Chapter 02 – Shallow and Deep Neural Networks

Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. Shallow Neural Networks

Shallow Networks are functions $y = f[x, \phi]$ with ϕ which maps multivariate inputs x to multivariate outputs y.

We have seen, in the previous chapter, that θ is the vector of parameters. Given this, we can write our vector ϕ as follows:

$$\phi = \{\phi_0, \phi_1, \phi_2, \phi_3, \theta_{10}, \theta_{11}, \theta_{20}, \theta_{21}, \theta_{30}, \theta_{31}\}$$

We know that a Neural Network is a linear function. Actually, it is a linear combination of a function (called Activation Function) of another linear function (Linear Layers).

Equivalently, we can finally write our output y with the following notation:

$$\begin{aligned} y &= \mathrm{f}[x, \boldsymbol{\phi}] \\ &= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x] \end{aligned}$$

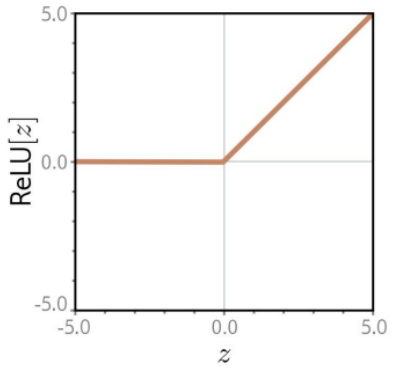
Linear layers

Where the red rectangles are Linear Layers of our model.

1.1 ReLU Activation function

We can see that, in the formula above, we have a letter "a", which stands for Activation Function. The most common AF that we can encounter is ReLU which can be expressed as:

$$\text{n choice } \mathrm{a}[z] = \mathrm{ReLU}[z] = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$$



SAPIENZA UNIVERSITÀ DI ROMA

1.2 Summing up

Taking into account the first formula, we introduce three temporary variables h_1, h_2, h_3 which are respectively the first, second and third red rectangles (each followed by the Activation Function "a").

We can refer to h_1, h_2, h_3 as **Hidden Units**.

By combining them with a linear function, we get: $y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$

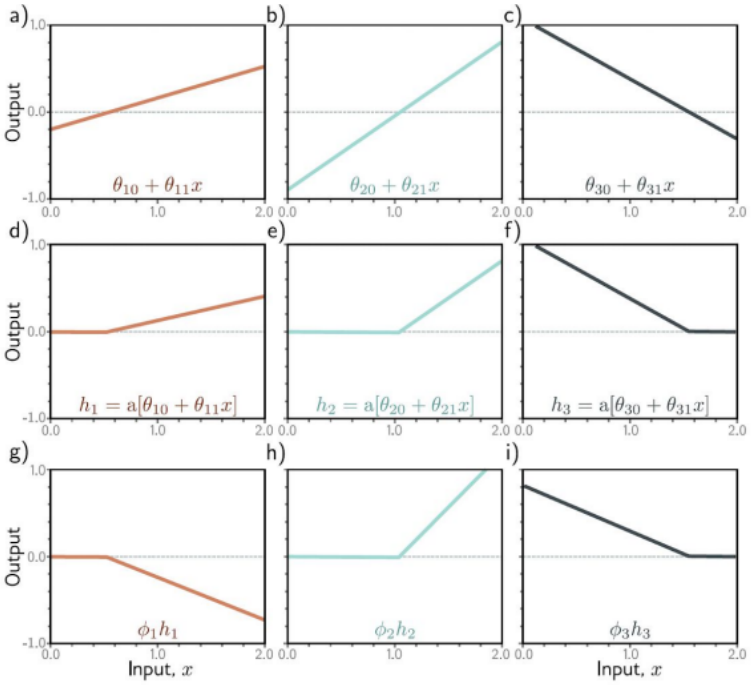
Below, I leave a very intuitive way of explaining ReLU function applied on h_1, h_2, h_3.

In this case, if we take the first column, we are talking about the first hidden unit.

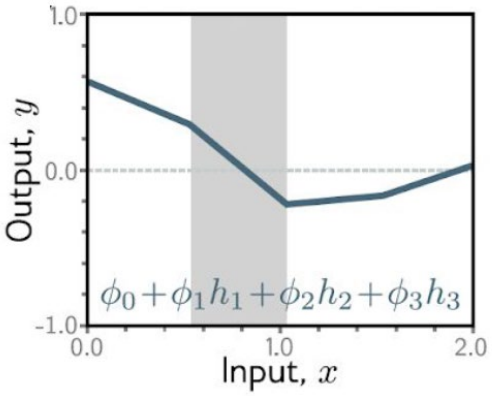
We can see that in the first row, we just have our input which is a straight line. If we pass this through the activation function (ReLU) which shrink negative values to 0, we'll see exactly this result. At the end, we weight the result from the Hidden Unit by the parameter ϕ_1 , having our final result (third row in the bottom 3x3 left image).

Each hidden unit contains a linear function $\theta_{\bullet 0} + \theta_{\bullet 1}x$ of the input, and that line is clipped by the ReLU function a[•] below zero. The positions where the three lines cross zero become the three "joints" in the final output. The three clipped lines are then weighted by ϕ_1, ϕ_2, and ϕ_3, respectively. Finally, the offset ϕ_0 is added, which controls the overall height of the final function.

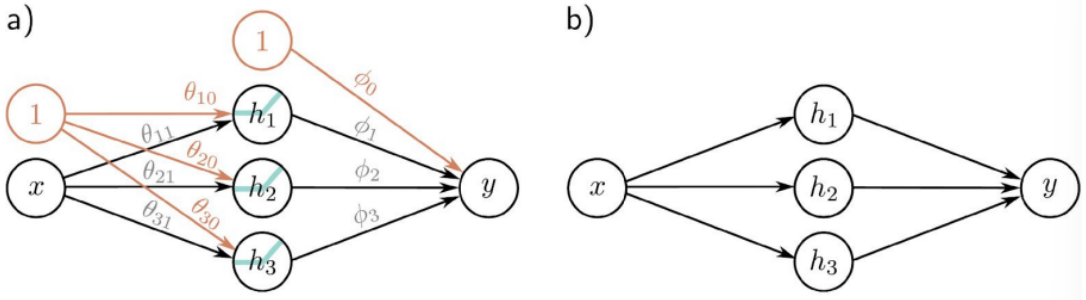
SAPIENZA UNIVERSITÀ DI ROMA



At the end, when we sum up all the hidden units weighted by the parameter ϕ_n, we'll have something like this here below:



This below it's just another way of representing what we explained until now. There are 2 special nodes which have values fixed to 1. What is the effect on the model if we don't have a bias? If we do not have a bias, we will never be able to represent a negative bias with this activation function (ReLU) and also our data will always be at the same height. The bias term allows to have some flexibility in activation functions even when the input is zero and also it helps the network to better fit complex patterns in the data.

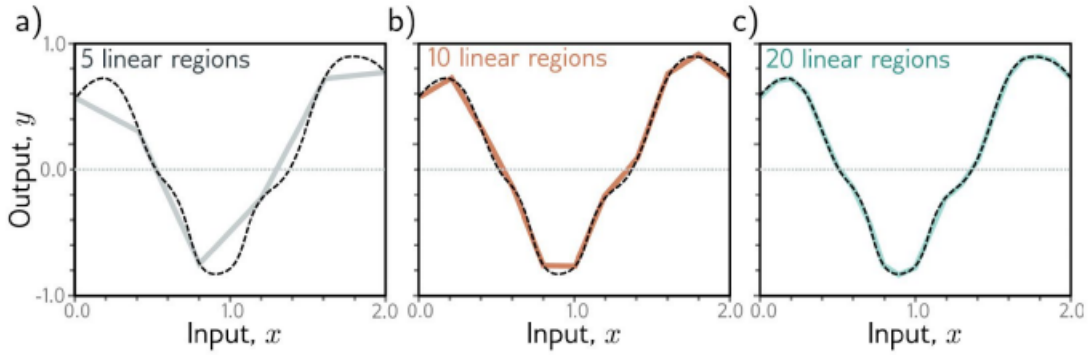


1.3 Universal Approximation Theorem

The UAT states that, the output of a network with D hidden units have at most D joints and so is a piecewise linear function with at most $D + 1$ linear regions. Why that "+ 1"? Because we have to take into consideration the bias ϕ_0. In fact, we have: $y = \phi_0 + \sum_{d=1}^{D} \phi_d h_d$. So, for example if we have 5 activation functions, we will surely have at most 6 linear regions.

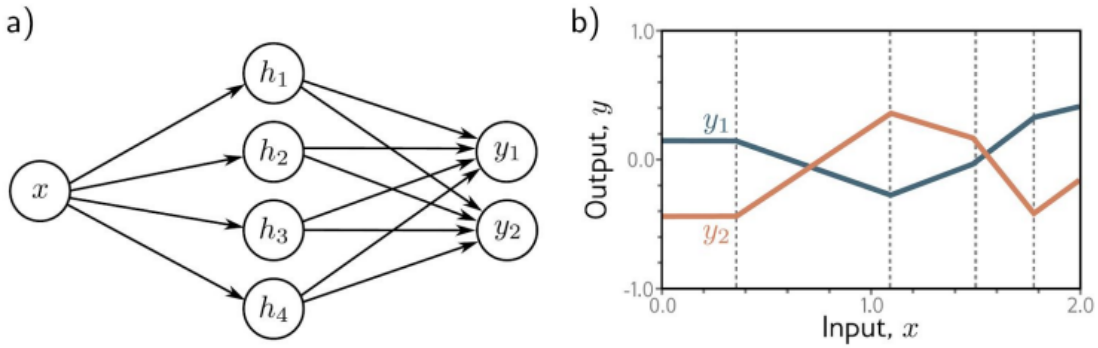
With this being said, a Shallow Network can describe any continuous 1D function. In order to visualize this statement, we can reason about the fact that every time we add a hidden unit, we add another linear region to the function. The further we add them, the more we'll approximate up to any specified precision.

"We say that the set of the functions represented by Neural Networks are Dense in the set of continuous functions".



1.4 Multivariate Input / Outputs

I think no description here is needed. We already explained everything earlier, we are just adding a new output y_1. Both y_0 and y_1 are not independent, due to backpropagation etc.



$$y_1 = \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}h_3 + \phi_{14}h_4$$

$$y_2 = \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}h_3 + \phi_{24}h_4$$

As for the inputs, instead, let's assume we have 2 of them, not just one.

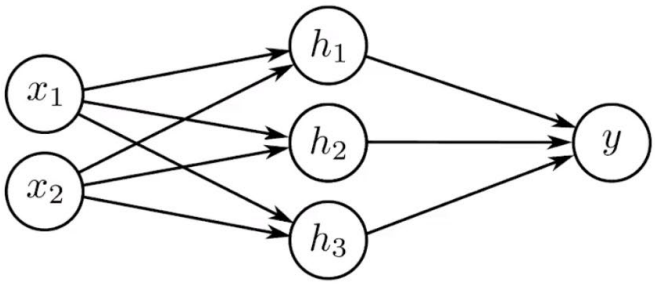
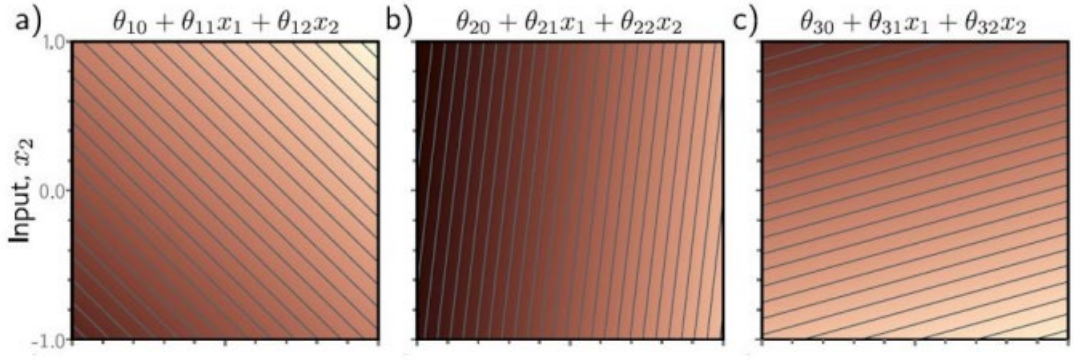
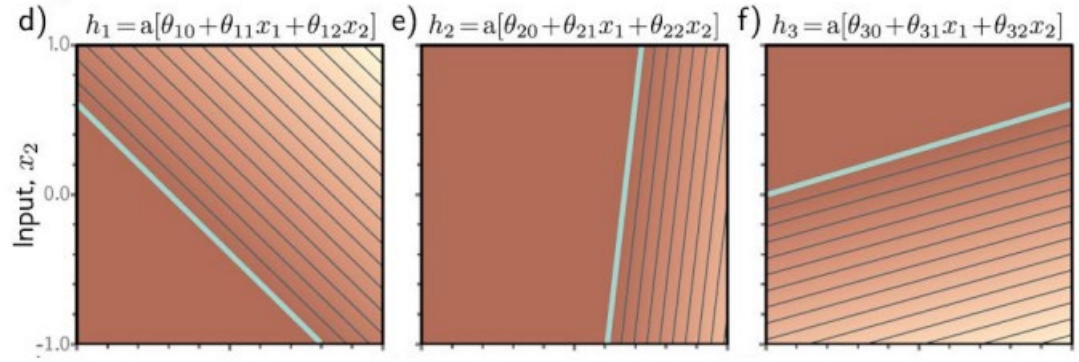


Figure 3.7 Visualization of neural network with 2D multivariate input $\mathbf{x} = [x_1, x_2]^T$ and scalar output y.

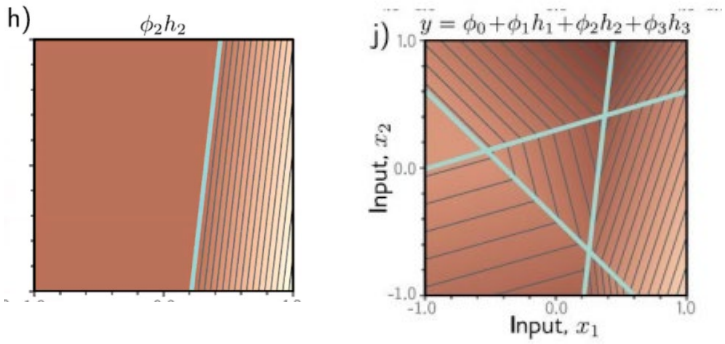
The input to each hidden unit is a linear function of the two inputs, which corresponds to an oriented plane.



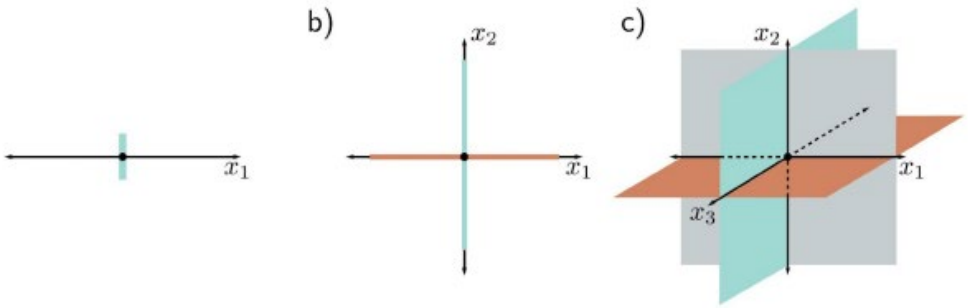
Then, each plane is clipped by the ReLU activation function which gives as output h_1, h_2, h_3. The cyan lines are the "joints" as in the 1D case.



Then, the clipped planes are then weighted (effectively we can see that the lines of h_2 are way closer, meaning that the plane is growing in a quicker way – **Picture H**.) Finally, we sum everything all together, giving in output our final prediction y. (On the right – **Picture J**). In this last picture, we have 7 linear regions and not 8 because we can see that there is a region which is always 0 (no lines in between – midleft part of the image) and so we have (*n. max regions* – 1) regions.



We can reason about this also in terms of regions, with the picture below showing, respectively, a single, double and triple input dimension:



One important aspect to consider is that, increasing the number of input dimensions, drastically increases the number of regions (and so parameters of our model).

The final architecture for a Shallow Network which has multiple inputs / outputs, is the one we can find on the right.

If we have to formally define an equation for a Shallow Neural Network $y = f[x, \phi]$, it would be:

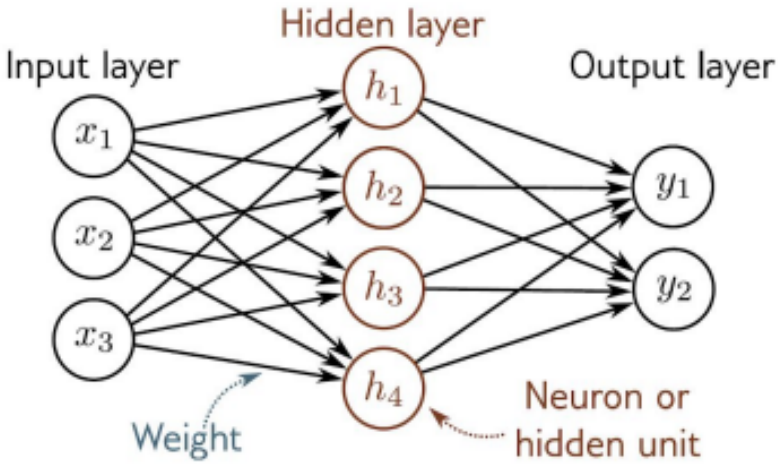
$$h_d = a[\theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} x_i]$$

Combined linearly to create the following output:

$$y_j = \phi_{j0} + \sum_{d=1}^{D} \phi_{jd} h_d$$

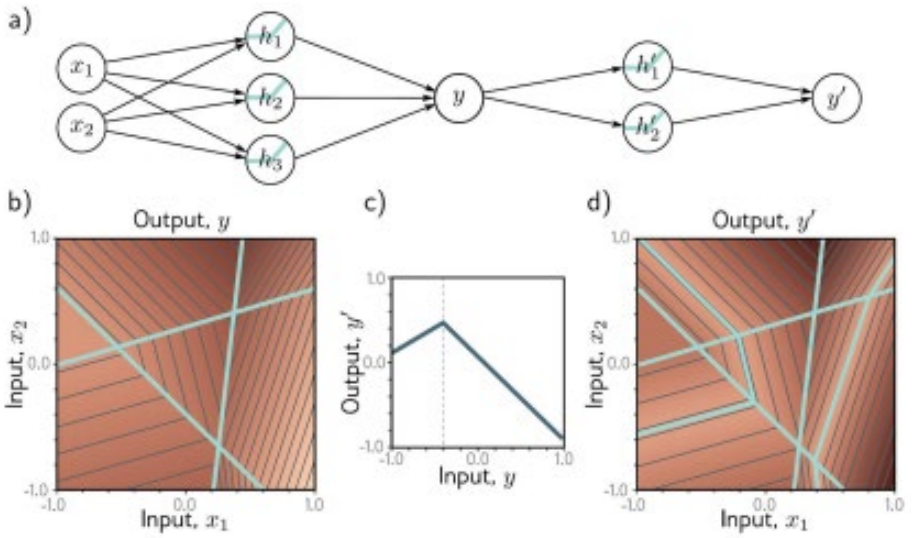
In the right image, we have some terminology. Here, this is a Neural Network with 4 Hidden Units (so we can have maximum 5 linear regions).

In total, though, we have 3 layers because we have the input layer, the hidden layer and the output layer.



2. Deep Neural Networks

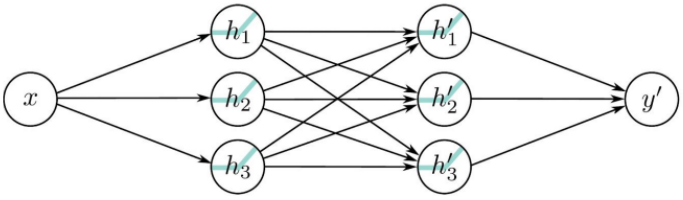
Very simply, a Deep Neural Network is a composition of 2 or multiple Shallow Neural Networks, where the first output y_1 is the input of our second hidden units: h'_1, h'_2, h'_3.



We can now define a Multi-Layer Perceptron (**MLP**) as a Neural Network with one input, one output and two or more hidden layers. This is the basic architecture:

- First Layer

$$\begin{aligned} h_1 &= a[\theta_{10} + \theta_{11}x] \\ h_2 &= a[\theta_{20} + \theta_{21}x] \\ h_3 &= a[\theta_{30} + \theta_{31}x] \end{aligned}$$



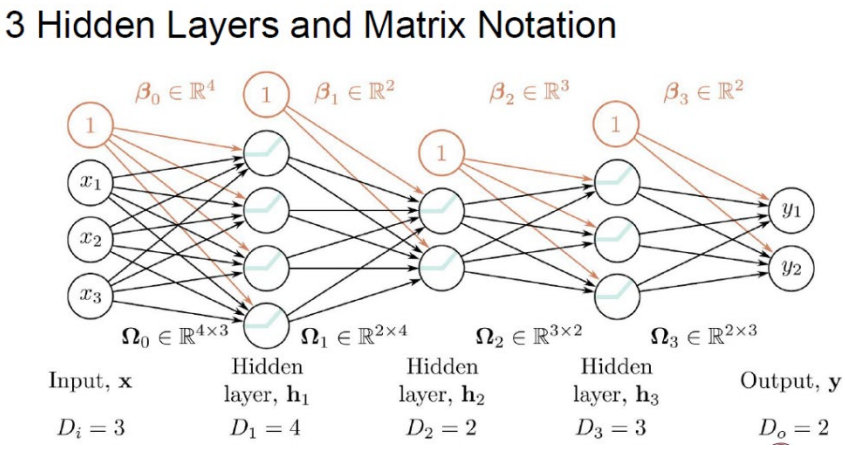
- Second Layer

$$\begin{aligned} h'_1 &= a[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3] \\ h'_2 &= a[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3] \\ h'_3 &= a[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3] \end{aligned}$$

- Output

$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$

The number of hidden units in each layer define the width of the network, while the number of hidden layers define the depth. Along with that, if we denote K as the number of Layers and $D_1, D_2, \ldots, D_k$ the hidden units, then we are talking about **Hyperparameters**, which are quantities chosen before we start the training of our model.



In the image above, when we are at the h_2 Hidden Layer, we would need 2^4 weights, because we need 2 lines starting from one single hidden unit of the first Hidden Layer h_1.

Typically, we need a Shallow Network with an exponential number of hidden units as we go further in the layers in order to approximate a complex function. The same approximation can be achieved easily with a Deep Neural Network. For this reason, we are denoting this phenomenon as "**Depth efficiency of Neural Networks**".

2.1 Advantages

Deep Neural Networks have more pros when compared to Shallow Networks.

Here we list some of them:

- *Fitting*:
    - Usually is easier training Deep Networks than training Shallow ones.
    - If we have a prominent number of parameters, we can easily find solutions. However, though, as we add more and more hidden layers, training becomes more difficult again.
- Deep Neural Networks generalize new data better than Shallow Networks.
- Most of the best results in literature are achieved through the use of Deep Neural Networks with tens or hundreds of Layers, rather than using Shallow Networks.