Chapter 6

Machine learning

Basanta Joshi, PhD
basanta@ioe.edu.np

Lecture notes can be downloaded from
www.basantajoshi.com.np

Introduction to Learning

What is Learning?

- Learning is one of those everyday terms which is broadly and vaguely used in the English language
  - **Learning is making useful changes in our minds**
  - **Learning is constructing or modifying representations of what is being experienced**
  - **Learning is the phenomenon of knowledge acquisition in the absence of explicit programming**

Herbert Simon, 1983

Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively next time.

Implications

- Learning involves 3 factors:

changes

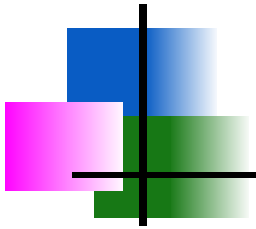
generalization

improvement

Learning changes the learner: for machine learning the problem is determining the nature of these changes and how to best represent them

Learning leads to generalization: performance must improve not only on the same task but on similar tasks

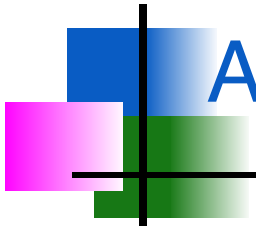
Learning leads to improvements: machine learning must address the possibility that changes may degrade performance and find ways to prevent it.



Why Study Machine Learning?

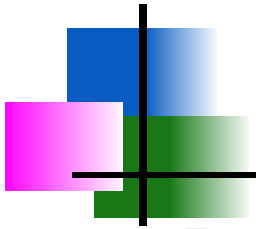
- Many tasks would benefit from adaptive systems:
  - Robot exploring Mars (or cleaning your house!)
  - Software agents (OS functions, web searching)
  - Speech, vision, language, …

- Easier to build a learning system than to hand code a program of similar performance



Areas of Influence for Machine Learning

- *Statistics:* How best to use samples drawn from unknown probability distributions to help decide from which distribution some new sample is drawn?
- *Brain Models:* Non-linear elements with weighted inputs (Artificial Neural Networks) have been suggested as simple models of biological neurons.
- *Adaptive Control Theory:* How to deal with controlling a process having unknown parameters that must be estimated during operation?



Growth in Machine Learning

- Recent Progress in algorithms and theory
- Growing flood of online data
- Computational power is available
- Budding Industry
- Three Niches
  - Data Mining: using historical data to improve decisions
  - SW Apps we cannot program by hand
  - Self customizing programs

Example: Spam filter

- Input: email
- Output: spam/ham
- Setup:
  - Get a large collection of example emails, each labeled "spam" or "ham"
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future emails

- Features: The attributes used to make the ham / spam decision
  - Words: FREE!
  - Text Patterns: $dd, CAPS
  - Non-text: SenderInContacts
  - …

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Example: Digit Recognition

- Input: images / pixel grids
- Output: a digit 0-9
- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images

- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops

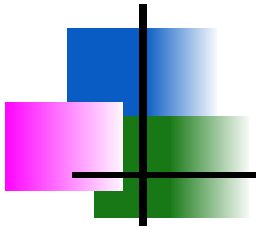

0


1


2


1


??



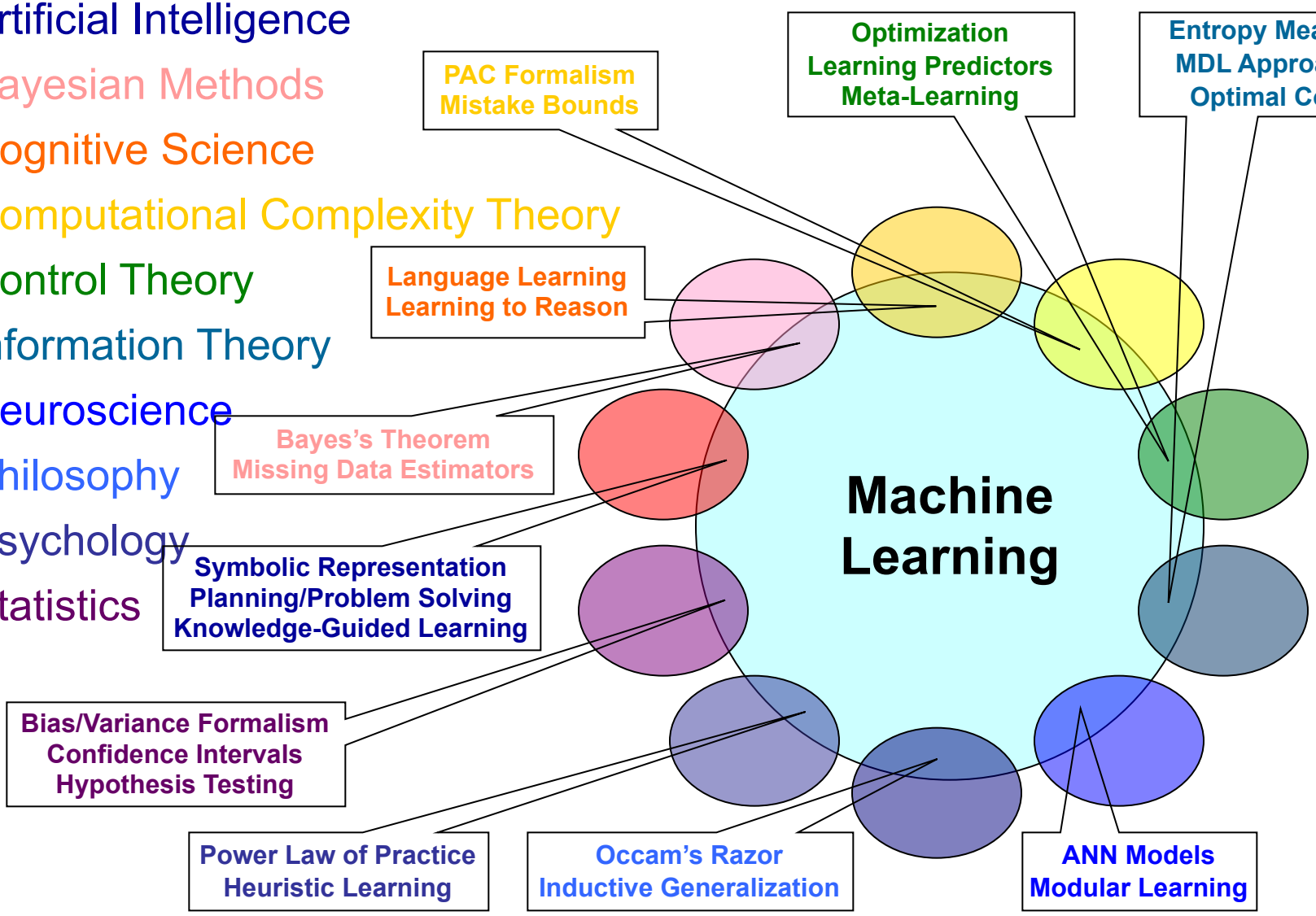
Other Classification Tasks

In classification, we predict labels y (classes) for inputs x

- Examples:
  - Spam detection (input: document, classes: spam / ham)
  - OCR (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grader (input: document, classes: grades)
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - Customer service email routing
  - … many more

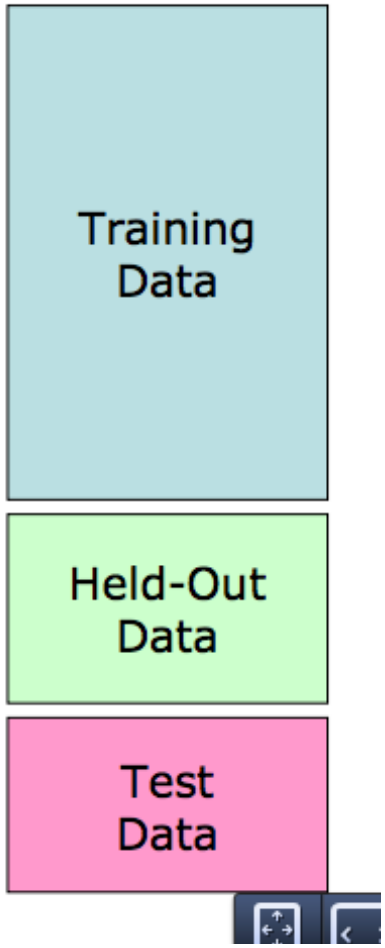
- Classification is an important commercial technology!

- Artificial Intelligence
- Bayesian Methods
- Cognitive Science
- Computational Complexity Theory
- Control Theory
- Information Theory
- Neuroscience
- Philosophy
- Psychology
- Statistics

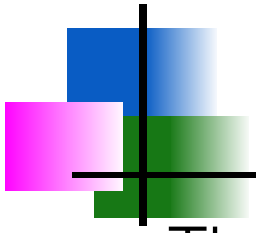


Important Concepts

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- Features: attribute-value pairs which characterize each x
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of test set
  - Very important: never "peek" at the test set!
- Evaluation
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well

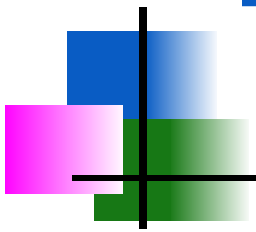


Learning Methods



- There are two different kinds of information processing which must be considered in a machine learning system
  - *Inductive learning* is concerned with determining general patterns, organizational schemes, rules, and laws from raw data, experience or examples.
  - *Deductive learning* is concerned with determination of specific facts using general rules or the determination of new general rules from old general rules.

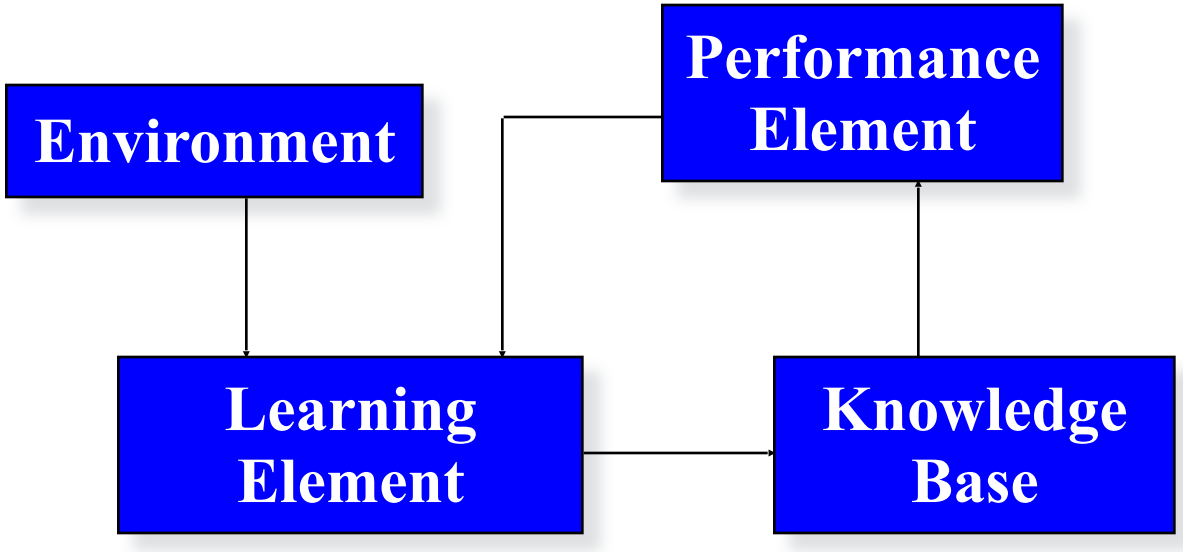
Different kinds of learning…

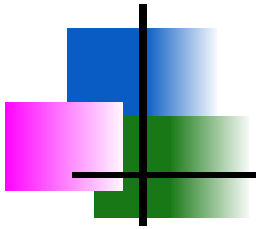


- Supervised learning:
  - Someone gives us examples and the right answer for those examples
  - We have to predict the right answer for unseen examples
- Unsupervised learning:
  - We see examples but get no feedback
  - We need to find patterns in the data
- Reinforcement learning:
  - We take actions and get rewards
  - Have to learn how to get high rewards

Learning Framework

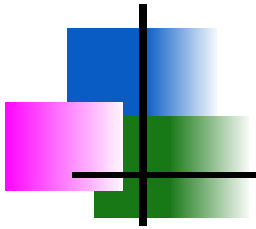
- There are four major components in a learning system:





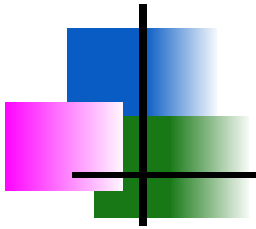
The Environment

- The environment refers the nature and quality of information given to the learning element
- The nature of information depends on its level (the degree of generality wrt the performance element)
  - high level information is abstract, it deals with a broad class of problems
  - low level information is detailed, it deals with a single problem.
- The quality of information involves
  - noise free
  - reliable
  - ordered



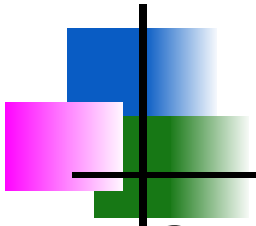
Learning Elements

- Four learning situations
  - Rote Learning
    - environment provides information at the required level
  - Learning by being told
    - information is too abstract, the learning element must hypothesize missing data
  - Learning by example
    - information is too specific, the learning element must hypothesize more general rules
  - Learning by analogy
    - information provided is relevant only to an analogous task, the learning element must discover the analogy



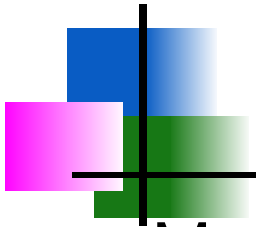
The Knowledge Base

- Expressive
  - the representation contains the relevant knowledge in an easy to get to fashion
- Modifiable
  - it must be easy to change the data in the knowledge base
- Extendibility
  - the knowledge base must contain meta-knowledge (knowledge on how the data base is structured) so the system can change its structure



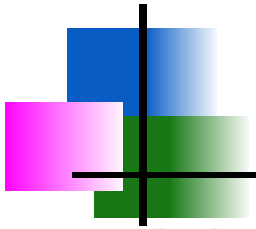
The Performance Element

- Complexity
  - for learning, the simplest task is classification based on a single rule while the most complex task requires the application of multiple rules in sequence
- Feedback
  - the performance element must send information to the learning system to be used to evaluate the overall performance
- Transparency
  - the learning element should have access to all the internal actions of the performance element



Rote Learning

- Memorization - saving new knowledge to be retrieved when needed rather than calculated
- Works by taking problems that the performance element has solved and memorizing the problem and the solution
- Only useful if it takes less time to retrieve the knowledge than it does to recompute it

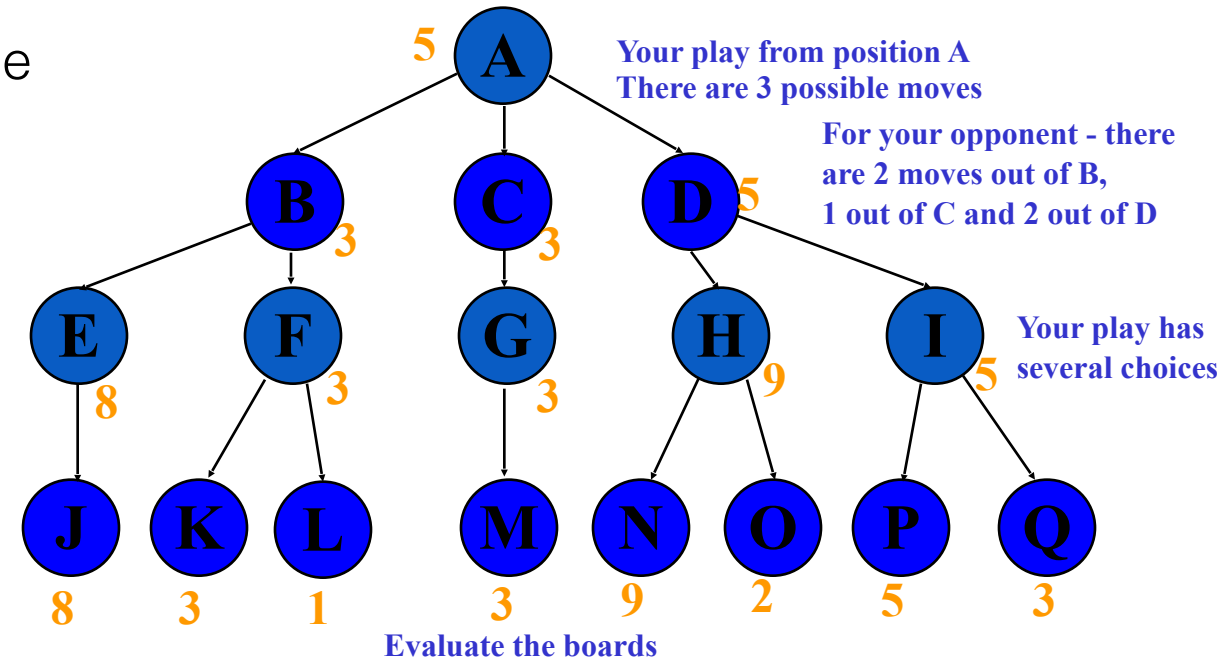


Checkers Example

- A.L. Samuels Checkers Player (1959-1967)
- The program knows and follows the rules of checkers
- It memorizes and recalls board positions it has encountered in previous games

MiniMax Game Tree

- Checkers used a minimax game tree with a 3 move lookahead and a static evaluation function:



Move the maximum values up when it is your move

Move the minimum values up when it is your opponents move

Finally, move the maximum up on your move

Learning Process

- The checkers program would memorize a board position such as A and its value 5 - [A,5].

- The next time the system ran into A , it used the memorized value of 5 rather than compute the static evaluation function

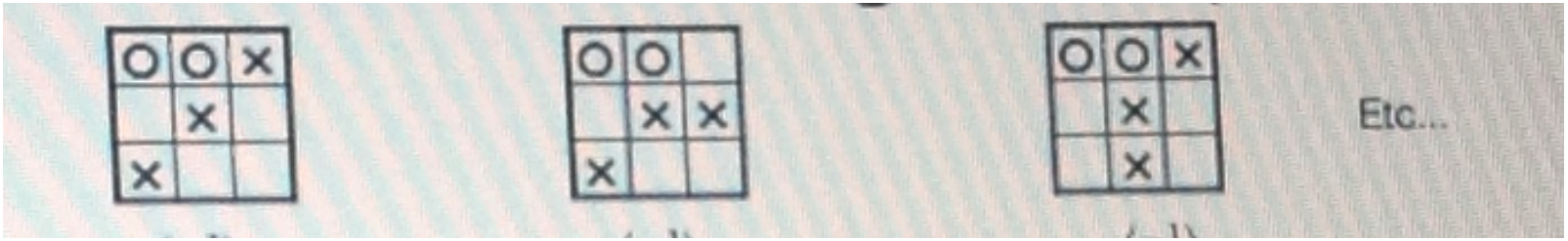
- *Result*: a lookahead of 5, 7 . . .

Learning by Induction

Learning from Examples: Induction

- Classification is the process of assigning, to a particular input, the name of a class to which it belongs.
- The classes from which the classification procedure can choose can be described in a variety of ways.
- Their definition will depend on the use to which they are put.
- Classification is an important component of many problem solving tasks.
- Before classification can be done, the classes it will use must be defined:
  - Isolate a set of features that are relevant to the task domain.Define each class by a weighted sum of values of these features. Ex: task is weather prediction, the parameters can be measurements such as rainfall, location of cold fronts etc.
  - Isolate a set of features that are relevant to the task domain. Define each class as a structure composed of these features. Ex: classifying animals, various features can be such things as color,length of neck etc
- The idea of producing a classification program that can evolve its own class definitions is called concept learning or induction.

Inductive learning - example A

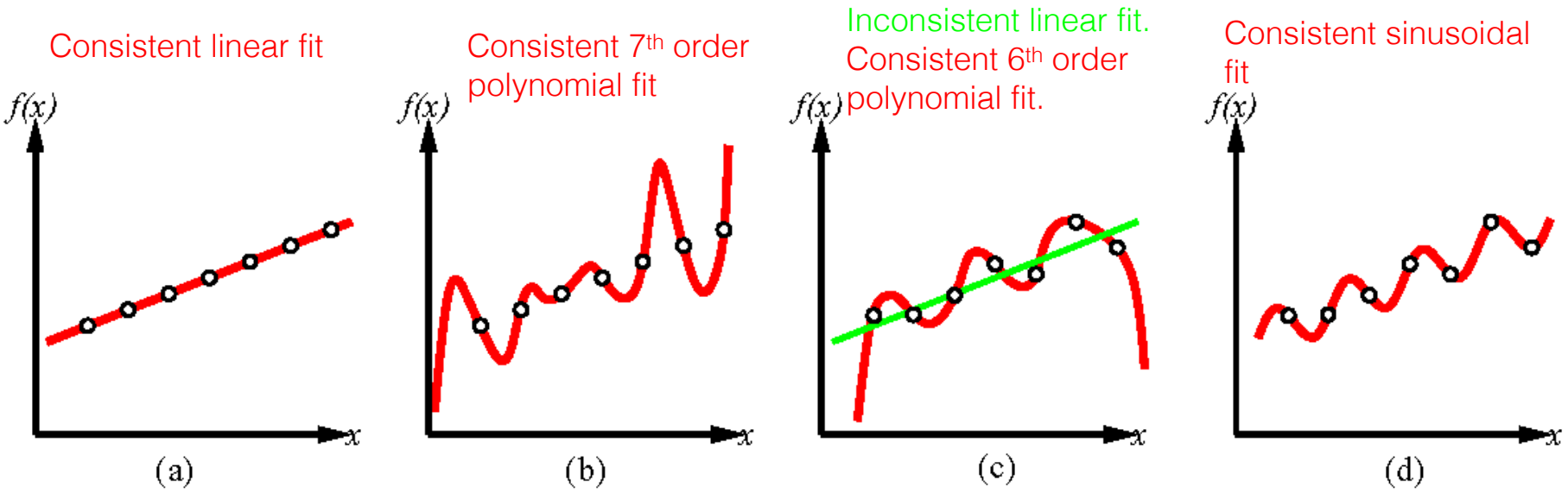


$$\mathbf{x} = \begin{pmatrix} -1 \\ -1 \\ +1 \\ 0 \\ +1 \\ 0 \\ +1 \\ 0 \\ 0 \end{pmatrix}, f(\mathbf{x}) = +1 \qquad \mathbf{x} = \begin{pmatrix} -1 \\ -1 \\ 0 \\ 0 \\ +1 \\ +1 \\ +1 \\ 0 \\ 0 \end{pmatrix}, f(\mathbf{x}) = -1 \qquad \mathbf{x} = \begin{pmatrix} -1 \\ -1 \\ 0 \\ 0 \\ +1 \\ +1 \\ 0 \\ +1 \\ 0 \end{pmatrix}, f(\mathbf{x}) = 0$$

- $f(\mathbf{x})$ is the *target function*
- An *example* is a pair $[\mathbf{x}, f(\mathbf{x})]$
- Learning task: find a *hypothesis* h such that $h(\mathbf{x}) \approx f(\mathbf{x})$ given a training set of examples $D = \{[\mathbf{x}_i, f(\mathbf{x}_i)]\}, i = 1,2,\ldots,N$

Inspired by a slide from V. Pavlovic

Inductive learning – example B



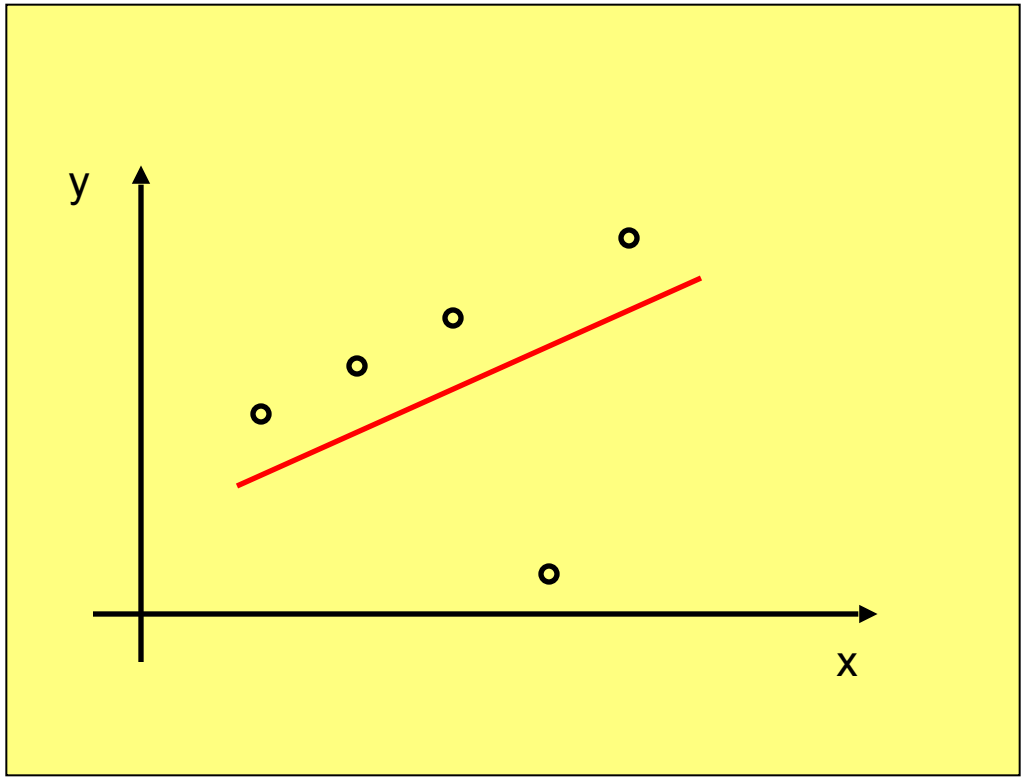
Construct h so that it agrees with f.

The hypothesis h is consistent if it agrees with f on all observations.

Ockham's razor: Select the simplest consistent hypothesis.

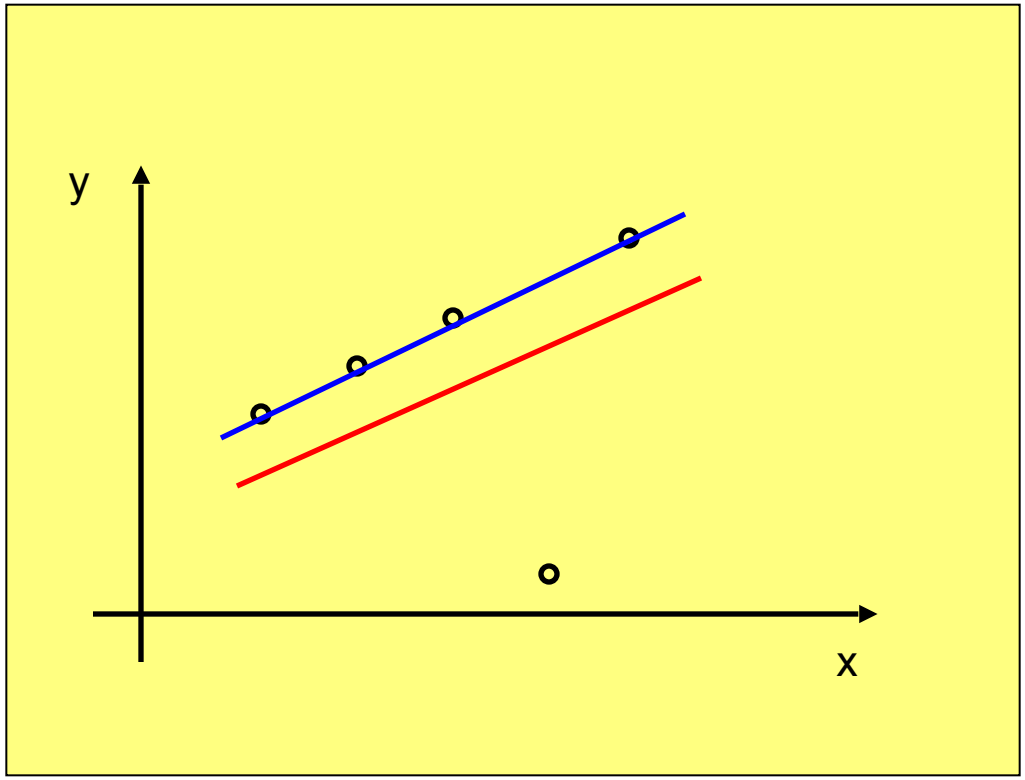
How achieve good generalization?

Inductive learning – example C



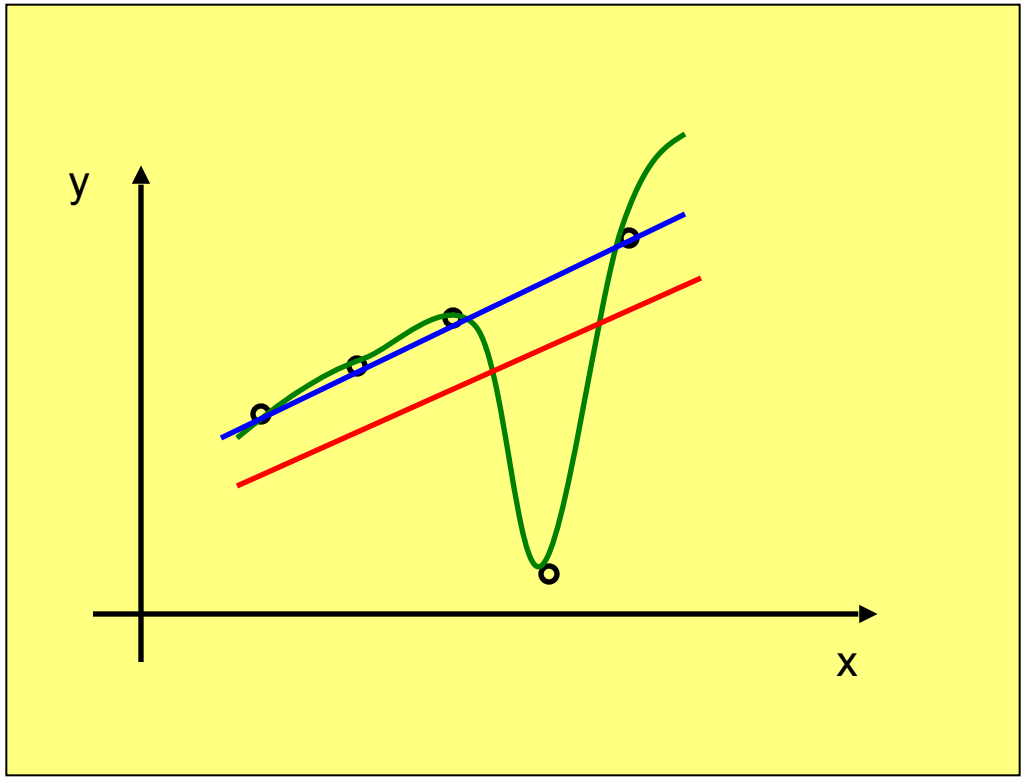
Example from V. Pavlovic @ Rutgers

Inductive learning – example C



Example from V. Pavlovic @ Rutgers

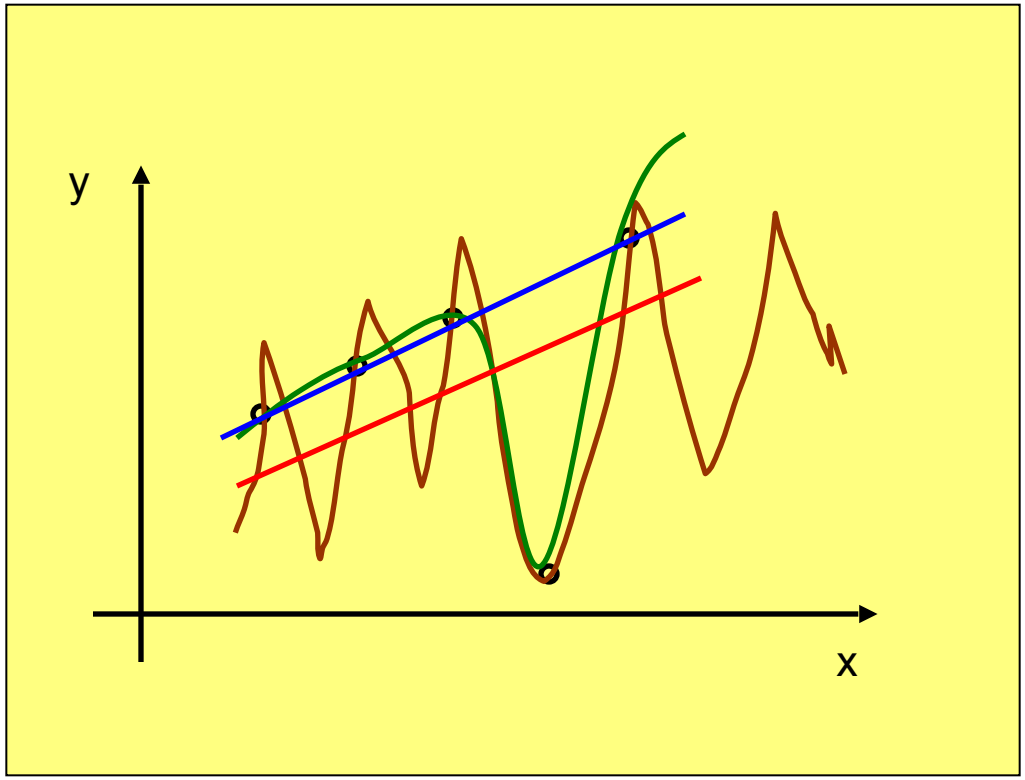
Inductive learning – example C



Example from V. Pavlovic @ Rutgers

Inductive learning – example C

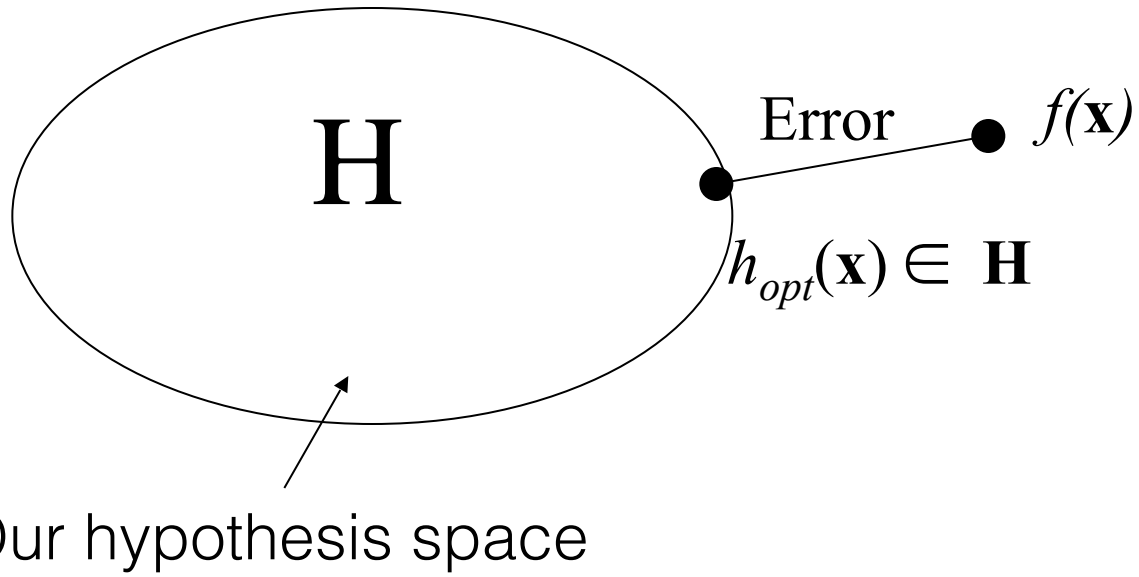
Sometimes a consistent hypothesis is worse than an inconsistent



Example from V. Pavlovic @ Rutgers

The idealized inductive learning problem

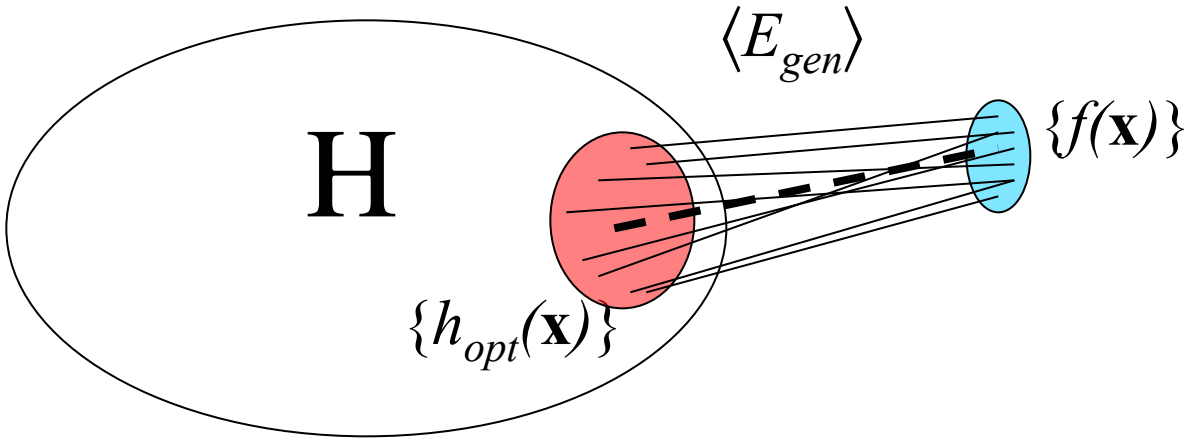
Find appropriate hypothesis space $\mathbf{H}$ and find $h(\mathbf{x}) \in \mathbf{H}$ with minimum "distance" to $f(\mathbf{x})$ ("error")



The learning problem is realizable if $f(\mathbf{x})$ 安 $\mathbf{H}$.

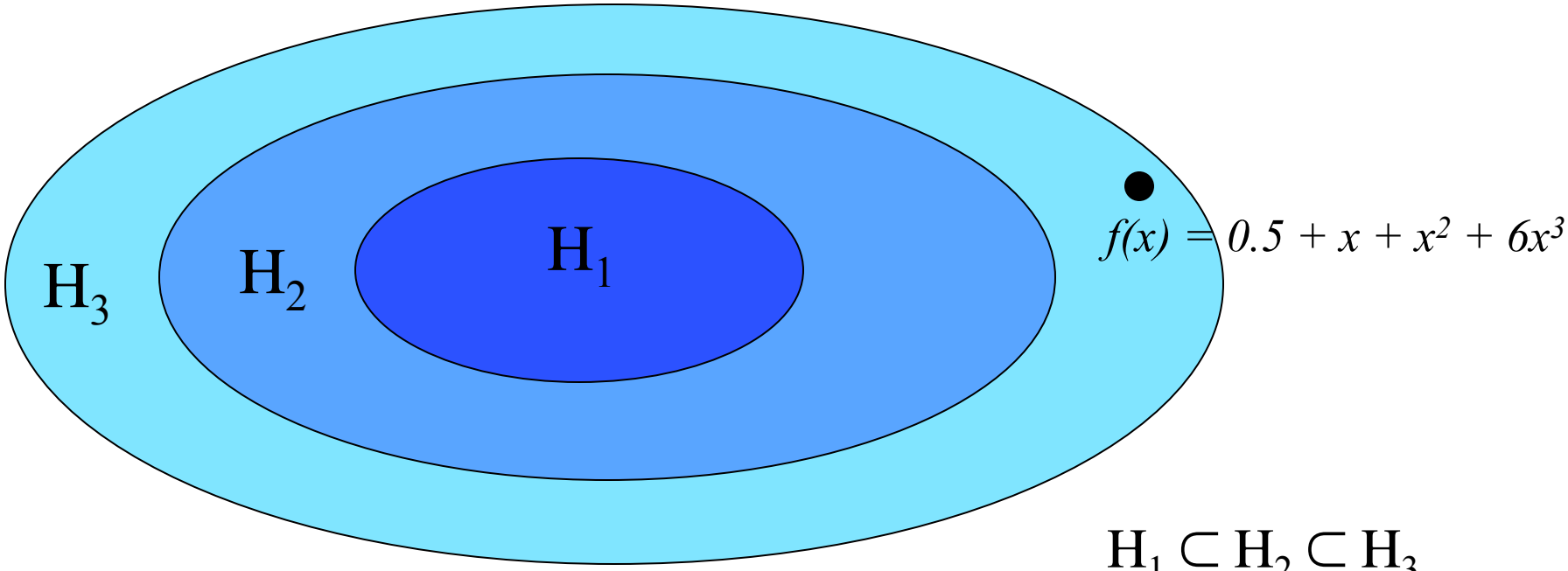
The real inductive learning problem

Find appropriate hypothesis space $\mathbf{H}$ and minimize the <u>expected</u> distance to $f(\mathbf{x})$ ("generalization error")



Data is never noise free and never available in infinite amounts, so we get variation in data and model.
The generalization error is a function of both the training data and the hypothesis selection method.

Hypothesis spaces (examples)

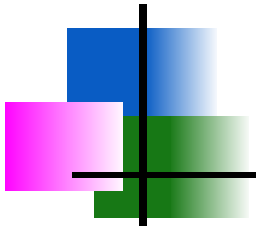


$H_1 \subset H_2 \subset H_3$

$H_1 = \{a + bx\}$; $H_2 = \{a + bx + cx^2\}$; $H_3 = \{a + bx + cx^2 + dx^3\}$;

Linear; Quadratic; Cubic;

Learning problems



- The hypothesis takes as input a set of attributes $\mathbf{x}$ and returns a ”decision” $h(\mathbf{x})$ = the predicted (estimated) output value for the input $\mathbf{x}$.

- Discrete valued function 宅 classification
- Continuous valued function 宅 regression

Classification

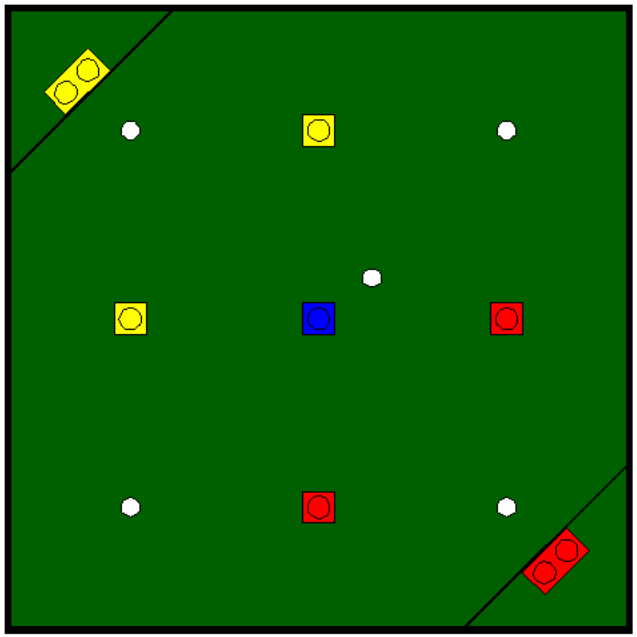
Order into one out of several classes

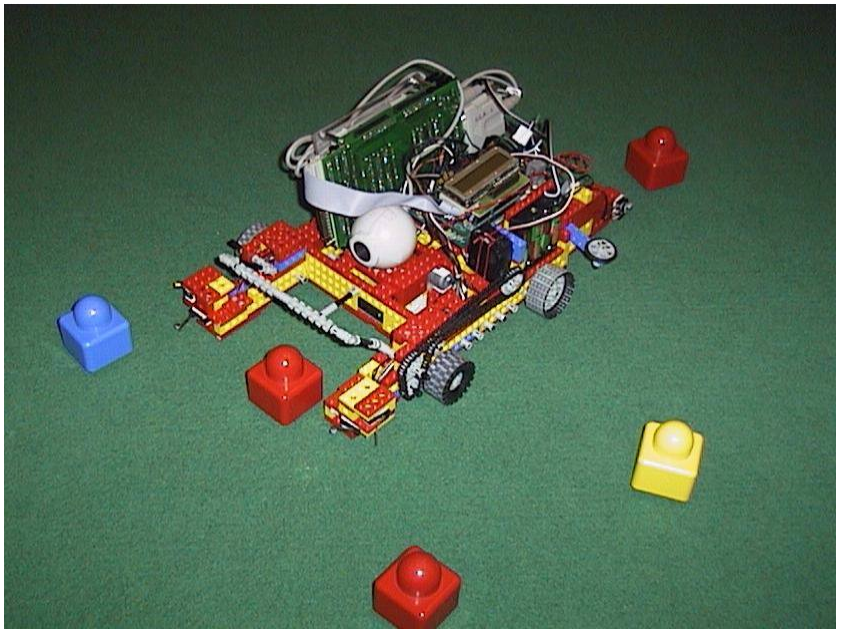
$$X^D \rightarrow C^K$$

Input space Output (category) space

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \mathsf{M} \\ x_D \end{pmatrix} \in X^D \qquad \mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \mathsf{M} \\ c_K \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ \mathsf{M} \\ 0 \end{pmatrix} \in C^K$$

Example: Robot color vision





Classify the Lego pieces into *red*, *blue*, and *yellow.*
Classify *white* balls, *black* sideboard, and *green* carpet.
Input = pixel in image, output = category

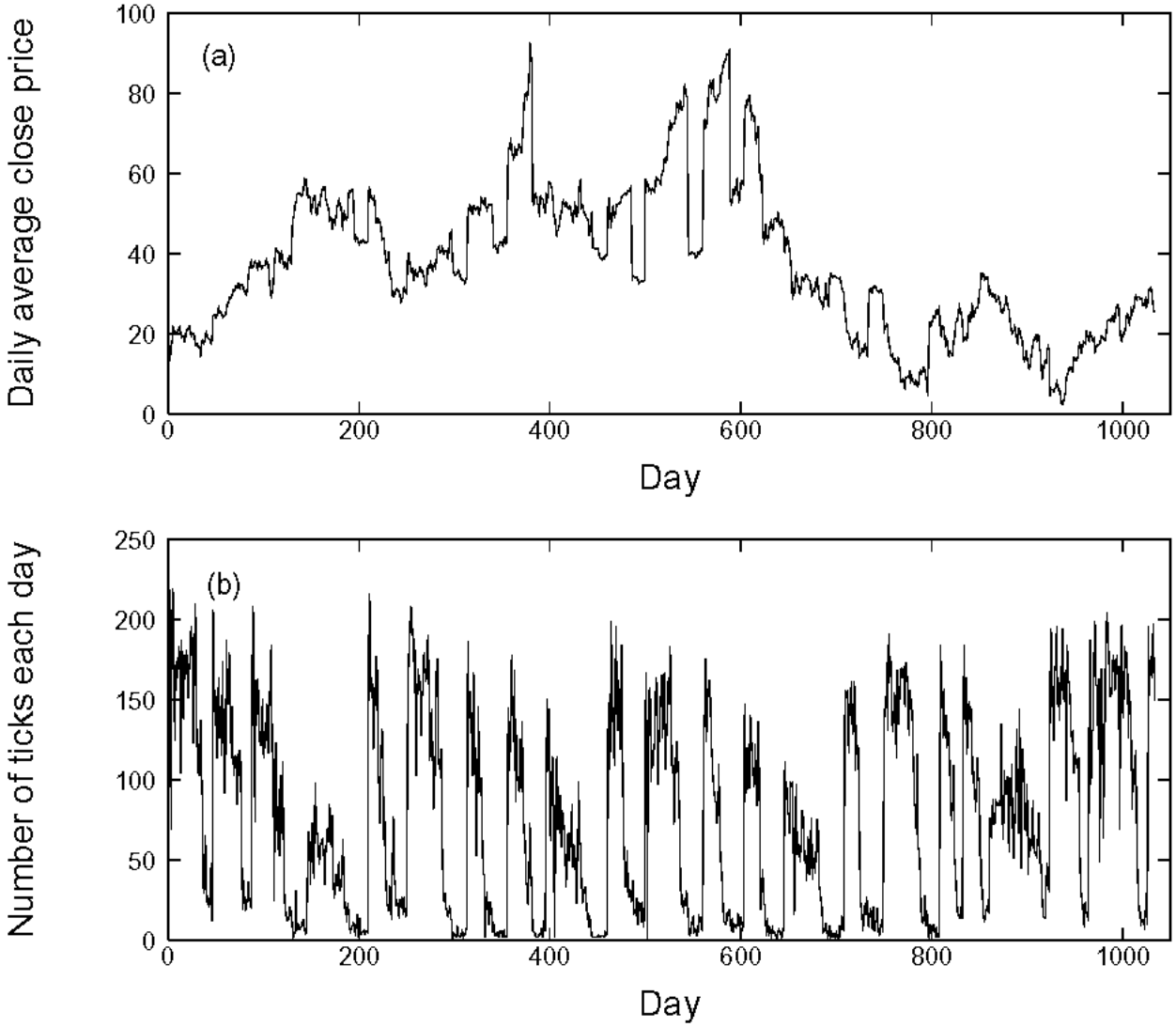
Regression

The “fixed regressor model”

$$f(\mathbf{x}) = g(\mathbf{x}) + \varepsilon$$

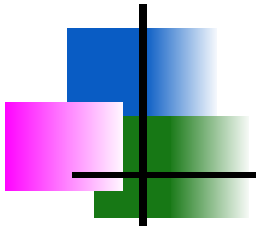
$\mathbf{x}$	Observed input
$f(\mathbf{x})$	Observed output
$g(\mathbf{x})$	True underlying function
ε	I.I.D noise process with zero mean

Example: Predict price for cotton futures



Input: Past history of closing prices, and trading volume

Output: Predicted closing price

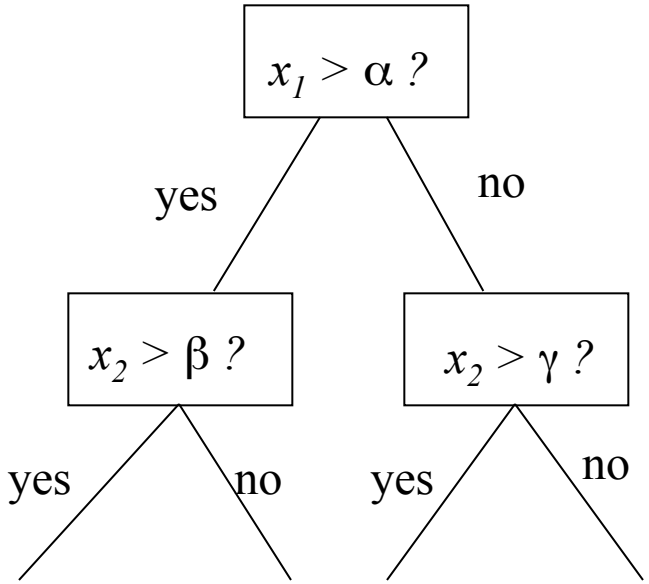


Now...

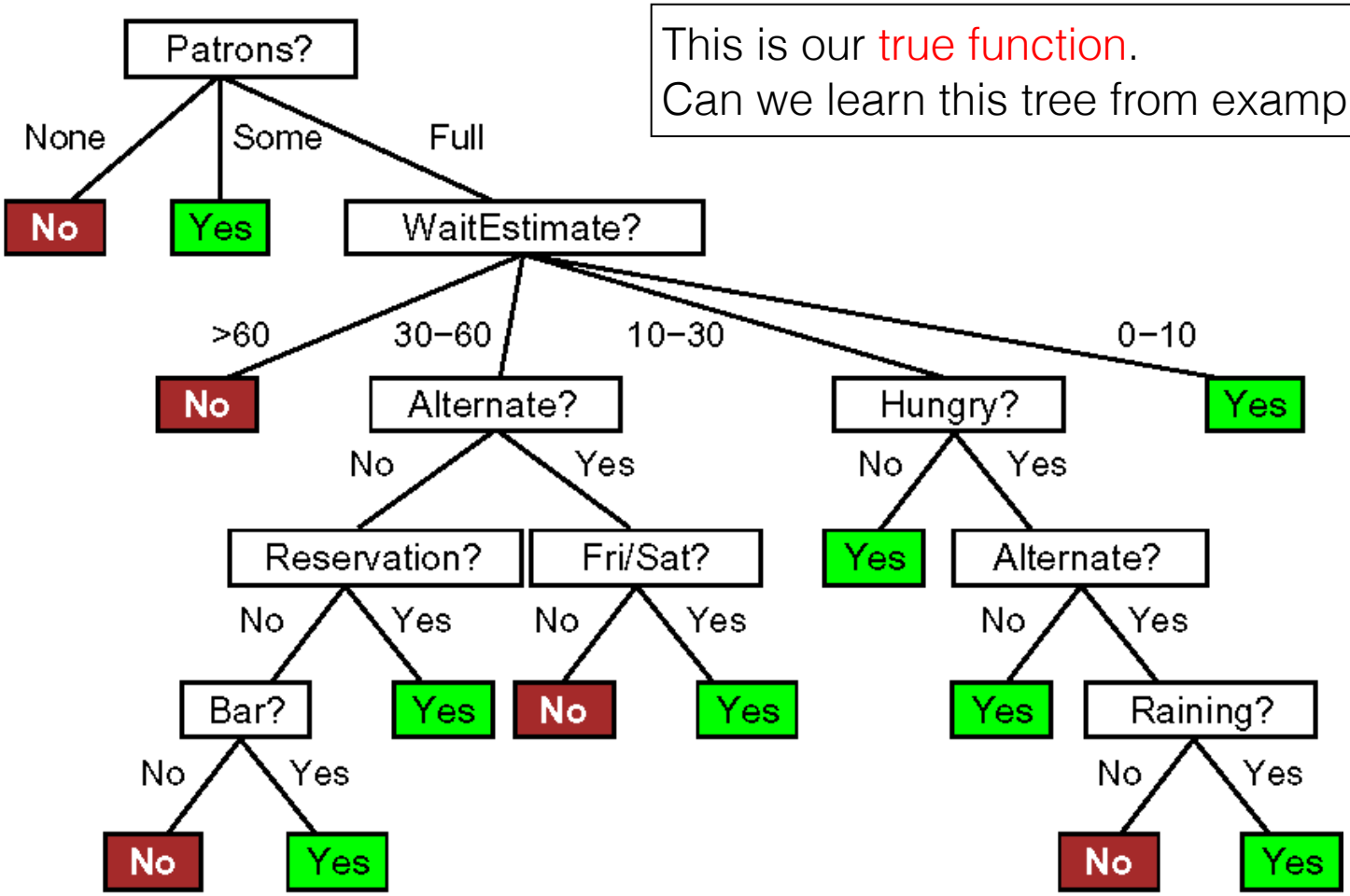
let's look at a classification problem: predicting whether a certain person will choose a particular restaurant.

Method: Decision trees

- "Divide and conquer": Split data into smaller and smaller subsets.
- Splits usually on a single variable

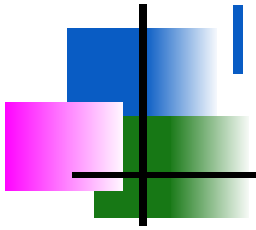


The wait@restaurant decision tree



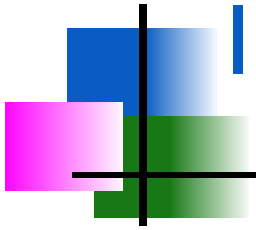
This is our true function.
Can we learn this tree from examples?

Inductive learning of decision tree



- *Simplest:* Construct a decision tree with one leaf for every example = memory based learning.
  Not very good generalization.

Inductive learning of decision tree



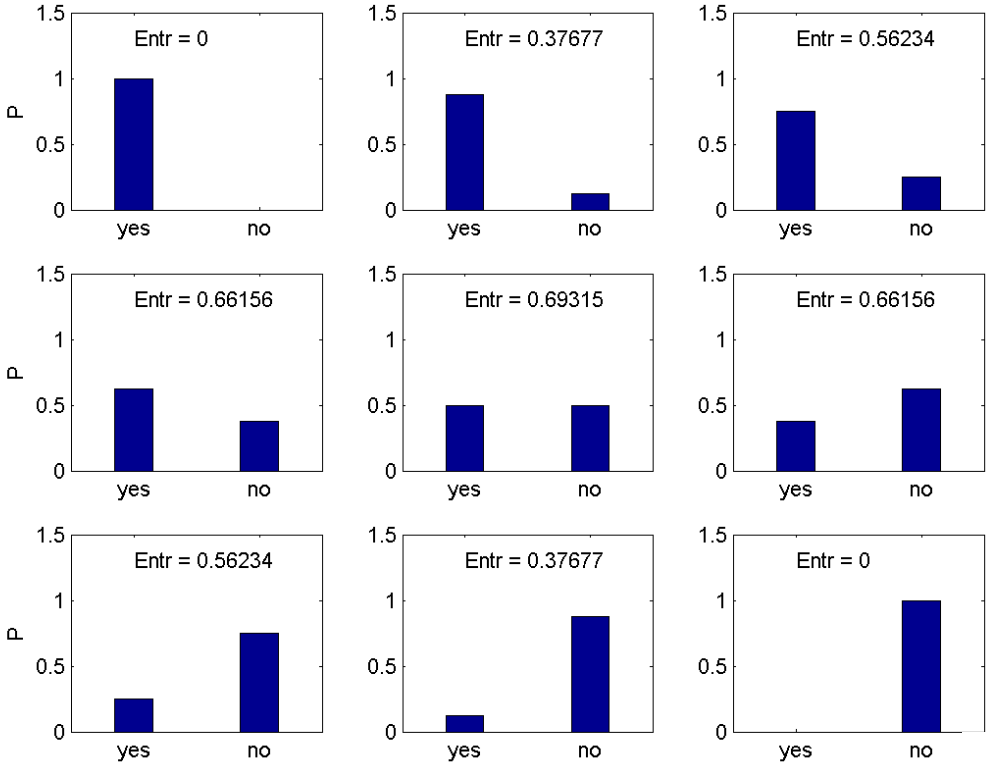
- *Simplest:* Construct a decision tree with one leaf for every example = memory based learning.
  Not very good generalization.
- *Advanced:* Split on each variable so that the purity of each split increases (i.e. either only yes or only no)
- Purity measured e.g. with entropy

Inductive learning of decision tree

- *Simplest:* Construct a decision tree with one leaf for every example = memory based learning.
  Not very good generalization.
- *Advanced:* Split on each variable so that the purity of each split increases (i.e. either only yes or only no)
- Purity measured,e.g, with entropy

$$\text{Entropy} = -P(yes)\ln[P(yes)] - P(no)\ln[P(no)]$$

General form: $$\text{Entropy} = -\sum_i P(v_i)\ln\left[P(v_i)\right]$$



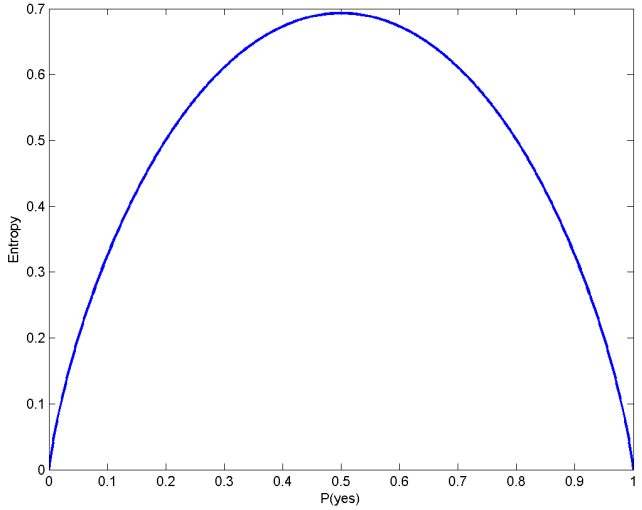
The entropy is maximal when all possibilities are equally likely.

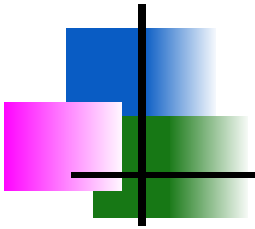
The goal of the decision tree is to decrease the entropy in each node.

Entropy is zero in a pure ”yes” node (or pure ”no” node).

Entropy is a measure of ”order” in a system.

The second law of thermodynamics:
Elements in a closed system tend
to seek their most probable distribution;
in a closed system entropy always increases

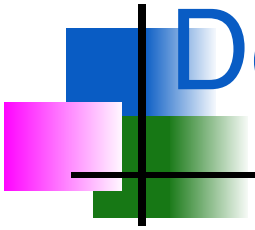




Decision tree learning algorithm

- Create pure nodes whenever possible
- If pure nodes are not possible, choose the split that leads to the largest decrease in entropy.

Decision tree learning example



10 attributes:

1. *Alternate:* Is there a suitable alternative restaurant nearby? {yes,no}
2. *Bar:* Is there a bar to wait in? {yes,no}
3. *Fri/Sat:* Is it Friday or Saturday? {yes,no}
4. *Hungry:* Are you hungry? {yes,no}
5. *Patrons:* How many are seated in the restaurant? {none, some, full}
6. *Price:* Price level {$,$$,$$$}
7. *Raining:* Is it raining? {yes,no}
8. *Reservation:* Did you make a reservation? {yes,no}
9. *Type:* Type of food {French,Italian,Thai,Burger}
10. *Wait:* {0-10 min, 10-30 min, 30-60 min, >60 min}

Decision tree learning example

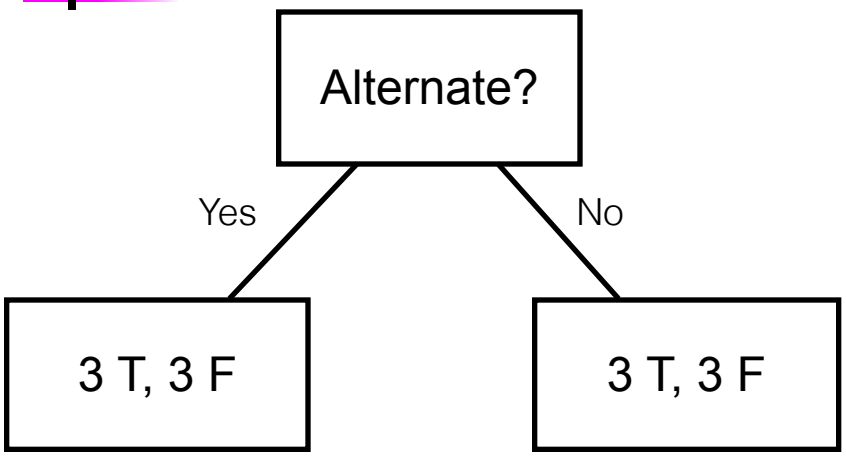
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

T = True, F = False

$$\text{Entropy} = -\left(\tfrac{6}{12}\right)\ln\left(\tfrac{6}{12}\right) - \left(\tfrac{6}{12}\right)\ln\left(\tfrac{6}{12}\right) = 0.30$$

6 True,
6 False

Decision tree learning example

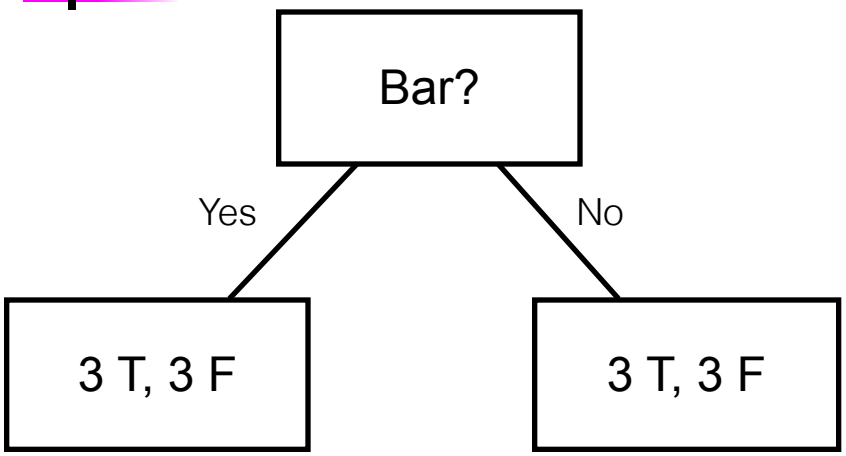


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

$$\text{Entropy} = \frac{6}{12}\left[-\left(\tfrac{3}{6}\right)\log_2\left(\tfrac{3}{6}\right)-\left(\tfrac{3}{6}\right)\log_2\left(\tfrac{3}{6}\right)\right]+\frac{6}{12}\left[-\left(\tfrac{3}{6}\right)\log_2\left(\tfrac{3}{6}\right)-\left(\tfrac{3}{6}\right)\log_2\left(\tfrac{3}{6}\right)\right]=0.30$$

Entropy decrease = 0.30 – 0.30 = 0

Decision tree learning example

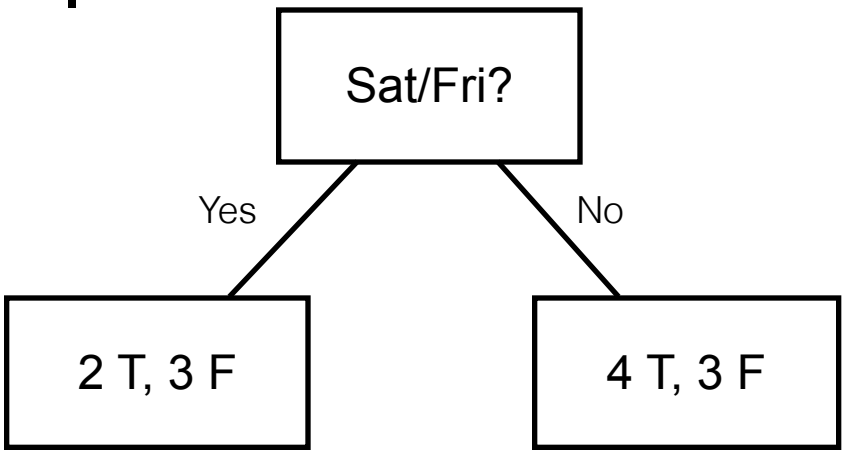


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

$$\text{Entropy} = \frac{6}{12}\left[-\left(\tfrac{3}{6}\right)\ln\left(\tfrac{3}{6}\right)-\left(\tfrac{3}{6}\right)\ln\left(\tfrac{3}{6}\right)\right]+\frac{6}{12}\left[-\left(\tfrac{3}{6}\right)\ln\left(\tfrac{3}{6}\right)-\left(\tfrac{3}{6}\right)\ln\left(\tfrac{3}{6}\right)\right]=0.30$$

Entropy decrease = 0.30 – 0.30 = 0

Decision tree learning example

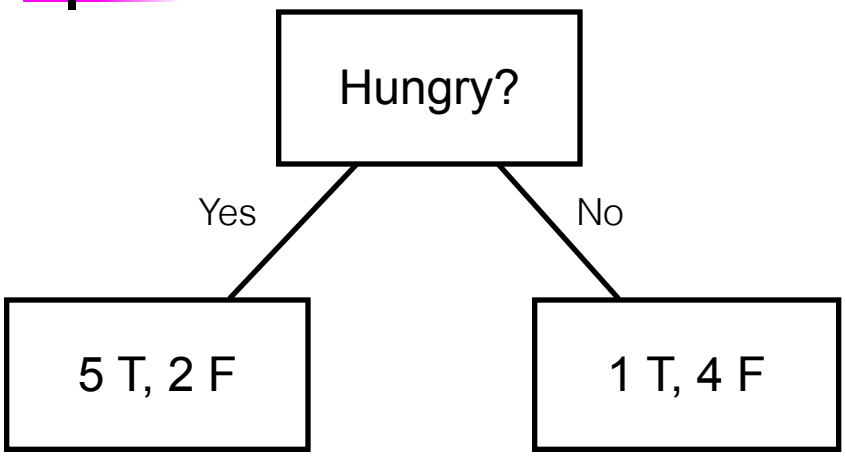


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

$$\text{Entropy} = \frac{5}{12}\left[-\left(\tfrac{2}{5}\right)\ln\left(\tfrac{2}{5}\right)-\left(\tfrac{3}{5}\right)\ln\left(\tfrac{3}{5}\right)\right]+\frac{7}{12}\left[-\left(\tfrac{4}{7}\right)\ln\left(\tfrac{4}{7}\right)-\left(\tfrac{3}{7}\right)\ln\left(\tfrac{3}{7}\right)\right]=0.29$$

Entropy decrease = 0.30 – 0.29 = 0.01

Decision tree learning example

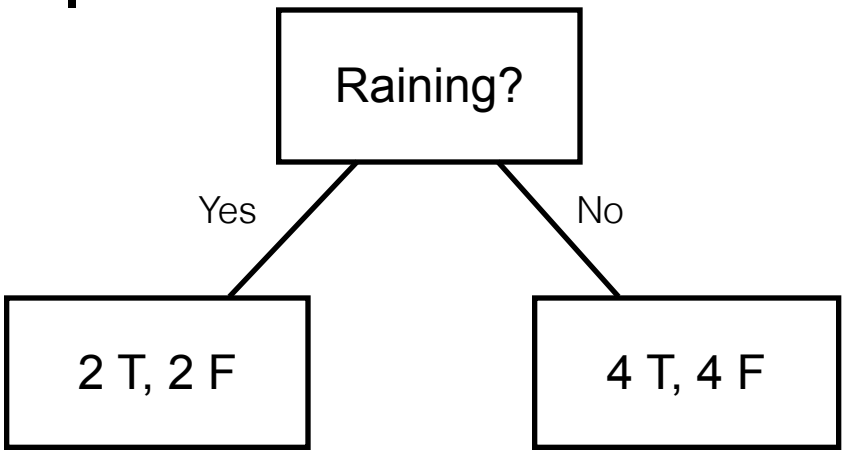


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

$$\text{Entropy} = \frac{7}{12}\left[-\left(\tfrac{5}{7}\right)\ln\left(\tfrac{5}{7}\right)-\left(\tfrac{2}{7}\right)\ln\left(\tfrac{2}{7}\right)\right]+\frac{5}{12}\left[-\left(\tfrac{1}{5}\right)\ln\left(\tfrac{1}{5}\right)-\left(\tfrac{4}{5}\right)\ln\left(\tfrac{4}{5}\right)\right]=0.24$$

Entropy decrease = 0.30 – 0.24 = 0.06

Decision tree learning example

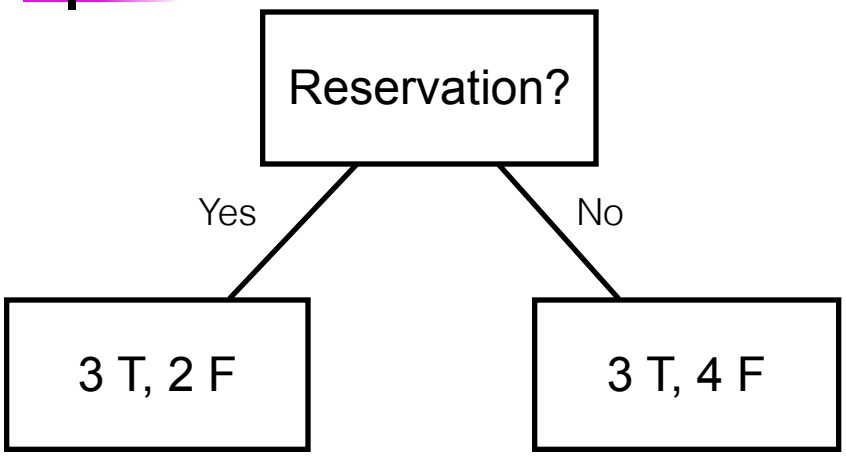


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

$$\text{Entropy} = \frac{4}{12}\left[-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)\right]+\frac{8}{12}\left[-\left(\tfrac{4}{8}\right)\ln\left(\tfrac{4}{8}\right)-\left(\tfrac{4}{8}\right)\ln\left(\tfrac{4}{8}\right)\right]=0.30$$

Entropy decrease = 0.30 – 0.30 = 0

Decision tree learning example

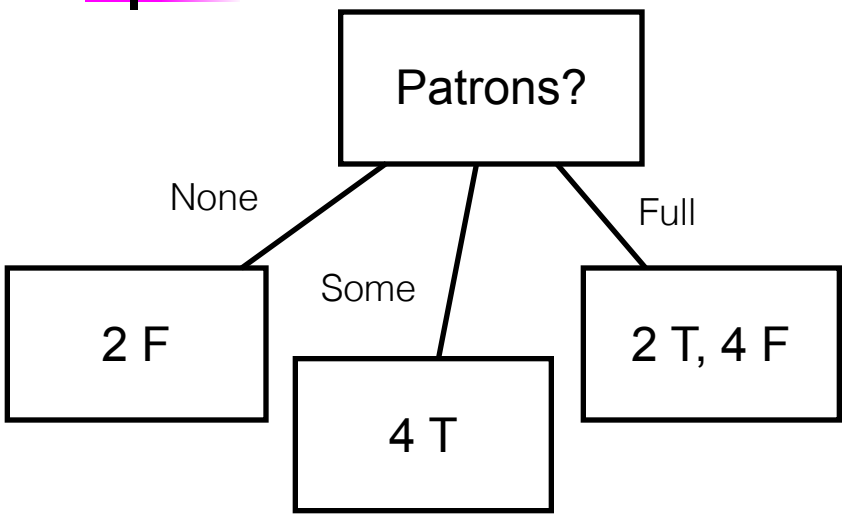


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

$$\text{Entropy} = \frac{5}{12}\left[-\left(\tfrac{3}{5}\right)\ln\left(\tfrac{3}{5}\right)-\left(\tfrac{2}{5}\right)\ln\left(\tfrac{2}{5}\right)\right]+\frac{7}{12}\left[-\left(\tfrac{3}{7}\right)\ln\left(\tfrac{3}{7}\right)-\left(\tfrac{4}{7}\right)\ln\left(\tfrac{4}{7}\right)\right]=0.29$$

Entropy decrease = 0.30 – 0.29 = 0.01

Decision tree learning example



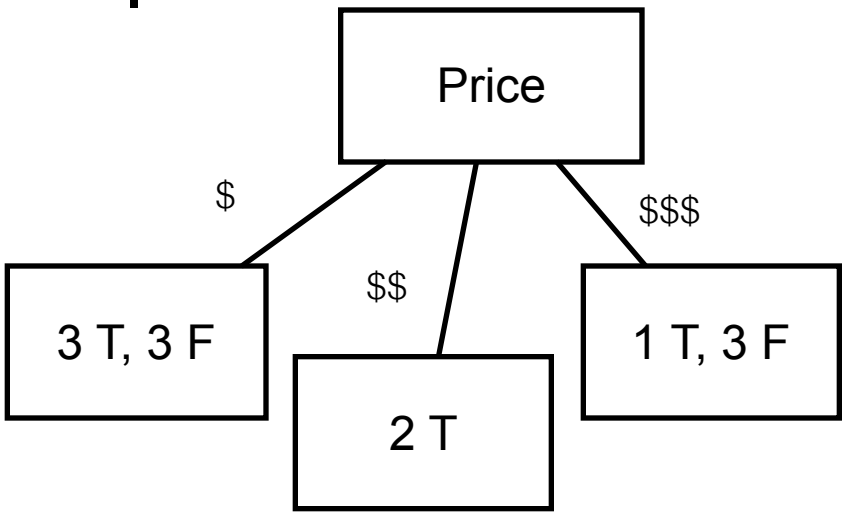
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

$$\text{Entropy} = \frac{2}{12}\left[-\left(\tfrac{0}{2}\right)\ln\left(\tfrac{0}{2}\right)-\left(\tfrac{2}{2}\right)\ln\left(\tfrac{2}{2}\right)\right]+\frac{4}{12}\left[-\left(\tfrac{4}{4}\right)\ln\left(\tfrac{4}{4}\right)-\left(\tfrac{0}{4}\right)\ln\left(\tfrac{0}{4}\right)\right]$$

$$+\frac{6}{12}\left[-\left(\tfrac{2}{6}\right)\ln\left(\tfrac{2}{6}\right)-\left(\tfrac{4}{6}\right)\ln\left(\tfrac{4}{6}\right)\right]=0.14$$

Entropy decrease = 0.30 – 0.14 = 0.16

Decision tree learning example



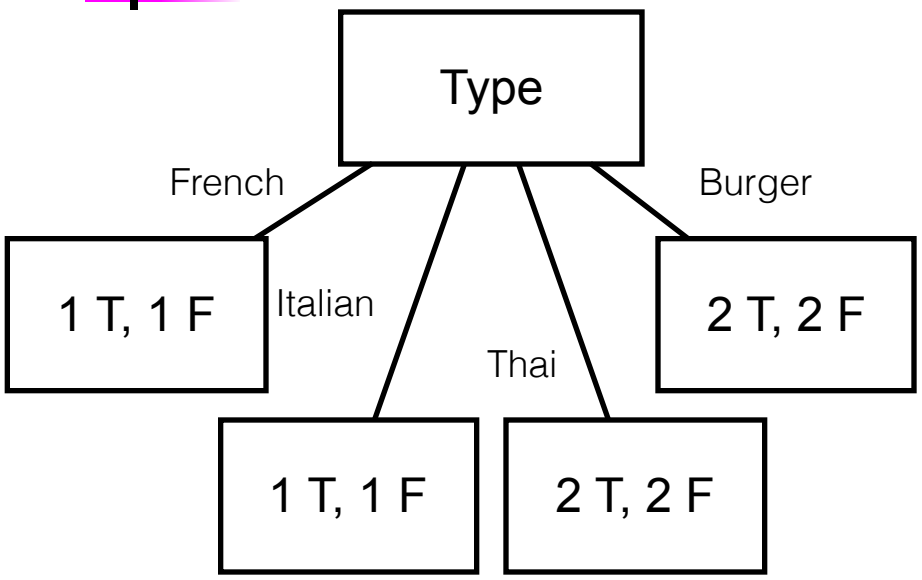
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

$$\text{Entropy} = \frac{6}{12}\left[-\left(\tfrac{3}{6}\right)\ln\left(\tfrac{3}{6}\right)-\left(\tfrac{3}{6}\right)\ln\left(\tfrac{3}{6}\right)\right]+\frac{2}{12}\left[-\left(\tfrac{2}{2}\right)\ln\left(\tfrac{2}{2}\right)-\left(\tfrac{0}{2}\right)\ln\left(\tfrac{0}{2}\right)\right]$$

$$+\frac{4}{12}\left[-\left(\tfrac{1}{4}\right)\ln\left(\tfrac{1}{4}\right)-\left(\tfrac{3}{4}\right)\ln\left(\tfrac{3}{4}\right)\right]=0.23$$

Entropy decrease = 0.30 – 0.23 = 0.07

Decision tree learning example



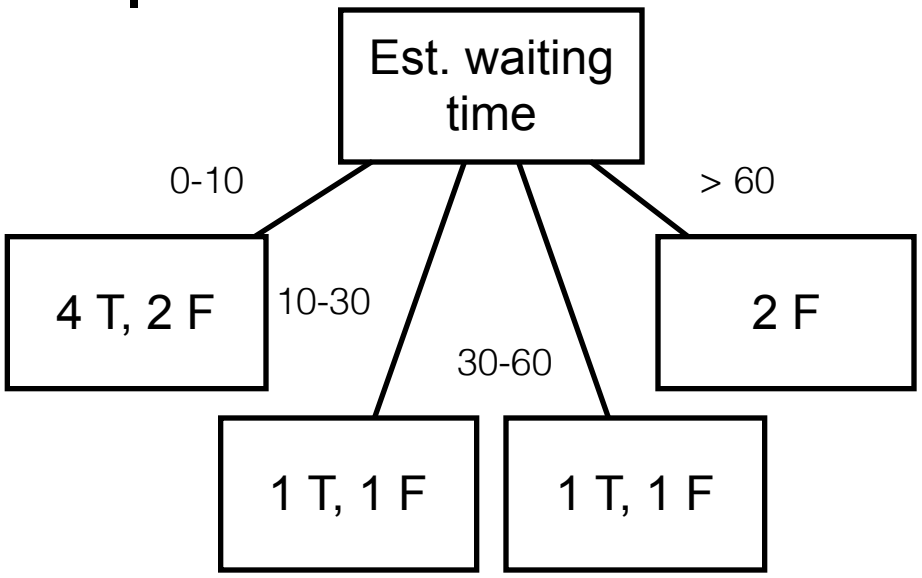
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

$$\text{Entropy} = \frac{2}{12}\left[-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)\right]+\frac{2}{12}\left[-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)\right]$$

$$+\frac{4}{12}\left[-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)\right]+\frac{4}{12}\left[-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)\right]=0.30$$

Entropy decrease = 0.30 – 0.30 = 0

Decision tree learning example



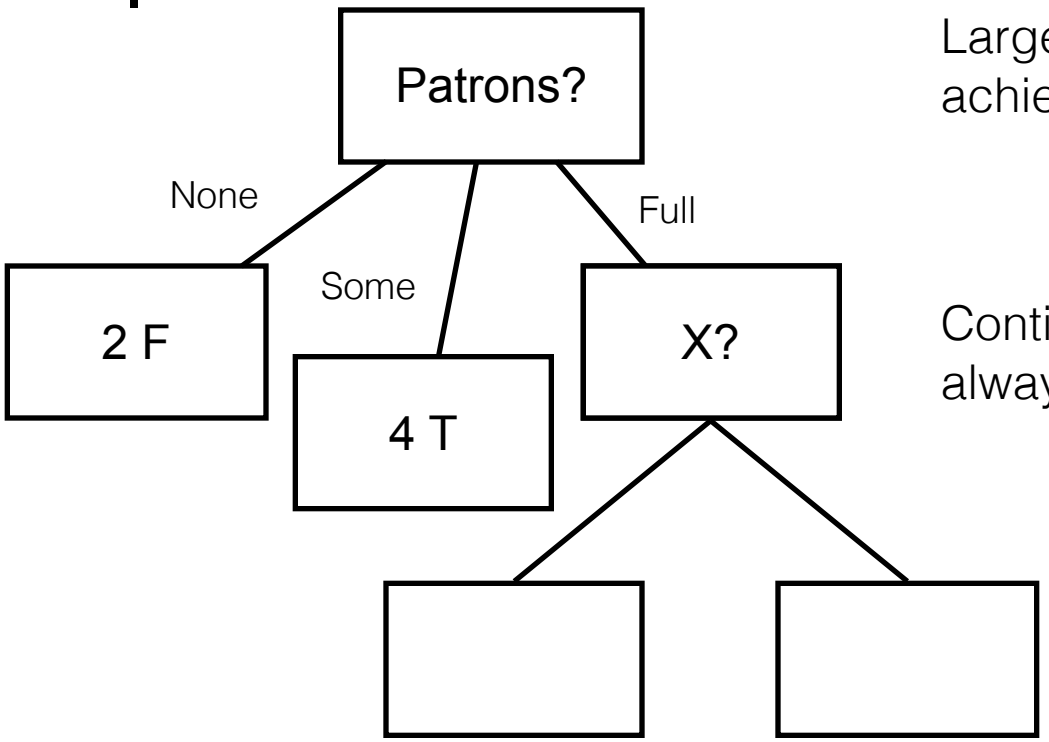
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

$$\text{Entropy} = \frac{6}{12}\left[-\left(\tfrac{4}{6}\right)\log_2\left(\tfrac{4}{6}\right)-\left(\tfrac{2}{6}\right)\log_2\left(\tfrac{2}{6}\right)\right]+\frac{2}{12}\left[-\left(\tfrac{1}{2}\right)\log_2\left(\tfrac{1}{2}\right)-\left(\tfrac{1}{2}\right)\log_2\left(\tfrac{1}{2}\right)\right]$$

$$+\frac{2}{12}\left[-\left(\tfrac{1}{2}\right)\log_2\left(\tfrac{1}{2}\right)-\left(\tfrac{1}{2}\right)\log_2\left(\tfrac{1}{2}\right)\right]+\frac{2}{12}\left[-\left(\tfrac{0}{2}\right)\log_2\left(\tfrac{0}{2}\right)-\left(\tfrac{2}{2}\right)\log_2\left(\tfrac{2}{2}\right)\right]=0.24$$

Entropy decrease = 0.30 – 0.24 = 0.06

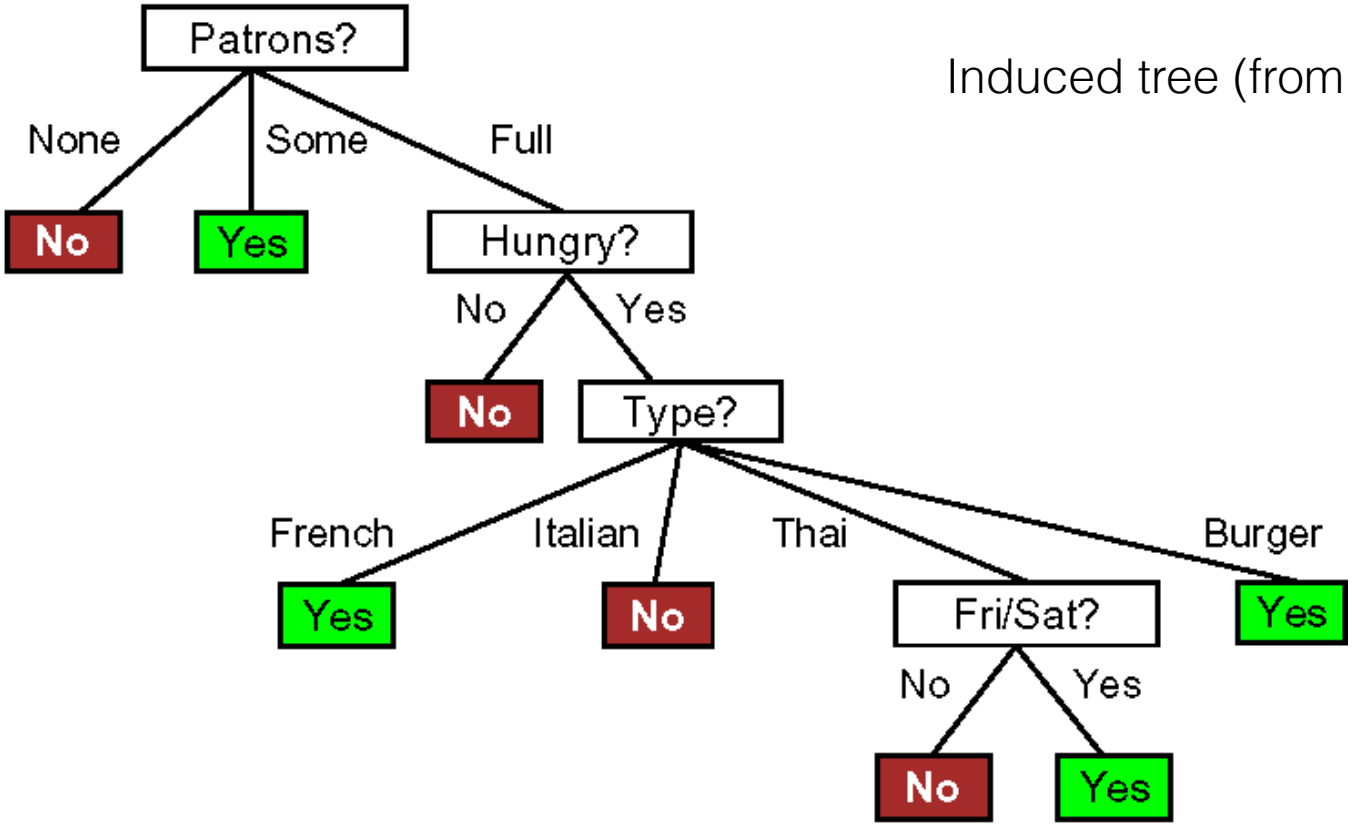
Decision tree learning example



Largest entropy decrease (0.16) achieved by splitting on Patrons.

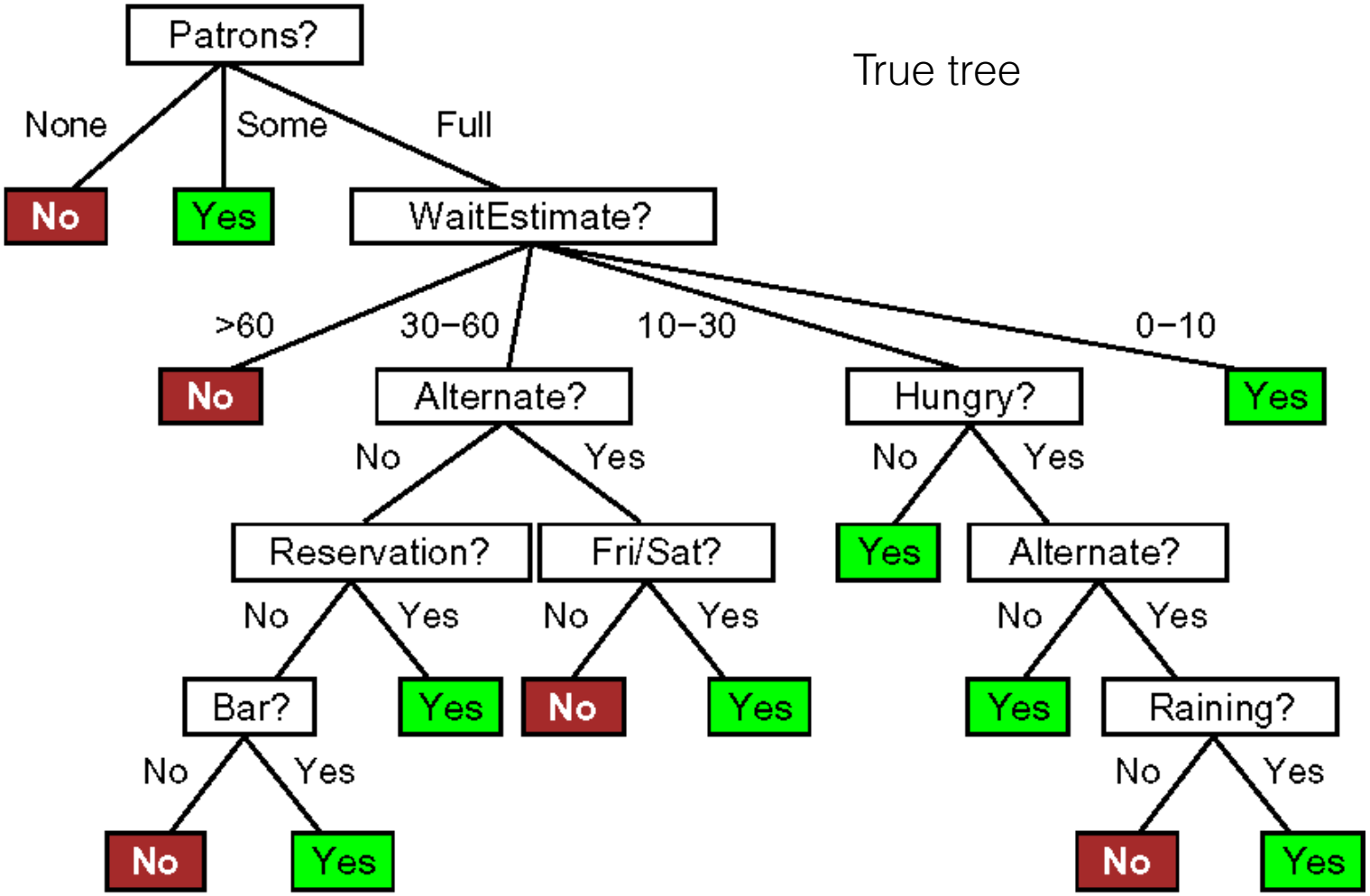
Continue like this, making new splits, always purifying nodes.

Decision tree learning example

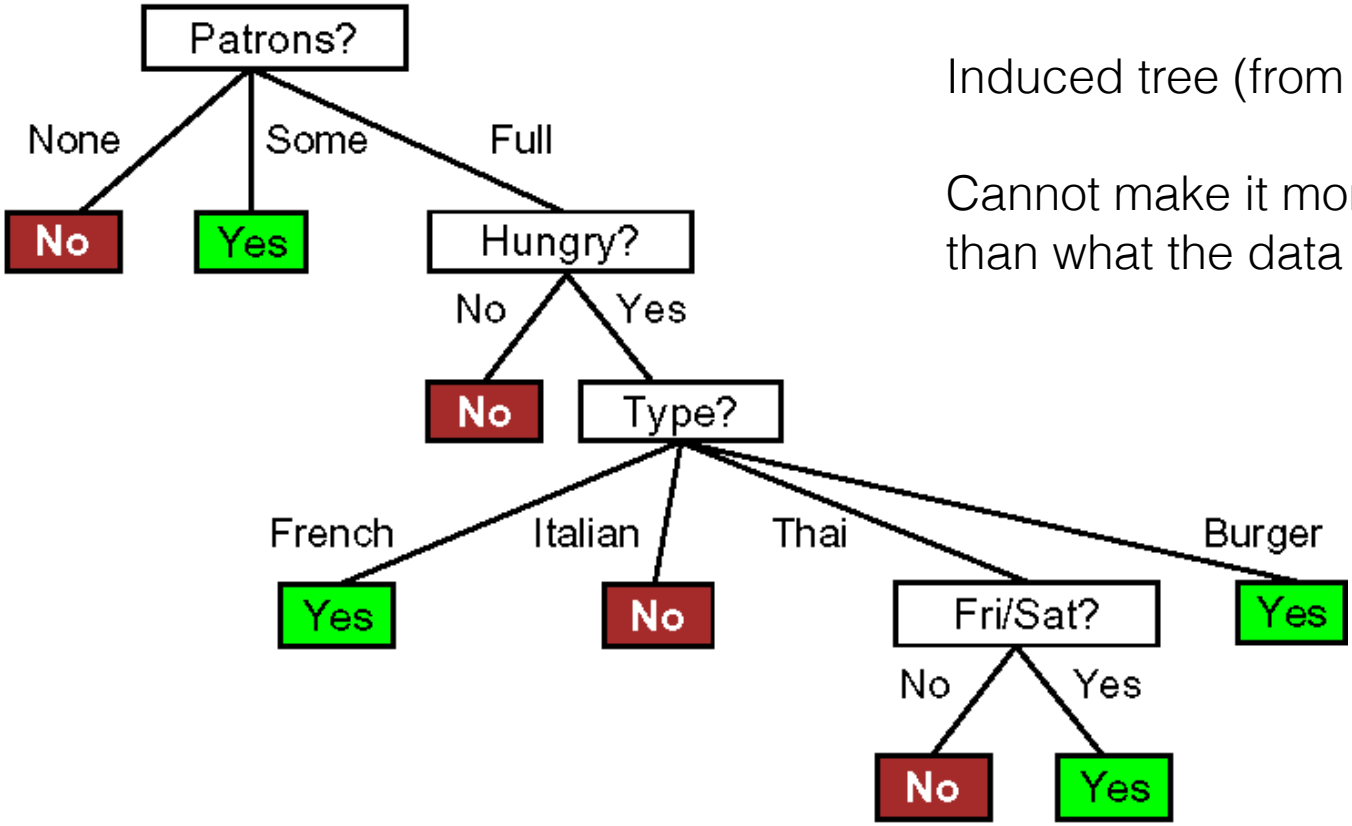


Induced tree (from examples)

Decision tree learning example



Decision tree learning example



Induced tree (from examples)

Cannot make it more complex than what the data supports.

How do we know it is correct?

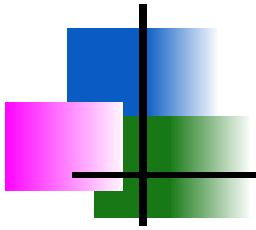
How do we know that $h \approx f$?

(Hume's Problem of Induction)

- Try h on a new test set of examples (cross validation)

...and assume the ”principle of uniformity”, i.e. the result we get on this test data should be indicative of results on future data. Causality is constant.

Inspired by a slide by V. Pavlovic

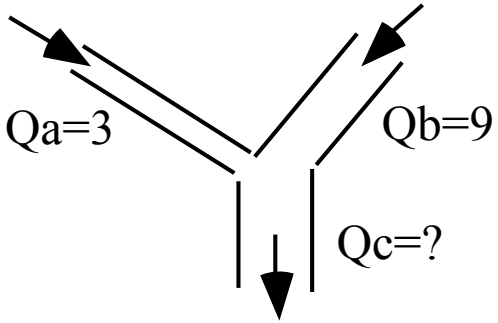


How make learning work?

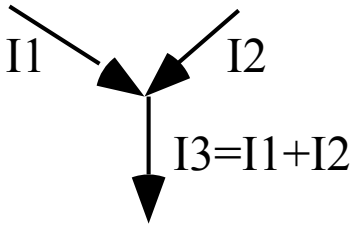
- Use simple hypotheses
  - Always start with the simple ones first
- Constrain $\mathbf{H}$ with priors
  - Do we know something about the domain?
  - Do we have reasonable a priori beliefs on parameters?
- Use many observations
  - Easy to say...
- Cross-validation...

Learning by analogy: definition

Learning by analogy means acquiring new knowledge about an input entity by transferring it from a known similar entity.



Simple Hydraulics Problem



Kirchoff's First Law

One may infer, by analogy, that hydraulics laws are similar to Kirchoff's laws, and Ohm's law.

Which is the central intuition supporting the learning by analogy paradigm?

Discussion

Central intuition supporting learning by analogy:

If two entities are similar in some respects then they could be similar in other respects as well.

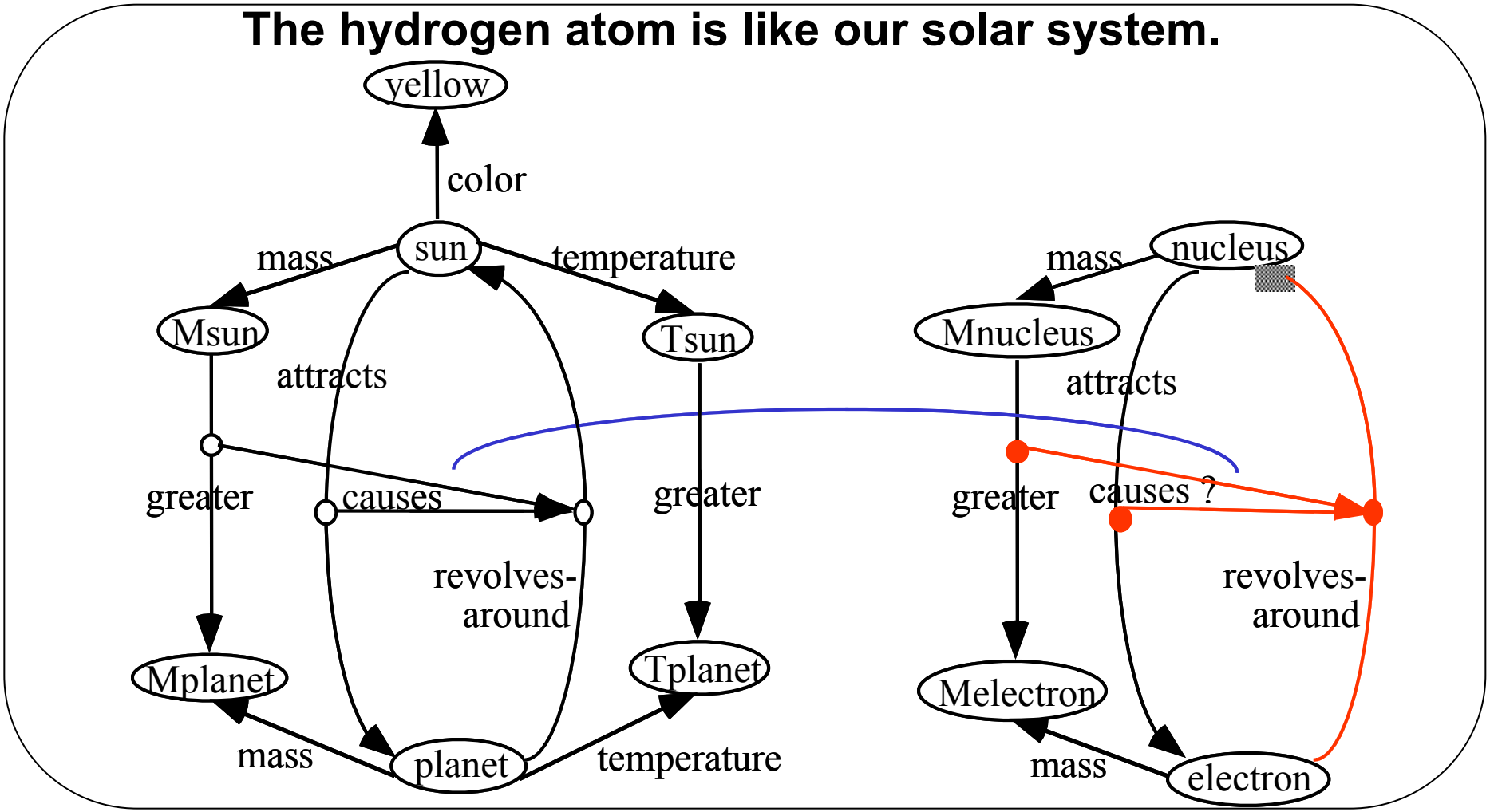
Examples of analogies:

Pressure Drop is like Voltage Drop

A variable in a programming language is like a box.

Provide other examples of analogies.

Learning by analogy: Rutherford's analogy



The Sun has a greater mass than the Earth and attracts it, causing the Earth to revolve around the Sun.

The nucleus also has a greater mass then the electron and attracts it. Therefore it is plausible that the electron also revolves around the nucleus.

Learning by analogy: the learning problem

Given:

- **A partially known target entity T and a goal concerning it.**

Partially understood structure of the hydrogen atom under study.

- **Background knowledge containing known entities.**

Knowledge from different domains, including astronomy, geography, etc.

Find:

- **New knowledge about T obtained from a source entity S belonging to the background knowledge.**

In a hydrogen atom the electron revolves around the nucleus, in a similar way in which a planet revolves around the sun.

Explanation-Based Learning (EBL)

One definition:

Learning *general* problem-solving techniques by observing and analyzing human solutions to *specific* problems.

"Hey! Look what Zog do!"

(drawn by Gary Larson)

Explanation-Based Learning

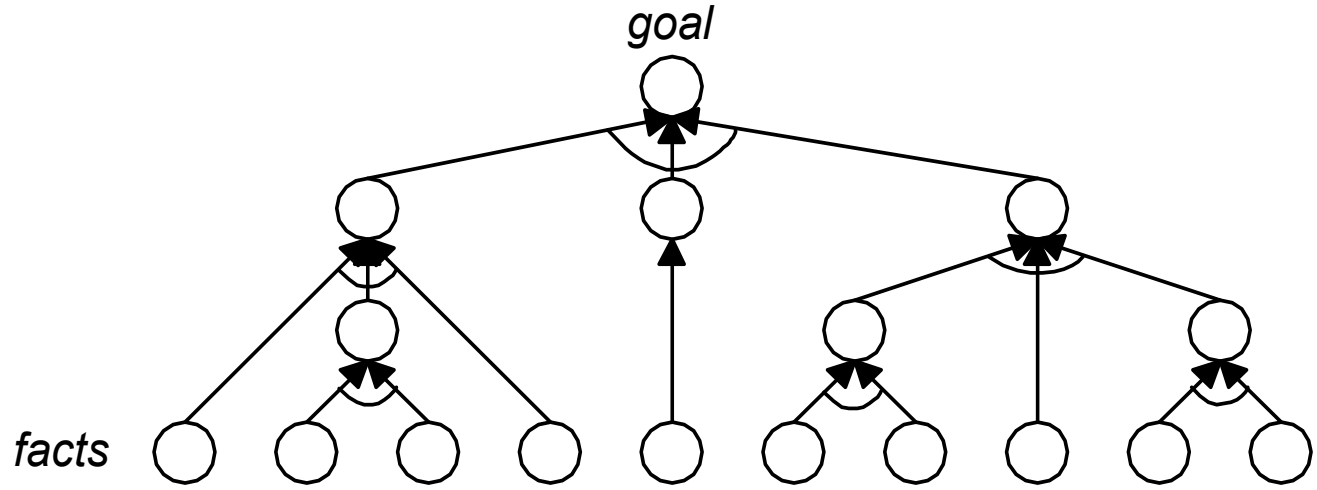
- Learning complex concepts using Induction procedures typically requires a substantial number of training instances.
- But people seem to be able to learn quite a bit from single examples.
- We don't need to see dozens of positive and negative examples of fork(chess) positions in order to learn to avoid this trap in the future and perhaps use it to our advantage.
- What makes such single-example learning possible? The answer is knowledge.
- Much of the recent work in machine learning has moved away from the empirical, data intensive approach described in the last section toward this more analytical knowledge intensive approach.
- A number of independent studies led to the characterization of this approach as explanation-base learning(EBL).
- An EBL system attempts to learn from a single example x by explaining why x is an example of the target concept.
- The explanation is then generalized, and then system's performance is improved through the availability of this knowledge.

Learning by Generalizing Explanations

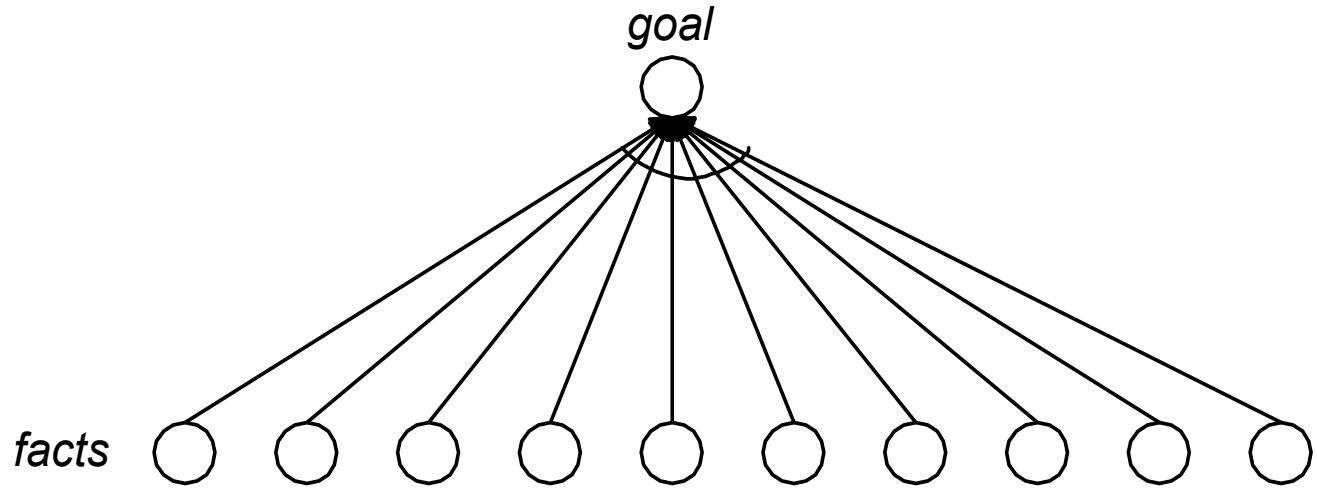
- We can think of EBL programs as accepting the following as input:
  - A training example
  - A goal concept: A high level description of what the program is supposed to learn
  - An operational criterion- A description of which concepts are usable.
  - A domain theory: A set of rules that describe relationships between objects and actions in a domain.
- From this EBL computes a generalization of the training example that is sufficient to describe the goal concept, and also satisfies the operationality criterion.

Standard Approach to EBL

An Explanation (detailed proof of goal)



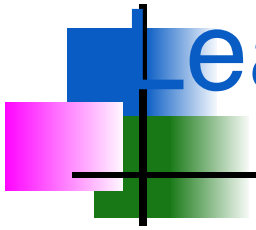
After Learning (go directly from facts to solution):



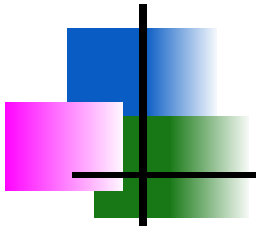
EBL

- Explanation-based generalization (EBG) is an algorithm for EBL and has two steps:
- (1) explain,
  - During the explanation step, the domain theory is used to prune away all the unimportant aspects of the training example with respect to the goal concept.
  - What is left is an explanation of why the training example is an instance of the goal concept. This explanation is expressed in terms that satisfy the operationality criterion.
- (2) generalize
  - The next step is to generalize the explanation as far as possible while still describing the goal concept.

Learning by Simulating Evolution



- Motivation:
  - Evolving a solution
- Genetic Algorithms
  - Modeling search as evolution
    - Mutation
    - Crossover
    - Survival of the fittest
    - Survival of the most diverse
- Conclusions

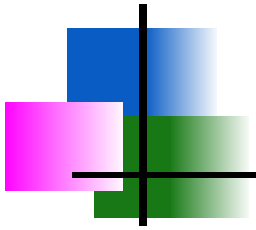


Simulated Evolution

- Evolving a solution
- Begin with population of individuals
  - Individuals = candidate solutions ~chromosomes
- Produce offspring with variation
  - Mutation: change features
  - Crossover: exchange features between individuals
- Apply natural selection
  - Select “best” individuals to go on to next generation
- Continue until satisfied with solution

Genetic Algorithms Applications

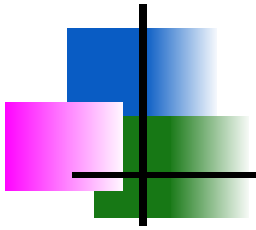
- Search parameter space for optimal assignment
  - Not guaranteed to find optimal, but can approach
- Classic optimization problems:
  - E.g. Traveling Salesman Problem
- Program design (“Genetic Programming”)
- Aircraft carrier landings
-



Genetic Algorithms

- Evolution mechanisms as search technique
  - Produce offspring with variation
    - Mutation, Crossover
  - Select “fittest” to continue to next generation
    - Fitness: Probability of survival
      - Standard: Quality *values* only
      - Rank: Quality *rank* only
      - Rank-space: Rank of sum of *quality & diversity* ranks
- Large population can be robust to local max

Introduction to Neural Nets

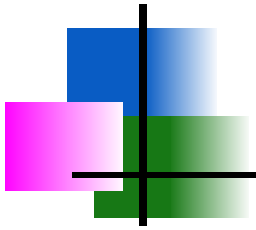


The Human Brain

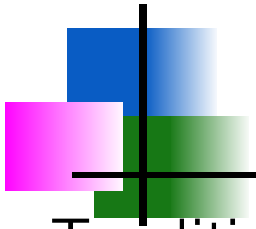
- Computers and the Brain: A Contrast

  - Arithmetic: 1 brain = 1/10 pocket calculator
  - Vision: 1 brain = 1000 super computers
  - Memory of arbitrary details: computer wins
  - Memory of real-world facts: brain wins
  - A computer must be programmed explicitly
  - The brain can learn by experiencing the world

Other Comparisons



ITEM



Computer Operations

Traditionally computers execute a sequence of instructions to accomplish a set task

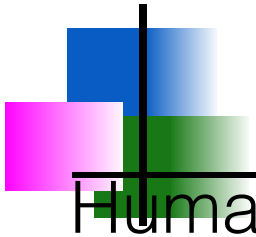
This is a powerful technique if you know the 'algorithm'

It's not very useful if you don't !!

There are many interesting tasks where the algorithm is either unknown or unclear

- Recognizing handwriting - Pattern recognition
- Playing table tennis - Interacting with the environment
- Balancing activities - Optimization

The Question

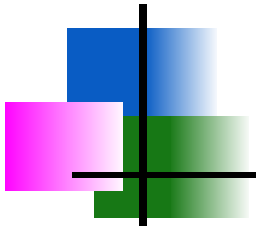


Humans find these tasks relatively simple

We learn by example

The brain is responsible for our ‘computing’ power

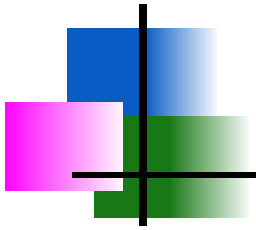
If a machine were constructed using the fundamental building blocks found in the *brain* could it *learn* to do ‘difficult’ tasks ???



Definition

- ". . . Neural nets are basically mathematical models of information processing . . ."
- ". . . (neural nets) refer to machines that have a structure that, at some level, reflects what is known of the structure of the brain . . ."
- "A neural network is a massively parallel distributed processor . . . "

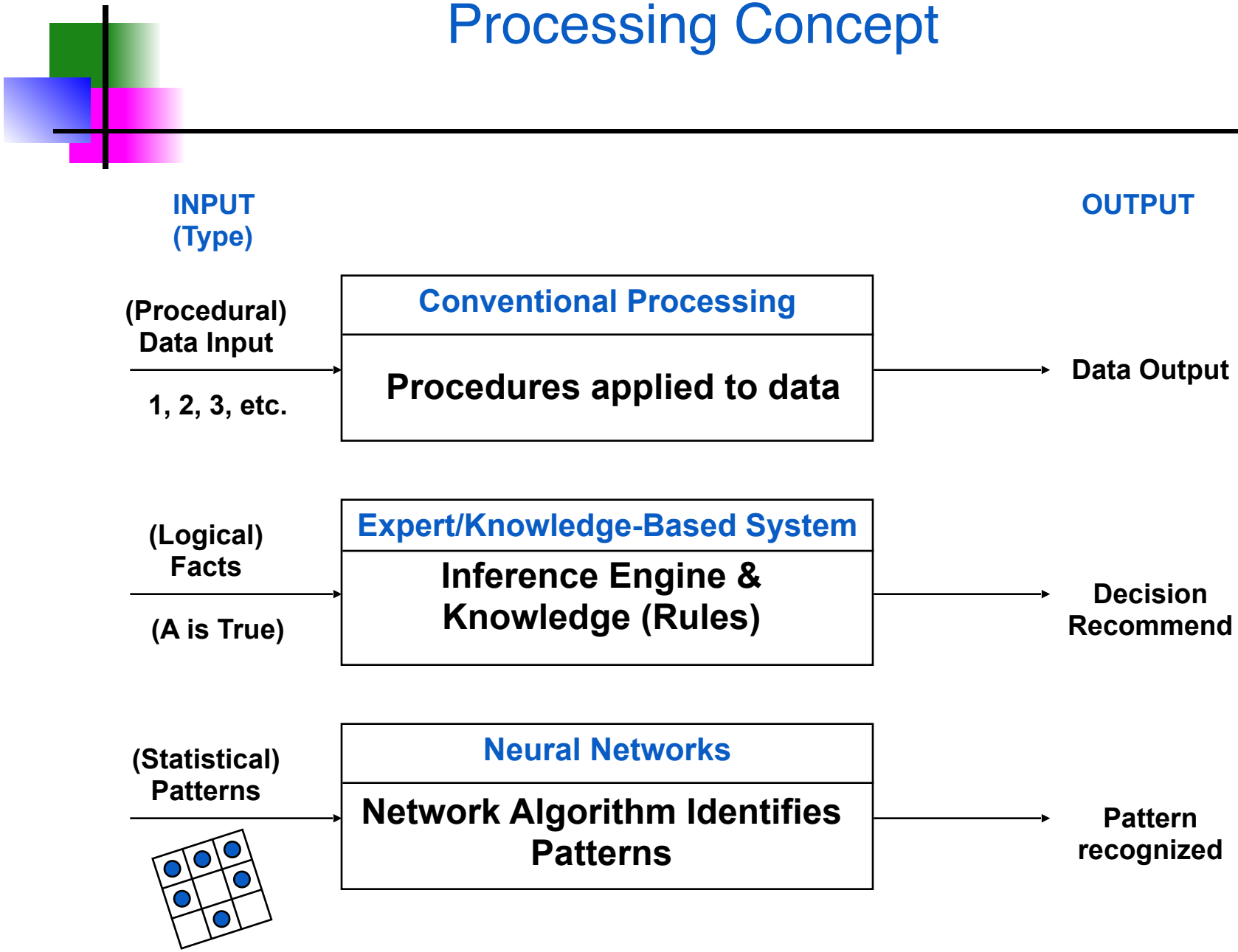
Neural Net Concept



- Artificial Neural Systems are called:
  - neurocomputers
  - neural networks
  - parallel distributed processors
  - connectionists systems

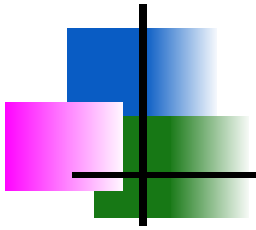
- Basic Philosophy
  - large number of simple “neuron-like” processors which execute global or distributed computation

Processing Concept

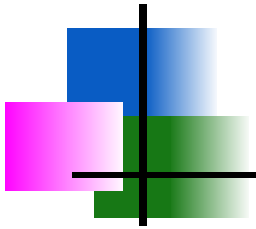


Learning in Neural Nets

Learning



- One of the powerful features of neural networks is learning.
- Learning in neural networks is carried out by adjusting the connection weights among neurons.
- It is similar to a biological nervous system in which learning is carried out by changing synapses connection strengths, among cells.
- Learning may be classified into 2 categories:
  (1) Supervised Learning
  (2) Unsupervised Learning



Supervised Learning

- In supervised learning, the network is presented with inputs together with the target (teacher signal) outputs.

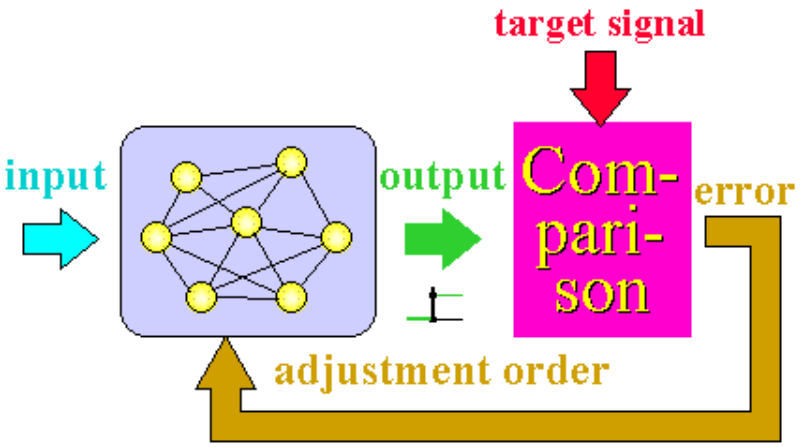
- Then, the neural network tries to produce an output as close as possible to the target signal by adjusting the values of internal weights.

- The most common supervised learning method is the "*error correction method*"

Error Correction Method

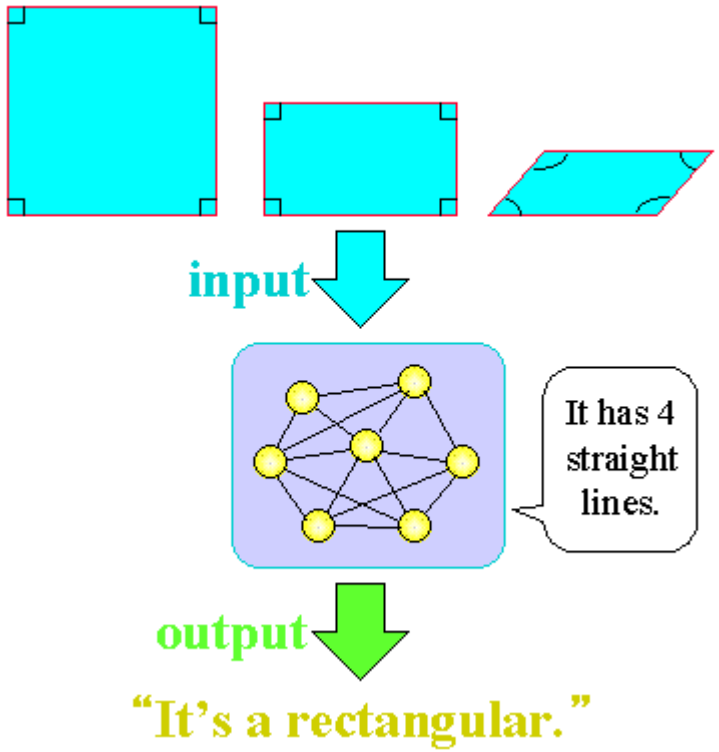
- Error correction method is used for networks which their neurons have discrete output functions.

- Neural networks are trained with this method in order to reduce the error (difference between the network's output and the desired output) to zero.



Unsupervised Learning

- In unsupervised learning, there is no teacher (target signal) from outside and the network adjusts its weights in response to only the input patterns.



- A typical example of unsupervised learning is Hebbian learning.
- clustering
- hidden Markov models,
- self-organizing map (SOM) etc.

Unsupervised Learning

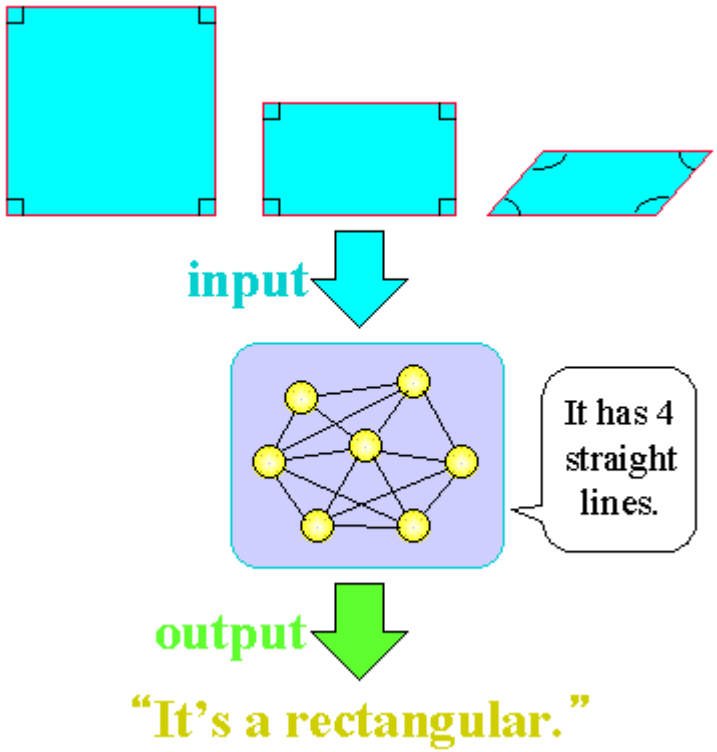
In unsupervised learning, there is no teacher (target signal) from outside and the network adjusts its weights in response to only the input patterns.

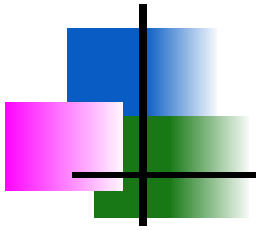
A typical example of unsupervised learning is Hebbian learning.

clustering

hidden Markov models,

self-organizing map (SOM) etc.

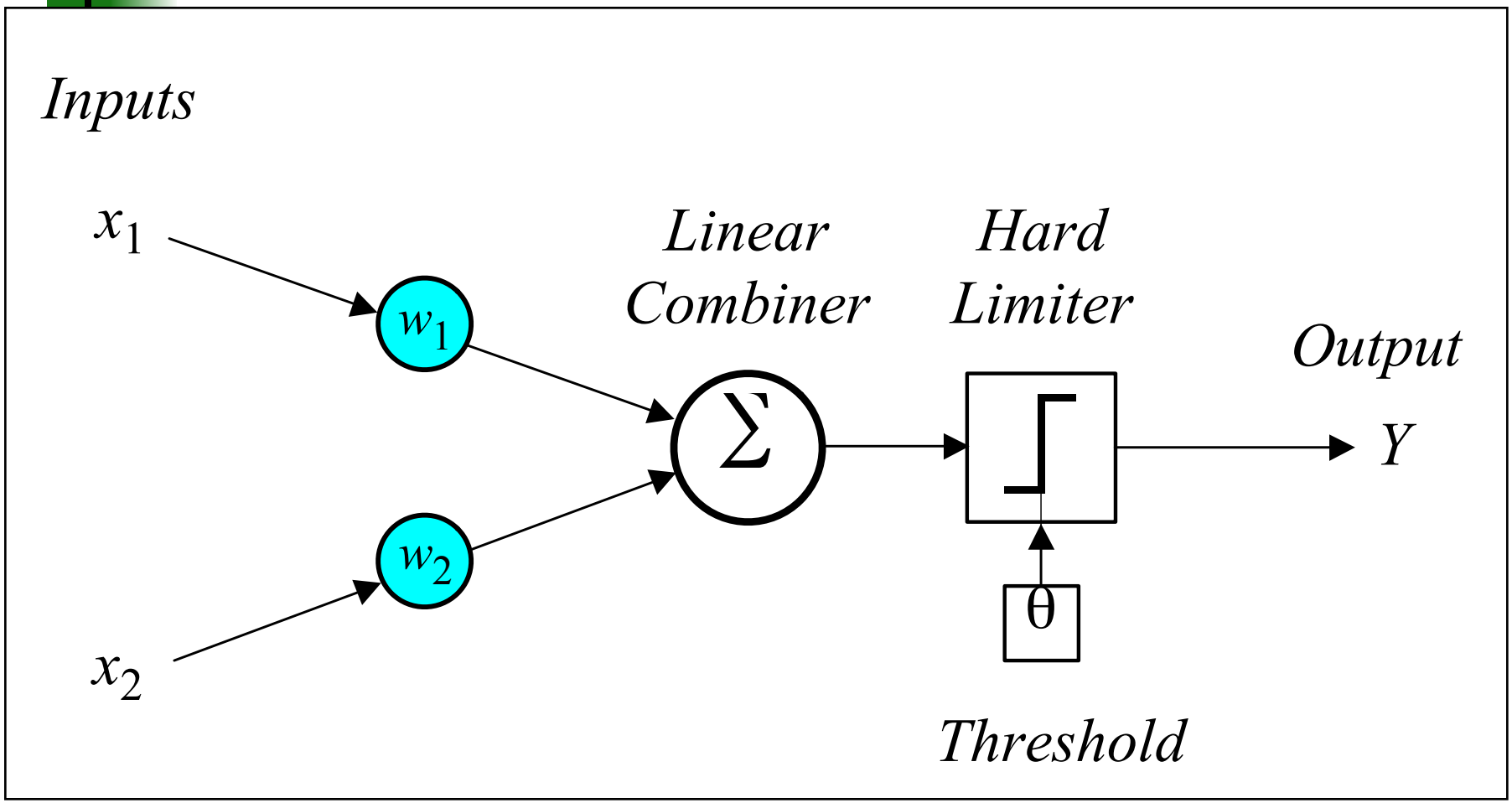




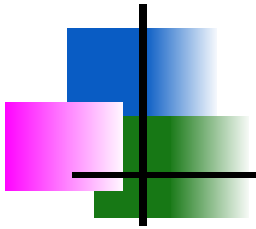
Perceptron

- In 1958, *Frank Rosenblatt* introduced a training algorithm that provided the first procedure for training a simple ANN: a *perceptron*.
- The perceptron is the simplest form of a neural network. It consists of a single neuron with *adjustable* synaptic weights and a *hard limiter*.
- The operation of Rosenblatt's perceptron is based on the *McCulloch and Pitts neuron model*. The model consists of a linear combiner followed by a hard limiter.
- The weighted sum of the inputs is applied to the hard limiter, which produces an output equal to +1 if its input is positive and –1 if it is negative.

Single-layer two-input perceptron



McCulloch/Pitts Neuron

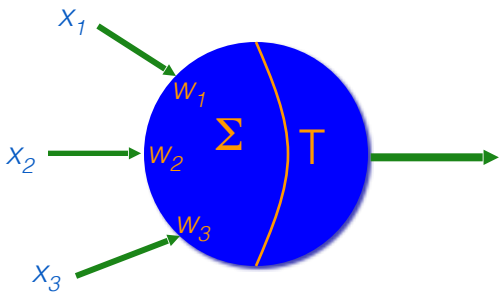


Common Features

- "weight" = strength of connection
- threshold = value of weighted input below which no response is produced
- signals may be:
  - real-valued, or
  - binary-valued:
    - "unipolar" {0, 1}
    - "bipolar" {-1, 1}

McCulloch/Pitts Neuron

- One of the first neuron models to be implemented
- Its output is 1 (fired) or 0
- Each input is weighted with weights in the range -1 to + 1
- It has a threshold value, T



The neuron fires if the following inequality is true:

$$x_1w_1 + x_2w_2 + x_3w_3 > T$$

Single Neuron Application

- *GOAL*: Construct an MCP (McCulloch/Pitts) neuron which will implement the function below:

x1	x2	F
0	0	0
0	1	1
1	0	1
1	1	1

What is this function?

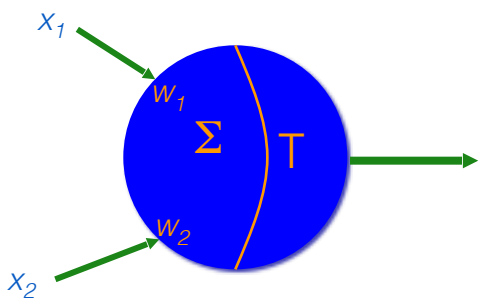
PROBLEM: find the threshold, T, and the weights w_1 and w_2 where:

F is 1 if $x_1w_1 + x_2w_2 > T$

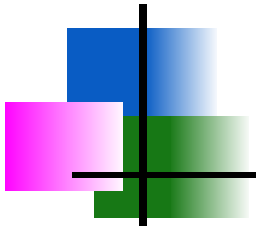
Each line of the function table places conditions on the unknown values:

$0 < T$ *$w_2 > T$*

$w_1 > T$ *$w_1 + w_2 > T$*



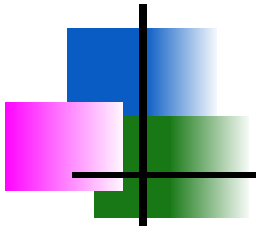
w_1 and w_2 = 0.7 T = 0.5 works



Perceptron

- The aim of the perceptron is to classify inputs, $x_1, x_2, \ldots, x_n$, into one of two classes, say A_1 and A_2.
- In the case of an elementary perceptron, the n-dimensional space is divided by a *hyperplane* into two decision regions. The hyperplane is defined by the *linearly separable function*:

$$\sum_{i=1}^{n} x_i w_i - \theta = 0$$



Perceptron

- *How does the perceptron learn its classification tasks?*
- This is done by making small adjustments in the weights to reduce the difference between the actual and desired outputs of the perceptron. The initial weights are randomly assigned, usually in the range [–0.5, 0.5], and then updated to obtain the output consistent with the training examples.

Perceptron

- If at iteration p, the actual output is $Y(p)$ and the desired output is $Y_d(p)$, then the error is given by:

$$e(p) = Y_d(p) - Y(p)$$

where p = 1, 2, 3, . . .

Iteration p here refers to the pth training example presented to the perceptron.

- If the error, $e(p)$, is positive, we need to increase perceptron output $Y(p)$, but if it is negative, we need to decrease $Y(p)$.

The perceptron learning rule

$$w_i(p+1) = w_i(p) + \alpha \cdot x_i(p) \cdot e(p)$$

where $p = 1, 2, 3, \ldots$

α is the *learning rate*, a positive constant less than unity.

The perceptron learning rule was first proposed by *Rosenblatt* in 1960. Using this rule we can derive the perceptron training algorithm for classification tasks.

Perceptron's training algorithm

Step 1: Initialisation

Set initial weights w_1, w_2,…, w_n and threshold θ to random numbers in the range [–0.5, 0.5].

If the error, $e(p)$, is positive, we need to increase perceptron output $Y(p)$, but if it is negative, we need to decrease $Y(p)$.

Perceptron's training algorithm

Step 2: Activation

Activate the perceptron by applying inputs $x_1(p)$, $x_2(p)$,…, $x_n(p)$ and desired output $Y_d(p)$. Calculate the actual output at iteration $p = 1$

$$Y(p) = step\left[\sum_{i=1}^{n} x_i(p)\, w_i(p) - \theta\right]$$

where n is the number of the perceptron inputs, and *step* is a step activation function.

Perceptron's training algorithm

Step 3: Weight training

Update the weights of the perceptron

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

where $\Delta w_i(p)$ is the weight correction at iteration *p*.

The weight correction is computed by the *delta*

$$\Delta w_i(p) = \alpha \cdot x_i(p) \cdot e(p)$$

Step 4: Iteration

Increase iteration *p* by one, go back to *Step 2* and repeat the process until convergence.

Example of perceptron learning: the logical operation AND

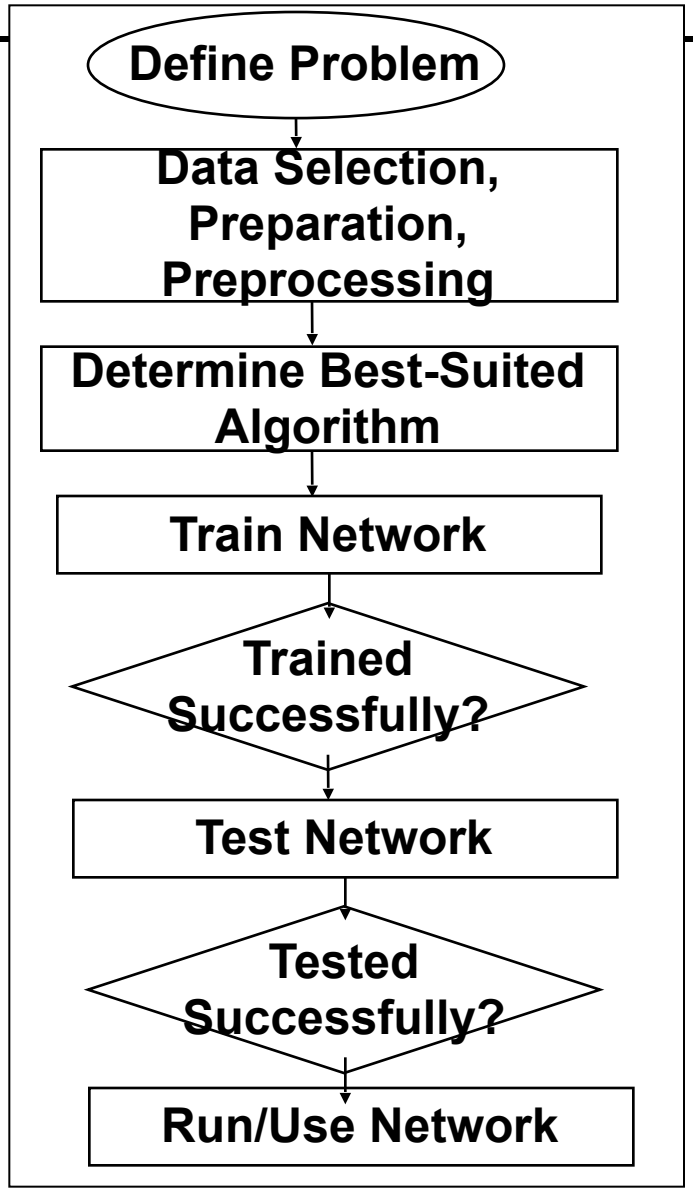
Epoch	Inputs		Desired output	Initial weights		Actual output	Error	Final weights	
	x_1	x_2	Y_d	w_1	w_2	Y	e	w_1	w_2
1	0	0	0	0.3	−0.1	0	0	0.3	−0.1
	0	1	0	0.3	−0.1	0	0	0.3	−0.1
	1	0	0	0.3	−0.1	1	−1	0.2	−0.1
	1	1	1	0.2	−0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	−1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	−1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	−1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$

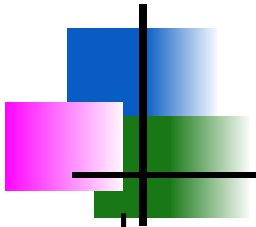
0.0 hunxa

NN Development

- There is no single methodology for NN development
  - A suggested approach is:



Learning by Example

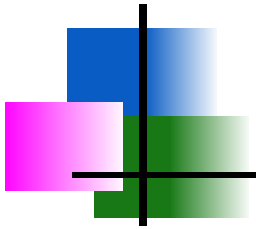


- Learning by induction is a general learning strategy where a concept is learned by drawing inductive inferences from a set of facts

- AI systems that learn by example can be viewed as searching a concept space by means of a decision tree

- The best known approach to constructing a decision tree is called ID3

Iterative Dichotomizer 3 - developed by J. Ross Quinlan in 1975



ID3 Process

- ID3 begins with a set of examples (known facts)
- It ends with a set of rules in the form of a decision tree which may then be applied to unknown cases
- It works by recursively splitting the example set into smaller sets in the most efficient manner possible.

Algorithm

- Given a set of training examples, E, which fall into one of two classes (+ or -)

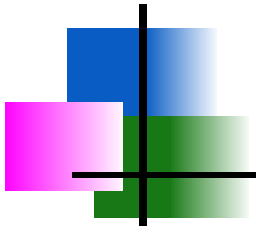
If all the examples of E are negative then create a negative node and stop

If all the examples of E are positive then create a positive node and stop

Otherwise use a criterion to select an attribute A for a new decision node

Use A to partition E into subsets and apply the algorithm to each subset

ID3 uses entropy as the criterion for selecting the next attribute



Aside: Entropy

- Entropy is the basis of Information Theory
- Entropy is a measure of randomness, hence the smaller the entropy the greater the information content
- Given a message with n symbols in which the ith symbol has probability p_i of appearing, the entropy of the message is:

Entropy in ID3

- For a selected attribute value $A = v_i$, determine the number of examples with v_i that are negative (N_n) and the number of examples with v_i that are positive (N_p).

$$H = -\sum_{i=1}^{n} \left[N_p \log_2 \left(\frac{N_p}{N_p + N_n} \right) + N_n \log_2 \left(\frac{N_n}{N_p + N_n} \right) \right]$$

Example

- Learn when it is best to play tennis

- Observe the weather for 14 days and note when tennis was played:

Day	Outlook	Temp	Humidity	Wind	Play
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

So, what is the general rule?

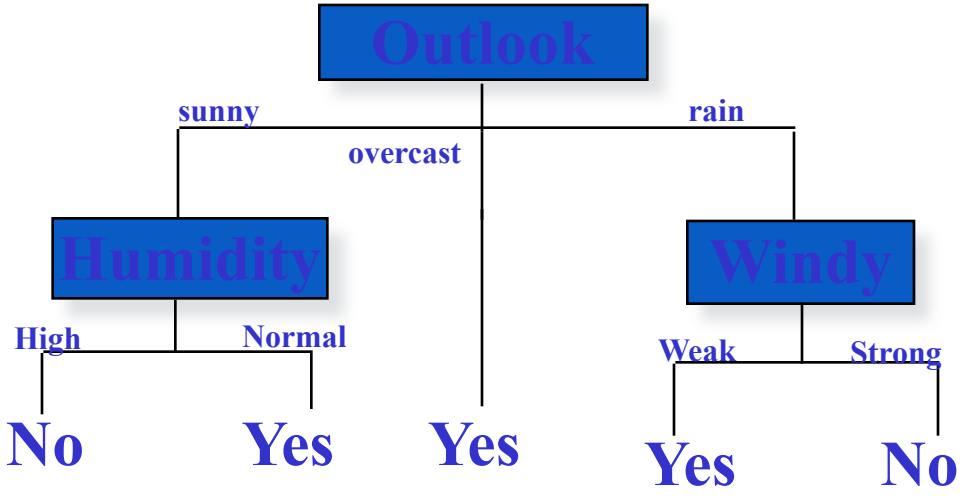
Decision Tree

- Because the examples are neither all positive or all negative, the best discriminator must be selected to become the root node (the one with the smallest entropy)

Attribute	Values	N_n	N_p	Entropy
Outlook	Sunny	3	2	
	Overcast	0	4	9.7096
	Rain	2	3	
Temp	Hot	2	2	
	Mild	2	4	12.7549
	Cool	1	3	
Humidity	High	4	3	11.0383
	Normal	1	6	
Windy	Weak	2	6	12.4902
	Strong	3	3	

The Next Level

- Now look for the minimum entropy among the three outlook cases:



Finished - what happened to Temp?

For the Sunny cases only:

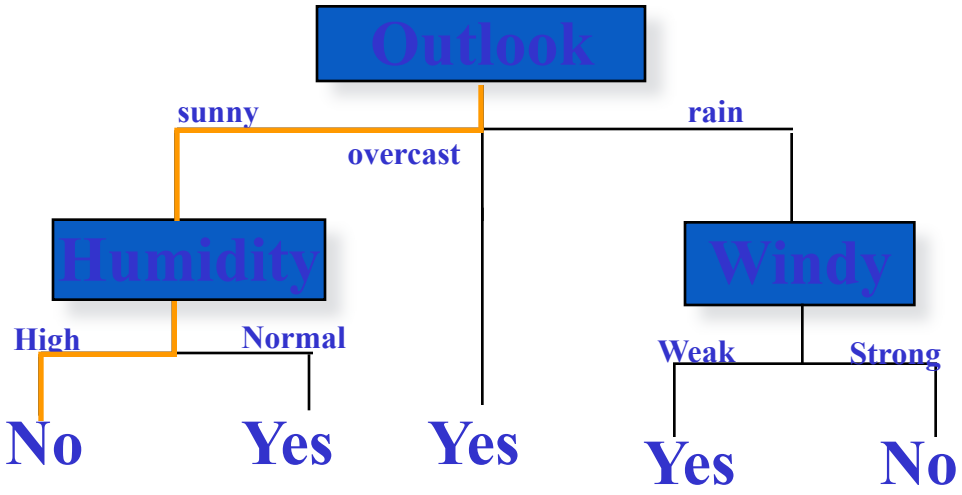
Temp	Hot	2	0	
	Mild	1	1	2.0
	Cool	0	1	
Humidity	High	3	0	0
	Low	0	2	
Windy	Weak	2	1	4.75
	Strong	1	1	

For the Rain cases only:

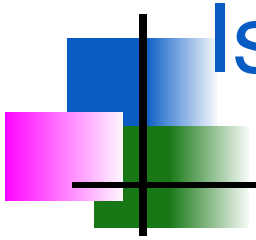
Temp	Hot	0	0	
	Mild	1	2	4.75
	Cool	1	1	
Humidity	High	1	1	4.75
	Low	1	2	
Windy	Weak	0	3	0
	Strong	2	0	

Rules

- The rules for playing tennis are given by the decision tree:



If the outlook is sunny and the humidity is high
Then do not play tennis



Issues in Machine Learning (i.e., Generalization)

- What algorithms are available for learning a concept? How well do they perform?
- How much training data is sufficient to learn a concept with high confidence?
- When is it useful to use prior knowledge?
- Are some training examples more useful than others?
- What are best tasks for a system to learn?
- What is the best way for a system to represent its knowledge?