Neural Networks
MLP ANNs

Topics

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

Signal (forward)

Error (backward)

Training
- Supervised
- Adjust two weights: \( \Delta W_{ij} = \eta x_i \delta_j \)

\[
\delta_j = \begin{cases} 
    f'(net) \text{error} & \text{if } j \text{ is the output layer} \\
    f'(net) \sum w_k \delta_k & \text{if } j \text{ is a hidden layer}
\end{cases}
\]

\[
\text{error}_j = \frac{1}{2} \sum_{q=1}^{L} (y_q - f(\text{net}_j))
\]

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

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

Training is not guaranteed to converge
Activation functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Plot</th>
<th>Equation</th>
<th>Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid</td>
<td>![Sigmoid Plot]</td>
<td>$f(x) = \frac{1}{1 + e^{-x}}$</td>
<td>$f'(x) = f(x)(1 - f(x))$</td>
</tr>
<tr>
<td>Tanh</td>
<td>![Tanh Plot]</td>
<td>$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$</td>
<td>$f'(x) = 1 - f(x)^2$</td>
</tr>
<tr>
<td>Rectified Linear Unit (relu)</td>
<td>![Rectified Linear Unit Plot]</td>
<td>$f(x) = \begin{cases} 0 &amp; \text{for } x &lt; 0 \ x &amp; \text{for } x &gt; 0 \end{cases}$</td>
<td>$f'(x) = \begin{cases} 0 &amp; \text{for } x &lt; 0 \ 1 &amp; \text{for } x &gt; 0 \end{cases}$</td>
</tr>
<tr>
<td>Leaky Rectified Linear Unit (Leaky relu)</td>
<td>![Leaky Rectified Linear Unit Plot]</td>
<td>$f(x) = \begin{cases} 0.01 \text{ for } x &lt; 0 \ x &amp; \text{for } x &gt; 0 \end{cases}$</td>
<td>$f'(x) = \begin{cases} 0.01 &amp; \text{for } x &lt; 0 \ 1 &amp; \text{for } x &gt; 0 \end{cases}$</td>
</tr>
</tbody>
</table>
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 > e
        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 ≤ e

Training updating weights

Input layer

Hidden layers

Output layer

Weighted connections
Training updating weights

Input layer

Hidden layers

Weighted connections

Output layer

André Ponce de Leon de Carvalho

15

16
Training updating weights

Input layer

Hidden layers

Weighted connections

Output layer

Training changing borders
Training changing borders
Training changing borders

André Ponce de Leon de Carvalho

21

22
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

Open  Open  Open
Closed Closed Closed

Combinations of convex regions
Combinations of convex regions

- Find decision boundaries that separate the data below:
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:

![shapes]

- class 1
- class 2
- class 1
- class 2
- class 3

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

![ANN diagram]

The activation function uses step function with threshold

© André de Carvalho - ICMC/USP
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
Other training algorithms

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

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
Next: Other Neural Networks