Tree Based Methods

Greg Strabel

July 1, 2020

1 Preliminaries

Definition: A *directed graph*, G, is a tuple (V, A), consisting of a set V, called vertices, and a set $A \subseteq \{(v,w) \,|v, w \in V, v \neq w\}$, called arrows, consisting of order pairs of distinct elements of V.

Definition: Given a directed graph $G = (V, A)$, a *directed path* from v to w is a tuple of distinct arrows $\{(v_i, w_i)\}_{i=1}^{P} \in A$ such that $v_1 = v$, $w_P = w$ and $\forall i < P,\ w_i = v_{i+1}$

Definition: A *binary tree* is a directed graph $G = (V, A)$ such that:

1. There is a unique $r \in V$, called *root*, such that $\forall v \in V : (v, r) \notin A$.
2. For each vertex $v \in V$, there is a unique directed path from root to v.
3. For each vertex $v \in V,\ |\{w \in V : (v,w) \in A\}| \in \{0, 2\}$.

If $(v,w) \in A$, then we say that v is a *parent* of w and w is a *child* of v. If $v \in V$ has children, we call it an *internal vertex* or *internal node*. If $v \in V$ does not have any children, we call it a $leaf$.

Given a binary tree $G_0 = (V_0, A_0)$ and a leaf $l \in V$, we can construct another binary tree that is an extension of G_0, by adding two new vertices v and w as children of l:

$$G_1 \coloneqq (V_1, A_1) \coloneqq (V_0 \cup \{v, w\},\ A \cup \{(l,v)\,,(l,w)\}) \tag{1}$$

Starting with a tree consisting of just a root, $G_0 = (\{r\}, \varnothing)$, we can apply the extension procedure above recursively to build arbitrarily large binary trees.

2 Basics of Recursive Binary Partitions

Assume that we have a dataset $\mathbb{D} \coloneqq \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$ where $\mathbf{x}_i$ is a n-tuple, $\mathbf{x}_i = (x_{i1}, ..., x_{in}) \in \mathbf{X} = X_1 \times ... \times X_n$, and $\mathbf{y}_i \in \mathbf{Y}$. Given a loss function L, we desire to find a function

$f : \mathbf{X} \to \mathbf{Y}$ of the form

$$f(\mathbf{x}) \coloneqq \sum_{j \in J} c_j \mathbf{1}\{\mathbf{x} \in R_j\} \tag{2}$$

where $\{R_j\}_{j=1}^{J}$ is a partition of $\mathbf{X}$ that approximately minimizes $\sum_{i=1}^{N} L(\mathbf{y}_i, f(\mathbf{x}_i))$ This approach is generally computationally infeasible. Instead, we search for a partition of $\mathbf{X}$ in a recursive, greedy manner.
Let

$$\mathbb{D}_R \coloneqq \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{D} : \mathbf{x}_i \in R\}, \tag{3}$$

$$\mathfrak{D} \coloneqq \{(R, \mathbb{D}_R) : R \subseteq \mathbf{X}\} \tag{4}$$

and, for $\mathbf{D} = (R, \mathbb{D}_R) \in \mathfrak{D}$,

$$2^{\mathbf{D}} \coloneqq \{((R_l, \mathbb{D}_{R_l}), (R_r, \mathbb{D}_{R_r})) \in \mathfrak{D}^2 : R_r \subseteq R,\ R_l = R \setminus R_r\} \tag{5}$$

For $\mathbf{D} = (R, \mathbb{D}_R) \in \mathfrak{D}$ the loss improvement from the refined binary partition $(\mathbf{D}_l, \mathbf{D}_r) = ((R_l, \mathbb{D}_{R_l}), (R_r, \mathbb{D}_{R_r})) \in 2^{\mathbf{D}}$ is

$$\text{lossImp}(\mathbf{D}, \mathbf{D}_l, \mathbf{D}_r) = \min_{c} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{D}_R} L(\mathbf{y}_i, c) - \min_{c_l, c_r} \left[\sum_{\mathbb{D}_{R_l}} L(\mathbf{y}_i, c_l) + \sum_{\mathbb{D}_{R_r}} L(\mathbf{y}_i, c_r) \right] \tag{6}$$

We require an algorithm which, for each $\mathbf{D} \in \mathfrak{D}$, searches over a representative subset of $2^{\mathbf{D}}$ for a binary partition of $\mathbf{D}$ that approximately maximizes lossImp. One such algorithm is the Basic Binary Split of *Algorithm* (1) on the following page. Starting with $\mathbf{D} = (\mathbf{X}, \mathbb{D})$, we apply the Basic Binary Split recursively to each leaf until some stopping criteria is reached. Common stopping criteria include a minimum number of observations in a leaf or a minimum loss improvement from splitting.

Unfortunately, the Basic Binary Split quickly becomes computationally infeasible when datasets contain categorical variables with even a small number of distinct values. For a categorical variable with K levels, there are $2^{K-1} - 1$ distinct partitions to consider when splitting. For instance, if $K = 30$, there are over 500 million possible partitions. We explore several alternative methods for splitting on categorical variables in the next section.

3 Splitting on a Categorical Variable

3.1 All possible partitions

3.2 Binary Target Variable

When $\mathbf{Y}$ is binary, there is a shortcut for finding the best partition of a leaf $(R, \mathbb{D}_R) \in \mathfrak{D}$ on a categorical variable X_j. Without loss of generality, assume $\mathbf{Y} = \{0, 1\}$ and for each

Algorithm 1: Basic Binary Split

Input: Subset $\mathbf{D} = (R, \mathbb{D}_R) \in \mathfrak{D}$, loss function L
Output: Partition $(\mathbf{D}_l, \mathbf{D}_r) \in 2^{\mathbf{D}}$

for $j \in \{1, ..., n\}$ **do**
  if X_j *is categorical* **then**
    $S = \textbf{enum}\left(\{(\mathbf{D}_l, \mathbf{D}_r) \in 2^{\mathbf{D}} : (\exists A \subset X_j : R_l = A \cap R)\}\right)$
  else if X_j *is ordinal* **then**
    $\mathrm{V} \leftarrow \textbf{cutpoints}\left(\{\mathbf{x}_{ij}\}\right)$ `// e.g. midpoints of adjacent order statistics`
    $S = \textbf{enum}\left(\{(\mathbf{D}_l, \mathbf{D}_r) \in 2^{\mathbf{D}} : R_l = R \cap \{x \leq d : d \in V\}\}\right)$
  $v = \overrightarrow{0} \in \mathbb{R}^{|S|}$
  for $t \in \{1, ..., |S|\}$ **do**
    $(\mathbf{D}_r, \mathbf{D}_l) = S\left(t\right)$
    $v_t = \mathrm{lossImp}\left(\mathbf{D}, \mathbf{D}_l, \mathbf{D}_r\right)$
  $b_j = \mathrm{argmax}_{t \in \{1, \ldots, |S|\}}\, v_t$
  $\mathbf{D}^{(j)} = S\left(b_j\right)$
$j^* = \mathrm{argmax}_{j \in \{1, \ldots, n\}} \{b_j\}$
return $\mathbf{D}^{(j^*)}$

$A \subseteq X_j$ define

$$N_A = \sum_{(\mathbf{x}_i, y_i) \in \mathbb{D}_R} \mathbf{1}\{\mathbf{x}_{ij} \in A\} \tag{7}$$

and

$$\overline{y}_A = \frac{1}{N_A} \sum_{(\mathbf{x}_i, y_i) \in \mathbb{D}_R} y_i \mathbf{1}\{\mathbf{x}_{ij} \in A\} \tag{8}$$

Then the optimal partition (R_l, R_r) of R is of the form $R_l = R \cap A$ and $R_r = R \cap B$ where $A = \{z \in X_j : \overline{y}_{\{z\}} < p\}$ for some $p \in (0, 1)$, $B = X_j \setminus A$ and the corresponding predictions for R_l and R_r are p_A and p_B where $p_A < p_B$. To see why this is the case, suppose the optimal partition of R on X_j is (R_l, R_r) where $R_l = R \cap A$ for some $A \subset X_j$, $B = X_j \setminus A$ and $p_A < p_B$. Suppose that $\alpha \in A$ and $\beta \in B$. Since the partition is optimal

$$\overline{y}_{\{\alpha\}} L\left(1, p_A\right) + \left(1 - \overline{y}_{\{\alpha\}}\right) L\left(0, p_A\right) < \overline{y}_{\{\alpha\}} L\left(1, p_B\right) + \left(1 - \overline{y}_{\{\alpha\}}\right) L\left(0, p_B\right) \tag{9}$$

and

$$\overline{y}_{\{\beta\}} L\left(1, p_B\right) + \left(1 - \overline{y}_{\{\beta\}}\right) L\left(0, p_B\right) < \overline{y}_{\{\beta\}} L\left(1, p_A\right) + \left(1 - \overline{y}_{\{\beta\}}\right) L\left(0, p_A\right) \tag{10}$$

Rearranging and combining these equations and using the assumptions that $L(1,p)$ is decreasing in p and $L(0,p)$ is increasing in p, we have

$$\begin{aligned}\overline{y}_{\{\alpha\}}\left[L(1,p_A)-L(1,p_B)\right] &< \left(1-\overline{y}_{\{\alpha\}}\right)\left[L(0,p_B)-L(0,p_A)\right] \\ &< \left(1-\overline{y}_{\{\beta\}}\right)\left[L(0,p_B)-L(0,p_A)\right] \\ &< \overline{y}_{\{\beta\}}\left[L(1,p_A)-L(1,p_B)\right]\end{aligned} \tag{11}$$

which implies that $\overline{y}_{\{\alpha\}} < \overline{y}_{\{\beta\}}$. The result follows immediately from the fact that α and β were arbitrary elements of A and B, respectively.

Therefore, finding the optimal split of a categorical variable X_j in a binary classification problem reduces to a problem of order $O(|X_j|)$.

3.3 Numeric Target Variable and Squared Loss

When $\mathbf{Y}$ is numeric and $L(y,f)=(f-y)^2$, there is a shortcut for finding the best partition of a leaf $(R,\mathbb{D}_R)\in\mathfrak{D}$ on a categorical variable X_j analogous to that for a binary response. For each $A\subseteq X_j$ define

$$N_A = \sum_{(\mathbf{x}_i,y_i)\in\mathbb{D}_R} \mathbf{1}\{\mathbf{x}_{ij}\in A\} \tag{12}$$

and

$$\overline{y}_A = \frac{1}{N_A}\sum_{(\mathbf{x}_i,y_i)\in\mathbb{D}_R} y_i\mathbf{1}\{\mathbf{x}_{ij}\in A\} \tag{13}$$

Then the optimal partition (R_l,R_r) of R is of the form $R_l=R\cap A$ and $R_r=R\setminus A$ where $A=\{z\in X_j : \overline{y}_{\{z\}}<p\}$ for some $p\in\mathbb{R}$ and the corresponding predictions for R_l and R_r are $\overline{y}_A$ and $\overline{y}_{X_j\setminus A}$. To see why this is the case, suppose the optimal partition of R on X_j is (R_l,R_r) where $R_l=R\cap A$ for some $A\subset X_j$, $B=X_j\setminus A$ and $\overline{y}_A<\overline{y}_B$. Since the slit is optimal and we are using squared error loss, the predictions on R_l and R_r are $\overline{y}_A$ and $\overline{y}_B$, respectively. Suppose that $\alpha\in A$ and $\beta\in B$. Optimality implies that

$$\begin{aligned}\sum_{(\mathbf{x}_i,y_i)\in\mathbb{D}_R:\mathbf{x}_{ij}=\alpha}\left[\left(y_i-\overline{y}_{\{\alpha\}}\right)^2+\left(\overline{y}_{\{\alpha\}}-\overline{y}_A\right)^2\right] &= \sum_{(\mathbf{x}_i,y_i)\in\mathbb{D}_R:\mathbf{x}_{ij}=\alpha}(y_i-\overline{y}_A)^2 \\ &< \sum_{(\mathbf{x}_i,y_i)\in\mathbb{D}_R:\mathbf{x}_{ij}=\alpha}(y_i-\overline{y}_B) \\ &= \sum_{(\mathbf{x}_i,y_i)\in\mathbb{D}_R:\mathbf{x}_{ij}=\alpha}\left[\left(y_i-\overline{y}_{\{\alpha\}}\right)^2+\left(\overline{y}_{\{\alpha\}}-\overline{y}_B\right)^2\right]\end{aligned} \tag{14}$$

It follows that

$$\left(\overline{y}_{\{\alpha\}}-\overline{y}_A\right)^2 < \left(\overline{y}_{\{\alpha\}}-\overline{y}_B\right)^2 \tag{15}$$

A similar argument shows that

$$\left(\overline{y}_{\{\beta\}} - \overline{y}_B\right)^2 < \left(\overline{y}_{\{\beta\}} - \overline{y}_A\right)^2 \tag{16}$$

It follows that

$$\overline{y}_{\{\alpha\}} < \frac{1}{2}\left(\overline{y}_A + \overline{y}_B\right) < \overline{y}_{\{\beta\}} \tag{17}$$

establishing the desired results. Hence, finding the optimal split of a categorical variable X in a regression problem with squared error loss reduces to a problem of order $O(|X|)$.

3.4 Numeric Response and Quadratic Taylor Approximation to the Loss Function

3.5 Start Right, Pull Left

Versions of this algorithm appeared in Buntine and Caruana (1991) and Mehta et al. (1996)

Algorithm 2: Start Right, Pull Left

Input: Subset $\mathbf{D} = (R, \mathbb{D}_R) \in \mathfrak{D}$, loss function L, categorical feature X_j
Output: Partition $(\mathbf{D}_l, \mathbf{D}_r) \in 2^{\mathbf{D}}$

$A = \emptyset,\ B = X_j$
for $j \in \{1, ..., |X_j| - 1\}$ **do**
  for $\alpha \in B$ **do**
    $W = A \cup \{\alpha\},\ U = B \setminus \{\alpha\}$
    $R_W = R \cap \{x \in \mathbf{X} : x_j \in W\},\ R_U = R \setminus R_W$
    $(\mathbf{D}_W, \mathbf{D}_U) = \left((R_W, \mathbb{D}_{R_W}), (R_U, \mathbb{D}_{R_U})\right)$
    $b_j(\alpha) = \text{lossImp}\left(\mathbf{D}, \mathbf{D}_{R_W}, \mathbf{D}_{R_U}\right)$
  $\alpha_j^* = \text{argmax}_{\alpha \in B}\, b_j(\alpha)$
  $b_j^* = b_j\left(\alpha_j^*\right)$
  $A_j^* = A \cup \{\alpha_j^*\},\ B_j^* = B \setminus \{\alpha_j^*\}$
  $A = A_j^*,\ B = B_j^*$
$j^* = \text{argmax}_j \{b_j^*\}$
$R_l = R \cap \{x \in \mathbf{X} : x_j \in A_{j^*}^*\},\ R_r = R \setminus R_r$
return $(\mathbf{D}_l, \mathbf{D}_r)$

3.6 Principal Component Scoring

Principal Component Scoring is a heuristic method for splitting categorical variables when the target is a C-class categorical variable. The method was developed by Coppersmith, Hong and Hosking (Partitioning Nominal Attributes in Decision Trees). Suppose $|X| = K$. For ease of notation, enumerate both the K levels of X and the C levels of Y. Define the $K \times C$ matrix $\mathbf{N}$ by $\mathbf{N}_{k,c} = \sum_i \mathbf{1}\{x_i = k, y_i = c\}$. For $k \in \{1, ..., K\}$, let $N_k = \sum_c \mathbf{N}_{k,c}$. Define the $K \times C$ matrix $\mathbf{P}$ by $\mathbf{P}_{k,c} = \mathbf{N}_{k,c}/N_k$. Let the vector of mean class probabilities be

$$\overline{\mathbf{p}} = \frac{1}{N} \sum_k \mathbf{N}_{k\cdot}$$

Let

$$\Sigma = \frac{1}{N-1} \sum_k \mathbf{N}_{k\cdot} \left(\mathbf{P}_{k\cdot} - \overline{\mathbf{p}}\right) \left(\mathbf{P}_{k\cdot} - \overline{\mathbf{p}}\right)^T$$

Let $\mathbf{v}$ be the first principal component of Σ. The principal component score of $\mathbf{P}_{k\cdot}$ is

$$S_k = \mathbf{v} \cdot \mathbf{P}_{k\cdot}$$

If $\{S_{(k)}\}$ are the order statistics for $\{S_k\}$ then we consider all partitions of the form

$$A_j = \{k : S_k < S_{(j)}\}, \qquad B_j = X \setminus A_j \tag{18}$$

Coppersmith et al. recommend also considering partitions of the form

$$A_j = \{k : S_k < S_{(j)} \text{or} S_k = S_{(j+1)}\}, \qquad B_j = X \setminus A_j \tag{19}$$

Experiments in Coppersmith et al. suggests that this heuristic performs comparatively well

3.7 Work

We require a Partition Refinement Algorithm (PRA):

$$C : \mathbf{D} \in \mathfrak{D} \mapsto C\left(\mathbf{D}\right) \in 2^R \tag{20}$$

which specifies a two-partition refinement of any $\mathbf{D} \in \mathfrak{D}$. An obvious PRA is the Basic Binary Split