[2주차]

<데이터분석과정>
1. 문제 정의
2. “데이터 준비” 수집, 준비 (web scraping)
3. 전처리 과정 (raw한 형태의 데이터를 분석하기 좋은 형태로 바꾸기)
4. 분석 (텍스트, 오디오, 이미지 등)
-> 여러 가지 사용 가능한 분석 방법 중 우리가 풀고자 하는 문제의 답을 찾는 데 더 적합하게 사용될 수 있는 방법 선택 (데이터의 특성을 잘 파악)

1~4를 배움

인터넷상에 있는 거의 모든 데이터는 다운로드 가능함
웹페이지에 접속하려면 웹브라우저를 사용해야 함

<브라우저의 기능과 역할>
ex) 크롬, 익스플로러

* 주소창에 주소를 입력하고 웹페이지가 화면에 뜰 때까지의 과정
1) 이동하고자 하는 웹페이지의 URL 주소를 입력
2) 이 URL 주소를 IP 주소로 바꿈 (인터넷 프로토콜 주소 -->165.132.172.31 ....)
IP주소: 컴퓨터를 위한 주소 (URL은 사람을 위한 주소)
DNS(도메인 네임 서버)가 URL을 IP 주소로 바꿈
웹페이지가 갖고 있는 데이터를 저장하고 있는 컴퓨터(서버)의 주소임 = IP 주소
내 컴퓨터는 client
3) 브라우저가 이 주소에 해당하는 서버컴퓨터에 접속함 (인터넷을 사용하여)
client가 서버 컴퓨터에 request messsage (요청 메시지)를 보냄
내 사용자가 이 웹페이지에 들어가려고 하니까 이 웹페이지의 정보를 담은 데이터를 내게 보내줘
서버 컴퓨터가 안전한지 문제 없는지 등을 체크하고 응답메시지를 보냄 (response message)
응답메시지에는 서버와 클라이언트가 통신하는 데 필요한 여러 정보와 웹페이지의 원본데이터가 들어가 있음
그러면 브라우저가 원본데이터를 보기 편하게 디스플레이 함

● 파이썬을 이용해서 직접적으로 특정 웹페이지의 URL주소를 이용하여 request 메시지를 보내고 원본데이터를 다운로드

● 원본데이터 보기 (우클릭, 페이지 소스 보기 / ctrl + u)
--> 이 원본데이터를 “소스코드”라고 함 (source code)

● 웹스크래핑을 하려면 1차적으로 웹페이지의 원본데이터의 소스코드를 다운
-> 웹페이지의 소스코드들은 Tag들로 구성되어 있음
시작 태그
끝태그
<title>언젠가 우리가 같은 별을 바라본다면 - 예스24 </title>
앞 title은 시작 태그
뒤 title은 끝 태그 (앞에 /)
그 사이에는 일반적으로 text 정보가 저장되어있음
("언젠가 우리가 같은 별을 바라보다면 - 예스24")
이 텍스트 정보가 우리가 웹페이지에서 확인할 수 있는 정보

정보를 수집할 때는 일부의 정보만 수집함
(문제와 관련 있는 정보만 수집)

<웹스크래핑> - 파이썬을 활용하여 직접적으로 함
1. 웹페이지의 소스코드 다운 (원본데이터)
=> requests라고 하는 library를 활용 (서버에 요청메시지를 전송하기 때문)
-> get() 함수 활용: 서버에 request 메시지 보냄 (서버는 client에 response메시지를 보내고, get이 받음)
-> BeautifulSoup이라는 class를 활용하여 필요한 텍스트 추출
-> bs4라고 하는 library에 저장
-> 각각의 태그에 정보가 있음
2. 여러 태그들 중 필요한 정보만 수집 (ex: 가격정보를 저장하고 있는 태그를 찾기)
3. 그 태그가 저장하고 있는 텍스트 추출

meta tag는 브라우저를 위한 정보임
메타 태그를 찾을 때는 name이라는 속성정보를 사용할 것

[4주차]

1. 문제 정의
2. 데이터 수집 (웹스크래핑)
3. 분석 (텍스트 / 이미지, 비디오) => 텍스트가 메인

● 자연어 처리
-> 자연어란? 컴퓨터 관점에서 사람이 사용하는 언어인 자연어를 이해하고 생성하는 것이 주목적
-> 사람이 어떤 텍스트를 입력하면 알아서 이해하게 하고 직접 생성하게 하는 것 (chat gpt가 하는 것)
-> 텍스트 분석과 자연어 처리의 공통점 (텍스트 분석 방법을 이용해서 자연어 처리에 적용 가능하고 그 반대도 가능함)

● 텍스트 데이터 수집 및 분석 단계
-> 어떠한 텍스트 데이터를 수집해야 하는가
-> 원하는 텍스트 선별 및 수집
-> 컴퓨터가 분석 가능한 형태로 가공 (텍스트 전처리)
-> 문서를 숫자 형태로 표현 (문서의 vectorization)
-> 우리가 원하는 결과, 정보 도출 (텍스트 분석)

● 텍스트 데이터
-> 텍스트들은 문서들(documents)로 이루어져 있음 / ex) 여러 개의 신문기사
-> 문서: 텍스트 데이터를 구성하는, 분석의 기본 단위 (리뷰, 영화평, 기사 등등)

● 텍스트 데이터 분석 방법
1) 기계 학습 알고리즘 기반 분석 방법 (숫자 데이터에만 적용됨)
-> 1. 군집화 (같은 군집에는 유사한 특성을 가진 문서들이 속하도록 함)
-> 2. 문서분류 (문서 특성에 따라 문서를 분류함. 대표적으로 감성분석이 있음)
: 여기서 감성이란 그 문서가 갖고 있는 특정 태도를 말함
(긍정의 영화평인지, 부정의 영화평인지)
-> 3. 토픽 모델링
: 각각의 문서의 주제를 찾을 때 사용하는 방법
2) No 기계 학습 알고리즘 기반 분석 방법
-> 1. 빈도분석 (단어들 사용 빈도 분석)
: 주제와 관련이 높은 단어, 즉 키워드가 무엇인지 찾고자 할 때 (주로 명사 단어)
-> 2, 텍스트 네트워크 분석
: 텍스트 데이터에서 사용되는 단어들 간의 연결관계를 연구할 때
(내가 관심 있어 하는 주제 중 특정 기사가 어떤 이슈에 포커스를 많이 했는지 볼 수 있음)

● 문서의 vectiorization
-> 기계학습 알고리즘은 숫자 데이터에만 적용 가능
-> 텍스트 데이터를 전처리 한 후 그 데이터를 숫자형으로 변환해야 함
-> 이 데이터를 숫자로 변환하는 것
(기계학습 기반 분석 방법인 경우에만 이 과정을 거치면 됨)

<No 기계학습 기반 분석방법>
● 영어 텍스트 데이터 전처리
● 한글 텍스트 데이터 전처리

[5주차]

1. 문제 정의
2. 데이터 준비
3. 전처리
4. 문서의 벡터화
5. 분석
1) 머신러닝 기반 방법
2) 머신러닝 기반 방법이 아님 (빈도분석 or Text Newtork Analysis)
-> 2) 경우는 문서의 벡터화 필요 없음

● 텍스트 전처리 과정
-> 전처리: raw한 데이터를 분석하기에 좋은 형태로 바꾸는 것
-> 불용어가 제거된 특정한 품사들의 단어들
(의미 없는 단어들 -> 제거하지 않으면 분석 결과의 정확도가 떨어지게 됨)
-> 최종적으로 선택되는 단어들은 해당 문서의 특성을 잘 나타내야 함
-> 최종적으로 선택되는 단어들은 분석 목적에 따라 달라짐
1) 주제와 관련된 분석: 명사
2) 감성분석: 형용사, 부사 포함 필요

● 영어 전처리 과정
1. 불필요한 기호 삭제 (예: !,.“:; 등)
2. 대소문자 통일 (case conversion, 소문자 <-> 대문자)_보통 대문자를 소문자로 많이 바꿈
3. 단어 단위로 쪼개기 (Tokenization)
-> 토큰(단어): 뜻을 갖고 사용될 수 있는 글의 단위 (텍스트를 구성하는 기본단위로, 영어에서는 일반적으로 단어가 하나의 토큰임)
ex) 단어, 문장
-> 한글에서는 형태소가 텍스트의 기본 구성 단위임
4. 단어의 품사 찾기 (POS of tagging: part of speech tagging)
5. 원하는 품사의 단어들만 선택 (명사 or 형용사, 부사)
6. 단어의 형태 통일
-> 단어의 원형 or 어근 찾기 (원형: lemma, 어근: stem)
-> 원형찾기: lemmatization / 어근찾기: steming
-> 어근찾기는 잘 안하고 원형찾기를 주로 함
7. 불용어 (Stopwords) 제거
-> 우리가 수행하는 분석에 있어서 별 의미가 없는 단어 or 형태소
-> 정확한 결과를 얻기 위해서는 불용어를 잘 제거해야 함

● 불용어
1. 분석의 목적과 상관없이 별 의미가 없는 단어들
▶ a, an, this, that, the 등
2. 분석의 목적에 따라 의미 여부 결정 (두 단계의 단어들 모두 제거해야 함)
▶ 신문사의 이름이 중요할 수도 있고, 아닐 수도 있음

● 기호제거
1. replace('!', '')
2. 정규표현식 (regular expression)
=> 문자열 값에 대한 패턴
=> 여기서는 기호가 문자열값임. 기호를 찾기 위해서 정규표현식을 만들 것인데, 기호에 대한 패턴을 만들면 됨. (삭제하거나 원하는 문자로 바꿈)
▶ 패턴: [^\w\d\s]: 하나의 캐릭터를 나타냄.
▶ ^는 부정의 의미를 가짐 (뒤에 나오는 애들)
▶ \w는 하나의 word character를 나타냄 (word)
▶ \d는 하나의 digit character를 나타냄 (숫자)
▶ \s는 하나의 space를 나타냄 (공백 문자: 띄어쓰기, 탭, 줄바꿈)
▶ 앞에 부정을 나타내는 ^이 붙었으므로 w도 아니고 d도 아니고 s도 아닌, 기호를 나타냄

● 한글 전처리 과정
▶ 한글에서 token은 형태소임. 텍스트의 기본단위가 형태소이기 때문임
▶ toenization의 결과로 형태소가 나옴
1. 불필요한 기호 삭제
2. Tokenization(형태소 분석)
: 형태소 추출(tokenization) -> 원형 찾기 -> POS tagging (한 번에 함)
```
from kiwipiepy import kiwi
kiwi = kiwi()
kiwi.tokenize(filtered_content, stopwords = stopwords)
```
3. 필요한 품사(형태소)의 단어들만 선택
4. 불용어 제거

● 형태소 분석기
▶ kiwi
▶ kkma, komoran, mecab
▶ kkma 등은 KoNLPy 라이브러리 안에 있는데, 설치가 어려움. 그래서 kiwi 라이브러리가 나음

● 미등록 단어 문제

▶ 형태소 분석기는 기본적으로 형태소 사전을 활용함. 근데 정보 저장이 되어 있지 않은 일부 단어 or 형태소가 있음. 대표적으로 신조어가 있음. 형태소 사전에 해당 단어나 형태소가 등록되어 있지 않으면 당연히 파악이 불가능함 (미등록 단어 문제)

▶ OOV_example.ipynb 파일 참고

▶ 이를 위해서는 형태소 사전에 특정한 품사로 새로운 단어를 저장(추가)해야 함

[6주차]

● 전처리
1. 영어 (7단계)
1) 기호 없애기
2) 대소문자 통일
3) 단어로 구분하여 토큰화
4) 품사 찾기
5) 특정 품사의 단어들만 선택
6) 원형을 이용한 단어 통일
7) 불용어 제거

2. 한글 (4단계)
1) 기호 없애기
2) 형태소 분석
3) 특정 품사의 단어들만 선택
4) 불용어 제거

● 형태소 분석
▶ kiwi 형태소 분석기 사용
(형태소 단위로 쪼개기 + 형태소의 품사 찾기(pos tagging) + 원형찾기)
▶ 형태소분석을 할 때 kiwi에서 제공되는 tokenize 함수를 사용해서 불용어 제거를 하기 때문에 따로 할 필요 x

● 빈도분석: 주제와 관련된 키워드를 찾음 (주제에 대해 파악 가능)
▶ Counter 클래스를 사용, 전처리로 사용한 특정 품사의 단어들을 입력
▶ 빈도분석을 시각화하기 위해 wordcloud 모듈 사용

● 이미지 픽셀
▶ 이미지는 여러 픽셀이 있고, 색상정보를 가짐. 각각의 색상 정보는 0~255로 표현함.

● 워드클라우드
: 빈도분석을 시각화한 결과
▶ 설치
pip install wordcloud

● 텍스트 데이터에 네트워크 분석을 적용한다는 것은
▶ 텍스트 데이터를 구성하고 있는 단어들로 구성된 네트워크
▶ 해당 문서를 구성하고 있는 단어들의 네트워크 (노드)
▶ 단어들간의 관계정보 파악
▶ 특정 단어가 분석하고자 하는 텍스트 문서에서 어떤 단어들과 얼마만큼 많이 같이 사용되었는지
▶ 특정 키워드들과 같이 더 많이 사용된 단어들
▶ 해당 주제와 관련한 어떤 단어들이 해당 문서에서 더 빈번히 사용되었는지, 어떤 주제에 대해서 더 다루고 있는지 파악함
▶ 북한 이슈에 대해 어떤 신문사는 미사일을, 어떤 신문사는 원조를 더 많이 함께 언급
▶ 정치 커뮤니케이션, 저널리즘 분야에서 많이 함

● 네트워크
▶ 여러 개의 노드들로 구성되어 있고, 연결정보까지 나와 있는 것을 네트워크라고 표현함
▶ 네트워크는 그래프라는 말로 표현되기도 함 (파이썬에서는 그래프라고 표현함)
▶ 연결은 tie라고 부르며, edge라고 부르기도 함
▶ 연결의 의미는 상황에 따라 다름 (사람으로 따지면 친구관계일수도, 직장동료일수도)

● NetworkX를 사용한 Text Network Analysis
▶ 파이썬에서 사용하는 모듈임
1. 네트워크 데이터를 준비하기 (노드와 연결관계에 대한 정보임)
2. NetworkX를 사용하여 우리가 분석하고자 하는 네트워크를 파이썬에 생성해야 함
3. 분석: 어떤 노드가 어떤 노드와 연결되는지

● 네트워크 생성
1. 빈 껍데기만 있는 네트워크
2. 노드 생성
3. 엣지(타이) 생성

● 해당 주제에 대해 어떤 이슈와 측면에서 어느 정도 이야기하고 있는지 알 수 있음
▶ 단어들로 구성되어있는 네트워크

● 노드의 속성정보
1. 명목변수: 해당 변수가 취할 수 있는 값이 몇 개로 한정되어 있거나 각각의 값들이 어떤 특정한 그룹의 이름을 나타내는 변수 (ex: 성별) > 파티션 사용 (노드색깔 변경)
2. 연속변수: 나이, 몸무게 등 > 랭킹

▶ 기본적으로 degree (직접 연결되어있는 다른 노드의 수)라는 속성이 부여되어있음
▶ 디그리가 큰 노드가 당연히 영향력이 큼 (인플루언서)
▶

[7주차]

● NetworkX
1. 네트워크 데이터 준비
2. 네트워크 생성 (파이썬상에)
3. 분석

● 네트워크
1. 노드
2. 엣지

● Gephi에서 변수별 속성
1. 명목 변수 (범주형 변수): 파티션 사용
2. 연속형 변수: 랭킹 사용

● degree
: 직접적으로 연결된 노드의 수
-> 영향력. 중심. 인플루언서 (오피니언 리더)

● 텍스트 데이터를 네트워크로 표현
1. 노드: 단어가 하나의 노드가 됨

● 네트워크 생성의 순서
1. 노드로 표현하고자 하는 단어들을 선택
▶ 어떠한 단어들을 선택해야 하는가?
▶ 단어 선택 방법
    1) 이론적 근거 - 주요 이슈, 단어 선택 (주요하게 사용되는 방법)
    2) 빈도분석 - 빈도를 기준으로 상위 몇 개를 노드로 사용 (좋은 방법은 x)
2. 노드(단어)간 타이 생성
▶ 타이 존재의 기준은 무엇인가?
(두 단어가 함께 사용되었다는 뜻. 근데 같이 사용되었다는 근거는?)
    1) 문장: 한 문장 안에서 두 단어가 함께 쓰인 경우
        -> 하나의 문서를 바탕으로 분석할 때
    2) 문서: 한 문서 안에서 두 단어가 함께 쓰인 경우
        -> 여러 문서를 바탕으로 분석할 때

● 하나의 문서를 네트워크로 표현하고자 하는 경우

1. 비어있는 네트워크 만들기
2. 노드(단어) 추가 -> 단어를 선택하여 (선행연구, 빈도분석을 바탕으로)
3. 문장을 기준으로 타이 추가

-> 문장 기준으로 쪼개고, 서로 다른 단어가 같은 문장에서 얼마나 함께 사용되었는지 등을 파악

-> 단어들로 구성되어있는 네트워크에 존재하는 타이도 연결강도가 있음 (이는 함께 사용된 문장의 수임)

-> 연결강도가 셀수록 두 단어의 관계가 더 긴밀하다는 것

● 디그리 중심도

: 디그리가 클수록 중심적 역할

▶ 정의: 노드_i의 디그리 중심도 (n = 네트워크에 포함된 노드 수)

= 디그리_i / 네트워크에서 디그리가 가질 수 있는 최대 디그리 수 (n-1)

= 디그리_i / n-1

$$d_i = {degree_i}\Big/{maximum\_degree_g}$$

- 노드 i의 디그리 중심도(d_i)는 네트워크에서 갖는 노드 i의 디그리($degree_i$)를 해당 네트워크에서 한 노드가 가질 수 있는 최대의 디그리($maximum\_degree_g$)로 나눈 것.
- $maximum_degree_g$는 노드의 수가 n개인 네트워크인 경우 n-1이 됨.

● 매개 중심도

: 한 노드가 해당 네트워크에서 얼마나 많은 다리 역할, 즉 매개자 역할을 하는지 보는 것

$$b_i = \sum_{s,t \in V} \frac{\sigma(i)_{s,t}}{\sigma_{s,t}}$$

- V는 전체 노드의 집합.
- 분모의 $\sigma_{s,t}$는 노드 s와 노드 t 사이의 가장 짧은 패스의 수
- 분자의 $\sigma(i)_{s,t}$는 그러한 패스 중에서 노드 i를 거치는 패스의 수

● 근접 중심도

: 특정 노드가 네트워크에 있는 다른 노드들과 얼마나 가깝게 연결되어 있는지를 의미

$$c_i = \frac{n-1}{\sum_{j=1}^{n-1} dis(i,j)}$$

- n = 네트워크에 속한 전체 노드의 수
- $dis(i,j)$ = 노드 i와 노드 j 간의 가장 짧은 패스의 거리

● 분석하고자 하는 (네트워킹하고자 하는) 문서를 문장 단위로 쪼개고 분석했음

● 이번에는 하나의 문서가 아닌 여러 개의 문서로 구성되어 있다고 가정
-> 엣지 기준이 문서가 됨

[9주차]

1. 문제 정의
2. 데이터 수집
3. 전처리
4. 문서의 벡터화 (머신러닝 기반 분석만)
1) 머신러닝 x
▶ 빈도분석, 텍스트 네트워크 분석
2) 머신러닝 o
▶ …

<기계학습 알고리즘 기반 분석 (머신러닝 기반 분석)>
▶ 특히 딥러닝 알고리즘이 핵심적인 역할
전통적인 기계학습 기술들은 요즘 인공지능을 구현하는 데 크게 쓰이진 않고, 딥러닝 알고리즘이 더 많이 쓰임
▶ 기계학습이란?: 컴퓨터가 학습한다 (기계가 학습)
=> 이를 위해서는 데이터가 꼭 있어야 함 (다양한 정보를 가짐)
=> 데이터에 존재하는 다양한 정보들 중에서 풀고자 하는 문제들 중에서 중요한 정보를 추출하는 것이 학습임
=> 이 작업을 기계를 이용해서 하면 기계학습임
=>> 즉 기계를 이용해서 풀고자 하는 문제의 중요 정보를 추출해서 푸는 것임

▶ 기계가 학습을 할 때, 정보를 추출할 때 쓰는 툴이 “기계학습 알고리즘”
=> 간단하게 표현하면 “수학적 모형(수학적 함수)”
=> 딥러닝 알고리즘도 수학적인 함수인 (마치 일차함수, 물론 훨씬 더 복잡)
=> 데이터에 존재하는 어떠한 정보를 추출함
=> X와 Y의 관계를 파악하는 것 등
ex) X: 직장 경력 // Y: 연봉
-> 이 문제를 잘 풀려면 정답(연봉)과 힌트(직장경력) 모두 들어있는 정보, 즉 학습 데이터를 적용해서 학습시키는 게 좋음. 그리고 문제는 직장 경력만 가지고 연봉을 예측하는 것.

▶ 기계학습 알고리즘 종류
1. 지도 학습 알고리즘 (70% 정도)
=> 학습 데이터에 **힌트 정보와 정답 정보** 포함
=> 알고리즘을 이용하여 정답과 힌트 간의 관계 파악
=> 정답이 없는 새로운 데이터에 적용
ex) 문서분류 (ex: 감성분석)

2. 비지도 학습 알고리즘 (25% 정도)
=> 관측치들의 **특성 정보**를 담고 있는 데이터를 사용 (힌트 정보와 정답 정보 구분 x)
=> 관측치들의 특성을 파악 또는 데이터에 존재하는 패턴을 찾을 때
ex) 군집화 분석, 차원 축소

3. 강화 학습 알고리즘 (5% 정도)
=> 컴퓨터가 주어진 문제를 해결하기 위해서 일련의 행동들을 (여러 번) 하는데, 각 행동을 수행한 결과에 따라 상 또는 벌을 내려 최종적으로 더 바람직한 결과를 유도하는 방식
=> 아직은 많이 사용되지 않음
=> 컴퓨터 게임 분야에서 많이 사용됨
ex) 알파고
=> 최근에는 chat GPT와 같은 생성형 AI를 구현하는 데에도 사용되고, 요즘 증가하는 추세

▶ 벡터의 기하학적인 의미 (중요)
=> 하나의 점(좌표)임
=> 차원이 중요함 (원소 개수로)
=> 벡터는 공간상의 점으로서 고유한 위치정보를 가짐
=> 벡터의 위치정보는 벡터가 가지는 원소의 값에 의해 결정됨
==> **(즉 벡터의 위치정보는 벡터의 고유한 특성을 나타냄)**
=> 두 벡터의 위치가 가까울수록 두 벡터의 원소값이 유사하다는 것을 알 수 있음
(두 벡터의 고유 특성이 유사하다는 뜻)

● 코사인 유사도
▶ 벡터간의 유사도를 계싼하는 가장 일반적인 방법
1) 유클리디안 거리
2) 코사인 유사도 방법
-> 방향이 유사할수록 (사이각이 작을수록) 유사
(즉 코사인값이 클수록 유사함)

● 문서 -> vectors
-> 단어 정보, 저자 정보, 신문사 정보, 날짜 정보들 중에 어떤 정보를 사용하여 벡터화
(보통 단어 정보를 사용 = Bag of words model)

● Bag of words model (BoW)
▶ 문서에서 단어의 빈도수 정보를 이용하여 문서를 벡터화하고 이때 사용된 각각의 단어를 feature라고 부름 (물론 전처리를 통해 얻은, 불용어가 제거된 특정한 품사의 단어들임)
▶ corpus에 포함된 문서들의 벡터 크기는 같음

● Document term matrix (DTM)
▶ 원소값: binary (단어의 출현 여부) / frequency / tf-idf
▶ 벡터 간의 거리를 계산하여 유클리디안 거리 or 코사인 유사도를 사용하여 문서간의 유사도를 계산

● TF - IDF (term frequency - inverse document frequency)
▶ 단어들의 빈도 정보 사용 -> 단어의 상대적 중요성을 알 수 없음 (특정 문서의 특성을 더 잘 나타내는 단어)
▶ 단어의 상대적 중요성을 수치로 표현
▶ 즉 각 단어들이 corpus에 존재하는 다른 문서들에서 얼마나 사용됐는지를 수치로 표현해야 함 (DF: 특정 단어가 사용된 다른 문서의 수)
▶ 해당 문서에 있는 특정 단어가 다른 문서에서도 많이 사용된다면, 즉 특정 단어가 사용된 다른 문서의 수가 많다면 해당 문서에서 그 단어의 상대적 중요성은 적어짐
=> 이렇게 되면 DF와 중요성이 반비례 관계를 가지므로, 쉽게 보기 위해 1을 더하고 역수를 취해준 것이 IDF (DF가 0이면 1을 더해서 역수 취함)
=> IDF = 1/(DF+1)
▶ IDF 값이 크면 클수록 상대적 중요성이 크다는 뜻, 즉 다른 문서에서 사용되지 않았다는 뜻
▶ TF-IDF = TF (단어의 빈도수) * IDF

● sklearn을 이용한 문서 벡터화
▶ TF: CountVertorizer 클래스 사용
▶ TF-IDF: TfidVectorizer 클래스 사용

[10주차]

● 문서의 벡터화
▶ 각 문서를 문서에 사용된 단어들로 구성이 된 vector로 만들어야 함
▶ 같은 corpus에 포함된 문서들의 vector 크기는 같음
▶ document term matrix

● 빈도 정보 기반 방법의 단점
▶ 특정 단어가 특정 문서의 고유한 특징 (중요도)를 잘 반영하지 못함

● TF-IDF (다른 문서에서 적게 사용되고 / 그 문서에서 특정 단어가 많이 사용될수록 커짐)
▶ DF: 특정 단어가 사용이 된 문서의 수
▶ IDF: DF의 역수 (얘가 클수록 다른 문서에서 사용되지 않은 것, 사용된 문서에서 중요도가 올라간다는 것)
▶ TF: 특정 문서에서 해당 단어의 빈도수

● sklearn을 이용한 문서 벡터화
▶ TF: CountVectorizer 클래스 사용
▶ IDF: TfidfVectorizer 클래스 사용

● 군집화 (비지도 알고리즘)
▶ 유사한 관측치들끼리 묶어주는 방법
▶ 관측치들간의 유사도를 기반으로 유사한 관측치들을 같은 그룹으로 묶어줌
▶ 유사도가 상대적으로 큰 벡터들을 같은 군집에 넣음
▶ K-Means 알고리즘 사용

● K-Means
▶ Clustering 방식
▶ cluster의 수는 k라고 가정 (k: 찾고자 하는 군집의 수)
▶ K=3이라고 가정하면, 전체의 관측치들 중에서 임의로 3개의 점을 선택
-> 이 3개의 점이 각 그룹의 중심이 됨
-> 그 다음 중심이 아닌 점들의 군집을 결정함 (한 점과 중심들 간의 유클리디안 거리를 계산하여 가장 가까운 중심의 군집에 넣는 것임)
▶ 그 후 각 군집에 대해 새로운 중심을 다시 정함 (각 군집에 소속된 점들의 평균으로 정함)
-> 새로운 중심으로 원래 있던 벡터가 아닐 확률이 큼

▶ 이전 단계에서 했던 것처럼 다시 중심과 점들간의 거리를 구하여 유사도를 구함
-> 가장 가까이 존재하는 중심이 달라지는 벡터가 생김
-> 재군집화

▶ 이를 반복하여 중심 (군집)이 수렴될 때까지

● 실루엣 스코어
▶ -1 ~ 1 사이의 값을 가짐
▶ 1에 가까울 정도로 잘 군집화 되었다는 것
▶ 이 지표를 통해 적절한 군집의 수, 즉 k값을 정할 수 있음
▶ 서로 다른 군집에 속한 벡터들 간 유사도가 작을수록, 같은 군집에 속한 벡터들 간 유사도가 클수록 실루엣 스코어 값이 커짐

● 같은 군집에는 일반적으로 비슷한 주제의 문서가 포함되게 됨

● 군집화 분석을 할 때는 빈도수 정보보다는 TF-IDF 정보를 활용하는 것이 옳음

● 차원을 축소할 필요가 있음 (벡터의 원소 수를 줄인다는 것)
- 관측치를 나타내는 벡터의 차원을 줄임
- 시각화가 어려움
- 차원축소를 해야 하는 이유
1) 보통 텍스트 분석의 벡터는 굉장히 많은 단어의 수를 가짐 (각 문서가 고차원 벡터로 변환)
-> 근데 대부분의 단어들은 문서의 특성을 구분하는 데 중요한 역할을 하지 못함
(이러한 단어의 정보가 담긴) 고차원의 벡터를 그대로 사용해 군집화하면 군집화 분석의 결과가 부정확해짐
-> 따라서 문서의 특성을 잘 나타내는 정보를 추출해서 차원 축소를 진행한 후 군집화 분석을 수행하는 것이 더 바람직할 수 있음

2) 시각화
-> 고차언 벡터가 되면 시각화가 불가능해짐
-> 2차원 or 3차원 공간에 시각화를 하기 위해서도 차원 축소 사용
-> pca라는 알고리즘 사용

● 지도학습 알고리즘 -> 문서분류
ex) 감성분석 (어떤 이슈나 주제에 대한 태도)
-> sentimental: 사실 attitude에 더 가까운 단어임
-> 긍/부정으로 나뉘는 경우가 많음

● 지도학습 알고리즘
▶ 기본적으로 서로다른 데이터 set 2개가 필요함
▶ 1) 학습 데이터 (정답과 힌트 정보가 모두 들어가 있음)
▶ 2) 문제 데이터 (only 힌트 정보만 있음)
ex) 직장경력을 통한 연봉 예측
hint: 직장경력
answer: 연봉

● 정답과 힌트 정보가 모두 존재하는 학습 데이터 사용
-> hint 정보: 독립변수 (X)
-> 정답: 종속변수 (Y)

● 학습데이터에 존재하는 X, Y간 관계를 알기 위해 지도학습 알고리즘을 사용함
-> 지도학습 알고리즘: “수학적 모형(수학적 함수)”
-> 이 함수를 파악하여 관측치에 대해 종속변수 값을 예측함
-> 학습 데이터에 지도학습 알고리즘 (수학적 모형)을 적용하여, 이 데이터에 존재하는 X와 Y의 관계를 파악함 (근데 이 X와 Y의 관계는 파라미터에 따라 달라짐)
-> 이를 통해 파라미터의 최적값을 찾아냄 (X와 Y의 관계를 가장 잘 설명할 수 있는 파라미터)
-> 이런 최적값을 찾는 문제가 바로 최적값 문제 (optimization value)
-> 이 함수를 다시 문제 데이터에 적용하여 정답을 찾아냄

● 그렇다면 파라미터의 최적값은 어떻게 찾는가
1) 학습 데이터를 설명하는 정도를 최대화하는 파라미터 값을 찾는 방법
2) 학습 데이터를 설명하지 못하는 정도를 최소화하는 파라미터 값을 찾는 방법
▶ 이 2가지는 동일한 문제임
▶ 기계학습에서는 두 번째를 사용함
▶ “비용함수(손실함수)”를 통해 ‘선택한 모형이 관계를 설명하지 못하는 정도’를 나타냄
▶ 비용함수가 커질수록 모양의 설명력이 떨어짐
▶ 즉 비용함수를 최소화하는 파라미터를 찾으면, 그것이 파라미터의 최적값임

[11주차]

● 지도학습 알고리즘 지난 수업내용 review
-> 학습데이터와 문제 데이터
-> 학습데이터: 힌트 + 정답 정보
-> 힌트와 정답 간의 관계 파악 -> 문제 풀이
-> 힌트와 정답 간의 관계 파악을 위해 지도학습 알고리즘 사용, 수학적 모형
-> 파라미터 (b_0, b_1 같은 것): 독립변수와 종속변수의 관계 정의
-> 학습: 지도학습 알고리즘에 존재하는 파라미터의 최적값
(최적값: 학습데이터를 가장 잘 설명할 수 있는 파라미터 값)
-> 파라미터의 최적값을 어떻게 찾을까: "비용함수(손실함수)" 사용
학습데이터를 설명하지 못하는 정도를 최소화하는 방법 채택
-> 비용함수: 데이터를 설명하지 못하는 정도를 나타냄

***정리: 비용함수(데이터를 설명하지 못하는 정도)를 통해 학습데이터에서 힌트와 정답 간의 관계를 가장 잘 나타내는 파라미터의 최적값을 찾음 (비용함수를 최소로 만드는 파라미터 채택)**
-> 비용함수는 파라미터의 함수임. 파라미터 값에 따라 값이 달라지므로, 비용함수 값을 최소로 만드는 파라미터를 잘 골라야 함

● 비용함수의 종류 (어떤 지도학습 알고리즘을 사용하는지는 크게 중요하지 않음)
-> 풀고자 하는 문제 종류에 따라 달라짐
-> 종속변수 종류에 따라 문제 종류도 달라짐 (연속형 변수 vs 범주형 변수)
-> 모형을 통한 예측이 잘못될수록 비용함수 값이 커짐
1) 회귀문제: 연속형 변수
▶ MSE (최소 제곱 오차: mean squared errors)
▶ 정규방정식: 접선의 기울기가 0이 되는 곳을 찾아서 최소 위치 찾기
▶ 경사하강법: 파라미터를 순차적으로 업데이트 하여 비용함수를 최소화하는 파라미터를 찾는 방식 => b_i,new = b_i,current - etha x dE/db_i(b_i,current)
--> 이거 점화식으로 만들어서 y=x와 만나는 점으로 파라미터가 수렴될 것임
▶ 여기서 etha는 학습률(learning rate, 0~1 사이)
2) 분류문제: 범주형 변수
▶ 교차 엔트로피 (이 식은 잘 몰라도 됨_ppt에 있음)

● 지도학습 적용 순서
1. 학습데이터와 문제 데이터 준비
2. 알고리즘 (모형) 선택
3. 학습: 파라미터의 최적값 도출
4. 평가 (평가에 사용되는 평가데이터가 필요함)
5. 문제에 대한 데이터에 적용

● 평가데이터의 조건
1. 힌트, 정답 정보 필요 (예측한 종속변수와 실제 종속변수 간의 차이를 통해 평가)
2. 학습에 사용된 데이터 x

● 필요한 데이터: 정답데이터 / 문제 데이터
1. 정답 데이터 -> 일부: 학습데이터 / 일부: 평가데이터
▶ 보통 7:3, 8:2 정도로 사용
2. 문제 데이터

● 평가 지표
▶ 풀고자 하는 문제의 종류에 따라 달라짐 (회귀문제 vs 분류문제)
1. 회귀문제: R^2 or MSE^1/2 (=RMSE)
-> R^2은 0~1 사이, 1에 가까울수록 설명력 좋음
2. 분류문제: 정확도(Accuracy), 재현율(Recall), 정밀도(Precision), F1
-> 정확도: 정확히 예측한 종속변수의 비율

● 과적합 문제 (지도학습 알고리즘의 가장 빈번하고 critical한 문제)
: 학습데이터는 잘 설명하는데, 새로운 데이터는 잘 설명하지 못함
-> 학습데이터에 너무 과하게 적합해서 그런 경우가 많음 (과적합)

● 과적합 문제의 이유
1. 모형의 복잡도 up
▶ 독립변수 or 파라미터가 너무 많음
▶ 과적합 가능성이 높아짐
2. 모형의 민감도 up
▶ 독립변수 값의 변화에 너무 민감하지 않도록 파라미터를 조정

● y = b_0 + b_1、x_1 + b_2、x_2
1. b_1 = 1
: x_1의 변화에 따라 y값의 변화가 크지 않음
2. b_1 = 10
: x_1의 변화에 따라 y값의 변화가 큼 (민감도가 높음)
▶ 민감도를 낮추기 위해 파라미터의 절댓값을 줄임

● 규제화
: 파라미터의 절댓값을 줄이기 위해 사용 (경우에 따라서 0으로 만들 수도 있음)
-> 일단 민감도를 낮춤 / 0으로 줄이면 복잡도 자체가 낮아짐
how?) 새로운 비용함수를 만듦. 원래 비용함수에 Penalty라고 하는 새로운 term을 더하면 됨 (E: 비용함수)
=> E_new = E_org + Penalty
=> abs(bi_new) < abs(bi_org) + Penalty

● Penalty term (규제화 항)의 종류
-> 수학적 모형의 파라미터: b_1,.....,b_k
1. L1 penalty
▶ lambda 、 (abs(b_1) + abs(b_2) + ... + abs(b_k))
▶ 파라미터의 절댓값을 0까지 줄일 수 있음 (일차항에 더해지니까)
2. L2 penalty
▶ lambda 、 (b_1^2 + b_2^2 + ... + b_k^2)
▶ 파라미터의 절댓값을 0까지 줄일 수는 없지만 보통 더 빠름 (이차항에 더해지니까)

=> lambda: 규제화 정도 (규제륾 많이 할수록 파라미터의 절댓값이 더 작아짐)

● 감성분석 (Sentiment Analysis)
▶ 특정 주제, 이슈에 대한 태도
▶ 긍정 vs 부정
▶ 근데 우리나라는 감성분석이라고 함 (정확한 번역은 아닌 듯)
▶ 감정분서이라고 할 때도 있는데 더더욱 아님
▶ 각 문서의 긍/부정을 찾음
▶ 분류문제 (긍, 부정이 종속변수)

● 문서분류
▶ 감성분석: 문서의 태도를 파악
1. 지도학습 알고리즘 사용
2. 감성어 사전 기반
▶ 예전에는 반반으로 썼으나, 최근에는 거의 첫 번째 방법만 씀

● 감성어 사전 기반 방법
: 단어 - 단어에 대한 감성 점수 (-1 ~ 1)
▶ 사전을 구축하는 게 매우 어려움

● 최근에는 지도학습 알고리즘 기반 방법이 훨씬 많이 사용됨

● 주제와 관련 => 빈도분석 등등

● 감성분석 => 형용사, 부사가 기본적으로 선되어야 함 (물론 명사 포함)
▶ 한번은 형용사만 해보고, 한번은 형용사와 부사만 해보고,,,
▶ 평가지표가 가장 좋은 방식으로

● 전처리 후 각 문서를 벡터화
-> 단어 정보를 이용해서 벡터화 (TF, TF-IFDF 정보)
-> 감성분석은 단어들의 빈도 정보와 TF-IDF 정보 중 빈도 정보 (TF 정보)를 사용하는 게 좋음

● 문서분류를 위해 사용되는 알고리즘
1. 전통적인 기계학습 알고리즘
- 로지스틱 회귀분석
- Naive Bayes
2. 신경망 기반(딥러닝 알고리즘)
- 순환신경망 (RNN)
- Transformer 기반 (BERT, GPT 등)

● 로지스틱 회귀 분석
-> 종속변수의 값: 0 (부정) / 1 (긍정)
▶ 로지스틱 회귀분석은 y가 1일 확률과 y가 0일 확률을 이용
▶ P(y=0|x) + P(y=1|x) = 1
▶ P(y=1) = 1/(1+e^(-z))
▶ z = b_0 + b_1、x_1 + b_2、x_2 + ... + b_k、x_k
▶ b_1 > 0 => y와 b_1 간의 관계가 양의 관계 (b_1이 사용될수록 긍정일 확률이 증가)
▶ b_1 < 0 => y와 b_1 간의 관계가 음의 관계
▶ 파라미터의 절댓값이 클수록 y가 1일 확률에 더 큰 영향을 미침 (음수든 양수든)

▶ 하나의 단어가 하나의 독립변수임

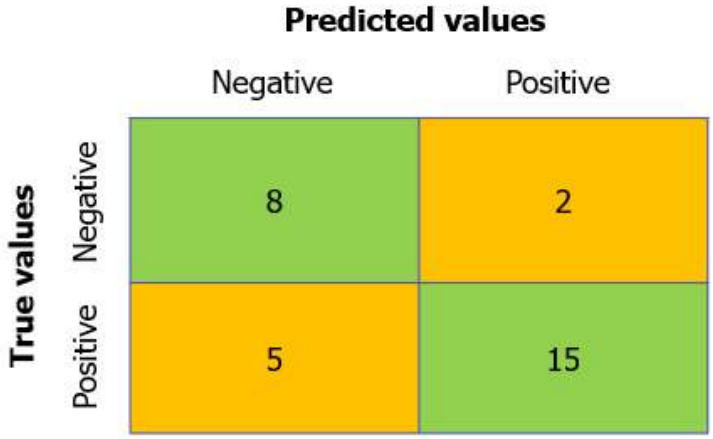
[12주차]

● 모형 성능 평가 지표
▶ 문제 종류에 따라 달라짐
▶ 회귀 문제: RMSE, R^2를 주로 사용
▶ 분류 문제: 정확도, 재현율, 정밀도, F1

● 로지스틱 회귀분석에서의 확률
1) b_i > 0 : x_i & p(y=1) -> 양의 관계
2) b_i < 0 : x_i & p(y=1) -> 음의 관계

● 규제화
▶ 과적합 문제 해결
▶ L1, L2 사용

● 분류 모델 성능 평가지표
▶ confusion matrix
: 종속변수의 실제값과 모형을 통해 예측이 된 종속변수 값에 따른 관측치들의 분포를 나타내는 행렬 (y가 취할 수 있는 값: positive -> 1 / negative ->0)
▶ True negative (진음성?)
▶ False Positive (위양성?)
▶ False Negative (위음성?)
▶ True Positive (진양성?)



1. 정확도 (Accuracy)

▶ 전체 중에 초록색 부분 (모형이 종속변수 값을 정확히 예측한 수)

▶ 여기서는 23/30

2. 정밀도 (Recall)

▶ for positive class: 모형을 통해 종속변수 값이 positive라고 예측된 애들 중 진짜 종속변수 값이 positivie인 애들 (15/17)

▶ for negative class: 모형을 통해 종속변수 값이 negative라고 예측된 애들 중 진짜 종속변수 값이 negative인 애들 (8/13))

3. 재현율 (Precision)

▶ for positive class: 원래 종속변수가 positive인 애들 중에서 positive로 예측한 애들 (15/20)

▶ for negative class: 원래 종속변수가 negative인 애들 중에서 negative로 예측한 애들 (8/10)

4. F1

▶ F1 score는 재현율과 정밀도의 조화평균

▶ 정밀도와 재현율 모두를 고려한 지표

▶ 2*정밀도*재현율/(정밀도 + 재현율)

● 벡터화

▶ 빈도 vs TFIDF => 군집화는 TFIDF를 기반으로 분석

▶ 감성분석 => 빈도를 기반으로 분석

● 토픽 모델링

▶ 특정 문서의 주제를 찾는 것

1) 행렬기반 방법 (LSI)

2) 확률기반 방법 (LDA)

3) BERT

● 확률적 생성 모형

▶ 데이터가 특정 확률 모형 (혹은 확률 분포)을 이용해서 생성되었다고 가정

▶ 관측된 데이터를 가지고 역으로 확률 분포의 형태를 찾는 방법

● 주요 가정

▶ 하나의 문서는 k개의 여러 주제를 서로 다른 확률로 다루고 있음

▶ 코퍼스에 존재하는 문서들은 동일한 수의 주제를 다루고 있음

● LDA

▶ 확률적 생성모형: LDA의 작동 방식

▶ 이 두 개의 확률분포를 실제 문서, 단어 정보를 가지고 역으로 추정 (베이지안 추정)

● 두 가지 확률 분포

1) 문서별 주제 확률분포

문서 1: 주제1(0.8), 주제2(0.1), 주제3(0.3)

문서 2: 주제1(0.2), 주제2(0.7), 주제3(0.1)

문서 3:

2) 주제별 단어 분포

주제1: 단어1(0.6), 단어2(0.2), 단어3(0.2)

● total_sections_morphs.p

-> 네이버의 정치, 경제, 사회 섹션 기사

-> 사람들이 많이 본 순 하루 30개, 3달 기간

-> 주제의 수는 연구자가 결정

● LDA는 sckit learn에서 제공하는 CountVectorizer 클래스를 사용하는 것이 아니라, 젠심에서 제공하는 별도의 또다른 클래스를 사용하여 각각의 문서를 벡터로 변환

● 적절한 주제의 수를 찾기 위한 방법 (실루엣 스코어 사용 불가)

1. 혼잡도 (perplexity) -> 우도(likelihood_데이터를 설명하는 정도)와 반비례

▶ 우리의 모형이 데이터를 잘 설명할수록 혼잡도 값이 작아짐 (제일 작을 때가 좋은 것)

▶ gensim은 log(perplexity) 값을 사용함

2. 응집도 (coherence)

▶ 우리의 모형이 데이터를 잘 설명할수록 응집도 값이 커짐 (제일 클 때가 좋은 것)

▶ 근데 응집도, 혼잡도는 안정적이지 않음. 결과가 자꾸 달라지거나 이상할 수 있음

▶ 즉 두 지표는 그렇게 신뢰적이진 않고 그저 참고용

[13주차]

● LDA는 확률적 생성모형
✓ 토필모델링 수행결과:
1. 주제별 단어분포
▶ 각각의 문서에서 어떠한 단어의 확률이 높은지를 보는 확률분포
2. 문서별 주제분포
▶ 텍스트 데이터 안에 존재하는 여러 개의 문서에 대한 주제 분포를 확인

● LDA에서 찾고자 하는 주제의 수는 우리가 결정
(해당 주제에 대한 기본 지식이 필요함)
▶ 토픽모델링 결과가 좋으려면, 시각화 때 겹치는 애들의 수가 많지 않도록 해야 함
▶ 오른쪽 막대 그래프는, 각각의 주제별로 각각의 단어들이 몇퍼센트 정도로 관련이 있는지 정리

● 딥러닝 모델: 신경망 모형 (또다른 기계학습 알고리즘) 기반

● 신경망 모형: 사람의 뇌에 존재하는 신경망을 모방해서 만든 모형

● 은닉층이 여러 개 있는 신경망 모형: 다층 신경망
▶ 딥러닝 모형 (딥러닝 알고리즘) = 다층 신경망
▶ 이것이 인공지능 기술을 구현하는 데 있어 주요 역할을 함
▶ 각 층 (layer)에서 원은 노드
▶ 입력층은 입력노드, 은닉층은 은닉노드, 출력층은 출력노드
▶ 입력층에 존재하는 입력노드는 각각의 관측치에 대해 데이터를 입력받고 이를 그 다음 층으로 전달하는 역할
▶ 은닉노드는 이전 단계에서 전달이 된 데이터의 정보들 중 정답을 예측하는 데 있어서 중요한 역할을 하는 정보를 추출하는 것이 은닉노드
▶ 출력노드는 종속변수에 대한 예측치를 계산 및 출력

● 신경망에 존재하는 층들 중 은닉층이 제일 중요함
▶ 은닉층이 여러 개 있을수록 중요한 정보를 더 잘 추출할 수 있음
▶ 입력층과 은닉층에는 입력노드와 은닉노드 외에 추가로 편향노드가 하나씩 더 있음. 선형회귀분석에서 절편의 역할을 해줌

● 신경망 모형의 구조를 어떻게 결정할 수 있는가

● 각 층의 수(입력층의 수, 은닉층의 수, 출력층의 수), 각 층에 존재하는 노드의 수 (입력 노드의 수, 은닉 노드의 수, 출력 노드의 수) 결정해야 함
▶ 직접적으로 결정, 변경할 수 있는 것은 은닉층의 수 및 은닉 노드의 수
▶ 나머지는 자동으로 결정됨

● 입력 노드의 수는 독립변수의 수

● 출력 노드의 수는 문제의 종류에 따라 달라짐
1. 회귀문제 (종속변수가 연속형)
▶ 1개 (회귀문제에서 출력노드에 입력된 것이 그냥 그대로 출력되어 예측치가 됨)
2. 분류문제 (종속변수가 범주형)
▶ 종속변수가 취할 수 있는 값의 수

● 출력노드가 출력하는 값
: 종속변수가 각 값을 취할 확률
▶ P(X_1=0)
▶ P(X_2=1).....
▶ 여기서 출력한 게 비용함수에 들어가서 교차엔트로피를 계산함

● 신경망 작동원리
▶ 지도학습을 기준으로 설명
▶ 지도학습 ML 알고리즘 작동 방식과 거의 동일

● 여기서 check (지도학습 알고리즘의 작동방식이 뭐였지?)
1. 데이터 준비
▶ 정답데이터 (+hint) + 문제데이터
▶ 정답데이터 = 학습데이터 + 평가데이터
▶ 학습데이터 (정답&힌트)
▶ 학습데이터의 정답과 힌트의 관계를 파악하기 위해 수학적 모형 사용
▶ 우린 학습을 통해 파라미터의 최적값을 사용해야 하고, 이를 위해 비용함수를 이용
▶ 비용함수는 우리가 사용하는 수학적 모형의 종류가 아닌 우리가 풀고자 하는 문제 종류에 따라 달라짐

● 신경망 모형; 일반적인 지도학습 알고리즘과 거의 동일
▶ 이 자체로 하나의 수학적 모형이기 때문
▶ 독립변수와 종속변수의 관계를 파악하기 위해 수학적 모형을 사용하는데, 신경망 모형 자체가 수학적 모형임
▶ 그러니 파라미터를 가짐
▶ 파라미터를 어떤 식으로 가지는가?
▶ 각각의 노드가 화살표로 연결되어 있는데, 각각의 화살표가 하나의 고유한 파라미터를 가짐
▶ 신경망 모형이 갖는 파라미터: 2가지
1) 편향 파라미터 / 2) 가중치 파라미터
이러한 파라미터는 독립변수들과 종속변수 간의 관계를 정의하는 역할
▶ 역시 최적값을 찾으면 됨

● 비용함수의 값을 최소화하는 과정에서 딥러닝은 경사하강법을 사용
▶ 일반적으로 텍스트나 비정형 데이터에 대해 문제를 잘 품
▶ 정형데이터는 잘 못 품

● 신경망 모형 결정

● 은닉노드가 입력된 값을 변경할 때 사용하는 함수는 활성화 함수
▶ 같은 층의 노드들은 동일한 활성화 함수 사용

● 활성화 함수의 조건
1. 비선형 함수 (그래야 비선형 관계를 더 잘 파악)
▶ 즉 정답을 예측하는 데 있어서 중요한 정보를 더 잘 추출할 수 있음
2. 경사하강법 사용
▶ 업데이트 공식 사용
3. 미분이 용이해야 함

● 소프트 맥스 함수 (분류문제의 은닉노드가 사용하는 함수)

● optimizer의 종류
▶ 최적 문제를 풀 때 쓰는 도구 (방법)
▶ 비용함수의 값을 최소화할 때 쓰는 방법 (딥러닝에서는 경사하강법)
▶ 기본 경사하강법은 한계를 가짐
1) 특정 단계에서 업데이트를 할 때 지금까지 업데이트된 정도가 반영되지 않음
2) 업데이트 횟수와 상관없이 learning rate가 고정되어 있음
+ 안장점의 경우, 편미분계수가 0이 되어서 더 이상 업데이트가 되지 않음
+ 속도가 느림
▶ 기본 경사하강법의 한계를 보완하기 위한 optimizers (요즘은 얘네들이 많이 사용됨)
1) momentum (첫 번째 한계점 보완)
2) Adaptive Gradient, RMSprop, Adadelta (두 번째 한계점 보완)
3) Adam (momentum, RMSprop의 장점을 합친 것으로 => 두 한계 보완)

● CNN: 합성곱 신경망 (이미지 분석)
=> 이미지 분류, 객체 탐지(->얼굴탐지), 이미지 분할, 고해상도 작업, 이미지 생성
● RNN: 순환 신경망 (텍스트 분석)

● Transformer은 encoder, decoder 모형 사용 (구글에서 발표한 알고리즘)

● 객체탐지 (object detection)
1. 객체가 있을 만한 상자 (경계 상자)
2. 예측이 된 경계 상자 안에 있는 물체가 무엇인지 탐지

● Pascal VOC: 20 classes만 찾을 수 있음
● MS COCO는 80개에 대한 물체 정보를 가지고 있음

● YOLO.v11
: 객체탐지 알고리즘 (you live look once)
▶ 울트라리틱스 라이브러리 설치

[14주차]

● 분석
1. 기계학습 x
▶ 빈도분석, 텍스트네트워크
2. 기계학습 기반
▶ 군집화, 문서분류(감성분석), 토픽모델링
▶ 지도학습 알고리즘 (로지스틱 알고리즘): 문서분류
▶ 비지도학습 알고리즘: 군집화

● 스크래핑
1. 소스코드 다운로드
▶ requests, selenium
2. 태그 추출
▶ Beautiful Soup

● 전처리
▶ 불용어가 제거된 특정한 품사의 단어들

● 문서의 벡터화 (기계학습 알고리즘을 사용할 때)
▶ 단어 정보 (빈도 vs TF-IDF)