T https://developers.google.com/machine-learning/crash-course/reducing-loss/video-lecture

Un ejemplo es una instancia en particular de los datos, x

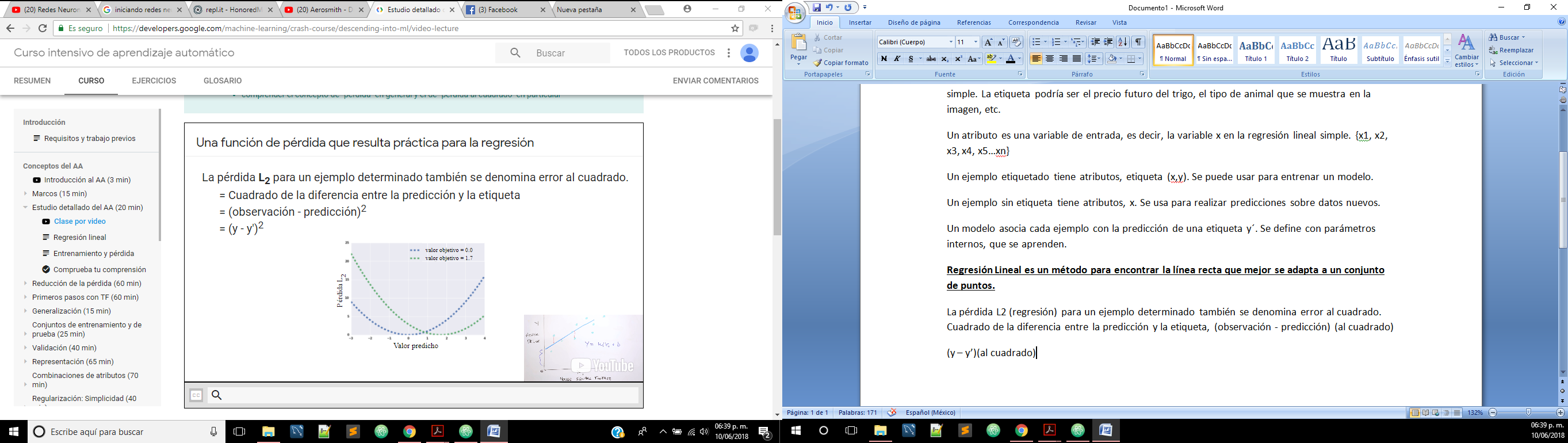
Una etiqueta es el valor que estamos prediciendo, es decir, la variable y en la regresión lineal simple. La etiqueta podría ser el precio futuro del trigo, el tipo de animal que se muestra en la imagen, etc.

Un atributo es una variable de entrada, es decir, la variable x en la regresión lineal simple. {x1, x2, x3, x4, x5…xn}

Un ejemplo etiquetado tiene atributos, etiqueta (x,y). Se puede usar para entrenar un modelo.

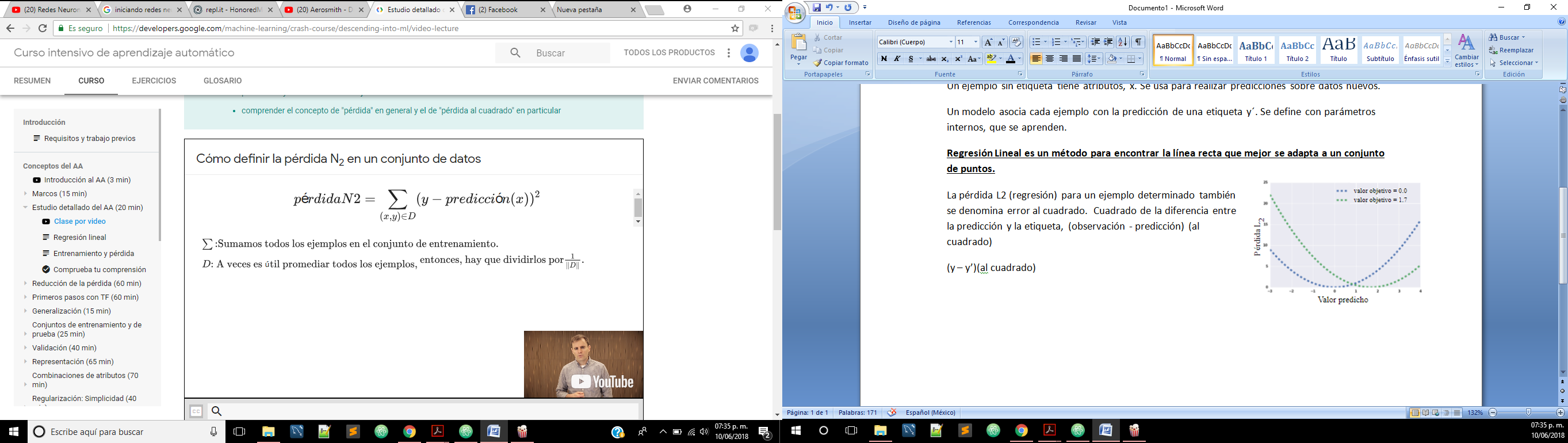
Un ejemplo sin etiqueta tiene atributos, x. Se usa para realizar predicciones sobre datos nuevos.

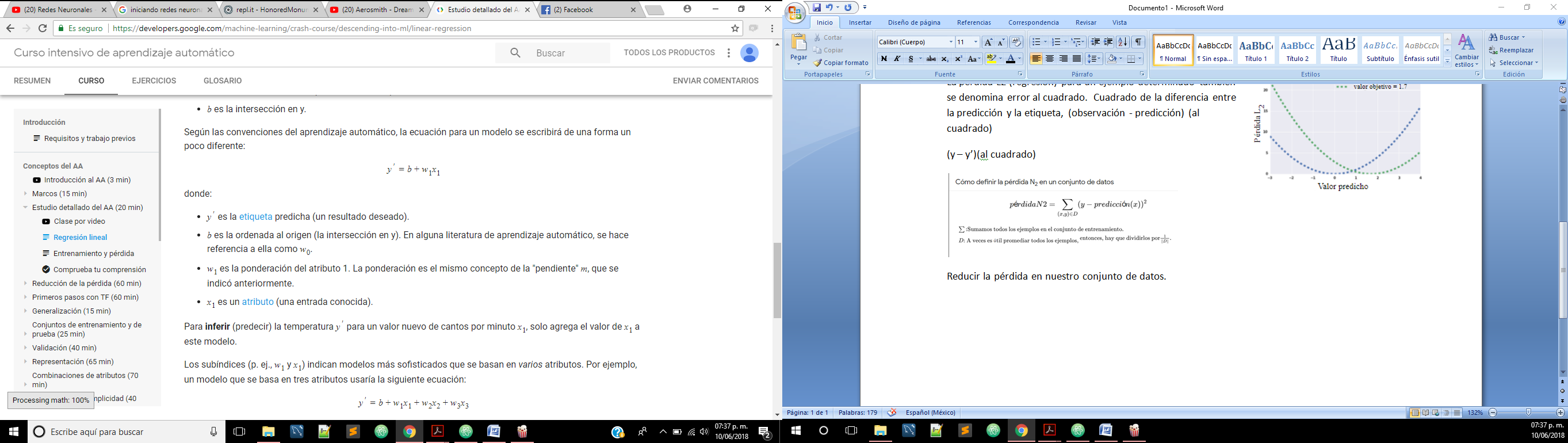
Un modelo asocia cada ejemplo con la predicción de una etiqueta y´. Se define con parámetros internos, que se aprenden.

**Regresión Lineal es un método para encontrar la línea recta que mejor se adapta a un conjunto de puntos.**

La pérdida L2 (regresión) para un ejemplo determinado también se denomina error al cuadrado. Cuadrado de la diferencia entre la predicción y la etiqueta, (observación - predicción) (al cuadrado)

(y – y’)(al cuadrado)



Reducir la pérdida en nuestro conjunto de datos.

Entrenar un modelo significa aprender valores correctos para todas las ponderaciones y las ordenadas al origen de los ejemplos etiquetados. En el aprendizaje supervisado, el algoritmo de un aprendizaje automático construye un modelo al examinar varios ejemplos e intentar encontrar un modelo que minimice la pérdida. Este proceso de le denomina minimización del riesgo empírico.

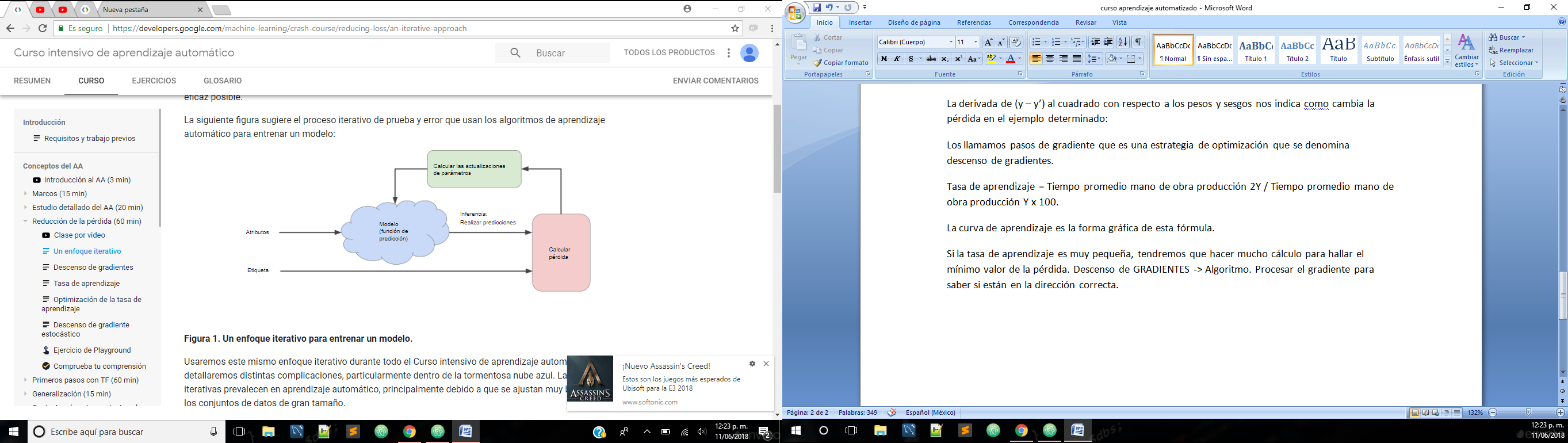
***Como reducir la pérdida?***

La derivada de (y – y’) al cuadrado con respecto a los pesos y sesgos nos indica como cambia la pérdida en el ejemplo determinado:

Los llamamos pasos de gradiente que es una estrategia de optimización que se denomina descenso de gradientes.

Tasa de aprendizaje = Tiempo promedio mano de obra producción 2Y / Tiempo promedio mano de obra producción Y x 100.

La curva de aprendizaje es la forma gráfica de esta fórmula.

Si la tasa de aprendizaje es muy pequeña, tendremos que hacer mucho cálculo para hallar el mínimo valor de la pérdida. Descenso de GRADIENTES -> Algoritmo. Procesar el gradiente para saber si están en la dirección correcta.

Este es un proceso iterativo de prueba y error que usan los algoritmos de aprendizaje automático para entrenar a un modelo. En la parte de “Actualizar parámetros” del diagrama, el sistema de aprendizaje automático examina el valor de la función de pérdida y genera valores nuevos para b y w1.

Reducción de la pérdida: Descenso de gradientes:

https://www.youtube.com/watch?v=-\_A\_AAxqzCg

<https://www.youtube.com/channel/UCy5znSnfMsDwaLlROnZ7Qbg> este es el canal de IA

import numpy as np

import scipy as sc #funciones cientificas

import matplotlib.pyplot as plt #visualización de datos

#optimización de funciones

func = lambda th: np.sin(1 / 2 \* th[0] \*\* 2 - 1/4\*th[1]\*\*2 +3) \* np.cos(2 \* th[0] +1 - np.e \*\* th[1])#función anónima, th es un parámetro

#generar un vector con una secuencua de x, y. Y los vamos a evaluar con la función para despu´pes representar en gráfia.

resolucion = 100

\_x = np.linspace(-2, 2, resolucion) #Genera 100 valores aleatorios entre el -2 y 2

\_y = np.linspace(-2, 2, resolucion)

\_Z = np.zeros((resolucion, resolucion)) #crear la matriz con ceros primero

for ix, x in enumerate(\_x):

for iy, y in enumerate(\_y):

\_Z[iy, ix] = func([x, y])

plt.contourf(\_x, \_y, \_Z, 100) #éste método pasamos las variables x, y, z. El número cien nos sirve para aumentar la resolución de la gráfica.

#contourf nos da la parte sólida mientras que contour nos dá la superficie en mallado.

plt.colorbar() #añade un indicador de valores con sus respectivos colores

#plt.show()

#print func([5,3]) #mandándola a llamar

#Ahora vamos a generar un punto aleatorio sobre la supeficie

theta = np.random.rand(2) \* 4 -2 #números del rango -2 a 2

\_T = np.copy(theta)

#ahora vamos con las derivadas parciales, que es donde calculamos la pendiente en dicho punto para poder ir descendiendo.

h = 0.001

ratioAprendizaje = 0.001

#generando el punto aleatorio

plt.plot(theta[0], theta[1], "o", c = "white")

gradiente = np.zeros(2)

for \_ in range(10000):

for it, th in enumerate(theta):

\_T = np.copy(theta)

\_T[it] = \_T[it] + h

deriv = (func(\_T) - func(theta)) / h #derivada parcial del primer vector, o sea la pendiente de movernos en el eje Z[i]

gradiente[it] = deriv

#ahora actualizamos parámetros

theta = theta - ratioAprendizaje \* gradiente

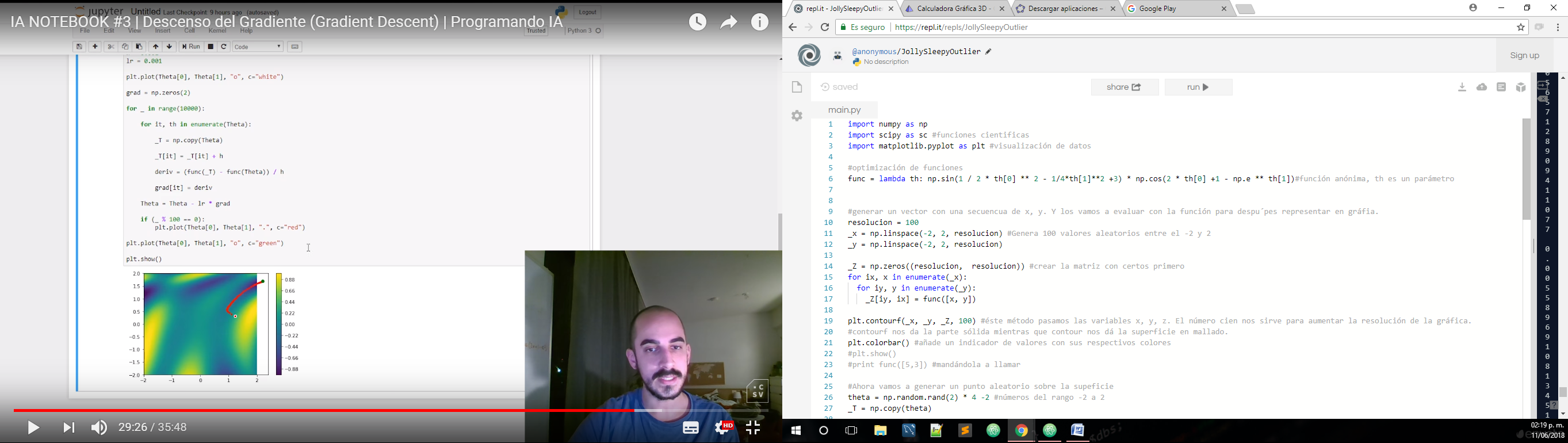
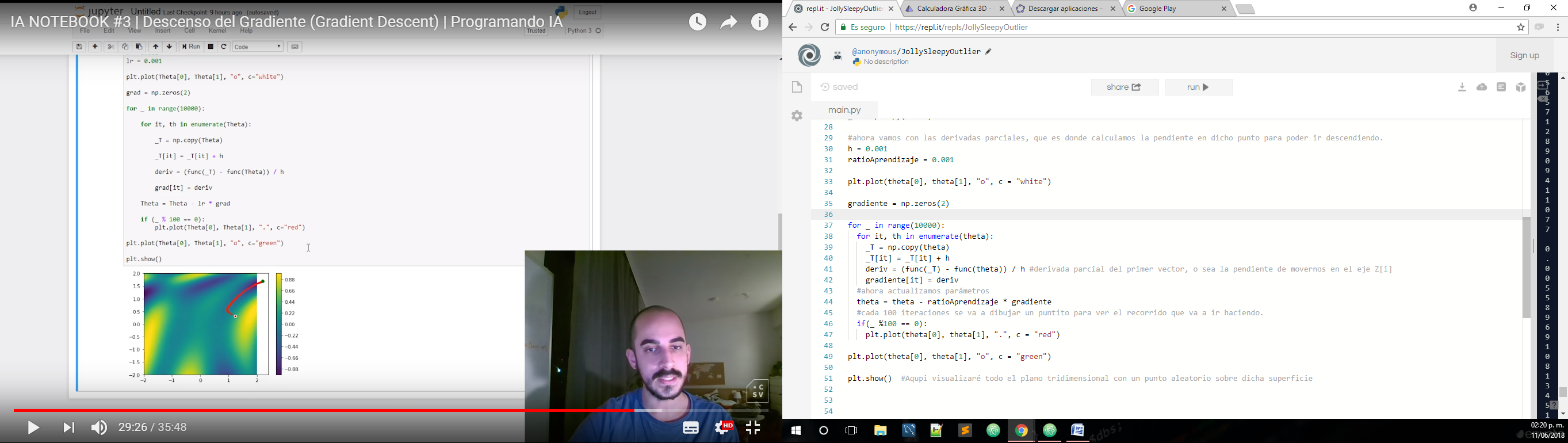
#cada 100 iteraciones se va a dibujar un puntito para ver el recorrido que va a ir haciendo.

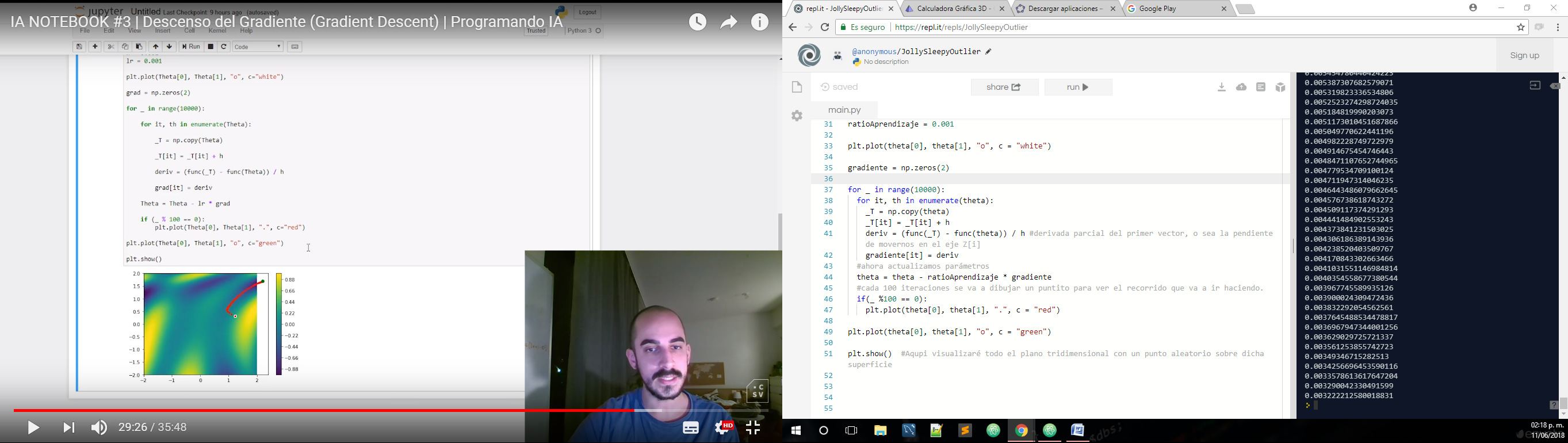
if(\_ %100 == 0):

plt.plot(theta[0], theta[1], ".", c = "red")

plt.plot(theta[0], theta[1], "o", c = "green")

plt.show() #Aqupi visualizaré todo el plano tridimensional con un punto aleatorio sobre dicha superficie





clc, clear, close all

p = [1 1 1 1 1 1 0; %0

0 1 1 0 0 0 0; %1

1 1 0 1 1 0 1; %2

1 1 1 1 0 0 1; %3

0 1 1 0 0 1 1; %4

1 0 1 1 0 1 1; %5

1 0 1 1 1 1 1; %6

1 1 1 0 0 0 0; %7

1 1 1 1 1 1 1; %8

1 1 1 1 0 1 1]; %9

p = p';

q = size(p, 2);

z = [p; ones(1, q)];

tpar = [ 1 -1 1 -1 1 -1 1 -1 1 -1];

tmay5 = [-1 -1 -1 -1 -1 -1 1 1 1 1 ];

tprim = [-1 -1 1 1 -1 1 -1 1 -1 -1];

T = [tpar; tmay5; tprim];

%Aprendizaje -WidrowHoff

R = 1 / q\*z\*z';

H = 1 / q\* z\*T';

Xm = inv(R) \* H;

w = Xm(1:7, :);

b = Xm(8, :);

w1 = Xm(1:7, 1);

w2 = Xm(1:7, 2);

w3 = Xm(1:7, 3);

b1 = Xm(8, 1);

b2 = Xm(8, 2);

b3 = Xm(8, 3);

entrada = [0 1 1 0 0 0 0]; %ingresando el número 1

a1 = purelin(w1 \* entrada + b1); %par

a2 = purelin(w2 \* entrada + b2); %tmay5

a3 = purelin(w3 \* entrada + b3); %tprim

a1

a2

a3

%http://www.compileonline.com/execute\_matlab\_online.php

Codigo matlab para ADALINE WIDROWHOFF 3 NEURONAS

clc, clear, close all

p = [0.7 1.5 2.0 0.9 4.2 2.2 3.6 4.5;

3 5 9 11 0 1 7 5];

t = [ 0 0 0 0 1 1 1 1;

0 0 1 1 0 0 1 1];

%Algoritmo de aprendizaje perceptrón

w = 2 \* rand(2) -1;

b = 2 \* rand(2, 1) -1;

for Epocas = 1: 30

for q = 1: 8

e(: , q) = t(: , q) - hardlim(w \* p(: , q) + b);

w = w + e(: , q) \* p(: , q)';

b = b + e(: , q);

end

end

CÓDIGO PERCEPTRÓN PARA EL EJERCICIO DEL BIBKIOTECARIO

RED NEURONAL CON WIDROW HOFF PYTHON CLASIFICANDO NUMEROS:

import numpy as np

from scipy import linalg

p = np.array([[1, 1, 1, 1, 1, 1, 0], #0

[0, 1, 1, 0, 0, 0, 0], #1

[1, 1, 0, 1, 1, 0, 1], #2

[1, 1, 1, 1, 0, 0, 1], #3

[0, 1, 1, 0, 0, 1, 1], #4

[1, 0, 1, 1, 0, 1, 1], #5

[1, 0, 1, 1, 1, 1, 1], #6

[1, 1, 1, 0, 0, 0, 0], #7

[1, 1, 1, 1, 1, 1, 1], #8

[1, 1, 1, 1, 0, 1, 1]])#9

cero = p[0]

uno = p[1]

dos = p[2]

tres = p[3]

cuatro = p[4]

cinco = p[5]

seis = p[6]

siete = p[7]

ocho = p[8]

nueve = p[9]

unos = np.array([1,1,1,1,1,1,1,1,1,1])

p = p.transpose()

#La matriz z surge de hacer transpuesta la matriz P y añadirle una matriz al final

#de unos

z =np.array( [ [1, 0, 1, 1, 0, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 0, 0, 1, 1, 1],

[1, 1, 0, 1, 1, 1, 1, 1, 1, 1],

[1, 0, 1, 1, 0, 1, 1, 0, 1, 1],

[1, 0, 1, 0, 0, 0, 1, 0, 1, 0],

[1, 0, 0, 0, 1, 1, 1, 0, 1, 1],

[0, 0, 1, 1, 1, 1, 1, 0, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])

q = 10 #Tiene que regresarme el valor de 10

tpar = np.array([ 1, -1, 1, -1, 1, -1, 1, -1, 1, -1])

tmay5 = np.array([-1, -1, -1, -1, -1, -1, 1, 1, 1, 1 ])

tprim = np.array([-1, -1, 1, 1, -1, 1, -1, 1, -1, -1])

T = np.array([tpar, tmay5, tprim])

#Aprendizaje -WidrowHoff

R = float(1)/float(q) \* np.dot(z, z.transpose())

H = float(1)/float(q) \* np.dot(z, T.transpose())

Xm = np.dot(linalg.inv(R), H)

#Obteniendo los pesos sinápticos

w1 = []

w2 = []

w3 = []

for i in range(len(Xm) - 1):

w1.append(Xm[i][0])

w2.append(Xm[i][1])

w3.append(Xm[i][2])

w1 = np.array(w1)

w2 = np.array(w2)

w3 = np.array(w3)

#Obteniendo las polaridades

b1 = Xm[7][0]

b2 = Xm[7][1]

b3 = Xm[7][2]

def net\_input(entrada, W, B):

   return np.dot(entrada, W) + B

def definir(expresion):

if expresion < 0:

print "desactivada\n"

else:

print "activada\n"

entrada = nueve

print "Números pares:"

definir(net\_input(entrada, w1, b1))

print "Numeros mayores a 5:"

definir(net\_input(entrada, w2, b2))

print "Numeros primos:"

definir(net\_input(entrada, w3, b3))