

Deep Learning (I)

BASIC CONCEPTS

SILVERIO GARCIA CORTES

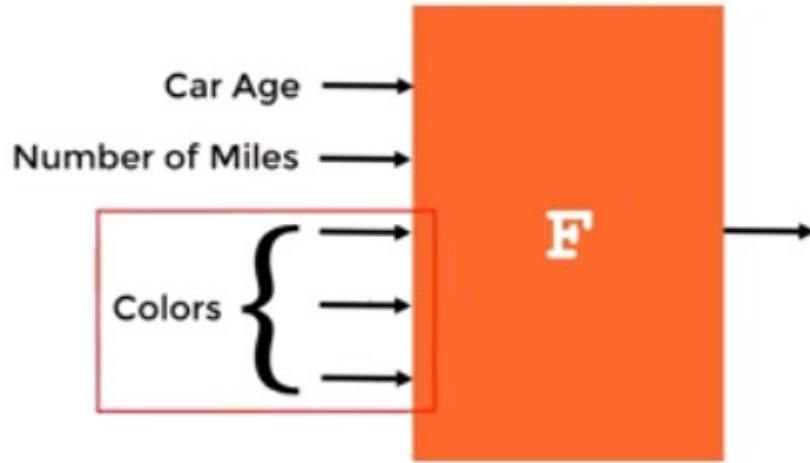
UNIVERSIDAD DE OVIEDO

NAPLES. MAY 2023, UNIV. FEDERICO II



Problem statement, Feature matrix and Normalization

Used car value prediction problem:



AGE	MILES	COLOR
3	100000	Red
5	60000	Blue
7	120000	White

Feature matrix

PRICE
5000
4500
3000

Ground truth

Categorical feature codification: ONE SHOT

- Preserves equality distance between different classes (as opposed to 0,1,2)

Numerical feature Normalization

- Avoids one feature dominates over others because its magnitude

Feature matrix

AGE	MILES	COLOR
3	100000	1,0,0
5	60000	0,1,0
7	120000	0,0,1

Ground truth

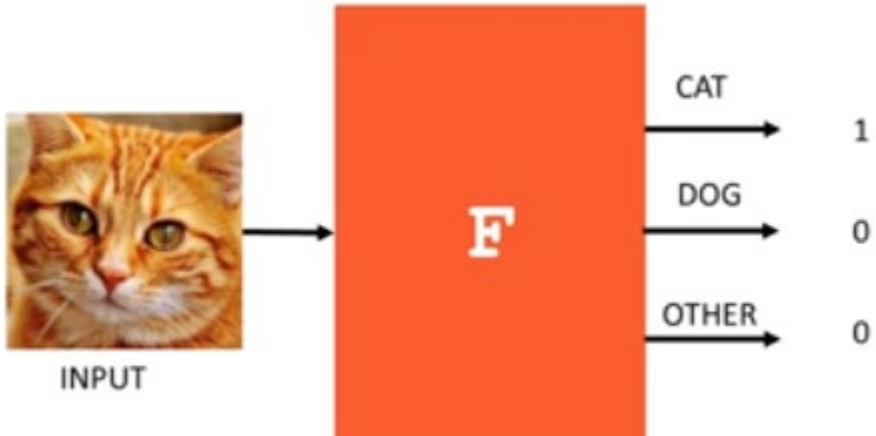
PRICE
5000
4500
3000

$$x_1^{scaled} = \frac{x_1 - \min_{x_1}}{\max_{x_1} - \min_{x_1}}$$

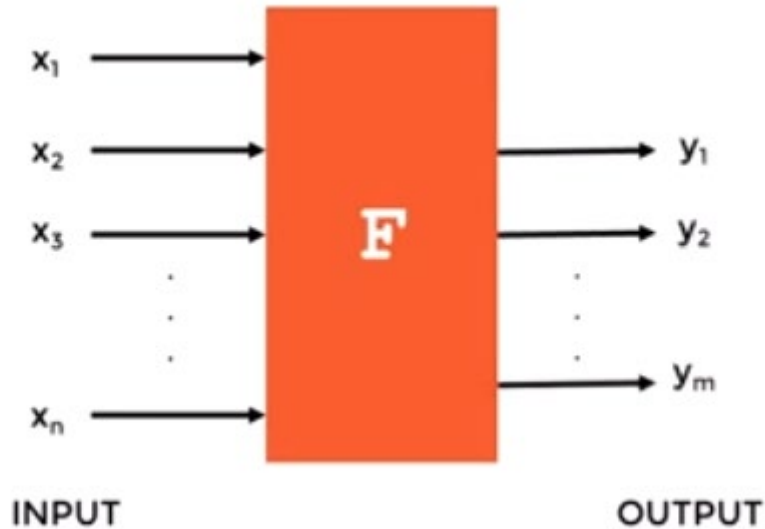
$$x_1 = \frac{x_1 - \mu_1}{\sigma_1}$$

Different Problem statements:

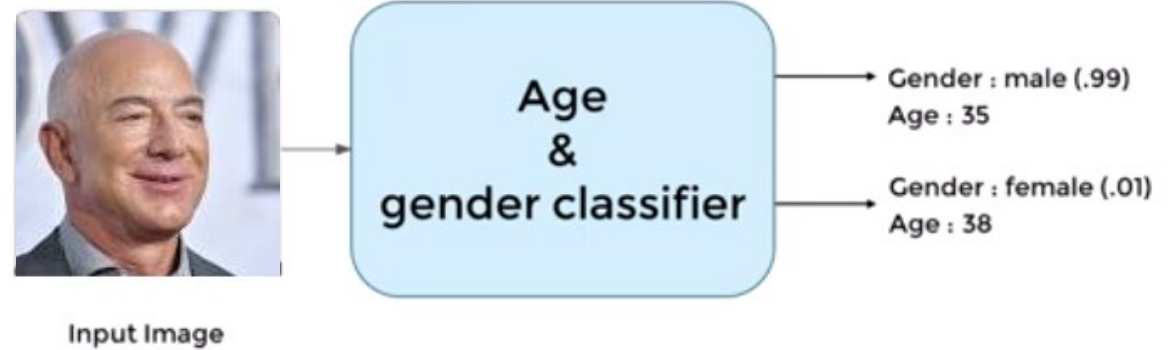
Instance Multiclass Classification problem



- Input data should be converted to feature vectors (matrix)
- If the problem is a classification the output value will be also one hot coded
- Categorical features should be one hot encoded
- Numerical features must be normalized
- Also problems with several outputs are possible

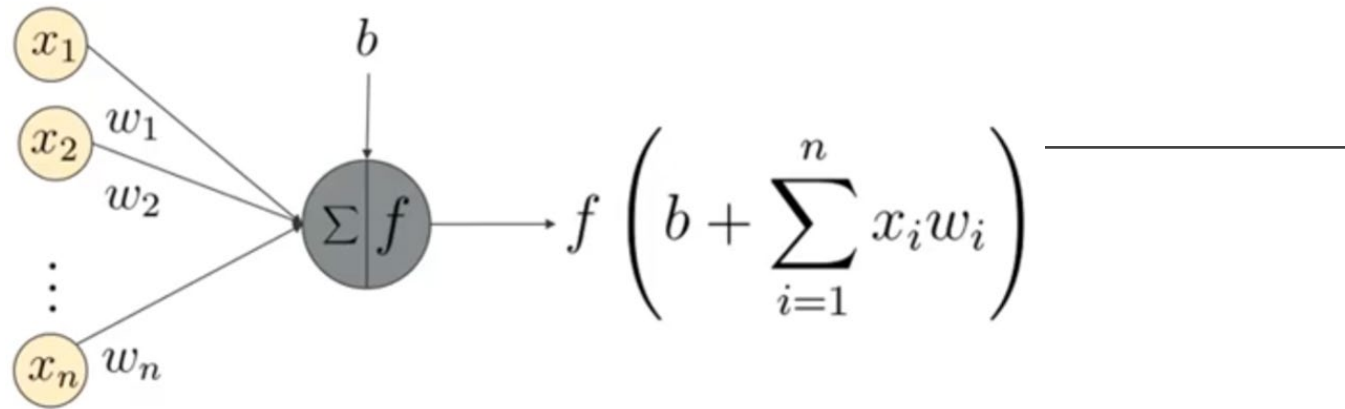


Multi Regression problem



Multi Regression problema with images as input data

Neural Network basic component: Artificial Neuron. Perceptron



Input

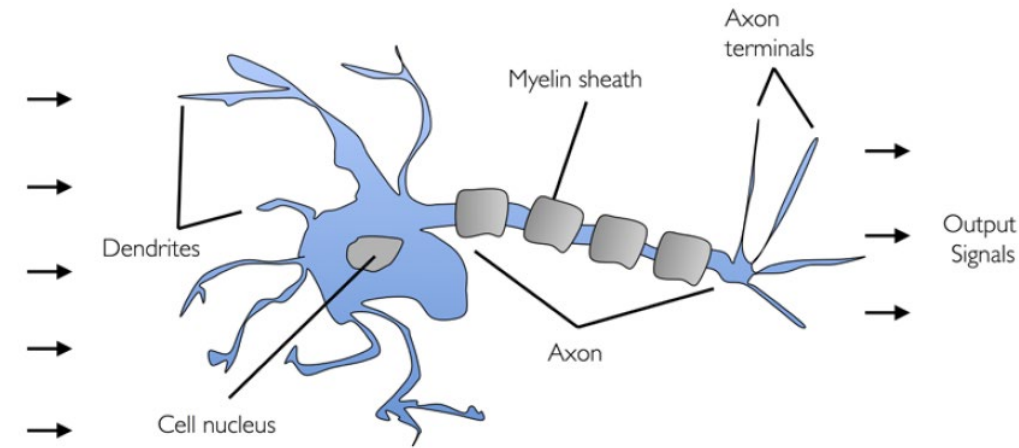
- An artificial neuron artificial is a model of Machine Learning inspired in biological neural networks.
- It can model relationships between inputs and outputs
- The Perceptron is the unit of neural networks and was initially built as a binary classifier similar to a bio neuror

$$z = w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_m \cdot x_m = \sum_{j=0}^m \mathbf{x}_j \cdot \mathbf{w}_j$$

$$= \mathbf{w}^T \cdot \mathbf{x}$$

- A perceptron receives a number of inputs (x_i) which are combined with weights and bias (one per neuron) to obtain the “net input” z .
- Finally an activation function is applied to the “ z ” value to obtain the final output $\phi(z)$.

Perceptron Model

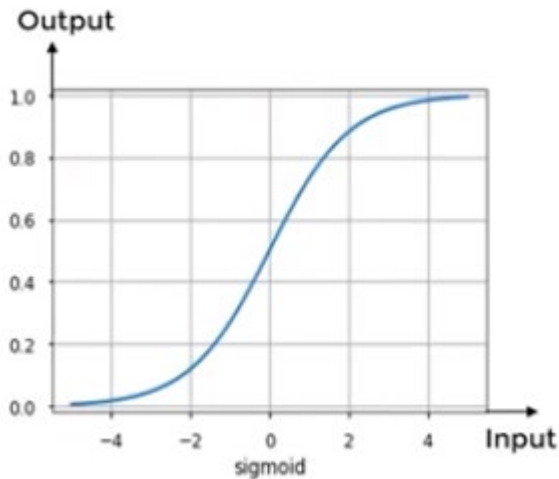


Activation Function Example

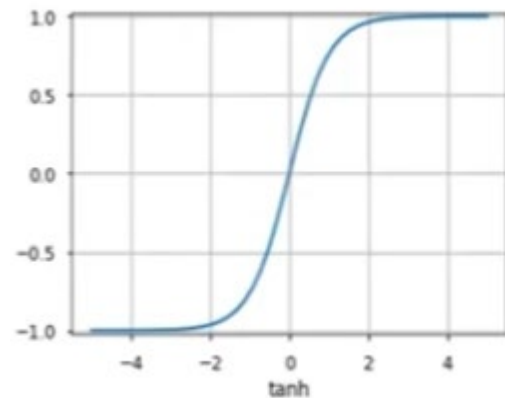
$$\phi(\mathbf{z}) = \begin{cases} 1 & \mathbf{z} \geq 0 \\ -1 & \mathbf{z} < 0 \end{cases}$$

Activation functions

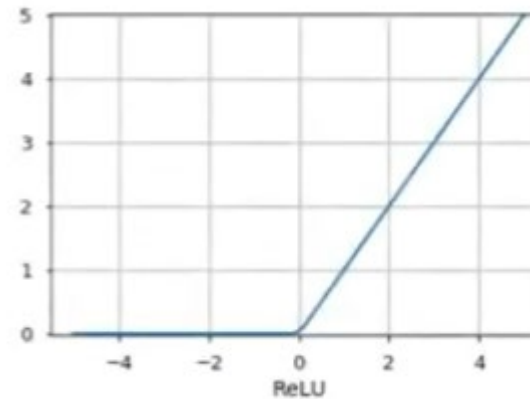
Commonly activation functions are non-linear. In this way the NN can model non linear relations between inputs and outputs. Some common activation functions are:



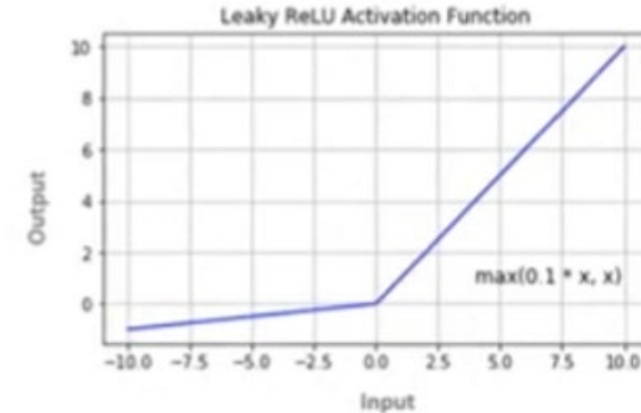
Sigmoid: Logistic Regression:
 $[-\infty, \infty] \rightarrow [0, 1]$



tanh: Logistic Regression:
 $[-\infty, \infty] \rightarrow [-1, 1]$



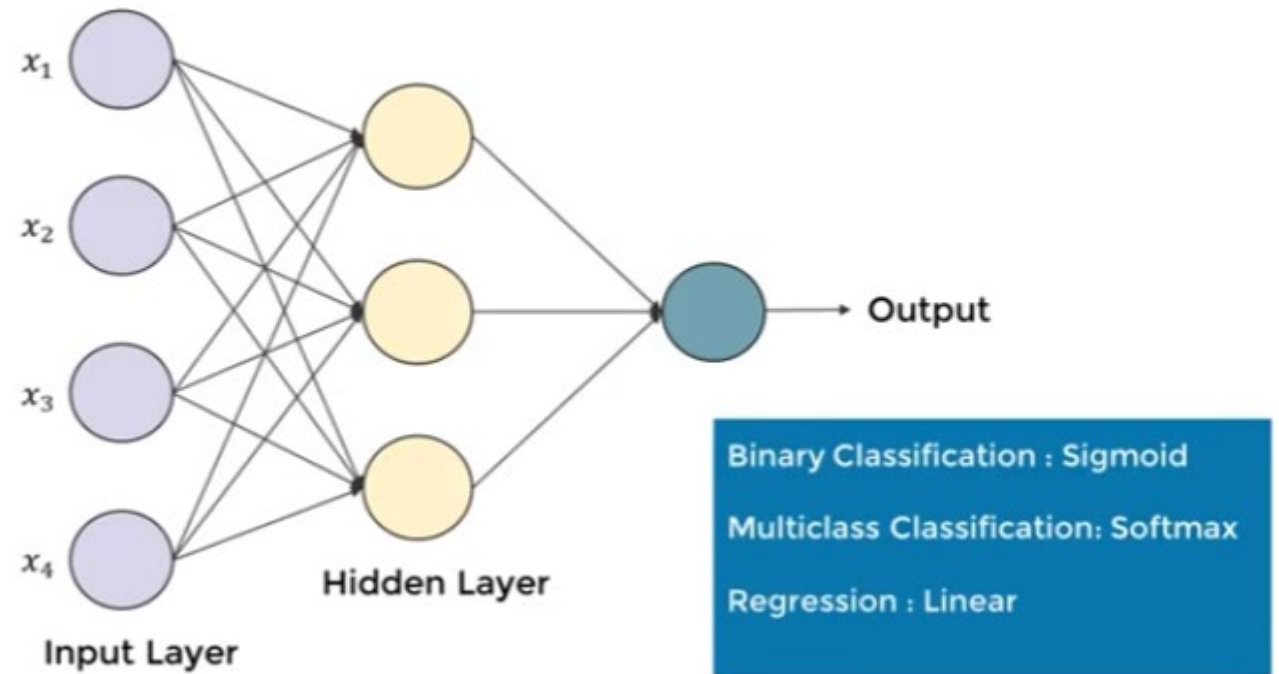
ReLU: Rectified linear unit:
 $[-\infty, \infty] \rightarrow [0, \infty]$



Leaky ReLU:
Piecewise
Avoids dying gradient

From Perceptron to Neural Networks (NN)

- A perceptron is combined in layers.
- Each neuron receives all the inputs (features)
- And has its own weights and bias
- The goal of the NN is to output the correct value for the input features (x_1, x_2, x_3, \dots)

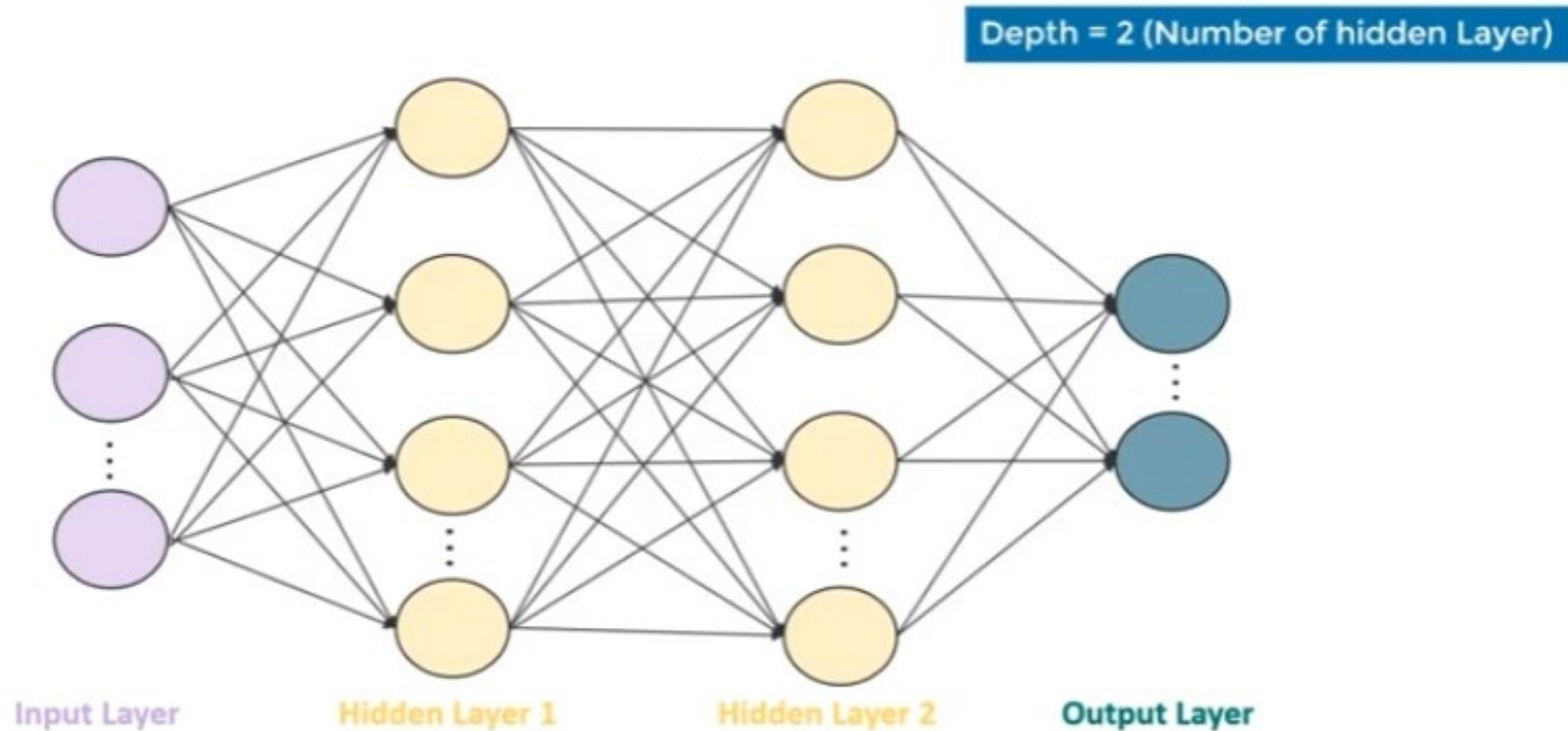


Different problems need different activation functions in the output layer

NN depth

A network can model more complex relationships the deeper it is.

A hidden layer which all its neurons connected to the neurons from the previous layer (or the following) is called a **DENSE** layer



How does a NN learns? Error and Cost function

- At the output of the NN the error between the predicted value and the ground truth value is calculated
- The total mean error is calculated. For example, Means Square error (no outliers) or Meand absolute error (tolerate better outliers)

Objective Function : Cost Function

Error

$$e_i = y_i - \hat{y}_i$$

True value

Predicted value

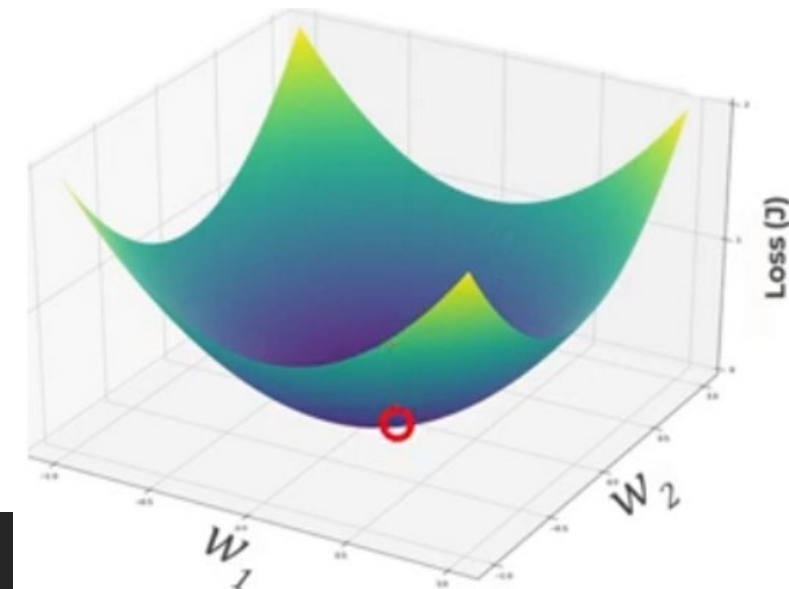
MEAN SQUARED ERROR

$$L = \frac{1}{n} \sum_{i=1}^n ||y_i - \hat{y}_i||^2$$

MEAN ABSOLUTE ERROR

$$L = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Average error per data point

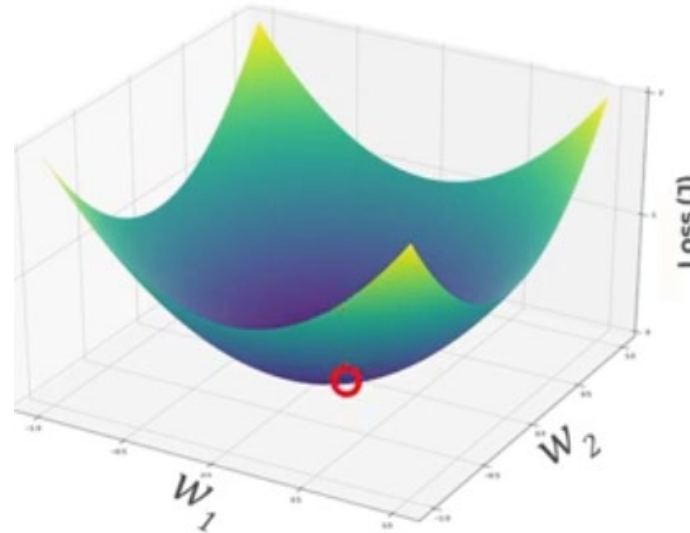
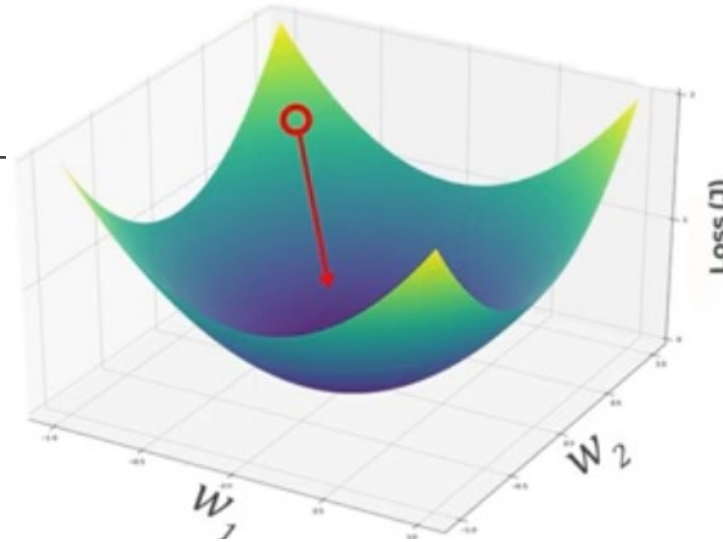


- Total mean error L must be minimized
- It mathematically depends on weights and bias of each neuron
- MAE is more used if gross errors are present y data (because its minor impact in L with respect to MSE)

How does a NN learns? Gradient descent and Backpropagation

- Corrections for weights can be calculated if we know the slope of the cost function
- The slope of the cost function is the gradient.
- We will proceed iteratively correcting the weights
- We modulate the gradient with a factor called **learning rate** to avoid big steps
- To calculate the gradients we will use the chain rule:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \times \frac{\partial y}{\partial x}$$



$$w_1^{(1)} = w_1^{(0)} - \lambda \frac{\partial J}{\partial w_1}$$

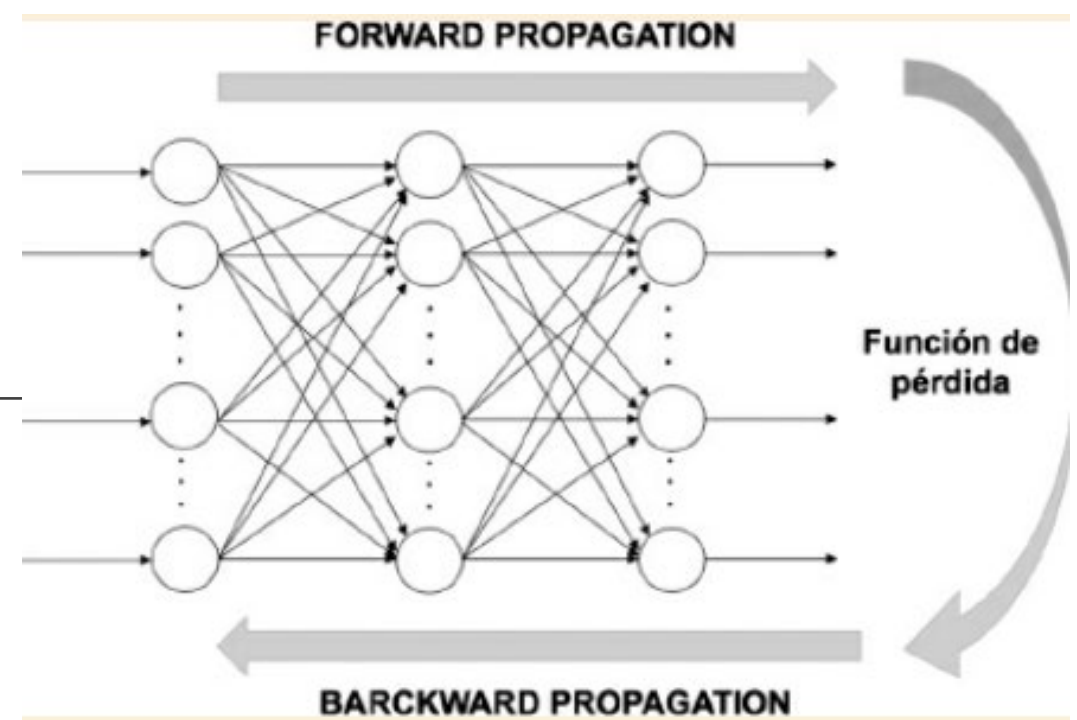
$$w_2^{(1)} = w_2^{(0)} - \lambda \frac{\partial J}{\partial w_2}$$

↑
Learning Rate

$$w_1^{(t+1)} = w_1^{(t)} - \lambda \frac{\partial J}{\partial w_1}$$

$$w_2^{(t+1)} = w_2^{(t)} - \lambda \frac{\partial J}{\partial w_2}$$

Backpropagation

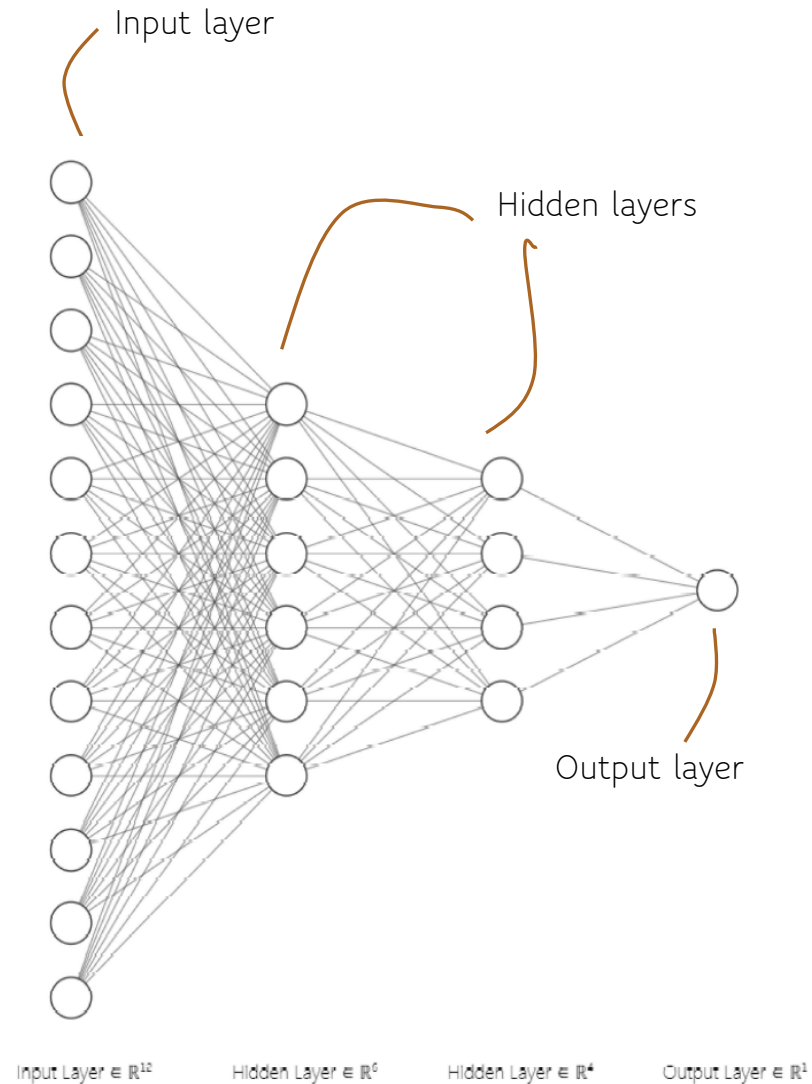


1. Start with random weights.
2. Select a small batch of images from the traning set.
3. Using forward propagation, calculate loss for the current weights.
4. Calculate gradients using back propagation.
5. Perform Gradient Descent to update weights.
6. Repeat until convergence.

Multilayer Perceptron (MLP)




- MLP (Multilayer Perceptron) is composed of dense layers where each neuron of one layer is connected with all the neurons from the previous one and the following one
- The MLP needs to have at least one hidden layer different from the input and output layers
- Activation functions introduce the capacity to model non linear relations in the NN.
- Non linear activations are needed to model non linear relations between features and outputs
- It has been shown that certain activation functions with at least one hidden layer can approximate any function if the number of neurons is high enough

NN's are then Universal approximators



TensorFlow Playground

- [Tensorflow Playground](#)



Epoch
000,000

Learning rate
0.03

Activation
Tanh

Regularization
None

Regularization rate
0

Problem type
Classification

DATA

Which dataset do you want to use?



Ratio of training to test data: 50%



Noise: 0



Batch size: 10



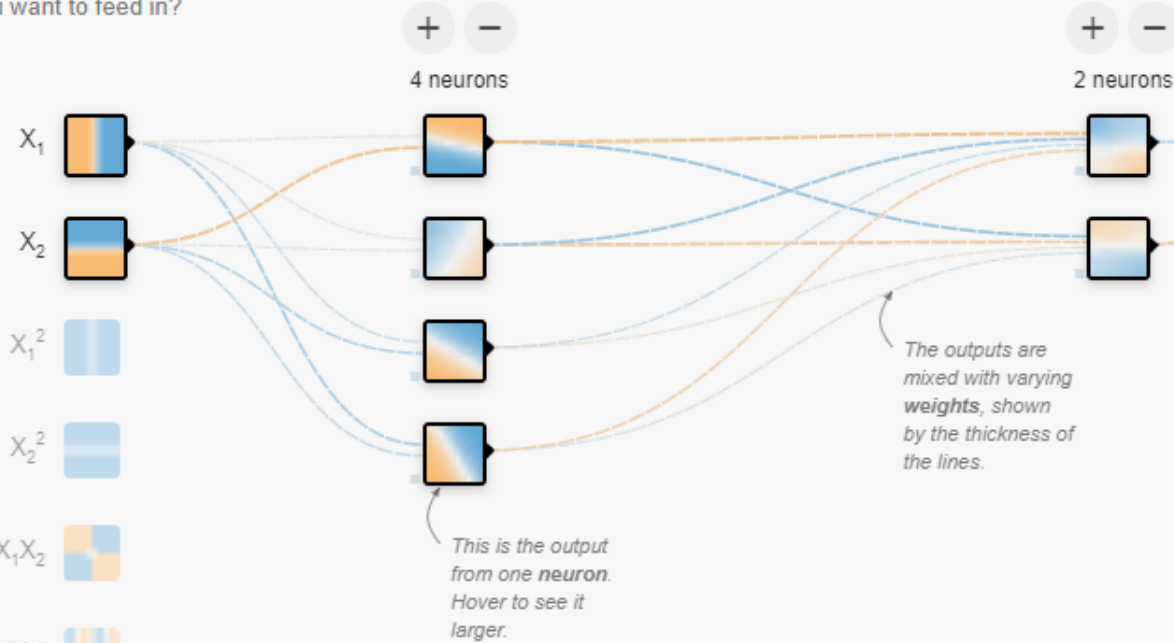
REGENERATE

FEATURES

Which properties do you want to feed in?



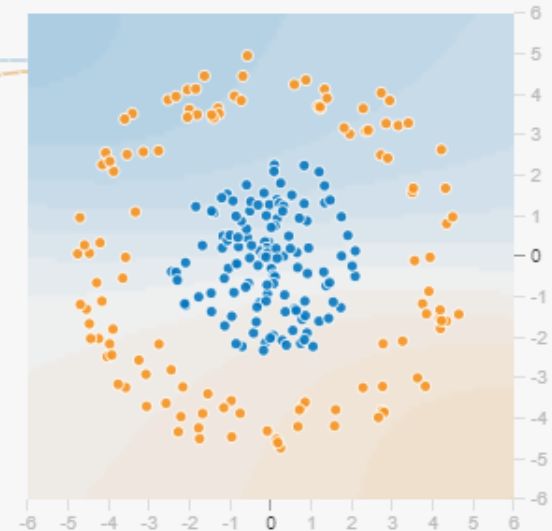
+ - 2 HIDDEN LAYERS



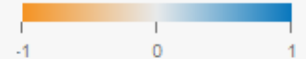
OUTPUT

Test loss 0.527

Training loss 0.521



Colors shows data, neuron and weight values.



☐ Show test data

☐ Discretize output

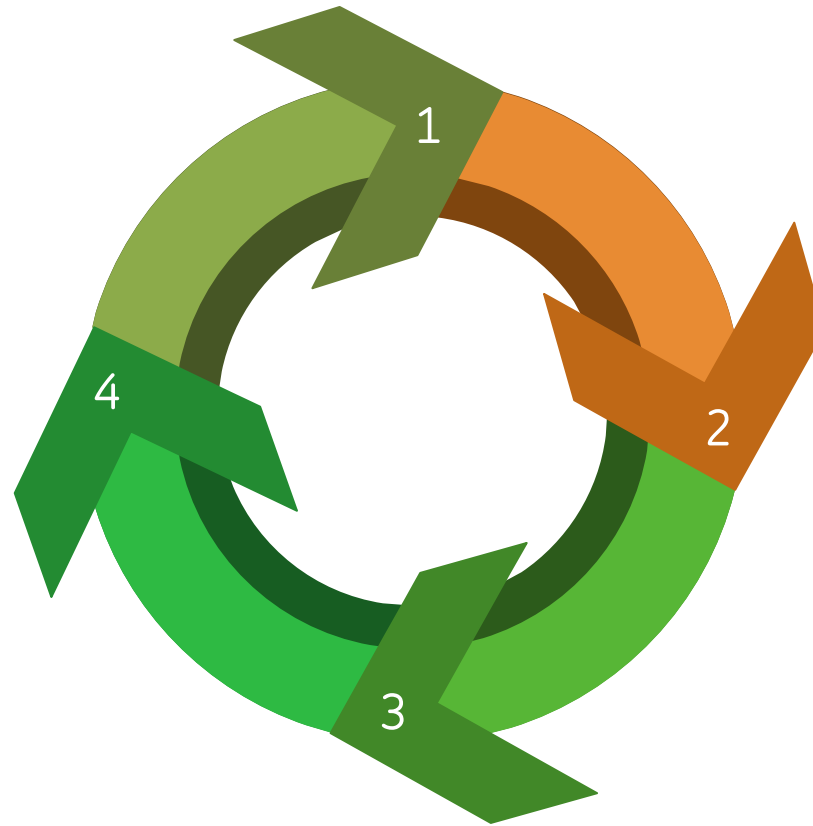
Tensor Playground: Things to see

01 Single Layer Perceptron (SLP)

A single layer perceptron model can only solve linear problems.

04 MLP + Non linear activation

An MLP with non-linear activation function can solve many non-linear problems with only the original features as input.



02 Feature Engineering

Feature Engineering can solve non linear problems with only SLP but probably you will need to change the features for different problems. This approach lacks generalization .

03 MLP + Linear activation

A multi layer Perceptron (MLP) is requires to solve non-linear problems, but Linear activation allows only to solve Linear problems even using MLP.