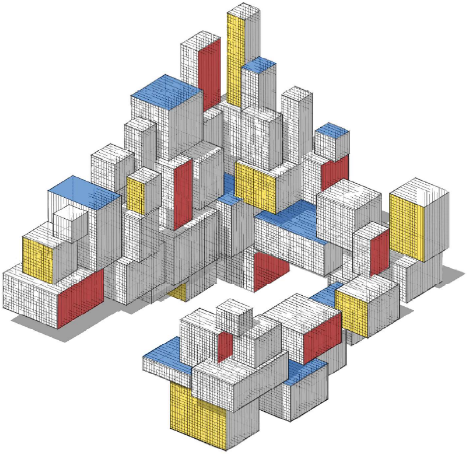
Tree Methods in Practice



Purpose

In this lecture we discuss various practical issues concerning decision trees:

- Binary vs non-binary trees
- Data preprocessing
- Alternative splitting rules
- Dealing with categorical variables and missing values
- Decision tree pruning
- Cost–complexity pruning
- Advantages and disadvantages of decision trees

Binary Versus Non-Binary Trees

While it is possible to split a tree node into more than two groups (multiway splits), it generally produces inferior results compared to the simple binary split.

The major reason is that multiway splits can lead to too many nodes near the tree root that have only a few data points, thus leaving insufficient data for later splits. As multiway splits can be represented as several binary splits, the latter is preferred.

Data Preprocessing

Sometimes, it can be beneficial to preprocess the data prior to the tree construction.

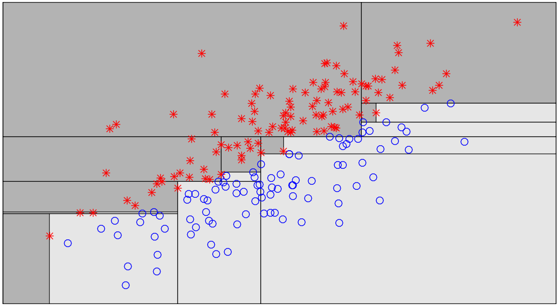
For example, PCA can be used with a view to identify the most important dimensions, which in turn will lead to simpler and possibly more informative splitting rules in the internal nodes.

Alternative Splitting Rules

We restricted our attention to splitting rules of the type $s(\boldsymbol{x}) = \mathbb{I}\{x_j \leqslant \xi\}$, where $j \in \{1, \dots, p\}$ and $\xi \in \mathbb{R}$.

These types of rules may not always result in a simple partition of the feature space.

In this figure, the feature space could have been partitioned into just two regions, separated by a straight line.



Alternative Splitting Rules

In the previous example, the classification tree is rather elaborate, dividing the feature set into too many regions. An obvious remedy is to use splitting rules of the form

$$s(\boldsymbol{x}) = \mathbb{I}\{\boldsymbol{a}^\top \boldsymbol{x} \leqslant \xi\}.$$

It may be useful to use a splitting rule that involves several variables, as opposed to a single one.

The decision regarding the split type clearly depends on the problem domain. For example, for logical (binary) variables our domain knowledge may indicate that a different behavior is expected when both x_i and x_j ($i \neq j$) are `True`.

In this case, we will naturally introduce a decision rule of the form:

$$s(\boldsymbol{x}) = \mathbb{I}\{x_i = \texttt{True} \text{ and } x_j = \texttt{True}\}.$$

Categorical Variables

When an explanatory variable is categorical with labels (levels) say $\{1, \ldots, k\}$, the splitting rule is generally defined via a partition of the label set $\{1, \ldots, k\}$ into two subsets.

Specifically, let L and R be a partition of $\{1, \ldots, k\}$. Then, the splitting rule is defined via

$$s(\boldsymbol{x}) = \mathbb{I}\{x_j \in L\}.$$

For the general supervised learning case, finding the optimal partition in the sense of minimal loss requires one to consider 2^k subsets of $\{1, \ldots, k\}$.

Consequently, finding a good splitting rule for categorical variables can be challenging when the number of labels p is large.

Missing Values

When working with incomplete feature vectors, where one or more values are missing, it is typical to either completely delete the feature vector from the data (which may distort the data) or to impute (guess) its missing values from the available data.

Tree methods allow an elegant approach for handling missing data via *surrogate* splitting rules that use different variables and thresholds but affect a similar split in the data as the original splitting rule.

The similarity is generally measured via a binary misclassification loss, where the true classes of observations are determined by the primary splitting rule and the surrogate splitting rules serve as classifiers.

Example: Surrogate Splitting Rule

Suppose that the primary splitting rule at node v is $\mathbb{I}\{\text{Age} \leqslant 25\}$, leading to a $\{1, 2\}$, $\{3, 4, 5\}$ split in the data below.

Next, consider the following surrogate splitting rules:

1. $\mathbb{I}\{\text{Salary} \leqslant 1500\}$, and
2. $\mathbb{I}\{\text{Height} \leqslant 173\}$.

Table: Example data with three variables (Age, Height, and Salary).

Id	Age	Height	Salary
1	20	173	1000
2	25	168	1500
3	38	191	1700
4	49	170	1900
5	62	182	2000

Example: Surrogate Splitting Rule

The $\mathbb{I}\{\text{Salary} \leqslant 1500\}$ surrogate rule completely mimics the primary rule, leading to $\{1, 2\}$, $\{3, 4, 5\}$ split.

The $\mathbb{I}\{\text{Height} \leqslant 173\}$ rule is less similar to the primary rule, since it causes the different partition $\{1, 2, 4\}$ and $\{3, 5\}$.

It is up to the user to define the number of surrogate rules for each tree node.

As soon as these surrogate rules are available, we can use them to handle a new data point, even if the main rule cannot be applied due to a missing value of the primary variable x_{j^*}.

Specifically, if the observation is missing the primary split variable, we apply the first (best) surrogate rule. If the first surrogate variable is also missing, we apply the second best surrogate rule, and so on.

Controlling the Tree Shape

Ultimately, we are interested in getting the right-size tree. Namely, a tree that shows good generalization properties. It was already discussed that shallow trees tend to underfit and deep trees tend to overfit the data.

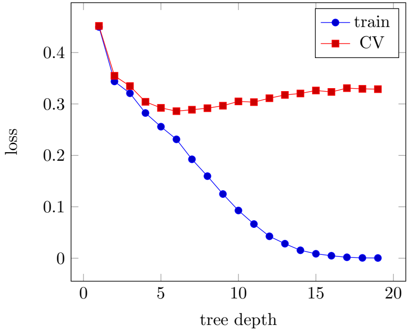


Figure: The cross-validation and the training loss as a function of the tree depth for a binary classification problem.

Tree Pruning

To address the issue over over- and under-fitting, a so-called pruning routine can be employed.

Idea: first grow a very deep tree and then prune (remove nodes) it upwards until we reach the root node.

While the tree is being pruned, the generalization risk gradually decreases up to the point where it starts increasing again, at which point the pruning is stopped.

Tree Pruning

We say that v' is a descendant of v if there is a path down the tree, which leads from v to v'. If such a path exists, we also say that v is an ancestor of v'.

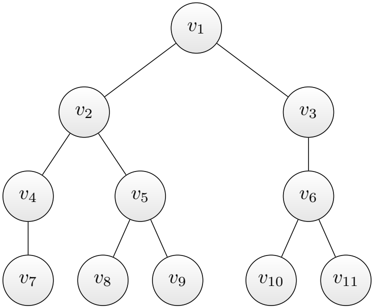


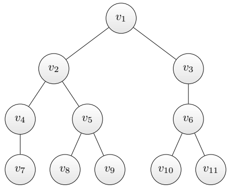
Figure: The node v_9 is a descendant of v_2, and v_2 is an ancestor of $\{v_4, v_5, v_7, v_8, v_9\}$, but v_6 is not a descendant of v_2.

Definition: Branches and Pruning

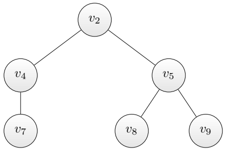
1. A tree branch $\mathbb{T}_v$ of the tree $\mathbb{T}$ is a sub-tree of $\mathbb{T}$ rooted at node $v \in \mathbb{T}$.
2. The pruning of branch $\mathbb{T}_v$ from a tree $\mathbb{T}$ is performed via deletion of the entire branch $\mathbb{T}_v$ from $\mathbb{T}$ except the branch's root node v. The resulting pruned tree is denoted by $\mathbb{T} - \mathbb{T}_v$.
3. A sub-tree $\mathbb{T} - \mathbb{T}_v$ is called a pruned sub-tree of $\mathbb{T}$. We indicate this with the notation $\mathbb{T} - \mathbb{T}_v \prec \mathbb{T}$ or $\mathbb{T} \succ \mathbb{T} - \mathbb{T}_v$.

Example: Tree Pruning

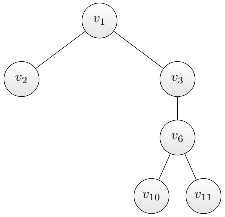
The pruned tree $\mathbb{T} - \mathbb{T}_{v_2}$ in (c) is the result of pruning the $\mathbb{T}_{v_2}$ branch in (b) from the original tree $\mathbb{T}$ in (a).



(a) $\mathbb{T}$



(b) $\mathbb{T}_{v_2}$



(c) $\mathbb{T} - \mathbb{T}_{v_2}$

```
Algorithm 1: Decision Tree Pruning
Input: Training set $\tau$.
Output: Sequence of decision trees $\mathbb{T}^0 \succ \mathbb{T}^1 \succ \cdots$
1 Build a large decision tree $\mathbb{T}^0$. [A possible termination criterion
    for that algorithm is to have some small predetermined number
    of data points at each terminal node of $\mathbb{T}^0$.]
2 $\mathbb{T}' \leftarrow \mathbb{T}^0$
3 $k \leftarrow 0$
4 while $\mathbb{T}'$ has more than one node do
5     $k \leftarrow k + 1$
6     Choose $v \in \mathbb{T}'$.
7     Prune the branch rooted at $v$ from $\mathbb{T}'$.
8     $\mathbb{T}^k \leftarrow \mathbb{T}' - \mathbb{T}_v$ and $\mathbb{T}' \leftarrow \mathbb{T}^k$.
9 return $\mathbb{T}^0, \mathbb{T}^1, \ldots, \mathbb{T}^k$
```

Tree Selection

Let $\mathbb{T}^0$ be the initial (deep) tree and let $\mathbb{T}^k$ be the tree obtained after the k-th pruning operation, for $k = 1, \ldots, K$.

As soon as the sequence of trees $\mathbb{T}^0 \succ \mathbb{T}^1 \succ \cdots \succ \mathbb{T}^K$ is available, one can choose the best tree of $\{\mathbb{T}^k\}_{k=1}^K$ according to the smallest generalization risk.

Specifically, we can split the data into training and validation sets. In this case, Algorithm 1 is executed using the training set and the generalization risks of $\{\mathbb{T}^k\}_{k=1}^K$ are estimated via the validation set.

While Algorithm 1 and the corresponding best tree selection process look appealing, there is still an important question to consider; namely, how to choose the node v and the corresponding branch $\mathbb{T}_v$ in Line 6 of the algorithm. In order to overcome this problem, Breiman proposed a method called cost complexity pruning.

Cost-Complexity Pruning

Let $\mathbb{T} \prec \mathbb{T}^0$ be a tree obtained via pruning of a tree $\mathbb{T}^0$. Denote the set of leaf (terminal) nodes of $\mathbb{T}$ by $\mathcal{W}$.

The number of leaves $|\mathcal{W}|$ is a measure for the complexity of the tree; recall that $|\mathcal{W}|$ is the number of regions $\{\mathcal{R}_w\}$ in the partition of $\mathcal{X}$.

Corresponding to $\mathbb{T}$ is a prediction function g. In cost-complexity pruning the objective is to find a prediction function g (or tree $\mathbb{T}$) that minimizes the training loss $\ell_\tau(g)$ while taking into account the complexity of the tree.

The idea is to regularize the training loss, by adding a penalty term for the complexity of the tree.

Definition: Cost-Complexity Measure

Let $\tau = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be a data set and $\gamma \geqslant 0$ be a real number. For a given tree $\mathbb{T}$, the cost-complexity measure $C_\tau(\gamma, \mathbb{T})$ is defined as:

$$\begin{aligned} C_\tau(\gamma, \mathbb{T}) &:= \frac{1}{n} \sum_{w \in \mathcal{W}} \left(\sum_{i=1}^n \mathbb{I}\{\boldsymbol{x}_i \in \mathcal{R}_w\} \operatorname{Loss}(y_i, g^w(\boldsymbol{x}_i)) \right) + \gamma\, |\mathcal{W}| \\ &= \ell_\tau(g) + \gamma\, |\mathcal{W}|, \end{aligned}$$

where $\ell_\tau(g)$ is the training loss.

Cost–Complexity Pruning

Small values of γ result in a small penalty for the tree complexity $|\mathcal{W}|$, and thus large trees (that fit the entire *training* data well) will minimize the measure $C_\tau(\gamma, \mathbb{T})$.

In particular, for $\gamma = 0$, $\mathbb{T} = \mathbb{T}^0$ will be the minimizer of $C_\tau(\gamma, \mathbb{T})$.

On the other hand, large values of γ will prefer smaller trees or, more precisely, trees with fewer leaves.

For sufficiently large γ, the solution $\mathbb{T}$ will collapse to a single (root) node.

It can be shown that, for every value of γ, there exists a smallest minimizing sub-tree with respect to the cost-complexity measure.

In practice, a suitable γ is selected via observing the performance of the learner on the validation set or by cross-validation.

Advantages of Decision Trees

1. The tree structure can handle both categorical and numerical features in a natural and straightforward way (e.g., no need for dummy variables).
2. The final tree obtained after the training phase can be compactly stored for the purpose of making predictions for new feature vectors. The prediction process only involves a single tree traversal from the tree root to a leaf.
3. The hierarchical nature of decision trees allows for an efficient encoding of the feature's conditional information.
4. The tree structure can be easily understood and interpreted by domain experts with little statistical knowledge, since it is essentially a logical decision flow diagram.
5. The sequential decision tree growth procedure provides an implicit step-wise variable elimination procedure.

6. Decision trees are invariant under monotone transformations of the data. To see this, consider the (optimal) splitting rule $s(\boldsymbol{x}) = \mathbb{I}\{x_3 \leqslant 2\}$, where x_3 is a positive feature. Suppose that x_3 is transformed to $x_3' = x_3^2$. Now, the optimal splitting rule will take the form $s(\boldsymbol{x}) = \mathbb{I}\{x_3' \leqslant 4\}$.
7. In the classification setting, it is common to report not only the predicted value of a feature vector, but also the respective class probabilities. Decision trees handle this task without any additional effort. Specifically, consider a new feature vector. During the estimation process, we will perform a tree traversal and the point will end up in a certain leaf w. The probability of this feature vector lying in class z can be estimated as the proportion of training points in w that are in class z.
8. As each training point is treated equally in the construction of a tree, their structure of the tree will be relatively robust to outliers. In a way, trees exhibit a similar kind of robustness as the sample median does for real-valued data.

Limitations of Decision Trees

1. The predictive accuracy of decision tress is generally inferior to other established statistical learning methods.
2. Decision trees, and in particular very deep trees that were not subject to pruning, are heavily reliant on their training set. A small change in the training set can result in a dramatic change of the resulting decision tree.

The inferior predictive accuracy is a direct consequence of the bias–variance tradeoff. Specifically, a decision tree model generally exhibits a high variance.

To overcome the above limitations, several ensemble techniques can be employed:

- Bagging
- Random forests
- Boosting

The bagging approach was initially introduced in the context of an ensemble of decision trees. However, both the bagging and the boosting methods can be applied to improve the accuracy of general prediction functions.