알고리즘(Algorithms)

-머신러닝 알고리즘-

동아대학교 소프트웨어대학 컴퓨터공학과

2025년 2학기

임 한 신

기말고사 일정(잠정)

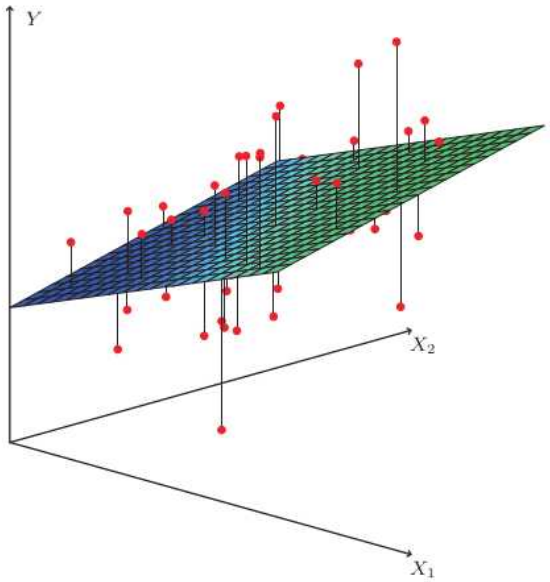
- 12월 15일 월요일 오후 3시 ~ 4시 30분

다변량 선형 회귀

- 두 개 이상의 독립변수 $x_1, x_2, \ldots, x_n$ 을 사용하여 하나의 종속변수 y를 예측하는 선형 모델
- 모델은 평면(plane) 혹은 고차원 초평면(hyperplane)

$$\hat{y} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

w_i: 회귀계수 (해당 변수의 영향력)
b: 절편(intercept)



독립 변수가 2개인 경우(x_1 ,x_2)의 다변량 선형 회귀 모델(평면) 예시

다변량 선형 회귀

- 데이터 $(x_1, x_2, \ldots, x_n, y)$에 대한 다변량 선형 회귀식

$$\mathrm{y} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

- m개의 데이터가 있다고 할 때, 다변량 선형 회귀식을 아래와 같은 행렬로 표현할 수 있음

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad \mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} & 1 \\ x_{21} & x_{22} & \cdots & x_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} & 1 \end{pmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}$$

다변량 선형 회귀

- m개의 데이터가 있다고 할 때, 다변량 선형 회귀식을 아래와 같은 행렬로 표현할 수 있음

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

- 가중치의 추정량 $\hat{\mathbf{w}}$가 아래의 오차제곱합을 최소화한다면

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \underbrace{X\mathbf{w}}_{\hat{\boldsymbol{y}}})^T(\mathbf{y} - \underbrace{X\mathbf{w}}_{\hat{\boldsymbol{y}}})$$

가중치의 추정량 $\hat{\mathbf{w}}$은 아래의 정규방정식(normal equation)으로 구할 수 있음

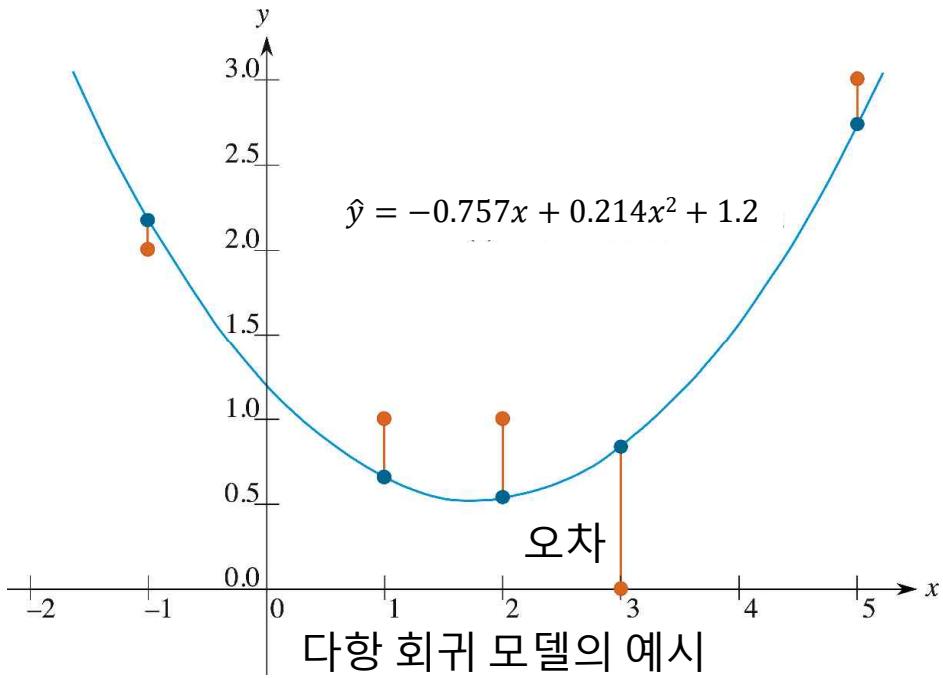
$$\hat{\mathbf{w}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

- $\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}} = \mathbf{X}^{+}$를 행렬 X의 유사 역행렬(pseudoinverse)이라고 함

다항 회귀

- 독립 변수 x 의 다항식 항$(x^2, x^3, \ldots)$을 추가하여 비선형 관계를 선형회귀 형태로 모델링하는 방법

$$\hat{y} = w_1 x + w_2 x^2 + \cdots + wnxn + b$$



다항 회귀 모델의 예시

다항 회귀

- m개의 데이터가 있다고 할 때, 다변량 선형 회귀식과 마찬가지로 아래와 같은 행렬로 표현할 수 있음

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad \mathbf{X} = \begin{pmatrix} x_1 & {x_1}^2 & \cdots & {x_1}^n & 1 \\ x_2 & {x_2}^2 & \cdots & {x_2}^n & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_m & {x_m}^2 & \cdots & {x_m}^n & 1 \end{pmatrix} \qquad w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}$$

- 마찬가지로 가중치의 추정량 $\hat{\mathbf{w}}$은 아래의 정규방정식(normal equation)으로 구할 수 있음

$$\hat{\mathbf{w}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

다항 회귀

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

$$\mathbf{y} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 3 \end{bmatrix} \qquad \mathbf{X} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 4 & 1 \\ 3 & 9 & 1 \\ 5 & 25 & 1 \end{pmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix}$$

- 위의 행렬식으로부터 아래의 정규방정식을 적용하여 가중치의 추정량 $\hat{\mathbf{w}}$을 구한다.

$$\hat{\mathbf{w}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

$$\mathbf{w} = \begin{bmatrix} -0.757 \\ 0.214 \\ 1.2 \end{bmatrix}$$

$$\hat{y} = -0.7572x + 0.214x^2 + 1.2$$

다항 회귀

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

$$\mathbf{y} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 3 \end{bmatrix} \qquad \mathbf{X} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 4 & 1 \\ 3 & 9 & 1 \\ 5 & 25 & 1 \end{pmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix}$$

- 위의 행렬식으로부터 아래의 정규방정식을 적용하여 가중치의 추정량 $\hat{\mathbf{w}}$을 구한다.

$$\hat{\mathbf{w}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

$$\mathbf{w} = \begin{bmatrix} -0.757 \\ 0.214 \\ 1.2 \end{bmatrix}$$

$$\hat{y} = -0.7572x + 0.214x^2 + 1.2$$

다변량 선형 회귀

- 아래와 같은 4개의 데이터가 있다고 하자.

(1.0, 2.0, 5.0), (3.0, 6.0, 13.0), (-3.0, -6.0, -11.0), (-2.0, -4.0, -7.0)

$$y = w_1 x_1 + w_2 x_2 + b$$

$$\mathbf{y} = \begin{bmatrix} 5.0 \\ 13.0 \\ -11.0 \\ -7.0 \end{bmatrix} \qquad \mathbf{X} = \begin{pmatrix} 1.0 & 2.0 & 1 \\ 3.0 & 6.0 & 1 \\ -3.0 & -6.0 & 1 \\ -2.0 & -4.0 & 1 \end{pmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix}$$

$$\hat{\mathbf{w}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

역행렬이 존재하지 않음

↓

추정량 $\hat{\mathbf{w}}$을 구할 수 없음

다변량 선형 회귀

- 아래와 같은 4개의 데이터가 있다고 하자.

(1.0, 2.0, 5.0), (3.0, 6.0, 13.0), (-3.0, -6.0, -11.0), (-2.0, -4.0, -7.0)

$$y = w_1 x_1 + w_2 (x_2)^2 + b$$

$$\mathbf{y} = \begin{bmatrix} 5.0 \\ 13.0 \\ -11.0 \\ -7.0 \end{bmatrix} \qquad \mathbf{X} = \begin{pmatrix} 1.0 & 4.0 & 1 \\ 3.0 & 36.0 & 1 \\ -3.0 & 36.0 & 1 \\ -2.0 & 16.0 & 1 \end{pmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix}$$

$$\hat{\mathbf{w}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

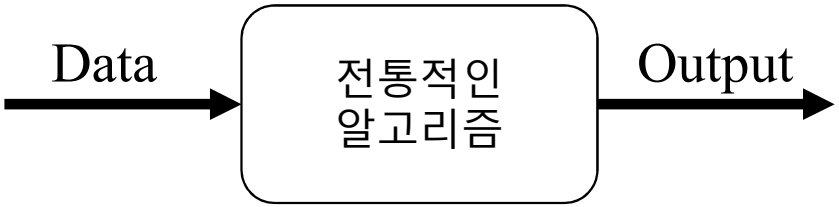
역행렬이 존재

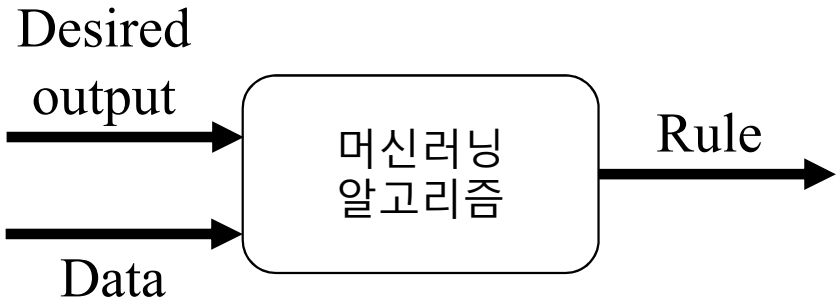
↓

추정량 $\hat{\mathbf{w}}$을 구할 수 있음

머신러닝 알고리즘

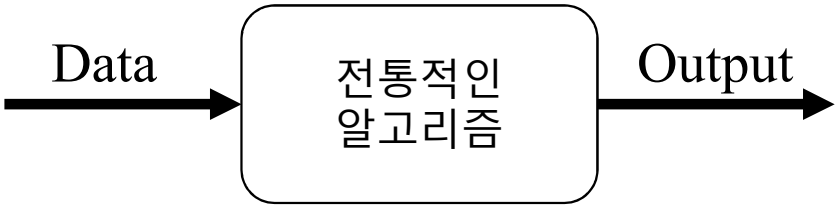
- 머신러닝 알고리즘
  - Rule을 생성하기 위한 rule/function/algorithm/code 등
  - 그러나, rule을 사람이 해석하기는 힘듦

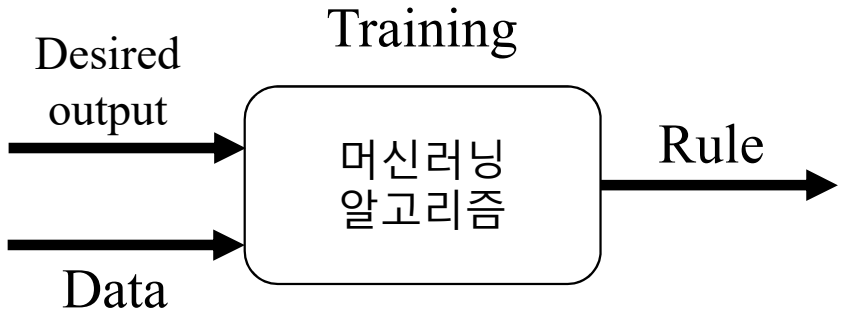


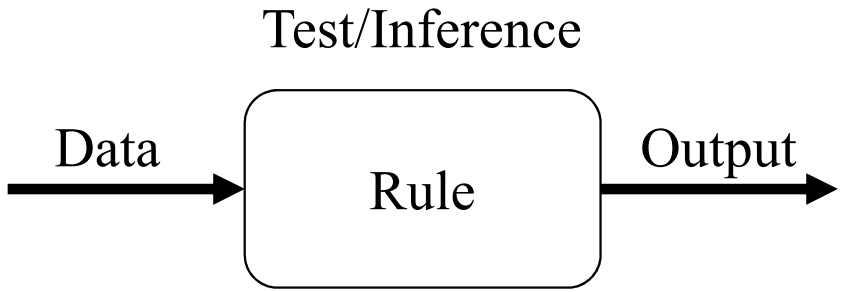


머신러닝 알고리즘

- 머신러닝 알고리즘
  - Rule을 생성하기 위한 rule/function/algorithm/code 등
  - 그러나, rule을 사람이 해석하기는 힘듦





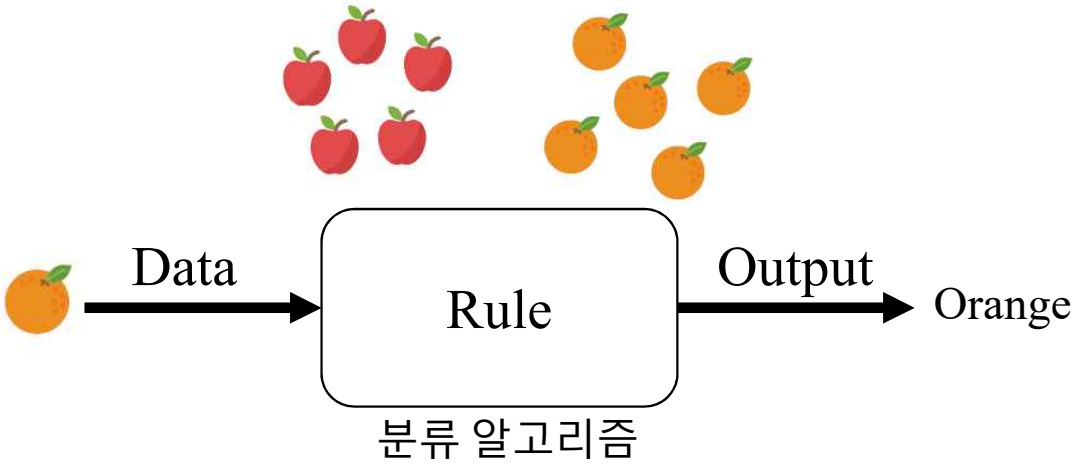


머신러닝 알고리즘

- 머신러닝 알고리즘의 종류
  - 지도 학습(Supervised learning)
    - 정답(레이블)이 있는 데이터로 학습하는 방법
    - 입력 X 와 정답 y 를 함께 주고 모델이 입력→출력의 관계를 학습하도록 함
  - 비지도 학습(Unsupervised learning)
    - 정답(레이블) 없이, 데이터의 구조나 패턴을 스스로 찾는 학습 방법
    - 레이블이 없이 X만 주어진 상태에서 데이터 내의 군집, 분포, 패턴, 이상치 등을 찾음
  - 강화 학습(Reinforcement learning)
    - 환경 안에서 행동을 선택하면 보상(reward)을 받고, 최대 보상을 얻는 행동 전략을 학습하는 방법
    - 모델은 시행착오(Trial & Error)를 통해 학습함

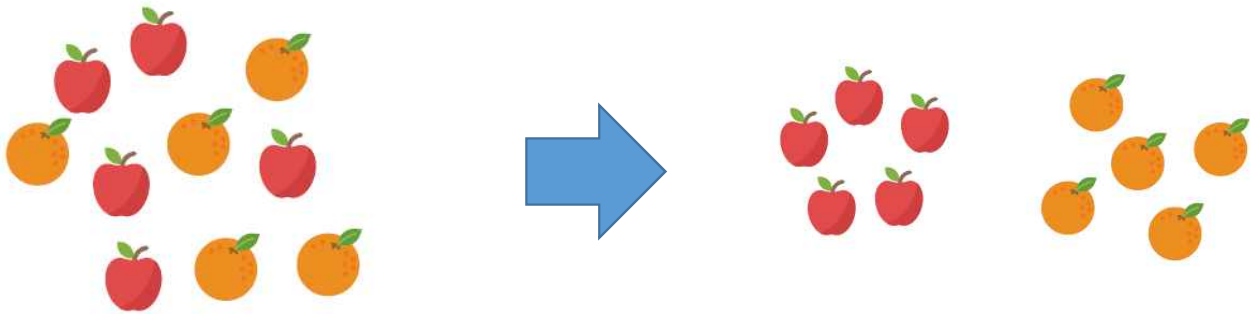
머신러닝 알고리즘

- 머신러닝 알고리즘의 종류
  - 지도 학습(Supervised learning)
    - 정답(레이블)이 있는 데이터로 학습하는 방법
    - 입력 X 와 정답 y 를 함께 주고 모델이 입력→출력의 관계를 학습하도록 함
    - 대표 알고리즘
      - 분류 알고리즘(결정 트리, 랜덤 포레스트, 그래디언트 부스팅, SVM 등)
      - 회귀 알고리즘(선형 회귀, 비선형 회귀 등)



머신러닝 알고리즘

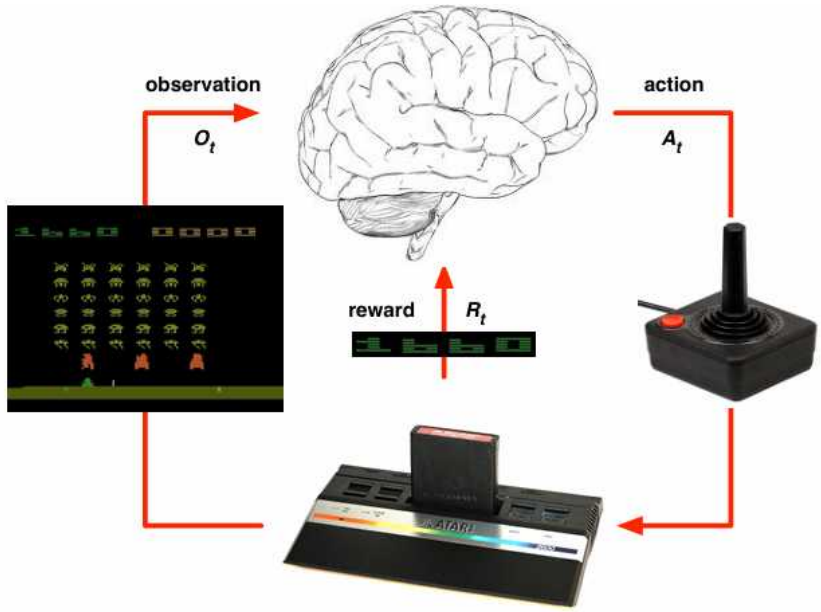
- 머신러닝 알고리즘의 종류
  - 비지도 학습(Unsupervised learning)
    - 정답(레이블) 없이, 데이터의 구조나 패턴을 스스로 찾는 학습 방법
    - 레이블이 없이 X만 주어진 상태에서 데이터 내의 군집, 분포, 패턴, 이상치 등을 찾음
    - 대표 알고리즘
      - 클러스터링 알고리즘(K-Means, EM 알고리즘, DBSCAN 등)
      - 차원 축소 알고리즘(PCA 등)



클러스터링 알고리즘

머신러닝 알고리즘

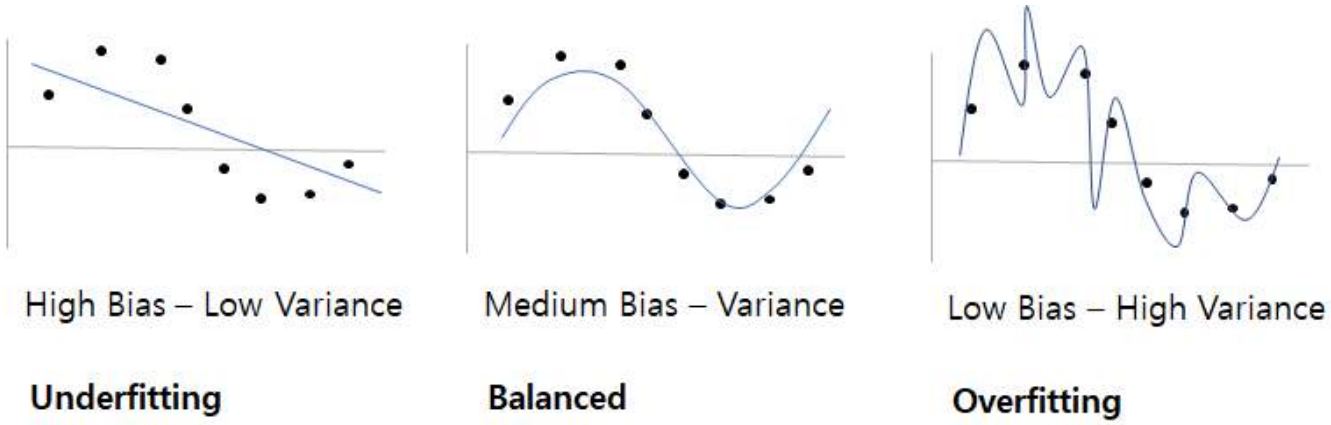
- 머신러닝 알고리즘의 종류
  - 강화 학습(Reinforcement learning)
    - 환경 안에서 행동을 선택하면 보상(reward)을 받고, 최대 보상을 얻는 행동 전략을 학습하는 방법
    - 모델은 시행착오(Trial & Error)를 통해 학습함
    - 대표 알고리즘
      - Q-learning
      - Deep Q-Network
      - …



아타리 게임 적용 예시

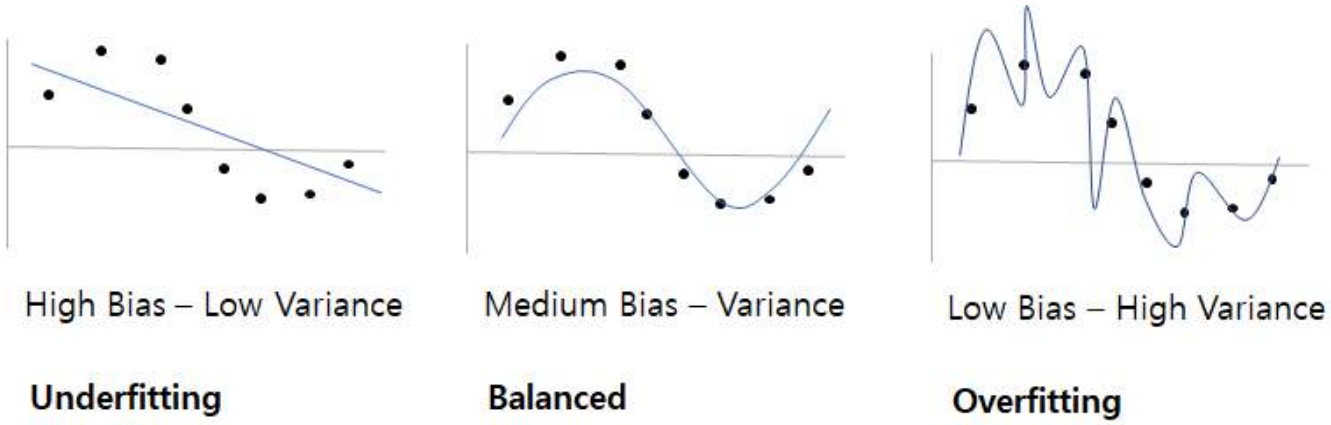
편향(Bias)과 분산(Variance)

- 편향(Bias)
  - 학습 데이터셋으로 학습시킨 모델의 출력의 평균값과 실제 모델 출력 사이의 편차
  - 즉, 모델이 평균적으로 실제 함수에서 얼마나 벗어나는지를 나타냄
  - 모델이 너무 단순해서 진짜 패턴을 제대로 따라가지 못할 때 발생
  - Underfitting 때 크게 발생



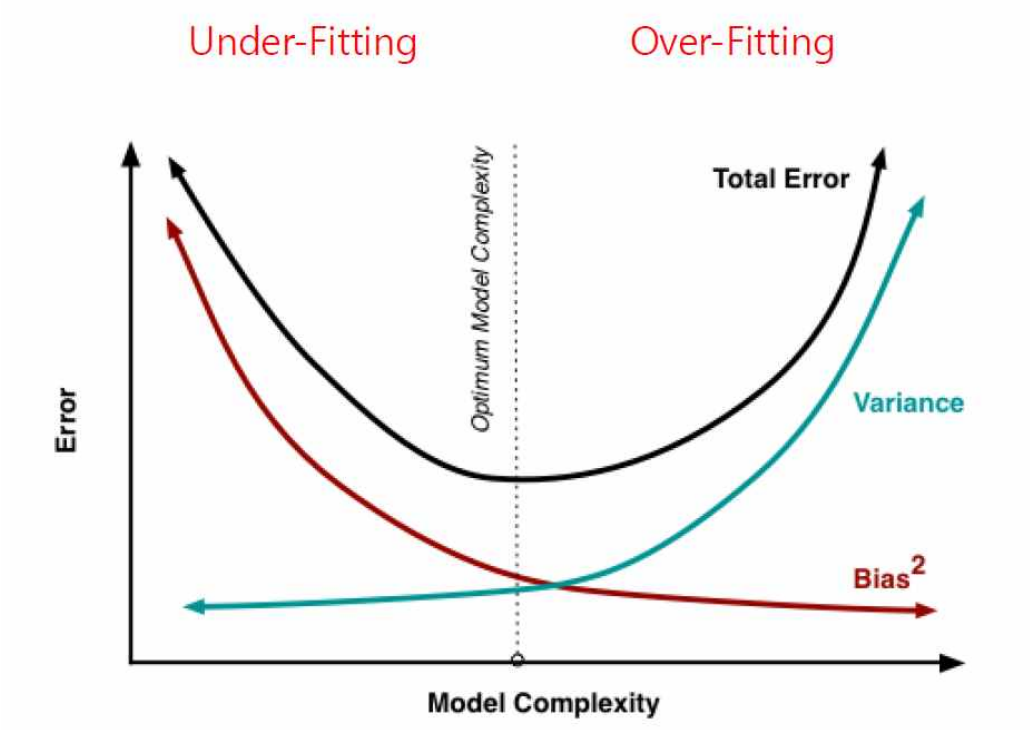
편향(Bias)과 분산(Variance)

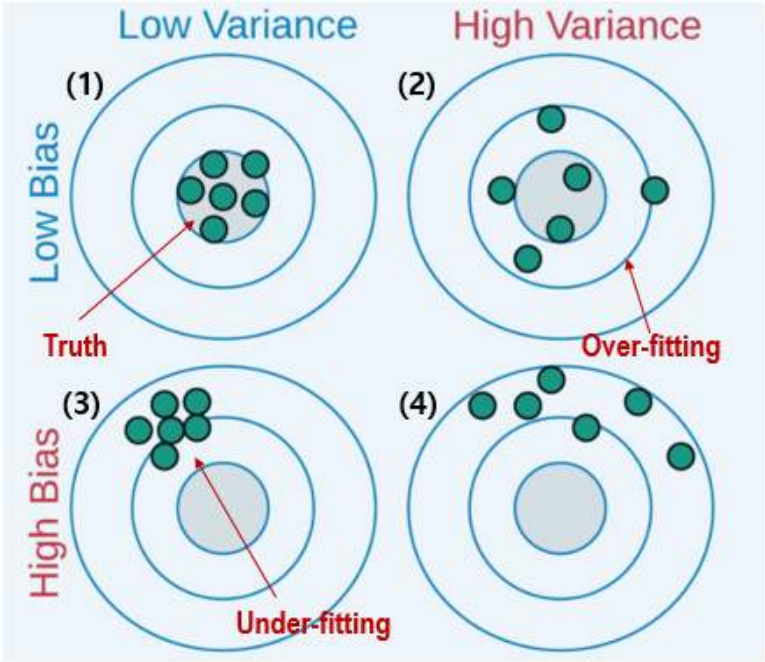
- 분산(Variance)
  - 학습 데이터를 바꿔 학습시키면 예측 결과가 얼마나 흔들리는지를 나타내는 값
  - 분산이 큰 모델은
    - 훈련 데이터에 너무 민감해져 작은 변화에도 예측이 크게 요동치게 됨
    - 모델이 너무 복잡해서 훈련 데이터의 노이즈까지 외워버리는 상태가 됨 (Overfitting)



편향(Bias)과 분산(Variance)

- 편향-분산 tradeoff





엔트로피(Entropy)

- 정보량(Information content)
  - 사건이 얼마나 예측하기 어려운가를 나타내는 수치
  - 어떤 사건 x가 일어났을 확률을 $p(x)$라고 하면 사건 x가 발생했을 때 얻는 정보량 $I(x)$는 아래와 같은 특성을 갖는다.
    - 확률이 낮을수록 정보량이 크고 확률이 높을수록 정보량은 작다.
    - 독립 사건의 정보량은 더해져야 한다.

      예) 두 독립 사건 A, B가 있을 때 A가 발생한 정보가 1bit이고 B가 발생한 정보가 1bit이면 A와 B가 동시에 발생한 정보량은 2bits가 되어야 함

  - 따라서, 어떤 사건 x가 일어났을 확률을 $p(x)$라고 하면 사건 x가 발생했을 때 얻는 정보량 $I(x)$는 아래와 같이 표현됨

$$I(x) = \log_2 \left(\frac{1}{p(x)}\right) = -\log_2 p(x)$$

※ logAB = logA + logB

엔트로피(Entropy)

- 정보량(Information content)
  - 예) 동전을 던졌을 때 앞면과 뒷면이 나올 확률이 각각 0.5, 0.5이다. 동전을 2회 던졌을 때 모두 앞면이 나올 경우의 정보량은?
    - 동전을 2회 던지는 사건은 독립사건이므로 $p(\text{앞면, 앞면}) = p(\text{앞면})p(\text{앞면}) = 0.25$

1) $I(\text{앞면, 앞면}) = -\log_2(p(\text{앞면, 앞면})) = -\log_2(0.25) = 2 \text{ (bits)}$

2) $I(\text{앞면, 앞면}) = -\log_2(p(\text{앞면})p(\text{앞면})) = -\log_2(p(\text{앞면})) - \log_2(p(\text{앞면}))$

$= -\log_2(0.5) - \log_2(0.5) = 1 + 1 = 2 \text{ (bits)}$

엔트로피(Entropy)

- 확률변수 X가 여러 사건 x_i를 가질 때, 엔트로피 $H(X)$는 아래와 같이 계산됨

$$H(X) = -\sum_i p(x_i) \log_2 p(x_i)$$

- 즉, 엔트로피는 정보량의 기대값(평균)임
- 해당 확률 분포가 만들어내는 평균적인 놀라움을 나타냄
- 예측이 어려운 데이터일수록 엔트로피가 크고 예측이 쉬운 데이터일수록 엔트로피가 작음
- 예) 모든 사건의 확률이 균등할 때 가장 예측이 어렵고 이때 엔트로피가 최대

엔트로피(Entropy)

- 확률변수 X가 여러 사건 x_i를 가질 때, 엔트로피 $H(X)$는 아래와 같이 계산됨

$$H(X) = -\sum_i p(x_i) \log_2 p(x_i)$$

- 엔트로피 계산의 예
  - 동전의 앞, 뒷면이 나올 확률이 0.5, 0.5인 경우

    $H = -[0.5\log_2(0.5) + 0.5\log_2(0.5)] = -[-0.5 - 0.5] = 1$ (bit)

  - 동전의 앞, 뒷면이 나올 확률이 0.7, 0.3인 경우

    $H = -[0.7\log_2(0.7) + 0.3\log_2(0.3)] =$ 약 0.881 (bit)

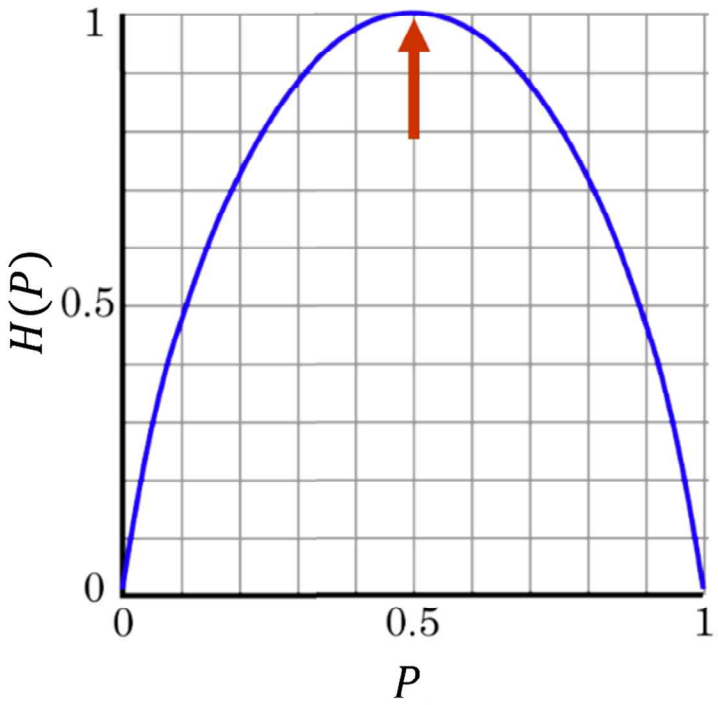
  - 동전의 앞, 뒷면이 나올 확률이 1.0, 0.0인 경우

    $H = -[1.0\log_2(1.0) + 0.0\log_2(0.0)] = -[\log_2(1.0)] = 0$

엔트로피(Entropy)

- 확률변수 X가 여러 사건 x_i를 가질 때, 엔트로피 $H(X)$는 아래와 같이 계산됨

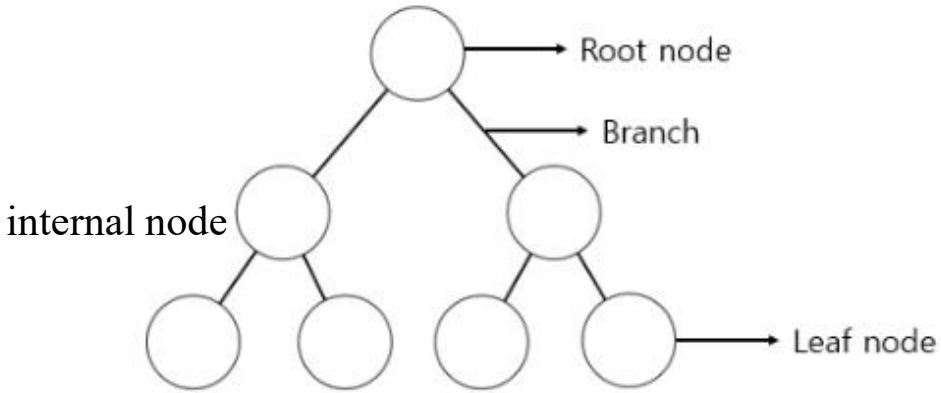
$$H(X) = -\sum p(x_i) \log_2 p(x_i)$$

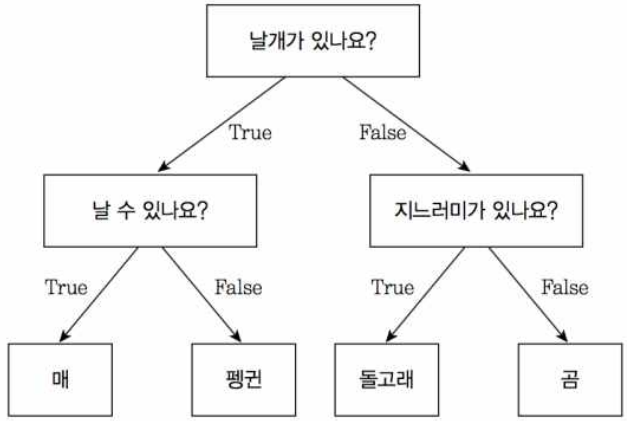


엔트로피 그래프

결정 트리(Decision Tree)

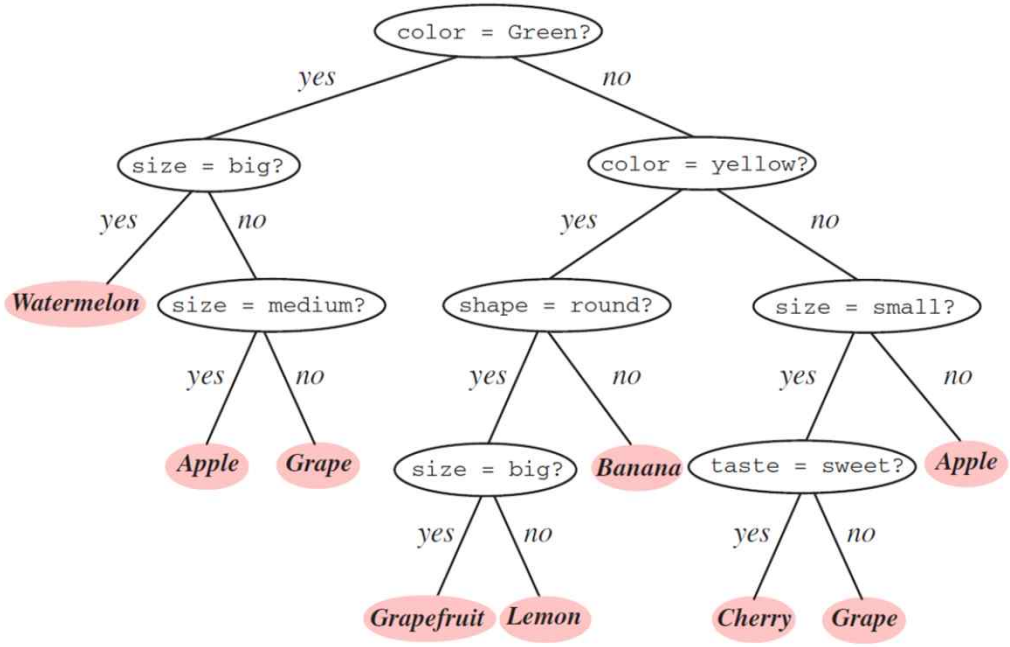
- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사
- 결정 트리는 다음과 같은 구조를 가짐
  - 루트 노드(root): 처음 데이터를 받는 위치
  - 내부 노드(internal node): 조건 기준으로 데이터를 분류할 수 있는 특성, 자식 노드를 가짐
  - 리프 노드(leaf node): 최종 예측 결과(클래스 레이블 또는 숫자)





결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

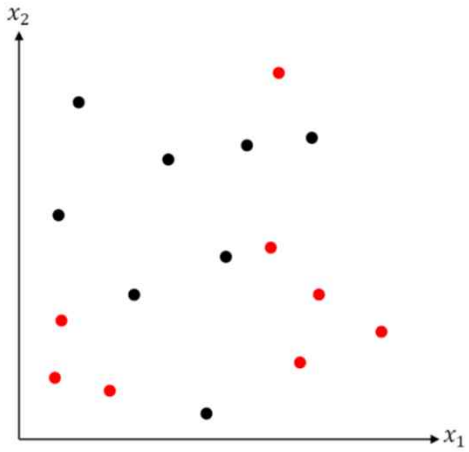


이진 결정 트리

결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

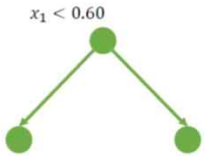
결정 트리 1

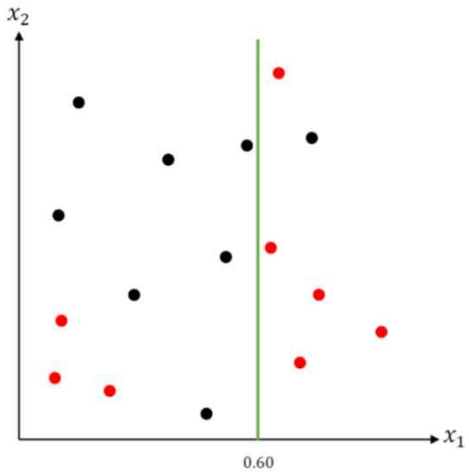


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 1

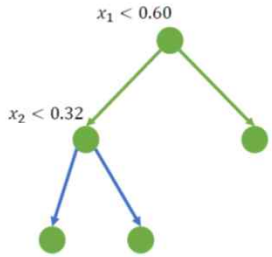


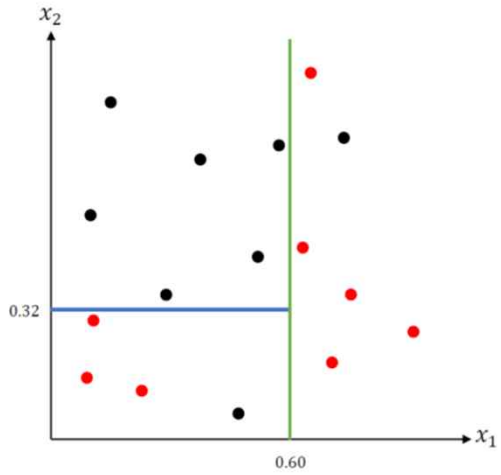


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 1

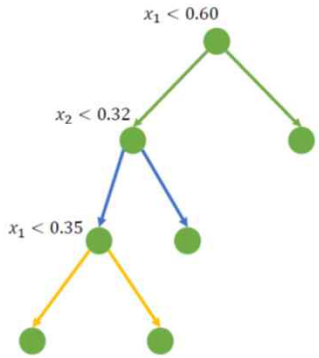


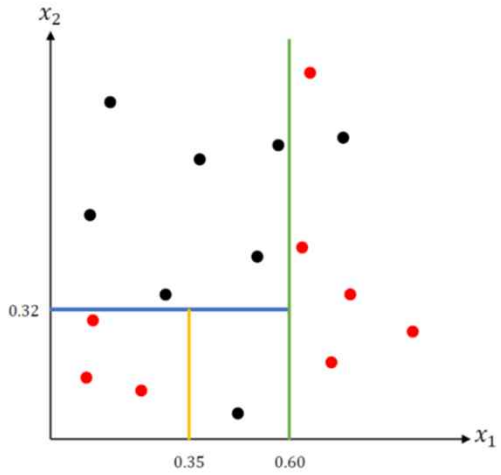


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 1

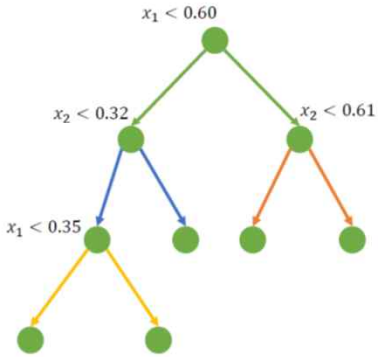


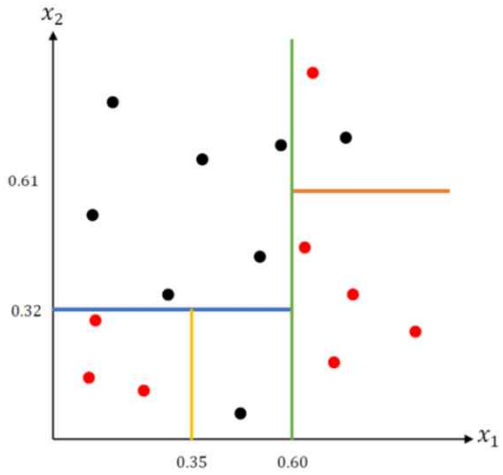


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 1

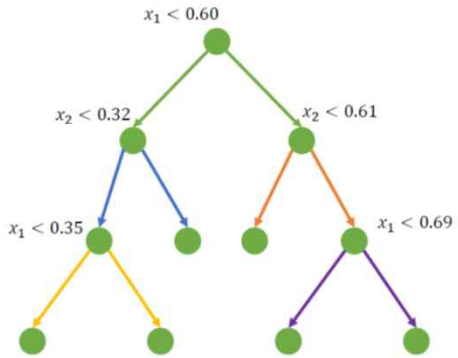


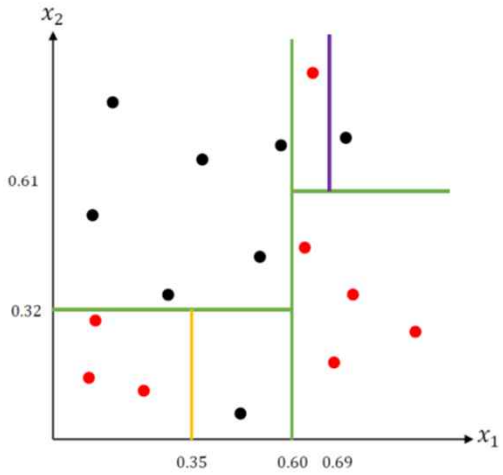


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 1

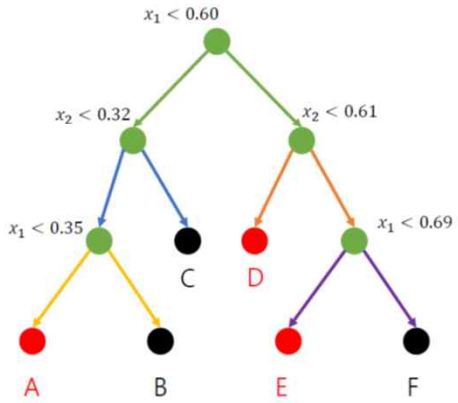


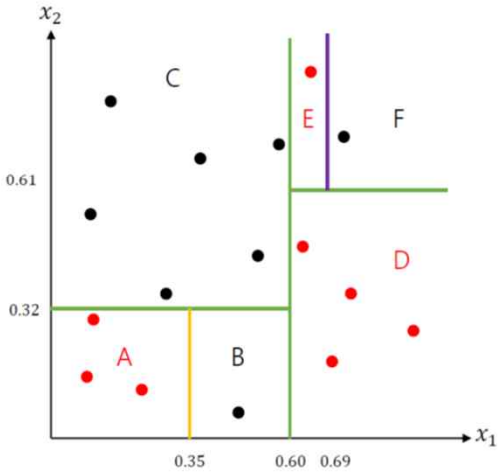


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 1

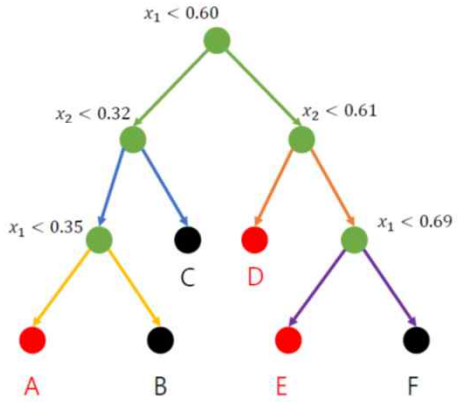


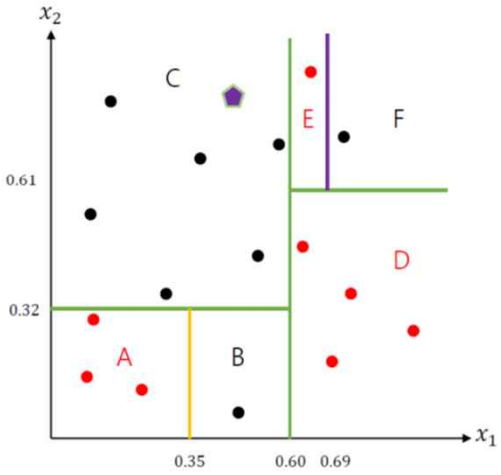


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 1

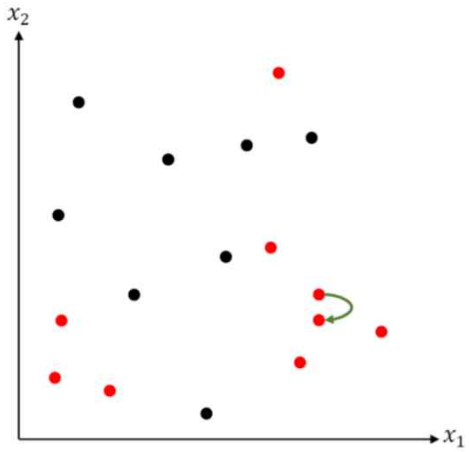




결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

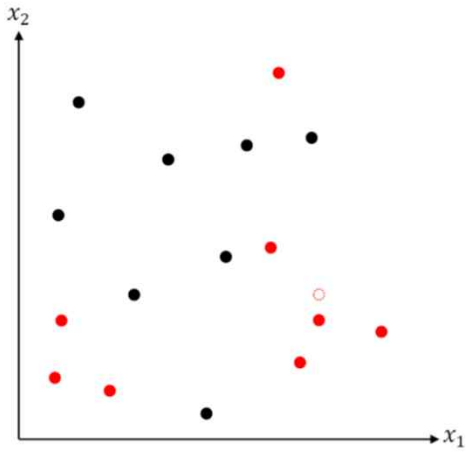
데이터 위치 변화



결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

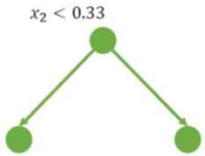
결정 트리 2

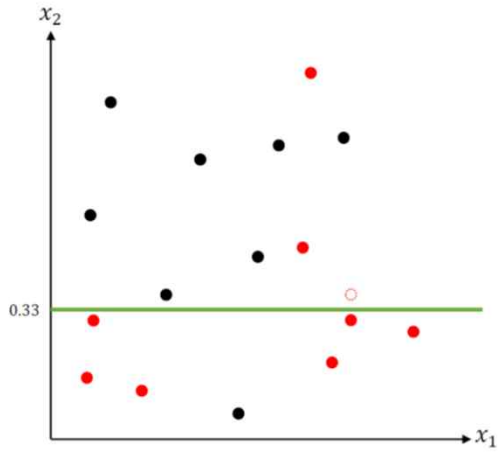


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 2

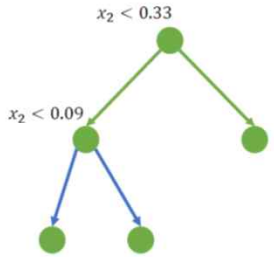


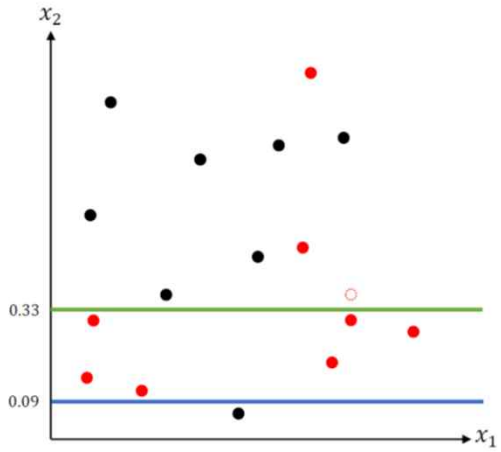


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 2

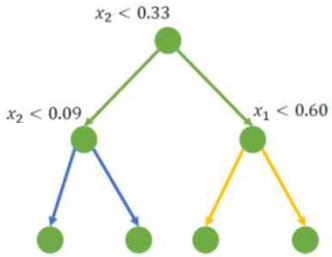


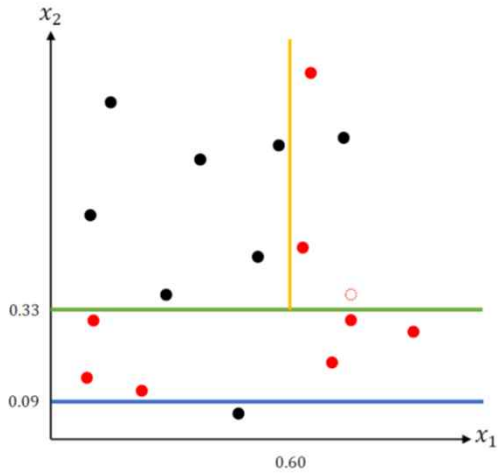


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 2

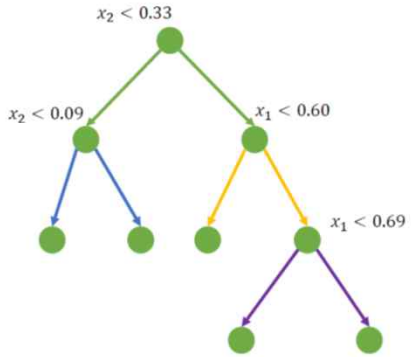


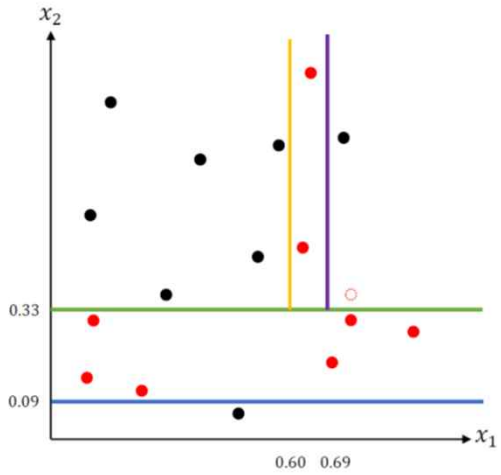


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 2

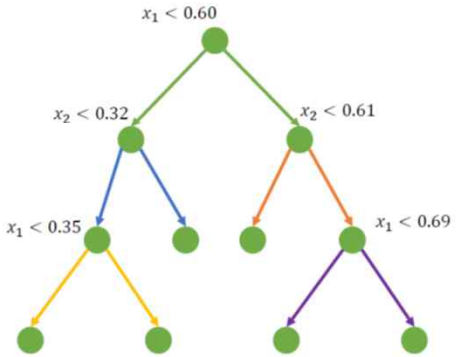


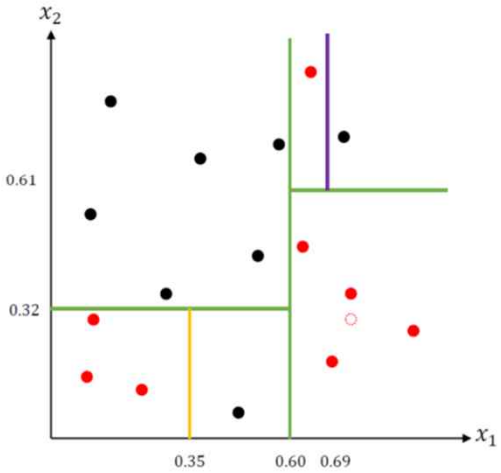


결정 트리(Decision Tree)

- 데이터를 여러 개의 질문(조건)으로 단계적으로 분리하여 최종적으로 분류(Classification) 또는 회귀(Regression)를 수행하는 모델
- 즉, 사람이 "질문-답변"을 하며 결론에 도달하는 방식과 매우 유사

결정 트리 1





결정 트리(Decision Tree)

- 결정 트리의 학습
  - 결정트리는 "어떤 질문으로 데이터를 나누면 좋을까?"를 판단하기 위해 노드의 불순도(impurity) 를 계산
  - 가장 불순도 감소가 큰 분할을 선택
  - 불순도 감소가 큰 분할의 의미는 섞여 있던 클래스가 최소화되도록 나누는 것

    예) 원래 노드가 [고양이 50, 개 50]이었다면 → 매우 불순함 (50:50)

    불순도 감소가 큰 분할의 의미는
    - 왼쪽: [고양이 48, 개 2]
    - 오른쪽: [고양이 2, 개 48]

    처럼 나누는 분할

결정 트리(Decision Tree)

- 불순도 계산
  - 대표적인 불순도 계산 방법은 엔트로피 기반 방법과 지니 불순도(Gini impurity)가 있음
  - 엔트로피 기반 방법
    - 정보 이득(Information Gain)
      - 분할 전 → 데이터가 섞여 있을수록 엔트로피 높음
      - 분할 후 → 노드들이 더 순수해지면 엔트로피 감소
      - 정보 이득 = 분할하기 전의 엔트로피 – 분할한 후의 엔트로피(가중평균)
    - 정보 이득이 가장 큰 분할을 선택 → 엔트로피를 가장 줄이는 분할을 선택 → 불확실성을 가장 줄이는 분할을 선택

결정 트리(Decision Tree)

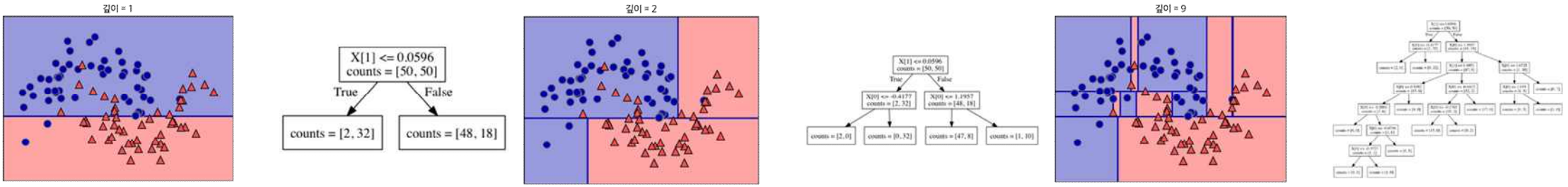
- 불순도 계산
  - 지니 불순도(Gini Impurity)
    - 클래스가 k개 있는 노드에서 각 클래스의 비율을 $p_1, p_2, \ldots, p_k$ 라고 하면

$$\text{Gini} = 1 - \sum_{i=1}^{k} p_i^2$$

    - 즉, 한 노드에서 임의로 두 샘플을 뽑았을 때 두 샘플이 서로 다른 클래스가 될 확률을 의미
    - 엔트로피 기반 계산에 비해 계산이 더 단순
    - 노드 안이 많이 섞여 있을수록(불순할수록) → 같은 클래스를 뽑을 확률이 줄어듦 → Gini가 커짐
    - 반대로 한 클래스만 있다면 → 무조건 같은 클래스 → Gini = 0
    - 후보 분할 전후 지니 불순도 변화량을 계산하여 불순도를 가장 많이 줄여주는 분할 선택

결정 트리(Decision Tree)

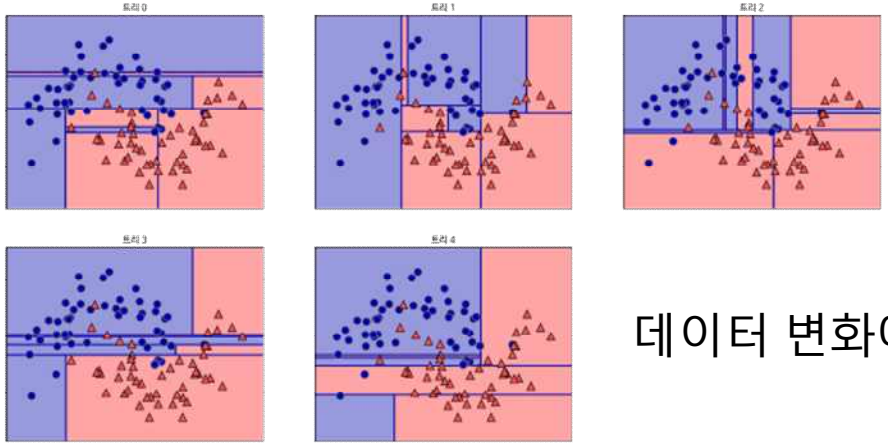
- 결정 트리의 단점
  - 과적합(overfitting)에 매우 취약
    - 노드를 계속 분할하면 데이터가 점점 작아짐 → 더 순수한 노드 생성
    - 실제로는 의미 없는 패턴(우연한 잡음)을 학습함으로써 일반화 성능 저하
  - 데이터 변화에 매우 민감함
    - 데이터가 조금만 바뀌어도 트리 전체 구조가 크게 달라짐(high variance)
    - 분기점이 하나 달라지면 그 아래 구조가 크게 변화 → 구조가 불안정함



과적합(overfitting)의 예시

결정 트리(Decision Tree)

- 결정 트리의 단점
  - 과적합(overfitting)에 매우 취약
    - 노드를 계속 분할하면 데이터가 점점 작아짐 → 더 순수한 노드 생성
    - 실제로는 의미 없는 패턴(우연한 잡음)을 학습함으로써 일반화 성능 저하
  - 데이터 변화에 매우 민감함
    - 데이터가 조금만 바뀌어도 트리 전체 구조가 크게 달라짐(high variance)
    - 분기점이 하나 달라지면 그 아래 구조가 크게 변화 → 구조가 불안정함



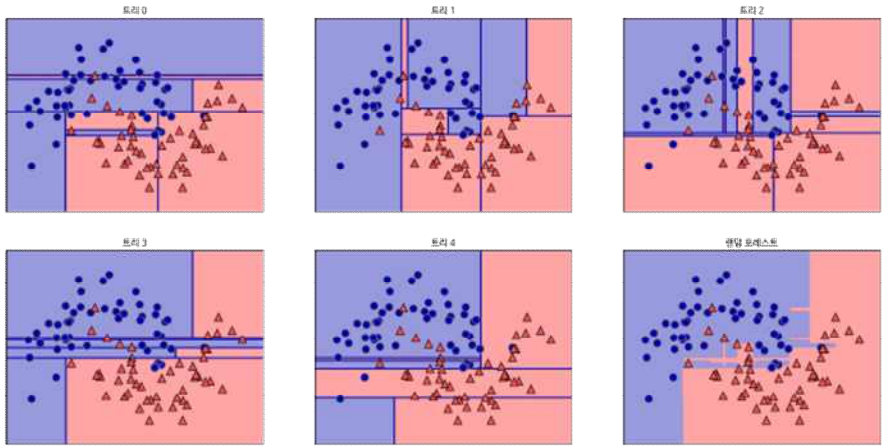
데이터 변화에 민감성 예시

결정 트리(Decision Tree)

- 결정 트리의 단점
  - 과적합(overfitting)에 매우 취약
    - 노드를 계속 분할하면 데이터가 점점 작아짐 → 더 순수한 노드 생성
    - 실제로는 의미 없는 패턴(우연한 잡음)을 학습함으로써 일반화 성능 저하
    - 해결 방법
      - 가지치기(Pruning)
        - 트리가 일정한 깊이에 도달하면 성장을 멈춤
        - 노드의 샘플 수가 어떤 임계값보다 작아지면 성장을 멈춤
        - 분할이 정확도 향상에 미치는 영향을 계산하여 일정한 임계값보다 작아지면 성장을 멈춤

결정 트리(Decision Tree)

- 결정 트리의 단점
  - 데이터 변화에 매우 민감함
    - 데이터가 조금만 바뀌어도 트리 전체 구조가 크게 달라짐(high variance)
    - 분기점이 하나 달라지면 그 아래 구조가 크게 변화 → 구조가 불안정함
    - 해결 방법
      - 여러 트리를 만들고 평균 → 랜덤 포레스트(Random forest)



랜덤 포레스트

앙상블 기법

- 앙상블 기법의 개념
  - 여러 모델의 예측을 결합(average, voting, boosting)하여 더 좋은 예측을 만드는 방법
  - 약한 모델(weak learner)도 다수 결합하면 강한 모델(strong learner)이 될 수 있음
  - 대표적인 기법으로 배깅(Bagging), 부스팅(Boosting)이 있음
  - 앙상블 기법의 일반적인 과정
    1. 오차가 상호 독립적인 기초 분류기를 찾음
    2. 기초 분류기를 훈련
    3. 기초 분류기 결과를 병합

앙상블 기법

- 배깅(Bagging(Bootstrap AGGregatING))
  - 여러 모델을 독립적으로 병렬 학습하고 평균 또는 다수결로 최종 결정
  - 데이터 부트스트랩 샘플링 사용
  - 대표 알고리즘
    - 랜덤 포레스트(Random forest)
  - 장점
    - 분산(variance) 감소
    - 안정적인 모델
    - Overfitting에 강함
  - 단점
    - 편향(bias)은 크게 줄이지 못함
    - 대규모 결정 트리가 필요할 수 있음

앙상블 기법

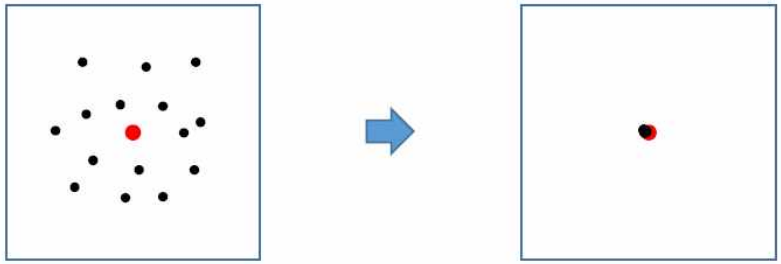
- 부스팅(Boosting)
  - 약한 모델을 순차적으로 학습시키면서 이전 모델의 오류를 점점 보완해 나가는 방식
  - 매번 학습을 수행할 때마다 이미 생성된 약한 분류기 집합(현재 모델)의 예측 결과에 기반해 새로운 분류기에서 잘못 분류한 샘플에 초점을 맞춰 학습
  - 대표 알고리즘
    - 에이다부스트(AdaBoost)
    - 그래디언트 부스팅(Gradient boosting)
  - 장점
    - 편향(bias)과 분산(variance) 둘 다 낮춤
    - 비교적 높은 성능
  - 단점
    - 배깅(bagging)보다 overfitting 위험
    - 비교적 학습 시간 길고 튜닝 필요

랜덤 포레스트(Random Forest)

- 랜덤 포레스트의 개념
  - 상관도가 낮은 결정트리 여러 개를 만들고 그 예측을 평균(회귀) 또는 다수결(분류) 로 결합(앙상블)
  - 서로 다른 모델 $h_1, h_2, \ldots, h_n$이 있고 각 모델의 분산이 σ_2 이라고 하고 각 모델이 서로 독립이라면 앙상블의 분산은 아래와 같음

$$\mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n} h_i\right) = \frac{\sigma^2}{n}$$

  - 예) (서로 독립인) 모델을 100개 만들고 평균하면 → 분산이 1/100로 줄어듦



랜덤 포레스트(Random Forest)

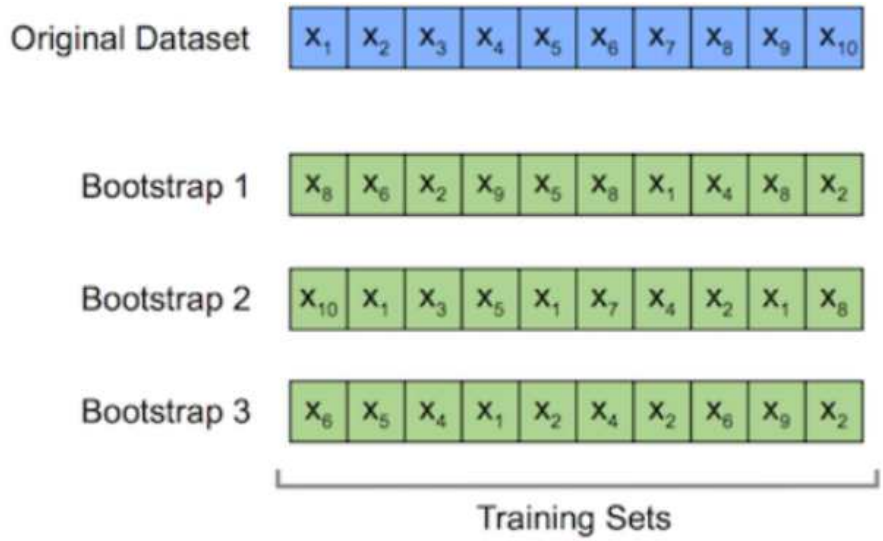
- 랜덤 포레스트의 개념
  - 상관도가 낮은 결정트리 여러 개를 만드는 방법
    - 배깅(Bagging)
      - 부트스트랩 샘플링(Bootstrap sampling) 기법으로 각 트리를 학습
    - 특징 서브샘플링(Feature Subsampling)
      - 각 노드에서 분할할 때 전체 feature가 아니라, 무작위로 선택된 일부 feature들만 사용

랜덤 포레스트(Random Forest)

- 배깅(Bagging(Bootstrap AGGregatING))
  - 원래의 학습 데이터셋에서 부트스트랩(bootstrap) 샘플을 통해 학습하는 기법
  - 원래의 데이터셋에서 학습 데이터셋을 여러 개 만들어서 결정 트리들의 학습에 사용
  - 부트스트랩(bootstrap) 샘플
    - 중복을 허용한 랜덤 샘플 방법을 의미
    - 과정
      1. m개의 샘플을 가지는 데이터셋에서 랜덤으로 하나의 샘플을 추출하고 다시 돌려놓음
      2. 같은 프로세스를 m번 반복
      3. m개의 샘플이 있는 데이터셋을 얻음
    - 원본 데이터와 같은 크기지만, 일부 데이터가 중복으로 들어가고 일부는 빠질 수 있는 데이터셋 생성

랜덤 포레스트(Random Forest)

- 배깅(Bagging(Bootstrap AGGregatING))
  - 원래의 학습 데이터셋에서 부트스트랩(bootstrap) 샘플을 통해 학습하는 기법
  - 원래의 데이터셋에서 학습 데이터셋을 여러 개 만들어서 결정 트리들의 학습에 사용



부트스트랩 샘플의 예시

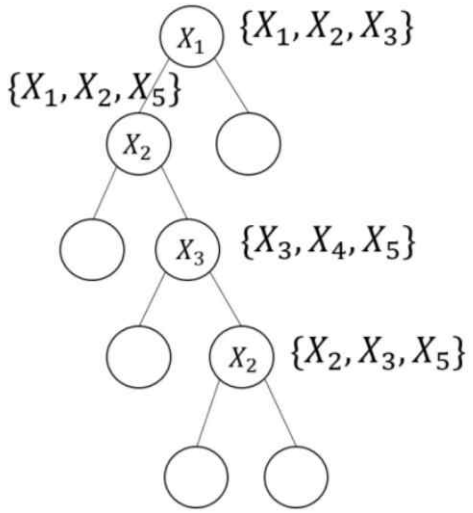
랜덤 포레스트(Random Forest)

- 특징 서브샘플링(Feature Subsampling)
  - 각 노드를 만들 때마다 전체 feature 중 일부만 무작위로 선택하고 그 중에서 최적 분할을 선택
  - 즉, 모든 노드에서 계속 feature를 랜덤하게 샘플링
  - 예) 어떤 데이터셋의 데이터의 feature가 100개라고 하면 각 노드마다 10개의 랜덤한 feature를 선택하고 이들 중 가장 좋은 분할을 최종 선택
  - 결정트리들 간의 상관도를 감수 시킬 수 있음
    - 만약 특징 서브샘플링이 없으면 결정트리의 루트노드 및 상위 노드들 대부분은 비슷한 feature를 선택할 가능성이 많음

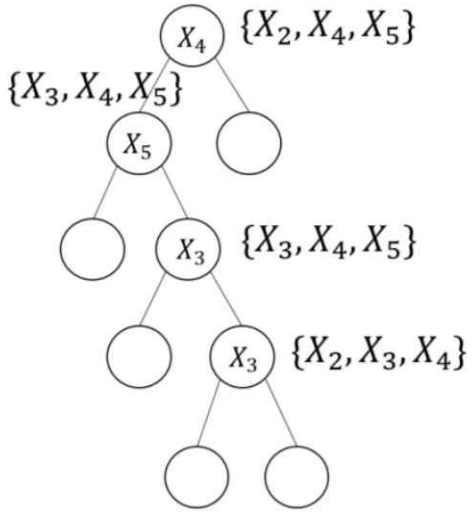
랜덤 포레스트(Random Forest)

- 특징 서브샘플링(Feature Subsampling)
  - 각 노드를 만들 때마다 전체 feature 중 일부만 무작위로 선택하고 그 중에서 최적 분할을 선택
  - 즉, 모든 노드에서 계속 feature를 랜덤하게 샘플링

데이터의 feature :
$\{X_1, X_2, X_3, X_4, X_5\}$



결정트리 1

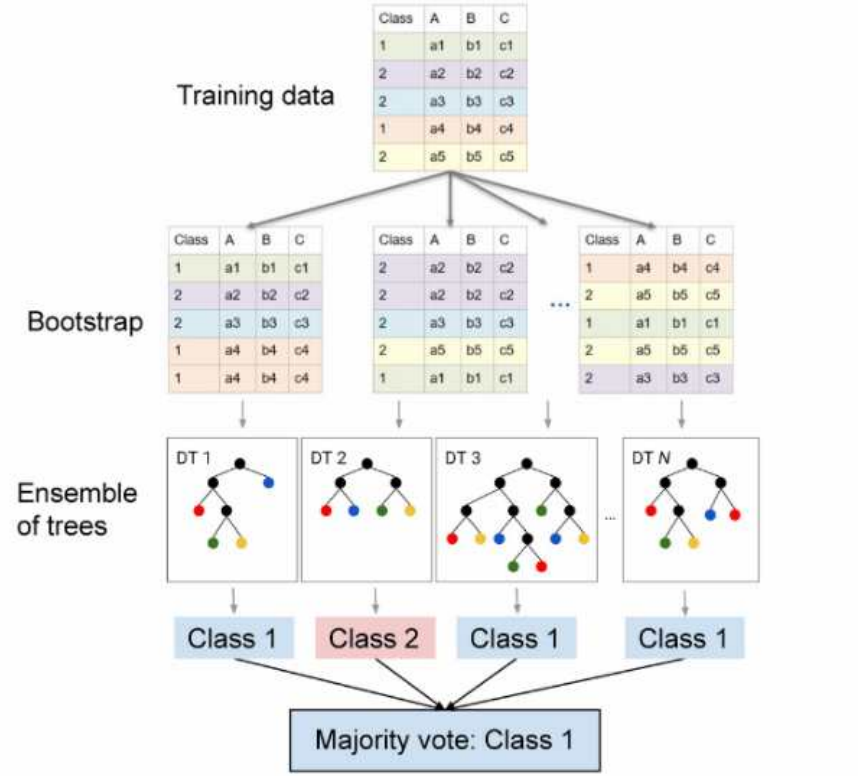


결정트리 2

.....

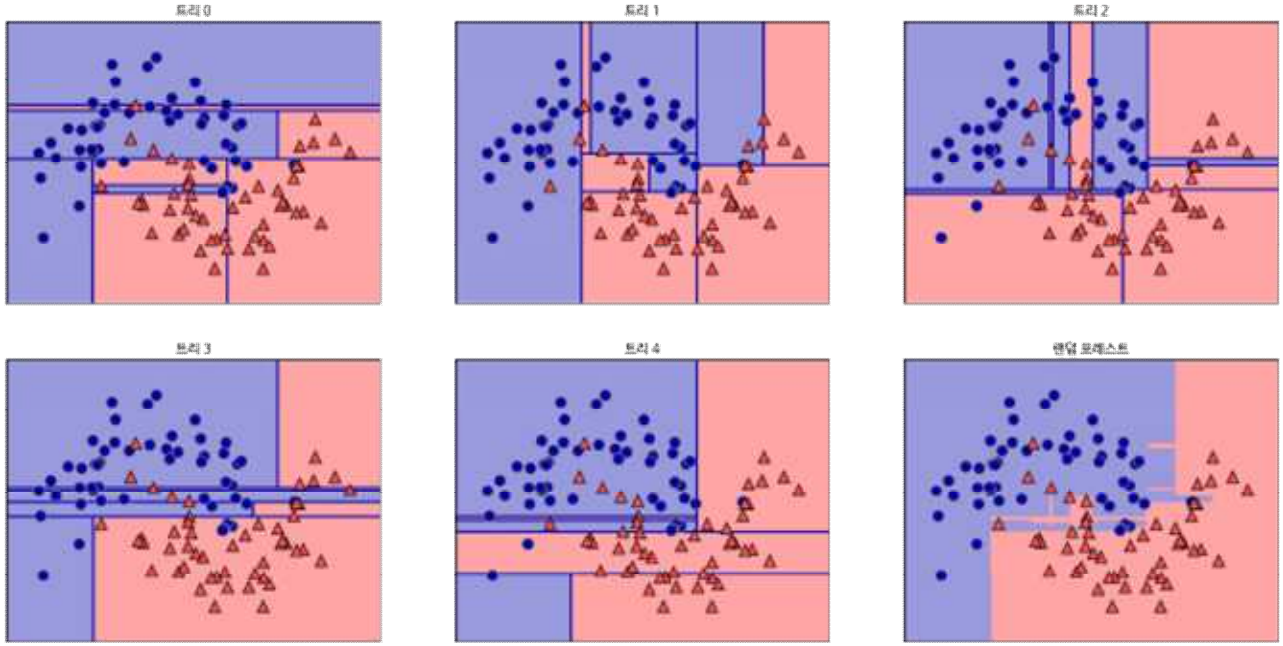
랜덤 포레스트(Random Forest)

- 랜덤 포레스트 전체 과정



랜덤 포레스트(Random Forest)

- 랜덤 포레스트 결과 예시
  - 결정 바운더리(decision boundary)의 예시

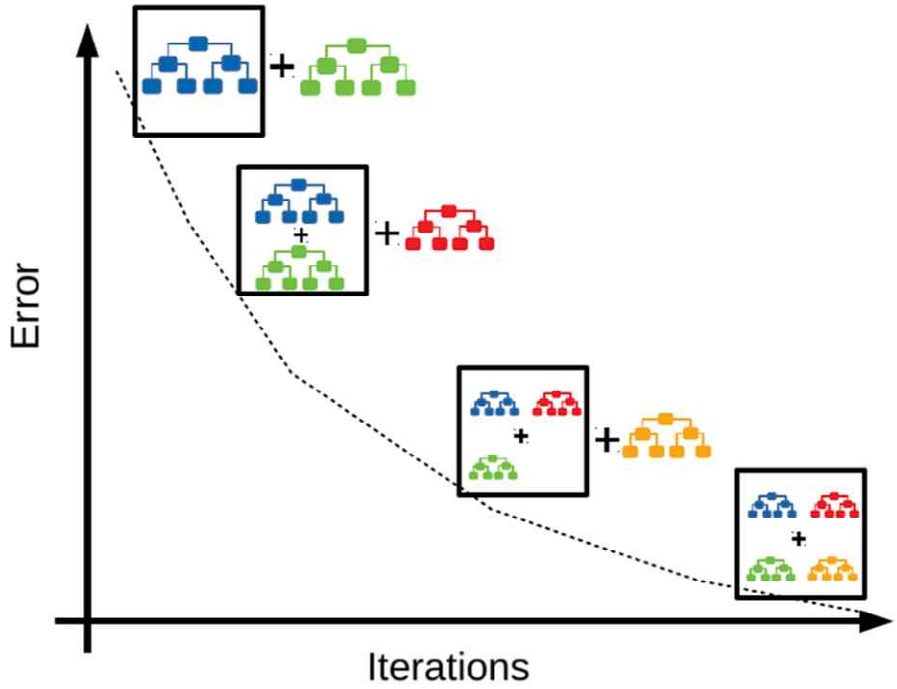


랜덤 포레스트

각 개별 트리보다 일반화 성능이 좋음

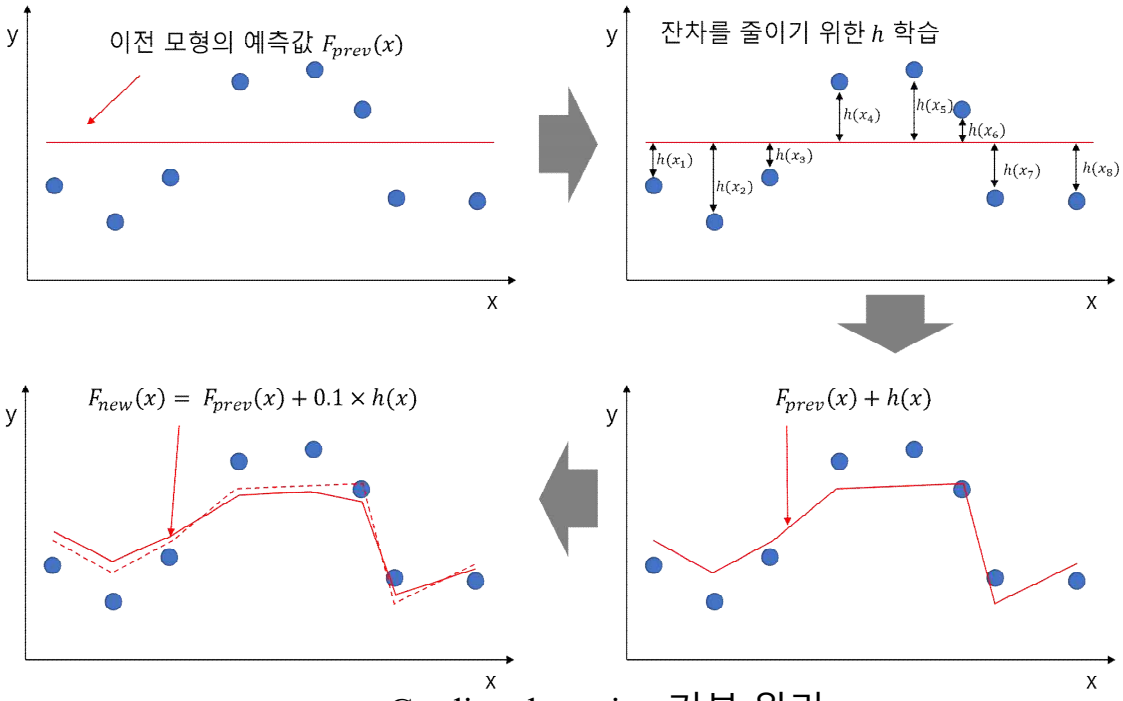
부스팅(Boosting) 기법

- Gradient boosting의 기본 개념
  - 각 단계에서 이전 모델(결정 트리)이 만든 잔차(residual)를 새 결정 트리가 학습하여 점점 오류를 보완해가는 방식



부스팅(Boosting) 기법

- Gradient boosting의 기본 개념
  - 각 단계에서 이전 모델(결정 트리)이 만든 잔차(residual)를 새 결정 트리가 학습하여 점점 오류를 보완해가는 방식



Gradient boosting 기본 원리

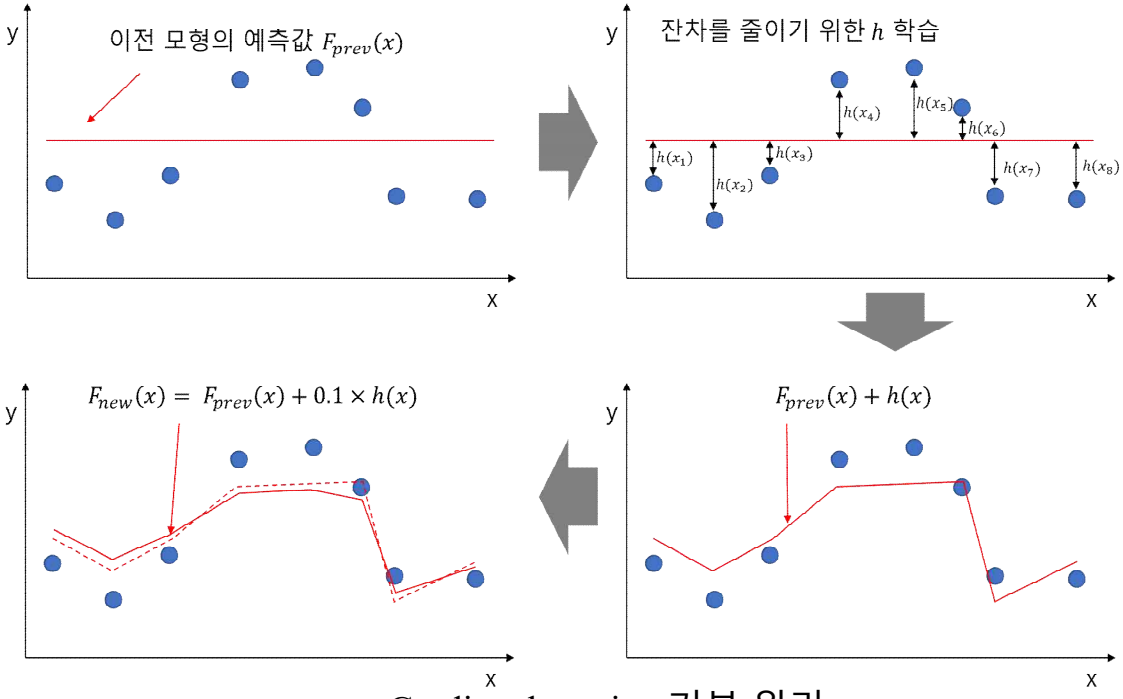
h : 실제값과 예측값의 차이를 줄여주는 함수

$$F_{\text{new}}(x) = F_{\text{prev}}(x) + lh(x)$$

l(학습률)을 곱해주어 overfitting을 방지

부스팅(Boosting) 기법

- Gradient boosting의 기본 개념
  - 각 단계에서 이전 모델(결정 트리)이 만든 잔차(residual)를 새 결정 트리가 학습하여 점점 오류를 보완해가는 방식



Gradient boosting 기본 원리

m : 순차적으로 적합할 모형의 개수
$F_0(x) = h_0(x)$: 첫 번째(시작) 모형의 예측값

최종 모델 $F_m(x) = h_0(x) + lh_1(x) + \ldots + lh_m(x)$

부스팅(Boosting) 기법

- Gradient boosting의 전체 과정
  1. 초기 모델 생성
     - 보통 target의 평균값으로 예측하는 단순한 모델(F_0)
  2. 잔차(residual) 계산

$$r_i = y_i - F_0(x_i)$$

  3. 잔차(residual)를 예측하는 트리 학습
     - 새로운 트리 $h_1(x)$를 만들어 잔차(residual)를 예측
  4. 기존 모델 업데이트

$$F_1(x_i) = F_0(x_i) + lh_1(x)$$

  5. 위 과정을 M번 반복

$$F_M(x) = F_0(x) + l\sum_{m=1}^{M} h_m(x)$$

부스팅(Boosting) 기법

- Gradient boosting의 전체 과정
  1. 초기 모델 생성
     - 보통 target의 평균값으로 예측하는 단순한 모델(F_0)
  2. 잔차(residual) 계산

$$r_i = y_i - F_0(x_i)$$

  3. 잔차(residual)를 예측하는 트리 학습
     - 새로운 트리 $h_1(x)$를 만들어 잔차(residual)를 예측
  4. 기존 모델 업데이트

$$F_1(x_i) = F_0(x_i) + lh_1(x)$$

  5. 위 과정을 M번 반복

$$F_M(x) = F_0(x) + l\sum_{m=1}^{M} h_m(x)$$

손실 함수의 기울기(gradient) 방향으로 모델을 개선

손실 함수를 줄이는 방향으로 모델을 개선

부스팅(Boosting) 기법

- Gradient boosting의 전체 과정
  1. 초기 모델 생성
     - 보통 target의 평균값으로 예측하는 단순한 모델(F_0)
  2. 잔차(residual) 계산

$$r_i = y_i - F_0(x_i)$$

  3. 잔차(residual)를 예측하는 트리 학습
     - 새로운 트리 $h_1(x)$를 만들어 잔차(residual)를 예측
  4. 기존 모델 업데이트

$$F_1(x_i) = F_0(x_i) + lh_1(x)$$

  5. 위 과정을 M번 반복

$$F_M(x) = F_0(x) + l\sum_{m=1}^{M} h_m(x)$$

예: 회귀의 MSE 손실 함수 :

$$L = \frac{1}{2}(y - F(x))^2$$

Gradient : $\frac{\partial L}{\partial F(x)} = -(y - F(x)) = -r_i$

손실 함수의 기울기(gradient) 방향으로 모델을 개선

손실 함수를 줄이는 방향으로 모델을 개선

부스팅(Boosting) 기법

- Gradient boosting에서 결정 트리를 사용하는 이유
  - 매우 유연하고 비선형 구조를 잘 표현하여 잔차를 근사하는 모델로 적합
  - 트리는 분할 기준만 보므로 스케일의 영향을 거의 받지 않음(스케일 안정성이 큼)
  - 깊이를 낮추면 아주 단순한 모델이 되어 약한 모델로도 사용하기 좋음(약한 모델들의 선형 결합)
  - 결정트리는 공간을 분할하고 그 구간마다 서로 다른 잔차를 학습할 수 있음

부스팅(Boosting) 기법

- Gradient boosting와 랜덤 포레스트의 비교

항목	Gradient Boosting	Random Forest
학습 방식	순차적(Sequential): 이전 모델의 오차를 보완하며 쌓음	병렬적(Parallel): 트리 여러 개를 독립적으로 학습
트리 역할	각 트리는 오차 보정(gradient)을 담당	각 트리는 독립적인 예측기
앙상블 방식	예측값을 가중합(learning rate)	평균 또는 다수결
목표	편향(Bias) 줄이기(고성능)	분산(Variance) 줄이기(안정성)
과적합	Overfitting되기 쉬움	비교적 overfitting에 강함
학습 속도	느림 (순차적 구조)	빠름 (병렬 학습 가능)
결정 트리의 깊이	보통 2~4	특별히 깊이 제한 없음

부스팅(Boosting) 기법

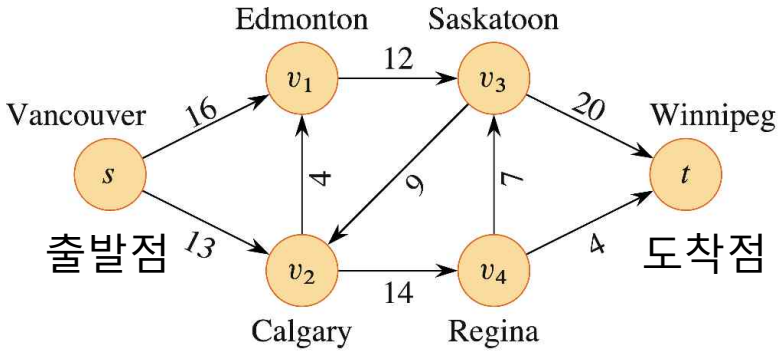
- Gradient boosting 관련 라이브러리
  - XGBoost
    - 결정 트리에 정규화항을 추가하여 Overfitting 문제 방지
    - 결정 트리의 Feature subsampling 지원
    - 손실함수의 1차 미분(gradient) 뿐만 아니라 2차 미분(hessian)도 같이 고려
  - LightGBM
    - Gradient Boosting을 매우 빠르고 메모리 효율적으로 구현
    - GPU 가속 지원

과제 #5

- Minimum Cut Maximum Flow 알고리즘 구현 (2점(미완성 0.5 점, 오작동 1 점))
  - 에드몬드-카프(Edmond-Karp) 알고리즘을 구현하고 아래 플로우 네트워크의 Maximum flow 및 minimum cut을 구하시오

    결과 예시) Maximum flow : 23

    Minimum cut : S = {s, v1, v2, v4}, T = {v3, t}



과제 #4

- 제출 방법 : LMS로 노트북 파일 제출
- 코랩에서 다음과 같은 이름을 가진 노트북 파일을 생성한다.

  Algorithms_5thHW_Class본인소속반_학번_영문이름.ipynb

  예) Algorithms_5thHW_Class1_23xxxxx_HanshinLim.ipynb
- 생성된 코랩 노트북 파일(.ipynb)에서 코드를 구현 및 결과를 출력한다. 결과 출력 시 어떤 결과인지를 명시한다.
- Due : 12월 12일 밤 12시 (이후 제출 0점)

내용 정리

- 결정 트리 알고리즘 분석
- 앙상블 기법 분석

다음 주 강의 내용

- 대표적인 비지도학습 알고리즘 분석
- NP-완비 문제의 의미