Classification

Basic Concepts, Decision Trees, and Model Evaluation
Classification definition

- Given a collection of samples (training set)
  - Each sample contains a set of attributes.
  - Each sample also has a discrete class label.
- Learn a model that predicts class label as a function of the values of the attributes.
- Goal: model should assign class labels to previously unseen samples as accurately as possible.
  - A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.
Stages in a classification task

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>
Examples of classification tasks

- **Two classes**
  - Predicting tumor cells as benign or malignant
  - Classifying credit card transactions as legitimate or fraudulent

- **Multiple classes**
  - Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil
  - Categorizing news stories as finance, weather, entertainment, sports, etc
Classification techniques

- Decision trees
- Rule-based methods
- Logistic regression
- Discriminant analysis
- k-Nearest neighbor (instance-based learning)
- Naïve Bayes
- Neural networks
- Support vector machines
- Bayesian belief networks
Example of a decision tree

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Training data to model: decision tree
Another example of decision tree

There can be more than one tree that fits the same data!
Decision tree classification task

Training Set

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Test Set

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>
Apply model to test data

Start from the root of tree.

Refund

Yes

No

NO

MarSt

Single, Divorced

Married

TaxInc

< 80K

> 80K

NO

YES

NO

Test data

Refund | Marital Status | Taxable Income | Cheat
-------|----------------|----------------|-------
No     | Married       | 80K            | ?     

Apply model to test data

Test data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund

Yes

NO

No

MarSt

Single, Divorced

TaxInc

< 80K

NO

> 80K

YES

NO

Married
Apply model to test data

Test data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

- **Refund**: Yes or No
- **Marital Status (MarSt)**: Single, Divorced or Married
- **Taxable Income (TaxInc)**: < 80K or > 80K
- **Cheat**: Yes or No
Apply model to test data

Test data

Refund | Marital Status | Taxable Income | Cheat
---|---|---|---
No | Married | 80K | ?

Refund

Yes

NO

Refund

No

MarSt

Single, Divorced

Married

TaxInc

< 80K

NO

> 80K

YES
Apply model to test data

Test data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund

Marriage

Married

Married

< 80K

> 80K
Apply model to test data

Test data

Refund | Marital Status | Taxable Income | Cheat
--- | --- | --- | ---
No | Married | 80K | ?

Assign Cheat to “No”
Decision tree classification task

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>
Decision tree induction

- Many algorithms:
  - Hunt’s algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT
Hunt’s algorithm is recursive. General procedure:

- Let $D_t$ be the set of training records that reach a node $t$.
  - If all records in $D_t$ belong to the same class $y_t$, then $t$ is a leaf node labeled as $y_t$.
  - If $D_t$ is an empty set, then $t$ is a leaf node labeled by the default class, $y_d$.
  - If $D_t$ contains records that belong to more than one class, use an attribute test to split the data into smaller subsets, then apply the procedure to each subset.
Applying Hunt’s algorithm

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Tree induction

- Greedy strategy
  - Split the records at each node based on an attribute test that optimizes some chosen criterion.

- Issues
  - Determine how to split the records
    - How to specify structure of split?
    - What is best attribute / attribute value for splitting?
  - Determine when to stop splitting
Tree induction

● Greedy strategy
  – Split the records at each node based on an attribute test that optimizes some chosen criterion.

● Issues
  – Determine how to split the records
    ◆ How to specify structure of split?
    ◆ What is best attribute / attribute value for splitting?
  – Determine when to stop splitting
Specifying structure of split

- Depends on attribute type
  - Nominal
  - Ordinal
  - Continuous (interval or ratio)

- Depends on number of ways to split
  - Binary (two-way) split
  - Multi-way split
Splitting based on nominal attributes

- **Multi-way split**: Use as many partitions as distinct values.

  ![Diagram](image)

  - CarType
    - Family
    - Luxury
    - Sports

- **Binary split**: Divides values into two subsets. Need to find optimal partitioning.

  ![Diagram](image)

  - CarType
    - {Sports, Luxury} → {Family}
    - OR
    - CarType
      - {Family, Luxury} → {Sports}
Splitting based on ordinal attributes

- **Multi-way split**: Use as many partitions as distinct values.

  ![Multi-way split diagram]

- **Binary split**: Divides values into two subsets. Need to find optimal partitioning.

  ![Binary split diagram]

- **What about this split?**

  ![What about this split diagram]
Splitting based on continuous attributes

- Different ways of handling
  - **Discretization** to form an ordinal attribute
    - static – discretize once at the beginning
    - dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - **Threshold decision**: $(A < v)$ or $(A \geq v)$
    - consider all possible split points $v$ and find the one that gives the best split
    - can be more compute intensive
Splitting based on continuous attributes

- Splitting based on threshold decision

(i) Binary split

(ii) Multi-way split
Tree induction

- **Greedy strategy**
  - Split the records at each node based on an attribute test that optimizes some chosen criterion.

- **Issues**
  - Determine how to split the records
    - How to specify structure of split?
    - What is best attribute / attribute value for splitting?
  - Determine when to stop splitting
Determining the best split

Before splitting: 10 records of class 1 (C1)
10 records of class 2 (C2)

Which attribute gives the best split?
Determining the best split

- Greedy approach:
  Nodes with **homogeneous** class distribution are preferred.
- Need a measure of node **impurity**:

<table>
<thead>
<tr>
<th>Class 1</th>
<th>Class 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>

  - Non-homogeneous, high degree of impurity
  - Homogeneous, low degree of impurity
Measures of node impurity

- Gini index
- Entropy
- Misclassification error
Using a measure of impurity to determine best split

Before splitting:

<table>
<thead>
<tr>
<th>Class 1</th>
<th>N00</th>
<th>Class 2</th>
<th>N01</th>
</tr>
</thead>
</table>

**Attribute A?**
- Yes
  - Node N1
    - Class 1: N10
    - Class 2: N11
- No
  - Node N2
    - Class 1: N20
    - Class 2: N21

**Attribute B?**
- Yes
  - Node N3
    - Class 1: N30
    - Class 2: N31
- No
  - Node N4
    - Class 1: N40
    - Class 2: N41

Gain = M0 – M12 vs. M0 – M34

Choose attribute that maximizes gain
Measure of impurity: Gini index

- Gini index for a given node $t$:

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

$p(j \mid t)$ is the relative frequency of class $j$ at node $t$

- Maximum $(1 - 1 / n_c)$ when records are equally distributed among all classes, implying least amount of information ($n_c = \text{number of classes}$).

- Minimum $(0.0)$ when all records belong to one class, implying most amount of information.

<table>
<thead>
<tr>
<th>C1</th>
<th>0</th>
<th>C1</th>
<th>1</th>
<th>C1</th>
<th>2</th>
<th>C1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>6</td>
<td>C2</td>
<td>5</td>
<td>C2</td>
<td>4</td>
<td>C2</td>
<td>3</td>
</tr>
<tr>
<td>Gini=0.000</td>
<td>Gini=0.278</td>
<td>Gini=0.444</td>
<td>Gini=0.500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Examples of computing Gini index

\[ GINI(t) = 1 - \sum_j [p(j | t)]^2 \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

\[
p( C1 ) = 0 / 6 = 0 \quad p( C2 ) = 6 / 6 = 1
\]

\[
Gini = 1 - p( C1 )^2 - p( C2 )^2 = 1 - 0 - 1 = 0
\]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

\[
p( C1 ) = 1 / 6 \quad p( C2 ) = 5 / 6
\]

\[
Gini = 1 - (1 / 6)^2 - (5 / 6)^2 = 0.278
\]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

\[
p( C1 ) = 2 / 6 \quad p( C2 ) = 4 / 6
\]

\[
Gini = 1 - (2 / 6)^2 - (4 / 6)^2 = 0.444
\]
Splitting based on Gini index

- Used in CART, SLIQ, SPRINT.
- When a node $t$ is split into $k$ partitions (child nodes), the quality of split is computed as,

$$ GINI_{\text{split}} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i) $$

where $n_i = \text{number of records at child node } i$

$n = \text{number of records at parent node } t$
Computing Gini index: binary attributes

- Splits into two partitions
- Effect of weighting partitions: favors larger and purer partitions

\[
\text{Gini( N1 )} = 1 - \left(\frac{5}{7}\right)^2 - \left(\frac{2}{7}\right)^2 = 0.408
\]

\[
\text{Gini( N2 )} = 1 - \left(\frac{1}{5}\right)^2 - \left(\frac{4}{5}\right)^2 = 0.320
\]

\[
\text{Gini( children )} = \frac{7}{12} \times 0.408 + \frac{5}{12} \times 0.320 = 0.371
\]
Computing Gini index: categorical attributes

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

**Multi-way split**

<table>
<thead>
<tr>
<th>CarType</th>
<th>Family</th>
<th>Sports</th>
<th>Luxury</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gini</td>
<td>0.393</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Two-way split**

(find best partition of attribute values)

<table>
<thead>
<tr>
<th>CarType</th>
<th>{Sports, Luxury}</th>
<th>{Family}</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Gini</td>
<td>0.400</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CarType</th>
<th>{Sports}</th>
<th>{Family, Luxury}</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Gini</td>
<td>0.419</td>
<td></td>
</tr>
</tbody>
</table>
Computing Gini index: continuous attributes

- Make binary split based on a threshold (splitting) value of attribute
- Number of possible splitting values = (number of distinct values attribute has at that node) - 1
- Each splitting value \( v \) has a count matrix associated with it
  - Class counts in each of the partitions, \( A < v \) and \( A \geq v \)
- Simple method to choose best \( v \)
  - For each \( v \), scan the attribute values at the node to gather count matrix, then compute its Gini index.
  - Computationally inefficient! Repetition of work.

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Computing Gini index: continuous attributes

- For efficient computation, do following for each (continuous) attribute:
  - Sort attribute values.
  - Linearly scan these values, each time updating the count matrix and computing Gini index.
  - Choose split position that has minimum Gini index.

<table>
<thead>
<tr>
<th>Cheat</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxable Income</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>70</td>
<td>75</td>
<td>85</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>120</td>
<td>125</td>
<td>220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
</tr>
<tr>
<td>Yes</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>No</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Gini</td>
<td>0.420</td>
<td>0.400</td>
<td>0.375</td>
<td>0.343</td>
<td>0.417</td>
<td>0.400</td>
<td>0.300</td>
<td>0.343</td>
<td>0.375</td>
<td>0.400</td>
<td>0.420</td>
</tr>
</tbody>
</table>
Comparison among splitting criteria

For a two-class problem:

![Graph showing comparison among splitting criteria](image-url)
Tree induction

- Greedy strategy
  - Split the records at each node based on an attribute test that optimizes some chosen criterion.

- Issues
  - Determine how to split the records
    - How to specify structure of split?
    - What is best attribute / attribute value for splitting?
  - Determine when to stop splitting
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 identical (or very similar) attribute values
  - No remaining basis for splitting.
- Early termination.

Can also prune tree post-induction.
Border between two neighboring regions of different classes is known as decision boundary.

In decision trees, decision boundary segments are always parallel to attribute axes, because test condition involves one attribute at a time.
Classification with decision trees

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy comparable to other classification techniques for many simple data sets

- Disadvantages:
  - Easy to overfit
  - Decision boundary restricted to being parallel to attribute axes
MATLAB interlude

matlab_demo_04.m

Part A
Producing useful models: topics

- Generalization
- Measuring classifier performance
- Overfitting, underfitting
- Validation
Generalization

- Definition: model does a good job of correctly predicting class labels of previously unseen samples.
- Generalization is typically evaluated using a test set of data that was not involved in the training process.
- Evaluating generalization requires:
  - Correct labels for test set are known.
  - A quantitative measure (*metric*) of tendency for model to predict correct labels.

NOTE: Generalization is separate from other performance issues around models, e.g. computational efficiency, scalability.
Generalization of decision trees

- If you make a decision tree deep enough, it can usually do a perfect job of predicting class labels on training set.
  
  \textit{Is this a good thing?}

  \textbf{NO!}

- Leaf nodes do not have to be pure for a tree to generalize well. In fact, it’s often better if they aren’t.

- Class prediction of an impure leaf node is simply the majority class of the records in the node.

- An impure node can also be interpreted as making a probabilistic prediction.
  - Example: \(7 / 10\) class 1 means \(p(1) = 0.7\)
Metrics for classifier performance

- **Accuracy**
  
  \[ a = \text{number of test samples with label correctly predicted} \]
  
  \[ b = \text{number of test samples with label incorrectly predicted} \]
  
  \[ \text{accuracy} = \frac{a}{a + b} \]

  example
  
  - 75 samples in test set
  - correct class label predicted for 62 samples
  - wrong class label predicted for 13 samples
  - accuracy = 62 / 75 = 0.827
Metrics for classifier performance

- Limitations of accuracy as a metric
  - Consider a two-class problem
    - number of class 1 test samples = 9990
    - number of class 2 test samples = 10
  - What if model predicts everything to be class 1?
    - accuracy is extremely high: 9990 / 10000 = 99.9%
    - but model will never correctly predict any sample in class 2
    - in this case accuracy is misleading and does not give a good picture of model quality
### Metrics for classifier performance

- **Confusion matrix**

  example (continued from two slides back)

<table>
<thead>
<tr>
<th>predicted class</th>
<th>actual class</th>
</tr>
</thead>
<tbody>
<tr>
<td>class 1</td>
<td>class 2</td>
</tr>
<tr>
<td>class 1</td>
<td>21</td>
</tr>
<tr>
<td>class 2</td>
<td>7</td>
</tr>
</tbody>
</table>

  accuracy = \( \frac{21 + 41}{21 + 6 + 7 + 41} = \frac{62}{75} \)
# Metrics for classifier performance

- **Confusion matrix**
  - derived metrics (for two classes)

<table>
<thead>
<tr>
<th>predicted class</th>
<th>actual class</th>
</tr>
</thead>
<tbody>
<tr>
<td>class 1 (negative)</td>
<td>21 (TN)</td>
</tr>
<tr>
<td>class 2 (positive)</td>
<td>7 (FP)</td>
</tr>
</tbody>
</table>

*TN*: true negatives  
*FN*: false negatives  
*FP*: false positives  
*TP*: true positives
Metrics for classifier performance

- Confusion matrix
  - derived metrics (for two classes)

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>Actual class</th>
</tr>
</thead>
<tbody>
<tr>
<td>class 1 (negative)</td>
<td>21 (TN)</td>
</tr>
<tr>
<td>class 2 (positive)</td>
<td>7 (FP)</td>
</tr>
</tbody>
</table>

sensitivity = \( \frac{TP}{TP + FN} \)  

specificity = \( \frac{TN}{TN + FP} \)
MATLAB interlude

matlab_demo_04.m

Part B
Underfitting and overfitting

- Fit of model to training and test sets is controlled by:

  - model capacity (≈ number of parameters)
    - example: number of nodes in decision tree
  - stage of optimization
    - example: number of iterations in a gradient descent optimization
Underfitting and overfitting

The graph shows the error rate (%) on the y-axis and the number of nodes on the x-axis. The optimal fit is between underfitting and overfitting, where the error rate is minimized for both the training set and the test set.
Sources of overfitting: noise

Decision boundary distorted by noise point
Sources of overfitting: insufficient examples

- Lack of data points in lower half of diagram makes it difficult to correctly predict class labels in that region.
  - Insufficient training records in the region causes decision tree to predict the test examples using other training records that are irrelevant to the classification task.
Occam’s Razor

- Given two models with similar generalization errors, one should prefer the simpler model over the more complex model.

- For complex models, there is a greater chance it was fitted accidentally by errors in data.

- Model complexity should therefore be considered when evaluating a model.
Decision trees: addressing overfitting

- **Pre-pruning (early stopping rules)**
  - 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
  - Early stopping conditions (more restrictive):
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using $\chi^2$ test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
Decision trees: addressing overfitting

- **Post-pruning**
  - Grow full decision tree
  - Trim nodes of full tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree
  - Can use various measures of generalization error for post-pruning (see textbook)
Example of post-pruning

Training error (before splitting) = 10/30
Pessimistic error = (10 + 0.5)/30 = 10.5/30
Training error (after splitting) = 9/30
Pessimistic error (after splitting)
= (9 + 4 \times 0.5)/30 = 11/30
PRUNE!
MNI ST database of handwritten digits

- Gray-scale images, 28 x 28 pixels.
- 10 classes, labels 0 through 9.
- Training set of 60,000 samples.
- Test set of 10,000 samples.
- Subset of a larger set available from NIST.
- Each digit size-normalized and centered in a fixed-size image.
- Good database for people who want to try machine learning techniques on real-world data while spending minimal effort on preprocessing and formatting.
- We will use a subset of MNIST with 5000 training and 1000 test samples and formatted for MATLAB (mnistabridged.mat).
MATLAB interlude

matlab_demo_04.m

Part C
Every (useful) model offers choices in one or more of:
- model structure
  - e.g. number of nodes and connections
- types and numbers of parameters
  - e.g. coefficients, weights, etc.

Furthermore, the values of most of these parameters will be modified (optimized) during the model training process.

Suppose the test data somehow influences the choice of model structure, or the optimization of parameters ...
Model validation

The one commandment of machine learning

TRAIN on TEST
Divide available labeled data into three sets:

- **Training set:**
  - Used to drive model building and parameter optimization

- **Validation set**
  - Used to gauge status of generalization error
  - Results can be used to guide decisions during training process
    - typically used mostly to optimize small number of high-level meta
      parameters, e.g. regularization constants; number of gradient descent
      iterations

- **Test set**
  - Used only for final assessment of model quality, after training +
    validation completely finished
Validation strategies

- Holdout
- Cross-validation
- Leave-one-out (LOO)

- Random vs. block folds
  - Use random folds if data are independent samples from an underlying population
  - Must use block folds if any there is any spatial or temporal correlation between samples
Validation strategies

● Holdout
  – Pro: results in single model that can be used directly in production
  – Con: can be wasteful of data
  – Con: a single static holdout partition has the potential to be unrepresentative and statistically misleading

● Cross-validation and leave-one-out (LOO)
  – Con: do not lead directly to a single production model
  – Pro: use all available data for evaluation
  – Pro: many partitions of data, helps average out statistical variability
Validation: example of block folds