National Cheng Kung University

NEURAL NETWORKS AND MACHINE LEARNING

NEURAL NETWORKS AND MACHINE LEARNING

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Neural Networks and Machine Learning
Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

To my family

CONTENTS IN BRIEF

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ANN	Artificial Neural Network

GLOSSARY

Node	Artificial Neurons with inner parameters; each node can receive and process incoming information from other nodes and transmit the processed signal towards other nodes in the network.
Artificial Neural Network	computer simulation of how populations of neurons perform a given task.
Distributed Representation	Inherent representations to the ANNs as patterns of activation among many network's elements.
Local Representation	Inherent representation of a node within an ANNs as patterns of activation in a single network's element.
Loss function	The change in the error signal over the set of learning trials.
link	Is the connection between two nodes between a network and have a weight value.
weight	The multiplication coefficient applied to the activation value of the link between two nodes of the network.

INTRODUCTION

The application fields of artificial neural networks is undergoing an exponential growth during this decade and the possible applications in areas such as psychology and cognitive psychology are countless. By the time I got interested in these topics I did not have a strong background in computer science but I could see the potential of these techniques for solving the kind of problems that psychologist and social scientist in general deal with, mainly diagnosis, classification and prediction. Therefore, during my PhD studies at NCKU[1] I decided to approach this topic and write a guide that helped to understand better not only the theory but also its implementation for different kind of problems.

This book introduces the fundamentals to implement these techniques using MATLAB, a computing environment published by MathWorks which can run on many platforms such as Windows and Macintosh. MATLAB is able to perform or use the following

- Matrix Arithmetic - add, divide, inverse, transpose, etc.
- Relational Operators - less than, not equal, etc.
- Logical Operators - AND, OR, NOT, XOR
- Data Analysis - minimum, mean, covariance, etc.
- Elementary functions - sin, cos, log, etc.
- Numerical Linear Algebra - LU decomposition, etc.
- Polynomials - roots, fit polynomial, divide, etc.

[1] National Cheng Kung University

- Non-linear Numerical Methods - solve DE, minimize functions, etc.

More on MATLAB goes in this section...

PART I

BASICS OF NEURAL NETWORKS

CHAPTER 1

BASICS OF NEURAL NETWORKS

1.1 Early conceptions of Neural Networks

One of the earliest conceptions about Neural Networks was proposed by Warren McCulloch and Walter Pitts in 1943, who provided an explanation on the functioning of biological networks and also made simple assumptions about how neurons might operate. From their perspective a neuron had a binary output determined by a threshold value, that is, it could either send out a signal (i.e. being *on*) or not send a signal (i.e. being *off*). Additionally, they assumed the weights of the connections between neurons to be at a fixed value. Under these settings, networks are capable of computing simple logical operations (e.g. AND, OR, and NOT) [Friedenberg and Silverman, 2011, chap. 7].

Donald O. Hebb (1949) was the first to propose how changes among neurons might explain learning. Hebb's rule states that when on cell repeatedly activates another, the strength of the connection between the two cells is increased; in this way, circuits among neurons are formed which are considered to be the neural foundation of learning and memory. Hebb defined two types of cell groupings; a **cell assembly** is a relatively small group of neurons that stimulate each other repeatedly, while a **phase sequence** is a group of connected cell assemblies that fire synchronously, as represented in Figure 1.1. A cell assembly can code a simple perceptual quality, such as *yellow* or *round*, and these qualities could become linked so as to form a phase sequence during learning and code for a higher order concept such as *sun*.

In the 1950's the research on neural networks focused on mimicking the functioning of real biological networks, time when was introduced an artificial nervous system called

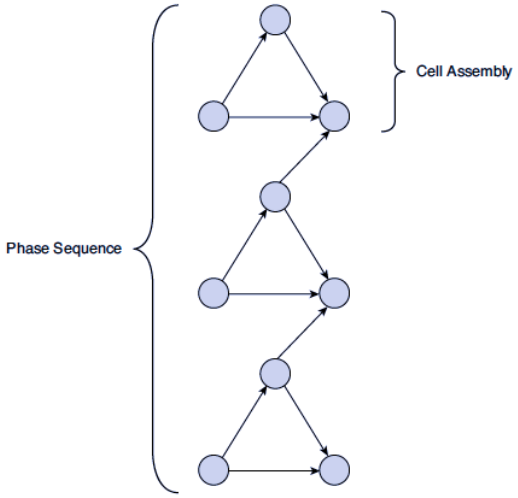


Figure 1.1: A phase sequence consisting of cell assemblies, Friedenberg and Silverman [2011]

the **perceptron**. A perceptron is a neural net designed to detect and recognize patterned information about the world, and its is able to store and use this information (i.e. learn from experience). The learning occurs because the network can modify their connection strengths by comparing their actual output to a desired output. The earliest perceptron was an artificial retina called *Mark I* (Rosenblantt, 1958) which contained a single layer of input units as well as an output layer as the one depicted in Figure 1.2; this network is able to recognize simple visual patterns such as vertical and horizontal lines, but is unable to distinguish more complex patterns due to its relatively weak computing power.

1.2 Mathematical Foundations of Neural Networks

Artificial Neural Networks (ANN) are computational frameworks created to mimic the functioning and operations of biological neuronal networks, these can be thought of as computational simulations of how the actual neurons might perform during a given task, such as pattern recognition or classification, the resemblance with their biological counterparts is termed *biological plausibility*[1]. In an ANN, information is conceived as a pattern of activation within the network, these activation patterns can occur sequentially or simultaneously across different network's **nodes**. **Parallel distributed processing** refers to an architecture in which one computing unit is not required to wait for another to finish its computation before it can begin its work, the units can operate in parallel and are not limited to receive input and process outputs from and towards only a single unit, the previous being some of the ANNs' distinctive features [Friedenberg and Silverman, 2011, chap. 7]. Nodes can be thought of as artificial neurons with inner parameters, thus each node can

[1]This plausibility is demonstrated with the following three criteria,**1** ANNs share general structural and functional correlates with biological networks, **2** are capable of learning and **3** react to damage as biological networks do.

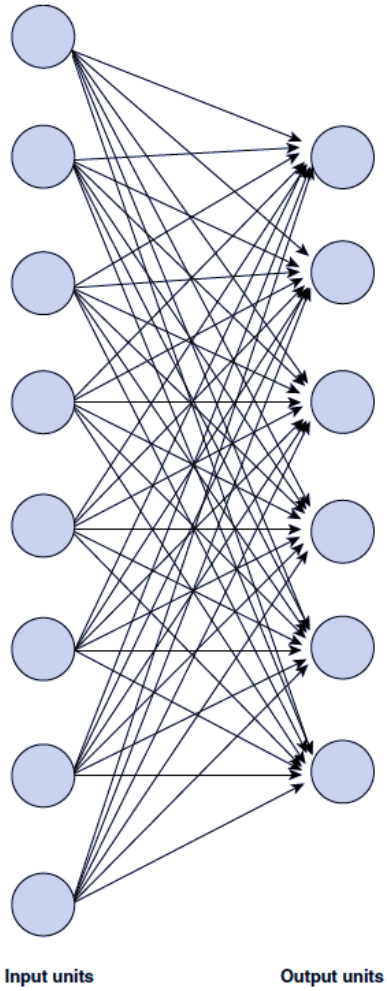


Figure 1.2: Early perceptron with two layers in which each input unit maps onto every output unit, Friedenberg and Silverman [2011]

receive and process incoming information in the same way neurons do, have connections with other nodes and transmit the processed signals towards other nodes in the network.

Researchers that use ANNs are mainly concerned about the overall behavior of the network without paying much attention to symbolic representations or rules; from this approach, the representations are rather inherent to the ANNs as patterns of activation and do not exist in the form of symbols (i.e. *Distributed representation*). This problem solving strategy is referred as to **behavior-based approach**, and leaves the computational details up to the network itself. Additionally the networks are capable of undergoing a learning-like process, the nodes can adaptively change their threshold responses over time by modifying the weights of the links after receiving new information, usually the feedback that results from the validation of the output and the target. For example, within the problems that ANNs can cope with are the classification problems, these involve assigning a categorical label to a new stimulus after a concept formation has been established by the network through a training phase after a series of n trials in which some feedback about the network's performance is used to adjust the overall performance of the network.

The basic computational unit in a neural network is represented as a **node**, after following the inherent mechanisms that characterize it (v.g. set of internal rules), the node sends a signal towards other nodes via their links if, for example, the activation value or threshold for that node is surpassed. A **link** is the connection between two nodes within a network and have a **weight** value which can be understood as the strength of the signal; this weight usually takes values between -1 and 1 and thus the net output of a node is its value times the weight of the link, which can either *inhibit* or *facilitate* the activation of the next node downstream. For example, if the a unit with activation value equal to 1 pass along a link that has a weight of 0.5, it will stimulate the nodes to which it is connected by a factor of 0.5; on the contrary, if the link has a weight of -0.5 is likely to negatively stimulate the nodes to which it is connected. The greater the value of a node's net output, the more likely is that the nodes it is connected to will fire and similarly negative outputs serves the function of shutting down the activation from other nodes [Friedenberg and Silverman, 2011, chap. 7]. The total amount of stimulation that a given node receives can be calculated as the summation of the inputs received by that node specified by a **basis function**, which can be described as the Equation 1.1

$$y_j = f\Big(\sum_{i=1}^{n} a_i w_{ij}\Big) \tag{1.1}$$

where a_i is the value of the i^{th} input signal, and w_{ij} is the weight associated with the connections between the i^{th} and j^{th} node, thus y_j is the summed stimulation over the n inputs towards the j^{th} node within the network [Friedenberg and Silverman, 2011, chap. 7]. Figure 1.3 graphically presents the computations involved within a simple neural network; the activation values are displayed inside each node and the weights are indicated along the links, the output value is indicated at the endpoint of each link. Other authors Jiang et al. [2010], Tsoukalas and Uhrig [1996] have added to this equation the parameter b_i, defined as the bias associated with the i^{th} node, which can be described as the Equation 1.2

$$y_j = f\Big(\sum_{i=1}^{n} a_i w_{ij} + b_i\Big) \tag{1.2}$$

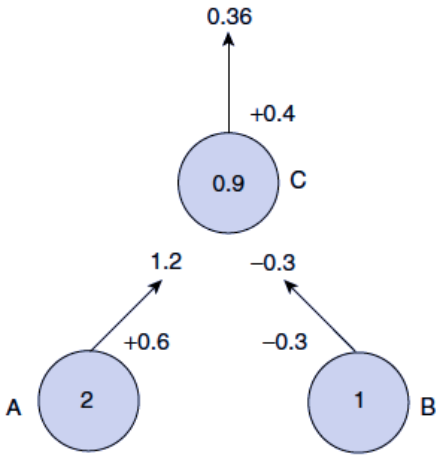


Figure 1.3: A simple neural network, Friedenberg and Silverman [2011]

Afterwards, the *basis function* sends its signal to an **activation function**[2], which maps the strength of the inputs a node receives onto the node's output and can assume its distinctive shape from different mathematical expressions. For example, a linear activation function's outputs is equal to its input, as shown in Equation 1.3. The MATLAB code of such function taking a range between -5 and 5 is like follows; additionally, as to evidence the function's shape it is possible to generate a graphical description, as shown in Figure 1.4a.

$$f(x) = x \tag{1.3}$$

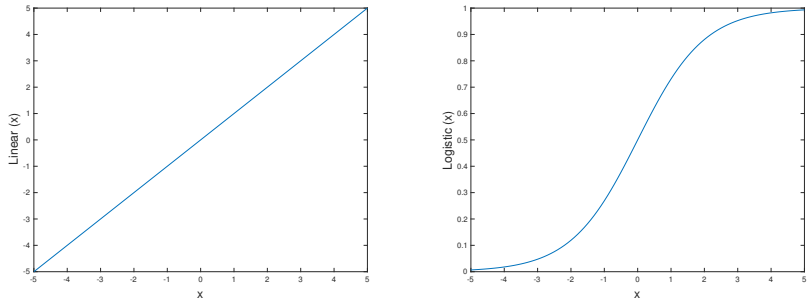
```
x = 5;
x = -(x):0.1:x;
y = x;
plot(y,x);
title('Linear Activation Function','Fontsize',18)
xlabel('x','FontSize',16) ; ylabel('Linear (x)','FontSize',16);
```

There are also several types of non-linear activation functions; for example, differentiable, non-linear activation functions as the logistic function and the hyperbolic tangent function are commonly used in networks which use back-propagation training. The logistic function is described by Equation 1.4

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.4}$$

Where x is the input value and e is the base of the natural logarithm[3]. As can be seen in Figure 1.4b, the output range of this function is between 0 and 1. Below is the MATLAB code used to generate the graph for the logistic function.

[2]also labeled as *transfer function*
[3]The number whose natural logarithm is equal to one, approximately 2.71828



(a) Linear Activation Function. (b) Logistic Activation Function.

Figure 1.4: Activation Functions.

```
x = 5;
x = -(x):0.1:x;
y = zeros(1,length(x));
for n = 1:length(y)
    y(n)=1/(1+exp(1)^(-(x(n))));
end
plot(x,y)
title('Logistic Activation Function','Fontsize',18)
xlabel('x','FontSize',16) ; ylabel('Logistic (x)','FontSize',16);
```

Besides the logistic function described above, the activation function can also set to be either an *a* hard limit, *b* Linear or *c* RBF[4] function. Additionally, non-differentiable non-linear activation functions such as the threshold function (which output is either 0 or 1) and the signum's (which output is either -1 or 1) are sometimes used as outputs of perceptrons and competitive networks.

As can be inferred, it is plausible to specify different activation functions for different nodes within a given network in order to produce specified outputs or deal with information processing problems of different nature Jiang et al. [2010]. Besides, the functions defined above can take a vector as input, such vector could be the multiplication of the input and weight vectors, such that if we have an input vector $x = \begin{bmatrix}1, 3, 6\end{bmatrix}$, and a weight vector $x = \begin{bmatrix}.5, .20, .33\end{bmatrix}$ with a $b = -0.5$ and the activation function is an hyperbolic tangent function the output might be computed as follows

```
x = [1 3 5] ; w = [0.25 -0.25 0.30] ; b = -0.5;
y = tanh(sum(x.*w)+b);
y =

    0.4621
```

Now, the nodes within a neural networks can be organized in several layers, such that the first layer or **input layer** receives a representation of the stimulus, which in turn send outputs towards the nodes of a **hidden layer** which then send stimuli towards to the nodes in the **output layer**; in the final stage, this generates a representation of the response such as the one depicted in Figure 1.5. The **error signal** constitutes the difference between

[4]Radial Basis Function, a non linear transfer function (normally Gaussian)

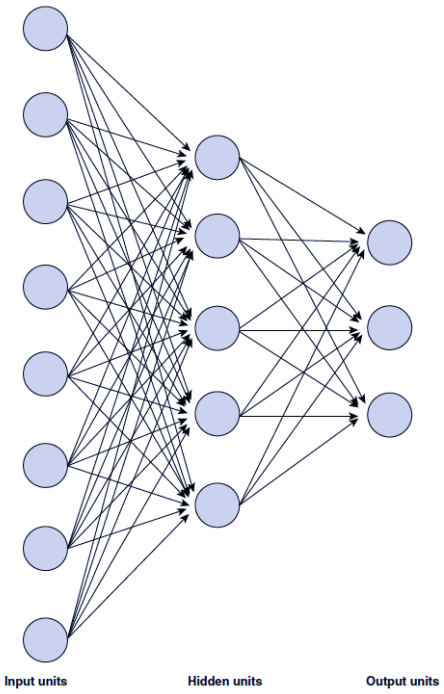


Figure 1.5: A three-layers neural network, Friedenberg and Silverman [2011]

the actual and desired output response of the network given by the **teacher**[5], information that then feeds back to the output layer and is used to modify the weights of the links between the nodes within a network. This interaction can be understood as an underlaying training based on the feedback provided by the error signal which is frequently used in **back-propagation** learning models, formally described as the addition or subtraction from the weights during the learning phase and constitutes the simplest perceptron learning rule which can be described by Equation 1.5. The steps in the training of a three-layers network using the back-propagation learning model is shown in Figure 1.6. The modified weights allow the network to generate a response closer to the desired one in the next trial which accuracy improves giving a certain number of repeated presentations; this kind of training is called the **generalized delta rule** or simply the **back-propagation** learning model.

$$w_{ji}^{new} = w_{ji}^{old} + C(t_j - x_j)a_i \tag{1.5}$$

In Equation 1.5 C is a constant (usually a value < 1) equivalent to the learning rate; t_j is the target value for the output of unit j^{th} following the pattern presentation, and x_j is the actual output value of unit j^{th} that follows the pattern presentation; if a difference between these two values exist some error correction is likely to take place and the weight is subsequently modified. Finally, the term a_i is the activation status of the input unit i^{th}

[5] Also frequently called **target**

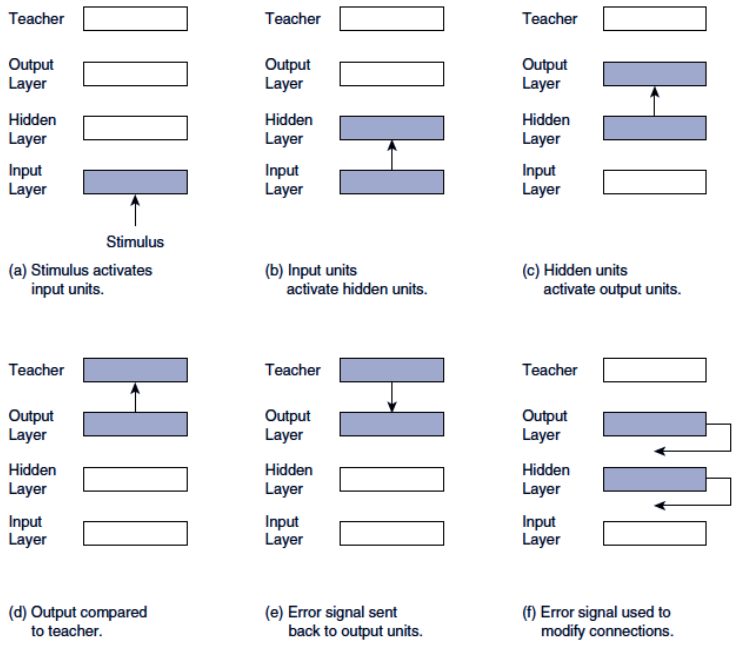


Figure 1.6: Steps in the training of a three-layers network using back-propagation, Friedenberg and Silverman [2011]

which is either 1 or 0 in a given moment, therefore if the value of a_i equals zero (i.e. no activation of node i^{th}) the weight of the link undergoes no change.

1.3 Artificial Neural Network Typologies

The way the network updates the values of its weights over time can be classified into three broad categories according to their **dynamics**, which could be: *a. Convergent*, *b. Oscillatory* or *c. Chaotic*. These typologies vary in the order and pattern of their activation fluctuations which depend on the architecture of the network and on the learning rule implemented. As one example, one might consider the dynamics of a biological neural network in the brain, which is usually oscillatory and chaotic in nature, this is, fluctuates over time and does not settles; on the contrary, in a convergent network the stimulus presentation elicits the trained response in the form of an activation pattern among nodes. In oscillatory and chaotic networks such representation is subject to constant change and might correspond to more global characteristics, such as the networks frequency and phase of activity.

1. *Convergent*: At first the network undergoes a significant amount of activity which then slows down until the network settles back in a stable state. Most ANNs exhibit convergent properties.

2. *Oscillatory*: The weights hold by the links in this network fluctuate periodically in a regular fashion over time.

3. *Chaotic*: The network's activity pattern varies chaotically displaying both, periodic and non-periodic fluctuations.

On the other hand ANNs can also be classified based on their organizing schemes having into account mainly the following criteria: *1. Supervision*, referring to whether the network is given the right answer (i.e. target) for each training trial or not (i.e. Supervised vs Unsupervised), *2. Layers*, according to the number of layers within a network (e.g. single-layer, multi-layer, etc), and *3. Information flow*, on this regard the network can be either feed-forward (i.e. signals being processed can only pass through the network in a single direction) or recurrent (i.e. both directions, forward and backward).

1. *Supervision*: The two categories under this criterion are based on the way the networks learn, being either *supervised* or *unsupervised*; in the first case, networks are presented with target answers for each pattern they receive as input. In the second case the network determine the answer on its own lacking a given correct answer.

2. *Layers*: This criterion makes reference to the number of layers contained within a network, these can either be categorized as *single-layer* (i.e. one layer of nodes), or *multi-layer* (i.e. two, three or more layers).

3. *Information flow*: There are two categories defined by this criterion, a network is either *feed-forward* or *recurrent*. In the first case the activation occurs forwards only through the different layers defined in the network. Information in a recurrent network can flow in two directions, both forward and backwards, the backward flow can be, for example, from output layers to hidden or from output layers to input layers.

Perceptrons as described in section 1.1 are *supervised* networks as the teacher provides the correct information that allows further adjustments of the weights during learning phase; they are *multi-layer* because they contain two or more layers of nodes. Additionally under the back-propagation learning model the operations occur in a feed-forward manner. Perceptrons are typically used for task of pattern classification. **Hopfield-Tank networks** are a type of supervised, laterally connected, single layer networks (Hopfield and Tank, 1985), in which the nodes within the one single layer are connected to every other node. These networks are good at solving optimization problems and for generating cleaner version of input patterns that contain noisy or incomplete version of previously known patterns, nonetheless they could usually end up at a local minima which represents an error state, that is when the error level drops very quickly and the network is unable to perform its task. Some ways to correct for local minima include restarting the learning process at different points and introducing noise.

Kohonen networks[6] are two-layer, unsupervised networks able to create a topological map or spatial representation of the features that receive as input (Konohen, 1990). These maps are similar to the topological maps in the brain such as in the ones within the primary visual cortex; this area represents various features of the visual input such as orientation and ocular dominance. On a different account, the **Adaptive Resonance Theory** network is an example of a multi-layer, unsupervised, recurrent network able to classify input patterns and allocate them in different categories in the absence of a previously known target information (Carpenter and Grossberg, 1988). A disadvantage of the ART network is that individual nodes represent categories so when the node undergoes damage or degradation the information about the category is lost.

[6] Also called *feature maps*

1.4 Medical image analysis with artificial neural networks

All of the previous features that have been discussed about ANNs makes them worth using as research tools in many areas; for example, the use of ANNs have been staggering in the research community of medical imaging in the last decade, some of the common uses in this field include *a)* computer-aided diagnosis, *b)* medical image segmentation and edge detection towards visual content analysis, *c)* medical image registration, *d)* image enhancement and noise suppression, and *e)* functional connectivity simulations Jiang et al. [2010]. Below I will describe in further details some of the characteristics of the last ANNs application in medical imaging processing.

1.4.1 fMRI activation pattern simulation

ANNs are tools that can be used for simulating the connectivity and function of special areas in the brain; for example, semantic networks have been used to simulate the connectivity between a set of concepts and their semantic relationships resembling how semantic memory might be organized in the human brain. From this approach one might have a hierarchical architecture constituted by a **superordinate** category such as *animals*, which at the next level could have exemplars or animal classes, such as *birds* and *cats*, which constitute **ordinate** categories. At the bottom of this hierarchy are nodes that might correspond to other features of each animal such as *can fly* or *have whiskers*, corresponding to their **subordinate** categories [Friedenberg and Silverman, 2011, chap. 7]. This is only one example of how our semantic memory might be organized, nonetheless the models fail to embrace the complexity and diversity of how people from different cultural backgrounds might acquire and construct their semantic networks.

Neural networks can also be used to model other cognitive functions such as speech acquisition and production, and some success in this area have been achieved by simulating a wide range of acoustic, kinematic and neuroimaging data accounting for the control movements involved in speech production. In this approach, the components of the ANN model correspond to several brain regions such as premotor, motor, auditory and somatosensory cortical areas; specific anatomical locations of the model are then estimated to simulate fMRI experiments of syllable production. Alternatively, can be used to validate the inferences about functional connectivity that is evidenced by the fMRI techniques under event-related designs and overall condition of the subject, and partially replicating some results from the psycho-physiological analysis. From this approach the underlying interregional neural interaction was simulated at two levels, the neuronal activity and multi-regional activity from fMRI data. Another approach based on the partial correlation matrix and structural equation modeling (EMS) was proposed to develop data-driven measures of the connection paths as established by fMRI images throughout the use of a neurobilogically realistic ANN modelJiang et al. [2010].

CHAPTER 2

WEB RESOURCES ON NEURAL NETWORKS

The MathWorks offer comprehensive tutorials about this topic with videos and useful code, here are some of the links that can be interesting

- https://www.mathworks.com/solutions/deep-learning/resources.html

Other websites include the following

-

Bibliography

Jay Friedenberg and Gordon Silverman. *Cognitive science: An introduction to the study of mind*. Sage, 2011.

Jianmin Jiang, P Trundle, and Jinchang Ren. Medical image analysis with artificial neural networks. *Computerized Medical Imaging and Graphics*, 34(8):617–631, 2010.

Lefteri H Tsoukalas and Robert E Uhrig. *Fuzzy and neural approaches in engineering*. John Wiley & Sons, Inc., 1996.