



Inteligencia Artificial - IAI84

Instituto Tecnológico Metropolitano

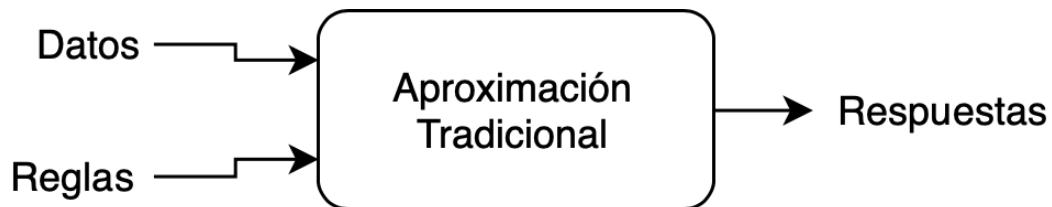
Pedro Atencio Ortiz - 2018

Agenda:

1. Introducción.
2. Tipos de problemas y datasets.

1. 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:



Azure
Notebooks
(/#)

Previous
My
Projects
(/help/recent/)



https://docs.microsoft.com/en-
us/learn/courses/10107/1/1/what-is-machine-learning/notebooks/)

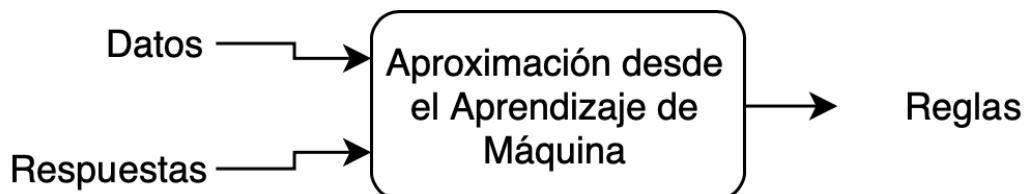
```
if (velocidad < 4 km/h):
    actividad = caminata
```

```
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

1.1. 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 \(/help/projects\)](#)[My Projects \(/atehortua1907/projects#\)](#)[Help \(/code/notebooks/\)](#)[\(https://docs.microsoft.com/en-us/\)](https://docs.microsoft.com/en-us/)

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

```
In [1]: import keras  
from keras.layers import Dense  
  
import numpy as np  
import matplotlib.pyplot as plt
```

Using TensorFlow backend.

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

```
In [3]: print(X)  
print(Y)
```

```
[-1.  0.  1.  2.  3.  4.]  
[-3. -1.  1.  3.  5.  7.]
```

1.1.1 Creamos un modelo de red neuronal simple (1 neurona)

```
In [4]: model = keras.Sequential()  
model.add(Dense(units=1, input_shape=[1]))
```

1.1.2. Compilamos el modelo

Definimos el optimizador y el error a minimizar.

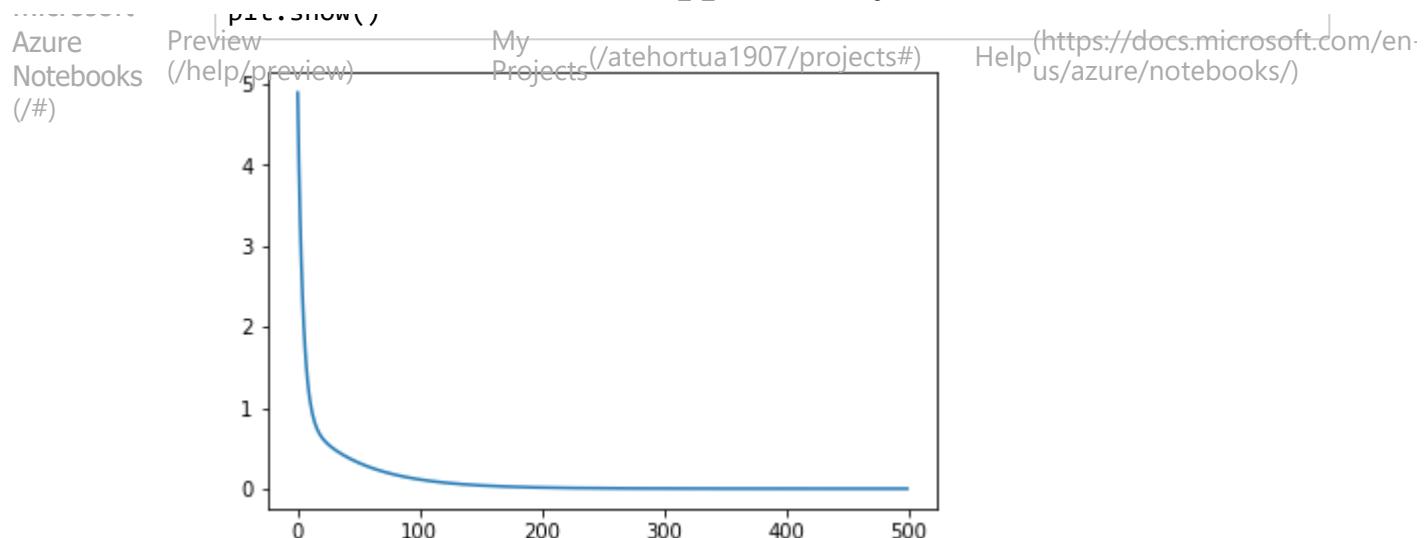
```
In [5]: model.compile(optimizer='sgd', loss='mean_squared_error')
```

1.1.3. Entrenamiento

Hacemos que el modelo aprenda.

```
In [6]: history = model.fit(X, Y, epochs=500, verbose=False)
```

```
In [7]: plt.plot(history.history['loss'])  
plt.show()
```



1.1.4. Utilizamos el modelo entrenado

```
In [8]: model.predict([5])
```

```
Out[8]: array([[8.995698]], dtype=float32)
```

1.1.4. Qué aprendió el modelo???

```
In [9]: model.get_weights()
```

```
Out[9]: [array([[1.9977359]], dtype=float32), array([-0.9929808], dtype=float32)]
```

1.2. Trabajemos en clase:

1.2.1. 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 número 2

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

1.2.2. 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?

2. Tipos de problemas y datasets

Usualmente los problemas de predicción se dividen en dos categorías:

1. **Clasificación:** En cuyo caso la salida del modelo es DISCRETA, es decir una etiqueta o categoría que se asigna al objeto en cuestión.

$$G(x) = k; k = \{label_1, label_2, label_3, \dots, label_n\}$$

2. **Regresión:** En cuyo caso la salida del modelo es CONTINUA, es decir, un valor real que estima a partir de la entrada.

$$G(x) = k; k \in R^n$$

Analicemos el siguiente dataset de la librería <a href=<https://scikit-learn.org/stable/index.html>>sklearn (<https://scikit-learn.org/stable/index.html>>sklearn):

```
In [ ]: from sklearn.datasets import load_boston
```

```
In [ ]: ??load_boston
```

```
In [ ]: from tabulate import tabulate
```

```
In [ ]: dataset = load_boston() #cargamos el dataset
```

```
In [ ]: entrada = dataset.data #aislamos Los valores de entrada del problema
salida = dataset.target #aislamos Los valores de salida del problema
```

```
In [ ]: #usualmente Los datos se cargan de esta manera
```

Azure Notebooks Previous (/help/preview) My dataset target Projects Help (<https://docs.microsoft.com/en-us/azure/notebooks/>)

```
(/#) In [ ]: #concatenemos X y Y para poder tabularlos
m, n = np.shape(X)

data = np.zeros([m, n+1])

data[:, 0:n] = X
data[:, n] = Y

In [ ]: headers = ["x"+str(i) for i in range(n)]+['y']

tabla = tabulate(data, headers=headers, tablefmt="fancy_grid", floatfmt=
print(tabla)
```

Trabajemos en clase

1. Revisar la diferencia entre [Toy Datasets](https://scikit-learn.org/stable/datasets/index.html#toy-datasets) (<https://scikit-learn.org/stable/datasets/index.html#toy-datasets>) y [Real World Datasets](https://scikit-learn.org/stable/datasets/index.html#real-world-datasets) (<https://scikit-learn.org/stable/datasets/index.html#real-world-datasets>).
2. Consultar algunos datasets de ambos casos y:
 - A. Revisar las entradas, salidas y tipo de problema en cada caso.
 - B. Tabular los datos.

()

In []: