Chapter 06 선형 판별함수

[학습목표]
이 장에서는 각 클래스별 확률밀도함수를 추정하는 대신 학습을 통하여 결정경계를 직접 찾는 방법에 대하여 알아본다. 특히 이 장에서는 간단한 선형 판별함수에 대해서 살펴보고, 이에 대한 확장으로 11장과 12장에서 정교화된 방법을 소개할 것이다.

6. 선형 판별함수

6.1 판별함수와 패턴인식

4장과 5장에서 분류를 위한 다양한 결정규칙을 알아보았다. 이러한 규칙들은 기본적으로 확률적 접근법을 취하고 있으므로, 먼저 데이터로부터 클래스별 확률밀도함수 $p(\boldsymbol{x}|C_i)$를 추정하고 베이즈 정리에 의해 클래스의 후험확률 $p(C_i|\boldsymbol{x})$를 계산하여 이를 바탕으로 판별함수가 정의되었다. 이 장에서는 관점을 조금 달리하여 판별함수 자체에 주목한다.
먼저 판별함수에 대하여 다시 한 번 정리하면, 두 개의 클래스를 사용하는 이분류 문제의 경우에는 하나의 판별함수 $g(\boldsymbol{x})$가 정의되어 다음과 같은 결정규칙이 사용된다.

$$y(\boldsymbol{x}) = \begin{cases} 1 & \text{if } g(\boldsymbol{x}) > 0 \\ -1 & \text{otherwise} \end{cases} \qquad \text{[식 6-1]}$$

다중 클래스 분류 문제의 경우에는 각 클래스별로 판별함수 $g_i(\boldsymbol{x})$가 존재하여 이를 이용하여 다음과 같은 결정규칙을 사용할 수 있다.

$$y(\boldsymbol{x}) = \text{argmax}_i\{g_i(\boldsymbol{x})\} \qquad \text{[식 6-2]}$$

판별함수 $g_i(\boldsymbol{x})$의 형태는 사용하는 방법론에 따라 매우 다양하게 결정될 수 있다. 지금까지 4장과 5장에서 소개된 판별함수들은 확률분포함수로부터 유도되었으나, 이 장에서는 확률분포 함수를 추정하지 않고 직접적으로 판별함수의 형태를 정의하고 데이터로부터 학습하는 방법에 대하여 생각한다.

6.1.1 판별함수의 형태

확률모델을 세우지 않고 판별함수를 직접 학습하기 위해서는 먼저 그 기본 형태를 정의해 주어야 한다. 가장 간단한 판별함수로 직선 형태(고차원 공간의 경우 초평면)의 선형 판별함수를 생각할 수 있을 것이다. n차원 입력 특징벡터 $\boldsymbol{x}=[x_1,x_2,...,x_n]^T$에 대한 클래스 C_i의 선형 판별함수는 다음과 같이 명시적으로 정의할 수 있다.

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} = \sum_{j=1}^{n} w_{ij}x_j + w_{i0} \qquad \text{[식 6-3]}$$

여기서 파라미터 벡터 $\boldsymbol{w}_i$는 초평면의 기울기를 결정하는 벡터로 초평면의 법선벡터가 되고, w_{i0}는 $\boldsymbol{x}=\boldsymbol{0}$인 원점에서 직선까지의 거리를 결정하는 바이어스이다. 이들은 결국 입력공간에서 결정경계가 자리하는 위치를 결정하는 파라미터로, 데이터를 통해 추정해야 하는 값이다.

특히 입력 특징이 이차원 벡터인 경우, 데이터들은 2차원 공간상의 점들로 표시되고 선형 판별함수는 직선으로 나타난다. 선형 판별함수에 의해서 정해지는 결정경계와 결정영역에 대해서는 다음 절에서 자세히 알아보겠다.
보다 일반적인 형태의 판별함수로 다항식 형태의 판별함수를 생각해 볼 수 있다. 클래스 C_i에 대한 이차 다항식 형태의 판별함수는 다음과 같이 정의된다.

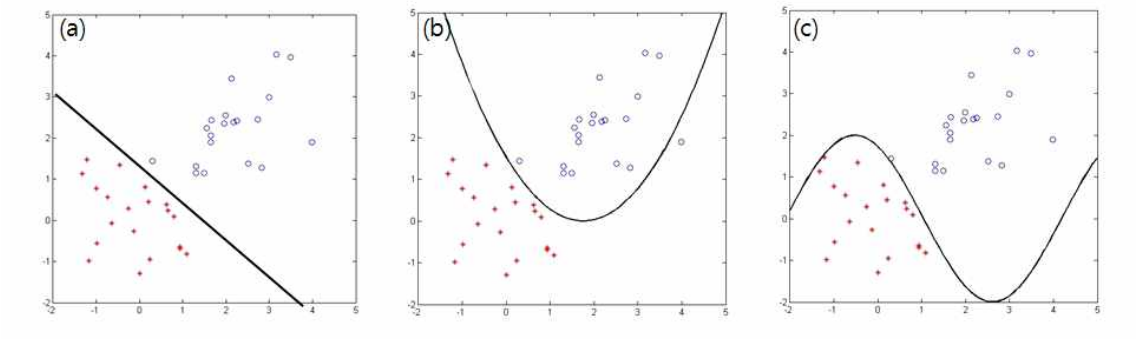
$$g_i(\boldsymbol{x}) = \boldsymbol{x}^T \mathbb{W}^i \boldsymbol{x} + \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} = \sum_{k=1}^{n}\sum_{j=1}^{n} w_{jk}^i x_j x_k + \sum_{j=1}^{n} w_{ij} x_j + w_{i0} \quad \text{[식 6-4]}$$

다항식 판별함수는 선형 판별함수에 비해 훨씬 다양한 결정경계를 표현할 수 있어서, 복잡한 분포를 가진 데이터에 대해서도 적합한 결정경계를 찾을 수 있다. 그러나 [식 6-4]와 [식 6-3]을 비교해 보면, 이차 다항식을 사용할 때 학습을 통해 추정해야 하는 파라미터는 모두 $\mathbb{W}^i$, $\boldsymbol{w}_i$, w_{i0}임을 알 수 있다. 여기서 $\mathbb{W}^i$는 벡터 $\boldsymbol{x}$의 두 원소들로 이루어지는 2차항 $x_j x_k$들의 계수를 원소로 가지는 행렬이므로 n차원 입력 벡터의 경우 n^2개의 원소를 가진다. 따라서 n차원 데이터가 주어진 경우 추정해야할 파라미터의 개수는 모두 n^2+n+1개가 된다. 결국 p차 다항식을 사용할 때 추정해야 하는 파라미터 수는 $O(n^p)$개가 되어 그 수가 기하급수적으로 늘어남을 알 수 있다. 파라미터의 개수가 늘어나면 단순히 계산량이 늘어나는 것 뿐 아니라, 한정된 수의 데이터 집합을 사용할 경우에는 추정의 정확도도 떨어지게 되어 좋은 성능을 가진 분류기를 얻기 힘들다.
선형 분류기의 제약점을 해결하면서 다항식 분류기가 가지는 파라미터 수의 문제도 극복하기 위한 방법으로 비선형 <기저함수 (basis function)>를 사용하는 방법을 생각해 볼 수 있다. 입력공간상에서 정의되는 m개의 비선형 기저함수를 $\phi_i(\boldsymbol{x})$ $(i=1,...,m)$이라고 하면, 이들의 선형합을 이용하여 다음과 같이 판별함수를 정의할 수 있다.

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T \boldsymbol{\phi}(\boldsymbol{x}) + w_{i0} = \sum_{j=1}^{m} w_{ij}\phi_j(\boldsymbol{x}) + w_{i0} \quad \text{[식 6-5]}$$

여기서 $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), ..., \phi_m(\boldsymbol{x})]^T$는 m개의 기저함수들을 요소로 가지는 m차원 열벡터이다. 이때 판별함수 $g_i(\boldsymbol{x})$의 형태는 사용된 기저함수의 형태에 의존하게 되는데, 비선형 기저함수를 사용하므로 써 다양한 형태의 결정경계를 표현할 수 있다. [식 6-3]의 경우는 선형 기저함수로 볼 수 있고, 비선형 기저함수의 예로는 $\sin(\boldsymbol{x})$나 $\cos(\boldsymbol{x})$와 같은 삼각함수나 [식 6-6]과 같이 정의되는 가우시안 함수, 11장에서 소개하는 하이퍼탄젠트 함수나 시그모이드 함수 등을 생각해 볼 수 있다.

$$\phi_j(\boldsymbol{x}) = \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T(\boldsymbol{x}-\boldsymbol{\mu}_j)\right\} \quad \text{[식 6-6]}$$



[그림 6-1] 여러 가지 판별함수와 그에 따른 결정경계:
(a) 선형함수 (b) 2차 다항식 (c) 삼각함수

[그림 6-1]은 2차원 입력공간상에서 정의된 다양한 판별함수에 의해 정해지는 결정경계를 보여주고 있다. (a)는 선형판별함수에 의해 정해지는 직선형태의 결정경계이며, (b)는 2차 함수에 의해 결정되는 이차곡선형태의 결정경계, 그리고 (c)는 삼각함수에 의해 결정되는 곡선형태의 결정경계이다. 세 가지 결정경계 모두 같은 데이터 집합을 오류 없이 분류하고 있음을 알 수 있다. 이와 같이 결정경계는 판별함수에 크게 의존하므로, 어떤 판별함수를 사용하느냐에 따라 분류기의 성능과 계산의 효율성 등이 직접적으로 영향을 받게 된다. 따라서 각 판별함수의 특성에 대하여 잘 이해해 두는 것은 좋은 분류기를 설계하기 위한 핵심적인 지식이 된다.

이 장에서는 가장 기본적인 판별함수인 선형 판별함수의 특성과, 데이터를 이용하여 그 파라미터를 추정하는 방법에 대하여 알아볼 것이다. 선형 판별함수는 그 형태가 단순하여 표현할 수 있는 결정경계가 매우 제한적이나, 추정이 간단할 뿐 아니라 영상데이터와 같이 입력 차원이 매우 높은 경우에는 선형평면으로도 분류 가능한 경우가 많아 유용하게 쓰이고 있다. 또한 [식 6-5]의 판별함수도 기저함수가 이루는 공간상에서는 선형 판별함수로 볼 수 있어서, 일반적인 판별함수에 대한 논의의 기본이 된다. 이후 이 책의 후반부에서 선형 판별함수를 확장하여 비선형 결정경계를 표현하는 다양한 접근법에 대해 자세히 알아볼 것이다.

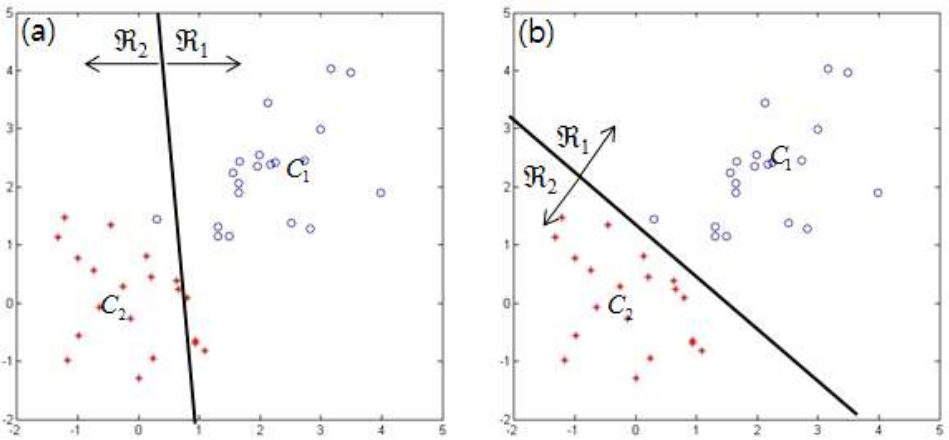
6.1.2 판별함수와 결정영역

선형 판별함수의 특성을 알아보기 위하여 판별함수에 의해 정해지는 결정영역에 대하여 살펴본다. 먼저 분류 대상이 되는 클래스가 두 개인 경우에 대하여 생각한다. 하나의 판별함수 $g(\boldsymbol{x})$를 이용하여 [식 6-1]과 같은 결정규칙으로 분류를 수행하는 경우, 결정경계는 $g(\boldsymbol{x})=0$가 되고, 클래스 C_1과 C_2에 대응되는 결정영역 $\Re_1$과 $\Re_2$는 각각 다음과 같이 정의된다.

$$\Re_1 = \{\boldsymbol{x} | g(\boldsymbol{x}) > 0\},\ \Re_2 = \{\boldsymbol{x} | g(\boldsymbol{x}) < 0\} \qquad \text{[식 6-7]}$$

[그림 6-2]에서는 두 개의 서로 다른 클래스 C_1, C_2에 속하는 데이터를 각각 기호 'o'와 '+'를 이용하여 표현하였다. 우리의 목적은 두 클래스를 잘 분류할 수 있는 판별함수 $g(\boldsymbol{x})$를 찾는 것으로, 즉 결정영역 $\Re_1$에는 클래스 C_1에 속하는 데이터가 모두 포함되고 $\Re_2$에는

클래스 C_2에 속하는 데이터가 모두 포함되는 판별함수가 바람직할 것이다. [그림 6-2a]는 임의로 정해진 결정경계가 두 클래스를 제대로 분류하지 못하고 있는 반면, [그림 6-2b]는 모든 데이터가 바르게 분류되는 결정경계이다.



[그림 6-2] 잘못된 결정경계(a)와 바른 결정경계(b)

한편, 세 개 이상의 클래스를 가지는 다중 클래스 분류 문제의 경우에는 결정경계가 다소 복잡해진다. 우선 각 클래스별로 판별함수 $g_i(\boldsymbol{x})$를 다음과 같은 특성을 가지도록 학습한 경우를 생각해 보자.

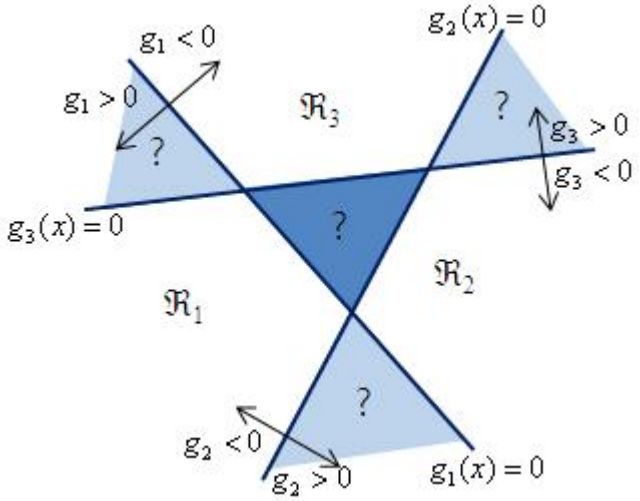
$$\begin{cases} g_i(\boldsymbol{x}) > 0 & \text{if} \quad \boldsymbol{x} \in C_i \\ g_i(\boldsymbol{x}) < 0 & \text{if} \quad \boldsymbol{x} \notin C_i \end{cases} \qquad \text{[식 6-8]}$$

이 경우 각 클래스에 대한 판별함수는 입력이 해당 클래스에 속하는지 여부만 판단하면 되기 때문에 일종의 이진 분류기가 되어 학습하기 쉬운 장점을 가진다. 이 판별함수를 이용한 결정규칙은 다음과 같이 정의해 볼 수 있다.

$$y(\boldsymbol{x}) = i \quad \text{if} \;\; g_i(\boldsymbol{x}) > 0 \;\text{and}\; g_j(\boldsymbol{x}) < 0 \;\text{for all}\; j \neq i \qquad \text{[식 6-9]}$$

이와 같이 각 클래스에 대해, 그에 속하는지 아닌지를 각각 판단함으로써 분류를 수행하는 방법을 <일대다 방식 (one-to-all method)>라고 한다. 그런데 이 결정규칙을 사용하여 전체 입력공간을 결정영역들로 나누면 [그림 6-3]과 같이 되어, [식 6-9]의 조건을 만족하지 못하는 부분이 존재한다. 어떤 영역의 경우는 두 개 이상의 판별함수가 양의 값을 가지게 되어 어디로 할당해야 할 지 선택이 모호한 영역이 되고, 또 다른 영역의 경우는 어떤 판별함수도 양의 값을 가지지 않아 어떤 클래스에도 속하지 않는 미결정영역이 된다. 그림의 중심부분에 짙은 음영으로 표시된 영역에 속하는 데이터가 주어진 경우, 모든 클래스에 대한 판별함수가 음의 값을 가지어 결국 어떤 클래스로도 할당할 수 없게 된다. 반대로, 그림의 왼쪽 윗부분에 음영으로 표시된 영역에 속하는 데이터가 주어진 경우에는 판별함수 $g_1(\boldsymbol{x})$와

$g_3(\boldsymbol{x})$가 동시에 양의 값을 가지게 되어, 이 역시 둘 중 어떤 클래스로 할당해야 할지 판단할 수 없게 된다.



[그림 6-3] 선형 판별함수와 미결정영역 (일대다방식)

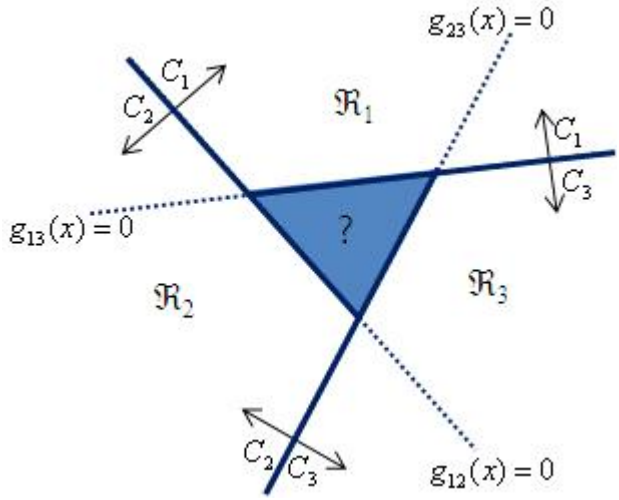
이러한 문제를 해결하기 위하여 임의의 두 클래스 쌍에 대하여 판별함수를 정의하고 학습하는 방법을 생각해 볼 수 있다. 즉, 모든 가능한 두 클래스 쌍 C_i ,C_j에 대해 판별함수 $g_{ij}(\boldsymbol{x})$, $g_{ji}(\boldsymbol{x})$를 정의하여 다음 조건이 만족하도록 학습한다.

$$\begin{cases} g_{ij}(\boldsymbol{x}) > 0 & \text{if } \boldsymbol{x} \in C_i \\ g_{ji}(\boldsymbol{x}) > 0 & \text{if } \boldsymbol{x} \in C_j \end{cases} \qquad \text{[식 6-10]}$$

이와 같은 접근법을 <일대일 방식 (one-to-one method)>이라고 한다. 이때 $g_{ij}(\boldsymbol{x})$, $g_{ji}(\boldsymbol{x})$는 그 식은 같고 부호만 반대인 $g_{ij}(\boldsymbol{x}) = -g_{ji}(\boldsymbol{x})$인 관계에 있음에 유의하자. 따라서 이 방법을 사용하는 경우, 분류 대상이 되는 클래스의 수가 m개일 때, 모두 $m(m-1)/2$개의 판별함수를 학습할 필요가 있고, 이는 [식 6-7]의 판별함수를 사용하는 경우보다 학습과정이 복잡해짐을 의미한다. 이 판별함수를 이용한 결정규칙은 다음과 같이 정의해 볼 수 있다.

$$y(\boldsymbol{x}) = i \quad \text{if } g_{ij}(\boldsymbol{x}) > 0 \text{ for all } j \neq i \qquad \text{[식 6-11]}$$

이 결정규칙에 의한 결정영역을 [그림 6-4]에 나타내었다. 그림에서와 같이 이 방법을 사용한 경우에는 선택이 모호해지는 영역은 발생하지 않으나 여전히 미결정영역이 존재한다. 그림의 중앙부분에 음영으로 나타낸 영역의 입력데이터에 대해서는 모든 판별함수에 대해서 음의 값이 주어지므로, 어떤 클래스로도 할당할 수 없다.



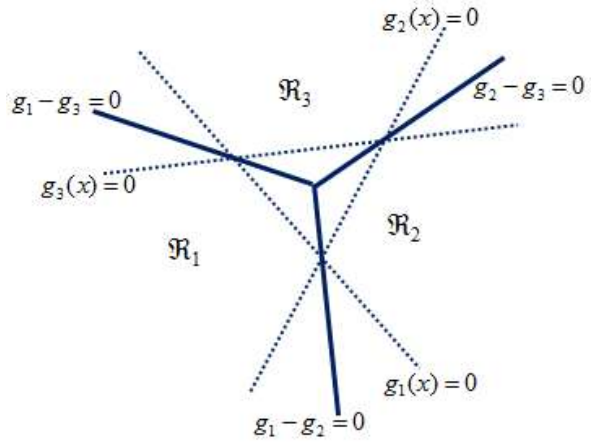
[그림 6-3] 선형 판별함수와 미결정영역 (일대일방식)

이와 같이 하나의 클래스로 명확하게 분류되지 않는 영역을 없애기 위해서는 [식 6-8]에서와 유사하게 클래스별 판별함수를 정의하되, 결정규칙으로 [식 6-2]를 사용하는 방법을 생각해 볼 수 있겠다. 이 방법에 의해 얻어지는 결정경계와 결정영역의 예를 [그림 6-5]에 나타내었다. 이 때 결정경계는 여러 개의 직선들의 조합으로 얻어지고, 각 직선은 $g_i(\boldsymbol{x})-g_j(\boldsymbol{x})=0$과 같은 형태로 정의된다. 이는 [식 6-9]의 결정규칙으로부터 얻어지는 결정경계([그림 6-3])와는 다른 형태이며, [그림 6-5]에서 확인할 수 있듯이 미결정영역은 존재하지 않음을 알 수 있다.

그런데, [식 6-2]와 같은 결정규칙이 제대로 작동하기 위해서는 각 클래스별 판별함수의 학습에 있어서 유의해야 할 점이 있다. 예를 들어, 클래스 C_i에 대한 판별함수 $g_i(\boldsymbol{x})$는 [식 6-8]의 조건 대신 다음 조건이 만족하도록 학습해야 한다.

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \text{ for all } \boldsymbol{x} \in C_i \quad j \neq i \qquad \text{[식 6-12]}$$

그런데 이 조건을 만족하도록 학습하기 위해서는 각 클래스에 대한 판별함수들이 동시에 학습되어야 한다. [식 6-8]이나 [식 6-10]의 판별함수는 각 경우에 대하여 음/양의 조건만 만족하면 되므로 각각 독립적으로 학습이 가능한 반면, [식 6-12]의 경우는 하나의 판별함수에 대한 출력값이 다른 판별함수의 출력값에 의존하는 형태를 가지기 때문이다. 4장에서 살펴본 베이즈 분류기의 경우에는 판별함수를 직접적으로 학습하는 대신 확률값으로부터 유도된 함수를 사용함으로써 이러한 어려움을 피할 수 있었다. 그러나 확률밀도함수가 아닌 일반적인 형태의 판별함수를 정의하여 직접 학습하는 식별적 접근법에서는 이 문제에 대한 근본적인 해결책을 얻기는 힘들다. 따라서 판별함수를 직접 학습하여 사용하는 경우에는 미결정영역에 대한 문제를 염두에 두어야 할 것이다.



[그림 6-5] 미결정영역이 없는 결정규칙의 예

6.2 판별함수의 학습

6.2.1 최소제곱법

각 클래스에 대한 선형 판별함수가 다음과 같이 정의되었다고 가정하자.

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} \qquad \text{[식 6-13]}$$

여기서는 식의 전개를 간단하게 하기 위하여 $\boldsymbol{w}_i$와 w_{i0}를 하나의 벡터로 합쳐서 $\boldsymbol{\omega}_i = [\boldsymbol{w}_i^T, w_{io}]^T$로 정의하고, 마찬가지로 입력 벡터에 대해서도 새로운 $n+1$ 차원의 벡터를 $\boldsymbol{\xi} = [\boldsymbol{x}^T, 1]^T$와 같이 정의하였다. 이 정의를 이용하면 [식 6-13]은 다음과 같이 간단한 형태로 표현된다.

$$g_i(\boldsymbol{\xi}) = \boldsymbol{\omega}_i^T\boldsymbol{\xi} \qquad \text{[식 6-14]}$$

이 절에서는 계산의 편의를 위해 이 표현을 사용하고, 따라서 학습 데이터 집합 $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$도 $\boldsymbol{\xi}$를 이용하여 $\Xi = \{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \cdots, \boldsymbol{\xi}_N\}$로 나타낸다.
판별함수 g_i가 원하는 결정경계를 만들기 위해서는 학습 데이터 집합을 이용하여 파라미터 벡터 $\boldsymbol{\omega}_i = [\boldsymbol{w}_i^T, w_{io}]^T$의 값을 찾아주어야 한다. 이를 위해 먼저 하나의 데이터 $\boldsymbol{\xi}_j$가 주어졌을 때, 판별함수 $g_i(\boldsymbol{\xi}_j)$의 목표 출력값을 t_j^i라 두고, 이 값을 정해주어야 한다. $g_i(\boldsymbol{\xi}_j)$는 클래스 C_i를 위한 판별함수이므로, 만약 데이터 $\boldsymbol{\xi}_j$가 클래스 C_i에 속한다면 목표 출력값은 $t_j^i = 1$으로 두고, 다른 클래스에 속한다면 목표 출력값은 $t_j^i = 0$으로 두면 될 것이다. 모든 데이터에 대하여 이와 같은 방식으로 목표 출력값을 정해주면, 학습 데이터 집합 $\Xi = \{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \cdots, \boldsymbol{\xi}_N\}$

전체에 대한 판별함수 g_i의 목표 출력값은 N차원 벡터 $\boldsymbol{t}^i = [t_1^i, t_2^i, \dots, t_N^i]^T$로 나타낼 수 있다. 이 목표 출력값을 이용하면 판별함수 g_i의 목적함수를 다음 식과 같이 정의할 수 있을 것이다.

$$J(\boldsymbol{\omega}_i) = \frac{1}{2}\sum_{j=1}^{N}(t_j^i - g_i(\boldsymbol{\xi}_j))^2 = \frac{1}{2}\sum_{j=1}^{N}(t_j^i - \boldsymbol{\omega}_i^T\boldsymbol{\xi}_j)^2 \qquad \text{[식 6-15]}$$

이 식의 의미는, 각각의 데이터 $\boldsymbol{\xi}_j$ $(j = 1,\dots,N)$에 대한 판별함수의 출력값 $g_i(\boldsymbol{\xi}_j)$와 목표 출력값 t_j^i의 차이의 제곱을 계산하여 이들을 모두 합한 값이 된다. 따라서 이 목적함수 $J(\boldsymbol{\omega}_i)$를 제곱오차함수라고 하며, 이 값을 최소로 하는 파라미터 $\boldsymbol{\omega}_i$를 찾는 것을 최소제곱법이라고 한다.
최소제곱법의 해는 목적함수 $J(\boldsymbol{\omega}_i)$를 파라미터 벡터 $\boldsymbol{\omega}_i$에 대해 미분하여 $\mathbf{0}$이 되는 점에서 주어지므로, 다음 방정식의 해를 구함으로써 얻을 수 있다.

$$\frac{\partial J(\boldsymbol{\omega}_i)}{\partial \boldsymbol{\omega}_i} = \sum_{j=1}^{N}(t_j^i - \boldsymbol{\omega}_i^T\boldsymbol{\xi}_j)\boldsymbol{\xi}_j^T = \sum_{j=1}^{N} t_j^i\boldsymbol{\xi}_j^T - \boldsymbol{\omega}_i^T\sum_{j=1}^{N}\boldsymbol{\xi}_j\boldsymbol{\xi}_j^T = \mathbf{0} \qquad \text{[식 6-16]}$$

여기서 $\boldsymbol{\xi}$에 대한 데이터 행렬을 이용하여 [식 6-16]을 다시 쓰면 다음과 같다.

$$(\boldsymbol{\Xi}\boldsymbol{\Xi}^T)\boldsymbol{\omega}_i = \boldsymbol{\Xi}\boldsymbol{t}^i \qquad \text{[식 6-17]}$$
$$\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \dots, \boldsymbol{\xi}_N] = \begin{bmatrix} \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N \\ 1,\ 1,\ \dots,\ 1 \end{bmatrix}$$

이 행렬식에 대한 해는 다음과 같이 간단히 얻을 수 있다.

$$\boldsymbol{\omega}_i = (\boldsymbol{\Xi}\boldsymbol{\Xi}^T)^{-1}\boldsymbol{\Xi}\,\boldsymbol{t}^i \qquad \text{[식 6-18]}$$

이 식을 이용하면 주어진 데이터 집합으로부터 한 번에 파라미터 값을 추정할 수 있다. 그러나 입력차원이 크고 데이터 수가 많은 경우 행렬 계산에 지나치게 많은 시간이 소모되어 실질적으로 사용하는 것은 불가능하다.
이러한 문제를 해결하기 위하여 순차적 반복 알고리즘을 이용할 수 있다. 즉, 한 번에 하나의 데이터 $\boldsymbol{\xi}$에 대해 제곱오차를 감소시키는 방향으로 파라미터를 반복적으로 수정해 가는 방법이다. 수정방향은 오차함수의 기울기를 따라서 오차값이 줄어드는 방향으로, [식 6-16]을 이용하면 다음과 같이 파라미터의 변화량 $\Delta\boldsymbol{\omega}_i$를 얻을 수 있다.

$$\Delta\boldsymbol{\omega}_i = \frac{\partial J(\boldsymbol{\omega}_i)}{\partial \boldsymbol{\omega}_i} = (t_j^i - \boldsymbol{\omega}_i^T\boldsymbol{\xi})\boldsymbol{\xi} \qquad \text{[식 6-19]}$$

각 데이터에 대하여 [식 6-19]를 계산 후 파라미터 $\boldsymbol{\omega}_i$를 수정하는 과정을 반복적으로 수행함으로써 [식 6-18]에서 얻어지는 해를 얻을 수 있다. τ번째 수정시의 파라미터가 $\boldsymbol{\omega}_i^{(\tau)}$라고 할 때 $\tau+1$번째의 새로운 파라미터를 계산하는 수정식은 다음과 같이 쓸 수 있다.

$$\boldsymbol{\omega}_i^{(\tau+1)} = \boldsymbol{\omega}_i^{(\tau)} - \eta \Delta \boldsymbol{\omega}_i^{(\tau)} \qquad \text{[식 6-20]}$$

여기서 η는 파라미터 값이 한 번에 크게 변하는 것을 방지하기 위해 곱해주는 작은 상수값으로, <학습률 (learning rate)>이라고 부른다. 이 파라미터 수정식에 의해 파라미터를 학습하는 방법을 <최소평균제곱 알고리즘 (Least Mean Square Algorithm, LMS algorithm)>이라고 한다.
최소평균제곱 알고리즘은 가장 기본적인 최적화 알고리즘 중의 하나로, 분류문제 뿐 아니라 함수 근사 및 회귀 (regression)문제에 일반적으로 적용할 수 있다. 함수 근사 문제의 측면에서 보면, 입력 $\boldsymbol{\xi}$와 원하는 출력 $\boldsymbol{t}$와의 입출력 매핑을 가장 잘 근사 하는 선형함수를 찾는 것으로 해석될 수 있다. 또한 이 알고리즘은 선형 판별함수를 일반화시킨 기저함수의 선형합에 의해 정의되는 판별함수 [식 6-5]에도 마찬가지로 적용될 수 있다. 그 경우에는 원래 입력 벡터 $\boldsymbol{x}$를 사용하는 대신 입력 벡터의 기저함수를 이용한 매핑값들로 이루어진 벡터 $\boldsymbol{\phi}(\boldsymbol{x})$를 사용함으로써 다음과 같이 파라미터 수정식을 얻을 수 있다.

$$\Delta \boldsymbol{\omega}_i = \frac{\partial J(\boldsymbol{\omega}_i)}{\partial \boldsymbol{\omega}_i} = \eta\,(t_j^i - \boldsymbol{\omega}_i^T \boldsymbol{\phi}(\boldsymbol{\xi}))\boldsymbol{\phi}(\boldsymbol{\xi}) \qquad \text{[식 6-21]}$$

6.2.2 퍼셉트론 학습

선형 판별함수의 또 다른 예로 Rosenblatt에 의해 개발된 패턴인식기인 퍼셉트론 (Perceptrons)이 있다. 퍼셉트론은 인간 뇌의 정보처리 방식을 모방하여 만든 인공신경회로망의 기초적인 모델이다. 신경회로망에 대해서는 11장에서 상세히 설명할 것이며, 이 장에서는 퍼셉트론에 대하여 선형 판별함수의 관점에서 살펴보겠다.
입력 $\boldsymbol{x}$에 대한 퍼셉트론의 i번째 출력 y_i는 다음과 같이 결정된다.

$$y_i = \mathrm{step}(\boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}) \qquad \text{[식 6-22]}$$

여기서 함수 $\mathrm{step}(u)$는 다음과 같은 출력을 내는 이진함수이다.

$$\mathrm{step}(u) = \begin{cases} 1 & \text{if } u > 0 \\ 0 & otherwise \end{cases} \qquad \text{[식 6-23]}$$

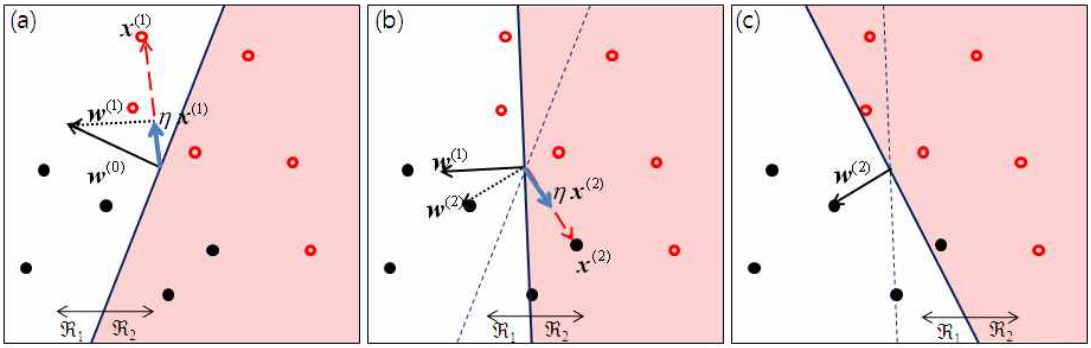
결국 퍼셉트론의 출력은 선형 판별함수를 이용한 결정규칙 [식 6-1]과 같은 역할을 수행함을 알 수 있다.
M개의 패턴을 분류하는 문제에 대해서는, 각 클래스별로 하나의 출력을 정의하여 모두 M개의 퍼셉트론 판별함수를 만든다. 이후 학습 데이터를 이용하여 원하는 출력을 낼 수 있도

록 파라미터를 조정해야 하는데, 이때 i번째 클래스에 속하는 입력이 들어온 경우 이에 해당하는 출력 y_i만 1의 값을 갖고, 나머지 출력들은 모두 0의 값을 가질 수 있도록 목표 출력값 t_i $(i=1,...,M)$을 정의해 둔다.

τ번째 학습에서 하나의 입력 $\boldsymbol{x}^{(\tau)}=[x_1, x_2, \dots, x_n]^T$와 목표 출력 $\boldsymbol{t}^{(\tau)}=[t_1, t_2, \dots, t_M]^T$가 주어지면, 출력 y_i를 결정하는 파라미터 $\boldsymbol{w}_i^{(\tau)}$ = $[w_{i1}, w_{i2}, \dots, w_{in}]^T$와 $w_{i0}^{(\tau)}$은 다음과 같은 식에 의하여 파라미터를 수정한다.

$$\boldsymbol{w}_i^{(\tau+1)} = \boldsymbol{w}_i^{(\tau)} + \eta(t_i - y_i)\boldsymbol{x} \qquad \text{[식 6-24]}$$
$$w_{i0}^{(\tau+1)} = w_{i0}^{(\tau)} + \eta(t_i - y_i)$$

여기서 η는 가중치 수정이 한 번에 많이 일어나서 학습이 불안정해지는 것을 방지하기 위한 작은 상수값으로, [식 6-20]의 경우와 마찬가지로 학습률이라고 부른다. 이 식을 이용한 퍼셉트론의 학습 과정을 [그림 6-5]에 나타내었다. 퍼셉트론은 기본적으로 하나의 데이터가 주어질 때마다 그에 맞추어 파라미터를 수정하는 온라인 학습을 수행한다. 그림에서 살펴보면, 각 시점에서 주어진 학습 데이터에 대해 잘못된 분류가 일어날 때 마다 그것을 수정할 수 있는 방향으로 조금씩 파라미터를 수정함으로써 결정경계가 변화됨을 알 수 있다. 이 때 파라미터의 수정은, 현재 주어진 입력 $\boldsymbol{x}^{(\tau)}$에 작은 상수값을 곱한 벡터를 현재 가지고 있는 파라미터 $\boldsymbol{w}_i^{(\tau)}$에 더해주거나 빼주는 방향으로 진행된다. [그림 6-5(a)]에서 파선으로 나타난 것이 현재 주어진 입력벡터로, 이것에 작은 상수값 (η)가 곱해진 것이 굵은 실선 벡터로 표현되어 있다. 데이터는 두 개의 클래스로 나뉘어져 있는데, 심벌 •로 표시된 데이터가 C_1에 속하는 것으로, 목표출력값 t는 1이다. 반대로 심벌 ◦로 표시된 데이터는 C_2에 속하는 것으로, 목표출력값 t는 0이다. (a)에서 현재의 파라미터 $\boldsymbol{w}^{(0)}$에 의해 결정되는 결정영역 $\Re_1$(출력값 y가 1이 되는 영역)과 $\Re_2$(출력값 y가 0이 되는 영역)가 나타나 있다. 이때 주어진 데이터 $\boldsymbol{x}^{(1)}$은 목표출력값 t가 0인 데이터이지만, 현재의 결정경계에 의해 출력값 y는 1인 상태이다. 따라서 [식 6-24]에 의해 계산되는 새로운 파라미터 $\boldsymbol{w}^{(1)}$는 $\boldsymbol{w}^{(0)}$에 수정량 $\eta(t_i - y_i)\boldsymbol{x}^{(1)} = -\eta\boldsymbol{x}^{(1)}$ 가 더해져서 얻어진다. 이렇게 얻어진 새로운 $\boldsymbol{w}^{(1)}$에 의해 만들어지는 결정경계와 결정영역은 그림(b)에 나타나 있다. 이어서 그림 (b)에서는 새로운 데이터 $\boldsymbol{x}^{(2)}$가 주어졌을 때의 가중치 수정과정에 대해 보여주고 있다. 이번에는 목표출력값 t가 1이지만 퍼셉트론의 출력값 y는 0인 경우로, 새로운 $\boldsymbol{w}^{(2)}$는 $\boldsymbol{w}^{(1)}$에 $\eta\boldsymbol{x}^{(2)}$ 가 더해져서 얻어진다. 이렇게 얻어진 $\boldsymbol{w}^{(2)}$에 의해 결정되는 결정경계가 그림 (c)에 나타나 있다. 그림 (c)의 결정경계는 모든 학습데이터에 대해 목표출력값과 퍼셉트론의 출력값이 같아져서 더 이상 파라미터의 수정은 일어나지 않는 상태가 된다. 이와 같이 학습데이터 들에 대한 반복적 파라미터의 수정에 의해 원하는 결정경계를 찾아가는 과정이 퍼셉트론의 학습 과정이 된다.



[그림 6-5] 퍼셉트론의 학습에 의한 결정경계의 변화

전체 학습 데이터에 대한 퍼셉트론 학습 단계를 정리하면 다음과 같다.

[퍼셉트론 학습 단계]

① 임의의 값으로 가중치를 초기화한다.

② 하나의 입력 데이터 $\boldsymbol{x}$에 대해 출력값 $\boldsymbol{y}$를 계산

③ 목표 출력값 $\boldsymbol{t}$와 퍼셉트론의 출력값 $\boldsymbol{y}$의 차를 계산

④ 퍼셉트론 학습 규칙을 이용하여 가중치 수정

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \eta(t_i - y_i)x_j \qquad (j = 1, ..., N,\ i = 1, ..., M)$$

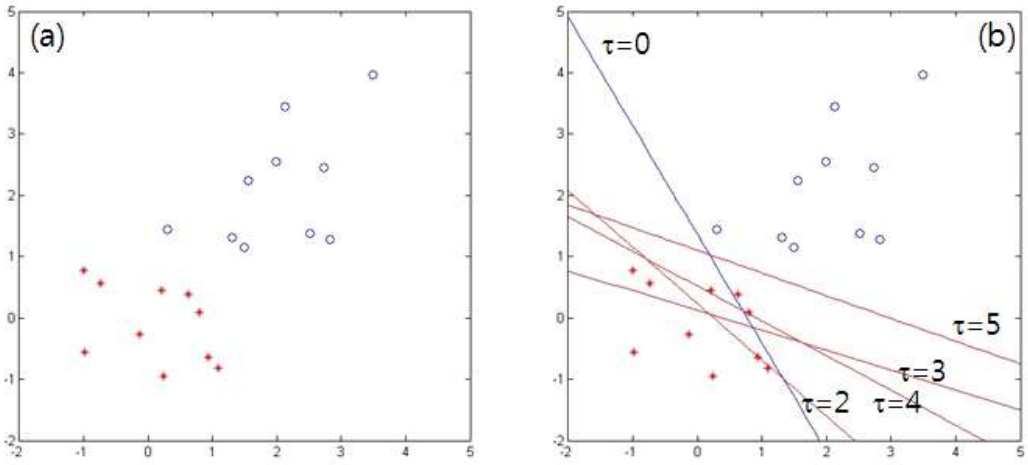
⑤ ②~④ 과정을 모든 입력 패턴에 대해서 반복

⑥ $\boldsymbol{t}$와 $\boldsymbol{y}$의 차의 제곱으로 정의되는 오차가 원하는 값으로 줄어들거나 미리 정해둔 반복 횟수까지 ②~⑤ 반복

퍼셉트론 학습은 선형 판별함수의 학습을 위해 가장 기본적으로 사용하는 방법으로, 원하는 출력 결과를 얻을 때까지 반복적으로 파라미터의 값을 수정해 간다. 그런데 앞서 언급한 것처럼 선형 초평면으로 데이터를 분류하는 것에는 한계가 있으므로, 오차가 항상 0에 가까워진다고 보장할 수는 없다. 이러한 문제를 해결하기 위해서는 [식 6-5]와 같이 비선형 기저함수를 이용해 볼 수 있다. 또한 퍼셉트론 판별함수는 6.1절에서 설명한 미결정영역에 대한 문제도 존재한다. 즉, 학습 데이터에 대해서 완전히 학습이 이루어진 경우에도, 새롭게 주어지는 데이터에 대해서는 2개 이상의 $y_i(i = 1, ..., M)$가 1의 값을 내거나 모든 출력이 0이 되는 경우도 발생하게 된다. 이런 출력에 대하여 어떻게 처리할 것인지에 대해서도 고려해 주어야 할 것이다.

6.3 매트랩에 의한 선형 판별함수 분류기 실험

앞 절에서 살펴본 선형 판별함수의 최소제곱오차 학습법과 퍼셉트론 학습법은 그 정의 방식에 차이가 있을 뿐 기본적인 학습 방식은 동일하다. 여기서는 퍼셉트론 학습법을 이용하여 간단한 분류 실험을 수행해 보겠다.



[그림 6-7] 학습에 사용된 데이터와 퍼셉트론의 학습에 의한 결정경계의 변화

[그림 6-7a]에 학습에 사용된 데이터 집합이 나타나 있다. 각 클래스별로 10개 씩 모두 20개의 데이터에 대해 두 개의 클래스로 분류하는 이분류 문제를 수행하는 퍼셉트론을 정의하였다. 퍼셉트론의 학습 알고리즘은 [프로그램 6-1]에 나타나 있다.

먼저 학습 데이터를 가우시안 분포로부터 생성한 후, 퍼셉트론의 입출력 차원 수와 파라미터의 초기값, 그리고 학습률을 설정한다. 또한 학습을 반복할 횟수를 미리 정하여 어느 정도 학습이 수행된 후에 종료될 수 있도록 설정하였다. 학습을 시작하기 전에 $\tau = 1$일 때 초기화된 파라미터에 의해 결정된 결정경계를 [그림 6-7b]에 나타내었다. 그림에서 알 수 있듯이 오분류된 데이터들이 존재한다. 이후 학습을 반복해 감에 따라 변화되는 결정경계를 그림에 나타내었으며 5번 학습을 수행한 후 $\tau = 5$일 때 찾아진 결정경계는 모든 데이터를 잘 분류하고 있음을 알 수 있다.

현재 주어진 데이터의 경우, 선형분리가 가능한 형태이므로 퍼셉트론은 5번 학습한 후에 계속 학습을 반복하여도 더 이상 파라미터 값이 변하지 않게 된다. 그러나 만약 주어진 문제가 선형분리가 불가능한 문제라면 퍼셉트론 학습에서는 계속해서 결정경계를 움직여 가면서 학습을 수행해 갈 것이다. 이러한 문제를 방지하기 위하여 최대 반복횟수를 미리 정해 둔 것이다. 또한 앞서도 설명한 바와 같이 퍼셉트론은 선형 결정경계를 가지므로, 선형으로 분리가 불가능한 문제에서는 좋은 성능을 얻기 힘들다. 이러한 문제를 해결하기 위해 비선형 결정경계를 학습하는 방법에 대하여서는 12장에서 살펴볼 것이다.

프로그램 6-1 Perceptron Classifier
퍼셉트론을 이용한 이분류 문제

```
001 % 학습데이터의 생성 ------------------------------------
002 NumD=10;
003 train_c1 = randn(NumD,2);
004 train_c2 = randn(NumD,2)+repmat([2.5,2.5],NumD,1);
005 train=[train_c1; train_c2];
006 train_out(1:NumD,1)=zeros(NumD,1);
007 train_out(1+NumD:2*NumD,1)=zeros(NumD,1)+1;
008 plot(train(1:NumD,1), train(1:NumD,2),'r*'); hold on
009 plot(train(1+NumD:2*NumD,1), train(1+NumD:NumD*2,2),'o');
010
011 % 퍼셉트론 학습 시작 ------------------------------------
012 Mstep=5;                                  %최대 반복횟수 설정
013 INP=2; OUT=1;                             %입출력 차원 설정
014 w=rand(INP,1)*0.4-0.2;  wo=rand(1)*0.4-0.2;   %파라미터 초기화
015 a=[-3:0.1:6];                                  %결정경계의 출력
016 plot(a, (-w(1,1)*a-wo)/w(2,1));
017 eta=0.5;                                      %학습률 설정
018 for j = 2:Mstep              %모든 학습 데이터에 대해 반복횟수만큼 학습 반복
019   for i=1:NumD*2
020     x=train(i,:);  ry=train_out(i,1);
021     if (x*w+wo>0) y=1; else y=0; end;          %퍼셉트론 출력 계산
022     e=ry-y;                                    %오차 계산
023     E(i,1)=e;
024     dw= eta*e*x';                              %파라미터 수정량 계산
025     dwo= eta*e*1;
026     w=w+dw;                                    %파라미터 수정
027     wo=wo+dwo;
028   end
029 plot(a, (-w(1,1)*a-wo)/w(2,1),'r');            %변화된 결정경계 출력
030 end
```

연습문제

1. 다음 순서에 따라 OR 함수를 학습하는 퍼셉트론을 구현하시오.

(1) 네 개의 학습 데이터를 다음과 같이 만드시오.

데이터 1: 입력 (0,0) 출력 (0)
데이터 2: 입력 (1,0) 출력 (1)
데이터 3: 입력 (0,1) 출력 (1)
데이터 4: 입력 (1,1) 출력 (1)

(2) 입력노드 2개와 출력 노드 1개로 구성된 다층 퍼셉트론을 만들고, 위에서 생성한 4개의 데이터를 이용하여 학습을 수행하시오.

(3) 학습이 진행되는 과정에서, 4개의 데이터가 한 번씩 학습에 사용되고 난 뒤 신경회로망의 출력오차를 계산하여 그 오차가 줄어드는 과정을 그래프로 나타내시오.

(4) 학습이 완료된 후, 다음 데이터에 대한 출력값을 계산해 보시오.

데이터 1: 입력 (0.1, 0.1)
데이터 2: 입력 (0.9, 0.9)
데이터 3: 입력 (0.1, 0.9)
데이터 4: 입력 (0.9, 0.1)

(5) 학습이 완료된 후 찾아진 결정경계를 2차원 평면상에 나타내 보시오.

2. 1번과 같이 OR 문제를 해결하는 결정경계를 최소제곱법에 의해 찾기 위해 다음과 같이 입출력을 정의한다.

데이터 1: 입력 (0,0) 출력 (0)
데이터 2: 입력 (1,0) 출력 (1)
데이터 3: 입력 (0,1) 출력 (1)
데이터 4: 입력 (1,1) 출력 (1)

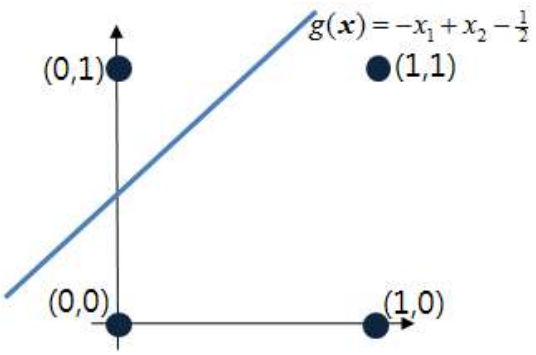
이 데이터를 이용하여 [식 6-18]에 의해 찾아지는 결정경계를 계산해 보고, 1번에서 퍼셉트론에 의해 찾아진 해와 비교해 보시오.

3. XOR 문제에 대하여 1번과 같은 과정으로 퍼셉트론 학습을 수행해 보시오.

4. 2. 다음과 같이 4개의 입력데이터와 목표 출력값이 각각 주어졌다.

$\boldsymbol{x}_1$: (0,0) t_1: (0), $\boldsymbol{x}_2$: (1,0) t_2: (0)
$\boldsymbol{x}_3$: (0,1) t_3: (0), $\boldsymbol{x}_4$: (1,1) t_4: (1)

이 데이터를 퍼셉트론을 이용하여 분류하고자 한다. 파라미터의 초기치가 오른쪽 그림의 직선을 나타내고 있을 때, 4개의 데이터에 대해 순서대로 퍼셉트론 학습을 적용한 후 얻어지는 판별함수를 찾으시오. (단, 학습률은 $\eta = 0.5$로 둔다.) (20점)



참고 자료

이 장에서 살펴본 퍼셉트론은 선형판별 분류기의 가장 대표적인 예로, 11장에서 살펴볼 신경망의 기초적인 모델이기도 하다. 이 장에서는 선형판별함수를 학습하는 방법으로 퍼셉트론과 그 학습에 대해 알아보았으나, 11장에서는 이와는 조금 다른 개념으로 인간의 뇌신경회로망을 모방하는 기초 모델로써 퍼셉트론에 대하여 다시 살펴볼 것이다. 이와 관련된 내용은 신경망에 대해 잘 소개하고 있는 [Haykin 99]와 [Bishop 96]을 참조할 수 있을 것이다. 또 다른 학습법인 최소평균제곱 (LMS) 알고리즘은 신호처리분야에서 적응적 필터 (adaptive filter)로 개발된 것으로 이와 관련된 내용은 [Haykin 02]에서 찾아볼 수 있다. 이밖에 선형분류기에 대한 기본적인 내용은 1장에서 소개한 [Duda & Hart & Stork 01]과 [Alpaydin 04]에 잘 나와 있다.

[Haykin 99] S. Haykin. Neural Networks: a Comprehensive Foundation. Prentice Hall, 1999.

[Bishop 96] C. M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 1995.

[Haykin 02] S. Haykin. Adaptive Filter Theory. Prentice Hall, 2002