Data mining & Machine Learning

CS 373
Purdue University

Dan Goldwasser
dgoldwas@purdue.edu

Today's Lecture

Supervised Learning!

- *Reminder – supervised vs. unsupervised learning*
- *Reminder - the KNN algorithm*
- *Your first "real" supervised learning algorithm – Decision Trees*

Descriptive vs. predictive modeling

- Descriptive models **summarize** the data
  - Provide insights into the domain
  - Focus on modeling joint distribution *P(X)*
  - May be used for classification, but prediction is not the primary goal

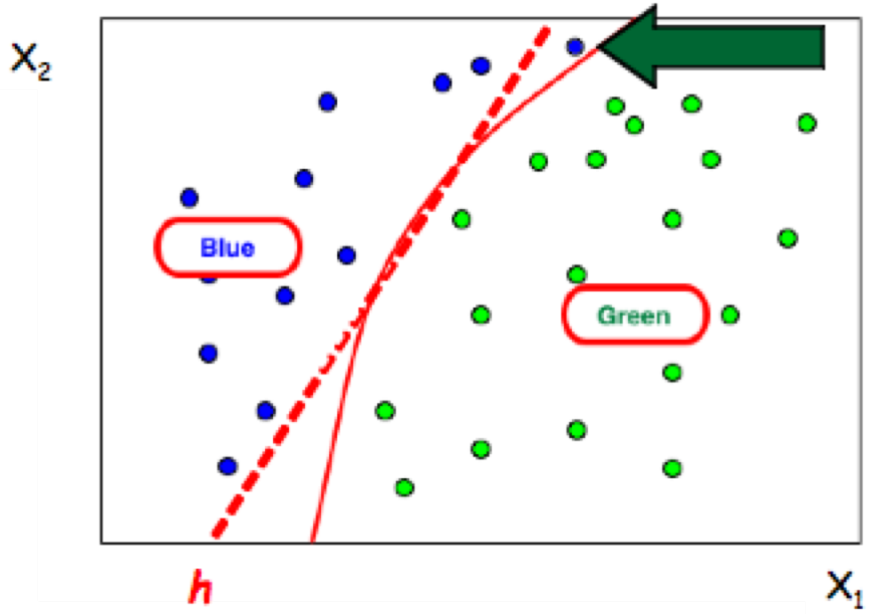
- Predictive models **predict** the value of one variable of interest given known values of other variables
  - Focus on modeling the conditional distribution *P(Y | X)* or on modeling the decision boundary for *Y*

Predictive Modeling

- **Data representation:**
  - **Training set**: Paired attribute vectors and class labels `<y(i), x(i)>`
- **Task**: estimate a predictive function $f(x;\theta)=y$
  - Assume that there is a function $y=f(x)$ that **maps** data instances (**x**) to class labels (y)
- Construct a model that approximates the mapping
  - **Classification**: if y is categorical
  - **Regression**: if y is real-valued

Classification

- In its simplest form, ***a classification model defines a decision boundary (h) and labels for each side of the boundary***
- Input: $\mathbf{x}=\{x_1,x_2,...,x_n\}$ is a set of attributes, function f assigns a label y to input **x**, where y is a discrete variable with a finite number of values



Classification output

- **Different classification tasks can require different kinds of output**
  - *Each requires progressively more accurate models (e.g., a poor probability estimator can still produce an accurate ranking)*
- **Class labels** — Each instance is assigned a single label
  - *Model only need to decide on crisp class boundaries*
- **Ranking** — Instances are ranked according to their likelihood of belonging to a particular class
  - *Model implicitly explores many potential class boundaries*
- **Probabilities** — Instances are assigned class probabilities $p(y|\boldsymbol{x})$
  - *Allows for more refined reasoning about sets of instances*

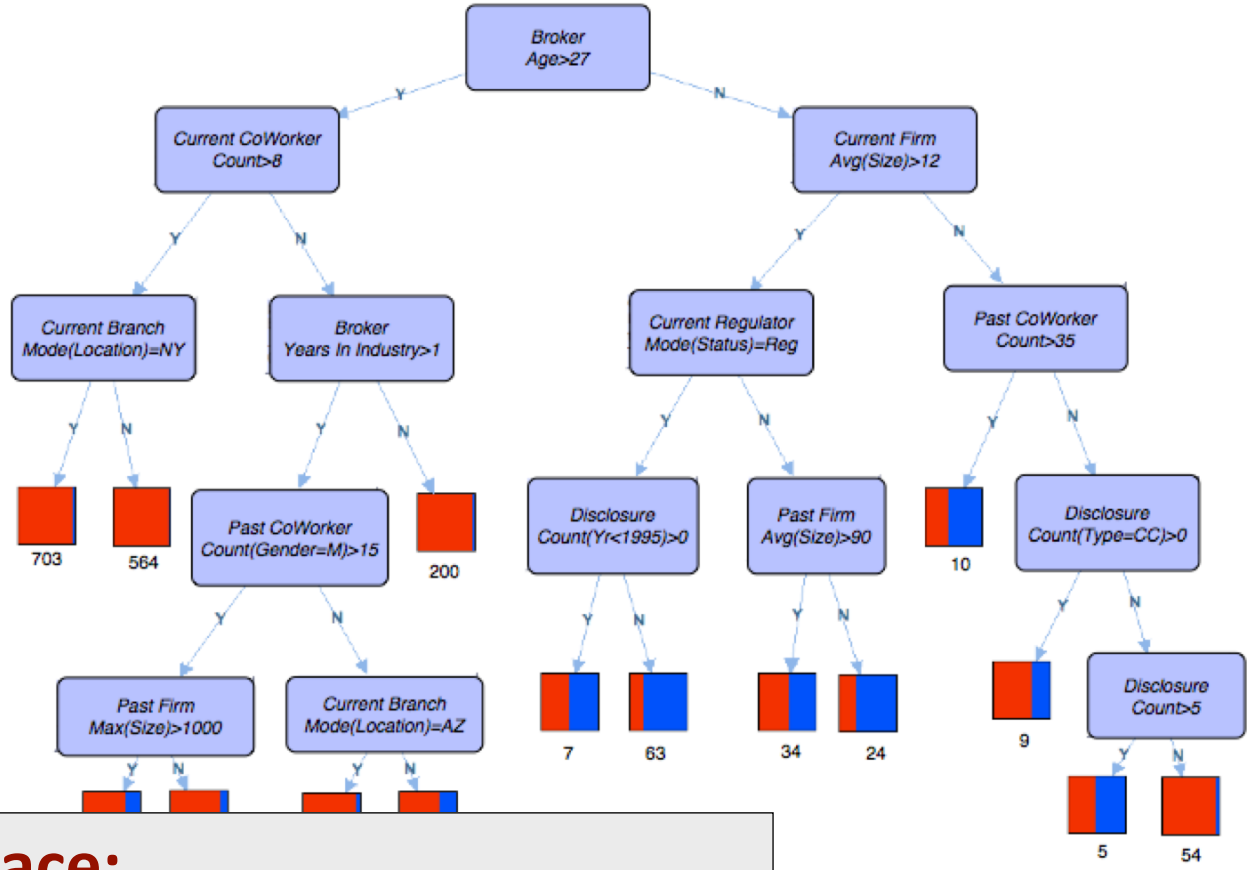
Probabilistic classification

- Model the underlying probability distributions
  - Posterior class probabilities: $p(y|\boldsymbol{x})$
  - Class-conditional and class prior: $p(\boldsymbol{x}|y)$ and $p(y)$
- Maps from inputs **x** to class label y indirectly through posterior class distribution $p(y|\boldsymbol{x})$
- Examples:
  - Naive Bayes classifier, logistic regression, probability estimation trees

Analyzing Supervised Learning Algorithms

- Similar to our previous discussions, supervised learning algorithms can be analyzed according to:
  - **Model/hypothesis space**(knowledge representation)
  - **Scoring function**
  - **Search procedure**

Classification tree



Model space:
all possible decision trees

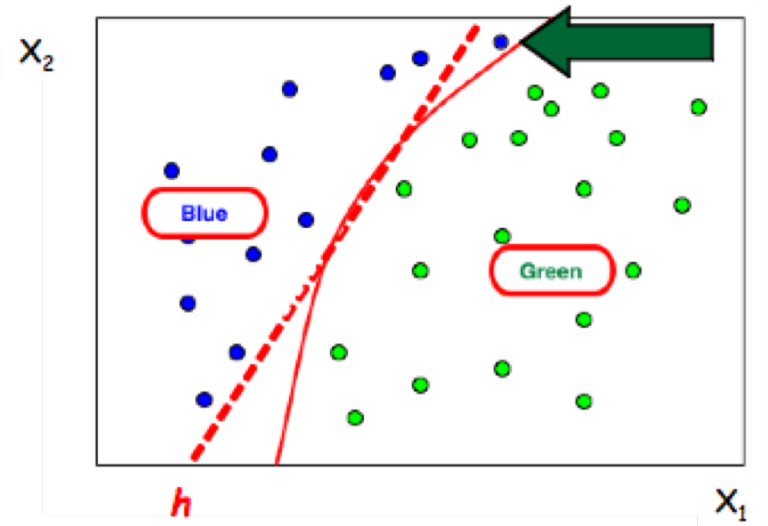
Model space

- How large is the space?
- Can we search exhaustively?
- Simplifying assumptions
  - Binary tree
  - Fixed depth
  - 10 binary attributes

Tree depth	Number of trees
1	10
2	8×10^2
3	3×10^6
4	2×10^{13}
5	5×10^{25}

Perceptron

$$f(x) = \begin{cases} 1 & \sum w_j x_j > 0 \\ 0 & \sum w_j x_j \leq 0 \end{cases}$$

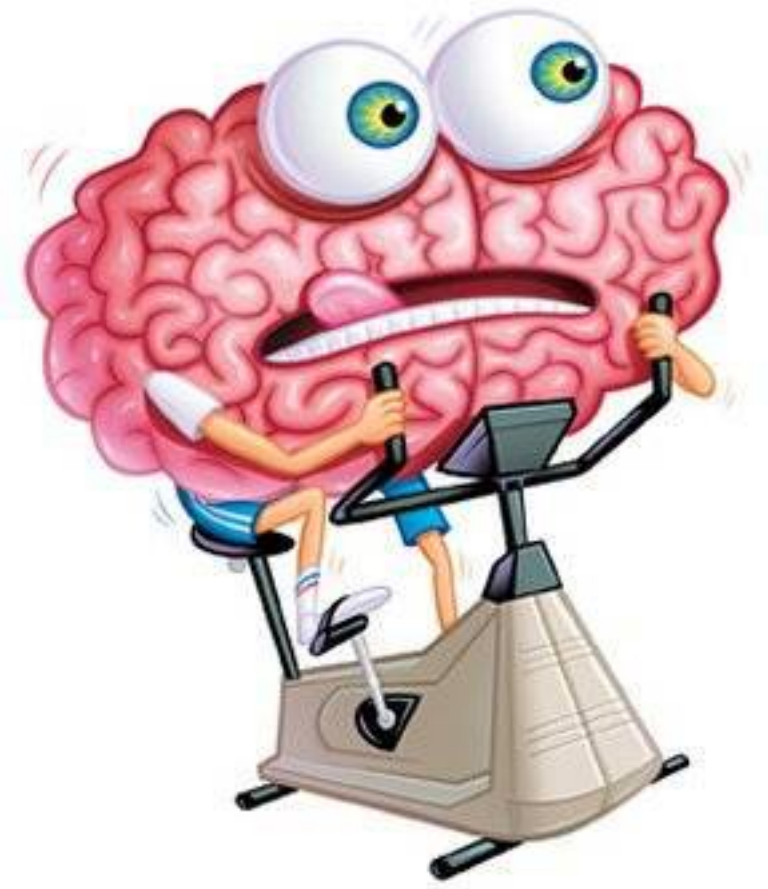


Model space:
weights *w*, for each of j attributes

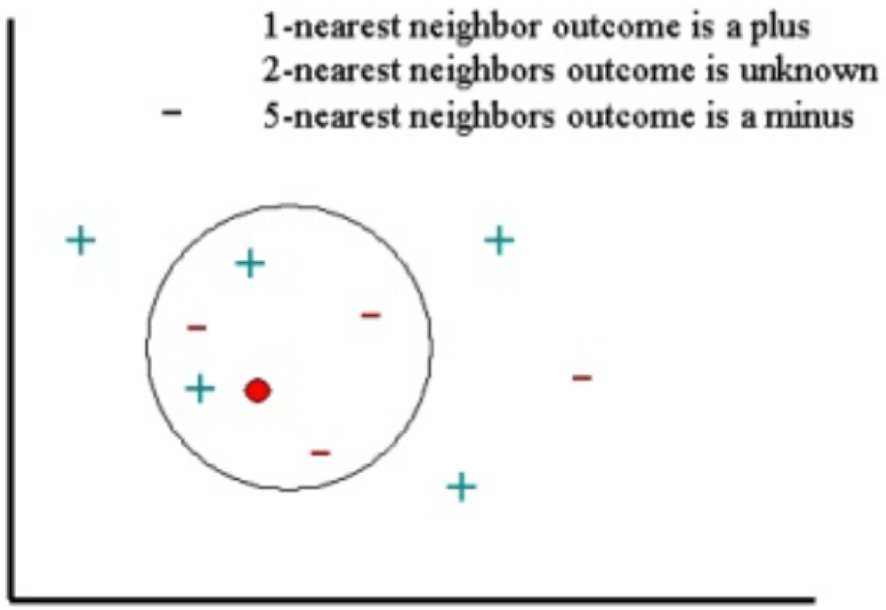
Parametric vs. non-parametric models

- **Parametric**
  - Particular functional form is assumed (e.g., Binomial)
  - **Number of parameters is fixed in advance**
  - Examples: Naive Bayes, perceptron
- **Non-parametric**
  - Few assumptions are made about the functional form
  - **Model structure is determined from data**
  - Examples: classification tree, nearest neighbor

Classification Warm Up



K Nearest Neighbors



K Nearest Neighbors

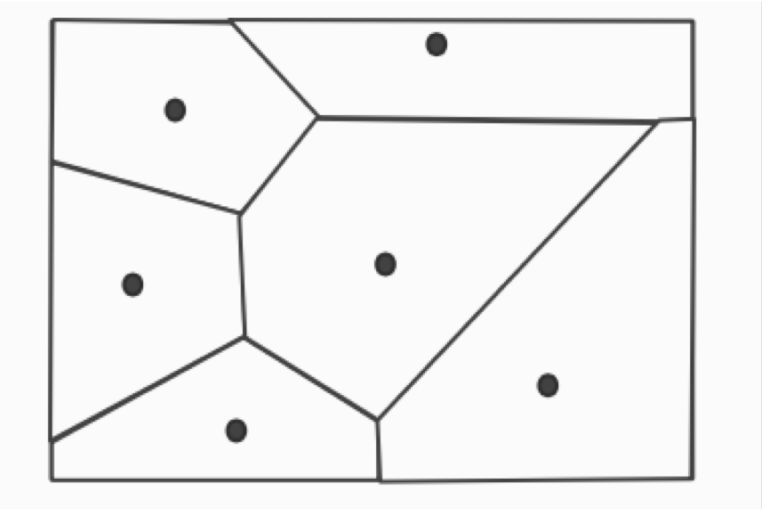
- **Learning**: just storing the training examples
- **Prediction**:
  - Find the K training example closest to **x**
- **Predict a label:**
  - Classification: majority vote
  - Regression: mean value
- KNN is a type of *instance based learning*
- This is called *lazy* learning, since most of the computation is done at prediction time

KNN analysis

- ***What are the advantages and disadvantages of KNN?***
  - *What should we care about when answering this question?*
- ***Complexity***
  - ***Space*** *(how memory efficient is the algorithm?)*
    - *Why should we care?*
  - ***Time*** *(computational complexity)*
    - *Both at training time and at test (prediction) time*
- ***Expressivity***
  - *What kind of functions can we learn?*

KNN analysis

- We discussed the importance of the model space
  - Expressive (we can represent the right model)
  - Constrained (we can search effectively, using available data)
- Let's try to characterize the model space, by looking at the **decision boundary**
- **How would it look if K=1?**

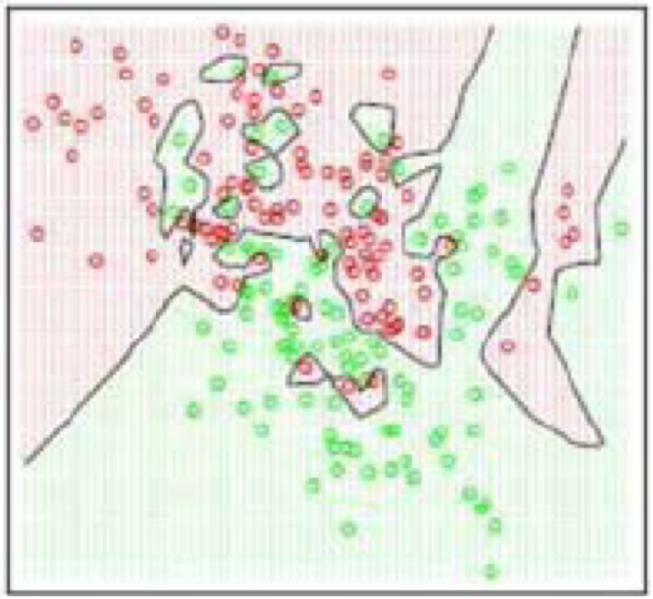


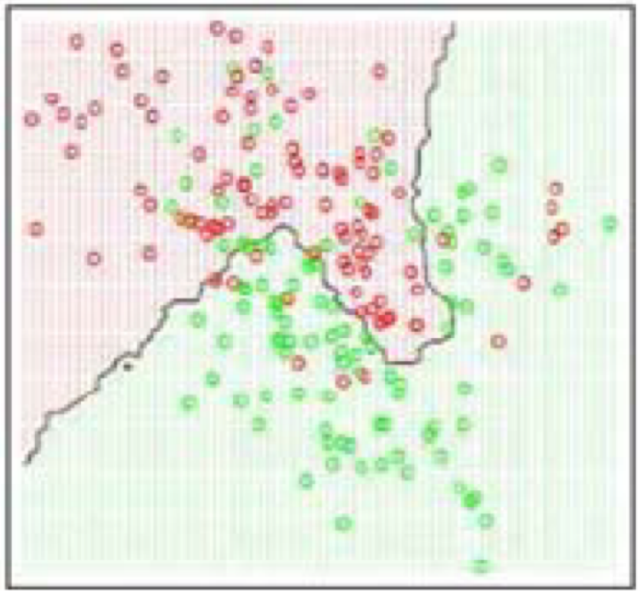
We define the model space to be our choice of K.

Does the complexity of the model space increase of decrease with K?

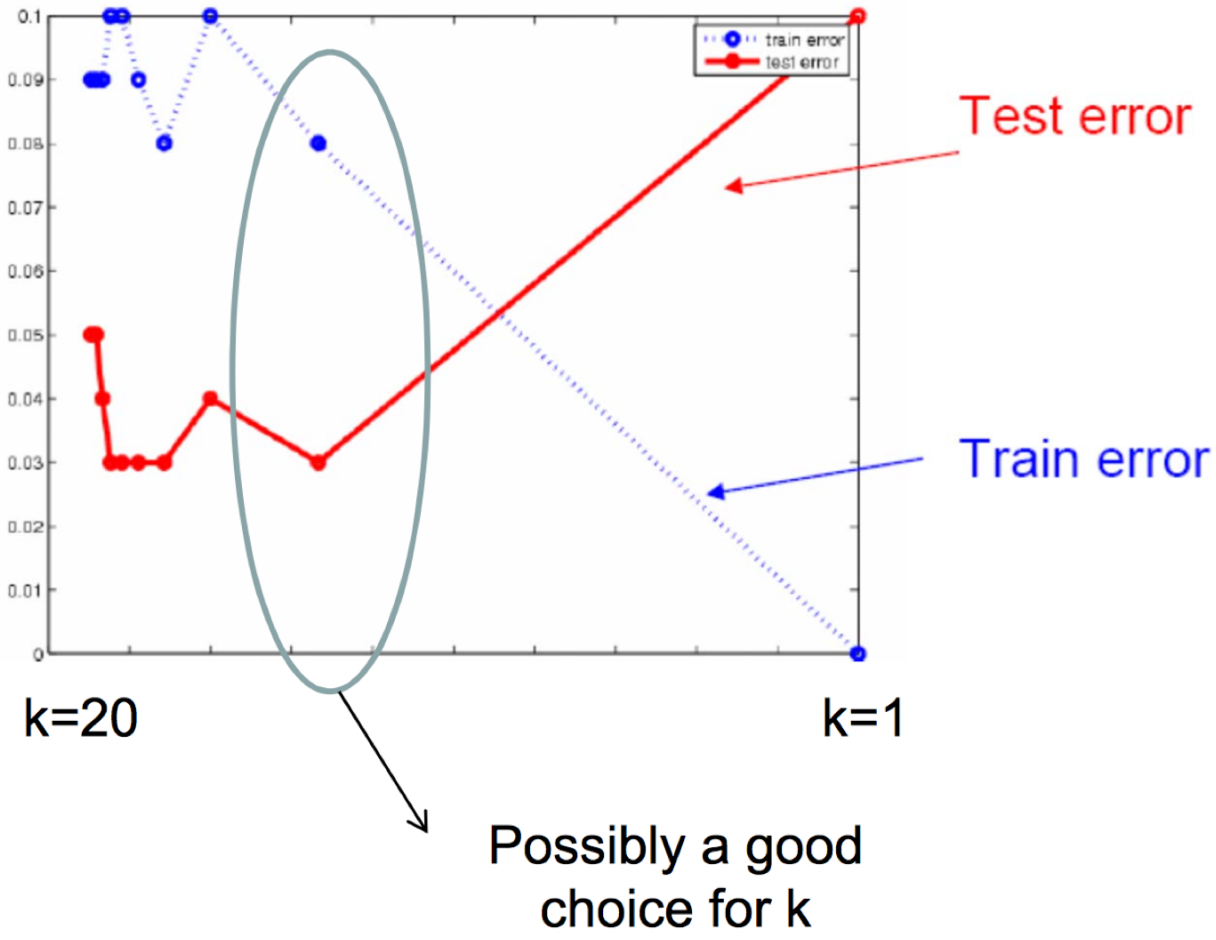
KNN analysis

- Which model has a higher K value?
- Which model is more complex?
- Which model is more sensitive to noise?





Determining the value of K



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

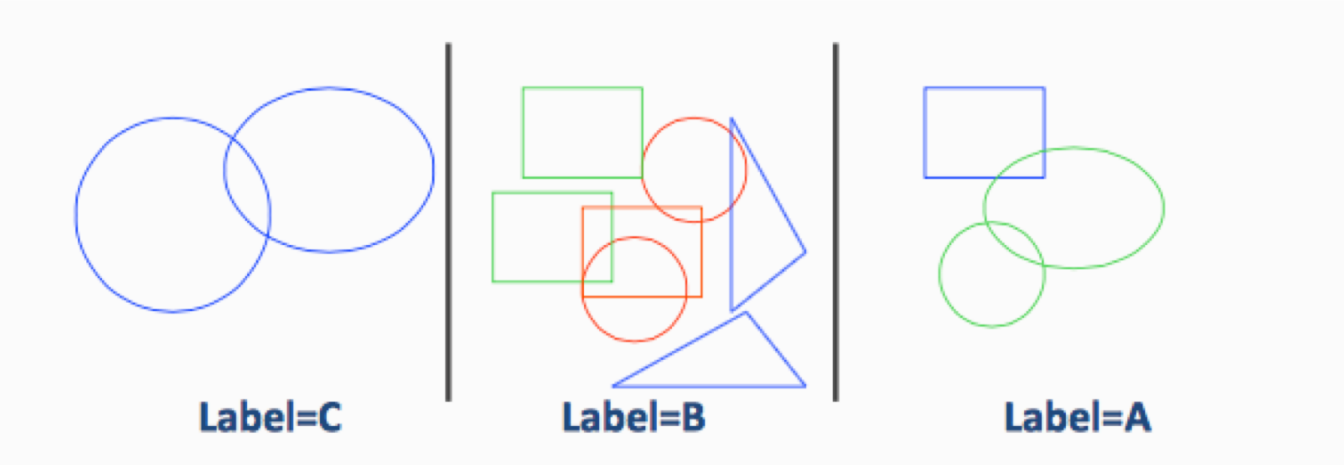
- **The KNN algorithm essentially just memorizes the training set.**
- ***Do we really need to memorize the data?***
- ***Can we represent the decision rule in a better (more concise way?)***

Decision Tree Learning

Decision Trees

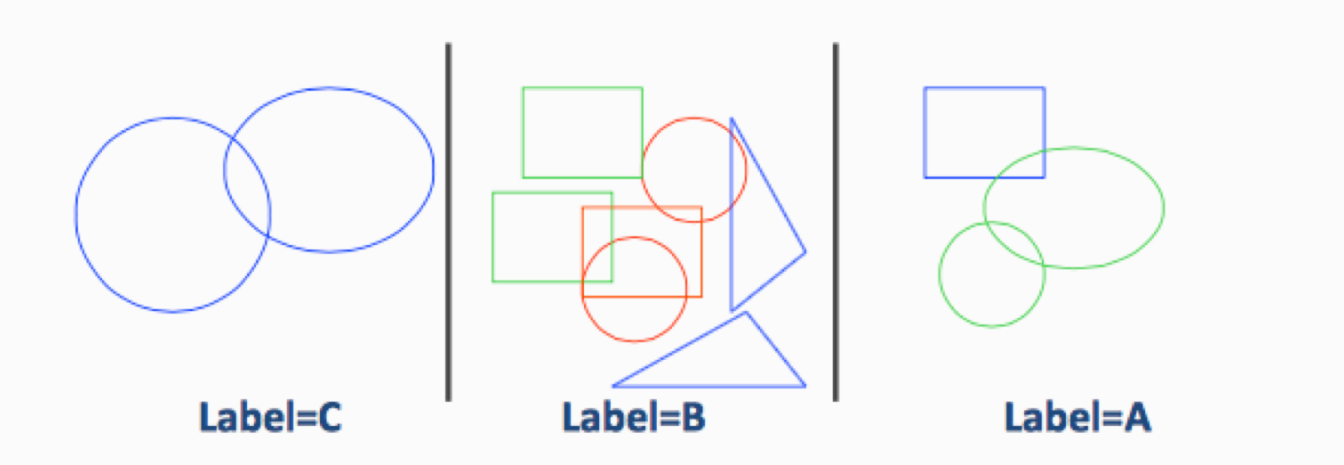
- A **hierarchical data structure** (tree) that represents data by implementing a divide and conquer strategy
- **Nodes** are tests for feature values
  - There is one branch for every value that the feature can take
- **Leaves** of the tree specify the class labels
- Given a collection of examples, **learn a decision tree that represents it.**
- Use this representation to **classify new examples**
  - The tree can be used as a non-parametric classification and regression method

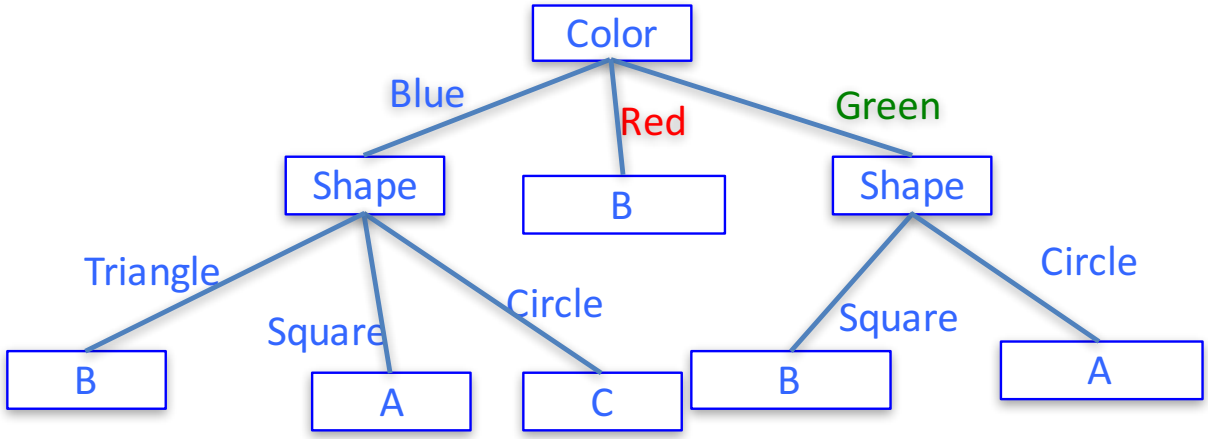
Decision Trees: Representation



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

Decision Trees: Representation



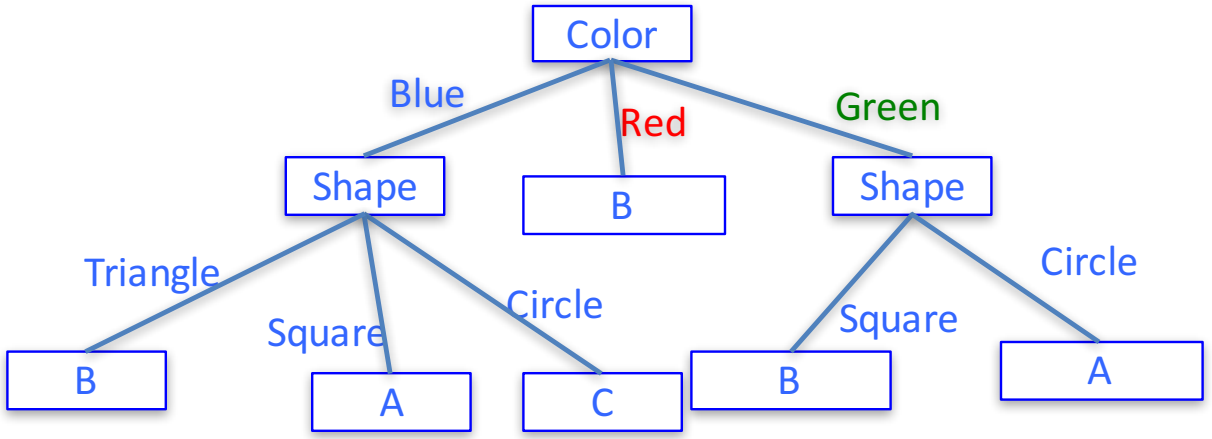


Decision Trees: Representation

- **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**?*

What is the problem? How can we fix it?

  - **How can we learn decision trees from data?**



Decision Trees: Representation

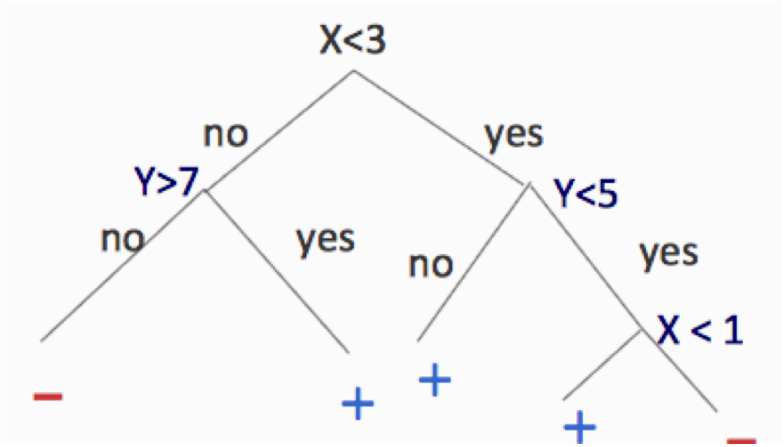
- ***Concise representation of the training data***
  - Shorter trees help generalize to unseen examples
  - Methods for handling noisy data (noise in the label or in the features) and for handling missing attributes
  - **Pruning trees** helps with noise
- ***Output***:
  - Boolean
  - Multiclass (discrete categories)
  - Real valued (regression tree)

Decision Trees: Expressivity

- ***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?*

Decision Trees: Expressivity

- ***How can we deal with 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*

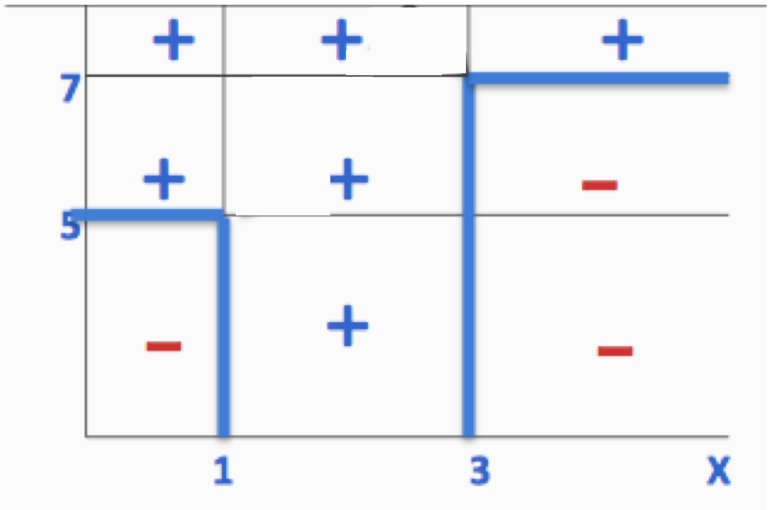


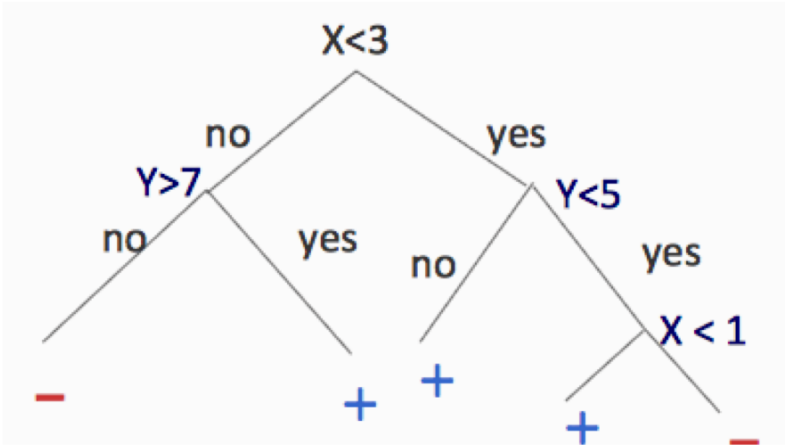
Decision Trees: Expressivity

– *Using discrete features divides the feature space into axis parallel rectangles*

This is the decision boundary for DT

How would you characterize its complexity? Compare to KNN





Summary Model Space

- DT can represent any Boolean function.
  - You can also think about it as a compact representation of data
- **Knowledge representation**: very intuitive!
  - 20 questions
  - Prediction is easy, and we can **understand** the prediction!
- *Given a dataset, is there a 1-1 mapping to a DT?*

Key question: how can we 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)

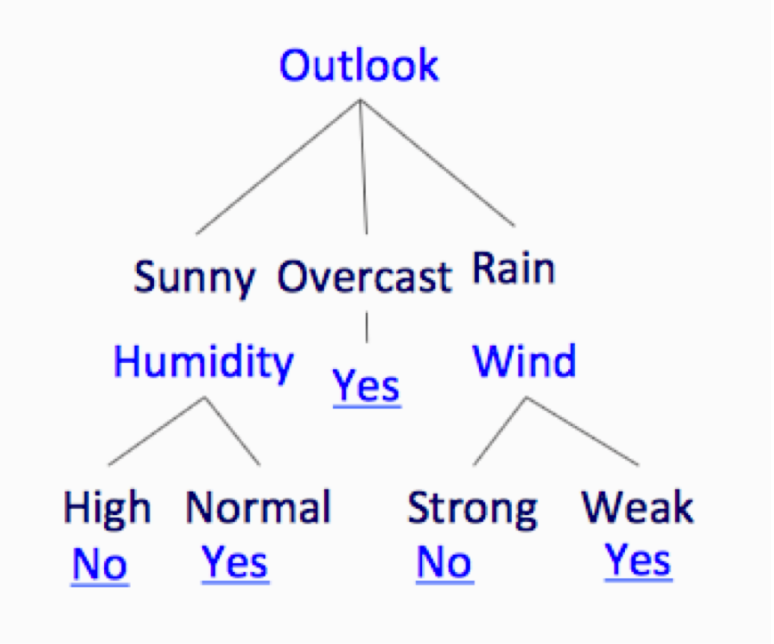
Humidity: H(igh), N(ormal), L(ow)

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.



Tree learning

- **Top-down recursive divide and conquer algorithm**
  - Start with all examples at root
  - Select **best** attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat
- **Other issues:**
  - How to construct features
  - When to stop growing
  - Pruning irrelevant parts of the tree

Basic Decision Tree Learning Algorithm: ID3

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

- 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

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).

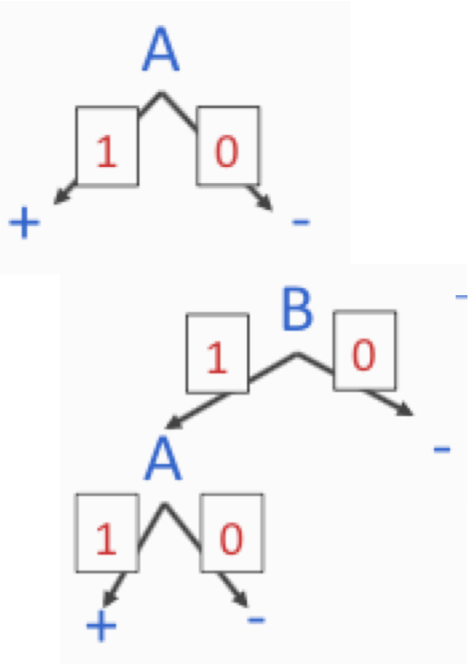
< (A=0,B=0), – >: 50 examples
< (A=0,B=1), – >: 50 examples
< (A=1,B=0), – >: 0 examples
< (A=1,B=1), + >: 100 examples

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**

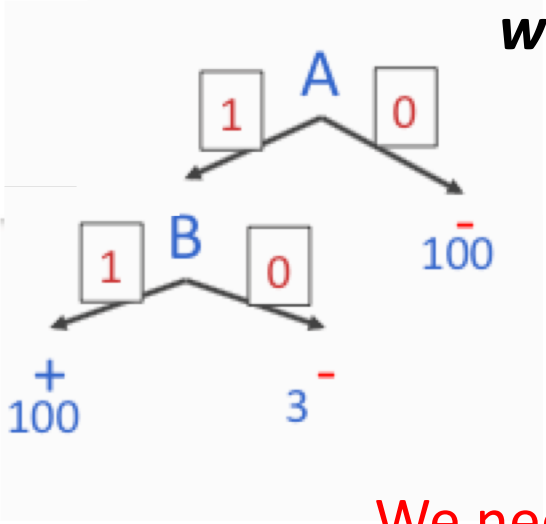


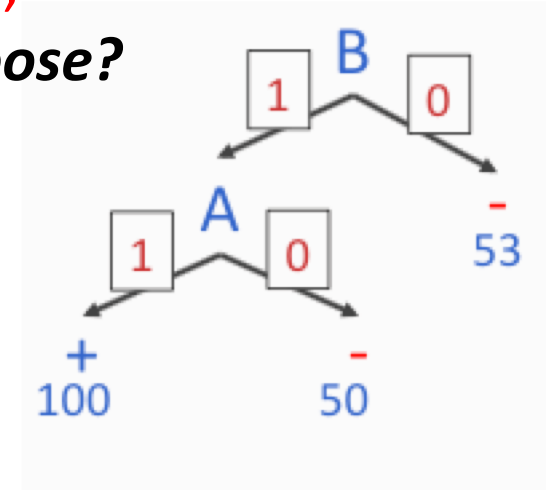
Picking the Root Attribute

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

< (A=0,B=0), – >: 50 examples
< (A=0,B=1), – >: 50 examples
< (A=1,B=0), – >: 0 ~~examples~~ 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

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

- *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

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

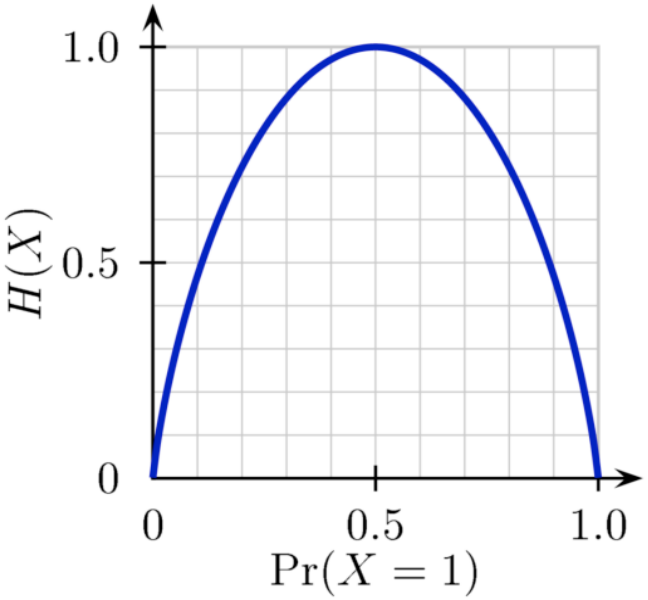
Entropy of a random variable

A **completely random binary variable** with X=[0.5,0.5] has entropy:
H(X) = -(0.5 log 0.5 + 0.5 log 0.5) = -(-.05 + -0.5) = 1

A **deterministic variable** with X=[1,0] has entropy:
H(X) = -(1 log 1 + 0 log 0) = -(0+0) = 0

A **biased variable** with X=[0.75,0.25] has entropy:
H(X) = 0.8113

The entropy of a probability distribution ***p*** expresses the ***amount of uncertainty*** that we have about the values of X



Information gain

- *How much does a feature split decrease the entropy?*

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_A|}{|S|} Entropy(S_A)$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

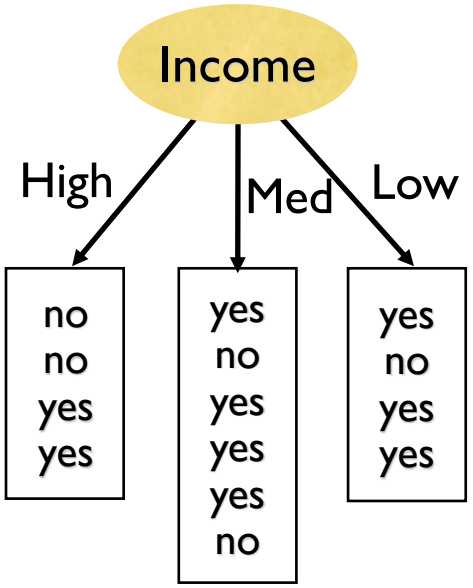
Entropy(D)

= -9/14 log 9/14 -5/14 log 5/14

= 0.9400

Information gain

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_A|}{|S|} Entropy(S_A)$$



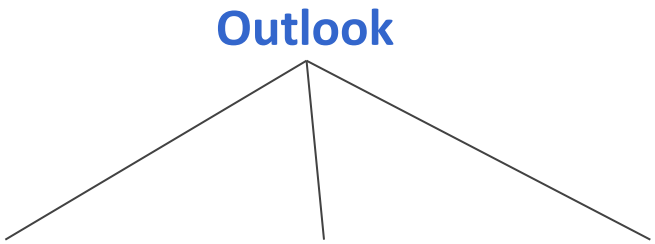
Entropy(Income=high)
= -2/4 log 2/4 -2/4 log 2/4 = 1

Entropy(Income=med)
= -4/6 log 4/6 -2/6 log 2/6 = 0.9183

Entropy(Income=low)
= -3/4 log 3/4 -1/4 log 1/4 = 0.8113

Gain(D,Income)
= 0.9400 - (4/14 [1] + 6/14 [0.9183] + 4/14 [0.8113])
= 0.029

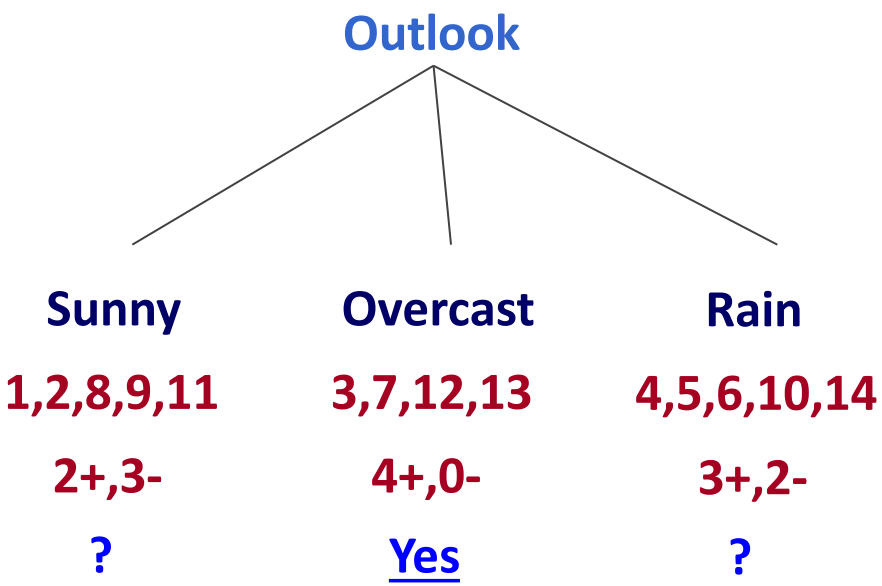
An Illustrative Example



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

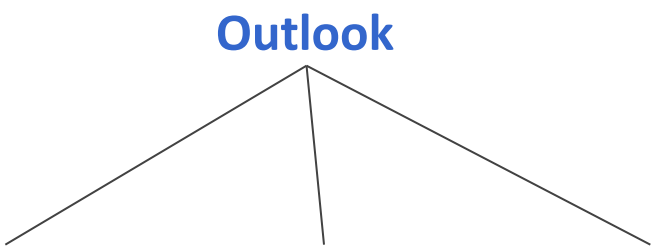
1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

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	--

An Illustrative Example



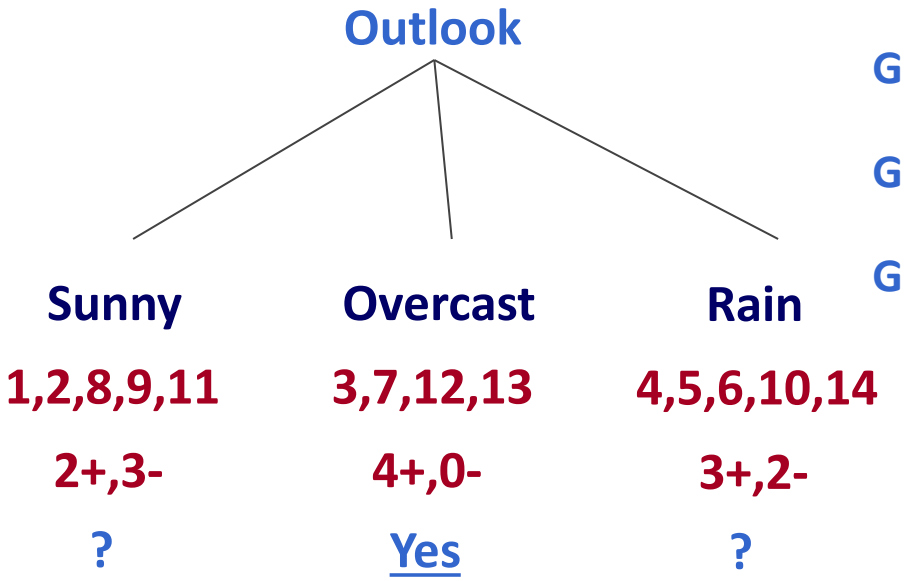
Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

	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



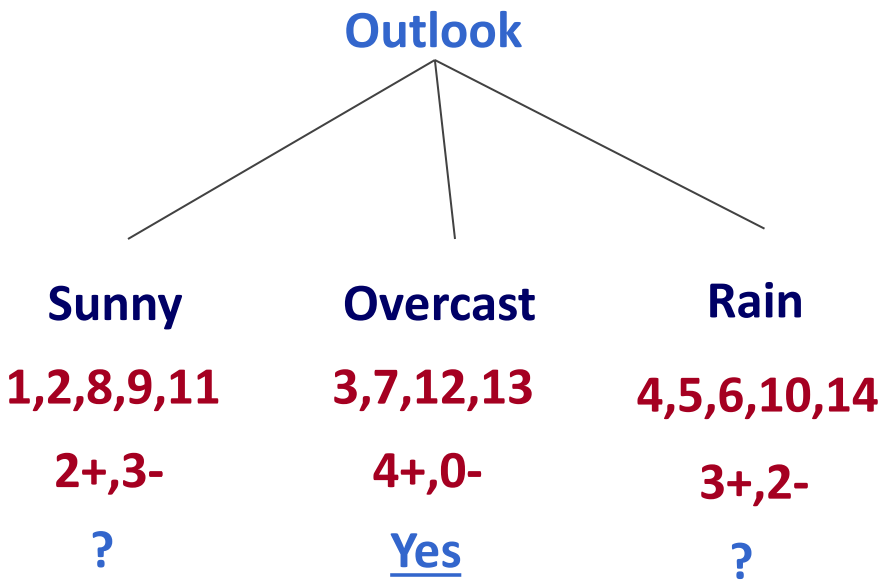
Gain(S_{sunny}, Humidity) .97-(3/5) 0-(2/5) 0= .97

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

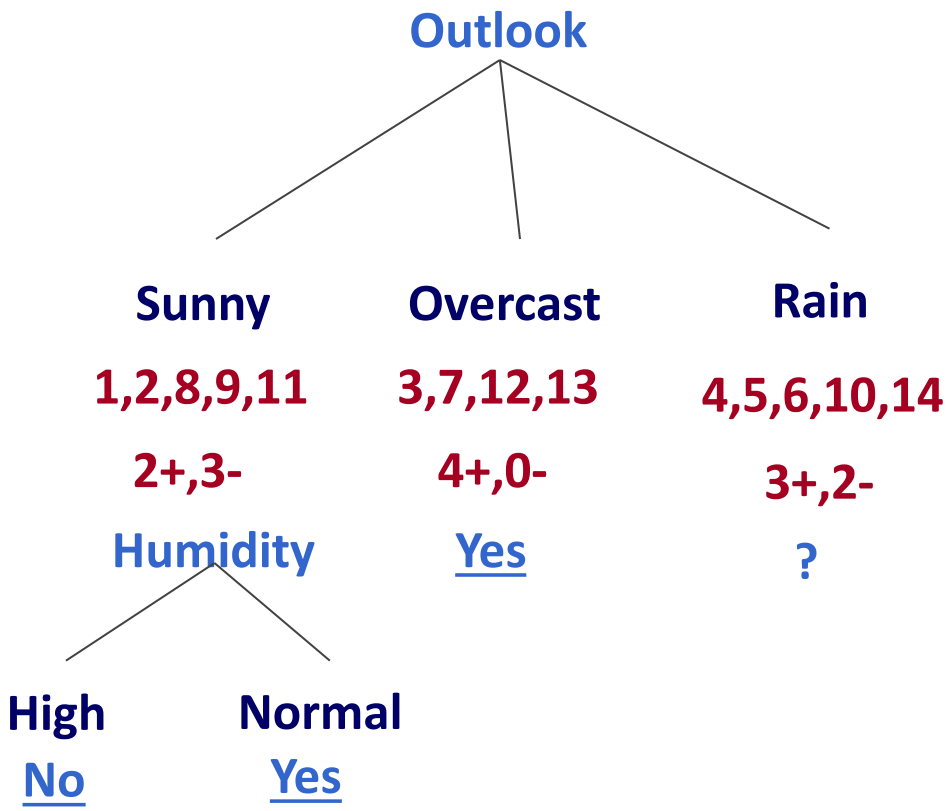
Gain(S_{sunny}, wind) = .97-(2/5) 1 - (3/5) .92 = .02

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

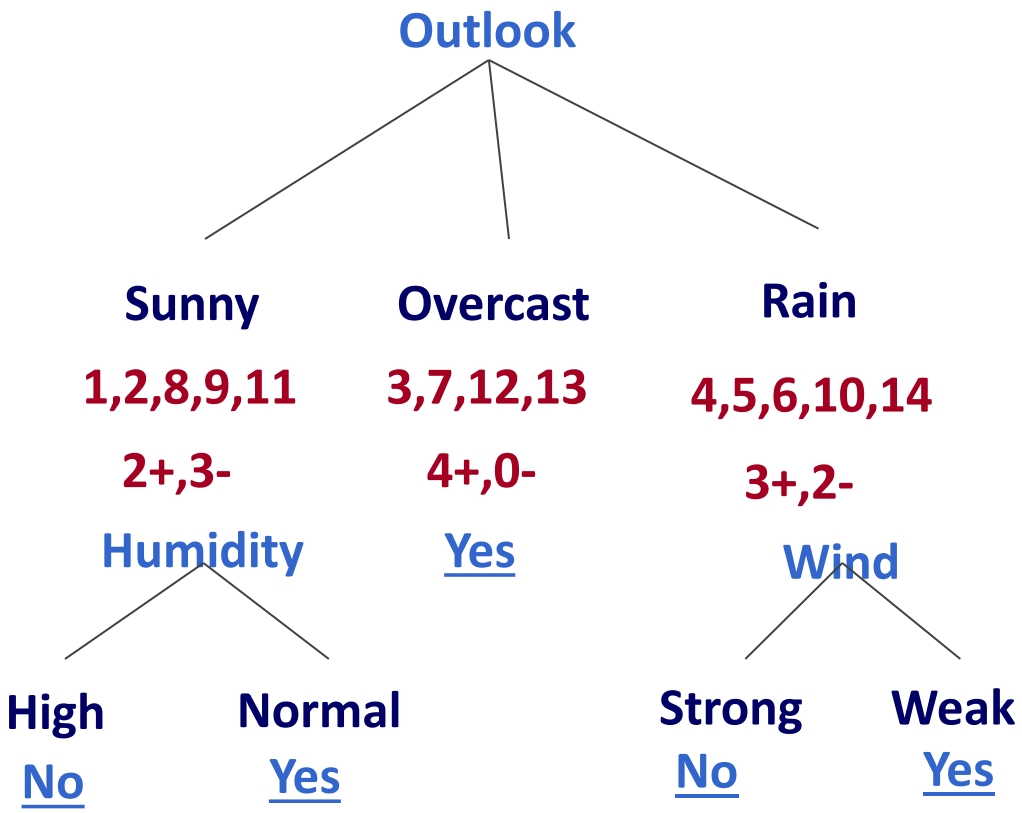
An Illustrative Example



An Illustrative Example



An Illustrative Example



Dealing with Missing Values

- **Suppose an example is missing the value of an attribute**

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

- **"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*

Dealing with Numeric Values

What can you do with numeric features?

→ Easy, 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 $X_j > c$

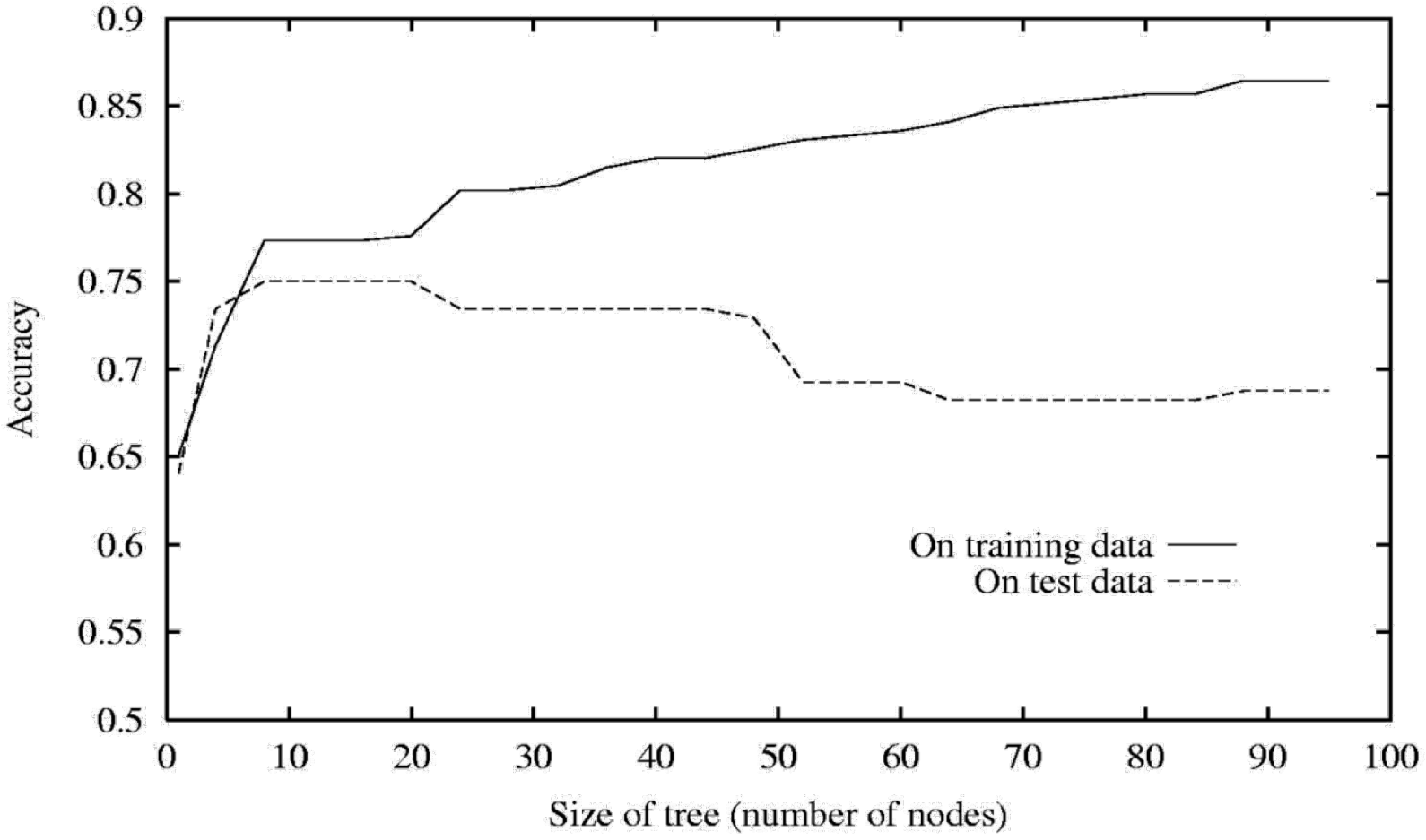
<u>Is there an easier way?</u>

Overfitting in Decision Trees

- Learning a tree classifying the training data perfectly may not have the best generalization
  - Algorithm would fit to noise in the data
  - Sparse dataset
- Decision trees often **overfit**

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.

Overfitting in Decision Trees



Expressing Bias in Decision Tree Learning

- **Inductive Bias**: Expressing preference based on our knowledge (not the data)
  - We prefer shorter trees!
- How can we include this bias when learning DT?
  - **Search bias (dynamic)**
    - We bias the search (hill climbing) to prefer shorter trees
  - **Language bias (static)**
    - Restrict the length of the trees

Expressing Bias in Decision Tree Learning

- **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

- **Post Pruning:** Prune an *existing* tree
- *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*

Expressing Bias in DT Learning

- **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

- **Post Pruning:** Prune an *existing* tree
- 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

Model Selection

- **What are the algorithm hyper-parameters?**
  - ***Decision trees:***
    - Depth of the tree, Pruning strategy, Pruning decision
    - Attribute selection heuristic,
    - *Other choices?*
  - ***KNN***
    - *Value of K, Similarity metric*
  - ***Every learning algorithm has a set of hyper-parameters***

Model Selection

- *Think about selecting the best model as a **secondary** learning problem*
  - Split the data into: (1) train set (2) test set
  - Split the train data into: (1) train set (2) validation set
  - **Training**: train m models, with different parameters
    - E.g., Different ways to control the size of the tree
  - **Validation:** estimate the prediction error for each model
  - **Test**: use the model with the least validation error
- ***The secondary learning problem***:
  - New hypothesis space: m different hypothesis to chose from
  - Pick the one that minimizes validation error

K-Fold Cross validation

- *The previous approach was "risky"* ***(why?)***
  - Random selection of training examples for train/test/validation
    - You could be very unlucky
  - *You validation data may not reflect the same distribution as your test data*
    - Optimizing on the validation data will lead to worse performance

K-Fold Cross validation

- You could get really unlucky
  - **But that's not likely to happen too frequently!**
- **K-Fold cross validation**:
- repeat the process K times, and average the results.
- Randomly partition the data into K equal-size subsets $S_1..S_k$
  - For i=1… K
    - Train a hypothesis on $S_1..S_{i-1}\ S_{i+1}..S_k$
    - Evaluate on S_i ($Err(S_i)$)
  - Return ($\Sigma_i\ Err(S_i)/K$)

Precision and Recall

- Given a dataset, we train a classifier that gets 99% accuracy
- **Did we do a good job?**
- Build a classifier for brain tumor:
  - 99.9% of brain scans do not show signs of tumor
  - ***Did we do a good job?***
- By simply saying "NO" to all examples we reduce the error by a factor of 10!
  - ***Clearly Accuracy is not the best way to evaluate the learning system when the data is heavily skewed!***
- **Intuition**: we need a measure that captures the class we care about! (rare)

Precision and Recall

- The learner can make two kinds of mistakes:
  - False Positive
  - False Negative

	1 True Label	0 True Label
1 Predicted	True Positive	False Positive
0 Predicted	False Negative	True Negative

- **Precision**: $\frac{\text{True Pos}}{\text{Predicted Pos}} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Pos}}$
- *"when we predicted the rare class, how often are we right?"*
- **Recall** $\frac{\text{True Pos}}{\text{Actual Pos}} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Neg}}$
- "Out of all the instances of the rare class, how many did we catch?"

Precision and Recall

- **Precision and Recall give us two reference points to compare learning performance**

	Precision	Recall
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1

- *Which algorithm is better?*

We need a single score

- Option 1: **Average** $\frac{P+R}{2}$
- Option 2: **F-Score** $2\frac{PR}{P+R}$

Properties of f-score:

- Ranges between 0-1
- Prefers precision and recall with similar values

Decision Tree Summary

- **Very popular method**
  - 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