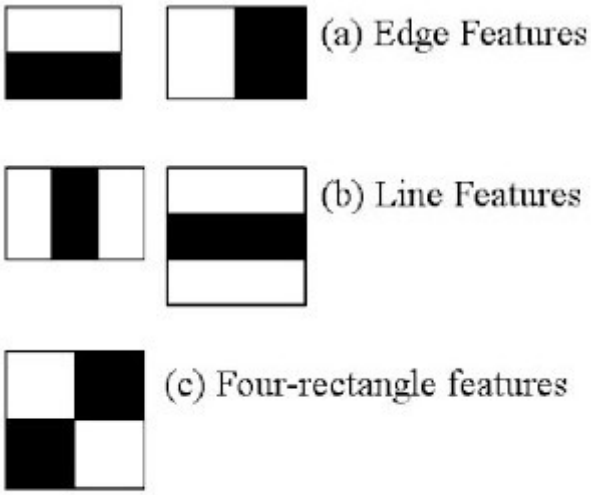
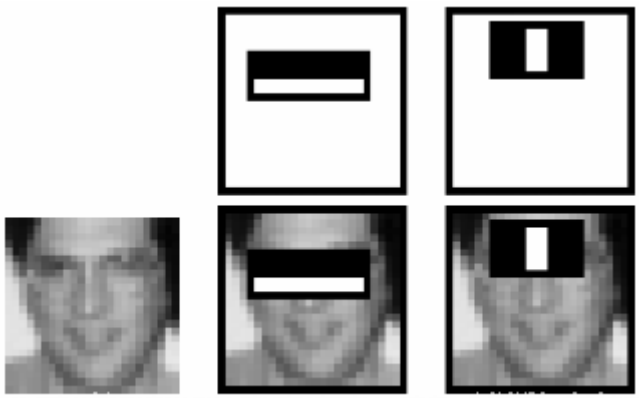
1) Face Detection using Haar Cascades

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.



image



image

2) Biological Neuron

In animals, learning occurs within the brain. If we can understand how the brain works, then there might be things in there for us to copy and use for our machine learning systems.

While the brain is an impressively powerful and complicated system, the basic building blocks that it is made up of are fairly simple and easy to understand.

We'll look at them shortly, but it's worth noting that in computational terms the brain does exactly what we want. It deals with noisy and even inconsistent data, and produces answers that are usually correct from very high dimensional data (such as images) very quickly.

So how does it actually work?

The most basic level, which is the processing units of the brain. These are nerve cells called neurons.

There are lots of them (100 billion = 10^{11} is the figure that is often given) and they come in lots of different types, depending upon their particular task.

However, their general operation is similar in all cases : transmitter chemicals within the fluid of the brain raise or lower the electrical potential inside the body of the neuron. If this membrane potential reaches some threshold, the neuron spikes or fires, and a pulse of fixed strength and duration is sent down the axon.

The axons divide (arborise) into connections to many other neurons, connecting to each of these neurons in a synapse. Each neuron is typically connected to thousands of other neurons, so that it is estimated that there are about 100 trillion (= 10^{14}) synapses within the brain.

After firing, the neuron must wait for some time to recover its energy (the refractory period) before it can fire again.

Each neuron can be viewed as a separate processor, performing a very simple computation: deciding whether or not to fire. This makes the brain a massively parallel computer made up of 10^{11} processing elements.

If that is all there is to the brain, then we should be able to model it inside a computer and end up with animal or human intelligence inside a computer. This is the view of strong AI.

We aren't aiming at anything that grand in this book, but we do want to make programs that learn.

So how does learning occur in the brain? The principal concept is plasticity : modifying the strength of synaptic connections between neurons, and creating new connections. We don't know all of the mechanisms by which the strength of these synapses gets adapted, but one method that does seem to be used was first postulated by Donald Hebb in 1949.

3) Hebb's Rule

Hebb's rule says that the changes in the strength of synaptic connections are proportional to the correlation in the firing of the two connecting neurons.

So if two neurons consistently fire simultaneously, then any connection between them will change in strength, becoming stronger. However, if the two neurons never fire simultaneously, the connection between them will die away.

The idea is that if two neurons both respond to something, then they should be connected.

Example 1 :

Let’s see a trivial example :

Suppose that you have a neuron somewhere that recognises your grandmother (this will probably get input from lots of visual processing neurons, but don’t worry about that). Now if your grandmother always gives you a chocolate bar when she comes to visit, then some neurons, which are happy because you like the taste of chocolate, will also be stimulated. Since these neurons fire at the same time, they will be connected together, and the connection will get stronger over time. So eventually, the sight of your grandmother, even in a photo, will be enough to make you think of chocolate.

Example 2 :

Sound familiar? Pavlov used this idea, called classical conditioning, to train his dogs so that when food was shown to the dogs and the bell was rung at the same time, the neurons for salivating over the food and hearing the bell fired simultaneously, and so became strongly connected. Over time, the strength of the synapse between the neurons that responded to hearing the bell and those that caused the salivation reflex was enough that just hearing the bell caused the salivation neurons to fire in sympathy.

There are other names for this idea that synaptic connections between neurons and assemblies of neurons can be formed when they fire together and can become stronger. It is also known as long term potentiation and neural plasticity, and it does appear to have correlates in real brains.

4) Artificial Neural Networks

Artificial Neural Networks is a very powerful, strong as well as a very complicated Machine Learning technique which mimics a human brain and how it functions. Artificial neural network take their inspiration from the brain.

The human brain is quite different from a computer. Whereas a computer generally has one processor, the brain is composed of a very large (10^{11}) number of processing units, namely, neurons, operating in parallel.

Artificial neural networks (ANNs) provide a general, practical method for learning real-valued, discrete-valued, and vector-valued functions from examples. Algorithms such as Back Propagation use gradient descent to tune network parameters to best fit a training set of input-output pairs.

ANN learning is robust to errors in the training data and has been successfully applied to problems such as interpreting visual scenes, speech recognition, and learning robot control strategies.

For certain types of problems, such as learning to interpret complex real-world sensor data, artificial neural networks are among the most effective learning methods currently known. For example, the BACKPROPAGATION algorithm has proven surprisingly successful in many practical problems such as learning to recognize handwritten characters (LeCun et al. 1989), learning to recognize spoken words (Lang et al. 1990), and learning to recognize faces (Cottrell 1990).

Example , a learned ANN to steer an autonomous vehicle driving at normal speeds on public highways. The input to the neural network is a 30 x 32 grid of pixel intensities obtained from a forward-pointed camera mounted on the vehicle. The network output is the direction in which the vehicle is steered. The ANN is trained to mimic the observed steering commands of a human driving the vehicle for approximately 5 minutes. ALVINN has used its learned networks to

successfully drive at speeds up to 70 miles per hour and for distances of 90 miles on public highways (driving in the left lane of a divided public highway, with other vehicles present).

The kind of representation typical of many ANN systems. Each node (i.e., circle) in the network diagram corresponds to the output of a single network unit, and the lines entering the node from below are its inputs. As can be seen, there are four units that receive inputs directly from all of the 30 x 32 pixels in the image. These are called "hidden" units because their output is available only within the network and is not available as part of the global network output. Each of these four hidden units computes a single real-valued output based on a weighted combination of its 960 inputs. These hidden unit outputs are then used as inputs to a second layer of 30 "output" units. Each output unit corresponds to a particular steering direction, and the output values of these units determine which steering direction is recommended most strongly.

5) Neural Network Architecture

An ANN consists of a number of artificial neurons connected among themselves in certain ways. Sometime these neurons are arranged in layers, with interconnections across the layers. The network may, or may not, be fully connected. Moreover, the nature of the interconnection paths also varies. They are either unidirectional, or bidirectional. The topology of an ANN, together with the nature of its interconnection paths, is generally referred to as its architecture.

Single Layer Feed Forward Artificial Neural Networks

Single layer feed forward is perhaps the simplest ANN architecture. It consists of an array of input neurons connected to an array of output neurons. Since the input neurons do not exercise any processing power, but simply forward the input signals to the subsequent neurons, they are not considered to constitute a layer. Hence, the only layer in the ANN is composed of the output neurons.

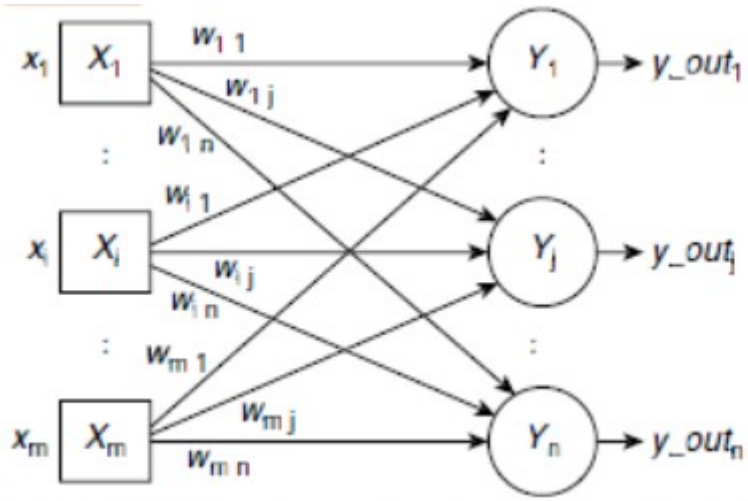


Fig. 6.25. Structure of a single-layer feed forward ANN

Multilayer Feed Forward Artificial Neural Networks

A multilayer feed forward net is similar to single layer feed forward net except that there is (are) one or more additional layer(s) of processing units between the input and the output layers. Such additional layers are called the hidden layers of the network.

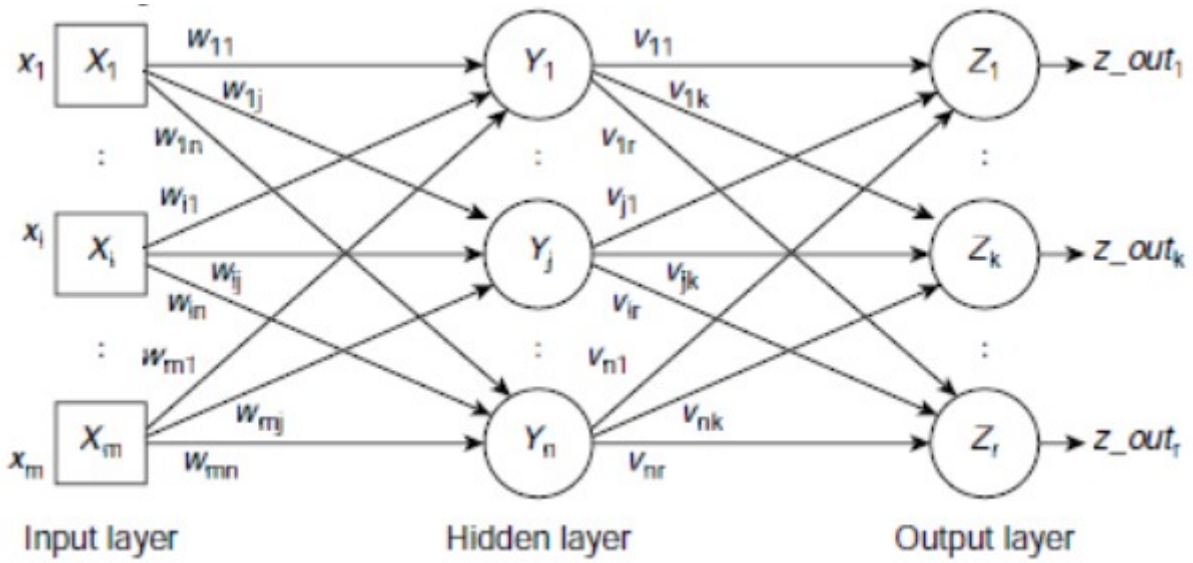


Fig. 6.26. A multi-layer feed forward network with one hidden layer

Recurrent Networks

In a feed forward network signals flow in one direction only and that is from the input layer towards the output layer through the hidden layers, if any. Such networks do not have any feedback loop. In contrast, a recurrent network allows feedback loops.

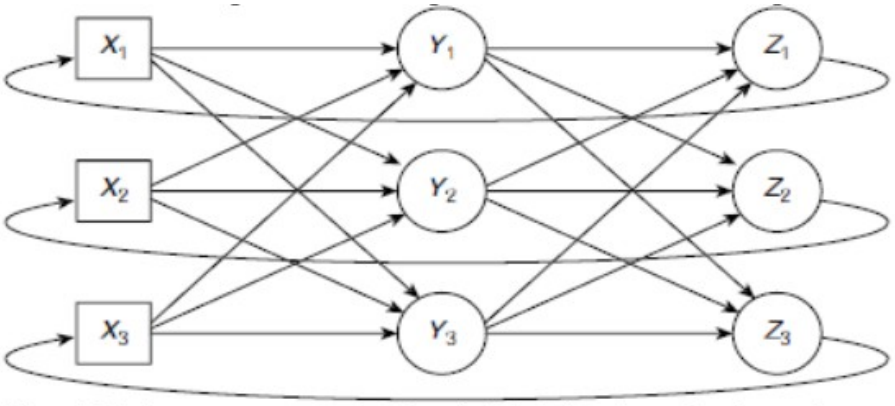


Fig. 6.28 A recurrent network with feedback paths from the output layer to the input layer

6) Activation Function

Activation functions main purpose is to convert a input signal of a node in a Artificial Neural Network to an output signal.

Sigmoid or Logistic
Tanh : Hyperbolic tangent
ReLu : Rectified linear units

An activation function decides whether the neuron fires (‘spikes’) for the current inputs.

For neural networks, g(·) is a mathematical function that describes the firing of the neuron as a response to the weighted inputs, such as the threshold function. Activation function decides whether or not to fire. Activation function that decides whether or not the node fires.

The sigmoidal activation function that we have created is aimed at making the nodes act a bit like neurons, either firing or not firing. This is very important in the hidden layer, but earlier in the chapter we have observed two cases where it is not suitable for the output neurons.

$$\textbf{Linear} \quad y_\kappa = g(h_\kappa) = h_\kappa$$

$$\textbf{Sigmoidal} \quad y_\kappa = g(h_\kappa) = 1/(1+\exp(-\beta h_\kappa)$$

$$\textbf{Soft-max} \quad y_\kappa = g(h_\kappa) = \exp(h_\kappa)/\sum_{k=1}^{N}\exp(h_k)$$

7) Mc Culloch Pitts Model

We're going to look at a mathematical model of a neuron that was introduced in 1943.

The purpose of a mathematical model is that it extracts only the bare essentials required to accurately represent the entity being studied, removing all of the extraneous details.

McCulloch and Pitts produced a perfect example of this when they modelled a neuron as :

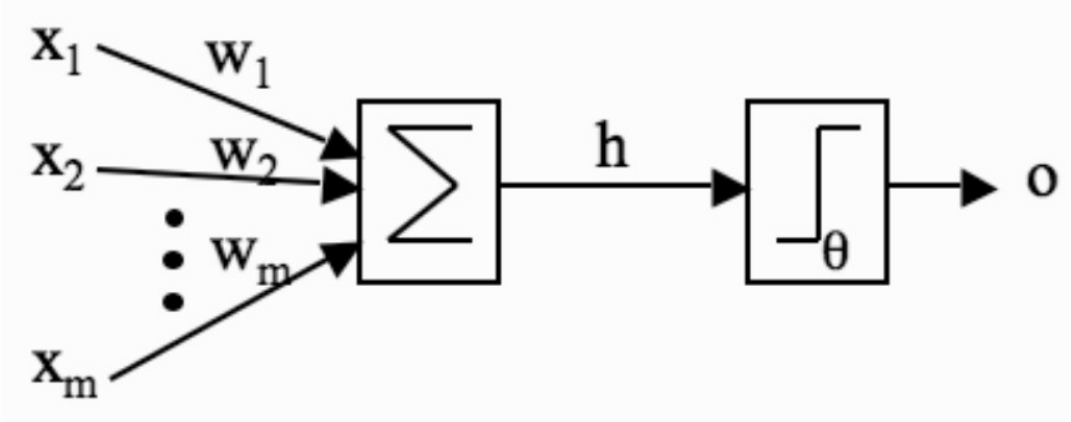


FIGURE 3.1 A picture of McCulloch and Pitts' mathematical model of a neuron. The inputs x_i are multiplied by the weights w_i, and the neurons sum their values. If this sum is greater than the threshold θ then the neuron fires; otherwise it does not.

(1) a set of weighted inputs w_i that correspond to the synapses

(2) an adder that sums the input signals (equivalent to the membrane of the cell that collects electrical charge)

(3) an activation function (initially a threshold function) that decides whether the neuron fires ('spikes') for the current inputs

Bibliography

1) “OpenCV with Python By Example” by Prateek Joshi
2) Tom M.Mitchell “Machine Learning” McGraw Hill
3) Stephen Marsland, “Machine Learning An Algorithmic Perspective” CRC Press
4) J.-S.R.Jang "Neuro-Fuzzy and Soft Computing" PHI 2003
5) Samir Roy and Chakraborty, “Introduction to soft computing”, Pearson Edition
6) Kevin P. Murphy , Machine Learning , “A Probabilistic Perspective”

Appendices

Installing Python

To follow all of the examples in this book, you're going to need Python 2.7, NumPy, and Matplotlib. The examples aren't guaranteed to work with Python 3.X, because Python doesn't provide backward compatibility. The easiest way to get these modules is through package installers. These are available on Mac OS and Linux.

NumPy

Having installed the NumPy library, you may be wondering, "What good is this?" Officially, NumPy is a matrix type for Python, and a large number of functions to operate on these matrices. Unofficially, it's a library that makes doing calculations easy and faster to execute, because the calculations are done in C rather than Python.

Despite the claim that it's a matrix library, there are actually two fundamental data types in NumPy: the array and the matrix. The operations on arrays and matrices are slightly different. If you're familiar with MATLAB TM, then the matrix will be most familiar to you. Both types allow you to remove looping operators you'd have to have using only Python.