CMPS 287: Artificial Intelligence **Fall 2016**

Lecture 3: Neural Networks (Representation)

Lecturer: Prof. Haitham Bou Ammar *Scribes: Haitham Bou Ammar*

Abstract

Neural networks are rich forms of function approximators that can be used to perform either classification or regression. Due to their rich architectures, neural networks have been shown to outperform state-of-the-art methods in a variety of applications including image and speech recognition. Their key capabilities are in their ability of autonomously discovering features useful to solve a certain problem. These notes will cover neural nets and detail their mathematical derivations. The notes are split in two. Here, you will learn how to represent neural network functions, while in the second you will understand their optimization peculiarities.

3.1 Neural Network Representation

Neural Networks (NNs) were originally motivated by looking at machines which replicate the brain's functionality. In an artificial neural network, a neurone is a logistic unit, which is fed inputs through input wires. This unit can perform computations resulting in outputs that are transmitted through the output wires.

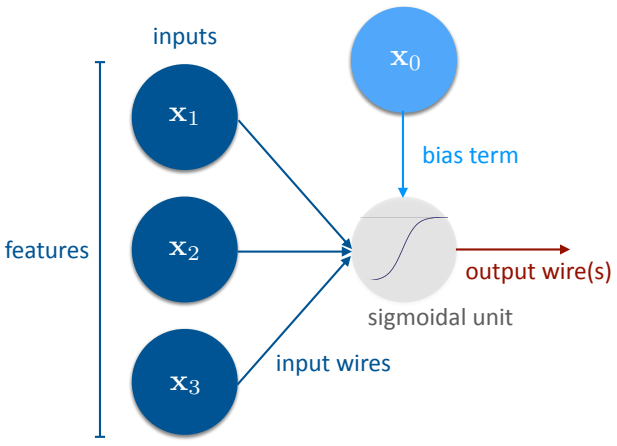

Figure 3.1: A high-level depiction of a logistic unit showing the: 1) inputs with their input wires, 2) nonlinear sigmoidal transformation, and 3) the output wires.

An illustration of a single neurone is shown in Figure 3.1 for a three dimensional input vector $\boldsymbol{x} = [x_1, x_2, x_3]^\mathsf{T}$, and an appended bias term x_0. In the logistic unit, the input vector is first linearly combined using a weight vector, $\boldsymbol{\theta}$, and then nonlinearly transformed using a sigmoidal function. Consequently, the output delivered on the output wire can be written as:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}}} = \frac{1}{1 + e^{-\sum_{i=0}^{3}\theta_i x_i}},$$

with $x_0 = 1$ denoting the bias term, and $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \theta_3]^\mathsf{T}$ representing the weights connecting the inputs.

An artificial neural network is simply a set of these logistic units strung together as shown in Figure 3.1. Each two layers are connected together using weight parameters. As such, the neural network in Figure 3.1 possesses two weighting matrices, $\boldsymbol{\Theta}^{(1)}$ and $\boldsymbol{\Theta}^{(2)}$. Here, we used $\boldsymbol{\Theta}^{(l)}$ to denote the weights connecting layers l and $l+1$. Definitely, the dimensionality of $\boldsymbol{\Theta}^{(l)}$ depends on the number of units in each of the two layers. For example, if layer l consists of s_l units and layer $l+1$ of s_{l+1}, $\boldsymbol{\Theta}^{(l)}$ has a dimensionality of $s_{l+1} \times s_l + 1$, where we have added another dimension to incorporate the bias term.

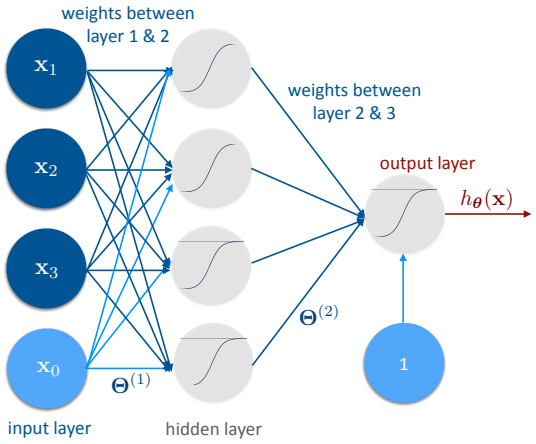

Figure 3.2: A high-level depiction of an artificial neural network showing three layers as well as the weight connections between them.

To better understand the above notation, let us reconsider the example of Figure 3.1. Specifically, let us illustrate the dimensionality and meaning of $\boldsymbol{\Theta}^{(1)}$, which connects layers one and two (i.e., input to hidden layer connections). It is easy to see that the input layer consists of three units and a bias, while the hidden layer of four sigmoidal units. Hence, the dimensionality of $\boldsymbol{\Theta}^{(1)}$ is given by $4 \times (3+1)$. In other words, the row-count of $\boldsymbol{\Theta}$ is given by the number of units in the successor layer (i.e., layer $l+1$), while the column-count by the number of units in the current layer (i.e., l) appended by an additional dimension for the bias term:

$$\boldsymbol{\Theta}^{(1)} = \begin{bmatrix} \Theta^{(1)}_{10} & \Theta^{(1)}_{11} & \Theta^{(1)}_{12} & \Theta^{(1)}_{13} \\ \Theta^{(1)}_{20} & \Theta^{(1)}_{21} & \Theta^{(1)}_{22} & \Theta^{(1)}_{23} \\ \Theta^{(1)}_{30} & \Theta^{(1)}_{31} & \Theta^{(1)}_{32} & \Theta^{(1)}_{33} \\ \Theta^{(1)}_{40} & \Theta^{(1)}_{41} & \Theta^{(1)}_{42} & \Theta^{(1)}_{43} \end{bmatrix}.$$

If we consider row one in $\boldsymbol{\Theta}^{(1)}$, for example, we come to recognize that it corresponds to representing the connections between all the nodes from the input layer (i.e., $l = 1$) to the *first* node in the hidden (i.e., when $l = 2$). Hence, $\boldsymbol{\Theta}^{(l)}_{ij}$ denotes the connecting weight initiating from node j in layer l to node i in layer $l+1$, see Figure 3.1 for a pictorial illustration.

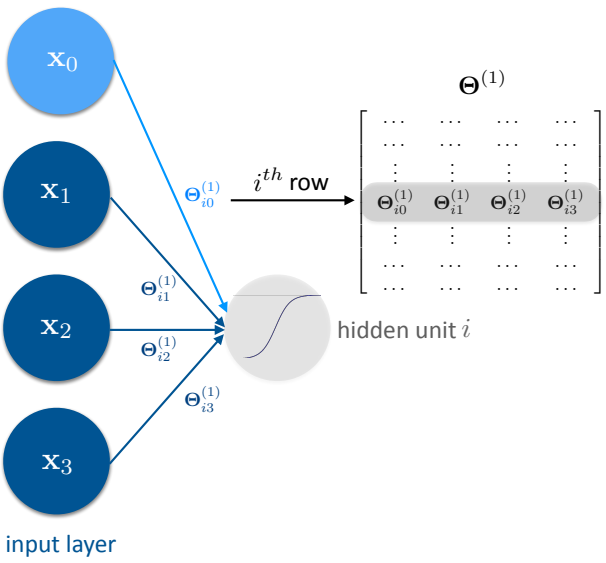

Figure 3.3: An illustration of the weights connecting the units in the input layer to unit i in the hidden layer.

3.1.1 Feed Forward Propagation

Given the notation introduced above, we are now ready to discuss the computations that are performed by a neural network. Intuitively, between every two layers the inputs from the previous layer are, first, linearly (through the weight matrices) propagated forward and then nonlinearly transformed (through the sigmoids) to produce an output on the successor layer. Recursing this process, which we refer to as forward propagation, over the total layers of the network will produce an output on the final layer L. In what comes next, we will present an efficient vectorized implementation of forward propagation.

We call the computation performed by node i in layer l an activation and denote it by $a_i^{(l)} = \text{sigmoid}\left(\sum_{j=0}^{s_{l-1}} \mathbf{\Theta}_{ij}^{(l-1)} a_j^{(l-1)}\right)$, with $a_0^{(l-1)} = 1$, and $a_j^{(l-1)}$ for $j \in \{1, \ldots, s_{l-1}\}$ being the activations of the previous layer provided that the activations of the input (or first layer) are the data points themselves. Clearly, $a_i^{(l)}$ can be written in a matrix-vector product form with the help of $\mathbf{\Theta}^{(l-1)}$. Essentially, the input to the sigmoidal function is a linear combination between the i^{th} row of the weight matrix and the activations on the previous layer:

$$a_i^{(l)} = \text{sigmoid}\left(\mathbf{\Theta}_{i,:}^{(l-1),\mathsf{T}} \boldsymbol{a}^{(l-1)}\right) = \frac{1}{1 + e^{-\mathbf{\Theta}_{i,:}^{(l-1),\mathsf{T}} \boldsymbol{a}^{(l-1)}}},$$

where $\mathbf{\Theta}_{i,:}^{(l-1),\mathsf{T}}$ denotes the i^{th} row of $\mathbf{\Theta}^{(l-1)}$, and $\boldsymbol{a}^{(l-1)}$ is a vector collecting all activations (including that of the bias term) of layer $l-1$, i.e., $\boldsymbol{a}^{(l-1)} = \left[1, a_1^{(l-1)}, a_2^{(l-1)}, \ldots, a_{s_{l-1}}^{(l-1)}\right]^{\mathsf{T}}$. Now, we can easily generalize

the above notion to compute the activations in layer l by considering the whole matrix $\boldsymbol{\Theta}^{(l-1)}$:

$$
\begin{aligned}
\boldsymbol{a}^{(l)} &= \text{sigmoid}\left(\boldsymbol{\Theta}^{(l-1)}\boldsymbol{a}^{(l-1)}\right) \\
&= \text{sigmoid}\left[\begin{pmatrix} \boldsymbol{\Theta}_{10}^{(l-1)} & \boldsymbol{\Theta}_{12}^{(l-1)} & \dots & \boldsymbol{\Theta}_{1s_{l-1}}^{(l-1)} \\ \boldsymbol{\Theta}_{20}^{(l-1)} & \boldsymbol{\Theta}_{21}^{(l-1)} & \dots & \boldsymbol{\Theta}_{2s_{l-1}}^{(l-1)} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{\Theta}_{s_{l+1}0}^{(l-1)} & \boldsymbol{\Theta}_{s_{l+1}1}^{(l-1)} & \dots & \boldsymbol{\Theta}_{s_{l+1}s_{l-1}}^{(l-1)} \end{pmatrix}\begin{bmatrix} 1 \\ a_1^{(l-1)} \\ \vdots \\ a_{s_{l-1}}^{(l-1)} \end{bmatrix}\right] \\
&= \begin{bmatrix} \text{sigmoid}\left(\boldsymbol{\Theta}_{1,:}^{(l-1),\mathsf{T}}\boldsymbol{a}^{(l-1)}\right) \\ \text{sigmoid}\left(\boldsymbol{\Theta}_{2,:}^{(l-1),\mathsf{T}}\boldsymbol{a}^{(l-1)}\right) \\ \vdots \\ \text{sigmoid}\left(\boldsymbol{\Theta}_{s_{l+1},:}^{(l-1),\mathsf{T}}\boldsymbol{a}^{(l-1)}\right) \end{bmatrix}.
\end{aligned}
$$

To illustrate, consider the problem of performing classification that differentiates between four (or K) classes. To do so, we are going to change the architecture of the network to produce a four-dimensional output. This can be achieved by constructing a network with four output units as depicted in Figure 3.4. Here, we also chose to introduce an additional hidden layer hoping to learn more descriptive features.

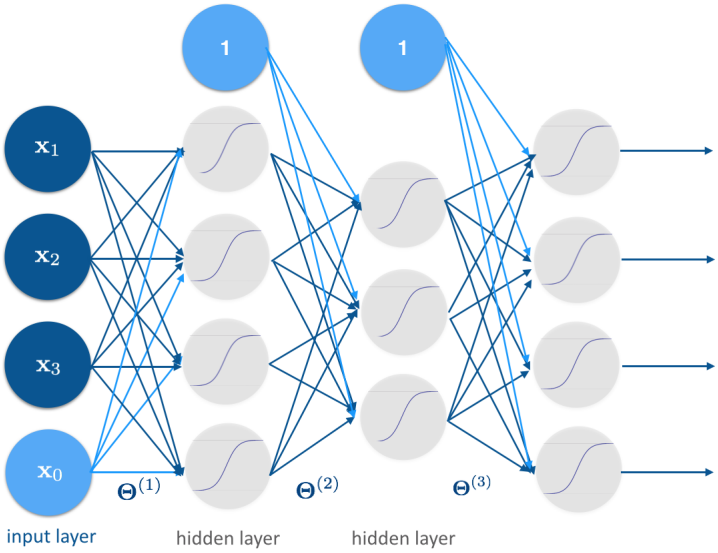

Figure 3.4: A high-level depiction of an artificial neural network showing four layers as well as the weight connections between them.

This network has an input, 2 hidden, and an output layer. Given that we have 4 layers in total, we need three weight matrices (i.e., $\boldsymbol{\Theta}^{(1)}$, $\boldsymbol{\Theta}^{(2)}$, and $\boldsymbol{\Theta}^{(3)}$) to denote the connections among them. The dimensions of these weights can be determined as follows:

$$\boldsymbol{\Theta}^{(1)} \in \mathbb{R}^{4\times 4}, \qquad \boldsymbol{\Theta}^{(2)} \in \mathbb{R}^{3\times 5}, \qquad \boldsymbol{\Theta}^{(3)} \in \mathbb{R}^{4\times 4}.$$

As detailed previously, during feed-forward propagation, our goal is to determine the activations on each of the layers. The activations of the input layer are set to the input data themselves in addition to a bias term.

That is:

$$\boldsymbol{a}^{(1)} = \begin{bmatrix} 1 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \end{bmatrix}.$$

Given, $\boldsymbol{\Theta}^{(1)}$, we can now compute an intermediate vector $\boldsymbol{z}^{(2)}$, which will be used to determine the activations on the second layer (i.e., first hidden layer) that are denoted by $\boldsymbol{a}^{(2)}$:

$$\boldsymbol{z}^{(2)} = \underbrace{\boldsymbol{\Theta}^{(1)}}_{\in \mathbb{R}^{4\times 4}} \overbrace{\boldsymbol{a}^{(1)}}^{\in \mathbb{R}^{4\times 1}} \qquad \implies \boldsymbol{a}^{(2)} = \text{sigmoid}(\boldsymbol{z}^{(2)}) = \text{sigmoid}\left(\boldsymbol{\Theta}^{(1)}\boldsymbol{a}^{(1)}\right).$$

Given, $\boldsymbol{a}^{(2)}$, we can continue our recursion to determine the activations on each of the successor layers. Note, however, that at each layer we need to append an additional dimension for the bias term. In other words, create $\boldsymbol{a}^{(2)} = [1, a_1^{(2)}, a_2^{(2)}, a_3^{(2)}, a_4^{(2)}]^\mathsf{T}$, and the perform:

$$\boldsymbol{z}^{(3)} = \underbrace{\boldsymbol{\Theta}^{(2)}}_{\in \mathbb{R}^{3\times 5}} \overbrace{\boldsymbol{a}^{(2)}}^{\in \mathbb{R}^{5\times 1}} \qquad \implies \boldsymbol{a}^{(3)} = \text{sigmoid}(\boldsymbol{z}^{(3)})$$

$$\boldsymbol{z}^{(4)} = \underbrace{\boldsymbol{\Theta}^{(3)}}_{\in \mathbb{R}^{4\times 4}} \overbrace{\boldsymbol{a}^{(3)}}^{\in \mathbb{R}^{4\times 1}} \qquad \implies \boldsymbol{a}^{(3)} = \text{sigmoid}(\boldsymbol{z}^{(3)}) \quad \text{(we already appended 1 to } \boldsymbol{a}^{(3)}\text{)}.$$