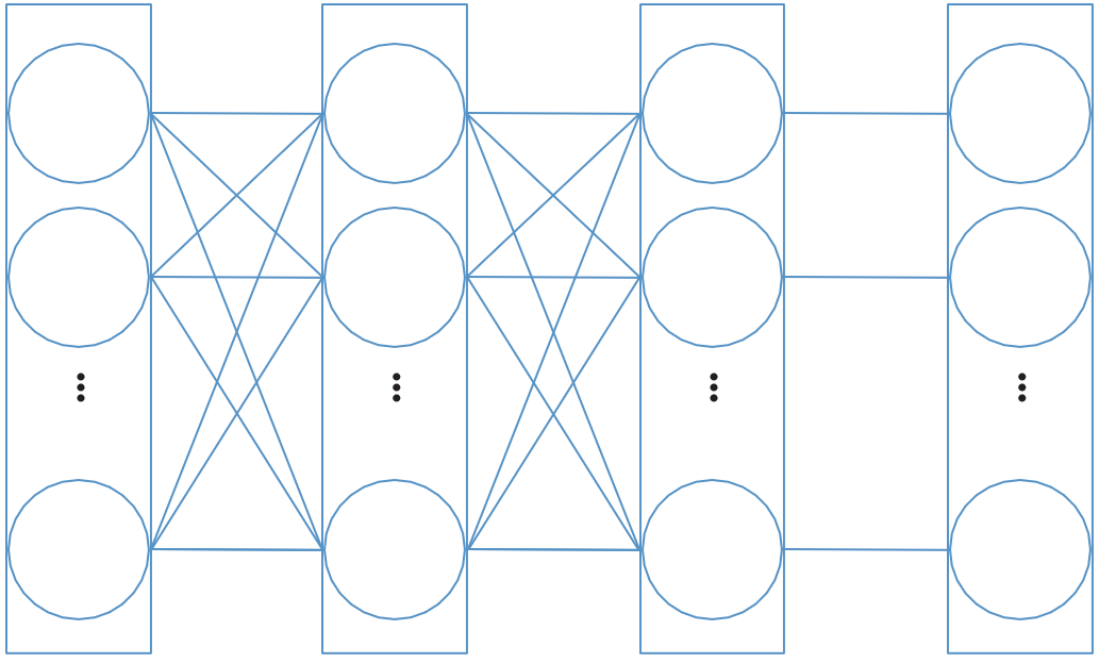
딥러닝 개요

1 딥러닝의 탄생

- 딥러닝은 인간의 뇌가 생각하는 방식을 머신러닝 알고리즘으로 설계한 것.
- 과거부터 이 알고리즘은 신경망이라고 알려짐.
- 딥러닝은 오히려 정식 이름이라기보다는 심층신경망의 별칭에 가까움.
- 심층신경망은 마치 뇌 속의 여러 개의 뉴런이 서로 엉켜져서 추론을 하는 모양 같음.

심층신경망의 기본 구조



1 딥러닝의 탄생

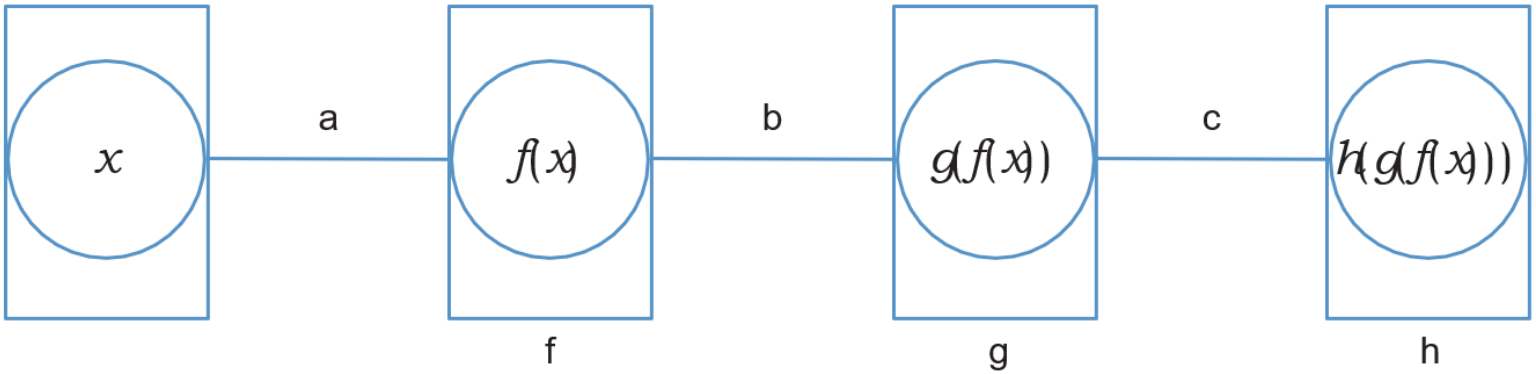
- 위 그림에서 동그라미는 뉴런이고, 여러 뉴런이 일렬로 존재하는 네모박스를 레이어(layer)라고 부름.
- 뉴런은 때로는 노드라고 불리기도 함. 뉴런은 다른 레이어의 뉴런들과 선으로 연결되어 영향을 주는데, 각각의 뉴런에 대한 영향 범위를 다르게 하기 위해서 선 위에는 가중치라는 변수가 존재.
- 한 개의 레이어에는 여러 개의 뉴런이 존재할 수 있음. 기본적으로 뉴런이 가지고 있는 정보는 가중치와 곱해져서 다음 레이어의 뉴런으로 전파.

2 딥러닝과 머신러닝의 관계

- 다른 머신러닝 알고리즘들처럼 딥러닝 역시 데이터를 입력받아 예측을 출력.
- 딥러닝을 다른 머신러닝과 다르게 분류하는 이유는 딥러닝은 사람의 뇌를 형상화한 머신러닝 알고리즘이라는 기본 개념 안에 뇌가 사물을 이해하는 과정(CNN), 뇌가 문맥을 이해하는 과정(RNN) 등, 각 상황에 따라 뇌가 이해하는 과정을 형상화한 세부적인 딥러닝 알고리즘 역시 깊은 이해를 필요하기 때문

3 딥러닝 이름의 유례

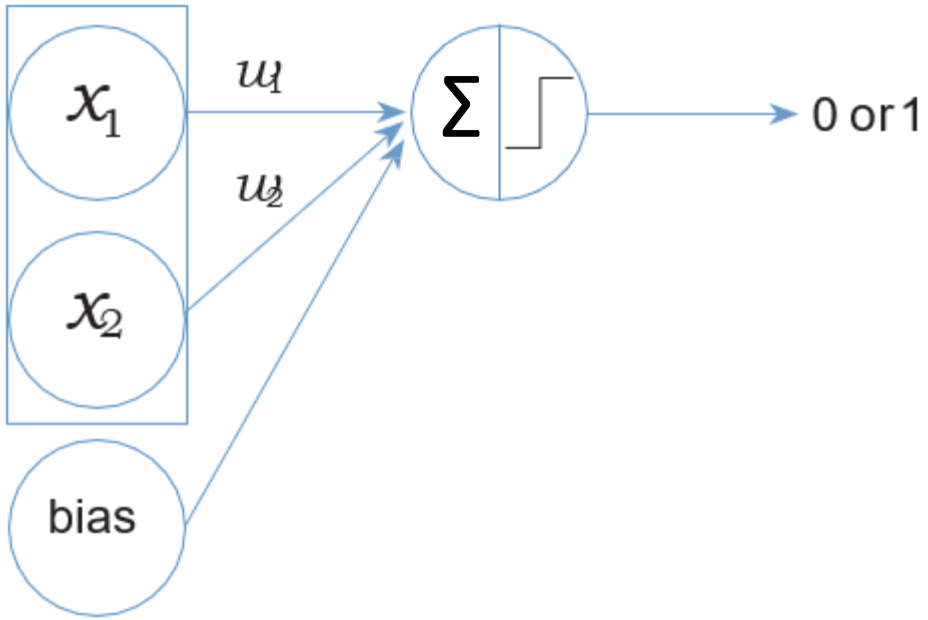
딥러닝의 계산 원리



- 노드와 가중치의 곱이 다음 레이어로 전파.
- 가중치와 노드의 곱을 하나의 함수라고 가정할 경우 두 번째 레이어를 f(x), 세 번째 레이어를 g(f(x)), 마지막 레이어를 h(g(f(x)))라고 간추려 말할 수 있음.
- 이처럼 x라는 입력이 들어왔을 때 결괏값을 구하기 위해서는 h(g(f(x)))를 계산해야 하고, 그 계산 과정이 함수 안의 함수로 깊게 이어져 있기 때문에 딥러닝이라는 별칭이 생긴 것.

4 딥러닝 탄생 배경

- 딥러닝이라는 브랜드가 생기기 전에 신경망이라는 기술이 있었고, 신경망이 있기 전에 뉴런연구가 이미 이뤄지고 있었음.
- 오래전 뉴런 하나로 AND 연산이나 OR 연산 등을 해결하는 기술이 탄생했는데, 그 기술의 이름이 바로 퍼셉트론(perceptron). 퍼셉트론의 구조는 아래 그림과 같음.



퍼셉트론의 구조

5 퍼셉트론

- 퍼셉트론은 두 개의 입력이 있을 때 하나의 뉴런으로 두 개의 입력을 계산한뒤, 최종 결괏값으로 0 또는 1을 출력하는 모델.

- 두 개의 입력은 가중치와 곱해져서 뉴런의 첫 번째 단계인 시그마로 들어갑니다. 시그마(z)단계에서는 모든 가중치*입력값과 편향값(bias)을 더해지는 과정이 이뤄짐.

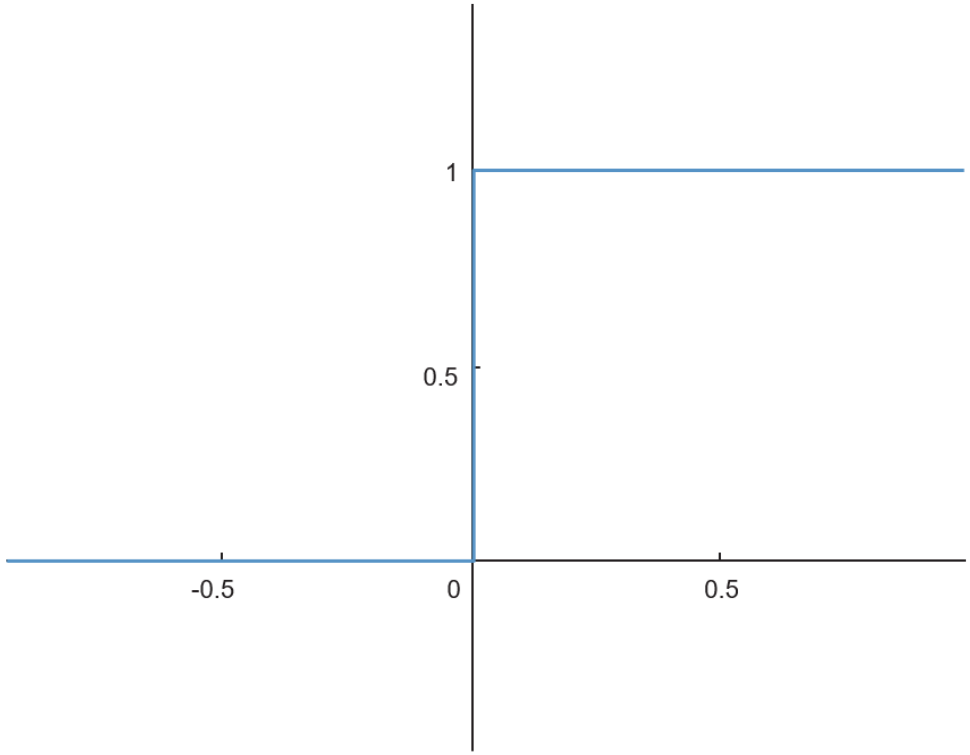
$$z = w_1 x_1 + w_2 x_2 + \text{bias}$$

5 퍼셉트론

- 편향값은 모델이 좀 더 쉽고 빠르게 목적을 달성하는 데 도움을 줌.
- 시그마(z)의 값은 뉴런의 두 번째 단계인 활성화 함수(a, activation function)의 입력값으로 들어갑니다. 퍼셉트론은 스텝 함수라는 다음과 같은 활성화 함수를 사용함.

$z < 0$일 경우, $a(z) = 0$

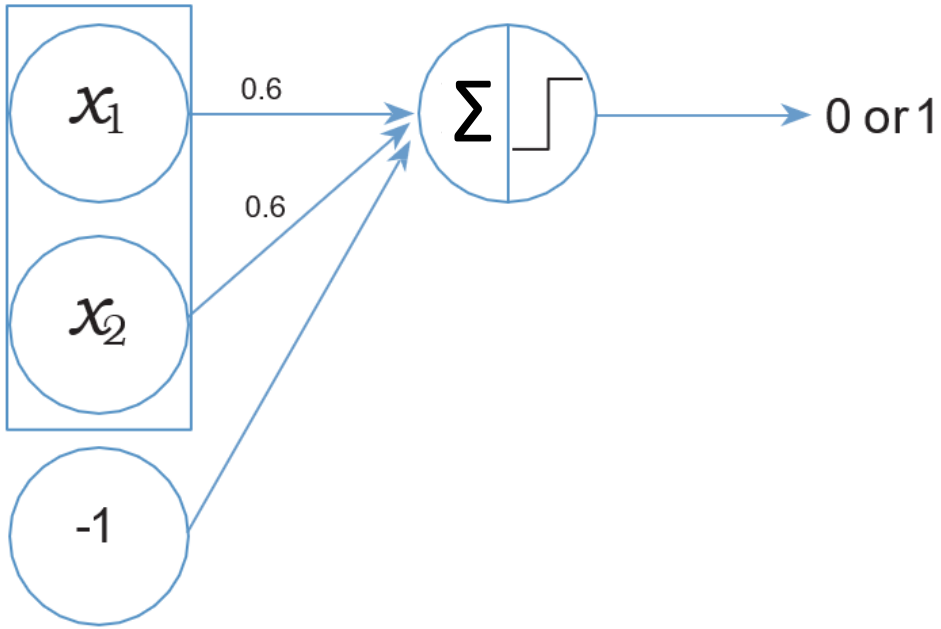
$z >= 0$일 경우, $a(z) = 1$



스텝함수

5 퍼셉트론

- AND 연산을 하는 퍼셉트론의 예



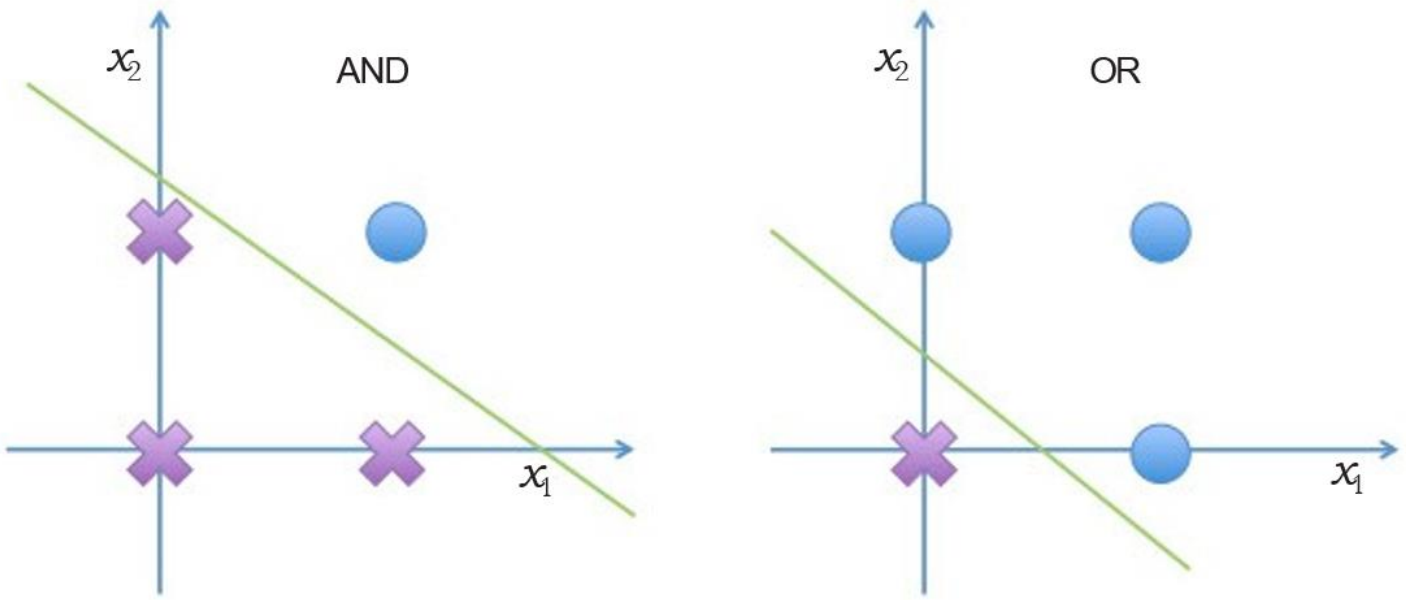
퍼셉트론 연산 예제

- 계산 결과, 아래 표에서 AND 연산과 동일한 것을 확인할 수 있음.

x1	x2	a(z)	x1 AND x2
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

5 퍼셉트론

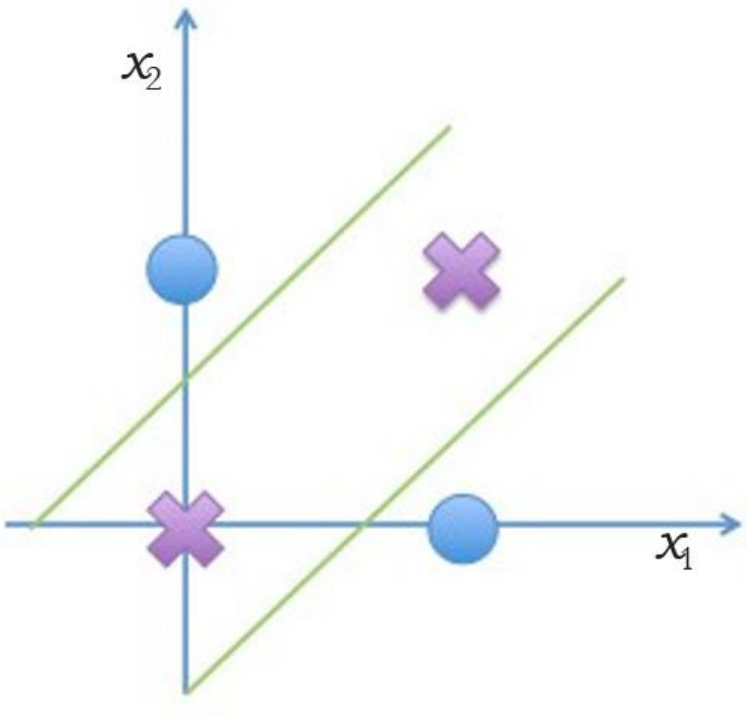
- 하나의 퍼셉트론으로 AND 연산을 할수 있었던 이유



퍼셉트론의 의사결정선

5 퍼셉트론

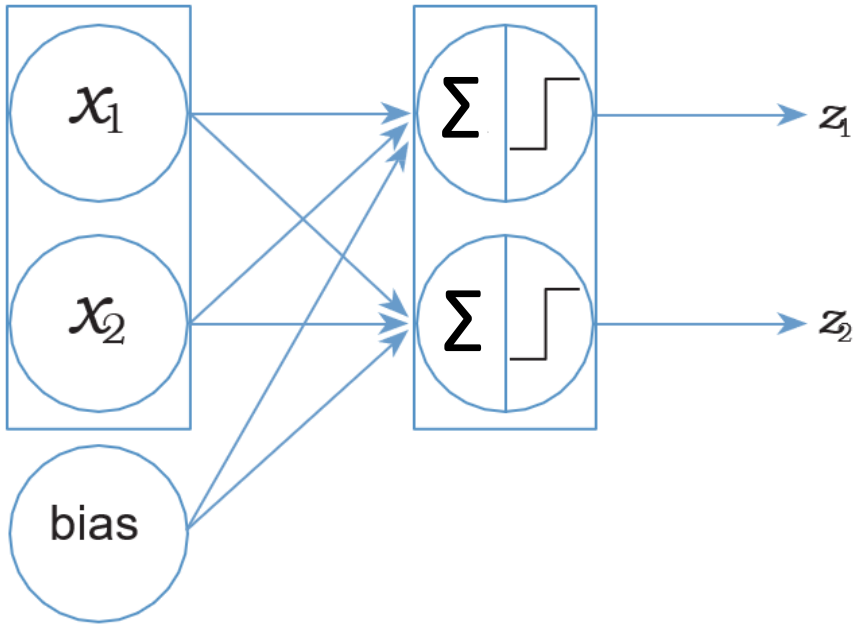
- 과거 퍼셉트론 연구자들은 퍼셉트론으로 XOR 연산이 불가능하다고 판단했음.



의사결정선이 2개 필요한 XOR 연산

6 다층 퍼셉트론

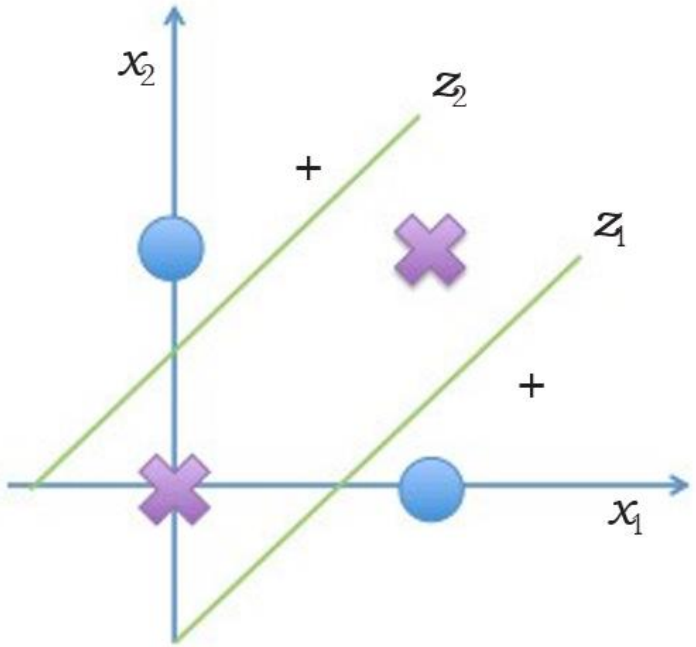
- 두 개의 퍼셉트론이면 두 개의 의사결정선을 그릴 수 있지 않을까 생각해 볼 수 있음.



두 개의 퍼셉트론

6 다층 퍼셉트론

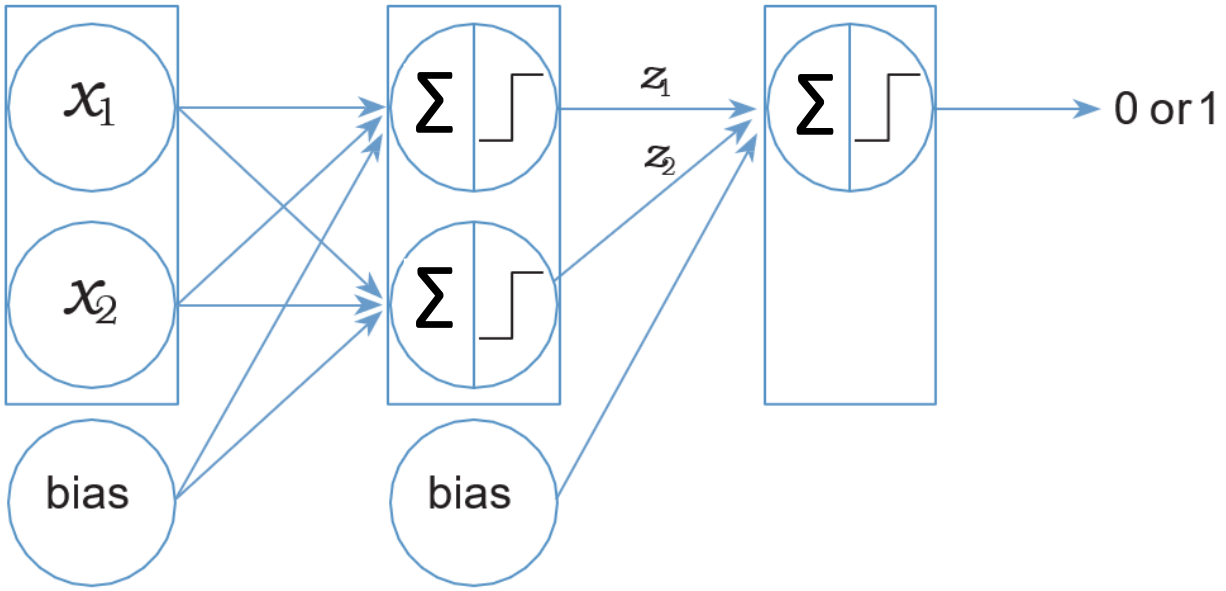
- 두 개의 퍼셉트론이 있으면 z1과 z2라는 의사결정선을 다음과 같이 그릴 수 있음.



두 개의 퍼셉트론에 의한 두 개의 의사결정선

6 다층 퍼셉트론

- 앞의 그림과 같이 하나의 의사결정선을 그리기 위해 다음 레이어에 퍼셉트론 하나를 더 추가

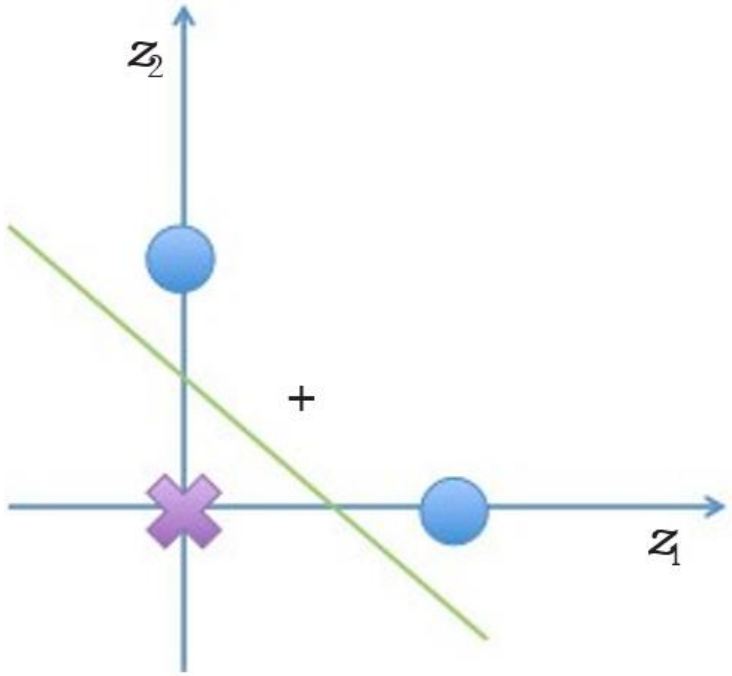


XOR 연산을 위한 다층 퍼셉트론 구조

6 다층 퍼셉트론

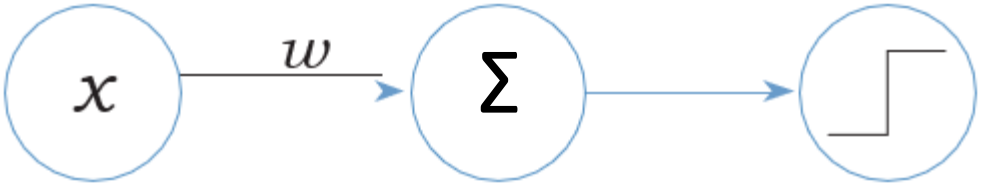
- 첫 번째 x1, x2의 입력을 받는 층을 입력 레이어(input layer). 그리고 퍼셉트론들이 존재하는 두 번째, 세 번째 층을 히든 레이어(hidden layer). 그리고 마지막으로 0 또는 1이 출력되는 마지막 층을 출력 레이어(output layer).
- 이처럼 다수의 뉴런으로 구성된 여러 층 구조를 갖춘 딥러닝을 다층 퍼셉트론(Multi Layer Perceptron, MLP).
- z1과 z2를 입력으로 받는 퍼셉트론은 z1과 z2를 축으로 하는 2차원 공간에 새로운 의사결정선을 아래와 같이 그림.

다층 퍼셉트론의 결과, 1개의 의사결정선으로 구분된 XOR 데이터포인트



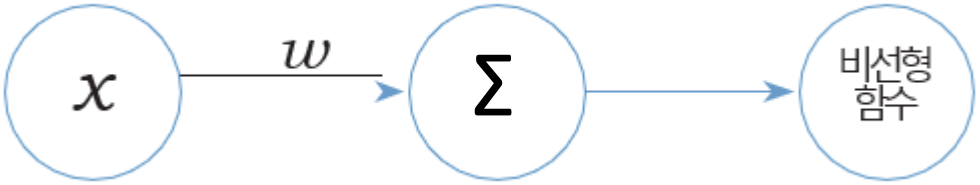
7 뉴런(노드)

- 퍼셉트론의 구조



퍼셉트론의 구조

- 퍼셉트론은 모든 입력을 가중치와 곱해준 후 합산한 뒤, 스텝 함수에 그 결괏값을 넘겨주어 0또는 1 값을 출력. 퍼셉트론에서 스텝 함수를 활성화 함수(activation function)라고 부름.
- 오늘날의 뉴런은 활성화 함수로 다양한 비선형 함수를 사용.

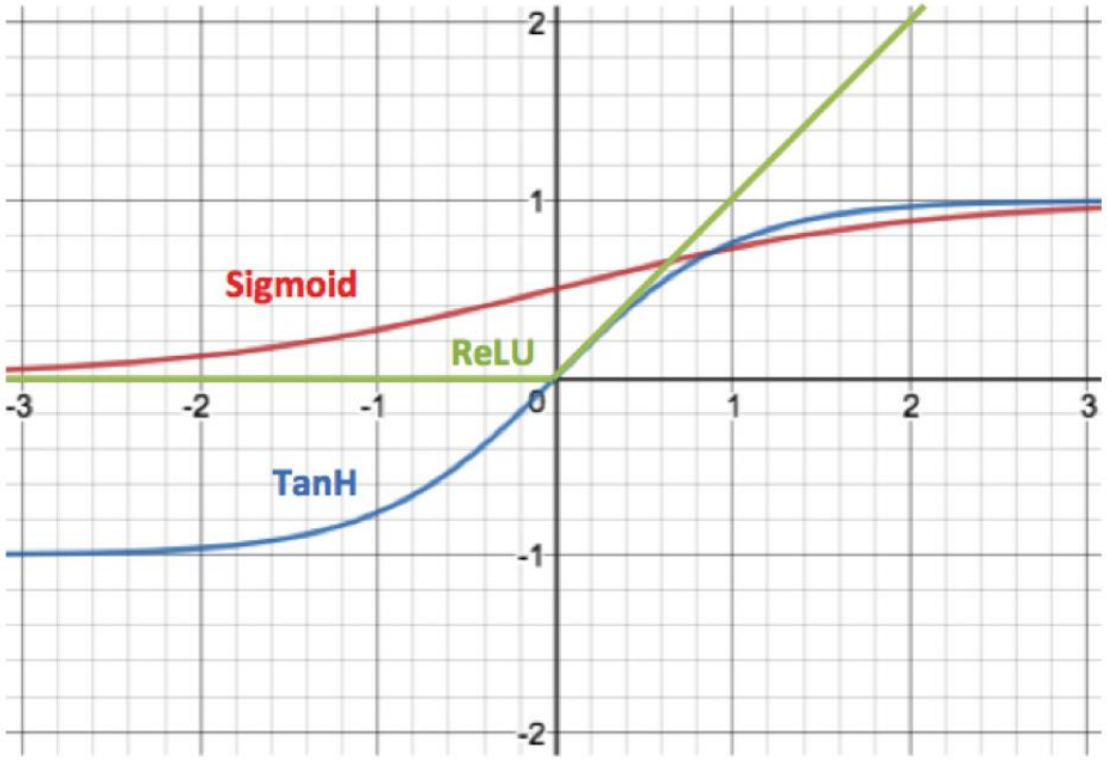


뉴런의 구조

- 스텝 함수 대신 비선형 함수를 활성화 함수로 사용하는 대표적인 이유로는 역전파(backpropagation)를 사용한 모델 학습 시 활성화 함수가 미분 가능해야 하는데, 스텝 함수는 미분이 불가능하기 때문.

7 뉴런(노드)

- 최근 딥러닝 모델에서 많이 사용되는 대표적인 비선형 활성화 함수로는 Sigmoid, TanH, ReLU 등이 있음.



활성화 함수의 예제

$$\text{Sigmoid} = \frac{1}{1+e^{-x}}$$

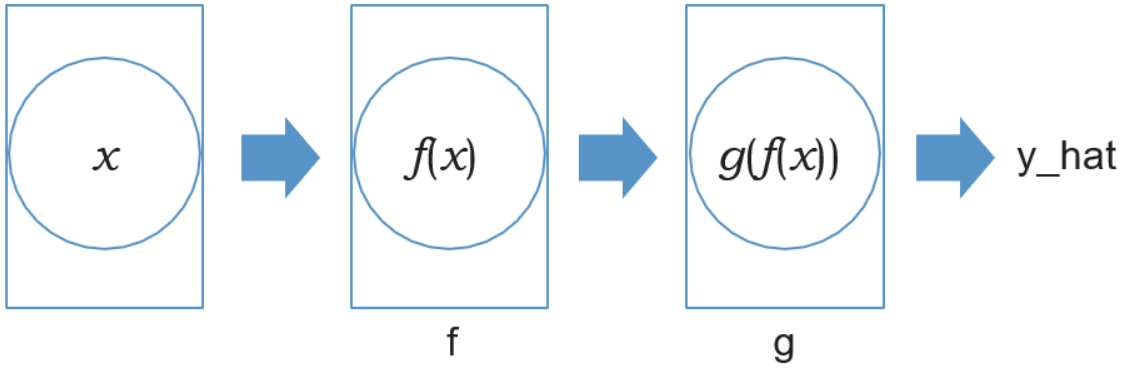
$$\text{TanH} = \frac{2}{2+e^{-2x}} - 1$$

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

8 딥러닝의 학습

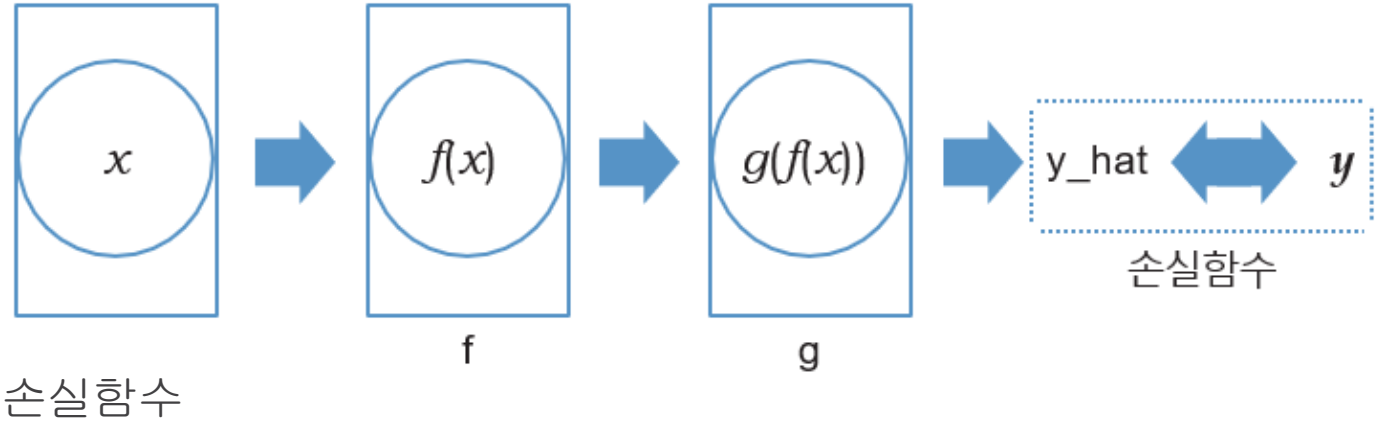
8.1 순전파(forward propagation)

- 순전파란 딥러닝에 값을 입력해서 출력을 얻는 과정.



x라는 입력이 순전파를 통해 y_hat이라는 출력이 되는 과정

- 출력값과 정답의 차이를 구하기 위한 함수를 손실함수라고 함.

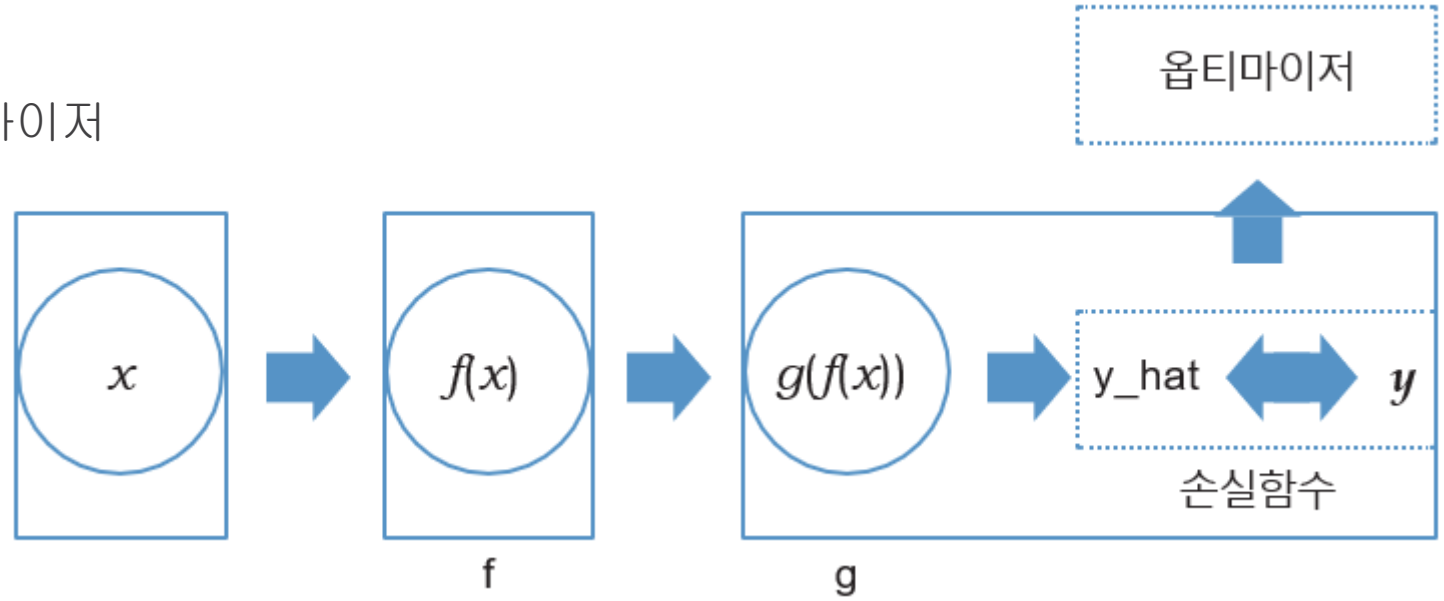


8 딥러닝의 학습

8.2 손실함수

- 손실함수는 출력값과 정답의 차이를 계산.
- 보통 회귀에는 평균제곱오차를, 분류 문제에는 크로스 엔트로피를 손실함수로 사용.
- 매개변수(가중치, 편향값)를 조절해서 손실함수의 값을 최저로 만드는 과정을 최적화(optimization) 과정이라고 부르고,
- 최적화 과정은옵티마이저(optimizer)를 통해 이뤄짐.
- 옵티마이저는 역전파(back propagation) 과정을 수행해서 딥러닝 모델의 매개변수를 최적화.

옵티마이저



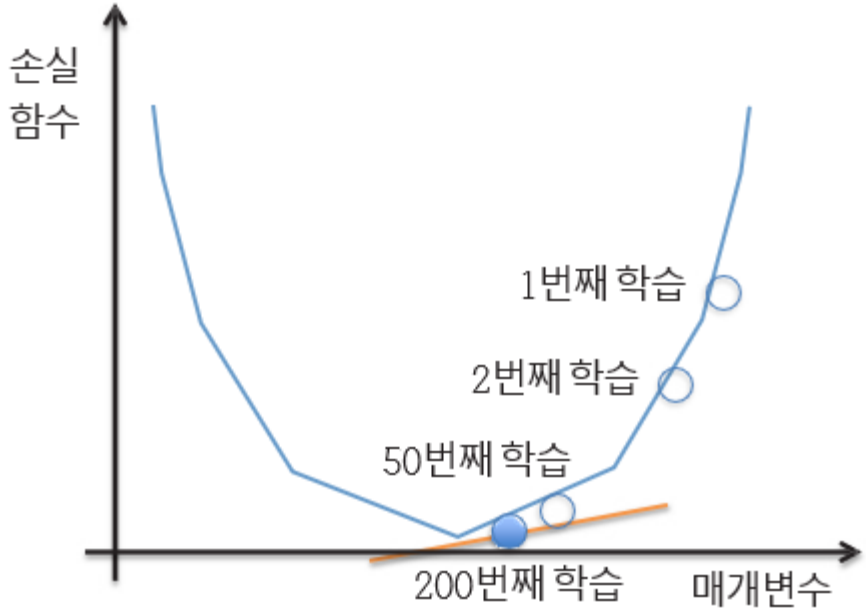
8 딥러닝의 학습

8.3 최적화

- 대표적인 최적화 방법은 경사하강법.
- 반복적으로 손실함수에 대한 모델 매개변수의 미분값을 구한 후, 그 미분값의 반대 방향으로 매개변수를 조절해나가면 결국에는 최저 손실함수 값에 도달한다는 이론.

매개변수 new = 매개변수 old – 학습률 * dl/dw

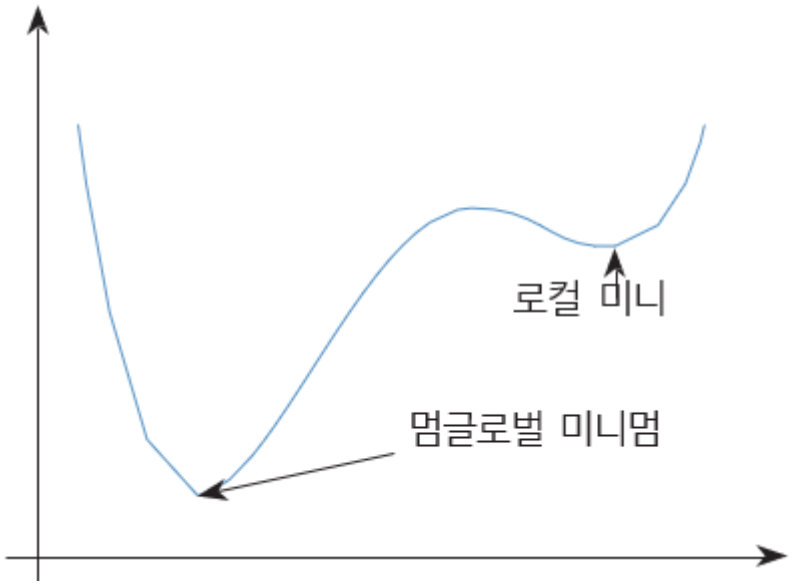
최적화 과정



8 딥러닝의 학습

- 이 변곡점 지점을 로컬 미니멈이라고 부르고, 여러 개의 로컬 미니멈이 있을 때 가장 낮은 로컬 미니멈을 글로벌 미니멈.

- 전통적으로 모든 학습 데이터의 손실함수를 계산한 후 경사하강법으로 로컬 미니멈을 찾아나가는 방법으로 모델 매개변수를 최적화했는데, 이 방법을 배치 경사하강법(batch gradient descent).

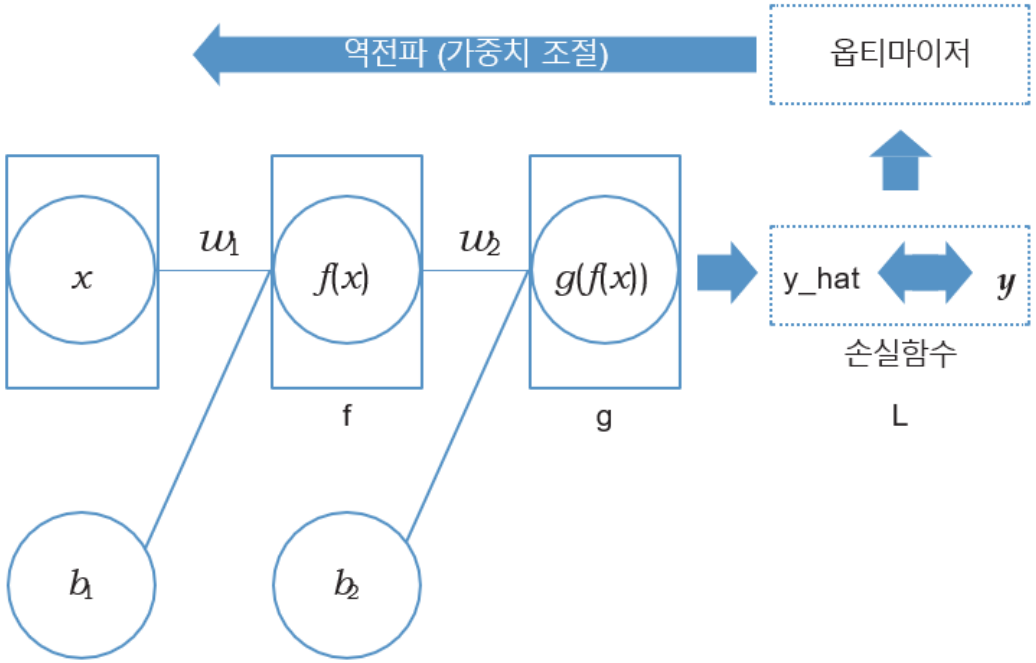


로컬 미니멈과 글로벌 미니멈

8 딥러닝의 학습

8.4 역전파

- 옵티마이저는 손실함수의 값을 최저로 하기 위해 역전파(back propagation)를 사용해 딥러닝 모델의 모든 매개변수를 변경.



역전파

8 딥러닝의 학습

- 손실함수(L)에 대한 각 매개변수의 미분값은 다음과 같이 연쇄법칙(Chain Rule)을 적용해서 구할 수 있음.

```
dL/dw2 = dL/dg * dg/dw2
dL/db2 = dL/dg * dg/db2
dL/dw1 = dL/dg * dg/df * df/dw1
dL/db1 = dL/dg * dg/df * df/db1
```

- 매개변수별 미분값을 다음과 같이 학습률을 적용해 변경.

```
w2 new = w2 old – 학습률 * dL/dw2
b2 new = b2 old – 학습률 * dL/db2
w1 new = w1 old – 학습률 * dL/dw1
b1 new = b1 old – 학습률 * dL/db1
```

- 이 같은 방법으로 충분한 반복 학습 과정을 거치고 나면 모든 매개변수가 손실함수를 최저로 하는 값으로 수렴하게 됨.

8 딥러닝의 학습

8.5 옵티마이저

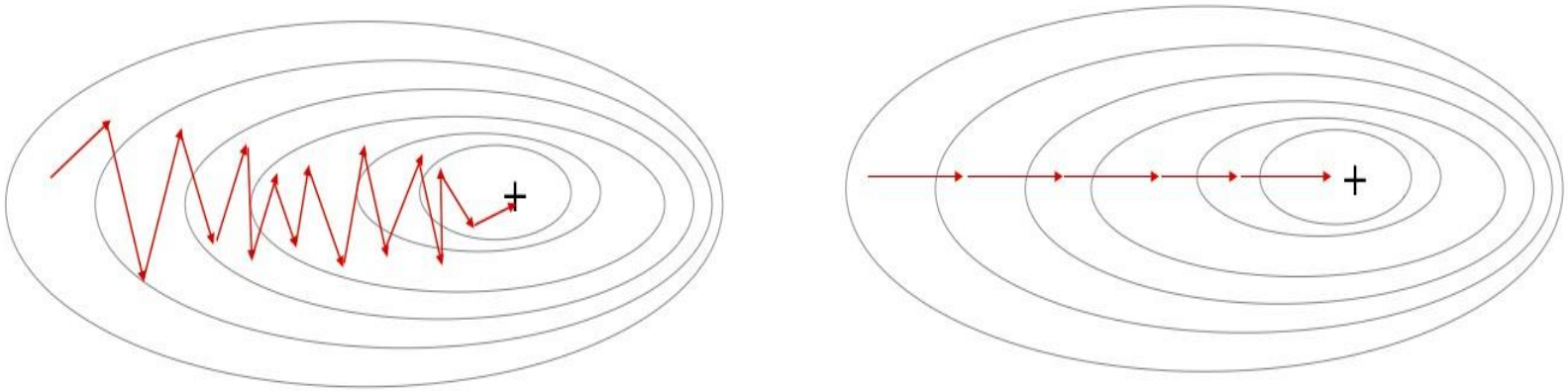
배치 경사하강법

- 배치 경사하강법은 딥러닝 모델을 최적화하는 가장 기본적인 방법.
- 가중치에 대한 손실함수의 1차 미분을 구하는 것만으로도 손실함수의 로컬 미니멈, 즉 출력값과 정답의 차이가 적은 딥러닝 모델을 찾을 수 있다는 것부터 획기적.
- 하지만 모든 학습 데이터의 손실함수를 계산한 후에만 딥러닝 모델의 매개변수가 조금씩 변경되기 때문에 로컬 미니멈까지 매개변수를 변경하는 데 걸리는 시간이 오래 걸리고, 매개변수 조절에 필요한 계산량도 많다는 단점이 있음.
- 또한 보통 딥러닝 모델을 학습시킬 때 로컬 미니멈은 여러 개 존재할 수 있음. 로컬 미니멈이 여러 개 존재할 때 최적의 모델은 글로벌 미니멈의 매개변수를 가진모델이어야 하는데, 배치 경사하강법은 무작위로 부여된 매개변수에서부터 가장 가까운 로컬미니멈에 멈추게 될 것.

8 딥러닝의 학습

SGD

- 배치 경사하강법이 한 번 매개변수를 변경하는 데 드는 계산량이 너무 크고 시간이 오래 걸린다는 단점이 있어서 고안된 방법 중 하나가 SGD(stochastic gradient descent).
- 모든 데이터를 계산해서 매개변수를 변경하는 배치 경사하강법과 달리, 하나의 데이터마다 매개변수를 변경하는 방법.



SGD(왼쪽)와 경사하강법(오른쪽) 비교[1]

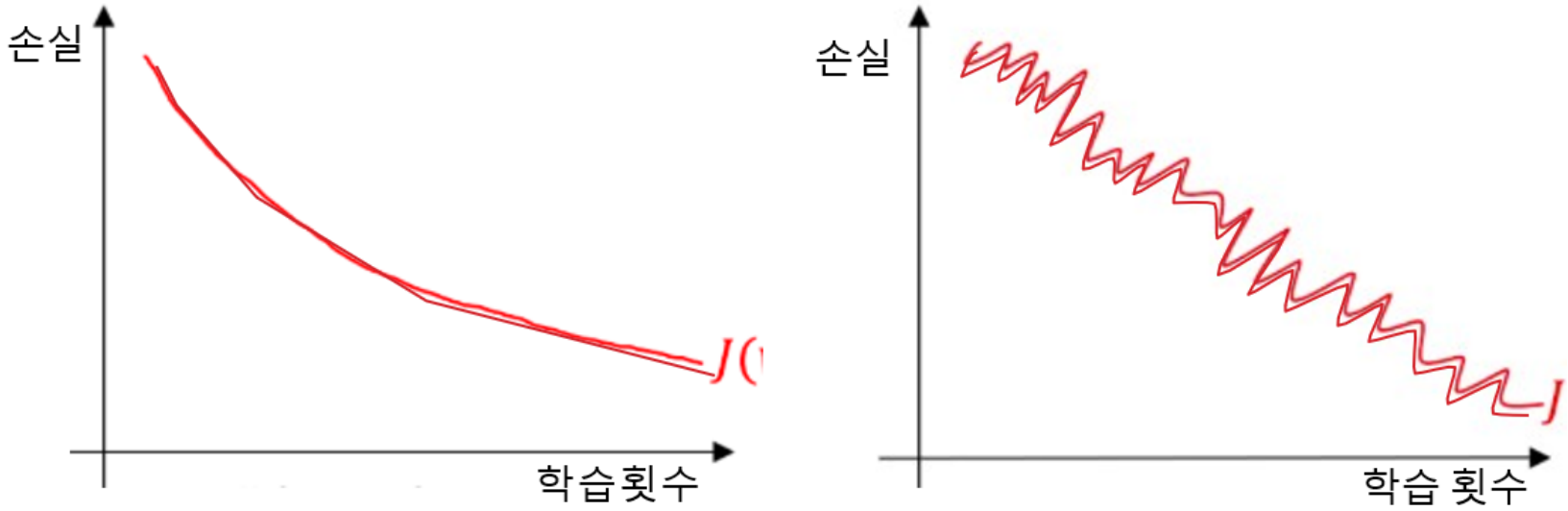
8 딥러닝의 학습

- SGD의 눈에 띄는 장점은 제한된 자원으로도 충분히 딥러닝 모델을 학습시킬 수 있다는 것.
- 모든 데이터에 대한 계산 결과를 저장해야 하는 배치 경사하강법과 달리, 하나의 데이터에 대한 계산만 메모리에 저장하면 되므로 자원이 적은 컴퓨터에서 딥러닝 모델을 학습하기가 용이.
- 또한 조금씩 매개변수가 변하면서 근처의 로컬 미니멈으로 수렴하는 배치경사하강법에 비해 매개변수가 요동을 치며 변하면서 때로는 근처의 로컬 미니멈을 지나쳐글로벌 미니멈에 수렴하는 행운이 있을 수도 있음.
- 반면 배치 경사하강법보다 못한 매개변수로 학습이 완료될 수도 있음. 손실함수가 매우 불규칙하고 로컬 미니멈이 많을 때는 SGD가 배치 경사하강법보다 더 나은 최적화 알고리즘일 수 있음.

8 딥러닝의 학습

미니 배치

- 배치 경사하강법은 너무 느리고, 리소스도 많이 사용하고, SGD는 확실히 빨리 학습되지만 모델이 최적화가 안 돼 있는 경우가 많으므로 이러한 두 방법의 절충안이 바로 미니 배치.
- 전체 데이터를 계산해서 매개변수를 변경하는 대신 정해진 양만큼 계산해서 매개변수를 최적화하는 방법.



배치 경사하강법과 미니 배치의 학습에 따른 손실 비교

8 딥러닝의 학습

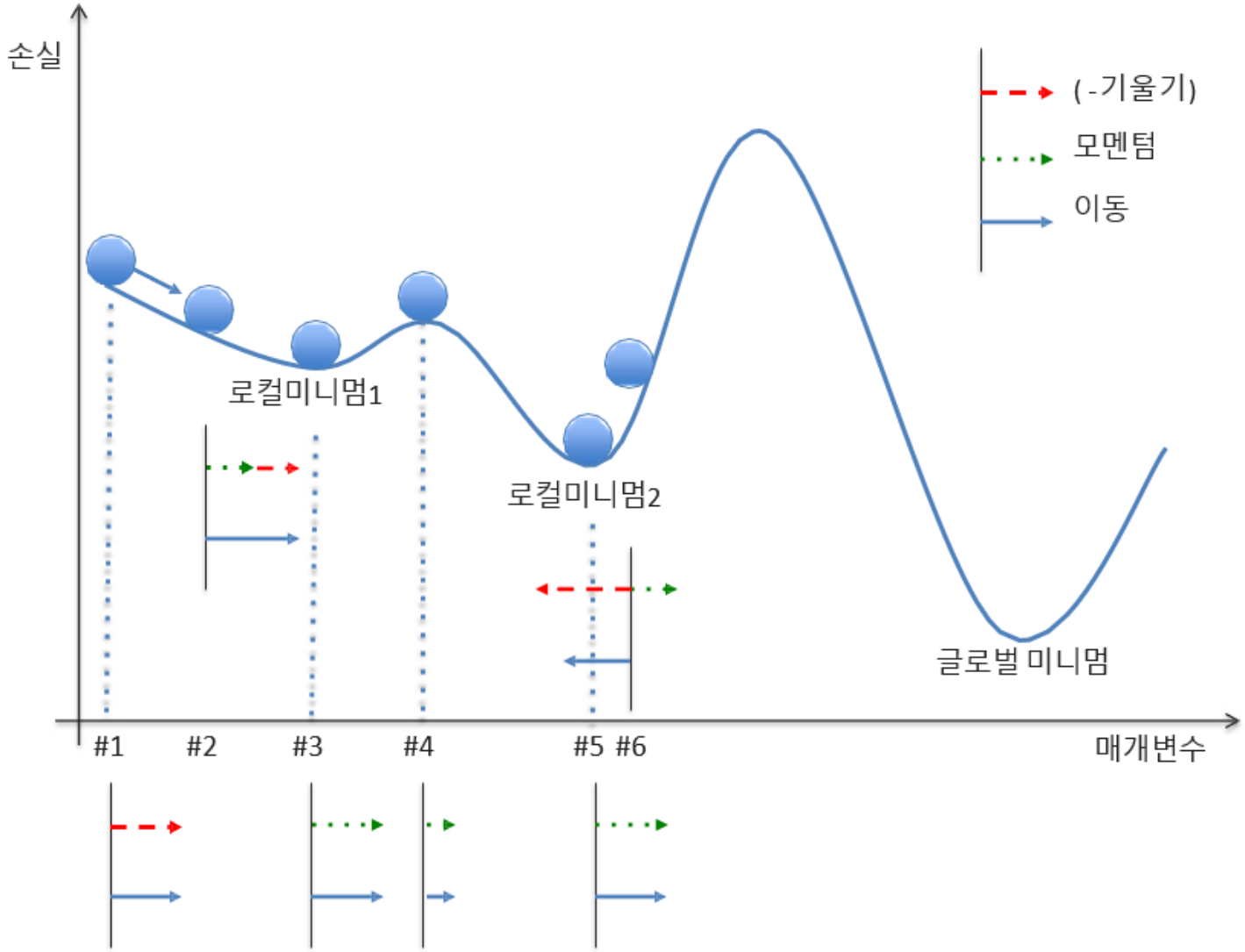
모멘텀

- 단순히 1차 미분만으로 정직하게 최적화를 진행하다 보면 당연히 근처의 로컬 미니멈으로 딥러닝 모델이 학습 됨.
- 이를 개선하기 위한 방안으로 물리학을 응용한 방법이 있는데, 바로 모멘텀.
- 마치 공을 언덕에서 굴렸을 때 위치 에너지와 운동 에너지의 영향으로 공이 가던 방향으로 힘을 받아 최초 로컬 미니멈에 머무르지 않고, 더 낮은 로컬 미니멈까지 공이 굴러갈 수 있다는 이론.

매개변수 new = 매개변수 old + 이동변수
이동변수 = 모멘텀 – 학습률 * dl/dw
모멘텀 = 모멘텀조정률 * 이동변수

8 딥러닝의 학습

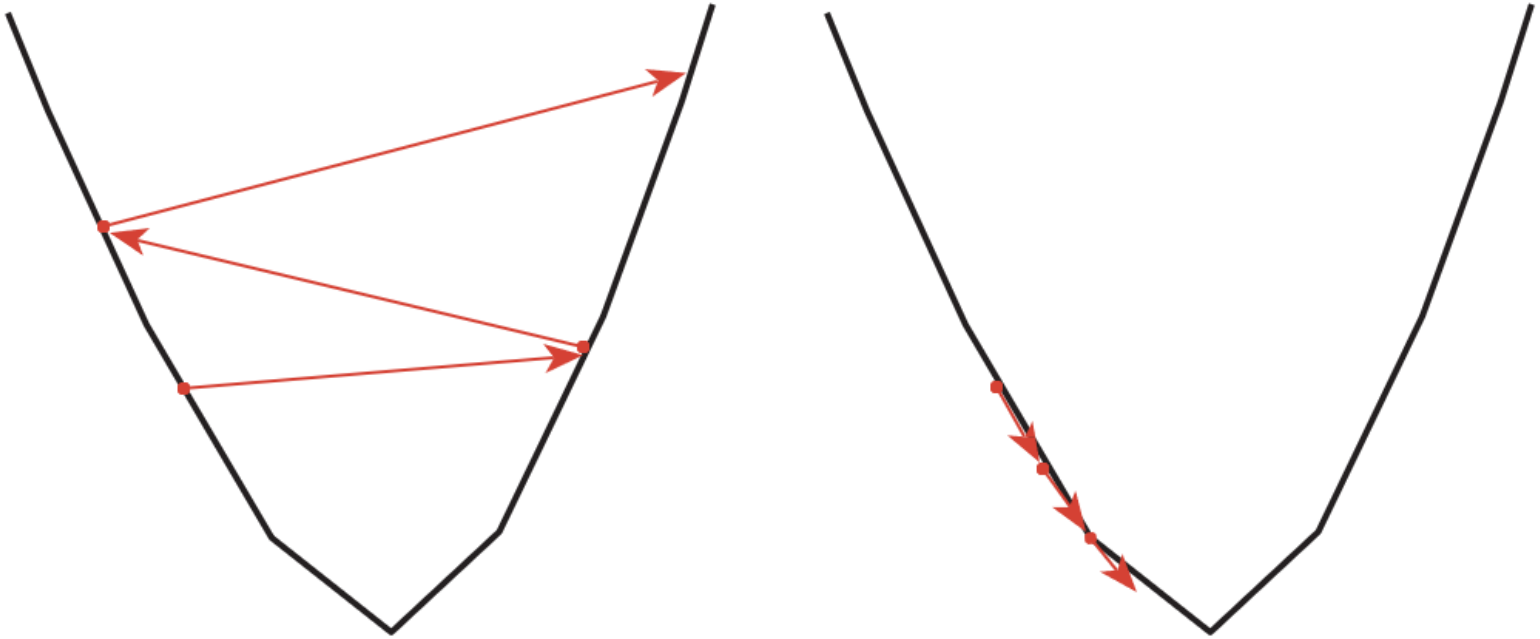
모멘텀에 의해 최초 로컬 미니멈에 머무르지 않는 케이스



8 딥러닝의 학습

학습률

- 경사하강법 공식에는 항상 학습률(learning rate)이 있음.
- 학습률을 크게 설정하면 매개변수 변경치가 커져서 로컬 미니멈으로 수렴하지 않을 수 있고, 또 너무 작게 설정하면 학습시간이 상당히 오래 걸릴 수 있음.



학습률이 클 때 학습률이 작을 때

학습률이 클 때와 작을 때의 손실 변화 비교

8 딥러닝의 학습

- 전통적인 배치 경사하강법의 경우, 최초 고정된 학습률을 학습이 종료될 때까지 유지하지만 최근 들어 많은 연구와 함께 시간 기반 학습률 조정(time based decay), 스텝 기반 학습률 조정(step decay) 등, 학습 중간에 학습률을 조정하는 방법이 많이 생기고 있고, 딥러닝 프레임워크에서도 이를 지원.
- 이처럼 중간에 학습률을 조정하는 방식을 decay라고 함.

8 딥러닝의 학습

Adagrad

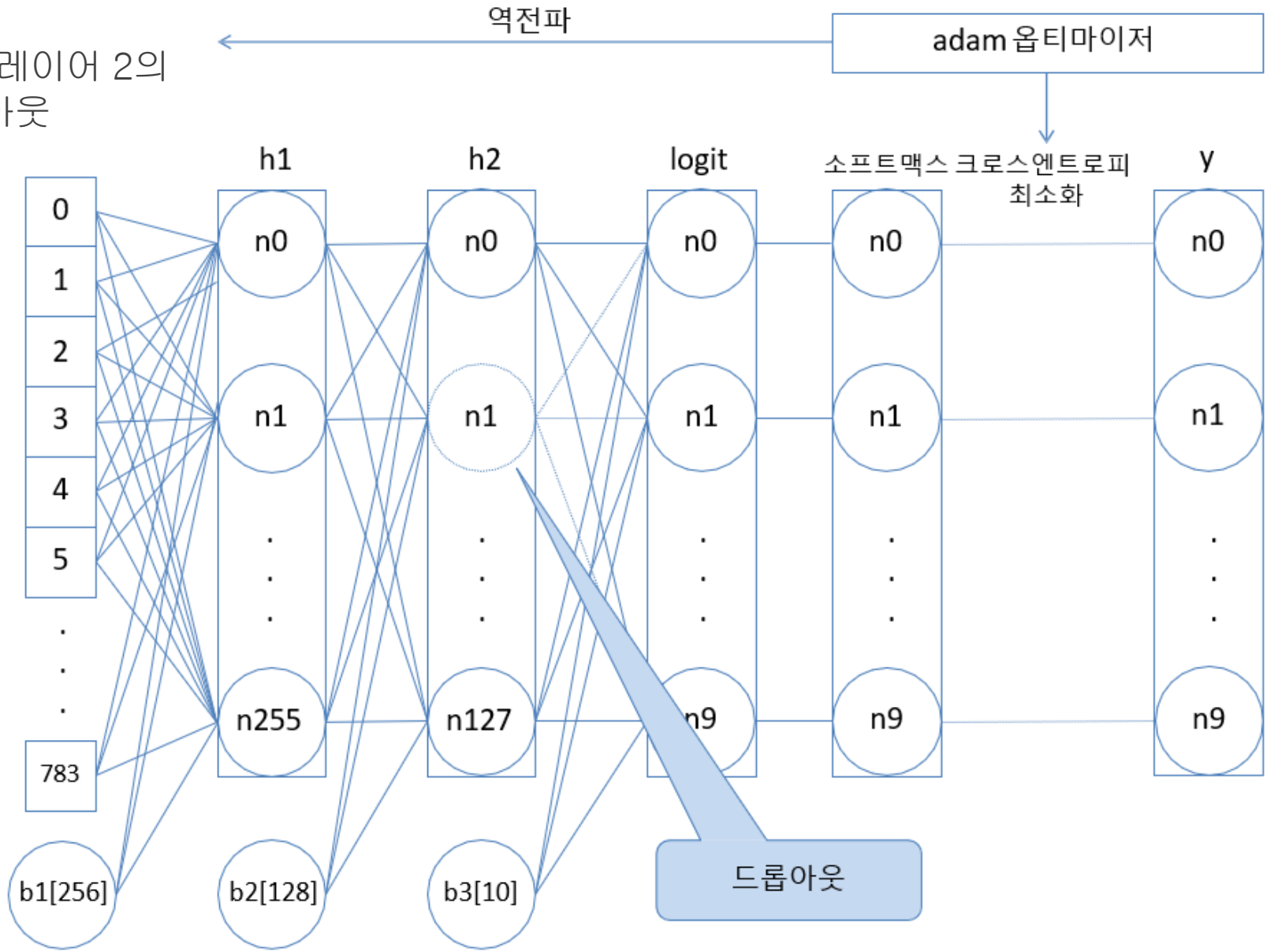
- decay를 사용한 학습률 조정에는 한계가 있음.
- 첫째로 딥러닝의 상당히 많은 가중치에동일한 학습률을 적용하게 됨. 각 가중치마다 저마다의 역할이 있는데, 동일한 학습률을적용한다는 것은 지나친 일반화일 수 있음.
- 둘째로, 학습률 조정이 탄력적이지 않고, 최초에는 로컬 미니멈에서 멀다가 학습이 어느 정도 지난 후 로컬 미니멈에 가까이 왔다는 가설이 언제나 맞지는 않음
- Adagrad는 각 매개변수에 각기 다른 학습률을 적용하고, 빈번히 변화가 찾아오는 가중치는학습률이 작게 설정되고, 변화가 적은 가중치는 학습률이 높게 설정되는 옵티마이저.
- 따라서 Adagrad는 개발자가 직접 decay를 신경 쓰지 않아도 모델 학습에 따라 모델이 알아서 가중치별로 학습률을 지정하게 됨.

Adam

- Adam 옵티마이저는 Adagrad의 학습률 자율 조정과 모멘텀의 효율적인 매개변수 변경 알고리즘을 조합한 알고리즘.
- 이러한 이유로 최근 들어 가장 많이 활용되는 옵티마이저.

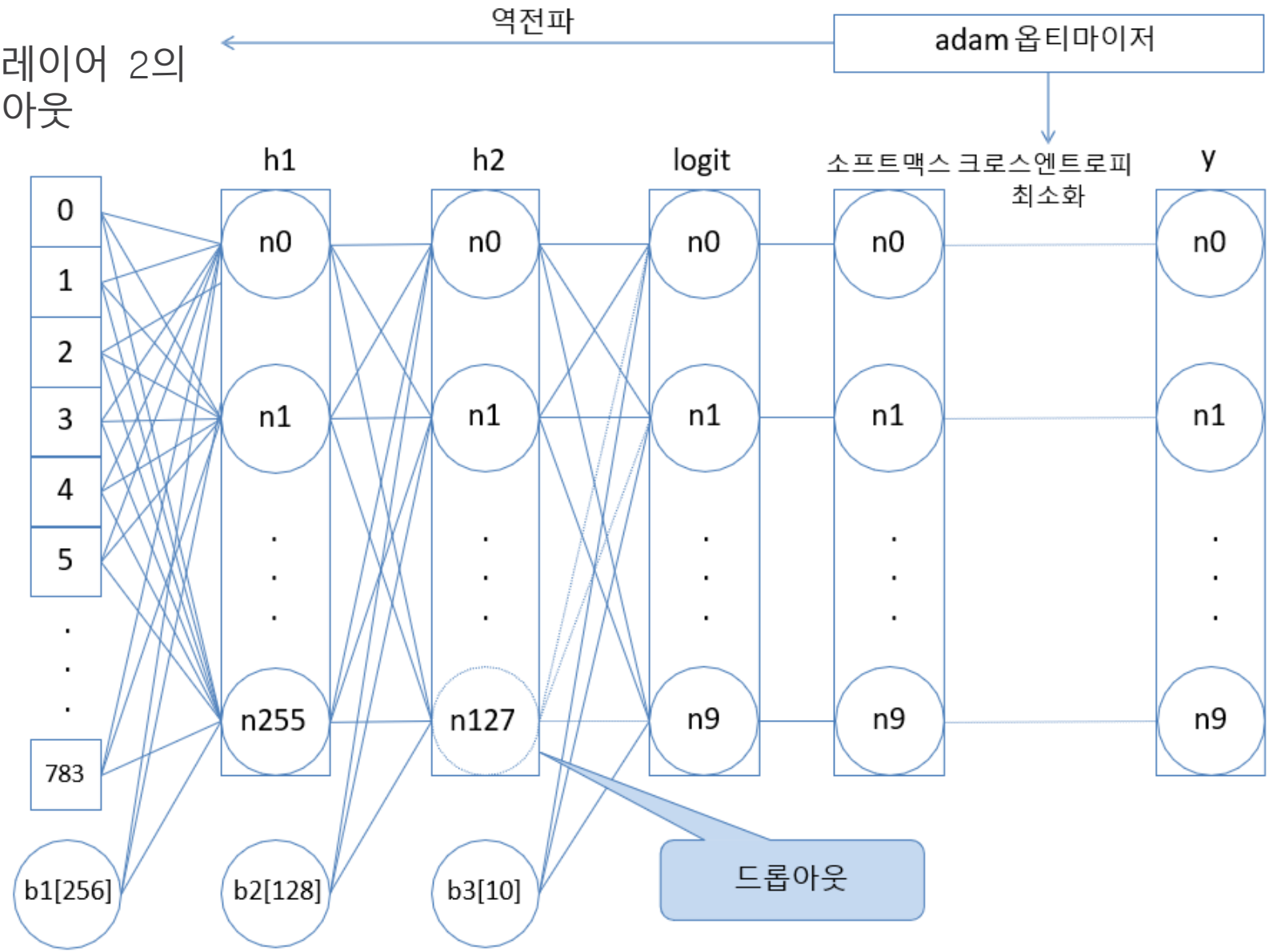
9 딥러닝의 과대적합

스텝 1. 히든 레이어 2의
노드 1 드롭아웃



9 딥러닝의 과대적합

스텝 2. 히든 레이어 2의 노드 127 드롭아웃



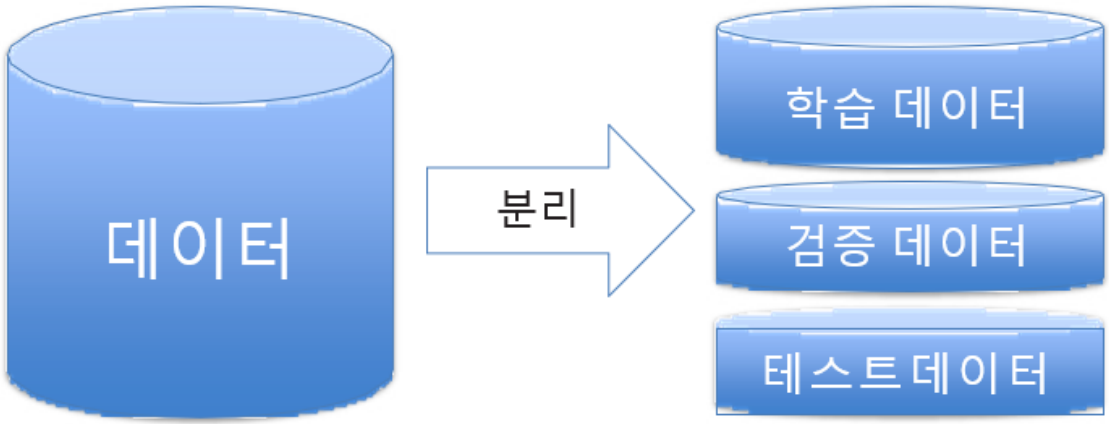
9 딥러닝의 과대적합

- 드롭아웃은 매개변수 중 일정량을 학습 중간마다 무작위로 사용하지 않는 방법.
- 매 스텝마다 드롭아웃이 설정된 히든 레이어에서 무작위로 선택된 노드가 학습에 사용되지않고 매개변수가 조절됨.
- 드롭아웃이 과대적합에 도움되는 이유가 무엇일까? 드롭아웃은 모델에 앙상블 효과를 줍니다.
- 이러한 방법은 모델의 분산을 줄이는 데 효과적.
- 드롭아웃을 사용해 소규모 노드들로 다양하게 여러 번 학습되어 결과적으로 과대적합의 위험을 줄일 수 있음.

9 딥러닝의 과대적합

9.1 조기종료

- 오버피팅을 줄일 수 있는 두 번째 방법으로 조기 종료(Early Stopping)가 있음.
- 학습을 진행할 때 최대 학습 반복 횟수를 설정하되, 모델이 과대적합될 소지가 있을 경우 학습을 중단하고 지금까지 학습된 모델 중에서 최고의 모델을 선택해야 됨.
- 딥러닝 학습에 조기 종료를 도입하기 위해 가장 먼저 해야 할 일은 데이터를 학습 데이터, 검증 데이터, 테스트 데이터로 분리하는 것.



데이터 분리

9 딥러닝의 과대적합

- 데이터를 분리할 때 학습 데이터, 검증 데이터, 테스트 데이터의 비율을 6:2:2 정도로 설정하면 적당.
- 딥러닝 모델을 학습할 때 오직 학습 데이터로만 모델의 매개변수를 조정.
- 매개변수를 조정한 후, 검증 데이터(validation set)로 모델의 정확도를 측정.

검증 정확도에 따른 조기 종료 시점