COMP9414/9814/3411: Artificial Intelligence

13. Neural Networks

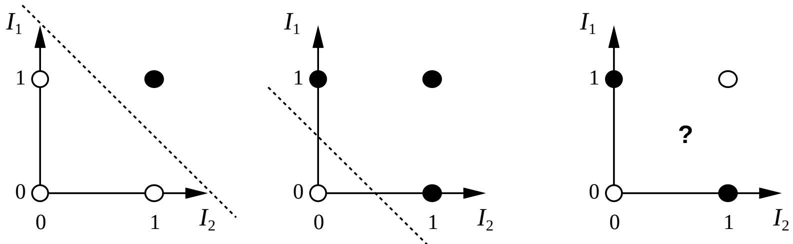
Russell & Norvig: 18.7

Outline

- Multi-Layer Neural Networks
- Backpropagation
- Application - ALVINN
- Training Tips

Recall: Limitations of Perceptrons

Problem: many useful functions are not linearly separable (e.g. XOR)



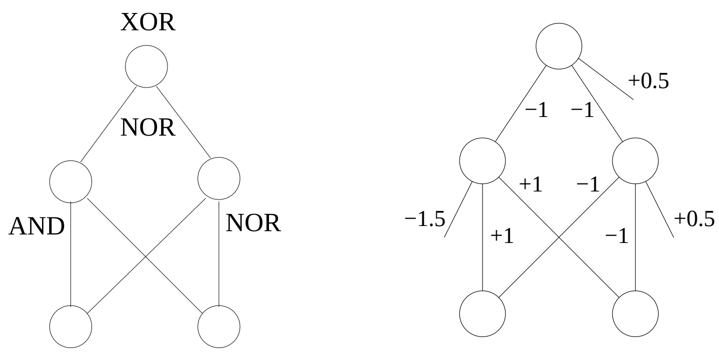
(a) I_1 **and** I_2 **(b)** I_1 **or** I_2 **(c)** I_1 **xor** I_2

Possible solution:

x_1 XOR x_2 can be written as: (x_1 AND x_2) NOR (x_1 NOR x_2)

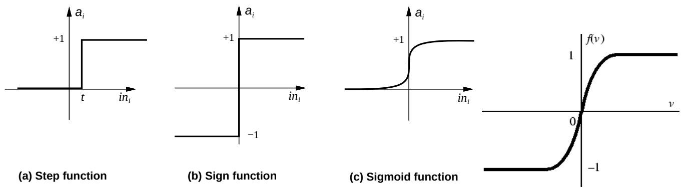
Recall that AND, OR and NOR can be implemented by perceptrons.

Multi-Layer Neural Networks



Problem: How can we train it to learn a new function? (credit assignment)

Key Idea



(a) Step function (b) Sign function (c) Sigmoid function

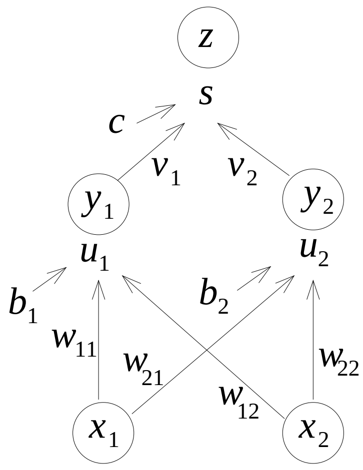
Replace the (discontinuous) step function with a differentiable function, such as the sigmoid:

$$g(s) = \frac{1}{1+e^{-s}}$$

or hyperbolic tangent

$$g(s) = \tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} = 2\Big(\frac{1}{1+e^{-2s}}\Big) - 1$$

Forward Pass



$$\begin{aligned}
u_1 &= b_1 + w_{11}x_1 + w_{12}x_2 \\
y_1 &= g(u_1) \\
s &= c + v_1 y_1 + v_2 y_2 \\
z &= g(s) \\
E &= \frac{1}{2}\sum (z-t)^2
\end{aligned}$$

Gradient Descent

We define an **error function** E to be (half) the sum over all input patterns of the square of the difference between actual output and desired output

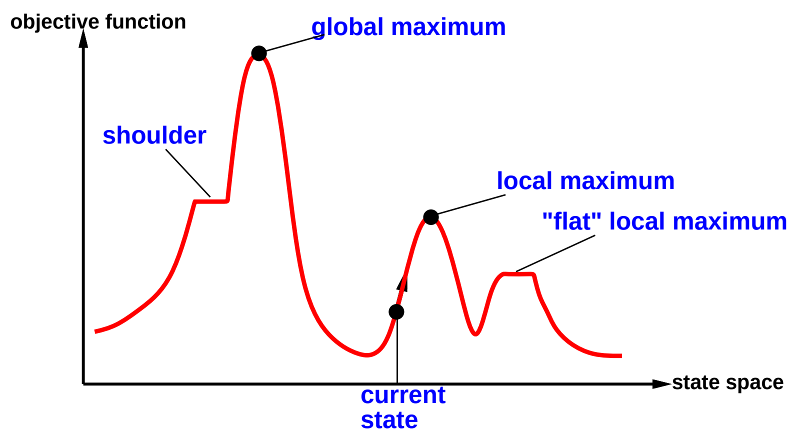
$$E = \frac{1}{2}\sum (z-t)^2$$

If we think of E as height, it defines an error **landscape** on the weight space. The aim is to find a set of weights for which E is very low.
This is done by moving in the steepest downhill direction.

$$w \leftarrow w - \eta\,\frac{\partial E}{\partial w}$$

Parameter η is called the learning rate.

Local Search in Weight Space



Chain Rule

If, say

$$y = y(u)$$

$$u = u(x)$$

Then

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x}$$

This principle can be used to compute the partial derivatives in an efficient and localized manner. Note that the transfer function must be differentiable (usually sigmoid, or tanh).

Note: if $z(s) = \frac{1}{1+e^{-s}}$, $\qquad z'(s) = z(1-z)$.

if $z(s) = \tanh(s)$, $\qquad z'(s) = 1 - z^2$.

Backpropagation

Partial Derivatives

$$\begin{aligned}
\frac{\partial E}{\partial z} &= z - t \\
\frac{dz}{ds} &= g'(s) = z(1-z) \\
\frac{\partial s}{\partial y_1} &= v_1 \\
\frac{dy_1}{du_1} &= y_1(1-y_1)
\end{aligned}$$

Useful notation

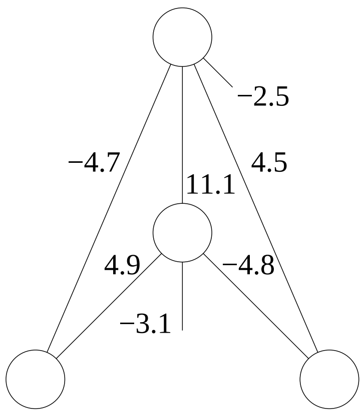
$$\delta_{\text{out}} = \frac{\partial E}{\partial s} \quad \delta_1 = \frac{\partial E}{\partial u_1} \quad \delta_2 = \frac{\partial E}{\partial u_2}$$

Then

$$\begin{aligned}
\delta_{\text{out}} &= (z-t)\, z\, (1-z) \\
\frac{\partial E}{\partial v_1} &= \delta_{\text{out}}\, y_1 \\
\delta_1 &= \delta_{\text{out}}\, v_1\, y_1\, (1-y_1) \\
\frac{\partial E}{\partial w_{11}} &= \delta_1\, x_1
\end{aligned}$$

Partial derivatives can be calculated efficiently by packpropagating deltas through the network.

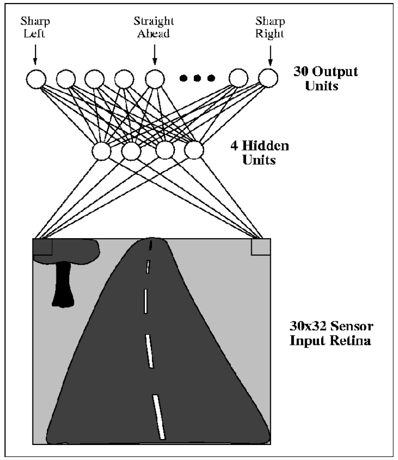
Trained XOR Network

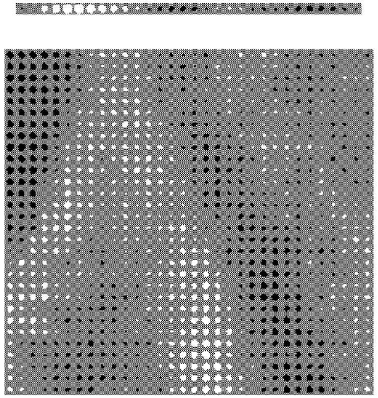


Neural Network – Applications

- Autonomous Driving
- Game Playing
- Credit Card Fraud Detection
- Handwriting Recognition
- Financial Prediction

ALVINN





ALVINN

- Autonomous Land Vehicle In a Neural Network
- later version included a sonar range finder
  - 8×32 range finder input retina
  - 29 hidden units
  - 45 output units
- Supervised Learning, from human actions (Behavioral Cloning)
  - additional “transformed” training items to cover emergency situations
- drove autonomously from coast to coast

Training Tips

- re-scale inputs and outputs to be in the range 0 to 1 or -1 to 1
- initialize weights to very small random values
- on-line or batch learning
- three different ways to prevent overfitting:
  - limit the number of hidden nodes or connections
  - limit the training time, using a validation set
  - weight decay
- adjust learning rate (and momentum) to suit the particular task