AI 실습

Contents

4.1. Classification(인식)

- Credit scoring example:
  - Inputs are income and savings
  - Output is low-risk vs. high-risk
- Formally speaking
  - Input: $\mathbf{x} = [x_1, x_2]^T$
  - Output: $C \in \{0, 1\}$
- Decision rule: if we know $P(C|X_1, X_2)$

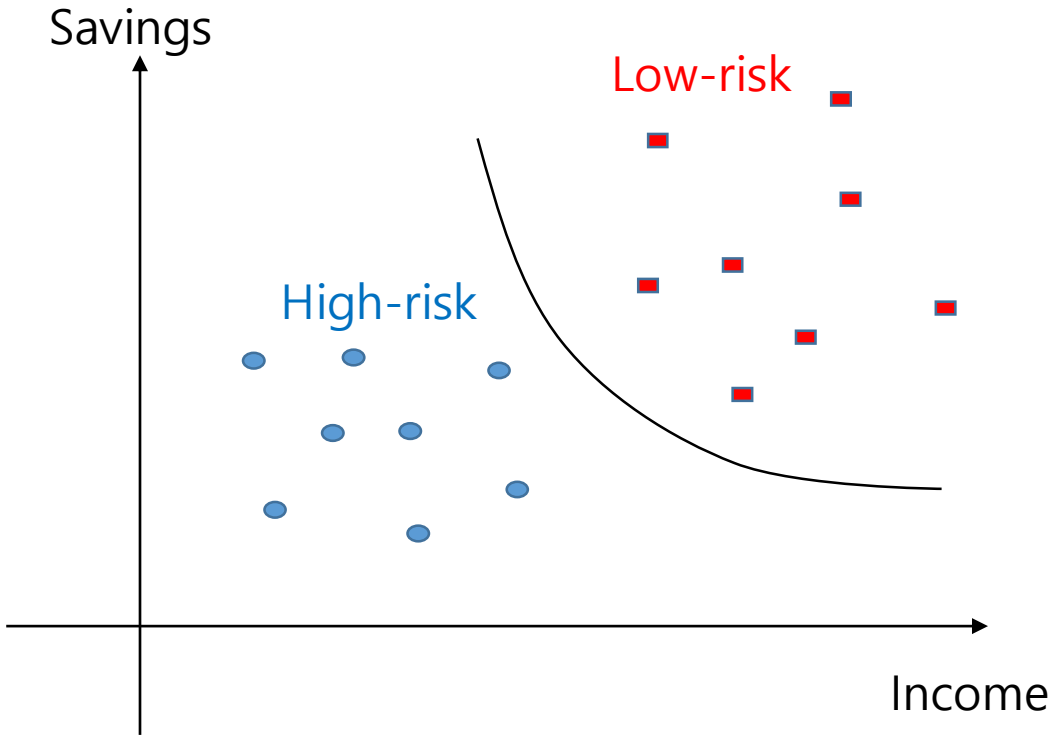
$$\text{choose} \begin{cases} C = 1 \text{ if } P(C = 1|x_1, x_2) > 0.5 \\ C = 0 \text{ otherwise} \end{cases}$$

  - Or equivalently,

$$\text{choose} \begin{cases} C = 1 \text{ if } P(C = 1|x_1, x_2) > P(C = 0|x_1, x_2) \\ C = 0 \text{ otherwise} \end{cases}$$

  - And the probability of error:

$$1 - \max\left[P(C = 1|x_1, x_2), P(C = 0|x_1, x_2)\right]$$



4.2. Bayes' Optimal Classifier

- Choose $\arg\max_{C_k} P(C_k|\mathbf{x})$

4.3. Losses and Risks (손실과 위험)

- Credit scoring problem
  - Accept low-risk applicant → **increasing profits**
  - Reject high-risk applicant → **decreasing losses**
  - ➢ increased loss by accepted high-risk applicant ≠ decreased gains by rejected low-risk applicant
  - **Errors are not symmetric!** → Maximizing gains? Minimizing Losses?
- Define

  α_i: Action assigning input to class C_i

  λ_{ik}: Loss of α_i although the real class is C_k

- Expected risk: $R(\alpha_i|\boldsymbol{x}) = \sum_{k=1}^{K} \lambda_{ik} P(C_k|\boldsymbol{x})$

- Decision rule (minimum risk classifier): Choose $\arg\min_{\alpha_k} R(C_k|\mathbf{x})$

참조:

이해를 돕기 위하여 두 부류(Two-Class) 문제에 대한 최소 위험도 분류기를 다루어보자. 클래스에 대한 손실을 도표화 시켜보면 다음과 같다.

		True Classes	
		C_1	C_2
Hypothesized Classes	$C_1(\alpha_1)$	λ_{11}	λ_{12}
	$C_2(\alpha_2)$	λ_{21}	λ_{22}

이 경우 각 클래스별로 위험도 기대치는

$$R(\alpha_1|\boldsymbol{x}) = \lambda_{11}P(C_1|\boldsymbol{x}) + \lambda_{12}P(C_2|\boldsymbol{x}) \tag{4.3.3}$$

$$R(\alpha_2|\boldsymbol{x}) = \lambda_{21}P(C_1|\boldsymbol{x}) + \lambda_{22}P(C_2|\boldsymbol{x}) \tag{4.3.4}$$

와 같이 계산된다. 만약, 클래스를 맞추면 손실이 0 ($\lambda_{ii} = 0$)이고, 클래스가 틀리면 손실이 1($\lambda_{ik} = 1, i \neq k$)이 되도록 – 이를, "0/1 손실" 이라고 함 – 정해지면 $R(\alpha_1|\boldsymbol{x}) = P(C_2|\boldsymbol{x})$이 되고 $R(\alpha_2|\boldsymbol{x}) = P(C_1|\boldsymbol{x})$이 되어, 위험도가 작은 클래스로 결정하는 것은 확률이 큰 클래스로 결정하는 것과 같아진다.

예제 4.3-1

두 부류 문제에서 C_1이 아주 중요하여 C_2로 판별되면 손실이 심각한 경우를 가정하여, 손실표가 아래와 같이 주어졌다. 판별식을 구하고 판별 영역을 $(P(C_1|\boldsymbol{x}), P(C_2|\boldsymbol{x}))$ 공간에 표시하라. 이를 "0/1" 손실인 경우와 비교하여 보라.

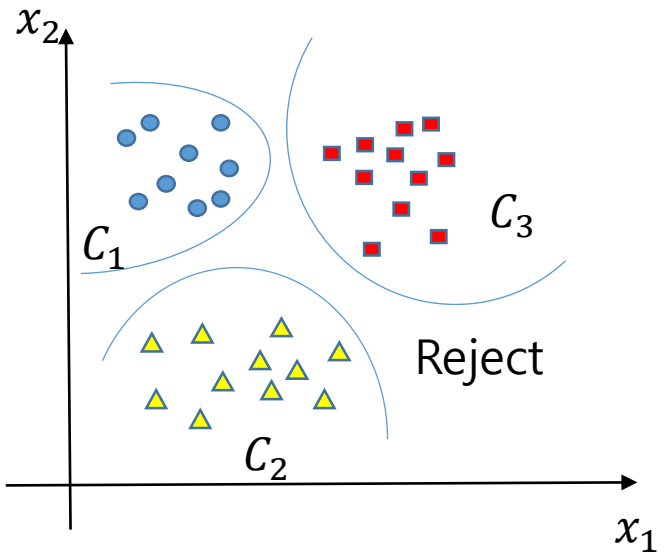
		True Classes	
		C_1	C_2
Hypothesized Classes	$C_1(\alpha_1)$	0	1
	$C_2(\alpha_2)$	2	0

4.4. Discriminant Functions(구별함수)

인식 = 구별함수 $g_i(\boldsymbol{x})$

- Choose C_i if $g_i(\mathbf{x}) = \max_k g_k(\mathbf{x})$
- Note:

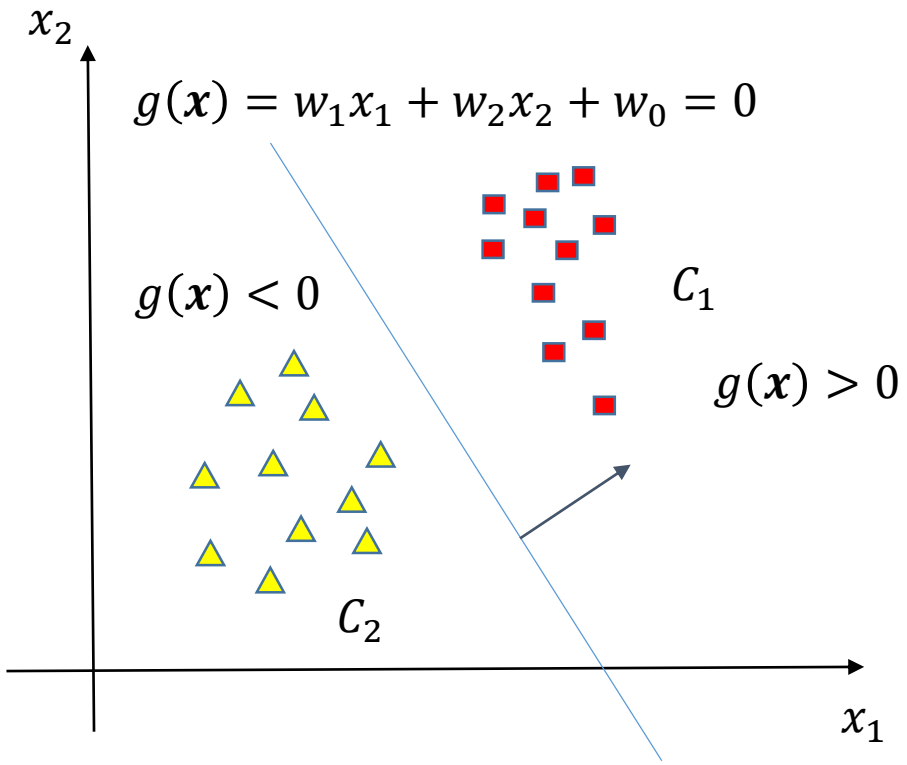
$$g_i(\mathbf{x}) = \begin{cases} -R(\alpha_i|\mathbf{x}) & \text{Minimum risk classifier} \\ P(C_i|\mathbf{x}) & \text{Bayes classifier with 0/1 loss} \\ P(C_i)p(\mathbf{x}|C_i) & \text{ditto} \end{cases}$$



경계면을 찾아 내어도 충분함.
경계면 내의 지점에 대한 확률값을 알아낼 필요 없음
=> 구별함수

4.5. Linear Discriminant Function

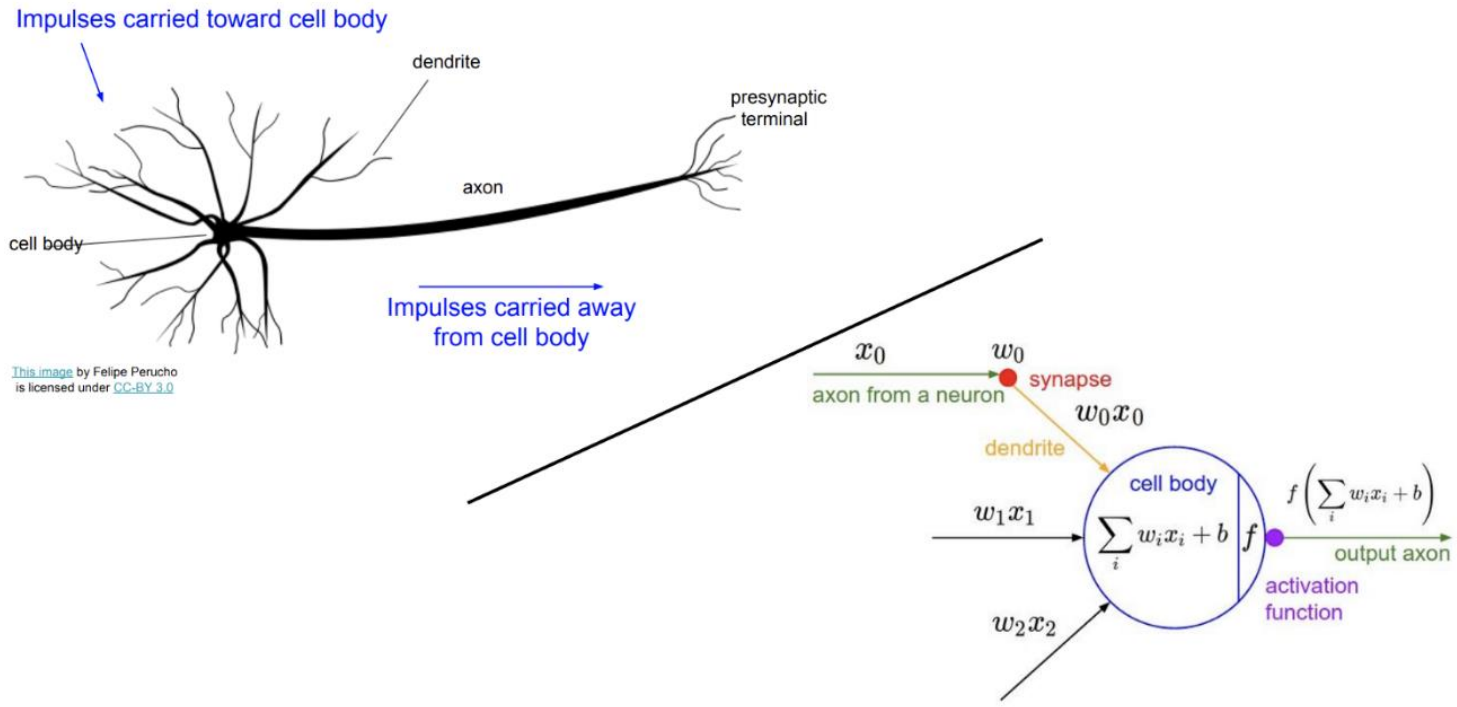
$$g_i(\boldsymbol{x}) = \sum_{j=1}^{d} w_{ij}x_j + w_{i0}$$



$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 \tag{4.5.2}$$

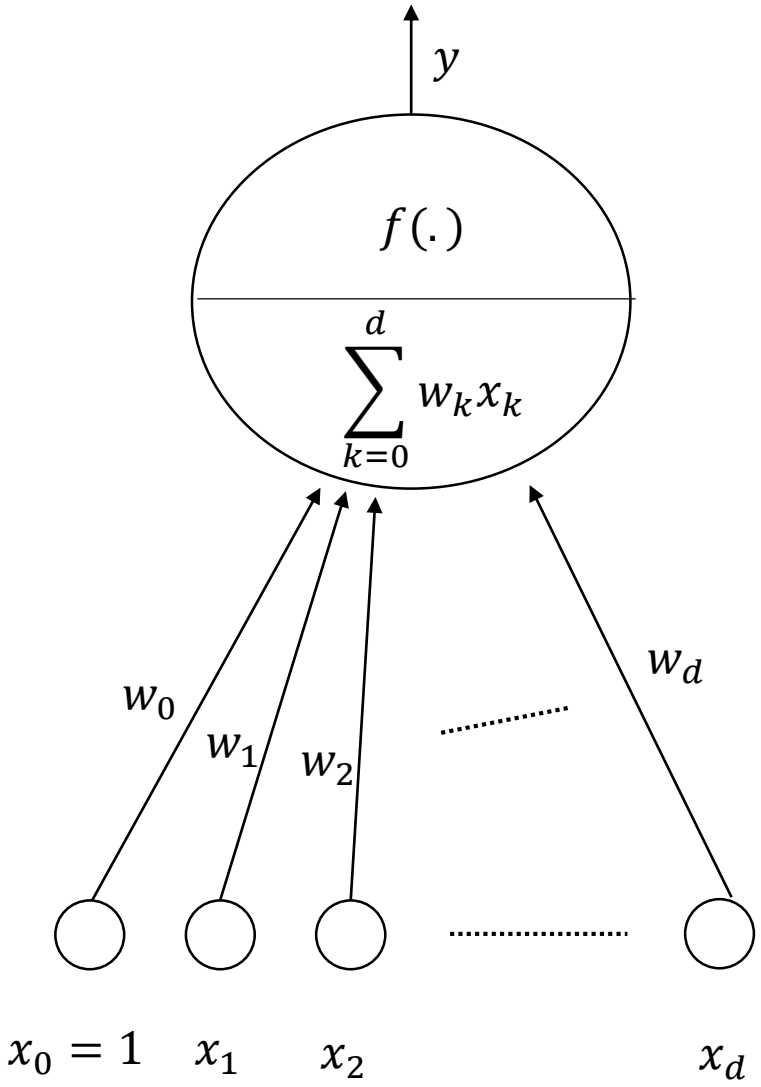
$$\text{choose} \begin{cases} C_1 \text{ if } g(\mathbf{x}) > 0 \\ C_2 \text{ otherwise} \end{cases}$$

4.6. 단층퍼셉트론(Single Layer Perceptron)



Side-by-side illustrations of biological and artificial neurons, via Stanford's CS231n. This analogy can't be taken too literally—biological neurons can do things that artificial neurons can't, and vice versa—but it's useful to understand the biological inspiration. See Wikipedia's description of biological vs. artificial neurons for more detail.

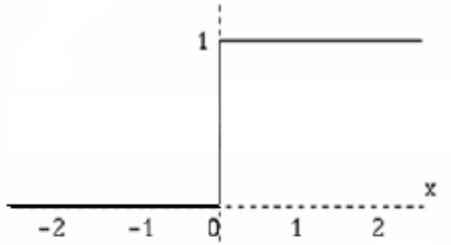
4.6. 단층퍼셉트론(Single Layer Perceptron)



$$\hat{y} = \sum_{k=1}^{d} w_k x_k + w_0 = \sum_{k=0}^{d} w_k x_k \text{ where } x_0 = 1 \qquad (4.6.1)$$

$$y = f(\hat{y}) = \begin{cases} 1 & \text{if } \hat{y} > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (4.6.2)$$

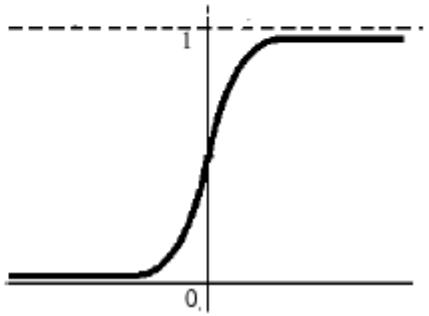
Threshold function: outputs 1 when input is positive and 0 otherwise – perceptron



Logistic sigmoid function

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

Differentiable – a good property for learning



$$y = f(\hat{y}) = \frac{1}{1+e^{-\hat{y}}} \qquad (4.6.3)$$

4.7. Training Perceptron (퍼셉트론 학습)

Gradient Descent (경사 강하법)

Training Sample $\boldsymbol{x}^t = [x_1^t, x_2^t, \ldots, x_d^t]^T$

Perceptron Output y^t ⟷ Desired Output r^t

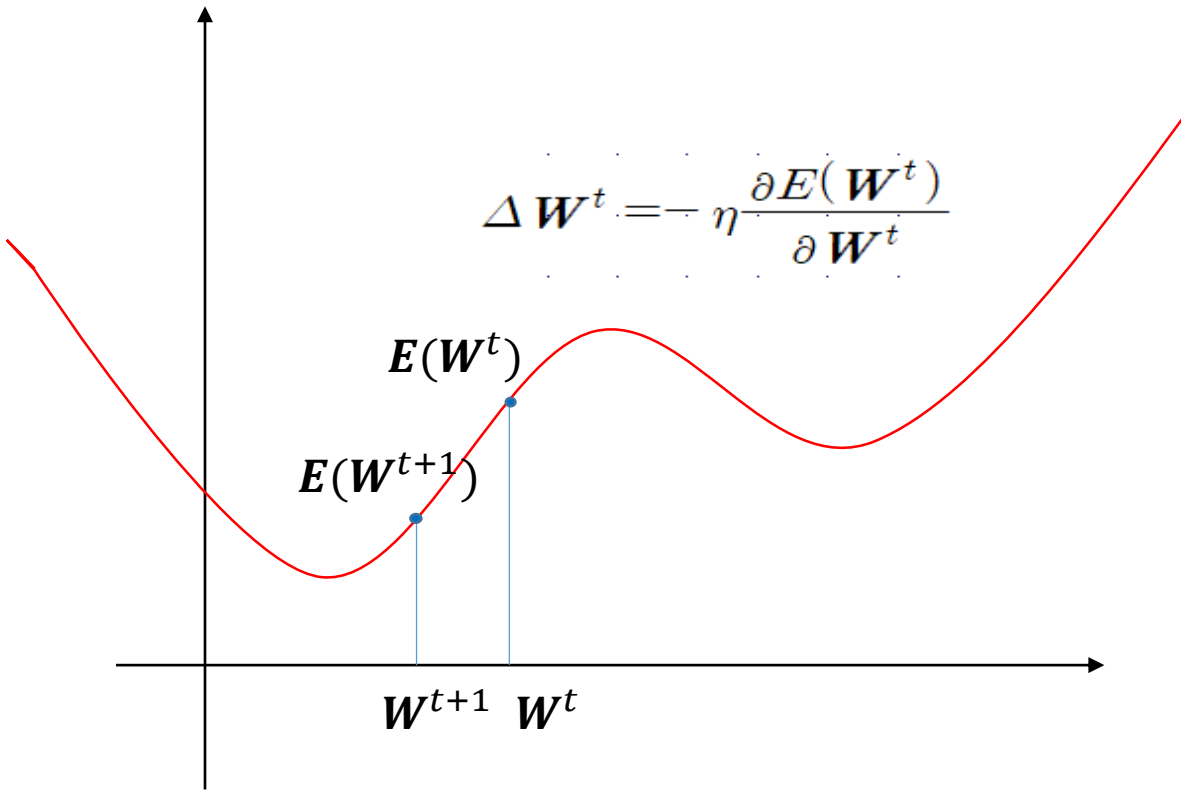
Regression (Linear Output)

$$y^t = \sum_{k=0}^{d} w_k x_k^t \qquad (4.7.1)$$

$$E = \frac{1}{2}(r^t - y^t)^2 \qquad (4.7.2)$$

$$\frac{\partial E}{\partial w_k} = \frac{\partial E}{\partial y^t}\frac{\partial y^t}{\partial w_k} = -(r^t - y^t)x_k^t \qquad (4.7.3)$$

$$\Delta w_k = -\eta\frac{\partial E}{\partial w_k} = \eta(r^t - y^t)x_k^t \qquad (4.7.4)$$



Gradient Descent

Training Sample $\boldsymbol{x}^t = [x_1^t, x_2^t, \ldots, x_d^t]^T$

Perceptron Output y^t ⟷ Desired Output r^t

Classification (Sigmoid Output)

$$\hat{y}^t = \sum_{k=0}^{d} w_k x_k^t \tag{4.7.5}$$

$$y^t = f(\hat{y}^t) = \frac{1}{1+e^{-\hat{y}^t}} \tag{4.7.6}$$

$$E = \frac{1}{2}(r^t - y^t)^2 \tag{4.7.2}$$

$$\frac{\partial E}{\partial w_k} = \frac{\partial E}{\partial y^t}\frac{\partial y^t}{\partial \hat{y}^t}\frac{\partial \hat{y}^t}{\partial w_k} = -(r^t - y^t)y^t(1-y^t)x_k^t \tag{4.7.7}$$

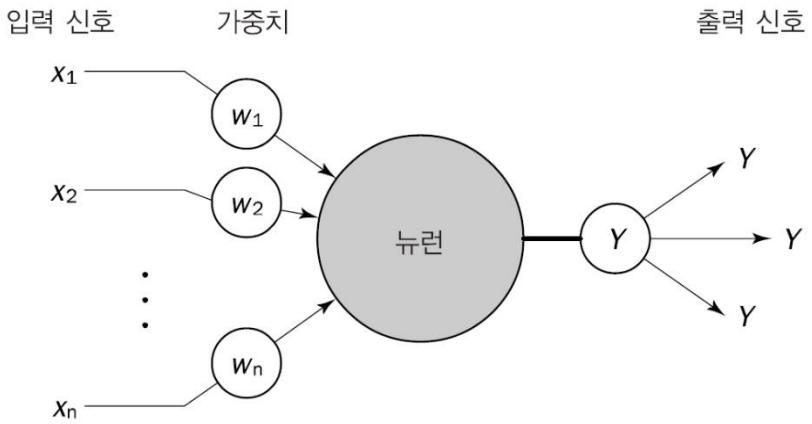
$$\Delta w_k = -\eta\frac{\partial E}{\partial w_k} = \eta(r^t - y^t)y^t(1-y^t)x_k^t \tag{4.7.8}$$

$$E_{CE} = -r^t \log y^t - (1-r^t)\log(1-y^t) \tag{4.7.9}$$

$$\frac{\partial E_{CE}}{\partial w_k} = \frac{\partial E_{CE}}{\partial y^t}\frac{\partial y^t}{\partial \hat{y}^t}\frac{\partial \hat{y}^t}{\partial w_k} = -(r^t - y^t)x_k^t \tag{4.7.10}$$

$$\Delta w_k = -\eta\frac{\partial E_{CE}}{\partial w_k} = \eta(r^t - y^t)x_k^t \tag{4.7.11}$$

Logistic Regression vs. Perceptron



[Scikit-Learn class]
Logistic Regression: 시그모이드 출력에 Cross_Entropy Loss 함수 사용한 경우에 해당
Perceptron: Threshold 출력에 해당

$$X = \sum_{i=1}^{n} x_i w_i$$

$$Y = \begin{cases} +1 & \text{if } X \geq \theta \\ -1 & \text{if } X < \theta \end{cases}$$

$$Y = sign\left[\sum_{i=1}^{n} x_i w_i - \theta\right]$$

$$\Delta w_i = \eta(t - y)\chi_i$$

퍼셉트론 알고리즘

○입력과 목표값의 쌍으로 구성된 학습패턴 $D=\{(\boldsymbol{x}^t,r^t)\}_{t=1}^{P}$를 저장한다.

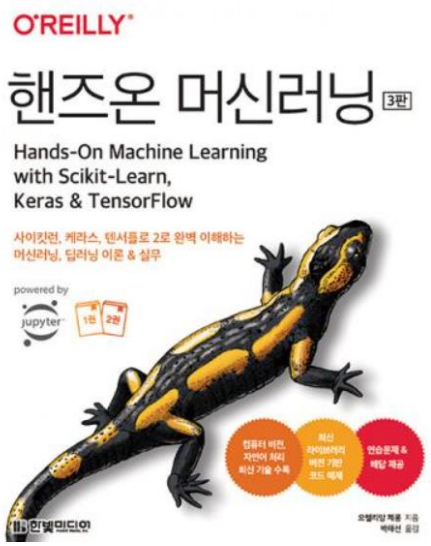
① 가중치를 임의의 값으로 초기화 시킨다.

② 입력 $\boldsymbol{x}^t(t=1,2,3,\ldots,P)$에 대하여 출력 y^t를 계산한다.

③ 가중치 변경식에 따라 가중치를 변경한다.

④ 오차가 원하는 수준 이하이면 학습을 종료시키고, 그렇지 않으면 ②부터 다시 수행한다.

Perceptron 실습

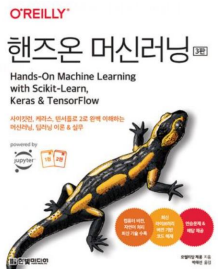



```
>>> from sklearn.datasets import load_iris
>>> iris=load_iris(as_frame=True)
>>> list(iris)
['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename']
>>> iris.data.head(3)
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
>>> iris.target.head(3)
0    0
1    0
2    0
Name: target, dtype: int32
>>> iris.target_names
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

그림 4-22 세 종류의 붓꽃[35]

perceptron-10장-p217~218-p378-3rdEd.py

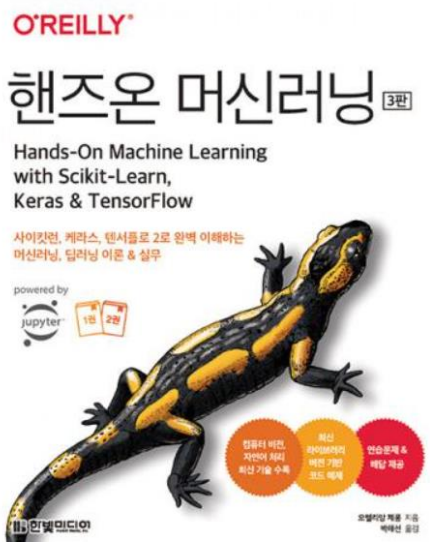
Perceptron 실습




```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

X=iris.data[["petal width (cm)"]].values
y=iris.target_names[iris.target]=='virginica'
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=42)

log_reg= LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)


log_reg.predict(X_test)


>>> log_reg.predict(X_test)
array([False, False,  True, False, False, False, False,  True, False,
       False,  True, False, False, False, False, False,  True, False,
       False,  True, False,  True, False,  True,  True,  True,  True,
        True, False, False, False, False, False, False, False,  True,
       False, False])
```

perceptron-10장-p217~218-p378-3rdEd.py

Perceptron 실습

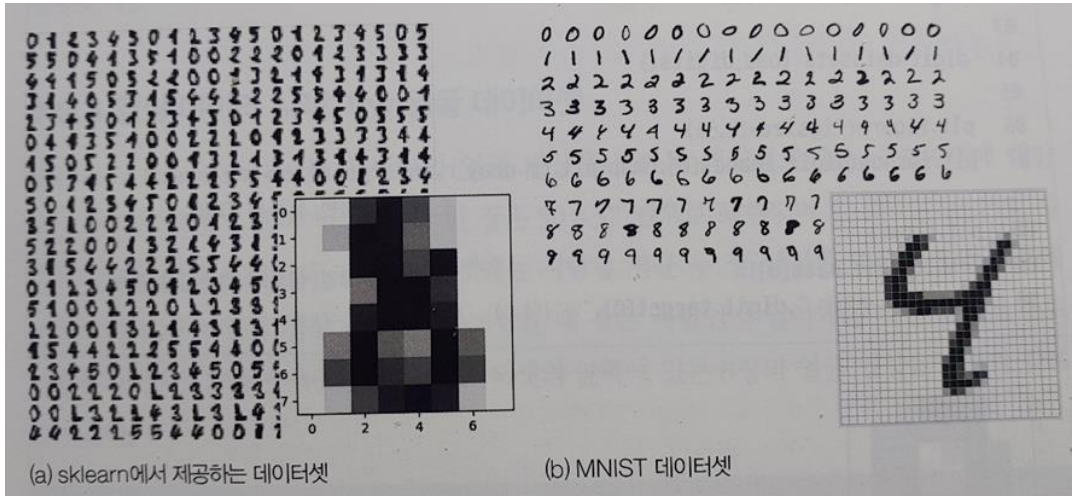



```
import numpy as np

from sklearn.linear_model import Perceptron
X=iris.data[["petal length (cm)","petal width (cm)"]].values
y=(iris.target==0) #Iris-setosa

per_clf = Perceptron(random_state=42)
per_clf.fit(X,y)

X_new=[[2,0.5],[3,1]]
y_pred=per_clf.predict(X_new)
```

```
>>> y_pred
array([ True, False])
```

perceptron-10장-p217~218-p378-3rdEd.py

4-2.py




```
from sklearn import datasets
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
import numpy as np

# 데이터셋을 읽고 훈련 집합과 테스트 집합으로 분할
digit=datasets.load_digits()
x_train,x_test,y_train,y_test=train_test_split(digit.data,digit.target,train_size=0.6)

# fit 함수로 Perceptron 학습
p=Perceptron(max_iter=100,eta0=0.001,verbose=0)
p.fit(x_train,y_train) # digit 데이터로 모델링

res=p.predict(x_test) # 테스트 집합으로 예측

# 혼동 행렬
conf=np.zeros((10,10))
for i in range(len(res)):
    conf[res[i]][y_test[i]]+=1
print(conf)

# 정확률 계산
no_correct=0
for i in range(10):
    no_correct+=conf[i][i]
accuracy=no_correct/len(res)
print("테스트 집합에 대한 정확률은 ", accuracy*100, "%입니다.")
```

```
[[69.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0. 70.  0.  0.  0.  0.  0.  0.  0.  3.]
 [ 0.  0. 65.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1. 65.  0.  0.  0.  0.  1.  0.]
 [ 1.  0.  0.  0. 74.  0.  0.  0.  1.  0.]
 [ 1.  2.  0.  3.  0. 79.  1.  0.  1.  2.]
 [ 0.  3.  0.  0.  0.  0. 63.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.  1.  0. 73.  1.  1.]
 [ 0.  3.  0.  2.  1.  0.  0.  0. 61.  4.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.  0. 64.]]
테스트 집합에 대한 정확률은  94.99304589707927 %입니다.
```