

인 공 지 능

[다층 퍼셉트론 II]

본 자료는 해당 수업의 교육 목적으로만 활용될 수 있음.
일부 내용은 다른 교재와 논문으로부터 인용되었으며, 모든 저작권은 원 교재와 논문에 있음.

3.3 다층 퍼셉트론

3.3 다층 퍼셉트론

■ 퍼셉트론: 선형 분류기 linear classifier 한계

- [그림 3-7(b)]의 선형 분리 불가능한 상황에서 일정한 양의 오류
 - 예) XOR 문제에서 75% 정확도 한계

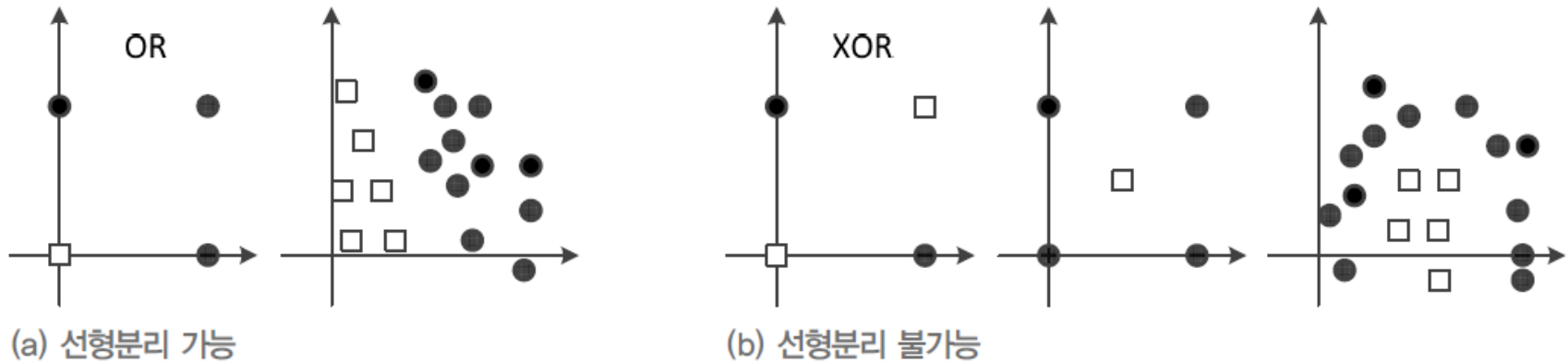


그림 3-7 선형분리가 가능한 상황과 불가능한 상황

- 1969년 Minsky와 Papert의 『Perceptrons』
 - 퍼셉트론의 한계를 지적하고 다층 구조를 이용한 극복 방안 제시하였지만 당시 기술로 불가능
- 1974년 Werbos는 오류 역전파 error backpropagation 알고리즘의 신경망 활용 가능성 확인
- 1986년 Rumelhart, Hinton의 『Parallel Distributed Processing』
 - 다층 퍼셉트론 이론 정립하고, 신경망 재부활

3.3 다층 퍼셉트론

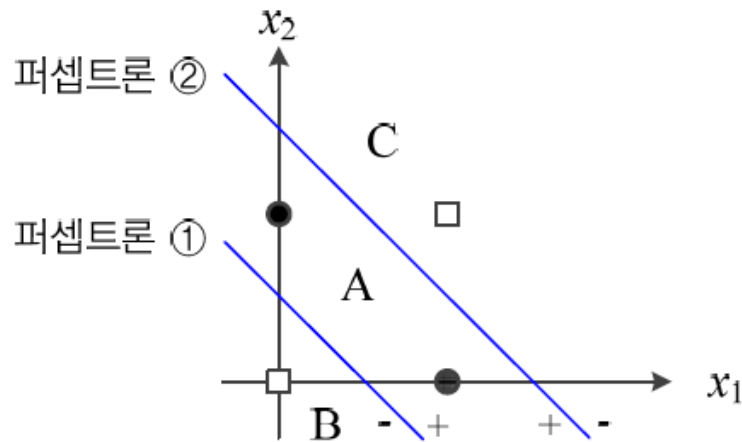
■ 다층 퍼셉트론의 핵심 아이디어

- 은닉층을 둔다. 은닉층은 원래 특징 공간을 분류하는 데 훨씬 유리한 새로운 특징 공간으로 변환한다. 3.3.1절에서 다층 퍼셉트론의 공간 변환 능력을 설명한다.
- 시그모이드 활성화함수를 도입한다. 퍼셉트론은 [그림 3-3(b)]의 계단함수를 활성화함수로 사용하였다. 이 함수는 경성^{hard} 의사결정에 해당한다. 반면, 다층 퍼셉트론은 연성^{soft} 의사결정이 가능한 [그림 3-12]의 시그모이드함수를 활성화함수로 사용한다. 연성에서는 출력이 연속값인데, 출력을 신뢰도로 간주함으로써 더 융통성 있게 의사결정을 할 수 있다. 3.3.2절에서 시그모이드 활성화함수를 자세히 설명한다.
- 오류 역전파 알고리즘을 사용한다. 다층 퍼셉트론은 여러 층이 순차적으로 이어진 구조이므로, 역방향으로 진행하면서 한 번에 한 층씩 그레디언트를 계산하고 가중치를 갱신하는 방식의 오류 역전파 알고리즘을 사용한다. 이 학습 알고리즘에 대해서는 3.4절에서 다룬다.

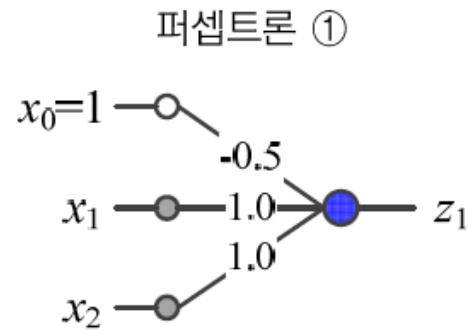
3.3.1 특징 공간 변환

■ 퍼셉트론 2개를 사용한 XOR 문제의 해결

- 퍼셉트론①과 퍼셉트론②가 모두 +1이면 ● 부류이고 그렇지 않으면 □ 부류임



(a) 퍼셉트론 2개를 이용한 공간분할



(b) 퍼셉트론 2개

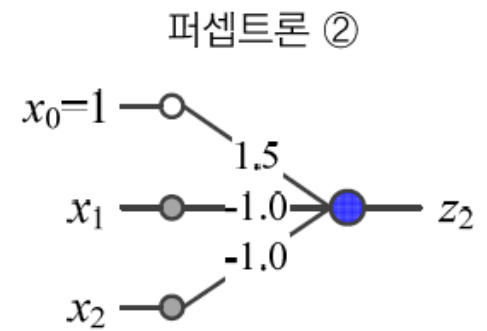
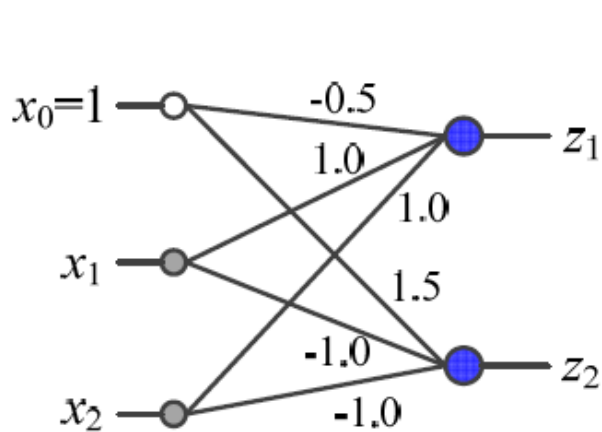


그림 3-8 XOR 문제의 해결

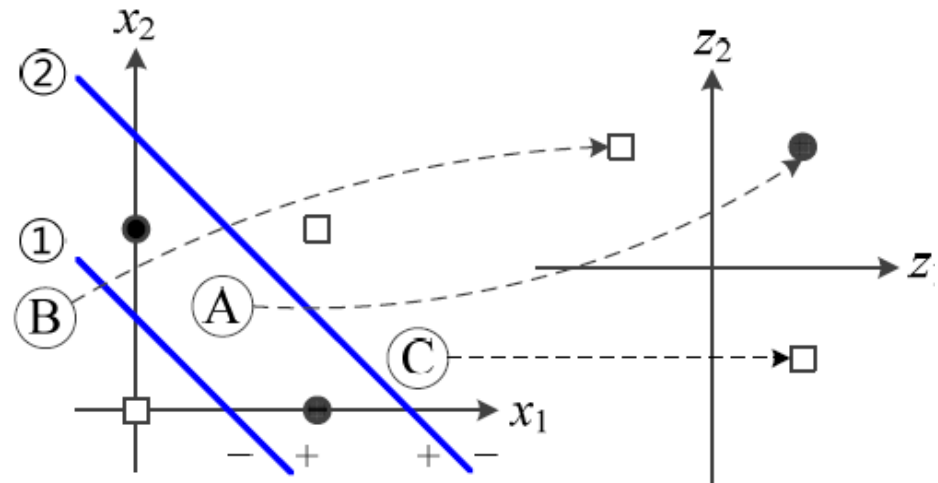
3.3.1 특징 공간 변환

■ 퍼셉트론 2개를 병렬 결합하면,

- 원래 공간 $\mathbf{x} = (x_1, x_2)^T$ 를 새로운 특징 공간 $\mathbf{z} = (z_1, z_2)^T$ 로 변환
→ 새로운 특징 공간 \mathbf{z} 에서 선형 분리 가능함



(a) 두 퍼셉트론을 병렬로 결합



(b) 원래 특징 공간 \mathbf{x} 를 새로운 특징 공간 \mathbf{z} 로 변환

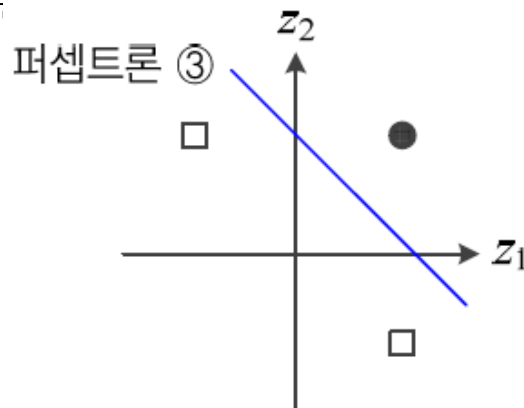
그림 3-9 특징 공간의 변환

- 사람이 수작업 특징 학습(hand-craft features learning)을 수행한 것과 유사함 ← 표현 학습

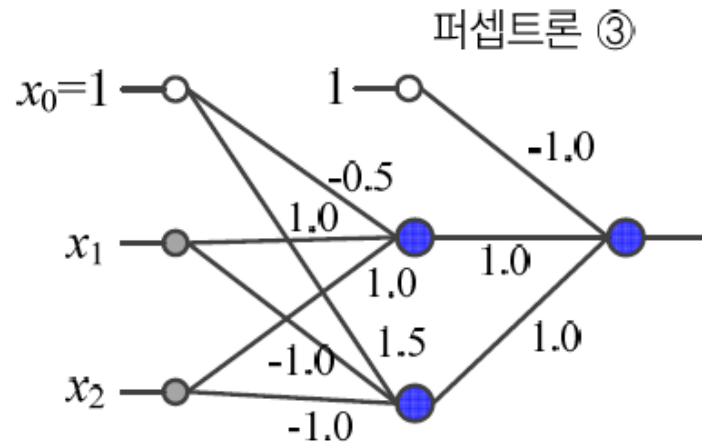
3.3.1 특징 공간 변환

■ 추가 퍼셉트론 1개를 순차 결합하면,

- 새로운 특징 공간 z 을 선형 분리를 수행하는 퍼셉트론③을 순차 결합하면, [그림 3-10(b)]의 다층 퍼셉트론이



(a) 새로운 특징 공간에서 분할



(b) 퍼셉트론 3개를 결합한 다층 퍼셉트론

그림 3-10 다층 퍼셉트론

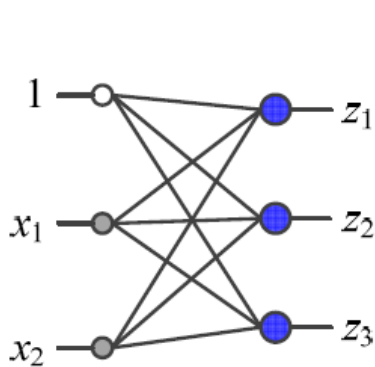
- 해당 다층 퍼셉트론은 XOR 훈련집합에 있는 4개 샘플 $\begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ 을 제대로 분류하나?

→ YES!

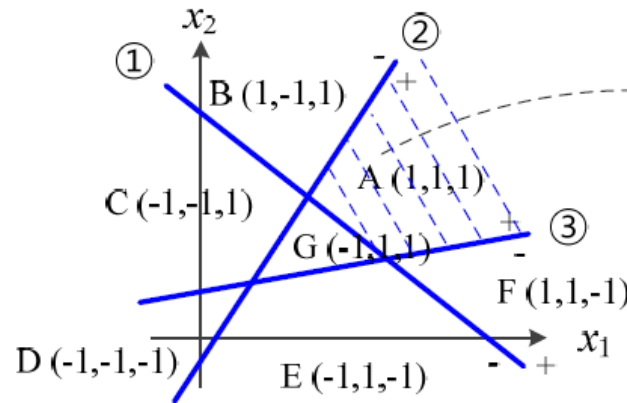
3.3.1 특징 공간 변환

■ 다층 퍼셉트론의 용량capacity

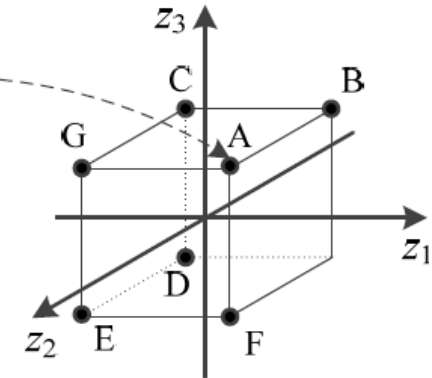
- [그림 3-11]처럼 3개 퍼셉트론을 결합한 경우
 - 2차원 공간을 7개 영역으로 나누고 각 영역을 3차원 점으로 변환
 - 계단함수를 활성화함수 τ 로 사용을 가정하였으므로 영역을 점으로 변환



(a) 퍼셉트론 3개를 결합



(b) 7개 부분공간으로 나눔




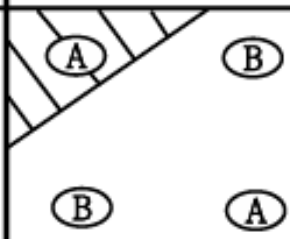
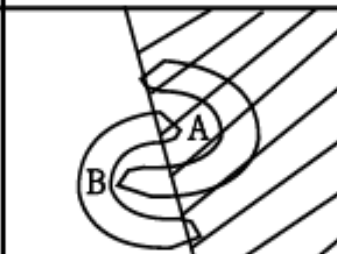
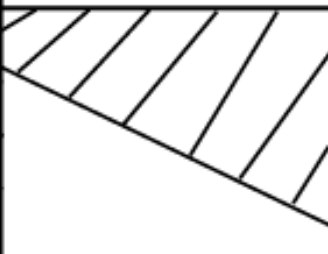

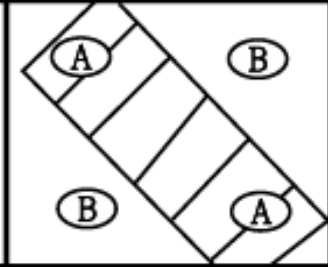
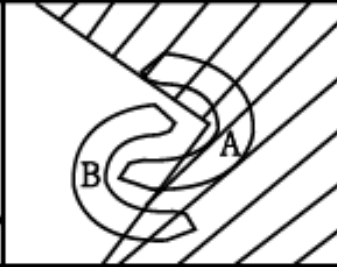
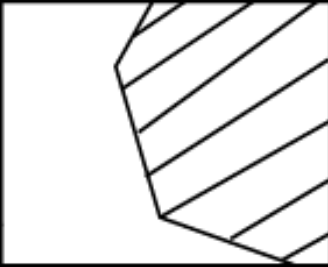

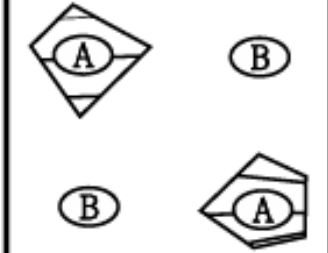
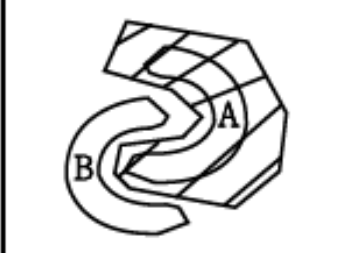
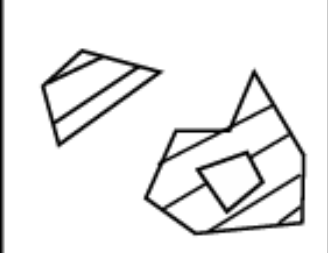
(c) 3차원 공간의 점으로 매핑

그림 3-11 퍼셉트론을 3개 결합했을 때 공간 변환

- 일반화하여, p 개 퍼셉트론을 결합하면 p 차원 공간으로 변환

3.3.1 특징 공간 변환

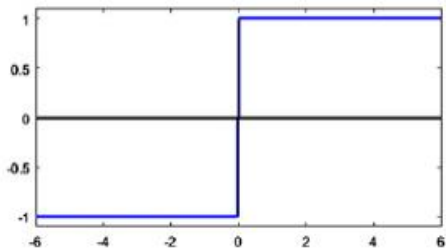
■ 다층 퍼셉트론의 용량

구 조	결정 구역	Exclusive -or	Classes with Meshed Regions	Most General Region Shapes
Single-layer j i 	Half Plane Bounded by Hyperplane			
Two-layer k j i 	Convex Open or Closed Regions			
Three-layer l k j i 	Arbitrary (Complexity limited by Number of Units)			

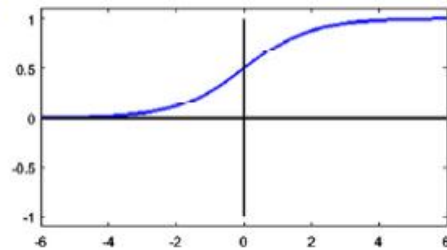
3.3.2 활성화 함수

■ 딱딱한 공간 분할과 부드러운 공간 분할

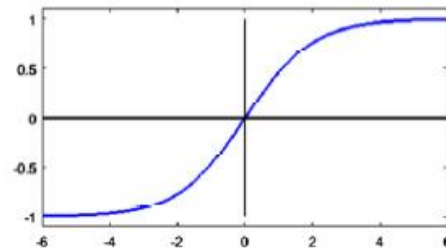
- 계단함수는 딱딱한 의사결정 → 영역을 점으로 변환
- 그외 활성화함수는 부드러운 의사결정 → 영역을 영역으로 변환



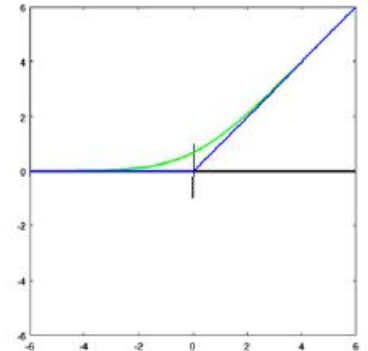
(a) 계단 함수



(b) 로지스틱 시그모이드

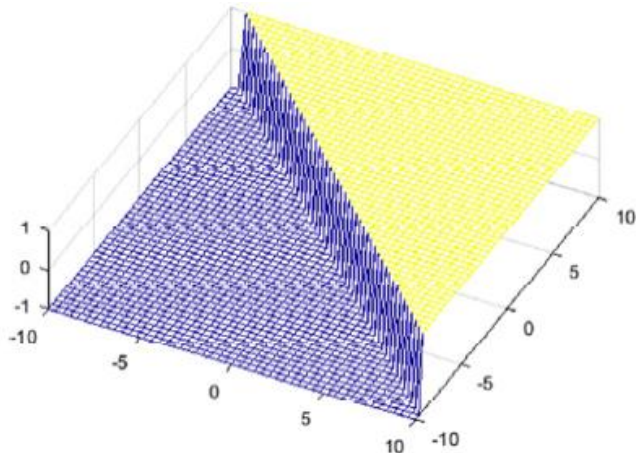


(c) 하이퍼볼릭 탄젠트 시그모이드

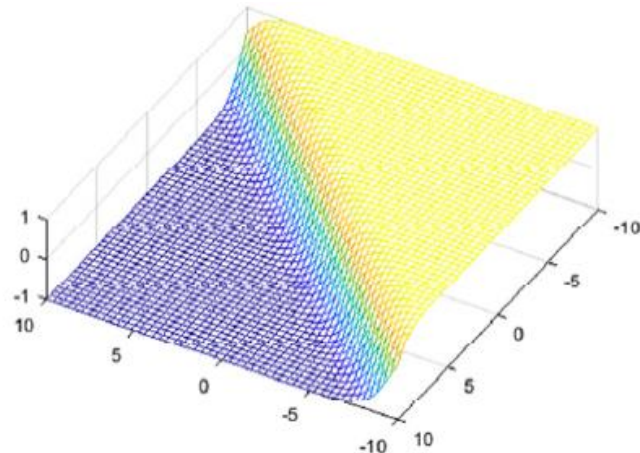


(d) softplus와 rectifier

그림 3-12 신경망이 사용하는 활성화 함수



(a) 계단함수의 딱딱한 공간 분할



(b) 로지스틱 시그모이드의 부드러운 공간 분할

그림 3-13 퍼셉트론의 공간 분할 유형

3.3.2 활성화 함수

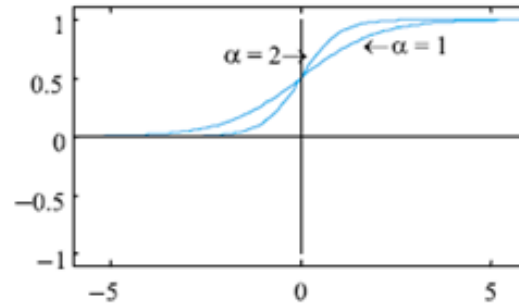
- 대표적인 비선형 함수인 S자 모양의 sigmoid를 활성화 함수로 사용

이진 시그모이드 함수:

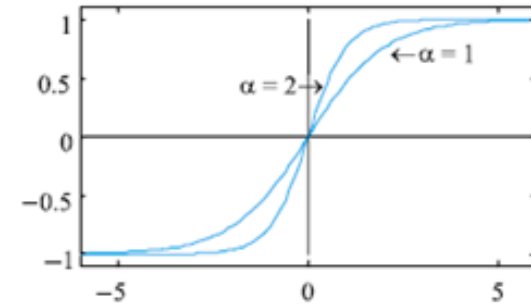
$$\left. \begin{aligned} \tau_1(x) &= \frac{1}{1 + e^{-\alpha x}} \\ \tau_1'(x) &= \alpha \tau_1(x)(1 - \tau_1(x)) \end{aligned} \right\}$$

양극 시그모이드 함수:

$$\left. \begin{aligned} \tau_2(x) &= \frac{2}{1 + e^{-\alpha x}} - 1 \\ \tau_2'(x) &= \frac{\alpha}{2} (1 + \tau_2(x))(1 - \tau_2(x)) \end{aligned} \right\}$$



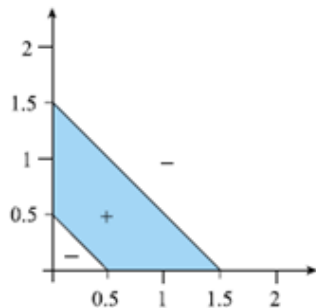
(a) 이진 시그모이드



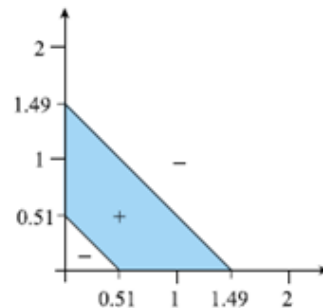
(b) 양극 시그모이드

활성 함수로 널리 사용되는 두 가지 시그모이드 함수

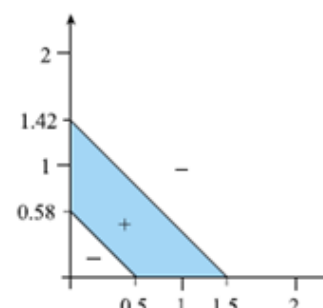
- 활성화 함수에 따른 다층 퍼셉트론의 공간 분할 능력 변화 (경성 부분 변화)



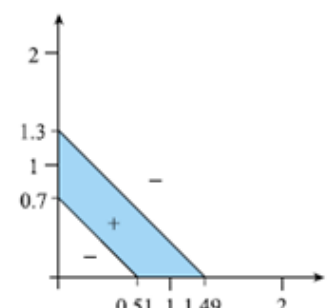
(a) 계단 함수 (양극 시그모이드 $\alpha = \infty$)



(b) 양극 시그모이드 $\alpha = 5$



(c) 양극 시그모이드 $\alpha = 3$



(d) 양극 시그모이드 $\alpha = 2.5$

활성 함수에 따른 다층 퍼셉트론의 공간 분할 능력

3.3.2 활성화함수

■ 신경망이 사용하는 다양한 활성화함수

- logistic sigmoid와 tanh는 a 가 커질수록 계단함수에 가까워짐
- 모두 1차 도함수 계산이 빠름 (특히, ReLU는 비교 연산 한 번)

표 3-1 활성화함수로 사용되는 여러 함수

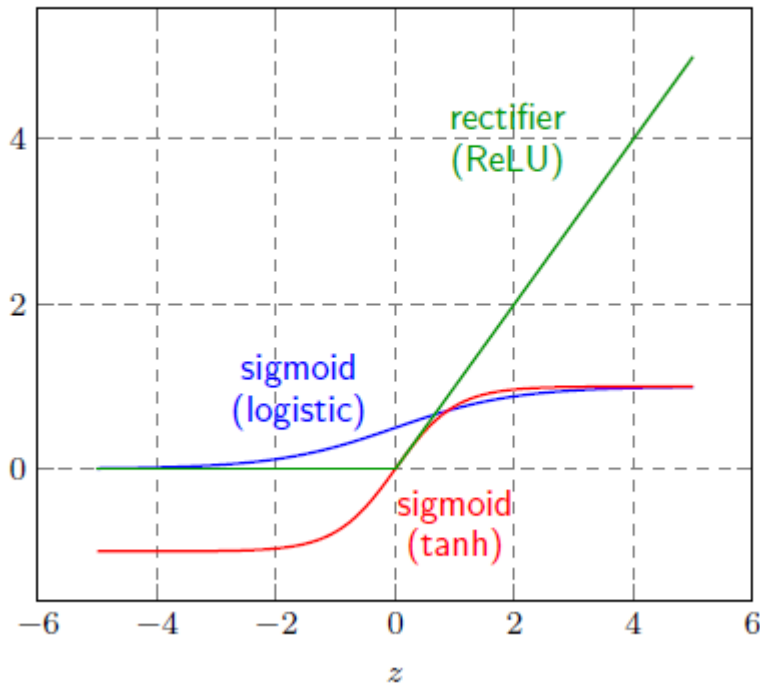
함수 이름	함수	1차 도함수	범위
계단	$\tau(s) = \begin{cases} 1 & s \geq 0 \\ -1 & s < 0 \end{cases}$	$\tau'(s) = \begin{cases} 0 & s \neq 0 \\ \text{불가} & s = 0 \end{cases}$	-1과 1
로지스틱 시그모이드	$\tau(s) = \frac{1}{1 + e^{-as}}$	$\tau'(s) = a\tau(s)(1 - \tau(s))$	(0,1)
하이퍼볼릭 탄젠트	$\tau(s) = \frac{2}{1 + e^{-as}} - 1$	$\tau'(s) = \frac{a}{2}(1 - \tau(s)^2)$	(-1,1)
소프트플러스	$\tau(s) = \log_e(1 + e^s)$	$\tau'(s) = \frac{1}{1 + e^{-s}}$	$(0, \infty)$
렉티파이어(ReLU)	$\tau(s) = \max(0, s)$	$\tau'(s) = \begin{cases} 0 & s < 0 \\ 1 & s > 0 \\ \text{불가} & s = 0 \end{cases}$	$[0, \infty)$

- 활용 예
 - 퍼셉트론은 계단함수
 - 다층 퍼셉트론은 logistic sigmoid와 tanh
 - 심층학습은 ReLU^{rectified linear activation}

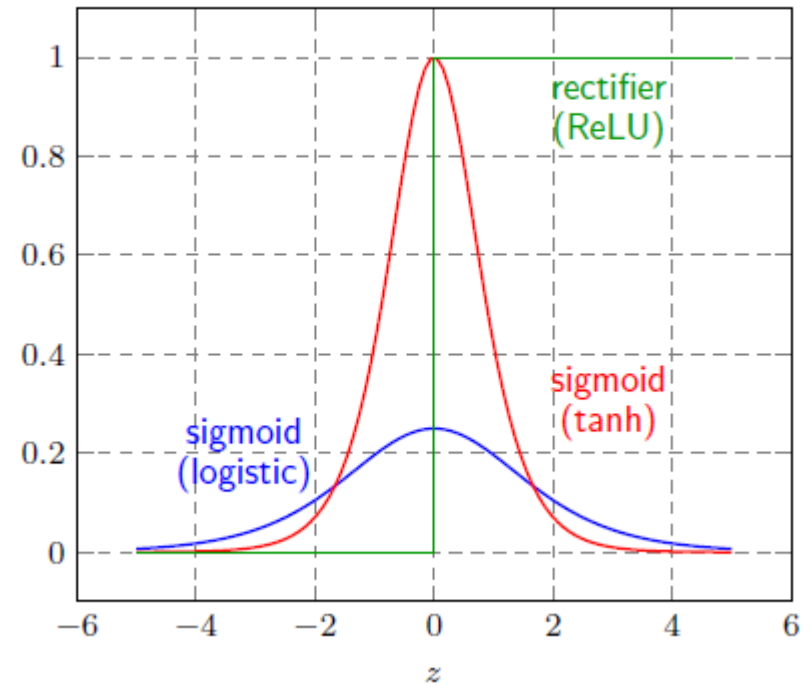
3.3.2 활성화 함수

- 일반적으로 은닉층에서 **logistic sigmoid**를 활성화 함수로 많이 사용
 - S자 모양의 넓은 포화곡선은 경사도 기반한 학습(오류 역전파)을 어렵게 함
- 깊은 신경망에서는 ReLU 활용

activation function: $g(z)$



gradient: $g'(z)$

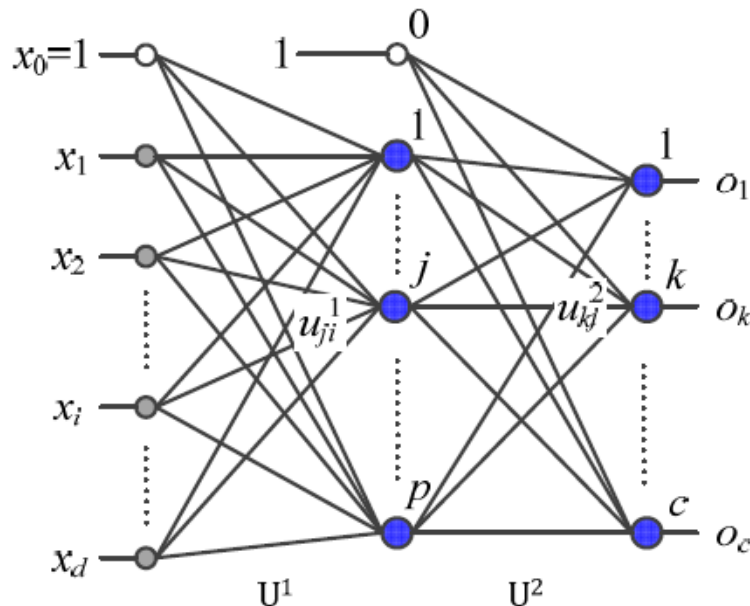


3.3.3 구조

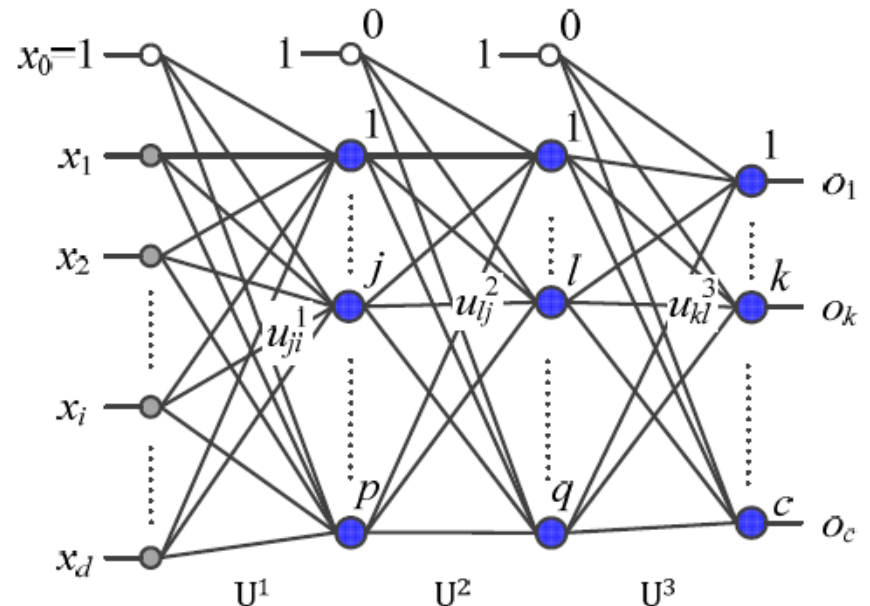
■ [그림 3-14(a)]는 입력-은닉층-출력의 2층 구조

- $d+1$ 개의 입력 노드 (d 는 특징의 개수). c 개의 출력 노드 (c 는 부류 개수)
- p 개의 은닉 노드: p 는 하이퍼 매개변수 (사용자가 정해주는 매개변수)
 - p 가 너무 크면 과잉적합, 너무 작으면 과소적합 → 5.5절의 하이퍼 매개변수 hyper-parameter 최적화

■ [그림 3-14(b)]는 입력-은닉층-은닉층-출력의 3층 구조



(a) 2층 퍼셉트론





(b) 3층 퍼셉트론

그림 3-14 다층 퍼셉트론의 구조

3.3.3 구조

■ (2층) 다층 퍼셉트론의 매개변수 (가중치)

- 입력-은닉층을 연결하는 \mathbf{U}^1 (u_{ji}^1 은 입력의 i 번째 노드를 은닉층의 j 번째 노드와 연결)

- 은닉층-출력을 연결하는 \mathbf{U}^2 (u_{kj}^2 은 은닉층의 j 번째 노드를 출력의 k 번째 노드와 연결)


2층 퍼셉트론의 가중치 행렬:

$$\mathbf{U}^1 = \begin{pmatrix} u_{10}^1 & u_{11}^1 & \cdots & u_{1d}^1 \\ u_{20}^1 & u_{21}^1 & \cdots & u_{2d}^1 \\ \vdots & \vdots & \ddots & \vdots \\ u_{p0}^1 & u_{p1}^1 & \cdots & u_{pd}^1 \end{pmatrix}, \quad \mathbf{U}^2 = \begin{pmatrix} u_{10}^2 & u_{11}^2 & \cdots & u_{1p}^2 \\ u_{20}^2 & u_{21}^2 & \cdots & u_{2p}^2 \\ \vdots & \vdots & \ddots & \vdots \\ u_{c0}^2 & u_{c1}^2 & \cdots & u_{cp}^2 \end{pmatrix} \quad (3.11)$$

- 일반화하면 u_{ji}^l 은 $l-1$ 번째 은닉층의 i 번째 노드를 l 번째 은닉층의 j 번째 노드와 연결하는 가중치
 - 입력을 0번째 은닉층, 출력을 마지막 은닉층으로 간주

3.3.4 동작

- 특징 벡터 \mathbf{x} 를 출력 벡터 \mathbf{o} 로 사상(mapping)하는 함수로 간주할 수 있음

$$\left. \begin{array}{l} \text{2층 퍼셉트론: } \mathbf{o} = \mathbf{f}(\mathbf{x}) = \mathbf{f}_2(\mathbf{f}_1(\mathbf{x})) \\ \text{3층 퍼셉트론: } \mathbf{o} = \mathbf{f}(\mathbf{x}) = \mathbf{f}_3(\mathbf{f}_2(\mathbf{f}_1(\mathbf{x}))) \end{array} \right\} \quad (3.12)$$

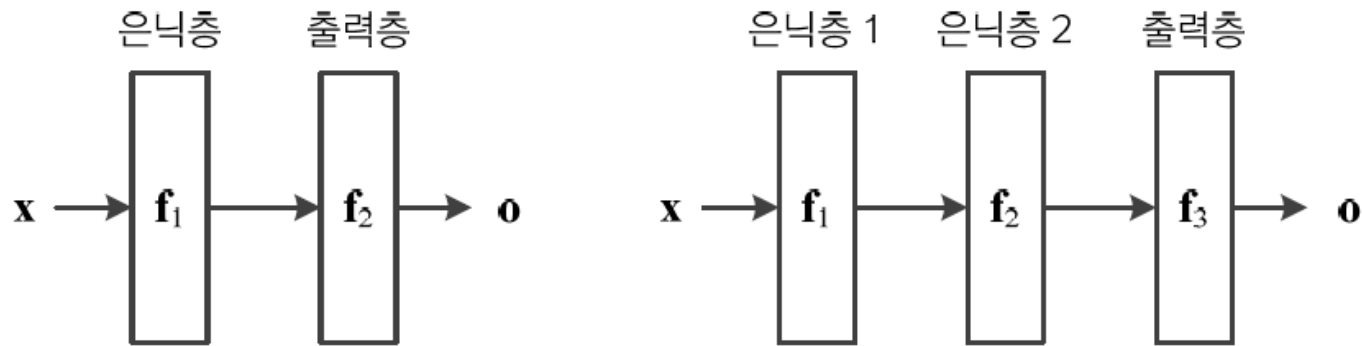


그림 3-15 다층 퍼셉트론을 간략화한 구조

- 깊은 신경망(deep neural networks)은 $\mathbf{o} = \mathbf{f}_L(\cdots \mathbf{f}_2(\mathbf{f}_1(\mathbf{x})))$, $L \geq 4$

← 심층학습(4장)을 통해 학습됨

3.3.4 동작

- 노드가 수행하는 연산을 구체적으로 쓰면,

j 번째 은닉 노드의 연산:

$$z_j = \tau(\text{zsum}_j), j = 1, 2, \dots, p$$

이때 $\text{zsum}_j = \mathbf{u}_j^1 \mathbf{x}$ 이고 $\mathbf{u}_j^1 = (u_{j0}^1, u_{j1}^1, \dots, u_{jd}^1)$, $\mathbf{x} = (1, x_1, x_2, \dots, x_d)^T$

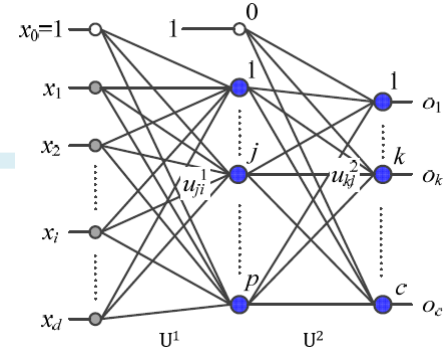
- \mathbf{u}_j^1 은 j 번째 은닉 노드에 연결된 가중치 벡터 (식 (3.11)의 \mathbf{U}^1 의 j 번째 행)

k 번째 출력 노드의 연산:

$$o_k = \tau(\text{osum}_k), k = 1, 2, \dots, c$$

이때 $\text{osum}_k = \mathbf{u}_k^2 \mathbf{z}$ 이고 $\mathbf{u}_k^2 = (u_{k0}^2, u_{k1}^2, \dots, u_{kp}^2)$, $\mathbf{z} = (1, z_1, z_2, \dots, z_p)^T$

- \mathbf{u}_k^2 은 k 번째 출력 노드에 연결된 가중치 벡터 (식 (3.11)의 \mathbf{U}^2 의 k 번째 행)



(3.13)

- 다층 퍼셉트론의 동작을 행렬로 표기하면,

$$\mathbf{o} = \tau(\mathbf{U}^2 \tau_h(\mathbf{U}^1 \mathbf{x}))$$

(3.15)

3.3.4 동작

■ 은닉층은 특징 추출기

- 은닉층은 특징 벡터를 분류에 더 유리한 새로운 특징 공간으로 변환
- 현대 기계학습에서는 특징 학습이라 feature learning 혹은 data-driven features 부름
(심층학습은 더 많은 층을 거쳐 계층화 된 특징 학습을 함)

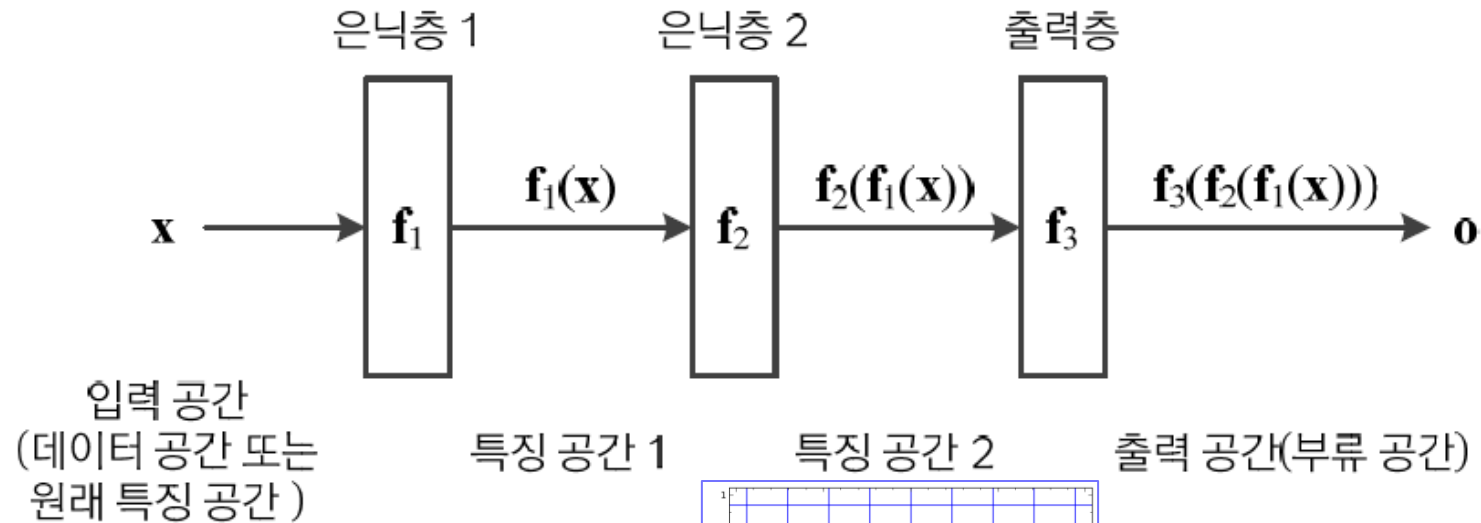
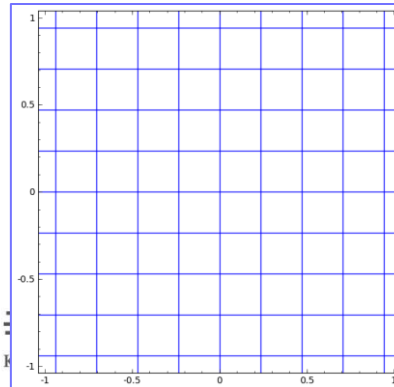


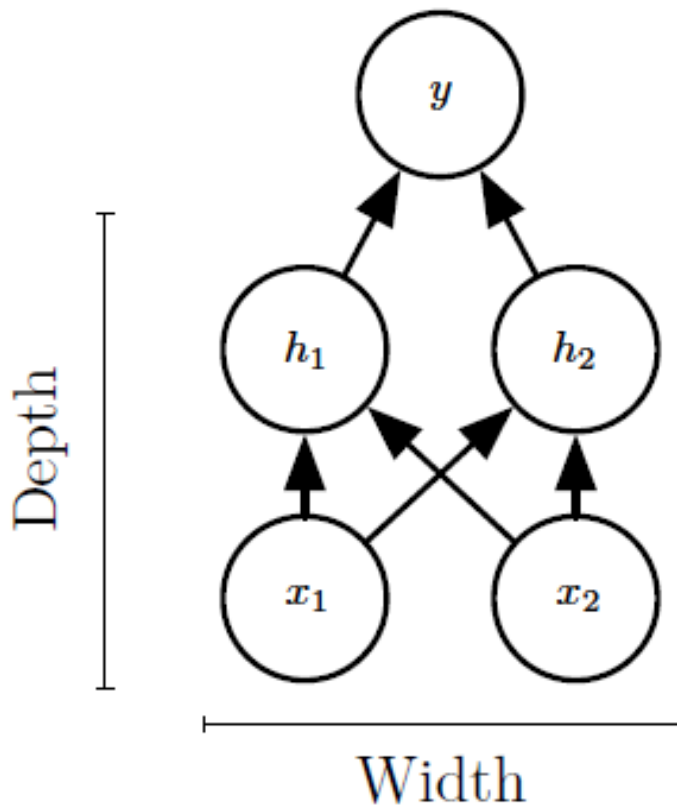
그림 3-16 특징 추출기로서의 은닉층



- 은닉층을 통한
특징공간의 변환
- 행렬 곱: 회전
 - 편향: 이동
 - 비선형 함수: 왜곡

3.3.3 구조

■ 기본 구조



■ 범용적 근사 이론 universal approximation theorem

- 하나의 은닉층은 함수의 근사를 표현

→ 다층 퍼셉트론도 공간을 변환하는 근사 함수

■ 얇은 은닉층의 구조

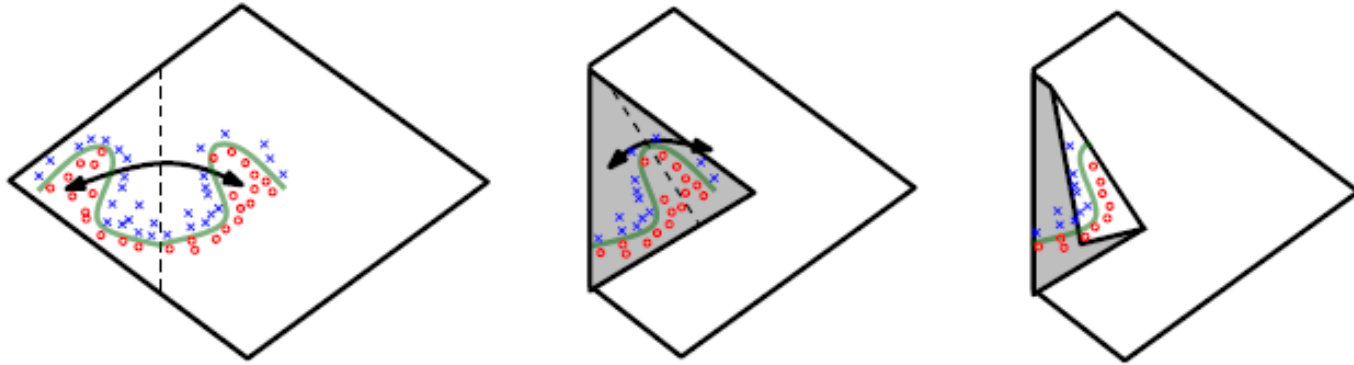
- 지수적으로 더 넓은 폭^{width}이 필요할 수 있음
- 더 과잉적합 되기 쉬움

→ 일반적으로 깊은 은닉층의 구조가 좋은 성능을 가짐

3.3.3 구조

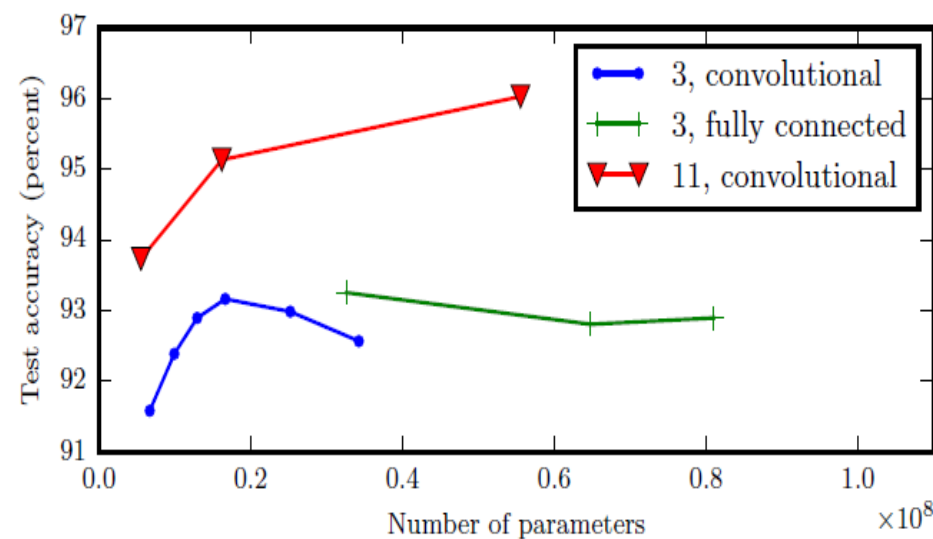
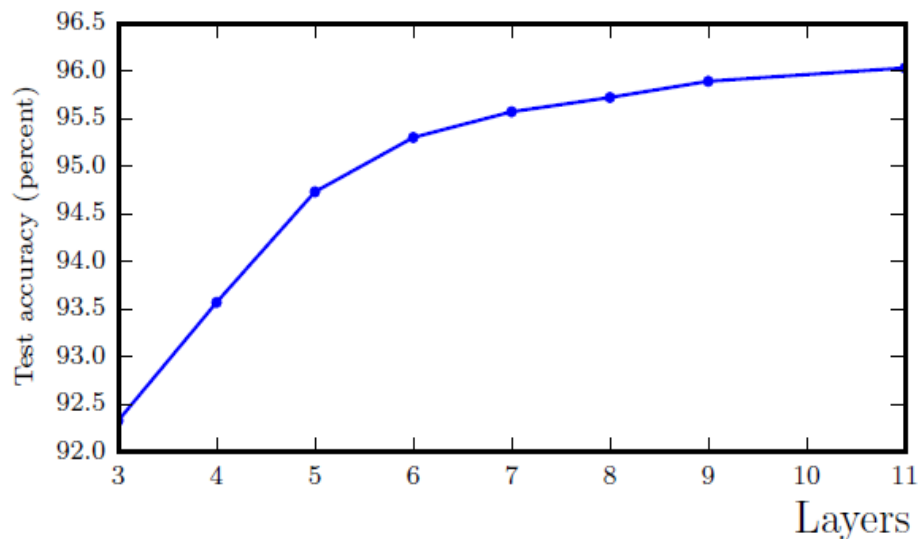
■ 은닉층의 깊이에 따른 이점

- 지수의 표현 exponential representation



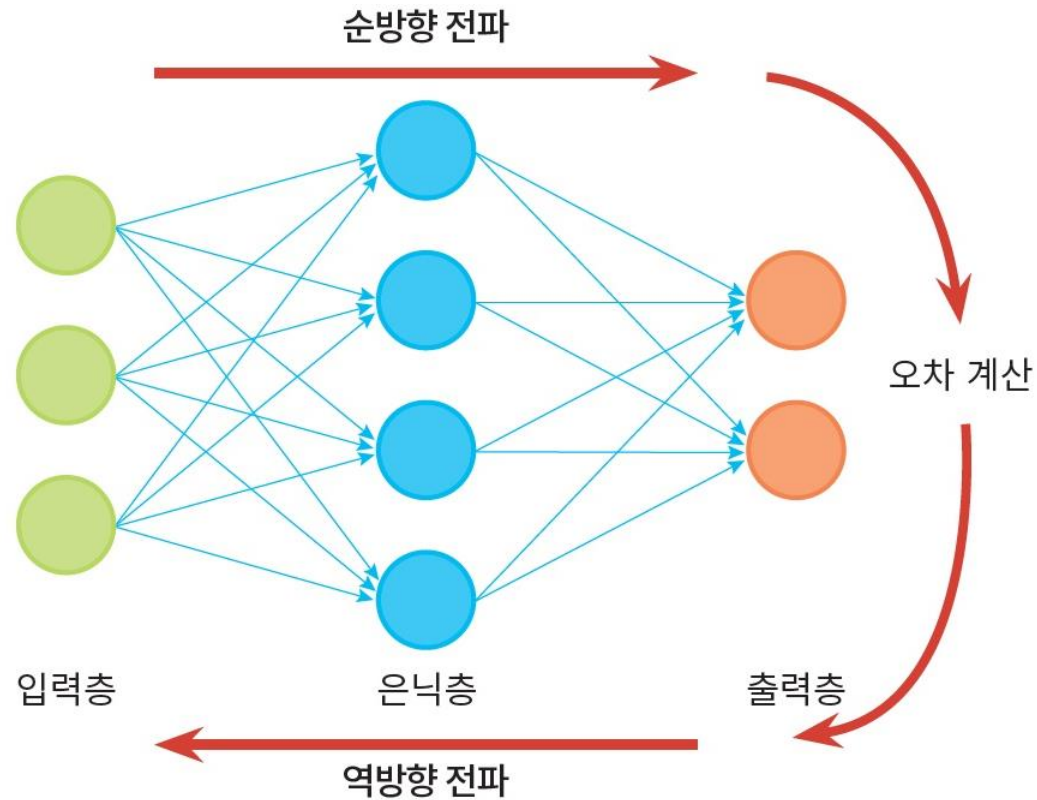
- 각 은닉층은 입력 공간을 어디서 접을지 지정 → 지수적으로 많은 선형적인 영역 조각들

- 성능 향상



3.6 다층 퍼셉트론에 의한 인식

■ 다층 퍼셉트론 학습 과정



3.6 다층 퍼셉트론에 의한 인식

■ 예측 단계

- 학습을 마친 후 현장 설치하여 사용 (또는 테스트 집합으로 성능 테스트)

알고리즘 3-7 다층 퍼셉트론을 이용한 인식

입력: 테스트 샘플 \mathbf{x} // 신경망의 가중치 \mathbf{U}^1 과 \mathbf{U}^2 는 이미 설정되었다고 가정함.

출력: 부류 y

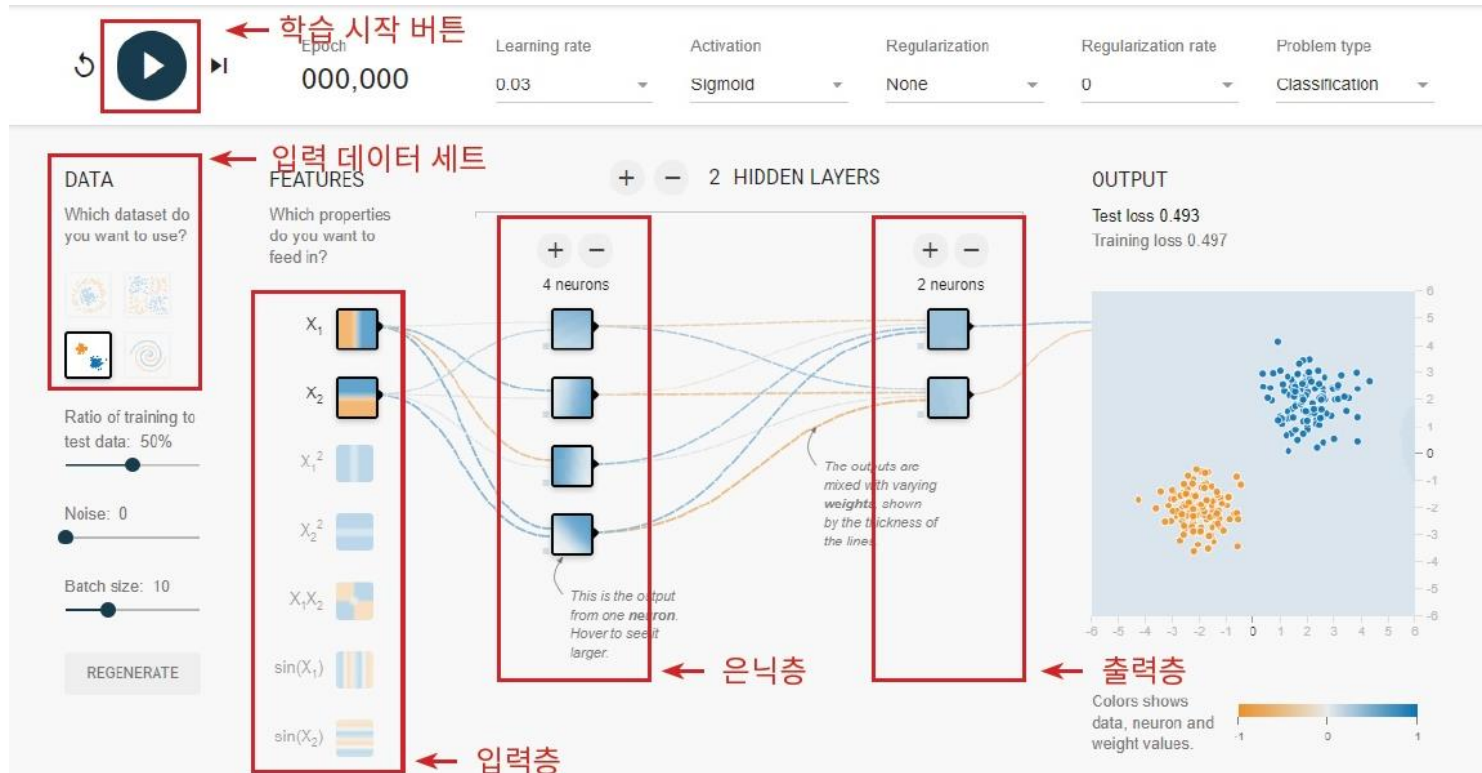
- 1 $\mathbf{x} = (x_0, x_1, x_2, \dots, x_d)^T$ 로 확장하고, x_0 과 z_0 을 1로 설정한다.
- 2 $\mathbf{zsum} = \mathbf{U}^1 \mathbf{x}$
- 3 $\tilde{\mathbf{z}} = \tau(\mathbf{zsum})$ // 식 (3.13)
- 4 $\mathbf{osum} = \mathbf{U}^2 \tilde{\mathbf{z}}$
- 5 $\mathbf{o} = \tau(\mathbf{osum})$ // 식 (3.14)
- 6 \mathbf{o} 에서 가장 큰 값을 가지는 노드에 해당하는 부류 번호를 y 에 대입한다.

- 6번째 줄을 수식으로 표현하면 $y = \underset{k}{\operatorname{argmax}} o_k$
- 전방 계산 한번만 사용하므로 빠름

3.6 다층 퍼셉트론에 의한 인식

■ 다층 퍼셉트론 시각화

- <https://playground.tensorflow.org>



3.7 다층 퍼셉트론의 특성

3.7.1 오류 역전파 알고리즘의 빠른 속도

■ 연산 복잡도 time complexity 비교

표 3-2 전방 계산과 오류 역전파 과정이 사용하는 연산 횟수

과정	수식	덧셈	곱셈
전방 계산	식 (3.13)	dp	dp
	식 (3.14)	pc	pc
오류 역전파	식 (3.22)	c	c
	식 (3.23)		cp
	식 (3.24)	cp	cp
	식 (3.25)		dp
	식 (3.21)	$cp+dp$	$cp+dp$

- 오류 역전파: 전방 계산 대비 약 1.5~2배의 시간 소요 ← 비교적 빠름 (연쇄법칙)
 - c : 분류 수, d : 특징 차원, p : 은닉층 차원
- 학습 알고리즘은 오류 역전파 반복하여 점근적 시간 복잡도는 $\theta((cp + dp)nq)$
 - n : 훈련집합의 크기, q : 세대^{epoch} 수

3.7.2 모든 함수를 정확하게 근사할 수 있는 능력

■ Hornik 주장[1989]

- 은닉층을 하나만 가진 다층 퍼셉트론은 범용근사자 universal approximator

“... standard multilayer feedforward network architectures using arbitrary squashing functions can approximate virtually any function of interest to any desired degree of accuracy, provided sufficiently many hidden units are available. ... 은닉 노드가 충분히 많다면, 포화함수(활성함수)로 무엇을 사용하든 표준 다층 퍼셉트론은 어떤 함수라도 원하는 정확도만큼 근사화할 수 있다.”

3.7.3 성능 향상을 위한 경험의 중요성

■ 순수한 최적화 알고리즘으로는 높은 성능 불가능

- 데이터 희소성, 잡음, 미숙한 신경망 구조 등 이유
- **성능 향상**을 위한 다양한 경험^{heuristics}을 개발하고 공유함
→ 예) 『Neural Networks: Tricks of the Trade』 [2012]

■ 신경망의 경험적 개발에서 중요 쟁점

- **아키텍처**: 은닉층과 은닉 노드의 개수를 정해야 한다. 은닉층과 은닉 노드를 늘리면 신경망의 용량은 커지는 대신, 추정할 매개변수가 많아지고 학습 과정에서 과잉적합할 가능성이 커진다. 1.6절에서 소개한 바와 같이 현대 기계 학습은 복잡한 모델을 사용하되, 적절한 규제 기법을 적용하는 경향이 있다.
- **초깃값**: [알고리즘 3-4]의 라인 1에서 가중치를 초기화한다. 보통 난수를 생성하여 설정하는데, 값의 범위와 분포가 중요하다. 이 주제는 5.2.2절에서 다룬다.
- **학습률**: 처음부터 끝까지 같은 학습률을 사용하는 방식과 처음에는 큰 값으로 시작하고 점점 줄이는 적응적 방식이 있다. 5.2.4절에서 여러 가지 적응적 학습률 기법을 소개한다.
- **활성함수**: 초창기 다층 퍼셉트론은 주로 로지스틱 시그모이드나 tanh 함수를 사용했는데, 은닉층의 개수를 늘림에 따라 그레이디언트 소멸과 같은 몇 가지 문제가 발생한다. 따라서 깊은 신경망은 주로 ReLU 함수를 사용한다. 5.2.5절에서 여러 가지 ReLU 함수를 설명한다.