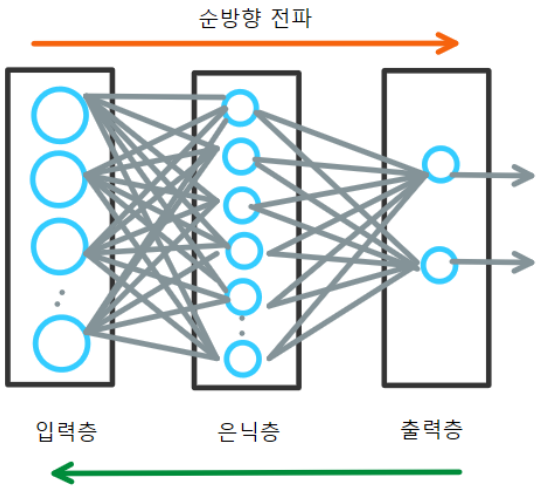
인공지능

MLP(Multi-Layer Perceptron)란?

여러 개의 퍼셉트론 뉴런을 여러 층으로 쌓은 다층신경망 구조로

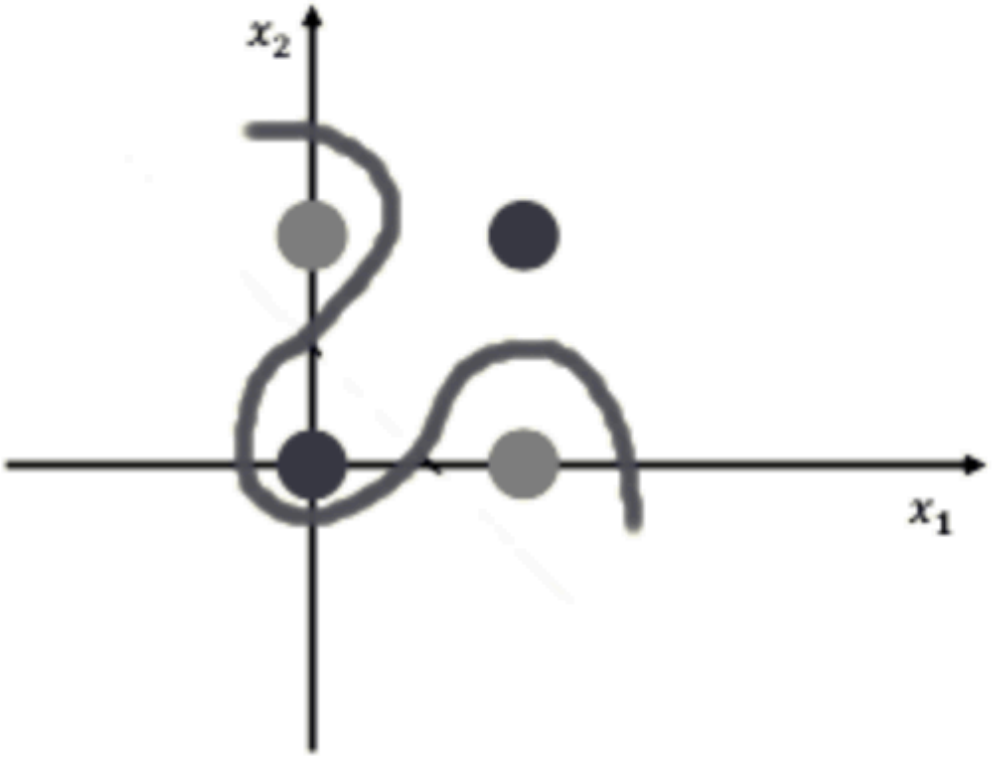
입력층과 출력층 사이에 하나 이상의 은닉층을 가지고 있는 신경망이다.



▼ SLP가 아닌 MLP를 쓰는 이유

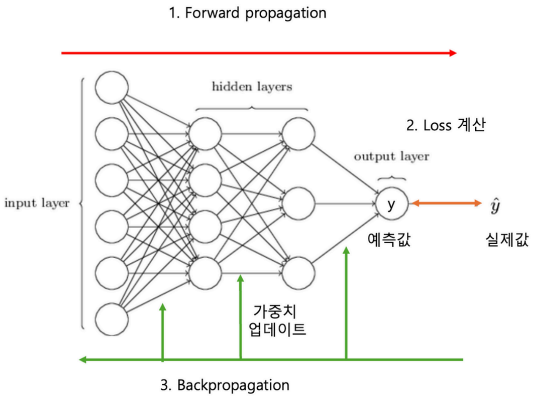
→ SLP에서 OR, AND 게이트 구현 가능

→ XOR 게이트는 선형적으로 분리될 수 없어 구현 불가능, 이와 같이 복잡한 문제는 MLP에서 구현



다층퍼셉트론은 복잡한 비선형 관계를 학습할 수 있어 이미지 분류, 텍스트 분류, 음성 인식 등 다양한 분야에서 활용된다.

- 입력 : 입력층의 노드 수는 Feature의 수와 일치함.
- 가중치 : 각 입력 특성에 대해 가중치가 할당됨. 가중치는 모델이 학습하는 매개변수로, 특징의 중요도를 나타냄
- 출력 : 입력에 대한 결과 값(Label)에 해당함.



다층 퍼셉트론은 적절한 가중치를 찾아가는 과정

이때 사용되는 손실 함수는 모델의 예측과 실제 값 간의 차이를 측정하는 것으로 **역전파** 알고리즘이 사용된다.

▼ 역전파 알고리즘

출력과 정답 간의 오차를 역방향으로 전파하여 각 층의 가중치를 조정하는 과정을 반복

경사하강법과 같은 최적화 알고리즘을 통해 손실 함수를 최소화한다.

손실 함수

- 손실 함수는 모델의 단일 데이터에 대한 예측이 얼마나 잘못되었는지를 측정
- 회귀 모델에서는 MSE(Mean Squared Error) 평균 제곱 오차가 사용됨
- 손실 함수의 값이 낮을 수록 모델의 예측률이 좋은 것

비용함수

- 비용 함수는 전체 훈련 데이터셋에 대한 모델의 평균적인 성능을 평가
- 손실 함수의 결과 값들을 평균내어 계산
- 모델을 최적화하는 것이 목적으로, 비용 함수의 값이 낮을 수록 모델의 성능이 최적화됨.

과적합

- 모델이 학습 데이터를 많이 학습한 경우로, 데이터의 노이즈나 불필요한 패턴까지 학습
- 학습 데이터가 적거나 특성의 개수가 많은 경우에는 과적합이 될 수 있다.
- 과적합 방지
    1. 데이터의 양 늘리기
    2. 모델의 복잡도 줄이기 → 은닉층의 수나 매개변수의 수 등에 변화를 줌
    3. 가중치 규제(Regularzation) 적용
        a. L1 규제 : 가중치 w들의 절대값 합계를 비용 함수에 추가
        b. L2 규제 : 모든 가중치 w들의 제곱합을 비용 함수에 추가
    4. 드롭아웃(Dropout) : 학습 과정에서 신경망의 일부를 사용하지 않음

Keras

케라스는 딥러닝 모델을 만들기 위한 고수준의 구성요소를 제공하는 모델 수준의 라이브러리이다.

텐서 조작이나 미분 같은 저수준의 연산은 백엔드 엔진으로 제공하는 텐서플로, CNTK, 씨아노 등의 텐서 라이브러리를 사용한다.

특징

- 사용 용의성 : 사용자 친화적이고 직관적인 API를 제공, 간단한 코드로 복잡한 딥러닝 모델을 설계할 수 있음. 이로 인해 학습 곡선이 낮음
- 모듈화 : 모듈화된 방식으로 설계되어, 모델 구축의 각 요소를 독립적으로 정의하고 조합할 수 있다. → 복잡한 모델을 설계하고 실험할 때 유용
- 백엔드 엔진 기반
- 확장성 : 사용자 정의 기능 지원
- 멀티플랫폼 및 CPU/GPU 지원 : 빠른 연산 가능
- 다양한 데이터 타입 지원 : 텍스트, 이미지, 시계열 데이터 등 다양한 데이터 유형을 지원
- TensorFlow와의 통합 : TensorFlow의 성능과 유연성을 활용하면서도, Keras의 사용성을 유지한 환경을 제공

구성요소

1. 모델구조
    - Sequential API : 레이어를 순차적으로 쌓는 방식으로 간단한 모델 구축에 적합

```
model = Sequential([
    Dense(64, activation='relu', input_shape=(100,)),
    Dense(10, activation='softmax')
])
```

    - Functional API : 복잡한 네트워크를 정의할 수 있는 방식

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense

inputs = Input(shape=(100,))
x = Dense(64, activation='relu')(inputs)
outputs = Dense(10, activation='softmax')(x)
model = Model(inputs, outputs)
```

- Subclassing API : 고도로 사용자 정의된 모델을 구현하기 위한 객체지향 방식

```
from tensorflow.keras import Model
from tensorflow.keras.layers import Dense

class MyModel(Model):
    def __init__(self):
        super(MyModel, self).__init__()
        self.dense1 = Dense(64, activation='relu')
        self.dense2 = Dense(10, activation='softmax')

    def call(self, inputs):
        x = self.dense1(inputs)
        return self.dense2(x)
model = MyModel()
```

2. 레이어
    - 모델의 기본 구성 요소로, 데이터를 처리하는 주요 연산 단위
3. 컴파일
    - 모델 학습 전에 손실 함수, 최적화 알고리즘, 평가지표를 설정하는 과정
4. 학습 및 평가
    - fit() : 모델 학습
    - evaluate() : 모델 평가
    - predict() : 새로운 데이터에 대한 예측

문법

모델 학습 관련 설정(최적화 함수, 손실 함수, 평가 지표)

```
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['ac
```

1. Optimizer(최적화 함수) : 모델의 가중치 설정(손실 함수 최소화 역할)
    a. SGD(Stochastic Gradient Decent) : 무작위로 일부 데이터를 사용해 기울기 계산
    b. Momentum : 이전 단계의 기울기를 가속도로 활용, SGD보다 빠르게 수렴하고 싶을 때
    c. RMSProp : 학습률을 동적으로 조정, 각 방향의 기울기를 따로 조정해 진동 감소, RNN등 시계열 모델에 효과적
    d. adam(Adaptive Moment Estimation) : RMSProp와 Momentum의 장점을 결합
    e. adagrad : 희소한 특징에 적합, 학습률이 점점 줄어듦, 자연어처리나 희소데이터에 적합
2. Loss 함수 : 모델 학습을 위한 가중치 최적화 목표

3. Mertics(평가 지표) : 학습 후 모델의 성능 측정
    a. 회귀 : MSE < MAE, R-squared
    b. 분류 : Accuracy, Precision, Recall, F1 score, ROC-AUC
    c. 시계열 : MASE
4. 모델 학습 : model.fit

```
model.fit(X_train, y_train, epochs = (몇 번을 학습시킬지))
```

- epochs : 전체 데이터 셋을 몇 번 반복해서 학습할 지 결정
- batch_size : 몇 개의 샘플 단위로 나눠서 학습할 지 설정, 미니배치 경사하강법
- callbacks : 학습 과정 중 특정 조건에서 실행될 작업(콜백 함수)를 설정
- validation_data : 별도의 검증 데이터셋이 있을 때

- validation_split : 별도의 검증 데이터셋이 없지만 검증하고 싶을 때
- shuffle : 학습 데이터의 순서를 랜덤으로 섞을지 여부 (시계열의 경우 섞지 않음)

데이터 전처리

데이터 분석을 위해 수집한 데이터를 분석에 적합한 형태로 가공하는 과정이다.

데이터 전처리를 통해 **불필요한 데이터를 제거**하고, 결측치나 이상치를 처리하여 **데이터의 질을 향상** 시킬 수 있다. → 분석 모델을 구축하고 결과를 도출하는 데에 더욱 유용하게 활용될 수 있다.

절차 및 방법

<과정>

데이터 수집 → 데이터 정제 → 데이터 통합 → 데이터 축소 → 데이터 변환

1. 정제
    a. 결측값 처리
        i. 결측값 제거(개체/속성), 입력(수동, 전역 상수)
        ii. 추정(회귀 분석, 의사결정트리)
    b. 잡음 제거
        i. 오류/오차로 인한 경향성 훼손 방지
        ii. 데이터 평활화(구간화, 회귀, 군집화)
2. 통합
    a. 개체 식별
        i. 서로 다른 데이터 집합의 개체 식별
        ii. 함수적 종속성, 메타데이터 활용
    b. 중복 제거/ 이상 해소
        i. 데이터 중복에 따른 공간 낭비/이상 발생 해소
        ii. 정규화/반정규화, 유도 속성, 상관 분석

3. 축소
    a. 데이터 큐브
        i. 다차원 집계 정보 추상화로 데이터 축소
        ii. 데이터 큐브 슬라이싱, 큐브 격자
    b. 속성 부분집합 선택
        i. 연관성이 낮거나 중복된 속성을 제거
        ii. 의사결정 트리, 엔트로피, 지니 계수, 가지치기
    c. 차원 축소
        i. 원천 데이터 부호화 및 압축
        ii. 웨이블릿 변화, PCA, DWT, 회귀/로그 선형 모형
    d. 수량 축소
        i. 표본 추출(데이터 샘플 부분집합 표현)
        ii. 히스토그램 구간화, 군집화(그룹화)
4. 변환
    a. 정규화
        i. 데이터세트 범위의 차이를 공통 척도로 변경
        ii. 최소-최대 및 Z-score 정규화, 소수 척도화
    b. 수치 데이터 이산화
        i. 엔트로피 기반 클래스 분포 계층화 이산화
        ii. 카이제곱 x^2 결합, 직관적 분활 이산화
    c. 집합화
        i. 범주형 데이터 계층 생성
        ii. 스키마 단계 생성, 명시적 그룹화

기초 문법

데이터 불러오기

- csv 파일

  pandas 라이브러리의 read_csv() 함수를 사용

  ex) df = pd.read_csv('data.csv')

- 엑셀 파일

  pandas 라이브러리의 read_excel() 함수를 사용

  ex) df = pd.read_excel('data.xlsx')

데이터 탐색

- 기본 정보 확인

  df.head(), df.tail(), df.info(), df.describe() 함수를 사용하여 데이터의 앞부분, 뒷부분, 데이터 타입, 기술 통계 등을 확인

데이터 정제

- 열 이름 변경

  df.columns = ['새로운 열 이름'] 또는 df.rename(columns={'기존 열 이름' : '새로운 열 이름'}, inplace = True)

- 불필요한 열 제거

  df.drop('열 이름', axis = 1, inplace = True)

- 행 필터링

  df[df['열 이름'] > 값]

결측치 처리

<결측치 제거>

- df.dropna() 함수를 사용하여 결측치가 있는 행 또는 열을 제거할 수 있다.

<결측치 대체>

- 평균값 대체 : df['열 이름'].fillna(df['열 이름'].mean(), inplace = True)
- 중앙값 대체 : df['열 이름']fillna(df['열 이름'].median(), inplace = True)
- 최빈값 대체 : df['열 이름']fillna(df['열 이름'].model()[0], inplace = True)

- 특정 값으로 대체 : df['열 이름']fillna(값, inplace = True)

데이터 변환

- 범주형 데이터 변환

  pd.get_dummies()함수를 사용하여 범주형 데이터를 원-핫 인코딩할 수 있다.
- 수치형 데이터 변환

  np.log(), np.sqrt() 함수 등을 사용하여 수치형 데이터를 변환할 수 있다.
- 문자열 데이터 변환

  str.lower(), str.upper(), str.replace() 함수 등을 사용하여 문자열 데이터를 변환할 수 있다.

주피터 사용법

1. 모드

- 일반 모드 : 셀을 선택, 이동, 추가, 삭제 등의 작업을 수행할 수 있다.
  - 셀의 왼쪽 테두리가 파란색
- 편집 모드 : 셀의 내용을 편집하거나 코드 작성
  - 셀의 왼쪽 테두리가 초록색

2. 코드 실행

- 셀을 선택한 후 shift + enter(다음 셀로 이동)를 누르거나 셀 왼쪽 실행 버튼을 클릭

3. 단축키

<일반 모드>

- A : 선택된 셀 위에 새로운 셀 추가
- B : 선택된 셀 아래에 새로운 셀 추가
- X : 선택된 셀 잘라내기
- C : 선택된 셀 복사

- V : 선택된 셀 아래에 붙여넣기
- Shift + V : 선택된 셀 위에 붙여넣기
- DD : 선택된 셀 삭제
- Z : 셀 삭제를 취소
- Y : 선택된 셀을 코드 셀로 변경
- M : 선택된 셀을 마크다운 셀로 변경
- R : 선택된 셀을 Raw 셀로 변경
- Ctrl + Enter : 선택된 셀 실행(다음 셀 이동 x)
- Alt + Enter : 선택되니 셀 실행 후 아래에 새로운 셀 추가

<편집 모드>

- Ctrl + Y : 되돌리기 취소
- Ctrl + A : 셀 내용 전체 선택
- Ctrl + X : 선택한 내용 잘라내기
- Ctrl + C : 선택한 내용 복사
- Ctrl + V : 선택한 내용 붙여넣기
- Ctrl + Home : 셀 맨 위로 이동
- Ctrl + End : 셀 맨 아래로 이동
- Ctrl + D : 현재 줄 삭제