INFORME PRÀCTICA 2

ETIQUETATGE

Matemàtica Computacional i Analítica de Dades

Inteligència Artificial

Jon Carles Montero (1587634), Genís Ruiz (1633623),

Joaquin Flores (1587646) i Manel Carrillo (1633426)

14/05/2023

INDEX:

1. Introducció a la pràctica

En aquesta pràctica hem implementat un classificador d'imatges per color i forma gràcies a la implementació dels algorismes K-means i KNN respectivament amb les seves funcions corresponents.

A més, hem creat un seguit de funcions que ens han permès analitzar que tan bé funcionen, podent fer anàlisi qualitativa i quantitatius sobre els temps d'execució, precisions, etc.

Finalment, hem plantejat diverses millores pels dos algorismes amb l'objectiu de millorar les seves inicialitzacions o bé de les seves precisions, aplicant transformacions al conjunt d'imatges, per exemple.

2. Mètodes d'anàlisi implementats

Per avaluar el rendiment dels algorismes K-means i KNN, hem implementat diversos mètodes d'anàlisi quantitatius i qualitatius.

2.1 Funcionalitat del K-means

El K-means és un algorisme que serveix per a classificar un conjunt de dades no etiquetades (aprenentatge no supervisat), el qual utilitzem per classificar el color de les nostres imatges. Usem només train_imgs (pel simple fet de tenir més imatges que el test_imgs, ja que no hi ha un conjunt d'entrenament) i per l'accuracy utilitzem test_color_labels (per a tenir les etiquetes vertaderes amb el conjunt que utilitzem). La funció no retorna res, però fem servir les seves funcions fit, find_bestk, etc.

2.2 Funcionalitat del KNN

El KNN és un algorisme que serveix per a classificar un conjunt de dades etiquetades, el qual utilitzem per a distingir els diferents tipus de roba que ens podem trobar. Usem train_data per a entrenar el classificador i després test_data per a trobar l'accuracy del classificador. La funció ens retorna una llista amb les prediccions de roba i un altre amb el percentatge amb el qual aquesta peça ha sigut escollida.

2.3 Mètodes Qualitatius i resultats

En aquesta part de l'informe presentarem els mètodes qualitatius que hem implementat per a la cerca d'imatges en el nostre projecte.

Aquests mètodes ens permeten cercar imatges en funció del color o de la forma que es desitgi.

Els mètodes qualitatius implementats són el " **Retrieval by Color",** que busca imatges segons el color, el "**Retrieval by Shape"** que busca imatges que coincideixin amb la forma demanada, i el tercer "**Retrieval Combined**" que combina les dues condicions obtenint d'aquesta manera peces de roba del color i forma que es demani.

En les següents seccions, es detallaran els diferents mètodes i es presentaran exemples de les cerques realitzades amb cada mètode.

2.3.1 Retrieval by Color

El primer mètode qualitatiu que s'ha implementat és el retrieval by color. Aquesta funció té com a objectiu obtenir imatges que continguin el/els colors específics que es demanen a través d'una llista de strings (que poden tenir un sol element o més d'un).

La funció implementada va a través de les imatges i les seves etiquetes, les quals descriuen el contingut de la imatge, i comprova si els colors indicats estan presents en l'etiqueta de la imatge. Si hi són, les afegim a una llista nova que contindrà les imatges amb els colors demanats.

Per altra banda, hem volgut ordenar les imatges per la probabilitat de què els centroides calculats siguin del color especificat. Aquesta probabilitat es calcula a la funció get colors del kmeans.

Execució Retrieval by Color

Exemple 1: Cerca color blau

```
selected_images = retrieval_by_color(train_imgs, train_color_labels, ["Blue"])
print(len(selected_images))
visualize_retrieval(selected_images,10)
```

=

En aquest primer exemple hem volgut buscar les peces de roba que contenien blau i ens han sortit un total de 555 peces.

Exemple 2: Cerca color vermell

```
selected_images = retrieval_by_color(train_imgs,
train_color_labels, ["Red"])
print(len(selected_images))
visualize_retrieval(selected_images,10)
```

Ara hem volgut buscar les peces de roba amb color vermell i ens han sortit un total de 262 peces de roba.

Exemple 3 : Cerca color

```
selected_images =
retrieval_by_color(train_imgs,
train_color_labels,
[[“Blue”","Green"])
visualize_retrievals(selected_images,4)
```

En aquest exemple hem volgut buscar per més d'un color passant aquests com una llista a la funció.
Ens han sortit un total de 13 imatges.

2.3.2 Retrieval by Shape

El segon mètode qualitatiu que hem implementat és el retrieval by shape, l'objectiu del qual és obtenir imatges que continguin una forma específica que se li especifica en el programa.

Per a fer-ho, aquesta funció itera a través de totes les imatges, les seves etiquetes i les seves probabilitats. Si la forma específica que s'està buscant coincideix amb la descripció de l'etiqueta, la imatge s'afegeix a una llista.

Aquesta llista que conté les imatges amb la forma demanada. Aquests s'ordenaran per probabilitats (les quals aconseguim de la funció "get_class" del KNN).
Aquesta funció fa una votació entre les classes dels veïns més propers de cada imatge per a determinar la classe més probable a la qual pertany (en aquest cas, tipus de roba)

La funció calcula el percentatge de vots que ha aconseguit la classe més probable entre els veïns més propers de cada imatge, això es deu a la fucnció get class pròpia del KNN, i aquest percentatge és el que utilitzem com a probabilitat per a ordenar les imatges que obtenim amb el mètode "retrieval by shape".

Així doncs, aconseguim una llista d'imatges ordenades per probabilitat, és a dir, les imatges amb una probabilitat més alta (amb una major coincidència de la forma desitjada) apareixen primer en la llista.

Execució Retrieval by Shape

Exemple 1: Cerca de "Flip Flops"

```
selected_images = retrieval_by_shape(test_imgs,class_labels,probabilities,"Flip Flops")
print(len(selected_images))
visualize_retrieval(selected_images,10)
```

En aquest primer exemple hem volgut buscar Flip Flops i ens han sortit un total de 117 Flip Flops.

Exemple 2: Cerca de "Shorts"

Un altre exemple seria la cerca de pantalons. En aquest cas ens han sortit un total de 88 Pantalons.

2.3.3 Retrieval Combined

Per al mètode retrieval combined, vam utilitzar una combinació dels mètodes anteriors per buscar imatges que coincideixen tant en forma com en color. En primer lloc, vam executar el mètode retrieval by color per buscar les imatges amb el color desitjat. A continuació, vam aplicar el mètode retrieval by shape per obtenir el tipus de roba desitjat.

Finalment, per a assegurar-nos que les imatges seleccionades compleixen ambdós criteris, vam fer una intersecció entre les llistes de roba del color i del tipus de roba. Això ens va permetre aconseguir una llista de les imatges que coincideixen tant en forma com en color.

Execució Retrieval by Shape

Exemple 1: Cerca de "Red Flip Flops"

```
Knn = KNN(train_imgs,train_class_labels)
class_labels, probabilities = Knn.predict(test_data=test_imgs,k=5)
selected_images = retrieval_combined(test_imgs,class_labels,probabilities,test_color_labels,"Flip Flops",["Red"])
print(len(selected_images))
visualize_retrieval(selected_images,10)
```

En aquest primer exemple, hem volgut buscar flip flops amb color vermell i ens han sortit un total de 21.

Exemple 2: Cerca de "Blue Socks"

Knn = KNN(train_imgs,train_class_labels)

class_labels, probabilities = Knn.predict(test_data=test_imgs,k=5)

selected_images=retrieval_combined(test_imgs,class_labels,probabilities,test_color_labels,"Socks", ["Blue"])

print(len(selected_images))

visualize_retrieval(selected_images,10)

Per aquest últim exemple hem buscat mitjons de color blau i ens han sortit un total de 12.

2.4 Mètodes Quantitatius i resultats

En aquesta secció presentem els mètodes quantitatius que hem utilitzat per analitzar les dades del nostre projecte.

Els mètodes usats són el "K-means Statistics", el "Get Shape Accuracy" i el "Get Color Accuracy". El primer ens ha permès com obtenir el funcionament òptim del K-means, el segon i el tercer ens han permès analitzar les precisions dels algorismes corresponents:

2.4.1 Kmean Statistics

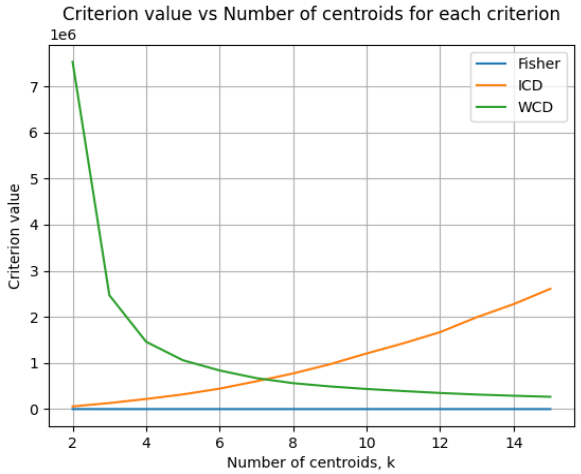
L'objectiu d'aquesta funció és mostrar una sèrie de gràfics i dades que ens permetin entendre amb més profunditat el funcionament del K-means. Per fer-ho, hem creat tres tipus gràfics, on es representa en cada un l'execució del fit del K-means donat el conjunt del train_imgs.

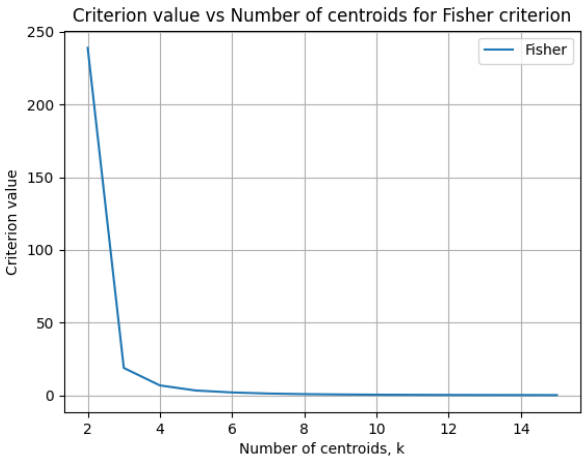
Com el K-means processa cada imatge de manera individual, hem calculat per a cada criteri i cada variable d'interès la mitjana dels resultats obtinguts , obtenint:

- **Valor del criteri o heurística en funció del nombre de centroides**:

En aquest primer gràfic hem pogut comprovar el que havíem vist a teoria, que la ICD creix i la WCD i la distància de Fisher decreix.

A causa de les dimensions de l'eix vertical (en milions), no es pot apreciar bé els diferents valors que pren la distància de Fisher (en centenars).



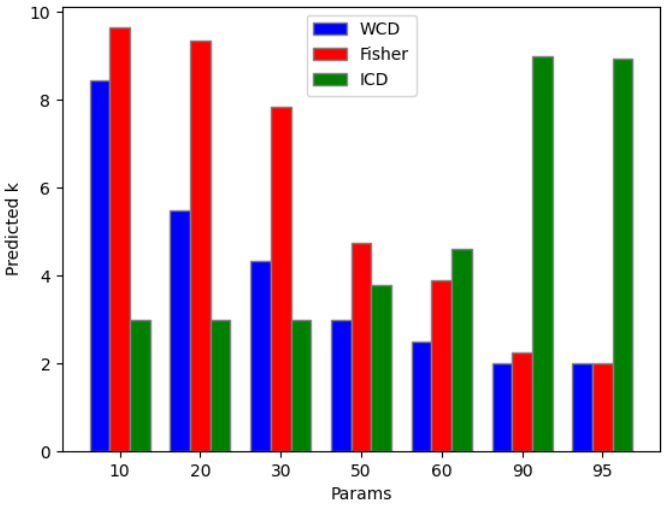


Una anàlisi molt útil serà veure l'elbow o punt de canvi de tendència, que ens pot determinar quina serà la k idònia.

Com bé es pot veure el WCD i Fisher tindran grans decreixements inicials i a poc a poc s'anirà estabilitzant. Per altra banda, la ICD tindrà un creixement inicial baix que anirà augmentant (al revés que els altres).

- **Millor k en funció del paràmetre:**

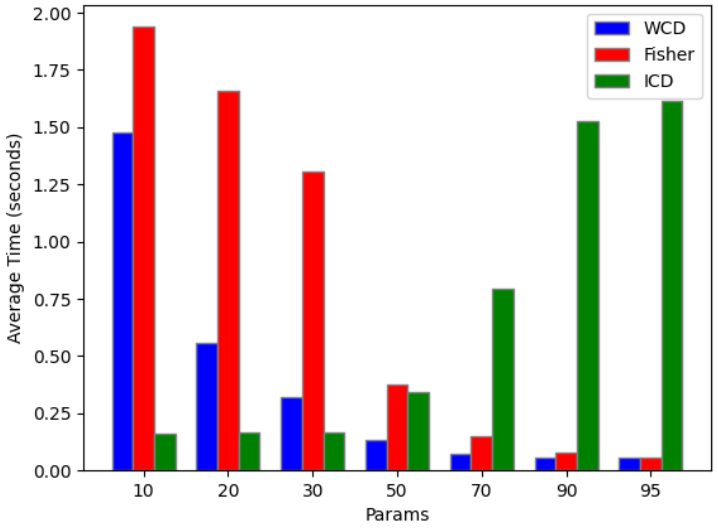
Per tal d'obtenir la millor k en funció del paràmetre, hem executat el find_bestk per a les tres heurístiques fent variar el valor del paràmetre, simbolitzant el "llindar de canvi". Com es pot apreciar en gràfic de barres podem observar dues tendències diferents. En primer lloc, WCD i Fisher troben una millor k més alta a mesura que fem petit el paràmetre. Això es deu a la interpretació que fem a la funció find_bestk, on per a aquests



dos criteris retornem la k si la relació (diferència en tant per 100 de la WCD i WCD anterior) és més gran que 100 menys el paràmetre.
Per tant, un paràmetre petit implicarà un canvi petit, havent d'executar més cops el K-means per tal de trobar aquelles k que estabilitzin el valor de la heurística.
Amb el ICD fem una cosa equivalent, mirant que la relació (sempre major de 100) sigui més gran que el paràmetre més 100. En conseqüència, per a valors petits, les millors k seran baixes perquè la funció inicialment creix "lentament". El colze de la funció el trobarem a l'altra banda per a valors propers a 80, on començarà a créixer de forma més ràpida.
És important fixar-se en l'heurística WCD és la que ens permet obtenir k més baixes al colze.

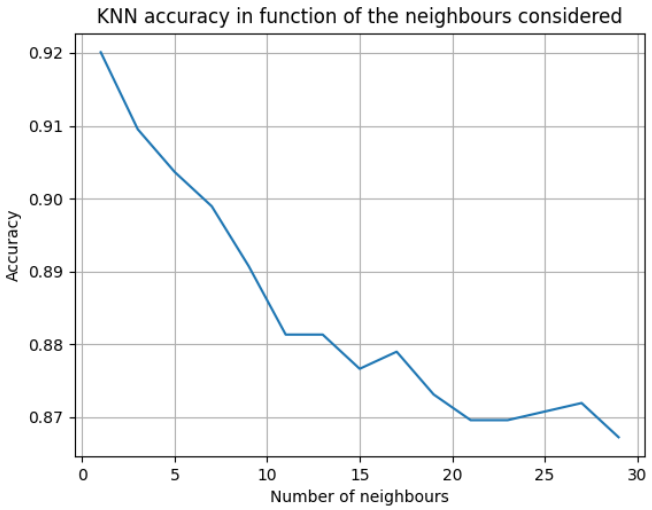
- **Temps d'execució (en segons) en funció del paràmetre:**

Com ens podem imaginar per la manera en com hem plantejat la funció, a una major k predita obtindrem un major temps d'execució. Aquest segueix la mateixa estructura que el gràfic anterior:



2.4.2 Get Shape Accuracy

Per tal de calcular la precisió del KNN hem creat una funció que donades unes etiquetes de classe troba quantes s'assemblen o no a les vertaderes. Ho hem pogut programar simplement contant el nombre de coincidències entre el nombre total d'imatges.



Per tal de visualitzar com afectava el nombre de veïns a considerar (k) hem creat un bucle i hem representat la precisió obtinguda d'executar el fit amb diferents k (amb el conjunt d'entrenament train_imgs i avaluant-lo amb el test_imgs). Els resultats són els següents:

Com podem observar, a mesura que augmentem la k, la precisió acostuma a decréixer. La màxima que hem pogut obtenir és per a k=1 un 92% (també es dona en k=2, però hem considerat una representació cada dos k per simplificar els càlculs).

Mirant de fer l'anàlisi per les cropped_images creiem que no té gaire sentit, ja que en moltes d'elles s'ha fet un ajust tan gran que només es veu el color d'aquestes i, per tant, la distinció entre les formes dels diferents objectes seria bastant aleatòria. A més, en tenir diferents mides, s'hauria d'ajustar el seu espai de característiques per fer l'anàlisi adequadament.

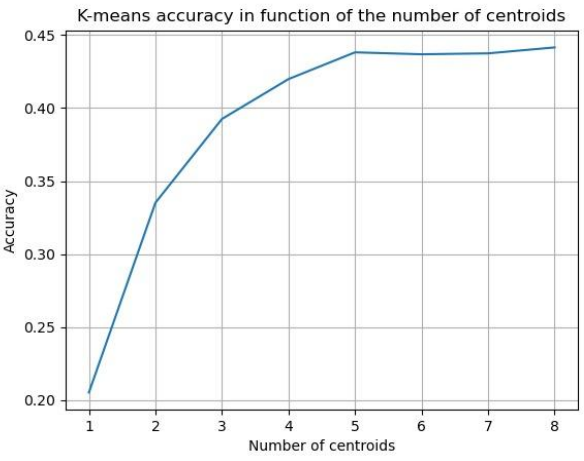
2.4.3 Get Color Accuracy

A l'hora de trobar la precisió del K-means hem hagut de tenir en compte com avaluaríem haver encertat parcialment els colors d'una imatge. Per això, hem creat una funció que per a cada parell de l'etiqueta predita (A) i etiqueta real (B) calcula: $\frac{\#(A \cap B)}{\#(A \cup B)}$, obtenint valors entre 0 i 1 segons si les prediccions són bones o dolentes.

Per tal de veure com varia la precisió en funció del criteri i nombre de centroides hem fet els següents gràfics. A causa de ser un algorisme no supervisat, hem utilitzat només el conjunt d'entrenament, fent la mitjana de tots els valors obtinguts per a cada imatge:

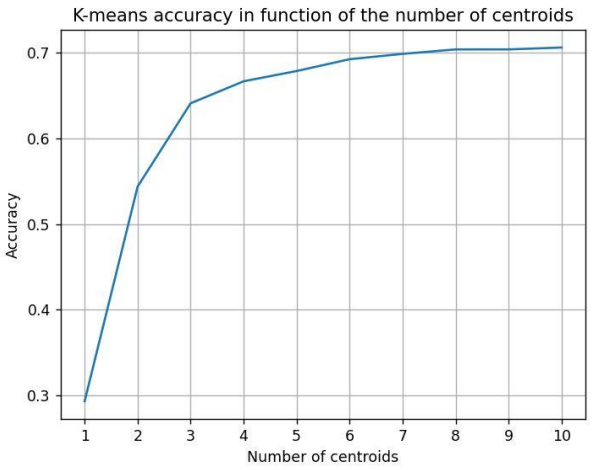
Com podem observar, hem pogut obtenir el millor resultat amb 8 centroides, amb un 44,36% de precisió.

Veient la imatge, aquesta s'estabilitza a partir de la k=5 i millora poc significativament. Realitzar les execucions amb valors de k més grans hem pensat que no tenia gaire sentit per l'increment de temps que suposa amb relació a la millora de



precisió (amb k grans es repeteixen els colors principals).

Una altra prova que hem fet ha estat fer l'execució amb les cropped_images (amb les imatges retallades), que ens ha permès obtenir una millora dels resultats anteriors tal com podem veure a continuació:



Com passava anteriorment, hem obtingut la precisió més gran amb el valor més gran de k. Quan k=8 la precisió ha arribat a ser del 71%. Com veiem aconseguim una millora molt significant respecte amb les altres imatges. El colze potser no és tan evident aquest cop, dependrà del paràmetre que considerem, però hi serà entre k=3 i k=6 en la majoria dels casos.

2.4.4 Accuracy analysis

Podem afirmar que el KNN funciona força bé a l'hora de classificar les imatges considerant k-veïns petits i que el K-means no encerta gaire bé tots els colors de les imatges, tot i que podem aconseguir millorar els resultats treballant amb retalls de les imatges originals.

3. Millores sobre K-means i KNN

En aquesta tercera part del projecte, hem provat diverses millores en el k-means i KNN per avaluar el seu rendiment. En el K-means, hem provat diferents nombres de clústers i diferents mètodes de distàncies a part de la distància intra-class que ja vam implementar.

3.1 Millora a la inicialització de K-means

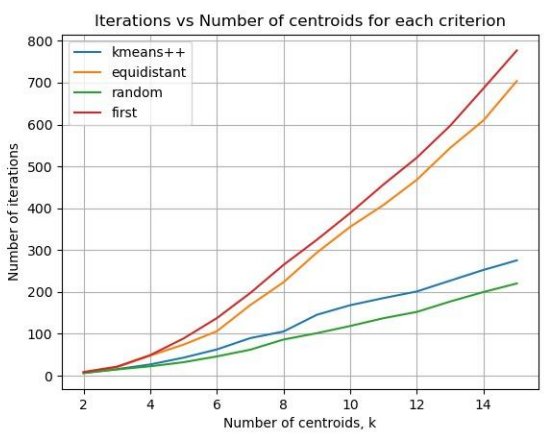
A banda de la inicialització aleatòria i la inicialització amb els primers k element diferents, hem implementat dues més:

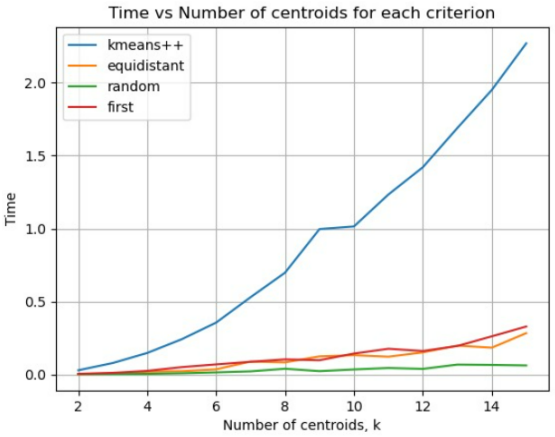
- **K-means++**:

Aquesta és la que hem trobat com la inicialització del K-means més popular. Es basa a triar els centroides de forma pseudoaleatòria, assignant a cada punt una probabilitat per tal de triar-los de manera separada.
Per tal de fer-ho calculem un vector de distàncies, format per les distàncies de cada punt al seu centroide més proper (fent així que els punts agafats com a centroide tinguin distància 0).
El vector de probabilitats és doncs aquest vector entre la suma total de distàncies. Amb aquest vector i la funció de numpy random.choice triem cada cop el proper centroide.
En cas de triar un centroide amb un color de píxel igual a un altre, simplement farem una nova iteració fins a trobar un vàlid.
Pel plantejament de l'algorisme, hem decidit ficar-lo com a mètode predeterminat d'inicialització.

- **Inicialització equidistant**:

Aquesta altra manera es basa a mirar de separar el màxim possible els centroides entre sí. Nosaltres, fins i tot l'hem retocat per a tenir uns centroides separats dels extrems de la imatge, que generalment seran blancs o colors del fons de la imatge.
Per tal de fer-la hem definit un pas (nombre de punts entre nombre de centroides), que multipliquem per la iteració per tal d'accedir a cada punt. En cas de ser únic (no tenir cap igual amb el mateix valor de RGB) el guardem, sinó, busquem un proper restant 1 fins a aconseguir-lo. Per això hem deixat un marge entre els cada centroide i el mateix extrem de la imatge. Trobar aquesta unicitat ho fem amb la funció de Python *any*.
Així i tot, aquesta inicialització no és gaire útil, ja que els colors més diferents o rellevants no tenen res a veure amb la posició en què surten a la imatge, per tant, obtenir punts equiparats pot segons les dades ser-nos més o menys d'utilitat.





En les gràfiques A i B, es veu la comparativa d'inicialització de centroides pel que fa el nombre d'iteracions i el temps respectivament, depenent del nombre de classes K. Aquestes comparatives s'han fet amb un recull ampli d'imatges. A més dels dos criteris explicats anteriorment, s'ha aplicat el *random*, que inicialitza centroides aleatòriament tenint en compte valors des de 0 fins a 255, i el *first* que simplement agafa els primers k punts com a centroides.

Veiem com pel que fa el nombre d'iteracions d'assignació de nous centroides, la inicialització *first* i l'*equidistant* triguen molt més en convergir que les que s'inicialitzen amb una part aleatòria o íntegrament, que són *kmeans++* i *random*. També és visible que l'assignació del nou centroide depenent del vector probabilitats no té gaire sentit en aquesta pràctica, ja que els centroides agafats tendeixen a estar a una distància mitjana respecte els altres. En aquest projecte, s'ha de tenir en compte que hi ha molts punts que contenen colors blancs i fons d'imatge, així que una inicialització *random* seria inclús millor que la *kmeans++*.

Pel que fa el temps, és clarament visible que la funció fit amb inicialització *kmeans++* triga molt més que les altres. Això es deu a que el procés d'inicialització és molt costós. A més, aquest criteri no és gaire útil ja que en general, no es necessiten moltíssimes iteracions per a que els centroides convergeixin. D'aquesta manera, no ens surt a compte sacrificar temps en la inicialització, ja que el procés de convergència no serà, en general, gaire costós. Respecte els altres criteris, veiem que el *random* també és el més òptim pel que fa al temps, i el *first* i *equidistant* són anàlegs.

Per tant, concloem en que la millor inicialització possible seria la *random*, i les dues implementacions fetes no tenen un millor impacte en aquest cas.

En casos on tinguéssim els colors distribuïts uniformement o treballéssim amb una matriu molt més gran de punts o coordenades, *equidistant* i *kmeans++* podrien ser determinants.

3.2 Millores a les heuristiques del Find Best K

A més de la distància intraclass ja definida en la primera pràctica, s'han definit dues heurístiques més, la distància Inter-Class i l'heurística del Coeficient de Fisher, quocient entre la Intra-Class i la Inter-Class.

3.2.1 Distància Inter-Class

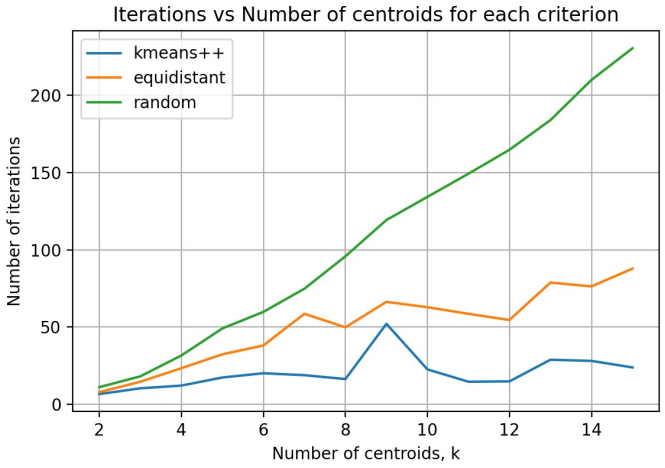
La distància Inter-Class és el sumatori de distàncies entre els diferents centroides. L'objectiu per a obtenir un nombre de classes K òptim serà maximitzar aquesta distància. La funció creada, bàsicament, per cada centroide calcula la distància a la resta. Totes aquestes distàncies es desen en una variable, que serà retornada.

3.2.2 Coeficient de Fisher

Pel que fa al coeficient de Fisher, la funció bàsicament efectua una divisió entre la distància Intra-Class i la Inter-Class, que seran cridades com a funció dins la funció de Fisher.

3.3 Millores al Find Best K

Dins la funció *find_best_k(),* s'ha contemplat les 3 heurístiques possibles. En cas que al diccionari options tinguem *options['fitting']=='WCD'*, computarem la WCD i calcularem el decreixement, i ens quedarem amb la K que l'estabilitzi. Per a fer-ho, es calcularà el percentatge del quocient del WCDk entre el WCDk-1. Si aquest valor sobrepassa un llindar passat com a paràmetre, ens quedarem amb aquesta k-1 i avortarem la funció.
Pel cas *options['fitting']=='ICD'*, cercarem aquella k òptima, trobada abans de l'estabilització del creixement de la distància. La resta del procés es realitzarà anàlogament, però la diferència serà que la condició que comprova si l'estabilització està dins del percentatge del paràmetre tindrà el signe girat, en treballar en aquest cas amb un creixement. És a dir, es té en compte que el quocient ICDk/ICDk-1 > 1, ja que la funció que relaciona les k amb les distàncies fins a la K ideal és monòtona no negativa.
Pel que fa a *options['fitting']=='Fisher'*, el procediment és molt semblant al del WCD, pel fet que la distància del coeficient de Fisher es va minimitzant. L'única diferència és que aquesta opció crida a la funció Fisher, quocient entre les dues heurístiques anteriors.



Hem representat una gràfica que relaciona el nombre de centroides (k) i el nombre d'iteracions amb les diferents inicialitzacions que hem definit.
D'aquesta gràfica podem veure que la millor inicialització es la K-means ++, ja que fa menys iteracions que les altres, sent la inicialització random la pitjor en quant a nombre d'iteracions.

3.4 Millores a Features per KNN

Per tal de millorar els resultats obtinguts en la classificació per K-NN (K-nearest neighbours), vam provar de modificar els espais de característiques. En concret, hem considerat dues possibilitats diferents. La primera que consisteix a calcular el valor mitjà dels píxels, i la segona, calculant la variància.
Per saber si aquestes dues millores de veritat milloren el nostre KNN accuracy, hem executat el get shape accuracy amb les millores i sense i les hem comparat i el mateix amb el cross validation que hem implementat.

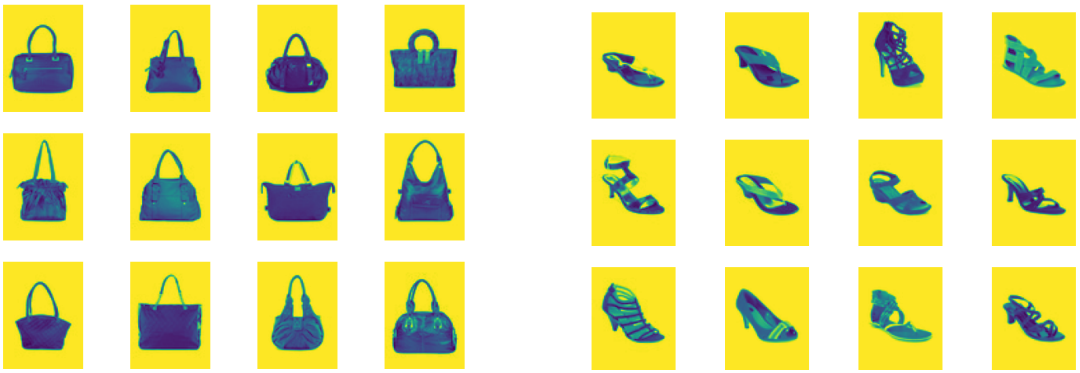
3.4.1 Cross Validation Accuracy

Per altra banda, hem implementat el cross validation accuracy per a veure l'accuracy del knn.
Aquesta tècnica consisteix a dividir les dades en k subconjunts (de la mateixa mida) i realitzar k iteracions on se selecciona a cada iteració un subconjunt diferent com a conjunt de proves, i la resta de subconjunts es fan servir per a l'entrenament del nostre algorisme.
Això ens permet obtenir k estimacions diferents de la precisió del model i calcular la seva mitjana i desviació estàndard, fet que ens ajuda a proporcionar una millor estimació del rendiment del model en dades noves i a reduir la variància en la seva avaluació.

Per tant, calcular la precisió de la validació creuada, ens proporciona una millor comprensió de la capacitat de generalització del model, ja que és menys susceptible a la variabilitat de la divisió de les dades.

3.4.2 Valor mitjà dels pixels

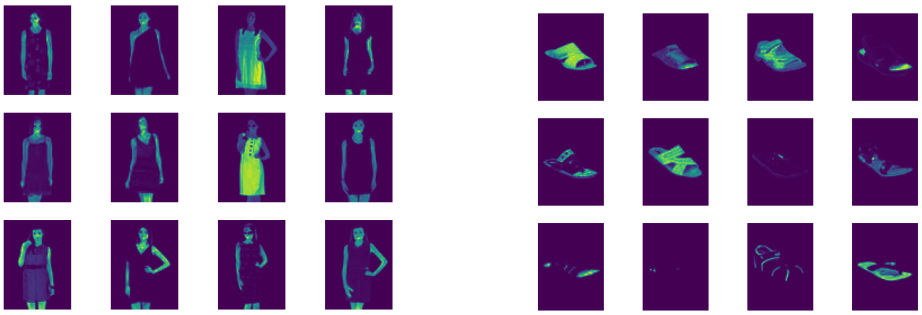
El primer estudi que vam fer va ser utilitzar com a característica el valor mitjà dels píxels de cada imatge. Per fer això, vam calcular la mitjana dels píxels de cada imatge del conjunt de train i del conjunt de test, obtenint així dues noves matrius de característiques. Un cop calculades, vam aplicar el classificador per k-NN usant aquestes noves matrius.



Hem provat amb dos exemples, en un hem buscat bosses de mà i en un altre tacons.
Com podem veure a les imatges, el fons blanc s'ha tornat groc i les siluetes de color fosc.
En aquest cas l'accuracy dels colors no sortiria bé, però és molt probable que l'accuracy de les shapes augmenti pel fet que el contrast del fons i de les shapes és més gran.

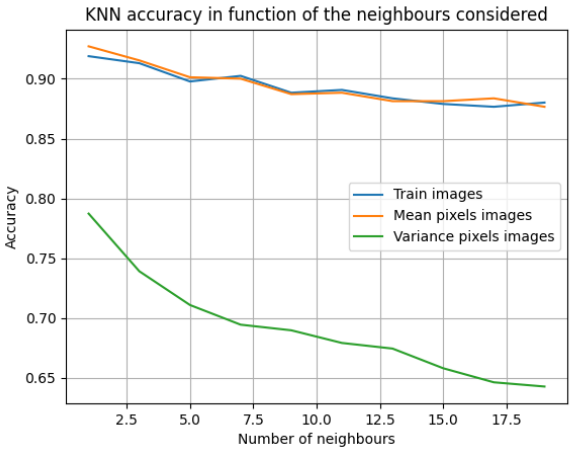
3.4.3 Variància dels pixels

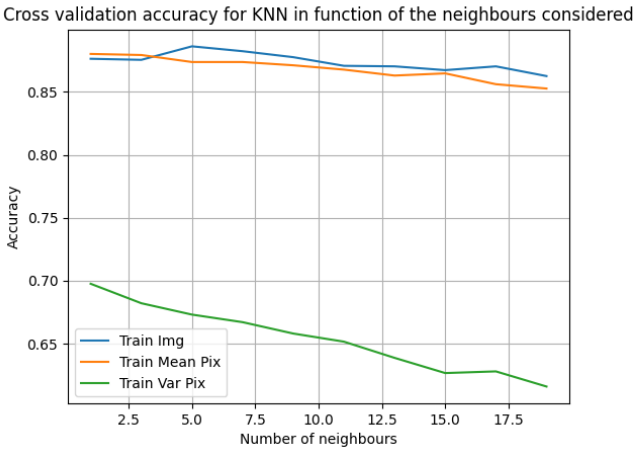
La segona millora que vam fer, va ser utilitzar com a característica la variància dels píxels de cada imatge.
De manera similar al punt anterior, vam calcular la variància dels píxels de cada imatge del conjunt de train i del conjunt de test, obtenint així dues noves matrius de característiques.
Un cop calculades, vam aplicar el classificador per k-NN usant aquestes noves matrius de característiques.



Per visualitzar com serien les imatges, hem provat amb dos exemples (vestits i sandàlies)
Tal com podem veure, podem distingir molt clarament entre colors foscos i colors clars, ja que aquests últims estan molt més il·luminats. Tot i això, no creiem que sigui una bona millora per distingir les formes de les peces de roba.

3.4.4 Gràfics i experiments





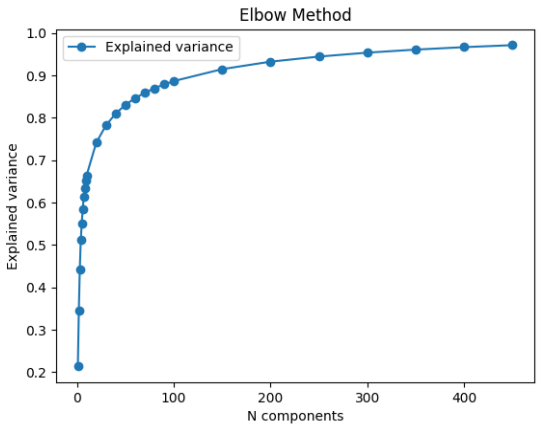
Tal com podem veure, agafant la variància dels píxels de les imatges no millora res la nostra precisió en cap dels dos estudis, per tant, aquesta implementació no es pot considerar una millora per al nostre KNN.
Per altra banda, al comparar l'ús de les imatges originals amb les transformades fent la mitjana dels píxels, s'observa que no hi ha diferències significatives en les precisions obtingudes en els dos estudis, fet que es corrobora amb l'estudi de la desviació típica obtinguda amb el "cross validation", ja que no hi ha evidències que digui que un estudi sigui millor que un altre en aquest aspecte.

Com no podem dir amb l'anàlisi de les precisions quin estudi és millor, hem d'analitzar l'eficiència computacional en la manera en la qual tractem les "Train Images".
Per defecte, podem llegir el conjunt d'imatges en escala de grisos i passar-les directament al KNN, ja que no necessitem el color de la roba per poder determinar la seva forma,
Per tal d'aconseguir les mitjanes i variàncies cal primer tractar les imatges en RGB per convertir els píxels segons el criteri que decidim.
Com estem veient que fent aquestes transformacions no millora la precisió i alhora implica un major cost computacional, hem decidit no contemplar-les.

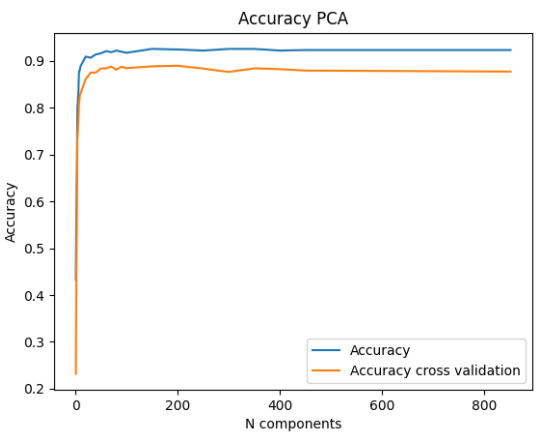
3.4.5 PCA

El PCA (Principal Component Analysis) és un algoritme de reducció de dimensionalitat que s'utilitza per analitzar i interpretar conjunts de dades complexes.
L'objectiu principal del PCA és trobar les components principals d'un conjunt de dades, que són combinacions lineals de les variables originals. Aquestes components principals són vectors propis que capturen la major variància de les dades. Mitjançant l'extracció d'aquestes components principals, podem reduir la dimensionalitat del conjunt de dades, mantenint al mateix temps la major part de la informació original.

Hem fet una anàlisi per trobar quin era el millor nombre de components a considerar tenint en compte que sempre fèiem servir la millor k = 2.
Com podem veure en el gràfic si augmentem la N obtenim més informació de la matriu original. Per tant, fent l'elbow method podem aconseguir la N que agafa la millor informació sense ser massa gran.



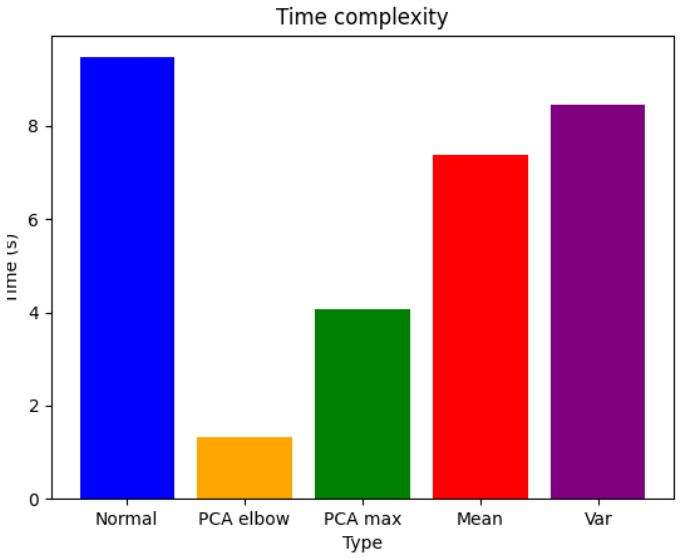
Però, a nosaltres el que més ens interessa és que aquesta N tingui associada una precisió alta. Per això, hem fet aquest segon gràfic on podem veure fent l'elbow method que per a N = 20 assolim una precisió del 0.9095 amb una variància explicada del 74,11%.



Si volguéssim aconseguir la millor precisió possible llavors tindríem N = 350 amb una accuracy de 0.926 i una variància explicada del 96%.

Podem veure doncs que la variància explicada és important, ja que si tenim menys de 74% aconseguirem un accuracy molt baix. D'altra banda, tampoc cal que sigui gaire gran perquè aquest accuracy s'estabilitzarà.

Per acabar, comparem el temps d'execució de fer cadascun dels diferents mètodes per poder comparar la complexitat en temps de tots aquests, amb la N=20 idònia en groc i amb la N=350 amb la que podem obtenir una molt alta precisió (com en el conjunt "normal" o fins i tot millor):



Com podem observar, si no fem res trigarem més temps que aplicant qualsevol transformació. A continuació, veiem que fer tant el PCA amb l'elbow method com amb la N=350 es redueix el temps d'execució, en el cas de l'elbow molt dràsticament (similar a ⅛ del temps original).
D'altra banda, podem veure que en relació amb l'apartat anterior, aplicant les transformacions de mean i var no aconseguim tampoc una millora significativa (tot i que totes dues són millors en temps).
Per tant, podem concloure que el millor mètode és fer l'algoritme PCA i aplicar l'elbow method, el qual ens permetrà obtenir una gran reducció de temps fins i tot obtenint una precisió molt alta.

4. Conclusió Global

Aquesta pràctica ha estat un bon aprenentatge per a tots quatre. Tot i que inicialment només s'havia de programar els algorismes i no contemplaven les aplicacions, hem acabat podent explorar les possibilitats del nostre codi i avaluant-lo amb imatges i casos reals. Considerem que el desenvolupament de la tercera part de la pràctica ha estat clau. Ens ha permès entendre més profundament el codi programat, explorar noves aplicabilitats i raonar maneres de millorar-lo, sense seguir un guió preestablert. Tots quatre creiem que aquest projecte ha estat una petita simulació d'una situació que es pot donar en el món laboral, atès que partint d'uns models d'algorismes ja creats i pautats, hem hagut de crear una de les moltes possibles implementacions.

Creiem que hem adquirit nous coneixements relacionats amb aquests dos algorismes, entre ells a crear populars implementacions de centroides com el K-means++ o bé a utilitzar la coneguda llibreria numpy per tal de fer tota mena d'operacions. Ens ha agradat poder veure de manera pràctica la importància d'aplicar un PCA o bé a fer una anàlisi quantitatiu o qualitatiu dels algorismes implementats.

En conclusió, la llibertat de codi i de creació d'aquesta pràctica ens ha permès adquirir noves nocions i maneres òptimes de programar, trets clau per a esdevenir un programador complet.