

## PROJECT 2 **Labelling**

SO: Sessió d'Orientació  
**Project 2 - Part 2**

Artificial intelligence

2023-2024  
Universitat Autònoma de Barcelona

## 1. Introducció

En aquesta pràctica, resoldrem un problema simple **d'etiquetatge d'imatges**.

Donat un conjunt d'imatges d'un catàleg de roba, desenvoluparem algorismes per aprendre com etiquetar imatges per tipus i color.

Per simplificar la pràctica, utilitzarem un conjunt petit d'etiquetes:

**8 tipus de roba i 11 colors bàsics.**

El sistema s'executarà en una imatge donada i retornarà una etiqueta per a un tipus de roba i una o més etiquetes per a colors. Implementareu els següents algorismes vistos a classe de teoria:

1. K-means (k-means): **Mètode de classificació no supervisada per trobar colors predominants.**




2. K-NN (k-nn) Mètode de classificació supervisada per classificar els tipus de roba.

En aquesta **2a part del Projecte**, ens centrarem en:

**l'algorisme K-NN**

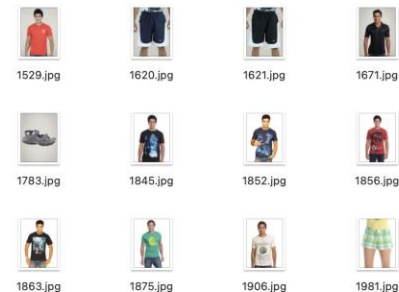
## 2. Fitxers requerits

Hem de tenir descarregades les següents carpetes:

1.  **Images:** Carpeta que conté el set d'imatges que utilitzarem.
  - A. **gt.json:** Fitxer que conté informació sobre la classe d'imatges a entrenar (Train).
  - B.  **Train:** Carpeta amb el set d'imatges que utilitzarem com a set d'entrenament. Sobre elles (**train set**) trobarem informació sobre quina classe pertany en el fitxer **gt.json**.
  - C.  **Test:** Carpeta amb el set d'imatges que volem etiquetar i de les quals no tenim informació, aquest serà el nostre set d'experimentació (**test set**).




Classes de Train data estan etiquetades per tipus i colors



Conjunt d'imatges de les carpetes Train/Test folders

## 2. Fitxers requerits

Per dur a terme la pràctica, hauràs de descarregar les següents carpetes:

**2.  Test:** Carpeta que conté el conjunt de fitxers necessaris per poder realitzar les proves sol·licitades en els scripts de prova (no els has d'utilitzar en els teus scripts, les funcions de prova els carreguen automàticament en el setUp).

`test_cases_knn.pkl`

**3. `utils.py`:** Conté un seguit de **functions** necessaries per convertir les imatges a color en altres espais, bàsicament les convertim a escala de grisos (grey-level) `rgb2gray()` i aconseguim els 11 colors bàsics `get_color_prob()`.

**4. `utils_data.py`:** Conté un seguit de **functions** necessaries per obrir i tractar les imatges. Com per exemple llegir les imatges i etiquetes amb `read_dataset`, .

**4. `KNN.py`:** Fitxer a on programareu les funcions necessaries per implementar K-NN per l'etiquetatge automàtic de forma.

**5. `TestCases_knn.py`:** Arxiu amb el qual podreu comprovar si les funcions que programeu en el fitxer `KNN.py` donen el resultat esperat.

### 3. Preliminars

Abans de començar a programar, és molt recomanable entendre la classe amb la qual anem a treballar:

#### **KNN**

KNN És una classe dissenyada per carregar totes les dades, transformar-les i executar l'algorisme KNN

Classe KNN:

#### **Atributs**

```
self.labels = np.array(labels)
self.train_data = np.random.randint(8, size=[10, 4800])
self.neighbors = np.random.randint(k, size=[test_data.shape[0], k])
```

### 3. Preliminars

Abans de començar a programar, és molt recomanable entendre la classe amb la qual anem a treballar:

#### **KNN**

KNN És una classe dissenyada per carregar totes les dades, transformar-les i executar l'algorisme KNN

Classe KNN:

#### **Funcions:**

```
def _init_train(self, train_data):    inicialitzar dades d'entrenament
def get_k_neighbours(self, test_data, k):  càlcul dels veïns més propers
def get_class(self): assignar la classe segons la majoria de vots
def predict(self, test_data, k): predir la classe al qual cada imatge pertany
```

## 4. Què és el que hem de programar?

Has de codificar l'algorisme de classificació K-NN per poder etiquetar la forma de cada imatge.

Haureu de realitzar quatre tasques:

4.1. Funció per **inicialitzar K-NN: `_init_train`**

4.2. Funció que troba els k veïns més propers: **`get_k_neighbours`**

4.3. Funció que selecciona l'etiqueta que més apareix als veïns: **`get_class`**

4.4. Funció que assigna l'etiqueta de classe a les dades de test: **`predict`**

## 4.1. Funció per inicialitzar K-NN: `_init_train`

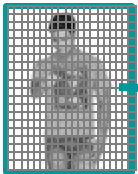
### Sessió de Teoria

Definició d'espai de característiques

- 1) Eliminarem el COLOR, ja que no el necessitem per representar la forma



- 2) Agafarem directament els píxels de la imatges com la característica de cada posició de la imatge.



$(1, 0, 0.5, \dots, 1)$  -> Vector de  $1 \times 4800$   
on  $4800 = 80 \times 60$  píxels

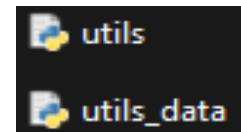
### Sessió pràctica

`_init_train`: Funció que s'assegura que la variable `train_data` de la classe KNN, la qual conté el nostre conjunt d'entrenament, tingui el format float, i tot seguit n'extreu les característiques

`train_data`: dimensions de  $N \times 4800$ , on

- $N$ : nombre d'imatges de `train_data`
- 4800: nombre de píxels d'una imatge

### Recordeu!



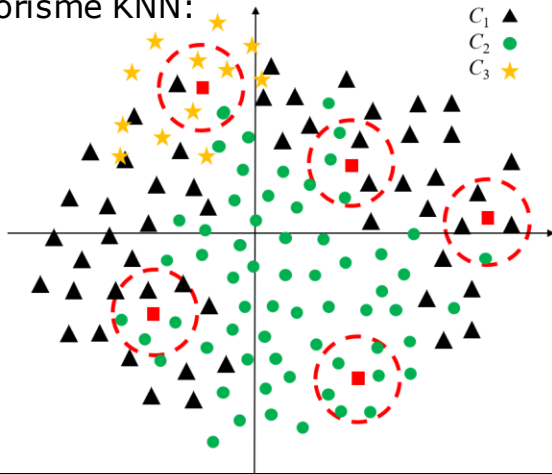
`rgb2gray()`



## 4.2. Funció que troba els k veïns més propers: **get\_k\_neighbours**

### Sessió de Teoria

L'algorisme KNN:



**Funció** Classificar( $\vec{y}$ )

1. **Per** ( $\vec{x}^j \in X$ ) fer  $L = \text{inserir}([d(\vec{y}, \vec{x}^j), C_j], L)$  **Fper**
  2.  $\text{veïns} = \text{Primers\_k}(\text{ordenar\_d}(L))$
  3. **Si** ( $\text{comptar}(\text{veïns}, C_1) > \text{comptar}(\text{veïns}, C_2)$ )  
    **llavors**
  4.      $\vec{y} \in C_1$
  5. **Sinó**  $\vec{y} \in C_2$
  6. **Fsi**
- FFunció**

### Sessió pràctica

**get\_k\_neighbours**: Funció que pren com a entrada el conjunt de test que volem etiquetar (**test\_data**) i fa el següent:

1. Canvia les dimensions de les imatges de la mateixa manera que ho hem fet amb el conjunt d'entrenament. (**test\_data**)
2. Calcula la distància entre les mostres del **test\_data** i les del **train\_data**.
3. Guarda a la variable de classe **self.neighbors** les K etiquetes de les imatges més pròximes per a cada mostra del test.

**Important:** El càlcul de distàncies és molt costós si es fa en un bucle.  
Es recomana usar la funció **cdist**, de la llibreria **scipy.spatial.distance**,

## 4.2. Funció que troba els k veïns més propers: **get\_k\_neighbours**

### Sessió de Teoria

#### Exemple

:



$$\vec{y} = [1, 1, 1, 0.2, 0.5, 0, 0, 1, \dots, 1]$$

$$\begin{pmatrix} x_1^1 & \dots & x_{4800}^1 & C_1 \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{n_1} & \dots & x_{4800}^{n_1} & C_1 \\ x_1^1 & \dots & x_{4800}^1 & C_2 \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{n_2} & \dots & x_{4800}^{n_2} & C_2 \\ \vdots & \ddots & \vdots & \vdots \\ x_1^1 & \dots & x_{4800}^1 & C_k \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{n_k} & \dots & x_{4800}^{n_k} & C_k \end{pmatrix}$$

$$d=23$$

$$[1, 1, 1, 0.1, 0.4, 0.2, 1, 1, \dots, 1]$$



$$d=59$$

$$[0, 0, 0, 0.7, 0.5, 1, 0, 1, \dots, 1]$$



$$d=103$$

$$[0, 0, 0, 0.2, 1, 1, 0, 0.4, \dots, 0]$$



#### Funció Classificar( $\vec{y}$ )

1. **Per** ( $\vec{x}^j \in X$ ) **fer**  $L = \text{inserir}([d(\vec{y}, \vec{x}^j), C_j], L)$  **Fper**
  2. **veïns** =  $\text{Primers\_k}(\text{ordenar\_d}(L))$
  3. **Si** ( $\text{comptar}(\text{veïns}, C_1) > \text{comptar}(\text{veïns}, C_2))$  **llavors**
  4.  $\vec{y} \in C_1$
  5. **Sinó**  $\vec{y} \in C_2$
  6. **Fsi**
- FFunció**

### Sessió pràctica

Més detalls:

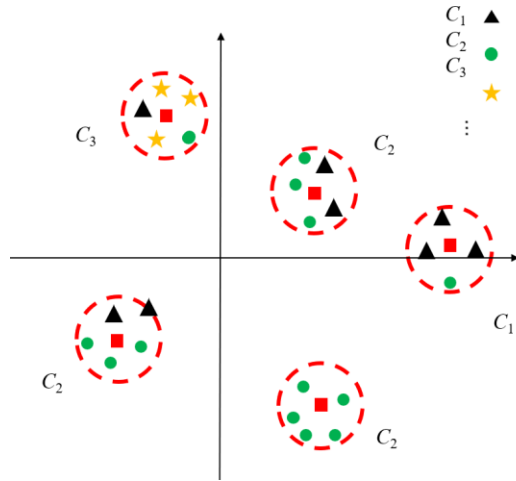
**get\_k\_neighbours**: Funció que pren com a entrada el conjunt de test que volem etiquetar (**test\_data**) i fa el següent:

- **Input:** **test\_data** i **k**
- **Retorna:** **self.neighbors**

```
def get_k_neighbours(self, test_data, k)
: param test_data: array that has to be
    shaped to a Nx D matrix
    (N points in a D dimensional
    space)
: param k: number of neighbors to look at
: return: the matrix self.neighbors of NxK
    where
    ij-th entry is the j-th nearest
    train point to the i-th test
    point
```

## 4.3. Funció que selecciona l'etiqueta que més apareix als veïns: **get\_class**

### Sessió de Teoria



### Sessió pràctica

**get\_class**: Funció que comprova quina és l'etiqueta que més vegades ha aparegut a la variable de classe neighbors per a cada imatge del conjunt de test (test\_data),

Retorna un array amb aquesta classe per a cada imatge. Aquest array tindrà tants elements com punts s'hauran entrat a la funció predict.

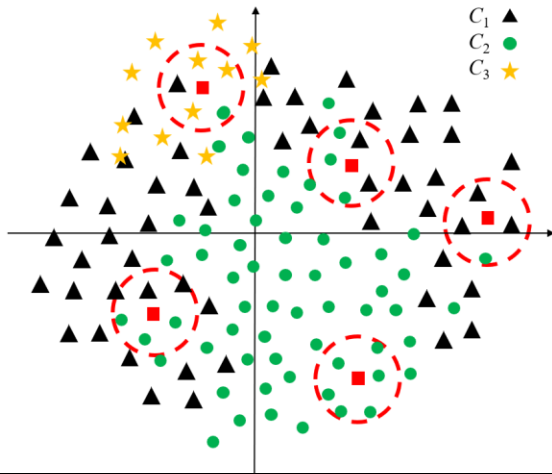
**Funció** Classificar( $\vec{y}$ )

```
1. Per ( $\vec{x}^j \in X$ ) fer  $L = \text{inserir}([d(\vec{y}, \vec{x}^j), C_j], L)$  Fper  
2.  $\text{veïns} = \text{Primer}_k(\text{ordenar}_d(L))$   
3. Si ( $\text{comptar}(\text{veïns}, C_1) > \text{comptar}(\text{veïns}, C_2)$ )  
   llavors  
4.    $\vec{y} \in C_1$   
5. Sinó  $\vec{y} \in C_2$   
6. Fsi  
FFunció
```

- **Input:** -
- **Retorna:** array class\_labels

## 4.4 Funció que assigna l'etiqueta de classe a les dades de test: **predict**

### Sessió de Teoria



### Sessió pràctica

**predict**: Funció que pren com a entrada el conjunt de test que volem etiquetar (`test_data`) i el nombre de veïns que volem tenir en compte (`k`), busca els seus veïns utilitzant

- **Input:** `test_data`, `k`
- **Retorna:** `array class_labels`

Aquesta funció bàsicament crida a les dues anteriors

**Funció** `Classificar( $\vec{y}$ )`

1. **Per** ( $\vec{x}^j \in X$ ) **fer** `L=inserir([d( $\vec{y}, \vec{x}^j$ ),  $C_j$ ], L)` **Fper**
2. `veïns=Primers_k(ordemar_d(L))`
3. **Si** (`comptar(veïns,  $C_1$ ) > comptar(veïns,  $C_2$ )`)  
    **llavors**
4.  `$\vec{y} \in C_1$`
5. **Sinó**  `$\vec{y} \in C_2$`
6. **Fsi**

**FFunció**

`get_k_neighbours`

`get_class`

## Entrega Part 2

Per a l'avaluació d'aquesta segona part de la pràctica haureu de pujar al Campus Virtual el vostre fitxer **KNN.py** que ha de contenir el **NIU** de tots els membres del grup a la variable authors (a l'inici de l'arxiu). Els NIUs s'hauran d'afegir encara que els grups sigui d'una sola persona (e.g., [1290010,10348822] o [23512434]).

L'entrega s'ha de fer abans del dia **28/04/2024 at 23:55**.

**ATENCIÓ!** és important que tingueu en compte els següents punts:

1. La **correcció** del codi es fa de manera **automàtica**, per tant, assegureu-vos de penjar els arxius amb la nomenclatura i format correctes. Si no ho poseu bé la nota serà un 0. (No canvieu el nom del fitxer o els imports al principi del fitxer)
2. El codi està sotmès a **detecció automàtica** de plagis durant la correcció.
3. Qualsevol part del codi que no estigui dins de les funcions de l'arxiu Kmeans.py **no** podrà ser **avaluada**, per tant, no modifiqueu res fora d'aquest arxiu.
4. Per evitar que el codi entri en bucles infinits hi ha un **límit de temps** per a cada exercici, per tant si les vostres funcions triguen massa les considerarà incorrectes.

## Recordeu el que es diu a la **guia docent sobre copiar o deixar copiar ....**

Sense perjudici d'altres mesures disciplinàries que s'estimin oportunes, i d'acord amb la normativa acadèmica vigent, les **irregularitats comeses per l'alumnat** que puguin conduir a una variació de la qualificació es qualificaran amb un zero (0). Les activitats d'avaluació qualificades d'aquesta forma i per aquest procediment no seran recuperables. Si és necessari superar qualsevol d'aquestes activitats d'avaluació per aprovar l'assignatura, aquesta assignatura quedarà suspesa directament, sense oportunitat de recuperar-la en el mateix curs. Aquestes irregularitats inclouen, entre d'altres:

- còpia total o parcial d'una pràctica, informe, o qualsevol altra activitat d'avaluació;
- deixar copiar;
- ús no autoritzat i/o no referenciat de la IA (p. ex, Copilot, ChatGPT o equivalents) per a resoldre exercicis, pràctiques i/o qualsevol altra activitat avaluable;
- presentar un treball de grup no fet íntegrament pels membres del grup;
- presentar com a propis materials elaborats per un tercer, encara que siguin traduccions o adaptacions, i en general treballs amb elements no originals i exclusius de l'alumnat.
- tenir dispositius de comunicació (com telèfons mòbils, smart watches, etc.) accessibles durant les proves d'avaluació teòric-pràctiques individuals (exàmens).

En resum: **copiar, deixar copiar o plagiar** en qualsevol de les activitats d'avaluació equival a un SUSPENS amb **nota inferior o igual a 3,0**.