

# **INTRODUCCIÓ A LA PRACTICA**

Sens planteja resoldre un problema simple d’etiquetatge d’imatges. Donat un conjunt d’imatges d’un catàleg de roba desenvoluparíem els algorismes que permetessin aprendre a etiquetar automàticament imatges per tipus de peça i per color. Com que el problema podia ser molt complex es va limitar a paraules en angles, 8 tipus de roba i 11 colors basics a més de treballar amb imatges de baixa resolució.

Es plantegen 3 problemes a resoldre:

* **Etiquetatge automàtic de color (no supervisat)**

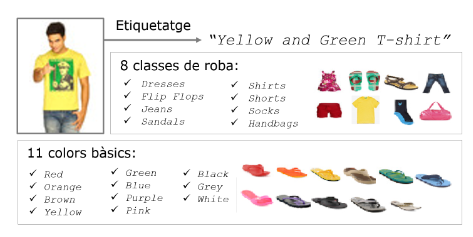
Es va decidir realitzar un agrupament no supervisat de punts utilitzant l’algorisme *K-means,* on es van implementar 10 funcions en les quals destacaven: fit(), withinClassDistance(), find\_best\_K(max\_K), i get\_colors(centroids).

* **Etiquetatge automàtic de forma (supervisat)**

Per a organitzar el tipus de roba, es va decidir utilitzar l’algorisme *KNN,* on es van implementar 3 funcions per a agrupar-les en classes.

* **Implementació del cercador**

Una vegada els mètodes son implementats es combinarien per a crear diferents test d’anàlisis i arribar a tenir una interfície similar a la d’una botiga online.



A continuació es parla amb més detall de cada una d’aquests problemes.

# **METODES D’ANALISI IMPLEMENTATS**

## **METODES D’ANALISIS QUALITATIUS**

Els mètodes d’anàlisis qualitatius son funcions que ens permeten avaluar d’una manera visual els nostres classificadors. Sens va donar tres funcions a fer de les quals es va decidir fer les tres: **Retrieval\_by\_color**, **Retrieval\_by\_shape** i **Retrival\_combined**. Aquestes funciones retornaran les imatges que contenen les etiquetes d’una pregunta que es passa per paràmetre.

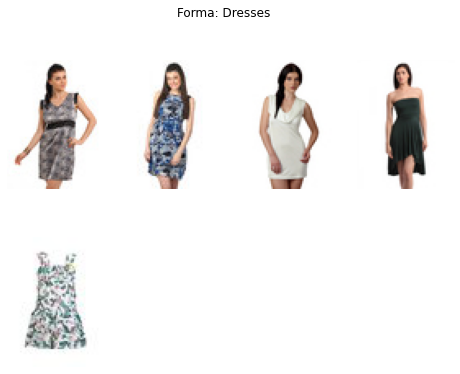
En la primera classe de practiques de la tercera part es va comentar que si es tenia una funció es tenien totes. Amb aquesta primera idea, es va començar a fer un esborrany de la primera funció: *‘Retrieval\_by\_color’*. No va costar arribar a una possible solució.

Es va fer la resta de funcions trobant una diferencia en cada una de les funcions:

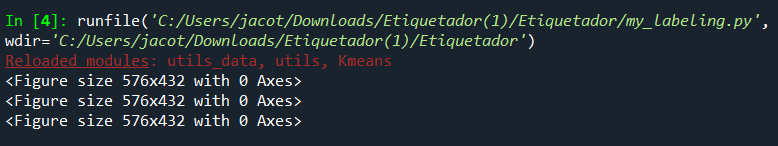
* **Retrieval\_by\_color:** Es va posar: ‘if pregunta in clase:’, perquè si s’hagués posat un == , al tractar-se d’una cadena, aquesta hauria fallat.
* **Retrieval\_by\_shape:** Al contrari que la funció anterior, en aquesta era necessària un == al tractar-se d’una única etiqueta.
* **Retrieval\_combined:** Com el seu nom indica, es va combinar les dos funcions en una sola, amb dos condicions *(forma i color)* dins un bucle for.

Les funcions no van suposar un problema.

Per comprovar que funcionés, un membre del grup va decidir fer els seus propis tests per a cada una de les tres funcions. Per fer-ho, es va informar de les funcions a utils\_data *(visualize\_retrieval),* llegit atentament el pdf de presentació de la part i va tomar inspiració dels tests proporcionats pels professors *(TestCases\_kmeans y TestCases\_knn).* Agafant test\_color\_labels per Retrival\_by\_color y test\_class\_labels per Retrieval\_by\_shape, el test va donar un resultat correcte en base al que se li demana.



**Imatges que retorna el test color y shape amb les etiquetes predeterminades**

El problema va ser amb Retrieval\_Combined el qual no mostrava cap imatge i imprimia aquest missatge a la terminal.

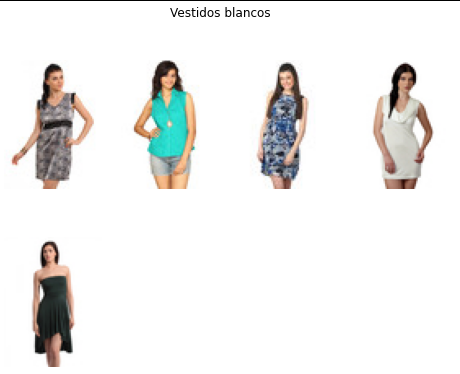
El problema es trobava en la condició del color, que només retornava les imatges correctes si es posava amb in, al contrari que la seva part per separat.



**Imatge que retorna el test combined amb les etiquetes predeterminades**

En un principi, els resultats donen correctament amb la primera implementació, però més tard es va adonar que no s’estava utilitzant les etiquetes implementades pel nostre Kmeans i Knn, i només s’estava comprovant que la funció fos correcta. El membre del grup que va fer els test va crear una funció on, en un bucle iterava les diferents imatges del conjunt, calculant les etiquetes de color i en la mateixa funció, calcular les etiquetes de forma però fora del bucle ja que ja havia un bucle implementat en la mateixa funció del Knn.

El resultat va donar imatges diferents però no mostrava signes de fallar o de donar un resultat inesperat.





**Imatges que retorna el test color amb la funció pròpia.**

***Observació:*** *Existeix la possibilitat de que el color no sigui l’indicat ja sigui per un píxel que el programa detecta com del color que es busca o que es detecta el color de la peça de roba externa que no ha sigut eliminada a l’hora de fer la retallada.*

## **METODES D’ANALISIS QUANTITATIUS**

Els mètodes quantitatius s’encarreguen d’observar y treure resultats per a poder qualificar el treball fet per els algorismes implantats a la practica. Per a aquesta part sens ofereix realitzar 3 mètodes: Kmean\_statistics, Get\_shape\_accuracy, Get\_color\_accuracy.

De moment tenim implementada la funció **Kmean\_statistics**, amb la qual hem pogut observar la distancia entre classes (WCD) així com el temps necessari per a convergir per a cada valor de K que ve des de K = 2 fins a Kmax, el qual hem escollit que sigui K = 7. (6 nombres de ‘K’ diferents)

Els resultats els dividirem en dues parts, primer analitzarem la **distancia intra-class**.

**WithinClassDistance:** [2377.4794200537094, 1075.0367413718914, 729.7420840406991, 577.6647183568346, 484.0979100647066, 415.5614643165185]

Com podem observar al gràfic, la WCD cau dràsticament quan el numero de classes ‘K’ va pujant, això es a causa de que contra mes opcions de classificació tenim, els punts es van dispersant mes entre les diferents classes y que, per tant, les distancies entre classes cada vegada siguin mes curtes.

D’altra banda, tenim l’anàlisi del **temps necessari per convergir** en funció del nombre de classes:

Time: [4.844397783279419, 20.479047298431396, 39.41282272338867, 48.98620820045471, 103.39242196083069, 123.68818950653076]

Texto, Carta

Descripción generada automáticamenteAmb aquesta gràfica podem observar com a mesura que augmentem el numero de classes, el temps per a l’execució del k-means va augmentant també. Per entendre millor perquè passa això podem observar la formula de la complexitat del algorisme el qual es:

Com el que variem es el nombre de classes, inevitablement això farà que necessitem mes iteracions per arribar a la convergència, per tant, cada vegada que incrementem K, estem augmentat dues de les tres variables que conformen la complexitat, deixant com a resultat un increment substancial del temps.

Gráfico, Gráfico circular

Descripción generada automáticamenteEl segon Metode quantitatiu implemntat ha estat

**90,951% accuracy**

El get\_shape\_acuracy, aquest ens indica a partir del

conjunt de prova, totes les etiquetes de forma que ha

estat capaç l' algoritme de classificar correctament.

Com podem observar al gráfic, el percentatge d’encerts

ha estat d'un 90,95%. És a dir, 9 de cada 10 etiquetes

han estat assignades correctament.

# **MILLORES SOBRE EL KMEANS I KNN**

S’han implementat varies millores sobre el Kmeans i el KNN. Primerament, en el Kmeans, en la funció init\_centroids aquests es poden inicialitzar amb tres mètodes. Aquests poden ser el mètode ‘first’, ‘random’ i ‘custom’. El mètode ‘first’ inicialitza els centroides assignant-lis els primers K punts de la imatge. El ‘random’ fa el mateix però amb punts aleatoris. I el ‘custom’ permet a l’usuari escollir una estrategia d’inicialització. Es modificarà el resultat dels centroides, ja que el mètode d’inicialització no és el mateix, però els paràmetres d’entrada de la funció no han cambiat ja que els K punts són els mateixos però amb ordre diferent. Abans d’implementar aquesta millora els centroides sempre s’inicialitzaven amb el mètode ‘first’.

A més en el Kmeans s’han implementat dos heurístiques més a part de WithinClassDistance. S’ha implementat interClassDistance, que calcula la distancia entre classes. Tindrà el paràmetre d’entrada self i en cambi de fer els càlculs per calcular la distancia entre la mateixa classe es faran els de calcular la distancia entre diferents classes. El resultat será diferente a WithinClassDistance ja que es calcula la distancia d’una altra cosa. També s’ha implementat CoeficientFisher que calcula la distància de Fisher. La funció també es semblant a les dues anteriors però fa els càlculs per calcular el Coeficient de Fisher. El paràmetre d’entrada segueix sent self però el resultat cambia respecte als anteriors perquè es calcula un altre concepte. Per aquesta última funció s’ha hagut d’afegir dos paràmatres a l’inicialització de la classe Kmeans: N i centroide global.

S’ha millorat també find\_best\_K de la classe Kmeans. La funció segueix igual excepte que s’ha modificat el valor del llindar. D’aquesta forma els càlculs tenen més exatitud i es troba una millor K que anteriorment. Per tant només es modifica el llindar i el resultat també cambiarà, ja que trobarà una millor K. Aquests són els resultats del llindar=20, el llindar=10, i el llindar=50. Com es pot observar hi ha un gran cambi amb llindar=50 respecte als dos altres.

Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

Finalment també s’ha modificat KNN. Per obtenir millors resultats es cambiarà la distancia que s’utilitza (abans s’utillitzava la distància euclídia). Per tant al modificar aquest paràmetre es cambiarà el resultat obtingut. Aquest será millor respecte l’anterior ja que s’utilitzarà una distancia més precisa que la que s’utilitzava abans de modificar el codi. Les distàncies a les quals es pot canviar son “minkowski”, i “sqeuclidian”.