Decision Trees for Classification and Regression

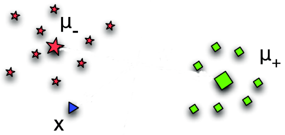
Piyush Rai

Introduction to Machine Learning (CS771A)

August 7, 2018

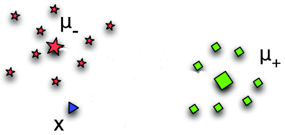
Recap

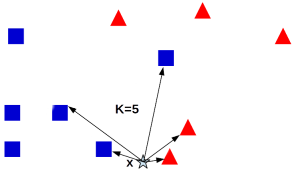
Previous class: Looked at Prototype based classification, and nearest neighbor methods



Recap

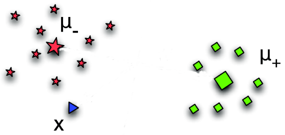
Previous class: Looked at Prototype based classification, and nearest neighbor methods

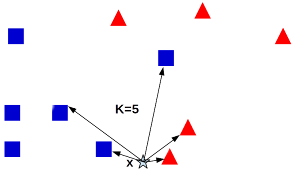


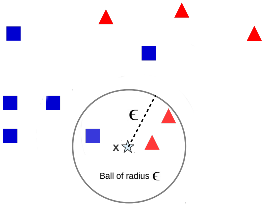


Recap

Previous class: Looked at Prototype based classification, and nearest neighbor methods

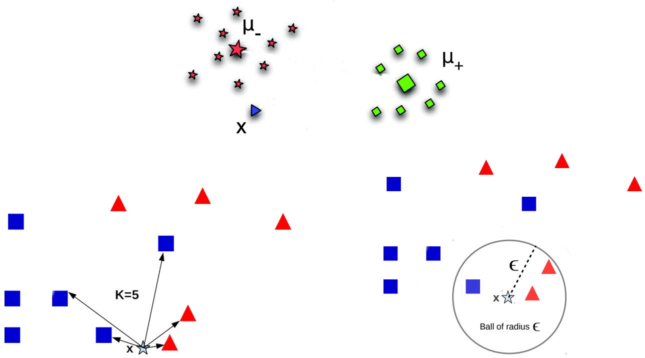






Recap

Previous class: Looked at Prototype based classification, and nearest neighbor methods

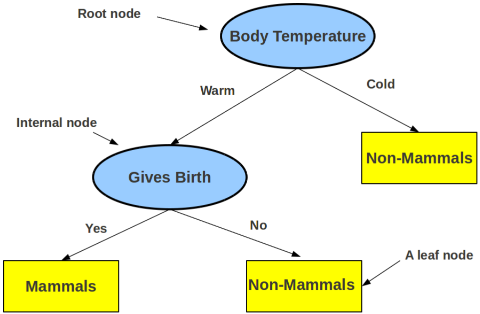


"Local" methods based on distance of the test input with the training inputs (or class prototypes)

Decision Trees
(Another example of a local method)

Decision Tree

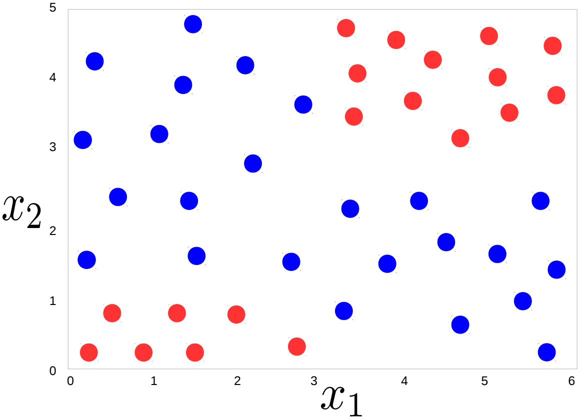
- Defines a tree-structured hierarchy of rules
- Consists of a root node, internal nodes, and leaf nodes



- Root and internal nodes contain the rules
- Leaf nodes define the predictions
- Decision Tree (DT) learning is about learning such a tree from labeled training data

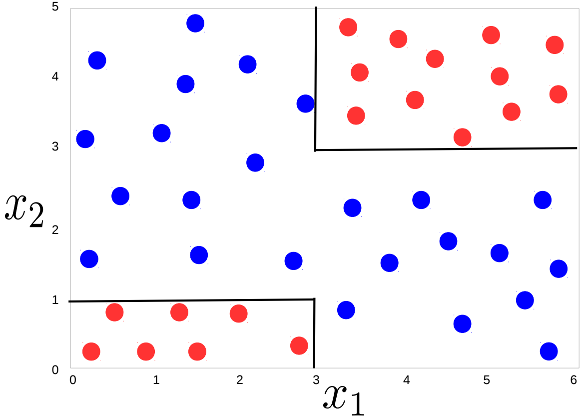
A Classification Problem

Consider binary classification. Assume training data with each input having 2 features (x_1, x_2)



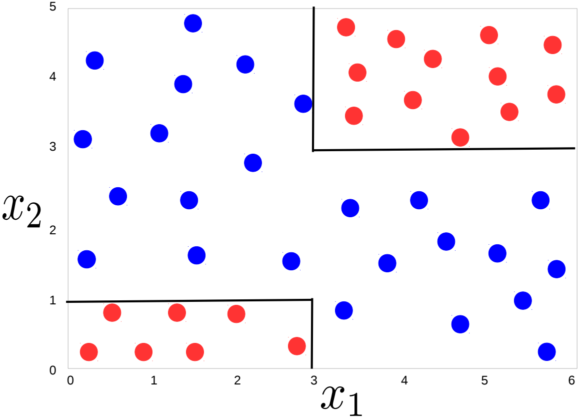
A Classification Problem

The "expected" decision boundary given this training data.



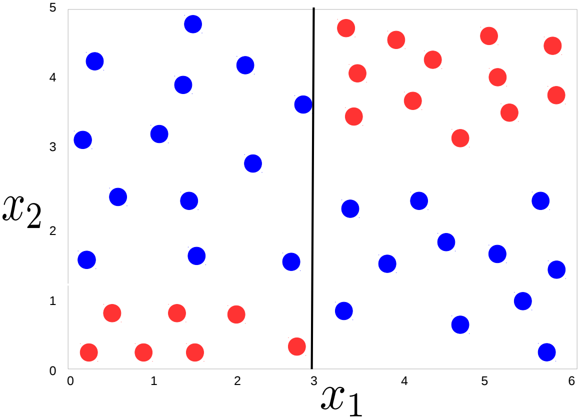
A Classification Problem

The "expected" decision boundary given this training data. Let's learn this!



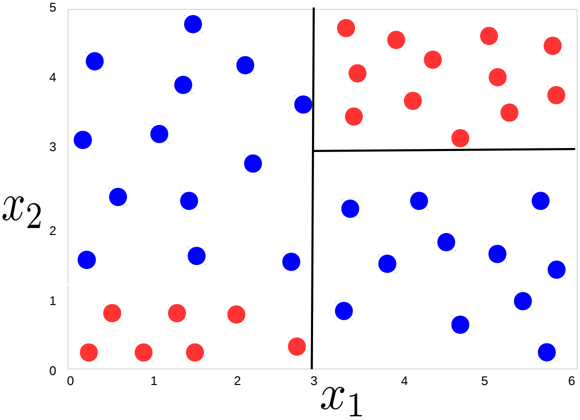
Learning by Asking Questions!

Is x_1 (feature 1) greater than 3 ?



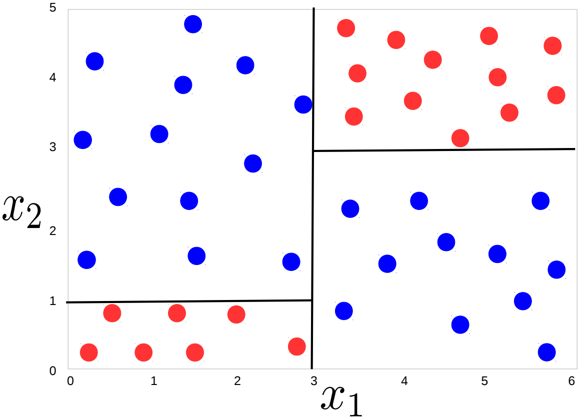
Learning by Asking Questions!

Given $x_1 > 3$, is feature 2 (x_2) greater than 3?



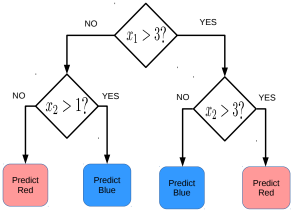
Learning by Asking Questions!

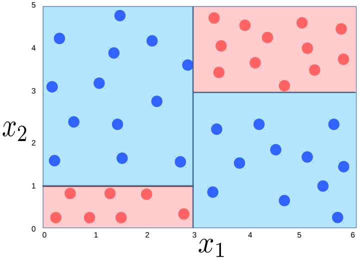
Given $x_1 < 3$, is feature 2 (x_2) greater than 1?



What We Learned

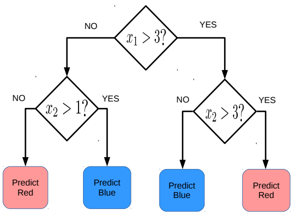
- A Decision Tree (DT) consisting of a set of rules **learned** from training data

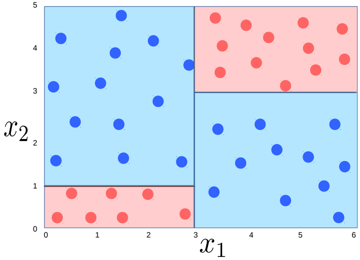




What We Learned

- A Decision Tree (DT) consisting of a set of rules **learned** from training data

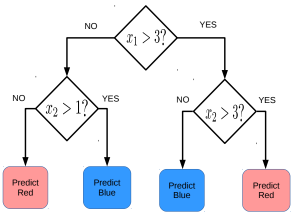


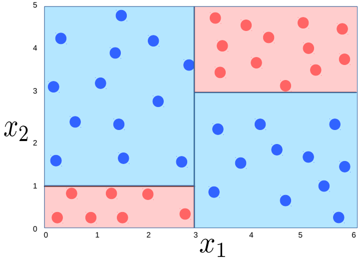


- These rules perform a recursive partitioning of the training data into "homogeneous" regions

What We Learned

- A Decision Tree (DT) consisting of a set of rules **learned** from training data

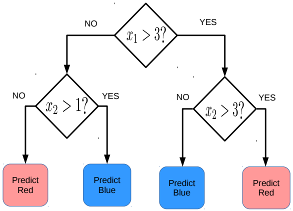


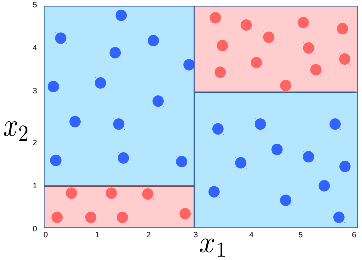


- These rules perform a recursive partitioning of the training data into "homogeneous" regions
  - Homogeneous means that the outputs are same/similar for all inputs in that region

What We Learned

- A Decision Tree (DT) consisting of a set of rules **learned** from training data

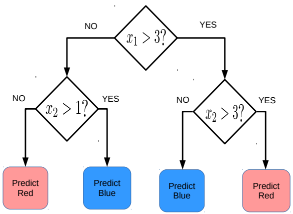


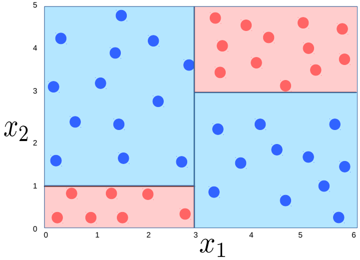


- These rules perform a recursive partitioning of the training data into “homogeneous” regions
  - Homogeneous means that the outputs are same/similar for all inputs in that region
- Given a new test input, we can use the DT to predict its label

What We Learned

- A Decision Tree (DT) consisting of a set of rules **learned** from training data





- These rules perform a recursive partitioning of the training data into "homogeneous" regions
  - Homogeneous means that the outputs are same/similar for all inputs in that region
- Given a new test input, we can use the DT to predict its label
- A key benefit of DT: Prediction at test time is very fast (just testing a few conditions)

Decision Tree for Classification: Another Example

- Deciding whether to play or not to play Tennis on a Saturday
  - Each input (a Saturday) has 4 categorical features: Outlook, Temp., Humidity, Wind
  - A binary classification problem (play vs no-play)

Example credit: Tom Mitchell

Decision Tree for Classification: Another Example

- Deciding whether to play or not to play Tennis on a Saturday
  - Each input (a Saturday) has 4 categorical features: Outlook, Temp., Humidity, Wind
  - A binary classification problem (play vs no-play)
  - Left: Training data, Right: A decision tree constructed using this data

Example credit: Tom Mitchell

Decision Tree for Classification: Another Example

- Deciding whether to play or not to play Tennis on a Saturday
  - Each input (a Saturday) has 4 categorical features: Outlook, Temp., Humidity, Wind
  - A binary classification problem (play vs no-play)
  - Left: Training data, Right: A decision tree constructed using this data

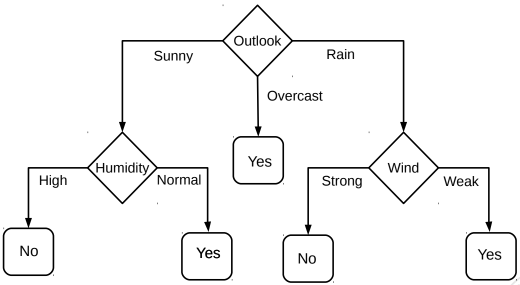
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

Example credit: Tom Mitchell

Decision Tree for Classification: Another Example

- Deciding whether to play or not to play Tennis on a Saturday
  - Each input (a Saturday) has 4 categorical features: Outlook, Temp., Humidity, Wind
  - A binary classification problem (play vs no-play)
  - Left: Training data, Right: A decision tree constructed using this data

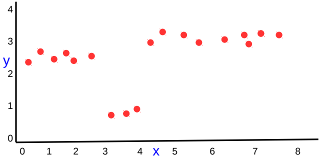
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



Example credit: Tom Mitchell

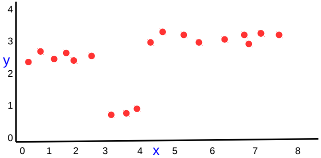
A Decision Tree for Regression

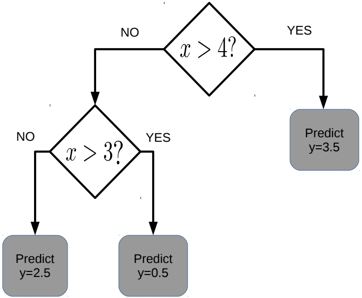
Decision Trees can also be used for regression problems



A Decision Tree for Regression

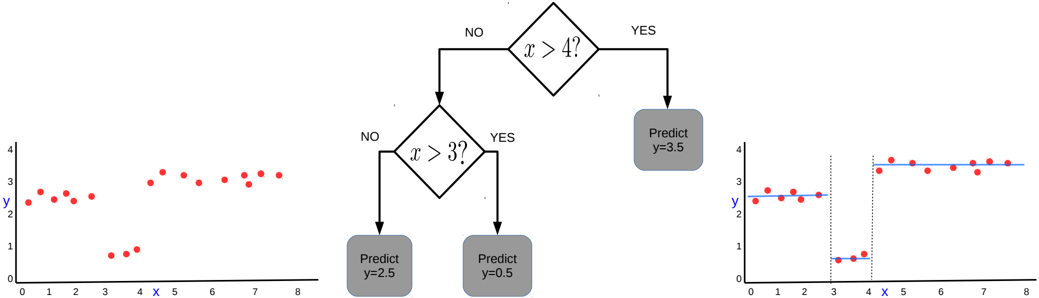
Decision Trees can also be used for regression problems





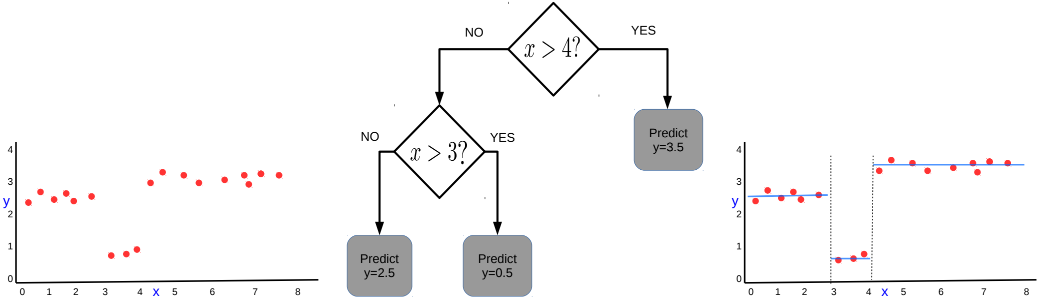
A Decision Tree for Regression

Decision Trees can also be used for regression problems



A Decision Tree for Regression

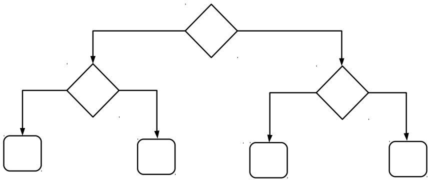
Decision Trees can also be used for regression problems



Here too, the DT partitions the training data into homogeneous regions (inputs with similar outputs)

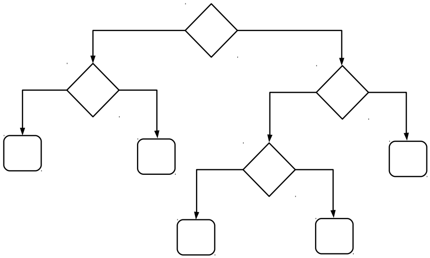
Some Considerations: Shape/Size of DT

- What should be the size/shape of the DT?
  - Number of internal and leaf nodes
  - Branching factor of internal nodes
  - Depth of the tree



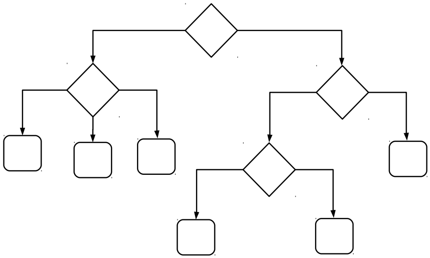
Some Considerations: Shape/Size of DT

- What should be the size/shape of the DT?
  - Number of internal and leaf nodes
  - Branching factor of internal nodes
  - Depth of the tree



Some Considerations: Shape/Size of DT

- What should be the size/shape of the DT?
  - Number of internal and leaf nodes
  - Branching factor of internal nodes
  - Depth of the tree

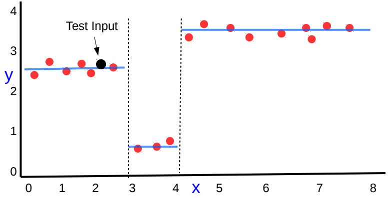


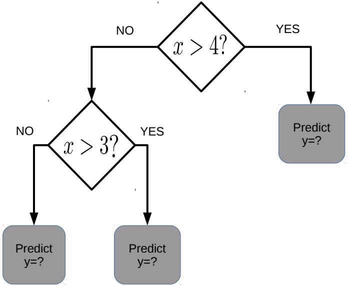
Some Considerations: Leaf Nodes

- What to do at each leaf node (the goal: make predictions)? Some options:
  - Make a constant prediction (majority/average) for every test input reaching that leaf node?
  - Use a nearest neighbors based prediction using all training inputs on that leaf node?

Some Considerations: Leaf Nodes

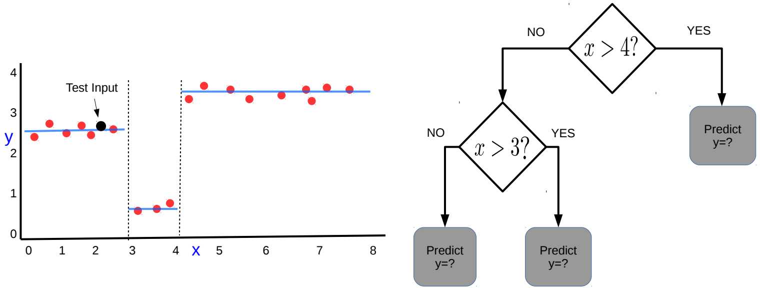
- What to do at each leaf node (the goal: make predictions)? Some options:
  - Make a constant prediction (majority/average) for every test input reaching that leaf node?
  - Use a nearest neighbors based prediction using all training inputs on that leaf node?





Some Considerations: Leaf Nodes

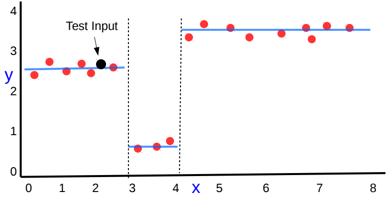
- What to do at each leaf node (the goal: make predictions)? Some options:
  - Make a constant prediction (majority/average) for every test input reaching that leaf node?
  - Use a nearest neighbors based prediction using all training inputs on that leaf node?

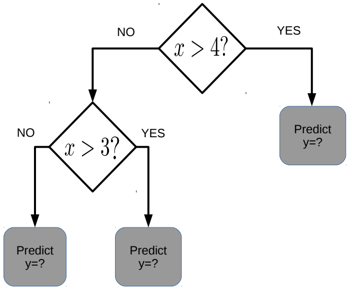


  - (Less common) Predict using an ML model learned using training inputs that belong to that leaf node?

Some Considerations: Leaf Nodes

- What to do at each leaf node (the goal: make predictions)? Some options:
  - Make a constant prediction (majority/average) for every test input reaching that leaf node?
  - Use a nearest neighbors based prediction using all training inputs on that leaf node?





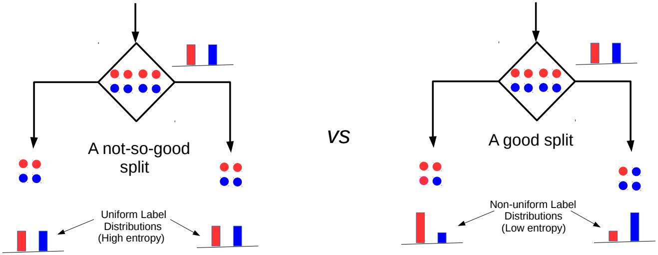
  - (Less common) Predict using an ML model learned using training inputs that belong to that leaf node?
- Constant prediction is the fastest at test time (and gives a piece-wise constant prediction rule)

Some Considerations: Internal Nodes

- How to split at each internal node (the goal: split the training data received at that node)?

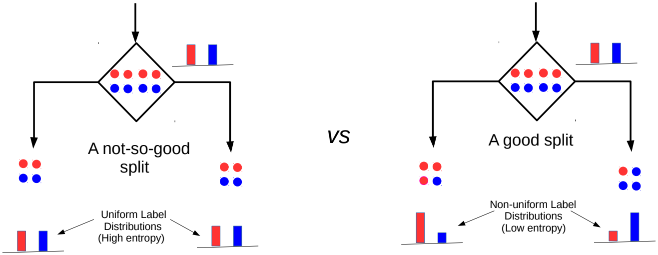
Some Considerations: Internal Nodes

- How to split at each internal node (the goal: split the training data received at that node)?
- No matter how we split, the goal should be to have splits that result in groups as "pure" as possible



Some Considerations: Internal Nodes

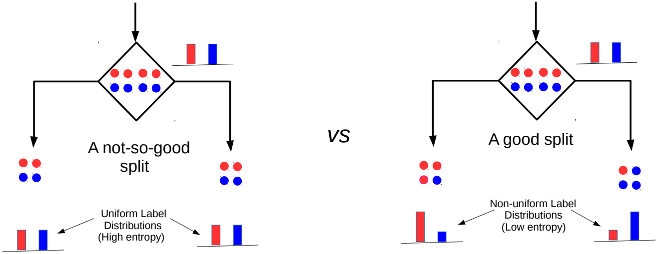
- How to split at each internal node (the goal: split the training data received at that node)?
- No matter how we split, the goal should be to have splits that result in groups as "pure" as possible



- For classification problems, entropy of the label distribution is a measure of purity

Some Considerations: Internal Nodes

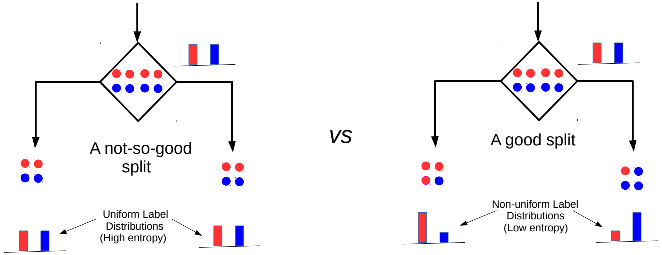
- How to split at each internal node (the goal: split the training data received at that node)?
- No matter how we split, the goal should be to have splits that result in groups as "pure" as possible



- For classification problems, entropy of the label distribution is a measure of purity
  - Low entropy $\Rightarrow$ high purity (less uniform label distribution)

Some Considerations: Internal Nodes

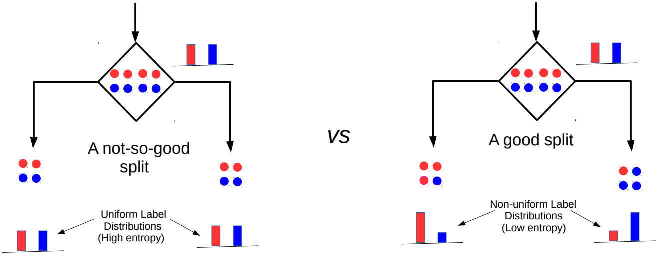
- How to split at each internal node (the goal: split the training data received at that node)?
- No matter how we split, the goal should be to have splits that result in groups as "pure" as possible



- For classification problems, entropy of the label distribution is a measure of purity
  - Low entropy $\Rightarrow$ high purity (less uniform label distribution)
  - Splits that give the largest reduction (before split vs after split) in entropy are preferred (this reduction is also known as "information gain")

Some Considerations: Internal Nodes

- How to split at each internal node (the goal: split the training data received at that node)?
- No matter how we split, the goal should be to have splits that result in groups as "pure" as possible



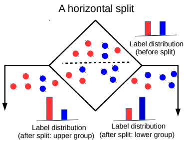
- For classification problems, entropy of the label distribution is a measure of purity
  - Low entropy $\Rightarrow$ high purity (less uniform label distribution)
  - Splits that give the largest reduction (before split vs after split) in entropy are preferred (this reduction is also known as "information gain")
- For regression, entropy doesn't make sense (outputs are real-valued). Typically variance is used.

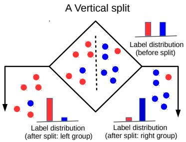
Some Considerations: Internal Nodes (Contd.)

- Note that splitting at internal node itself is like a classification problem
  - Data received at the internal node has to be routed along its outgoing branches

Some Considerations: Internal Nodes (Contd.)

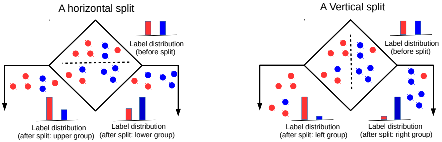
- Note that splitting at internal node itself is like a classification problem
  - Data received at the internal node has to be routed along its outgoing branches
- Some common techniques for splitting an internal node
  - Splitting by testing a single feature (simplest/fastest; used in ID3, C4.5 DT algos). For example:



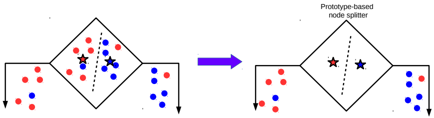


Some Considerations: Internal Nodes (Contd.)

- Note that splitting at internal node itself is like a classification problem
  - Data received at the internal node has to be routed along its outgoing branches
- Some common techniques for splitting an internal node
  - Splitting by testing a single feature (simplest/fastest; used in ID3, C4.5 DT algos). For example:

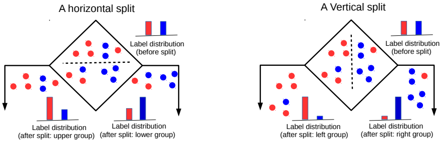


  - Splitting using a classifier learned using data on that node. For example, prototype based classifier

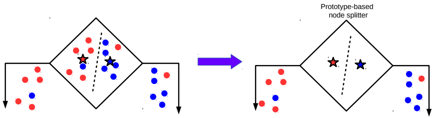


Some Considerations: Internal Nodes (Contd.)

- Note that splitting at internal node itself is like a classification problem
  - Data received at the internal node has to be routed along its outgoing branches
- Some common techniques for splitting an internal node
  - Splitting by testing a single feature (simplest/fastest; used in ID3, C4.5 DT algos). For example:



  - Splitting using a classifier learned using data on that node. For example, prototype based classifier

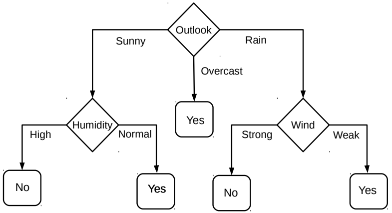


- The same splitting rule will be applied to route a test input that reaches this internal node

Decision Tree Construction

- As an illustration, let's look at one way of constructing a decision tree for some given data
- We will use the entropy/information-gain based splitting criterion for this illustration

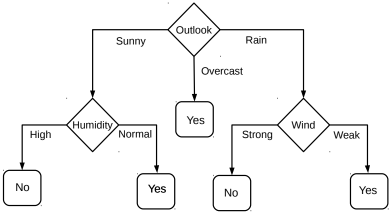
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



Decision Tree Construction

- As an illustration, let's look at one way of constructing a decision tree for some given data
- We will use the entropy/information-gain based splitting criterion for this illustration

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

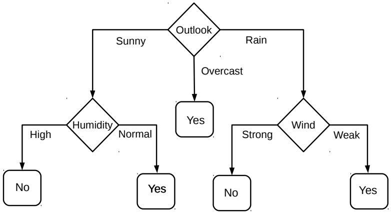


- Question: Why does it make more sense to test the feature "outlook" first?

Decision Tree Construction

- As an illustration, let's look at one way of constructing a decision tree for some given data
- We will use the entropy/information-gain based splitting criterion for this illustration

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

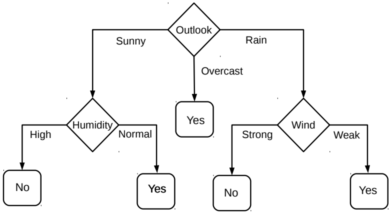


- Question: Why does it make more sense to test the feature "outlook" first?
- Answer: Of all the 4 features, it's most informative (highest information gain as the root node)

Decision Tree Construction

- As an illustration, let's look at one way of constructing a decision tree for some given data
- We will use the entropy/information-gain based splitting criterion for this illustration

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



- Question: Why does it make more sense to test the feature "outlook" first?
- Answer: Of all the 4 features, it's most informative (highest information gain as the root node)
- **Analogy:** Playing the game 20 Questions (the most useful questions first)

Entropy and Information Gain

- Consider a set S of inputs with a total C classes, $p_c =$ fraction of inputs from class/label c

Entropy and Information Gain

- Consider a set S of inputs with a total C classes, p_c = fraction of inputs from class/label c
- Entropy of the set S: $H(S) = -\sum_{c \in C} p_c \log_2 p_c$

Entropy and Information Gain

- Consider a set S of inputs with a total C classes, p_c = fraction of inputs from class/label c
- Entropy of the set S: $H(S) = -\sum_{c \in C} p_c \log_2 p_c$
- The difference in the entropy before and after the split is called information gain (IG)

Entropy and Information Gain

- Consider a set S of inputs with a total C classes, p_c = fraction of inputs from class/label c
- Entropy of the set S: $H(S) = -\sum_{c \in C} p_c \log_2 p_c$
- The difference in the entropy before and after the split is called information gain (IG)
- For one group S being split into two smaller groups S_1 and S_2, we can calculate the IG as follows

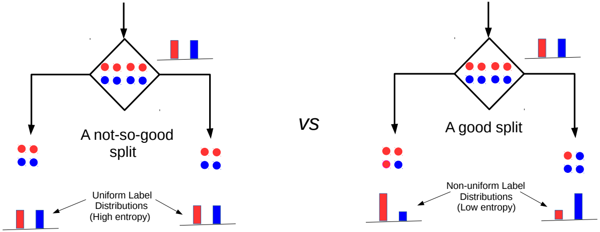
$$IG = H(S) - \frac{|S_1|}{|S|}H(S_1) - \frac{|S_2|}{|S|}H(S_2)$$

Entropy and Information Gain

- Consider a set S of inputs with a total C classes, p_c = fraction of inputs from class/label c
- Entropy of the set S: $H(S) = -\sum_{c \in C} p_c \log_2 p_c$
- The difference in the entropy before and after the split is called information gain (IG)
- For one group S being split into two smaller groups S_1 and S_2, we can calculate the IG as follows

$$IG = H(S) - \frac{|S_1|}{|S|}H(S_1) - \frac{|S_2|}{|S|}H(S_2)$$

- For DT construction, entropy/IG gives us a criterion to select the best split for an internal node



DT Construction using IG Criterion

- Let's look at IG based DT construction for the Tennis example
- Let's begin with the root node of the DT and compute *IG* of each feature
- Consider feature "wind" $\in$ {weak,strong} and its *IG* at root

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

DT Construction using IG Criterion

- Let's look at IG based DT construction for the Tennis example
- Let's begin with the root node of the DT and compute IG of each feature
- Consider feature "wind" $\in$ {weak,strong} and its IG at root

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

- Root node: $S = [9+, 5-]$ (all training data: 9 play, 5 no-play)
- Entropy: $H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$

DT Construction using IG Criterion

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

- Let's look at IG based DT construction for the Tennis example
- Let's begin with the root node of the DT and compute *IG* of each feature
- Consider feature "wind" $\in$ {weak,strong} and its *IG* at root

- Root node: $S = [9+, 5-]$ (all training data: 9 play, 5 no-play)
- Entropy: $H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$
- $S_{weak} = [6+, 2-] \implies H(S_{weak}) = 0.811$, $S_{strong} = [3+, 3-] \implies H(S_{strong}) = 1$

DT Construction using IG Criterion

- Let's look at IG based DT construction for the Tennis example
- Let's begin with the root node of the DT and compute *IG* of each feature
- Consider feature "wind" $\in$ {weak,strong} and its *IG* at root

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

- Root node: $S = [9+, 5-]$ (all training data: 9 play, 5 no-play)
- Entropy: $H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$
- $S_{weak} = [6+, 2-] \implies H(S_{weak}) = 0.811,\ S_{strong} = [3+, 3-] \implies H(S_{strong}) = 1$

$$
\begin{aligned}
IG(S, \text{wind}) &= H(S) - \frac{|S_{weak}|}{|S|} H(S_{weak}) - \frac{|S_{strong}|}{|S|} H(S_{strong}) \\
&= 0.94 - 8/14 * 0.811 - 6/14 * 1 \\
&= 0.048
\end{aligned}
$$

DT Construction using IG Criterion

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

- Let's look at IG based DT construction for the Tennis example
- Let's begin with the root node of the DT and compute IG of each feature
- Consider feature "wind" $\in$ {weak,strong} and its IG at root

- Root node: $S = [9+, 5-]$ (all training data: 9 play, 5 no-play)
- Entropy: $H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$
- $S_{weak} = [6+, 2-] \implies H(S_{weak}) = 0.811$, $S_{strong} = [3+, 3-] \implies H(S_{strong}) = 1$

$$\begin{aligned} IG(S, \text{wind}) &= H(S) - \frac{|S_{weak}|}{|S|}H(S_{weak}) - \frac{|S_{strong}|}{|S|}H(S_{strong}) \\ &= 0.94 - 8/14 * 0.811 - 6/14 * 1 \\ &= 0.048 \end{aligned}$$

Likewise, $IG(S, \text{outlook}) = 0.246$, $IG(S, \text{humidity}) = 0.151$, $IG(S, \text{temperature}) = 0.029 \Rightarrow$ outlook chosen

DT Construction using IG Criterion: Growing the tree

- Having decided which feature to test at the root, let's grow the tree

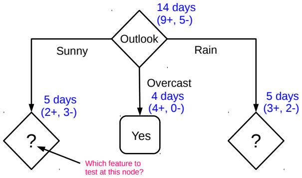
DT Construction using IG Criterion: Growing the tree

- Having decided which feature to test at the root, let's grow the tree
- How to decide which feature to test at the next level (level 2) ?

DT Construction using IG Criterion: Growing the tree

- Having decided which feature to test at the root, let's grow the tree
- How to decide which feature to test at the next level (level 2) ?
- **Rule:** Iterate - for each child node, select the feature with the highest IG

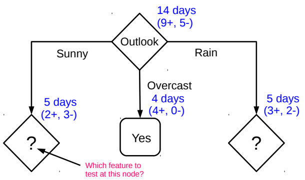
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



DT Construction using IG Criterion: Growing the tree

- Having decided which feature to test at the root, let's grow the tree
- How to decide which feature to test at the next level (level 2) ?
- **Rule:** Iterate - for each child node, select the feature with the highest IG

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

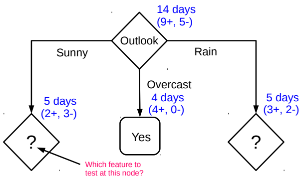


- Proceeding as before, for level 2, left node, we can verify that
  - $IG(S, \text{temperature}) = 0.570$, $IG(S, \text{humidity}) = 0.970$, $IG(S, \text{wind}) = 0.019$ (thus humidity chosen)
- No need to expand the middle node (already "pure" - all yes)

DT Construction using IG Criterion: Growing the tree

- Having decided which feature to test at the root, let's grow the tree
- How to decide which feature to test at the next level (level 2) ?
- **Rule:** Iterate - for each child node, select the feature with the highest IG

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

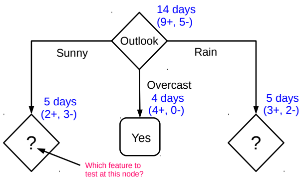


- Proceeding as before, for level 2, left node, we can verify that
  - $IG(S, \text{temperature}) = 0.570$, $IG(S, \text{humidity}) = 0.970$, $IG(S, \text{wind}) = 0.019$ (thus humidity chosen)
- No need to expand the middle node (already "pure" - all yes)
- Can also verify that wind has the largest IG for the right node

DT Construction using IG Criterion: Growing the tree

- Having decided which feature to test at the root, let's grow the tree
- How to decide which feature to test at the next level (level 2) ?
- **Rule:** Iterate - for each child node, select the feature with the highest IG

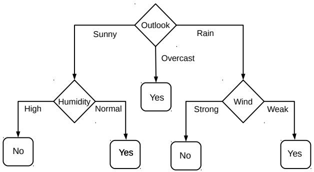
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



- Proceeding as before, for level 2, left node, we can verify that
  - $IG(S, \text{temperature}) = 0.570$, $IG(S, \text{humidity}) = 0.970$, $IG(S, \text{wind}) = 0.019$ (thus humidity chosen)
- No need to expand the middle node (already "pure" - all yes)
- Can also verify that wind has the largest IG for the right node
- Note: If a feature has already been tested along a path earlier, we don't consider it again

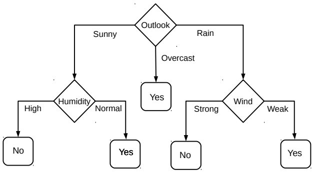
When to Stop Growing?

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



When to Stop Growing?

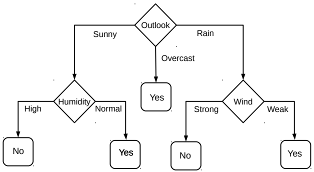
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



- Stop expanding a node further when

When to Stop Growing?

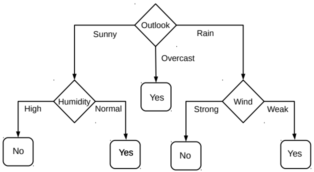
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



- Stop expanding a node further when
  - It consist of examples all having the same label (the node becomes "pure")

When to Stop Growing?

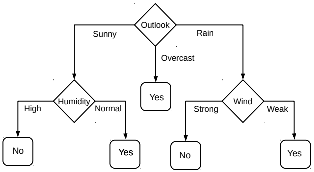
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



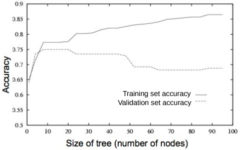
- Stop expanding a node further when
  - It consist of examples all having the same label (the node becomes "pure")
  - We run out of features to test along the path to that node

When to Stop Growing?

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



- Stop expanding a node further when
  - It consist of examples all having the same label (the node becomes "pure")
  - We run out of features to test along the path to that node
  - The DT starts to overfit (can be checked by monitoring the validation set accuracy)



Avoiding Overfitting: Decision Tree Pruning

- Desired: a DT that is not too big in size, yet fits the training data reasonably
- Note: An example of a very simple DT is "decision-stump"
  - A decision-stump only tests the value of a single feature
  - Not very powerful in itself but often used in large ensembles of decision stumps
- Mainly two approaches to prune a complex DT

Avoiding Overfitting: Decision Tree Pruning

- Desired: a DT that is not too big in size, yet fits the training data reasonably
- Note: An example of a very simple DT is "decision-stump"
  - A decision-stump only tests the value of a single feature
  - Not very powerful in itself but often used in large ensembles of decision stumps
- Mainly two approaches to prune a complex DT
  - Prune while building the tree (**stopping early**)

Avoiding Overfitting: Decision Tree Pruning

- Desired: a DT that is not too big in size, yet fits the training data reasonably
- Note: An example of a very simple DT is "decision-stump"
  - A decision-stump only tests the value of a single feature
  - Not very powerful in itself but often used in large ensembles of decision stumps
- Mainly two approaches to prune a complex DT
  - Prune while building the tree (**stopping early**)
  - Prune after building the tree (**post-pruning**)

Avoiding Overfitting: Decision Tree Pruning

- Desired: a DT that is not too big in size, yet fits the training data reasonably
- Note: An example of a very simple DT is "decision-stump"
  - A decision-stump only tests the value of a single feature
  - Not very powerful in itself but often used in large ensembles of decision stumps
- Mainly two approaches to prune a complex DT
  - Prune while building the tree (**stopping early**)
  - Prune after building the tree (**post-pruning**)
- Criteria for judging which nodes could potentially be pruned
  - Use a validation set (separate from the training set)

Avoiding Overfitting: Decision Tree Pruning

- Desired: a DT that is not too big in size, yet fits the training data reasonably
- Note: An example of a very simple DT is "decision-stump"
  - A decision-stump only tests the value of a single feature
  - Not very powerful in itself but often used in large ensembles of decision stumps
- Mainly two approaches to prune a complex DT
  - Prune while building the tree (**stopping early**)
  - Prune after building the tree (**post-pruning**)
- Criteria for judging which nodes could potentially be pruned
  - Use a validation set (separate from the training set)
    - Prune each possible node that doesn't hurt the accuracy on the validation set

Avoiding Overfitting: Decision Tree Pruning

- Desired: a DT that is not too big in size, yet fits the training data reasonably
- Note: An example of a very simple DT is "decision-stump"
  - A decision-stump only tests the value of a single feature
  - Not very powerful in itself but often used in large ensembles of decision stumps
- Mainly two approaches to prune a complex DT
  - Prune while building the tree (**stopping early**)
  - Prune after building the tree (**post-pruning**)
- Criteria for judging which nodes could potentially be pruned
  - Use a validation set (separate from the training set)
    - Prune each possible node that doesn't hurt the accuracy on the validation set
    - Greedily remove the node that improves the validation accuracy the most

Avoiding Overfitting: Decision Tree Pruning

- Desired: a DT that is not too big in size, yet fits the training data reasonably
- Note: An example of a very simple DT is "decision-stump"
  - A decision-stump only tests the value of a single feature
  - Not very powerful in itself but often used in large ensembles of decision stumps
- Mainly two approaches to prune a complex DT
  - Prune while building the tree (**stopping early**)
  - Prune after building the tree (**post-pruning**)
- Criteria for judging which nodes could potentially be pruned
  - Use a validation set (separate from the training set)
    - Prune each possible node that doesn't hurt the accuracy on the validation set
    - Greedily remove the node that improves the validation accuracy the most
    - Stop when the validation set accuracy starts worsening

Avoiding Overfitting: Decision Tree Pruning

- Desired: a DT that is not too big in size, yet fits the training data reasonably
- Note: An example of a very simple DT is "decision-stump"
  - A decision-stump only tests the value of a single feature
  - Not very powerful in itself but often used in large ensembles of decision stumps
- Mainly two approaches to prune a complex DT
  - Prune while building the tree (**stopping early**)
  - Prune after building the tree (**post-pruning**)
- Criteria for judging which nodes could potentially be pruned
  - Use a validation set (separate from the training set)
    - Prune each possible node that doesn't hurt the accuracy on the validation set
    - Greedily remove the node that improves the validation accuracy the most
    - Stop when the validation set accuracy starts worsening
  - Use model selection methods, such as Minimum Description Length (MDL); more on this later

Decision Tree: Some Comments

- Other alternatives to entropy for judging feature informativeness in DT classification?
  - Gini-index $\sum_{c=1}^{C} p_c(1 - p_c)$ is another popular choice

[1] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees

Decision Tree: Some Comments

- Other alternatives to entropy for judging feature informativeness in DT classification?
  - Gini-index $\sum_{c=1}^{C} p_c(1 - p_c)$ is another popular choice
- For DT regression (Regression Trees[1]), can split based on the variance in the outputs, instead of using entropy (which doesn't make sense for real-valued inputs)

[1] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees

Decision Tree: Some Comments

- Other alternatives to entropy for judging feature informativeness in DT classification?
  - Gini-index $\sum_{c=1}^{C} p_c(1 - p_c)$ is another popular choice
- For DT regression (Regression Trees[1]), can split based on the variance in the outputs, instead of using entropy (which doesn't make sense for real-valued inputs)
- Real-valued features (we already saw some examples) can be dealt with using thresholding
  - Need to be careful w.r.t. number of threshold points, how fine each range is, etc.

[1] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees

Decision Tree: Some Comments

- Other alternatives to entropy for judging feature informativeness in DT classification?
  - Gini-index $\sum_{c=1}^{C} p_c(1 - p_c)$ is another popular choice
- For DT regression (Regression Trees[1]), can split based on the variance in the outputs, instead of using entropy (which doesn't make sense for real-valued inputs)
- Real-valued features (we already saw some examples) can be dealt with using thresholding
  - Need to be careful w.r.t. number of threshold points, how fine each range is, etc.
- More sophisticated decision rules at the internal nodes can also be used (anything that splits the inputs at an internal node into homogeneous groups; e.g., a machine learning classification algo)

[1] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees

Decision Tree: Some Comments

- Other alternatives to entropy for judging feature informativeness in DT classification?
  - Gini-index $\sum_{c=1}^{C} p_c(1 - p_c)$ is another popular choice
- For DT regression (Regression Trees[1]), can split based on the variance in the outputs, instead of using entropy (which doesn't make sense for real-valued inputs)
- Real-valued features (we already saw some examples) can be dealt with using thresholding
  - Need to be careful w.r.t. number of threshold points, how fine each range is, etc.
- More sophisticated decision rules at the internal nodes can also be used (anything that splits the inputs at an internal node into homogeneous groups; e.g., a machine learning classification algo)
- Need to take care handling training or test inputs that have some features missing

[1] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees

Some Aspects about Decision Trees

Some key strengths:

- Simple and each to interpret
- Do not make any assumption about distribution of data
- Easily handle different types of features (real, categorical/nominal, etc.)
- Very fast at test time (just need to check the features, starting the root node and following the DT until you reach a leaf node)
- Multiple DTs can be combined via ensemble methods (e.g., Decision Forest)
  - Each DT can be constructed using a (random) small subset of features

Some Aspects about Decision Trees

Some key strengths:

- Simple and each to interpret
- Do not make any assumption about distribution of data
- Easily handle different types of features (real, categorical/nominal, etc.)
- Very fast at test time (just need to check the features, starting the root node and following the DT until you reach a leaf node)
- Multiple DTs can be combined via ensemble methods (e.g., Decision Forest)
  - Each DT can be constructed using a (random) small subset of features

Some key weaknesses:

- Learning the optimal DT is NP-Complete. The existing algorithms are heuristics (e.g., greedy selection of features)
- Can sometimes become very complex unless some pruning is applied