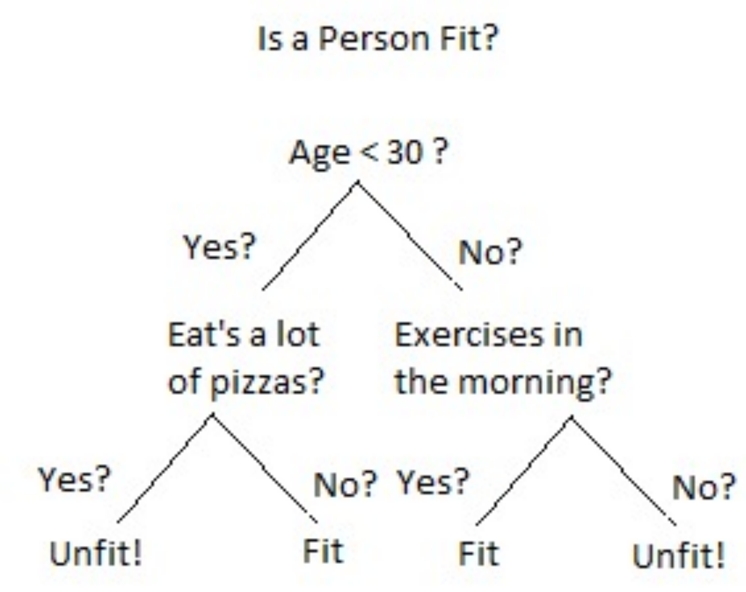
# Lecture 9: decision trees

Dr. Huiping Cao

# Decision trees - introduction

- Decision tree is a simple and easy to interpret classifier.
- In decision trees, the data is continuously split according to a certain parameter.
- The tree can be explained by two entities, namely **decision nodes** and **leaves**. The leaves are the decisions or the final outcomes, and the decision nodes are where the data is split.



# Decision tree learning

- How many trees? $2^m$ (exponential in the number of attributes *m*)
  - Trees containing only one attribute
  - Trees containing two attributes
  - …
  - Trees containing m attributes
- Many algorithms: reasonably accurate, suboptimal, reasonable amount of time
  - Hunt's Algorithm (basis of many others)
  - CART (Classification and Regression Trees), a book by Breiman et al.
  - ID3, C4.5 by Quinlan

# Hunt's algorithm

- Start from the tree root and split the data on the feature that results in the largest **Information Gain (IG)**.
- Input:
  - $D_t$ is the set of training records that reach a node $t$
    - Root node corresponds to the whole training dataset
  - $y = \{y_1, y_2, \cdots, y_c\}$ are class labels
- General procedure
  - If $D_t$ contains records that belong to the same class $y_t$ (i.e., **the node is pure**), then $t$ is a leaf node labeled as $y_t$
  - If $D_t$ contains records that belong to more than one class
    - Use an **attribute test** to split the data into smaller subsets
    - Recursively apply the procedure to each subset

# Hunt’s algorithm – questions to ask

- Which attribute to choose to split the node?
- How to split the attribute?

# Measures of node impurity

- Let $p(i|t)$ represent the probability that an instance $t$ in $D_t$ belongs to class $C_i$, estimated by $\frac{n_i}{|D_t|}$, where $n_i$ is the number of instances with class label $C_i$.
- Gini index

$$Gini(t) = 1 - \sum_{i=0}^{c-1} (p(i|t))^2$$

- Entropy (log uses base 2: information is encoded in bits)

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t) log_2 p(i|t)$$

- Classification error

$$Classification\ error(t) = 1 - max_i\{p(i|t)\}$$

# Gini index

$$Gini(t) = 1 - \sum_{i=0}^{c-1} (p(i|t))^2$$

| | |
|---|---|
| C1 | 0 |
| C2 | 6 |

(a) Node $N_1$

| | |
|---|---|
| C1 | 1 |
| C2 | 5 |

(b) Node $N_2$

| | |
|---|---|
| C1 | 2 |
| C2 | 4 |

(c) Node $N_3$

| | |
|---|---|
| C1 | 3 |
| C2 | 3 |

(d) Node $N_4$

Table 1: Data example 1

$$Gini(N_1) = 1 - \left(\frac{0}{6}\right)^2 - \left(\frac{6}{6}\right)^2 = 0$$

$$Gini(N_2) = 1 - \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2 = \frac{10}{36} = 0.278$$

$$Gini(N_3) = 1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2 = \frac{16}{36} = 0.444$$

$$Gini(N_4) = 1 - 0.5^2 - 0.5^2 = 0.5$$

- Maximum: $1 - \frac{1}{n_c}$ when records are equally distributed among all classes, implying least interesting information
- Minimum: (0.0) when all records belong to one class, implying most interesting information
- First used in CART, which allows only binary splitting

# Entropy

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t)log_2 p(i|t)$$

| | |
|---|---|
| C1 | 0 |
| C2 | 6 |

(a) Node $N_1$

| | |
|---|---|
| C1 | 1 |
| C2 | 5 |

(b) Node $N_2$

| | |
|---|---|
| C1 | 2 |
| C2 | 4 |

(c) Node $N_3$

| | |
|---|---|
| C1 | 3 |
| C2 | 3 |

(d) Node $N_4$

Table 1: Data example 1

$$Entropy(N_1) = -0log0 - 1log1 = 0$$

$$Entropy(N_2) = -\frac{1}{6}\log(\frac{1}{6}) - \frac{5}{6}\log\left(\frac{5}{6}\right) = 0.65$$

$$Entropy(N_3) = -\frac{2}{6}\log(\frac{2}{6}) - \frac{4}{6}\log\left(\frac{4}{6}\right) = 0.92$$

$$Entropy(N_4) = -0.5\log 0.5 - 0.5log0.5 = -(-1) = 1$$

- prob. is 0 means it does not happen, let 0*log*0 = 0
- Measures homogeneity of a node.
  - Maximum: $\log(n_c)$ when records are equally distributed among all classes implying least information. Given $n_c$ : the number of classes. Then, $-n_c * \left(\frac{1}{n_c}\right) * \log_2\left(\frac{1}{n_c}\right) = -\log_2\left(\frac{1}{n_c}\right) = \log_2(n_c)$
  - Minimum: 0.0 when all records belong to one class, implying most information

# Determine the best split – Maximizing the information gain

- The attribute that we choose to split the data to smaller subsets need to gain the **maximum information gain**.

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

  - $f$: the feature/attribute to perform the split.
  - $D_p$: dataset corresponding to the parent node.
  - $D_j$: dataset corresponding to the *j*th child node.
  - $N_p, N_\mathrm{j}$: number of samples in $D_p$ and $D_j$ respectively.
  - $I(\cdot)$: impurity measure of a node.

# Determine the best split

- Most libraries implement binary decision tree.

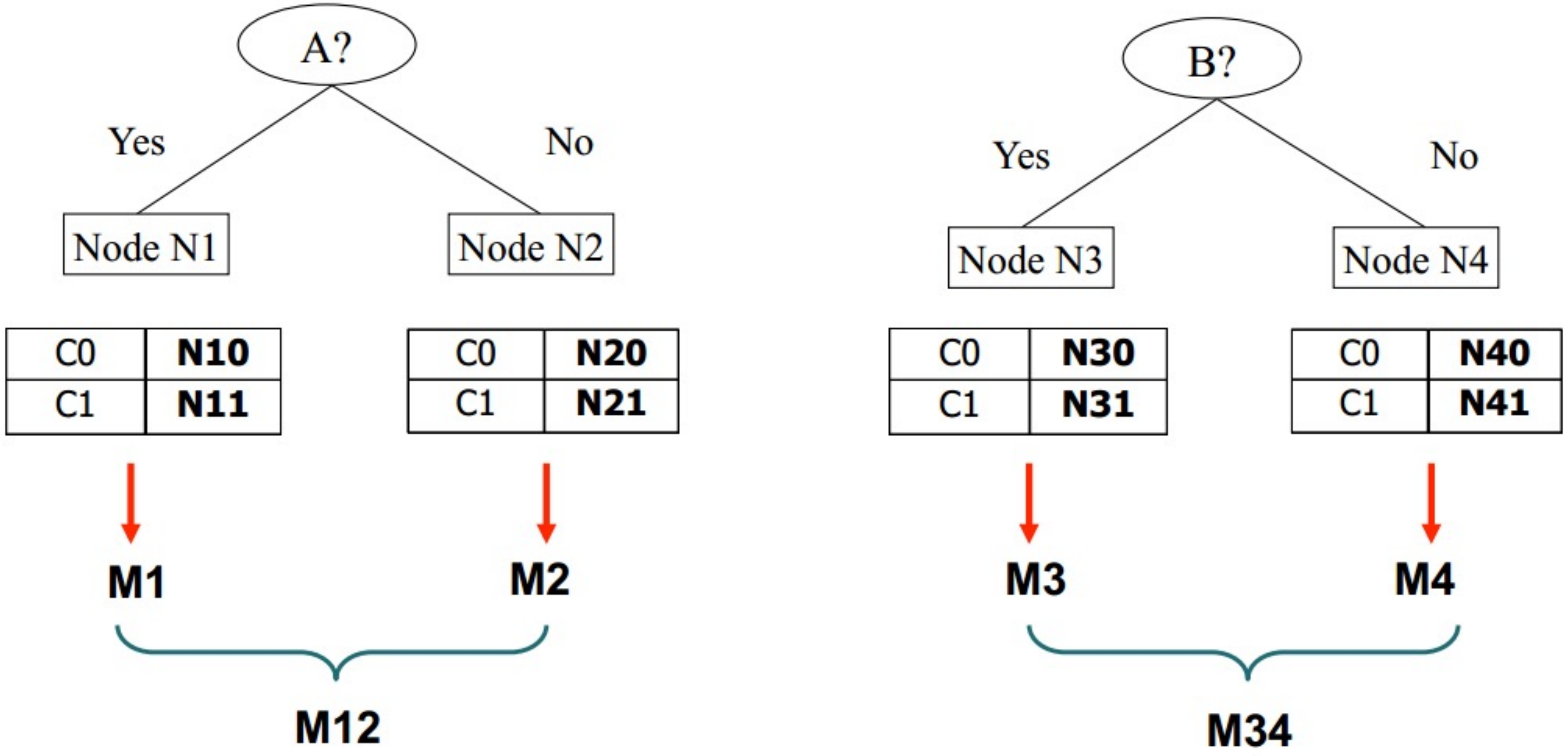
$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

# Binary decision tree - example

Before splitting:

| | |
|---|---|
| C0 | N00 |
| C1 | N01 |

$M0 = I(N)$




$$\Delta = I(N) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

$$Gain = M0 - M12 \ vs \ M0 - M34$$

# Binary decision tree - example

Splitting one attribute $A$

| | parent node $N_p$ | left child node $N_1$ | right child node $N_2$ |
|---|---|---|---|
| Instances belonging to class $C1$ | 40 | 30 | 10 |
| Instances belonging to class $C2$ | 40 | 10 | 30 |

Splitting one attribute $B$

| | parent node $N_p$ | left child node $N_3$ | right child node $N_4$ |
|---|---|---|---|
| Instances belonging to class $C1$ | 40 | 20 | 20 |
| Instances belonging to class $C2$ | 40 | 40 | 0 |

Using **gini index**:

Splitting on *A*: $\mathrm{IG}(\mathrm{N_p}, \mathrm{A}) = 0.5 - \frac{4}{8} * 0.375 - \frac{4}{8} * 0.375 = 0.125$

Splitting on *B*: $\mathrm{IG}(\mathrm{N_p}, \mathrm{B}) = 0.5 - \frac{6}{8} * 0.4 - 0 = 0.16$

Thus, splitting on *B* is preferred.

Using **Entropy**

Splitting on *A*: $\mathrm{IG}(\mathrm{N_p}, \mathrm{A}) = 1 - \frac{4}{8} * 0.81 - \frac{4}{8} * 0.81 = 0.19$

Splitting on *B*: $\mathrm{IG}(\mathrm{N_p}, \mathrm{B}) = 1 - \frac{6}{8} * 0.92 - 0 = 0.31$

Thus, splitting on *B* is preferred.

Using **classification error**

Splitting on *A*: $\mathrm{IG}(\mathrm{N_p}, \mathrm{A}) = 0.5 - \frac{4}{8} * 0.25 - \frac{4}{8} * 0.5 = 0.25$

Splitting on *B*: $\mathrm{IG}(\mathrm{N_p}, \mathrm{B}) = 0.5 - \frac{6}{8} * \frac{1}{3} - 0 = 0.25$

Thus, no preference on splitting attribute.

# Stopping criteria for tree induction

- Stop expanding a node when all the records belong to **the same class**
- Stop expanding a node when all the records have the **same attribute values**
- In practice, such stopping criteria may result in a deep tree with many nodes, which may lead to an issue called overfitting. Thus, the decision trees typically need to be pruned by setting a limit for the maximal depth of the tree.

# Decision tree based classification - discussions

- Advantages
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets
- Decision Boundary
  - Decision boundary is a border line between two neighboring regions of different classes
  - Decision boundary is parallel to axes because test condition involves a single attribute at-a-time.

# Overfitting

- **Classification errors**
  - **Training errors**: the number of misclassification errors on training records
  - **Generalization/test errors**: the expected error of the model on previously unseen records
- **Overfitting and Underfitting**
  - **Model overfitting**: A model fits the training data too well but has poorer generalization error than a model with a higher training error.
  - **Model underfitting**: a model has not learned the true structure of the data. It has high training error and generalization error.
  - Underfitting: when a model is too simple; both training and test errors are large

# Variance & Bias

- When a model suffers from overfitting, we also say that the model has a **high variance**.
    - Which can be caused by having too many parameters, leading to a model that is too complex given the underlying data.
- When a model suffers from underfitting, we also say that the model has **high bias**.
    - The model is not complex enough to capture the pattern in the training data well and therefore also suffers from low performance on unseen data.

# Overfitting reasons

- Possible reason 1: noise
- Possible reason 2: insufficient samples
- What is the primary reason for overfitting? a subject of debate
- Generally agreed: the complexity of a model has an impact on model overfitting

# How to address overfitting: pre-pruning (early stropping rule)

- Stop the algorithm before it becomes a fully-grown tree
- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- More restrictive conditions
  - Stop if the number of instances is less than some user-specified threshold
  - Stop if expanding the current node does not improve impurity measures or estimated generalization error.

# How to address overfitting: post-pruning

- Grow decision tree to its entirety
- Trim the nodes of the decision tree in a bottom-up fashion
- Replace a subtree by
  - a leaf node: class label is determined from majority class of instances in the sub-tree (subtree replacement)
  - most frequently used branch of the subtree (subtree raising)
- Termination: no further improvement on generalization errors

# Building a decision tree

- Feature scaling is not required for decision tree algorithms.

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

tree_model = DecisionTreeClassifier(criterion='gini',
                                    max_depth=4,
                                    random_state=1)
tree_model.fit(X_train, y_train)

tree.plot_tree(tree_model)
plt.show()
```

*class* sklearn.tree.DecisionTreeClassifier(***criterion***=*'gini'*, *splitter='best'*, ***max_depth***=*None*, ***min_samples_split***=*2*, ***min_samples_leaf***=*1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, ***min_impurity_split***=*None*, *class_weight=None*, *presort='deprecated'*, *ccp_alpha=0.0*)

- **criterion:** The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.
- **max_depth:** the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- **min_samples_split***:* the minimum number of samples required to split an internal node
- **min_samples_leaf***:* the minimum number of samples required to be at a leaf node.

# DecisionTreeClassifier parameter

- **min_impurity_decrease:** A node will be split if this split induces a decrease of the impurity greater than or equal to this value. The weighted impurity decrease equation is the following

$$\frac{N_t}{N} * impurity - \frac{N_{t_R}}{N_t} * right\_impurity - \frac{N_{t_L}}{N_t} * left\_impurity$$

- Where N is the total number of samples
- $N_t$ is the number of samples at the current node
- $N_{t_L}$ is the number of samples in the left child
- $N_{t_R}$ is the number of samples in the right child

# Prediction

- **predict**(*self*, *X*, *check_input=True*): Predict class (or regression value) for X.
- **predict_proba**(*self*, *X*, *check_input=True*) Predict class probabilities of the input samples X.