Machine Learning

Decision Trees

Mayson Ma

Courant Institute of Mathematical Sciences - New York University

June 2021

Table of Contents

Introduction

Definition

Tree with two node type:

- internal nodes that test feature values (usually just one feature) and branch accordingly
- leaf nodes specify $h(x)$

Features

- Nonlinear method for classification and regression
- Cut X-space into rectangular cells
- Works well with both quantitative and categorical features
- Interpretable result (inference)
- Decision boundary can be arbitrarily complicated

Heuristic

Greedy, top-down learning heuristic

Let $S \subseteq \{1, ..., n\}$ be the set of sample point indices
Top-level call: $S = \{1, ..., n\}$

GrowTree(S)
if y_i=c for all $i \in S$ and some c **then**
  return new Leaf(c)
else
  Choose best splitting feature j and splitting value β
  $S_l = \{i : X_{ij} < \beta\}$
  $S_r = \{i : X_{ij} \geq \beta\}$
  return new Node(j, β, GrowTree(S_l), GrowTree(S_r))
end if

Heuristic

How to choose best split?

- Try all splits
- For a set S, let $J(S)$ be the **cost** of S
- Choose the split that minimizes $J(S_l) + J(S_r)$; or the weighted average

$$\frac{|S_l|J(S_l) + |S_r|J(S_r)}{|S_l| + |S_r|}$$

How to choose $J(S)$

- Measure the entorpy

Entropy and Information

Definition

Let Y be a discrete random variable, and $\Pr(Y = c) = p_c$.
The **surprise** of Y being class c is $\log_2 \frac{1}{p_c} = -\log_2 p_c$.
Note that the event with probability $0/1$ gives infinite/zero surprise respectively!

The **entropy** of Y is defined as $H(Y) = -\sum_c p_c \log_2 p_c$

How to choose $J(S)$

The **entropy** of an index set S is the average surprise $H(S) = -\sum_c p_c \log_2 p_c$, where $p_c = \frac{|\{i \in S : y_i = c\}|}{|S|}$

- If all points in S belong to same class ? $H(S) = 0$
- Half class C, half class D? $H(S) = 1$
- n points, all in different classes? $H(S) = \log_2 n$

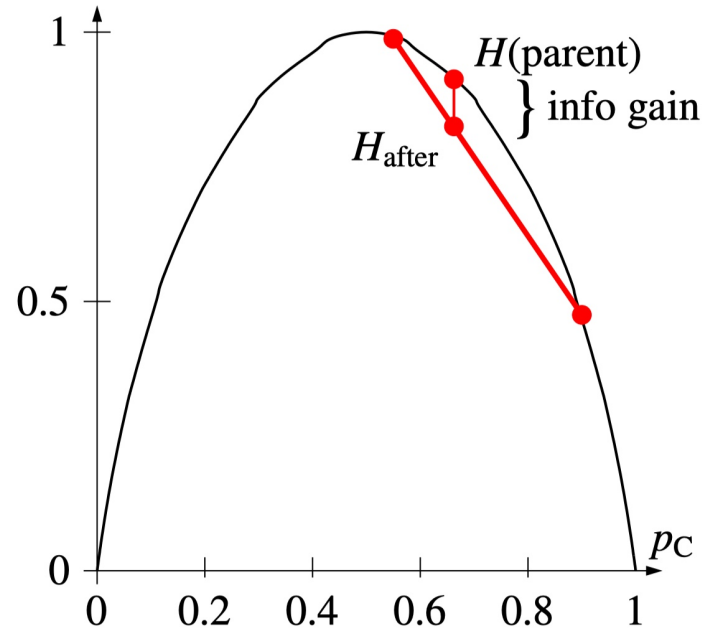
Entropy and Information

Choose split that maximizes **information gain**

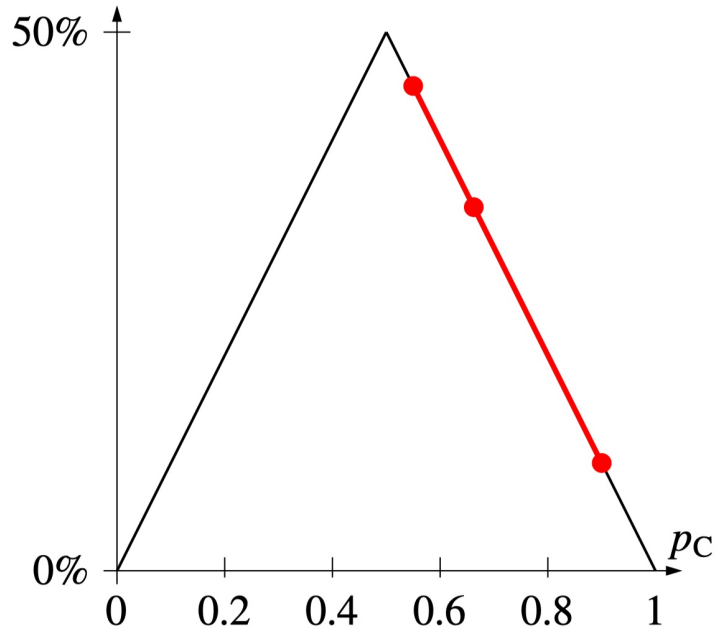
$H_{after} = \frac{|S_l|H(S_l)+|S_r|H(S_r)}{|S_l|+|S_r|}$; Info gain: $H(S) - H_{after}$

Info gain always positive **except** one child is empty, or for all class C, $\Pr(y_i = C | i \in S_l) = \Pr(y_i = C | i \in S_r)$

$H(p_C)$ entropy: strictly concave



% misclassified: concave, not strict



Algorithm

More on choosing a split

- For binary feature X_i: children are $X_i = 0$ and $X_i = 1$
- If X_i has 3+ discrete values: depends on application
- If X_i is quantitative: sort X_i values in S; try splitting between each pair of **unequal** consecutive values
  Note: as you can sorted list from left to right, you can update the entropy in $O(1)$ time per point!

Multivariate splits

- For non-axis-aligned, splits with other classification algorithms or by generating them randomly
- May gain better classifier at cost of wore interpretability or speed
- Can limit number of features per split: forward step-wise selection / Lasso

Stopping Early

Why?

- Limit tree depth (for speed)
- Limit tree size (big data sets)
- Complete tree may overfit
- Given noise or overlapping distributions, purity of leaves is counterproductive

How? Select stopping conditions:

- Next split does not reduce entropy / error enough. (Dangerous, consider XOR)
- Most of nodes' points (e.g., $> 95\%$) have same class
- Node contains few sample points (e.g., < 10)
- Cell's edges are all tiny
- Depth too great
- Use validation to decide

Algorithm

Leaves with multiple points return

- a majority vote or class posterior probabilities
- an average (regression)

Decision Tree Regression

- Create a piecewise constant regression function
- Cost $J(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y})^2$

Pruning

- Grow tree too large; greedily remove each split whose removal improves validation performance.
- More reliable than stopping early.

Random Forests

Observation

- Random sampling is not random enough
- One really strong predictor $\rightarrow$ same feature split at top of every tree

Idea

- At each tree node, take random sample of m features (out of d). Choose best split from m features.
- Different random sample for each tree node.

Notes

- In practice, $m = \sqrt{d}$ works well for classification; $m \approx d/3$ for regression
- Smaller $m \rightarrow$ more randomness, less correlation, more bias
- Disadvantage: loses interpretability