Lecture Series on

Intelligent Control

Lecture 12

Artificial Neural Networks
Multilayer Perceptrons

Kwang Y. Lee
Professor of Electrical & Computer Engineering
Baylor University
Waco, TX 76706, USA
Kwang_Y_Lee@baylor.edu

1

Multilayer Perceptrons

4.2 Multilayer Perceptrons

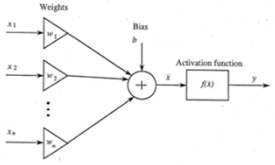

Figure 4.6: Single neuron model.

4.2.1 The Neuron

For a single neuron, suppose that we use x_i, $i = 1, 2, \ldots, n$, to denote its inputs and suppose that it has a single output y. Figure 4.6 shows the neuron. Such a neuron first forms a weighted sum of the inputs

$$\tilde{x} = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

2

Multilayer Perceptrons

The processing that the neuron performs on this $\tilde{x}$ signal is represented with an "activation function." This activation function is represented with a function f, and the output that it computes is

$$y = f(\tilde{x}) = f\left(\left(\sum_{i=1}^{n} w_i x_i\right) + b\right) \tag{4.1}$$

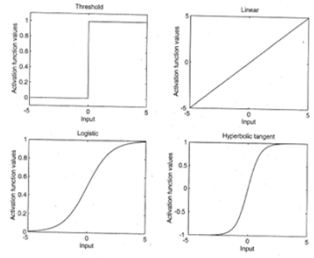

Figure 4.7: Activation functions for neurons.

3

Multilayer Perceptrons

The manner in which the neuron fires is defined by the activation function f. There are many ways to define the activation function:

- *Threshold function:* For this type of activation function we have

$$f(\bar{x}) = \begin{cases} 1 & \text{if } \bar{x} \geq 0 \\ 0 & \text{if } \bar{x} < 0 \end{cases}$$

so that once the input signal $\bar{x}$ is above zero the neuron turns on

- *Linear function:* For this type of activation function we simply have

$$f(\bar{x}) = \bar{x}$$

and we think of the neuron being on when $f(\bar{x}) > 0$ and off when $f(\bar{x})$ ≤ 0

- *Logistic function:* For this type of activation function, which is a type of "sigmoid function," we have

$$f(\bar{x}) = \frac{1}{1+\exp(-\bar{x})} \tag{4.2}$$

so that the input signal $\bar{x}$ continuously turns on the neuron

- *Hyperbolic tangent function:* There are many functions that take on a shape that is sigmoidal. For instance, one that is often used in neural networks is the hyperbolic tangent function

$$f(\bar{x}) = \tanh(\bar{x}) = \frac{1-\exp(-2\bar{x})}{1+\exp(-2\bar{x})}$$

4

Multilayer Perceptrons

4.2.2 Feedforward Network of Neurons

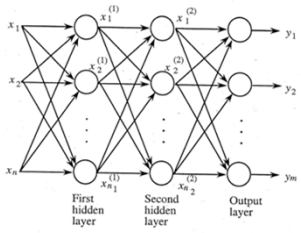

Figure 4.8: Multilayer perceptron model.

5

Multilayer Perceptrons

The neurons in the first layer of the multilayer perceptron perform computations, and the outputs of these neurons are given by

$$x_j^{(1)} = f_j^{(1)}\left(\left(\sum_{i=1}^{n} w_{ij}^{(1)} x_i\right) + b_j^{(1)}\right)$$

with $j = 1, 2, \ldots, n_1$. The neurons in the second layer of the multilayer perceptron perform computations, and the outputs of these neurons are given by

$$x_j^{(2)} = f_j^{(2)}\left(\left(\sum_{i=1}^{n_1} w_{ij}^{(2)} x_i^{(1)}\right) + b_j^{(2)}\right)$$

with $j = 1, 2, \ldots, n_2$. The neurons in the third layer of the multilayer perceptron perform computations, and the outputs of these neurons are given by

$$y_j = f_j\left(\left(\sum_{i=1}^{n_2} w_{ij} x_i^{(2)}\right) + b_j\right)$$

with $j = 1, 2, \ldots, m$.

For convenience, we sometimes use

$$y = F_{mlp}(x, \theta)$$

to denote the multilayer perceptron where θ is a parameter vector that holds all the tunable weights and biases of the multilayer perceptron.

6

Multilayer Perceptrons

4.3 Example: Multilayer Perceptron for Tanker Ship Steering

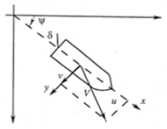

Figure 4.9: Tanker ship steering problem.

$$\dddot{\psi}(t) + \left(\frac{1}{\tau_1} + \frac{1}{\tau_2}\right)\ddot{\psi}(t) + \left(\frac{1}{\tau_1\,\tau_2}\right) H(\dot{\psi}(t)) = \frac{K}{\tau_1\,\tau_2}\left(\tau_3\,\dot{\delta}(t) + \delta(t)\right) \quad (4.5)$$

where $H(\dot{\psi})$ is a nonlinear function of $\dot{\psi}(t)$. The function $H(\dot{\psi})$ can be found from the relationship between δ and $\dot{\psi}$ in steady state such that $\dddot{\psi} = \ddot{\psi} = \dot{\delta} = 0$. An experiment known as the "spiral test" has shown that $H(\dot{\psi})$ can be approximated by

$$H(\dot{\psi}) = \bar{a}\dot{\psi}^3 + \bar{b}\dot{\psi}$$

7

Multilayer Perceptrons

Simulation of Nonlinear Systems

Suppose that the system to be simulated can be represented by the ordinary differential equation

$$\begin{aligned} \dot{x}(t) &= f(x(t), r(t), t) \\ y &= g(x(t), r(t), t) \end{aligned} \quad (4.6)$$

where $x = [x_1, x_2, \ldots, x_n]^\top$ is a state vector, $f = [f_1, f_2, \ldots, f_n]^\top$ is a vector of nonlinear functions, g is a nonlinear function that maps the states and reference input to the output of the system, and $x(0)$ is the initial state. Note that f and g are, in general, time-varying functions due to the explicit dependence on the time variable t. To simulate a nonlinear system, we will assume that the nonlinear ordinary differential equations are put into the form in Equation (4.6).

Euler's Method: Now, to simulate Equation (4.6), we could simply use Euler's method to approximate the derivative $\dot{x}$ in Equation (4.6) as

$$\begin{aligned} \frac{x(kh+h) - x(kh)}{h} &= f(x(kh), r(kh), kh) \\ y &= g(x(kh), r(kh), kh) \end{aligned} \quad (4.7)$$

8

Multilayer Perceptrons

The Runge-Kutta Method: While Euler's method is easy to understand and implement in code, sometimes to get good accuracy the value of h must be chosen to be very small. Most often, to get good simulation accuracy, more sophisticated methods are used, such as the Runge-Kutta method with adaptive step size or predictor-corrector methods. In the fourth-order Runge-Kutta method, we begin with Equation (4.6) and a given $x(0)$ and let

$$x(kh+h) = x(kh) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (4.8)$$

where the four vectors

$$\begin{aligned} k_1 &= hf(x(kh), r(kh), kh) \\ k_2 &= hf\left(x(kh) + \frac{k_1}{2}, r\left(kh + \frac{h}{2}\right), kh + \frac{h}{2}\right) \\ k_3 &= hf\left(x(kh) + \frac{k_2}{2}, r\left(kh + \frac{h}{2}\right), kh + \frac{h}{2}\right) \\ k_4 &= hf\left(x(kh) + k_3, r(kh+h), kh + h\right) \end{aligned}$$

9

Multilayer Perceptrons

Simulating the Ship and a Digital Controller

Next, we need to convert the n^{th}-order nonlinear ordinary differential equations representing the ship to n first-order ordinary differential equations; for convenience, let

$$a = \left(\frac{1}{\tau_1} + \frac{1}{\tau_2}\right)$$

$$b = \left(\frac{1}{\tau_1 \tau_2}\right)$$

$$c = \frac{K\tau_3}{\tau_1 \tau_2}$$

and

$$d = \frac{K}{\tau_1 \tau_2}$$

We would like the model in the form

$$\begin{aligned} \dot{x}(t) &= f(x(t), \delta(t)) \\ y(t) &= g(x(t), \delta(t)) \end{aligned}$$

where $x(t) = [x_1(t), x_2(t), x_3(t)]^\top$ and $f = [f_1, f_2, f_3]^\top$ for use in a nonlinear simulation program. We need to choose $\dot{x}_i$ so that f_i depends only on x_i and δ for $i = 1, 2, 3$. We have

$$\dddot{\psi}(t) = -a\ddot{\psi}(t) - bH(\dot{\psi}(t)) + c\dot{\delta}(t) + d\delta(t) \qquad (4.9)$$

10

Multilayer Perceptrons

Choose

$$\dot{x}_3(t) = \dddot{\psi}(t) - c\dot{\delta}(t)$$

so that f_3 will not depend on $c\dot{\delta}(t)$ and

$$x_3(t) = \ddot{\psi}(t) - c\delta(t)$$

Choose $\dot{x}_2(t) = \ddot{\psi}(t)$ so that $x_2(t) = \dot{\psi}(t)$. Finally, choose $x_1(t) = \psi$. This gives us

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) = f_1(x(t), \delta(t)) \\ \dot{x}_2(t) &= x_3(t) + c\delta(t) = f_2(x(t), \delta(t)) \\ \dot{x}_3(t) &= -a\ddot{\psi}(t) - bH(\dot{\psi}(t)) + d\delta(t) \end{aligned}$$

But, $\ddot{\psi}(t) = x_3(t) + c\delta(t)$, $\dot{\psi}(t) = x_2(t)$, and $H(x_2) = x_2^3(t) + x_2(t)$ so

$$\dot{x}_3(t) = -a\left(x_3(t) + c\delta(t)\right) - b\left(x_2^3(t) + x_2(t)\right) + d\delta(t) = f_3(x(t), \delta(t))$$

Also, we have $\psi = g(x, \psi_r) = x_1$. This provides the proper equations for the simulation. Next, suppose that the initial conditions are $\psi(0) = \dot{\psi}(0) = \ddot{\psi}(0) = 0$. This implies that $x_1(0) = x_2(0) = 0$ and $x_3(0) = \ddot{\psi}(0) - c\delta(0)$ or $x_3(0) = -c\delta(0)$.

For the ship steering problem we let the integration step size be $h = 1$ sec. and $\alpha = 10$ so that $T = \alpha h = 10$ sec. (i.e., the controller is implemented on a digital computer with a sampling period of $T = 10$ sec. so that a new plant input is calculated every 10 sec. and applied to the rudder). We will use this same approach for all the simulations for the tanker ship in this book.

11

Multilayer Perceptrons

4.3.2 Construction of a Multilayer Perceptron for Ship Steering

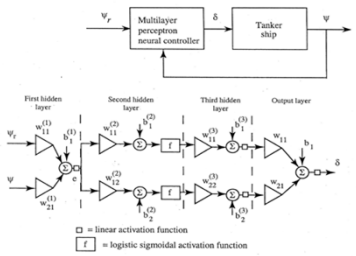

Figure 4.11: A multilayer perceptron for tanker ship steering.

12

Multilayer Perceptrons

Consider Figure 4.11. There, we use a four layer perceptron with both linear and logistic sigmoidal activation functions. First, consider the first hidden layer. For this we choose $w_{11}^{(1)} = 1$, $w_{21}^{(1)} = -1$, and $b_1^{(1)} = 0$. Hence, we see that the output of the first layer is the heading error

$$e = \psi_r - \psi$$

To a control engineer this may seem to be an odd approach to implement a simple summing junction to provide the heading error; however, it is interesting that a neuron can provide a method to compare two signals, something that is certainly of fundamental importance in making control decisions.

Choosing Weights and Biases: Building Nonlinearities with Smooth Step Functions

Next, we explain how to pick the weights and biases for the remaining layers. To do this, view the perceptron in Figure 4.11 as having two "paths" of processing from the signal e that is the output of the first hidden layer to the output δ. Imagine that you remove the path on the bottom and first focus on constructing the path on the top. We will think of the top path as being used to regulate the ship heading when

$$e = \psi_r - \psi \geq 0$$

In this case we want to have a *negative* rudder input.

13

Multilayer Perceptrons

To specify values for the weights and biases we think of each neuron as providing a type of "smooth switching" (a smooth step function as shown in Figure 4.7, e.g., for the logistic function) and tune the weights and biases to adjust these. Note that when only the top path is considered, we have

$$\delta = w_{11}\left(\frac{w_{11}^{(3)}}{1+\exp(-\bar{x})} + b_1^{(3)}\right) + b_1$$

where

$$\bar{x} = b_1^{(2)} + w_{11}^{(2)} e$$

The parameters in these equations affect the shape of the nonlinearity from e to δ in the following manner:

- b_1, $b_1^{(3)}$: Shift the mapping up and down.
- w_{11}, $w_{11}^{(3)}$: Scale the vertical axis.
- $b_1^{(2)}$: Shifts the smooth step (logistic function) horizontally, with $b_1^{(2)} > 0$ shifting it to the *left*.
- $w_{11}^{(2)}$: Scale the horizontal axis (you may think of this as a type of gain for the function, at least locally).

14

Multilayer Perceptrons

Using these ideas, choose

$$\begin{aligned} b_1 &= b_1^{(3)} = 0 \\ w_{11} &= 1 \\ w_{11}^{(3)} &= -\frac{80\pi}{180} \\ b_1^{(2)} &= -\frac{200\pi}{180} \\ w_{11}^{(2)} &= 10 \end{aligned}$$

With these choices we get the nonlinear mapping shown in the top plot of Figure 4.12. The general shape of the function is appropriate to use as a controller for $e > 0$ since it provides negative rudder input values for positive values of error, and it provides a type of proportionality between the size of e and the size of δ. Notice that the choice of $w_{11}^{(3)}$ results in the perceptron providing a maximum negative rudder deflection of -80 degrees. The choice of $b_1^{(2)}$ simply shifts the function to the right, so that the value of the function near $e = 0$ provides $\delta \approx 0$. The value of $w_{11}^{(2)}$ affects the slope of the function as $e > 0$ increases in size; if $w_{11}^{(2)}$ were chosen to be larger, then it would reach the maximum negative value of -80 degrees quicker as the size of e increases. This completes the construction of the perceptron for the top path, which is dedicated to control for the case where $e \geq 0$.

15

Multilayer Perceptrons

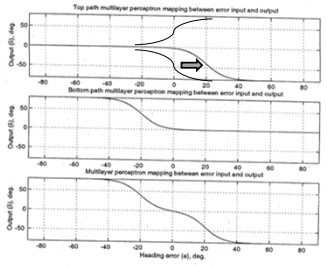

Figure 4.12: Multilayer perceptron mappings, top plot is for the top path of the perceptron from e to δ, middle plot is for the bottom path of the perceptron from e to δ, bottom plot is for the entire perceptron from e to δ.

16

Multilayer Perceptrons

Using the same ideas for case $e < 0$.

$$\begin{aligned} b_2^{(3)} &= \frac{80\pi}{180} \\ w_{21} &= 1 \\ w_{22}^{(3)} &= -\frac{80\pi}{180} \\ b_2^{(2)} &= \frac{200\pi}{180} \\ w_{12}^{(2)} &= 10 \end{aligned}$$

The resulting nonlinearity implemented by the bottom path is shown in the middle plot of Figure 4.12. First, note that its general shape is appropriate to use as a controller for the case where $e < 0$; as the size of e increases in the negative direction, increasingly positive values of a rudder input δ are provided to try to decrease the value of ψ to the given ψ_r. The values of $w_{22}^{(3)}$, $b_2^{(2)}$, and $w_{12}^{(2)}$ were chosen in a similar way as the corresponding values for the top path were chosen. The value of $b_2^{(3)}$ was chosen to shift the nonlinearity up by 80 degrees. The choice for w_{21} completes the specification of the output layer, which simply sums the functions generated by the top and bottom paths, and results in the overall mapping from e to δ shown in the bottom plot of Figure 4.12 (i.e., when both the top and the bottom paths in Figure 4.11 are used). This completes the construction of the multilayer perceptron for regulating the ship heading.

17

Multilayer Perceptrons

4.3.3 Multilayer Perceptron Stimulus-Response Characteristics

The multilayer perceptron construction procedure in the last subsection showed how to construct the controller

$$\delta = F_{mlp}(\psi_r, \psi)$$

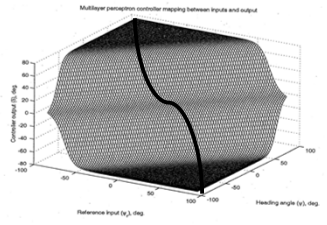

Figure 4.13: Control surface implemented by the multilayer perceptron for tanker ship steering.

18

Multilayer Perceptrons

4.3.4 Behavior of the Ship Controlled by the Multilayer Perceptron

To evaluate how a neural network can regulate the ship heading, we will use simulation studies for a variety of operating conditions for the ship.

Closed-loop Response, Nominal Conditions

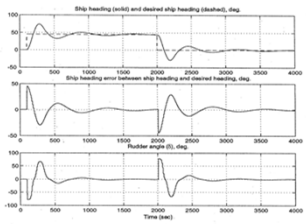

Figure 4.14: Closed-loop response resulting from using the multilayer perceptron for tanker ship steering.

19

Multilayer Perceptrons

Effects of Wind on Heading Regulation

Next, consider the effects of a wind disturbance on the ship. Suppose that the wind is gusting. It hits the side of the ship and moves the ship a bit, which then pushes the rudder against the water which induces a torque to move the rudder. To model this we add a disturbance onto the rudder angle input by adding

$$0.5\left(\frac{\pi}{180}\right)\sin\left(2\pi(0.001)t\right)$$

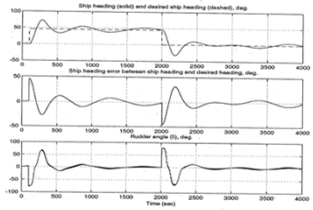

Figure 4.15: Closed-loop response resulting from using the multilayer perceptron for tanker ship steering, with wind.

20

Multilayer Perceptrons

Effects of Speed Changes on Heading Regulation

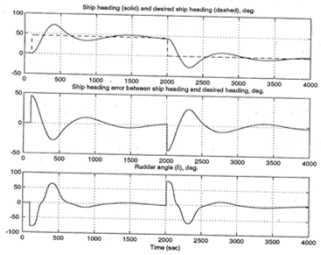

Figure 4.16: Closed-loop response resulting from using the multilayer perceptron for tanker ship steering with speed of 3 meters/sec.

21

Multilayer Perceptrons

Effects of Sensor Noise and Weight Changes on Heading Regulation

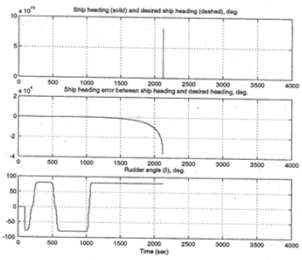

Figure 4.17: Closed-loop response resulting from using the multilayer perceptron for tanker ship steering, full rather than ballast conditions.

22

Multilayer Perceptrons

4.4 Radial Basis Function Neural Networks

A locally tuned overlapping receptive field is found in parts of the cerebral cortex, in the visual cortex, and in other parts of the brain. The radial basis function neural network model is based on these biological systems (but once again, the model is not necessarily accurate, just inspired by its biological counterpart).

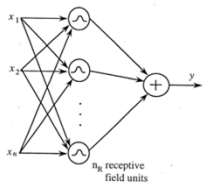

Figure 4.18: Radial basis function neural network model.

http://en.wikipedia.org/wiki/Cerebral_cortex

23

Multilayer Perceptrons

$$y = F_{rbf}(x, \theta) = \sum_{i=1}^{n_R} b_i R_i(x) \tag{4.10}$$

is the output of the radial basis function neural network, and θ holds the b_i parameters and possibly the parameters of the receptive field units.

There are several possible choices for the "receptive field units" $R_i(x)$:

1. We could choose
$$R_i(x) = \exp\left(-\frac{|x - c^i|^2}{(\sigma^i)^2}\right)$$
2. We could choose
$$R_i(x) = \frac{1}{1 + \exp\left(-\frac{|x-c^i|^2}{(\sigma^i)^2}\right)}$$
3. In each of the above cases you can choose to make the σ^i also depend on the input dimension (which makes sense if the input dimensions are scaled differently). In this case for 1 above, for example, we would have $\sigma^i = [\sigma_1^i, \sigma_2^i, \ldots, \sigma_n^i]^T$ and
$$R_i(x) = \exp\left(-\sum_{j=1}^{n} \frac{(x_j - c_j^i)^2}{(\sigma_j^i)^2}\right)$$
where σ_j^i is the spread for the j^{th} input for the i^{th} receptive field unit. This is the approach that we will use in the example in the next section.

24

Multilayer Perceptrons

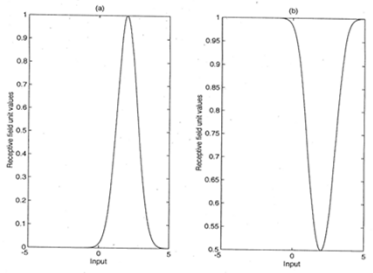

Figure 4.19: Example receptive field units.

25

Multilayer Perceptrons

4.5 Example: Radial Basis Function Neural Network for Ship Steering

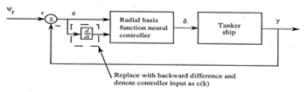

Figure 4.20: Radial basis function neural network used as a controller for ship heading.

$$e = \psi_r - \psi$$

and

$$\dot{e} = \dot{\psi}_r - \dot{\psi}$$

We will, however, use a backward difference approximation to the derivative which we will denote by $c(kT)$,

$$\dot{e} \approx \frac{e(kT) - e(kT - T)}{T} = c(kT)$$

where $T = 10$ sec. and k is an index for the time step (this is an Euler approximation of the derivative and T is the sampling period of the digital controller on which we will implement the controller).

26

Multilayer Perceptrons

Design of a Radial Basis Function Neural Network for Steering

Next, we construct a radial basis function neural network with $n = 2$ inputs, and $n_R = 121$ so we will have to pick 121 strengths b_i, $i = 1, 2, \ldots, 121$. For the $R_i(e(k), c(k))$ we create a uniform grid for the c^i centers, $i = 1, 2, \ldots, 121$.

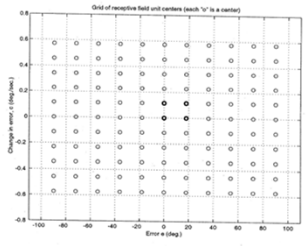

Figure 4.21: Receptive field unit centers.

27

Multilayer Perceptrons

$$e(k) \in [-\frac{\pi}{2}, \frac{\pi}{2}]$$

(which will hold if we do good regulation and do not get fast changes in ψ_r). Via simulations of the ship the angular rate of movement is often such that

$$c(k) \in [-0.01, 0.01]$$

so we will make that assumption. For convenience, we simply create a uniform grid with its four outer corners at $(-\frac{\pi}{2}, -0.01)$, $(-\frac{\pi}{2}, 0.01)$, $(\frac{\pi}{2}, 0.01)$, and $(\frac{\pi}{2}, -0.01)$ with $n_R = 121$ centers uniformly placed at the grid points (i.e., with 11 points along each input dimension). We show the centers of the receptive field units in Figure 4.21.

For the receptive field units we use spreads σ_j^i (i.e., so that the size of the spread depends on which input dimension is used) with

$$\sigma_1^i = 0.7\frac{\pi}{\sqrt{n_R}}$$

and

$$\sigma_2^i = 0.7\frac{0.02}{\sqrt{n_R}}$$

for $i = 1, 2, \ldots, 121$. For σ_1^i the $\frac{\pi}{11}$ factor makes the spread size depend on the number of grid points along the e input dimension (similarly for σ_2^i), and the 0.7 factor was chosen to get a smooth interpolation between adjacent receptive field

28

Multilayer Perceptrons

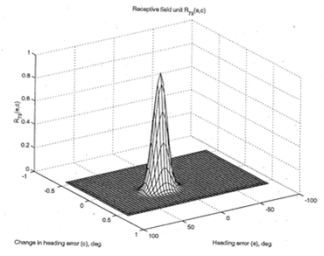

Figure 4.22: Mapping implemented by receptive field unit $R_{73}(e, c)$.

29

Multilayer Perceptrons

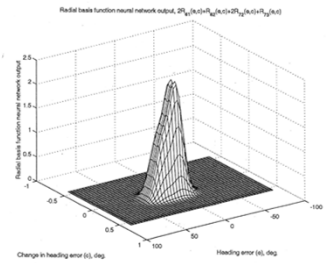

Figure 4.23: Scaling and addition of several receptive field units (i.e., $2R_{61}(e, c) + R_{62}(e, c) + 2R_{72}(e, c) + R_{73}(e, c)$).

30

Multilayer Perceptrons

$$\begin{bmatrix} b_1 & b_{12} & b_{23} & b_{34} & b_{45} & b_{56} & b_{67} & b_{78} & b_{89} & b_{100} & b_{111} \\ b_2 & & & & & \cdots & & & & & b_{112} \\ \vdots & & & & & \cdots & & & & & \vdots \\ b_{11} & b_{22} & b_{33} & b_{44} & b_{55} & b_{66} & b_{77} & b_{88} & b_{99} & b_{110} & b_{121} \end{bmatrix}$$

1.3963	1.3963	1.3963	1.3963	1.3963	1.3963	1.3963	1.0472	0.6981	0.3491	0
1.3963	1.3963	1.3963	1.3963	1.3963	1.3963	1.0472	0.6981	0.3491	0	-0.3491
1.3963	1.3963	1.3963	1.3963	1.3963	1.0472	0.6981	0.3491	0	-0.3491	-0.6981
1.3963	1.3963	1.3963	1.3963	1.0472	0.6981	0.3491	0	-0.3491	-0.6981	-1.0472
1.3963	1.3963	1.3963	1.0472	0.6981	0.3491	0	-0.3491	-0.6981	-1.0472	-1.3963
1.3963	1.3963	1.0472	0.6981	0.3491	0	-0.3491	-0.6981	-1.0472	-1.3963	-1.3963
1.3963	1.0472	0.6981	0.3491	0	-0.3491	-0.6981	-1.0472	-1.3963	-1.3963	-1.3963
1.0472	0.6981	0.3491	0	-0.3491	-0.6981	-1.0472	-1.3963	-1.3963	-1.3963	-1.3963
0.6981	0.3491	0	-0.3491	-0.6981	-1.0472	-1.3963	-1.3963	-1.3963	-1.3963	-1.3963
0.3491	0	-0.3491	-0.6981	-1.0472	-1.3963	-1.3963	-1.3963	-1.3963	-1.3963	-1.3963
0	-0.3491	-0.6981	-1.0472	-1.3963	-1.3963	-1.3963	-1.3963	-1.3963	-1.3963	-1.3963

31

Multilayer Perceptrons

4.5.2 Radial Basis Function Neural Network Stimulus-Response Characteristics

The stimulus-response characteristics of the radial basis function neural network $F_{rbf}(e, c)$ that we just designed are shown in Figure 4.24 in the form of a control surface, similar to how we illustrated the mapping for the multilayer perceptron (note that here the inputs are different).

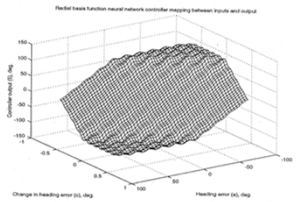

Figure 4.24: Stimulus-response characteristics of the radial basis function neural network for tanker ship heading regulation.

32

Multilayer Perceptrons

4.5.3 Behavior of the Ship Controlled by the Radial Basis Function Neural Network

To study how a radial basis function neural network can operate to regulate the ship heading we will use simulation studies for a variety of operating conditions, the same ones as used in Section 4.3.

Closed-loop Response, Nominal Conditions

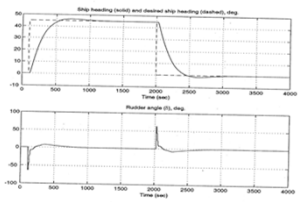

Figure 4.25: Closed-loop response resulting from using the radial basis function neural network for tanker ship steering.

33

Multilayer Perceptrons

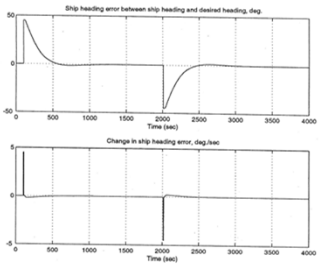

Figure 4.26: Closed-loop response resulting from using the radial basis function neural network for tanker ship steering.

34

Multilayer Perceptrons

Effects of Wind, Speed Changes, Sensor Noise, and Weight Changes on Heading Regulation

Next, consider the effects of a wind disturbance on the ship. In this case, we get the response in Figure 4.27. We see that the wind affects our ability to achieve very good steady state regulation of the ship heading.

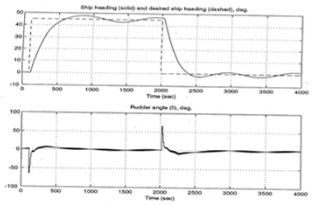

Figure 4.27: Closed-loop response resulting from using the radial basis function neural network for tanker ship steering, with wind.

35

Multilayer Perceptrons

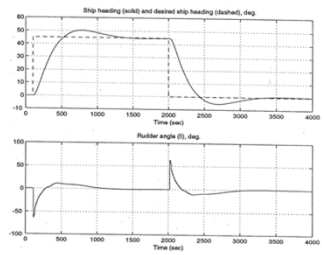

Figure 4.28: Closed-loop response resulting from using the radial basis function neural network for tanker ship steering, speed of 3 meters/sec.

36

Multilayer Perceptrons

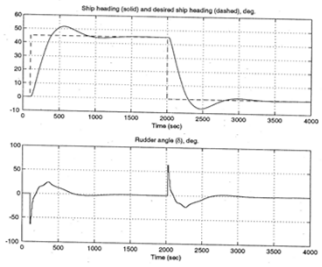

Figure 4.29: Closed-loop response resulting from using the radial basis function neural network for tanker ship steering, full rather than ballast conditions.

37