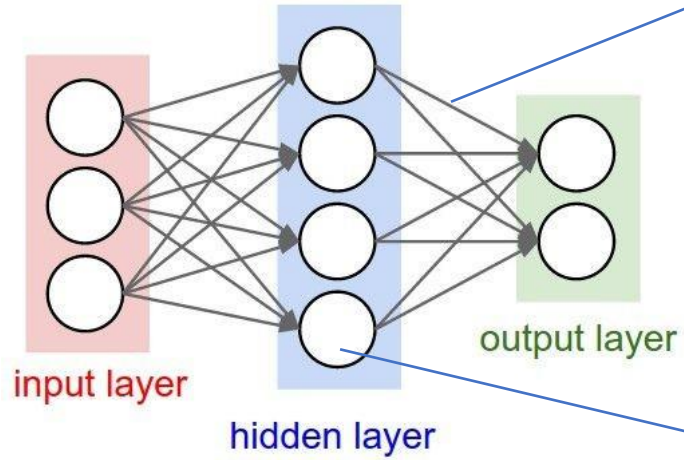
데이터가 뛰어노는 AI 놀이터, 캐글

4.4 신경망

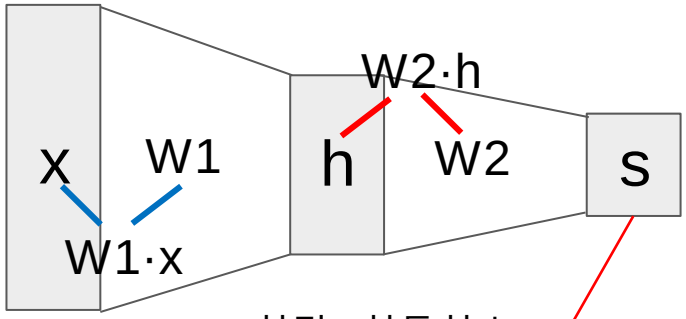
아이펠 강남 2기 김경현

4.1 신경망 개요

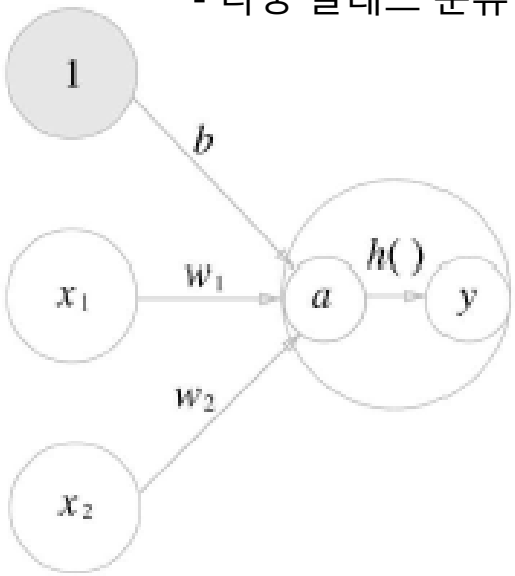
■ 신경망 구조



- 신경망(Neural Network)
- 다층 퍼셉트론(MLP)
- 완전 연결 계층(Fully-connected Layers)



- 회귀 : 항등함수
- 이진분류 : 시그모이드
- 다중 클래스 분류 : 소프트맥스



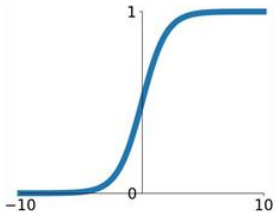
- 활성화 함수 처리 과정

4.1 신경망 개요

■ 활성화 함수(Activation Function)

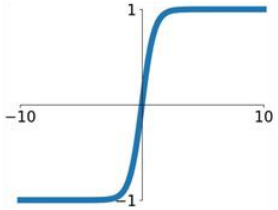
Sigmoid

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



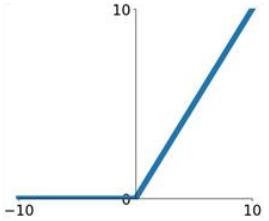
tanh

$$\tanh(x)$$



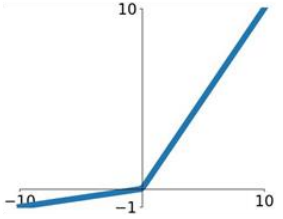
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

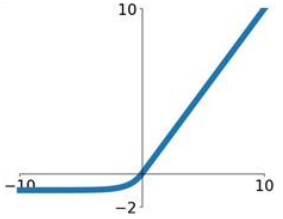


Maxout

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

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



- [추가 참고자료] CS231n – Training Neural Networks

■ 활성화 함수와 같은 비선형 변환이 필요한 이유?

- 아무리 깊게 층을 쌓더라도 그 변환이 선형이라면 하나의 층으로 그 변환을 표현할 수 있기 때문
- [추가 참고자료] 비전공자를 위한 딥러닝 – 비선형 변환이 필요한 이유

4.2 신경망의 특징 – 4.3 주요 라이브러리

■ 신경망의 특징

특징	내용
변수값은 수치로 표현	- 문자 형태로는 학습할 수 없음
결측값을 다룰 수 없음	- 신경망 연산 구조상 결측값을 다룰 수 없음
비선형성과 변수 간의 상호작용을 반영	- 신경망 구조로부터 어느 정도 비선형성과 변수 간 상호작용 반영
변숫값을 표준화 등으로 스케일링	- 변수값의 크기가 고르지 않으면 학습이 제대로 진행되지 않음
하이퍼파라미터에 따른 모델 성능의 편차	- 하이퍼파라미터 조정은 비교적 어려움(경험치) - 하이퍼파라미터에 따라 오버피팅 혹은 언더피팅 이슈 발생
다중 클래스 분류에 강점	- 구조상 다중 클래스 분류 모델링에 강점을 가짐
GPU로 고속화	- GPU는 신경망에서 필요로 하는 행렬 연산에 적합

■ 주요 라이브러리

텐서플로(케라스)	- difficult to debug : runtime 중 사용자가 interactive하게 개입하기 힘듦
파이토치	- runtime debugging : 코딩을 하면서도 사용자가 interactive하게 개입할 수 있음

4.4 신경망 구현

■ TensorFlow2 API로 모델 작성 및 학습하기: CIFAR-100 (1) Sequential API 활용

```
1  import tensorflow as tf
2  from tensorflow import keras
3
4  # 데이터 구성부분
5  cifar100 = keras.datasets.cifar100
6
7  (x_train, y_train), (x_test, y_test) = cifar100.load_data()
8  x_train, x_test = x_train / 255.0, x_test / 255.0
9  print(len(x_train), len(x_test))
10
11 # Sequential Model을 구성해주세요.
12 model = keras.Sequential([
13                 keras.layers.Conv2D(16, 3, activation='relu'),
14                 keras.layers.MaxPool2D((2,2)),
15                 keras.layers.Conv2D(32, 3, activation='relu'),
16                 keras.layers.MaxPool2D((2,2)),
17                 keras.layers.Flatten(),
18                 keras.layers.Dense(256, activation='relu'),
19                 keras.layers.Dense(10, activation='softmax')
20 ])
21
22 # 모델 학습 설정
23 model.compile(optimizer='adam',
24         loss='sparse_categorical_crossentropy',
25         metrics=['accuracy'])
26
27 model.fit(x_train, y_train, epochs=5)
28
29 model.evaluate(x_test,  y_test, verbose=2)
```

출처 : Fundamental 18번

4.5 케라스 사용 팁

■ 목적함수(Loss Function)

- 모델을 컴파일 할 때는 매개변수 loss에 목적함수를 설정하고 목적함수가 최소화하도록 학습을 진행

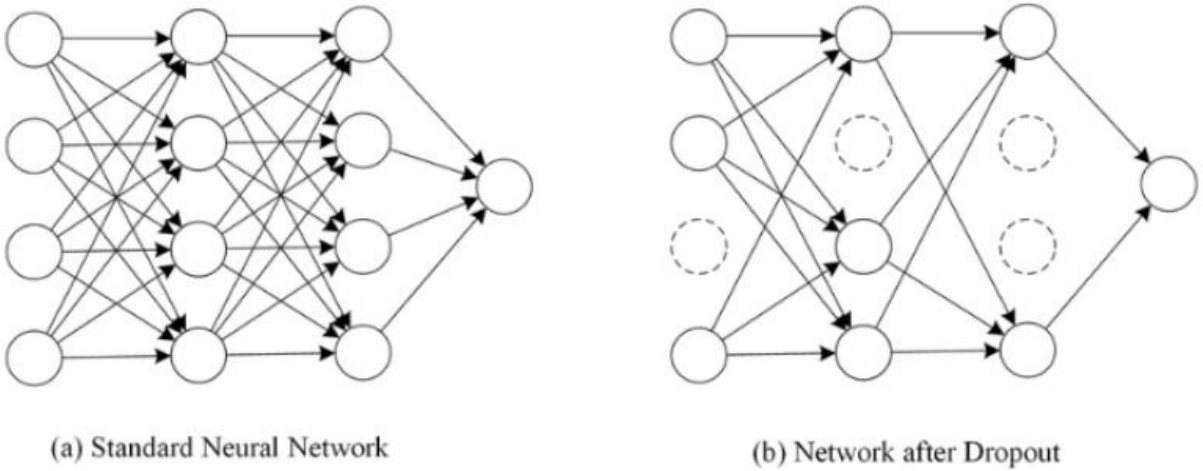
회귀	이진 분류	다중 클래스 분류
mean_squared_error	binary_crossentropy	categorical_crossentropy

■ 하이퍼파라미터

- 종류 : 옵티마이저 종류, 학습률(Learning-rate), 은닉층의 개수 및 노드 수, 드롭아웃 등

■ 드롭아웃(Dropout)

- 학습 과정에서 신경망의 일부를 사용하지 않는 방법
- 과적합(=Overfitting) 문제를 해결하는데 사용하는 방법 중 하나



(a) Standard Neural Network

(b) Network after Dropout

4.5 케라스 사용 팁 (2)

■ 학습 데이터와 검증 데이터의 점수 모니터링

- 각 에폭마다 학습 데이터와 검증 데이터의 점수를 출력할 수 있다.

■ 콜백(Callback)

- 개념 : 모델 학습 과정 중 미니배치 처리별 혹은 에폭별로 지정한 처리를 진행하는 것
- 종류 : **EarlyStopping**, 모델의 정기적인 저장, 학습률 스케줄링(조정)

```
from tensorflow.python.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping()
hist = model.fit(x_train,y_train,epochs=1000,batch_size=10,validation_data=
(x_val,y_val),callbacks=[early_stopping])
```

■ 임베딩 계층(Embedding Layer) – Fundamental

- 양의 정수(색인)를 고정된 크기의 밀집 벡터로 전환합니다.
- 예. [[4], [20]] -> [[0.25, 0.1], [0.6, -0.2]]
- 이 레이어는 모델의 첫 번째 레이어로만 사용할 수 있습니다.

■ 배치 정규화(Batch Normalization)

- 레이어의 출력 activation을 강제로 unit gaussian(0과 1 사이에 존재)으로 만들어 주는 아이디어
- learning rate를 높게 설정해 학습 속도 개선(gradient vanishing/exploding 문제 영향 X)
- gradient의 흐름을 원활하게 해주어 학습이 더 잘되게(robust) 해준다.
- regularization 효과가 있어 오버피팅을 억제(드롭아웃 필요성 감소)
- [추가 참고자료] CS231n – Training Neural Networks