Projecte 2: Etiquetatge

Intel·ligència Artificial

Daniel Rosa Díaz 1604158
Adrián Martínez García 1601959
Sergi Diaz Lopez 1599349

Index

Introducció

En les parts anteriors de la pràctica hem implementat els models KNN i KMeans per a poder fer l'etiquetatge de color i forma de peces de roba. En aquesta última part implementarem funcions per a cercar peces de roba per a verificar la qualitat dels resultats i analitzarem el funcionament dels algorismes mitjançant estadístiques.
Per últim, a partir de les conclusions extretes dels anàlisis, modificarem la implementació dels algorismes per a millorar el seu funcionament.

Métodes d'anàlisi

Anàlisi qualitatiu

Hem implementat les següents funcions per a verificar el funcionament dels models KNN i KMeans: `retrieval_by_color`, `retrieval_by_shape` i `retrieval_combined`. La entrada principal de les funcions serà la "query", una llista de strings amb els colors o formes que volem cercar. Retornará les imatges que continguin peces de roba coincidents amb la cerca. Si analitzem els resultats, veurem que retorna algunes imatges incorrectes, que no son del color/forma que hem especificat. Les imatges marcades de color verd son les que coincideixen amb el ground truth.

En la versió sense millores hi ha alguns colors per als quals funciona bastant bé, com el blau o el negre:

Veiem que en alguns, apareixen moltes peces de roba d'altres colors, ja que l'algorisme detecta el color de la pell o altres peces de roba del model com a vermell o taronja.

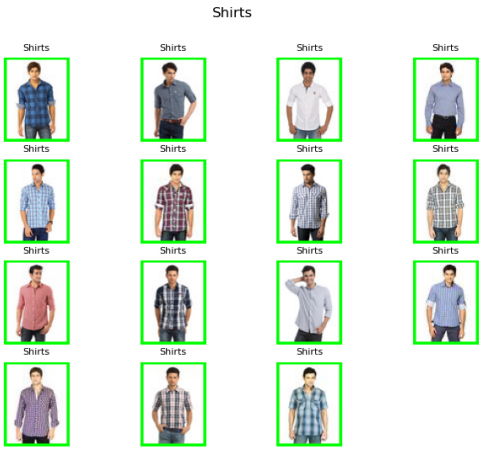
I en altres casos, com el gris i blanc, ens detecta el color del fons de la imatge.

Per a solucionar això podem donar com a input del KMeans les imatges retallades per a obtenir els colors del centre de la imatge. Això ja esta implementat en el extended ground truth i tenim una funció que retalla imatges.
Si executem la cerca per color utilitzant les imatges retallades i el extended ground truth, obtenim el següent:

Obtenim una millora en els colors que abans eren conflictius. També, al executar Kmeans amb les imatges retallades veiem un temps d'execució reduït.
Aplicarem les millores explicades més endavant per a millorar encara més aquests resultats.

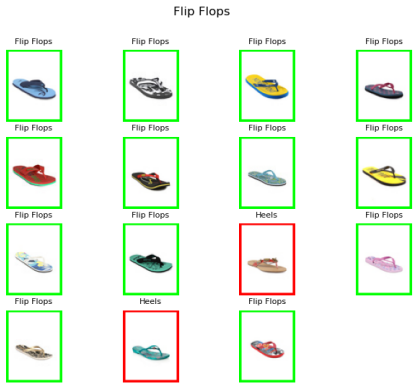
En el etiquetatge de formes obtenim un bon resultat, amb alguns errors puntuals.


Heels
Heels
Heels
Heels
Heels
Heels
Heels
Heels
Heels
Heels
Heels
Heels
Heels
Sandals
Heels


Shirts
Shirts
Shirts
Shirts
Shirts
Shirts
Shirts
Shirts
Shirts
Shirts
Shirts
Shirts
Shirts
Shirts
Shirts


Flip Flops
Flip Flops
Sandals
Heels
Heels
Sandals
Sandals
Sandals
Sandals
Sandals
Sandals
Sandals
Sandals
Sandals
Sandals


Dresses
Shirts
Dresses
Dresses
Dresses
Dresses
Dresses
Dresses
Dresses
Shirts
Dresses
Dresses
Dresses
Dresses
Socks


Flip Flops
Flip Flops
Flip Flops
Flip Flops
Flip Flops
Flip Flops
Flip Flops
Flip Flops
Flip Flops
Flip Flops
Heels
Flip Flops
Flip Flops
Heels
Flip Flops


Jeans
Jeans
Shorts
Jeans
Jeans
Jeans
Jeans
Jeans
Jeans
Jeans
Jeans
Jeans
Jeans
Jeans
Jeans

Finalment, hem implementat una funció per a cercar color i forma a la vegada, fent servir els dos algorismes. Un exemple del resultat d'aquesta cerca (green black dresses):

Anàlisi quantitatiu

Hem implementat les següents funcions per a verificar el funcionament dels models KNN i KMeans a la part qualitativa: `get_shape_accuracy`, `Kmean_statics` i `get_color_accuracy`. Per aquestes funcions, pasem un parámetre K en cas de KNN, i a més un parametre MaxK en cas del Kmean_statics, que ens indicará el interval de valors de K que analitzarem.

Hem implementat la funció get_shape_accuracy. Aquesta funció pren com a paràmetre un valor de K i retorna el percentatge de imatges que s'ha classificat correctament la forma.

Les següents imatges mostren el percentatge d'encerts per a valors de K de 2 i 3

```
K =  2  Percentatge =  92.01 %
```

```
K =  3  Percentatge =  90.95 %
```

A partir d'aquests resultats podem observar que per a diferents valors de K no varía gaire el percentatge.

Per la part del KMeans, hem implementat una funció semblant però per al color, get_color_accuracy. En aquesta funció, com que sempre ens dona K colors per imatge, hem donat per correcte la classificació si tots els colors que ens ha donat el Kmeans es troben al ground_truth.

Les següents imatges mostren el percentatge d'encerts per a K amb valor 2,3 i 4.

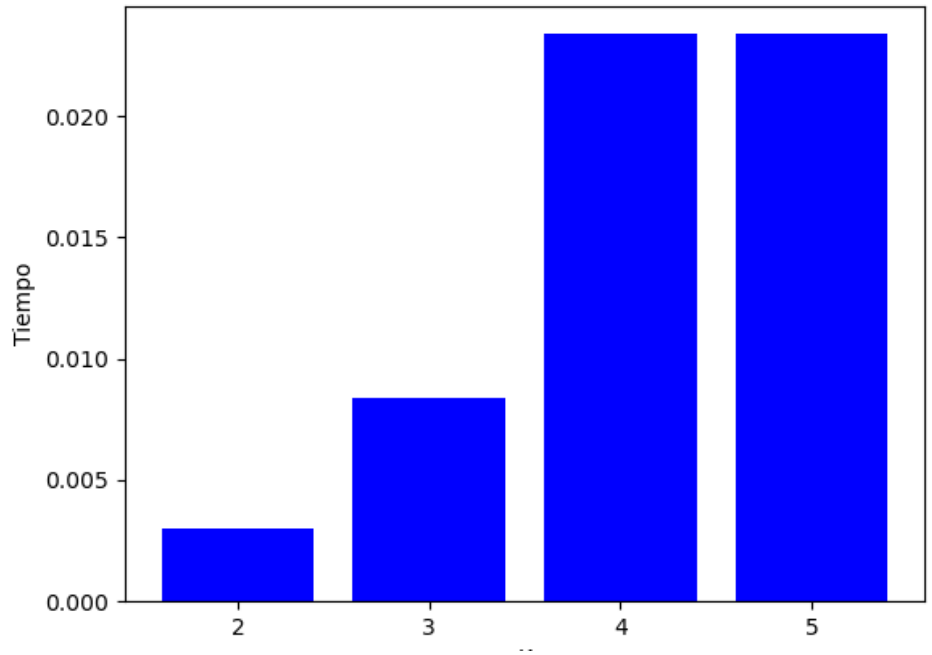
```
K =  2  Percentatge =  44.18 %
```

```
K =  3  Percentatge =  59.22 %
```

```
K =  4  Percentatge =  69.57 %
```

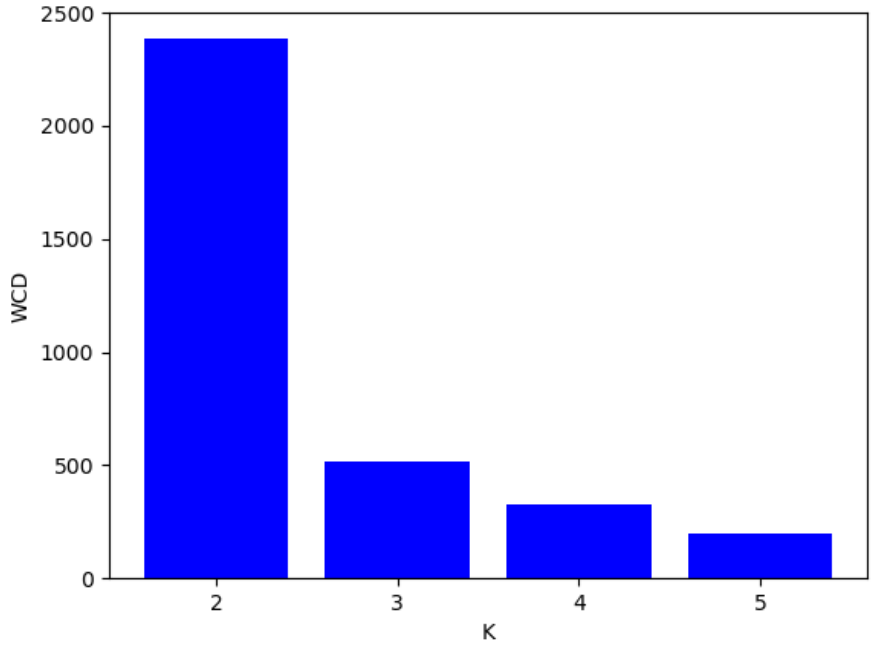
A partir d'aquests resultats, podem deduir que, al augmentar la K, augmenta el percentatge d'encerts.

A més, hem implementat la funció Kmean_statics, que ens serveix per poder comparar el temps y el WCD del nostre KMeans per a un rang de valors de K fins a KMax i mostrar-ho en una gràfica.
A la gràfica següent podem veure el temps que triga en iterar el Kmeans per a K entre 2 i 5.



Si veiem aquesta gràfica, observem que, quan més gran es el valor que assignem a K, més temps triga en iterar, degut a que augmenta la complexitat.

En la gràfica següent podem veure els valors del WCD per a cada valor del rang de K:



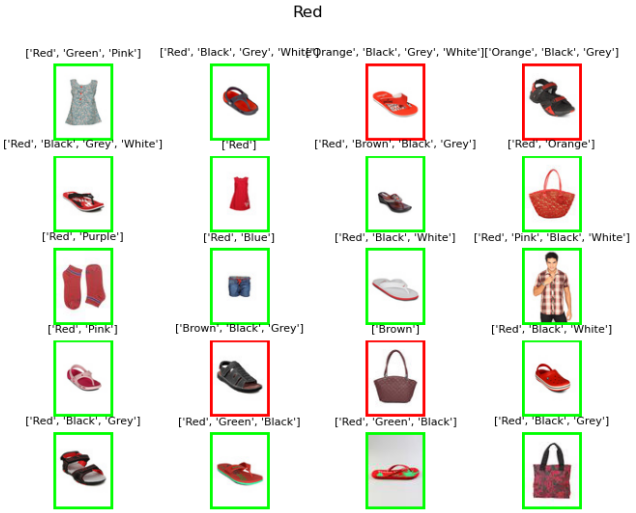
A partir d'aquesta gràfica podem veure que, quan més gran es el valor de K, més petit es el valor del WCD, es a dir, millora.

Millores

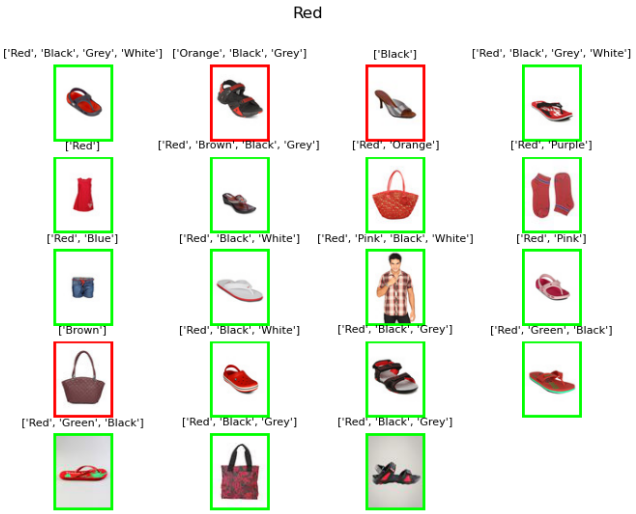
KMeans

La primera millora que volem fer en el kmeans es probar amb diferents heurístiques per la funció Find_BestK. Está implementada la funció withinClassDistance, que calcula la distància Intra-class. En aquesta part provarem d'implementar altres heurístiques, com la distancia Inter-class i el discriminant de Fisher i analitzarem si hi ha alguna millora.

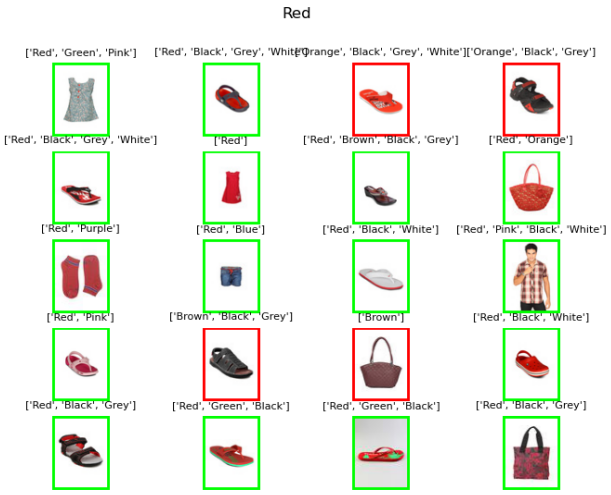
Cerca peces de color vermell amb WCD:



Cerca peces de color vermell amb ICD:



Cerca peces de color vermell amb discriminant de Fisher:



Al haver provat les diferents heurístiques ens hem donat compte que la millor es la ICD, que millora una pica respecte a la heurística per defecte que tenim que era la WCD

Una altre millora que volem fer en el KMeans es probar a cambiar el llindar o altres equacions en la funció Find_BestK. Jugarem provant si algun d'aquest cambies té alguna millora o es pitjor que el que ja tenim.

Al modificar el llindar de la funció Find_BestK podem veure que el valor per defecte del llindar, que es 0.2 (20%), fa que la millor k sigui 4. La millor k sempre será 4 desde el valor 0 a 0.36 del llindar, 3 si el llindar es 0.37 fins a 0.78 y 2 quan va de 0.79 fins a 1. Després de veure els diferents valor que podem trobar modificant el llindar tenim que veure quina de les tres opcions es la més eficient.

Amb la funció get_color_accuracy que hem fet hem pogut veure quina K d'aquestes és la millor, i per tant quin llindar utilitzarem.

K = 4	K = 3	K = 2
Percentatge = 69.57 %	Percentatge = 59.34 %	Percentatge = 44.07 %

Després de veure el percentatges podem veure que la opció que ya teniem per defecte és la millor de totes, per tant hem decidit no tocar el llindar predeterminat.

Per últim, hem implementat dos mètodes d'inicialització dels centroides més: "random" i "custom", a més de l'opció "first". La opció "random" selecciona els centroides de manera aleatòria, i la opció "custom" en la qual inicialitzem el primer element aleatòriament i selecciona la resta segons la distància màxima als centroides anteriors.

KNN

Respecte al KNN la millora que volem fer és provar diferents distàncies o modificar l'espai de característiques per a veure si podem obtenir una millora respecte al codi que hem fet.

Hem decidit fer la prova amb la distància euclidiana, que es la que tenim per defecte, la Manhattan y la Minkowski. Hem provat les distàncies amb la funció Retrieval_by_shape que havíem fet anteriorment. Veient els resultats hem vist que la distancia euclidiana y la Minkowski fan 14 errors de 90 posibles, y la distancia Manhattan fa només 11 errors, per tant hem decidit que la distancia per defecte será la distancia Manhattan

Conclusions

Amb aquesta part de la pràctica hem pogut fer un analisis del nostre codi y nosaltres mateixos fer funcions per a provar diferents aspectes del codi y poder fer proves veient dates que tinguin sentit. Gràcies aquestes proves hem pogut comprovar si les possibles millores que havíem pensat eren efectives o no veient els diferents resultats que donen amb les funcions que hem implementat. Per tant hem pogut afegir millores al codi que estan comprobades que son efectives.