Review – Biological Neuron

The Neuron - A Biological Information Processor

- *dentrites* - the receivers
- *soma* - neuron cell body (sums input signals)
- *axon* - the transmitter
- *synapse* - point of transmission
- neuron activates after a certain *threshold* is met

Learning occurs via electro-chemical changes in effectiveness of *synaptic junction*.

Review – Advantage of the Brain

Inherent Advantages of the Brain:

"distributed processing and representation"

- Parallel processing speeds
- Fault tolerance
- Graceful degradation
- Ability to generalize

OUTLINE

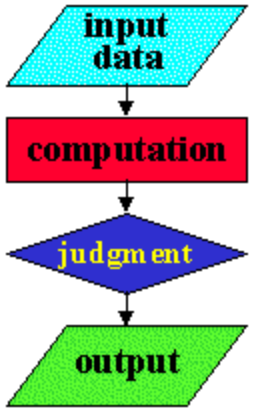
- Features of Neural Nets
- Learning in Neural Nets
- McCulloch/Pitts Neuron
- Hebbian Learning

Features of Neural Nets

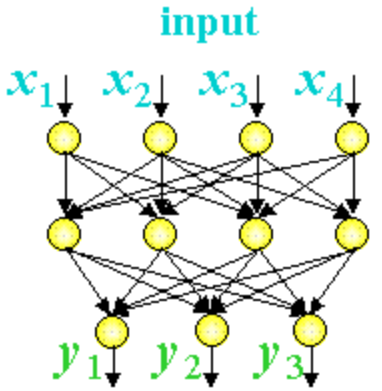
Neural Net Characteristics 1

- Neural nets have many unique features:

Massive Parallel Processing:



Sequential Information Processing (Computer)



Parallel Information Processing (Neural Network)

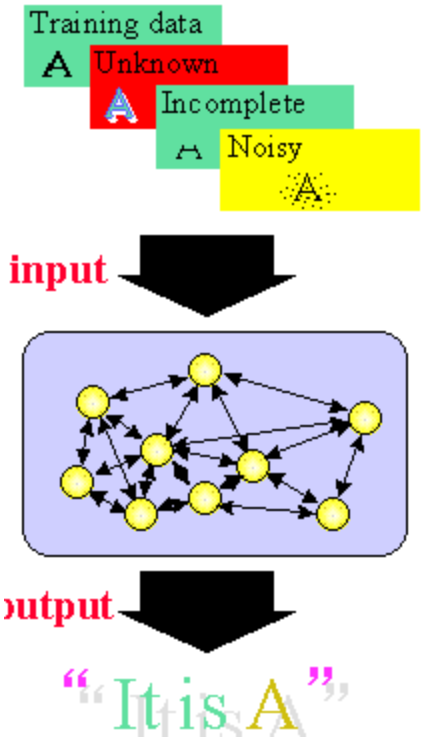
Neural Nets update in parallel

Computers calculate in sequence

Neural Net Characteristics 2

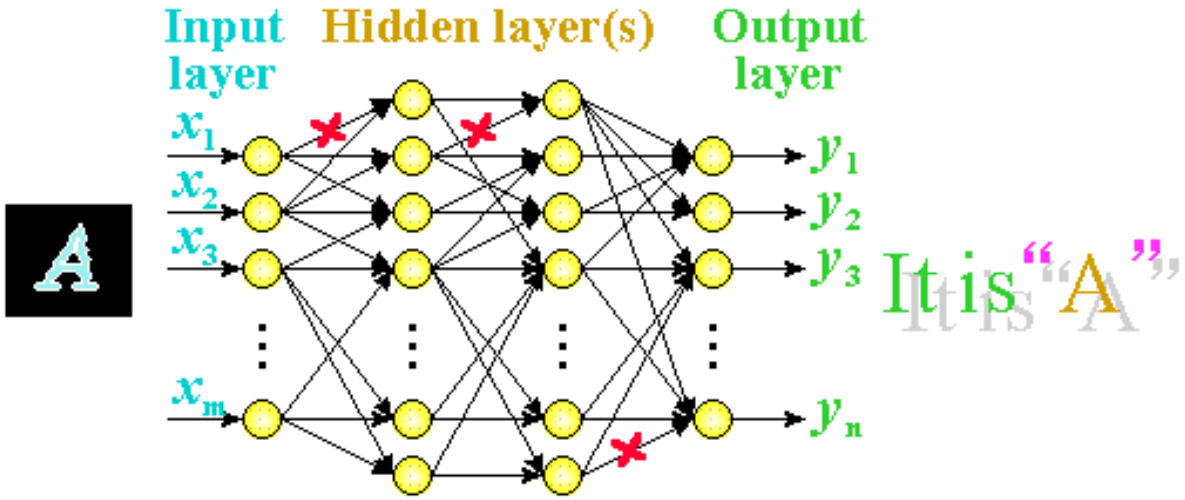
- Neural nets can learn:

Once a neural net is trained, it can handle unknown, incomplete or noisy data



Neural Net Characteristics 3

- Unlike computer memory, information is distributed over the entire network, so a failure in one part of the network does not prevent the system for producing the correct output.



Learning in Neural Nets

- 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

OUTLINE

- Introduction to Learning
- Learning Methods
- Introduction to Learning by Induction

Introduction to Learning

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.

Learning is . . .

- Learning involves the recognition of patterns in data or experience and then using that information to improve performance on another task

Implications

- Learning involves 3 factors:

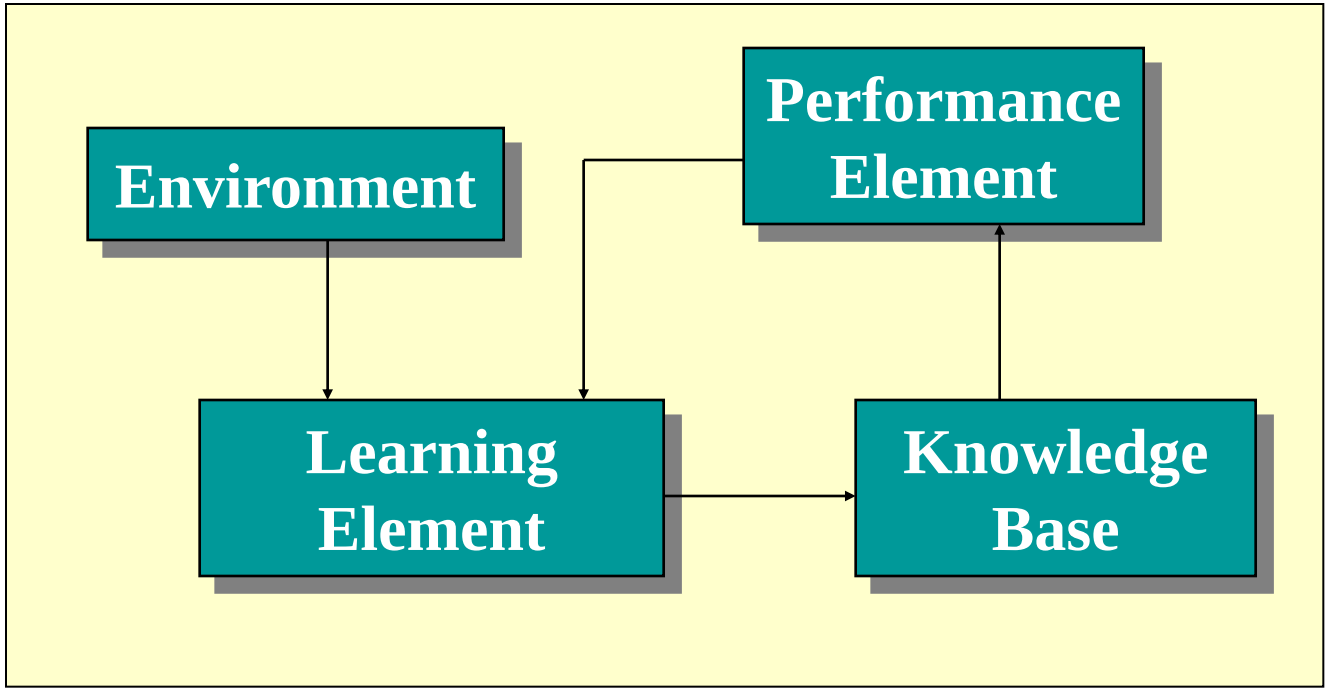
changes	**Learning changes the learner: for machine learning the problem is determining the nature of these changes and how to best represent them**
generalization	**Learning leads to generalization: performance must improve not only on the same task but on similar tasks**
improvement	**Learning leads to improvements: machine learning must address the possibility that changes may degrade performance and find ways to prevent it.**

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.

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

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

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

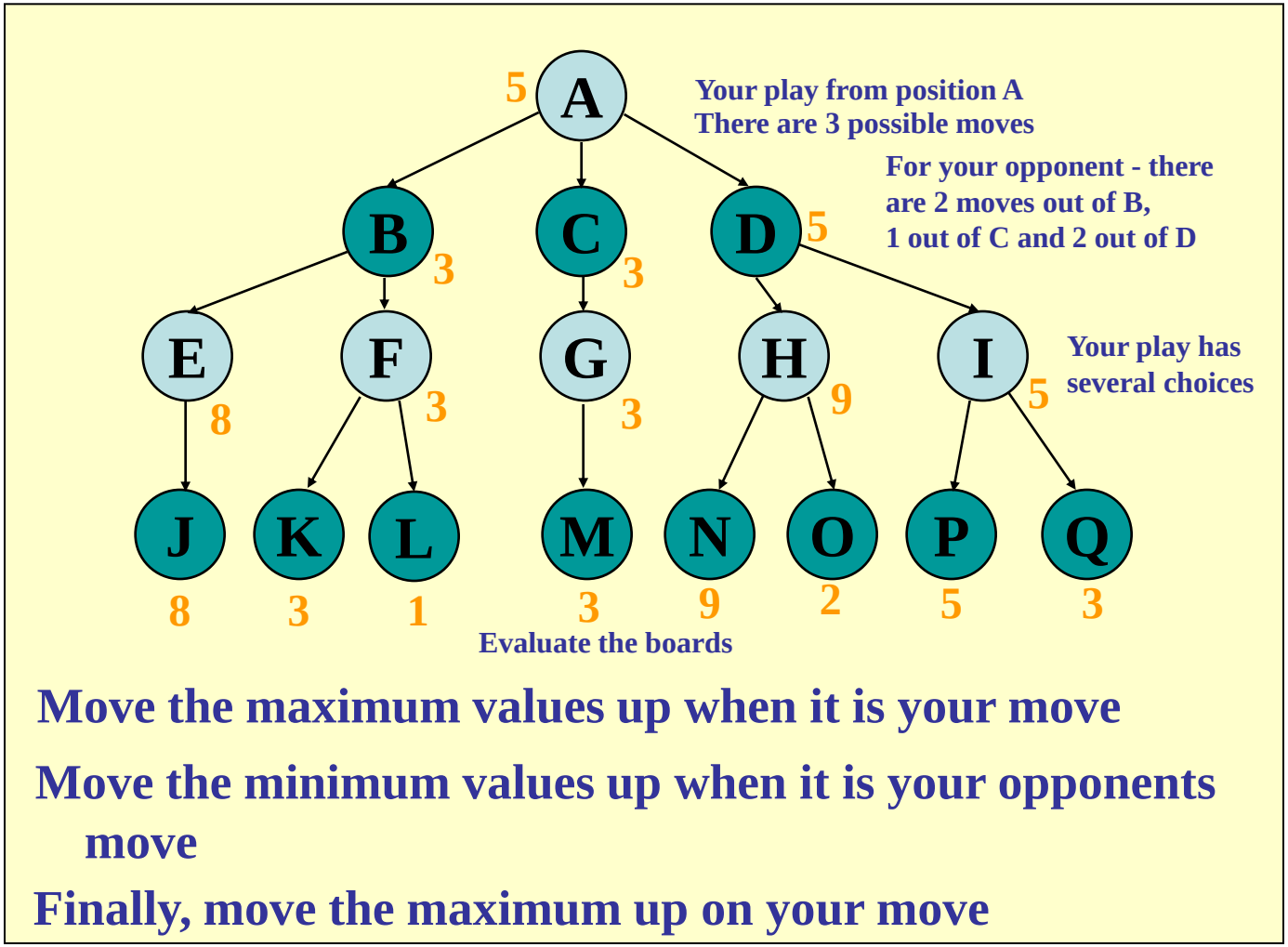
- Memorization - saving new knowledge to be retrieved when needed rather than calculated
- Works by taking problems that the performance element has solved and memorizing the problem and the solution
- Only useful if it takes less time to retrieve the knowledge than it does to recompute it

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

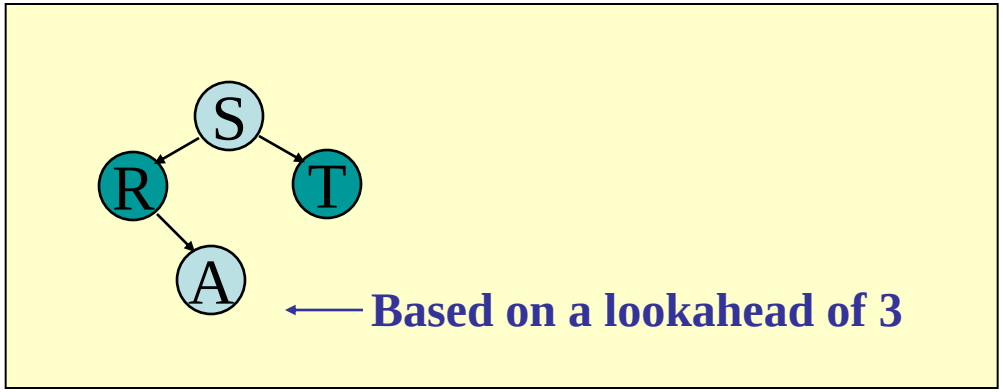
MinMax Game Tree

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



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 . . .



Introduction to Learning by Induction

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.

Possible Quiz

What is inductive learning?

How does rote learning work?

What is ID3?

Summary

- **Introduction to Learning**
- **Learning Methods**
- **Introduction to Learning by Induction**

Review – 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

Review – Example of Learning

- User is observed to file email messages as follows:-

Message	Domain	Size	Type	Rank
No.1	Internal	Small	Personal	**Important**
No.2	Internal	Medium	Mailing List	**Normal**
No.3	Internal	Small	Personal	**Important**
No.4	World	Small	Personal	**Important**
No.5	World	Large	Mailing List	**Normal**
No.6	IE	Small	Mailing List	**Normal**
No.7	IE	Small	Personal	**Important**
No.8	World	Large	Mailing List	**Normal**
No.9	IE	Large	Personal	**Normal**
No.10	IE	Medium	Personal	**Normal**

Learn users behaviour from examples in order to automate this process.

OUTLINE

- Background
- Decision Trees
- ID3

Background

- We are interested in datasets consisting of a number of *records*
- Each record has:
  - A set of *attribute-value pairs*
    E.g. “Color: green, Temperature: 120^{o}C”
  - A *classification*
    E.g. “Risk: Low”
- We assume that the classification is *dependent* on the attributes
  - We are assuming that the dataset “makes sense”

Inductive Learning

- Interesting problem: Can we find the mapping between the attributes and the classifications only using the dataset?
- This is the purpose of an *inductive learning algorithm*
- In practice, we can only find an *approximation* to the true mapping
  - Only a hypothesis, but still useful:
    - Classification of new records (prediction)
    - Identifying patterns in the dataset (data mining)

Classification Problem

- We model each record as a pair (x, y), where x is the set of attributes and y is the classification
- We are trying to find a function f which models the data:
  - We want to find f such that $f(x) = y$ for as many records in the dataset as possible
- To make things easier, we will use a fixed set of attributes and discrete classes

Choosing a Classifier

- Some issue to consider:
  - Construction time
  - Runtime performance
  - Transparency
  - Updateability
  - Scalability

- All these rest on how *f* is represented by the model

Common Classifiers

- Analytical (mathematical) models
- Nearest-neighbor classifiers
- Neural networks
- Genetic programming
- Genetic algorithms (classifier systems)
- Decision trees

Decision Trees

- What is a Decision Tree?
  - it takes as input the description of a situation as a set of attributes (features) and outputs a yes/no decision (so it represents a Boolean function)
  - each leaf is labeled "positive” or "negative", each node is labeled with an attribute (or feature), and each edge is labeled with a value for the feature of its parent node

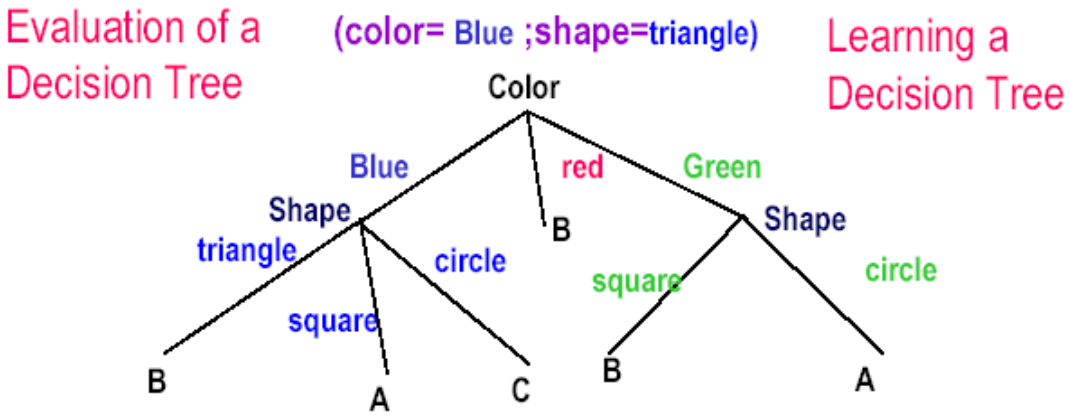
- Attribute-value language for examples
  - in many inductive tasks, especially learning decision trees, we need a representation language for examples
    - each example is a finite feature vector
    - a concept is a decision tree where nodes are features

History

- Independently developed in the 60's and 70's by researchers in ...
  - Statistics: L. Breiman & J. Friedman - CART (Classification and Regression Trees)
  - Pattern Recognition: Uof Michigan - AID, G.V. Kass - CHAID (Chi-squared Automated Interaction Detection)
  - AI and Info. Theory: R. Quinlan - ID3, C4.5 (Iterative Dichotomizer)

Decision Tree Representation

- Decision Trees are classifiers for instances represented as features vectors. (color= ;shape= ;label=)
  - Nodes are tests for feature values;
  - There is one branch for each value of the feature
  - Leaves specify the categories (labels)
  - Can categorize instances into multiple disjoint categories



Example

- Problem: How do you decide to leave a restaurant?

- Variables:

Alternative	Whether there is a suitable alternative restaurant nearby.
Bar	Is there a comfortable bar area?
Fri/Sat	True on Friday or Saturday nights.
Hungary	How hungry is the subject?
Patrons	How many people in the restaurant?
Price	Price range.
Raining	Is it raining outside?
Reservation	Does the subject have a reservation?
Type	Type of Restaurant.
Stay?	Stay or Go

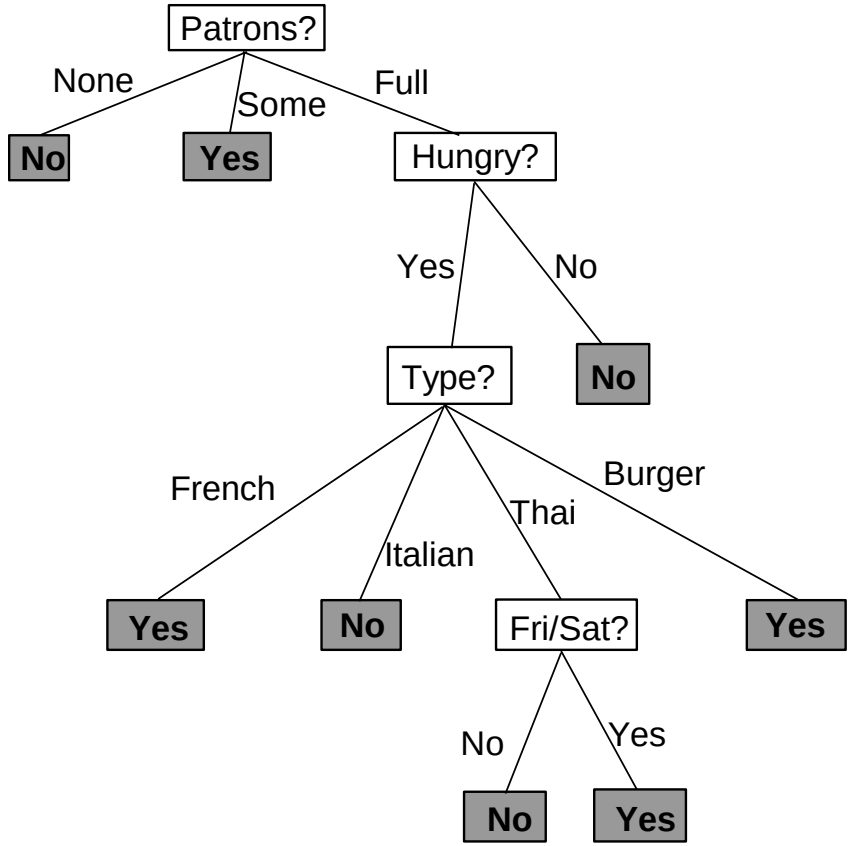
Method

- Gather test data using the variables and observe the decision.

Case	Alt.	Bar	Fri.	Hun	Pat	Price	Rain	Res	Type	Est.	Stay?
X1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
X2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
X3	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
X4	Yes	No	Yes	Yes	Full	$	No	No	Thai	Oct-30	Yes
X5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No
X6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	Yes
X7	No	Yes	No	No	None	$	Yes	No	Burger	0-10	No
X8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0-10	Yes
X9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No
X10	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	Oct-30	No
X11	No	No	No	No	None	$	No	No	Thai	0-10	No
X12	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes

Result

- **Very good Decision Tree**
  - Classifies all examples correctly
  - Very few nodes



How?

Given: Set of examples with Pos. & Neg. classes

Problem: Generate a Decision Tree model to classify a separate (validation) set of examples with minimal error

Approach: *Occam's Razor* - produce the simplest model that is consistent with the training examples -> narrow, short tree. Every traverse should be as short as possible

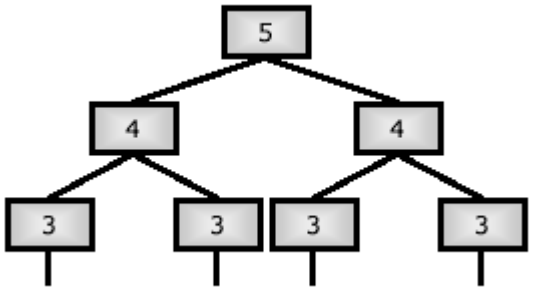
Formally: Finding the absolute simplest tree is intractable, but we can at least try our best

Exhaustive Search

- We could try to build every possible tree and the select the best one

- How many trees are there for a specific problem?
  - Given a dataset with 5 different binary attributes
  - How many trees are there in this case?

Answer



- The resulting tree must contain at least 1 node, and can be at most 5 levels deep
  - We can choose any of the 5 attributes at the root node
  - In the next level, we have 4 attributes to chose from, for each branch
  - This gives 5 x (4 x 4) x (3 x 3 x 3 x 3) x … x 1
  - This is 5 x 42 x 34 x 28 x 1 = 1658880 (for a tree of depth 5)
  - The total is 1659471 for all possible trees (depths 5,4,3,2&1)

Producing the Best DT

Heuristic (strategy) 1: Grow the tree from the top down. Place the *most important variable* test at the root of each successive subtree

The *most important variable:*

- the variable (predictor) that gains the most ground in classifying the set of training examples
- the variable that has the most significant relationship to the response variable
- to which the response is most dependent or least independent

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 *i*th symbol has probability p_i of appearing, the entropy of the message is:

$$H = -\sum_{i=1}^{n} (p_i \log_2 p_i)$$

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?

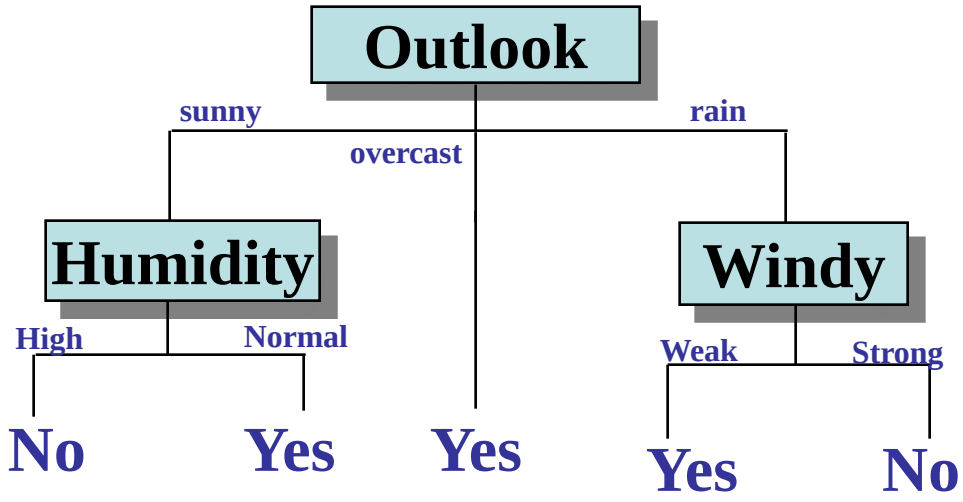
Constructing the Decision Tree

- 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	

Constructing the Next Level

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



Finished - what happened to Temp?

For the Sunny cases only:

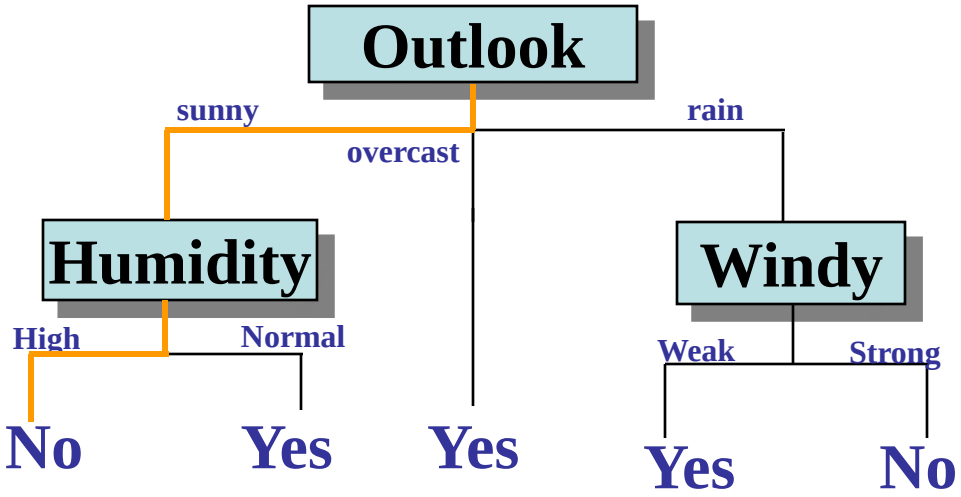
Temp	Hot	2	0	
	Mild	1	1	2.0
	Cool	0	1	
Humidity	High	3	0	0
	Low	0	2	
Windy	Weak	2	1	4.75
	Strong	1	1	

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

Possible Quiz

What is the purpose of an inductive learning algorithm?

What is Entropy?

What variable is placed at the root node of a Decision Tree?

Summary

- **Background**
- **Decision Trees**
- **ID3**

Review – Decision Trees

- What is a Decision Tree?
  - it takes as input the description of a situation as a set of attributes (features) and outputs a yes/no decision (so it represents a Boolean function)
  - each leaf is labeled "positive” or "negative", each node is labeled with an attribute (or feature), and each edge is labeled with a value for the feature of its parent node
- ID3 is one example of an algorithm that will create a Decision Tree

Review – Advantages

- Proven modeling method for 20 years
- Provides explanation and prediction
- Useful for non-linear mappings
- Generalizes well given sufficient examples
- Rapid training and recognition speed
- Has inspired many inductive learning algorithms using statistical regression

Review - Disadvantages

- Only one response variable at a time
- Different significance tests required for nominal and continuous responses
- Discriminate functions are often suboptimal due to orthogonal decision hyperplanes
- No proof of ability to learn arbitrary functions
- Can have difficulties with noisey data

OUTLINE

•**Overfitting & Pruning, Constraints, Rules**

Overfitting & Pruning

Overfitting

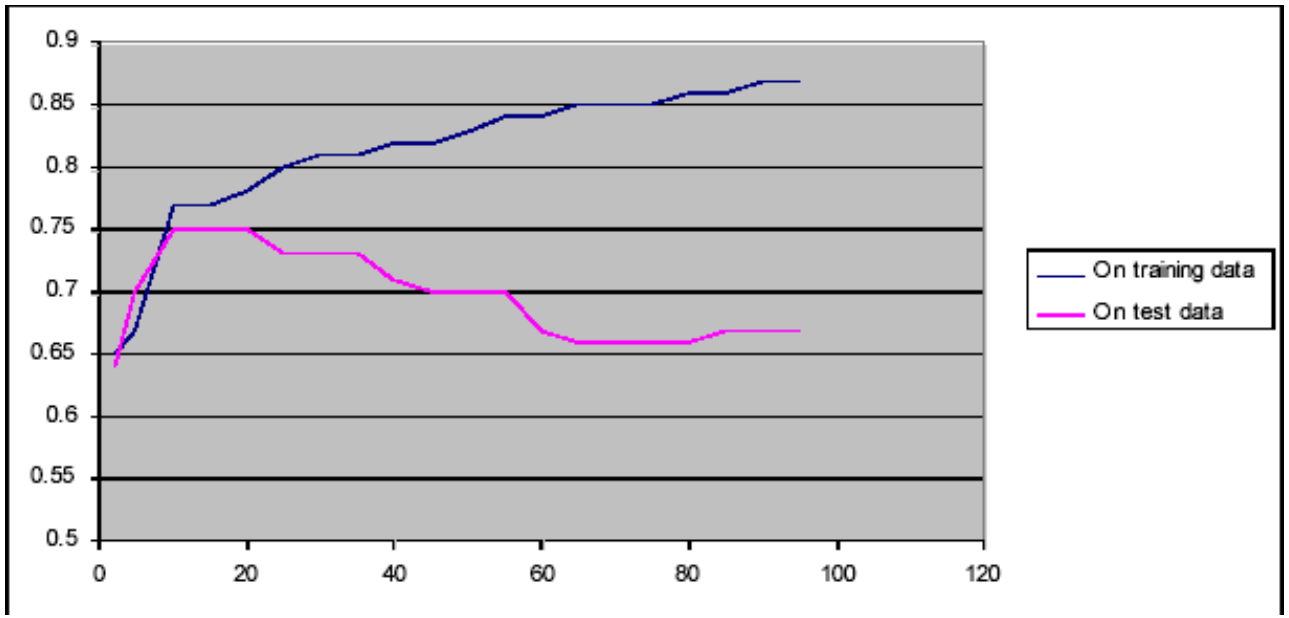
- **A tree generated may over-fit the training examples due to noise or too small a set of training data**

- **Two approaches to avoid over-fitting:**
  - **(Stop earlier): Stop growing the tree earlier**
  - **(Post-prune): Allow over-fit and then post-prune the tree**

- **Approaches to determine the correct final tree size:**
  - **Separate training and testing sets or use cross-validation**
  - **Use all the data for training, but apply a statistical test (e.g., chi-square) to estimate whether expanding or pruning a node may improve over entire distribution**
  - **Use Minimum Description Length (MDL) principle: halting growth of the tree when the encoding is minimized.**

Overfitting Effects

- The effect of overfitting is that it produces a tree that works very well on the training set but produces a lot of errors on the test set



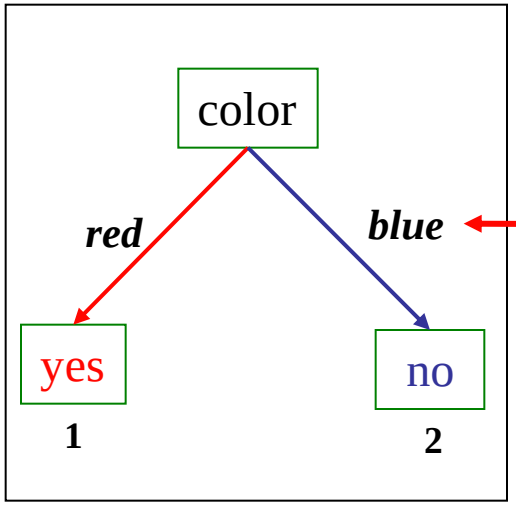
Accuracy on training and test data

Avoiding Overfitting

- Two basic approaches
  - Prepruning: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Postpruning: Grow the full tree and then remove nodes that seem not to have sufficient evidence.
- Methods for evaluating subtrees to prune:
  - Cross-validation: Reserve hold-out set to evaluate utility
  - Statistical testing: Test if the observed regularity can be dismissed as likely to be occur by chance
  - Maximum Description Length: Is the additional complexity of the hypothesis smaller than remembering the exceptions

Pruning 1

- Subtree Replacement: merge a subtree into a leaf node
  - Using a set of data different from the training data
  - At a tree node, if the accuracy without splitting is higher than the accuracy with splitting, replace the subtree with a leaf node; label it using the majority class



Suppose with test set we find 3 red "no" examples, and 1 blue "yes" example. We can replace the tree with a single "no" node. After replacement there will be only 2 errors instead of 5.

Pruning 2

- A post-pruning, cross validation approach
  - Partition training data into “grow” set and “validation” set.
  - Build a complete tree for the “grow” data
  - Until accuracy on validation set decreases, do:
    - For each non-leaf node in the tree
    - Temporarily prune the tree below; replace it by majority vote.
    - Test the accuracy of the hypothesis on the validation set
    - Permanently prune the node with the greatest increase in accuracy on the validation test

Constraints

Motivation

- Necessity of Constraints
  - Decision tree can be
    - Complex with hundreds or thousands of nodes
    - Difficult to comprehend
  - Users are only interested in obtaining an overview of the patterns in their data
    - A simple, comprehensible, but only approximate decision tree is much more useful
- Necessity of Pushing constraints into the tree-building phase
  - In tree-building phase, may waste I/O on building tree which will be pruned by applying constraints

Possible Constraints

- **Constraints**
  - **Size** : the number of nodes
    - For a given k, find a subtree with size at most k that minimizes either the total MDL cost or the total number of misclassified records.
  - **Inaccuracy** : the total MDL cost or the number of misclassified records
    - For a given C, find a smallest subtree whose total MDL cost or the total number of misclassified records does not exceed C.

Possible Algorithm

- Input
  - Decision tree generated by traditional algorithm
  - Size constraint : k

- Algorithm
  - Compute minimum MDL cost recursively

    minCost = the MDL cost when the root become a leaf.

    For k_1 = 1 to k–2 {
    minCost = Min (minCost, minimum MDL cost whose children have constraint k_1, k – 1 – k_1 for each)

    Delete all nodes which are not in the minimum cost subtree

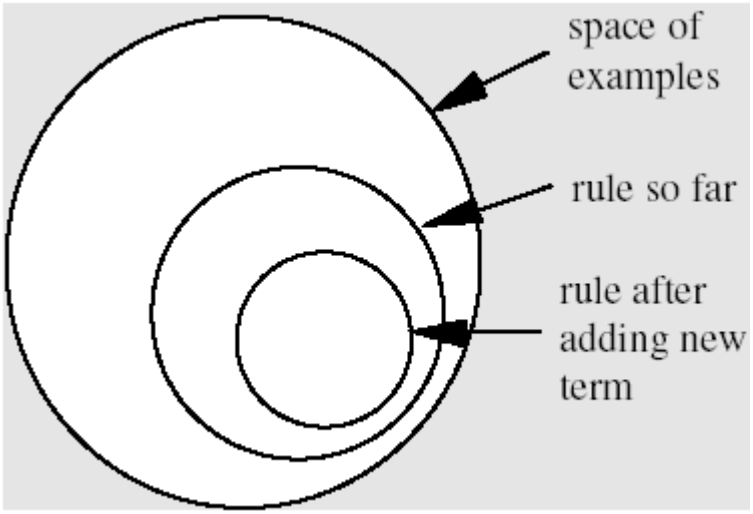
RULES

DT to Rules

- A Decision Tree can be converted into a rule set
  - Straightforward conversion: rule set overly complex
  - More effective conversions are not trivial
- Strategy for generating a rule set directly: for each class in turn find rule set that covers all instances in it (excluding instances not in the class)
- This approach is called a covering approach because at each stage a rule is identified that covers some of the instances

A Simple Covering Algorithm

- Generates a rule by adding tests that maximize rule's accuracy
- Similar to situation in decision trees: problem of selecting an attribute to split on
- Each new test reduces the rule's coverage:



Selecting a test

- Goal: maximizing accuracy
  - t: total number of instances covered by rule
  - p: positive examples of the class covered by rule
  - t-p: number of errors made by rule
- Select test that maximizes the ratio p/t
- We are finished when p/t = 1 or the set of instances can’t be split any further

Example: contact lenses data

- Rule we seek: If ? then recommendation = hard

- Possible tests:
  - Tear production rate = Normal 2/8
  - Tear production rate = Reduced 0/12
  - Astigmatism = yes 4/12
  - Astigmatism = no 0/12
  - Spectacle prescription = Hypermetrope 1/12
  - Spectacle prescription = Myope 3/12
  - Age = Presbyopic 1/8
  - Age = Pre-presbyopic 1/8
  - Age = Young 2/8

Best case

Out of 12 cases with Astigmatism 4 had hard lenses

Modified Rule

- Rule with best test added:
  - If astigmatism = yes then recommendation = hard

- Instances covered by modified rule:

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

This is really not a very good rule

Further Refinement

- Current state: If astigmatism = yes and ? then recommendation = hard

- Possible tests:
  - Tear production rate = Normal 4/6
  - Tear production rate = Reduced 0/6
  - Spectacle prescription = Hypermetrope 1/6
  - Spectacle prescription = Myope 3/6
  - Age = Presbyopic 1/4
  - Age = Pre-presbyopic 1/4
  - Age = Young 2/4

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses	
Young	Myope	Yes	Reduced	None	
Young	Myope	Yes	Normal	Hard	X
Young	Hypermetrope	Yes	Reduced	None	
Young	Hypermetrope	Yes	Normal	hard	X
Pre-presbyopic	Myope	Yes	Reduced	None	
Pre-presbyopic	Myope	Yes	Normal	Hard	X
Pre-presbyopic	Hypermetrope	Yes	Reduced	None	
Pre-presbyopic	Hypermetrope	Yes	Normal	None	
Presbyopic	Myope	Yes	Reduced	None	
Presbyopic	Myope	Yes	Normal	Hard	X
Presbyopic	Hypermetrope	Yes	Reduced	None	
Presbyopic	Hypermetrope	Yes	Normal	None	

New Rule

- Rule with best test added:
  - If astigmatism = yes and tear production rate = normal then recommendation = hard

- Instances covered by modified rule:

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Normal	None

Further Refinement

- Current state: If astigmatism = yes and tear production rate = normal and ? then recommendation = hard

- Possible tests:
  - Spectacle prescription = Hypermetrope 1/3
  - Spectacle prescription = Myope 3/3
  - Age = Presbyopic 1/2
  - Age = Pre-presbyopic 1/2
  - Age = Young 2/2

- Tie between the second and the fifth test
  - We choose the one with greater coverage

Result

- Final rule:
  - If astigmatism = yes and tear production rate = normal and spectacle prescription = myope then recommendation = hard
- Second rule for recommending hard lenses: (built from instances not covered by first rule)
  - If age = young and astigmatism = yes and tear production rate = normal then recommendation = hard
- These two rules cover all hard lenses:
  - Process is repeated with other two classes

Possible Quiz

What is overfitting?

What is pruning?

Name one possible constraint.

SUMMARY

- **Overfitting & Pruning**
- **Constraints**
- **Rules**

Supervised Learning

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

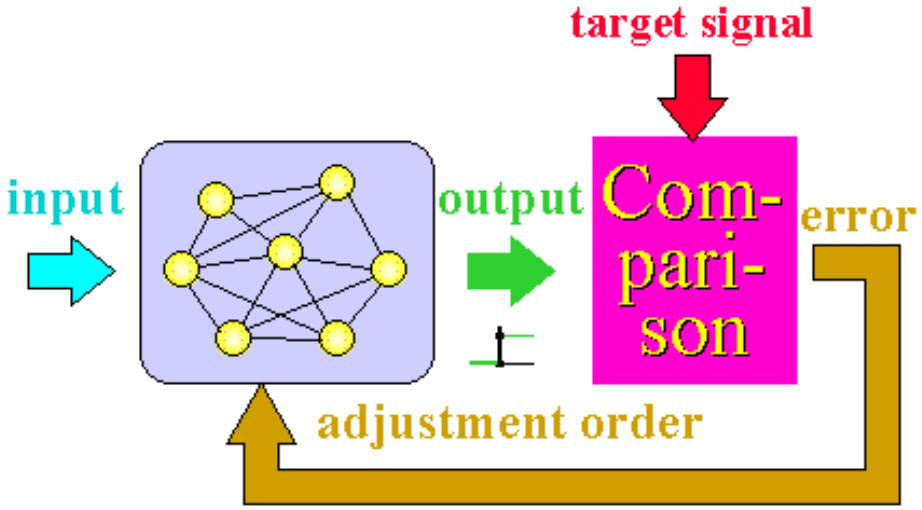
- Then, the neural network tries to produce an output as close as possible to the target signal by adjusting the values of internal weights.

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

Error Correction Method

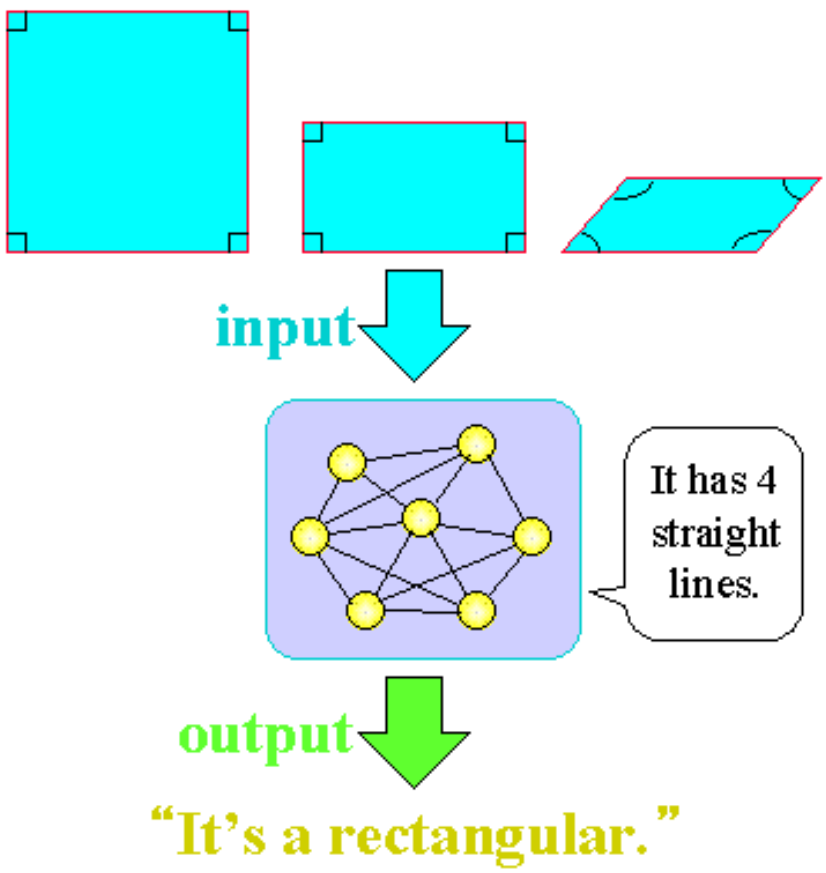
- Error correction method is used for networks which their neurons have discrete output functions.

- 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.

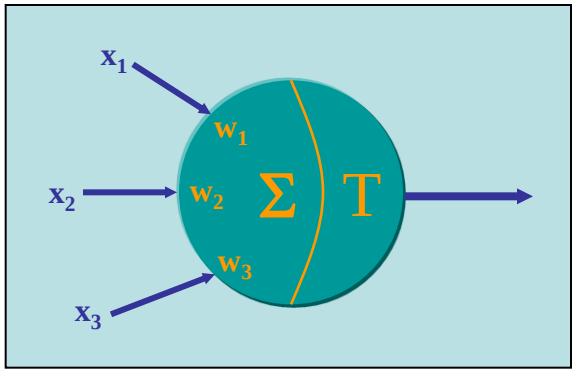


Common NN Features

- “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?

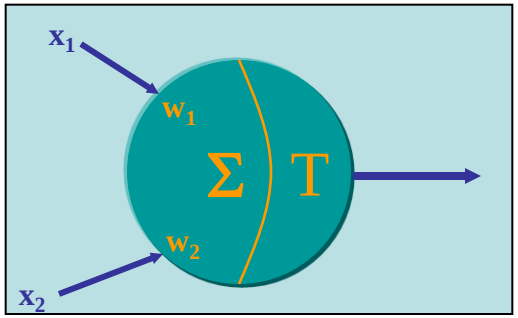
PROBLEM: find the threshold, T, and the weights w_1 and w_2 where:

F is 1 if $x_1w_1 + x_2w_2 > T$

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

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

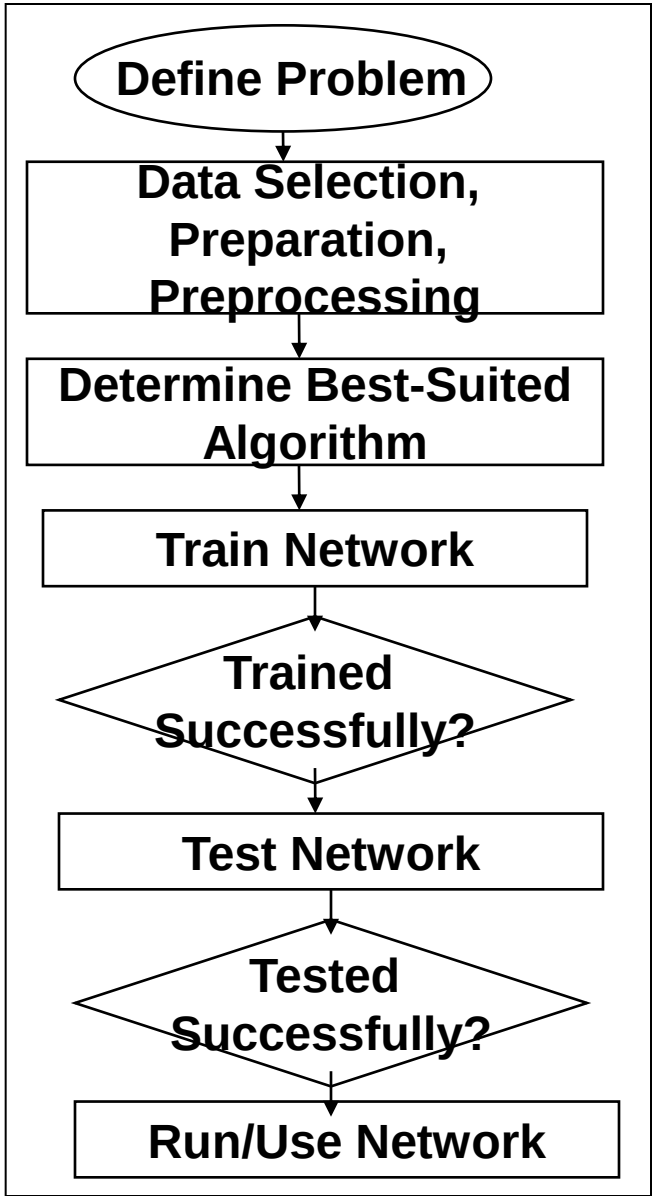
$w_1 > T$ **$w_1 + w_2 > T$**



w_1 and w_2 = 0.7 T = 0.5 works

NN Development

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



Learning in Neural Nets (again)

- The operation of a neural network is determined by the values of the interconnection weights
  - There is no algorithm that determines how the weights should be assigned in order to solve a specific problems
  - Hence, the weights are determined by a learning process

Hebbian Learning

- The oldest and most famous of all learning rules is Hebb's postulate of learning:

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B is increased"

Application to artificial neurons:

If two interconnected neurons are both "on" at the same time, then the weight between them should be increased

Hebb's Algorithm

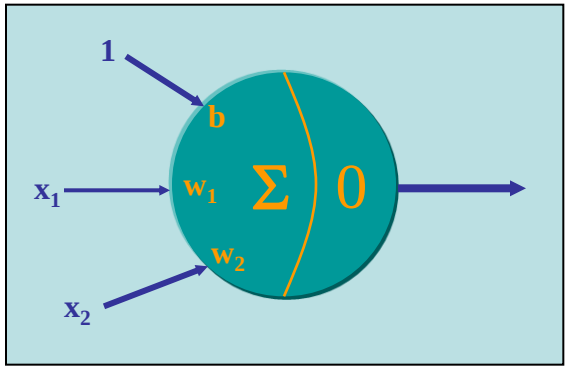
- **Step 0**: initialize all weights to 0
- **Step 1**: Given a training input, s, with its target output, t, set the activations of the input units: $x_i = s_i$
- **Step 2**: Set the activation of the output unit to the target value: $y = t$
- **Step 3**: Adjust the weights: $w_i(new) = w_i(old) + x_i y$
- **Step 4**: Adjust the bias (just like the weights): $b(new) = b(old) + y$

Example

- **PROBLEM**: Construct a Hebb Net which performs like an AND function, that is, only when both features are “active” will the data be in the target class.

- **TRAINING SET** (with the bias input always at 1):

x1	x2	bias	Target
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1



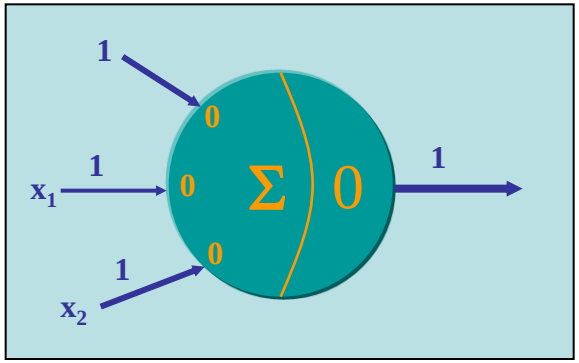
Training – First Input

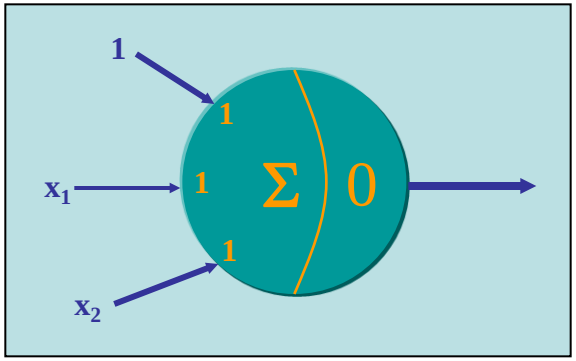
- Initialize the weights to 0

Present the first input (1 1 1) with a target of 1

Update the weights:

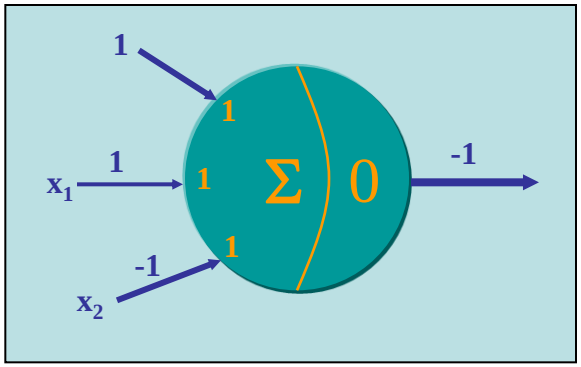
$$w_1(new) = w_1(old) + x_1 t$$
$$= 0 + 1 = 1$$
$$w_2(new) = w_2(old) + x_2 t$$
$$= 0 + 1 = 1$$
$$b(new) = b(old) + t$$
$$= 0 + 1 = 1$$





Training – Second Input

- Present the second input: (1 -1 1) with a target of -1



Update the weights:

$w_1(new) = w_1(old) + x_1 t$

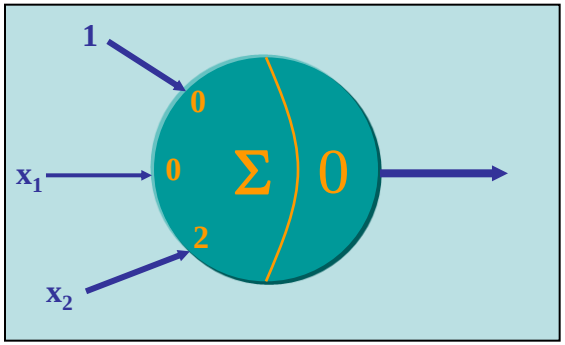
$= 1 + 1(-1) = 0$

$w_2(new) = w_2(old) + x_2 t$

$= 1 + (-1)(-1) = 2$

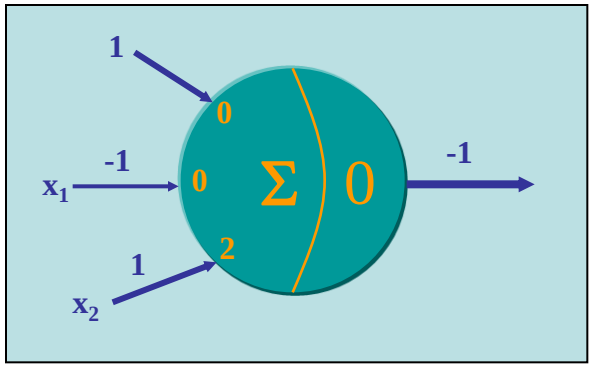
$b(new) = b(old) + t$

$= 1 + (-1) = 0$



Training – Third Input

- Present the third input: (-1 1 1) with a target of -1



Update the weights:

$w_1(new) = w_1(old) + x_1 t$

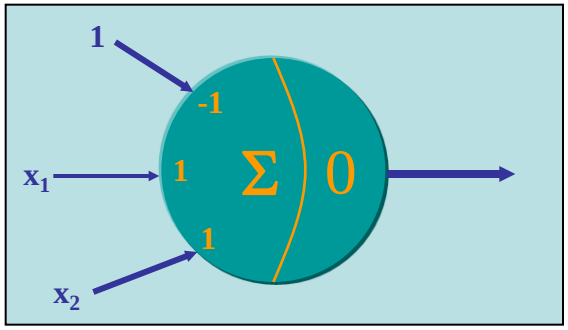
$= 0 + (-1)(-1) = 1$

$w_2(new) = w_2(old) + x_2 t$

$= 2 + 1(-1) = 1$

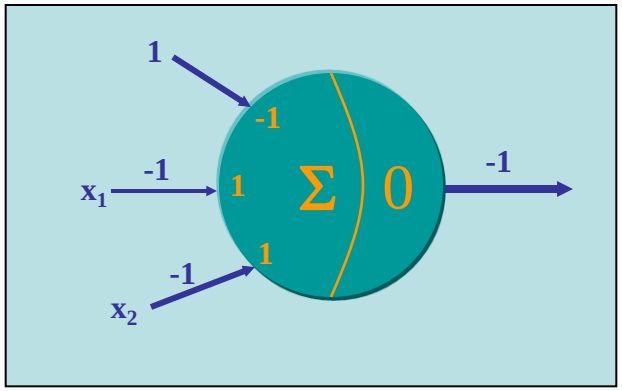
$b(new) = b(old) + t$

$= 0 + (-1) = -1$



Training – Fourth Input

- Present the fourth input: (-1 -1 1) with a target of -1



Update the weights:

$w_1(new) = w_1(old) + x_1 t$

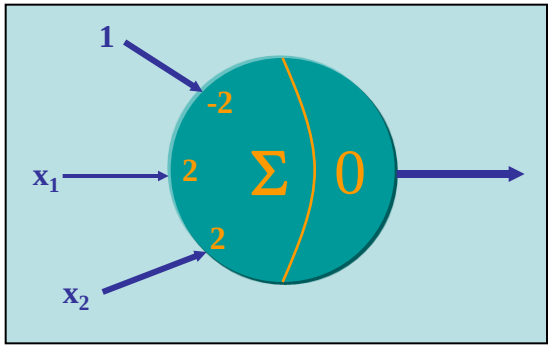
$= 1 + (-1)(-1) = 2$

$w_2(new) = w_2(old) + x_2 t$

$= 1 + (-1)(-1) = 1$

$b(new) = b(old) + t$

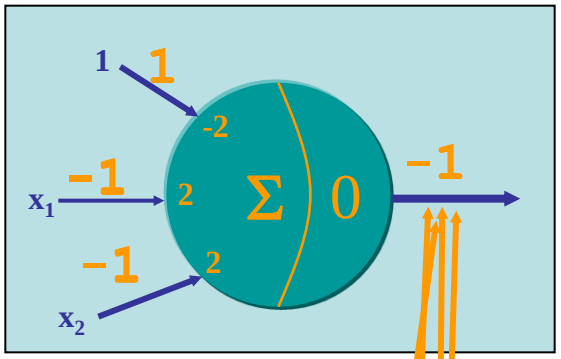
$= -1 + (-1) = -2$



Final Neuron

- This neuron works:

x1	x2	bias	Target
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1



1*2 + 1*2 + 1*(-2) = 2 > 0

(-1)*2 + 1*2 +1*(-2) = -2 < 0

1*2 + (-1)*2 + 1*(-2) = -2 < 0

(-1)*2 + (-1)*2 + 1 *(-2) = -6 <0

Possible Quiz

What is the difference between supervised and unsupervised learning?

What is the principle behind Hebbian learning?

Describe a MP Neuron.

SUMMARY

Features of Neural Nets

Learning in Neural Nets

McCulloch/Pitts Neuron

Hebbian Learning