Notes on forward and backward propagation in deep neural networks

(Dated: June 26, 2018)

The goals of these notes are twofold:

- expressing mathematically the procedures of forward and backward propagation;
- getting familiar with the notation employed in a full vectorized form of these operations, over features, hidden units, output units and training examples.

To the formality oriented, as myself, it is evident that forward and backward propagation are some complicated forms of function composition and chain rule application, respectively. However, these are not arbitrary instances of these mathematical rules. There is an underlying diagrammatic representation that defines the neural network and which we should take advantage of, but also that limits the class of functions expressible as the result of forward propagation. The generic architecture of a deep neural network is shown in Figure 1. We count the total number of layers including the output layer but excluding the input layer. Therefore a neural network with L layers consists of one input layer, followed by $L-1$ hidden layers, followed by an output layer.

I. FORWARD PROPAGATION

Let us consider the simplest case of a deep neural network, that with a single hidden layer and a single output unit, appropiate for a binary classification problem (see Figure 2). At the moment, we are going to consider only one training example $(\mathbf{x}, y)$, with $\mathbf{x}$ arranged as a column vector with $n^{[0]}$ entries and y a single classification

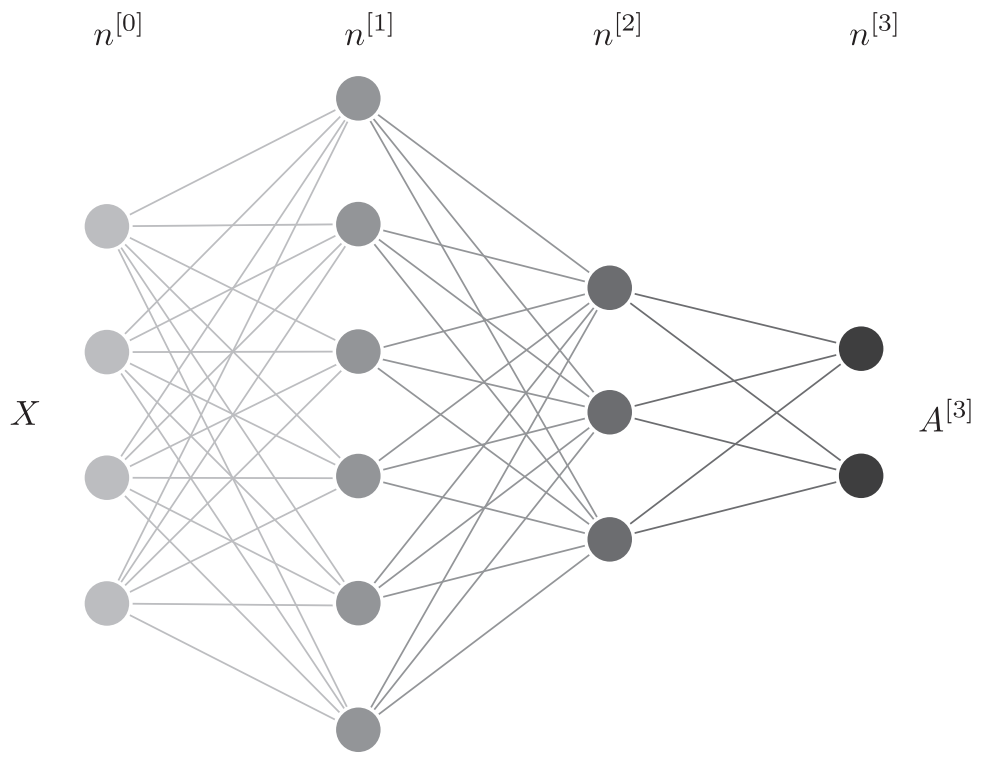

FIG. 1: Architecture for a neural network with three layers: one input layer, 2 hidden layers and one output layer.

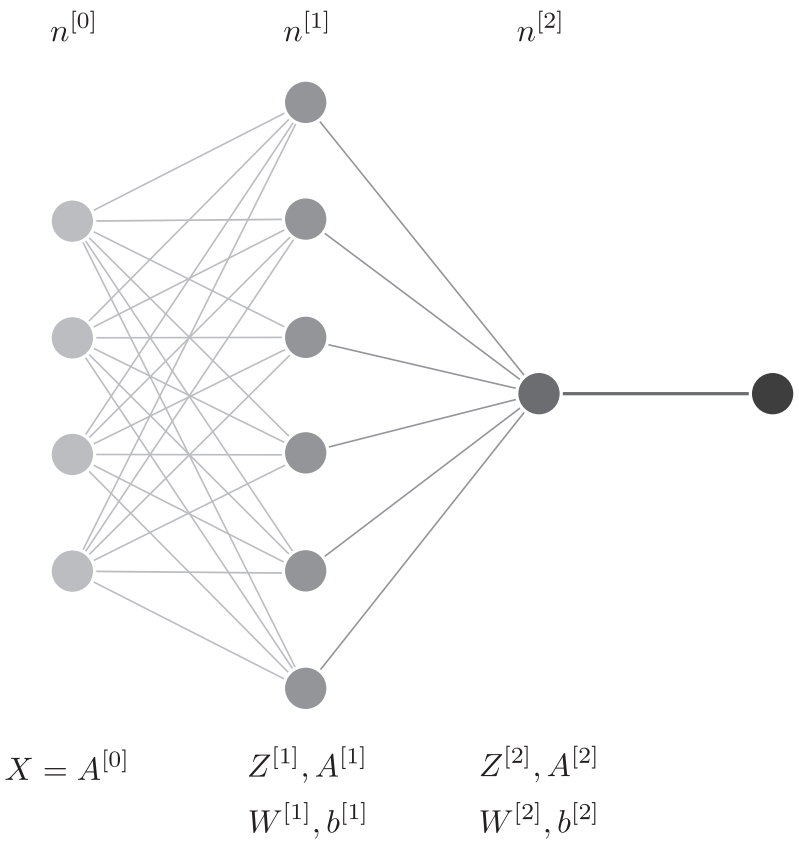

FIG. 2: Architecture for a shallow neural network with one hidden layer. The specifics of this network suffice to formulate forward and backward propagation for any depth neural network.

output, as

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^{n^{[0]}}, \qquad y \in \{0, 1\}. \tag{1}$$

The first and only hidden layer has $n^{[1]}$ units, which are activated by a function $\chi^{[1]}$ (tanh, relu or sigmoid, for example) and the output unit is activated by a function $\chi^{[2]}$. The diagrammatic picture is the following. The circles in the input layer, also called the input units, the first on the left, represent each a feature $a_i^{[0]} = x_i$, so we have in total n units. The following column of circles represent the units in the hidden layer. Consider a single unit j in the hidden layer. This receives as input the units from the preceding layer and outputs a real function $a_j^{[1]}$, so that

$$a_j^{[1]} : \mathbb{R}^{n^{[0]}} \to \mathbb{R}, \tag{2}$$

which is produced by first computing

$$z_j^{[1]} = \sum_k W_{jk}^{[1]} a_k^{[0]} + b_j^{[1]}, \tag{3}$$

followed by the activation function $\chi^{[i]}$, so that

$$a_j^{[1]} = \chi^{[1]} \left(\sum_k W_{jk}^{[1]} a_k^{[0]} + b_j^{[1]} \right), \tag{4}$$

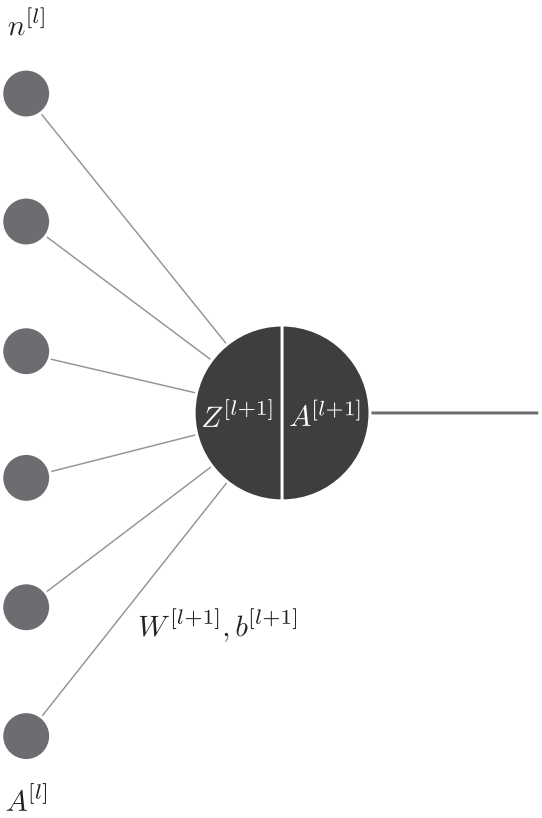

FIG. 3: Forward propagation over a single layer.

where W is a $n^{[1]} \times n^{[0]}$ matrix and $b^{[1]}$ is a $n^{[1]} \times 1$ matrix. And this happens for each unit in the hidden layer. Finally, the output unit performs the same action, takes as input the activation units from its preceding layer and outputs

$$a^{[2]} = \chi^{[2]} \left(\sum_k W_{1k}^{[2]} a_k^{[1]} + b^{[2]} \right), \tag{5}$$

where now $W^{[2]}$ is a $1 \times n^{[1]}$ matrix and $b^{[2]}$ is a 1×1 matrix. The output $a^{[2]}$ can be seen as $a^{[2]} = a^{[2]}\left(\mathbf{x}; W^{[1]}, W^{[2]}, b^{[1]}, b^{[2]}\right)$. This can be considered as a function $\mathbb{R}^{n^{[0]}} \to \mathbb{R}$ with parameters $W^{[1]}$, $W^{[2]}$, $b^{[1]}$, $b^{[2]}$ when, which varied, generate an entire subclass of functions.

II. OPTIMIZATION TASK

Let us define the loss function

$$\mathcal{L}(a^{[2]}, y) = -y \log a^{[2]} - (1-y) \log \left(1 - a^{[2]}\right). \tag{6}$$

This function penalizes that $a^{[2]}$ differs from y. We fix $\mathbf{x}$ and consider the optimization task of finding the parameters $W^{[1]}$, $W^{[2]}$, $b^{[1]}$, $b^{[2]}$ that minimize the loss function. This is achieved by starting from a random point in the space of parameters and following the path of greatest descent until some minimum is found. Since the surface over parameter space corresponding to the loss function is not convex, this minimum might be a local minimum. Random initialization allows to arrive at the global minimum. The path of greatest descent parameterized by s is defined by

$$\frac{d\theta_i}{ds} = -\sum_j \delta_{ij} \frac{\partial \mathcal{L}}{\partial \theta_j}, \tag{7}$$

where θ_i is some parameter and δ_{ij} is the metric in these coordinates. Therefore, it is important that we obtain an expression for the derivates of $\mathcal{L}$ with respect to parameters.

III. BACKWARD PROPAGATION

Let us first consider the derivatives of $\mathcal{L}$ with respect to parameters in the second layer, $W^{[2]}$, $b^{[2]}$. Typically the output unit is activated with the sigmoid function $\sigma(z)$, which satisfies $\sigma'(z) = \sigma(z)(1-\sigma(z))$. We have then

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial z^{[2]}} &= \frac{\partial \mathcal{L}}{\partial a^{[2]}} \frac{da^{[2]}}{dz^{[2]}} = a^{[2]} - y, \\
\frac{\partial \mathcal{L}}{\partial W^{[2]}_{1k}} &= \frac{\partial \mathcal{L}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial W^{[2]}_{1k}} = \left(a^{[2]} - y\right) a^{[1]}_k, \\
\frac{\partial \mathcal{L}}{\partial b^{[2]}} &= \frac{\partial \mathcal{L}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial b^{[2]}} = a^{[2]} - y.
\end{aligned} \tag{8}$$

That was easy. Let us now compute the derivatives of the loss function with respect to farther parameters, those in the first layer, $W^{[1]}$, $b^{[1]}$. In this case,

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial z^{[1]}_j} &= \sum_l \frac{\partial \mathcal{L}}{\partial a^{[2]}} \frac{da^{[2]}}{dz^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}_l} \frac{da^{[1]}_l}{dz^{[1]}_j} = \frac{\partial \mathcal{L}}{\partial z^{[2]}} W^{[2]}_{1j} \frac{d\chi^{[1]}(z)}{dz}\bigg|_{z=z^{[1]}_j}, \\
\frac{\partial \mathcal{L}}{\partial W^{[1]}_{jk}} &= \sum_m \frac{\partial \mathcal{L}}{\partial z^{[1]}_m} \frac{\partial z^{[1]}_m}{\partial W^{[1]}_{jk}} = \frac{\partial \mathcal{L}}{\partial z^{[2]}} W^{[2]}_{1j} \frac{d\chi^{[1]}(z)}{dz}\bigg|_{z=z^{[1]}_j} a^{[0]}_k, \\
\frac{\partial \mathcal{L}}{\partial b^{[1]}_j} &= \sum_m \frac{\partial \mathcal{L}}{\partial z^{[1]}_m} \frac{\partial z^{[1]}_m}{\partial b^{[1]}_j} = \frac{\partial \mathcal{L}}{\partial z^{[2]}} W^{[2]}_{1j} \frac{d\chi^{[1]}(z)}{dz}\bigg|_{z=z^{[1]}_j}.
\end{aligned} \tag{9}$$

In order to formulate a vectorization over features and units of the previous expressions, we consider the following. First, the series of derivatives along a set of parameters will be contained in an object of the same form as those parameters. For example, since $W^{[2]}$ is a $1 \times n^{[1]}]$ matrix, then $\nabla_{W^{[2]}}$ should be of the same form. Additionally, we define an operation $*$ ocurring between matrices of the same and denoting element-wise multiplication. Then a vectorization is given by

$$\begin{aligned}
\nabla_{z^{[2]}} \mathcal{L} &= a^{[2]} - y, \\
\nabla_{W^{[2]}} \mathcal{L} &= \left(\nabla_{z^{[2]}} \mathcal{L}\right) \left(a^{[1]}\right)^T, \\
\nabla_{b^{[2]}} \mathcal{L} &= \left(\nabla_{z^{[2]}} \mathcal{L}\right),
\end{aligned} \tag{10}$$

as well as

$$\begin{aligned}
\nabla_{z^{[1]}} \mathcal{L} &= \left(\nabla_{z^{[2]}} \mathcal{L}\right) \left(W^{[2]}\right)^T * \frac{d\chi^{[1]}(z^{[1]})}{dz}, \\
\nabla_{W^{[1]}} \mathcal{L} &= \left(\nabla_{z^{[2]}} \mathcal{L}\right) \left(W^{[2]}\right)^T * \frac{d\chi^{[1]}(z^{[1]})}{dz} \left(a^{[0]}\right)^T, \\
\nabla_{b^{[1]}} \mathcal{L} &= \left(\nabla_{z^{[2]}} \mathcal{L}\right) \left(W^{[2]}\right)^T * \frac{d\chi^{[1]}(z^{[1]})}{dz}.
\end{aligned} \tag{11}$$

Let us express the previous results from a diagrammatic perspective, considering Figure 4. Let us sit at the output unit, where the variation in the loss due to a variation in the linear output $z^{[2]}$ is $a^{[2]} - y$. Then we append that value to that unit's output. Then, going backward, we consider the line that joins the output unit with the k-th unit from the hidden layer. This line contributes with the weight $W^{[2]}_k$, whose variation produces a variation in the loss given by (10). Then we only multiply the value from the left with the activation $a^{[1]}_k$, the same as for the bias unit activation (which equals one). As we go on, from the k-th unit in the hidden layer to the j-th unit in the input layer, we multiply first by $W^{[2]}_{ik} d\chi^{[1]}(z_k)/dz$ and second by the activation $a_{[j]}$ or by one for the bias unit, according to (11).

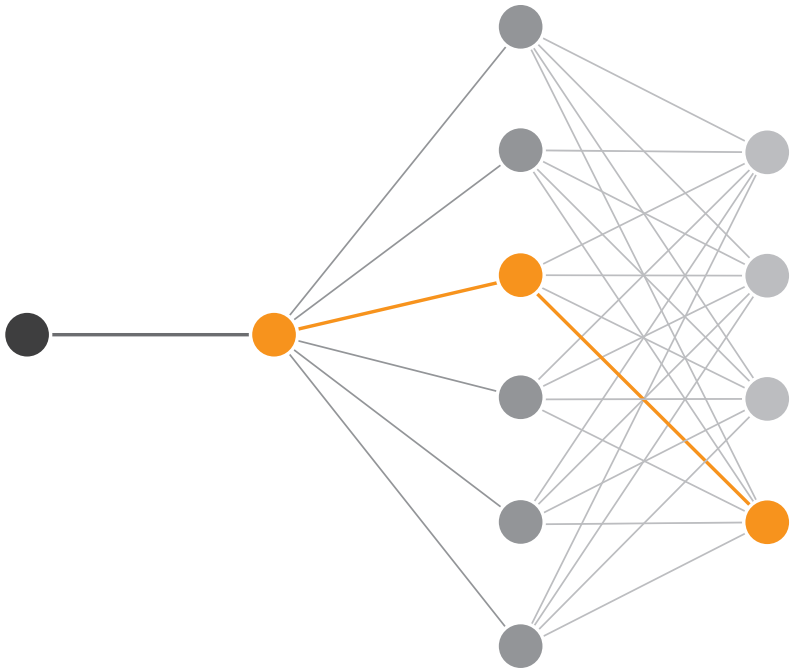

FIG. 4:

IV. VECTORIZING OVER m EXAMPLES

We now include in the $n^{[0]} \times m$ matrix X all the training examples with each training $\mathbf{x}^a$ example as column vector. We also consider $n^{[2]} \times m$ vector Y as consisting of all the training results. Then we have

$$X = \begin{pmatrix} x_1^1 & x_1^2 & \cdots & x_1^{m-1} & x_1^m \\ x_2^1 & x_2^2 & \cdots & x_2^{m-1} & x_2^m \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n^{[0]}-1}^1 & x_{n^{[0]}-1}^2 & \cdots & x_{n^{[0]}-1}^{m-1} & x_{n^{[0]}-1}^m \\ x_{n^{[0]}}^1 & x_{n^{[0]}}^2 & \cdots & x_{n^{[0]}}^{m-1} & x_{n^{[0]}}^m \end{pmatrix}, \qquad Y = \begin{pmatrix} y^1 & y^2 & \cdots & y^{m-1} & y^m \end{pmatrix}. \tag{12}$$

We consider $A^{[0]} = X$. Then **forward propagation** consists in the following series of steps

1. linear propagation to the first layer (yielding a $(n^{[1]} \times n^{[0]}) \cdot (n^{[0]} \times m) = n^{[1]} \times m$ matrix):

$$Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}; \tag{13}$$

2. unit activation in the first layer:

$$A^{[1]} = \chi^{[1]} \left(Z^{[1]} \right); \tag{14}$$

3. linear propagation to the second layer (yielding a $(n^{[2]} \times n^{[1]}) \cdot (n^{[1]} \times m) = n^{[2]} \times m$ matrix):

$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}; \tag{15}$$

4. unit activation in the second layer:

$$A^{[2]} = \sigma \left(Z^{[2]} \right). \tag{16}$$

We define $\mathrm{i}_{n_1 \times n_2}$ as a $n_1 \times n_2$ matrix full of 1's, which can be programmed through broadcasting operations. We compute the **cost function** as the average of the individial losses over all examples as

$$\mathcal{L}\left(A^{[2]}, Y \right) = -\frac{1}{m} \left[\left(\log A^{[2]} \right) Y^T + \left(\mathrm{i}_{n^{[2]} \times m} - \log A^{[2]} \right) \left(\mathrm{i}_{n^{[2]} \times m} - Y \right)^T \right]. \tag{17}$$

Finally, **backward propagation** is performed through the following series of steps:

1. output layer linear variation:

$$\nabla_{Z^{[2]}}\mathcal{L} = \frac{1}{m}\left(A^{[2]} - Y\right); \tag{18}$$

2. weights and biases variations in the second layer:

$$\nabla_{W^{[2]}}\mathcal{L} \ = \nabla_{Z^{[2]}}\mathcal{L} \cdot \left(A^{[1]}\right)^T, \tag{19}$$

$$\nabla_{b^{[2]}}\mathcal{L} \ = \nabla_{Z^{[2]}}\mathcal{L} \cdot \left(\mathrm{i}_{1\times m}\right)^T; \tag{20}$$

3. hidden layer linear variation:

$$\nabla_{Z^{[1]}}\mathcal{L} = \left[\left(W^{[2]}\right)^T \cdot \nabla_{Z^{[2]}}\mathcal{L}\right] * \frac{d\chi^{[1]}(Z^{[1]})}{dz} \tag{21}$$

4. weights and biases variations in the second layer:

$$\nabla_{W^{[1]}}\mathcal{L} \ = \nabla_{Z^{[1]}}\mathcal{L} \cdot \left(A^{[0]}\right)^T, \tag{22}$$

$$\nabla_{b^{[1]}}\mathcal{L} \ = \nabla_{Z^{[1]}}\mathcal{L} \cdot \left(\mathrm{i}_{1\times m}\right)^T; \tag{23}$$

V. GENERALIZING TO ARBITRARY NUMBER OF HIDDEN LAYERS

From the preceding discussion, it is straightforward to derive a generalization to a neural network with an arbitrary number of layers. Let us consider a neural network with the following architecture:

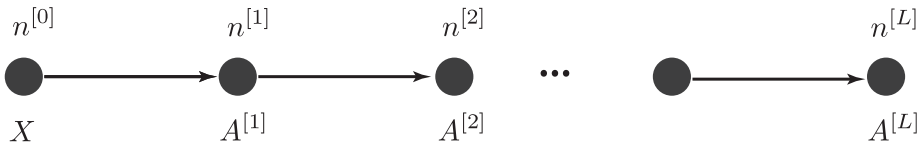

FIG. 5:

Forward propagation

$A^{[0]} \leftarrow X$
for $l = 1, \ldots, L$ **do**
  $Z^{[l]} \leftarrow W^{[l]}A^{[l-1]} + b^{[l]}$
  $A^{[l]} \leftarrow \chi^{[l]}\left(Z^{[l]}\right)$
end for

Backward propagation

$\nabla_{Z^{[L]}}\mathcal{L} \leftarrow \frac{1}{m}\left(A^{[L]} - Y\right)$
for $l = L, \ldots, 1$ **do**
  $\nabla_{W^{[l]}}\mathcal{L} \leftarrow \nabla_{Z^{[l]}}\mathcal{L} \cdot \left(A^{[l-1]}\right)^T$
  $\nabla_{b^{[l]}}\mathcal{L} \leftarrow \nabla_{Z^{[l]}}\mathcal{L} \cdot \left(\mathrm{i}_{n^{[l]}\times m}\right)^T$
  $\nabla_{Z^{[l-1]}}\mathcal{L} \leftarrow \left[\left(W^{[l]}\right)^T \cdot \nabla_{Z^{[l]}}\mathcal{L}\right] * \frac{d\chi^{[l-1]}(Z^{[l-1]})}{dz}$
end for