Neural Networks (Lecture 2 & 3)

What is a Neural Network?

Is a machine that is designed to model the way in which the brain performs a particular task or function of interest, the network is usually implemented by using electronic components or is simulated in software on a digital computer.

Benefits of Neural Networks

1. **Nonlinearity:** An Artificial Neuron Can Be Linear Or Nonlinear. A Neural Network, made up of an interconnection of nonlinear neurons, is itself nonlinear
2. **Input–Output Mapping:** A popular paradigm of learning, called learning with a teacher.
3. **Adaptivity:** Neural networks have a built-in capability to adapt their synaptic(= width in FIGURE 1) weights to changes in the surrounding environment.
4. **Evidential Response:** In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to select, but also about the confidence in the decision made.
5. **Contextual Information:** Knowledge is represented by the very structure and activation state of a neural network.
6. **Fault Tolerance:** A neural network, implemented in hardware form, has the potential to be inherently fault tolerant, or capable of robust computation, in the sense that its performance degrades gracefully under adverse operating conditions.

MODELS OF A NEURON

A neuron is an information-processing unit that is fundamental to the operation of a neural network. The block diagram of Fig. 1 shows the model of a neuron.

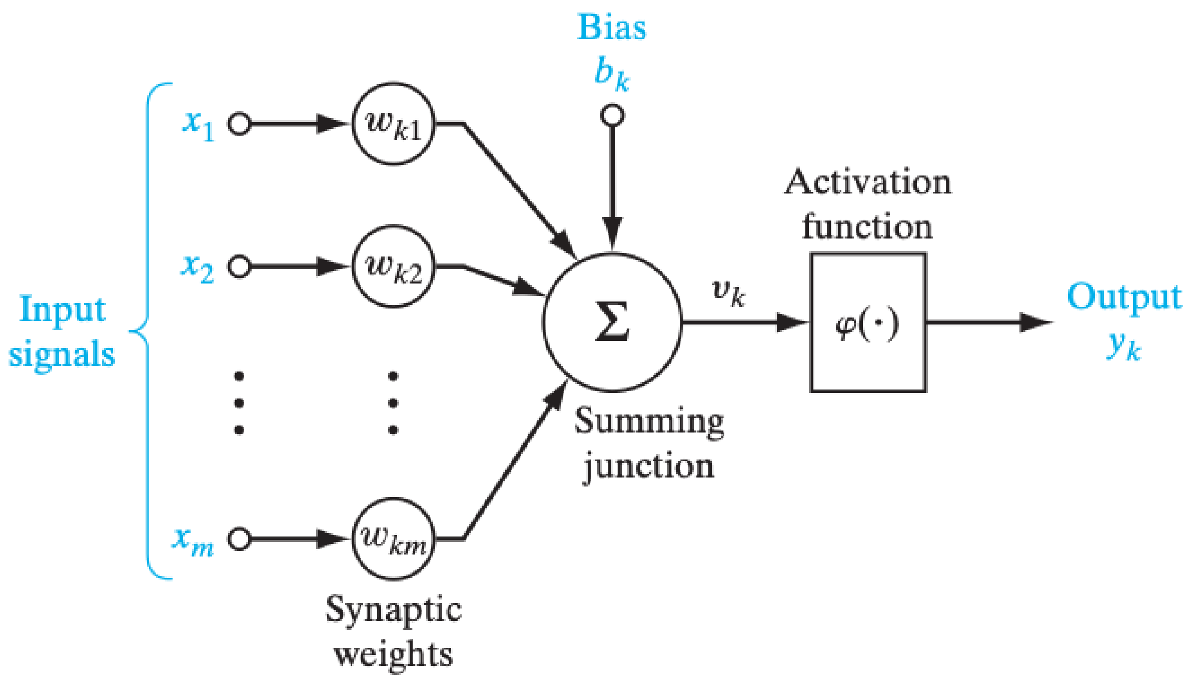


FIGURE 1

Nonlinear model of a neuron, labeled k

The basic unit in neural network is called **Perceptron**

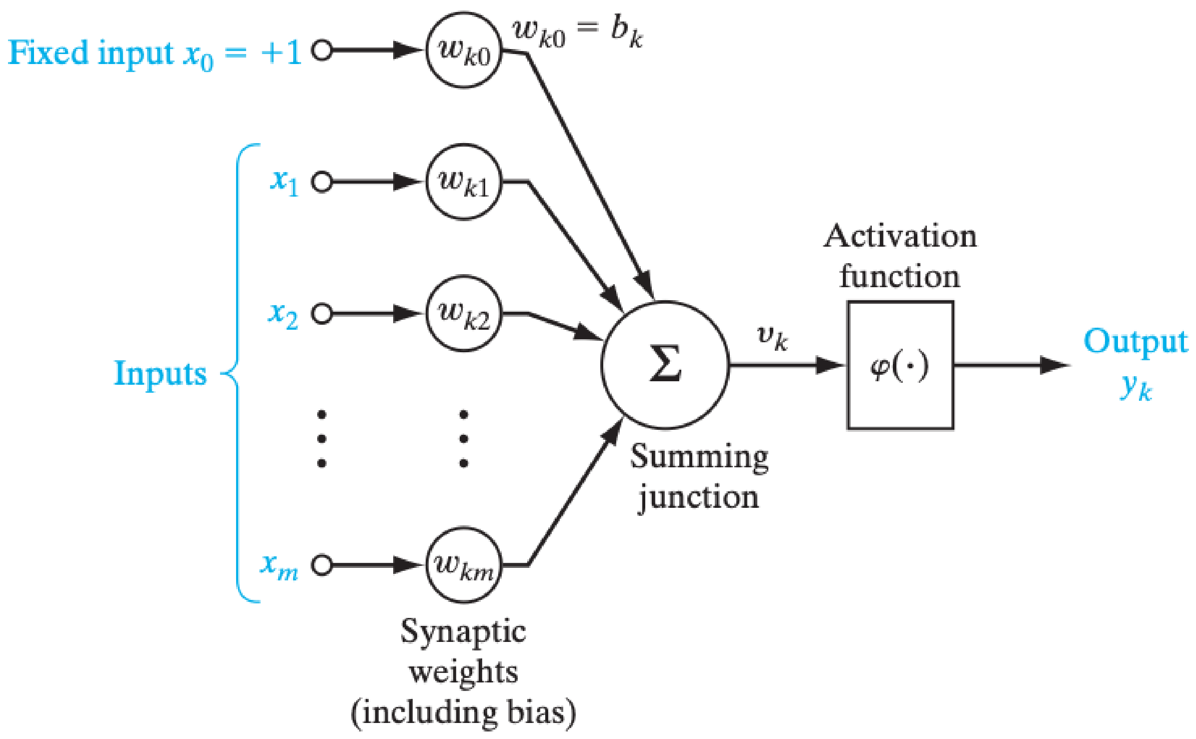


FIGURE 2

Another nonlinear model of a neuron; wk0 accounts for the bias bk.

Types of Activation Function ϕ

- $\phi(z) = Z\ linear$

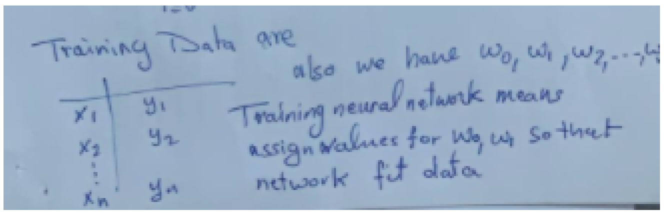
  $\phi(z)\ Threshold$

- $\phi(z)$ Sigmoid in non-linear case

$$v_k = \sum_{j=1}^{m} w_{kj}x_j + b_k$$

Let $w_0 = b,\ x_0 = 1$

$$v_k = \sum_{j=0}^{m} w_{kj}x_j$$



Training Data are

x_1	y_1
x_2	y_2
⋮	
x_n	y_n

also we have $w_0, w_1, w_2, \ldots,$

Training neural network means assign values for w_0, w_1 so that network fit data

Types of Neural Networks:

- Feedforward Networks
- Recurrent Neural Networks

Example

See perceptron algorithm

At each iteration we update weights we feed example 1 at time to network and how network performs on example we update the weight Suppose x_1 feed to network and we get on output Y and then pass through thresholding the output ≠1 or 0

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = \eta(y - \hat{y})x_i$$

Deep Feedforward Networks

Also often called feedforward neural networks, or multilayer perceptrons (MLPs), are the quintessential deep learning models.

The goal of a feedforward network

- to approximate some function f*.
  - ex, for a classifier, y = f*(x) maps an input x to a category y. A feedforward network defines a mapping y = f (x; θ) and

learns the value of the parameters θ that result in the best function approximation.

- Note -> θ = width (in FIGURE 1).

These models are called feedforward because information flows through the function being evaluated from x, through the intermediate computations used to define f , and finally to the output y.

There are no feedback connections in which outputs of the model are fed back into itself. When feedforward neural networks are extended to include feedback connections, they are called recurrent neural networks.

Feedforward neural networks are called networks because they are typically represented by composing together many different functions. The model is asso- ciated with a directed acyclic graph describing how the functions are composed together.

Ex, we might have three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain, to form f(x) = $f^{(3)}(f^{(2)}(f^{(1)}(x)))$. These chain structures are the most commonly used structures of neural networks. In this case, $f^{(1)}$ is called the first layer of the network, $f^{(2)}$ is called the second layer, and so on.

The overall length of the chain gives the **depth** of the model. It is from this terminology that the name “deep learning” arises. The final layer of a feedforward network is called the output layer.

During neural network training, we drive f(x) to match f*(x).

The training data provides us with noisy, approximate examples of f * (x) evaluated at different training points.

Each example x is accompanied by a label y ≈ f*(x). The training examples specify directly what the output layer must do at each point x; it must produce a value that is close to y.

The behavior of the other layers is not directly specified by the training data.

The learning algorithm must decide how to use those layers to produce the desired output, but the training data does not say what each individual layer should do.

Hidden layers: is located between the input and output of the algorithm, in which the function applies weights to the inputs and directs them through an activation function as the output.

These networks are called neural because they are loosely inspired by neuroscience.

Rather than thinking of the layer as representing a single vector-to-vector function, we can also think of the layer as consisting of many units that act in parallel, each representing a vector-to-scalar function.

It is best to think of feedforward networks as function approximation machines that are designed to achieve statistical generalization, occasionally drawing some insights from what we know about the brain, rather than as models of brain function.

Linear models, such as logistic regression and linear regression, are appealing because they may be fit efficiently and reliably, either in closed form or with convex optimization.

** To extend linear models to represent nonlinear functions of x, we can apply the linear model not to x itself but to a transformed input $\varphi(x)$, where φ is a nonlinear transformation.

We can think of φ as providing a set of features describing x, or as providing a new representation for x.

** Some design decisions as are necessary for a linear model:

1. Choosing the optimizer.
2. The cost function.
3. The form of the output units.

We must also design the architecture of the network, including how many layers the network should contain, how these layers should be connected to each other, and how many units should be in each layer.

Ex: Learning XOR

The XOR function provides the target function $y = f^*(x)$ that we want to learn. Our model provides a function $y = f(x;\theta)$ and our learning algorithm will adapt the parameters θ to make f as similar as possible to f^*.

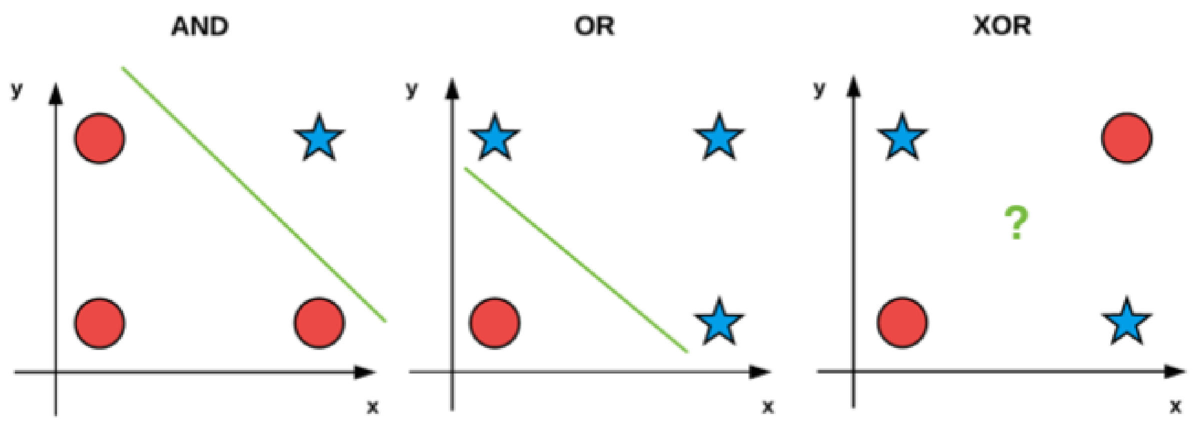


FIGURE 3

Both the AND and OR bitwise datasets are linearly separable, meaning that we can draw a *single line* (green) that separates the two classes. However, for XOR it is impossible to draw a single line that separates the two classes – this is therefore a nonlinearly separable dataset.

Perceptron Code:

https://github.com/Abdel-rahim/preceptron_test