Neural Networks

Robert M. Haralick

Computer Science, Graduate Center
City University of New York

Neural Networks

Robert M. Haralick

Computer Science, Graduate Center
City University of New York

Linearly Separable

Definition

The set of points $\{x_1, \ldots, x_M\}$ is Linearly Separable from the set of points $\{y_1, \ldots, y_N\}$ if and only if there exists a w such that

$$\begin{aligned} w' x_m &> 0, \ m = 1, \ldots, M \\ w' y_n &\leq 0, \ n = 1, \ldots, N \end{aligned}$$

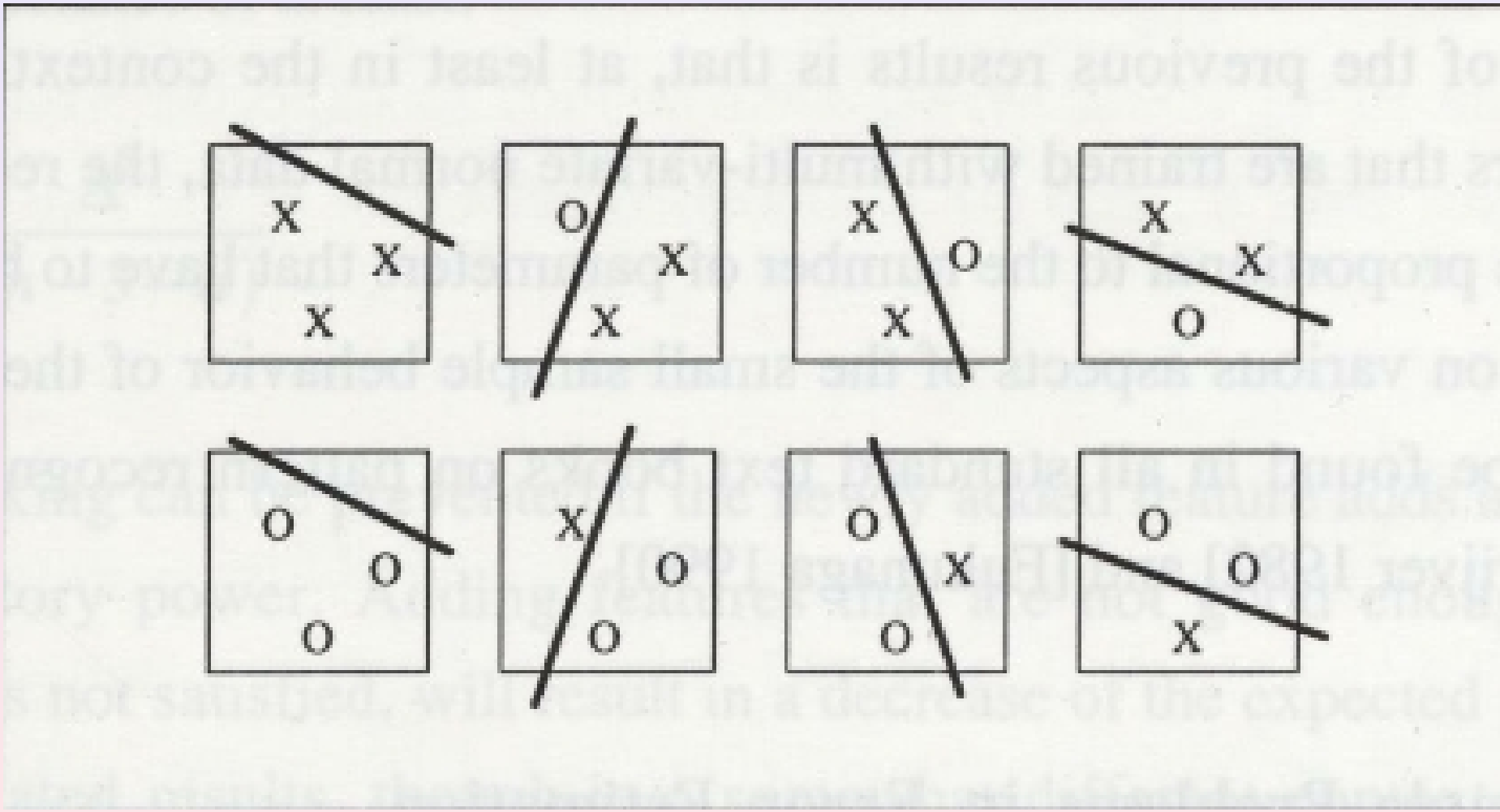
Affine Separable

Definition

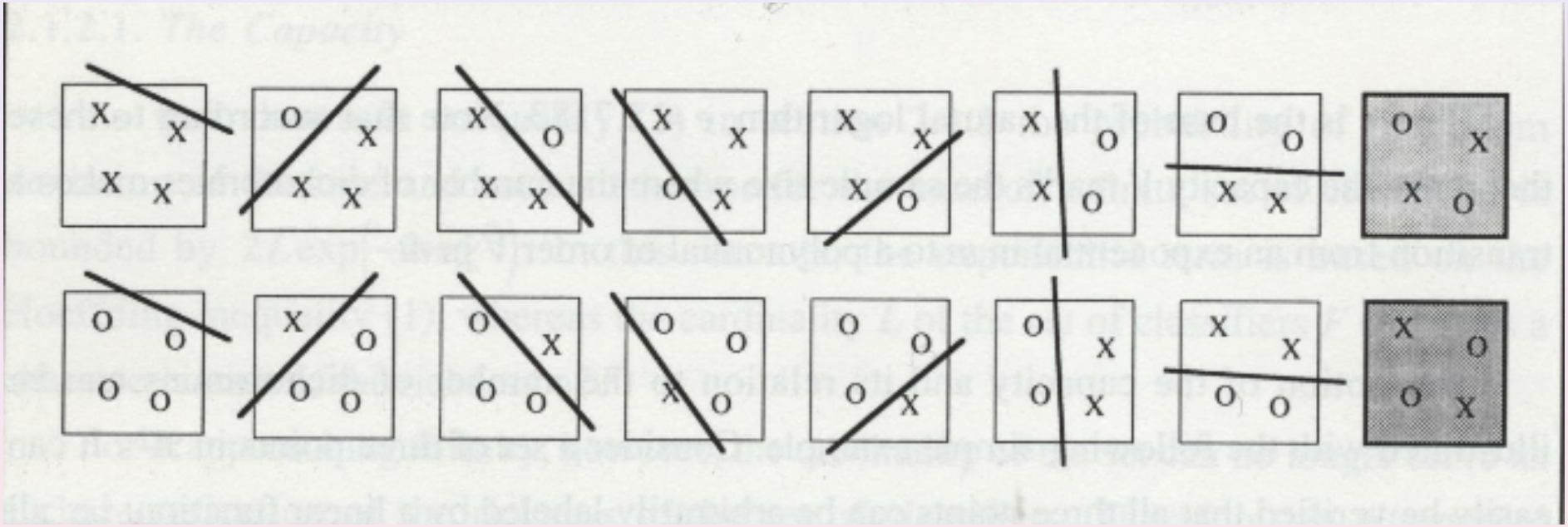
The set of points $\{x_1, \ldots, x_M\}$ is Affine Separable from the set of points $\{y_1, \ldots, y_N\}$ if and only if there exists a w and θ such that

$$\begin{aligned} w' x_m &> \theta, \ m = 1, \ldots, M \\ w' y_n &\leq \theta, \ n = 1, \ldots, N \end{aligned}$$

2D Dichotomies 3 Points Affine Separable



2D Dichotomies 4 Points Not Affine Separable



Linear and Affine Separable

Proposition

Let $x' = (x_1, x_2, \ldots, x_N)$ *and* $u' = (u_1, u_2, \ldots, u_N)$ *Define*

$$X = \{x \mid u'x > \theta\}$$

Let $y' = (y_1, y_2, \ldots, y_N, -1)$ *and* $v' = (u_1, u_2, \ldots, u_N, \theta)$ *Define*

$$Y = \{y \mid v'y > 0\}$$

Then

$$X = Y$$

Linearly Separable Dichotomies

Theorem

There are $C(N, D)$ homogeneously linearly separable dichotomies of N points in general position in Euclidean D-space, where

$$C(N, D) = 2 \sum_{k=0}^{D-1} \binom{N-1}{k}$$

Thomas Cover, *Geometrical and Statistical Properties of Systems of Non-Linear Inequalities with Applications in Pattern Recognition*,**IEEE Transactions on Electronic Computers**, Vol. EC-14, 1965, pp. 326-334.

Linearly Separable Dichomoties

- D Dimension of space
- N Number of points

$$C(N, D) = 2 \sum_{k=0}^{D-1} \binom{N-1}{k}$$

$$C(3,2) = 2\left[\binom{2}{0} + \binom{2}{1}\right]$$

$$C(3,3) = 2\left[\binom{2}{0} + \binom{2}{1} + \binom{2}{2}\right] = 2[1+2+1] = 8$$

Affine Separable Dichotomies

- D Dimension of Space
- $N+1$ Number of Points

$$\begin{aligned} C(N+1, N+1) &= 2\sum_{k=0}^{N}\binom{N}{k} \\ &= 2^N \end{aligned}$$

Theorem

Any $N+1$ distinct points are affine separable in an N-dimensional space.

Multilayer Perception

Definition

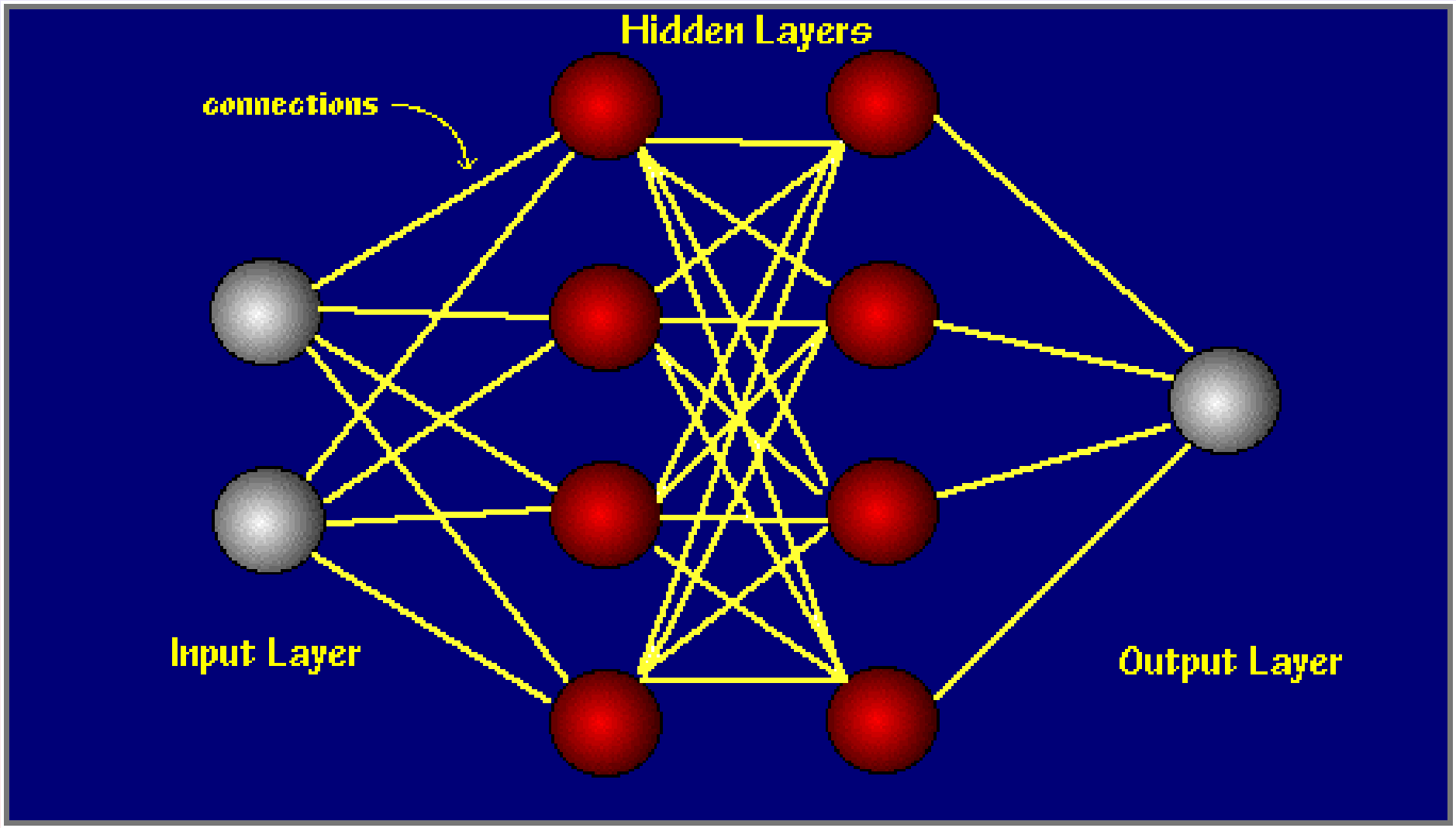
A Multilayer Perceptron consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a processing element with a nonlinear activation function.

Hidden Layer

Definition

A Hidden Layer of a multilayer perceptron is any layer that has no direct connection to both the Input and Output Layers.

Network With Hidden Layer



Generality

Theorem

A Multi-layer perceptron with at least one hidden layer, with sigmoidal or Gaussian non-linearities, can approximate any Borel measurable function.

- The theorem says it is possible
- It does not say how
- The number of hidden units may be very large

(Funahashi, 1989), (Hornik, 1989), (Hartman, 1990), (Park, 1991)

Finite Continuous Multivariate Representation Theorem

Kolmogorov and Arnold

Theorem

Any continuous function $g(x)$ defined on the N-dimensional unit hypercube can be represented in the form

$$g(x_1, \ldots, x_N) = \sum_{n=1}^{2N+1} \phi_n \left(\sum_{m=1}^{N} \psi_{mn}(x_m) \right)$$

Finite Continuous Multivariate Representation Theorem

Lorentz

Theorem

Any continuous function $g(x)$ defined on the N-dimensional unit hypercube can be represented in the form

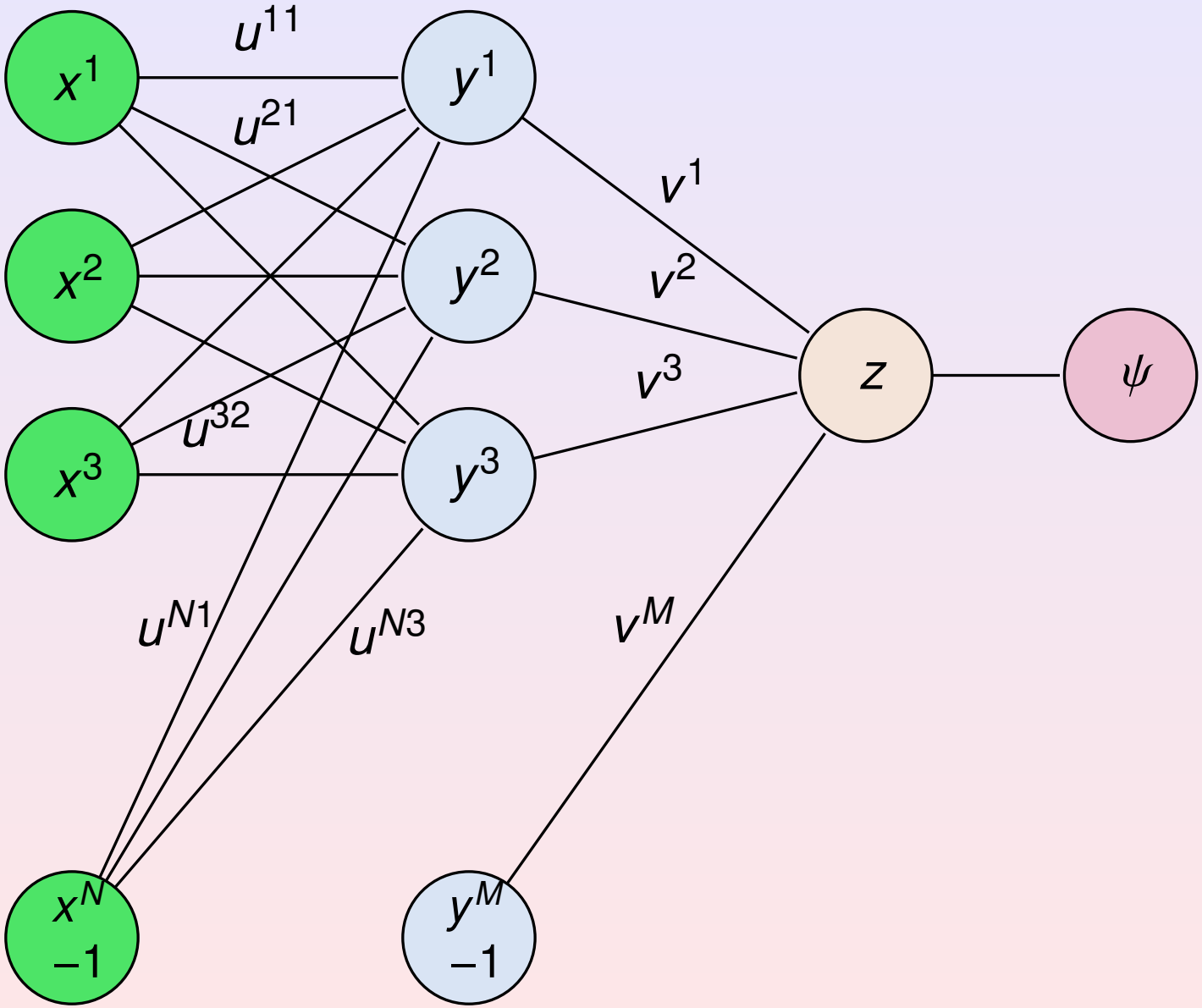
$$g(x_1, \ldots, x_N) = \sum_{n=1}^{2N+1} \phi\left(\sum_{m=1}^{N} \psi_{mn}(x_m)\right)$$

- *Theorem does not say how to determine the functions*
- *The functions may very complicated and not smooth*
- *Back propagation algorithm may not work*

Back Propagation

- Rosenblatt introduced Perceptrons beginning in the late 1950's
- Funded by the Office of Naval Research
- Done at Cornell facility at Buffalo NY
- Done at Cornell, Ithaca
- Computers at NYU
- Input layer, middle layer, output layer
- Minsky and Papert wrote the book *Perceptrons* in 1969
  - Explaining all the simple things the Rosenblatt Perceptrons could not do
- Rummelhart (1969) designed an iterative learning algorithm to train multi-layer networks: Back Propagation

Simple 4 Layer Network



Back Propagation Network

$$
\begin{aligned}
u_m &= \begin{pmatrix} u^{1m} \\ u^{2m} \\ \vdots \\ u^{Nm} \end{pmatrix}, m = 1, \ldots, M-1 \\
y^m &= \begin{cases} \frac{1}{1+e^{-u'_m x}} & m = 1, \ldots, M-1 \\ -1 & m = M \end{cases} \\
z &= \frac{1}{1+e^{-v'y}}
\end{aligned}
$$

Derivative of Sigmoidal Activation Function

$$
\begin{aligned}
f(x) &= \frac{1}{1+e^{-x}} \\
\frac{\partial f}{\partial x}(x) &= \frac{-e^{-x}(-1)}{(1+e^{-x})^2} \\
&= \frac{1+e^{-x}-1}{(1+e^{-x})^2} \\
&= \frac{1+e^{-x}}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} - \frac{1}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) \\
&= f(x)(1-f(x))
\end{aligned}
$$

Back Propagation Network

$$
\begin{aligned}
f(v'y) &= \frac{1}{1+e^{-v'y}} \\
\frac{\partial f}{\partial v}(v'y) &= \frac{-e^{-v'y}(-y)}{(1+e^{-v'y})^2} \\
&= \left(\frac{1+e^{-v'y}}{(1+e^{-v'y})^2} - \frac{1}{(1+e^{-v'y})^2}\right) y \\
&= \left(\frac{1}{1+e^{-v'y}} - \frac{1}{(1+e^{-v'y})^2}\right) y \\
&= \left(\frac{1}{1+e^{-v'y}}(1 - \frac{1}{1+e^{-v'y}})\right) y \\
&= f(v'y)(1 - f(v'y))y
\end{aligned}
$$

Back Propagation Network Error

- x is input
- c is the desired output
- E is the error function

$$
\begin{aligned}
z(v'y) &= f(v'y) = \frac{1}{1+e^{-v'y}} \\
E(z(y;v)) &= \frac{1}{2}(c - f(v'y))^2 \\
\frac{\partial E}{\partial v}(v'y) &= -(c - f(v'y))\frac{\partial f}{\partial v}(v'y) \\
&= -(c - f(v'y))f(v'y)(1 - f(v'y))y \\
&= f(v'y)(1 - f(v'y))(f(v'y) - c)y \\
&= z(1-z)(z-c)y \\
\frac{\partial E}{\partial v^m} &= z(1-z)(z-c)y^m \\
&= \delta_2 y^m
\end{aligned}
$$

Back Propagation Network Update

$$v^m \leftarrow v^m - \alpha \delta_2 y^m$$

Back Propagation Network Error

$$
\begin{aligned}
u_m &= \begin{pmatrix} u^{1m} \\ u^{2m} \\ \vdots \\ u^{Nm} \end{pmatrix} \\
y^m &= f(u'_m x) \\
E &= \frac{1}{2}(f(\sum_{m=1}^{M} v^m y^m) - c)^2 \\
\frac{\partial E}{\partial u^{ij}} &= \frac{\partial}{\partial u^{ij}} \frac{1}{2}(f(\sum_{m=1}^{M} v^m y^m) - c)^2 \\
&= (f(\sum_{m=1}^{M} v^m y^m) - c) \frac{\partial}{\partial u^{ij}} (f(\sum_{m=1}^{M} v^m y^m) - c)
\end{aligned}
$$

Back Propagation Network Error

$$
\begin{aligned}
\frac{\partial E}{\partial u^{ij}} &= (f(\sum_{m=1}^{M} v^m y^m) - c)\frac{\partial}{\partial u^{ij}}(f(\sum_{m=1}^{M} v^m y^m) - c) \\
\frac{\partial E}{\partial u^{ij}} &= (z - c)\frac{\partial}{\partial u^{ij}} f(\sum_{m=1}^{M} v^m y^m) \\
&= (z - c) f(\sum_{m=1}^{M} v^m y^m)(1 - f(\sum_{m=1}^{M} v^m y^m))\frac{\partial}{\partial u^{ij}} \sum_{m=1}^{M} v^m y^m \\
&= (z - c) z (1 - z)\frac{\partial}{\partial u^{ij}} \sum_{m=1}^{M} v^m y^m \\
&= (z - c) z (1 - z)\frac{\partial}{\partial u^{ij}} \left(\sum_{m=1}^{M-1} v^m f(u'_m x) - v^M \right)
\end{aligned}
$$

Back Propagation Network Error

For $i = 1, \ldots, N,\ j = 1, \ldots, M-1$

$$\begin{aligned}
\frac{\partial E}{\partial u^{ij}} &= (z-c)z(1-z)\frac{\partial}{\partial u^{ij}}\sum_{m=1}^{M-1} v^m f(u'_m x) \\
&= (z-c)z(1-z)\frac{\partial}{\partial u^{ij}}\sum_{m=1}^{M-1} v^m f(\sum_{n=1}^{N} x^n u^{nm}) \\
&= (z-c)z(1-z)\sum_{m=1}^{M-1}\frac{\partial}{\partial u^{ij}} v^m f(\sum_{n=1}^{N} x^n u^{nm}) \\
&= (z-c)z(1-z)\sum_{m=1}^{M-1} v^m \frac{\partial}{\partial u^{ij}} f(\sum_{n=1}^{N} x^n u^{nm})
\end{aligned}$$

$$= (z-c)z(1-z)\sum_{m=1}^{M-1} v^m f(\sum_{n=1}^{N} x^n u^{nm})(1-f(\sum_{n=1}^{N} x^n u^{nm}))\frac{\partial}{\partial u^{ij}}\sum_{n=1}^{N} x^n u^{nm}$$

Back Propagation Network Error

For $i = 1, \ldots, N,\ j = 1, \ldots, M-1$

$$
\begin{aligned}
\delta_2 &= (z-c)z(1-z) \\
\frac{\partial E}{\partial u^{ij}} &= \delta_2 \sum_{m=1}^{M-1} v^m f\Big(\sum_{n=1}^{N} x^n u^{nm}\Big)\Big(1 - f\Big(\sum_{n=1}^{N} x^n u^{nm}\Big)\Big)\frac{\partial}{\partial u^{ij}} \sum_{n=1}^{N} x^n u^{nm} \\
&= \delta_2 \sum_{m=1}^{M-1} v^m y^m (1-y^m) \frac{\partial}{\partial u^{ij}} \sum_{n=1}^{N} x^n u^{nm} \\
&= \delta_2 v^j y^j (1-y^j) x^i \\
\delta_{1j} &= y^j (1-y^j) \\
\frac{\partial E}{\partial u^{ij}} &= \delta_{1j} \delta_2 v^j x^i
\end{aligned}
$$

Back Propagation Update

For $i = 1, \ldots, N,\ j = 1, \ldots, M-1$

$$u^{ij} \leftarrow u^{ij} - \alpha \delta_{1j} \delta_2 v^j x^i$$

Paradigm Shift

Neural networks models marked a paradigm shift from high-level (symbolic) artificial intelligence, characterized by expert systems with knowledge embodied in if-then rules, to low-level (sub-symbolic) machine learning, characterized by knowledge embodied in the parameters of a dynamical system.

No Biologic Analog

It is interesting that although the original motivation for neural networks came from biology, there no known biologic analog for the back propagation calculation. F. Crick, *The Recent Excitement About Neural Networks*, **Nature**, Vol 337, January 1989

Criticisms

- Neural Networks are a black box
- Functional relationship between input and output is not made explicit
- Functional relationship between input and output is too complicated
- The importance of each input variable is not revealed
- Requires a lot of data
- There is no theoretical way to determine the structure of the network