Machine Learning

Professor Khaled Rasheed

Lecture Notes

Introduction

What is Machine Learning?

Simon Says:

"Learning denotes changes in the system that enable the system to do the same task ... more effectively next time."

Example Tasks:

- Classify an object as an instance (or non-instance) of a general concept. (Inductive Concept Learning.)
- Solve a search problem. (Speedup Learning.)

Types of Learning

- Learning by being told:

  Teacher states the rules of grammar.

  Knowledge Compilation.

- Learning from examples:

  Teacher shows examples of good and bad sentences.

  Inductive Learning.

- Learning by discovery:

  Student talks and teacher corrects his grammar.

  Automated Theory Formation.

Inductive Argument

Given:

$$
\begin{array}{lll}
P(a_1) \wedge Q(a_1) & \neg P(b_1) \wedge \neg Q(b_1) & \neg P(c_1) \wedge Q(c_1) \\
P(a_2) \wedge Q(a_2) & \neg P(b_2) \wedge \neg Q(b_2) & \neg P(c_2) \wedge Q(c_2) \\
\ldots & & \\
P(a_l) \wedge Q(a_l) & \neg P(b_m) \wedge \neg Q(b_m) & \neg P(c_n) \wedge Q(c_n)
\end{array}
$$

Conclude:

$$(\forall x) P(x) \Rightarrow Q(x)$$

The Old Problem of Induction

- Why are inductive arguments justified?
- Hume: Because they have worked in the past.
- Russell: That's using induction to justify induction.
- Goodman: Inductive arguments are not justified.

The New Problem of Induction

How does an intelligent agent choose among many possible inductive generalizations of his observations?

Goodman's Paradox

Examples:

$Emerald(a) \wedge Green(a)$ (Sunday).

$Emerald(b) \wedge Green(b)$ (Monday).

$Emerald(c) \wedge Green(c)$ (Tuesday).

Generalizations:

$(\forall x) Emerald(x) \Rightarrow Green(x)$

$(\forall x) Emerald(x) \Rightarrow Grue(x)$

An object is "grue" if it is green on Sunday, Monday and Tuesday, but is blue for the rest of the week.

Inductive Bias

"Any criteria for choosing one concept description over another, other than strict consistency with the training examples." (Mitchell)

- E.g., A biased concept description language.
- E.g., A biased learning algorithm.

Occam's Razor

- Choose the simplest hypothesis.
- That accounts for the observation.
- "Green" is simpler than "Grue"
- So choose a hypothesis involving the term "Green".

Concept Learning Problem Definition

- Given Training Data:
  - Positive Examples:
    $(ObjectDescription, +Label)$
  - Negative Examples:
    $(ObjectDescription, -Label)$
- Find rule for predicting whether future examples are positive or negative.
- "Concept Membership Rule".

Definitions

- Instance Description Language:
  - Language for describing example objects.
  - E.g., Boolean, Integer or Real Vector.
- Concept Description Language:
  - Language for describing concepts.
  - E.g., Conjunctive: $\wedge(v_i = k_i)$.

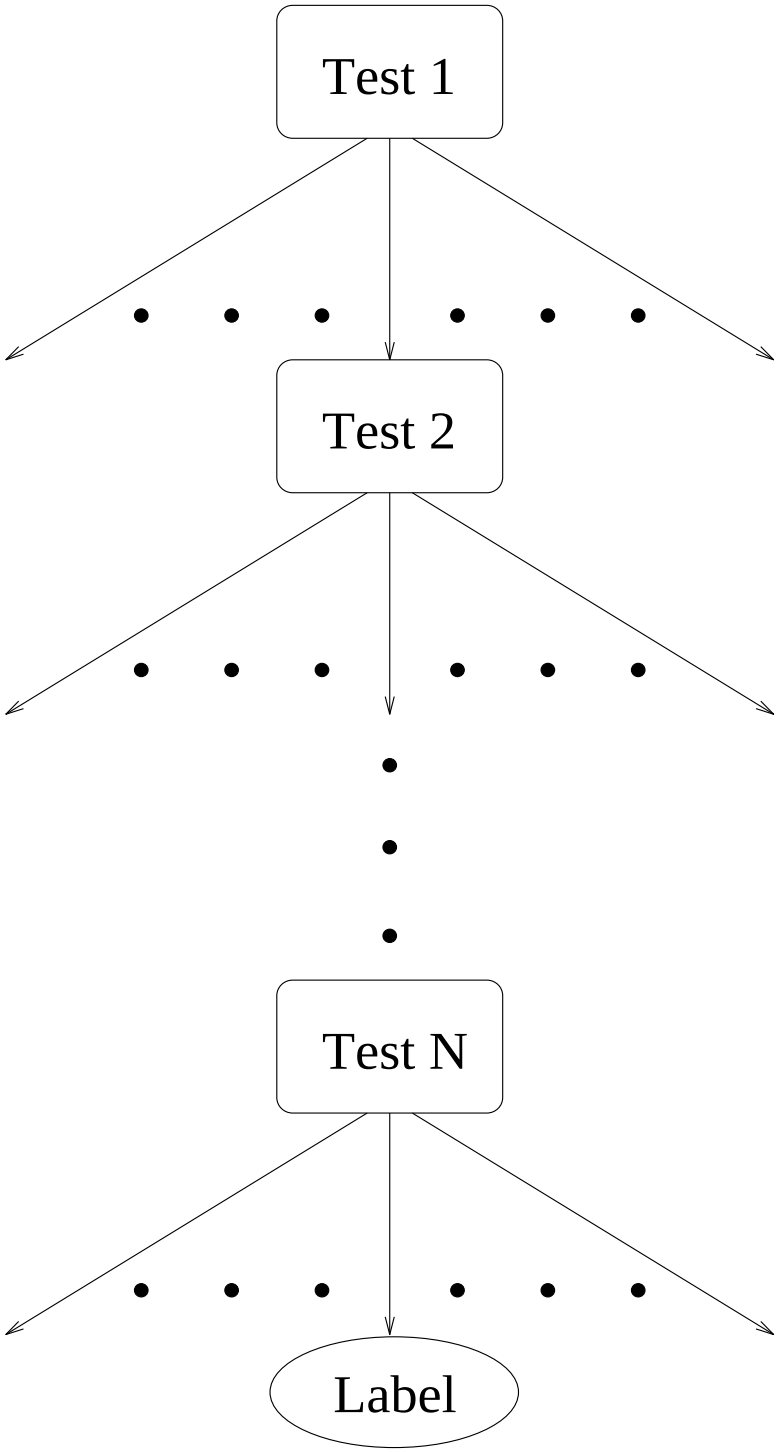
Decision Trees

- A concept description language.
- For instances represented as feature vectors.
- Each internal node checks the value of a feature.
- Branches are labeled by possible values.
- Leaves are labeled:

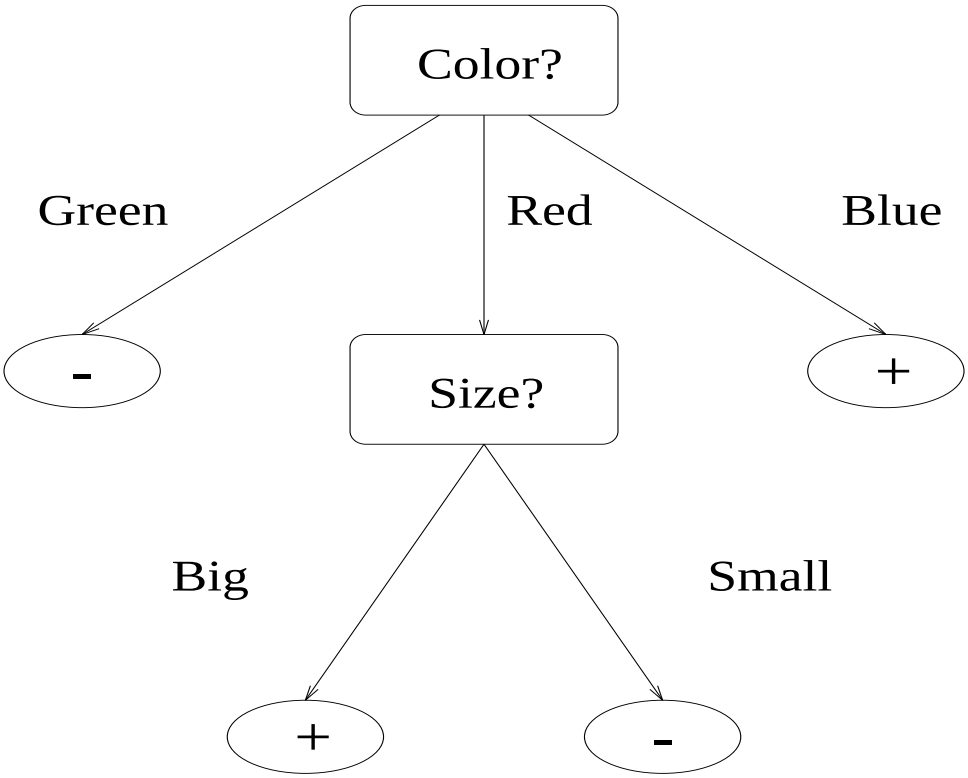
  "+" indicates member of the concept.

  "-" indicates not a member of the concept.

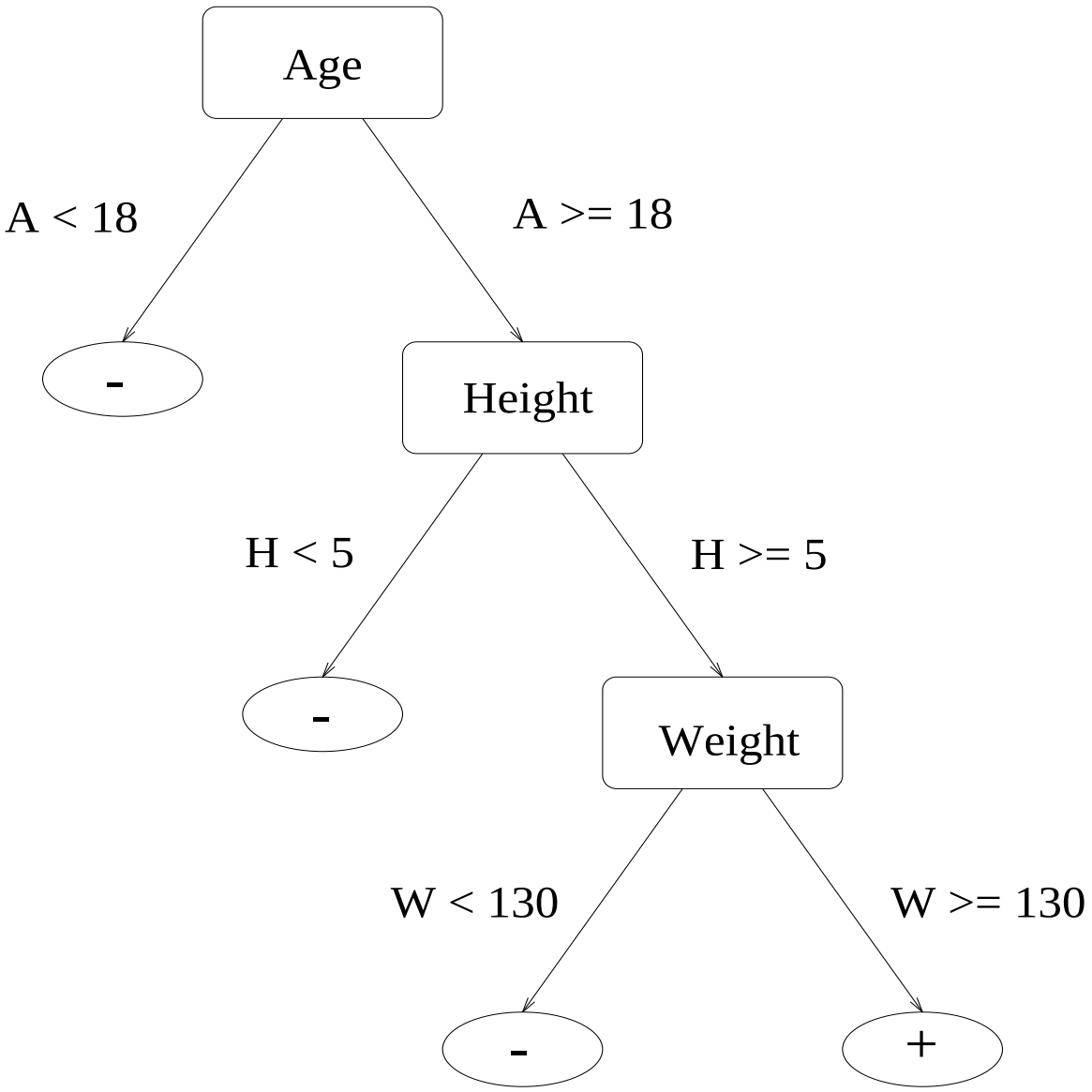
Generic Decision Tree



A Decision Tree for Discrete Feature Vectors



A Decision Tree for Continuous Feature Vectors



Learning Decision Trees

- Goal: Find a smallest tree that correctly classifies all the training examples.
- NP Hard: (Hyafil and Rivest, 1976).
- We must use heuristics if CPU time is limited.

ID3: Greedy Algorithm to Find Small Decision Trees

Given a set of labeled instances:

1. Find a feature that "best" divides the instances into uniform sets.
2. Recursively call ID3 on each subset.

What does "best" mean?

Using Information Theory

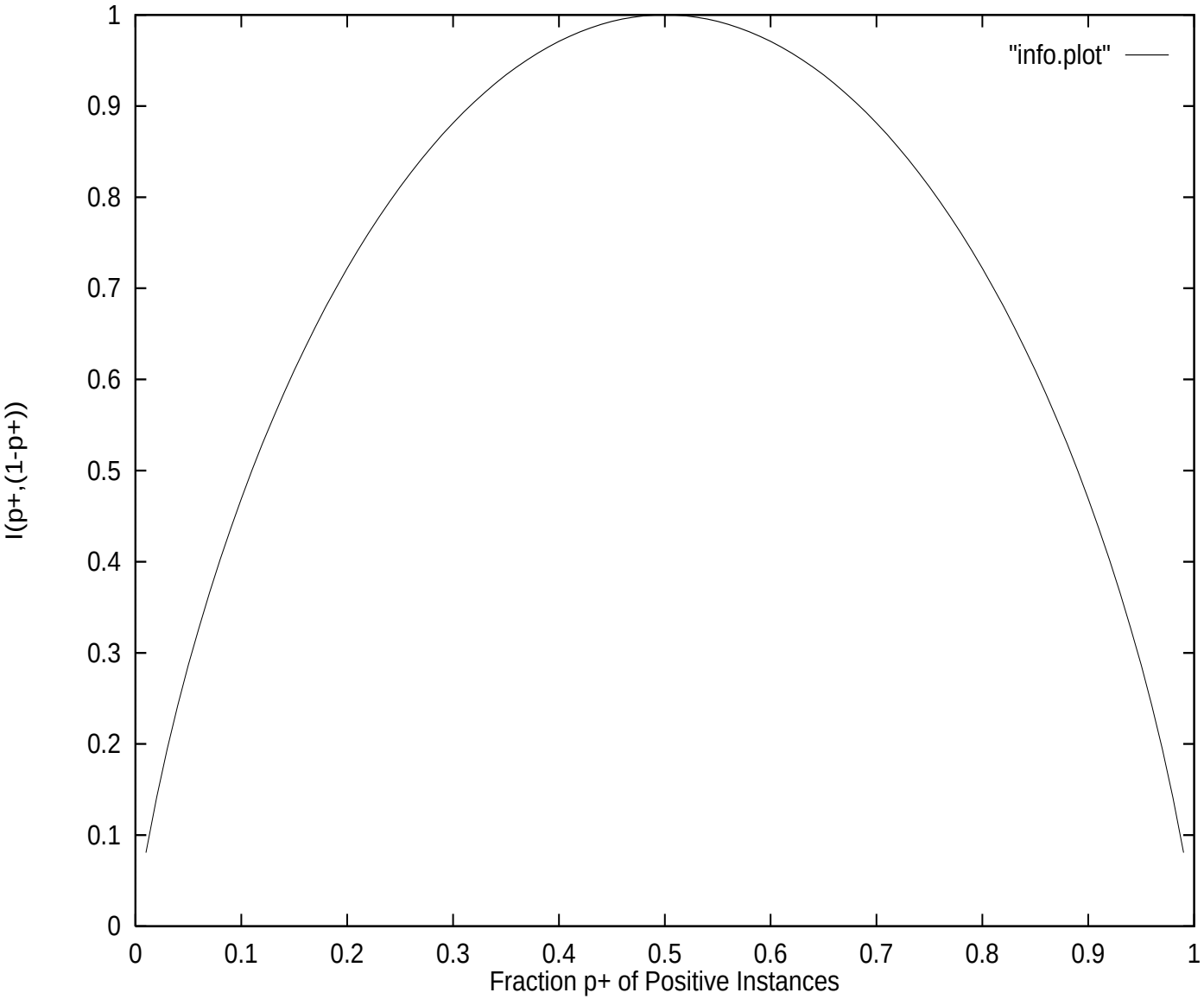
- Let S be a set of unclassified instances.
- Assume we know the fractions of positive and negative instances in S:

  $p^+ =$ Fraction of positive instances.

  $p^- =$ Fraction of negative instances.
- Now someone tells us the classification of some instance in set S.
- What is the *information value* of this new fact?

  $I(p^+, p^-) = -(p^+)log_2(p^+) - (p^-)log_2(p^-)$
- The information value of the fact is the *Entropy* of the instances.

Information Content Function



Finding a "Best" Feature

1. For each feature f do:
   (a) Use f to partition instances into sets $S_1, \ldots, S_n$.
   (b) For each set S_i, determine p_i^+ and p_i^-.
   (c) Let
   $$Gain(f) = I(p^+, p^-) - \Sigma_{i=1}^{n} \left(\frac{|S_i|}{|S|}\right) I(p_i^+, p_i^-).$$
2. Choose a feature f with highest value of $Gain(f)$.

ID3: Quinlan (Discrete Feature Vectors)

```
ID3(INSTANCES, FEATURES):

1. If all INSTANCES are positive, then Return(Positive-Leaf).
2. If all INSTANCES are negative, then Return(Negative-Leaf).
3. Let BEST = Maxarg (f in FEATURES) Gain(f).
4. Let R be the root of a decision tree splitting on BEST.
5. For each value V of BEST do:
   a. Let S = Subset of INSTANCES with BEST = V.
   b. Attach to R the subtree ID3(S,FEATURES - {BEST}).
```

An ID3 Example

	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Round	Small	-
4	Green	Square	Small	-
5	Red	Round	Big	+
6	Green	Round	Big	-

Initial Call:

```
ID3({1,2,3,4,5,6},{Color, Shape,Size})
```

Finding the Best Feature

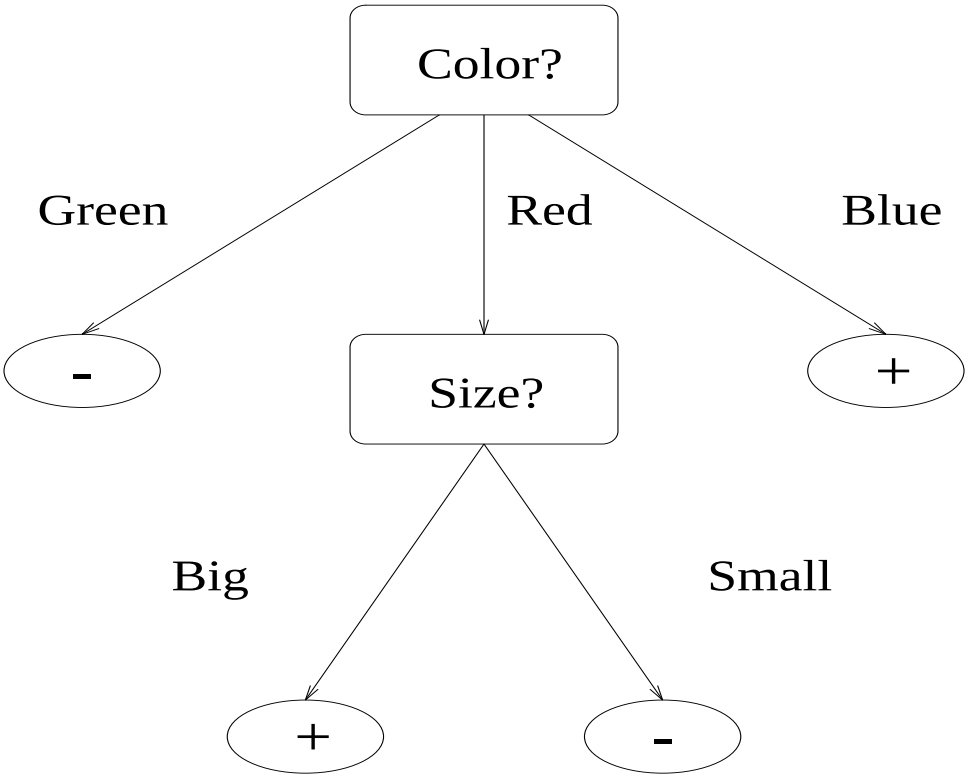
$$
\begin{aligned}
Gain(Color) &= I(3/6, 3/6) \\
&\quad -(3/6) \cdot I(2/3, 1/3) \\
&\quad -(1/6) \cdot I(1/1, 0/1) \\
&\quad -(2/6) \cdot I(0/2, 2/2) \\
&= 1 - 0.459 = \mathbf{0.541}
\end{aligned}
$$

$$
\begin{aligned}
Gain(Shape) &= I(3/6, 3/6) \\
&\quad -(3/6) \cdot I(2/3, 1/3) \\
&\quad -(3/6) \cdot I(1/3, 2/3) \\
&= 1 - 0.918 = \mathbf{0.082}
\end{aligned}
$$

$$
\begin{aligned}
Gain(Size) &= I(3/6, 3/6) \\
&\quad -(2/3) \cdot I(3/4, 1/4) \\
&\quad -(1/3) \cdot I(0/2, 2/2) \\
&= 1 - 0.541 = \mathbf{0.459}
\end{aligned}
$$

Color is best!

Tree Returned by ID3



Generating Rules from Decision Trees

- One rule for each path from root to a leaf node.
- Antecedent: Conjunction of all decisions on path.
- Consequent: Label of the leaf node.

$$Color = Green \Rightarrow -$$

$$Color = Blue \Rightarrow +$$

$$Color = Red \wedge Size = Big \Rightarrow +$$

$$Color = Red \wedge Size = Small \Rightarrow -$$

C4.5: Quinlan (Continuous Feature Vectors)

- Similar to handling of discrete feature vectors.
- For each internal, splitting node:
  - Choose best feature.
  - Choose direction $<$ or $\geq$ of test.
  - Choose threshold k of test: $f < k$ or $f \geq k$.

Estimating Accuracy of a Concept Description

- Data Rich: Separate Training and Test Sets.
  - Select a random subset of the training examples.
  - Withold it from the learning algorithm.
  - Use it as an unbiased test set.
- Data Poor: Cross Validation
  - Divide data into n subsets.
  - Learn one concept description for each collection of $n - 1$ subsets.
  - Test each concept description on the corresponding witheld subset.
  - Use average of n error rates as an estimate of the accuracy when learning from all the data.

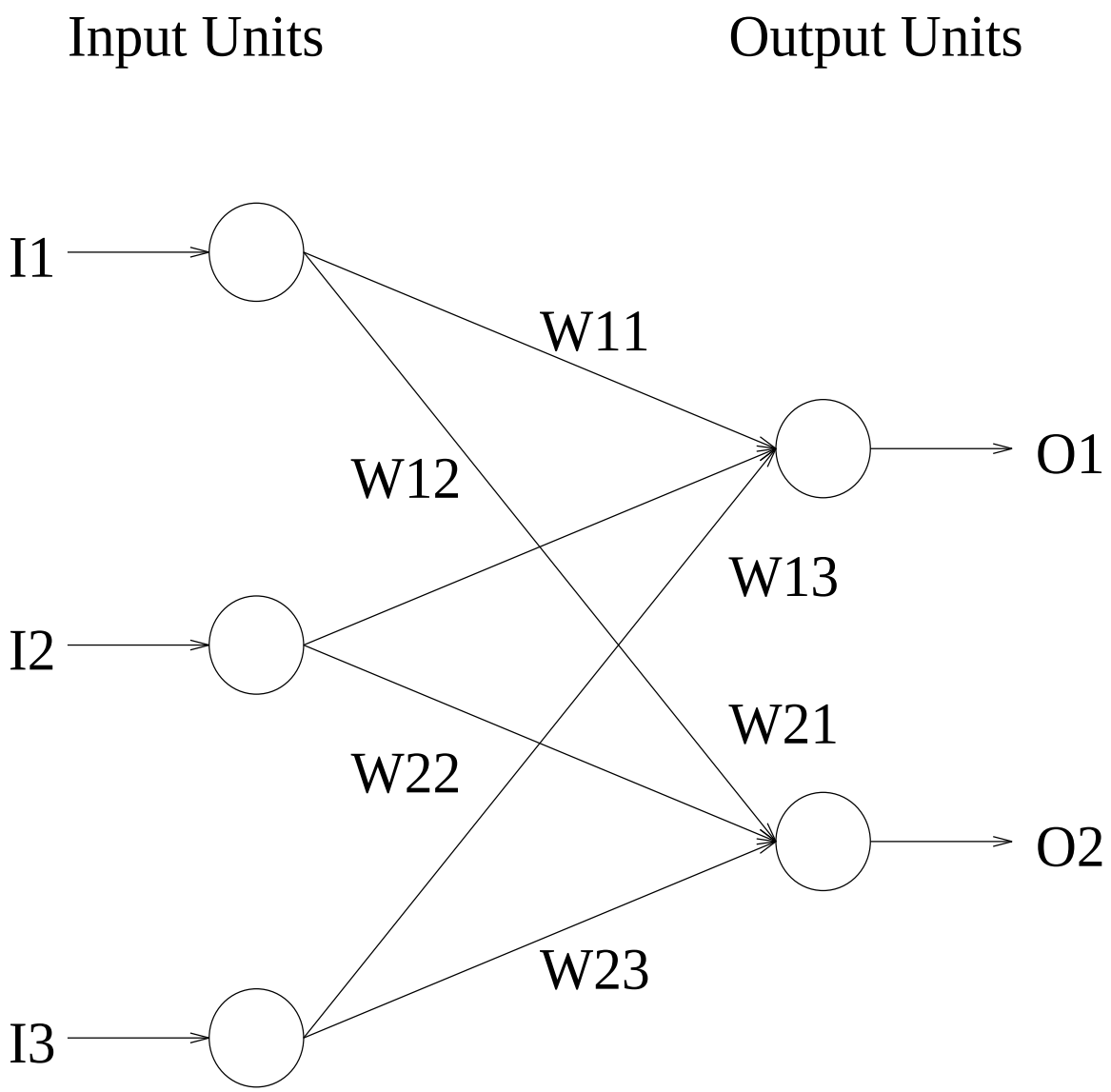
Neural Networks

- Also Known As:
  - Connectionist Machines.
  - Parallel Distributed Processing.
- Early Work: Perceptrons.
- Current Work: Backpropagation.

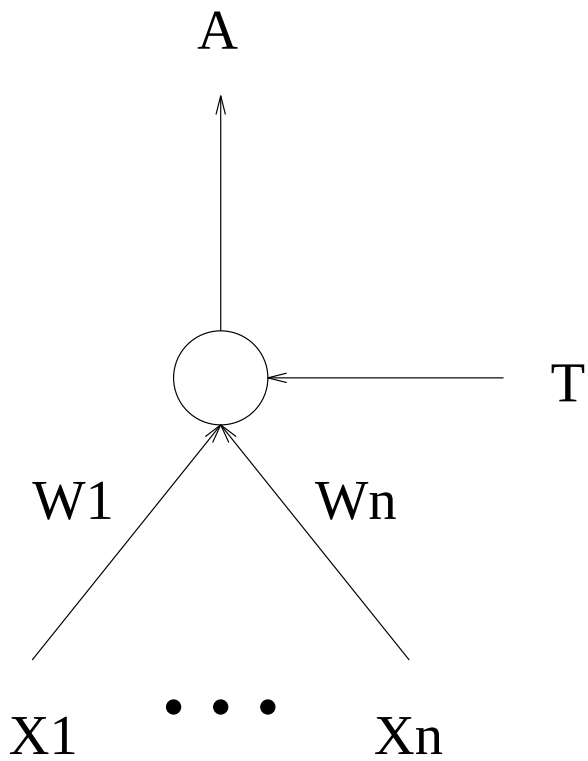
Perceptron

- Rosenblatt, 1958.
- Very simple computing device.
- Very simple learning device.
- Inspired by rough analogy with neuron.

Perceptron

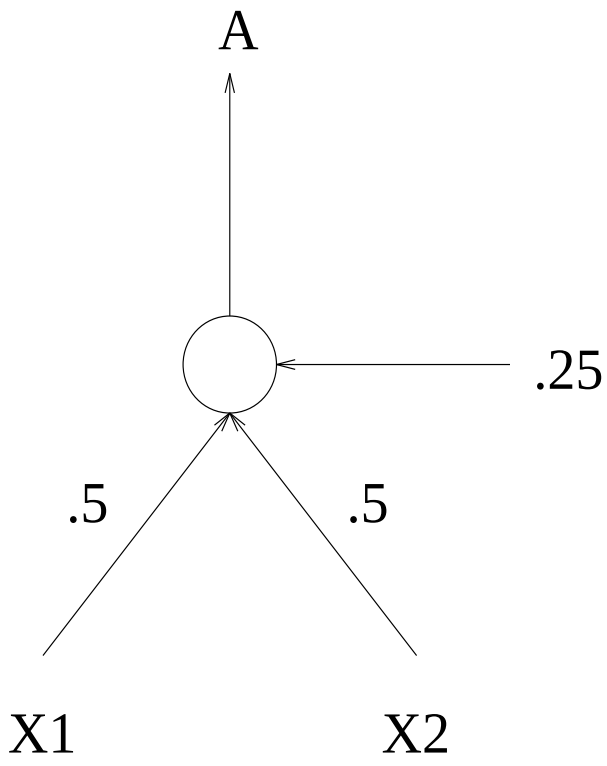


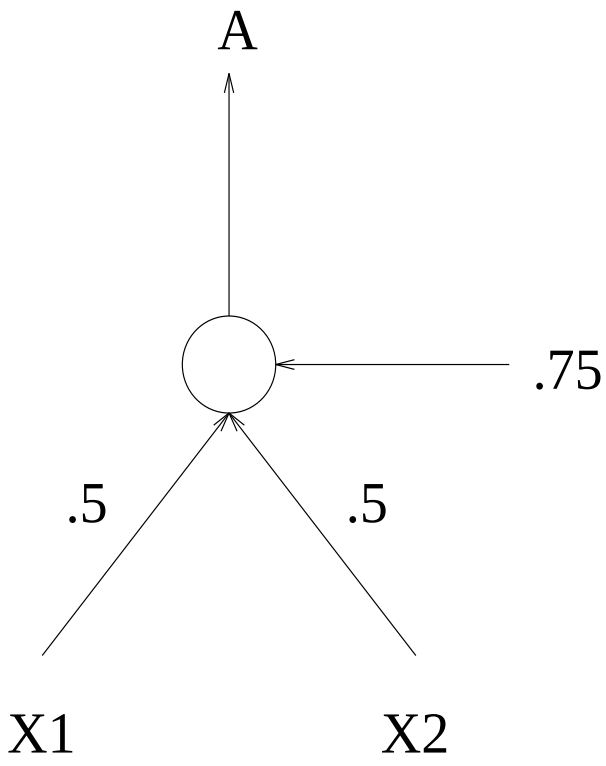
Perceptron Output Function



$$A = \begin{cases} 1 & \text{If } W_1X_1 + \ldots + W_nX_n > T \\ 0 & \text{Otherwise.} \end{cases}$$

Examples





Thresholds as Weights

Notice that

$$W_1X_1 + \ldots + W_nX_n > T$$

is equivalent to

$$W_1X_1 + \ldots + W_nX_n - T > 0$$

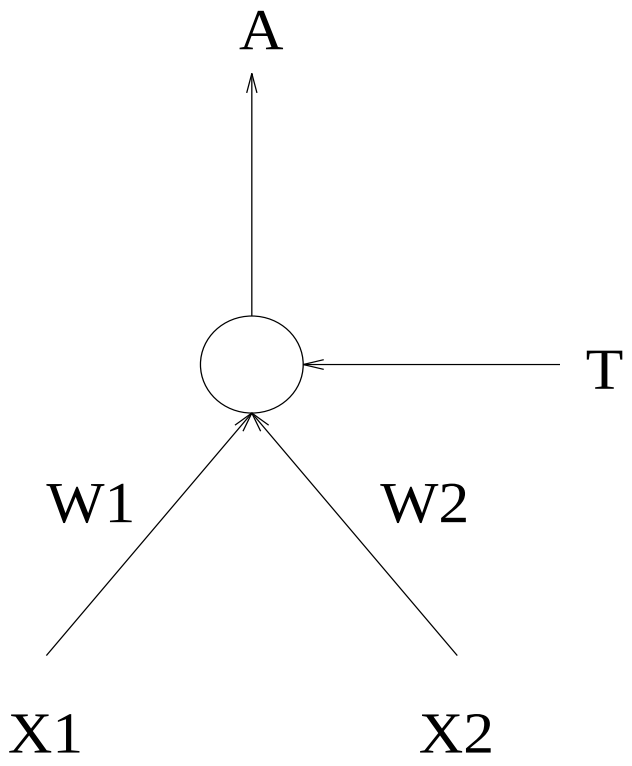
or

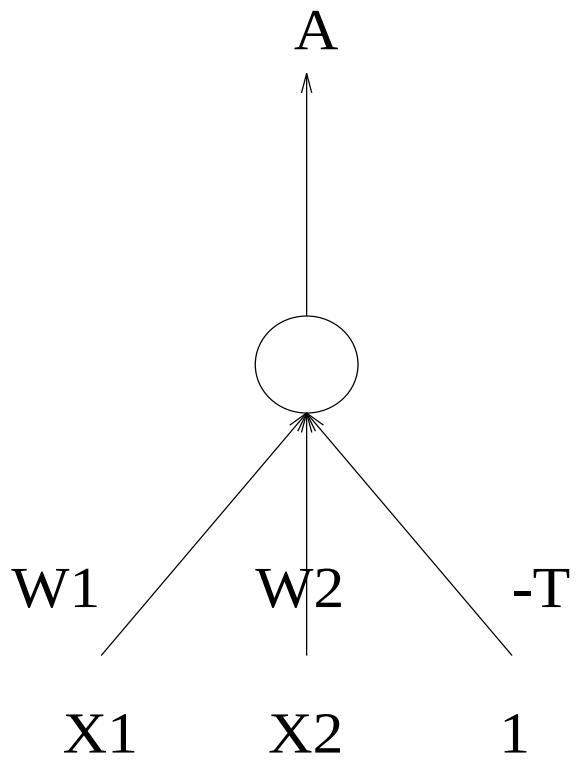
$$W_1X_1 + \ldots + W_nX_n - T \cdot 1 > 0$$

Thresholds as Weights

- Pretend each node has an extra input whose value is always 1 and whose weight is $-T$, called the "bias".
- Learning updates the bias just like all the other weights.

Thresholds as Weights





Learning in Neural Networks

- Learn values of weights from I/O pairs.
- Start with random weights.
- Load training example's input.
- Observe computed output.
- Compare to desired output.
- Modify weights to reduce difference.
- Iterate over all training examples.
- Terminate when weights stop changing.

Perceptron Learning Rule

$$\Delta W_i = \eta(D - A)X_i$$

X_i is a node's input.

W_i is the corresponding weight.

ΔW_i is the change in weight.

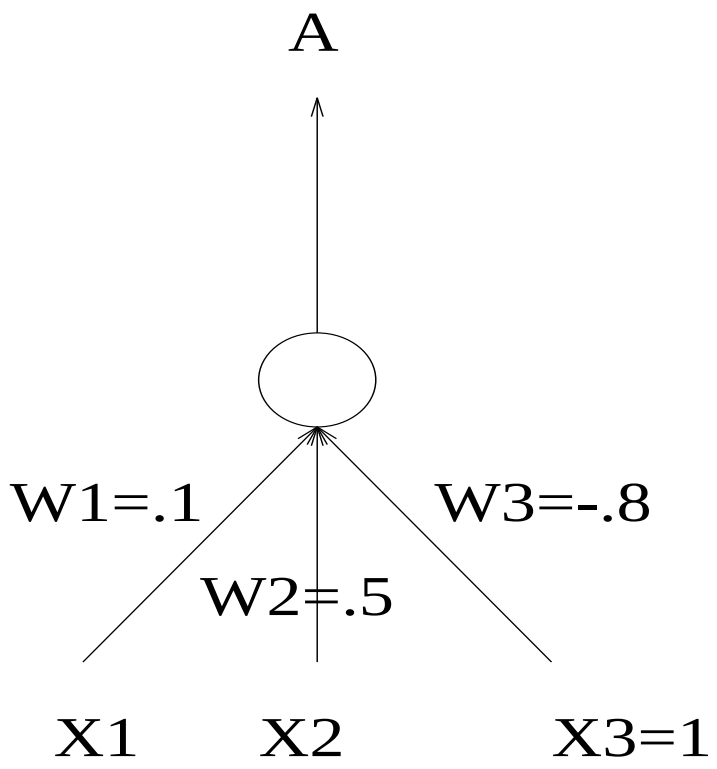
D is the desired output.

A is the actual observed output.

η is the learning rate.

Learning the *Or* Function

$$\Delta W_i = .2(D - A)X_i$$



Learning the *Or* Function

X_1	X_2	D	A	ΔW_1	W_1	ΔW_2	W_2	ΔW_3	W_3
					.1		.5		-.8
0	0	0	0	0	.1	0	.5	0	-.8
0	1	1	0	0	.1	.2	.7	.2	-.6
1	0	1	0	.2	.3	0	.7	.2	-.4
1	1	1	1	0	.3	0	.7	0	-.4
0	0	0	0	0	.3	0	.7	0	-.4
0	1	1	1	0	.3	0	.7	0	-.4
1	0	1	0	.2	.5	0	.7	.2	-.2
1	1	1	1	0	.5	0	.7	0	-.2
0	0	0	0	0	.5	0	.7	0	-.2
0	1	1	1	0	.5	0	.7	0	-.2
1	0	1	1	0	.5	0	.7	0	-.2

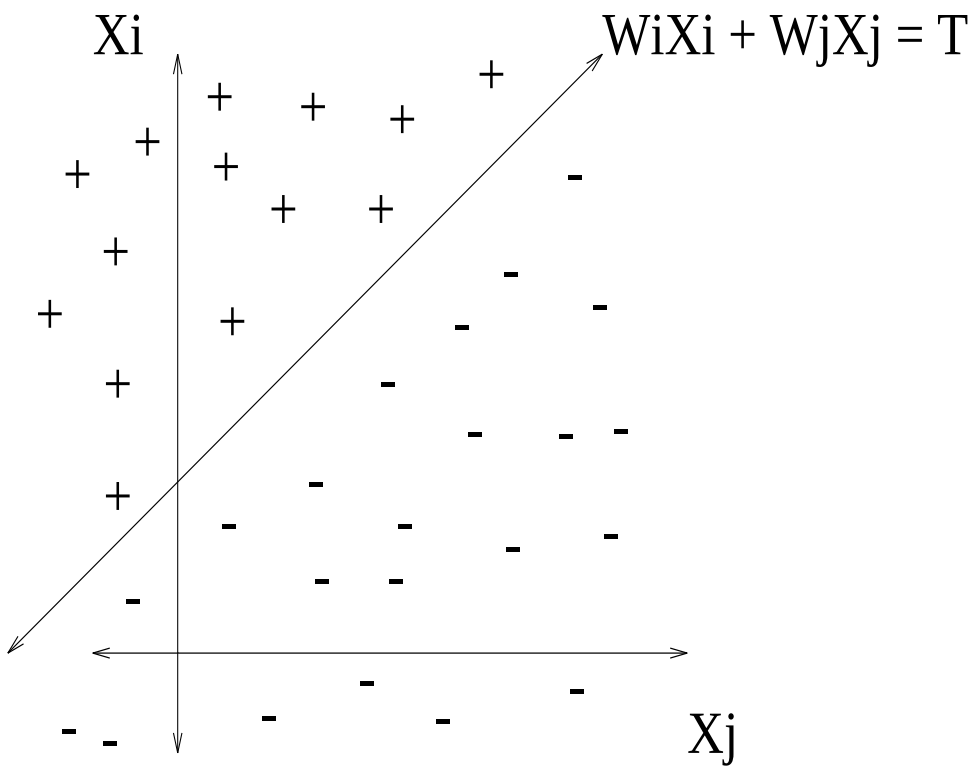
Perceptron Convergence Theorem

"If a set of I/O pairs is learnable, then the Perceptron Learning Rule will find the necessary weights."

Minsky and Papert, 1988.

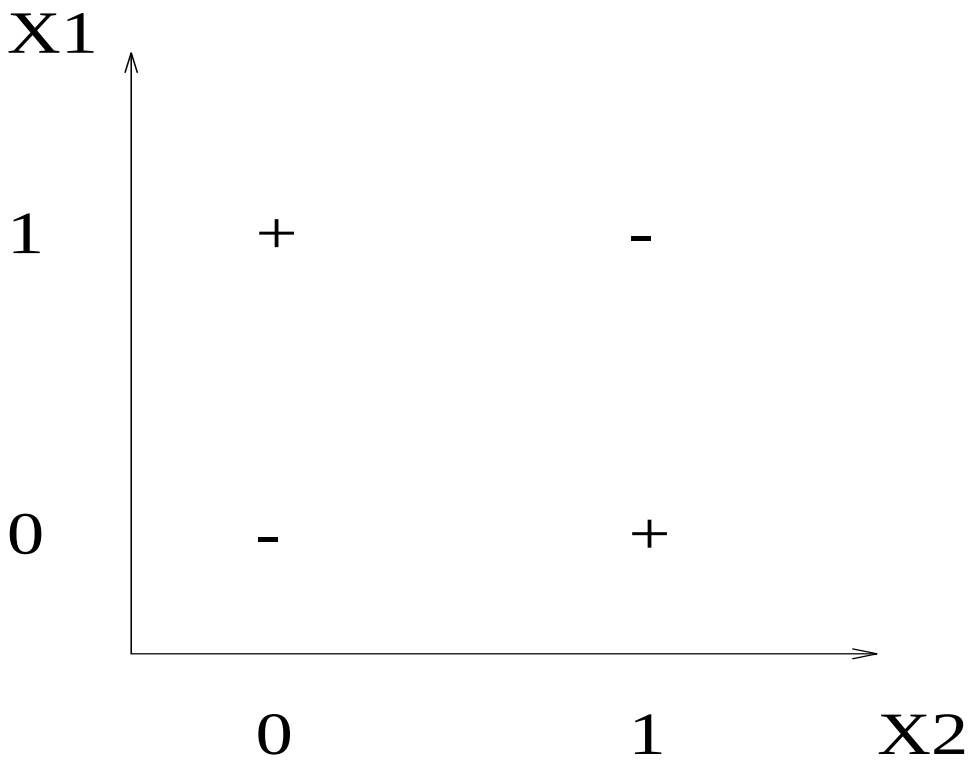
Linear Separability

The output function $W_1X_1 + \ldots + W_nX_n > T$ defines a hyperplane that splits the input space into two half spaces.

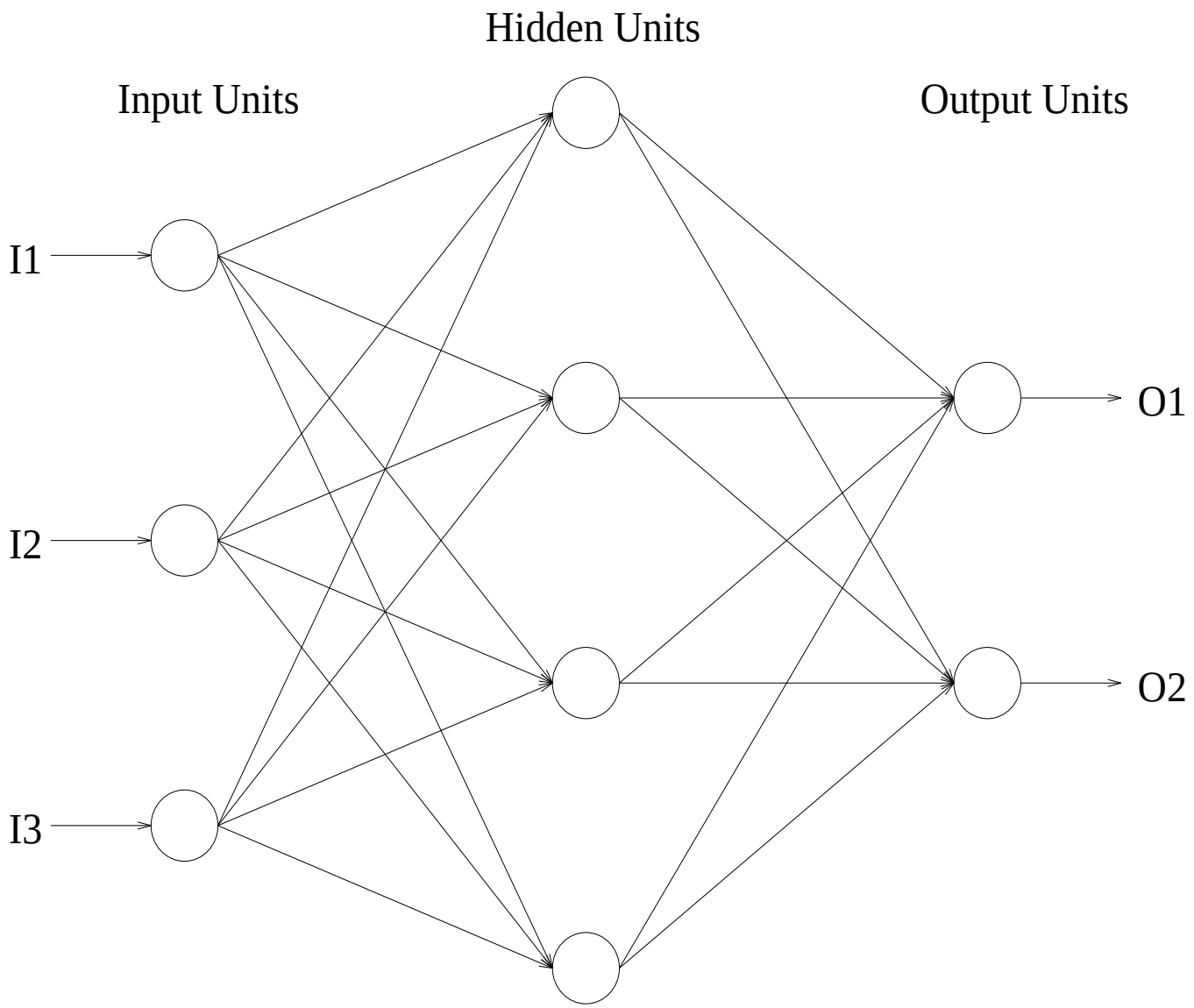


Linear Separability: The *XOR* Function

Can a single line separate the two classes?

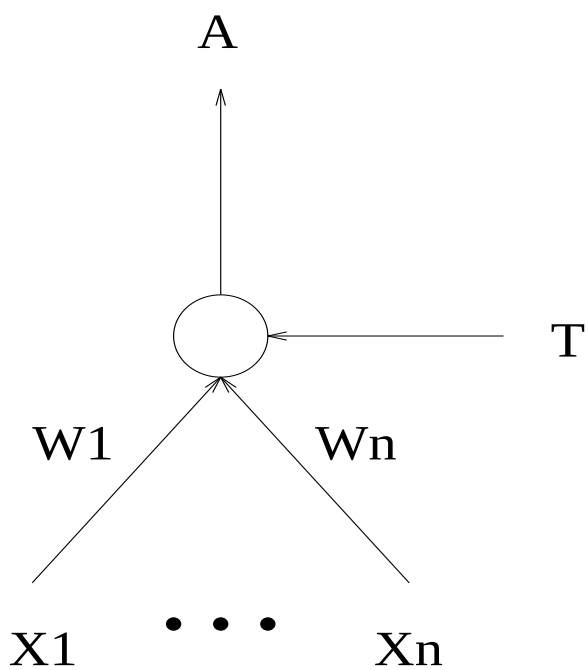


Solution: Hidden Units



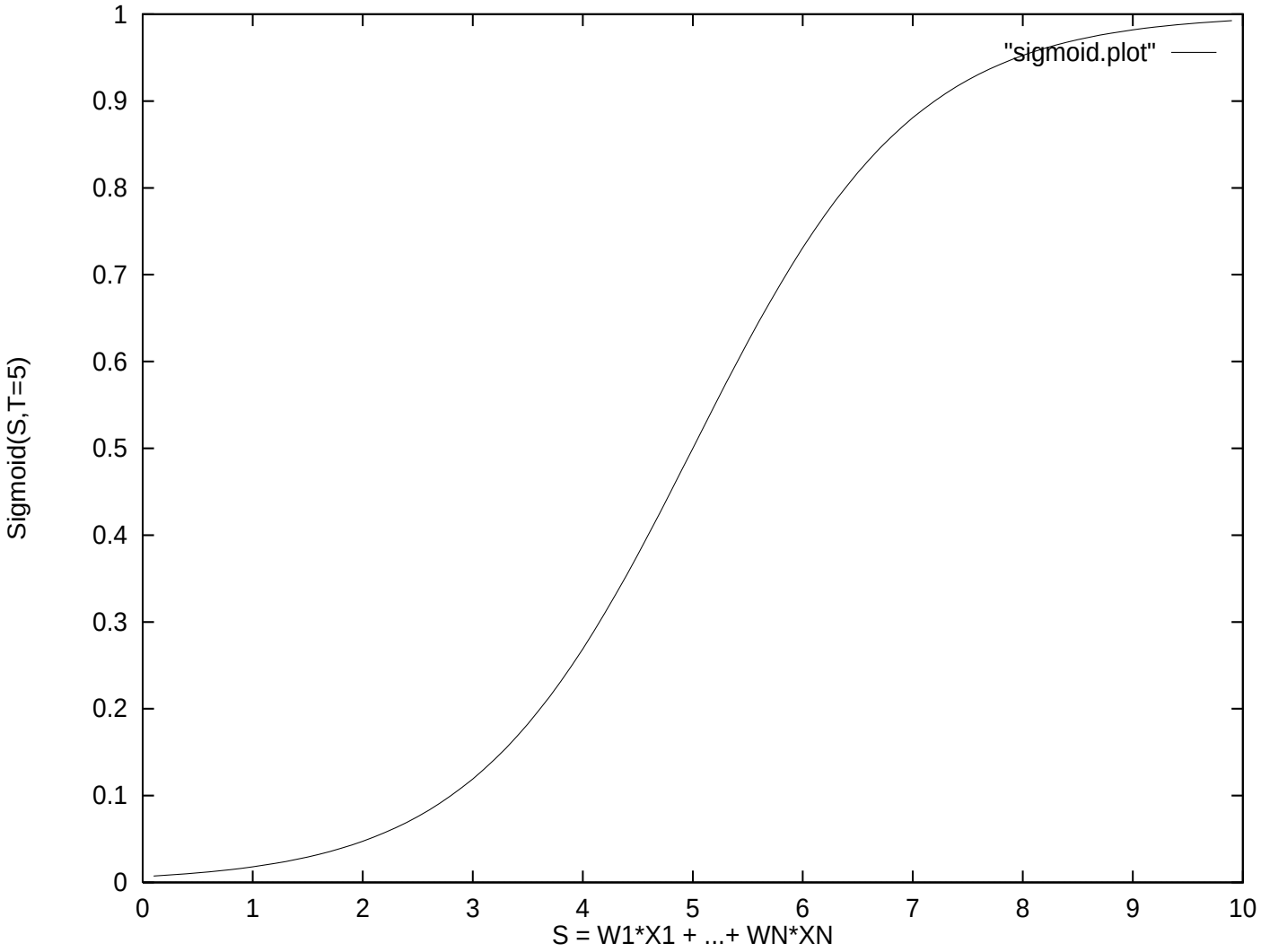
Backpropagation (Backprop) Networks

- Network contains hidden units.
- Each unit obeys "Sigmoidal Activation Function".

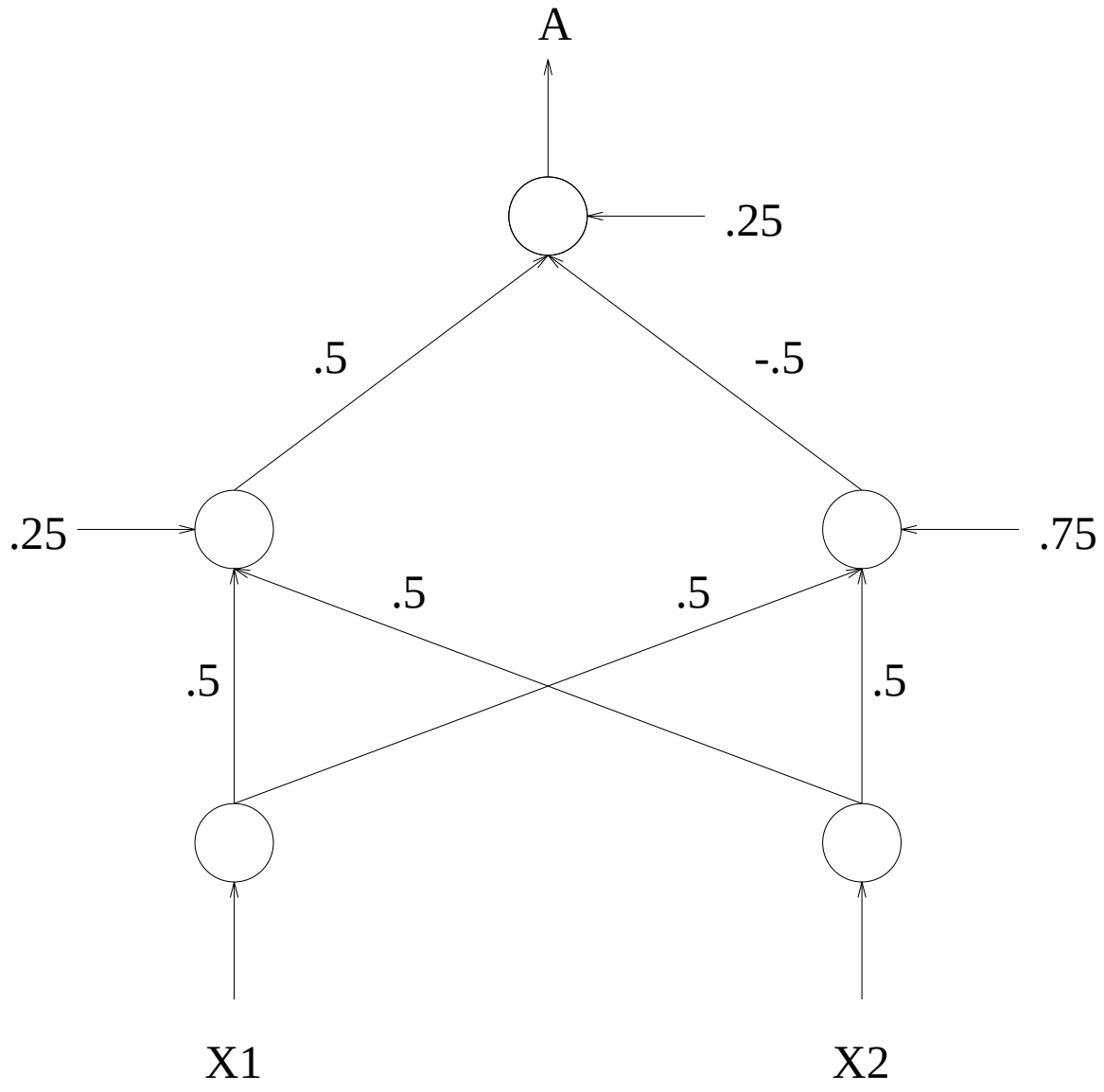


$$A = \frac{1}{1+e^{-(W_1X_1+\ldots+W_nX_n-T)}}$$

Behavior of Sigmoid Function



Implementation of the XOR Function



Backpropagation Learning Rule for Weights of Edges from Hidden Units to an Output Unit j

$$\Delta W_{ij} = \eta A_j(1 - A_j)(D_j - A_j)H_i$$

H_i is a node's input (a hidden node's output).

W_{ij} is the corresponding weight.

ΔW_{ij} is the change in weight.

D_j is the desired output.

A_j is the actual observed output.

η is the learning rate.

Backpropagation Learning Rule for Weights of Edges from Input Units to a Hidden Unit i

$$\Delta W_{ki} = \eta H_i(1 - H_i)E_i X_k$$

X_k is a node's input.

W_{ki} is the corresponding weight.

ΔW_{ki} is the change in weight.

$E_i = \Sigma_{j=1}^{n} W_{ij} A_j (1 - A_j)(D_j - A_j)$ is the propagated error from the n output units.

H_i is the actual observed output.

η is the learning rate.

Motivation for the Backpropagation Learning Rules

- The weights $\bar{W} = (W_1, \ldots, W_n)$ define a point in an n-dimensional Euclidean space.
- Each point $\bar{W}$ in the space defines a network.
- Each point $\bar{W}$ has an associated error rate:

$$E = \frac{1}{2}\Sigma_i (D_i - A_i)^2$$

- We compute the gradient ∇E with respect to $\bar{W}$.
- Each learning iteration changes $\bar{W}$ to $\bar{W} - \eta \nabla E$.
- A "Gradient Descent" algorithm.

Applications of Backpropagation Networks

- NETtalk converts character strings to phonemes.
- Neurogammon won the 1989 Computer Olympiad.
- ALVINN steers a vehicle along a single lane highway.

Advantages and disadvantages of Neural Networks

Advantages:

- Generalization capability.
- Low sensitivity to noise.

Disadvantages:

- Relative expressiveness.
- Computational efficiency.
- Transparency (black box).
- Hard to use prior knowledge.

Noise and Overfitting

- **Noise:** Incorrect data (label errors or attribute errors)
- **Overfitting:** Learner does very well on the training data but poorly on the testing data (unseen examples).
- Noise is the most common cause for overfitting.

Some solutions

- Decision tree pruning: simplify tree by shortening some branches with mostly positive (negative) examples.
- Use of less expressive concept description languages, such as linear classifiers (Perceptrons).
- Stopping the learning early when overfitting is detected.

Case Based Learning

- Very simple model for learning.
- Stores labeled examples.
- Classifies unseen examples based on their **similarity** to stored examples.
- Uses **distance functions** to measure similarity.

Nearest Neighbor

- Gives unseen example the label of the nearest stored exampled.
- Outcome depends on the distance function used.
- **Euclidean Distance** is suitable for continuous features.
- **Hamming Distance** is suitable for binary (discrete) features.

Problem: Very sensitive to noise.

K-Nearest Neighbors

- Finds the K nearest neighbors of the unseen example among the stored examples.
- Uses the labels of these neighbors to label the unseen example.
- **Voting K-Nearest Neighbors** labels the unseen example with the majority label among the K nearest neighbors.
- **Weighted K-Nearest Neighbors** uses distances to weigh the votes.

Batch vs. Incremental learning

- **Batch learning:** considers all training examples simultaneously.
- Examples: Backpropagation, ID3.
- **Incremental learning:** tries to update the old hypothesis whenever a new example arrives.
- Examples: Candidate elimination, Case based learning.