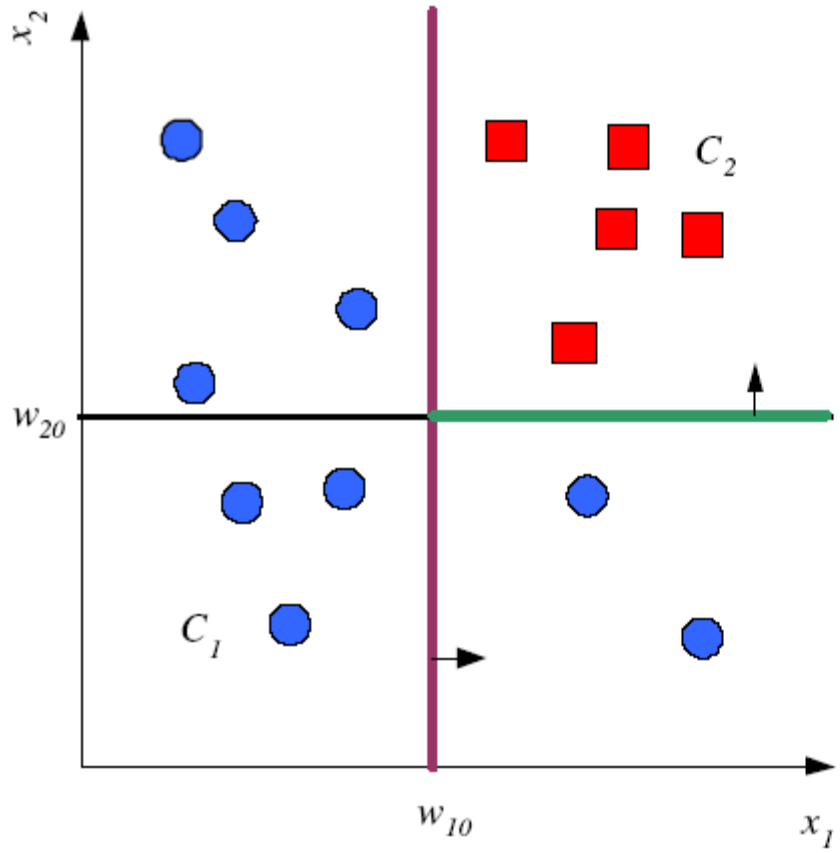
CHAPTER 9:

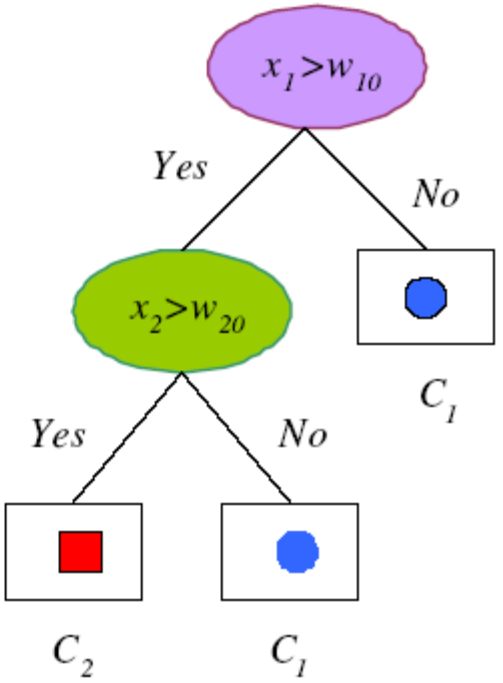
Decision Trees

Decision Tree

- Induce a general rule by observing labeled instances
- Structure
  - Each internal node specifies a test on attributes
  - Branch represents an attribute value or condition
  - Leaves represent a class

Example: Tree Uses Nodes, and Leaves





Decision tree: Construction

General Idea

- Select best Attribute for root node. Construct a branch for every possible value of that feature.
- Split data into mutually exclusive subsets for each branch
- Repeat this process recursively using only the portion of data arriving at each node
- Stop when training examples can be classified and create a leaf node with the class decision

Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute, x_i
    - Numeric x_i : Binary split : $x_i > w_m$
    - Discrete x_i : n-way split for n possible values
  - Multivariate: Uses all attributes, $\boldsymbol{x}$
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; r average, or local fit

Best Split: Selecting an attribute

- Information Gain (used in ID3, C4.5)
  - Information" an attribute gives us about the class.

- GiniMeasure (used in CART): $1-\sum_{i=k}^{K} {p_k}^2$

  K = number of classes

- Misclassification error : $1 - \max_k p_k$

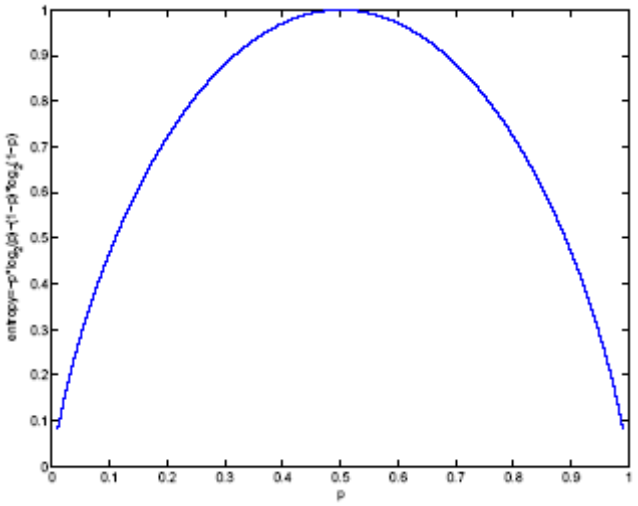
There are others criteria to select best split

Information gain

- attributes that perfectly partition should give maximal information
- It measures the reduction in entropy
  - Entropy: (im)purity in an arbitrary collection of examples

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

N_m instances reach m, N^i_m belong to C_i



  - *Probability of class C_i:*

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

  - Node m is pure if p^i_m is 0 or 1

Best Split

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

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

$$\mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- Find the variable and split with min impurity (among all variables -- and split positions for numeric variables)
  - High entropy: uniform distribution
  - Low entropy: varied distribution (more desirable)

Example: Weather data

Outlook	Temp	Humidity	Windy	Play?
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

```
GenerateTree(X)
    If NodeEntropy(X)< θ_I /* eq. 9.3
        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) /* eq. 9.8 */
            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
```

When to stop

- When no more attributes to split on
- When you have too few examples
  - To minimize variance or generalization error
- When minimum node impurity is reached
- When the node is pure.

Continuous values

- Partition into discrete set of intervals

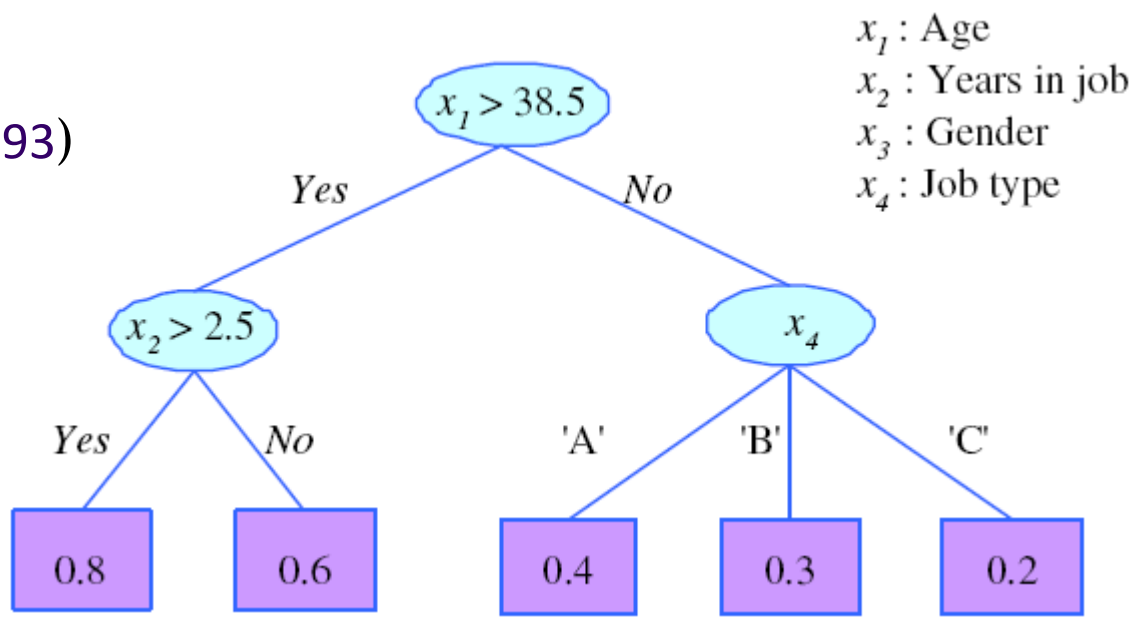
- Consider all possible splits and pick the one that gives minimum impurity.

Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - 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)

Rule Extraction from Trees

C4.5Rules
(Quinlan, 1993)



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

Multivariate Trees