DSAC Module-3(1)

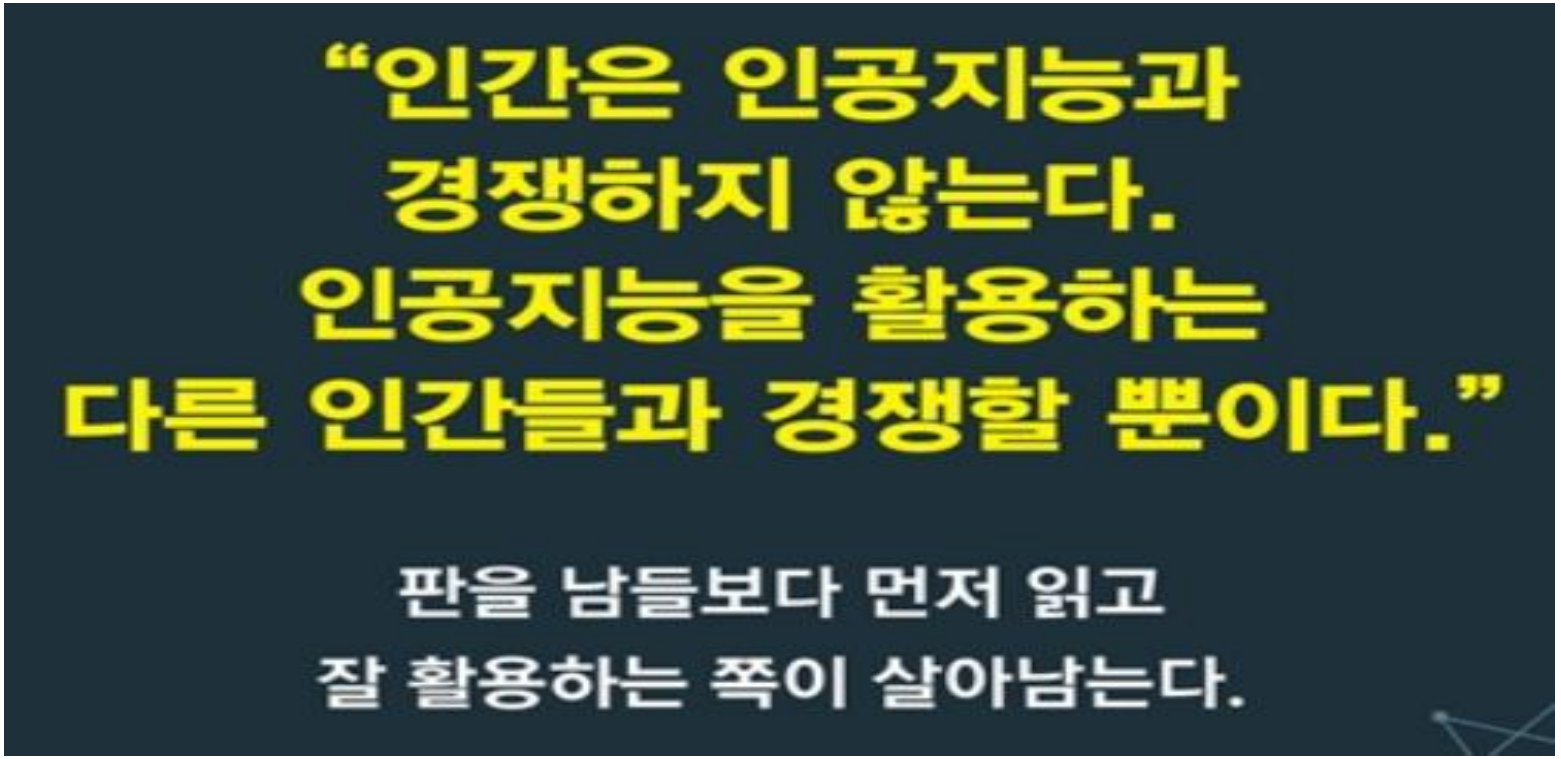
Machine Learning

2019, 2020, 2021, 2022

KPC

내용

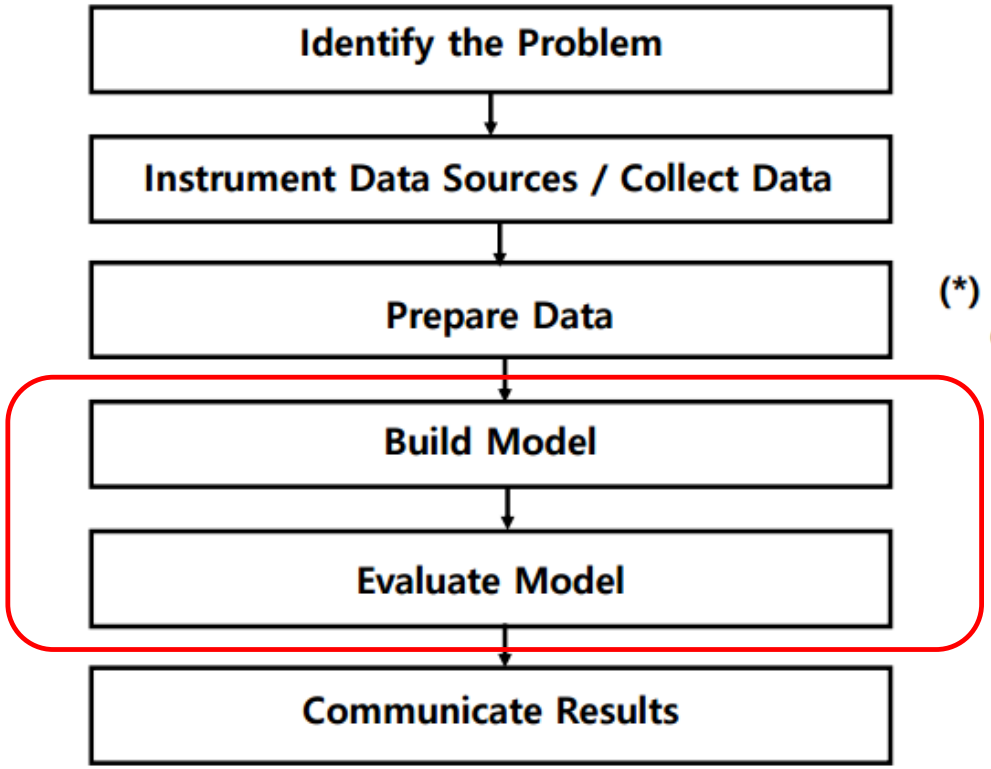
1. 머신러닝
2. kNN
3. 결정트리
4. 랜덤 포레스트
5. 서포트 벡터머신
6. 분류 성능
7. 특성공학
8. 모델 최적화
9. 이미지 분석
10. 텍스트 분석



출처: 인공지능시대의 비지니스전략

데이터 분석 모델

Data Analysis Model (Jeff Hammerbacher)



(*) integrate, transform,
clean, aggregate, alter

Machine Learning
(머신러닝 or 기계학습)

인공지능과 머신러닝

- 인공지능을 구현하는 방법은 다양
- 머신 러닝 기반의 AI가 2000년대 이후 급속히 발전
- 딥러닝: 신경망을 기반으로 하는 머신 러닝 기술
  - 마치 사람이 많은 정보에 접하면서 학습하듯이 컴퓨터도 데이터를 보고 학습하는 방법
  - 음성인식, 자동차 번호판 인식, 언어 번역, 채팅 대화, 글쓰기, 작곡 등 여러 분야에서 좋은 성과를 낸다

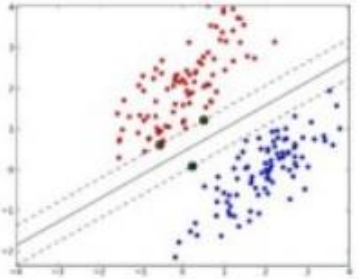
머신 러닝

인공지능(Artificial Intelligence)

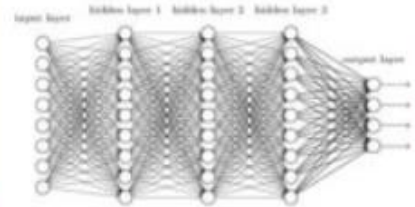
- Deep Blue

머신러닝(Machine Learning)

- Linear Regression
- Logistic Regression
- SVM



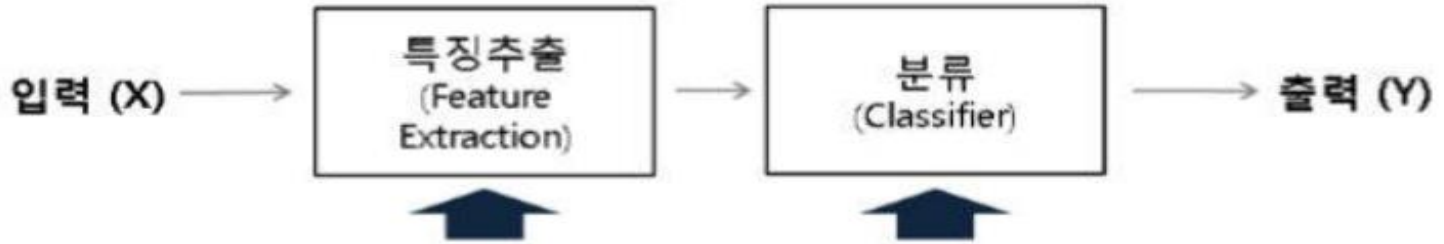
딥러닝(Deep Learning)



- CNN
- RNN
- LSTM

머신 러닝 vs 딥 러닝

< 머신러닝 (Machine Learning >



- 특징추출을 위한 알고리즘을 인간이 직접 제공
- 해당분야에 대한 지식 및 직관, 알고리즘 구축을 위한 상당한 노력 필요
- 컴퓨터 '학습'의 영역

< 딥러닝 (Deep Learning >

- 컴퓨터 '학습'의 영역

머신러닝 특징

- 예전에는 컴퓨터는 프로그래머가 코딩한 대로만 동작
  - 계산을 빨리 하든지,
  - 이미지를 처리하든지,
  - 정해진 알고리즘대로 빠르고 정확하게 동작하는 일
- 머신러닝
  - 컴퓨터가 데이터를 보고, 스스로 기능을 향상시키는 방법을 찾아내어서 점차 성능을 향상시킨다.

머신 러닝?

- 머신러닝 알고리즘
  - 머신(컴퓨터)이 데이터를 보고 학습을 하여 점차 지능적인 동작 수행
  - 목적 : 학습을 하여 어떤 "모델(model)" 을 만드는 것
    - 모델 : 예측 작업(회귀, 분류)을 수행하는 알고리즘 의 결과물
- 머신러닝 모델
  - 스팸 메일을 찾아내는 모델
  - 누가 게임에서 이길지 예측하는 모델
  - 내일 날씨를 예측하는 모델

머신러닝 알고리즘과 머신러닝 모델

- 머신러닝 **알고리즘**
  - 머신러닝 '모델'을 생성하기 위해 데이터에서 실행되는 절차
  - (ex) Linear Regression, Logistic Regression, Decision Tree, Neural Networks, k-Nearest neighbors, k_Means
- 머신러닝 **모델**
  - 데이터에서 실행된 기계학습알고리즘의 출력 (알고리즘에서 학습된 내용)
  - Model data + Prediction algorithm
  - (ex) linear regression: coefficient vector

    decision tree: tree of if-then statements

    neural network: vectors or matrices of weights
- Algorithm is used to find model, the model is the program that solves the problem.

모델의 가치

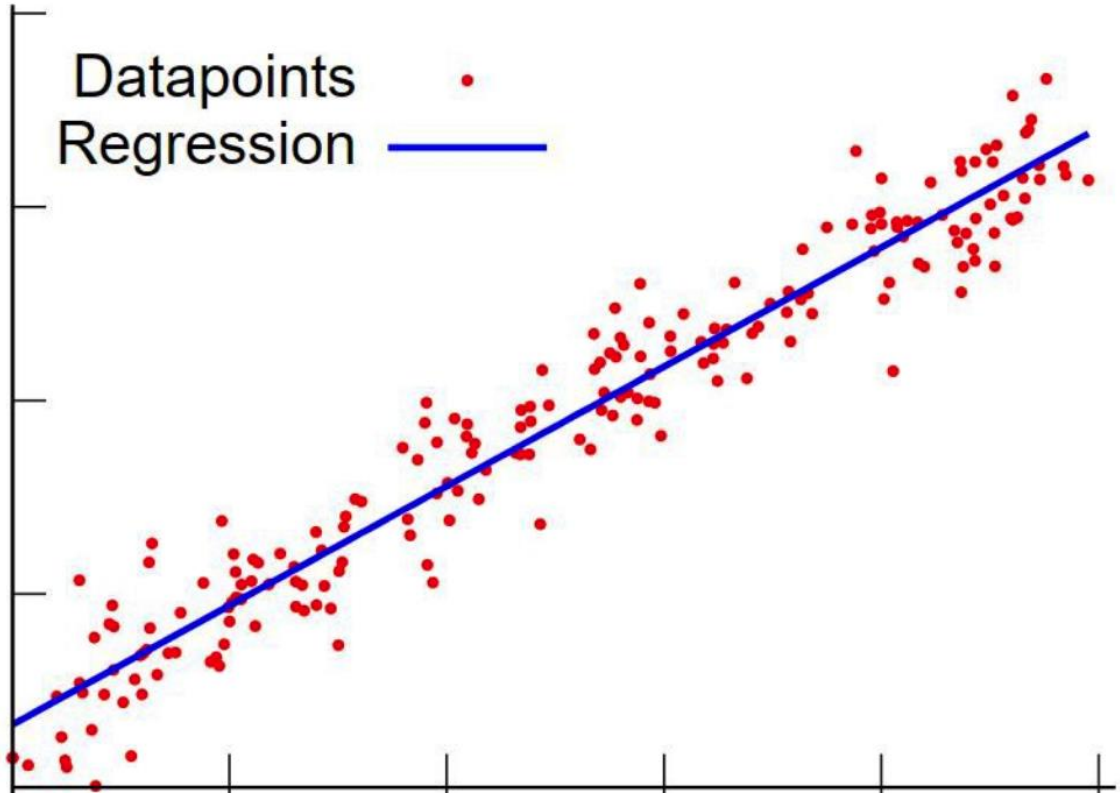
- 와인 품질 = 12.145 + (0.00117x겨울철 강수량)
  + (0.064x 재배철 평균기온) - (0.00386x 수확기 강수량)

머신러닝 모델

- 머신 러닝, AI 모델은 데이터 기반의 모델을 사용(학습)
- 현실 세계의 많은 현상
  - 수식으로 간단히 모델링하기 어렵고, 과학적으로 증명할 수도 없다.
  - 하지만 거의 정확히 예측할 수 있는 모델은 만들 수 있다.
    (단, 충분한 데이터 필요)
  - 머신 러닝 모델 예
    - 어느 고객이 불만이 많을 것인지
    - 어떤 영화가 관객을 많이 동원할지
    - 어떤 물건이 많이 팔릴지
    - 어떤 메일이 스팸일지
- 머신 러닝은 성능이 꽤 유용

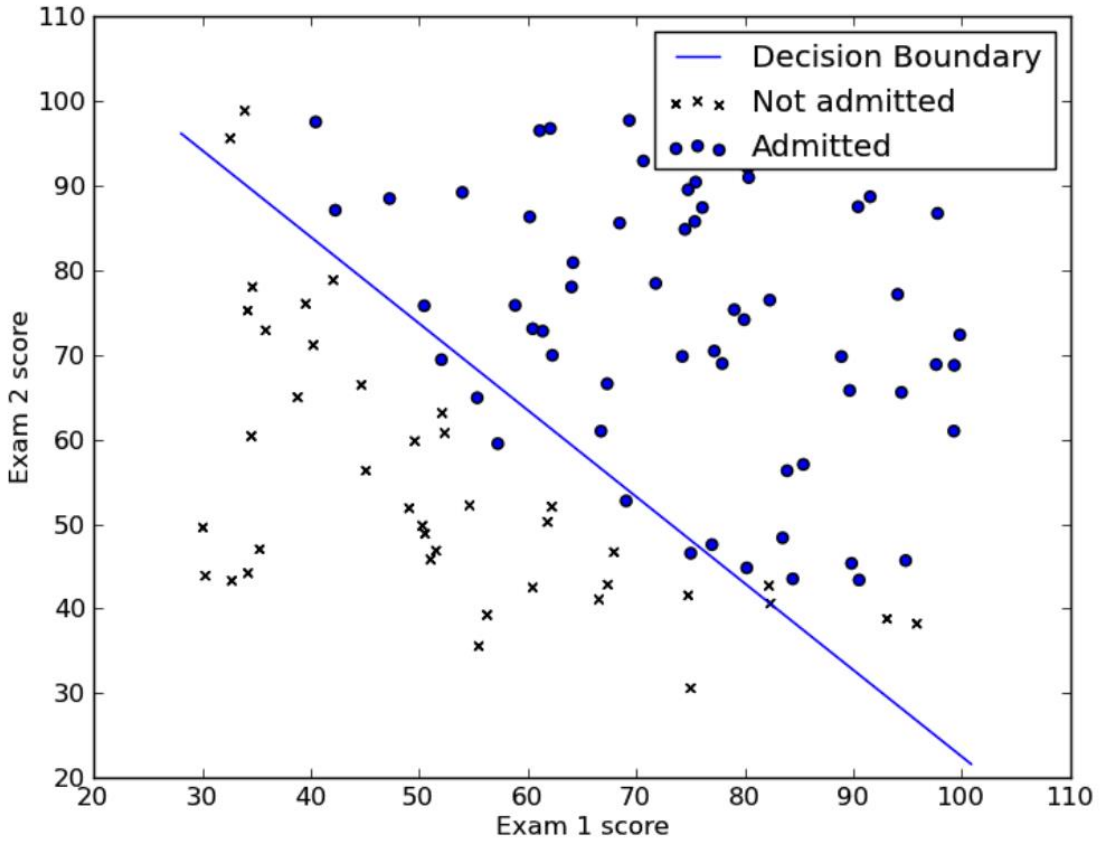
선형 회귀 모델

- 선형 회귀(regression) y = wX + b



선형 분류 모델

- 선형 분류(classification) ax1 + bx2 + c = 0



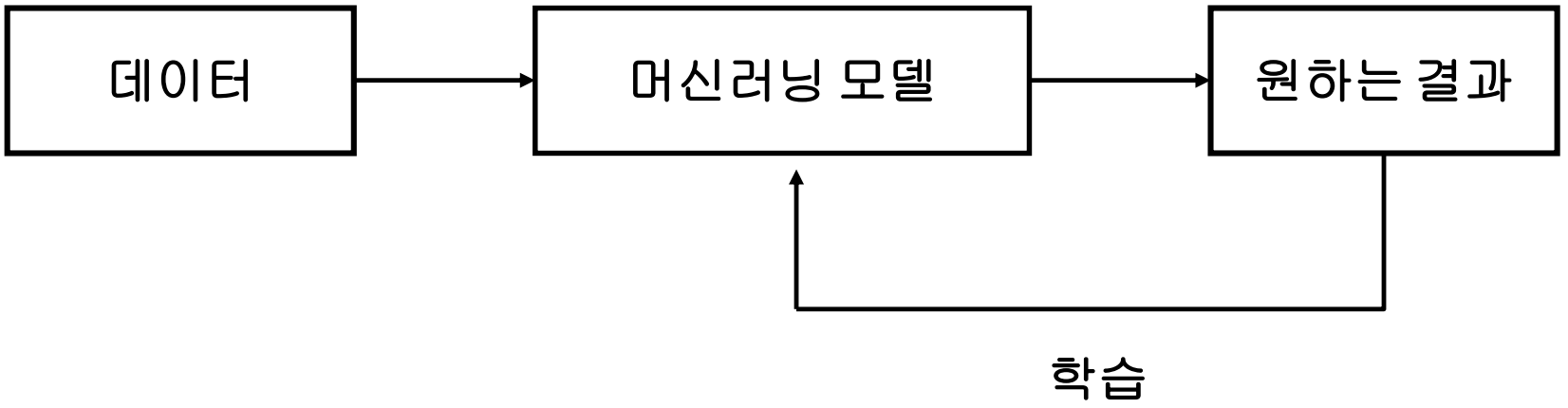
모델 파라미터

- 모델
  - 모델 구조 : 모델의 동작을 규정
  - 모델 파라미터 : 모델이 잘 동작하도록 정한 가중치 등 계수
- 특정 모델은 데이터 특성에 따라 예측 정확도가 달라질 수 있다
  - 적절한 모델 구조 : 프로그래머가 선택
  - 적절한 모델 파라미터 : 머신러닝 프로그램이 데이터 기반하여 학습

(*) Hyper-Parameter (하이퍼파라미터): 모델 외부에 있으며, 데이터에서 추정할 수 없는 파라미터. 주로 경험있는 사람이 주며, heuristic이나 경험에 의해 결정됨.

머신 러닝의 기본 동작

- **머신러닝 목표**
  - **주어진 학습 데이터를 보고 원하는 동작을 잘 수행하는 모델을 만드는 것**
  - **즉, 회귀 또는 분류 작업을 정확하게 수행**
  - **좋은 모델을 만들기 위해서는 → 좋은 데이터가 필요**



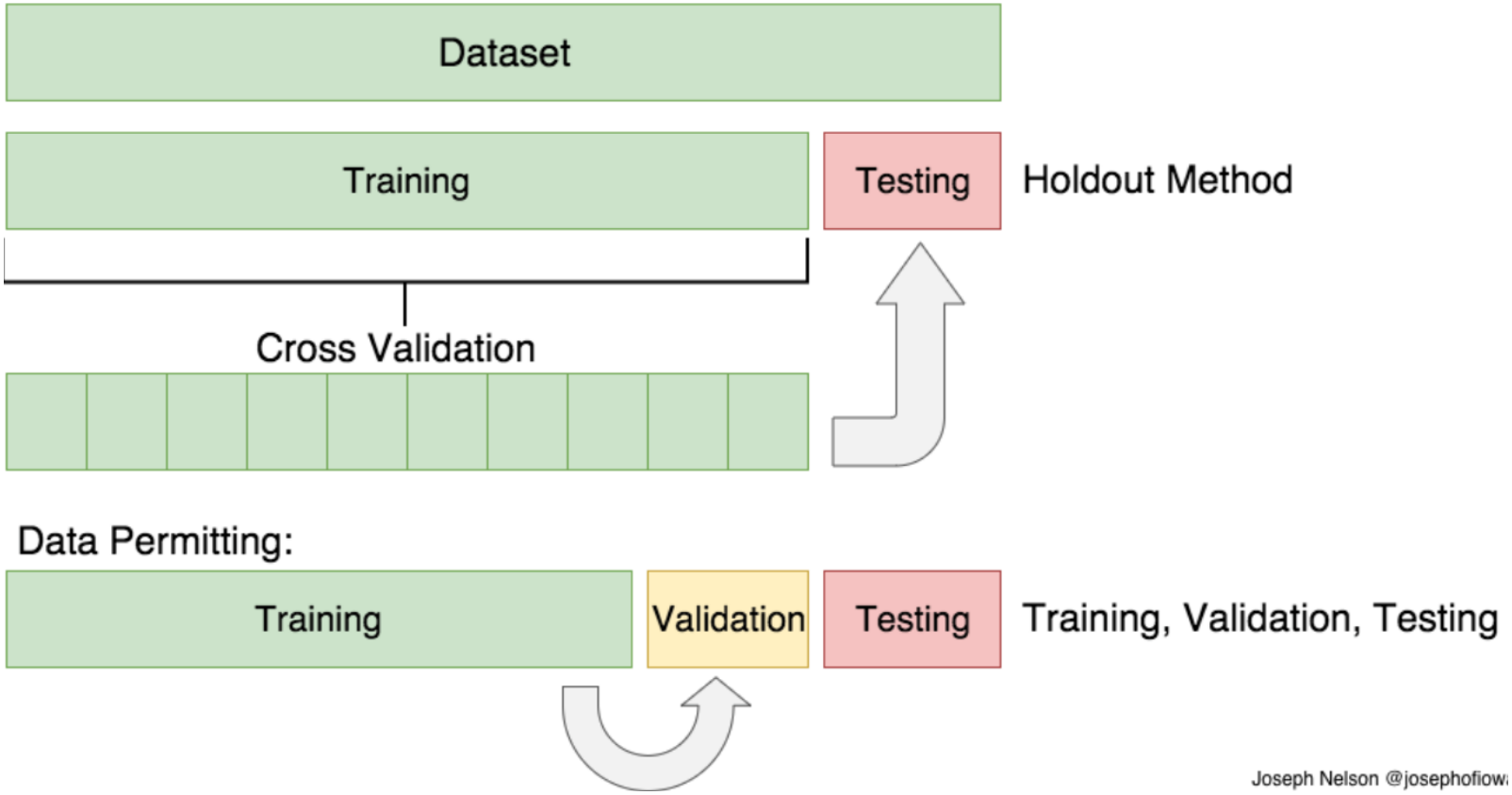
훈련과 검증

- **모델 훈련(Training)**
  - **모델이 데이터를 이용하여 모델 파라미터를 학습하는 과정**
  - **모델 파라미터 값 : 보통 랜덤한 값으로 초기화**
  - **학습**
    - **훈련 데이터에 기반하여 최적화 알고리즘에 의해서 모델 파라미터 값을 계속 갱신하여 모델의 예측 값이 실제 값에 수렴하도록 하는 훈련 과정**
- **모델 검증 (Validation)**
  - **모델을 학습시킨 후, 모델이 잘 동작하는지를 확인하는 과정**
  - **보통 검증 데이터를 따로 제공하지 않으므로 훈련에 사용할 데이터의 일부를 검증용으로 미리 확보해야**

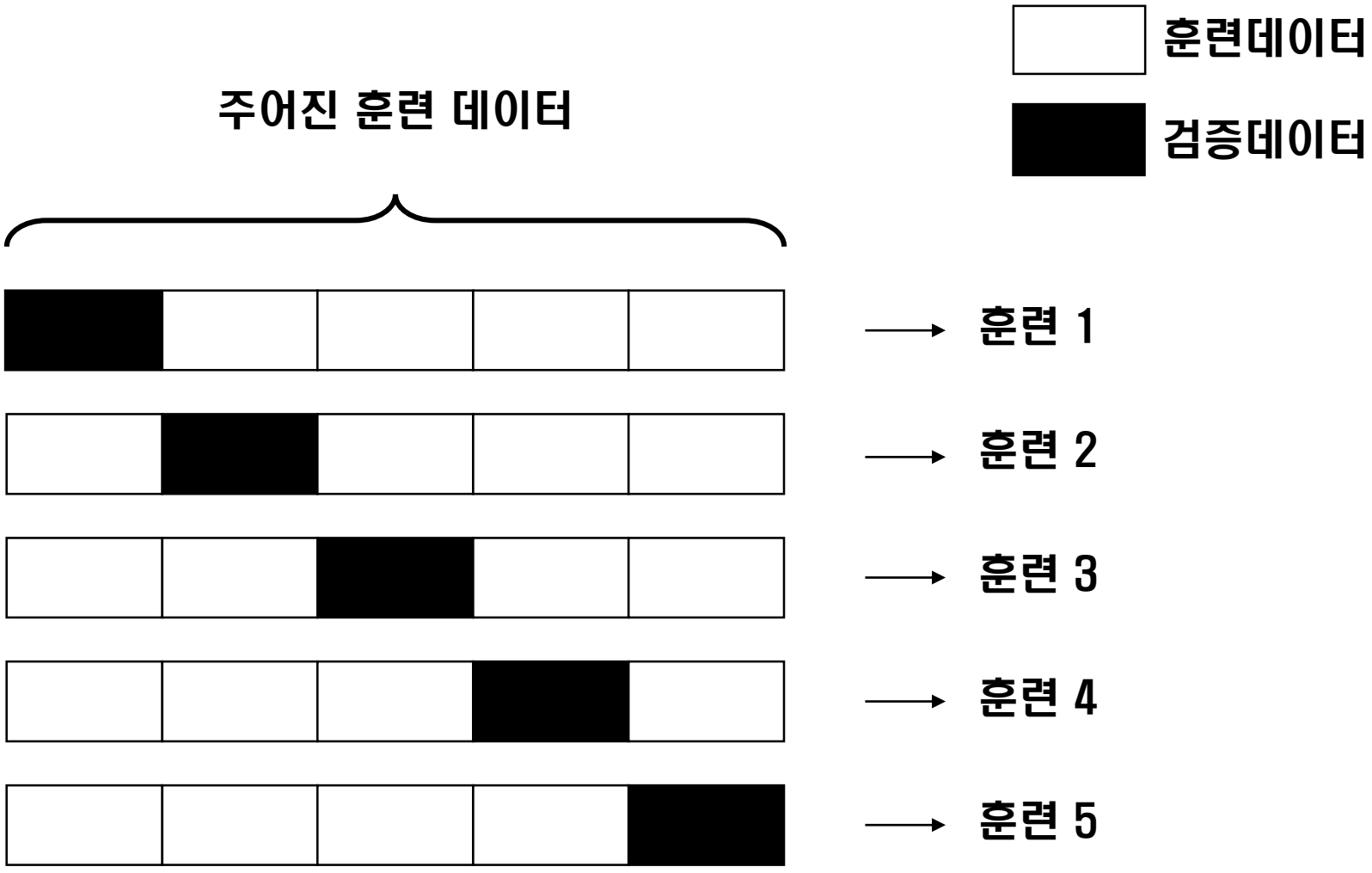
훈련, 검증, 테스트 데이터

- **훈련**(Training) 데이터
  - 모델 parameter를 학습시키는데 사용
- **검증**(Validation) 데이터
  - 모델의 학습 중에 과소적합, 과대적합을 검사하고 최적 모델 구조(hyper parameter 등)를 찾는데 사용
  - 훈련 데이터 중의 일부를 학습에 참여시키지 않고 남겨 둔 데이터
- **테스트**(Test) 데이터
  - 모델의 성능을 최종적으로 시험하는데 사용

훈련, 검증, 테스트 데이터


Dataset
Training
Testing
Holdout Method
Cross Validation
Data Permitting:
Training
Validation
Testing
Training, Validation, Testing
Joseph Nelson @josephofiow

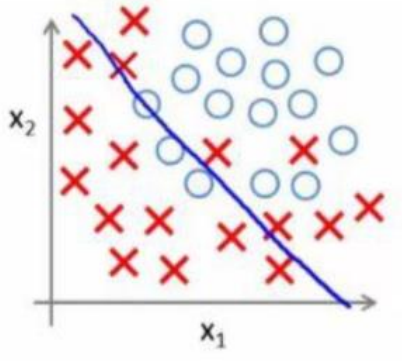
k-fold 교차 검증



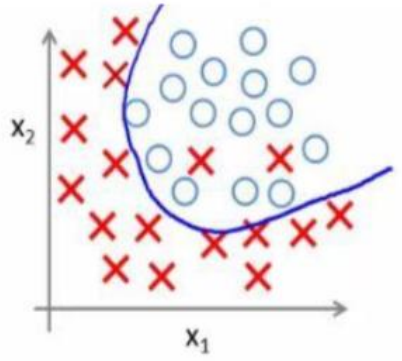
k-fold 교차 검증

- fold : 검증 데이터
- 주어진 데이터 전체를 골고루 검증용으로 사용
  - 모델의 동작을 보다 정교하게 확인하기 위함
  - K 값은 보통 5~10 주로 사용
  - cross_val_score() : 교차 검증 자동 수행 & 성능 평가
- 교차 검증의 목적은 성능 검증
- K개의 점수가 골고루 나와야 안정적인 모델

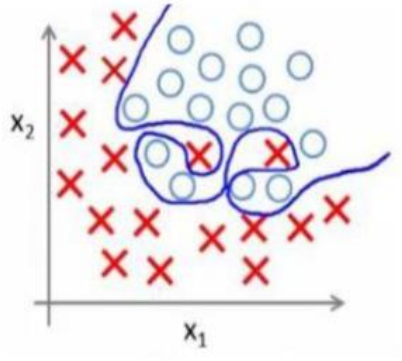
과대 적합, 과소 적합



과소 적합



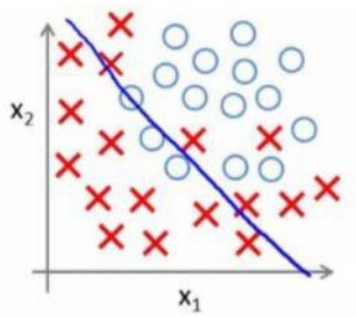
Good fit

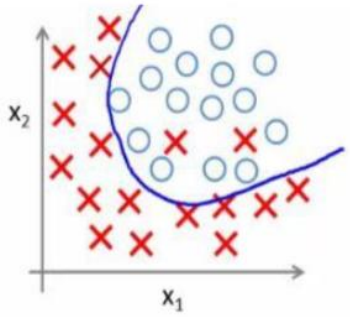


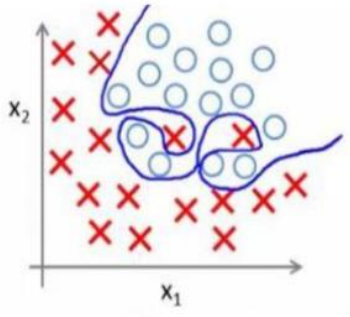
과대 적합

과대 적합(Overfitting)

- 모델이 훈련 데이터에 대해서만 잘 동작하도록 훈련되어, 새로운 데이터에 대해서는 오히려 잘 동작하지 못하는 것
  - 주어진 훈련 데이터를 너무 세밀하게 학습에 반영하여 발생하는 현상
- 과대 적합된 모델은 훈련 데이터에 대해서는 매우 우수한 성능을 보이지만 일반성이 떨어진다







일반화(Generalization)

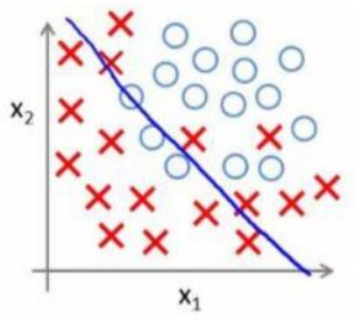
- **모델의 일반화(Generalization)**
  - **머신러닝에서는 과대 적합을 피해서 일반적으로 잘 동작하게 모델을 만드는 것이 매우 중요**
- **과대 적합의 원인 & 대책**
  - **원인 : 훈련 데이터가 너무 적어서 학습을 충분히 할 수 없는 경우**
    - **대책 : 다양한 경우를 고려한 훈련 데이터를 많이 확보**
  - **원인 : 모델이 너무 복잡한 경우**
    - **대책 : 모델을 좀 단순하게**

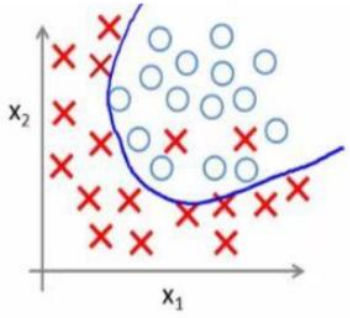
규제화(Regularization)

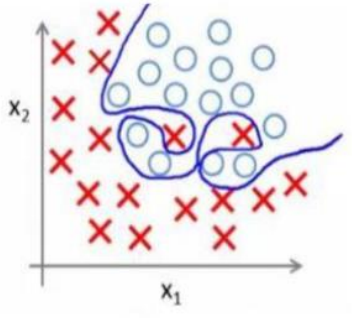
- 규제화
  - 모델이 일반화 능력을 갖도록 모델의 기능을 제한하는 것
  - 예) 만일 학습할 데이터가 부족하다면,
    - 모델 구조를 좀 단순하게 만들어서 주어진 데이터에 대한 과대 적합을 피해야 한다
- 머신러닝에서는 일반화 능력을 가진 모델을 만드는 것이 중요
  - 이를 위하여 모델에 적절한 제한을 가하는 기법을 사용

과소 적합(Underfitting)

- 문제의 복잡도에 비해 **모델이 너무 간단**하여 주어진 **훈련 데이터에서 조차도** 잘 동작하지 못하는 것
- 대책
  - 모델의 보다 복잡(상세)하게 구성
  - 제약을 줄여준다







데이터의 대표성

- 훈련 데이터 구성
  - 미래에 나타날 가능성이 있는 모든 데이터의 특징을 골고루 반영
  - 예) 투표 결과 예측
    - 실제 인구 구성에 비례하여 성별, 지역, 인종, 나이별, 소득별 등 균형성 유지
- 층화 샘플링 or 다단계 샘플링(stratified sampling)
  - 데이터의 대표성을 고려하여 데이터를 수집하는 방법
    - 예) 어느 학교의 남녀 학생 비율이 8:2 → 의견수렴 샘플도 8:2 유지
- 훈련, 검증, 테스트 샘플 데이터가 전체 데이터의 특징을 계속 유지할 수 있어야 함

모델 구축 과정

- 머신러닝 모델 선택
  - 해결할 문제에 최적의 모델 선택
    - 훈련 데이터, 원하는 목적(기능) 등 고려
  - 선형모델, 결정트리, 신경망, SVM, 랜덤포레스트 등
- 모델 학습 : 훈련 데이터 사용
  - fit() 함수
- 모델이 과대 적합 또는 과소 적합인지를 검증
  - 과대 적합 → 모델을 더 일반화(모델 단순화 또는 규제화)
  - 과소적합 → 모델을 더 복잡(상세)하게 설계
- 성능 평가 : 실제 테스트 데이터를 적용
  - predict(), score(), predict_proba() 함수

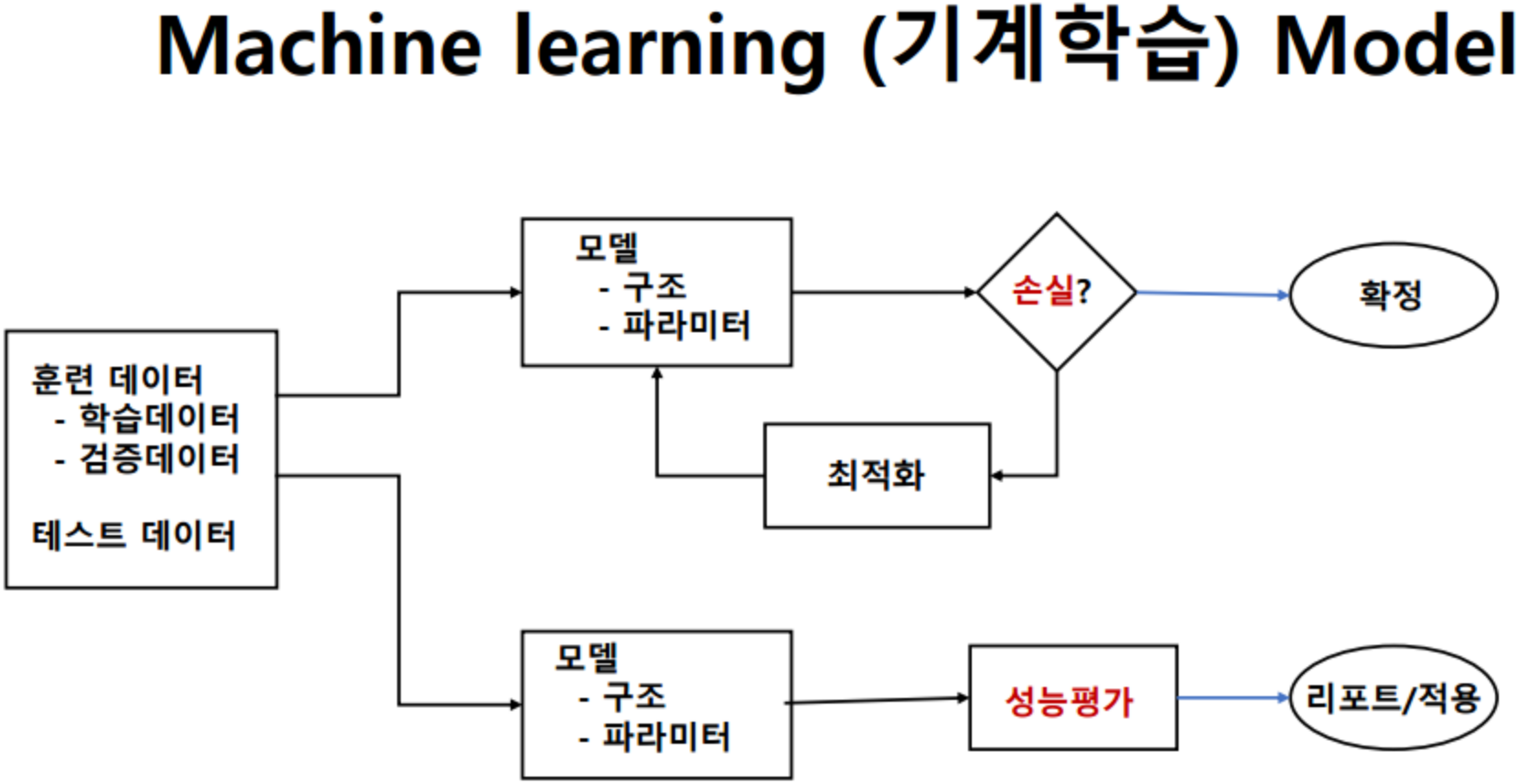
머신 러닝의 유형별 대표적 알고리즘

	머신러닝 유형	알고리즘
지도학습	분류	kNN, 베이즈, 결정 트리, 랜덤 포레스트, 로지스틱 회귀, 그라디언트부스팅, 신경망
	회귀	선형 회귀, SVM, 신경망
비지도학습	군집화	k-means, DBSCAN
	데이터 변환	스케일링, 정규화, 로그변환
	차원축소	PCA, 시각화

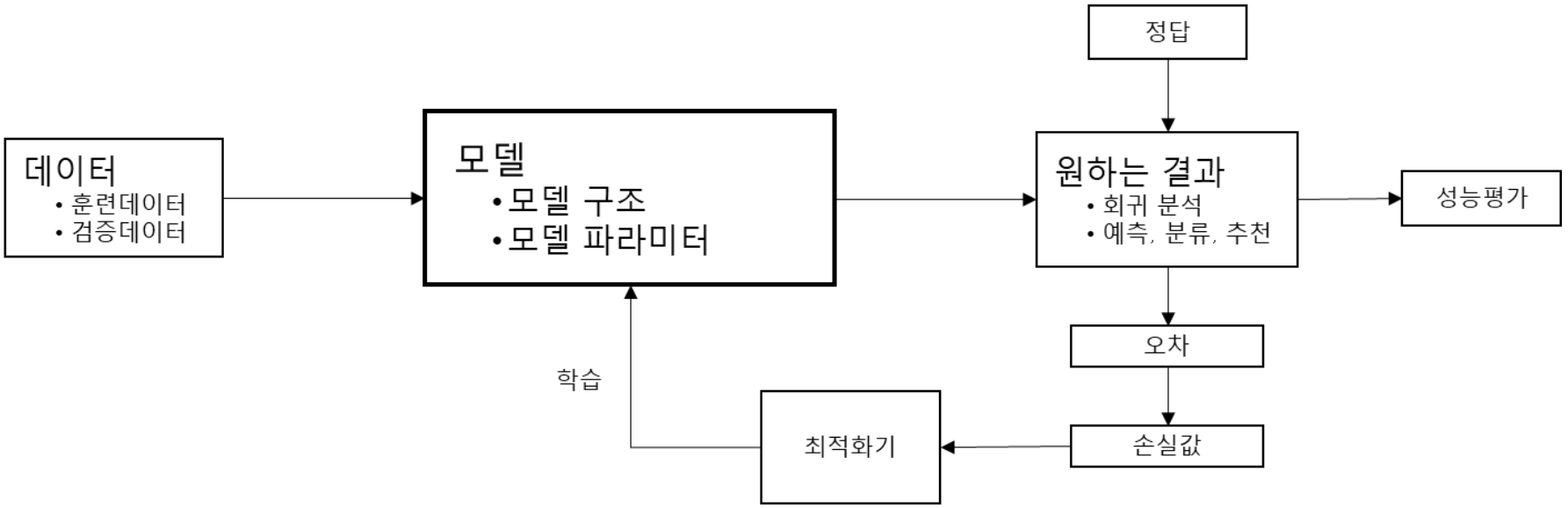
모델의 동작 성능

- 모델의 동작 속도
  - 학습 시간 : 모델을 만드는데 걸리는 시간
  - 동작 속도 : 모델을 적용하는데 걸리는 시간
- 일반적으로 모델이 정교하고 복잡할수록 성능은 좋아지지만 모델을 만들거나 적용하는데 시간이 오래 걸린다.

모델의 동작


Machine learning (기계학습) Model
훈련 데이터
- 학습데이터
- 검증데이터
테스트 데이터
모델
- 구조
- 파라미터
손실?
확정
최적화
모델
- 구조
- 파라미터
성능평가
리포트/적용

모델의 학습


데이터
• 훈련데이터
• 검증데이터
모델
• 모델 구조
• 모델 파라미터
정답
원하는 결과
• 회귀 분석
• 예측, 분류, 추천
성능평가
오차
손실값
최적화기
학습

손실 함수

- **손실함수**(loss function)
  - 모델의 **예측값**과 **실제 값**과의 **차이**, 즉 **오차(error)**를 계산
- 이 **오차를 줄이는 방향**으로 모델을 **최적화**(**학습**) 한다
- **회귀분석**에서 **많이 사용**하는 손실함수
  - 오차 자승의 합의 평균치(MSE: mean square error)

$$MSE = \sum_{k=1}^{N} (y - \hat{y})^2$$

  - N: 배치 크기

- 배치 크기 같은 설정 환경 변수를 hyper parameter라고 한다.
  - **hyper parameter** : 사람이 선택하는 변수
  - **parameter** : 기계 학습으로 자동으로 갱신되는 변수

분류의 손실 함수

- 분류에서는 손실함수로 MSE를 사용하는 대신 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) 개념

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

불확실성

- 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

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

Binary Cross Entropy Loss

$$L = -y * \log(p) - (1 - y) * \log(1 - p) :$$

- y: output label (0 and 1), p: predicted probability
- **즉, 확률 분포에 대한 엔트로피 값이 클수록 분포의 불확실성이 커지고, 마찬가지로 값이 작을수록 더 확실한 분포를 나타낸다.**

$$L = -y * \log(p) - (1 - y) * \log(1 - p) = \begin{cases} -\log(1 - p), & if\ y = 0 \\ -\log(p), & if\ y = 1 \end{cases}$$

- **이를** Log-loss **라고도 함. 확률** p**를 계산하기 위해 시그모이드 함수를 사용할 수 있다. 여기서** z**는** input feature **의 함수이다.**

$$S(z) = \frac{1}{1 + e^{-z}}$$

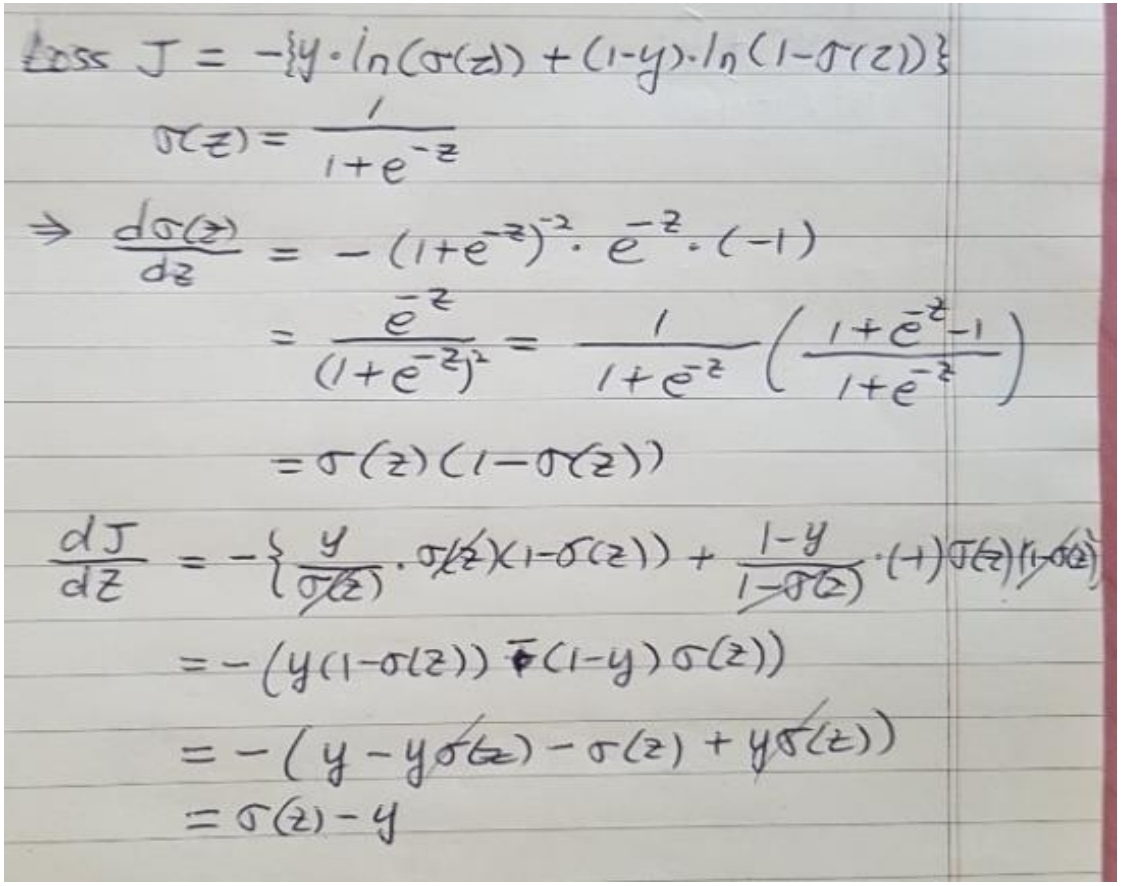
Binary Cross Entropy Loss

Binary cross entropy loss function:

$$J(\hat{y}) = \frac{-1}{m}\sum_{i=1}^{m} y_i \log(\hat{y}_i) + (1 - y_i)(\log(1 - \hat{y})$$

where

m = number of training examples
y = true y value
$\hat{y}$ = predicted y value



배치와 이포크

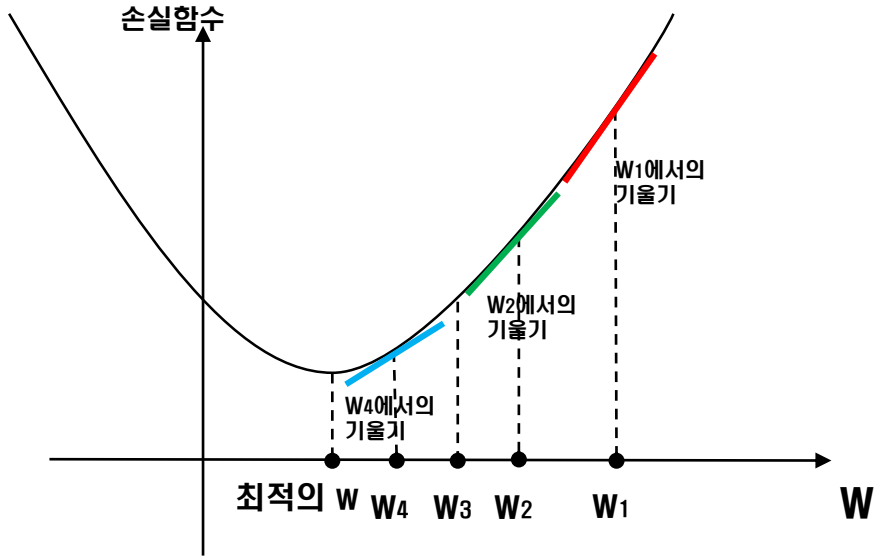
- 배치(Batch) 크기 : 한번 학습에 사용하는 샘플 수
- 예를 들어, 총 1,000개의 데이터로 예측 모델을 만들 때, 한번에 1,000개의 데이터를 모두 입력하여 손실함수를 구하고 학습을 시키는 것은 비효율적
  - 배치 크기가 클수록 학습이 정교하고, 기울기를 정확히 구할 수 있으나 계산량이 많아진다.
  - 한번에 필요한 메모리 사용량이 많아 메모리 오류가 날 가능성이 높다
  - 일반적으로 훈련 데이터를 일정 크기의 배치 단위로 나누어 학습을 시키는 것이 효과적
- 배치 크기가 작을 때에는 기울기가 상대적으로 정확하게 계산되지 못하므로 학습률도 작게 잡는 것이 좋다

배치와 이포크

- 이포크(Epoch) : 주어진 훈련 데이터 전체를 한번 학습에 사용하는 것
- 학습에 주어진 1,000개의 훈련 데이터를 모두 학습에 사용하였어도 아직 최적의 모델 파라미터를 찾지 못했으면 주어진 데이터를 다시 반복하여 사용할 필요 있다
  - 머신러닝에서는 일반적으로 여러 이포크를 수행

훈련 방법 : 최적화 – 경사 하강법

- **경사 하강법**(Gradient Descent)
  - **가장 일반적인** 최적화 알고리즘
  - 손실함수를 계수에 관한 그래프로 그렸을 때 최소값으로 빨리 도달하기 위해서는 현재 위치에서의 **기울기**(미분값)에 **비례**하여 **반대방향**으로 이동

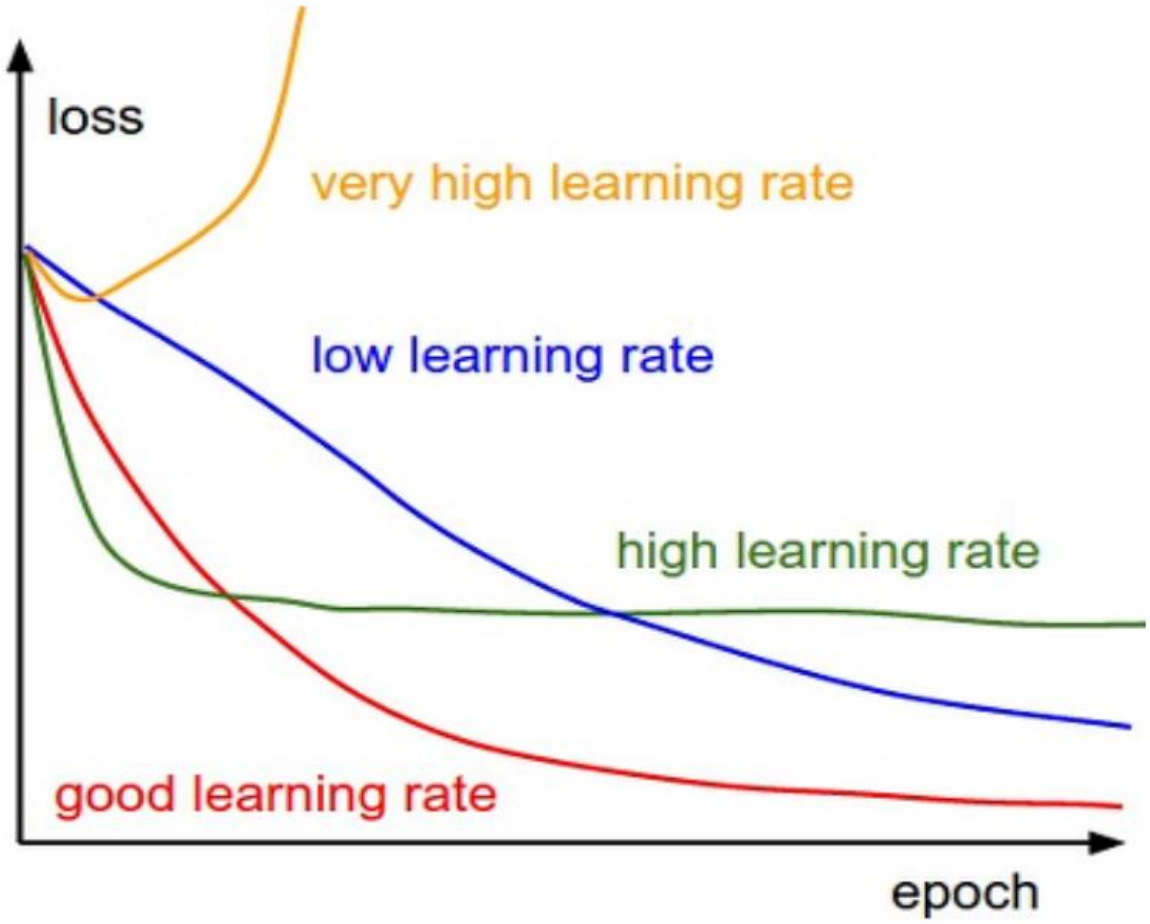


$$W_i = W_{i-1} - \eta Grad(i)$$

학습률 (Learning Rate)

- 학습률: 계수를 업데이트 하는 속도를 조정하는 변수
  - 학습률이 너무 작으면 수렴하는데 시간이 오래 걸리지만 최저점에 도달했을 때 흔들림 없이 안정적인 값을 얻게 되고,
  - 학습률을 너무 크게 정하면 학습하는 속도는 빠르나 자칫하면 최저점으로 수렴하지 못하고 발산하거나 수렴하더라도 흔들리는 오차가 남아있을 수 있다.
- 학습 스케줄(learning schedule) 기법
  - 초기에는 학습률을 크게 정하고 (학습률을 빠르게 하고) 오차가 줄어들면 학습률을 줄여서 안정상태(steady state)의 오차를 줄이는 방법

학습률 (Learning Rate)



출처: https://unabated.tistory.com/1122

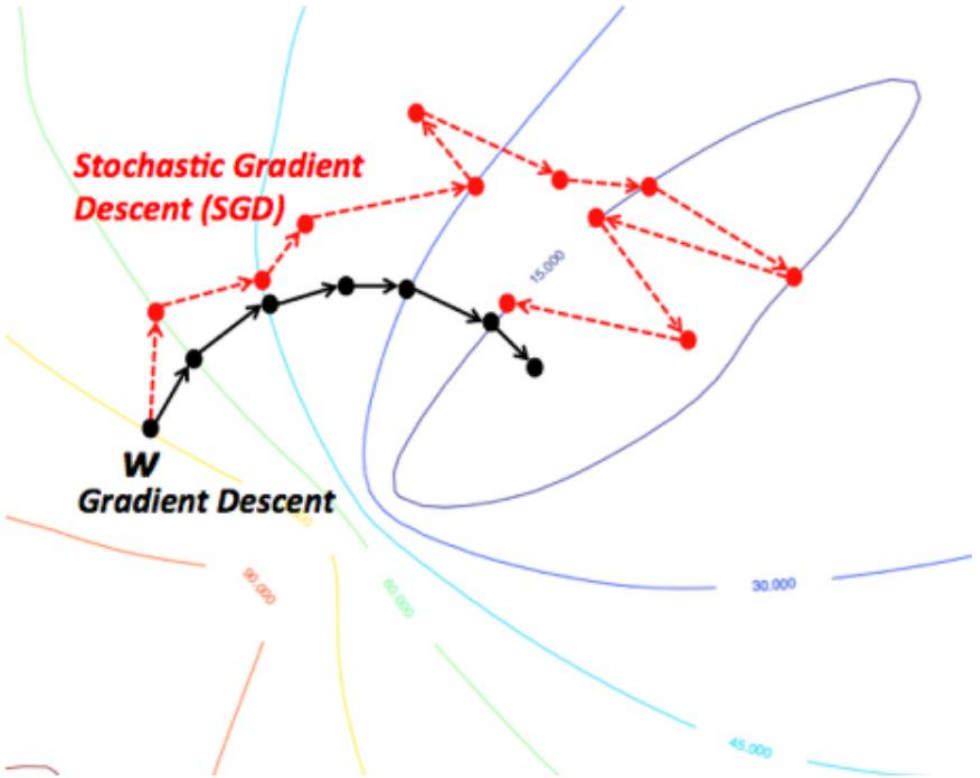
경사하강법 특징

- 경사하강법을 적용하려면 특성 변수들을 모두 동일한 방식으로 **스케일링**해야 한다.
- 특성 값마다 크기의 편차가 크면 특정 변수에 너무 종속되어 동작할 수 있고 이로 인해 수렴속도가 직선이 되지 않고 오래 걸릴 수가 있다.
- Local minimum 에 머무를 수 있음.
- 배치 사이즈가 커지면 시간이 오래 걸림.

경사하강법

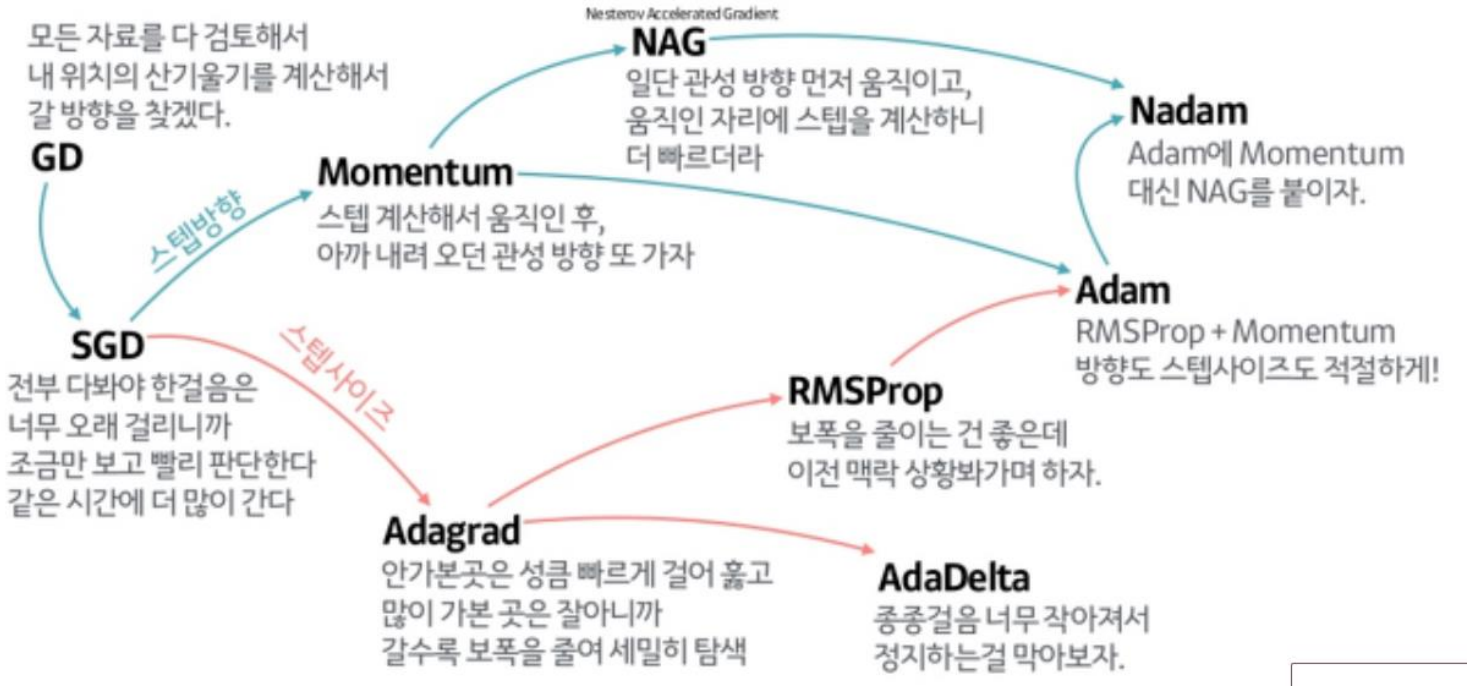
- 배치(Batch) GD (or Mini Batch GD)
  - 일반적으로 배치 GD방식을 많이 사용하는데, 적절한 크기 (10~ 1,000)의 배치단위로 입력 신호를 나누어 경사하강법을 적용하는 방식이다.
- SGD (확률적 경사하강법: Stochastic GD)
  - 한 번에 한 샘플씩 랜덤하게 골라서 훈련에 사용하는 방법이다.
  - 즉 샘플을 하나만 보고 계수를 조정한다. 계산량이 적어 동작속도가 빠르고, 랜덤한 방향으로 학습을 하므로 전역 최소치 (global minimum)를 찾을 가능성이 높아진다.
  - 매 샘플이 너무 랜덤하여 방향성을 잃고 수렴하는데 시간이 오래 걸릴 가능성도 있다. 노이즈가 심함.
  - In Python, use different Loss function and penalty
    - SGDClassifier() for classification
    - SGDRegressor() for regression

GD and Stochastic GD



출처: https://unabated.tistory.com/1122

Other Optimizers



출처: https://unabated.tistory.com/1122

경사 하강법의 원리

$$\underline{W}(n+1) = \underline{W}(n) - \mu \frac{\partial e^2(n)}{\partial \underline{W}(n)}$$

$$= \underline{W}(n) - \mu \frac{\partial e^2(n)}{\partial e(n)} \cdot \frac{\partial e(n)}{\partial \underline{W}(n)}$$

$$= \underline{W}(n) - 2\mu e(n) \cdot \frac{\partial \left[d(n) - \underline{W}^T(n) \cdot \underline{X}(n) \right]}{\partial \underline{W}(n)},$$

$$\frac{\partial \underline{A}^T \cdot \underline{B}}{\partial \underline{A}} = \underline{B}$$

$$= \underline{W}(n) + 2\mu e(n) \underline{X}(n)$$

$\because e = (W^T \cdot X - d)$ 이므로

학습률 (Learning Rate)

- **학습률: 학습 속도를 조정하는 변수**
  - **학습률이 너무 작게 잡으면**
    - **수렴하는데 시간이 오래 걸리지만 최저점에 도달했을 때 흔들림 없이 안정적인 값을 얻을 수 있다**
  - **학습률을 너무 크게 정하면**
    - **학습하는 속도는 빠르나 자칫하면 최저점으로 수렴하지 못하고 발산하거나 수렴하더라도 흔들리는 오차가 남아있을 수 있다.**
- **학습 스케줄(**learning schedule**) 기법**
  - **초기에는 학습률을 크게 정하고(학습을 빠르게 하고), 오차가 줄어들면 학습률을 줄여서 안정 상태(**steady state**)의 오차를 줄이는 방법**

경사 하강법 특징

- 경사하강법을 적용하려면 특성 변수들을 모두 동일한 방식으로 스케일링해야 한다.
- 특성 값마다 크기의 편차가 크면
  - 특정 변수에 너무 종속되어 동작할 수 있고 이로 인해 수렴속도가 직선이 되지 않고 오래 걸릴 수가 있다.

경사 하강법의 종류

- **배치(**Batch) GD
  - **일반적으로 배치** GD**방식을 많이 사용하는데, 적절한 크기의 배치 단위로 입력 신호를 나누어 경사 하강법을 적용하는 방식**
- SGD (**확률적 경사 하강법)**
  - **한 번에 한 샘플씩 랜덤하게 골라서 훈련에 사용하는 방법**
  - **즉 샘플을 하나만 보고 계수를 조정**
  - **계산량이 적어 동작속도가 빠르고, 랜덤한 방향으로 학습을 하므로 전역 최소치를 가능성이 높아진다**
  - **매 샘플이 너무 랜덤하여 방향성을 잃고 수렴하는데 시간이 오래 걸릴 가능성도 있다**

모델의 성능 지표

- 모델의 성능을 평가하는 척도 필요

- 분류에서는 성능 척도로 정확도(accuracy)를 주로 사용
  - (참고) 분류에서 손실함수로 크로스 엔트로피를 주로 사용

- 손실함수와 성능 지표의 차이점
  - **손실함수** :
    - 모델을 훈련시킬 때의 기준
    - 모델은 손실함수를 최소화 하는 방향으로 학습
  - **성능 지표**
    - 이렇게 만든 모델이 궁극적으로 얼마나 잘 동작하는지를 평가하는 척도

  - 예) 과속단속을 통한 교통사고율을 줄이는 작업
    - 손실함수에 해당 : 과속 단속
    - 성능 평가 지표(궁극적 목표) : 교통 사고를 줄이는 것 → 교통 사고율 측정

모델의 성능 지표

- MSE(또는 RMSE)
  - 키를 예측 : RMSE = 5.7
  - 몸무게를 예측 : RMSE = 3.8
    - 단순히 RMSE 값만으로는 각각 성능이 얼마나 우수한 지 알기 어렵다
    - 또한 서로 데이터의 성격, 범위가 다르므로 상호 객관적 비교 평가도 어렵다
- R^2
  - 회귀분석에서 어떠한 모델에서도 동일한 의미의 성능평가 척도
    - 데이터 종류와 값의 크기 범위와 관계없이 성능 평가를 객관적
  - 실제 값을 잘 예측 (= 1)
  - 평균치 예측 정도의 수준 (= 0)
  - 평균치 예측만도 못한 수준 (= 음수)

대표적인 손실함수와 성능지표

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

Regression (회귀) – 예측, 분류

❖ **What to reduce? (Loss Function: 손실함수)**

- **MSE** (Mean Square Error)

$$MSE = \sum_{k=1}^{N} (y - \hat{y})^2$$

❖ **How Good it is? (Performance: 성능지표)**

- **R^2** (R-Squared)

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2$$

Classification (분류)

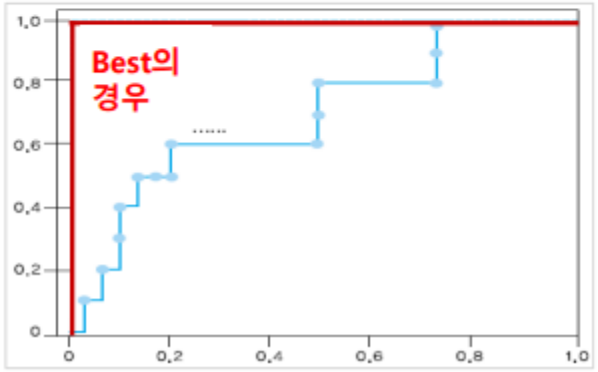
❖ **What to reduce? (Loss Function: 손실함수)**

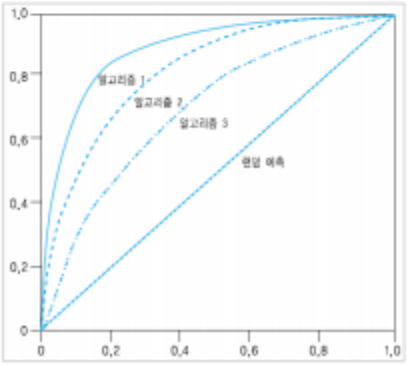
- **Cross Entropy (CE)**
- **Gini (지니계수)**

$$CE = \sum_i p_i \log(\frac{1}{p_i}) \qquad Gini = 1 - \sum_{k=1}^{m} p_k^2$$

❖ **How Good it is? (Performance: 성능지표)**

- **Confusion Matrix**: Accuracy, Recall, Precision, F-1 Score
- **Ranking(순서)**: ROC (Receiver Operating Characteristic), AUC (Area Under Curve)





모델의 성능 지표

- **정확도(accuracy): 정확하게 예측한 비율을 의미**
  - **accuracy = (TP+TN) / 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 => 97%**
  - **오류율 = 1-accuracy = 0.03 => 오진율은 3%**
- **리콜(recall): 관심 대상을 얼마나 잘 찾아내는가**
  - **recall = TP / (TP+FN)**
  - **실제 암 환자 발견률 = 6 / (6+4) = 0.6 => 60%**
- **정밀도(precision): 예측의 정확도**
  - **precision = TP / (TP+FP) = 6 / (6+2) = 0.75 => 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

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

$$\frac{1}{F_1} = \frac{\frac{1}{p} + \frac{1}{r}}{2}$$

- 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

머신러닝 유형

- **머신러닝을 사용하는 목적 즉, 머신러닝으로 문제를 해결하는 유형**
  - **설명**(description)
    - **클러스터링**
    - **비지도 학습**
  - **예측**
    - **회귀**(regression)
    - **분류**(classification)
    - **지도학습**
  - **추천**(recommendation)
  - **연관분석**
  - **강화학습**

데이터 분석의 유형 – 지도 학습

- 지도 학습(Supervised learning)
  - 입력 값(x)과 정답(y, label)를 포함하는 훈련용 데이터(training data)를 이용하여 학습하고, 그 학습된 결과를 바탕으로 미지의 데이터(test data)에 대해 미래 값을 예측(predict)하는 방법
  - 회귀나 분석 등 예측 모델은 시간이 지나면 정답을 확인할 수 있고, 모델의 성능에 대한 정확한 평가가 가능
  - 정답에 해당하는 값 : 목적변수(target variable), 레이블(label)
    - 회귀 : 수치 값
    - 분류 : 카테고리 변수
  - 예) 스팸 메일 분류기의 학습
    - 수집한 데이터로부터 어떤 메일이 스팸이었는 지 정답 샘플도 같이 주어져야 한다.

데이터 분석의 유형 – 지도 학습

- 회귀 분석
  - 수치를 예측하는 것

- 회귀 분석의 응용
  - 경제지표 예측
  - 사회학 연구
  - 마케팅
  - 의학에서 치료효과 분석

- 회귀 분석 알고리즘
  - 선형회귀
  - KNN
  - SVM
  - 로지스틱 회귀
  - 랜덤 포레스트
  - 신경망

데이터 분석의 유형 – 지도 학습

- 분류
  - 어떤 항목(item)이 어느 그룹에 속하는지를 판별
  - 이진 분류(binary classification)
    - 두 가지 카테고리를 나누는 작업
  - 다중 분류(multiclass classification)
    - 세 개 이상의 클래스를 나누는 작업

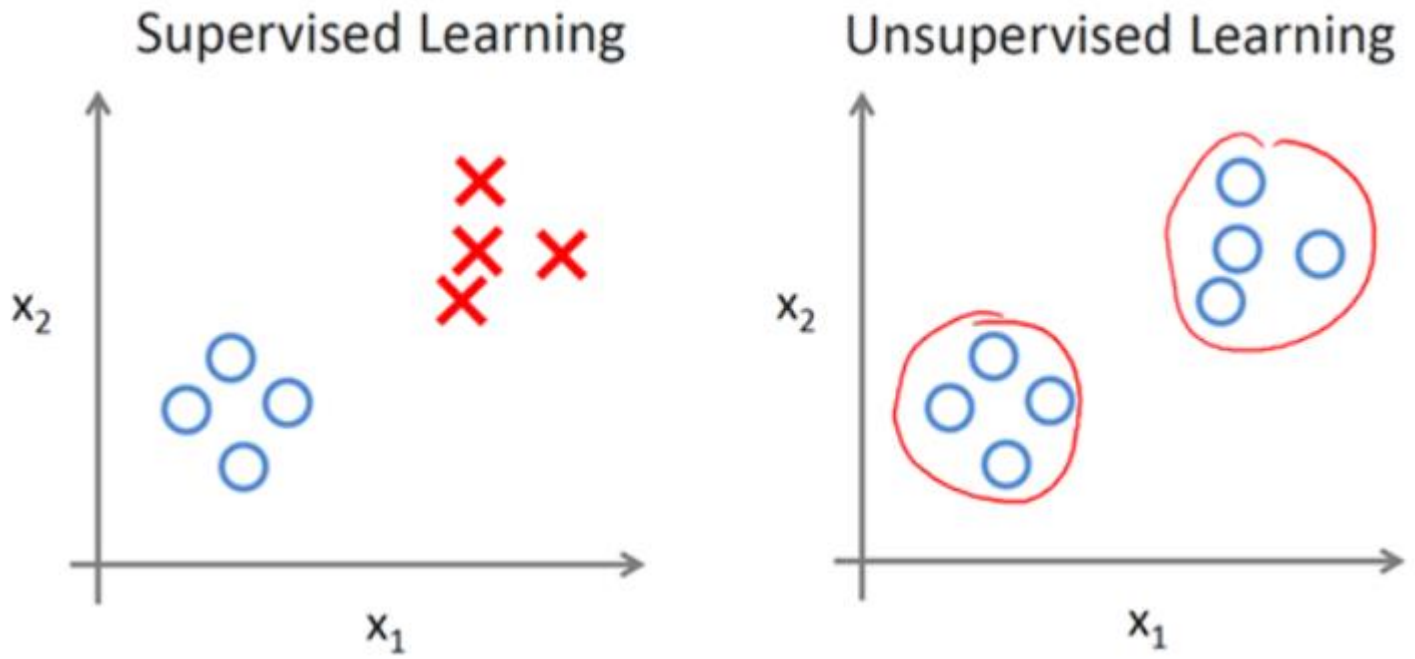
- 분류의 응용
  - 스팸 메일/우수 고객/충성심 높은 신입사원/투자할 좋은 회사 구분
  - 매장 입장 고객의 타입 분류
    - 물건을 구매, 단순히 구경, 항의 고객인지 판단하여 적절한 대응
  - 과거의 구매 이력/SNS 등을 분석하여 구매 확률이 높은 고객 구분
    - 광고 안내문, 기념품을 잠재 고객에게 보낼 때 필요

데이터 분석의 유형 – 비지도 학습

- 비지도 학습(Supervised learning)
  - 정답(label)은 없고 입력 데이터만 있는 훈련용 데이터(training data)를 이용한 학습을 통해 정답을 찾는 것이 아닌 입력 데이터의 패턴, 특성 등을 발견하는 방법
  - 데이터의 특성을 기술하는 서술형 모델

- 기법
  - 군집화(clustering)
    - 유사한 항목들을 같은 그룹으로 묶는다
  - 시각화
    - 데이터의 속성을 명확하게 시각화하기 위해서 고차원의 특성 값들을 2차원이나 3차원으로 차원을 축소하는 작업
  - 데이터 변환
    - 데이터를 분석하기 좋게 다른 형태로 변환
  - 주성분 분석(PCA)
    - 머신 러닝에 사용할 특성의 수를 줄인다.

지도 학습 vs 비지도 학습



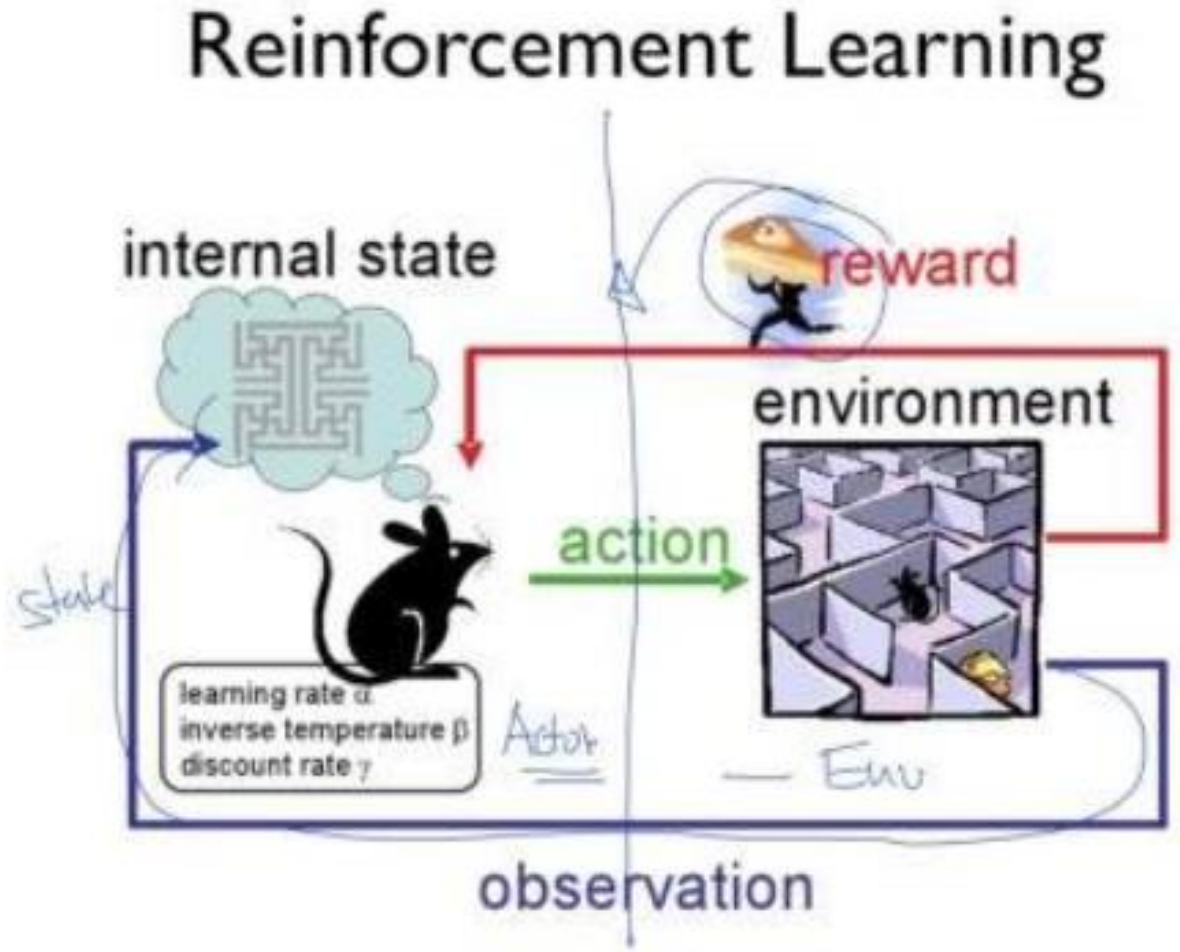
데이터 분석의 유형 – 비지도 학습

- **연관 분석**
  - 어떤 사건이 다른 사건과 얼마나 자주 동시에 발생하는지 파악
  - 자주 발생하는 패턴 찾기(상품의 연관성, 취향의 연관성 등 분석)
  - 같이 구매한 상품 분석(market basket analysis, 장바구니 분석)
  - 상품의 진열 배치 및 상품 프로모션(쿠폰 발행 등)에 활용

데이터 분석의 유형 – 강화 학습

- 강화 학습(Reinforcement learning)
  - 입력 샘플마다 정답이 있어 답을 알려주는 것이 아니라 일정 기간 동안의 행동(action)에 대해 보상(reward)을 해 줌으로써 어느 방향으로 학습해야 하는 지 방향성만 알려주는 학습 방법

- 응용 예
  - 게임의 경우 매 입력시마다 답을 주지는 못하지만, 게임을 이기고 있는 지, 지고 있는 지를 알려 줌
    - 스스로 게임을 잘 수행하는 방법을 터득
  - 로봇이 혼자 그네 타는 방법, 바둑 두는 방법을 터득
  - Alphago(바둑 프로그램)


Reinforcement Learning
internal state
reward
environment
action
learning rate α
inverse temperature β
discount rate γ
observation

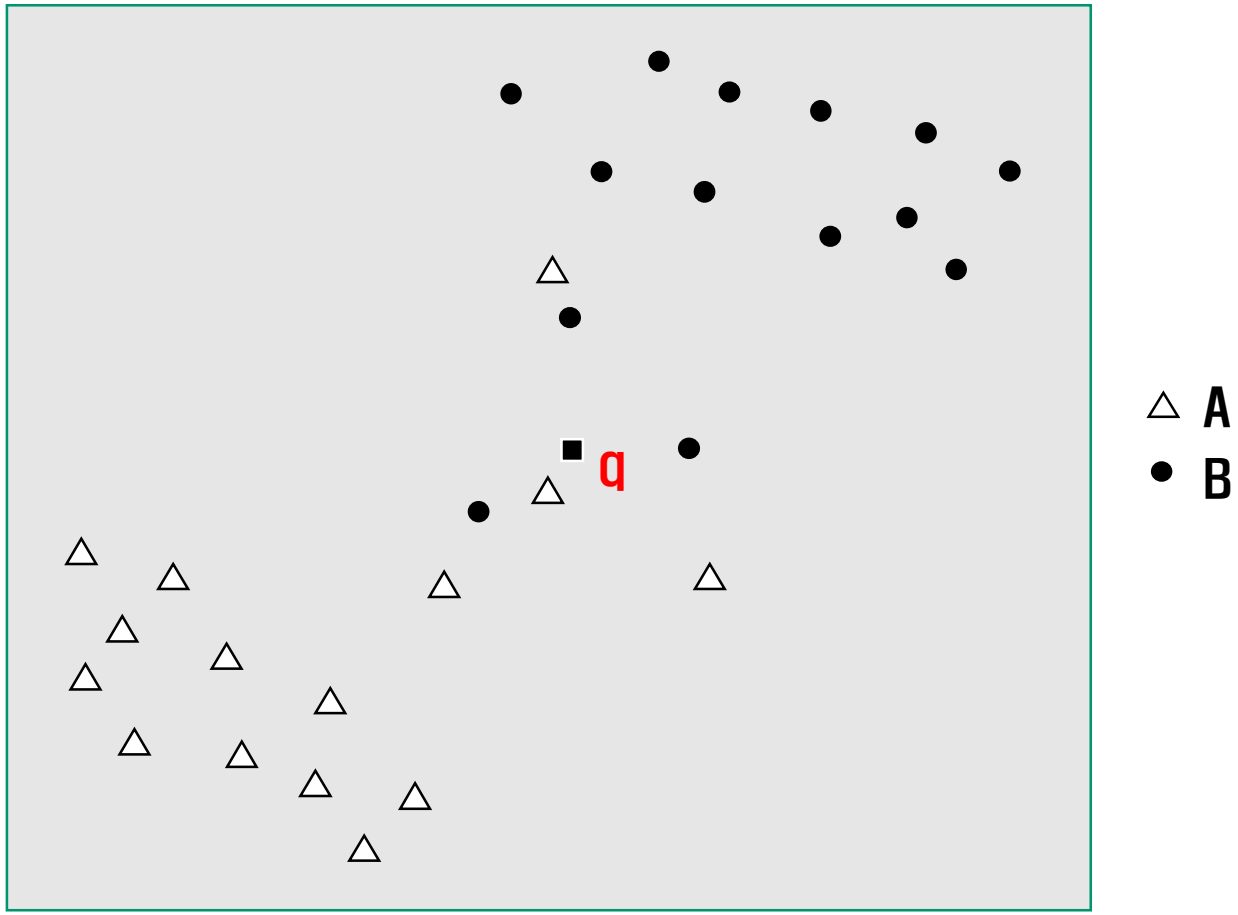
kNN

kNN (k-Nearest Neighbor)개요

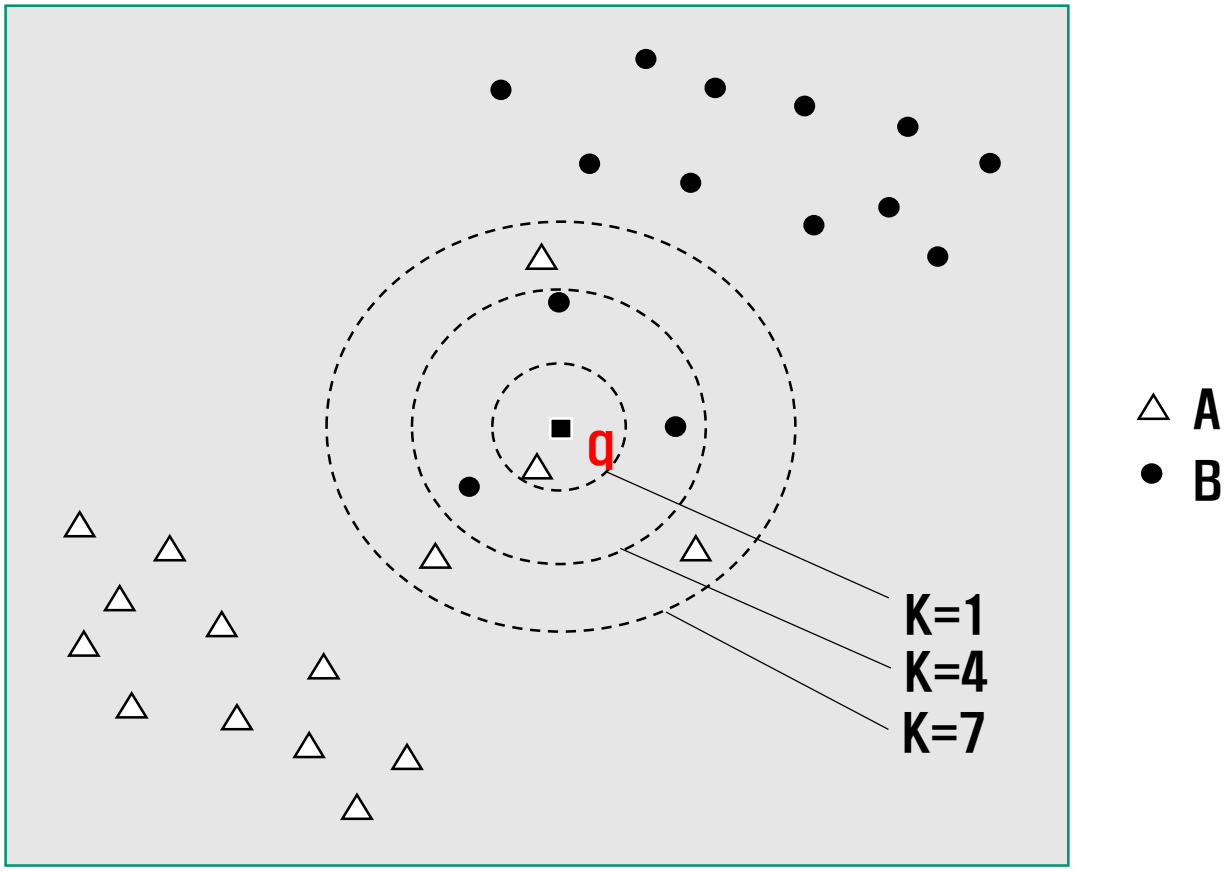
- kNN 알고리즘
  - 주어진 샘플의 특성 값과 가장 가까운 특성을 가진 이웃(neighbor)을 k개 선택
  - 선택된 이웃들 레이블의 평균치로 이 샘플이 속할 category를 예측
- 직관적으로 이해하기 쉬운 분류 알고리즘으로 추천 시스템에서 많이 사용
  - 적절한 추천을 하기 위해서 추천을 요청한 사람의 성향을 특성들로 파악
  - 그 사람과 가장 성향이 유사한 k명의 사람들이 좋아하는 품목을 추천
- 협업 필터링(collaborative filtering)
  - 서점에서 도서 추천
    - 먼저 추천 고객 A와 독서 취향이 비슷한 사람들의 그룹을 찾고
    - 그 그룹에서 평이 가장 좋은 책들 중에서 고객 A가 아직 보지 않은 책을 추천

    (서점은 평소 고객들이 어떤 책을 즐겨 보는지, 책에 대한 평가는 어떠했는지 등 조사
    → 비슷한 취향을 구분)
  - 사용자 기반 협업 필터링(User-based collaborative filtering)
  - 아이템 기반 협업 필터링(Item-based collaborative filtering)
    - a 라는 상품을 좋아한 사람들은 대부분 b나 c 상품도 같이 좋아한다는 사실을 이용
    - 상품을 기준으로 유사한 상품들을 Grouping
    - 어떤 사람이 새로운 상품의 추천을 원할 경우, 유사한 상품 그룹에서 추천

kNN 동작



kNN 동작

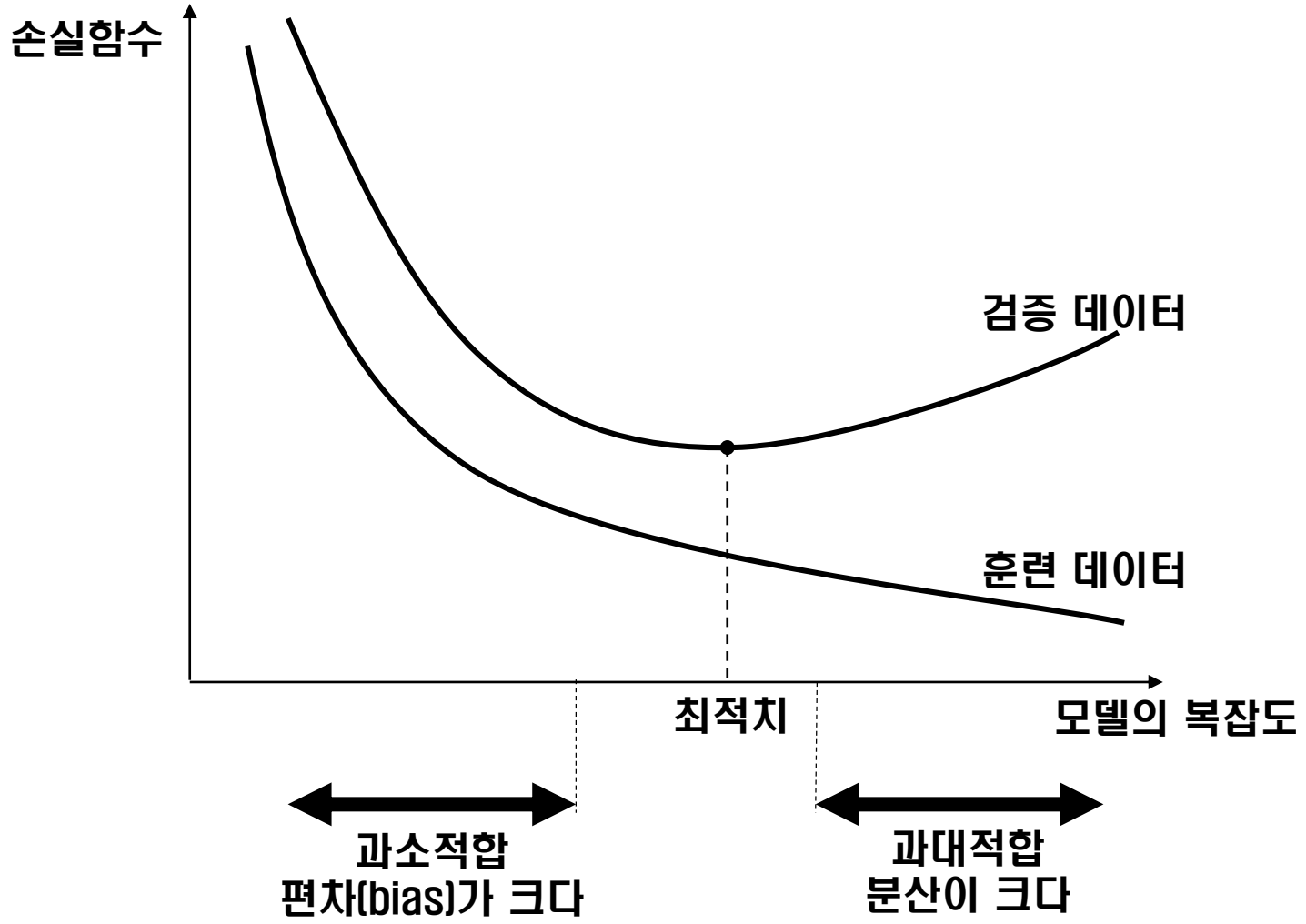


k 값에 따른 영향

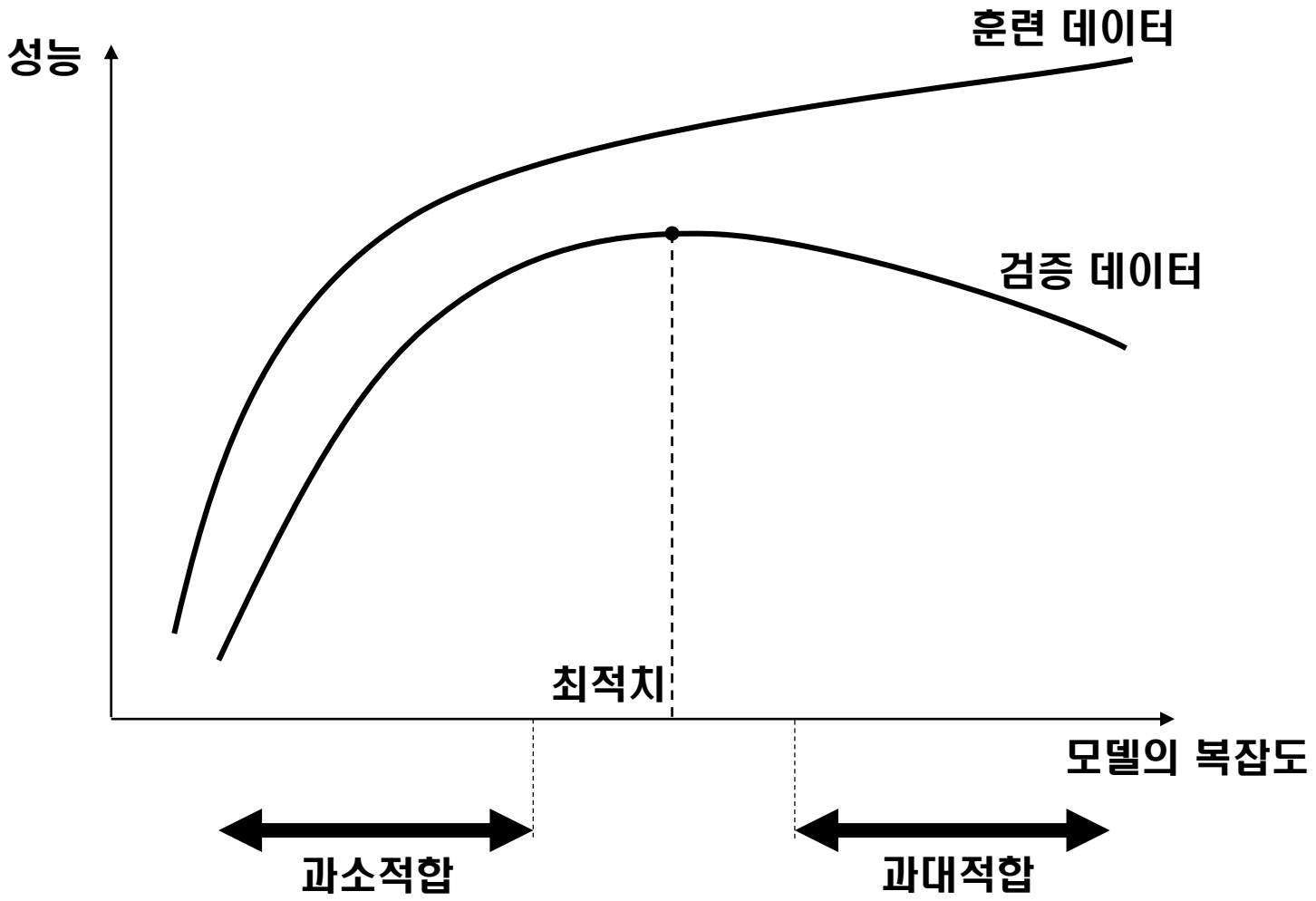
- k 값을 너무 작게 잡으면
  - 정확도가 올라갈 수 있으나
  - 주변 데이터(Noise)에 너무 예민하게 반응 → 과대 적합 가능성
- k 값을 너무 크게 잡으면
  - Noise에는 강하나
  - 주변에 너무 많은 데이터의 평균치를 사용하므로 정밀한 예측이 어려움
    → 과소 적합의 가능성
- 극단적으로 k=N(전체 샘플 수) 으로 하면
  - 항상 전체 데이터의 평균치 값을 예측
  - 영화 추천에서 이는 평균적으로 가장 많은 사람들이 본 영화 즉, 종합 베스트 셀러를 추천하는 것과 같은 결과

* Noise : 좋은 모델을 만드는데 도움이 되지 않는 혼란스러운 정보를 제공하는 샘플

과소 적합과 과대적합 판단 – 손실함수



과소 적합과 과대 적합 판단 – 성능



편향과 분산

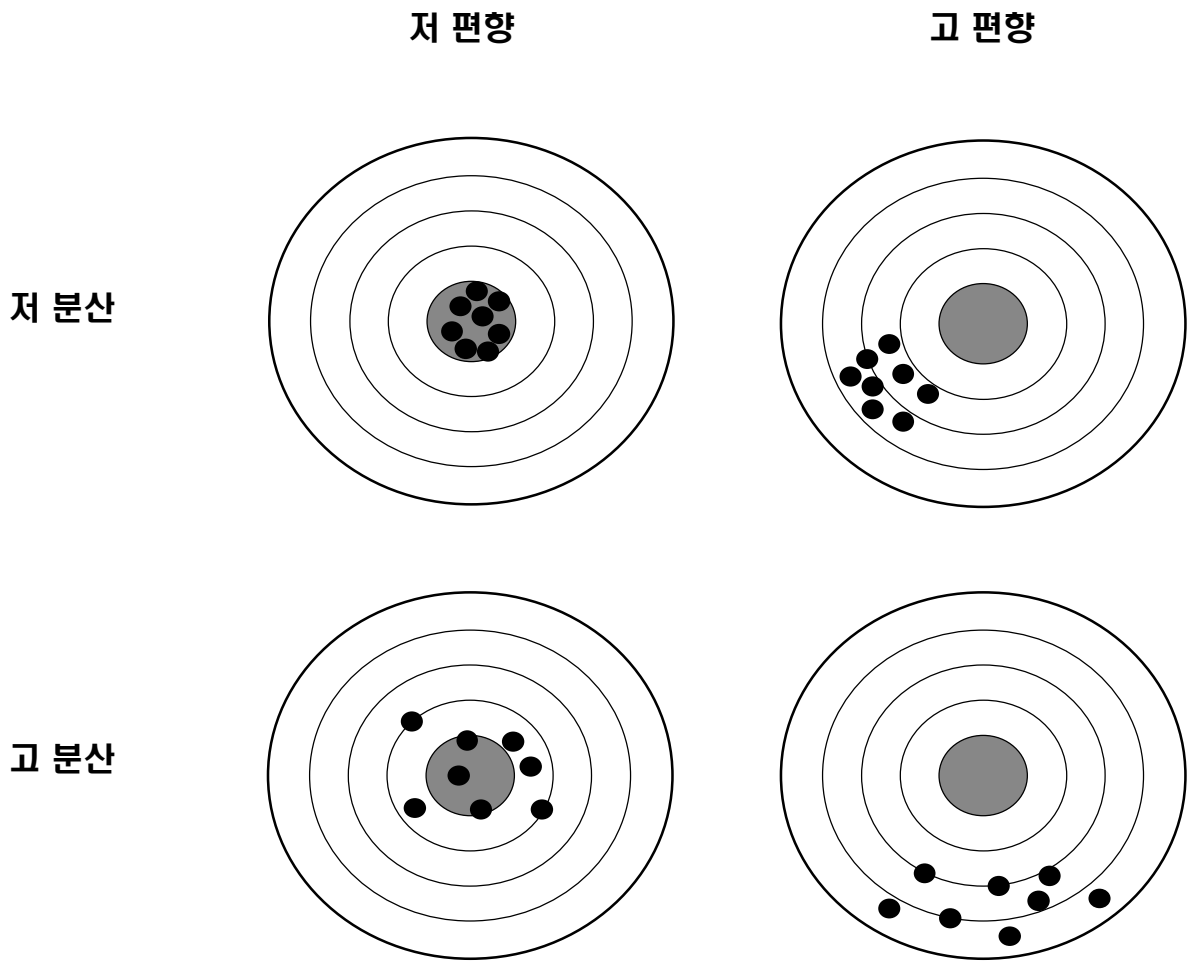
- 예측 모델에서 발생하는 오차는 분산(variance)과 편향(bias) 두가지 성분으로 설명할 수 있다.
- **분산 (Variation)** 이란 모델이 너무 복잡하거나 학습데이터에 민감하게 반응하여 예측 값이 산발적으로 나타나는 것이다.
- **편향 (Bias)** 이란 모델 자체가 부정확하여 피할 수 없이 발생하는 오차를 말한다.

편향과 분산

- 예를 들어 kNN알고리즘에서 k=1인 경우는 모델이 학습 데이터에 민감하게 반응하여 예측의 변화가 커지므로 분산 성분이 증가한다.
  - 그러나 편향은 발생하지 않는다.
  - 이러한 경우를 과대적합되었다고 한다.

- 반면에 k=N인 경우는 모델은 항상 평균치로 동일하게 예측하므로 분산은 거의 발생하지 않는다.
  - 그러나 모델 자체가 부정확하여 편향이 커진다.
  - 모든 사람의 키를 평균치로 예측하면, 예측치는 평균치로 동일하므로 분산은 없어지지만 편향 오차가 클 수 밖에 없다.
  - 이는 과소적합의 현상이다.

편향과 분산



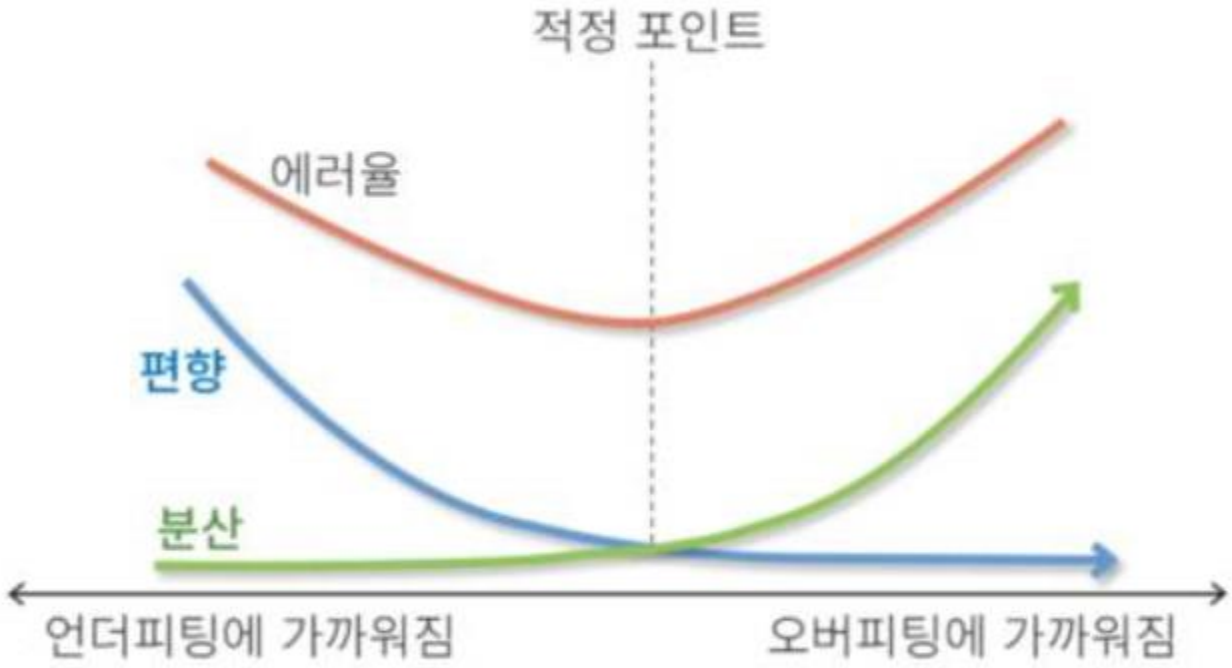
편향과 분산

- 모델이 훈련 데이터에 너무 종속적이거나 정교하면 (k=1 인 경우) 분산이 늘어나고 편향은 줄어든다 (위 그림에서 좌 하)

- 모델이 너무 단순하면 (k=N인 경우) 분산을 줄어드나 모델이 부족하여 생긴 편향이 증가한다.

- 뒤에서 설명할 결정 트리에서는 트리의 깊이(depth)에 따라서 편향과 분산 오류가 달라진다

머신 러닝



kNN 장단점

- kNN의 장점
  - 훈련시간이 거의 없다
  - 알고리즘의 개념이 명확
  - 모델링에 필요한 하이퍼 파라미터 : k 값 하나뿐
  - 특성 변수만 잘 선정하면 예측 성능도 좋다

- kNN의 단점
  - 분류를 처리하는 시간, 즉 알고리즘을 수행하는 시간이 길다
  - 확보한 샘플들을 모두 비교해서 어떤 그룹에 가까운 지 새로 계산해야
  - 또한 새로운 샘플이 계속 추가될 때마다 가까운 이웃이 달라진다
  - Lazy 알고리즘
    - 나중에 계산량이 많은 알고리즘

kNN 알고리즘의 변화

- 샘플들간의 거리에 대한 가중치를 고려
  - 가까이 있는 이웃에 대해서는 가중치를 크게

Decision Tree
(결정 트리)

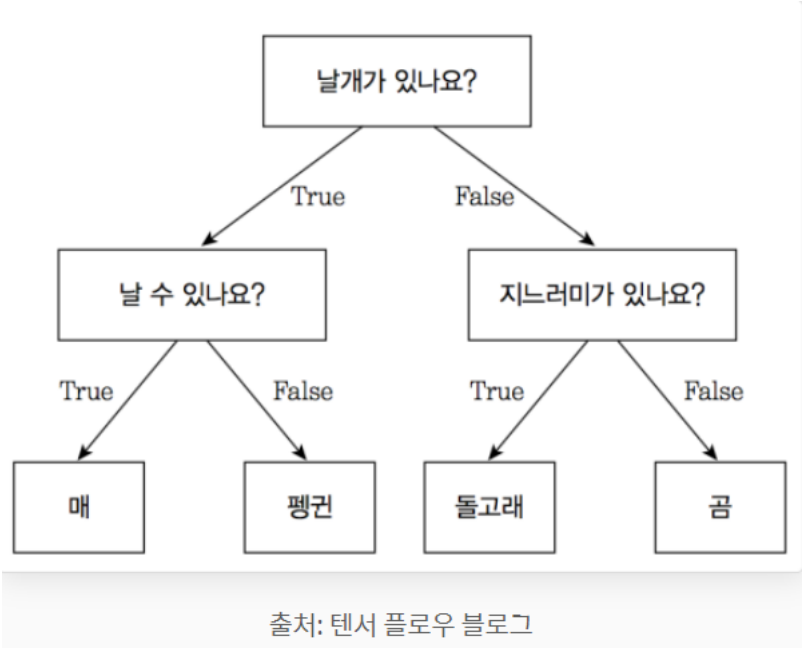
결정 트리 개요

- **선형 모델**
  - **특성들을 대상으로 곱셈과 덧셈과 같은 연산**
  - **그 값을 기준으로 회귀나 분류를 예측**

- **결정 트리(**decision tree)
  - **각 특성을 독립적으로 하나씩 검토하여 분류 작업을 수행**
  - **마치 스무고개 하여 예측을 하듯이 동작 한 번에 한 특성을 따져보는 방법**
  - **분류, 회귀 모두에 사용**
    - **분류용 모델 :** DecisionTreeClassifier() **함수**
    - **회귀분석 모델 :** DecisionTreeRegressor() **함수**

결정 트리 개요

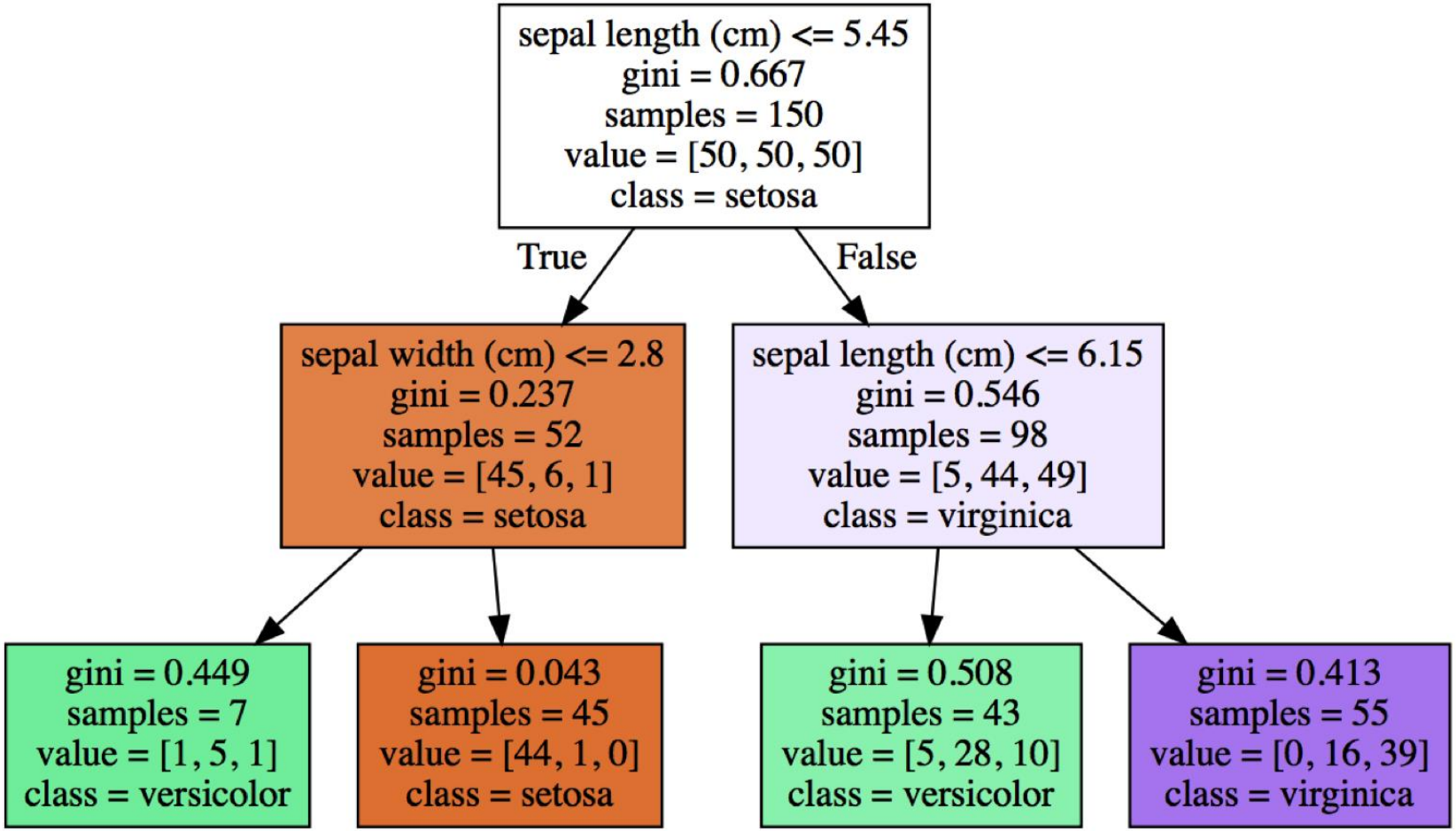
- **결정트리 (예)**
  - 매, 펭귄, 돌고래, 곰을 구분한다고 하자.
  - 매와 펭귄은 날개를 있고, 돌고래와 곰은 날개가 없다.
  - '날개가 있나요?'라는 질문을 통해 매, 펭귄 / 돌고래, 곰을 나눌 수 있다. 매와 펭귄은 '날 수 있나요?'라는 질문으로 나눌 수 있고, 돌고래와 곰은 '지느러미가 있나요?'라는 질문으로 나눌 수 있다. 즉,



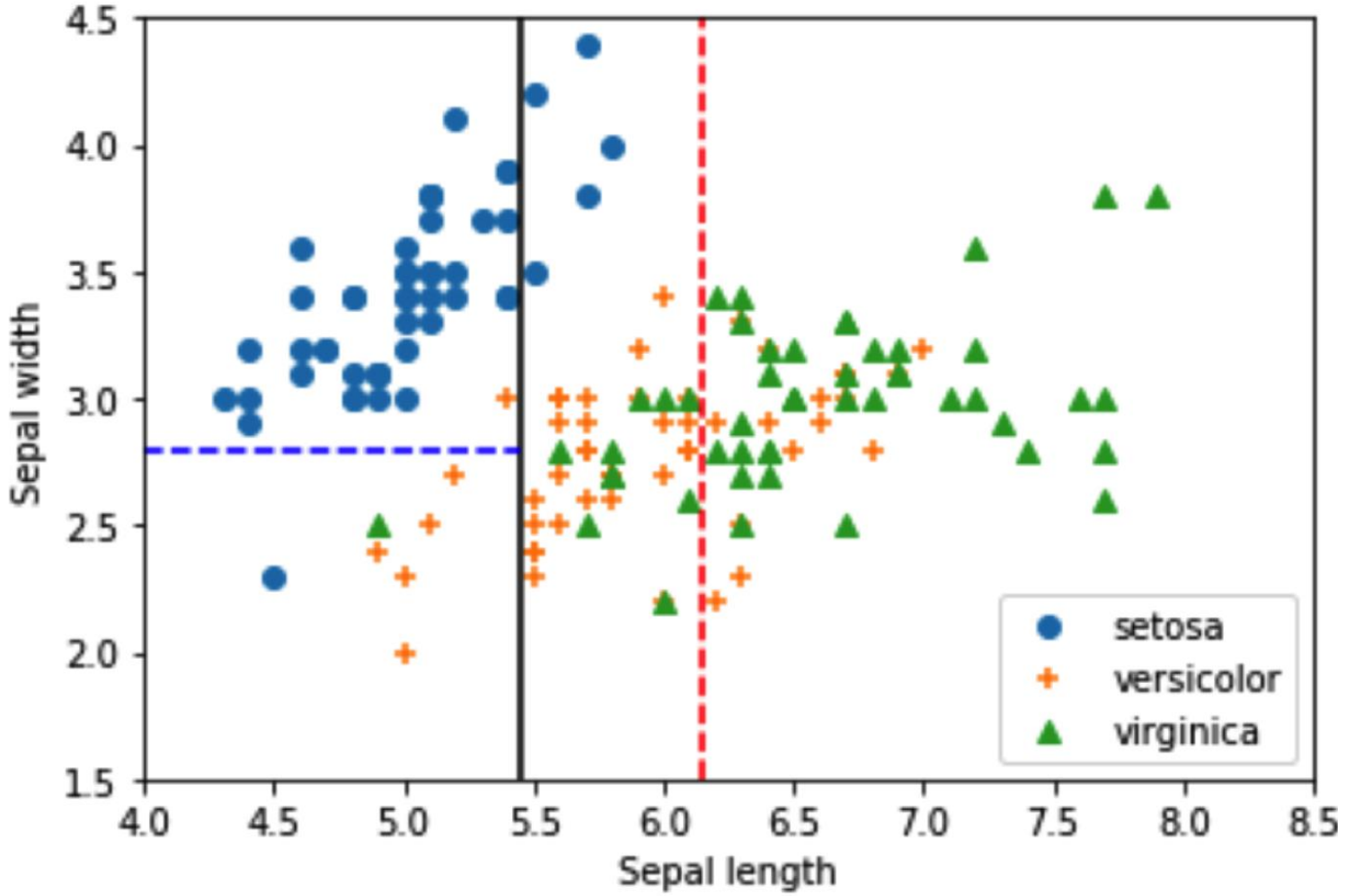
동작 원리

- 결정 트리에서 핵심이 되는 부분
  - 가장 효과적인 분류를 위해서 먼저 어떤 변수를 가지고 판별을 할지 결정하는 것
- 이 판별은 트리를 내려가면서 계속
  - 매 단계마다 어떤 변수를 기준으로 분류를 하는 것이 가장 효과적인지를 찾아야 함
- 그룹을 효과적으로 "잘 나누는 것" 의 기준
  - 그룹을 나눈 후에 생성되는 하위 그룹들에 가능하면 같은 종류의 아이템들이 모이는지를 기준으로 삼는다
  - 한 그룹에 같은 종류의 아이템이 많이 모일수록 순수(pure)하다고 한다
  - 만일 나누어진 하위 그룹이 100% 같은 항목들로만 구성 → 순도(purity) 100%

결정 트리 예


sepal length (cm) <= 5.45
gini = 0.667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
sepal width (cm) <= 2.8
gini = 0.237
samples = 52
value = [45, 6, 1]
class = setosa
sepal length (cm) <= 6.15
gini = 0.546
samples = 98
value = [5, 44, 49]
class = virginica
gini = 0.449
samples = 7
value = [1, 5, 1]
class = versicolor
gini = 0.043
samples = 45
value = [44, 1, 0]
class = setosa
gini = 0.508
samples = 43
value = [5, 28, 10]
class = versicolor
gini = 0.413
samples = 55
value = [0, 16, 39]
class = virginica

결정 트리 예


Sepal width
Sepal length
4.5
4.0
3.5
3.0
2.5
2.0
1.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
setosa
versicolor
virginica

판별 기준 – 불순도(impurity)

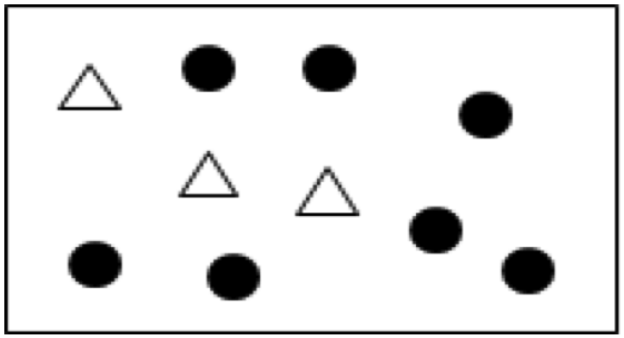
- 결정트리는 나누려는 그룹의 순도가 가장 높아지도록 그룹을 나누어야 한다.
- 불순도(impurity): 해당 범주 안에 서로 다른 데이터가 얼마나 섞여 있는지를 뜻함.
- 그룹의 순도를 표현
  - 지니(Gini) 계수, 엔트로피(entrophy) 주로 사용
- Gini 계수

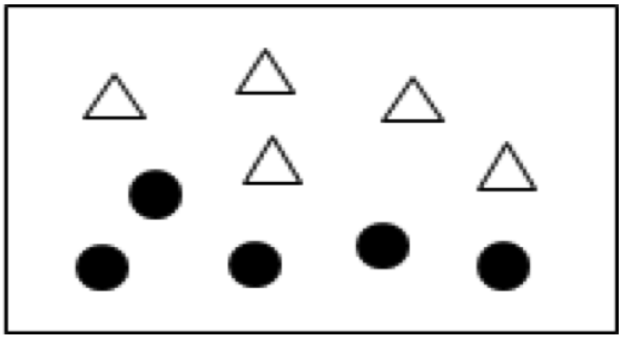
$$Gini = 1 - \sum_{k=1}^{m} p_k^2$$

- 엔트로피 (Entropy)

$$Entropy = -\sum_{k=1}^{m} p_k \log_2(p_k)$$

판별 기준





$$\text{좌측 박스: 지니}(7:3) = 1 - \left[\left(\frac{7}{10}\right)^2 + \left(\frac{3}{10}\right)^2\right] = 1 - (0.49 + 0.09) = 0.42$$

$$\text{우측 박스: 지니}(5:5) = 1 - \left[\left(\frac{5}{10}\right)^2 + \left(\frac{5}{10}\right)^2\right] = 1 - (0.25 + 0.25) = 0.5$$

(*) Worst: 0.5
Best: Gini=0 (all in one class)

정보량

- 데이터(이벤트)가 포함하고 있는 정보의 총 기대치, 정보의 가치
- 정보량을 표현: 해당 사건이 발생할 확률(probability)을 사용
- 사건 발생 확률에 따른 정보 가치
  - 확률 = 1 : 정보가 주는 가치가 없다
  - 사건 발생 확률이 낮을수록 : 정보가 주는 가치가 높음
- 정보량: 일어날 확률의 역수에 비례

정보량 정의 => $log\left(\frac{1}{p}\right)$

정보량과 엔트로피

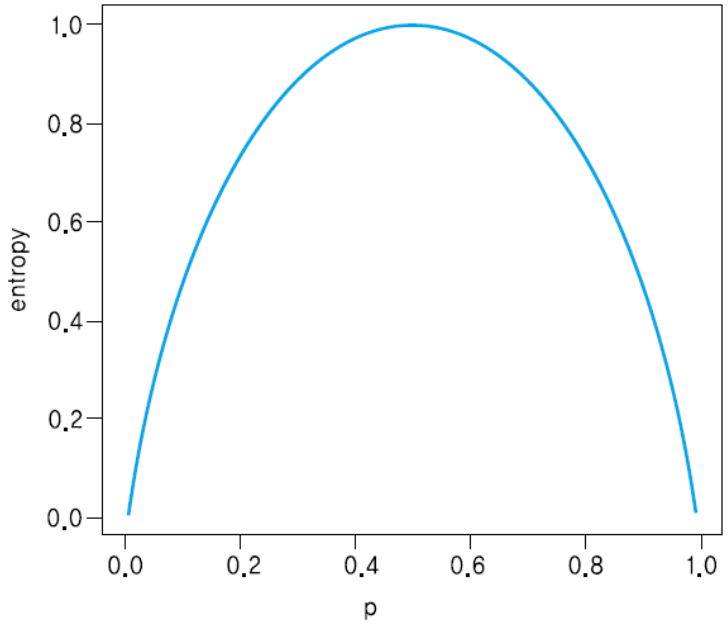
- 정보량의 기대치
  - 어떤 사건이 갖는 가치와 그 사건이 발생할 확률의 곱
  - 이를 엔트로피(entropy)라고 함

- 엔트로피(정보량의 기대치)

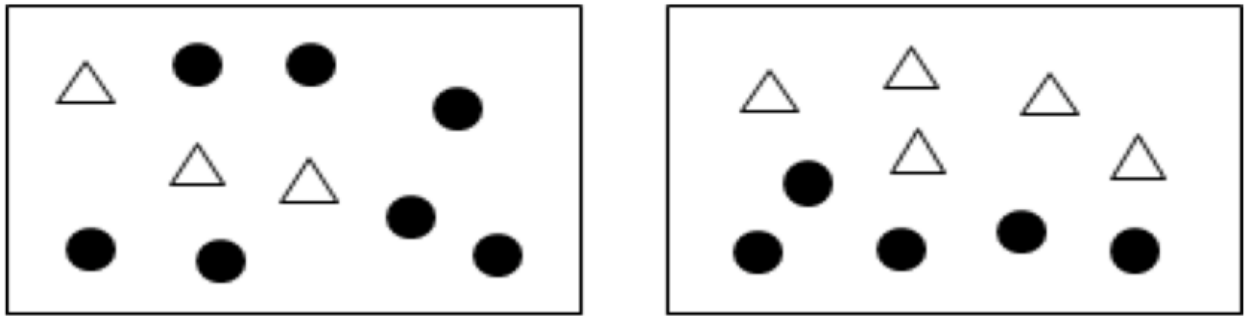
$$p \log\left(\frac{1}{p}\right) = -plog(p)$$

정보량과 엔트로피

- 바이너리(binary) 사건의 경우
  - 엔트로피는 p가 0.5 일 때 가장 높음
- 즉, 불확실성이 가장 높을 때 엔트로피가 가장 높음



정보량과 엔트로피



$$Entropy = -\sum_{k=1}^{m} p_k \log_2(p_k)$$

(p_k: 한 영역 안에 존재하는 데이터 가운데 범주 k 에 속하는 데이터 비율)

$$\text{좌측}(7:3) = -\left[0.7^*\log_2(0.7) + 0.3^*\log_2(0.3)\right]$$
$$= -\left[(-0.36) + (-0.52)\right] = -(-0.88) = 0.88$$

$$\text{우측}(5:5) = -\left[0.5^*\log_2(0.5) + 0.5^*\log_2(0.5)\right]$$
$$= -\left[(-0.5) + (-0.5)\right] = -(-1) = 1$$

지니 계수와 엔트로피

- 결정 트리의 성능
  - Gini 계수를 사용하든, entropy를 사용하든 성능에는 큰 차이가 없다

- 분류 속도(계산량)
  - Gini 계수의 계산 속도가 조금 빠르다 (entropy의 경우 log 연산 포함)

트리 종료 조건

- 결정 트리를 계속 만들어 상세하게 분류를 하면
  - 언젠가는 훈련 데이터에 대해서 100% 순도의 분류가 가능
  - 이는 과대적합 → 테스트 데이터에 대해서는 성능이 오히려 떨어지게 됨

- 결정 트리 모델의 트리 깊이
  - 깊이를 제한하지 않으면 과대적합할 위험이 높으므로 주의해야
  - 트리의 깊이를 적절한 값보다 너무 작게 제한하면 과소적합이 됨

트리 종료 조건 – 하이퍼 파라미터

- max_depth: **트리의 최대 깊이**
  - **이보다 깊은 트리를 만들지 않는다**
- max_leaf_nodes: **리프 노드의 최대 수**
  - **리프 노드를 이보다 많이 만들지 않는다**
- min_samples_split: **분할하기 위한 최소 샘플수**
  - **이보다 작으면 분할하지 않는다**
- min_samples_leaf: **리프 노드에 포함될 최소 샘플수**
  - **이보다 작은 노드는 만들지 않는다**
- max_features: **최대 특성수**
  - **분할할 때 이보다 적은 수의 특성만 사용한다**

내부 변수

- **결정 트리 모델을 만든 후에, 어떤 특성이 결정 트리를 생성할 때 중요한 역할을 했는지 비중을 파악 가능**
- **이 결과를 보고 중요하지 않은 특성은 향후에 제외하기도 함**
- **내부 변수로 확인**
  - feature_importances_

클래스 확률

- **테스트 결과**
  - **테스트 샘플이 속할 가장 확률이 높은 클래스 하나만 알려준다**
  - **이 샘플이 각 클래스에 속할 확률을 각각 알려주는 것도 가능**
- **소프트 투표(**soft voting**) 도입**
  - **보다 정확한 다중 분류를 수행할 수 있다**
  - **각 클래스에 속할 확률 :** predict_proba() **함수를 사용**

결정 트리의 특징

- 거의 모든 종류의 분류에 가장 많이 사용하는 범용 모델
- 장점
  - 알고리즘의 동작을 설명하기 수월함
    - 대출이 왜 거부되었는지
    - 당신의 신용도가 왜 낮은지
    - 왜 불합격되었는지 등
  - 특성 변수간의 연산이 없기 때문에 변수의 스케일링이 필요 없다
    - (분류 작업을 수행) 한번에 한 특성 변수씩 검토하여 어떤 기준으로 트리를 나누어 나가면 순도가 올라가는지만 점검하면 됨
- 단점
  - 훈련 데이터가 바뀌면 모델의 구조가 달라진다
  - 훈련 데이터에 따라 바뀌는 모델을 남에게 설명하기 어려울 수도

랜덤 포레스트

랜덤 포레스트(Random Forest)

- 결정 트리의 성능을 개선
- 비교적 간단한 구조의 결정 트리들을 수십~수백개를 랜덤하게 만들고 각 결정 트리의 동작 결과를 종합하여 판정
- 앙상블(ensemble) 방법
  - 여러 개의 모델을 만들고 평균을 구하는 방식
  - 하나의 모델만 만드는 것보다 좋은 성능을 보임
- 주어진 훈련 데이터를 모두 한 번에 사용해서 하나의 최상의 트리 모델을 만드는 방식이 아니라, 데이터의 일부 또는 속성의 일부만 랜덤하게 채택하여 결정 트리를 다양하게 만들고 그 결과의 평균치를 취하는 방식
- 나무(tree)가 많이 모였다는 의미로 숲(forest)라는 용어를 사용

개념 example

- 예1) 어떤 사람이 얼마나 훌륭한지를 평가
  - (방법 1)
    - 그 사람을 아는 모든 사람(예, 1,000명)에게 묻고 수집된 데이터로 한번에 평가
  - (앙상블 방법)
    - 그 사람을 아는 사람들을 랜덤한 조합(예, 50명씩 선택)으로 총 300개의 조합을 만들어 평가 → 이들 평가 점수의 평균치로 평가

- 예2) 나의 건강 정보에 대한 진단
  - (방법 1)
    - 나의 모든 건강 정보를 가장 훌륭한 의사 한 명에게만 주고 진단
  - (앙상블 방법)
    - 나의 모든 건강 정보의 일부를 랜덤하게 선택하는 것을 100번 수행하여 100명의 의사에게 각각 진단을 구하여 이들의 평균치(앙상블)로 최종 진단

- 일반적으로 앙상블 방법이 더 우수한 것으로 알려져 있다

특징

- 다수의 결정 트리를 생성하고 이의 평균을 구하면 단일 결정 트리를 사용하는 것보다 안정적이고 우수한 성능을 낸다.
  - 샘플도 랜덤하게 선택, 속성도 랜덤하게 선택
- 성능이 우수한 하나의 모델을 사용하는 것보다, 각각의 성능이 최상이 아니지만 다수의 모델을 사용하고 평균치를 구하는 방식이 더 우수
  - 이를 대중의 지혜, 큰 수의 법칙 등으로 설명하기도 함
- 단점
  - 모델의 동작을 한가지 트리를 선택하여 설명하기가 어렵다
  - 계산량이 많아진다

앙상블 기법

- 앙상블 기법
  - 여러 개의 작은 모델의 평균 값을 구하거나, 투표를 통하여 최적의 값을 찾는 절차 필요
  - 투표 방식
    - 직접 투표
    - 간접 투표

직접 투표와 간접 투표

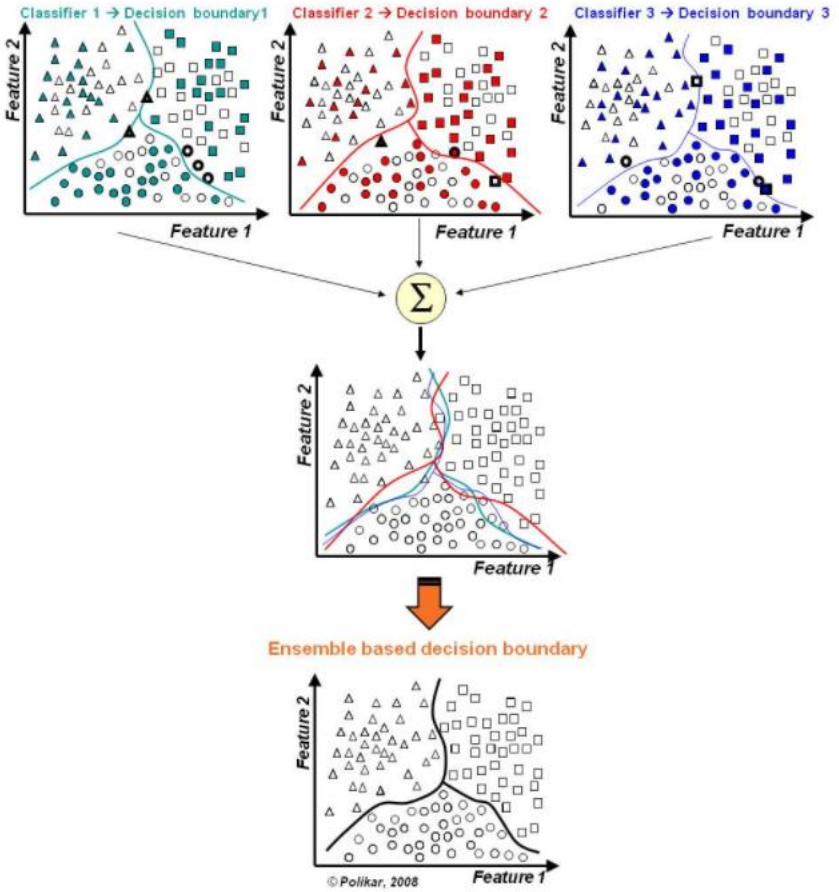
- 직접 투표(hard voting)
  - 각 세부 모델이 예측한 최종 class에 1점 부여
  - 최종 target 변수
    - 가장 많은 점수를 받은 class
- 간접 투표(soft voting)
  - 각 세부 모델이 각 class에 속할 확률을 제공
  - 최종 target 변수
    - 각 세부 모델이 제공한 확률을 모두 더해서 가장 큰 값을 받은 class
  - (회귀 적용) 최종 예측 값
    - 각 세부 모델이 예측한 여러 수치 값의 평균치
- 어떤 투표 방식이 좋을까?
  - 문제의 성격에 따라 다르다

간접 투표(soft voting)

	P일 확률	Q일 확률	판정결과 (직접투표)
세부 모델 A	0.9	0.1	P
세부 모델 B	0.4	0.6	Q
세부 모델 C	0.3	0.7	Q
확률의 평균 (간접 투표)	(1.6)/3 = 0.533	(1.4)/3 = 0.456	P or Q

Ensemble 방법

- **Bagging (배깅)**
  - 중복을 허용한 n 개의 샘플을 추출하여 평균을 구하는 작업을 m 번 반복.
  - Overfitting 을 효율적으로 줄이고 일반적인 모델을 만드는데 집중.
  - (ex) RandomForest

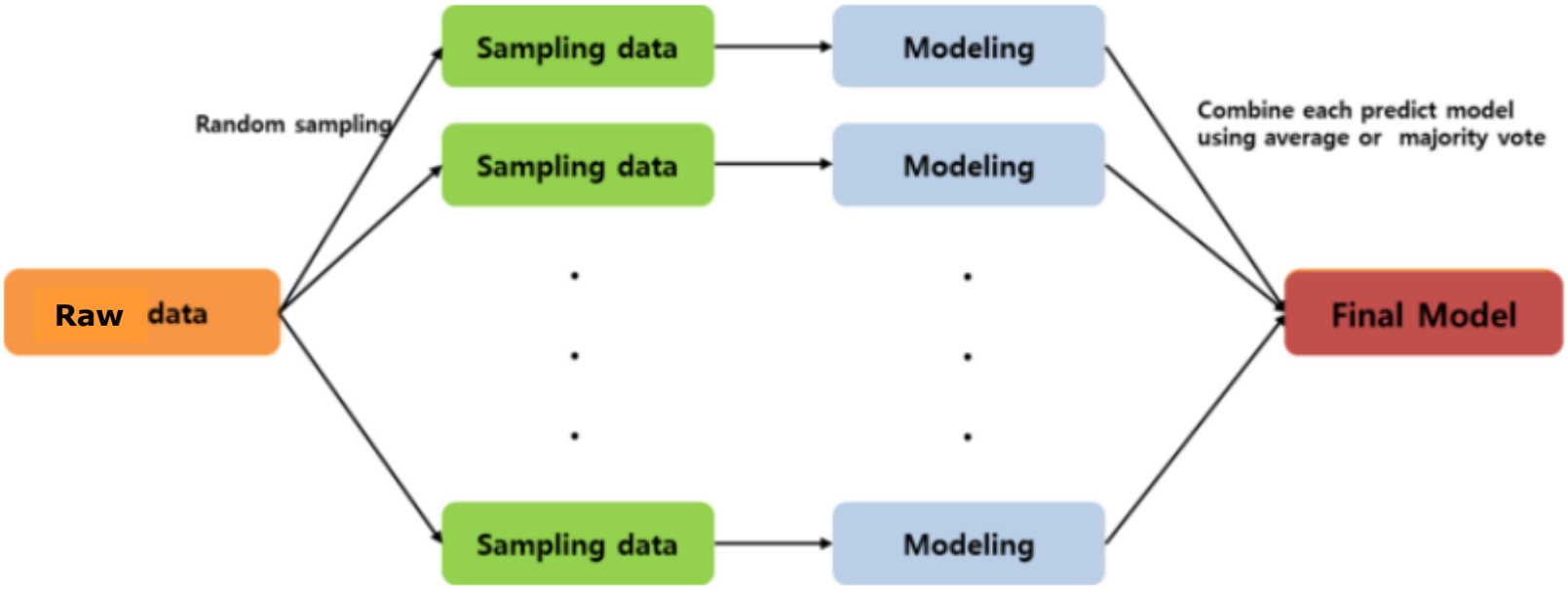


출처: https://swalloow.github.io/bagging-boosting

배깅

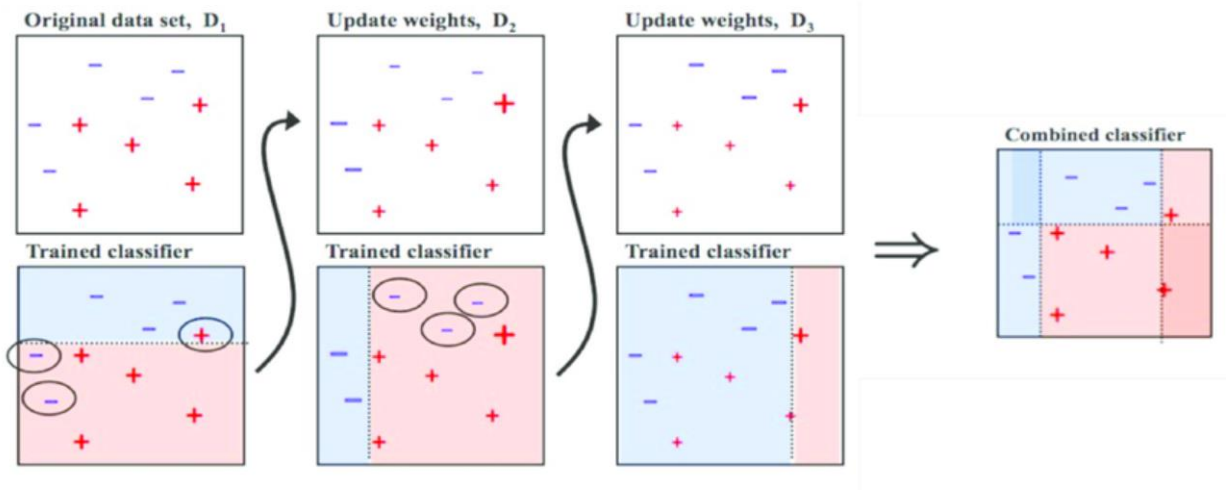
- **배깅(**bootstrap aggregation)
  - **전체 훈련 데이터에서 "중복을 허용" 하여 데이터를 샘플링을 하는 방법**
  - bootstrap resampling**의 줄임말로 부트 스트래핑이라고도 함**
  - **목적 : 부족한 훈련 데이터를 효과적으로 늘리기 위함**
- **페이스팅(**pasting)
  - **배깅과 달리 주어진 원래 데이터에서 중복을 허용하지 않고, 즉, 한 번 샘플링 된 것은 다음 샘플링에서 제외하는 방식**
- **배깅을 수행하면 학습에 선택되지 않는 샘플은 평균 37% 정도**
  - **이 샘플을** oob(out of bag) **샘플이라고 함**
  - **이** oob **데이터는 훈련에 사용되지 않았으므로 검증에 사용하기에 좋다**
- **랜덤 포레스트 모델**
  - **결정 트리 구조에 배깅을 적용한 방식**

배깅


Raw data
Random sampling
Sampling data
Sampling data
Sampling data
Modeling
Modeling
Modeling
Combine each predict model
using average or majority vote
Final Model

Ensemble 방법

- **Boosting (부스팅)**
  - Bagging은 독립적인 input data를 가지고(복원 추출) 독립적으로 예측하지만, 부스팅은 **이전 모델이 다음 모델에 영향을 준다. (틀린 부분에 더 큰 가중치)**
  - Bagging과 다르게 일반적인 모델에 집중되어 있지 않고, 맞추기 어려운 문제를 맞추는데 초점
  - (ex) XGBoost, AdaBoost, GradientBoost



출처: Medium (Boosting and Bagging explained with examples)

그리드 탐색

- 그리드 탐색
  - 여러가지 hyper parameter들의 조합에 대해서 가능한 경우를 모두 수행해 보고 가장 성능이 좋은 경우를 찾는 방식
  - 하이퍼 파라미터가 가질 수 있는 전체 범위를 몇 개의 구간으로 나누어 일일이 하나씩 점검
    - SVC 모델의 경우, gamma 변수와 C 변수의 값을 각각 5가지, 4가지로 나누고 총 4x5=20가지 경우를 시도
    - 이 중에서 최고의 score를 얻는 gamma와 C 값을 찾는다.
- 하이퍼 파라미터 예시
  - kNN: k값
  - 결정 트리: 트리의 깊이, 분류 조건, 분류를 위한 최소 샘플수
  - SVM: 커널 타입, 커널 계수, 규제화 파라미터(감마, C)
  - 랜덤포레스트: 트리수, 사용할 특성수, 분리 조건, 분류할 최소 샘플수
  - 그라디언트 부스팅: 트리수, 학습률, 트리 깊이, 분할할 조건, 분류할 최소 샘플 수

그리드 탐색

- 과대적합을 피하기 위한 규제화용 파라미터도 하이퍼 파라미터
  - 커널 SVM에서 감마 파라미터를 조절
  - 신경망에서는 드롭 아웃율 등
- GridSearchCV() 함수를 사용
  - 그리드 탐색을 하며 동시에 교차검증을 수행하여 예상되는 성능을 측정하기에 편리
  - GridSearchCV()로 생성한 모델 객체는 fit, predict, score 함수를 제공하며 fit을 호출할 때, 여러 파라미터 조합에 대해서 교차검증을 수행
  - best_estimator_ : 선택된 최적의 hyper parameter 값
  - best_score_ : 평가 점수

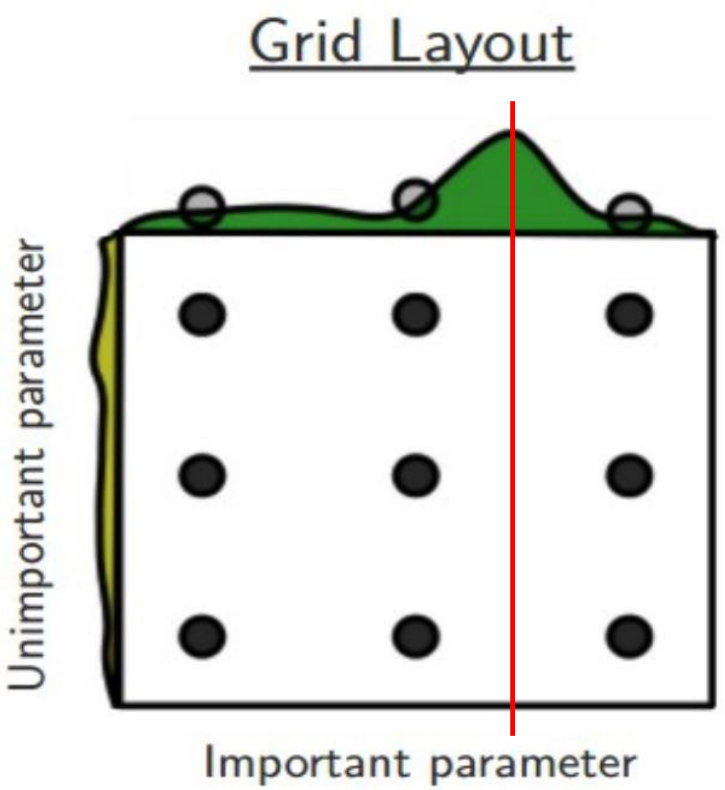
그리드 탐색

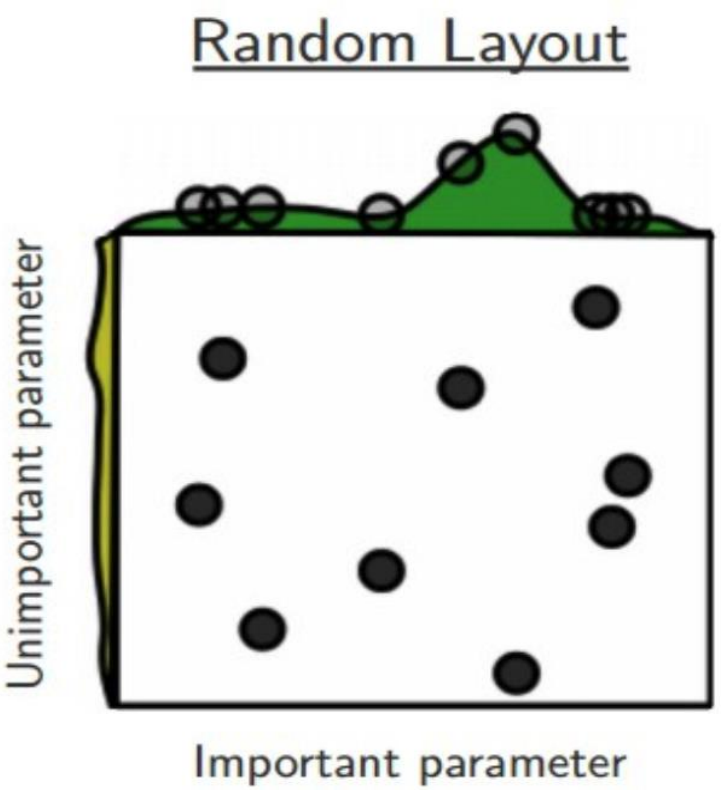
- 그리드 탐색의 특성
  - 단순하나
  - 여러 경우의 수를 모두 탐색하는데 시간도 오래 걸리고
  - 최적의 값을 놓치는 경우가 있다.
  - 격자 모양의 파라미터 조합의 값 사이에 실제로 최적의 하이퍼 파라미터 값이 있었다면 이를 찾아낼 방법이 없다.

랜덤 탐색

- 랜덤 탐색
  - 그리드 탐색의 단점 개선
    - 최적값을 놓치는 문제점
    - 탐색 시간도 감소
  - 두 단계로 수행
    - 1단계 : 일정한 범위 내에서 랜덤하게 하이퍼 파라미터를 선택하여 성능을 실험하여 대체로 어떤 영역에서 성능이 좋은지를 찾는다.
    - 2단계 : 다음에는 이 영역을 중심으로 세밀하게 탐색을 한다.
  - RandomizedSearchCV()함수를 사용

그리드 탐색 vs 랜덤 탐색



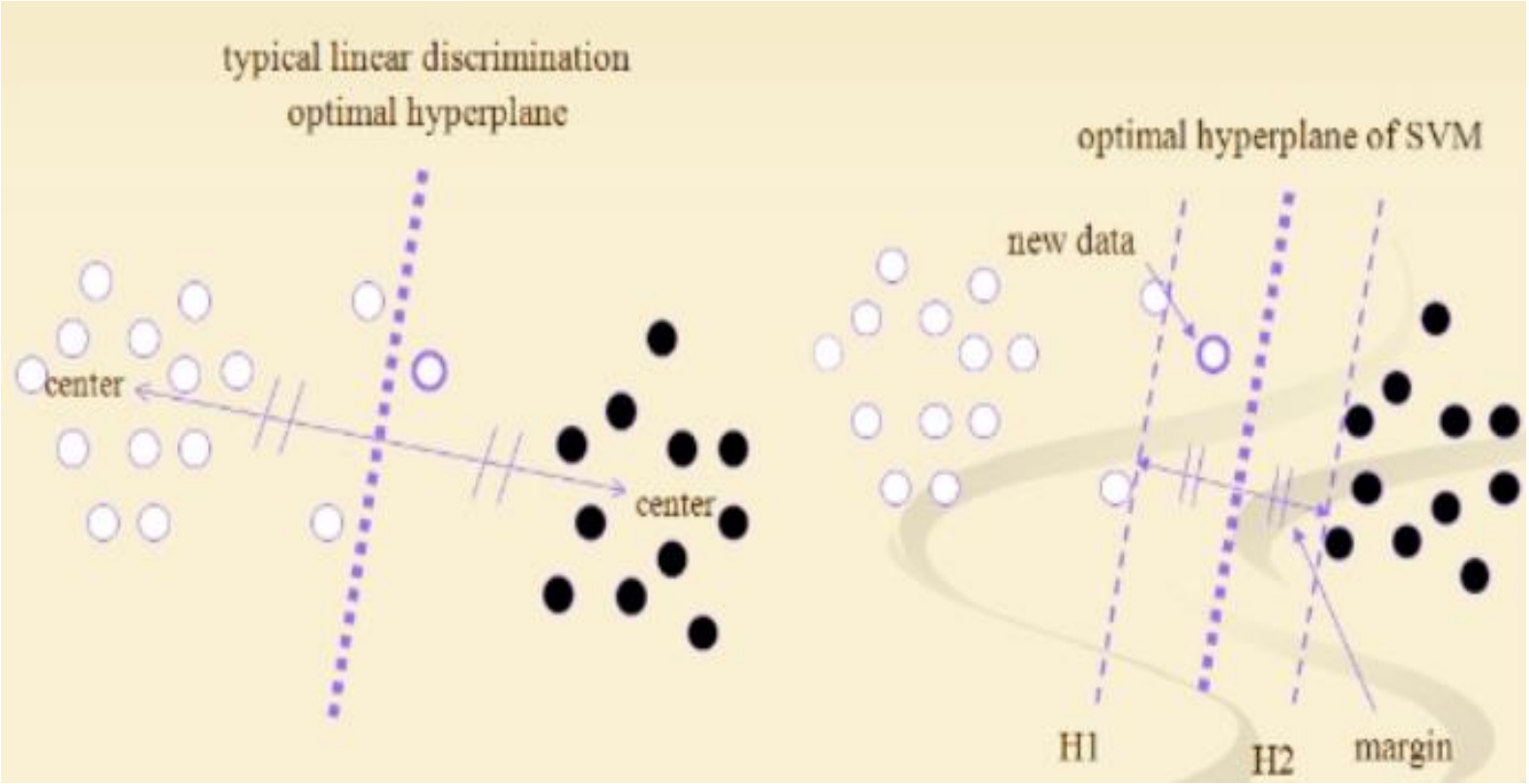


부스팅 알고리즘

- **부스팅 알고리즘**
  - **앙상블 방법 중의 하나**
  - **앞의 모델을 보고 성능을 순차적으로 점차 개선하는 방식으로 동작**
  - **병렬로 처리하지 못함 (cf, 랜덤포레스트 : 병렬 처리 가능)**
- **부스팅 알고리즘의 종류**
  - **아다 부스트와 그라디언트 부스트가 널리 사용**
- **아다 부스트(Adaptive Boosting)**
  - **앞에서 사용한 세부 모델에서 과소 적합했던 샘플, 즉 분류에 실패한 샘플의 가중치를 높여주는 방법**
  - **즉, 소외되었던 샘플을 주목하여 학습을 다시 시키는 방식**
- **Gradient Boosting**
  - 이전 Round의 분류기의 데이터 별 오류를 예측하는, 또 다른 새로운 약한 분류기를 학습시킴. 즉, 데이터의 오차를 학습하는 또다른 분류기를 형성

SVM

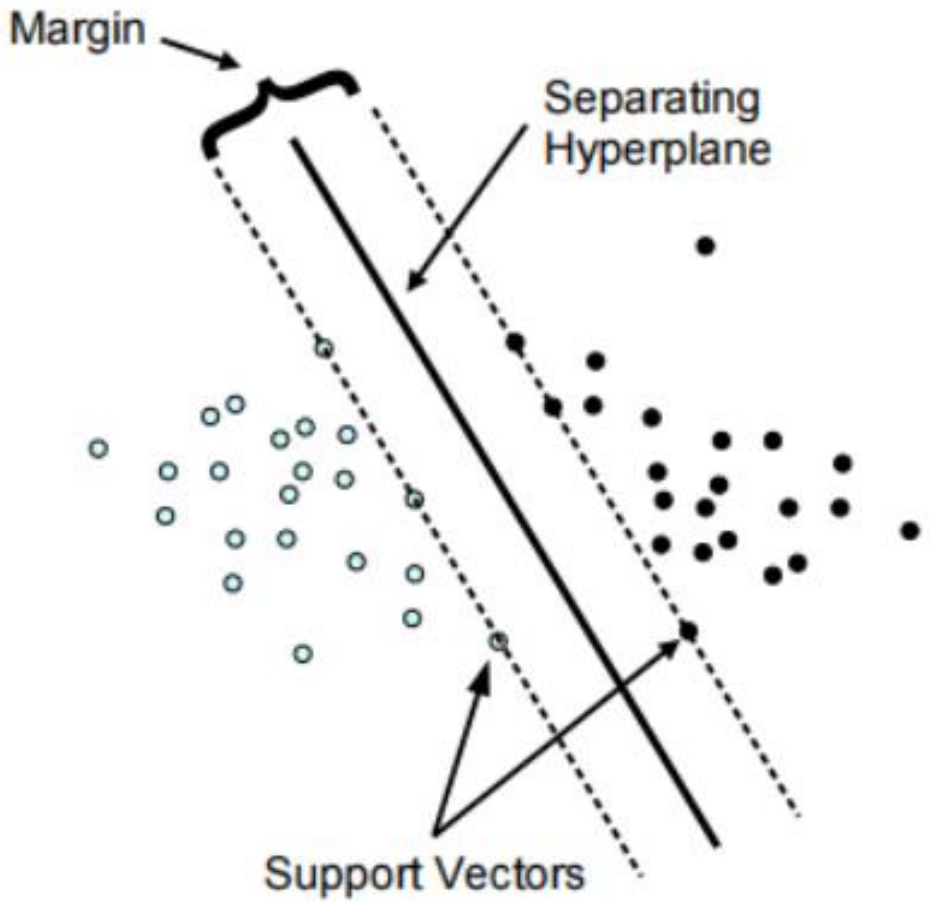
선형 분류 vs SVM

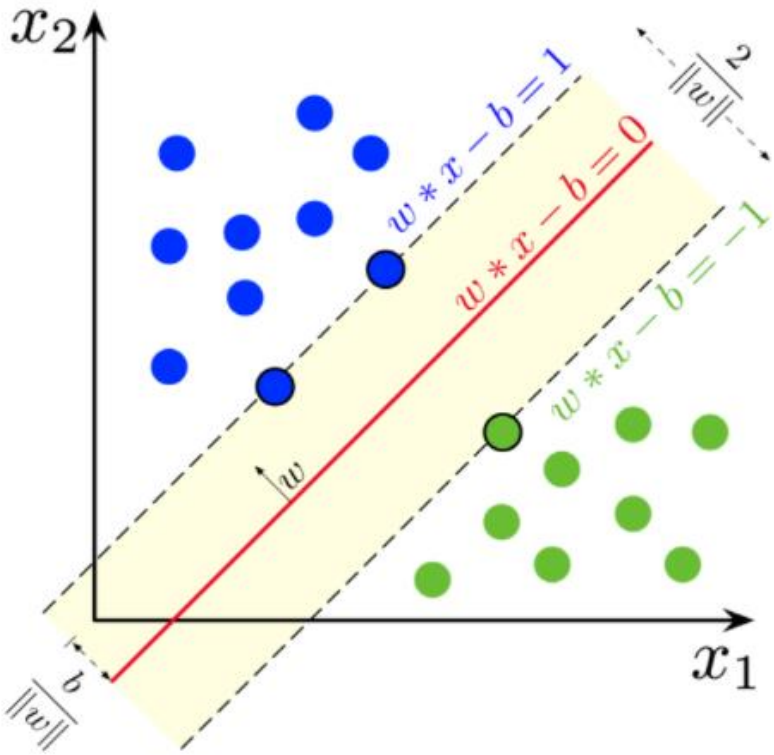

typical linear discrimination
optimal hyperplane
center
center
optimal hyperplane of SVM
new data
H1
H2
margin

마진(Margin)

- **support vector**
  - **각 class의 data vector들로부터 판별 경계까지의 거리 중 가장 짧은 것**
- **margin**
  - **support vector와 판별 경계 사이의 거리**
- **margin이 클수록 → 분류를 잘 하는 것**
  - **학습 데이터 중에서 noise를 무시하는 효과**
    - **즉, 과대 적합을 피할 수 있다**
  - **학습 데이터에 대해서 일정 오차를 내야 일반화 능력이 좋아진다**
- **Multi-class 의 경우 (m classes)**
  - **One-vs-rest (one-vs-all) : m hyperplanes (각 class 와 나머지); 각 hyperplane 에 대해 계산하고 확률값이 가장 큰 것을 output 으로 선택**
  - **One-vs-one : mC2 hyperplanes (모든 쌍의 classes 에 대해) :** mC2개의 분류기가 M 개의 class에 대해서 투표, 가장 높은 점수의 class 선택

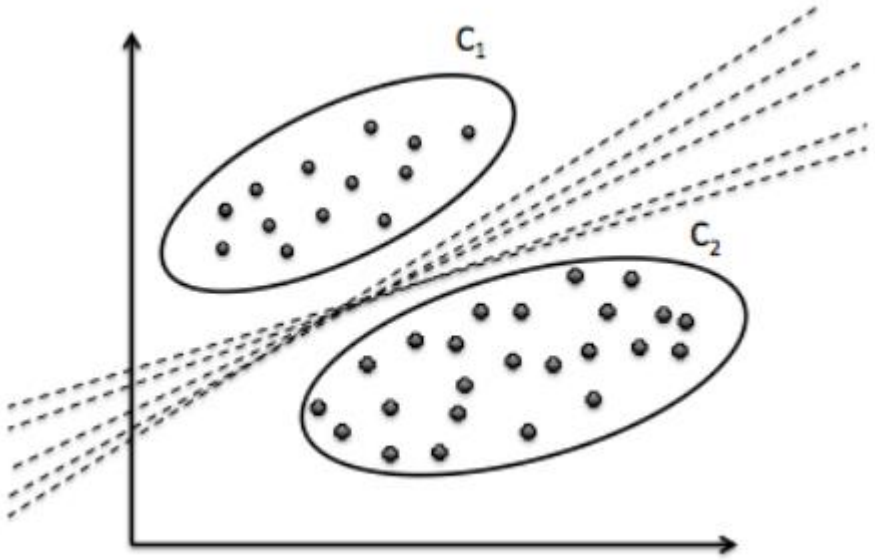
마진(Margin)


Margin
Separating
Hyperplane
Support Vectors


x_2
x_1
$w * x - b = 1$
$w * x - b = 0$
$w * x - b = -1$
w
$\frac{2}{\|w\|}$
$\frac{b}{\|w\|}$

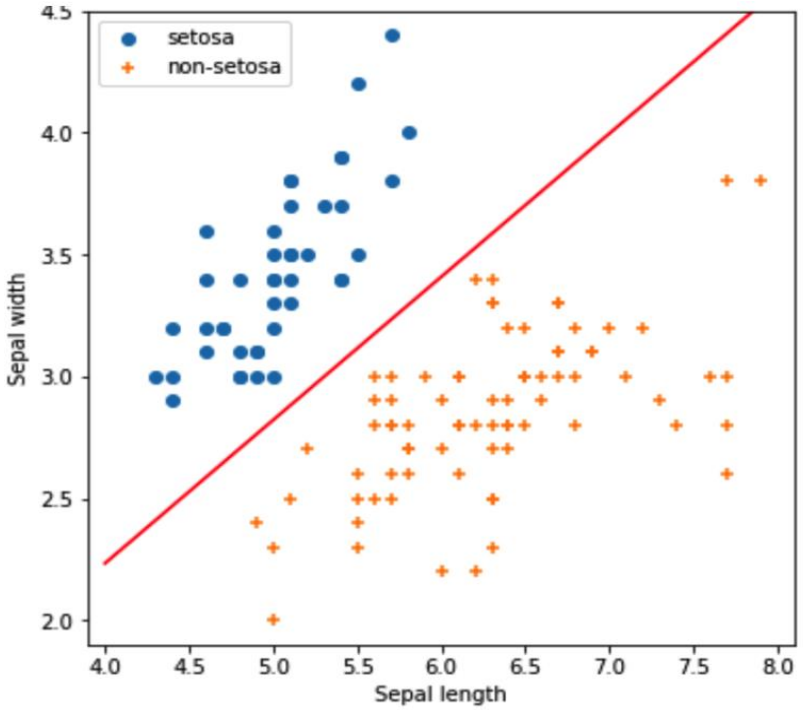
마진(Margin) – 판별 경계 선택 기준

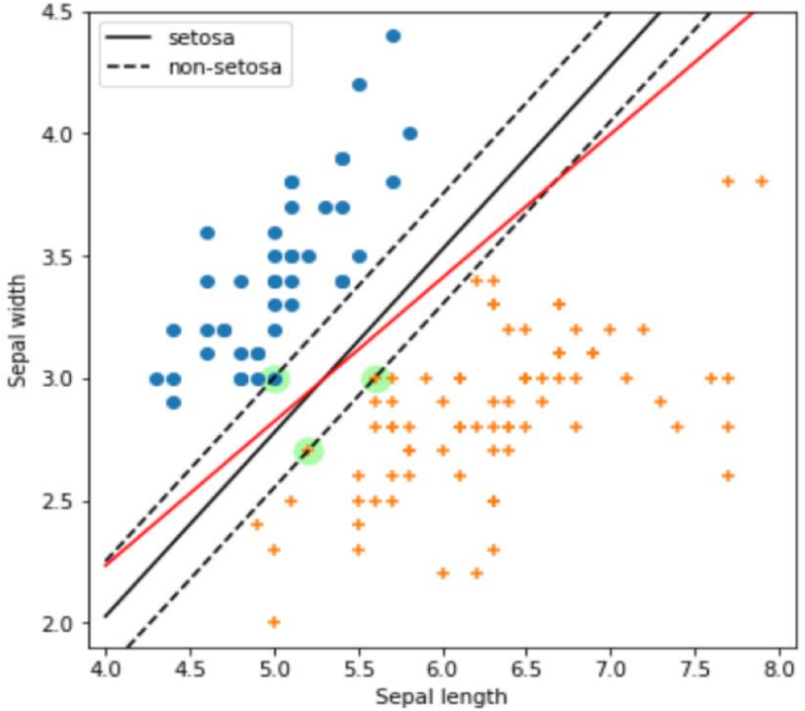
- 여러 개의 판별 경계 후보가 있을 때 선택 기준
  - margin : 넓은 것



서포트 벡터 머신(SVM)

- 선형 모델의 성능을 개선한 방법으로 널리 사용
- 분류 시에 결정 경계를 가능한 **마진을 넓게** 갖도록 만든 방식
  - 샘플들을 단순히 나누기만 하는 것이 아니라 가능한 거리를 멀리 나눌 수 있는 경계면을 찾는 작업을 한다





SVM의 특징

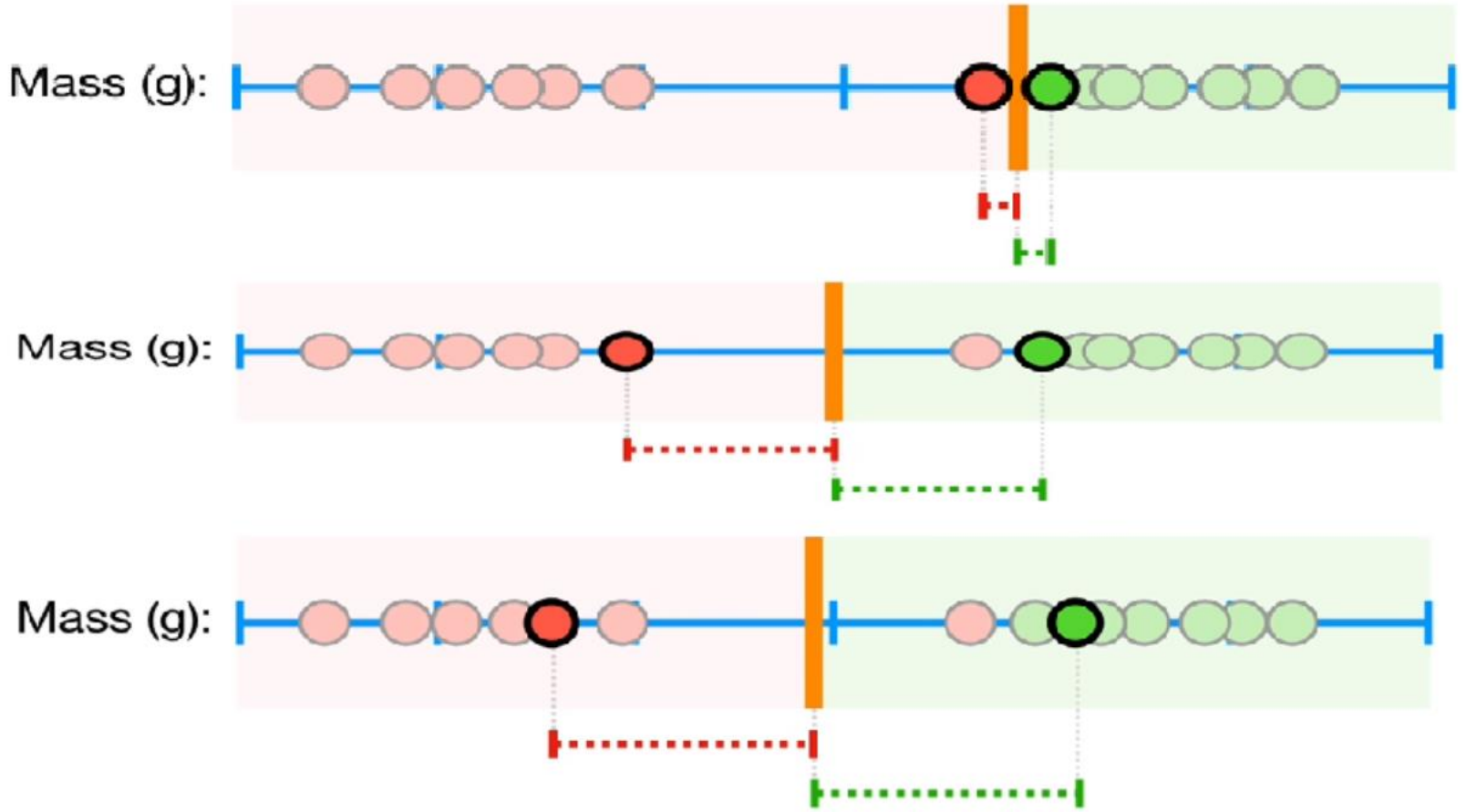
- 선형 분류 직선이 분류를 수행하는데 문제가 없어 보인다. 그러나 이러한 경계면은 새로운 샘플 데이터에 대해서는 잘 동작하지 않을 것이다
- 두께가 없는 선으로 경계를 만드는 것이 아니라 우측 그림의 점선으로 이루어진 두께가 있는 굵은 경계면을 만들고 그 두꺼운 경계면의 중앙을 지나는 선을 선택
  - 더 일반적인, 안정적인 경계선을 얻을 수 있으며 과대적합을 피할 수 있다

- (주의) SVM은 선형 모델과 마찬가지로 속성에 계수를 곱하고 덧셈을 하는 연산에 기반하므로 여러 속성을 함께 사용하려면 **반드시 스케일링**을 해야 한다.

SVM

- **SVM**
  - Outlier 에 민감
- **Support vector classifier (SVC)**
  - soft margin classifier를 사용하는 것을 support vector classifier
  - 데이터가 outliner인지 아닌지 판별 한 뒤 soft margin 을 적용하여 임계점을 나눔.
- **Soft margin**
  - soft margin은 모든 점에 대해서 cross validation으로 outlier를 판별하고, 가장 최적의 임계값을 찾아낸다.

SVM



SVM: outlier

Outlier 제외

최적의 임계값

출처: https://ekdud7667.tistory.com/50

SVM – Soft margin

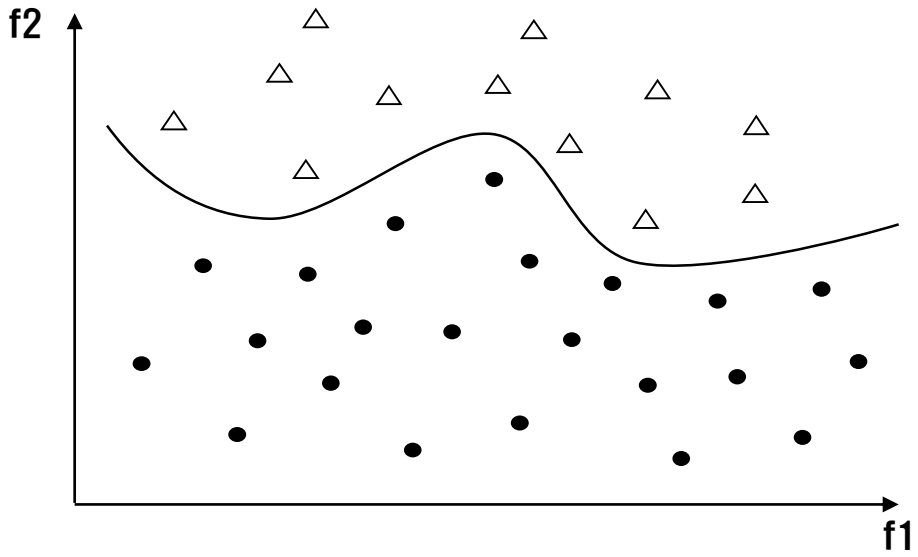
- **Soft Margin**
  - Hard margin 목적식에 regularization term이 붙는다. 즉, 에러(ξ)를 어느 정도 결정.

$$\underset{w,b,\xi}{minimize} \frac{1}{2}||w||_2^2 + C\sum \xi_i$$

  - 결정하는 파라미터는 총 3개(w,b,ξ)가 된다.
  - 여기서 C는 hyperparameter이며 training error를 얼마나 허용할 것인지를 결정. 즉, C가 크면 training error를 많이 허용하지 않고 정확한 분류를 추구하여 overfitting이 될 가능성이 크다.

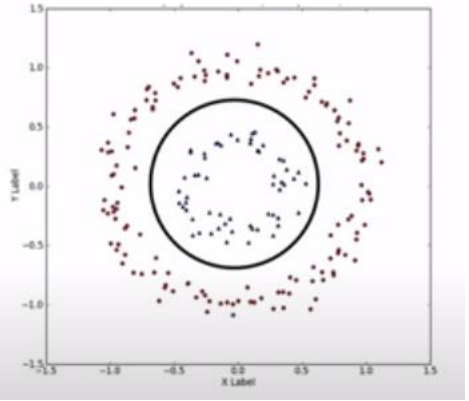
커널 방식

- SVM을 이용한 비선형 분류(kernel trick)
  - 선형 분리가 불가능한 저 차원의 특성을 그대로 사용하지 않고 이의 2승, 3승, 4승 등 고차원의 속성을 내부적으로 만들어서 사용하는 방식
  - RBF kernel이 가장 효율적인 것으로 알려져 있다
    - sklearn 모델의 기본 커널

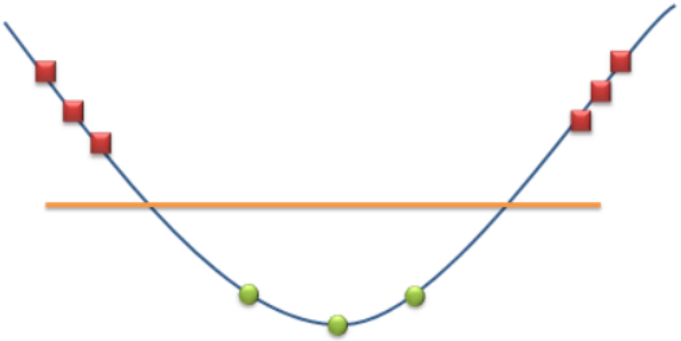


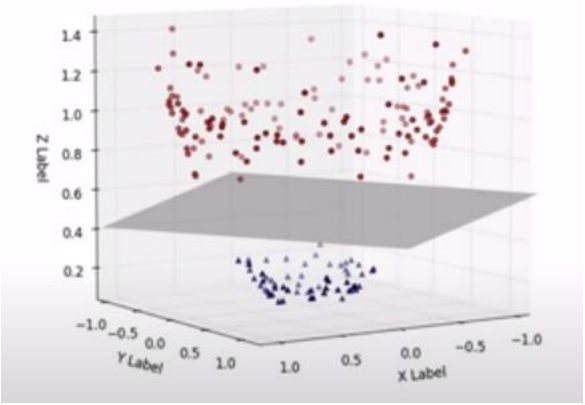
커널 방식 – 개념

- **주어진 데이터 집합**
  - **저 차원의 특성을 가진 입력 샘플 공간**



- **변형 데이터 집합**
  - **커널 트릭 사용 : 입력 샘플 공간을 고차원 특성 공간으로 변형**
  - **직선으로 분류**

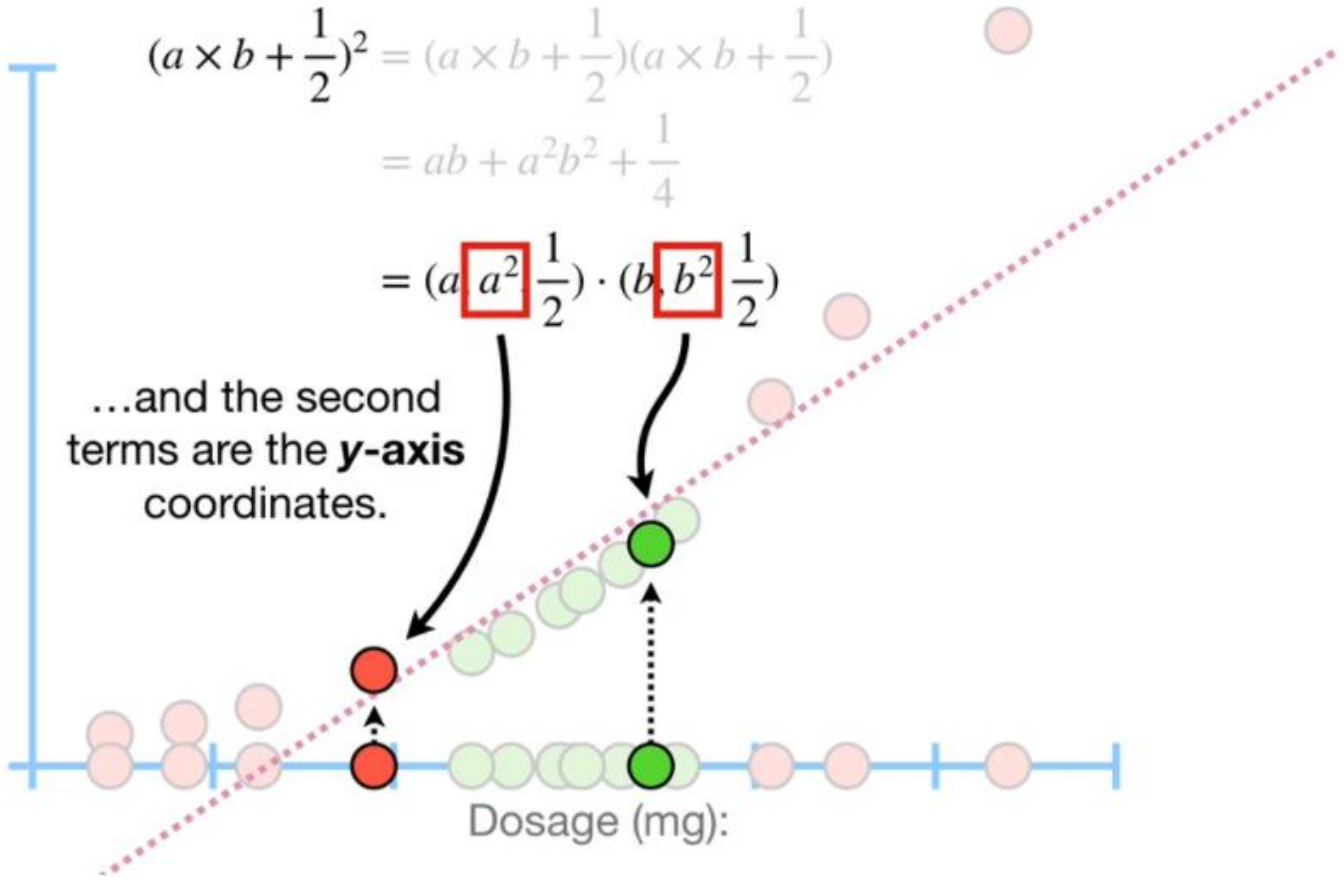




- **주요 커널**
  - Linear Kernel, Sigmoid, Polynomial Kernel, RBF(Radial Basis Function) Kernel

커널 방식 – 개념

(ex) polynomial kernel

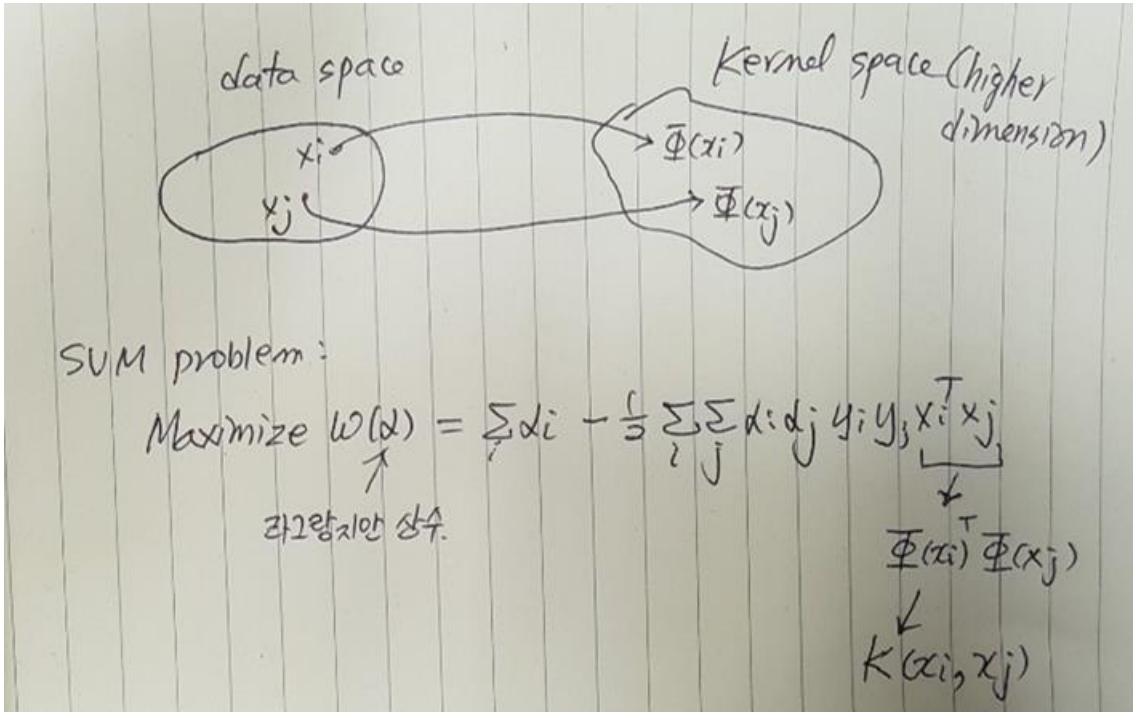


출처: https://ekdud7667.tistory.com/50

커널 방식 – kernel Trick

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$



$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

커널 방식

- Kernel function is a "similarity" function that corresponds to an inner product in some expanded space.
- **Polynomial Kernel**
  - **For two input vectors, x and y, the degree-d polynomial kernel is defined:**

$$K(x, y) = (x^{\mathsf{T}} y + c)^d$$

  - **C 와 d 는 cross validation 으로 결정.**
  - **K corresponds to an inner product in a feature space based on some mapping ϕ:**

$$K(x, y) = \langle \varphi(x), \varphi(y) \rangle$$

- **RBF kernel**
  - **For two samples, x and x' , RBF kernel (or feature vector) is defined:**

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

  - **See https://www.youtube.com/watch?v=mTyT-oHoivA**

커널 방식 – kernel Trick (예)

- Ex: Polynomial Kernel (d=2)

$$k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^d$$

$$\begin{aligned} k(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} x_1' \\ x_2' \end{pmatrix}) &= (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2' \\ &= \underbrace{\begin{pmatrix} 1 & \sqrt{2}x_1 & \sqrt{2}x_2 & x_1^2 & \sqrt{2}x_1 x_2 & x_2^2 \end{pmatrix}^T}_{\Phi(\mathbf{x})^T} \underbrace{\begin{pmatrix} 1 \\ \sqrt{2}x_1' \\ \sqrt{2}x_2' \\ x_1'^2 \\ \sqrt{2}x_1' x_2' \\ x_2'^2 \end{pmatrix}}_{\Phi(\mathbf{x}')} \end{aligned}$$

- Gaussian kernel

$$k_\sigma(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right) = \exp\left(-\frac{\mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{x}' + \mathbf{x}'^T\mathbf{x}'}{2\sigma^2}\right)$$

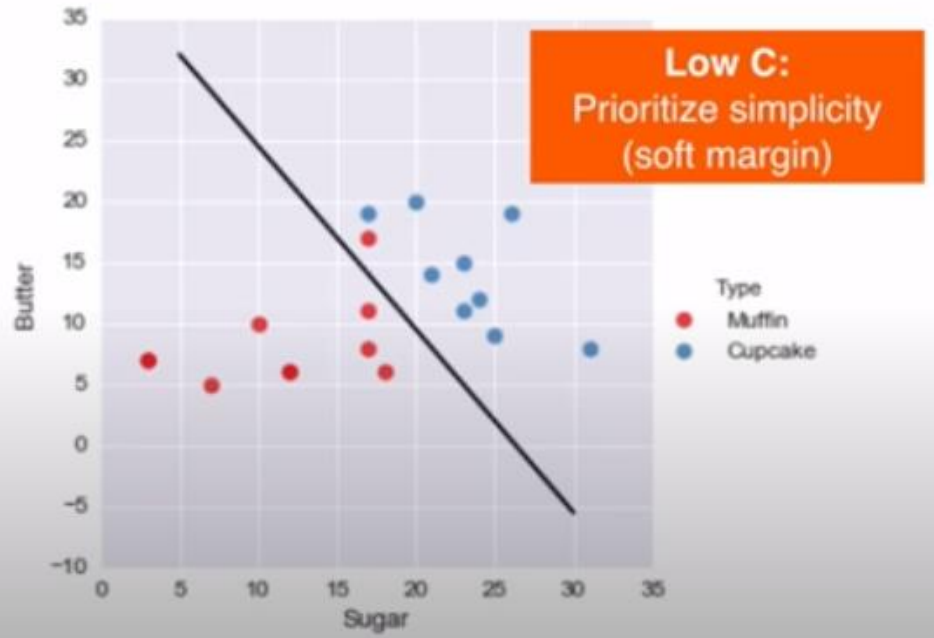
sklearn SVM Parameters

- Kernel parameter
  - Liner, polynomial, sigmoid, rbf(default)
- C
  - 훈련데이터의 올바른 분류와 의사 결정 기능의 마진 최대화를 절충
  - Larger values: 정확한 분류에 치중 (더 작은 마진 허용)
  - Lower value: 더 단순한 의사결정 기능 (훈련 정확도 희생)
  - Behaves as a regularization parameter in SVM
- Gamma
  - 단일 학습 데이터의 영향이 얼마나 멀리 도달하는지 정의, high values('close'), low value ('far')
  - 선택된 Support vector 샘플의 영향 반경의 역
  - (가까이 있는 점들의 영향력)

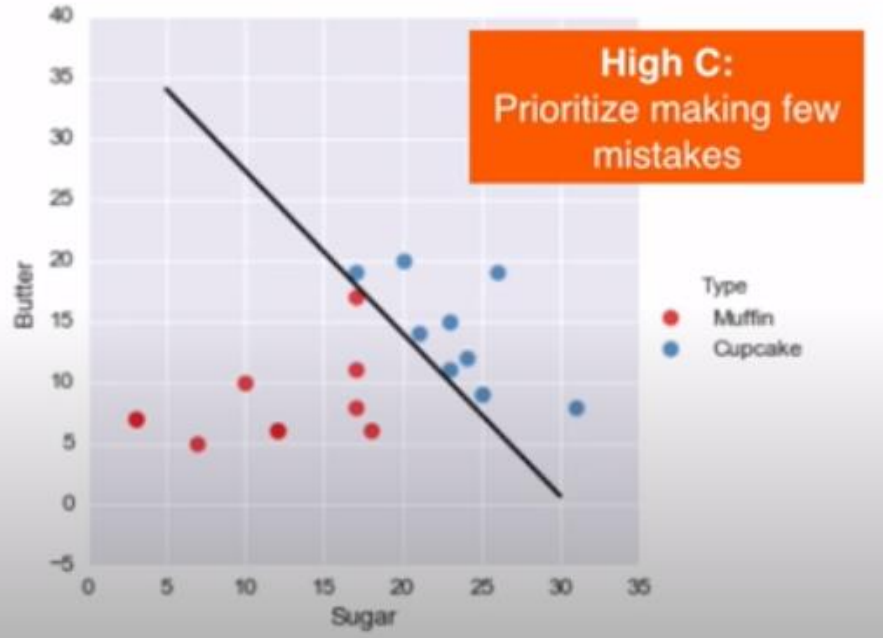
RBF SVM Parameters – C

C Parameter: Comparison

```
# Fit the SVM model with a LOW C
model = svm.SVC(kernel='linear', C=2**-5)
model.fit(sugar_butter, type_label)
```

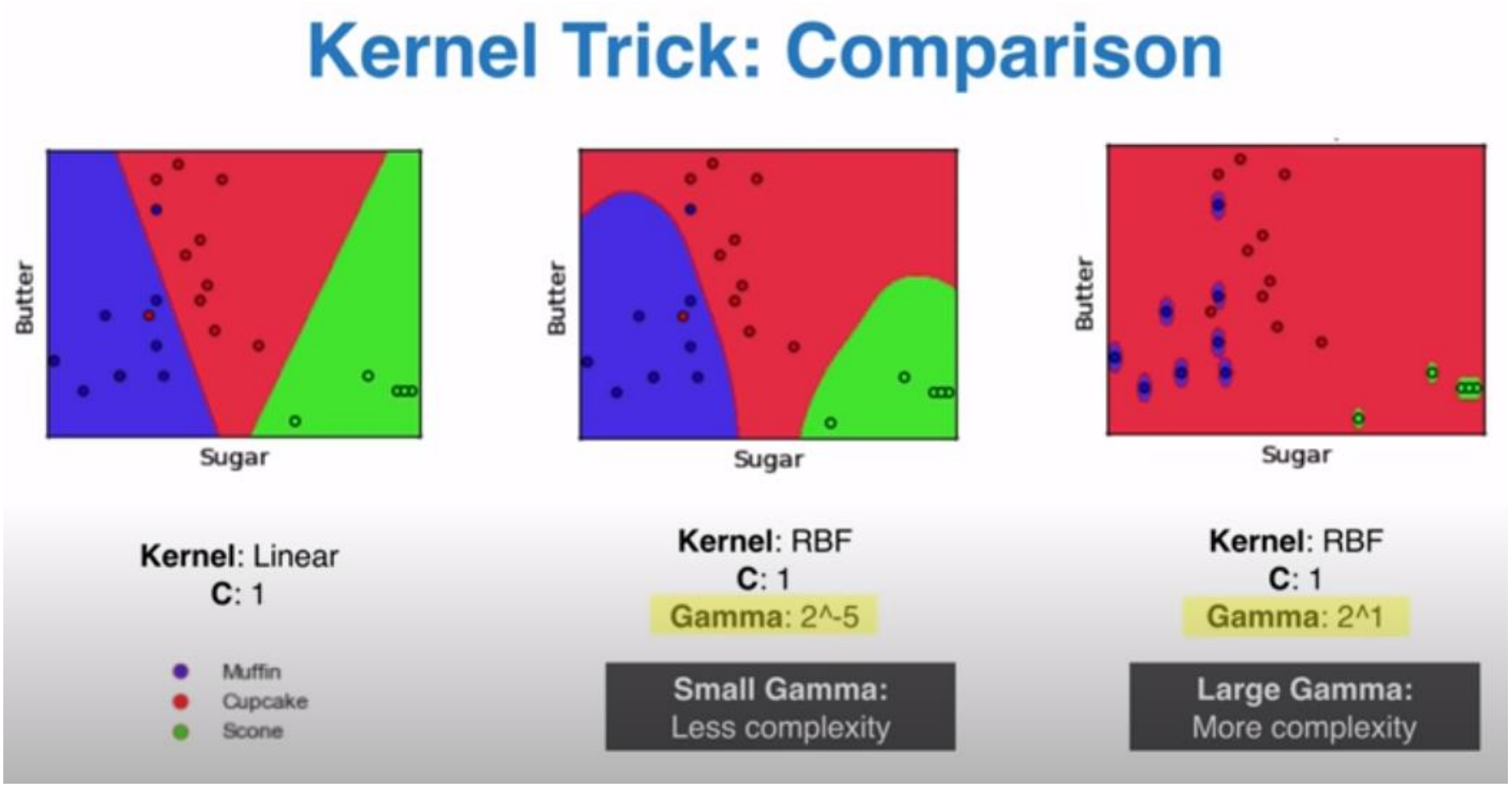



```
# Fit the SVM model with a HIGH C
model = svm.SVC(kernel='linear', C=2**5)
model.fit(sugar_butter, type_label)
```



[ref: https://www.youtube.com/watch?v=N1v0golbjSc]

sklearn SVM Parameters



[ref: https://www.youtube.com/watch?v=N1v0golbjSc]

Pros and Cons of SVM

- Pros
  - Good at dealing with high dimensional data
  - Works well on small data sets
- Cons
  - Picking the right kernel and parameters can be computationally intensive

수고하셨습니다.

Q & A