

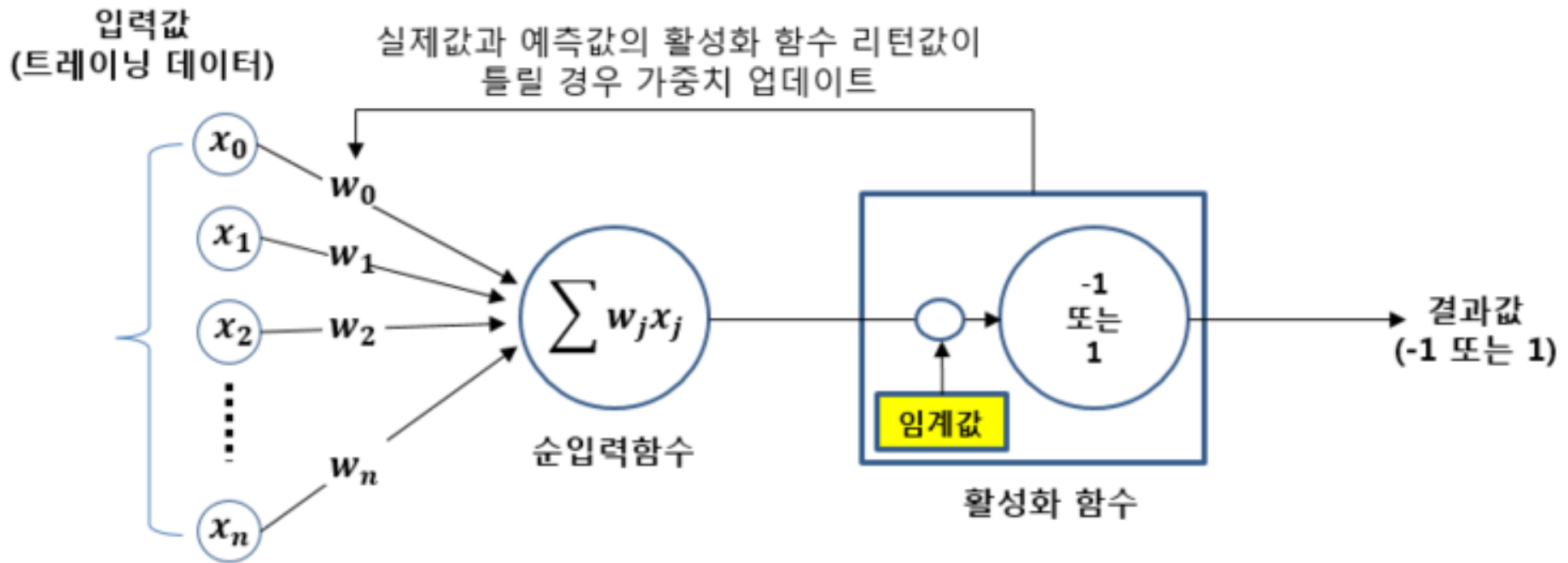
Chapter 2,3

아꿈사 박주희

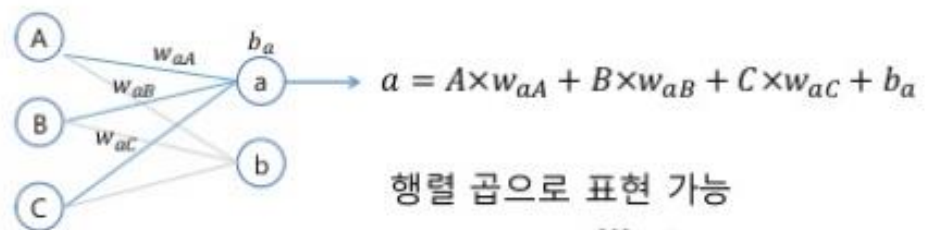
<http://cafe.naver.com/architect1>

유닛의 출력

- 앞먹인 신경망(Feed forward neural network) = 다층 퍼셉트론 (multi-layer perceptron)
= Dense Network = Fully connected neural network



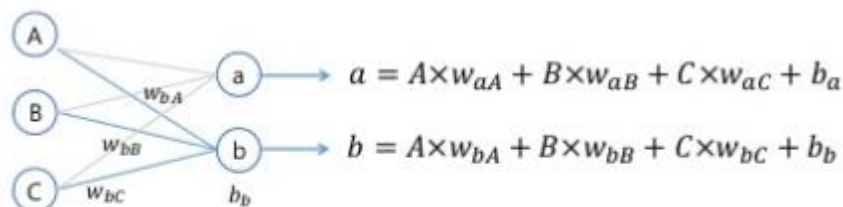
유닛의 출력



$$a = A \times w_{aA} + B \times w_{aB} + C \times w_{aC} + b_a$$

행렬 곱으로 표현 가능

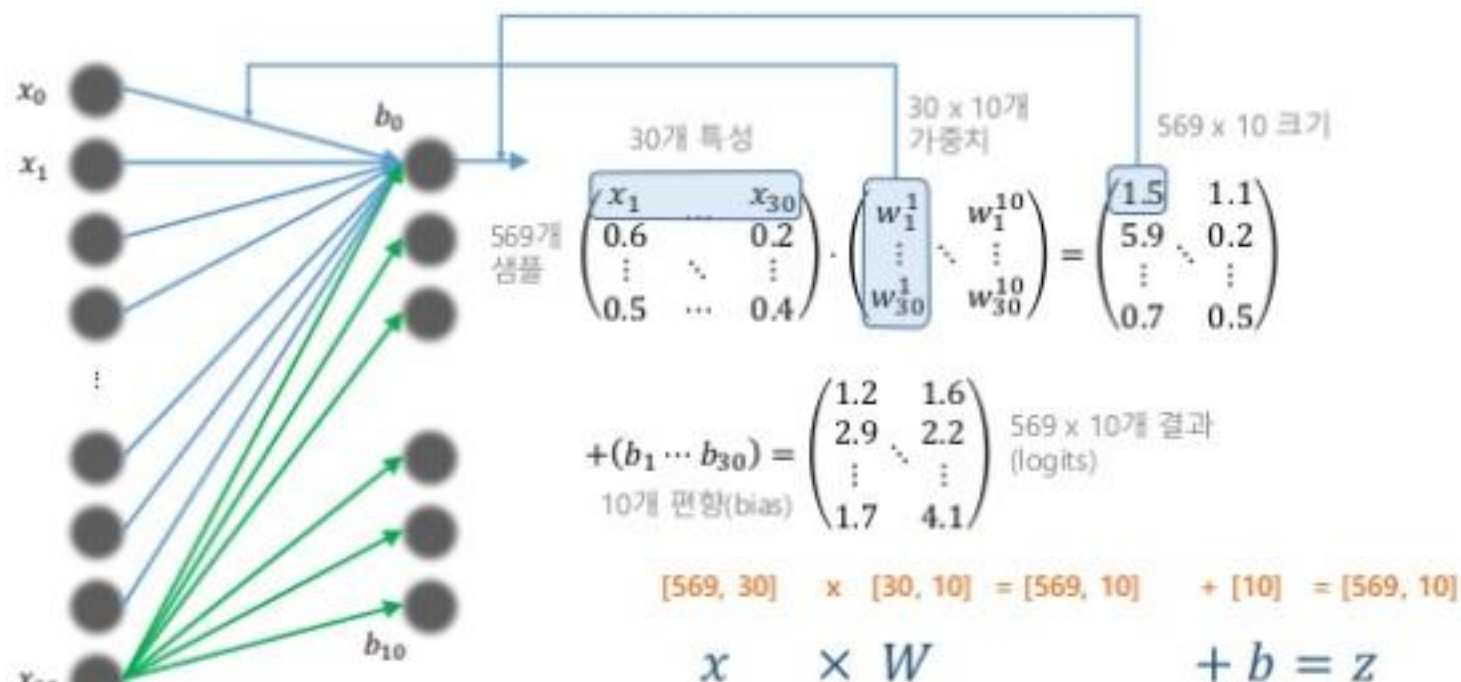
$$(A \ B \ C) \times \begin{pmatrix} w_{aA} \\ w_{aB} \\ w_{aC} \end{pmatrix} + (b_a) = (a)$$



$$a = A \times w_{aA} + B \times w_{aB} + C \times w_{aC} + b_a$$

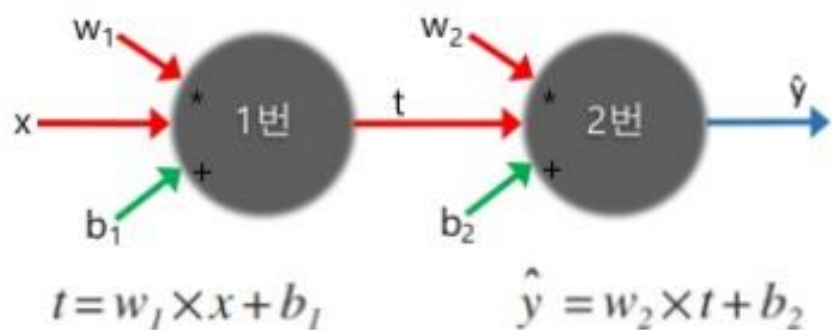
$$b = A \times w_{bA} + B \times w_{bB} + C \times w_{bC} + b_b$$

$$\boxed{(A \ B \ C)} \times \begin{pmatrix} w_{aA} & w_{bA} \\ w_{aB} & w_{bB} \\ w_{aC} & w_{bC} \end{pmatrix} + (b_a \ \boxed{b_b}) = (a \ \boxed{b})$$



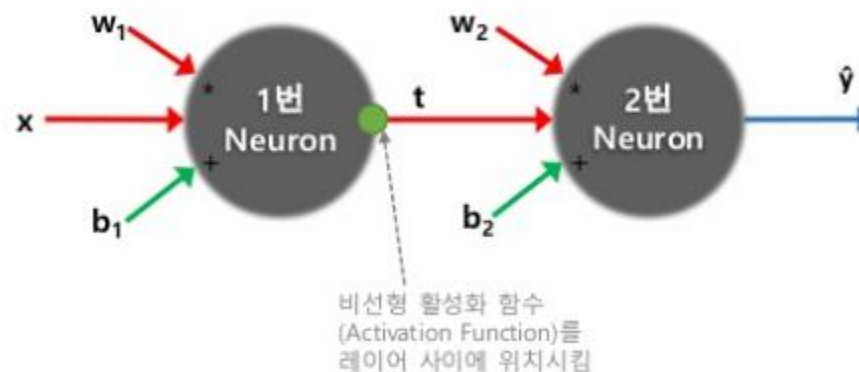
활성화 함수

- 활성화 함수(activation function)는 입력신호의 총합을 출력 신호로 변환하는 함수를 일반적으로 일컬음.
- MLP에서는 임계값을 경계로 출력을 바꾸는 역할을 함.



$$\begin{aligned}\hat{y} &= w_2 \times t + b_2 \\ &= w_2 \times (w_1 \times x + b_1) + b_2 \\ &= (w_2 \times w_1) \times x + (b_1 + b_2) = w \times x + b\end{aligned}$$

비선형 문제를
학습하기 어려움



활성화 함수

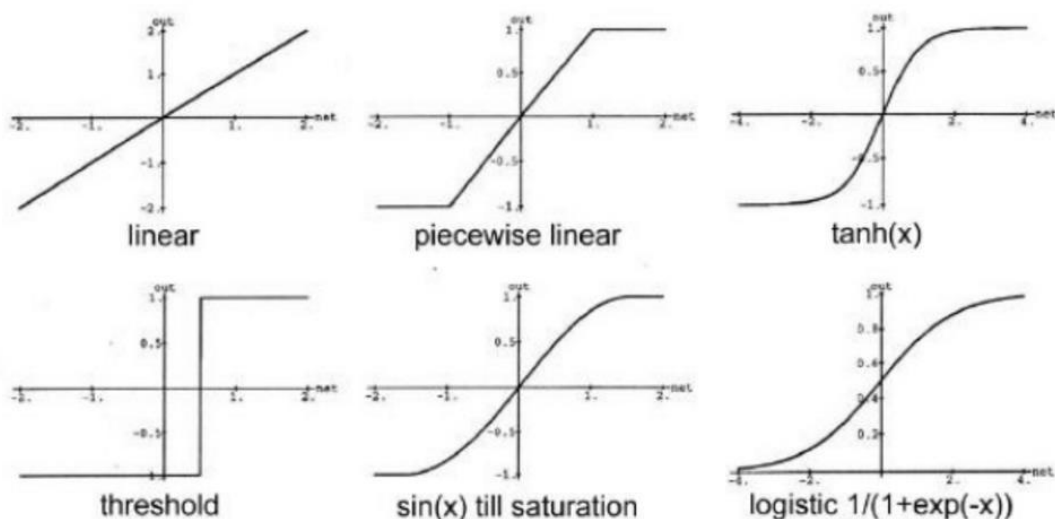
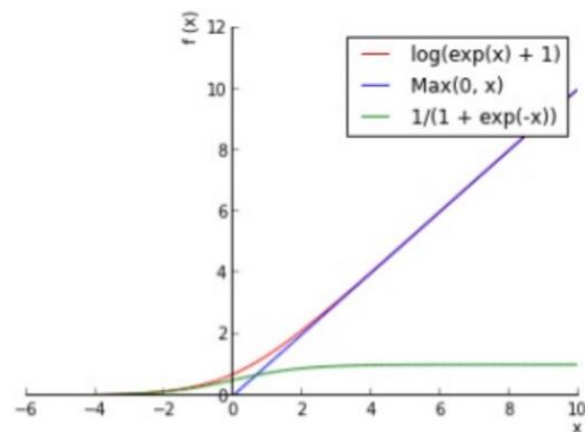


Figure 3.2: Neural network activation functions. Note the characteristic 'sigmoid' or S-shape. (A. Graves, 2012)



Rectified Linear Unit

$$\text{ReLU}(x) = \max(x, 0)$$

[장점]

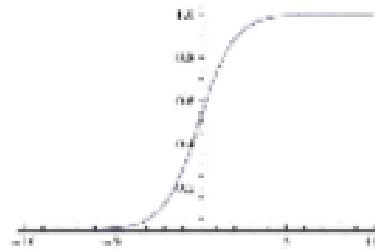
- Hidden unit에 sparsity가 나타난다
- Gradient vanishing 문제가 없다
- 계산이 단순해져 학습이 빠르면서 성능 향상에도 일조

- 입력값을 처리하여 ON 또는 OFF 출력. 보통 출력값에 상, 하한을 두려 함
- 신경망에서는 '미분 가능한' smooth한 S자 모양 함수를 적용
- 최근 딥러닝 모델에서는 ReLU를 선호
- 순환신경망 내부에서는 sigmoid와 tanh 함수를 적용함

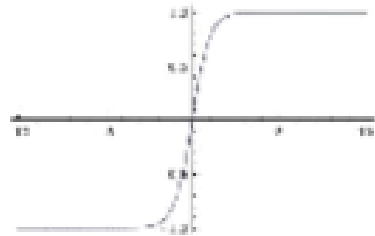
Activation Functions

Sigmoid

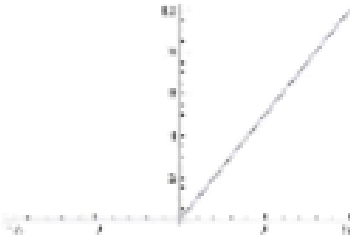
$$\sigma(x) = 1/(1 + e^{-x})$$



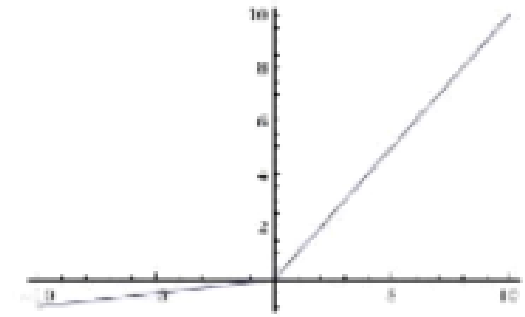
tanh tanh(x)



ReLU max(0,x)



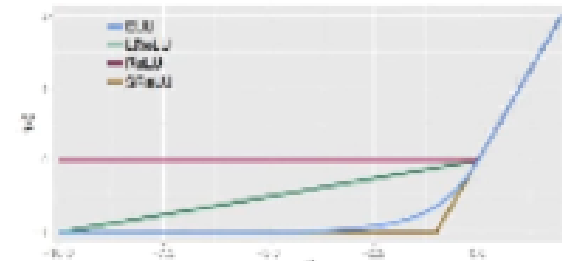
Leaky ReLU max(0.1x, x)



Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha (\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

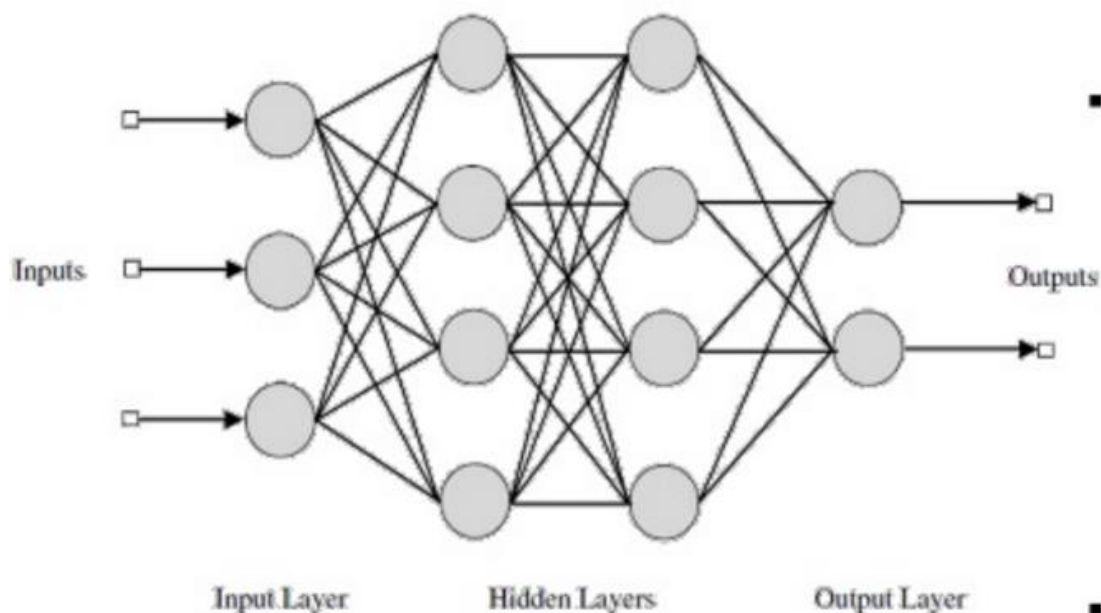
ELU



maxout	ReLU	VLeLU	tanh	Sigmoid
93.94	92.11	92.97	89.28	n/c
93.78	91.74	92.40	89.48	n/c
—	91.93	93.09	—	n/c
91.75	90.63	92.27	89.82	n/c
n/c†	90.91	92.43	89.54	n/c

다층 신경망

▪ Feedforward Neural Network





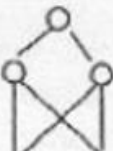
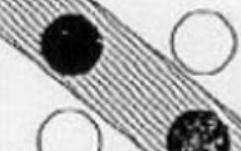








- Input Layer (입력층)
 - ✓ 입력 데이터를 받아들인다.
 - ✓ 입력층의 노드(뉴런) 수는 입력 데이터의 특성 개수와 일치 한다.
- Hidden Layer (은닉층)
 - ✓ 은닉층의 뉴런 수와 은닉층의 개수는 신경망 설계자의 직관과 경험에 의존
 - ✓ 은닉층의 뉴런 수가 너무 많으면 Overfitting이 발생, 너무 적으면 충분히 표현하지 못함.
 - ✓ 은닉층의 개수가 너무 많으면 비효율적 (예, 은닉층 수를 2배로 늘리면 컴퓨팅 시간은 400% 증가하지만 정확성은 10%로 증가함)
- Output Layer (출력층)
 - ✓ 해결하고자 하는 문제의 성격 (예, 필기체 숫자를 인식한다면 0에서 9 까지 10개 노드로 선정)

다층 신경망

■ 은닉층 (Hidden Layer)

- Hidden Layer을 추가하면 선형분리 불가능 (linearly inseparable) 문제를 풀 수 있다.
 - ✓ 하지만, 다층 퍼셉트론 발견 당시에는 다층 퍼셉트론을 학습시킬 수학적 모델이 없었다.
 - ✓ 은닉층의 개수가 늘어나면 더욱 복잡한 문제를 풀 수 있으나 컴퓨터 계산량이 늘어난다.
- Hidden Layer의 역할
 - ✓ 앞 단계에서 받은 데이터(신호)를 필터링해서 좀 더 구체화 한 후 다음 단계 층으로 전달
 - ✓ 각 은닉층마다 몇 개의 뉴런(노드)이 최적인지는 문제에 따라 다르다.
- 신경망 층에 따라 Decision Boundary는 더 정확

Structure	Description of decision regions	Exclusive-OR problem	Classes with meshed regions	General region shapes
 Single layer	Half plane bounded by hyperplane			
 Two layer	Arbitrary (complexity limited by number of hidden units)			
 Three layer	Arbitrary (complexity limited by number of hidden units)			

[2차원 평면에 분포된 데이터 세트를 은닉층 개수에 따라 분리 비교]

출력층의 설계와 오차함수

- 학습 : 신경망의 출력이 최대한 목표값(바람직한 출력값)에 가까워지도록 가중치를 조정하는 것을 일컫음.
- 오차함수 : 신경망 출력과 목표값의 차이(거리)를 측정할 때의 척도를 일컫음.
- 문제의 유형에 따라 사용되는 활성화 함수 및 오차함수의 종류

문제 유형	출력층 활성화 함수	오차 함수
회귀	항등 함수	제곱오차
이진 분류	로지스틱 함수	$c(H(x), y) = -y \log H(x) - (1-y) \log(1-H(x))$
다클래스 분류	소프트맥스 함수	교차 엔트로피

출력층의 설계와 오차함수 - 회귀

- 회귀 : 출력이 연속값을 갖는 함수를 대상으로 훈련 데이터를 잘 재현하는 함수를 찾는 것을 말한다.

이때 목적으로 하는 함수와 같은 치역을 갖는 함수를 출력층의 활성화 함수로 골라야 한다.

오차 함수로는 평균제곱 오차(MSE)를 사용한다.

- 평균 제곱 오차

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2$$

출력층의 설계와 오차함수 - 이진분류

- 조건부 확률 : 어떤 조건하에서 발생할 확률
- 정의 : $P(B|A) = P(A,B) / P(A)$ (여기서, $P(A) > 0$)
 - $P(B|A)$: A 조건 하에 B가 일어날 조건부 확률 (Conditional Probability)
 - $P(A,B) = P(AB) = P(A \cap B)$: 함께 일어날 결합 확률 (Joint Probability)
- 예) $S = \{1,2,3,4,5,6\}$, $A = \{2,4,6\}$, $B = \{4,5,6\}$ 이면,
 - $P(S) = 6/6 = 1$, $P(A) = 3/6 = 1/2$, $P(B) = 3/6 = 1/2$
 - . $P(B|A) = P(A,B) / P(A) = \{4,6\} / \{2,4,6\} = 2/3$
 - . $P(A|B) = P(A,B) / P(B) = \{4,6\} / \{4,5,6\} = 2/3$

출력층의 설계와 오차함수 - 이진분류

- 조건부 확률 : 어떤 조건하에서 발생할 확률
- 정의 : $P(B|A) = P(A,B) / P(A)$ (여기서, $P(A) > 0$)
 - $P(B|A)$: A 조건 하에 B가 일어날 조건부 확률 (Conditional Probability)
 - $P(A,B) = P(AB) = P(A \cap B)$: 함께 일어날 결합 확률 (Joint Probability)
 - 사건 A,B가 서로 독립적이면, $P(A|B) = P(A)$ 또는 $P(B|A) = P(B)$
 - 사건 A,B가 서로 종속적이면 $P(A|B) = P(A,B) / P(B)$ 또는 $P(B|A) = P(B,A) / P(A)$
 - 예) $S = \{1,2,3,4,5,6\}$, $A = \{2,4,6\}$, $B = \{4,5,6\}$ 이면,
 $P(S) = 6/6 = 1$, $P(A) = 3/6 = 1/2$, $P(B) = 3/6 = 1/2$
. $P(B|A) = P(A,B) / P(A) = \{4,6\} / \{2,4,6\} = 2/3$
. $P(A|B) = P(A,B) / P(B) = \{4,6\} / \{4,5,6\} = 2/3$

사전 확률 : 시행 순서 A, B로 구성되는 조건부 확률 : $P(B|A)$

사후 확률 : 시행 순서 A, B일 때 B가 알려져 있을 때 A가 발생할 확률 $P(A|B)$

출력층의 설계와 오차함수 - 이진분류

- 베이지 정리: 사후확률(posteriori)은 사전확률(priori) 및 조건부확률(likelihood)로 구할 수 있다.
- $P(A,B) = P(A|B) P(B) = P(B|A) P(A)$ 이므로 $P(A|B) = P(B|A) P(A) / P(B)$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

$$P(\omega_i | \mathbf{x}) = \frac{P(\mathbf{x} | \omega_i) P(\omega_i)}{P(\mathbf{x})} = \frac{\text{우도} \times \text{사전확률}}{P(\mathbf{x})}$$

- ω_i : 분류 범주/분류 영역/카테고리/클래스 등
- \mathbf{x} : 관측 벡터

출력층의 설계와 오차함수 - 이진분류

- 베이지 정리: 사후확률(posteriori)은 사전확률(priori) 및 조건부확률(likelihood)로 구할 수 있다.
- $P(A,B) = P(A|B) P(B) = P(B|A) P(A)$ 이므로 $P(A|B) = P(B|A) P(A) / P(B)$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

$$P(\omega_i | \mathbf{x}) = \frac{P(\mathbf{x} | \omega_i) P(\omega_i)}{P(\mathbf{x})} = \frac{\text{우도} \times \text{사전확률}}{P(\mathbf{x})}$$

- ω_i : 분류 범주/분류 영역/카테고리/클래스 등
- \mathbf{x} : 관측 벡터

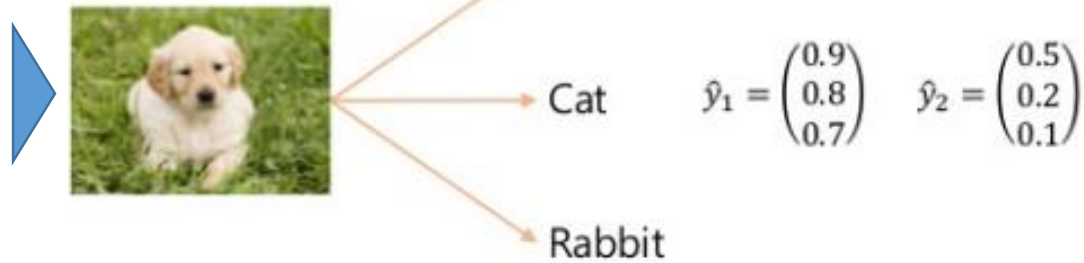
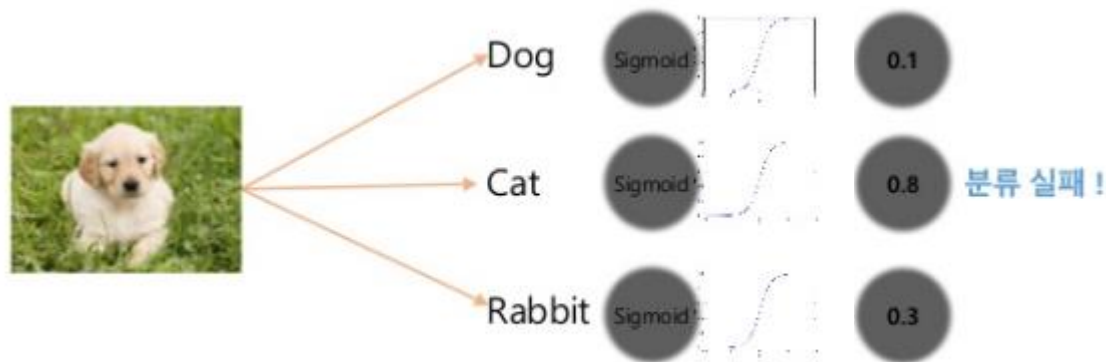
출력층의 설계와 오차함수 - 이진분류

- $p(d|x) = p(d=1|x)^d p(d=0|x)^{(1-d)}$
- $p(d=1|x) = y(x;w)$, $p(d=0|x)$ 는 $1 - y(x;w)$
- 결국 w 의 데이터에 대한 likelihood를 가장 큰 것을 고르는 것.

이진분류 사후 확률

- $P(d = 1 | x) = p(x, d=1) / (p(x, d=0) + p(x, d=1))$
결국 $u = \log p(x, d=1)/p(x, d=0)$

출력층의 설계와 오차함수 - 다클래스분류



$z = w \times x + b$

Dog $e^{z_{dog}}$

Cat $e^{z_{cat}}$

Rabbit $e^{z_{rabbit}}$

$$\hat{y}_{dog} = \frac{e^{z_{dog}}}{e^{z_{dog}} + e^{z_{cat}} + e^{z_{rabbit}}} = \frac{e^{z_{dog}}}{\sum_{i=1}^n e^{z_i}}$$

$$\hat{y} = \begin{pmatrix} \hat{y}_{dog} \\ \hat{y}_{cat} \\ \hat{y}_{rabbit} \end{pmatrix}$$

SOFTMAX

① $0 \sim 1$
② $\sum = 1$

Y $\begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$ $S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$ $\begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$

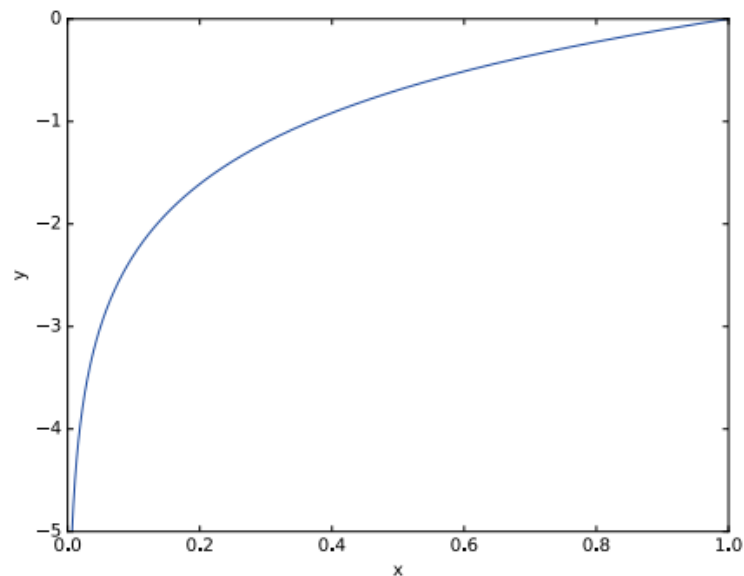
SCORES \longrightarrow PROBABILITIES

출력층의 설계와 오차함수 - 다클래스분류

- 교차 엔트로피 오차
- y_k 는 신경망의 출력
- t_k 는 정답 레이블(원-핫 인코딩)

$$E = -\sum_k t_k \log y_k$$

그림 4-3 자연로그 $y = \log x$ 의 그래프



- 결국 정답 레이블에 해당하는 신경망 출력이 커질수록 0에 다가가다가, 그 출력이 1일 때 0이 됨. 반대로 정답일 때의 출력이 작아질수록 오차는 커지게 됨.
- 추가적으로 소프트맥스 함수에 정수(Bias)를 넣어도 그 출력층이 변하지 않는다. 코드 구현시 Max 출력 값으로 개개의 값을 빼도 확률은 변하지 않는다.(오버플로 방지)

경사하강법

- 학습의 목표는 선택한 $E(W)$ 에 대해 최솟값을 주는 w 를 찾는 것이지만, 대부분 $E(W)$ 는 볼록함수가 아니므로 전역 극소점을 찾기 어렵다. 따라서 국소 극소점을 구하고 그때의 $E(W)$ 가 충분히 작다면 목적하는 클래스 분류 or 회귀문제를 나름대로 잘 풀수 있다.
- 이때 사용하는 것이 경사 하강법

$$W = \begin{pmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{pmatrix}$$

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

$$\frac{\partial L}{\partial W} = \begin{pmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{31}} \\ \frac{\partial L}{\partial w_{12}} & \frac{\partial L}{\partial w_{22}} & \frac{\partial L}{\partial w_{32}} \end{pmatrix}$$



경사 : 가중치 W 를 변경했을 때 손실함수 L 이 얼마나 변화했는지 나타낸다.

확률적 경사 강하법

- 배치 학습 : 모든 훈련 샘플 사용.
- 확률적 경사 강하법: '확률적으로 무작위로 골라낸 데이터'에 대해 수행하는 경사 하강법' 극단적으로는 샘플 하나만을 사용하여 파라미터를 업데이트 함.

미니배치의 이용

- 일반적으로 몇 개의 샘플을 하나의 작은 집합으로 묶은 집합 단위로 가중치를 업데이트한다.
이때 작은 집합을 미니 배치라 한다.
- 대개 10-100개 샘플 전후로 결정하고, 각 클래스별 샘플이 하나 이상 들어가게 구성한다. 만약 클래스 출현 빈도가 서로 같을 경우에는 클래스 수와 같은 크기의 미니배치를 구성하고, 그 이상일 경우 랜덤으로 섞은 후에 미니배치를 구성한다.

일반화 성능과 과적합

- 훈련 데이터에는 포함되지 않는, 아직 보지 못한 데이터가 주어져도 바르게 식별해내는 모델이 바람직. 결국 오버피팅을 억제하는 기술 필요.
- 주로 매개변수가 많고 표현력이 높은 모델, 훈련데이터가 적을 때 발생.

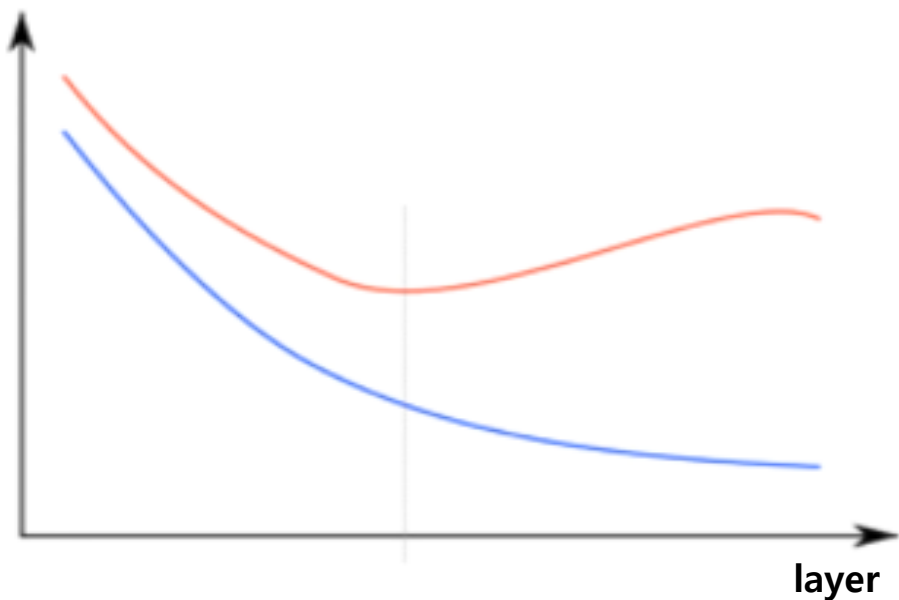
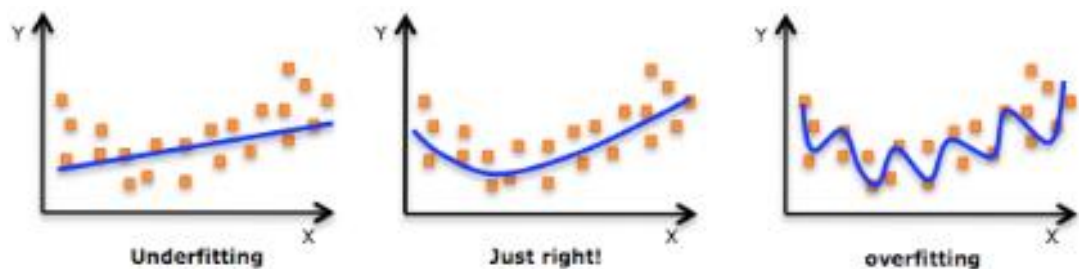
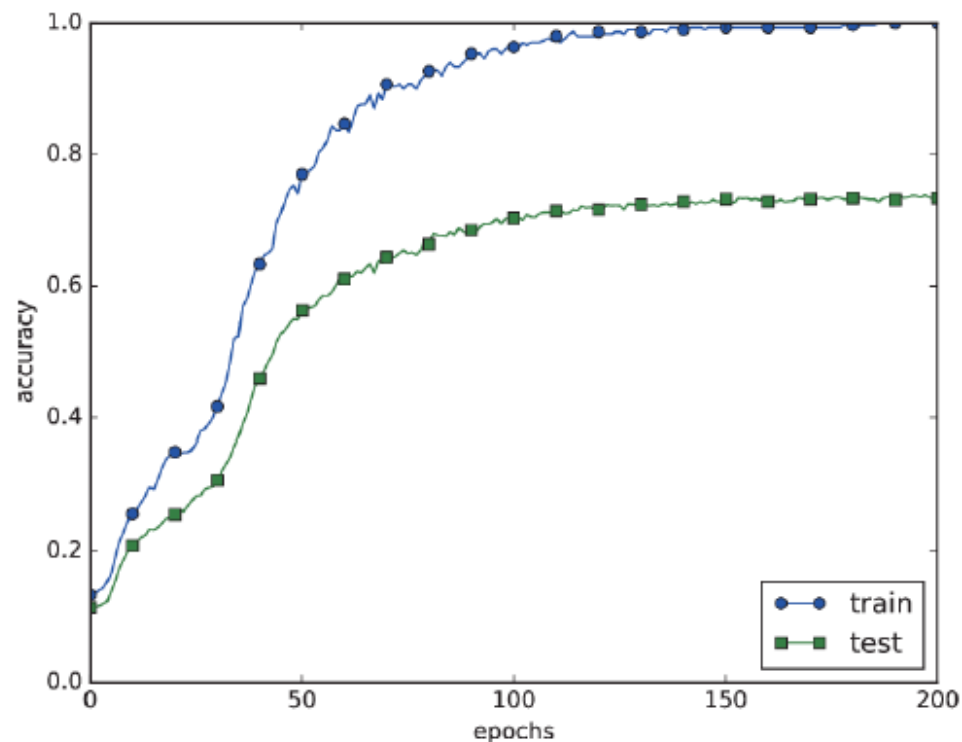


그림 6-20 훈련 데이터(train)와 시험 데이터(test)의 에폭별 정확도 추이



과적합을 완화시키는 법 - 규제화 (Regularization)

- 과적합이란, 결국 작은 국소 극소점에 갇힌 상황이라고 볼수 있다. 신경망의 자유도(가중치 수)가 높을수록 그럴 가능성이 높다고 할 수 있다.
- 단순히 가중치 수를 줄이는 것은 바람직하지 않고, 가중치의 자유도를 제약하는 규제화에 의해 과적합 문제를 완화시키는 몇 가지 방법이 제시되었다.

과적합을 완화시키는 법 - 가중치 감쇠 weight decay

- 학습 과정에서 큰 가중치에 대해서는 그에 상응하는 큰 페널티를 부과하여 오버피팅을 억제하는 방법
- 원래 오버피팅은 가중치 매개변수의 값이 커서 발생하는 경우가 많기 때문.
- 가중치 상한은 특정 크기 이상의 가중치일 경우 1보다 작은 상수를 곱하여 크기를 제한한다. 드롭아웃과 같이 쓸 경우 효과가 좋다.

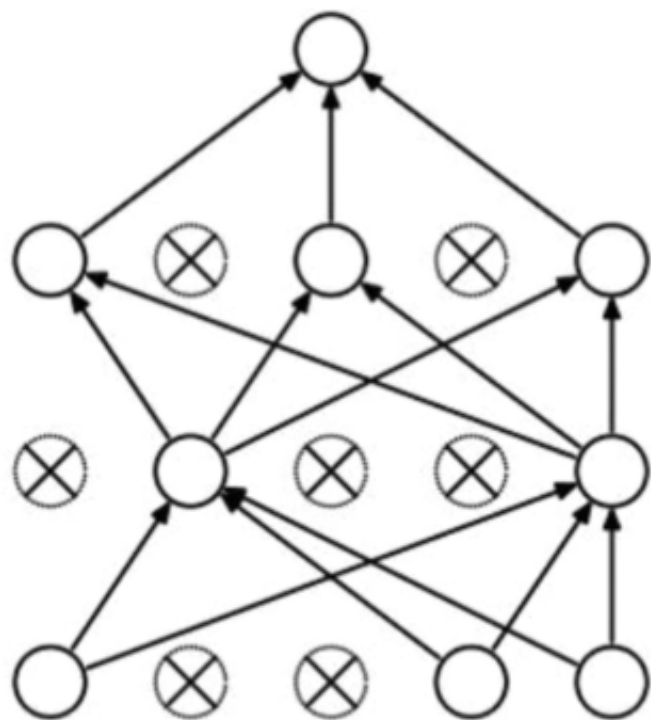
- Let's not have too big numbers in the weight

The diagram shows the loss function $\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(w x_i + b), L_i) + \lambda \sum W^2$ with handwritten annotations. A blue arrow labeled "LOSS" points to the entire equation. A blue arrow labeled "TRAINING SET" points to the index i in the summation. A blue arrow labeled "regularization strength" points to the coefficient λ . The terms w , x_i , b , L_i , and W^2 are highlighted in different colors (red, orange, green, blue, and green respectively).

$$\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(w x_i + b), L_i) + \lambda \sum W^2$$

과적합을 완화시키는 법 - 드롭아웃

- "Randomly set some neurons to zero in the forward pass"



Forces the network to have a redundant representation.



학습을 위한 트릭 - 데이터 정규화

- 훈련데이터에 어떠한 경향이 포함될때 그 경향이 학습에 방해가 될수 있다. 이때 전처리를 거치게 된다.
- 훈련데이터, 테스트 데이터, 향후 추론에 쓰이는 데이터에도 모두 동일한 전처리를 수행한다.
- 데이터 정규화(normalization of data or standardization of data)
각 샘플에 샘플 전체 평균을 빼고 그 표준편차로 나누는 것을 말한다.
결국 표준정규분포 (평균 =0, 분산 =1)을 따르게 하는 것.

학습을 위한 트릭 - 데이터 확장

- 확보된 샘플 데이터를 일정하게 가공하여, 양적으로 늘리기 하는 방법
- 예) 이미지의 경우 평행이동, 거울상 반전, 회전등의 변환
명암, 색등을 변화 시켜 샘플로 사용하는 것.
수치의 경우 가우스 분포를 따르는 랜덤 노이즈를
일괄적으로 적용하는 것도 유효하다.

학습을 위한 트릭 - 여러 신경망의 평균

- 랜덤 포레스트 등과 같이 여러 개의 신경망에 입력하여 얻어진 출력의 평균을 응답으로 삼는 것을 말한다.
- 신경망의 경우 같은 데이터이고 구조가 다른 것, 구조가 같아도 초기값이 다른것등의 형태로 구성한다.
- 복수의 신경망을 학습할 수록 정확도가 높게 나오는 경우가 드물지 않으나, 비시간 및 비용이 많이 든다.

학습을 위한 트릭 - 학습률의 결정 방법

- 학습률은 사람이 직접 시행착오를 통해 설정하는 것이 일반적
- 보통 학습 초기에 값을 크게 설정하다가 줄이거나, 신경망의 모든 층에 서로 다른 값을 사용하는 것.
- 보통 AdaGrad or RMSProp가 가장 많이 사용된다.

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

\mathbf{h} 는 기존 기울기 값을 제공하여 계속 더해준다. (\odot 기호는 행렬의 원소별 곱셈을 의미).

매개변수를 갱신할 때마다 학습률을 조정.

매개변수의 원소 중에서 많이 움직인(크게 갱신된) 원소는 학습률이 낮아진다는 뜻인데, 다시 말해 학습률 감소가 매개변수의 원소마다 다르게 적용됨을 뜻함.

학습을 위한 트릭 - 모멘텀

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

[SGD]

$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

[Momentum]

- α 는 0.5 ~ 0.9 사이의 값을 가진다.
- <http://aikorea.org/cs231n/neural-networks-3/>

그림 6-4 모멘텀의 이미지: 공이 그릇의 곡면(기울기)을 따라 구르듯 움직인다.



학습을 위한 트릭 - 가중치의 초기화

- 가중치 초기값을 정하는 방법은 올바른 학습을 하는 데 매우 중요하다.
- 신경망에서는 가중치가 클수록 오버피팅이 일어날 가능성이 높으므로, 가중치 값을 작게 만들어 사용한다.
- 단 가중치의 초기값을 0으로 만들거나 같을 경우, 신경망의 각 층을 진행해도 가중치 값이 여전히 같은 값을 가지게 된다. 따라서 가중치가 고르게 되어버리는 상황을 막으려면 초기값을 작고, 무작위하게 설정해야 된다.
- 일반적으로 정규분포 기반 랜덤값을 생성하여 가중치 초기값을 설정, 바이어스는 0으로 설정.
- 표준편차의 크기는 sigmoid 유형:Xavier의 경우 $1/\sqrt{n}$, Relu 유형 He의 경우 $2/\sqrt{n}$ 사용.

학습을 위한 트릭 - 샘플의 순서

- 일반적으로 신경망이 익숙하지 않은 샘플을 먼저 보이는 전략이 학습에 유리하나. 잘못된 목표 출력이 있을 경우 역효과가 날 수도 있다.
- 요즘의 경우 랜덤 샘플링한 샘플을 기계적으로 조합하여 사용한다.