

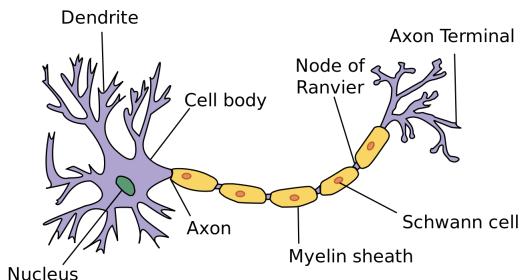
ECE/CS 559 - Neural Networks Lecture Notes #2

Mathematical models for the neuron, Neural network architectures

Erdem Koyuncu

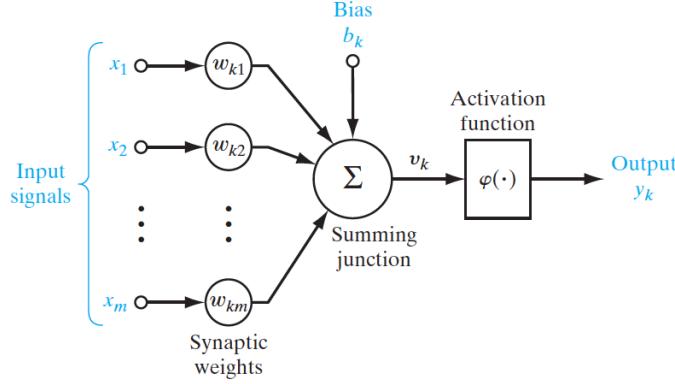
1 Mathematical models for the neuron

1.1 The biology of the neuron



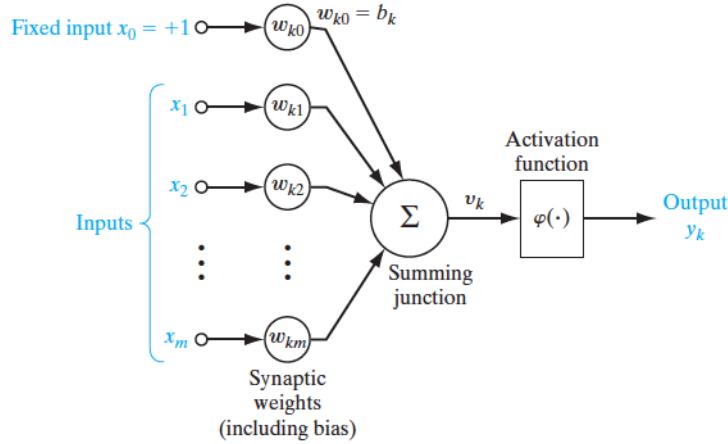
(Image taken from Wikipedia)

1.2 A mathematical model for the neuron



(Image taken from our coursebook: S. Haykin, "Neural Networks and Learning Machines," 3rd ed.)

- Input signals are from dendrites of other neurons.
- The synaptic weights correspond to the synaptic strengths: positivity/negativity \rightarrow excitation/inhibition.
- The summing unit models the operation of the cell body (soma).
- The nonlinearity $\varphi(\cdot)$ (activation function) models the axon operation.
- The output may be connected to dendrite of another neuron through another synapse.
- $v_k = \sum_{j=1}^n w_{kj}x_j + b_k$ is called the **induced local field** of neuron k .
- $y_k = \varphi(v_k) = \varphi\left(\sum_{j=1}^n w_{kj}x_j + b_k\right)$.
- Alternatively, we may consider a fixed input $x_0 = 1$ with weight $w_{k0} = b_k$:



(Image taken from our coursebook: S. Haykin, "Neural Networks and Learning Machines," 3rd ed.)

- $y_k = \varphi\left(\sum_{j=0}^n w_{kj}x_j\right)$. Note that now the summation starts from index 0.

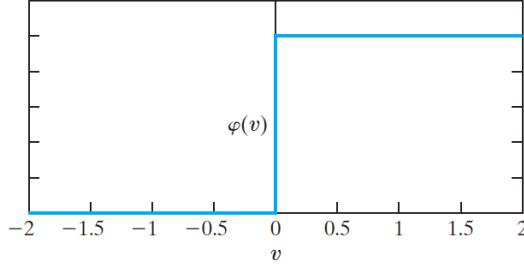
1.3 Types of activation function

Typically, $\varphi(\cdot)$ has bounded image (e.g. $[0, 1]$ or $[-1, 1]$), and thus is also called a squashing function. It limits the amplitude range of the neuron output.

1.3.1 Step function

- Threshold function (or the Heaviside/step function):

$$\varphi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases} .$$



(Image taken from our coursebook: S. Haykin, “Neural Networks and Learning Machines,” 3rd ed.)

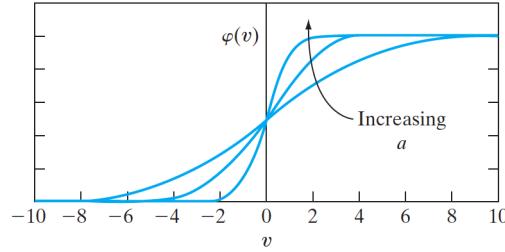
Such a neuron is referred to as the McCulloch-Pitts model (1943).

1.3.2 Sigmoid function

- The sigmoid function is defined as

$$\varphi(v) = \frac{1}{1 + \exp(-av)},$$

where a is called the slope parameter.



(Image taken from our coursebook: S. Haykin, “Neural Networks and Learning Machines,” 3rd ed.)

- As $a \rightarrow \infty$ the sigmoid function approaches the step function.
- Unlike the step function, the sigmoid function is continuous and differentiable. Differentiability turns out to be a desirable property of an activation function, as we shall see later.

1.3.3 Signum function

$$\varphi(v) = \text{sgn}(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v = 0 \\ -1, & v < 0 \end{cases} .$$

1.3.4 Hyperbolic tangent

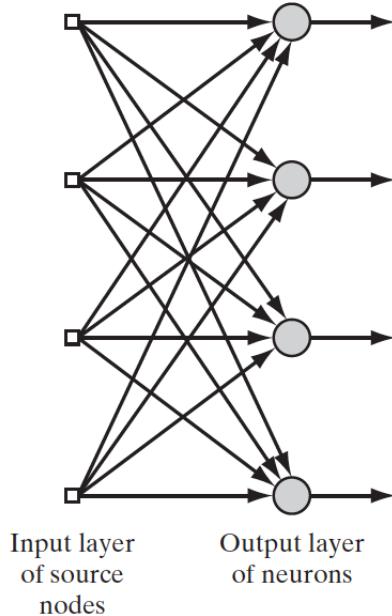
$$\varphi(v) = \tanh(av) = \frac{e^{av} - e^{-av}}{e^{av} + e^{-av}}.$$

for some parameter $a > 0$. Approaches the signum function as $a \rightarrow \infty$.

2 Neural network architectures

Having introduced our basic model of a (mathematical) neuron, we now introduce the different neural network architectures that we will keep revisiting throughout the course.

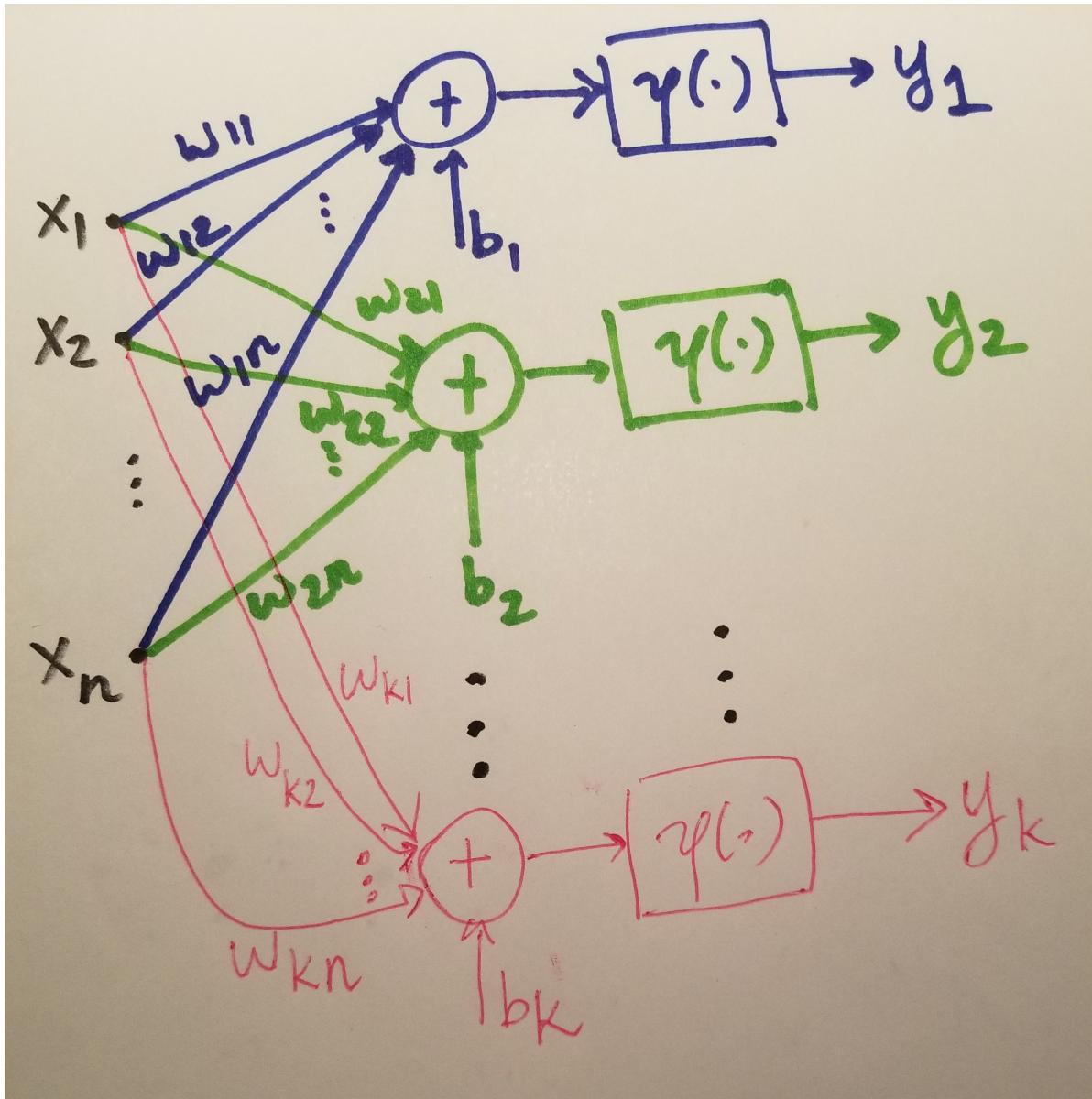
2.1 Single-layer feedforward networks



(Image taken from our coursebook: S. Haykin, “Neural Networks and Learning Machines,” 3rd ed.)

Figure 1: Feedforward network with a single layer of neurons

- We just count the number of layers consisting of neurons (not the layer of source nodes as no computation is performed there). Thus, the network in Fig. 1 is called a single-layer network.
- Also, note that the information flow is only over one direction, i.e. the input layer of source nodes project directly onto the output layer of neurons (according to the non-linear transformations as specified by the neurons.). There is no **feedback** of the network’s output to network’s input. We thus say that the network in Fig. 1 is of feedforward type.
- Let us try to formulate the input-output relationship of the network. Putting in the symbols, we have the following diagram:



We have

$$y_1 = \phi \left(b_1 + \sum_{i=1}^n w_{1i} x_i \right)$$

$$y_2 = \phi \left(b_2 + \sum_{i=1}^n w_{2i} x_i \right)$$

⋮

$$y_k = \phi \left(b_k + \sum_{i=1}^n w_{ki} x_i \right)$$

We can rewrite all k equations via a single equation:

$$y_j = \phi \left(b_j + \sum_{i=1}^n w_{ji} x_i \right), j = 1, \dots, k$$

Here w_{ji} is the weight from input i to Neuron j . We can further rewrite everything in a simple matrix form. Define

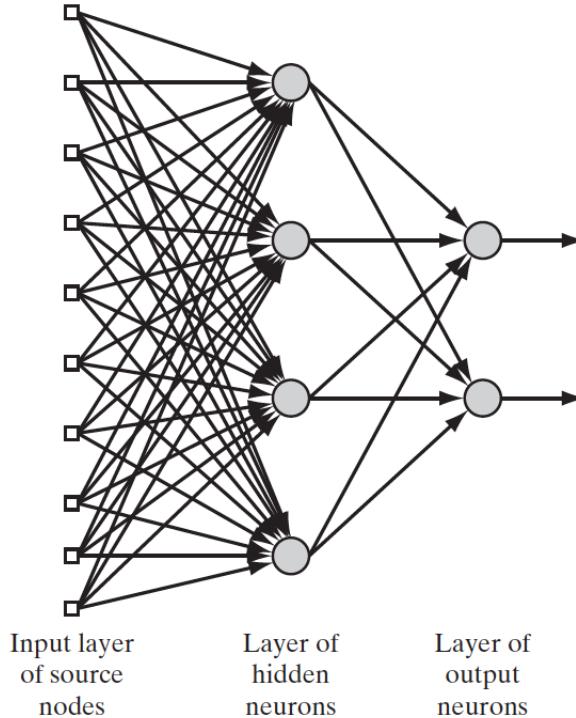
$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}, \mathbf{W}' = \begin{bmatrix} b_1 & w_{11} & \cdots & w_{1n} \\ b_2 & w_{21} & \cdots & w_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_k & w_{k1} & \cdots & w_{kn} \end{bmatrix}, \mathbf{x}' = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Then, the above input output relationship can just be written as $\mathbf{y} = \phi(\mathbf{W}'\mathbf{x}')$ in the sense that ϕ is applied component-wise. Sometimes biases are treated separately. Defining

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}, \mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ w_{21} & \cdots & w_{2n} \\ \vdots & \vdots & \vdots \\ w_{k1} & \cdots & w_{kn} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

we can write $\mathbf{y} = \phi(\mathbf{W}'\mathbf{x}') = \phi(\mathbf{W}\mathbf{x} + \mathbf{b})$.

2.2 Multilayer feedforward networks



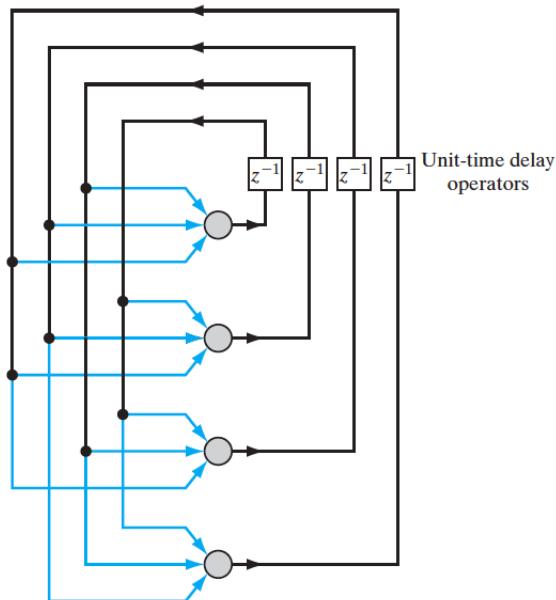
(Image taken from our coursebook: S. Haykin, “Neural Networks and Learning Machines,” 3rd ed.)

Figure 2: Fully-connected 2-layer feedforward network with one hidden layer and one output layer.

- We can add more layers to the feedforward network in Fig. 1.
- The end-result is a multilayer network with one or more hidden layers.
- Hidden layers refer to those layers that are not seen directly from either the input or the output of the network.

- We call a network with m source nodes in its input layer, h_1 hidden nodes in its first hidden layer, h_2 nodes in its second hidden layer, \dots , h_K nodes in its K th hidden layer, and finally q nodes in its output layer a $m \times h_1 \times h_2 \times \dots \times h_K \times q$ network. For example, the network in Fig. 2 is called a 10-4-2 network as it has 10 source nodes in its input layer, 4 nodes in the hidden layer, and 2 nodes in its output layer.
- Fully-connected network: Every node in each layer of the network is connected to every other node in the adjacent layer. Example: The network in Fig. 2 is fully connected. Otherwise, the network is partially connected.
- Deep network: Many (usually assumed to be > 1) hidden layers. Shallow network: The opposite.
- As will be made more precise later on, theoretically, only one hidden node is sufficient for almost any application provided that one can afford a large number of neurons. On the other hand, a deep network can perform the same tasks as a shallow network with the extra advantage of possibly a fewer number of neurons. Hence, deep networks, provided that they can be properly designed, may be better suited for complex practical applications.
- The input-output relationships may be formulated in a similar manner as the single-layer network discussed previously. For example, consider a two-layer network with n inputs, L_1 neurons in the first layer, and L_2 neurons in the second (output) layer. Let $\mathbf{x} \in \mathbb{R}^{n \times 1}$ be the vector of inputs, $\mathbf{W}_1 \in \mathbb{R}^{L_1 \times n}$ be the matrix of weights connecting the input layer to the first layer of neurons (where the i th row j th column of \mathbf{W}_1 corresponds to the weight between input node j and neuron i of the first layer), $\mathbf{b}_1 \in \mathbb{R}^{L_1 \times 1}$ be the vector of biases for the first layer of neurons, $\mathbf{W}_2 \in \mathbb{R}^{L_2 \times L_1}$ be the matrix of weights connecting the first layer of neurons to the second layer of neurons (where the i th row j th column of \mathbf{W}_2 corresponds to the weight between neuron j of the first layer and neuron i of the second layer), and $\mathbf{b}_2 \in \mathbb{R}^{L_2 \times 1}$ be the vector of biases for the second layer of neurons, and $\mathbf{y} \in \mathbb{R}^{L_2 \times 1}$ be the vector of outputs. Then, we have $\mathbf{y} = \phi(\mathbf{W}_2\phi(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)$.

2.3 Recurrent networks



(Image taken from our coursebook: S. Haykin, “Neural Networks and Learning Machines,” 3rd ed.)

Figure 3: Recurrent network with no self-feedback loops and no hidden neurons.

- What distinguishes recurrent networks from feedforward networks is that they incorporate **feedback**.
- In Fig. 3, the boxes with z^{-1} represent the unit discrete time delays.
- No self-feedback: The output of a given neuron is not fed back to its own input.
- One may have variations of the structure in Fig. 3. We shall discuss these variations and the details of recurrent networks later on.