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

1ofK

- 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...

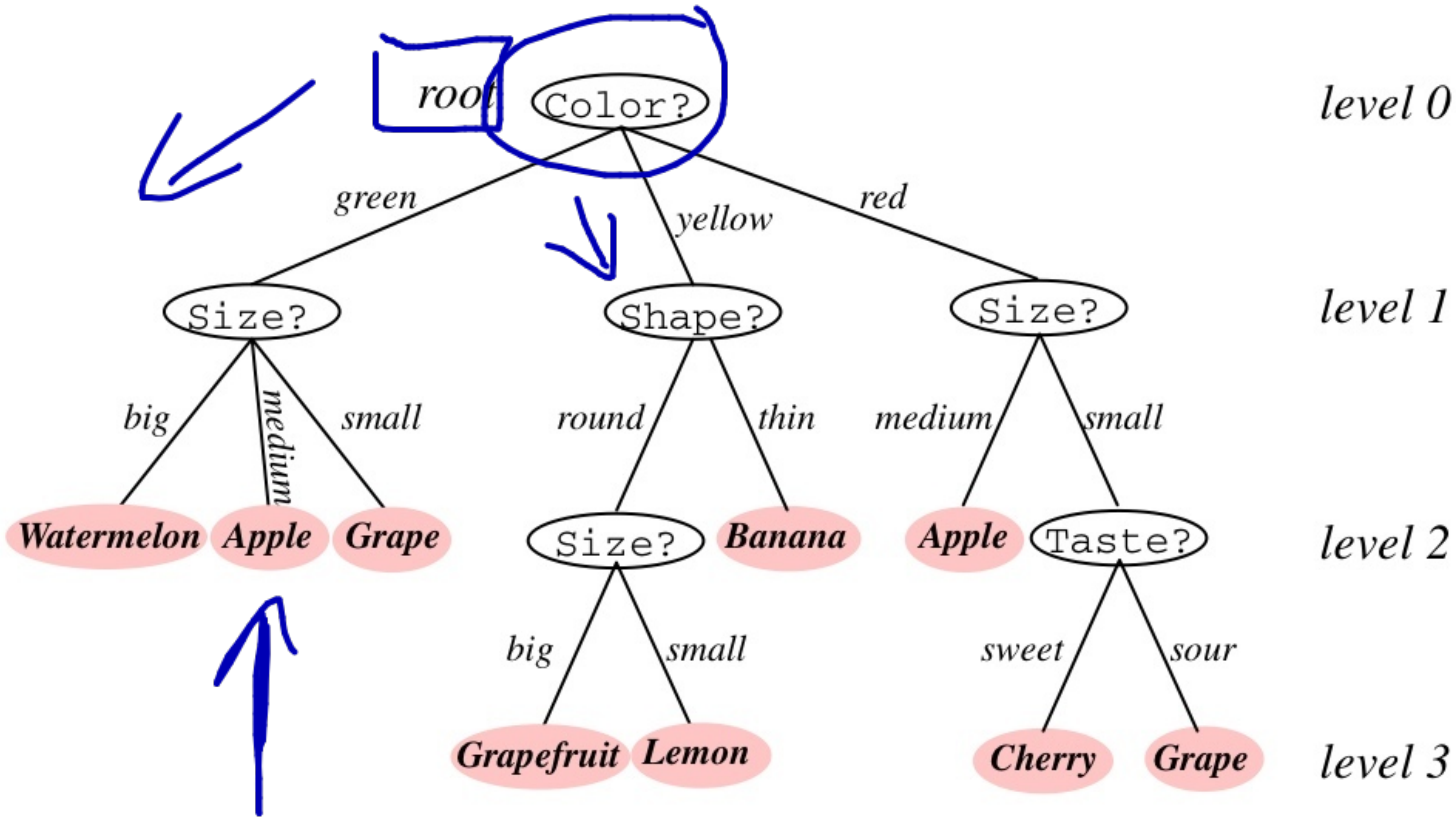
20 Questions

- 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 hy~~pothesis may be required.
- Possibly noisy training data.
- Examples
  - Equipment or medical diagnosis.
  - Credit risk analysis.
  - Modeling calendar scheduling preferences.

CART for Decision Tree Learning

$\mathcal{D} = \{x, y\}$ Breiman

- 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.**

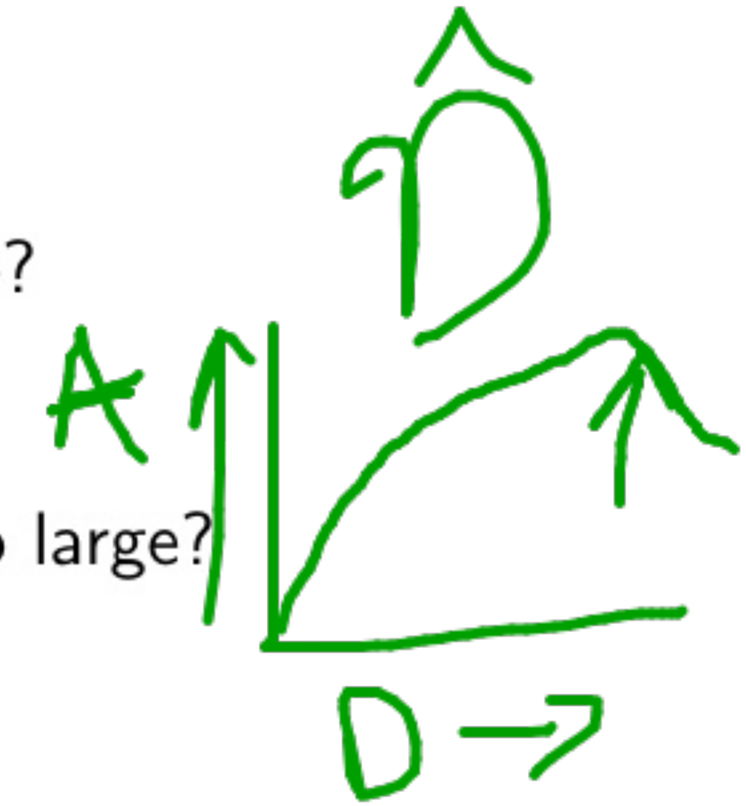
- The basic CART strategy to recursively defining the tree is the following: **Given the data represented at a node, either declare that node to be a leaf or find another property to use to split the data into subsets.**
- There are 6 general kinds of questions that arise:

- 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?

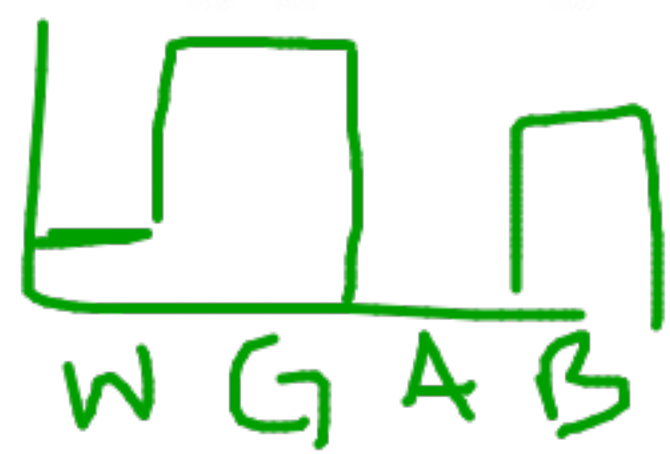
- The basic CART strategy to recursively defining the tree is the following: **Given the data represented at a node, either declare that node to be a leaf or find another property to use to split the data into subsets.**
- There are 6 general kinds of questions that arise:
    1. How many branches will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

ABCDE

Twoing

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

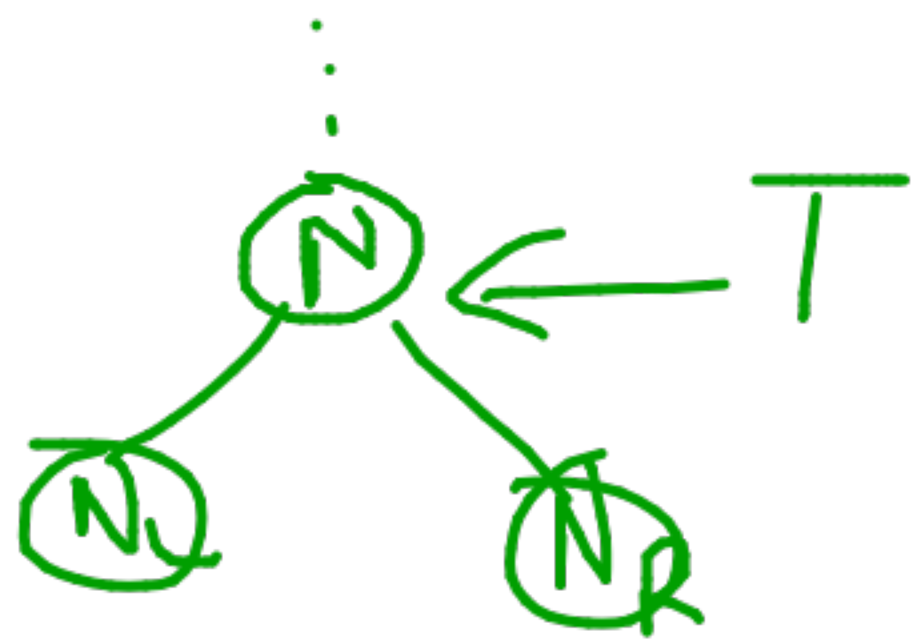
- The number of splits at a node, or its **branching factor** B, is generally set *by the designer* (as a function of the way the test is selected) and can vary throughout the tree.
- Note that any split with a factor greater than 2 can easily be converted into a sequence of binary splits.
- So, DHS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

$\log(0) = 0$

- 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:

30 C
60 W
.10 A

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

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**:

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

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**:

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

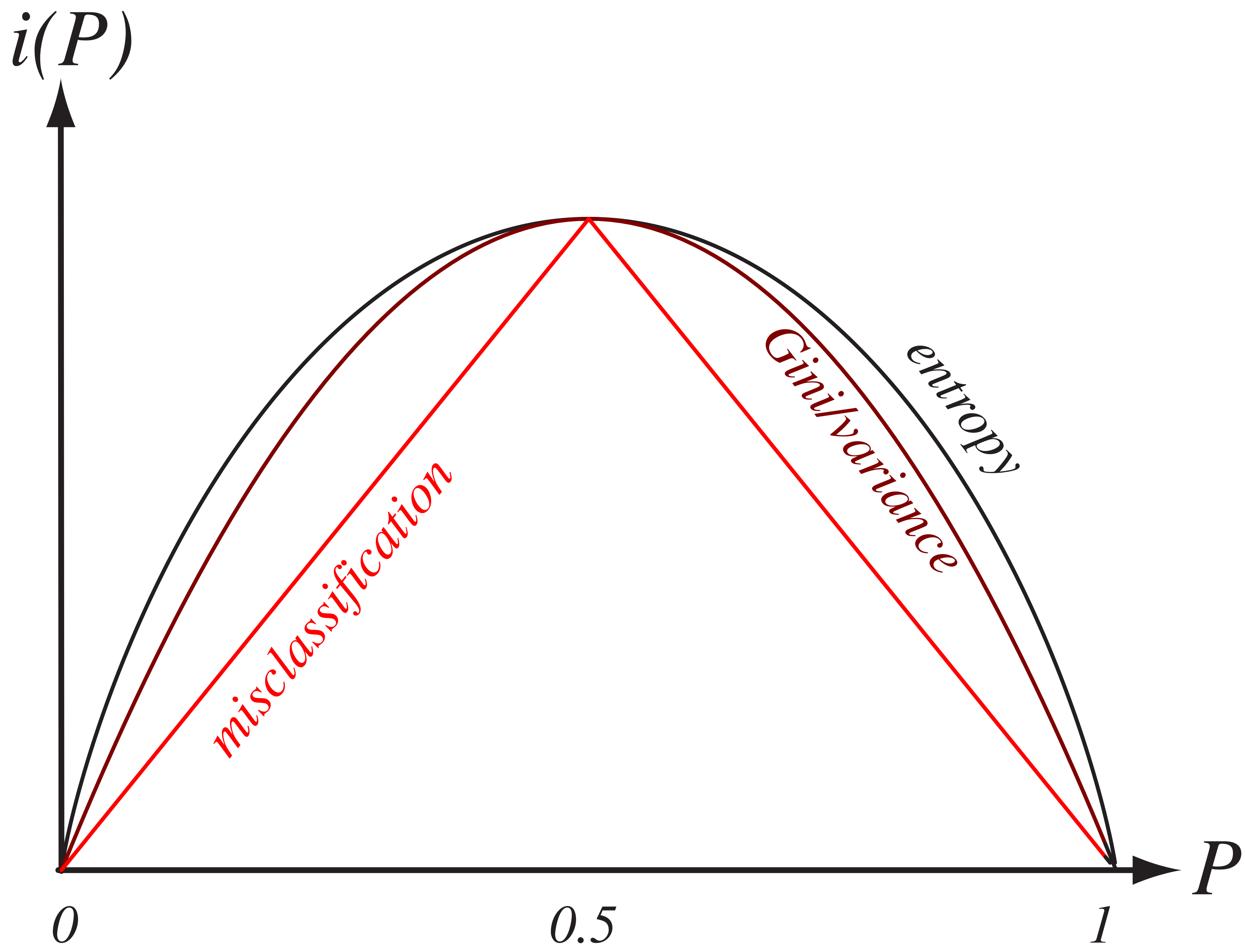
- Its generalization to the multi-class is the **Gini impurity**:

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

  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.

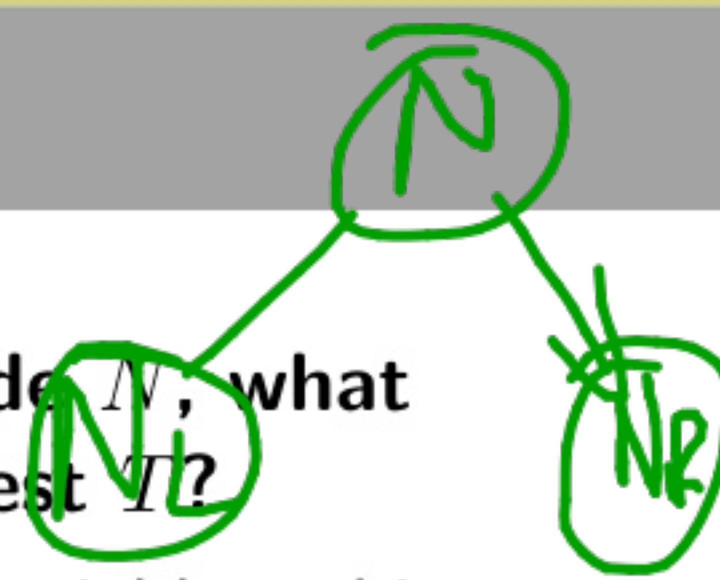
Query Selection

- 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

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

  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.

COVER & Thomas

- 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.