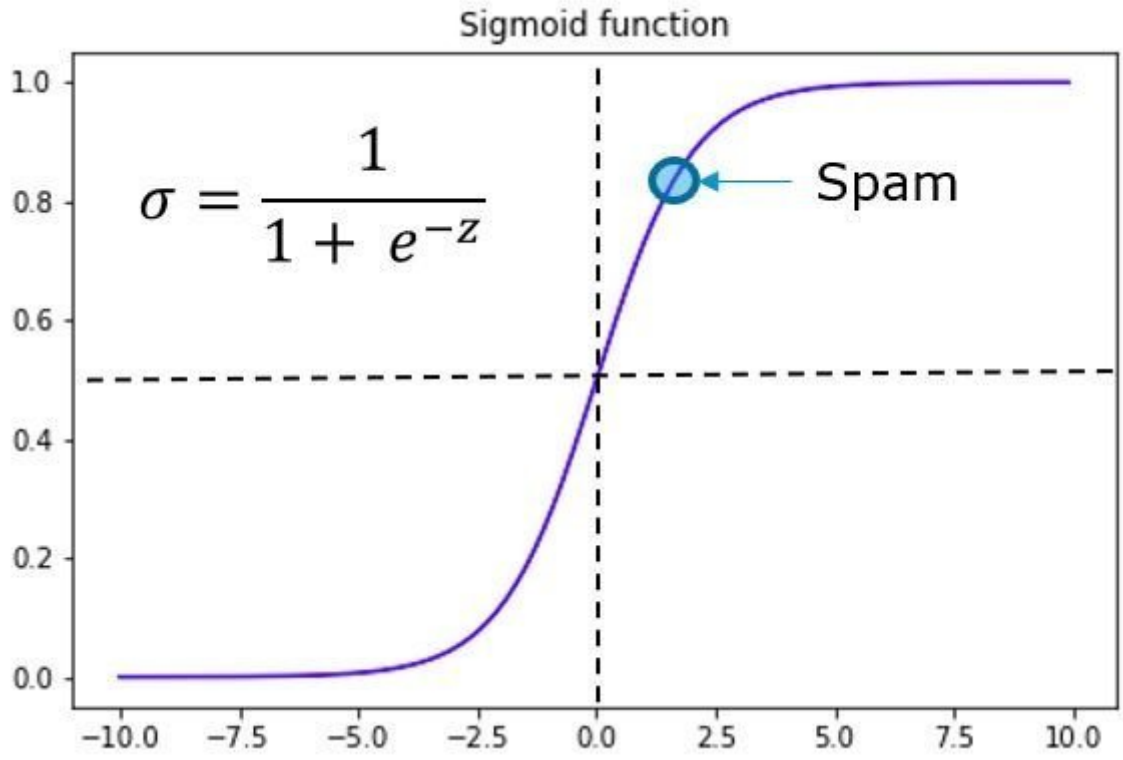
Construyendo un perceptrón sigmoide en Python

Autor: Mg Rubén Quispe ¶

```
En esta clase, nuestro objetivo es visualizar su capacitación con la ayuda de un conjunto de datos de muestra.
```

Función de activación

Primero comprendamos los conceptos básicos del modelo Sigmoid antes de construirlo. Como su nombre indica, el modelo gira en torno a la fórmula sigmoidea, que se puede representar como:



La propiedad de la curva sigmoidea (valor que varía entre 0 y 1) la hace beneficiosa para los problemas de regresión / clasificación primaria.

Función de pérdida

Como trataremos con valores reales para esta visualización, usaremos el error cuadrático medio como nuestra función de pérdida.

In [2]:

```
#Comencemos importando las bibliotecas que necesitamos
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors
import time
```

In [3]:

```
#para animacion simple
%matplotlib inline
from matplotlib import animation, rc
from IPython.display import HTML
```

Ahora enumeremos los componentes de los que formará nuestra clase SigmoidNeuron
1. función para calcular w * x + b
2. función para aplicar la función sigmoidea
3. función para predecir la salida para un marco de datos X proporcionado
4. función para devolver valores de gradiente para "w" y "b"
5. función para ajustarse al conjunto de datos proporcionado

In [4]:

```
X = np.asarray([[2.5, 2.5], [4, -1], [1, -4], [-3, 1.25], [-2, -4], [1, 5]])
Y = [1, 1, 1, 0, 0, 0]
```

In [6]:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-


class SigmoidNeuron:


  def __init__(self):
    self.w = None
    self.b = None
    self.wb=[]

  def perceptron(self, x):
    return np.dot(x, self.w.T) + self.b

  def perceptronIN(self, x,ww,bb):
    return np.dot(x, ww.T) + bb

  def sigmoid(self, x):
    return 1.0 / (1.0 + np.exp(-x))

  def predict(self, X):
    yp = []
    for x in X:
      ypt=(self.sigmoid(self.perceptron(x)))
      yp.append(ypt)
    return yp

  def grad_w(self, x, y):
    y_pred = self.sigmoid(self.perceptron(x))
    return (y_pred - y) * y_pred * (1 - y_pred) * x

  def grad_b(self, x, y):
    y_pred = self.sigmoid(self.perceptron(x))
    return (y_pred - y) * y_pred * (1 - y_pred)

  def rwb(self):
    return self.wb

  def fit(self,x,y,
        epochs=1,
        learning_rate=1,
        initialise=True,
        do_plot=False,
        ):
    if do_plot:
      loss={}


    if initialise:
      #self.w = np.random.randn(1, 2)
      self.w=np.asarray([[-1.52452385,  0.57205053]])
      self.b = np.asarray([0.24757113])
      self.wb.append([self.w,[self.b]])

    for i in range(epochs):
      dw = 0
      db = 0
      dw += self.grad_w(x, y)
```

```
      db += self.grad_b(x, y)

    self.w -= learning_rate * dw
    self.b -= learning_rate * db
    self.wb.append([self.w,self.b])
    if do_plot:
      loss[i]=mean_squared_error(self.sigmoid(self.perceptron(X)),Y)
    if do_plot:
      plt.plot(loss.values())
      plt.xlabel('Epochs')
      plt.ylabel('Error')
      plt.show()
```

In [7]:

```
my_cmap = matplotlib.colors.LinearSegmentedColormap.from_list("", ["red","yellow","green"])
```

In [9]:

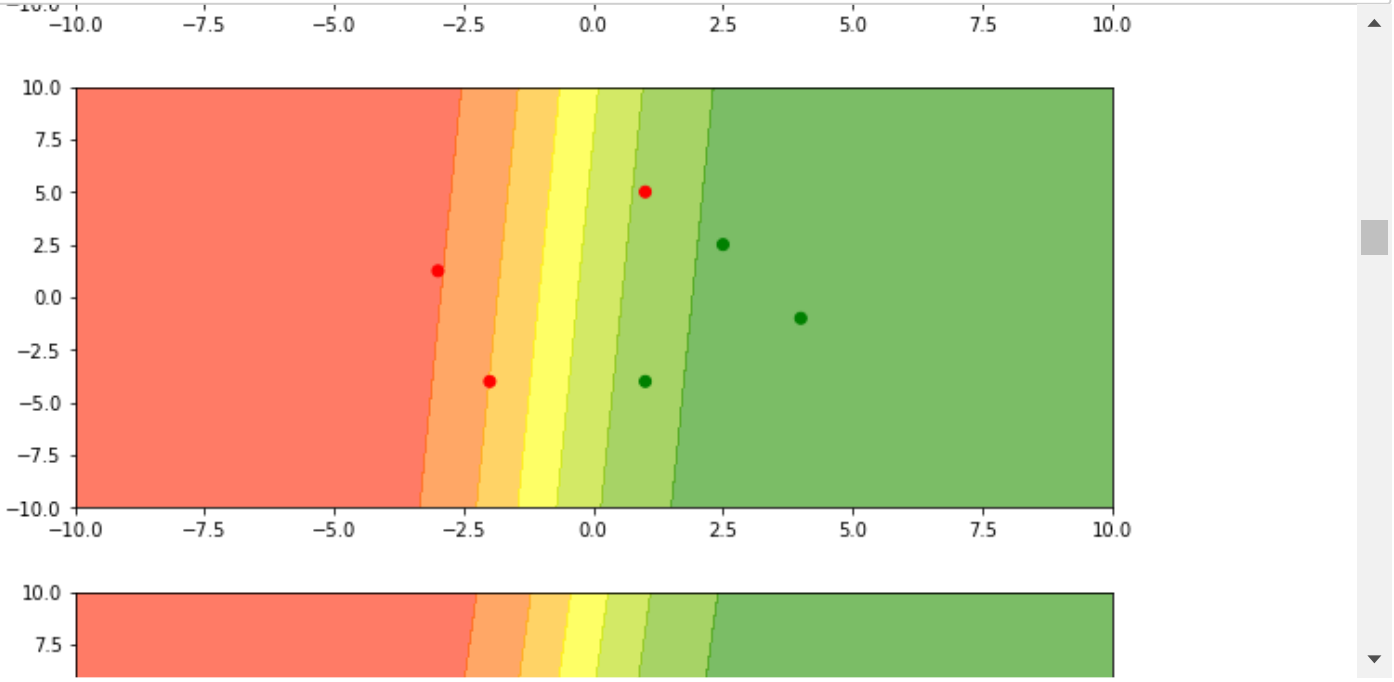
```
sn=SigmoidNeuron()
```

In [10]:

```
def plot_sn(X,Y,sn,ax,wbb): # Crea un diagrama de contorno para el valor actual de "w" y "b
  X1=np.linspace(-10,10,100)
  X2=np.linspace(-10,10,100)
  XX1,XX2=np.meshgrid(X1,X2)
  YY=np.zeros(XX1.shape)
  for i in range(X2.size):
    for j in range(X1.size):
      YY[i,j]=sn.sigmoid(sn.perceptron(np.asarray([X1[j],X2[i]])))
  ax.contourf(XX1,XX2,YY,cmap=my_cmap,alpha=0.6)
  ax.scatter(X[:,0],X[:,1],c=Y,cmap=my_cmap)
  ax.plot()
```

In [11]:

```
# inicializar el tamaño de la trama
plt.figure(figsize=(10,20*6*5))
sn.fit([2.5,2.5],1,1,1,True,False)
ci=0
wbb=[]
for i in range(20):
  #ci=0
  for (x,y) in zip(X,Y):
    ci+=1
    ax=plt.subplot(20*6,1,ci)
    sn.fit(x,y,1,0.7,False,False)
    plot_sn(X,Y,sn,ax,wbb)
```



Nuestro último paso es crear una animación para el entrenamiento.

In [12]:

```
def plot_sn2(X,Y,sn,ax,i,ww,bb):

  X1=np.linspace(-10,10,100)
  X2=np.linspace(-10,10,100)
  XX1,XX2=np.meshgrid(X1,X2)
  YY=np.zeros(XX1.shape)
  for i in range(X2.size):
    for j in range(X1.size):
      YY[i,j]=sn.sigmoid(sn.perceptronIN(np.asarray([X1[j],X2[i]]),np.asarray(ww),bb))
  ax.contourf(XX1,XX2,YY,cmap=my_cmap,alpha=0.6)
  ax.scatter(X[:,0],X[:,1],c=Y,cmap=my_cmap)
  #ax.plot()
```

In [16]:

```
def animate(i):
  #ax.clear()

  #for m in range(i+1):
  for x,y in zip(X,Y):
    sn.fit(x,y,1,0.4,False,False)
    ax.clear()
    plot_sn2(X,Y,sn,ax,i,list(sn.w),sn.b[0])
```

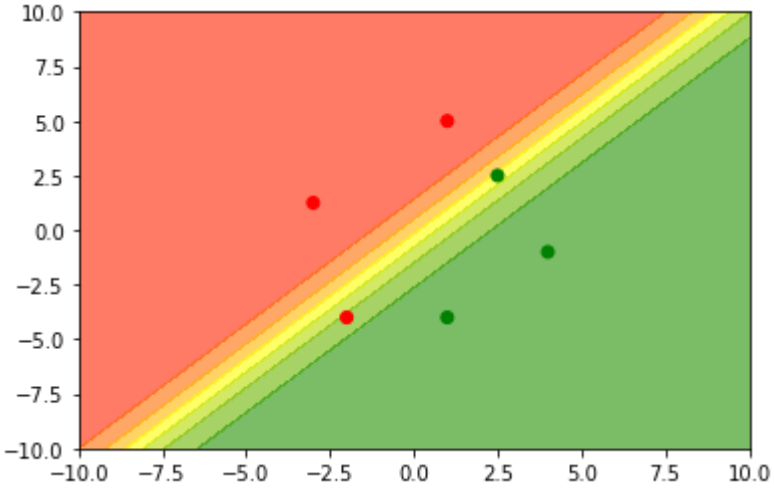
In [17]:

```
fig,ax = plt.subplots()
sn.fit([2.5,2.5],1,10,1,True,False)
rwb=sn.rwb()
interval = 1#in seconds
anim = animation.FuncAnimation(fig, animate, 30,interval=100, repeat=False,blit=False)
HTML(anim.to_html5_video())
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\animation.py in __geti
tem__(self, name)
    160         try:
--> 161             return self.avail[name]
    162         except KeyError:

KeyError: 'ffmpeg'

During handling of the above exception, another exception occurred:

RuntimeError                              Traceback (most recent call last)
<ipython-input-17-1d301789ae30> in <module>
      4 interval = 1#in seconds
      5 anim = animation.FuncAnimation(fig, animate, 30,interval=100, repeat
=False,blit=False)
----> 6 HTML(anim.to_html5_video())

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\animation.py in to_htm
l5_video(self, embed_limit)
   1337                 # We create a writer manually so that we can get the
   1338                 # appropriate size for the tag
-> 1339                 Writer = writers[rcParams['animation.writer']]
   1340                 writer = Writer(codec='h264',
   1341                                 bitrate=rcParams['animation.bitrate'
],

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\animation.py in __geti
tem__(self, name)
    162         except KeyError:
    163             raise RuntimeError(
--> 164                 'Requested MovieWriter ({}) not available'.format(na
me))
    165
    166

RuntimeError: Requested MovieWriter (ffmpeg) not available
```

In [18]:

```
ax1=plt.subplot()
plot_sn2(X,Y,sn,ax1,1,list(-sn.w[0]),-sn.b[0])
```



In [ ]:

```

```