Artificial Neural Networks

Lecture 12

Recurrent Neural Networks

Recurrent Neural Networks

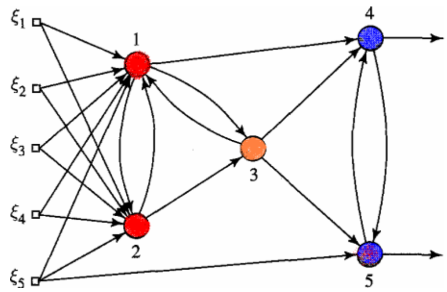
The **conventional feedforward neural networks** can be used to approximate ***any*** **spatiality finite function**. That is, for functions which have a *fixed* input space there is always a way of encoding these functions as neural networks.
For example in function approximation, we can use the automatic learning techniques such as backpropagation to find the weights of the network if sufficient samples from the function is available.

Recurrent neural networks are fundamentally different from feedforward architectures in the sense that they not only operate on an input space but also on an **internal *state* space**.

These are proposed to learn sequential or time varying patterns.

Recurrent Neural Networks

Recurrent Neural Networks, unlike the feed-forward neural networks, contain the feedback connections among the neurons.



Three subsets of neurons are presented in the recurrent networks:

1. ***Input neurons***
2. Output neurons
3. Hidden neurons, which are neither input nor output neurons.

Note that a neuron can be simultaneously an input and output neuron; such neurons are said to be ***autoassociative.***

Recurrent Neural Networks

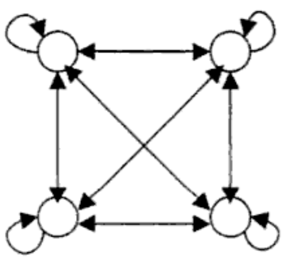


Figure 1. An example of a fully connected recurrent neural network.

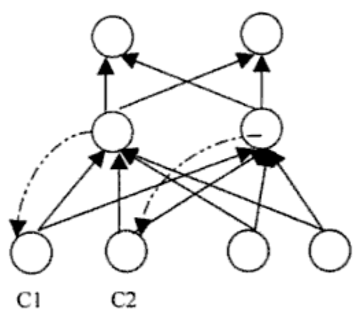
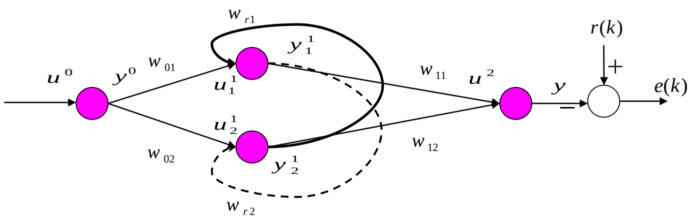


Figure 2. An example of a simple recurrent network.

Recurrent Neural Networks



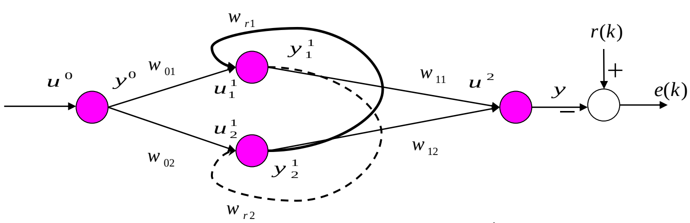
Forward Equations:

$$y^0(k) = u^0(k)$$

$$u^1(k) = \begin{bmatrix} w_{01}(k)y_0(k) + w_{r1}(k)y_2^1(k-1) \\ w_{02}(k)y_0(k) + w_{r2}(k)y_1^1(k-1) \end{bmatrix} \qquad y^1(k) = f_1(u^1(k))$$

$$u^2(k) = w_{11}(k)y_1^1(k) + w_{12}(k)y_2^1(k) \qquad y(k) = f(u^2(k))$$

Recurrent Neural Networks



$$e(k) = r(k) - y(k) \quad \longleftrightarrow \quad E = \frac{1}{2}\sum_k \|e(k)\|^2$$

Back Propagation Equations:

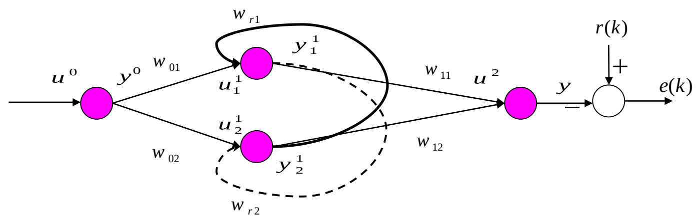
$$\Delta w_{1*} = -\eta \frac{\partial E}{\partial w_{1*}} \qquad u^1(k) = \begin{bmatrix} w_{01}(k)y_0(k) + w_{r1}(k)y_2^1(k-1) \\ w_{02}(k)y_0(k) + w_{r2}(k)y_1^1(k-1) \end{bmatrix}$$

$$\Delta w_{0*} = -\eta \frac{\partial E}{\partial w_{0*}} = \eta e(k) \frac{\partial y}{\partial u^2} \frac{\partial u^2}{\partial y^1} \frac{\partial y^1}{\partial u^1} \frac{\partial u^1}{\partial w_{0*}} \qquad u^2(k) = w_{11}(k)y_1^1(k) + w_{12}(k)y_2^1(k)$$

$$\frac{\partial u^1}{\partial w_{0*}} = y^0 + w_{r*} \frac{\partial y_*^1}{\partial w_{0*}}$$

→ ***Dynamic Back-propagation***

Recurrent Neural Networks



Back Propagation Equations:

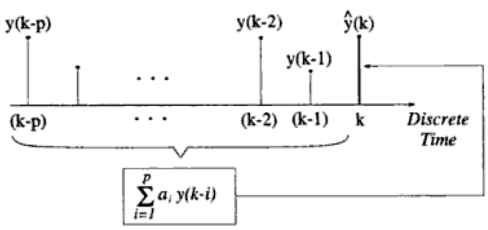
$$u^1(k) = \begin{bmatrix} w_{01}(k)y_0(k) + w_{r1}(k)y_2^1(k-1) \\ w_{02}(k)y_0(k) + w_{r2}(k)y_1^1(k-1) \end{bmatrix}$$

$$\Delta w_{r*} = -\eta \frac{\partial E}{\partial w_{r*}} = \eta e(k) \frac{\partial y}{\partial u^2} \frac{\partial u^2}{\partial y^1} \frac{\partial y^1}{\partial u^1} \frac{\partial u^1}{\partial w_{r*}}$$

$$\frac{\partial u^1}{\partial w_{r1}} = y_2^1 + w_{r1} \frac{\partial y_2^1}{\partial w_{r1}} \qquad \frac{\partial u^1}{\partial w_{r2}} = y_1^1 + w_{r1} \frac{\partial y_1^1}{\partial w_{r1}}$$

Linear Prediction

Linear Prediction:



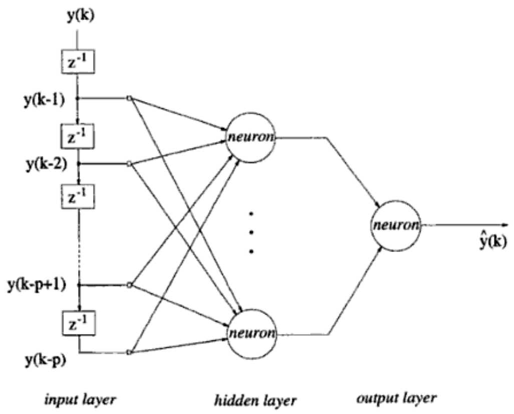
$$\hat{y}(k) = \sum_{i=1}^{p} a_i y(k-i)$$

$$e(k) = y(k) - \hat{y}(k) = y(k) - \sum_{i=1}^{p} a_i y(k-i)$$

The estimation of the parameters a_i is based on minimizing a function of error.

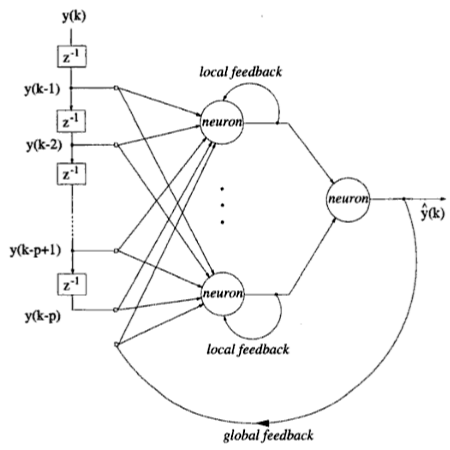
Prediction using FF Neural Network

F.F. Neural Network structure for Prediction:



Prediction using Recurrent N. N.

Recurrent Neural Network architecture for Prediction:



Example for one step ahead Prediction

$$\underbrace{y(k-n),\quad y(k-n+1),\quad \cdots,\quad y(k-n+p-1)}_{\downarrow \atop \hat{y}(k-n+p)},\quad y(k-n+p)$$

Window size: **P**

$$e(k-n+p)=y(k-n+p)-\hat{y}(k-n+p)$$

It is used for back-propagatoin.

$$y(k-n),\quad \underbrace{y(k-n+1)\quad \cdots\quad y(k-n+p-1)\quad y(k-n+p)}_{\downarrow \atop \hat{y}(k-n+p+1)},\quad y(k-n+p+1)$$

$$e(k-n+p+1)=y(k-n+p+1)-\hat{y}(k-n+p+1)$$

Example for one step ahead Prediction

$$y(k-n),\ \cdots,\ y(k-p-1),\ \underbrace{y(k-p),\quad \cdots,\quad y(k-2),\quad y(k-1)}_{\downarrow \atop \hat{y}(k)}\quad \boxed{x=?}$$

$$\Longrightarrow \quad x=\hat{y}(k)$$

4^{th} Mini Project

In this project, a chaotic time series is considered, *logistic* ***map***, whose dynamics is governed by the following difference equation

Window size = 5

$$x(n) = 4x(n-1)\left(1 - x(n-1)\right)$$

* Do this project using MLP and compare the results.

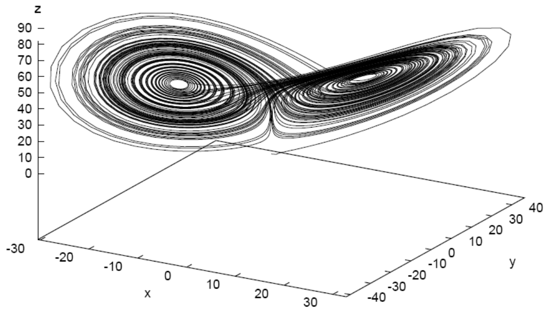
Final Project

In this project, a typical time series like the Lorenz data should be employed to one step ahead prediction by using of any neural network.

Time step = 0.01
Window size = 5

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = -xz + r(x - y) \\ \dot{z} = xy - bz \end{cases}$$

$r = 45.92,\ b = 4,\ \sigma = 16.5$



Literature Cited

The material of this lecture is based on:

[1] Mikael Boden. *A guide to recurrent neural networks and backpropagation*, Halmstad University, 2001.

[2] Danilo P. Mandic, Jonathon A. Chambers, **Recurrent neural networks for prediction: learning algorithms, architectures**, 2001.

[3] R.J.Frank, N.Davey, S.P.Hunt, *Time Series Prediction and Neural Networks*, Department of Computer Science, University of Hertfordshire, Hatfield, UK.