Machine Learning

Decision Trees

Dan Goldwasser

dgoldwas@purdue.edu

Goal for Today's class

- Wrap-up KNN and discuss Decision trees

- Intuition about hypothesis space and their expressiveness
- "Real learning": creating a model from data (unlike KNN that just memorized it)
- Overfitting vs. Underfitting in DT

Quick Review: The Hypothesis Space

- What functions should the learning algorithm consider?
  - **Does not** have to search over the same space as f(x)
  - Should it be a **more** expressive space? **Less** expressive?

- **Does it matter?**

Quick Review: Is learning possible?

- How many Boolean functions are there over 4 inputs?
- 2^{16} = 65536 functions (**why?**)
  - 16 possible outputs.
  - Two possibilities for each output
- Without any data, 2^{16} options

- ***Does the data identify the right function?***
- The training data contains 7 examples
  - We still have 2^9 options,

Is learning even possible?

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	**0** $\leftarrow$
0	0	1	1	**1** $\leftarrow$
0	1	0	0	**0** $\leftarrow$
0	1	0	1	**0** $\leftarrow$
0	1	1	0	**0** $\leftarrow$
0	1	1	1	?
1	0	0	0	?
1	0	0	1	**1** $\leftarrow$
1	0	1	0	?
1	0	1	1	?
1	1	0	0	**0** $\leftarrow$
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

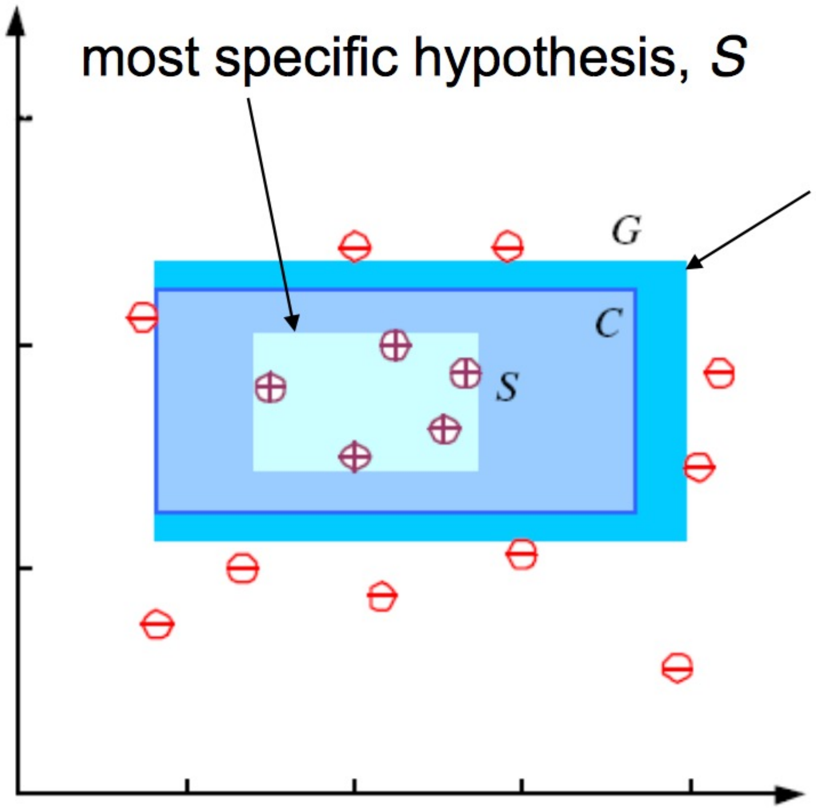
Quick Review: Learning

- Learning is removal of remaining uncertainty
- **Finding a good hypothesis class is essential!**
  - You can start small, and enlarge it until you can find a hypothesis that fits the data (similar argument: "grow" the **feature space**)

Question: *Can there be more than one function that is consistent with the data*? **How do you choose between them?**

A-priori preference towards some functions can be coded in different ways, often via the learning algorithm

Quick Review : hypothesis space and version space



$h \in$ H, between S and G is consistent

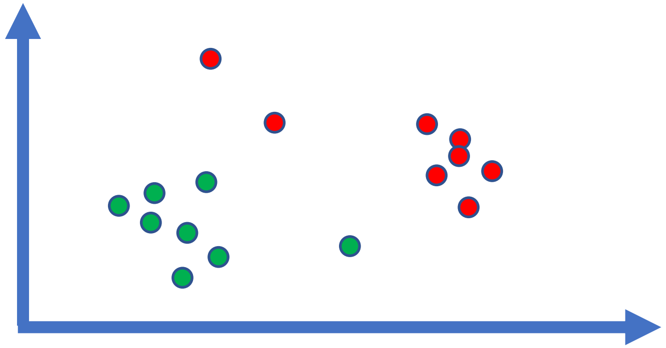
and make up the version space

(Mitchell, 1997)

Quick Review: KNN

Categorical attributes

Numerical Attributes:

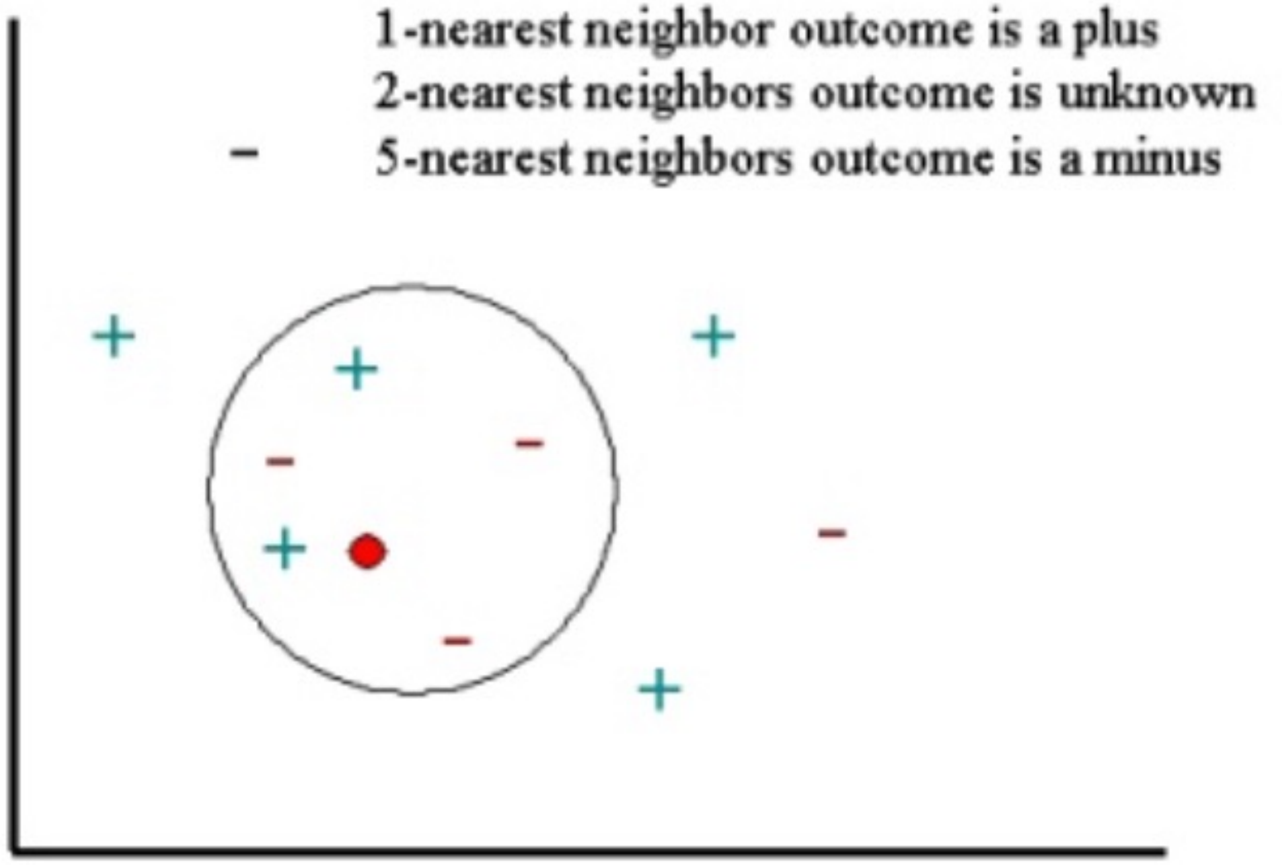


Distance Measures

$$d(x_1, x_2) = \sqrt[2]{(x_1 - x_2)^2}$$

$$d(x_1, x_2) = 1 - \frac{x_1 \bigcap x_2}{x_1 \bigcup x_2}$$

Quick Review: KNN



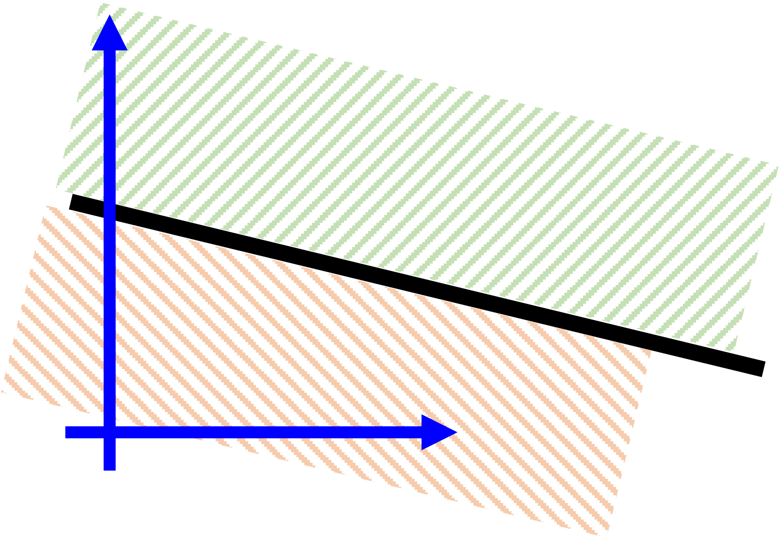
Quick Review: Let’s analyze KNN

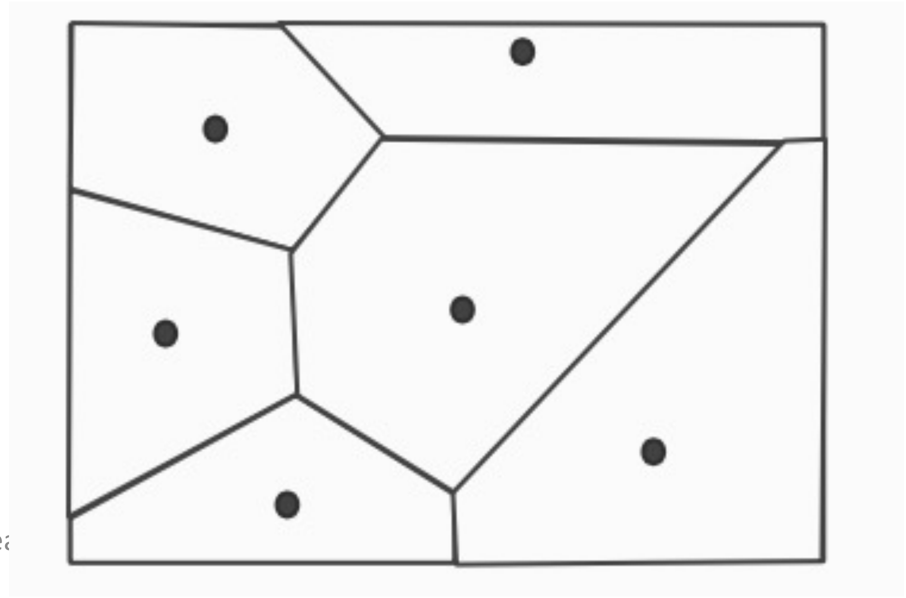
- **Time Complexity**
  - *Training is very fast*
  - *Prediction is slower..*
    - *O(dN) for N examples, of dimensionality d*
    - *Increases with number of training examples!*
- Space Complexity
  - *KNN needs to maintain all training examples*

- *Let’s Compare to the m-of-n rules*

Quick Review: Expressivity

- KNN Can learn very complex decisions models
- We can try to characterize the learned function using its ***decision boundary***
  - *Visualize which elements will be classified as positive/negative*
  - *Decision Boundary is the curve separating the negative and positive regions*

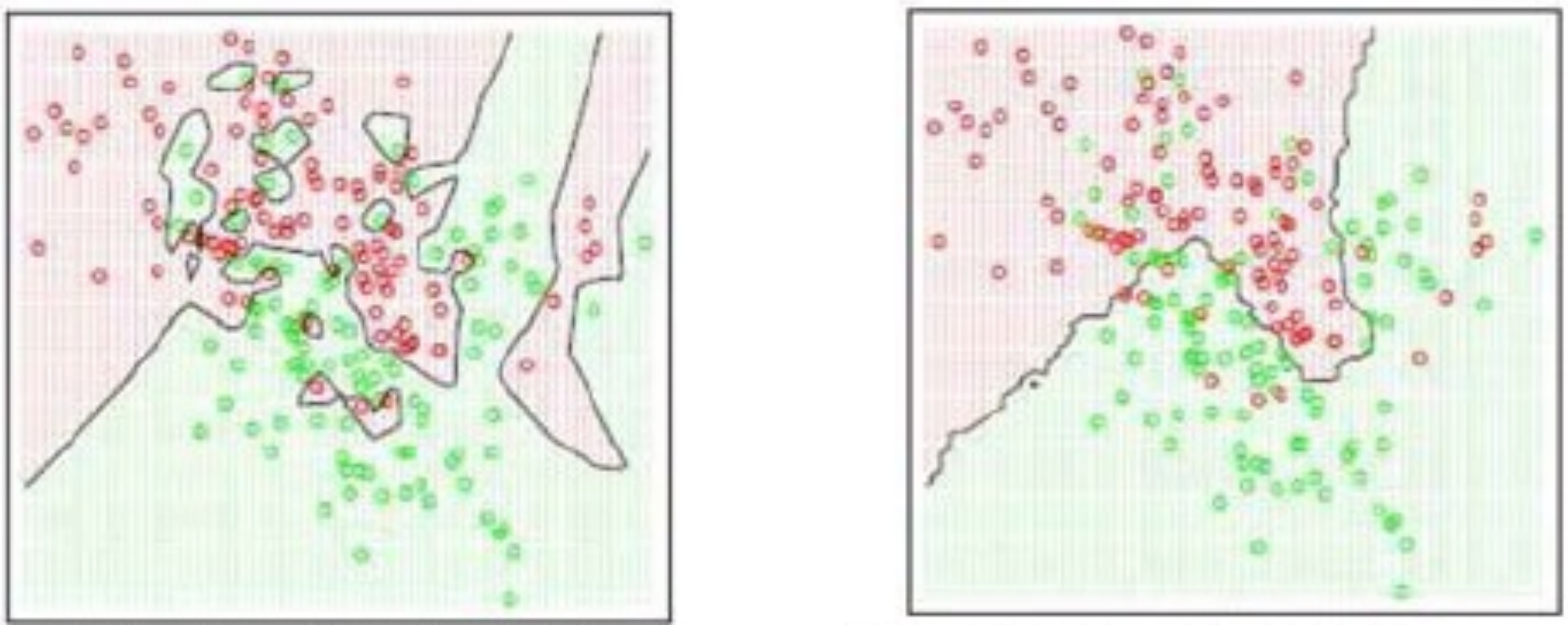




Quick Review: Expressivity

Let's take a closer look at the learned function

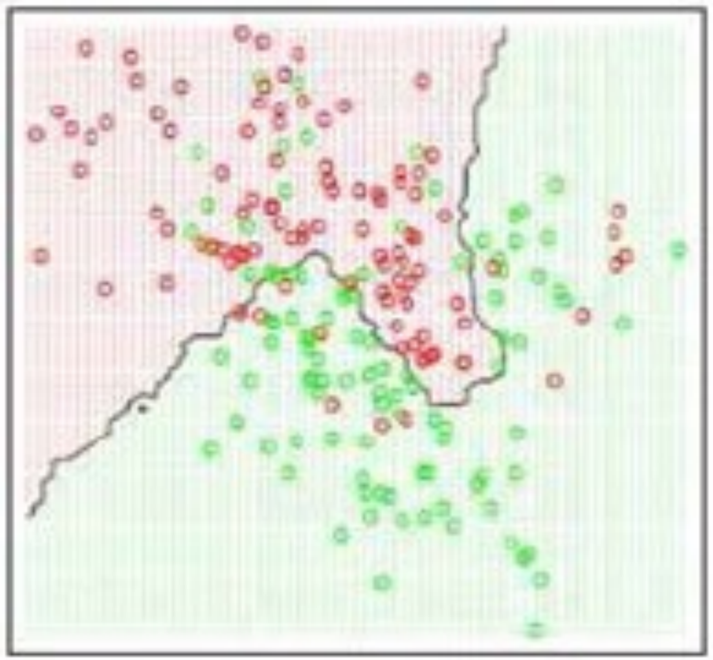
→ ***High sensitivity to noise!***



Higher k values results in smoother decision boundaries

Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Question



Higher k values results in smoother decision boundaries

What will happen if we keep increasing K?

→ Extreme scenario: if k = N?

How should we set the value of K?

- **Option** 1: Pick the K that minimizes the training error.
  - Training Error: number of errors on the training set, *AFTER* we learned a classifier

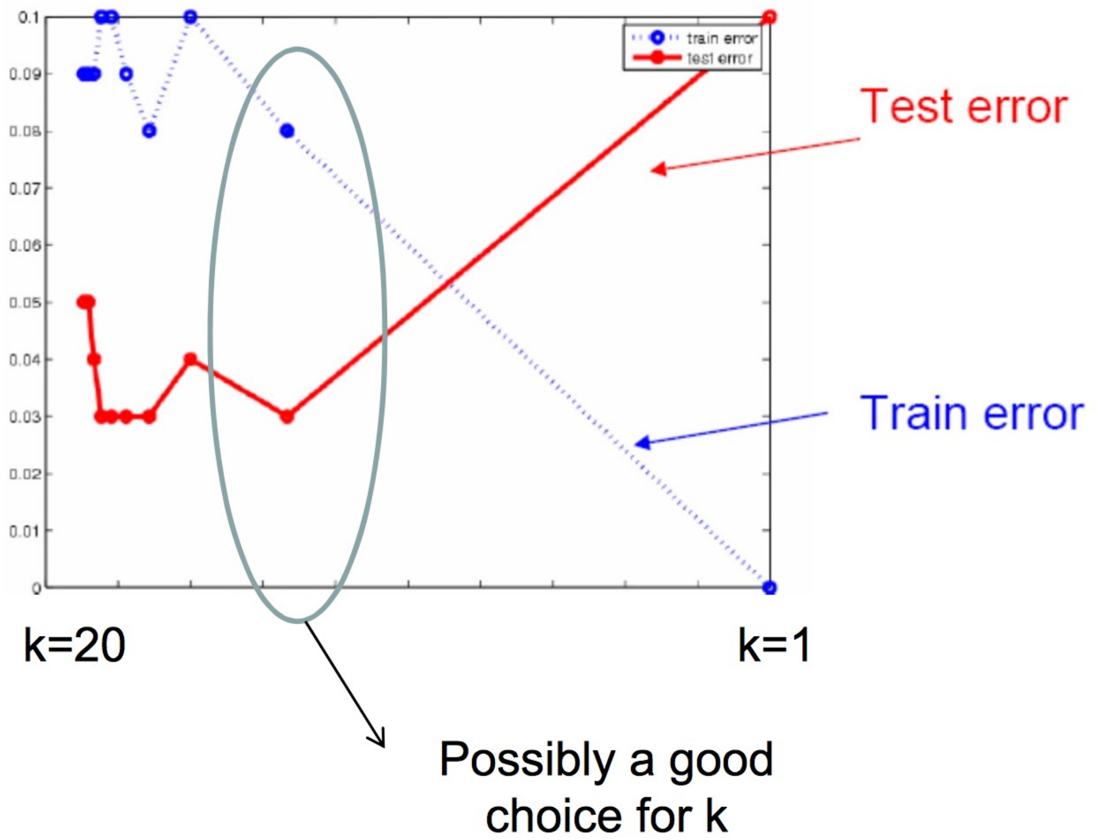
- **Question**: *What is the training error of 1-NN? 10-NN?*

- **Option** 2: choose k to minimize mistakes on test error
  - Test Error: number of errors on the test set, after we learned a classifier

- ***Note**: It's better to use a validation set, and not the test set. More on that later in the class..*

How should we set the value of K?

How would the test and train error change with K?



In general – using the training error to tune parameters will always result in a more complex hypothesis! **(why?)**

KNN Practical Considerations

- Very simple to understand and implement
- An odd value for k is better (why?)
- How can we find the right value for k?
  - Using a held out set
- Feature normalization is important! (why?)
  - Different scales for different features

KNN Practical Considerations

- Choosing the right features is important
  - Very sensitive to irrelevant attributes
  - Sensitive to the number of dimensions
- Choosing the right distance metric is important

– Euclidean distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

– Manhattan distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_1 = \sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$

– L_p-norm

- Euclidean = L_2
- Manhattan = L_1
- Exercise: What is L_∞?

$$||\mathbf{x}_1 - \mathbf{x}_2||_p = \left(\sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p\right)^{\frac{1}{p}}$$

Inductive Bias

- Unbiased learning is impossible!
- Inductive bias: a set of assumptions guiding learning **beyond the data**
- We have seen one type of bias: **Language bias**
  - **Pick the right hypothesis space**
- Today, as part of our discussion on Decision Trees we will introduce a second type – **search bias.**
  - **Similar objective**: Encode assumptions about learning, restrict the complexity of the resulting hypothesis

K Nearest Neighbors

- *Is KNN really a learning algorithm?*
  - *Yes, because…*
    - ***Well, we get a classifier out of it..***
  - *No, because…*
    - ***Well, it just maintains the data..***
- **Today we will look at another algorithm that only maintains the data**
  - **But in a compressed way ("lossy or lossless")**

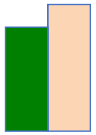
Learning Decision Trees

- KNN only stored the data, Decision trees store a "*compressed*" dataset.
  - Simplified view of DT Learning : ***better Compression, with less information loss = better generalization.***

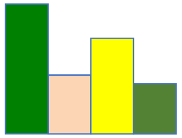
- **DT Learning overview:**
  - Decision Tree Representation
  - Algorithms for learning decision trees
  - Experimental issues
    - Controlling overfitting

Representing Data

- Think about a large table, N attributes, and assume you want to know something about the data represented as entries in the table
  - **E.g.** own an expensive car or not;
- **Simplest way**: Histogram on the first attribute – own



- **Then**, histogram on first and second (own & gender)



Representing Data

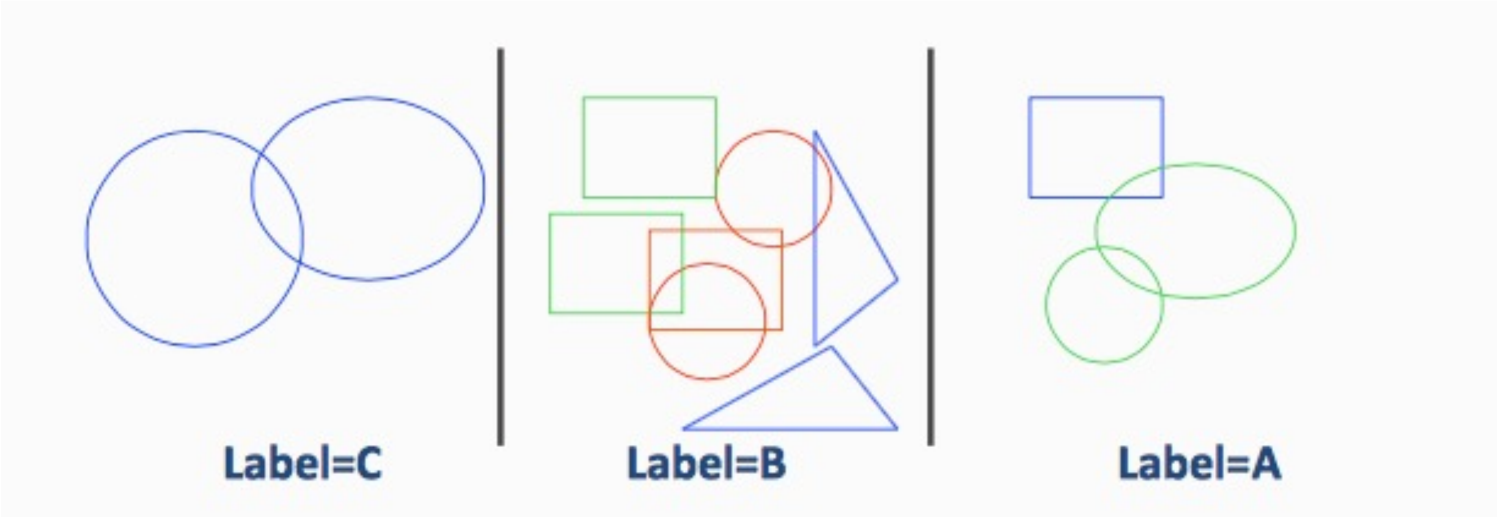
- Think about a large table, N attributes, and assume you want to know something about the data represented as entries in the table
- ***But, what if the # of attributes is larger: N=16***
  - How large are the 1-d histograms (contingency tables) ? ***16 numbers***
  - How large are the 2-d histograms? ***16-choose-2 = 120 numbers***
  - How many 3-d tables? **560 numbers**
  - **With 100 attributes, the 3-d tables need 161,700 numbers**
- *We need to figure out a way to represent data in a better way, and figure out what are the important attributes to look at first*.
  - we will use an information theoretic approach to represent the data better

Decision Trees

- A hierarchical data structure (tree) that represents data by implementing a divide and conquer strategy
- **Nodes** are tests for feature values
  - There is one branch for every value that the feature can take
- **Leaves** of the tree specify the class labels

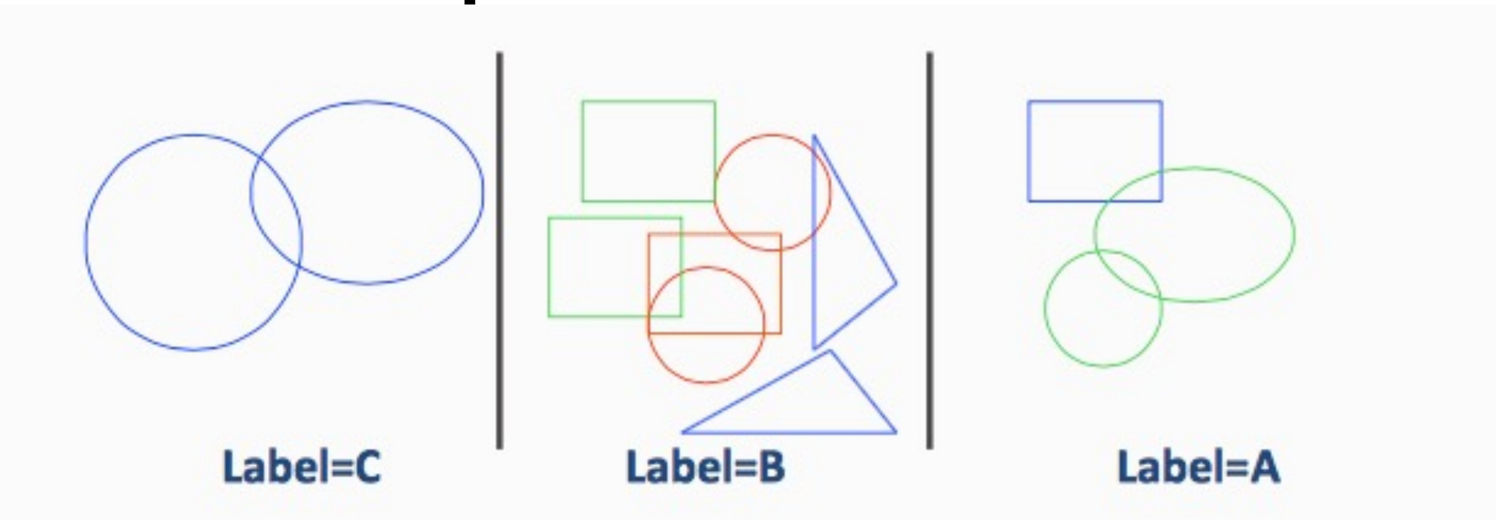
- Given a collection of examples, **learn a decision tree that represents it.**
- Use this representation to **classify new examples**
  - The tree can be used for non-parametric classification and regression

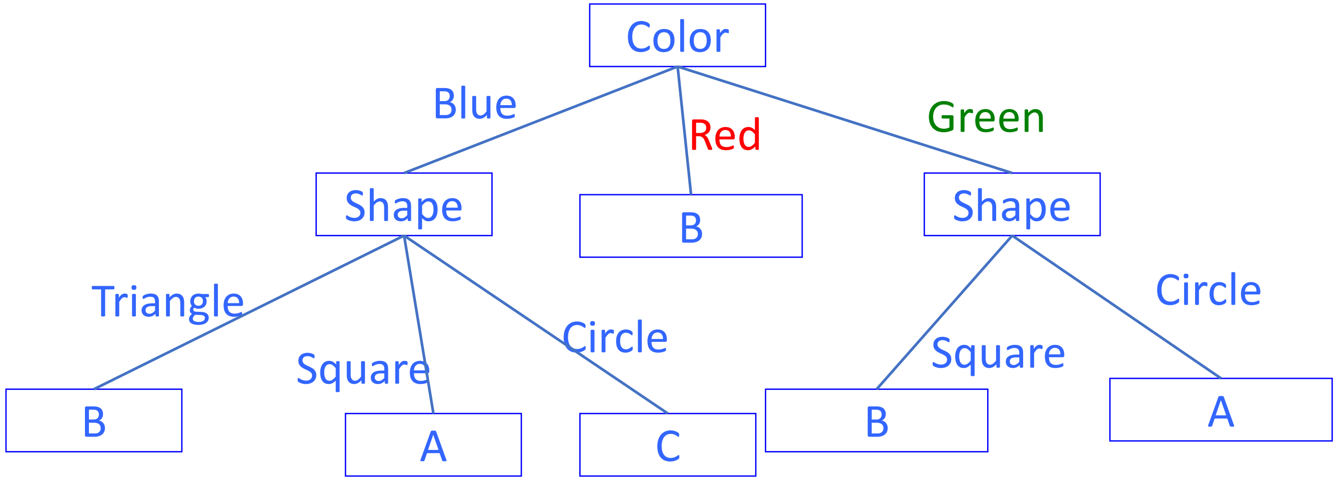
Decision Tree Representation



- Three **output** classes: A, B, C
- Two attributes
  - **Color**: Red, Blue, Green
  - **Shape**: Circle, Triangle, Square

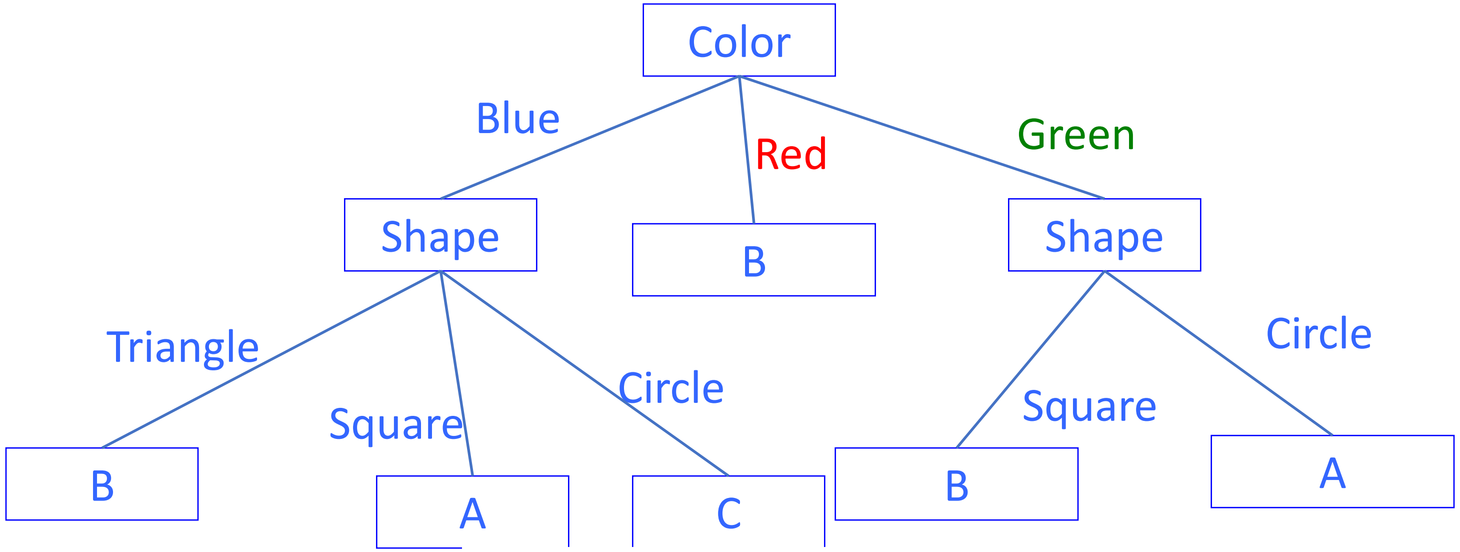
Decision Tree Representation





Decision Tree Representation

- **Basic Questions:**
  - *How to use Decision Trees for prediction?*
    - **follow the path from the root**
      - What is the label of *a red triangle*? *Green triangle*?
  - How can we learn decision trees from data?



Decision Trees

- *Output*:
  - Boolean
  - Multiclass (discrete categories)
  - Real valued (regression tree)

- *Concise representation of the training data*
  - Simply ”organizing” the training data in tree for will severely overfit and will be very sensitive to noise (***why***?)
  - Methods for handling noisy data (noise in the label or in the features) and for handling missing attributes
  - **Pruning trees** helps with noise

Expressivity of Decision Trees

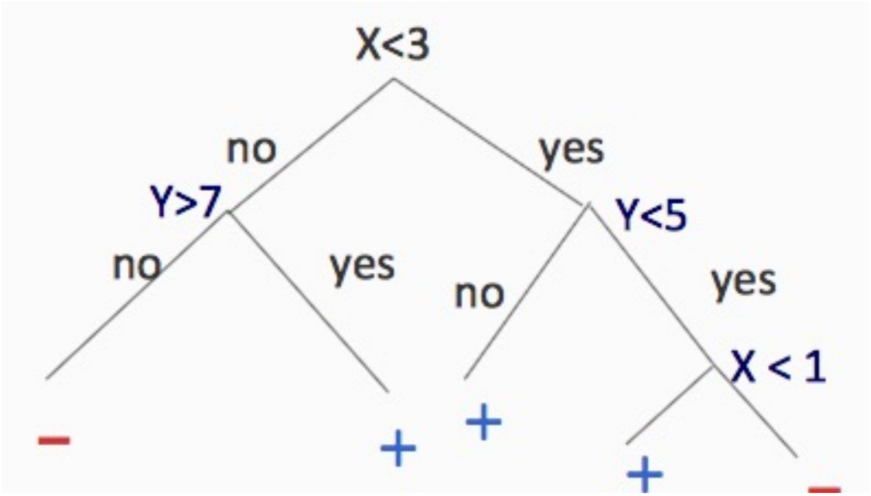
- What kind of Boolean functions can DT represent?
  - Any Boolean function! (***why?***)
- A decision tree can be rewritten as a DNF
  - *Each path from root to leaf can be written as a rule, the tree is a disjunction of these rules.*

  - **`Green ^ square` ➔ `positive`**
  - **`Blue ^ circle` ➔ `positive`**
  - **`Blue ^ square` ➔ `positive`**

- **Question**: *What is the size of the hypothesis space for the shape classification problem?*

Numeric Attributes

- We have seen instances represented as attribute-value pairs (color=blue, second letter=e, etc.)
  - Values have been categorical
- **What about numeric feature values? (e.g., length)**
  - Discretize them or use thresholds on the numeric values

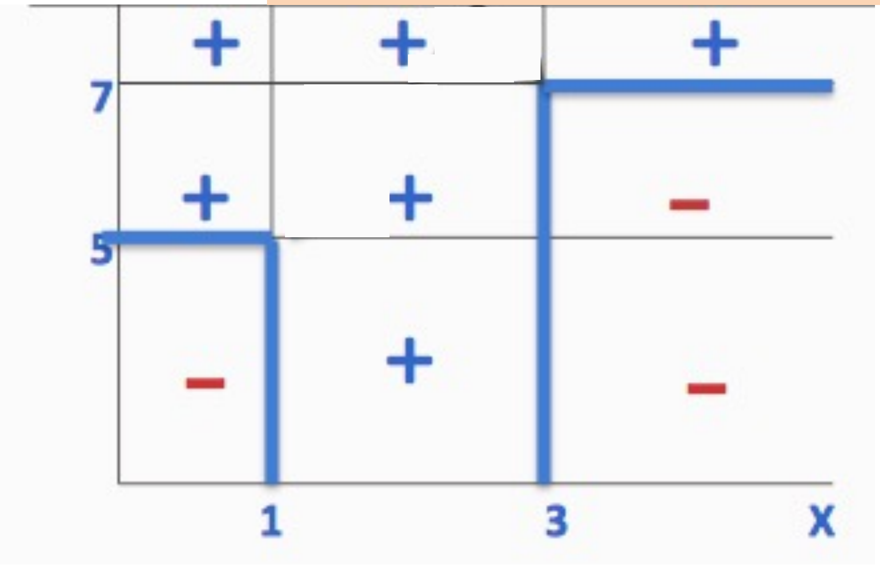


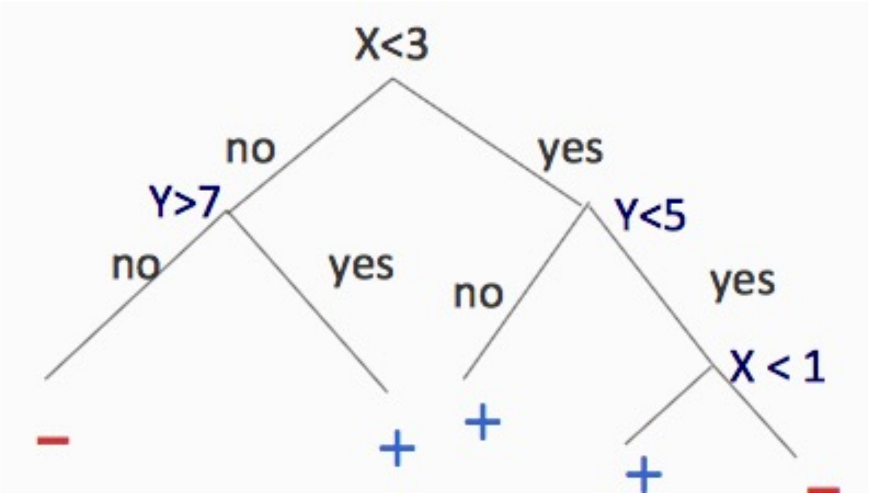
Expressivity

- **What about numeric feature values? (e.g., length)**
  - Discretize them or use thresholds on the numeric values
  - ***This divides the feature space into axis parallel rectangles***

This is the decision boundary for DT

How would you characterize the complexity of a hypothesis?





Summary: Decision Tree Representation

- ***Decision tree can represent any Boolean function***
  - A way to represent lots of data
    - **Compact representation**
  - A natural representation (think 20 questions)
    - "**Explainable**" model!
- *Prediction with a decision tree is easy*
- Clearly, given a dataset, there are many decision trees that can represent it. (**why?**)
  - **How can you learn a good representation from data?**

Example: Will I play tennis today?

- **Features**
  - *Outlook*: {Sun, Overcast, Rain}
  - *Temperature*: {Hot, Mild, Cool}
  - *Humidity*: {High, Normal, Low}
  - *Wind*: {Strong, Weak}

- **Labels**
  - *Binary classification* task: Y = {+, -}

Basic Decision Tree Learning Algorithm

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny), O(vercast), R(ainy)

Temperature: H(ot), M(edium), C(ool)

Humidity: H(igh), N(ormal), L(ow)

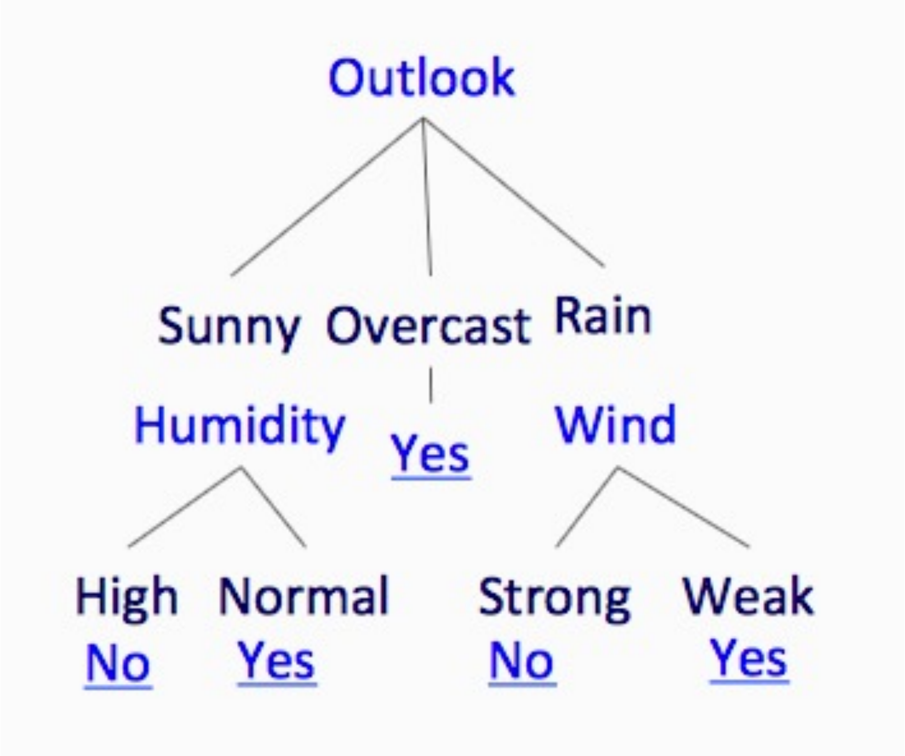
Wind: S(trong), W(eak)

Basic Decision Tree Learning Algorithm

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

- Data is processed in Batch (i.e. all the data available)

- Recursively build a DT top down.



Basic Decision Tree Learning Algorithm: ID3

- 1. **If** all examples are have same label:
  - *Return a single node tree with the label*
- 2. **Otherwise**
  - Create a Root node for tree
  - A = attribute in Attributes that ***best*** classifies S
  - for each possible value v of attribute A:
    - Add a new tree branch corresponding to A=v
    - Let *Sv* be the subset of examples in S with A=v
    - **if** *Sv* is empty:

      add leaf node with the common value of Label in S
    - **Else:**

      below this branch add the subtree:

      ID3(*Sv*, Attributes - {a}, Label)
- 4. **Return** Root node

Input:
S the set of Examples
Label is the target attribute (the prediction)
Attributes: set of measured attributes

Why?

Recursive call

Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (*Occam's Razor*)
  - But, finding the minimal decision tree consistent with the data is NP- hard
- The recursive algorithm is a greedy heuristic search for a simple tree, but **cannot guarantee optimality**
- The main decision in the algorithm is the selection of the next attribute to split on

Picking the Root Attribute

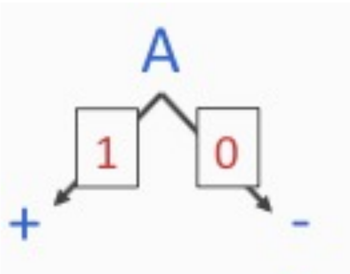
Consider data with two Boolean Attributes (A,B).

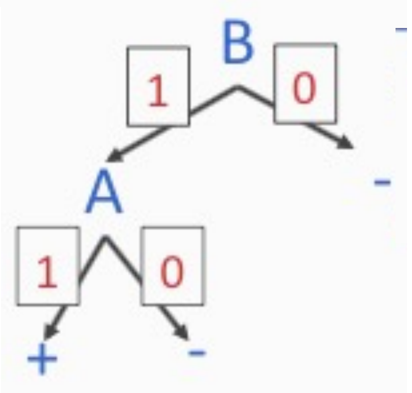
<(A =-0, B=0), - >: 50 Examples
<(A =-0, B=1), - >: 50 Examples
<(A =-1, B=0), - >: 0 Examples
<(A =-1, B=1), + >: 100 Examples

What should be the first attribute to split on?

Split on A: purely labeled nodes

Split on B: nodes are not purely labeled





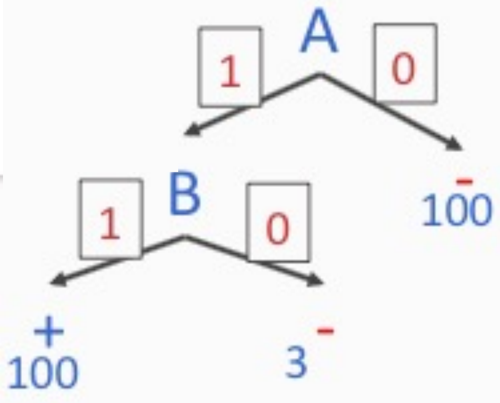
What if we have: ***"<(A=1,B=0), - >: 3"*** *examples?*

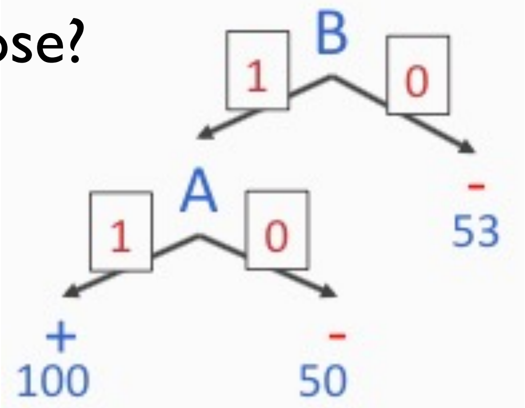
Picking the Root Attribute

Consider data with two Boolean Attributes (A,B).

<(A =-0, B=0), - >: 50 Examples
<(A =-0, B=1), - >: 50 Examples
<(A =-1, B=0), - >: *3 Examples*
<(A =-1, B=1), + >: 100 Examples

The trees look structurally similar,
which attributes should you choose?





We need a way to quantify this preference!

Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (Occam's Razor)

- The main decision in the algorithm is the selection of the next attribute to condition on

- **We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node.**

- **The most popular heuristic is based on information gain, originated with the ID3 system of Quinlan.**

Entropy

Entropy (impurity, disorder) of a set of examples S with respect to binary classification is

$$\boxed{Entropy(S) = H(S) = -p_+ \log(p_+) - p_- \log(p_-)}$$

- *The proportion of positive examples is p_+*
- *The proportion of negative examples is p_-*

In general, for a discrete probability distribution with K possible values, with probabilities {p_1, p_2, , p_K} the entropy is given by

$$\boxed{H(\{p_1, p_2, \cdots, p_K\}) = -\sum_{i=1}^{K} p_i \log(p_i)}$$

Entropy

Entropy (impurity, disorder) of a set of examples S with respect to binary classification is

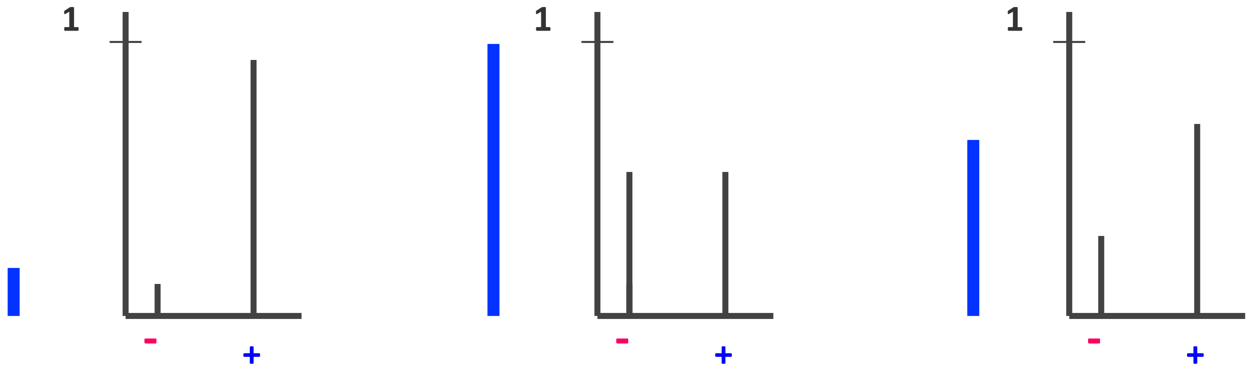
$$\boxed{Entropy(S) = H(S) = -p_+ \log(p_+) - p_- \log(p_-)}$$

- If all examples belong to the same category, entropy = 0
- If $p_+ = p_- = 0.5$, entropy = 1

Entropy

Entropy (impurity, disorder) of a set of examples S with respect to binary classification is

- If all examples belong to the same category, entropy = 0
- If $p_+ = p_- = 0.5$, entropy = 1



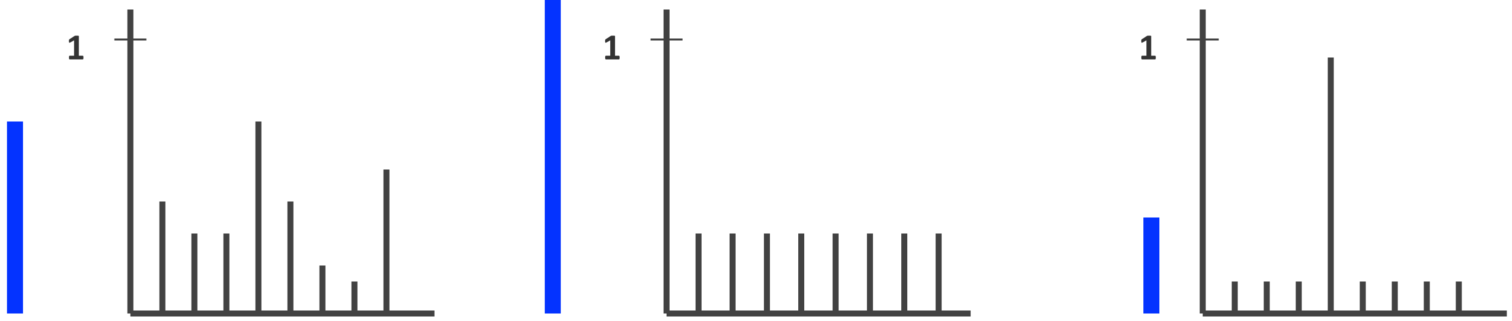
Entropy

Entropy (impurity, disorder) of a set of examples S with respect to binary classification is

The uniform distribution has the highest entropy

High Entropy – High level of Uncertainty

Low Entropy – No Uncertainty.



Information Gain

The *information gain* of an attribute A is the *expected reduction in entropy* caused by partitioning on this attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

S_v: the subset of examples where the value of attribute A is set to value v

Entropy of partitioning the data is calculated by weighing the entropy of each partition by its size relative to the original set

- partition of low entropy (imbalanced splits) leads to high gain

Go back to check which of the A, B splits is better!

Will I play tennis today?

1	**O**	**T**	**H**	**W**	**Play?**
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Outlook: S(unny), O(vercast), R(ainy)

Temperature: H(ot), M(edium), C(ool)

Humidity: H(igh), N(ormal), L(ow)

Wind: S(trong), W(eak)

Will I play tennis today?

1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Current entropy:

$p = 9/14$

$n = 5/14$

$H(Y) =$

$-(9/14)\log_2(9/14) \quad -(5/14)\log_2(5/14)$

$= 0.94$

Information Gain: outlook

1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Outlook = sunny: 5 of 14 examples

$p = 2/5$ $n = 3/5$ $\mathbf{H_S = 0.971}$

Outlook = overcast: 4 of 14 examples

$p = 4/4$ $n = 0$ $\mathbf{H_o = 0}$

Outlook = rainy: 5 of 14 examples

$p = 3/5$ $n = 2/5$ $\mathbf{H_R = 0.971}$

Expected entropy:

$(5/14)\times 0.971 + (4/14)\times 0$
$+ (5/14)\times 0.971 = \mathbf{0.694}$

Information gain:

$0.940 - 0.694 = \mathbf{0.246}$

Information Gain: outlook

1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Humidity = high:

$p = 3/7$ $n = 4/7$ $\mathbf{H_h = 0.985}$

Humidity = Normal:

$p = 6/7$ $n = 1/7$ $\mathbf{H_o = 0.592}$

Expected entropy:

$(7/14)\times 0.985 + (7/14)\times 0.592 = \mathbf{0.7885}$

Information gain:

$0.940 - 0.7885 = \mathbf{0.1515}$

Which feature to split on?

1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Information gain:

Outlook: 0.246

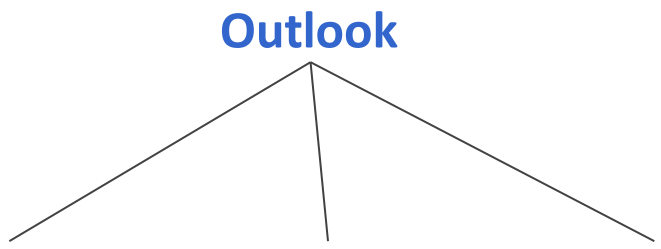
Humidity: 0.151

Wind: 0.048

Temperature: 0.029

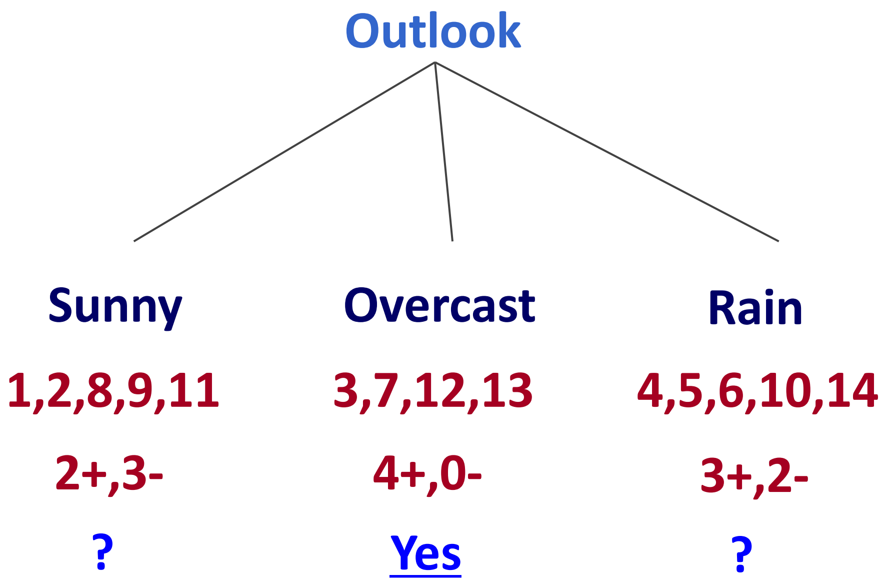
→ ***Split on Outlook***

An Illustrative Example



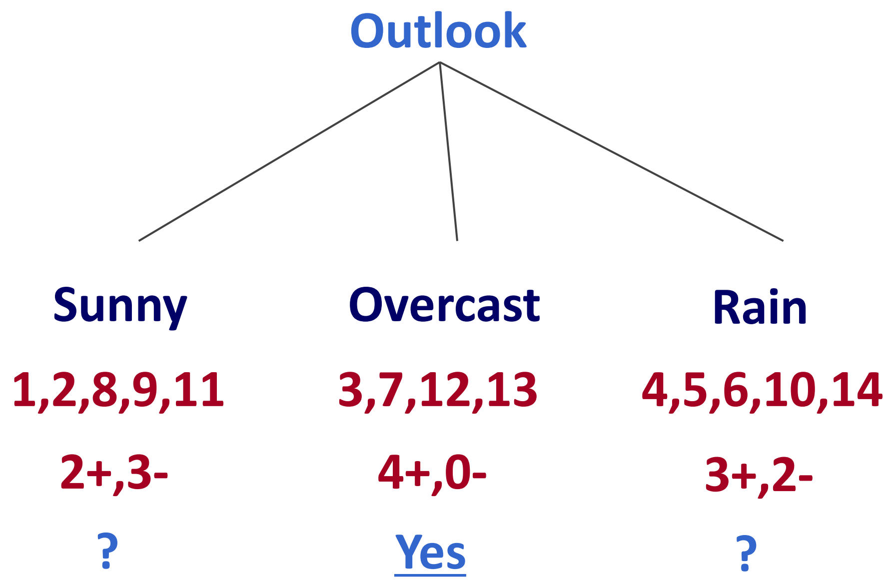
Gain(S,Humidity)=0.151
Gain(S,Wind) = 0.048
Gain(S,Temperature) = 0.029
Gain(S,Outlook) = 0.246

An Illustrative Example



1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

An Illustrative Example



1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Continue until:

- Every attribute is included in path, or,
- All examples in the leaf have same label

An Illustrative Example

Outlook

Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

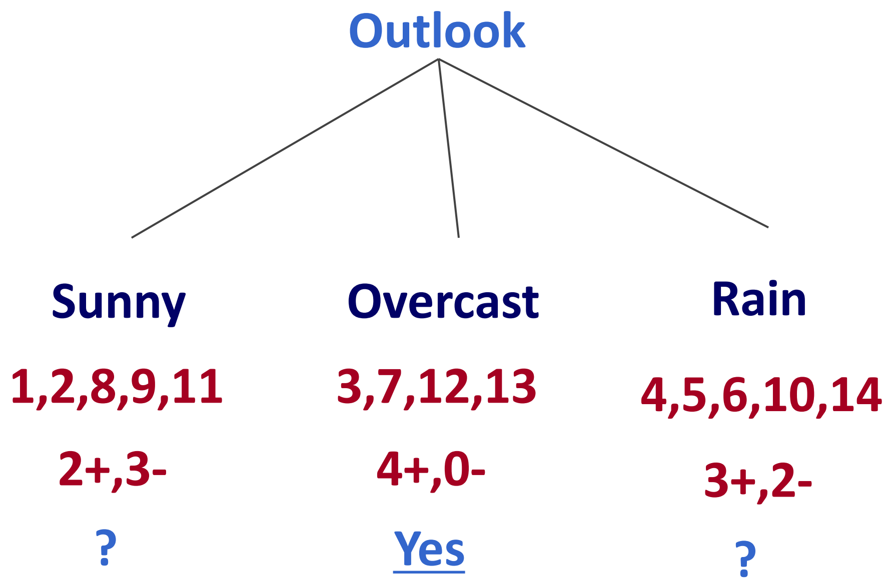
Gain(S_{sunny}, Humidity) .97-(3/5) 0-(2/5) 0= .97

Gain(S_{sunny},Temp) = .97- 0-(2/5) 1 = .57

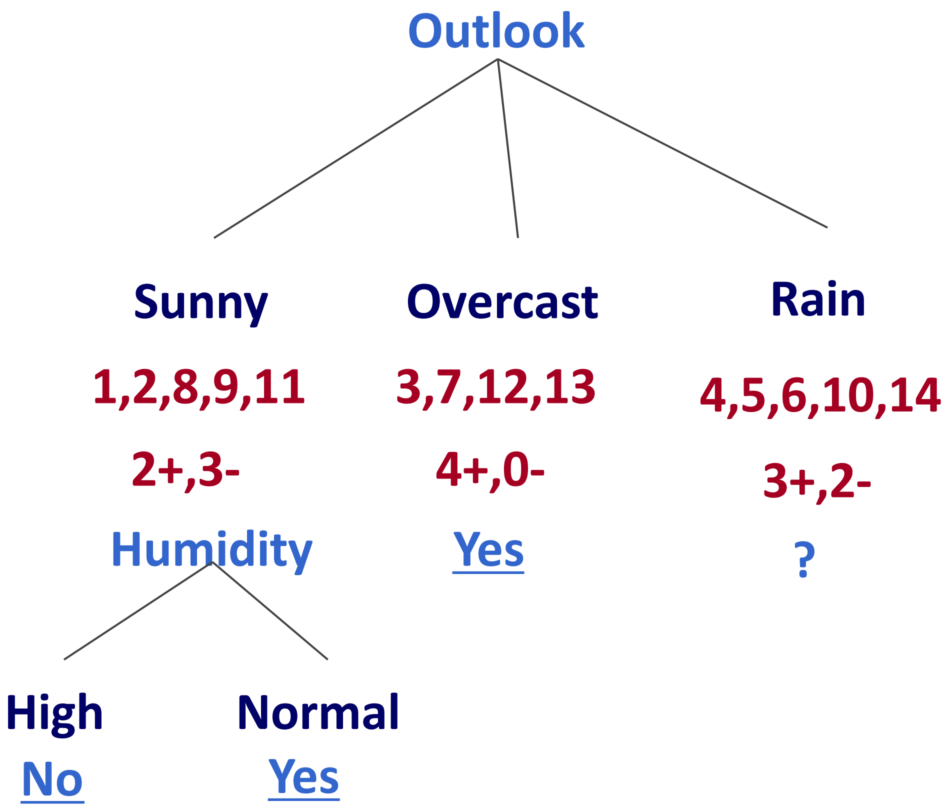
Gain(S_{sunny}, wind) = .97-(2/5) 1 - (3/5) .92 = .02

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

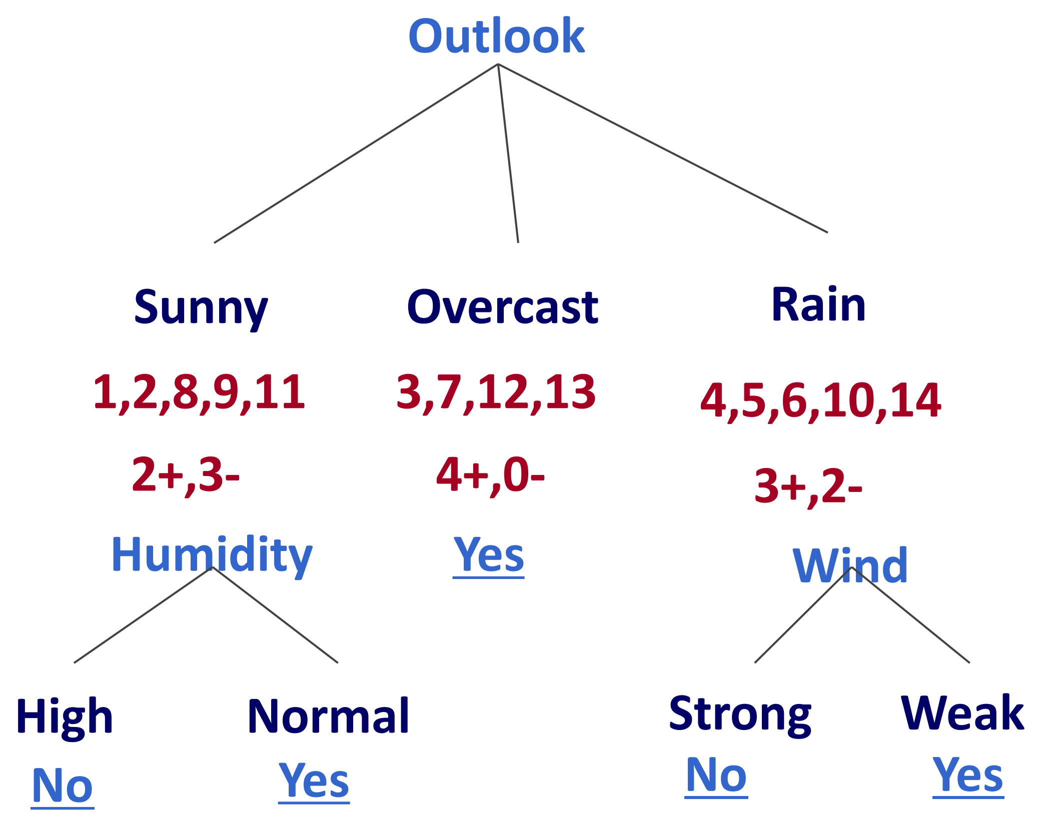
An Illustrative Example



An Illustrative Example



An Illustrative Example



induceDecisionTree(S)

- 1. Does S uniquely define a class?

  if all $s \in S$ have the same label y: **return** S;

- 2. Find the feature with the most information gain:

  $i = \text{argmax}_{i}\ \mathit{Gain}(S, X_i)$

- 3. Add children to S:

  for *k* **in** Values(X_i):

  $S_k = \{s \in S \mid x_i = k\}$

  addChild(S, S_k)

  induceDecisionTree(S_k)

  return S;

Variants of Information Gain

- Information gain is defined using entropy to measure the disorder/ impurity of the labels.

- **Other ways to measure disorde**r, e.g., *MajorityError*, which computes:
  - *"Suppose the tree was not grown below this node and the majority label were chosen, what would be the error?"*
  - Suppose at some node, there are 15 **+** and 5 **–** examples.
  - What is the *MajorityError*?
- Answer: ¼

- **Similar idea to entropy**

Missing feature values

Suppose an example is missing the value of an attribute. What can we?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	???	Weak	No
9	Sunny	Cool	High	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

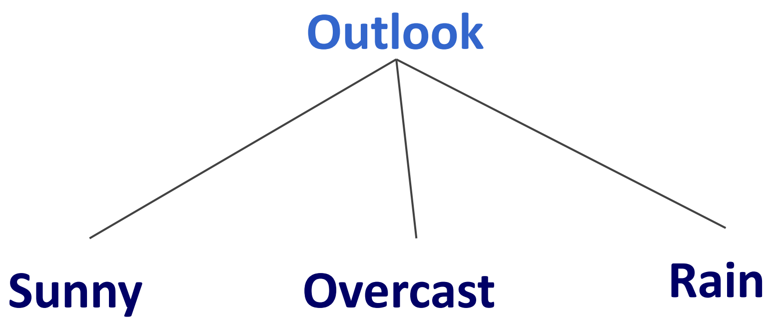
Missing feature values

- Suppose an example is missing the value of an attribute. What can we do?

- “Complete the example” by
  - Use the most common value of that attribute in data
  - Use the most common value of that attribute in data, with the same class label (at train time)
  - Assign fractional count instead of majority vote on value

Non Boolean Features

- If the features can take multiple values
  - We have seen one edge per value



Non Boolean Features

- If the features can take multiple values
  - We have seen one edge per value (i.e a multiway split)
  - Another option: Make the attributes Boolean by testing for each value

Outlook=Sunny . . [Outlook:Sunny=True,
Outlook:Overcast=False,
Outlook:Rain=False]

Non Boolean Features

What can you do with numeric features?

Use threshold *or* ranges to get Boolean tests

How should you determine the thresholds?

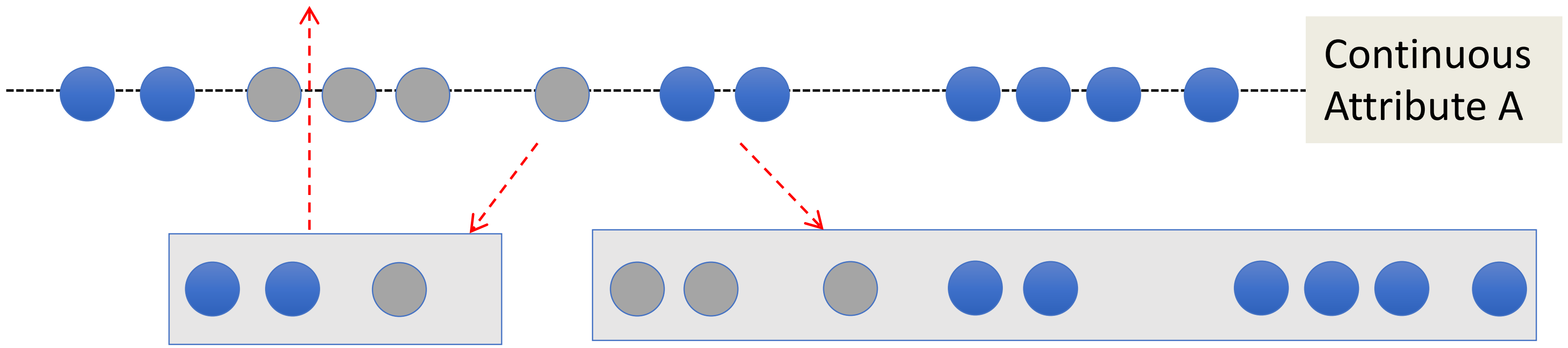
Problem:

You should consider all split points (c) to define node test $X_j > c$

<u>Is there an easier way?</u>

Continuous Attributes

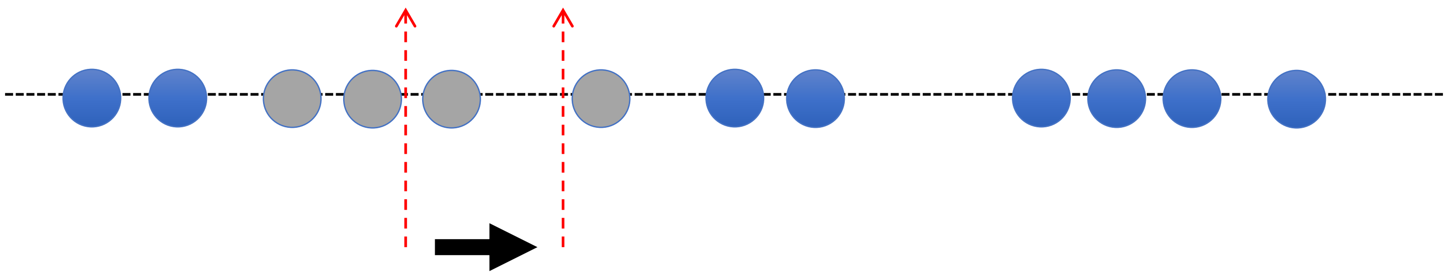
Information gain is minimized when children maintain the same distribution over output labels as the parent node



➔ The split should change the proportion of labels in the children
   ➔ Each child should have a higher proportion one of the label (different)

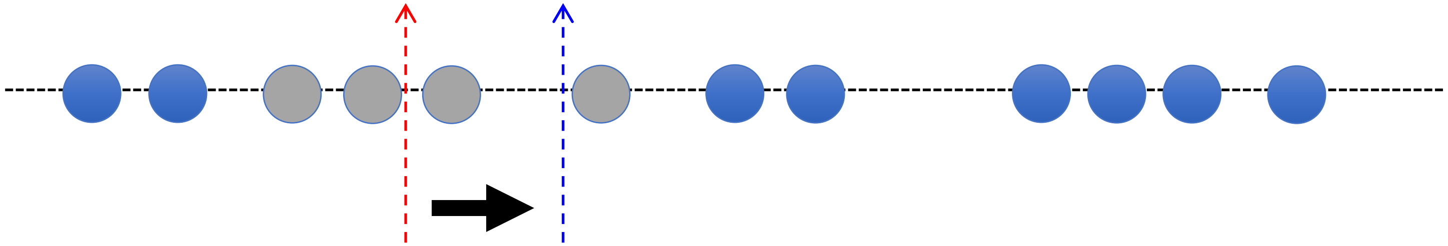
Continuous Attributes

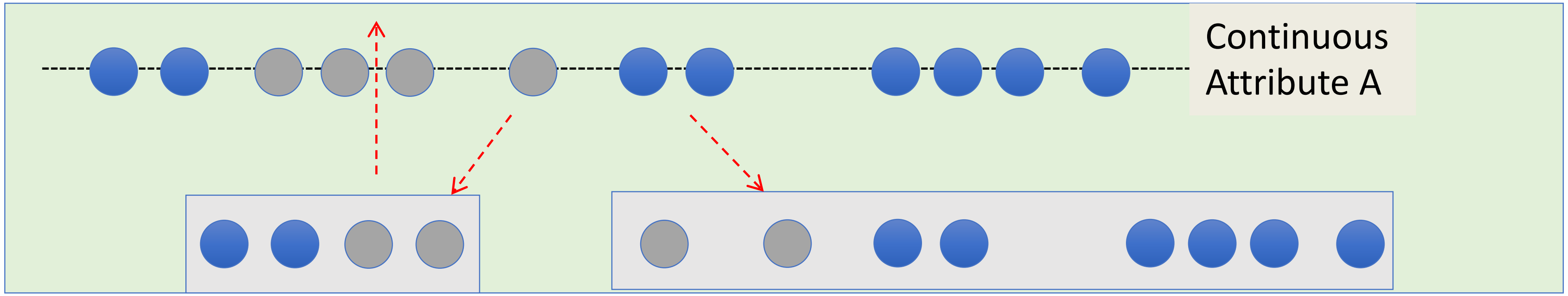
If a threshold splits two examples of the same label (say, positive), and examples on one side of the threshold (say, left), have a higher proportion than the parent distribution then:

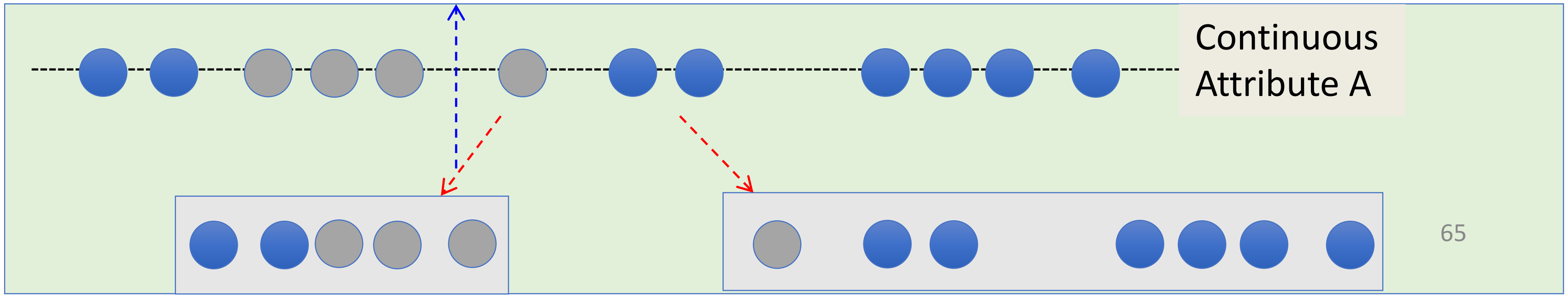


- *Examples on the other side (right) will have a higher proportion of examples with the other label (negative)*
- *Moving the threshold in this direction (right) until we get to an example with different label, will keep increasing the proportion of that label (positive/negative) in the respective children*

Continuous Attributes







Continuous Attributes

- *The highest information gain split is between examples with different labels.*
  - Simple approach: go over such split points, and find the one with highest information gain

- **Question:**

How many splits should you consider for each continuous attribute?

Bias in decision trees

- Conduct a search of the space of decision trees
  - Can represent all possible functions
- **We prefer short trees!**
  - In DT learning this is implemented as a search bias
  - We bias the search to prefer shorter trees
- Other alternatives?
  - How would you implement language bias on DT?
- Search bias is implemented using greedy heuristics
  - Hill-climbing without backtracking
  - **Overfitting can still be an issue..**

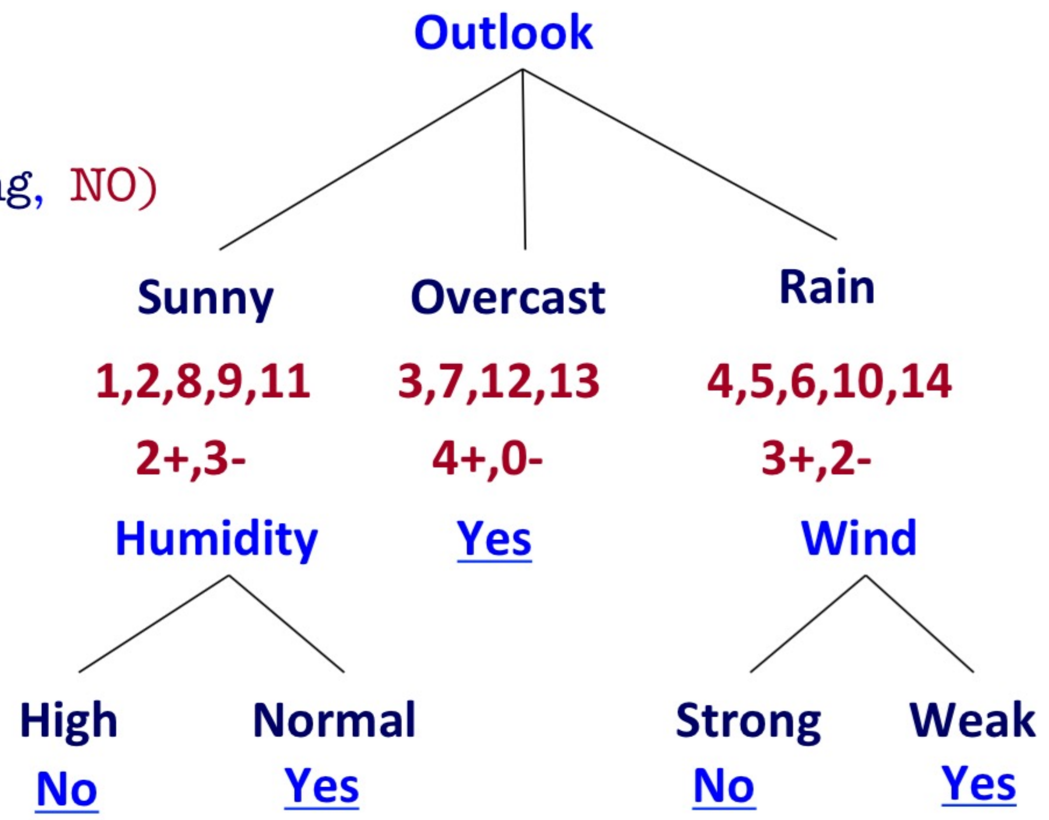
Overfitting

- Learning a tree classifying the training data perfectly may not have the best generalization
  - Algorithm fits tree to noise in training data
  - Sparse data set

A hypothesis h is said to overfit the training data if there is another hypothesis h', such that h has a smaller error than h' on the training data but h has larger error on the test data.

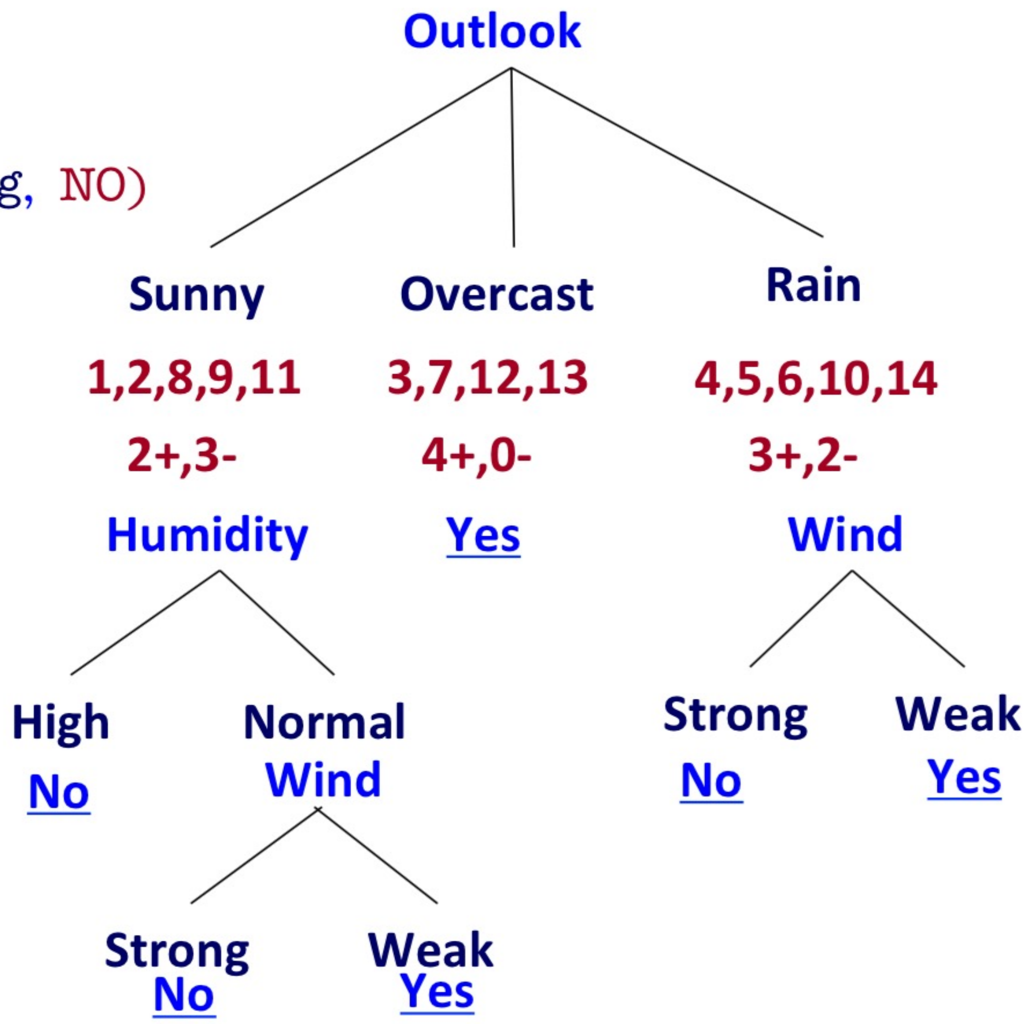
- Noisy example:

(Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO)



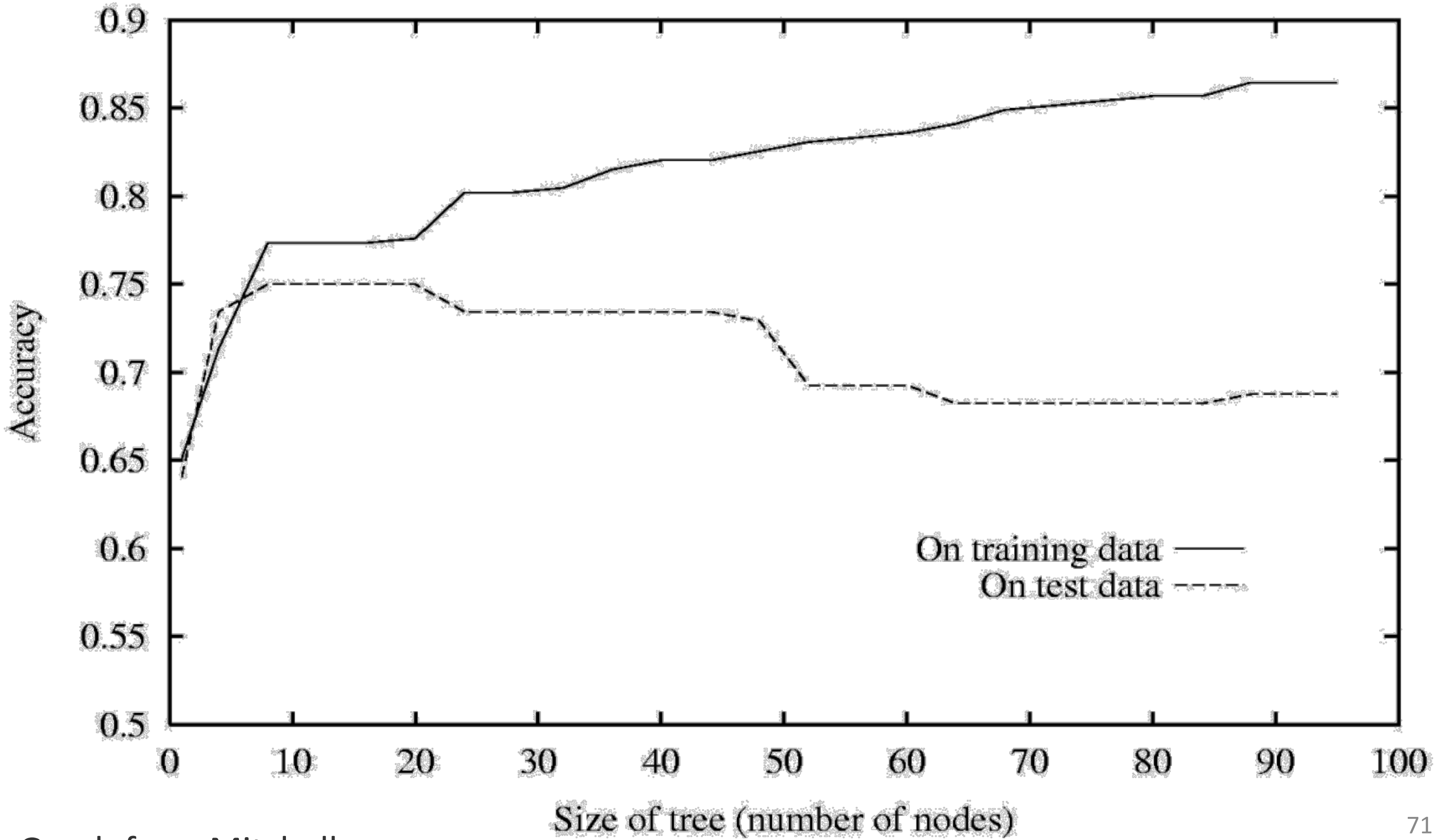
- Noisy example:

(Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO)



may fit noise or other coincidental regularities

Decision trees *will* overfit



Graph from Mitchell

Avoiding overfitting in decision trees

- **Occam's Razor**
  - Favor simpler (in this case, shorter) hypotheses
  - Fewer shorter trees, less likely to fit better by coincidence

- **Static**: Fix the depth of the tree
  - Only allow trees of size K
    - Tune K using held-out validation set
    - ***Decision stump*** = a decision tree with only one level
- **Dynamic**: optimize while growing the tree
  - Grow tree on training data
  - Check performance on held-out data after adding a new node

Avoiding overfitting in decision trees

- **Occam's Razor**
  - Favor simpler (in this case, shorter) hypotheses
  - Fewer shorter trees, less likely to fit better by coincidence

- **Post Pruning:**
- While accuracy on validation set decreases. Bottom up:
  - For each non leaf node:
    - Replace sub-tree under node by a majority vote
    - Test accuracy on validation set

Decision Trees as Features

- When learning over a large number of features, learning decision trees is difficult and the resulting tree may be very large
- **Instead of pruning you can try:**
  - learn small decision trees, with limited depth.
  - Then, learn another function over these trees

- For example, Linear combination of decision stumps

Summary

- **Very popular tool**
  - Prediction is easy (and cheap!)
  - *Expressive and easy to interpret*
  - "debugging" the model is easy!
- **Learning**: greedy heuristic for representing the data
  - ID3 based on information gain
- Prone to **overfitting**!
  - Several ways to deal with it!

Further reading

Machine Learning. Tom Mitchell.

Chapter 3

A Course in Machine Learning. Hal Daumé III.

Chapter 1

(available on line: http://ciml.info/dl/v0_9/ciml-v0_9-ch01.pdf)

Questions

- What is *inductive bias*? Why is it important?
- What is the difference between **Language** bias and **search** bias?
- *Which hypothesis space is more expressive?*
  - Boolean functions , Decision trees, linear functions
- How does tree size effect generalization?
- What is the main decision when learning DT?
- How can you deal with noise? Missing attributes? Continuous values?
- *why are decision trees popular?*
  - *When should you use them?*
- *Do you think you can implement a decision tree?*