



## Inteligencia Artificial - IAI84

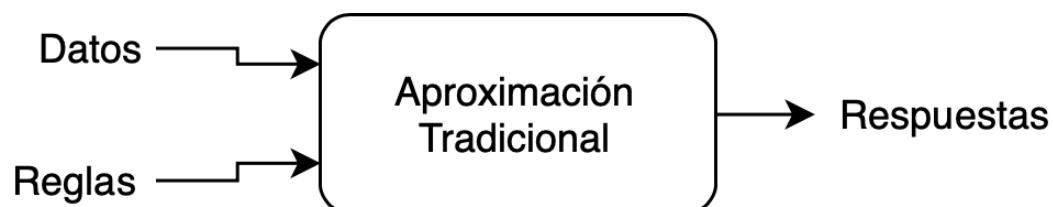
Instituto Tecnológico Metropolitano

Pedro Atencio Ortiz - 2018

---

## Una aproximación inicial al Machine Learning

La aproximación clásica al desarrollo de una solución reza lo siguiente:



Analicemos el siguiente caso para una aplicación de reconocimiento de actividad:





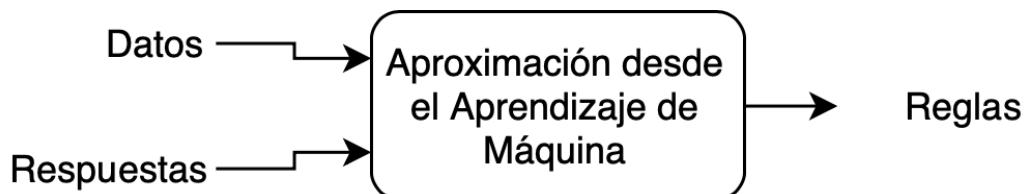
```
if (velocidad < 4 km/h):
    actividad = caminata
```

```
if (velocidad < 4 km/h):
    actividad = caminata
else:
    actividad = carrera
```

```
if (velocidad < 4 km/h):
    actividad = caminata
elif (velocidad < 12):
    actividad = carrera
else:
    actividad = ciclismo
```

???????

La aproximación del aprendizaje de maquina reza:



Analicemos el siguiente caso de reconocimiento de actividad:



```
010100100101001010
000100101110110010
001001011101010100
```

**Etiqueta:** Caminata



```
110100100101001010
000100101000110010
001001011101110101
```

**Etiqueta:** Carrera



```
110100100101001010
000100101000110010
001001011101110101
```

**Etiqueta:** Ciclismo



```
110100100101001010
000100101000110010
001001011101110101
```

**Etiqueta:** Tennis

## Ejemplo utilizando Keras

Cuál es el patrón que siguen los siguientes datos?

$$X = [-1, 0, 1, 2, 3, 4]$$

$$Y = [-3, -1, 1, 3, 5, 7]$$

Azure Notebooks  
Preview  
(/help/notebooks/)

My Projects  
(/atehortua1907/projects#)  
Help  
(https://docs.microsoft.com/en-us/learn/modules/101-what-is-machine-learning/)

Hagamos que la máquina aprenda por nosotros el patrón subyacente en los datos.

```
In [ ]: import keras
         from keras.layers import Dense

         import numpy as np
         import matplotlib.pyplot as plt
```

```
In [ ]: X = np.array([-1,0,1,2,3,4], dtype=float)
         Y = np.array([-3,-1,1,3,5,7], dtype=float)
```

```
In [ ]: print(X)
         print(Y)
```

## 1. Creamos un modelo de red neuronal simple (1 neurona)

```
In [ ]: model = keras.Sequential()
         model.add(Dense(units=1, input_shape=[1]))
```

## 2. Compilamos el modelo

Definimos el optimizador y el error a minimizar.

```
In [ ]: model.compile(optimizer='sgd', loss='mean_squared_error')
```

## 3. Entrenamiento

Hacemos que el modelo aprenda.

```
In [ ]: history = model.fit(X, Y, epochs=500, verbose=False)
```

```
In [ ]: plt.plot(history.history['loss'])
         plt.show()
```

## 4. Utilizamos el modelo entrenado

```
In [ ]: model.predict([5])
```

**Qué aprendió el modelo???**

```
In [ ]: model.get_weights()
```

# Trabajemos en clase:

## 1. Renta de apartamentos

Construyamos una red neuronal simple para el siguiente problema:

Imaginemos que el precio de alquiler de un apartamento viene dado por  $500k + 150k$  por cuarto, por ejemplo, 650 mil pesos para un apartamento con un cuarto, 800 mil pesos para dos cuartos y así sucesivamente.

Cómo podríamos crear un modelo de red neuronal simple para que aprenda estas relaciones de tal forma que el sistema pueda predecir valores por fuera de lo que aprendió?

**Pista 1:** Estos modelos de aprendizaje funcionan mejor si los datos se escalan, es decir, en términos del resultado final es lo mismo que el sistema reciba el numero 2 (dos cuartos) y entregue como resultado 8 (de 800 mil) y luego se escala a cientos de miles. Sin embargo, para la red es mas facil trabajar con números pequeños o escalados.

**Pista 2:** Si se cuentan con valores intermedios para la entrada, tales como 2.1, 1.4, 3.2, etc. el sistema aprenderá mejor.

## 2. Función $y = x^2$

Genere un conjunto de datos para la ecuación  $y = x^2$  en intente hacer que el sistema aprenda dicha relación. Puede agregar mas capas y neuronas al modelo. Qué resultado obtiene?

In [ ]: