- 딥러닝 기초

  - 딥러닝 개요

  기존 알고리즘과 데이터를 입력하여 결과값을 갖는 기존 프로그래밍 방식과 달리 데이터와 출력결과를 학습하여 알고리즘을 도출해내는 기계학습을 사용한다. 딥러닝은 머신러닝의 일부로, ANN, DNN, CNN RNN 이 있다.

  - MLP(Multilayer perceptrons)

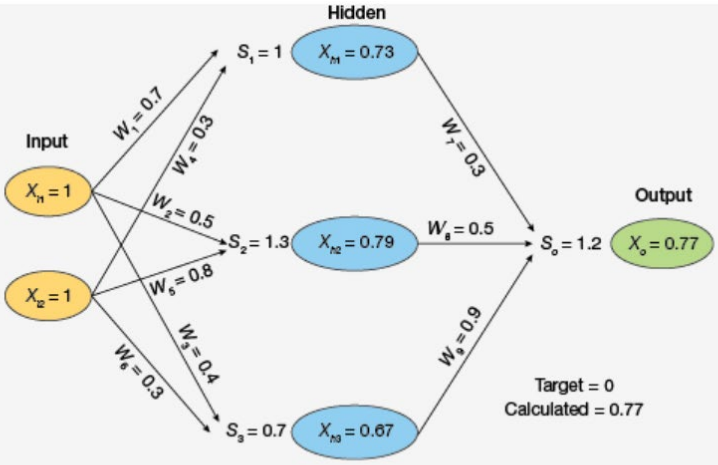
    입력과 출력 사이에 여러 층의 뉴런이 있는 인공 신경망이다. 한 방향으로만 흐르며 입력과 출력 사이에 있는 layer는 은닉층이라 한다.

    MLP는 패턴 분류, 인식, 예측 및 근사에 많이 사용되며 선형 곡선 혹은 쉽게 표현가능한 곡선을 이용하여 복잡한 패턴을 학습한다. 이때, 뉴런의 계층수에 따라 복잡한 패턴을 학습할 수 있는 용량이 증가한다.

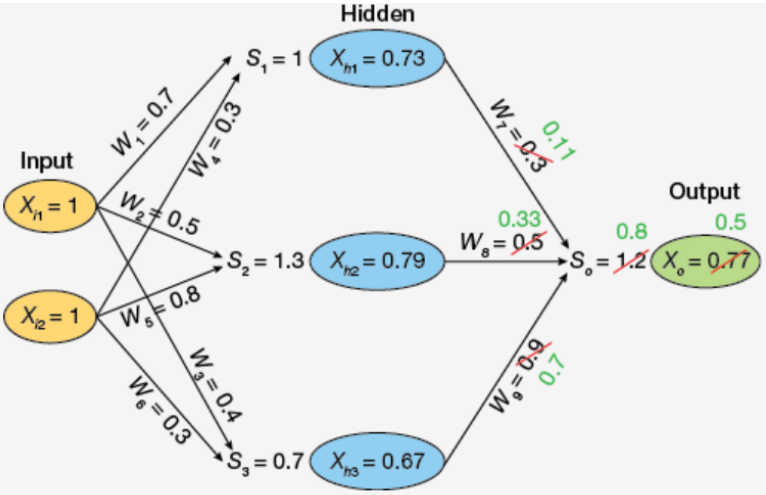
    MLP를 훈련하는데 가장 일반적인 모델로 사용되는 알고리즘은 역전파(Backpropagation)이다. 이를 사용하여 출력이 목표값에 가까워지도록 한다.

    시그모이드 함수 출력오류값과 시그모이드함수 변화율 을 곱하여 은닉층의 해당 출력에 비례하여 들어오는 가중치를 조절 후 입력과 은닉층 사이의 가중치를 조절하는 것을 반복한다.

    (W = 가중치, S = 가중치의 합 X = 뉴런)



▲**Figure 12.** An example MLP with three layers accepts an input of [1, 1] and computes an output of 0.77.



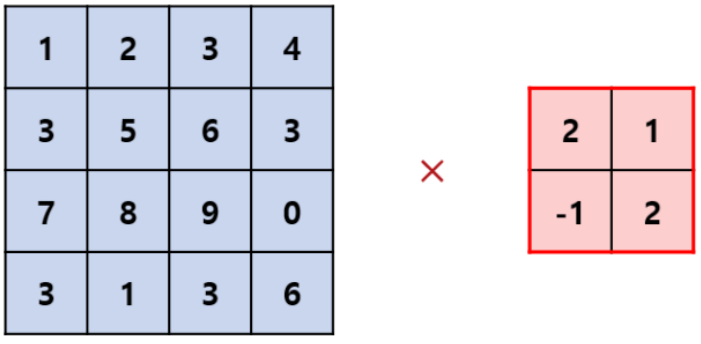
▲ **그림 13.** 역전파 알고리즘을 사용하여 은닉층 뉴런과 출력 뉴런 간의 가중치를 조절하여 출력이 목표값(0)에 가까워지도록 한다.

- 합성곱 연산

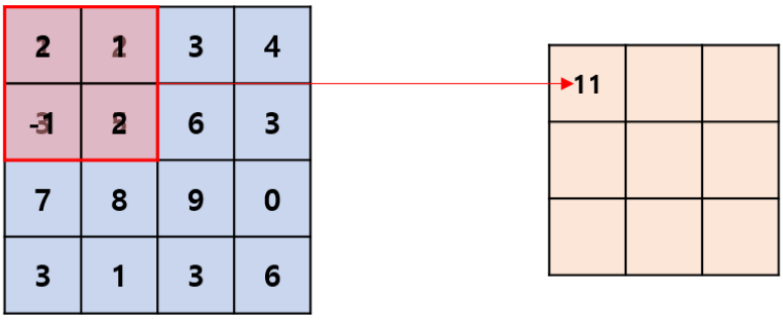
변화시키고 싶은 함수가 주어졌을 때, 원하는 목적에 따라 다른 함수를 선정하여 분해, 필터, 변환 등에 사용하기 위해 사용되는 계산법이다.

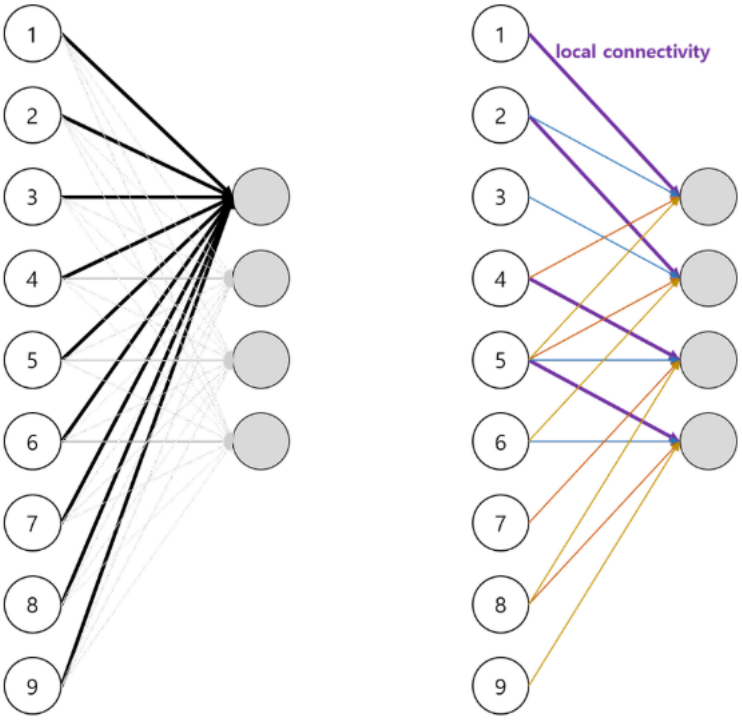
$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau$$

(convolution 계산)



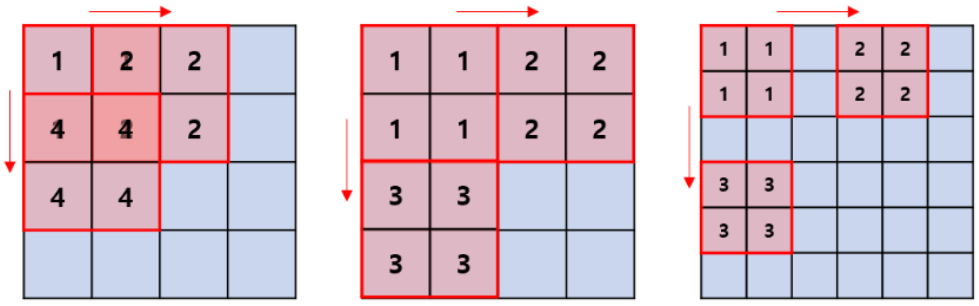
[이미지] [필터]





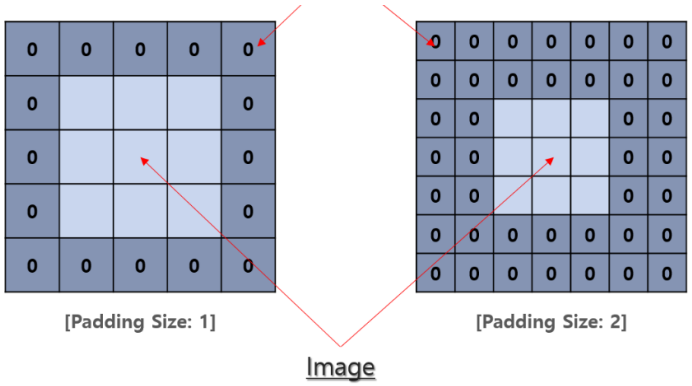
[Fully Connected Layer] [Convolutional Layer]

Stride(스트라이드) : 필터를 입력데이터나 특성에 적용할 때 움직이는 간격



[Stride: 1] [Stride: 2] [Stride: 3]

Padding(패딩): 반복적으로 합성곱 연산을 적용 시 행렬의 크기가 작아짐을 방지와 정보손실을 줄임



- 비용함수

  훈련 단계에서 손실을 단일 실수 형태로 정량화 하는데 사용되는 함수

  단일 훈련에 대해 오류를 나타날 때 사용

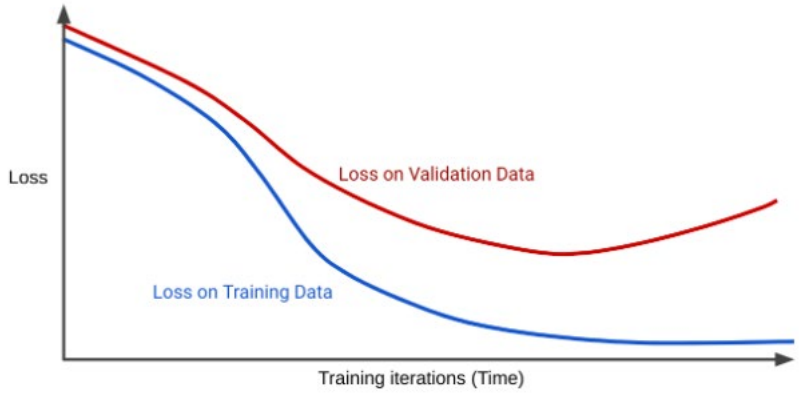
  회귀 비용 함수, 이진 분류 비용 함수, 다중 클래스 분류 비용 함수 등 사용

- 손실함수

  전체 훈련 데이터에 대한 손실 함수의 평균을 나타내는 데 사용

- 과적합

  학습을 과도하게 최적화 과정을 진행했을 시 생기는 문제이다. 일반화 곡선을 사용하여 두개의 손실 곡선이 유사성을 판단하여 과적합을 감지할 수 있다.



- Keras

  가) 환경설정

  - Tensorflow.keras lib 을 사용하기위해서 구글코랩을 사용하면 별다른 설치없이 사용가능하다. 또 다른 방법은 window11 기준 wsl2를 다운받아 우분투를 설치 후 pip을 이용해 tensorflow 와 keras를 설치 후 가상환경에서 실행하면 된다.
  - Import tensorflow as tf #tensorflow모듈을 tf변수이름으로 지정
  - Import numpy as np #numpy모듈을 np변수이름으로 지정
  - From tensorflow import keras #tensorflow의 keras 모듈를 사용

- From tensorflow.keras import # keras의 부분적인 모듈을 가져오기

  Layers

나) 구성요소 & 문법 & 특징

  1) Sequential

- 한 종류의 입력값과 한 종류의 출력값만 사용 가능, 데이터 층은 순서대로 적용된다.

- Summary()

Model: "mnist_model"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 784)	0
dense (Dense)	(None, 64)	50,240
dense_1 (Dense)	(None, 64)	4,160
dense_2 (Dense)	(None, 10)	650

Total params: 110,102 (430.09 KB)
Trainable params: 55,050 (215.04 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 55,052 (215.05 KB)

  A) 특별히 지정하지 않으면 이름은 자동할당

  B) 층별 출력값의 모양(Output Shape): 유닛수 확인 none은 임의의 배치 크기를 나타낸다.

  C) 층별 파라미터의 수(Param #): 훈련을 통해 학습되어야 하는 파라미터들의 수

  D) Total Params: 파라미터의 총개수

  E) Trainable Params: 가중치 행렬과 편향 벡터에 포함된 파라미터처럼 훈련을 통해 학습되는 총 개수

  F) Non-trainable Params: 훈련에는 관여하지만 학습은 되지않는 파라미터

- Input()

  A) Keras.Input(shape=(dim x, dim y, dim z)) #(x,y,z 는 차원의 샘플 개수)

- layers

    a) Dense

    Layers.Dense(출력 뉴런 수, 입력 뉴런 수 , activation= "fuction"),

    b) Convolution

    Conv2D(필터수, 필터 사이즈,padding=vaild or same, input_shape=(x, y, z)'첫 레이어일시', activation),

    c) max pooling

    MaxPooling2D(pool_size=(2,2))

    d) flatten

    flatten()

    e) LSTM

    LSTM(메모리 셀 수, input_dim=입력 속성 수, input_length=시퀀스 데이터의 입력 길이)

2) API

- 입력층

  하나의 입력층만 사용

  inputs = keras.Input(shape=(3,), name="my_input")

- 은닉층

  입력층의 값을 이어받는다. 여러 개의 레이어로 구성 가능

  features = layers.Dense(64, activation="relu")(inputs)

- 출력층

  하나의 출력층만 사용

  outputs = layers.Dense(10, activation="softmax")(features)

- 모델 빌드

  입력층과 출력층을 이용하여 모델 지정

  model = keras.Model(inputs=inputs, outputs=outputs)

A) 다중 입력, 다중 출력 모델

B) 신경망 모델 구조 그래프

3) 서브클래싱

- 데이터 전처리

1) 데이터 전처리란?

- 데이터세트에서 손상되거나 부정확한 레코드를 탐지하고 수정하는 과정 즉, 불필요한 데이터를 삭제, 수정, 교체를 하여 가공하는 단계이다.
- 가공하게 되면 필요한 모델의 정확도가 증가하여 신뢰성을 개선할 수 있다.

2) 데이터 전처리 목적

- 정확도와 신뢰성을 향상
- 데이터를 일관되게 가공
- 데이터 알고리즘의 가독성을 높임

3) 데이터 전처리 과정

가) 데이터 평가

- 원하는 데이터를 식별하고 수정하는 과정
- Imputation(대치), removal(제거), transformation(변환) 을 사용

나) 데이터 정리

- 여러 소스의 데이터를 결합하여 통합된 데이터 세트를 만드는 과정
- 누락된 데이터를 대체하고 관련성이 없거나 중복된 데이터를 수정
- Regression(회귀 모델), Binning(구간화 모델), clustering(군집 모델)

다) 데이터 통합 및 변환

- 알고리즘이 쉽게 읽고 해석할 수 있는 적절한 형식으로 데이터를 변환하는 과정
- 집계(aggregation), 정규화(normalization), 이산화(discretization), 일반화 (generalization) 방법이 있다.

- 이산화: 연속 적인 데이터를 이산화 범주로 변환
- 정규화: 데이터를 공통 범위로 조정
- 표준화(일반화): 평균이 0이고 분산이 1이 되도록 변환

라) 데이터 축소

- 중요 정보 이외의 데이터들을 제거하여 데이터 세트의 크기를 줄이는 과정
- 피처 선택 및 피처 추출을 이용

4) 데이터 전처리 문법(pandas)

- Pandas : 파이썬 라이브러리이며 데이터 분석과 전처리를 위해 사용

가) 데이터 파일

a) CSV 파일 읽기와 쓰기

- pd.read_scv((1) filepath,(2) sep = ',' ,(3) header = 'infer',

(4) names = ,(5) index_col = ,(6) usecols = ,(7) dtype = ,

(8) skiprows = ,(9) nrows= , (10) encoding = ,)

(1) Filepath : 파일을 읽어오기위해 경로 지정 보통 변수에 경로를 선언하여 기입

(2) Sep: 데이터 필드를 구분하는 문자, 값과 값 사이의 구분자 기본값은 (,)이다.

(3) Header: 데이터프레임의 열 이름(column header)사용될 행의 번호 기본값은 첫번째 행(header=0)이 열 이름으로 사용, 특정 행을 열 이름으로 지정하지 않고 모두 데이터를 처리하려면 'header=None'으로 설정

(4) Names: 열 이름을 명시적으로 지정. 주어진 CSV파일에 헤더가 있어도 헤더를 무시하고 지정 이름을 사용하며, 이때 CSV파일의 헤더가 데이터의 일부로 처리되지 않게 하기 위해 'header=0'으로 지정. CSV파일에 헤더가 없는 경우 header옵션을 사용하지 않거나 None으로 설정한 다음 사용

(5) Index_col : CSV파일의 특정 열을 행 인덱스로 사용하기 위해 해당 열의 번호나 이름을 전달하는 매개변수

(6) Usecols: CSV파일에서 특정 열만 선택하여 가지고 오는데 사용하는 매개변수

(7) Skiprows: 파일의 시작 부분에서 몇 줄을 건너뛰고 데이터를 로드할지 결정하는 매개변수. 특정 줄을 무시하거나 파일헤더가 비정상적인 위치에 있을 때 사용

(8) Nrows: 파일에서 몇 줄을 읽을지 지정하여 파일의 일부분만 가져올 수 있게 하는 매개변수

(9) Encoding: 파일의 문자 인코딩 방식을 지정. 기본값은 'utf-8'이 지정

- df.to_csv((1) file_name,(2) sep=',',(3) na_rep=' ',

(4) columns= ,(5) header= ,(6) index = ,(7) encoding,)

(1) file_name: 경로를 포함하여 저장할 파일명을 지정하여 데이터프레임 데이터를 파일로 저장한다.

(2) Sep: 데이터 필드를 구분하는 문자. 값과 값 사이의 구분자를 지정하는 매개변수로, 기본값은 쉼표(,)

(3) Na_rep: CSV파일로 저장할 때 결측치를 대체할 문자열을 지정하는 매개변수. 기본적으로 빈 문자열로 대체

(4) Columns: CSV파일로 저장할 때 포함시킬 열을 지정하는 매개변수. 전체 데이터프레임에서 특정 열만 선택하여 파일로 저장하기 위해 사용

(5) Header: CSV파일로 데이터를 지정할 때 열 이름을 포함시킬지 여부를 결정. True와 False로 포함여부를 결정

(6) Index: 데이터프레임의 행 인덱스를 파일에 포함할지 여부를 지정. True False로 index를 별도로 설정할지 결정

b) 엑셀파일 읽기 & 쓰기

- (읽기) pd.read_excel((1)excel_file,(2) sheet_name=' ',(3)header=None ,

(4) names=None , (5) index_col=None ,(6) usecols=None,

(7) skiprows=None, (8) nrows=None,)

(1) Excel_file: 엑셀을 읽어와 판다스 데이터프레임의 데이터로 변환. 특정 경로를 포함가능

(2) Sheet_name: 엑셀 파일에서 읽어올 특정 시트를 지정. None으로 지정하면 모든 시트를 가저옴. 기본값은 0

(3) Header: 데이터프레임의 열 이름으로 사용될 행의 번호를 의미. 별도로 지정하지 않으면 기본적으로 첫번째 행이 열 이름으로 사용. 엑셀 파일에서 특정 행을 열 이름으로 지정하지 않고 모두 데이터로 처리하고자 한다면 'header=None'으로 설정

(4) Names: 열 이름을 명시적으로 지정하기 위해 사용되며 열 이름 리스트를 인자로 전달.

(5) (6) (7) 은 CSV와 동일

- (쓰기) df.to_excel((1) excel_file, (2) header,(3) sheet_name= '',

(4) startrow= ,(5) startcol= ,)

나) Info() : 데이터프레임의 열, 데이터 타입, 결측치의 유무, 메모리 사용량 등 데이터 프레임에 대한 요약 정보를 제공

다) 행렬 데이터 선택

a) (열 선택): 변수 = 데이터프레임명['열 이름']

Loc[:, 'column name']: 전체 행(:) 중 선택된 여러 열 출력

b) (행 선택): Head(n): 데이터프레임의 처음 n개의 행을 출력

Tail(n): 데이터프레임의 끝의 n개의 행을 출력

.Loc[]: 선택한 여러 행을 출력

.iloc[]: loc과 같은 기능 하지만 슬라이싱이 차이남

c) (슬라이싱): 데이터프레임명[시작인덱스:종료인덱스]

라) 데이터프레임 수정

a) 결측치 처리

Isnull() : bool(True&False)값으로 결측치 여부를 나타냄 .sum() 함수를 같이 사용하면 결측치의 수를 계산한다.

Fillna(value) : 결측치를 value값으로 대체

Dropna() : 데이터프레임이나 시리즈의 결측치를 포함하는 행이나 열을 제거 할 때 사용

마) 데이터프레임 병합

a) merge([dt1, dt2 …],on= ' ') : 공통된 컬럼을 기반으로 합쳐짐 공통컬럼은 on에 파라미터로 리스트형태로 입력

- 코랩 사용법

1) 노트북의 부족한 스펙 & 개발 환경 구축과정을 생략하여 사용가능한 구글의 서비스
2) 구글 드라이브에 접속하여 새 폴더를 생성
3) 새로 만들기 에서 더보기 란에 구글 코랩을 선택
4) 런타임 유형에서 CPU 대신 T4(GPU)를 선택 후 자유롭게 코딩