Neural Networks

Henrik I Christensen

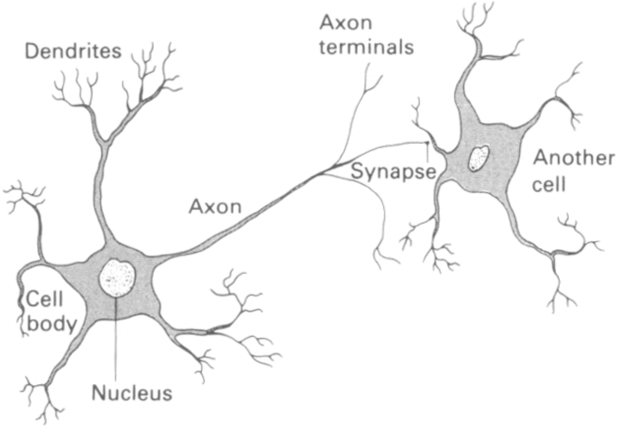
Robotics & Intelligent Machines @ GT
Georgia Institute of Technology,
Atlanta, GA 30332-0280
hic@cc.gatech.edu

Outline

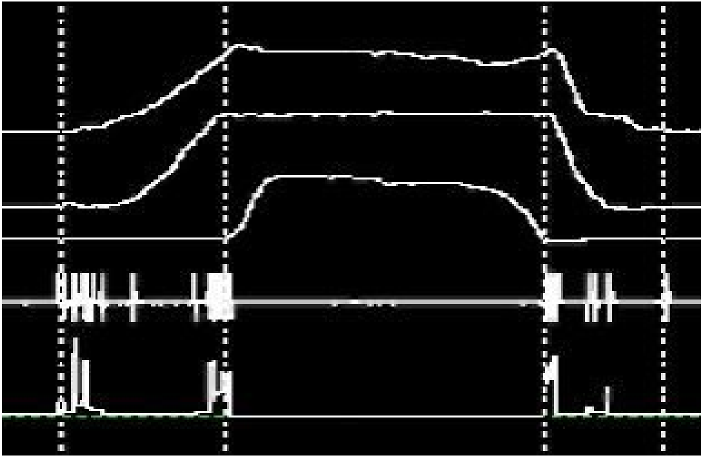
Introduction

- Initial motivation for design from modelling of neural systems
- Perceptrons emerged about same time as we started to have real neural data
- Studies of functional specialization in the brain

Neurons - the motivation



Neural Code Example

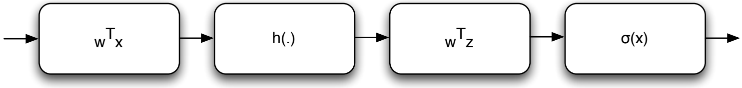


Outline

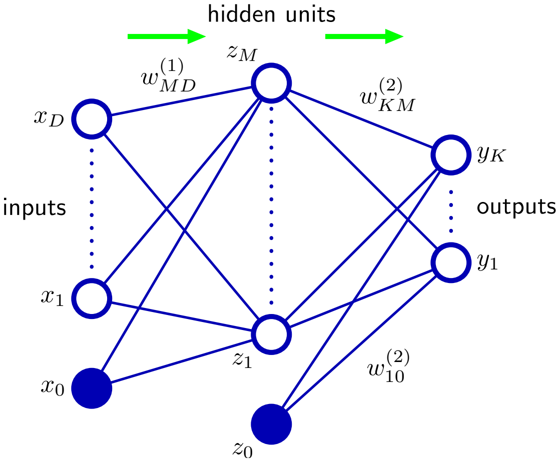
- Outline of ANN architecture
- Formulation of the criteria function
- Optimization of weights
- Example from image analysis
- Next time: Bayesian Neural Networks

Outline

Data Process w. Two-Layer Neural Network



Neural Net Architecture as a Graph



Neural Network Equations

- Consider an input layer

$$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i$$

where w_{j0} and x_0 represent the bias weight / term

- The activation, a_j, is mapped by an activation function

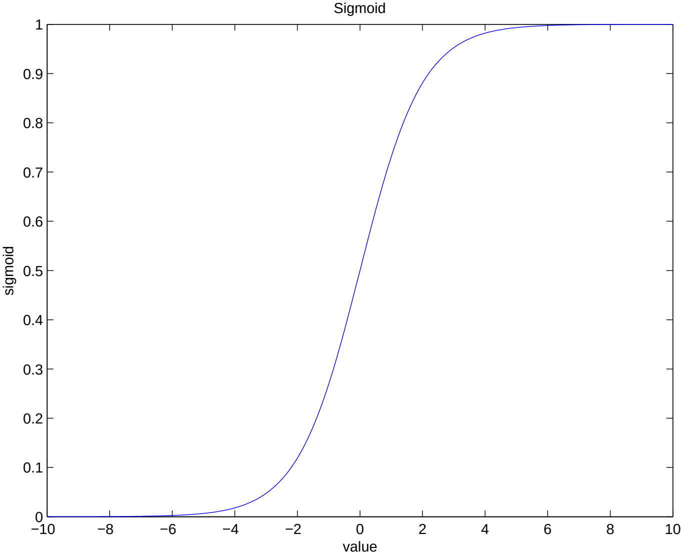
$$z_j = h(a_j)$$

which typically is a Sigmoid or tanh

- The output is considered the hidden activations
- Output unit activations are computed, similarly

$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j$$

Sigmoid - $1/(1 + exp(-v))$



Neural Networks - A few more details

- The full system is then

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

- The information is flowing "forward" through the system
- Naming is sometimes complicated!
  - 3-layer network
  - single-hidden-layer network
  - two-layer network (input/output)

Outline

Training Neural Networks

- For optimization we consider the error function:

$$E(w) = \sum_{n=1}^{N} ||y(x_n, w) - t_n||^2$$

- The optimization is similar to earlier searches
- Objective

$$\nabla E(w) = 0$$

- Due to non-linearity closed form solution is a challenge
- Newton-Raphson type solutions are possible

$$\Delta w = -H^{-1} \nabla E_w$$

- Often an iterated solution is realistic

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

Error Backpropagation

- Consider the error composed of parts

$$E(w) = \sum_{n=1}^{N} E_n(w)$$

- Considering errors by parts we get

$$y_k = \sum_i w_{ki} x_i$$

with the error

$$E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2$$

the associated gradient is

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj}) x_{ni}$$

Computing gradients

- Given

$$a_j = \sum_i w_{ji} z_i$$

  and

$$z_j = h(a_j)$$

- The gradient is (using chain rule)

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$$

- We already know

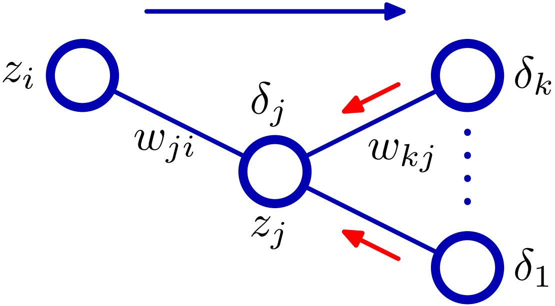
$$\frac{\partial E_n}{\partial a_j} = (y_k - t_j) = \delta_j$$

  and

$$\frac{\partial a_j}{\partial w_{ji}} = z_i$$

Updating of weights

- Updating backwards in the systems



- Error Propagation

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$

Update Algorithm

1. Enter a training sample x_n, propagate and compare to expected value $t_n, y(x_n)$
2. Evaluate δ_k at all outputs
3. Backpropagate δ to correct hidden unit weights
4. Evaluate derivatives to correct input level weights

Issues related to training of networks

- The Sigmoid is "linear" at 0 so random values around 0 is a good start.
- Be aware that training a network too much could result in over fitting
- There can be multiple hidden layers

Outline

Small Example

- From (Le Cun 1989) on state of the art of ANN's for recognition
- Recognition of handwritten characters has been widely studied
- Still considered an important benchmark for new recognition methods

ZIP code data

- Data normalized to 16x16 pixels

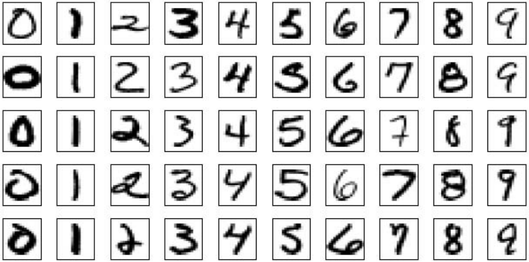


FIGURE 11.9. *Examples of training cases from ZIP code data. Each image is a* 16×16 *8-bit grayscale representation of a handwritten digit.*

- 320 digits in training set and 160 digits in test set

Different types of networks

- No hidden layer - pure 1 level regression
- 1 hidden layer with 12 hidden units - fully connected
- 2 hidden layers and local connectivity
- 2 hidden layers, locally connected and weight sharing
- 2 hidden layers, locally connected and 2 level weight sharing

Example - Net Architectures

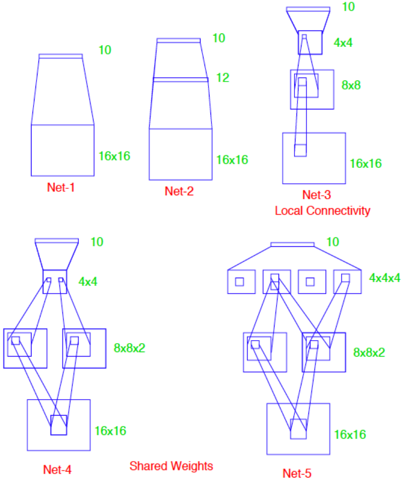


FIGURE 11.10. *Architecture of the five networks used in the ZIP code example.*

Example Results

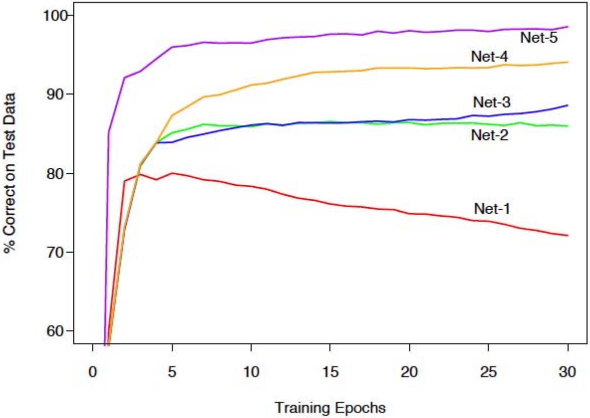


FIGURE 11.11. *Test performance curves, as a function of the number of training epochs, for the five networks of Table 11.1 applied to the ZIP code data. (?)*

Example - Summary

- Careful design of network architectures is important
- Neural Networks offer a rich variety of solutions
- Later results have shown improved performance with SVN's

Outline

Summary

- Neural networks are general approximators
- Useful both for regression and discrimination
- Some would term them - "self-parameterized lookup tables"
- There is a rich community engaged in design of systems
- Rich variety of optimization techniques