**Trabajo Práctico 6**

**Entrenamiento de la red neuronal para reconocimiento de imágenes**

Valentin Faraz

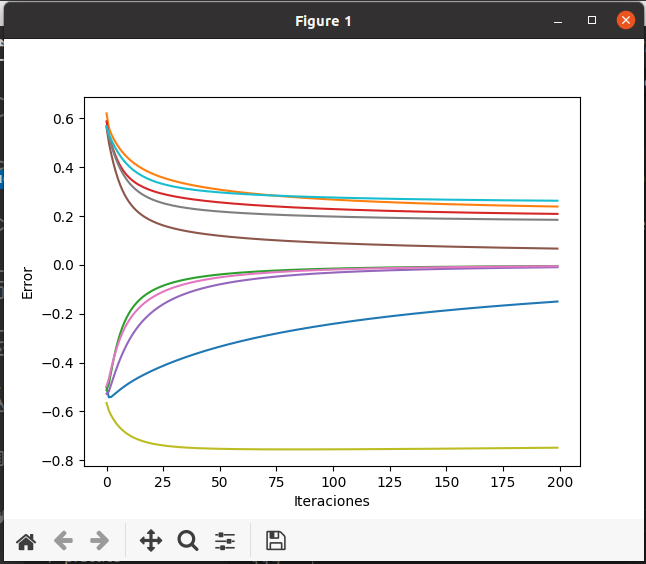
58018

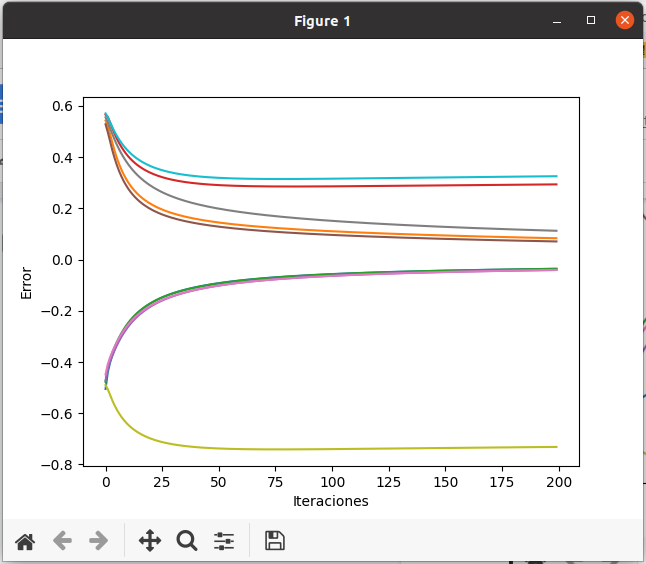
Ing.Computación

**Gráfico de errores para cada fila de datos**

10 neuronas 200 iteraciones

Tiempo de cómputo: 1 min

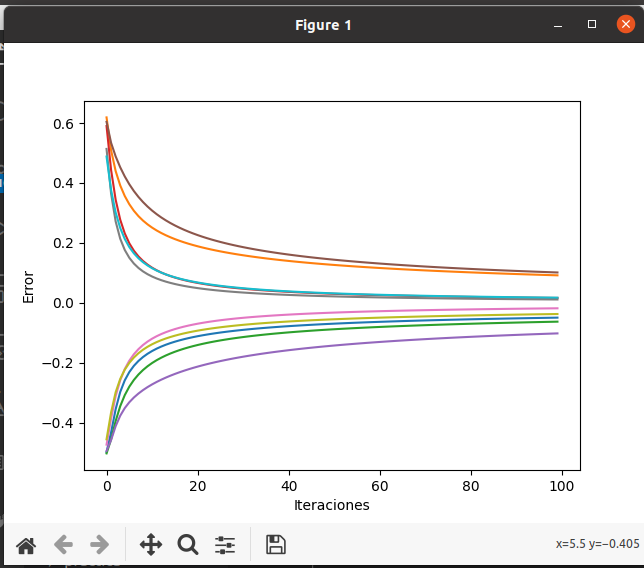


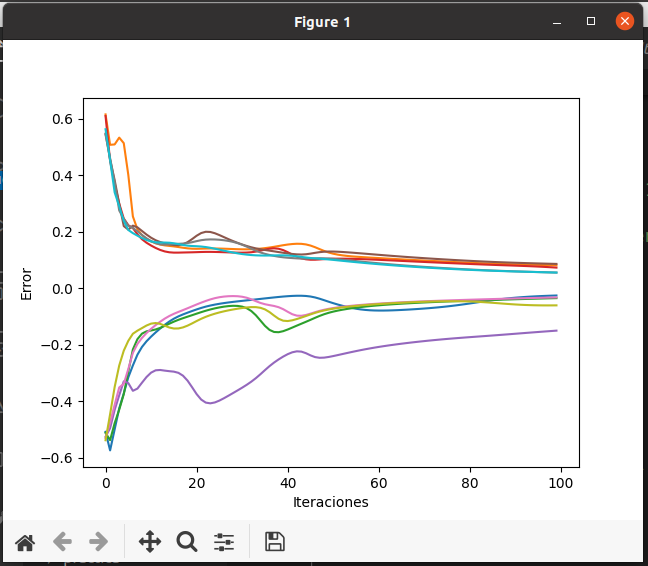


Se puede observar que para una de las imágenes la red no aprende, esto se soluciona agregando más neuronas como se observa a continuación.

20 neuronas 100 iteraciones

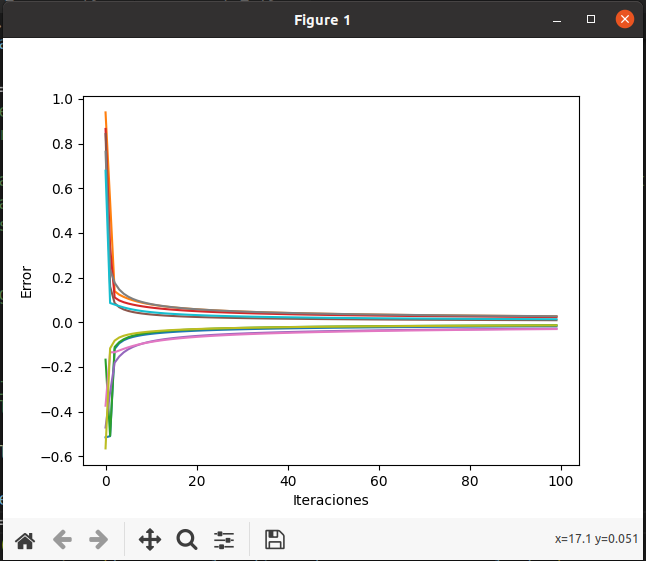
Tiempo de cómputo: 1,3 min





100 neuronas 100 iteraciones

Tiempo de cómputo: 5,21 min



from math import e

from random import random, uniform

import matplotlib.pyplot as plt

from PIL import Image

import numpy as np

def entrenar\_perceptron(imagenes, lr, iteraciones, n,solucion):

errores= [[],[],[],[],[],[],[],[],[],[]]

# Pesos de la Capa oculta

w\_oculta = []

# Pesos ultima Neurona

w\_salida = []

for i in range(n):

w\_oculta.append([])

for \_ in range(7681):

w\_oculta[i].append(round(uniform(-0.01, 0.01), 2))

# Pesos de la Ultima Neurona

for i in range(n+1):

w\_salida.append(round(uniform(-0.01, 0.01), 2))

#print("Pesos capa oculta:", w\_oculta)

#print("Pesos ultima neurona:", w\_salida)

for iteraciones in range(iteraciones):

print("Iteracion: ", iteraciones+1)

for pix in range(len(imagenes)):

x = 0

# Salida de las neuronas de la capa oculta

salida = [1]

# Delta de los pesos de la neurona de salida

dw\_n\_salida = []

# Delta de las neuronas capa oculta

delta\_co = []

# Delta de los pesos de las neuronas capa oculta

dw\_co = []

solucion\_deseada = int(solucion[pix])

#print(solucion\_deseada)

x\_neu\_salida = 0

#print(imagenes[pix][0])

#print(len(imagenes[pix]))

for i in range(n):

x = 0

for j in range(len(imagenes[pix])):

# x es sumatoria de cada peso por su entrada

x = x + imagenes[pix][j]\*w\_oculta[i][j]

# salida de las neuronas capa oculta

salida.append(1/(1+e\*\*(-x)))

#print(salida)

for i in range(n+1):

# salida de la ultima neurona

x\_neu\_salida += salida[i]\*w\_salida[i]

salida\_real = 1/(1+e\*\*(-x\_neu\_salida))

error = solucion\_deseada-salida\_real

deltaf = salida\_real\*(1-salida\_real)\*error

# Delta de los pesos de la neurona de salida

for i in range(n+1):

dw\_n\_salida.append(lr\*salida[i]\*deltaf)

# Nuevos pesos ultima Neurona

w\_salida[i] = w\_salida[i]+dw\_n\_salida[i]

salida.pop(0)

for i in range(n):

# Deltas de las neuronas capa oculta

delta\_co.append(salida[i]\*(1-salida[i])\*deltaf)

for en in imagenes[pix]:

# Delta de los pesos de la capa oculta

dw\_co.append(lr\*en\*delta\_co[i])

dw\_co.append(lr\*en\*delta\_co[i])

dw\_co.append(lr\*en\*delta\_co[i])

count = 0

for i in range(n):

for j in range(3):

# Nuevos pesos de la capa oculta

w\_oculta[i][j] = w\_oculta[i][j]+dw\_co[count]

count += 1

#print("Deltas de los pesos ultima Neurona:",dw\_n\_salida)

#print("Pesos ultima neurona:",w\_salida)

#print("Nuevos pesos:",w\_oculta)

#print("Salida real:", salida\_real)

#print("Salida deseada:", solucion\_deseada)

#print("error:", error)

#print("-------------------------")

errores[pix].append(error)

plt.plot(errores[0])

plt.plot(errores[1])

plt.plot(errores[2])

plt.plot(errores[3])

plt.plot(errores[4])

plt.plot(errores[5])

plt.plot(errores[6])

plt.plot(errores[7])

plt.plot(errores[8])

plt.plot(errores[9])

plt.ylabel('Error')

plt.xlabel('Iteraciones')

plt.show()

def leer\_imagenes():

ruta='/home/valentin/Escritorio/5to/InteligenciaArtificial/tp6/fotos/'

lista=[]

imagenes=[]

for gesto in range(1,6):

for persona in 'AB':

nombre=str(gesto)+persona+'58018.jpg'

print(nombre)

img = Image.open(ruta+nombre)

w,h=img.size

#print("entradas:",w\*h)

#print(h)

pixel=img.load()

for i in range(w):

for j in range(h):

lista.append(pixel[i,j][0])

lista.insert(0,1)

imagenes.append(lista)

lista=[]

#print(len(imagenes))

return(imagenes)

if \_\_name\_\_ == "\_\_main\_\_":

imagenes=leer\_imagenes()

lr = 0.5

iteraciones = 100

neuronas = 100

solucion=(0,1,0,1,0,1,0,1,0,1)

entrenar\_perceptron(imagenes, lr, iteraciones, neuronas,solucion)