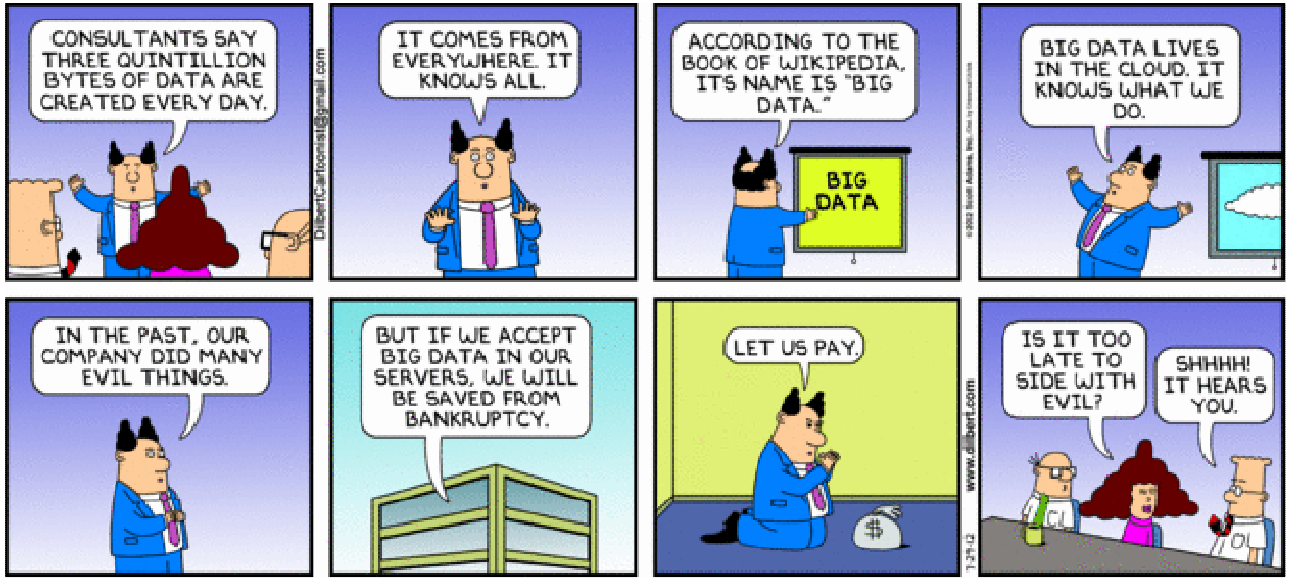
Learning from Observations

CS4881 Artificial Intelligence

Jay Urbain, PhD

Outline

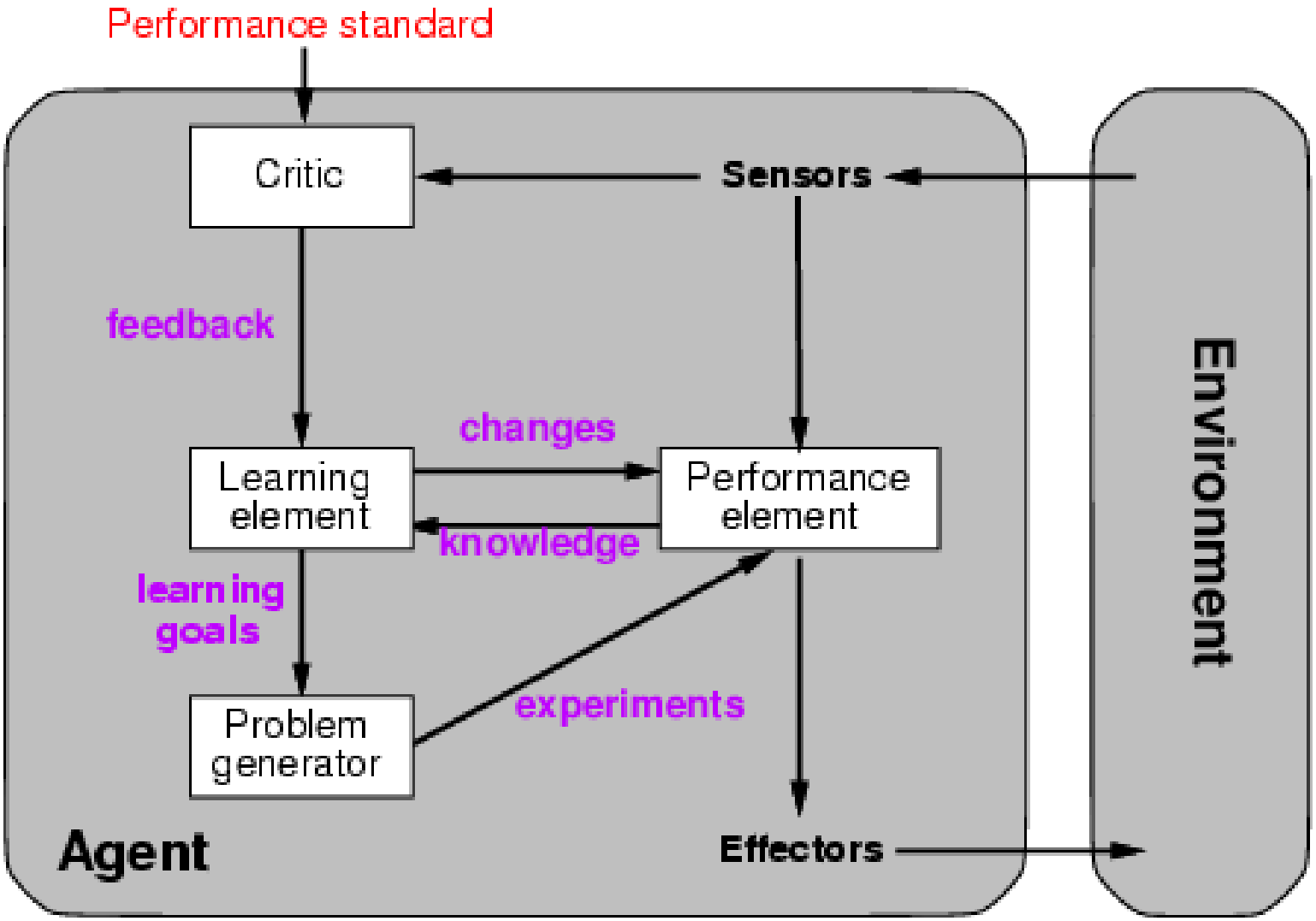
- Learning agents
- Inductive learning
- Decision tree learning



Learning

- Learning is essential for *unknown environments*,
  - i.e., when designer lacks omniscience.
- Learning is useful as a *system construction method*,
  - i.e., expose the agent to reality rather than trying to write it down.
- Learning modifies the agent's *decision mechanism* to improve performance.

Learning agents



Learning element

- Design of a learning element is affected by
  - Which components of the performance element are to be learned
  - What feedback is available to learn for these components
  - What representation is used for the components

- Type of feedback:
  - Supervised learning: correct answers for each example
  - Semi-supervised learning: correct answers for some examples
  - Unsupervised learning: correct answers not given
  - Reinforcement learning: occasional rewards

Machine Learning

1. Supervised learning algorithms
   - Naïve Bayes
   - Decision Tree
   - Neural Networks
   - Support Vector Machines
2. Unsupervised learning
   - Association Rules
   - Clustering
   - Markov Chain Monte Carlo
3. Reinforcement learning (later)
   - Q-learning, Temporal Difference learning
4. Semi-Supervised learning
   - Use limited set of labeled examples to bootstrap learning
5. Active learning

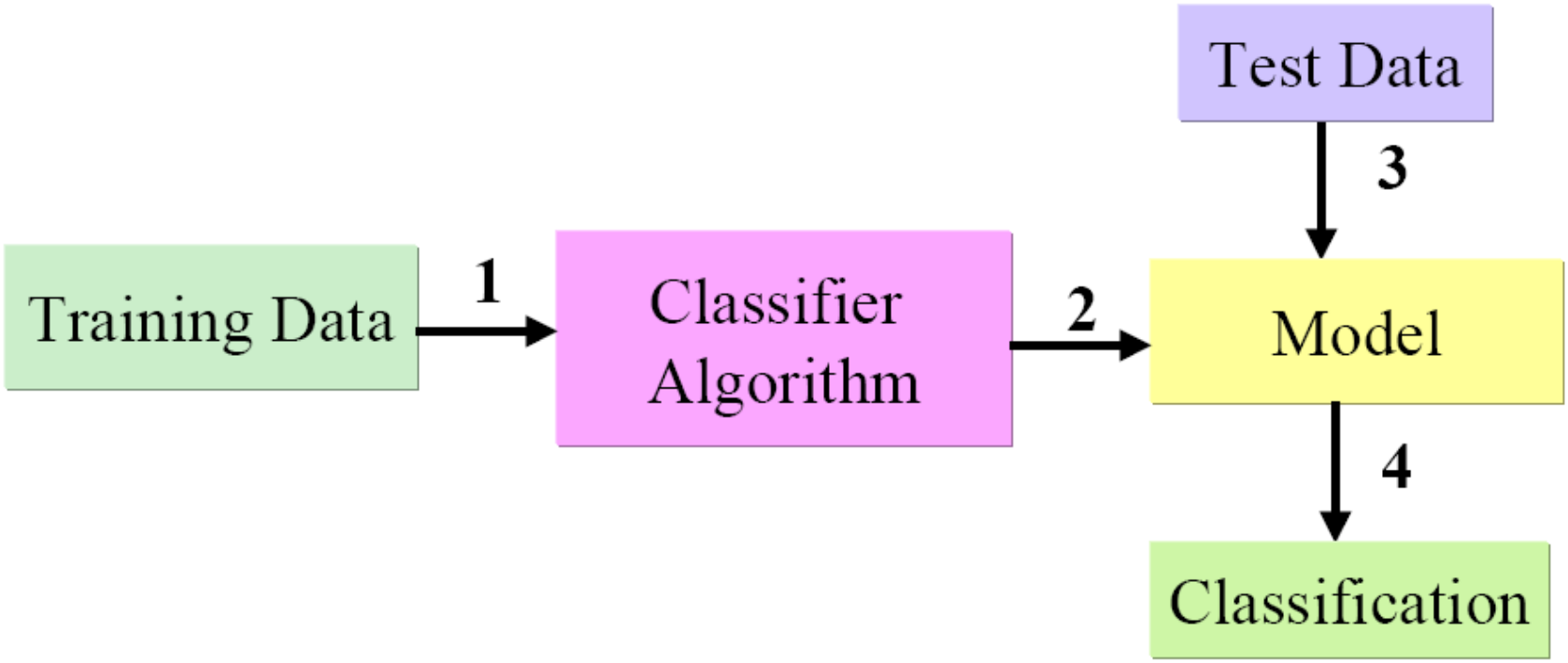
Machine Learning Algorithms

- **Supervised algorithms** - learn by example.
  1. *Use training data* which has correct answers (class label).
  2. *Create a classification model* (performance element) by running the algorithm on the training data.
  3. Test the model. If accuracy is low, regenerate model after changing features, training samples, etc.
  4. Use model to *predict class* label for new incoming data.

- **Unsupervised algorithms** – *find hidden relationships* in data
  - Do not use training data.
  - Classes may not be known in advance.

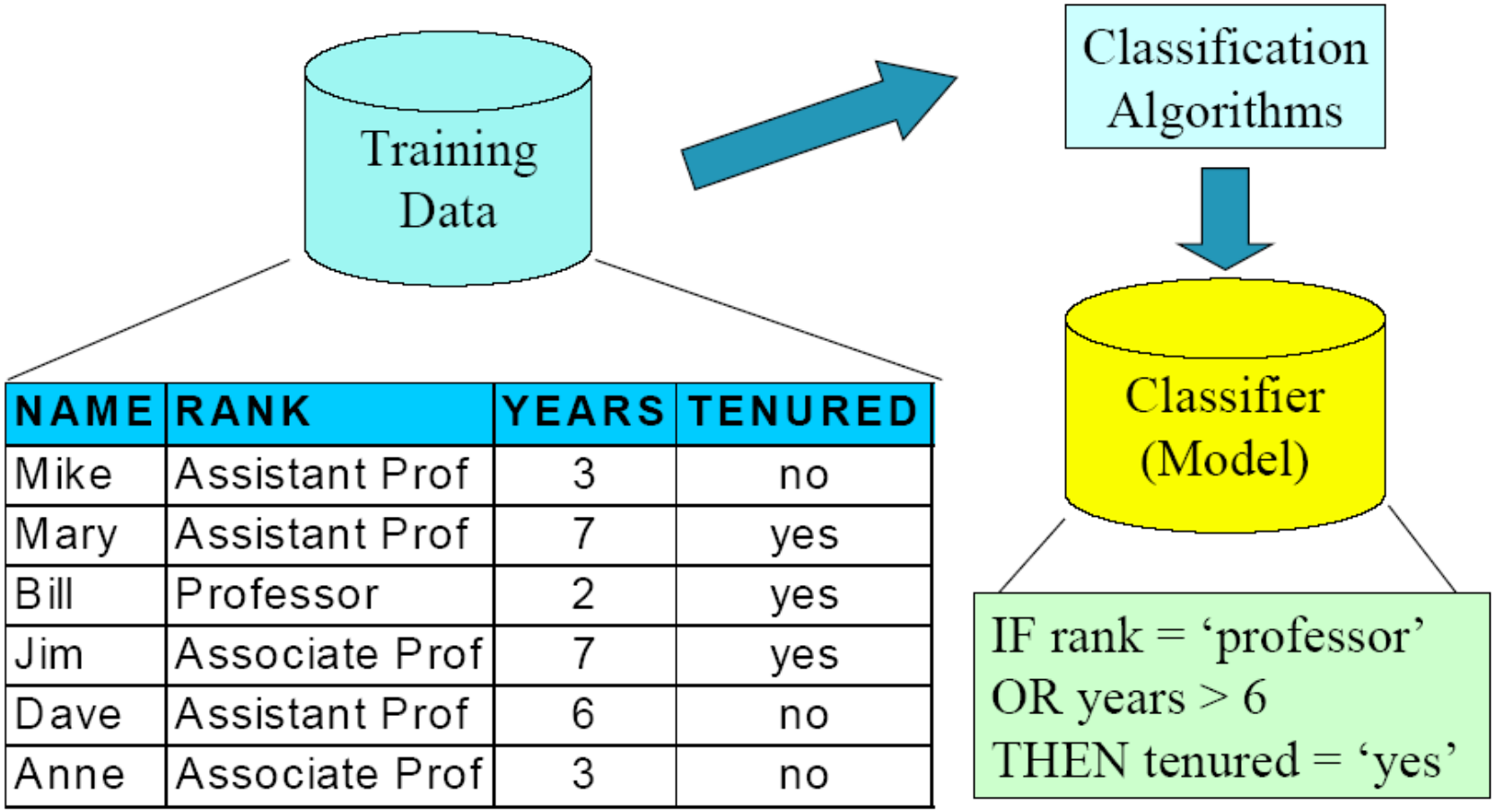
- **Reinforcement learning** –
  - How an *agent* ought to take *actions* in an *environment* so as to *maximize some notion of long-term reward.*

Supervised Learning



Classification

Taken from: Jiawei Han and Micheline Kamber "Data Mining and Concepts"



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

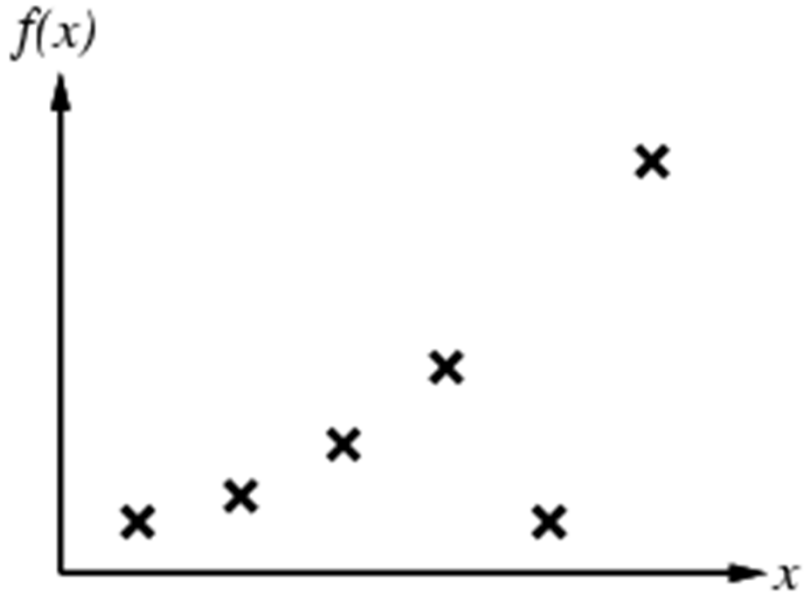
Inductive learning

Simplest form: learn a function from examples

- Type of *reasoning* that involves moving from a set of specific facts to a general conclusion.
- *f* is the target function
- An example is a pair (*x*, *f(x)*)
- Problem: find a hypothesis *h*
  - such that $h \approx f$
  - given a training set of examples
- This is a highly simplified model of real learning:
  - Ignores prior knowledge
  - Assumes examples are given

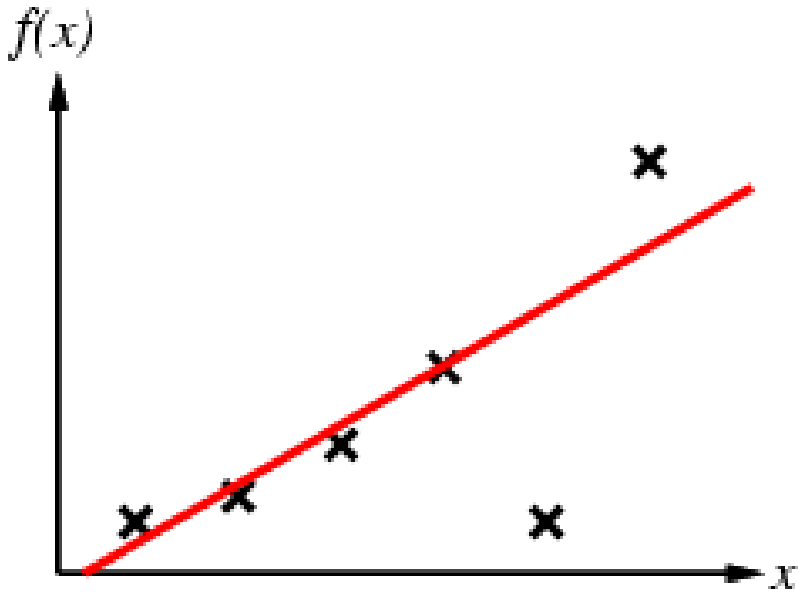
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



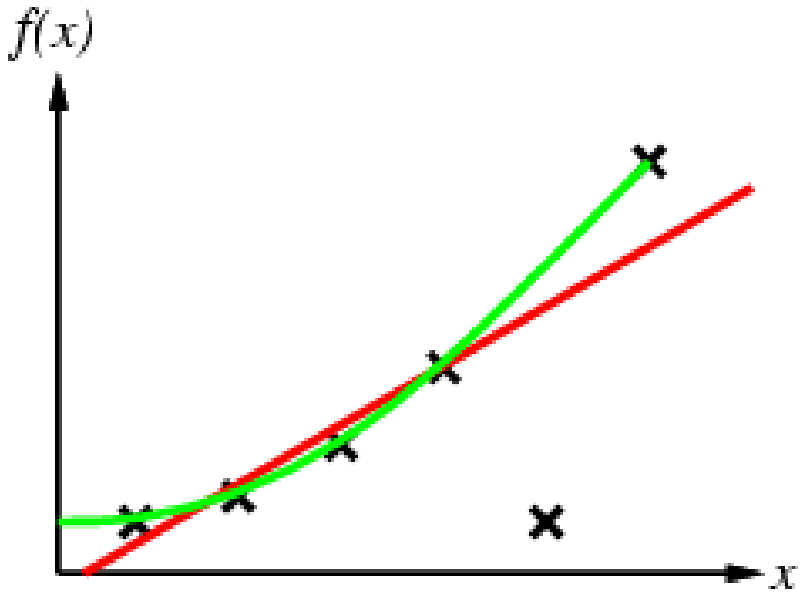
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



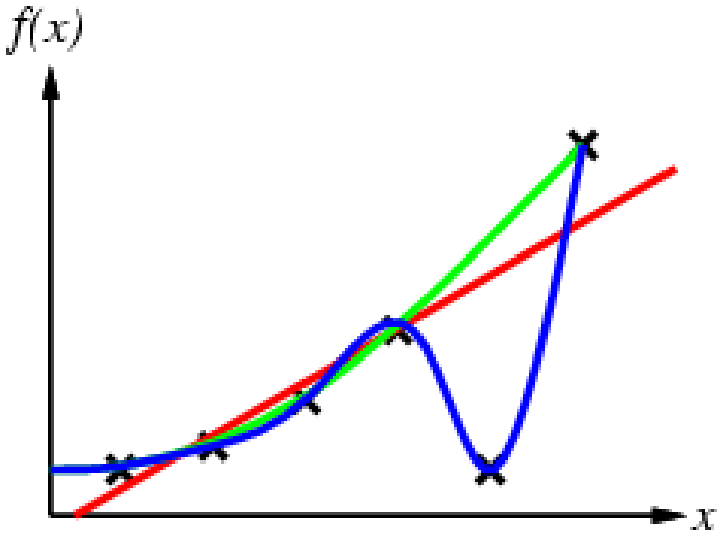
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



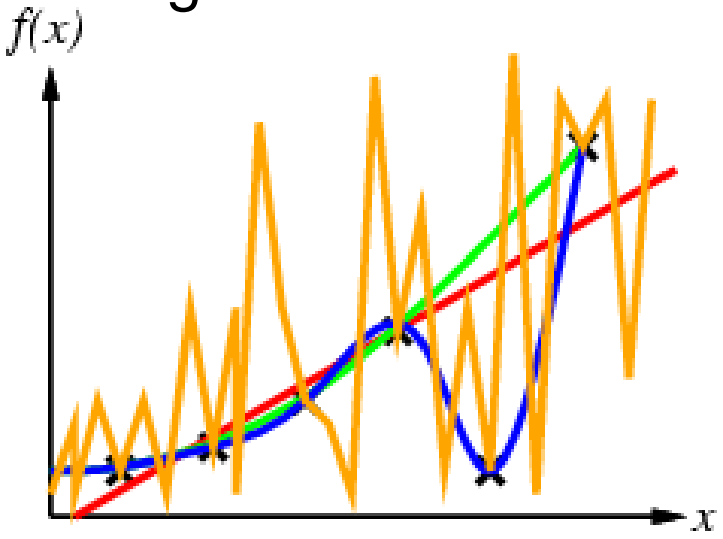
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



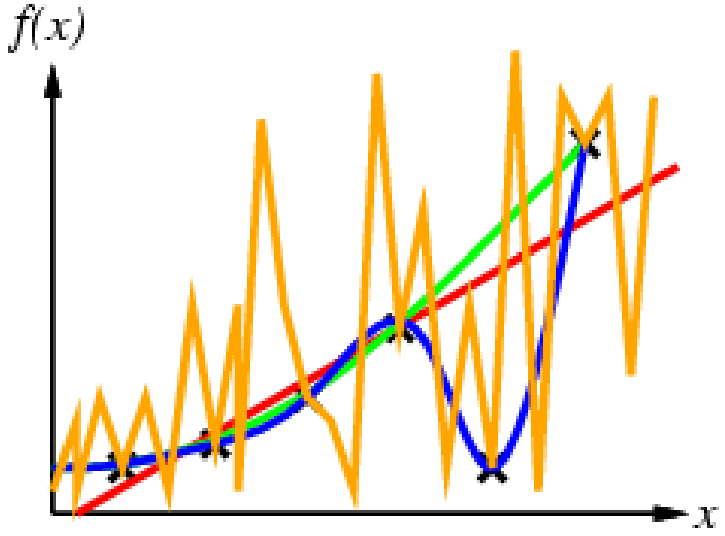
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



- Ockham's razor: prefer the simplest hypothesis consistent with the data

Review conditional probability (skip)

- Can factor joint probability using chain rule:

  $P(a, b) = P(a \mid b)\, P(b) = P(b \mid a)\, P(a)$

- This is amazing, we can express joint probability by conditioning on a *or* b:

  $P(a \mid b)\, P(b) = P(b \mid a)\, P(a)$

- … and derive Bayes Theorem:

  $P(a \mid b) = P(b \mid a)\, P(a) / P(b)$

  $P(b \mid a) = P(a \mid b)\, P(b) / P(a)$

Naïve Bayes Classifier

- Lets say we have a hypothesis, & we want to calculate the probability of the hypothesis being correct.

- Hypothesis:
  - given a set of features (gene expression levels) what is the probability that a set of genes are co-expressed for cancer?
  - Where $\mathbf{X}=\{x_1, x_{2,\ldots,}x_n\}$ is the set of gene expression levels.

- Classification probabilities:
  - *P(C=genes co-expressed | **X**)*
  - *P(not C=genes not co-expressed | **X**)*

Bayes Theorem - Learning

1) From training data calculate *P(X|C), P(X),* and *P(C).*

- P(X|C) *Likelihood* of X conditioned on class *C*
  - Shows the confidence/probability of X given C from training data.
  - Given *C* is true, calculate probability that feature set **X,** where **X=** {x_1, x_2} has the following feature values:
  - x_1= gene 1 expressed
  - x_2= gene 2 expressed.
- P(X) *Prior* probability of **X** (normalization constant).
  - Represents the *prior* probability that x_1 and x_2 are expressed independent of us knowing anything else.
- $P(C_i)$ is the ratio of total samples with class C_i = cancer to all samples.

Bayes Theorem - Learning

- Hypothesis that the 2 genes are co-expressed for cancer is the class C_i = cancer
  - Note: *P (X)* can be ignored as it is constant for all classes. For proper probabilities include in denominator.

- Assuming the independence assumption, $P(X/C_i)$ is:

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

Naïve Bayes Classification

- P(C|X) *Posterior* probability of hypothesis class C
  - X: {x_1, $x_{2...}$, x_n}
  - Shows the confidence/probability of C given **X**
  - x_1: *gene 1 expression level*, x_2: *gene 2 expression level*
  - C: cancer

$$P(C_i \mid X) = P(C_i) \prod_{k=1}^{n} P(x_k \mid C_i)$$

Naïve Bayes Classification (skip)

- For categorical attribute:
  - $P(x_k/C_i)$ is the frequency of samples having value x_k in class C_i.

- For continuous (numeric) attribute:
  - $P(x_k/C_i)$ is calculated via a Gaussian density function.

Play Tennis? *(Tom Mitchell)*

(skip)

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Play Tennis Example: estimating $P(x_i/C)$ (skip)

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	**P(sunny|n) = 3/5**
P(overcast|p) = 4/9	**P(overcast|n) = 0**
P(rain|p) = 3/9	**P(rain|n) = 2/5**
temperature	
P(hot|p) = 2/9	**P(hot|n) = 2/5**
P(mild|p) = 4/9	**P(mild|n) = 2/5**
P(cool|p) = 3/9	**P(cool|n) = 1/5**
humidity	
P(high|p) = 3/9	**P(high|n) = 4/5**
P(normal|p) = 6/9	**P(normal|n) = 1/5**
windy	
P(true|p) = 3/9	**P(true|n) = 3/5**
P(false|p) = 6/9	**P(false|n) = 2/5**

Play Tennis Example: estimating $P(C_i|x_i)$ (skip)

- An incoming sample: X = <sunny, cool, high, true>
- P(play|X) = P(X|p)*P(p) =
  P(p)*P(sunny|p)*P(cool|p)*P(high|p)*p(true|p)
  9/14 * 2/9 * 3/9 * 3/9 * 3/9= 0.0053
  0.0053/0.0259 = 0.204633

- P(no play|X) = P(X|p)*P(n) =
  P(n)*P(sunny|n)*P(cool|n)*P(high|n)*p(true|n)
  5/14 * 3/5 * 1/5 * 4/5 * 3/5 = 0.0206
  0.795367/0.0259 = 0.795367
  Class n (no play) has higher probability than class p (play) for example X.

Decision Trees (DT)

- Widely used, practical method for inductive inference.
- Method for approximating discrete valued functions.
- Exhaustively search hypothesis space.
- Can be used for representing *if-then-else* rules.

DT Representation

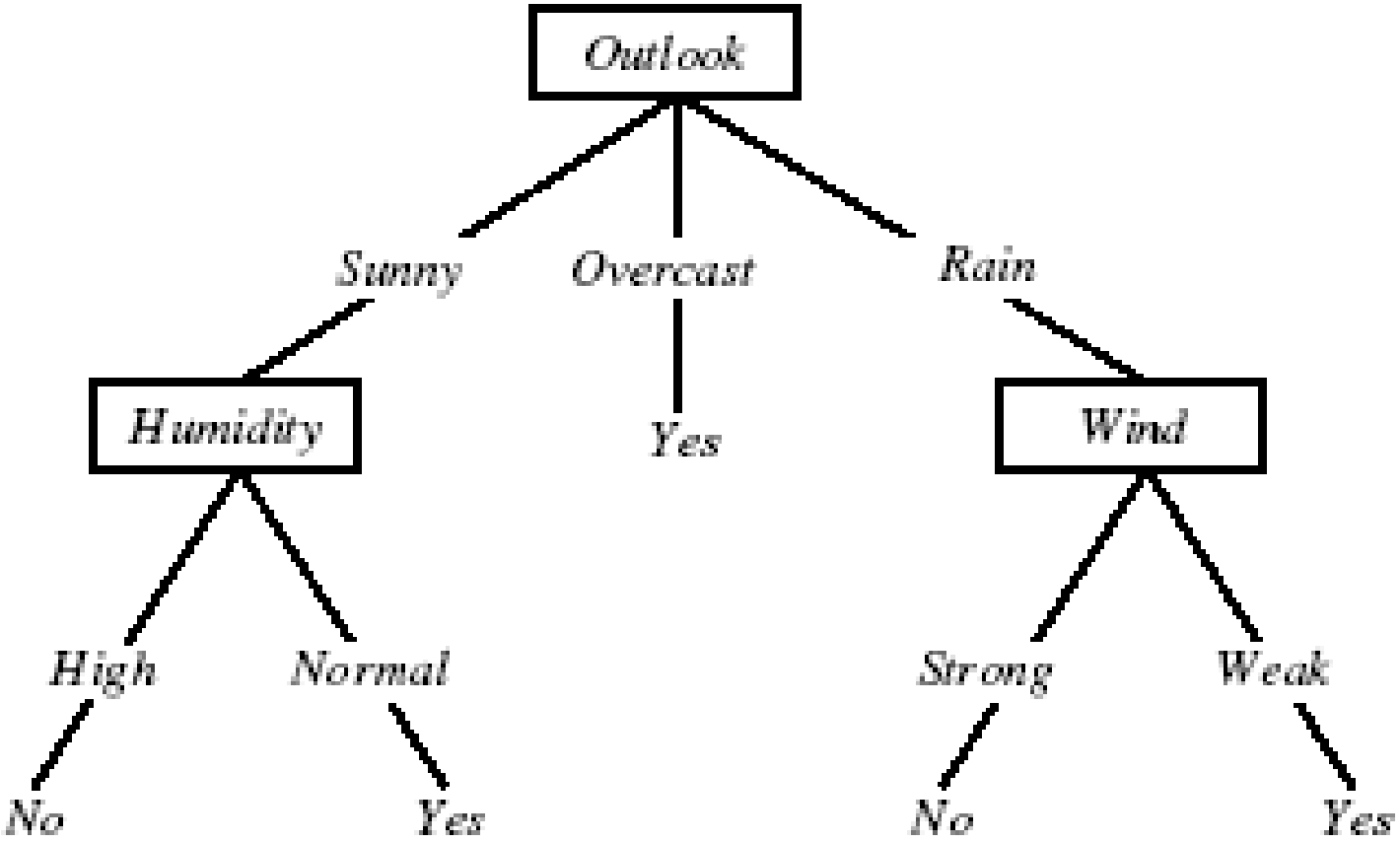
- Classify instances by sorting them down a tree from the root to some leaf node.
- Leaf node provides classification instance.
- Each node in tree specifies a test of some attribute (feature) of the instance.
- Each branch descending from that node represents one of the possible values for this attribute.

DT Classification

Classifying a sample using a decision tree:

1. Starting at the root node of the tree
2. Test the attribute specified by this node.
3. Moving down the tree branch corresponding to the value of the attribute in the given example.
4. Repeat process for each subtree rooted at the new node.
5. Each leaf represents classification of instance.

DT for Play Tennis



Play Tennis Example

- How to classify?:
  - {outlook=sunny, temperature=hot, humidity=high, wind=strong}

- Decision trees represent a *disjunction of conjunctions* of the attribute values of instances.

- For positive classification, the previous decision tree corresponds to:

  (outlook=sunny ^ humidity=high) v (outlook=overcast) v (outlook=rain ^ wind=weak)

Predict C-Section Risk Example

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .2
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Appropriate problems DT's

- Instances describable by *attribute-value* pairs
- Target function is discrete values
  - Note DT's can be modified to handle real valued parameters.
- Disjunctive hypothesis may be required
- Possibly noisy training data

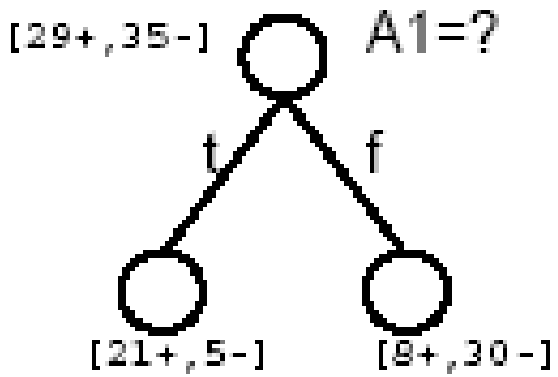
- Examples:
  - Gene expression
  - Equipment, or medical diagnosis
  - Credit risk analysis
  - Modeling calendar scheduling preferences

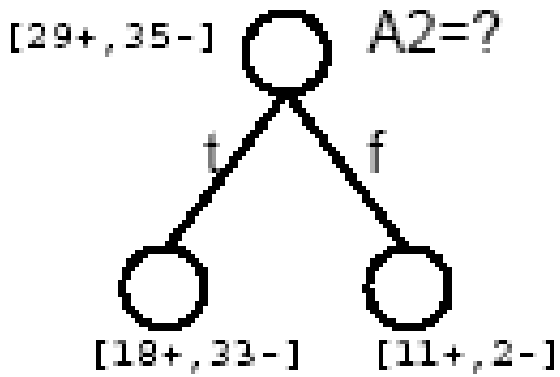
Top-down induction (creation) of DTs

Main loop:

1. *A* <- select 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 current leaf *nodes*
5. If training examples perfectly classified, then STOP
6. Else iterate over new leaf nodes

Which attribute is best?

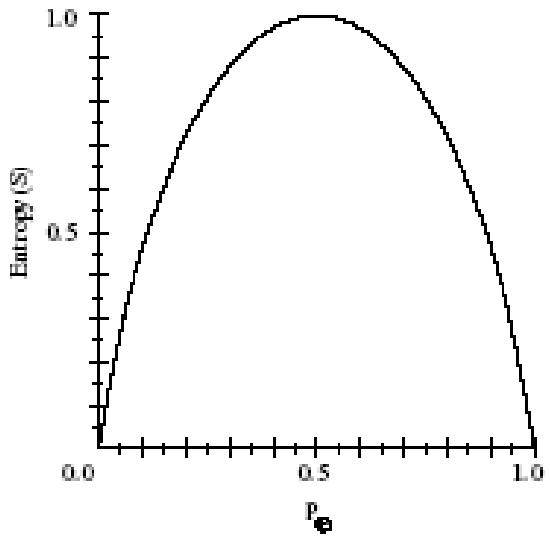




Entropy

Given *S*, a set of training examples:

- Entropy measures the impurity of S
- p+ is proportion of positive examples of S
- p- is proportion of negative examples of S



$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

STOP for Bioinformatics, continue to inflict pain for AI

- Entropy calculation included in the following slides.

Entropy

Entropy(S) = expected number of bits needed to encode class(+/-) of randomly drawn member of S (under the optimal, shortest-length code)

- MDL – minimum description length

Why?

Information theory (Claude Shanon): optimal length code assigns $-log_2 p$ bits to message having probability p.

So, expected number of bits to encode + or – examples of random member of S:

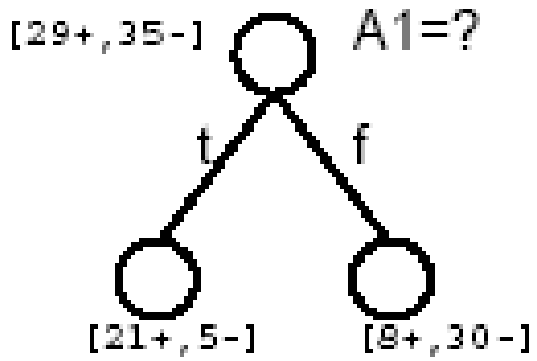
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

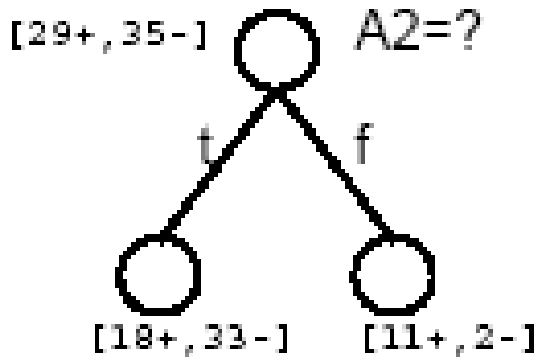
$$Entropy(S) \equiv -p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus}$$

Information Gain

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

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





Calculating Info Gain

entropy(s)	0.940286	// 9/14*(-LOG(9/14,2))+5/14*(LOG(5/14,2))
entropy (s, humidity for high)	0.985228	// 3/7*(-LOG(3/7,2))+4/7*(-LOG(4/7,2))
entropy (s, humidity for normal)	0.591673	// 6/7*(-LOG(6/7,2))+1/7*(-LOG(1/7,2))

Gain(S, Humidity) = entropy(s) - 7/14*entropy (s, humidity for high) - 7/14*entropy (s, humidity for normal)

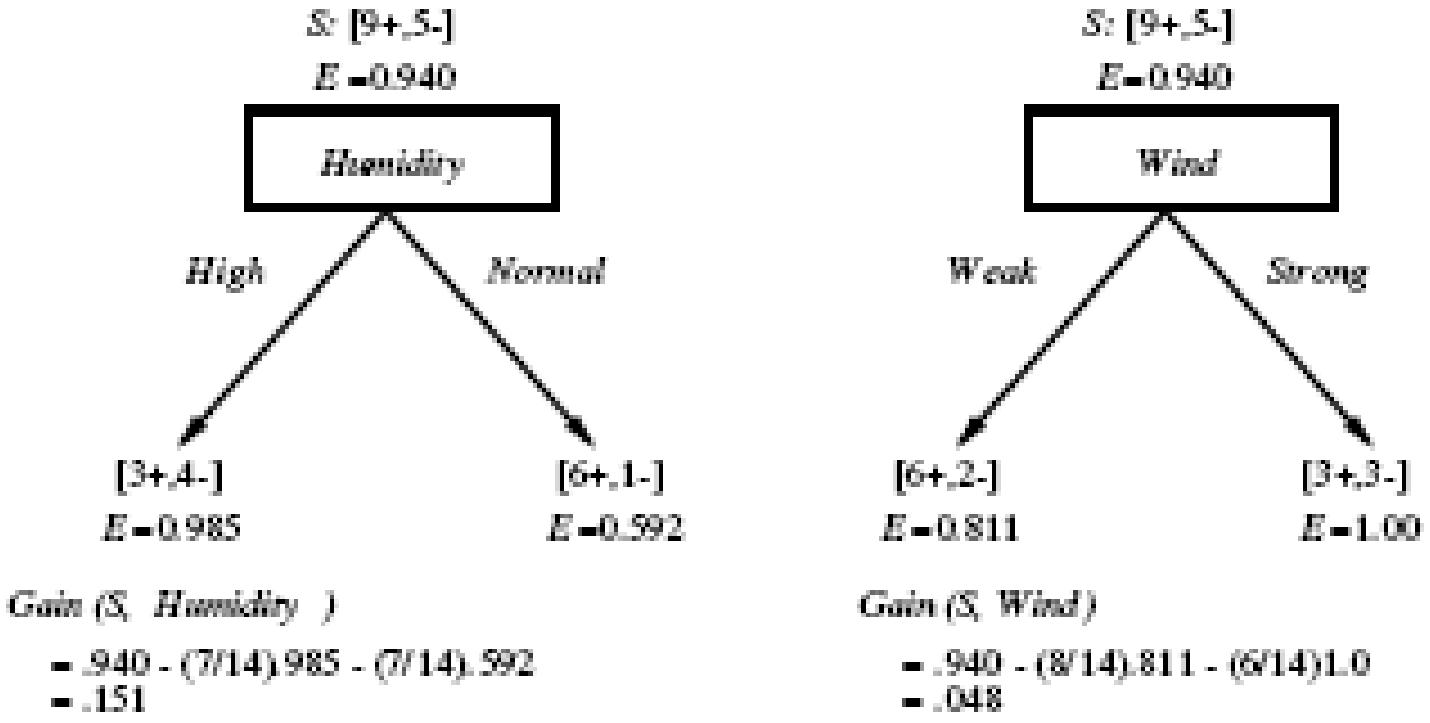
Gain(S, Humidity) =	0.151836	// D2- 7/14*D4 - 7/14*D6

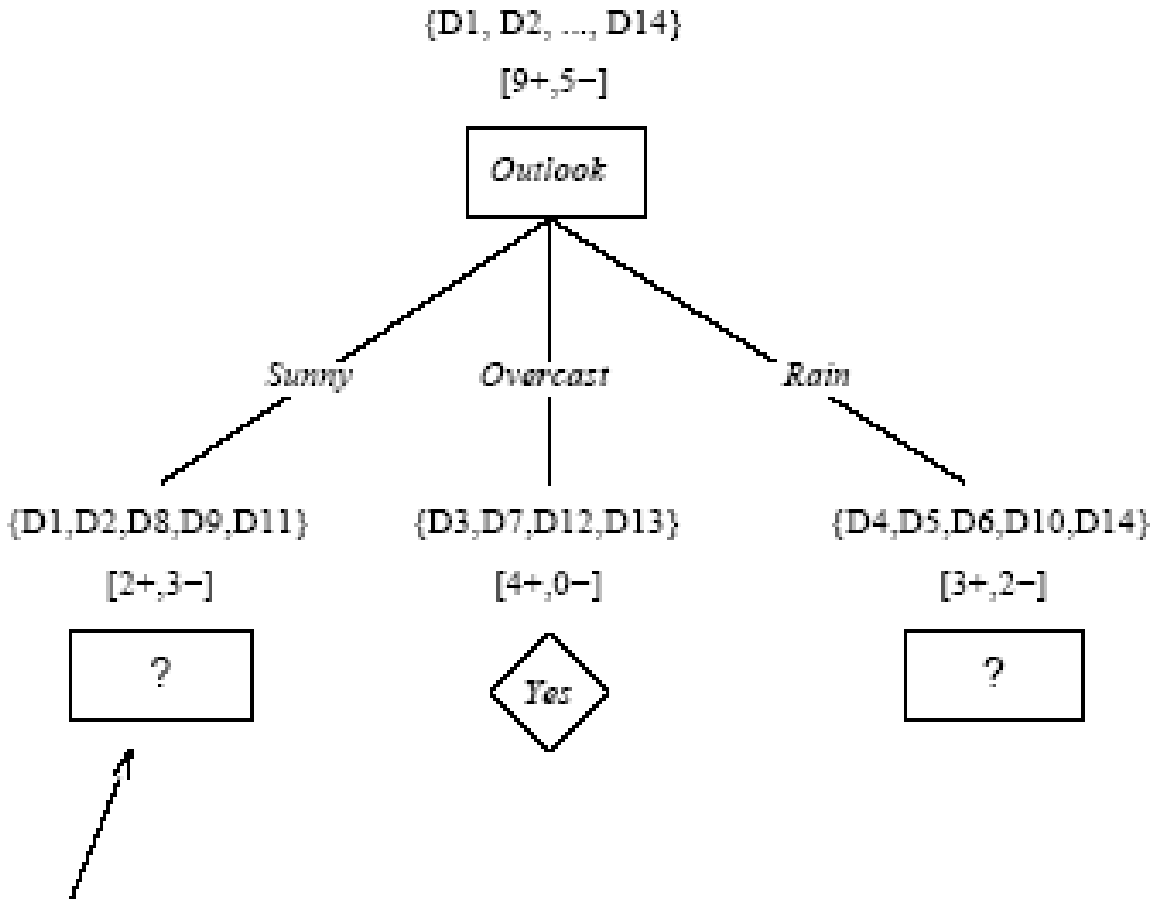
Training Examples

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

Selecting the Next Attribute

Which attribute is the best classifier?





Which attribute should be tested here?

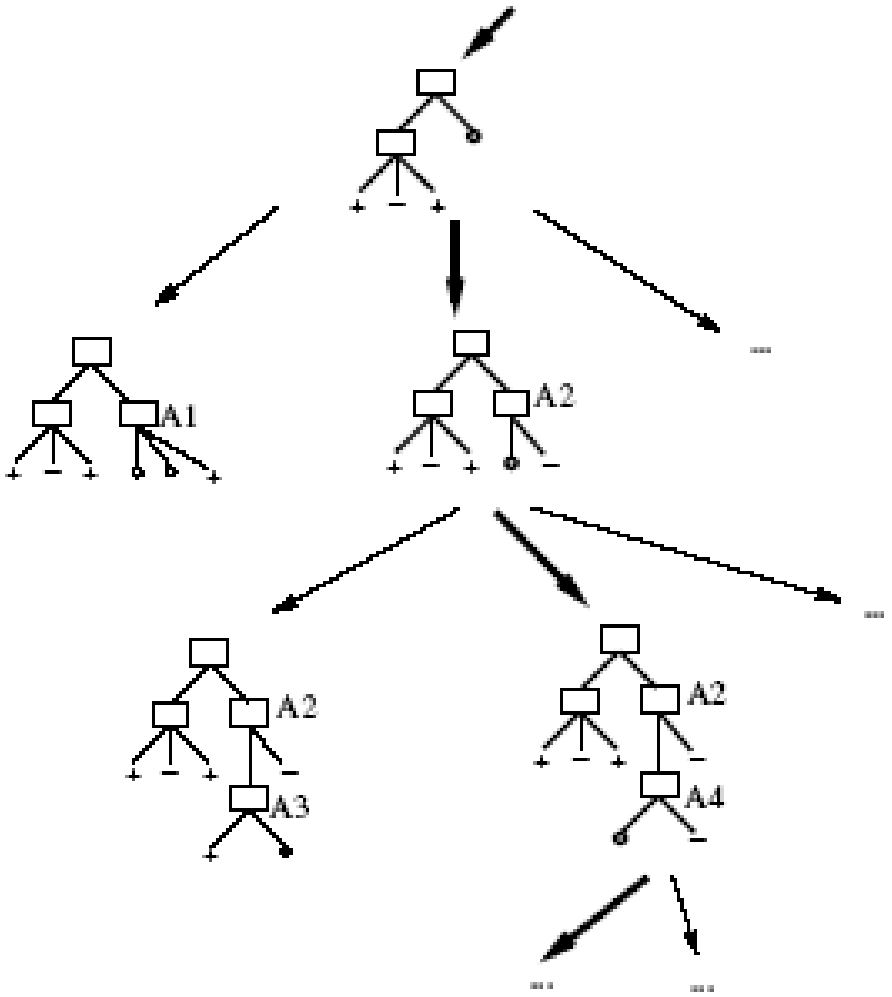
S_{Sunny} = {D1,D2,D8,D9,D11}

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

Hypothesis Space Search by ID3



Hypothesis Space Search by ID3

- Hypothesis space is complete!
  - Target function is surely in there…
- Outputs a single hypothesis
- No backtracing
  - Local minima
- Statiscally based search choices
  - Robust to noisy data
- Inductive bias: approx "*prefer shortest tree*"

Inductive Bias of ID3

Note *H* is the power set of instances *X*

- Unbiased?

Not really…

- Prefer short trees, and for those with high info gain attributes near the root
- Bias is a preference for some *hypothesis*, rather than a restriction of hypothesis space *H*
- *Occam's razor: prefer the shortest hypothesis that fits the data.*

Occam's Razor

Why prefer short hypotheses?

Argument in favor:

- Fewer short hypo's than long hypo's
- A short hypo that fits data unlikely to be coincidence
- A long hypo that fits data might be coincidence

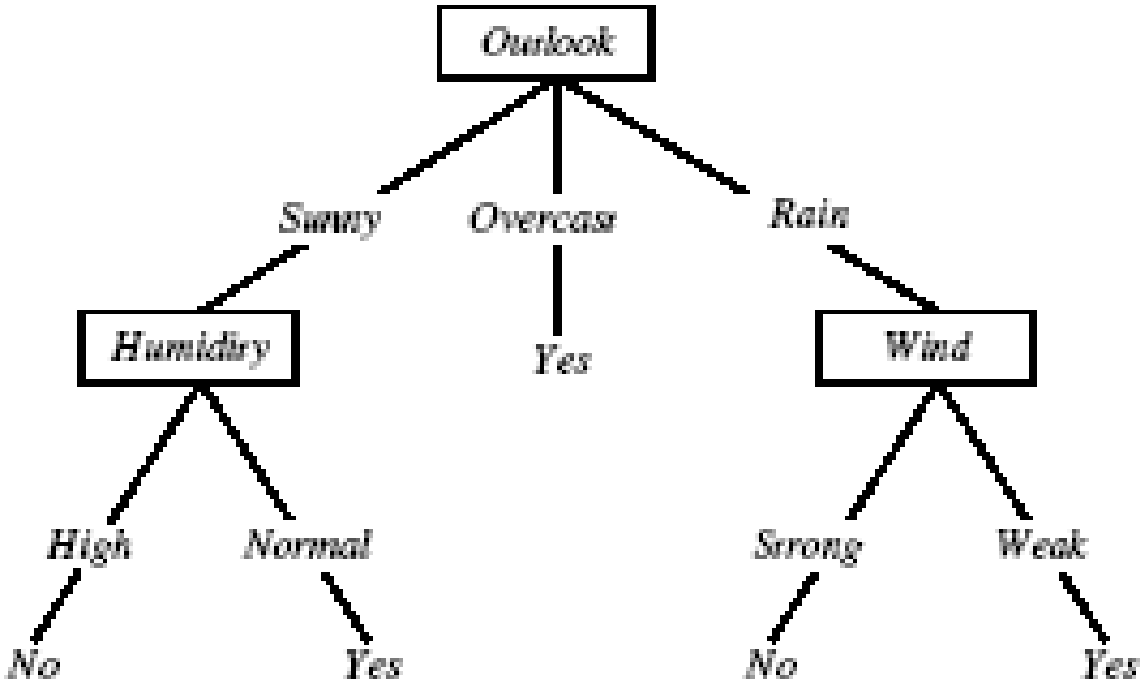
Argument opposed:

- There are many ways to define small sets of hypo's
  - *e.g., all trees with a prime number of nodes use attributes with "Z"*
- *What's so special about small sets based on size of hypothesis?*

Overfitting in Decision Trees

Consider adding noisy training example D15:

- *Sunny, Hot, Normal, Strong, Play tennis=No*



Overfitting

- Consider error of hypothesis *h* over:
  - Training data: $error_{train}(h)$
  - Entire distribution *D* of data: $error_D(h)$

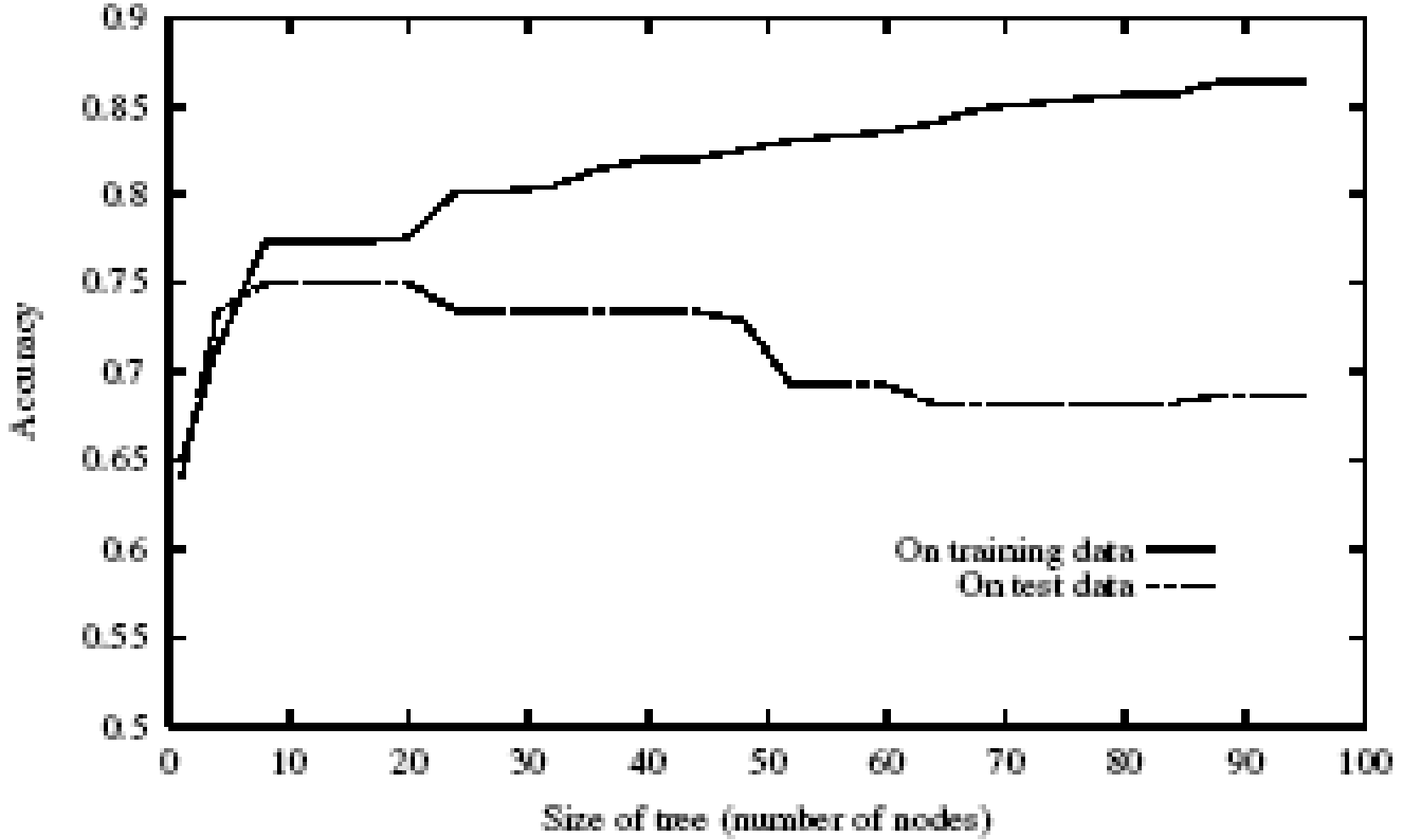
- Hypothesis $h \in H$ *overfits* training data if there is an alternative hypothesis $h' \in H$ such that:

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

and

$$error_D(h) > error_D(h')$$

Overfitting



Avoiding Overfitting

Avoid Overfitting

1. Stop growing when data split is not statistically significant
2. Grow full tree, then prune

How to select best tree:

- Measure performance over training data
- Measure performance over separate validation set

Minimum Description Length

- *MDL: minimize (size(tree)+size(missclassification))*

Reduced-Error Pruning

- Start with completed tree
- Split data into training and validation set
- Do until further pruning is harmful:
  1. Evaluate impact on validation set of pruning each possible node (plus those below it)
  2. *Greedily* remove the one that most improves validation set accuracy

- Produces smallest version of most accurate subtree

Rule Post-Pruning

1. Convert tree to equivalent rules
2. Prune each rule independently of others
3. Evaluate impact on validation set
4. Sort final rule into desired sequence of use.

- Most frequently used method, e.g., C4.5

Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Summary

- Learning needed for unknown environments, lazy designers
- Learning agent = performance element + learning element + critic
- For supervised learning, the aim is to find a simple hypothesis *approximately* consistent with training examples
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set
- Overfitting hampers a machine learners ability to generalize to unseen examples