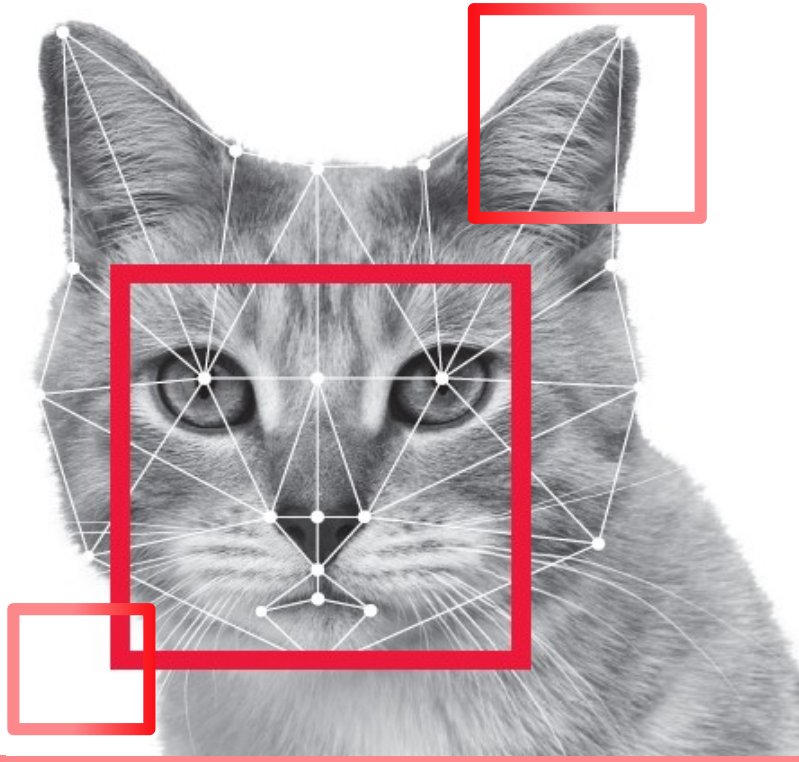
한빛아카데미
Hanbit Academy, Inc.

COMPUTER VISION 컴퓨터 비전

기본 개념부터 최신 모바일 응용 예까지



8장. 기계 학습

PREVIEW

학습

- 사람의 가장 뛰어난 능력 (외부 환경이 사람을 지도)

그림 8-1 사람의 학습 과정

기계 학습

- 어떤 목적에 활용하나?
- 어떤 학습 모델이 있으며, 어떻게 학습할까?

PREVIEW

- 기계 학습이 현대 컴퓨터 비전에서 차지하는 비중은 나날이 커지고 있다.
  - 컴퓨터 비전의 발전을 견인하는 중요한 축 중의 하나가 기계 학습이라고 할 수 있다. 기존의 알고리즘은 사람이 수작업으로 매개변수를 설정하거나 규칙을 개발하여 일 일이 프로그램에 삽입해야 한다. 반면 기계 학습은 학습 집합을 보고 필요한 최적의 설정을 자동으로 수행해 준다. 시스템이 다루어야 할 환경이 바뀌면 새로운 환경에서 수집한 학습 집합으로 다시 학습하면 된다. 게다가 성능 실험에 따르면 이전에 주로 사용하던 수작업 방식에 비해 성능이 뛰어나다. 이 장에서 소개하는 기계 학습 방법은 모두 뛰어난 성능이 검증된 것들로서 많은 응용 문제에 널리 활용된다.

- 기계 학습은 어떤 형태를 띨까?
  - 사람은 외부 환경의 지도를 받으면 성능을 개선하는 방향으로 자신의 기억을 조금씩 바꾸어 간다. 기계도 비슷한 원리를 따를까? 그렇다면 외부 환경은 무엇이고, 지도는 어떻게 이루어지며, 기억은 무엇이고, 성능이 개선되는 방향은 어느 쪽일까?

각 절에서 다루는 내용

1. 기계 학습의 기초

   → 기계 학습의 기초 원리에 대해 설명한다.

2. 신경망

   → 퍼셉트론과 다층 퍼셉트론을 소개하고 최근 주목을 받는 깊은 학습도 설명한다.

3. SVM

   → 여백을 최대화하는 원리를 사용하는 SVM에 대해 설명한다.

4. 분류기 앙상블

   → 다수의 단순한 분류기를 결합하는 원리를 이용하는 에이더부스트와 임의 숲을 공부한다.

5. 기계 학습을 이용한 얼굴 검출

   → 비올라-존스 얼굴 검출기를 포함하여 기계 학습으로 응용 문제를 해결한 사례를 소개한다.

8.1 기계 학습의 기초

8.1.1 지도 학습과 비지도 학습

8.1.2 재 샘플링을 이용한 성능 평가

8.1 기계 학습의 기초

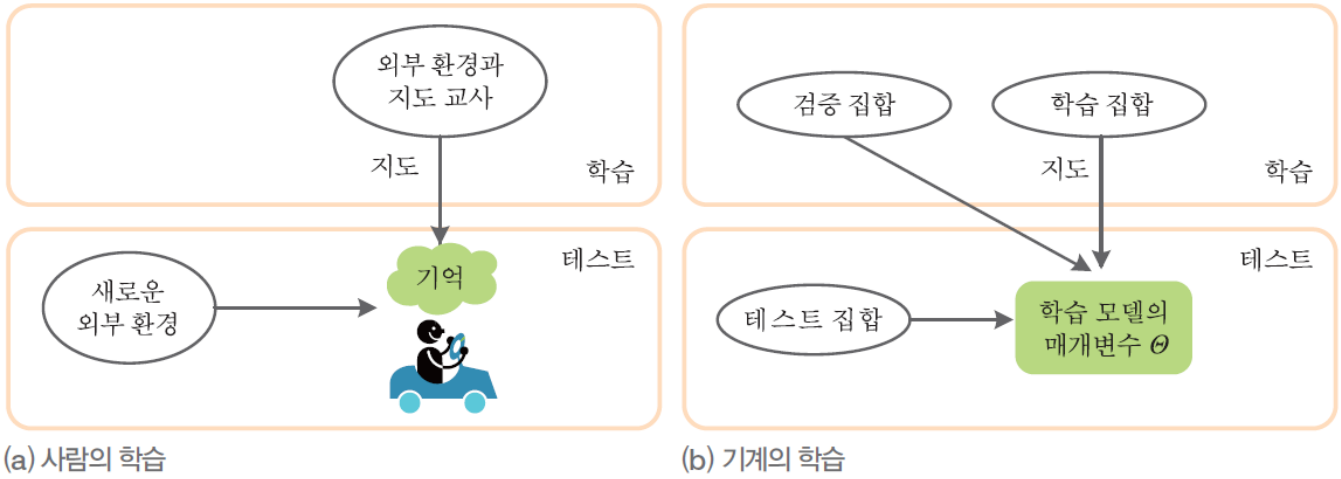
- Mitchell의 정의
  - "A computer program is said to learn from experience E with respect to some class of tasks T andperformance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

→ 어떤 컴퓨터 프로그램이 T라는 작업을 수행한다. 이 프로그램의 성능을 P라는 척도로 평가했을 때 경험 E를 통해 성능이 개선된다면 이 프로그램은 학습을 한다고 말할 수 있다[Mitchell97, 2쪽].

  - 컴퓨터 비전에 대입하면
    - *T*=분류(인식)
    - *E*=학습 집합
    - *P*=인식률

8.1.1 지도 학습과 비지도 학습

- 사람과 기계의 학습 과정
  - 학습 단계: 외부 지도에 따라 배우는 과정 (학습 집합 사용)
  - 테스트 단계: 현장에서 성능을 평가 하는 과정 (테스트 집합 사용)



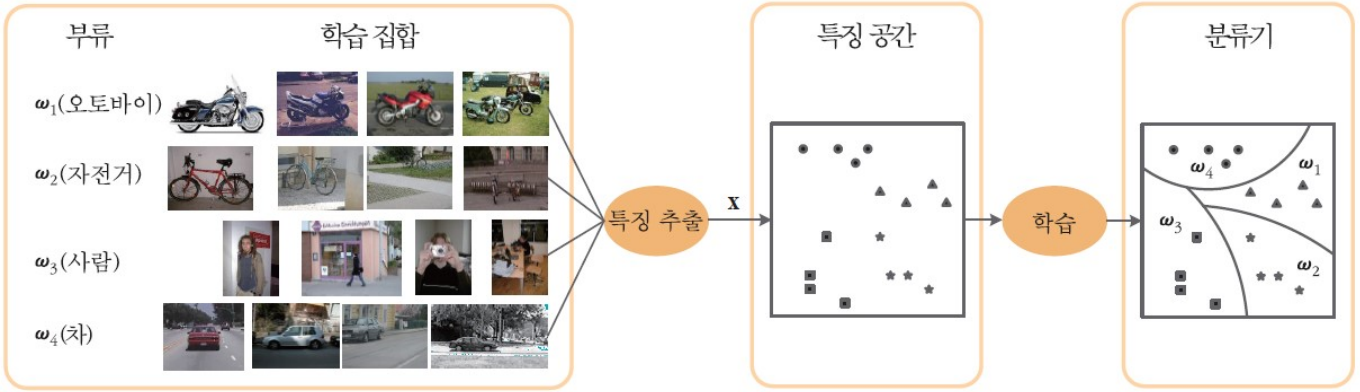
(a) 사람의 학습 (b) 기계의 학습

그림 8-2 사람과 기계의 학습 과정 비교

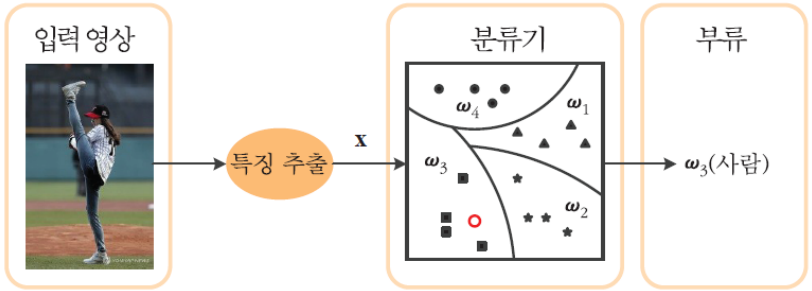
8.1.1 지도 학습과 비지도 학습

지도 학습

- 부류 정보를 가진 샘플 $(\mathbf{x}, t)$로 구성된 학습 집합 사용
  - $\mathbf{x}$는 특징 벡터이고 t는 $\mathbf{x}$가 속한 부류



(a) 학습 단계



(b) 테스트(인식) 단계

그림 8-3 컴퓨터 비전에서 기계 학습

8.1.1 지도 학습과 비지도 학습

- 학습 모델
  - 학습 모델은 매개변수 집합 Θ로 표현됨 (예, 그림 8-3에서 Θ는 분할 경계선을 표현)
  - 모델 선택이 필요하면 추가적으로 검증 집합 사용
  - 학습의 목표는 최적의 Θ 값을 찾는 것
    - Θ 값이 얼마나 좋은지 측정하는 목적 함수 필요

- 학습 알고리즘의 원리
  - 학습 집합을 사용하여 현재 Θ 값을 평가한 후, 개선하는 방향을 알아내 갱신
  - 이런 과정을 수렴할 때까지 반복

8.1.1 지도 학습과 비지도 학습

- 특징 추출은 어떻게 할까?

(a) 영역 (b) 이동 윈도우sliding window (c) 관심점

그림 8-4 기계 학습에 사용할 특징 벡터를 수집하는 여러 가지 시나리오

8.1.1 지도 학습과 비지도 학습

- 일반화 능력
  - 학습 과정에서 사용하지 않은 테스트 집합으로 학습이 완료된 분류기 성능을 평가
  - 학습 집합과 테스트 집합에 대한 성능이 비슷하면 일반화 능력이 뛰어남
  - 과적합 하면 일반화 능력이 떨어짐

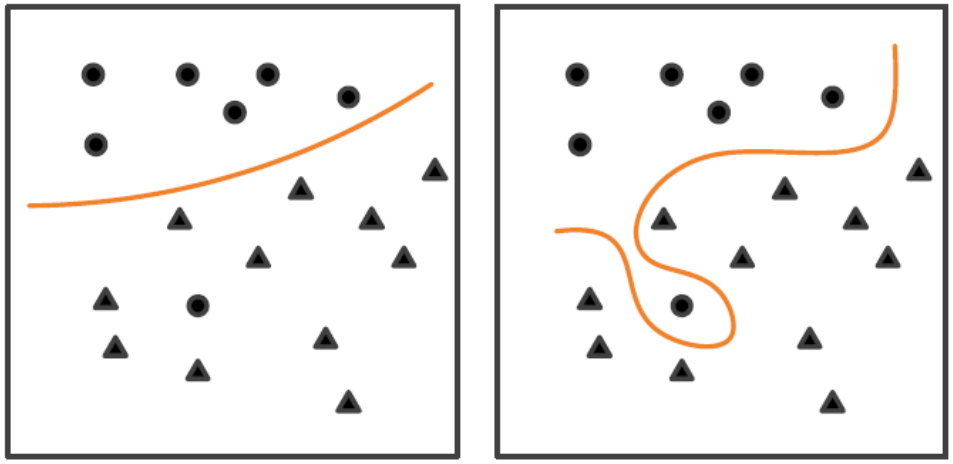


그림 8–5 오른쪽은 과적합 상황

8.1.1 지도 학습과 비지도 학습

- 지도 학습에서 사용했던 $(\mathbf{x}, t)$ 중에 부류 정보 t가 없는 상황의 학습

- 비지도 학습
  - 유사한 특징 벡터들을 끼리끼리 모으는 군집화 수행 (K-means, SOM 신경망, 민시프트 등의 군집화 알고리즘)
  - 군집에서 유용한 정보 추출 (데이터 마이닝, 빅데이터, 정보 검색 등의 많은 응용)

- 준지도 학습
  - 부류 정보가 있는 샘플과 없는 샘플이 섞여 있는 상황의 학습
  - 최근 중요성 부각
    - 아직 부류 정보를 부여하지 못한 샘플이 시시각각 인터넷에서 발생하기 때문
  - 원리
    - 부류 정보가 있는 샘플로 학습한 후 부류 정보가 없는 샘플의 부류 정보를 추정
    - 추정된 정보로 반복 학습

8.1.2 재 샘플링을 이용한 성능 평가

- 학습에 사용할 데이터베이스가 작은 경우,
  - 같은 샘플을 여러 번 사용하여 성능 측정의 통계적 신뢰도를 높이는 아이디어

- *k*-겹 교차 검증
  - 샘플 집합을 *k*개의 부분 집합으로 등분
  - *k*-1개를 학습에 사용하고 나머지 하나를 테스트에 쓰는 과정을 *k* 번 반복

알고리즘 8-1 교차 검증

입력 : $k(2 \le k \le N)$, 훈련 집합 $X=\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$

출력 : 성능 q

```
1  X를 k개의 부분 집합으로 등분하고 그들을 X1, X2, ···, Xk라 한다.
2  for(i=1 to k) {
3     X' = ∪(j=1,j≠i)^k Xj로 분류기를 학습시킨다.
4     Xi로 분류기의 성능을 측정하여 qi라 한다.
5  }
6  q=(q1+···+qk)/k;
```

8.1.2 재 샘플링을 이용한 성능 평가

붓스트랩

- 훈련 집합의 샘플링과 성능 측정을 여러 번 반복한 후, 평균 성능을 구함

알고리즘 8-2 붓스트랩

입력 : 훈련 집합 $X=\{(\mathbf{x}_1, t_1),(\mathbf{x}_2, t_2),\cdots,(\mathbf{x}_N, t_N)\}$, 샘플링 비율 $p(0<p\leq 1)$, 반복 횟수 T

출력 : 성능 q

```
1  for(t=1 to T) {
2    X에서 임의로 pN개의 샘플을 뽑아 X_t라 한다. 이때 대치를 허용한다.
3    X_t로 분류기를 학습시킨다.
4    X-X_t로 분류기의 성능을 측정하여 q_t라 한다.
5  }
6  q=(q_1+…+q_T)/T;
```

8.2 신경망

8.2.1 구조와 작동 원리

8.2.2 학습

8.2.3 깊은 학습

8.2 신경망

- 신경망은 연결주의 계산 모형
  - 방대하게 연결된 많은 뉴런으로 구성된 뇌 구조를 모방한 계산 모형
  - 1950년대 Rosenblatt의 퍼셉트론
  - 1980년대 퍼셉트론을 확장한 다층 퍼셉트론(MLP)
  - 일반화 능력이 뛰어남

8.2.1 구조와 작동 원리

퍼셉트론

- 입력은 특징 벡터 $\mathbf{x}=(x_1,x_2,\ldots,x_d)$
- $\mathbf{x}$를 두 개의 부류 ω_1과 ω_2 중의 하나로 분류하는 이진 분류기

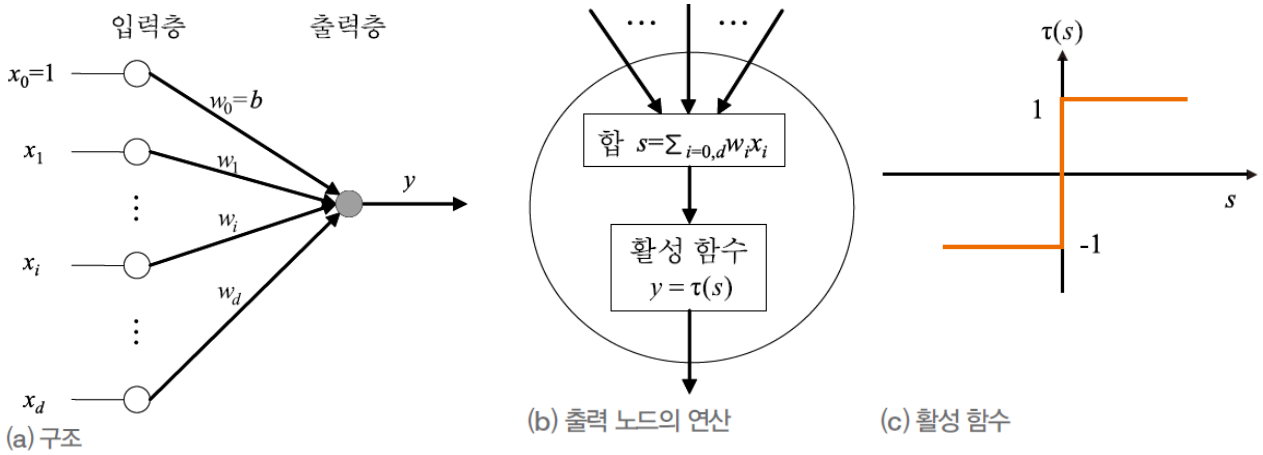


그림 8-6 퍼셉트론의 구조

8.2.1 구조와 작동 원리

- 수식으로 쓰면,
  - 출력 노드는 가중치 합과 활성 함수 계산
    - 특징 벡터 $\mathbf{x}=(x_1,x_2,...,x_d)$, 가중치 벡터 $\mathbf{w}=(w_1,w_2,...,w_d)$로 표기하면

$$y = \tau(s) = \tau\left(\sum_{i=1}^{d} w_i x_i + b\right) = \tau(\mathbf{w}\mathbf{x}^{\mathrm{T}} + b) \quad (8.1)$$

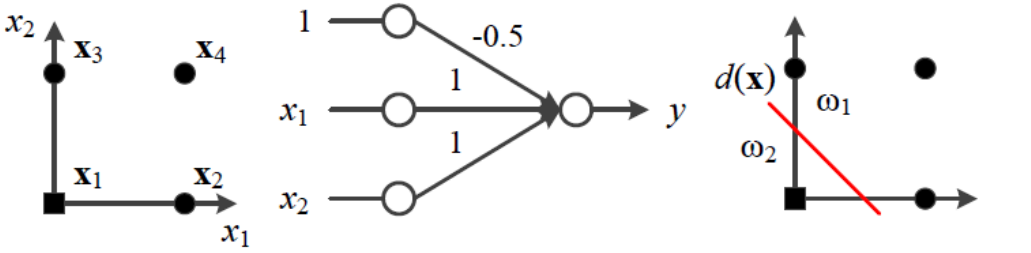
$$\text{이때 } \tau(s) = \begin{cases} +1, & s \geq 0 \\ -1, & s < 0 \end{cases}$$

  - 계단 함수를 활성 함수로 사용
    - 출력은 +1(ω_1에 해당) 또는 -1(ω_2에 해당) → 이진 분류기

8.2.1 구조와 작동 원리

예제 8-1 **퍼셉트론의 동작 과정**

2차원 공간 상에 [그림 8-7(a)]와 같이 네 개의 샘플 $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$가 존재하며 $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$는 ω_1에 속하고 $\mathbf{x}_1$은 ω_2에 속한다고 가정하자. 샘플과 부류를 값으로 표현하면 $\mathbf{x}_1=(0,0)$, $t_1=-1$, $\mathbf{x}_2=(1,0)$, $t_2=1$, $\mathbf{x}_3=(0,1)$, $t_3=1$, $\mathbf{x}_4=(1,1)$, $t_4=1$이다. 이때 t_i는 $\mathbf{x}_i$가 속하는 부류 정보로서, ω_1에 속하면 1이고 ω_2에 속하면 −1이다.



(a) OR 분류 (b) 퍼셉트론 (c) 퍼셉트론은 선형 분류기

그림 8-7 퍼셉트론의 예

이 예제는 OR 게이트의 동작을 분류 문제로 간주한다. [그림 8-7(b)]는 이 분류 문제를 해결하는 퍼셉트론을 보여준다. 이 퍼셉트론은 $\mathbf{w}=(1,1)$, $b=-0.5$를 가진다고 보면 된다. 네 개 샘플 중에 $\mathbf{x}_3=(0,1)$을 입력해 보자.

$$y=\tau(\mathbf{w}\mathbf{x}_3^{\mathrm{T}}+b)=\tau\left((1\quad 1)\begin{pmatrix}0\\1\end{pmatrix}-0.5\right)=\tau(0.5)=1$$

출력으로 1을 얻었으므로 원하는 값 t_3와 같다. 따라서 샘플 $\mathbf{x}_3$를 옳게 인식했다고 말할 수 있다. 나머지 샘플 세 개도 연필을 들고 계산하여 제대로 인식하는지 확인해 보자. 이 퍼셉트론은 [그림 8-7(c)]의 결정 직선에 해당하며 수식으로 표현하면 다음과 같다.

$$d(\mathbf{x})=w_1x_1+w_2x_2+b=x_1+x_2-0.5=0$$

8.2.1 구조와 작동 원리

- 지금까지는,
  - 학습이 완료된(가중치가 결정되어 있는) 퍼셉트론을 가지고 동작을 살펴봄
  - 그림 8-3에서 학습 단계가 아니라, 테스트 단계를 해본 셈

- 학습을 어떻게 할 것인가?
  - 예제 8-1의 경우, 주어진 네 개의 샘플을 가지고 가중치 벡터의 값을 어떻게 알아낼 것인가?

  → 학습에 대한 공부는 2.2절

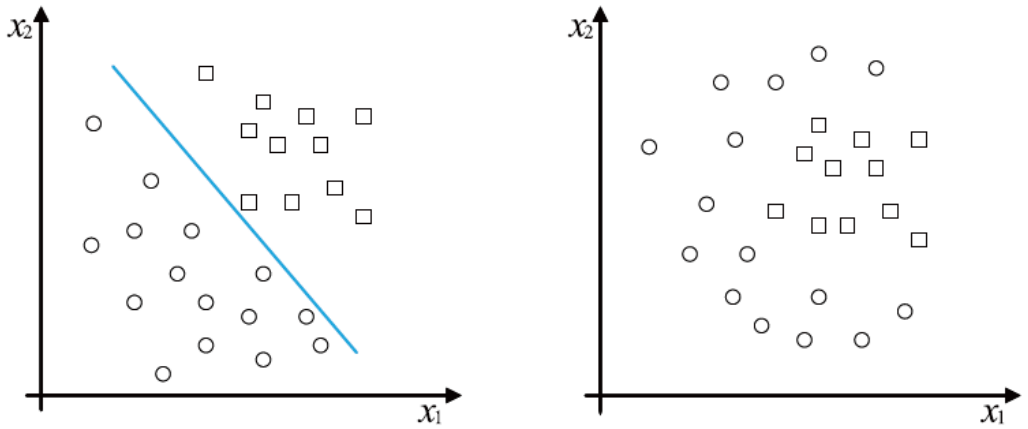
8.2.1 구조와 작동 원리

퍼셉트론의 한계

- 퍼셉트론은 식 (8.2)의 결정 초평면 역할을 하는 선형 분류기

$$d(\mathbf{x}) = \mathbf{w}\mathbf{x}^{\mathrm{T}} + b \geq 0 \text{이면 } \mathbf{x} \in \omega_1$$
$$d(\mathbf{x}) = \mathbf{w}\mathbf{x}^{\mathrm{T}} + b < 0 \text{이면 } \mathbf{x} \in \omega_2 \qquad (8.2)$$

- 선형 분리 불가능한 상황에 대처하지 못함



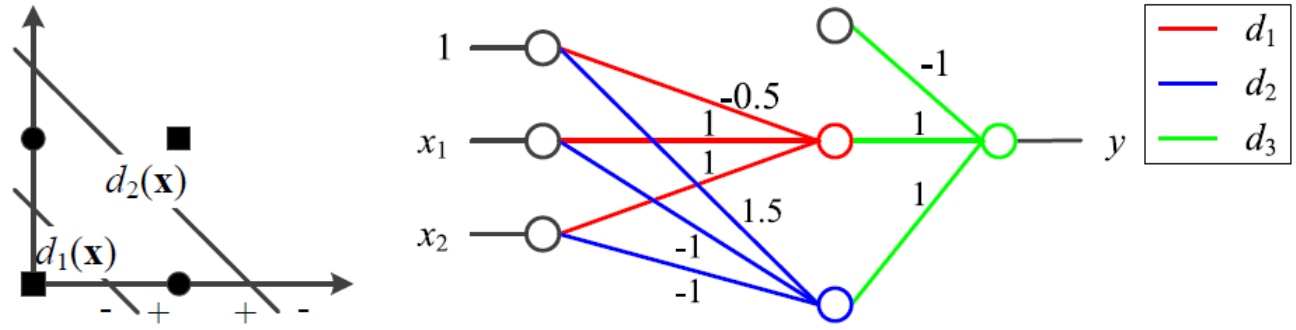
(a) 선형 분리 가능 (b) 선형 분리 불가능

그림 8-8 선형 분리 가능과 불가능

8.2.1 구조와 작동 원리

다층 퍼셉트론으로 확장

- XOR 분류 문제: 퍼셉트론은 75% 정인식률이 한계
- 어떻게 100% 성능을 달성할 것인가? → 세 개의 퍼셉트론을 사용하여 해결
  - d_1의 +영역과 d_2의 +영역이 겹친 영역은 ω_1, 나머지는 ω_2



(a) XOR 분류 문제 (b) 세 개 퍼셉트론의 결합

그림 8-9 다층 퍼셉트론

첫 번째 단계에서 두 개의 퍼셉트론 d_1과 d_2를 이용하여 특징 벡터를 새로운 공간으로 매핑한 후, 새로운 공간에서 하나의 퍼셉트론 d_3을 이용하여 최종 분류한 것으로 간주할 수 있다.

8.2.1 구조와 작동 원리

- 다층 퍼셉트론의 구조
  - 입력층→은닉층→출력층
    - 입력층: 특징 벡터의 차원에 따라 d개의 노드 (여분의 바이어스 노드)
    - 출력층: 부류 개수에 따라 m개의 노드 (m은 부류 개수)
    - 은닉층: 노드 개수 p를 사용자가 설정
  - 입력층-은닉층을 잇는 가중치는 u_{ji}, 은닉층-출력층은 v_{jk}로 표기

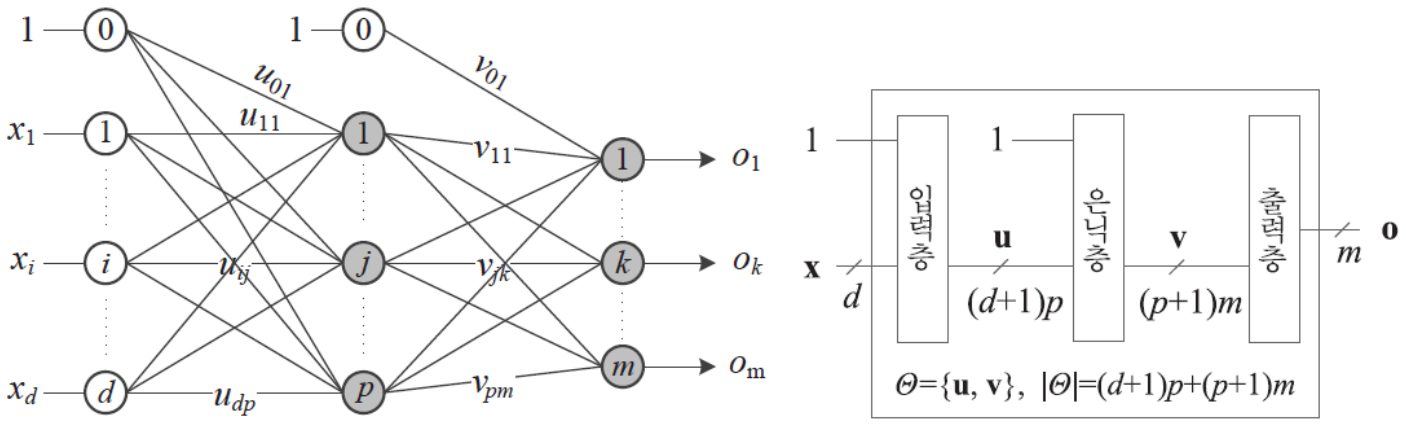


그림 8-10 다층 퍼셉트론의 구조와 표기

8.2.1 구조와 작동 원리

- 다층 퍼셉트론의 동작을 수식으로 쓰면,

$$\mathbf{o} = f(\mathbf{x}) \text{ 또는 두 단계로 나누어 쓰면 } \mathbf{o} = q(\mathbf{z}),\ \mathbf{z} = p(\mathbf{x}) \tag{8.3}$$

은닉층의 j번째 노드, $1 \le j \le p$:

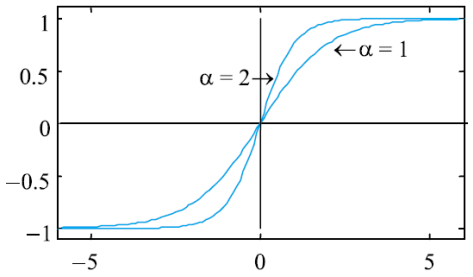
$$z\_sum_j = \sum_{i=1}^{d} x_i u_{ij} + u_{0j} \tag{8.4}$$
$$z_j = \tau(z\_sum_j)$$

출력층의 k번째 노드, $1 \le k \le m$:

$$o\_sum_k = \sum_{j=1}^{p} z_j v_{jk} + v_{0k} \tag{8.5}$$
$$o_k = \tau(o\_sum_k)$$

양극 시그모이드 함수와 도함수 :

$$\tau(x) = \frac{2}{1 + e^{-\alpha x}} - 1$$
$$\tau'(x) = \frac{\alpha}{2}(1 + \tau(x))(1 - \tau(x))$$



8.2.2 학습

다층 퍼셉트론 학습의 정의

- MLP 학습이란?
  - 훈련 집합 X={(x1,t1), (x2,t2), …, (xN,tN)}이 주어졌을 때, 이들을 분류하는 다층 퍼셉트론(즉, Θ={u,v})을찾아라. (xi,ti)에서 xi는 특징 벡터이고 ti는 목적 벡터target vector로서 xi∈ωj이면 ti=(-1,…,1,…,-1)이다. 즉 j번째 요소만 1이고 나머지 요소는 모두 -1을 갖는다.

- 목적 함수: $\Theta=\{\mathbf{u},\mathbf{v}\}$의 품질을 측정해주는 역할
  - 기대한 $E(\Theta)=\frac{1}{2}\sum_{i=1}^{m}(t_i-o_i)^2$ P의 출력 **o**의 오류 함수 E로 정의

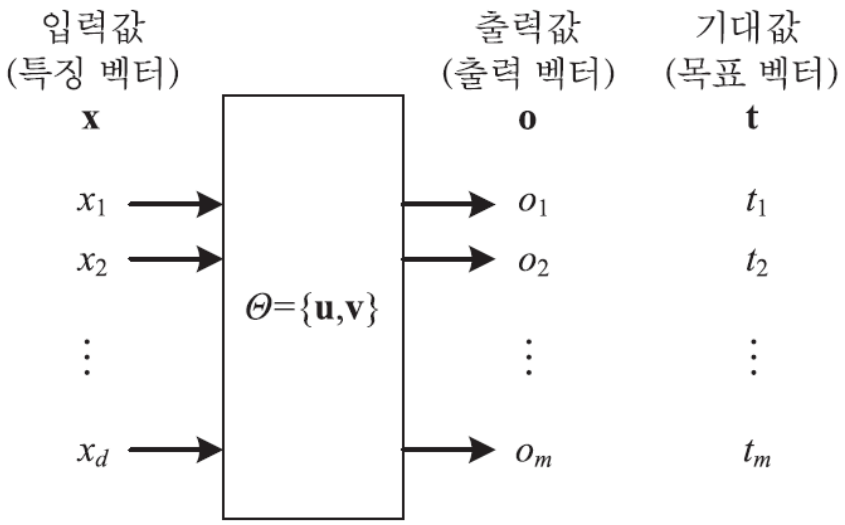


그림 8–11 다층 퍼셉트론의 입력값, 출력값, 기댓값

8.2.2 학습

- 학습의 원리
  - 수렴할 때까지, 오류 $E(\Theta)$를 줄이는 쪽으로 Θ를 수정해 나감 ← 내리막 경사법
    - 미분으로 오류가 줄어드는 방향을 알아냄
    - $\frac{\partial E}{\partial \mathbf{v}}$와 $\frac{\partial E}{\partial \mathbf{u}}$는 커지는 방향이므로, 현재 Θ에 $-\frac{\partial E}{\partial \mathbf{v}}$와 $-\frac{\partial E}{\partial \mathbf{u}}$를 더해주면 오류를 줄이는 방향으로 갱신한 셈
    - ρ는 학습률로서, 갱신하는 양을 조절해 줌

$$\mathbf{v}(h+1) = \mathbf{v}(h) + \Delta\mathbf{v} = \mathbf{v}(h) - \rho\frac{\partial E}{\partial \mathbf{v}}$$

$$\mathbf{u}(h+1) = \mathbf{u}(h) + \Delta\mathbf{u} = \mathbf{u}(h) - \rho\frac{\partial E}{\partial \mathbf{u}} \qquad (8.8)$$

8.2.2 학습

- 알고리즘으로 정리하면,

알고리즘 8-3 다층 퍼셉트론(MLP) 학습 알고리즘

입력 : 훈련 집합 $X=\{(\mathbf{x}_1,t_1), (\mathbf{x}_2,t_2), \cdots, (\mathbf{x}_N,t_N)\}$, 학습률 ρ

출력 : 가중치 $\mathbf{u}$와 $\mathbf{v}$

```
1  u와 v를 초기화한다.
2  repeat {
3    for(X의 샘플 각각에 대해) {
4      식 (8.4)와 (8.5)로 전방 계산을 한다.
5      ∂E/∂v와 ∂E/∂u를 계산한다.
6      식 (8.8)로 새로운 u와 v를 계산한다.
7    }
8  } until(stop-condition);
```

← 오류 역전파 알고리즘

8.2.2 학습

- 5 행이 사용할 수식 (일반 델타 규칙)

$$\delta_k = (t_k - o_k)\tau'(o\_sum_k),\ 1 \le k \le m \tag{8.9}$$

$$\Delta v_{jk} = -\rho\frac{\partial E}{\partial v_{jk}} = \rho\delta_k z_j,\ 0 \le j \le p,\ 1 \le k \le m \tag{8.10}$$

$$\eta_j = \tau'(z\_sum_j)\sum_{k=1}^{m}\delta_k v_{jk},\ 1 \le j \le p \tag{8.11}$$

$$\Delta u_{ij} = -\rho\frac{\partial E}{\partial u_{ij}} = \rho\eta_j x_i,\ 0 \le i \le d,\ 1 \le j \le p \tag{8.12}$$

- 학습할 때 고려할 사항
  - 가중치 초기화
  - 학습률
  - 종료 조건
  - 샘플 처리 순서 등

8.2.2 학습

알고리즘 8-4 다층 퍼셉트론(MLP) 학습을 위한 오류 역전파 알고리즘

입력 : 훈련 집합 $X=\{(\mathbf{x}_1,\mathbf{t}_1),(\mathbf{x}_2,\mathbf{t}_2),\cdots,(\mathbf{x}_N,\mathbf{t}_N)\}$, 학습률 ρ

출력 : 가중치 u와 v

```
// 초기화
u와 v를 초기화한다.
x0=z0=1; // 바이어스
repeat {
  for(X의 샘플 각각에 대해) {
    현재 샘플을 x=(x1,x2,⋯,xd)와 t=(t1,t2,⋯,tm)으로 표기한다.

// 전방 계산
    for(j=1 to p) {z_sum_j = Σ_{i=0}^{d} x_i u_ij ; z_j = τ(z_sum_j); }  // 식 (8.4)
    for(k=1 to m) {o_sum_k = Σ_{j=0}^{p} z_j v_jk ; o_k = τ(o_sum_k);}  // 식 (8.5)

// 오류 역전파
    for(k=1 to m) δ_k = (t_k - o_k)τ'(o_sum_k);   // 식 (8.9)
    for(모든 v_jk, 0 ≤ j ≤ p, 1 ≤ k ≤ m에 대해) Δv_jk = ρδ_k z_j ;   // 식 (8.10)
    for(j=1 to p)  η_j = τ'(z_sum_j) Σ_{k=1}^{m} δ_k v_jk  ;  // 식 (8.11)
    for(모든 u_ij, 0 ≤ i ≤ d, 1 ≤ j ≤ p에 대해)  Δu_ij = ρη_j x_i ;  // 식 (8.12)
    for(모든 v_jk, 0 ≤ j ≤ p, 1 ≤ k ≤ m에 대해)  v_jk = v_jk + Δv_jk ;  // 식 (8.8)
    for(모든 u_ij, 0 ≤ i ≤ d, 1 ≤ j ≤ p에 대해)  u_ij = u_ij + Δu_ij ;  // 식 (8.8)
  }
}until(stop-condition);
```

8.2.2 학습

- MLP로 인식
  - 전방 계산을 한 번만 하므로 빠르게 동작

알고리즘 8-5 다층 퍼셉트론(MLP)에 의한 인식

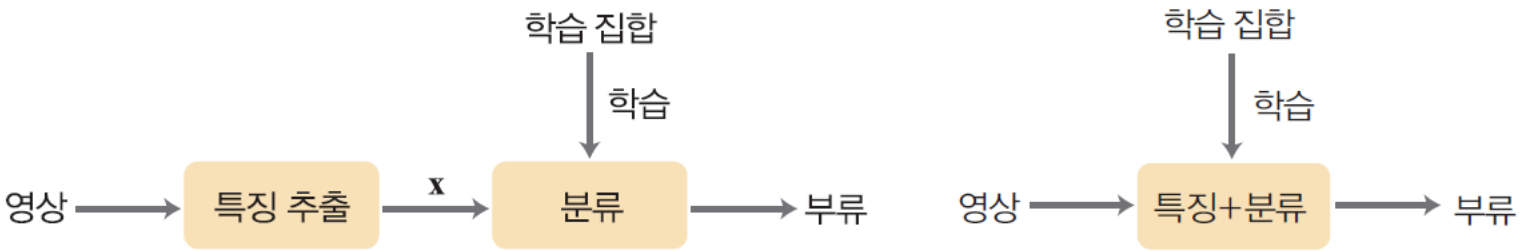
입력 : MLP(**u**와 **v**), 테스트 샘플의 특징 벡터 **x**

출력 : 부류

1. **u**와 **v**를 읽어 MLP를 설정한다.
2. $x_0=z_0=1$; // 바이어스
3. for(j=1 to p) {$z\_sum_j = \sum_{i=0}^{d} x_i u_{ij}$; $z_j = \tau(z\_sum_j)$;} // 은닉 노드 연산(식 8.4)
4. for(k=1 to m) {$o\_sum_k = \sum_{j=0}^{p} z_j v_{jk}$; $o_k = \tau(o\_sum_k)$;} // 출력 노드 연산(식 8.5)
5. 출력 노드 중 가장 큰 값을 갖는 o_i를 찾아 **x**를 ω_i로 분류한다.

8.2.3 깊은 학습

- 인식 프로그램 제작: 전통적인 방식 vs. 깊은 학습 방식
  - 전통적인 방식
    - 특징 추출의 세밀한 단계까지 사람이 설계하고 구현
    - 분류기는 기계 학습으로 따로 제작한 후, 연결하여 사용
  - 1990년대에 깊은 학습 창안 → 특징 추출과 분류를 하나의 학습 모델로 처리



(a) 사람이 설계한 특징+학습에 의한 분류 (b) 학습에 의한 특징+분류

그림 8-12 인식 프로그램을 제작하는 두 가지 접근 방식

8.2.3 깊은 학습

- 컨볼루션 신경망 [LeCun 98]
  - 특징은 고정된 마스크로 컨볼루션을 수행함으로써 추출 (예, 에지 마스크)

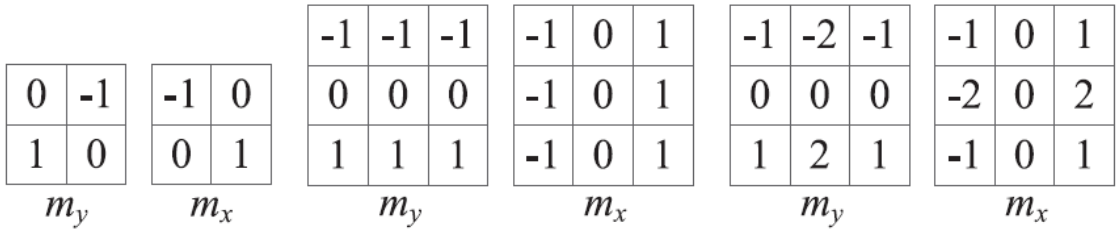


그림 3–5 에지 연산자

  - 주어진 문제에 가장 적합한 마스크를 학습으로 알아내자. ← 컨볼루션 신경망의 발상!

8.2.3 깊은 학습

- 컨볼루션 신경망의 구조
  - 일곱 층으로 구성
    - 앞 단 C1-S2-C3-S4-C5는 특징 추출 담당 (C층은 컨볼루션, S는 다운샘플링 수행)
    - 뒤 단 F6-O는 분류 담당 (MLP와 같은 구조)
  - 빨간 선은 학습이 알아내야 하는 가중치

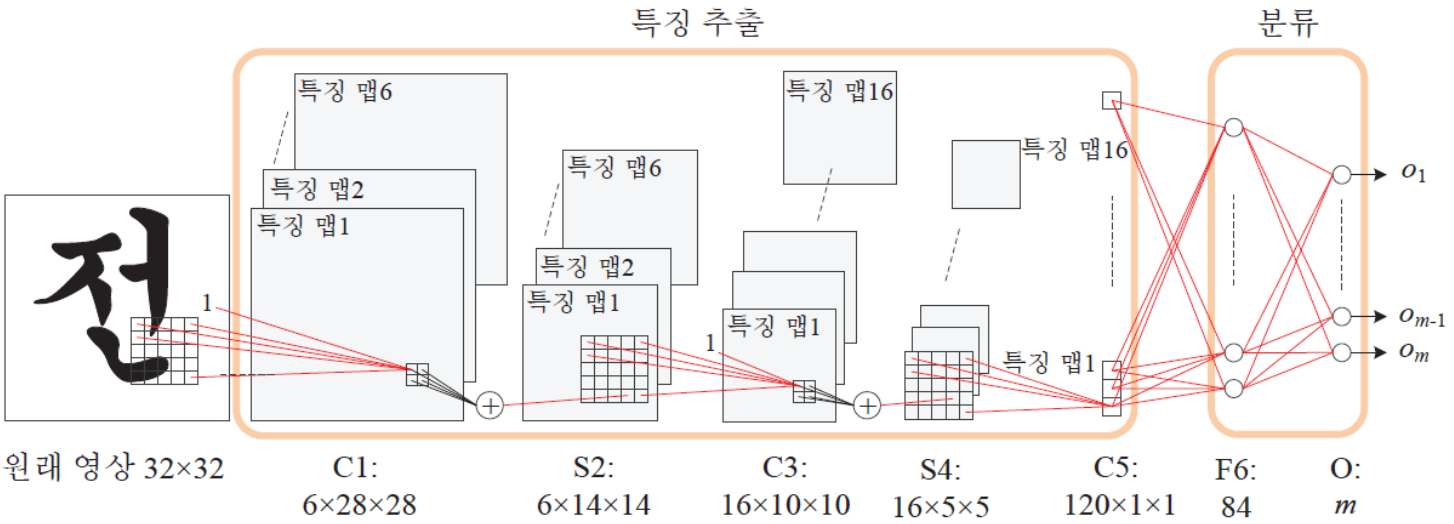


그림 8-13 컨볼루션 신경망

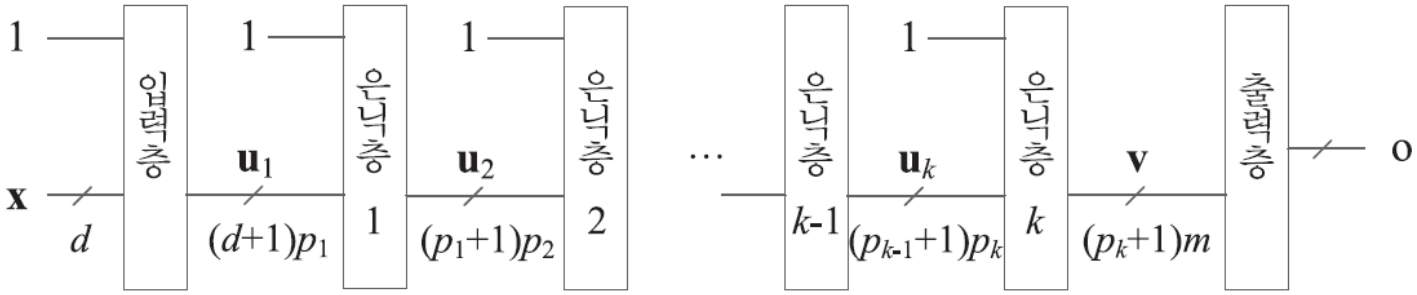
8.2.3 깊은 학습

- 컨볼루션 신경망의 학습
  - 빨간 선은 학습이 알아내야 하는 가중치
    - C 층의 마스크를 학습이 설계하는 셈

    →컨볼루션 신경망은 특징 추출에 쓸 마스크를 학습이 자동으로 알아냄
    - 뒷 단은 MLP와 같음
  - 층이 깊어 MLP가 사용하는 오류 역전파 알고리즘만으로 학습 불가능
    - LeCun은 개선된 학습 알고리즘으로 해결

8.2.3 깊은 학습

- MLP의 은닉층 개수를 늘린 깊은 신경망 [Ciresan2010]
  - 필기 숫자 인식(MNIST 데이터베이스)에서 뛰어난 성능 확인

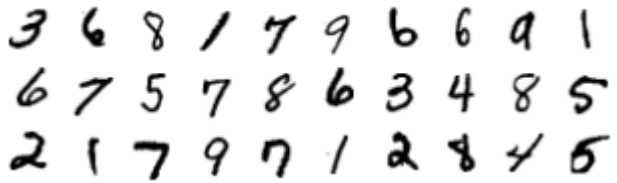


$$\Theta=\{\mathbf{u}_1,\mathbf{u}_2,\cdots,\mathbf{u}_k,\mathbf{v}\},\ |\Theta|=(d+1)p_1+(p_1+1)p_2+\cdots+(p_{k-1}+1)p_k+(p_k+1)m$$

그림 8–14 MLP를 확장한 깊은 신경망

8.2.3 깊은 학습

- 깊은 신경망의 우수성
  - MNIST 데이터베이스에서 챔피온 (http://yann.lecun.com/exdb/mnist/)



  - 1000부류의 자연 영상을 인식하는 ILSVRC 대회에서 챔피온

그림 8–15 이미지넷 데이터베이스의 '망치'와 '종' 부류에 속한 영상[5]

8.2.3 깊은 학습

- 2012 대회는 토론토 대학 팀이 제출한 수퍼비전이라는 프로그램이 우승을 차지하였다.
  - [그림 8-13]과 비슷한 컨볼루션 신경망을 사용한 프로그램
  - 65만 개의 뉴런(노드), 6천만 개의 매개변수를 갖는 엄청나게 큰 구조의 신경망
  - 두 개의 NVIDIA GPU에서 병렬 프로그래밍으로 기계 학습을 수행하였는데, 학습에 1주일가량 걸렸다고 보고하였다.
  - 구체적인 구조와 훈련 방법은 [Krizhevsky2012] 참고

ILSVRC 대회

- 현재 깊은 학습은 컴퓨터 비전 연구의 중요한 동력이다. 또한 깊은 학습 기술은 컴퓨터 비전, 음성 인식, 자연어 처리 등의 인공지능 영역에서 성능 도약을 가져왔다.
  - 보다 폭넓고 깊은 공부를 원한다면 → [Bengio2009, Bengio2013] 참고
  - 구글, 페이스북, IBM, 마이크로소프트와 같은 회사가 이 기술을 어떻게 바라보며 활용 전략을 짜는지 조망하고 싶다면 → [Jones2014] 참고

8.3 SVM

8.3.1 선형 SVM

8.3.2 비선형 SVM

8.3.3 학습과 인식

8.3 SVM

- SVM의 뛰어난 일반화 능력
  - 학습 집합 입장에서는 ②와 ③은 동일
  - 테스트 집합에서는 (즉 일반화 능력 측면에서는) 여백이 더 큰 ③이 ②보다 우수
  - 신경망은 ①에서 시작하여 ②에 도달하였다면 거기서 멈춤
  - SVM은 최대 여백을 구함

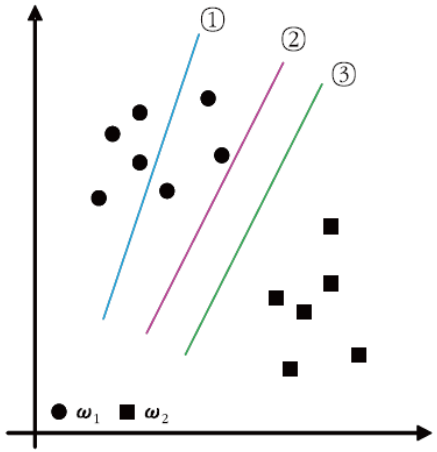


그림 8-16 분류기의 일반화 능력

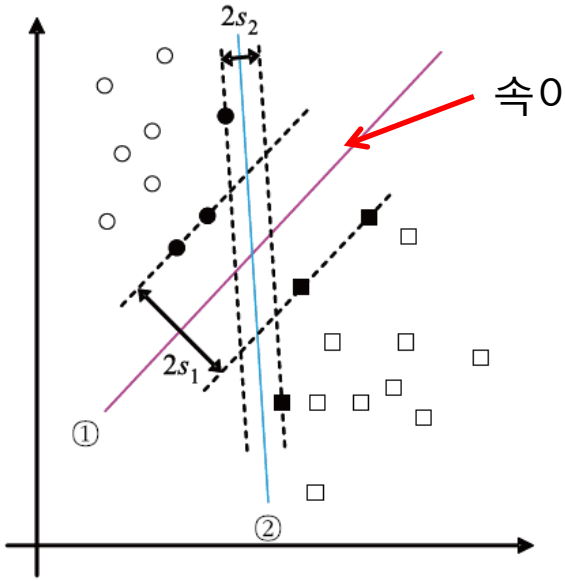
8.3.1 선형 SVM

- 선형 분리 가능한 상황
  - 결정 초평면

$$d(\mathbf{x}) = \mathbf{w}\mathbf{x}^{\mathrm{T}} + b = 0$$

여백을 가장 크게 하는 결정 초평면의 방향, 즉 **w**를 찾아라. (8.14)

  - ①이 ②보다 우수. 더 좋은 **w**가 있나? → 수학으로 찾아야 함



속이 찬 샘플이 서포트 벡터

그림 8-17 선형 분리가 가능한 상황

8.3.1 선형 SVM

- 조건부 최적화 문제
  - 결정 초평면에서 서포트 벡터까지 거리가 1이 되도록 방정식을 크기 조절하면,

$$\text{여백} = \frac{2|d(\mathbf{x})|}{\|\mathbf{w}\|}\text{인데, 크기를 조절하면 여백} = \frac{2}{\|\mathbf{w}\|} \qquad (8.15)$$

  - 학습 집합 $X=\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \ldots, (\mathbf{x}_N, t_N)\}$일 때, $\mathbf{x}_i$가 ω_1에 속하면 $t_i=1$, ω_2에 속하면 $t_i=-1$이라고 표기하면,

$$\text{아래 조건 하에,}$$
$$t_i(\mathbf{w}\mathbf{x}_i^T + b) - 1 \geq 0, \quad 1 \leq i \leq N \qquad (8.16)$$
$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2\text{을 최소화하라.}$$

8.3.1 선형 SVM

조건부 최적화 문제

- 라그랑주 승수를 도입하면,

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (t_i(\mathbf{w}\mathbf{x}_i^{\mathrm{T}} + b) - 1) \tag{8.17}$$

- $L(.)$을 $\mathbf{w}$와 b로 미분한 후 0으로 놓고 KKT 조건을 유도하면,

$$\text{아래 조건 하에,}$$
$$\sum_{i=1}^{N} \alpha_i t_i = 0, \quad \alpha_i \geq 0, \quad 1 \leq i \leq N$$
$$\tilde{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \mathbf{x}_i \cdot \mathbf{x}_j \text{를 최대화하라.} \tag{8.18}$$

  - $\mathbf{w}$와 b가 사라짐 → 라그랑주 승수 α_i를 구하는 문제로 바뀜
  - 특징 벡터가 내적의 형태로 나타남 → 비선형으로 확장하는 근거가 됨

8.3.1 선형 SVM

선형 분리가 불가능한 상황

- 분할 띠 내부에 샘플이 존재하도록 허락하면, 세 가지 경우 발생

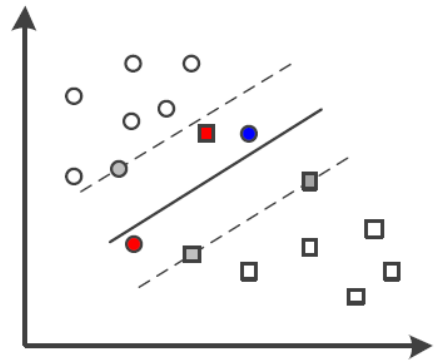


그림 8-18 선형 분리가 불가능한 상황의 SVM

- 경우1 : 분할 띠의 바깥에 있다. 속이 비었거나 회색인 점들이다. 회색 점이 서포트 벡터이다. $1 \leq t(\mathbf{w}\mathbf{x}^{\mathrm{T}} + b)$를 만족한다.
- 경우2 : 분할 띠의 안쪽에 있는데 자기가 속한 부류의 영역에 있다. 속이 파란색 점이다. 분할 띠 바깥에 있어야 한다는 조건은 어겼지만 옳게 분류된다. $0 \leq t(\mathbf{w}\mathbf{x}^{\mathrm{T}} + b) < 1$을 만족한다.
- 경우3 : 결정 경계를 넘어 자신이 속하지 않은 부류의 영역에 놓여 있다. 빨간색 점으로서 틀리게 분류된다. $t(\mathbf{w}\mathbf{x}^{\mathrm{T}} + b) < 0$을 만족한다.

- 슬랙 변수 ξ를 도입하여 다시 쓰면,

$$t(\mathbf{w}\mathbf{x}^{\mathrm{T}} + b) \geq 1 - \xi$$

  - 경우 1은 ξ=0, 경우 2는 0<ξ≤1, 경우 3은 1<ξ

8.3.1 선형 SVM

- 최적화 문제를 다시 쓰면,

여백을 될 수 있는 한 크게 하며(목적1), 동시에 경우2 또는 경우3에 해당하는 $0<\xi$인 샘플의 수를 될 수 있는 한 적게 하는(목적2) 결정 초평면의 방향 $\mathbf{w}$를 찾아라. (8.20)

  - 목적 함수를 다시 쓰면,
    - C는 두 가지 목적 중에 어느 것이 비중을 둘 지 결정하는 매개변수 (SVM을 사용할 때 지정해야 하는 사용자 매개변수)
    - C가 0이면 목적2를 무시, 아주 크면 목적2만 고려

$$J(\mathbf{w},\boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i, \quad C \geq 0 \tag{8.21}$$

8.3.1 선형 SVM

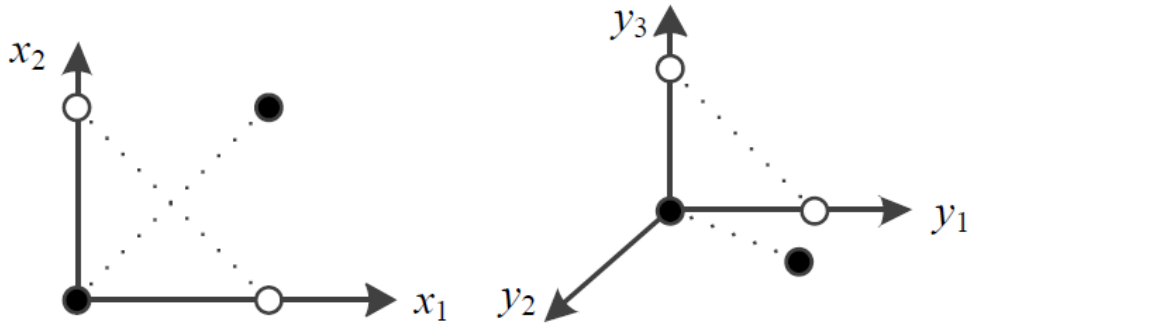
- 라그랑주 승수를 도입하고 KKT 조건을 유도하면,

$$\text{아래 조건 하에,}$$
$$\sum_{i=1}^{N} \alpha_i t_i = 0$$
$$0 \le \alpha_i \le C,\ 1 \le i \le N \qquad (8.22)$$
$$\tilde{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \mathbf{x}_i \cdot \mathbf{x}_j \text{를 최대화하라.}$$

  - 조건식 하나 빼고 선형 분리 가능한 경우의 수식 (8.18)과 동일
    - $0 \le \alpha_i$ → $0 \le \alpha_i \le C$

8.3.2 비선형 SVM

- 비선형 SVM으로 확장
  - 원래 특징 공간을 더 높은 차원의 공간으로 매핑하면 선형 분리에 유리해짐
  - 예) 2차원 공간에서 선형 분리 불가능한 XOR 문제를 3차원으로 매핑하면 선형 분리 가능해짐



(a) 원래 공간 $\mathbf{x}=(x_1, x_2)$ (b) 매핑된 공간 $\Phi(\mathbf{x}) = (y_1, y_2, y_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$

그림 8-19 공간 매핑

8.3.2 비선형 SVM

커널 트릭의 도입

- 매핑 함수 **Φ**(**x**)를 사용하여 식 (8.22)의 목적 함수를 다시 쓰면,

$$\tilde{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \boldsymbol{\Phi}(\mathbf{x}_i) \cdot \boldsymbol{\Phi}(\mathbf{x}_j) \qquad (8.23)$$

- 매핑 함수 **Φ**(**x**)를 직접 구하는 일은 불가능하거나 어려움
- 내적 계산을 커널 함수 계산으로 대치할 수 있음

$$\boldsymbol{\Phi}(\mathbf{x}_i) \cdot \boldsymbol{\Phi}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) \qquad (8.24)$$

많이 활용되는 커널

$$\begin{aligned} &\text{다항식 커널} && K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^p \\ &\text{RBF(radial basis function) 커널} && K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2} \\ &\text{하이퍼볼릭 탄젠트 커널} && K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i \cdot \mathbf{x}_j + \beta) \end{aligned} \qquad (8.25)$$

8.3.3 학습과 인식

- SVM 학습
  - 식 (8.22)를 풀어 라그랑주 승수 α_i를 계산하는 일
  - $\alpha_i \neq 0$인 샘플이 서포트 벡터임
  - 인식 단계에 사용하기 위해 서포트 벡터를 저장

- 인식
  - Y는 학습 단계에서 구해놓은 서포트 벡터 집합

비선형 SVM 분류기:

$d(\mathbf{x}) \geq 0$이면 $\mathbf{x} \in \omega_1$, $d(\mathbf{x}) < 0$이면 $\mathbf{x} \in \omega_2$로 분류하라. (8.27)

$$d(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b = \sum_{\mathbf{x}_k \in Y} \alpha_k t_k \Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}) + b$$
$$= \sum_{\mathbf{x}_k \in Y} \alpha_k t_k K(\mathbf{x}_k, \mathbf{x}) + b \quad (8.26)$$

8.3.3 학습과 인식

- *M*개의 부류로 분류하는 SVM으로 확장
  - 지금까지 공부한 SVM은 이진 분류기
  - *M* 부류로 확장하는 두 가지 방법: 1:*M*-1 방법과 1:1 방법
  - 1:*M*-1 방법
    - ω_j와 나머지 M-1 개 부류를 분류하는 이진 SVM을 M 개 제작

$$M\text{ 부류 SVM 분류기}:$$
$$k = \underset{j}{\operatorname{argmax}}\, d_j(\mathbf{x})\text{를 찾아, }\mathbf{x}\text{를 }\omega_k\text{로 분류하라.} \qquad (8.28)$$

  - 1:1 방법
    - 모든 부류 쌍에 대해 이진 SVM 제작 (총 $M(M\text{-}1)/2$ 개)
    - 투표로 부류를 정함

8.3.3 학습과 인식

SVM 특성

- 사용자가 설정해야 하는 매개변수가 적음
  - 식 (8.21)의 C 그리고 커널 종류와 그에 따른 매개변수
- 최적 커널을 자동 선택하는 방법은 없음
  - 성능 실험으로 결정
  - 도움이 되는 가이드라인 [Hsu2014]

공개 소프트웨어

- SVMLight(http://svmlight.joachims.org/)
  - T. Joachims이 개발한 소프트웨어
  - 학습은 Osuna 알고리즘 사용 [Joachims99]

- LIBSVM(http://www.csie.ntu.edu.tw/~cjlin/libsvm)
  - Chih-Chung Chang과 Chih-Jen Lin이 개발한 소프트웨어
  - 학습은 개선된 SMO 알고리즘을 사용 [Fan2005, Chang2011].
  - 매개변수를 설정하는 실용적인 방법도 제시 [Hsu2014]

8.4 분류기 앙상블

8.4.1 중요한 질문

8.4.2 에이더부스트

8.4.3 임의 숲

8.4 분류기 앙상블

- 협동의 중요성
  - 개인이 가진 서로 다른 능력을 하나로 모음
  - 협동하면 더 좋은 결과 ← 분류기 앙상블의 발상

8.4.1 중요한 질문

- 기초 분류기로 무엇을 사용할까?
  - 앞에서 공부한 MLP나 SVM? 이들은 강한 분류기
  - 앙상블 기법의 전략
    - 강한 분류기를 적게 사용하는 대신, '약한 분류기'를 많이 사용
  - 약한 분류기
    - 무작위로 분류하는 임의 짐작보다 약간 나은 성능을 가진 분류기
    - 예) 하나의 특징 x_i만 사용하여, "$x_i > T$이면 ω_1, 그렇지 않으면 ω_2"

- 여러 개의 분류기를 어떻게 결합할 것인가?
  - 주로 가중 투표 방식 사용: 최다 득표한 부류로 분류
  - k번째 분류기의 출력을 $L_k=(l_{k1}, l_{k2}, \ldots, l_{km})$, 신뢰도를 α_k라 표기하면,

$$\omega_q = \underset{\omega_j}{\operatorname{argmax}} \sum_{k=1}^{K} \alpha_k l_{kj} \qquad (8.29)$$

8.4.1 중요한 질문

- 기초 분류기를 어떻게 만들 것인가?
  - 붓스트랩을 확장한 배깅

알고리즘 8–6 배깅

입력 : 학습 집합 $X=\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$, 샘플링 비율 $p(0<p\leq 1)$, 분류기 개수 K

출력 : 분류기 앙상블 $C=\{c_k,\ 1\leq k\leq K\}$

```
1  C=Ø;
2  for(k=1 to K) {
3     X에서 임의로 pN개의 샘플을 뽑아 X_k라 한다. 이때 대치를 허용한다.
4     X_k로 분류기 c_k를 학습시킨다.
5     C=C∪c_k;
6  }
```

  - 부분 공간 방법 [Ho98]
    - 특징의 부분 공간을 임의로 여럿 뽑고, 각 부분 공간에서 분류기 제작

8.4.2 에이더부스트

부스팅

- 배깅에 비해 정교한 재 샘플링 사용
  - k 번째 분류기 c_k와 그 다음의 c_{k+1}이 연관성을 가지도록 제작
- 학습 집합 X에 있는 샘플은 c_k가 맞추는 것과 틀리는 것으로 구분
  - 맞춘 샘플은 가중치를 낮추고, 틀린 샘플은 여전히 '까다로운' 상대이므로 가중치를 높여 줌
  - c_{k+1}은 가중치를 고려하여 학습함

가장 널리 쓰이는 부스팅: 에이더부스트

- 동일한 가중치를 가지고 출발 (2행)
- 가중치를 고려한 학습 (4행)
- 분류기의 신뢰도 계산 (8행)
- 가중치 갱신 (10~12행)

8.4.2 에이더부스트

알고리즘 8-7 에이더부스트

입력 : 학습 집합 $X=\{(\mathbf{x}_1, t_1),(\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$, 분류기 개수 K

출력 : 분류기 앙상블 $C=\{(c_k, \alpha_k), \ 1 \le k \le K\}$ // 신뢰도 α_k를 갖는 분류기 c_k

```
1   C=∅;
2   for(j=1 to N) w_j=1/N;   // 처음에는 같은 가중치를 줌
3   for(k=1 to K) {
4     w_1, w_2, ···, w_N을 감안하여 분류기 c_k를 학습시킨다.   // 가중치가 큰 샘플을 좀더 중요시함
5     ε=0;
6     for(j=1 to N) if(c_k(x_j)≠t_j) ε=ε+w_j;   // 오류(틀린 샘플의 가중치 합) 계산
7     if(ε<0.5) {   // 0.5보다 작은 경우만 분류기 c_k를 취함
8       α_k = (1/2) log((1-ε)/ε);   // 분류기 c_k의 신뢰도
9       for(j=1 to N)
10        if(c_k(x_j)≠t_j) w_j=w_j×e^α;   // 틀린 샘플의 가중치 높임
11        else w_j=w_j×e^(-α);   // 맞춘 샘플의 가중치 낮춤
12      w_1, w_2, ···, w_N이 합이 1이 되도록 정규화한다.
13      C=C∪(c_k, α_k);
14    }
15    else {c_k=Nil; C=C∪(c_k,0);}
16  }
```

8.4.2 에이더부스트

- 가중치를 고려한 분류기 학습
  - 가중치가 큰 샘플의 선택 확률을 높게 하는 기법

4.1 | X에서 pN개의 샘플을 뽑아 X'라 한다. 이때 $\mathbf{x}_j$가 뽑힐 확률이 w_j가 되도록 한다.

4.2 | X'로 c_k를 학습시킨다.

  - 가중치가 큰 샘플을 틀리면 더 큰 벌점을 주는 기법

4 | $\sum_{j=1, c_k(x_j) \neq t_j}^{N} w_j$가 최소가 되도록 c_k를 학습시킨다.

- 에이더부스트는 분류기의 신뢰도를 제공
  - 식 (8.29)의 가중 투표에 활용 가능

8.4.3 임의 숲

- 원리
  - 분류기가 독립성을 가지도록 제작
  - 기초 분류기로 트리 분류기 사용

- 트리 분류기

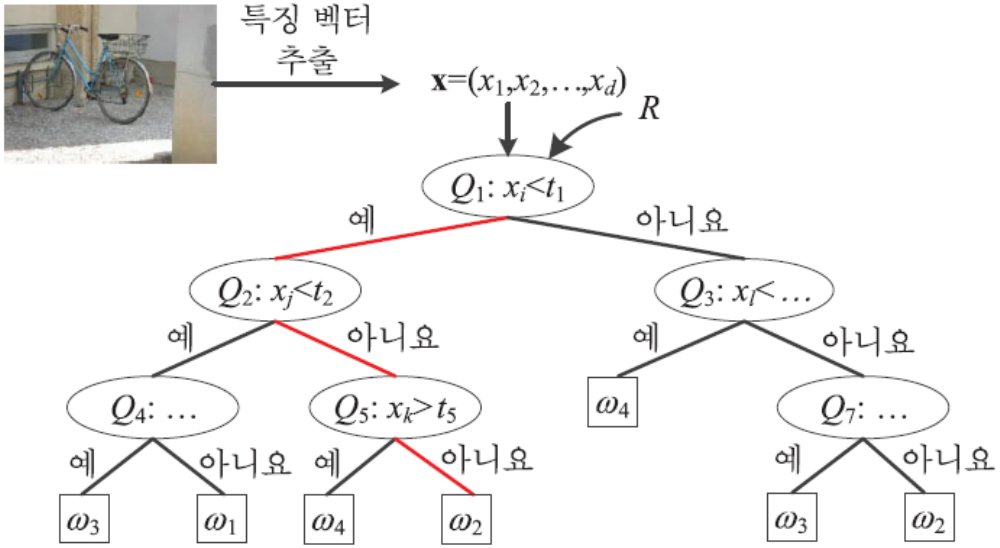


그림 8-20 트리 분류기의 인식 예

8.4.3 임의 숲

- 트리 분류기의 인식 알고리즘

알고리즘 8-8 트리 분류기를 통한 물체 인식

입력 : 트리 분류기 R, 특징 벡터 **x**

출력 : **x**의 부류 ω

```
1  T=R;
2  while(T≠Nil) {
3     x에 대한 T의 질문 Q의 결과를 r이라 한다.
4     if(r=예) T=T의 왼쪽 자식;
5     else T=T의 오른쪽 자식;
6     if(T가 단말 노드) {ω=T의 부류; T=Nil;}
7  }
```

8.4.3 임의 숲

- 트리 분류기의 학습
  - 노드의 질문 Q를 만드는 방법과 잎 노드에 부류를 할당하는 방법을 결정해야 한다.
  - 트리 분류기는 학습 집합 $X=\{(\mathbf{x}_1,t_1),(\mathbf{x}_2,t_2),\cdots,(\mathbf{x}_N,t_N)\}$ 을 가지고 학습해야 한다.
  - 루트 노드의 질문 Q_1은 X를 두 개의 부분집합 X_{left}와 X_{right}로 나눈다.
    - 이때 X_{left}와 X_{right} 는 순도가 높을수록 좋다.
    - 예를 들어, 두 부류 분류인 경우 ω_1은 모두 X_{left}, ω_2는 모두 X_{right}에 포함되면 이상적이다
    - 이 경우에는 두 개의 잎 노드를 만들고 각각ω_1과 ω_2를 갖게 하고 학습을 종료하면 된다.
  - 하지만 현실에서는 그런 이상적인 특징은 없다.

    → 순도를 최대로 하는 최적의 '특징과 분할점(임계값)'을 찾아야 한다. 이 작업을 위한 구체적인 방법은 [오일석2008, 6.1절]을 참고하라.

8.4.3 임의 숲

알고리즘 8-9 트리 분류기 학습

입력 : 학습 집합 $X=\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$

출력 : 트리 분류기 R // 루트 노드를 가리키는 포인터

```
1   노드 하나를 생성하고 그것을 R이라 한다.
2   split(R, X);
3   function split(T, X) {
4     if(X가 멈춤 조건을 만족)
5       T를 잎 노드로 설정하고 부류를 결정한다.
6     else {
7       T에서 최적의 특징과 그것의 최적 분할점을 찾아 질문 Q를 만든다.
8       Q로 X를 X_left와 X_right로 나눈다.
9       새로운 노드 T_left와 T_right를 생성한다.
10      T.leftchild=T_left;
11      T.rightchild=T_right;
12      split(T_left, X_left);
13      split(T_right, X_right);
14    }
15  }
```

8.4.3 임의 숲

임의 숲 제작

- 배깅과 비슷한데, 4행에서 기초 분류기로 트리 분류기를 사용한다는 점이 다름
  - 임의 숲은 배깅을 이용하여 임의성 (분류기의 독립성)을 확보하는 셈

알고리즘 8-10 임의 숲

입력 : 학습 집합 $X=\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$, 샘플링 비율 $p(0<p\leq 1)$, 분류기 개수 K

출력 : 분류기 앙상블 $C=\{c_k,\ 1\leq k\leq K\}$

```
1  C=Ø;
2  for(k=1 to K) {
3    X에서 임의로 pN개의 샘플을 뽑아 X′라 한다. 이때 대치를 허용한다.
4    X′를 학습 집합으로 하고 [알고리즘 8-9]의 '수정된' 버전으로 트리 분류기 c_k를 만든다.
5    C=C∪c_k;
6  }
```

8.4.3 임의 숲

- 추가적인 임의성 확보 전략
  - 4행에서 '수정된' 버전을 사용하여 이중의 임의성
    - 알고리즘 8-9의 7행을 다음과 같이 수정하여 부분 공간도 임의 선택

7.1	d개의 특징 중 임의로 하나의 특징 x_j를 뽑는다.
7.2	T에서 x_j의 최적 분할점을 찾아 질문 Q를 만든다.

- 임의 숲은 메타 알고리즘이다.
  - 임의성을 가진 트리 분류기 앙상블을 만드는 방식은 모두 임의 숲으로 볼 수 있다.
  - 예를 들어 [알고리즘 8-10]의 4행에서 '수정된'을 빼면 학습 집합의 임의성은 남지만 부분 공간의 임의성은 제거된다. 그렇게 만든 분류기 앙상블도 임의 숲에 속한다. 또 다른 방식으로 두 측면 모두 임의성을 제거하고, 대신 노드의 질문을 만들 때 20개의 최적 질문을 생성한 후 그 중에서 임의로 하나를 선택하여 임의성을 투입하는 기법도 있다 [ietterich2000].

8.5 기계 학습을 이용한 얼굴 검출

- 얼굴 인식 시스템
  - 검출과 인식

그림 8-21 얼굴 검출과 인식

8.5 기계 학습을 이용한 얼굴 검출

- 얼굴 검출
  - 서베이 논문 [Zhang2010, Yang2002]
  - 신경망을 이용한 연구 사례 [Rowley98]

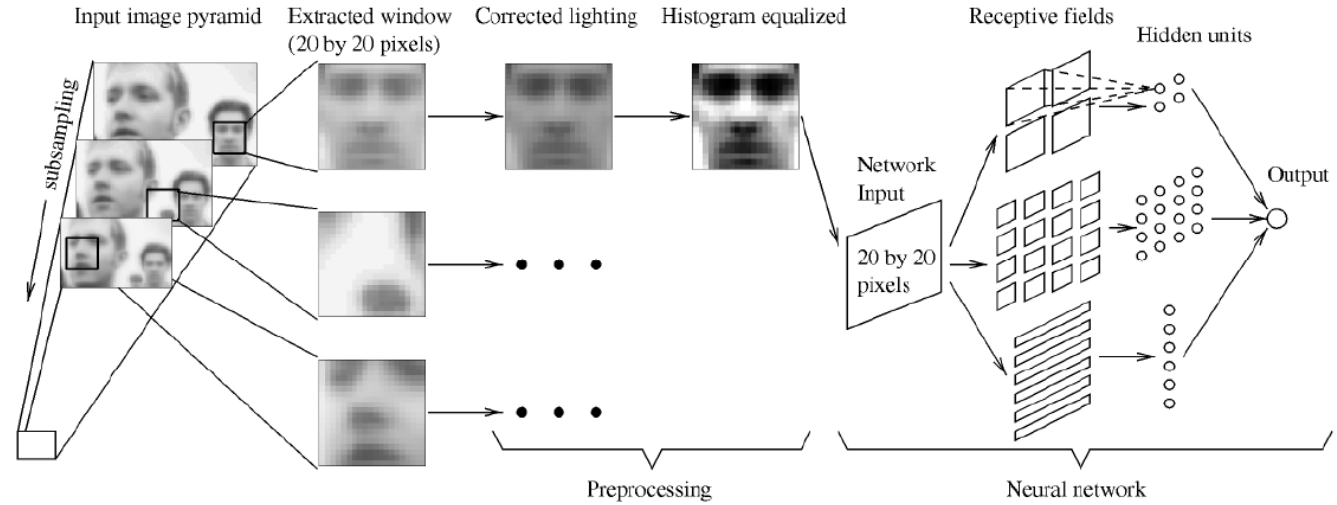


그림 8-22 신경망을 이용한 얼굴 검출 알고리즘

  - SVM을 이용한 사례 [Osuna97]
  - 컨볼루션 신경망(깊은 학습)을 이용한 사례 [Garcia2004, Osadchy2005]

8.5 기계 학습을 이용한 얼굴 검출

- 비올라 존스 얼굴 검출 [Viola2004]
  - 얼굴 검출에서 사실상 표준 노릇
  - 새로운 발상 → 앙상블
  - 전통적인 접근 방법은 복잡한 분류기를 사용하는데 반해, 비올라와 존스는 정반대의 접근 방법을 취하였다[Viola2001].
  - 비올라 존스 얼굴 검출 방법의 특징
    - 웨이블릿 중에서 가장 단순한 형태인 하르 웨이블릿(Haar wavelet)을 닮은 유사 하르 특징(Haar-like feature) 사용
    - 분류기로는 깊이가 낮은 단순한 결정 트리를 여러 개 연결한 직렬 분류기(cascade of classifiers) 사용
    - 이 방법처럼 단순한 특징과 단순한 분류기를 가지고 어떻게 [그림 8-21]과 같은 영상에서 얼굴을 찾아낼까?

8.5 기계 학습을 이용한 얼굴 검출

유사 하르 특징

- 네 종류의 연산자: 검은 영역의 합에서 흰 영역의 합을 빼는 단순 연산
- 예)
  - f_2는 대략 눈과 미간에 위치하므로 얼굴 샘플에서 일정한 경향을 띨 것이나 얼굴 아닌 샘플에서는 무작위 값 발생 → 얼굴과 얼굴 아님을 구분하는 분별력 지님
  - f_3는 볼 부근이라 분별력 없음
- 24*24 영상에 네 종류 연산자를 위치와 크기를 변화시키며 적용할 때, 160,000가지의 서로 다른 특징 추출이 가능 → 이들 중 분별력이 좋은 특징만 선택하여 씀

(a) 유사 하르 연산자

(b) 얼굴 샘플

(c) 얼굴이 아닌 샘플

그림 8-23 유사 하르 연산자를 이용한 특징 추출

8.5 기계 학습을 이용한 얼굴 검출

- 분류기 앙상블
  - 기초 분류기로 루트 노드 하나뿐인 트리 그루터기 사용 ← 약한 분류기
    - 루트 노드는 '$f_j < \theta_j$' 형태의 질문을 가짐
  - 학습은 알고리즘 8-7을 개조한 에이더부스트 사용
    - 학습은 특징 선택과 질문 제작을 동시에 수행
  - 실험 결과, 단지 200개의 트리 그루터기로 95%의 검출 성공률

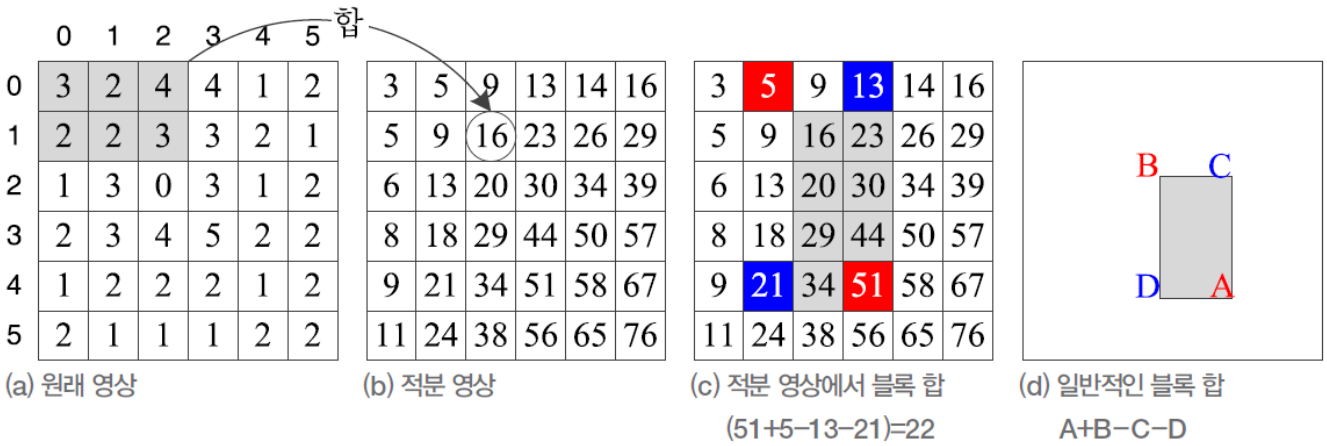
8.5 기계 학습을 이용한 얼굴 검출

적분 영상을 이용한 속도 향상

- 적분 영상은 왼쪽 위 구석부터 현재 화소까지 합으로 정의

$$ii(y,x) = \sum_{\acute{y} \le y, \acute{x} \le x} f(\acute{y}, \acute{x}) \qquad (8.30)$$

- 적분 영상을 사용하면 블록 합을 블록 크기에 무관하게 세 번의 덧셈으로 가능



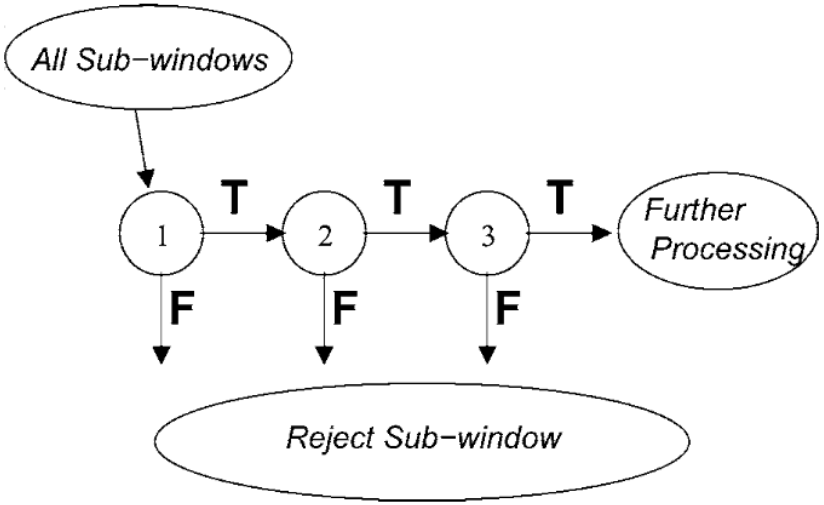
(a) 원래 영상 (b) 적분 영상 (c) 적분 영상에서 블록 합 (51+5−13−21)=22 (d) 일반적인 블록 합 A+B−C−D

그림 8-24 적분 영상을 이용한 매우 효율적인 특징 계산

8.5 기계 학습을 이용한 얼굴 검출

얼굴 검출 알고리즘

- 지금까지는 얼굴과 얼굴 아닌 영상(그림 8-23)을 분류하는 작업을 설명
- 자연 영상에서는 영상의 어디에 어떤 크기의 얼굴이 나타날 지 알 수 없으므로,
  - 영상 전체에 걸쳐 24*24 크기의 윈도우를 촘촘하게 몇 화소 씩 이동시키며 검사
  - 윈도우 크기를 점점 키우면서 여러 번 검사 (실험에서는 1.25배씩 키움)
- 직렬 분류기 아이디어를 이용하여 속도 향상
  - 얼굴 가능성 없는 윈도우를 조기에 기각하는 전략



8.5 기계 학습을 이용한 얼굴 검출

- OpenCV의 `detectMultiScale()` 함수로 검출한 예 (부록 A)

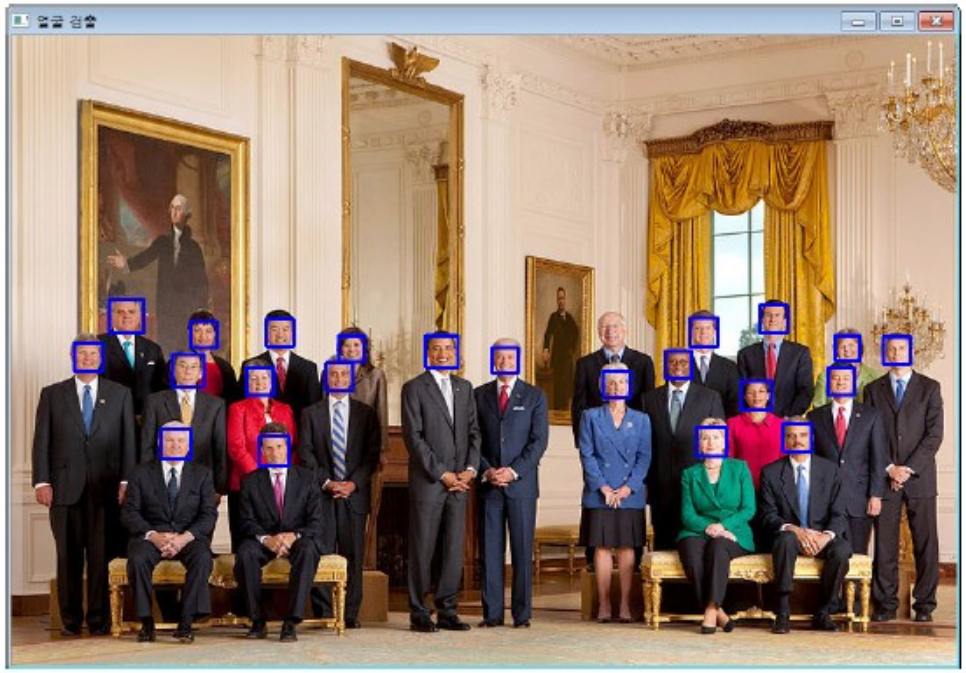


그림 A-18 얼굴 검출 결과