

초보자를 위한 딥러닝의 이해

1. 머신러닝 훈련하기

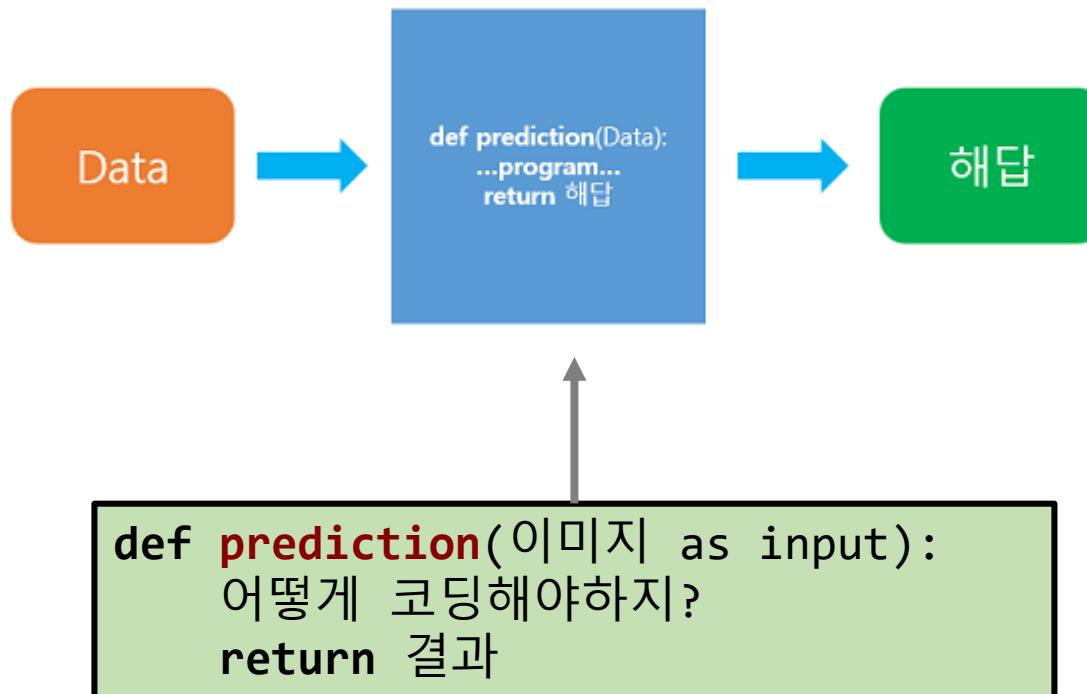
Machine Learning이 아닌 접근 방법의 한계

- 기존의 프로그래밍 접근 방법
 - ✓ Ex) 주어진 사진으로부터 고양이 사진인지 강아지 사진인지 판별하는 일.



Machine Learning이 아닌 접근 방법의 한계

- 전통적인 코딩 방식은 공통된 명확한 특징을 잡아내는 것이 쉽지 않습니다



머신 러닝

- 기존 프로그래밍의 한계에 대한 해결책

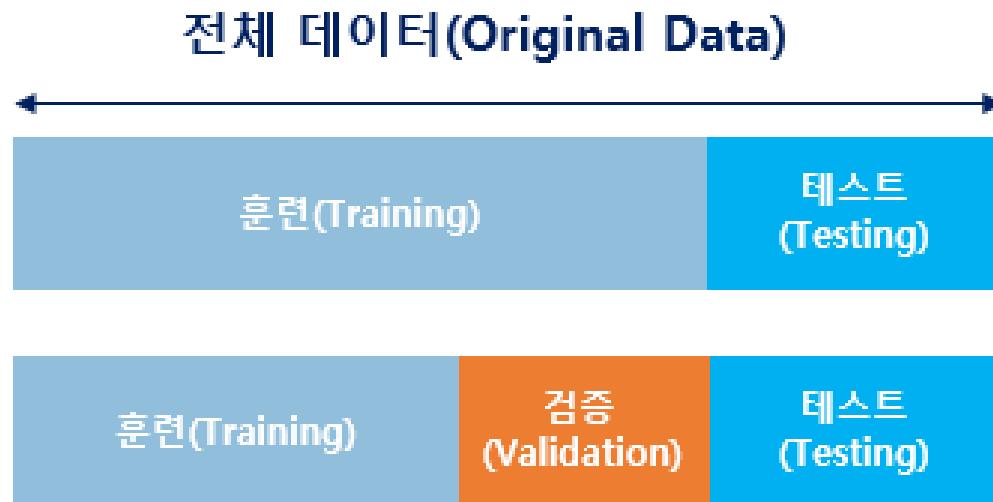


✓ 학습(Train)

- 머신 러닝은 주어진 데이터로부터 규칙성 또는 패턴을 찾는 것에 초점이 맞추어져 있다.
- 주어진 데이터로부터 규칙성을 찾는 과정을 우리는 학습(training)이라고 합니다.
- 일단 규칙성을 발견해내면, 그 후에 들어오는 새로운 데이터에 대해서 발견한 규칙성을 기준으로 정답을 찾아내는데, 이는 기존의 프로그래밍 방식으로 접근하기 어려웠던 문제의 해결책이 되기도 합니다

머신 러닝 모델을 위한 데이터 분할

- 실제 모델 훈련 및 평가하기
 - ✓ 데이터를 훈련용, 검증용, 테스트용 이렇게 세 가지로 분리
 - 테스트 데이터는 20%~30% 정도
 - 검증용 데이터 약 10%정도는 과적합을 판단하거나 하이퍼파라미터의 조정을 위한 용도



머신 러닝 모델의 평가

- 사용자가 직접 적용(지정)해야 하는 변수

- ✓ 하이퍼파라미터(초매개변수)

- 값에 따라서 모델의 성능에 영향을 주는 변수
 - 학습률(Learning rate), 은닉층 수, 뉴런 수, Batch size 등

- ✓ 매개변수는 (Random)

- 가중치와 편향과 같은 학습을 통해 바뀌어져가는 변수

- ✓ 하이퍼파라미터 최적화 (Tuning)

- 모델 최적화를 통해서 진행

분류(Classification)와 회귀(Regression)

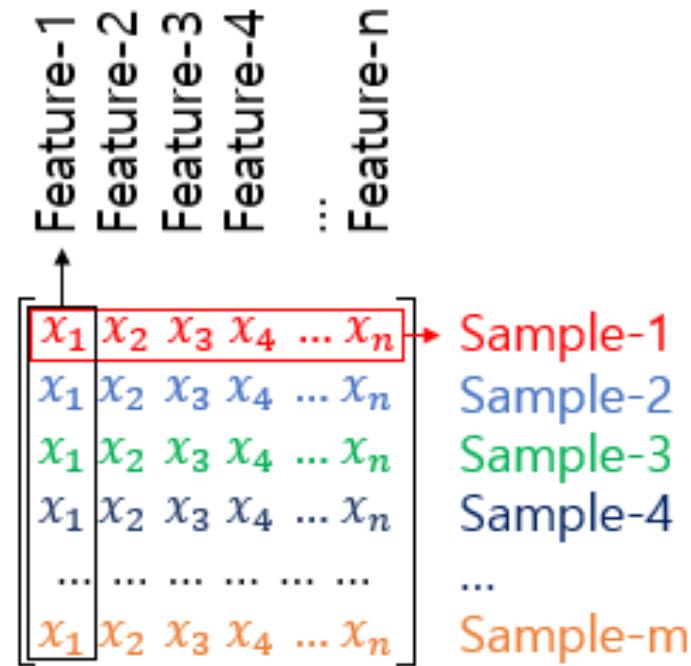
- 회귀는 연속된 값을 결과로 예측에 사용한다.
 - ✓ 선형 회귀(Lineare Regression) ~ 회귀 문제
 - ✓ 로지스틱 회귀(Logistic Rgression) ~ 주로 이진 분류 문제를 잘 다룬다.
- 분류
 - ✓ 이진 분류 문제(Binary Classification)
 - 이진 분류는 주어진 입력에 대해서 둘 중 하나의 답을 정하는 문제
 - 시험 성적에 대해서 합격, 불합격; 메일로부터 정상 메일, 스팸 메일인지를 판단
 - ✓ 다중 클래스 분류(Multi-class Classification)

지도 학습과 비지도 학습

- 지도 학습(Supervised Learning)
 - ✓ 레이블(Label)이라는 정답과 함께 학습
- 비지도 학습(Unsupervised Learning)
 - ✓ 비지도 학습은 레이블이 없이 학습하는 것

샘플(Sample)과 특성(Feature)

- 샘플은 데이터의 개수가 m개
- 특성은 독립변수(입력)가 n개



혼동 행렬(Confusion Matrix)

- 정확도
 - ✓ 머신 러닝에서는 맞춘 문제수를 전체 문제수로 나눈 값을 정확도(Accuracy)
 - 하지만 정확도는 맞춘 결과와 틀린 결과에 대한 세부적인 내용을 알려주지는 않습니다. 이를 위해서 사용하는 것이 혼동 행렬(Confusion Matrix)입니다.
- 혼동행렬

		실제 정답	
		True	False
분류 결과	True	True Positive	False Positive
	False	False Negative	True Negative

정밀도(Precision)와 재현율(Recall)

- TP(True Positive), TN(True Negative), FP(False Postivie), FN(False Negative)
- True는 정답을 맞춘 경우고 False는 정답을 맞추지 못한 경우
- Positive와 Negative는 각각 제시했던 정답
- TP는 양성(Postive)이라고 대답하였고 실제로 양성이라서 정답을 맞춘 경우
- TN은 음성(Negative)이라고 대답하였는데 실제로 음성이라서 정답을 맞춘 경우

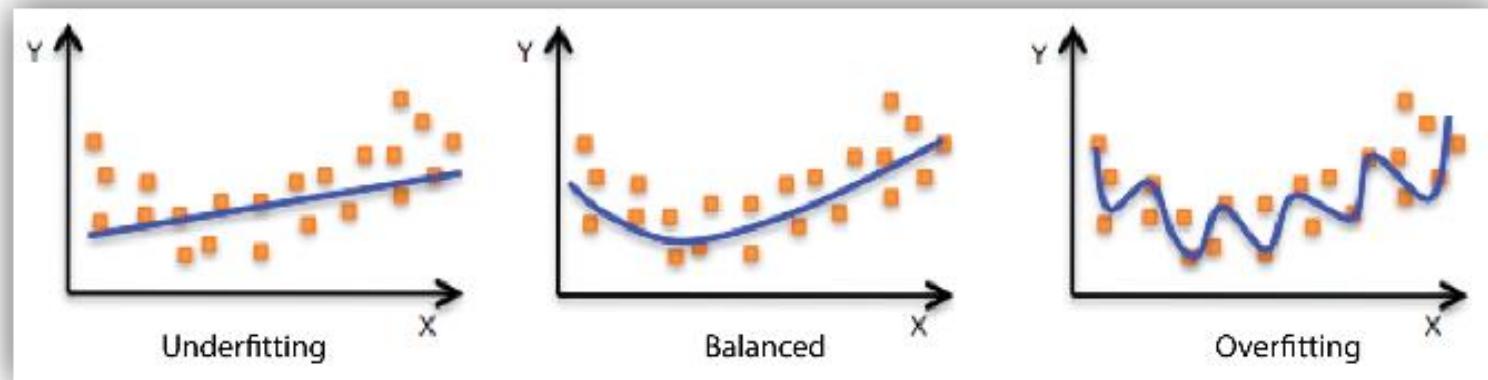
$$\text{정확도} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{정밀도} = \frac{TP}{TP + FP}$$

$$\text{재현율} = \frac{TP}{TP + FN}$$

과적합(Overfitting)과 과소 적합(Underfitting)

- 과적합(Overfitting)이란 훈련 데이터를 과하게 학습한 경우

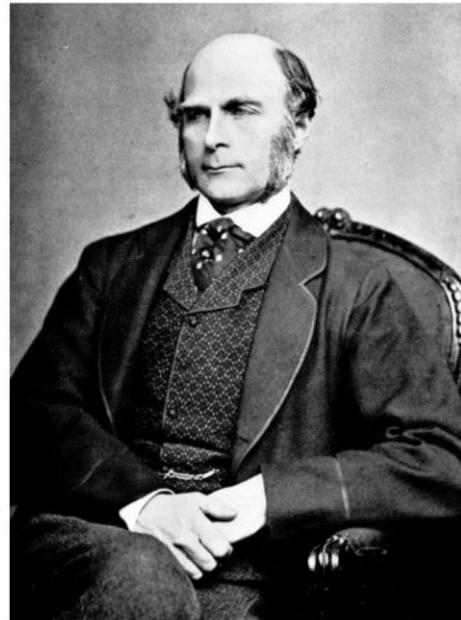


선형회귀 소개 및 실습

회귀의 목적함수로는 무엇을 사용하는가?

회귀(Regression) 소개(1)

- 회귀의 역사
 - ✓ 영국의 통계학자 갈تون(Galton)의 유전적 특성중에 부모와 자식의 키 관계
 - “사람의 키는 평균 키로 회귀(Regression)하려는 경향을 가진다는 자연의 법칙이 있다”
 - ✓ 회귀 분석은
 - 데이터 값이 평균과 같은 일정한 값으로 돌아가려는 경향을 이용한 통계학 기법

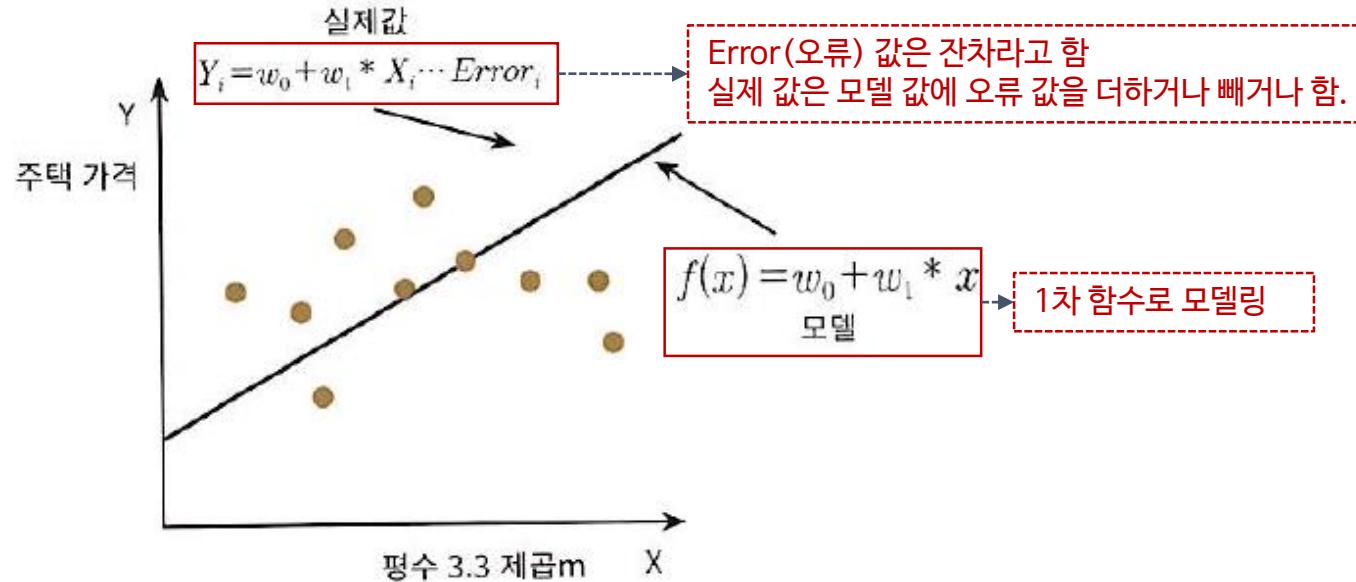


Sir Francis Galton (1822 ~ 1911)

회귀(Regression) 소개(2)

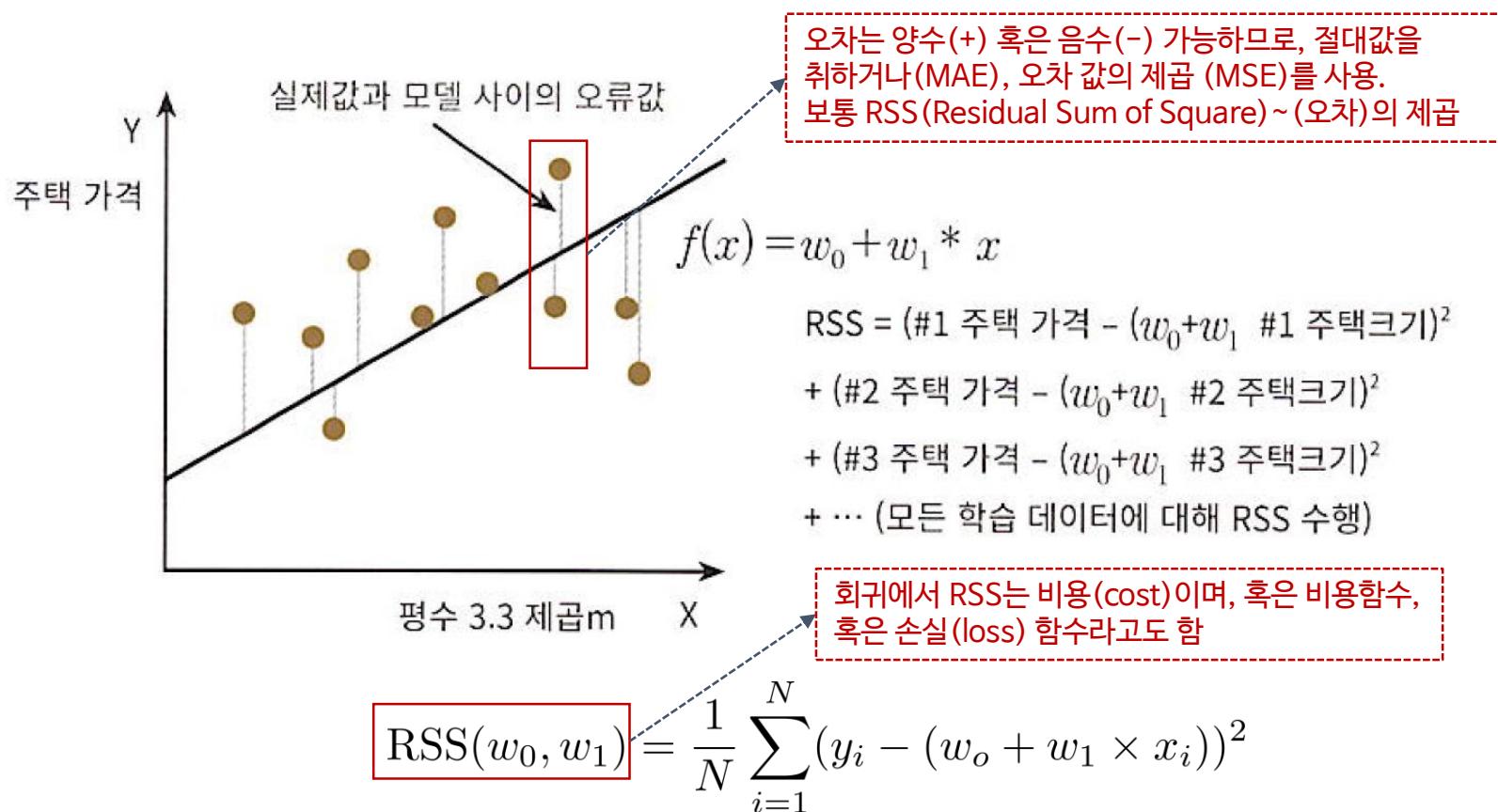
- 지도학습은 2가지 유형으로 나눔
 - ✓ 회귀는 연속적인 숫자 값
 - ✓ 분류는 예측값이 카테고리와 같은 이산형 클래스 값
- 선형회귀
 - ✓ 가장 많이 사용되며, 실제 값과 예측 값의 차이(오류의 제곱)를 최소화하는 직선형 회귀선을 최적화하는 방식
 - 단순 선형회귀는 1개의 독립변수, 1개의 종속변수.
 - (예) 주택 가격이 주택의 크기로만 결정된다고 해보면,

회귀(Regression) 소개(4)



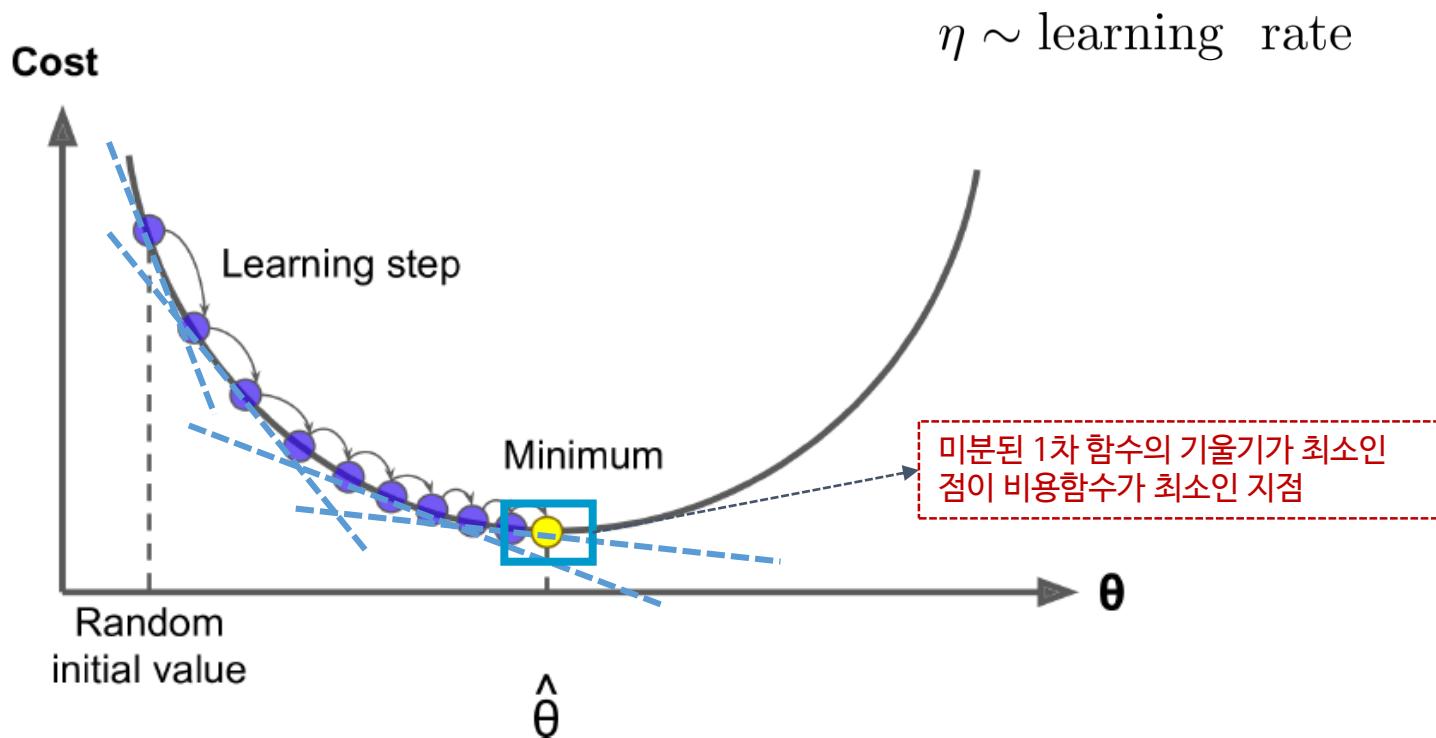
회귀(Regression) 소개(7)

최적의 회귀 모델은 전체 데이터의 잔차(오차) 합이 최소가 되는 모델을 만드는 것임!



비용 최소화 _ 경사 하강법(1)

- 경사 하강법 (Gradient Descent)



비용 최소화 _ 경사 하강법(2)

$$\text{RSS}(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N (y_i - (w_0 + w_1 \times x_i))^2$$

$$\frac{\partial \text{RSS}(w_0, w_1)}{\partial w_1} = \frac{2}{N} \sum_{i=1}^N -x_i \times (y_i - (w_0 + w_1 \times x_i)) = -\frac{2}{N} \sum_{i=1}^N x_i * (\text{real}_i - \text{pred}_i)$$

$$\frac{\partial \text{RSS}(w_0, w_1)}{\partial w_0} = \frac{2}{N} \sum_{i=1}^N -(y_i - (w_0 + w_1 \times x_i)) = -\frac{2}{N} \sum_{i=1}^N (\text{real}_i - \text{pred}_i)$$

$$w = w - \boxed{\eta \frac{2}{N} \sum_{i=1}^N (\text{real}_i - \text{pred}_i)}$$

학습률 도입 $\eta \sim \text{learning rate}$

비용 최소화 _ 경사 하강법(3)

- 데이터를 학습하는 방법으로 배치(Batch)
 - ✓ 매 경사 하강법 스텝에서 전체 훈련 세트 (X)에 대해 계산
 - ✓ 전체 입력 데이터를 훈련에 사용해서 큰 메모리 필요, 계산 시간 오래 걸림

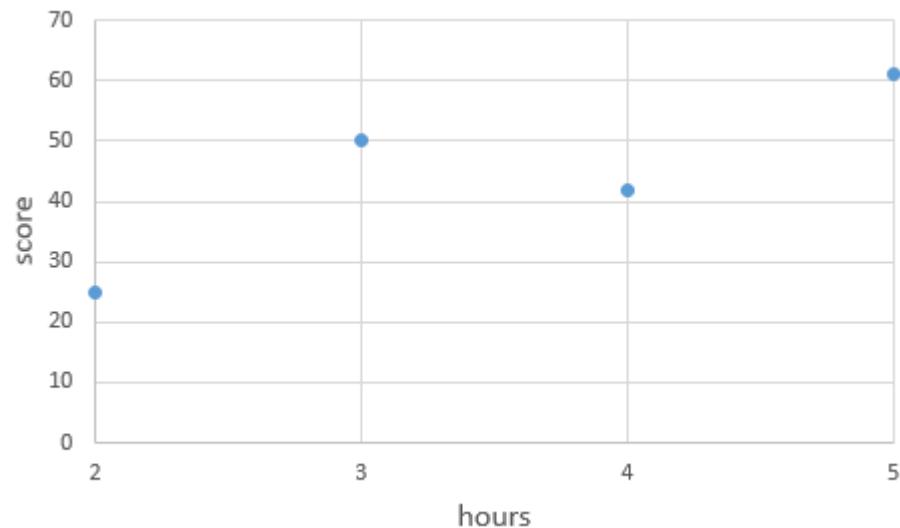
$$\frac{\partial}{\partial \theta_j} \text{MSE}(\theta) = \frac{2}{m} \sum_{i=1}^m (\theta^T \mathbf{x}^{(i)} - y^{(i)}) x_j^{(i)}$$

$$\theta^{(\text{next step})} = \theta - \eta \nabla_{\theta} \text{MSE}(\theta)$$

가설(Hypothesis) 세우기

- 어떤 학생의 공부 시간에 따라서 얻은 점수 데이터

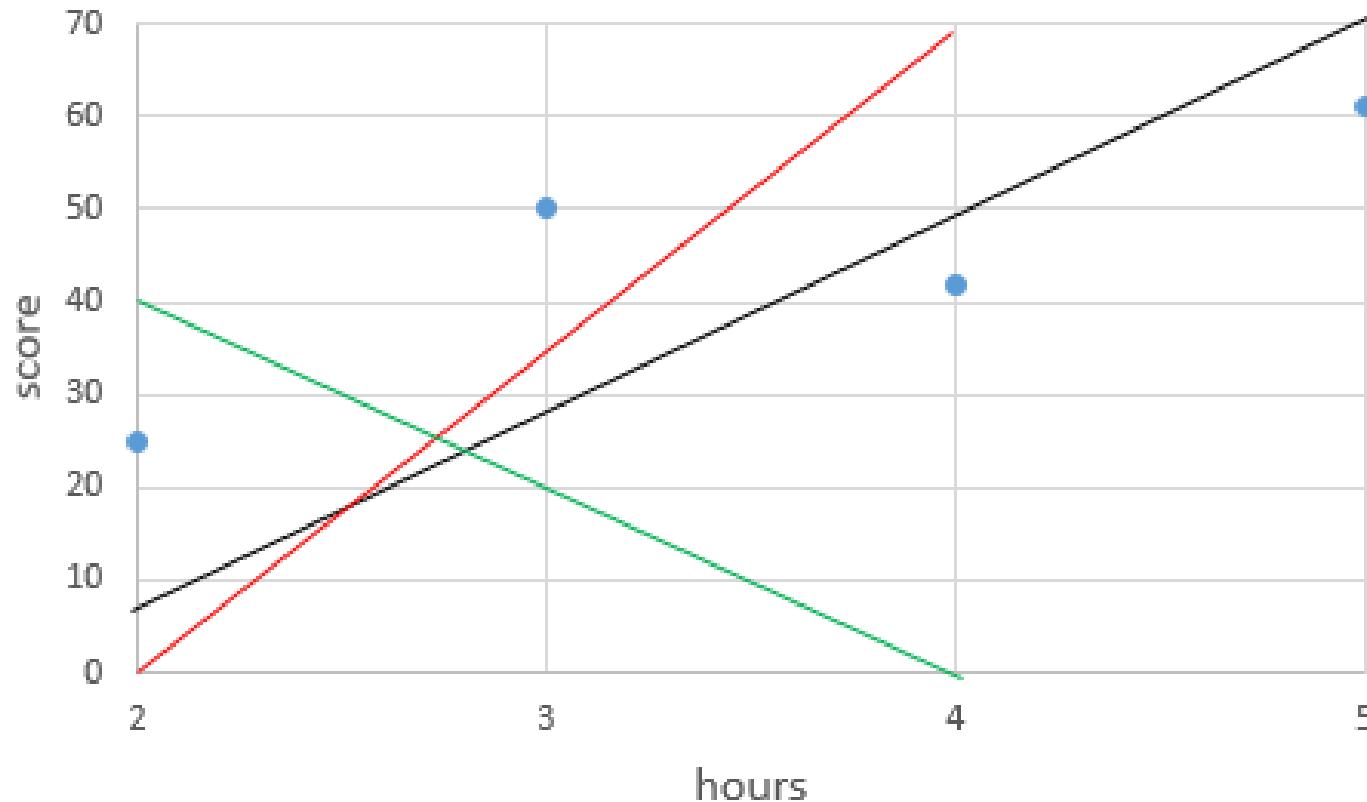
hours(x)	score(y)
2	25
3	50
4	42
5	61



- 예측을 위한 선형 가설을 세우자

$$H(x) = Wx + b$$

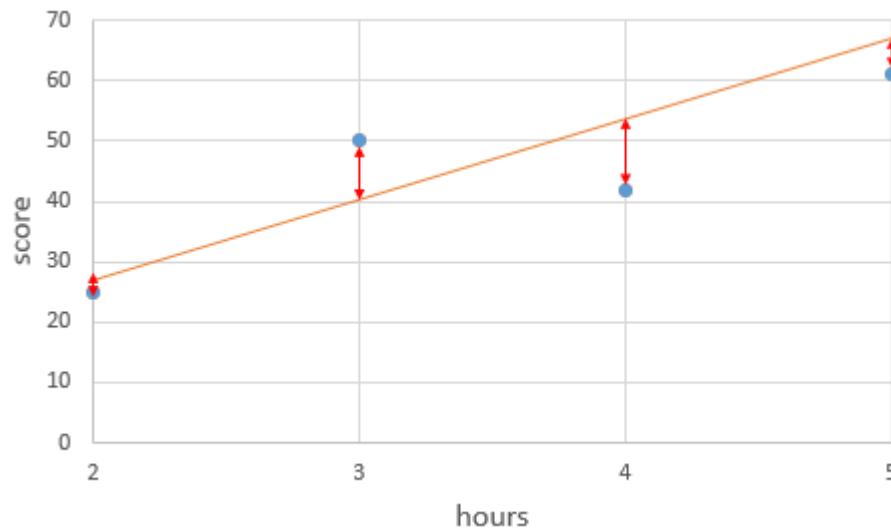
Which hypothesis is better?



3. 비용 함수(Cost function) : 평균 제곱 오차(MSE)

- 머신러닝에서는 아래의 3가지를 같은 표현으로 사용한다.
 - ✓ 함수(Objective function)=비용 함수(Cost function)=손실 함수(Loss function)
- 실제값과 예측값에 대한 오차로 평균 제곱 오차(Mean Squared Error)

$$y = H(x) = 13x + 1$$



비용함수의 최적화

- MSE 오차의 크기를 최소화를 위한 비용 함수(MSE)를 구하자
 - ✓ 모든 오차를 제곱하여 더하자.

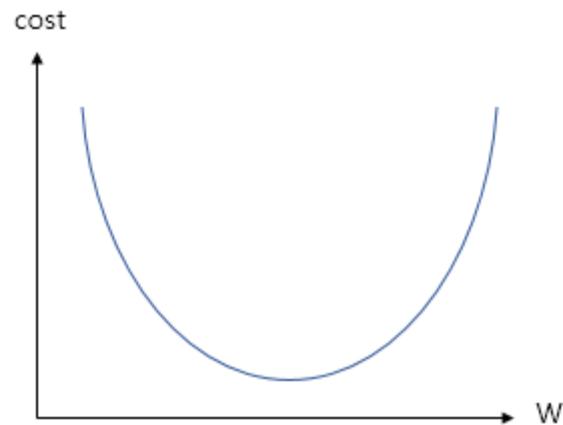
hours(x)	2	3	4	5
실제값	25	50	42	61
예측값	27	40	53	66
오차	-2	10	-7	-5

$$\sum_{i=1}^n [y^{(i)} - H(x^{(i)})]^2 = (-2)^2 + 10^2 + (-7)^2 + (-5)^2 = 178$$

$$\frac{1}{n} \sum_{i=1}^n [y^{(i)} - H(x^{(i)})]^2 = 178/4 = 44.5$$

비용함수의 최적화

$$cost(W, b) = \frac{1}{n} \sum_{i=1}^n [y^{(i)} - H(x^{(i)})]^2$$

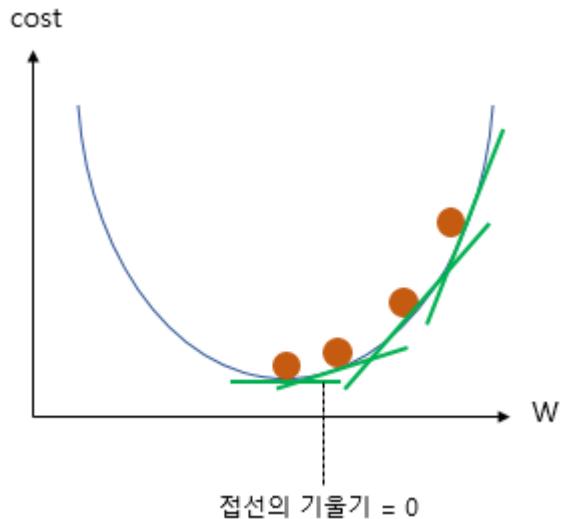


$W, b \rightarrow \text{minimize } cost(W, b)$

4. 옵티마이저 : 경사하강법(Gradient Descent)

- 옵티마이저(Optimizer) 또는 최적화 알고리즘

- 머신 러닝 학습은 결국 비용 함수를 최소화하는 가중치, 편향을 찾기 위한 작업을 수행



$$cost(W, b) = \frac{1}{n} \sum_{i=1}^n [y^{(i)} - H(x^{(i)})]^2$$

$$W, b \rightarrow \text{minimize } cost(W, b)$$

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

How to minimize cost?

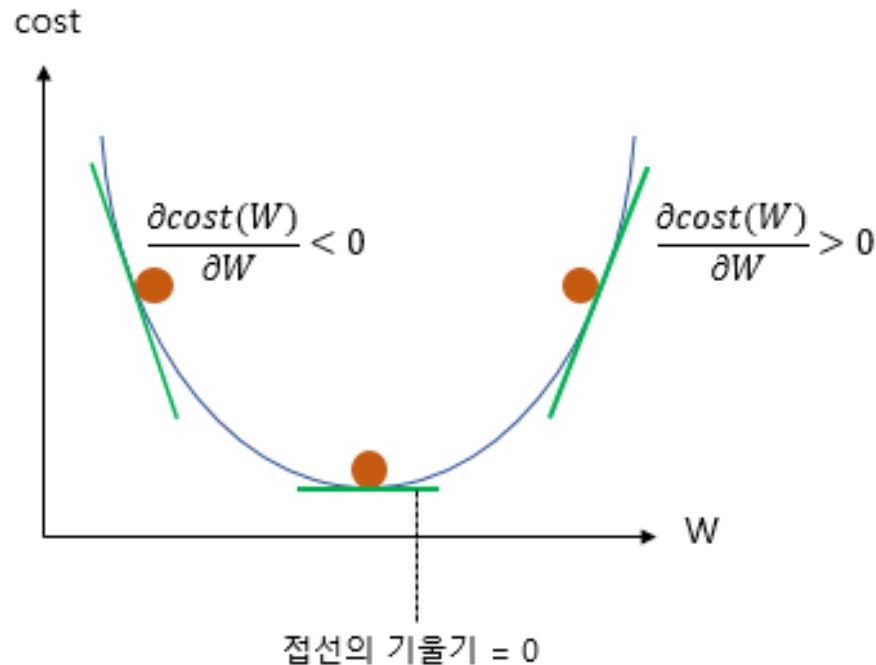
$$cost(W) = \frac{1}{m} \sum_{i=1}^m (Wx_i - y_i)^2$$



Gradient Descent

$$cost(W, b) = \frac{1}{2m} \sum_{i=1}^m (H(x_i) - y_i)^2$$

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$



Formal definition : chain rule

$$W := W - \alpha \frac{\partial}{\partial W} \frac{1}{2m} \sum_{i=1}^m (W(x_i) - y_i)^2$$

$$W := W - \alpha \frac{1}{2m} \sum_{i=1}^m 2(W(x_i) - y_i)$$

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^m (W(x_i) - y_i)$$

Gradient descent algorithm : Learning Rate

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^m (W(x_i) - y_i)x_i$$

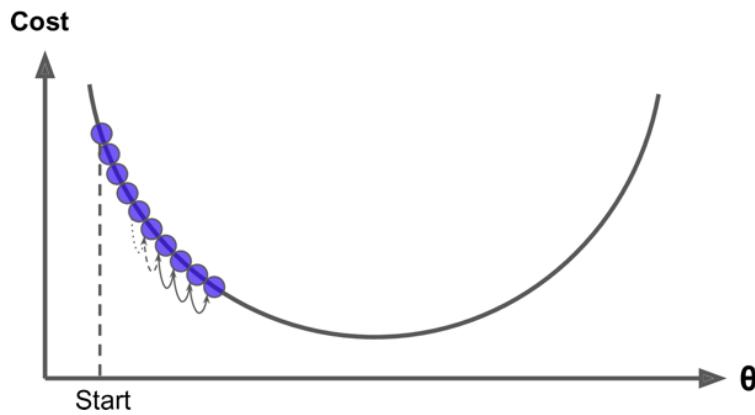
$$\text{cost}(w, b) = \sum_{i=0}^{M-1} (f(h(x^i)) - y_-^i)^2$$

$$\frac{\partial \text{cost}}{\partial w} = \frac{\partial \text{cost}}{\partial y} \cdot \frac{\partial y}{\partial f} \cdot \frac{\partial f}{\partial h} \cdot \frac{\partial h}{\partial w}$$

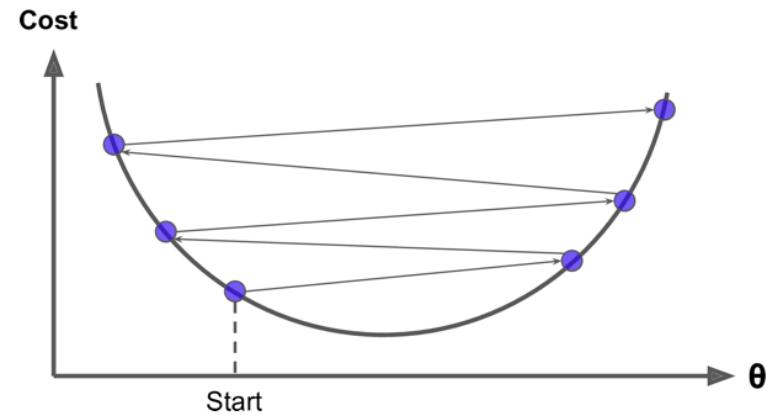
$$w_{\text{updated}} \approx w - \alpha \cdot \frac{\partial \text{cost}}{\partial w} \quad \alpha \text{ ia the learning rate.}$$

Learning Rate

$$w_{\text{updated}} \approx w - \alpha \cdot \frac{\partial \text{cost}}{\partial w}$$



a) too small



a) too big

Lab01 Linear Regression

케라스로 구현하는 선형 회귀

(lab) 케라스로 구현하는 선형 회귀

- 케라스로 모델을 만드는 기본적인 형식
- `model = keras.models.Sequential()`
 - ✓ Sequential로 모델을 이라는 이름을 만들고
- `model.add(keras.layers.Dense(1, input_dim=1))`
 - ✓ add를 통해서 필요한 사항을 추가해 나간다.
 - ✓ 첫 인자 1은 출력의 차원
 - ✓ 두 번째 인자 input_dim은 입력의 차원을 정의

케라스로 구현하는 선형 회귀

```
import tensorflow as tf
import numpy as np

print(tf.__version__)

import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras import optimizers

X=np.array([1,2,3,4,5,6,7,8,9]) # 공부하는 시간
y=np.array([11,22,33,44,53,66,77,87,95]) # 각 공부하는 시간에 맵핑되는 성적

model = Sequential()
model.add(Dense(1, input_dim=1, activation='linear'))

# sgd는 경사 하강법을 의미. 학습률(learning rate, lr)은 0.01.
sgd = optimizers.SGD(lr=0.01)

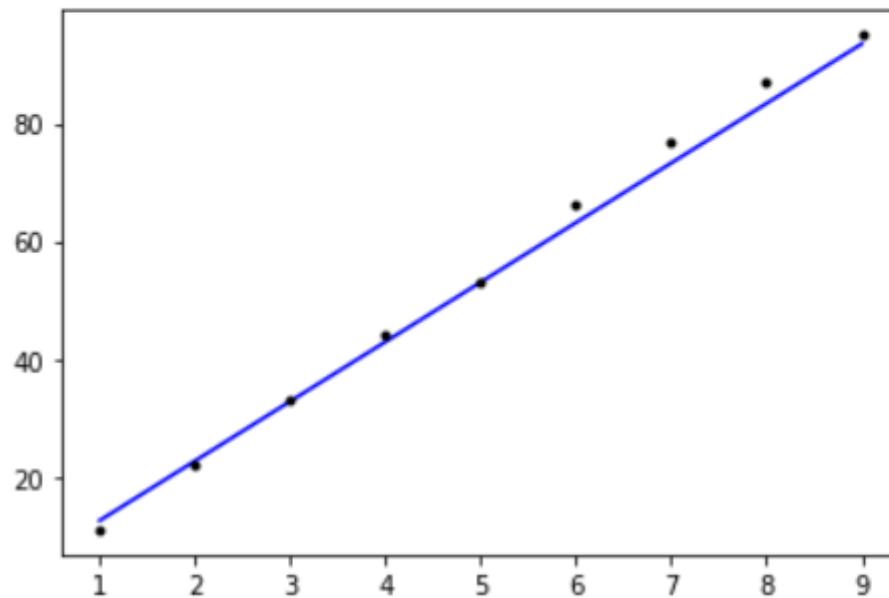
# 손실 함수(Loss function)은 평균제곱오차 mse를 사용합니다.
model.compile(optimizer=sgd, loss='mse', metrics=['mse'])

# 주어진 X와 y데이터에 대해서 오차를 최소화하는 작업을 300번 시도합니다.
history=model.fit(X,y, batch_size=1, epochs=10, shuffle=False)
```

케라스로 구현하는 선형 회귀

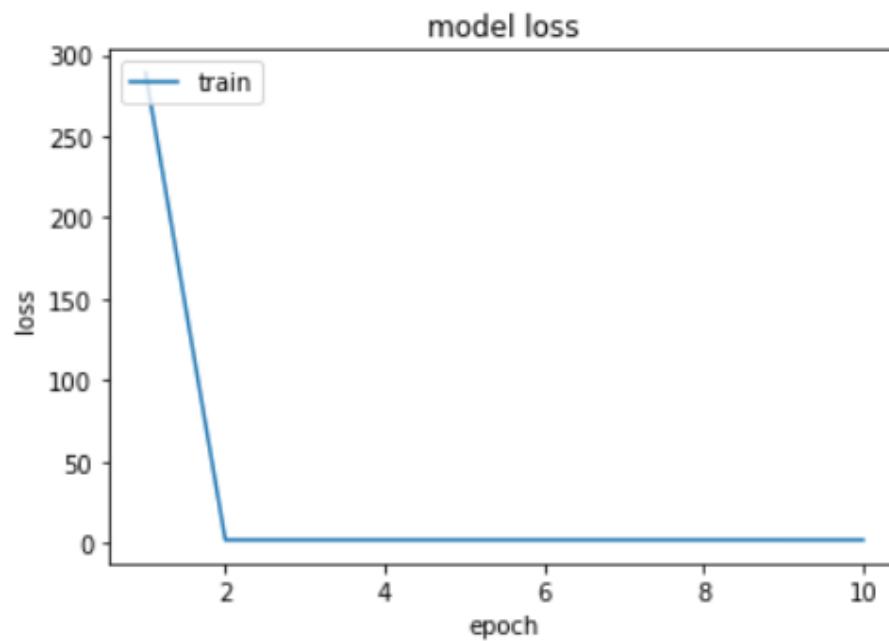
```
%matplotlib inline  
import matplotlib.pyplot as plt  
plt.plot(X, model.predict(X), 'b', X,y, 'k.')
```

```
[<matplotlib.lines.Line2D at 0x2120cdc3688>,  
<matplotlib.lines.Line2D at 0x2120cdc37c8>]
```



(lab) 100번 이포크에 대하여 손실값을 그려라.

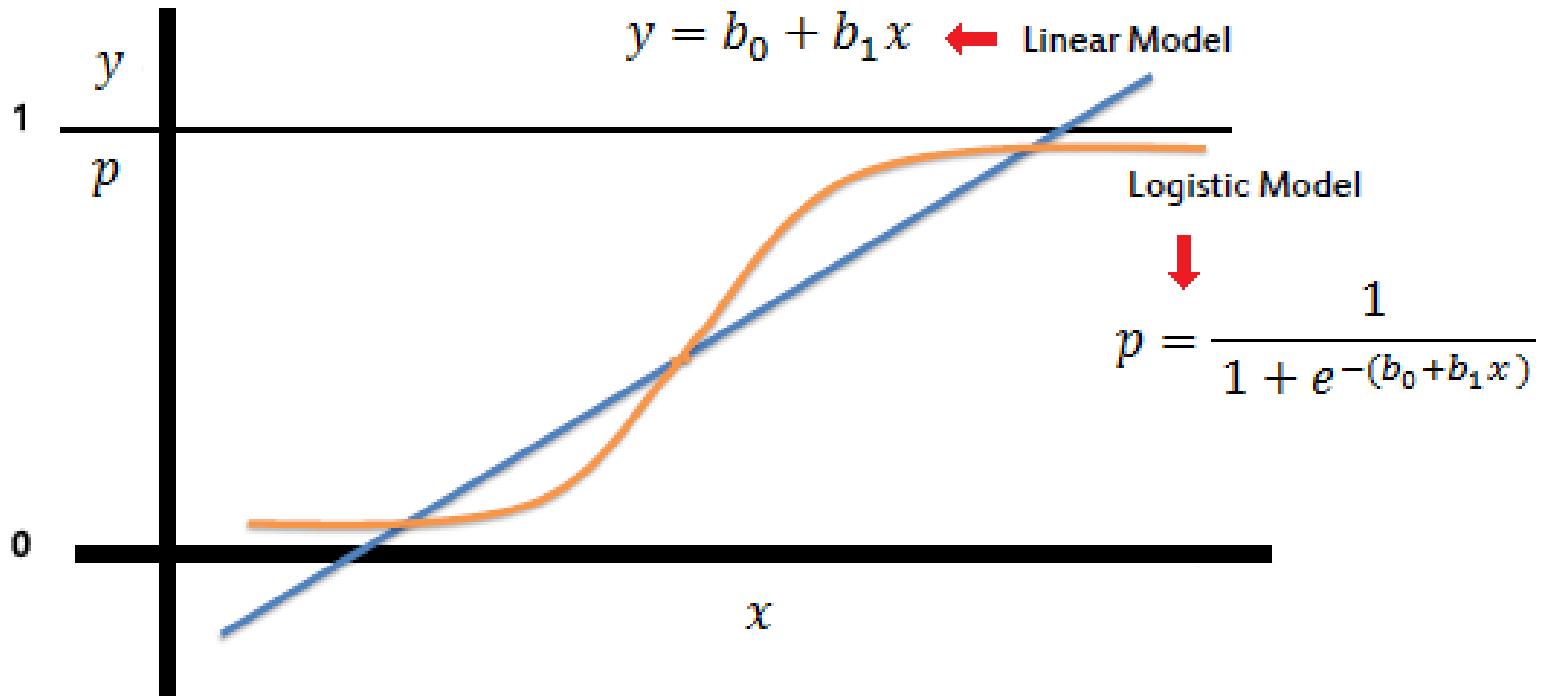
```
epochs = range(1, len(history.history['mse']) + 1)
plt.plot(epochs, history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train'], loc='upper left')
plt.show()
```



비선형 회귀(Regression)

로지스틱 회귀(Logistic Regression)

선형 모델과 비선형(로지스틱) 회귀 소개



로지스틱 회귀(Logistic Regression)

- **로지스틱 회귀 개요**
 - 일상 속 많은 문제 중에서는 두 개의 선택지 중에서 정답을 고르는 문제가 많다.
 - 예를 들어 시험을 봤는데 이 시험 점수가 합격인지 불합격인지가 궁금할 수도 있고,
 - 어떤 메일을 받았을 때 이게 정상 메일인지 스팸 메일인지를 분류하는 문제
- ✓ **둘 중 하나를 결정하는 문제를 이진 분류(Binary Classification)라고 합니다.**
 - 이런 문제를 풀기 위한 알고리즘으로 로지스틱 회귀(Logistic Regression)가 있다.

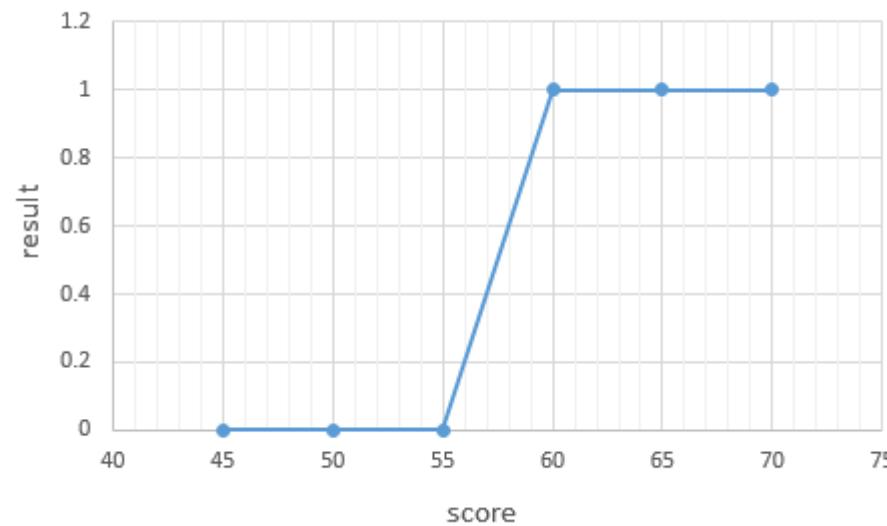
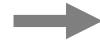
이진 분류(Binary Classification)

- 예제

- ✓ 학생들이 시험 성적에 따라서 합격, 불합격이 기재된 데이터가 있다고 가정

- 시험 성적이 x 라면, 합불 결과는 y 입니다.
 - 이 시험의 커트라인은 공개되지 않았는데 이 데이터로부터 특정 점수를 얻었을 때의 합격, 불합격 여부를 판정하는 모델을 만들고자 합시다

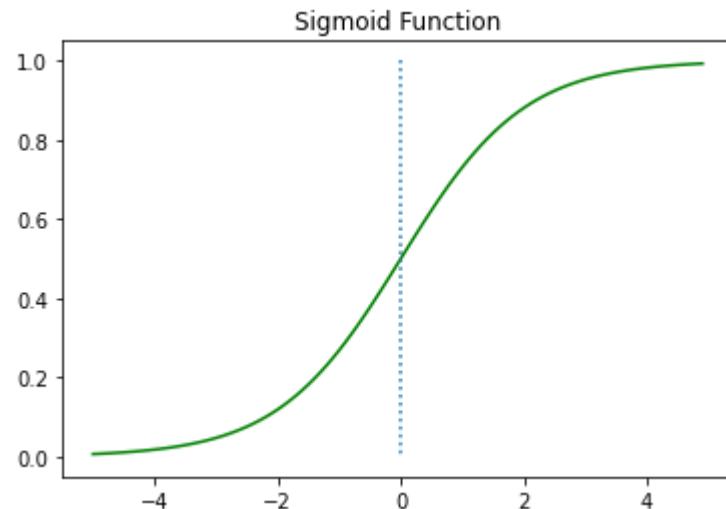
score(x)	result(y)
45	불합격
50	불합격
55	불합격
60	합격
65	합격
70	합격



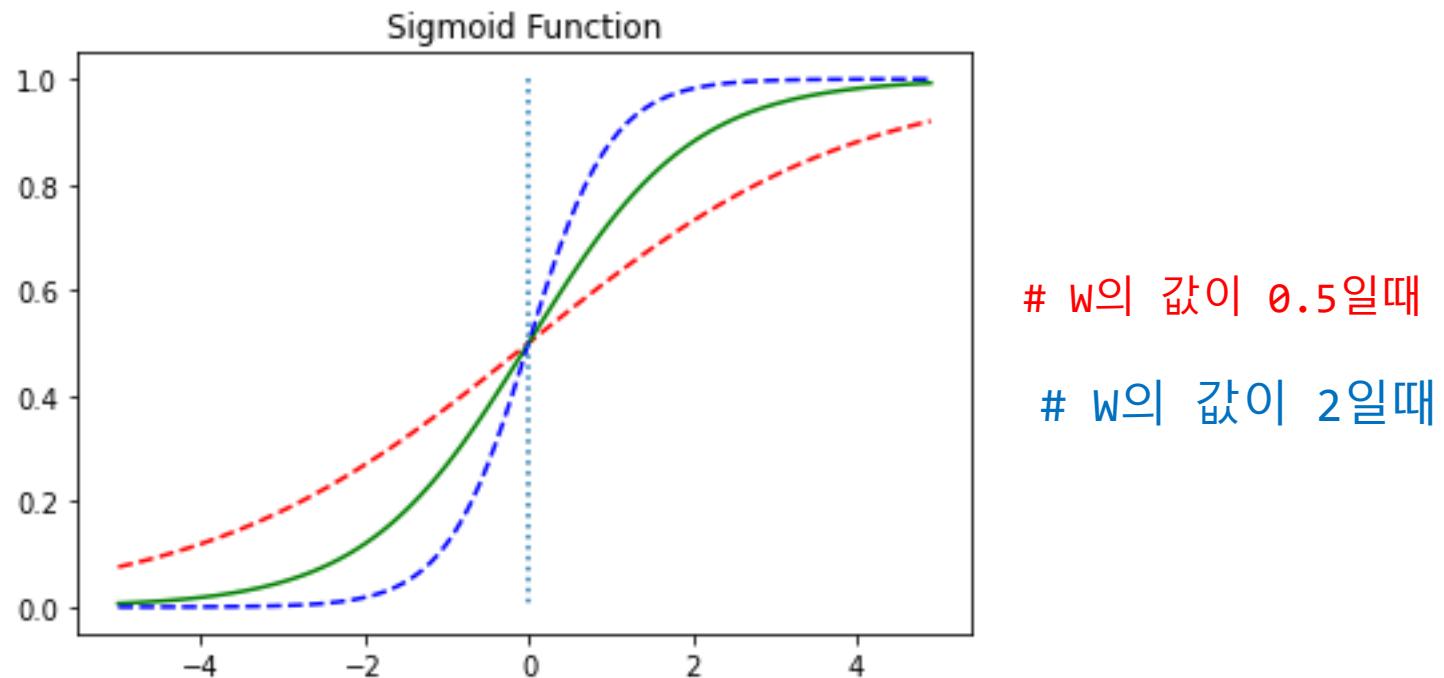
시그모이드 함수(Sigmoid function)

$$H(X) = \frac{1}{1+e^{-(Wx+b)}} = \text{sigmoid}(Wx + b) = \sigma(Wx + b)$$

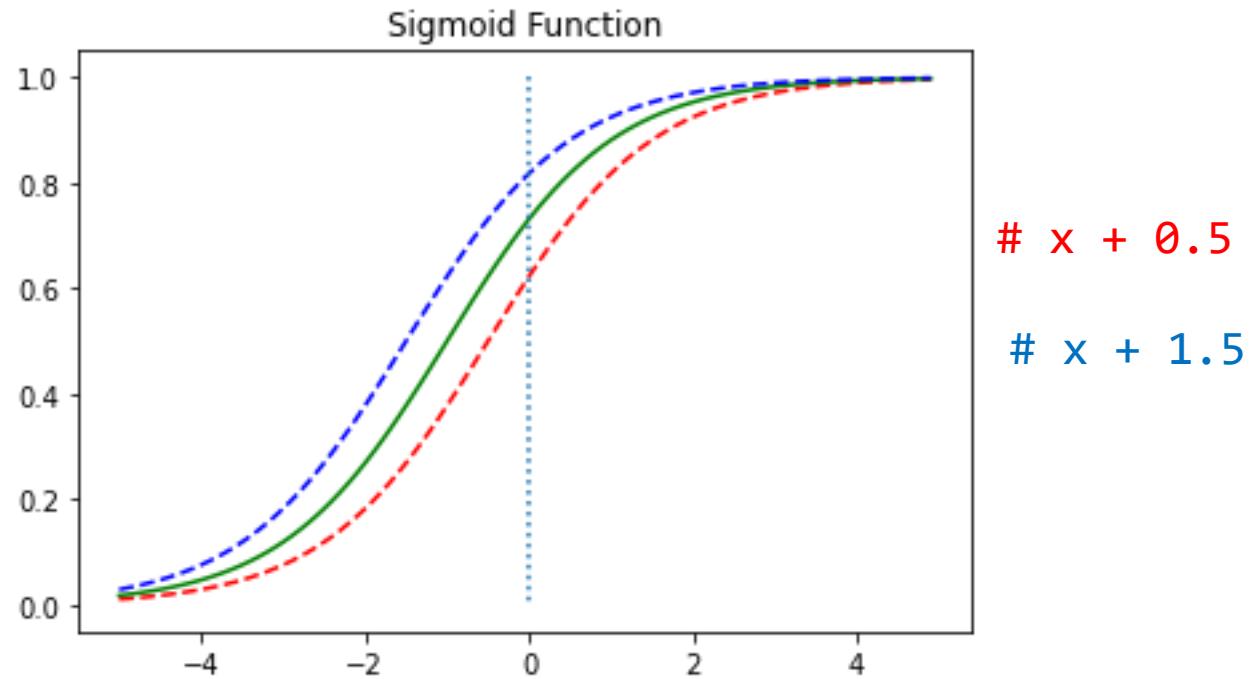
여기서 e($e=2.718281\dots$)는 자연 상수입니다.



시그모이드 함수(Sigmoid function)



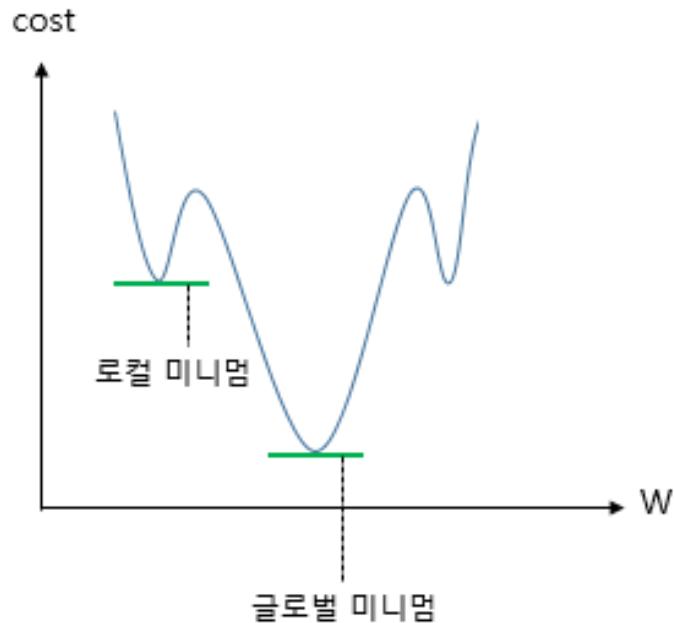
시그모이드 함수(Sigmoid function)



로지스틱 비용 함수(Cost function)

- **로컬미니멈의 위험**

- 로지스틱 회귀 또한 경사 하강법을 사용하여 가중치를 찾아내지만, 비용 함수로는 평균 제곱 오차를 사용하지 않습니다.
- 이유는 시그모이드 함수에 비용 함수를 평균 제곱 오차로 하여 그래프를 그리면 다음과 비슷한 형태가 되기 때문입니다.



로지스틱 비용 함수(Cost function)

- 목적 함수(objective function)

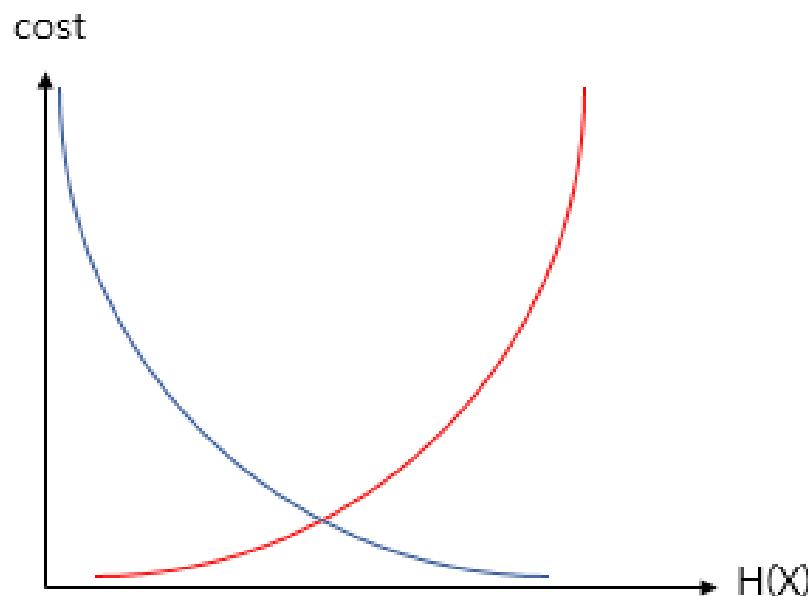
$$J(W) = \frac{1}{n} \sum_{i=1}^n \text{cost} \left(H(x^{(i)}), y^{(i)} \right)$$

$$H(X) = \frac{1}{1+e^{-(Wx+b)}} = \sigma(Wx + b)$$

로지스틱 비용 함수(Cost function)

if $y = 1 \rightarrow \text{cost}(H(x), y) = -\log(H(x))$

if $y = 0 \rightarrow \text{cost}(H(x), y) = -\log(1 - H(x))$



로지스틱 함수의 비용 및 목적함수

- 크로스 엔트로피 함수 (Cross Entropy)
 - ✓ 로지스틱 회귀에서 찾아낸 비용함수를 말한다.

$$\text{cost}(H(x), y) = -[y \log H(x) + (1 - y) \log(1 - H(x))]$$

$$J(W) = -\frac{1}{n} \sum_{i=1}^n [y^{(i)} \log H(x^{(i)}) + (1 - y^{(i)}) \log(1 - H(x^{(i)}))]$$

(lab02) 케라스로 구현하는 로지스틱 회귀

```
import numpy as np
from tensorflow.keras.models import Sequential # 케라스의 Sequential()을 임포트
from tensorflow.keras.layers import Dense # 케라스의 Dense()를 임포트
from tensorflow.keras import optimizers # 케라스의 옵티마이저를 임포트

X=np.array([-50, -40, -30, -20, -10, -5, 0, 5, 10, 20, 30, 40, 50])
y=np.array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1]) #숫자 10부터 1

model=Sequential()
model.add(Dense(1, input_dim=1, activation='sigmoid'))
sgd=optimizers.SGD(lr=0.01)
model.compile(optimizer=sgd ,loss='binary_crossentropy',metrics=['binary_accuracy'])
# 옵티마이저는 경사하강법 sgd를 사용합니다.
# 손실 함수(Loss function)는 binary_crossentropy(이진 크로스 엔트로피)를 사용합니다.
model.fit(X,y, batch_size=1, epochs=2, shuffle=False)
# 주어진 X와 y데이터에 대해서 오차를 최소화하는 작업을 200번 시도합니다.
```

Epoch 1/2

13/13 [=====] - 0s 3ms/step - loss: 0.5466 - binary_accuracy: 0.9231

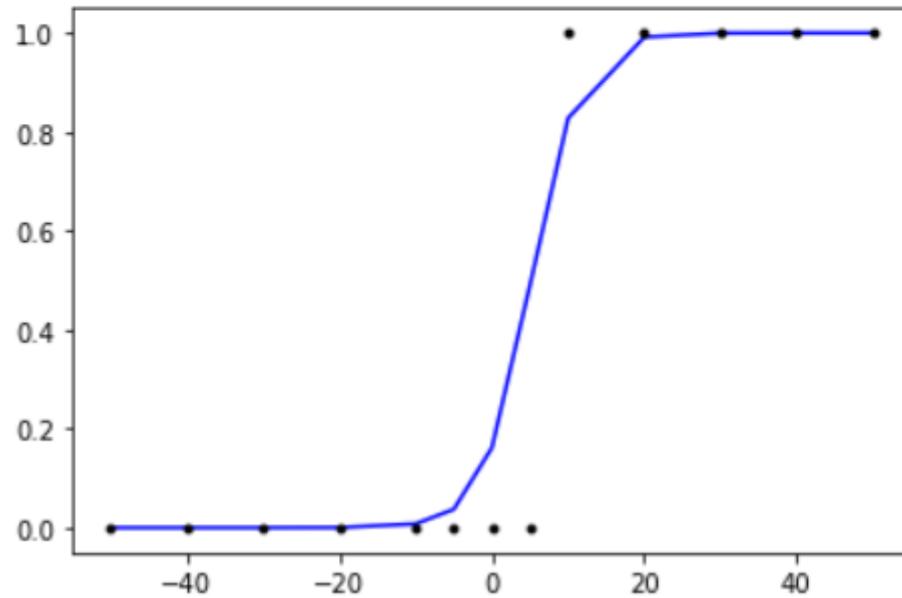
Epoch 2/2

13/13 [=====] - 0s 3ms/step - loss: 0.5258 - binary_accuracy: 0.9231

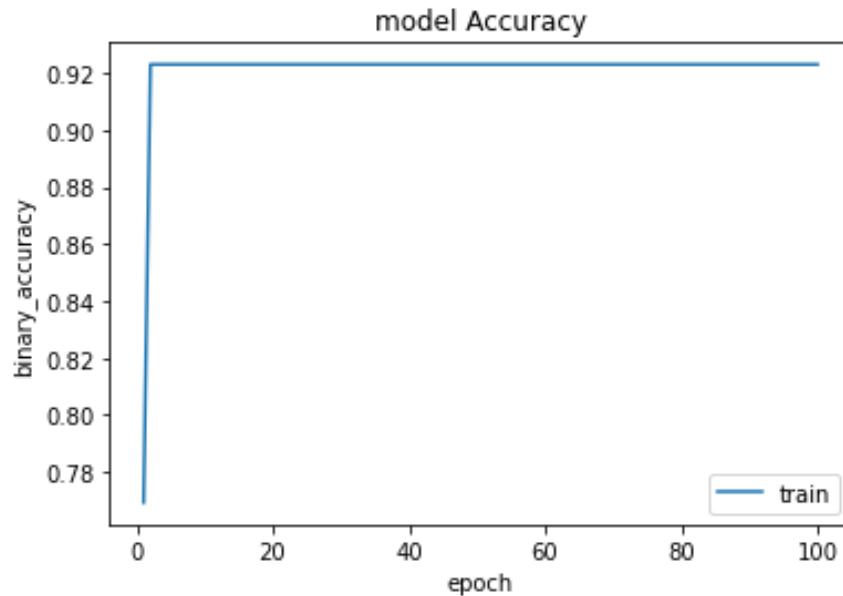
(lab02) 케라스로 구현하는 로지스틱 회귀

```
%matplotlib inline  
import matplotlib.pyplot as plt  
plt.plot(X, model.predict(X), 'b', X,y, 'k.')
```

```
[<matplotlib.lines.Line2D at 0x1ae31bbf188>,  
<matplotlib.lines.Line2D at 0x1ae31d9ffc8>]
```



(실습) 100번 실행해서 아래의 그림을 그려보세요



```
epochs = range(1, len(history.history['binary_accuracy']) + 1)
plt.plot(epochs, history.history['binary_accuracy'])
plt.title('model Accuracy')
plt.ylabel('binary_accuracy')
plt.xlabel('epoch')
plt.legend(['train'])
plt.show()
```

다중입력 선형 회귀

Multivariable Linear Regression의 예제
중간 및 기말고사 시험 성적을 예측해보자

다중 선형 회귀

- 데이터
 - ✓ 중간 고사, 기말 고사, 추가 점수를 어떤 공식을 통해 최종 점수를 계산한 데이터

$$H(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x_1, x_2, x_3) - y_i)^2$$

(lab1-03) 다중 선형 회귀

```
import numpy as np
from tensorflow.keras.models import Sequential # 케라스의 Sequential()을 임포트
from tensorflow.keras.layers import Dense # 케라스의 Dense()를 임포트
from tensorflow.keras import optimizers # 케라스의 옵티마이저를 임포트

X=np.array([[70,85,11],[71,89,18],[50,80,20],[99,20,10],[50,10,10]]) # 중간, 기말, 가산점
# 입력 벡터의 차원은 3입니다. 즉, input_dim은 3입니다.
y=np.array([73,82,72,57,34]) # 최종 성적
# 출력 벡터의 차원은 1입니다. 즉, output_dim은 1입니다.

model=Sequential()
model.add(Dense(1, input_dim=3, activation='linear'))
sgd=optimizers.SGD(lr=0.00001)
# 학습률(learning rate, lr)은 0.00001로 합니다.
model.compile(optimizer=sgd ,loss='mse',metrics=['mse'])
# sgd는 경사 하강법을 의미.
# 손실 함수(Loss function)은 평균제곱오차 mse를 사용합니다.
model.fit(X,y, batch_size=1, epochs=2000, shuffle=False)
# 주어진 X와 y데이터에 대해서 오차를 최소화하는 작업을 2,000번 시도합니다.
```

(lab1-03) 다중 선형 회귀

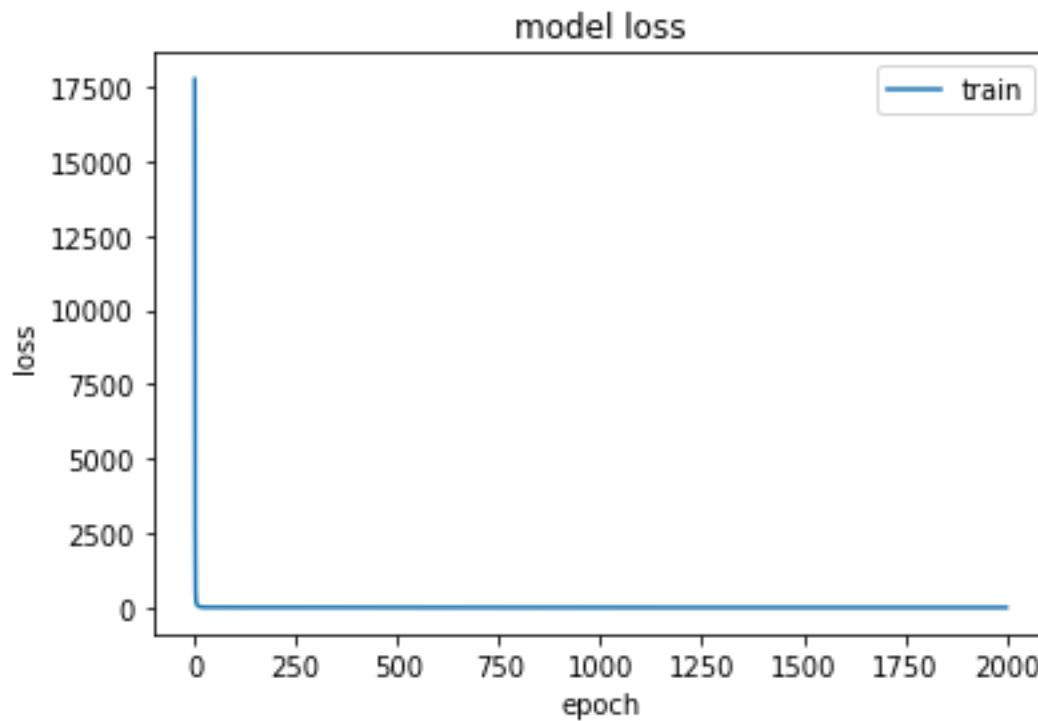
```
print(model.predict(X))
```

```
[[73.07846 ]  
 [81.97684 ]  
 [71.97683 ]  
 [57.152187]  
 [33.704407]]
```

```
X_test=np.array([[20,99,10],[40,50,20]]) # 각각 58점과 56점을 예측해야 합니다.  
print(model.predict(X_test))
```

```
[[58.00618 ]  
 [55.829018]]
```

(실습) 아래 그림을 출력해보세요

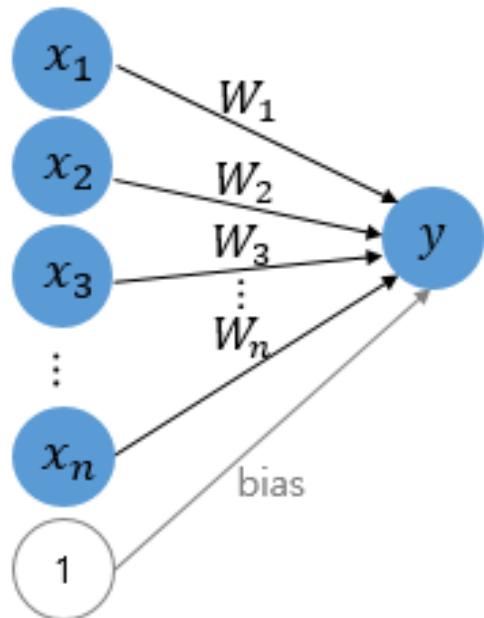


다중입력 로지스틱 회귀

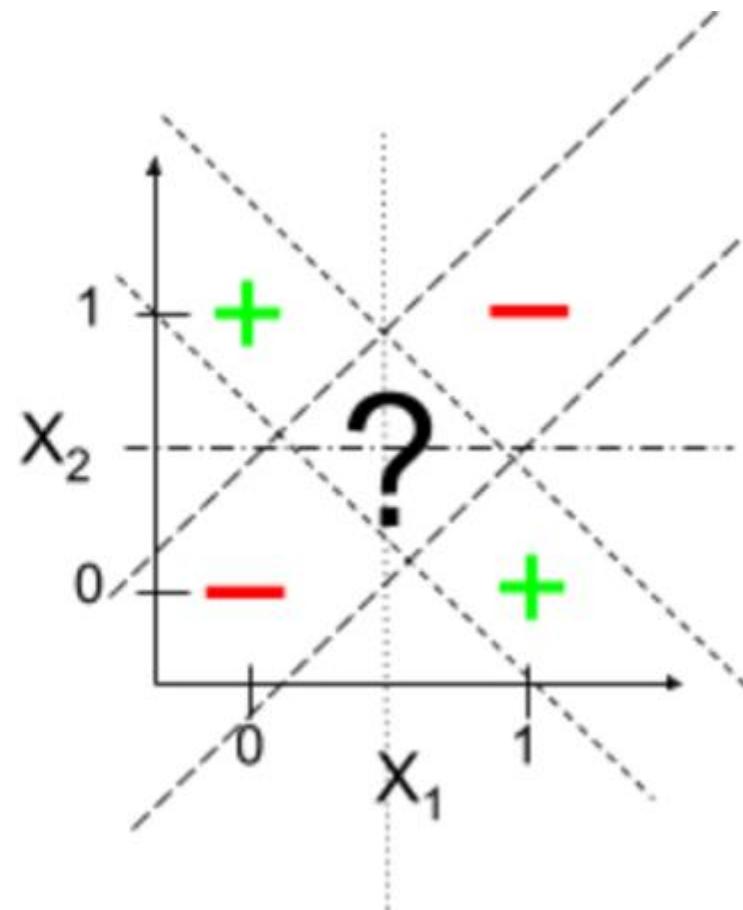
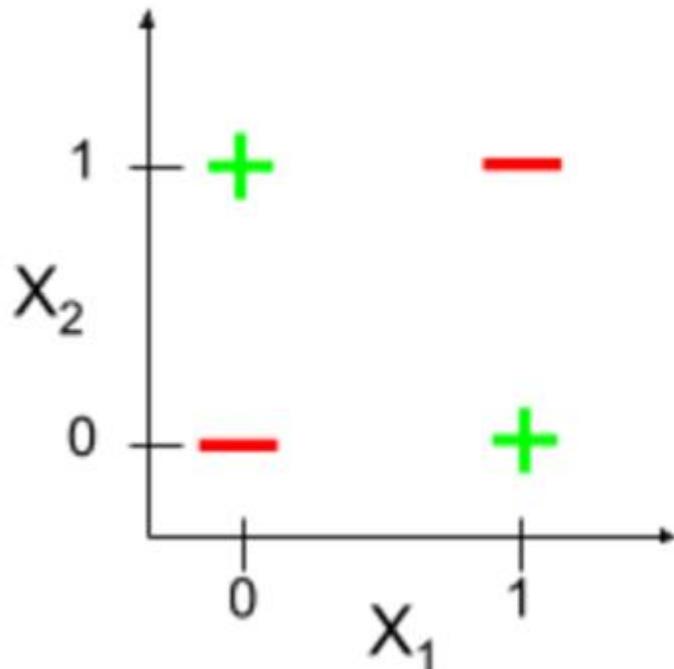
Multivariable Logistic Regression의 예제
XOR 문제를 풀자

인공 신경망 다이어그램과 유사하다.

$$y = \sigma(W_1x_1 + W_2x_2 + W_3x_3 + \dots + W_nx_n + b)$$



XOR 문제



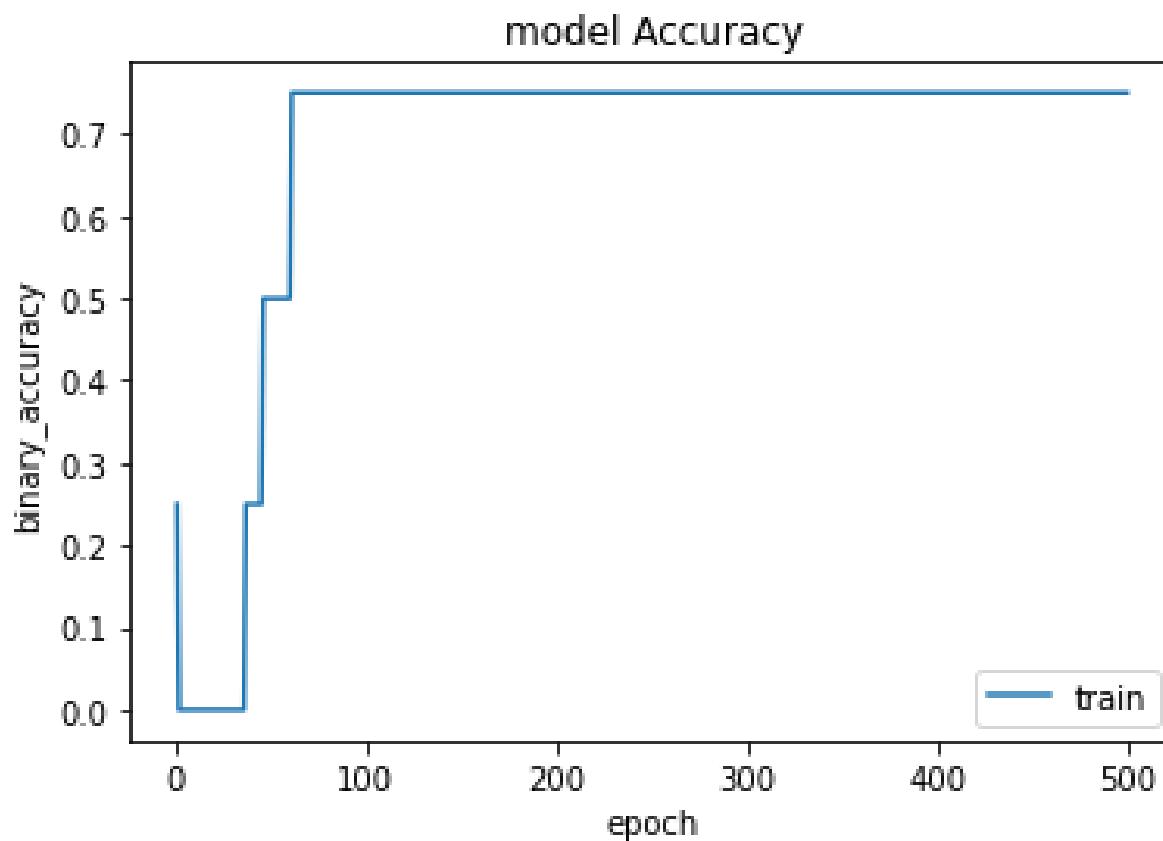
다중 로지스틱 회귀

```
import numpy as np
X=np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
# 입력 벡터의 차원은 2입니다. 즉, input_dim은 2입니다.
y=np.array([0, 1, 1, 1])
# 출력 벡터의 차원은 1입니다. 즉, output_dim은 1입니다.
```

```
from tensorflow.keras.models import Sequential # 케라스의 Sequential()을 임포트
from tensorflow.keras.layers import Dense # 케라스의 Dense()를 임포트
from tensorflow.keras import optimizers # 케라스의 옵티마이저를 임포트

model=Sequential()
model.add(Dense(1, input_dim=2, activation='sigmoid')) # 이제 입력의 차원은 2입니다.
model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['binary_accuracy'])
# sgd는 경사 하강법을 의미.
# 손실 함수(Loss function)는 binary_crossentropy(이진 크로스 엔트로피)를 사용합니다.
history=model.fit(X,y, batch_size=1, epochs=1000, shuffle=False)
# 주어진 X와 y데이터에 대해서 오차를 최소화하는 작업을 800번 시도합니다.
```

(실습) 다중 로지스틱 회귀의 정확도 그래프를 그리세요

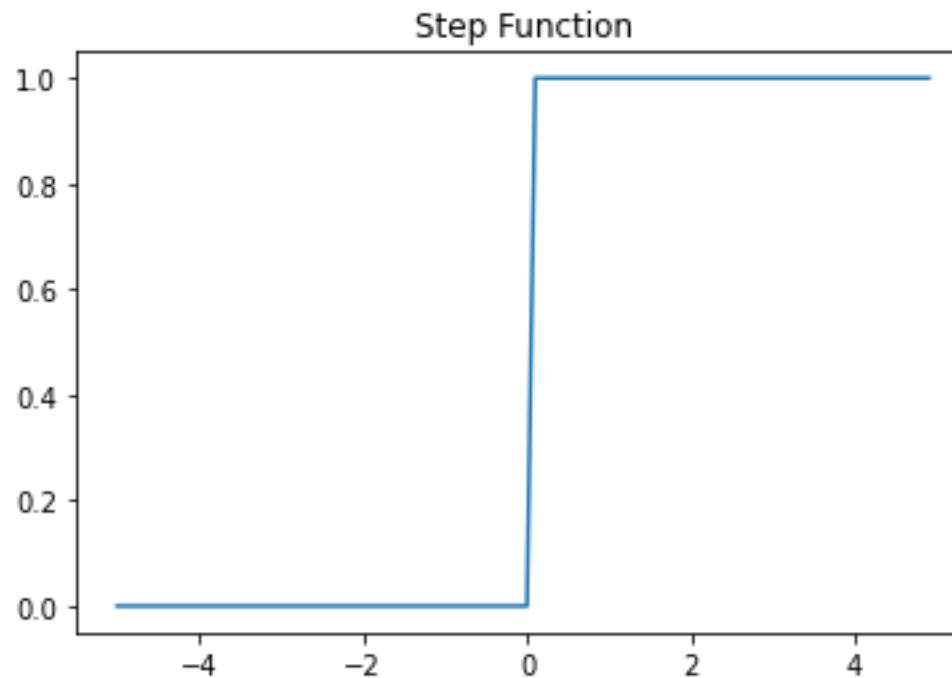


활성함수 만들기

Softmax, Tanh(x), RuLU 등 만들어 보자

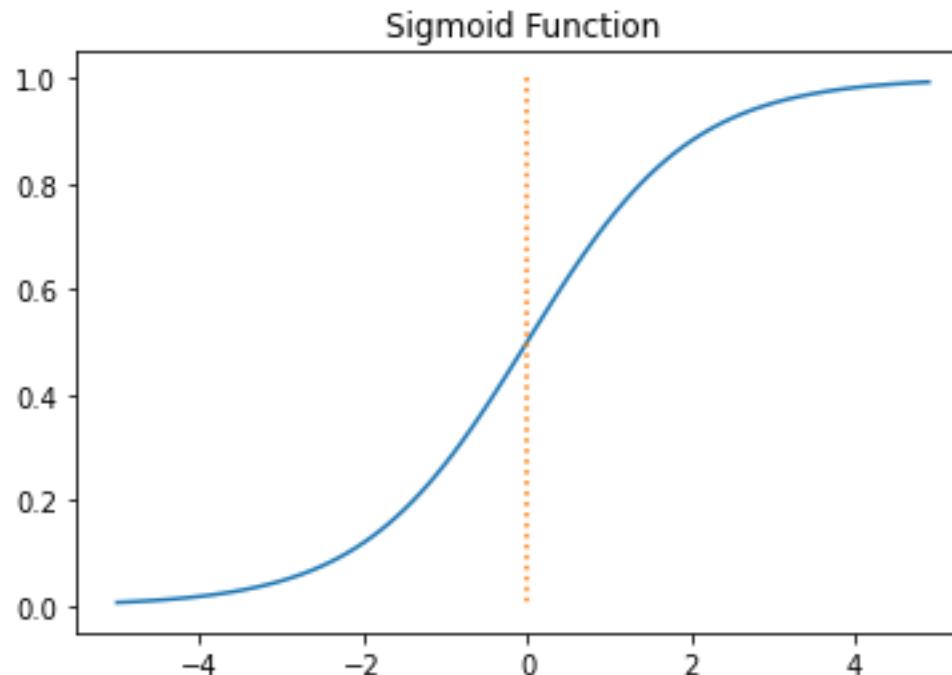
Step function

```
def step(x):  
    return np.array(x > 0, dtype=np.int)
```



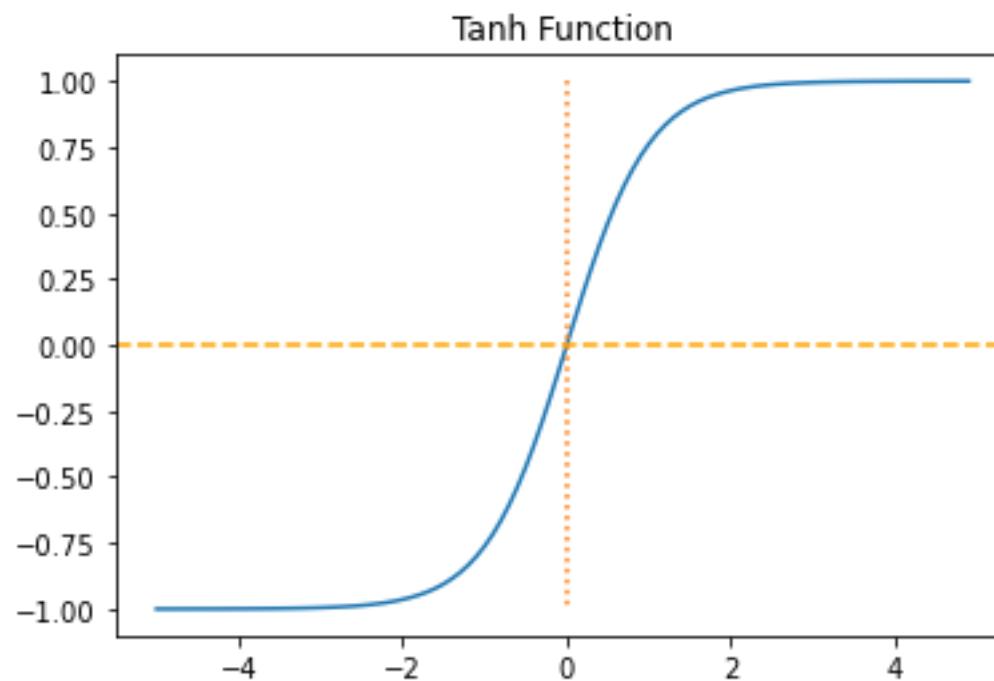
Sigmoid

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



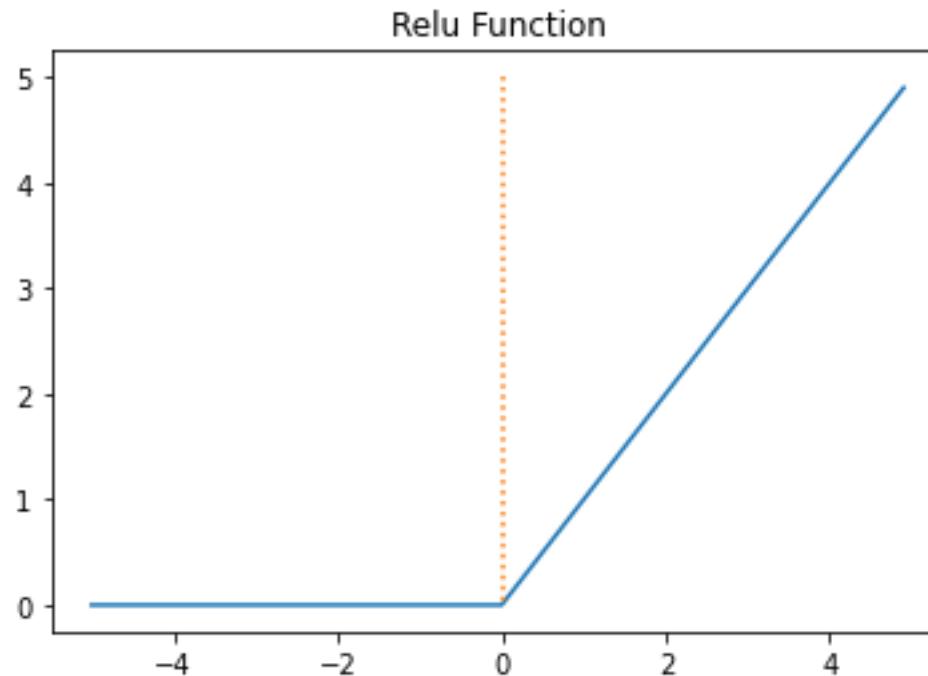
Tanh(x)

$$y = \text{np.tanh}(x)$$



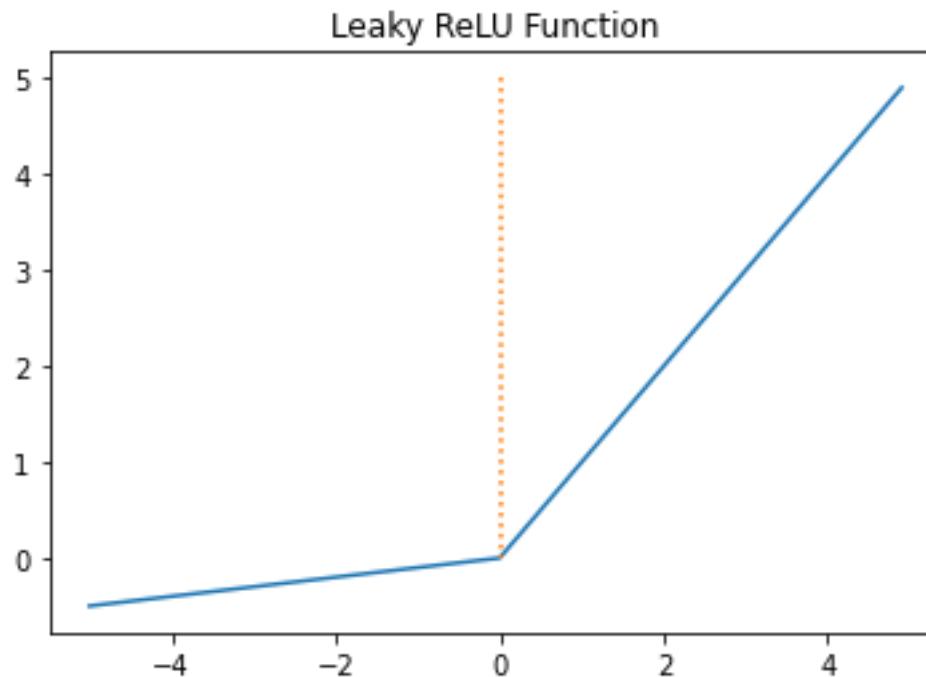
ReLU

```
def relu(x):  
    return np.maximum(0, x)
```



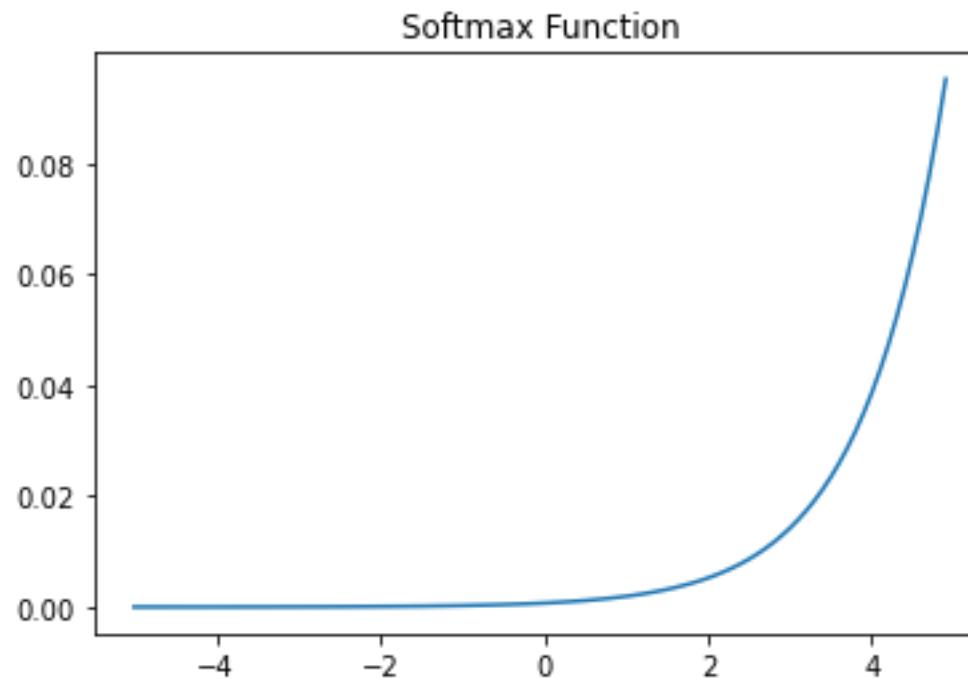
Leaky ReLU

```
def leaky_relu(x):  
    return np.maximum(a*x, x)
```



Softmax

$$y = \text{np.exp}(x) / \text{np.sum}(\text{np.exp}(x))$$



소프트맥스 회귀(Softmax) – 다중 클래스 분류

- 소프트맥스 회귀(Softmax Regression)
 - ✓ 3개 이상의 선택지 중에서 1개를 고르는 다중 클래스 분류(회귀)
 - ✓ 붓꽃 (IRIS)

SepalLengthCm(x_1)	SepalWidthCm(x_2)	PetalLengthCm(x_3)	PetalWidthCm(x_4)	Species(y)
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
5.8	2.6	4.0	1.2	versicolor
6.7	3.0	5.2	2.3	virginica
5.6	2.8	4.9	2.0	virginica

소프트맥스 함수(Softmax function)

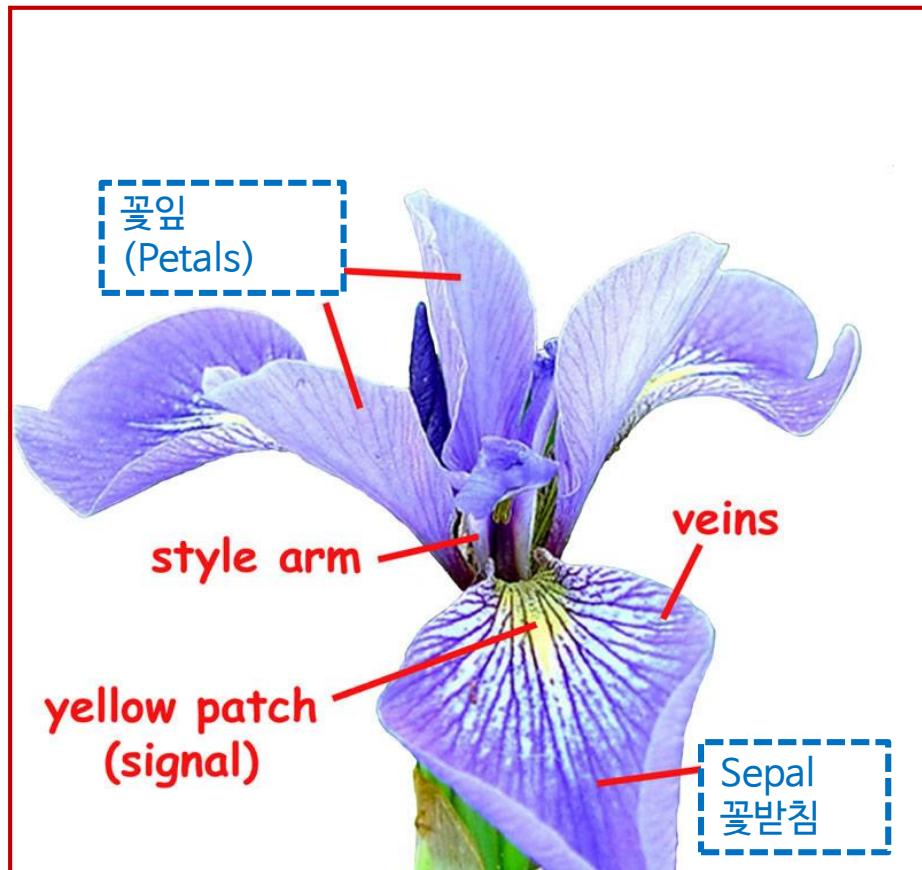
$$p_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}} \quad for \ i = 1, 2, \dots k$$

$$\text{softmax}(z) = [\frac{e^{z_1}}{\sum_{j=1}^3 e^{z_j}} \ \ \frac{e^{z_2}}{\sum_{j=1}^3 e^{z_j}} \ \ \frac{e^{z_3}}{\sum_{j=1}^3 e^{z_j}}] = [p_1, p_2, p_3] = \hat{y} = \text{prediction}$$

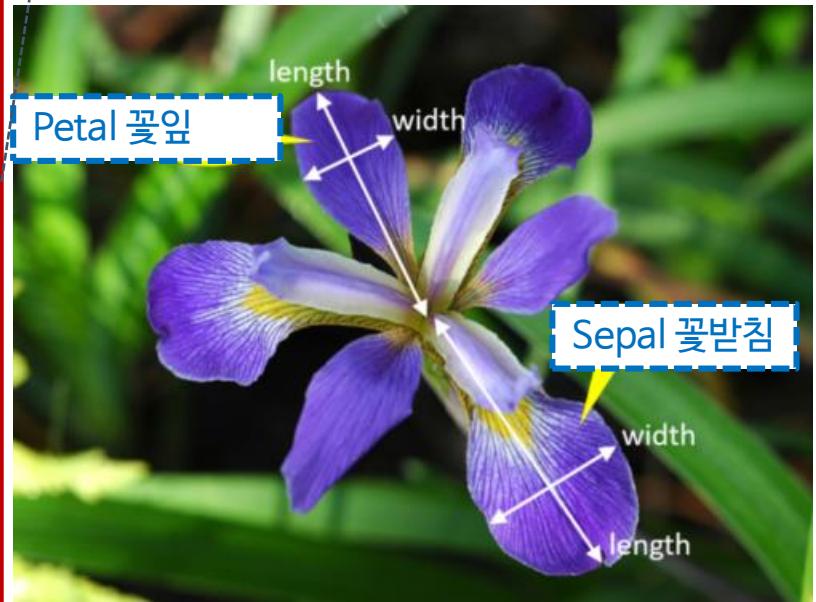
활성함수 만들기

Softmax, Tanh(x), RuLU 등 만들어 보자

(실습2) 붓꽃 데이터를 불러오기



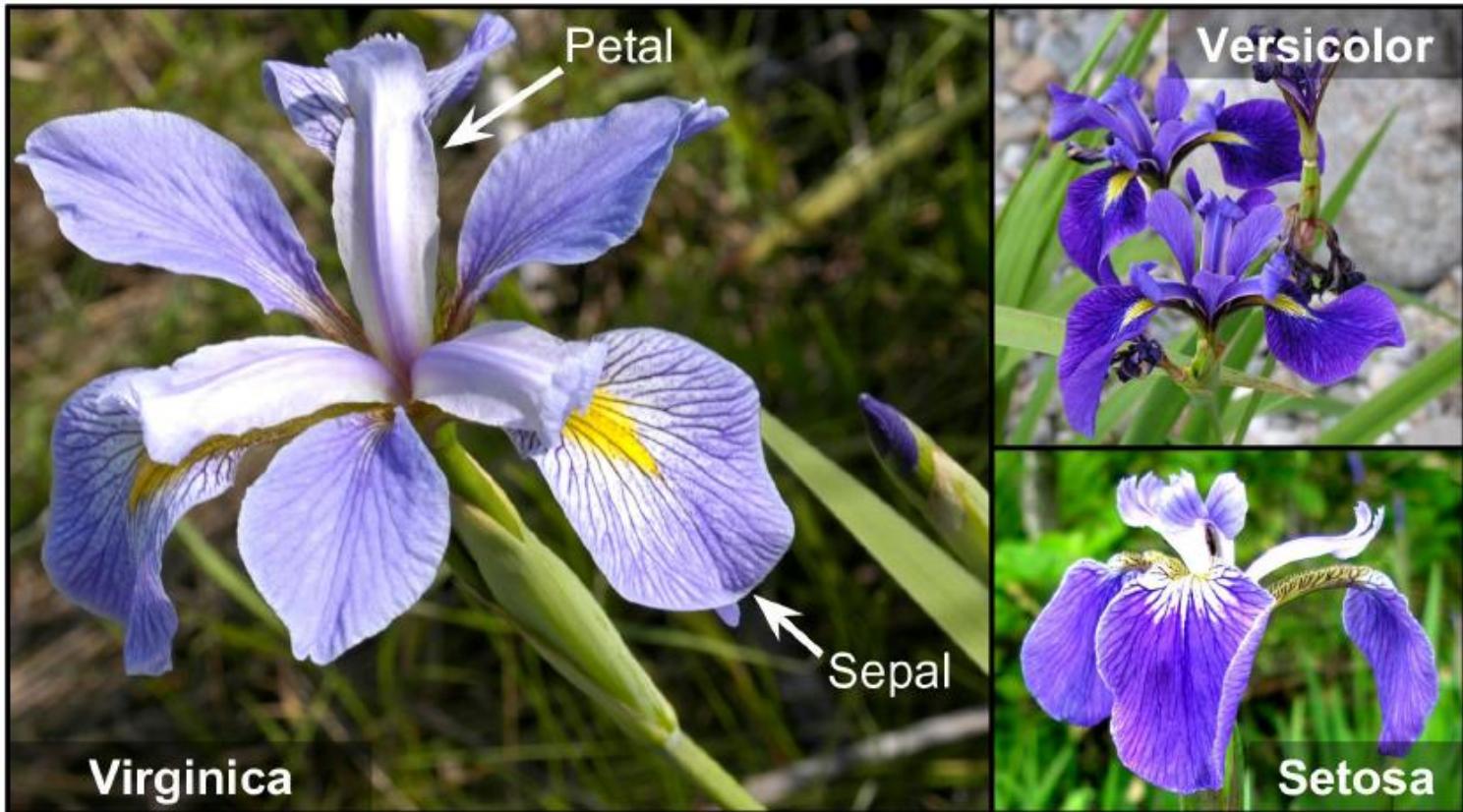
그림을 보면 직관적으로, (크기를 가름해보면)
꽃받침(sepal) 길이, 넓이, 꽃잎 길이, 넓이
순으로 데이터 값이 될 것이라고 추정해 볼 수
있다.



다중 로지스틱 회귀

- **붓꽃 (IRIS) 데이터**

- 꽃받침(Sepal)의 길이와 꽃잎(Petal)의 길이와 해당 꽃이 A인지 B인지가 적혀져 있는 데 이터가 있을 때, 새로 조사한 꽃받침의 길이와 꽃잎의 길이로부터 무슨 꽃인지 예측



붓꽃 예측 프로세스 정리

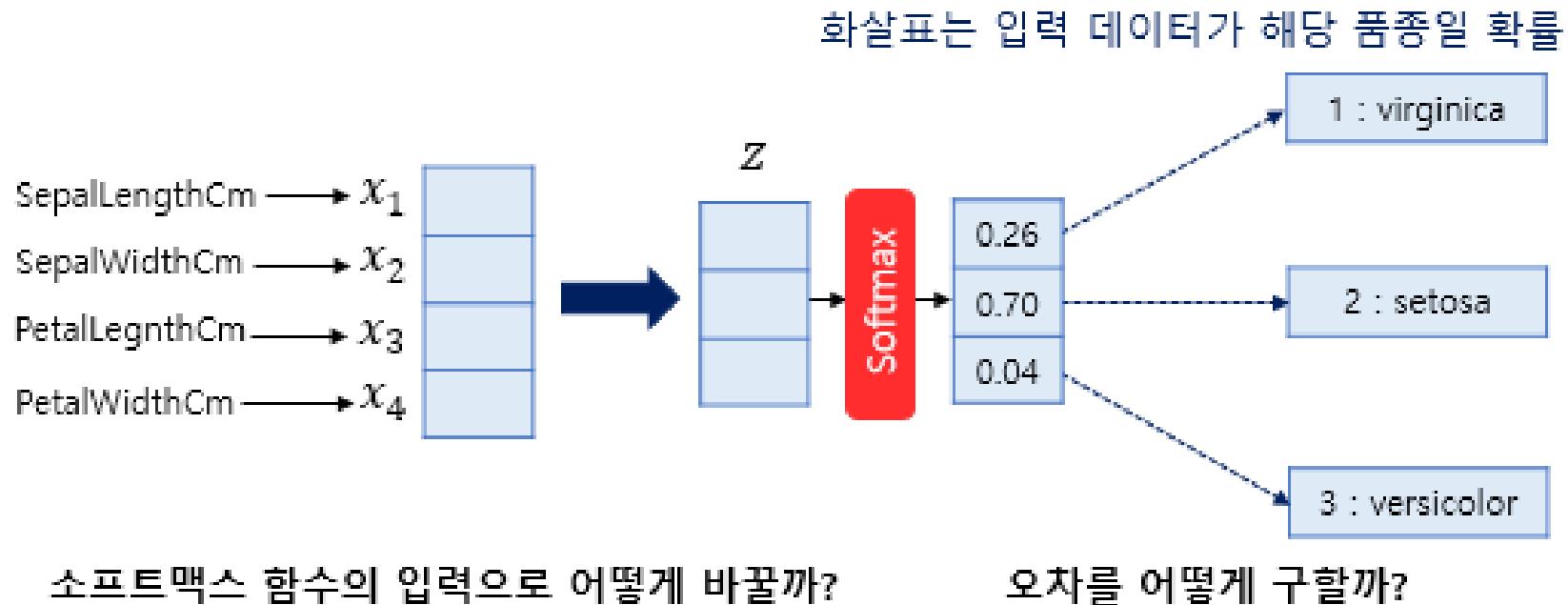
- 데이터 세트 분리
 - ✓ 학습, 테스트, (검증) 데이터
- 모델 학습
 - ✓ 가장중요. 어떻게 학습하는 것인가?
 - ✓ 신경망은 이해하기 쉬운데. 블랙박스인가?
- 예측 수행
 - ✓ 예측 수행이란 무엇인가? 학습은 왜 했으며, 무엇을 하고자 하는 것인가?
 - ✓ 학습한 것이란 무엇인가? 왜 학습을 해야만 하는가?
 - ✓ 신경망에서는 웨이트 저장으로 이해할 수 있다.
- 평가
 - ✓ 예측의 의미는 무엇인가? 실제 예측하는 것인가?
 - ✓ 라벨은 이미 다 되어 있는데 무엇을 예측하는 것인가?

다중 로지스틱 회귀는 바이너리 (2개의 라벨)

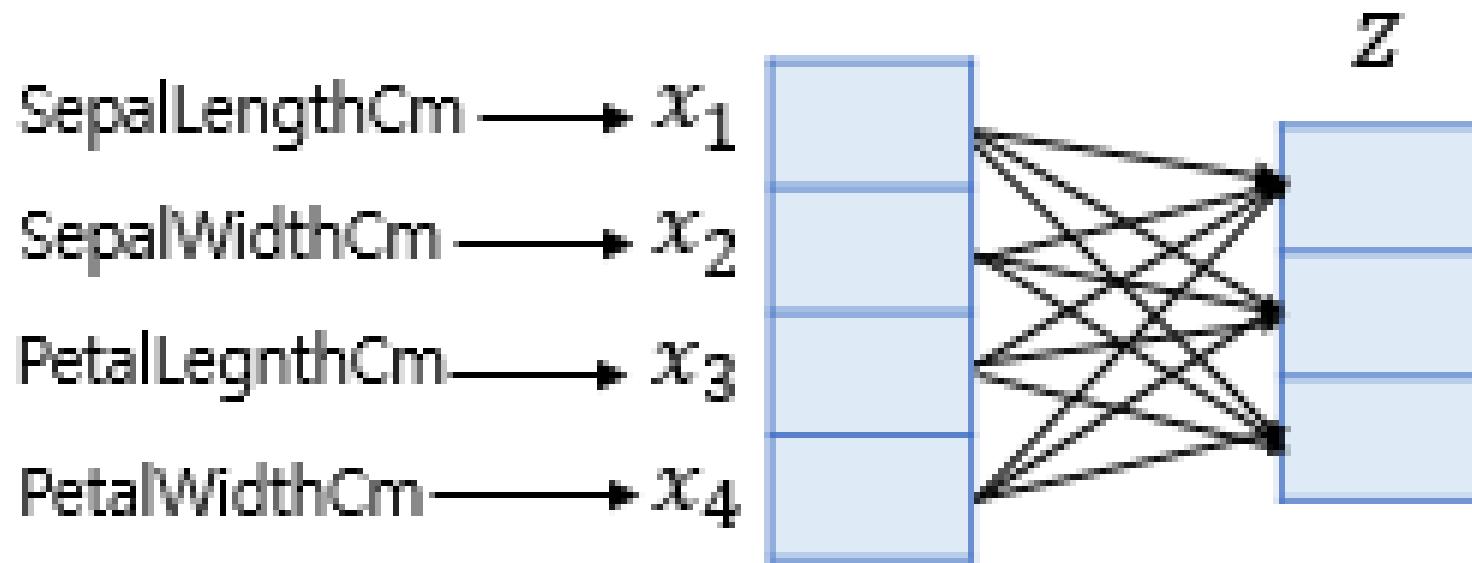
SepalLengthCm(x_1)	PetalLengthCm(x_2)	Species(y)
5.1	3.5	A
4.7	3.2	A
5.2	1.8	B
7	4.1	A
5.1	2.1	B

$$H(X) = \text{sigmoid}(W_1x_1 + W_2x_2 + b)$$

소프트맥스 :라벨이 2개 이상



소프트맥스 함수(Softmax function)



소프트맥스 함수(Softmax function)

1
0
0

virginica의 원-핫 벡터

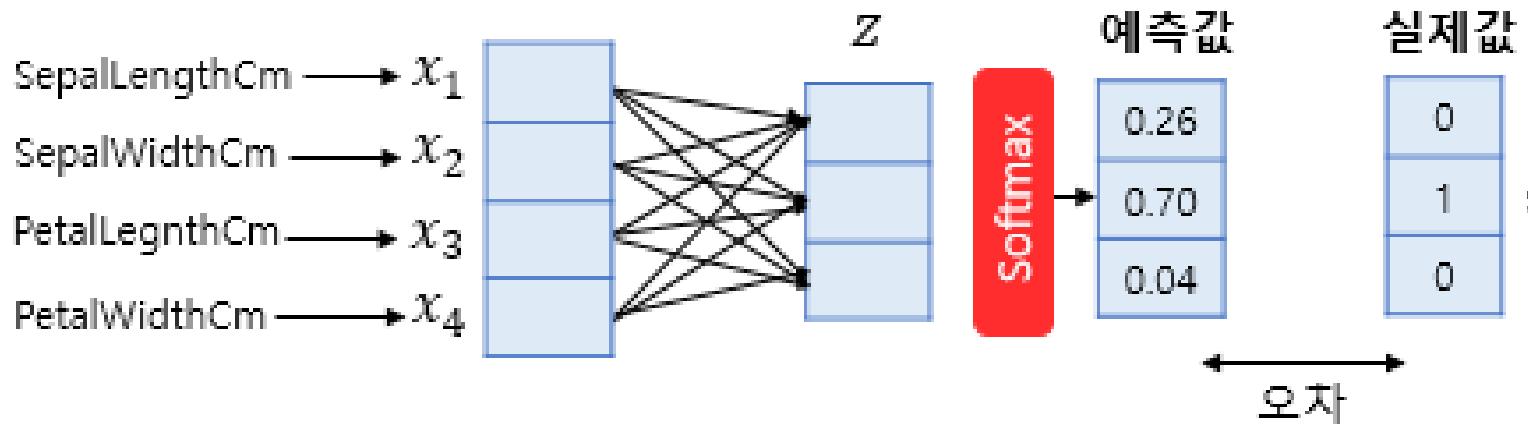
0
1
0

setosa의 원-핫 벡터

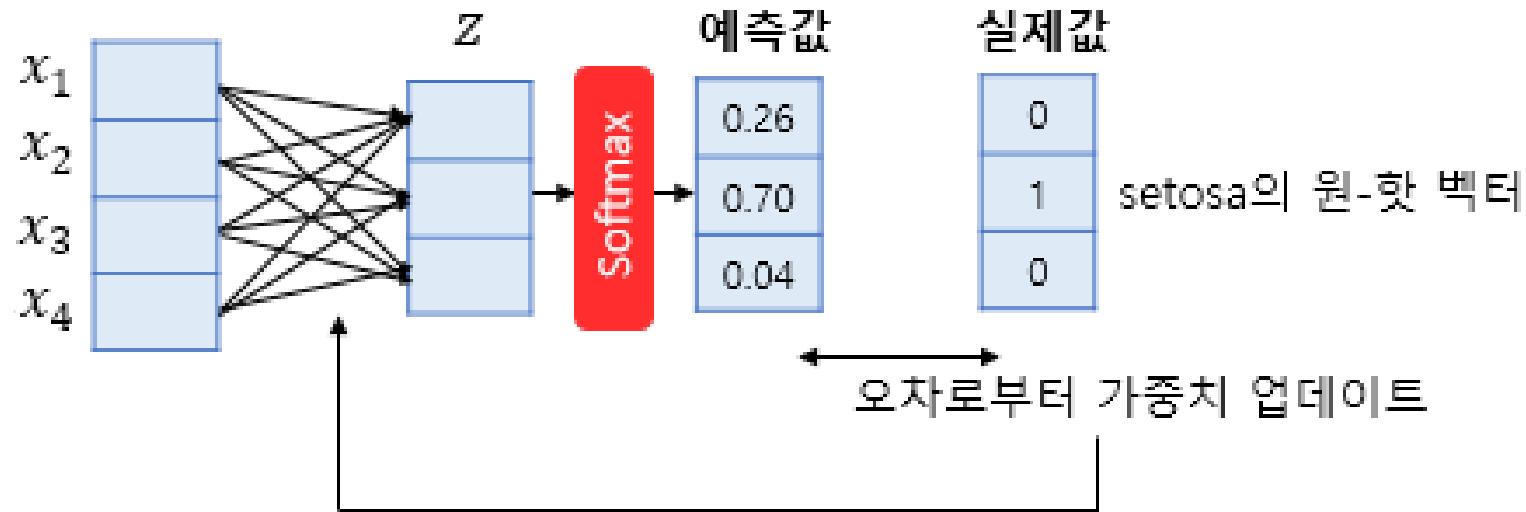
0
0
1

versicolor의 원-핫 벡터

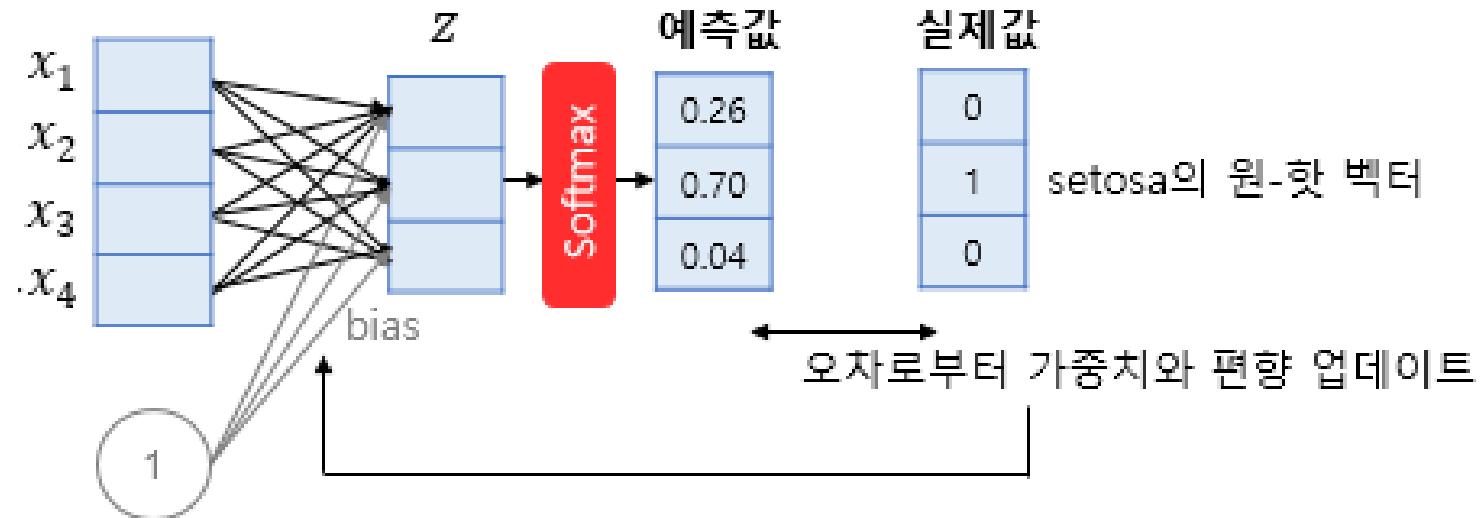
소프트맥스 함수(Softmax function)



소프트맥스 함수(Softmax function)



소프트맥스 함수(Softmax function)



소프트맥스 함수(Softmax function)

$$\text{softmax} \left(\begin{array}{c} W \\ \times \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \\ \begin{array}{c} 3 \times 4 \\ \times \\ 4 \times 1 \end{array} \end{array} \right) + \begin{array}{c} b \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \\ 3 \times 1 \end{array} = \begin{array}{c} \text{예측값} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \\ 3 \times 1 \end{array}$$

소프트맥스의 비용 함수(Cost function)

- 크로스 엔트로피 함수

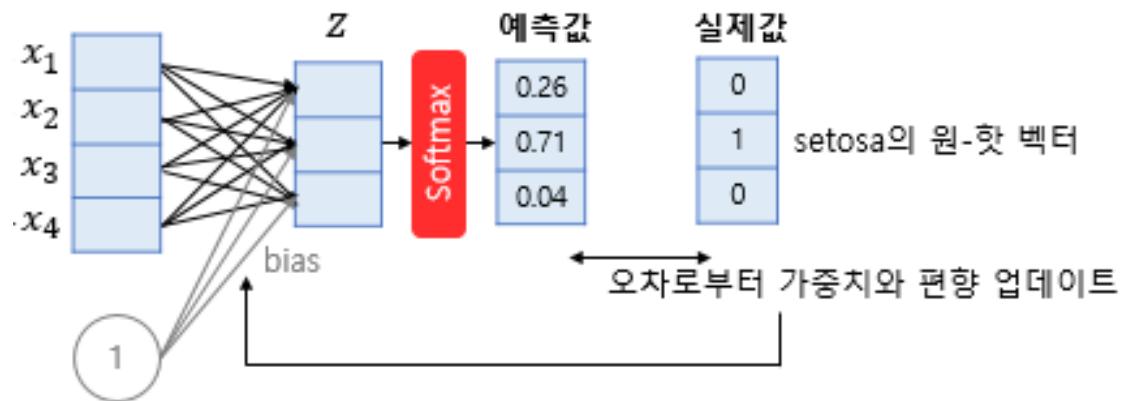
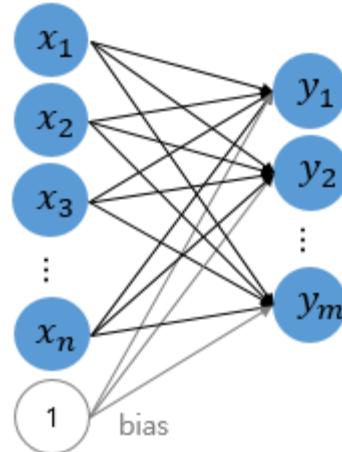
$$cost(W) = - \sum_{j=1}^k y_j \log(p_j)$$

- 개수 n개의 평균

$$cost(W) = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^k y_j^{(i)} \log(p_j^{(i)})$$

$$cost(W) = -\frac{1}{n} \sum_{i=1}^n [y^{(i)} \log(p^{(i)}) + (1 - y^{(i)}) \log(1 - p^{(i)})]$$

소프트맥스 인공 신경망 다이어그램



붓꽃 데이터를 Softmax로 분류해보자

1. 라이브러리 설정하기

```
In [1]: import tensorflow as tf  
from tensorflow import keras  
  
from tensorflow.keras import Sequential  
from tensorflow.keras.layers import Dense, Softmax  
#from tensorflow.keras import optimizers
```

```
In [2]: %matplotlib inline
```

2. 데이터 가져오기

```
In [3]: import pandas as pd  
  
df = pd.read_csv('./input/Iris.csv')  
df.shape
```

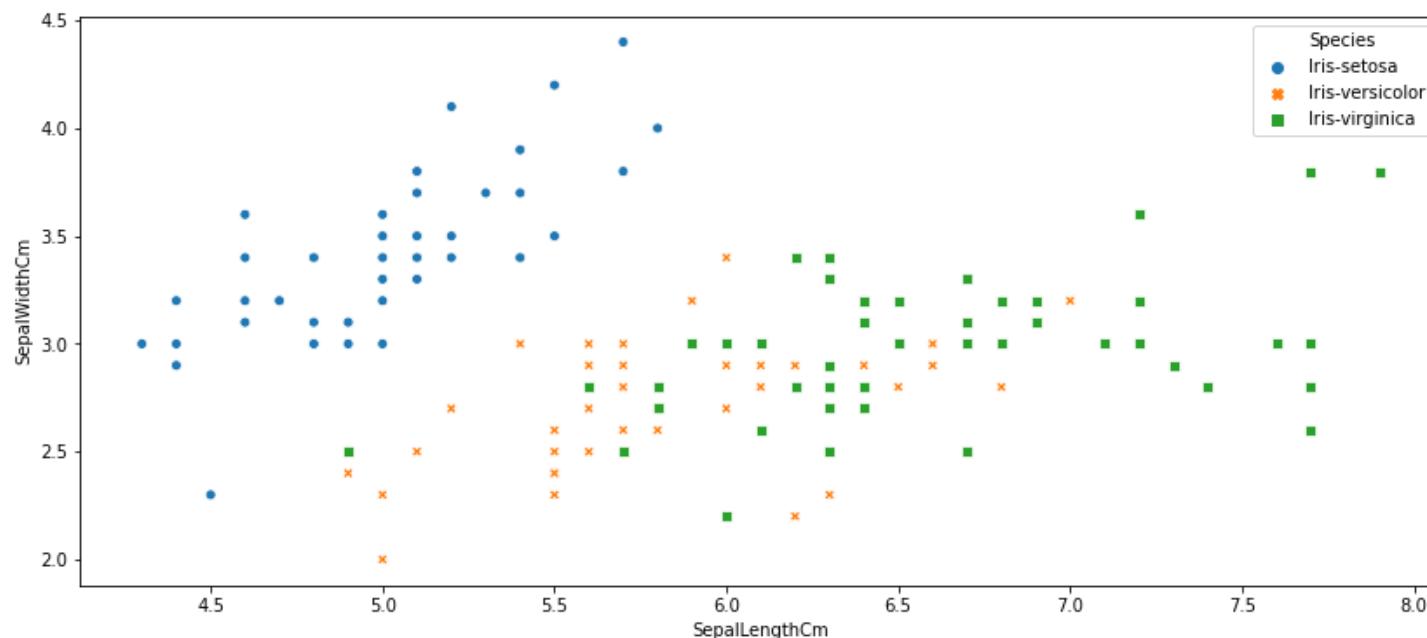
```
Out[3]: (150, 6)
```

3. 데이터 가시화

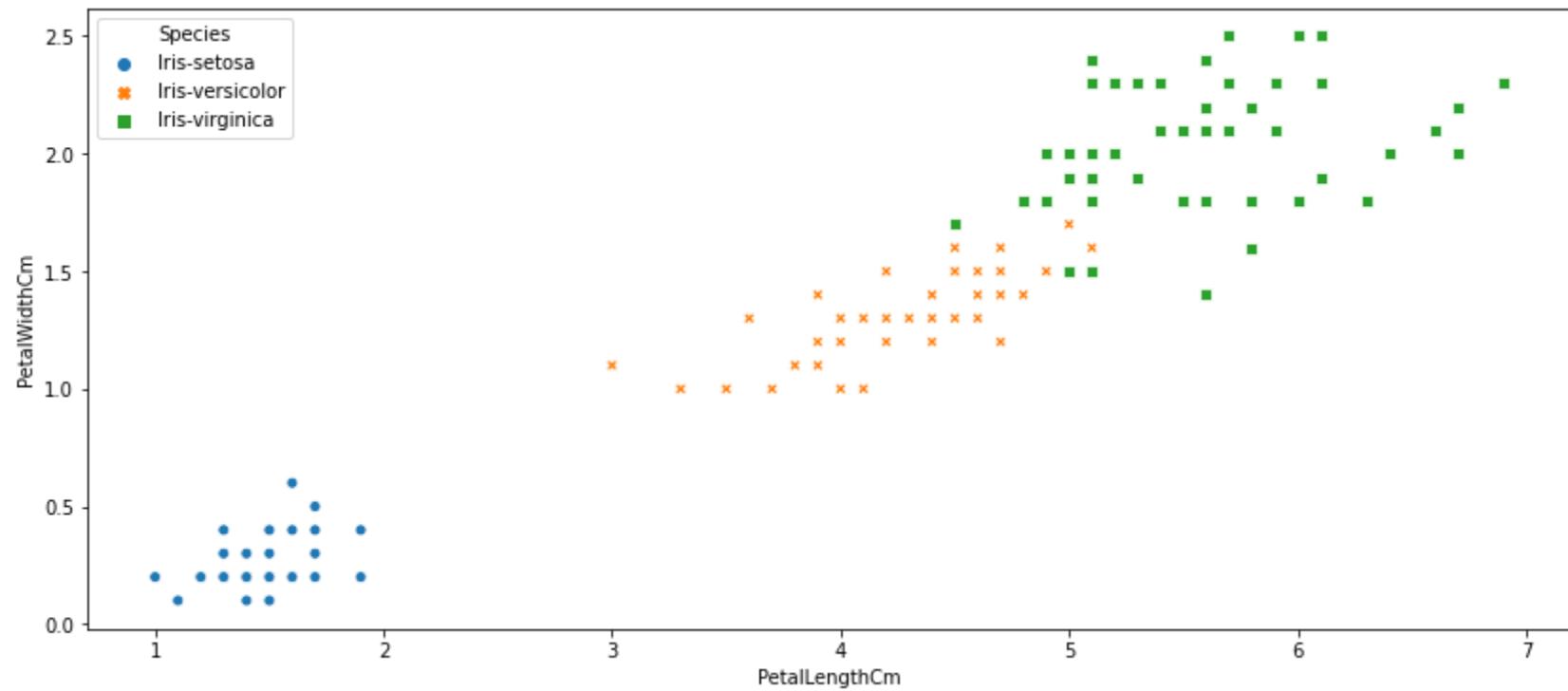
In [30]:

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(14,6))
sns.scatterplot(data=df, x = 'SepalLengthCm',
                 y = 'SepalWidthCm', hue='Species', style='Species')
plt.figure(figsize=(14,6))
sns.scatterplot(data=df, x = 'PetalLengthCm',
                 y = 'PetalWidthCm', hue='Species', style='Species')
```



Petal



머신러닝을 위한 입력 데이터 만들자

1) 입력 X와 출력 y의 값을 정하기

```
In [7]: df['Species'].unique()
```

```
df['Species'] = df['Species'].str[5:]  
df.head()
```

```
Out [7]:
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
In [8]: feature_cols = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']  
target_col = 'Species'  
X = df[feature_cols]  
y = df[target_col]
```

출력용 라벨을 머신러닝용을 다루자

- 타켓의 텍스트를 숫자로 바꾸자

```
In [9]: class_dic = {'setosa':0, 'versicolor':1, 'virginica':2}  
y_ohc = y.apply(lambda z: class_dic[z])
```

머신러닝 모델을 만들자

```
In [10]: import sklearn

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

In [11]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_ohc, test_size=0.2, random_state=20)

X_train.shape

In [13]: model = keras.Sequential([
    keras.layers.Dense(3, activation = 'softmax', input_shape=[4])])

model.compile(loss='SparseCategoricalCrossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()

Model: "sequential"

Layer (type)          Output Shape         Param #
=====
dense (Dense)         (None, 3)            15
=====
Total params: 15
Trainable params: 15
Non-trainable params: 0
```

```
In [14]: history = model.fit(X_train, y_train, epochs = 300, batch_size = 64, verbose=2)
```

```
Epoch 291/300
2/2 - 0s - loss: 0.8892 - accuracy: 0.5583
Epoch 292/300
2/2 - 0s - loss: 0.8884 - accuracy: 0.5583
Epoch 293/300
2/2 - 0s - loss: 0.8876 - accuracy: 0.5667
Epoch 294/300
2/2 - 0s - loss: 0.8867 - accuracy: 0.5667
Epoch 295/300
2/2 - 0s - loss: 0.8859 - accuracy: 0.5667
Epoch 296/300
2/2 - 0s - loss: 0.8851 - accuracy: 0.5667
Epoch 297/300
2/2 - 0s - loss: 0.8842 - accuracy: 0.5750
Epoch 298/300
2/2 - 0s - loss: 0.8834 - accuracy: 0.5750
Epoch 299/300
2/2 - 0s - loss: 0.8826 - accuracy: 0.5750
Epoch 300/300
2/2 - 0s - loss: 0.8818 - accuracy: 0.5750
```

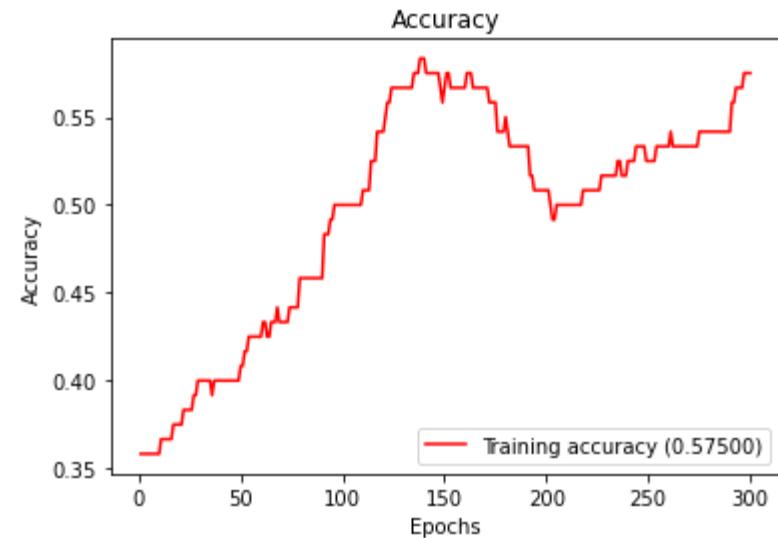
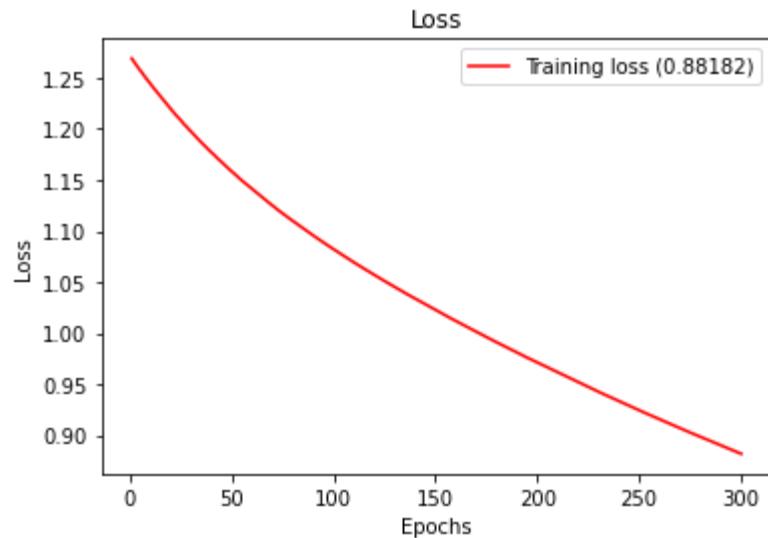
```
In [16]: model.evaluate(X_test, y_test)
```

```
1/1 [=====] - 0s 179ms/step - loss: 0.9124 - accuracy: 0.5667
```

```
Out[16]: [0.9123761653900146, 0.5666666626930237]
```

실행 결과

```
In [33]: import yidL_util as myplot  
myplot.plot_history(history)
```



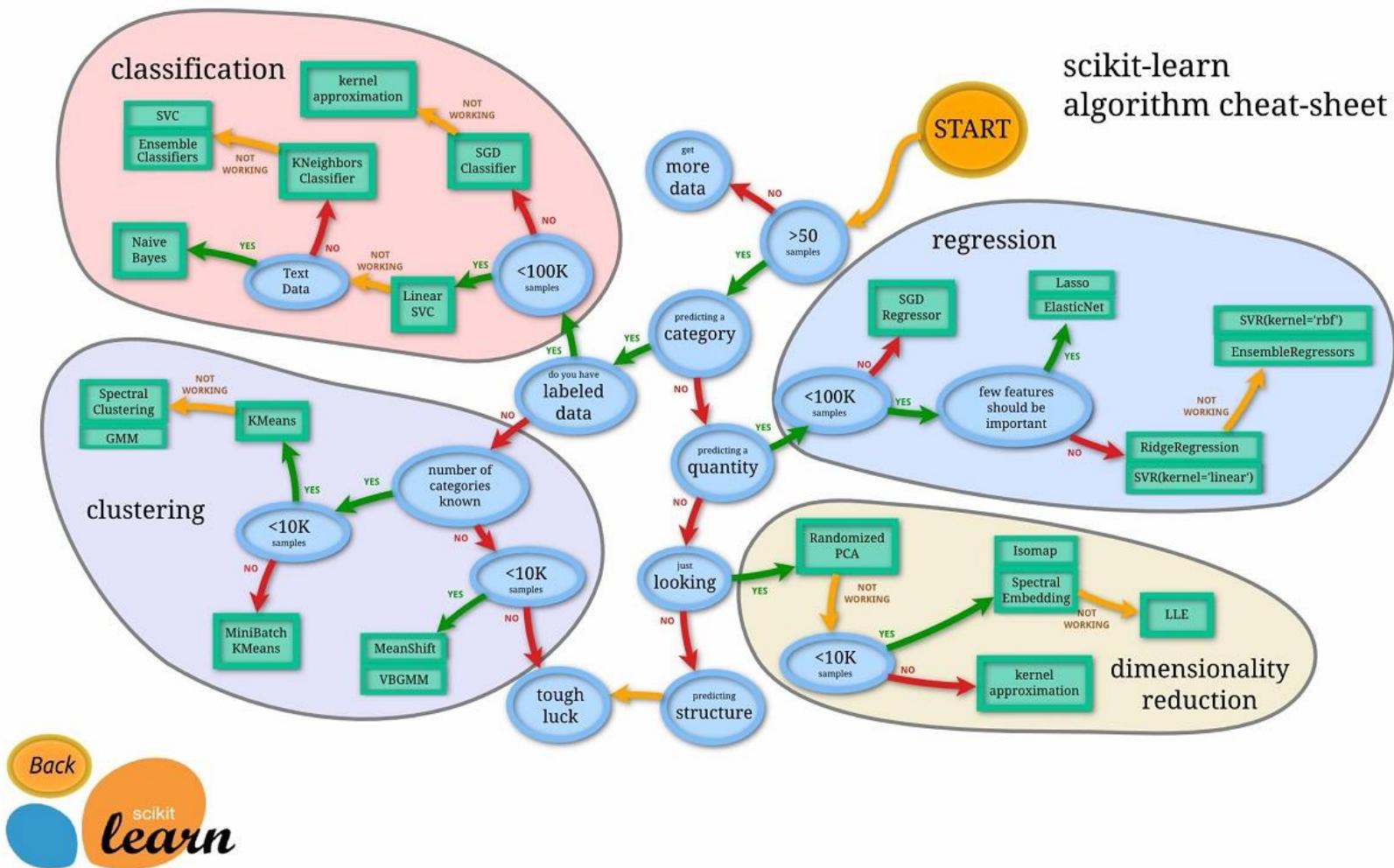
```
In [32]: import numpy as np  
acc_dnn = history.history['accuracy'][np.argmin(history.history['loss'])]  
print('The accuracy of the Deep Learning is:', acc_dnn)
```

The accuracy of the Deep Learning is: 0.574999988079071

붓꽃(IRIS) 분류 및 회귀

RNN, LSTM, GRU

머신러닝 분류/회귀/클러스터링/차원축소





Iris Versicolor



Iris Setosa



Iris Virginica

라이브러리 설정하기

```
In [1]: import warnings  
warnings.filterwarnings("ignore")
```

```
In [2]: import matplotlib as mpl
```

```
In [3]: import matplotlib.pyplot as plt  
import seaborn as sns  
  
import numpy as np  
import pandas as pd  
import sklearn
```

```
In [4]: import sys  
import time
```

```
In [5]: import tensorflow as tf  
from tensorflow import keras  
  
from tensorflow.keras import Sequential  
from tensorflow.keras.layers import Flatten, Dense, Softmax  
#from tensorflow.keras import optimizers
```

```
In [6]: print("python", sys.version)  
for module in mpl, np, pd, sklearn, tf, keras, sns:  
    print(module.__name__, module.__version__)
```

데이터 가져오기

```
In [8]: df = pd.read_csv('./input/Iris.csv')
df.shape
```

```
Out[8]: (150, 6)
```

```
In [9]: df.head(2)
```

```
Out[9]:
```

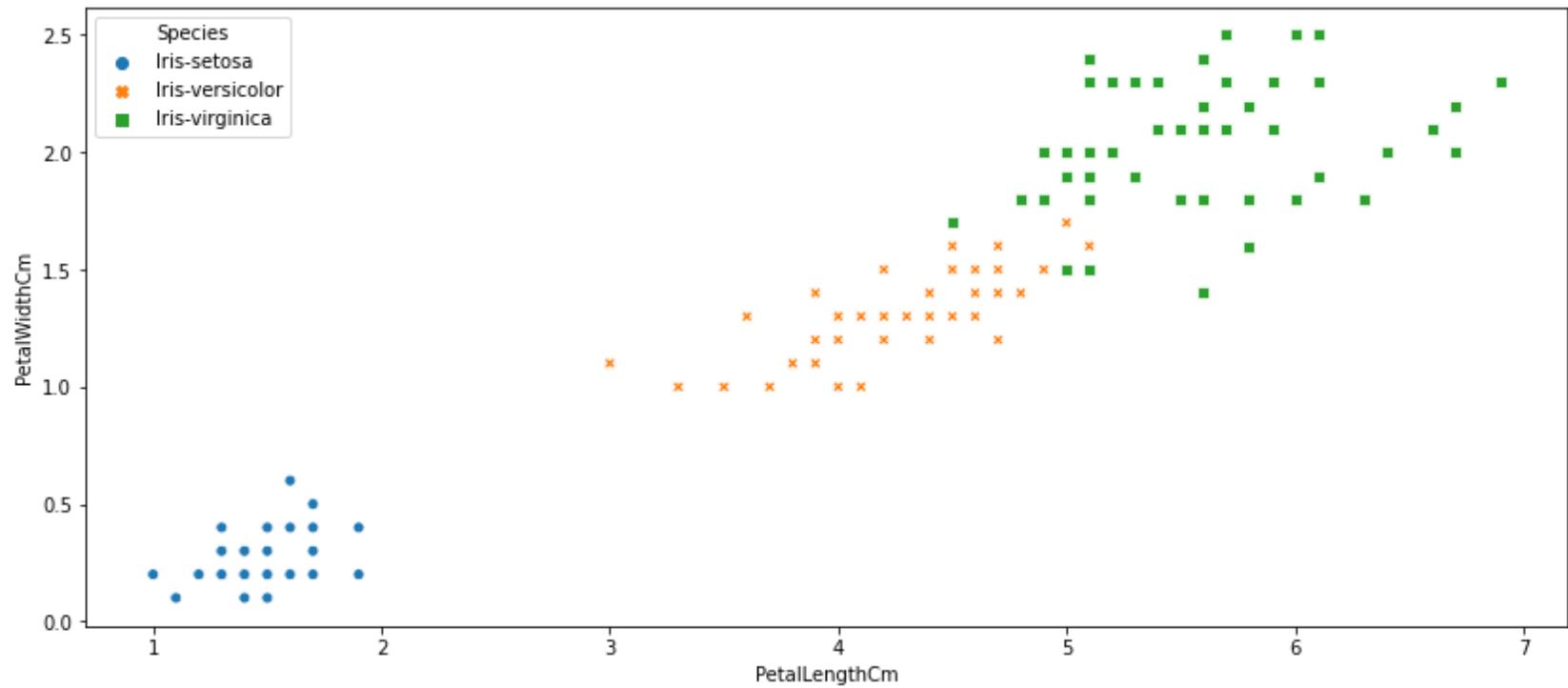
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa

```
In [10]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Id               150 non-null    int64  
 1   SepalLengthCm    150 non-null    float64 
 2   SepalWidthCm     150 non-null    float64 
 3   PetalLengthCm    150 non-null    float64 
 4   PetalWidthCm     150 non-null    float64 
 5   Species          150 non-null    object  
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

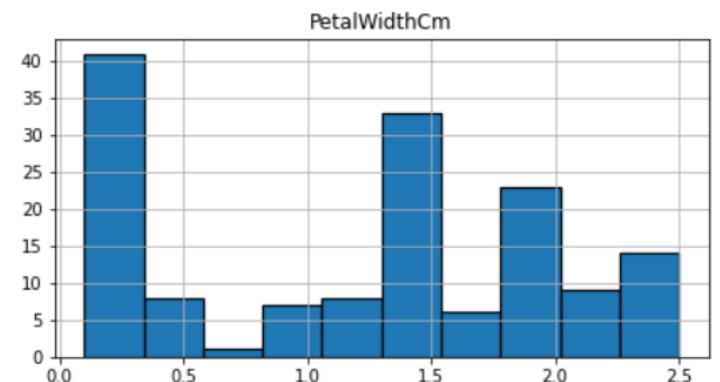
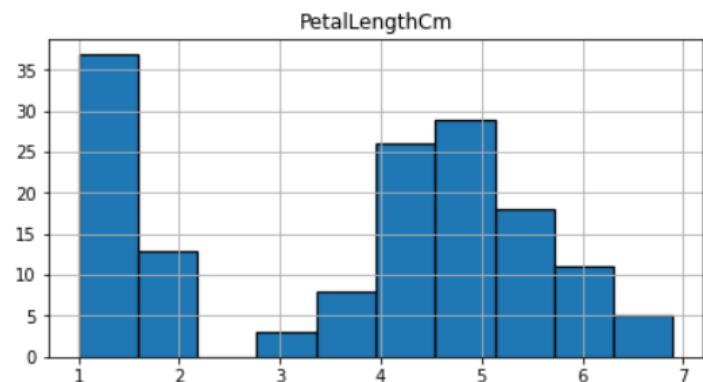
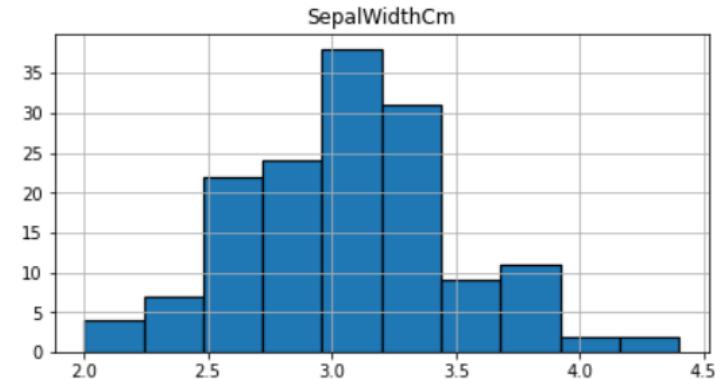
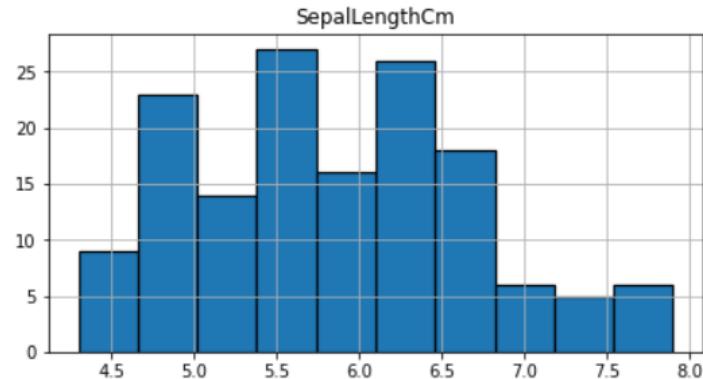
데이터 가시화

```
In [14]: plt.figure(figsize=(14,6))
sns.scatterplot(data=df, x = 'PetalLengthCm', y = 'PetalWidthCm', hue='Species', style='Species')
```



```
In [15]: df.drop('Id',axis=1,inplace=True)
```

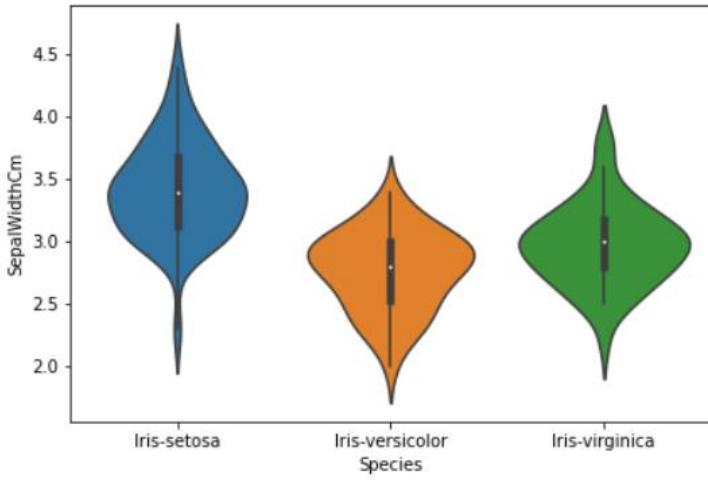
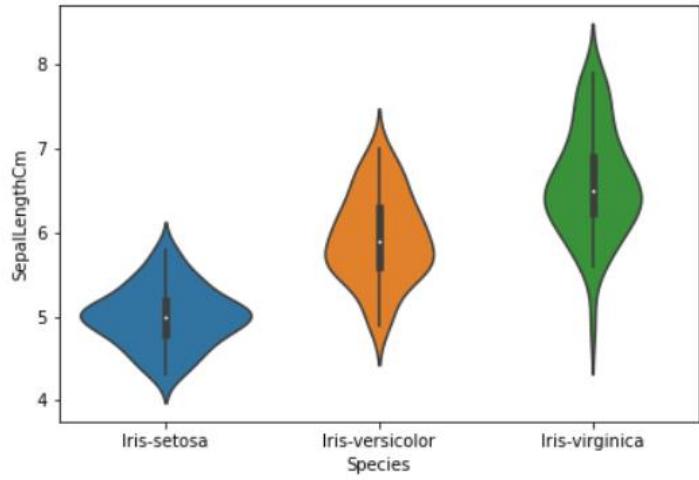
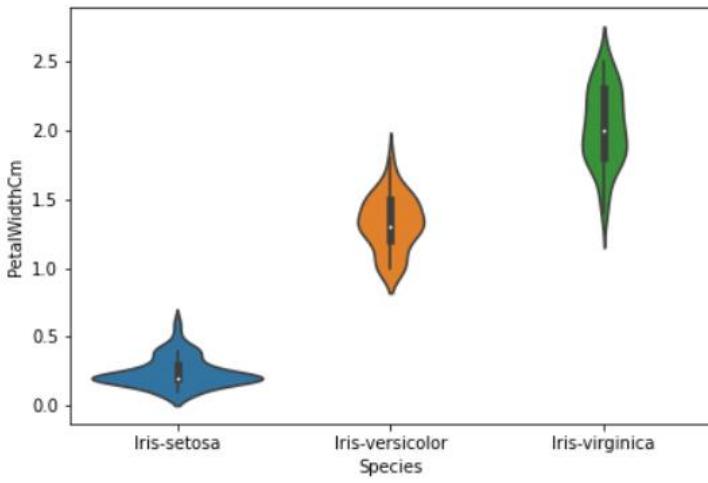
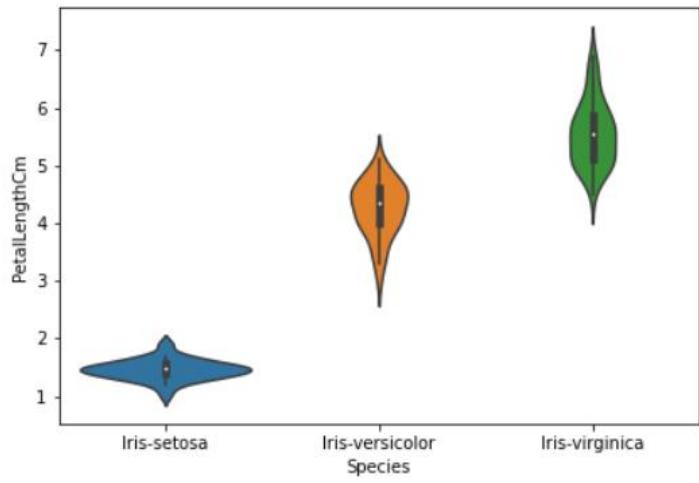
```
In [16]: df.hist(edgecolor='black', linewidth=1.2)
fig=plt.gcf()
fig.set_size_inches(16,8)
plt.show()
```



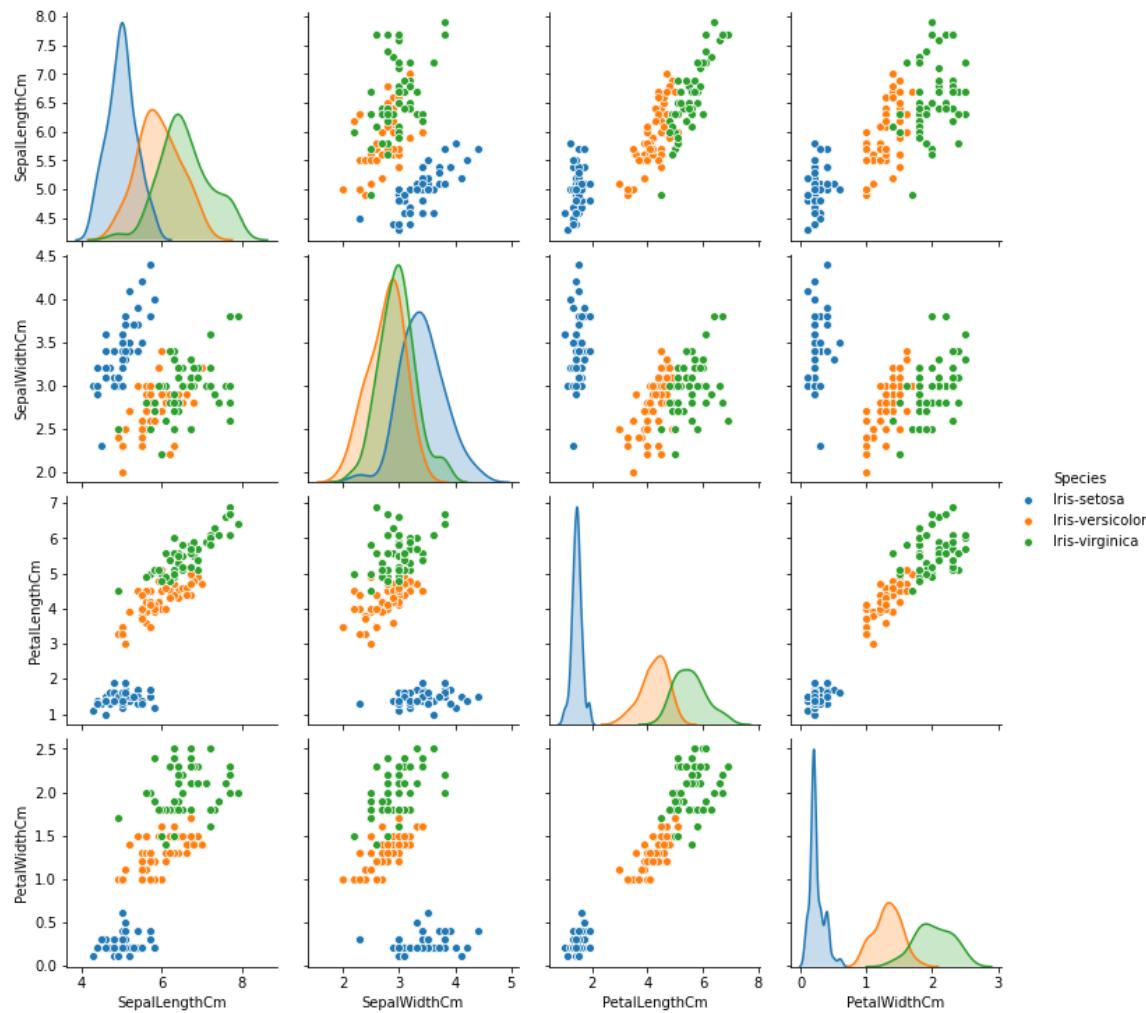
가시화

```
In [17]: plt.figure(figsize=(15,10))

plt.subplot(2,2,1)
sns.violinplot(x='Species',y='PetalLengthCm',data=df)
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='PetalWidthCm',data=df)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='SepalLengthCm',data=df)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='SepalWidthCm',data=df)
```



`sns.pairplot(df,hue='Species')`



머신러닝을 위한 입력 데이터 만들자

1) 입력 X와 출력 y의 값을 정하기

```
In [20]: df['Species'].unique()
```

```
Out[20]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
In [21]: df['Species'] = df['Species'].str[5:]  
df.head()
```

```
Out[21]:
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
In [22]: feature_cols = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']  
target_col = 'Species'  
X = df[feature_cols]  
y = df[target_col]
```

출력용 라벨을 머신러닝용을 다루자

```
In [25]: class_dic = {'setosa':0, 'versicolor':1, 'virginica':2}
y_ohc = y.apply(lambda z: class_dic[z])
```

```
In [27]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y_ohc, test_size=0.20, random_state=80)
```

```
In [28]: print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)

(120, 4) (120,)
(30, 4) (30,)
```

```
In [29]: import sklearn
from sklearn.model_selection import train_test_split
```

```
In [30]: from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

머신러닝 모델을 만들자

```
In [31]: from sklearn.ensemble import RandomForestClassifier  
from sklearn.tree import DecisionTreeClassifier  
from sklearn import svm  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.datasets import make_classification  
from sklearn.linear_model import LogisticRegression  
from sklearn import metrics
```

```
In [32]: from sklearn.model_selection import ShuffleSplit
```

로지스틱 회귀

1) 로지스틱 회귀

```
In [33]: m_lr = LogisticRegression()  
m_lr.fit(X_train,y_train)
```

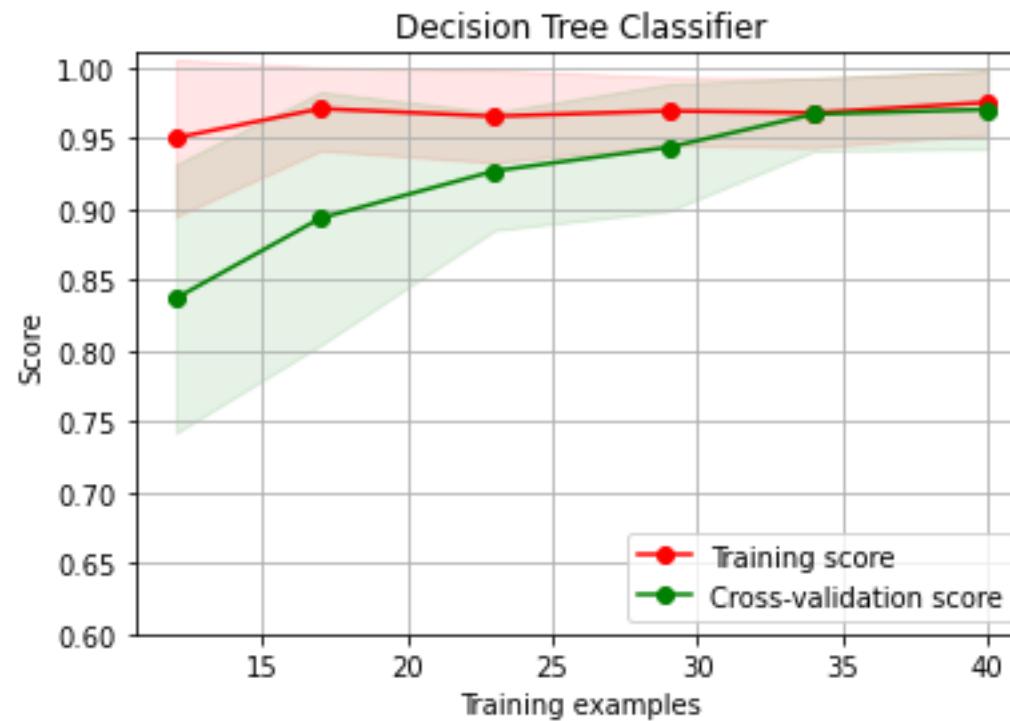
```
Out[33]: LogisticRegression()
```

```
In [34]: pred = m_lr.predict(X_test)  
  
acc_lr = metrics.accuracy_score(pred,y_test)  
print('The accuracy of the Logistic Regression is', acc_lr)
```

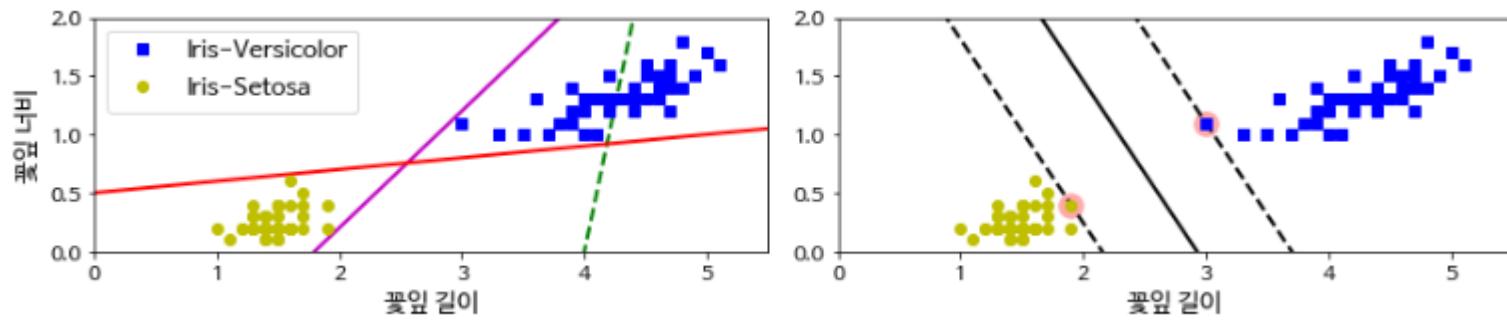
```
In [36]: title = "Learning Curves (Logistic Regression)"  
cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)
```

```
In [*]: import myUtil as myutil  
myutil.plot_ml_curve(m_lr, title, X, y, ylim=(0.6, 1.01), cv=cv)
```

로지스틱 회귀



SVM



Support Vector Machine (SVM)

```
sv = svm.SVC()
sv.fit(X_train,y_train)
pred = sv.predict(X_test)
acc_svm = metrics.accuracy_score(pred,y_test)
print('The accuracy of the SVM is:', acc_svm)
```

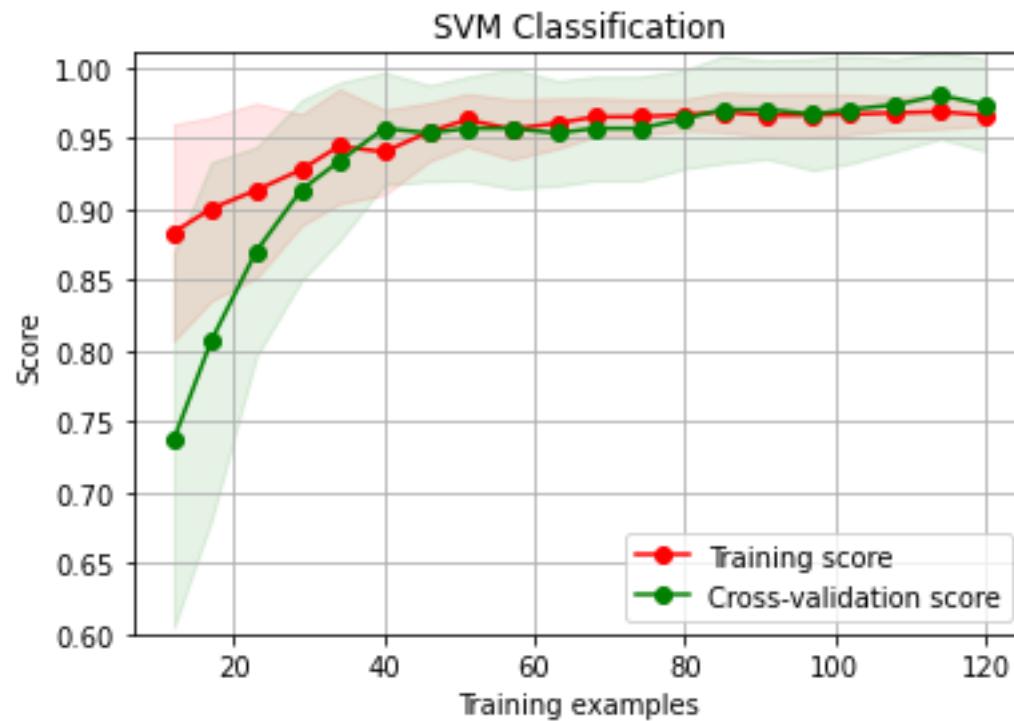
The accuracy of the SVM is: 0.9666666666666667

```
%%time
title = "SVM Classification"
cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)

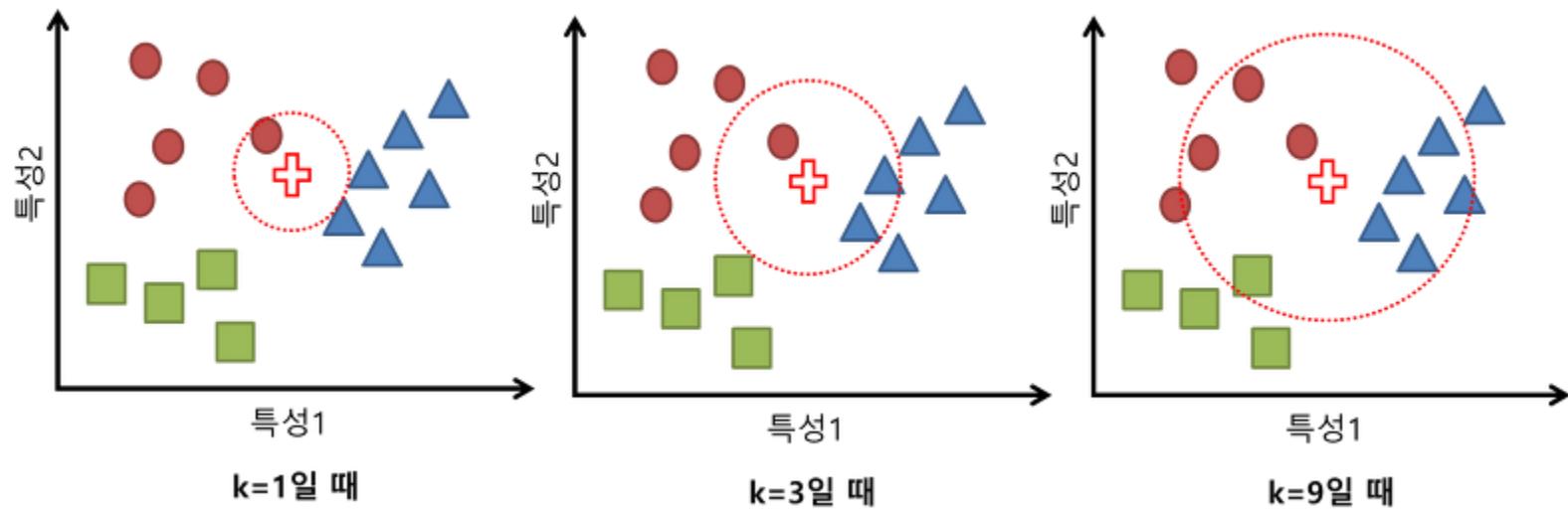
myutil.plot_ml_curve(sv, title, X, y, ylim=(0.6, 1.01), cv=cv )
```

Wall time: 1.39 s

Support Vector Machine (SVM)



KNN



K-Nearest Neighbours

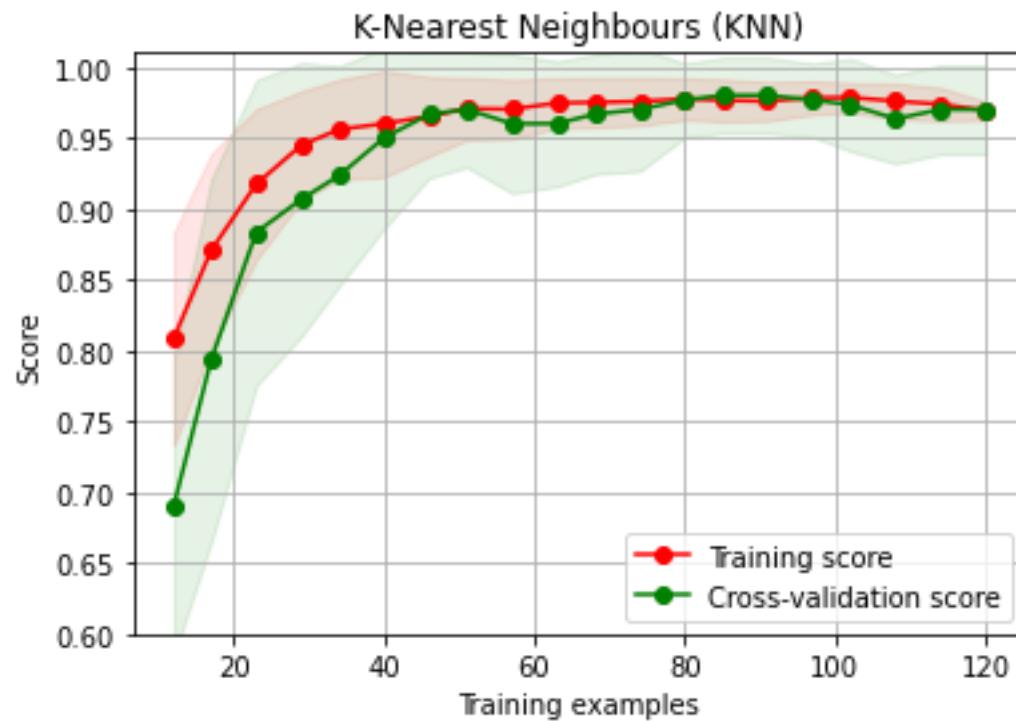
```
In [41]: knc = KNeighborsClassifier(n_neighbors=6)
knc.fit(X_train,y_train)
pred = knc.predict(X_test)
acc_knn = metrics.accuracy_score(pred,y_test)
print('The accuracy of the KNN is', acc_knn)
```

The accuracy of the KNN is 0.9666666666666667

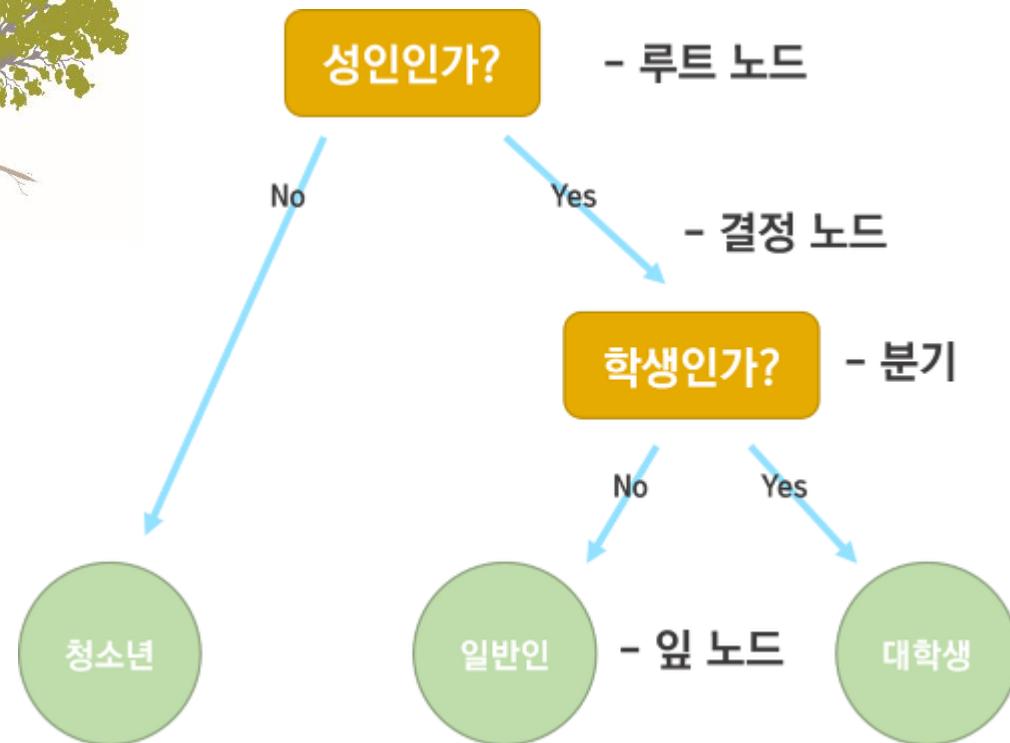
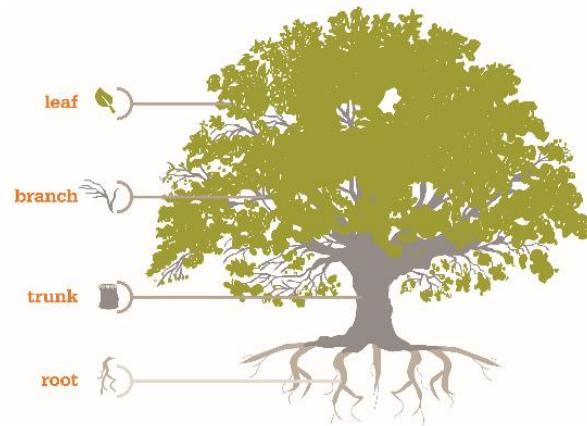
```
In [42]: title = "K-Nearest Neighbours (KNN)"
cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)

myutil.plot_ml_curve(knc, title, X, y, ylim=(0.6, 1.01), cv=cv, n_jobs=4)
```

K-Nearest Neighbours



의사결정나무



의사결정나무 (DecisionTreeClassifier)

```
In [49]: m_tree = DecisionTreeClassifier()  
m_tree.fit(X_train, y_train)
```

```
Out[49]: DecisionTreeClassifier()
```

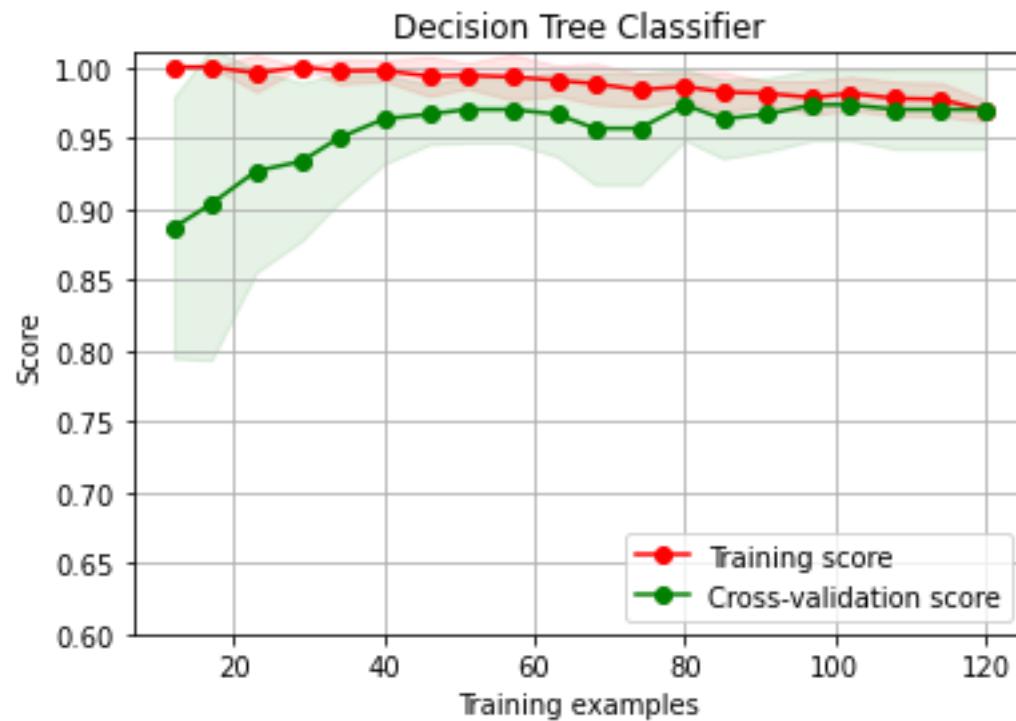
```
In [50]: prd = m_tree.predict(X_test)
```

```
In [51]: acc_dt = metrics.accuracy_score(pred,y_test)  
print('The accuracy of the SVM is:', acc_dt)
```

```
The accuracy of the SVM is: 0.9333333333333333
```

```
In [52]: title = "Decision Tree Classifier"  
cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)  
  
myutil.plot_ml_curve(m_rf, title, X, y, ylim=(0.6, 1.01), cv=cv)
```

의사결정나무 (DecisionTreeClassifier)



딥러닝

```
In [56]: model = keras.Sequential([
    keras.layers.Dense(64, activation = 'relu', input_shape=[4]),
    keras.layers.Dense(32, activation = 'relu'),
    keras.layers.Dense(16, activation = 'relu'),
    keras.layers.Dense(3, activation = 'softmax')
])

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

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	320
dense_1 (Dense)	(None, 32)	2080
dense_2 (Dense)	(None, 16)	528
dense_3 (Dense)	(None, 3)	51

Total params: 2,979

Trainable params: 2,979

Non-trainable params: 0

```
In [59]: model.evaluate(X_test, y_test)
```

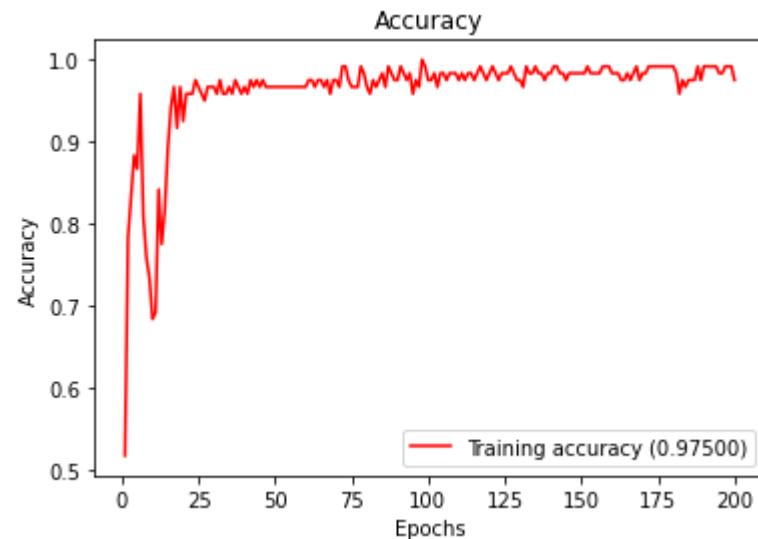
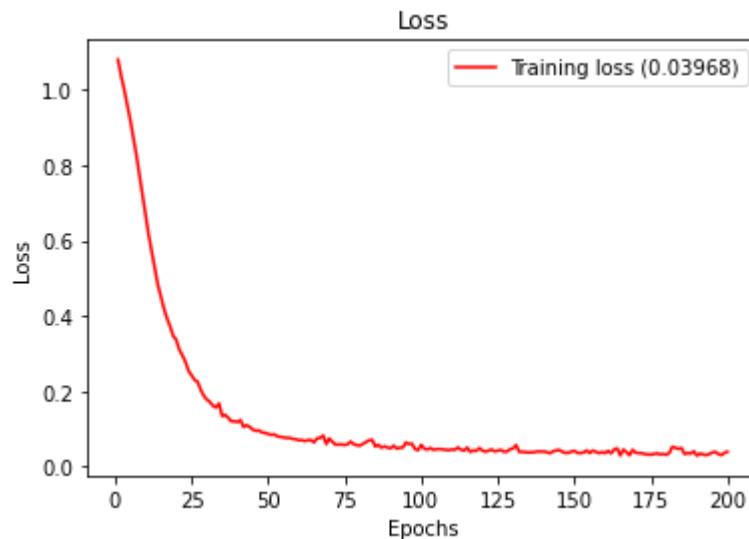
```
1/1 [=====] - 0s 157ms/step - loss: 0.1757 - accuracy: 0.9333
```

```
Out[59]: [0.1756865233182907, 0.9333333373069763]
```

```
In [60]: acc_dnn = history.history['accuracy'][np.argmin(history.history['loss'])]
print('The accuracy of the Deep Learning is:', acc_dnn)
```

```
The accuracy of the Deep Learning is: 0.9916666746139526
```

```
In [63]: myutil.plot_history(history)
```



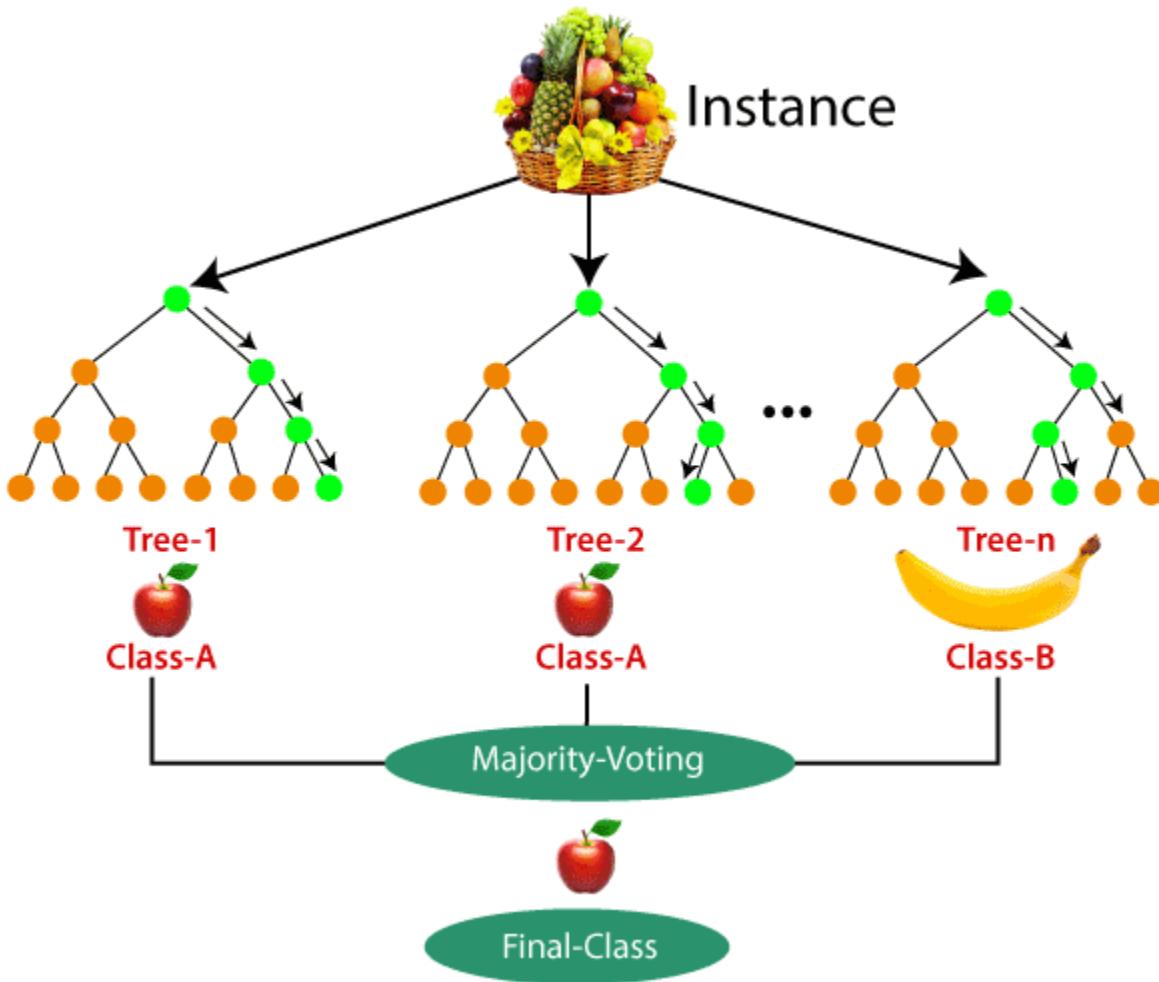
결과 정리

```
In [61]: models = pd.DataFrame({  
    'Model': ['Logistic Regression', 'Support Vector Machines', 'RandomForest',  
              'K-Nearest Neighbours', 'Decision Tree', 'Deep Learning'],  
    'Score': [acc_lr, acc_svm, acc_rf, acc_knn, acc_dt, acc_dnn]})  
models.sort_values(by='Score', ascending=False)
```

Out[61]:

	Model	Score
5	Deep Learning	0.991667
1	Support Vector Machines	0.966667
3	K-Nearest Neighbours	0.966667
0	Logistic Regression	0.933333
2	RandomForest	0.933333
4	Decision Tree	0.933333

Random Forest Classifier



랜덤포레스트 (RandomForestClassifier)

```
In [44]: m_rf = RandomForestClassifier(n_estimators=100, max_depth = 3)
```

```
In [45]: m_rf.fit(X_train, y_train)
```

```
Out[45]: RandomForestClassifier(max_depth=3)
```

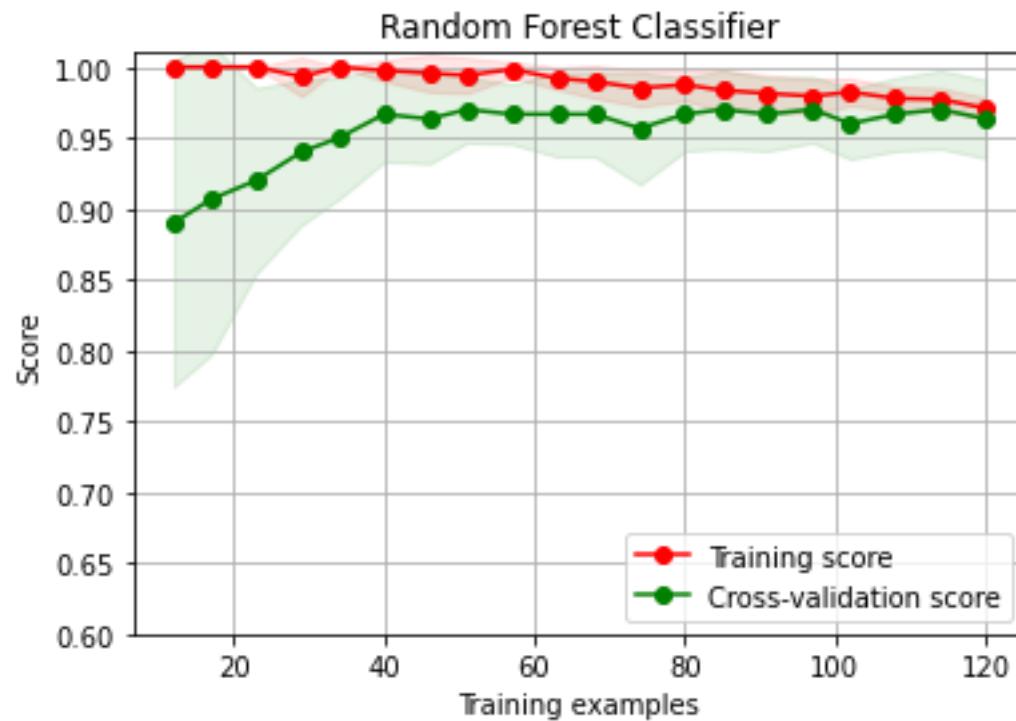
```
In [46]: pred = m_rf.predict(X_test)
```

```
In [47]: acc_rf = metrics.accuracy_score(pred,y_test)
print('The accuracy of the SVM is:', acc_rf)
```

```
The accuracy of the SVM is: 0.9333333333333333
```

```
In [48]: title = "Random Forest Classifier"
cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)
myutil.plot_ml_curve(m_rf, title, X, y, ylim=(0.6, 1.01), cv=cv)
```

랜덤포레스트 (RandomForestClassifier)



Thank You!

www.ust.ac.kr