Neural Networks
Simple ANNs

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

Topics
- Perceptron
  - Basics
  - Training
  - Test
  - Example
- Adaline
  - Basics
  - Differences

© André de Carvalho - ICMC/USP
**Perceptron**

- First ANN implementation
  - Rosenblat, 1958
  - McCulloch-Pitts neuron model
- Training
  - Supervised
  - Error correction
    - $w_i(t) = w_i(t-1) + \Delta w_i$
    - $\Delta w_i = \eta x_i \delta$
    - $\Delta w_i = \eta x_i (y - f(\mu))$
- Convergence theorem

**Perceptron architecture**

- Diagram showing the architecture of a perceptron with inputs $x_1, x_2, \ldots, x_m$, weights $w_1, w_2, \ldots, w_n$, and output $f$. The bias is represented as $-\theta$. The formula for weight updating is $w_i(t) = w_i(t-1) + \Delta w_i$.
Perceptron implementation

- First implementation:
  - Mark I Perceptron
  - Cornell Aeronautical Laboratory, USA

Starting the implementation
Preparing the Perceptron

Concluding Perceptron
Perceptron working

Perceptron

- Answer / network output
  - Applies threshold activation function on total input sum received by a neuron

\[ u = \sum_{i=1}^{m} x_i w_i \]

\[ f(u) = \begin{cases} 
+1 & \text{if } u \geq \theta \\
-1 & \text{if } u < \theta 
\end{cases} \]

\[ net = \sum_{i=0}^{m} x_i w_i \]

- \[ f(u) = \text{sinal}(u) \]
- \[ f(u-\theta) = f(u) \]
Training algorithm

initialise all weights with \( w_j = 0 \)

repeat
  for each training example \((X, d)\) do
    calculate the network output \( y \)
    if \( d \neq y \) then
      update weight value
  until an acceptable error is achieved
Training

Error surface

Training moving borders
Training moving borders
Training moving borders

© André de Carvalho - ICMC/USP
for each test example X do
  present X to the network input
  calculate network output y
  if (y = -1 )
    then X ∈ class A
  else X ∈ class B

Example

- Given a Perceptron network with:
  - Three input terminals, using the following initial weights $w_0 = 0.4$, $w_1 = -0.6$ e $w_2 = 0.6$, and threshold $\theta = 0.5$:
    - Teach the networks with the training dataset (001, -1) and (110, +1)
      - Using as learning rate $\eta = 0.4$
    - Define the class for the samples: 111, 000, 100 e 011
Example

Example: item a)

a) Train the network

a.1) For the input pattern 001

Step 1: define the network output

\[ u = 0(0.4) + 0(-0.6) + 1(0.6) - 1(0.5) = 0.1 \]

\[ y = u = +1 \text{ (since } 0.1 \geq 0) \]

Step 2: adjust weights (\( d \neq y \))

\[ w_0 = 0.4 + 0.4(0)(-1 - (+1)) = 0.4 \]

\[ w_1 = -0.6 + 0.4(0)(-1 - (+1)) = -0.6 \]

\[ w_2 = 0.6 + 0.4(1)(-1 - (+1)) = -0.2 \]

\[ w_3 = 0.5 + 0.4(-1)(-1 - (+1)) = 1.3 \]
Example: item a)

a) Train the network

a.2) For the input pattern 110 \( (d = +1) \)

Step 1: define the network output

\[
    u = 1(0.4) + 1(-0.6) + 0(-0.2) -1(1.3) = -1.5
\]

\[
y = u = -1 \text{ (since } -1.5 < 0)\]

Step 2: adjust weights \((d \neq y)\)

\[
w_0 = 0.4 + 0.4(1)(1 - (-1)) = 1.2
\]

\[
w_1 = -0.6 + 0.4(1)(1 - (-1)) = 0.2
\]

\[
w_2 = -0.2 + 0.4(0)(1 - (-1)) = -0.2
\]

\[
w_3 = 1.3 + 0.4(-1)(1 - (-1)) = 0.5
\]

Example: item a)

a) Train the network

a.3) For the input pattern 001 \( (d = -1) \)

Step 1: define the network output

\[
    u = 0(1.2) + 0(0.2) + 1(-0.2) -1(0.5) = -0.7
\]

\[
y = u = -1 \text{ (since } -0.7 < 0)\]

Step 2: adjust weights \((d = y)\)

Since \(d = y\), there is no need to adjust the weights
Example: item a)

a) Train the network

a.4) For the input pattern 110 \( (d = +1) \)

Step 1: define the network output

\[
u = 1(1.2) + 1(0.2) + 0(-0.2) -1(0.5) = +0.7
\]

\[
y = u = +1 \text{ (since } 0.7 > 0)\]

Step 2: adjust weights \( (d = y) \)

Since \( d = y \), there is no need to adjust the weights

Example: item b)

b) Test the network

b.1) For the input pattern 111

\[
u = 1(1.2) + 1(0.2) + 1(-0.2) -1(0.5) = 0.7
\]

\[
y = u = 1 \text{ (since } 0.7 \geq 0) \Rightarrow \text{ class 1}
\]

b.2) For the input pattern 000

\[
u = 0(1.2) + 0(0.2) + 0(-0.2) -1(0.5) = -0.5
\]

\[
y = u = -1 \text{ (since } -0.5 < 0) \Rightarrow \text{ class 0}
\]
Example: item b)

b) Test the network
   
b.3) For the input pattern 100
   \[ u = 1(1.2) + 0(0.2) + 0(-0.2) + 1(-0.5) = 0.7 \]
   \[ y = u = 1 \text{ (since } 0.7 \geq 0) \Rightarrow \text{ class 1} \]

   b.4) For the input pattern 011
   \[ u = 0(1.2) + 1(0.2) + 1(-0.2) + 1(0.5) = -0.5 \]
   \[ y = u = -1 \text{ (since } -0.5 < 0) \Rightarrow \text{ class 0} \]

Example 2

- Distinguish male face images from female face images
Example 2

- Distinguish male face images from female face images
  - Use images directly
    - Matrix of pixels
  - Use features extracted from the image
    - Distance between components from the face (Ex. eyes)
    - Texture
      - Moustache, Hair, Beard
Consider the following patients

<table>
<thead>
<tr>
<th>Name</th>
<th>Fever</th>
<th>Dizzy</th>
<th>Spots</th>
<th>Pain</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>yes</td>
<td>yes</td>
<td>small</td>
<td>yes</td>
<td>ill</td>
</tr>
<tr>
<td>Peter</td>
<td>no</td>
<td>no</td>
<td>large</td>
<td>no</td>
<td>healthy</td>
</tr>
<tr>
<td>Mary</td>
<td>yes</td>
<td>yes</td>
<td>small</td>
<td>no</td>
<td>healthy</td>
</tr>
<tr>
<td>Joe</td>
<td>yes</td>
<td>no</td>
<td>large</td>
<td>yes</td>
<td>ill</td>
</tr>
<tr>
<td>Ann</td>
<td>yes</td>
<td>no</td>
<td>small</td>
<td>yes</td>
<td>healthy</td>
</tr>
<tr>
<td>Lyn</td>
<td>no</td>
<td>no</td>
<td>large</td>
<td>yes</td>
<td>ill</td>
</tr>
</tbody>
</table>

Example 3

Consider the following patients

<table>
<thead>
<tr>
<th>Fever</th>
<th>Dizzy</th>
<th>Spots</th>
<th>Pain</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Example 3

- Train a Perceptron network to distinguish:
  - Potentially healthy patients
  - Potentially ill patients
- Test the network for the following new cases
  - (Louis, no, no, small, yes)
  - (Laura, yes, yes, large, yes)

Adaline

- Problem with Perceptron:
  - Weight adjustment does not take into account the true distance between
    - Produced output and desired output
- Adaline network
  - Proposed by Widrow and Hoff in 1960
  - Also based on McCulloch-Pitts nodes
Adaline

- **Training**
  - Supervised
  - Error correction (LMS, delta rule)
    - \( \Delta w_{ij} = \eta x_i (d_j - y_j) \) (\( d \neq y \))
    - \( \Delta w_{ij} = 0 \) (\( d = y \))
  - Gradual weight adjustment
    - Takes into account the distance between the desired output (d) and the produced output (y)

Activation functions

- **Step function**
  - \( y_{\text{step}} = \begin{cases} 1 & \text{if } X \geq 0 \\ 0 & \text{if } X < 0 \end{cases} \)

- **Sign function**
  - \( y_{\text{sign}} = \begin{cases} 1 & \text{if } X \geq 0 \\ -1 & \text{if } X < 0 \end{cases} \)

- **Sigmoid function**
  - \( y_{\text{sigmoid}} = \frac{1}{1 + e^{-X}} \)

- **Linear function**
  - \( y_{\text{linear}} = X \)
Quiz 1

What is the main difference between Perceptron and Adaline?
A) Learning rule
B) Architecture
C) Activation function
D) Learning paradigm

A bit of history

(1969) Minsky & Papert analysed the Perceptron network and pointed out its limitations
- Could only deal with linear separable problems
  - Not with XOR and parity
- Largely reduced the research activity in ANNs
Perceptron limitation

0, 0 → 0
0, 1 → 1
1, 0 → 1
1, 1 → 0

Perceptron

- Linearly separable patterns
Perceptron

- Linearly separable patterns

Perceptron limitation

- One-layer networks can only deal with linearly separable problems
- A large number of important application problems are non-linearly separable
- Many problems have more than 2 classes
  - Multiclass problems
Quiz 2

What is the main difference between Perceptron and Adaline?
A) Learning rule
B) Architecture
C) Activation function
D) Learning paradigm

Perceptron limitation

Solution: use more than one layer
Problem: how to train the first layers
Next:
MLP Neural Networks

© André de Carvalho - ICMC/USP