Decision Trees
An Early Classifier

Jason Corso

SUNY at Buffalo

Introduction to Non-Metric Methods

- We cover such problems involving **nominal data** in this chapter—that is, data that are discrete and without any natural notion of similarity or even ordering.
  - For example (DHS), some teeth are small and fine (as in baleen whales) for straining tiny prey from the sea; others (as in sharks) come in multiple rows; other sea creatures have tusks (as in walruses), yet others lack teeth altogether (as in squid). There is no clear notion of similarity for this information about teeth.

Introduction to Non-Metric Methods

- We cover such problems involving **nominal data** in this chapter—that is, data that are discrete and without any natural notion of similarity or even ordering.
    - For example (DHS), some teeth are small and fine (as in baleen whales) for straining tiny prey from the sea; others (as in sharks) come in multiple rows; other sea creatures have tusks (as in walruses), yet others lack teeth altogether (as in squid). There is no clear notion of similarity for this information about teeth.
- Most of the other methods we study will involve real-valued feature vectors with clear metrics.
- We may also consider problems involving data tuples and data strings. And for recognition of these, decision trees and string grammars, respectively.

20 Questions

- I am thinking of a person. Ask me up to 20 yes/no questions to determine who this person is that I am thinking about.
  - Consider your questions wisely...

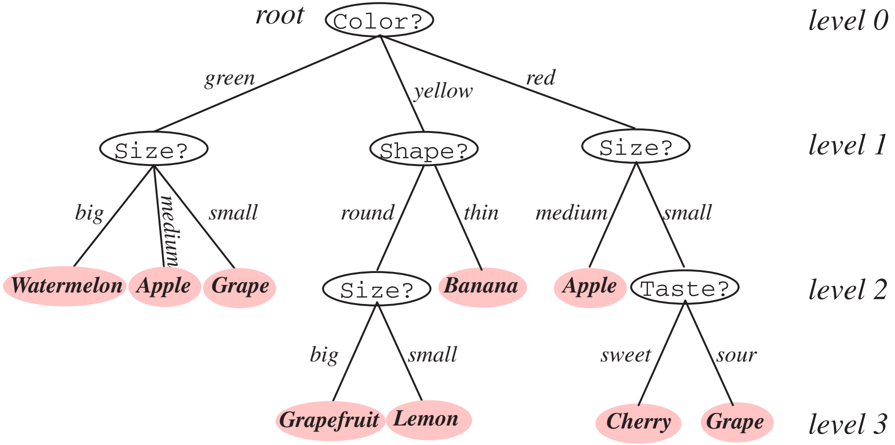
20 Questions

- I am thinking of a person. Ask me up to 20 yes/no questions to determine who this person is that I am thinking about.
  - Consider your questions wisely...
- How did you ask the questions?
- What underlying measure led you the questions, if any?

20 Questions

- I am thinking of a person. Ask me up to 20 yes/no questions to determine who this person is that I am thinking about.
  - Consider your questions wisely...
- How did you ask the questions?
- What underlying measure led you the questions, if any?
- Most importantly, iterative yes/no questions of this sort require no metric and are well suited for nominal data.

These sequence of questions are a decision tree...



Decision Trees 101

- The **root node** of the tree, displayed at the top, is connected to successive **branches** to the other nodes.

Decision Trees 101

- The **root node** of the tree, displayed at the top, is connected to successive **branches** to the other nodes.
- The connections continue until the **leaf nodes** are reached, implying a decision.

Decision Trees 101

- The **root node** of the tree, displayed at the top, is connected to successive **branches** to the other nodes.
- The connections continue until the **leaf nodes** are reached, implying a decision.
- The classification of a particular pattern begins at the root node, which queries a particular property (selected during tree learning).

Decision Trees 101

- The **root node** of the tree, displayed at the top, is connected to successive **branches** to the other nodes.
- The connections continue until the **leaf nodes** are reached, implying a decision.
- The classification of a particular pattern begins at the root node, which queries a particular property (selected during tree learning).
- The links off of the root node correspond to different possible values of the property.

Decision Trees 101

- The **root node** of the tree, displayed at the top, is connected to successive **branches** to the other nodes.
- The connections continue until the **leaf nodes** are reached, implying a decision.
- The classification of a particular pattern begins at the root node, which queries a particular property (selected during tree learning).
- The links off of the root node correspond to different possible values of the property.
- We follow the link corresponding to the appropriate value of the pattern and continue to a new node, at which we check the next property. And so on.

Decision Trees 101

- The **root node** of the tree, displayed at the top, is connected to successive **branches** to the other nodes.
- The connections continue until the **leaf nodes** are reached, implying a decision.
- The classification of a particular pattern begins at the root node, which queries a particular property (selected during tree learning).
- The links off of the root node correspond to different possible values of the property.
- We follow the link corresponding to the appropriate value of the pattern and continue to a new node, at which we check the next property. And so on.
- Decision trees have a particularly high degree of interpretability.

When to Consider Decision Trees

- Instances are wholly or partly described by attribute-value pairs.
- Target function is discrete valued.
- Disjunctive hypothesis may be required.
- Possibly noisy training data.
- Examples
  - Equipment or medical diagnosis.
  - Credit risk analysis.
  - Modeling calendar scheduling preferences.

CART for Decision Tree Learning

- Assume we have a set of $\mathcal{D}$ labeled training data and we have decided on a set of properties that can be used to discriminate patterns.

CART for Decision Tree Learning

- Assume we have a set of $\mathcal{D}$ labeled training data and we have decided on a set of properties that can be used to discriminate patterns.
- Now, we want to learn how to organize these properties into a decision tree to maximize accuracy.

CART for Decision Tree Learning

- Assume we have a set of $\mathcal{D}$ labeled training data and we have decided on a set of properties that can be used to discriminate patterns.
- Now, we want to learn how to organize these properties into a decision tree to maximize accuracy.
- Any decision tree will progressively split the data into subsets.

CART for Decision Tree Learning

- Assume we have a set of $\mathcal{D}$ labeled training data and we have decided on a set of properties that can be used to discriminate patterns.
- Now, we want to learn how to organize these properties into a decision tree to maximize accuracy.
- Any decision tree will progressively split the data into subsets.
- If at any point all of the elements of a particular subset are of the same category, then we say this node is **pure** and we can stop splitting.

CART for Decision Tree Learning

- Assume we have a set of $\mathcal{D}$ labeled training data and we have decided on a set of properties that can be used to discriminate patterns.
- Now, we want to learn how to organize these properties into a decision tree to maximize accuracy.
- Any decision tree will progressively split the data into subsets.
- If at any point all of the elements of a particular subset are of the same category, then we say this node is **pure** and we can stop splitting.
- Unfortunately, this rarely happens and we have to decide between whether to stop splitting and accept an imperfect decision or instead to select another property and grow the tree further.

- The basic CART strategy to recursively defining the tree is the following: **Given the data represented at a node, either declare that node to be a leaf or find another property to use to split the data into subsets.**

- The basic CART strategy to recursively defining the tree is the following: **Given the data represented at a node, either declare that node to be a leaf or find another property to use to split the data into subsets.**
- There are 6 general kinds of questions that arise:

- The basic CART strategy to recursively defining the tree is the following: **Given the data represented at a node, either declare that node to be a leaf or find another property to use to split the data into subsets.**
- There are 6 general kinds of questions that arise:
  1. How many branches will be selected from a node?

- The basic CART strategy to recursively defining the tree is the following: **Given the data represented at a node, either declare that node to be a leaf or find another property to use to split the data into subsets.**
- There are 6 general kinds of questions that arise:
  1. How many branches will be selected from a node?
  2. Which property should be tested at a node?

- The basic CART strategy to recursively defining the tree is the following: **Given the data represented at a node, either declare that node to be a leaf or find another property to use to split the data into subsets.**
- There are 6 general kinds of questions that arise:
    1. How many branches will be selected from a node?
    2. Which property should be tested at a node?
    3. When should a node be declared a leaf?

- The basic CART strategy to recursively defining the tree is the following: **Given the data represented at a node, either declare that node to be a leaf or find another property to use to split the data into subsets.**
- There are 6 general kinds of questions that arise:
  1. How many branches will be selected from a node?
  2. Which property should be tested at a node?
  3. When should a node be declared a leaf?
  4. How can we prune a tree once it has become too large?

- The basic CART strategy to recursively defining the tree is the following: **Given the data represented at a node, either declare that node to be a leaf or find another property to use to split the data into subsets.**
- There are 6 general kinds of questions that arise:
  1. How many branches will be selected from a node?
  2. Which property should be tested at a node?
  3. When should a node be declared a leaf?
  4. How can we prune a tree once it has become too large?
  5. If a leaf node is impure, how should the category be assigned?

- The basic CART strategy to recursively defining the tree is the following: **Given the data represented at a node, either declare that node to be a leaf or find another property to use to split the data into subsets.**
- There are 6 general kinds of questions that arise:
  1. How many branches will be selected from a node?
  2. Which property should be tested at a node?
  3. When should a node be declared a leaf?
  4. How can we prune a tree once it has become too large?
  5. If a leaf node is impure, how should the category be assigned?
  6. How should missing data be handled?

Number of Splits

- The number of splits at a node, or its **branching factor** B, is generally set *by the designer* (as a function of the way the test is selected) and can vary throughout the tree.

Number of Splits

- The number of splits at a node, or its **branching factor** B, is generally set *by the designer* (as a function of the way the test is selected) and can vary throughout the tree.
- Note that any split with a factor greater than 2 can easily be converted into a sequence of binary splits.

Number of Splits

- The number of splits at a node, or its **branching factor** B, is generally set *by the designer* (as a function of the way the test is selected) and can vary throughout the tree.
- Note that any split with a factor greater than 2 can easily be converted into a sequence of binary splits.
- So, DHS focuses on only binary tree learning.

Number of Splits

- The number of splits at a node, or its **branching factor** B, is generally set *by the designer* (as a function of the way the test is selected) and can vary throughout the tree.
- Note that any split with a factor greater than 2 can easily be converted into a sequence of binary splits.
- So, DHS focuses on only binary tree learning.
- But, we note that in certain circumstances for learning and inference, the selection of a test at a node or its inference may be computationally expensive and a 3- or 4-way split may be more desirable for computational reasons.

Query Selection and Node Impurity

- The fundamental principle underlying tree creation is that of simplicity: **we prefer decisions that lead to a simple, compact tree with few nodes**.

Query Selection and Node Impurity

- The fundamental principle underlying tree creation is that of simplicity: **we prefer decisions that lead to a simple, compact tree with few nodes**.
- We seek a property query T at each node N that makes the data reaching the immediate descendant nodes as "pure" as possible.

Query Selection and Node Impurity

- The fundamental principle underlying tree creation is that of simplicity: **we prefer decisions that lead to a simple, compact tree with few nodes**.
- We seek a property query T at each node N that makes the data reaching the immediate descendant nodes as "pure" as possible.
- Let $i(N)$ denote the impurity of a node N.

Query Selection and Node Impurity

- The fundamental principle underlying tree creation is that of simplicity: **we prefer decisions that lead to a simple, compact tree with few nodes**.
- We seek a property query T at each node N that makes the data reaching the immediate descendant nodes as “pure” as possible.
- Let $i(N)$ denote the impurity of a node N.
- In all cases, we want $i(N)$ to be 0 if all of the patterns that reach the node bear the same category, and to be large if the categories are equally represented.

Query Selection and Node Impurity

- The fundamental principle underlying tree creation is that of simplicity: **we prefer decisions that lead to a simple, compact tree with few nodes**.
- We seek a property query T at each node N that makes the data reaching the immediate descendant nodes as "pure" as possible.
- Let $i(N)$ denote the impurity of a node N.
- In all cases, we want $i(N)$ to be 0 if all of the patterns that reach the node bear the same category, and to be large if the categories are equally represented.
- **Entropy impurity** is the most popular measure:

$$i(N) = -\sum_{j} P(\omega_j) \log P(\omega_j) \quad . \tag{1}$$

It will be minimized for a node that has elements of only one class (pure).

- For the two-category case, a useful definition of impurity is that **variance impurity**:

$$i(N) = P(\omega_1)P(\omega_2) \tag{2}$$

- For the two-category case, a useful definition of impurity is that **variance impurity**:

$$i(N) = P(\omega_1)P(\omega_2) \tag{2}$$

- Its generalization to the multi-class is the **Gini impurity**:

$$i(N) = \sum_{i \neq j} P(\omega_i)P(\omega_j) = \frac{1}{2}\left[1 - \sum_j P^2(\omega_j)\right] \tag{3}$$

  which is the expected error rate at node N if the category is selected randomly from the class distribution present at the node.

- For the two-category case, a useful definition of impurity is that **variance impurity**:

$$i(N) = P(\omega_1)P(\omega_2) \tag{2}$$

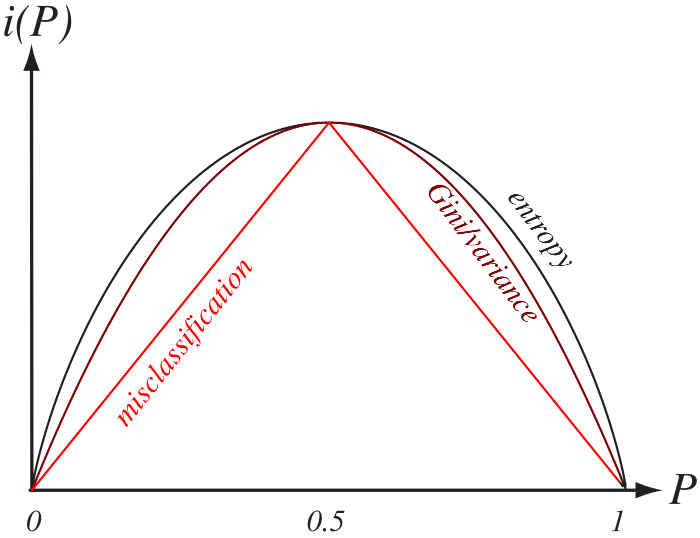
- Its generalization to the multi-class is the **Gini impurity**:

$$i(N) = \sum_{i \neq j} P(\omega_i)P(\omega_j) = \frac{1}{2}\left[1 - \sum_j P^2(\omega_j)\right] \tag{3}$$

  which is the expected error rate at node N if the category is selected randomly from the class distribution present at the node.

- The **misclassification impurity** measures the minimum probability that a training pattern would be misclassified at N:

$$i(N) = 1 - \max_j P(\omega_j) \tag{4}$$



For the two-category case, the impurity functions peak at equal class frequencies.

Query Selection

- Key Question: **Given a partial tree down to node N, what feature s should we choose for the property test T?**

Query Selection

- Key Question: **Given a partial tree down to node N, what feature s should we choose for the property test T?**
- The obvious heuristic is to choose the feature that yields as big a decrease in the impurity as possible.

Query Selection

- Key Question: **Given a partial tree down to node N, what feature s should we choose for the property test T?**
- The obvious heuristic is to choose the feature that yields as big a decrease in the impurity as possible.
- The impurity gradient is

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R) \ , \tag{5}$$

where N_L and N_R are the left and right descendants, respectively, P_L is the fraction of data that will go to the left sub-tree when property T is used.

Query Selection

- Key Question: **Given a partial tree down to node N, what feature s should we choose for the property test T?**
- The obvious heuristic is to choose the feature that yields as big a decrease in the impurity as possible.
- The impurity gradient is

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R) \ , \tag{5}$$

  where N_L and N_R are the left and right descendants, respectively, P_L is the fraction of data that will go to the left sub-tree when property T is used.
- The strategy is then to choose the feature that maximizes $\Delta i(N)$.

Query Selection

- Key Question: **Given a partial tree down to node N, what feature s should we choose for the property test T?**
- The obvious heuristic is to choose the feature that yields as big a decrease in the impurity as possible.
- The impurity gradient is

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R) \ , \tag{5}$$

  where N_L and N_R are the left and right descendants, respectively, P_L is the fraction of data that will go to the left sub-tree when property T is used.
- The strategy is then to choose the feature that maximizes $\Delta i(N)$.
- If the **entropy impurity** is used, this corresponds to choosing the feature that yields the highest **information gain**.

What can we say about this strategy?

- For the binary-case, it yields one-dimensional optimization problem (which may have non-unique optima).

What can we say about this strategy?

- For the binary-case, it yields one-dimensional optimization problem (which may have non-unique optima).
- In the higher branching factor case, it would yield a higher-dimensional optimization problem.
  - In multi-class binary tree creation, we would want to use the **twoing criterion**. The goal is to find the split that best separates groups of the c categories. A candidate "supercategory" C_1 consists of all patterns in some subset of the categories and C_2 has the remainder. When searching for the feature s, we also need to search over possible category groupings.

What can we say about this strategy?

- For the binary-case, it yields one-dimensional optimization problem (which may have non-unique optima).
- In the higher branching factor case, it would yield a higher-dimensional optimization problem.
    - In multi-class binary tree creation, we would want to use the **twoing criterion**. The goal is to find the split that best separates groups of the c categories. A candidate "supercategory" C_1 consists of all patterns in some subset of the categories and C_2 has the remainder. When searching for the feature s, we also need to search over possible category groupings.
- This is a local, greedy optimization strategy.

What can we say about this strategy?

- For the binary-case, it yields one-dimensional optimization problem (which may have non-unique optima).
- In the higher branching factor case, it would yield a higher-dimensional optimization problem.
  - In multi-class binary tree creation, we would want to use the **twoing criterion**. The goal is to find the split that best separates groups of the c categories. A candidate "supercategory" C_1 consists of all patterns in some subset of the categories and C_2 has the remainder. When searching for the feature s, we also need to search over possible category groupings.
- This is a local, greedy optimization strategy.
- Hence, there is no guarantee that we have either the global optimum (in classification accuracy) or the smallest tree.

What can we say about this strategy?

- For the binary-case, it yields one-dimensional optimization problem (which may have non-unique optima).
- In the higher branching factor case, it would yield a higher-dimensional optimization problem.
  - In multi-class binary tree creation, we would want to use the **twoing criterion**. The goal is to find the split that best separates groups of the c categories. A candidate "supercategory" C_1 consists of all patterns in some subset of the categories and C_2 has the remainder. When searching for the feature s, we also need to search over possible category groupings.
- This is a local, greedy optimization strategy.
- Hence, there is no guarantee that we have either the global optimum (in classification accuracy) or the smallest tree.
- In practice, it has been observed that the particular choice of impurity function rarely affects the final classifier and its accuracy.

A Note About Multiway Splits

- In the case of selecting a multiway split with branching factor B, the following is the direct generalization of the impurity gradient function:

$$\Delta i(s) = i(N) - \sum_{k=1}^{B} P_k i(N_k) \tag{6}$$

A Note About Multiway Splits

- In the case of selecting a multiway split with branching factor B, the following is the direct generalization of the impurity gradient function:

$$\Delta i(s) = i(N) - \sum_{k=1}^{B} P_k i(N_k) \tag{6}$$

- This direct generalization is biased toward higher branching factors.
  - To see this, consider the uniform splitting case.

A Note About Multiway Splits

- In the case of selecting a multiway split with branching factor B, the following is the direct generalization of the impurity gradient function:

$$\Delta i(s) = i(N) - \sum_{k=1}^{B} P_k i(N_k) \tag{6}$$

- This direct generalization is biased toward higher branching factors.
    - To see this, consider the uniform splitting case.
- So, we need to normalize each:

$$\Delta i_B(s) = \frac{\Delta i(s)}{-\sum_{k=1}^{B} P_k \log P_k} \quad . \tag{7}$$

And then we can again choose the feature that maximizes this normalized criterion.

When to Stop Splitting?

- If we continue to grow the tree until each leaf node has its lowest impurity (just one sample datum), then we will likely have over-trained the data. This tree will most definitely not generalize well.

When to Stop Splitting?

- If we continue to grow the tree until each leaf node has its lowest impurity (just one sample datum), then we will likely have over-trained the data. This tree will most definitely not generalize well.
- Conversely, if we stop growing the tree too early, the error on the training data will not be sufficiently low and performance will again suffer.

When to Stop Splitting?

- If we continue to grow the tree until each leaf node has its lowest impurity (just one sample datum), then we will likely have over-trained the data. This tree will most definitely not generalize well.
- Conversely, if we stop growing the tree too early, the error on the training data will not be sufficiently low and performance will again suffer.
- So, how to stop splitting?

When to Stop Splitting?

- If we continue to grow the tree until each leaf node has its lowest impurity (just one sample datum), then we will likely have over-trained the data. This tree will most definitely not generalize well.
- Conversely, if we stop growing the tree too early, the error on the training data will not be sufficiently low and performance will again suffer.
- So, how to stop splitting?

1. Cross-validation...
2. Threshold on the impurity gradient.
3. Incorporate a tree-complexity term and minimize.
4. Statistical significance of the impurity gradient.

Stopping by Thresholding the Impurity Gradient

- Splitting is stopped if the best candidate split at a node reduces the impurity by less than the preset amount, β:

$$\max_s \Delta i(s) \leq \beta \quad . \tag{8}$$

Stopping by Thresholding the Impurity Gradient

- Splitting is stopped if the best candidate split at a node reduces the impurity by less than the preset amount, β:

$$\max_s \Delta i(s) \leq \beta \ . \tag{8}$$

- Benefit 1: Unlike cross-validation, the tree is trained on the complete training data set.

Stopping by Thresholding the Impurity Gradient

- Splitting is stopped if the best candidate split at a node reduces the impurity by less than the preset amount, β:

$$\max_s \Delta i(s) \leq \beta \quad . \tag{8}$$

- Benefit 1: Unlike cross-validation, the tree is trained on the complete training data set.
- Benefit 2: Leaf nodes can lie in different levels of the tree, which is desirable whenver the complexity of the data varies throughout the range of values.

Stopping by Thresholding the Impurity Gradient

- Splitting is stopped if the best candidate split at a node reduces the impurity by less than the preset amount, β:

$$\max_s \Delta i(s) \leq \beta \ . \tag{8}$$

- Benefit 1: Unlike cross-validation, the tree is trained on the complete training data set.
- Benefit 2: Leaf nodes can lie in different levels of the tree, which is desirable whenver the complexity of the data varies throughout the range of values.
- Drawback: But, how do we set the value of the threshold β?

Stopping with a Complexity Term

- Define a new global criterion function

$$\alpha \cdot \text{size} + \sum_{\text{leaf nodes}} i(N) \ . \qquad (9)$$

which trades complexity for accuracy. Here, size could represent the number of nodes or links and α is some positive constant.

Stopping with a Complexity Term

- Define a new global criterion function

$$\alpha \cdot \text{size} + \sum_{\text{leaf nodes}} i(N) \quad . \tag{9}$$

  which trades complexity for accuracy. Here, size could represent the number of nodes or links and α is some positive constant.
- The strategy is then to split until a minimum of this global criterion function has been reached.

Stopping with a Complexity Term

- Define a new global criterion function

$$\alpha \cdot \text{size} + \sum_{\text{leaf nodes}} i(N) \ . \tag{9}$$

  which trades complexity for accuracy. Here, size could represent the number of nodes or links and α is some positive constant.
- The strategy is then to split until a minimum of this global criterion function has been reached.
- Given the entropy impurity, this global measure is related to the minimum description length principle.
  - The sum of the impurities at the leaf nodes is a measure of uncertainty in the training data given the model represented by the tree.

Stopping with a Complexity Term

- Define a new global criterion function

$$\alpha \cdot \text{size} + \sum_{\text{leaf nodes}} i(N) \ . \tag{9}$$

  which trades complexity for accuracy. Here, size could represent the number of nodes or links and α is some positive constant.
- The strategy is then to split until a minimum of this global criterion function has been reached.
- Given the entropy impurity, this global measure is related to the minimum description length principle.
  - The sum of the impurities at the leaf nodes is a measure of uncertainty in the training data given the model represented by the tree.
- But, again, how do we set the constant α?

Stopping by Testing the Statistical Significance

- During construction, estimate the distribution of the impurity gradients Δi for the current collection of nodes.

Stopping by Testing the Statistical Significance

- During construction, estimate the distribution of the impurity gradients Δi for the current collection of nodes.
- For any candidate split, estimate if it is statistical different from zero. One possibility is the chi-squared test.

Stopping by Testing the Statistical Significance

- During construction, estimate the distribution of the impurity gradients Δi for the current collection of nodes.
- For any candidate split, estimate if it is statistical different from zero. One possibility is the chi-squared test.
- More generally, we can consider a hypothesis testing approach to stopping: we seek to determine whether a candidate split differs significantly from a random split.

Stopping by Testing the Statistical Significance

- During construction, estimate the distribution of the impurity gradients Δi for the current collection of nodes.
- For any candidate split, estimate if it is statistical different from zero. One possibility is the chi-squared test.
- More generally, we can consider a hypothesis testing approach to stopping: we seek to determine whether a candidate split differs significantly from a random split.
- Suppose we have n samples at node N. A particular split s sends Pn patterns to the left branch and $(1-P)n$ patterns to the right branch. A random split would place P_{n_1} of the ω_1 samples to the left, P_{n_2} of the ω_2 samples to the left and corresponding amounts to the right.

- The chi-squared statistic calculates the deviation of a particular split s from this random one:

$$\chi^2 = \sum_{i=1}^{2} \frac{(n_{iL} - n_{ie})^2}{n_{ie}} \tag{10}$$

where n_{iL} is the number of ω_1 patterns sent to the left under s, and $n_{ie} = Pn_i$ is the number expected by the random rule.

- The chi-squared statistic calculates the deviation of a particular split s from this random one:

$$\chi^2 = \sum_{i=1}^{2} \frac{(n_{iL} - n_{ie})^2}{n_{ie}} \tag{10}$$

where n_{iL} is the number of ω_1 patterns sent to the left under s, and $n_{ie} = Pn_i$ is the number expected by the random rule.

- The larger the chi-squared statistic, the more the candidate split deviates from a random one.

- The chi-squared statistic calculates the deviation of a particular split s from this random one:

$$\chi^2 = \sum_{i=1}^{2} \frac{(n_{iL} - n_{ie})^2}{n_{ie}} \tag{10}$$

where n_{iL} is the number of ω_1 patterns sent to the left under s, and $n_{ie} = Pn_i$ is the number expected by the random rule.

- The larger the chi-squared statistic, the more the candidate split deviates from a random one.
- When it is greater than a critical value (based on desired significance bounds), we reject the null hypothesis (the random split) and proceed with s.

Pruning

- Tree construction based on “when to stop splitting” biases the learning algorithm toward trees in which the greatest impurity reduction occurs near the root. It makes no attempt to *look ahead* at what splits may occur in the leaf and beyond.

Pruning

- Tree construction based on "when to stop splitting" biases the learning algorithm toward trees in which the greatest impurity reduction occurs near the root. It makes no attempt to *look ahead* at what splits may occur in the leaf and beyond.
- **Pruning** is the principal alternative strategy for tree construction.

Pruning

- Tree construction based on "when to stop splitting" biases the learning algorithm toward trees in which the greatest impurity reduction occurs near the root. It makes no attempt to *look ahead* at what splits may occur in the leaf and beyond.
- **Pruning** is the principal alternative strategy for tree construction.
- In pruning, we exhaustively build the tree. Then, all pairs of neighboring leafs nodes are considered for elimination.

Pruning

- Tree construction based on "when to stop splitting" biases the learning algorithm toward trees in which the greatest impurity reduction occurs near the root. It makes no attempt to *look ahead* at what splits may occur in the leaf and beyond.
- **Pruning** is the principal alternative strategy for tree construction.
- In pruning, we exhaustively build the tree. Then, all pairs of neighboring leafs nodes are considered for elimination.
- Any pair that yields a satisfactory increase in impurity (a small one) is eliminated and the common ancestor node is declared a leaf.

Pruning

- Tree construction based on "when to stop splitting" biases the learning algorithm toward trees in which the greatest impurity reduction occurs near the root. It makes no attempt to *look ahead* at what splits may occur in the leaf and beyond.
- **Pruning** is the principal alternative strategy for tree construction.
- In pruning, we exhaustively build the tree. Then, all pairs of neighboring leafs nodes are considered for elimination.
- Any pair that yields a satisfactory increase in impurity (a small one) is eliminated and the common ancestor node is declared a leaf.
- Unbalanced trees often result from this style of pruning/merging.

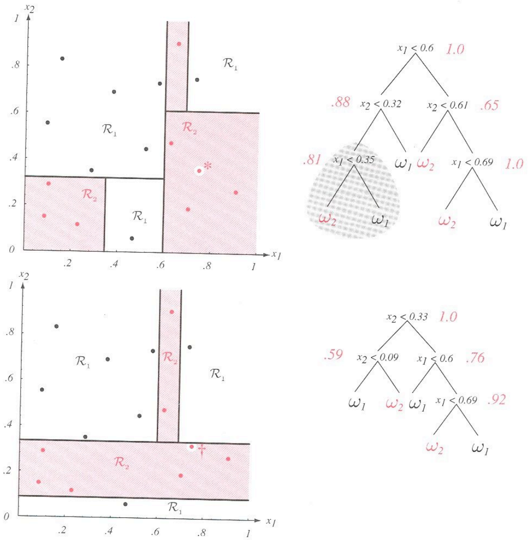
Pruning

- Tree construction based on "when to stop splitting" biases the learning algorithm toward trees in which the greatest impurity reduction occurs near the root. It makes no attempt to *look ahead* at what splits may occur in the leaf and beyond.
- **Pruning** is the principal alternative strategy for tree construction.
- In pruning, we exhaustively build the tree. Then, all pairs of neighboring leafs nodes are considered for elimination.
- Any pair that yields a satisfactory increase in impurity (a small one) is eliminated and the common ancestor node is declared a leaf.
- Unbalanced trees often result from this style of pruning/merging.
- Pruning avoids the "local"-ness of the earlier methods and uses all of the training data, but it does so at added computational cost during the tree construction.

Assignment of Leaf Node Labels

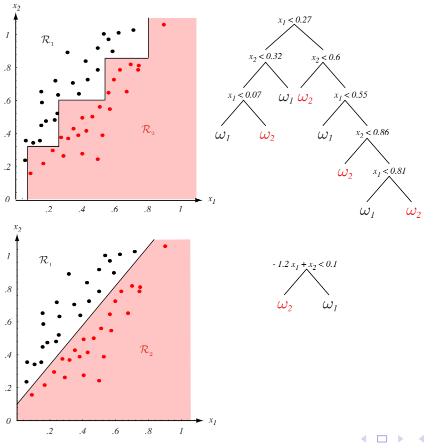
- This part is easy...a particular leaf node should make the label assignment based on the distribution of samples in it during training. Take the label of the maximally represented class.
- We will see clear justification for this in the next chapter on Decision Theory.

Instability of the Tree Construction

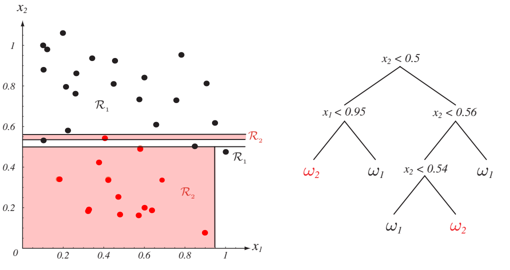


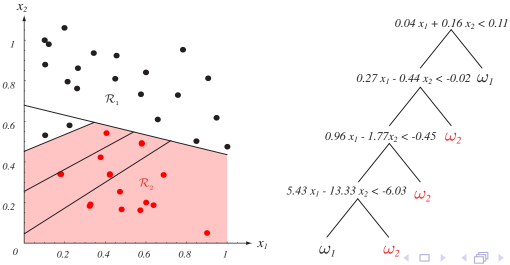
Importance of Feature Choice

- The selection of features will ultimately play a major role in accuracy, generalization, and complexity.
- This is an instance of the Ugly Duckling principle.



- Furthermore, the use of multiple variables in selecting a decision rule may greatly improve the accuracy and generalization.





ID3 Method

- ID3 is another tree growing method.

ID3 Method

- ID3 is another tree growing method.
- It assumes nominal inputs.

ID3 Method

- ID3 is another tree growing method.
- It assumes nominal inputs.
- Every split has a branching factor B_j, where B_j is the number of discrete attribute bins of the variable j chosen for splitting.

ID3 Method

- ID3 is another tree growing method.
- It assumes nominal inputs.
- Every split has a branching factor B_j, where B_j is the number of discrete attribute bins of the variable j chosen for splitting.
- These are, hence, seldom binary.

ID3 Method

- ID3 is another tree growing method.
- It assumes nominal inputs.
- Every split has a branching factor B_j, where B_j is the number of discrete attribute bins of the variable j chosen for splitting.
- These are, hence, seldom binary.
- The number of levels in the trees are equal to the number of input variables.

ID3 Method

- ID3 is another tree growing method.
- It assumes nominal inputs.
- Every split has a branching factor B_j, where B_j is the number of discrete attribute bins of the variable j chosen for splitting.
- These are, hence, seldom binary.
- The number of levels in the trees are equal to the number of input variables.
- The algorithm continues until all nodes are pure or there are no more variables on which to split.

ID3 Method

- ID3 is another tree growing method.
- It assumes nominal inputs.
- Every split has a branching factor B_j, where B_j is the number of discrete attribute bins of the variable j chosen for splitting.
- These are, hence, seldom binary.
- The number of levels in the trees are equal to the number of input variables.
- The algorithm continues until all nodes are pure or there are no more variables on which to split.
- One can follow this by pruning.

C4.5 Method (in brief)

- This is a successor to the ID3 method.

C4.5 Method (in brief)

- This is a successor to the ID3 method.
- It handles real valued variables like CART and uses the ID3 multiway splits for nominal data.

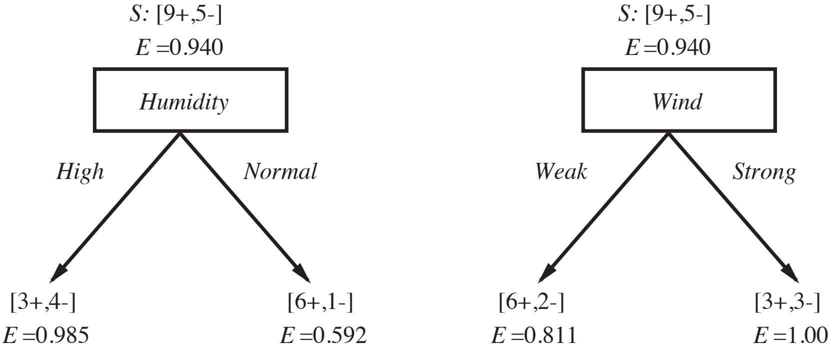
C4.5 Method (in brief)

- This is a successor to the ID3 method.
- It handles real valued variables like CART and uses the ID3 multiway splits for nominal data.
- Pruning is performed based on statistical significance tests.

Example from T. Mitchell Book: PlayTennis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Which attribute is the best classifier?

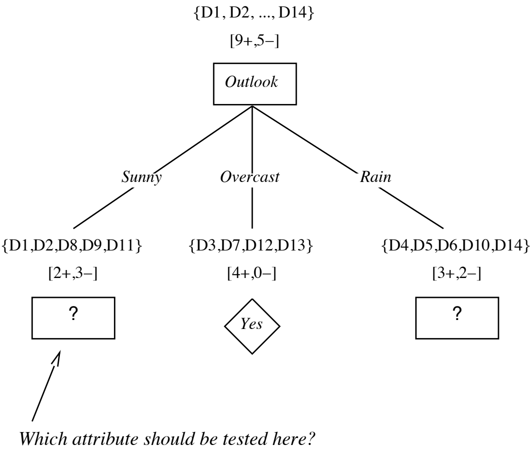


Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048



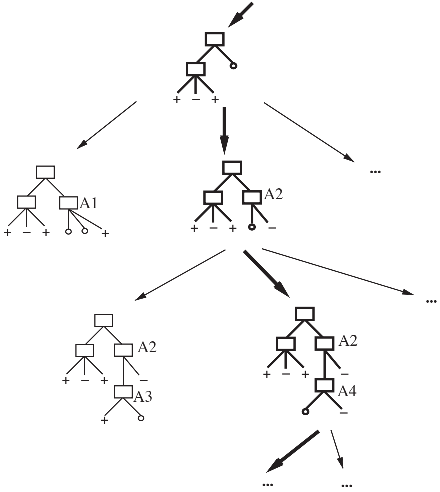
$S_{sunny} = \{D1, D2, D8, D9, D11\}$

$Gain(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$

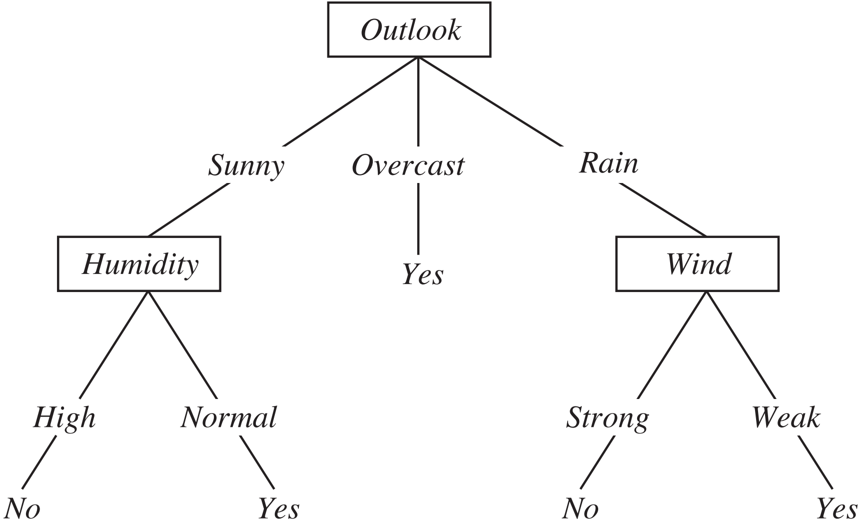
$Gain(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$

$Gain(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$

Hypothesis Space Search by ID3



Learned Tree



Overfitting Instance

- Consider adding a new, noisy training example #15:

$$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$$

- What effect would it have on the earlier tree?

Overfitting Instance

- Consider adding a new, noisy training example #15:

$$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$$

- What effect would it have on the earlier tree?