Connectionist and Statistical Language Processing

Lecture 3: Multi-layer networks

Matthew W Crocker

Computerlinguistik
Universität des Saarlandes

So far …

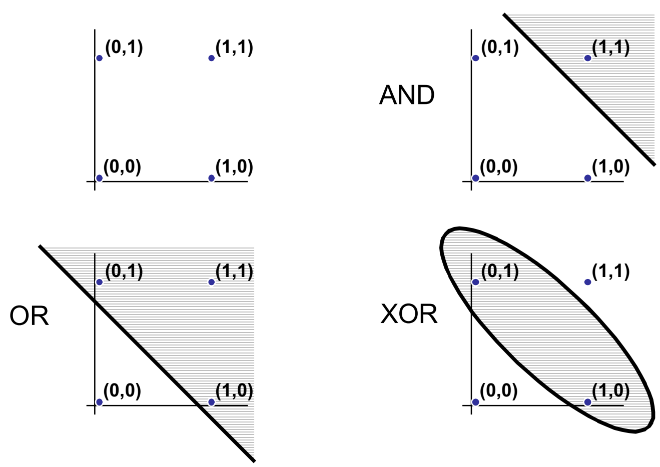
- Structure of nodes:
  - Netinput: weight sum of input activations
  - Output: activation functions, f(netinput)
- Learning Rules:
  - The Delta rule
  - The perceptron convergence rule
  - Gradient descent
- Training:
  - Global error (RMS)
- Properties of single layer networks:
  - A solution will be found, if it exists
  - But, many problems aren't solveable with single layer networks
    - E.g. XOR

Overview

- Characterising the limits of single layer networks
  - Linearly separable problems only
- Multi-layer networks:
  - Solution to XOR
  - Learning “inferences”: the family tree example
  - Properties of multi-layer networks
- Training networks with hidden layers:
  - The back-propagation algorithm

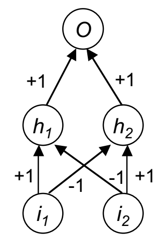
2-D Representation of Boolean Functions

- We can visual the relationship between inputs (plotted in 2-D space) and the desired output (represented as a line dividing the space):



Solving XOR with hidden units

- Consider the following network:
  - 3-layer, feedforward
  - 2 units in a "hidden"-layer
  - Hidden and output units are threshold units: $\theta = 1$

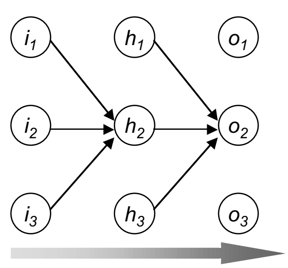


- Representations at hidden layer:

Input	Hidden		Target
	h_1	h_2	
0 0	0	0	0
1 0	1	0	1
0 1	0	1	1
1 1	0	0	0

- Problem: current learning rules cannot be used for hidden units:
  - Why? We don't know what the "error" is at these nodes
  - "Delta" requires that we know the desired activation $\Delta w = 2\varepsilon\delta F * a_{in}$

Backpropagation of Error



(a) Forward propagation of activity:

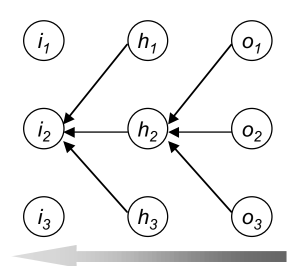
$$\text{netinput}_{out} = \sum w \cdot a_{hidden}$$

$$a_{out} = F(\text{netinput}_{out})$$

(b) Backward propagation of error:

$$\text{netinput}_{hidden} = \sum w \cdot \delta_{out}$$

$$\delta_{hidden} = F(\text{netinput}_{hidden})$$



Learning in Multi-layer Networks

- The generalised Delta rule:

$$\Delta w_{ij} = \eta \delta_{ip} a_j$$

For output nodes:

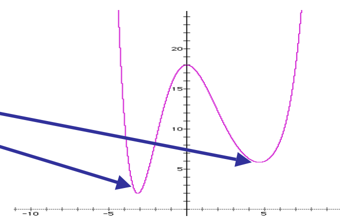
$$\delta_{ip} = f'(net_{ip})(t_{ip} - a_{ip})$$

For hidden nodes:

$$\delta_{ip} = f'(net_{ip}) \sum_k \delta_{kp} w_{ki}$$

where, $f'(net_{ip}) = a_{ip}(1 - a_{ip})$

- Multi-layer networks can, in principle, learn any mapping function:
  - Not constrained to problems which are linearly separable
- While there exists a solution for any mapping problem, backpropagation is not guaranteed to find it
  - Unlike the perceptron convergence rule
- Why? Local minima:
  - Backprop can get trapped here
  - Global minimum (solution) is here



Example of Backpropagation

- Consider the following network, containing hidden nodes
- Calculate the weight changes for both layers of the network, assuming targets of: 1 1

The generalised Delta rule:

$$\Delta w_{ij} = \eta \delta_{ip} a_j$$

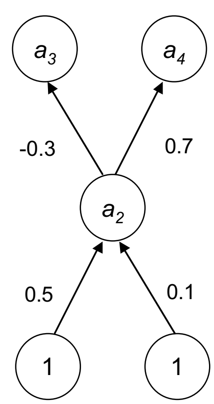
For output nodes:

$$\delta_{ip} = f'(net_{ip})(t_{ip} - a_{ip})$$

For hidden nodes:

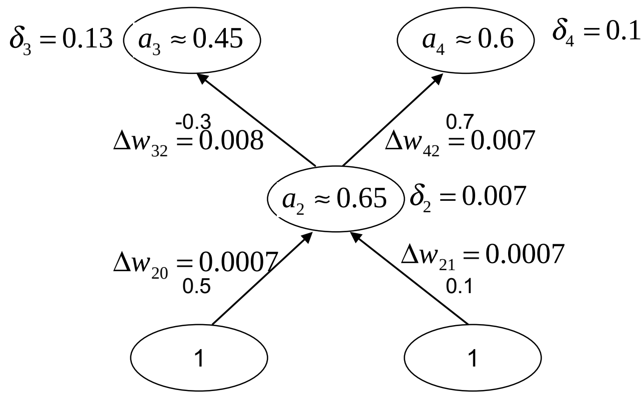
$$\delta_{ip} = f'(net_{ip}) \sum_k \delta_{kp} w_{ki}$$

where, $f'(net_{ip}) = a_{ip}(1 - a_{ip})$



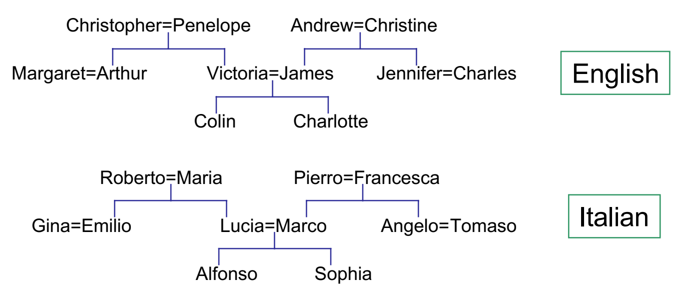
For calculations see (Plunkett & Elman, Ch. 1)

Calculations (Plunkett & Elman, Ch. 1)



The Family Tree Problem

- Family trees encode more complex relationships:



- 24 people, 12 relationships
  - Mother, daughter, sister, wife, aunt, niece (+ masculine versions)
- Training: trained on 100 of 104 possible relationships
- Learned the other 4: e.g. Victoria’s son is Colin

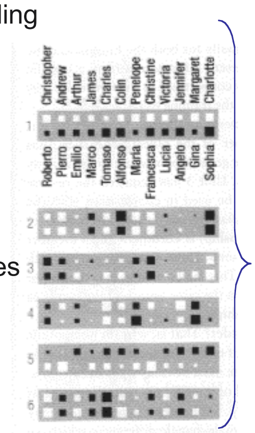
What does the Network Learn

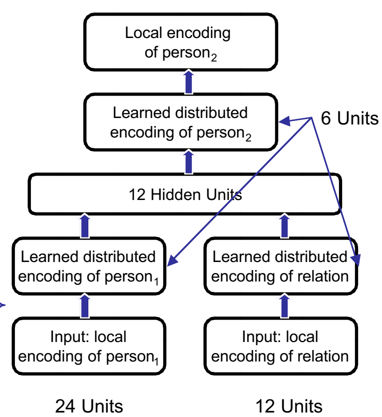
- E.g. Victoria's son is Colin:
  - Input: Victoria & Son
  - Output: Colin
- In a single-layer network:
  - Victoria would activate all the people victoria was (known to be) related to
  - Son would activate all people who are (known to be) sons
    - Colin would be partially activated, because he is James' son
  - But Colin would not have very high activation
    - James and Arthur are both sons, and related to Victoria
- A solution to this problem requires deduction:
  - Transitive inference:
    - Victoria's husband is James AND James' son is Colin
    - THEREFORE Victoria's son is Colin
- Thus the structure of the tree is learned from exemplars

Learning family tree relationships

- The network architecture, using hidden units:
- The learned encoding of people:

1. Active for English
2. Active for older generation
3. Active for the leaves
4. Encodes right side
5. Active for Italian
6. Encodes left side





Some comments

- Single layer networks (perceptrons)
  - Can only solve problems which are linearly separable
  - But a solution is guaranteed by the perceptron convergence rule

- Multi-layer networks (with hidden units)
  - Can in principle solve any input-output mapping function
  - Backpropagation performs a gradient descent of the error surface
  - Can get caught in a local minimum
  - Cannot guarantee to find the solution

- Finding solutions:
  - Manipulate learning rule parameters: learning rate, momentum
  - Brute force search (sampling) of the error surface to find a set of starting position in weight space
    - Computationally impractical for complex networks

Biological plausibility

- Backpropagation requires bi-directional signals
  - Forward propagation of activation
  - Backward propagation of error
  - Nodes must “know” the strengths of all synaptic connections to compute error: non-local
- Axons are uni-directional transmitters
- Possible justification:
  - Backpropagation explains *what* is learned,
  - Not *how* it is learned
- Network architecture:
  - Successful learning crucially depends on the number of hidden units
  - There is know way to know, a priori, what that number is
- Another solution: use a network with a local learning rule
  - E.g. Hebbian learning

Material we have covered includes:

- McLeod, Plunkett & Rolls
  - Chapter 1: Basic of connectionist processing, intro to Tlearn
  - Chapter 5:
    - The perception convergence rule
    - Linear separability
    - Gradient descent
    - Multi-layer networks
    - Backpropagation

- Elman and Plunkett
  - Chapter 1: Overview of above + exercises
  - Chapter 3: Training the Tlearn simulator to leaner Boolean operations

Tutorial

- Question 5. Assume we allow error of .4 on outputs:
  - Activation > .6 to corresponds to 1
  - Activation < .4 to corresponds to 0
  - What value for RMS guarantees the network has learned to criterion?

  - Worst (most misleading) case is when error is zero on (I.e. has learned) all patterns and outside the criterion (has not learned) one:
    - The network has its lowest error, without having learned all patterns

$$RMS = \sqrt{\frac{(0^2 + 0^2 + 0^2 + .4^2)}{4}} = 0.2$$