Training Models [3] – Logistic Regression & Softmax Regression

Samkeun Kim <skim@hknu.ac.kr>

http://cyber.hankyong.ac.kr

Logistic Regression

Regression 알고리즘을 Classification에 사용할 수 있다 => **Logistic Regression**

- 인스턴스가 특정 클래스에 속하는 확률을 추정하는데 사용됨(What is the probability that this email is spam?)
- 추정된 확률이 **50%**보다 크면 모델은 해당 인스턴스가 positive 클래스("1")에 속한다고 예측
- 추정된 확률이 **50%**보다 작으면 모델은 해당 인스턴스가 negative 클래스("0")에 속한다고 예측
- 이 경우 binary classifier가 된다.

Estimating Probabilities

Linear Regression과 마찬가지로 입력 특성들의 가중치 합 계산

- 하지만 Linear Regression 모델처럼 결과를 직접 출력하지 않고, 결과의 로지스틱을 출력한다.
- Logistic Regression model estimated probability (vectorized form)

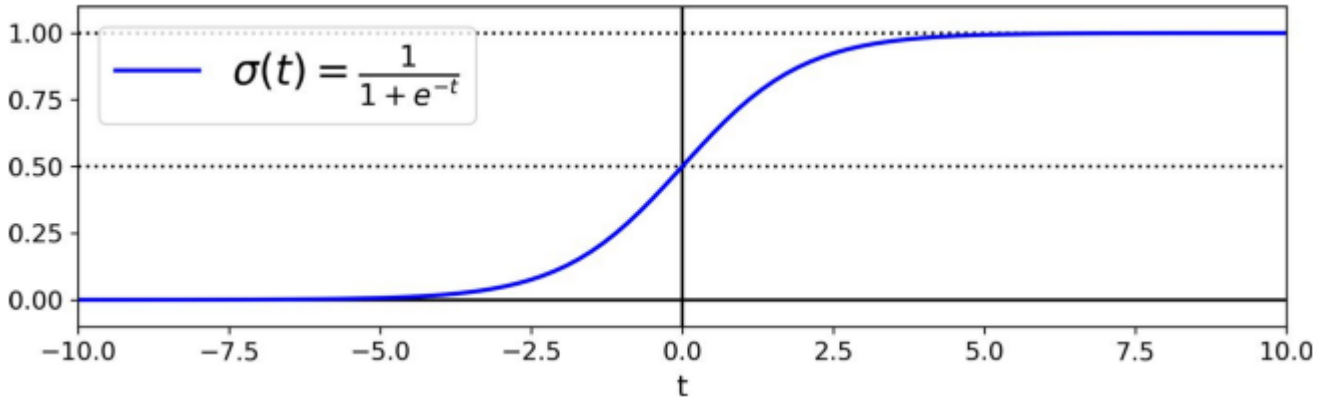
$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x})$$

- **로지스틱**=> 0 ~ 1 사이의 숫자를 출력하는 Sigmoid 함수

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x})$$

Logistic 함수:

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



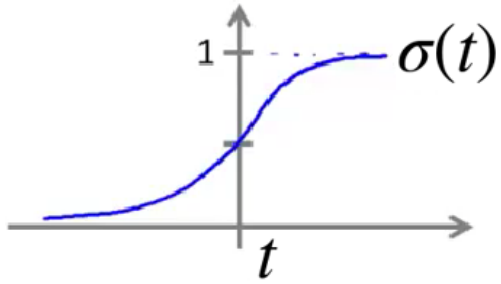
Estimating Probabilities

Logistic Regression 모델 => 인스턴스 $\mathbf{x}$가 positive 클래스에 속하는 확률 $\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x})$를 추정할 수 있다면

예측 값 $\hat{y}$을 쉽게 계산할 수 있다:

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma\left(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}\right) = P(y = 1|x;\theta)$$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



$\sigma(t) \geq 0.5 \quad whenever\ t \geq 0$

$h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma\left(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}\right) \geq 0.5$

whenever $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x} \geq \mathbf{0}$

Suppose predict "y=1" if $h_{\boldsymbol{\theta}}(\mathbf{x}) \geq 0.5$

$\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x} \geq \mathbf{0}$

predict "y=0" if $h_{\boldsymbol{\theta}}(\mathbf{x}) < 0.5$

$h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma\left(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}\right)$

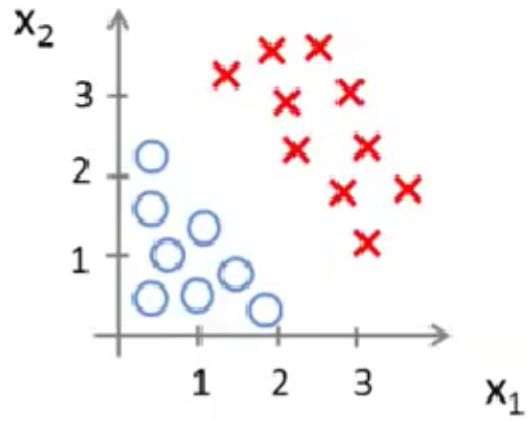
whenever $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x} < 0$

$\sigma(t) < 0.5$

Logistic Regression 모델:

$\Rightarrow \boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}$가 positive이면 1을, negative이면 0을 출력

Decision Boundary

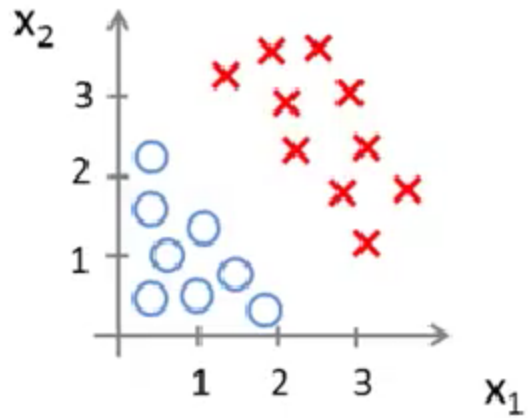


$x_0 = 1$

σ

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Decision Boundary

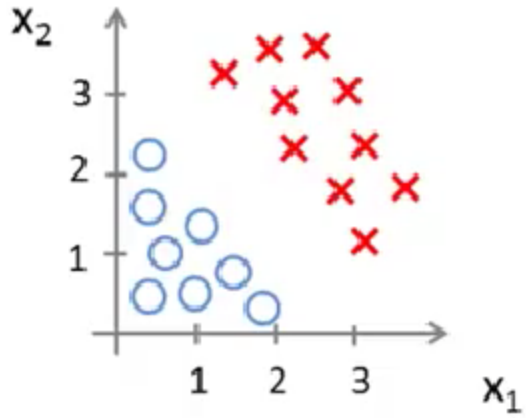


$$\rightarrow h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

$-3 \quad 1 \quad 1$

Decision Boundary



$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

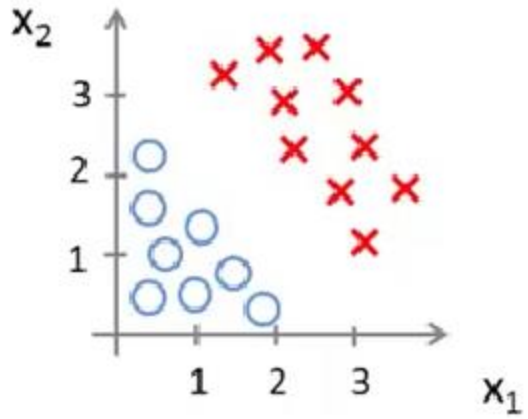
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$-3 \quad 1 \quad 1$

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

$\theta^T x$

Decision Boundary



$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

$-3 \quad 1 \quad 1$

x_1, x_2

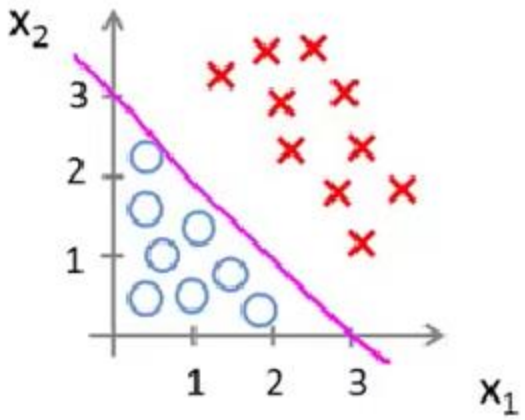
Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

$\theta^T x$

$x_1 + x_2 \geq 3$

Decision Boundary

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$



$$\rightarrow h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$-3 \quad 1 \quad 1$

x_1, x_2

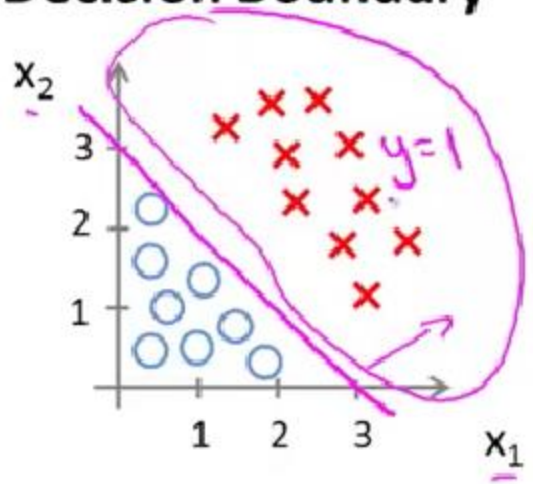
Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

$\theta^T x$

$x_1 + x_2 = 3$

$\rightarrow x_1 + x_2 \geq 3$

Decision Boundary



$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

$-3 \quad 1 \quad 1$

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

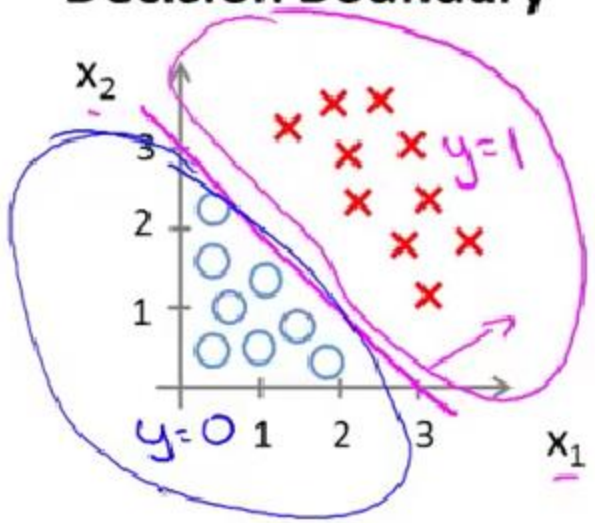
$\theta^T x$

x_1, x_2

$x_1 + x_2 = 3$

$x_1 + x_2 \geq 3$

Decision Boundary



$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$-3 \quad 1 \quad 1$

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

$\theta^T x$

x_1, x_2

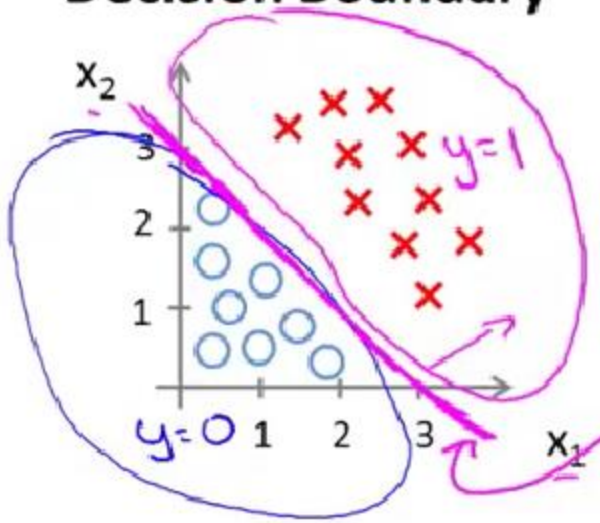
$x_1 + x_2 = 3$

$x_1 + x_2 \geq 3$

$x_1 + x_2 < 3$

$y = 0$

Decision Boundary



$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

Decision boundary

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

$\theta^T x$

x_1, x_2

$h_\theta(x) = 0.5$

$x_1 + x_2 = 3$

$x_1 + x_2 \geq 3$

$x_1 + x_2 < 3$

$y = 0$

Training and Cost Function

Logistic Regression 모델의 학습 목표:

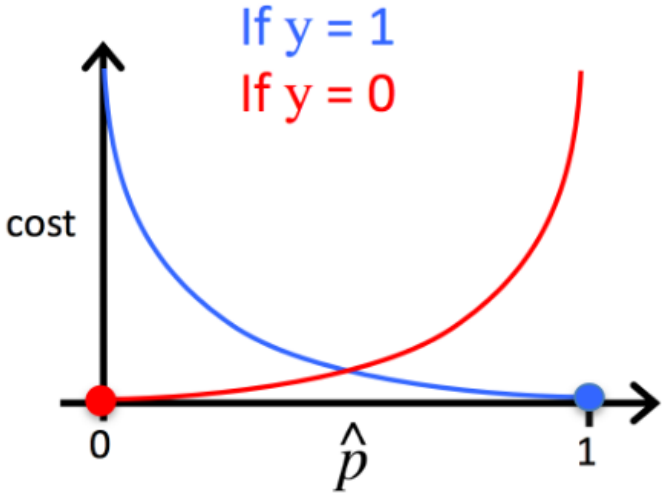
모델이 positive 인스턴스($y = 1$)에 대해서는 높은 확률을 추정하고,

negative 인스턴스($y = 0$)에 대해서는 낮은 확률을 추정하도록

파라메터 벡터 $\boldsymbol{\theta}$를 설정하는 것

Cost function of a single training instance:

$$c(\boldsymbol{\theta}) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1 \\ -\log(1 - \hat{p}) & \text{if } y = 0 \end{cases}$$



전체 training set에 대한 비용 함수 => 전체 training 인스턴스들에 대한 평균 비용

- Log loss라는 1개의 식으로 표현 가능 (log loss)

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

- j번째 모델 파라메터 θ_j에 대한 비용 함수의 편도함수:

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x})$$

$$\frac{\partial}{\partial\theta_j}\mathrm{J}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}(\sigma(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}^{(i)}) - y^{(i)})\,x_j^{(i)}$$

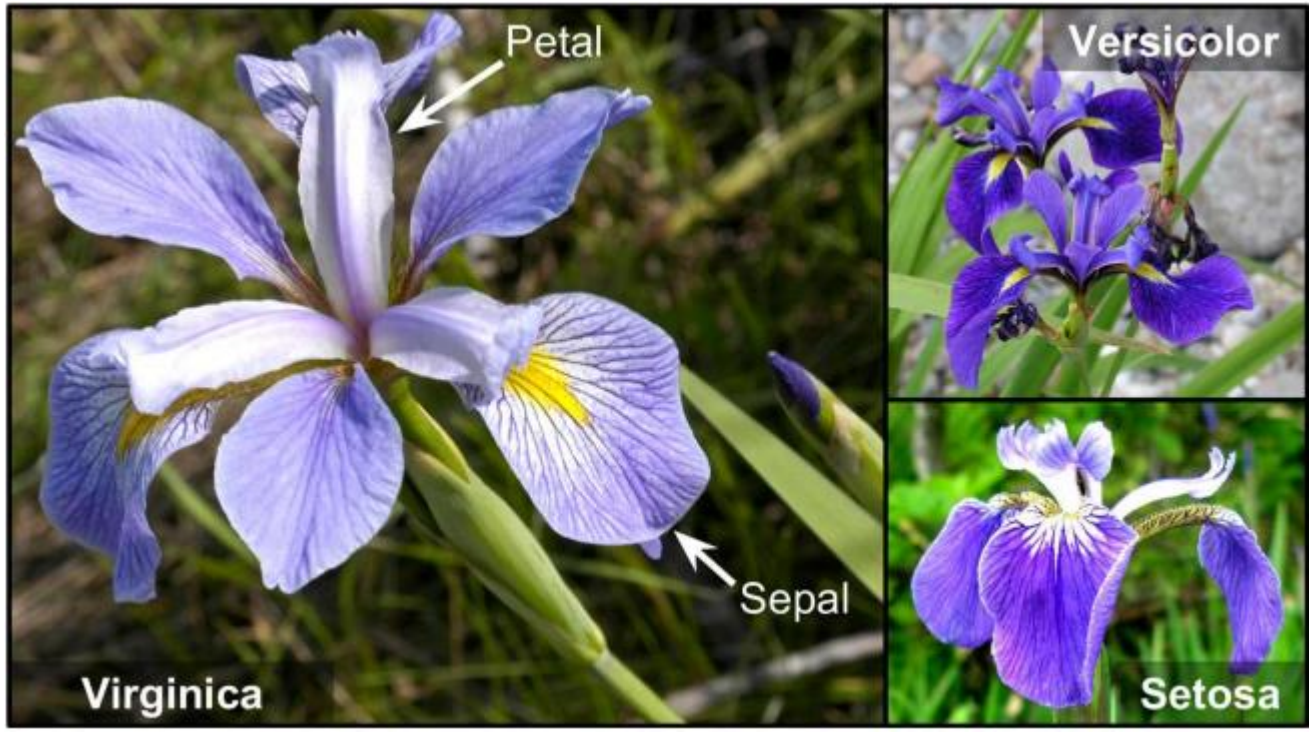
각 인스턴스에 대해 예측 에러를 계산하고, 에러 값을 j번째 특성 값에 곱하고, 전체 training 인스턴스에 대해 평균 값을 계산한다.

⇒ 모든 편도함수를 포함하는 기울기 벡터를 가진다면 Batch Gradient Descent 알고리즘에서 사용 가능

Decision Boundaries

Logistic Regression의 구체적인 예를 보여주기 위해 Iris 데이터셋을 이용한다.

- Iris 꽃의 3가지 종(Iris setosa, Iris versicolor, Iris virginica)으로 구성된 150개의 서로 다른 꽃받침과 꽃잎의 길이와 폭을 가진 데이터셋



Decision Boundaries

꽃잎 폭(petal width) 특성에만 기반한 Iris virginica 타입을 찾기 위한 Classifier를 만든다:

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> list(iris.keys())
['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename']
>>> X = iris["data"][:, 3:]  # petal width
>>> y = (iris["target"] == 2).astype(np.int)  # 1 if Iris virginica, else 0
```

Logistic Regression 모델 학습:

```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()
log_reg.fit(X, y)
```

Iris setosa - 0
Iris versicolor - 1
Iris virginica - 2

sepal.length	sepal.width	petal.length	petal.width
5.1	3.5	1.4	.2

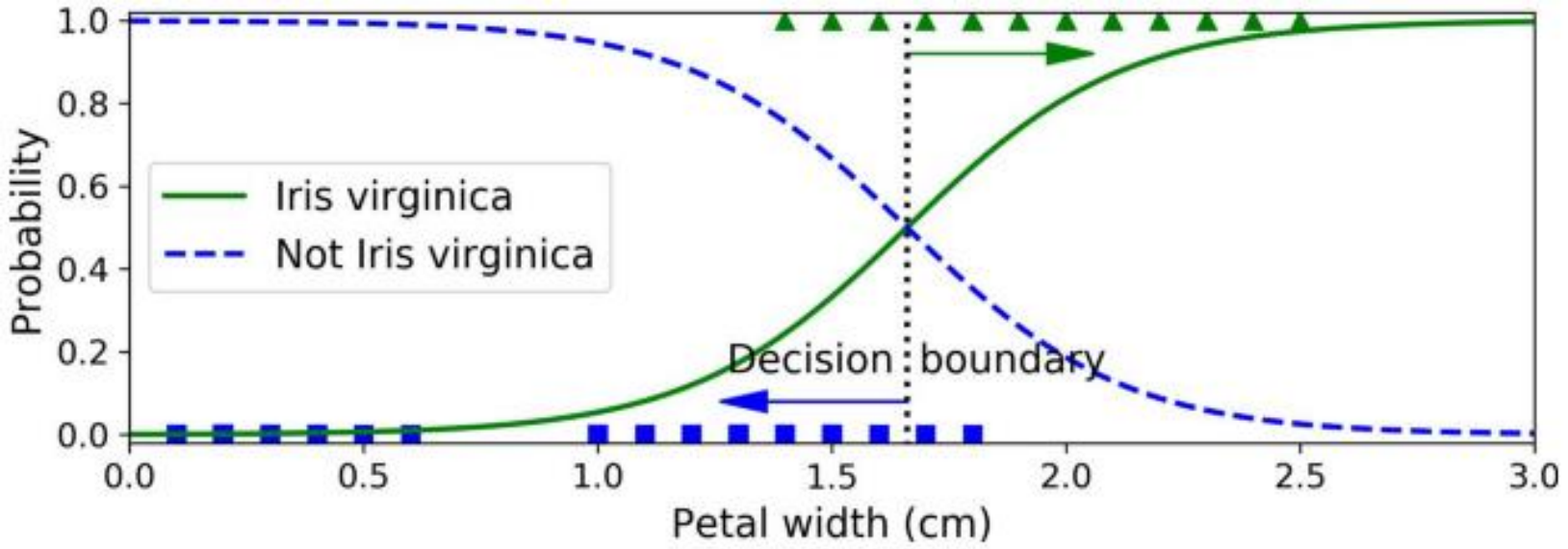
Decision Boundaries

꽃잎 폭을 1 ~ 3cm 구간에서 값을 바꾸면서 모델의 추정된 확률을 살펴보자:

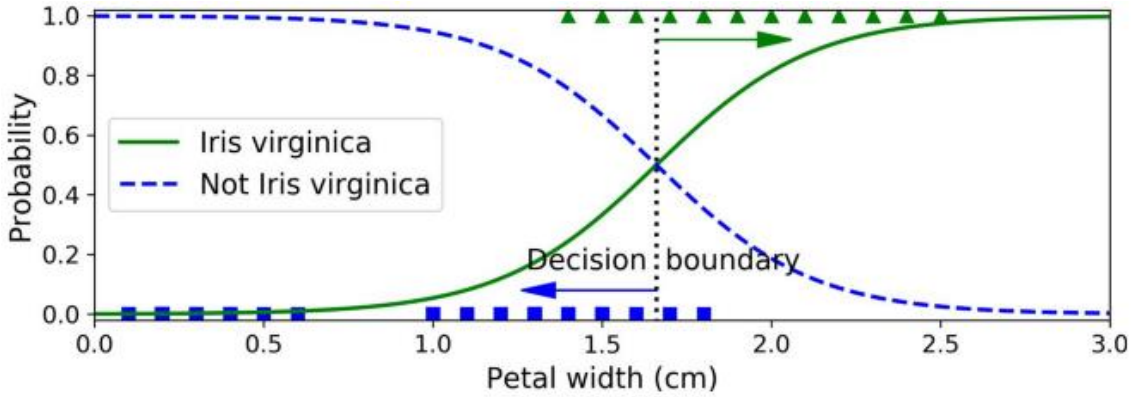
```
X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)
plt.plot(X_new, y_proba[:, 1], "g-", label="Iris virginica")
plt.plot(X_new, y_proba[:, 0], "b--", label="Not Iris virginica")
# + more Matplotlib code to make the image look pretty
```

← petal width의 전체 열벡터로 재구성

Estimated probabilities and decision boundary:



Decision Boundaries



- Iris virginica의 꽃잎 폭은 1.4 ~ 2.5cm, 다른 종류의 꽃은 0.1 ~ 1.8cm, 약간 겹치는 부분이 있음
- 꽃잎 폭이 2cm 이상인 경우 확실하게 "Iris virginica" 꽃이라고 판단할 수 있음
- 꽃잎 폭이 1cm 이하인 경우 확실하게 "Not Iris virginica" 꽃이라고 판단할 수 있음
- 두 극한 경우 사이(1 ~ 2cm)에는 불확실함

Classifier에게 반드시 클래스를 예측하라고 한다면 어떤 클래스이던 선택하여 반환할 것이다.

⇒ 양쪽 확률이 50% 정도 되는 1.6cm 근처에 decision boundary가 있다:

⇒ 꽃잎 폭이 1.6cm보다 크면 Iris virginica 꽃이라고 예측

⇒ 그렇지 않으면 Iris virginica 꽃이 아니라고 예측 (확신은 없다 할지라도)

```
>>> log_reg.predict([[1.7], [1.5]])
array([1, 0])
```

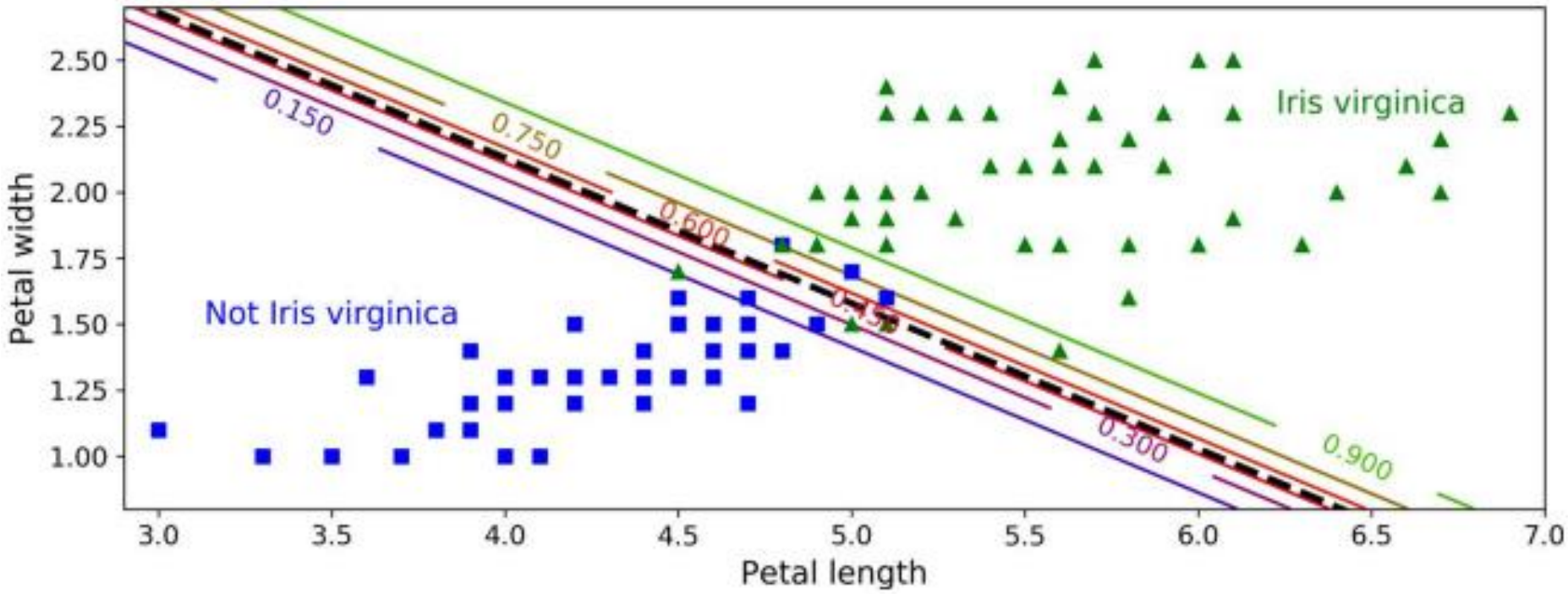
Decision Boundaries

동일한 데이터셋이지만 이번에는 꽃잎 폭과 꽃잎 길이 2개의 특성을 보여주는 그래프를 살펴보자:

Logistic Regression classifier => 2개의 특성에 기반하여 새로운 꽃이 Iris virginica일 확률을 추정할 수 있다.

Dashed line => 모델이 50% 확률로 추정하는 점들을 의미: 모델의 linear decision boundary라고 함

Top-right line 이상의 모든 꽃 => Iris virginica일 확률이 90% 이상임



Softmax Regression (Multinomial Logistic Regression)

Softmax Regression

- 다중 클래스를 직접 분류할 수 있도록 Logistic Regression 모델을 일반화시킨 Classifier
- 인스턴스 $\mathbf{x}$가 주어지면, 각 클래스 k에 대한 score $s_k(\mathbf{x})$를 먼저 계산
- 클래스 k에 대한 *Softmax score* :

$$s_k(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\theta}^{(k)}$$

- 이렇게 얻은 score에 *Softmax* 함수를 적용하여 각 클래스의 확률을 추정한다.

인스턴스 $\mathbf{x}$에 대한 모든 클래스의 score를 계산했다면

- Score를 Softmax 함수에 통과시켜 해당 인스턴스가 클래스 k에 속하는 확률 $\hat{p}_k$을 추정 가능
- Softmax 함수 => 모든 score에 대한 지수 값을 계산하여 정규화시킨다:

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

- K is the number of classes.
- $\mathbf{s}(\mathbf{x})$ is a vector containing the scores of each class for the instance $\mathbf{x}$.
- $\sigma(\mathbf{s}(\mathbf{x}))_k$ is the estimated probability that the instance $\mathbf{x}$ belongs to class k, given the scores of each class for that instance.

Softmax Regression classifier:

- Logistic Regression classifier와 마찬가지로 가장 높은 추정 확률을 가진 클래스를 예측한다.
- Softmax Regression classifier prediction:

$$\hat{y} = \underset{k}{\operatorname{argmax}}\, \sigma(\mathbf{s}(\mathbf{x}))_k = \underset{k}{\operatorname{argmax}}\, s_k(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\, \left((\boldsymbol{\theta}^{(k)})^{\intercal}\mathbf{x}\right)$$

argmax operator => 함수 값을 최대로 하는 변수 값을 반환한다.

추정된 확률 $\sigma(\mathbf{s}(\mathbf{x}))_k$을 최대로 하는 k 값을 반환한다.

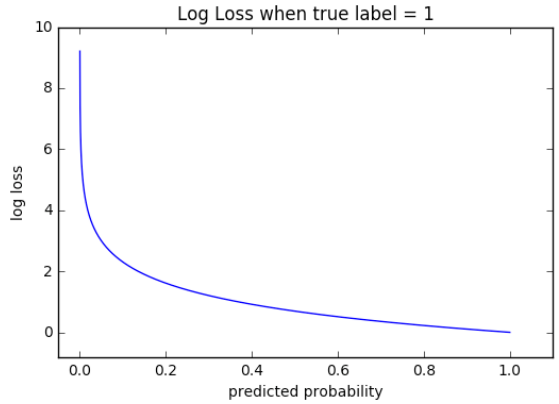
- Softmax Regression classifier => 한번에 오로지 1개의 클래스만 예측한다.

Softmax Regression 모델을 학습시켜 보자:

- 목표: 타깃 클래스에 대해 높은 확률을, 다른 클래스에 대해서는 낮은 확률을 추정하는 모델을 찾는 것
- 아래 비용 함수(cross entropy)를 최소화

  => 타깃 클래스에 대해 낮은 확률 값을 추정하면 모델에 벌점을 부과하므로 목적에 부합
- *Cross entropy cost function*

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$$

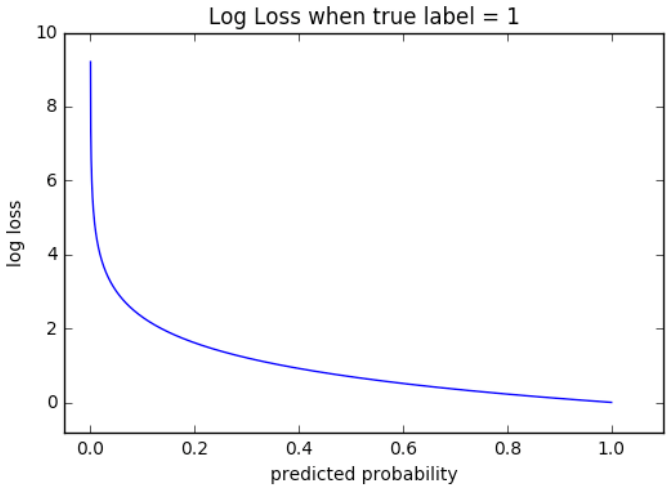


- $y_k^{(i)}$: i번째 인스턴스가 클래스 k에 속하는 타깃 확률

  => 일반적으로 인스턴스가 클래스에 속하는가 아닌 가에 따라 1 또는 0 값이다.

Cross entropy: 추정된 클래스 확률이 타깃 클래스를 얼마나 잘 매칭시키는가를 측정할 때 자주 사용된다.

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



i번째 인스턴스가 클래스 k에 속하는 타깃 확률:

$$-\frac{1}{m}\sum_{k=1}^{K} y_k \log(\hat{p}_k) = \frac{1}{m}\sum_{k=1}^{K} y_k * (-\log(\hat{p}_k))$$

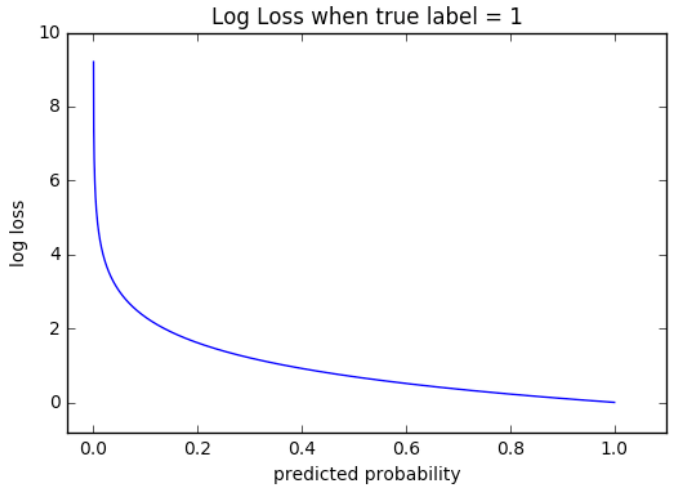
$y_k = \begin{bmatrix}0\\1\end{bmatrix} = B$ (클래스: A, B)

K=2:
Logistic Regression의 cost 함수와 같음

$\hat{p}_k = \begin{bmatrix}0\\1\end{bmatrix} = B$ (O), $\begin{bmatrix}0\\1\end{bmatrix} \odot -log\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} \odot \begin{bmatrix}\infty\\0\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix} \Rightarrow 0$

$\hat{p}_k = \begin{bmatrix}1\\0\end{bmatrix} =$ A (X), $\begin{bmatrix}0\\1\end{bmatrix} \odot -log\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} \odot \begin{bmatrix}0\\\infty\end{bmatrix} = \begin{bmatrix}0\\\infty\end{bmatrix} \Rightarrow \infty \uparrow$

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



i번째 인스턴스가 클래스 k에 속하는 타깃 확률:

$$-\frac{1}{m}\sum_{k=1}^{K} y_k \log(\hat{p}_k) = \frac{1}{m}\sum_{k=1}^{K} y_k * (-\log(\hat{p}_k))$$

$y_k = \begin{bmatrix}1\\0\end{bmatrix} = A$ (클래스: A, B)

K=2:
Logistic Regression의 cost 함수와 같음

$\hat{p}_k = \begin{bmatrix}1\\0\end{bmatrix}$ = A (O), $\begin{bmatrix}1\\0\end{bmatrix} \odot -log\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} \odot \begin{bmatrix}0\\\infty\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix} \Longrightarrow 0$

$\hat{p}_k = \begin{bmatrix}0\\1\end{bmatrix}$ = B (X), $\begin{bmatrix}1\\0\end{bmatrix} \odot -log\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} \odot \begin{bmatrix}\infty\\0\end{bmatrix} = \begin{bmatrix}\infty\\0\end{bmatrix} \Longrightarrow \infty \uparrow$

$$J(\boldsymbol{\Theta}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$$

$\boldsymbol{\theta}^{(k)}$에 대해 비용 함수의 기울기(gradient) 벡터:

$$\nabla_{\boldsymbol{\theta}^{(k)}} J(\boldsymbol{\Theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{p}_k^{(i)} - y_k^{(i)}\right)\mathbf{x}^{(i)}$$

- 모든 클래스에 대해 기울기 벡터를 계산할 수 있다.
- Gradient Descent(또는 임의의 최적화 알고리즘)를 사용하여 비용 함수를 최소화하는 파라메터 행렬 $\boldsymbol{\Theta}$를 찾는다.

Softmax Regression을 사용하여 Iris 꽃을 3개의 클래스로 분류해 보자:

- Scikit-Learn의 LogisticRegression => 3개 이상의 클래스를 학습시키면 디폴트로 OvR을 사용한다.
- 그러나 `multi_class="multinomial"`로 지정하면 Softmax Regression으로 전환된다:
- Softmax Regression을 지원하는 `solver="lbfgs"`도 지정

```
X = iris["data"][:, (2, 3)]  # petal length, petal width
y = iris["target"]

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

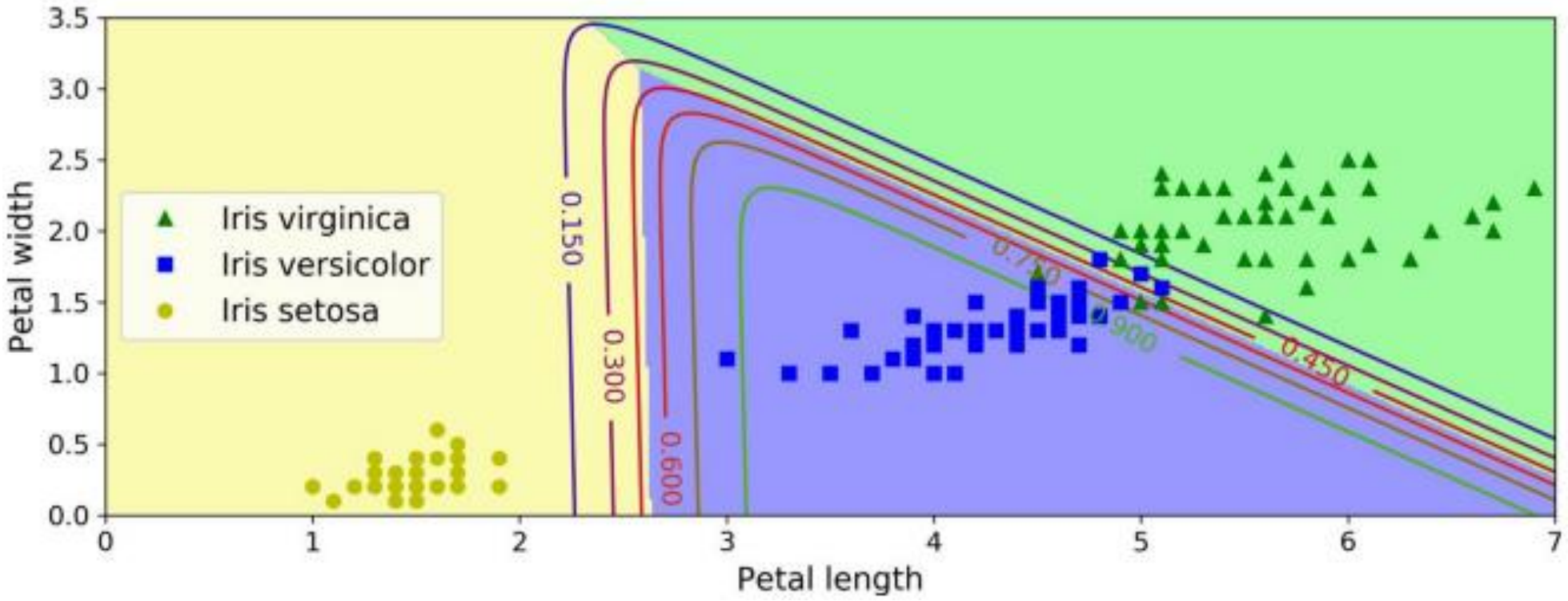
- 꽃잎의 길이 5cm, 폭 2cm인 Iris 꽃을 발견했다고 하자. 모델에게 물어보자:

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

- 모델의 답: 94.2% 확률로 Iris virginica (class 2)라고 응답!!

Softmax Regression decision boundaries: (백그라운드 컬러로 나타낸)

- 두 클래스 사이의 decision boundary는 linear
- 0.450 labeled line => Iris versicolor일 확률이 45%임을 의미



실습과제 7-1

본문에 나오는 전체 내용 PyCharm에서 실행하기

★ 반드시 결과 값 및 결과로 나온 모든 그래프에 대해 자세한 분석을 하시오.

참고: 제 07강 실습과제 #7 Training Models [3] - Logistic Regression & Softmax Regression.pdf

실습과제 7-2

Implement Batch Gradient Descent with early stopping for Softmax Regression (without using Scikit-Learn).

(참조: https://github.com/ageron/handson-ml2)

★ 반드시 결과 값 및 결과로 나온 모든 그래프에 대해 자세한 분석을 하시오.