Week 7

Data Mining and Knowledge Discovery

Dr John Evans
j.evans8@herts.ac.uk

University of Hertfordshire UH

Plan for today

Classification

Decision Tree Classification

The Algorithm

Attribute Selection Measures

Classification

- Classification is a data mining task in which we attempt to extract models describing important data classes. Such models, called *classifiers*, predict categorical (discrete, unordered) class labels.

Classification

- Classification is a data mining task in which we attempt to extract models describing important data classes. Such models, called *classifiers*, predict categorical (discrete, unordered) class labels.
- e.g. We can build a classification model to categorise bank loan applications as either safe or risky. This could include looking at attributes relating to an applicant's income, savings, financial history etc.
- e.g. We can build a classification model to categorise lizards into one of several species. This could include looking at attributes relating to length, width, colour, diet, waking habits (nocturnal or diurnal) etc.
- e.g. We can build a classification model that categorises email as spam or not spam. This could include looking at attributes relating to header and content etc.
- e.g. We could classify cells as malignant or benign based upon MRI scans.
- e.g. We could classify galaxies based upon their shapes. Are they a spiral galaxy or an elliptical galaxy, for example.

Preliminaries

- The input data for a classification task is a collection of records.

Preliminaries

- The input data for a classification task is a collection of records.
- Each record (also known as an instance or example) is characterised by a tuple $(\mathbf{x}, y)$, where $\mathbf{x}$ is the attribute set and y is a special attribute, designated as the class label (also known as category or target attribute).
- In general, the attributes can be discrete or continuous (or a combination of both), but the class label must be a discrete attribute.

Preliminaries

- The input data for a classification task is a collection of records.
- Each record (also known as an instance or example) is characterised by a tuple $(\mathbf{x}, y)$, where $\mathbf{x}$ is the attribute set and y is a special attribute, designated as the class label (also known as category or target attribute).
- In general, the attributes can be discrete or continuous (or a combination of both), but the class label must be a discrete attribute.
- This is a key characteristic that distinguishes classification from *regression*.
- For example, suppose we have a marketing manager who wants to guess whether a customer with a given profile will buy a new computer. This is a *classification task* where the classifiers are ‘yes’ and ‘no’.

Preliminaries

- The input data for a classification task is a collection of records.
- Each record (also known as an instance or example) is characterised by a tuple $(\mathbf{x}, y)$, where $\mathbf{x}$ is the attribute set and y is a special attribute, designated as the class label (also known as category or target attribute).
- In general, the attributes can be discrete or continuous (or a combination of both), but the class label must be a discrete attribute.
- This is a key characteristic that distinguishes classification from *regression*.
- For example, suppose we have a marketing manager who wants to guess whether a customer with a given profile will buy a new computer. This is a *classification task* where the classifiers are 'yes' and 'no'.
- However, if the manager wanted instead to predict how much this given customer will spend, then this becomes one of *numeric prediction*, or regression. In this case, y is a continuous attribute.

Example

Consider the problem of classifying vertebrates into one of the following categories: mammal, bird, fish, reptile, or amphibian. The following is a table which shows a sample data set.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal

Example (Data set continued)

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class Label
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

Example (The idea)

Idea: Construct a model based on this data, and then use this model to characterise a creature not present in the set.

For example, suppose we encounter a gila monster which is cold-blooded, has skin covered in scales, does not give birth, is not aquatic, is not aerial, has legs and hibernates. What class label should we give it, based on the data set above?

General approach to classification (Step 1)

1. Learning Step - this is where the classification model $y = f(\mathbf{x})$ is constructed.
   - Training Phase
     We take a predetermined dataset. Our classification then uses a *learning algorithm* to build the classifier by analysing (or ‘learning from’) a *training set* which is a subset of this dataset. The key point here is that we know the class label of each data point.

General approach to classification (Step 1)

1. Learning Step - this is where the classification model $y = f(\mathbf{x})$ is constructed.
   - Training Phase
     We take a predetermined dataset. Our classification then uses a *learning algorithm* to build the classifier by analysing (or ‘learning from’) a *training set* which is a subset of this dataset. The key point here is that we know the class label of each data point.
   - The individual data points making up the training set are referred to as *training tuples, samples, examples, instances, data points* or *objects.* They are randomly sampled from the database being analysed.

General approach to classification (Step 1 continued)

1. Learning Step - this is where the classification model $y = f(\mathbf{x})$ is constructed.
   - Supervised learning
     Because the class label of each training tuple is provided, this step is also known as *supervised learning* (i.e. the learning of the classifier is 'supervised' in that it is told to which class each training tuple belongs). This contrasts with *unsupervised learning* in which the class label of each training tuple is not known, and the number or set of classes to be learned may not be known in advance. In this case, we can use clustering to try to determine 'groups of similar tuples', and from there attempt to derive labels.

General approach to classification (Step 1 continued)

1. Learning Step - this is where the classification model $y = f(\mathbf{x})$ is constructed.
   - Supervised learning
     Because the class label of each training tuple is provided, this step is also known as *supervised learning* (i.e. the learning of the classifier is ‘supervised’ in that it is told to which class each training tuple belongs). This contrasts with *unsupervised learning* in which the class label of each training tuple is not known, and the number or set of classes to be learned may not be known in advance. In this case, we can use clustering to try to determine ‘groups of similar tuples’, and from there attempt to derive labels.
   - The class label attribute is discrete-valued and unordered. Our classification model is represented in the form of rules, decision trees or mathematical formulae. This could include neural networks, support vector machines, naive Bayes classifiers etc.

General approach to classification (Step 2)

2. Classification Step - where the model $y = f(\mathbf{x})$ is used to predict class labels for the given data.
   - First, the predictive accuracy of the classifier is estimated. If we were to use the training set to measure the classifier's accuracy, this estimate would likely be optimistic, because the classifier tends to *overfit* the data (i.e. during learning it may incorporate some particular anomalies of the training data that are not present in the general data set overall). As such, our goal is to not only create a model that fits the data, but to fit the data *and* be generalisable.

General approach to classification (Step 2)

2. Classification Step - where the model $y = f(\mathbf{x})$ is used to predict class labels for the given data.
   - First, the predictive accuracy of the classifier is estimated. If we were to use the training set to measure the classifier's accuracy, this estimate would likely be optimistic, because the classifier tends to *overfit* the data (i.e. during learning it may incorporate some particular anomalies of the training data that are not present in the general data set overall). As such, our goal is to not only create a model that fits the data, but to fit the data *and* be generalisable.
   - A *test* set is used, made up of *test tuples* and their associated class labels. They are independent of the training tuples, meaning that they were not used to construct the classifier.
   - The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. We simply compare the associated class label of each test tuple compared with the learned classifier's class prediction for that tuple. If the accuracy of the classifier is considered acceptable, the classifier can be used to classify future data tuples, for which the class label is unknown.

General approach to classification (Confusion matrix)

- For a binary classification problem, we can represent the accuracy testing step using a *confusion matrix*:

		Predicted	
		Class = 1	Class = 0
Actual	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

General approach to classification (Confusion matrix)

- For a binary classification problem, we can represent the accuracy testing step using a *confusion matrix*:

		Predicted	
		Class = 1	Class = 0
Actual	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

Each entry f_{ij} in this table denotes the number or records from class i predicted to be of class j, e.g. f_{01} is the number of records from class 0 incorrectly predicted to be of class 1.

General approach to classification (Confusion matrix)

- For a binary classification problem, we can represent the accuracy testing step using a *confusion matrix*:

		Predicted	
		Class = 1	Class = 0
Actual	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

Each entry f_{ij} in this table denotes the number or records from class i predicted to be of class j, e.g. f_{01} is the number of records from class 0 incorrectly predicted to be of class 1. We can then compute accuracy as follows:

$$Accuracy = \frac{\textit{Number of correct predictions}}{\textit{Total number of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}.$$

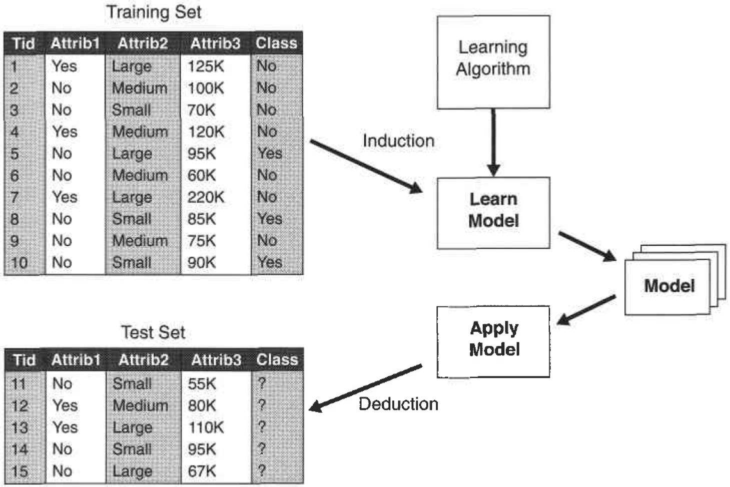
General approach to classification (Error rate)

- Equivalently, the performance of a model can be expressed in terms of its *error rate*, which is given as follows:

$$Error\ rate = \frac{\textit{Number of wrong predictions}}{\textit{Total number of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}.$$

- Most classification algorithms seek models that attain the highest accuracy, or equivalently, the lowest error rate when applied to the test set.
- See the Week 7 Lab Notebook for a brief discussion on other ways to evaluate accuracy.

Summary



Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Decision tree classifiers

- Decision tree classifers are a method of classification by which we employ a flowchart-like tree structure, where each internal node denotes a test on an attribute, and each branch represents an outcome of the test, and each terminal node holds a class label.

Decision tree classifiers

- Decision tree classifers are a method of classification by which we employ a flowchart-like tree structure, where each internal node denotes a test on an attribute, and each branch represents an outcome of the test, and each terminal node holds a class label.
- For example, consider the animal table shown earlier and suppose we wish to distinguish mammals from non-mammals. Perhaps we have a new species and want to decide if it is a mammal.

Decision tree classifiers

- Decision tree classifers are a method of classification by which we employ a flowchart-like tree structure, where each internal node denotes a test on an attribute, and each branch represents an outcome of the test, and each terminal node holds a class label.
- For example, consider the animal table shown earlier and suppose we wish to distinguish mammals from non-mammals. Perhaps we have a new species and want to decide if it is a mammal.
  - The first question we may ask is whether the species is cold- or warm-blooded. If it is cold-blooded, then it is definitely not a mammal.
  - Otherwise, it is either a bird or a mammal.

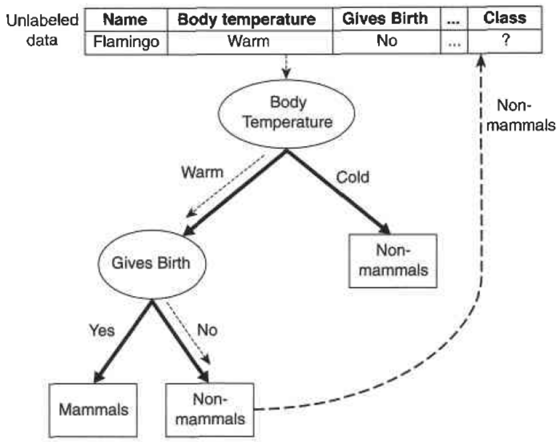
Decision tree classifiers

- Decision tree classifers are a method of classification by which we employ a flowchart-like tree structure, where each internal node denotes a test on an attribute, and each branch represents an outcome of the test, and each terminal node holds a class label.
- For example, consider the animal table shown earlier and suppose we wish to distinguish mammals from non-mammals. Perhaps we have a new species and want to decide if it is a mammal.
    - The first question we may ask is whether the species is cold- or warm-blooded. If it is cold-blooded, then it is definitely not a mammal.
    - Otherwise, it is either a bird or a mammal.
    - Suppose the species is warm-blooded. The next question we may ask is whether the females of the species gives birth to their young.
    - Those that do are mammals, while those that do not are either birds, or egg-laying mammals such as the platypus or spiny anteater.

Decision tree classifiers

- Decision tree classifers are a method of classification by which we employ a flowchart-like tree structure, where each internal node denotes a test on an attribute, and each branch represents an outcome of the test, and each terminal node holds a class label.
- For example, consider the animal table shown earlier and suppose we wish to distinguish mammals from non-mammals. Perhaps we have a new species and want to decide if it is a mammal.
  - The first question we may ask is whether the species is cold- or warm-blooded. If it is cold-blooded, then it is definitely not a mammal.
  - Otherwise, it is either a bird or a mammal.
  - Suppose the species is warm-blooded. The next question we may ask is whether the females of the species gives birth to their young.
  - Those that do are mammals, while those that do not are either birds, or egg-laying mammals such as the platypus or spiny anteater.
  - We therefore ask if the species has mammary glands. If they do, then they are definitely mammals. Otherwise, they are birds.

Mammal classification example



Name	Body temperature	Gives Birth	...	Class
Flamingo	Warm	No	...	?

Different types of nodes

In this way we construct a decision tree with three types of nodes:

- Root node
  These have no incoming links and zero or more outgoing links. They are the topmost nodes and are linked with the first attribute test condition.

Different types of nodes

In this way we construct a decision tree with three types of nodes:

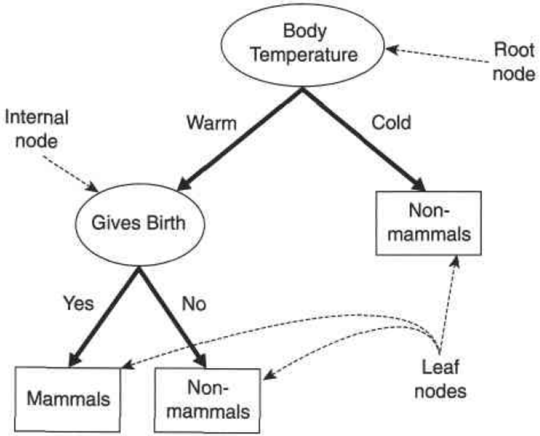
- Root node
  These have no incoming links and zero or more outgoing links. They are the topmost nodes and are linked with the first attribute test condition.
- Branch/Internal node
  These have exactly one incoming link and two or more outgoing links.

Different types of nodes

In this way we construct a decision tree with three types of nodes:

- Root node
  These have no incoming links and zero or more outgoing links. They are the topmost nodes and are linked with the first attribute test condition.
- Branch/Internal node
  These have exactly one incoming link and two or more outgoing links.
- Leaf/Terminal node
  These have exactly one incoming link and no outgoing links, and are associated with a class label. They are linked with an attribute test condition.

The mammal example (again!)



Greedy algorithms and tree pruning

Our intention: Find the optimal decision tree.

Problem: Due to the exponential size of the search space, this is often a computationally expensive task.

Greedy algorithms and tree pruning

Our intention: Find the optimal decision tree.

Problem: Due to the exponential size of the search space, this is often a computationally expensive task.

Solution: Employ a greedy strategy, growing the decision tree in a top-down fashion.

N.B. When decision trees are built, many of the branches may reflect noise or outliers in the training data. *Tree pruning* attempts to identify and remove such branches, with the goal of improving classification accuracy on unseen data.

Hunt’s Algorithm: Generate decision tree

Generate a decision tree from the training tuples of data partition, D.

Input:

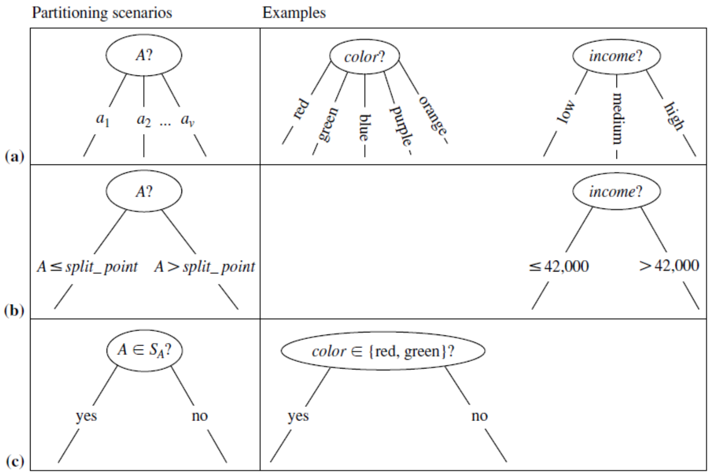
- Data partition, D, which is a set of training tuples and their associated class labels;
- attribute_list, the set of candidate attributes;
- Attribute_selection_method, a procedure to determine the splitting criterion that ‘best’ partitions the data tuples into individual classes. This criterion consists of a splitting_attribute and, possibly, either a *split point* or *splitting subset*.

Output: A decision tree.

Method

```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C, then
(3)      return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let $D_j$ be the set of data tuples in D satisfying outcome j; // a partition
(12)     if $D_j$ is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree($D_j$, attribute_list) to node N;
     endfor
(15) return N;
```

Possible splitting criterion

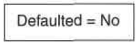


Example (Loan borrower data)

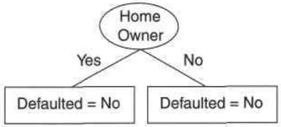
Suppose we have the following training set:

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125k	No
2	No	Married	100k	No
3	No	Single	70k	No
4	Yes	Married	120k	No
5	No	Divorced	95k	Yes
6	No	Married	60k	No
7	Yes	Divorced	220k	No
8	No	Single	85k	Yes
9	No	Married	75k	No
10	No	Single	90k	Yes

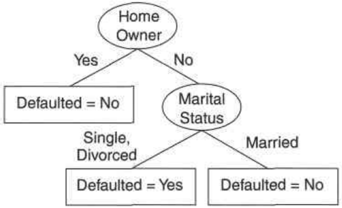
Example (Decision tree)



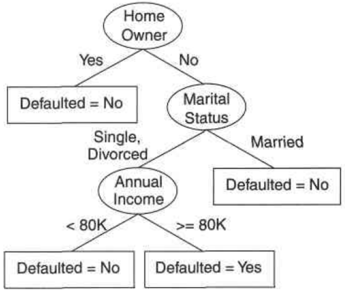
(a)



(b)



(c)



(d)

Example (Discussion)

- We start at the single leaf node in Figure (a). This node is labelled `Defaulted = No` since the majority of borrowers did not default on their loan payments.
- The training error of this tree is 30% as three out of ten training instances have the class label `Defaulted = Yes`. The leaf node than therefore be expanded further as it contains instances from more than one class.

Example (Discussion)

- We start at the single leaf node in Figure (a). This node is labelled `Defaulted = No` since the majority of borrowers did not default on their loan payments.
- The training error of this tree is 30% as three out of ten training instances have the class label `Defaulted = Yes`. The leaf node than therefore be expanded further as it contains instances from more than one class.
- Let `Home Owner` be the attribute chosen to split the training instances. The resulting binary split on the `Home Owner` attribute is then shown in Figure (b).
- All training instances for which `Home Owner = Yes` are propagated to the left, and the rest to the right child. Hunt's algorithm is then recursively applied to each child.
- The left child becomes a leaf node labelled `Defaulted = No` since all instances associated with this node have identical class label `Defaulted = No`. The right child has instances from each label and so we need to further split this node.

Example (Discussion)

- We start at the single leaf node in Figure (a). This node is labelled `Defaulted = No` since the majority of borrowers did not default on their loan payments.
- The training error of this tree is 30% as three out of ten training instances have the class label `Defaulted = Yes`. The leaf node than therefore be expanded further as it contains instances from more than one class.
- Let `Home Owner` be the attribute chosen to split the training instances. The resulting binary split on the `Home Owner` attribute is then shown in Figure (b).
- All training instances for which `Home Owner = Yes` are propagated to the left, and the rest to the right child. Hunt's algorithm is then recursively applied to each child.
- The left child becomes a leaf node labelled `Defaulted = No` since all instances associated with this node have identical class label `Defaulted = No`. The right child has instances from each label and so we need to further split this node.
- The resulting subtrees after recursively expanding the right child are shown in Figures (c)-(d).

Attribute selection measures

- There are several measures that can be used to determine the suitability of an attribute test condition.
- These measures try to give preference to attribute test conditions that partition the training instances into *purer* subsets in the child nodes, which mostly have the same class labels.

Attribute selection measures

- There are several measures that can be used to determine the suitability of an attribute test condition.
- These measures try to give preference to attribute test conditions that partition the training instances into *purer* subsets in the child nodes, which mostly have the same class labels.
- In general, larger trees are less desirable as they are more susceptible to model overfitting, a condition that may degrade the classification performance on unseen instances. They are also difficult to interpret and incur more training and test time compared to smaller trees.
- A big part of this is avoiding impure nodes containing training instances from multiple classes.

Impurity measure for a single node

- The impurity of a node measures how dissimilar the class labels are for the data instances belonging to a common node. We can use several measures to evaluate the impurity of a node t, for instance:

Impurity measure for a single node

- The impurity of a node measures how dissimilar the class labels are for the data instances belonging to a common node. We can use several measures to evaluate the impurity of a node t, for instance:

$$\begin{aligned} Entropy &= -\sum_{i=0}^{c-1} p_i(t)\log_2(p_i(t)), \\ Gini\ index &= 1-\sum_{i=0}^{c-1}(p_i(t))^2, \\ Classification\ error &= 1-max_i[p_i(t)], \end{aligned}$$

where $p_i(t)$ is the relative frequency of training instances that belong to class i at node t, c is the total number of classes, and $0\log_2(0)=0$ in entropy calculations.

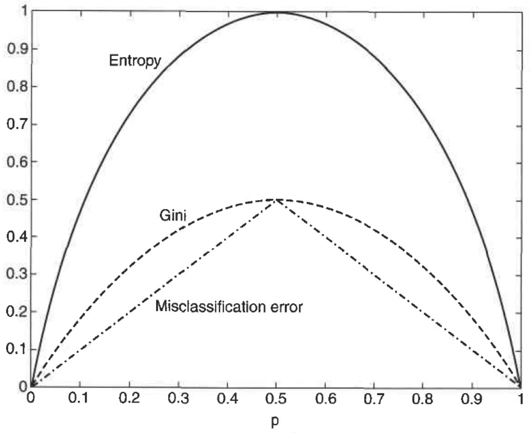
Impurity measure for a single node

- The impurity of a node measures how dissimilar the class labels are for the data instances belonging to a common node. We can use several measures to evaluate the impurity of a node t, for instance:

$$\begin{aligned} Entropy &= -\textstyle\sum_{i=0}^{c-1} p_i(t)\log_2(p_i(t)), \\ Gini\ index &= 1-\textstyle\sum_{i=0}^{c-1}(p_i(t))^2, \\ Classification\ error &= 1-max_i[p_i(t)], \end{aligned}$$

  where $p_i(t)$ is the relative frequency of training instances that belong to class i at node t, c is the total number of classes, and $0\log_2(0)=0$ in entropy calculations.
- All three measures give a zero impurity value if a node contains instances from a single class and maximum impurity if the node has equal proportion of instances from multiple classes.

Comparison of relative magnitudes of impurity measures (binary classification problem)



Comparison of relative magnitudes of impurity measures (binary classification problem)

- In binary classification problems there are only two classes and so $p_0(t) + p_1(t) = 1$.
- The horizontal axis of the figure refers to the fraction of instances that belong to one of the two classes.
- We note that in each case, the measures attain their maximum value when the class distribution is uniform, i.e. $p_0(t) = p_1(t) = 0.5$.
- Likewise, they obtain their minimum when all instances belong to a single class, i.e. $p_0(t)$ or $p_1(t)$ is equal to 1.

Example

- At node 1, we suppose `Class = 0` has a count of 0, while `Class = 1` has a count of 6. Then,

$$\begin{aligned} Entropy &= -(0/6)\log_2(0/6) - (6/6)log_2(6/6) = 0 \\ Gini &= 1 - (0/6)^2 - (6/6)^2 = 0 \\ Error &= 1 - max[0/6,\ 6/6] = 0. \end{aligned}$$

Example

- At node 1, we suppose `Class = 0` has a count of 0, while `Class = 1` has a count of 6. Then,

$$\begin{aligned} Entropy =&\quad -(0/6)\log_2(0/6) - (6/6)log_2(6/6) = 0 \\ Gini =&\quad 1 - (0/6)^2 - (6/6)^2 = 0 \\ Error =&\quad 1 - max[0/6,\ 6/6] = 0. \end{aligned}$$

- At node 2, we suppose `Class = 0` has a count of 1, while `Class = 1` has a count of 5. Then,

$$\begin{aligned} Entropy =&\quad -(1/6)\log_2(1/6) - (5/6)log_2(5/6) = 0.650 \\ Gini =&\quad 1 - (1/6)^2 - (5/6)^2 = 0.278 \\ Error =&\quad 1 - max[1/6,\ 5/6] = 0.167. \end{aligned}$$

Example continued

- At node 3, we suppose `Class = 0` has a count of 3, while `Class = 1` has a count of 3. Then,

$$
\begin{aligned}
Entropy &= \quad -(3/6)\log_2(3/6) - (3/6)log_2(3/6) = 1 \\
Gini &= \quad 1 - (3/6)^2 - (3/6)^2 = 0.5 \\
Error &= \quad 1 - max[3/6,\, 3/6] = 0.5.
\end{aligned}
$$

Based on these computations, node N_1 has the lowest impurity value, followed by N_2 and then N_3.

Consistency among impurity measures

- As the above example demonstrates, there is a consistency among impurity measures, i.e. if a node N_1 has lower entropy than node N_2, then the Gini index and error rate of N_1 will also be lower than that of N_2.
- Nevertheless, the attribute chosen as a splitting criterion by the impurity measure can still be different.

Collective impurity of child nodes

- Consider now an attribute test condition that splits a node containing N training instances into k children, $\{v_1, v_2, \ldots, v_k\}$.
- Each child node represents a partition of the data resulting from one of the k outcomes of the attribute test condition.

Collective impurity of child nodes

- Consider now an attribute test condition that splits a node containing N training instances into k children, $\{v_1, v_2, \ldots, v_k\}$.
- Each child node represents a partition of the data resulting from one of the k outcomes of the attribute test condition.
- We make the following definitions:
  - Let $N(v_j)$ be the number of training instances associated with a child node v_j.
  - Let the impurity value of v_j be $I(v_j)$.

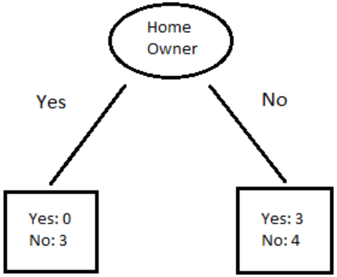
Collective impurity of child nodes

- Consider now an attribute test condition that splits a node containing N training instances into k children, $\{v_1, v_2, \ldots, v_k\}$.
- Each child node represents a partition of the data resulting from one of the k outcomes of the attribute test condition.
- We make the following definitions:
  - Let $N(v_j)$ be the number of training instances associated with a child node v_j.
  - Let the impurity value of v_j be $I(v_j)$.
- Since a training instance in the parent node reaches node v_j for a fraction of $N(v_j)/N$ times, the collective impurity of the child nodes can be computed by taking a weighted sum of the impurities of the child nodes. This is given as follows:

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

Example (Loan borrower data)

First, we can split the `Home Owner` attribute, generating two child nodes as follows:



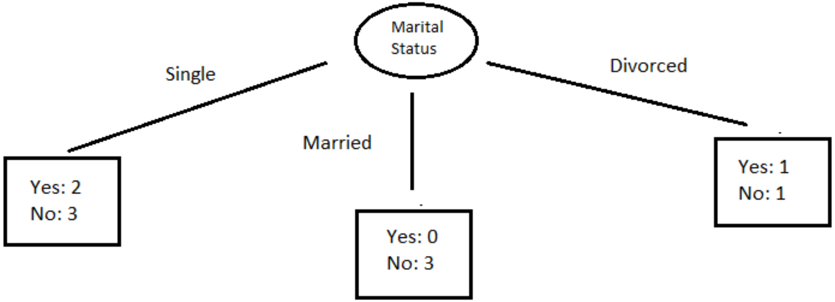
Example (Computations)

We can then compute the weighted entropy as follows:

$$\begin{aligned}
I(\texttt{Home Owner = Yes}) &= -\tfrac{0}{3}\log_2(\tfrac{0}{3}) - \tfrac{3}{3}\log_2(\tfrac{3}{3}) = 0 \\
I(\texttt{Home Owner = No}) &= -\tfrac{3}{7}\log_2(\tfrac{3}{7}) - \tfrac{4}{7}\log_2(\tfrac{4}{7}) = 0.985 \\
I(\texttt{Home Owner}) &= \tfrac{3}{10} \times 0 + \tfrac{7}{10} \times 0.985 = 0.690.
\end{aligned}$$

Example (Alternative split)

Alternatively, we can split on `Marital Status`, as follows:



Example (Computations)

This leads to three child nodes with weighted entropy given by

$$\begin{aligned}
I(\texttt{Marital Status = Single}) &= -\tfrac{2}{5}\log_2(\tfrac{2}{5}) - \tfrac{3}{5}\log_2(\tfrac{3}{5}) = 0.971 \\
I(\texttt{Marital Status = Married}) &= -\tfrac{0}{3}\log_2(\tfrac{0}{3}) - \tfrac{3}{3}\log_2(\tfrac{3}{3}) = 0 \\
I(\texttt{Marital Status = Divorced}) &= -\tfrac{1}{2}\log_2(\tfrac{1}{2}) - \tfrac{1}{2}\log_2(\tfrac{1}{2}) = 1 \\
I(\texttt{Marital Status}) &= \tfrac{5}{10} \times 0.971 + \tfrac{3}{10} \times 0 + \tfrac{2}{10} \times 1 = 0.686.
\end{aligned}$$

Thus, `Marital Status` has a lower weighted entropy than `Home Owner`.

How suitable is an attribute test condition?

Answer: Compare the degree of impurity of the parent node (before splitting) with the weighted degree of impurity of the child nodes (after splitting).

What we want: The larger the difference, the better the test condition.

How suitable is an attribute test condition?

Answer: Compare the degree of impurity of the parent node (before splitting) with the weighted degree of impurity of the child nodes (after splitting).

What we want: The larger the difference, the better the test condition.

- We call this difference (denoted Δ) the *gain* in purity of an attribute test condition and we define it as follows:

$$\Delta = I(parent) - I(children),$$

  where

  - $I(parent)$ is the impurity of a node before splitting, and
  - $I(children)$ is the weighted impurity after splitting.

Information gain

- When entropy is used as the impurity measure, the difference in entropy is commonly known as *information gain* and is denoted by Δ_{info}.
- $\Delta \geq 0$ for any reasonable measure, such as those presented above.
- The higher the gain, the purer are the classes in the child nodes relative to the parent node.
- Furthermore, maximising the gain at a given node is equivalent to minimising the weighted impurity measure of its children since $I(parent)$ is the same for all candidate attribute test conditions.

Example (Loan borrower)

- Consider the two candidate splits shown in the previous example: `Home Owner` and `Marital Status`.

Example (Loan borrower)

- Consider the two candidate splits shown in the previous example: `Home Owner` and `Marital Status`.
- The initial class distribution at the parent node is $(0.3, 0.7)$ since there are three `Yes` and seven `No` classes in the training data. Thus,

$$I(parent) = -\frac{3}{10}\log_2(\frac{3}{10}) - \frac{7}{10}\log_2(\frac{7}{10}) = 0.881.$$

Example (Loan borrower)

- Consider the two candidate splits shown in the previous example: `Home Owner` and `Marital Status`.
- The initial class distribution at the parent node is $(0.3, 0.7)$ since there are three `Yes` and seven `No` classes in the training data. Thus,

$$I(parent) = -\frac{3}{10}\log_2(\frac{3}{10}) - \frac{7}{10}\log_2(\frac{7}{10}) = 0.881.$$

The information gains for `Home Owner` and `Marital Status` are each given by

$$\begin{aligned}\Delta_{info}(\texttt{Home Owner}) &= \quad 0.881 - 0.690 = 0.191\\ \Delta_{info}(\texttt{Marital Status}) &= \quad 0.881 - 0.686 = 0.195.\end{aligned}$$

Conclusion: the information gain for `Marital Status` is higher (due to its lower weighted entropy), and will therefore be considered for splitting.

Example (Sticking with the loan borrower)

This time we consider building a decision tree using only binary splits and the Gini index as the impurity measure. We consider the following four candidate splitting criteria for the `Home Owner` and `Marital Status` attributes:

1. `Home Owner`: `Yes` or `No`
2. `Marital Status`: `Single` or `Married/Divorced`
3. `Marital Status`: `Single/Married` or `Divorced`
4. `Marital Status`: `Single/Divorced` or `Married`

Example (Breakdown of the data)

Using the data from earlier, we have the following tables:

(1)	N_1	N_2
No	3	4
Yes	0	3
$Gini$	0.343	

(2)	N_1	N_2
No	3	4
Yes	2	1
$Gini$	0.400	

(3)	N_1	N_2
No	6	1
Yes	2	1
$Gini$	0.400	

(4)	N_1	N_2
No	4	3
Yes	3	0
$Gini$	0.343	

Example (Putting it all together)

- Since there are 3 borrowers in the training set who defaulted and 7 others who repaid their loan, the Gini index of the parent node before splitting is,

$$1 = \left(\frac{3}{10}\right)^2 - \left(\frac{7}{10}\right)^2 = 0.420.$$

Example (Putting it all together)

- Since there are 3 borrowers in the training set who defaulted and 7 others who repaid their loan, the Gini index of the parent node before splitting is,

$$1 = \left(\frac{3}{10}\right)^2 - \left(\frac{7}{10}\right)^2 = 0.420.$$

- If `Home Owner` s chosen as the splitting attribute, the Gini index for the child nodes N_1 and N_2 is 0 and 0.490 respectively. The weighted average Gini index for the children is then given by,

$$\frac{3}{10} \times 0 + \frac{7}{10} \times 0.490 = 0.343.$$

Example (Putting it all together)

- Since there are 3 borrowers in the training set who defaulted and 7 others who repaid their loan, the Gini index of the parent node before splitting is,

$$1 = \left(\frac{3}{10}\right)^2 - \left(\frac{7}{10}\right)^2 = 0.420.$$

- If `Home Owner` s chosen as the splitting attribute, the Gini index for the child nodes N_1 and N_2 is 0 and 0.490 respectively. The weighted average Gini index for the children is then given by,

$$\frac{3}{10} \times 0 + \frac{7}{10} \times 0.490 = 0.343.$$

  The gain using `Home Owner` as the splitting attribute is then given by

$$\Delta_1 = 0.420 - 0.343 = 0.077.$$

Example (Conclusion)

- We can likewise apply a binary split on the `Marital Status`. However, since this is a nominal attribute with three outcomes, there are three possible ways to group the attribute values into a binary split.
- By repeating the arguments above, we obtain the gains given in the tables above.
- Based on these results, `Home Owner` and the last binary split using `Marital Status` are the best candidates since they both produce the lowest weighted average Gini index.

Summary

- We have discussed Hunt's algorithm.
- We have seen measures of impurity:

$$\begin{aligned} Entropy &= -\sum_{i=0}^{c-1} p_i(t)\log_2(p_i(t)), \\ Gini\ index &= 1-\sum_{i=0}^{c-1}(p_i(t))^2, \\ Classification\ error &= 1-max_i[p_i(t)], \end{aligned}$$