Neural Networks
MLP ANNs

Advanced Institute for Artificial Intelligence

André C P L F de Carvalho
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo, Brazil

1

Topics

- Multi-Layer Perceptron
- Training
- Backpropagation network
- Input space partition
- Classification

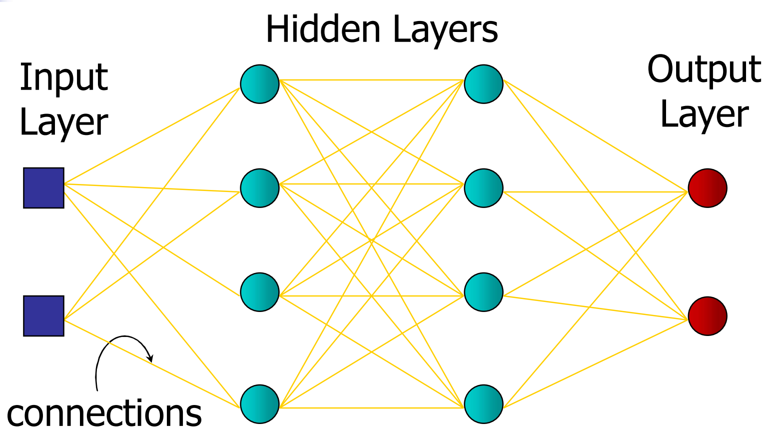
 2

2

A bit of history

- (1982) Hopfield showed that ANNs could be seen as dynamic systems
- (1986) Hinton, Rumelhart e Williams, proposed an algorithm to train multi-layer perceptron (MLP) networks
  - *Parallel Distributed Processing*
  - Bryson e Ho (1959), Werbos (1974), Parker (1985) and Le Cun (1985)

Multi-Layer Perceptron (MLP)



Multi-Layer Perceptron (MLP)

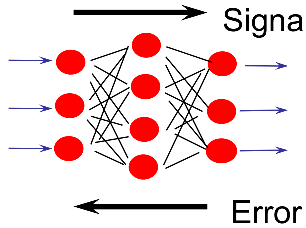
- Most used ANN model
  - One or more hidden layers
- Increased functionality
  - One hidden layer: any Boolean or continue function
  - Two hidden layers: any function
- Trained by the Backpropagation algorithm

Backpropagation

- Train by reducing errors made by the MLP network
  - Supervised
  - Error correction
    - Output layer
    - Hidden layers
      - Proportional to the error made by the nodes in the next layer

Backpropagation

- Training follows two directions
  - Forward
  - Backward

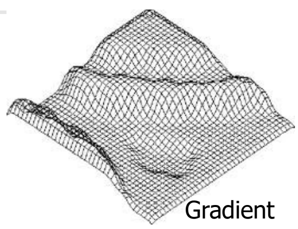


7

Backpropagation

- Training
  - Supervised
  - Adjust two weights: $\Delta w_{ij} = \eta x_i \delta_j$

$$\delta_j = \begin{cases} f'(net) error_j & \text{if } j \text{ is the output layer} \\ f'(net) \sum w_{jk} \delta_k & \text{if } j \text{ is a hidden layer} \end{cases}$$



Gradient function

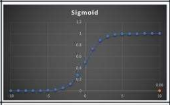
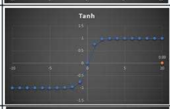
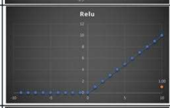
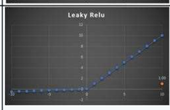
$$error_j = \frac{1}{2} \sum_{q=1}^{c} (y_q - f(net_q)) \qquad net = \sum_{i=0}^{m} x_i w_i$$

If f (net) is a sigmoidal function, $f'(net) = f(net)(1 - f(net))$

Training is not guaranteed to converge

8

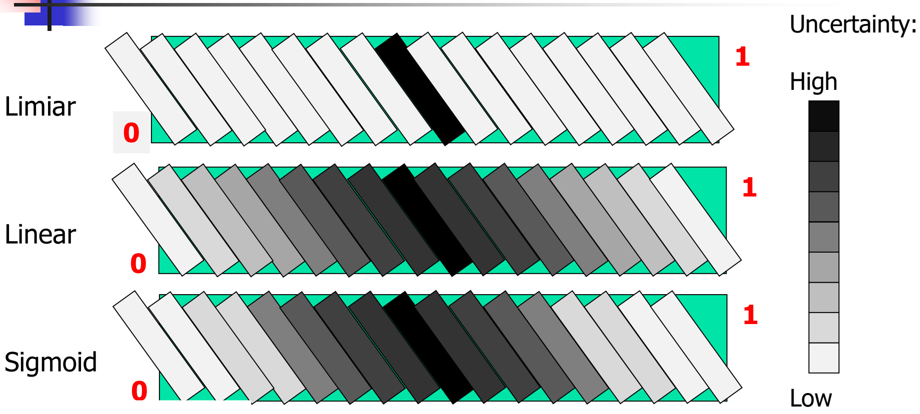
Activation functions

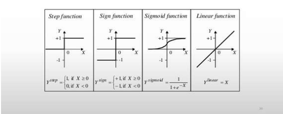
Name	Plot	Equation	Derivative
Sigmoid		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$
Rectified Linear Unit (relu)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Leaky Rectified Linear Unit (Leaky relu)		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

 9

9

Activation functions and borders



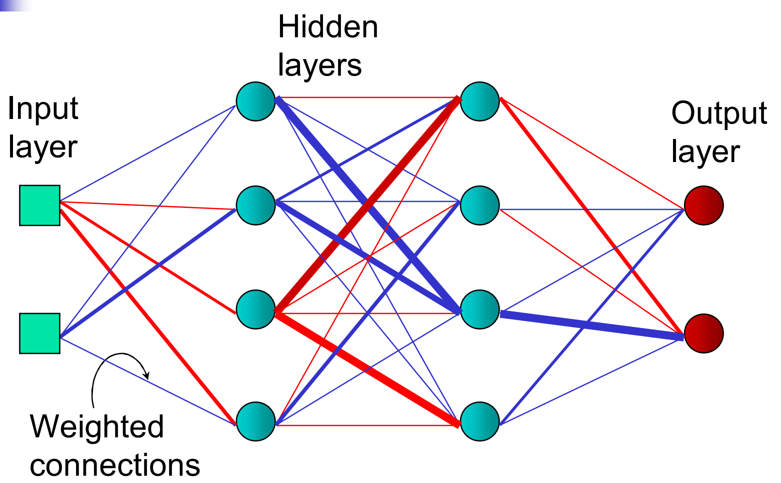


10

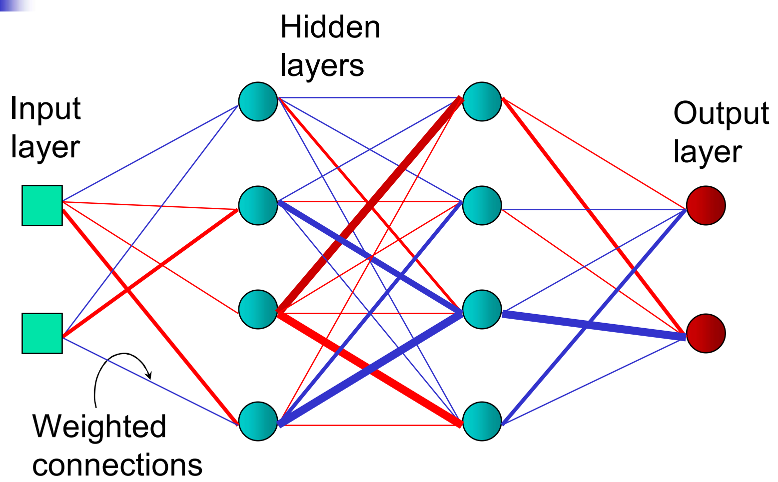
Training

start all the weights with random values
repeat
for *each training pattern (X, d)* ***do***
for *each layer k from 1 to n* ***do***
for *each node j from 1 to* m_k ***do***
calculate the output y_{jk}
if layer = n
then calculate error
if error $> \varepsilon$
then for *each layer k from n to 1* ***do***
for *each node j from 1 to* m_k ***do***
update weight values
until *error* $\leq \varepsilon$

Training updating weights



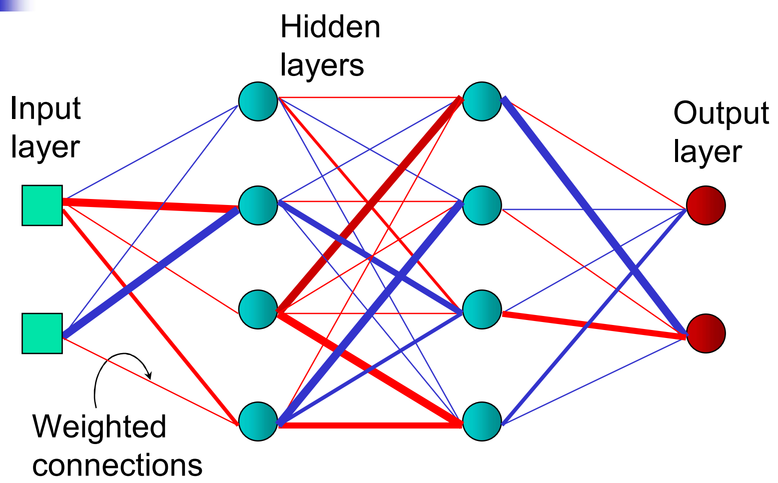
Training updating weights



André Ponce de Leon de Carvalho 13

13

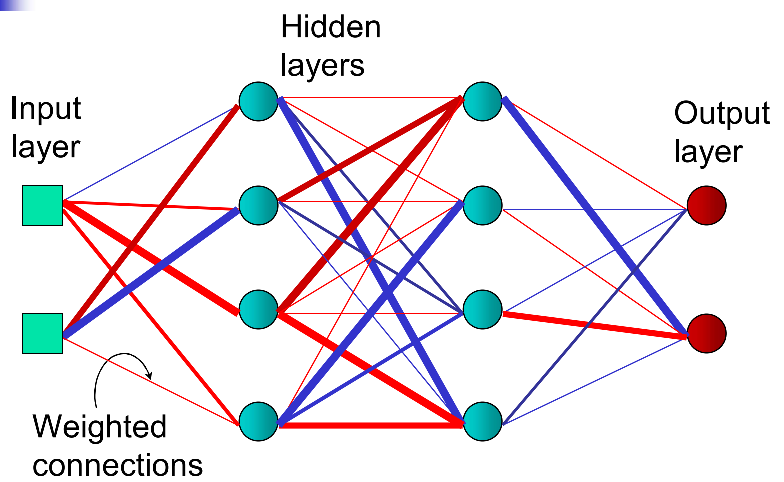
Training updating weights



André Ponce de Leon de Carvalho 14

14

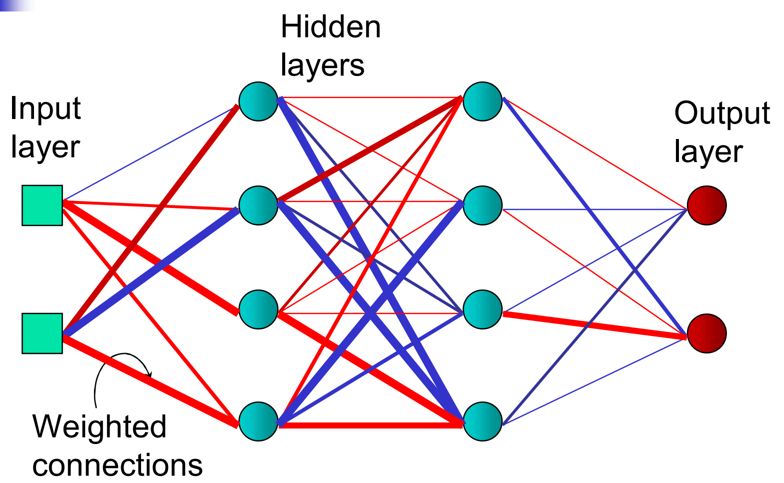
Training updating weights



André Ponce de Leon de Carvalho 15

15

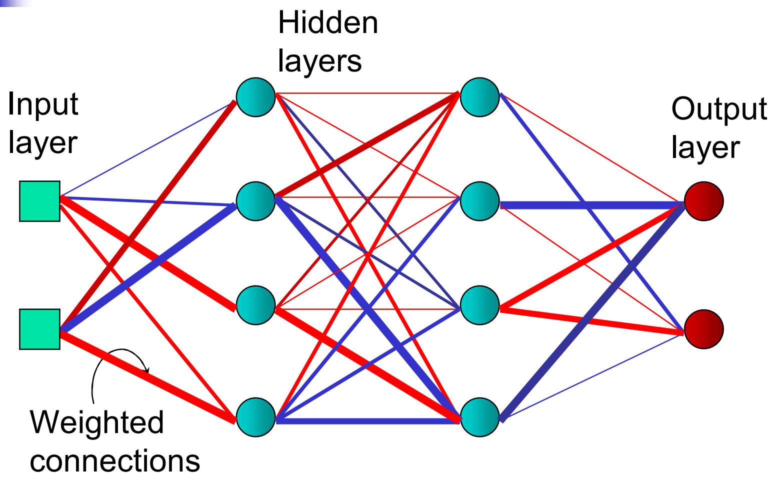
Training updating weights



André Ponce de Leon de Carvalho 16

16

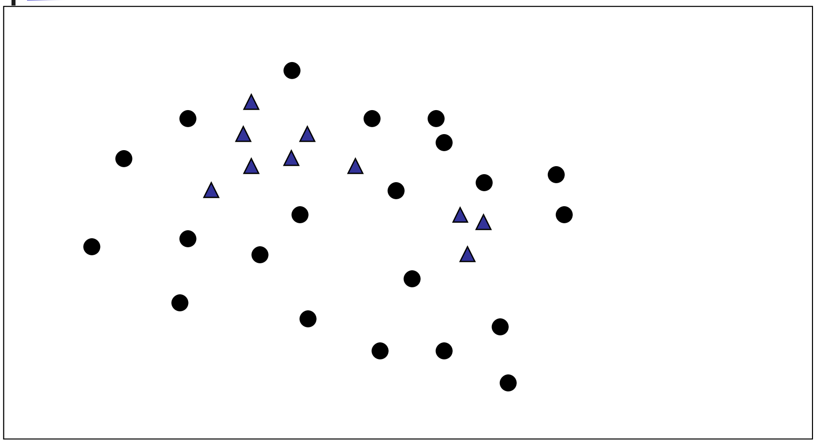
Training updating weights



André Ponce de Leon de Carvalho 17

17

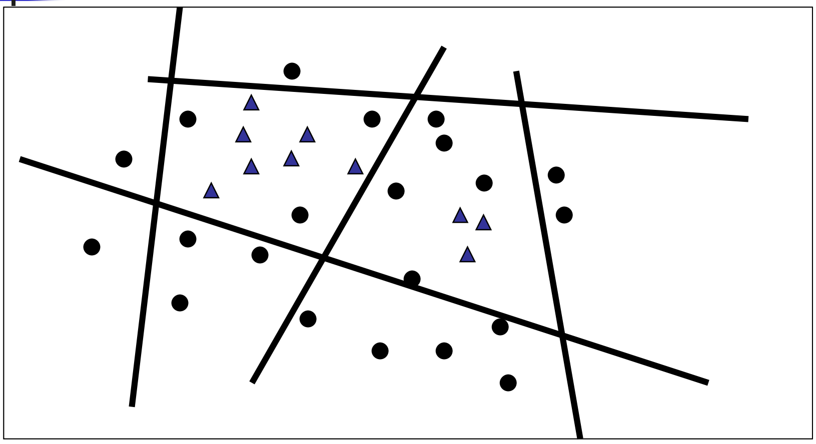
Training changing borders



André Ponce de Leon de Carvalho 18

18

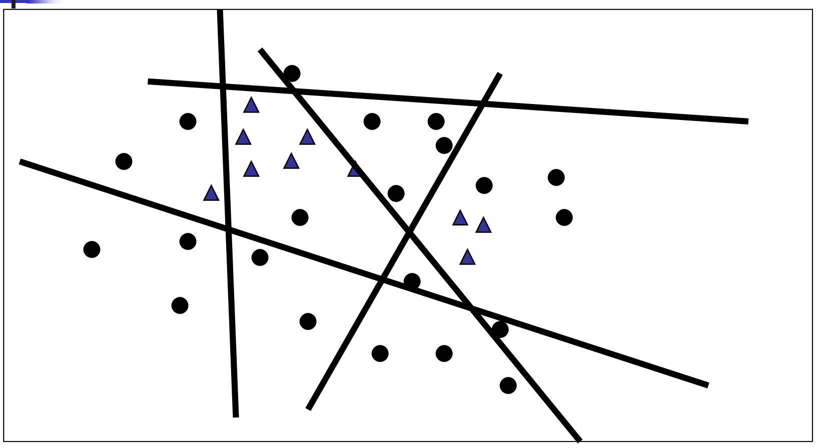
Training changing borders



André Ponce de Leon de Carvalho 19

19

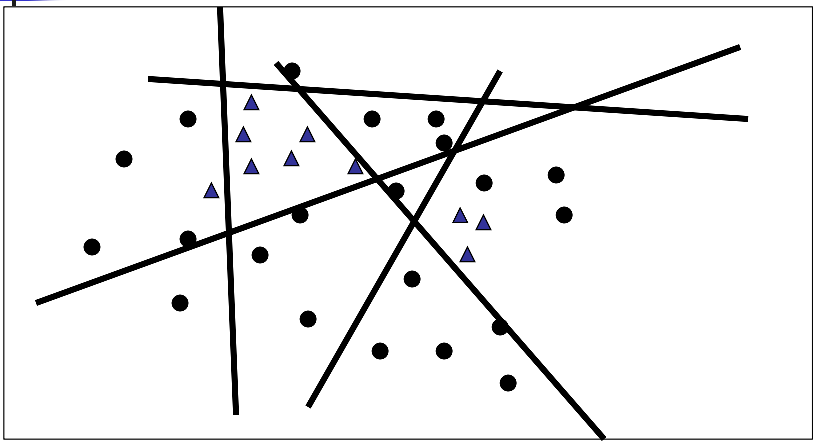
Training changing borders



André Ponce de Leon de Carvalho 20

20

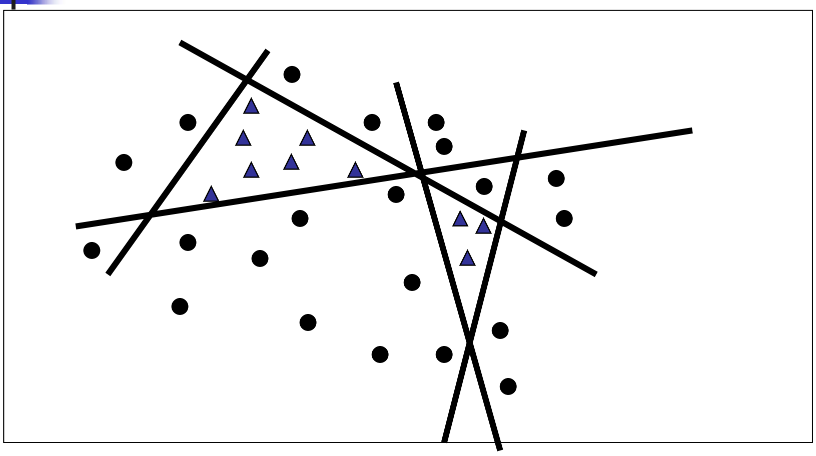
Training changing borders



André Ponce de Leon de Carvalho 21

21

Training changing borders



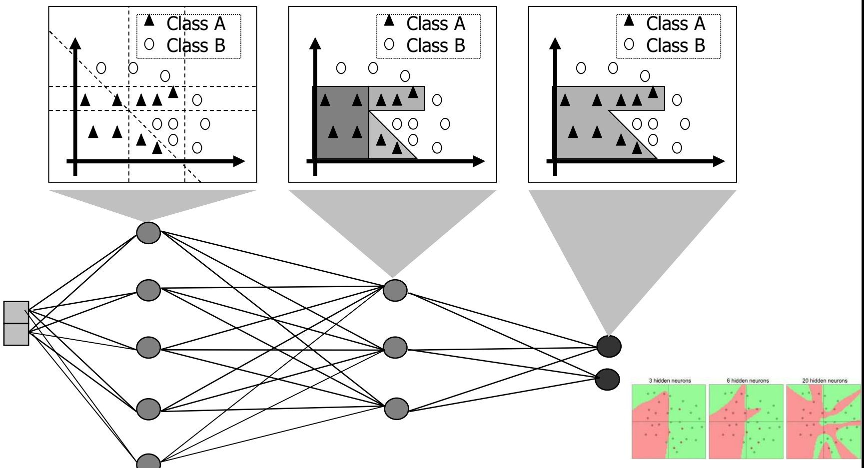
André Ponce de Leon de Carvalho 22

22

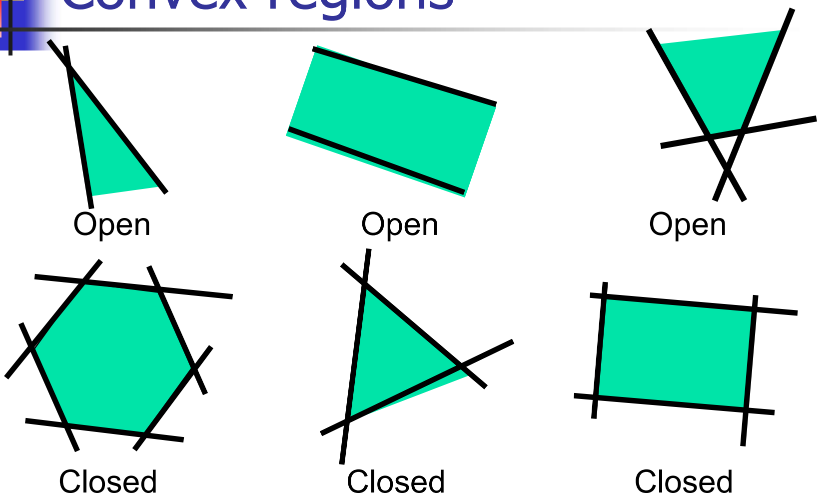
Test

for *each test pattern* ***do***
 for *each layer k from 1 to n* ***do***
  for *each node j from 1 to* m_k ***do***
   calculate the output y_{jk}
   compare Y and D
   classify the test pattern X as belonging to the class whose desired output is closer to the produced output

MLPs as classifiers



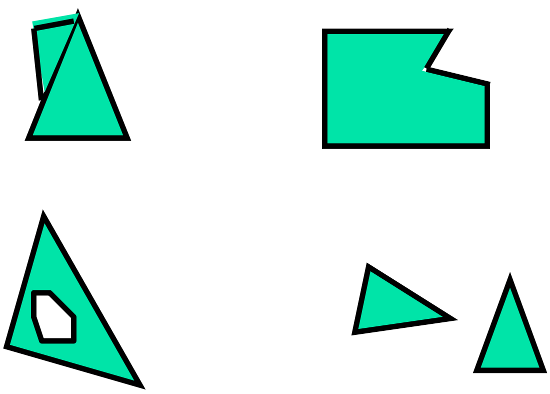
Convex regions



 25

25

Combinations of convex regions

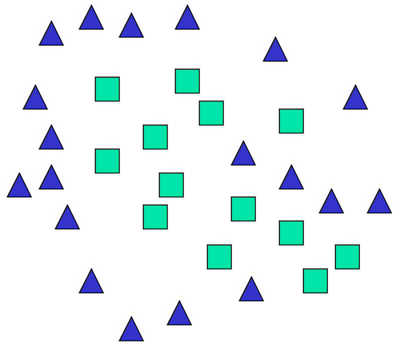


 26

26

Combinations of convex regions

- Find decision boundaries that separate the data below:

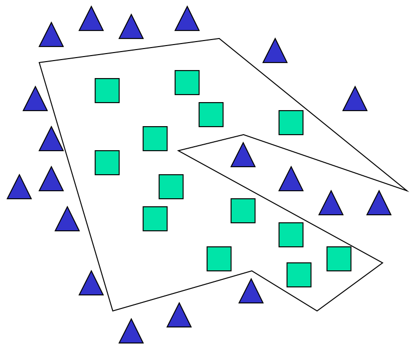


 27

27

Combinations of convex regions

- Find decision boundaries that separate the data below:

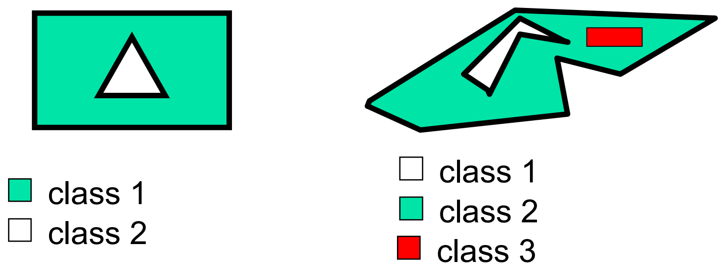


 28

28

Quiz 1

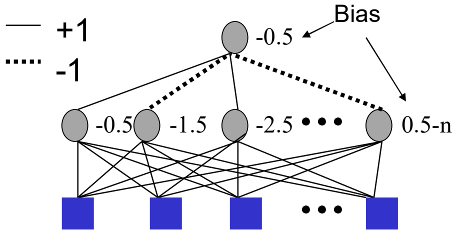
- How many layers and at least how many neurons in each layer have the networks that divide the input space of the shapes below:



Exercise

- Given the ANN below whose input is a n-bit binary vector and output is a binary value:
  - Indicate the function implemented:
  - Explain what each neuron does

The activation function uses step functin with threshold

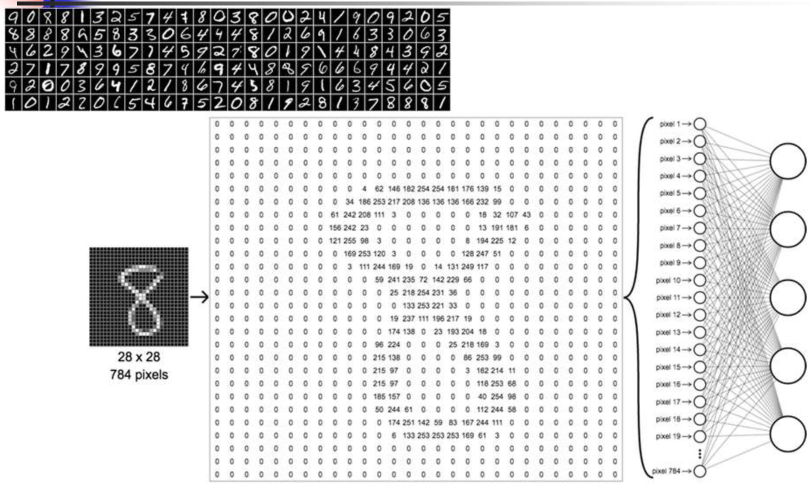


Exercise

- Parity
  - One of the limitations of the Perceptron raised by Minsky and Papert
- Difficult problem
  - More similar patterns require different responses
  - Uses n intermediate units to detect parity in binary vectors with n elements

Example

Example



33

Other training algorithms

- Backpropagation momentum
- Resilient propagation (Rprop)
- Quickprop
- Newton
- Levenberg Marquardt
- Super Self-Adjusting Backpropagation (superSAB)
- Conjugate gradient algorithms
- ...

34

Backpropagation momentum

- Adds a momentum term to the weight update equation
  - If the last and current weight update go in the same direction, current update is larger
    - Direction: increase or decrease weight
  - Specifies the amount of the old weight change to be added to the current change
  - Increase chance to escape from local minima

35

Quiz 2

- What are limitations of MLP?

  A) Only works with up to 2 hidden layers

  B) Can learn any function using 2 hidden layers

  C) Can use any nonlinear activation function

  D) Can be used only for classification tasks

36

Next:
Other Neural Networks

 37

37