UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

Lecture 03

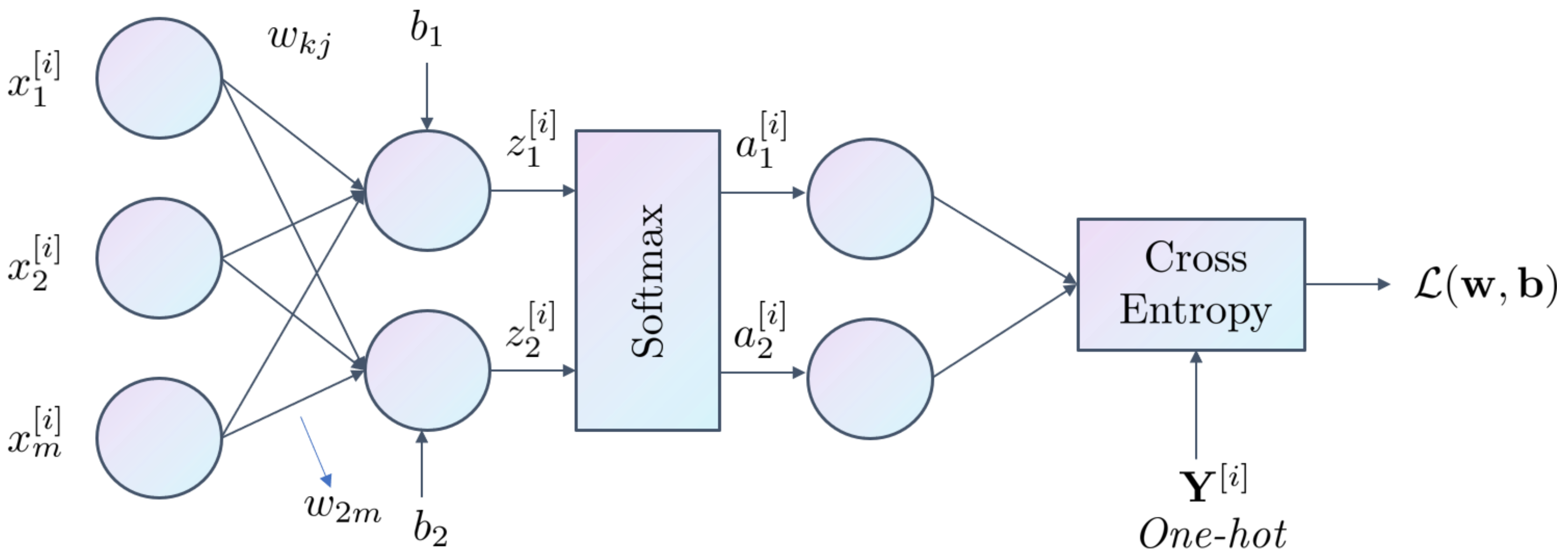
Redes Neuronales Feed Forward

Temas de la Lección

Universidad EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Redes neuronales multi-capa

Softmax



$\mathbf{x}^{[i]} \in \Re^{m \times 1}$

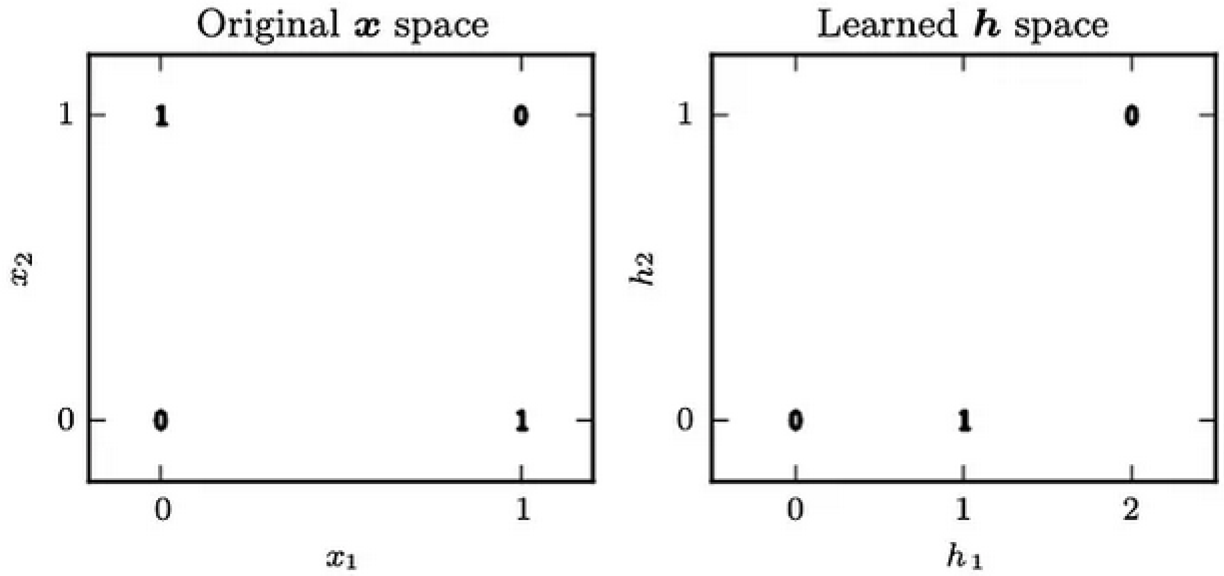
$\mathbf{X} \in \Re^{b \times m}$

$\mathbf{W} \in \Re^{K \times m}$ $\quad$ $\mathbf{z} \in \Re^{K \times 1}$ $\quad$ $\mathbf{a} \in \Re^{K \times 1}$

$\mathbf{Z} \in \Re^{b \times K}$ $\quad$ $\mathbf{A} \in \Re^{b \times K}$

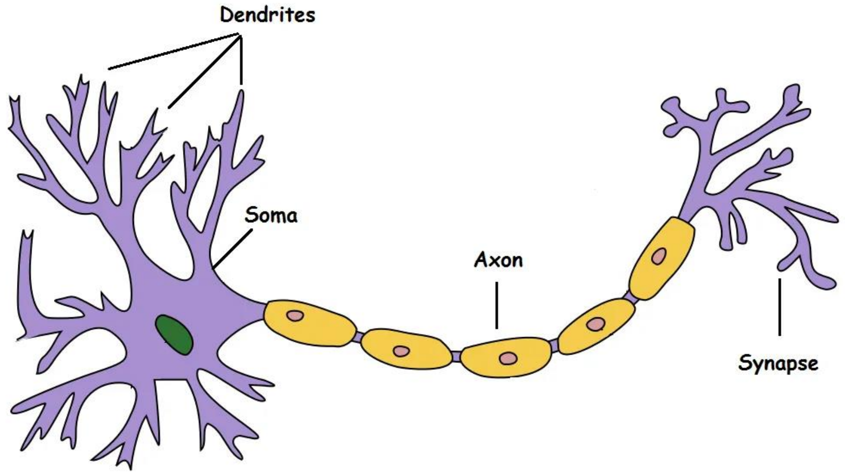
$k = 1, \ldots, K$: llega
$j = 1, \ldots, m$: sale

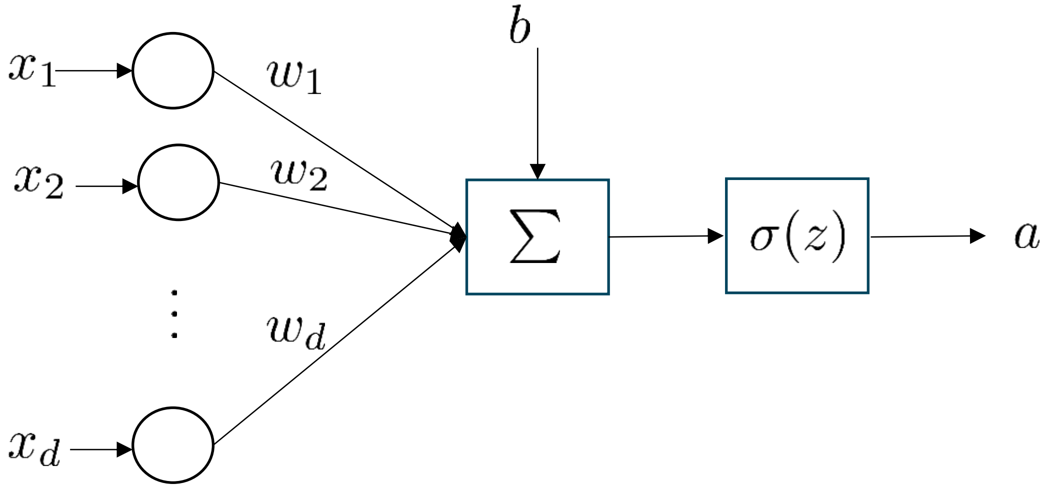
XOR



Original $\boldsymbol{x}$ space
Learned $\boldsymbol{h}$ space
x_1
x_2
h_1
h_2

Neurona de McCulloch-Pitts





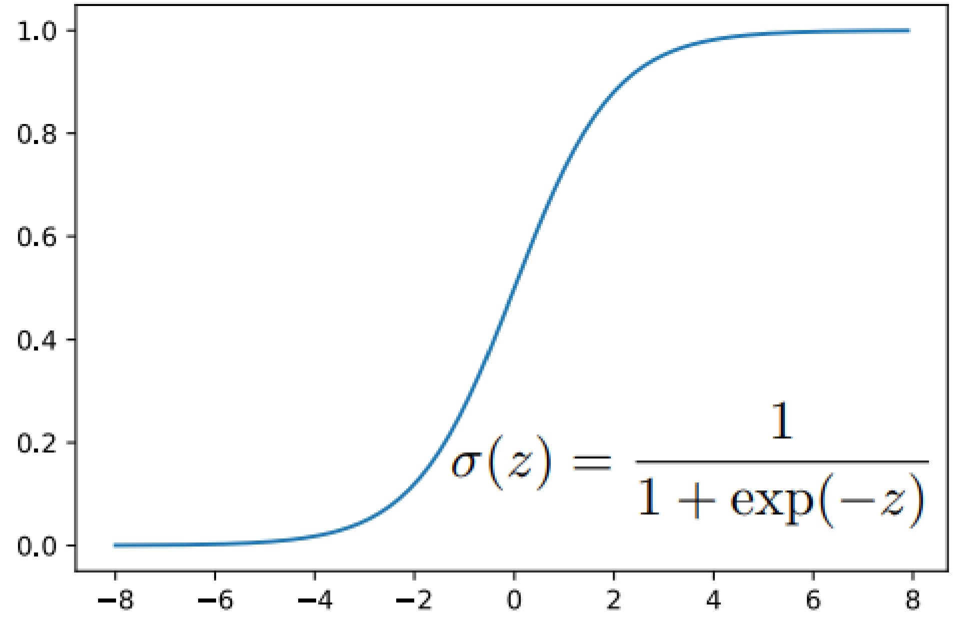
Be very careful with your brain analogies!

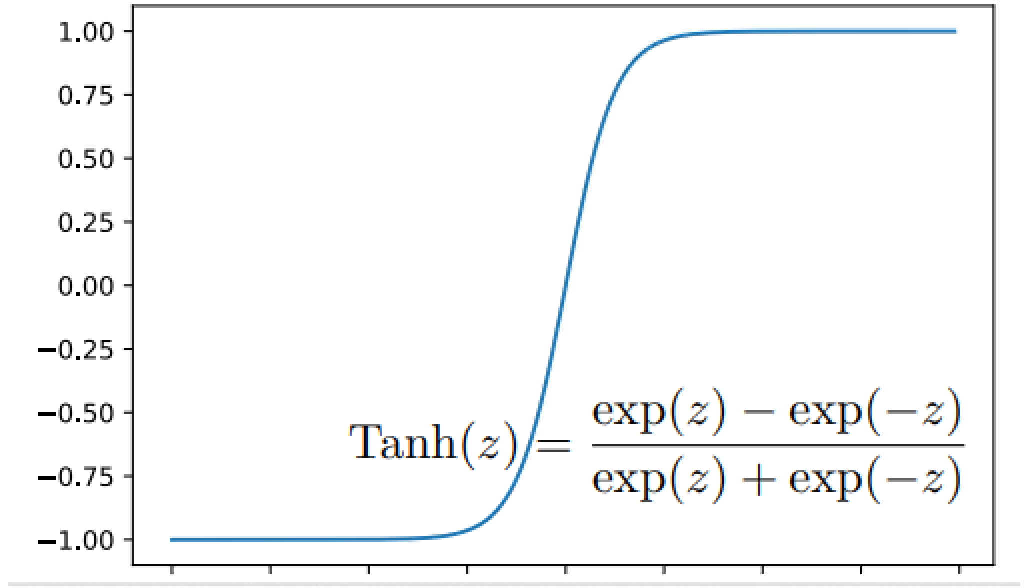
Biological Neurons:

- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system

Funciones de activación

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

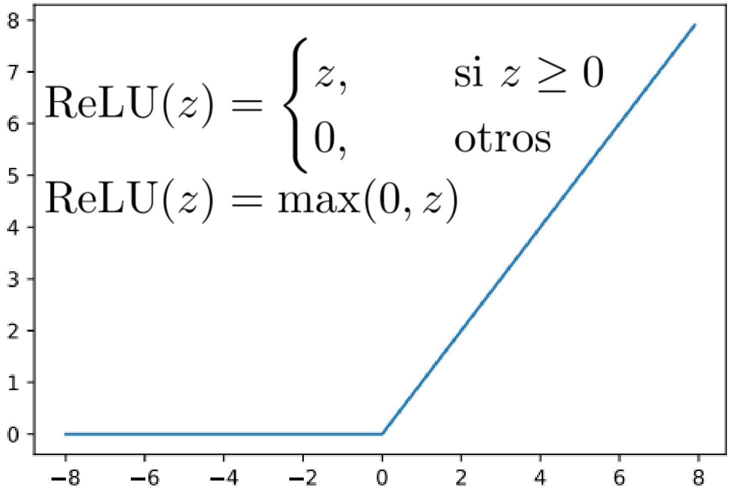




Algunas ventajas de la tangente hiporbólica son:

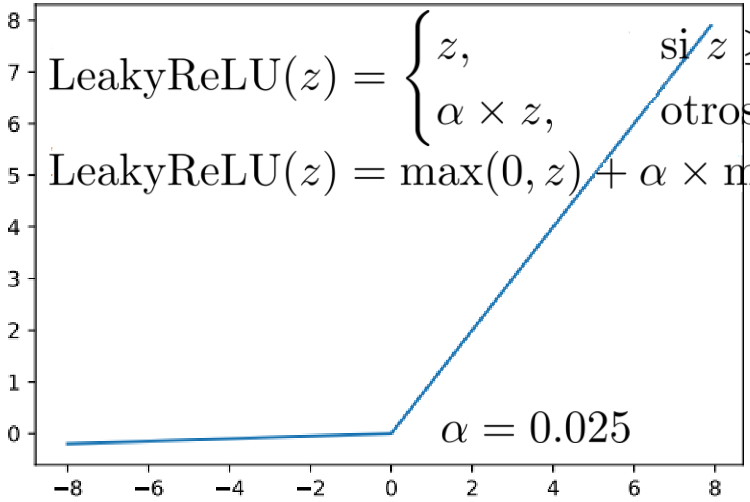
Media centrada, valores positivos y negativos, grandes gradientes, normaliza las entradas a media cero, derivada simple

Funciones de activación



$$\mathrm{ReLU}(z) = \begin{cases} z, & \text{si } z \geq 0 \\ 0, & \text{otros} \end{cases}$$

$$\mathrm{ReLU}(z) = \max(0, z)$$

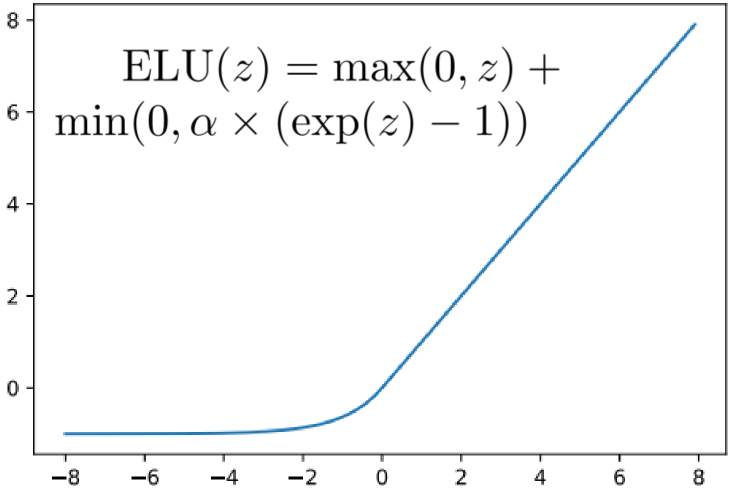


$$\mathrm{LeakyReLU}(z) = \begin{cases} z, & \text{si } z \geq 0 \\ \alpha \times z, & \text{otros} \end{cases}$$

$$\mathrm{LeakyReLU}(z) = \max(0, z) + \alpha \times \min(0, z)$$

$$\alpha = 0.025$$

ELU (Exponential Linear Unit)



$$\mathrm{ELU}(z) = \max(0, z) + \min(0, \alpha \times (\exp(z) - 1))$$

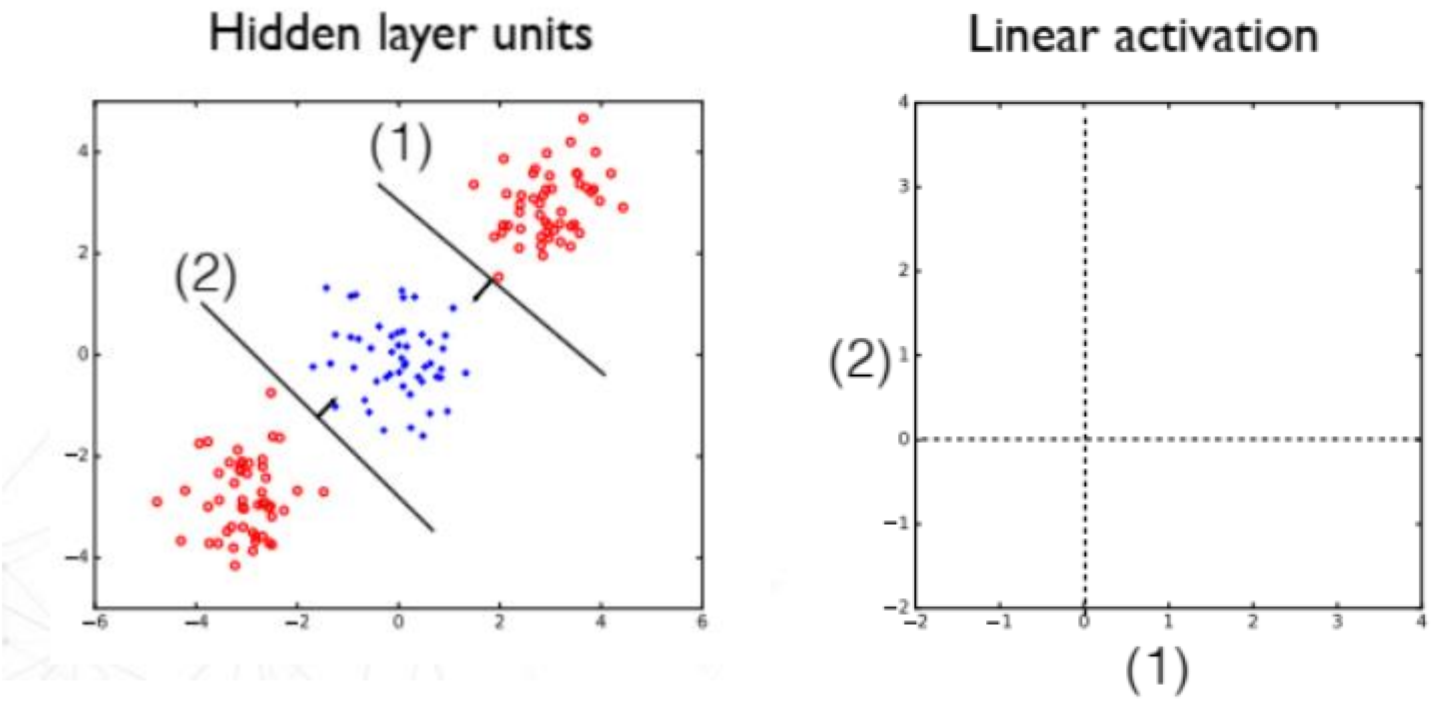
PReLU (Parameterized Rectified Linear Unit)

Aquí α es un parámetro entrenable

$$\mathrm{PReLU}(z) = \begin{cases} z, & \text{si } z \geq 0 \\ \alpha z, & \text{otros} \end{cases}$$

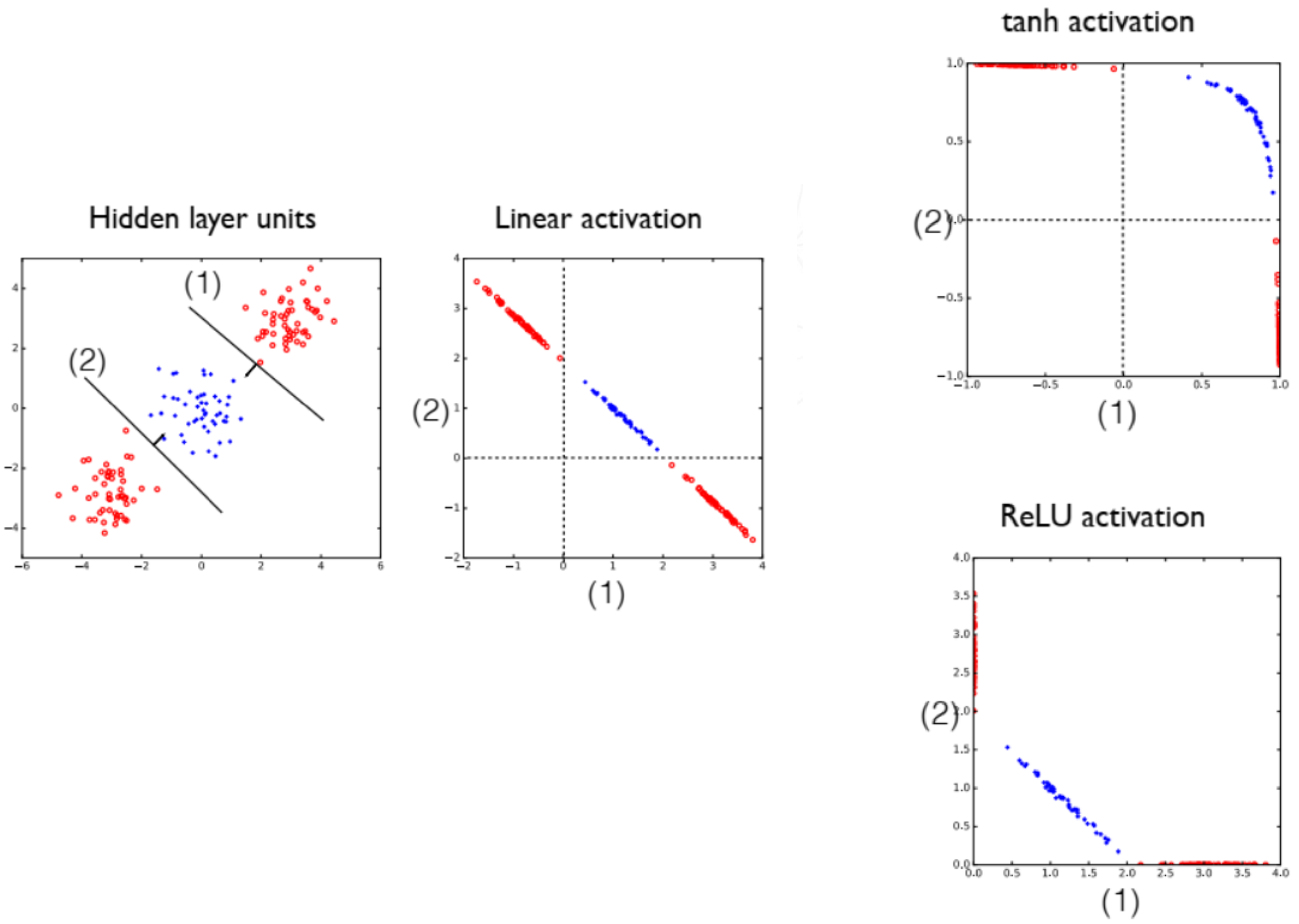
$$\mathrm{PReLU}(z) = \max(0, z) + \alpha \times \min(0, z)$$

Ejemplo



Ejemplo

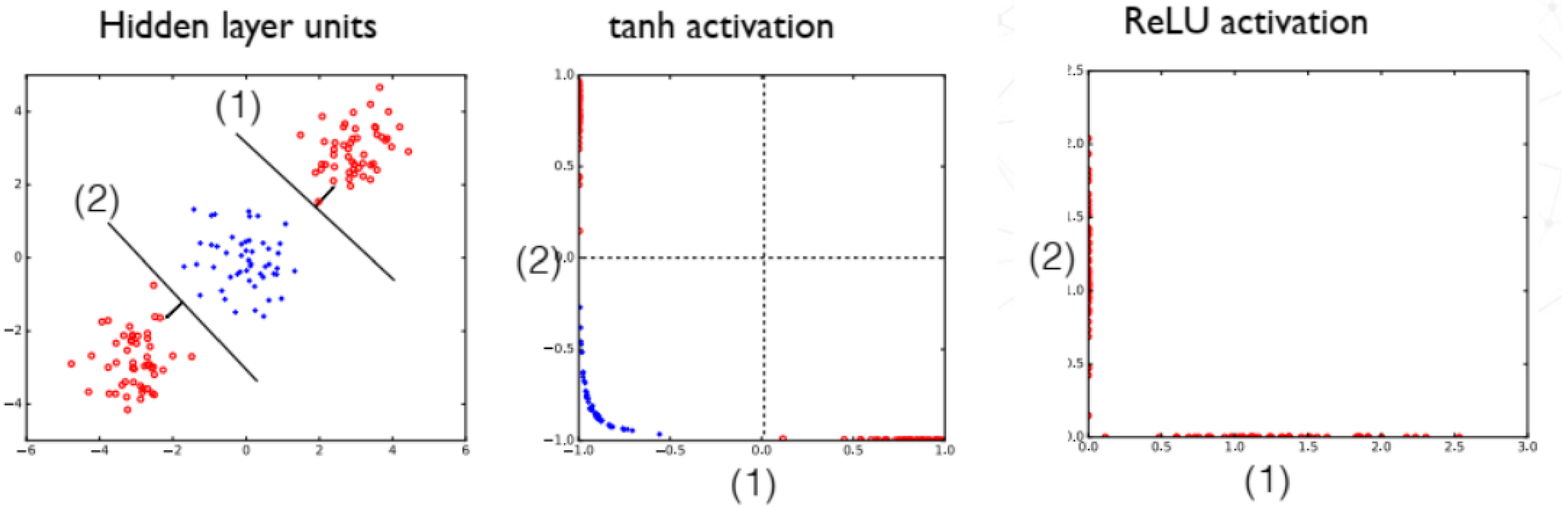
UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

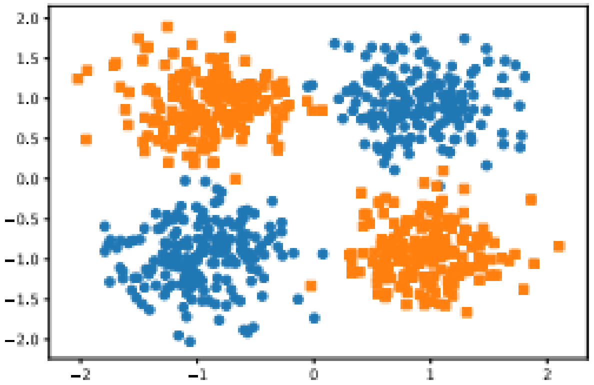


Ejemplo

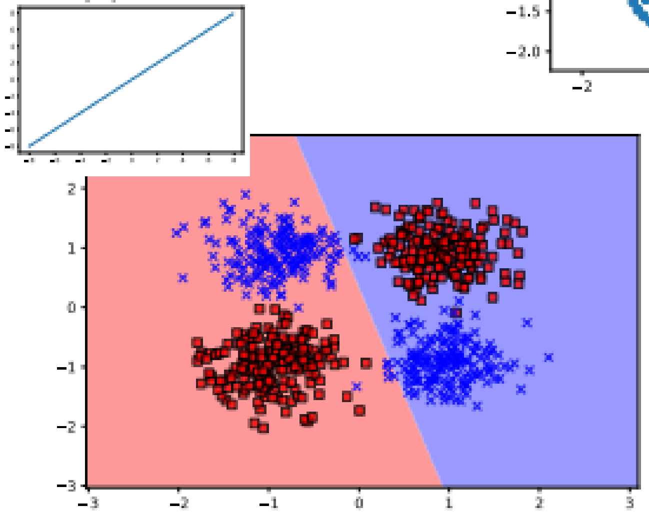
UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Dirección de la ecuación normal



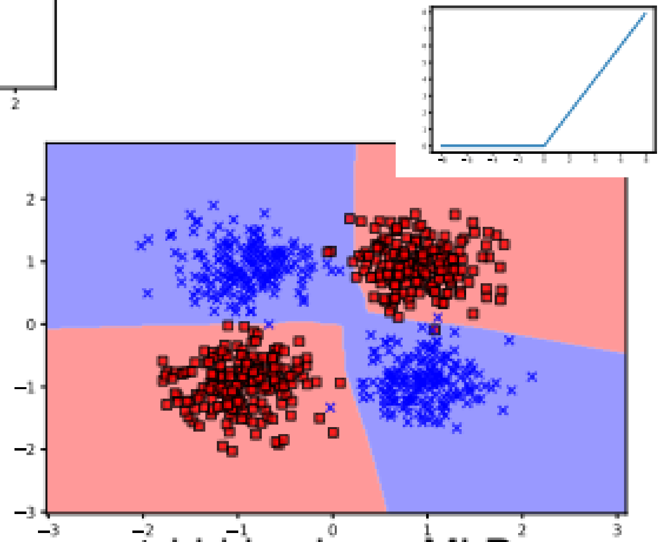


$\sigma(z) = z$



1-hidden layer MLP
with linear activation function

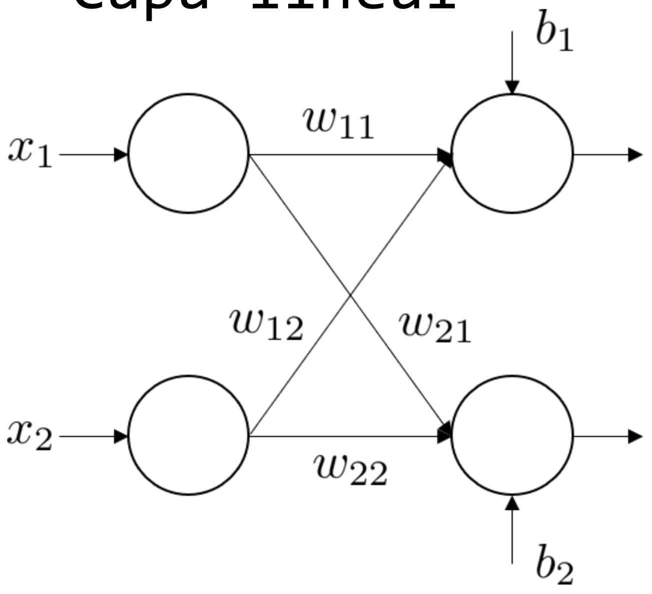
$\sigma(z) = \max(0, z)$



1-hidden layer MLP
with non-linear activation function (ReLU)

Feed Forward Neural Networks

- Capa lineal



$$z_1 = w_{11}x_1 + w_{12}x_2 + b_1$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$z_2 = w_{21}x_1 + w_{22}x_2 + b_2$$

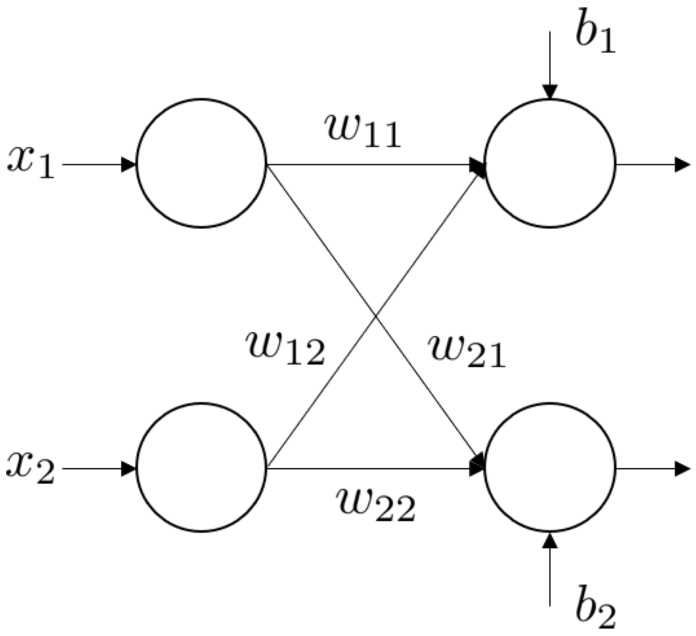
$\mathbf{z} \in \Re^{h_{out} \times 1}$ $\quad$ $\mathbf{W} \in \Re^{h_{out} \times h_{in}}$ $\quad$ $\mathbf{x} \in \Re^{h_{in} \times 1}$ $\quad$ $\mathbf{b} \in \Re^{h_{out} \times 1}$

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Feed Forward Neural Networks

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Capa lineal



$$\mathbf{X} \in \Re^{n \times h_{in}}$$

$$\mathbf{X} = \begin{bmatrix} x_1^{[1]} & \dots & x_{h_{in}}^{[1]} \\ \vdots & \vdots & \vdots \\ x_1^{[n]} & \dots & x_{h_{in}}^{[n]} \end{bmatrix}$$

$$\mathbf{W} \in \Re^{h_{out} \times h_{in}}$$

$$\mathbf{w}_h \in \Re^{h_{in} \times 1}$$

$$\mathbf{W} = \begin{bmatrix} - - \mathbf{w}_1^\top - - \\ \vdots \\ - - \mathbf{w}_{h_{out}}^\top - - \end{bmatrix}$$

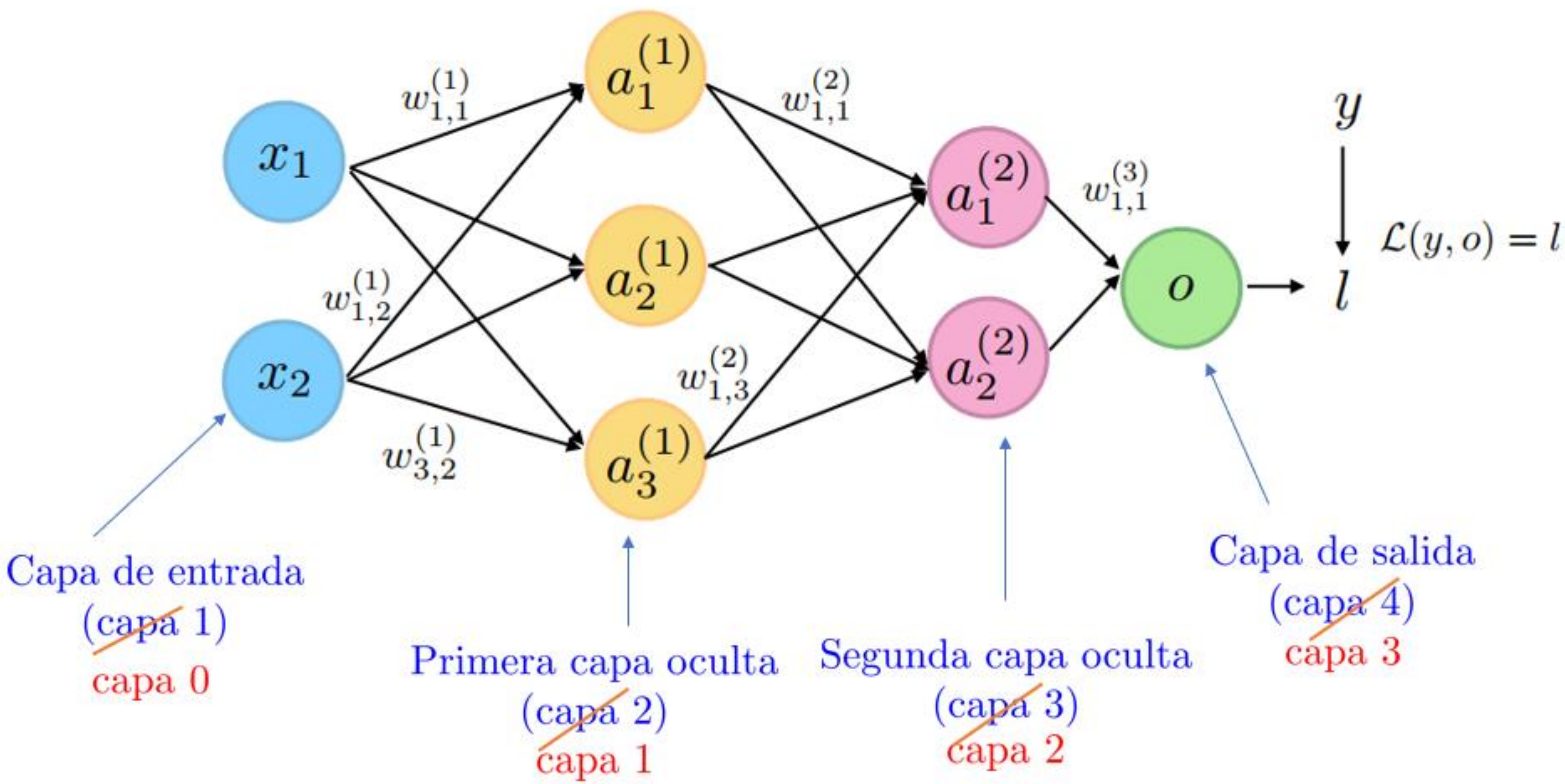
$$\mathbf{b} \in \Re^{h_{out} \times 1}$$

$$\mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_{h_{out}} \end{bmatrix}$$

$$\mathbf{Z} \in \Re^{n \times h_{out}}$$

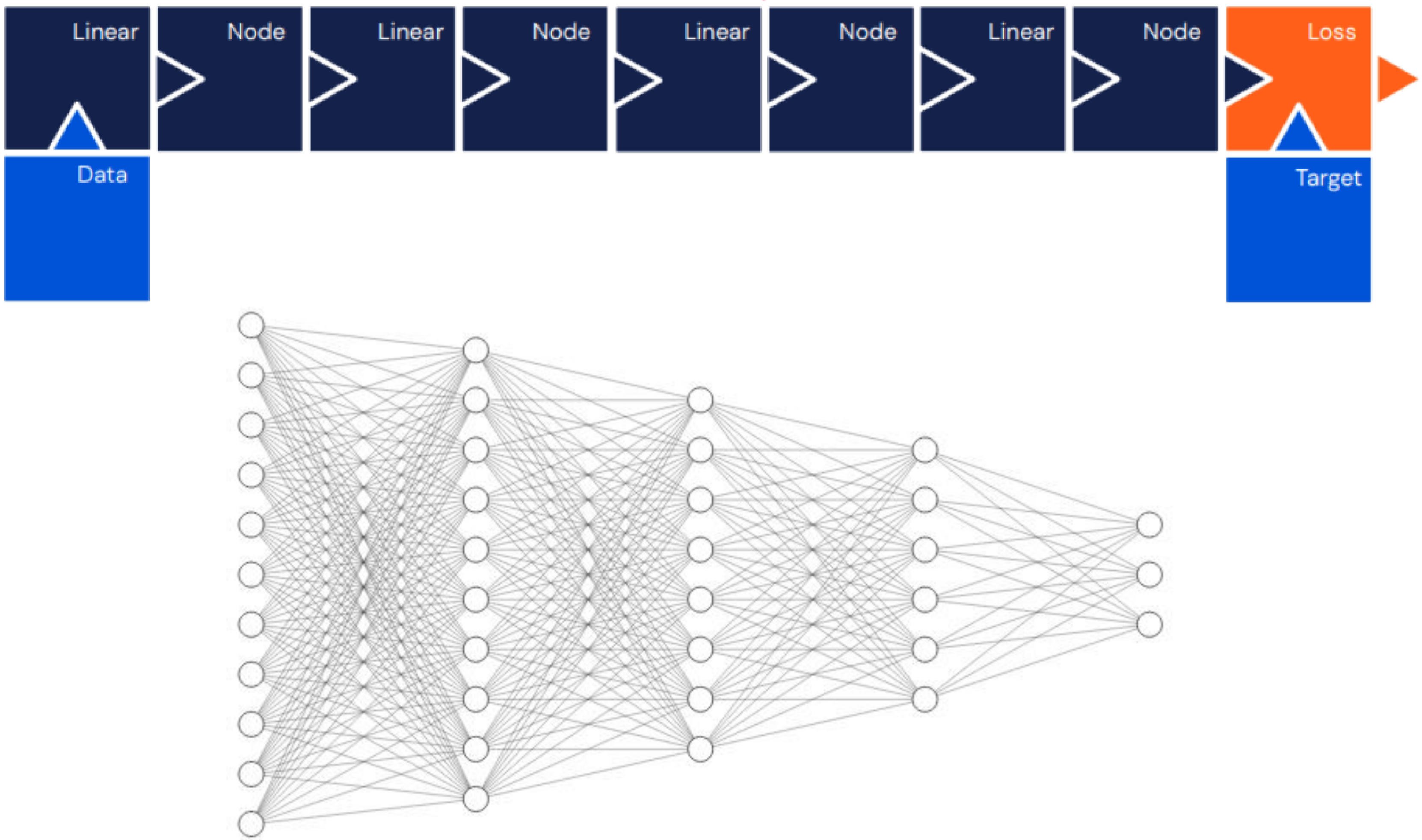
$$\mathbf{Z} = \mathbf{X}\mathbf{W}^\top + \mathbf{b}$$

Feed Forward Neural Networks



Feed Forward Neural Networks

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería



Pytorch

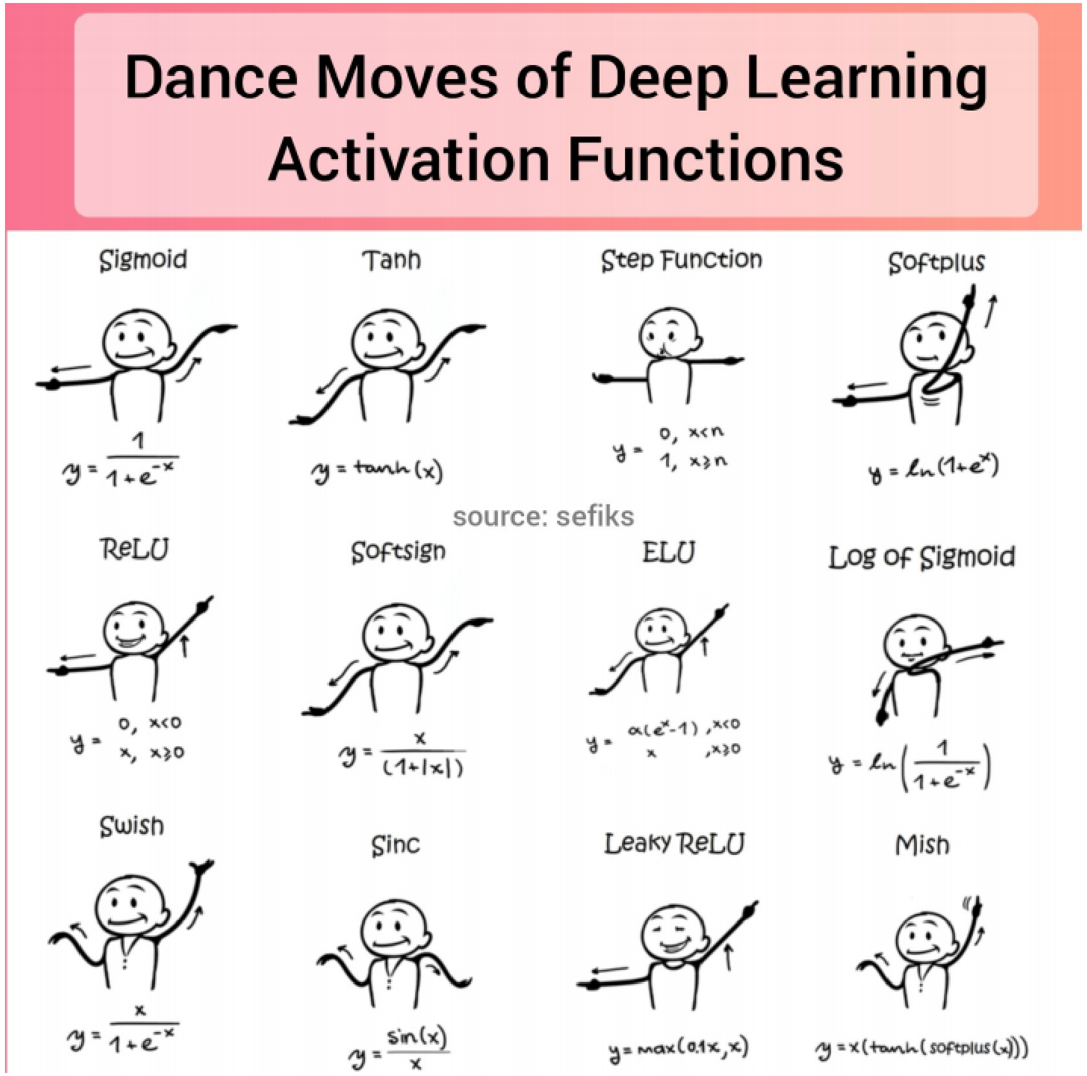
Universidad EAFIT · Escuela de Ciencias Aplicadas e Ingeniería

```python
import torch.nn.functional as F


class MultilayerPerceptron(torch.nn.Module):

    def __init__(self, num_features, num_classes):
        super(MultilayerPerceptron, self).__init__()

        ### 1st hidden layer
        self.linear_1 = torch.nn.Linear(num_features,
                                        num_hidden_1)

        ### 2nd hidden layer
        self.linear_2 = torch.nn.Linear(num_hidden_1,
                                        num_hidden_2)

        ### Output layer
        self.linear_out = torch.nn.Linear(num_hidden_2,
                                          num_classes)

    def forward(self, x):
        out = self.linear_1(x)
        out = F.relu(out)
        out = self.linear_2(out)
        out = F.relu(out)
        logits = self.linear_out(out)
        probas = F.log_softmax(logits, dim=1)
        return logits, probas
```

```python
class MultilayerPerceptron(torch.nn.Module):

    def __init__(self, num_features, num_classes):
        super(MultilayerPerceptron, self).__init__()

        self.my_network = torch.nn.Sequential(
            torch.nn.Linear(num_features, num_hidden_1),
            torch.nn.ReLU(),
            torch.nn.Linear(num_hidden_1, num_hidden_2),
            torch.nn.ReLU(),
            torch.nn.Linear(num_hidden_2, num_classes)
        )

    def forward(self, x):
        logits = self.my_network(x)
        probas = F.softmax(logits, dim=1)
        return logits, probas
```

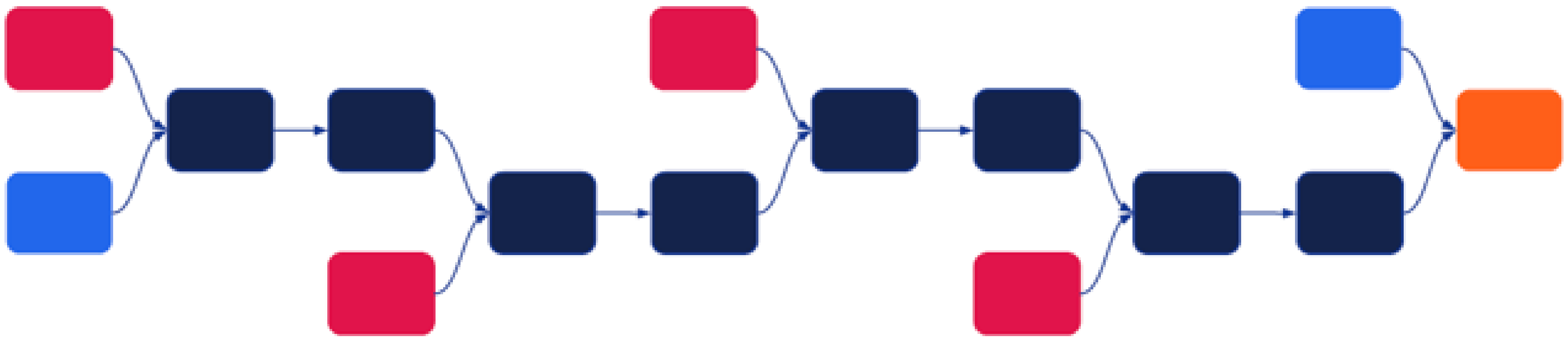
Más funciones de activación

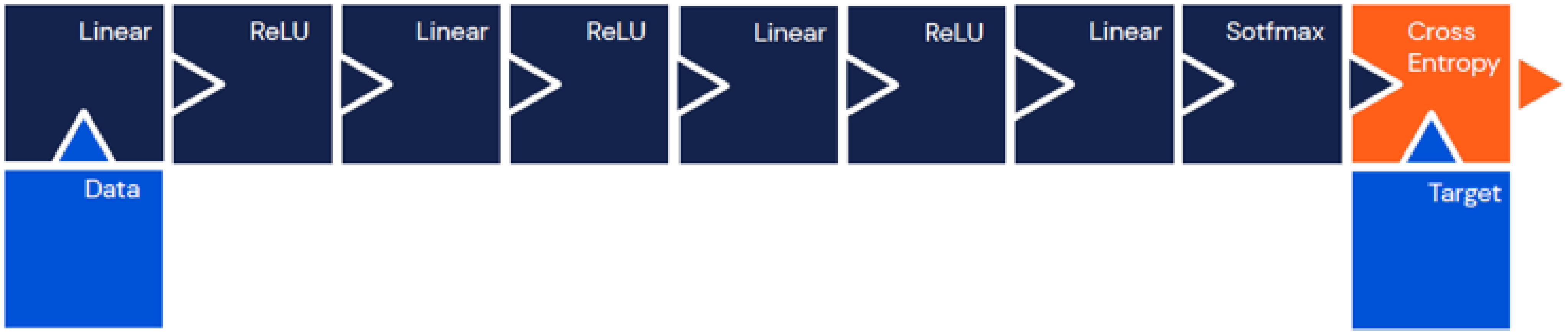
Escuela de Ciencias Aplicadas e Ingeniería



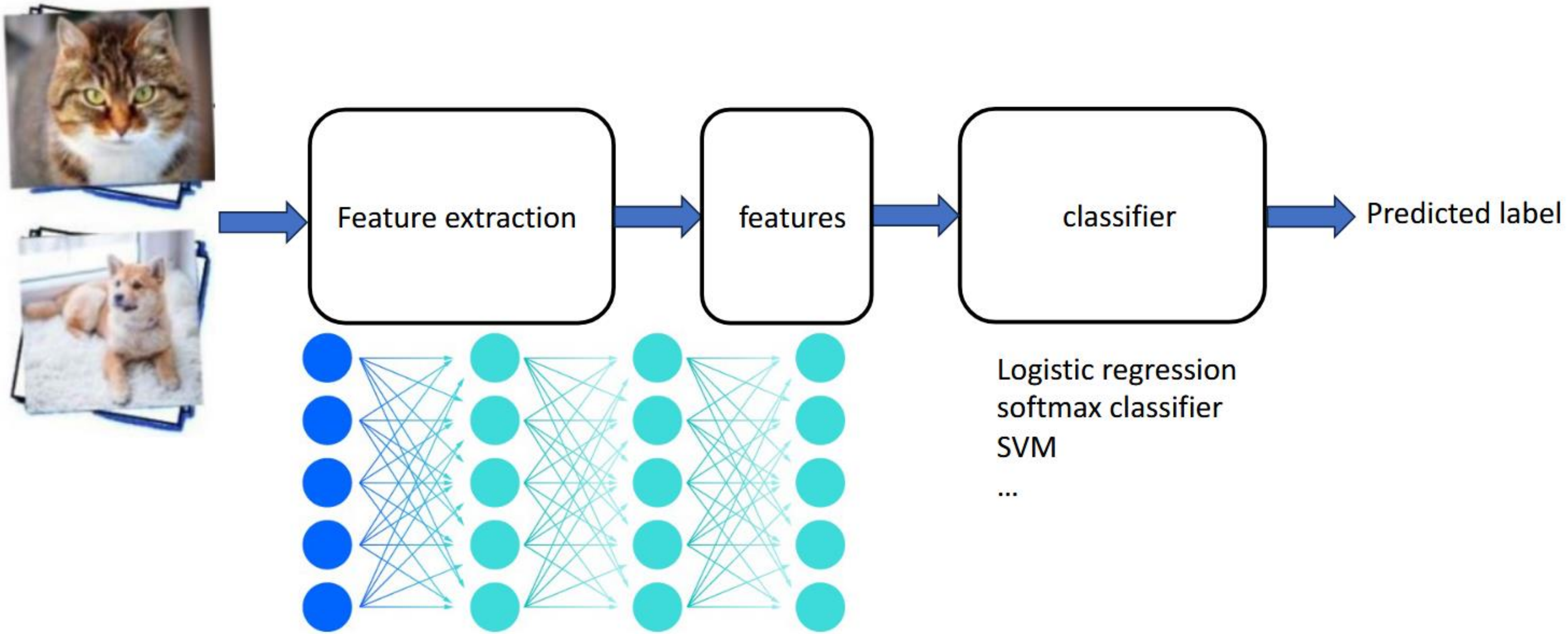
Grafo computacional

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería





Feature Extraction



Logistic regression
softmax classifier
SVM
...

Definición de Deep Learning

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

Yann LeCun
@ylecun

Some folks still seem confused about what deep learning is. Here is a definition:

DL is constructing networks of parameterized functional modules & training them from examples using gradient-based optimization....
facebook.com/722677142/post...

3:32 PM · Dec 24, 2019 · Facebook

517 Retweets **1.9K** Likes

Danilo J. Rezende
@DeepSpiker

Rephrasing @ylecun with my own words: DL is a collection of tools to build complex modular differentiable functions. These tools are devoid of meaning, it is pointless to discuss what DL can or cannot do. What gives meaning to it is how it is trained and how the data is fed to it

3:43 PM · Dec 25, 2019 · Twitter for iPhone

90 Retweets **464** Likes

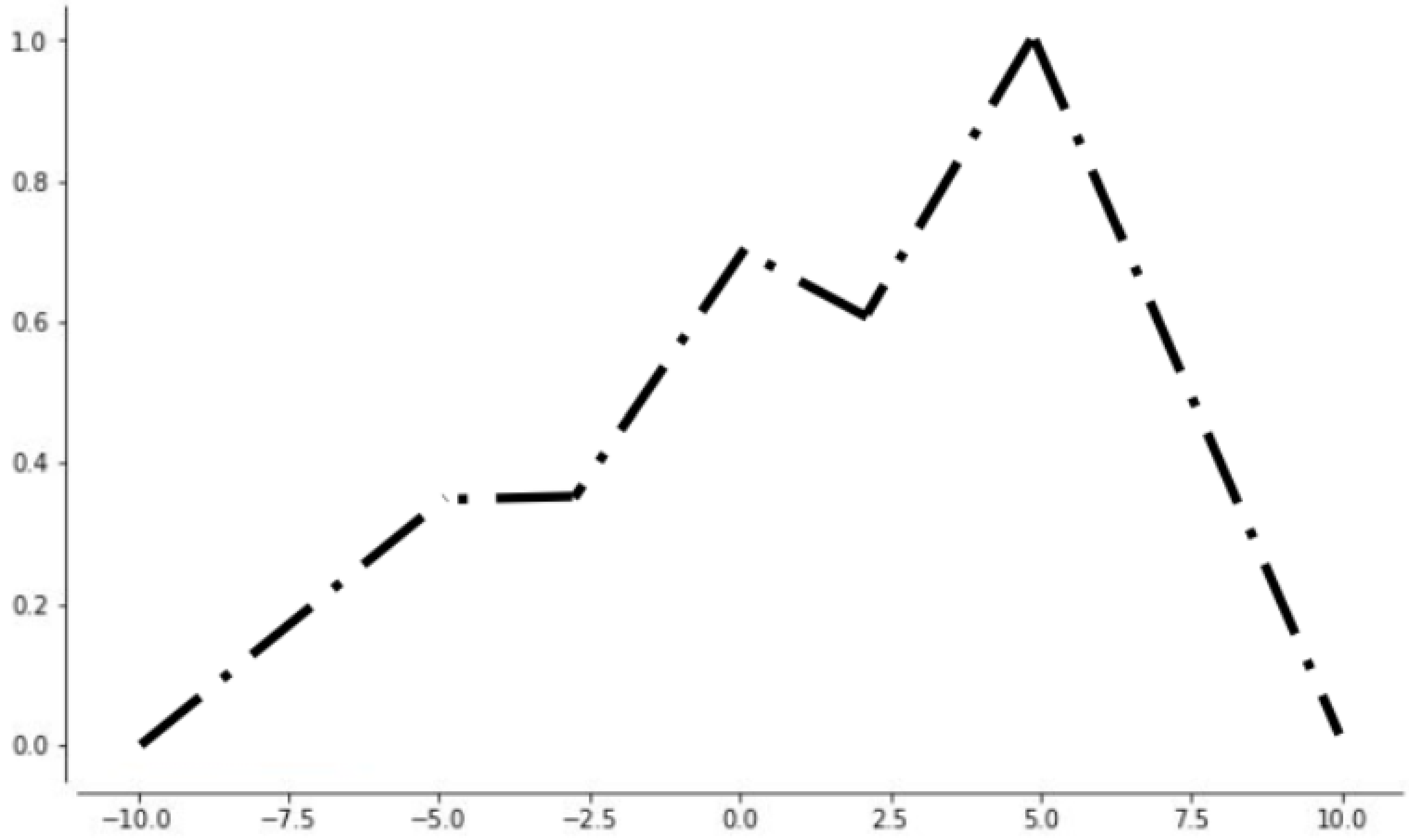
Teorema de aproximación universal

Para cualquier función continua desde un hipercubo $[0,1]^d$ a números reales, una función de activación f que sea no constante, acotada y continua, y para cada ϵ positivo, existe una red neuronal con una capa oculta usando f que obtiene como máximo un error ϵ en el espacio funcional.

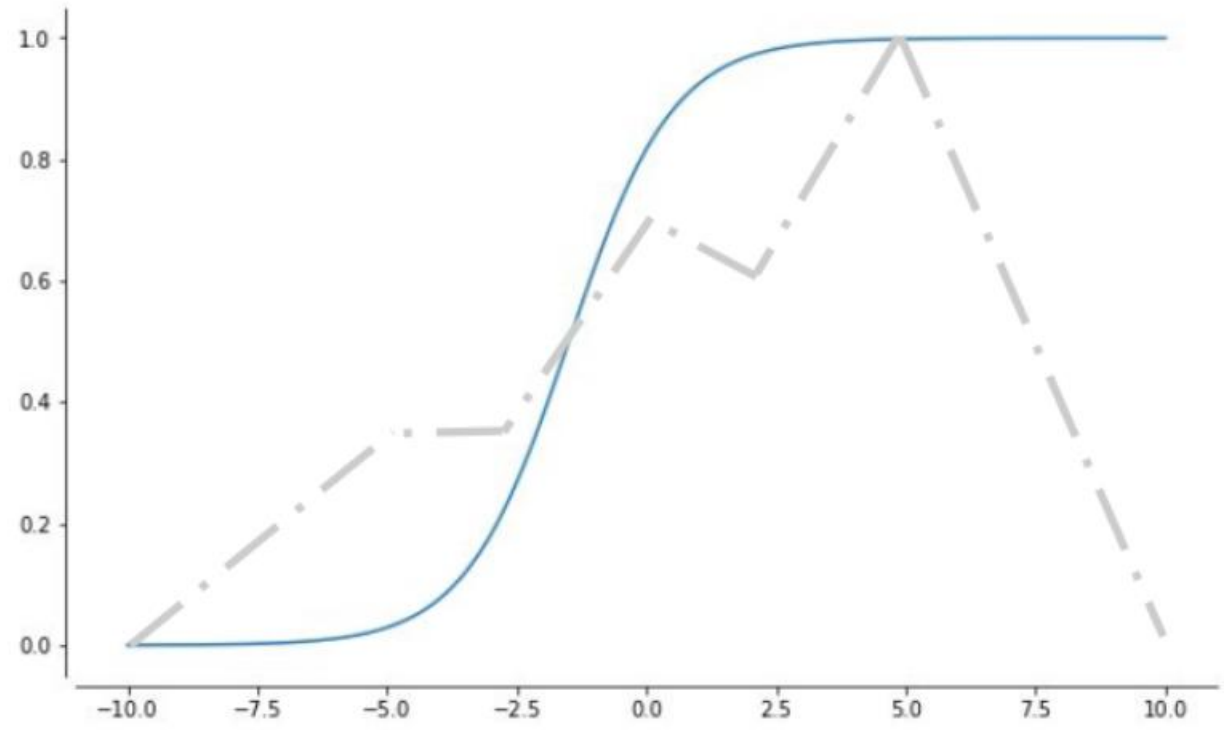
Teorema de aproximación universal

Para cualquier función continua desde un hipercubo $[0, 1]^d$ a números reales, una función de activación f que sea no constante, acotada y continua, y para cada ϵ positivo, existe una red neuronal con una capa oculta usando f que obtiene como máximo un error ϵ en el espacio funcional.

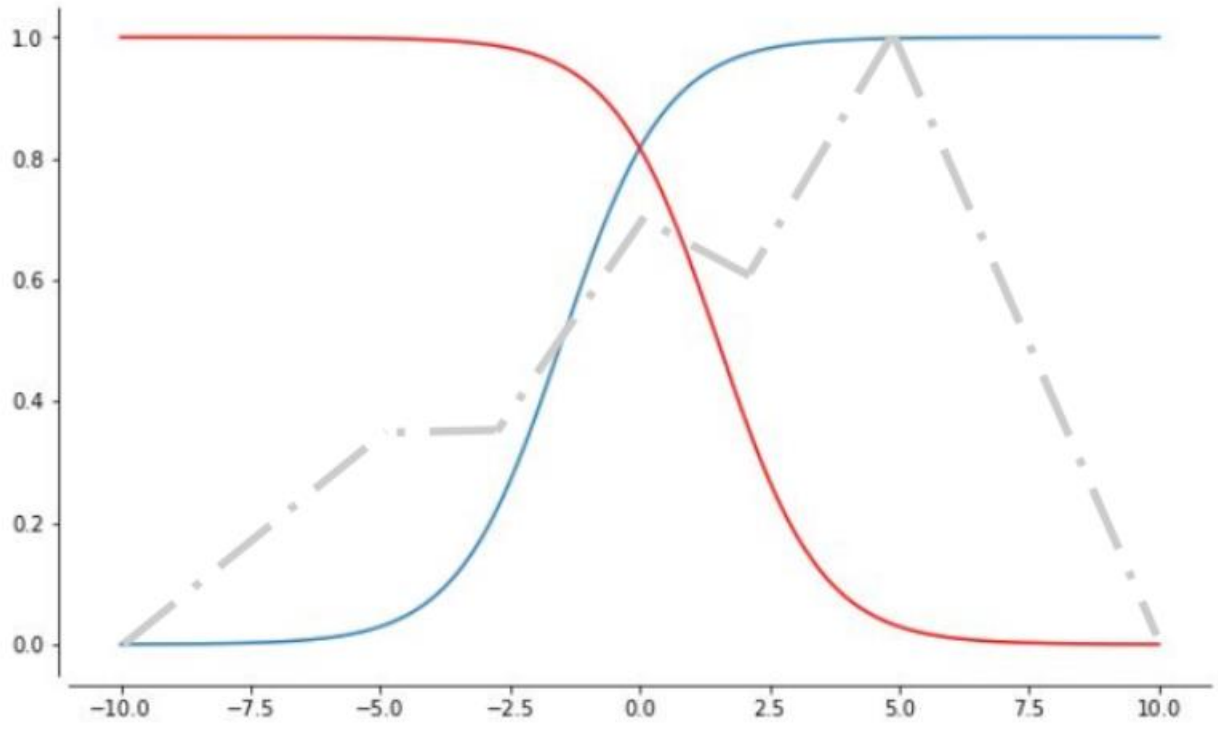
Teorema de aproximación universal



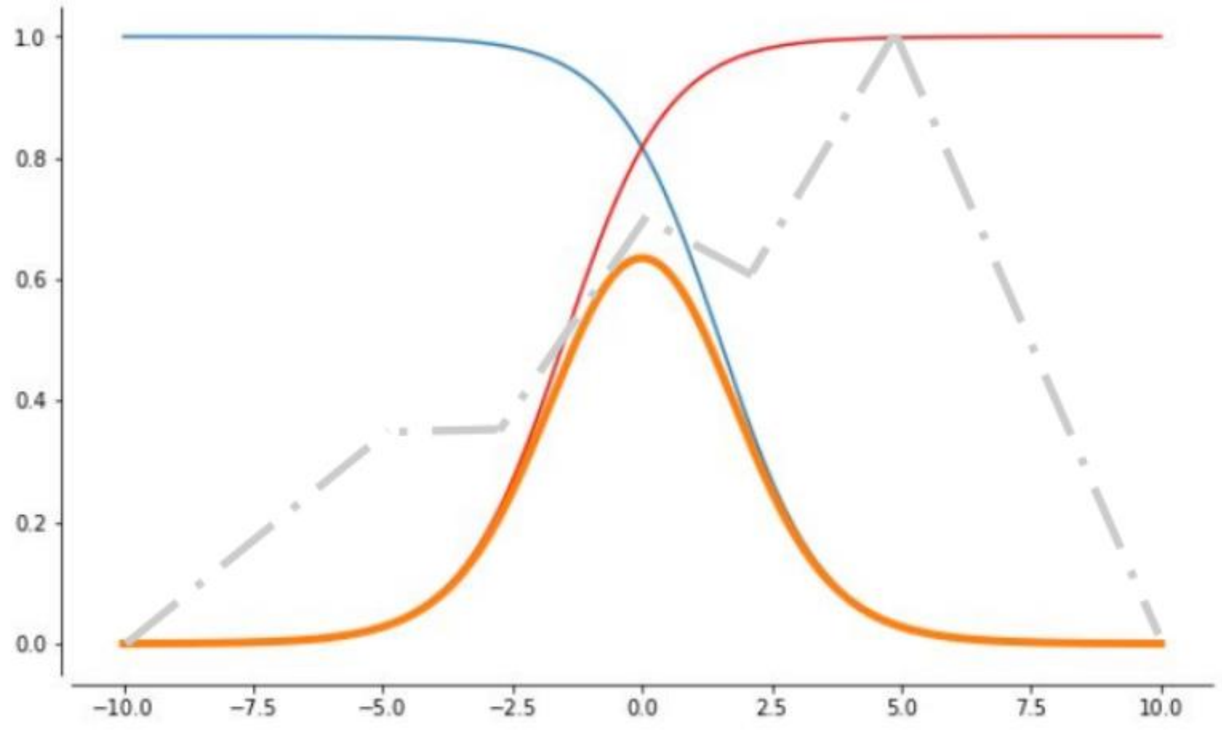
Teorema de aproximación universal



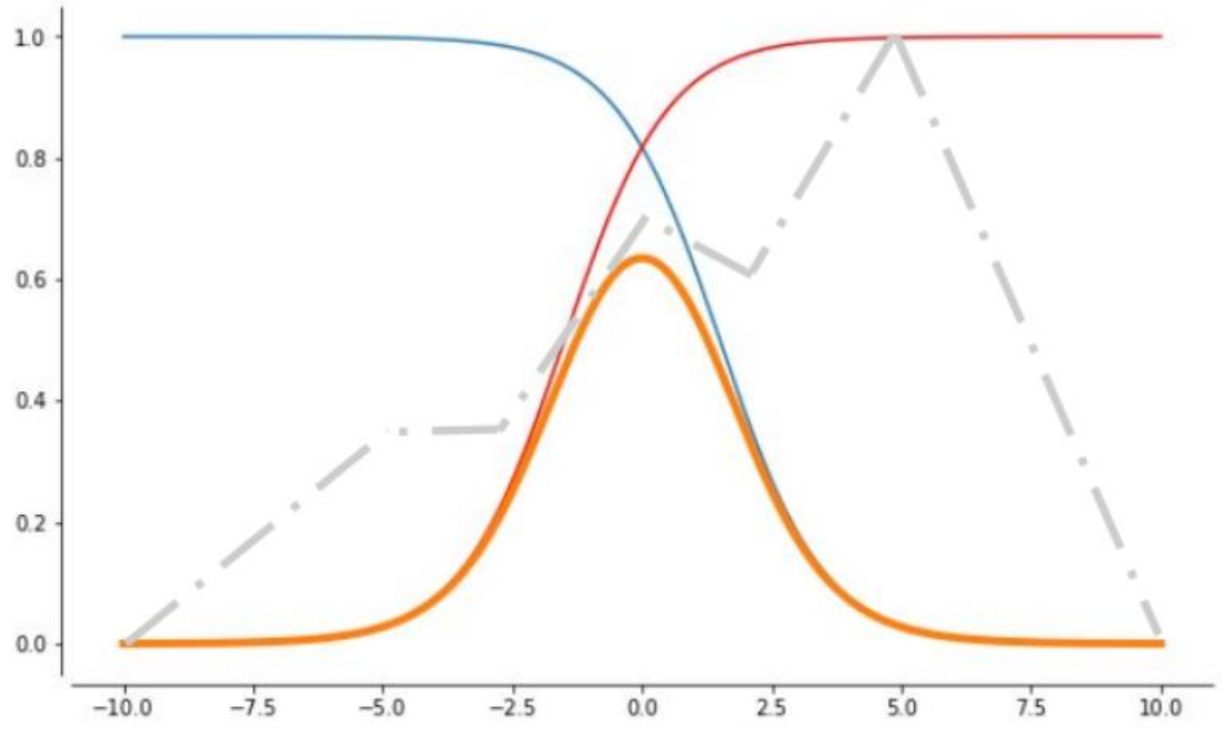
Teorema de aproximación universal



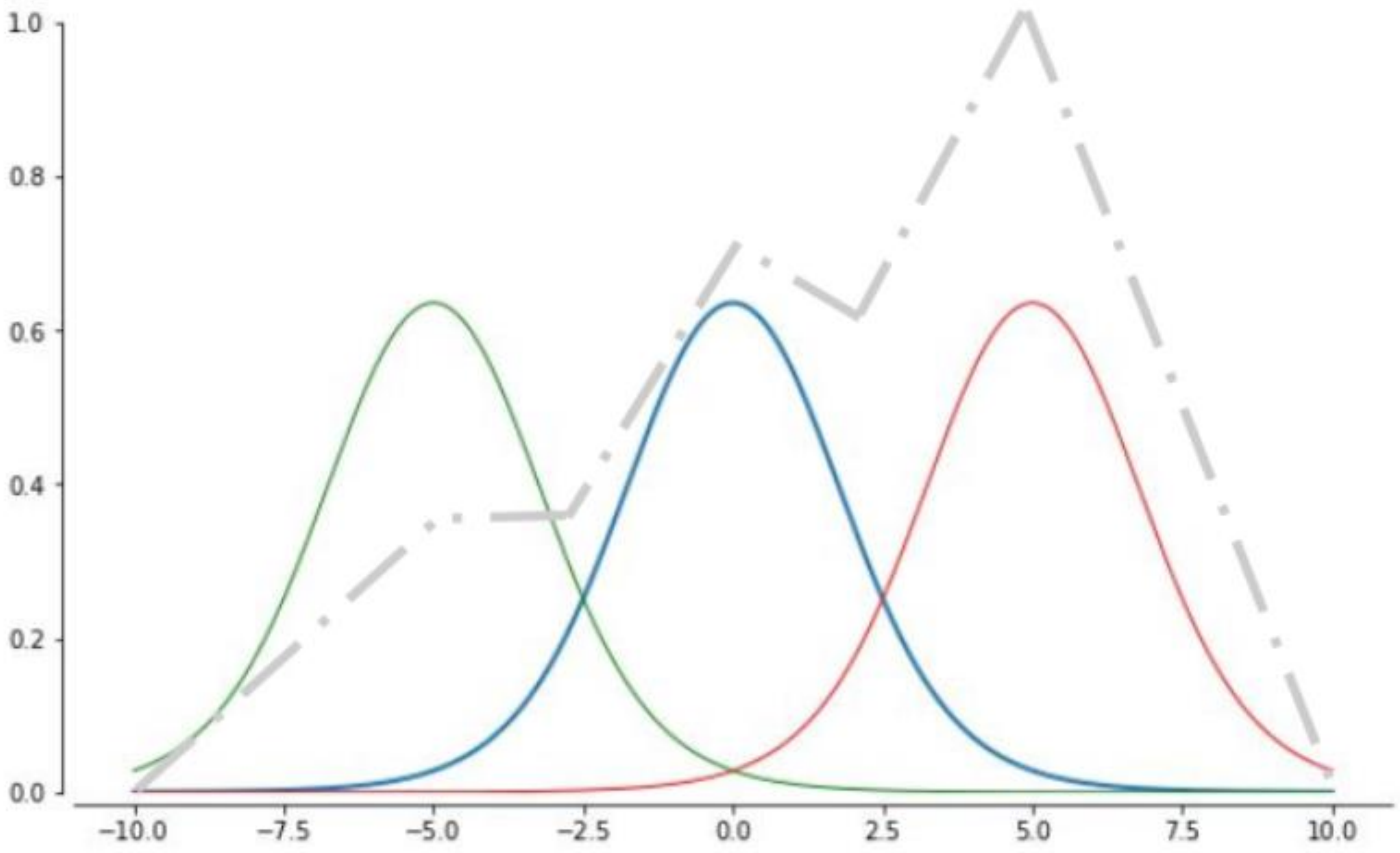
Teorema de aproximación universal



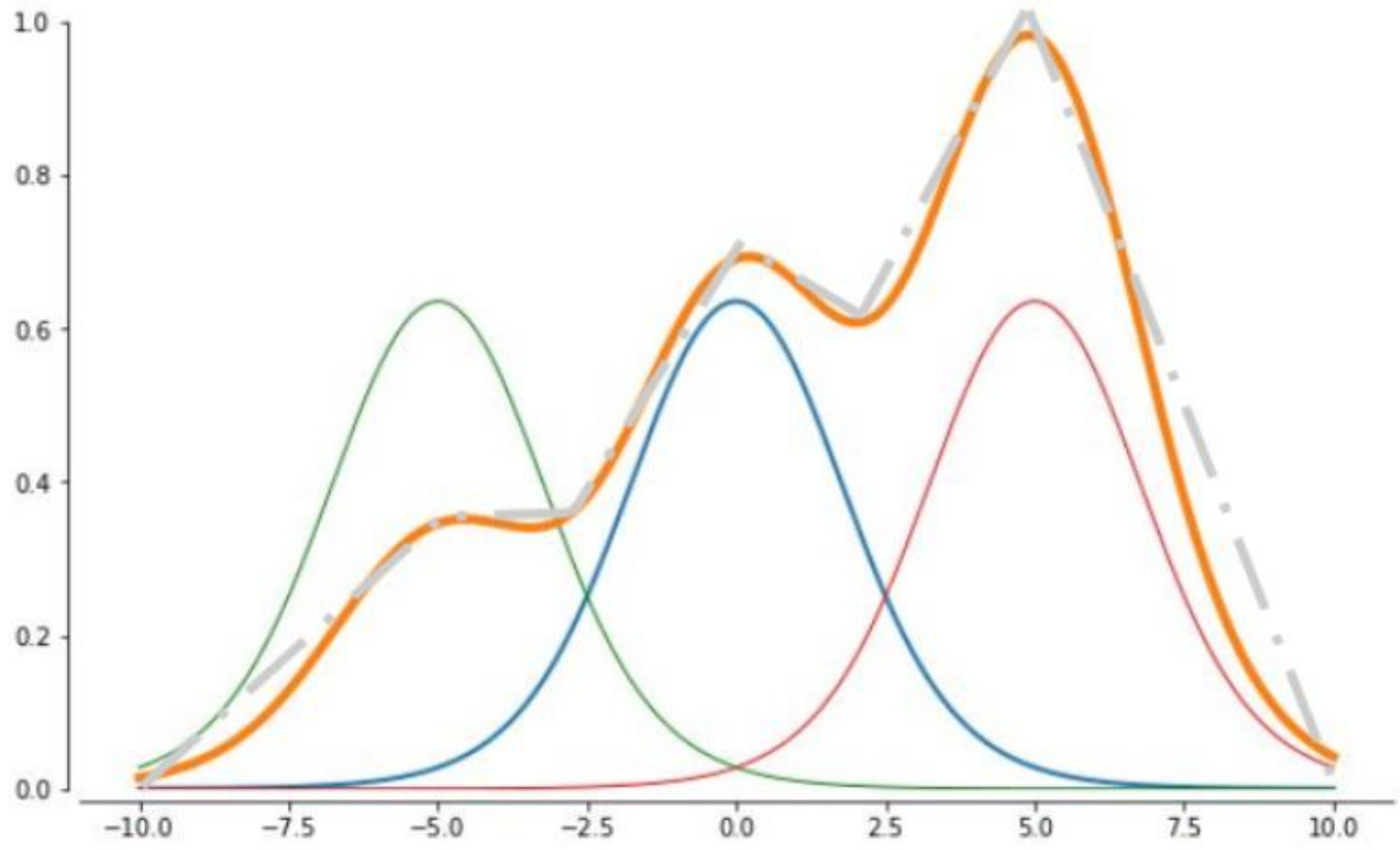
Teorema de aproximación universal



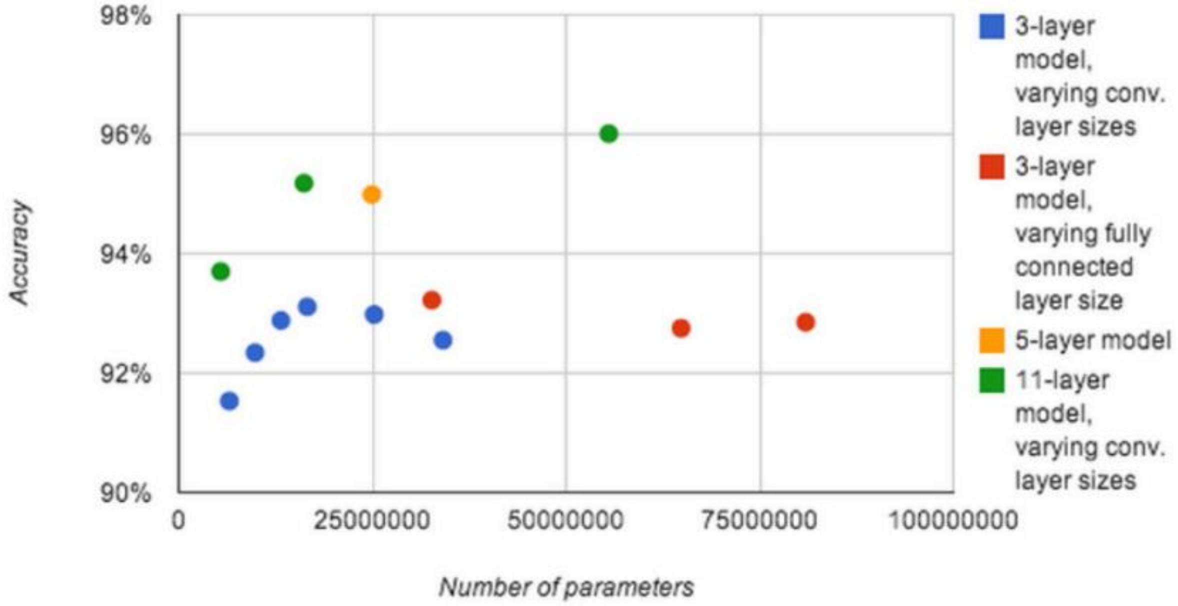
Teorema de aproximación universal



Teorema de aproximación universal



Arquitecturas Anchas vs Profundas



Increasing the number of parameters in shallower models does not allow such models to reach the same level of performance as deep models, primarily due to overfitting.

Plot from [Goodfellow et. al, 2014]

- (linear) Regions grows exponentially w.r.t depth
- Deeper model use fewer parameter to achieve necessary number of (linear) regions

Arquitecturas Anchas vs Profundas

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Pueden alcanzar la misma expresividad con más capas pero menos parámetros (combinatorias); menos parámetros $\rightarrow$ menos sobreajuste
- Además, teniendo más capas provee cierta forma de regularización: capas posteriores están restringidas por el comportamiento de capas anteriores
- Sin embargo, más capas $\rightarrow$ gradientes que explotan/desaparecen
- Después: diferentes capas para diferentes niveles de abstracción (DL es en realidad más sobre aprendizaje de funciones más que simplemente apilar multiples capas)

Arquitecturas Anchas vs Profundas

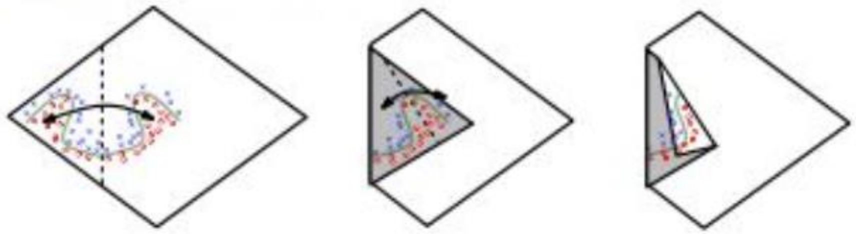


Figure 3: Space folding of 2-D space in a non-trivial way. Note how the folding can potentially identify symmetries in the boundary that it needs to learn.

Number of **linear regions** grows **exponentially** with **depth**, and **polynomially** with **width**.

[Submitted on 8 Feb 2014 (v1), last revised 7 Jun 2014 (this version, v2)]

On the Number of Linear Regions of Deep Neural Networks

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, Yoshua Bengio

We study the complexity of functions computable by deep feedforward neural networks with piecewise linear activations in terms of the symmetries and the number of linear regions that they have. Deep networks are able to sequentially map portions of each layer's input-space to the same output. In this way, deep models compute functions that react equally to complicated patterns of different inputs. The compositional structure of these functions enables them to re-use pieces of computation exponentially often in terms of the network's depth. This paper investigates the complexity of such compositional maps and contributes new theoretical results regarding the advantage of depth for neural networks with piecewise linear activation functions. In particular, our analysis is not specific to a single family of models, and as an example, we employ it for rectifier and maxout networks. We improve complexity bounds from pre-existing work and investigate the behavior of units in higher layers.

Por qué Deep Learning?

- **Razón #1:** Grandes cantidades de datos
- **Razón #2:** Recursos computacionales (GPUs)
- **Razón #3:** Modelos grandes fáciles de entrenar
- **Razón #4:** Las redes neuronales se pueden ver como piezas de lego

Optimización

Objetivo: minimizar la función de pérdida $\mathcal{L}(\mathbf{w})$

Algoritmo:

1. Inicializar los pesos: $\mathbf{w}^{(0)} \sim$ aleatorio
2. Para cada iteración $t = 0, 1, 2, \ldots$:
   - Calcular el gradiente:
     $$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^{(t)})$$
   - Actualizar los pesos:
     $$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^{(t)})$$
     donde $\eta > 0$ es la tasa de aprendizaje.
3. Repetir hasta convergencia