Notes on Neural Networks

Marcio Woitek

May 25, 2020

Model Representation

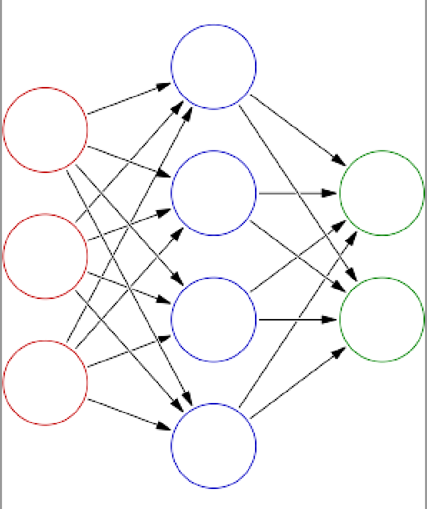

Figure: Basic Structure of a Neural Network

Model Representation

Fundamental Ideas

- A neural network is formed by nodes or *units*.
- The units are organized in *layers*.
- There are at least two layers, the *input layer* and the *output layer*.
- However, very often, between these layers there are *hidden layers*.
- In Fig. 1, the red nodes form the input layer, the blue nodes form the hidden layer, and the green nodes form the output layer.
- For simplicity, in the next slides, we discuss the particular case of the neural network in this figure.

Model Representation

Underlying Mathematical Concepts

- First, consider the input layer.
- We assume there are n_1 input units. In the case of Fig. 1, we have $n_1 = 3$.
- We denote the input values by $x_1, x_2, \ldots, x_{n_1}$.
- There can also be an extra unit, the so-called *bias unit*. Its value is represented by x_0. We shall set every bias unit value equal to 1.
- It is convenient to organize all these values in a single $(n_1 + 1)$-dimensional column vector:

$$\mathbf{a}^{(1)} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n_1} \end{bmatrix}. \tag{1}$$

Model Representation

Underlying Mathematical Concepts

- We can use a similar idea to represent the values related to the other layers.
- To be more precise, we assume the neural network has L layers.
- Then the values associated with the j-th layer will be organized in a column vector $\mathbf{a}^{(j)}$ ($j = 1, \ldots, L$).
- For $j \neq L$, the vector $\mathbf{a}^{(j)}$ is $(n_j + 1)$-dimensional, where n_j denotes the number of units in the j-th layer.
- This vector has an extra dimension because the value associated with the bias unit of this layer is also included in $\mathbf{a}^{(j)}$.
- However, the output layer doesn't have a bias unit. For this reason, the vector $\mathbf{a}^{(L)}$ is n_L-dimensional.

Example of Fig. 1

- As an example, we discuss the neural network in Fig. 1.
- Clearly, this network has $L = 3$ layers.
- Then the unit values are organized in 3 vectors, $\mathbf{x} = \mathbf{a}^{(1)}$, $\mathbf{a}^{(2)}$ and $\mathbf{y} = \mathbf{a}^{(3)}$.
- From Fig. 1, we can see that the numbers of units are given by $n_1 = 3$, $n_2 = 4$ and $n_3 = 2$.
- Therefore, taking into account the values of the bias units, we can state the following: $\mathbf{a}^{(1)} \in \mathbb{R}^4$, $\mathbf{a}^{(2)} \in \mathbb{R}^5$ and $\mathbf{a}^{(3)} \in \mathbb{R}^2$.

Forward Propagation

- Next, we explain how the input values are propagated in the network.
- Essentially, we start from the input vector $\mathbf{x} = \mathbf{a}^{(1)}$, and map $\mathbf{a}^{(j)}$ to $\mathbf{a}^{(j+1)}$ until we obtain the output vector $\mathbf{y} = \mathbf{a}^{(L)}$.
- In our example, the idea is the following: first, the input vector $\mathbf{x} = \mathbf{a}^{(1)}$ is mapped to $\mathbf{a}^{(2)}$, and then this vector is mapped to $\mathbf{a}^{(3)}$, producing the output $\mathbf{y}$.
- To be specific about how these mappings work, we introduce 2 vectors, $\mathbf{z}^{(2)}$ and $\mathbf{z}^{(3)}$.
- In general, we introduce $L - 1$ vectors $\mathbf{z}^{(j)}$, where $j = 2, \ldots, L$.
- We also define a set with $L - 1$ functions $g^{(j)}$, where $j = 2, \ldots, L$. The function $g^{(j)}$ is called the *activation function* of the j-th layer. Notice that there isn't an activation function associated with the input layer.
- To discuss our example, we consider the particular case in which all activation functions are the same. We denote this function by h.

Forward Propagation

Activation Function

- For the activation functions, there are many choices.
- To discuss our example, we consider the particular case in which h is the *sigmoid* or *logistic function*:

$$h(x) = \frac{1}{1+\exp(-x)}. \tag{2}$$

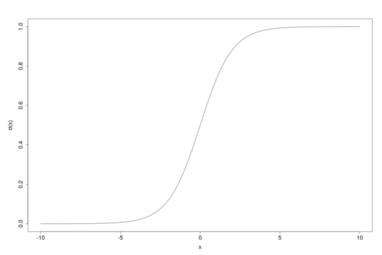

Figure: Graph of the sigmoid function.

Forward Propagation

Forward Propagation Equations

- We continue by explaining how the vector $\mathbf{z}^{(j)}$ is computed from the unit values in $\mathbf{a}^{(j-1)}$.
- We imagine all units in the $(j-1)$-th layer (including the bias unit) are connected to every unit in the j-th layer (excluding the bias unit).
- Then between these layers there are $(n_{j-1}+1)\,n_j$ connections.
- Each of these connections has a different *weight*. We will organize all these weights in a matrix denoted by $\Theta^{(j-1)}$.
- The element $\Theta^{(j-1)}_{ik}$ in the i-th row and k-th column of this matrix is interpreted as follows: it is the weight of the connection between the k-th unit in the $(j-1)$-th layer and the i-th unit in the j-th layer.
- Then the dimension of the matrix $\Theta^{(j-1)}$ is $n_j \times (n_{j-1}+1)$.

Forward Propagation

Forward Propagation Equations

- Finally, we can write down the equations that determine the vector $\mathbf{z}^{(j)}$.
- The i-th component of this vector is given by the *weighted sum* of the components of $\mathbf{a}^{(j-1)}$:

$$\begin{aligned} z_i^{(j)} &= \Theta_{i0}^{(j-1)} a_0^{(j-1)} + \Theta_{i1}^{(j-1)} a_1^{(j-1)} + \ldots + \Theta_{i,n_{j-1}}^{(j-1)} a_{n_{j-1}}^{(j-1)} \\ &= \sum_{k=0}^{n_{j-1}} \Theta_{ik}^{(j-1)} a_k^{(j-1)}. \end{aligned} \tag{3}$$

- The last expression can be written in matrix form as follows:

$$\mathbf{z}^{(j)} = \Theta^{(j-1)} \mathbf{a}^{(j-1)}. \tag{4}$$

- Notice that the vector $\mathbf{z}^{(j)}$ is n_j-dimensional.

Forward Propagation Equations

- For $j \neq L$, to obtain the vector $\mathbf{a}^{(j)}$, we only need to apply the activation function $g^{(j)}$ to every component of $\mathbf{z}^{(j)}$, and add the bias unit of the j-th layer:

$$\mathbf{a}^{(j)} = \begin{bmatrix} a_0^{(j)} \\ g^{(j)}\left(z_1^{(j)}\right) \\ \vdots \\ g^{(j)}\left(z_{n_j}^{(j)}\right) \end{bmatrix}, \tag{5}$$

  where $a_0^{(j)} = 1$. Clearly, the above vector is $(n_j + 1)$-dimensional.

- Equivalently, we can write

$$\mathbf{a}^{(j)} = \begin{bmatrix} a_0^{(j)} \\ g^{(j)}\left(\mathbf{z}^{(j)}\right) \end{bmatrix} = \begin{bmatrix} 1 \\ g^{(j)}\left(\Theta^{(j-1)}\mathbf{a}^{(j-1)}\right) \end{bmatrix}. \tag{6}$$

Forward Propagation

Forward Propagation Equations

- As explained, we treat the vector $\mathbf{a}^{(L)}$ differently.
- Recall that, in this case, it isn't necessary to add the value of the bias unit.
- Therefore, to compute $\mathbf{a}^{(L)}$, we only need to apply the activation function $g^{(L)}$ to every component of the vector $\mathbf{z}^{(L)}$:

$$\mathbf{a}^{(L)} = g^{(L)}\left(\mathbf{z}^{(L)}\right) = g^{(L)}\left(\Theta^{(L-1)}\mathbf{a}^{(L-1)}\right). \tag{7}$$

- Notice that this vector is n_L-dimensional.

Forward Propagation

Neural Network as a Composition of Mappings

- Then we have found the mappings we were looking for.
- We can propagate the input values by using $L-1$ functions.
- These functions are denoted by $f^{(j)}$, where $j = 2, \ldots, L$.
- First, we present their definition for $j \neq L$.
- In this case, to obtain $\mathbf{a}^{(j)}$ from $\mathbf{a}^{(j-1)}$, we apply the function $f^{(j)} : \mathbb{R}^{n_{j-1}+1} \to \mathbb{R}^{n_j+1}$ defined by

$$\mathbf{a}^{(j)} = f^{(j)}\left(\mathbf{a}^{(j-1)}\right) = \begin{bmatrix} 1 \\ g^{(j)}\left(\Theta^{(j-1)}\mathbf{a}^{(j-1)}\right) \end{bmatrix}. \tag{8}$$

- Moreover, when $j = L$, we use the function $f^{(L)} : \mathbb{R}^{n_{L-1}+1} \to \mathbb{R}^{n_L}$ given by

$$\mathbf{a}^{(L)} = f^{(L)}\left(\mathbf{a}^{(L-1)}\right) = g^{(L)}\left(\Theta^{(L-1)}\mathbf{a}^{(L-1)}\right). \tag{9}$$

Forward Propagation

Neural Network as a Composition of Mappings

- To make the above discussion clearer, we analyze the neural network of Fig. 1.
- We already know the following about this example:
- the numbers of units are $n_1 = 3$, $n_2 = 4$ and $n_3 = 2$;
- the unit values are organized in $L = 3$ vectors, $\mathbf{x} = \mathbf{a}^{(1)} \in \mathbb{R}^4$, $\mathbf{a}^{(2)} \in \mathbb{R}^5$ and $\mathbf{y} = \mathbf{a}^{(3)} \in \mathbb{R}^2$.
- Then, in this case, we need 2 activation functions $g^{(2)}$ and $g^{(3)}$. Both are single-variable real functions, since they act on components of vectors.
- We use these functions to define 2 more mappings $f^{(2)}$ and $f^{(3)}$.
- The function $f^{(2)} : \mathbb{R}^4 \to \mathbb{R}^5$ is defined as follows:

$$\mathbf{a}^{(2)} = f^{(2)}(\mathbf{x}) = \begin{bmatrix} 1 \\ g^{(2)}\left(\Theta^{(1)}\mathbf{x}\right) \end{bmatrix}, \tag{10}$$

where the dimension of the matrix $\Theta^{(1)}$ is $n_2 \times (n_1 + 1)$, i.e., 4×4.

Forward Propagation

Neural Network as a Composition of Mappings

- The function $f^{(3)} : \mathbb{R}^5 \rightarrow \mathbb{R}^2$ is defined as follows:

$$\mathbf{y} = f^{(3)}\left(\mathbf{a}^{(2)}\right) = g^{(3)}\left(\Theta^{(2)}\mathbf{a}^{(2)}\right), \tag{11}$$

  where the dimension of the matrix $\Theta^{(2)}$ is $n_3 \times (n_2 + 1)$, i.e., 2×5.

- By using Eqs. (10) and (11), we can show that the output vector $\mathbf{y}$ is obtained from the input vector $\mathbf{x}$ by means of a composition of the mappings $f^{(2)}$ and $f^{(3)}$:

$$\mathbf{y} = f^{(3)}\left(\mathbf{a}^{(2)}\right) = f^{(3)}\left(f^{(2)}(\mathbf{x})\right). \tag{12}$$

- To make this even clearer, we define a mapping $F : \mathbb{R}^4 \rightarrow \mathbb{R}^2$ by

$$F = f^{(3)} \circ f^{(2)}. \tag{13}$$

- Hence:

$$\mathbf{y} = F(\mathbf{x}). \tag{14}$$

Forward Propagation

Neural Network as a Composition of Mappings

- Next, we present the general version of the above idea.
- In general, we consider the composition of the $L-1$ functions $f^{(j)}$, where $j = 2, \ldots, L$.
- This allows us to write the output vector as

$$\mathbf{y} = F(\mathbf{x}), \tag{15}$$

where the mapping $F : \mathbb{R}^{n_1+1} \to \mathbb{R}^{n_L}$ is defined by

$$F = f^{(L)} \circ \cdots \circ f^{(3)} \circ f^{(2)}. \tag{16}$$

- The most important conclusion from this discussion is the following:
- **the propagation of the input values is implemented mathematically by means of a composition of mappings**.

Classification Problems

- Next, we apply the ideas presented so far to solve *classification problems*.
- Specifically, we want to solve multi-class classification problems.
- We denote the number of classes by K ($K \in \mathbb{N}$, $K > 1$).
- This quantity must be equal to the number of units in the output layer, i.e., $K = n_L$.
- To proceed, we can adopt an approach that generalizes *logistic regression*.
- Then we begin by setting every activation function $g^{(j)}$ equal to the sigmoid function h, Eq. (2).
- This means that the propagation of the input values is implemented with the aid of the following mappings:

$$\mathbf{a}^{(j)} = f^{(j)}\left(\mathbf{a}^{(j-1)}\right) = \begin{bmatrix} 1 \\ h\left(\Theta^{(j-1)}\mathbf{a}^{(j-1)}\right) \end{bmatrix} \quad (j \neq L) \tag{17}$$

and

$$\mathbf{a}^{(L)} = f^{(L)}\left(\mathbf{a}^{(L-1)}\right) = h\left(\Theta^{(L-1)}\mathbf{a}^{(L-1)}\right). \tag{18}$$

Learning

Classification Problems

- As before, the input values in $\mathbf{x}$ are propagated with the aid of F, and we obtain the output vector $F(\mathbf{x})$.
- This vector is K-dimensional, i.e., $F(\mathbf{x}) \in \mathbb{R}^K$.
- It's convenient to introduce the following notation for the components of this vector: $F_k(\mathbf{x}) = [F(\mathbf{x})]_k$, where $k = 1, \ldots, K$.

Cost Function

- To continue, we assume our training set has m examples.
- The i-th training example is denoted as follows: $\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right)$.
- It's important to notice that, unlike in logistic regression, the class of the i-th training example is specified through a K-dimensional vector $\mathbf{y}^{(i)}$.
- If the i-th example belongs to the k-th class, then almost every component of $\mathbf{y}^{(i)}$ is zero. The only exception is the k-th component, which is equal to 1.

Cost Function

- Finally, we are in position to write down the formula for the *cost function* J.
- We will express J as a sum of two terms: $J(\Theta) = J_0(\Theta) + J_{\mathrm{r}}(\Theta)$, where Θ represents the totality of the weights in the matrices $\Theta^{(1)}, \ldots, \Theta^{(L-1)}$.
- The first term of $J(\Theta)$ is given by

$$J_0(\Theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} \left\{ y_k^{(i)} \ln \left[F_k \left(\mathbf{x}^{(i)} \right) \right] + \left(1 - y_k^{(i)} \right) \ln \left[1 - F_k \left(\mathbf{x}^{(i)} \right) \right] \right\}. \tag{19}$$

- The first summation on the RHS of this equation is over the training examples.
- The second sum is for taking into account the contributions from all K components of the vectors $\mathbf{y}^{(i)}$ and $F\left(\mathbf{x}^{(i)}\right)$.
- This term is a generalization of the cost function for (unregularized) logistic regression.

Cost Function

- The second term of $J(\Theta)$ is a *regularization term*. It can be expressed as follows:

$$J_{\mathrm{r}}(\Theta) = \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{n_l} \sum_{j=1}^{n_{l+1}} \left(\Theta_{ji}^{(l)} \right)^2, \tag{20}$$

  where λ is the *regularization parameter*.
- The first summation on the RHS of this equation is over the layers of the neural network. Almost every layer makes a contribution. The only exception is the output layer.
- The other two sums are for taking into account the contributions from the different rows and columns of $\Theta^{(l)}$.
- Notice that the columns associated with the bias units don't contribute to $J_{\mathrm{r}}(\Theta)$.
- This is similar to what we have for regularized logistic regression.