Classification/Decision Trees (II)

Jia Li

Department of Statistics
The Pennsylvania State University

Email: jiali@stat.psu.edu
http://www.stat.psu.edu/∼jiali

Right Sized Trees

- Let the expected misclassification rate of a tree T be $R^*(T)$.
- Recall the resubstitution estimate for $R^*(T)$ is

$$R(T) = \sum_{t \in \tilde{T}} r(t)p(t) = \sum_{t \in \tilde{T}} R(t) .$$

- $R(T)$ is biased downward.

$$R(t) \geq R(t_L) + R(t_R) .$$

Digit recognition example

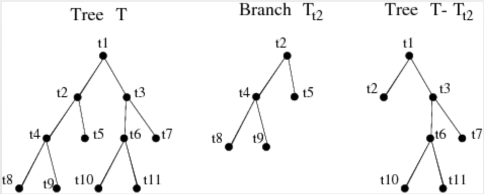
No. Terminal Nodes	$R(T)$	$R^{ts}(T)$
71	.00	.42
63	.00	.40
58	.03	.39
40	.10	.32
34	.12	.32
19	.29	.31
10	.29	.30
9	.32	.34
7	.41	.47
6	.46	.54
5	.53	.61
2	.75	.82
1	.86	.91

- The estimate $R(T)$ becomes increasingly less accurate as the trees grow larger.
- The estimate R^{ts} decreases first when the tree becomes larger, hits minimum at the tree with 10 terminal nodes, and begins to increase when the tree further grows.

Preliminaries for Pruning

- Grow a very large tree T_{max}.
  1. Until all terminal nodes are pure (contain only one class) or contain only identical measurement vectors.
  2. When the number of data in each terminal node is no greater than a certain threshold, say 5, or even 1.
  3. As long as the tree is sufficiently large, the size of the initial tree is not critical.

1. Descendant: a node t' is a descendant of node t if there is a connected path down the tree leading from t to t'.
2. Ancestor: t is an ancestor of t' if t' is its descendant.
3. A branch T_t of T with root node $t \in T$ consists of the node t and all descendants of t in T.
4. Pruning a branch T_t from a tree T consists of deleting from T all descendants of t, that is, cutting off all of T_t except its root node. The tree pruned this way will be denoted by $T - T_t$.
5. If T' is gotten from T by successively pruning off branches, then T' is called a pruned subtree of T and denoted by $T' \prec T$.



Subtrees

- Even for a moderate sized T_{max}, there is an enormously large number of subtrees and an even larger number ways to prune the initial tree to them.
- A "selective" pruning procedure is needed.
  - The pruning is optimal in a certain sense.
  - The search for different ways of pruning should be of manageable computational load.

Minimal Cost-Complexity Pruning

- Definition for the cost-complexity measure:
  - For any subtree $T \preceq T_{max}$, define its complexity as $|\tilde{T}|$, the number of terminal nodes in T. Let $\alpha \geq 0$ be a real number called the *complexity parameter* and define the *cost-complexity measure* $R_\alpha(T)$ as

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}| \; .$$

- For each value of α, find the subtree $T(\alpha)$ that minimizes $R_\alpha(T)$, i.e.,

$$R_\alpha(T(\alpha)) = \min_{T \preceq T_{max}} R_\alpha(T) .$$

  - If α is small, the penalty for having a large number of terminal nodes is small and $T(\alpha)$ tends to be large.
  - For α sufficiently large, the minimizing subtree $T(\alpha)$ will consist of the root node only.

- Since there are at most a finite number of subtrees of T_{max}, $R_\alpha(T(\alpha))$ yields different values for only finitely many α's. $T(\alpha)$ continues to be the minimizing tree when α increases until a jump point is reached.
- Two questions:
  - Is there a unique subtree $T \preceq T_{max}$ which minimizes $R_\alpha(T)$?
  - In the minimizing sequence of trees T_1, T_2, ..., is each subtree obtained by pruning upward from the previous subtree, i.e., does the nesting $T_1 \succ T_2 \succ \cdots \succ \{t_1\}$ hold?

- *Definition*: The smallest minimizing subtree $T(\alpha)$ for complexity parameter α is defined by the conditions:
  1. $R_\alpha(T(\alpha)) = \min_{T \preceq T_{max}} R_\alpha(T)$
  2. If $R_\alpha(T) = R_\alpha(T(\alpha))$, then $T(\alpha) \preceq T$.
- If subtree $T(\alpha)$ exists, it must be unique.
- It can be proved that for every value of α, there exists a smallest minimizing subtree.

- The starting point for the pruning is not T_{max}, but rather $T_1 = T(0)$, which is the smallest subtree of T_{max} satisfying

$$R(T_1) = R(T_{max}) .$$

  - Let t_L and t_R be any two terminal nodes in T_{max} descended from the same parent node t. If $R(t) = R(t_L) + R(t_R)$, prune off t_L and t_R.
  - Continue the process until no more pruning is possible. The resulting tree is T_1.

- For T_t any branch of T_1, define $R(T_t)$ by

$$R(T_t) = \sum_{t' \in \tilde{T}_t} R(t') ,$$

  where $\tilde{T}_t$ is the set of terminal nodes of T_t.
- For t any nonterminal node of T_1, $R(t) > R(T_t)$.

Weakest-Link Cutting

- For any node $t \in T_1$, set $R_\alpha(\{t\}) = R(t) + \alpha$.
- For any branch T_t, define $R_\alpha(T_t) = R(T_t) + \alpha|\tilde{T}_t|$.
- When $\alpha = 0$, $R_0(T_t) < R_0(\{t\})$. The inequality holds for sufficiently small α. But at some critical value of α, the two cost-complexities become equal. For α exceeding this threshold, the inequality is reversed.
- Solve the inequality $R_\alpha(T_t) < R_\alpha(\{t\})$ and get

$$\alpha < \frac{R(t) - R(T_t)}{|\tilde{T}_t| - 1} .$$

The right hand side is always positive.

- Define a function $g_1(t)$, $t \in T_1$ by

$$g_1(t) = \begin{cases} \frac{R(t) - R(T_t)}{|\tilde{T}_t| - 1}, & t \notin \tilde{T}_1 \\ +\infty, & t \in \tilde{T}_1 \end{cases}$$

- Define the *weakest link* $\bar{t}_1$ in T_1 as the node such that

$$g_1(\bar{t}_1) = \min_{t \in T_1} g_1(t) \ .$$

and put $\alpha_2 = g_1(\bar{t}_1)$.

- When α increases, $\bar{t}_1$ is the first node that becomes more preferable than the branch $T_{\bar{t}_1}$ descended from it.
- α_2 is the first value after $\alpha_1 = 0$ that yields a strict subtree of T_1 with a smaller cost-complexity at this complexity parameter. That is, for all $\alpha_1 \leq \alpha < \alpha_2$, the tree with smallest cost-complexity is T_1.
- Let $T_2 = T_1 - T_{\bar{t}_1}$.

- Repeat the previous steps. Use T_2 instead of T_1, find the weakest link in T_2 and prune off at the weakest link node.

$$
\begin{aligned}
g_2(t) &= \begin{cases} \frac{R(t)-R(T_{2t})}{|\tilde{T}_{2t}|-1}, & t \in T_2, t \notin \tilde{T}_2 \\ +\infty, & t \in \tilde{T}_2 \end{cases} \\
g_2(\bar{t}_2) &= \min_{t \in T_2} g_2(t) \\
\alpha_3 &= g_2(\bar{t}_2) \\
T_3 &= T_2 - T_{\bar{t}_2}
\end{aligned}
$$

- If at any stage, there are multiple weakest links, for instance, if $g_k(\bar{t}_k) = g_k(\bar{t}'_k)$, then define $T_{k+1} = T_k - T_{\bar{t}_k} - T_{\bar{t}'_k}$.
  - Two branches are either nested or share no node.

- A decreasing sequence of nested subtrees are obtained:

$$T_1 \succ T_2 \succ T_3 \succ \cdots \succ \{t_1\} .$$

- *Theorem*: *The* $\{\alpha_k\}$ *are an increasing sequence, that is,* $\alpha_k < \alpha_{k+1}$, $k \geq 1$, *where* $\alpha_1 = 0$. *For* $k \geq 1$, $\alpha_k \leq \alpha < \alpha_{k+1}$, $T(\alpha) = T(\alpha_k) = T_k$.

- At the initial steps of pruning, the algorithm tends to cut off large subbranches with many leaf nodes. With the tree becoming smaller, it tends to cut off fewer.
- Digit recognition example:

Tree	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}
$\lvert\tilde{T}_k\rvert$	71	63	58	40	34	19	10	9	7	6	5	2	1

Best Pruned Subtree

- Two approaches to choose the best pruned subtree:
  - Use a test sample set.
  - Cross-validation
- Use a test set to compute the classification error rate of each minimum cost-complexity subtree. Choose the subtree with the minimum test error rate.
- Cross validation: tree structures are not stable. When the training data set changes slightly, there may be large structural change in the tree.
  - It is difficult to correspond a subtree trained from the entire data set to a subtree trained from a majority part of it.
  - Focus on choosing the right complexity parameter α.

Pruning by Cross-Validation

- Consider V-fold cross-validation. The original learning sample $\mathcal{L}$ is divided by random selection into V subsets, $\mathcal{L}_v$, $v = 1, ..., V$. Let the training sample set in each fold be $\mathcal{L}^{(v)} = \mathcal{L} - \mathcal{L}_v$.
- The tree grown on the original set is T_{max}. V accessory trees $T_{max}^{(v)}$ are grown on $\mathcal{L}^{(v)}$.

- For each value of the complexity parameter α, let $T(\alpha)$, $T^{(v)}(\alpha)$, $v = 1, ..., V$, be the corresponding minimal cost-complexity subtrees of T_{max}, $T_{max}^{(v)}$.
  - For each maximum tree, we obtain a sequence of jump points of α:
    $\alpha_1 < \alpha_2 < \alpha_3 \cdots < \alpha_k < \cdots$.
  - To find the corresponding minimal cost-complexity subtree at α, find α_k from the list such that $\alpha_k \leq \alpha < \alpha_{k+1}$. Then the subtree corresponding to α_k is the subtree for α.

- The cross-validation error rate of $T(\alpha)$ is computed by

$$R^{CV}(T(\alpha)) = \frac{1}{V}\sum_{v=1}^{V}\frac{N_{miss}^{(v)}}{N^{(v)}} \ ,$$

where $N^{(v)}$ is the number of samples in the test set $\mathcal{L}_v$ in fold v; and $N_{miss}^{(v)}$ is the number of misclassified samples in $\mathcal{L}_v$ using $T^{(v)}(\alpha)$, a pruned tree of $T_{max}^{(v)}$ trained from $\mathcal{L}^{(v)}$.
- Although α is continuous, there are only finite minimum cost-complexity trees grown on $\mathcal{L}$.

- Let $T_k = T(\alpha_k)$. To compute the cross-validation error rate of T_k, let $\alpha'_k = \sqrt{\alpha_k \alpha_{k+1}}$.
- Let
$$R^{CV}(T_k) = R^{CV}(T(\alpha'_k)) \ .$$
- For the root node tree $\{t_1\}$, $R^{CV}(\{t_1\})$ is set to the resubstitution cost $R(\{t_1\})$.
- Choose the subtree T_k with minimum cross-validation error rate $R^{CV}(T_k)$.

Computation Involved

1. Grow $V+1$ maximum trees.
2. For each of the $V+1$ trees, find the sequence of subtrees with minimum cost-complexity.
3. Suppose the maximum tree grown on the original data set T_{max} has K subtrees.
4. For each of the $(K-1)$ α'_k, compute the misclassification rate of each of the V test sample set, average the error rates and set the mean to the cross-validation error rate.
5. Find the subtree of T_{max} with minimum $R^{CV}(T_k)$.