ANN ee543

CHAPTER VII

Learning in Recurrent Networks

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

Introduction

- We have examined the dynamics of recurrent neural networks in detail in Chapter 2.
- Then in Chapter 3, we used them as associative memory with fixed weights.
- In this chapter, the backpropagation learning algorithm that we have considered for feedforward networks in Chapter 6 will be extended to recurrent neural networks [Almeida 87, 88].
- Therefore, the weights of the recurrent network will be adapted in order to use it as associative memory.
- Such a network is expected to converge to the desired output pattern when the associated pattern is applied at the network inputs.

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

- Consider the recurrent system shown in the Figure 7.1, in which there are *n* neurons, some of them being input units, and some others outputs.
- In such a network, the units, which are neither input nor output, are called hidden neurons.

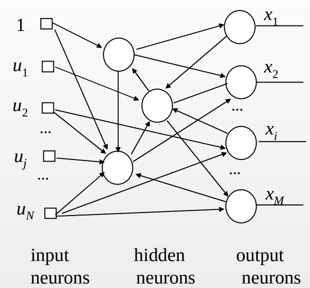

Figure 7.1 Recurrent network architecture

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

- We will assume a network dynamic defined as:

$$\frac{dx_i}{dt} = -x_i + f\left(\sum_j w_{ji} x_j + \theta_i\right) \qquad (7.1.1)$$

- This may be written equivalently as

$$\frac{da_i}{dt} = -a_i + \sum_j w_{ji} f(a_i) + \theta_i \qquad (7.1.2)$$

through a linear transformation.

ANN ee543

CHAPTER VI : ***Learning in Recurrent Networks***

7.1. Recurrent Backpropagation

- Our goal is to update the weights of the network so that it will be able to remember predefined associations, $\mu^k=(\mathbf{u}^k,\mathbf{y}^k)$, $\mathbf{u}^k\in R^N$, $\mathbf{y}^k\in R^N$, $k=1..K$.
- With no loss of generality, we extended here the input vector $\mathbf{u}$ such that $u_i=0$ if the neuron i is not an input neuron. Furthermore, we will simply ignore the outputs of the unrelated neurons.
- We apply an input $\mathbf{u}^k$ to the network by setting

$$\theta_i = u_i^k \quad i=1..N \qquad (7.1.3)$$

- Therefore, we desire the network with an initial state $\mathbf{x}(0)=\mathbf{x}^{k0}$ to converge to

$$\mathbf{x}^k(\infty)=\mathbf{x}^{k\infty}=\mathbf{y}^k \qquad (7.1.4)$$

whenever $\mathbf{u}^k$ is applied as input to the network.

ANN ee543

CHAPTER VI : ***Learning in Recurrent Networks***

7.1. Recurrent Backpropagation

- The recurrent backpropagation algorithm, updates the connection weights aiming to minimize the error

$$e^k = \tfrac{1}{2}\sum_i (\varepsilon_i^k)^2 \qquad (7.1.5)$$

so that the mean error is also minimized

- $$e = \langle(\varepsilon^k)^2\rangle \qquad (7.1.6)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

- Notice that, e^k and e are scalar values while ε^k is a vector defined as

$$\boldsymbol{\varepsilon}^k = \mathbf{y}^k - \mathbf{x}^k \quad (7.1.7)$$

whose i^{th} component ε_i^k, $i=1..M$, is

$$\varepsilon_i^k = \alpha_i (y_i^k - x_i^k) \quad (7.1.8)$$

- In equation (7.1.8) the coefficient α_i used to discriminate between the output neurons and the others by setting its value as

$$\alpha_i = \begin{cases} 1 & \text{if } i \text{ is an output neuron} \\ 0 & \text{otherwise} \end{cases} \quad (7.1.9)$$

- Therefore, the neurons, which are not output, will have no effect on the error.

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

- Notice that, if an input $\mathbf{u}^k$ is applied to the network and if it is let to converge to a fixed point $\mathbf{x}^{k\infty}$, the error depends on the weight matrix through these fixed points. The learning algorithm should modify the connection weights so that the fixed points satisfy

$$x_i^{k\infty} = y_i^k \quad (7.1.10)$$

- For this purpose, we let the system to evolve in the weight space along trajectories in the opposite direction of the gradient, that is

$$\frac{d\mathbf{w}}{dt} = -\eta \nabla e^k \quad (7.1.11)$$

- In particular w_{ij} should satisfy

$$\frac{d\, w_{ij}}{dt} = -\eta \frac{\partial e^k}{\partial w_{ij}} \quad (7.1.12)$$

- Here η is a positive constant named the learning rate, which should be chosen so small.

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

- Since,

$$\alpha_i \varepsilon_i = \varepsilon_i \qquad (7.1.13)$$

the partial derivative of e^k given in Eq. (7.1.5) with respect to w_{sr} becomes:

$$\frac{\partial e^k}{\partial w_{sr}} = -\sum_i \varepsilon_i^k \frac{\partial x_i^{k\infty}}{\partial w_{sr}} \qquad (7.1.14)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

- On the other hand, since $\mathbf{x}^k$ is a fixed point, it should satisfy

$$\frac{dx_i^{k\infty}}{dt} = 0 \qquad (7.1.15)$$

for which Eq. (7.1.1) becomes

$$x_i^{k\infty} = f\left(\sum_j w_{ji} x_j^{k\infty} + u_i^k\right) \qquad (7.1.16)$$

- Therefore ,

$$\frac{\partial x_i^{k\infty}}{\partial w_{sr}} = f'(a_i^{k\infty}) \sum_j \left(x_j^{k\infty} \frac{\partial w_{ji}}{\partial w_{sr}} + w_{ji} \frac{\partial x_j^{k\infty}}{\partial w_{sr}}\right) \qquad (7.1.17)$$

where

$$f'(a_i^{k\infty}) = \frac{d\,f(a)}{da} \Big|_{a=\sum_j w_{ij} x_j^{k\infty} + u_i^k} \qquad (7.1.18)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

- Notice that,

$$\frac{\partial w_{ij}}{\partial w_{sr}} = \delta_{js}\delta_{ir} \qquad (7.1.19)$$

where δ_{ij} is the Kronecker delta which have value 1 if $i=j$ and 0 otherwise, resulting

$$\sum_j x_j^{k\infty}\delta_{js}\delta_{ir} = \delta_{ir}x_s^{k\infty} \qquad (7.1.20)$$

- Hence,

$$\frac{\partial x_i^{k\infty}}{\partial w_{sr}} = f'(a_i^{k\infty})(\delta_{ir}x_s^{k\infty} + \sum_j w_{ji}\frac{\partial x_j^{k\infty}}{\partial w_{sr}}) \qquad (7.1.21)$$

- By reorganizing the above equation, we obtain

$$\frac{\partial x_i^{k\infty}}{\partial w_{sr}} - f'(a_i^{k\infty})\sum_j w_{ji}\frac{\partial x_j^{k\infty}}{\partial w_{sr}} = f'(a_i^{k\infty})\delta_{ir}x_s^{k\infty} \qquad (7.1.22)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

Remember

$$\frac{\partial x_i^{k\infty}}{\partial w_{sr}} - f'(a_i^{k\infty})\sum_j w_{ji}\frac{\partial x_j^{k\infty}}{\partial w_{sr}} = f'(a_i^{k\infty})\delta_{ir}x_s^{k\infty} \qquad (7.1.22)$$

- Notice that,

$$\frac{\partial x_i^{k\infty}}{\partial w_{sr}} = \sum_j \delta_{ji}\frac{\partial x_j^{k\infty}}{\partial w_{sr}} \qquad (7.1.23)$$

- Therefore, Eq. (7.1.22), can be written equivalently as,

$$\sum_j \delta_{ji}\frac{\partial x_i^{k\infty}}{\partial w_{sr}} - f'(a_i^{k\infty})\sum_j w_{ji}\frac{\partial x_j^{k\infty}}{\partial w_{sr}} = f'(a_i^{k\infty})\delta_{ir}x_s^{k\infty} \qquad (7.1.24)$$

or,

$$\sum_j ((\delta_{ji} - w_{ji}f'(a_i^{k\infty}))\frac{\partial x_j^{k\infty}}{\partial w_{sr}} = \delta_{ir}f'(a_i^{k\infty})x_s^{k\infty} \qquad (7.1.25)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

Remember

$$\sum_j ((\delta_{ji} - w_{ji} f'(a_i^{k\infty}))\frac{\partial x_j^{k\infty}}{\partial w_{sr}} = \delta_{ir} f'(a_i^{k\infty}) x_s^{k\infty} \quad (7.1.25)$$

- If we define matrix $\mathbf{L}^k$ and vector $\mathbf{R}^k$ such that

$$L_{ij}^{k\infty} = \delta_{ij} - f'(a_i^{k\infty}) w_{ji} \quad (7.1.26)$$

and

$$R_i^{k\infty} = \delta_{ir} f'(a_i^{k\infty}) \quad (7.1.27)$$

the equation (7.1.25) results in

$$\mathbf{L}^{k\infty} \frac{\partial}{\partial w_{sr}} \mathbf{x}^{k\infty} = \mathbf{R}^{k\infty} x_s^{k\infty} \quad (7.1.28)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

- Hence, we obtain,

$$\frac{\partial}{\partial w_{sr}} \mathbf{x}^{k\infty} = (\mathbf{L}^{k\infty})^{-1} \mathbf{R} x_s^{k\infty} \quad (7.1.29)$$

- In particular, if we consider the i^{th} row we observe that

$$\frac{\partial}{\partial w_{sr}} x_i^{k\infty} = (\sum_j (L^{k\infty})_{ij}^{-1} R_j) x_s^{k\infty} \quad (7.1.30)$$

- Since

$$\sum_j (L^{k\infty})_{ij}^{-1} R_j = \sum_j (L^{k\infty})_{ij}^{-1} \delta_{jr} f'(a_j^{k\infty}) = (L^{k\infty})_{ir}^{-1} f'(a_r^{k\infty}) \quad (7.1.31)$$

we obtain

$$\frac{\partial}{\partial w_{sr}} x_i^{k\infty} = (L^{k\infty})_{ir}^{-1} f'(a_r^{k\infty}) x_s^{k\infty} \quad (7.1.32)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

Remember

$$\frac{d\,w_{ij}}{dt} = -\eta \frac{\partial e^k}{\partial w_{ij}} \qquad (7.1.12)$$

$$\frac{\partial e^k}{\partial w_{sr}} = -\sum_i \varepsilon_i^k \frac{\partial x_i^{k\infty}}{\partial w_{sr}} \qquad (7.1.14)$$

$$\frac{\partial}{\partial w_{sr}} x_i^{k\infty} = (L^{k\infty})_{ir}^{-1} f'(a_r^{k\infty}) x_s^{k\infty} \qquad (7.1.32)$$

- Insertion of (7.1.32) in equation (7.1.14) and then (7.1.12), results in

$$\frac{d\,w_{sr}}{dt} = \eta \sum_i \varepsilon_i^{k\infty} (L^{k\infty})_{ir}^{-1} f'(a_r^{k\infty}) x_s^{k\infty} \qquad (7.1.33)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

- When the network with input $\mathbf{u}^k$ has converged to $\mathbf{x}^{k\infty}$, the local gradient for recurrent backpropagation at the output of the r^{th} neuron may be defined in analogy with the standard backpropagation as

$$\delta_r^{k\infty} = f'(a_r^{k\infty}) \sum_i \varepsilon_i^{k\infty} (L^{k\infty})_{ir}^{-1} \qquad (7.1.34)$$

- So, it becomes simply

$$\frac{d\,w_{sr}}{dt} = \eta \delta_r^{k\infty} x_s^{k\infty} \qquad (7.1.35)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

- In order to reach the minimum of the error e^k, instead of solving the above equation, we apply the delta rule as it is explained for the steepest descent algorithm:

$$\mathbf{w}(k+1)=\mathbf{w}(k)-\eta\nabla e^k \qquad (7.1.36)$$

in which

$$w_{sr}(k+1)=w_{sr}(k)+\eta\delta_r^{k\infty}x_s^{k\infty} \qquad (7.1.37)$$

for $s=1..N$, $r=1..N$

- The recurrent backpropagation algorithm for recurrent neural network is summarized in the following.

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

Step 0. Initialize weights:
to small random values

Step 1. Apply a sample:
apply to the input a sample vector $\mathbf{u}^k$ having desired output vector $\mathbf{y}^k$

Step 2. Forward Phase:
Let the network relax according to the state transition equation

$$\frac{d}{dt}x_i^k = -x_i^k + f(\sum_j w_{ji}x_j^k + u_i^k)$$

to a fixed point $\mathbf{x}^{k\infty}$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.1. Recurrent Backpropagation

Step 3. Local Gradients:

Compute the local gradient for each unit as:

$$\delta_r^{k\infty} = f'(a_r^{k\infty})\sum_i \varepsilon_i^{k\infty}(\mathbf{L}^{k\infty})_{ir}^{-1}$$

Step 4. Update weights according to the equation

$$w_{sr}(k+1) = w_{sr}(k) + \eta\delta_r^{k\infty}x_s^{k\infty}$$

Step 5. Repeat steps 1-4 for $k+1$, until mean error

$$e = \langle e^k \rangle = \langle \tfrac{1}{2}\sum_i \alpha_i (y_i^k - x_i^{k\infty})^2 \rangle$$

is sufficiently small

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.2 Backward Phase

- Notice that, in the computation of local gradients, it is needed to find out $\mathbf{L}^{-1}$, which requires global information processing.
- In order to overcome this limitation, a local method to compute gradients is proposed in [Almeida 88,89].
- For this purpose an adjoint dynamical system in cooperation with the original recurrent neural network is introduced (Figure 7.2)

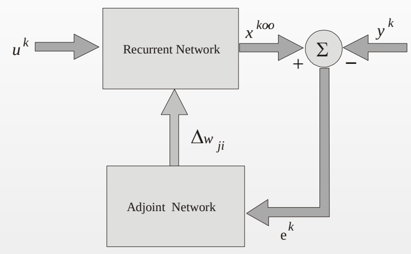

Figure 7.2. Recurrent neural network and cooperating gradient network

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.2 Backward Phase

Remember

$$\delta_r^{k\infty} = f'(a_r^{k\infty})\sum_i \varepsilon_i^{k\infty}(L^{k\infty})_{ir}^{-1} \quad (7.1.34)$$

- The local gradient given in Eq (7.1.34) can be redefined as

$$\delta_r^{k\infty} = f'(a_r^{k*})v_r^{k\infty} \quad (7.2.1)$$

by introducing a new vector variable $\mathbf{v}$ into the system whose r^{th} component defined by the equation

$$v_r^{k\infty} = \sum_i (\mathbf{L}^{k*})_{ir}^{-1}\varepsilon_i^{k*} \quad (7.2.2)$$

in which * is used instead of ∞ in the right handside to denote the fixed values of the recurrent network in order to prevent confusion with the fixed points of the adjoint network.

- They have constant values in the derivations related to the fixed-point $\mathbf{v}^{k\infty}$ of the adjoint dynamic system.

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.2 Backward Phase

- The equation (7.2.2) may be written in the matrix form as

$$\mathbf{v}^{k\infty} = ((\mathbf{L}^{k*})^{-1})^T \varepsilon^{k*} \quad (7.2.3)$$

or equivalently

$$(\mathbf{L}^{k*})^T \mathbf{v}^{k\infty} = \varepsilon^{k*}. \quad (7.2.4)$$

that implies

$$\sum_j L_{jr}^{k*} v_j^{k\infty} = \varepsilon_r^{k*} \quad (7.2.5)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.2 Backward Phase

Remember

$$L_{ij}^{k\infty} = \delta_{ij} - f'(a_i^{k\infty})w_{ji} \qquad (7.1.26)$$

- By using the definition of $\mathbf{L}_{ij}$ given in Eq. (7.1.26), we obtain,

$$\sum_j (\delta_{jr} - f'(a_j^{k*})w_{rj})v_j^{k\infty} = \varepsilon_r^{k*} \qquad (7.2.6)$$

that is

$$0 = -v_r^{k\infty} + \sum_j f'(a_j^{k*})w_{rj}v_j^{k\infty} + \varepsilon_r^{k*} \qquad (7.2.7)$$

- Such a set of equations may be assumed as a fixed-point solution to the dynamical system defined by the equation

$$\frac{dv_r}{dt} = -v_r + \sum_j f'(a_j^{k*})w_{rj}v_j + \varepsilon_r^{k*} \qquad (7.2.8)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.2 Backward Phase

Remember

$$\delta_r^{k\infty} = f'(a_r^{k*})v_r^{k\infty} \qquad (7.2.1)$$

- Therefore $\mathbf{v}^{k\infty}$ and then $\delta^{k\infty}$ in equation (7.2.1) can be obtained by the relaxation of the adjoint dynamical system instead of computing $\mathbf{L}^{-1}$.

- Hence, a backward phase is introduced to the recurrent backpropagation as summarized in the following:

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.2 Backward Phase: Recurrent BP having backward phase

Step 0. Initialize weights: to small random values

Step 1. Apply a sample: apply to the input a sample vector $\mathbf{u}^k$ having desired output vector $\mathbf{y}^k$

Step 2. Forward Phase:
Let the network to relax according to the state transition equation

$$\frac{d}{dt}x_i^k(t) = -x_i^k + f(\sum_j w_{ji}x_j^k + u_i^k)$$

to a fixed point $\mathbf{x}^{k\infty}$

Step 3. Compute:

$$a_i^{k*} = a_i^{k\infty} = \sum_j w_{ji}x_j^{k\infty} + u_i^k$$

$$f'(a_i^{k*}) = \frac{\partial f}{\partial a}\Big|_{a=a_i^{k*}}$$

$$\varepsilon_i^{k*} = \varepsilon_i^{k\infty} = \alpha_i(y_i^k - x_i^{k\infty}) \quad i = 1..N$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.2 Backward Phase

Step 4. Backward phase for local gradients :
Compute the local gradient for each unit as:

$$\delta_r^{k\infty} = f'(a_r^{k*})v_r^{k\infty}$$

where $v_r^{k\infty}$ is the fixed point solution of the dynamic system defined by the equation:

$$\frac{dv_r}{dt} = -v_r + \sum_j f'(a_j^{k*})w_{rj}v_j(t) + \varepsilon_r^{k*}$$

Step 4. Weight update: update weights according to the equation

$$w_{sr}(k+1) = w_{sr}(k) + \eta\delta_r^{k\infty}x_s^{k\infty}$$

Step 5. Repeat steps 1-4 for k+1, until the mean error

$$e = < e^k > = < \tfrac{1}{2}\sum_i \alpha_i (y_i^k - x_i^{k\infty})^2 >$$

is sufficiently small.

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.3. Stability of Recurrent Backpropagation

Remember

$$\frac{dx_i}{dt} = -x_i + f(\sum_j w_{ji} x_j + \theta_i) \qquad (7.1.1)$$

$$\frac{dv_r}{dt} = -v_r + \sum_j f'(a_j^{k*}) w_{rj} v_j + \varepsilon_r^{k*} \qquad (7.2.8)$$

- Due to difficulty in constructing a Lyapunov function for recurrent backpropagation, a local stability analysis [Almeida 87] is provided in the following. In recurrent backpropagation, we have two adjoint dynamic systems defined by Eqs. (7.1.1) and (7.2.8).
- Let $\mathbf{x}^*$ and $\mathbf{v}^*$ be stable attractors of these systems.
- Now we will introduce small disturbances $\Delta\mathbf{x}$ and $\Delta\mathbf{v}$ at these stable attractors and observe the behaviors of the systems.

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.3. Stability of Recurrent Backpropagation

- First, consider the dynamic system defined by the Eq. (7.1.1) for the forward phase and insert $\mathbf{x}^*+\Delta\mathbf{x}$ instead of $\mathbf{x}$, which results in:

$$\frac{d}{dt}(x_i^* + \Delta x_i) = -(x_i^* + \Delta x_i) + f(\sum_j w_{ji}(x_j^* + \Delta x_j) + u_i) \qquad (7.3.1)$$

satisfying

$$x_i^* = f(\sum_j w_{ji} x_j^* + u_i) \qquad (7.3.2)$$

- If the disturbance $\mathbf{x}$ is small enough, then a function $g(.)$ at $\mathbf{x}^*+\Delta\mathbf{x}$ can be linearized approximately by using the first two terms of the Taylor expansion of the function around $\mathbf{x}^*$, which is

$$g(\mathbf{x}^* + \Delta\mathbf{x}) \cong g(\mathbf{x}^*) + \nabla g(\mathbf{x}^*)^{\mathrm{T}} \Delta\mathbf{x} \qquad (7.3.3)$$

where $\nabla g(\mathbf{x}^*)$ is the gradient of $g(.)$ evaluated at $\mathbf{x}^*$.

ANN ee543

CHAPTER VI : ***Learning in Recurrent Networks***

7.3. Stability of Recurrent Backpropagation

- Therefore, *f* (.) in Eq. (7.3.1) can be approximated as

$$f(\sum_j w_{ji}(x_j^* + \Delta x_j) + u_i)$$
$$= f(\sum_j w_{ji} x_j^* + u_i) + \sum_j f'(\sum_j w_{ji} x_j^* + u_i) w_{ji} \Delta x_j \qquad (7.3.4)$$

where *f'*(.) is the derivative of *f* (.).

- Notice that

$$a_i^* = \sum_j w_{ji} x_i^* + u_i \qquad (7.3.5)$$

- Therefore, insertion of Eqs. (7.3.2) and (7.3.5) in equation (7.3.4) results in

$$f(\sum_j w_{ji}(x_j^* + \Delta x_j) + u_i) = x_i^* + \sum_j f'(a_i^*) w_{ji} \Delta x_j \qquad (7.3.6)$$

ANN ee543

CHAPTER VI : ***Learning in Recurrent Networks***

7.3. Stability of Recurrent Backpropagation

- Furthermore, notice that

$$\frac{d}{dt}(x_i^* + \Delta x_i) = \frac{d}{dt}\Delta x_i \qquad (7.3.7)$$

- Therefore, by inserting equations (7.3.6) and (7.3.7) in equation (7.3.1), it becomes

$$\frac{d\Delta x_i}{dt} = -\Delta x_i + \sum_j f'(a_i^*) w_{ji} \Delta x_j \qquad (7.3.8)$$

- This may be written equivalently as

$$\frac{d\,\Delta x_i}{dt} = -\sum_j (\delta_{ij} - f'(a_i^*) w_{ji}) \Delta x_j \qquad (7.3.9)$$

- Referring to the definition of L_{ij} given by Eq. (7.1.26), it becomes

$$\frac{d\,\Delta x_i}{dt} = -\sum_j L_{ij}^* \Delta x_j \qquad (7.3.10)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.3. Stability of Recurrent Backpropagation

- In a similar manner, the dynamic system defined for the backward phase by Eq. (7.2.8) at $\mathbf{v}^*+\Delta\mathbf{v}$ becomes

$$\frac{d}{dt}(v_i^* + \Delta v_i) = -(v_i^* + \Delta v_i) + \sum_j f'(a_j^*) w_{ij} (v_j^* + \Delta v_j) + \varepsilon_i^* \qquad (7.3.11)$$

satisfying

$$v_i^* = \sum_j f'(a_j^*) w_{ij} v_j^* + \varepsilon_i^* \qquad (7.3.12)$$

- When the disturbance $\Delta\mathbf{v}$ in is small enough, then linearization in Eq. (7.3.11) results in

$$\frac{d\Delta v_i}{dt} = -\sum_j (\delta_{ji} - f'(a_j^*) w_{ij}) \Delta v_j \qquad (7.3.13)$$

- This can be written shortly

$$\frac{d\Delta v_i}{dt} = -\sum_j L_{ji}^* \Delta v_j \qquad (7.3.14)$$

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.3. Stability of Recurrent Backpropagation

- In matrix notation, the equation (7.3.10) may be written as

$$\frac{d}{dt}\Delta\mathbf{x} = -\mathbf{L}^*\Delta\mathbf{x} \qquad (7.3.15)$$

- In addition, the equation (7.3.14) is

$$\frac{d\Delta\mathbf{v}}{dt} = -(\mathbf{L}^*)^{\mathsf{T}}\Delta\mathbf{v} \qquad (7.3.16)$$

- If the matrix $\mathbf{L}^*$ has distinct eigenvalues, then the complete solution for the system of homogeneous linear differential equation given by (7.3.15) is in the form

$$\Delta\mathbf{x}(t) = \sum_j \gamma_j \xi_j e^{-\lambda_j t} \qquad (7.3.17)$$

- where ξ_j is the eigenvector corresponding to the eigenvalue λ_j of $\mathbf{L}^*$ and γ_j is any real constant to be determined by the initial condition.

ANN ee543

CHAPTER VI : *Learning in Recurrent Networks*

7.3. Stability of Recurrent Backpropagation

- On the other hand, since $\mathbf{L}^{*T}$ has the same eigenvalues as $\mathbf{L}^*$, the solution (7.3.16) will be the same as given in Eq. (7.3.17) except the coefficients, that is

$$\Delta\mathbf{v}(t) = \sum_j \beta_j \xi_j e^{-\lambda_j t} \qquad (7.3.18)$$

- If it is true that each λ_j has a positive real value then the convergence of both $\mathbf{x}(t)$ and $\mathbf{y}(t)$ to vector $\mathbf{0}$ are guaranteed.
- It should be noticed that, if weight vector $\mathbf{w}$ is symmetric, it has real eigenvalues.
- Since $\mathbf{L}$ can be written as

$$\mathbf{L} = \mathbf{D}(1) - \mathbf{D}(f'(a_i))\mathbf{W} \qquad (7.3.19)$$

where $\mathbf{D}(c_i)$ represents diagonal matrix having i^{th} diagonal entry as c_i, real eigenvalues of $\mathbf{W}$ imply that they are also real for $\mathbf{L}$.