UNIVERSITÀ PARTHENOPE
DiST
Dipartimento di Scienze e Tecnologie

Machine Learning (part II)

Multi-Layer Neural Network

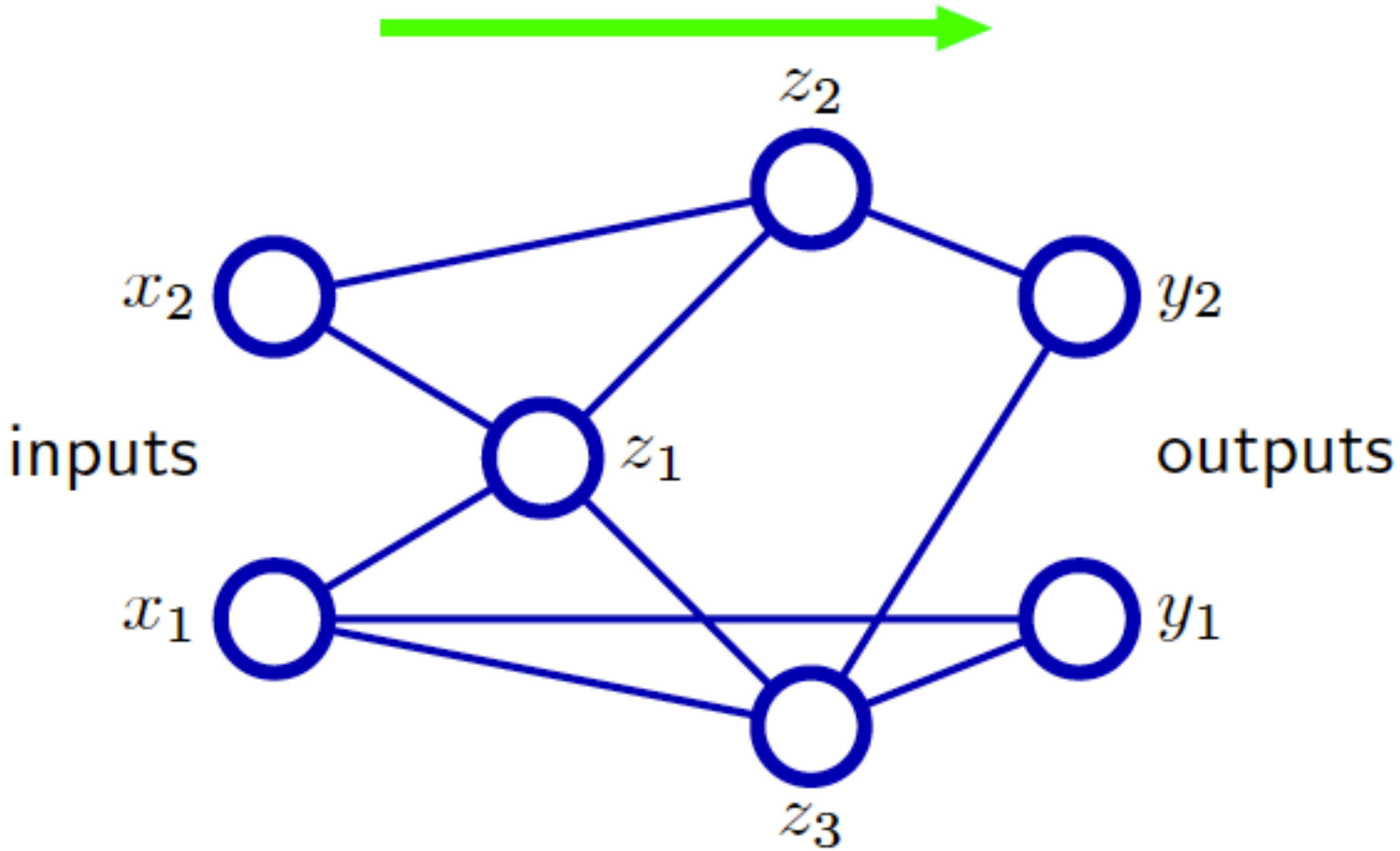
Angelo Ciaramella

Introduction

- Feed-forward Neural Network
  - Multi-Layer Perceptron

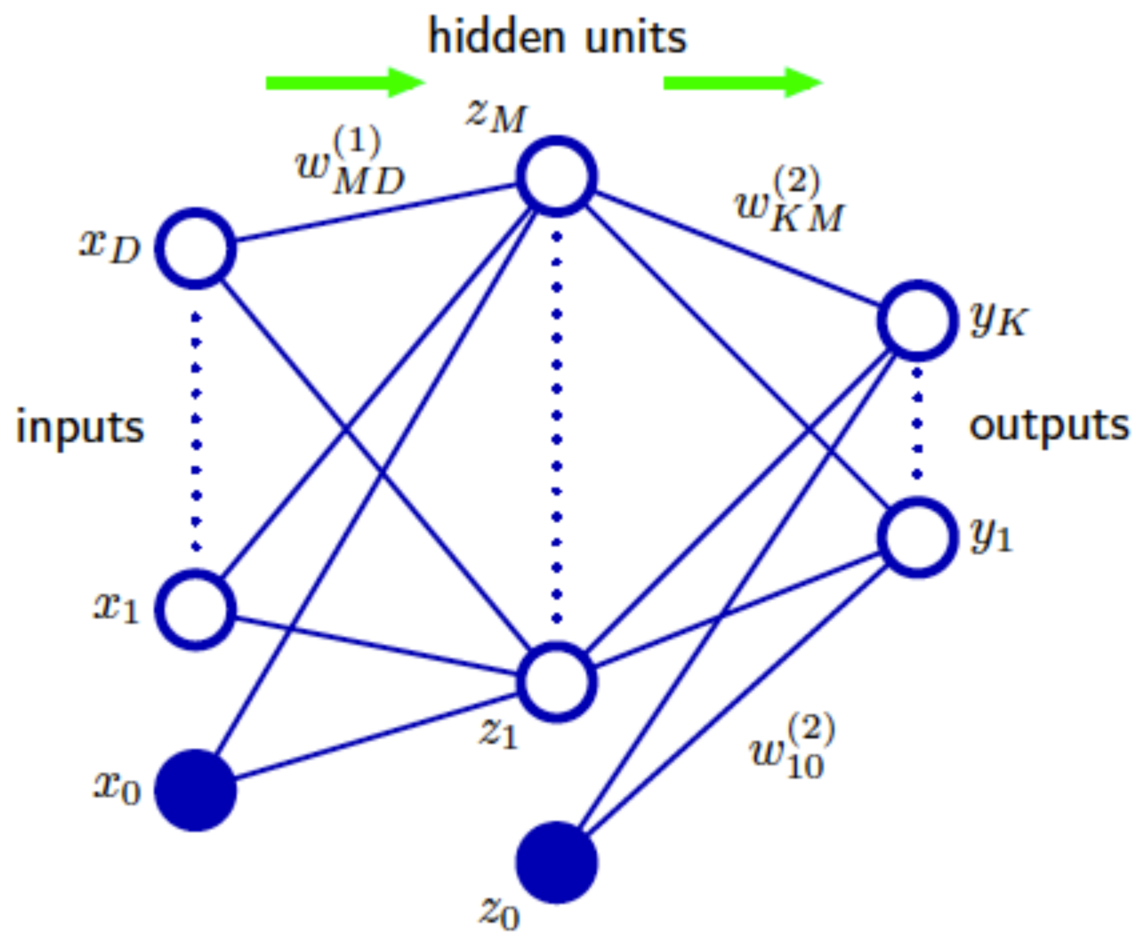
- Learning
  - error backpropagation

General feed-forward topology



MLP architecture

Architecture



MLP architecture

Neurons' activation

- Combination of input variables (first hidden level)

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

- Activation of the hidden unit

$$z_j = h(a_j)$$

- Combiantion of hidden units

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

Neurons' activation

- Output unit

$$y_k = \sigma(a_k) \qquad\qquad \sigma(a) = \frac{1}{1 + \exp(-a)}$$

- Overall function network

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

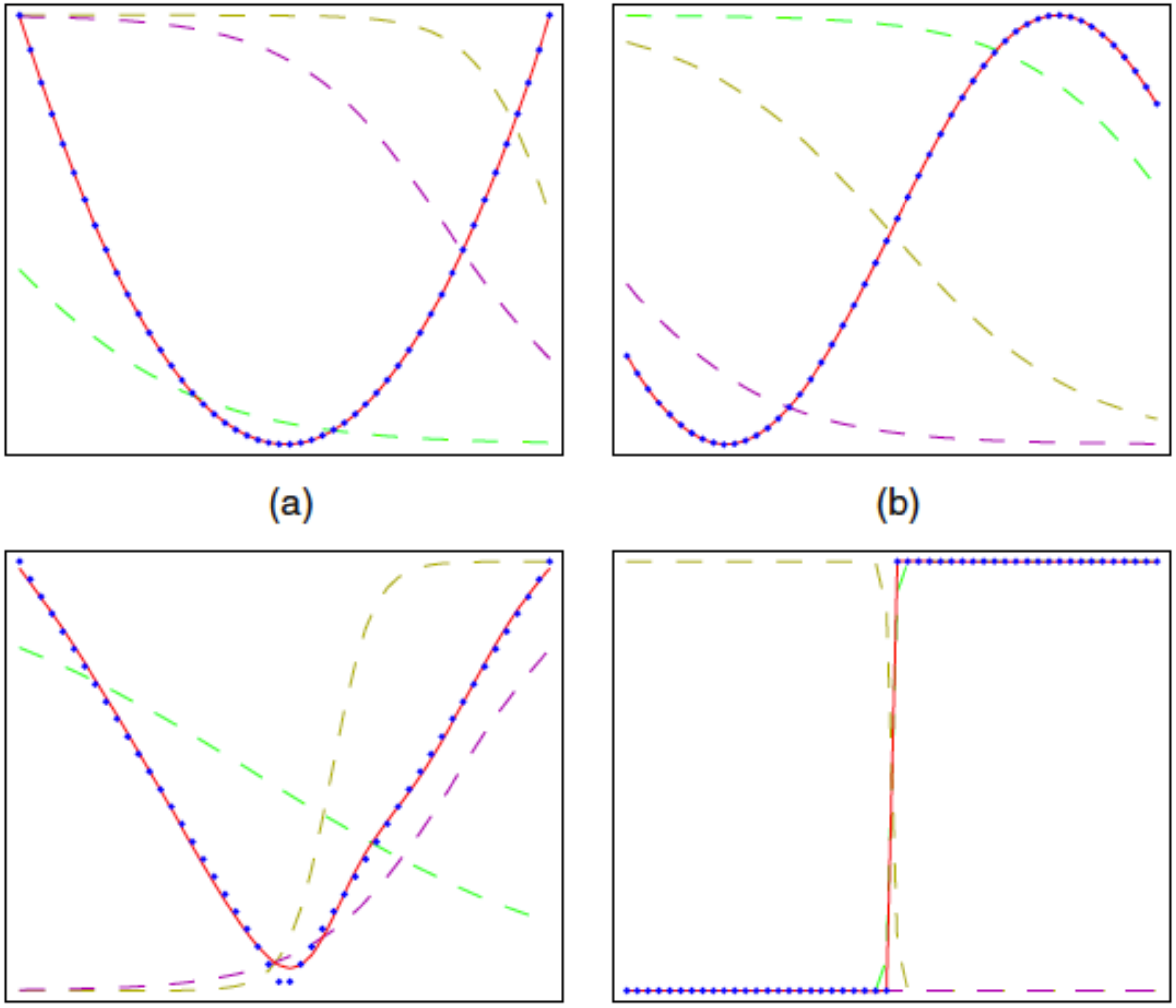
Neurons' activation

- Absorbing bias

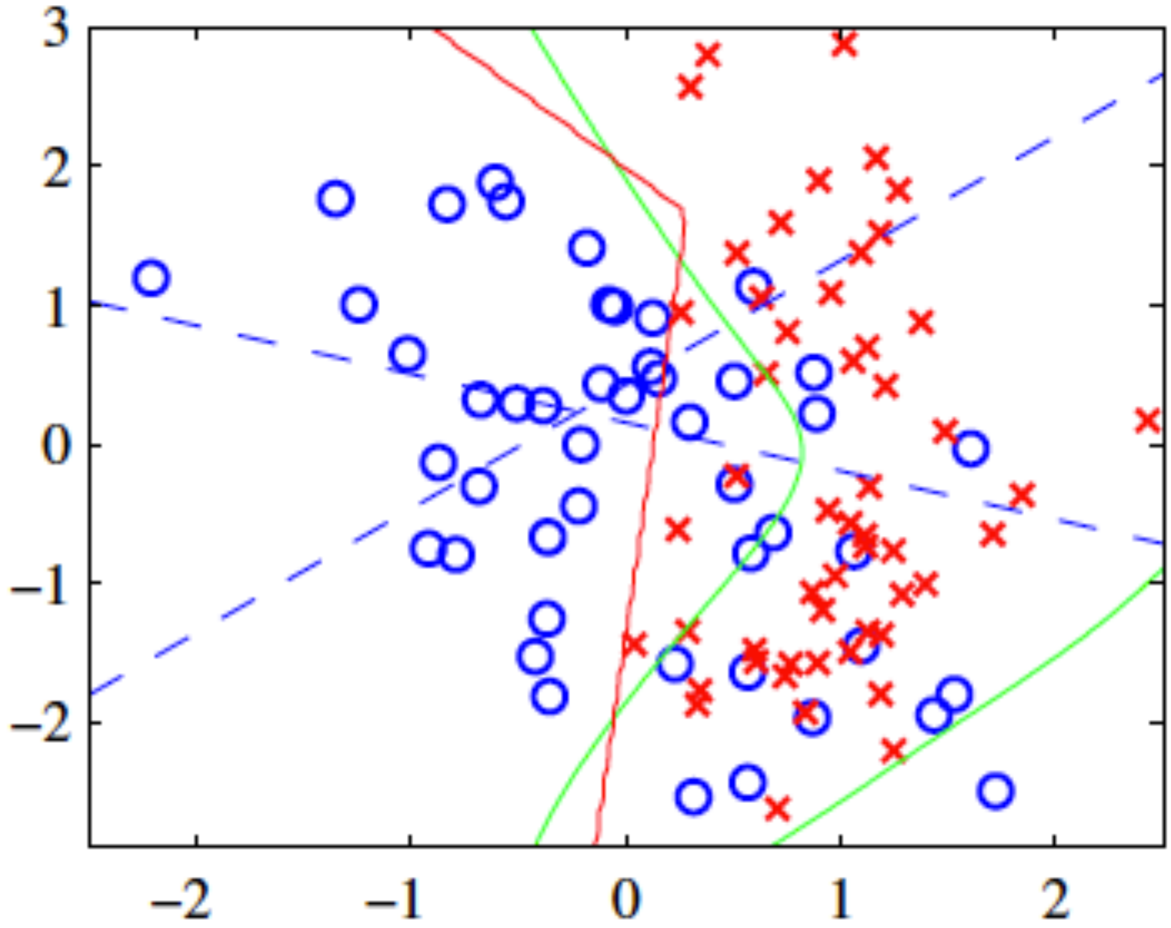
$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

- MLP are general parametric non-linear functions

Function approximation


(a) (b) (c) (d)

Classification problem

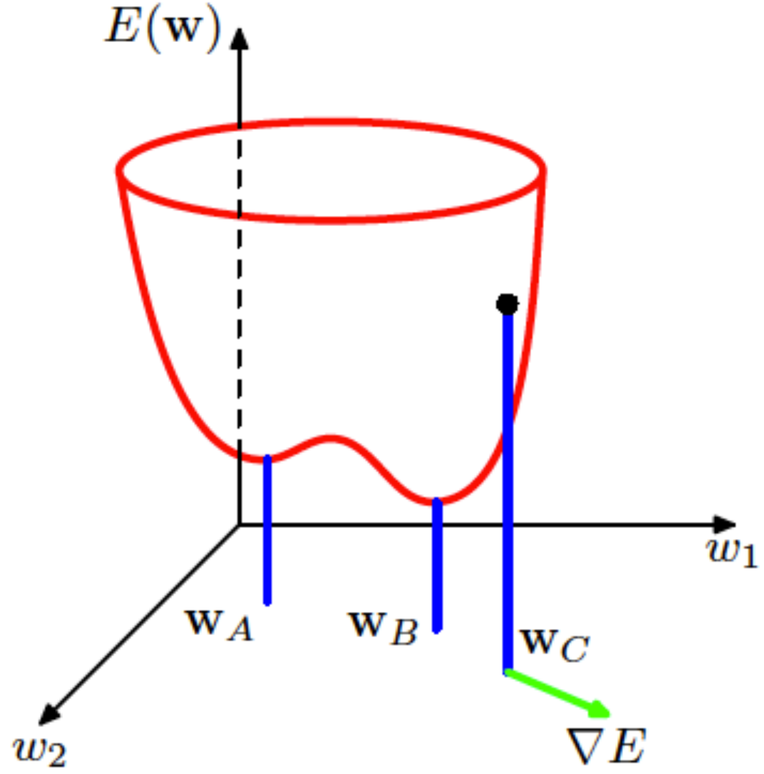


Error function

- Minimize the error function

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2$$

- Geometrical wiev



Parameters optimization

- Gradient of the error function

$$\nabla E(\mathbf{w}) = 0$$

- Gradient descent optimization

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

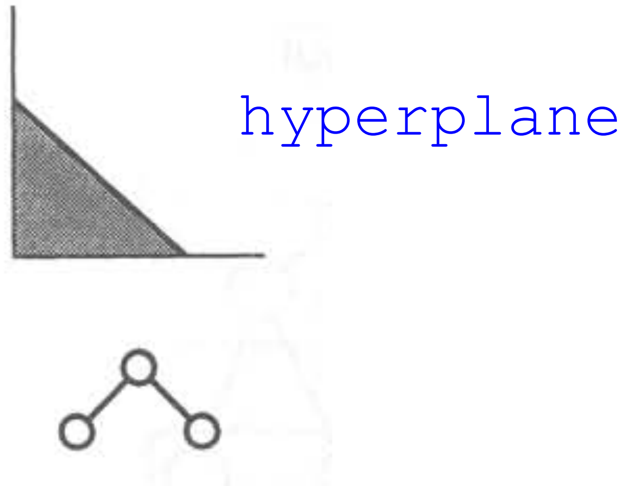
- Sequencial

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}) \qquad \mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)})$$

Number of layers

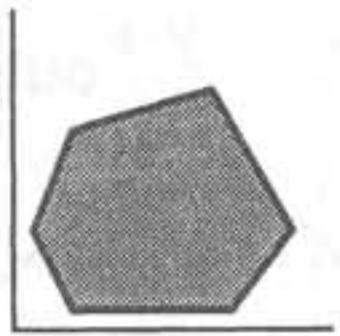
- Decision boundary by
  - Continuous input variables
  - Units with threshold activation functions

- Single layer of weights

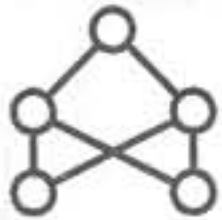


Number of layers

- Two layers of weights



Convex region of the input space



AND of hyperplanes

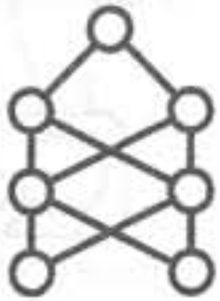
Two-layer net

- *Hidden units*
  - *Divides the input space with a hyperplane*
    - $z = 0$ *and* $z = 1$
- *Logical AND*
  - *M hidden neurons and bias = -M*
  - *output unit has 1 only if all the hidden units have oputput 1*

Number of layers

- Three layers of weights

Non-convex and disjoint regions

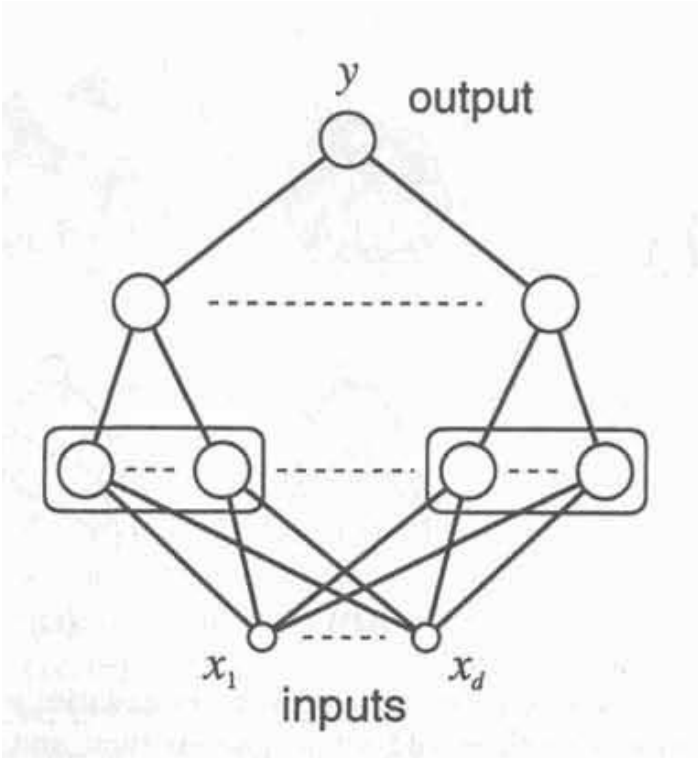


OR of hyperplanes

AND of hyperplanes

Three-layer nets

- *Result*
  - Three-layer of weights can generate arbitrary decision regions, which may be non-convex and disjoint (Lippmann, 1987)

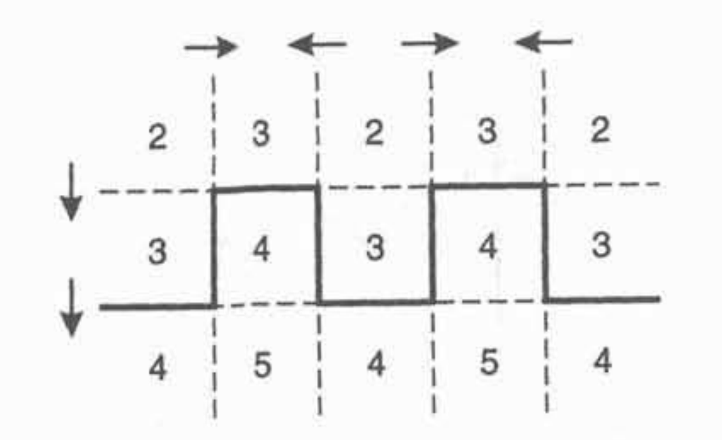


Topolgy of NN

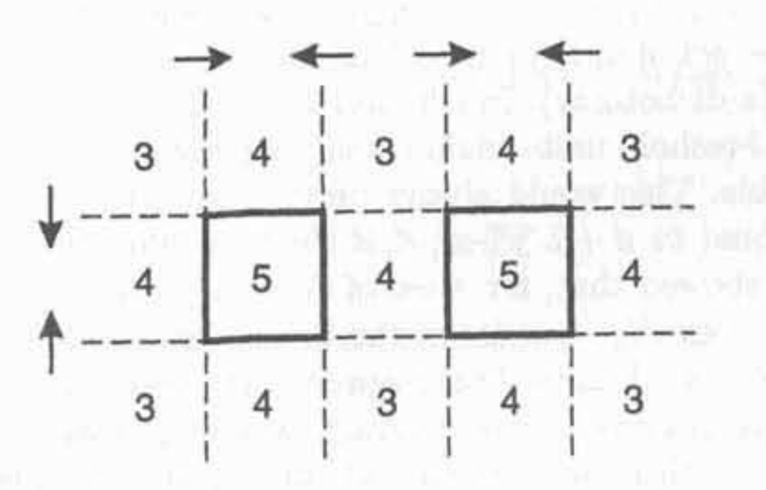
Three-layer nets

- *Input space*
  - *divided into a fine grid of hypercubes labelled as classes C_1 or C_2*
- *First hidden layer*
  - One group of fisrt-layer units is assign to each hypercube which corresponding to C_1
- *Second hidden layer*
  - units generate AND
- *Output*
  - The output unit has a *bias* = -1 for computing OR

Relaxing AND in two-layers NN

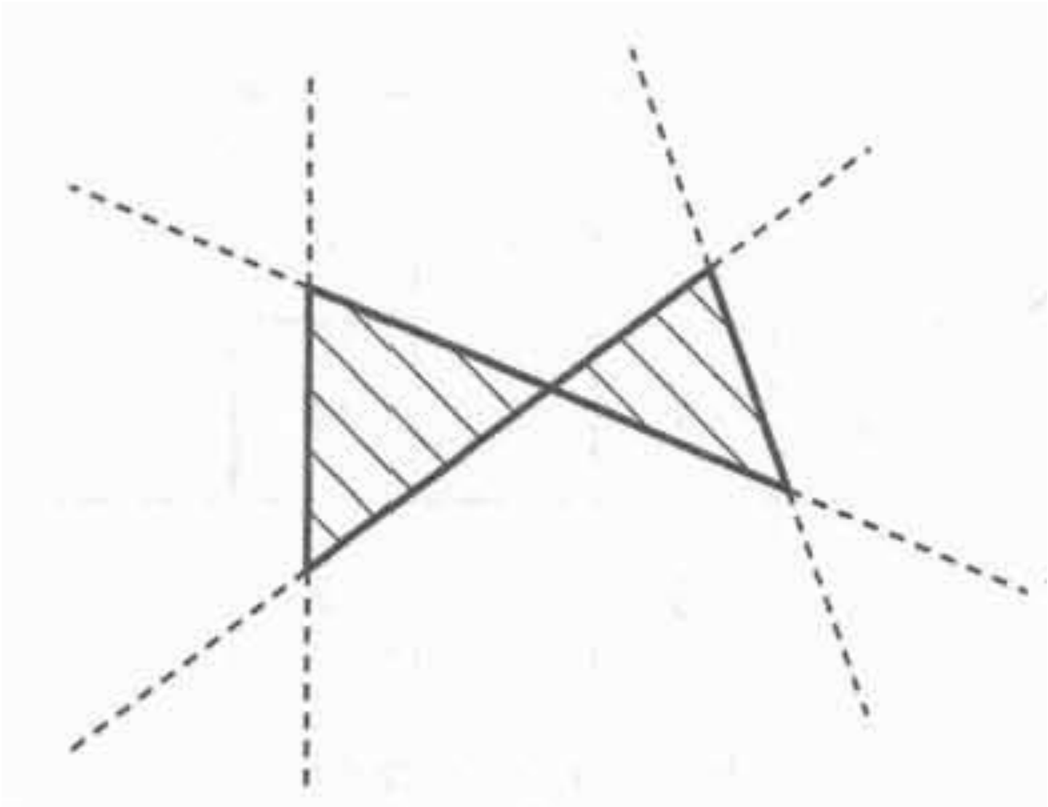


Non-convex region
bias = - 3.5



Non-convex region
bias = - 4.5

Two-layers of weights

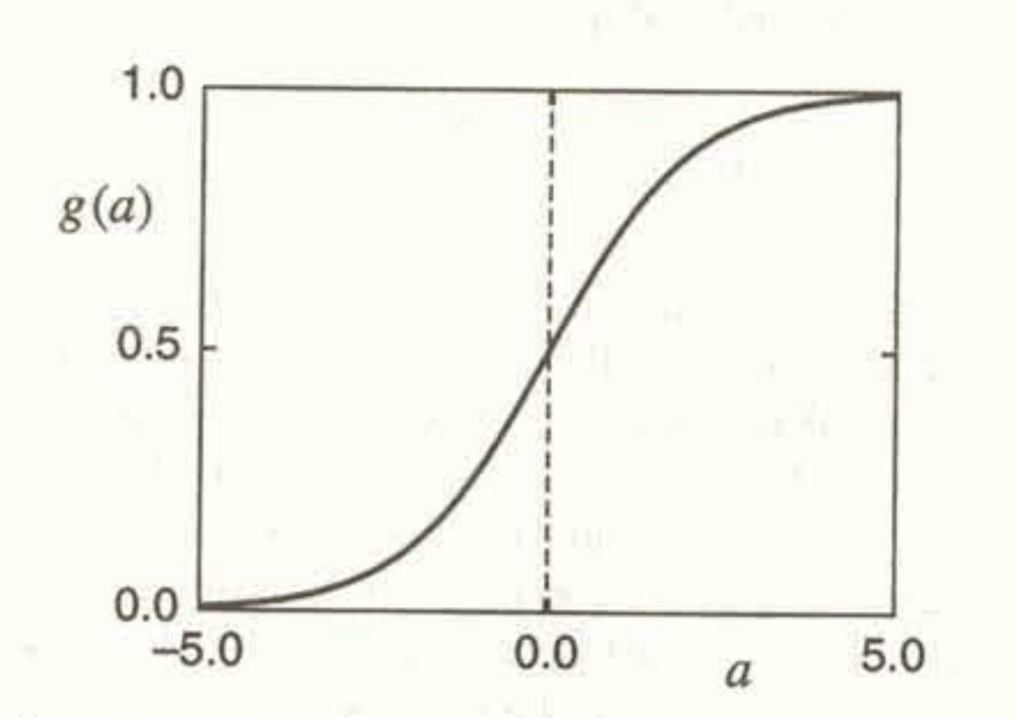


Decision boundary which cannot be produced by a network having two layers of threshold units

Sigmoidal units

- Logistic sigmoid activation function

$$g(a) = \frac{1}{1 + e^{-a}}$$

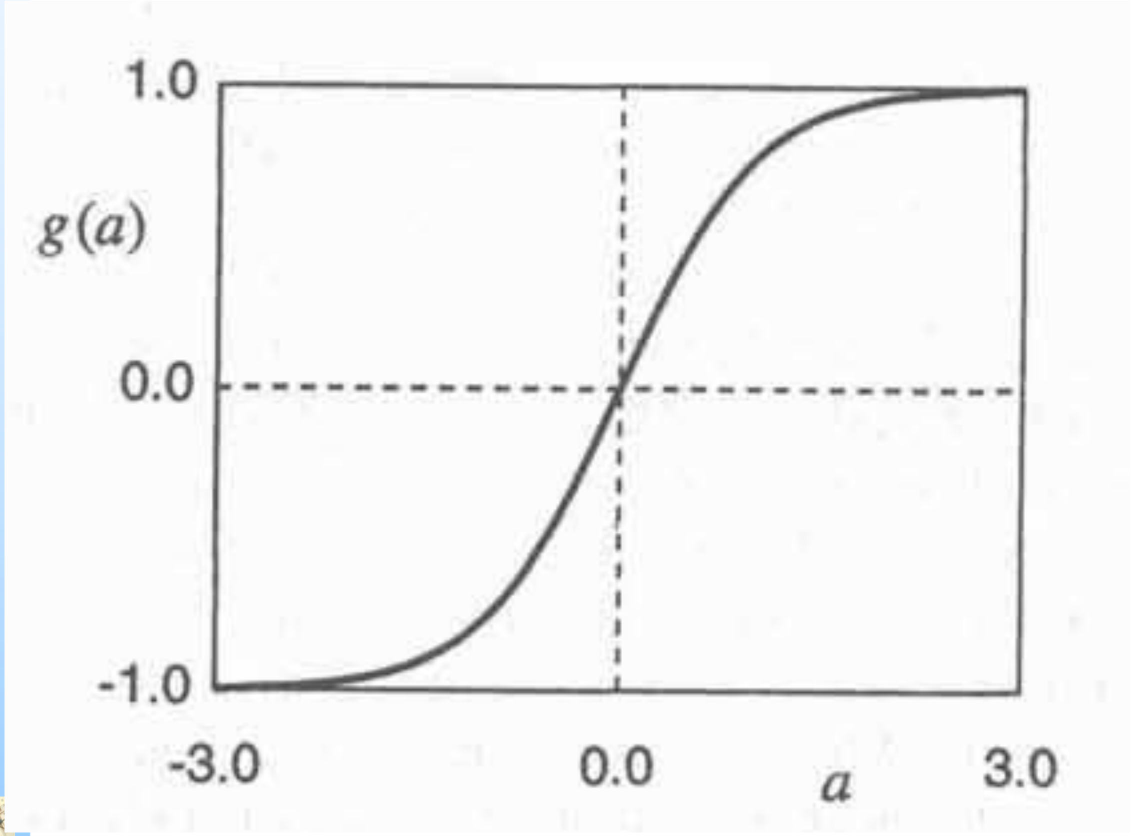


Logistic sigmoid activation function

tanh units

- *tanh* activation function

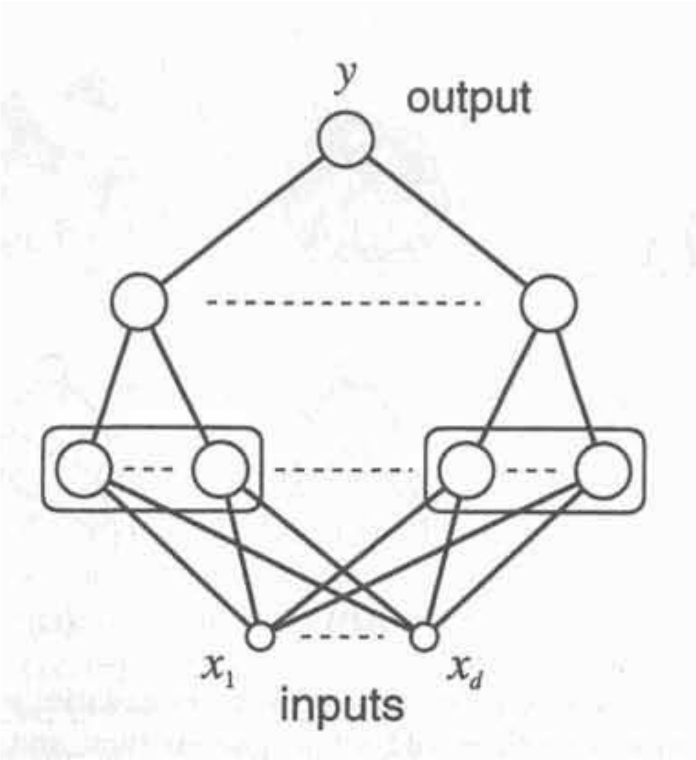
$$g(a) \equiv \tanh(a) \equiv \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$



tanh activation function

Three-layer nets

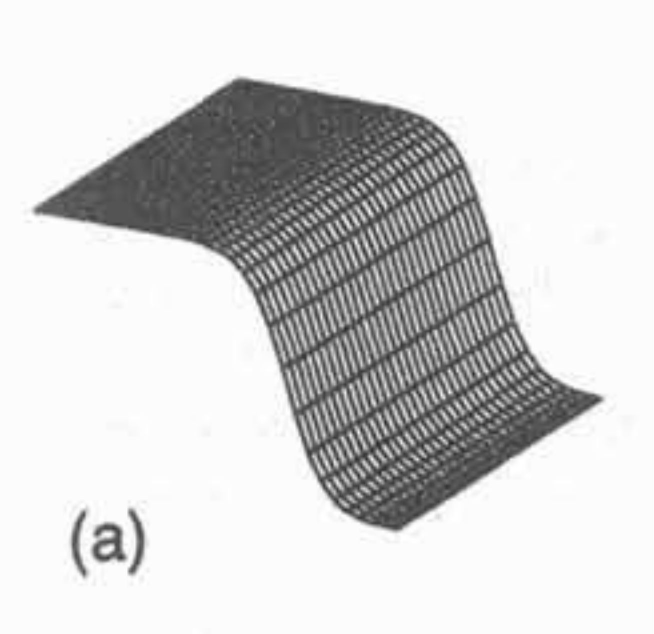
- *Result*
  - Three-layer of weights and sigmoidal activation functions can approximate, to arbitrary accuracy, any smooth mapping (Lepedes and Farber, 1988)



Topology of NN

Three-layer nets

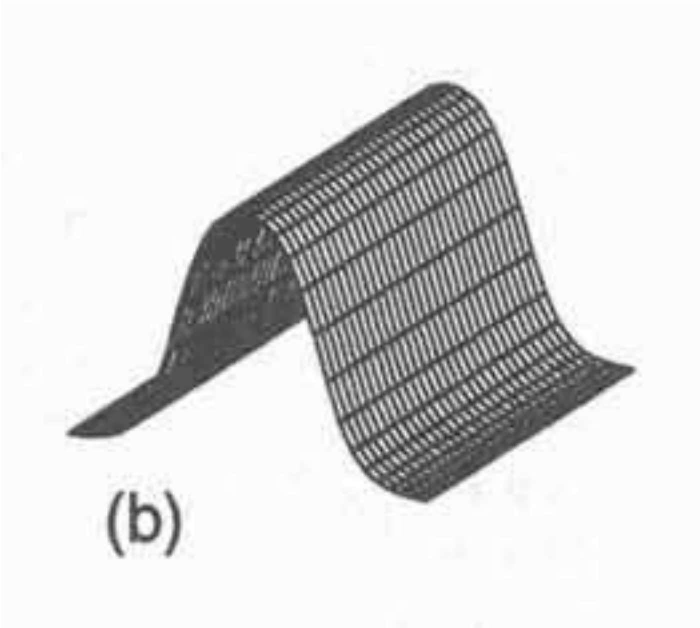
- *Input space*
  - two dimensions
- *First hidden layer*



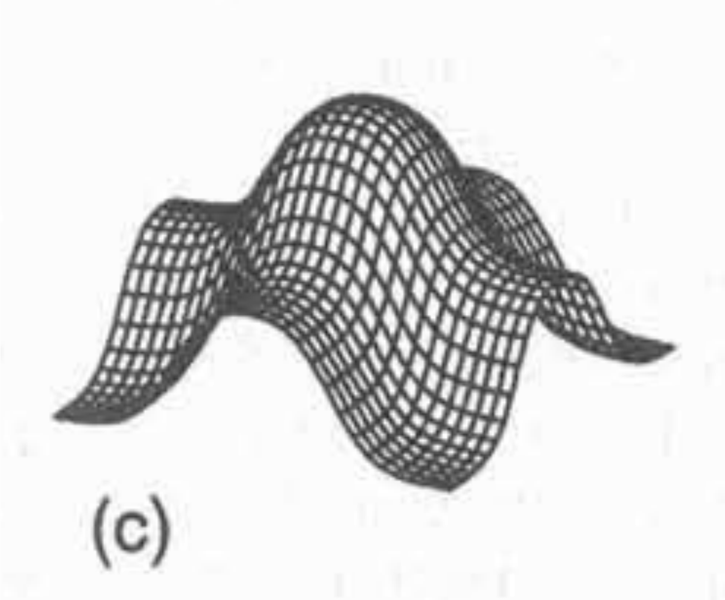
$$z = \boldsymbol{g}(\mathbf{w}^T\mathbf{x} + w_0)$$

Three-layer nets

- *First hidden layer*
  - Orientation of the sigmoid is determined by the diretion of **w** and location by $-w_0$
  - Linear cobinations of functions



Two functions

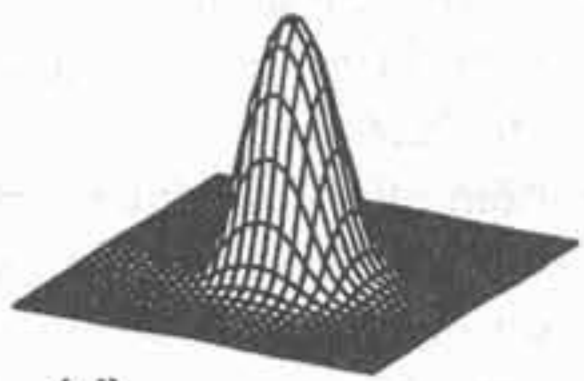


d functions

Three-layer nets

- *third hidden layer*
  - sigmoid function isolate the central bump

Bump function



(d)

- Intiuitive idea
  - *Any reasonsble function can be approximated to arbitrary accuracy by a linear superposition of sufficientrly large number of localized «bump» functions*

Two-layer nets

- *Result*
  - Two-layer nets can approximate arbitrarily well any functional (one-one or many-one) continuous mapping from one finite-dimensional space to another, provided a number M of hidden units is sufficiently large (universal approximation)

Two-layer nets

- *Input*
  - x_1 and x_2
- *Output*
  - $y(x_1, x_2)$
- Approximation by Fourier decomposition

$$y(x_1, x_2) \approx \sum_s A_s(x_1)\cos(s x_2)$$

Two-layer nets

- Fourier decomposition

$$y(x_1, x_2) \approx \sum_s \sum_l A_{sl} \cos(l x_1) \cos(s x_2)$$

- Trigonometric identity

$$\cos\alpha \cdot \cos\beta = \frac{1}{2}\cos(\alpha+\beta) + \frac{1}{2}\cos(\alpha-\beta)$$

- Linear combination

$$y(x_1, x_2) \approx \sum_s \sum_l \cos(z_{sl}) \cos(z'_{sl})$$

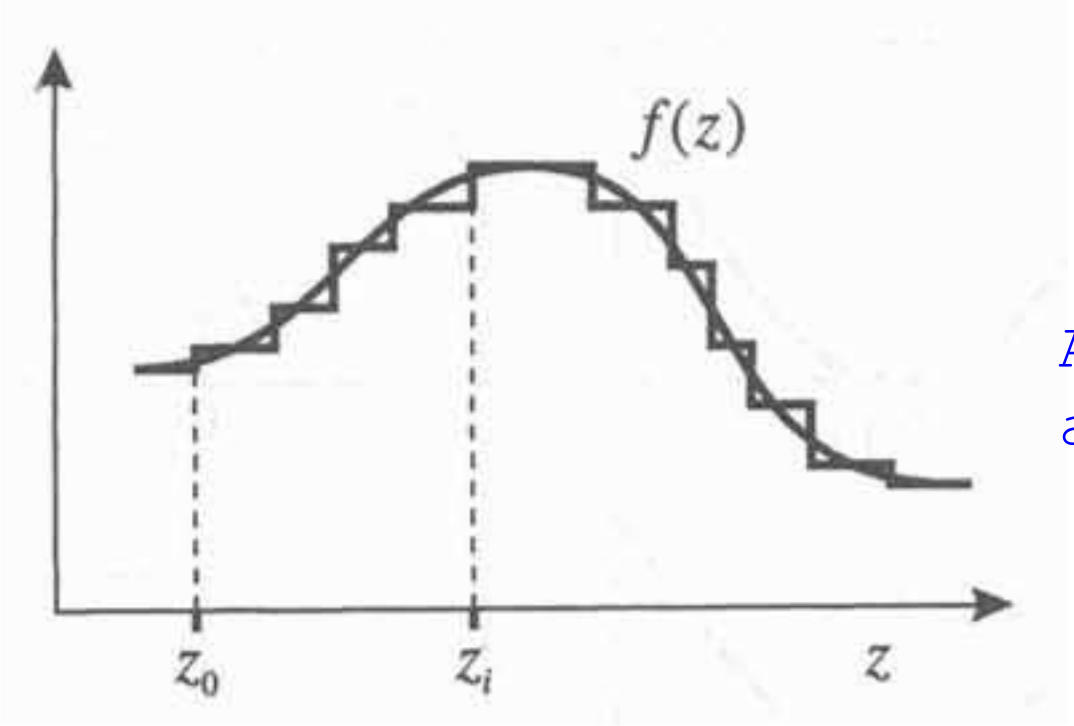
$$z_{sl} = l x_1 + s x_2 \quad z'_{sl} = l x_1 - s x_2$$

Two-layer nets

- cos(z) approximation

$$f(z) \approx f_0 + \sum_{i=0}^{N} \{f_{i+1} - f_i\} H(z - z_i)$$

Heaviside step function



Approximation of a function

Two-layer nets

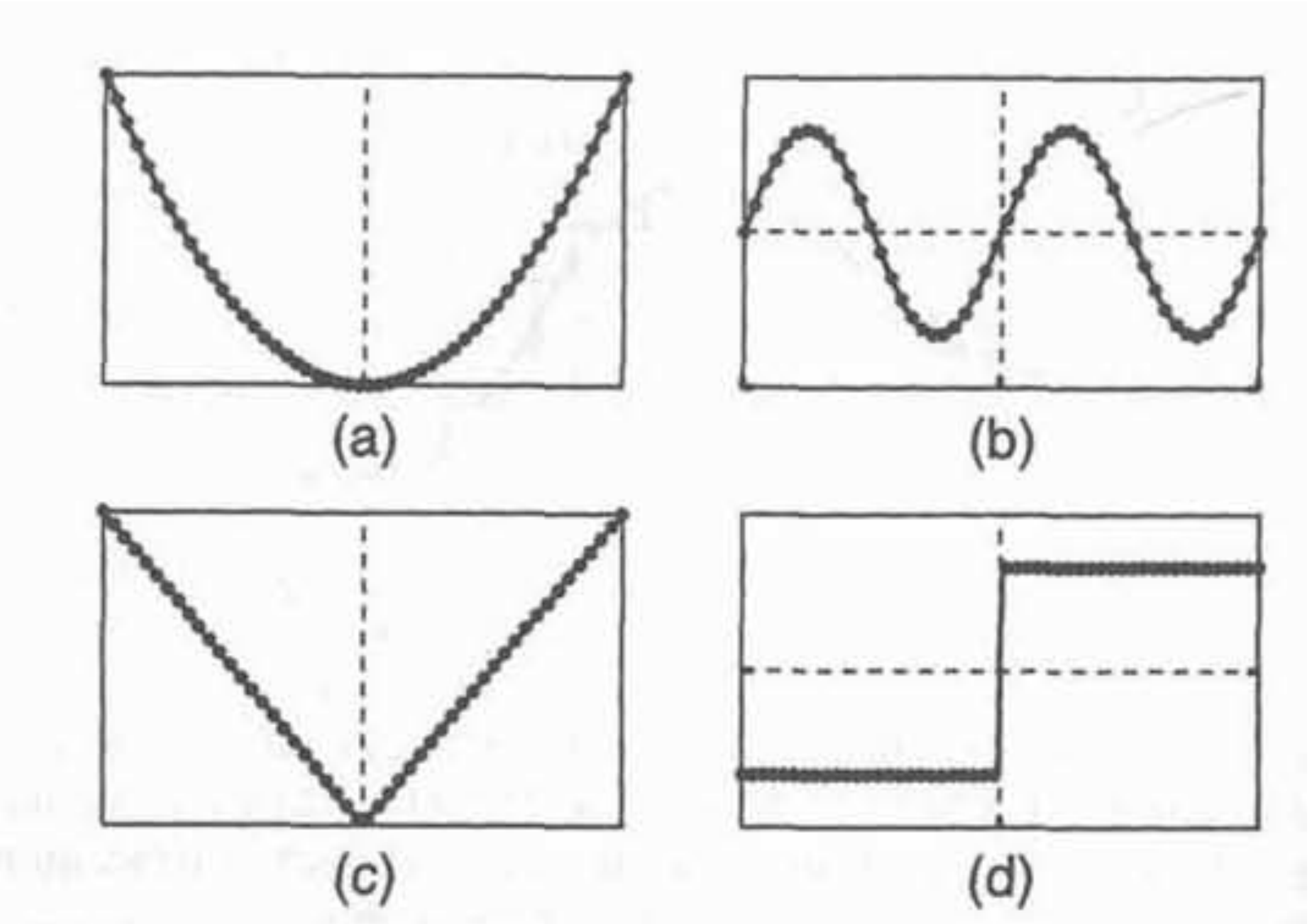
- cos(z) approximation

$$f(z) \approx f_0 + \sum_{i=0}^{N} \{f_{i+1} - f_i\} H(z - z_i)$$

Heaviside step function

- Result
  - function $y(x_1, x_2)$ can be expressed as a linear combination of step functions whose arguments are linear combinations of x_1 and x_2
  - function $y(x_1, x_2)$ can be approximated by a two-layer NN with threshold hidden units (can be approximated by sigmoidal functions)

Approximation example

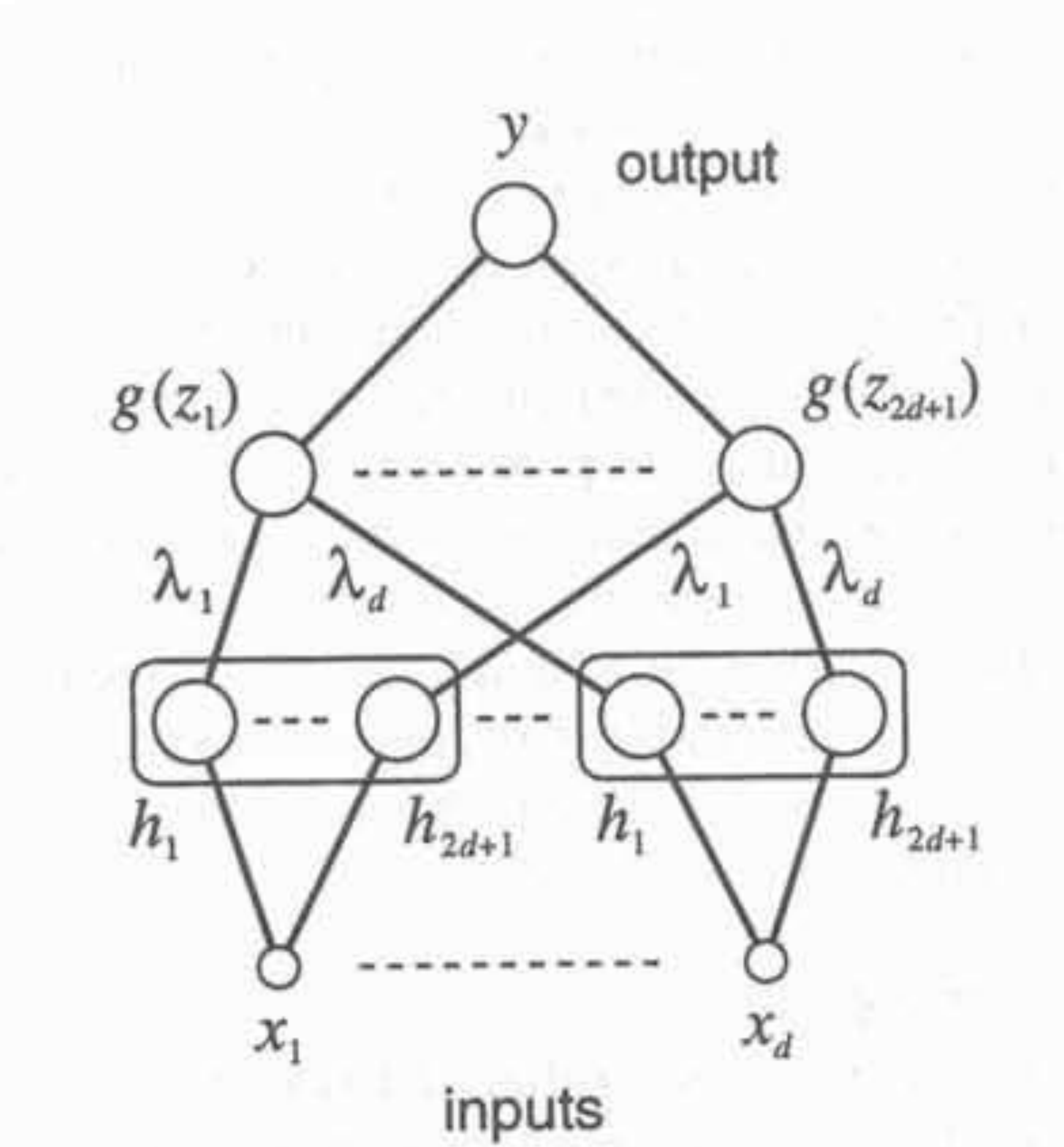


Examples of functions approximations

Kolmogorov's theorem

- Origins
  - End of nineteenth century mathematician Hilbert compiled a list of 23 unsolved problems as a challenge for twentieth century researchers
  - Hilbert's thirteenth problem
    - Concerns the issue of whether functions of several variables can be represented in terms of superpositions of functions of two variables
  - Kolmogorov (1957)
    - Every continuous function of several variables (for a closed and bounded input domain) can be represented as the superposition of a small number of functions of one variable

Kolmogorov's theorem



$$y = \sum_{j=1}^{2d+1} g(z_j)$$

$$z_j = \sum_{i=1}^{d} \lambda_i h_j(x_i)$$