Decision Trees

Andrew Nobel

April, 2020

Histogram Classification Rule

General setting

- Data set $(x_1, y_1), \dots, (x_n, y_n) \in \mathcal{X} \times \{\pm 1\}$
- Partition $\pi = \{A_1, \dots, A_m\}$ of feature space $\mathcal{X}$
- Membership function $\pi[x] =$ cell A_j of π containing x

Histogram rule ϕ_π assigns every point in cell A_j to same class, typically

$$\phi_\pi(x) = \text{ majority vote among } \{y_i : \pi[x_i] = \pi[x]\}$$

which minimizes total number of misclassifications on the data

Good partition: Small number of regular cells, each of which contains points from a single class

Binary Trees

Definition: Binary tree T is a (directed, acyclic) graph characterized by

1. A distinguished node t_0 called the **root** with no parent
2. Every other node $t \in T$ has one parent and zero or two children
   - Nodes with two children are called **internal**, denoted by T^o
   - Nodes with no children are called **leaves**, denoted by ∂T

Note: Tree usually drawn upside-down, with root node at the top

Decision Trees

Simplest case: Feature space $\mathcal{X} = \mathbb{R}^p$

- Binary tree T: root t_0, interior nodes T^o, leaves ∂T
- For each $t \in T^o$, variable $j(t) \in \{1, \ldots, p\}$ and threshold $\tau(t) \in \mathbb{R}$
- Class assignment $c(t) \in \{\pm 1\}$ for each leaf $t \in \partial T$

Note: Each $t \in T$ corresponds to region $R(t) \subseteq \mathbb{R}^p$. Recursively defined

- For root, $R(t_0) = \mathbb{R}^p$.
- For $t \in T^o$ region $R(t)$ split between L/R children using $j(t)$ and $\tau(t)$

$$R(t_l) = R(t) \cap \{x : x_{j(t)} \leq \tau(t)\} \text{ and } R(t_r) = R(t) \cap \{x : x_{j(t)} > \tau(t)\}$$

Decision Trees, cont.

Easy to see: Regions $\pi_T = \{R(t) : t \in \partial T\}$ partition the feature space

Decision tree: Assigns class $c(t)$ to every $x \in R(t)$

$$\phi_T(x) \; = \sum_{t \in \partial T} c(t) \, \mathbb{I}(x \in R(t))$$

Qu: How to obtain a decision tree T from data?

- Grow a large tree T_0 in a greedy fashion
- Prune T_0 balancing performance and size

Impurity Measures for Regions

Given: Data $(x_1, y_1), \ldots, (x_n, y_n)$ and region $R \subseteq \mathcal{X}$ define

- $|R| =$ number of $x_i \in R$
- $\hat{p} =$ fraction of $x_i \in R$ with $y_i = +1$

Impurity Measures

- Misclassification rate $M(R) = \min(\hat{p}, 1 - \hat{p})$
- Gini Index $G(R) = \hat{p}\,(1 - \hat{p})$
- Entropy $H(R) = -\hat{p} \log \hat{p} - (1 - \hat{p}) \log(1 - \hat{p})$

Idea: IMs quantify extent to which R contains points from one class. Small when $\hat{p}$ close to 0 or 1.

Impurity Reduction from Region Splitting

Definition: If region R split into R_l and R_r the *change* in misclassification is

$$\Delta_M \;=\; M(R) - \left[\frac{|R_l|}{|R|} M(R_l) + \frac{|R_r|}{|R|} M(R_r)\right]$$

Changes Δ_G and Δ_H for Gini and Entropy defined similarly

Fact: Changes $\Delta_M, \Delta_G, \Delta_H \geq 0$. Larger Δ means better split

Idea: Grow initial tree T_0 by recursively improving terminal regions using best single coordinate splits. Stop splitting when terminal regions have few points, are sufficiently pure.

Tree Growing from Data

```
Initialize: Let $T := \{t_0\}$ with $R(t_0) = \mathbb{R}^p$

Iterate: For each leaf $t \in \partial T$ with $|R(t)| \geq n_0$ do

  1. For each $j \in \{1, \ldots, p\}$ and each $\tau \in \mathbb{R}$ find $\Delta_M$ for split

       $R(t) \cap \{x : x_j \leq \tau\}$ and $R(t) \cap \{x : x_j > \tau\}$

  2. Identify optimal variable $j(t)$ and threshold $\tau(t)$

  3. If optimal $\Delta_M \leq \Delta_0$ stop. Otherwise add leaves $t_l$ and $t_r$ to node $t$

       ▸ assign $R(t_l) = R(t) \cap \{x : x_{j(t)} \leq \tau(t)\}$

       ▸ assign $R(t_r) = R(t) \cap \{x : x_{j(t)} > \tau(t)\}$

Output: Baseline tree $T_0$
```

Cost-Complexity Pruning

Note: Let $T \leq T_0$ be a binary subtree of T_0 with same root, possibly $T = T_0$

- Leaf regions $\{R(t) : t \in \partial T\}$ of T partition the feature space $\mathcal{X}$
- Using majority voting in leaf regions, T determines a decision tree and a corresponding histogram classification rule ϕ_T
- Let $R_n(T) =$ empirical risk of ϕ_T and let $|T| =$ number of nodes in T

Definition: For each $\lambda \geq 0$ define optimal cost-complexity subtree

$$T_\lambda \; = \; \underset{T \leq T_0}{\operatorname{argmin}} \{R_n(T) + \lambda |T|\}$$

Tradeoff performance of tree vs. its size

Cost-Complexity Pruning

Recall: Optimal cost-complexity tree $T_\lambda = \text{argmin}_{T \leq T_0} \{R_n(T) + \lambda|T|\}$

- Tradeoff: If $T_1 \leq T_2$ then $|T_1| \leq |T_2|$ while $R_n(T_2) \leq R_n(T_1)$
- For $\lambda = 0$, $T_\lambda =$ full tree T_0; for λ large, $T_\lambda = \{t_0\}$

Fact: If $\lambda_1 \leq \lambda_2$ then $T_{\lambda_2} \leq T_{\lambda_1}$

In practice: Choose value of λ using cross validation