LIKE LION

2강

정답이 있는 문제, 어떻게 풀까요?

WHATEVER YOU WANT, MAKE IT REAL.

강사 최윤호

- AI: thinking / acting + humanly / rationally
- 머신러닝: learning from data to improve performance on future tasks
- 딥러닝: hierarchical representation learning (deep neural network + big data + GPU)

- 머신러닝
  - 목표: $\mathrm{E}_{train} \simeq 0$ + $\mathrm{E}_{test} \simeq \mathrm{E}_{train}$
  - 과제: approximation-generalization or bias-variance tradeoff
  - 해결책: big data, optimization, regularization

- 실습: Python, NumPy, Linear Regression

LIKE LION

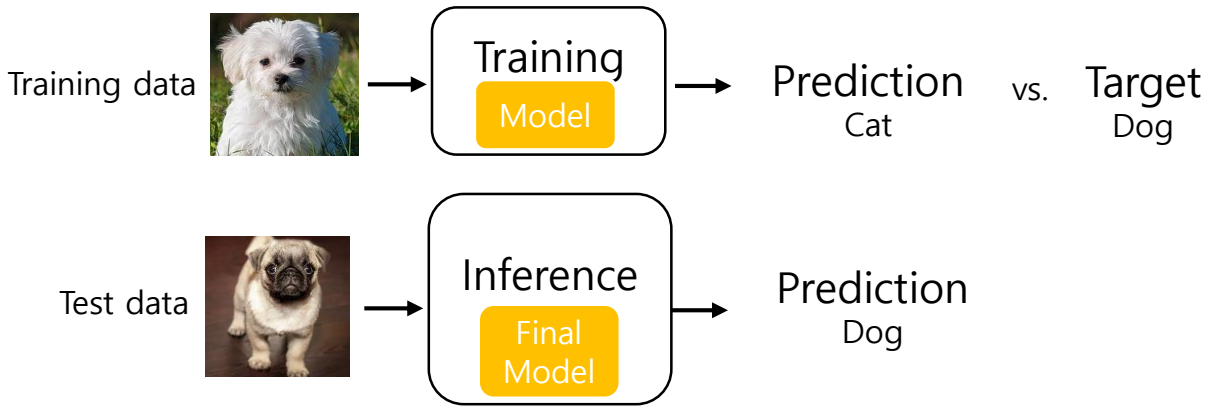
1. Supervised Learning
2. Linear Model
3. Decision Tree
4. Ensemble: Bagging, Boosting
5. (실습) Logistic Regression
6. (실습) Decision Tree

LIKE LION

{ 1. Supervised Learning }

Supervised Learning (지도학습)

- 가장 흔하고, 성공적인 학습 방식
- 정답 레이블이 있는 트레이닝셋 $\{(x_i, y_i): i=1,...,N\}$ 이 주어짐.
  - x : input variable or input feature, y : output or target variable
  - y 는 어떤 함수 $y = f(x)$에 의해 생성
  - 원래의 함수를 가장 잘 근사하는 $h \in \mathcal{H}$(hypothesis) 찾기.
- Classification, Regression



Supervised Learning (지도학습)

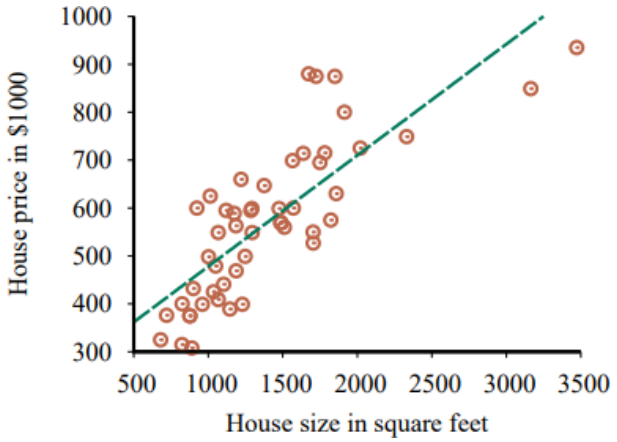
- 학습 알고리즘은 어떤 모델 $h \in \mathcal{H}$ 을 쓰냐에 따라 달라짐.
- 예시
  - Linear Regression
  - Logistic Regression
  - Support Vector Machine (SVM)
  - Naïve Bayes
  - Gaussian Process
  - K-Nearest Neighbors (KNN)
  - Decision Tree, Random Forest
  - Neural network
- 데이터의 형태, 데이터셋의 크기 등에 따라서 다른 알고리즘 사용

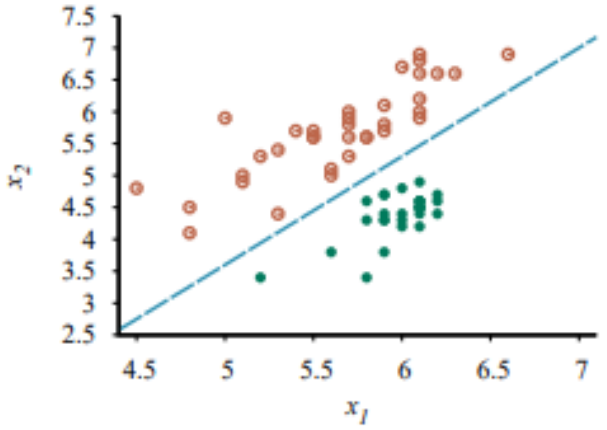
Linear Model

- 다른 복잡한 방식들의 기본이 되는 모델

$$w_1 x_1 + w_2 x_2 + \cdots w_d x_d + b = \mathbf{w}^{\mathrm{T}}\mathbf{x}+\mathrm{b}$$

- linear regression: $y = \mathbf{w}^{\mathrm{T}}\mathbf{x}+\mathrm{b}$
- logistic regression: 분류를 위한 decision boundary를 찾는 것



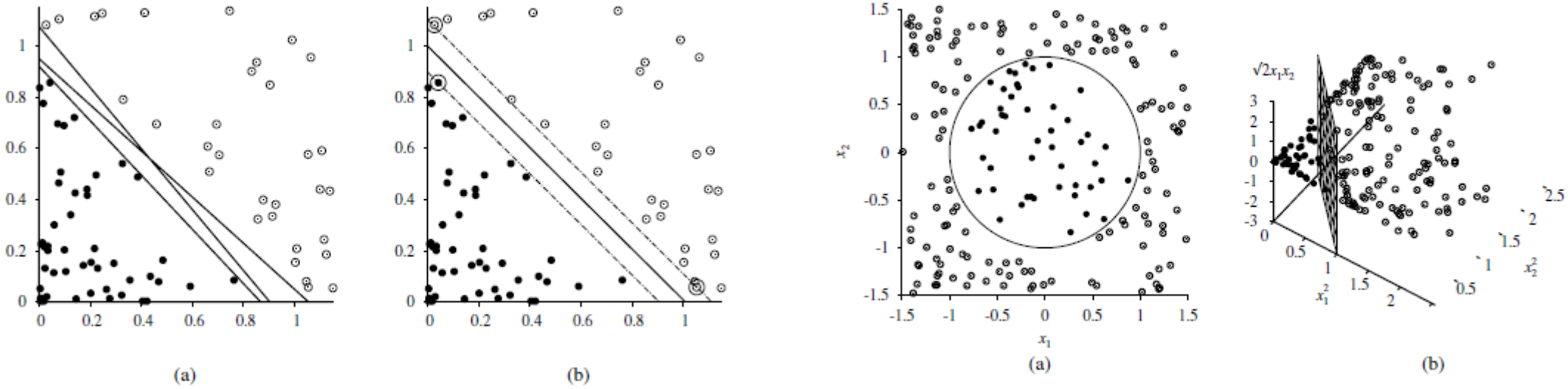


source: http://aima.cs.berkeley.edu/figures.pdf

LIKE LION

Support Vector Machine (SVM)

- 딥러닝 이전에 가장 인기있던 학습 알고리즘
- Maximum margin separator를 찾는 것이 목표.
- 수학적으로 잘 정의되어 Convex Optimization를 통해 해결.
- Linearly inseparable한 경우: Kernel trick을 통해 데이터를 고차원으로 보낸 후 분류
- Input feature가 저차원이고 학습 데이터가 적을 때 유용



source: http://aima.cs.berkeley.edu/figures.pdf

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Naïve Bayes Classification

- Bayes rule + Input feature들이 서로 독립이라고 가정

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

- 적은 트레이닝 데이터로도 잘 작동하고, 계산 속도가 매우 빠름
- Spam filtering, sentiment analysis, recommend system 등에 활용

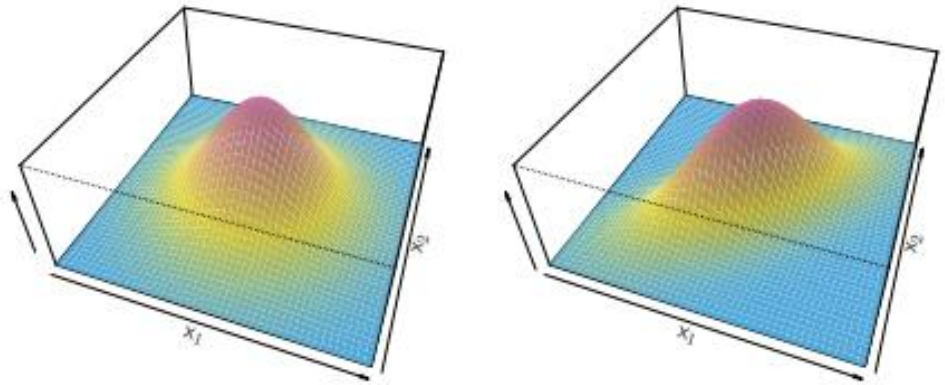
x_1	x_2	x_3	y
키	몸무게	발사이즈	성별
164	52	250	여
181	73	270	남
159	48	240	여
177	65	260	남

LIKE LION

Gaussian Process

- 데이터 $\{x_i, f(x_i)\}$ 가 가우시안 프로세스에 의해 생성됐다고 가정.
- 즉, output이 jointly Gaussian이라고 가정 (Multivariate Gaussian)
- 새로운 input x_*에 대한 output f_*를 조건부 확률을 통해 계산
- 분산값을 통해 Confidence를 알 수 있음
- input feature가 고차원이거나 데이터가 많아지면 적용이 어려워짐

$$\begin{pmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ \dots \\ f(x_n) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(x_1) \\ m(x_2) \\ m(x_3) \\ \dots \\ m(x_n) \end{pmatrix}, \begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) & k(x_1,x_3) & \dots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & k(x_2,x_3) & \dots & k(x_2,x_n) \\ k(x_3,x_1) & k(x_3,x_2) & k(x_3,x_3) & \dots & k(x_3,x_n) \\ \dots & \dots & \dots & \dots & \dots \\ k(x_n,x_1) & k(x_n,x_2) & k(x_n,x_3) & \dots & k(x_n,x_n) \end{pmatrix} \right)$$

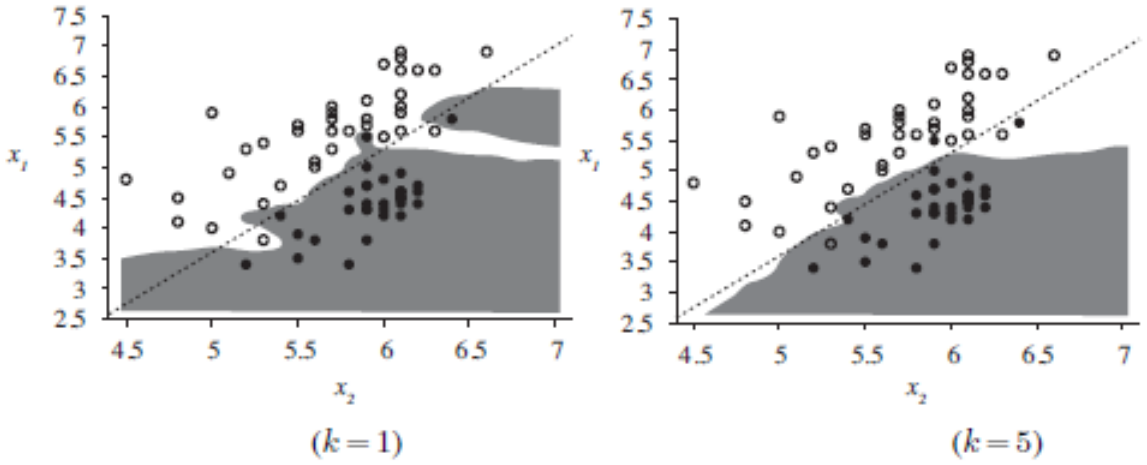


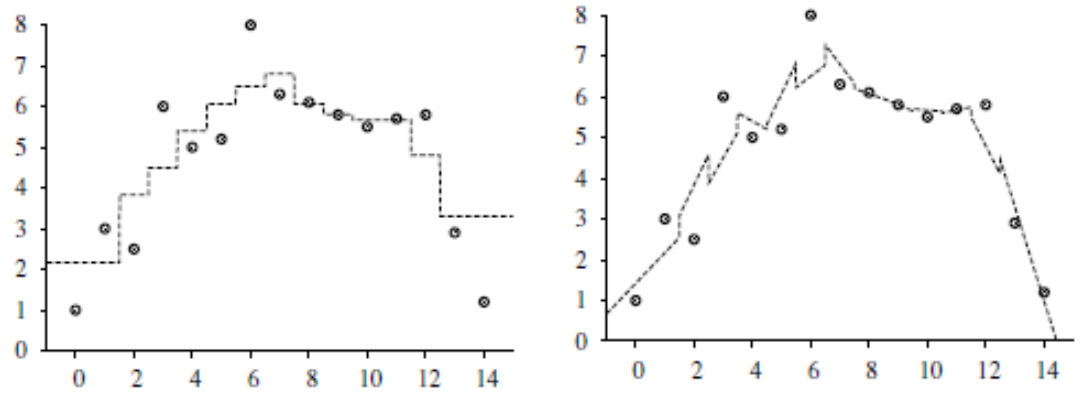
https://www.statlearning.com/resources-second-edition

LIKE LION

K-Nearest Neighbors (KNN)

- Nonparametric approach: # of data ↑ → # of parameters
- Classification: k-nearest neighbors에서 다수결
- Regression: k-nearest neighbors의 average 혹은 linear regression
- Curse of dimensionality: input feature가 저차원이고 학습 데이터가 많을 때 잘 작동
- 단, 학습 데이터가 많으면 계산 cost가 커짐



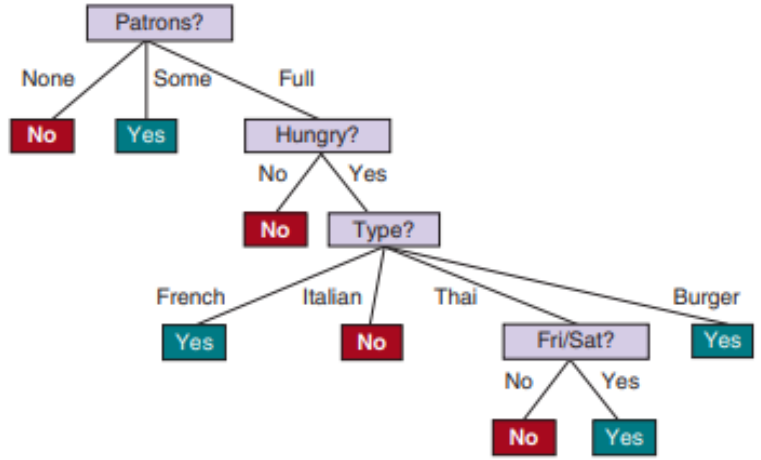


source: http://aima.cs.berkeley.edu/figures.pdf

Decision Tree, Random Forest

- classification, regression에 모두 적용 가능한 간단하지만 강력한 접근법
- Explainable! (사람의 사고방식과 유사)
- 계산 비용 대비 성능이 좋음
- Overfitting 되기 쉬움

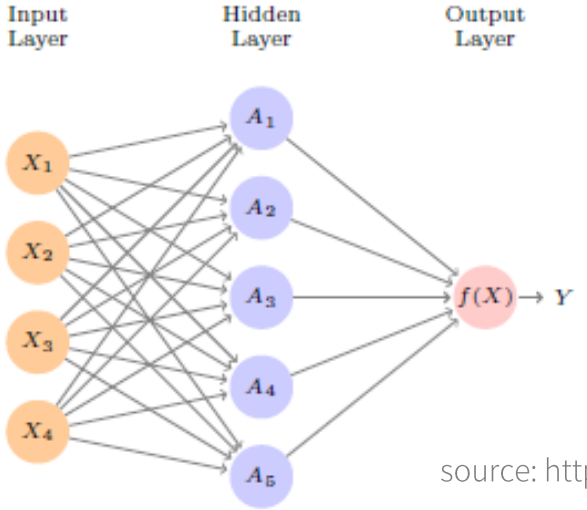
- Random Forest는 Decision Tree의 Ensemble
- Variance 감소 효과



source: http://aima.cs.berkeley.edu/figures.pdf

Neural Network

- 일반적으로 좋은 성능을 내지만 explainability는 떨어지는 black box model
- Gradient descent를 통해 학습
- 학습이 비교적 오래 걸리고, Overfitting이 잘됨.
  - Regularization 기술들을 잘 써야하고, 학습데이터가 많아야 함.
- input 데이터가 고차원 (ex. 이미지)인 경우에도 효과적

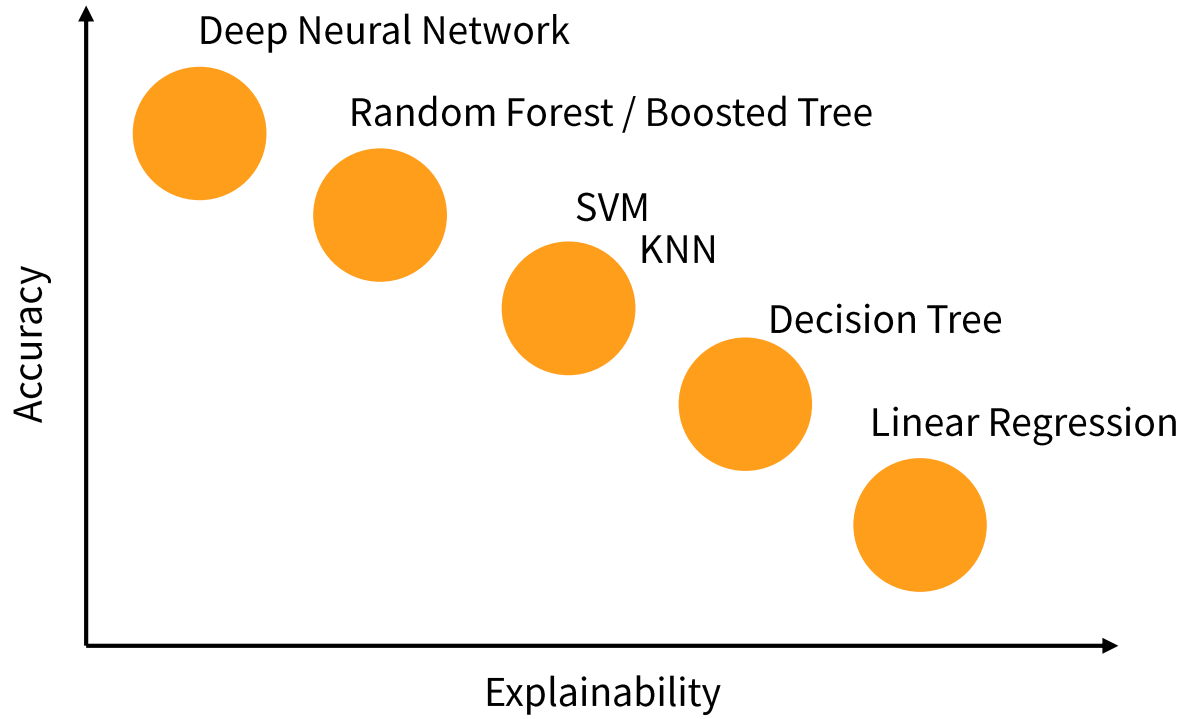


$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X)$$
$$= \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j).$$

source: https://www.statlearning.com/resources-second-edition

Wrap-up

- 상황에 맞는 알고리즘 사용해야 함. (데이터 형태, 개수, computational budget)
- AI 기반 서비스의 신뢰도를 위해서는 explainability도 중요함

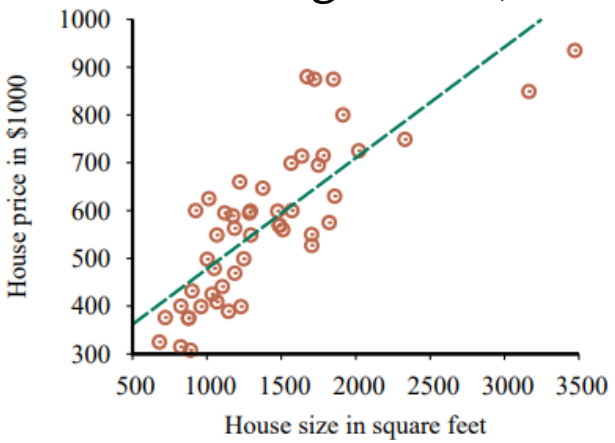


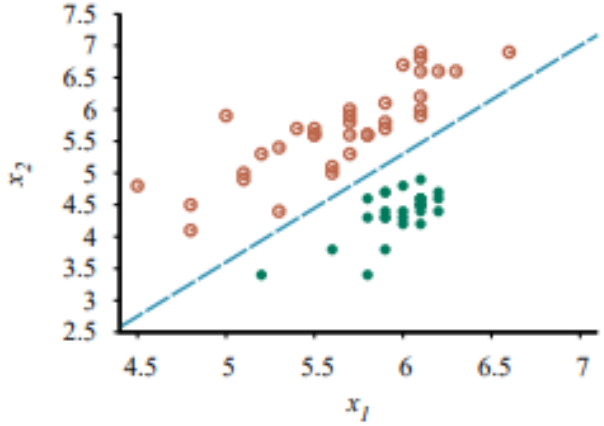
LIKE LION

{ 2. Linear Model }

Linear Model

- 다른 복잡한 방식들의 기본이 되는 모델
- Input feature $\mathbf{x}$가 d차원일 때, $w_1 x_1 + w_2 x_2 + \cdots w_d x_d + b = \mathbf{w}^{\mathrm{T}}\mathbf{x}+\mathrm{b}$
- 다양한 장점
  - 구현하기 쉽고, 해석하기 쉬움
  - 간단한 모델 → Generalization ↑
  - 확장성: nonlinear transform, kernel trick, neural nets
- Regression, Classification





source: http://aima.cs.berkeley.edu/figures.pdf

Linear Regression

- input, output 모두 1-D인 경우 예시

$$h_{\mathbf{w}}(x) = w_1 x + w_0, \text{ where } w_0, w_1 \in \mathbb{R}.$$

- 트레이닝 셋 $\{(x_i, y_i): i=1,...,N\}$에 가장 fit한 $h_{\mathbf{w}}$를 찾기: 손실함수는 MSE (Mean Squared Error)

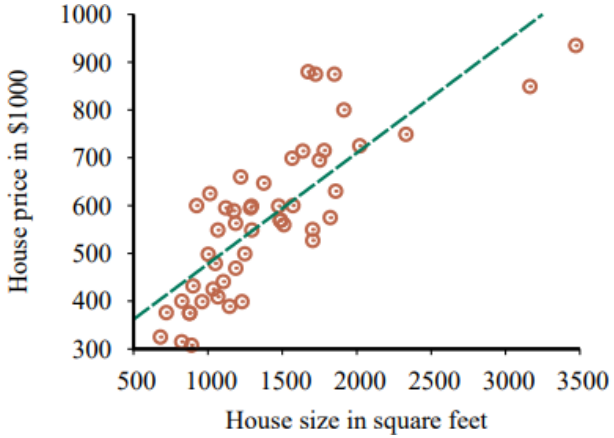
$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} Loss(h_{\mathbf{w}})$$

- Closed-form solution

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 \text{ and } \frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0$$

$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}; \quad w_0 = (\sum y_j - w_1(\sum x_j))/N \ .$$



source: http://aima.cs.berkeley.edu/figures.pdf

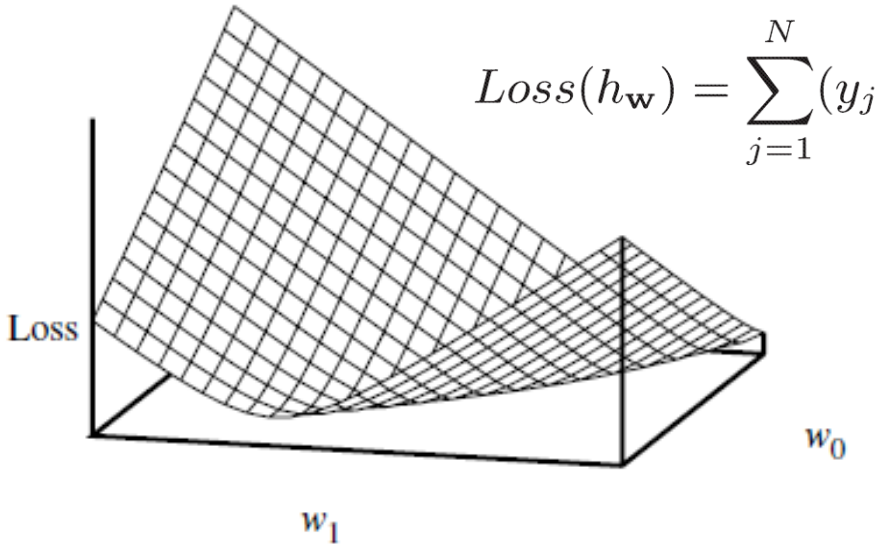
Linear Regression

- Closed-form solution을
  계산하기 어려운 경우 경사하강법 사용

w ← any point in the parameter space
loop until convergence **do**
  for each w_i **in w do**

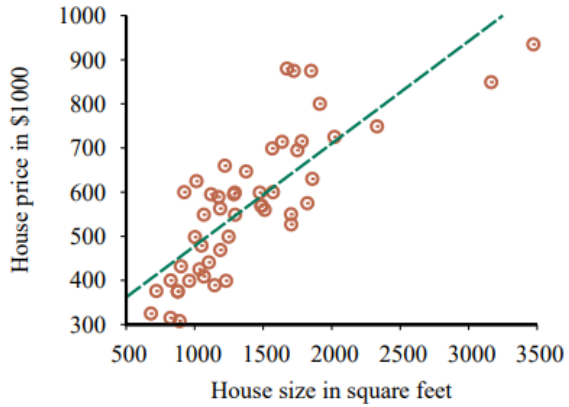
$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

α: step size or learning rate.



$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2.$$

$$y = 0.232x + 246.$$



$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)); \quad w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$$

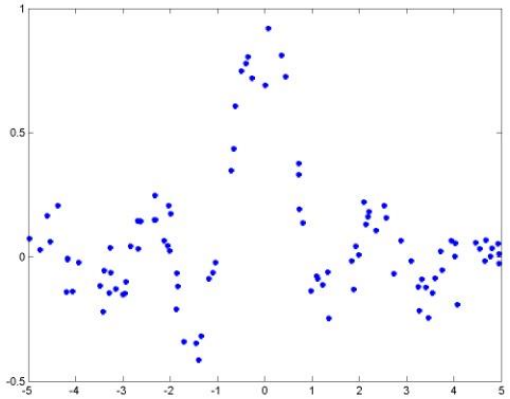
source: http://aima.cs.berkeley.edu/figures.pdf

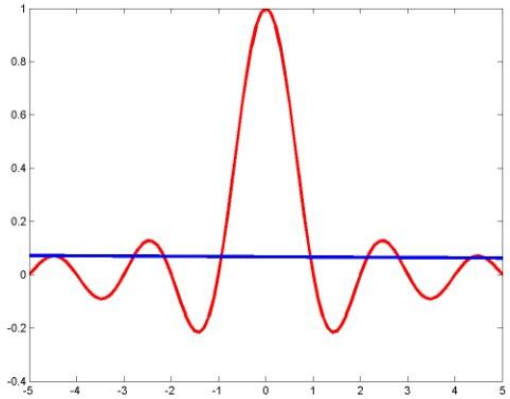
Linear Regression

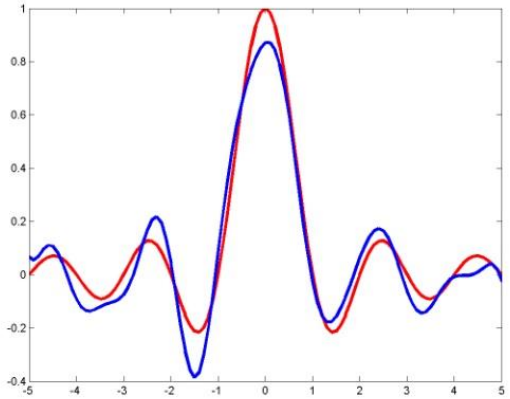
- General linear model: nonlinear 함수 $h_i(x)$ 를 이용

$$y = \sum_{i=1}^{M} w_i h_i(x) \quad , \qquad h_i(x) = x^{i-1}$$

$$h_i(x) = \exp\left(-\frac{1}{2s^2}(x - \mu_i)^2\right)$$



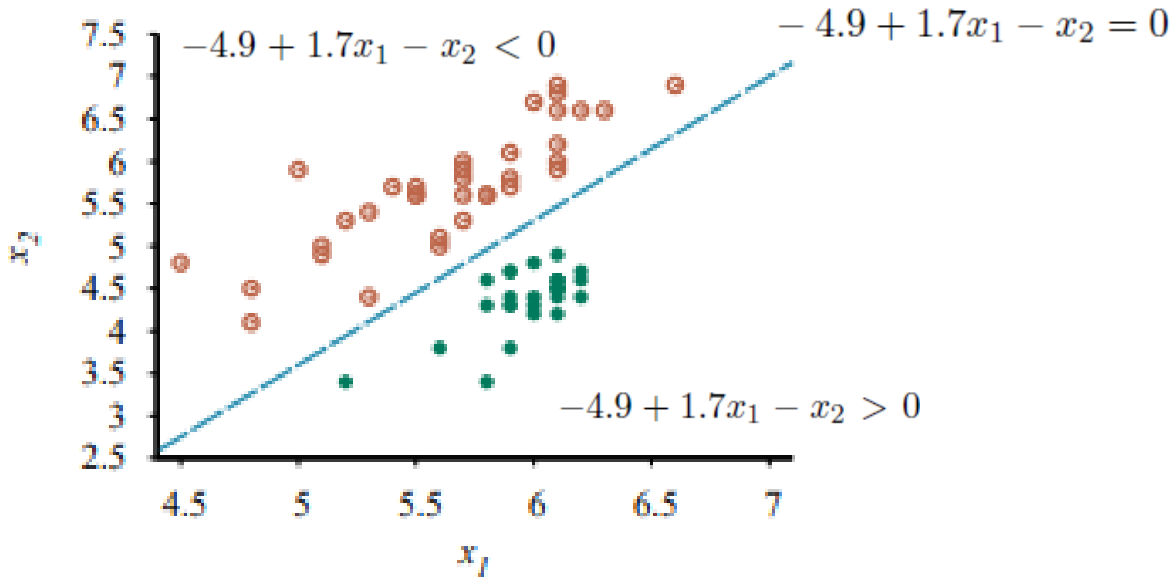




source: http://aima.cs.berkeley.edu/figures.pdf

Linear Classification

- binary classification 예시: $y_i = \{0, 1\}$
- Decision boundary (a line separating two classes)를 찾는 것이 목표



source: http://aima.cs.berkeley.edu/figures.pdf

Linear Classification

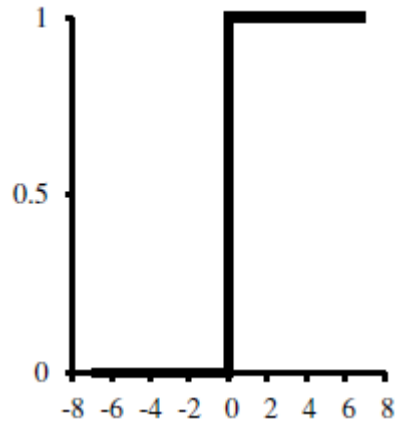
- regression과 달리 output의 범위가 있음: threshold function을 사용한 분류

$$h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w} \cdot \mathbf{x}) \text{ where } Threshold(z) = 1 \text{ if } z \geq 0 \text{ and } 0 \text{ otherwise.}$$

- Perceptron Learning Rule

$$w_i \leftarrow w_i + \alpha\,(y - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$

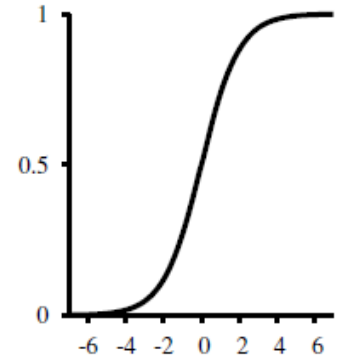
- Linearly separable한 경우에만 수렴



source: http://aima.cs.berkeley.edu/figures.pdf

Logistic Regression

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



source: http://aima.cs.berkeley.edu/figures.pdf

- logistic function을 이용한 분류

$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}}}$$

- 손실함수로 MSE 대신 log loss 사용 (why?)

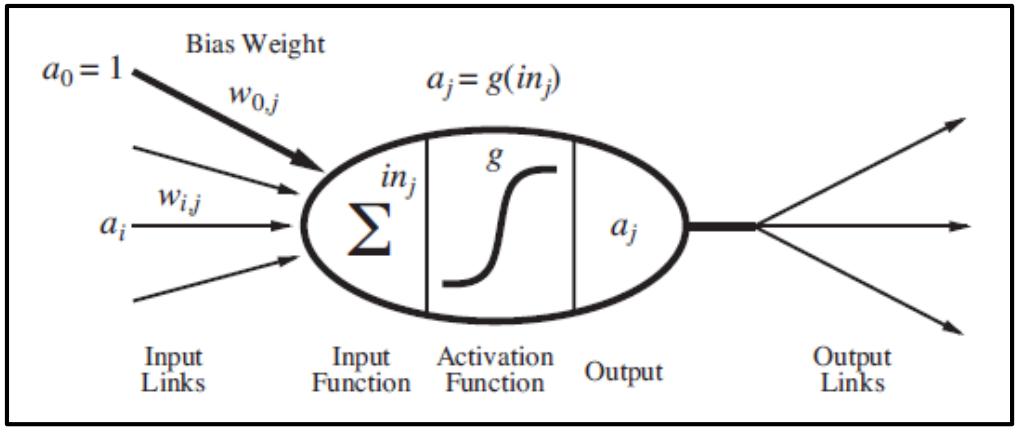
$$Loss(w) = -\frac{1}{N}\sum_i y_i \log(h_{\mathbf{w}}(\mathbf{x}_i)) + (1-y_i)\log(1-h_{\mathbf{w}}(\mathbf{x}_i))$$

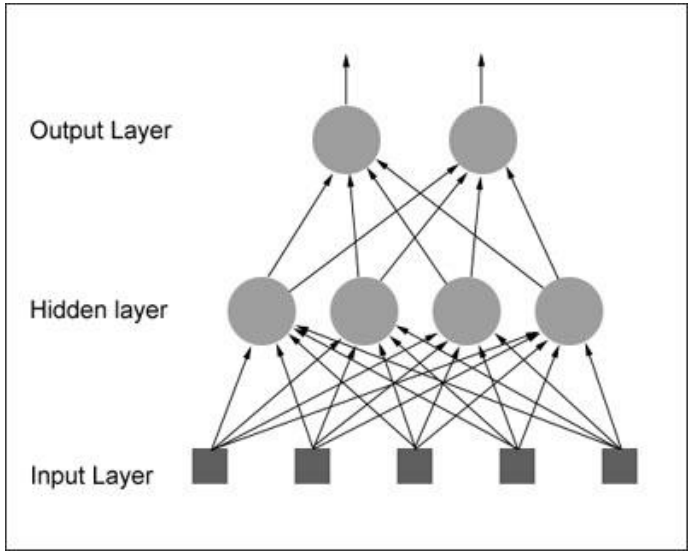
- 경사하강법을 통해 업데이트

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

Neural Network로의 확장

- 뉴런 모델:



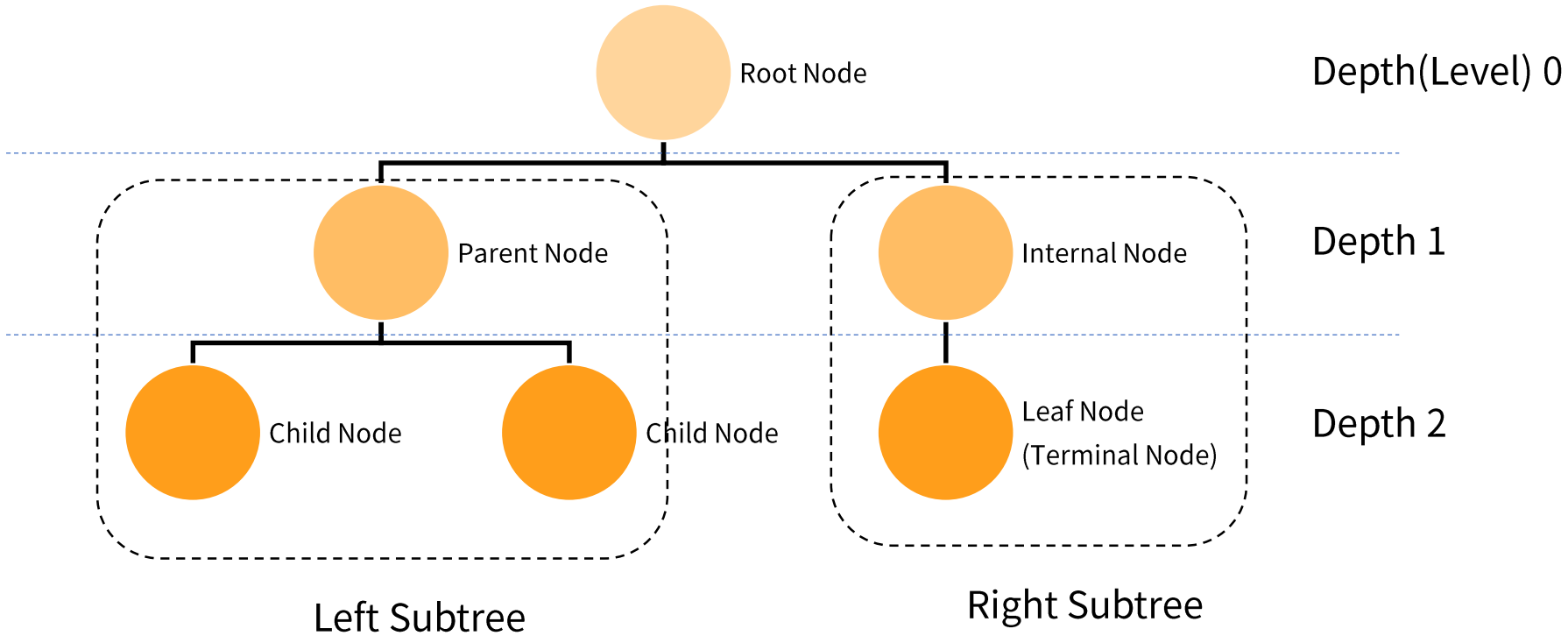


source: http://aima.cs.berkeley.edu/figures.pdf

LIKE LION

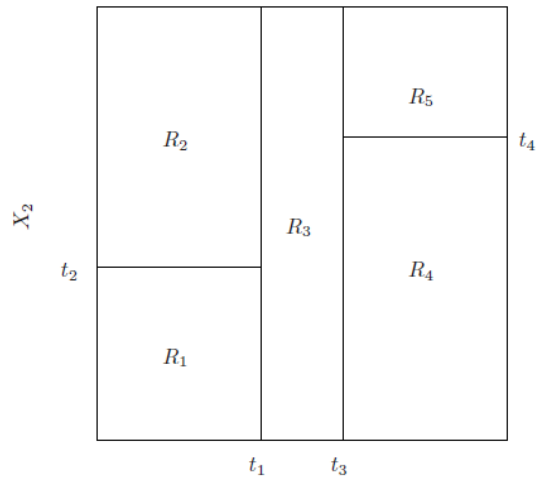
{ 3. Decision Tree }

Tree Basics

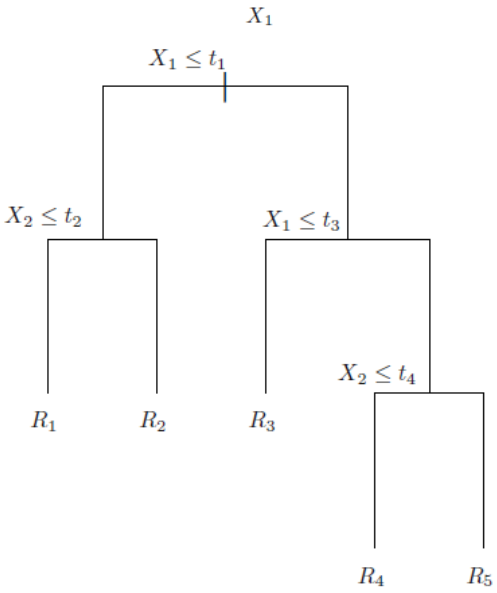

Root Node
Depth(Level) 0
Parent Node
Internal Node
Depth 1
Child Node
Child Node
Leaf Node
(Terminal Node)
Depth 2
Left Subtree
Right Subtree

LIKE LION

Decision Tree Basics

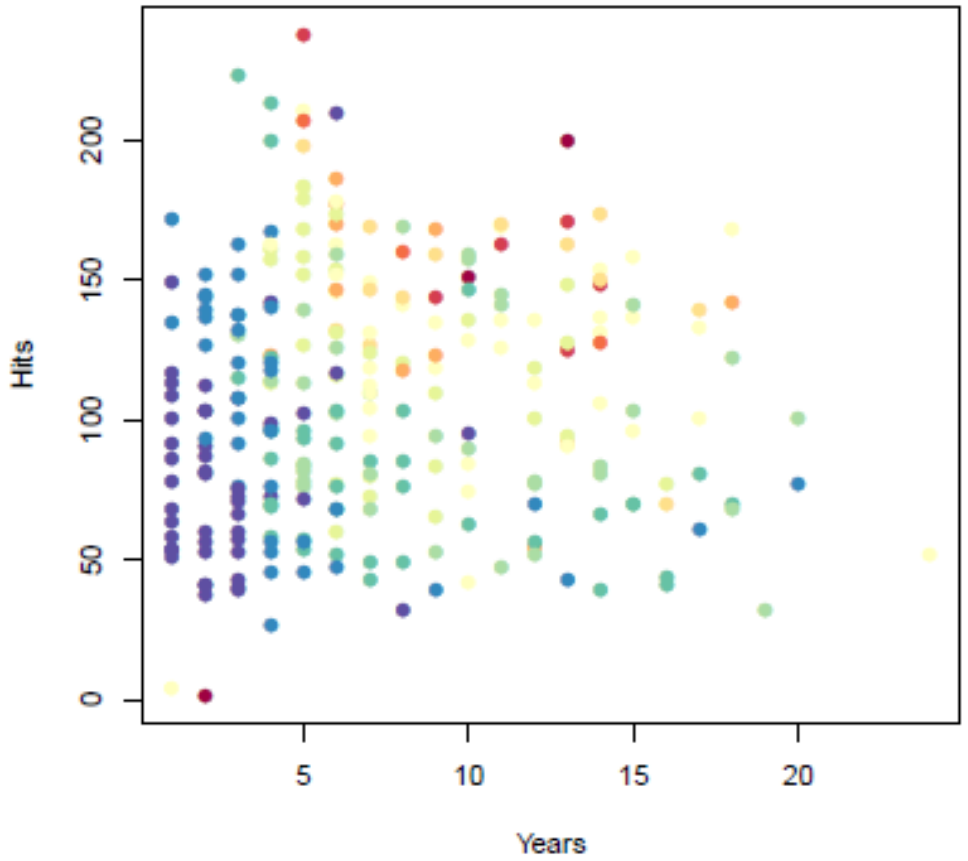


- 예측변수(Predictor): input feature
- Decision Tree (의사결정트리): 예측변수의 전체 공간을 단순한 여러 영역으로 계층화, 분할하는 것.
- 공간을 분할하는 규칙을 tree 형식으로 나타낼 수 있음.
  - internal node에서는 split(분기)가 일어남
  - 분할된 영역들은 leaf node들에 위치
- 단순하고 설명력이 좋아서 현업에서 많이 쓰임.
- Regression, Classification 모두에 적용 가능



source: https://www.statlearning.com/resources-second-edition

Regression Tree

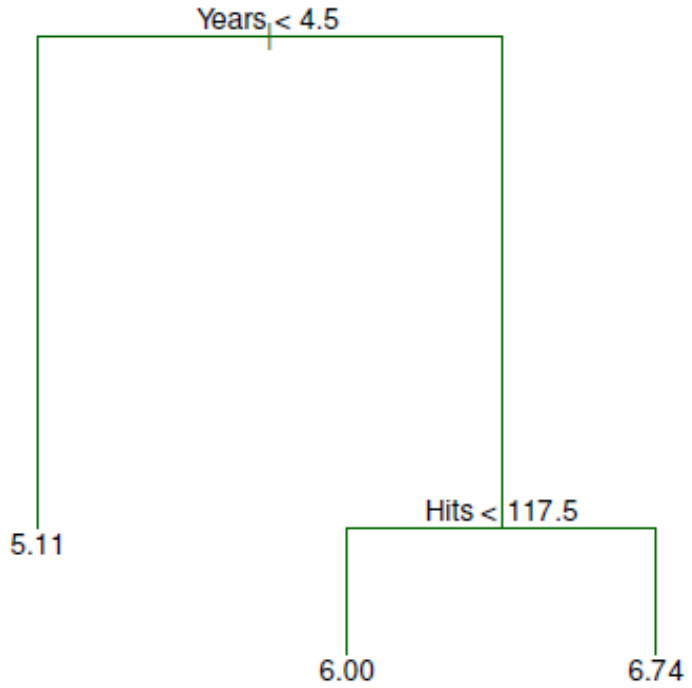


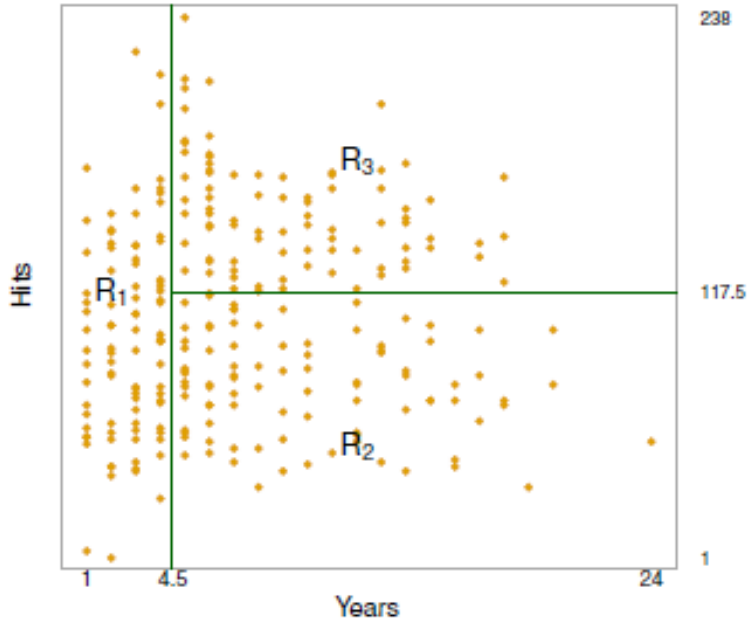
source: https://www.statlearning.com/resources-second-edition

- 메이저리그 타자들의 급여 데이터
- 급여는 color-coded 되어있음
- 예측변수: Hits, Years

LIKE LION

Regression Tree





- Regression Tree
  - 분기 조건에 부합하면 왼쪽 child node로, 아니면 오른쪽 child node로
  - Leaf의 숫자는 해당 영역에 속하는 training data의 평균
  - Years가 더 중요한 결정요소
  - Years가 어느정도 이상될 때 Hits가 영향
  - 문제를 너무 단순화 but 시각화, 해석, 설명하기 좋음

source: https://www.statlearning.com/resources-second-edition

Regression Tree

- Regression Tree를 만드는 방법
  - 전체 예측변수 공간을 J개의 겹치지않는 영역 $R_1, R_2, ..., R_J$ 으로 분할한다.
  - 각 영역 안에서는 그곳에 속하는 training data의 평균을 통해 일관된 예측값을 반환한다.

- J개의 영역으로 분할은 어떻게?

$$\{X | l_1 \leq X_1 \leq u_1, l_2 \leq X_2 \leq u_2, ..., l_p \leq X_p \leq u_p\}$$

  - 이론상 어떤 모양이든 가능하지만, decision tree는 해석의 편의를 위해 Box들로 분할한다.
  - Sum of squared errors (SSE)을 최소화하는 box를 찾는 것이 목표이다.

→ $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$, $\hat{y}_{R_j}$ 는 j번째 box 안의 output 평균값

Regression Tree

- 그러나 모든 가능한 box를 고려하는 것은 불가능
- 따라서, top-down, greedy 접근을 사용한다: Recursive Binary Splitting
  - Top down: tree의 꼭대기부터 차례대로 binary split
  - Greedy: future step들까지 고려하여 더 좋은 tree를 찾는 것이 아닌, 당장 singe step 이후의 SSE를 최소화하는 split을 찾는다.

- 즉, 각 internal node에서 매번 아래의 SSE를 최소화하는 예측변수 X_j와 분할점 s를 찾는다.

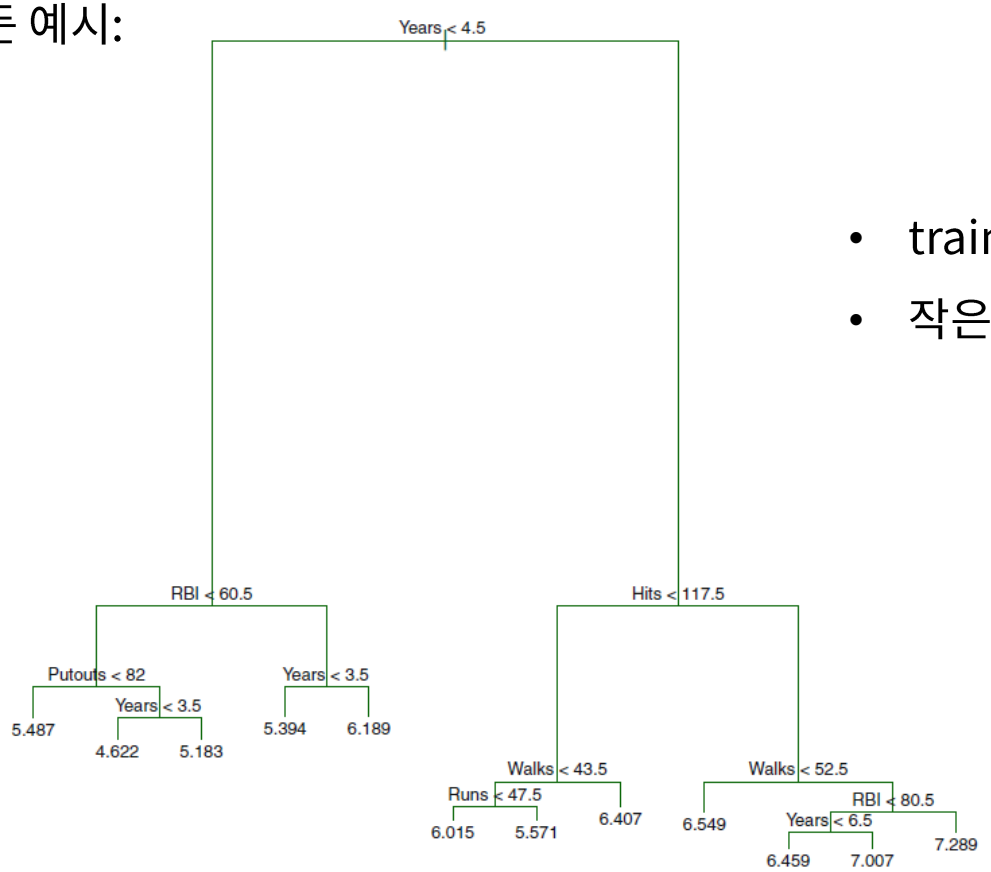
$$R_1(j,s) = \{X | X_j < s\} \text{ and } R_2(j,s) = \{X | X_j \geq s\}$$

$$\text{SSE} = \sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

LIKE LION

Regression Tree

- Stopping criterion 예시: 모든 leaf가 5개 이하의 데이터를 포함할 때 까지
- 최소 기준으로 Tree를 만든 예시:



- training data에 overfit
- 작은 tree가 variance도 적고 설명력이 좋다

source: https://www.statlearning.com/resources-second-edition

Pruning a Tree

- SSE를 감소시킨 비율이 어느정도 이상일 때 split ? → 근시안적
- 큰 tree T_0를 만든 다음에 pruning 해주는게 낫다.
- **Cost complexity pruning (weakest link pruning)** :
  - 아래 식을 최소화하는 Subtree $T \subset T_0$ 를 찾는다. α는 음이 아닌 tuning parameter.

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

T의 leaf node의 숫자

m번째 leaf node의 영역

  - α가 subtree의 크기와 training data와의 fitting 사이의 trade-off를 조절한다.

→ cross validation을 통해서 결정한다.

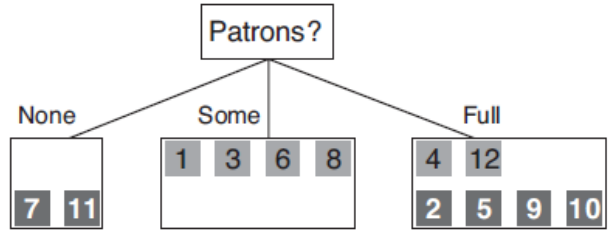
Tree Algorithm

1. Recursive binary splitting을 이용해 큰 tree를 만든다. stop 기준은 각 leaf에 최소 기준보다 적은 수의 data가 남는 것이다.
2. Cost complexity pruning을 통해서 큰 tree를 pruning한다.
3. K-fold cross validation을 통해 α를 찾는다. 즉, k=1,…, K에 대하여 아래를 반복한다.
    1. k번째 그룹을 제외한 트레이닝 데이터에 대해 스텝 1과 2를 적용한다.
    2. k번째 그룹의 데이터에 대해 SSE를 계산한다.

    SSE를 평균내어 validation error를 계산하고 이를 가장 적게 하는 α를 고른다.
4. 골라진 α로 스텝 2에서의 큰 tree에 pruning을 한 subtree를 반환한다.

LIKE LION

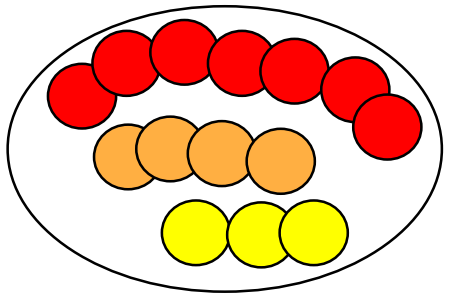
Classification Tree



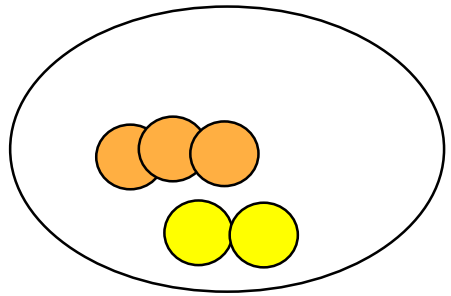
- 최종적으로 class를 예측하는 것만 다르고 regression tree와 매우 유사
- leaf node에서의 training data중 가장 많이 등장하는 class가 예측 class가 된다.
- split 조건으로 SSE가 아닌 다음 척도들을 고려한다.
  - Classification error rate $E = 1 - \max_k(\hat{p}_{mk})$.
    $\hat{p}_{mk}$는 m번째 영역 안의 데이터 중 k번째 class인 것들의 비율
  - Gini index
    $$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$
  - Entropy
    $$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$
    ← fair coin flip의 경우 $D = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1\ bits$
    biased coin flip의 경우 $D = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) \approx 0.08\ bits$
  - Gini index와 Entropy가 split 척도로 선호됨. 이들은 node impurity라고도 함
- Pruning에서는 최종 예측의 정확도를 위해 Classification error rate를 주로 사용.

source: http://aima.cs.berkeley.edu/figures.pdf

Classification Tree



- Classification error rate: $1 - \frac{7}{14} = 0.5$
- Gini index: $\frac{7}{14}\left(1 - \frac{7}{14}\right) + \frac{4}{14}\left(1 - \frac{4}{14}\right) + \frac{3}{14}\left(1 - \frac{3}{14}\right) = 0.6224$
- Entropy: $-(\frac{7}{14}\log_2\frac{7}{14} + \frac{4}{14}\log_2\frac{4}{14} + \frac{3}{14}\log_2\frac{3}{14}) = 1.4926\ bits$

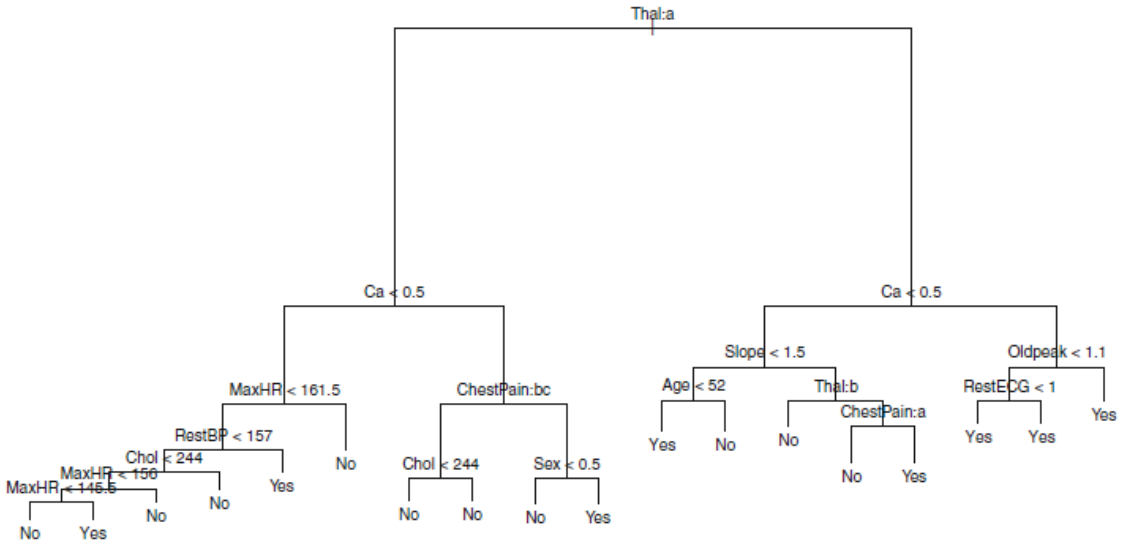


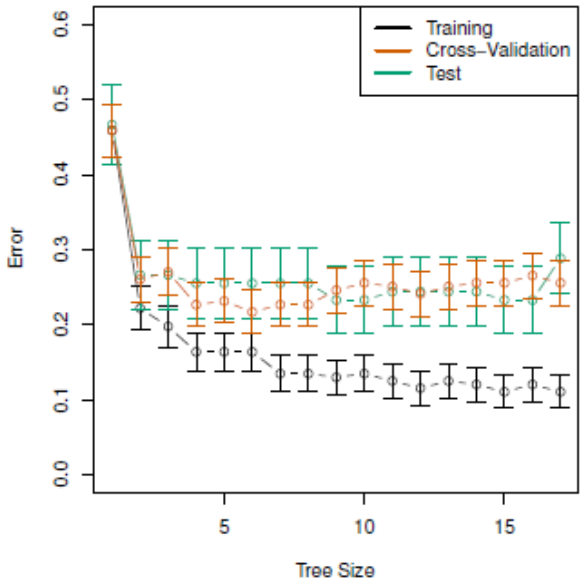
- Classification error rate: $1 - \frac{3}{5} = 0.4$
- Gini index: $\frac{3}{5}\left(1 - \frac{3}{5}\right) + \frac{2}{5}\left(1 - \frac{2}{5}\right) = 0.48$
- Entropy: $-(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}) = 0.9710\ bits$

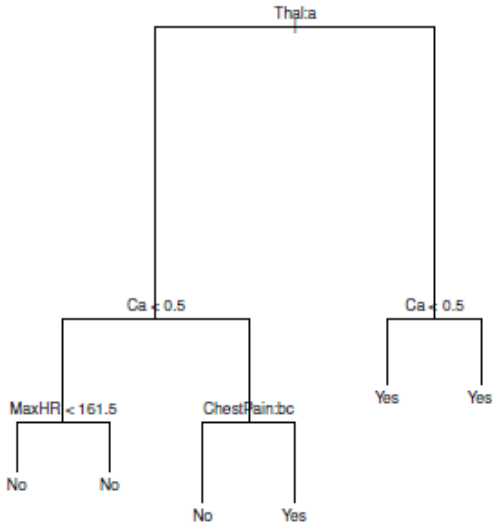
LIKE LION

Classification Tree

- 심장질환 유무 분류

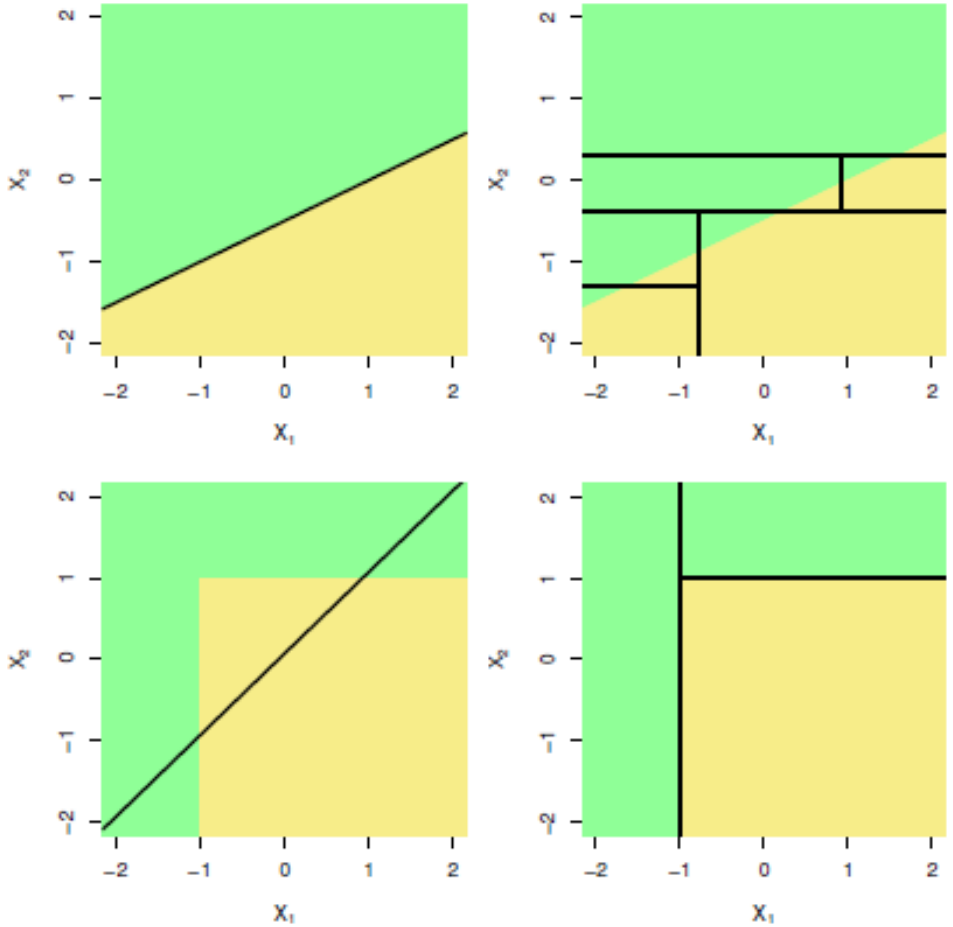






source: https://www.statlearning.com/resources-second-edition

Linear Model vs. Decision Tree



source: https://www.statlearning.com/resources-second-edition

Decision Tree의 장단점

↑ 설명력이 아주 뛰어나다. 사람의 의사결정과 닮았다.

↑ 시각화하기 좋다.

↑ 양적 변수, 질적 변수를 모두 다룰 수 있다.

↓ 예측 정확도가 좋지는 못하다.

↓ 강인하지 못하다(Non-robust). input data의 작은 변화에도 최종 예측값은 크게 변화할 수 있다.

→ 여러 개의 decision tree를 종합하는 방법을 이용하여 극복 가능!

LIKE LION

{ 4. Ensemble: Bagging, Boosting }

Ensemble Methods

- 여러 개의 간단한 “building block” 모델들을 결합해 하나의 강력한 모델을 만드는 법
  - 이 모델들을 weak learners라고 하기도 함

- Decision Tree를 building block으로 활용하면 Bagging, Random Forest, Boosting 등의 Ensemble 기법을 활용할 수 있음

Bagging

- Bootstrap aggregation
- Recall) n개의 variance가 σ^2인 독립변수 $Z_1, \ldots, Z_n$이 있을 때 이들의 평균 $\overline{Z}$의 variance는 $\frac{\sigma^2}{n}$이다.
- Idea) 따라서 다른 tree 여러 개의 결과를 평균해서 최종 예측을 하면 variance를 줄일 수 있지 않을까?

- Bootstrapping: 데이터셋에서 random한 복원추출을 통해 B개의 bootstrapped 데이터셋을 만드는 것
- Bagging: b번째 bootstrapped 데이터셋에 모델을 학습시켜 $\hat{f}^{*b}(x)$ 를 얻고, 전체 B개의 모델의 예측에 평균을 취해서 최종 예측값을 얻는다. classification tree의 경우 B개의 vote에 대해 다수결로 결정

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

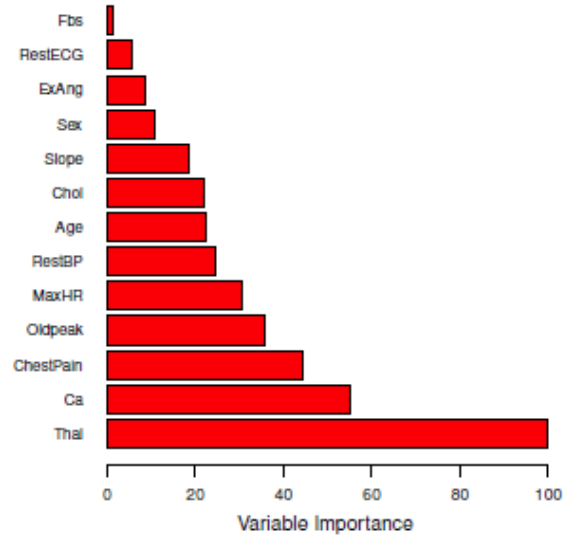
- 이 때 B개의 tree들은 pruning을 하지 않는다 → Low bias, high variance

Bagging

- Out-of-Bag (OOB) error estimation:
  - Bootstrapping 시에 복원추출을 하기때문에 전체 데이터의 약 1/3 정도는 한 번도 뽑히지 않게 된다!
  - 이 데이터들을 OOB 데이터라고 부르고, 이들을 활용해서 validation error를 계산할 수 있다.

- Variable Importance Measures:
  - Bagging은 설명력을 희생하여 예측 성능을 올린 것
  - SSE를 통하여 예측변수들의 중요도를 확인하는 방법:

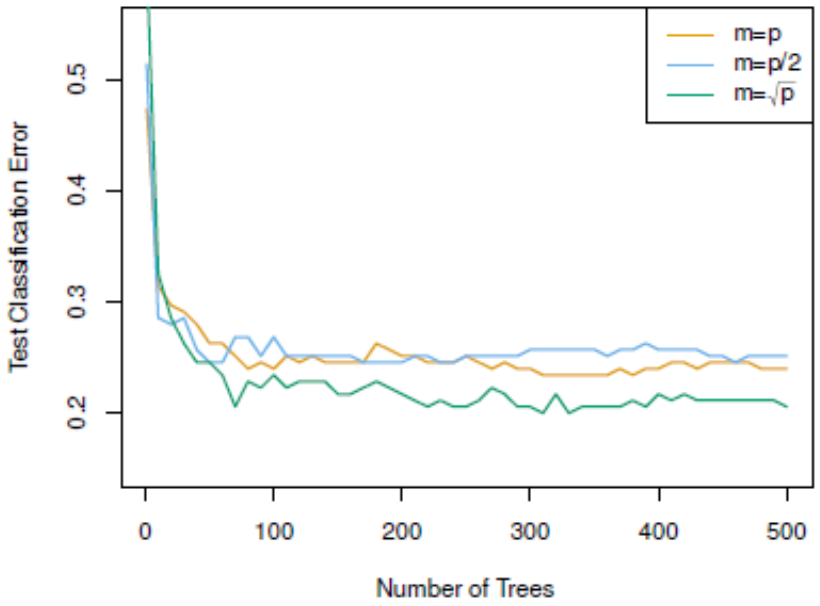
    모든 tree에 대해 각 예측변수의 split으로 인해 SSE가 감소한 정도를 측정하여 평균을 취한다. (분류의 경우 node impurity)

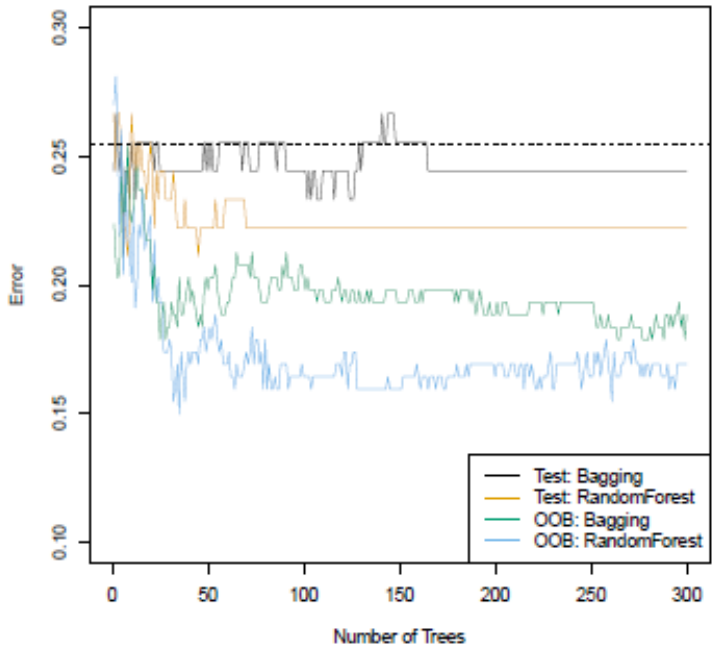


source: https://www.statlearning.com/resources-second-edition

Random Forests

- Bootstrapped 데이터셋은 서로 correlation이 높기 때문에 Bagging을 해도 독립된 모델들을 얻지 못한다.
  → variance 감소 효과가 적다!
- Random Forest는 tree들을 decorrelate 해주고자 split을 진행할 때마다 전체 p개의 예측변수 중 랜덤하게 m개의 변수를 뽑고, 이들만 고려하여 split을 진행한다. $m \approx \sqrt{p}$를 주로 사용한다.





source: https://www.statlearning.com/resources-second-edition

Boosting

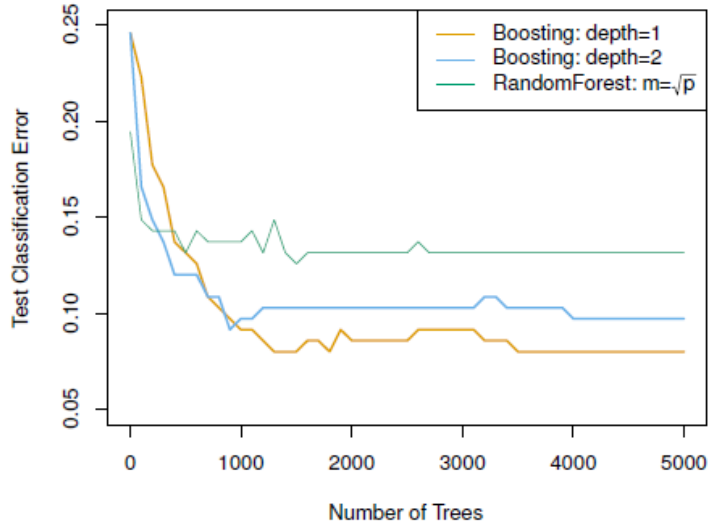
- Boosting은 각 트리들의 학습이 독립적으로 이루어지는 Bagging과 다르게, 이전까지 학습한 트리들의 정보를 이용하여 순차적으로 트리 학습이 이루어진다.
- 전체 데이터셋을 사용하되 잘못 예측한 데이터에 집중하여 반복 학습을 시킨다.

1. 최종 예측값 $\hat{f}(x) = 0$ 으로, 트레이닝셋의 모든 i에 대해 잔차(residual)을 $r_i = y_i$ 로 초기화 한다.
2. $b = 1,2, \ldots, B$에 대해 다음을 반복한다.
    1. d개만큼의 split (d+1 leaf nodes)를 가지고 있는 tree $\hat{f}^b$ 를 training data (X, r)에 피팅시킨다.
    2. 새로운 tree $\hat{f}^b$ 를 λ라는 비율 만큼 반영해준다. 즉, $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$.
    3. 잔차를 업데이트한다. 즉, $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$
3. 다음과 같은 최종 boosted model을 반환한다.

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Boosting

- 3개의 hyperparameter
  1. Tree의 개수 B: B가 크면 overfit 될 수 있다. cross validation을 통해 B를 선택한다.
  2. Shirinkage parameter λ: 학습 속도를 조절, 0.01~0.001 사용, 작아지면 B가 커져야 한다.
  3. split 횟수 d: boosted tree의 complexity를 조절, boosting을 통해 bias를 줄이므로 d가 클 필요가 없다.



- Classification tree를 위한 boosting 방법은 잔차를 계산하는 방식이 달라지는데, 복잡하기 때문에 생략.

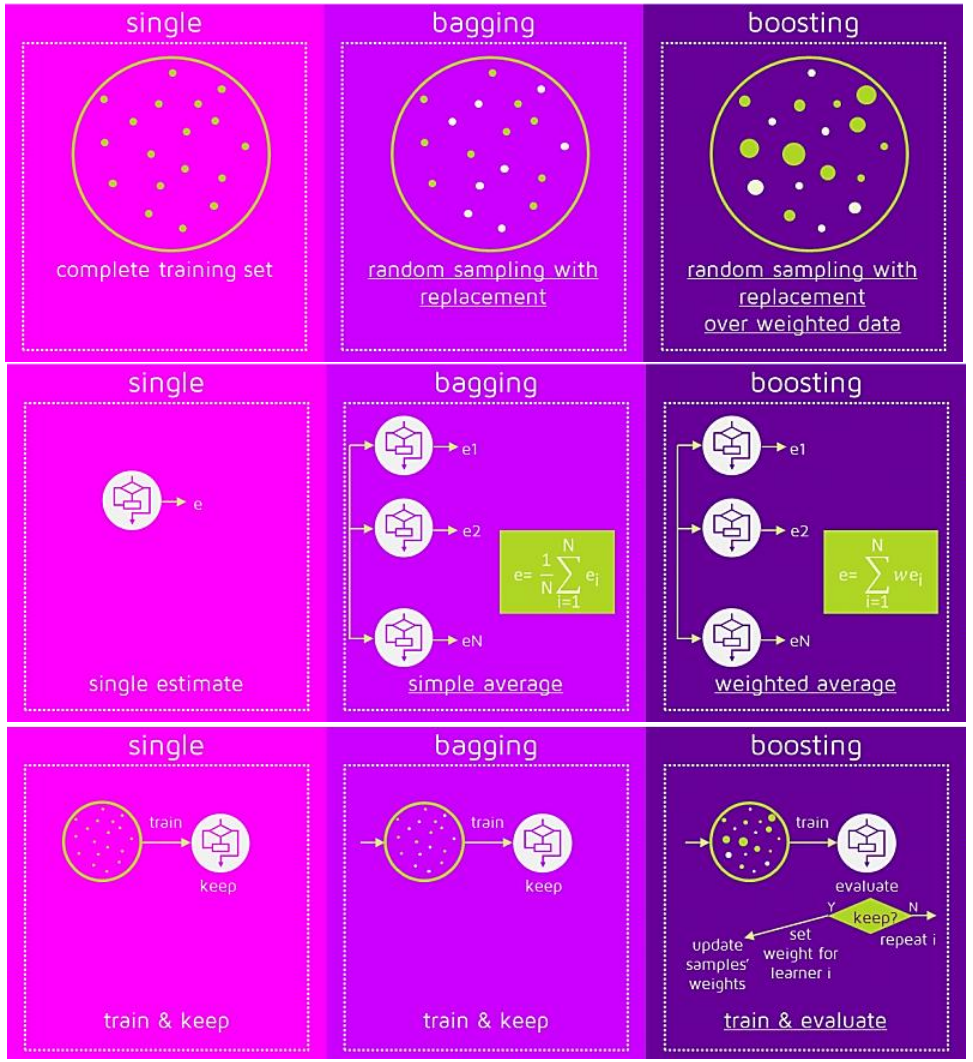
  (참고자료: Elements of Statistical Learning, Ch. 10)

source: https://www.statlearning.com/resources-second-edition

LIKE LION

Bagging vs. Boosting

비교	Bagging	Boosting
특징	병렬 앙상블 (각 모델이 독립적)	순차적 앙상블(이전 모델의 오류를 고려)
목적	Variance 감소	Bias 감소
적합한 상황	High variance, low bias 모델	Low variance, high bias 모델
대표 알고리즘	Random Forest	Gradient Boosting, AdaBoost
Sampling	Random Sampling	Weights on errors



https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

Summary

- Decision tree는 classification, regression에 모두 사용가능한 간단하고 설명력 있는 모델.
- 그러나 정확도 측면에서 성능이 떨어진다.
- Bagging, random forests, boosting과 같은 방법들로 여러 개의 tree를 학습시키고 이들의 결과를 결합함으로써 정확도를 높일 수 있다.
- Random forests와 tree boosting은 특히 성능이 뛰어난 supervised learning 알고리즘에 속한다.
- 설명력은 떨어지지만 variance importance measure를 활용해 중요한 예측변수를 알 수 있다.

LIKE LION

{

5. (실습) Logistic Regression

6. (실습) Decision Tree

}