Decision Tree Learning

Lecture Slides 2

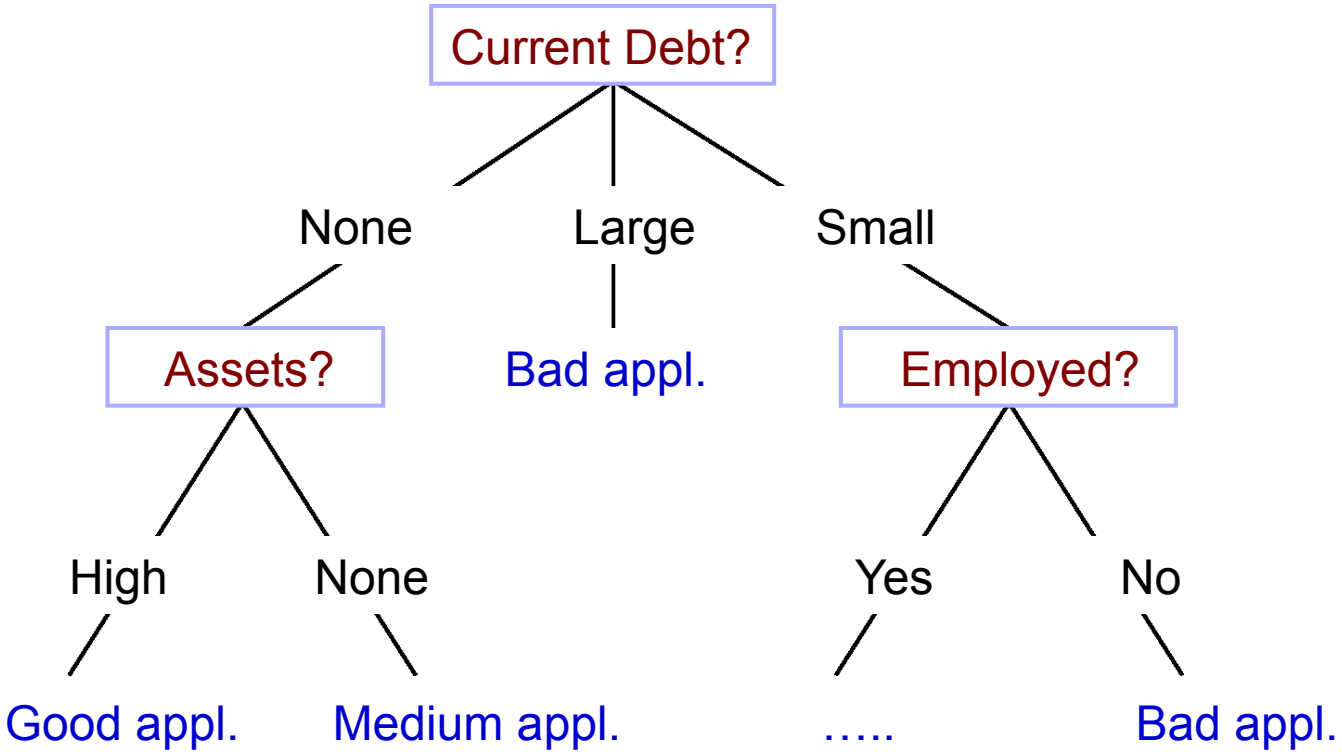
- In these slides;
  - you will strengthen your understanding of the introductory concepts of machine learning
  - learn about the Decision tree approach which is one of the fundamental approaches in machine learning
    - Decision trees are also the basis of a new method called “random forest” which we will see towards the end of the course.

- Note that you can skip Advanced marked slides, they are extra information

Decision Trees

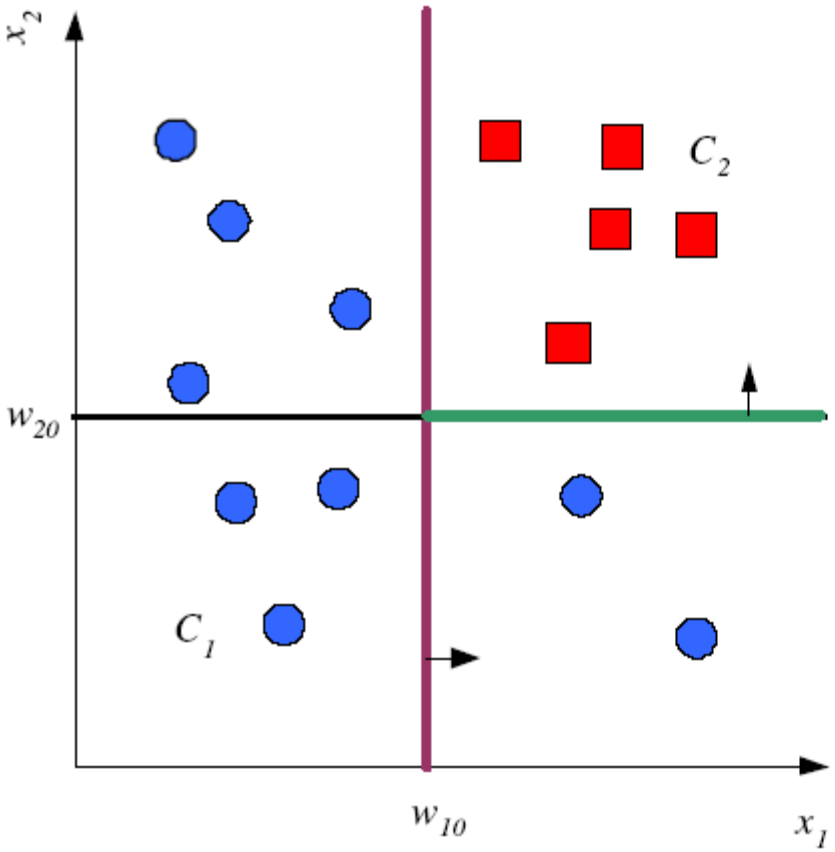
- One of the most widely used and practical methods for inductive inference
- Can be used for classification (most common use) or regression problems

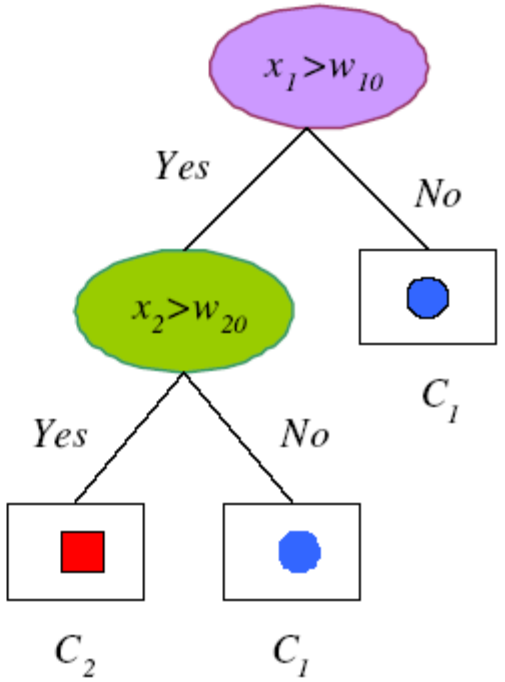
Decision Tree for *"Good credit applicant?"*



- Each internal node corresponds to a test
- Each branch corresponds to a result of the test
- Each leaf node assigns a classification♪

Decision Regions





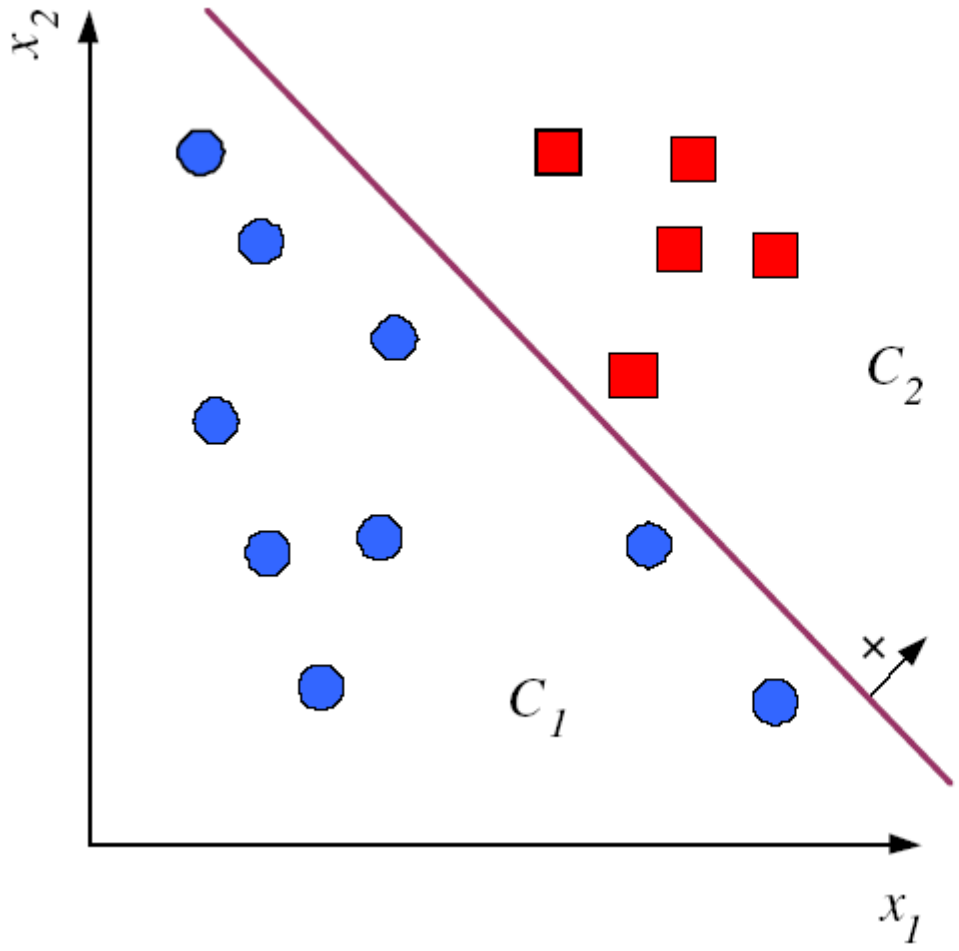
Divide and Conquer

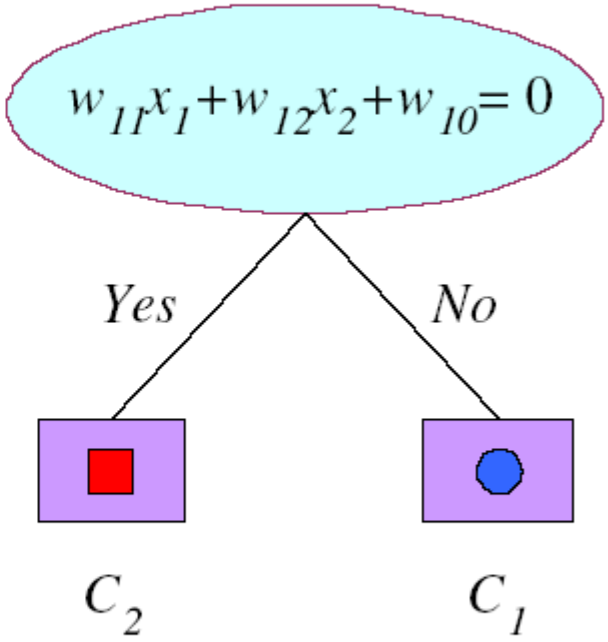
- Internal decision nodes
  - Univariate: Uses a single attribute, x_i
    - Discrete x_i : n-way split for n possible values
    - Continuous x_i : Binary split : $x_i > w_m$
  - Multivariate: Uses more than one attributes

- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; r average, or local fit

- Once the tree is trained, a new instance is classified by starting at the root and following the path as dictated by the test results for this instance.

Multivariate Trees

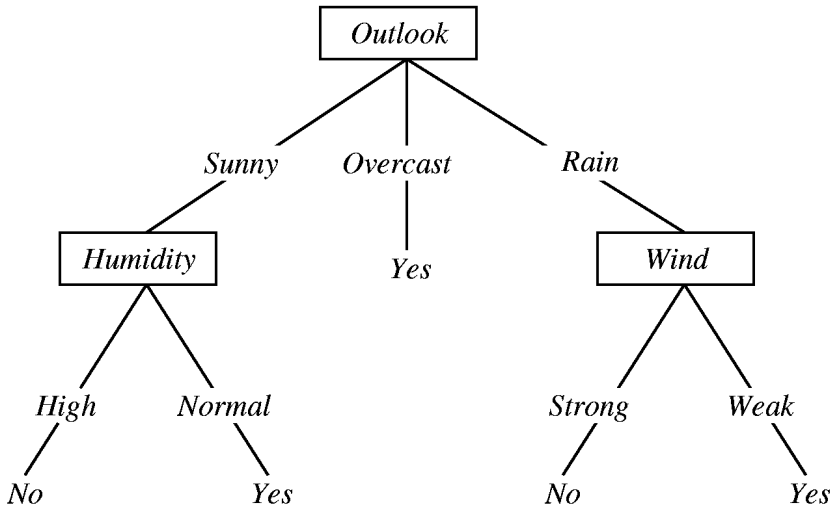




Expressiveness

- A decision tree can represent a disjunction of conjunctions of constraints on the attribute values of instances.
  - Each path corresponds to a conjunction
  - The tree itself corresponds to a disjunction

Decision Tree



If (O=Sunny AND H=Normal) OR (O=Overcast) OR (O=Rain AND W=Weak) then YES

- "A disjunction of conjunctions of constraints on attribute values"

Advanced

- How expressive is this representation?

- How would we represent:
  - (A AND B) OR C
  - A XOR B
  - M–of-N
    - 2-of-(A,B,C,D))
    - AB + AC + AD + BC + BD + CD

Decision tree learning algorithm

- For a given training set, there are many trees that code it without any error

- Finding the smallest tree is NP-complete (Quinlan 1986), hence we are forced to use some (local) search algorithm to find reasonable solutions

- Learning is greedy; find the best split **recursively** (Breiman et al, 1984; Quinlan, 1986, 1993)

- If the decisions are binary, then in the best case, each decision eliminates half of the regions (leaves).

- If there are b regions, the correct region can be found in $log_2 b$ decisions, in the best case.

The basic decision tree learning algorithm

- A decision tree can be constructed by considering attributes of instances one by one.
  - Which attribute should be considered first?

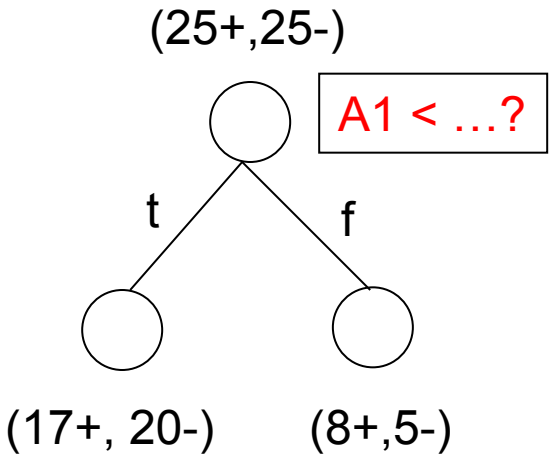
- The height of a decision tree depends on the order attributes that are considered.

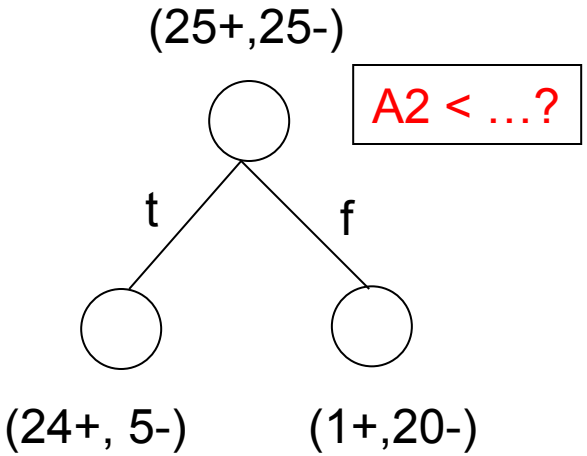
Top-Down Induction of Decision Trees

Main loop:

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

- Which attribute is best?





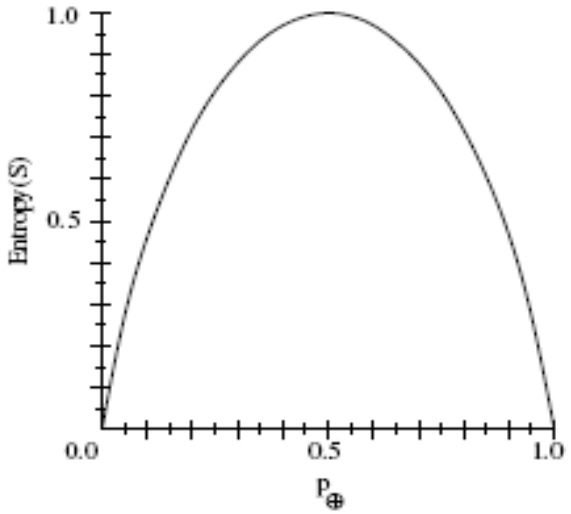
Entropy

- **Measure of uncertainty**
- **Expected number of bits to resolve uncertainty**
- **Entropy measures the information amount in a message**

- Important quantity in
  - coding theory
  - statistical physics
  - machine learning
  - …

- Show the high school form example with gender field

Entropy of a Binary Random Variable



- S is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in S
- $p_\ominus$ is the proportion of negative examples in S

Entropy of a Binary Random Variable

- **Entropy measures the impurity of *S:***

$$\textit{Entropy}(S) = -p \log_2 p + -(1-p) \log_2 (1-p)$$

Note: Here p=p-positive and 1-p= p_negative in the previous slide

- Example: Consider a binary random variable X s.t. Pr{X = 0} = 0.1

$$\text{Entropy(X)} = 0.1 \times \lg \frac{1}{0.1} + (1-0.1) \times \lg \frac{1}{(1-0.1)}$$

Entropy – General Case

- When the random variable has **multiple** possible outcomes, its entropy becomes:

$$\mathrm{H}[x] = -\sum_{x} p(x) \log_2 p(x)$$

- Here x is the possible outcomes (Male/Female or Dice outcomes,...)

Entropy

Example from Coding theory:

Random variable x discrete with 8 possible states; how many bits are needed to transmit the state of x?

1. All states equally likely

$$\mathrm{H}[x] = -8 \times \frac{1}{8} \log_2 \frac{1}{8} = 3 \text{ bits.}$$

2. We have the following distribution for x?

x	a	b	c	d	e	f	g	h
$p(x)$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$

$$\begin{aligned} \mathrm{H}[x] &= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{16}\log_2\frac{1}{16} - \frac{4}{64}\log_2\frac{1}{64} \\ &= 2 \text{ bits} \end{aligned}$$

Advanced

- Indeed, this process is used in designing codes for messages, such that the total transmission costs are minimized.

x	a	b	c	d	e	f	g	h
$p(x)$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$
code	0	10	110	1110	111100	111101	111110	111111

$$\begin{aligned} \text{average code length} &= \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + 4 \times \frac{1}{64} \times 6 \\ &= 2 \text{ bits} \end{aligned}$$

Use of Entropy in Choosing the Next Attribute

- We will use the entropy of the remaining tree as our measure to prefer one attribute over another.

- In summary, we will consider
  - the entropy over the distribution of samples falling under each leaf node and
  - we will take a weighted average of that entropy – weighted by the proportion of samples falling under that leaf.

- We will then choose the attribute that brings us the biggest information gain, or equivalently, results in a tree with the lower weighted entropy.

Information Gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

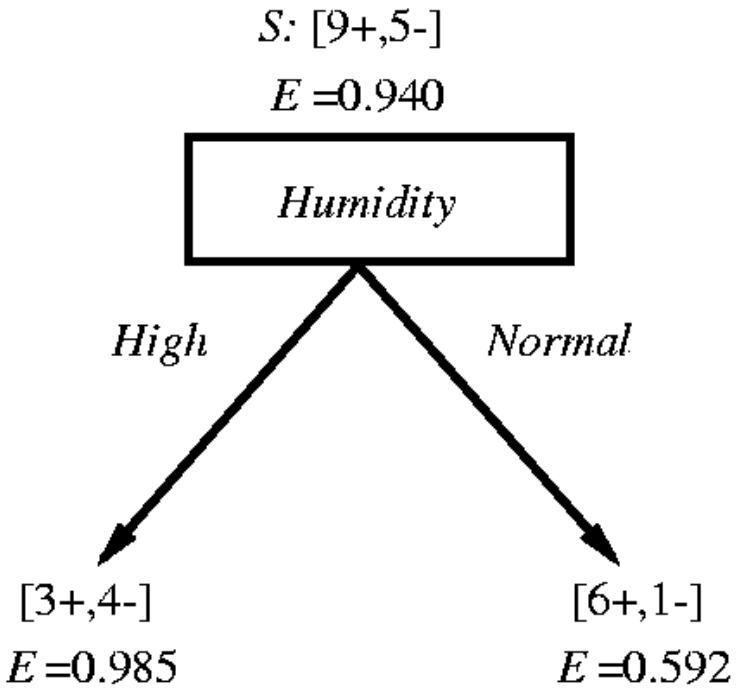
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

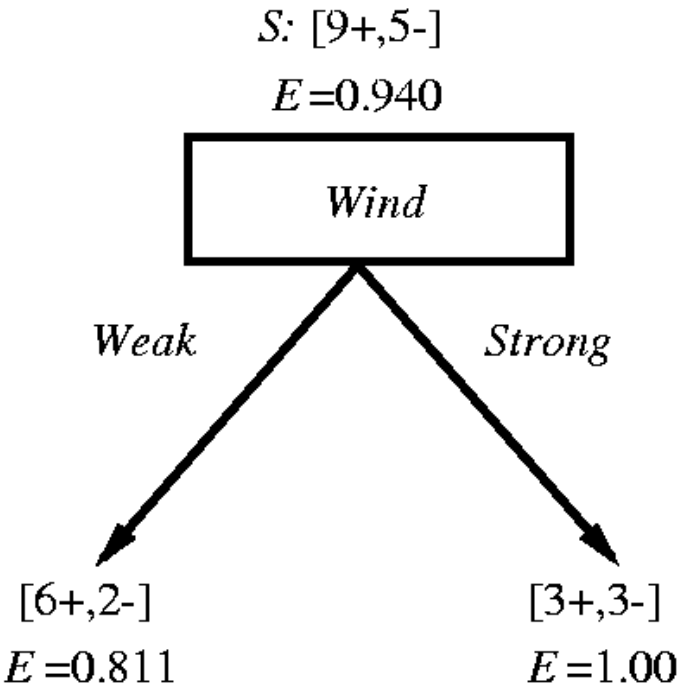
Please note: Gain is what you started with MINUS Remaining entropy.
We can also simply choose the tree with the **smallest remaining entropy**!

Training Examples

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

Which attribute is the best classifier?





Gain (S, Humidity)

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

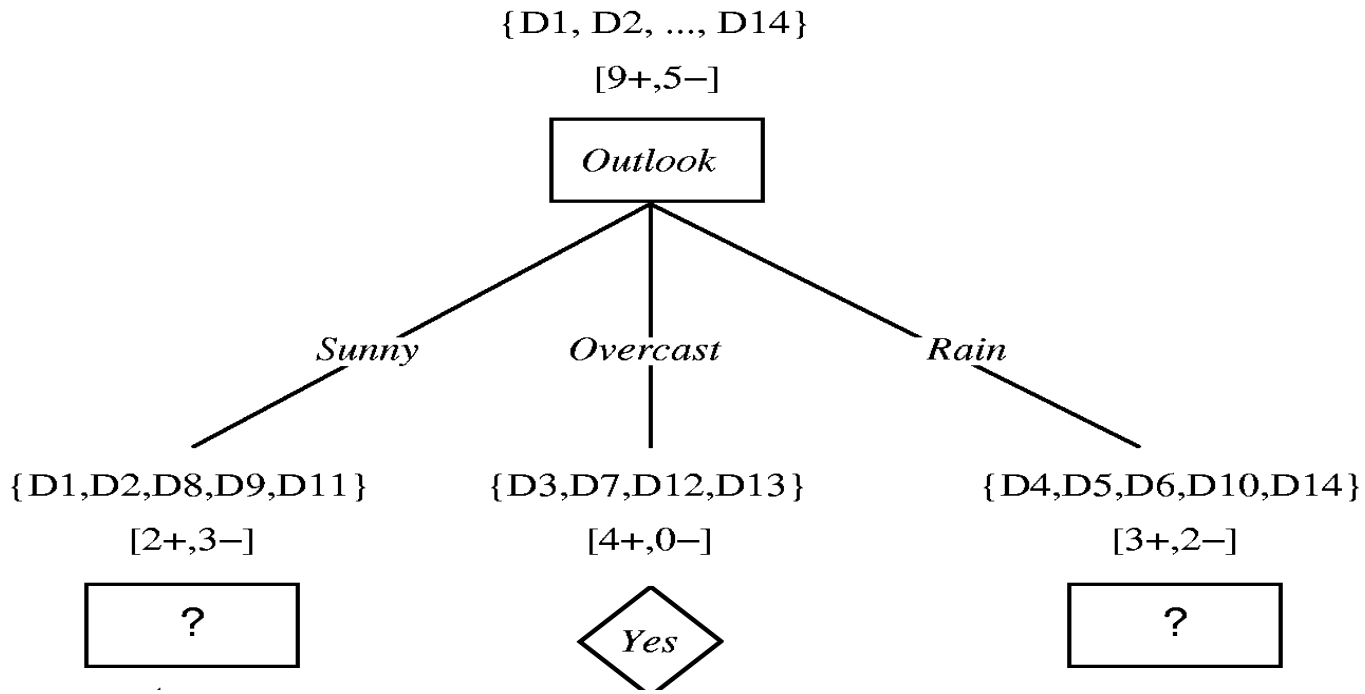
Gain (S, Wind)

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

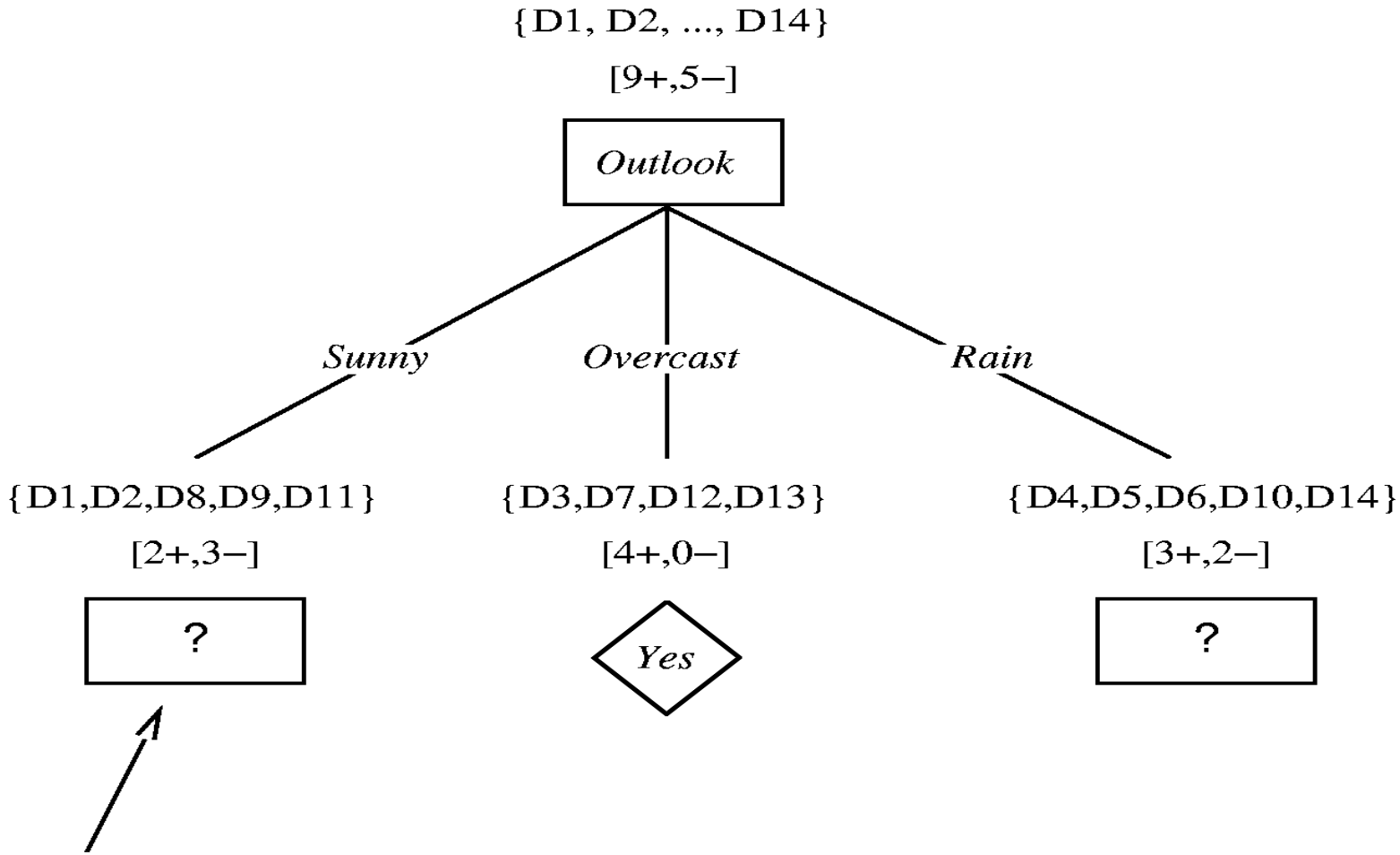
We would select the Humidity attribute to split the root node as it has a higher Information Gain.

Selecting the Next Attribute

- Computing the information gain for each attribute, we selected the *Outlook* attribute as the first test, resulting in the following partially learned tree:



- We can repeat the same process recursively, until Stopping conditions are satisfied.



Which attribute should be tested here?

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

Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Until stopped:

- Select one of the unused attributes to partition the remaining examples at each non-terminal node using only the training samples associated with that node

Stopping criteria:

- each leaf-node contains examples of one type
- algorithm ran out of attributes
- …

Advanced: Other measures of impurity

- Entropy is not the **only** measure of impurity. If a function satisfies certain criteria, it can be used as a measure of impurity.
- For Binary random variables (only p value is indicated), we have:
  - **Gini index: 2p(1-p)**
    - p=0.5 Gini Index=0.5
    - p=0.9 Gini Index=0.18
    - p=1 Gini Index=0
    - p=0 Gini Index=0
  - **Misclassification error: 1 – max(p,1-p)**
    - p=0.5 Misclassification error=0.5
    - p=0.9 Misclassification error=0.1
    - P=1 Misclassification error=0
    - P=0 Misclassification error=0

Other Issues With Decision Trees

Continuous Values

Missing Attributes

…

Continuous Valued Attributes

- Create a discrete attribute to test continuous variables

  Temperature = 82:5

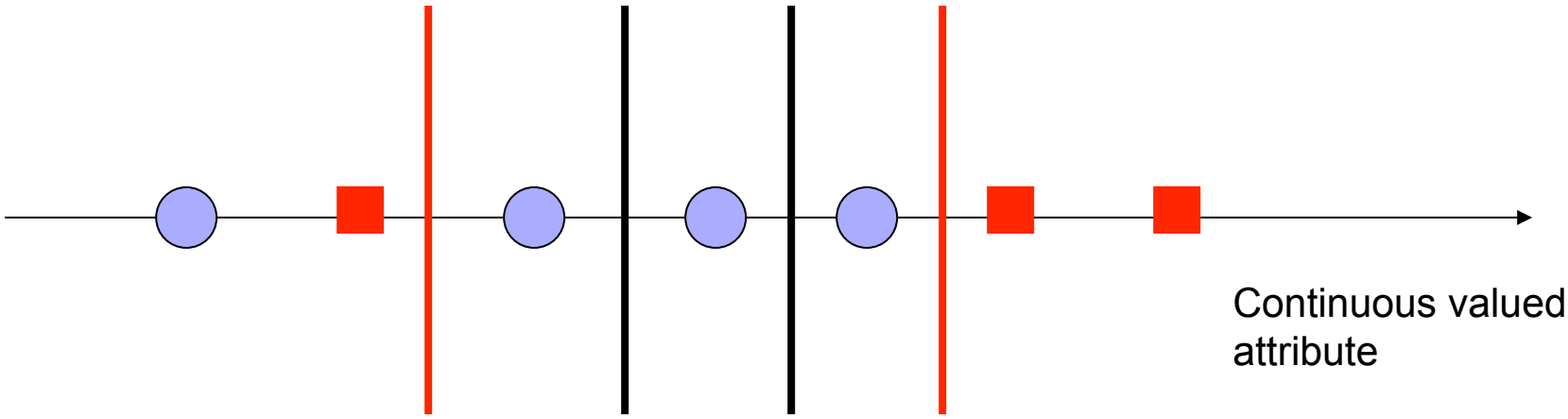
  (*Temperature* > 72:3) = *t; f*

- *How to find the threshold?*

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

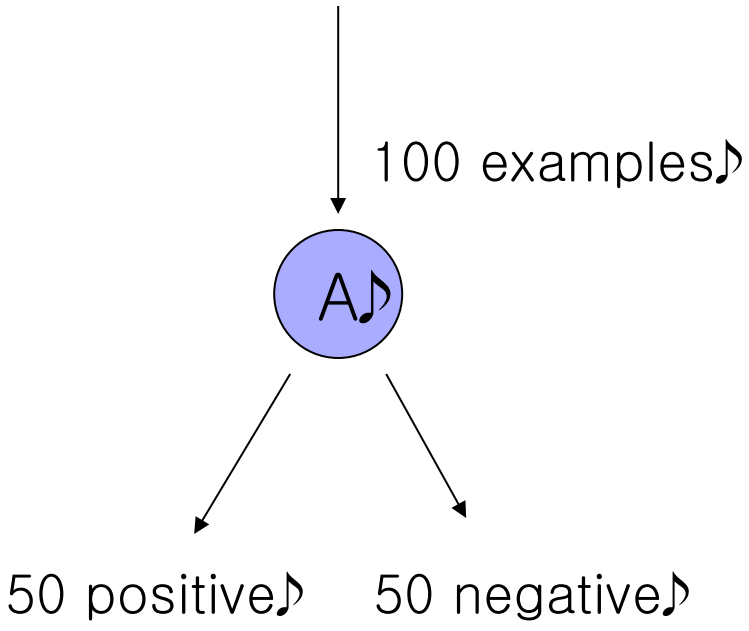
Incorporating continuous-valued attributes

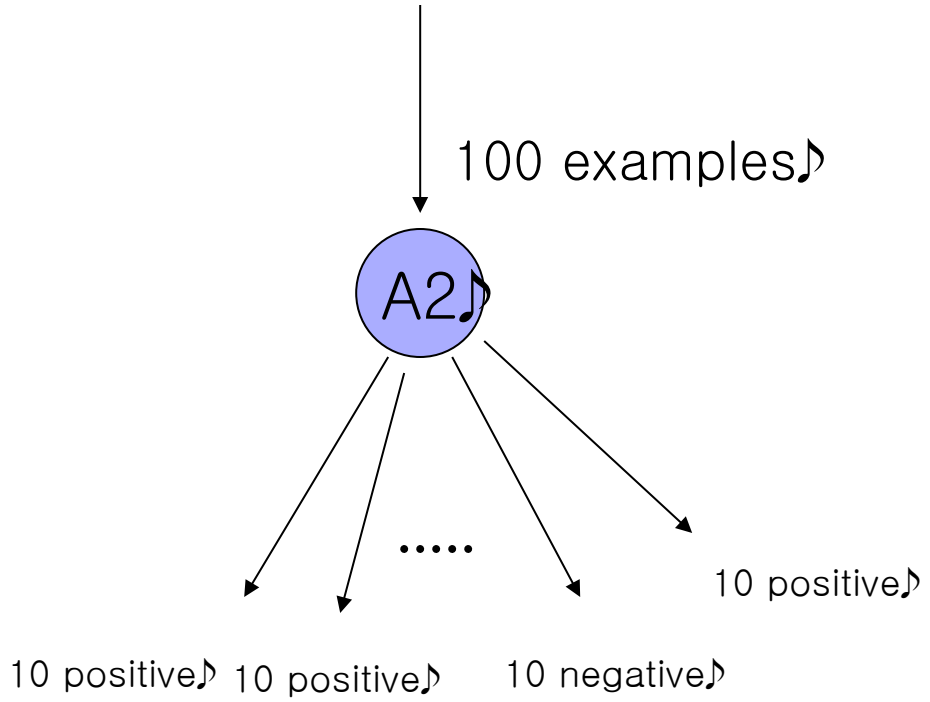
- Where to cut?
  - We can show that the threshold is always the the transitions (shown in red boundaries between the two classes)



Advanced: Split Information?

- In each tree, the leaves contain samples of only one kind (e.g. 50+, 10+, 10- etc).
  - Hence, the remaining entropy is 0 in each one.
- But we would often prefer A1.
  - How to indicate this preference?





Advanced: Attributes with Many Values –

- One way to penalize such attributes is to use the following alternative measure:

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitInformation(S,A)}$$

$$SplitInformation(S,A) = -\sum_{i=1} \frac{|S_i|}{|S|} \lg \frac{|S_i|}{|S|}$$

Entropy of the **attribute** A:♪
Experimentally determined by the training samples♪

Split Information is 3.3 versus 1, for the previous example.
-10*0.1*log_2(0.1) vs.
- 2*0.5*log_2(0.5)

Handling training examples with missing attribute values

- What if an example x is missing the value an attribute A?

- Simple solution:
  - Use the most common value among examples at node n.
  - Or use the most common value among examples at node n that have classification c(x)

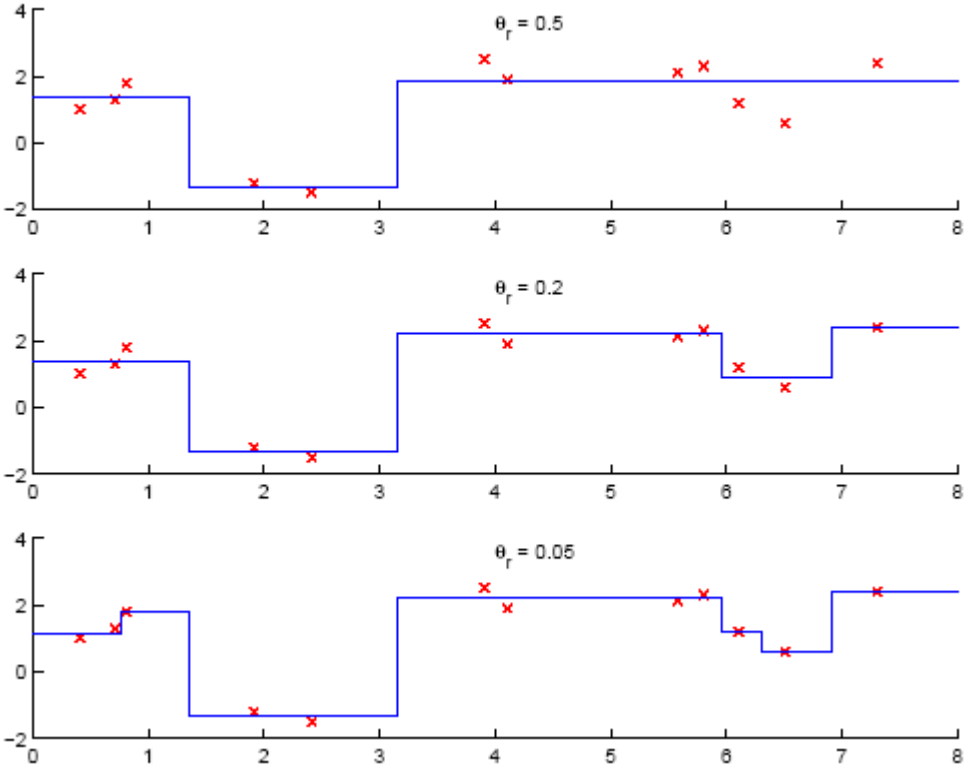
- More complex, probabilistic approach
  - Assign a probability to each of the possible values of A based on the observed frequencies of the various values of A
  - Then, propagate examples down the tree with these probabilities.
  - The same probabilities can be used in classification of new instances (used in C4.5)

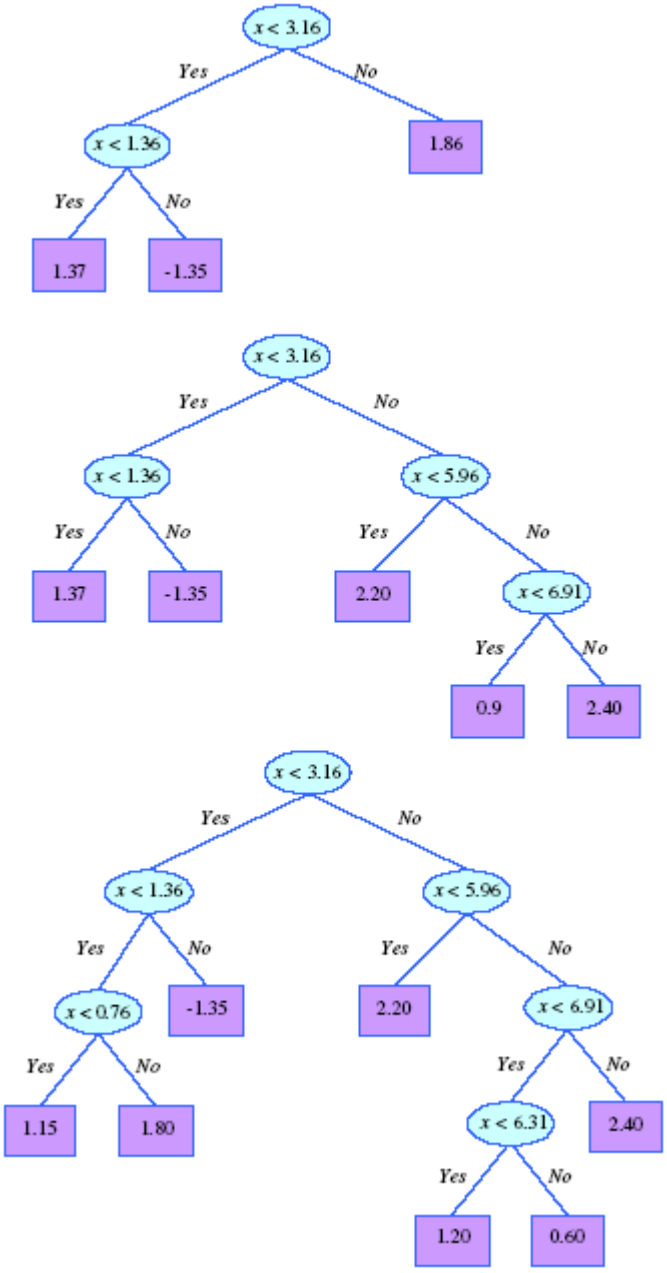
Handling attributes with differing costs

- Sometimes, some attribute values are more expensive or difficult to prepare.
  - medical diagnosis, BloodTest has cost $150
- In practice, it may be desired to postpone acquisition of such attribute values until they become necessary.
- To this purpose, one may modify the attribute selection measure to penalize expensive attributes.
  - Tan and Schlimmer (1990) $\frac{Gain^2(S,A)}{Cost(A)}$
  - Nunez (1988) $\frac{2^{Gain(S,A)} - 1}{(Cost(A)+1)^w}, \; w \in [0,1]$

Regression with Decision Trees

Model Selection in Trees:





Strengths and Advantages of Decision Trees

- Rule extraction from trees
  - A decision tree can be used for feature extraction (e.g. seeing which

  features are useful)

- Interpretability: human experts may verify and/or discover patterns

- Compact and fast classification method

Overfitting

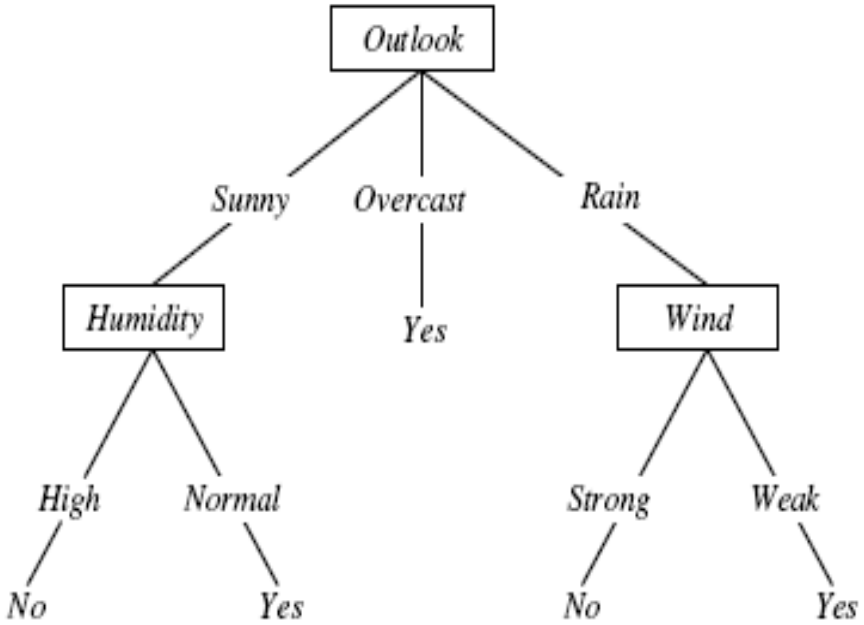
Over fitting in Decision Trees

- Why “over”-fitting?
  - A model can become more complex than the true target function when it tries to satisfy noisy data as well.

- Consider adding the following training example which is **incorrectly labeled as negative:**

Sky; Temp; Humidity; Wind; PlayTennis

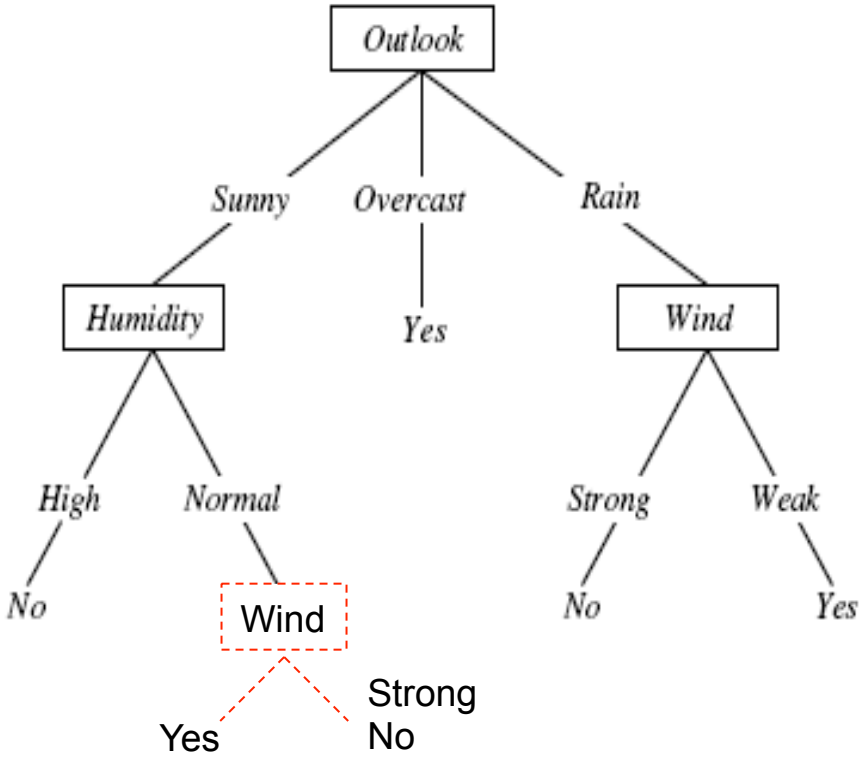
Sunny; Hot; Normal; Strong; PlayTennis = No



- Consider adding the following training example which is **incorrectly labeled as negative:**

Sky; Temp; Humidity; Wind; PlayTennis

Sunny; Hot; Normal; Strong; PlayTennis = No



- ID3 (the Greedy algorithm that was outlined) will make a new split and will classify future examples following the new path as negative.

- Problem is due to ”overfitting” the training data which may be thought as insufficient generalization of the training data
  - Coincidental regularities in the data
  - Insufficient data
  - Differences between training and test distributions

- Definition of overfitting
  - A hypothesis is said to overfit the training data if there exists some other hypothesis that has **larger** error over the training data but **smaller** error over the entire instances.

Overfitting

What is the formal description of overfitting?

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

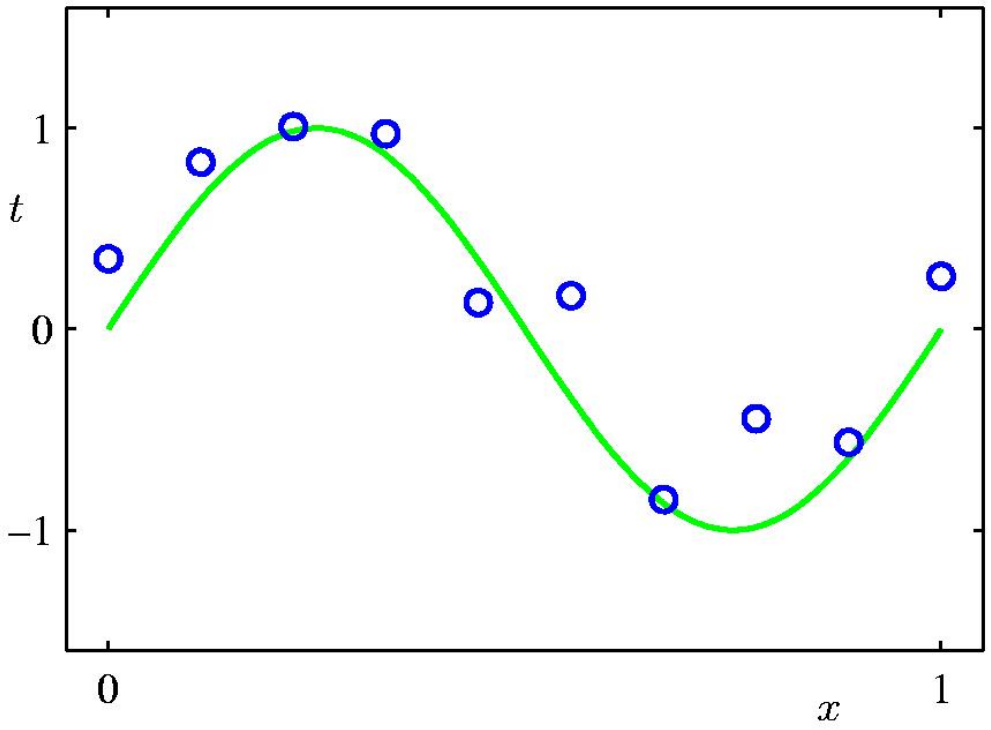
and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

- Remember polynomial curve fitting with more and more complex models (higher degree polynomials)

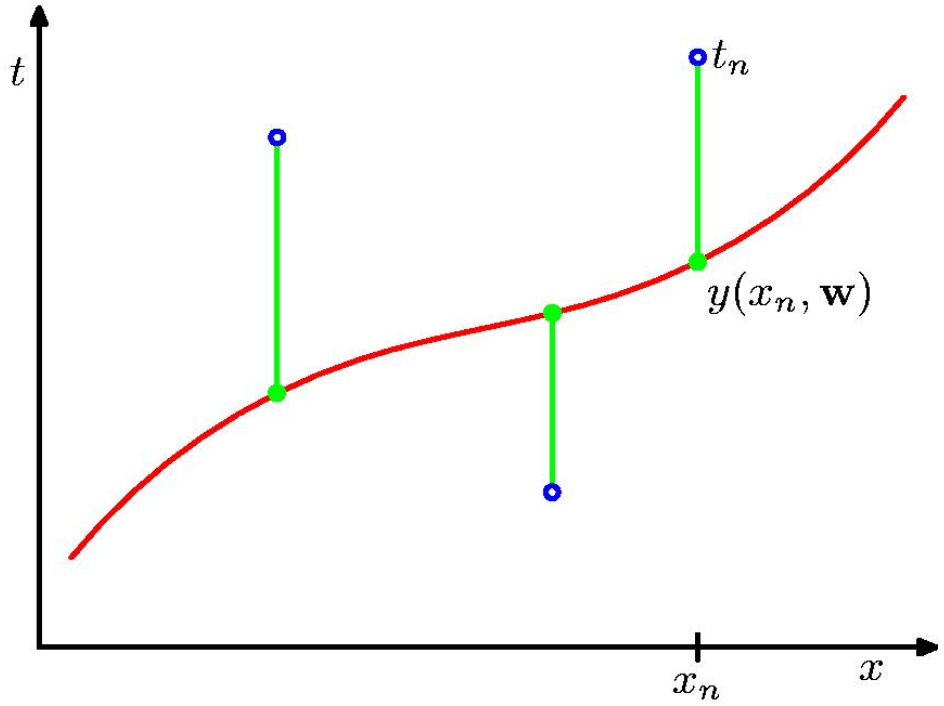
- We will see what happens regarding train versus test accuracies and regarding weights, as models get more complex.

Polynomial Curve Fitting



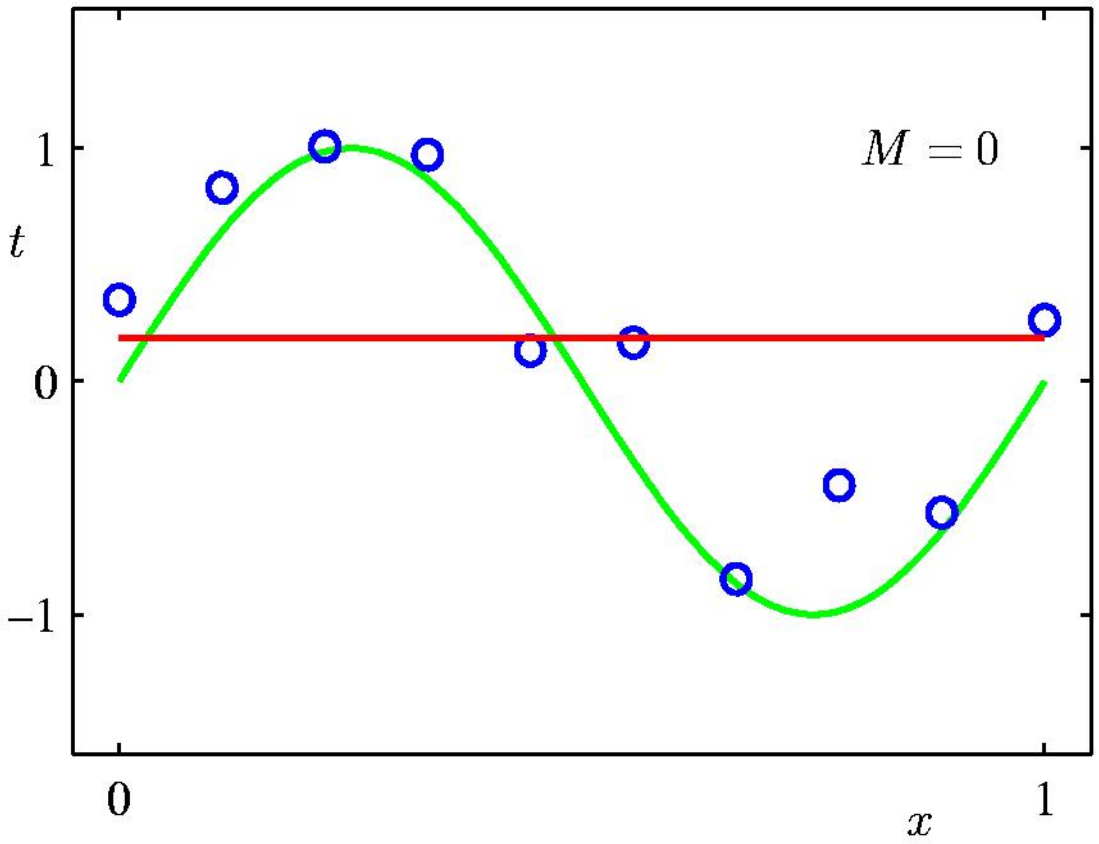
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Sum-of-Squares Error Function

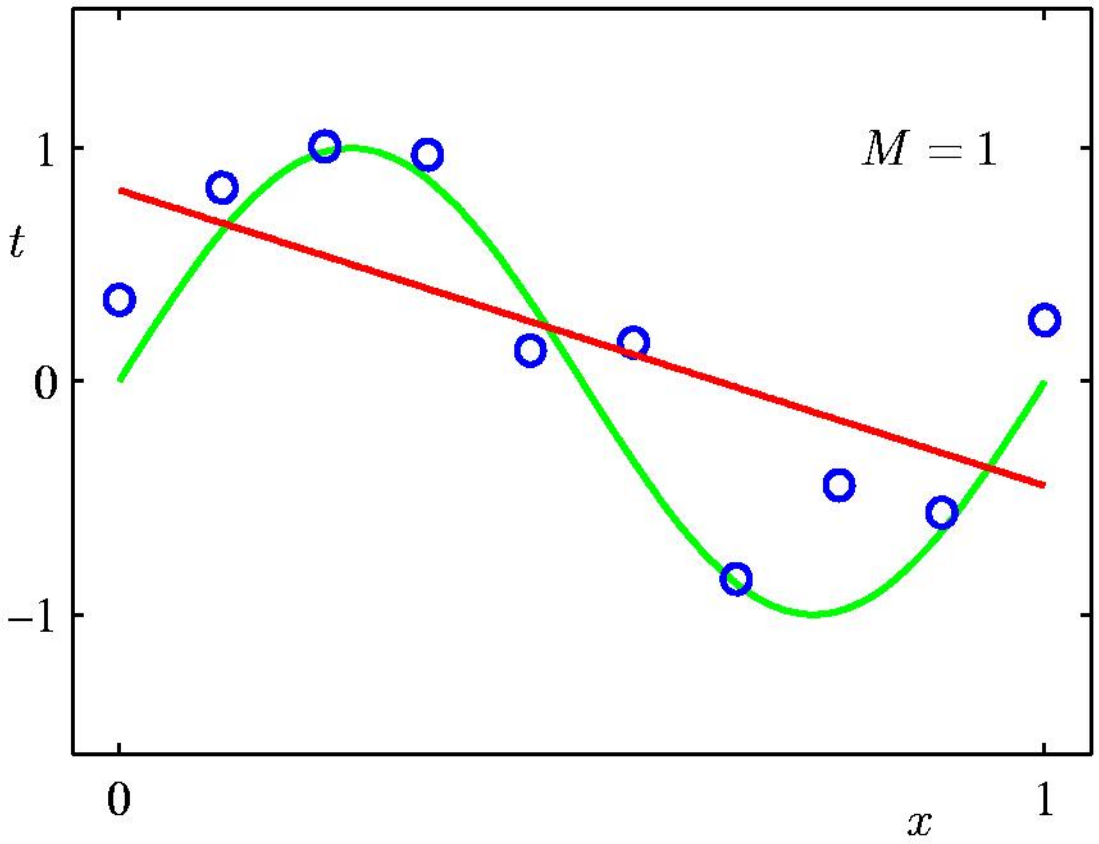


$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

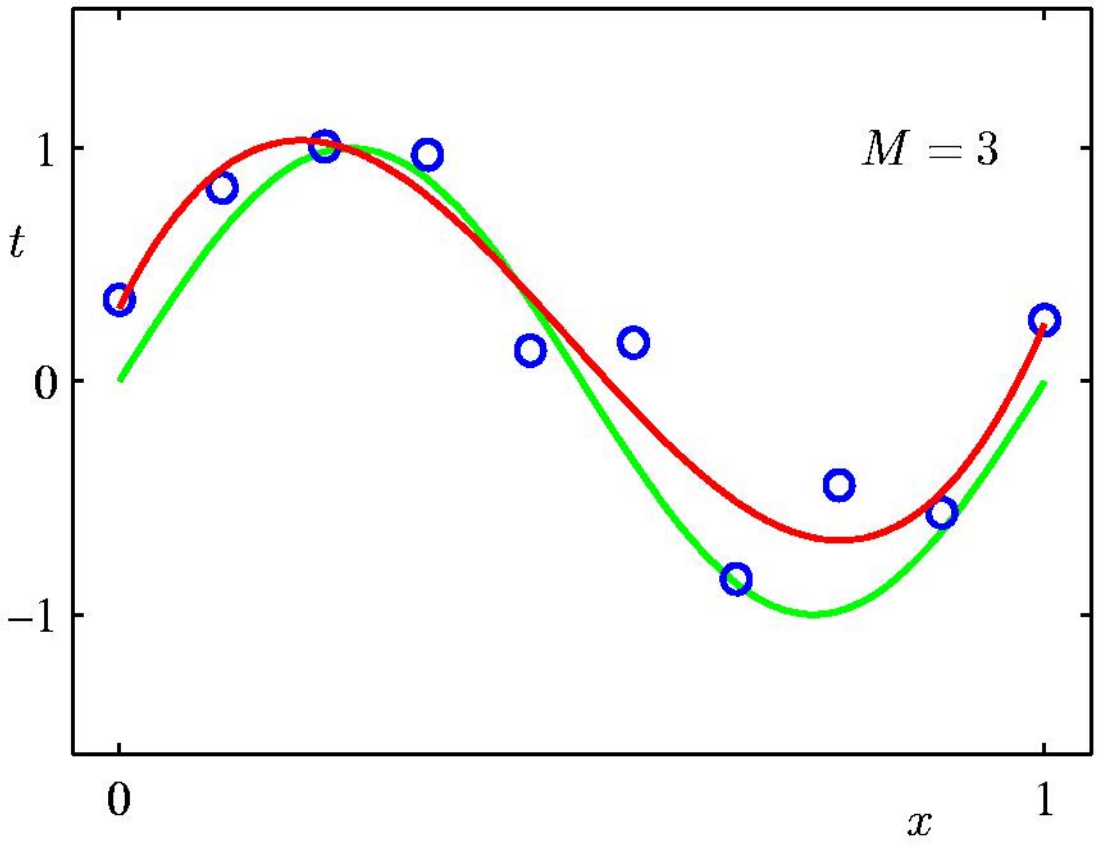
0^{th} Order Polynomial



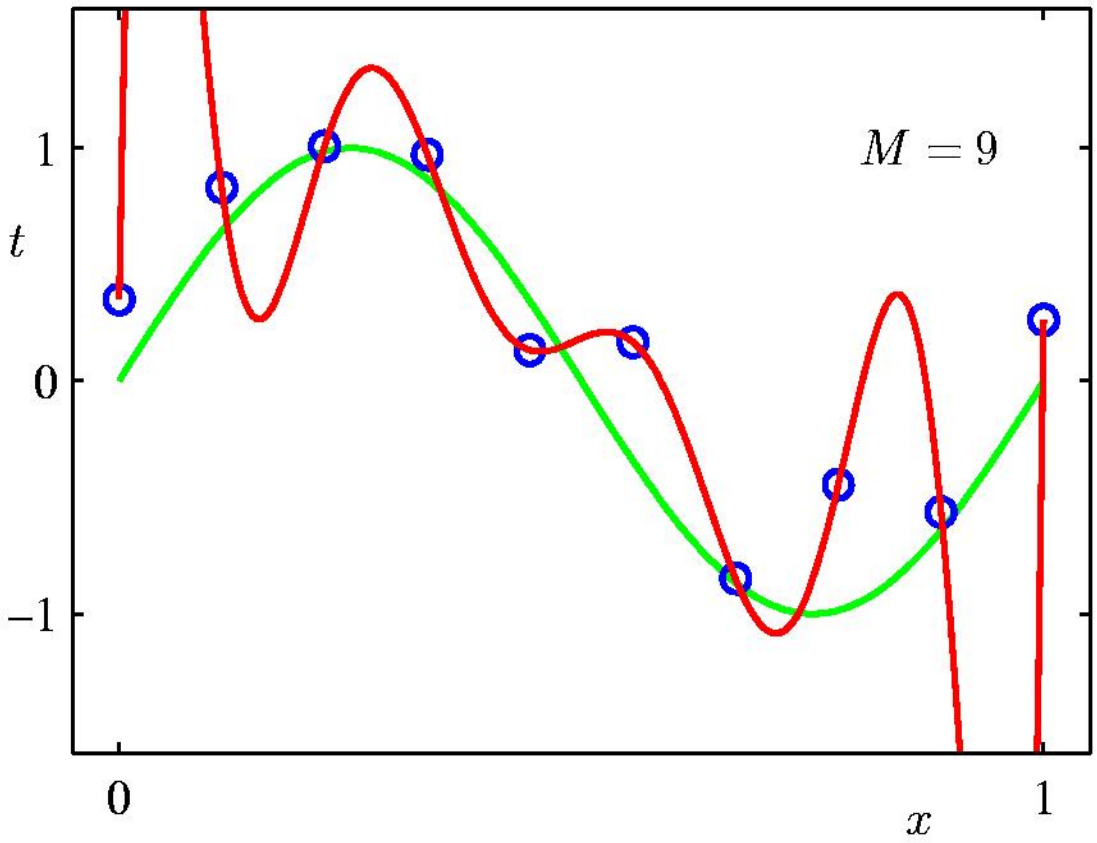
1st Order Polynomial



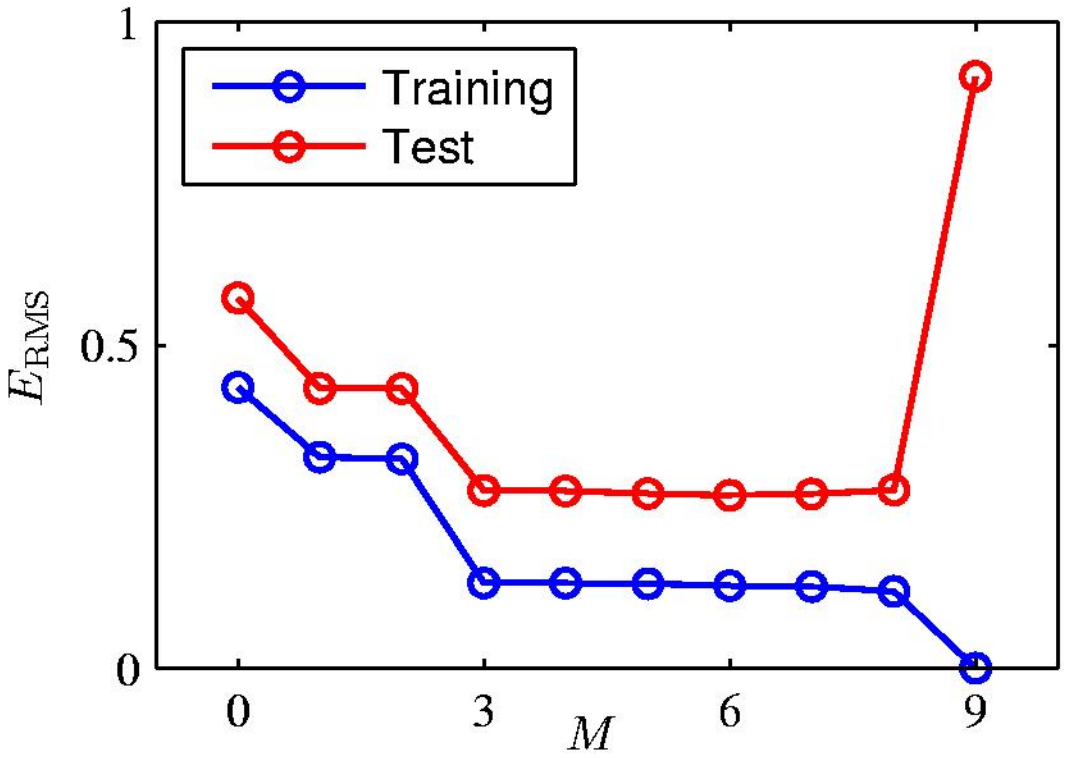
3rd Order Polynomial



9th Order Polynomial



Over-fitting



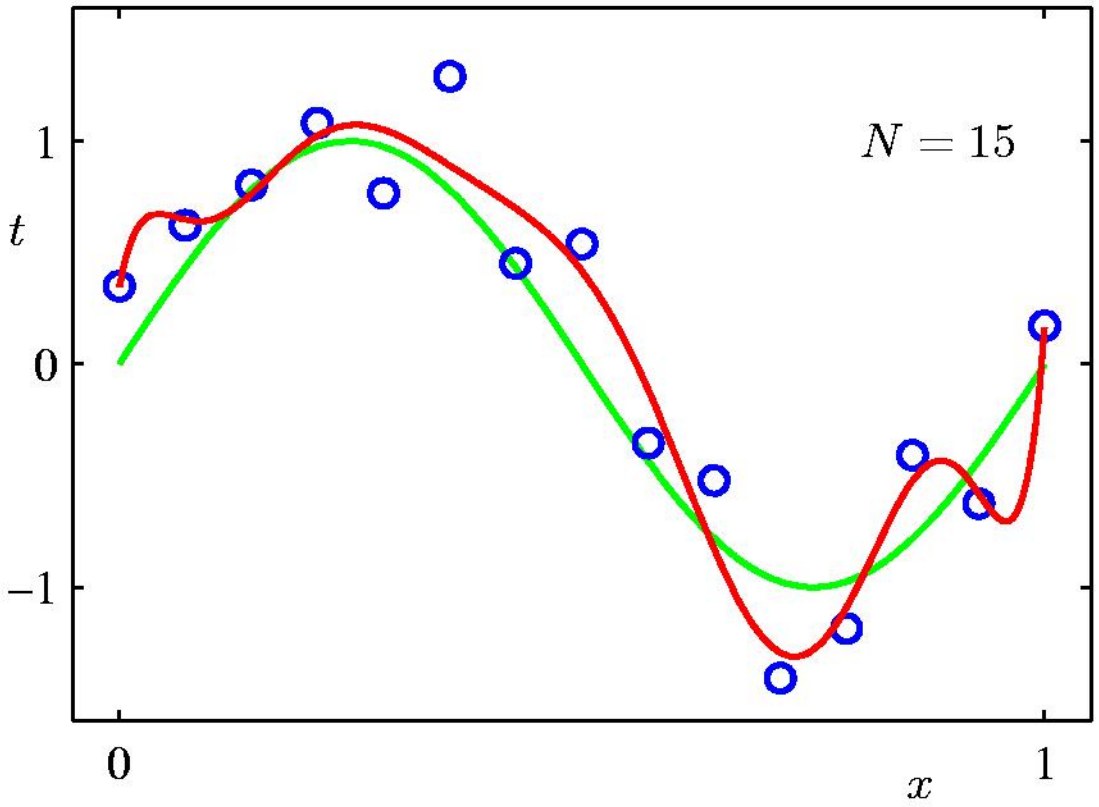
Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$

Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

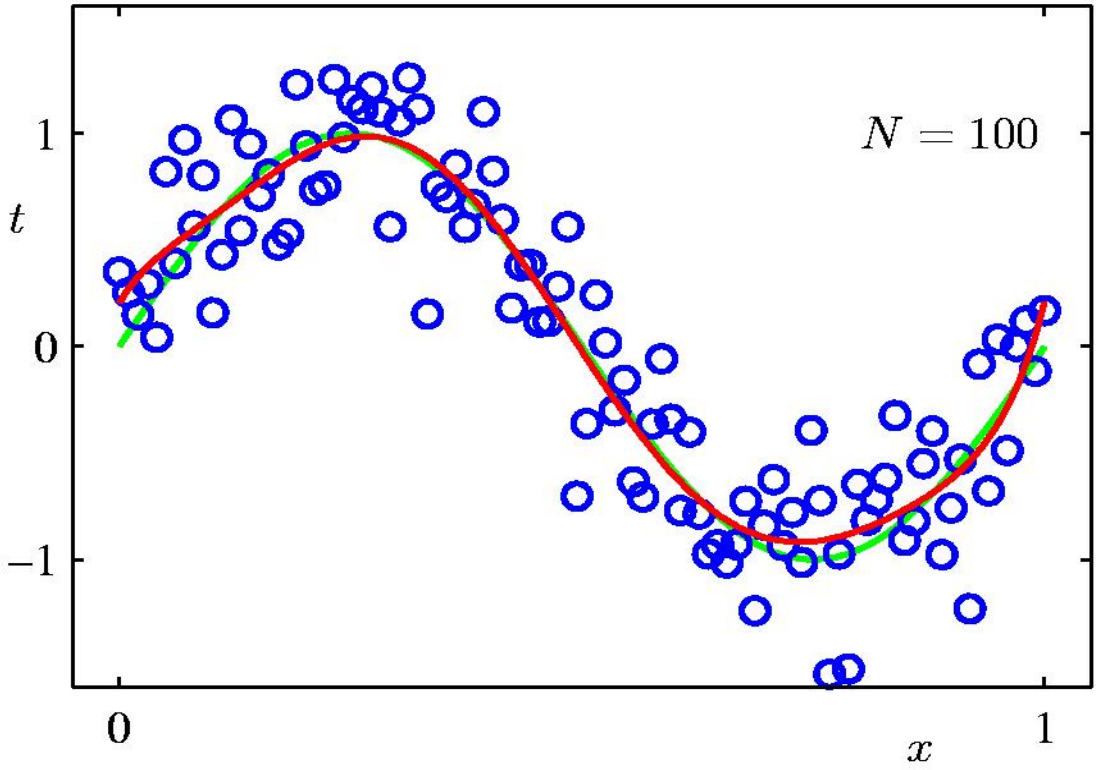
Data Set Size: $N = 15$

9th Order Polynomial



Data Set Size: $N = 100$

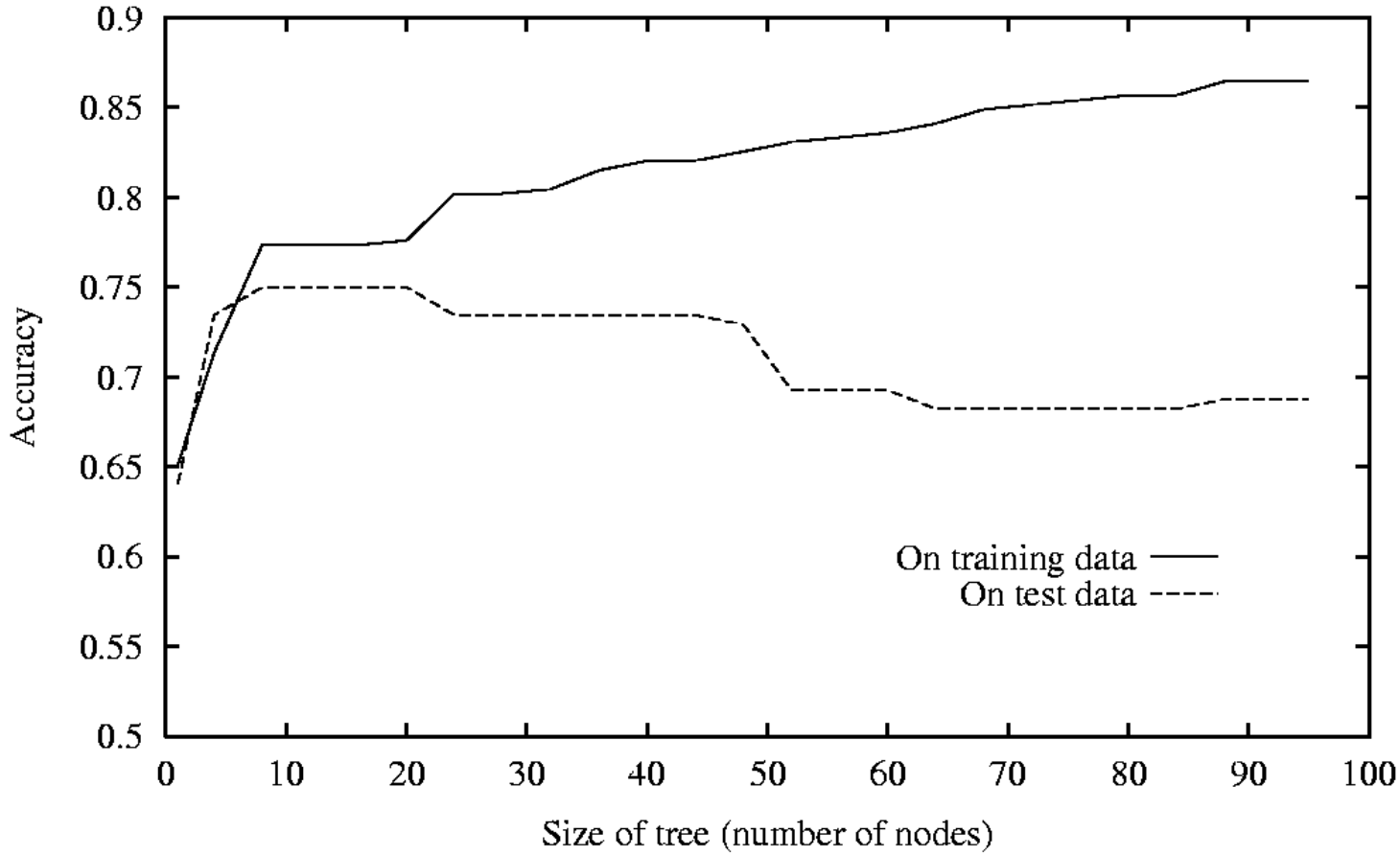
9th Order Polynomial



As data increases, the polynomial is further bound, reducing wild changes.

Back to Decision Trees

Over fitting in Decision Trees



Avoiding over-fitting the data

- How can we avoid overfitting? There are 2 approaches:
  1. **Early stopping**: stop growing the tree before it perfectly classifies the training data
  2. **Pruning:** grow full tree, then prune
     - Reduced error pruning
     - Rule post-pruning

  - **Pruning**: replacing a subtree with a leaf with the most common classification in the subtree.

  - Pruning approach is found more useful in practice.

- Whether we are pre or post-pruning, the important question is how to select “best” tree:
  - Measure performance over separate validation set
  - Measure performance over training data
    - apply a statistical test to see if expanding or pruning would produce an improvement beyond the training set (Quinlan 1986)
  - MDL: Minimum description length
  - …

Reduced-Error Pruning (Quinlan 1987)

- Split data into *training* and *validation* set
- Do until further pruning is harmful:
  - 1. Evaluate impact of pruning each possible node (plus those below it) on the *validation* set
  - 2. Greedily remove the one that most improves *validation* set accuracy
- Produces smallest version of the (most accurate) tree
- What if data is limited?
  - We would not want to separate a validation set.

Advanced: Alternative: Rule post-pruning

- Algorithm
  - Build a complete decision tree.
  - Convert the tree to set of rules.
  - Prune each rule:
    - Remove any preconditions if any improvement in accuracy
  - Sort the pruned rules by accuracy and use them in that order.

- Perhaps most frequently used method (e.g., in C4.5)

- More details can be found in
  http://www2.cs.uregina.ca/~hamilton/courses/831/notes/ml/dtrees/4_dtrees3.html
  (read only if interested)

Curse of Dimensionality

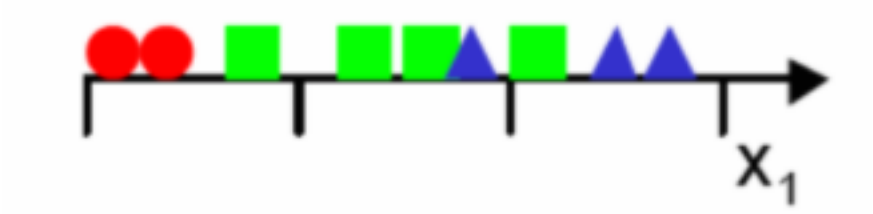
Curse of Dimensionality

- In a learning task, it seems like adding more attributes would help the learner, as more information never hurts, right?

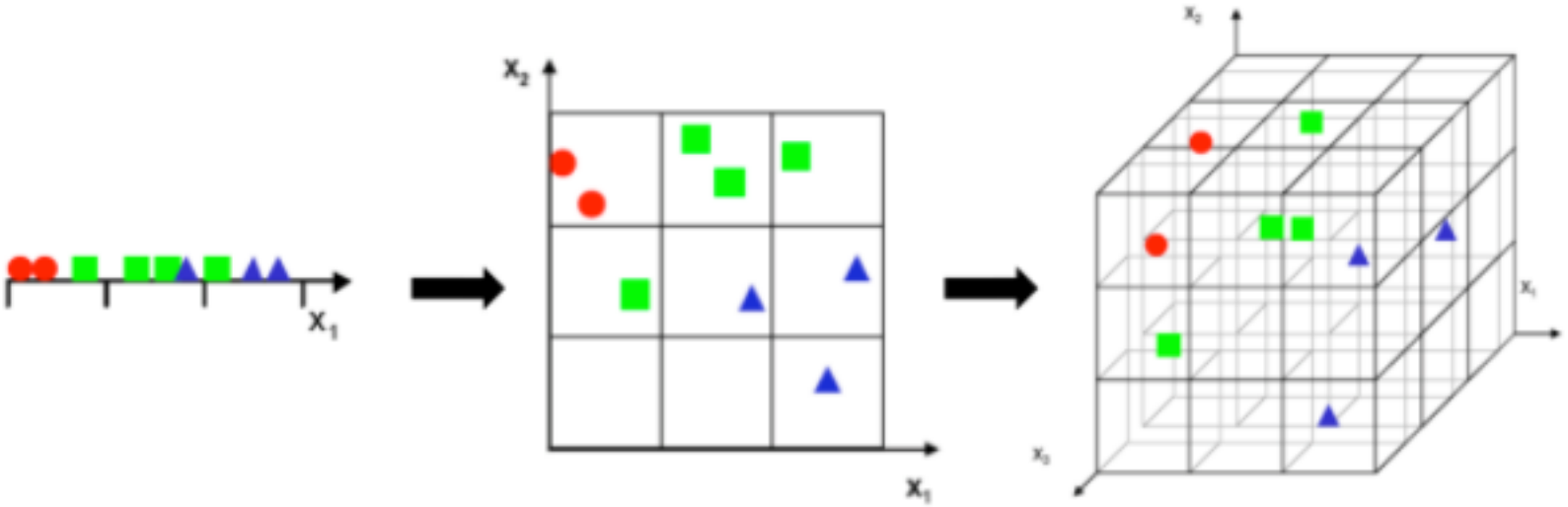
- In fact, sometimes it does, due to what is called **curse of dimensionality.**

- In a toy learning problem (tx to Gutierrez-Osuna), our algorithm:
  - divides the feature space uniformly into bins and
  - for each new example that we want to classify, we just need to figure out the bin the example falls into and find the predominant class in that bin as the label.

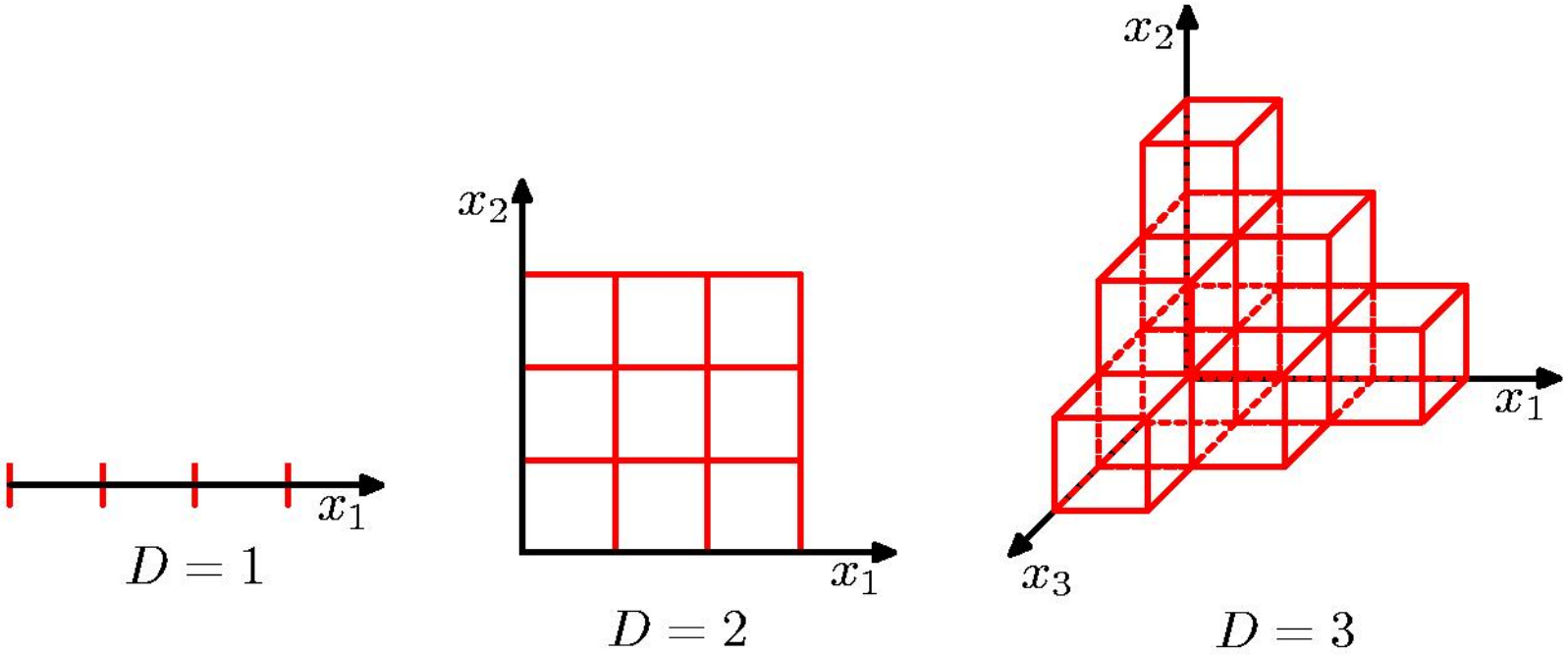
  - **One feature where the input space is divided into 3 bins:**



  - **Noticing the overlap, we add one more feature:**


X1
X2
X1
X3
X1
X2

Curse of Dimensionality



As dimensionality D increases, the amount of data needed increases exponentially with D.

- Examples in high dimensional spaces, irrelevant attributes,...

- We can still find effective techniques applicable to high-dimensional spaces
  - Real data will often be confined to a region of the space having lower effective dimensionality
  - Real data will typically exhibit smoothness properties
- Feature selection
- Dimensionality reduction
- Controlling model complexity
- ...