Kristian Stoyanov Belkin, Paul Daniel Pop i Raúl Jiménez Ayza

NIU: 1667799, 1607129 i 1688916

**Proyecto 2 Parte 3 ETIQUETATGE Inteligencia artificial**

Contenido

[Introducción 2](#_Toc166447454)

[Funciones de análisis cualitativo 2](#_Toc166447455)

[Retrieval\_by\_color: 2](#_Toc166447456)

[Retrieval\_by\_shape: 2](#_Toc166447457)

[Retrieval\_combined: 3](#_Toc166447458)

[Funciones de análisis cuantitativo 3](#_Toc166447459)

[Kmean\_statistics: 3](#_Toc166447460)

[Get\_shape\_accuracy: 3](#_Toc166447461)

[Descripción de los experimentos 3](#_Toc166447462)

[Mejoras o cambios k-means y KNN 4](#_Toc166447463)

[Kmeans ++ 4](#_Toc166447464)

[Coeficiente de Fisher 4](#_Toc166447465)

[Find\_BestK: 4](#_Toc166447466)

[Visualización 4](#_Toc166447467)

[Resultats preliminars 4](#_Toc166447468)

# Introducción

En esta práctica tenemos la tarea general de realizar un etiquetaje automático de imágenes de ropa, a través de nuestro código deberíamos poder asignar etiquetas a diferentes tipos de ropa (8) con sus respectivos colores (11).

Para realizar estas tareas utilizamos dos archivos para dos algoritmos llamados KNN (K-nearest neighbors o k vecinos más próximos) y kmeans (k-means clustering o K-medias). A continuación, explicaremos un poco más en detalle ambos algoritmos:

Kmeans es un método de clasificación no supervisada, que utilizamos para generar clústeres a partir de la información que tenemos de los colores, para poder agrupar los pixeles de cada imagen y así determinar los colores predominantes.

KNN a diferencia del Kmeans es un método de clasificación supervisado, que utilizamos para encontrar el tipo de ropa de cada imagen, lo que acaba haciendo es buscar K vecinos más cercanos y se asigna el tipo mirando por la mayoría de los vecinos más cercanos.

# Funciones de análisis cualitativo

## Retrieval\_by\_color:

Retrieval\_by\_color permite buscar una serie de imágenes por los valores de su color.

Texto

Descripción generada automáticamente

Lo que realizamos en esta función es pasar por todos los kmeans tags y enumeramos cada iteración con x. Luego iteramos por todos los elementos y en caso de coincidencia guardamos la imagen en una lista que creamos previa al bucle. Finalmente ejecutamos visualize\_retrieval para ver el resultado.

## Retrieval\_by\_shape:

Retrieval\_by\_shape permite buscar una serie de imágenes por los valores de su forma.

Texto

Descripción generada automáticamente

Como en la previa función creamos una lista para todas las imágenes que coinciden con nuestros criterios de búsqueda. Luego empezamos iterando por todos los elementos de knn\_tags mientras que en i enumeramos cada iteración, posteriormente comprobamos si el tamaño que buscamos esta entre los tags, de ser así guardamos la imagen en la lista previamente creada. Finalmente mostramos los resultados con visualize\_retrieval.

## Retrieval\_combined:

La función retrieval\_combined permite buscar imágenes que los criterios coinciden con la forma y color que hemos intentado buscar.

Texto

Descripción generada automáticamente

/\* Por acabar de hacer, actualmente no funciona WORK IN PROGRESS\*/

# Funciones de análisis cuantitativo

## Kmean\_statistics:

Debería ser una función que recibe la clase kmeans con una serie de imágenes y número de K que representa el máximo que queremos analizar de K=2 hasta K=max. Ejecutará la función fit, calculará el WCD, el número de iteraciones/tiempo etc.

/\* La función está por hacer \*/

## Get\_shape\_accuracy:

Esta función recibe los tags y el ground\_truth y debería devolver el porcentaje de etiquetado correcto. Nos servirá para validar los tags que hemos podido obtener del algoritmo KNN.

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

Lo primero que hacemos es comprobar que los tags sean del mismo tamaño que el ground\_truth, en caso contrario saltamos un error. En la variable total guardamos la longitud de los tags para posteriormente sacar el porcentaje y empezamos a comprobar en correct la cantidad de tags que se predicen correctamente sumando 1 por cada correcto. Finalmente devolvemos el porcentaje que se saca dividiendo los correctos entre el total y multiplicando por 100 (en caso de no tener una lista de tags vacíos).

# Descripción de los experimentos

Para experimentos que se realizarán posteriormente tenemos pensado, en el caso de análisis cualitativo observar el ground-truth de diferentes casos de test/prueba y utilizar las funciones de retrieval anteriores para comprobar visualmente si el algoritmo es correcto. Para el caso del análisis cuantitativo tenemos pensado representar gráficamente cómo las diferentes mejoras que plantearemos después afectan a diferentes parámetros como: tiempo de ejecución, número de centroides, exactitud de la solución, etc.

# Mejoras o cambios k-means y KNN

## Kmeans ++

Los principales problemas del K-means es su inicialización de los centroides. Este requiere que los centroides iniciales se elijan de forma aleatoria o de otras formas muy simples, lo que puede llevar a soluciones subóptimas y muy sensibles a la inicialización de dichos centroides. Además, en conjuntos de datos no uniformes o que representen clases muy diferentes en sus características, K-means estándar puede converger hacia óptimos locales en lugar de llegar a soluciones globales.

K-means++ es una mejora del algoritmo que aborda concretamente estos problemas. La clave del K-means++ es elegir los centroides de una manera más inteligente y efectiva para distribuirlos más uniformemente en el espacio de características.

La idea principal radica en seleccionar los centroides de acuerdo con un criterio basado en la distancia. Para conseguir este resultado se pueden seguir estos pasos:

1. **Selección del primer centroide**
2. **Encontrar candidatos para el próximo centroide**
3. **Selección del siguiente centroide**
4. **Repetir hasta seleccionar todos los centroides**

## Coeficiente de Fisher

La idea es mejorar la clasificación del K-means aplicando el coeficiente de Fisher, el cual utiliza tanto el Intra-class y el Inter-class, de cara a programar solo tendríamos que diseñar la Inter-class ya que en la entrega anterior ya teníamos diseñada la Intra-class, y el coeficiente de Fisher que equivale a la siguiente formula que aplica las dos anteriores:

Imagen que contiene Texto

Descripción generada automáticamente

Esto mejorara la clasificación, ya que proporciona una mejor relación tanto dentro de la classe como fuera de ella de ahí que utilice ambos discriminantes Intra y Inter.

## Find\_BestK:

En la primera parte utilizamos la siguiente formula con un límite del 20% para encontrar la mejor K:

Imagen que contiene Texto

Descripción generada automáticamente

En esta mejora buscaremos fórmulas alternativas o probar mejores límites para intentar conseguir una mejor K.

# Visualización

# Resultats preliminars