Decision Trees

Prof. Ziping Zhao

School of Information Science and Technology
ShanghaiTech University, Shanghai, China

CS182: Introduction to Machine Learning (Fall 2021)
http://cs182.sist.shanghaitech.edu.cn

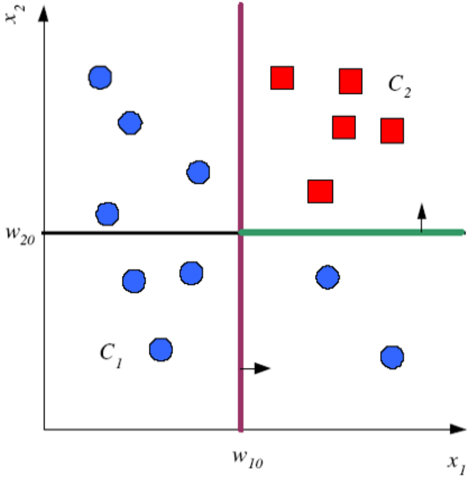
Outline

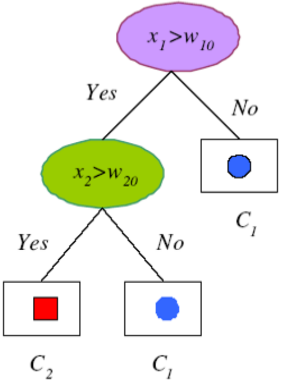
Outline

Introduction

- In nonparametric estimation, the input space is divided into local regions each of which corresponds to a local model computed from the training data in that region.
- A decision tree is a nonparametric hierarchical model for supervised learning whereby the local region is identified through a sequence of recursive splits in a small number of steps – divide-and-conquer approach.
- Decision trees were first made popular in statistics and later in machine learning.
- Two types of nodes in a decision tree:
  - (Internal) decision node: a test function with discrete outcomes labeling the branches.
  - (Terminal) leaf node: the value associated with it constitutes the output (class label for classification; numeric value for regression).

Data Set and Corresponding Decision Tree





Discriminants

- The test function $f_m(\mathbf{x})$ at each decision node m defines a discriminant in the input space dividing it into smaller regions which are further subdivided as we take a path from the root down.
- A leaf node defines a localized region in the input space where instances falling in the region have the same output.
- The region boundaries are defined by the discriminants that are coded in the internal nodes along the path from the root to the leaf node.
- Advantages of decision trees:
  - Fast localization of the region covering the input as a result of the hierarchical placement of decisions.
  - High interpretability: can be converted easily into a set of IF-THEN rules that are easily understandable.

Outline

Univariate Trees I

- In a univariate tree, the test in each internal decision node uses only one of the input dimensions.
- n-way split: a discrete-valued input dimension x_j with n possible values leads to n branches from the decision node (binary split is a special case with $n = 2$).
- A numeric input should be discretized into $n \geq 2$ values using suitably chosen threshold(s). Usually we choose $n = 2$, and the discriminant is

$$f_m(\mathbf{x}) : x_j > w_{m0}$$

- Successive splits are orthogonal to each other. The leaf nodes define hyperrectangles in the input space.

Univariate Trees II

- Tree induction or tree learning is the construction of a tree given a training sample.
- For a given training set, there exists many trees that code it with no error.
- We are interested in finding the smallest one, where tree size is measured as the number of nodes in the tree and the complexity of the decision nodes.
- Given a training sample, finding the smallest tree to code the data is NP-complete, making it necessary to use greedy local search algorithms for tree learning.
  - starting at the root with the complete training data, we look for the best split at each step.
  - we continue splitting recursively with the corresponding subset until we do not need to split anymore, at which point a leaf node is created.

Classification Trees

- In the case of a decision tree for classification, namely, a classification tree.
- For node m:
  - N_m training instances
  - N_m^i of N_m instances belong to class C_i , i.e., $\sum_i N_m^i = N_m$.
- Given that an instance reaches node m, an estimate for the probability of class C_i is

$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p_m^i = \frac{N_m^i}{N_m}$$

- Node m is considered pure if there is a class C_i with $p_m^i = 1$ (and hence all other p_m^j with $j \neq i$ are 0). No further splitting is needed and the node becomes a leaf node with class label C_i .
- The goodness of a split in classification trees is quantified by an impurity measure.
- Some impurity measures:
  - Entropy
  - Gini index
  - Misclassification error

Entropy

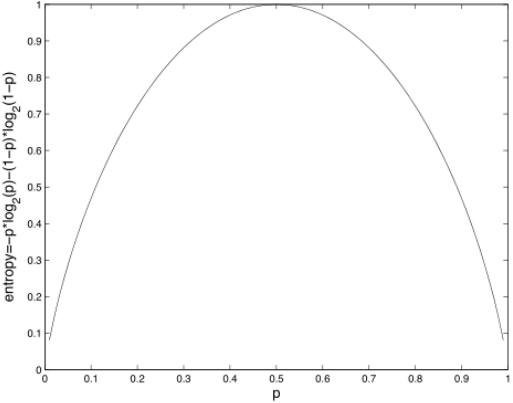
- Entropy at node m:

$$\mathcal{I}_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$$

  where we assume $0 \log 0 = 0$.

- The largest entropy is $\log_2 K$ when all $p_m^i = 1/K$.

- Entropy function for a two-class problem:



Other Impurity Measures

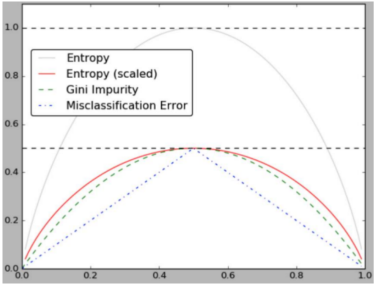
- For a two-class problem where $p^1 = p$ and $p^2 = 1 - p$, a nonnegative function $\phi(p, 1-p)$ measures the impurity of a split if it satisfies the following properties:
  - $\phi(1/2, 1/2) \geq \phi(p, 1-p)$, $\forall p \in [0, 1]$
  - $\phi(0, 1) = \phi(1, 0) = 0$
  - $\phi(p, 1-p)$ is increasing in p on $[0, 1/2]$ and decreasing in p on $[1/2, 1]$ (very often we need it to be symmetric).

- Examples other than entropy:
  - Gini index:

    $$\phi(p, 1-p) = 2p(1-p)$$

    which is used in economics as a measure of unequal distribution of wealth.
  - Misclassification error:

    $$\phi(p, 1-p) = 1 - \max(p, 1-p)$$



Best Split

- At node m, let N_{mj} of the N_m instances take branch j, i.e., $f_m(\mathbf{x}) = j$. Also, N^i_{mj} of the N_{mj} instances belong to class C_i. So,
$$\sum_{j=1}^{n} N_{mj} = N_m \qquad \sum_{i=1}^{K} N^i_{mj} = N_{mj} \qquad \sum_{j=1}^{n} N^i_{mj} = N^i_m.$$
- If $f_m(\mathbf{x}) = j$, the estimate for the probability of class C_i is
$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$
and the total impurity after the split is
$$\mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log p^i_{mj}$$
- All attributes are tried to implement the split and the one that gives the minimum entropy is chosen for the test function.
- Tree construction continues recursively and in parallel for all the branches that are not pure, until all are pure.

Classification Tree Construction Algorithm

```
GenerateTree(𝒳)
    If NodeEntropy(𝒳) < θ_I
        Create leaf labelled by majority class in 𝒳
        Return
    i ← SplitAttribute(𝒳)
    For each branch of x_i
        Find 𝒳_i falling in branch
        GenerateTree(𝒳_i)
SplitAttribute(𝒳)
    MinEnt ← MAX
    For all attributes i = 1, ..., d
        If x_i is discrete with n values
            Split 𝒳 into 𝒳_1, ..., 𝒳_n by x_i
            e ← SplitEntropy(𝒳_1, ..., 𝒳_n)
            If e<MinEnt MinEnt ← e; bestf ← i
        Else /* x_i is numeric */
            For all possible splits
                Split 𝒳 into 𝒳_1, 𝒳_2 on x_i
                e ← SplitEntropy(𝒳_1, 𝒳_2)
                If e<MinEnt MinEnt ← e; bestf ← i
    Return bestf
```

Regression Trees

- A regression tree is constructed in a similar manner as a classification tree, except that the impurity measure for classification is replaced by a measure appropriate for regression.
- For node m, let $\mathcal{X}_m \subset \mathcal{X}$ denote the set of instances reaching m. We define the following indicator function:

$$b_m(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{X}_m \\ 0 & \text{otherwise} \end{cases}$$

- Estimated value at node m:

$$g_m = \frac{\sum_\ell b_m(\mathbf{x}^{(\ell)}) r^{(\ell)}}{\sum_\ell b_m(\mathbf{x}^{(\ell)})}$$

- Mean squared error of estimated value measures goodness of split:

$$E_m = \frac{1}{N_m} \sum_\ell (r^{(\ell)} - g_m)^2 b_m(\mathbf{x}^{(\ell)})$$

where $N_m = |\mathcal{X}_m| = \sum_\ell b_m(\mathbf{x}^{(\ell)})$.

Tree Expansion

- If $E_m < \theta_r$ for some threshold θ_r, the error is acceptable and hence node m is designated as a leaf node with value g_m stored.
- If E_m is too large, we look for a split threshold w_{m0} for further splitting so that the sum of the errors in the branches is minimum, and then we continue recursively.
- At node m, let $\mathcal{X}_{mj}$ be the subset of $\mathcal{X}_m$ taking branch j. We define

$$b_{mj}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{X}_{mj} \\ 0 & \text{otherwise} \end{cases}$$

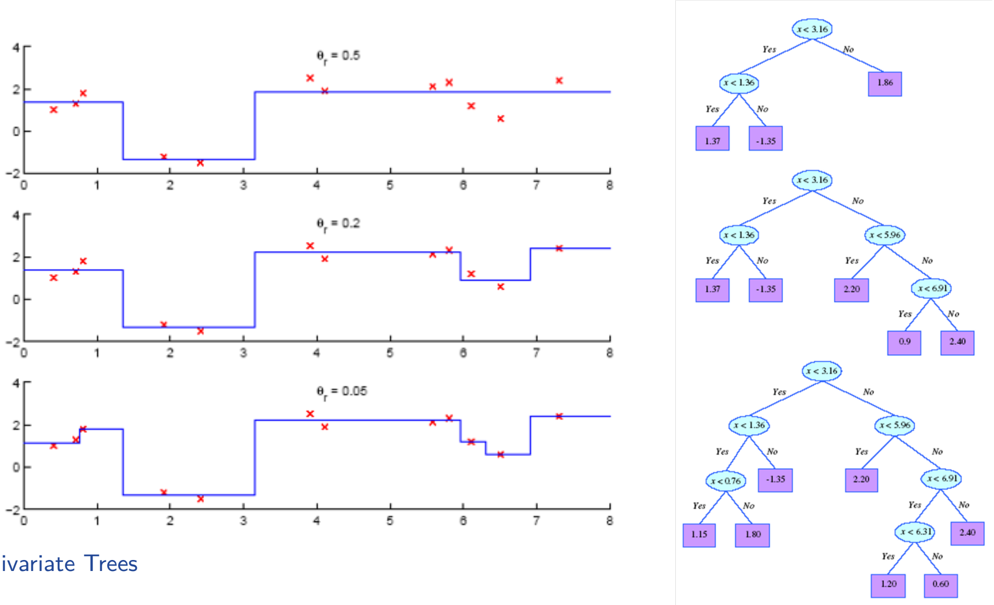
- Estimated value in branch j of node m:

$$g_{mj} = \frac{\sum_\ell b_{mj}(\mathbf{x}^{(\ell)}) r^{(\ell)}}{\sum_\ell b_{mj}(\mathbf{x}^{(\ell)})}$$

- Error after split:

$$E'_m = \frac{1}{N_m} \sum_j \sum_\ell (r^{(\ell)} - g_{mj})^2 b_{mj}(\mathbf{x}^{(\ell)})$$

Regression Trees for Different Values of θ_r



Best Split

- As in classification, we look for the split that results in the smallest error E'_m and then split the node to expand the tree.
- Besides the mean squared error, other error functions may also be used. E.g., worst possible error:

$$E_m = \max_j \max_\ell \left\{ |y^{(\ell)} - g_{mj}| b_m(\mathbf{x}^{(\ell)}) \right\}$$

  which can guarantee that the error for any instance is never larger than a given threshold.
- Similar to going from running mean to running line in nonparametric regression, instead of taking an average at a leaf that implements a constant fit, we can also do a linear regression fit over the instances choosing the leaf:

$$g_m(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_{m0}$$

  which makes the estimate dependent on $\mathbf{x}$ and generates smaller trees, but introduces extra computation.

Outline

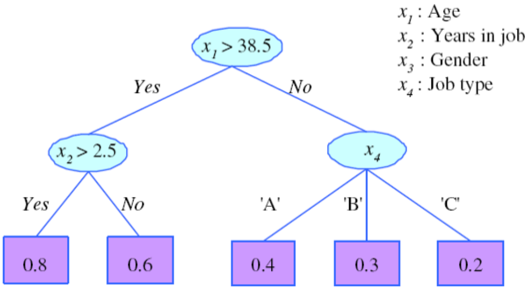
Pruning

- If very few training instances reach a node, decision based on the instances may give high generalization error.
- Solution 1 – prepruning:
  - Stop node split when the number of instances reaching a node is below a certain percentage of the training set regardless of the impurity or error.
- Solution 2 – postpruning:
  - Grow the tree full until all leaves are pure.
  - Find subtrees and do the following for each of them:
    - Replace the subtree by a leaf node set with an appropriate label (for either classication or regression) based on the training instances covered by the subtree.
    - If the leaf node does not perform worse than the subtree on the pruning set (a separate labeled data set), the subtree is pruned and replaced by the leaf node.
- Prepruning is faster but postpruning generally leads to more accurate trees.

Outline

Feature Extraction

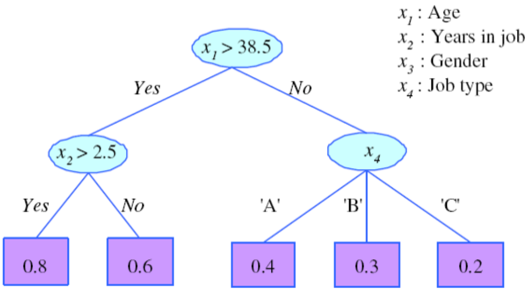
- The univariate tree only uses the variables that are necessary, so it is possible to use the tree for feature extraction.
- E.g., only features x_1, x_2, x_4 are extracted:



- In general, features closer to the root are more important globally.

Rule Extraction

- A main advantage of decision trees is interpretability.
- A set of IF-THEN rules (R), i.e., a rule base:



R1: IF (age > 38.5) AND (years-in-job > 2.5) THEN $y = 0.8$

R2: IF (age > 38.5) AND (years-in-job ≤ 2.5) THEN $y = 0.6$

R3: IF (age ≤ 38.5) AND (job-type $=$ 'A') THEN $y = 0.4$

R4: IF (age ≤ 38.5) AND (job-type $=$ 'B') THEN $y = 0.3$

R5: IF (age ≤ 38.5) AND (job-type $=$ 'C') THEN $y = 0.2$

More on Rules

- For classification, more than one leaf node may be labeled with the same class. So the condition part of the corresponding rule can be expressed as a disjunction (OR) of conjunctions (AND), e.g.

$$\text{IF } (x_1 \le w_{10}) \text{ OR } ((x_1 > w_{10}) \text{ AND } (x_2 \le w_{20})) \text{ THEN } C_1$$

- Pruning rules (i.e., pruning a term from one rule without touching other rules) is possible for simplification. But after the rules are pruned, it may not be possible to write them back as a tree anymore.

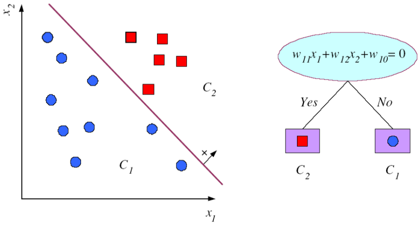
- Instead of extracting rules from a decision tree learned from data, it is also possible to learn the rules directly from data.
- Rule induction is similar to tree induction, but:
  - Tree induction is breadth-first.
  - Rule induction is depth-first; one rule at a time.

Outline

Multivariate Trees

- At each decision node of a multivariate tree, all input attributes can be used to dene a test function for the split.
- Linear multivariate node for numeric attributes:

$$f_m(\mathbf{x}) : \mathbf{w}_m^T\mathbf{x} + w_{m0} > 0$$



- A node $f_m(\mathbf{x})$ defines a hyperplane with arbitrary orientation and leaf nodes define polyhedra in the input space.
- A univariate tree can be seen as a special case.

Nonlinear Multivariate Nodes

- Nonlinear multivariate nodes provide even more exibility, e.g.:
  - Quadratic multivariate node:

$$f_m(\mathbf{x}) : \mathbf{x}^T \mathbf{W}_m \mathbf{x} + \mathbf{w}_m^T \mathbf{x} + w_{m0} > 0$$

  - Multilayer perceptron
  - Sphere node:

$$f_m(\mathbf{x}) : \ \|\mathbf{x} - \mathbf{c}_m\| \leq \alpha_m$$

    where $\mathbf{c}_m$ is the center and α_m is the radius.
- Omnivariate decision tree: a hybrid tree architecture where the tree may have univariate, linear multivariate, or nonlinear multivariate nodes.
- While providing additional flexibility is good, multivariate decision trees also increase the computational requirement significantly. Univariate trees are still the more popular choice in practical applications.