Chapter 4

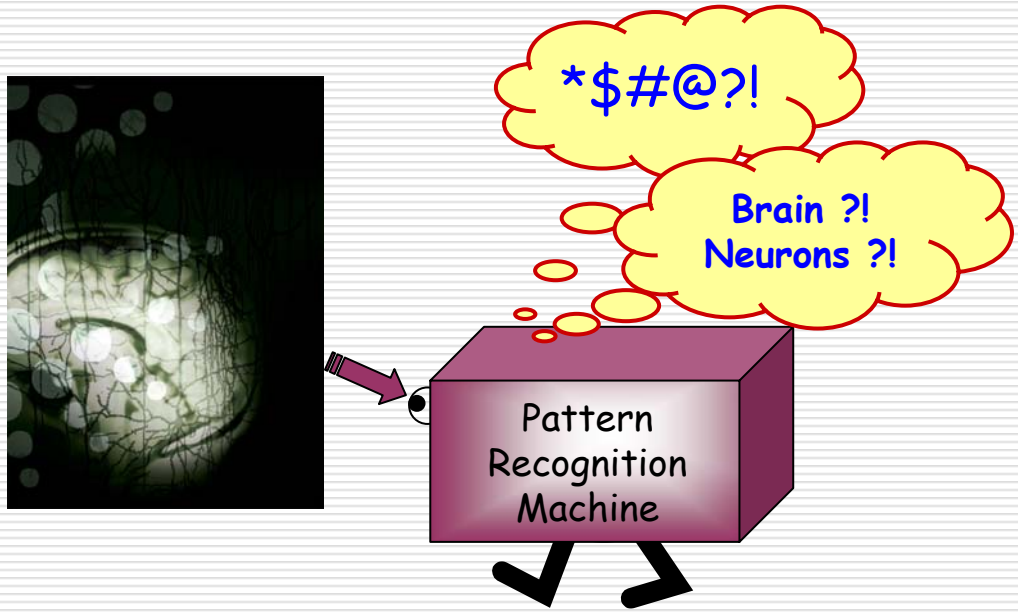
Geometry of Binary Threshold Neurons and their Networks

Neural Networks: A Classroom Approach

Satish Kumar
Department of Physics & Computer Science
Dayalbagh Educational Institute (Deemed University)

Pattern Recognition and Data Classification

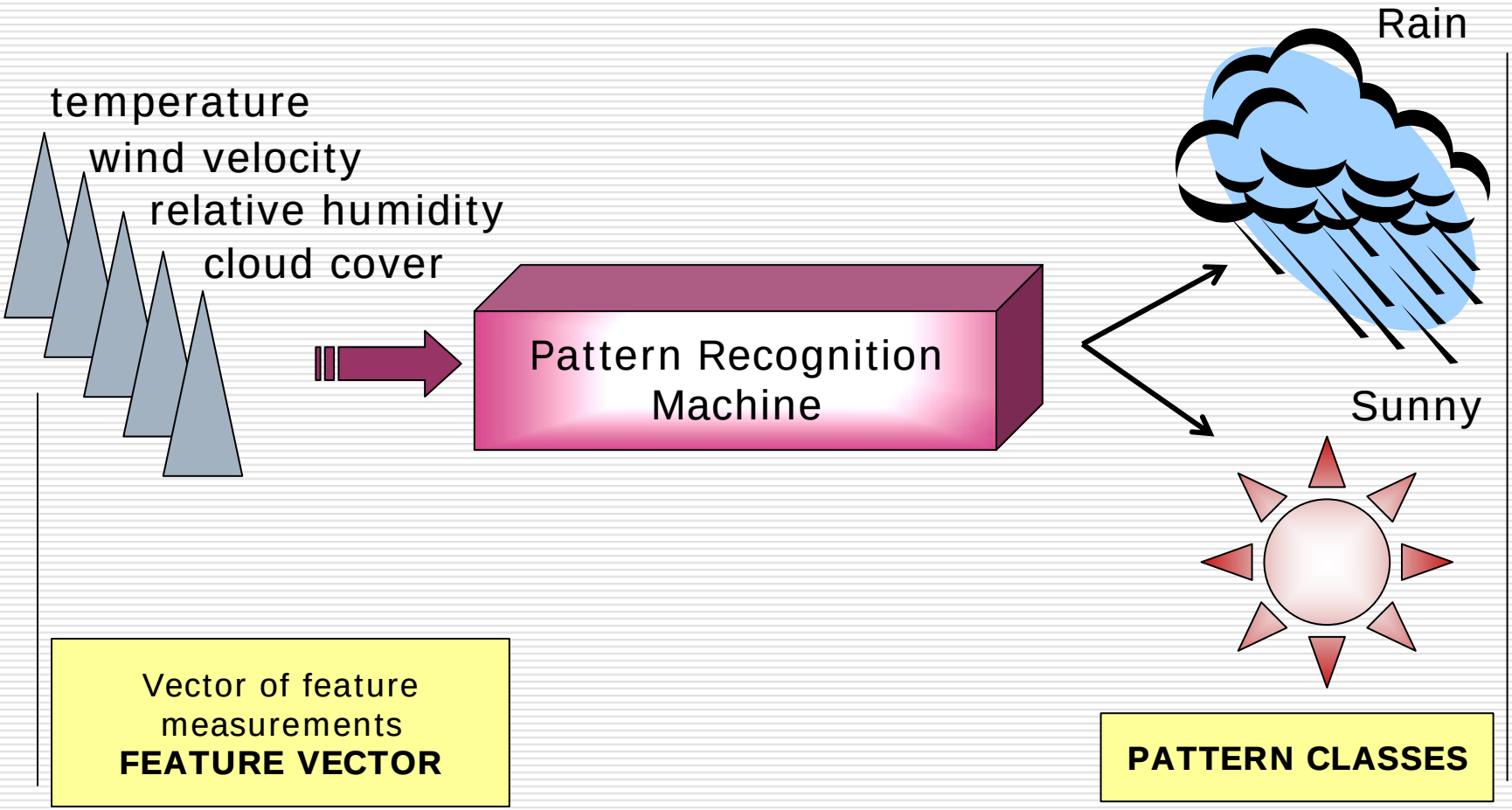
- Pattern recognition is
  - concerned with machine recognition of regularities in noisy or complex environments.
  - the search for structure in data.
- Involves placing an input pattern into one of possibly many decision classes.



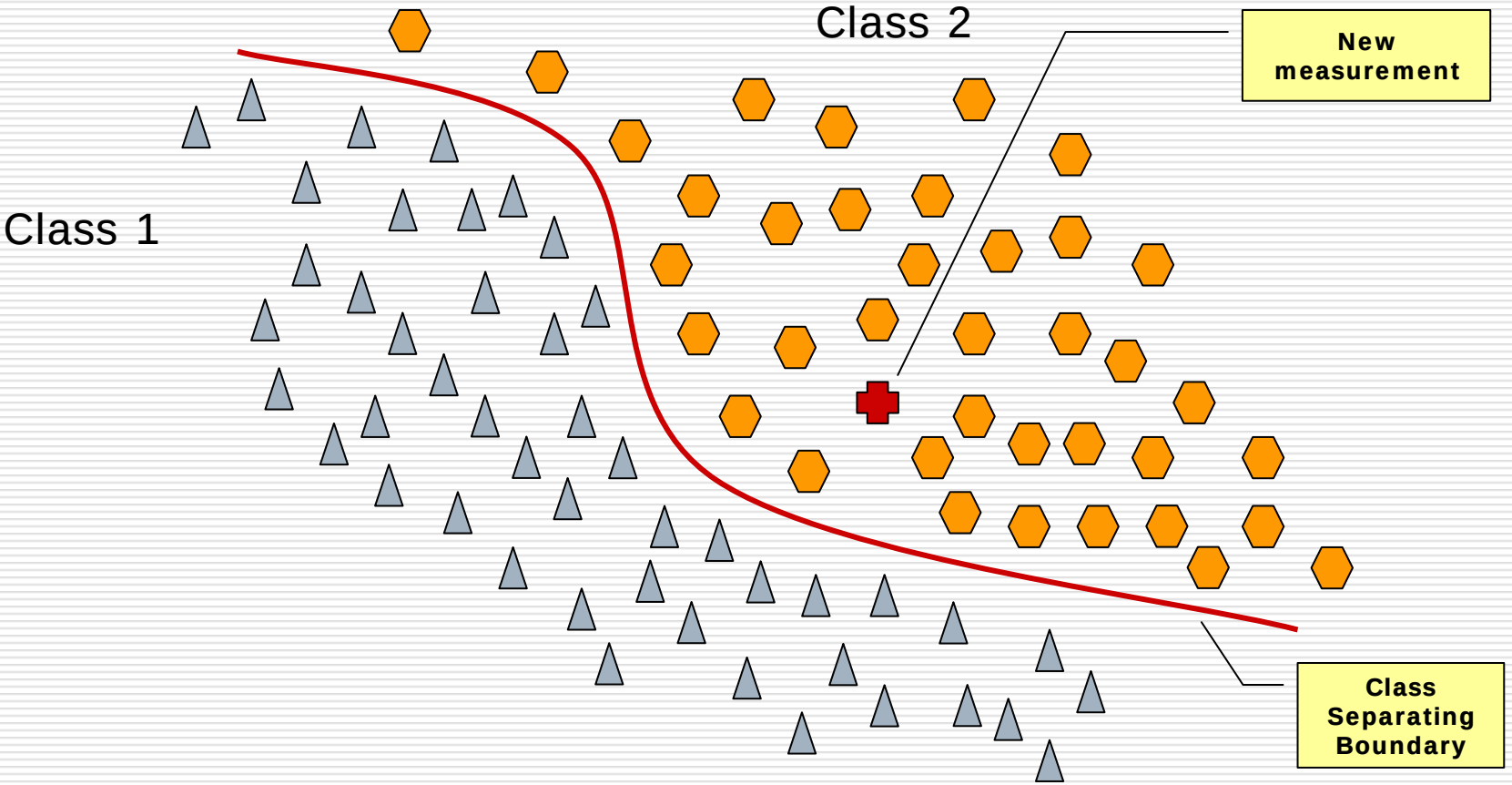
Recognition

- The ability to classify data into (known) categories.
- Human beings have an innate ability to recognize complex patterns with effortless ease, based on categories created internally without conscious intervention.
  - Parents recognize the cry of a child
  - Easily identify an individual from a silhouette
  - Recognize a fast moving car at dusk.
- Real world embedded pattern recognition systems draw on the brain metaphor
- Applications include
  - ZIP code readers
  - DNA fingerprinting
  - Medical diagnosis
  - Occluded target recognition
  - Signature verification

A Pattern Classifier



A Decision Boundary

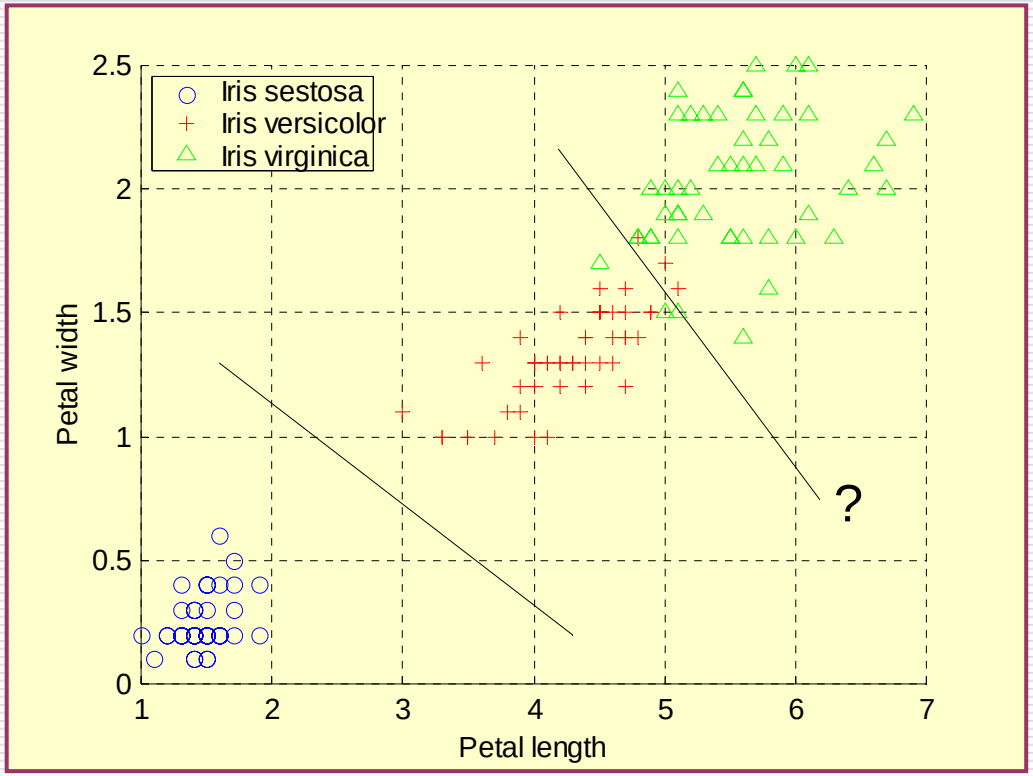


Basic Pattern Classification Problem

- Given some samples of data measurements of a real world system along with correct classifications for patterns in that data set, make accurate decisions for future unseen examples.

Iris Data Classification

- It is easy to separate *iris sestosa* data from *iris versicolor* and *iris virginica*
- It is much more difficult to decide where to place a separating line between *iris versicolor* and *iris virginica*
- Any placement of the straight line will cause some pattern to get misclassified!

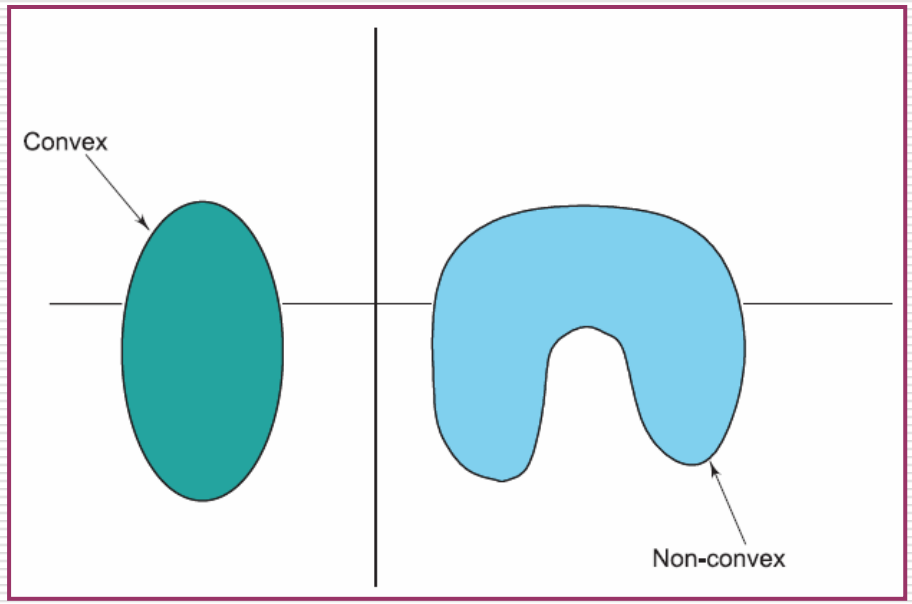


Two-Class Data Classification Problem

- Given a set of Q data samples or patterns $X = \{X_1, \ldots, X_Q\}, X_i \in R^n$, drawn from two classes C_1 and C_2, find an appropriate hyperplane that separates the two classes so that the resulting classification decisions based on this class assignment are on average in close agreement with the actual outcome.

Convex Sets, Convex Hulls and Linear Separability

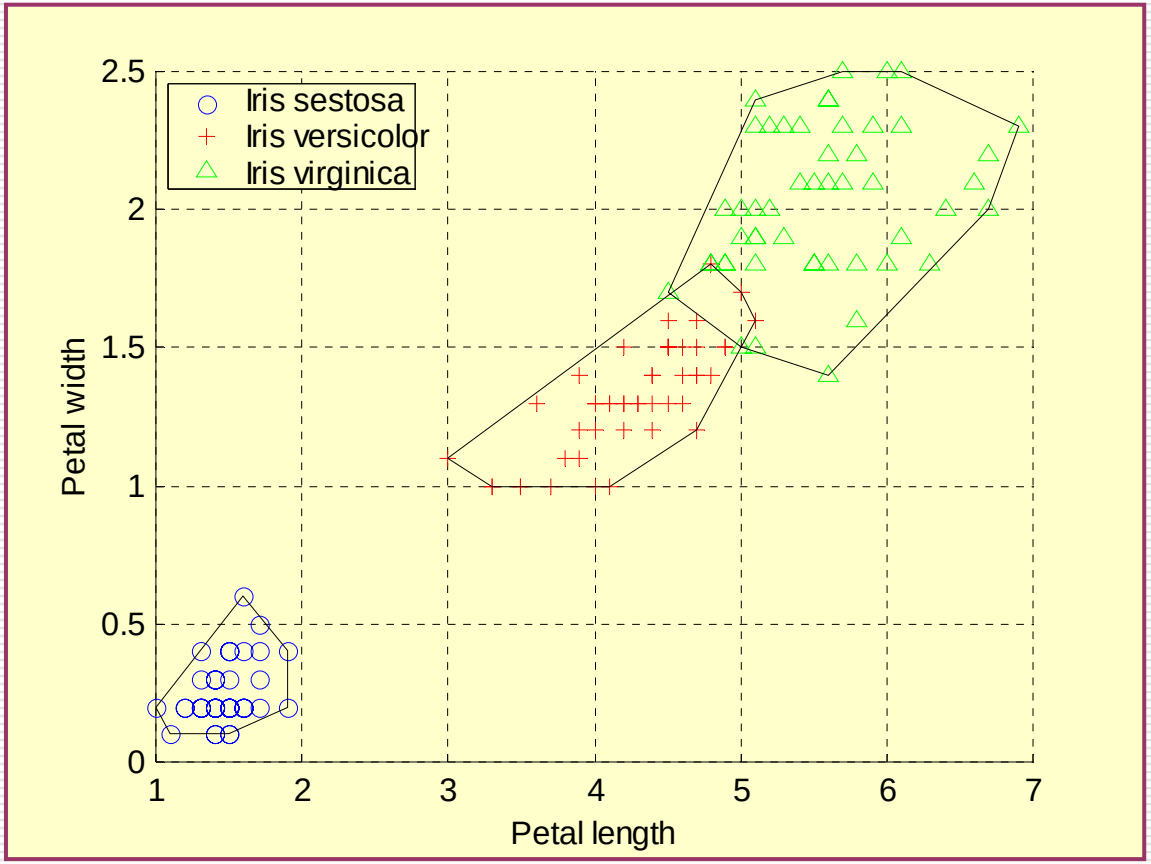
- Let $X, Y \in S \subseteq R^n$, then S is convex iff $\lambda X + (1 - \lambda)Y \in S, 0 \leq \lambda \leq 1, \forall X, Y \in S$.
- Equivalently, a set S is convex if it contains all points on all line segments with end points in S.



Convex Hull

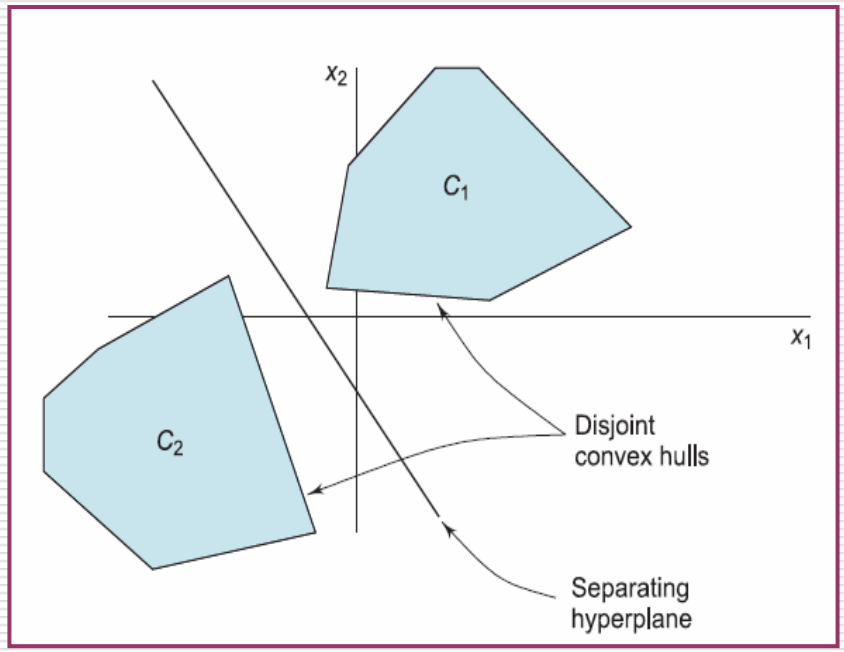
- The convex hull, $C(X_i)$, of a pattern set X_i is the smallest convex set in R^n which contains the set X_i.
- Consider every convex set S_α, such that $X_i \subseteq S_\alpha \subseteq R^n$, $\alpha \in I$, where I is an index set. Then the convex hull of X_i, $C(X_i) = \cap_{\alpha \in I} S_\alpha$.

MATLAB Generated Convex Hulls For Iris Data



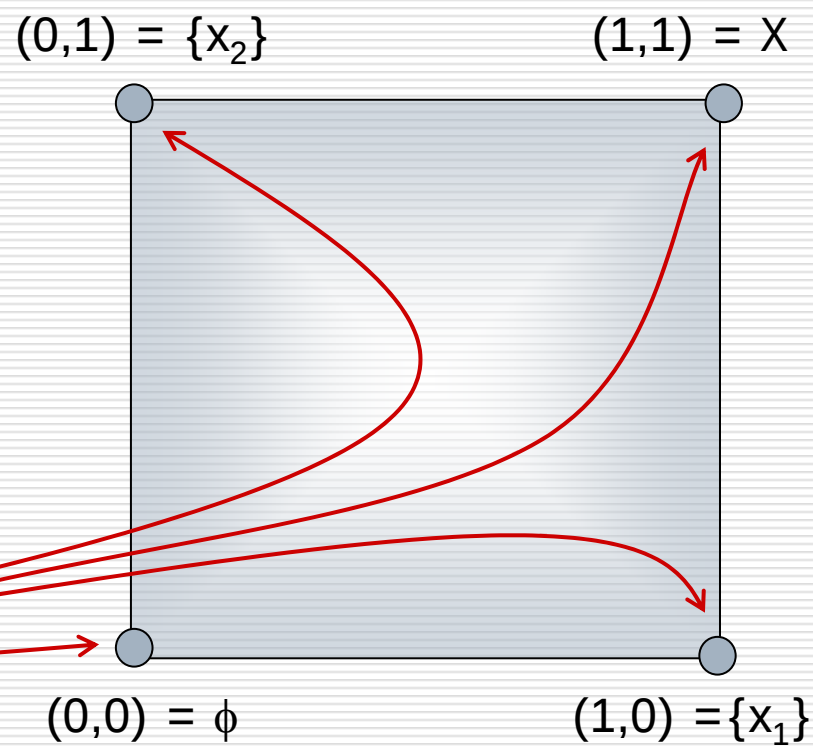
Linearly Separable Classes and Separating Hyperplane

- Two pattern sets X_i and X_j are said to be linearly separable if their convex hulls are disjoint, that is if $C(X_i) \cap C(X_j) = \varphi$.
- One separating hyperplane is the perpendicular bisector of the straight line joining the closest two points on the disjoint convex hulls.



Space of Boolean Functions

- An n-dimensional Boolean function is a map from a domain space that comprises 2^n possible combinations of the n variables into the set {0, 1}.
- B^2



Boolean Functions in n-Dimensions

- In a *Boolean function*, each of 2^n domain points maps to either 0 or 1, $f : B^n \rightarrow \{0, 1\}$.
- Let X_0 denote the set of points that map to 0 and X_1, the set of those that map to 1. Sets X_0, X_1 comprise points that are pre-images of range points 0, 1: $X_0 = f^{-1}(0)$ and $X_1 = f^{-1}(1)$.
- Each unique assignment of 0-1 values to the 2^n possible inputs in n-dimensions represents a Boolean function.
- Therefore, in n-dimensions, there are 2^{2^n} such unique assignments that can be made, and thus 2^{2^n} possible functions.

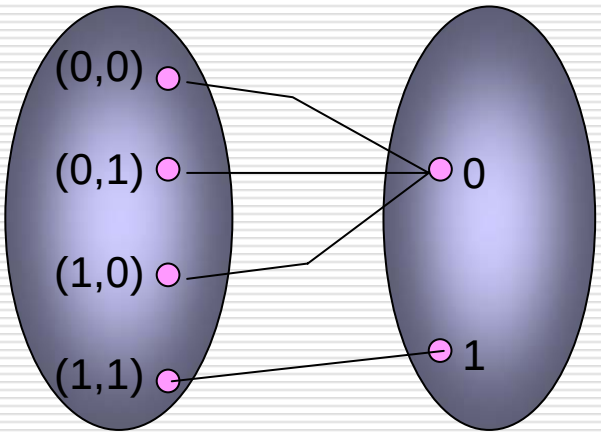
Separability of Boolean Functions

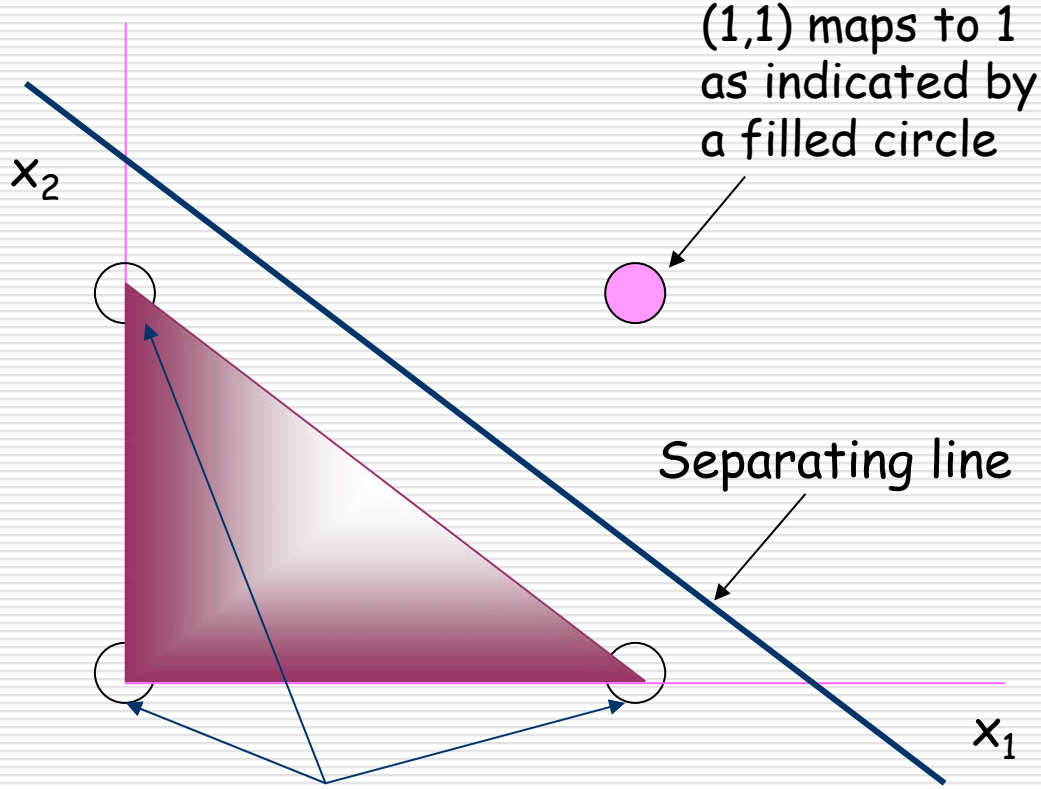
- Linearly separable sets can be separated by a straight line in 2 dimensions.
- In higher dimensions we say the sets are segregated by a separating hyperplane

The Boolean AND Function is Linearly Separable

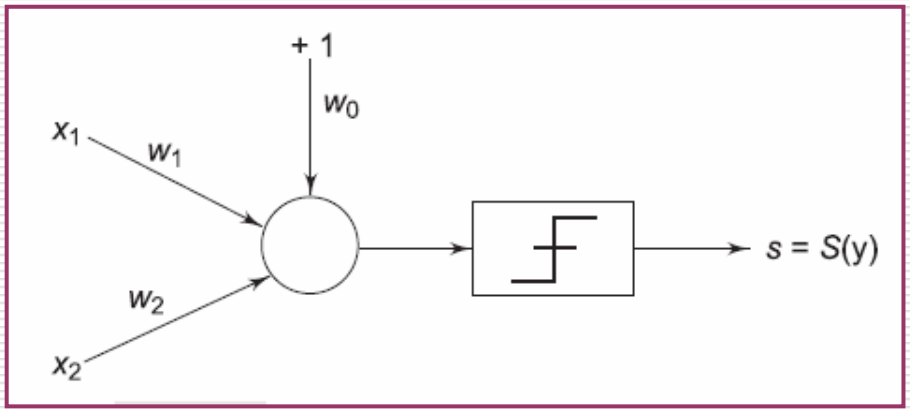
Truth Table

x_1	x_2	$f_\wedge$
0	0	0
0	1	0
1	0	0
1	1	1





Binary Neurons are Pattern Dichotomizers



- Neuron Input vector $X = (1, x_1, x_2)$
  Weight vector $W = (w_0, w_1, w_2)$
- Internal bias modelled by weight w_0, with a constant +1 input.
- Neuronal activation $y = X^T W = w_1 x_1 + w_2 x_2 + w_0$
- Neuron discriminant function $y(X) = 0$ if $y > 0$, $s = 1$, and if $y < 0$, $s = 0$.

Discriminant Function

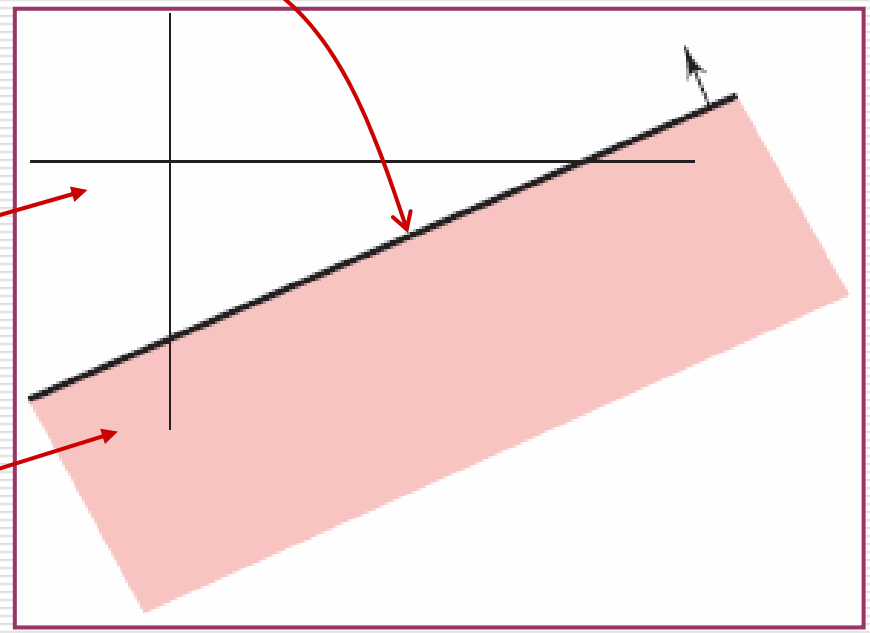
- Given a fixed set of weights, the neuron fires a +1 signal for inputs (x_1, x_2) which yield positive activation and fires a 0 for inputs that yield negative activation.
- Inputs are thus separated into two "classes". Inputs satisfying the discriminant function yield zero activation.
- The discriminant function can be written as

$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_0}{w_2}$$

Discriminant Function

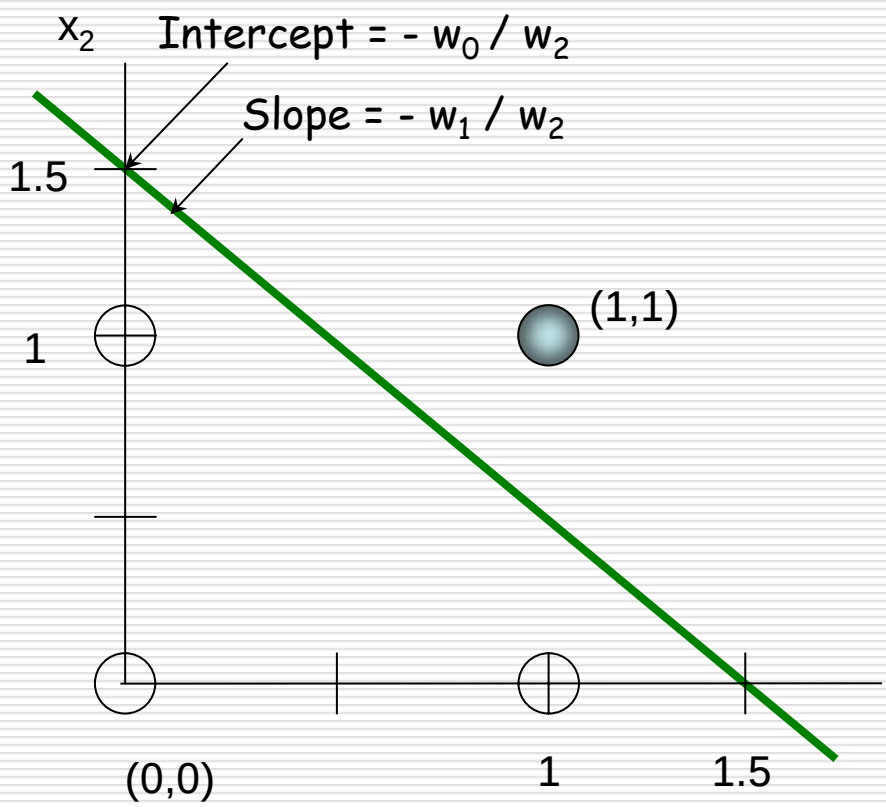
- The neuron discriminant function thus represents a straight line in the two dimensional pattern space
- This straight line slices the plane into two halves:
  - one half where patterns cause the neuron to generate positive inner products and thus fire a +1;
  - the other half where pattern space points cause the neuron to generate negative inner products and thus fire a 0.

Small arrow on the straight line indicate the half plane that translates to +1 signal value



Example: Geometrical Design of AND Classifier

- slope $m = -w_1/w_2 = -1$, intercept $c = -w_0/w_2 = 1.5$.
- Choosing $w_1 = w_2 = 1$ and $w_0 = -1.5$ yields a neuron classifier that appropriately partitions the pattern space for AND classification.
- The value of the threshold is –1.5.

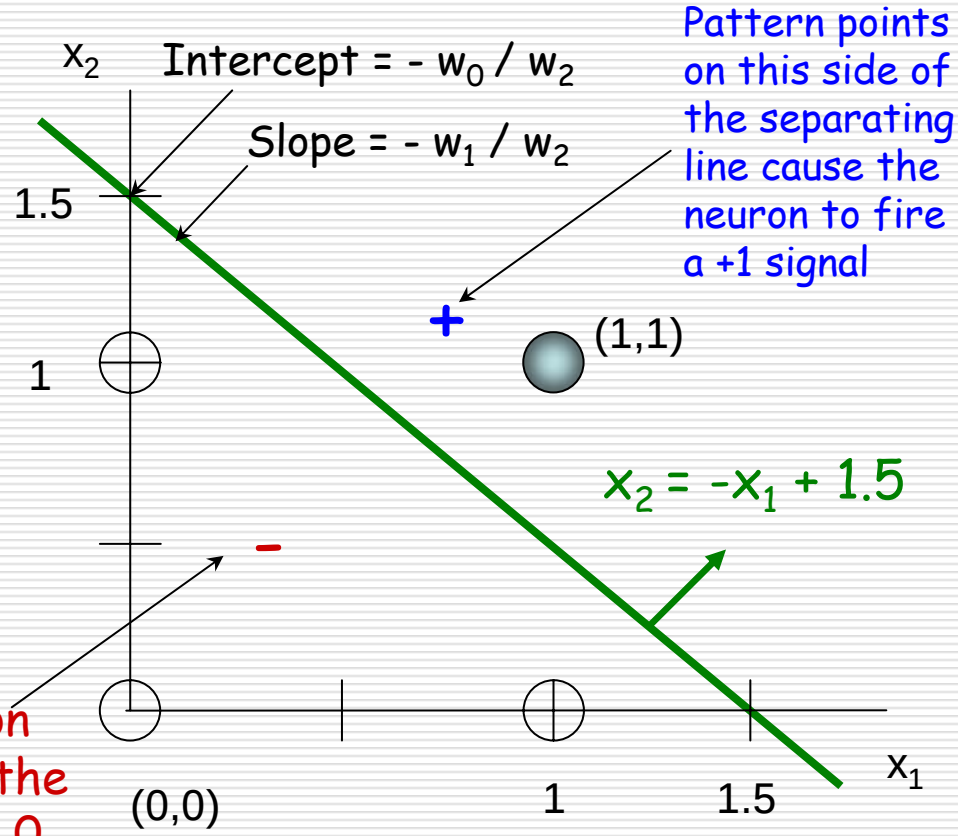


Example: Geometrical Design of AND Classifier

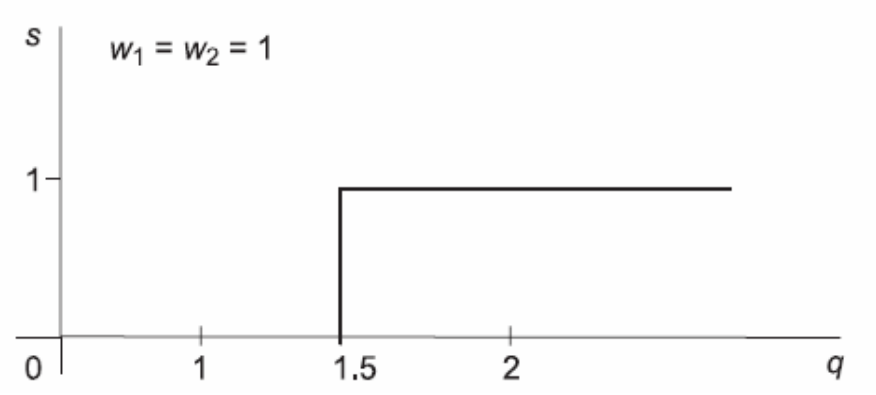
- Discriminant function thus becomes

$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_0}{w_2}$$

$$= -x_1 + 1.5$$



Solving The AND Problem By Setting The Signal Function Shift In (1,2)



x_1	x_2	$q = w_1x_1 + w_2x_2$ $(w_1 = w_2 = 1)$	$y = q - 1.5$	s
0	0	0	-1.5	0
0	1	1	-0.5	0
1	0	1	-0.5	0
1	1	2	0.5	1

- Looking at the design problem in s-q plane, s should be 1 only if q = 2 and zero otherwise.
- The signal function anywhere in the interval (1,2) i.e., 1+ε and 2-ε would get the appropriate result.
- Our design placed signal function at 1.5

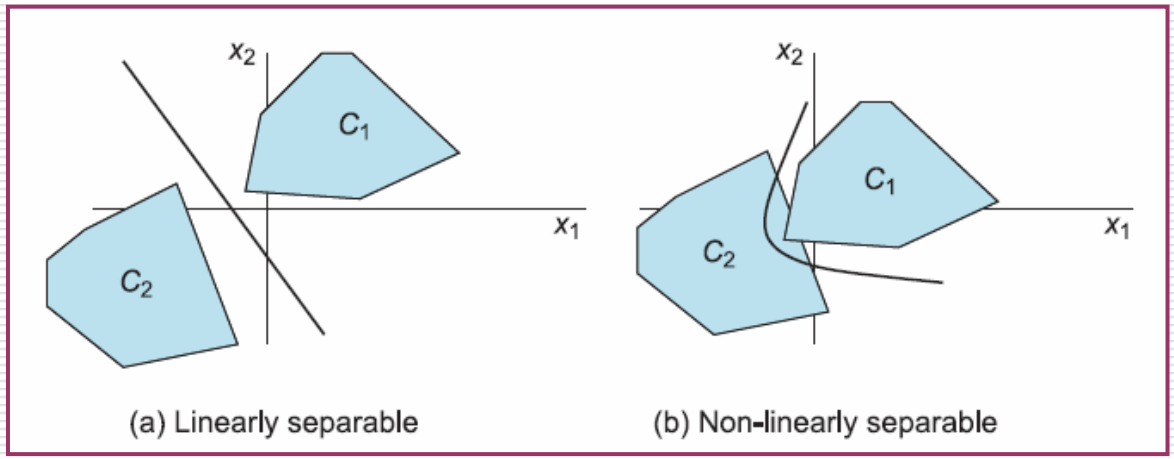
Summary of Important Properties of TLNs

- A threshold logic neuron employs a single inner product based linear discriminant function $y : R^{n+1} \rightarrow R$, $y(X) = X^T W$ where $X, W \in R^{n+1}$ and the bias or threshold value w_0, is included into the weight vector.
- The hyperplane decision surface $y(X) = 0$ divides the space into two regions, one of which the TLN assigns to class C_0 and the other to class C_1. We say that the TLN dichotomizes patterns by a hyperplane decision surface in R^n.

Summary of Important Properties of TLNs (contd.)

- The orientation of the hyperplane is determined by the weights $w_1, \ldots, w_n$.
- The position of the hyperplane is proportional to w_0.
- The distance from the hyperplane to an arbitrary point $X \in R^n$ is proportional to the value of $y(X)$.
- A pattern classifier which employs linear discriminant functions is called a linear machine.

Non-linearly Separable Problems



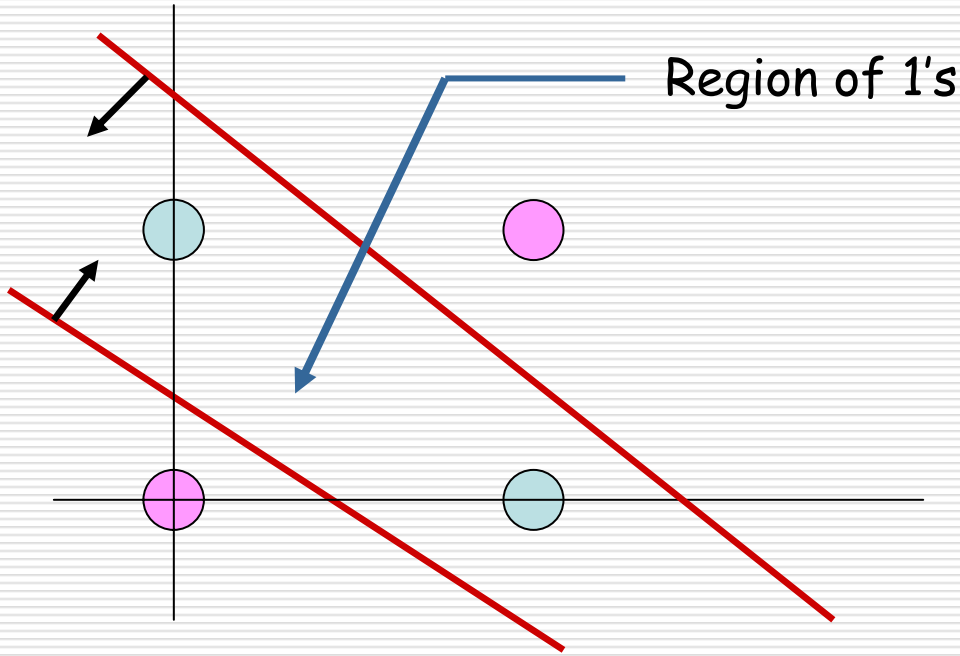
- The two convex hulls C_1 and C_2 of two pattern sets in figure (a) are sufficiently separated to ensure decision surfaces comprise only of hyperplanes.

- When convex hulls overlap as in figure (b), the pattern sets become linearly non-separable but may be non-linearly separable

XOR is Not Linearly Separable

Truth Table

x_1	x_2	S
0	0	0
0	1	1
1	0	1
1	1	0



- The geometry of the Boolean XOR function shows that two straight lines are required for proper class separation

Theorem

- *No single threshold logic neuron exists that can solve the XOR classification problem*

Proof: By Contradiction

- Note: XOR classification problem is equivalent to the simple modulo-2 addition
- Assume a TLN with weights (w_0, w_1, w_2) and inputs x_1 and x_2.
- For XOR operation $x_1 \oplus x_2 = 1 \text{ iff } w_1 x_1 + w_2 x_2 \geq -w_0$
- Since mod 2 arithmetic is symmetric,

$$x_1 \oplus x_2 = 1 \text{ iff } w_2 x_1 + w_1 x_2 \geq -w_0$$

- This implies $\frac{w_1 + w_2}{2} x_1 + \frac{w_1 + w_2}{2} x_2 \geq -w_0$

$$\Rightarrow \quad wx + w_0 \geq 0 \text{ where } w = \frac{w_1 + w_2}{2} \text{ and } x = x_1 + x_2$$

Proof: By Contradiction (contd.)

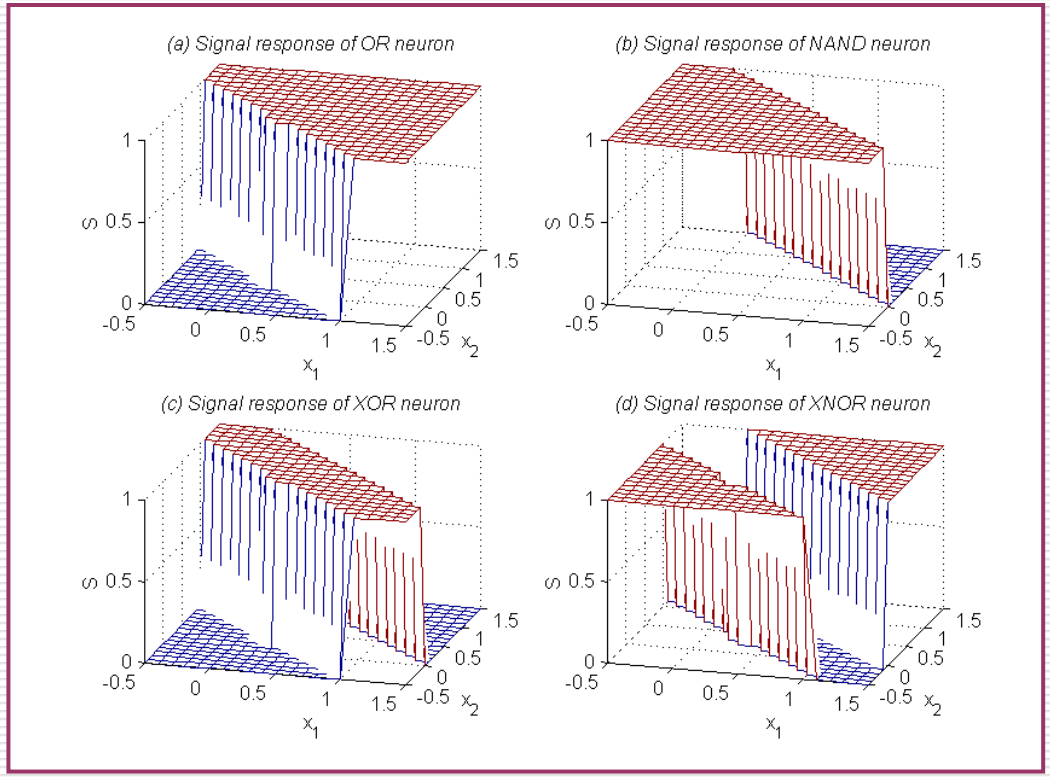
- Notice that $wx + w_0$ is a first degree polynomial in x, which must be less than zero for $x=0$ ($0 \oplus 0$), greater than zero for $x=1$ ($0 \oplus 1$ and $1 \oplus 0$), and less than zero for $x = 2$ ($1 \oplus 1$).

- This is impossible since a first degree polynomial is monotonic and cannot therefore change sign more than once.

- We thus conclude that there is no such polynomial, and that there is no TLN that can solve the XOR problem.

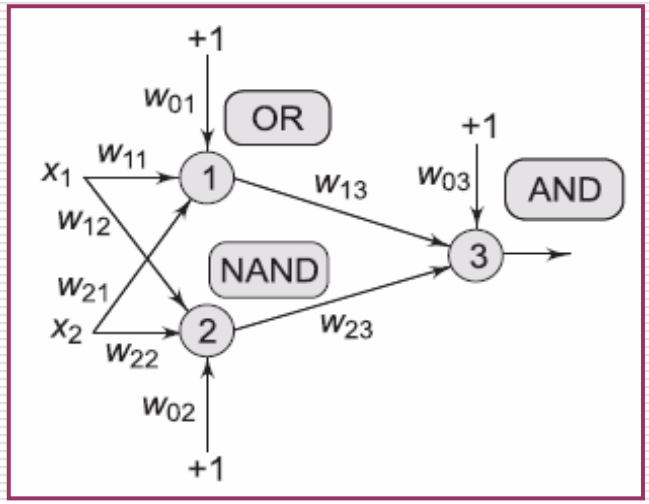
Solving XOR Problem

- OR and NAND functions realizable using a single binary neuron
- Combining OR and NAND neurons using a logical AND (which can be implemented using a third binary neuron), XOR problem can be solved

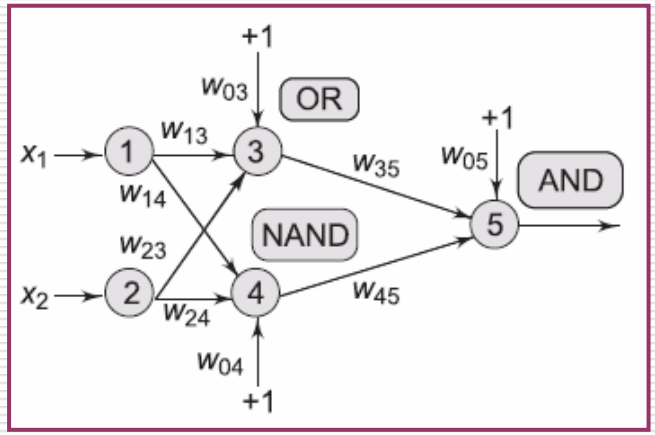


Network of TLNs to solve XOR

- A two-layered network architecture to implement the XOR function.



- A more common three-layered version of the same network with linear input units shown explicitly



Capacity of a Simple Threshold Logic Neuron

- A simple TLN implements a specific class of functions from R^n into the set {0, 1}, namely, the class of linearly separable functions.
- If we were to arbitrarily assign image points 0 or 1 to some *p* domain points, how many such random assignments could we expect the TLN to successfully implement?
- The number of such assignments that the TLN can correctly implement is called its capacity.

Points in General Position

- Denote the number of linear dichotomies that can be induced on p points in n-dimensional space as $L(p, n)$.
- $L(p, n)$ is then equal to twice the number of ways in which p points can be partitioned by an $(n - 1)$-dimensional hyperplane, since for each distinct partition, there are two different classifications.
- Assumption: No three points in space are collinear, since this would effectively reduce the number of inducible dichotomies.

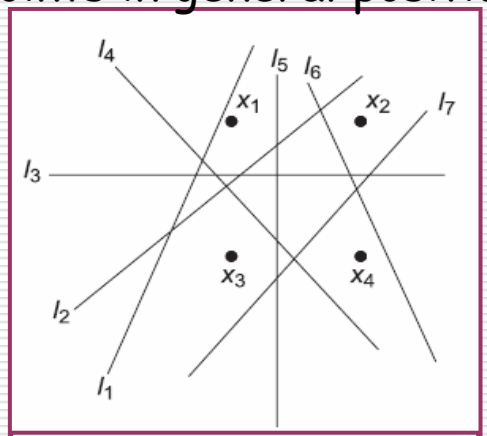
Definition: General Position

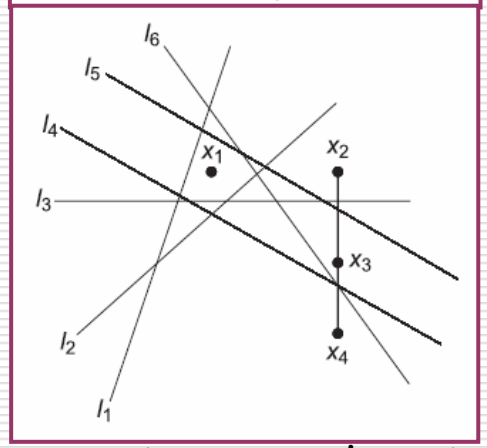
- For $p > n$, we say that a set of p points is in general position in a n-dimensional space if and only if no subset of $n + 1$ points lie on an $(n - 1)$-dimensional hyperplane.
- When $p \leq n$, a set of p points is in general position if no $(n - 2)$-dimensional hyperplane contains the set.

Example

- In top figure the lines $l_1, \ldots, l_7$ implement all possible *linear partitions* of of four points.
- Line l_3 in bottom figure could be a decision surface that implements either one of the following classifications: $\{X_1,X_2\} \in C_0$, $\{X_3,X_4\} \in C_1$; or $\{X_1,X_2\} \in C_1$ and $\{X_3,X_4\} \in C_0$.
- C_0 is the class identified by a TLN signal 0; C_1 is the class identified by a TLN signal 1.

Points in general position;





Points not in general position

The Essential Question

- Given a set of p random points in n-dimensional space, for how many of the 2^p possible $\{0, 1\}$ assignments can we find a separating hyperplane that divides one class from another?

- The answer is precisely $L(p, n)$.

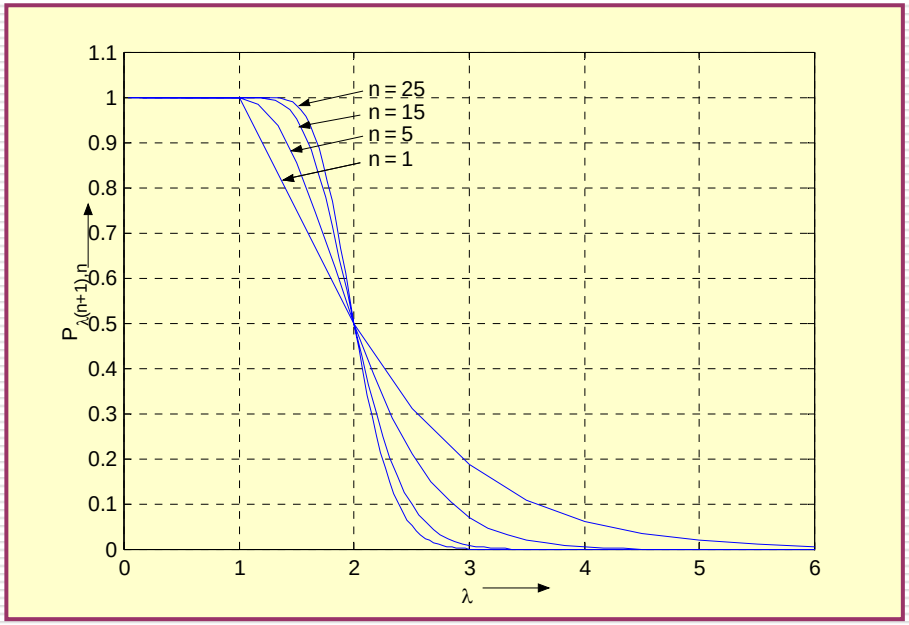
Number of Dichotomies

$$L(p,n) = 2 \sum_{i=0}^{\min(n,p-1)} \binom{p-1}{i}$$

- See text for details of the derivation

Probabilistic Estimate of Computable Dichotomies

- With p points in n-dimensions there are 2^p possible functions. The probability that they are implementable by a TLN is then directly the ratio of L(p, n) and 2^p.
- The probability that a dichotomy defined on p points in n-dimensions is computable by a TLN is



$$P(p,n) = \frac{L(p,n)}{2^p} = \begin{cases} 2^{1-p}\sum_{i=0}^{n}\binom{p-1}{i} & \text{for } p > n+1 \\ 1 & \text{for } p \le n+1 \end{cases}$$

Capacity of a TLN

$$C_{max} = 2(n+1)$$

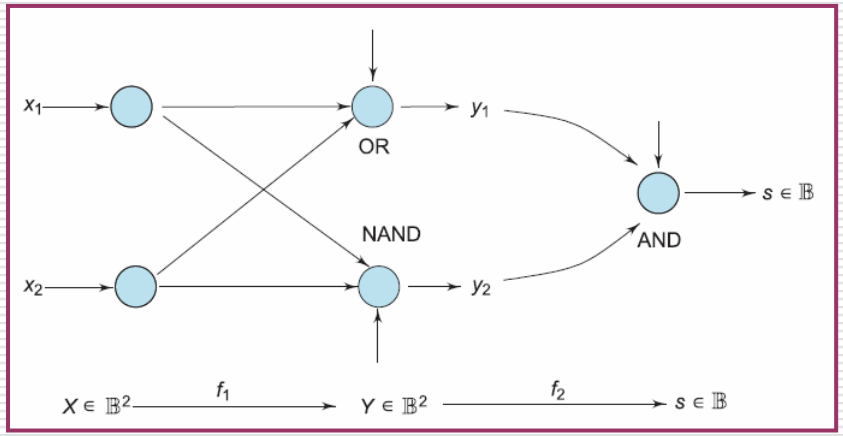
Revisiting the XOR Problem

- The number of dichotomies $L(p, n)$ of p points in n dimensions provides a direct measure of the number of Boolean functions defined on p points that are computable by a TLN with n inputs.
- For example, a TLN with 2 inputs (and a bias weight) can compute the entire set of Boolean functions that are defined on 3 points in R^2.
- Because the total number of functions that can be defined on 3 points is $2^3 = 8$; and the total number of dichotomies that can be generated by 3 points in 2 dimensions is $L(3, 2) = 8$.
- Adding a fourth point increases the total number of possible functions to $2^4 = 16$, whereas the total number of dichotomies that can be generated with 4 points in 2 dimensions is only $L(4, 2) = 14$.
- Two functions (for example, XOR and XNOR considering the functions defined on B^2) are not computable by a TLN (because they are not linearly separable).

To gain further insight…

- If we have a large number of dichotomies, the *probability* that the TLN will be able to solve a classification problem increases i.e., there is a larger chance that some dichotomy will be able to solve the problem.
- Since a TLN can solve only a linearly separable problem, to solve a linearly non-separable problem we have to make it linearly separable. How can this transformation be effected ?
- There are two ways:
  - If somehow we can reduce the number of points p, then it might be possible that the mapping from the reduced set of points into the range space may become linearly separable.
  - We might increase the dimension since the number of possible linear dichotomies then increases, and the probability that the linearly nonseparable problem at hand becomes linearly separable, increases.

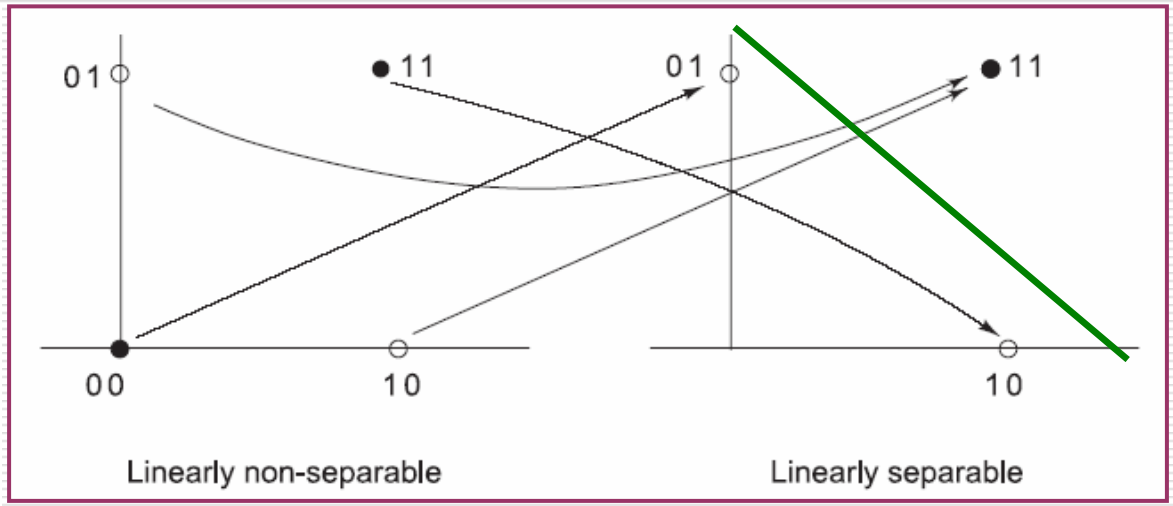
Solution 1: Reduce the Number of Points



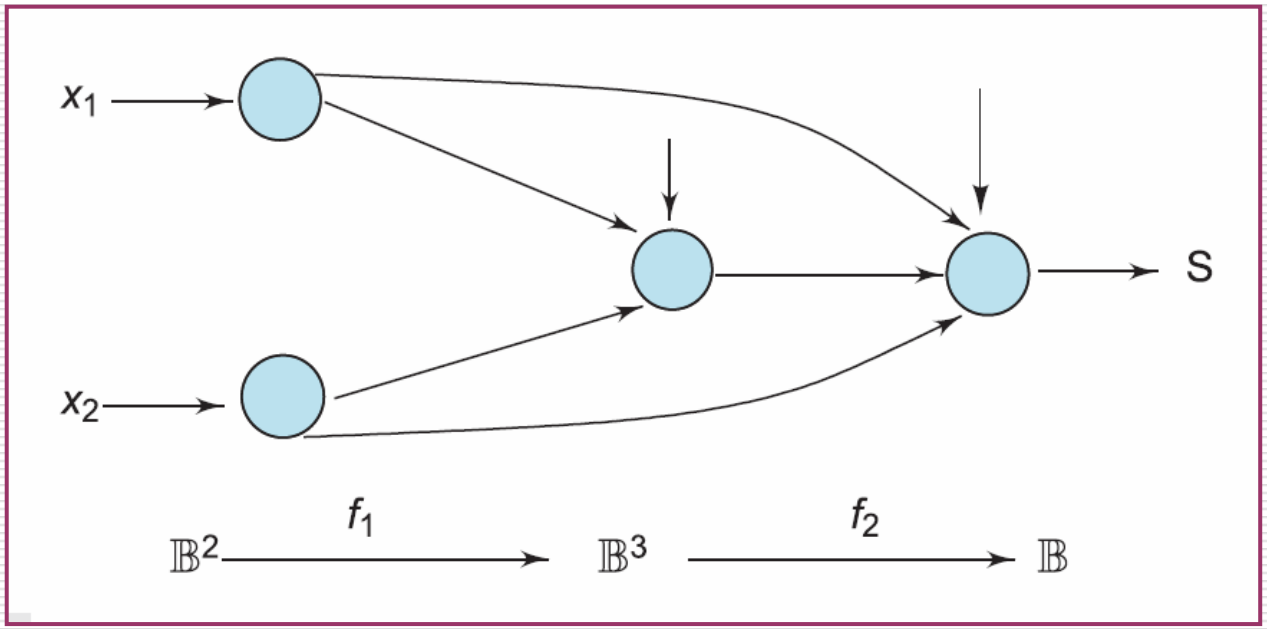
- The logical functions implemented by each TLN (first layer nodes do not compute anything) is indicated in the figure.
- The XOR problem has been solved in two steps:
  - first there is a map $f_1 : B^2 \rightarrow B^2$
  - Then, there is a second map $f_2 : B^2 \rightarrow B$
- Note that $Y = f_1(X)$ where $y_1 = x_1 + x_2; y_2 = \overline{x_1 x_2}$

Four Points In 2-d Mapped To Three Make XOR Linearly Separable

x_1	x_2	y_1	y_2	s
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0



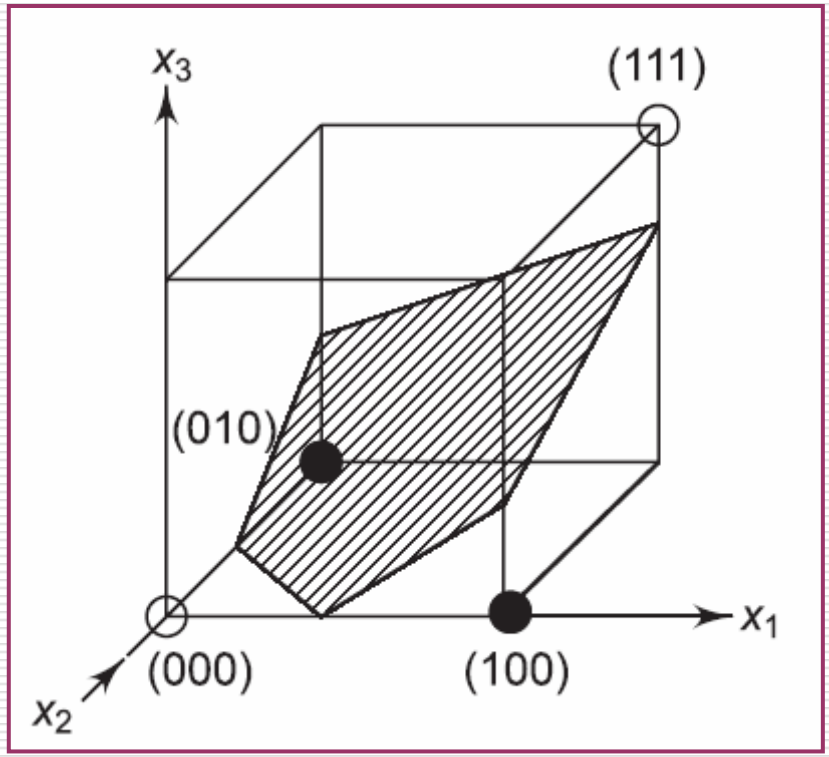
Solution 2: Increase the Dimension



- We are now looking for an appropriate mapping $f_1 : B^2 \rightarrow B^3$ such that $f_2 : B^3 \rightarrow B$ is linearly separable.

The AND Function Added As The Third Dimension

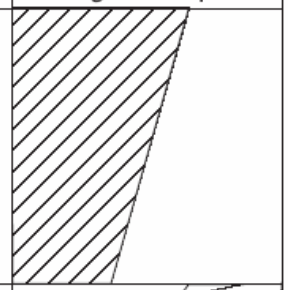
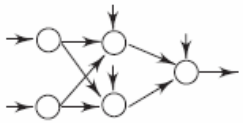
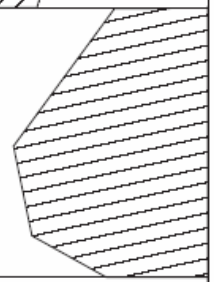
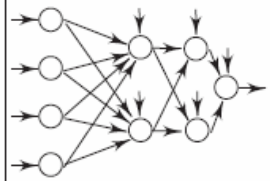
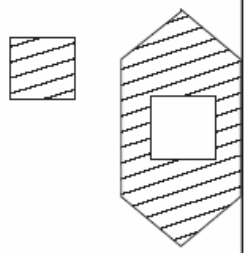
x_1	x_2	$x_3 (= x_1 x_2)$	$x_1 \oplus x_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Multilayer Networks

- TLNs can be connected into multiple layers in a feedforward fashion.
- It is common to have more than one hidden layer when solving a classification problem especially when one has a number of disjoint regions in the pattern space that are associated into a single class.
- We make the following observations for TLNs with two hidden layers apart from the input and the output layer.
  - Each neuron in the first hidden layer forms a hyperplane in the input pattern space.
  - A neuron in the second hidden layer can form a hyper-region from the outputs of the first layer neurons by performing an AND operation on the hyperplanes. These neurons can thus approximate the boundaries between pattern classes.
  - The output layer neurons can then combine disjoint pattern classes into decision regions made by the neurons in the second hidden layer by performing logical OR operations.

The Geometry Of Multilayered TLN Networks

Architecture	Types of Decision Regions	General Region Shapes
Single Layer 	Dichotomy	
Two Layer 	Open/Closed	
Three Layer 	Arbitrary	

Theorem

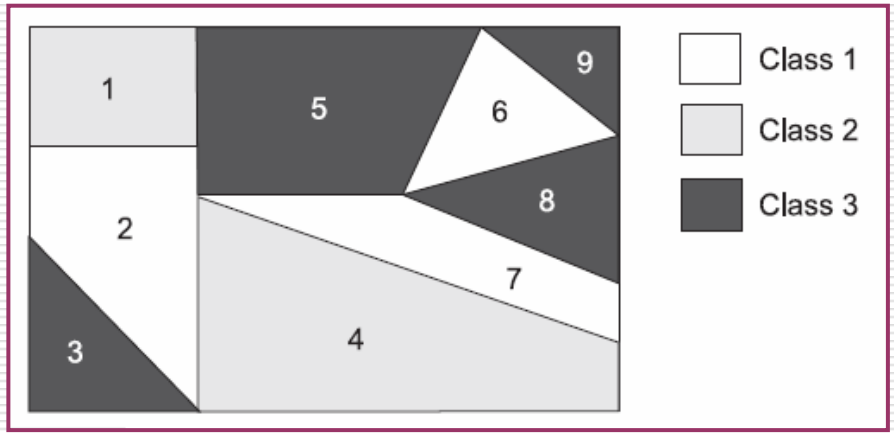
- No more than three layers in binary threshold feedforward networks are required to form arbitrarily complex decision regions.

Proof: By Construction

- Consider the n-dimensional case: $X \subset R^n$.
- Partition the desired decision regions into small hypercubes.
- Each hypercube requires 2n neurons in the first layer (one for each side of the hypercube).
- One neuron in the second layer takes the logical AND of the outputs from the first layer neurons. Outputs of second layer neurons will be high only for points within the hypercube.
- Hypercubes are assigned to the proper decision regions by connecting the outputs of second layer neurons to third layer neurons corresponding to the decision region that the hypercubes represent by taking the logical OR of appropriate second layer outputs.

How Many Hidden Nodes Are Enough? Theorem (Cao, Mirchandani, 1989)

- In n-dimensional space, the maximum number of regions that are linearly separable using h hidden nodes are



$$M(h,n) = \sum_{k=0}^{n} \binom{h}{k} \text{ where } \binom{\mathrm{h}}{\mathrm{k}} = 0, \mathrm{h} < \mathrm{k}.$$

- See text for details.