CSC 480: Artificial Intelligence

Dr. Franz J. Kurfess
Computer Science Department
Cal Poly

Logistics - Nov. 15, 2012

- **AI Nugget presentations scheduled**
  - Section 1:
    - Forrest Reiling: Interaction between Humans and Machines
    - Trevor DeVore: AI in Aerial Vehicles (delayed to Nov. 15 or 20)
  - Section 3:
    - Steve Shenouda: Simulated Therapists (??)
    - Vansteenwyk, Donald W.: Crosswords and Computers (delayed from Nov. 6)
    - Kane Carroll: Facial Recognition Software (delayed from Nov. 6)
    - John Biddle: IBM's Watson, the Jeopardy! Playing Robot (delayed from Nov. 6)
    - Steve Choo: AI and Space Exploration (standby starting Nov. 15)
- **Project**
- **Quiz**
- **Labs**
  - Lab 9 (Decision Tree Learning)
  - Lab 10 (AI and Humor)
- **A3 Competitions**
  - optional; let me know if you're planning to submit something

?

Course Overview

- Introduction
- Intelligent Agents
- Search
  - problem solving through search
  - informed search
- Games
  - games as search problems
- Knowledge and Reasoning
  - reasoning agents
  - propositional logic
  - predicate logic
  - knowledge-based systems
- **Learning**
  - learning from observation
  - neural networks
- Conclusions

Chapter Overview
Learning

- Motivation
- Objectives
- Learning from Observation
  - Learning Agents
  - Inductive Learning
  - Learning Decision Trees
- Computational Learning Theory
  - Probably Approximately Correct (PAC) Learning
- Learning in Neural Networks
  - Neurons and the Brain
  - Neural Networks
  - Perceptrons
  - Multi-layer Networks
  - Applications
- Important Concepts and Terms
- Chapter Summary

Bridge-In

- “knowledge infusion” is not always the best way of providing an agent with knowledge
  - impractical,tedious
  - incomplete, imprecise, possibly incorrect
- adaptivity
  - an agent can expand and modify its knowledge base to reflect changes
- improved performance
  - through learning the agent can make better decisions
- autonomy
  - without learning, an agent can hardly be considered autonomous

Motivation

- learning is important for agents to deal with
  - unknown environments
  - changes
- the capability to learn is essential for the autonomy of an agent
- in many cases, it is more efficient to train an agent via examples, than to "manually" extract knowledge from the examples, and "instill" it into the agent
- agents capable of learning can improve their performance

Objectives

- be aware of the necessity of learning for autonomous agents
- understand the basic principles and limitations of inductive learning from examples
- apply decision tree learning to deterministic problems characterized by Boolean functions
- understand the basic learning methods of perceptrons and multi-layer neural networks
- know the main advantages and problems of learning in neural networks

Learning

- an agent tries to improve its behavior through observation, reasoning, or reflection
  - learning from experience
    - memorization of past percepts, states, and actions
    - generalizations, identification of similar experiences
  - forecasting
    - prediction of changes in the environment
  - theories
    - generation of complex models based on observations and reasoning

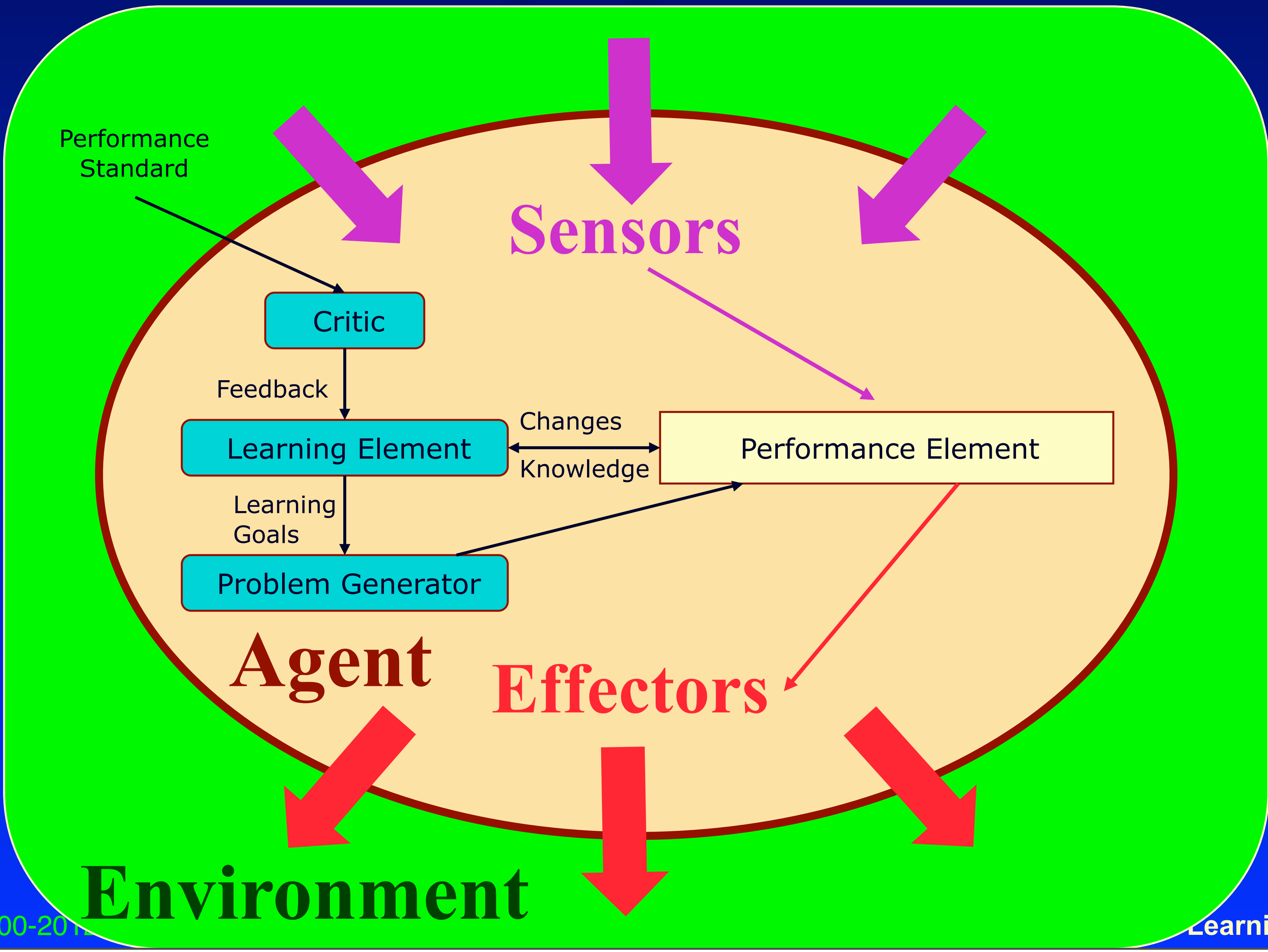
Learning from Observation

- Learning Agents
- Inductive Learning
- Learning Decision Trees

Learning Agents

- based on previous agent designs, such as reflexive, model-based, goal-based agents
  - those aspects of agents are encapsulated into the *performance element* of a learning agent
- a learning agent has an additional *learning element*
  - usually used in combination with a critic and a problem generator for better learning
- most agents learn from examples
  - inductive learning

Learning Agent Model



Forms of Learning

- supervised learning
  - an agent tries to find a function that matches examples from a sample set
    - each example provides an input together with the correct output
  - a teacher provides feedback on the outcome
    - the teacher can be an outside entity, or part of the environment
- unsupervised learning
  - the agent tries to learn from patterns without corresponding output values
- reinforcement learning
  - the agent does not know the exact output for an input, but it receives feedback on the desirability of its behavior
    - the feedback can come from an outside entity, the environment, or the agent itself
    - the feedback may be delayed, and not follow the respective action immediately

Feedback

- provides information about the actual outcome of actions
- supervised learning
  - both the input and the output of a component can be perceived by the agent directly
  - the output may be provided by a teacher
- reinforcement learning
  - feedback concerning the desirability of the agent's behavior is availab
    - not in the form of the correct output
  - may not be directly attributable to a particular action
    - feedback may occur only after a sequence of actions
  - the agent or component knows that it did something right (or wrong), but not what action caused it

Prior Knowledge

- background knowledge available before a task is tackled
- can increase performance or decrease learning time considerably
- many learning schemes assume that no prior knowledge is available
- in reality, some prior knowledge is almost always available
  - but often in a form that is not immediately usable by the agent

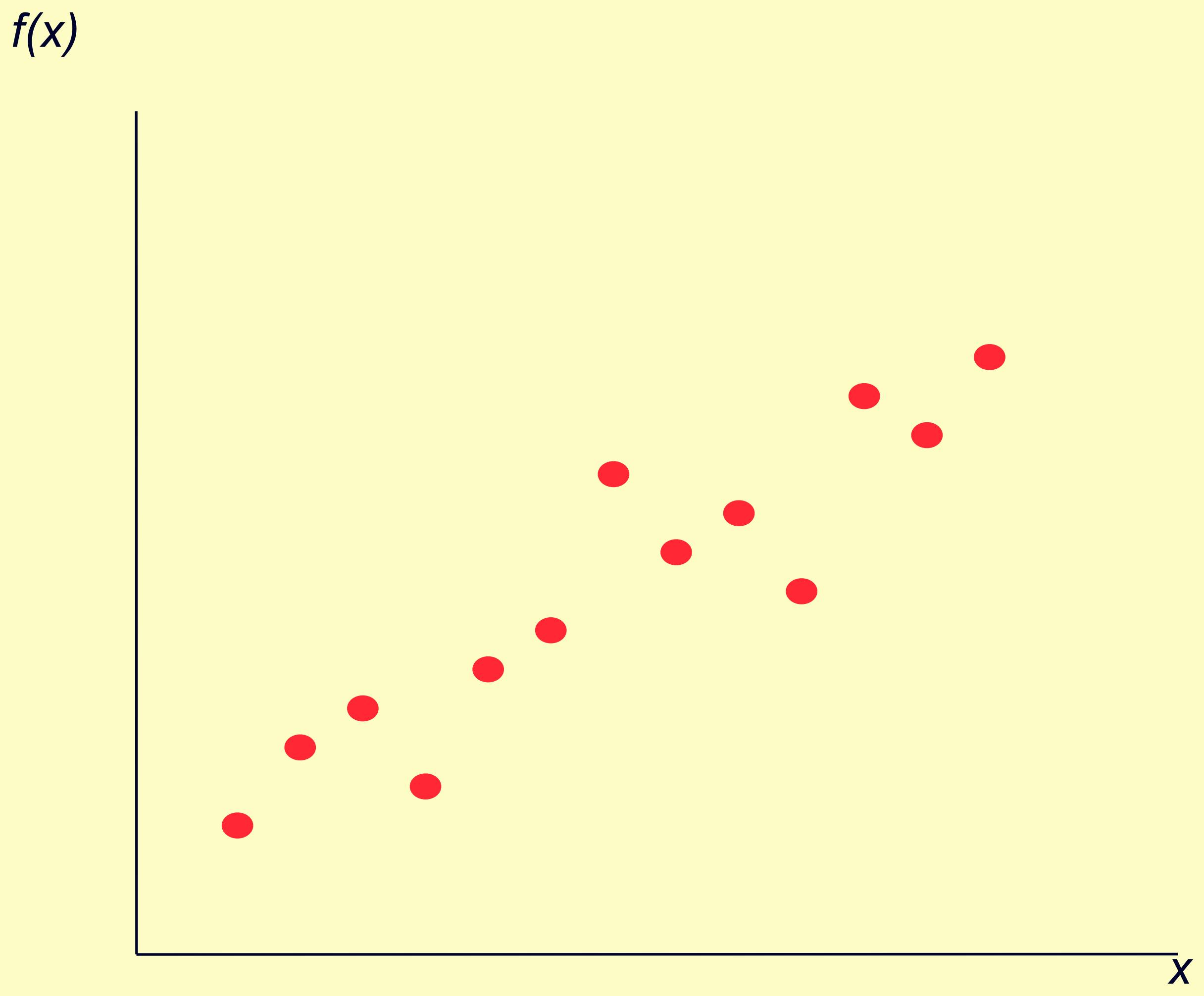
Inductive Learning

- tries to find a function *h* (the *hypothesis*) that approximates a set of samples defining a function *f*
  - the samples are usually provided as input-output pairs *(x, f(x))*
- supervised learning method
- relies on inductive inference, or induction
  - conclusions are drawn from specific instances to more general statements

Hypotheses

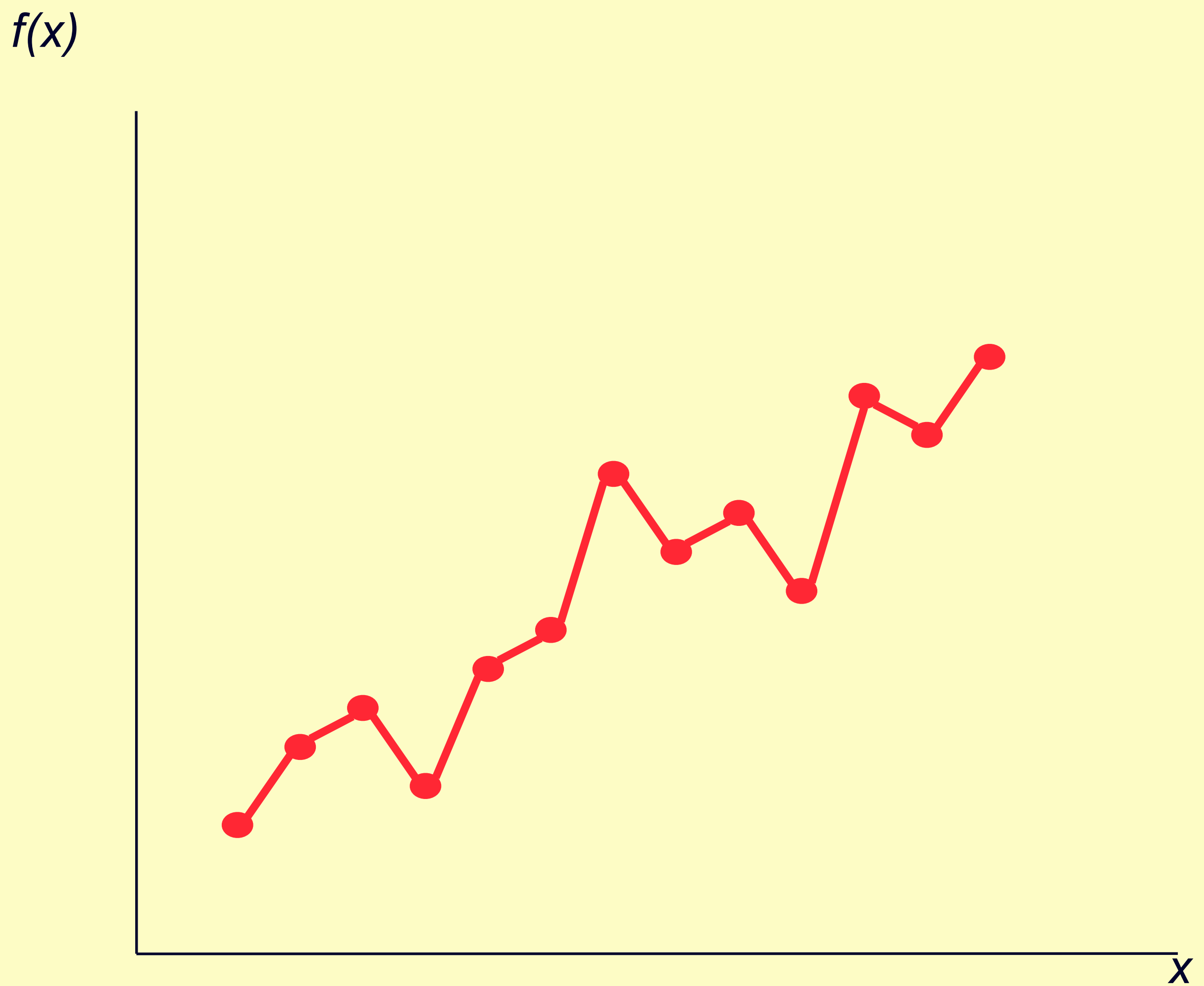
- finding a suitable hypothesis can be difficult
  - since the function *f* is unknown, it is hard to tell if the hypothesis *h* is a good approximation
- the *hypothesis space* describes the set of hypotheses under consideration
  - e.g. polynomials, sinusoidal functions, propositional logic, predicate logic, ...
  - the choice of the hypothesis space can strongly influence the task of finding a suitable function
  - while a very general hypothesis space (e.g. Turing machines) may be guaranteed to contain a suitable function, it can be difficult to find it
- Ockham's razor: if multiple hypotheses are consistent with the data, choose the simplest one

Example Inductive Learning 1



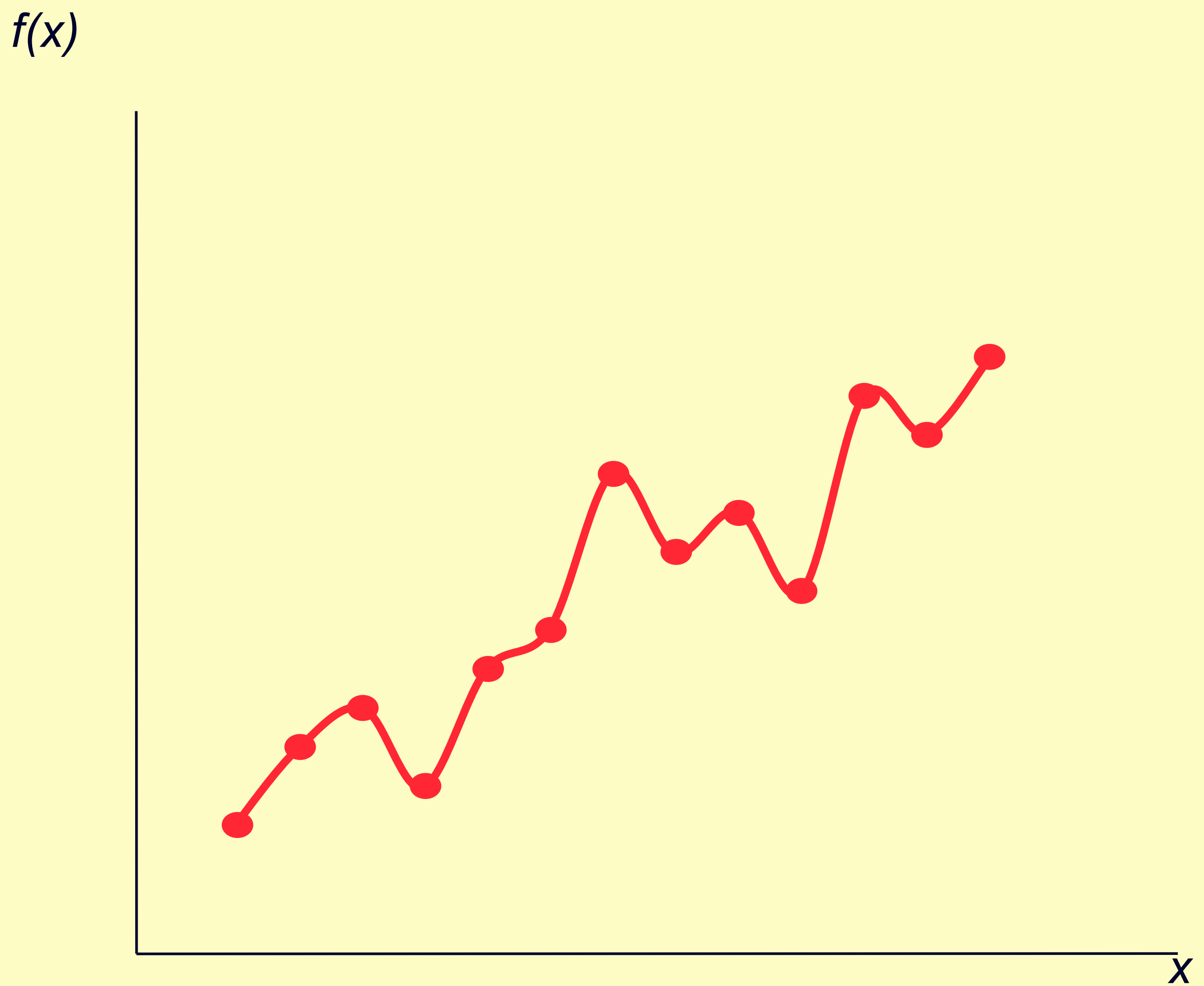
- input-output pairs displayed as points in a plane
- the task is to find a hypothesis (functions) that connects the points
  - either all of them, or most of them
- various performance measures
  - number of points connected
  - minimal surface
  - lowest tension

Example Inductive Learning 2



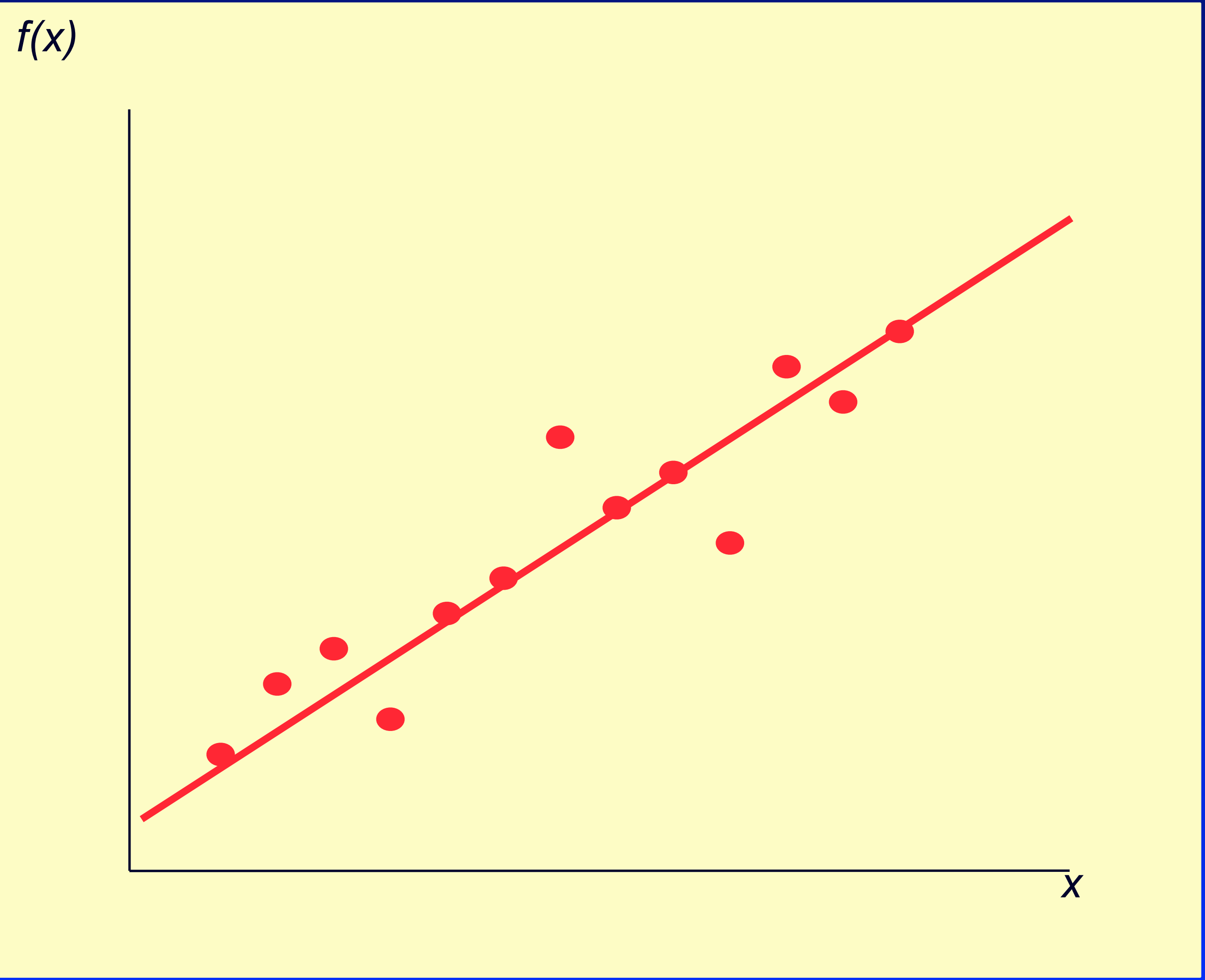
- hypothesis is a function consisting of linear segments
- fully incorporates all sample pairs
  - goes through all points
- very easy to calculate
- has discontinuities at the joints of the segments
- moderate predictive performance

Example Inductive Learning 3



- hypothesis expressed as a polynomial function
- incorporates all samples
- more complicated to calculate than linear segments
- no discontinuities
- better predictive power

Example Inductive Learning 4



- hypothesis is a linear functions
- does not incorporate all samples
- extremely easy to compute
- low predictive power

Learning and Decision Trees

- based on a set of attributes as input, predicted output value, the *decision* is learned
  - it is called *classification* learning for discrete values
  - *regression* for continuous values
- Boolean or binary classification
  - output values are true or false
  - conceptually the simplest case, but still quite powerful
- making decisions
  - a sequence of test is performed, testing the value of one of the attributes in each step
  - when a leaf node is reached, its value is returned
  - good correspondence to human decision-making

Boolean Decision Trees

- compute yes/no decisions based on sets of desirable or undesirable properties of an object or a situation
  - each node in the tree reflects one yes/no decision based on a test of the value of one property of the object
    - the root node is the starting point
    - leaf nodes represent the possible final decisions
  - branches are labeled with possible values
- the learning aspect is to predict the value of a *goal predicate* (also called goal concept)
  - a hypothesis is formulated as a function that defines the goal predicate

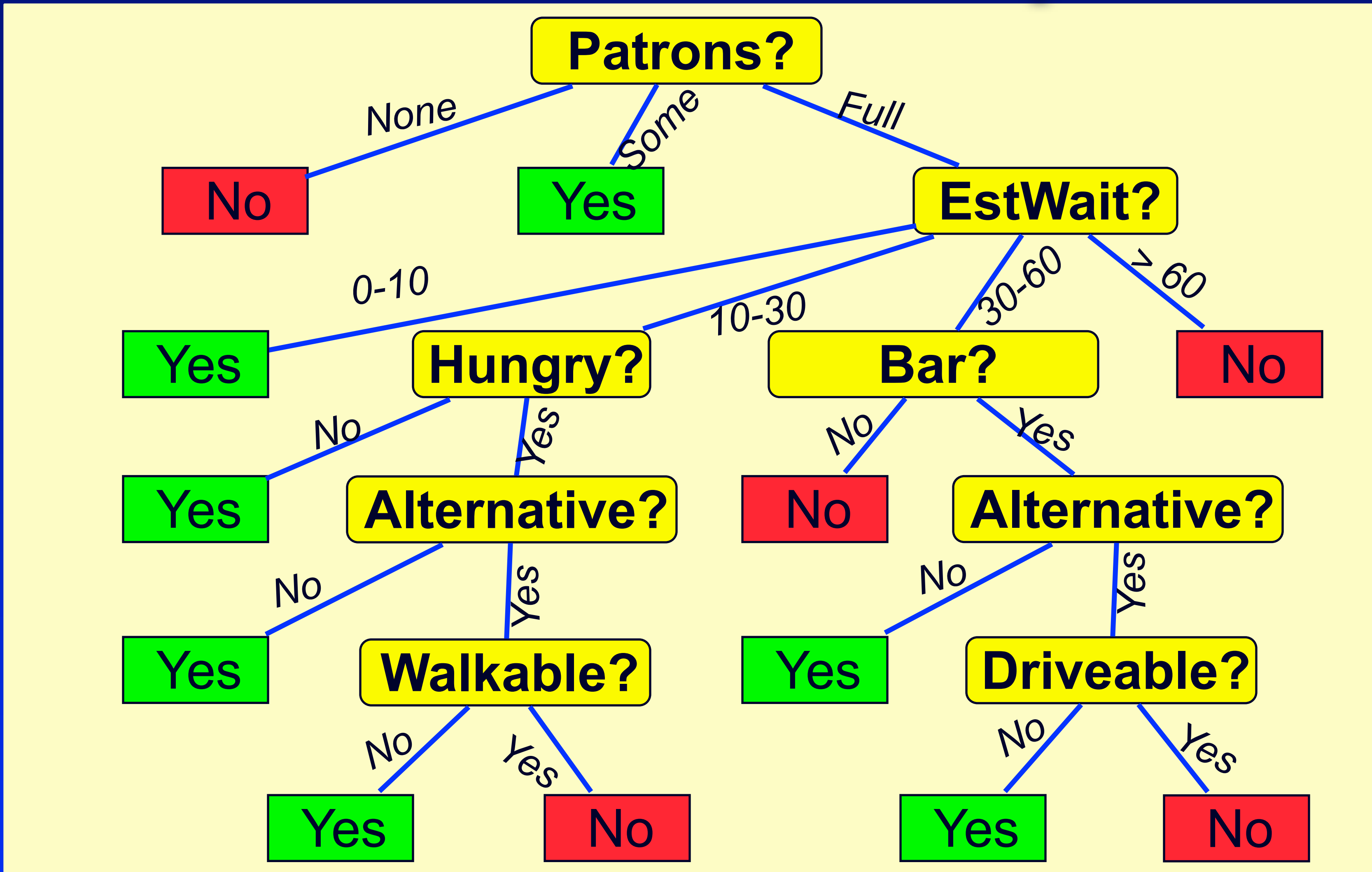
Terminology

- example or sample
  - describes the values of the attributes and the goal
    - a positive sample has the value true for the goal predicate, a negative sample false
- sample set
  - collection of samples used for training and validation
- training
  - the training set consists of samples used for constructing the decision tree
- validation
  - the test set is used to determine if the decision tree performs correctly
    - ideally, the test set is different from the training set

Restaurant Sample Set

Example	Attributes										Goal	Exar
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*	
X1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes	X1
X2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No	X2
X3	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes	X3
X4	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes	X4
X5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No	X5
X6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	Yes	X6
X7	No	Yes	No	No	None	$	Yes	No	Burger	0-10	No	X7
X8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0-10	Yes	X8
X9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No	X9
X10	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10-30	No	X10
X11	No	No	No	No	None	$	No	No	Thai	0-10	No	X11
X12	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes	X12

Decision Tree Example



To wait, or not to wait?

Decision Tree Exercise

- Formulate a decision tree for the following question: *Should I take the opportunity to eliminate a low score in an assignment by doing an extra task?*
  - some possible criteria
    - need for improvement
    - amount of work required
    - deadline
    - other obligations

Expressiveness of Decision Trees

- decision trees can also be expressed in logic as implication sentences
- in principle, they can express propositional logic sentences
  - each row in the truth table of a sentence can be represented as a path in the tree
  - often there are more efficient trees
- some functions require exponentially large decision trees
  - parity function, majority function

Learning Decision Trees

- problem: find a decision tree that agrees with the training set
- trivial solution: construct a tree with one branch for each sample of the training set
  - works perfectly for the samples in the training set
  - may not work well for new samples (generalization)
  - results in relatively large trees
- better solution: find a concise tree that still agrees with all samples
  - corresponds to the simplest hypothesis that is consistent with the training set

Ockham's Razor

The most likely hypothesis is the simplest one that is consistent with all observations.

- general principle for inductive learning
- a simple hypothesis that is consistent with all observations is more likely to be correct than a complex one

Constructing Decision Trees

- in general, constructing the smallest possible decision tree is an intractable problem
- algorithms exist for constructing reasonably small trees
- basic idea: test the most important attribute first
  - attribute that makes the most difference for the classification of an example
    - can be determined through information theory
  - hopefully will yield the correct classification with few tests

Decision Tree Algorithm

- recursive formulation
  - select the best attribute to split positive and negative examples
  - if only positive or only negative examples are left, we are done
  - if no examples are left, no such examples were observed
    - return a default value calculated from the majority classification at the node's parent
  - if we have positive and negative examples left, but no attributes to split them, we are in trouble
    - samples have the same description, but different classifications
    - may be caused by incorrect data (noise), or by a lack of information, or by a truly non-deterministic domain

Restaurant Sample Set

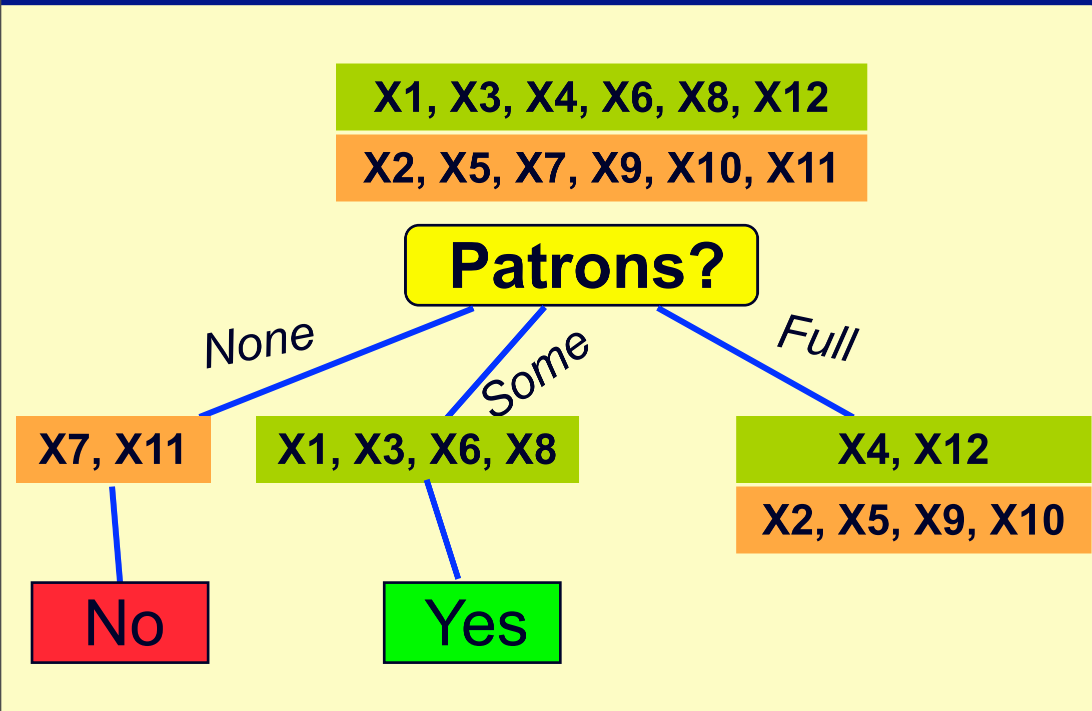
Example					Attributes						Goal	Exa
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*	
X1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes	X1
X2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No	X2
X3	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes	X3
X4	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes	X4
X5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No	X5
X6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	Yes	X6
X7	No	Yes	No	No	None	$	Yes	No	Burger	0-10	No	X7
X8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0-10	Yes	X8
X9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No	X9
X10	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10-30	No	X10
X11	No	No	No	No	None	$	No	No	Thai	0-10	No	X11
X12	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes	X12

Restaurant Sample Set

Example				Attributes						Goal	Exa
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait	
X1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes	X1
X2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No	X2
X3	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes	X3
X4	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes	X4
X5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No	X5
X6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	Yes	X6
X7	No	Yes	No	No	None	$	Yes	No	Burger	0-10	No	X7
X8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0-10	Yes	X8
X9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No	X9
X10	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10-30	No	X10
X11	No	No	No	No	None	$	No	No	Thai	0-10	No	X11
X12	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes	X12

- select best attribute
  - candidate 1: *Pat* — *Some* and *None* in agreement with goal
  - candidate 2: *Type* — No values in agreement with goal

Partial Decision Tree



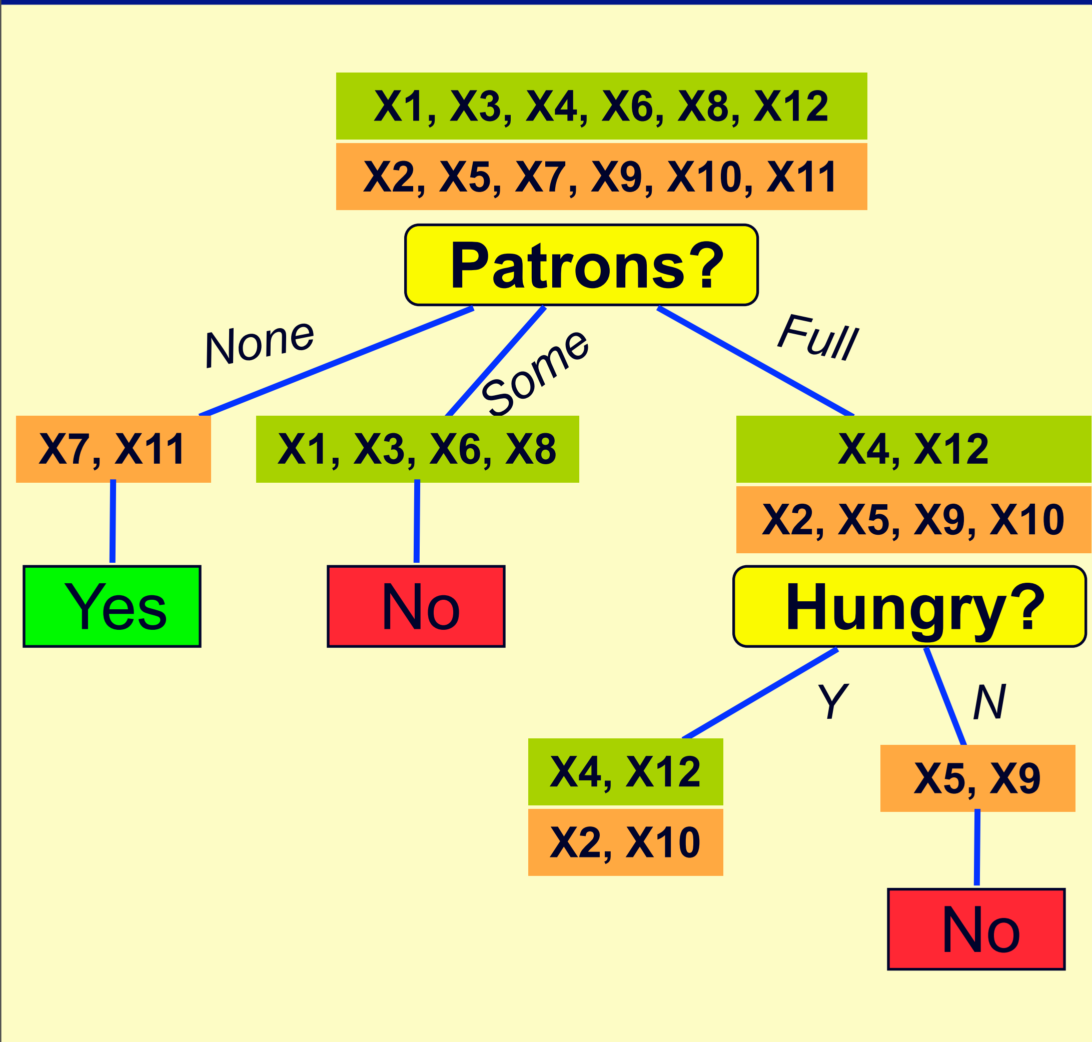
- *Patrons* needs further discrimination only for the *Full* value
- *None* and *Some* agree with the *WillWait* goal predicate
- the next step will be performed on the remaining samples for the *Full* value of *Patrons*

Restaurant Sample Set

Example				Attributes						Goal	Exa
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait	
X2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No	X2
X4	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes	X4
X5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No	X5
X9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No	X9
X10	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10-30	No	X10
X12	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes	X12

- select next best attribute
  - candidate 1: *Hungry* — *No* in agreement with goal
  - candidate 2: *Type* — No values in agreement with goal

Partial Decision Tree



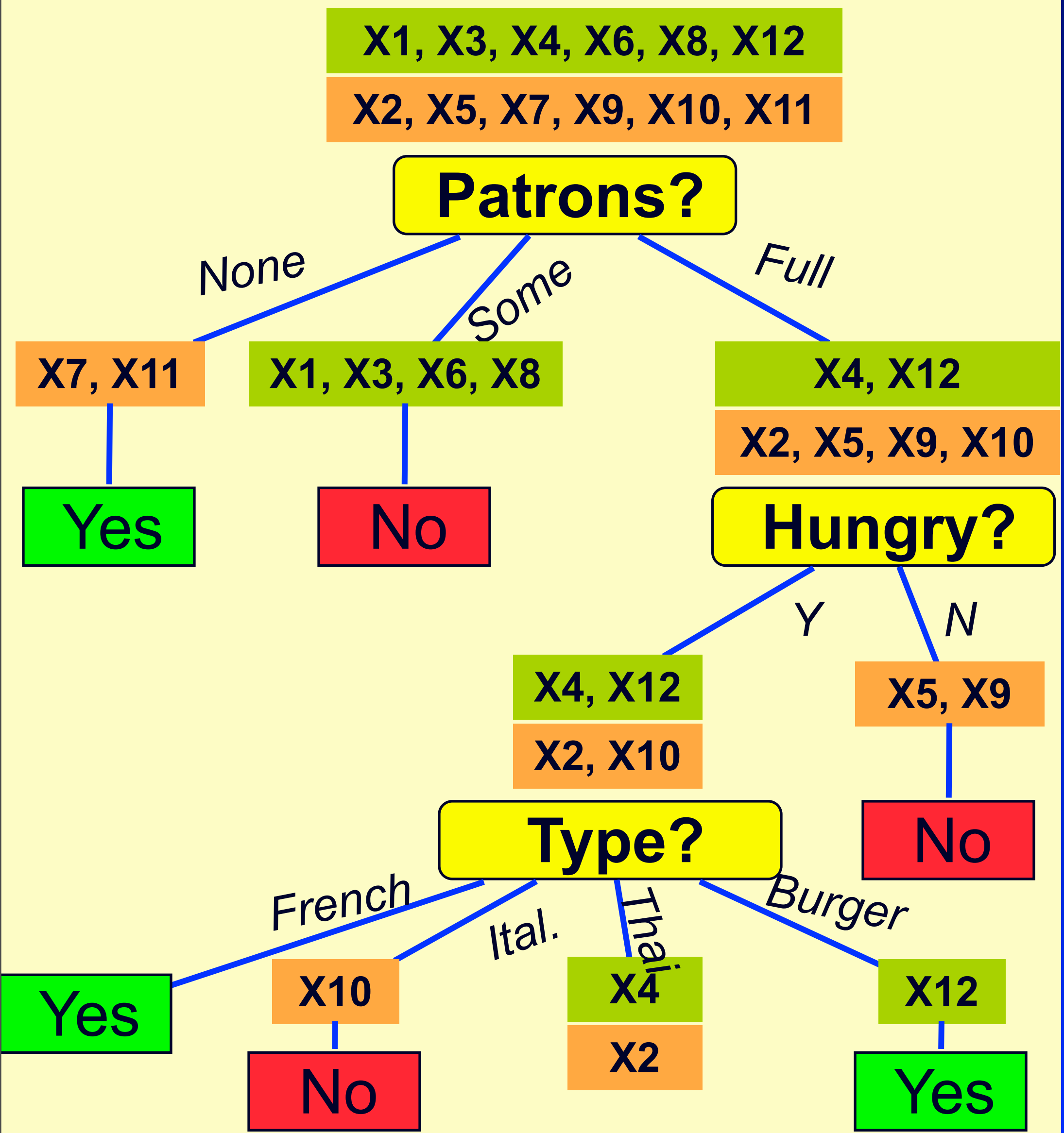
- *Hungry* needs further discrimination only for the *Yes* value
- *No* agrees with the *WillWait* goal predicate
- the next step will be performed on the remaining samples for the *Yes* value of *Hungry*

Restaurant Sample Set

Example				Attributes							Goal	Exa
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*	
X2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No	X2
X4	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes	X4
X10	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10-30	No	X10
X12	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes	X12

- select next best attribute
  - candidate 1: *Type* — *Italian, Burger* in agreement with goal
  - candidate 2: *Friday* — *No* in agreement with goal

Partial Decision Tree

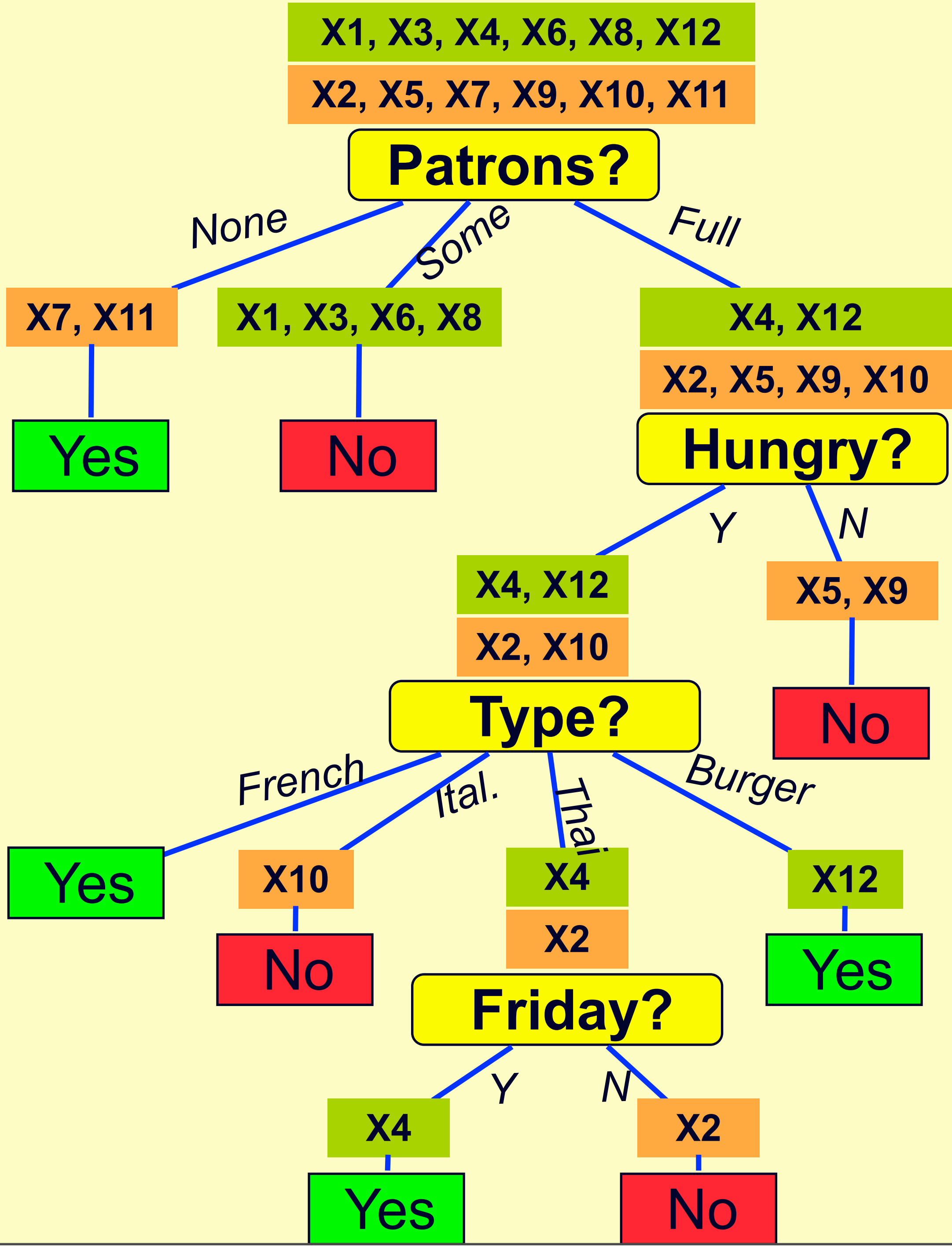


- *Hungry* needs further discrimination only for the *Yes* value
- *No* agrees with the *WillWait* goal predicate
- the next step will be performed on the remaining samples for the *Yes* value of *Hungry*

Restaurant Sample Set

Example					Attributes						Goal	Exa
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*	
X2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No	X2
X4	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes	X4

- select next best attribute
  - candidate 1: *Friday* *Yes* and *No* in agreement with goal



Decision Tree

- the two remaining samples can be made consistent by selecting *Friday* as the next predicate
- no more samples left

Performance of Decision Tree Learning

- quality of predictions
  - predictions for the classification of unknown examples that agree with the correct result are obviously better
  - can be measured easily after the fact
  - it can be assessed in advance by splitting the available examples into a training set and a test set
    - learn the training set, and assess the performance via the test set
- size of the tree
  - a smaller tree (especially depth-wise) is a more concise representation

Noise and Over-fitting

- the presence of irrelevant attributes ("noise") may lead to more degrees of freedom in the decision tree
  - the hypothesis space is unnecessarily large
- *overfitting* makes use of irrelevant attributes to distinguish between samples that have no meaningful differences
  - e.g. using the day of the week when rolling dice
  - over-fitting is a general problem for all learning algorithms
- *decision tree pruning* identifies attributes that are likely to be irrelevant
  - very low information gain
- *cross-validation* splits the sample data in different training and test sets
  - results are averaged

Ensemble Learning

- multiple hypotheses (an *ensemble*) are generated, and their predictions combined
  - by using multiple hypotheses, the likelihood for misclassification is hopefully lower
  - also enlarges the hypothesis space
- *boosting* is a frequently used ensemble method
  - each example in the training set has a weight associated
  - the weights of incorrectly classified examples are increased, and a new hypothesis is generated from this new weighted training set
  - the final hypothesis is a weighted-majority combination of all the generated hypotheses

Computational Learning Theory

- relies on methods and techniques from theoretical computer science, statistics, and AI
- used for the formal analysis of learning algorithms
- basic principles
  - a hypothesis is seriously wrong
    - it will most likely generate a false prediction even for small numbers of examples
  - hypothesis is consistent with a large number of examples
    - most likely it is quite good, or *probably approximately correct*

Probably Approximately Correct (PAC) Learning

- *approximately correct* hypothesis
  - its error lies within a small constant of the true result
  - by testing a sufficient number of examples, one can see if a hypothesis has a high probability of being approximately correct
- *stationary assumption*
  - training and test sets follow the same probability distribution
  - there is a connection between the past (known) and the future (unknown)
  - a selection of non-representative examples will not result in good learning

Learning in Neural Networks

- Neurons and the Brain
- Neural Networks
- Perceptrons
- Multi-layer Networks
- Applications

Neural Networks

- complex networks of simple computing elements
- capable of learning from examples
  - with appropriate learning methods
- collection of simple elements performs high-level operations
  - thought
  - reasoning
  - consciousness

Neural Networks and the Brain

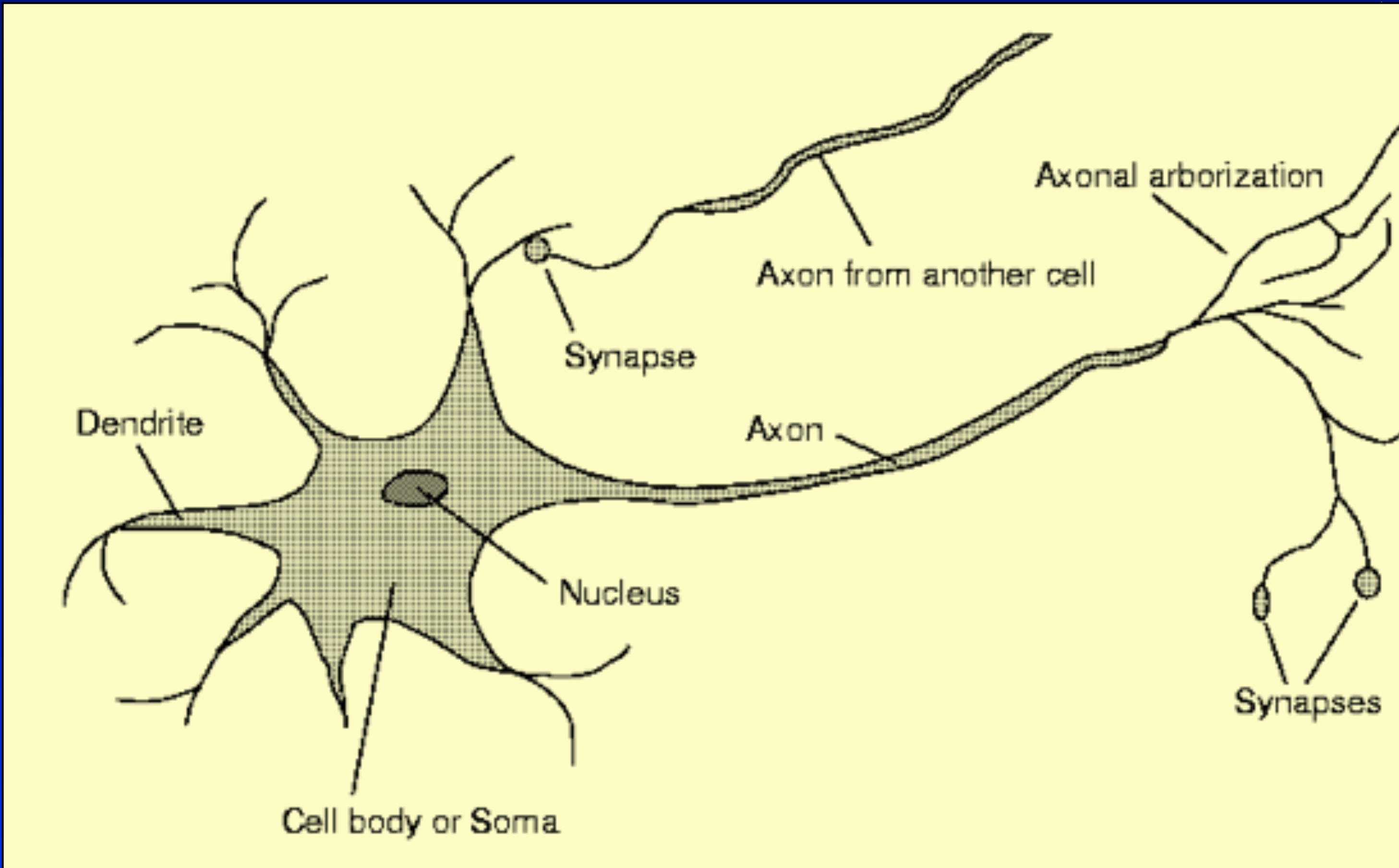
- *brain*
  - set of interconnected modules
  - performs information processing operations at various levels
    - sensory input analysis
    - memory storage and retrieval
    - reasoning
    - feelings
    - consciousness
- *neurons*
  - basic computational elements
  - heavily interconnected with other neurons

Neuron Diagram

- soma
  - cell body
- dendrites
  - incoming branches
- axon
  - outgoing branch
- synapse
  - junction between a dendrite and an axon from another neuron

[Russell & Norvig, 1995]

Neuron Diagram



[Russell & Norvig, 1995]

soma
- cell body

dendrites
- incoming branches

axon
- outgoing branch

synapse
- junction between a dendrite and an axon from another neuron

Computer vs. Brain

	Computer	**Brain**
Computational units	1-1000 CPUs 10^7 gates/CPU	10^{11} neurons
Storage units	10^{10} bits RAM 10^{11} bits disk	10^{11} neurons 10^{14} synapses
Cycle time	10^{-9} sec (1GHz)	10^{-3} sec (1kHz)
Bandwidth	10^9 sec	10^{14} sec
Neuron updates/sec	10^5	10^{14}

Computer Brain vs. Cat Brain

- in 2009 IBM makes a supercomputer significantly smarter than cat
  - "IBM has announced a software simulation of a mammalian cerebral cortex that's significantly more complex than the cortex of a cat. And, just like the actual brain that it simulates, they still have to figure out how it works."

http://arstechnica.com/science/news/2009/11/ibm-makes-supercomputer-significantly-smarter-than-cat.ars?utm_source=rss&utm_medium=rss&utm_campaign=rss

http://static.arstechnica.com/cat_computer_ars.jpg

Google Neural Network learns about ???

- What does a really large NN learn from watching Youtube videos for one week?
- NN implementation
  - computation spread across 16,000 CPU cores
  - more than 1 billion connections in the NN
- http://googleblog.blogspot.com/2012/06/using-large-scale-brain-simulations-for.html

Cat Discovery

- "cat" discovery in NN
  - learned to identify a category of images with cats
  - Google blog post
    - https://plus.google.com/u/0/+ResearchatGoogle/posts/EMyhnBetd2F
  - published paper
    - http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/archive/unsupervised_icml2012.pdf

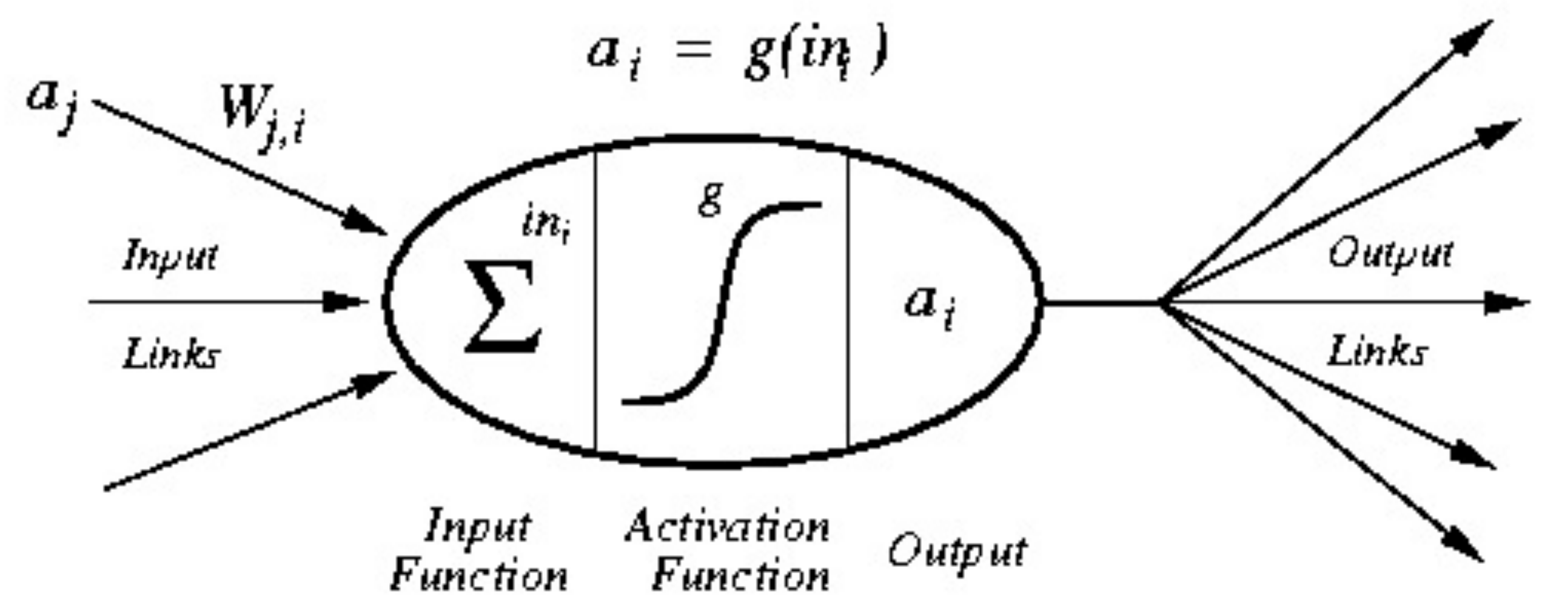
http://1.bp.blogspot.com/-VENOsYD1uJc/T-nkLAiANtI/AAAAAAAAJWc/2KCTI3OsI18/s1600/cat+detection.jpeg

Artificial Neuron Diagram

[Russell & Norvig, 1995]

- weighted inputs are summed up by the *input function*
- the (nonlinear) *activation function* calculates the activation value, which determines the output

Artificial Neuron Diagram



[Russell & Norvig, 1995]

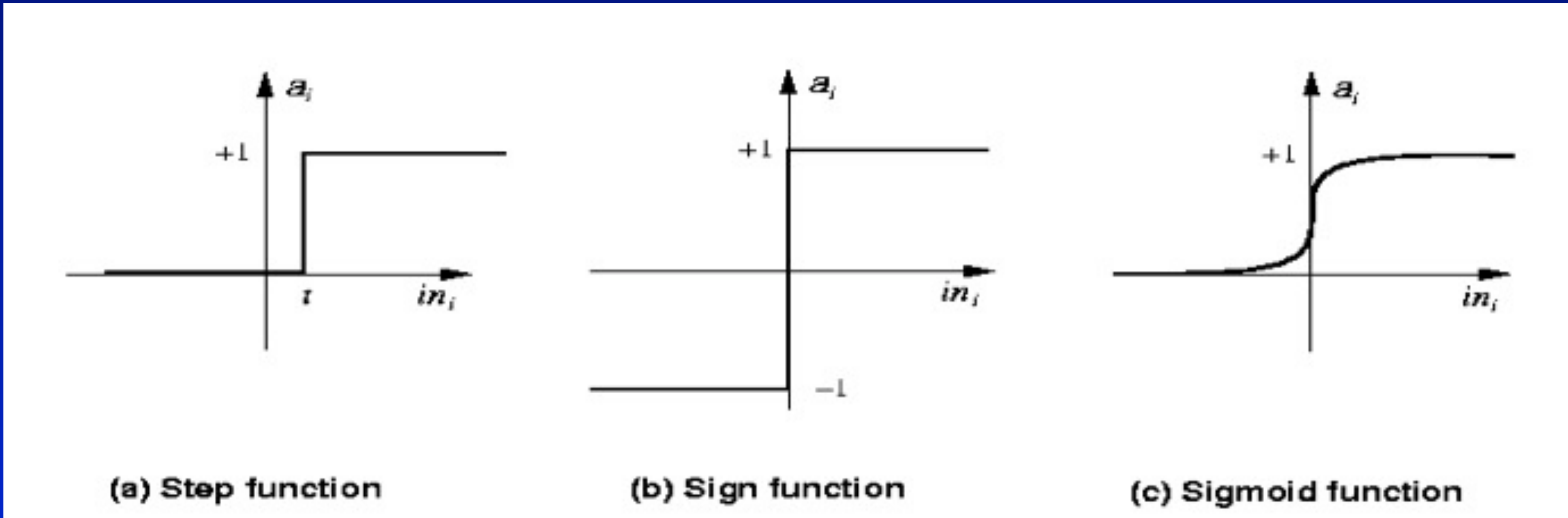
- weighted inputs are summed up by the *input function*
- the (nonlinear) *activation function* calculates the activation value, which determines the output

Common Activation Functions

[Russell & Norvig, 1995]

- $\text{Step}_t(x) = 1$ if $x >= t$, else 0
- $\text{Sign}(x) = +1$ if $x >= 0$, else -1
- $\text{Sigmoid}(x) = 1/(1+e^{-x})$

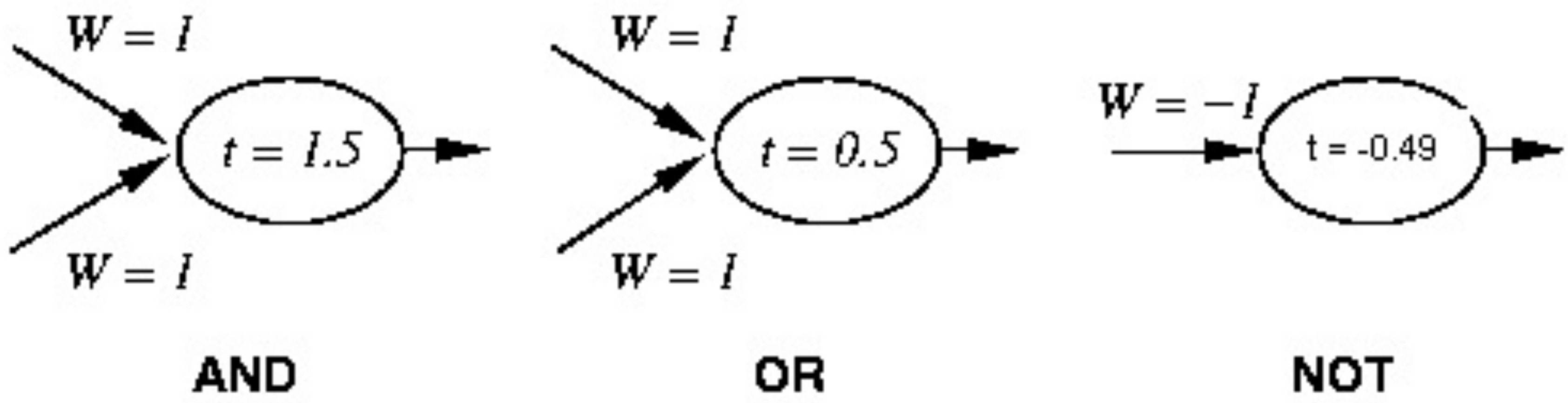
Common Activation Functions



[Russell & Norvig, 1995]

- $Step_t(x)$ = 1 if x >= t, else 0
- Sign(x) = +1 if x >= 0, else –1
- Sigmoid(x) = $1/(1+e^{-x})$

Neural Networks and Logic Gates



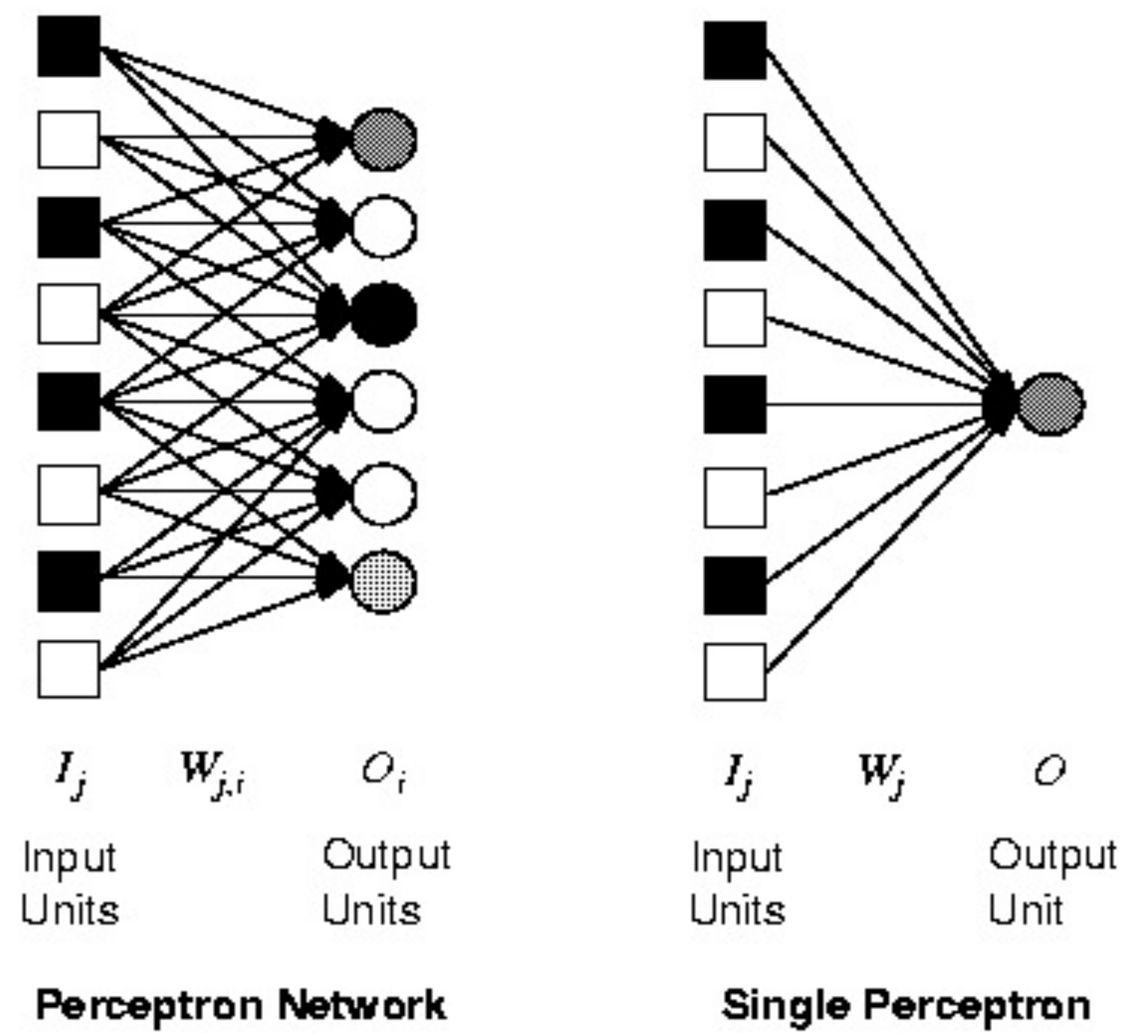
[Russell & Norvig, 1995]

- simple neurons with can act as logic gates
    - appropriate choice of activation function, threshold, and weights
        - step function as activation function

Network Structures

- in principle, networks can be arbitrarily connected
  - occasionally done to represent specific structures
    - semantic networks
    - logical sentences
  - makes learning rather difficult
- layered structures
  - networks are arranged into layers
  - interconnections mostly between two layers
  - some networks have feedback connections

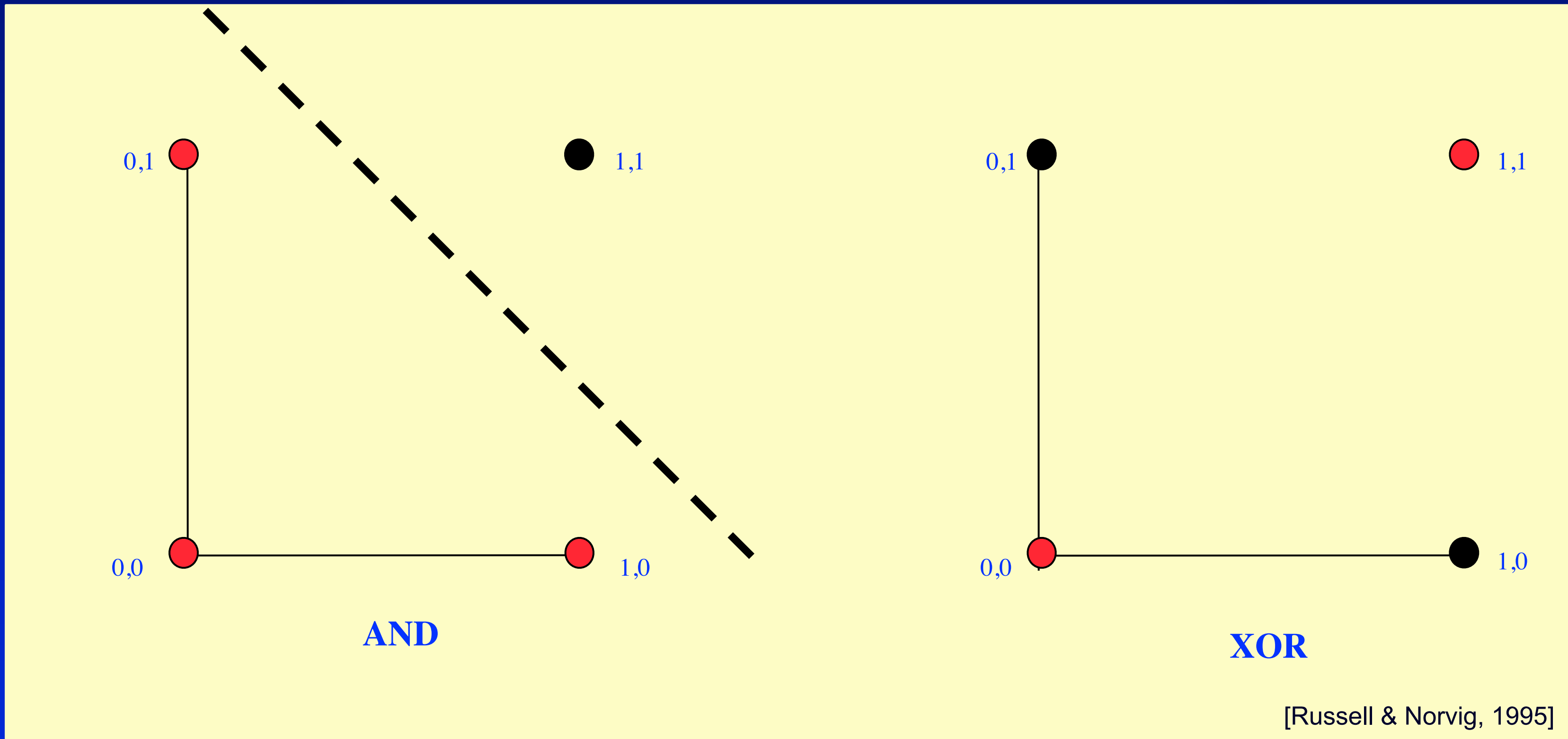
Perceptrons



[Russell & Norvig, 1995]

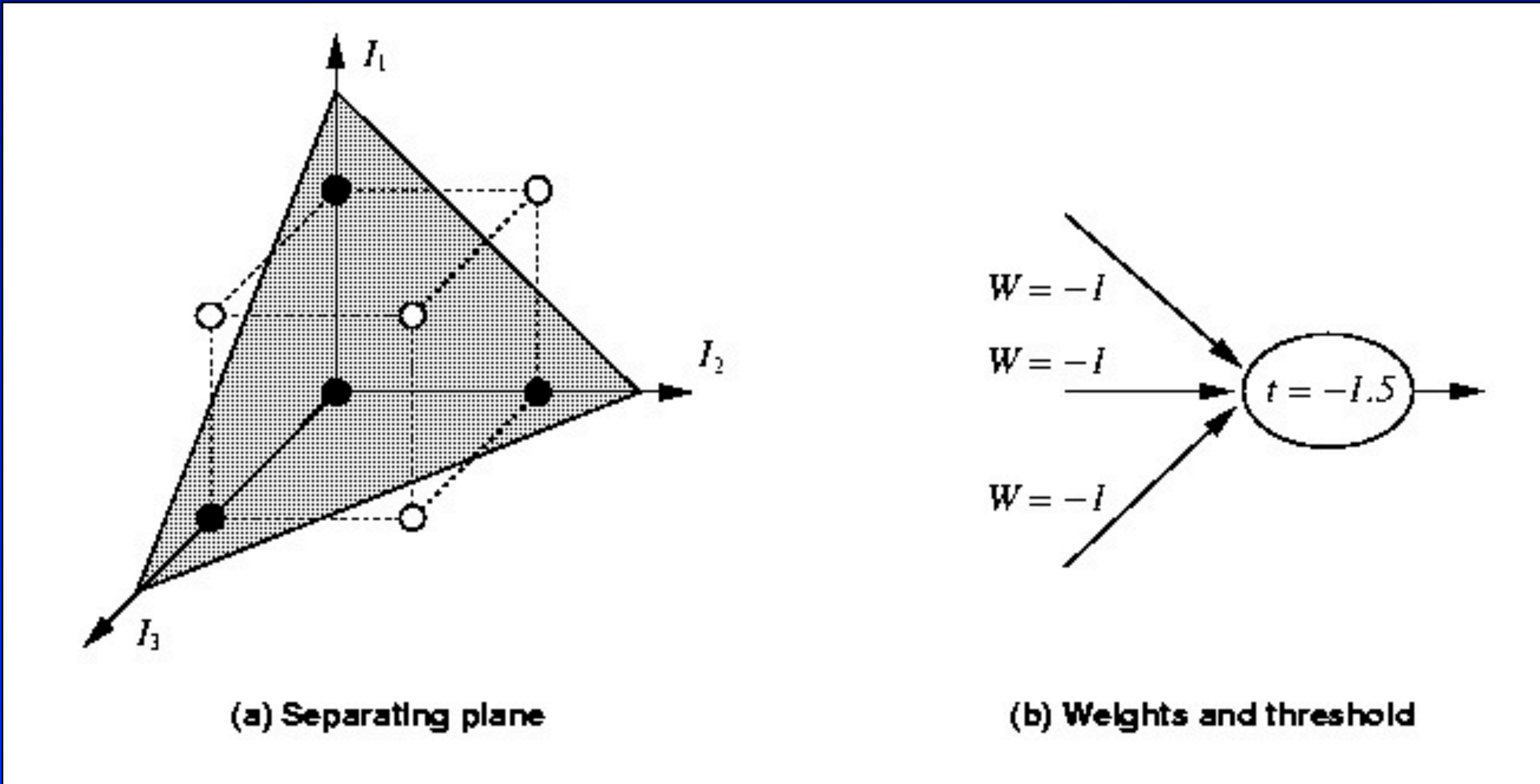
- single layer, feed-forward network
- historically one of the first types of neural networks
  - late 1950s
- the output is calculated as a step function applied to the weighted sum of inputs
- capable of learning simple functions
  - linearly separable

Perceptrons and Linear Separability



- perceptrons can deal with linearly separable functions
- some simple functions are *not* linearly separable
  - XOR function

Perceptrons and Linear Separability



[Russell & Norvig, 1995]

- linear separability can be extended to more than two dimensions
- more difficult to visualize

Perceptrons and Learning

- perceptrons can learn from examples through a simple learning rule
  - calculate the error of a unit Err_i as the difference between the correct output T_i and the calculated output O_i

    $$Err_i = T_i - O_i$$

  - adjust the weight W_j of the input I_j such that the error decreases

    $$W_{ij} := W_{ij} + \alpha * I_{ij} * Err_{ij}$$

    - α is the learning rate
  - this is a gradient descent search through the weight space
  - lead to great enthusiasm in the late 50s and early 60s until Minsky & Papert in 69 analyzed the class of representable functions and found the linear separability problem

Generic Neural Network Learning

- basic framework for learning in neural networks

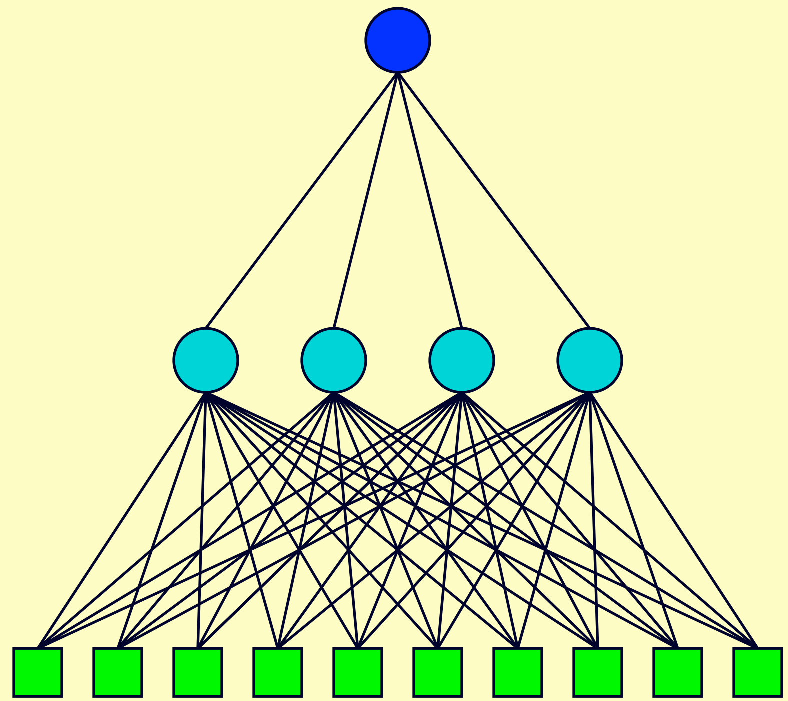
function NEURAL-NETWORK-LEARNING(*examples*) **returns** *network*
  network := a network with randomly assigned weights
  for each *e* **in** *examples* **do**
    O := NEURAL-NETWORK-OUTPUT(*network,e*)
    T := observed output values from *e*
    update the weights in *network* based on *e, O,* and *T*
return *network*

adjust the weights until the predicted output values *O* and the observed values *T* agree

Multi-Layer Networks

- research in the more complex networks with more than one layer was very limited until the 1980s
  - learning in such networks is much more complicated
  - the problem is to assign the blame for an error to the respective units and their weights in a constructive way
- the back-propagation learning algorithm can be used to facilitate learning in multi-layer networks

Diagram Multi-Layer Network



- two-layer network
  - input units I_k
    - usually not counted as a separate layer
  - hidden units a_j
  - output units O_i
- usually all nodes of one layer have weighted connections to all nodes of the next layer

Back-Propagation Algorithm

- assigns blame to individual units in the respective layers
  - essentially based on the connection strength
  - proceeds from the output layer to the hidden layer(s)
  - updates the weights of the units leading to the layer
- essentially performs gradient-descent search on the error surface
  - relatively simple since it relies only on local information from directly connected units
  - has convergence and efficiency problems

Capabilities of Multi-Layer Neural Networks

- expressiveness
  - weaker than predicate logic
  - good for continuous inputs and outputs
- computational efficiency
  - training time can be exponential in the number of inputs
  - depends critically on parameters like the learning rate
  - local minima are problematic
    - can be overcome by simulated annealing, at additional cost
- generalization
  - works reasonably well for some functions (classes of problems)
    - no formal characterization of these functions

Capabilities of Multi-Layer Neural Networks (cont.)

- sensitivity to noise
  - very tolerant
  - they perform nonlinear regression
- transparency
  - neural networks are essentially black boxes
  - there is no explanation or trace for a particular answer
  - tools for the analysis of networks are very limited
  - some limited methods to extract rules from networks
- prior knowledge
  - very difficult to integrate since the internal representation of the networks is not easily accessible

Applications

- domains and tasks where neural networks are successfully used
  - handwriting recognition
  - control problems
    - juggling, truck backup problem
  - series prediction
    - weather, financial forecasting
  - categorization
    - sorting of items (fruit, characters, phonemes, …)

Important Concepts and Terms

- axon
- back-propagation learning algorithm
- bias
- decision tree
- dendrite
- feedback
- function approximation
- generalization
- gradient descent
- hypothesis
- inductive learning
- learning element
- linear separability
- machine learning
- multi-layer neural network
- neural network
- neuron
- noise
- Ockham’s razor
- perceptron
- performance element
- prior knowledge
- sample
- synapse
- test set
- training set
- transparency

Chapter Summary

- learning is very important for agents to improve their decision-making process
  - unknown environments, changes, time constraints
- most methods rely on inductive learning
  - a function is approximated from sample input-output pairs
- decision trees are useful for learning deterministic Boolean functions
- neural networks consist of simple interconnected computational elements
- multi-layer feed-forward networks can learn any function
  - provided they have enough units and time to learn