Deep Learning

1. *MLP (multi-layer perceptron, MLP)*

   a.

   퍼셉트론(인공신경망 모델 중 하나)**으로 이루어진 층(layer) 여러 개를 순차적으로 붙여 놓은 형태**

   비선형 활성화 함수를 갖는 완전히 연결된 뉴런으로 구성된 신경망의 한 유형

   선형 분리가 불가능한 데이터를 구별하는데 많이 사용

   b.

   1. **입력층** (Input Layer)

      퍼셉트론과 마찬가지로 입력 데이터를 받아들이는 Layer (입력 데이터의 특징(Feature)에 해당)

      입력 데이터를 수신하는 노드 또는 뉴런으로 구성

      각 뉴런은 입력 데이터의 특정이나 차원을 나타냄

      입력층의 뉴런 수는 **입력 데이터의 차원**에 따라 결정

   2. **은닉층** (Hidden Layer)

      입력층과 출력층 사이에 위치한 층 (입력층에서 전달받은 특성들을 훈련(연산)시키는 층)

      MLP의 성능을 높이는 중요한 역할

      은닉층의 각 뉴런은 이전 층의 모든 뉴런으로부터 입력을 받고 다음 층으로 전달되는 출력을 생성

      출력층으로 훈련된 데이터를 전달

      신경망의 외부에서는 이 층에 직접 접근할 수 없음 (계산의 결과를 사용자가 볼 수 없기 때문에 '은닉층')

      은닉층이 2개 이상일 때 심층 신경망이라 하고, deep learning으로 봄

   3. **출력층** (Output Layer)

      은닉층의 출력 결과를 받아 최종 출력을 생성하는 Layer (출력은 해당 입력에 대한 결과값(Label)에 해당)

출력층 또한 활성화 함수를 적용하는데 이 활성화 함수는 일반적으로 숨겨진 계층에서 사용되는 것과 다름

은닉층에서 받은 데이터의 총 합을 활성화 함수를 통해 변환

2. *비용 함수, 손실 함수*
   a. **비용 함수** (Cost Function)

      전체 훈련 데이터셋에 대한 **모델의 평균적인 성능**을 측정

      목적은 모델을 최적화하는 것으로, **이 값이 최소화될 때 모델의 성능이 최적화**됨

   b. **손실 함수** (Loss Function)

      모델의 단일 데이터 포인트에 대한 **예측이 얼마나 잘못되었는지**를 측정(개별 데이터 포인트에 대한 모델의 오류를 측정)

      손실 함수의 값이 낮을수록 모델의 예측이 실제 값에 가까워진다는 것을 의미

   c.

      1. **회귀 모델**에서의 **MSE** (Mean Squared Error)
         - 손실 함수 : 단일 주택 가격 예측 모델에서, **MSE**는 **개별 주택의 예측 가격과 실제 가격 사이의 오차**를 제곱한 값을 계산 (해당 주택에 대한 모델의 성능)
         - 비용 함수 : 같은 모델에서, 모든 주택 데이터에 대한 **MSE의 평균**을 계산하여, **모델의 전체적인 성능**을 평가 (모든 주택 가격 예측에 대한 평균 오차)
      2. **분류 모델**에서의 **교차 엔트로피** (Cross-Entropy)
         - 손실 함수 : 이진 분류 모델에서, **교차 엔트로피**는 **개별 데이터 포인트의 예측이 얼마나 정확한 지**를 측정 (이메일이 스팸인지 아닌지를 예측할 때, 실제 레이블과 예측 레이블 사이의 차이를 측정)
         - 비용 함수 : 전체 이메일 데이터셋에 대한 **교차 엔트로피의 평균**을 계산하여, 모델이 **스팸을 얼마나 잘 구별하는지 전반적인 성능**을 평가
      3. **딥러닝 모델**에서의 **소프트맥스 손실** (Softmax Loss)
         - 손실 함수 : 다중 클래스 분류에서, **소프트맥스 손실**은 특정 이미지(예: 고양이, 개, 새 등)에 대한 **모델의 예측이 실제 레이블과 얼마나 일치하는지** 측정
         - 비용 함수 : 전체 이미지 데이터셋에 대한 **소프트맥스 손실의 평균**을 계산하여, 모델이 **다양한 클래스를 얼마나 잘 구별하는지** 측정

3. *과적합 (방지 방법)*
    a. **과적합** (overfitting) : 신경망이 훈련 데이터에만 지나치게 적응되어, 그 외의 데이터에는 제대로 대응하지 못하는 상태
    b. 방지 방법
        1. **데이터의 양을 늘리기**

            데이터의 양이 적을 경우, 해당 데이터의 특정 패턴이나 노이즈까지 쉽게 암기하기 되므로 과적합 현상이 발생할 확률이 늘어남

            → 데이터의 양을 늘릴수록 모델은 데이터의 일반적인 패턴을 학습하여 과적합 현상 방지
        2. **모델의 복잡도 줄이기**

            인공 신경망의 복잡도는 은닉층의 수나 매개변수의 수 등으로 결정

            과적합 현상이 포착되었을 경우, 인공 신경망의 복잡도를 줄이는 것
        3. **가중치 규제** (Regularization) 적용

            복잡한 모델이 간단한 모델(적은 수의 매개변수를 가진 모델)보다 과적합될 가능성이 높음

            → 복잡한 모델을 좀 더 간단하게 하는 방법, 가중치 규제
        4. **드롭아웃** (Dropout)

            학습 과정에서 신경망의 일부를 사용하지 않는 방법

            드롭아웃의 비율을 0.5로 한다면 학습 과정마다 랜덤으로 절반의 뉴런을 사용하지 않고, 절반의 뉴런만을 사용
        5. **조기학습 종료** (Early stopping)

            일정 횟수 이상 validation loss가 증가하는 시점부터 overfitting이 발생했다고 판단하여, 학습을 종료
        6. **배치 정규화** (Batch Normalization)

            데이터 분포를 통일 시켜줘서 과적합 방지

4. *Keras 구성요소, 문법, 특징*
    a. 구성요소 및 문법

        모델 구조

1. **Sequential API** : 레이어를 순차적으로 쌓는 방식으로 간단한 모델 구축에 적합 (모델이 단순하고 순차적인 구조가 필요한 경우)
2. **Functional API** : 복잡한 네트워크(다중 입력/출력, 잔차 연결 등)를 정의할 수 있는 방식 (복잡한 구조(병렬/스킵, 연결, 다중 입력/출력)가 필요한 경우)
3. **Subclassing API** : 고도로 사용자 정의된 모델을 구현하기 위한 객체지향 방식

레이어

Keras는 여러 종류의 레이어를 제공, 레이어는 모델의 기본 구성 요소로, 데이터를 처리하는 주요 연산 단위

- Dense (Fully Connected Layer)
- Conv2D (Convolution Layer)
- LSTM, GRU (Recurrent Layers)
- Dropout (Regularization Layer)
- Embedding (Word Embedding Layer)

컴파일

모델 학습 전에 **손실 함수**, **최적화 알고리즘**, **평가지표**를 설정하는 과정

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

Optimizer(최적화함수) : 모델의 가중치 설정(손실 함수 최소화 역할), 주로 경사하강법에 기반한 알고리즘

Loss(손실함수) : 모델 학습을 위한 **가중치 최적화 목표**, 모델이 얼마나 잘못된 예측을 하는지 측정

Metrics(평가지표) : 학습 후 모델의 성능 측정

학습 및 예측, 평가

- **fit()** : 모델 학습

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

주요 파라미터

1. **epochs** : 전체 데이터셋을 몇 번 반복해서 학습할지를 결정
2. **batch_size** : 몇 개의 샘플 단위로 나눠서 학습할 지 설정, 미니배치 경사하강법
3. **callbacks** : 학습 과정 중 특정 조건에서 실행될 작업(콜백 함수)를 설정

4. **validation_data** : 별도의 검증 데이터셋이 있을때
5. **validation_split** : 별도의 검증 데이터셋이 없지만 검증하고 싶을때
6. **shuffle** : 학습 데이터의 순서를 랜덤으로 섞을지 여부(시계열의 경우 섞지 말아야 한다)

- **evaluate()** : 모델 평가

  `loss, accuracy = model.evaluate(X_test, y_test)`

- **predict()** : 새로운 데이터에 대한 예측

  `prediction = model.predict(new_data)`

b. 특징

1. **사용자 친근성** (User Friendliness) Keras는 기계가 아닌 인간을 위해 설계된 API(Application Programming Interface, 응용 프로그램 프로그래밍 인터페이스)입니다. Keras는 일관성 있고 간단한 API를 제공하고, 일반적으로 필요한 사용자 작업의 수를 최소화하며, 사용자 오류 시 명확하고 실용적인 피드백을 제공합니다.
2. **모듈화** (Modularity) 모델은 가능한 최소한으로의 제한하여 독립적으로 생성 가능 합니다. 또한, 신경층(Neural Layers), 비용함수(Cost Functions), 최적화기(Optimizers), 초기화 스킴(Initialization Schemes), 활성 함수(Activation Functions), 정규화 스킴(Regularization Schemes)은 모두 새로운 모델을 생성하기 위해 결합할 수있는 독립 실행형 모듈입니다.
3. **쉬운 확장성** (Easy Extensibility) 새로운 모듈은 (새로운 클래스와 함수로서) 추가하기 쉽습니다. 새로운 모듈을 쉽게 만들 수 있기 때문에 Keras가 고급 연구에 적합합니다.
4. **Python 작업** (Work with Python) 모델은 작고, 디버그하기 쉽고, 확장성이 용이한 Python을 기반으로 두고 있습니다.

   그러나 코드에 오류가 발생하였을 경우 케라스 자체 에러일 수도 있고 사용한 백엔드(텐서플로우 등) 문제일 수도 있기에 원인 찾기가 쉽지 않습니다. 또한 문서화가 제대로 되어 있지 않습니다. 제공되는 함수 목록은 빨리 업데이트 되는 편이지만 함수에 대한 설명을 찾기가란 쉽지 않습니다.

   직관성이 떨어지고 진입 장벽이 높은 Tensorflow와는 달리 일반 Python API 형태로 제공하다보니 사용하기가 훨씬 쉽고 사용자 친화적인 환경을 제공하고 있습니다. 향후 Python 표준 딥러닝 API 형태로 자리 잡을 가능성이 있습니다.

5. *데이터 전처리 (목적성, 절차 및 방법, 기초 문법)*

   a. 목적성

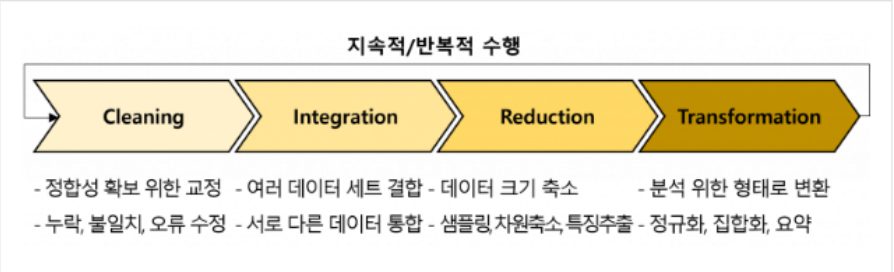
      아무리 좋은 도구나 분석 기법도 **품질이 낮은 데이터로는 좋은 결과를 얻기가 힘듦**

      → **데이터 전처리**를 통해, 원시 데이터로부터 유용한 정보를 추출하고, 머신러닝 모델이 이를 효과적으로 학습하고 일반화할 수 있도록 **데이터의 품질을 향상시키는 것**

      데이처 전처리, **모델의 성능에 직접적인 영향을 미치는 핵심적인 단계**로, 높은 품질의 데이터와 효과적인 전처리는 모델이 실제 세계에서 효과적으로 작동하고 일반화되도록 보장

   b. 절차 및 방법

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



1. **데이터 정제** (Data Cleaning)
   - 잡음, 부적합, 결측치를 수정하고 읽을 수 없는 요소를 제거
   - 형식의 일관성을 유지하고 적합한 포맷으로 변환함
2. **데이터 통합** (Data Integration)
   - 다양한 소스에서 얻은 데이터를 단일 형식으로 변경한 후 동일한 단위나 좌표로 변환함
   - 이름 의미의 일관성을 유지하고 메타 데이터 또한 유지함
3. **데이터 축소** (Data Reduction)
   - 일반적인 데이터는 크기가 너무 커서 한번에 분석이 어렵기 때문에 축소시킨 데이터를 사용하는 것이 효과적 (샘플링, 차원축소, 특징 선택, 이산화)
4. **데이터 변환** (Data Transformation)
   - 데이터를 변환하여 새로운 정보를 추출하거나 모델이 더 잘 이해할 수 있도록 함 (텍스트 데이터의 토큰화 또는 이미지 데이터의 전처리가 여기에 해당)

- 정규화, 집계 등을 통해 데이터 형식이나 구조를 변환함

→ 원본 데이터에서 유용한 특성을 추출하거나 데이터를 적절한 형태로 변환하여 모델의 성능을 향상

1. **결측치** (Missing Data)
2. **중복된 데이터**
3. **이상치** (Outlier)
4. **정규화** (Normalization)
5. **원-핫 인코딩** (One-Hot Encoding)
6. **구간화** (Binning)

6. *주피터 (or 코랩) 사용법*

코랩

파일→새노트를 통해 코드 작성할 노트 생성

생성을 눌러 원하는 값에 대한 내용을 적으면 AI가 자동으로 코드를 작성해줌

코드를 작성했다면, 코드 셀 왼쪽에 재생 버튼 클릭 or Ctrl + Enter

다음 코드를 작성하길 원하면, 상단에 +코드/ +텍스트를 선택하여 새로운 코드 및 텍스트를 작성할 수 있음

코드 실행시키고 바로 새로운 셀을 추가하는 방법 Alt +Enter

자주 사용하는 단축키

본인이 원하는 키 설정은 Ctrl + M + H를 통해 맞춤 설정할 수 있습니다.

1.1 실행 단축키

1. Ctrl(Command) + Enter : 해당 셀 실행
2. Shift + Enter : 해당 셀 실행 + 커서를 다음 셀로 이동
3. Alt(Option) + Enter : 해당 셀 실행 + 코드 블록 하단 추가

1.2 셀 삽입/삭제 단축키

1. Ctrl(Command) + M A : 코드 셀 위에 삽입
2. Ctrl(Command) + M B : 코드 셀 아래 삽입
3. Ctrl(Command) + M D : 셀 지우기
4. Ctrl(Command) + M Z : 실행 취소

1.3 수정 단축키

1. Ctrl(Command) + Alt(Option) + 화살표위아래 : 동시 수정
2. Ctrl(Command) + D : 같은 단어 찾아 동시 수정(각각 검색)
3. Ctrl(Command) + Shift + L : 동일 단어를 전체로 찾아 동시 수정
4. Ctrl(Command) + Alt(Option) + 화살표위아래 : 위아래 동시 수정
5. Ctrl + / : 주석
6. Shift + Del : 한 줄 지우기