<데이터분석과기계학습 9주차>

앙상블 학습 / 감성분석 / 회귀분석

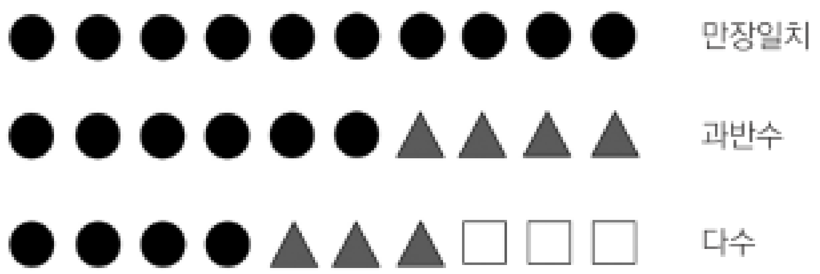
인공지능융합공학부 데이터사이언스전공
곽찬희

앙상블 학습

7.1. 앙상블 학습

- 앙상블 학습(ensemble learning)
  - ✓ 여러 분류기를 하나의 메타 분류기로 연결해 개별 분류기보다 더 좋은 일반화 성능 달성을 목표
    - 1명의 전문가 vs. 10명의 전문가
  - ✓ 이진 분류일 때는 과반수 투표 (majority voting): 과반수가 넘어가는 것을 선택
  - ✓ 다중 분류일 때는 다수결 투표 (plurality voting): 가장 많은 빈도(최빈값, mode)를 선택

그림 7-1 과반수 투표와 다수결 투표



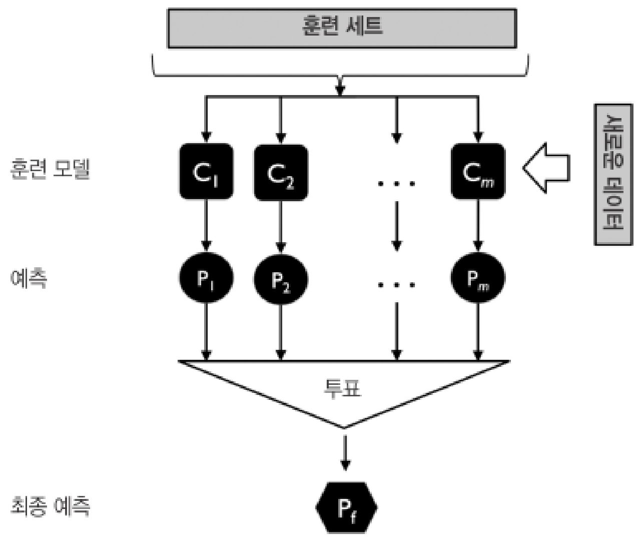
7.1. 앙상블 학습

- 앙상블은 다양하게 구성 가능
  - ✓ 여러 알고리즘을 사용하는 경우: SVM + DT + LogisticRegression
  - ✓ 한 알고리즘에 훈련 세트의 부분 집합(subset of training set)을 다르게 적용하는 경우: RF

- 개별 분류기 C_j 의 예측 레이블을 모아 예측값을 정함

$$\hat{y} = mode\{C_1(x), C_2(x), \dots, C_m(x)\}$$

▼ 그림 7-2 과반수 투표를 사용한 앙상블 방법



7.1. 앙상블 학습

- 예) class1 = -1, class2 = +1 이라고 할 때 과반수 투표는 다음과 같이 표현 가능
  - ✓ $C(x) = sign[\sum_j^m C_j(x)] = \begin{cases} 1 \ \sum C_j(x) \geq 0 \text{ 일 때,} \\ \quad -1 \text{ 그 외} \end{cases}$
- 앙상블이 개별 분류기보다 성능이 좋은 이유?
  - ✓ 이진 분류 작업의 애러율을 ε 라고 정하고, n 개의 분류기가 있다고 가정
  - ✓ 각 분류기는 독립적이고 발생하는 오차는 서로 상관관계가 없다고 한다면, 분류기들을 포함한 앙상블의 오차 확률은 이항 분포의 확률 질량 함수로 표현 가능

$$P(y \geq k) = \sum_k^n \varepsilon^k \binom{n}{k} (1-\varepsilon)^{n-k} = \varepsilon_{ensemble}$$

  - ✓ 만약에 ε 가 .25 인 분류기 11개를 묶은 앙상블 모델이라면 애러율은

$$P(y \geq k) = \sum_6^{11} 0.25^k \binom{11}{k} (1-0.25)^{11-k} = 0.034\ (3.4\%)$$

7.2. 다수결 투표를 사용한 분류 앙상블

- 가중치가 적용된 다수결 투표는 다음과 같이 표현 가능

$$\hat{y} = argmax \sum_{j=1}^{m} w_j \chi_A (C_j(x) = i)$$

✓ w_j : 분류기 C_j 에 대한 가중치
✓ χ_A: 특성함수 (characteristic function) $[C_j(x) = i \ \in A]$

- 만약 가중치가 동일하다면,

$$\hat{y} = mode\{C_1(x), C_2(x), \ldots, C_m(x)\}$$

7.2. 다수결 투표를 사용한 분류 앙상블

- 예) 세 개의 분류기 $C_j (j \in \{1, 2, 3\})$ 가 있고, 샘플 x 의 클래스($i \in \{0, 1\}$) 예측

$$C_1(x) \rightarrow 0, C_2(x) \rightarrow 0, C_3(x) \rightarrow 1$$

$$\hat{y} = mode\{0, 0, 1\} = 0$$

만약 가중치가 존재한다면

$$\hat{y} = argmax \sum_{j=1}^{m} w_j \chi_A (C_j(x) = i)$$

$$= argmax[0.2 \times i_0 + 0.2 \ \times i_0 + 0.6 \ \times i_1] = 1$$

C_3의 가중치가 0.6, C_1, C_2가 0.2일 때, C_3의 영향을 세 배 증가. 즉,

$$\hat{y} = mode\{0, 0, 1, 1, 1\} = 1$$

7.2. 다수결 투표를 사용한 분류 앙상블

- **만약 확률을 사용한다면,**

$$\hat{y} = argmax \sum_{j=1}^{m} w_j p_{ij}$$

✓ **여기서** p_{ij} **는 클래스 I 에 대한 j 번째 분류기의 예측 확률**

- **예) 만약** $i \in \{0, 1\}, C_j(j \ \in \{1, 2, 3\})$ **일 때,**
  $C_1(x) \rightarrow [0.9, 0.1], C_2(x) \rightarrow [0.8, 0.2], C_3(x) \rightarrow [0.4, 0.6]$ **이라면 각 클래스의 확률은**

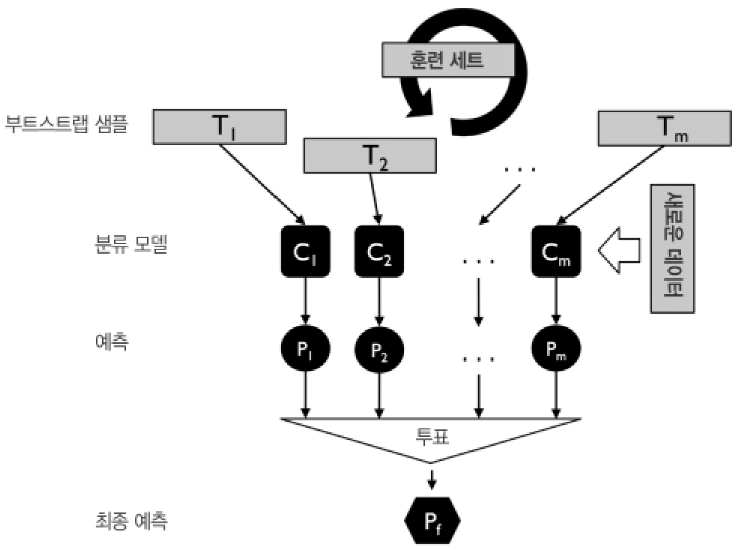
$$p(i_0|x) = 0.2 \ \times 0.9 + 0.2 \ \times 0.8 + 0.6 \times 0.4 = 0.58$$
$$p(i_1|x) = 0.2 \ \times 0.1 + 0.2 \ \times 0.2 + 0.6 \times 0.6 = 0.42$$

$$\hat{y} = argmax \sum_{j=1}^{m} w_j p_{ij} = argmax[p(i_0|x), p(i_1|x)] = \ 0$$

7.3. 배깅: 부트스트랩 샘플링을 통한 분류 앙상블

- **배깅(bagging)**
  - ✓ 앙상블에 있는 개별 분류기를 동일한 훈련 세트로 학습하는 것이 아닌, 원본 훈련 세트에서 부트스트랩(bootstrap) 샘플 (중복을 허용한 랜덤 샘플, RF에서 다뤘죠?) 을 뽑아서 사용
  - ✓ Bootstrap aggregating 이라고도 불림

▼ 그림 7-6 배깅



▼ 그림 7-7 부트스트랩 샘플링의 작동 방식

샘플 인덱스	배깅 1	배깅 2	···
1	2	7	···
2	2	3	···
3	1	2	···
4	3	1	···
5	7	1	···
6	2	7	···
7	4	7	···

C_1 C_2 C_m

7.4. 약한 학습기를 이용한 에이다부스트

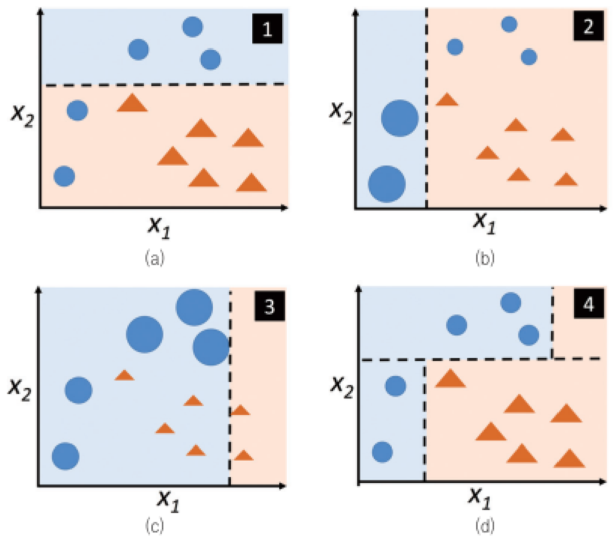
- 부스팅 (boosting)
  - ✓ 여러 개의 약한 학습기 (weak learner)라 불리는 간단한 분류기들을 활용한 앙상블 학습법
  - ✓ 여기서는 가장 유명한 부스팅인 AdaBoost (에이다부스트 혹은 아다부스트)를 다룸

- 부스팅 과정
  - ✓ 훈련 세트 D에서 중복을 허용하지 않고 랜덤한 부분 집합 d_1 를 뽑아 약한 학습기 C_1를 훈련
  - ✓ 훈련 세트에서 중복을 허용하지 않고 두 번째 훈련 부분 집합 d_2를 뽑고, 이전에 잘못 분류된 샘플의 50%를 더해 약한 학습기 C_2를 훈련
  - ✓ 훈련 세트 D 에서 C_1 과 C_2 에서 잘못 분류한 훈련 샘플 d_3 를 찾아 세 번째 약한 학습기 C_3 를 훈련
  - ✓ C_1 , C_2 , C_3 의 다수결 투표 진행

7.4. 약한 학습기를 이용한 에이다부스트

- AdaBoost
  - ✓ 약한 학습기 훈련 시 전체 데이터 사용
  - ✓ 훈련 샘플은 반복마다 가중치를 다시 부여하고, 이전 학습기의 실수를 줄이는 방향으로 가중치 업데이트
  - ✓ 예시
    - a: 깊이가 1인 트리를 이용해 두 개의 클래스로 나눔
    - b: a에서 잘못 분류한 데이터는 가중치 up, 잘 분류한 것은 가중치 down
    - c: b에서 잘못 분류한 데이터 가중치 up, 잘 분류한 것은 가중치 down
    - d: 세 개의 learner를 이용하여 다수결 투표

그림 7-9 에이다부스트



강남대학교 KANGNAM UNIVERSITY

7.4. 약한 학습기를 이용한 에이다부스트

- AdaBoost 의 Pseudo Code

1. 가중치 벡터 w를 동일하게 설정 $\sum w_i = 1$
2. m번의 부스팅 반복의 j 번째에서 다음을 수행
   a. 가중치가 부여된 약한 학습기 훈련 $C_j = train(X, y, w)$
   b. 클래스 레이블 예측 $\hat{y} = predict\left(C_j, X\right)$
   c. 가중치가 적용된 애러 계산 $\varepsilon = w \cdot (\hat{y} \neq y)$
   d. 학습기 가중치 계산 $\alpha_j = 0.5 \log \frac{1-\varepsilon}{\varepsilon}$
   e. 가중치 업데이트 $w := w \times \exp(-\alpha_j \times \hat{y} \times y)$
   f. 합이 1이 되도록 가중치 정규화 $w := w \,/\, \sum w_i$
3. 최종 예측 수행 $\hat{y} = \left(\sum_{j=1}^{m} \alpha_j \times predict(C_j, X)) > 0\right)$

✓ $(\hat{y} \neq y)$ 는 예측이 잘못되면 1, 맞으면 0을 표시

7.4. 약한 학습기를 이용한 에이다부스트

- **애러율 계산**
  - ✓ $\varepsilon = 0.1 \times 0 + 0.1 \times 0 + 0.1 \times 0 + 0.1 \times 0 + 0.1 \times 0 + 0.1 \times 0$
    $+0.1 \times 1 + 0.1 \times 1 + 0.1 \times 1 + 0.1 \times 0 = 0.3$
    - 10개의 샘플이므로 각 weight 는 1/10, 예측이 맞은 경우 0, 아니면 1을 곱함

▼ 그림 7-10 열 개의 훈련 샘플로 구성된 훈련 세트

샘플 인덱스	x	y	가중치	$\hat{y}(x <= 3.0)?$	정답인가?	업데이트된 가중치
1	1.0	1	0.1	1	Yes	0.072
2	2.0	1	0.1	1	Yes	0.072
3	3.0	1	0.1	1	Yes	0.072
4	4.0	−1	0.1	−1	Yes	0.072
5	5.0	−1	0.1	−1	Yes	0.072
6	6.0	−1	0.1	−1	Yes	0.072
7	7.0	1	0.1	−1	No	0.167
8	8.0	1	0.1	−1	No	0.167
9	9.0	1	0.1	−1	No	0.167
10	10.0	−1	0.1	−1	Yes	0.072

- **학습 가중치 계산**
  - ✓ $\alpha_j = 0.5 \log\left(\frac{1-\varepsilon}{\varepsilon}\right) \cong .424$
- **가중치 업데이트**
  - ✓ $w := w \times \exp(-\alpha_j \times \hat{y} \times y)$
    - $\hat{y} \times y$ 는 예측 클래스 레이블 벡터 * 실제 클래스 레이블 이므로 맞으면 양수값이 됨
    - Alpha 는 항상 양수라 가정 (무작위 추정인 0.5 보다 ε가 낮다고 가정)
    - 실제론, 예측이 맞은 샘플 가중치는 업데이트 하지 않음

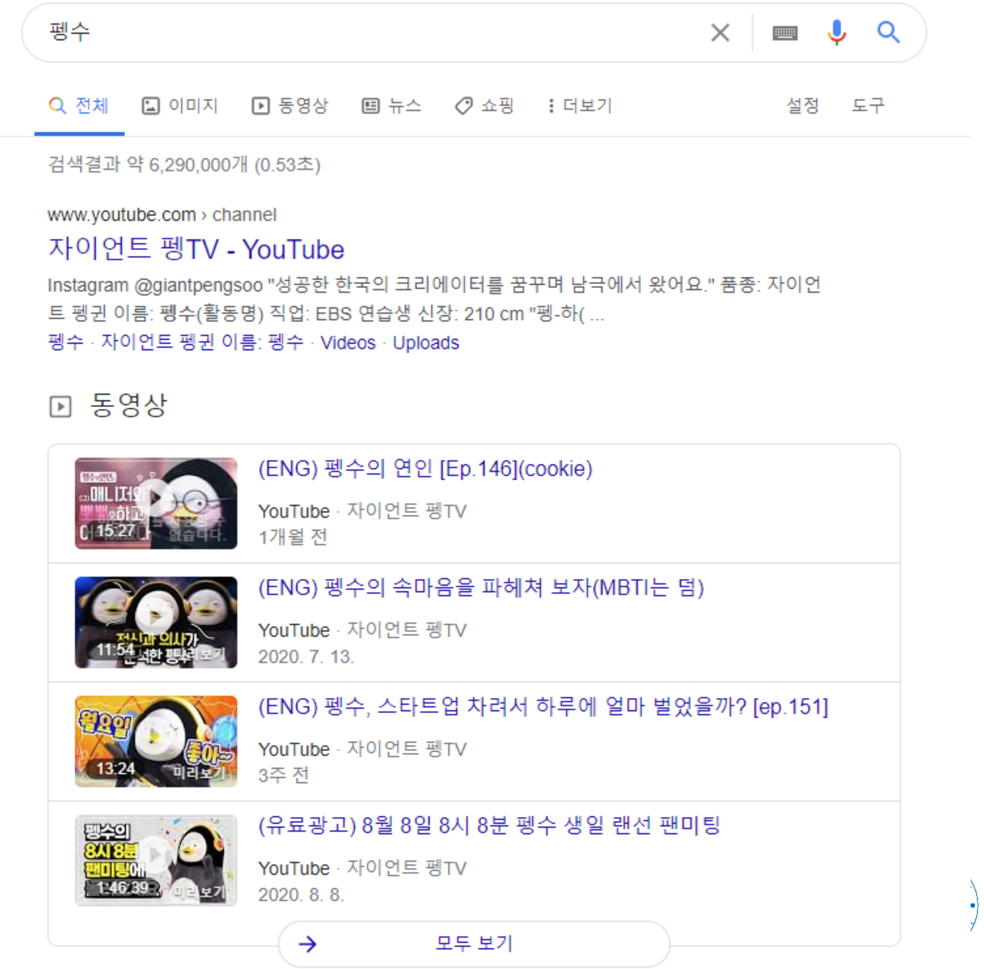
7.4. 약한 학습기를 이용한 에이다부스트

- 따라서 예측이 맞은 경우는,
  - ✓ $0.1 \times \exp(-0.424 \times 1 \times 1) \approx 0.065$
- 예측이 틀린 경우는
  - ✓ $0.1 \times \exp(-0.424 \times 1 \times (-1)) \approx 0.153$ 혹은
  - ✓ $0.1 \times \exp(-0.424 \times (-1) \times 1) \approx 0.153$
- 마지막으로 정규화를 하면
  - ✓ $\sum w_i = 7 \times 0.065 + 3 \times 0.153 = 0.914$ 이므로
  - ✓ 옳게 분류된 샘플의 가중치는 $\frac{0.065}{0.914} \approx 0.071$ 로 감소
  - ✓ 잘못 분류된 샘플의 가중치는 $\frac{0.153}{0.914} \approx 0.167$로 증가
  - ✓ (처음 기준값이 0.1이었던 것을 기억하세요!)

감성 분석

감성 분석

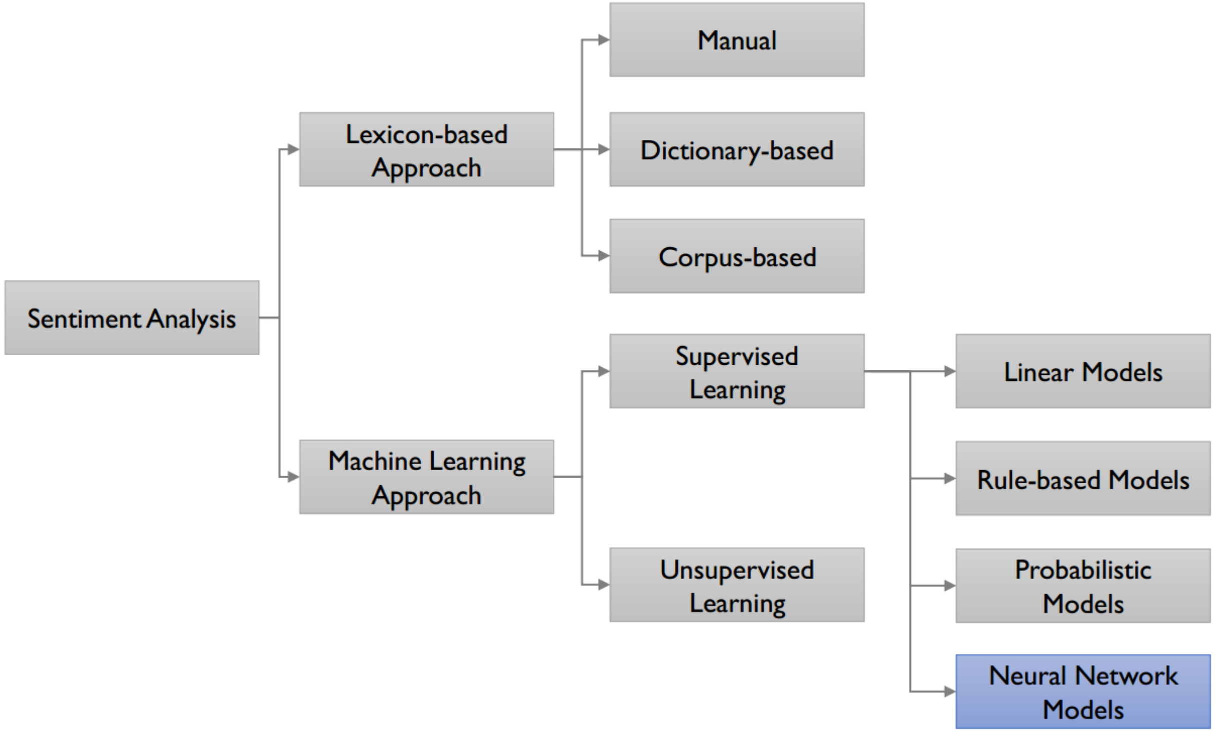
- **감성분석 (Sentimental Analysis)**
  - ✓ 문장의 단어들의 구성을 근거로, 문장의 긍정/부정 등을 판단하는 방법



https://img1.daumcdn.net/thumb/R800x0/?scode=mtistory2&fname=https%3A%2F%2Ft1.daumcdn.net%2Fcfile%2Ftistory%2F99C9FA335DC91AB810

감성 분석

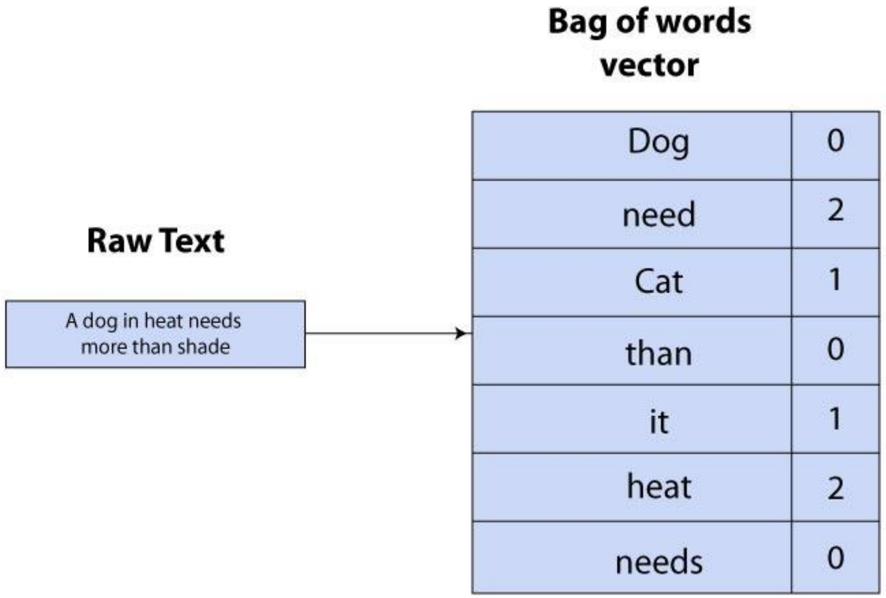
- **크게 어휘 기반 (Lexicon-based) 와 머신러닝 기반 (ML)로 나눌 수 있음**



https://user-images.githubusercontent.com/45377884/88989668-1cda5280-d317-11ea-8704-6c6555e171c1.png

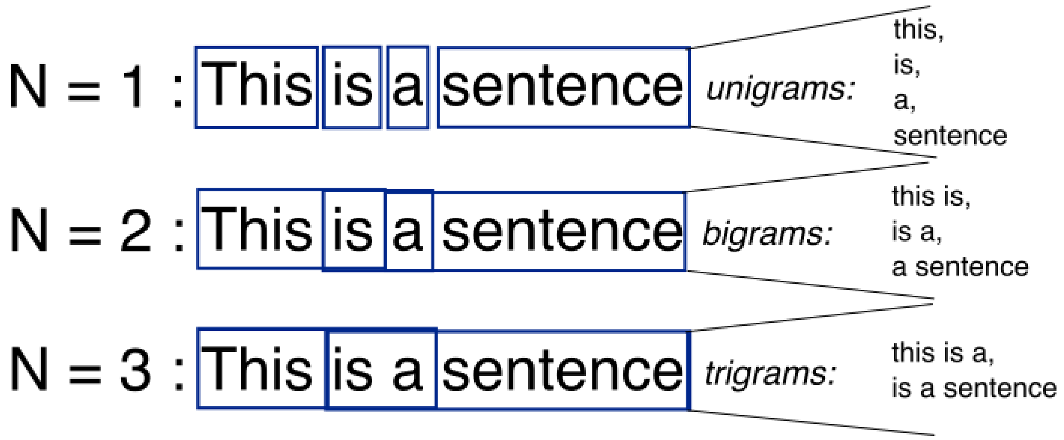
8.2. BoW 모델

- **BoW(Bag of Words)**
  - ✓ 전체 문서에 대해 고유한 토큰(ex. 단어로 이뤄진 어휘 사전) 을 만들고, 특정 문서에서 각 단어가 얼마나 자주 등장하는지 체크하는 벡터를 생성



8.2. BoW 모델

- **n-gram**
  - ✓ n개의 연속된 단어로 이뤄진 토큰을 사용
  - ✓ 예) 1-그램(혹은 unigram): 'the', 'sun', 'is', 'shining'
    2-그램: 'the sun', 'sun is', 'is shining'
  - ✓ n에 어떤 값을 설정하느냐에 따라 모델의 성능이 달라짐



https://images.deepai.org/glossary-terms/867de904ba9b46869af29cead3194b6c/8ARA1.png

8.2.2. TF-IDF

- TF-IDF (Term Frequency - Inverse Document Frequency)
  - ✓ 핵심 개념: 자주 등장하는 단어는 판별에 용이하지 않음(필요한 정보를 제공하지 않음)
  - ✓ TFIDF는 단어 빈도(TF)와 역문서 빈도(IDF)의 곱으로 나타냄

$$tfidf(t,d) = \; tf(t,d) \times idf(t,d)$$

$tf(t,d)$: 문서 d에 등장한 단어 t의 빈도

$$idf(t,d) = \log \frac{n_d}{1 + df(d,t)}$$

n_d: 전체 문서의 개수

$tf(t,d)$: 단어 t 가 포함된 문서 d 의 개수

8.2.2. TF-IDF

- Sklearn 에서는 공식을 조금 수정해서 사용 (0이 되는 것을 방지)

$$tfidf(t,d) = \ tf(t,d) \times (idf(t,d) + 1)$$

$$idf(t,d) = \log \frac{n_d + 1}{1 + df(d,t)}$$

- 일반적으로 TF-IDF를 계산할 때는 단어의 빈도를 정규화하지만, sklearn은 l2-normalization 적용

$$v_{norm} = \frac{v}{\|v\|_2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + \ \dots + v_n^2}} = \frac{v}{(\sum v_i^2)^{\frac{1}{2}}}$$

8.2.3. 텍스트 데이터의 정제

- 분석에 텍스트를 사용하기 전 필요 없는 문자들을 삭제해야 함
  - ✓ HTML 마크업
  - ✓ 구두점 (. , ‘ " -)
  - ✓ 이모티콘 (:-), :-(, :D :O)

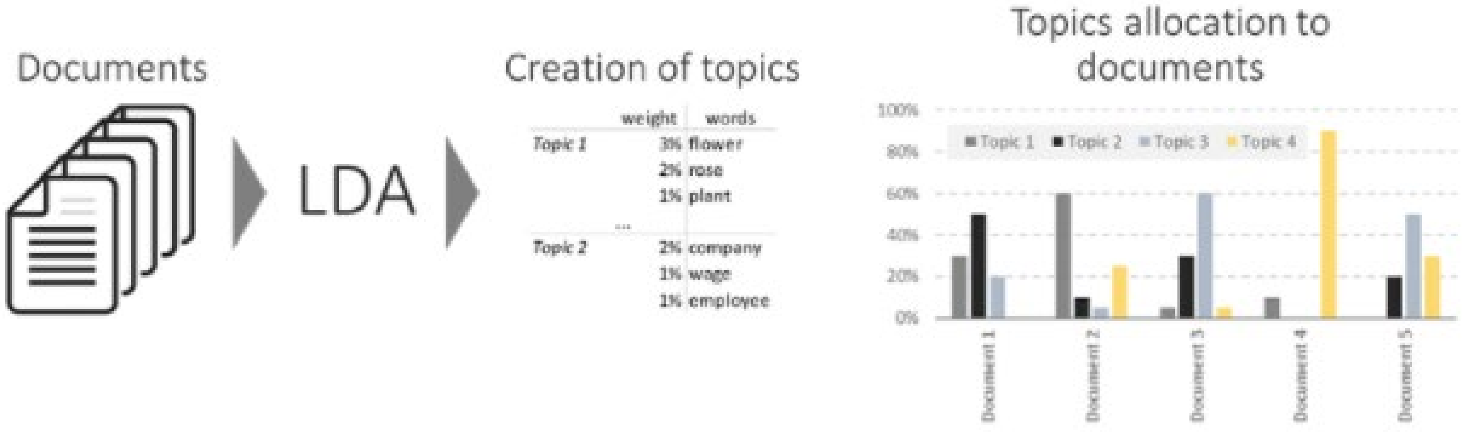
- 정규 표현식 (Regular Expression)을 사용하면 편리하게 제거 가능
  - ✓ 정규식은 계산비용이 비싼 함수이지만, 작은 규모의 텍스트를 처리하기에 용이함
  - ✓ 참고
    - http://www.nextree.co.kr/p4327/

8.2.4. 문서를 토큰으로 나누기

- 전처리한 데이터를 낱개의 토큰으로 나누는 법을 생각해봐야 함
- 가장 간단한 방법: 공백을 기준으로 개별 단어로 나누기
  - ✓ We are the champions -> 'we' 'are' 'the' 'champions'
- 어간 추출 (Stemming)
  - ✓ 변하지 않는 기본 형태인 어간을 추출하는 방법
  - ✓ Porter Stemmer 가 유명 (python nltk 패키지)
  - ✓ 예) Works, worked, working -> work
- 표제어 추출 (Lemmatization)
  - ✓ 기본형을 찾는다는 점에서 어간 추출과 유사
  - ✓ 품사를 보존한다는 차이점이 있음 (문맥 고려)

8.5. 잠재 디리클레 할당을 사용한 토픽 모델링(LDA)

- LDA
  - ✓ 비지도학습의 일종으로, 자주 등장하는 단어의 그룹(토픽)을 찾는 확률적 생성 모델
  - ✓ BoW행렬을 (1)문서-토픽 행렬, (2) 단어-토픽 행렬 로 분해하여, 두 행렬의 곱이 최소 오차로 BoW 입력 행렬을 재구성할 수 있게 함
  - ✓ 미리 토픽의 개수를 정해야 함 (하이퍼파라미터)
  - ✓ 앞서 PCA와 함께 나왔던 LDA와는 다르니 혼동 주의
  - ✓ https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/06/01/LDA/



선형 회귀

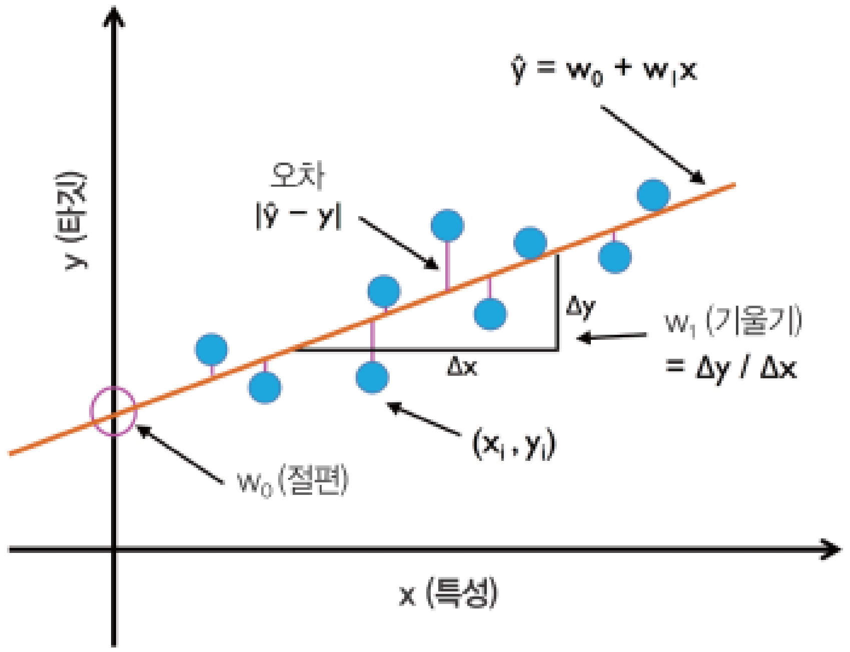
10.1. 선형 회귀

- **단순 선형 회귀(Simple Linear Regression)**
  - ✓ 하나의 특성 (설명변수, explanatory variable) x 와 연속적인 타깃 (응답변수, response variable) y 사이의 선형적인 관계를 모델링

$$y = w_0 + w_1 x$$

  - ✓ w_0 은 y축의 절편, w_1은 특성의 가중치
  - ✓ 회귀직선: 데이터에 가장 잘 맞는 직선
  - ✓ 회귀 직선과 데이터 간의 차(오차): 잔차(residual)
    - 회귀로 설명되지 않는 남은 차이로 이해

그림 10-1 선형 회귀



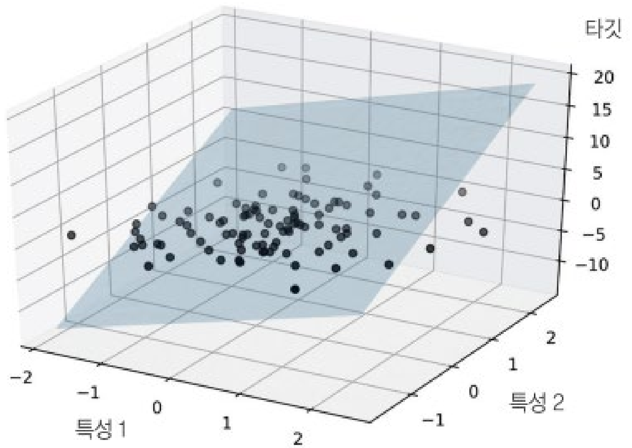
10.1. 선형 회귀

- **다중 선형 회귀(Multiple Linear Regression)**
  - ✓ 단순 선형 회귀를 일반화해 여러 독립 변수를 이용한 종속 변수 추정

$$y = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \sum_{i=0}^{m} w_i x_i = w^T x$$

  - ✓ 만약 m=2 라면 다음과 같이 나타낼 수 있음
  - ✓ 3차원 이상은? 시각화가 어려움

그림 10-2 다변량 회귀 모델



(추가) 회귀의 네 가지 가정

1. 선형성: 독립변수와 종속변수는 선형적인 관계를 갖는다
2. 독립성: 잔차 간은 서로 독립이다
3. 등분산성: 잔차 간의 분산은 같다
4. 정규성: 잔차는 정규분포를 따른다

10.2. 주택 데이터셋

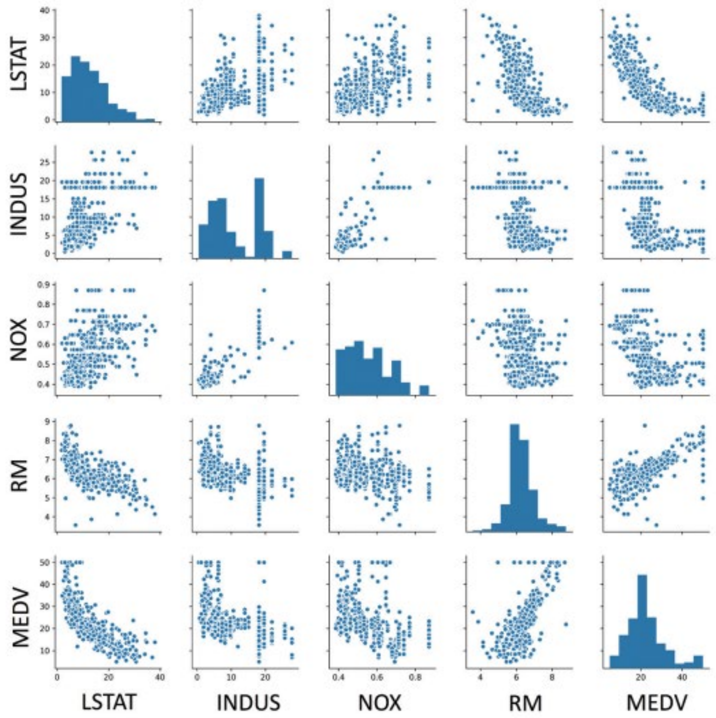
- 1978년 Harrison & Rubinfeld 가 수집한 유명한 데이터
  - ✓ https://raw.githubusercontent.com/rickiepark/python-machine-learning-book-2nd-edition/master/code/ch10/housing.data.txt
  - ✓ 최근 UCI 머신러닝 데이터에서 삭제
    - 흑인 인구가 인종차별적이라는 지적

10.2.2. 시각화

- 탐색적 데이터 분석 (Exploratory Data Analysis)
  - ✓ 데이터를 탐색적으로 접근하면서 특성을 파악하고, 적절한 변환을 준비

- 산점도 행렬(Scatterplot matrix)
  - ✓ seaborn 의 pairplot 을 이용하면 편리

그림 10-4 주택 데이터셋의 산점도 행렬



10.2.3. 상관관계 행렬

- 상관 관계(correlation)
  - ✓ 두 개의 변수가 선형적으로 어떤 관계를 보여주는가?
  - ✓ 공분산(covariance)를 스케일 조정한 것
  - ✓ 피어슨 상관계수 (Pearson Correlation Coefficient)

$$r = \frac{\sum_{i=1}^{n}[(x^{(i)} - \mu_x)(y^{(i)} - \mu_y)]}{\sqrt{\sum_{i=1}^{n}(x^{(i)} - \mu_x)^2}\sqrt{\sum_{i=1}^{n}\left(y^{(i)} - \mu_y\right)^2}} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

σ_{xy}: x, y의 공분산

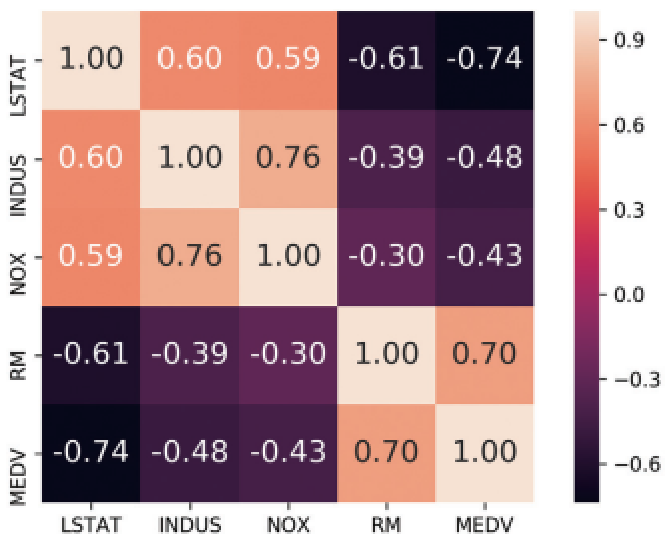
$\sigma_x \sigma_y$: x, y의 표준편차

즉, 공분산을 x, y의 분포를 고려하여 크기 조정한 것

10.2.3. 상관관계 행렬

- 상관 관계(correlation)
  - ✓ 상관관계의 히트맵을 이용하면 큰 경향성을 볼 수 있음
  - ✓ seaborn 의 heatmap 사용

그림 10-5 상관관계 행렬의 히트맵



10.3. 최소 제곱 선형 회귀

- 최소 제곱법 (Ordinary Least Squares, OLS)
  - ✓ 샘플 포인트까지의 수직 거리 제곱의 합 (잔차 제곱의 합)을 최소화하여 직선을 정의
  - ✓ 선형 최소 제곱법(linear least squares)라고도 함
  - ✓ 어디서 봤더라?

- Adaline 의 Gradient Descent & Stochastic Gradient Descent

$$J(w) = \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

  - ✓ 예측값과 실제값의 차이의 제곱
  - ✓ 잔차 = 예측값 - 실제값
  - ✓ 즉, 잔차의 제곱

10.4. RANSAC

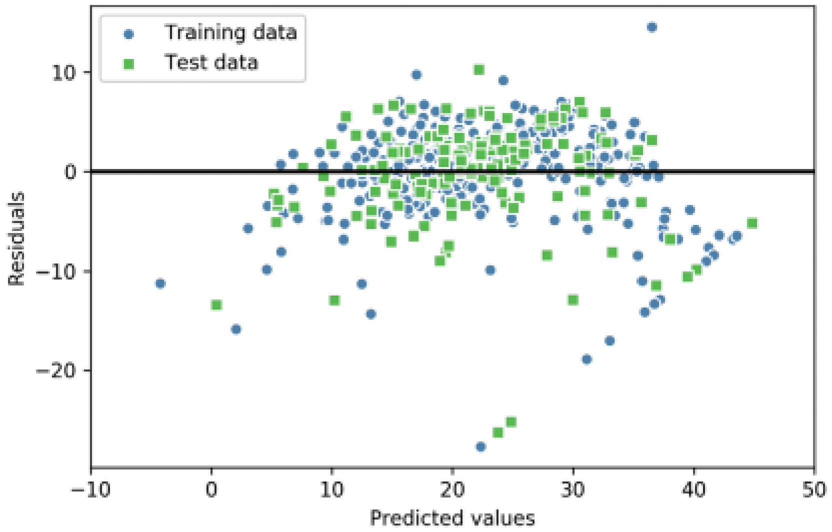
- RANSAC(RANdom Sample Consensus)
  - ✓ 정상치(inlier)라는 일부 데이터로 회귀 모델을 훈련
  - ✓ 특이치(outlier)가 추정에 영향을 끼치지 못하는 효과

  - ✓ 알고리즘 정리
    1. 랜덤하게 일부 샘플을 정상치로 선택하여 모델 훈련
    2. 훈련 모델에서 다른 모든 포인트 테스트 후, 사용자가 입력한 허용 오차 안에 속한 포인트를 정상치에 추가
    3. 모든 정상치를 사용하여 모델 다시 훈련
    4. 훈련된 모델과 정상치 간 오차 추정
    5. 성능이 사용자가 정한 임계값에 도달하거나, 지정된 반복 횟수에 도달하면 종료. 그렇지 않으면 1로

10.5. OLS 성능 평가

- 잔차 그래프 (residual plot)
  - ✓ 오차가 랜덤하게 퍼져 있는지, 선형성을 가지지 않는지 확인

▼ 그림 10-10 잔차 그래프



10.5. OLS 성능 평가

- 평균 제곱 오차 (Mean Squared Error, MSE)
  - ✓ 제곱 오차합(Sum of Squared Error)를 샘플 개수로 나눈 것
  - ✓ 평균적으로 얼마나 예측 오차가 발생하는지 확인

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{SSE}{n}$$

10.5. OLS 성능 평가

- 결정 계수 (Coefficient of Determination, R^2)
  - ✓ MSE의 표준화 버전
  - ✓ 타깃의 분산에서 모델이 잡아낸 비율(설명력)

$$R^2 = 1 - \frac{SSE}{SST}$$

  - ✓ 제곱 오차합(SST, total sum of squares)

$$SST = \sum_{i=1}^{n} \left(y^{(i)} - \mu_y\right)^2$$

  - ✓ 이를 이용해 정리하면,

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2}{\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \mu_y\right)^2} = 1 - \frac{MSE}{Var(y)}$$

10.6. 회귀에 규제 적용

- **Ridge Regression (L2 Normalization)**

$$J(w)_{ridge} = \sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda\|w\|_2^2$$

$$L2: \lambda\|w\|_2^2 = \lambda\sum_{j=1}^{m} w_j^2$$

✓ λ 가 증가하면 규제 강도가 올라가면서 모델의 가중치값이(계수가) 감소

10.6. 회귀에 규제 적용

- **LASSO Regression (L1 Normalization)**

$$J(w)_{LASSO} = \sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda\|w\|_1$$

$$L1: \lambda\|w\|_1 = \lambda\sum_{j=1}^{m}\left|w_j\right|$$

✓ λ 가 증가하면 규제 강도가 올라가면서 특정 가중치가 0이 될 수 있음
✓ SVM Kernel 에서 기억나나요?

10.6. 회귀에 규제 적용

- **ElasticNet Regression (L1 + L2)**
  - ✓ L1과 L2의 절충안

$$J(w)_{ElasticNet} = \sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda_1\|w\|_2^2 + \lambda_2\|w\|_1$$

10.7. 다항 회귀

- 선형 회귀는 특성과 타깃이 선형 관계를 갖는다 가정
- 만약 선형이 아니면? 다항(Polynomial)을 이용

$$y = w_0 + w_1 x + w_2 x^2 + \ \ldots + w_d x^d$$

  - ✓ d는 다항식의 차수
- x^2 를 계산해서 column에 넣어준 뒤 회귀식에 사용 가능

- Feature engineering(존재하는 정보로부터 새로운 속성 제작)에 주로 사용

10.8.1. 결정 트리 회귀

- 분류에서 결정 트리를 사용할 때, 정보이득(IG)가 최대화되는 특성 분할 결정을 위해 불순도 지표로 엔트로피 정의

$$IG(D_p, x_i) = I(D_p) - \frac{N_{left}}{N_P} I(D_{left}) - \frac{N_{right}}{N_P} I(D_{right})$$

x: 분할이 수행될 특성, N_P: 부모 노드 샘플 개수, D_{left} , D_{right} : 분할 후 왼/오른쪽 자식의 훈련 샘플 집합

- 트리를 회귀에 사용하기 위해 불순도 지표를 MSE로 사용

$$I(t) = MSE(t) = \frac{1}{N}\sum_{i \in D_i}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

$\hat{y}^{(i)} = \frac{1}{N_i}\sum_{i \in D_i}^{n} y^{(i)}$ (예측된 타깃 값의 평균)

10.8.1. 결정 트리 회귀

- 결정 트리 회귀에서 MSE를 노드 내 분산(within-node variance)라고도 함
- 따라서 이 분할 기준을 분산 감소(variance reduction)이라고 부름

그림 10-14 결정 트리 회귀