6. 로지스틱 회귀

목적 : 샘플이 특정 클래스에 속할 확률 추정

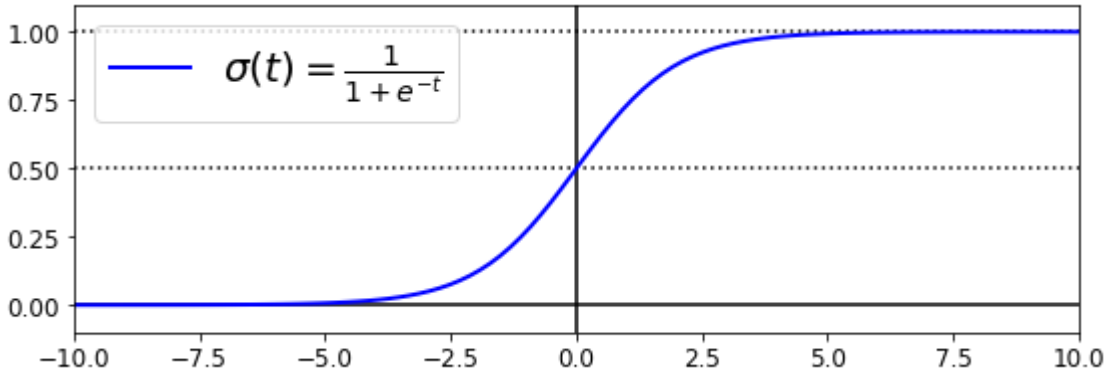
$\hat{p} = \sigma(\theta^T X) = \frac{1}{1+e^{-\theta X}}$: $(-\infty, \infty) \rightarrow (0, 1)$

$\hat{p} > 50\%$: 양성 (label = 1) → $\hat{y} = 1$

$\hat{p} < 50\%$: 음성 (label = 0) → $\hat{y} = 0$

$y^{(i)} \sim Ber(p^i)\ (0 < p^i < 1)$

$P(Y = y) = p^y(1-p)^{1-y}\ (y = 0, 1)$



6. 로지스틱 회귀

비용 함수

$$P(Y=y) = p^y(1-p)^{1-y}\ (y=0,1)$$
$$\rightarrow \log\big(P(Y=y)\big) = y log(\hat{p}) + (1-y) log(1-\hat{p})\ (y=0,1)$$

<하나의 훈련 샘플에 대한 비용 함수>

평균

$$\therefore c(\theta) = \begin{cases} -\log(\hat{p})\ \ (y=1) \\ -\log(1-\hat{p})\ (y=0) \end{cases}$$

<로지스틱 회귀의 비용 함수>

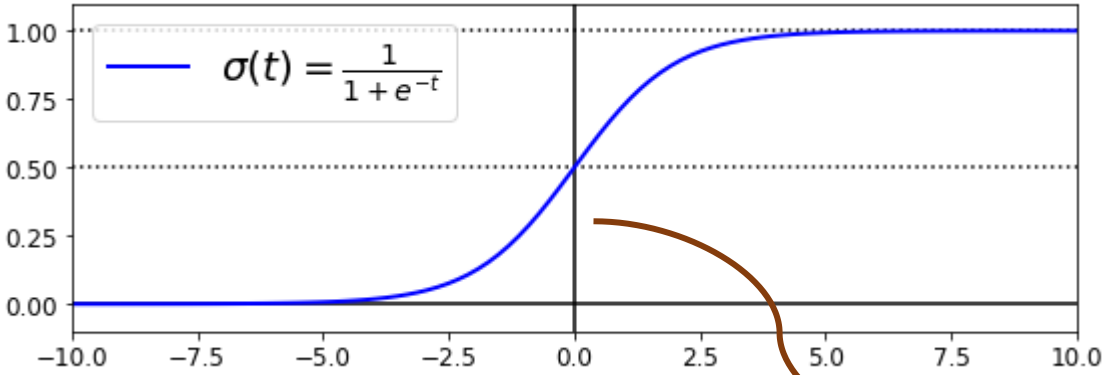
$$J(\theta) = -\frac{1}{m}\sum [y^i \log\big(\hat{p}^i\big) + (1-y^i)\log(1-\hat{p}^i)]$$

편미분

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum \big(\sigma\big(\theta^T \mathrm{X}^i\big) - y^i\big)\mathrm{X}_j^i$$

6. 로지스틱 회귀

결정 경계



t = 0일 때를 기준으로 클래스 구분

∴ $\sigma(\theta^T X)$에서 $\theta^T X$=0이 되게 하는 x의 집합 → **결정 경계**

6. 로지스틱 회귀

결정 경계 – 특성이 1개일 때,

```
from sklearn import datasets
iris=datasets.load_iris()

X=iris['data'][:, 3:]
y=(iris['target']==2).astype(np.int)
```

X=iris['data'][:, 3:] → 꽃잎의 너비 (특성)

y=(iris['target']==2).astype(np.int) → Iris virginica (0|진 분류 기준)

```
from sklearn.linear_model import LogisticRegression
log_reg=LogisticRegression(solver='lbfgs', random_state=42)
log_reg.fit(X,y)
```

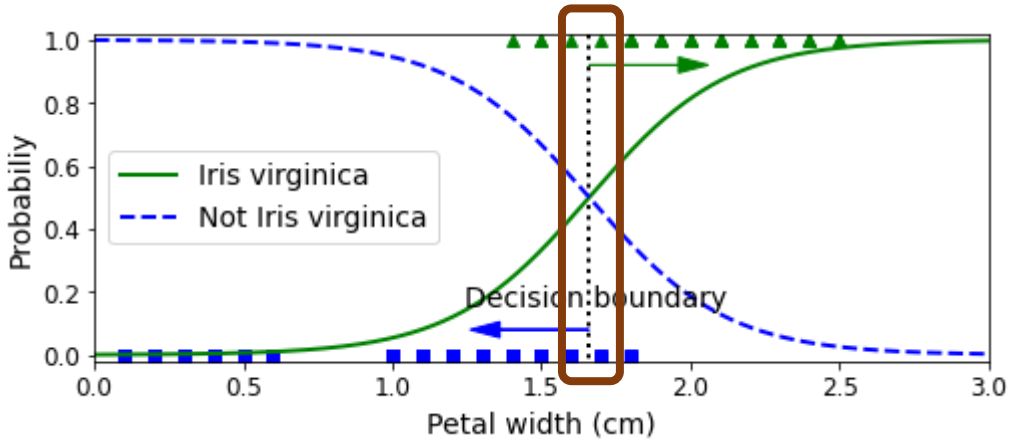
log_reg.fit(X,y) → 로지스틱 회귀 모델 훈련

6. 로지스틱 회귀

```
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]
```

→ 결정 경계 ($\hat{p} \geq 0.5$인 가장 작은 X 값)

양성 클래스에 대한 확률



특성이 하나 → 결정 경계를 만드는 X 값 : 1개
∴ 결정 경계가 **축에 평행**하게 나타남

6. 로지스틱 회귀

결정 경계 – 특성이 2개일 때,

```
from sklearn.linear_model import LogisticRegression

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

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)
```

특성 2개

Iris virginica (이진 분류 기준)

로지스틱 회귀 모델 훈련

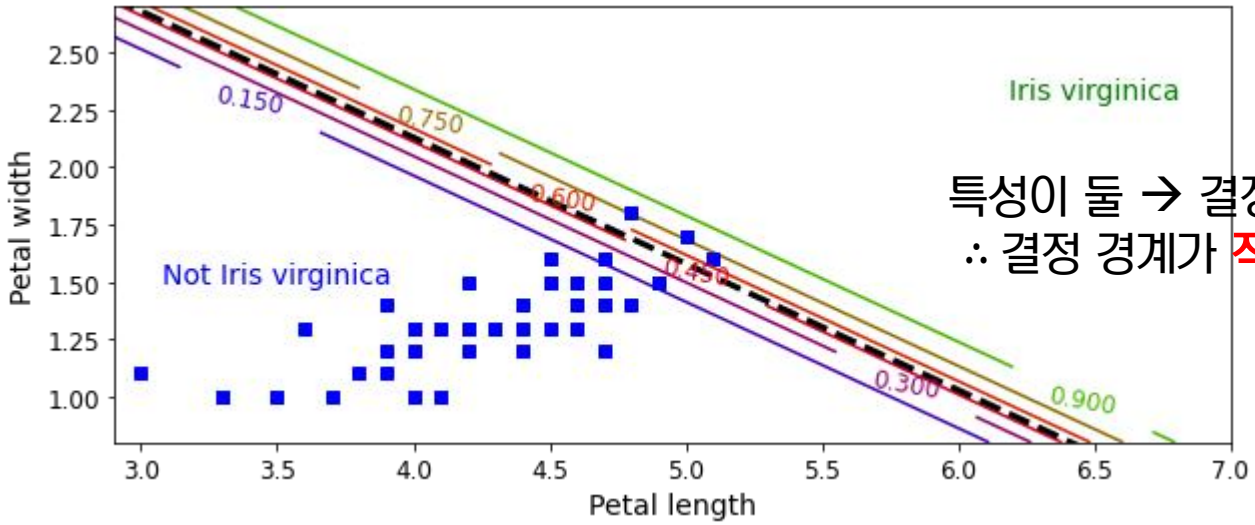
축에 대한 point를 입력 받고
point들이 교차하는 좌표를 모두 계산

6. 로지스틱 회귀

```
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]
```

→ 결정 경계 ($\hat{p} \geq 0.5$인 가장 작은 X 값)



특성이 둘 → 결정 경계를 만드는 X 값 : 2개
∴ 결정 경계가 **직선 방정식**으로 나타남

6. 로지스틱 회귀

소프트맥스 함수

$$\hat{p}_k = \sigma\big(s(\mathrm{X})\big)_k = \frac{\exp(s_{k(\mathrm{X})})}{\sum\exp(s_{j(\mathrm{X})})}, \quad where\ s_{k(x)} = \left(\theta^k\right)^T \boldsymbol{X}$$

$\hat{p}_k$: 샘플이 클래스 k에 속할 추정 확률

K : 클래스 수

$s_{k(x)}$: 샘플 x에 대한 각 클래스의 점수를 담은 벡터

$$\therefore \sum \hat{p}_k = 1 \quad \& \quad \hat{y} = argmax_k \boldsymbol{\sigma}\big(\boldsymbol{s}(\boldsymbol{X})\big)_{\boldsymbol{k}} = argmax_k \boldsymbol{s_k}(\boldsymbol{X}) = argmax_k \left(\boldsymbol{\theta^k}\right)^{\boldsymbol{T}} \boldsymbol{X}$$

주의! 소프트맥스 회귀 분류기는 **한 번에 하나의 클래스만 예측**

∴ **상호 배타적인 클래스**에서만 사용 (여러 사람의 얼굴을 인식하는 용도 X)

6. 로지스틱 회귀

소프트맥스 함수

i번째 샘플이 클래스 k에 속할 확률

<크로스 엔트로피 비용 함수>

$$J(\Theta) = -\frac{1}{m}\sum\sum y_k^i \log(\hat{p}_k^i)$$

클래스 2개일 때, → 로지스틱 회귀의 비용함수와 동일

<클래스 k에 대한 크로스 엔트로피의 그레이디언트 벡터> *(비용 함수 최소화 목적)*

$$\nabla_{\Theta_k} J(\Theta) = \frac{1}{m}\sum (\hat{p}_k^i - y_k^i)\mathrm{X}^i$$

6. 로지스틱 회귀

소프트맥스 함수

```
X = iris['data'][:, (2, 3)]  # 꽃잎 길이, 꽃잎 너비
y = iris['target']

softmax_reg = LogisticRegression(multi_class='multinomial',solver="lbfgs", C=10, random_state=42)
softmax_reg.fit(X, y)
```

소프트맥스 회귀

l_2 규제

```
print(softmax_reg.predict([[5, 2]]))
# [2]
print(softmax_reg.predict_proba([[5, 2]]))
# [[6.38014896e-07, 5.74929995e-02, 9.42506362e-01]]
```

최댓값 (index = 2)

6. 로지스틱 회귀

소프트맥스 함수의 결정 경계