Contents

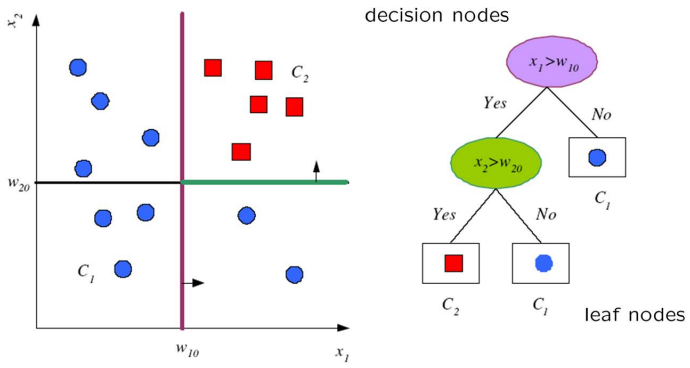
1 Decision Tree for Classification

Decision Trees (1/2)

- A decision tree is a hierarchical model for supervised learning
  - Implementing the divide-and-conquer strategy
  - An efficient nonparametric method
- A decision tree has internal decision nodes and leaf nodes (terminal leaves)
  - Each decision node m implements a test function $f_m(\mathbf{x})$ with discrete outcomes labeling the branches.
  - Given an input, a test is applied at a decision node, and then one of the branches is taken depending on the outcome. This process starts at the root and is repeated recursively until a leaf node is reached.

Decision Trees (2/2)

- A simple example of decision tree



Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute x_i

    Numeric x_i: Binary split with $x_i > w_m$ (i.e. $f_m(\mathbf{x})$)

    Discrete x_i: n-way split for n possible values

    e.g., In the previous figure, from the root to a leaf generating splits orthogonal to each other.
  - Multivariate: Uses all attributes, $\mathbf{x}$

Tree Learning (Univariate) (1/2)

- Tree induction is the construction of a (decision) tree given a training dataset.
- For a given dataset, there exist many trees that code it with no error.

  We are interested in finding the smallest among them, where tree size is measured as the number of nodes in the tree and the complexity of the decision nodes.
- However, finding the smallest tree is NP-complete.

  In practice, greedy algorithms based on local search are used to yield reasonable trees in reasonable time.

6

Tree Learning (Univariate) (2/2)

- Starting at the root with the complete training data, look for the best split that divides the training data into two or n parts, depending on whether the chosen attribute is numeric or discrete.
- The splitting is applied recursively with the corresponding subset of training data until a certain criterion is met, at which point a leaf node is created and labeled.

7

Classification Trees (Univariate)

- The goodness of a split in a classification tree is quantified by an impurity measure.
- A split is pure if after the split, for all branches, all the instances choosing a branch belong to the same class.
- Suppose N_m instances reaching node m, and N_m^i belong to class C_i. Then the estimate for the probability of class C_i is given by

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

  Node m is pure $\exists i$ such that $P_m^i = 1$ and $P_m^j = 0$ for all $j \neq i$, and then node m is a leaf with class label i.

8

Measure of Impurity

- One possible function to measure impurity is entropy

$$\mathscr{I}_m = -\sum_{i=1}^{K} p_m^i \log p_m^i \qquad (*)$$

  where $0 \log 0 \equiv 0$.
- In general, for 2-class classification problem, $\phi(p, 1-p)$ is a nonnegative function measuring the impurity of a split if it satisfies
  1. $\phi(1/2, 1/2) \geq \phi(p, 1-p)$ for any $p \in [0,1]$
  2. $\phi(0,1) = \phi(1,0) = 0$
  3. $\phi(p, 1-p)$ is increasing in $p \in [0, 1/2]$ and decreasing in $p \in [1/2, 1]$

9

Best Split

- If node m is pure, generate a leaf and stop, otherwise split and continue recursively
- Impurity after split: N_{m_j} of N_m take branch j, and $N^i_{m_j}$ belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p^i_{m_j} = \frac{N^i_{m_j}}{N_{m_j}}$$

$$\mathscr{I}'_m = -\sum_{j=1}^{n} \frac{N_{m_j}}{N_m} \sum_{i=1}^{K} p^i_{m_j} \log p^i_{m_j} = \sum_{j=1}^{n} \frac{N_{m_j}}{N_m} \mathscr{I}_{m_j} \qquad (**)$$

$\mathscr{I}_m - \mathscr{I}'_m$ is the information gain of the split at node m.

- Find the variable and split yielding min impurity

Classification Tree Construction (1/2)

- **GenerateTree**($\mathscr{X}$)

```
1. if NodeEntropy (X < θ_I)      /* eqn. (*) */
2.   Create leaf labeled by majority class in X
3.   return
4. i ⟵ SplitAttribute(X)
5. for each branch of x_i
6.   Find X_i falling in branch
7.   GenerateTree (X_i)
```


Classification Tree Construction (2/2)

- **SplitAttribute**($\mathscr{X}$)

```
1. MinEnt ⟵ MAX
2. for all attributes i = 1,...,d
3.   if x_i is discrete with n values
4.     Split X into X_1,...,X_n by x_i     /* eqn. (**) */
5.     e ⟵ SplitEntropy(X_1,...,X_n)
6.     if e < MinEnt  MinEnt ⟵ e;  bestf ⟵ i
7.   else
8.     for all possible splits
9.       split X into X_1, X_2 on x_i
10.      e ⟵ SplitEntropy(X_1, X_2)
11.      if e < MinEnt  MinEnt ⟵ e;  bestf ⟵ i
12. return bestf
```

2 Decision Tree for Regression

Regression Trees (Univariate) (1/2)

- A regression tree is constructed analogously as a classification tree, except that the impurity measure is replaced by a measure appropriate for regression.
- Notations
  - Training dataset: $D = (\mathbf{x}_i, y_i)$
  - $\mathbf{x} \in D$ reaching node m is denoted by $b_m(\mathbf{x}) = 1$, and 0, otherwise.
  - Data reaching node m: $D_m = \{\mathbf{x} \in D \mid b_m(\mathbf{x}) = 1\}$. Let $|D_m| = N_m$.
- Mean Square Error at node m:

$$E_m = \frac{1}{N_m} \sum_i (y_i - g_m)^2 b_m(\mathbf{x}_i)$$

where g_m is the estimated value at node m. For example, we may set $g_m = \frac{\sum_i b_m(\mathbf{x}_i) y_i}{\sum_i b_m(\mathbf{x}_i)}$. Then E_m now becomes variance.

13

Regression Trees (Univariate) (2/2)

- If at node m, $E_m < \theta_y$, then a leaf node is created and it stores the g_m value. Otherwise, D_m is split further.

  $\mathbf{x} \in D$ reaching node m taking branch j is denoted by

$$b_{m_j}(\mathbf{x}) = 1$$
$$D_{m_j} = \{\mathbf{x} \in D \mid b_{m_j}(\mathbf{x}) = 1\}, \bigcup_j D_{m_j} = D_m$$

- Mean Square Error after the split at node m:

$$E'_m = \frac{1}{N_m} \sum_j \sum_i (y_i - g_{m_j})^2 b_{m_j}(\mathbf{x}_i)$$

where g_{m_j} is the estimated value in branch j of node m.

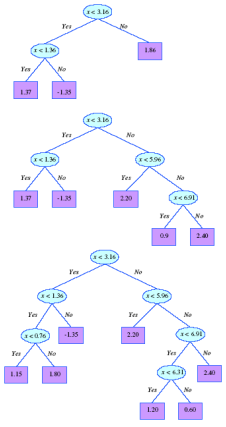
- Instead of taking average, we may use linear regression at a leaf node

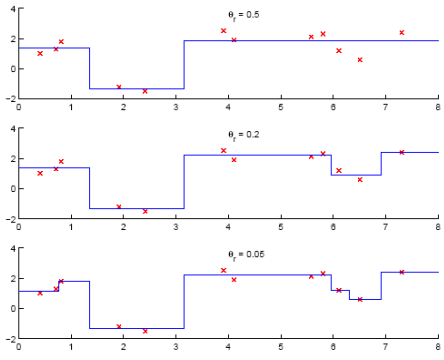
$$g_m(\mathbf{x}) = \mathbf{w}_m^T \mathbf{x} + \mathbf{w}_{m0}$$

14

Regression Trees: Model Selection

- The error threshold θ_y is the complexity parameter. When it is small,we obtain large trees and risk overfitting; when it is large, we underfit and smooth too much.





15

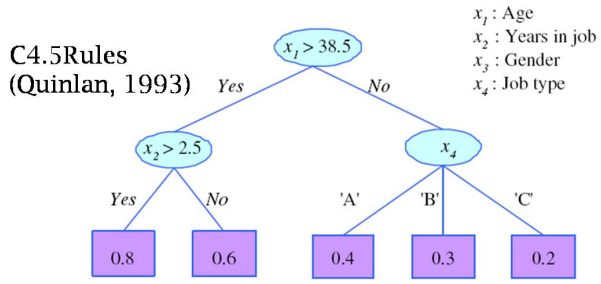
Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping (at a node with too few training instances)
  - Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)

16

Rule Extraction from Trees

- A decision tree does its own feature extraction.

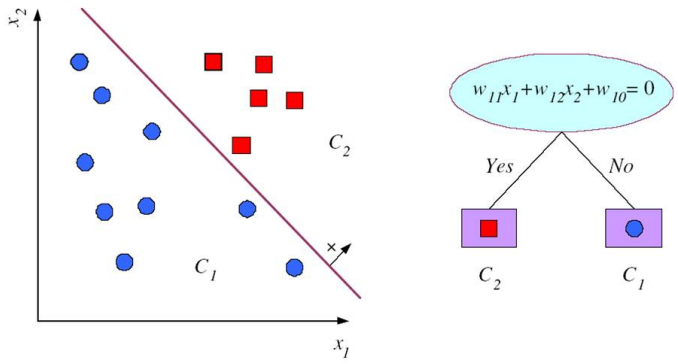


R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

17

Multivariate Decision Trees

- In a multivariate tree, at a decision node, all input dimensions can be used. (cf. only one input dimension is considered at each decision node of a univariate tree)



18

3 Random Forest

Random Forests (1/5)

- Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.
- The generalization error of a random forest converges almost surely to a limit as the number of trees in the forest becomes large.
- The generalization error of a random forest depends on the strength of the individual trees in the forest and the correlation between them.
- Using a random selection of features to split each node yields error rates that compare favorably (?) to AdaBoost, but are more robust with respect to noise.

19

Random Forests (2/5)

- Margin function: Suppose the training set is derived by drawing at random from the distribution P of the random vectors $\mathbf{X}, Y$. Given an ensemble of classifiers $h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x})$, we define the margin function as

$$\text{margin}(\mathbf{x}, y) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(h_t(\mathbf{x}) = y) - \max_{j \neq y} \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(h_t(\mathbf{x}) = j)$$

  where $\mathbb{I}(\cdot)$ is the indicator function.
- A *random forest* is a classifier consisting of a collection of tree-structured classifiers $\{h(\mathbf{x}, \Theta_t), t = 1, \ldots\}$ where the $\{\Theta_t\}$ are independent identically distributed (i.i.d.) random vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$.

20

Random Forests (3/5)

- The generalization error is given by

$$\text{err}^* = P_{(\mathbf{x},y)}\{\text{margin}(\mathbf{x}, y) < 0\}$$

- As the number of trees increases, for almost surely all sequences $\Theta_1, \ldots$, the generalization error of the random forest converges to

$$P_{(\mathbf{x},y)}\{P_\Theta(h(\mathbf{x}, \Theta) = y) - \max_{j \neq y} P_\Theta(h(\mathbf{x}, \Theta) = j) < 0\}$$

- Indeed for random forests, an upper bound can be derived for the generalization error in terms of two parameters that are measures of how accurate the individual classifiers are and of the dependence between them.

21

Random Forests (4/5)

- The margin function for a random forest is

$$\text{margin}(\mathbf{x}, y) = P_\Theta\left(h(\mathbf{x}, \Theta) = y\right) - \max_{j \neq y} P_\Theta\left(h(\mathbf{x}, \Theta) = j\right)$$

  and the strength of the set of classifiers $\{h(\mathbf{x}, \Theta)\}$ is

$$s = E_{(\mathbf{x},y)\sim P}\text{margin}(\mathbf{x}, y)$$

- Assuming $s \geq 0$, it can be shown from Chebychev's inequality that

$$\text{err}^* \leq \text{var}(\text{margin}(\mathbf{x}, \mathrm{y}))/s^2$$

- Let $\hat{j}(\mathbf{x}, y) = \underset{j \neq y}{argmax} P_\Theta(h(\mathbf{x}, \Theta) = j)$. We can rewrite the definition of margin function by

$$\begin{aligned} \text{margin}(\mathbf{x}, y) &= P_\Theta\left(h(\mathbf{x}, \Theta) = y\right) - \max_{j \neq y} P_\Theta\left(h(\mathbf{x}, \Theta) = j\right) \\ &= P_\Theta\left(h(\mathbf{x}, \Theta) = y\right) - P_\Theta(h(\mathbf{x}, \Theta) = \hat{j}(\mathbf{x}, y)) \\ &= E_\Theta[\mathbb{I}\left(h(\mathbf{x}, \Theta) = y\right) - \mathbb{I}(h(\mathbf{x}, \Theta) = \hat{j}(\mathbf{x}, y))] \end{aligned}$$

Random Forests (5/5)

- Define the raw margin function as

$$\text{rmargin}(\mathbf{x}, y, \Theta) = \mathbb{I}\left(h(\mathbf{x}, \Theta) = y\right) - \mathbb{I}(h(\mathbf{x}, \Theta) = \hat{j}(\mathbf{x}, y))$$

  Thus, $\text{margin}(\mathbf{x}, y)$ is the expectation of $\text{rmargin}(\mathbf{x}, y, \Theta)$ with respect to Θ.

- Let $\rho(\Theta, \Theta')$ be the correlation between $\text{rmargin}(\mathbf{x}, y, \Theta)$ and $\text{rmargin}(\mathbf{x}, y, \Theta')$ holding Θ, Θ' fixed. Also denote $\bar{\rho}$ as the mean value of the correlation $\rho(\Theta, \Theta')$. Then the upper bound for the generalization error can be reduced to

$$\boxed{\text{err}^* \leq \frac{\bar{\rho}(1 - s^2)}{s^2}}$$