

Optativa Profesionalizante II

Machine Learning y Deep Learning

▼ Primer Parcial

▼ 23-01-23

Docente: Francisco Javier Luna Rosas

Materia: Optativa Profesionalizante II / Machine Learning y Deep Learning

Introducción a la materia

▼ 24-01-23

Examen diagnóstico

▼ 25-01-23

No hubo clase

▼ 26-01-23

Bases de las redes neuronales

▼ 27-01-23

No fui a clase

▼ 30-01-23

Teoría de Machine Learning

▼ 31-01-23

Teoría y ejercicio de la neurona de McCulloch - Pitts

▼ 01-02-23

Teoría del perceptrón

▼ 02-02-23

Implementación del perceptrón en Python

▼ 03-02-23

No fui a clase :(

▼ 07-02-23

Repaso del viernes donde se hicieron ejercicios del perceptrón multicapa para clasificar problemas linealmente no separables

Buscar el bias y los pesos para poder clasificar la compuerta XOR:

0 0 0
0 1 1
1 0 1
1 1 0

Solución:

La idea es preparar cada neurona para cada uno

De este modo, θ_1, w_{11}, w_{12} deben entrenarse para conseguir el primer cero:

$$\begin{aligned}w_{11} \cdot 0 + w_{12} \cdot 0 &= 0 \\w_{11} \cdot 0 + w_{12} \cdot 1 &= 1 \\w_{11} \cdot 1 + w_{12} \cdot 0 &= -1 \\w_{11} \cdot 1 + w_{12} \cdot 1 &= 0\end{aligned}$$

Con $w_{11} = -1, w_{12} = 1, \theta_1 = 0.5$

De este modo, θ_2, w_{21}, w_{22} deben entrenarse para conseguir el primer cero:

$$\begin{aligned}w_{21} \cdot 0 + w_{22} \cdot 0 &= 0 \\w_{21} \cdot 0 + w_{22} \cdot 1 &= -1 \\w_{21} \cdot 1 + w_{22} \cdot 0 &= 1 \\w_{21} \cdot 1 + w_{22} \cdot 1 &= 0\end{aligned}$$

Con $w_{21} = 1, w_{22} = -1, \theta_2 = 0.5$

Las salidas anteriores producen una tabla tipo:

0 0 0
1 0 1
0 1 1
0 0 0

De este modo, θ_3, w_{31}, w_{32} deben entrenarse para conseguir el primer cero:

$$\begin{aligned}w_{31} \cdot 0 + w_{32} \cdot 0 &= 0 \\w_{31} \cdot 1 + w_{32} \cdot 0 &= 1 \\w_{31} \cdot 0 + w_{32} \cdot 1 &= 1 \\w_{31} \cdot 0 + w_{32} \cdot 0 &= 0\end{aligned}$$

Con $w_{31} = 1, w_{32} = 1, \theta_3 = 0.5$ se tiene el resultado deseado

▼ 08-02-23

Output: A trained neural network.

Method:

- (1) Initialize all weights and biases in *network*;
- (2) while terminating condition is not satisfied {
- (3) for each training tuple X in D {
- (4) // Propagate the inputs forward:
- (5) for each input layer unit j {
- (6) $O_j = I_j$; // output of an input unit is its actual input value
- (7) for each hidden or output layer unit j {
- (8) $I_j = \sum_i w_{ij} O_i + \theta_j$; // compute the net input of unit j with respect to the previous layer, i
- (9) $O_j = \frac{1}{1 + e^{-I_j}}$; } // compute the output of each unit j
- (10) // Backpropagate the errors:
- (11) for each unit j in the output layer
- (12) $Err_j = O_j(1 - O_j)(T_j - O_j)$; // compute the error
- (13) for each unit j in the hidden layers, from the last to the first hidden layer
- (14) $Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$; // compute the error with respect to the next higher layer, k

Example 6.9 Sample calculations for learning by the backpropagation algorithm. Figure 6.18 shows a multilayer feed-forward neural network. Let the learning rate be 0.9. The initial weight and bias values of the network are given in Table 6.3, along with the first training tuple, $X = (1, 0, 1)$, whose class label is 1.

Table 6.3 Initial input, weight, and bias values.

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

Tenemos:

$$x_1 \cdot w_{14} + x_2 \cdot w_{24} + x_3 \cdot w_{34} = -0.3$$

$$-0.3 > -0.4$$

Por lo tanto sí pasa

Tenemos:

$$x_1 \cdot w_{15} + x_2 \cdot w_{25} + x_3 \cdot w_{35} = -0.8$$

$$-0.8 < 0.2$$

Por lo tanto no pasa

$$x_4 \cdot w_{46} + x_5 \cdot w_{56} = -0.3$$

$$-0.3 < 0.1$$

Por lo tanto no pasa

▼ **09-02-23**

Ejercicio de Backpropagation

▼ **10-02-23**

Repaso, perceptrón, backpropagation y algoritmos

▼ **13-02-23**

Avances en la tarea de la red neuronal

▼ **14-02-23**

No entré a clase

▼ **15-02-23**

Avances en la tarea

▼ **16-02-23**

Resolución de dudas de Machine Learning

▼ **17-02-23**

Resolución de dudas de Machine Learning

▼ **20-02-23**

Dudas y avances en el examen

▼ **21-02-23**

Dudas y avances en el examen

▼ **22-02-23**

Dudas y avances en el examen

▼ **23-02-23**

Dudas y avances en el examen

▼ **24-02-23**

Dudas y avances en el examen

▼ **27-02-23**

Dudas y avances en el examen

▼ **28-02-23**

Dudas y avances en el examen

▼ **01-03-23**

Dudas y avances en el examen

▼ 02-03-23

Revisión del examen

▼ 03-03-23

Revisión del examen

▼ Segundo Parcial

▼ 06-03-23

Revisión del examen

▼ 07-03-23

Teoría del descenso del gradiente

▼ 08-03-23

Tarea del gradiente en 2D

▼ 09-03-23

Teoría del gradiente en 3D

▼ 10-03-23

Tarea del gradiente en 3D

▼ 13-03-23

Deep Learning

▼ 14-03-23

Teoría de las redes convolucionales

▼ 15-03-23

Teoría de Redes Neuronales Convolucionales

▼ 16-03-23

Teoría de Redes Neuronales Convolucionales

▼ 17-03-23

Teoría de Redes Neuronales Convolucionales

▼ 21-03-23

Teoría de Redes Neuronales Convolucionales

▼ 22-03-23

Teoría de Redes Neuronales Convolucionales

▼ 23-03-23

Teoría de Redes Neuronales Convolucionales

▼ **24-03-23**

Teoría de Redes Neuronales Convolucionales

▼ **27-03-23**

Fundamentos de las convoluciones

▼ **28-03-23**

Código de las convoluciones

▼ **29-03-23**

Tarea de convoluciones

▼ **30-03-23**

La red neuronal convolucional

▼ **31-03-23**

Código de red neuronal convolucional

▼ **Tercer Parcial**

▼ **03-04-23**

Tarea de la red neuronal convolucional

▼ **04-04-23**

Ejemplos de CNN

▼ **05-04-23**

Ejemplos de CNN

El lunes hablaremos del examen

▼ **10-04-23**

Código de CNN

Examen

Será desarrollar una red neuronal convolucional 1D

Clase importante para el examen

▼ **11-04-23**

Procesamiento del lenguaje natural y enlace con el examen

También clase importante para el proyecto final de análisis de sentimientos con imágenes

▼ **12-04-23**

Continuación de las explicaciones del examen y el proyecto

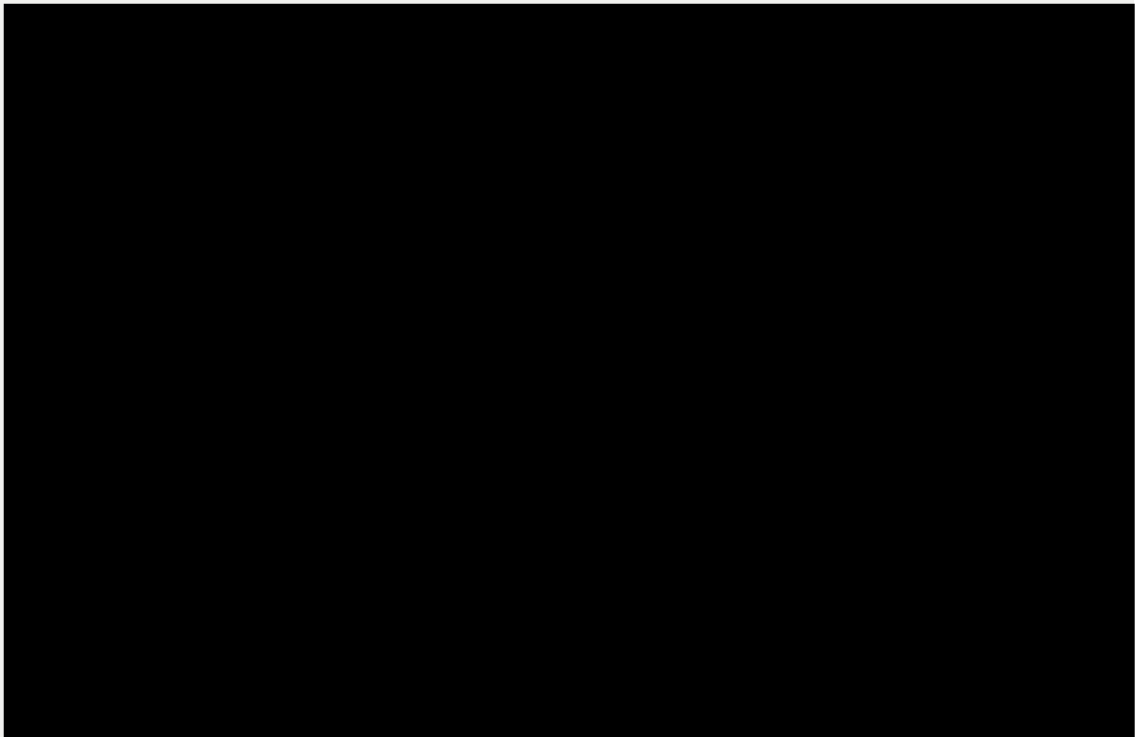
▼ **13-04-23**

Continuación de las explicaciones del examen y el proyecto

▼ 14-04-23

Continuación de las explicaciones del examen y el proyecto

▼ Mi investigación para examen y proyecto



Proyecto

En el min 22 de la clase del proyecto empieza

En 23:32 de ese mismo video abre un archivo que es análisis de sentimientos por face detection con el siguiente código:

```
In [5]: from keras.models import load_model
        from time import sleep
        from keras.preprocessing.image import img_to_array
        from keras.preprocessing import image
        import cv2
        import numpy as np

In [6]: #C:\Users\Lizzr\anaconda3\Lib\site-packages\opencv_contrib_python-4.4.0.42.dist-info
        face_classifier = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml') #Carga Haarcascades
        classifier = load_model('C:/LUNA/PYTHON_2022/BASURA/DATASET/Modelo_3.h5') #Carga Modelo
        #class_labels = ['Enojado', 'Feliz', 'Neutral', 'Sorprendido', 'Triste']
        class_labels = ['Feliz', 'Enojado', 'Sorpres']

        #PARA CARGAR LAS IMAGENES
        '''
        def face_detector(img):
            # Preprocesa imagen
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = face_classifier.detectMultiScale(gray, 1.3, 5)
            if faces is ():
                return (0, 0, 0, 0), np.zeros((48, 48), np.uint8), img
```

```

for (x,y,w,h) in faces: #Enmarca rostro
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h,x:x+w]
    try:
        roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)
    except:
        return (x,w,y,h),np.zeros((48,48),np.uint8),img
    ...
    return (x,w,y,h),roi_gray,img

#DETECCION EN TIPO REAL CON WEBCAM
cap = cv2.VideoCapture(0) #Webcam

while True:
    ret, frame = cap.read() #Lee webcam
    #Preprocesa la imagen
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY) #escala de grises
    faces = face_classifier.detectMultiScale(gray,1.3,5) #Deteccion de caras (ubicacion de la cara)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2) #obtiene solo el area de la cara (recorta el la cara)
        roi_gray = gray[y:y+h,x:x+w] #vuelve a generar la escala de grises
        roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA) #Interpolacion -> Para minimizar imagen (hace homoge

        #Si detecta rostro genera un recuadro
        if np.sum([roi_gray])!=0:
            roi = roi_gray.astype('float')/255.0 #escala y convierte a grises
            roi = img_to_array(roi) #convierte a numeros
            roi = np.expand_dims(roi,axis=0) #

            preds = classifier.predict(roi)[0] #prediccion (el cero implica el canal que hace referencia a la web cam)
            label=class_labels[preds.argmax()]
            label_position = (x,y) #posicion x,y para escribir la etique (feliz o enojado) ensima del recuadro
            cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3) #escribe la estique (feliz o enojado)
        else:
            cv2.putText(frame,'No Face Found',(20,60),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3) # No se abre la camara hasta que se
            # el rostro

```

```

cv2.imshow('Detector de emociones',frame)

if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

```

Abrió otro archivo llamado Captura Imagenes Dataset Webcam:

```

import cv2
import os

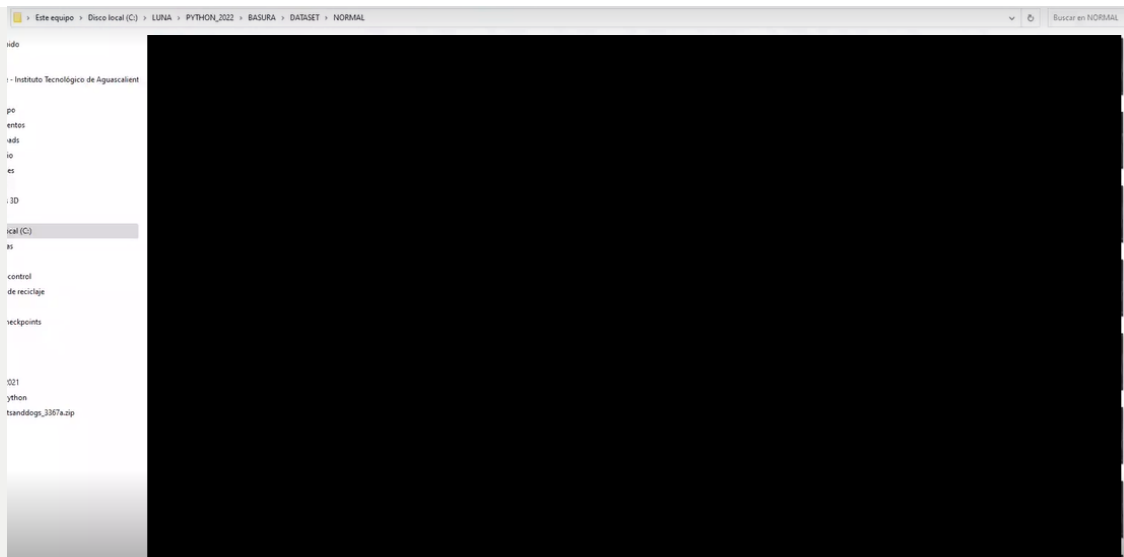
web_cam = cv2.VideoCapture(0) # WebCam
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml') #Carga Haarcascades
count = 0

while(web_cam.isOpened()):
    ret, frame = web_cam.read() #Lee WebCAM
    #Preprocesamiento
    grises = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rostro = faceCascade.detectMultiScale(grises, 1.5, 5)
    cv2.imshow("Creando Dataset",frame) #Muestra Lectura de webcam
    for(x,y,w,h) in rostro:
        cv2.rectangle(frame, (x,y), (x+w, y+h), (255,0,0), 4) #Enmarca rostro
        count += 1 #Rostros detectados
        cv2.imwrite("C:/LUNA/PYTHON_2022/BASURA/DATASET/NORMAL/normal_"+str(count)+".jpg", grises[y:y+h, x:x+w]) #Guarda rostro e
        #cv2.imshow("Creando Dataset",frame) #Muestra Lectura de webcam

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    elif count >= 400:
        break
#Liberamos la captura
web_cam.release()
cv2.destroyAllWindows()

```

Lo cual genera las imágenes del profesor



Luego abre otro archivo llamado CNN Face Detection:

```
from __future__ import print_function
import os

import tensorflow.keras as keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization
from tensorflow.keras.layers import Conv2D, MaxPooling2D

from tensorflow.keras.optimizers import RMSprop, SGD, Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
```

```
#CONFIGURACIÓN DE IMAGENES DE ENTRADA

#Directorio de imágenes train y validación
train_data_dir = 'C:/LUNA/PYTHON_2022/BASURA/DATASET/Entrenamiento'
validation_data_dir = 'C:/LUNA/PYTHON_2022/BASURA/DATASET/Validacion'

#Función de Lotes con modificaciones aleatorias
train_datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.4,
    height_shift_range=0.4,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    rescale=1./255)
validation_datagen = ImageDataGenerator(rescale=1./255) #Cambiar la escala de los valores en la matriz

#Generación de Lotes normalizados
batch_size = 32
img_rows, img_cols = 48, 48
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_rows, img_cols), #cada imagen será redimensionada a este tamaño
    color_mode='grayscale',
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True) #mezclar el orden
```

```

validation_generator = validation_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_rows,img_cols),
    color_mode='grayscale',
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True)

#####
#MODELADO DE CNN

num_classes = 3 #Feliz, Enojado
model = Sequential()

# Block-1 Esta capa crea un núcleo de convolución que se convoluciona con la entrada de la capa para producir un tensor de salida
model.add(Conv2D(32,3,padding='same',kernel_initializer='he_normal',input_shape=(img_rows,img_cols,1)))
model.add(Activation('elu'))
model.add(BatchNormalization()) #entradas cambian, lo que provoca la variedad de no estacionariedad.

model.add(Conv2D(32,3,padding='same',kernel_initializer='he_normal',input_shape=(img_rows,img_cols,1)))
model.add(Activation('elu'))
model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2,2)))#reduce la dimensionalidad de la imagen, tomando grupos de 2x2 y nos quedamos con el máximo
model.add(Dropout(0.2)) #La capa de abandono establece aleatoriamente las unidades de entrada en 0 para evitar sobreajuste

```

```

# Block-2
model.add(Conv2D(64,3,padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())

model.add(Conv2D(64,3,padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-3
model.add(Conv2D(128,3,padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())

model.add(Conv2D(128,3,padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-4
model.add(Conv2D(256,3,padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())

```

|| teams.microsoft.com está compartiendo tu pantalla. [Dejar de compartir](#) [Ocultar](#)

```

model.add(Conv2D(256,3,padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-5
model.add(Flatten())
model.add(Dense(64,kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
# Block-6
model.add(Dense(64,kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
# Block-7
model.add(Dense(num_classes,kernel_initializer='he_normal'))
model.add(Activation('softmax'))

print(model.summary()) #resumen de cada capa

model.compile(optimizer = Adam(lr=0.001), #Optimizer that implements the Adam algorithm.
              loss='categorical_crossentropy', #Calcula La pérdida de entropía cruzada entre Las etiquetas y Las predicciones.
              metrics=['accuracy']) #Calcula la frecuencia con la que Las predicciones son iguales a Las etiquetas.

```

```

#####
#ENTRENAMIENTO DE CNN

#Detenga el entrenamiento cuando una métrica monitoreada haya dejado de mejorar.
earlystop = EarlyStopping(monitor='val_loss',
                          min_delta=0, #min change
                          patience=5, #no change
                          verbose=1,
                          restore_best_weights=True
                          )

```

```

#Configuración para guardar un modelo
checkpoint = ModelCheckpoint(filepath='Modelo_3.h5', #Nombre
                             monitor='val_loss',
                             verbose=1,
                             save_best_only=True
                             mode='min'

```

```

        verbose=1,
        min_delta=0.0001)

callbacks = [earlystop,checkpoint,reduce_lr]
epochs=25
history=model.fit(
    train_generator, #1600 imagenes
    steps_per_epoch=None, #Hasta que se acaben Las imagenes de train segun Los Lotes es La epoca
    epochs=epochs,
    callbacks=callbacks,
    validation_data=validation_generator, #400 imagenes
    validation_steps=None) #Hasta que se acaben Las imagenes de validation segun Los Lotes es La epoca

```

Parece que el primer archivo se ejecuta al final. El segundo es primero y luego el tercero

Empecé a escribirlo:

```

from keras.models import load_model
from time import sleep
from keras.preprocessing.image import img_to_array
from keras.preprocessing import image
import cv2
import numpy as np

#

```

```
face_classifier = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml') #  
Carga Haarcascades
```

▼ **02-05-23**

Teoría de redes neuronales recurrentes

▼ **03-05-23**

Teoría de redes neuronales recurrentes

▼ **04-05-23**

Teoría de redes neuronales recurrentes

▼ **05-05-23**

Teoría de redes neuronales recurrentes

▼ **08-05-23**

Teoría de redes neuronales recurrentes

▼ **09-05-23**

Teoría de redes neuronales recurrentes

▼ **11-05-23**

Teoría de redes neuronales recurrentes

▼ **12-05-23**

Teoría de redes neuronales recurrentes

▼ **16-05-23**

Teoría de suavizado con medias móviles

▼ **17-05-23**

Teoría de suavizado con medias móviles

▼ **18-05-23**

Teoría de suavizado con medias móviles

▼ **19-05-23**

Teoría de suavizado con medias móviles

▼ **22-05-23**

Teoría de RNN

▼ **24-05-23**

Explicación del examen final

▼ **25-05-23**

Explicación del examen final

▼ **26-05-23**

Explicación del examen final

▼ **29-05-23**

No entré a clase

▼ **30-05-23**

No entré a clase

▼ **31-05-23**

No entré a clase

▼ **01-06-23**

No entré a clase

▼ **02-06-23**

No entré a clase