
인공신경망

Contents

교육 과정 개요

Part 1 인공신경망

- 1.1 신경망에서 인공신경망으로
- 1.2 인공신경망 학습

Part 2 인공신경망 구현

- 2.1 인공신경망 설계
- 2.2 인공신경망 설계 실습
- 2.3 인공신경망의 주요 함수
- 2.4 인공신경망 구현

Part 3 인공신경망 성능개선

- 3.1 인공신경망 성능개선
 - 3.2 데이터 정규화를 통한 성능 개선
-

인공신경망

1.1 신경망에서 인공신경망으로

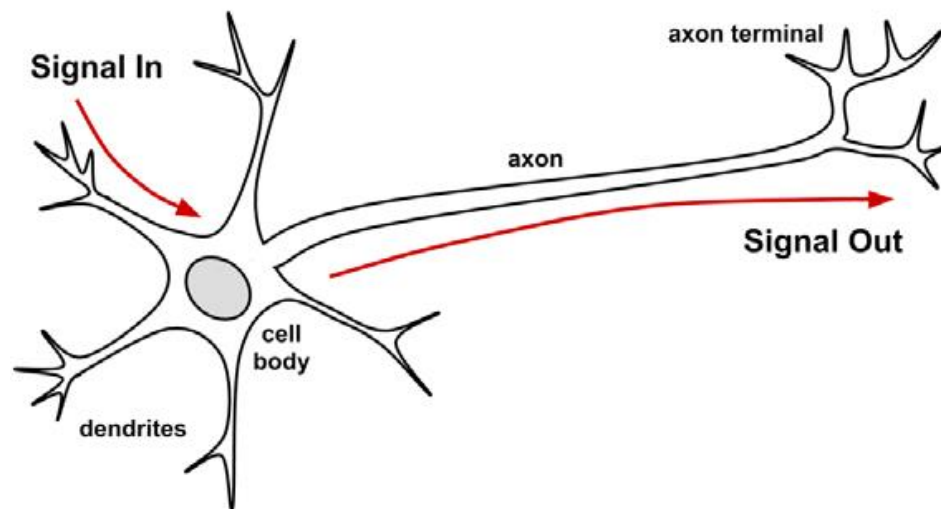
1.2 인공신경망 학습

신경망에서 인공신경망으로...

◆ 인공 신경망(ANN, Artificial Neural Network)

- 뇌가 어떻게 감각 입력의 자극에 반응하는지에 대한 이해로부터 얻어진 모델
- 입력 신호와 출력 신호 간의 관계를 모델링

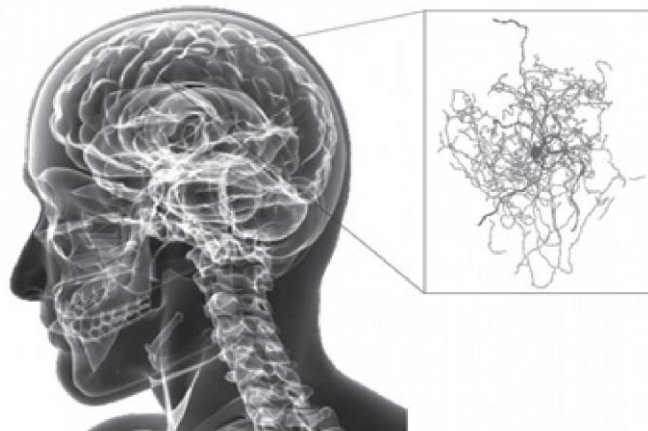
◆ 사람의 신경망이 어떻게 동작하는지 이해하면, 인공신경망을 학습시킬 수 있다!



신경망

◆ 신경망 특징 1: 많다!

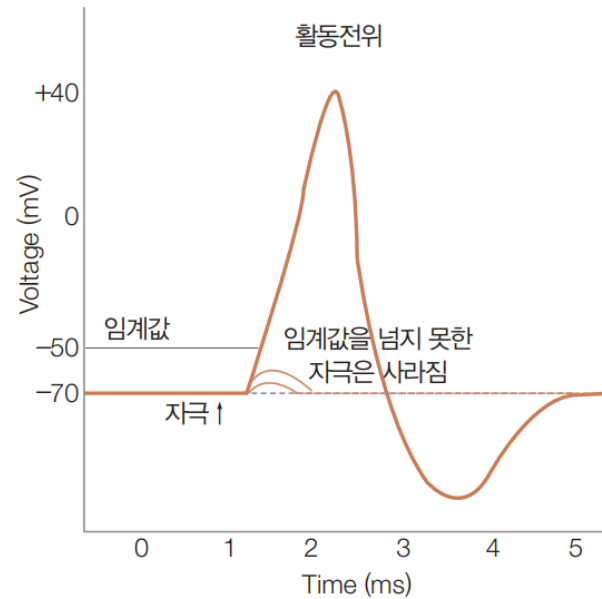
- 인간의 뇌: 약 860억 개의 뉴런



신경망

◆ 신경망 특징 2: 활성화!

- 자극의 정도 > 임계치
 - 자극이 다음 신경으로 전달

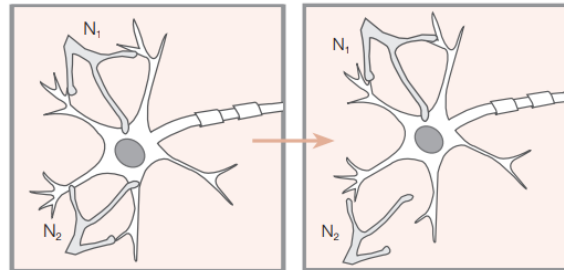


신경망

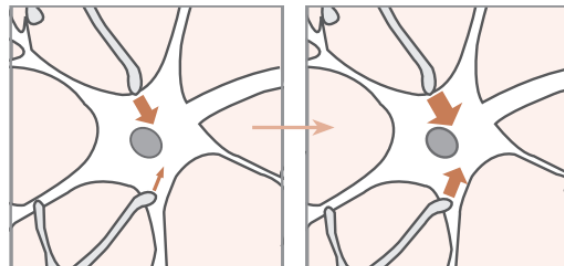
◆ 신경망 특징 3: 강화!

■ 헤브의 규칙(Hebb's Rule)

- 세포의 반복적이고 지속적인 자극에 따라 자극이 전달되는 효과가 증가



(a) 시냅스는 활성화에 의존하여 연결이 강화되기도 하고 약화되기도 한다. N_1 신경세포의 시냅스의 활성이 높아지면 시냅스가 추가적으로 생긴다. N_2 신경세포의 시냅스에서 활성이 사라지면 연결이 끊기게 된다.

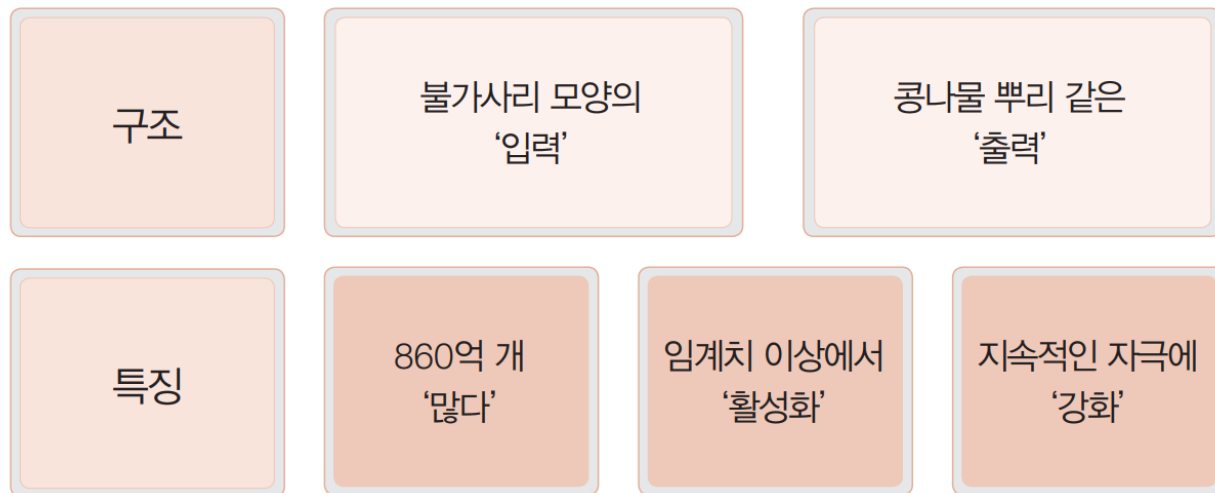


(b) 두 시냅스가 종종 동시에 활성화되면 두 시냅스에서 모두 시냅스 후 반응이 강화된다.

신경망

- ◆ 사람의 신경망이 어떻게 동작하는지 이해하면, 인공신경망을 학습시킬 수 있다!

사람의 신경망



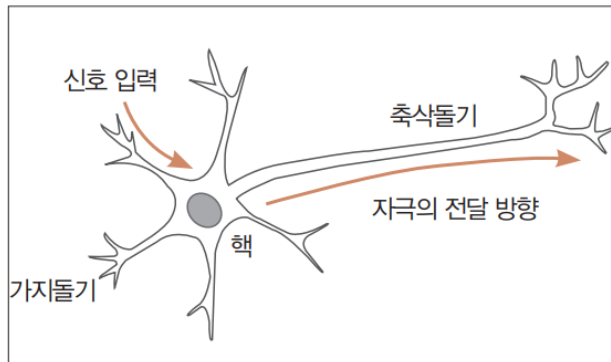
신경망에서 인공신경망으로...

◆ 생물학적 신경망을 추상화한 것이 인공신경망이다.

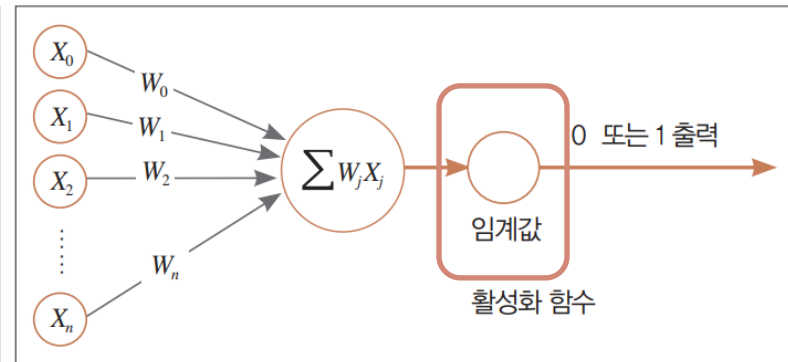
사람의 신경망



[생물학적 신경망]



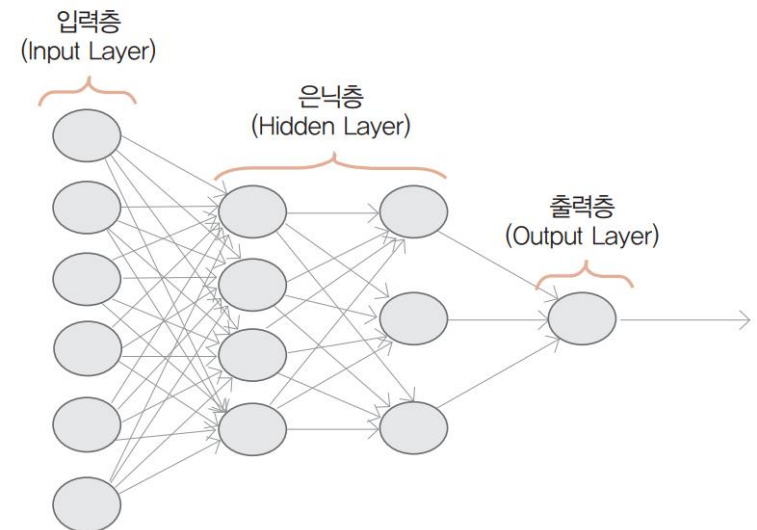
[인공 신경망]



신경망에서 인공신경망으로...

◆ 인공신경망의 활성화 함수, 계단함수(Step Function)

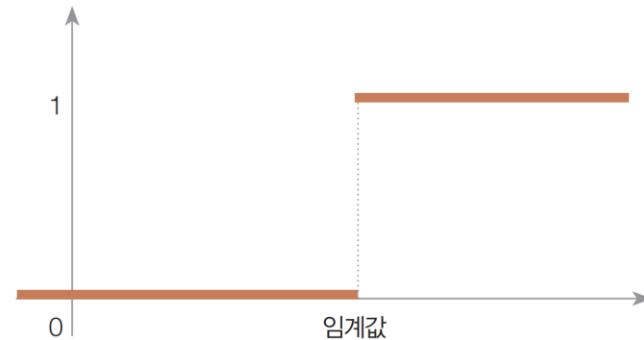
사람의 신경망



신경망에서 인공신경망으로...

◆ 인공신경망의 활성화 함수, 계단함수(Step Function)

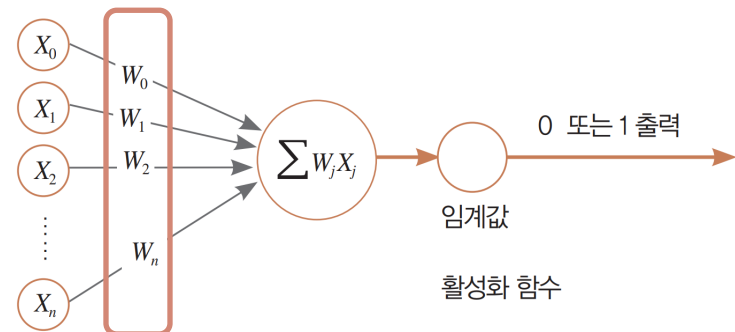
사람의 신경망



신경망에서 인공신경망으로...

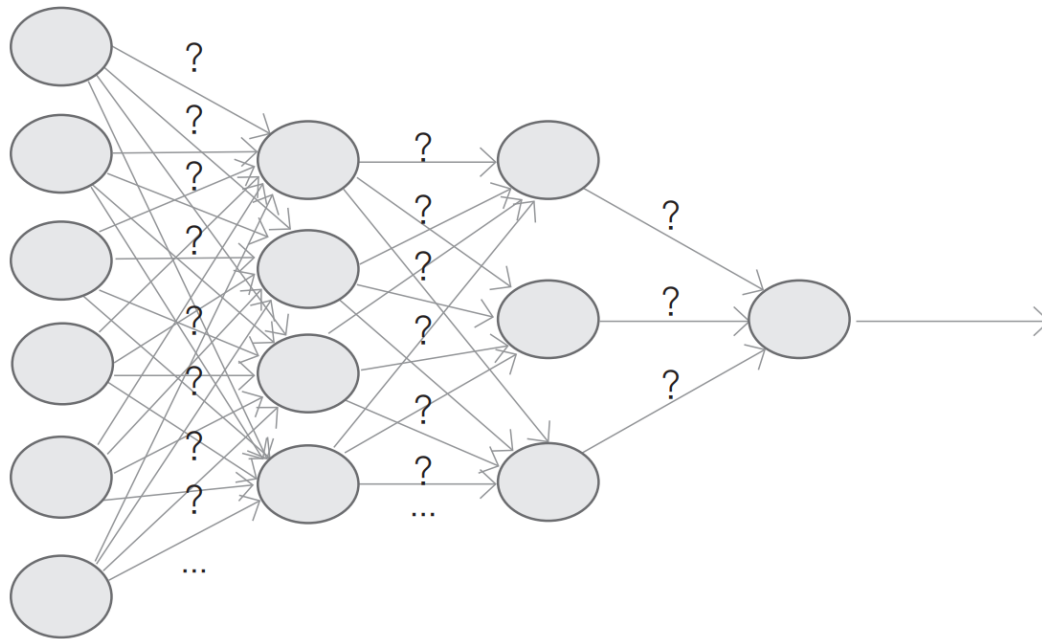
◆ 인공신경망의 활성화 함수, 계단함수(Step Function)

사람의 신경망



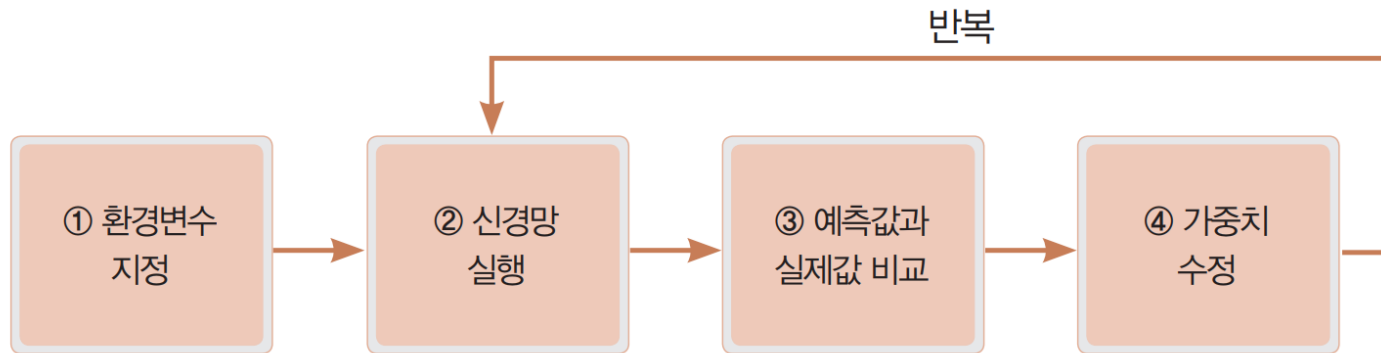
인공신경망의 학습

◆ 인공신경망의 학습은 가중치를 찾아내는 과정!



인공신경망의 학습

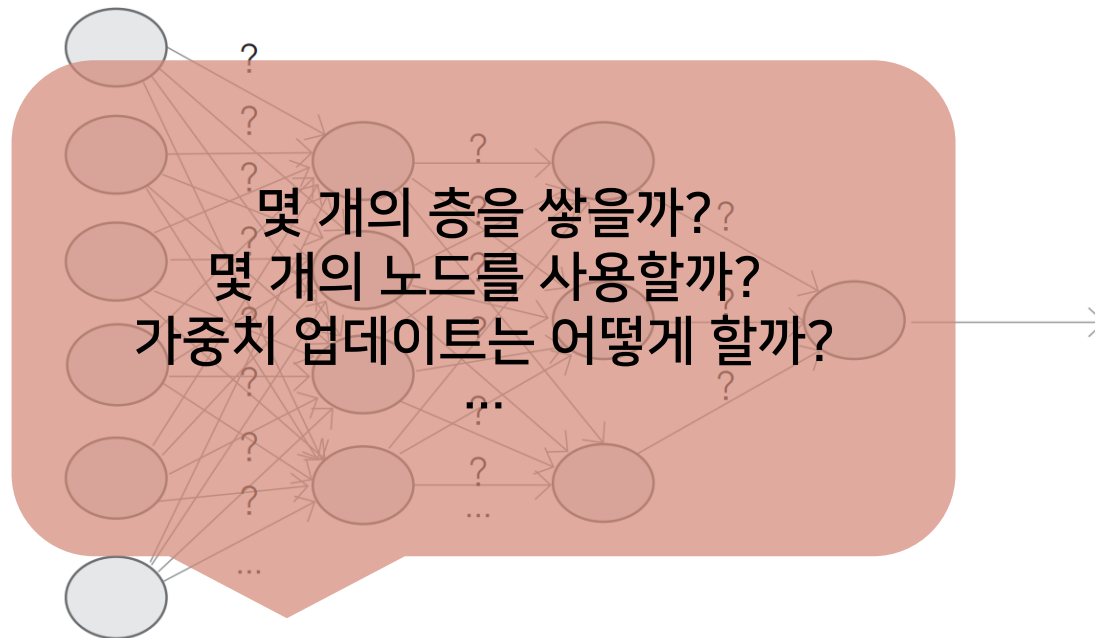
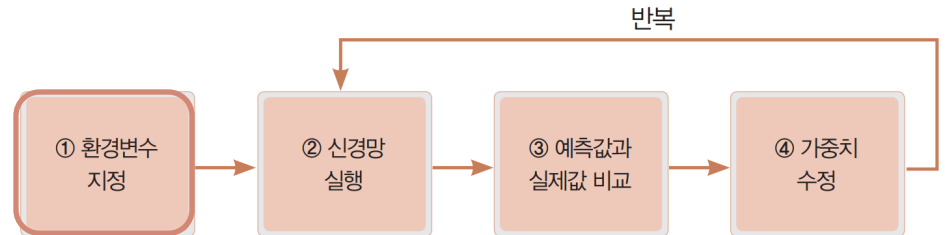
◆ 인공신경망 학습 프로세스



인공신경망의 학습

◆ 인공신경망 학습 프로세스

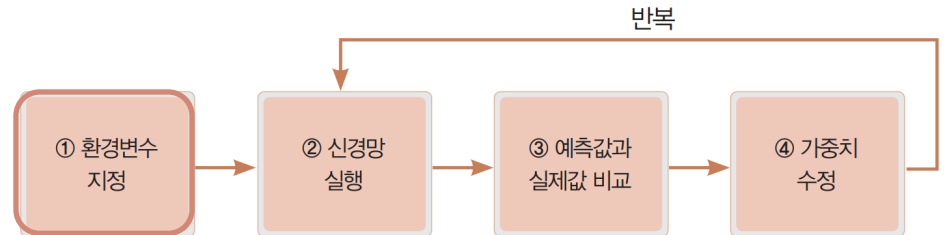
■ 환경변수 지정



인공신경망의 학습

◆ 인공신경망 학습 프로세스

■ 환경변수 지정



- 망 구성

- 신경망의 성능은 뉴런의 연결 구조 및 연산 방법에 따라 차이가 발생
- 망 형태의 3가지 특성
 - » 층(layer)의 개수
 - » 망에서 정보가 뒷단(backward)으로 전파될 수 있는지
 - » 망의 각 층 내에 있는 노드 개수
 - » 일반적으로 한 층의 모든 노드가 다음 층의 모든 노드와 완전 연결 됨
- 토폴로지는 망이 학습해야 할 태스크의 복잡성에 좌우됨

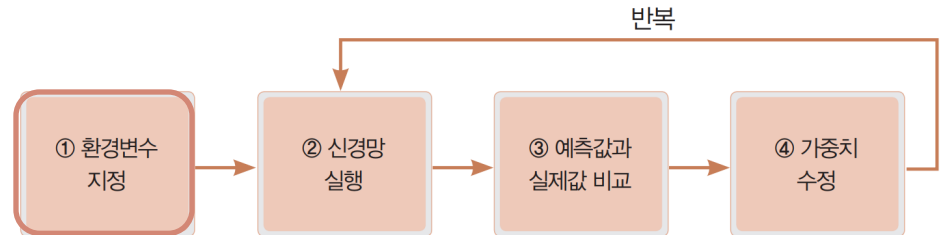
- 층의 개수

- 층의 개수와 정보 진행 방향과 더불어, 각 층의 노드 개수에 따라 복잡성이 다양해짐
- 입력 노드의 수는 입력 데이터의 속성 개수로 결정
- 출력 노드의 수는 결과의 분류 개수나 모델의 결과 수로 결정
- 모델을 시작하기 전 은닉층의 노드 개수를 결정하지만, 적당한 은닉층의 노드 개수를 결정하는 규칙은 없음

인공신경망의 학습

◆ 인공신경망 학습 프로세스

■ 환경변수 지정



- 하이퍼파라미터(Hyper Parameter)

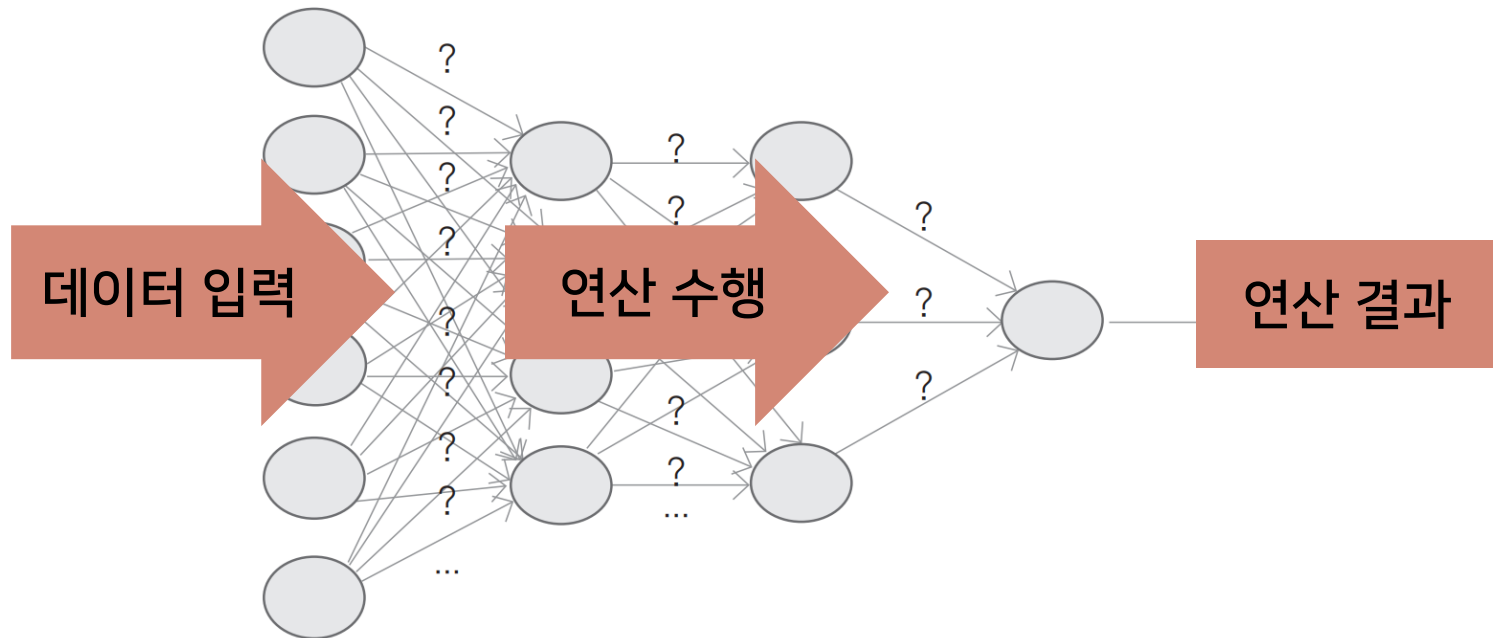
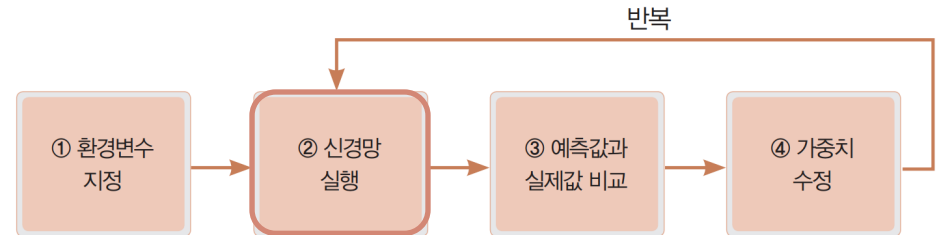
- 신경망의 학습에 의해서 자동으로 획득되지 않고, 사람이 직접 설정해야 하는 값
- 에폭(Epoch)
 - » 딥러닝을 수행하면서 학습데이터가 모두 소진되는 하나의 단위
 - » 1 epoch은 전체 데이터 셋에 대해 한 번 학습을 완료
 - » 예) 10,000개를 100개의 batch_size로 학습할 때, 100회가 1epoch
 - » 모든 데이터 셋에 대해 역전파(backpropagation)을 수행함
 - » 적절한 epoch 설정으로 underfitting과 overfitting 방지
- 배치사이즈 (Batch size)
 - » 한 번의 batch(mini-batch)마다 주는 데이터의 sample size
 - » batch size와 성능 간의 상관관계는 없으나 메모리 한계와 속도 저하 때문에 한번의 epoch에서 모든 데이터 학습 불가능

인공신경망의 학습

◆ 인공신경망 학습 프로세스

■ 신경망 실행

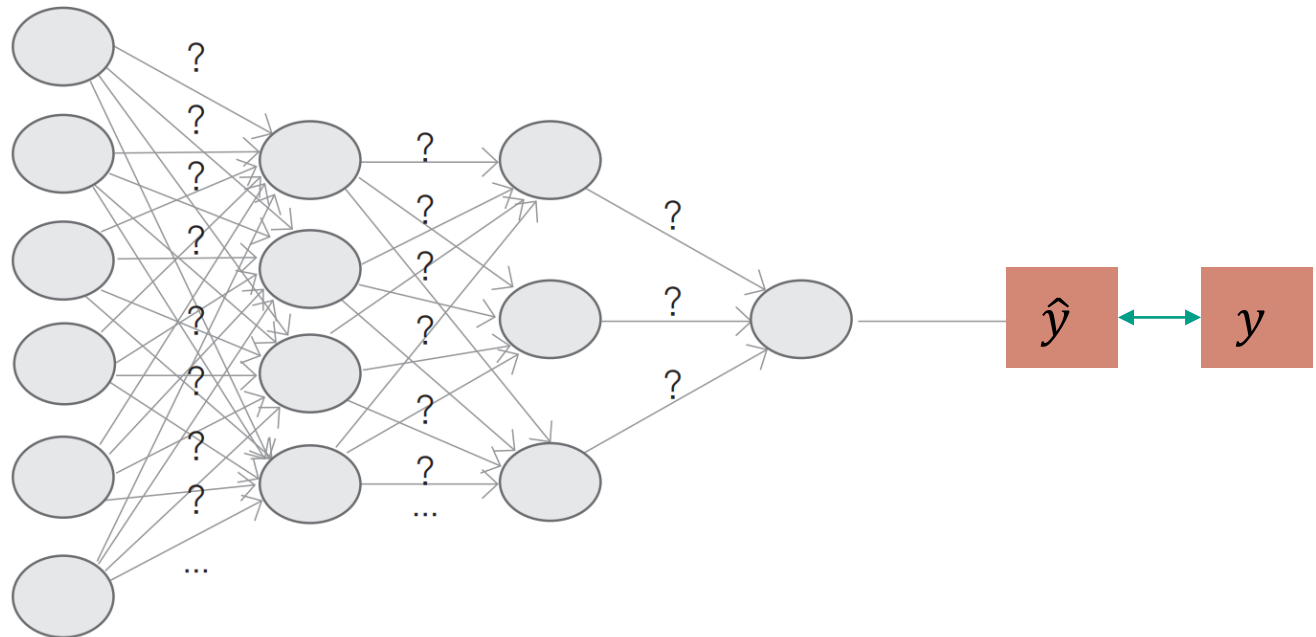
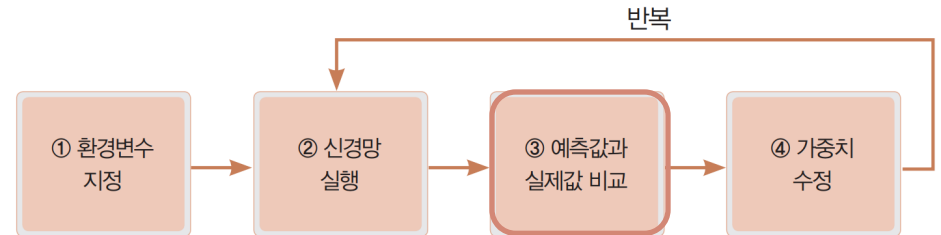
- Feedforward Neural Network(FNN)



인공신경망의 학습

◆ 인공신경망 학습 프로세스

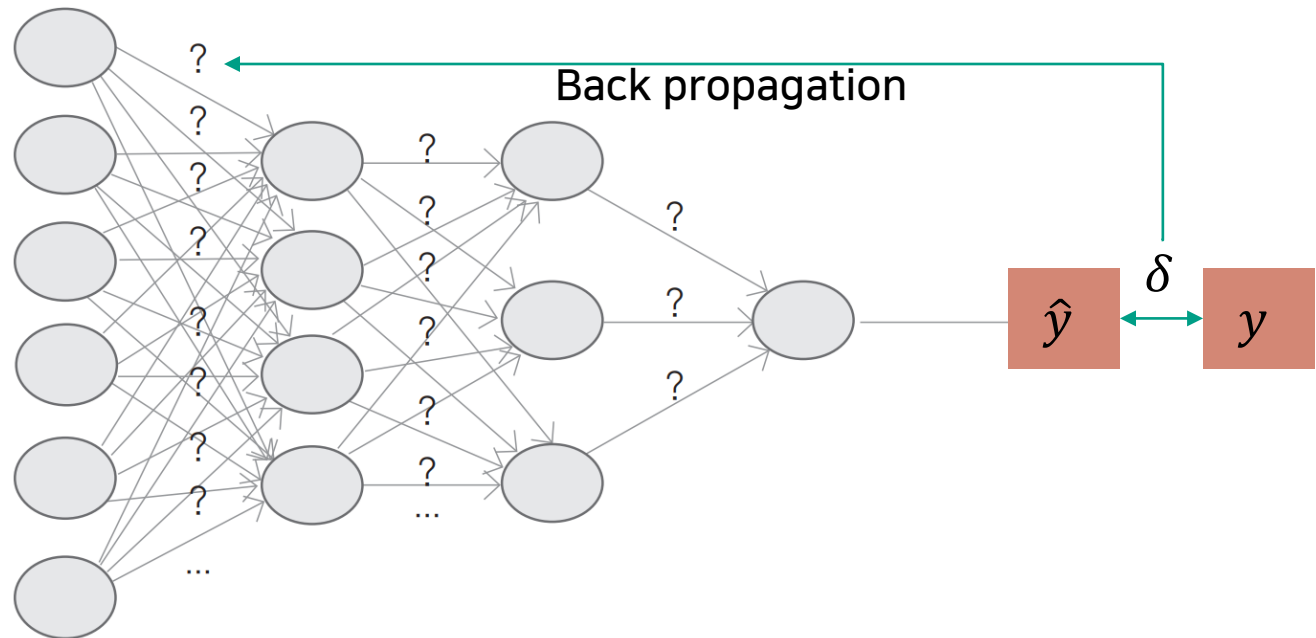
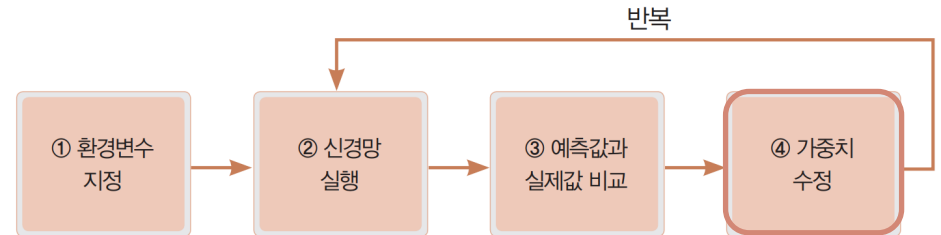
- 예측값과 실제값 비교



인공신경망의 학습

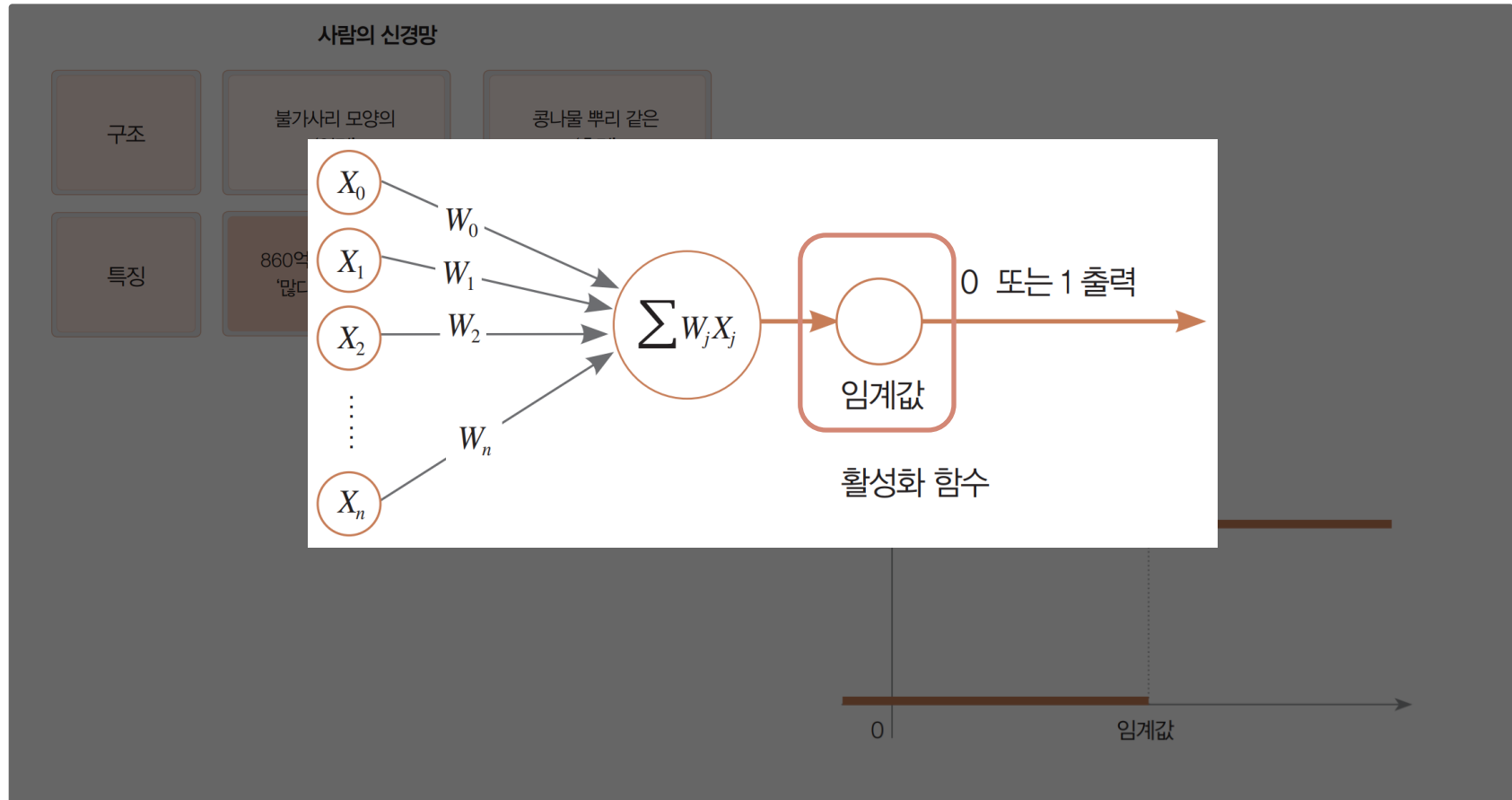
◆ 인공신경망 학습 프로세스

■ 가중치 수정



인공신경망의 학습

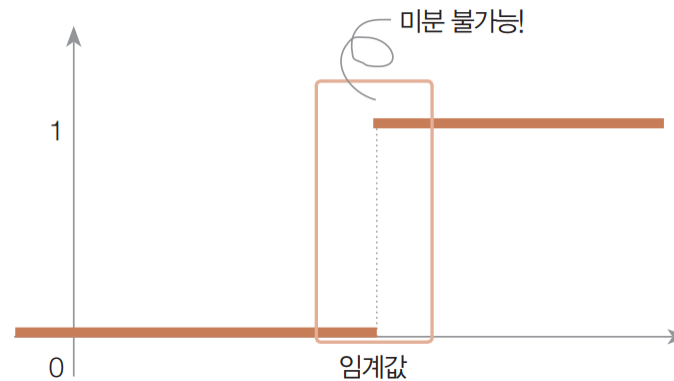
◆ 활성화함수(Activation Function)



인공신경망의 학습

◆ 활성화함수(Activation Function)

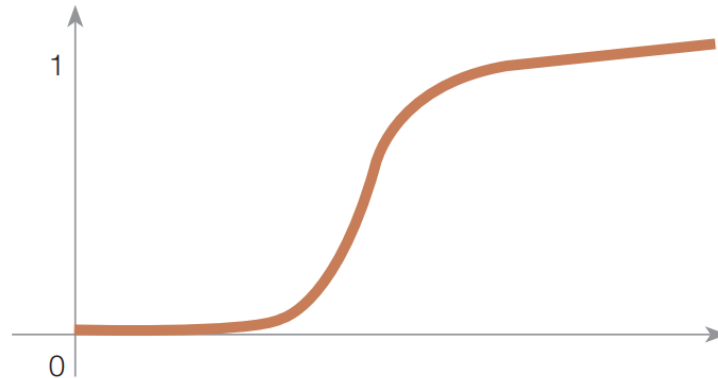
■ 계단함수(Step Function)



인공신경망의 학습

◆ 활성화함수(Activation Function)

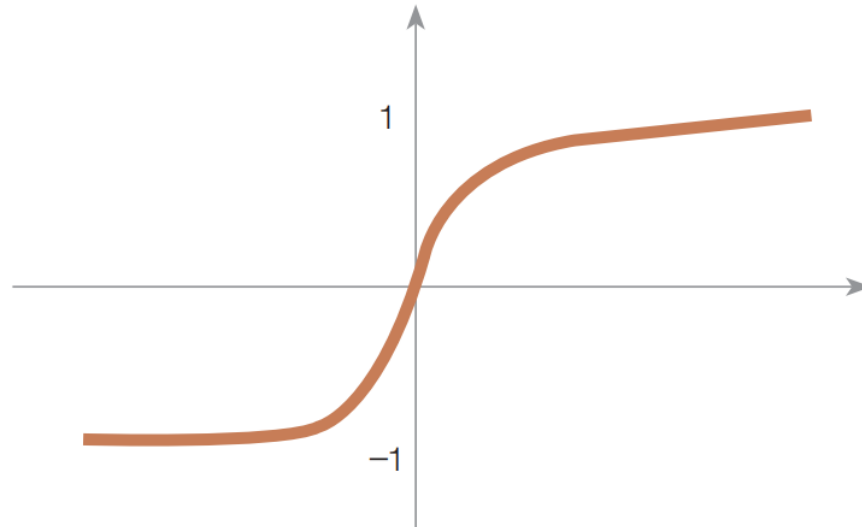
- 시그모이드(Sigmoid)



인공신경망의 학습

◆ 활성화함수(Activation Function)

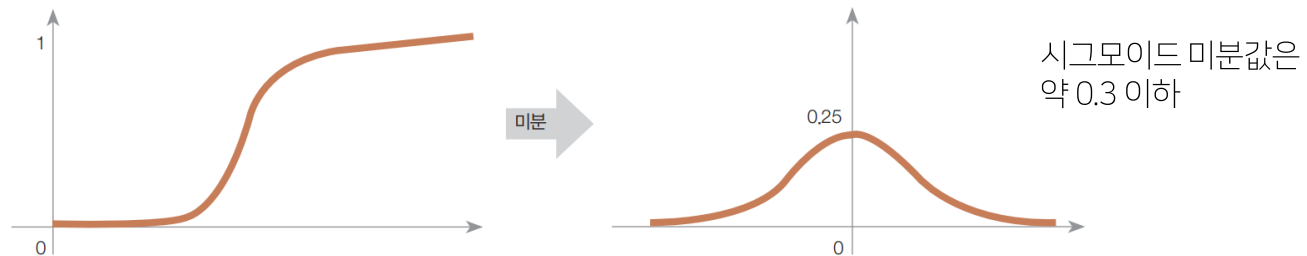
- \tanh



인공신경망의 학습

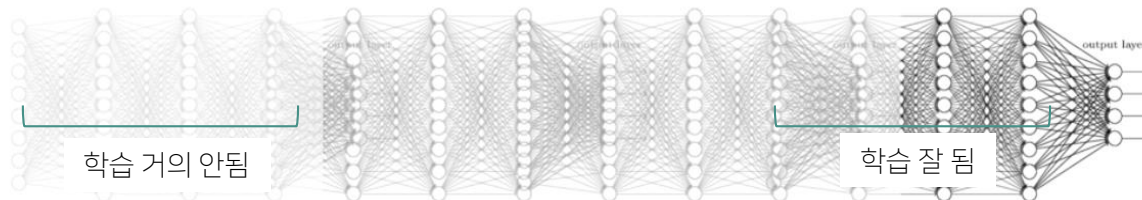
◆ 활성화함수(Activation Function)

■ 시그모이드(Sigmoid)



■ 기울기 소멸 문제(Vanishing Gradient Problem)

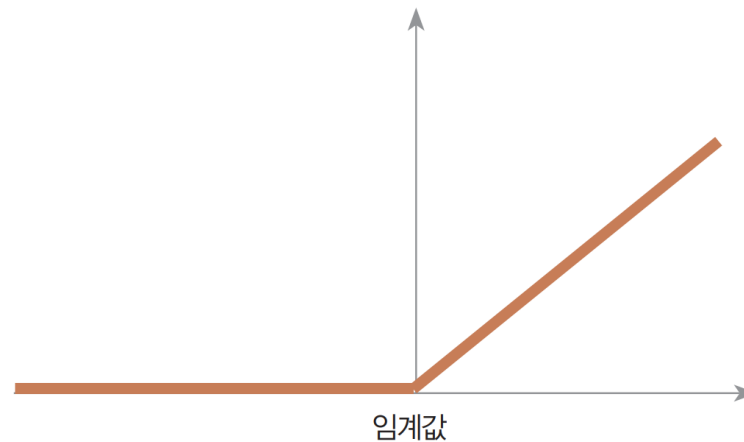
- 역전파 시, 은닉층으로 오차가 거의 전달 되지 않아 학습이 제대로 되지 않는 문제
- 은닉층을 여러 단계 거치는 딥러닝에서 가중치를 수정하기 위해 미분을 반복적으로 사용하면서 기울기가 0이 되어버려 학습이 중단
 - sigmoid나, tanh 등: input을 매우 작은 output range로 짓이겨 넣는('squash') 형태의 연산



인공신경망의 학습

◆ 활성화함수(Activation Function)

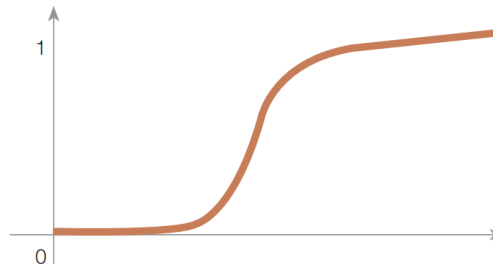
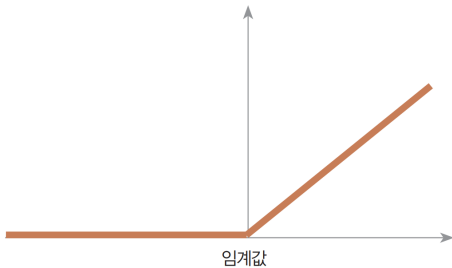
■ 렐루(ReLU)



인공신경망의 학습

◆ 출력층의 활성화함수(Activation Function)

- 렐루(ReLU), 시그모이드(Sigmoid), 소프트맥스(Softmax)



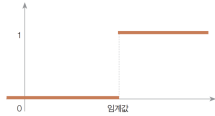
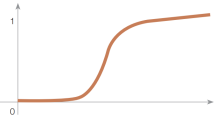
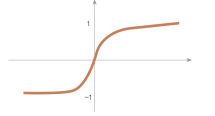
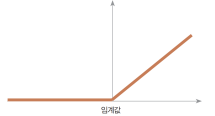
Softmax 확률

$$\frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \rightarrow \begin{bmatrix} 0.02 \\ 0.90 \\ 0.05 \\ 0.01 \\ 0.02 \end{bmatrix}$$

인공신경망의 학습

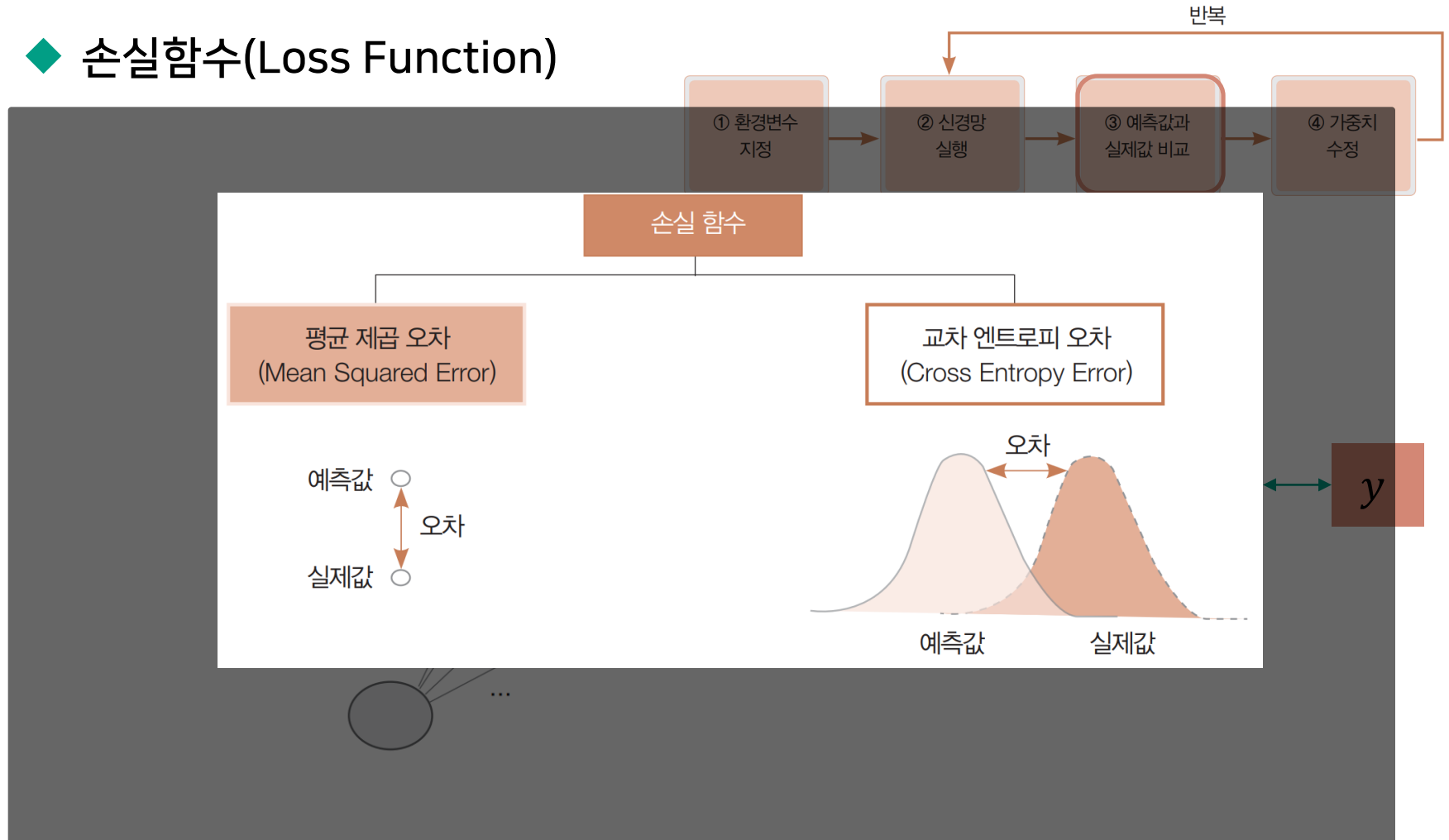
◆ 활성화함수(Activation Function)

- 입력 신호의 총합을 그대로 사용하지 않고, 입력 신호의 총합이 활성화를 일으키는지 아닌지를 정하는 역할로, 입력 신호를 규칙에 따라 출력 신호로 변환하는 함수

유형	도식	설명
계단함수(step function)		<ul style="list-style-type: none"> 양극성 이진 함수 디지털 형태의 출력
시그모이드 함수		<ul style="list-style-type: none"> 비선형 연속 함수
tanh함수		<ul style="list-style-type: none"> LSTM, GRU의 활성화함수로 많이 사용
ReLU함수		<ul style="list-style-type: none"> 학습속도가 빠르고 학습이 잘 되어 가장 많이 사용
Softmax 함수	$\text{Softmax} \quad \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \rightarrow \begin{bmatrix} 0.02 \\ 0.90 \\ 0.05 \\ 0.01 \\ 0.02 \end{bmatrix}$ <div style="display: flex; justify-content: space-around; margin-top: 5px;"> Softmax 확률 </div>	<ul style="list-style-type: none"> 출력층에서 사용 출력의 결과로 벡터값을 얻고 싶을 때

인공신경망의 학습

◆ 손실함수(Loss Function)



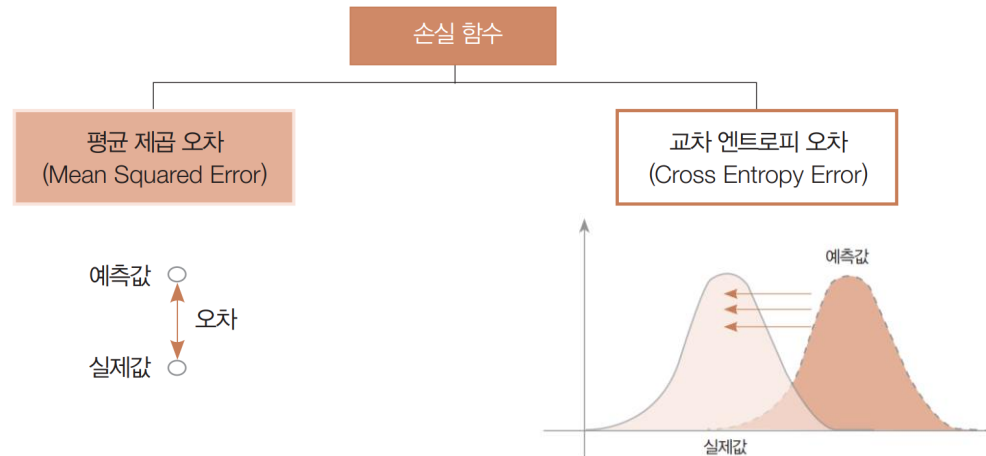
인공신경망의 학습

◆ 손실 함수(Loss Function)

- 비용함수(Cost Function)
- 가중치에 따라 실제값과 예측값의 오차가 어느 정도인지를 평가
- 신경망 성능의 “나쁨”을 나타내는 지표
 - 현재의 신경망이 훈련 데이터를 얼마나 잘 처리하지 못하느냐를 나타내는 지표
 - 손실함수 값을 작게 하는 매개변수(가중치와 편향)를 찾는 과정이 인공신경망의 학습
 - 미분 : 매개변수 값을 아주 조금 변경했을 때 손실 함수가 어떻게 변하나
- 경사하강법(Gradient descent algorithm)을 이용하여 loss 최소화

인공신경망의 학습

◆ 손실 함수(Loss Function)

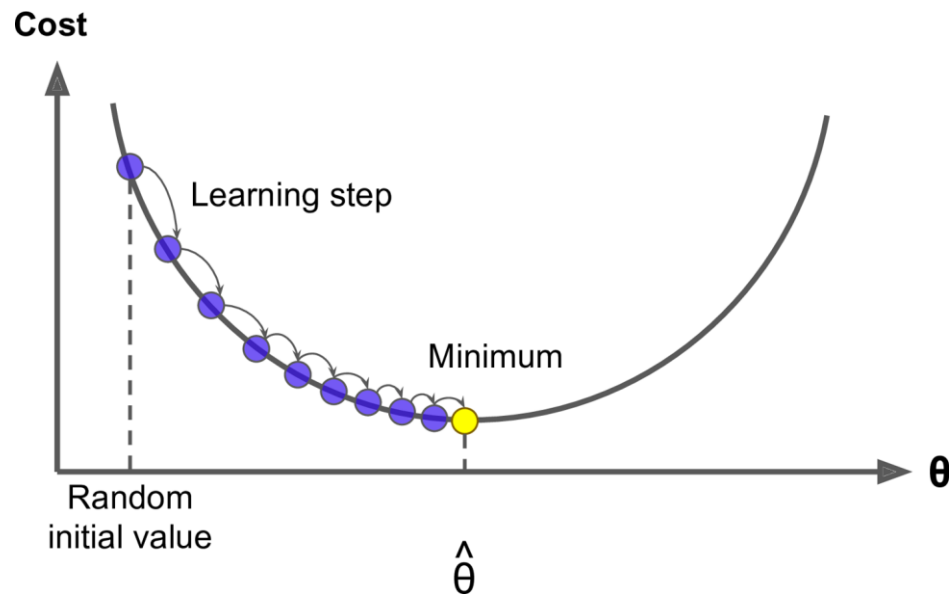


손실함수	설명	설명
평균제곱오차 (Mean Squared Error, MSE)	각 원소의 출력(추정) 값(\hat{y}_i)과 정답 레이블(참) 값(y_i)의 차를 제곱한 총합	$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
교차 엔트로피 오차 (Cross Entropy Error, CEE)	특정 클래스에 속할 정보량의 확률의 합 범주형 데이터를 분류할 때 주로 사용	$-\sum_i y_i \log(\hat{y}_i)$

인공신경망의 학습

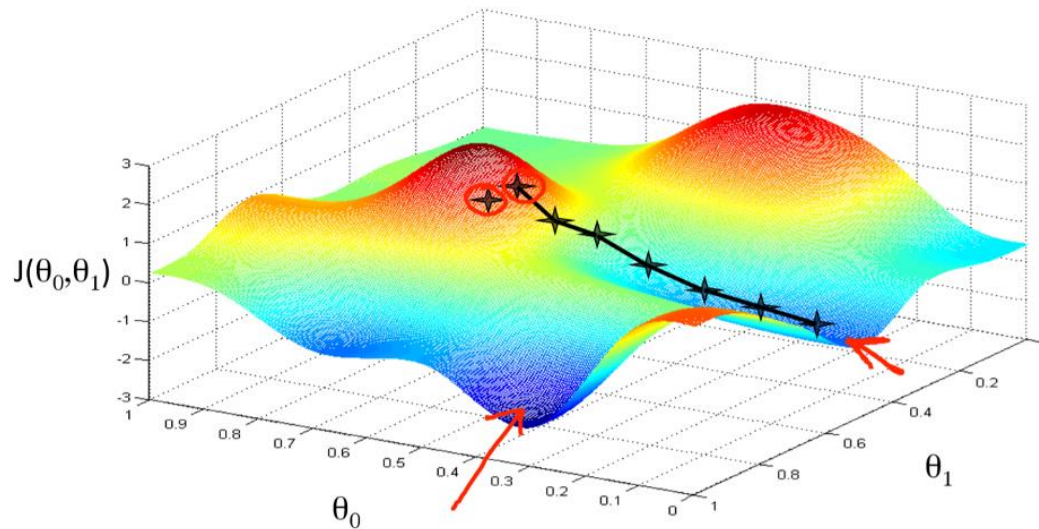
◆ 경사하강법(Gradient Descent Algorithm)

- 가중치에 대한 손실함수의 최소값의 위치를 찾기 위해 손실함수를 미분하고, 그 미분값의 방향과 크기를 활용해 가중치를 보상하는 방법
- 손실 함수의 최소값을 찾는 옵티마이저 (Optimizer)의 한 유형
 - 신경망의 연결 가중치 최적화



인공신경망의 학습

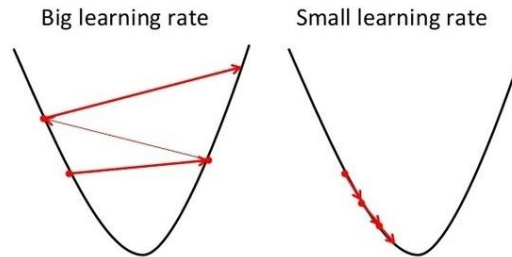
◆ 경사하강법(Gradient Descent Algorithm)



인공신경망의 학습

◆ 학습률(Learning Rate)

- 경사하강법에서 손실함수의 최소값의 위치를 찾기 위한 이동하는 거리의 비율
- 경사하강법은 오차의 변화에 따라 이차 함수 그래프를 만들고, 적절한 학습률을 설정해 미분값이 0인 지점을 구함
- 학습률을 너무 크게 설정한 경우
 - 손실함수의 최소값을 찾지 못하고 값이 발산(explode)하거나 소실(vanish)하는 문제 발생
 - local minima에 빠지는 문제 발생
- 학습률을 너무 작게 설정한 경우
 - 학습시간이 오래 소요되고, 최소값에 미쳐 도달하지 못한 상태에서 학습이 종료



인공신경망의 학습

◆ 옵티마이저(Optimizer)

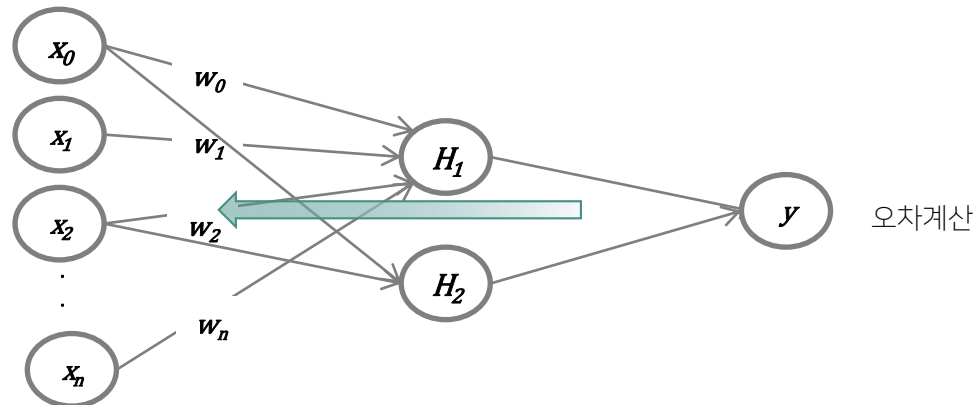
- 손실함수의 최소값을 찾기 위한 다양한 경사하강법(Gradient Descent Algorithm)

기법	설명	특징
확률적 경사 하강법 (SGD)	매개변수의 기울기를 구해, 기울어진 방향으로 매개변수 값을 갱신하는 일을 반복	(단점)급격한 변곡점이 있는 경우 SGD 한계
모멘텀 (Momentum)	기존에 사용한 기울기의 일정 비율(%)을 현재 기울기에 반영	이전 그래디언트의 관성, 가속도기반 보정
Adagrad (Adaptive Gradient)	변수의 업데이트 회수에 따라 학습률을 조절	학습을 진행하면서 학습률을 점차 줄여가는 방법(Learning rate decay)
Adam	모멘텀 + Adagrad	관성, 탄력, 가속도, 학습률조절

인공신경망의 학습

◆ 역전파(Backpropagation)

- 정방(Feedforward) 연산 이후, 예측값과 실제값과의 오차를 후방(Backward)으로 다시 보내줌으로써, 최적의 Weight와 Bias를 학습하는 기법
 - 가중치 매개변수의 기울기는 수치 미분 → 단순하고 구현 쉬우나 계산 시간 오래걸림
 - 빠르게 계산하기 위해 오차역전파(Back propagation)사용

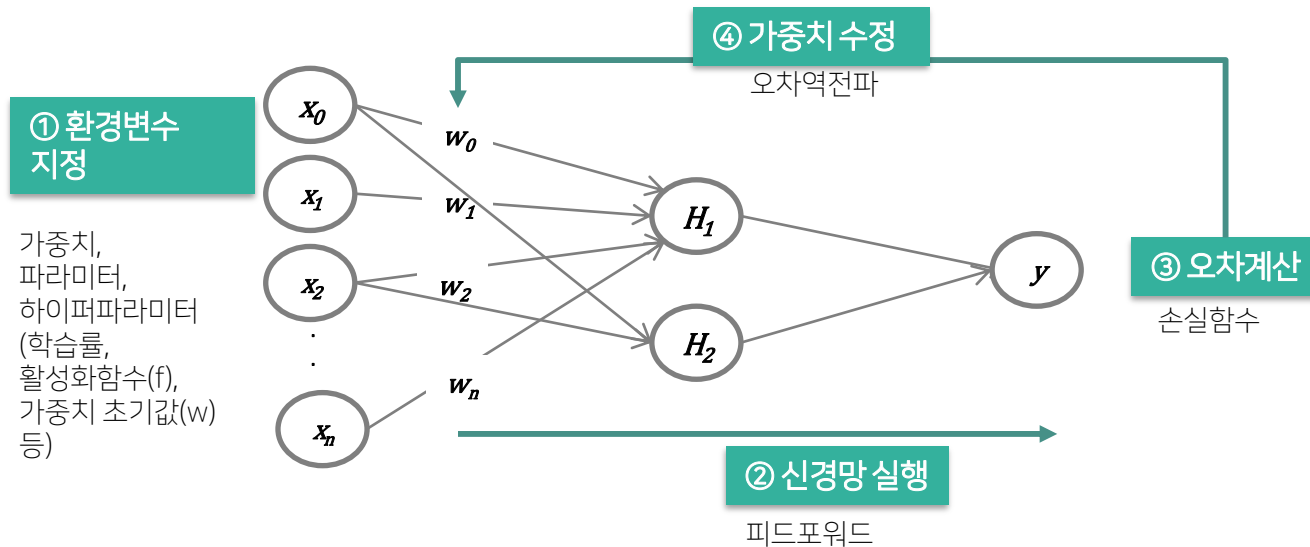


Summary

Summary

◆ 신경망 학습 매커니즘

- 신경망은 각 은닉층에서 활성화함수를 이용하여 각 층의 연산결과를 여러 층에 걸쳐 전달하여 학습을 진행
- 각 은닉층의 가중치 매개변수의 최적값을 자동으로 획득
- 신경망의 학습 목표는 가중치의 매개변수를 최적화 하여 손실함수의 결과값을 가장 작게 하는 것



Summary

◆ 용어체크

- Perceptron
- Hidden layer
- MLP
- Back propagation
- Feedforward Neural Network
- Activation Function
- Loss Function
- Gradient Descent Algorithm
- Learning Rate
- Vanishing Gradient
- Overfitting
- Drop out
- Epoch
- Batch size

[01강] 학습 내용 확인하기

[01강] 학습 내용 확인하기

문제1. 다음 중 신경망의 학습에 의해서 자동으로 획득되지 않고, 사람이 직접 설정해야 하는 값은?

- ① 파라미터
- ② 가중치
- ③ 편향
- ④ 하이퍼파라미터

문제2. 다음 중 다중 분류 문제의 활성화 함수로 올바른 것은?

- ① 계단함수
- ② 시그모이드
- ③ 소프트맥스
- ④ 렐루

문제3. 예측값과 예측값과 실제값과의 오차를 후방(Backward)으로 다시 보내줌으로써, 최적의 가중치와 편향을 찾는 방법은?

- ① 순전파
- ② 역전파
- ③ 손실함수
- ④ 활성화함수

[01강] 학습 내용 확인하기

문제1. 다음 중 신경망의 학습에 의해서 자동으로 획득되지 않고, 사람이 직접 설정해야 하는 값은?

- ① 파라미터
- ② 가중치
- ③ 편향
- ④ 하이퍼파라미터

문제2. 다음 중 다중 분류 문제의 활성화 함수로 올바른 것은?

- ① 계단함수
- ② 시그모이드
- ③ 소프트맥스
- ④ 렐루

문제3. 예측값과 예측값과 실제값과의 오차를 후방(Backward)으로 다시 보내줌으로써, 최적의 가중치와 편향을 찾는 방법은?

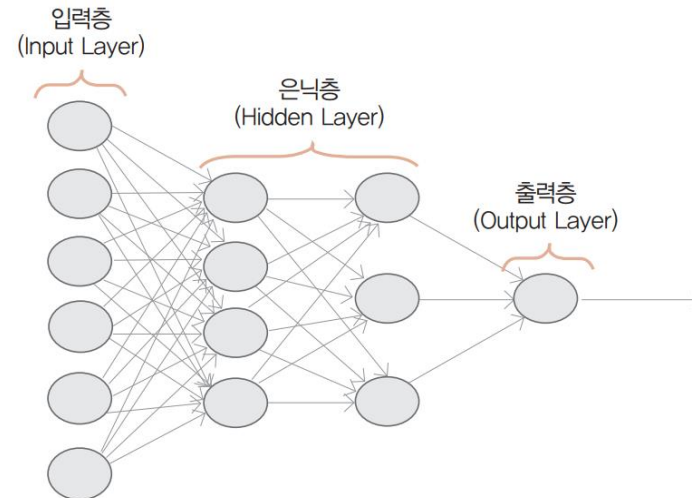
- ① 순전파
- ② 역전파
- ③ 손실함수
- ④ 활성화함수

인공신경망 구현

- 2.1 인공신경망 설계
 - 2.2 인공신경망 설계 실습
 - 2.3 인공신경망의 주요 함수
 - 2.4 인공신경망 구현
-

인공신경망 설계

◆ 인공신경망의 구성 요소



Layer(층)			
각 Layer의 노드 수			
활성화함수			
손실함수			
옵티마이저			

인공신경망 설계

◆ 붓꽃 데이터 분류를 위한 인공신경망



Iris Versicolor



Iris Setosa



Iris Virginica

이미지 출처: <http://www.lac.inpe.br/~rafael.santos/Docs/CAP394/WholeStory-Iris.html>

- sepal : 꽃받침
- petal : 꽃잎

인공신경망 설계

◆ 붓꽃 데이터 분류를 위한 인공신경망

- 3종의 붓꽃 데이터(Versicolor, Setosa, Virginica)
- 4가지 속성(꽃받침 길이/너비, 꽃잎 길이/너비) 150개

150개

4가지 속성				3종	
꽃받침 길이	꽃받침 너비	꽃잎 길이	꽃잎 너비	품종	
sepal_length	sepal_width	petal_length	petal_width	species	
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

150개

인공신경망 설계

◆ 붓꽃 데이터 분류를 위한 인공신경망

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Layer(층)	Input	Hidden	Output
각 Layer의 노드 수	4	8	3
활성화함수		relu	softmax
손실함수	sparse_categorical_crossentropy		
옵티마이저	adam		

인공신경망 설계 실습 #1

◆ 주어진 데이터

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	293	1	3.80	2.80	0	0	0	0	0	0	12	0	0	0	1	0	62	0
1	1	2	2.88	2.16	1	0	0	0	1	1	14	0	0	0	1	0	60	0
2	8	2	3.19	2.50	1	0	0	0	1	0	11	0	0	1	1	0	66	1
3	14	2	3.98	3.06	2	0	0	0	1	1	14	0	0	0	1	0	80	1
4	17	2	2.21	1.88	0	0	1	0	0	0	12	0	0	0	1	0	56	0

◆ 어떤 문제인가요?

- 종속변수가 0,1 두 가지 유형만 있음, 이진 분류 문제

인공신경망 설계 실습 #1

◆ 딥러닝 모델 설계

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	293	1	3.80	2.80	0	0	0	0	0	0	12	0	0	0	1	0	62	0
1	1	2	2.88	2.16	1	0	0	0	1	1	14	0	0	0	1	0	60	0
2	8	2	3.19	2.50	1	0	0	0	1	0	11	0	0	1	1	0	66	1
3	14	2	3.98	3.06	2	0	0	0	1	1	14	0	0	0	1	0	80	1
4	17	2	2.21	1.88	0	0	1	0	0	0	12	0	0	0	1	0	56	0

Layer(층)	Input	Hidden	Output
각 Layer의 노드 수	17	30	1
활성화함수		relu	sigmoid
손실함수	binary_crossentropy		
옵티마이저	adam		

인공신경망 설계 실습 #2

◆ 주어진 데이터

	0	1	2	3	4	5	6	7	8
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

◆ 어떤 문제인가요?

- 종속변수가 0,1 두 가지 유형만 있음, 이진 분류 문제

인공신경망 설계 실습 #2

◆ 딥러닝 모델 설계

	0	1	2	3	4	5	6	7	8
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Layer(층)	Input	Hidden	Output
각 Layer의 노드 수	8	12, 8	1
활성화함수		relu, relu	sigmoid
손실함수	binary_crossentropy		
옵티마이저	adam		

인공신경망 설계 실습 #3

◆ 주어진 데이터

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

◆ 어떤 문제인가요?

- 종속변수가 연속된 실수 값임, 회귀문제

인공신경망 설계 실습 #3

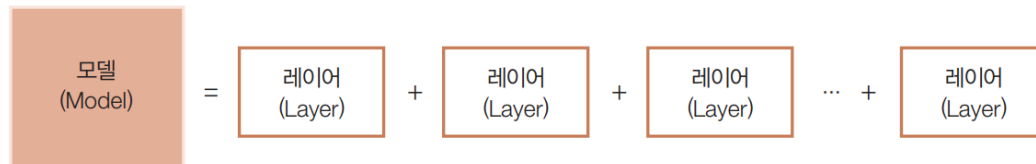
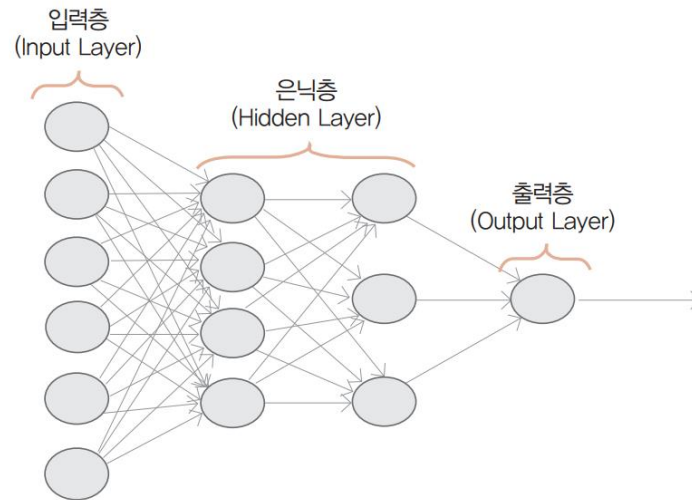
◆ 딥러닝 모델 설계

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

Layer(층)	Input	Hidden	Output
각 Layer의 노드 수	13	30, 6	1
활성화함수		relu, relu	relu
손실함수	mean_squared_error		
옵티마이저	adam		

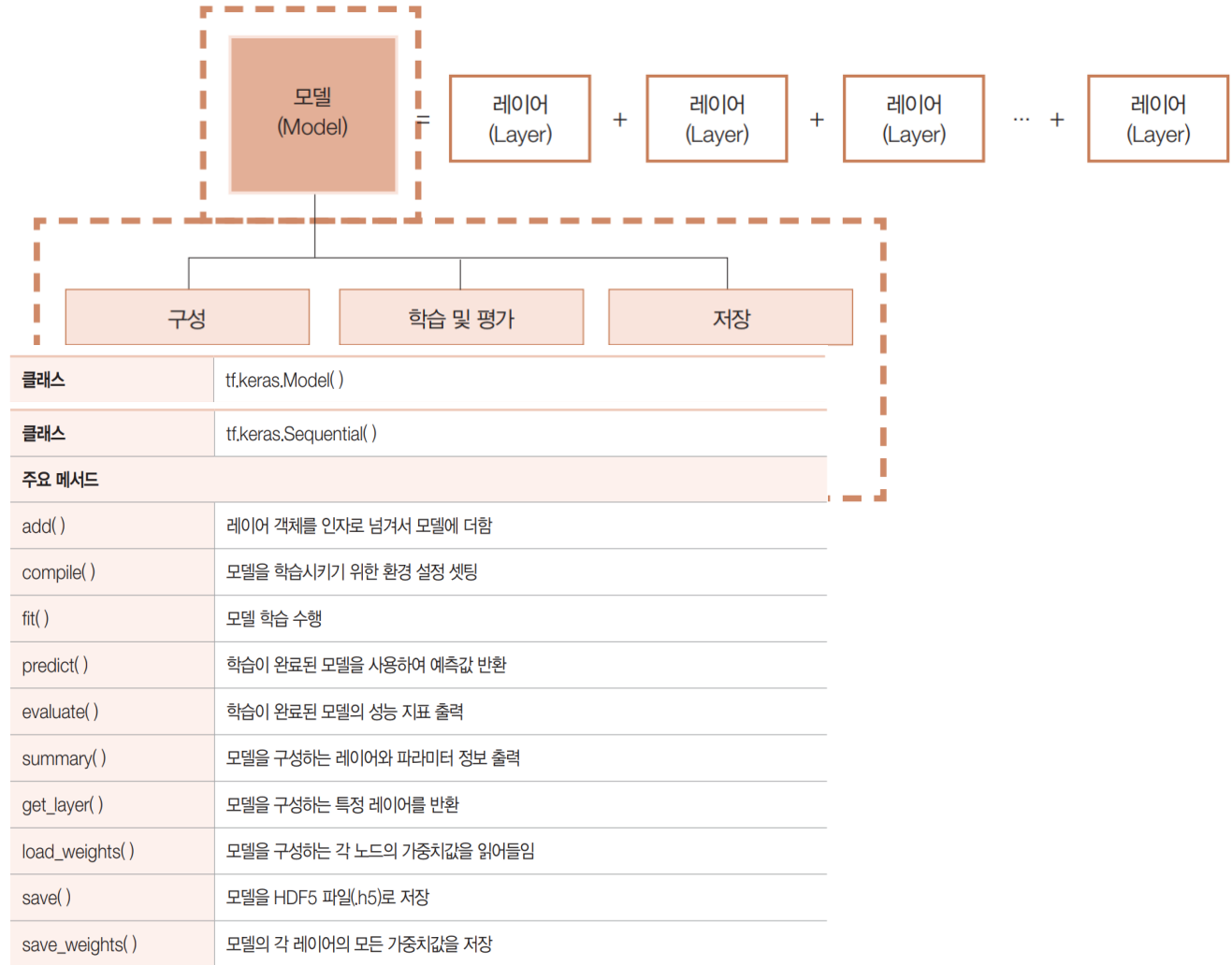
인공신경망의 주요 함수

◆ 모델의 구성



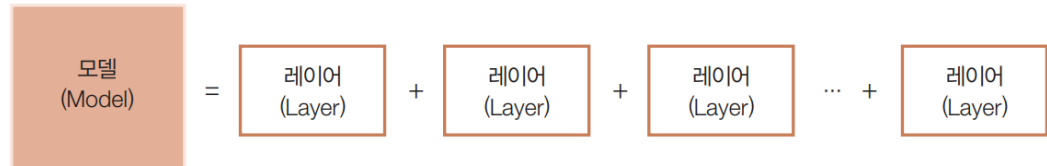
인공신경망의 주요 함수

◆ 모델의 구성



인공신경망의 주요 함수

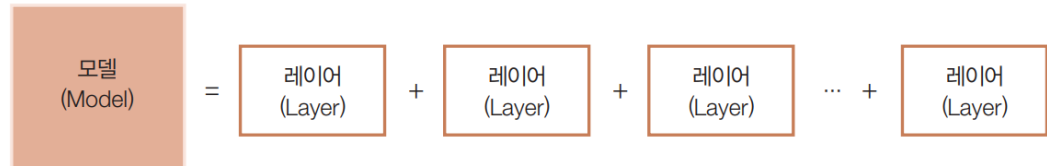
◆ 모델의 구성



클래스	<code>tf.keras.Model()</code>
클래스	<code>tf.keras.Sequential()</code>
주요 메서드	
<code>add()</code>	레이어 객체를 인자로 넘겨서 모델에 더함
<code>compile()</code>	모델을 학습시키기 위한 환경 설정 셋팅
<code>fit()</code>	모델 학습 수행
<code>predict()</code>	학습이 완료된 모델을 사용하여 예측값 반환
<code>evaluate()</code>	학습이 완료된 모델의 성능 지표 출력
<code>summary()</code>	모델을 구성하는 레이어와 파라미터 정보 출력
<code>get_layer()</code>	모델을 구성하는 특정 레이어를 반환
<code>load_weights()</code>	모델을 구성하는 각 노드의 가중치값을 읽어들이м
<code>save()</code>	모델을 HDF5 파일(h5)로 저장
<code>save_weights()</code>	모델의 각 레이어의 모든 가중치값을 저장

인공신경망의 주요 함수

◆ 모델의 환경설정

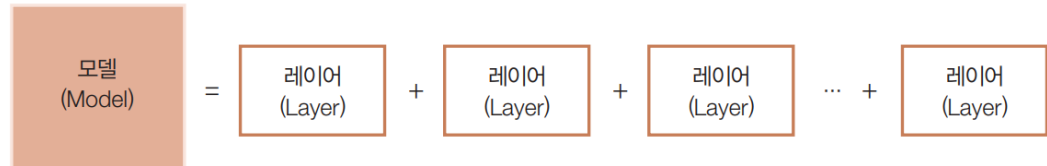


메서드	model.compile()
주요 인자	
loss	손실 함수
optimizer	옵티마이저(기본값 = 'rmsprop')
metrics	모델 성능 지표

```
In [ ]: model.compile(loss = 'sparse_categorical_crossentropy',
                    optimizer = 'adam',
                    metrics = ['accuracy'])
```

인공신경망의 주요 함수

◆ 모델의 학습

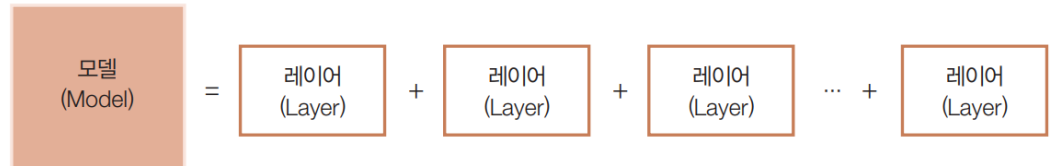


클래스	model.fit()
주요 인자	
x	학습 데이터의 독립변수, 피처
y	학습 데이터의 종속변수, 레이블
epochs	주어진 학습 데이터를 몇 번 반복하여 학습할지 횟수 지정
batch_size	gradient update를 진행할 샘플의 수
verbose	학습 진행 로그 출력 설정(0 : 출력하지 않음, 1 : 진행바로 표시, 2 : 에폭당 한 줄로 표시, 기본값 = 1)
callbacks	콜백 인스턴스(tf.keras.callbacks) 설정
validation_split	학습 데이터에서 검증용으로 사용할 데이터의 비율
validation_data	학습 중 검증용으로 사용할 데이터 설정
shuffle	각 에폭에서 데이터를 섞을지 여부 설정

```
In [ ]: model.fit(X_train, y_train, epochs = 50, batch_size = 4)
```

인공신경망의 주요 함수

◆ 모델의 평가

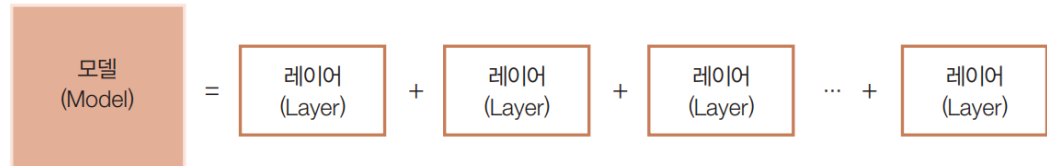


클래스	model.evaluate ()
주요 인자	
x	테스트 데이터의 독립변수, 피쳐
y	테스트 데이터의 종속변수, 레이블

```
In [ ]: model.evaluate(X_test, y_test)
```

인공신경망의 주요 함수

◆ 모델의 활용(예측)

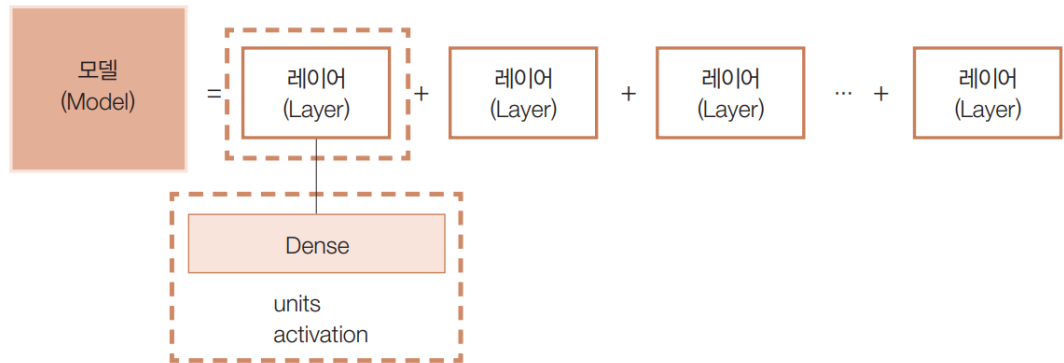


클래스	model.predict()
주요 인자	
x	학습시킨 데이터와 같은 형태로 변환한 독립변수

```
In [ ]: model.predict(X_test)
```

인공신경망의 주요 함수

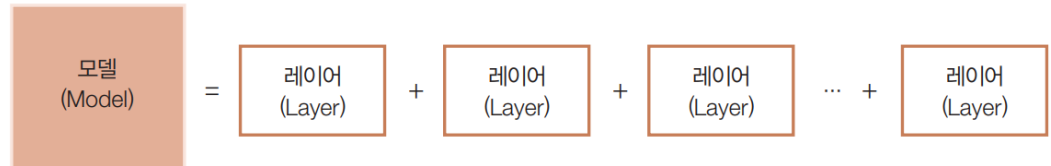
◆ 레이어



클래스	<code>tf.keras.layers.Layer()</code>
-----	---------------------------------------

인공신경망의 주요 함수

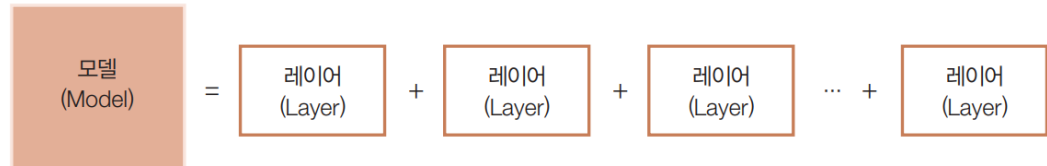
◆ 레이어



클래스	<code>tf.keras.layers.InputLayer()</code>
주요 인자	
<code>input_shape</code>	입력 데이터의 구조
<code>name</code>	레이어의 이름(문자열)

인공신경망의 주요 함수

◆ 레이어



클래스	tf.keras.layers.Dense()
주요 인자	
units	레이어를 구성하는 노드의 수
activation	활성화 함수 설정
kernel_regularizer	가중치 행렬에 적용된 정규화 함수 설정
bias_regularizer	편향 벡터에 적용된 정규화 함수 설정
activity_regularizer	레이어의 출력에 적용된 정규화 함수 설정
kernel_constraint	가중치 행렬에 적용되는 제약 조건 함수 설정
bias_constraint	편향 벡터에 적용되는 제약 조건 함수 설정

```
In [ ]: model = Sequential()
model.add(Dense(6, input_dim = 4, activation = 'relu'))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(12, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))
```

인공신경망 구현

◆ 붓꽃 데이터 분류를 위한 인공 신경망

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 데이터 불러오기
iris = load_iris()
df = pd.DataFrame(data = iris.data, columns = iris.feature_names)
df['label'] = iris.target

# 데이터 분할
y = df['label']
X = df.drop(['label'], axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42,
                                                    stratify = y)

# 모델의 설정
model = Sequential()
model.add(Dense(8, input_dim = 4, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))

# 모델 컴파일
model.compile(loss = 'sparse_categorical_crossentropy',
              optimizer = 'adam',
              metrics = ['accuracy'])

# 모델 실행
model.fit(X_train, y_train, epochs = 50, batch_size = 4)

# 모델 평가
model.evaluate(X_test, y_test)

# 값 예측
model.predict(X_test)[0]
```


인공신경망 구현

◆ 데이터 준비

In []: `X_train.shape`

Out []: (112, 4)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 데이터 불러오기
iris = load_iris()
df = pd.DataFrame(data = iris.data, columns = iris.feature_names)
df['label'] = iris.target

# 데이터 분할
y = df['label']
X = df.drop(['label'], axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42,
stratify = y)

# 모델의 설정
model = Sequential()
model.add(Dense(8, input_dim = 4, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))

# 모델 컴파일
model.compile(loss = 'sparse_categorical_crossentropy',
              optimizer = 'adam',
              metrics = ['accuracy'])

# 모델 실행
model.fit(X_train, y_train, epochs = 50, batch_size = 4)

# 모델 평가
model.evaluate(X_test, y_test)

# 값 예측
model.predict(X_test)[0]
```

인공신경망 구현

◆ 인공신경망 구성

Layer(층)	Input	Hidden	Output
각 Layer의 노드 수	4	8	3
활성화함수		relu	softmax
손실함수	sparse_categorical_crossentropy		
옵티마이저	adam		

In []: `model.summary()`

Out []: Model: "sequential"

```

-----
Layer (type)                 Output Shape              Param #
-----
dense (Dense)                 (None, 8)                 40
-----
dense_1 (Dense)               (None, 3)                 27
-----
Total params: 67
Trainable params: 67
Non-trainable params: 0
-----

```

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 데이터 불러오기
iris = load_iris()
df = pd.DataFrame(data = iris.data, columns = iris.feature_names)
df['label'] = iris.target

# 데이터 분할
y = df['label']
X = df.drop(['label'], axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42,
stratify = y)

# 모델의 설정
model = Sequential()
model.add(Dense(8, input_dim = 4, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))

# 모델 컴파일
model.compile(loss = 'sparse_categorical_crossentropy',
              optimizer = 'adam',
              metrics = ['accuracy'])

# 모델 실행
model.fit(X_train, y_train, epochs = 50, batch_size = 4)

# 모델 평가
model.evaluate(X_test, y_test)

# 값 예측
model.predict(X_test)[0]

```

인공신경망 구현

◆ 인공신경망 학습

```
In [ ]: # 모델 실행
        model.fit(X_train, y_train, epochs = 50, batch_size = 4)
```

```
Out [ ]: Epoch 1/50
28/28 [=====] - 0s 1ms/step - loss: 1.1521 - accuracy: 0.1429
Epoch 2/50
28/28 [=====] - 0s 1ms/step - loss: 1.0838 - accuracy: 0.1964
Epoch 3/50
28/28 [=====] - 0s 2ms/step - loss: 1.0287 - accuracy: 0.1875
Epoch 4/50
28/28 [=====] - 0s 1ms/step - loss: 0.9815 - accuracy: 0.3929

... 중략 ...
Epoch 45/50
28/28 [=====] - 0s 1ms/step - loss: 0.3479 - accuracy: 0.9554
Epoch 46/50
28/28 [=====] - 0s 1ms/step - loss: 0.3472 - accuracy: 0.9375
Epoch 47/50
28/28 [=====] - 0s 1ms/step - loss: 0.3414 - accuracy: 0.9643
Epoch 48/50
28/28 [=====] - 0s 2ms/step - loss: 0.3364 - accuracy: 0.9643
Epoch 49/50
28/28 [=====] - 0s 2ms/step - loss: 0.3288 - accuracy: 0.9554
Epoch 50/50
28/28 [=====] - 0s 1ms/step - loss: 0.3250 - accuracy: 0.9643
<keras.callbacks.History at 0x7f08831e11d0>
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 데이터 불러오기
iris = load_iris()
df = pd.DataFrame(data = iris.data, columns = iris.feature_names)
df['label'] = iris.target

# 데이터 분할
y = df['label']
X = df.drop(['label'], axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42,
                                                    stratify = y)

# 모델의 설정
model = Sequential()
model.add(Dense(8, input_dim = 4, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))

# 모델 컴파일
model.compile(loss = 'sparse_categorical_crossentropy',
              optimizer = 'adam',
              metrics = ['accuracy'])

# 모델 실행
model.fit(X_train, y_train, epochs = 50, batch_size = 4)

# 모델 평가
model.evaluate(X_test, y_test)

# 값 예측
model.predict(X_test)[0]
```

인공신경망 구현

◆ 인공신경망 평가 및 예측

```
In [ ]: model.evaluate(X_test, y_test)
```

```
Out [ ]: 2/2 [=====] - 0s 9ms/step - loss: 0.3608 - accuracy: 0.9474
[0.3608211874961853, 0.9473684430122375]
```

```
In [ ]: model.predict(X_test)[0]
```

```
Out [ ]: array([0.9544583 , 0.0440354 , 0.00150629], dtype = float32)
```

```
In [ ]: # 테스트 데이터로 예측한 값이 가장 큰 값을 반환(argmax)한 결과
np.argmax(model.predict(X_test)[0])
```

```
Out [ ]: 0
```

```
In [ ]: # 테스트셋의 실제값
y_test.iloc[0]
```

```
Out [ ]: 0
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 데이터 불러오기
iris = load_iris()
df = pd.DataFrame(data = iris.data, columns = iris.feature_names)
df['label'] = iris.target

# 데이터 분할
y = df['label']
X = df.drop(['label'], axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42,
stratify = y)

# 모델의 설정
model = Sequential()
model.add(Dense(8, input_dim = 4, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))

# 모델 컴파일
model.compile(loss = 'sparse_categorical_crossentropy',
              optimizer = 'adam',
              metrics = ['accuracy'])

# 모델 실행
model.fit(X_train, y_train, epochs = 50, batch_size = 4)

# 모델 평가
model.evaluate(X_test, y_test)

# 값 예측
model.predict(X_test)[0]
```

Summary

Summary

◆ Activation Function

- sigmoid, tanh, relu, softmax

◆ Loss Function

- categorical_crossentropy, binary_crossentropy, mean_squared_error

◆ Optimizer

- adam, adagrad, rmsprop, sgd

◆ Metric

- accuracy, mse, ce

◆ Epochs

◆ Batch size

```
# 모델의 설정
model = Sequential()
model.add(Dense(16, input_dim=4, activation='relu'))
model.add(Dense(3, activation='softmax'))

# 모델 컴파일
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# 모델 실행
model.fit(X_train, y_train, epochs=50, batch_size=1)
```

Summary - 딥러닝 모델링 절차

◆ 데이터셋 생성

- 원본 데이터를 불러오거나 시뮬레이션을 통해 데이터 생성하고, 데이터 정제(cleansing)진행
 - 딥러닝 모델의 학습 및 평가를 할 수 있도록 포맷 변환
- 데이터로부터 훈련셋(training set), 검증셋(validation set), 시험셋(test set)을 생성

◆ 모델 구성

- 시퀀스 모델 생성 후, 필요한 레이어 추가
 - layers, models 이용
 - layers : 각 계층 만드는 모듈, add()함수
 - models : 각 layer연결하여 신경망 모델 생성, 컴파일, 학습
 - model 함수 : compile(), fit(), predict(), evaluate()
- 학습을 위한 설정값 지정: 에폭(epochs), 배치사이즈(batch size)
- 손실 함수(loss function) 및 최적화 방법(optimizer) 정의
 - 교차 엔트로피(cross entropy), MSE 등
 - SGD, Adam, Adagrad, RMSprop 등
- compile() 이용

Summary - 딥러닝 모델링 절차

◆ 모델 학습 시키기

- 훈련셋(training set)을 이용, 구성된 모델로 학습
- fit() 이용
 - `model.fit(x_train, y_train, validation_data=(x_test, y_test), batch_size=10, epochs=20)`

◆ 학습과정 살피기

- 모델 학습 시 훈련셋(training set), 검증셋(validation set)의 손실 및 정확도 측정
- 반복 횟수에 따른 손실 및 정확도 추이를 보면서 학습 상황 판단

◆ 모델 평가

- 준비된 시험셋(test set) 이용, 학습한 모델 평가.
- evaluate() 이용

◆ 모델 사용(예측)

- 임의의 입력 모델의 출력값 도출
- predict() 이용 : `pred = model.predict(x_test)`

[02강] 학습 내용 확인하기

[02강] 학습 내용 확인하기

문제1. 8개의 독립변수와 1개의 종속변수를 갖는 데이터를 학습하기 위한 인공신경망을 구성할 때, 입력 노드의 수는 ?

- ① 7
- ② 8
- ③ 9
- ④ 10

문제2. 다음 중 모델의 학습 실행 시 호출하는 메서드는?

- ① compile()
- ② fit()
- ③ predict()
- ④ evaluate()

문제3. 주어진 학습 데이터를 몇 번 반복하여 학습할지 횟수를 지정하는 하이퍼파라미터는?

- ① epochs
- ② batch_size
- ③ x
- ④ y

[02강] 학습 내용 확인하기(정답)

문제1. 8개의 독립변수와 1개의 종속변수를 갖는 데이터를 학습하기 위한 인공신경망을 구성할 때, 입력 노드의 수는 ?

- ① 7
- ② 8
- ③ 9
- ④ 10

문제2. 다음 중 모델의 학습 실행 시 호출하는 메서드는?

- ① compile()
- ② fit()
- ③ predict()
- ④ evaluate()

문제3. 주어진 학습 데이터를 몇 번 반복하여 학습할지 횟수를 지정하는 하이퍼파라미터는?

- ① epochs
- ② batch_size
- ③ x
- ④ y

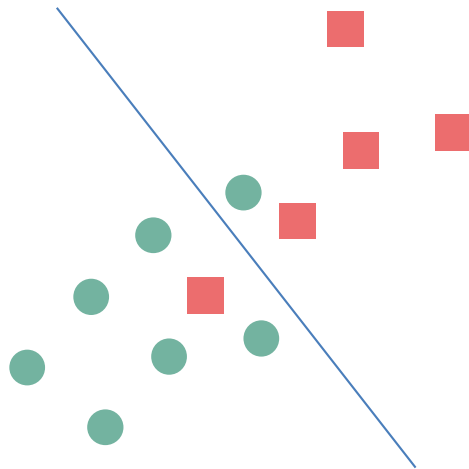
인공신경망 성능개선

3.1 인공신경망 성능개선

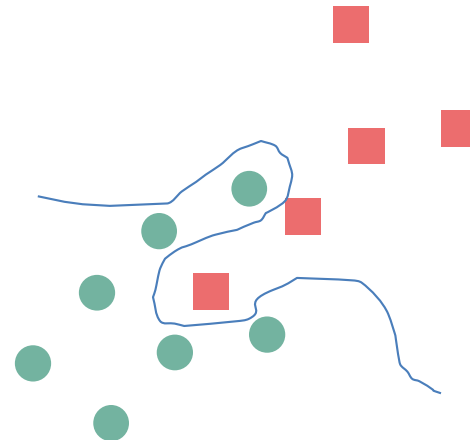
3.2 데이터 정규화를 통한 성능 개선

과적합

◆ 어떤 모델이 더 좋은 모델인가?



1번



2번

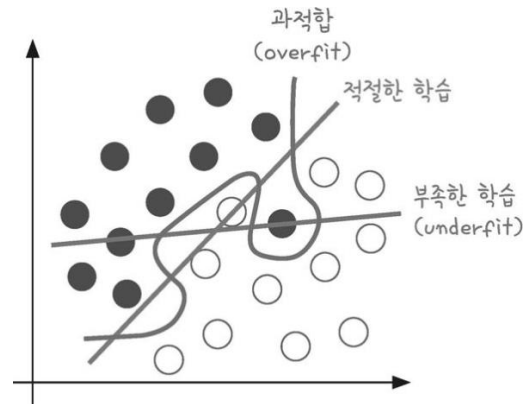
과적합

◆ 과적합(Overfitting)

- 학습데이터에 성능이 좋지만 실제 데이터에 관해 성능이 떨어지는 현상
 - 지도학습을 통해 만들어진 모델이 학습 데이터(Training data) 내에서는 분류가 잘 되나, 새로운 데이터(Unseen Data or Test Data)에서는 분류성능이 떨어지는 상황

◆ 부적합(Underfitting)

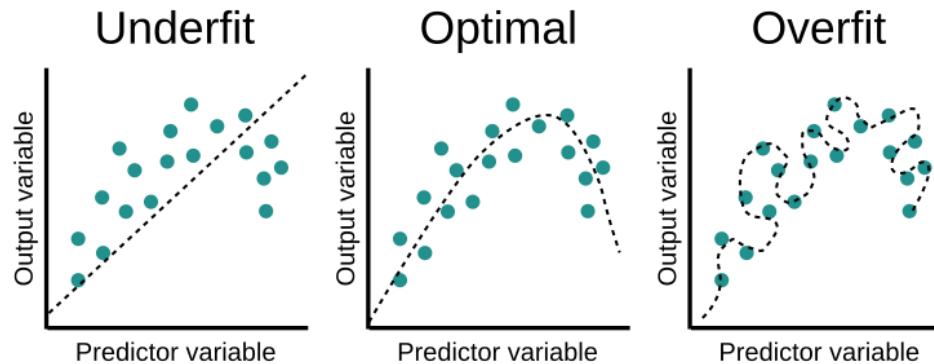
- 적정 수준의 학습을 하지 못해 실제 성능이 떨어지는 현상



일반화된 모델

◆ 은닉층의 수와 노드 수를 증가시킬수록 분류가 정확해진다!

- 이것의 문제점?
 - 학습데이터만 분류를 잘하는 분류기가 탄생함



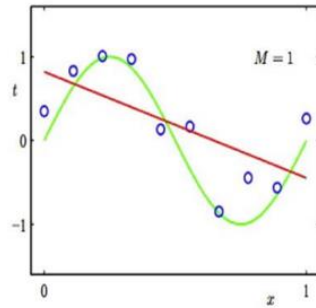
◆ 딥러닝은 일반적인 데이터에 대해서도 잘 동작하는 모델을 만드는 것이 목표!

- 일반화(Generalization)
- 일반화된 모델이란?

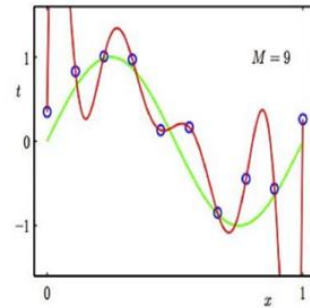
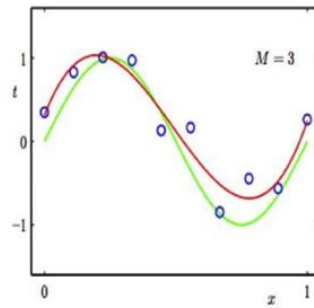
일반화된 모델

◆ Underfitting, Generalization, Overfitting

Regression:

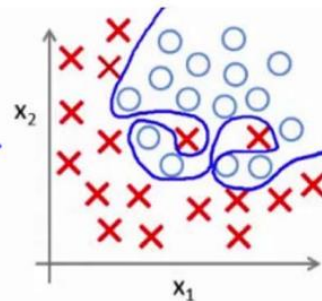
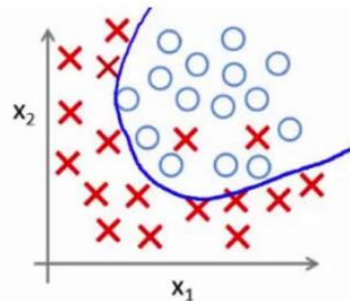
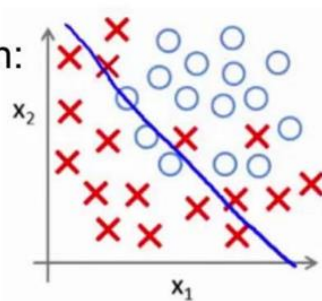


predictor too inflexible:
cannot capture pattern



predictor too flexible:
fits noise in the data

Classification:



Copyright © 2014 Victor Lavrenko

일반화된 모델

◆ Underfitting, Overfitting이 아닌 Generalization 모델을 만드는 방법

- 1) 학습 데이터를 충분하게 늘린다
 - Data Augmentation
- 2) 과적합이 발생하기 전에 학습을 종료한다
 - Early stopping
- 3) 모델의 복잡도를 낮춘다
 - L2, L1
 - Dropout

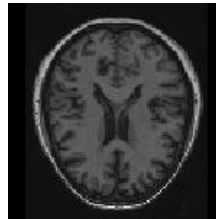
일반화된 모델

◆ Underfitting, Overfitting이 아닌 Generalization 모델을 만드는 방법

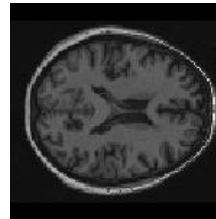
- 1) 학습 데이터를 충분하게 늘린다
 - Data Augmentation
- 2) 과적합이 발생하기 전에 학습을 종료한다
 - Early stopping
- 3) 모델의 복잡도를 낮춘다
 - L2, L1, Dropout



원본이미지



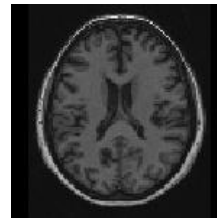
180도 회전



90도 회전



상하 반전

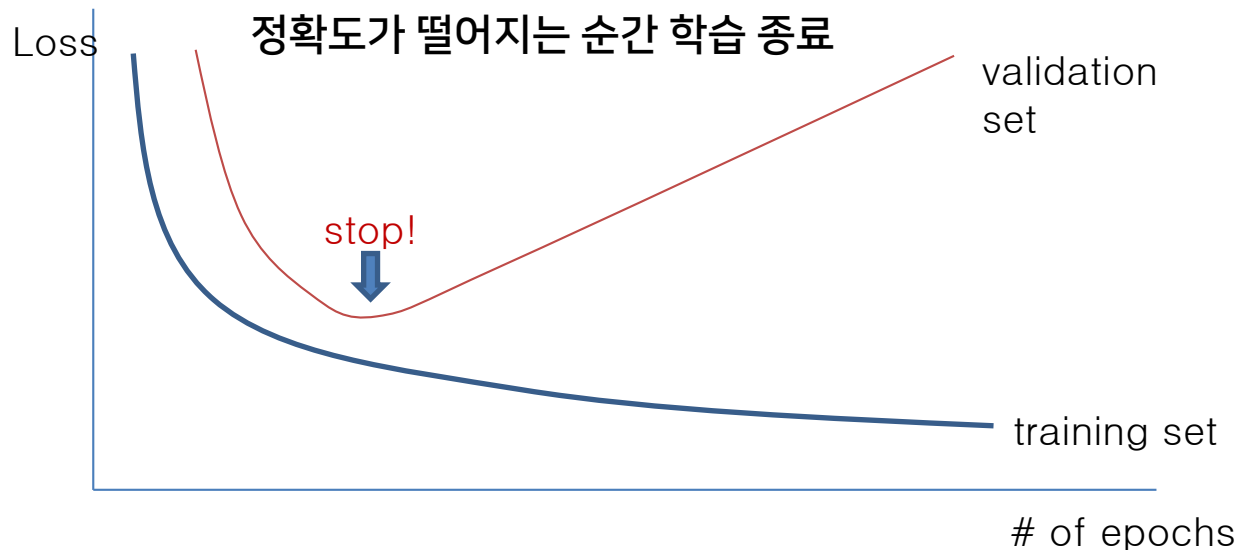


좌우 반전

일반화된 모델

◆ Underfitting, Overfitting이 아닌 Generalization 모델을 만드는 방법

- 1) 학습 데이터를 충분하게 늘린다
 - Data Augmentation
- 2) 과적합이 발생하기 전에 학습을 종료한다
 - Early stopping
- 3) 모델의 복잡도를 낮춘다
 - L2, L1, Dropout



일반화된 모델

◆ Underfitting, Overfitting이 아닌 Generalization 모델을 만드는 방법

- 1) 학습 데이터를 충분하게 늘린다
 - Data Augmentation
- 2) 과적합이 발생하기 전에 학습을 종료한다
 - Early stopping
- 3) **모델의 복잡도를 낮춘다**
 - **L2, L1, Dropout**

정규화= 모델의 복잡도를 낮춘다

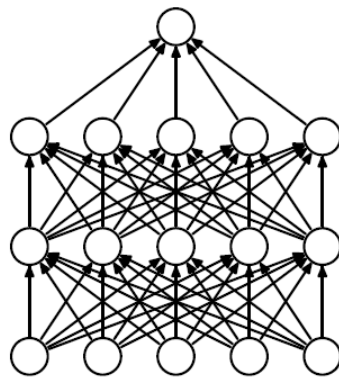
손실함수에 페널티(penalty)를
부여한다(Regularization)

학습 중에 랜덤하게 노드를
꺼뜨린다
= 학습을 진행하지 않는다
= Dropout

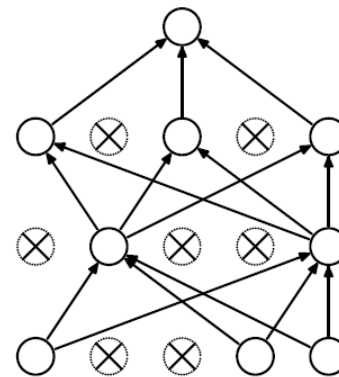
일반화된 모델

◆ Underfitting, Overfitting이 아닌 Generalization 모델을 만드는 방법

- 드롭아웃(Drop out)
 - 딥러닝 학습 중에 은닉층의 뉴런을 무작위로 삭제하여 과적합을 예방하는 기법
- dropout(p)
 - p라는 확률로 출력 노드의 신호를 보내다 말다 함
 - 드롭아웃을 적용한 다음에 오는 계층은 앞 계층의 일부 노드들의 신호가 p라는 확률로 단절되기 때문에 훨씬 더 견고하게 신호에 적응



(a) Standard Neural Net



(b) After applying dropout.

모델 성능 개선

◆ 과적합 해결방법

- 가중치 감소(Weight decay)
 - 딥러닝 학습 과정에서 큰 가중치에 대해서 큰 페널티를 부과하여 과적합을 예방하는 기법
- 정규화(Normalization)
 - 데이터 값의 범위를 변환(rescaling)시켜 학습 결과가 특정 데이터 분포에 과적합될 가능성을 낮추는 방법

모델 성능 개선

◆ 정규화(Normalization)

■ bank.csv

```
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0	unknown	yes
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0	unknown	yes

```
df.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407	0.832557
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282	2.292007
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000	0.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000	0.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000	0.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000	1.000000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000	58.000000

모델 성능 개선

◆ 정규화(Normalization)

■ 표준화(standardization)

- 입력 데이터에서 평균을 빼고 표준편차로 나눠주는 과정을 거쳐 데이터 분포를 표준정규분포 형태로 변경
- 평균은 0, 분산은 1
- 입력값이 일정한 분포로 들어오기 때문에 학습에 유리

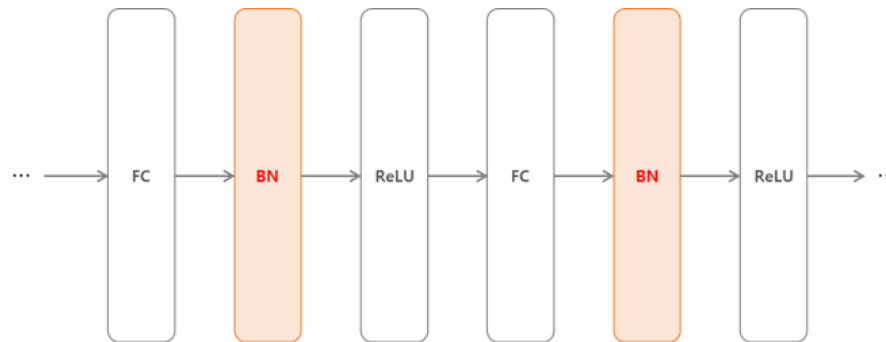
■ 최소극대화(minmax)

- 입력 데이터를 0에서 1사이로 압축하거나 늘리는 방법
- 데이터에서 최소값을 빼준 값을 데이터의 최대값과 최소값의 차이로 나눠주어 변형
- 0~1 사이 밖에 있는 값: 0과 1사이로 압축
- 전체 범위가 1이 안되던 값: 0과 1사이로 늘어남
- 평균적 범위를 넘어서는 너무 작거나 너무 큰 이상치가 있는 경우: 오히려 학습에 방해가 됨

모델 성능 개선

◆ 배치 정규화(Batch Normalization)

- 한 번에 입력으로 들어오는 배치 단위로 데이터 분포의 평균이 0, 분산이 1이 되도록 정규화 진행
- 정규화된 데이터를 스케일 및 시프트(scale & shift)하여 다음 레이어에 일정한 범위의 값들만 전달
- 은닉층 단위마다 배치 정규화를 넣어줌
- 효과: 학습 전체 과정의 안정화
 - 빠른 학습 가능: learning rate를 높게 잡을 수 있음
 - 자체적인 regularization 효과
 - drop out 제거 가능: drop out을 사용하면 학습 속도가 느려짐



[BN을 사용한 신경망 예]

인공신경망 성능개선

◆ Boston dataset

```
In [ ]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import load_boston

# 데이터 불러오기
boston = load_boston()
boston_df = pd.DataFrame(data = boston.data, columns = boston.feature_names)
boston_df['PRICE'] = boston.target
boston_df.head()
```

```
Out [ ]:
```

	CRIM	ZN	INDUS	...	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	...	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	...	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	...	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	...	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	...	18.7	396.90	5.33	36.2

인공신경망 성능개선

◆ 데이터 분할 및 모델 생성

```
In [ ]: from sklearn.model_selection import train_test_split
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense

        # 데이터 분할
        y = boston_df['PRICE']
        X = boston_df.drop(['PRICE'], axis = 1)

        X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42)

        # 모델의 설정
        model = Sequential()
        model.add(Dense(16, input_dim = 13, activation = 'relu'))
        model.add(Dense(1, activation = 'relu'))
```

인공신경망 성능개선

◆ 모델 컴파일

```
In [ ]: # 모델 컴파일
        model.compile(loss = 'mean_squared_error',
                      optimizer = 'adam',
                      metrics = ['mse'])

        model.summary()
```

Out []: Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
dense (Dense)	(None, 16)	224

dense_1 (Dense)	(None, 1)	17
=====		
Total params: 241		
Trainable params: 241		
Non-trainable params: 0		

인공신경망 성능개선

◆ 모델 학습

In []: # 모델 실행

```
model.fit(X_train, y_train, epochs = 50, batch_size = 10)
```

Out []: Epoch 1/50

38/38 [=====] - 0s 620us/step - loss: 749.4234 - mse: 749.4234

Epoch 2/50

38/38 [=====] - 0s 728us/step - loss: 747.5175 - mse: 747.5174

Epoch 3/50

38/38 [=====] - 0s 674us/step - loss: 667.5957 - mse: 667.5957

Epoch 4/50

38/38 [=====] - 0s 728us/step - loss: 640.8848 - mse: 640.8848

Epoch 5/50

38/38 [=====] - 0s 809us/step - loss: 575.4298 - mse: 575.4298

... 중략 ...

Epoch 47/50

38/38 [=====] - 0s 701us/step - loss: 52.0202 - mse: 52.0202

Epoch 48/50

38/38 [=====] - 0s 809us/step - loss: 45.9028 - mse: 45.9028

Epoch 49/50

38/38 [=====] - 0s 809us/step - loss: 47.5085 - mse: 47.5085

Epoch 50/50

38/38 [=====] - 0s 809us/step - loss: 47.4613 - mse: 47.4613

<tensorflow.python.keras.callbacks.History at 0x1bfa11172e0>

인공신경망 성능개선

◆ 예측값 확인

- 하나의 예측값 비교

```
In [ ]: y_pred = model.predict(X_test)
        y_pred[0]
```

```
Out [ ]: array([29.119976], dtype = float32)
```

```
In [ ]: y_test[0]
```

```
Out [ ]: 24.0
```

인공신경망 성능개선

◆ 예측값 확인

■ 여러 개의 예측값 비교

```
In [ ]: y_pred = np.reshape(y_pred,(127,))
```

```
Out [ ]: result = pd.DataFrame({'y': y_test.values,
                                'y_pred': y_pred,
                                'diff': np.abs(y_test.values - y_pred)})
result.sort_values(by = ['diff'], ascending = False)
```

```
In [ ]: result = pd.DataFrame({'y': y_test.values, 'y_pred': y_pred, 'diff':
                                np.abs(y_test.values - y_pred)})
result.sort_values(by = ['diff'], ascending = False)
```

```
Out [ ]:
```

	y	y_pred	diff
96	50.0	21.840014	28.159986
111	27.5	4.602501	22.897499
18	50.0	37.318615	12.681385
35	19.4	7.782295	11.617705
42	50.0	38.431679	11.568321
...
102	12.0	11.762398	0.237602
59	18.5	18.596060	0.096060
122	8.8	8.705296	0.094704
28	21.7	21.743273	0.043273
61	35.4	35.384048	0.015952

127 rows × 3 columns

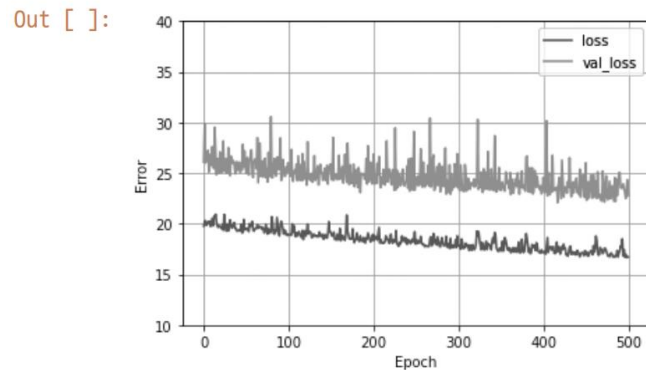
인공신경망 성능개선

◆ 검증용 데이터 셋 설정

```
In [ ]: history = model.fit(X_train, y_train, epochs = 500, verbose = 0, validation_
split = 0.2)
```

```
In [ ]: def plot_loss(history):
    plt.plot(history.history['loss'], label = 'loss')
    plt.plot(history.history['val_loss'], label = 'val_loss')
    plt.ylim([0, 40])
    plt.xlabel('Epoch')
    plt.ylabel('Error')
    plt.legend()
    plt.grid(True)
```

```
In [ ]: plot_loss(history)
```



인공신경망 성능개선

◆ 데이터 정규화

■ 모델 생성

```
In [ ]: from tensorflow.keras.layers.experimental import preprocessing

normalizer = preprocessing.Normalization(axis = -1)
normalizer.adapt(np.array(X_train))

normalized_model = Sequential()
normalized_model.add(normalizer)
normalized_model.add(Dense(16, activation = 'relu'))
normalized_model.add(Dense(1, activation = 'relu'))

normalized_model.summary()
```

Out []: Model: "sequential_1"

Layer (type)	Output Shape	Param #
normalization (Normalization (None, 13))		27
dense_2 (Dense)	(None, 16)	224
dense_3 (Dense)	(None, 1)	17
Total params: 268		
Trainable params: 241		
Non-trainable params: 27		

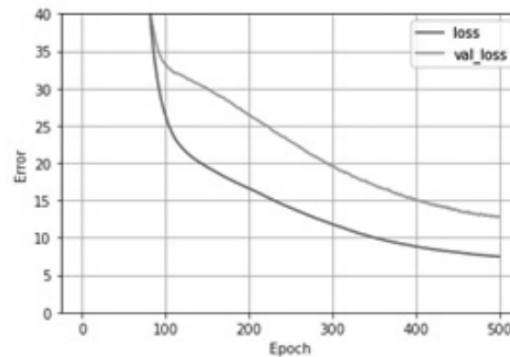
인공신경망 성능개선

◆ 데이터 정규화

■ 모델 컴파일 및 학습

```
In [ ]: normalized_model.compile(loss = 'mean_squared_error',  
                                optimizer = 'adam',  
                                metrics = ['mse'])  
  
normalized_history = normalized_model.fit(X_train, y_train, epochs = 500,  
                                         verbose = 0, validation_split = 0.2)  
  
plot_loss(normalized_history)
```

Out []:



인공신경망 성능개선

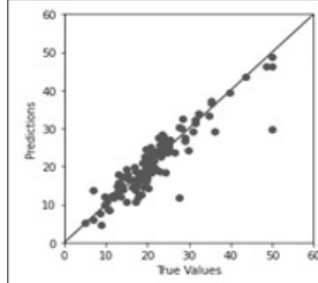
◆ 데이터 정규화

- 예측값과 실제값 시각화

```
In [ ]: y_pred = normalized_model.predict(X_test).flatten()
```

```
a = plt.axes(aspect = 'equal')
plt.scatter(y_test, y_pred)
plt.xlabel('True Values')
plt.ylabel('Predictions')
lims = [0, 60]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
```

```
Out [ ]:
```



인공신경망 성능개선

◆ 데이터 정규화

■ 예측값과 실제값 시각화

```
In [ ]: result = pd.DataFrame({'y': y_test.values, 'y_pred': y_pred, 'diff': y_test.
values - y_pred, 'diff(abs)': np.abs(y_test.values - y_pred)})
result.sort_values(by = ['diff(abs)'], ascending = False)
```

```
Out [ ]:
```

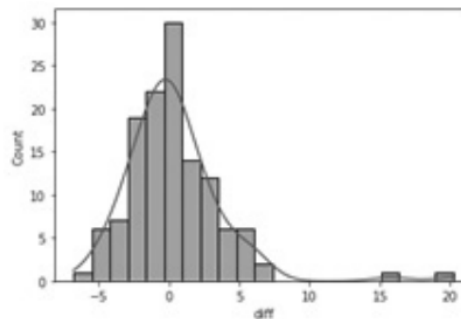
	y	y_pred	diff	diff(abs)
	96	50.0	29.784805	20.215195
	111	27.5	11.743855	15.756145
	113	36.2	29.294373	6.905627
	79	7.0	13.791144	-6.791144
	99	17.2	10.795241	6.404759

	54	43.5	43.560184	...
	101	21.4	21.384632	...
	5	20.0	19.992371	...
	121	24.4	24.394968	...
	62	31.5	31.495457	...

127 rows x 4 columns

```
In [ ]: sns.histplot(data = result['diff'], kde = True)
```

```
Out [ ]: <AxesSubplot:xlabel = 'diff', ylabel = 'Count'>
```



Summary

Summary

◆ Underfitting, Overfitting이 아닌 Generalization 모델을 만드는 방법

- 1) 학습 데이터를 충분하게 늘린다
 - Data Augmentation
- 2) 과적합이 발생하기 전에 학습을 종료한다
 - Early stopping
- 3) 모델의 복잡도를 낮춘다
 - L2, L1
 - Dropout

[03강] 학습 내용 확인하기

[03강] 학습 내용 확인하기

문제1. 학습데이터에 성능이 좋지만 실제 데이터에 관해 성능이 떨어지는 현상은?

- ① 적합
- ② 부적합
- ③ 과적합
- ④ 일반화

문제2. 이른 종료를 위해 학습 과정을 성능을 모니터링하는데 사용하는 데이터 셋은?

- ① 학습용 데이터 셋
- ② 검증용 데이터 셋
- ③ 테스트용 데이터 셋
- ④ 모두 사용

문제3. 데이터 값의 범위를 변환시켜 학습 결과가 특정 데이터 분포에 과적합 될 가능성을 낮추는 방법은?

- ① 정규화
- ② 이른종료
- ③ 드롭아웃
- ④ 데이터 증강

[03강] 학습 내용 확인하기

문제1. 학습데이터에 성능이 좋지만 실제 데이터에 관해 성능이 떨어지는 현상은?

- ① 적합
- ② 부적합
- ③ 과적합
- ④ 일반화

문제2. 이른 종료를 위해 학습 과정을 성능을 모니터링하는데 사용하는 데이터 셋은?

- ① 학습용 데이터 셋
- ② 검증용 데이터 셋
- ③ 테스트용 데이터 셋
- ④ 모두 사용

문제3. 데이터 값의 범위를 변환시켜 학습 결과가 특정 데이터 분포에 과적합 될 가능성을 낮추는 방법은?

- ① 정규화
- ② 이른종료
- ③ 드롭아웃
- ④ 데이터 증강

Thanks
