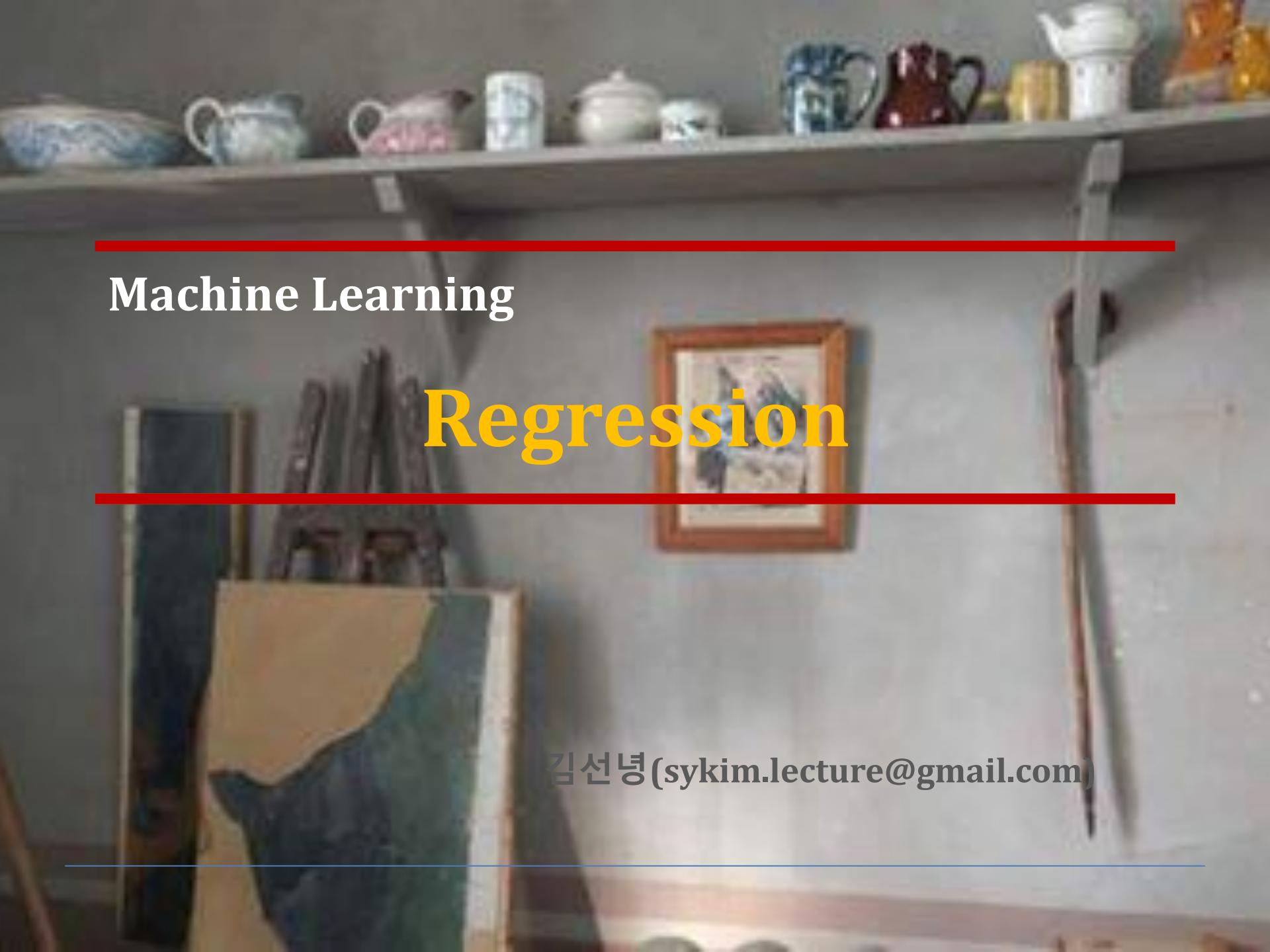


Machine Learning

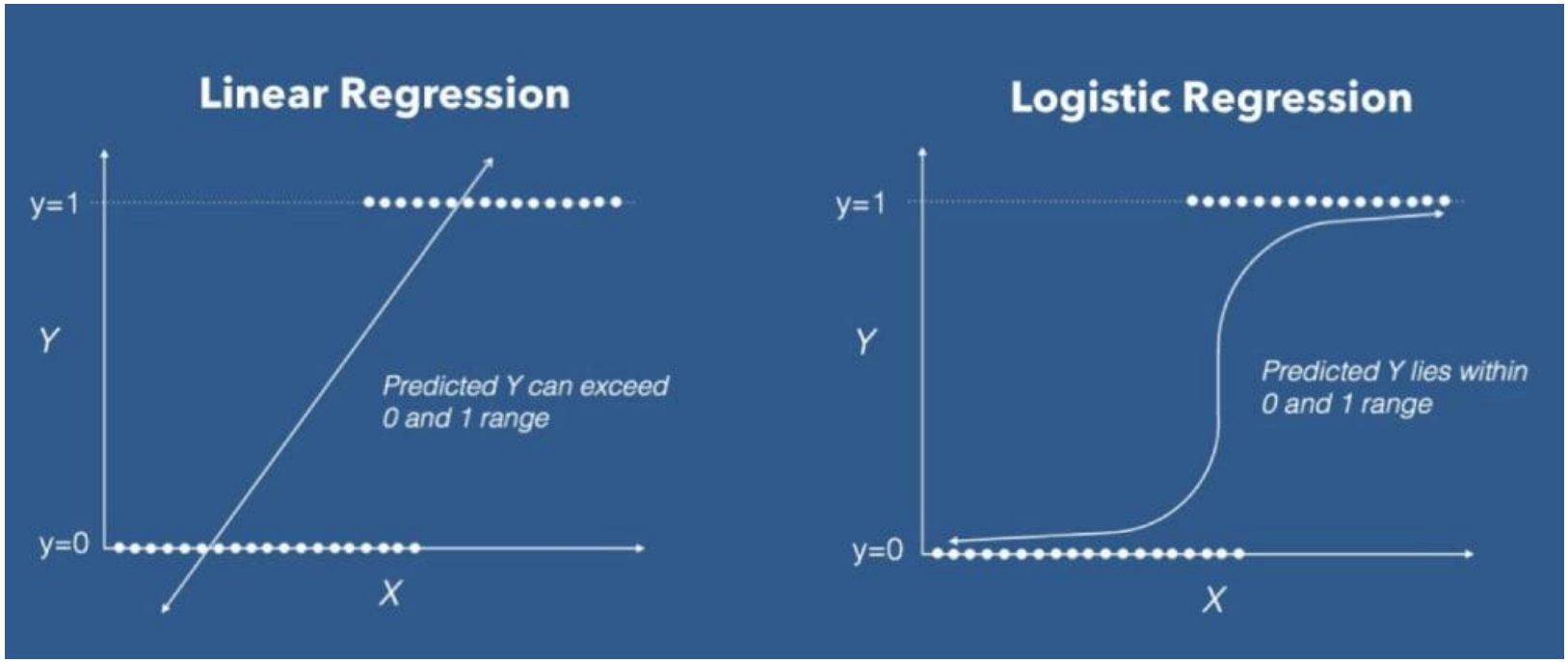
Regression

김선녕(sykim.lecture@gmail.com)


Logistic Regression
Binary Classification
Softmax Regression

개요

- 회귀를 사용하여 데이터가 어떤 **범주**에 속할 확률을 0에서 1사이의 값으로 예측
- 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해주는 알고리즘
- 로지스틱 회귀 분석은 **이진 분류**를 수행하는 데 사용.
  - 데이터 샘플을 양성(1) 또는 음성(0) 클래스 둘 중 어디에 속하는지 예측한다.
- 로지스틱 회귀 모델의 확률추정
  - $\hat{y} = h_w(x) = \sigma(w^T x)$



https://dev.to/adityaberi8/logistic-regression-eg1

Odds

- Odds
  - 임의의 사건 A가 발생하지 않을 확률 대비 일어날 확률의 비율을 뜻하는 개념
  - 실패에 비해 성공할 확률의 비.

$$odds = \frac{P(A)}{1 - P(A)}$$

- (예) 게임에서 이길 확률 $1/5$

$$odds = \frac{1/5}{1 - 1/5} = \frac{1}{4}$$

(5번 게임에서 4번 질 동안 1번 이긴다)

example

	의약품 A	의약품 B	합계
생존율(0)	32	24	56
생존율(1)	20	42	62
	52	66	118

- $Odds(A)$
  - $P(A) = 20/52 = 0.38$
  - $Odds(A) = 0.38/1 - 0.38 = 0.61$
  - A를 복용하면 100명 사망할 동안 61명 생존
- $Odds(B)$
  - $P(B) = 42/66 = 0.63$
  - $Odds(B) = 0.63/1 - 0.63 = 1.7$
  - B를 복용하면 100명 사망할 동안 170명 생존
- B에 대한 A의 $Odds\ ratio = 0.61/1.7 = 0.36$
  - B에 비해 A일 때 생존(성공)이 0.36배 = 64%가 생존율(성공율)이 떨어진다

Logistic Function

$$odds = \frac{p}{1-p}$$

· $0 < p < 1, \quad 0 < 1-p < 1$

· p가 0에 가까우면 $\frac{0}{1-0} = 0$, p가 1에 가까우면 $\frac{1}{1-1} = \infty$(무한대)

· 음의 무한대를 포함시키기 위하여 log를 취한다
입력 값의 범위가 $[0,1]$일 때 출력 값의 범위를 $[-\infty, \infty]$로 조정

$$-\infty < \log(\frac{p}{1-p}) < \infty$$

[참고]

- $\log_e 0 = ?$: 정의되지 않는다. $e^x = 0$를 만족시키는 x는 없다
- x가 양의 변 $(0+)$에서 0에 가까워 질 때 x의 자연 로그 한계는 마이너스 무한대

$$\lim_{x \to 0} \log_e x = -\infty$$

Logistic Function

$linear\ regression:\ y_i = \boldsymbol{\beta_0 + \beta_1 x_i + \epsilon_i}$

$logistic\ regression:\ \log\left(\frac{p}{1-p}\right) = \boldsymbol{\beta_0 + \beta_1 x_i + \epsilon_i}$

$$y = \beta_0 + \beta_1 x = W \cdot x$$

$$log\,\frac{p}{1-p} = W \cdot x$$

$$\frac{p}{1-p} = e^{Wx}$$

$$p = e^{Wx}(1-p) = e^{Wx} - e^{Wx}p$$

$$p + e^{Wx}p = e^{Wx}, \quad p(1+e^{Wx}) = e^{Wx}$$

$$\boldsymbol{p = \frac{e^{Wx}}{1+e^{Wx}} = \frac{1}{1+e^{-Wx}}}$$

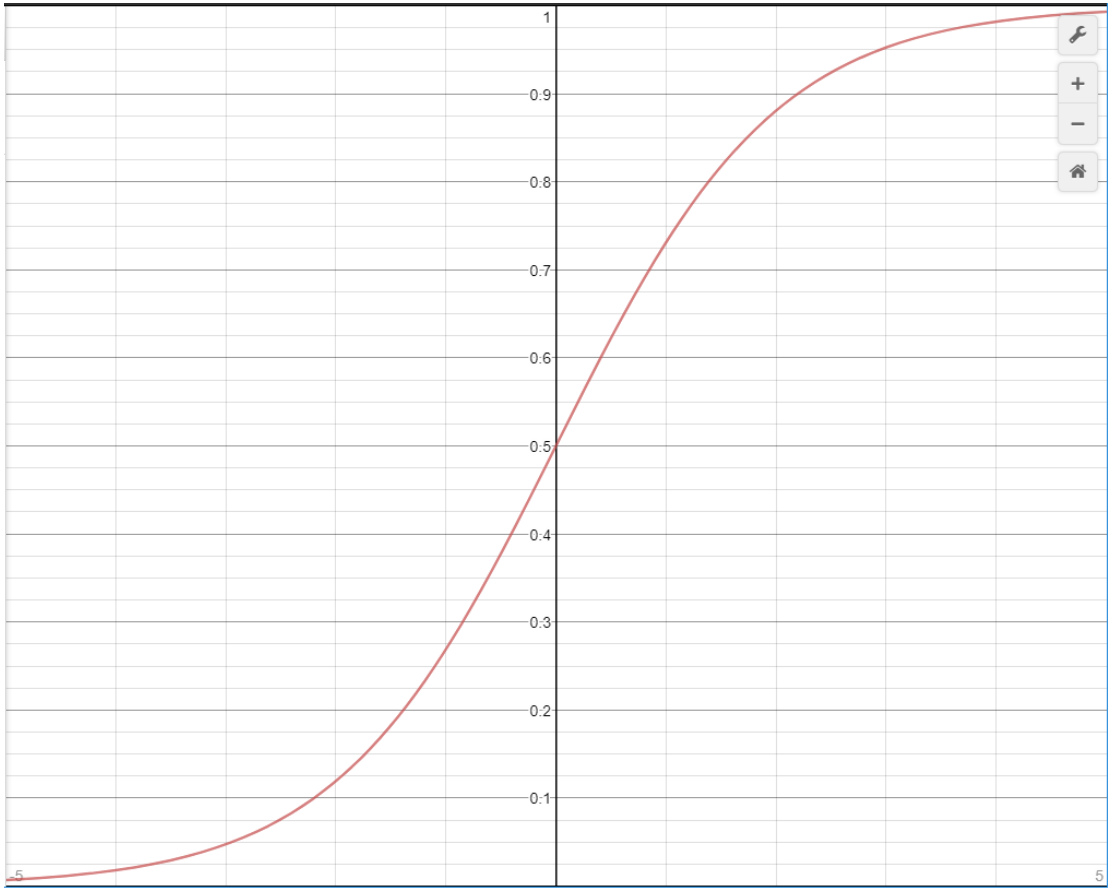
$cf: y = a^x \Rightarrow x = log_a y, \quad y = e^x \Rightarrow x = log_e y$

Sigmoid Function/Logistic Function

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

- $0 < \sigma(x) < 1$
- e : $2.7182 \cdots$ (자연상수)
- $\sigma(1) = 0.731 \ldots$
- $\sigma(2) = 0.880 \ldots$
- 회귀모델예측

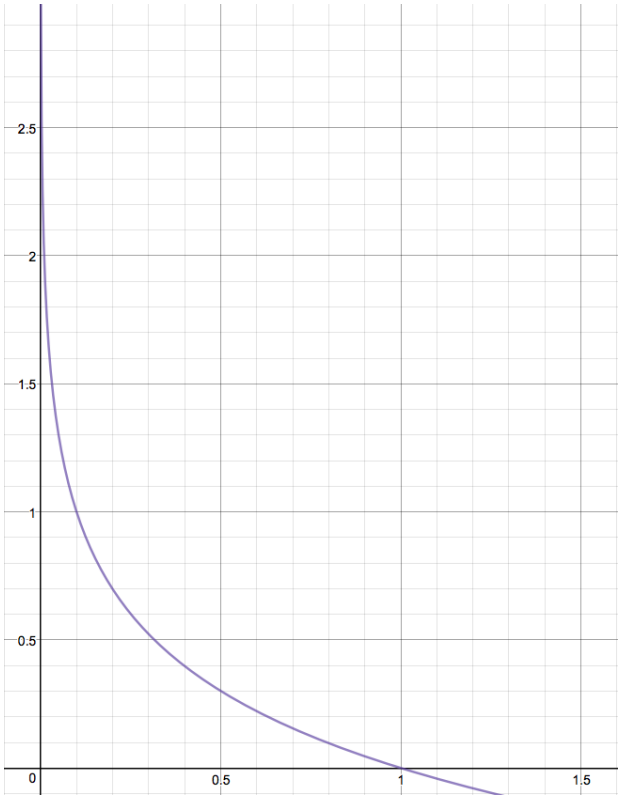
$$\hat{y} = \begin{cases} 0 & if \quad \sigma(x) < 0.5 \\ 1 & if \quad \sigma(x) \geq 0.5 \end{cases}$$

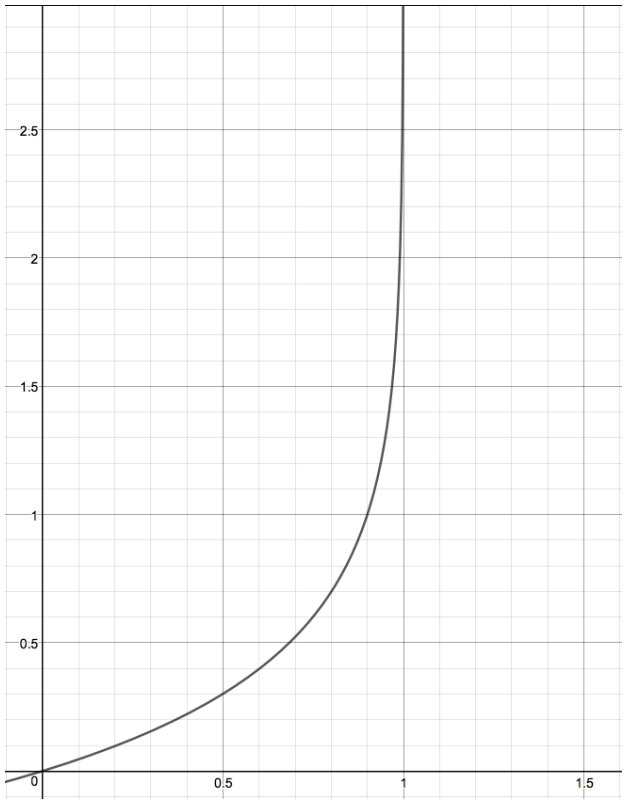


Cost function

양성샘플($y = 1$)에 대해서는 높은 확률을, 음성샘플($y = 0$)에 대해서는 낮은 확률을 추정하는 모델의 파라미터 θ를 찾는 것

$$cost(\theta) = \begin{cases} -\log(\hat{y}) & if\ y = 1 \\ -\log(1 - \hat{y}) & if\ y = 0 \end{cases}$$





$y = 1$ $\hat{y} = 1$, $cost(\theta) = 0$
$\hat{y} = 0$, $cost(\theta) = \infty$

$y = 0$ $\hat{y} = 0$, $cost(\theta) = 0$
$\hat{y} = 1$, $cost(\theta) = \infty$

Cost function of Logistic Regression

- 하나의 훈련 샘플에 대한 비용함수

$$cost(\theta) = \begin{cases} -log(\hat{y}) & if\ y = 1 \\ -log(1-\hat{y}) & if\ y = 0 \end{cases}$$

- 전체 훈련세트에 대한 비용함수(log loss, cross entropy)

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y_i log(\hat{y}_i) + (1-y_i)\, log(1-\hat{y}_i)]$$

- 로지스틱 회귀 모델 훈련
  - 최솟값을 계산하는 알려진 해가 없다.
  - 하지만 위 비용함수는 블록 함수이므로 경사 하강법이 전역 최솟값을 찾는 것을 보장한다

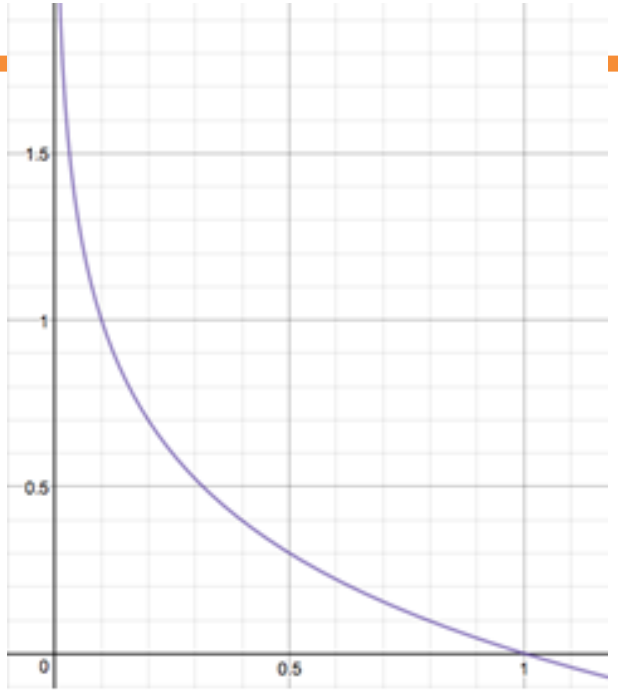
$$\frac{\delta}{\delta\theta}J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\sigma(\theta x_i) - y_i)\, x_i$$

$\therefore\ y = 1,\quad f = -log(\hat{y})$ X

$\therefore\ y = 0,\quad f =$ X $-1 * log(1-\hat{y})$

예제

$$D(\hat{y}, y) = -\sum_{i}^{N} y_i \, log(\hat{y}_i)$$



- 실제값$(y) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ 이다. 예측해 보자.

  $\hat{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, true인 경우, $\begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot -\log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow 0 + 0 = \mathbf{0}$ (Σ이므로)

  $\hat{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, false인 경우, $\begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot -\log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ \infty \end{bmatrix} \rightarrow 0 + \infty = \infty$

- 실제값$(y) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ 일 때

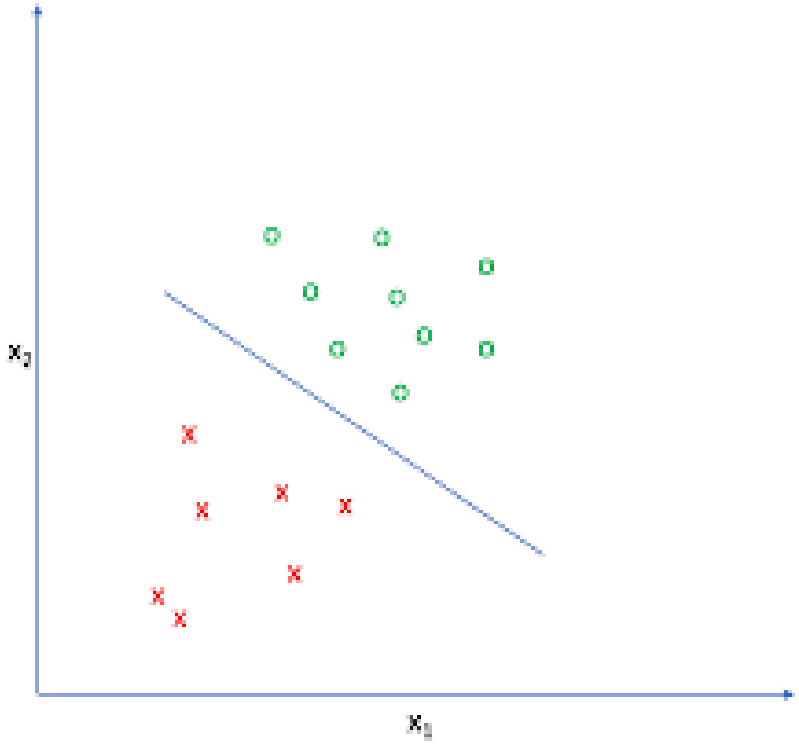
  $\hat{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, true인 경우, $\begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot -\log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow 0 + 0 = \mathbf{0}$

  $\hat{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, false인 경우, $\begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot -\log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} \infty \\ 0 \end{bmatrix} \rightarrow \infty + 0 = \infty$

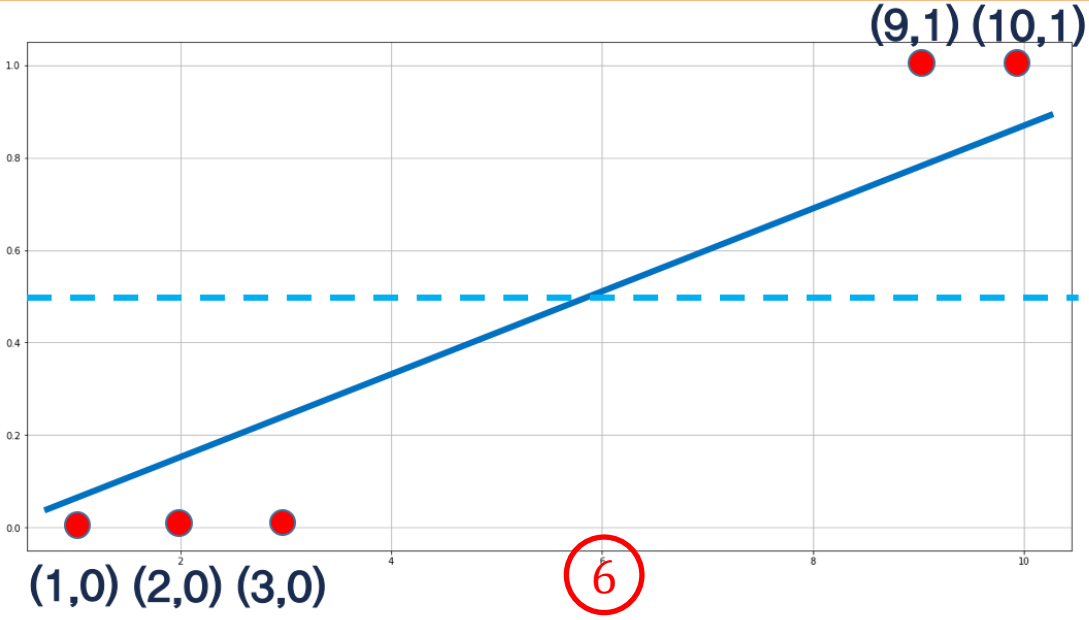

Logistic Regression
Binary Classification
Softmax Regression

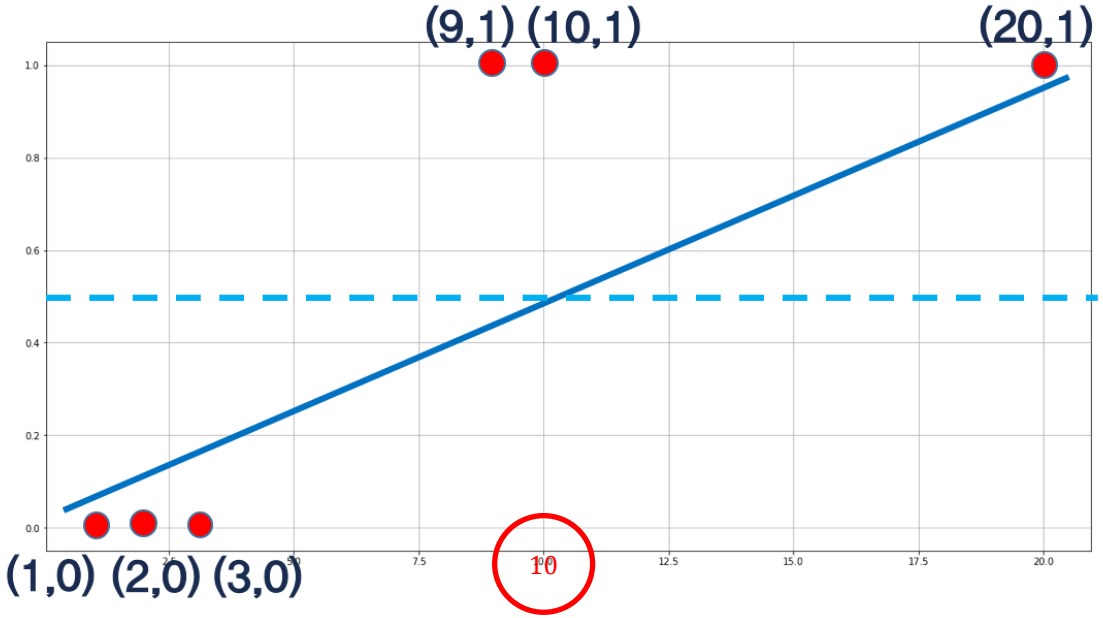
이진 분류(Binary Classification)

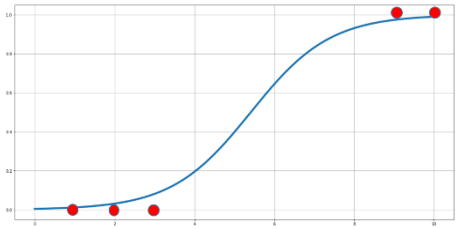
- 학생이 문제를 맞은 것인지 틀린 것인지?
- 내일 비가 올지 안 올지?
- 이메일 : Spam or not?
- 레이블이 1 이면 양성클래스, 0이면 음성클래스
- **위의 모든 예측을 1 or 0 으로 Encoding(설정)**
  - o 과 x 를 구분할 수 있도록 학습하는 것



이진 분류(Binary Classification)

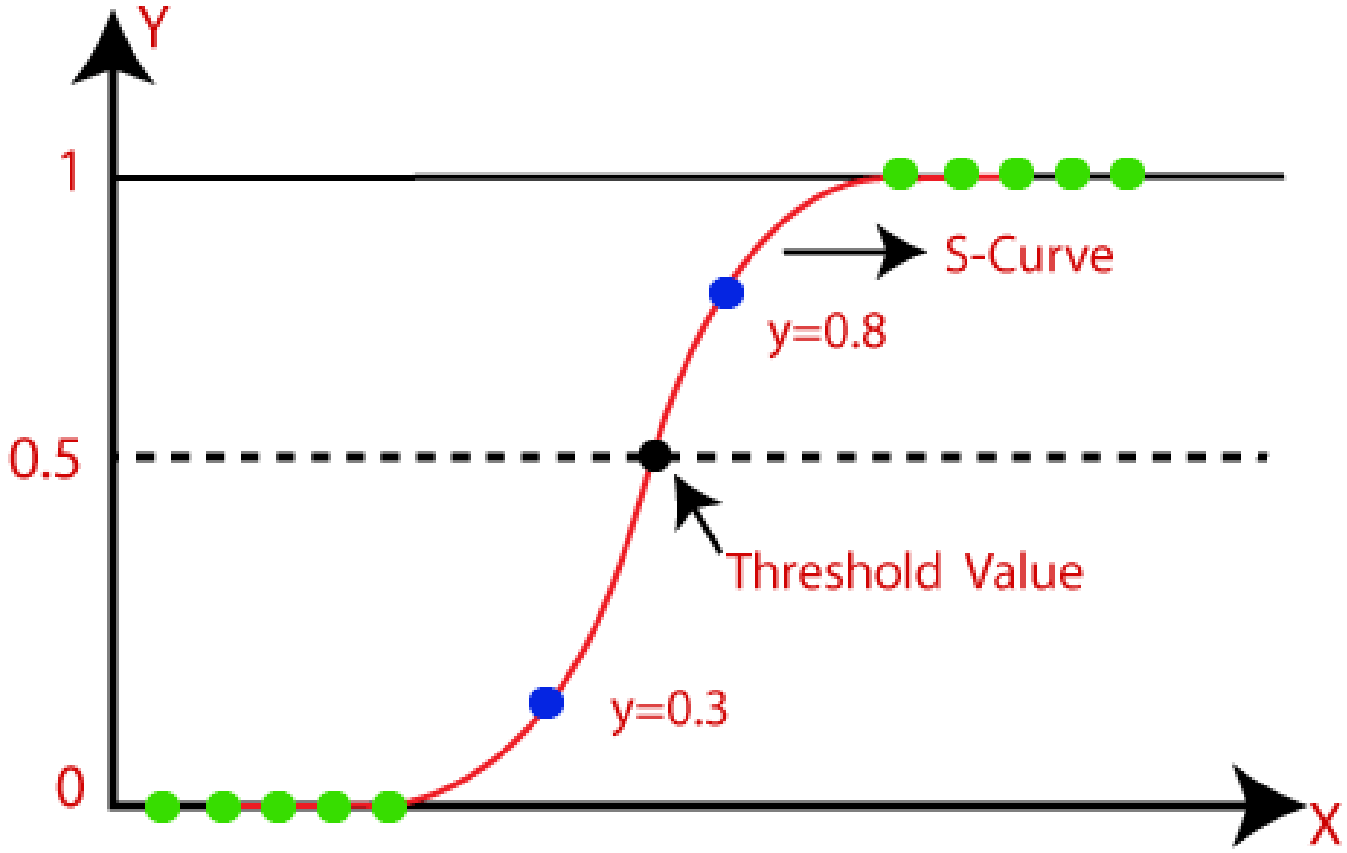






https://icim.nims.re.kr/post/easyMath/64

Threshold



https://www.javatpoint.com/logistic-regression-in-machine-learning

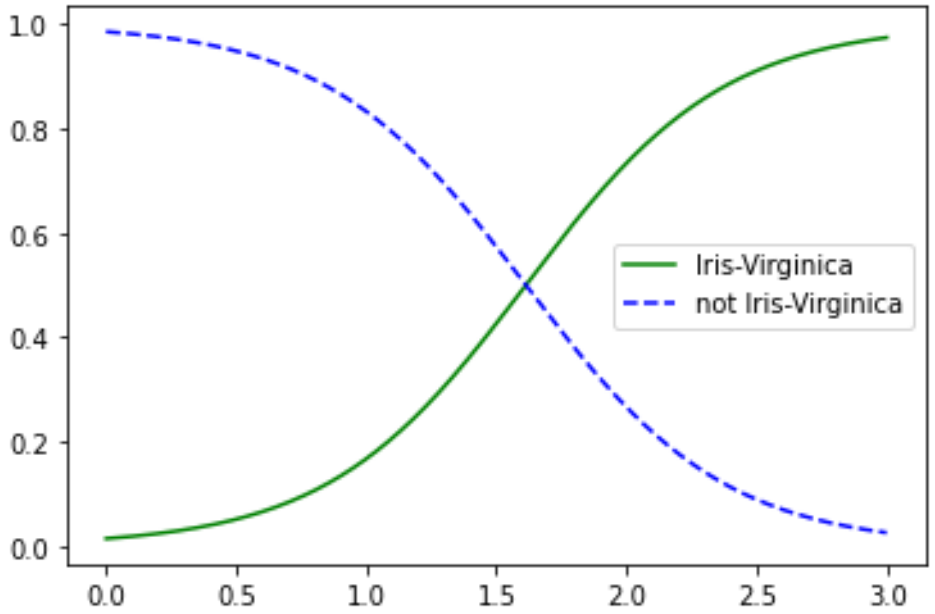
결정경계 - Iris Versinica 분류


```
# 붓꽃 데이터 로드
from sklearn import datasets

iris = datasets.load_iris()
list(iris.keys())
```

['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename']

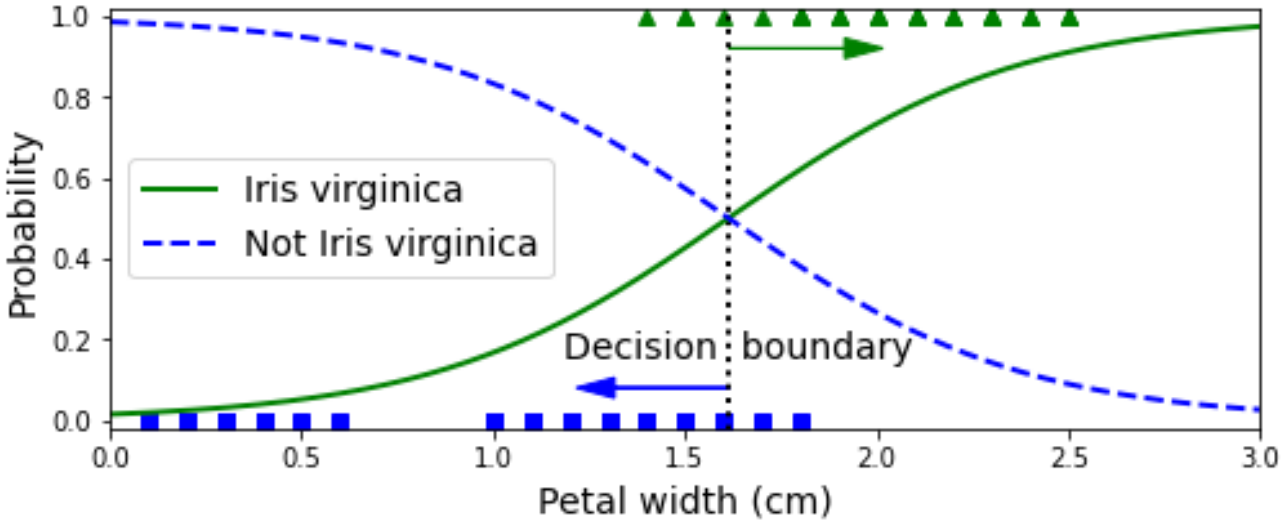
```
X = iris["data"][:,3:]    # 꽃잎의 너비만 사용
y = (iris["target"]==2).astype("int") #iris-Versinica이면 1,아니면 0

# 로지스틱 회귀모델 훈련
from sklearn.linear_model import LogisticRegression

# 향후 버전이 바뀌더라도 동일한 결과를 만들기 위해
# 사이킷런 0.22 버전의 기본값인 solver="lbfgs"로 지정
log_reg = LogisticRegression(solver="lbfgs", random_state=42)
log_reg.fit(X,y)
```

```
import matplotlib.pyplot as plt
import numpy as np

# 꽃잎의 너비가 0~3cm인 꽃에 대해 모델의 추정확률을 계산
X_new = np.linspace(0,3,1000).reshape(-1,1)
y_proba = log_reg.predict_proba(X_new)
plt.plot(X_new, y_proba[:,1], "g-", label = "Iris-Virginica")
plt.plot(X_new, y_proba[:,0], "b--", label = "not Iris-Virginica")
plt.legend()
plt.show()
```



```
X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)
decision_boundary = X_new[y_proba[:, 1] >= 0.5][0]

plt.figure(figsize=(8, 3))
plt.plot(X[y==0], y[y==0], "bs")  # 음성범주 pointing
plt.plot(X[y==1], y[y==1], "g^")  # 양성범주 pointing

# 결정경계 표시
plt.plot([decision_boundary, decision_boundary], [-1, 2], "k:", linewidth=2)

# 추정확률 plotting
plt.plot(X_new, y_proba[:, 1], "g-", linewidth=2, label="Iris virginica")
plt.plot(X_new, y_proba[:, 0], "b--
", linewidth=2, label="Not Iris virginica")

plt.text(decision_boundary+0.02, 0.15, "Decision  boundary", fontsize=14, co
lor="k", ha="center")
plt.arrow(decision_boundary, 0.08, -
0.3, 0, head_width=0.05, head_length=0.1, fc='b', ec='b')
plt.arrow(decision_boundary, 0.92, 0.3, 0, head_width=0.05, head_length=0.1,
 fc='g', ec='g')
plt.xlabel("Petal width (cm)", fontsize=14)
plt.ylabel("Probability", fontsize=14)
plt.legend(loc="center left", fontsize=14)
plt.axis([0, 3, -0.02, 1.02])

plt.show()
```

- Iris-Verginica(y=1)의 꽃잎 너비 : 1.4~2.5cm 사이에 분포(초록 삼각형)
- Iris-Verginica가 아닌 붓꽃의 꽃잎 너비 : 0.1~1.8cm에 분포(파란 사각형)
- 중첩되는 구간이 존재




```
decision_boundary
```

array([1.61561562])

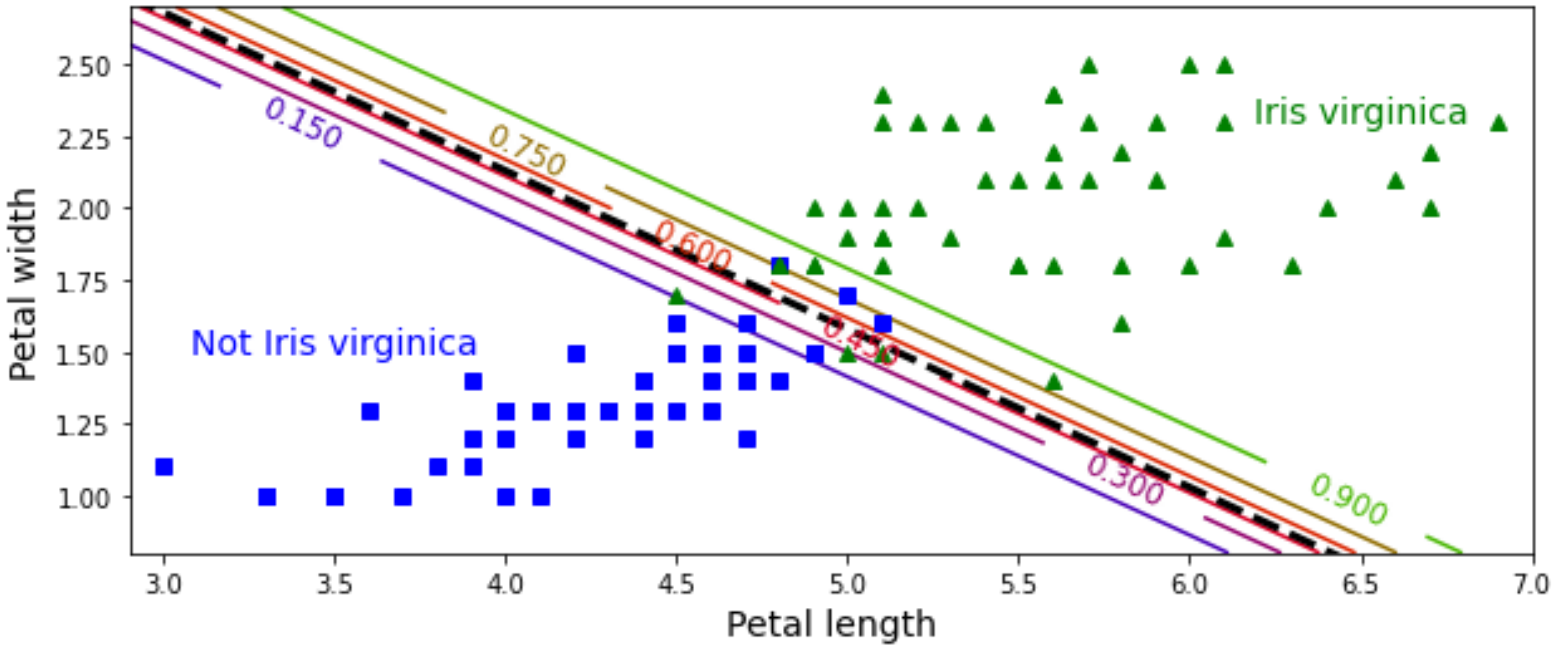
```
# 양쪽의 확률이 50%가 되는 1.6cm 근방에서 결정경계가 만들어지고,
# 분류기는 1.6cm보다 크면 Iris-Verginica로 분류하고 작으면 아니라고 예측한다.
log_reg.predict([[1.7], [1.5]])
```

array([1, 0])

꽃잎 너비와 꽃잎 길이 훈련


```
# 두번째 꽃잎 너비와 꽃잎 길이 2개의 특성을 이용해 훈련
from sklearn.linear_model import LogisticRegression

X = iris["data"][:, (2, 3)]  # petal length, petal width
y = (iris["target"] == 2).astype(np.int)

# LogisticRegression모델의 규제강도를 조절하는 하이퍼파라미터는 alpha가 아니라 그 역수에 해당하는 C이다. C가 높을수록 모델의 규제가 줄어든다.
log_reg = LogisticRegression(solver="lbfgs", C=10**10, random_state=42)
log_reg.fit(X, y)

x0, x1 = np.meshgrid(
        np.linspace(2.9, 7, 500).reshape(-1, 1),
        np.linspace(0.8, 2.7, 200).reshape(-1, 1),
    )
X_new = np.c_[x0.ravel(), x1.ravel()]

y_proba = log_reg.predict_proba(X_new)
```

```
plt.figure(figsize=(10, 4))
plt.plot(X[y==0, 0], X[y==0, 1], "bs")
plt.plot(X[y==1, 0], X[y==1, 1], "g^")

zz = y_proba[:, 1].reshape(x0.shape)
contour = plt.contour(x0, x1, zz, cmap=plt.cm.brg)

left_right = np.array([2.9, 7])
boundary = -
(log_reg.coef_[0][0] * left_right + log_reg.intercept_[0]) / log_reg.coef_[0
][1]

plt.clabel(contour, inline=1, fontsize=12)
plt.plot(left_right, boundary, "k--", linewidth=3)
plt.text(3.5, 1.5, "Not Iris virginica", fontsize=14, color="b", ha="center")
plt.text(6.5, 2.3, "Iris virginica", fontsize=14, color="g", ha="center")
plt.xlabel("Petal length", fontsize=14)
plt.ylabel("Petal width", fontsize=14)
plt.axis([2.9, 7, 0.8, 2.7])

plt.show()
```



- 점선은 모델이 50% 확률을 추정하는 지점으로, 이 모델의 결정경계이다.
- 이 경계는 $\theta_0 + \theta_1 x_1 + \theta_1 x_2 = 0$을 만족하는 (x_1, x_2)의 집합인 선형경계이다
- Iris-Verginica에 속할 확률을 15%부터 90%까지 나타내고 있다.

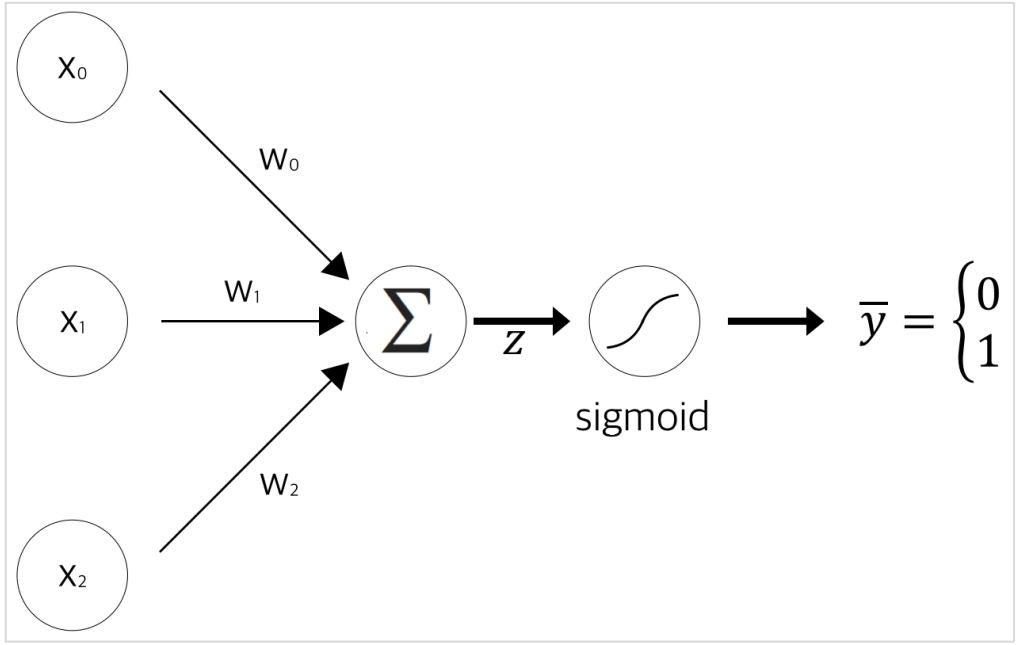
Logistic Regression

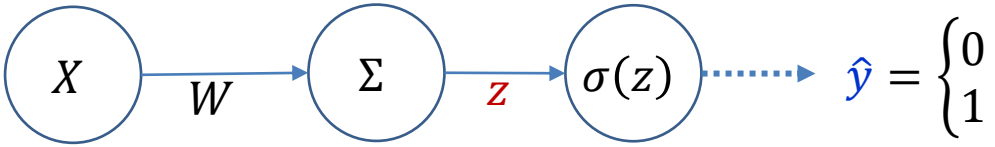
$$\hat{y} = Wx + b$$

$$z = Wx$$

$$\hat{y} = \frac{1}{1 + e^{-z}}$$

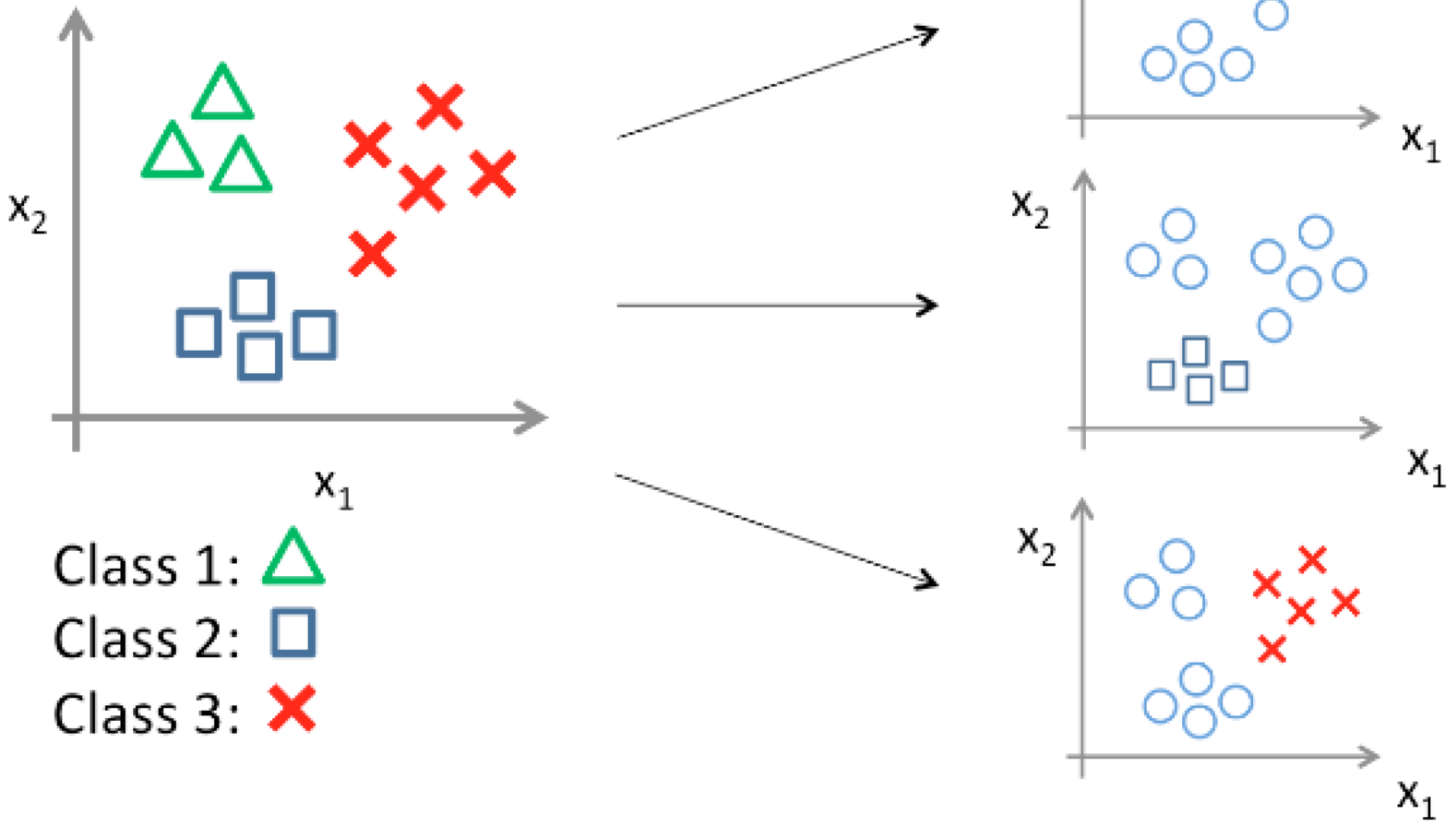
$$\hat{y} = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e}^{-\boldsymbol{Wx}}}$$





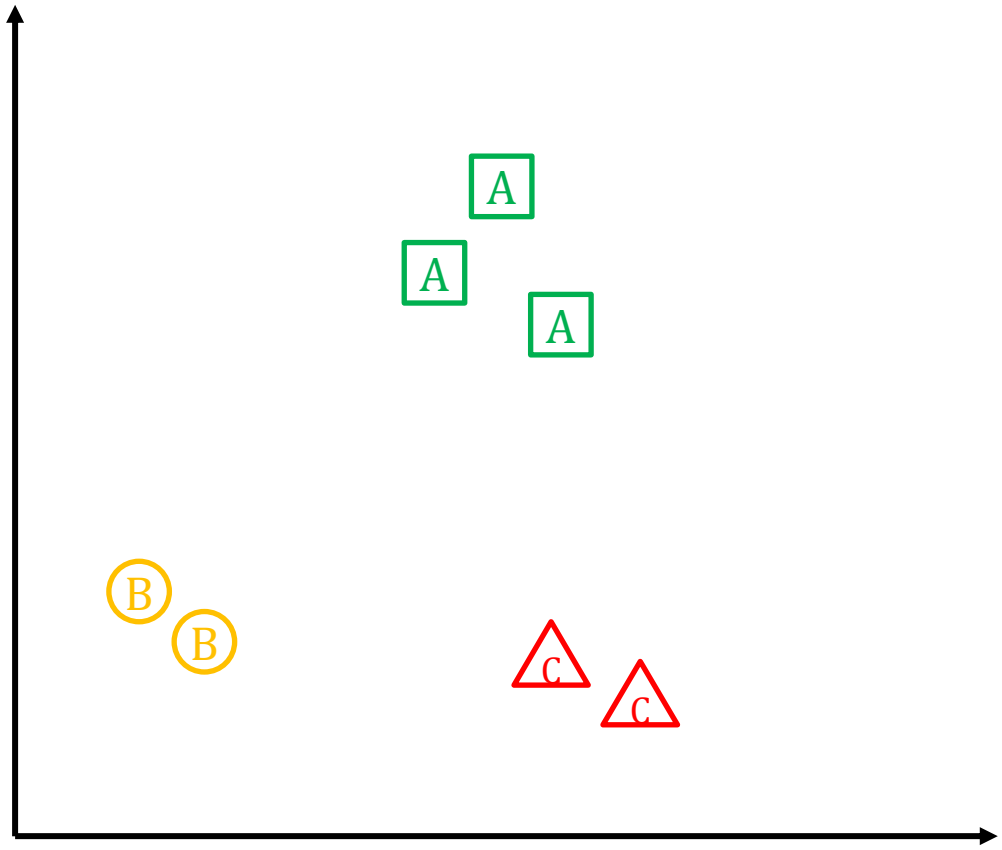
소프트맥스 회귀(다항 로지스틱 회귀)

One-vs-all (one-vs-rest):

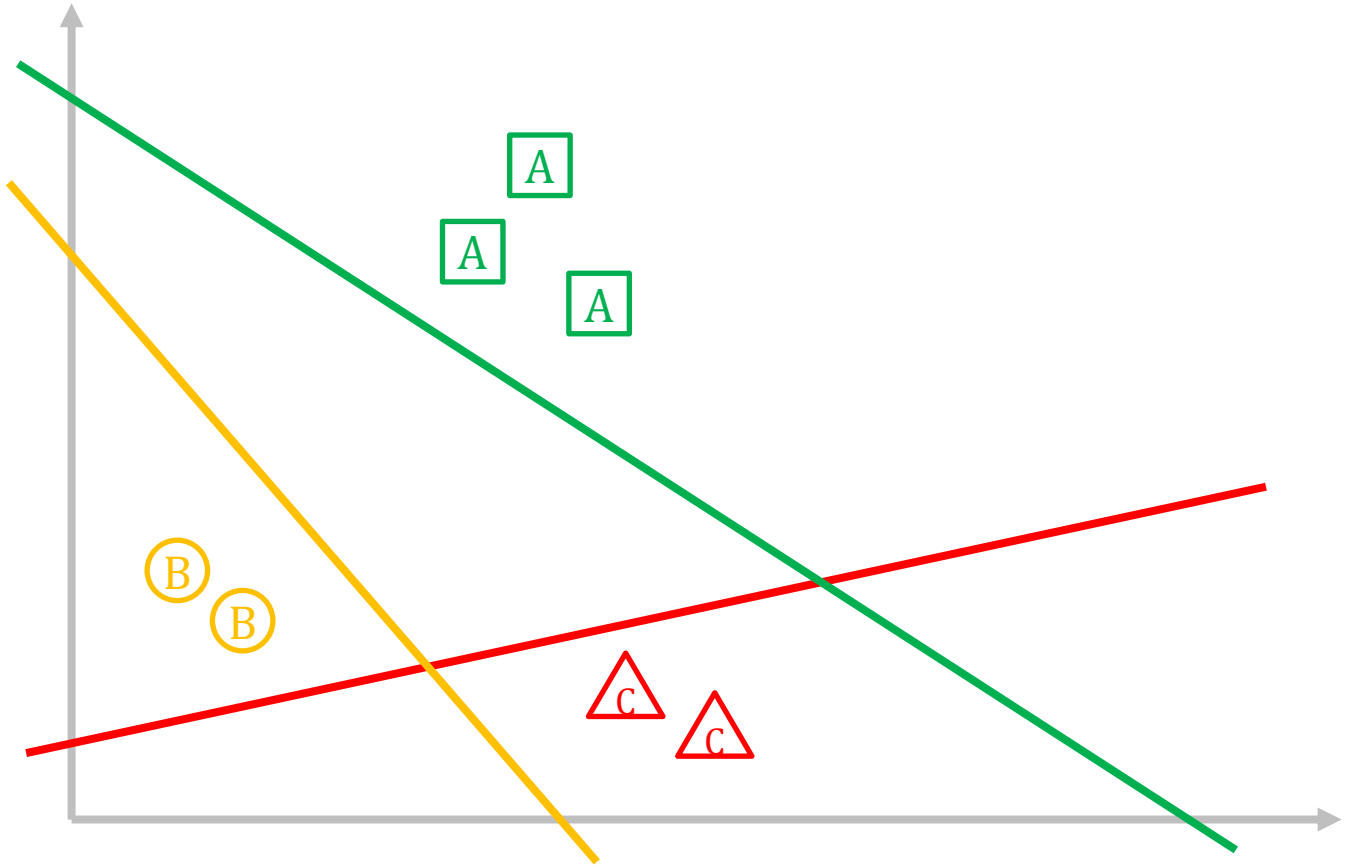


Example

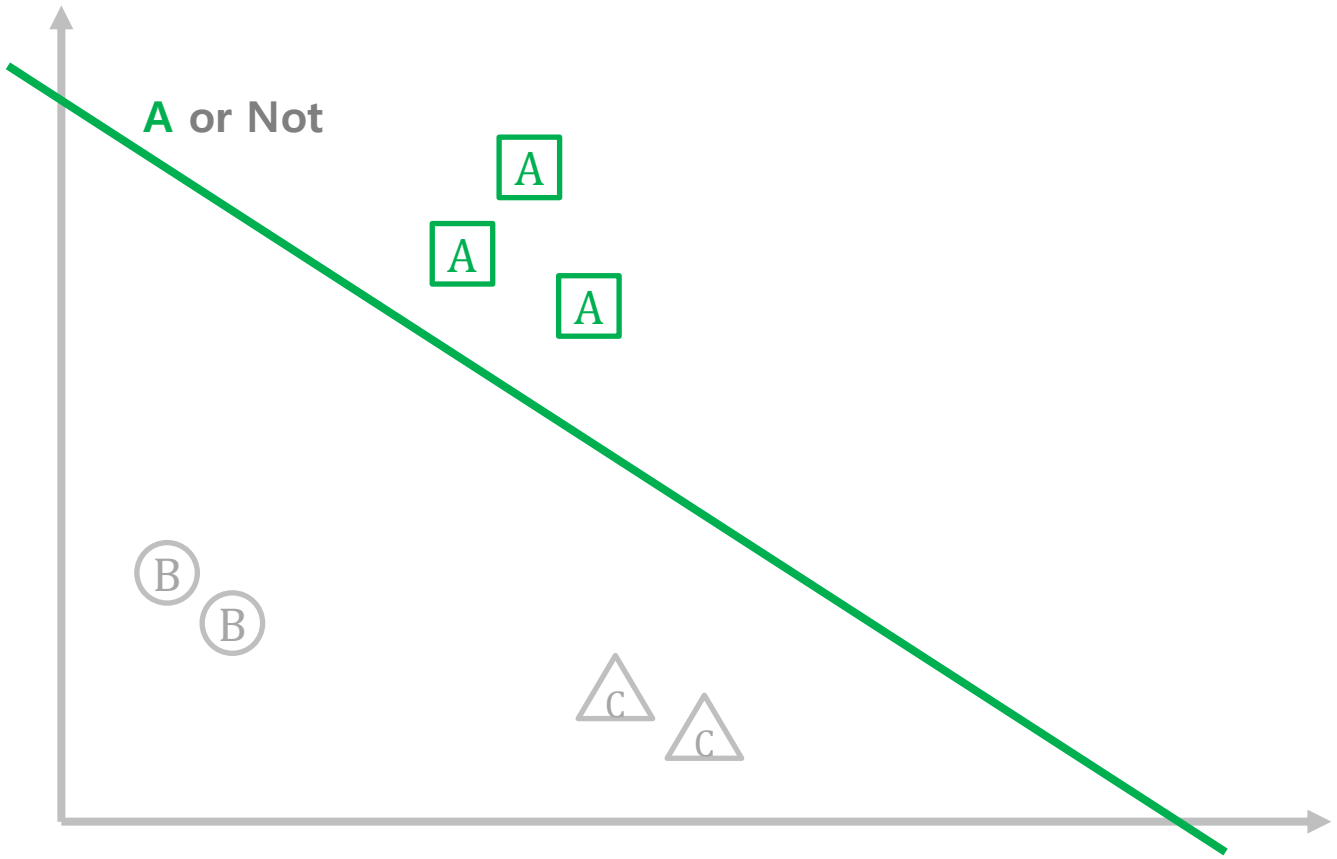
x1 (시험1)	x2 (시험2)	x3 (출석)	Y (학점)
100	100	5	A
90	100	5	A
80	80	4	A
70	90	4	B
90	80	2	B
50	50	3	C
70	50	2	C

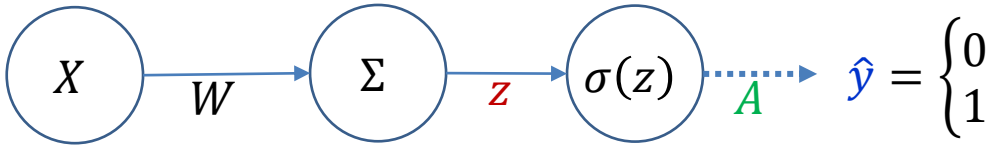


Example

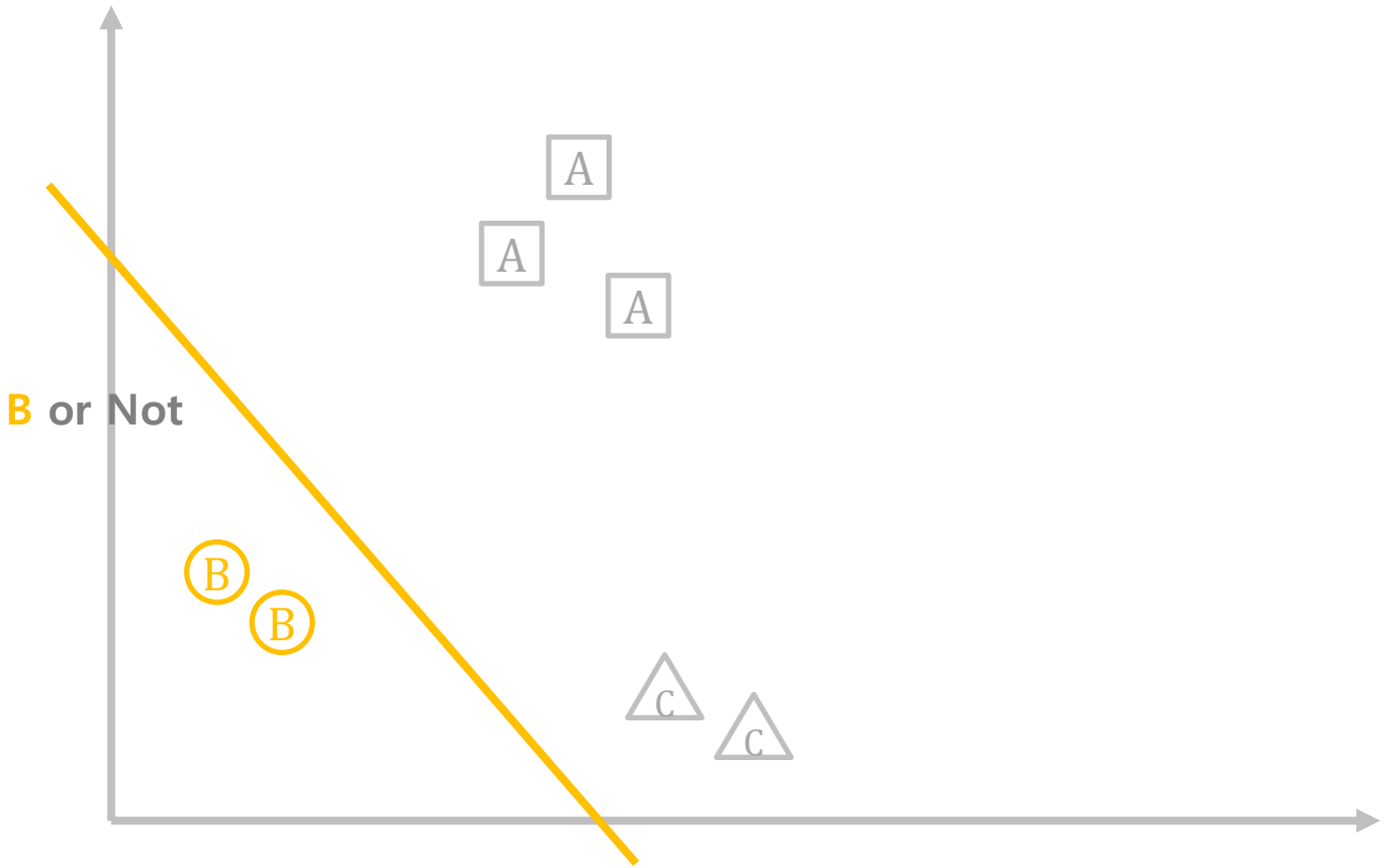


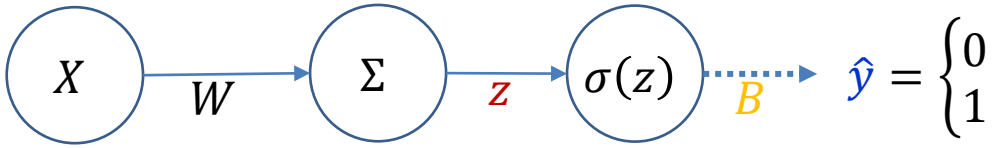
A or Not



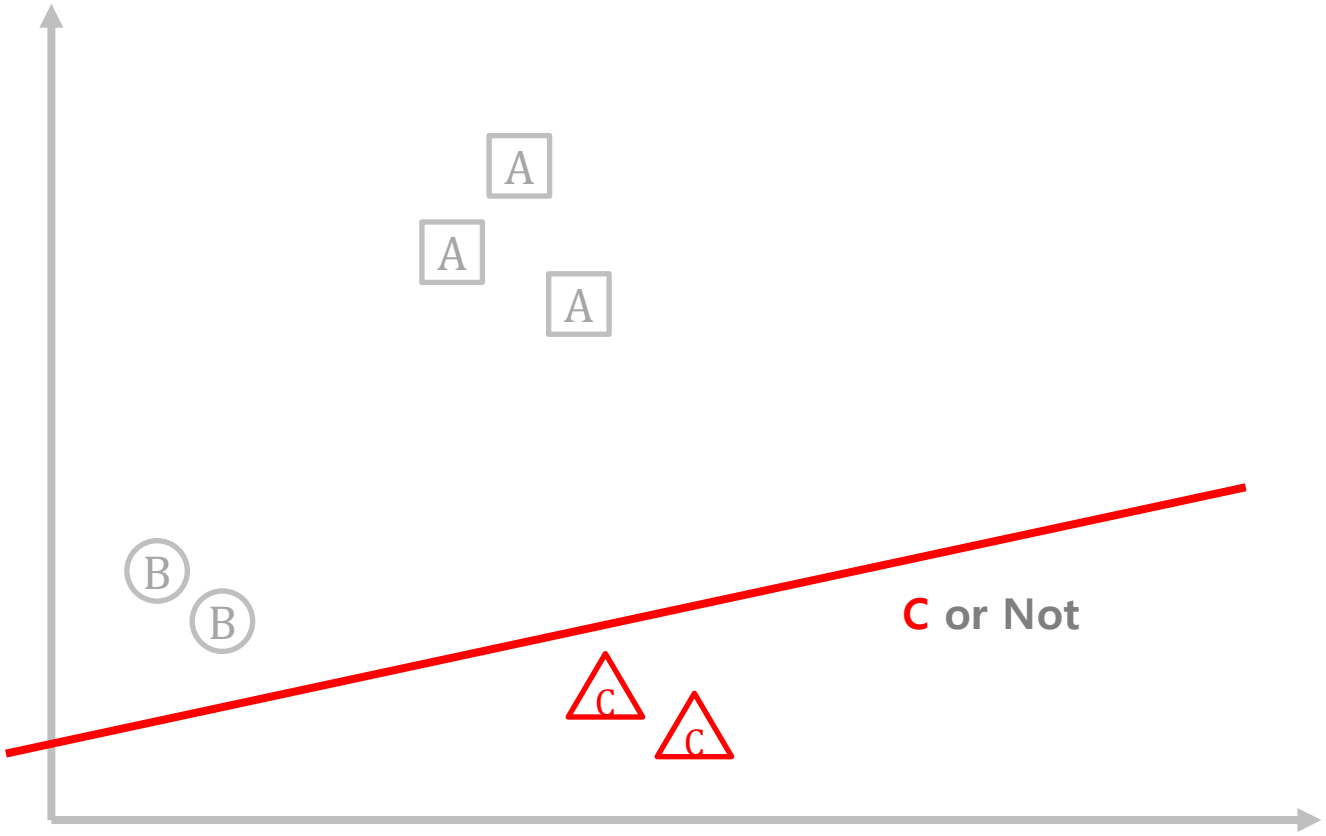


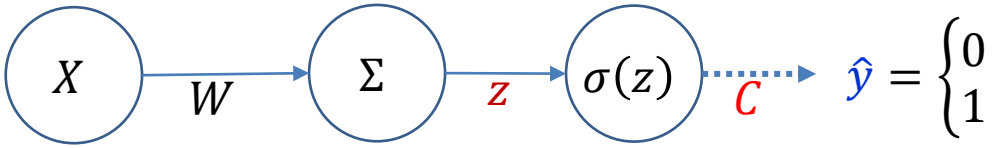
B or Not

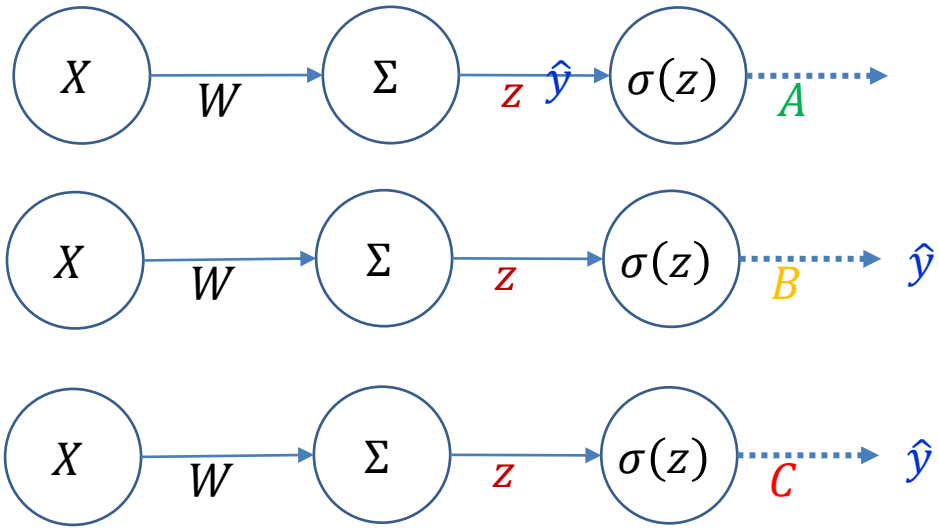




C or Not







$$[x_{A1} \quad x_{A2} \quad x_{A3}] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = [x_{A1}w_1 + x_{A2}w_2 + x_{A3}w_3]$$

$$[x_{B1} \quad x_{B2} \quad x_{B3}] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = [x_{B1}w_1 + x_{B2}w_2 + x_{B3}w_3]$$

$$[x_{C1} \quad x_{C2} \quad x_{c3}] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = [x_{C1}w_1 + x_{C2}w_2 + x_{C3}w_3]$$

$$\begin{bmatrix} x_{A1} & x_{A2} & x_{A3} \\ x_{B1} & x_{B2} & x_{B3} \\ x_{C1} & x_{C2} & x_{C3} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = [?]$$

$$\begin{bmatrix} x_{A1} & x_{A2} & x_{A3} \\ x_{B1} & x_{B2} & x_{B3} \\ x_{C1} & x_{C2} & x_{C3} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = [?]$$

$$\begin{bmatrix} x_{A1} & x_{A2} & x_{A3} \\ x_{B1} & x_{B2} & x_{B3} \\ x_{C1} & x_{C2} & x_{C3} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} x_{A1}w_1 + x_{A2}w_2 + x_{A3}w_3 \\ x_{B1}w_1 + x_{B2}w_2 + x_{B3}w_3 \\ x_{C1}w_1 + x_{C2}w_2 + x_{C3}w_3 \end{bmatrix} = \begin{bmatrix} \hat{y}_A \\ \hat{y}_B \\ \hat{y}_C \end{bmatrix} = \begin{bmatrix} \mathbf{2.0} \\ \mathbf{1.0} \\ \mathbf{0.1} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{2.0} \\ \mathbf{1.0} \\ \mathbf{0.1} \end{bmatrix} \begin{matrix} \rightarrow \ \boldsymbol{p} = \mathbf{0.7}\ A(\mathrm{O}) \\ \rightarrow \ \boldsymbol{p} = \mathbf{0.2}\ B(\times) \\ \rightarrow \ \boldsymbol{p} = \mathbf{0.1}\ C(\times) \end{matrix}$$

$$\therefore \hat{\boldsymbol{y}} = \boldsymbol{A}$$

소프트맥스 회귀

- 로지스틱 함수를 여러 개의 입력인 경우 로지스틱 함수를 사용할 수 있도록 일반화 한 것
- 이진 분류기를 훈련시켜 연결하지 않고 직접 다중 클래스를 지원할 수 있도록 일반화한 것
- 다항 로지스틱 회귀$(multinomial\ logistic\ regression)$라고도 한다
- 샘플에 대해 각 클래스의 점수가 계산되면 소프트맥스 함수를 통과시켜 해당되는 클래스에 속할 **확률을 추정한다**

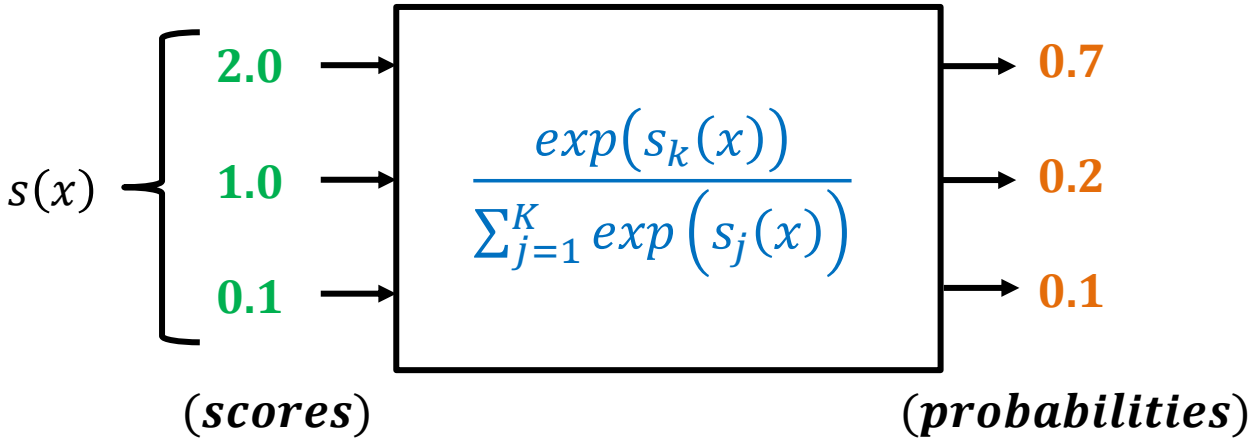
$$logistic\ function \xrightarrow{\text{일반화}(generalization)} softmax\ function$$

소프트맥스 함수(softmax function)

- 출력합계는 1. n차 실수 벡터를 0과 1 사이의 실수로 변환하여 출력

$$\sigma\big(s(x)\big)_k = \frac{exp\big(s_k(x)\big)}{\sum_{j=1}^{K} exp\left(s_j(x)\right)}$$

- K : 클래스의 수
- $s(x)$: 샘플 x에 대한 각 클래스의 점수를 담은 벡터
- $\sigma\big(s(x)\big)_k$: 샘플 x 에 대한 각 클래스의 점수가 주어졌을 때 이 샘플이 클래스 k에 속할 추정 **확률**



CIFAR10

- 다중선형회귀(Multivariable linear regression)

50,000 training images
each image is **32x32x3**

10,000 test images.

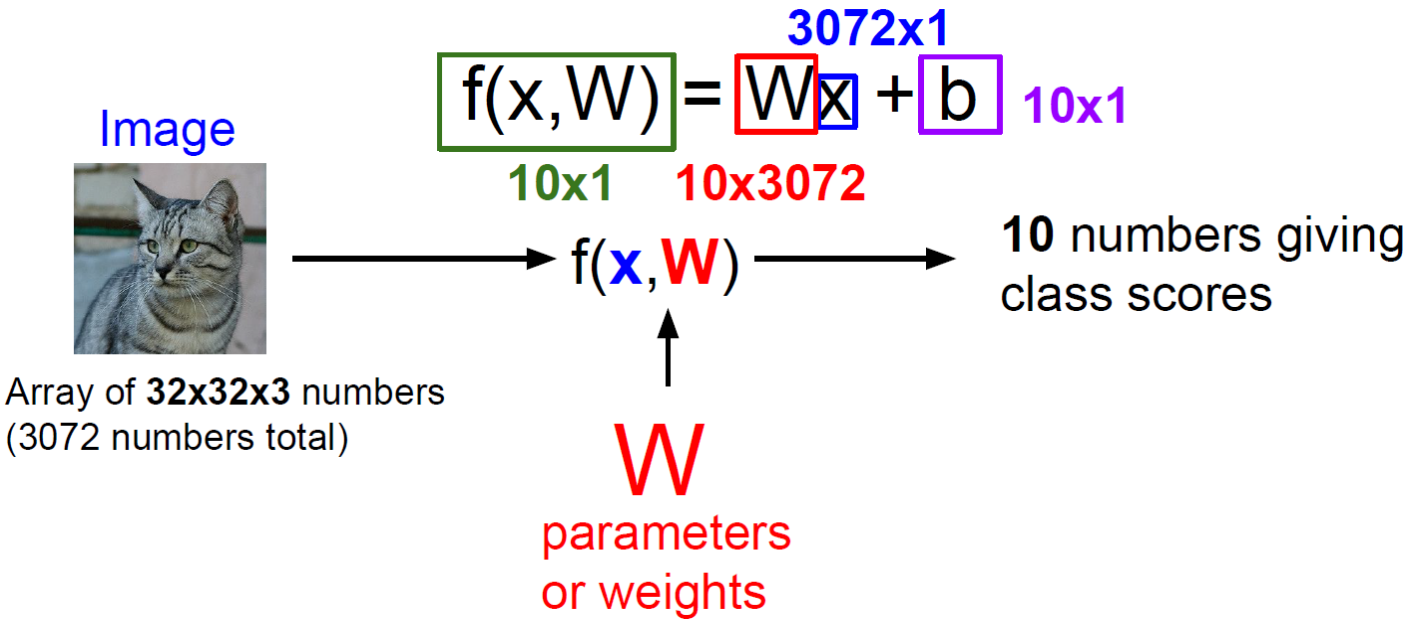
Parameterized mapping from images to label scores

- 이 접근의 첫 번째 요소는 이미지의 픽셀 값을 각 클래스의 신뢰도 점수에 매핑하는 score function을 정의하는 것.
- training dataset of images : $x_i \in R^D, each\ associated\ with\ a\ label\ y_i$
  $(i = 1 \ldots N\ \ and\ y_i \in 1 \ldots K)$
- we have $\boldsymbol{N}$ examples (each with a dimensionality $\boldsymbol{D}$) and $\boldsymbol{K}$ distinct categories.
- For example, in CIFAR-10
  - training set of $\boldsymbol{N}$ = 50,000 images, each with $\boldsymbol{D}$ = 32 x 32 x 3 = 3072 pixels, and $\boldsymbol{K}$ = 10, since there are 10 distinct classes (dog, cat, car, etc).
  - We will now define the score function $f: R^D \mapsto R^K$ that maps the raw image pixels to class scores.

http://cs231n.github.io/linear-classify/

Linear classifier

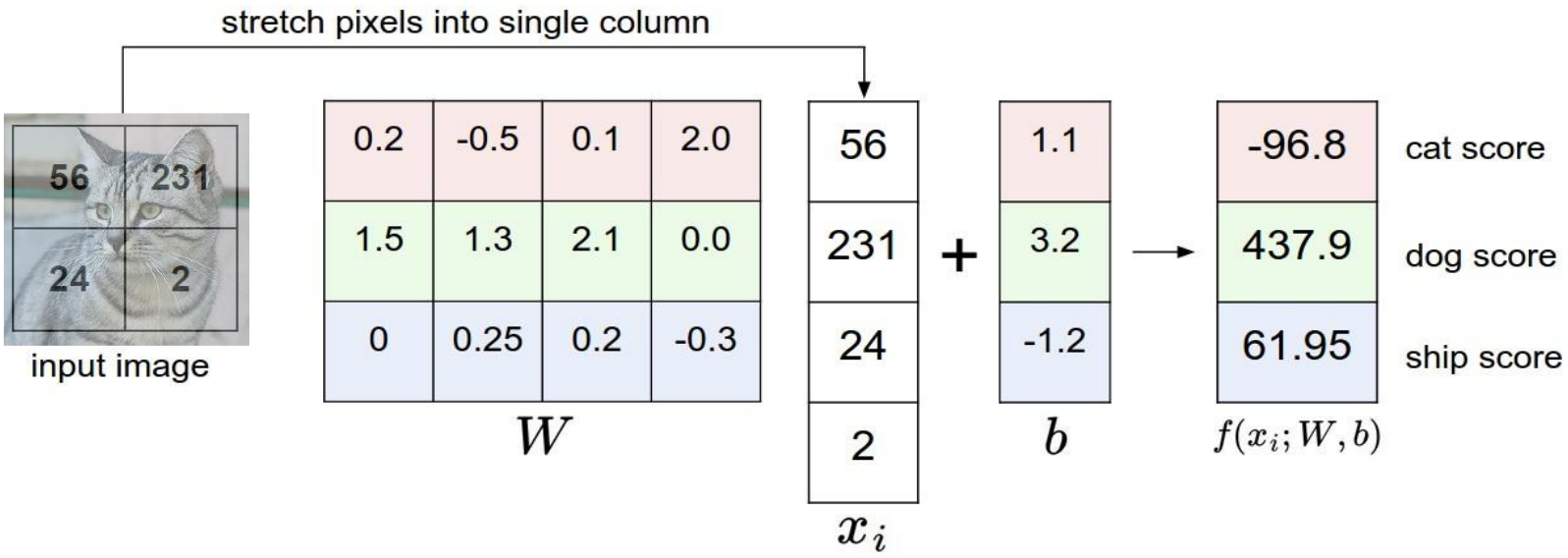
- $f(x_i, W, b) = Wx_i + b$
  - x_i : image. has all of its pixels flattened out to a single column vector of shape [D x 1].
  - W : matrix, of size [K × D], weights
  - b : vector, of size [K × 1], bias vector
- In CIFAR-10
  - x_i : $all\ pixels\ in\ the\ i-th\ image\ flattened\ into\ a\ single\ [3072 \times 1]\ column$
  - W : $[10 \times 3072]$
  - b : $[10 \times 1\ ]$



Linear Classification

- 간결화를 위하여 4 pixel
- 3 class : red(cat), green(dog), blue(ship). (rgb채널과 관련없음)
- 이미지의 픽셀을 열로 늘리고, 행렬 곱셈을 수행하여 각 클래스의 점수를 얻는다.

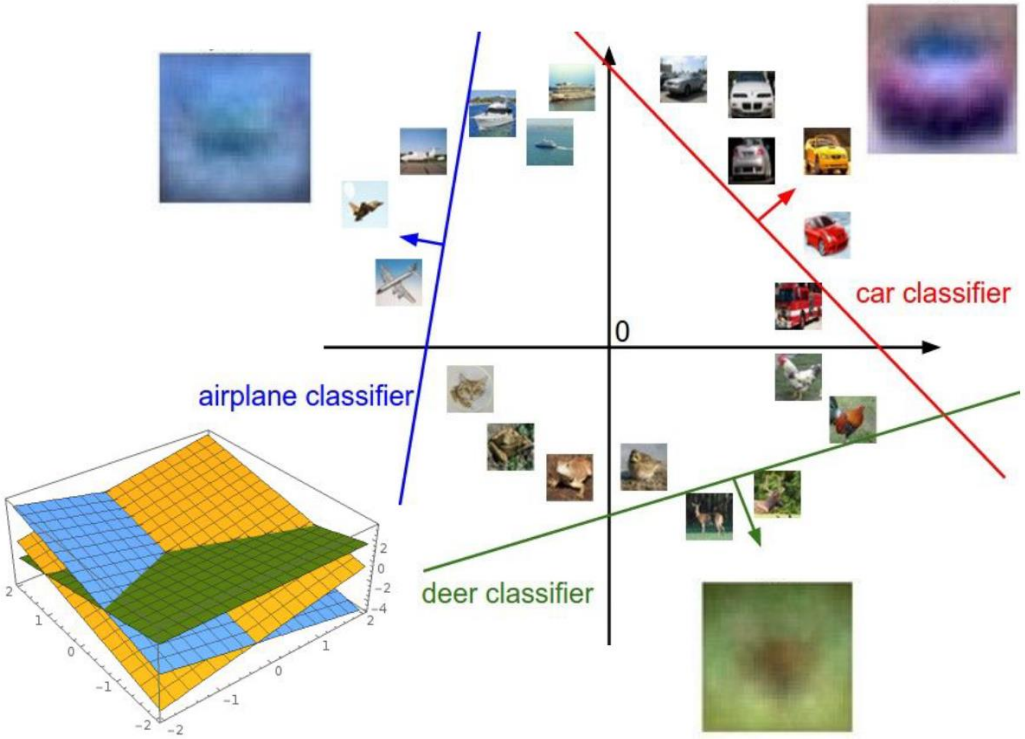
- 아래 예의 weight W는 좋지 않다. dog score가 가장 높다.



http://cs231n.github.io/linear-classify/

Analogy of images as high-dimensional points

- CIFAR-10의 각 이미지는 32 × 32 픽셀의 3072 차원 공간에 있는 점.
  - 마찬가지로 전체 데이터 세트는 (레이블이 지정된) 점 집합.
- 각 클래스의 점수를 모든 이미지 픽셀의 가중치로 정의 했으므로 각 클래스 점수는 이 공간에 대한 선형 함수(3072차원을 2차원으로 압축).



$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

- W의 모든 행은 클래스 중 하나의 분류자(classifier)
- W의 행 중 하나를 변경하면 픽셀 공간의 해당 선이 다른 방향으로 회전한다는 것.
- 편향 b는 분류자(classifier)가 선을 번역 할 수 있게 한다.

http://cs231n.github.io/linear-classify/

Template Class

- 선박 템플릿에는 예상대로 많은 파란색 픽셀이 포함.
- 따라서 이 템플릿은 내부 제품으로 바다에 있는 선박 이미지와 일치하면 높은 점수를 부여.

How can we tell whether this W is good or bad?

- Defined a (linear) score function
- How can we tell whether this W is good or bad?



airplane	-3.45	-0.51	3.42
automobile	-8.87	**6.04**	4.64
bird	0.09	5.31	2.65
cat	**2.9**	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	**-4.34**
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

http://cs231n.github.io/linear-classify/


Logistic Regression
Binary Classification
Softmax Regression

소프트맥스 회귀

- 로지스틱 함수를 여러 개의 입력인 경우 로지스틱 함수를 사용할 수 있도록 일반화 한 것
- 이진 분류기를 훈련시켜 연결하지 않고 직접 다중 클래스를 지원할 수 있도록 일반화한 것
- 다항 로지스틱 회귀$(multinomial\ logistic\ regression)$라고도 한다
- 샘플에 대해 각 클래스의 점수가 계산되면 소프트맥스 함수를 통과시켜 해당되는 클래스에 속할 **확률을 추정한다**

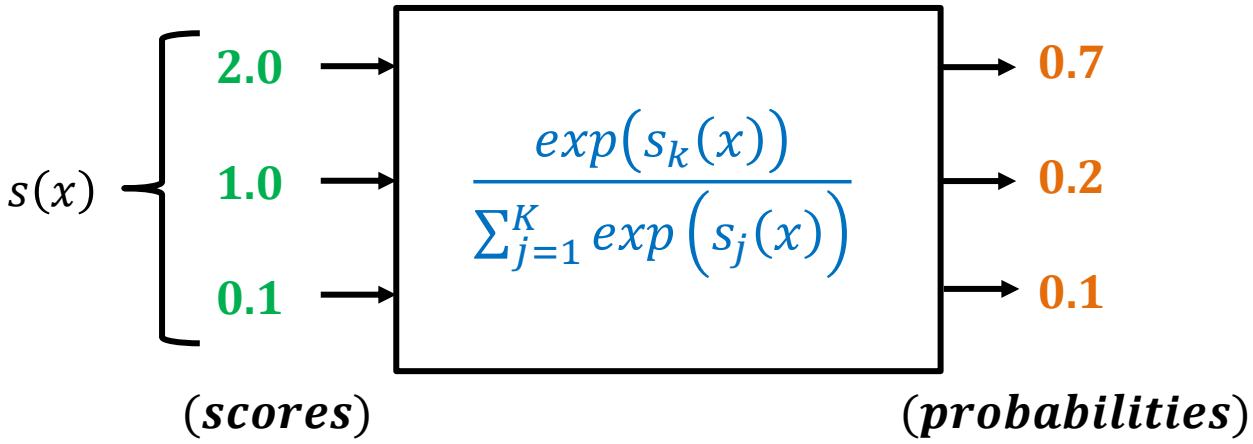
$$logistic\ function \xrightarrow{\text{일반화}(generalization)} softmax\ function$$

소프트맥스 함수(softmax function)

- 출력합계는 1. n차 실수 벡터를 0과 1 사이의 실수로 변환하여 출력

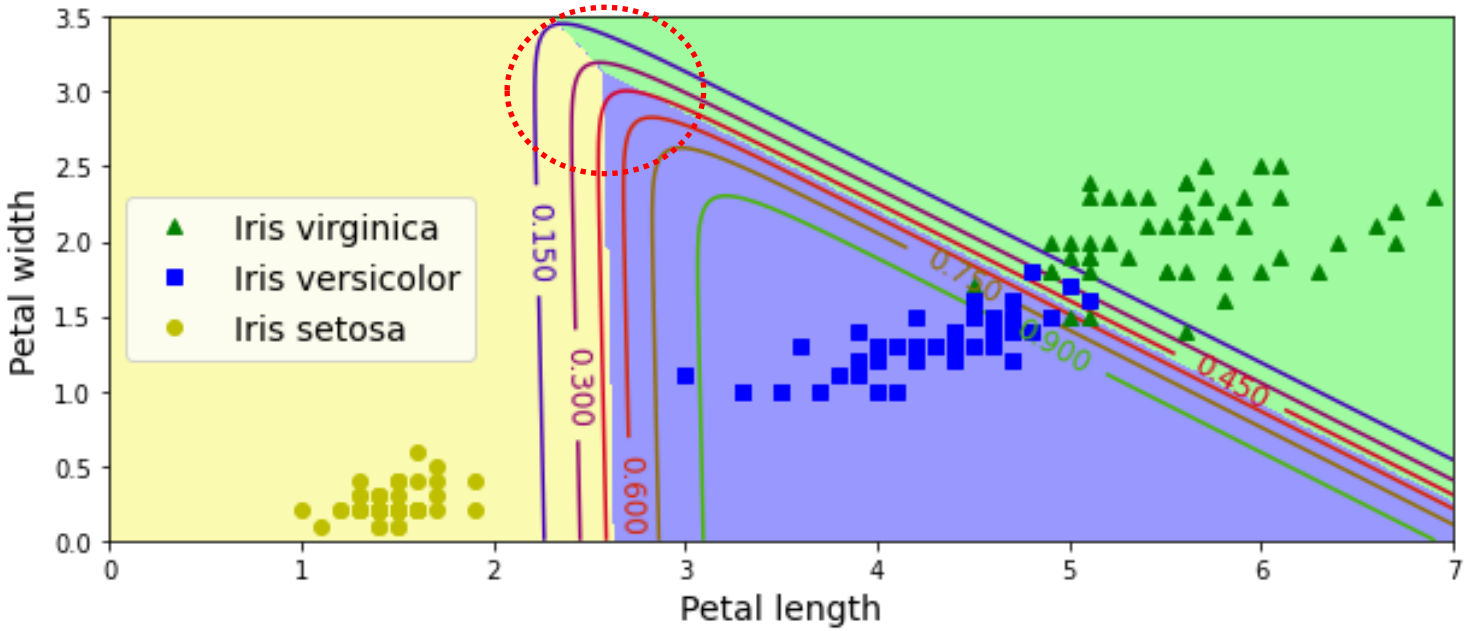
$$\sigma\left(s(x)\right)_k = \frac{exp\left(s_k(x)\right)}{\sum_{j=1}^{K} exp\left(s_j(x)\right)}$$

- K : 클래스의 수
- $s(x)$: 샘플 x에 대한 각 클래스의 점수를 담은 벡터
- $\sigma\left(s(x)\right)_k$: 샘플 x 에 대한 각 클래스의 점수가 주어졌을 때 이 샘플이 클래스 k 에 속할 추정 **확률**



```
'''
소프트맥스 함수
로지스틱 회귀모델은 여러 개의 이진 분류기를 훈련시켜 연결하지 않고,
직접 다중 범주를 지원하도록 일반화될 수 있다.
이를 소프트맥스 회귀 또는 다항 로지스틱 회귀라고 한다.
'''

X = iris["data"][:, (2, 3)]  # 꽃잎 길이와 너비 변수
y = iris["target"]           # 3개의 범주 그대로 사용

'''
- 사이킷런의 LogisticRegression은 범주가 2이상이면 일대다(OvA) 전략을 사용
- 하지만 multi_class='multinomial'옵션: 소프트맥스 회귀를 사용
- 소프트맥스 회귀를 사용하려면 solver='lbfgs'옵션을 준다
- lbfgs:소프트맥스 회귀를 지원하는 알고리즘
'''
softmax_reg = LogisticRegression(multi_class="multinomial",
                                 solver="lbfgs", C=10, random_state=42)
softmax_reg.fit(X, y)
```

```
# 훈련시킨 소프트맥스 분류기의 결정경계를 시각화. 새로운 샘플 생성
x0, x1 = np.meshgrid(
        np.linspace(0, 8, 500).reshape(-1, 1),
        np.linspace(0, 3.5, 200).reshape(-1, 1),
    )
X_new = np.c_[x0.ravel(), x1.ravel()]

y_proba = softmax_reg.predict_proba(X_new)
y_predict = softmax_reg.predict(X_new)

zz1 = y_proba[:, 1].reshape(x0.shape)
zz = y_predict.reshape(x0.shape)

plt.figure(figsize=(10, 4))
plt.plot(X[y==2, 0], X[y==2, 1], "g^", label="Iris virginica")
plt.plot(X[y==1, 0], X[y==1, 1], "bs", label="Iris versicolor")
plt.plot(X[y==0, 0], X[y==0, 1], "yo", label="Iris setosa")

from matplotlib.colors import ListedColormap
custom_cmap = ListedColormap(['#fafab0','#9898ff','#a0faa0'])

plt.contourf(x0, x1, zz, cmap=custom_cmap)
contour = plt.contour(x0, x1, zz1, cmap=plt.cm.brg)
plt.clabel(contour, inline=1, fontsize=12)
plt.xlabel("Petal length", fontsize=14)
plt.ylabel("Petal width", fontsize=14)
plt.legend(loc="center left", fontsize=14)
plt.axis([0, 7, 0, 3.5])

plt.show()
```



- **Iris versicolor클래스의 확률곡선**
- 이 모델은 이진분류와 달리 0.5의 경계가 아니라 범주에 속할 확률이 0.5 이하라도 분류한다는 점을 유의하자(범주가 2개보다 많으므로).
- 즉, 모든 결정경계가 만나는 지점은 동일하게 33.3%의 추정확률을 갖는다.

Ref : Hands-on Machine Learning with Scikit-Learn,keras&tensorflow - O'REILLY

정보이론(Information Theory)

- 섀넌($Claude\ Shannon$)의 정보이론
  - 잘 일어나지 않는 사건($unlikely\ event$)은 자주 발생하는 사건보다 정보량($informative$)이 많다
  - 확률 값이 0에 가까울수록 정보량은 무한대, 1에 가까울수록 정보량은 0이다
  - $P(x)$는 x 가 발생될 확률. $I(x)$는 x 의 정보량

$$I(x) = log_2 \frac{1}{P(x)} = -log_2 I$$

- 동전던지기
  - $I(x) = -log_2 P(x) = -\log(0.5) = 1bit$
- 8면체 주사위던지기
  - $I(x) = -log_2 P(x) = -\log\left(\frac{1}{8}\right) = 3bit$

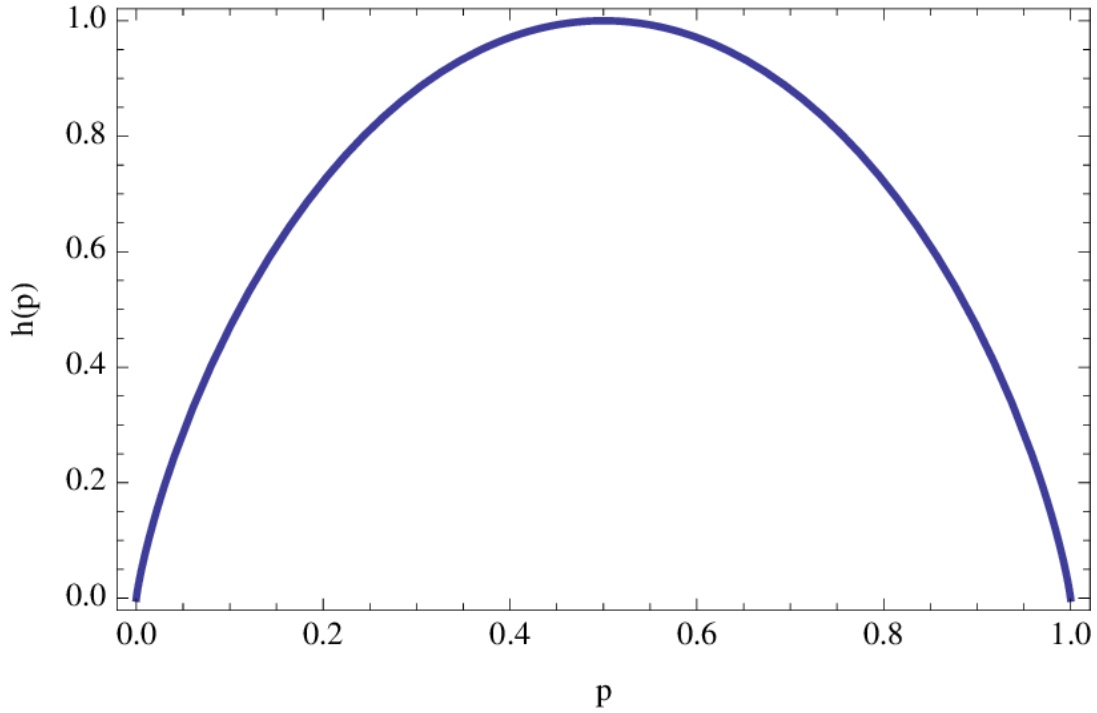
➢ 클로드 섀넌(Claude Shannon, 1916년~2001년)
정보 이론의 아버지라고 불리며,디지털 회로 이론의 창시자

Entropy

- 정보량($-log_2 P(x)$)의 기대 값(정보량의 평균)

$$H(x) = -\sum_{i=1}^{n} p(x_i)\, log_2 p(x_i) = E_p[-log_2 p(x)]$$

- 동전을 던졌을 때 앞/뒷면이 나올 확률
  - (앞/뒷면 = 50%/50%) $H(x) = -(0.5log_2 0.5 + 0.5log_2 0.5) = \mathbf{1}$
  - (앞/뒤면 = 100%/0%) $H(x) = -(1.0log_2 1.0 + 0.0log_2 0.0) = \mathbf{0}$
  - (앞/뒤면 = 90%/10%) $H(x) = -(0.9log_2 0.9 + 0.1log_2 0.1) = \mathbf{0.47}$



Cost function(Cross Entropy)

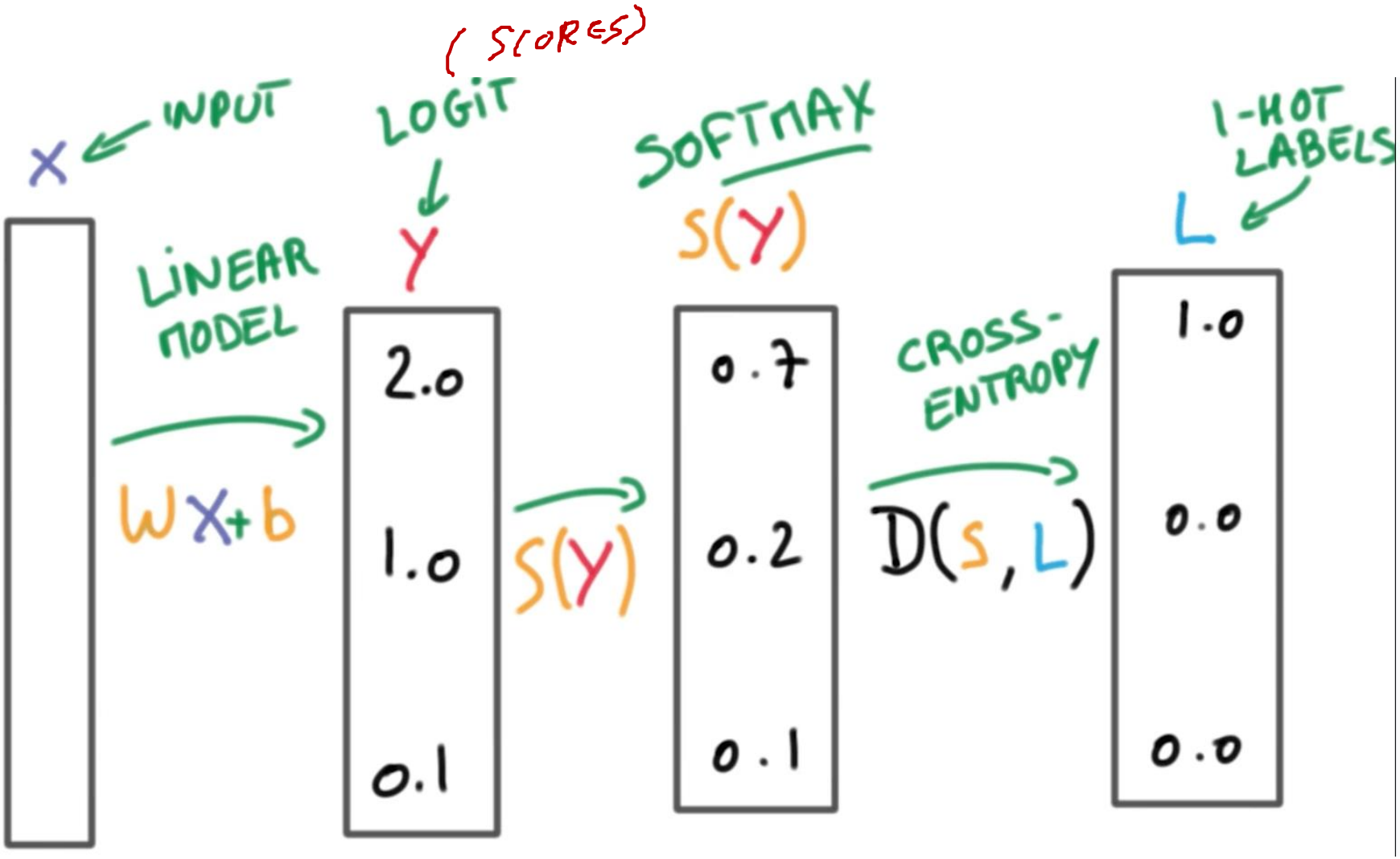
- 높은 확률을 추정하도록 만드는 것이 목적
  - 클래스가 2인경우는 로지스틱 회귀의 비용함수와 동일
- 실제확률 $p(x_i)$ 에 대하여 예측(학습)확률 $q(x_i)$를 통하여 예측하는 것

$$H(p,q) = \sum_{i=1}^{n} p(x_i)\, log_2 q\left(\frac{1}{x_i}\right) = -\sum_{i=1}^{n} p(x_i)\, log_2 q(x_i)$$

- $Cross\ Entropy \geq Entropy$
- $q(x_i)$가 학습하여 $p(x_i)$에 가까울 수록 $Cross\ Entropy$는 작아진다. 즉, 최적은 $Cross\ Entropy$ 가 최소인 값을 찾는다 – 경사하강법

[예제] 가방에 0.8, 0.1, 0.1의 비율로 빨간,녹색,파란 공이 들어가 있다.
0.2, 0.2, 0.6의 비율로 들어가 있다고 예측하자.
이 때 $entropy$ 와 $cross\ entropy$ 는 아래와 같이 계산된다.

$$H(p) = -[(0.8\log(0.8) + 0.1\log(0.1) + 0.1\log(0.1)] = 0.63$$
$$H(p,q) = -[(0.8\log(0.2) + 0.1\log(0.2) + 0.1\log(0.6)] = 1.50$$



https://www.udacity.com