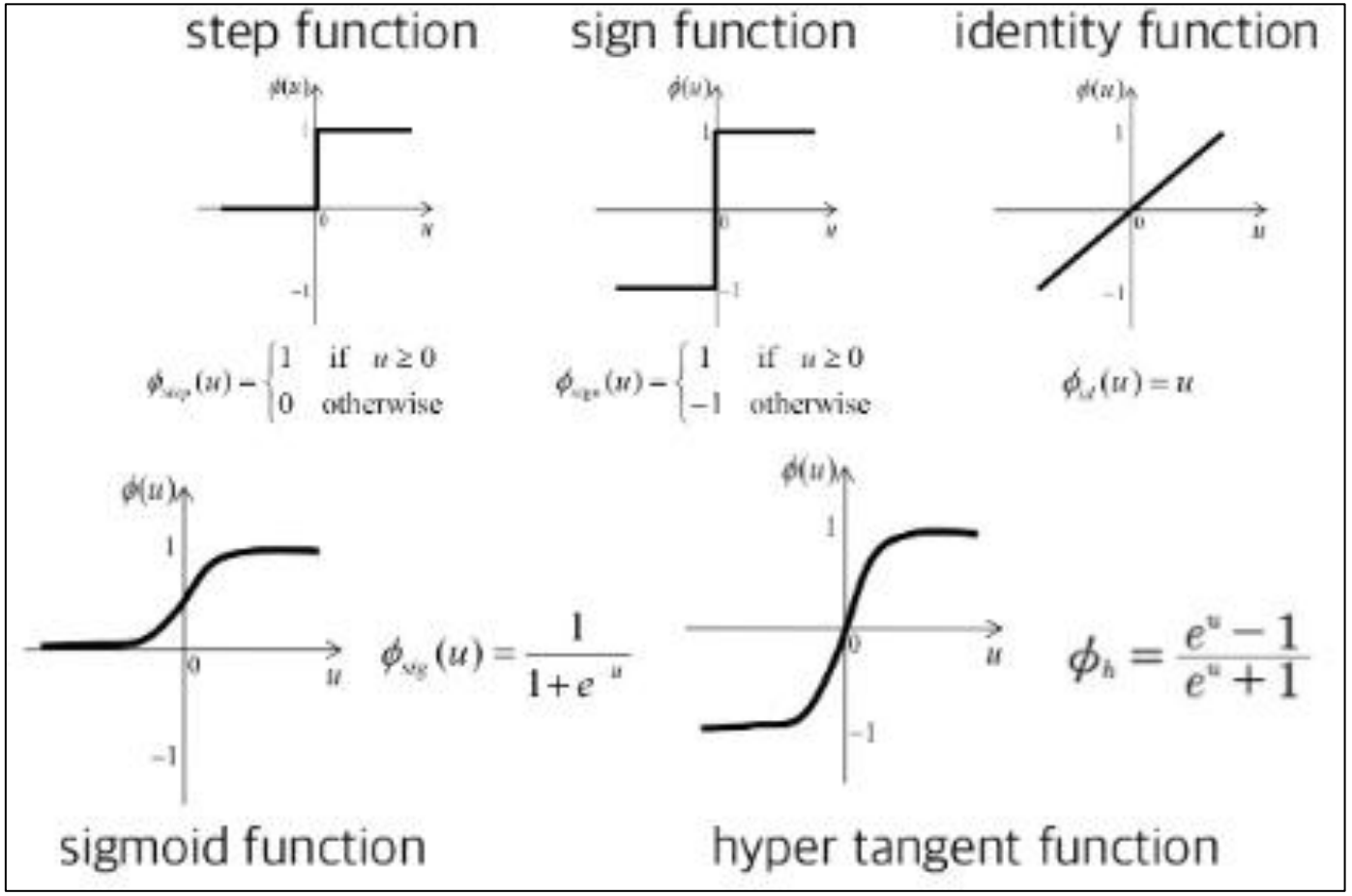
$$z_2 = XW_1$$
$$a_2 = f(z_2)$$
$$z_3 = a_2 W_2$$
$$\hat{y} = f(z_3)$$

Activation Function 활성함수

- output y를 만들기 위해 은닉된 부분의 합을 가중하는 값. e.g., sigmoid, sign activation function.



Activation Functions 활성함수

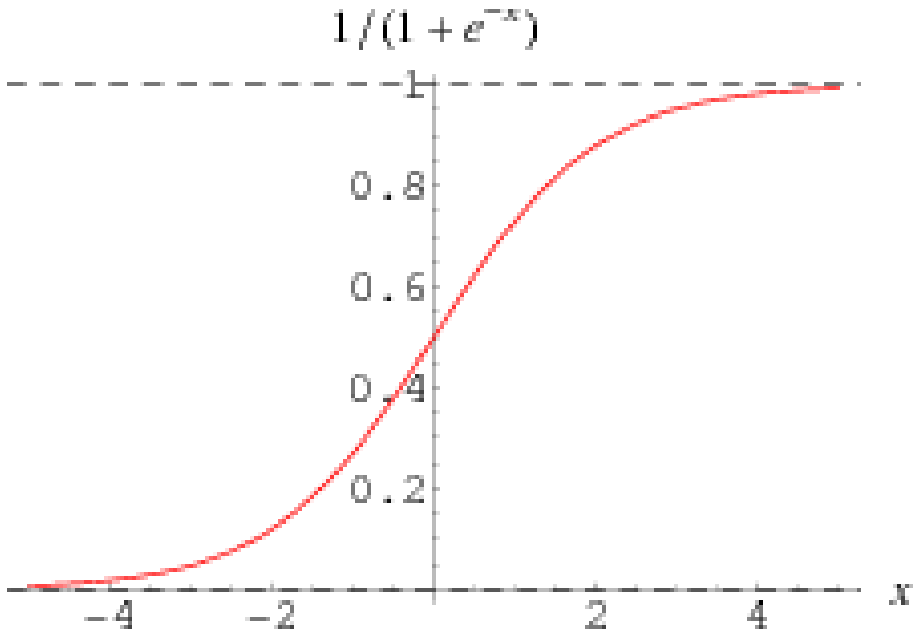


- Can capture non-linearity in the network so it would be capable of learning more complex pattern.
  네트워크 상에 비선형성을 나타낼 수 있어서 복잡한 패턴도 배울 수 있음

Activation: Sigmoid 시그모이드

- A mathematical function having a characteristic "S"-shaped curve or sigmoid curve. s자형곡선을 그리는 함수들
- Also called the sigmoidal curve or logistic function. 시그모이드 또는 로지스틱 함수라고도 함

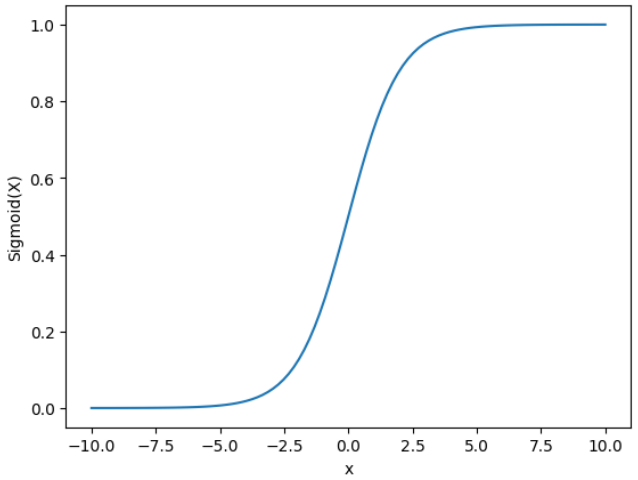
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Sigmoid 코드

```
import numpy as np


def sigmoid(x):
  return 1 / (1+np.exp(-x))
sigmoid(3)
sigmoid(1)
```

```
x = np.linspace(-10, 10, 100)
z = sigmoid(x)
plt.plot(x, z)
plt.xlabel("x")
plt.ylabel("Sigmoid(X)")
plt.show()
```

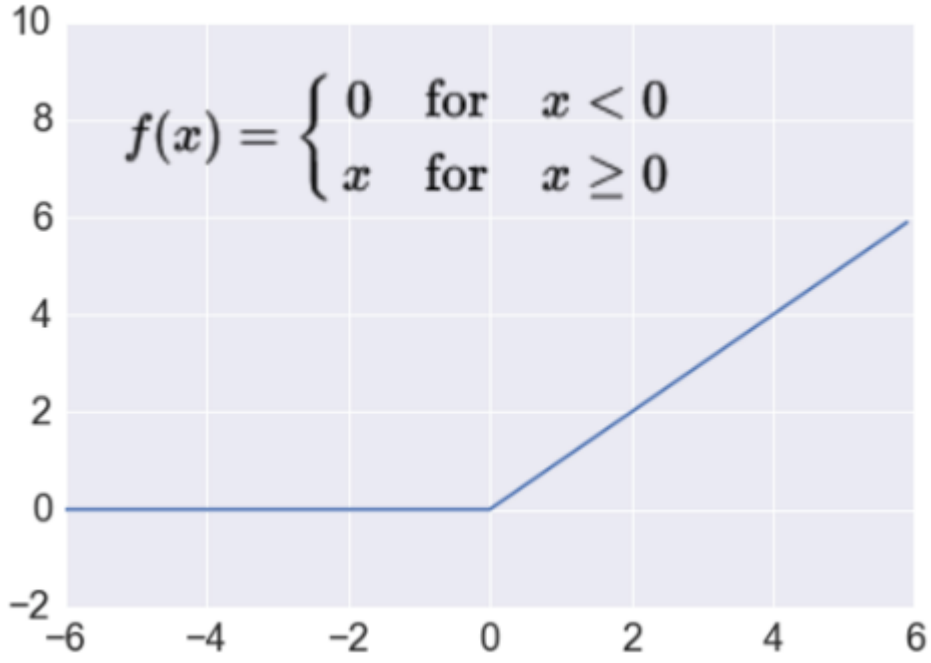


Activation: ReLU 레루 함수

- Most Deep Networks use ReLU nowadays. 딥러닝 네트워크에서는 거의 레루함수를 사용
- Trains much faster and less expensive operations. 훈련 속도가 더 빠르고, 계산이 적음
- Prevents the gradient vanishing problem. 기울기가 소실되는 문제를 방지

$$R^n \rightarrow R_+^n$$

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

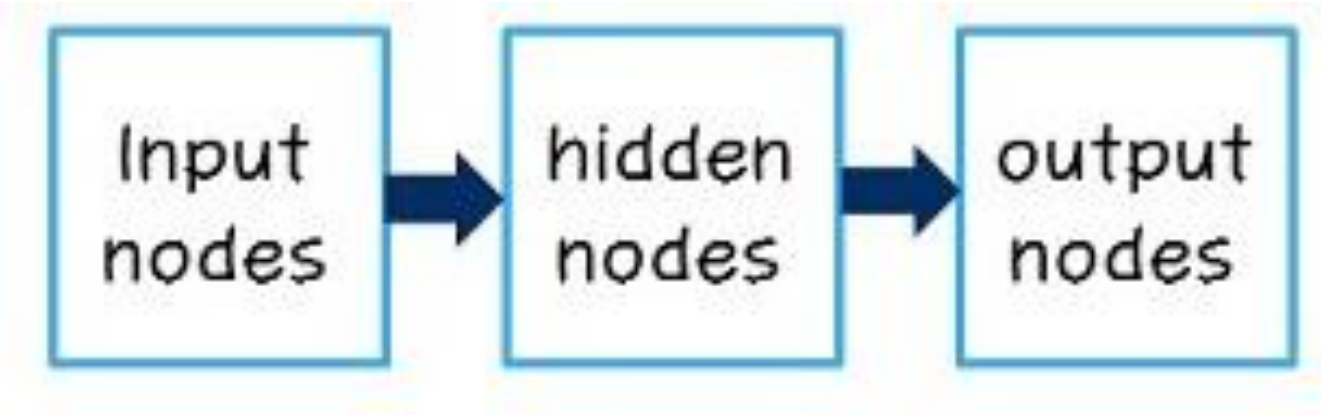


- Takes a real-valued number and thresholds it at zero. 실수를 가져다가 영보다 큰수로 만듬

Feedforward 피드포워드

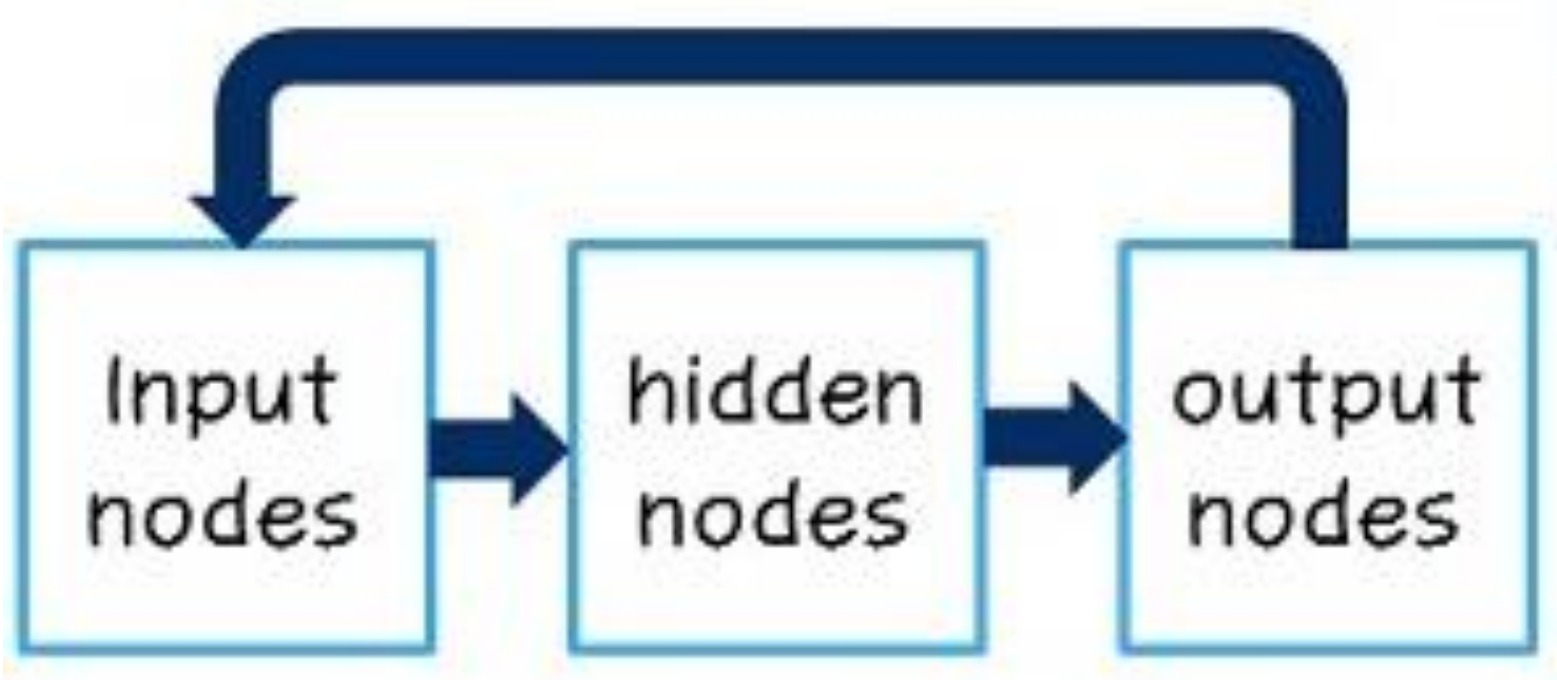
- Just simple calculus (input에서 hidden으로 hidden에서 output으로 쭉 한 방향으로 진행)

$$\hat{y} = \sigma(W_2 \sigma(W_1 x + b_1) + b_2)$$



Backpropagation 역전파

- Find a way to propagate the error back, and to update our weights and biases. 가중치의 에러를 줄이는 방향으로 과정을 계속 반복
- Need to know the derivative of the loss function with respect to the weights and biases. 로스 함수에서 가중치와 바이어스의 변화의 정도를 보여줘야 함



Loss Function

- Evaluate the “goodness” of the predictions (i.e. how far off are the predictions). 예측이 얼마나 정확한지 보여주는 함수
- Can use a simple sum-of-sqaures error (SSE). 에러의 제곱의 합을 사용
- The sum-of-squares error is simply the sum of the difference between each predicted value and the actual value. The difference is squared so that we measure the absolute value of the difference. 에러는 예측값과 실제값의 차이
- Our goal in training is to find the best set of weights and biases that minimizes the loss function. 목적함수, 즉 에러를 최소화하는 것이 훈련의 목적

$$Sum - of - Squares\ Error = \sum_{i=1}^{n} (y - \hat{y})^2$$

Loss Function 손실함수

- Objective function 목적함수
  - The function used to evaluate a candidate solution (i.e. a set of weights) in an optimization algorithm. 최적화 알고리즘에서 후보가 되는 해답을 평가하는 데 사용되는 함수
  - The function to minimize or maximize. 최소화하거나 최대화하기 위한 함수
- Loss function 손실함수
  - When minimizing the objective function, we may call it the cost function, loss function, or error function. 목적함수를 최소화하는 경우 비용함수, 손실함수, 오차함수라고 부름
  - Cross-entropy and mean squared error are the two main types of loss functions to use when training neural network models. 신경망에서 가장 많이 쓰는 손실함수는 크로스 엔트로피와 평균제곱오차임!

정확도를 사용하면 미분값이 0이 나오기 때문에 손실함수를 미분하여 최소값을 찾음

Types of Loss Function 손실함수의 종류

- 회귀모델
  - MSE(Mean Squared Error) 평균제곱오차
  - MAE(Mean Absolute Error) 평균절대오차
  - RMSE(Root Mean Squared Error) 평균제곱근오차
- 분류모델
  - Binary cross-entropy 이진 교차 엔트로피
  - Categorical cross-entropy 범주형 교차 엔트로피
  - Sparse categorical cross-entropy 희박한 범주형 교차 엔트로피

MSE(Mean Square Error) 평균제곱오차

- Calculated as the average of the squared differences between the predicted and actual values. 예측값과 실제값의 차이의 제곱을 평균
- Can be used in a maximization optimization process by making the score negative. 음수를 붙여서 최대화 문제로 풀 수 있음
- Can use the scikit-learn mean_squared_error() function

```
def mean_squared_error(actual, predicted):
	sum_square_error = 0.0
	for i in range(len(actual)):
		sum_square_error += (actual[i] - predicted[i])**2.0
	mean_square_error = 1.0 / len(actual) * sum_square_error
	return mean_square_error
```

MAE(Mean Absolute Error) 평균절대오차

- 실제 값과 예측 값의 차이를 절대값으로 변환하여 평균을 구함

```
actual = [2, 3, 5, 5, 9]
predicted = [3, 3, 8, 7, 6]


n = 5
sum = 0


for i in range(n):
        sum += abs(actual[i] - predicted[i])
error = sum/n
print("Mean absolute error : " + str(error))
```

```
from sklearn.metrics import
  mean_absolute_error
actual = [2, 3, 5, 5, 9]
predicted = [3, 3, 8, 7, 6]


error = mean_absolute_error(actual, predicted)
print("Mean absolute error : " + str(error))
```

Cross Entropy 교차 엔트로피

- 두 확률 분포가 얼마나 비슷한지 정량적으로 나타내 주는 수치
- “cross-entropy,” “logarithmic loss,” “logistic loss,” or “log loss”라고도 불림
- 실제값과 예측값의 일치하면 0이 되고 일치하지 않으면 값이 커짐

```
from math import log
def cross_entropy(actual, predicted):
	sum_score = 0.0
	for i in range(len(actual)):
		sum_score += actual[i] * log(
1e-15 + predicted[i])
	mean_sum_score = 1.0 / len(actual) *
sum_score
	return -mean_sum_score
```

- 밑이 e인 자연로그에서 진수가 0이 나오는 경우 로그의 값은 -inf가 되어 수치연산이 불가능함 → 아주 작은 값(delta, 1e-15)를 더함

$$H_p(q) = -\sum_{i=1}^{n} q(x_i) log p(x_i)$$

q는 실제 확률분포, p는 예측 확률분포

Binary Crossentropy 이진교차 엔트로피

- true/false, 양성/음성처럼 2개의 클래스로 분류할 때 적절
- scikit-learn log_loss function를 사용할 수 있음

$$BinaryCrossEntropy = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(y_i^{prob}) + (1-y_i) log(1-y_i)]$$

```
def binary_cross_entropy(actual, predicted):

    result = -(np.dot(actual,np.log(predicted+1e-7))+np.dot((1-actual)
.T,np.log(1-predicted+1e-7)))/len(actual)
    return result
actual = np.array([0, 1, 0, 0, 1])
predicted = np.array([0, 1, 1, 1, 1])
binary_cross_entropy(actual, predicted)
```

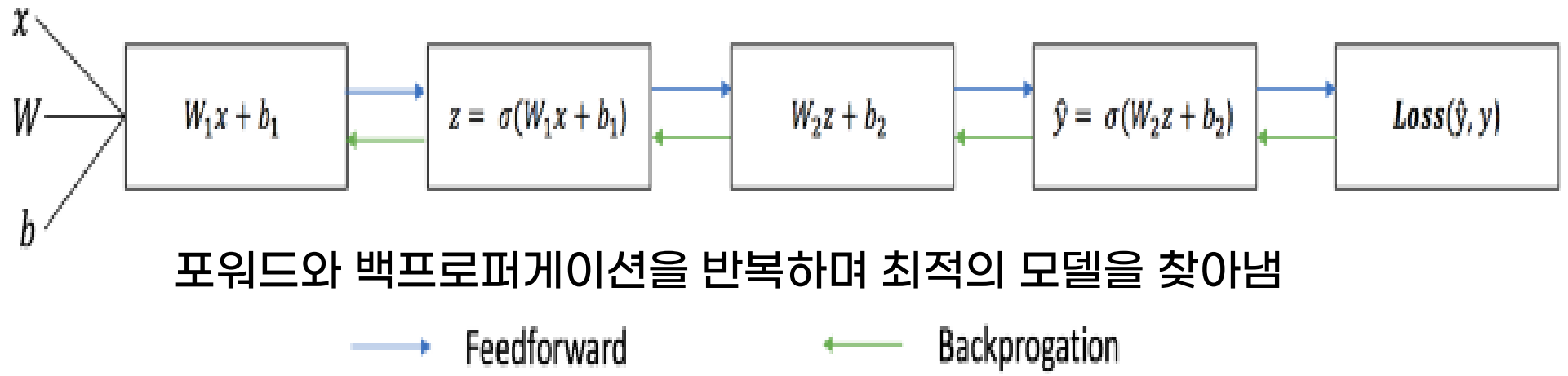
Categorical Crossentropy
범주형 교차 엔트로피

- 클래스가 3개 이상이고, one-hot encoding으로 처리된 경우
- 클래스가 3개 이상이고, label encoding으로 처리된 경우는 Sparse Categorical Crossentropy를 사용

```
from math import log
def categorical_cross_entropy(actual, predicted):
	sum_score = 0.0
	for i in range(len(actual)):
		for j in range(len(actual[i])):
			sum_score += actual[i][j] * log(1e-15 + predicte
d[i][j])
	mean_sum_score = 1.0 / len(actual) * sum_score
	return -mean_sum_score
```

Sequential Graph 순차그래프

- The process of fine-tuning the weights and biases from the input data. 학습을 통해 가중치와 바이어스의 값을 조정함
- Calculating the predicted output $\hat{y}$, known as feedforward. 처음 출력값을 계산할 때는 피드포워드
- Updating the weights and biases, known as backpropagation. 백워드로 가중치와 바이어스를 업데이트



포워드와 백프로퍼게이션을 반복하며 최적의 모델을 찾아냄

College Dataset 대학 데이터세트

- Use College Data Set which has several features of a college and a categorical column and build a predictive model to determine whether or not the School is Public or Private. 컬리지 데이터 세트에는 대학에 관한 여러가지 특징들과 범주형 변수들이 포함돼 있습니다. 그 대학이 공립인지 사립인지 결정하는 예측모델을 구축하시오.

Variables 변수들

- Private : Public/private indicator
- Apps : Number of applications received
- Accept : Number of applicants accepted
- Enroll : Number of new students enrolled
- Top10perc : New students from top 10% of high school class
- Top25perc : New students from top 25% of high school class
- F.Undergrad : Number of full-time undergraduates
- P.Undergrad : Number of part-time undergraduates
- Outstate : Out-of-state tuition
- Room.Board : Room and board costs
- Books : Estimated book costs
- Personal : Estimated personal spending
- PhD : Percent of faculty with Ph.D.'s
- Terminal : Percent of faculty with terminal degree
- S.F.Ratio : Student/faculty ratio
- perc.alumni : Percent of alumni who donate
- Expend : Instructional expenditure per student
- Grad.Rate : Graduation rate

It contains a number of variables for 777 different universities and colleges in the US. **777개의 대학 데이터**

Exercise #1

- x and y split
- Scaling x and Encoding y before building model
- Train and test split
- Resample imbalanced data (SMOTE)
- Build a Logistic Regression model
- Evaluate the results (accuracy_score, confusion_matrix, classification_report)
- Predict with test data.

Exercise #2

- Build a NN Model (First layer: 10 nodes, with relu activation, 2nd layer: 10 nodes, with relu activation, 3rd layer: output, with sigmoid activation). 신경망모델 구축
- Compile the model (loss='binary_crossentropy', optimizer='adam', metrics=['accuracy']). 모델 컴파일
- Fit the model (epochs=500). 모델훈련
- Evaluate the results(accuracy_score, confusion_matrix, classification_report, roc_curve) 결과평가

Neural Network Model 신경망 모델

```
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(10, input_dim=x_train.shape[1], activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.summary()
```

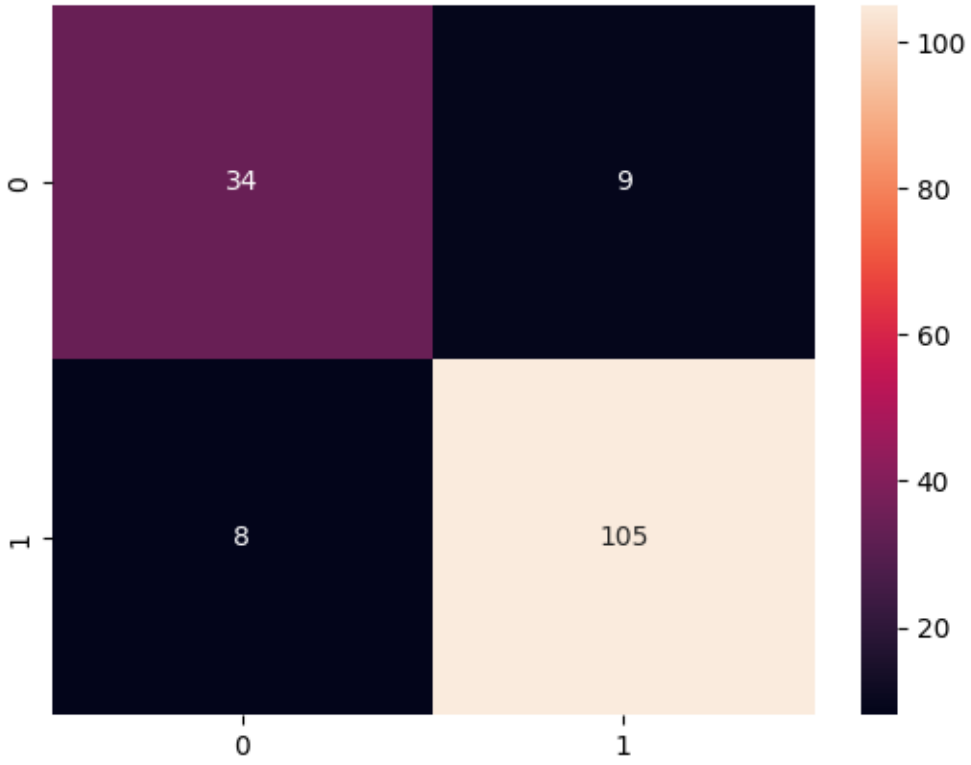
Model Summary 모델 요약

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense_11 (Dense)             (None, 10)                180

 dense_12 (Dense)             (None, 10)                110

 dense_13 (Dense)             (None, 10)                110

 dense_14 (Dense)             (None, 1)                 11

=================================================================
Total params: 411
Trainable params: 411
Non-trainable params: 0
_________________________________________________________________
```

Model Evaluation 모델평가

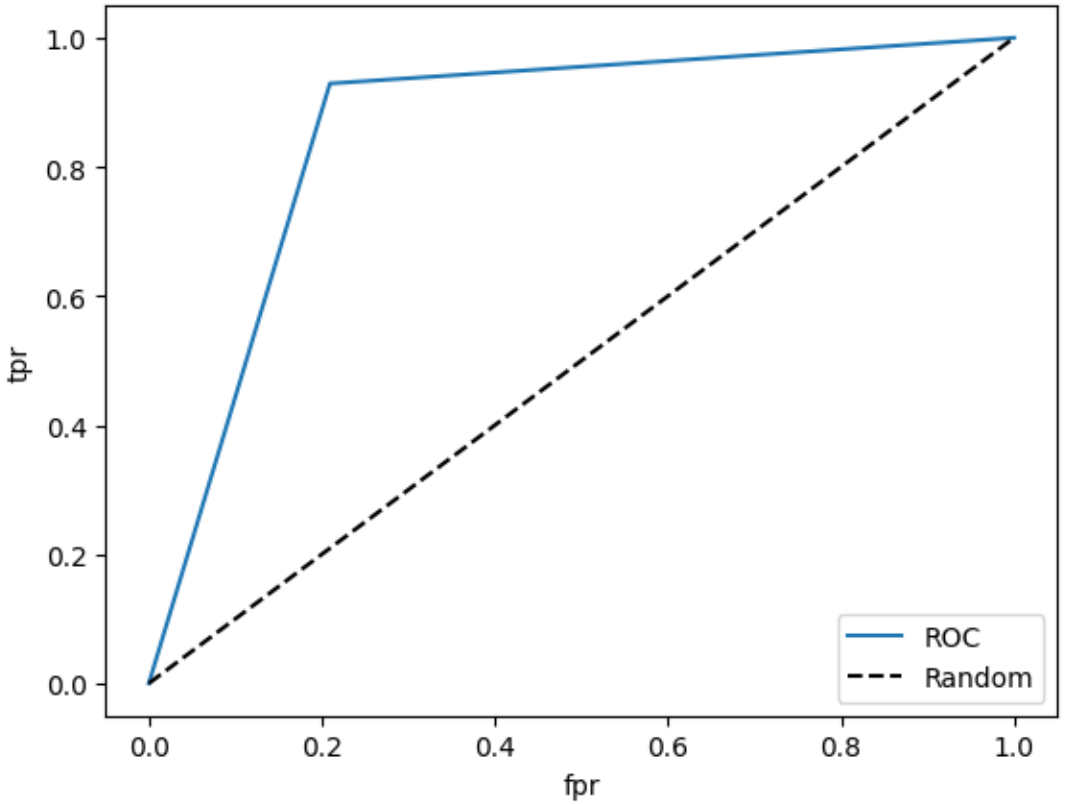
```
model.fit(x_train, y_train, epochs=30, validation_data=(x_test,
y_test))

model.evaluate(x_test, y_test)

y_pred = model.predict(x_test).round()

nn_score = accuracy_score(y_test, y_pred) #test performance

print("Neural Network Test Accuracy: {}".format(nn_score))

confusion_matrix(y_test, y_pred) #test performance

print(classification_report(y_test, y_pred))
```

Performance Measures 성과측정



	precision	recall	f1-score	support
0	0.81	0.79	0.80	43
1	0.92	0.93	0.93	113
accuracy			0.89	156
macro avg	0.87	0.86	0.86	156
weighted avg	0.89	0.89	0.89	156

ROC Curve




```
fpr , tpr , thresholds =
roc_curve(y_test,
model.predict(x_test).round())

plt.plot(fpr , tpr, label='ROC')

plt.plot([0, 1], [0, 1], 'k--',
label='Random')  # 가운데 대각선

plt.legend()

plt.xlabel('fpr')

plt.ylabel('tpr')

plt.show()
```

Libraries for Deep Learning
딥러닝을 위한 라이브러리

Theano

Keras

TensorFlow

PyTorch

TensorFlow 텐서플로우

- An open-source library for numerical computation, for which it uses data flow graphs. 데이터 플로우 그래프를 사용하기 위한 계산 라이브러리
- Developed by Google Brain and actively used at Google both for research and production needs. 구글 브레인에 의해 개발되어 연구, 생산 목적으로 사용됨
- You can install it with Python pip or conda. 피아피나 콘다를 이용하여 설치할 수 있음

```
conda install tensorflow
import tensorflow as tf
print(tf.__version__)
```

PyTorch 파이토치

- An open source machine learning library based on the Torch library. 토치 라이브러리를 기반으로 하는 오픈 소스 머신 러닝 라이브러리
- Used for applications such as computer vision and natural language processing. 컴퓨터 비전 및 자연어 처리와 같은 응용 프로그램에 사용
- Primarily developed by Facebook's AI Research lab. 페이스북의 인공지능 연구랩에 의해 개발

Keras 캐라스

- A powerful easy-to-use Python library for developing and evaluating deep learning models. 딥러닝모델을 개발하고 평가하기 위한 라이브러리
- Run on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). 텐서플로우, 티아노, CNTK 위에서 돌아감
- You can install it with Python conda or pip. 콘다나 피아피를 이용하여 설치할 수 있음

```
pip install keras
```

Theano 티아노

- A python library which provides a set of functions for building deep nets that train quickly on our machine. 딥넷을 개발하기 위한 수학 함수를 제공
- Have to build the deep net from ground up. 바닥부터 딥넷을 개발해야 함
- TensorFlow and Keras can be used with Theano as backend. 텐서플로우와 캐라스는 티아노와 함께 백엔드로 사용
- Developed at the University of Montreal, Canada under the leadership of Yoshua Bengio a deep net pioneer. 몬트리올 대학에서 개발

Pre-Processing 전처리

- Normalize the data values to the range [0, 1]. 0과 1사이의 값으로 정규화시킴

```
x_train = x_train/255
x_test = x_test/255
```

Sequential Model 시퀀셜 레이어

- Sequential is the easiest way to build a model in Keras. 캐라스로 가장 쉽게 만들수 있는 모델
- It allows you to build a model layer by layer.
  층층으로 모델을 만들수 있게 함

```
model = Sequential()
```

Flatten Layer 플랫튼 레이어

- Serves as a connection between the convolution and dense layers.
  컨볼루션과 덴스레이어 사이를 연결

```
model.add(Flatten())
```

Dense Layer 덴스레이어

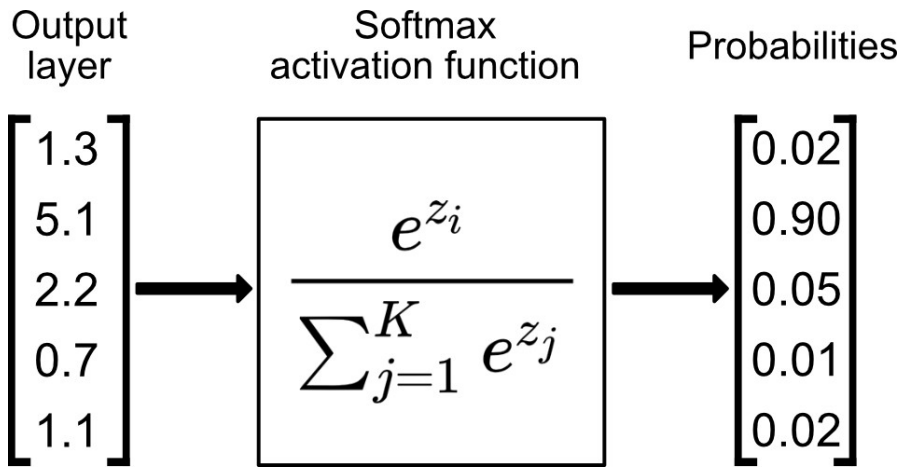
- Layer type used in output layer. 출력 레이어에 사용
- 10 nodes in output layer (possible outcome between 0 and 9). 10개의 노드는 0에서 9까지를 출력
- Softmax - output sum up to 1 as probabilities. Can make prediction based on the probability. 출력의 확률의 합은 1. 확률을 기초로 예측

```
model.add(Dense(10, activation='softmax'))
```

입력층, 은닉층, 출력층 등이 각각 어떤 특성을 가질 지 옵션을 설정하는 역할

Softmax 소프트맥스

- Assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. 다중 클래스 문제의 각 클래스에 소수 확률을 할당. 확률의 합은 1.0 이어야 함




```
import numpy as np


def softmax(x):
    """Compute softmax values for each
  sets of scores in x."""
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum()


scores = [3.0, 1.0, 0.2]
print(softmax(scores))
```

Dropout Layer 드랍아웃 레이어

- Randomly switches off some neurons in the network which forces the data to find new paths. Therefore, this reduces overfitting. 무작위로 뉴론을 꺼버림으로써 데이터가 새로운 경로를 찾게 하여 오버피팅을 방지

```
from keras.layers import Dropout
model.add(Dropout(0.25))
```

Testing Keras 케라스 테스팅

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

model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test, y_test, verbose=2)
```

Types of Loss Function 손실함수의 종류

- Regression Loss Functions
  - Mean Squared Error Loss
  - Mean Squared Logarithmic Error Loss
  - Mean Absolute Error Loss
- Binary Classification Loss Functions
  - Binary Cross-Entropy
  - Hinge Loss
  - Squared Hinge Loss
- Multi-Class Classification Loss Functions
  - Multi-Class Cross-Entropy Loss
  - Sparse Multiclass Cross-Entropy Loss
  - Kullback Leibler Divergence Loss

```
loss=‘categorical_crossentropy‘

loss='sparse_categorical_crossentropy‘

loss='binary_crossentropy‘

loss='squared_hinge‘

loss = tf.reduce_mean(tf.square(y_pred-y))
```

categorical_crossentropy vs sparse_categorical_crossentropy

- For one-hot encoded y, use categorical_crossentropy. 원핫 인코드된 y값은 카테고리컬 크로스엔트로피 사용

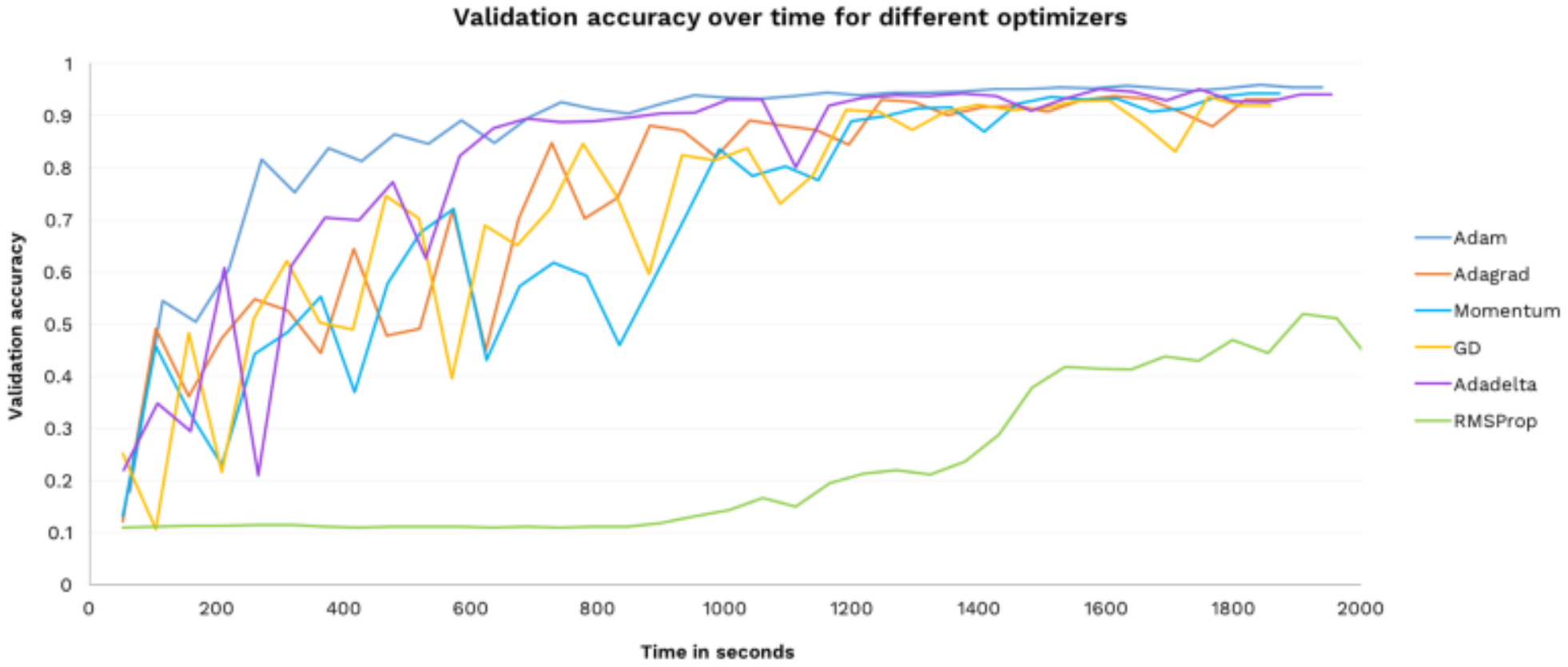
  [1,0,0]
  [0,1,0]
  [0,0,1]

- For integer y, use sparse_categorical_crossentropy. 와이값이 정수이면 스팔스 카테고리컬 크로스엔트로피 사용

  1
  2
  3

Optimizers 옵티마이저

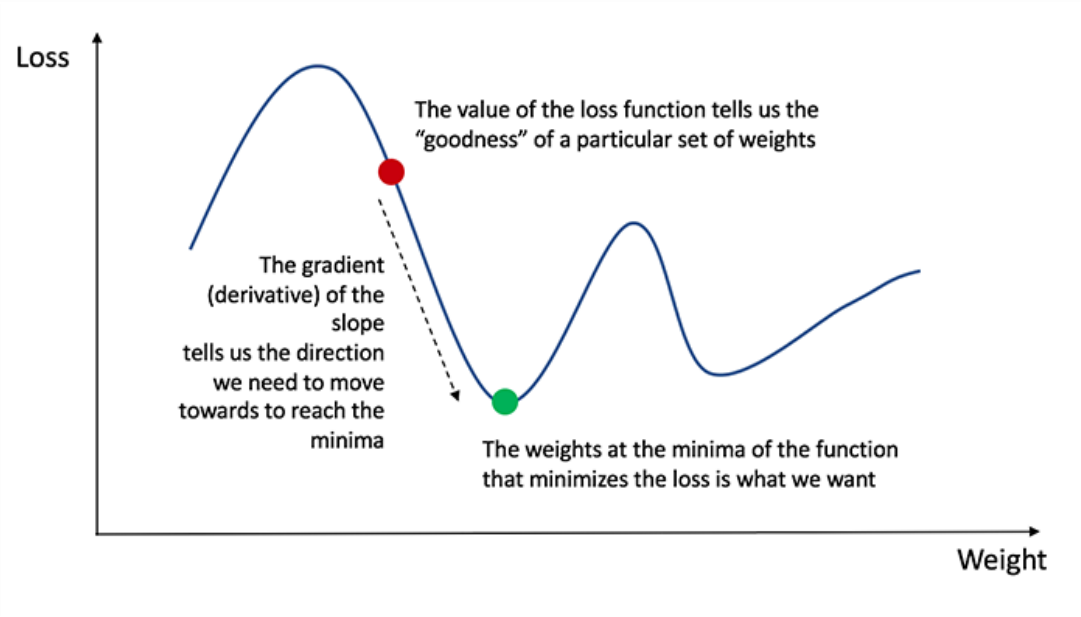
```
from keras.optimizers import SGD
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```



Gradient Descent 경사하강

- Simply update the weights and biases by increasing/reducing the loss function. 로스함수의 증가/감소를 이용하여 가중치와 바이어스를 조정
- Adjusts weights according to the error they caused. 에러를 최소화하는 방향으로 가중치를 조정, 즉 기울기의 경사가 최소화를 위한 방향을 정해줌

미분값이 0인 지점이 최소값




```
optimizer =
tf.train.GradientDescentOptimizer(learning_rate=0.1)
train_op = optimizer.minimize(loss)
```

Adam (Adaptive Moment Estimation)

- Another method that computes the adaptive learning rates for each parameter by considering the exponentially decaying average of past squared gradients and the exponentially decaying average of past gradients. 기하급수적으로 쇠퇴하는 과거의 경사도의 제곱의 평균과 기하급수적으로 쇠퇴하는 과거 경사도의 평균을 고려하여 각 계수에 대해 학습률을 계산

```
opt = tf.keras.optimizers.Adam(lr=1e-3, decay=13-5)
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])
```

ADAgrad Optimizer

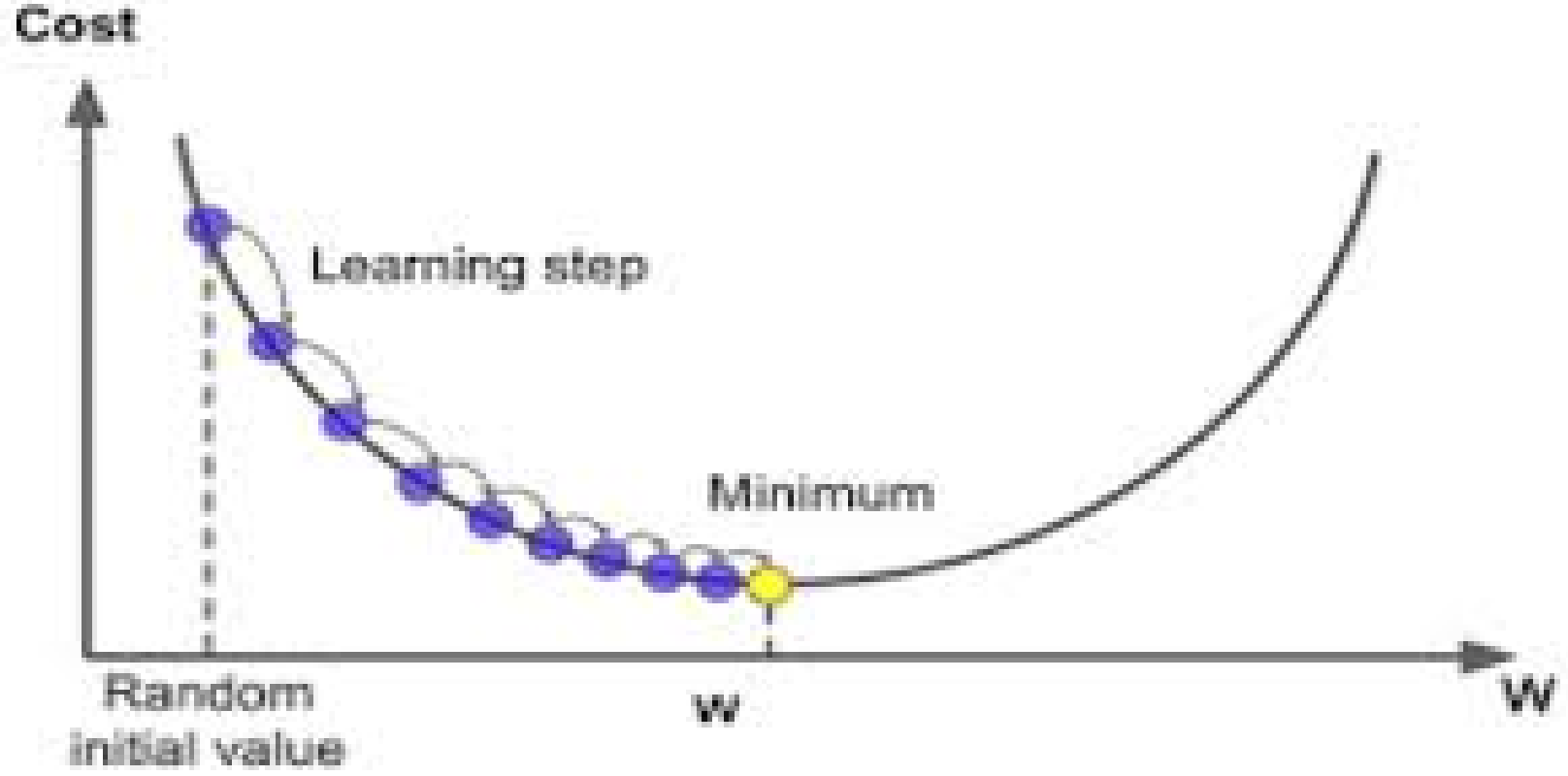
- Uses a different learning rate for every parameter and every time step. 계수마다 시간 간격마다 학습률이 다름
- The parameters that are infrequent must have larger learning rates while parameters that are frequent must have smaller learning rates. That is, the stochastic gradient descent update for parameters. 자주 쓰이지 않는 계수는 학습률이 크고, 자주 쓰이는 계수는 학습률을 낮게 조정. 확률적인 계수의 업데이트를 사용
- The learning rate is calculated based on the past gradients that have been computed for each parameter. 계수 계산시 사용됐던 경사도가 학습률 계산에 사용됨
- The learning rates start vanishing very quickly as the iterations increase. 반복될 수 록 학습률이 빨리 사라짐

RMSprop

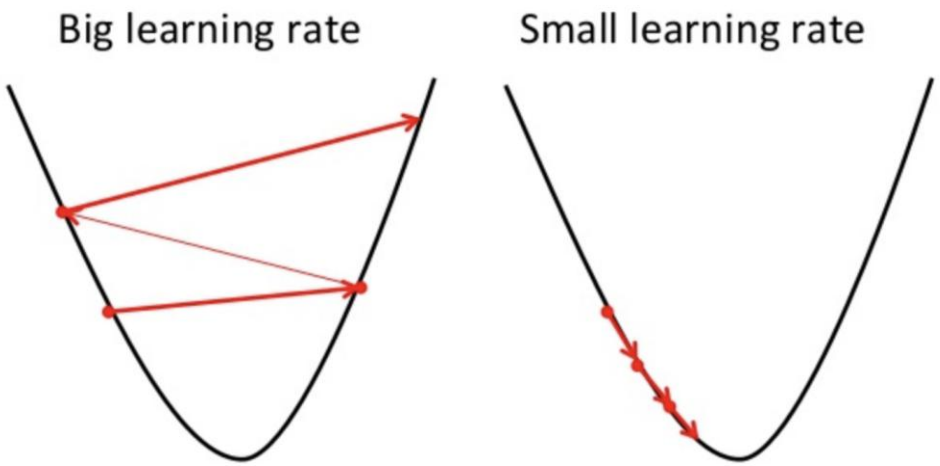
- Considers fixing the diminishing learning rate by only using a certain number of previous gradients. 과거 경사도 중 특정한 갯수를 사용해서 학습률이 줄어드는 것을 고정시킴

Learning Rate 학습률

- Too small, then converge very slowly. 너무 작으면 수렴되는 데 오래걸림
- Too big, then overshoot and even diverge. 너무 크면 최소점을 지나치거나 갈라짐



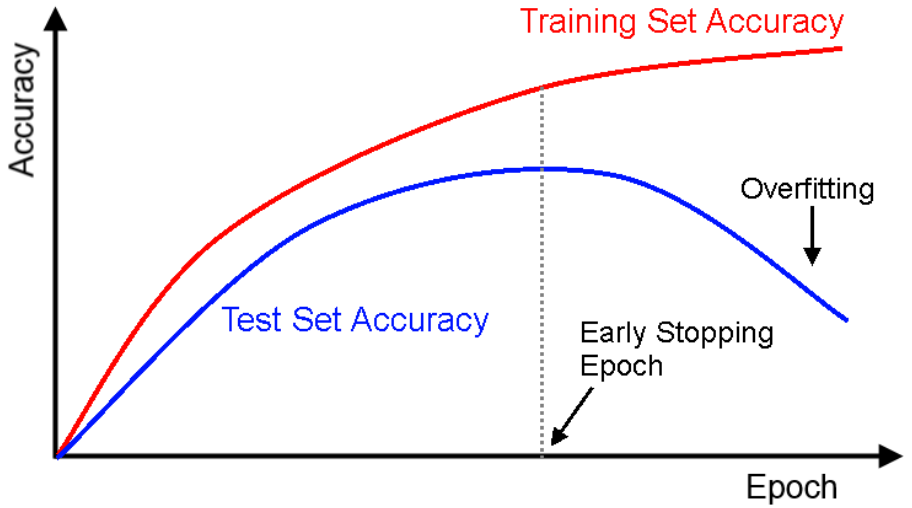
Adjusting Learning Rate 학습률 조정



- Start with a large value like 0.1, then try exponentially lower values: 0.01, 0.001, etc. 0.1에서 시작해서 0.01, 0.001을 시도

Epochs 포워드, 백워드 횟수

- The number of times the model will cycle through the data. The more epochs we run, the more the model will improve, up to a certain point. After that point, the model will stop improving during each epoch. 데이터를 가지고 한번 포워드, 백워드 계산을 하는 수. 이팍횟수가 많아질 수록 모델이 향상됨. 어떤 지점에 이르면 값이 수렴되며 더이상 향상되지 않음

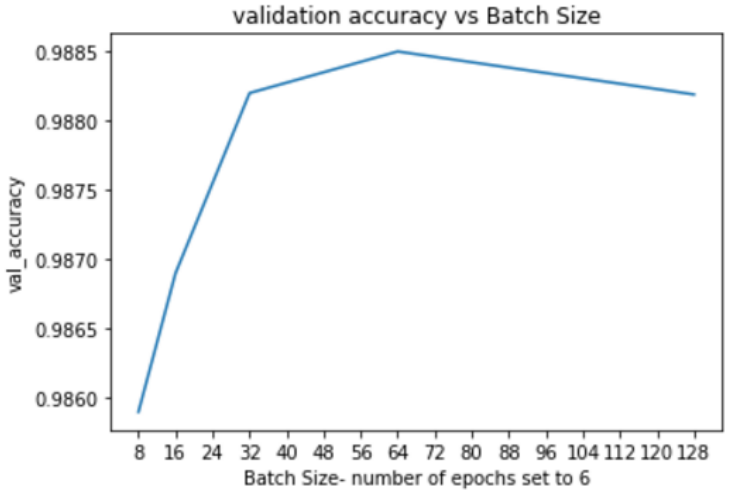


No right numbers of epochs
It is related to how diverse your data is. 정답이 없음. 데이터의 다양성에 따라 효과적인 이팍의 횟수가 달라짐

Batch Size 배치크기

- The number of training examples utilized in one iteration. 한번 훈련할 때 사용하는 데이터의 갯수
- For example, you can divide the dataset of 2000 examples into batches of 500 then it will take 4 iterations to complete 1 epoch. 2000개의 데이터를 가지고 500개의 배치로 나눠서 모델을 만드는 경우 한번의 이팍에 4번의 계산을 수행하게 됨

In general, batch size of 32 is a good starting point, and then try 64, 128, and 256. 32로 시작해서 64, 128, 256로 증가시킴



Verbose 출력설정

- How you want to see the training progress for each epoch. 훈련과정을 어떻게 보고 싶은지 결정

  - verbose=0 will show nothing. 아무것도 안보임
  - verbose=1 will show an animated progress bar. 진행바를 보여줌

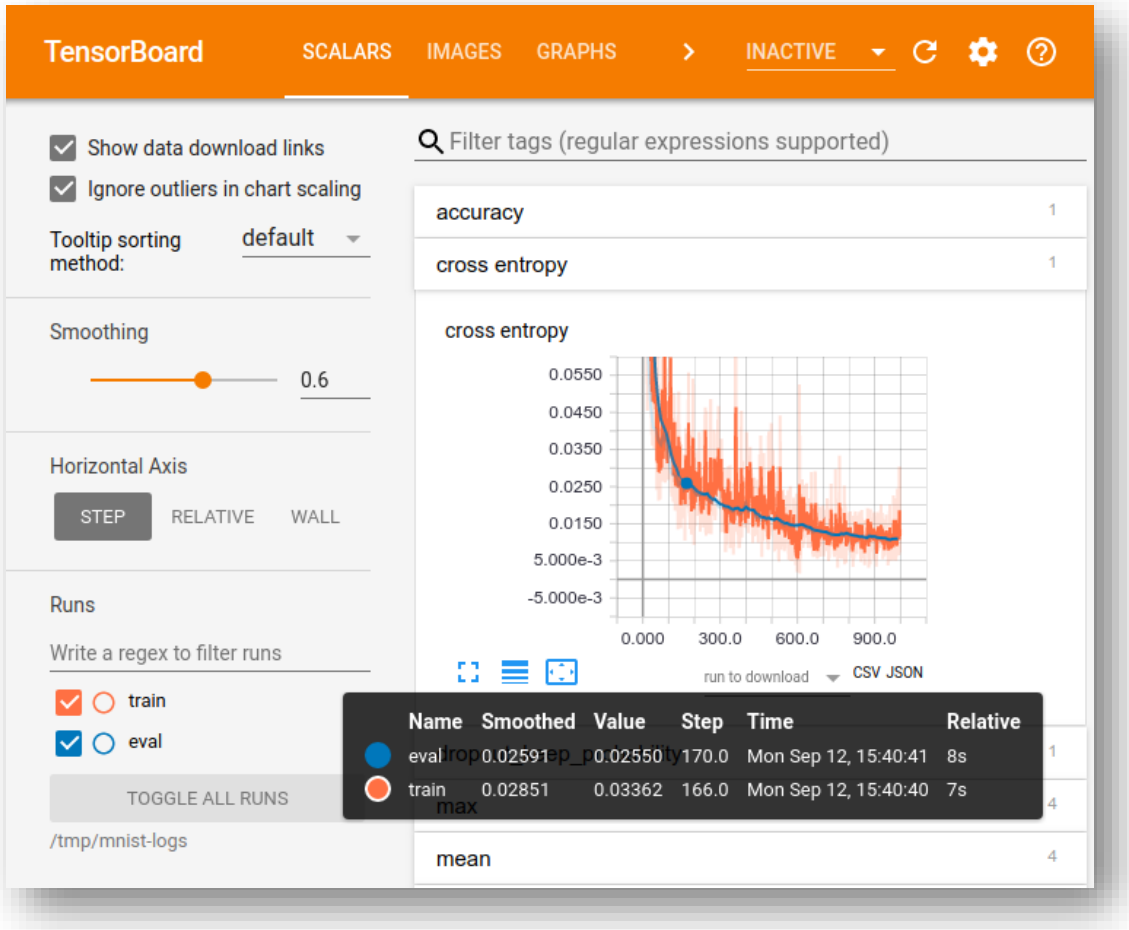
```
[==============================]
```

  - verbose=2 will show the number of epoch. 진행수를 보여줌

```
Epoch 1/10
```

TensorBoard 텐서보드

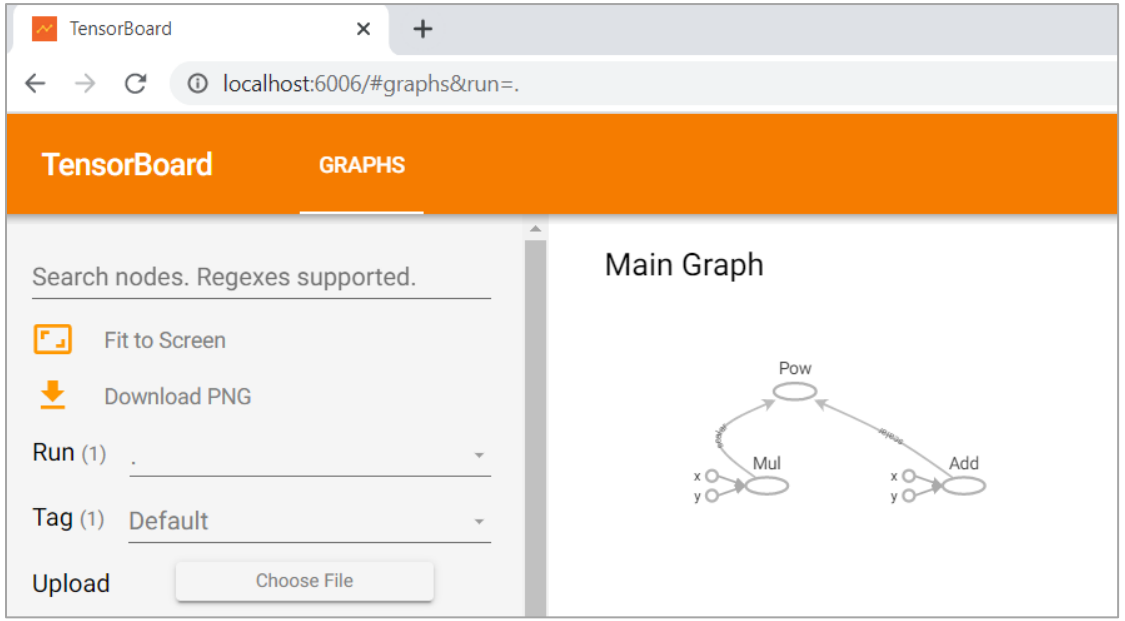
- A visualization software that comes with any standard TensorFlow installation. 텐서플로우 설치하면 같이 오는 시각화 소프트웨어



Visualizing Graph 그래프시각화

1

```
from keras.callbacks import
TensorBoard

from time import time

tensorboard =
TensorBoard(log_dir='logs\\{}'.for
mat(time()))

model.fit(X_train, y_train,
validation_data=(X_test, y_test),
epochs=1,callbacks=[tensorboard])
```



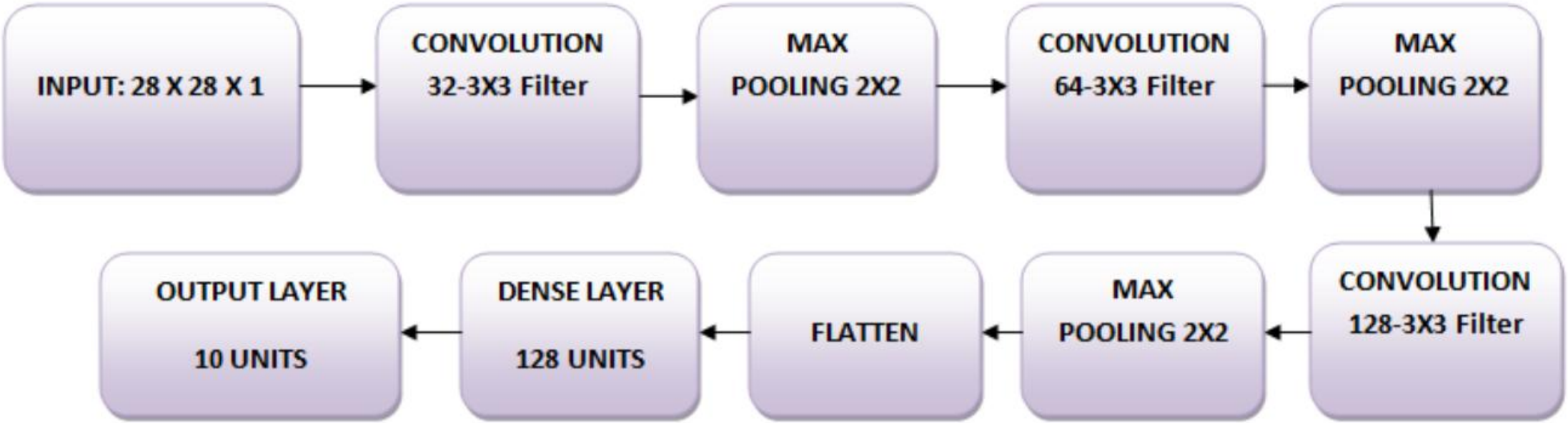
2

명령프롬프트에서 텐서보드 실행
tensorboard --logdir=logs/

3

브라우저에 로컬호스트치고 텐서보드 들어감
http://localhost:6006

Architecture of CNN Model 모델 구조



Exercise #3

- Load fashion_mnist dataset and split it into train and test set (keras.datasets.fashion_mnist). 패션이미지를 가지고 와서 훈련, 테스트세트로 분리
- Exploratory data analysis (imshow of 9 images of train and the 9 images of test, shape of train and test, and number of classes and names of classes). 데이터 탐색
- Data preprocessing (reshape, divided by 255 to normalize, encoding y) 전처리

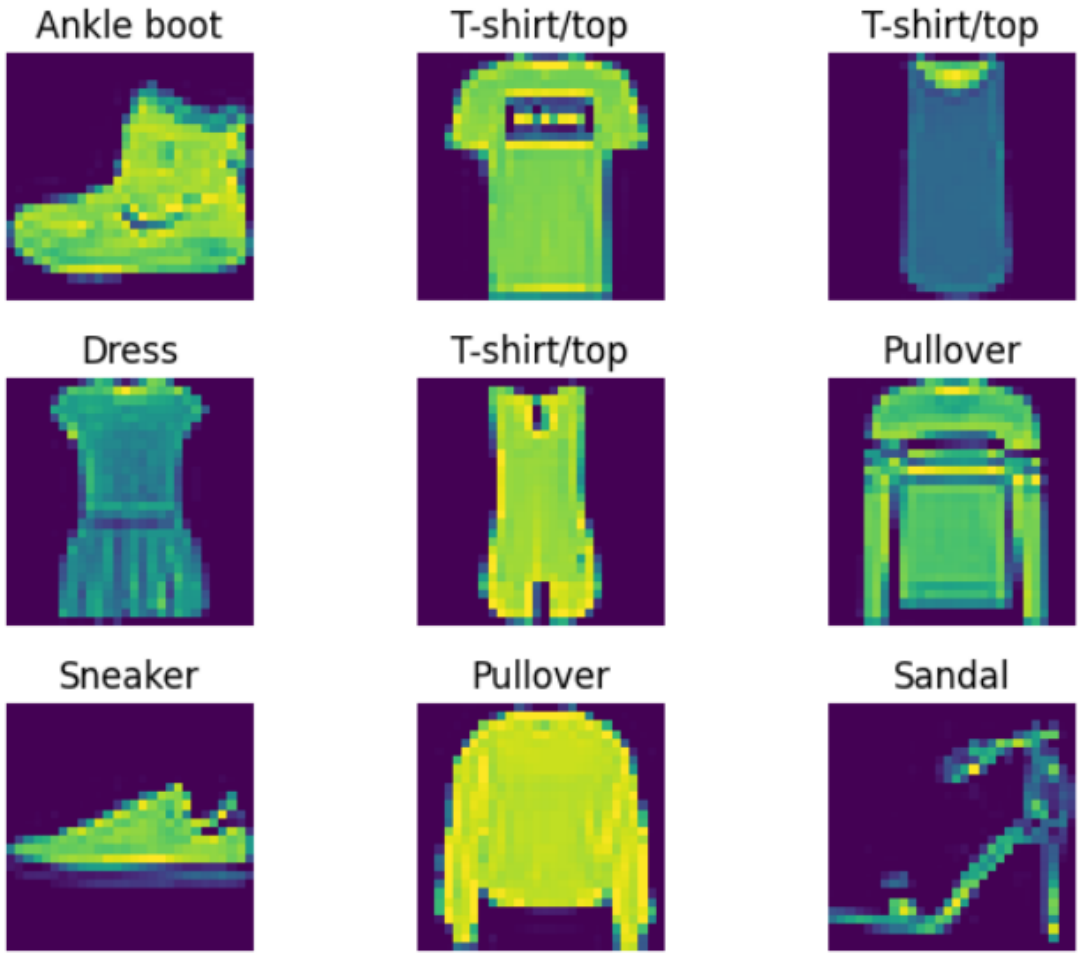
Refer to https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python

Loading Dataset 데이터 불러오기

```
# 패션이미지를 가지고 와서 훈련, 테스트세트로 분리
from keras.datasets import fashion_mnist


(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

Exploratory Data Analysis 탐색분석

```
# 훈련, 테스트 사이즈 확인

x_train.shape, x_test.shape


# 훈련 데이터 시각화

label_names = ["T-shirt/top", "Trouser",
"Pullover", "Dress", "Coat", "Sandal",
"Shirt", "Sneaker", "Bag", "Ankle boot"]


for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(x_train[i])
    plt.title(label_names[y_train[i]])
    plt.axis('off')
plt.tight_layout()
plt.show()
```



Data Pre-processing 데이터 전처리

```
# 이미지 사이즈 변환
x_train = x_train.reshape(60000,28,28,1)
x_test = x_test.reshape(10000,28,28,1)

# 이미지 정규화
x_train = x_train/255.0
x_test = x_test/255.0

# 라벨 인코딩
from keras.utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

Exercise #3

- Build a CNN model. 컨볼루션 신경망 구축
  - Conv2d with 32 nodes, 3x3 filters, relu activation function
  - Maxpooling with 2x2 filers
  - Conv2d with 64 nodes, 3x3 filters, relu activation function
  - Maxpooling with 2x2 filers
  - Conv2d with 128 nodes, 3x3 filters, relu activation function
  - Maxpooling with 2x2 filers
  - Flatten
  - Dense with 128 nodes with relu activation
  - Dense with softmax function

Building Model 모델 구축

```
# CNN 모델 구축
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten

model = Sequential()
model.add(Conv2D(32, 3, activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64, 3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(128, 3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
```

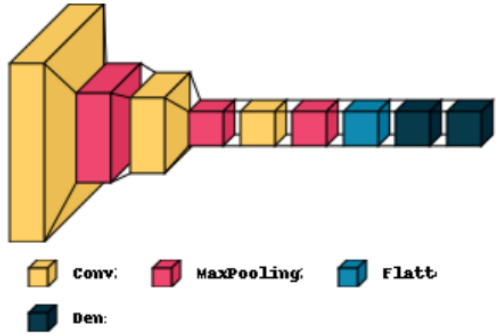
Model Summary 모델 요약

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_9 (MaxPoolin  (None, 13, 13, 32)        0
 g2D)

 conv2d_10 (Conv2D)          (None, 11, 11, 64)        18496

 max_pooling2d_10 (MaxPooli  (None, 5, 5, 64)          0
 ng2D)

 conv2d_11 (Conv2D)          (None, 3, 3, 128)         73856

 max_pooling2d_11 (MaxPooli  (None, 1, 1, 128)         0
 ng2D)

 flatten_3 (Flatten)         (None, 128)               0

 dense_6 (Dense)             (None, 128)               16512

 dense_7 (Dense)             (None, 10)                1290

=================================================================
Total params: 110474 (431.54 KB)
Trainable params: 110474 (431.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

모델 확인

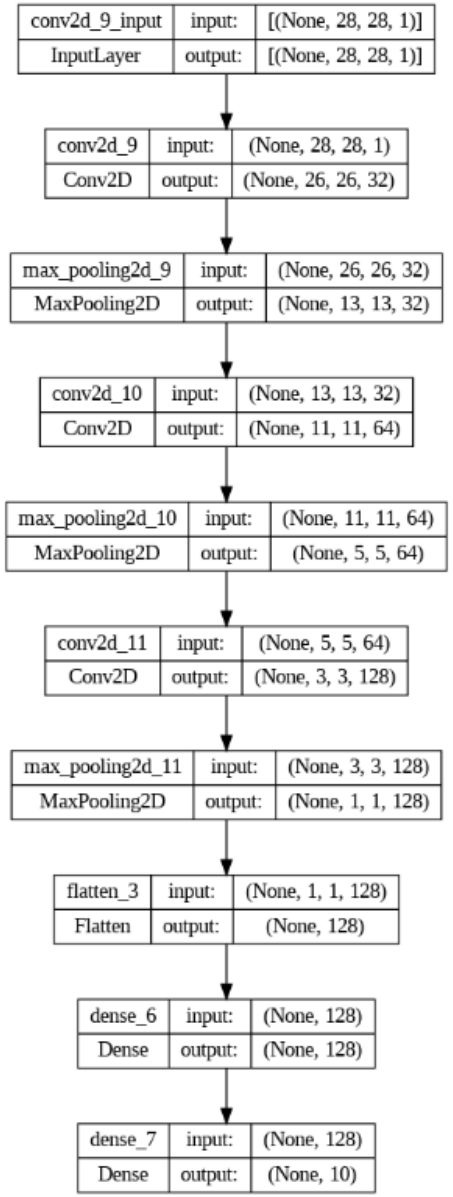
- 시각화 시켜서 구조 확인

```
!pip install visualkeras
import visualkeras
visualkeras.layered_view(model, legend=True)
```



- 이미지 파일로 시각화

```
from tensorflow.keras.utils import plot_model
plot_model(model, 'model.png', show_shapes=True)
```



Exercise #3

- Compile the model. 컴파일 모델
  - loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy’]
- Fit the model. 모델훈련
  - batch_size=64, epochs=20, verbose=1
- Prediction. 예측
  - y_pred = model.predict_classes(x_test)

Compiling Model 컴파일

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

- For one-hot encoded y, use categorical_crossentropy. 원핫 인코드된 y값은 카테고리컬 크로스엔트로피 사용
  - [1,0,0]
  - [0,1,0]
  - [0,0,1]

- For integer y, use sparse_categorical_crossentropy. 와이값이 정수이면 스팔스 카테고리컬 크로스엔트로피 사용
  - 1
  - 2
  - 3

Checkpoint 체크 포인트

```
# 체크 포인트 객체 생성
from keras.callbacks import ModelCheckpoint


ckpt_model="fashion_mnist.h5"
checkpoint = ModelCheckpoint(ckpt_model, monitor='val_accuracy',
verbose=1, save_best_only=True, mode='max')
```

Training Model 모델 훈련

```
# 모델 훈련

history = model.fit(x_train, y_train, validation_data=(x_test,
y_test), batch_size=64, epochs=20, callbacks=[checkpoint],
verbose=1)
```

Save and Load Model 모델저장 및 로딩

```
# 모델 저장할 때
model.save('fashion_mnist.h5')

# 나중에 사용할 때 (컴파일 할 필요 없음)
from keras.models import load_model

new_model = load_model('fashion_mnist.h5')
pred = np.argmax(new_model.predict(X_test), axis=1)
pred[0]
```

.h5와 .hdf5는 모두 Hierarchical Data Format (HDF) 파일의 확장자. Keras에서는 .h5 확장자를 사용

Predictions 예측

```
# 예측
y_pred = model.predict(x_test)
Y_pred
```

```
[[2.75874375e-11 4.99889921e-11 1.35116061e-14 ... 8.40097414e-07
  1.06408446e-13 9.99999046e-01]
 [1.32927099e-07 5.20787673e-16 9.99995828e-01 ... 7.99402046e-21
  5.36150748e-14 1.63414587e-20]
 [3.35109788e-18 1.00000000e+00 4.61758952e-15 ... 2.26489109e-32
  2.17630801e-15 3.76326379e-23]
 ...
 [6.43176539e-13 5.13549128e-18 4.13288878e-21 ... 6.51630449e-19
  1.00000000e+00 1.50515068e-16]
 [1.03569299e-18 1.00000000e+00 1.20533390e-14 ... 1.23134608e-28
  4.78780617e-19 1.02939507e-25]
 [7.25679683e-07 2.92154123e-09 5.40796539e-11 ... 1.37815054e-03
  7.21886521e-04 6.26847640e-09]]
```

```
y_pred = np.argmax(y_pred, axis=1)
y_pred
```

```
[9 2 1 ... 8 1 5]
```

Exercise #3

- Evaluate the model. 모델평가
  - Loss and accuracy score (model.evaluate(x_test, y_test))
  - Training and validation accuracy plot and Training and validation loss plot
  - Confusion matrix (
  - Classification report (classification_report(np.argmax(y_test, axis=1), y_pred))

- Predict with test data. 테스트 데이터 예측
  - Crop, resize, FLIP_LEFT_RIGHT
  - Array, negative image, reshape, normalization

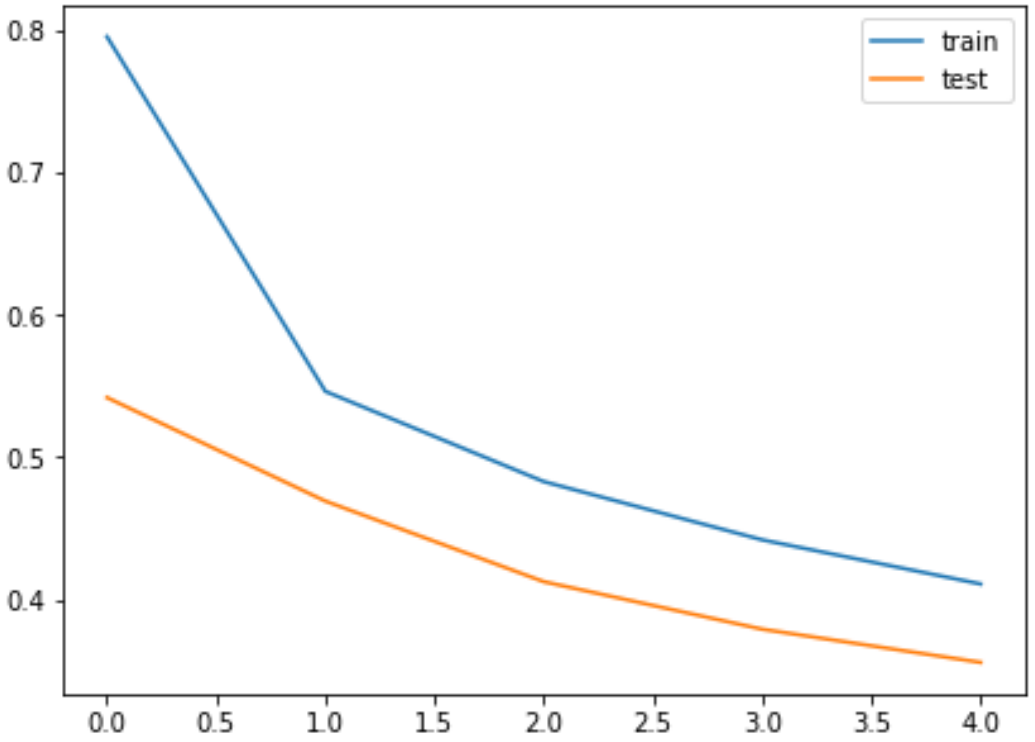
Evaluate Model 모델 평가

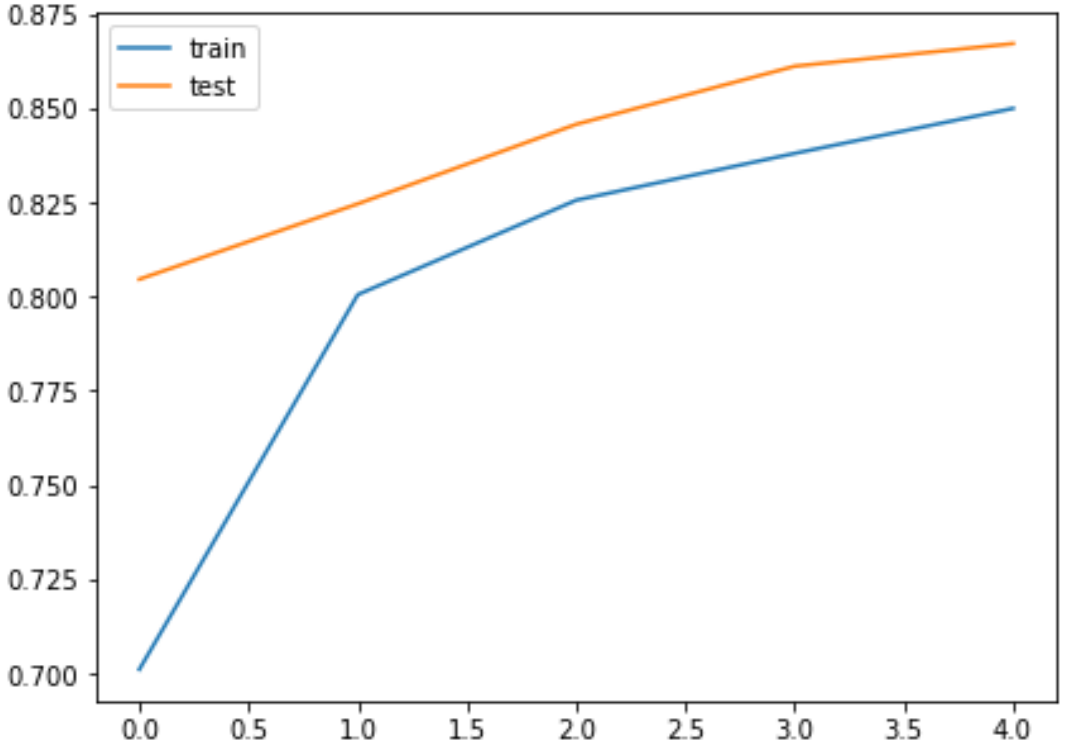
```
score = model.evaluate(x_test, y_test, verbose=0)
print(model.metrics_names[0], score[0])
print(model.metrics_names[1], score[1])

# 정확도 계산
from sklearn.metrics import accuracy_score
y_pred = np.argmax(model.predict(x_test), axis=1)
accuracy_score(np.argmax(y_test, axis=1), y_pred)
```

History Graph 히스토리 그래프

```
# 히스토리 그래프
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'],label='test')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.title('loss')
plt.legend()
plt.subplot(122)
plt.plot(history.history['accuracy'],label='train')
plt.plot(history.history['val_accuracy'],label='test')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.title('accuracy')
plt.legend()
plt.show()
```

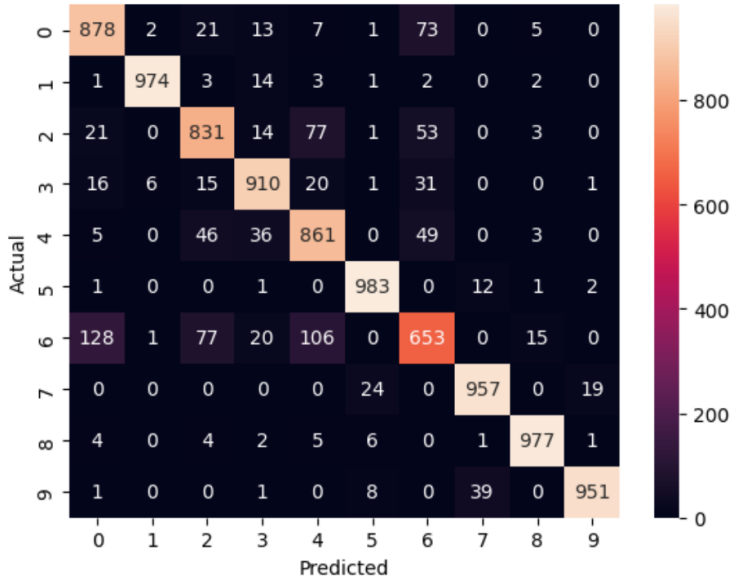
Loss and Accuracy Graph
손실 및 정확도 그래프





Confusion Matrix 혼동행렬

```
# 혼동행렬
from sklearn.metrics import confusion_matrix
cfm = confusion_matrix(np.argmax(y_test, axis=1), y_pred)
sns.heatmap(cfm, annot=True, fmt='g')
plt.show()
```



Classification Report 분류 리포트

```
# 분류리포트
from sklearn.metrics import classification_report
print(classification_report(np.argmax(y_test,axis=1),y_pred))
```

	precision	recall	f1-score	support
0	0.83	0.88	0.85	1000
1	0.99	0.97	0.98	1000
2	0.83	0.83	0.83	1000
3	0.90	0.91	0.91	1000
4	0.80	0.86	0.83	1000
5	0.96	0.98	0.97	1000
6	0.76	0.65	0.70	1000
7	0.95	0.96	0.95	1000
8	0.97	0.98	0.97	1000
9	0.98	0.95	0.96	1000
accuracy			0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000

테스트 데이터 준비

```
# 테스트 데이터 가져오기

from PIL import Image

ankleboot =
Image.open('/content/ankleboot.jpg').conve
rt('L')
```

```
# 사이즈 확인

ankleboot.size
```

```
# 이미지 자르고 사이즈 변환, 좌우반전

ankleboot = ankleboot.crop((100, 400, 600,
900))

ankleboot = ankleboot.resize((28,28))

ankleboot =
ankleboot.transpose(Image.FLIP_LEFT_RIGHT)
```

```
# 배열변환

ankleboot =
np.asarray(ankleboot)


plt.imshow(ankleboot)

plt.show()
```

```
# 이미지 반전

ankleboot = 255-ankleboot


plt.imshow(ankleboot)

plt.show()
```

테스트 데이터 전처리 및 예측

```
# 이미지 전처리
ankleboot = ankleboot.reshape(1,28,28,1)
ankleboot = ankleboot/255.0

# 테스트 데이터로 예측
label_names[np.argmax(model.predict(ankleboot), axis=1)[0]]
```

ChatGPT로 코드 개선 연습

- 다음 코드에 대한 개선된 코드 받아보기. 해설도 요청하세요

```
# 잘못된 코드
test_img = Image.open('seven.jpg').convert('L')
test_array = np.array(test_img).reshape(28, 28, 1)
prediction = model.predict(test_array)
```

- 출력 결과

❗ 문제점

- model.predict()는 입력으로 **4차원 텐서(batch 포함)**를 기대합니다.
  - 기대하는 입력 shape: (1, 28, 28, 1) ← 배치가 1장인 이미지
- 하지만 위 코드는 3차원: (28, 28, 1) → 그래서 ValueError 발생
- 또한 **정규화(0255 → 01)**가 되어 있지 않으면 예측 정확도가 떨어질 수 있습니다.

ChatGPT로 코드 개선 연습

- 다음 코드에 대한 개선된 코드 받아보기. 해설도 요청하세요

```
model = Sequential()
model.add(Dense(128, input_shape=(784,),
activation='relu'))
model.add(Dense(10, activation='softmax'))
```

- 출력 결과

❗ **한계점**

- 이미지의 공간 구조(위치, 패턴 등)를 고려하지 못함 → 성능이 CNN보다 떨어짐
- 이미지를 평탄화(Flatten)해야 함 → 전처리가 필요함