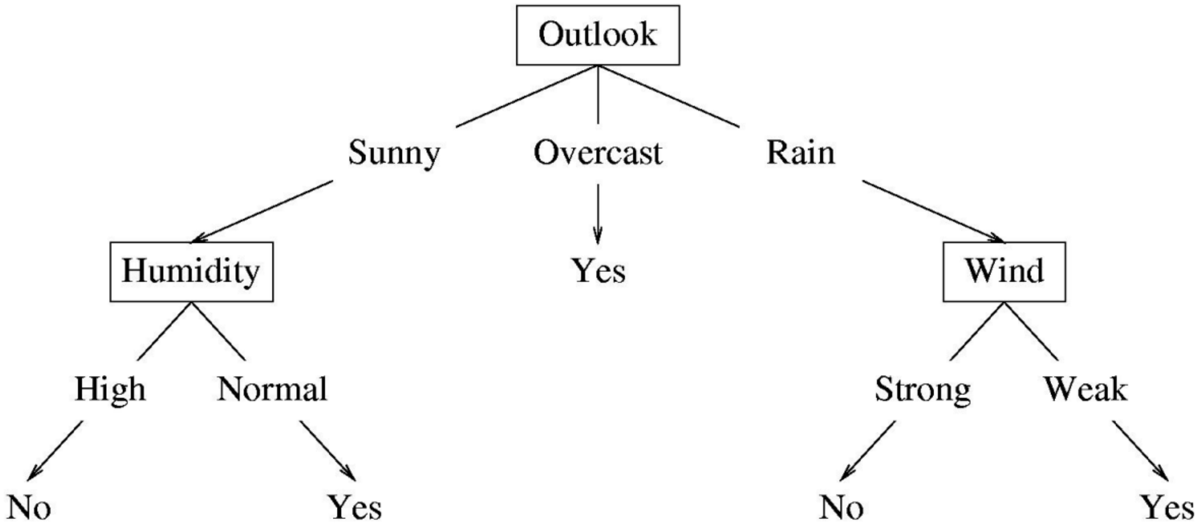
AI 기본 교육

2021. 10. 01

곽대훈

Decision Tree

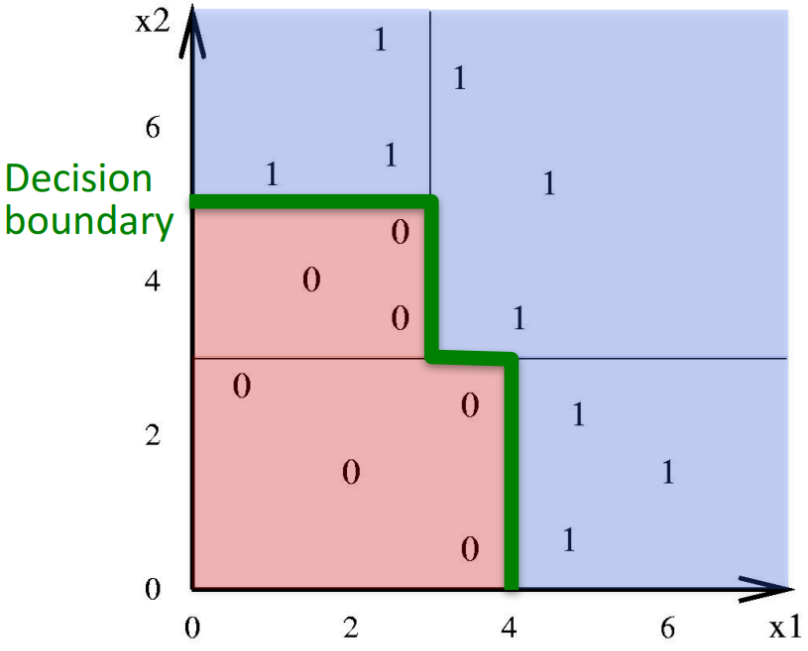
- A possible decision tree for the data:

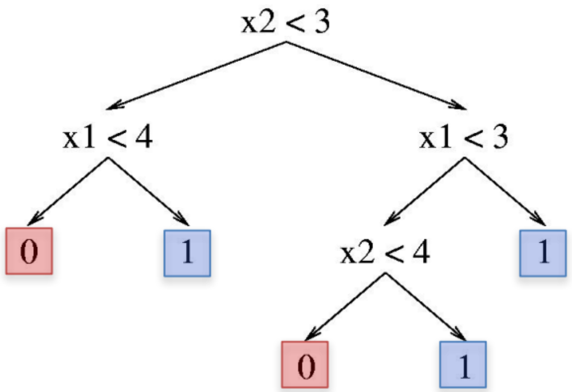


- Each internal node: test one attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: predict Y

Decision Tree

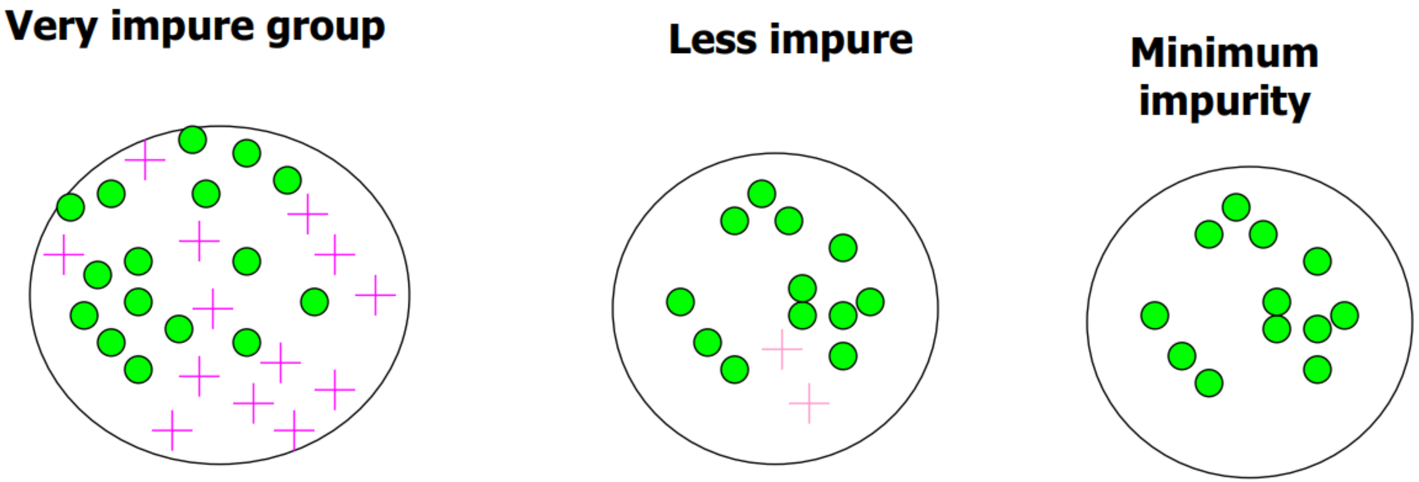
- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label
  - or a probability distribution over labels





Decision Tree

Impurity

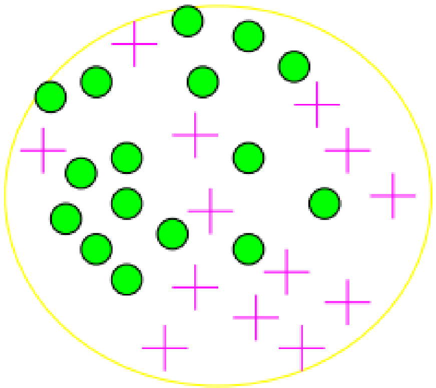


Decision Tree

Entropy: a common way to measure impurity

- Entropy = $\sum_i -p_i \log_2 p_i$

  p_i is the probability of class i
  Compute it as the proportion of class i in the set.

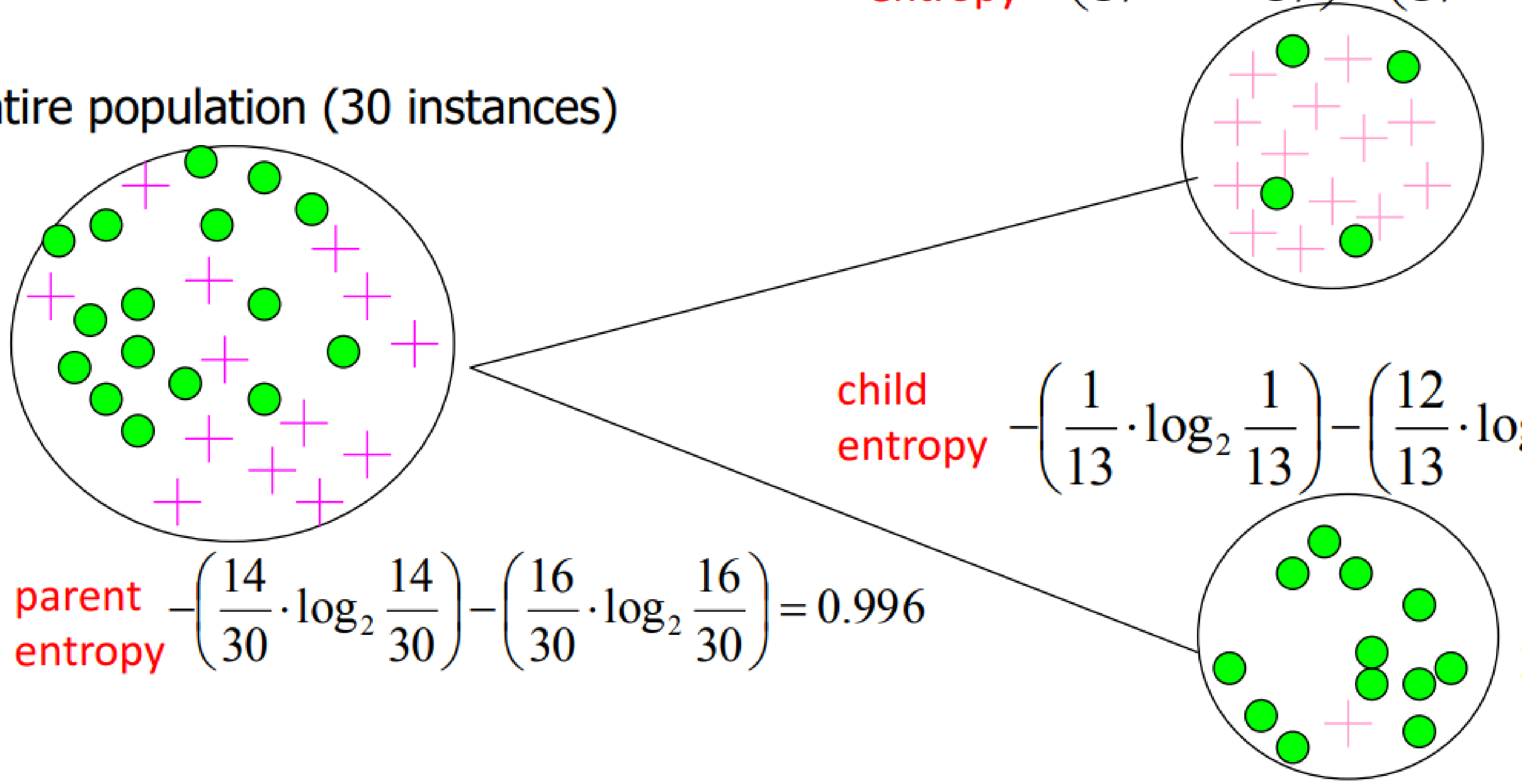


- Entropy comes from information theory. The higher the entropy the more the information content.

Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$



child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Ensemble Philosophy

- Build many models and combine them
- Only through averaging do we get at the truth!
- It's too hard (*impossible*?) to build a single model that works best
- Two types of approaches:
  - Models that don't use randomness
  - Models that incorporate randomness

Ensemble Approaches

- Bagging
  - **B**ootstrap aggregating
- Boosting
- Random Forests
  - Bagging reborn

Bagging

- Main Assumption:
  - Combining many unstable predictors to produce a ensemble (stable) predictor.
  - Unstable Predictor: small changes in training data produce large changes in the model.
    - e.g. Neural Nets, trees
    - Stable: SVM (sometimes), Nearest Neighbor.
- Hypothesis Space
  - Variable size (nonparametric):
    - Can model any function if you use an appropriate predictor (e.g. trees)

The Bagging Algorithm

Given data: $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

For $m = 1 : M$

- Obtain bootstrap sample D_m from the training data D
- Build a model $G_m(\mathbf{x})$ from bootstrap data D_m

The Bagging Model

- Regression

$$\hat{y} = \frac{1}{M}\sum_{m=1}^{M} G_m(\mathbf{x})$$

- Classification:
  - Vote over classifier outputs $G_1(\mathbf{x}),\ldots,G_M(\mathbf{x})$

Bagging Details

- Bootstrap sample of N instances is obtained by drawing N examples at random, with replacement.
- On average each bootstrap sample has 63% of instances
  - Encourages predictors to have uncorrelated errors
    - This is why it works

Bagging Details 2

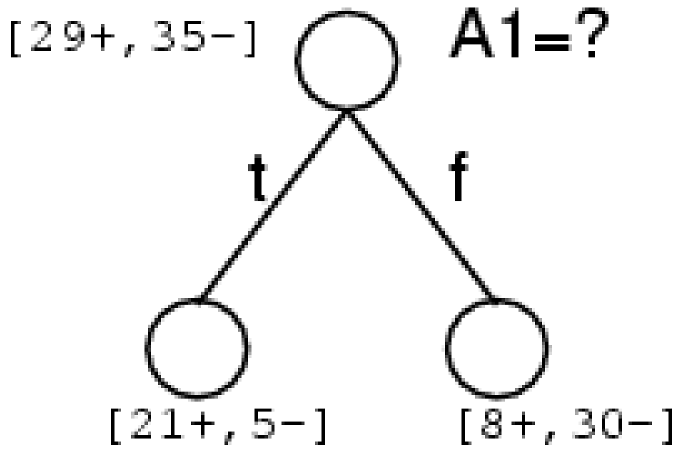
- Usually set $M = \sim 30$
  - Or use validation data to pick M
- The models $G_m(\mathbf{x})$ need to be unstable
  - Usually full length (or slightly pruned) decision trees.

Boosting

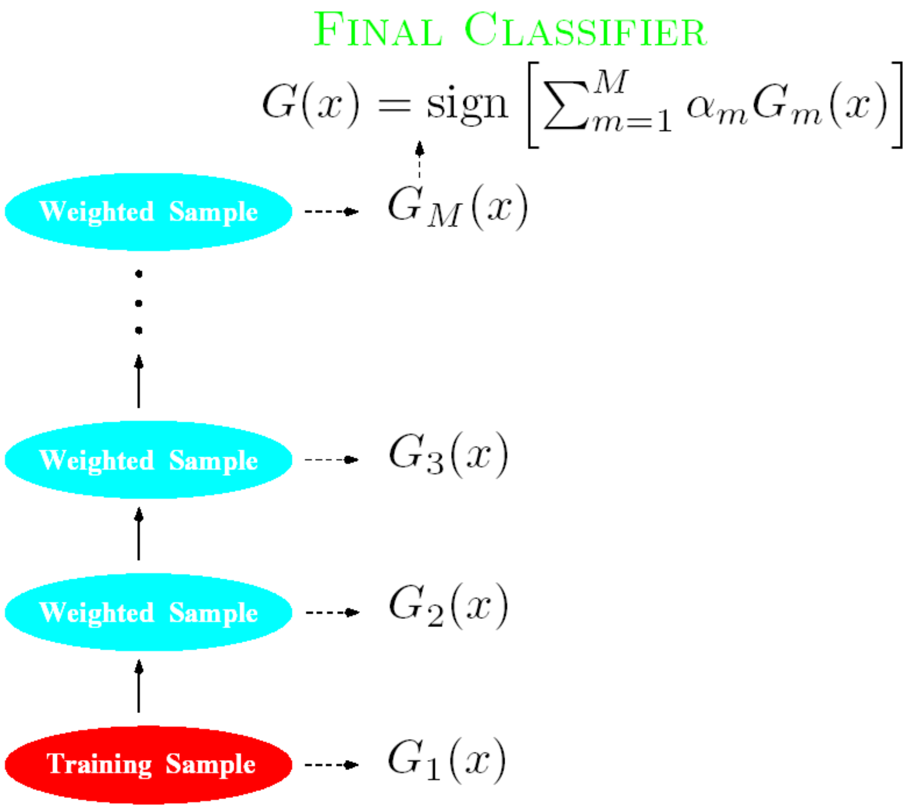
- Main Assumption:
  - Combining many weak predictors (e.g. tree stumps or 1-R predictors) to produce an ensemble predictor
  - The weak predictors or classifiers need to be **stable**
- Hypothesis Space
  - Variable size (nonparametric):
    - Can model any function if you use an appropriate predictor (e.g. trees)

Commonly Used Weak Predictor (or classifier)

A Decision Tree Stump (1-R)



Boosting



Each classifier $G_m(\mathbf{x})$ is trained from a weighted Sample of the training Data

Boosting (Continued)

- Each predictor is created by using a biased sample of the training data
  - Instances (training examples) with high error are weighted higher than those with lower error
- Difficult instances get more attention
  - This is the motivation behind boosting

Background Notation

- The
- The

- The $I(s)$ function is defined as:

$$I(s) = \begin{cases} 1 \text{ if } s \text{ is true} \\ 0 \quad \text{otherwise} \end{cases}$$

- The $\log(x)$ function is the natural logarithm

The AdaBoost Algorithm

(Freund and Schapire, 1996)

Given data: $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$

1. Initialize weights $w_i = 1/N, i = 1, \dots, N$
2. For $m = 1 : M$
   a) Fit classifier $G_m(\mathbf{x}) \in \{-1, 1\}$ to data using weights w_i
   b) Compute
   $$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(\mathbf{x}_i))}{\sum_{i=1}^{N} w_i}$$
   c) Compute $\alpha_m = \log((1 - err_m)/err_m)$
   d) Set $w_i \leftarrow w_i \exp[\alpha_m I(y_i \neq G_m(\mathbf{x}_i))], \quad i = 1, \dots, N$

Gradient Boosting Model

- The prediction at round t is $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$

This is what we need to decide in round t

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant
\end{aligned}
$$

Goal: find f_t to minimize this

- Consider square loss

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} \left(y_i - (\hat{y}_i^{(t-1)} + f_t(x_i))\right)^2 + \Omega(f_t) + const \\
&= \sum_{i=1}^{n} \left[2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2\right] + \Omega(f_t) + const
\end{aligned}
$$

This is usually called residual from previous round

Gradient Boosting Model

- XGBOOST
- CATBOOST
- LIGHT GBM
- …