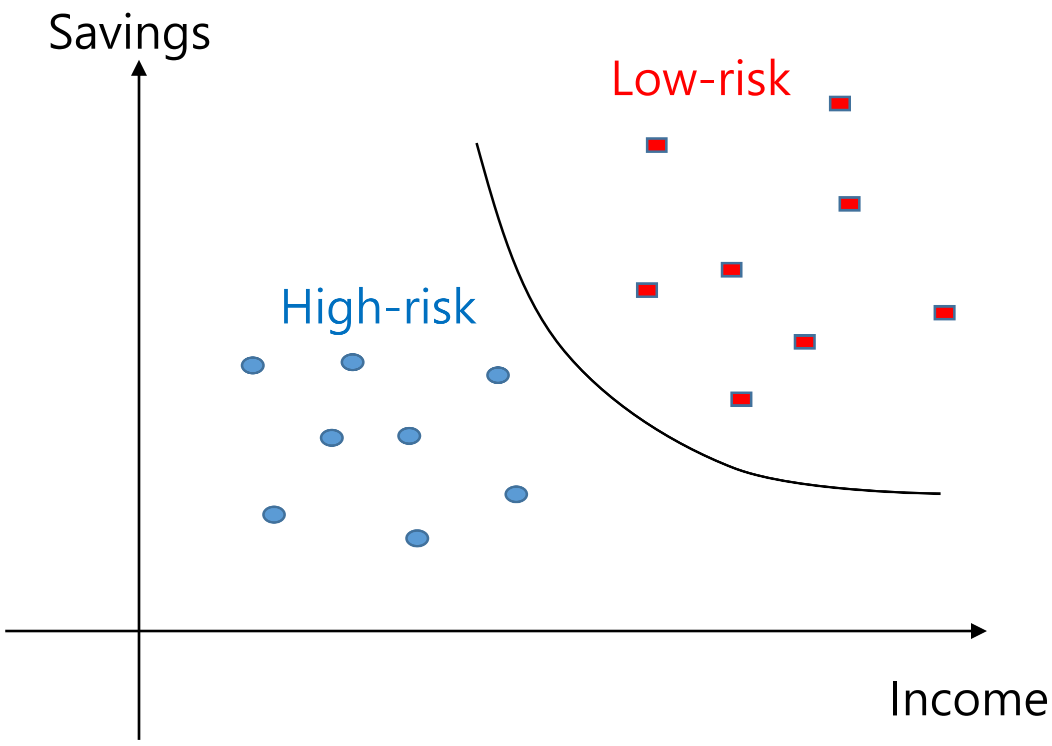
Machine Learning

Contents

1. Introduction
2. K-Nearest Neighbor Algorithm
3. LDA(Linear Discriminant Analysis)
4. **Perceptron**
5. Feed-Forward Neural Networks
6. RNN(Recurrent Neural Networks)
7. SVM(Support Vector Machine)
8. Ensemble Learning
9. CNN(Convolutional Neural Network)
10. PCA(Principal Component Analysis)
11. ICA(Independent Component Analysis)
12. Clustering
13. GAN(Generative Adversarial Network)

4.1. Classification

- 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

- Bayes rule for one concept

posterior, prior, likelihood, evidence

$$P(C|\mathbf{x}) = \frac{P(C)p(\mathbf{x}|C)}{p(\mathbf{x})}$$

- Bayes rule for K > 1 concepts

$$P(C_i|\mathbf{x}) = \frac{P(C_i)p(\mathbf{x}|C_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|C_i)P(C_i)}{\sum_{k=1}^{K} p(\mathbf{x}|C_k)P(C_k)}$$

- Decision rule using Bayes rule (**Bayes optimal classifier**):
  - Choose $\arg\max_{C_k} P(C_k|\mathbf{x})$

참조:

베이의 법칙에 의해

$$P(C|\boldsymbol{x}) = \frac{P(C, \boldsymbol{x})}{P(\boldsymbol{x})} \tag{4.2.4}$$

이고,

$$P(\boldsymbol{x}|C) = \frac{P(C, \boldsymbol{x})}{P(C)} \tag{4.2.5}$$

이다. 또한, C가 K 가지일 경우 결합확률에 대하여

$$P(\boldsymbol{x}) = \sum_{k=1}^{K} P(\boldsymbol{x}, C_k) = \sum_{k=1}^{K} P(\boldsymbol{x}|C_k)P(C_k) \tag{4.2.6}$$

이 성립된다.

4.3. Losses and Risks

- Back to credit scoring example
  - Accepted low-risk applicant **increases profit**, Rejected high-risk applicant **decreases loss**
  - In general, loss by accepted high-risk applicant ≠ potential gain by rejected low-risk applicant
  - **Errors are not symmetric!**
- Define
  - α_i : Action assigning input to class C_i
  - λ_{ik}: Loss of α_i when the actual class is C_k
- Expected risk: $$R(\alpha_i|\mathbf{x}) = \sum_{k=1}^{K} \lambda_{ik} P(C_k|\mathbf{x})$$
- Decision rule (minimum risk classifier):
  - Choose $\arg\min_{\alpha_k} R(\alpha_k|\mathbf{x})$

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.

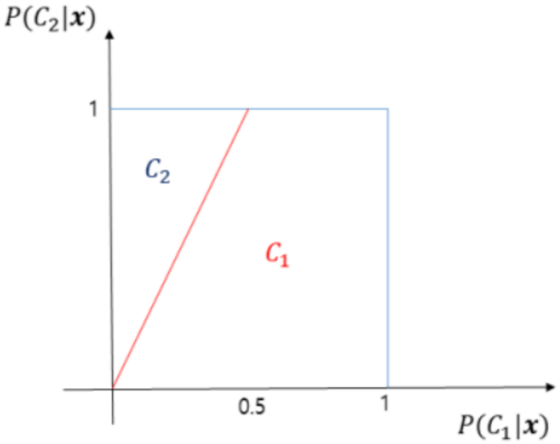
		True Classes	
		C_1	C_2
Hypothesized Classes	$C_1(\alpha_1)$	0	1
	$C_2(\alpha_2)$	2	0

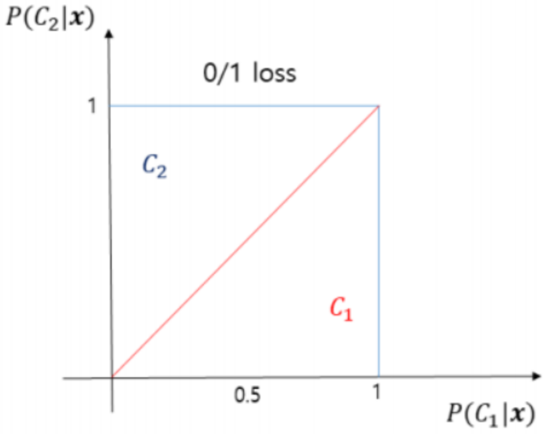
풀이

각 클래스별 위험도 기대치는 $R(\alpha_1|\boldsymbol{x}) = P(C_2|\boldsymbol{x})$, $R(\alpha_2|\boldsymbol{x}) = 2P(C_1|\boldsymbol{x})$이다. 그러면 판별식은

$$\arg\min_{\alpha_k} R(\alpha_k|\boldsymbol{x}) = \begin{cases} C_1, \text{ if } R(\alpha_1|\boldsymbol{x}) < R(\alpha_2|\boldsymbol{x}) \\ C_2, \text{ otherwise} \end{cases} = \begin{cases} \mathrm{C}_1, \text{ if } P(C_2|\boldsymbol{x}) < 2P(C_1|\boldsymbol{x}) \\ C_2, \text{ otherwise} \end{cases}$$

이다. 즉, C_1으로 판별되는 영역은 $2P(C_1|\boldsymbol{x}) - P(C_2|\boldsymbol{x}) > 0$이다. 이를 $(P(C_1|\boldsymbol{x}), P(C_2|\boldsymbol{x}))$ 공간에 표시하면 아래와 같다. “0/1 손실”인 경우보다 C_1의 영역이 크게 확대되었다.





More on Losses and Risks

0/1 loss

$$\lambda_{ik} = \begin{cases} 0 \text{ if } i = k \\ 1 \text{ if } i \neq k \end{cases}$$

$$R(\alpha_i|\mathbf{x}) = \sum_{k=1}^{K} \lambda_{ik} P(C_k|\mathbf{x}) = \sum_{k \neq i} P(C_k|\mathbf{x}) = 1 - P(C_i|\mathbf{x})$$

- For minimum risk, choose the most probable class

Rejection

$$\lambda_{ik} = \begin{cases} 0 \text{ if } i = k \\ \lambda \text{ if } i = K+1, 0 < \lambda < 1 \\ 1 \text{ otherwise} \end{cases}$$

$$R(\alpha_{K+1}|\mathbf{x}) = \sum_{k=1}^{K} \lambda P(C_k|\mathbf{x}) = \lambda$$

$$R(\alpha_i|\mathbf{x}) = \sum_{k \neq i} P(C_k|\mathbf{x}) = 1 - P(C_i|\mathbf{x})$$

- Decision rule:

Choose C_i if $R(\alpha_i|\mathbf{x}) < R(\alpha_k|\mathbf{x}), \forall k \neq i$ and $R(\alpha_i|\mathbf{x}) < R(\alpha_{K+1}|\mathbf{x})$

Reject if $R(\alpha_{K+1}|\mathbf{x}) < R(\alpha_i|\mathbf{x}), i = 1, \ldots, K$

Choose C_i if $P(C_i|\mathbf{x}) > P(C_k|\mathbf{x}), \forall k \neq i$ and $P(C_i|\mathbf{x}) > 1 - \lambda$

Reject otherwise

4.4. Discriminant Functions

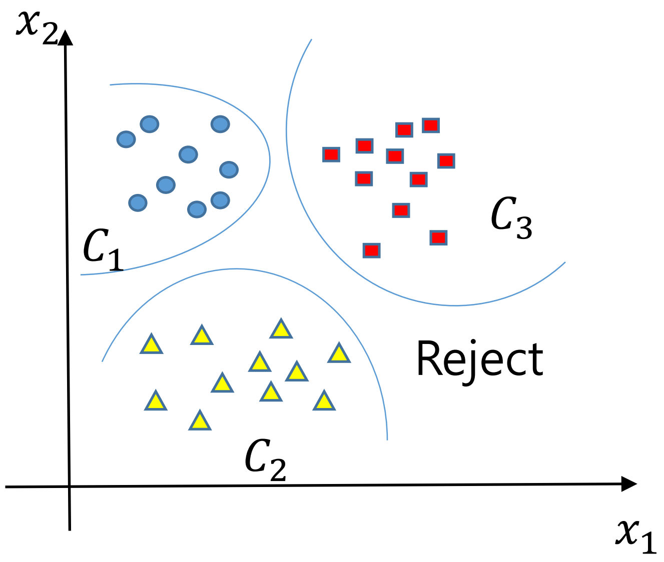
- Classification = implementing a set of discriminant functions $g_i(x)$
  - Choose C_i if $g_i(\mathbf{x}) = \max_k g_k(\mathbf{x})$
  - Note:

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

- Discriminant function divides the input space into K decision regions
  - $\mathcal{R}_i = \left\{ \mathbf{x} | g_i(\mathbf{x}) = \max_k g_k(\mathbf{x}) \right\}$
  - If K=2, $g(x) = g_1(x) - g_2(x)$ and

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

  - When is ...? $g(\mathbf{x}) = \log \frac{P(C_1|\mathbf{x})}{P(C_2|\mathbf{x})}$



Likelihood-based vs. Discriminant-based

- Likelihood-based classification

  Learn (estimate) distribution $p(\mathbf{x}|C_i)$, and use Bayes' rule to calculate $p(C_i|\mathbf{x})$: $g_i(\mathbf{x}) = \log P(C_i|\mathbf{x})$

- Discriminant-based classification

  Learn $g_i(\mathbf{x}|\Phi_i)$ directly from data; no density estimation

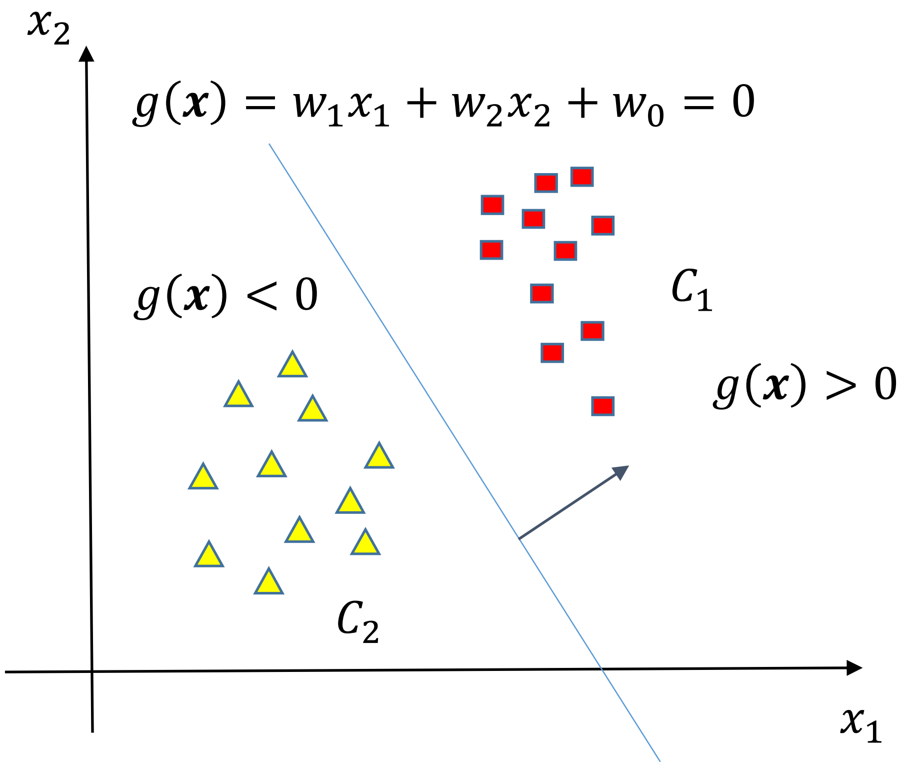
- Estimating the boundaries is enough; no need to accurately estimate the densities inside the boundaries!

4.5. Linear Discriminant Function

- Linear discriminant

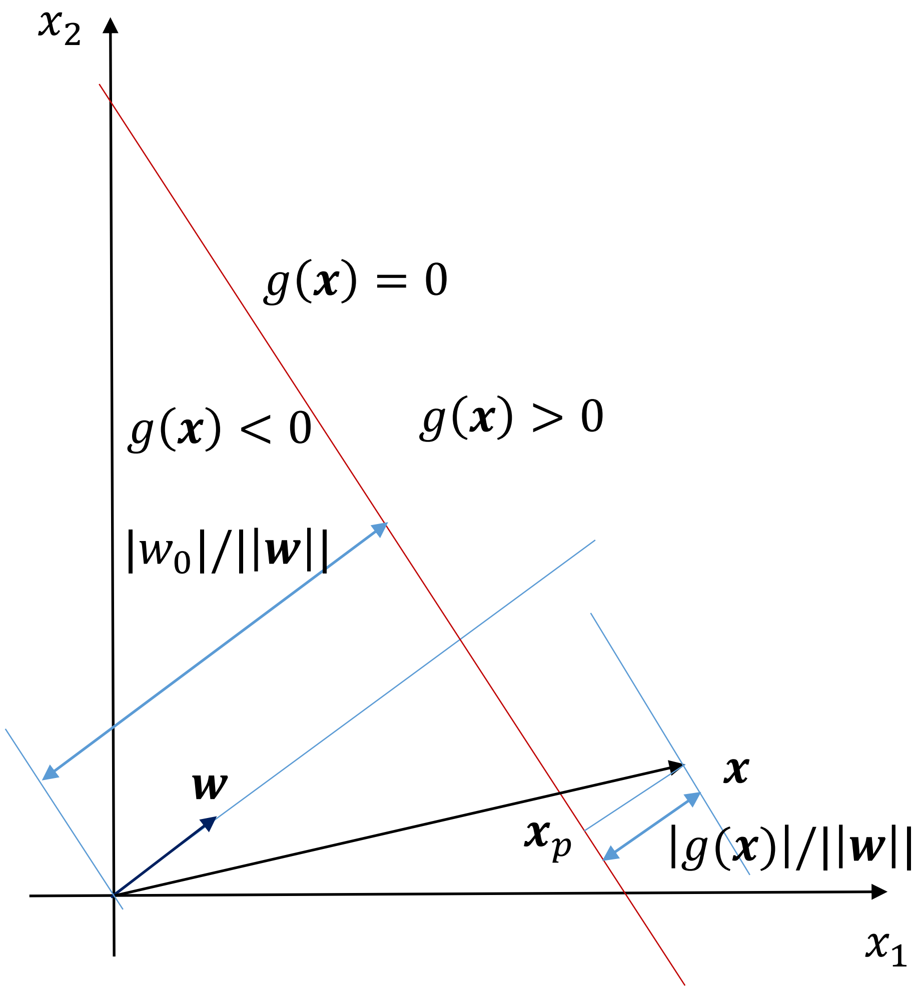
$$g_i(\mathbf{x}|\mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T\mathbf{x} + w_{i0} = \sum_{j=1}^{d} w_{ij}x_j + w_{i0}$$

- Advantages:
  - Simple: O(d) space/computation
  - Knowledge extraction: Weighted sum of attributes; positive/negative weights, magnitudes (credit scoring)
  - Optimal when $p(\mathbf{x}|C_i)$ are Gaussian with shared covariance matrix; useful when classes are (almost) linearly separable



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

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



Two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ on the decison surface

$$\boldsymbol{w}^T\boldsymbol{x}_1 + w_0 = \boldsymbol{w}^T\boldsymbol{x}_2 + w_0 \tag{4.5.3}$$

$$\boldsymbol{w}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0 \tag{4.5.4}$$

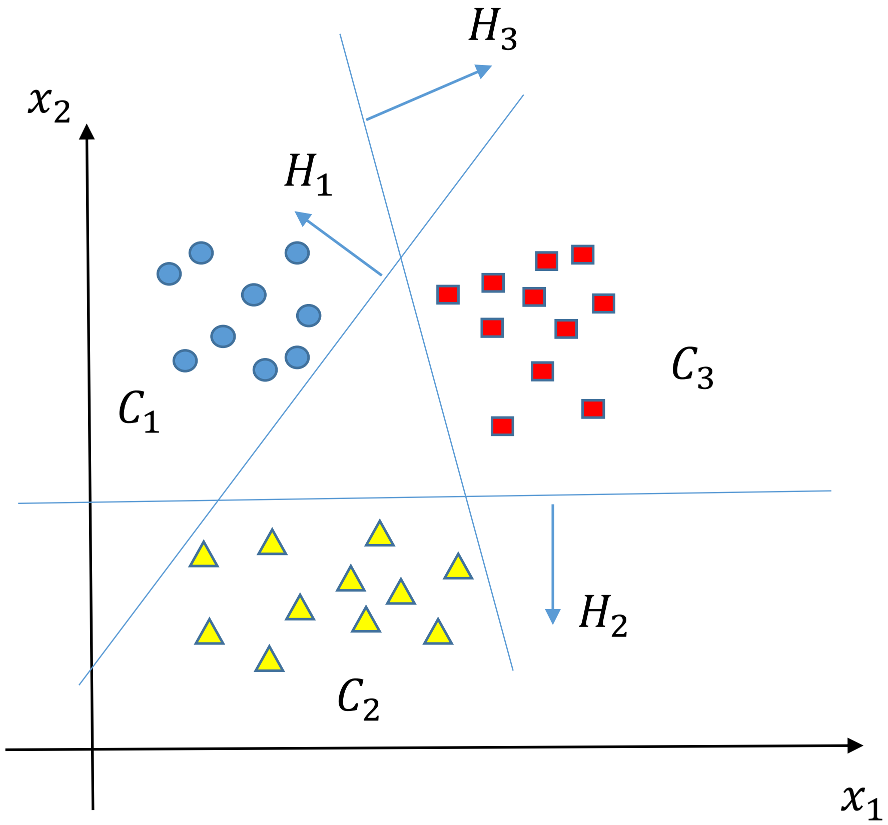
$$\boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \tag{4.5.5}$$

$$r = \frac{g(\boldsymbol{x})}{\|\boldsymbol{w}\|} \tag{4.5.6}$$

Position from the origin

$$r_0 = \frac{g(\boldsymbol{0})}{\|\boldsymbol{w}\|} = \frac{w_0}{\|\boldsymbol{w}\|} \tag{4.5.7}$$

Multiple Classes (One-vs-All)

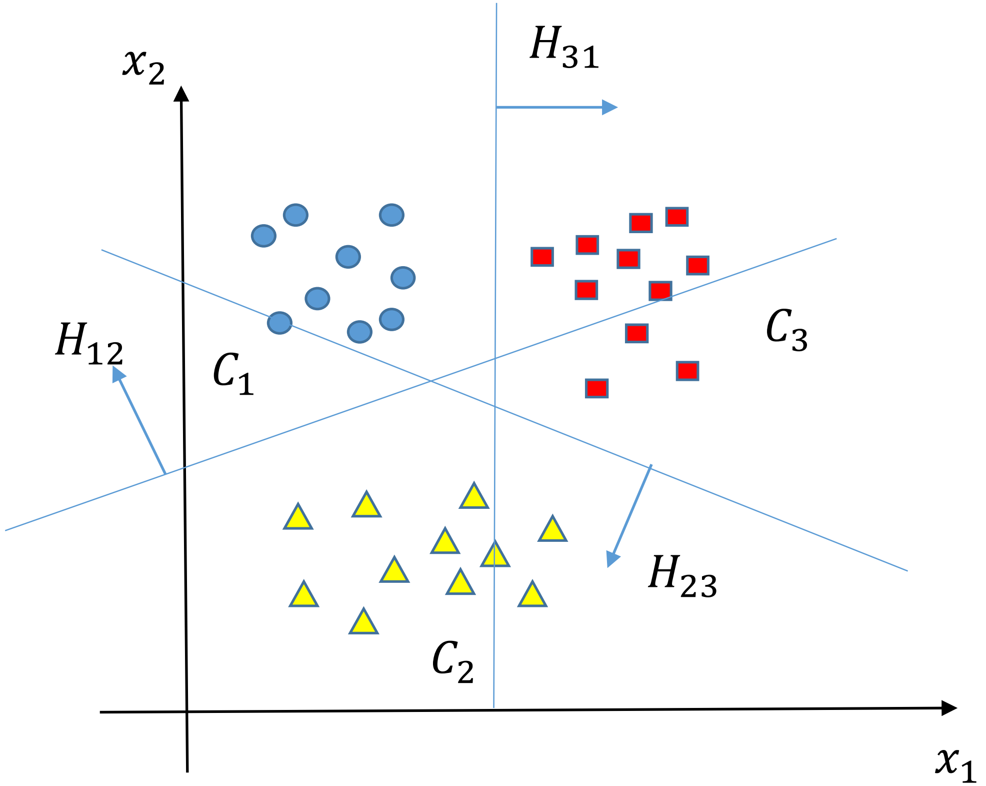


$$g_i(\mathbf{x}|\mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T\mathbf{x} + w_{i0}$$

Choose C_i if $g_i(\mathbf{x}) = \max_j g_j(\mathbf{x})$

Classes are linearly separable

Pairwise Separation (One-vs-One)

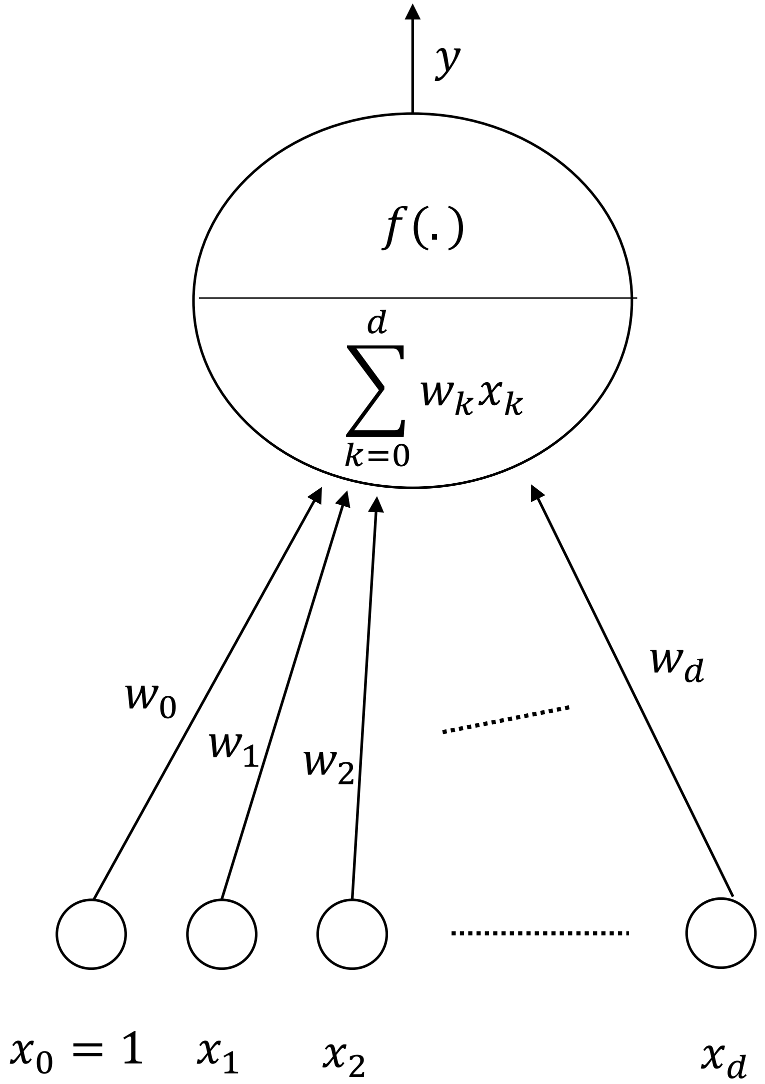


$$g_{ij}(\mathbf{x}|\mathbf{w}_{ij}, w_{ij0}) = \mathbf{w}_{ij}^T\mathbf{x} + w_{ij0}$$

$$g_{ij}(\mathbf{x}) = \begin{cases} > 0 \text{ if } \mathbf{x} \in C_i \\ \leq 0 \text{ if } \mathbf{x} \in C_j \\ \text{don't care otherwise} \end{cases}$$

$$\text{Choose } C_i \text{ if } \forall j \neq i, g_{ij}(\mathbf{x}) > 0$$

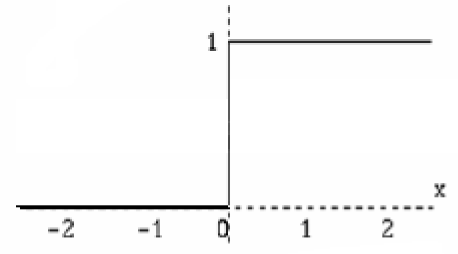
4.6. Single Layer Perceptron



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

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

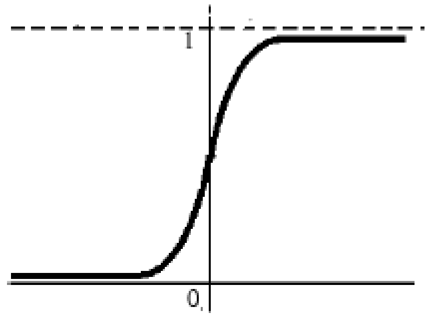
Threshold function: outputs 1 when input is positive and 0 otherwise – perceptron



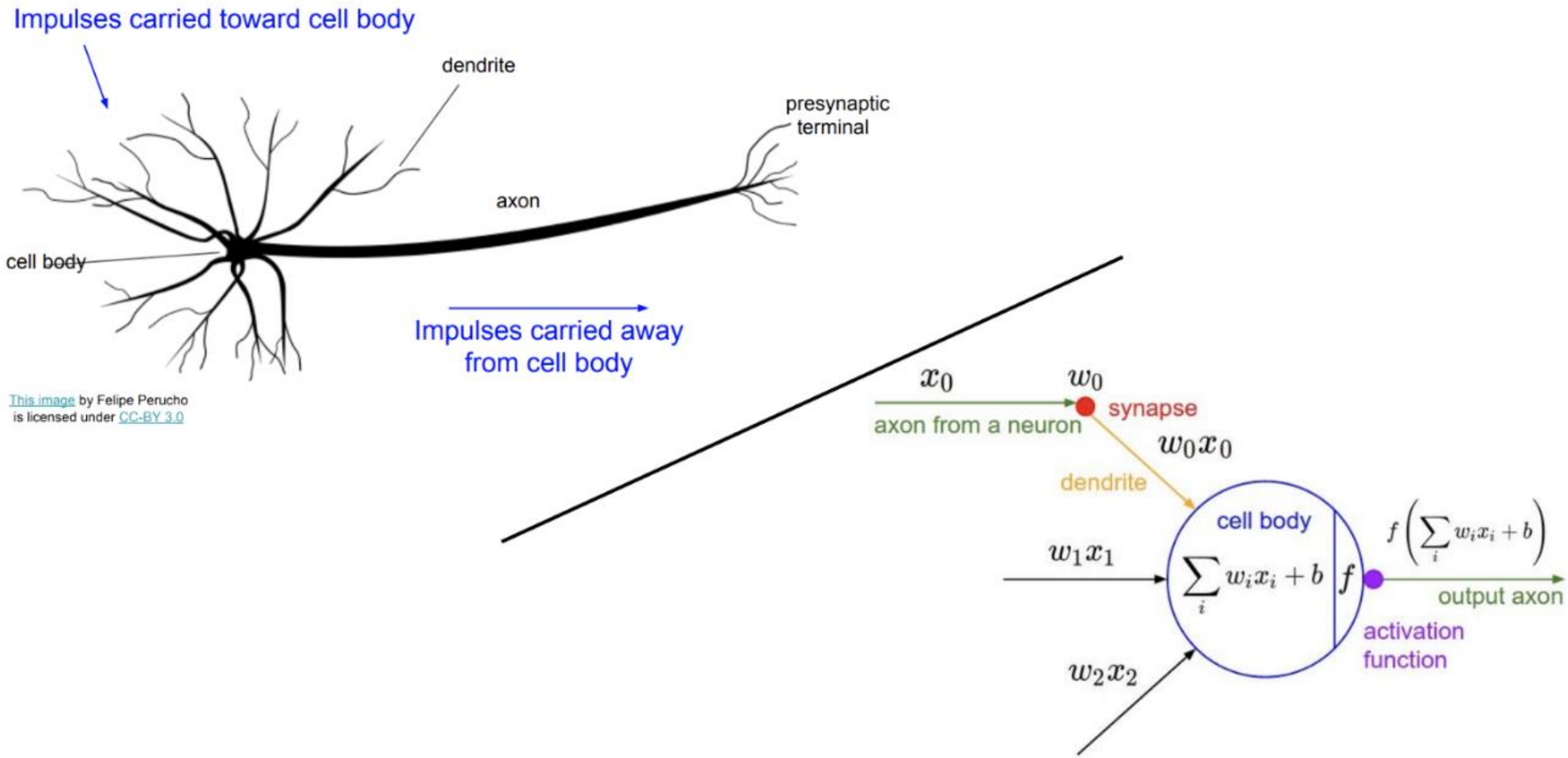
Logistic sigmoid function

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

Differentiable – a good property for learning

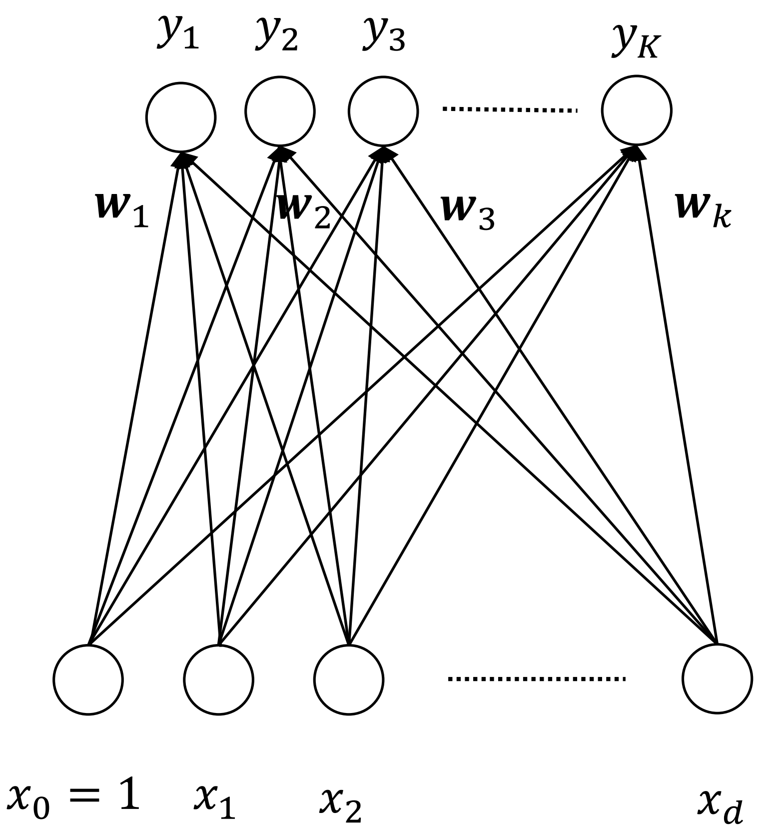


$$y = f(\hat{y}) = \frac{1}{1+e^{-\hat{y}}} \qquad (4.6.3)$$



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.

Single-Layer Perceptron with K Outputs



$$y_k = f(\hat{y_k}) = \frac{1}{1 + e^{-\sum_{i=0}^{d} w_{ki} x_i}} \quad (4.6.4)$$

$$\text{choose } C_i \text{ if } y_i = \max_k y_k$$

4.7. Training Perceptron

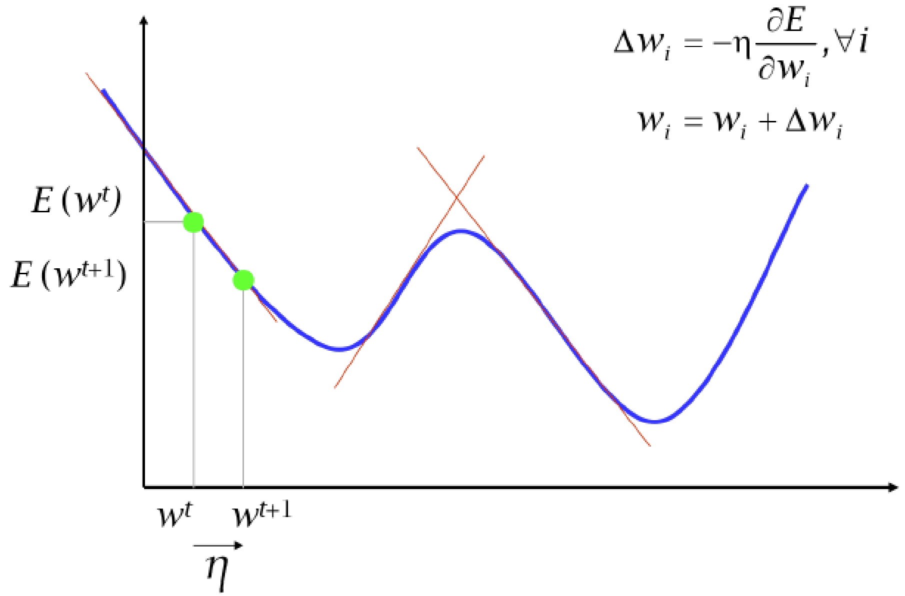
Gradient Descent

$Err(\mathcal{X}|\mathbf{w})$ is the error with parameters w on sample $\mathcal{X}$

- Want: $\mathbf{w}^* = \arg\min_w Err(\mathcal{X}|\mathbf{w})$

Gradient

$$\nabla_w Err = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_d}\right]^T$$



Gradient-descent

- Start from random w and
- update w iteratively in the negative direction of gradient

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

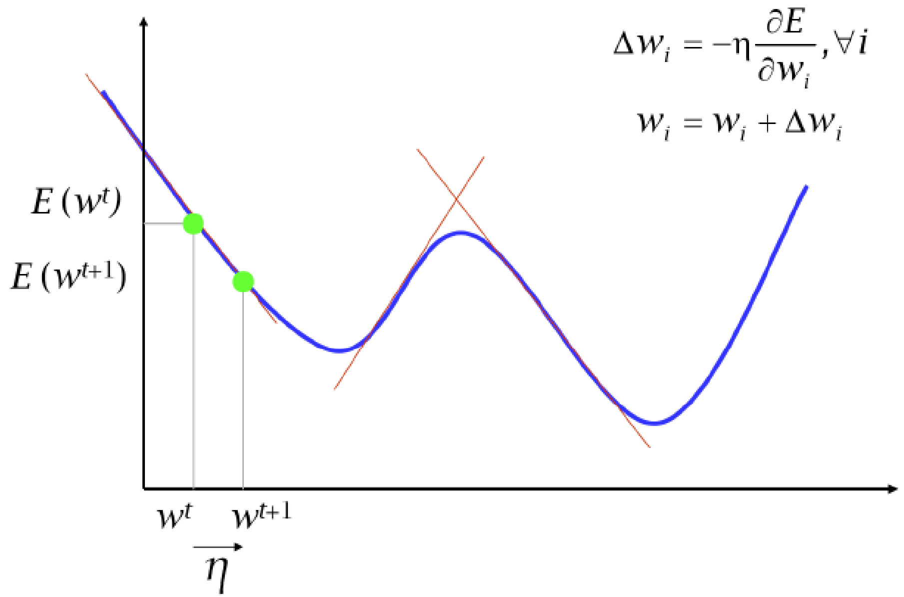
Regression (Linear Output)

$$y^t = \sum_{k=0}^{d} w_k x_k^t \tag{4.7.1}$$

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

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

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



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)

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

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

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

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

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

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

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

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

퍼셉트론 알고리즘

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

① 가중치를 임의의 값으로 초기화 시킨다.

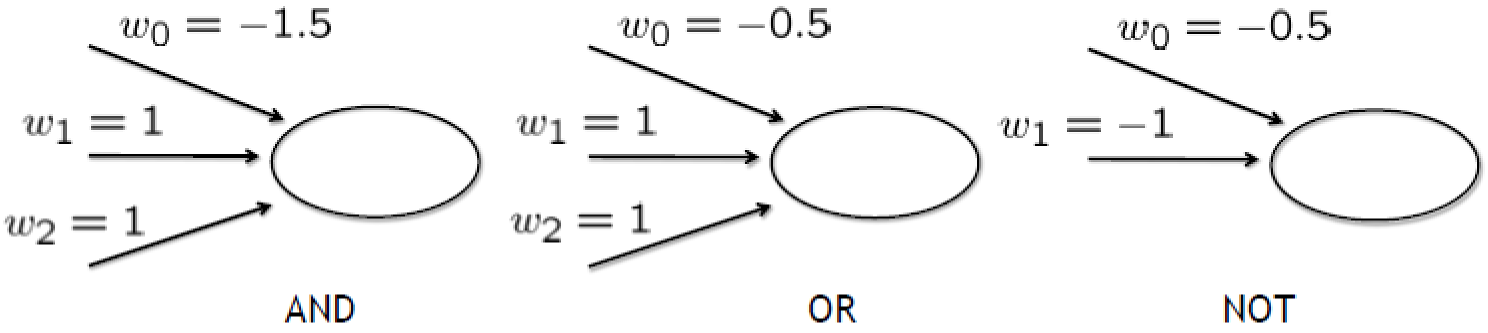
② 입력 $\boldsymbol{x}^t\,(t=1,2,3,\ldots,P)$에 대하여 출력 y^t를 계산한다.

③ 가중치 변경식에 따라 가중치를 변경한다.

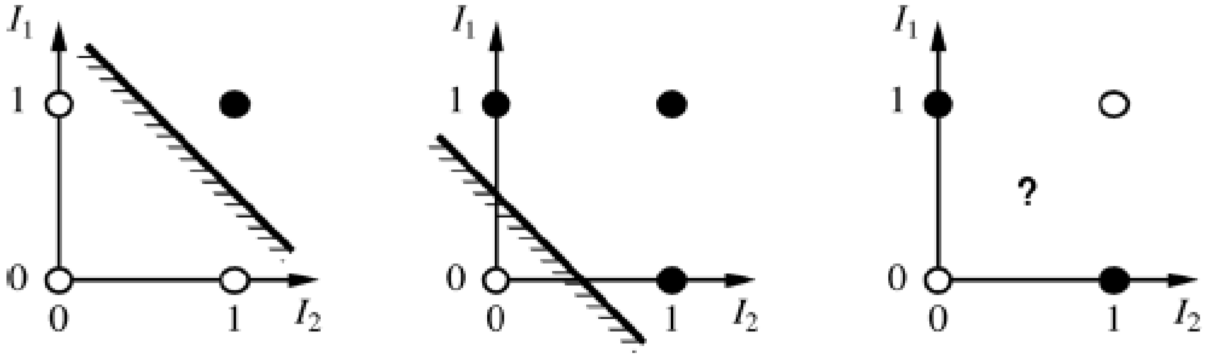
④ 오차가 원하는 수준 이하이면 학습을 종료시키고, 그렇지 않으면 ②부터 다시 수행한다.

Expressiveness of Perceptrons

- Consider perceptron with a = step function



- Can represent AND, OR, NOT, majority, etc., but not XOR

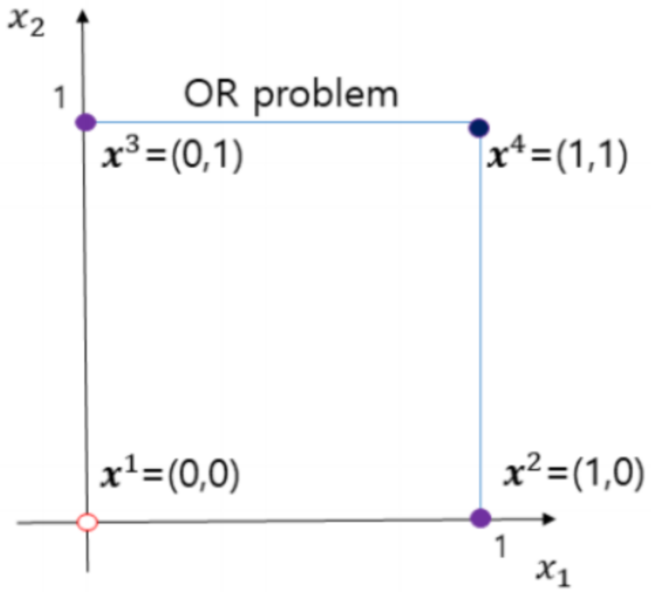


(a) I_1 and I_2 (b) I_1 or I_2 (c) I_1 xor I_2

- Represents a linear separator in input space: $\sum_j w_j x_j > 0 \Leftrightarrow \mathbf{w}^T \mathbf{x} > 0$

예제 4.7-1

아래 그림과 같이 OR 문제를 입력 2 출력 1개의 노드를 지닌 퍼셉트론으로 학습하기 위하여 가중치를 $\boldsymbol{w}=(w_0,w_1,w_2)^T=(0,0.3,0.6)^T$로 초기화 하였다고 가정하자. 출력 목표값은 입력 $\boldsymbol{x}^1$에 대해서만 0이고 나머지 입력 $\boldsymbol{x}^2,\boldsymbol{x}^3,\boldsymbol{x}^4$에 대해서는 1이다. 4개의 입력 중 임의의 하나를 골라서 퍼셉트론에 입력하여 CE 오차함수에 따른 가중치 변경량을 구하여 보아라. 학습률은 $\eta=0.1$이고, 퍼셉트론의 출력노드는 시그모이드 활성화 함수로 가정하라.



풀이

$\boldsymbol{x}^4=(1,1)^T$이 입력되었다고 하자. 그러면 퍼셉트론의 출력노드에 대한 가중치 합은 $\hat{y}=(0,0.3,0.6)(1,1,1)^T=0.9$ 이고 출력은 $y=\dfrac{1}{1+\exp(-\hat{y})}=0.7109$ 이다. 목표값이 $r^4=1$이므로 $\Delta w_0=\eta(r-y)=0.02891$이고, $\Delta w_1=\eta(r-y)x_1=0.02891$, $\Delta w_2=\eta(r-y)x_2=0.02891$이다. 따라서, 가중치는 $w_0=w_0+\Delta w_0=0.02891$, $w_1=w_1+\Delta w_1=0.32891$, $w_2=w_2+\Delta w_2=0.62891$ 와 같이 변경된다.

다음으로 $\boldsymbol{x}^1$이 입력되면 $\hat{y}=(0.02891,0.32891,0.62891)(1,0,0)^T=0.02891$이고 $y=\dfrac{1}{1+\exp(-\hat{y})}=0.5072$ 이고, $r^1=0$이므로, $\Delta w_0=\eta(r-y)=-0.05072$, $\Delta w_1=\eta(r-y)x_1=0$, $\Delta w_2=\eta(r-y)x_2=0$이다. 즉, w_1과 w_2는 변동이 없고, $w_0=w_0+\Delta w_0=0.02891-0.05072=-0.0218$로 변경된다.