

# **Data Science**

## **데이터 분석 과정 3-2**

**2021년 7월 6일 (2일간)**

**참고자료: DSAC Module3 교재**

# 내용

1. 머신러닝
2. kNN
3. 결정트리
4. 랜덤 포레스트
5. 서포트 벡터머신
6. 분류 성능
7. 특성공학
8. 모델 최적화
9. 이미지 분석
10. 텍스트 분석

# 분류 성능

# 분류의 손실 함수

- **분류**에서는 손실함수로 **MSE**를 **사용할 수 없다**
- 대신, 분류에서 **정확도**(accuracy)를 손실함수로 사용할 수 있다
  - 예) 100명에 대해 남녀 분류 문제
    - 96명을 맞추고 4명을 오 분류 : 정확도 0.96
  - 그러나 정확도를 손실함수로 사용하는 데에는 다음과 같은 문제가 있다
- **Category 분포 불균형**시 문제
  - 예)
    - Group : 남자 95명, 여자 5명
    - 오 분류 케이스 - 남자 1명, 여자 3명
    - 정확도는 여전히 0.96 :
    - 문제 : 여자의 경우, 5명 중 3명을 오 분류 → 결과 심각
  - 데이터 분포가 **비대칭**인 상황 : **질병 진단**의 경우 **자주 발생**
  - 손실을 제대로 측정하지 못함
  - 이를 보완하기 위해서 **크로스 엔트로피**(cross entropy)를 사용
    - Category가 둘 이상인 경우에도 동일한 개념으로 적용 가능

# 크로스 엔트로피(Cross Entropy)

$$CE = \sum_i p_i \log\left(\frac{1}{p_i}\right)$$

- $p_i$  : 어떤 사건이 일어날 실제 확률,  $p_i'$  : 예측한 확률

- 남녀가 50명씩 같은 경우

$$CE = -0.5 \times \log\left(\frac{49}{50}\right) - 0.5 \times \log\left(\frac{47}{50}\right) = 0.02687$$

- 남자가 95명 여자가 5명인 경우

$$CE = -0.95 \times \log\left(\frac{94}{95}\right) - 0.05 \times \log\left(\frac{2}{5}\right) = 0.17609$$

- 크로스 엔트로피 값이 작을수록 분류가 잘 수행된 것

# 대표적인 손실함수와 성능지표

	손실함수	성능 지표
정 의	손실함수를 줄이는 방향으로 <b>학습</b>	성능을 높이는 것이 머신러닝을 사용하는 <b>최종 목적</b>
회귀 모델	MSE (오차 자승의 평균)	$R^2$
분류 모델	크로스 엔트로피	정확도, 정밀도, 재현률, F1점수

# 분류 모델 성능

- 분석 모델이 얼마나 잘 동작하는지의 성능 평가 기준 필요
  - 예측의 정확도
  - 동작 속도
- 데이터 분석 프로젝트의 종료 기준으로 사전 정의

# 혼돈 매트릭스 - 다중 분류

**y\_pred**

**y\_true**

예측 클래스 0

예측 클래스 1

예측 클래스 2

정답  
클래스 0

정답 클래스가 0, 예측 클래스가 0인  
표본의 수

정답 클래스가 0, 예측 클래스가 1인  
표본의 수

정답 클래스가 0, 예측 클래스가 2인  
표본의 수

정답  
클래스 1

정답 클래스가 1, 예측 클래스가 0인  
표본의 수

정답 클래스가 1, 예측 클래스가 1인  
표본의 수

정답 클래스가 1, 예측 클래스가 2인  
표본의 수

정답  
클래스 2

정답 클래스가 2, 예측 클래스가 0인  
표본의 수

정답 클래스가 2, 예측 클래스가 1인  
표본의 수

정답 클래스가 2, 예측 클래스가 2인  
표본의 수



# 혼돈 매트릭스 – 이진 분류

- 혼돈 매트릭스(Confusion Matrix)
  - 분류의 결과가 잘 맞았는지를 평가하는 **채점표**와 유사
  - 오차 행렬로 번역하는 것이 더 타당함
- 결과 값이 P(Positive)또는 N(Negative) 둘 중 하나만 가질 수 있는 **binary 예측**의 경우를 설명하는 일반적인 용어
- Positive는 **찾고자 하는 현상**(ex. 암에 걸린 사실, 결함 등)이 나타난 것인지를 구분하는 것일 뿐, 긍정적인 결과를 찾았다는 뜻은 아님

실제 \ 예측	P로 예측	N로 예측
실제로 P	True positive (TP)	False negative (FN)
실제로 N	False positive (FP)	True negative (TN)

# 혼돈 매트릭스(Confusion Matrix)

- 용어의 의미 예시
  - True positive (TP)
    - 암/결함이라고 예측했는데 실제로 암에 걸린 경우
  - False positive (FP)
    - 암/결함이라고 예측했는데 실제로는 암에 걸리지 않은 경우
  - False negative (FN)
    - 암/결함이 아니라고 예측했는데 실제로는 암인 경우
  - True negative (TN)
    - 암/결함이 아니라고 예측했는데 실제로도 암이 아닌 경우

True: 예측이 맞음	Positive: positive로 예측
False: 예측이 틀림	Negative: negative로 예측

# 모델의 성능 지표 – 혼돈 매트릭스

- **정확도(accuracy)**: 정확하게 예측한 비율을 의미
  - $\text{accuracy} = (TP+TN) / \text{전체 경우의 수}(N)$

실제 / 예측	암(예측)	정상(예측)	합계
암환자(실제)	6 (TP)	4 (FN)	10
정상(실제)	2 (FP)	188 (TN)	190
합계	8	192	200

- 암진단 정확도 =  $(6 + 188)/200 = 194/200 = 0.97 \Rightarrow 97\%$
  - 오류율 =  $1 - \text{accuracy} = 0.03 \Rightarrow$  오진율은 3%
- **리콜/재현율(recall)**: 관심 대상을 얼마나 잘 찾아내는가
  - $\text{recall} = TP / (TP+FN)$
  - 실제 암 환자 발견률 =  $6 / (6+4) = 0.6 \Rightarrow 60\%$
- **정밀도(precision)**: 예측의 정확도
  - $\text{precision} = TP / (TP+FP) = 6 / (6+2) = 0.75 \Rightarrow 75\%$

# 모델의 성능 지표

- recall과 precision의 두 가지 지표를 동시에 높이는 것은 어려움,
- F1은 이러한 두 요소를 동시에 반영한 새로운 지표임
- F1은 recall과 precision의 조화 평균을 구한 것

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

- 두 지표의 값이 각각 0.5와 0.7일 때
  - 산술 평균  $c = (a+b)/2 = (0.5)+(0.7)/2 = 0.6$
  - 조화 평균  $c = 2ab/(a+b) = 0.7/1.2 = 0.58$
- 두 지표의 값이 각각 0.9와 0.3일 때
  - 산술 평균  $c = (a+b)/2 = (0.9)+(0.3)/2 = 0.6$
  - 조화 평균  $c = 2ab/(a+b) = 0.54/1.2 = 0.45$

# 조화 평균

$$\text{조화 평균: } \frac{1}{c} = \frac{\left(\frac{1}{a} + \frac{1}{b}\right)}{2}$$

$$c = \frac{2ab}{a+b}$$

# 비용 최소화 – 병원 사례

- 1)의 경우 수익이 발생하며 이를 편의상 200만원 (E1)
- 2)의 경우 병원은 약간의 손실이 생기며 이를 -3만원 (E2)
- 3)의 경우는 병원에 큰 손실이 생기며 이를 -500만원 (E3)
- 4)의 경우 환자를 더 유치하므로 평균 이득이 2만원 (E4)
- 각 경우의 발생 확률과 각 이득 또는 비용을 곱하여 더하면 총 기대치를 구할 수 있다

# 비용 최소화 – 병원 사례

- 확률

	암이라고 예측	암이 아니라고 예측
실제 암환자	$p1 = 6/200=0.03$	$p3 = 4/200=0.02$
실제로는 암 없음	$p2 = 2/200=0.01$	$p4 = 188/200=0.94$

- 기대치

	암이라고 예측	암이 아니라고 예측
실제 암환자	$E1 = 200\text{만원}$	$E3 = -500\text{만원}$
실제로는 암 없음	$E2 = -3\text{만원}$	$E4 = 2\text{만원}$

# 비용 최소화 – 병원 사례

- 전체 기대치 : 위 두 테이블을 항목별로 곱한 후 더한다

$$\begin{aligned}\text{전체 기대치} &= p_1E_1 + p_2E_2 + p_3E_3 + p_4E_4 \\ &= (0.03*200) + (0.01*(-3)) + (0.02*(-500)) + (0.94*2) = -7.55\text{만원}\end{aligned}$$

- 위의 의사는 암환자 진단 결과로 오히려 병원에 손실을 발생
- 이렇게 손실이 발생한 원인은 무엇일까?
  - 실제로는 암인데 암이 아니라고 잘못 판정한 경우  
(즉, FN의 댓가가 평균 -500만원으로 크기 때문)
  - 이 비용을 줄려면  $p_3$  확률을 줄이도록 노력해야 한다.
  - 조금만 의심이 들어도 모두 “암 같은데요” 라고 판정해 주면 FN을 줄일 수 있다.
  - 500만원 비용보다 -3만원 손실이 훨씬 적기 때문



# 비용 최소화 – 병원 사례

- 보수적인 의사

	암이라고 예측	암이 아니라고 예측
실제 암환자	10	0
실제로는 암 없음	90	100

실제 / 예측	암이라고 예측	암이 아니라고 예측
실제 암환자	$p1 = 10/200=0.05$	$p3 = 0/200=0$
실제로는 암 없음	$p2 = 90/200=0.45$	$p4 = 100/200=0.5$

$$\begin{aligned}
 \text{[전체 기대치]} &= p1E1 + p2E2 + p3E3 + p4E4 \\
 &= (0.05*200) + (0.45*(-3)) + 0 + (0.5*2) = 9.65\text{만원}
 \end{aligned}$$

# 비용 최소화 – 병원 사례

- 이 의사의 진단 능력을 종합해 보면
  - 정확도(accuracy) : 55%로 나쁜 편 (110/200)
  - 재현률 :  $10/10 = 100\%$  (10명의 암환자를 모두 찾음)
- 앞의 의사의 경우
  - 정확도(accuracy) : 97%
  - 재현률(recall)이 60%
- 평가
  - 암 환자를 찾아내는 비율(재현률)이 높아졌다
    - 따라서 오진으로 인해 병원이 지출할 비용도 줄여주었다.
  - 그러나 정밀도(precision)는  $10 / (10+90) = 5\%$ 이며 이는 앞의 의사의 75%에서 크게 감소했다. 즉, 암이라고 진단해도 그 중에 5%만 암이고 나머지는 과잉 진단
  - 과잉 진단은 장기적으로 병원과 의사의 신뢰도를 떨어뜨릴 수 있다
- 판정 기준은 병원의 철학과 전체 비용에 따라 다르게 설정될 것이다.

# 동적 성능 평가

- 정적 성능 평가
  - 최종 분류의 결과만 본다
- 동적 성능 평가
  - 정적인 성능 평가 보다 알고리즘의 동작을 좀 더 세밀하게 평가
  - 최종 분류 결과만 보는 것이 아니라 분류한 순서를 평가하는 방법
  - ROC, AUC

# 분류 순서 평가

환자번호	성별	점수	순위	실제 값
7	F	0.98	1	N
125	M	0.96	2	C
4	F	0.95	3	N
199	M	0.86	4	C
2	F	0.84	5	N
200	M	0.82	6	C
176	M	0.81	7	C
73	M	0.80	8	N
82	M	0.79	9	C
3	F	0.77	10	N
123	F	0.76	11	N
		...		C
43	F	0.48	198	N
93	M	0.42	199	N
120	F	0.40	200	N

점수 : 예) 암일 확률

↓  
보수적

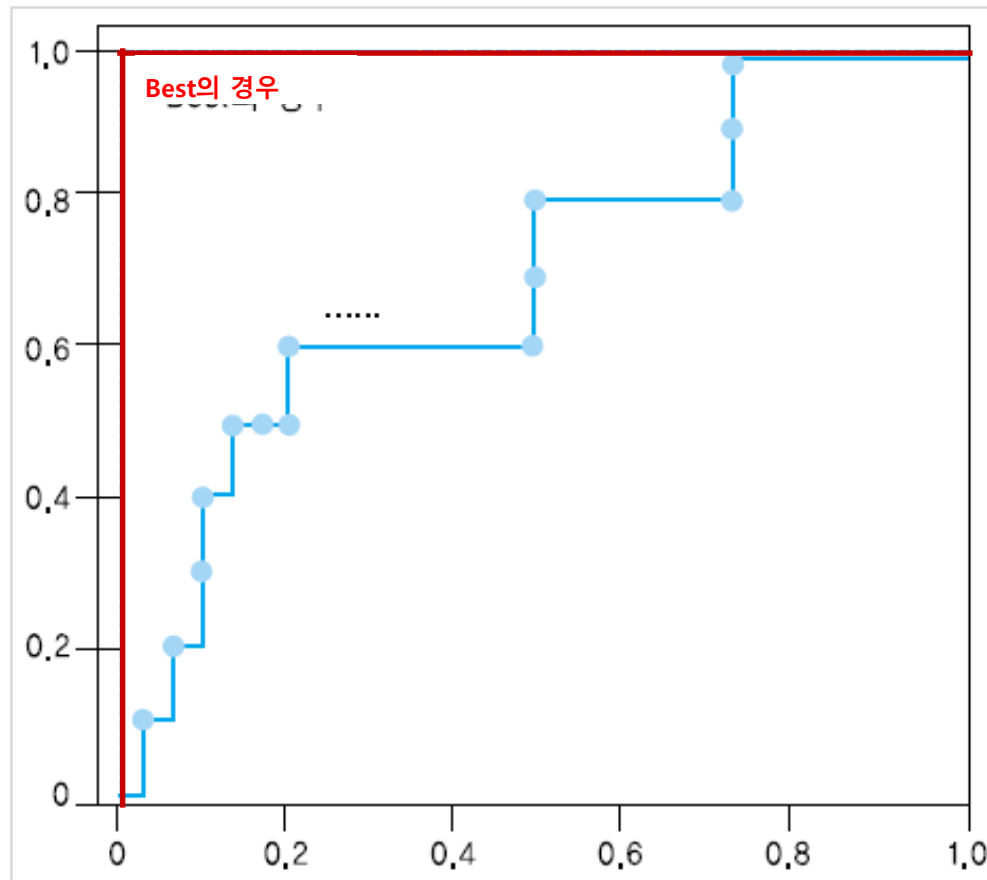
순서의 **어디까지**를  
양성이라고 판정할지에  
따라 **혼돈 매트릭스**가  
달라진다

# ROC (Receiver Operating Characteristic)

- 예측 결과를 순서대로 제시한 것이 실제 값과 얼마나 순서에 따라 잘 맞는지는 검증하는 2차원 그래프
- ROC 커브는 (0,0)점에서 시작하여 한 행씩 진행하면서 정답을 맞추었으면 y축 위로 한 칸 이동, 정답을 맞추지 못했으면 x축 방향으로 한 칸 이동  
· 종점은 (1, 1) 지점
- 그래프의 x 축으로는 예측 오류가 날 때마다 이동하고, y축으로는 정답을 맞출 때마다 이동
- x축은 예측이 틀린 것을 나타내므로 false positive rate, y축은 예측이 맞은 것을 나타내므로 true positive rate를 나타냄

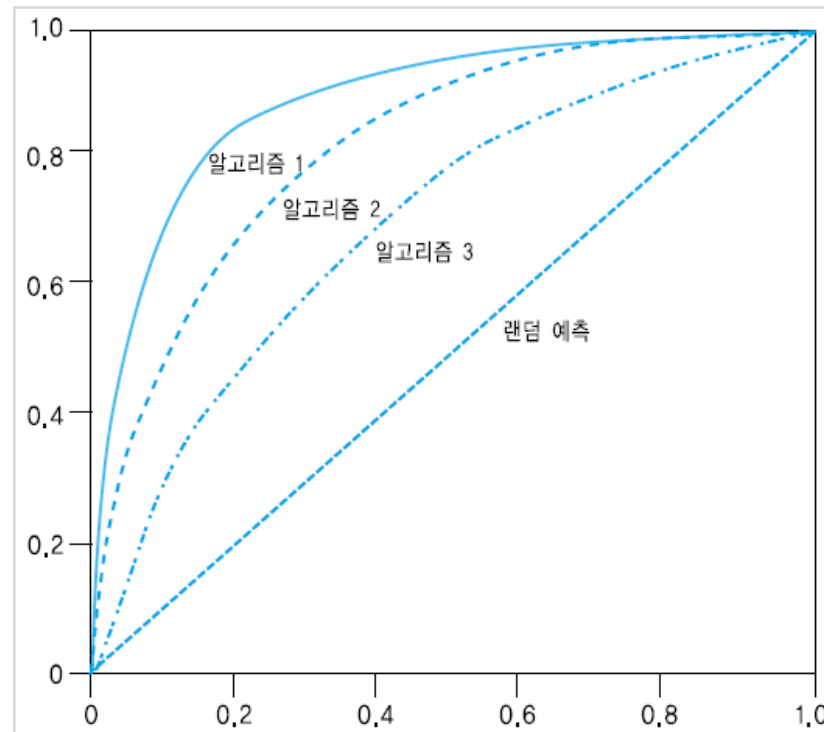
# 분류 순서 평가

- 각 데이터 항목에 대해 계단형 그래프가 만들어짐
- 만일 의사의 예측 정확도가 높다면, 그래프가 초반에 위로 올라갈 것임



# AUC (Area Under Curve)

- 예측 알고리즘의 성능을 간단히 수치로 나타내기 위해서 **ROC 그래프의 면적**을 계산하는 방법을 사용
- 우수한 알고리즘일수록 초반에 y축 상단 방향으로 이동하므로 ROC 커브의 면적이 넓어짐



- **분류성능 실습**

- <실습: gg-40-분류성능>
- 오차 행렬(혼돈 매트릭스)를 통한 정적인 성능 평가
- ROC 를 통한 동적인 성능 평가
- <실습: gg-41-분류성능비교\_포도주\_ROC>
- 포도주 품질 분류에 대해 성능 평가



# 다중 분류

- 다중 분류
  - 여러 개의 **category**를 가질 수 있는 입력 데이터를 분류
- sklearn의 이진분류 함수들은 다중 분류를 지원
  - 내부적으로 이진 분류를 확장해서 수행(해당 category인 것과 아닌 것)
  - 이러한 방식 : One-versus-Rest(**OvR**)방법
- 분류 결과만 알려면
  - **predict()**
- 다중 클래스 각각에 해당할 점수 또는 확률을 알려면
  - **decision\_function()**
  - **predict\_proba()**

# 다중 분류

- 3개 이상의 클래스 중에 하나를 예측해야 하는 경우
  - 로지스틱 회귀를 그대로 사용할 수 없다
  - 다항 로지스틱 회귀(multinomial logistic regression)를 이용
    - 다중 분류 (multiclass classification)
    - Softmax 함수 사용
- 이진 분류를 이용한 다중 분류 : OvR(One vs Rest) 방법
  - (예, A, B, C 분류)
    - A, {B, C}
    - B, {A, C}
    - C, {A, B}
- 한번에 다중 분류 가능 : 랜덤 포레스트, 나이브 베이즈 등
- 소프트맥스(softmax) 함수를 사용

$$\sigma(j) = \frac{\exp(\mathbf{w}_j^\top \mathbf{x})}{\sum_{k=1}^K \exp(\mathbf{w}_k^\top \mathbf{x})} = \frac{\exp(z_j)}{\sum_{k=1}^K \exp(z_k)}$$

# 소프트맥스 (Softmax)

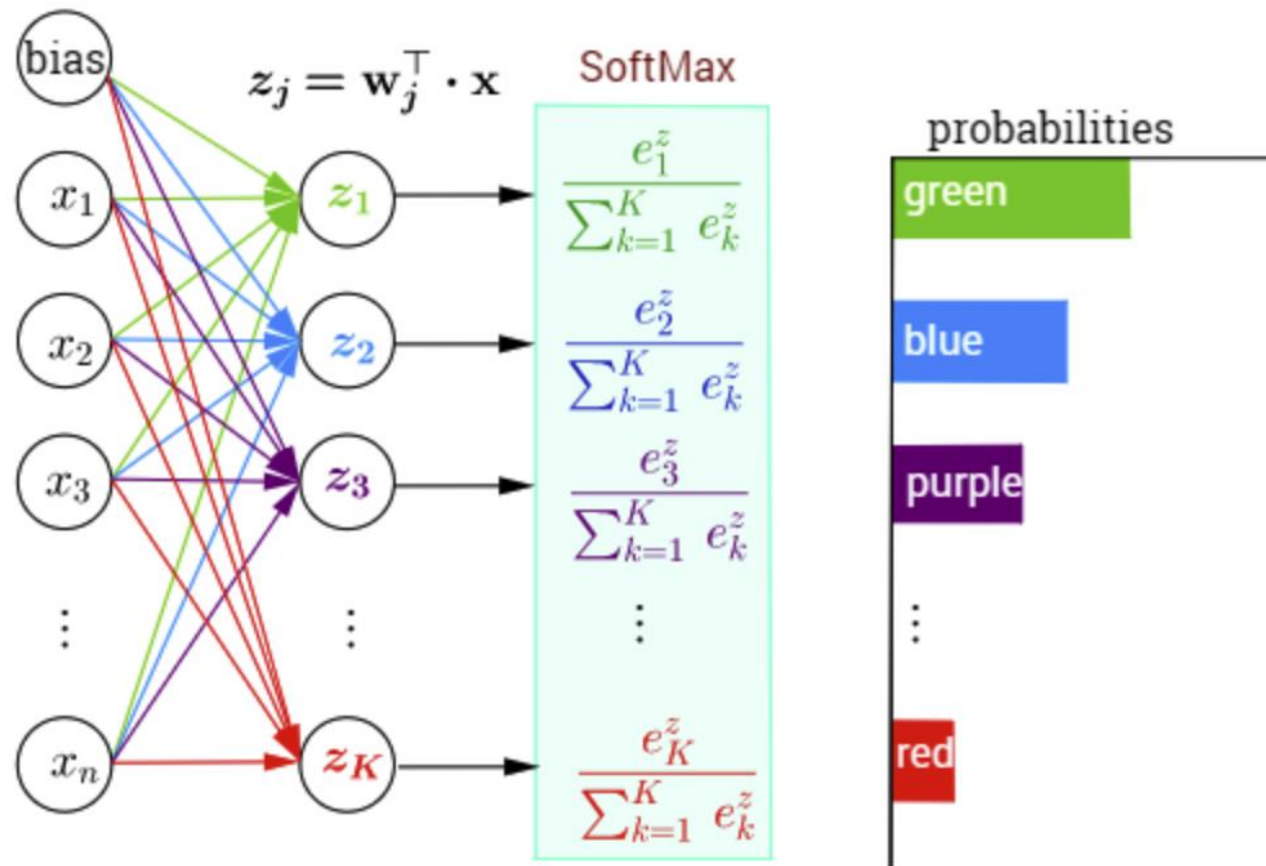
- 소프트맥스(softmax) 함수를 사용

$$\hat{p}_k = \sigma(s(x))_k = \frac{\exp(s_k(x))}{\sum_{j=1}^K \exp(s_j(x))}$$

- $\hat{p}_k$ : 클래스 k에 속할 확률
  - $x$ : 주어진 샘플
  - $S_k(x)$ : 소프트맥스 회귀 모델이 각 클래스k에 대한 점수
- 예) 얼굴을 보고 한국인, 중국인, 일본인 3 class로 구분
    - 어떤 샘플에 대해서 모델의 예측 값이 1.5, 2.0, 1.8 로 가정
    - 소프트맥스 적용 : 0.23, 0.43, 0.34 (합 = 1.0)
      - 각각의 점수를 확률처럼 사용 가능
      - 모델 예측 값이 음수(-)이어도, 소프트맥스 출력 값은 0~1 사이 값

# 소프트맥스

- 상대적인 점수 비교 : 확률처럼 0~1 사이 값으로 매핑



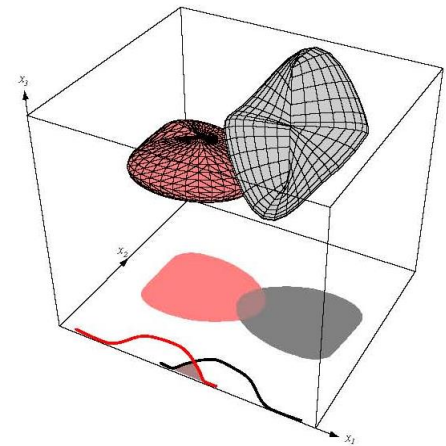
# 특성공학

# 차원 축소

- 기존 여러 특성 중 분석에 **가장 영향력이 있는 특성을 선택**하는 것이 필요한 때가 있다
  - 학생들의 학업 능력을 평가
    - 모든 과목의 성적을 다 사용하지 않고 **국어, 수학** 성적이 **대표성이 있다면** 이 두 과목의 성적만 평가 점수로 사용
- 특성 공학
  - 머신 러닝에서 사용할 **특성을 잘 선택**하는 것
  - 유효한 특성을 잘 선택하면 **학습 속도**가 빨라지고 **성능도** 좋아진다.
- 차원 축소
  - 머신 러닝의 **성능을 떨어뜨리지 않으면서 특성의 수를 줄이는** 기술
- 차원 축소가 필요한 이유
  - 계산량, 메모리 사용량을 줄이기 위해
  - 샘플의 특징을 보기 좋게 시각화하기 위해

# 차원의 저주

- 이론적으로 특성의 수가 증가할수록 더 좋은 성능을 가질 가능성이 큼
- 차원의 저주
  - 실제로는 특성의 수가 증가하면 더 나쁜 성능을 가짐
- 사례
  - 반지름이 1인 구/hypersphere 인 경우 차원이 커질수록 부피가 점점 작아짐
  - 즉, 구가 포함된 박스 안의 임의 지점을 샘플링하면 차원이 커질수록 대부분 지점이 구 바깥에 존재
  - 다르게 표현하면 차원이 커질수록 박스 안의 임의의 두 지점 간 평균거리가 점점 커짐
    - 1x1 4각형: 평균 거리 0.52m, 3차원큐브: 0.62m,
    - 1백만차원 하이퍼큐브: 408.25m
- 훈련 예제의 수는 차원에 따라 기하급수적으로 증가되어야 함



# 차원 축소

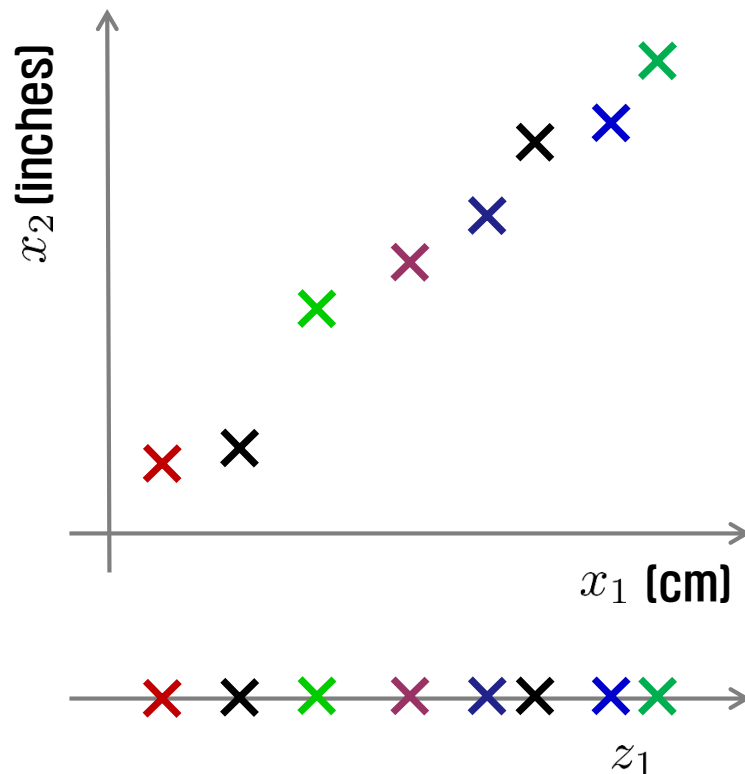
- 자동으로 영향력이 큰 특성 선택 방법
  - 목적 변수와 상관관계가 높은 변수를 선택
  - SelectPercentile()
    - 상관관계가 높은 순서대로 특성들을 나열하고
    - 상위 몇 %까지의 특성을 선택해 줌
- 머신 러닝에 별로 도움이 되지 않는 특성들을 제거함으로써 모델 개발 시간을 줄일 수 있다



# 차원 축소

- 차원 축소

- 고차원 데이터 지점을 저차원 지점으로 단순화하는 방법
- 차원의 저주를 극복하는 대표적인 방법
- 투영(Projection)과 매니폴드 학습(manifold learning) 등의 방식



2D에서 1D로 데이터  
축소

$$x^{(1)} \rightarrow z^{(1)}$$

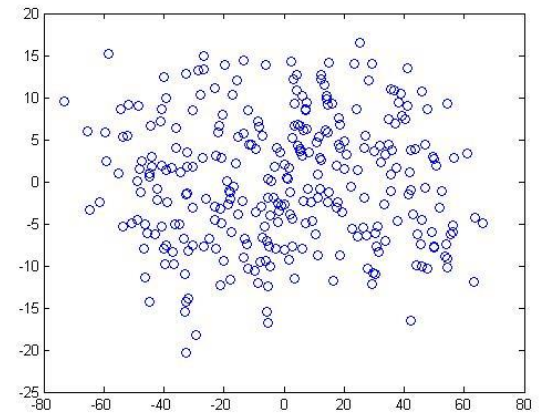
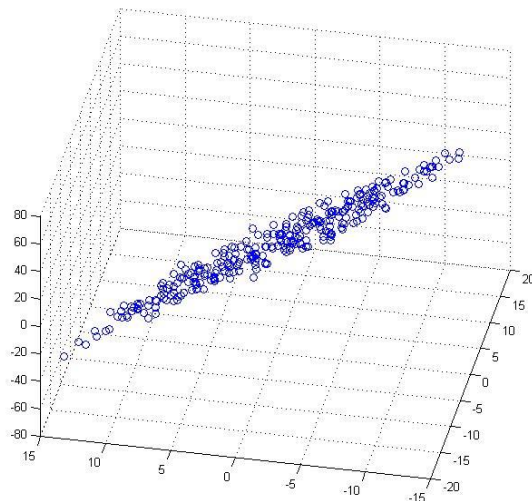
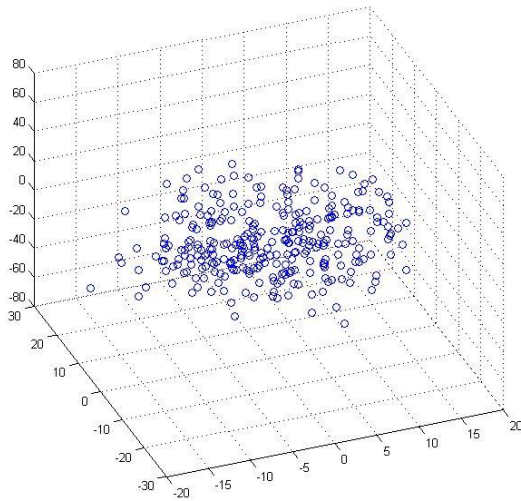
$$x^{(2)} \rightarrow z^{(2)}$$

$\vdots$

$$x^{(m)} \rightarrow z^{(m)}$$

# 차원 축소

- 투영
  - 예) 3D에서 2D로 데이터 축소

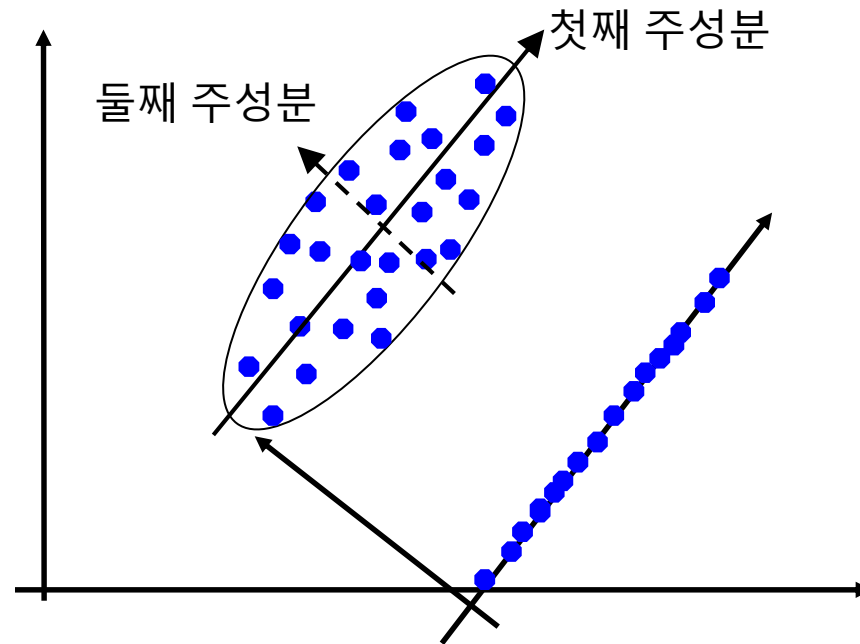


# 주성분 분석 (PCA)

- 주성분 분석 (Principal Component Analysis)
  - 여러 속성들을 조합하여 이들을 대표할 수 있는 적은 수의 특성을 찾아내는 작업
  - 기존 속성들의 선형 조합 : 가장 많이 사용
    - 예 : 학생의 능력 - 주성분 (수업 능력, 활동 지수)
    - 수업 능력 : 국어, 영어, 수학 성적(가중치 0.4, 0.3, 0.3)을 각각 곱해 더한다
    - 활동 지수 : 독서량, 운동량, 친구(가중치 0.4, 0.3, 0.3)을 각각 곱해 더한다
- 가장 적절한 주성분 찾기
  - 기존의 속성 값들을 어떤 비율로 가중 합산해야 할 지는 컴퓨터가 자동으로 여러 조합을 만들어 보고 최적의 조합을 찾아준다
- 최종적으로 필요한 주성분의 개수는 알려 주어야 한다.

# 주성분 분석 (PCA)

- 주성분 분석 (Principal Component Analysis)
  - 데이터 집합의 좌표 시스템을 새로운 좌표 시스템으로 변환
  - 가능한 한 많은 정보가 유지되도록 하는 투영 변환을 찾음
    - 공분산 행렬의 고유벡터 중 가장 큰 방향 벡터들이 주성분임
  - 특성들이 통계적으로 상관관계가 없도록 데이터 집합을 회전



# 주성분 분석 (PCA)

- 주성분 분석 순서

- 분산/정보를 가장 잘 유지하는 저차원 투영식을 구함

- 데이터 셋에 대한 평균 벡터와 공분산 행렬을 계산

- $\mu = \frac{1}{N} \sum_{j=1}^N x_j$

- $\Phi_i = x_i - \mu, \Phi = [\Phi_1 \Phi_2 \cdots \Phi_N]$

- $C = \frac{1}{N} \sum_{j=1}^N (x_i - \mu)(x_i - \mu)^T = \frac{1}{N} \Phi \Phi^T$

- 공분산 행렬에 대한 고유벡터와 고유값 계산

- $eig(C) = \{\lambda_1 > \lambda_2 > \cdots > \lambda_N\}, eigvec(C) = \{c_1, c_2, \cdots, c_N\}$

- 고유벡터는 SVD 를 통해서도 구할 수 있음

- $C = U \Sigma V^T, V^T = (c_1 \ c_2 \ \cdots \ c_n)$

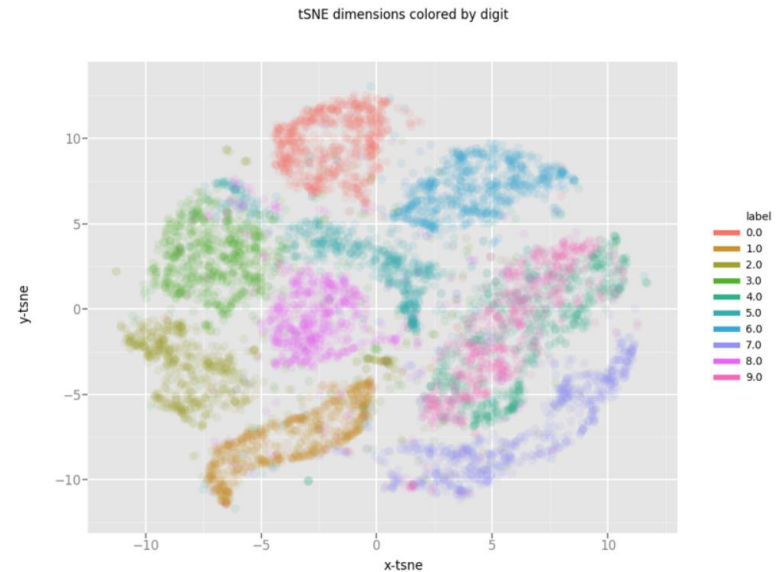
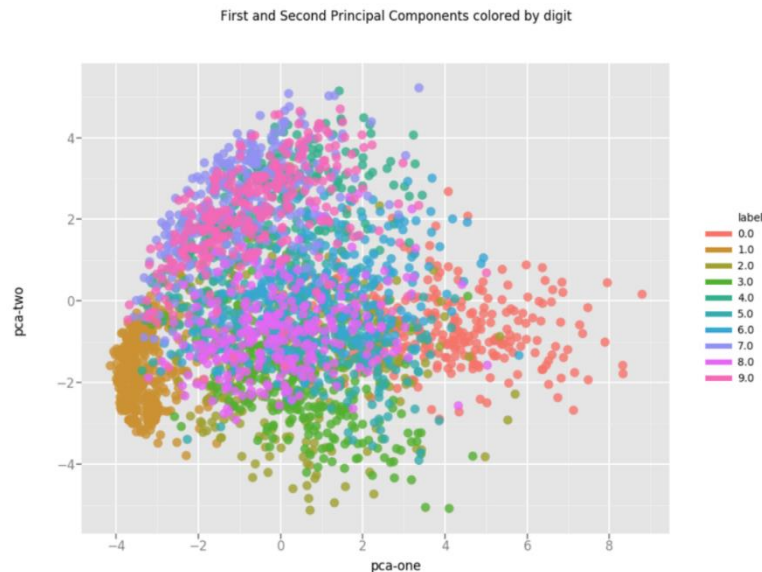
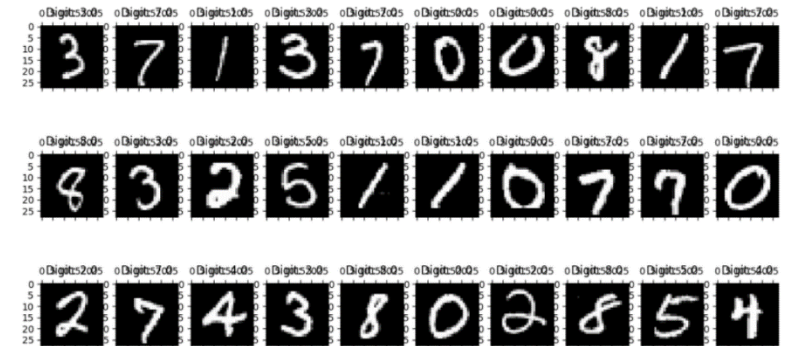
- 가장 큰 고유벡터들을 선택하고 이들로 구성된 부공간으로 투영

- $y = A(x - \mu)$

# t-SNE (Stochastic Neighbor Embedding)

- 데이터의 특성을 한 눈에 파악하는데는 시각화가 매우 유용
  - 그러나 실제 세계의 데이터는 속성의 차원이 높으므로 이러한 다차원 공간에 속성들의 관계를 그리는 것은 불가능
  - 명확한 시각화를 위해서는 데이터를 2, 3차원 이하로 변환 필요
    - 비지도 학습
- t-SNE
  - 시각화를 위해 고차원의 특성을 가진 데이터를 저차원으로 축소하는 기술
  - 데이터 지점 사이의 거리를 가장 잘 보존하는 2차원 표현을 찾기 위해서 각 데이터 지점을 2차원에 무작위로 표현한 후 원본 특성 공간에서 가까운 지점은 가깝게, 먼 지점은 멀어지게 함. 데이터 지점 간의 거리를 확률로 변환
  - 보통 word2vec으로 임베딩한 단어벡터를 시각화하는 데 많이 사용
  - 단, 계산 시간이 많이 걸림

- MNIST 데이터 30개
  - 28x28의 차원(즉, 784 차원)의 데이터
  - 2차원으로 축소 : PCA, t-SNE



- **주성분분석 및 tSNE 실습**
  - <실습: gg-42-주성분분석\_tSNE\_숫자>
  - 숫자 이미지에 대한 주성분분석 및 tSNE 적용
  - <실습: gg-42-주성분분석\_tSNE\_유방암>
  - 유방암분류 문제에 대해 주성분분석 및 tSNE 적용
  - <실습: gg-42-커널PCA\_스위스롤>
  - 스위스롤 데이터에 대해 커널PCA 적용



# 모델 최적화

# 릿지 규제

- 모델을 일반화 하는 방법으로, 손실함수로 MSE 항목과 함께 **계수의 크기 자체도 줄이도록** 하는 방법을 도입한 것

$$J(W) = MSE(W) + \alpha \frac{1}{2} \sum_{i=1}^n W_i^2$$

- 계수의 자승의 합을 손실함수에 추가
  - 계수 자체가 **가능하면 작은 값**이 되기를 원한다
  - $\alpha$  : 이 축 항목의 비중을 얼마나 크게 할지를 정하는 하이퍼파라미터
- 이는 **여러 계수들**을 가능한 **골고루 반영**하라는 의미
  - 왜냐하면 계수의 자승의 합을 줄이려면 각 계수의 크기를 줄여야 전체 자승의 합이 최소화되기 때문

# 릿지 규제

- 그러나 릿지 규제를 너무 강하게 하면 MSE 항은 무시되고, 모든 계수의 값이 동일하게 된다.
- 릿지 규제는 선형회귀에서만 사용되는 것이 아니라 SVM, 신경망 등 다른 머신러닝 모델에서도 사용될 수 있다.
- 릿지 규제는 L2 규제라고도 부른다

# 라쏘 규제

- 라쏘 규제에서는 모든 계수의 **절대치들의 합**을 추가로 더하는 방법 (자승을 취하지 않음)

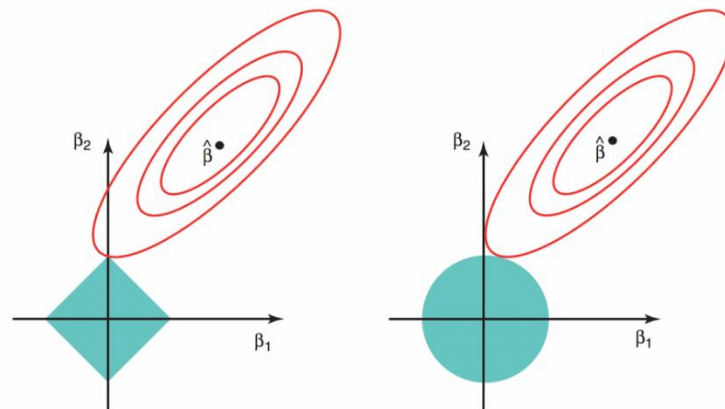
$$J(W) = MSE(W) + \alpha \sum_{i=1}^n |W_i|$$

- 릿지 규제와 유사한 것처럼 보이지만 사실은 **반대의 효과**
- **릿지 규제**에서는 특별히 비중이 큰 계수를 지양하고, **가능한 여러 가중치**를 **골고루 반영**하는 효과
- **라쏘 규제**를 적용하면 **절대값이 작은 계수가 먼저 사라지는 효과**
  - 라쏘 규제는 **L1 규제**라고도 함

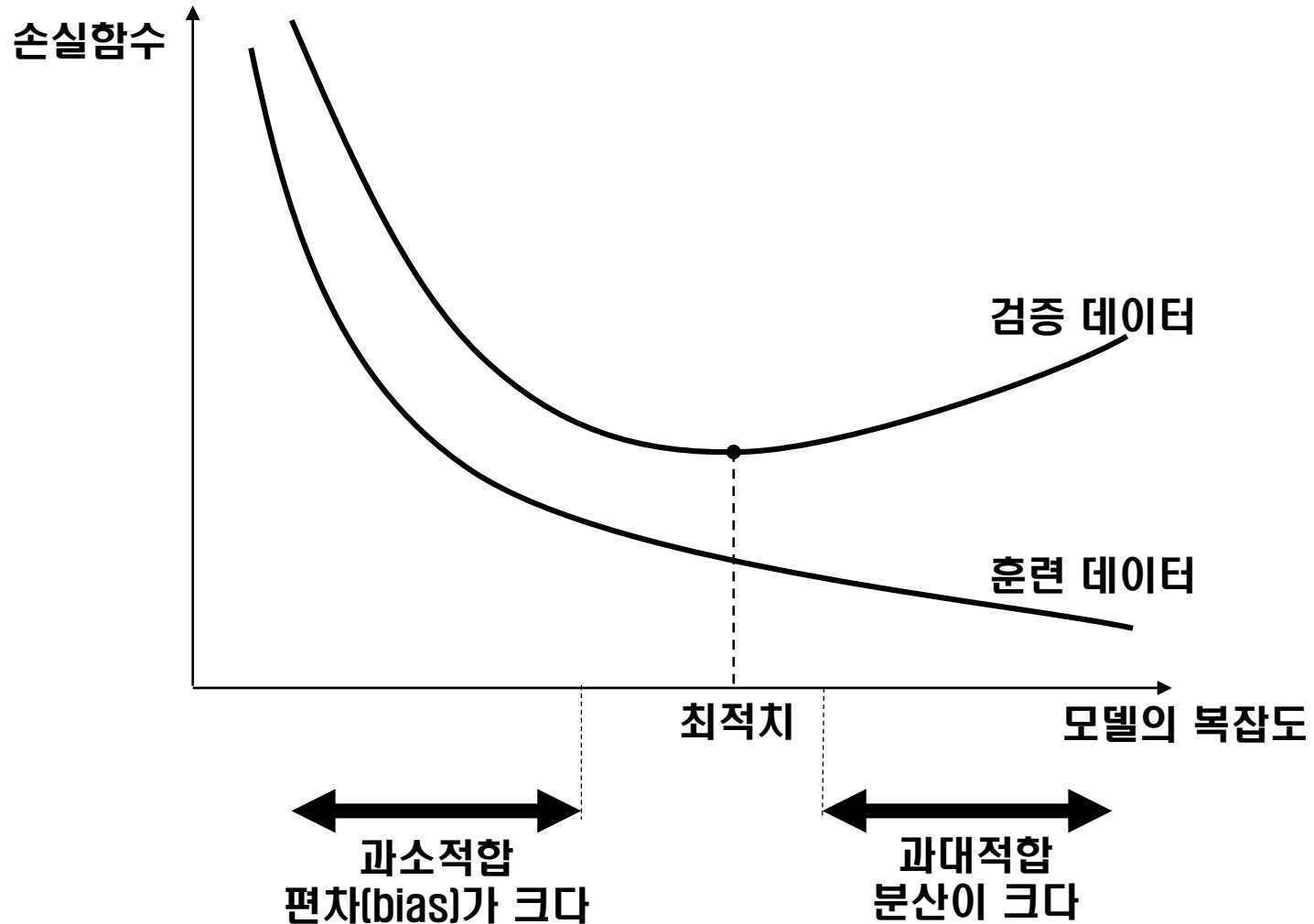
# 엘라스틱 넷

- 릿지 규제와 라쏘 규제가 동시에 필요한 경우가 있다.
  - 특정 계수가 크게 영향을 주는 것을 피하고 싶고(L2 규제)
  - 동시에 영향력이 적은 계수의 수를 줄일 필요가 있다(L1 규제)
- 아래에서
  - $\alpha$ : 일반화의 정도를 조정하는 하이퍼파라미터
  - $\lambda$ : L2와 L1 규제의 반영 비중을 조절하는 하이퍼파라미터

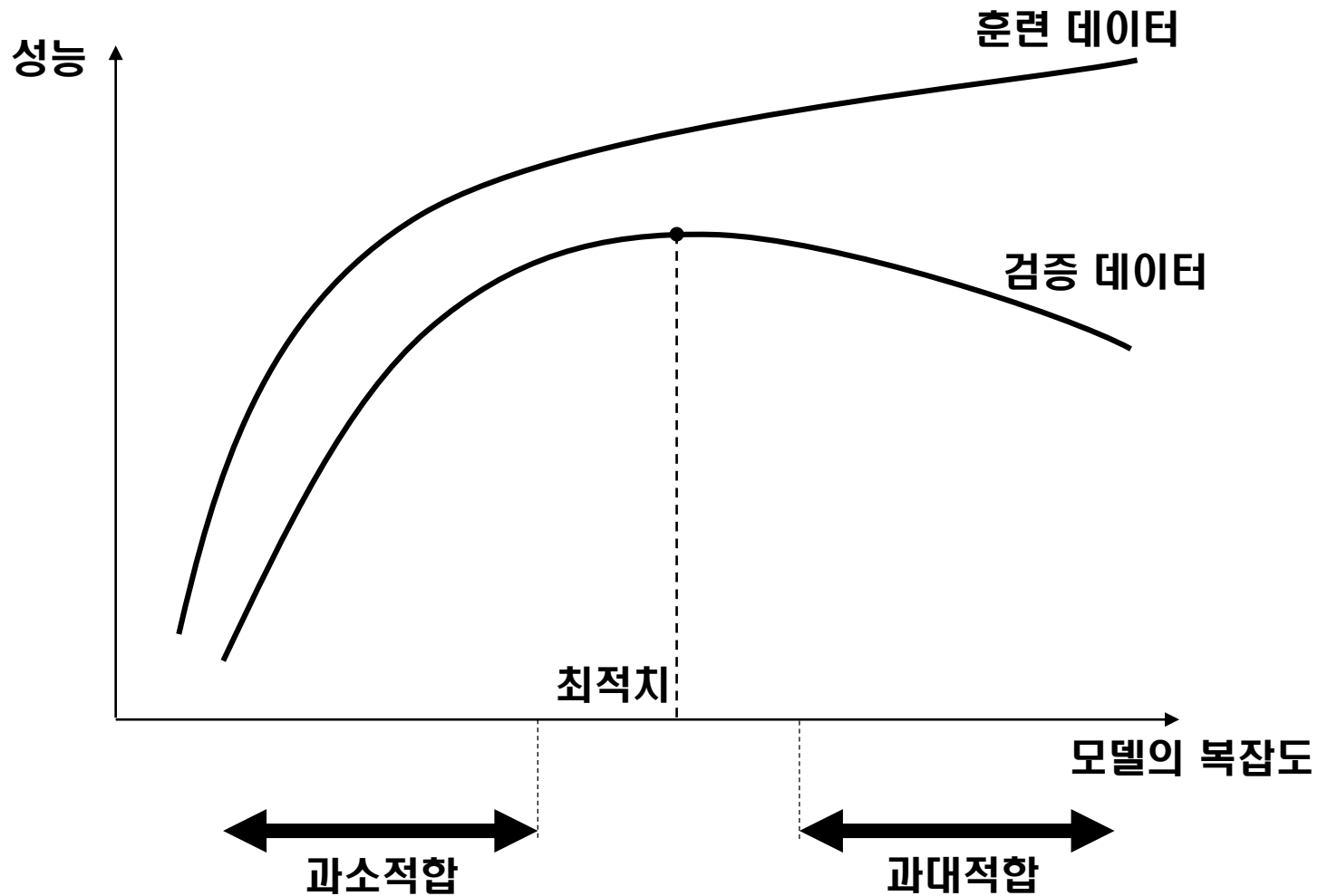
$$J(\theta) = MSE(\theta) + \gamma \alpha \sum_{i=1}^n |\theta_i| + \frac{1-\gamma}{2} \alpha \sum_{i=1}^n \theta_i^2$$



# 과소 적합과 과대적합 판단 - 손실함수



# 과소 적합과 과대 적합 판단 - 성능

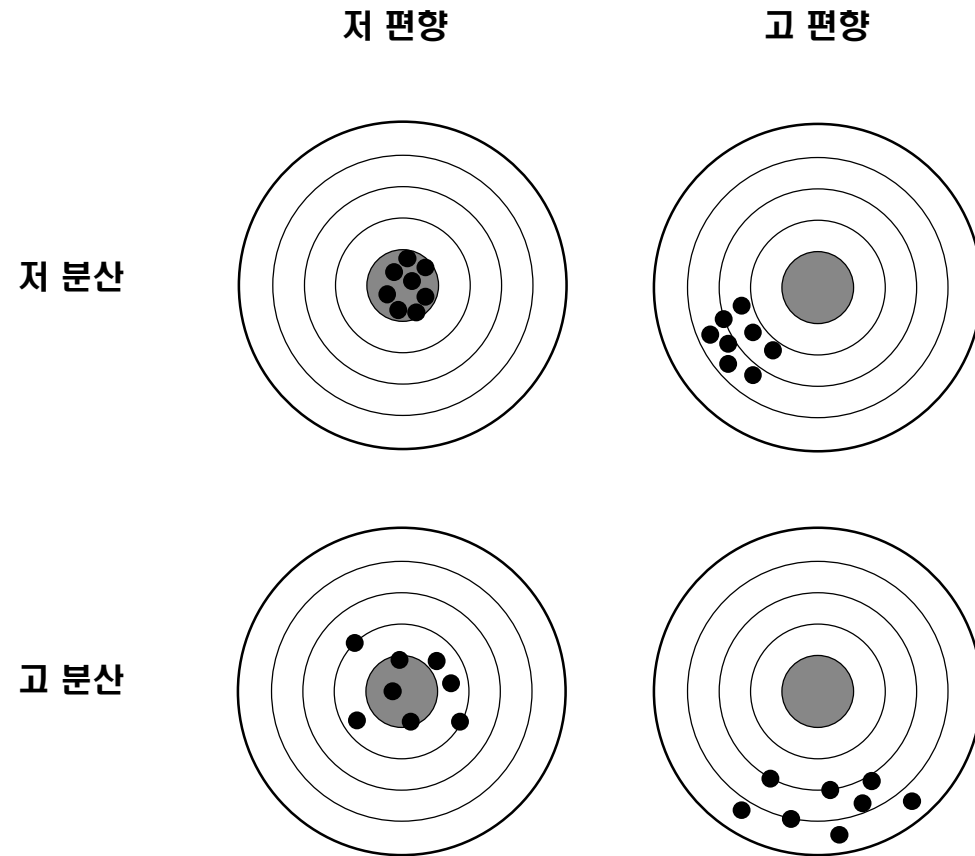


# 편향과 분산

- 예측 모델에서 발생하는 오차는 **분산**(variance)과 **편향**(bias) 두 가지 성분으로 설명할 수 있다.
- **분산**이란 모델이 너무 복잡하거나 학습데이터 민감하게 반응하여 **예측 값이 산발적**으로 나타나는 것이다.
- **편향**이란 **모델 자체가 부정확**하여 피할 수 없이 발생하는 오차를 말한다.



# 편향과 분산



- 선형모델규제 실습
  - 〈실습: gg-43-선형모델규제〉
  - 슈퍼마켓 매출을 예측하는 선형모델 비교
  - 릿지 및 라쏘 규제 실습

# 이미지 분석

# 영상 분석 개요

- 영상 분석

- 컴퓨터를 사용하여 영상 데이터를 분석하여 중요한 의미를 추출하거나 미시각 기능을 가진 장치를 만드는 기술

- 분야

- 영상 처리 (Processing)
- 검출 (Detection)
- 분류 (Classification)
- 인식 (Recognition)

# 영상 처리

- 영상을 입력으로 받아 **처리**하여 새로운 영상을 **출력**
- 필터링
- 이진화
- 모폴로지
- 기하학적 변환
- 영상 변환
- 영상 분할

# 컴퓨터 비전 처리 과정

- 전처리
  - 주로 영상 처리
- 특징 추출
  - 에지, 선분, 영역, 텍스처, 지역 특징 등을 검출하고 특징 벡터 추출
- 해석
  - 응용에 따라 다양한 형태

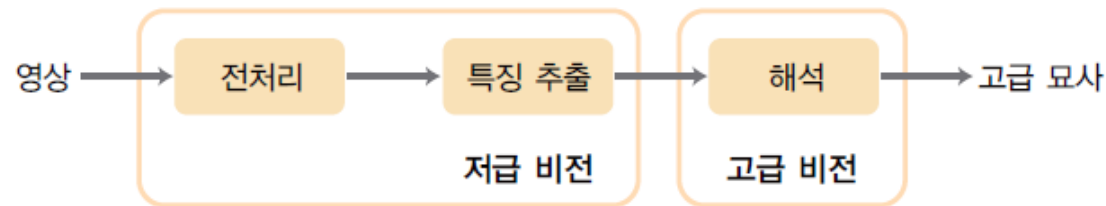
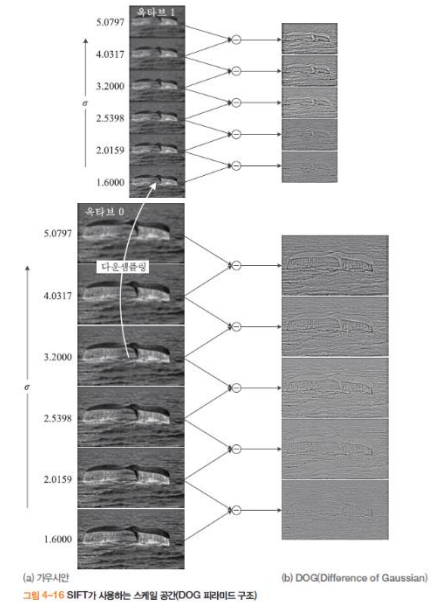


그림 1-6 컴퓨터 비전의 계층적 처리

# 검출

- 영상을 입력으로 받아 **특징** 또는 **객체**를 **검출**
- **특징 검출**
  - 에지 검출
  - 지역 특징 검출
- **특징 기술**
  - 관심점 기술자 - SIFT, SURF
  - 영역 기술자 - 모멘트, 모양, 푸리에 기술자
  - 텍스처, 지역 관계 기술자 - LBP



136	165	147	133	139	125
105	150	142	143	163	140
113	153	160	176	177	140
160	186	204	200	177	139
184	178	188	188	166	148
147	139	146	148	140	136

그림 6-17 LBP 계산

0	1	0
0		0
0	1	1

$$00110010 = 2 + 16 + 32 = 50$$

# 영상 분류

- 영상을 입력으로 받아 **객체**를 **분류**
- 유형별 분류
  - 문자인식
  - 딥러닝을 활용한 영상 객체 분류
- 범주 분류
- 대표적인 컴퓨터비전 대회
  - PASCAL VOC([http://host.robots.ox.ac.uk/pascal\\_voc/](http://host.robots.ox.ac.uk/pascal_voc/))
    - 분류(20),검출,분할에 대한 5가지 문제
    - 2012년 종료
  - ImageNet ILSVRC(<http://www.image-net.org/>)
    - 1000종류 분류 문제
    - 2017년 인간의 성능을 능가하면서 종료
  - ICDAR RRC
    - 자연 영상에서 텍스트 검출



(a) 사례 인식



(b) 범주 인식

그림 9-2 사례 인식과 범주 인식



# 영상 인식

- 영상을 입력으로 받아 **객체**를 감지하고 분류하고 **인식**

- 객체 탐지**

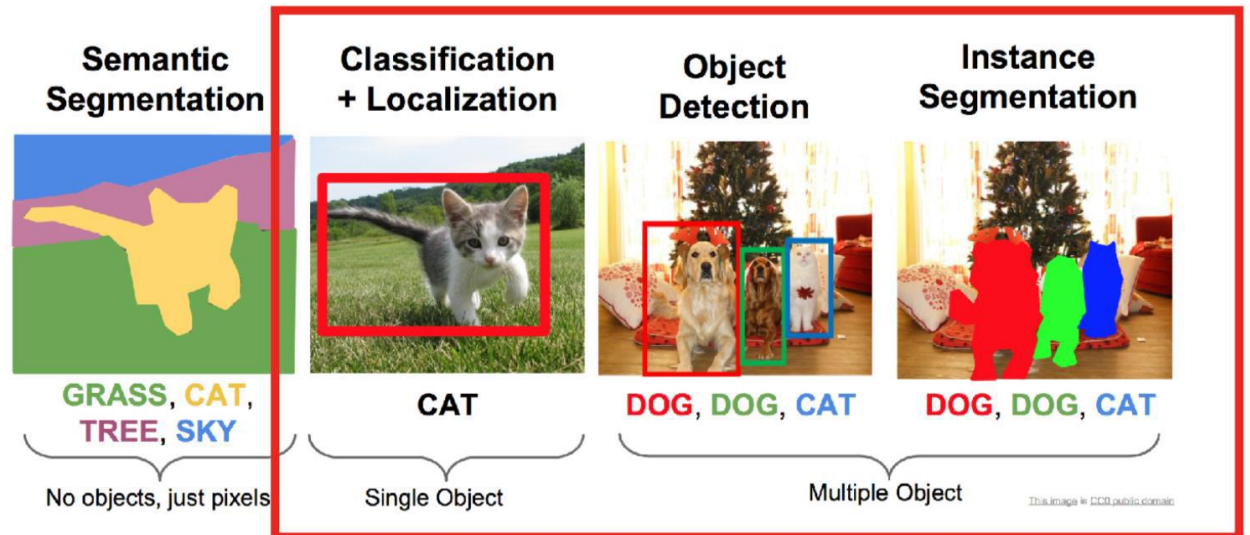
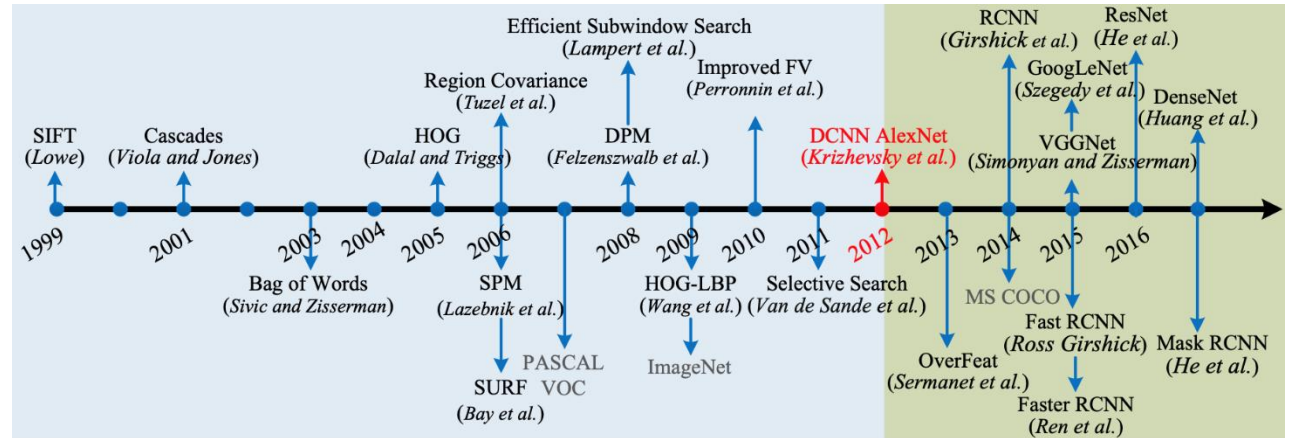
- 사물 및 위치 인식
- 얼굴인식

- 영상 분할**

- 의미론적 분할
- 객체 분할

- 3D 객체 인식**

- Objectron



# OpenCV 개요

- **OpenCV**
  - 인텔 주도로 개발된 컴퓨터 비전용 라이브러리
  - 폰 소스 소프트웨어 (BSD 라이선스)
  - 공식 웹 사이트는 <http://opencv.org/>이다.
- **지원 기능**
  - OpenCV는 영상처리, 얼굴 인식, 물체 감지, 비디오 캡처 및 분석, 딥러닝 프레임워크인 TensorFlow, Torch/PyTorch, Caffe도 지원하고 있다.
  - 인텔의 데이터 중심 연산 최적화 라이브러리인 IPP(Integrated Performance Primitives)가 설치되어 있으면 활용
- **주요 기능**
  - 영상 파일의 읽기 및 쓰기
  - 비디오 캡처 및 저장
  - 영상 처리(필터, 변환)
  - 영상이나 비디오에서 얼굴, 눈, 자동차와 같은 특정 물체를 감지
  - 비디오를 분석하여 움직임을 추정하고, 배경을 없애고, 특정 물체를 추적
  - 기계 학습 알고리즘을 사용하여 물체 인식 가능

# OpenCV 개요

- 라이브러리 구성

- 현재 4.5 버전까지 개발
- 기본 기능

패키지명	기능	패키지명	기능
core	핵심 기능	ml	기계 학습
imgproc	이미지 처리	flann	다차원 클러스터링 및 탐색
highgui	고급 GUI 및 미디어 I/O	gpu	GPU 가속
video	비디오 분석	photo	연산 사진술(photography)
calib3d	카메라 캘리브레이션 및 3D 재구성	stitching	이미지 스티칭(붙이기)
features2d	2D 특징 추출 프레임워크	gapi	그래프 API
objdetect	객체 탐지	nonfree	유료 기능
dnn	심층 신경망	contrib	기증된/실험적인 기능

- 추가 기능 : aruco(마커 탐지), cnn\_3dobj(3D 객체인식및자세추정),cudaxx(가속기능), rgbd(RGB-깊이처리), tracking(객체추적), viz(3D시각화) 등

# OpenCV 설치

- **아나콘다 환경에서 OpenCV 설치**
  - `conda install OpenCV` 를 입력하여 설치
- **코랩 환경에서 OpenCV 설치**
  - `pip install opencv-python` 입력하여 설치
- **설치 확인**
  - 파이썬 셸(Python shell)에서 아래의 명령을 입력하여 OpenCV버전을 확인
  - `>>> import cv2`
  - `>>> print(cv2.__version__)`

```
(cm) C:\Users\cm>python
Python 3.6.10 |Anaconda, Inc.| (default, May 7 2020, 19:46:08) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> print(cv2.__version__)
3.4.4
>>>
```

- **파이썬 예제 참고자료**
  - <https://wjddy66.github.io/categories/#opencv>

# 주요 함수

- 이미지 읽기 : `cv2.imread()`
  - 컬러 이미지를 읽으려면 `cv2.IMREAD_COLOR`를, 그레이스케일 모드로 읽으려면 `cv2.IMREAD_GRAYSCALE`을 입력
- 이미지 컬러 공간 변환 : `cv2.cvtColor()`
  - BGR 색공간→gray-scale로 변환 : `cv2.COLOR_BGR2GRAY`,
  - BGR 색공간→HSV 색공간으로 변환 : `cv2.COLOR_BGR2HSV`를 선택
- 이미지 사이즈 변경 : `cv2.resize()`
  - 이미지를 축소시킬 때는 보간법(interpolation) 설정을 `cv2.INTER_AREA`로, 확대시킬 때는 `cv2.INTER_CUBIC` 또는 `cv2.INTER_LINEAR`를 사용
- 이미지 보여주기 : `cv2.imshow()`
  - 보통 `cv2.waitKey(0)`, `cv2.destroyAllWindows()`와 함께 사용
- 이미지 저장 : `cv2.imwrite()`
- 선/사각형/원/타원 그리기 : `cv2.line()`, `cv2.rectangle()`, `cv2.circle()`, `cv2.ellipse()`
- 텍스트 넣기 : `cv2.putText()`
- 이미지 채널 분리 및 병합 : `cv2.split()`, `cv2.merge()`
- 값의 범위 바꿔주기 : `cv2.normalize()`

- **OpenCV 기본 동작 실습**

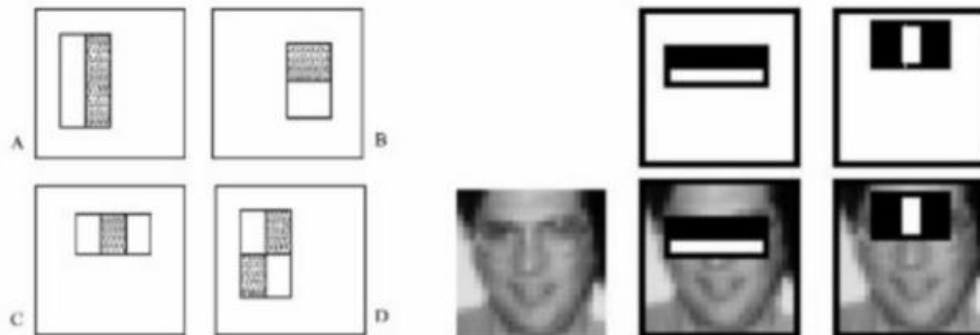
- <실습:gg-44-opencv\_intro>

- **얼굴인식**

- 최근 딥러닝 기술이 아니라 OpenCV가 제공하는 패턴 인식 기술 활용
    - 사람 얼굴에는 공통된 특징이 있어서 모든 사람의 눈,코,입 부분의 명암의 매우 유사한 패턴을 가지고 있다.
    - 이러한 얼굴 고유의 특징을 데이터베이스화하여 사람의 얼굴을 이미지에서 추출하는 방법으로 캐스케이드 파일을 이용할 수 있다.
    - haarcascade 검색하면 찾을 수 있으며 아래는 Haar-like 특징 학습기를 다운로드 받는 방법이다.

# 얼굴 인식 (openCV)

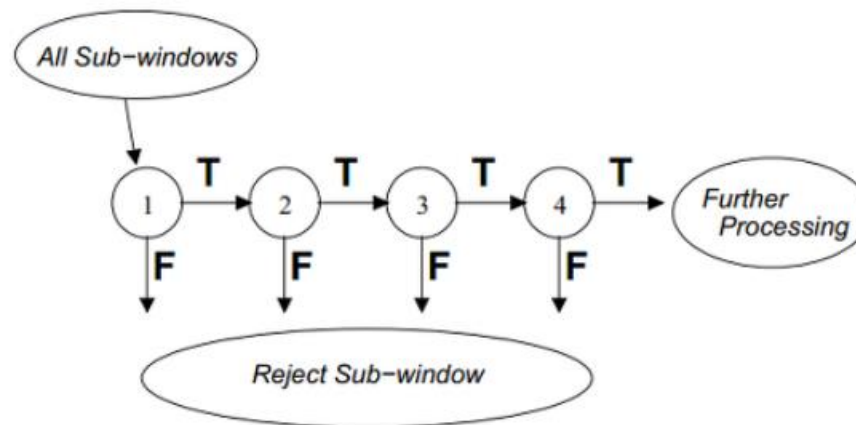
- Haar-like 특징 학습을 이용한 얼굴 인식
  - 사람 얼굴에는 공통된 특징이 있다
    - 예) 모든 사람의 눈, 코, 입 부분의 명암이 매우 유사한 패턴
  - 3종류의 특징 사용
    - A, B(Two-rectangle feature)
    - C(Three-rectangle feature)
    - D(four-rectangle feature)
  - 입력영상에서 A,B,C,D와 같은 Sub-Window를 Sliding시켜 탐색
  - 영상에서 검정 부분과 흰색 부분의 밝기 값을 빼서 임계값 이상인 것을 찾는다.
  - 이때 Sub-Window내의 feature의 크기와 모양은 다양할 수 있다.
    - 단, sliding window(검정부분과 흰색부분의 크기와 모양)은 동일



# 얼굴 인식 (openCV)

- Cascade 방식

- 사람 얼굴 고유의 특징을 database화하여 이미지에서 추출하는 방법 사용
- 연산이 복잡한 강 분류기를 모든 영역에 적용시키는 것은 비효율적
  - Object는 입력 영상의 일부분일 뿐 모든 부분에 퍼져 있지 않다는 가정
  - 예, 단체 사진에서 사람의 얼굴이 사진에서 차지하는 비율은 극히 일부분
- 전체 특징을 여러 분류기 단계로 그룹화, 단계별로 하나씩 적용
  - 처음 몇 단계에는 매우 적은 수의 특징을 포함
  - 해당 단계에서 실패하면 나머지 특징에 대한 검출은 포기하고 바로 다음 창으로 이동
  - 통과하면 기능의 두 번째 단계를 적용하고 해당 특징들을 검출





# 얼굴 인식 (openCV)

- Haar-like 특징 학습을 이용한 얼굴 인식
  - 단점
    - 영상의 밝기 값을 이용하기 때문에 조명,대조비에 많이 영향
    - sample의 양이나 질에 따라 성능에 크게 영향

- **OpenCV 예제 실습**
  - **<실습:gg-45-opencv\_digit>**
  - **손글씨 숫자 인식**
    - 8x8 픽셀 숫자 이미지를 데이터로 SVM 분류기 학습하고 학습한 파라미터를 피클 형식 파일로 저장
    - 임의의 손글씨 숫자 이미지에 대해 인식 결과 확인
  - **<실습:gg-46-object\_detect>**
  - **윤곽선 검출**
    - 윤곽선 추출을 위해 그레이로 스케일링한 후 블러링을 통해 노이즈 제거
    - 이진화 수행

# 텍스트 분석

# 텍스트 분석의 개요

- 텍스트 분석의 목적
  - 텍스트의 의미를 알아내는 것
    - 글의 목적
    - 글쓴이의 성향(찬성/반대)
    - 기분(기쁨/슬픔/우울함 등)
    - 제품 피드백 등
- 텍스트 자체는 대표적인 비정형 데이터
- 의미를 추출하려면 비정형 데이터에서 정형화된 정보를 먼저 얻어야 함
- 텍스트 구문을 분석하여 의미를 파악하고 이것을 정량적으로 측정함

# 텍스트 분석

- SNS(트위터, 블로그, 페이스북 북 등) 글을 분석
  - 소비자들의 반응, 감성, 트렌드를 파악
  - 개인별 마케팅, 상품 피드백을 분석하는데 사용
- 이메일, 웹사이트 댓글, 신문기사, 콜센터 상담기록, 도서 등을 분석
  - 글의 주요 내용을 파악
  - 문서의 특징을 추출
  - 유사한 글이나 저자를 찾는 작업 등을 수행
- 참고문헌이나 본문 인용의 관계를 통해서 문서간의 연계성, 전문가들의 인적 네트워크 등을 파악하는데도 사용
- 인공지능 스피커, 챗봇 등에서도 기본적으로 텍스트 분석이 필요

# 텍스트 분석 응용

- 챗봇 (Chatbot)
  - 사람과 대화하듯이 음성, 키보드 입력으로 **대화**를 나누는 인공지능 서비스
  - 챗봇의 유형
    - 미리 답을 준비하여 관련 질문이 나오면 해당 답을 하는 간단한 방식 (저수준)
    - 신경망을 사용하여 최적의 답을 찾아주는 방식 (고수준)
- QA 시스템
  - 질문을 하면 **검색**을 통해 **적절한 답**을 찾아주는(대답) 서비스
    - 대한민국의 수도는?
    - 오늘 날씨는?
- 자연어 처리
  - **언어 모델**을 사용
    - 가장 자연스러운 다음 문장을 완성
    - 문장을 번역
    - 문서 요약, 주제 분석, 감성 분석 등을 수행

# 텍스트 표현 방법

- 사람이 단어나 문장의 의미를 인식하듯이 컴퓨터가 단어 자체 의미를 직접 파악할 수는 없다
- 텍스트 데이터 처리
  - 대표적인 비정형 데이터
  - 먼저 비정형 데이터인 글자로부터 정형화된 데이터인 수치 데이터로 변환
- 토큰화(tokenize) : 텍스트 분석의 첫 단계
  - 컴퓨터가 다루는 텍스트의 단위 : 토큰
    - 단어 (word) or 글자(character)
  - 주어진 텍스트를 토큰으로 나누는 작업

# 코퍼스 (말 뭉치)

- **말뭉치**(corpus)
  - 데이터 분석에 주어진 전체 문서 집합
- **문서**(document)
  - 코퍼스 내의 한 단위의 텍스트
  - 예) **하나의 블로그**는 **문서**이고, 분석할 대상 블로그가 1천개이면 이 **1천개 블로그 집합**이 **말뭉치**
- **파싱**(parsing)
  - 코퍼스에서 **의미 있는 단어**를 **추출**하는 작업



- 토큰화 단위 (크게 3가지)
  - 단어(word)
    - 사람이 말을 이해할 때, 단어 단위로 인식하기 때문에 많은 연구에서 선호
  - 글자(character)
  - n-gram
- 단어 단위로 정보를 표현하는 과정에서 많은 정보를 잃게 된다
  - “정말” , “정말로” , “정말은” 등 단어
    - 같은 단어로 취급, 아니면 각각 다른 단어로 처리할지에 따라 분석 결과 달라짐
    - 같은 단어로 취급하기 위해 단어를 어근(stem)으로 변환하면 어미 변화를 무시하거나 조사를 무시하게 되어 텍스트에 들어 있던 정보를 잃게 된다
- 일반적으로 단어의 종류는 보통 10만 단어 이상 (언어마다 상이)
  - 신조어, 특수한 단어 포함하면 수십만개로 확대

- **글자** 단위로 토큰화를 하면 어근으로 변환할 때 정보를 잃는 문제를 피할 수 있다.
  - “정” , “말 “ , “로” , “은 “ 등
- **음절** 단위 토큰의 수
  - 영어
    - 음절단위의 토큰의 수가 매우 적다 : 알파벳이 26글자
  - 한글
    - 음절의 수가 수천 가지 이상

# 토큰화 – n-gram

- n-gram
  - n개의 연속된 단어를 하나로 취급하는 방법
- 예를 들어 “러시아 월드컵”이라는 표현을 “러시아”와 “월드컵” 두 개의 독립된 단어로만 취급하지 않고 두 단어로 구성된 하나의 토큰으로 취급
  - n=2 경우, bi-gram
  - 단어의 수가 매우 크게 증가
    - 실제로는 빈도 수가 최소한 몇 개 이상인 것만 다룬다

# 토큰화 – n-gram [예]

**텍스트:** “어제 러시아에 갔다가 러시아 월드컵을 관람했다”

**단어토큰:** { “어제” , “러시아” , “갔다” , “러시아” , “월드컵” ,  
“관람” }

**2-gram 토큰:** { “어제 러시아” , “러시아 갔다” , “갔다 러시아” ,  
“러시아 월드컵” , “월드컵 관람” }

# 토큰화

- n-gram을 허용하면 토큰화 대상의 수가 **매우 크게 증가**
  - 이론적으로는 10만개의 단어를 두 개 붙여서 나올 수 있는 경우의 수는 10만의 자승이 된다.
- 실제로는 **빈도수**가 최소한 **몇 개 이상**인 것만을 다룬다.
- 토큰화한 결과를 **수치**로 만드는 방법
  - 원핫(one-hot) 인코딩
  - BOW(단어모음)
  - 단어벡터(Word Vector) 방법

# 원 핫 (One-hot) 인코딩

- 원 핫 인코딩

- 토큰에 고유 번호를 배정
- 모든 고유번호 위치의 한 컬럼만 1, 나머지 컬럼은 0인 벡터로 표시

텍스트: “어제 러시아에 갔다가 러시아 월드컵을 관람했다”

토큰 사전: { “어제” :0, “러시아” :1, “갔다” :2, “월드컵” :3, “관람” :4}

원핫 인코딩:

어제 = [1, 0, 0, 0, 0]

러시아 = [0, 1, 0, 0, 0]

갔다 = [0, 0, 1, 0, 0]

월드컵 = [0, 0, 0, 1, 0]

관람 = [0, 0, 0, 0, 1]

# BOW (Bag of Word, 단어 모음)

- 원형 인코딩 방식으로 단어(토큰)을 표현하면
  - 단어의 수가 적을 때에는 문제가 안되지만
  - 단어가 모두 10만개이면
    - 모든 단어가 항목이 10만개인 (0과 1로 구성된) 벡터로 표시
  - 주어진 텍스트가 20개의 단어로 구성되어 있다면
    - 20 x 100,000개 크기의 벡터가 필요
- 텍스트 분석은 “문장” 을 단위로 하는 경우가 많다
- 단어 모음(BOW) 방식 : 한 문장을 하나의 벡터로 만드는 방법
  - 한 문장을 단어 사전 크기의 벡터로 표현하고 그 문장에 들어 있는 단어의 컬럼만 1로, 단어가 없는 컬럼은 모두 0으로 표현
- 먼저 단어 사전을 만들고 각 문장에 어떤 단어가 들어 있는지 조사하여 해당 컬럼만 1로, 나머지는 0으로 코딩

- 단어 사전: { “어제” :0, “오늘” :1, “미국” :2, “러시아” :3, “갔다” :4, “축구” :5, “월드컵” :6, “올림픽” :7, “관람” :8, “나는” :9, ..., “중국” :4999 }
- Text\_1: “어제 러시아에 갔다가 러시아 월드컵을 관람했다” 를

## BOW로 표현하면

문장번호	0	1	2	3	4	5	6	7	8	9	10	...	4998	4999
Text_1	1	0	0	1	1	0	1	0	1	0	0	0	0	0
Text_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Text_3	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Text_4	0	0	0	0	0	0	0	1	0	0	0	0	1	0
...														
Text_50	0	0	0	0	0	0	1	0	0	0	0	0	0	0



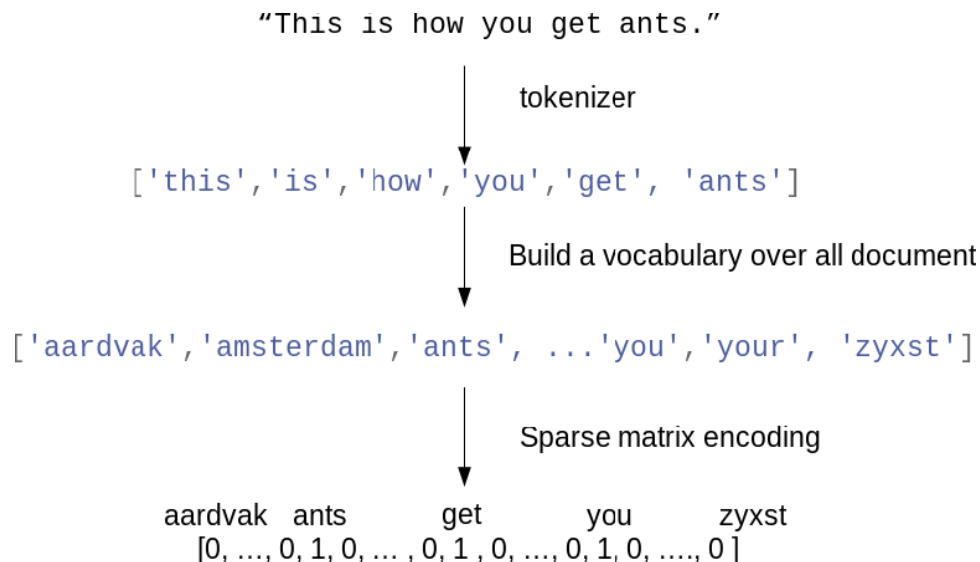
- **문서-단어(document-term) 행렬**
  - 문장 단위로 어떤 단어들이 있는지를 나타내는 **BOW**의 **확장**
  - **문서(document)** 단위로 어떤 단어들이 있는지를 표현
  - 같은 단어가 **여러번 등장**하면 **1 이상의 값**을 갖는다

문서번호	0	1	2	3	4	5	6	7	8	9	10	...	4998	4999
Doc_1	1	2	3	1	4	0	2	0	1	3	0	0	0	0
Doc_2	0	0	0	0	2	0	0	0	0	0	0	0	2	0
Doc_3	0	0	0	1	0	0	0	0	3	0	0	0	0	1
Doc_4	4	0	0	0	0	0	0	1	0	0	4	0	1	0
...														
Doc_100	0	2	0	0	0	0	1	4	0	1	0	0	0	0

# BOW (Bag of Word, 단어 모음)

- BOW(Bag of Words)

- 머신러닝에서 텍스트 데이터를 표현하는 방법으로 널리 쓰임
- 전체 말뭉치(corpus)에 대한 BOW 표현 계산
  - 토큰화 – 각 문서를 문서에 포함된 단어(토큰)로 나눔
  - 어휘 사전 구축 – 모든 문서에 나타난 모든 단어 목록(알파벳순)
  - 인코딩 – 어휘 사전의 단어가 문서마다 몇 번 나타나는지 셈



- term frequency-inverse document frequency
  - **tf** : 단어가 각 문서에서 발생한 빈도
  - **df**(document frequency) : 그 단어가 등장한 ‘문서’ 의 빈도
- 적은 문서에서 발견될수록 가치 있는 정보
- 많은 문서에 등장하는 단어일수록 일반적인 단어
  - 이러한 공통적인 단어는 tf가 크다고 하여도 비중을 낮추어야 분석이 제대로 이루어질 수 있다.
- 따라서 단어가 특정 문서에만 나타나는 희소성을 반영하기 위해서 **idf(df의 역수)**를 tf에 곱한 값을 tf 대신 사용
  - $tfidf(w, d) = tf \left( \log \left( \frac{N+1}{N_w+1} \right) + 1 \right)$
  - Tf: 단어 w가 대상 문서 d에 나타난 횟수
  - N: 훈련 세트에 있는 문서 수,  $N_w$ : 단어 w가 포함된 훈련세트 문서 수
- Sklearn에서는 Tfidf 변환을 위한 2개의 클래스 제공
  - TfidfTransformer – countvectorizer가 만든 희소행렬을 변환
  - TfidfVectorizer – 텍스트 데이터를 입력받아 BOW 특성 추출 및 tfidf 변환 수행
  - 둘 다 적용 후 L2 정규화를 적용하므로 데이터 크기에 상관없음

- BOW 예제 실습

- <실습:gg-51-뉴스기사>
- 뉴스 기사를 활용하여 문서단어행렬 제작
  - 국내언론사들의 뉴스기사 데이터를 읽고 CountVectorizer를 이용하여 문서-단어 행렬을 저장하고 빈도 수 확인
  - TfidfVectorizer 로 문서-단어 행렬 저장
- <실습:gg-52-스팸분류>
- 영어 문자메시지 중에서 스팸 및 정상 메시지 구분
  - 영어 문자 데이터를 읽고 tfidf를 구하고 다양한 머신러닝 알고리즘을 통해 스팸메일 분류 및 비교분석
- <실습:gg-53-네이버영화평점>
- 네이버 영화 평점 데이터를 이용한 감성 분석
  - 네이버 영화 평점 데이터를 읽고 konlpy 패키지를 이용하여 한글 형태소 분석 후 로지스틱 회귀를 통해 분류

# 단어 임베딩

# 단어 임베딩의 정의

- 앞에서 소개한 세 가지 텍스트 코딩 방식인 원핫 인코딩, BOW(단어모음), 문서-단어 행렬방식은 **단어**마다 **고유번호**를 **배정**하여 사용
- 그러나 이 고유 번호 숫자에는 **아무런 의미가 들어 있지 못하며 단지 인덱스의 성격만** 갖는다.
- 단어를 인덱싱이 아니라, **의미 있는 숫자**들의 **집합**, 즉, **벡터**로 표현하는 방법이 **단어 임베딩** (Word Embedding)이다.

# 단어 벡터

- 단어 벡터

- 각 단어를 50~300개 정도의 차원으로 구성된 벡터로 표현

학교 = [0.23, 0.58, 0.97, ... , 0.87, 0.95]

바다 = [0.45, 0.37, 0.81, ... , 0.22, 0.64]

- 단어 벡터를 사용하면

- 각 단어들 사이의 “거리” 를 계산이 가능
- 거리를 기반으로 유의어/반대어 등을 찾아낼 수 있다
- 동물의 성별, 단수/복수, 동사/명사를 구분할 수도 있다
- 그러나 각 벡터 값의 의미는 알 수 없다

# 단어 벡터

- 단어 벡터는 **대형 말뭉치로부터 학습**
  - 말뭉치의 문장들을 계속 입력하여 학습을 시키면 단어 벡터를 얻을 수 있다
  - 예를 들어 음식과 관련된 다음과 같은 문장들로 학습을 시키면 다음과 같은 단어 벡터를 얻을 수 있을 것이다.
  - 학습에 사용된 문장 예:

“나는 어제 바나나를 맛있게 먹었다”

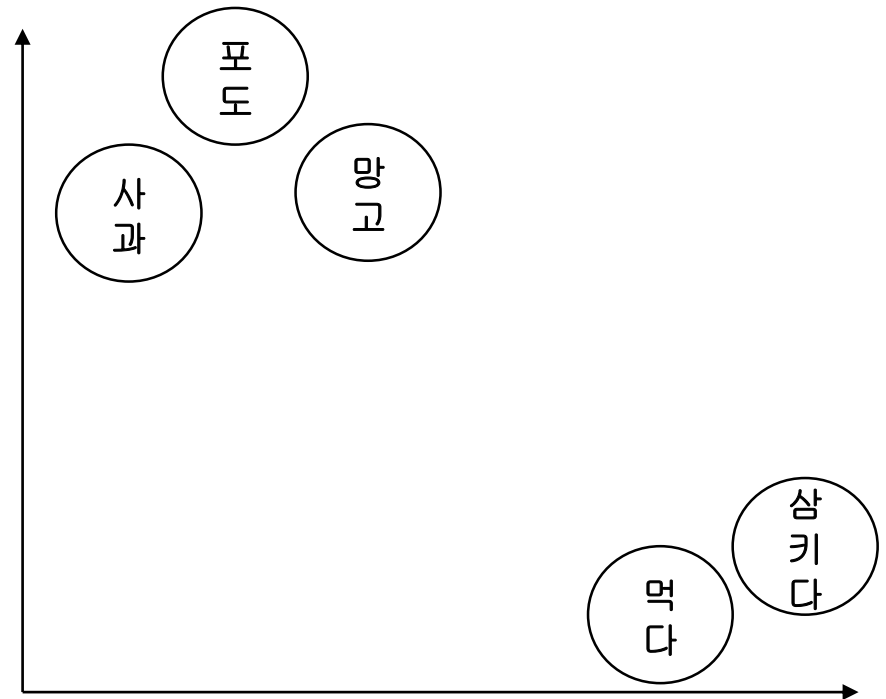
“이 망고는 먹기가 힘들다”

“이 사과는 씹는 맛이 아주 좋다”

“바나나가 사과보다 맛있다”

“잘 씹어야 맛있게 먹을 수 있다”

...



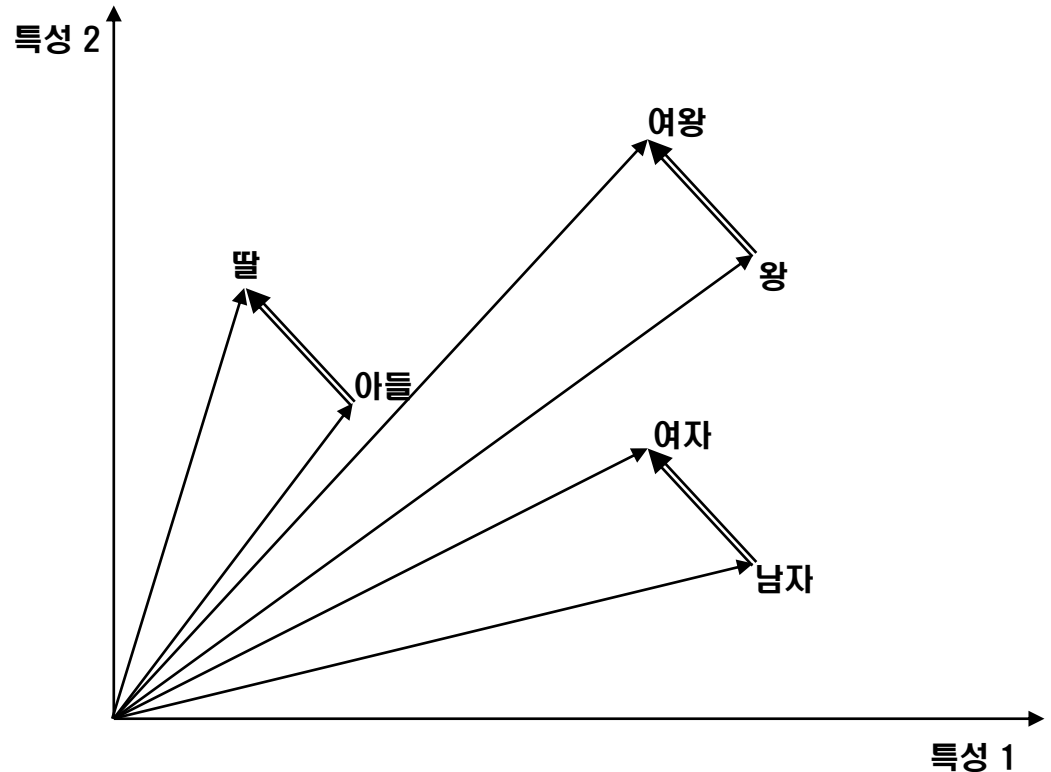


# 단어 벡터

- 이미 만들어져 있는 단어 벡터를 가져다 사용할 수도 있다.
- **glove**
  - 2014년 스탠포드에서 만든 Global Vectors for Word Representations
  - 위키피디아 데이터로부터 학습
  - 40만개 단어를 100차원으로 임베딩
  - [nlp.stanford.edu/projects/glove](http://nlp.stanford.edu/projects/glove) 에서 다운로드

# 단어 벡터

- 단어 벡터를 사용한  $A:B = C: ?$ 의 관계를 만족하는 ?를 찾을 수 있다.
  - 왕 : 여왕 = 아들 : ?  $\rightarrow$  ? 부분 : 딸
  - 이러한 연산은  $(B-A)$  벡터, 즉 (왕 - 여왕) 성분을 구한 후 이를 벡터 C(아들)에 더하면 딸을 구할 수 있다.
  - 이들의 관계는 아래와 같다.



# 단어 벡터

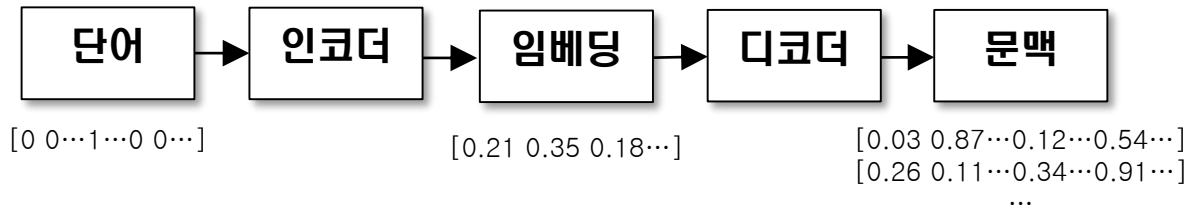
- 학습 방법

- 한 단어  $w$  가 주어졌을 때 그 주변 단어가 존재하면 0, 그렇지 않으면 1인 원-핫 인코딩을 생성하여 학습

- 구현 방법

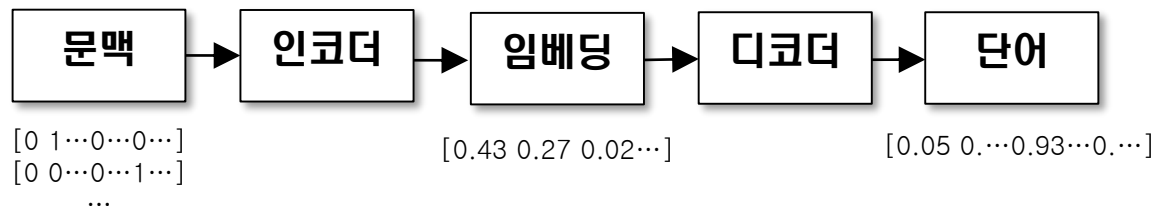
- 스킵-그램(skip-gram)

- 단어( $x_i$ )에 대해 주변에 나타날 확률이 높은 단어들(context)을 구함, 즉  $p(\text{context}|x_i)$  를 구함



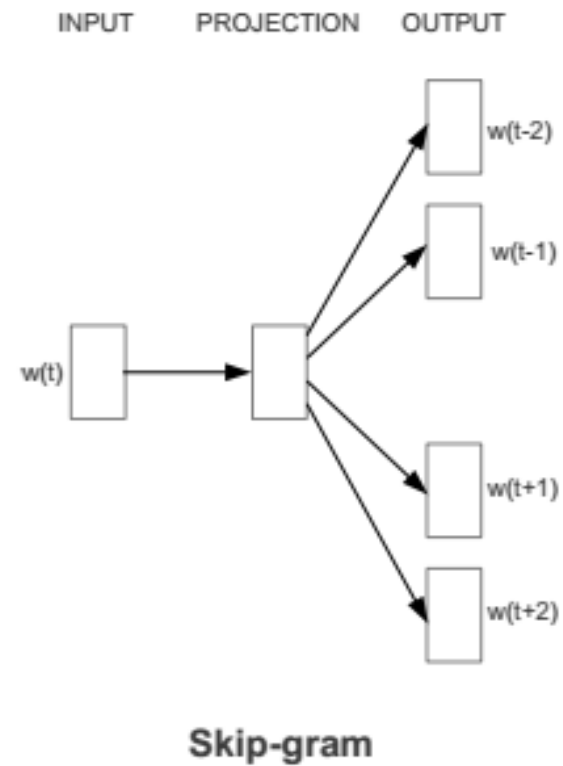
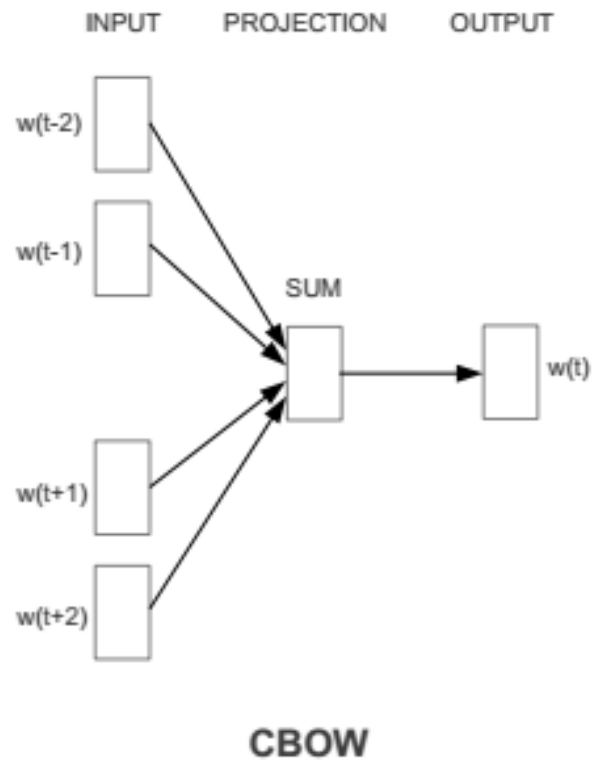
- CBOW(Continuous bag-of-words)

- 주변 단어(context)를 받아 그 단어들과 같이 나타날 확률이 높은 단어( $x_i$ )을 구함, 즉  $p(x_i | \text{context})$



# 단어 벡터

- 구조
  - 신경망으로 분산표현을 학습하여 모델 구현



# 네이버 영화 평점 분석

- **네이버 영화 평점 데이터를 이용한 감성분석 방법을 소개**
  - **문서-단어 행렬을 이용**
  - **데이터**
    - Naver sentiment movie corpus v1.0 ( [github.com/e9t/nsmc/](https://github.com/e9t/nsmc/) ) **사용**
    - **영화 리뷰 20만 건이 저장**
    - **각 평가 데이터는 0과 1로 레이블링 (0 : 부정, 1 : 긍정 리뷰)**
  - **한글 자연어 처리**
    - konlpy 패키지에서 제공하는 Twitter 문서 분석 라이브러리를 사용

# 단어 벡터 생성

- 단어 벡터 만드는 과정을 소개
  - 가장 널리 사용되는 라이브러리 : **Gensim**
    - pip install gensim

```
from gensim.models.word2vec import Word2Vec
model = Word2Vec(sentence_list, min_count=1)
model.most_similar(positive="조선")
```

```
##
[('일본', 0.9953970909118652),
 ('관련', 0.9941188097000122),
 ('인물', 0.9938454031944275),
 ('러시아', 0.9931197166442871),
 ('주요', 0.9918481111526489),
 ('대원군', 0.9915156960487366),
 ...]
```

# 문장 유사도 측정

- 단어의 유사도
  - 두 개의 문자열이 얼마나 다른지를 나타내는 **편집 거리**를 이용
- 편집 거리
  - 한 단어에서 다른 단어로 바꿀 때 필요한 **최소한의 편집 행동**의 횟수
  - 편집 행동
    - 글자를 추가, 제거, 변경
- 두 문장의 편집 거리 계산
  - NLTK 라이브러리를 활용

- 단어 벡터 예제 실습
  - <실습:gg-54-단어벡터>
  - 뉴스 기사를 활용하여 단어벡터 제작
    - 국내언론사들의 뉴스기사 데이터를 읽고 단어벡터를 제작하고 단어의 유사도 확인
  - 문장유사도 분석
    - Nltk 라이브러리를 활용하여 문장 간 편집거리 계산 및 ngrams 분석



# 형태소 분석

- 단어 구분
  - 영어
    - 단어들이 대부분 스페이스로 구분, 단어 구분이 어렵지 않다
    - 예) I am a boy
  - 한글
    - 스페이스로 나뉜 단어가 조사를 포함하거나 복합명사인 경우 등이 있어 품사를 구분하는 작업이 영어처럼 간단하지 않다
    - 예) 나는 소년이다
      - 단어 구분 : ‘나는’ , ‘소년이다’
      - 추가적인 형태소 분석 : ‘나 ‘, ‘는’ , ‘소년 ‘, ‘이다 ’
- 형태소 분석(morphological analysis)
  - 한글 문장을 처리하려면 단어를 다시 더 작은 단위인 형태소로 나누는 절차가 필요

# KoNLPy 형태소 분석기

- KoNLPy 형태소 분석기
  - 한국어 자연어 처리를 위한 파이썬 모듈
  - <http://konlpy.org>
- 구성
  - Tag 서브패키지 – 형태소 분석기 엔진
    - Hannanum (한나눔) : KAIST
    - Kkma (꼬꼬마) : 서울대
    - Komoran (코모란) : Shineware
    - Mecab (메카브) - 일본어용 형태소 분석기를 한국어 사용할 수 있도록 수정
    - Twitter (Okt, Open-korean-text) : twitter 개발
    - analyze(형태소 후보, Hannanum), morphs(형태소 파싱), nouns(명사 추출), pos(POS 태거) 메소드 포함
  - Corpus 서브패키지 – 다양한 말뭉치 포함
    - Kolaw(대한민국헌법), kobill(법률)
  - 기타 모듈
    - data(패키지경로), downloader(konlpy서버데이터접근), jvm(JVM초기화), utils(코드변환, csv파일처리, 텍스트출력 등)

- **형태소 분석 예제 실습**
  - <실습:gg-55-형태소\_분석>
  - 한글 형태소 분석기인 konlpy 실습
    - Konlpy 라이브러리를 활용하여 한글 형태소 분석

# 토픽 모델링

- 토픽 모델링
  - 문서의 주제(카테고리)를 구분하는 것
  - 미리 카테고리가 정해져 있지 않으므로 비지도 학습에 해당
- 관련된 단어나 문서의 집합을 찾는 방법이 필요
  - 잠재 디리클레 할당, LDA(Latent Dirichlet Allocation) 방법을 주로 사용
  - 관련성이 높은 단어들이 발생 → 같은 토픽으로 분류
- 한 문서에는 여러 토픽이 복합적으로 존재할 수 있다
  - 각 토픽의 비중은 다를 수 있다

# LDA (Latent Dirichlet Allocation)

- LDA
  - 문서의 각 토픽들이 디리클레 분포를 따른다고 가정
  - 각 문서를 각 토픽에 “할당” 하는 방식으로 동작
    - 문서마다 토픽이 어떻게 분포되어 있는지, 그리고 토픽마다 단어의 분포가 어떤지 파악
    - 토픽에 따라 단어의 분포를 결정하고 그중 가장 높은 확률의 단어를 선택

- LDA
  - 말뭉치로부터 **대표적인 토픽**을 먼저 선정
  - **해당 토픽**으로부터 **단어들을 뽑아서** 문서를 생성
- LDA의 학습 과정
  - 주어진 문서에 등장한 단어들이 어떤 토픽에서 뽑혔고
  - 그 토픽의 확률이 어떻게 분포하였는지를 추론해 내는 것
- 말뭉치의 단어가 어떤 토픽에 해당하는지 명시적으로 표시되어 있지 않기 때문에 학습을 통해 ‘잠재적’ 인 정보를 추출해야한다

# 뉴스 분석 실습

- 토픽 모델링 예제 실습
  - 〈실습:gg-56-토픽모델링〉
- 뉴스 기사를 사용하여 토픽 모델링을 수행하는 예
  - 먼저 여러 문서에서 자주 나타나는 공통 단어를 제거
  - 상위 10,000개의 단어를 선택하여 BOW 모델을 생성
- LDA 분석으로 얻은 결과는 주제를 구별하는데 도움을 주지만 비지도 학습이기 때문에 완벽한 정답은 아니다
  - 주제에 할당된 문서를 확인하여 평가, 검증하는 과정은 사람이 해주어야 한다

수고하셨습니다.

Q & A



가야캠퍼스 전경