



# 인공지능

## MLP란?

Multi Layer Perceptron의 약어로, 인간의 뉴런을 본따 만든 Perceptron을 기반으로 여러 개의 입력층/은닉층/출력층으로 구성된 인공신경망 모델을 의미한다.  
(Input/Hidden/Output Layer)

Input Layer → 입력과 가중치의 곱들의 합(Weighted sum)이 된다. (ex)  $y = w1 \times 1 + w2 \times 2 + w3 \times 3$ )

입력데이터의 특징에 해당한다.

Hidden Layer → Input 값들이 Sigmoid, ReLU, tanh 등 활성화 함수를 거쳐 출력층으로 나온다.

MLP의 성능을 높이는 중요한 부분이다. 여러 개의 Hidden Layer를 심층이라고 한다.

Output Layer → 은닉층의 최종 출력을 받아 출력한다. 출력을 Label이라고도 한다.

- Fully Connected Network(FCN, Dense Network)의 구조를 가진다.

입력 x에서 레이어들을 거친 출력 y가 원하는 목표값과의 차이(Error)를 통해 가중치와 편향을 조절한다.

---

## 손실함수 & 비용함수

Error를 통해 원하는 y값과 얼마나 차이나는지( $y_{true} - y_{pred}$ )를 알 수 있는데, 신경망에서 최적의 매개변수를 탐색하기 위해 사용되는 함수이다.

---

## 손실함수

에러를 줄여 원하는 값에 가깝게 하는 것이 모델의 주요 목표이다.

하나의 입력데이터에 대한 오차를 구하는 것이 손실함수이다.

$$E = \frac{1}{2}(y - T)^2$$

$$E = -t \log y$$

## 비용함수

손실함수와의 차이는 모든 입력 데이터셋에 대한 오차를 계산하는 것이다.

대표적으로 MSE(평균 제곱 오차), CEE(교차 엔트로피 오차)가 있다.

$$E = \frac{1}{n} \sum (y_k - T_k)^2$$

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

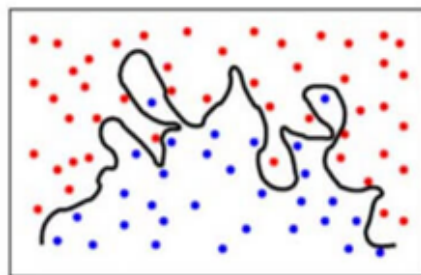
Mean Squared Error

Cross Entropy Error

과적합(방지 방법)

데이터는 학습 시키기 위한 데이터인 훈련 데이터(Train data), 검증 데이터(Validation data), 실제 성능을 확인하기 위한 시험 데이터(Test Data) 로 나뉘어 지는데, 이때 훈련 데이터에 지나치게 맞게 신경망이 학습하는 경우를 과적합(Overfitting)이라고 한다.

## Overfitting



파란색 데이터를 분류하는 과정

매개변수가 많고 표현력이 높은 모델이나 훈련 데이터가 적은 경우 발생한다.

과적합을 방지하기 위해서는 학습 parameter 개수를 줄이는 방법, 학습을 과적합이 발생하기 전에 중단하는 Early Stopping 기법, 학습 parameter의 값이 커지면 패널티를 추가하는 방법, 일부 은닉층의 뉴런을 꺼서 학습하고 테스트할 때는 모든 뉴런을 키는(삭제 비율을 출력에 곱하는) Dropout 기법 등이 있다.

## Keras 구성요소, 특징

Keras란 인공지능 모델을 위한 파이썬 라이브러리이다.

모델은 시퀀스나 독립실행형으로 구분되고 API는 순차형(Sequential) 또는 함수형(Functional) API 두가지로 나뉜다.

넘파이를 이용하여 직접 최적화 함수를 정의해 구현하는 것보다 단순하게

Sequential API: 선형 pile 계층으로 여러 개의 입출력을 만들어낼 수 없는 비교적 단순한 모델을 만들 때 사용한다.(하나의 입력, 하나의 출력)

위에서 아래로 순차적으로 진행한다.

Funtional API: 하나의 계층에 여러 계층을 연결할 수 있어 더 복잡한 모델을 다룰 수 있다.(다중입출력)

## 문법

- Sequential과 관련한 문법

```
from keras.models import Sequential
model = Sequential(layers=None, trainable=True, name=None)
```

- Sequential 모형 클래스 객체 생성

layer를 직접 삽입 또는 train 가능 여부, 이름을 설정할 수도 있다.

```
Sequential.add(layer, rebuild=True)
```

모델에 layer를 추가한다. (레이어 스택의 상단)

rebuild = false로 지정하면 다시 build 하지 않는다.

```
Sequential.pop(rebuild=True)
```

마지막 에서 추가한 특성을 제거한다.

rebuild = false로 지정하면 다시 build 하지 않는다.

```
from keras.layers import Dense
Dense(units, activation=None, ... , **kwargs)
```

- Fully Connected Layer를 구현한 클래스로, [(입력벡터와 가중치행렬의 내적) + (bias)] 이 활성화 함수 activation을 거쳐 출력값이 계산된다.

- units: 출력 뉴런의 수
- activation: 활성화 함수 지정(linear, relu, sigmoid, tanh, leakly\_relu, softmax 등)
- input\_shape: 입력층에서 사용할 때만 지정(필수 인자 아님)

ex) model.add(Dense(64, activation = "relu", input\_layer = (100,)))

입력이 1차원 데이터 100개, relu 활성화 함수를 거쳐 64개의 output으로 출력값을 넘겨준다.

첫번째 레이어에서는 필수적으로 입력 레이어가 있어야한다.

- 입력레이어 튜플(input\_layer = (,)) 형태
  - 1차원 데이터: (특징 수,) - input\_dims = (특징 수) 로도 가능
  - 2차원 데이터: (시퀀스 길이, 특징수)
  - 3차원 데이터: (높이, 너비, 채널수)
  - 4차원 데이터: (배치크기, 높이, 너비, 채널수)

```
from keras.layers import Flatten

model.add(Flatten())
```

Flatten은 평탄화하다 라는 영어의미 그대로 입력 데이터를 1차원으로 바꿔주는 역할을 한다.

2D나 3D를 처리하기 쉽도록 1차원으로 처리한다.

모델을 완성하면 compile 메서드로 모델을 완성한다.

```
model.compile(optimizer, loss, metrics)
```

- optimizer는 최적화 알고리즘을 지정: SGD(확률적 경사하강법), RMSprop, Adagrad, Adam 등
  - 각 옵티마이저는 학습률과 필요한 인자에 따라 함수형태로 작성
    - EX) keras.optimizers.SGD(learning\_rate=0.01, momentum=0.0)
- loss는 손실함수 지정: mean\_squared\_error, categorical\_crossentropy, binary\_crossentropy 등
- metrics는 모델 성능을 평가할 기준을 설정: mse, accuracy, precision, recall, f1-score 등

모델을 설계하면, 모델을 학습시키기 위해 fit() 함수가 필요하다.

모델을 생성해 입력데이터(x), 정답데이터(y)를 모델에 넣으면 모델의 가중치를 업데이트하면서 진행된다.

```
fit(x, y, epoch = 1, verbose = 1, validation_data = None, callbacks,  
    shuffle = True, ...)
```

fit 함수를 통해 나온 객체(History 객체)는 모델의 손실과 평가지표를 그래프로 나타내거나, 학습 과정에서 최적의 모델을 저장하는데 사용한다.

history = fit(x, y, ...)로 객체 생성 후 history.history['loss']를 그래프에 나타내는 방식으로 사용한다.

- 훈련 후 세션의 모든 객체 삭제: keras.backend.clear\_session()

```
import keras.backend as K  
  
K.clear_session()
```

## 데이터 전처리(목적성, 절차 및 방법, 기초 문법)

- 목적성

데이터는 형태가 다양하고 품질도 다르기 때문에 데이터를 전처리 하는 과정이 중요하다.

Raw 데이터는 머신러닝에 적합하지 않기 때문에 반드시 필요하다.

데이터 분석의 정확도를 높이고 유효한 데이터 분석을 하도록 한다.

잘못된 전처리는 결과에 영향을 끼쳐 왜곡된 결과를 사용하게 될 수 있다.

- 데이터 전처리 절차/방법

1. 데이터 수집: 데이터 원본으로부터 수집한다. 웹 크롤링, DB, 파일, API 등을 통해 이뤄진다.
2. 데이터 탐색 및 이해: 수집한 데이터에 대해 구조/특성/분포를 분석하여 패턴을 이해한다.
3. 결측치 처리: 결측치를 채우거나 비운다. (NaN(Not a Number) 혹은 빈값으로 표시될 수 있음)
  - a. 삭제하거나 평균값/중앙값/최빈값으로 채워넣는다.
4. 이상치(Outlier) 처리: 다른 값들과 크게 차이나는 값인 Outlier를 제거한다.(튀는 값 제거 or 대체)
5. 데이터 스케일링: 표준정규분포나 정규분포를 이용해 표준화(Z-score), 정규화하여 작은 범위의 값으로 조정한다.
6. 범주형 데이터 인코딩: 문자열/정수 값으로 표현된 데이터를 수치형태로 바꾼다.
  - a. one-hot 인코딩, 레이블 인코딩 주로 사용
7. 특성 선택 및 추출: 분석에서 필요한 특성을 선택(=불필요한 특성 제거) 하고, 추출한다.
8. 데이터 정규화: 값 범위를 조정하여 분석에 적합한 형태로 만드는 과정이다.
  - a. 데이터를 평균 0, 표준편차가 1인 정규분포에 가깝게 만든다.
9. 데이터 변환: 원본 데이터에서 유용한 특성을 추출하거나 데이터를 적합한 형태로 변환한다.
  - a. 텍스트 데이터를 토큰화(Tokenization), 이미지 데이터를 전처리한다.

---

- 데이터 전처리 기초 문법(pandas)

- Series: 1차원, 1개의 column(pd.series([1,2,3,4])) 데이터 저장
- DataFrame: 2차원 가로세로축이 있는 데이터를 저장

```
import pandas as pd

# 예시용 데이터프레임 생성

data = pd.DataFrame({
    'A': [ 1,  2, None,  4,  5],
    'B': [None, 20, 30, 40, 50],
    'C': [100, 200, 300, 400, 500]
})
```

pandas의 DataFrame 메소드로 데이터프레임 생성 후 data 변수에서 메소드 호출을 통해 처리할 수 있다.

- 결측치 처리 기법
  1. dropna(): 결측치를 그냥 삭제
  2. fillna(data.mean()): 결측치를 평균으로 채우기
  3. fillna(data.mead()): 결측치를 중앙값으로 채우기
  4. apply(lamda x: x.fillna(x.mode()[0])): 결측치를 최빈값으로 채우기
  5. ffill() - 결측치를 front 값으로 채우기
  6. bfill() - 결측치를 back 값으로 채우기

```
# 결측치를 평균값으로 채우기
data_filled_mean = data.fillna(data.mean())
print("\n평균값으로 결측치 채우기:\n", data_filled_mean)

# 결측치를 중앙값으로 채우기
data_filled_median = data.fillna(data.median())
print("\n중앙값으로 결측치 채우기:\n", data_filled_median)

# 결측치를 최빈값으로 채우기
data_filled_mode = data.apply(lambda x: x.fillna(x.mode()[0]))
print("\n최빈값으로 결측치 채우기:\n", data_filled_mode)
```

- 이상값 제거(IQR: 4분위 범위 수 이용)

```
In [33]: Q3, Q1 = np.percentile(data, [75, 25])
         IQR = Q3 - Q1
         IQR
```

```
Out[33]: 1.1644925829790964
```

```
In [34]: data[(Q1-1.5*IQR > data)|(Q3+1.5*IQR < data)]
```

```
Out[34]: array([ 2.31256634,  8.          , 10.          , -3.          , -5.          ])
```

numpy의 `np.percentile(data, [start, end])`를 활용하여 75%부분과 25%부분을 빼 IQR을 구한다.

IQR에 Q1, Q3과 특정 범위(예시에선 1.5)를 곱해서 지정한 후 이를 넘어서면 이상치이므로 제거하면 된다.

Pandas의 DataFrame이나 Series는 대괄호 안에 boolean index가 들어오면 True인것만 반환하므로,

이를 통해 `data[조건문|조건문]` 형태로 True인 것만 반환하여 이상값을 제거한 데이터 배열을 반환한다.

- 압축파일 가져오기 / 정규화

```
import gzip
path = os.path.join(folder, filename)
with gzip.open(path, 'rb') as f:
    datas = np.frombuffer(f.read(), np.uint8, offset=8)
    data.reshape(-1, 28 * 28) / 255.0 # 정규화
```

1. 경로를 `os.path.join(folder, filename)`으로 파일경로를 찾아 path에 저장한뒤 `gzip.open`으로 압축해제
2. `np.frombuffer()`를 통해 label들을 버퍼로부터 읽어 넘파이 배열로 변경, 헤더정보 8 바이트는 버림

- 데이터 분할



```
# 훈련/검증 분할
n_samples = len(x_train)
train_size = 55000
val_size = n_samples - train_size

indices = np.random.permutation(n_samples)
x_train_split = x_train[indices[:train_size]]
y_train_split = y_train[indices[:train_size]]
x_val = x_train[indices[train_size:]]
y_val = y_train[indices[train_size:]]
```

`np.random.permutation(int)` - 정수형 array 배열 반환

인덱스 슬라이싱을 통해 훈련/검증데이터를 앞부분과 뒷부분으로 분할한다.

- pandas에서 외부 데이터 읽기

```
pd.read_csv('https://bit.ly/ds-house-price')

## 모든 열을 string으로 불러오기
pd.read_csv('sample.csv', dtype = str)

## 특정 열만 string으로 불러오기
#1단계 string으로 불러오고 싶은 열 이름을 list에 담기
lst_str_cols = ['prefix', 'serial']
```

`read_csv(파일이름 or 다운로드링크)`로 읽어온다.

string으로 지정을 하지않고 바로 불러오면 전화번호와 같이 앞에 0이 있는 경우 생략해 버린다.

- One-hot Encoding

벡터에서 하나의 '1'과 나머지는 '0'으로 구성된 형태이다.

개, 고양이, 여우 세 개의 범주가 있다면 첫 번째 범주는 [1, 0, 0], 두 번째 범주는 [0, 1, 0], 세 번째 범주는 [0, 0, 1]과 같이 정답 label을 설정한다.

Tensorflow 라이브러리에 원핫인코딩에 관한 메소드가 존재한다.

기본적으로 `tf.one_hot(label_index_list, depth)` 형태로 사용하고, 라벨 인덱스 목록, 라벨 개수 두개의 인자이다.

추가적으로 `dtype, axis, value`를 따로 지정할 수 있다.

자료형, 나열하는 기준축, 구성값에 해당한다.

```
import tensorflow as tf
idx = [0, 3, 0, 1, 2, 1, 3, 2, 1, 0]

tf.one_hot(idx, depth = 4)

'''
array([[1., 0., 0., 0.],
       [0., 0., 0., 1.],
       [1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 1., 0., 0.],
       [0., 0., 0., 1.],
       [0., 0., 1., 0.],
       [0., 1., 0., 0.],
       [1., 0., 0., 0.]
      ])
'''

tf.one_hot(idx, depth = 4, dtype = tf.int32)
tf.one_hot(idx, depth = 4, axis = 0)
tf.one_hot(idx, depth = 4, on_value = 3, off_value = -1)
```

- 첫번째 원핫은 인덱스가 0, 1, 2, 3에 따라 1,0을 매칭해 리스트를 인덱스 개수만큼 반환
- 두번째 원핫은 정수형태로 0과 1 출력
- 세번째 원핫은 열을 기준으로 행을 출력
- 네번째 원핫은 0과 1대신 -1과 3으로 출력
- pandas는 `pd.get_dummies(data)`를 사용하여 원핫인코딩을 출력할 수 있다.

- 정규화

scikit-learn으로 정규화하는 방법은 아래의 코드와 같다.

```
from sklearn.preprocessing import StandardScaler
import pandas as pd

df = pd.DataFrame({'x1' : np.arange(11), 'x2' : np.arange(11) ** 2})

scaler = StandardScaler()
df_std = scaler.fit_transform(df)

pd.DataFrame(df_std, columns = ['x1_std', 'x2_std'])
```

scaler를 StandardScaler()로 지정하고, 데이터셋 df에 fit\_transform를 하면 scaler에 의해 정규화가 된다.

scaler를 MinMaxScaler()로 지정하면 최솟값 0, 최댓값 1을 기준으로 정규화한다.

scaler를 RobustScaler()로 지정하면 데이터의 50%(Q2)를 0으로 하여 IQR(Q2와 25%차를 기준인 Q1, Q3)의 차로 정규화를 진행한다. 이상치(Outlier)에 강한 정규화 방법이다.

---

## 주피터(or 코랩) 사용법

### 주피터 사용방법

Anaconda Prompt를 사용해 Python 가상환경을 생성하고 활성화시킨다.

각종 라이브러리 간 충돌을 막기 위해 가상환경을 만들고 conda activate 명령어를 통해 활성화 시킨 뒤 프롬프트 내부에서 pip를 사용해 각종 라이브러리를 설치하고 가상환경을 이용해 파이썬 파일을 다루기 위한 ipykernel을 설치할 수 있다.

주피터 노트북을 키면 cmd창과 함께 브라우저를 통해 localhost에 연결된 home 화면으로 진입한다.

주피터에서 upload를 통해 로컬 파일을 작업환경으로 업로드하거나, New를 통해 ipykernel을 지정하고, 파일, 폴더, 터미널 등을 생성할 수 있다.

파이썬 파일을 생성하면 .py가 아닌 .ipynb 확장자로 저장되고 내부에서는 셀단위로 코드를 실행한다.

셀은 쓰고싶은 형식을 따라 Code, Markdown, Raw를 지정할 수 있다.

Run Selected Cell	Shift+Enter
Run Selected Cell and Insert Below	Alt+Enter
Run Selected Cell and Do not Advance	Ctrl+Enter

- Ctrl+Enter → 현재 선택중인 셀 실행 후 현재 셀에 머무른다.
- Shift+Enter → 현재 선택중인 셀 실행 후 다음 셀로 이동
- Alt+Enter → 현재 입력중인 셀 실행 후 아래에 셀 삽입
- 텍스트모드 ↔ 셀모드: Enter 시 수정모드 ↔ Esc 시 셀모드
- Shift+L → 라인(몇번째 줄인지) 표시
- 텍스트모드에서 Ctrl+z는 실행취소이지만 되돌리기는 Ctrl+Shift+z이다.
- 셀에 대해서는 셀에 관한 실행취소는 z, 되돌리기는 Shift+z이다.
- 셀에 대해서 x는 잘라내기, c는 복사, d+d는 삭제, v는 아래에 붙여넣기이다.

## 참고자료

[\[Machine Learning\] 머신러닝의 핵심 이해: 비용 함수\(Cost Function\)와 손실 함수\(Loss Function\)의 차이점](#)

[1. 간단한 예제로 케라스 맛보기](#) - 자료 사이트의 1번 목차들, 문법의 많은 부분에 참고

[07-08 케라스\(Keras\) 훑어보기 - 딥 러닝을 이용한 자연어 처리 입문](#)


[The Sequential class](#) - 케라스 공식 사이트 메소드 참고

밑바닥부터 시작하는 딥러닝 Chapter 6 ppt(오버피팅 사전, 방지법)

[\[Deep learning\] 데이터 전처리 방법 및 기술](#) - 데이터 전처리 목적

[\[Machine Learning\] 데이터 전처리\(Data Preprocessing\) 개념과 종류 및 예시 — 공간](#)  
- 데이터 전처리 목적, 절차

[\[Python\] 데이터 전처리 하기 / 결측치, 중복 데이터, 이상치, 정규화, 원-핫 인코딩, 구간화](#)  
— 초보 개발자의 일기장

 [\[Python\] 데이터 전처리](#)

[Tensorflow] 텐서플로우 원핫 인코딩 함수 : tf.one\_hot()

pdf파일 개인톡으로 보내주세요. 금요일 스터디때 발표합니다!