Reconocimiento de frutas mediante visión artificial usando *k-means* y *k-nn*

Gino Hernán Avanzini[1]

[1]Facultad de Ingeniería - Universidad Nacional de Cuyo

Febrero 2020, Mendoza - Argentina

Resumen

Se desarrolló un sistema de reconocimiento de frutas mediante el uso de técnicas de procesamiento de imagen e inteligencia artificial. Para ello se utilizó la última versión del *dataset* "Fruit 360" junto con fotografías propias. Se realizó un trabajo de preprocesamiento de las imágenes para eliminar ruido y acentuar características para la posterior extracción de las mismas. Dichas características fueron usadas para el reconocimiento de las frutas con el uso de los algoritmos *k-means* y *k-nn*. Se evalúan y comparan los resultados y se propone la adopción de uno de los dos algoritmos.

1. Introducción

Bajo la directiva de intentar agilizar el proceso de cobro en una caja de supermercado se propone desarrollar un sistema de reconocimiento de frutas por visión artificial mediante un agente prototipo que pueda reconocer bananas, naranjas y limones.

Para eso se diseñó e implementó un agente capaz de realizar tal tarea en el lenguaje de programación *Python* con la ayuda de las librerías *numpy*, *matplotlib* y *scikit-image*. Mediante técnicas y algoritmos de procesamiento de imagenes se lograron extraer características de un dataset de bananas, naranjas y limones para ser utilizadas como datos de entrada para el agente. Se usaron los algoritmos *k-means* y *k-nn* de clustering y de clasificación, respectivamente, para poder definir de qué clase es cada fruta.

En primer lugar se utilizó el entorno de desarrollo *Jupyter* con el kernel *IPython* para entrar en contacto con las funciones y el *workflow* de la librería de procesamiento de imagen scikit-image [5] utilizada en el trabajo. Para eso fue de gran ayuda seguir un *workshop* dictado por los creadores de la librería [6].

Luego se procedió a importar en un *Jupyter notebook* el dataset utilizado para realizar un análisis acerca de los diferentes métodos de extracción de características y se eligieron aquellas que permiten clusterizar de mejor manera las imágenes. Una vez elegidas las características a extraer se realizó en scripts de *Python* la implementación de los algoritmos *k-means* y *k-nn*.

Este proyecto se desarrolló en el marco del Trabajo Final de la cátedra Inteligencia Artificial I en la Facultad de Ingeniería de la Universidad Nacional de Cuyo, a cargo de la Dra. Selva Rivera.

2. Agente

Para la tarea especificada se requiere el desarrollo de un *agente* que realice el procesamiento, la extracción de características y finalmente la clasificación de las imágenes de las frutas. En las siguientes subsección se plantea una descripción formal del mismo.

2.1. Especificación del agente

El agente es racional, es decir, no es solamente algo que actúa, sino que se espera que opere autónomamente, perciba su entorno y siga objetivos. El agente racional actúa buscando el mejor resultado o, cuando hay incertidumbre, el mejor resultado esperado [4]. Es este además un agente que aprende, por la propia naturaleza de los algoritmos empleados.

2.1.1. REAS: Rendimiento, Entorno, Actuadores y Sensores

Para describir el entorno de la tarea recurrimos a la descripción **REAS** [4], la cual especifica la medida de **R**endimiento, el **E**ntorno, los **A**ctuadores y los **S**ensores con los que cuenta el agente.

- Medida de Rendimiento: se toma la cantidad de aciertos del conjunto de test como una medida de qué tan bien clasifica el algoritmo. Si bien las imágenes del conjunto de test tienen etiqueta y por lo tanto se podría pensar que esta es una medida "sintética", dado que no participan en la etapa de training, se puede considerar que esta medida es representativa de la realidad;
- Entorno: este es el área en el que está ubicada la fruta a ser reconocida. Afectan la iluminación y el fondo. Dado que el entorno está restringido a una iluminación fija y fondo blanco, el problema de diseño se simplifica;
- Actuadores: los actuadores serán las formas que tenga el agente en comunicar el resultado de la clasificación. Puede ser un aviso por pantalla, un LED indicador o una señal electrónica para comunicar el resultado. En esta implementación la clasificación final se comunica por consola.
- Sensores: el principal sensor es la cámara con la que se toman las fotos de las frutas.

2.2. Propiedades del entorno de la tarea

Los campos para clasificar el entorno del agente han sido extraídos de Russell y Norvig [4].

- Parcialmente observable: de la fruta a analizar se obtiene una foto, la cual es una proyección bidimensional del cuerpo. Esta proyección no otorga una visión completa de la fruta. Podría suceder que la foto no dé toda la información relevante si el ángulo en el cual fue tomada no es el más adecuado. Por eso decimos que el entorno es parcialmente observable.
- Agente simple: el único agente racional que participa es el programa desarrollado
- Estocástico: el siguiente estado del entorno, es decir, la fruta siguiente a analizar, no está determinada por la fruta que se tomó en el momento actual.
- Episódico: la experiencia del agente se divide en episodios atómicos. Estos son los instantes en que se toma la fotografía y se clasifica la fruta. La decisión actual no afecta las decisiones futuras.
- Estático: el entorno no cambia mientras el agente está deliberando.
- Discreto: tanto el tiempo, las percepciones y las acciones del agente son discretas. Esto es, de una fotografía tomada en instantes episódicos, el agente debe decidir clasificar la fruta entre una cantidad finita de clases.

2.3. Base de datos

2.3.1. Fruits 360

Como fuente de imágenes se planteó en un primer momento realizar una base de datos a partir de fotografías propias pero al ser estas imágenes crudas sin ningún tipo de preprocesamiento la tarea del aprendizaje de los algoritmos y filtros se vería complejizada. Es por esto que se optó por utilizar una base de datos disponible en *Kaggle*, una comunidad de *data science* con múltiples recursos disponibles.

El dataset en cuestión es *Fruits 360* [3] y está publicado con licencias que permiten el uso en este trabajo. Este dataset cuenta con más de 80000 fotos de distintos tipos de frutas con su respectiva etiqueta y además por cada tipo de fruta hay un set de entrenamiento y otro de test. Cada imagen es una foto a color de 100x100 píxeles. En la figura 1 se observa un ejemplo de banana, naranja y limón obtenidos del dataset.

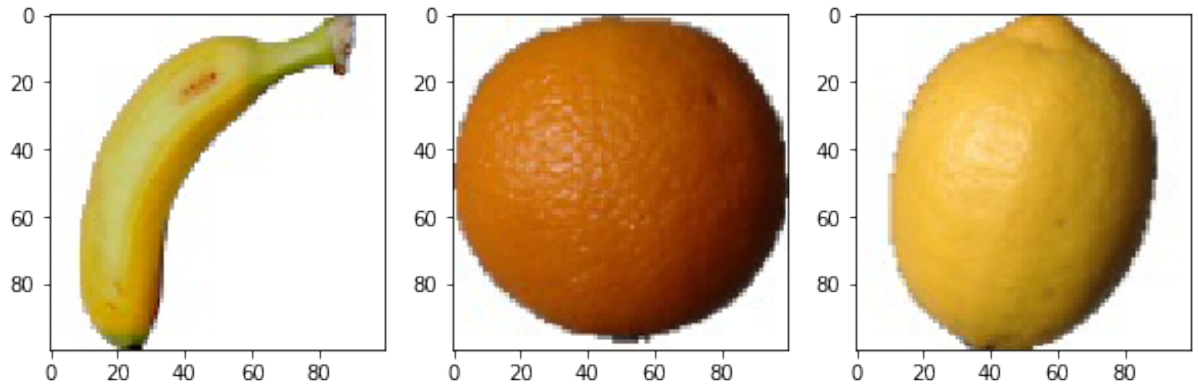

Figura 1: Ejemplos de cada fruta obtenidos de Fruit 360

2.3.2. Dataset propio

Posteriormente y luego de haber elegido las características a elegir se probaron los algoritmos desarrollados con fotografías propias, totalmente distintas a las de Fruits 360.

Se tomaron imágenes de bananas, naranjas y limones de forma casera con la ayuda de un celular Huawei P9. El setup consistió de un reflector LED de 10W de luz blanca fría que iluminaba a la fruta en cuestión sobre un fondo blanco. Se utilizó esta fuente de luz porque las luces comunmente encontradas en las casas no proveen suficiente intensidad lumínica además de que poseen demasiada dispersión y generalmente son de un color blanco cálido. El flash del celular tenía el problema de ser demasiado puntual por lo que generaba zonas mal iluminadas en las esquinas de la imagen. Además se tomaron las fotografías ajustando el ISO y aumentando el tiempo de obturación para que más luz ingrese al sensor de la cámara y de esta forma obtener un fondo con un color más cercano al blanco puro. Esto es para poder realizar un *thresholding* más óptimo y de esta forma separar mejor el *background* del *foreground*, la fruta. La resolución de las fotos es de 3000x3000 = 9MP. Por último, es valioso aclarar que las fotos fueron tomadas con un sensor monocromático por lo que no fue necesaria la transformación de RGB a escala de grises. Sin embargo sí se tuvo que transformar la escala de valores de los pixeles de la imagen: de números enteros de 0 a 255 (8 bits) a números en punto flotante de 0 a 1.

La cantidad de imágenes en el dataset propio es mucho menor ya que estas se usaron únicamente para testear los algoritmos. Además debido a que la resolución de las fotos es significativamente más grande que la de las frutas de Fruits 360 el tiempo de ejecución aumentó notablemente por lo que se decidió mantener un número bajo de imágenes. En un trabajo futuro se podrían explorar algoritmos para hacer *downscaling* y evaluar su efecto en el rendimiento.

2.4. Filtrado y acondicionamiento de imagen

Como uno de los requerimientos es analizar las imágenes en escala de grises, se cargan todas las imágenes del dataset con la ayuda de un método de *skimage* y se obtienen las imágenes como se observa en la figura 2.

2.4.1. Filtro gaussiano

Luego se hicieron experimentaciones tratando las imágenes en *raw*, es decir, sin ningún tipo de filtrado. Esto funcionó bien para el dataset *Fruits 360* ya que este viene preprocesado y el *background* ya está bastante bien separado del objeto. Cuando se comenzó el análisis con el dataset propio comenzaron los inconvenientes. Estas imágenes, aunque *a priori* no parecían tener ningún problema, tenían cierta cantidad de ruido que afectaba la performance de los algoritmos de *thresholding* que se explican luego en la sección 2.4.2 y consecuentemente la performance de los algoritmos *k-means* y *k-nn* se veía disminuida.

Es por esto que se decidió aplicar un filtro para mitigar el ruido que poseen las fotos. Se usó un filtro gaussiano con una desviación estándar igual al 0.5 % de la cantidad horizontal de pixeles de la imagen. Este valor es paramétrico ya que los distintos datasets tienen imágenes de diferentes resoluciones. Así se obtiene que para las imágenes de *Fruits 360* $\sigma = 0{,}5$ y para el dataset propio, $\sigma = 15$.

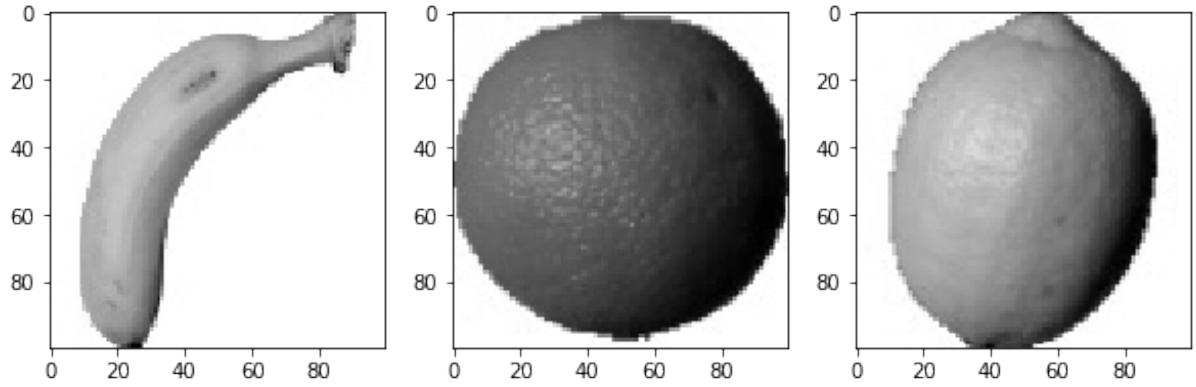

Figura 2: Ejemplos cargados en escala de grises

2.4.2. Thresholding

En primera instancia se intentó hacer la clasificación teniendo realizando un análisis textural de las imágenes en escala de grises. Esto significa que cada pixel de la imagen tendrá una ponderación distinta y esta será un número de punto flotante de 0 a 1. Viéndolo desde un punto de vista mecánico, cada pixel tiene una masa distinta y hay regiones de la imagen que tienen distintas densidades. Sin embargo este enfoque no resultó en una buena clusterización de los datos, lo que implicó un muy mal desempeño de los algoritmos.

Es por esto último que se optó por dar un paso más y binarizar la imagen. De esta forma todos los pixeles que son parte del objeto a analizar tienen un valor de 1 (blanco puro) y el fondo o *background* un valor de 0 (negro puro).

Para realizar este proceso existen varios algoritmos que realizan el thresholding de forma automática y adaptativa. La librería *scikit-image* provee implementaciones para varios, entre ellos *Isodata*, *Otsu*, *Mean*, *Triange* y más. Además cuenta con una función en la que se pueden evaluar rápidamente los resultados para distintos algoritmos. Luego de evaluar la performance de estos se eligió como algoritmo el método *Triangle*. Aunque este no presentó el mejor rendimiento en el dataset propio, sí tiene un comportamiento muy aceptable y consistente entre todas las frutas del dataset *Fruit 360*. Otras opciones, como *Otsu* o *Minimum*, que tenían una performance excelente con bananas, fallaban en gran medida con limones. Como ya se mencionó, antes de aplicar el *threshold* a la imagen se le aplicó un filtro gaussiano. En la figura 3 se puede apreciar este método.

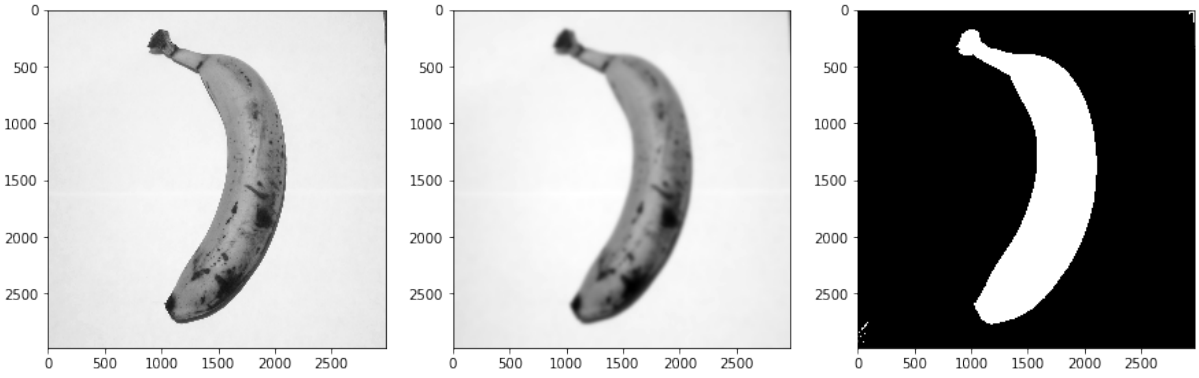

Figura 3: Banana ejemplo del dataset propio. A la izquierda, la foto original. En el centro, luego de pasar por un filtro gaussiano con $\sigma = 30$. A la derecha el algoritmo *Triangle threshold* aplicado a la figura del centro.

2.5. Extracción de características

Luego del procesamiento de las imágenes detallado anteriormente, se debe realizar la extracción de características. Debemos encontrar un valor o un conjunto de valores que sean representativos de cada

imagen para poder usarlos como datos de entrada a los algoritmos implementados.

Muchas técnicas distintas existen para poder extraer características de una imagen. Ya que en la sección 2.4 se obtuvo una imagen binaria de cada fruta, tenemos información acerca de la *forma* de cada fruta. Es por esto que en este trabajo se decidió aplicar un enfoque geométrico para obtener las características.

2.5.1. Relación de Inercia

Dado que las tres frutas en cuestión tienen formas bastante diferentes entre sí, se supuso que sus tres momentos de área deben ser lo suficientemente distintos como para distinguirlas entre sí. Primero se pensó en los momentos de la imagen respecto a los ejes que crean la fila cero y la columna cero pero estos valores no son invariantes respecto a la translación, por lo que la ubicación de la fruta en la foto impactaría mucho. La solución es encontrar los momentos centrales o centroidales en los dos ejes, horizontal y vertical. Estos son invariantes respecto a la translación pero no respecto a la rotación. Además estos dos parámetros son función de la cantidad de pixeles de la imagen, lo que supone un gran problema. Esto último se soluciona calculando los momentos centroidales normalizados, pero aún así no se obtiene una característica que sea invariante a la rotación, translación y la escala de la fruta.

Finalmente se calculó el tensor de inercia de la imagen. Los autovalores de esta matriz indican las magnitudes de los momentos principales de inercia y los correspondientes autovectores indican la dirección de los ejes principales de inercia. Aunque los autovalores de la matriz son función de la escala de la imagen, no lo es el cociente entre ellos. Definimos como relación de inercia al cociente entre el mayor momento principal y el menor. De esta forma, la relación de inercia da una estimación del alargamiento de la fruta analizada. Para las naranjas, dado que su forma asemeja una esfera, se espera una relación de inercia cercana a la unidad. Para los limones, siempre que las fotografías se tomen en un plano paralelo al eje del limón, se esperan valores superiores a la unidad. Por último, para las bananas, se espera que la relación de inercia sea la máxima de las tres.

En la figura 4 se observa un histograma de las relaciones de inercia para las distintas frutas. Si bien se ve una buena diferenciación entre naranjas y limones, las bananas presentan mucha dispersión en sus valores y se observa cierta interferencia con los valores de los limones.

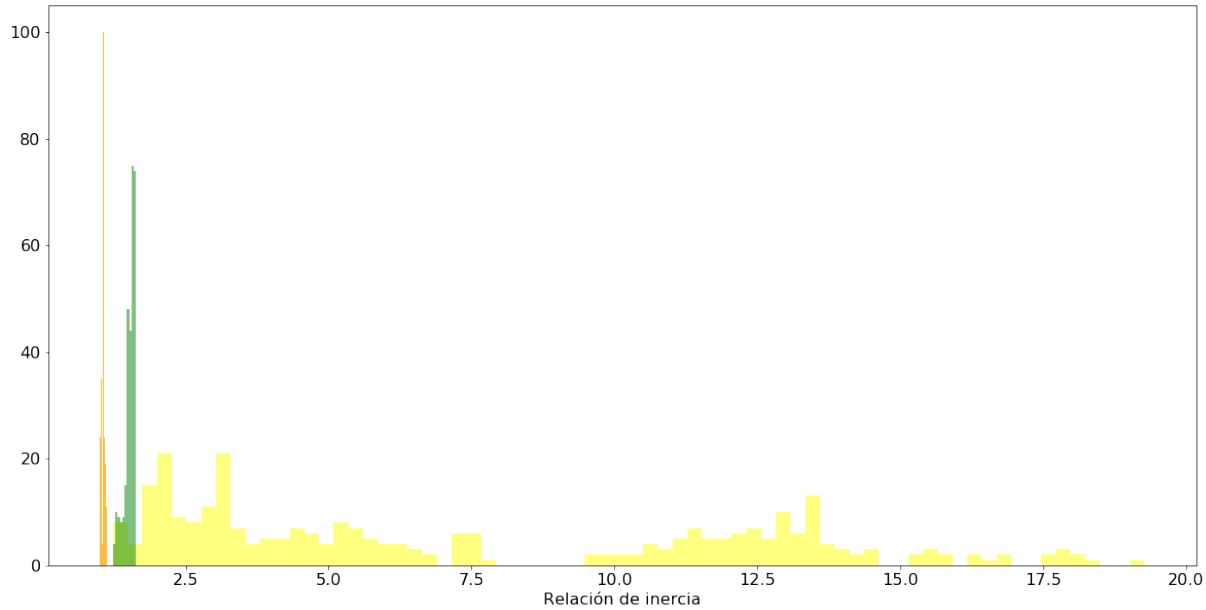

Figura 4: Histograma de las relaciones de inercia para las distintas frutas. Naranjas en color naranja, limones en verde y bananas en amarillo

Aunque se quiso utilizar la relación de inercia como una de las características, se observó en la implementación que su introducción empeoraba el rendimiento del agente. Es por esto que se deja la exploración de este análisis para trabajos futuros.

2.5.2. Hu Moments

Otra alternativa fue utilizar los *Hu moments*. Estos son un grupo de 7 valores propuestos por Hu [1] [2] que son invariantes a la translación, rotación y escala de la imagen. Estos se calculan en función de los momentos centroidales del objeto.

Los valores y las escalas de los 7 Hu moments son muy distintos entre sí, con diferencias de hasta 3 o 4 órdenes de magnitud entre ellos y escalas en el orden de 10^{-12} o 10^{-14}. Es por esto que se aplica una transformación logarítmica para que los valores sean comparables entre ellos y que además las escalas de los números sean más apreciables para un humano.

$$H_i = -sign(H_i) * log_{10}(abs(H_i))$$

De esta forma se obtuvieron los histogramas para las 7 componentes (de 0 a 6) de los Hu moments para el conjunto de training de *Fruits 360*. Esto se observa en la figura 5. Queremos elegir como características aquellas componentes que presenten la mayor separación entre las frutas, es decir, que nos permitan clusterizar de una mejor forma el dataset.

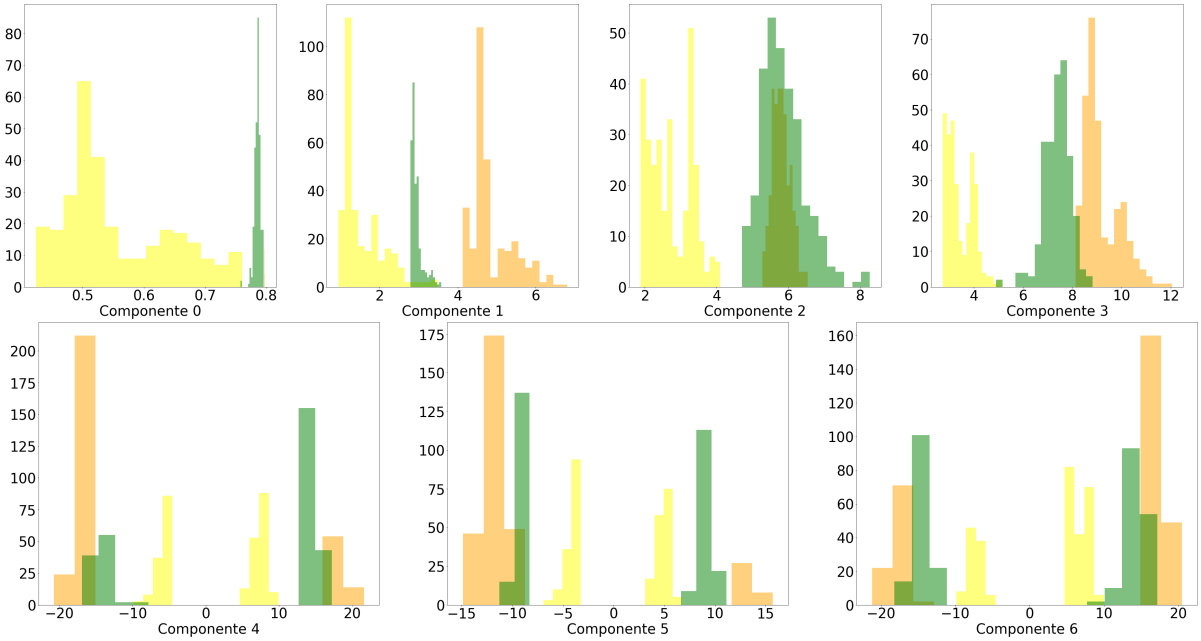

Figura 5: Histograma de los Hu moments para las frutas de Fruits 360. Colores como en la figura 4. Componentes de 0 a 6, nótese la buena separación en las componentes 1 y 3.

Se observa que las componentes 1 y 3 presentan una buena separación de las frutas, por lo que se decide tomar estas dos como características para representar cada imagen. En la figura 6 se observa el resultado de ubicar en un plano x-y cada fruta del set de training de Fruits 360 y la clusterización con *k-means*.

2.6. Algoritmos

2.6.1. K-means

K-means es un algoritmo de clustering en el que se intenta particionar n observaciones en k clusters, en el cual cada observación pertenece al cluster con el centroide (o media) más cercano.

El algoritmo se puede resumir en los siguientes pasos:

1. Elegir la cantidad de clusters k;
2. Elegir aleatoriamente k puntos, que serán los centroides semilla;
3. Asignar cada punto al centroide más cercano;
4. Computar y ubicar el nuevo centroide de cada cluster;

5. Reasignar cada observación al nuevo centroide más cercano. Si hubo alguna reasignación, repetir el paso anterior. Si no, el modelo está listo y se encuentra en un mínimo.

El modelo está plenamente definido con la ubicación de los k centroides por lo que una vez calculados, las nuevas asignaciones son muy rápidas. Si se quiere asignar un nuevo punto hay que calcular las distancias en espacio de características a los centroides y el nuevo punto pertenecerá al cluster más cercano. Podemos decir que una vez finalizado el cálculo del modelo, *k-means* es muy eficiente en cuanto a tiempo de ejecución y memoria. Este algoritmo no asegura la optimalidad de la solución; dependiendo de los centroides semilla la solución puede terminar en un mínimo local que no caracterice fielmente el dataset. Esto se puede observar en la figura 6.

Para evitar este problema se decidió que cada uno de los tres centroides semilla pertenezcan a cada uno de los tres tipos de fruta. Aunque esta no es una implementación *vanilla* del algoritmo, evita de gran manera caer en situaciones como la que se observa en la figura 6 der. Como la clase de cada fruta del set de training es conocida, no tiene sentido elegir tres al azar y esperar que el algoritmo converja a un resultado favorable cuando se puede "ayudar" al algoritmo y mejorar la performance. Naturalmente, en datasets sin etiqueta esto no se puede realizar.

Otro dato a destacar es que este algoritmo no hace uso de las *labels* de los datos (salvo en lo detallado en el párrafo anterior), sino que únicamente los agrupa por cercanía en espacio de características. Para poder realizar una clasificación, lo que se hizo en este trabajo es tomar los clusters generados y, por voto de la mayoría, elegir la *label* del cluster como se detalla en el cuadro 1. De esta forma, en la figura 6 izq. se puede ver el cluster rojo que está compuesto mayoritariamente por bananas, el azul por limones y el verde por naranjas.

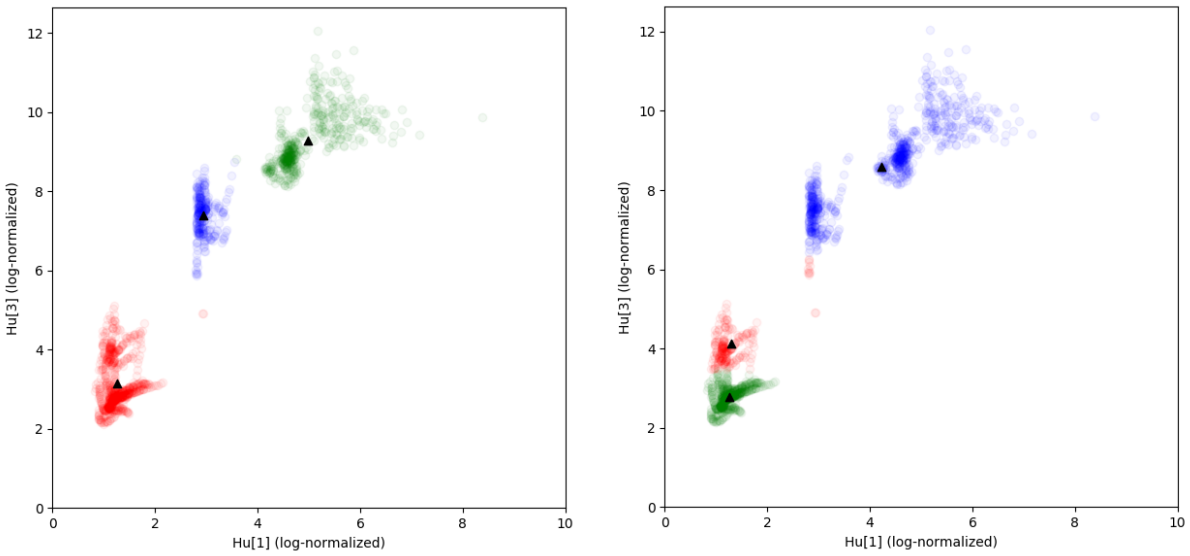

Figura 6: Efecto de la aleatoriedad de los centroides semilla para Fruits 360. A la izquierda se ve una clusterización fiel a los datos de entrada. A la derecha se observa una configuración en un mínimo local no representativa de los datos. En la figura de la izquierda, en rojo el cluster asignado a bananas, en azul el de limones y en verde el de naranjas. Además se observan los centroides de cada cluster.

2.6.2. K-nn

El algoritmo *k-nearest neighbours* o *k-nn* es un algoritmo de inteligencia artificial utilizado para clasificación y regresión. Usado para clasificar, consiste en ubicar cierta cantidad de observaciones en espacio de características (o *feature space*) junto con su respectiva etiqueta. Luego, a cada una de las nuevas observaciones sin etiqueta le será asignada una clase en base a los k vecinos más cercanos. Existen distintas formas de ponderar las contribuciones de los k vecinos; una de ellas es tomar $1/d$ como coeficiente u otra más sencilla, es tomar a cada vecino con una contribución de 1. De esta forma, la clase es asignada por voto de la mayoría. Este enfoque es el utilizado en este trabajo.

Dado que para cada fruta a la cual se le requiere asignar su clase hay que calcular la distancia en espacio de características con todos las frutas del set de training, este algoritmo resulta más ineficiente

en términos de tiempo de ejecución que *k-means* si el set de training es muy grande. El tiempo crece linealmente con la cantidad de ejemplos en el set de training y además hay que realizar un ordenamiento para elegir los k vecinos más cercanos. De acuerdo al algoritmo utilizado para el *sorting* puede variar la complejidad de *k-nn*. Para datasets muy grandes el tiempo de cálculo de distancias puede tornarse excesivamente largo.

Se evaluaron distintas cantidades de vecinos pero no se encontraron grandes diferencias en la performance. Se decidió tomar 20 vecinos ya que esta cantidad generaba resultados aceptables. Una exploración más exhaustiva de este hiperparámetro se deja como trabajo a futuro.

3. Resultados

3.1. K-means

En la primera parte del algoritmo *k-means* se generan los clusters a partir del conjunto de training de *Fruits 360*. Se ubican todas las frutas en espacio de características como se observa en la figura 6. A cada cluster se le asigna la clase de acuerdo al voto de la mayoría, es decir, si en cierto cluster hay mayoría de bananas entonces se lo etiqueta como un cluster de bananas. De esta forma quedan conformados los clusters A, B y C como se ve en el cuadro 1. Esta es la etapa de entrenamiento o *training* y los datos aquí obtenidos se consideran *ground truth*.

Cuadro 1: Etapa de training de *k-means*. Cantidad de frutas en cada cluster y los respectivos centroides. Por voto de la mayoría, se asigna una clase a cada cluster.

	Clusters		
	A	B	C
Bananas	298	0	0
Naranjas	0	304	0
Limones	0	3	293
Centroide	[1.60; 3.54]	[5.07; 9.03]	[2.99; 7.64]
Identificado como:	'banana'	'orange'	'lemon'

Las asignaciones de las frutas de los conjuntos de test se hacen en base a las medias obtenidas a partir del conjunto de training. De esta forma, se observa en el cuadro 2 el desempeño del algoritmo evaluándolo con los sets de test de *Fruits-360* y con el dataset propio. Este último nunca fue utilizado para entrenar y es por lo tanto información totalmente nueva para el algoritmo. Es decir, se evalúa el desempeño con imágenes muy distintas a las utilizadas para entrenar. Aún así se obtienen buenos resultados aunque se debería aumentar la cantidad de imágenes del dataset propio para tener más fuerza estadística. La efectividad se calcula como la cantidad de aciertos (asignaciones correctas) sobre el total de imágenes.

Como se explicó, el cuadro 2 muestra cuántas frutas de cada clase se han asignado a cada cluster. El set de test de *Fruits-360* muestra una altísima efectividad por la gran similitud que hay entre entre las imágenes de training y las de test. El dataset propio contiene imágenes "desconocidas y con el adicional de haber sido tomadas de forma casera por lo que nunca se esperaría un rendimiento del 100 %, sino siempre menor.

Cuadro 2: Etapa de test de *k-means*. Se observa la cantidad de frutas de cada clase asignado a cada cluster. Por ejemplo, se observa que para el dataset propio, en el cluster *C: lemon* se han asignado hay 2 naranjas y 14 limones. La designación *A*, *B*, *C* es consistente con el cuadro 1.

		A: 'banana'	B: 'orange'	C: 'lemon'	Efectividad
Fruits-360 (test)	Bananas	166	0	0	100 %
	Naranjas	0	160	0	
	Limones	0	0	164	
Dataset propio	Bananas	16	0	0	91.7 %
	Naranjas	0	14	2	
	Limones	2	0	14	

3.2. K-nn

En este algoritmo también existe una etapa de training. En ella se tiene que generar, igual a como se explicó anteriormente, un mapa de características en el cual se ubican todas las frutas del set de training de *Fruits-360*. Nuevamente esta clasificación es considerada como *ground truth* y a partir de ella se clasificarán las frutas del conjunto de test de *Fruits-360* y del dataset propio. La asignación se realiza en base a los 20 vecinos más cercanos. Este número fue encontrado empíricamente pero no se realizó un estudio extensivo del mismo. Esto se deja como trabajo a futuro.

Los resultados se observan en el cuadro 3.

Cuadro 3: Etapa de test de *k-nn*. Se observa a qué clase se asignaron las frutas de cada dataset. Por ejemplo, de las 16 naranjas del dataset propio, 15 fueron categorizadas correctamente y una fue identificada como limón.

		Identificado como			Efectividad
		Banana	Naranja	Limón	
Fruits-360 (test)	Bananas	166	0	0	100 %
	Naranjas	0	160	0	
	Limones	0	0	164	
Dataset propio	Bananas	16	0	0	89.6 %
	Naranjas	0	15	1	
	Limones	4	0	12	

Sorpresivamente los resultados son muy similares a aquellos del algoritmo *k-means*, con un solo acierto menos de diferencia. Esto indicaría que la naturaleza de los datos aquí analizados son favorables para cualquiera de los dos algoritmos analizados. Es por eso que se deben tener en cuenta otros factores a la hora de elegir uno de los algoritmos.

Como se explicó anteriormente, una vez realizado el training, *k-means* presenta una gran performance en tiempo de ejecución y en memoria. Esto es porque todo el modelo queda representado en los k centroides de los k clusters. Así es que para cada nueva asignación, solo hay que calcular k distancias y la menor de ellas definirá la clase. Por otro lado, *k-nn* tiene el mismo *overhead* a la hora del training pero además, a la hora de las asignaciones, debe realizar n cálculos. Siendo n la cantidad de ejemplos del dataset de training, si no hay una suficiente capacidad de cómputo a disposición, este algoritmo tendrá una performance muy inferior en tiempo de ejecución siempre que la cantidad de ejemplos sea mayor a la cantidad de clusters, lo cual es algo común. Además hay que tener en cuenta que el modelo queda caracterizado con todos los ejemplos del dataset de training y no solo unos pocos centroides. Esto genera un uso de memoria mayor. Sin embargo *k-nn* tiene un desempeño superior cuando las frutas no están claramente dentro de un cluster sino en los exteriores. En este caso *k-means* tiende a generar clusters del mismo tamaño por lo que no puede clasificar igual de bien a los ejemplos que caen en la periferia de un cluster.

Por esto es que se recomienda utilizar *k-means* para aplicaciones en las que no se tenga una gran capacidad de cálculo o que se deban hacer en un tiempo muy pequeño. Cuando los recursos no son problema se aconseja el uso de *k-nn* ya que presenta mejores resultados en ejemplos ubicados en zona de interfaz de clusters.

De esta forma se recomienda el uso del algoritmo *k-means* para una caja de supermercado ya que provee buenos resultados, no se necesita hardware muy potente y, sobre todo, puede dar un resultado en un tiempo muy bajo. Esta característica es crucial en este entorno. *K-nn* también podría implementarse pero requiere una mayor inversión en equipamiento.

4. Conclusión

En este trabajo se diseñó un agente inteligente capaz de identificar y clasificar frutas entre bananas, naranjas y limones. En la primera sección se da una clasificación formal del agente especificando la tabla REAS y las propiedades del entorno de la tarea. El agente actúa sobre dos datasets, uno es *Fruits 360* y el otro es un dataset propio. Luego se explican los métodos utilizados para filtrar las imágenes y realzar características de las mismas. Se utilizó un filtro gaussiano y posteriormente un *thresholding* con el algoritmo *Triangle*.

A la hora de elegir las características a extraer se avanzó primero en un método geométrico en el cual se utilizaba la relación de los momentos principales de inercia para evaluar el "alargamiento"de las frutas.

Este enfoque se debió abandonar ya que no generó buenos resultados pero se decidió documentar debido a su sencillez y la posibilidad de expandir su uso en trabajos futuros. El otro enfoque finalmente utilizado es también geométrico y usa los Hu moments. Las componentes 1 y 3 presentaron la mejor clusterización por lo que estos dos valores fueron usados para representar cada imagen.

Luego se explicó la implementación de los algoritmos y los resultados obtenidos. La performance de ambos fue muy satisfactoria obteniendo una efectividad del 100 % para las frutas de *Fruits 360* y alrededor de 90 % para las del dataset propio. Como no se encontraron diferencias significativas en los desempeños la decisión de utilizar uno u otro dependerá de las características particulares de la implementación. Para aquellas en las que no haya disponible mucha capacidad de cómputo ni memoria se recomienda utilizar *k-means* por su bajo impacto en ambas variables. Si el procesamiento y la memoria no son problema se recomienda *k-nn* ya que tiene un desempeño ligeramente superior en ejemplos que no encajan perfectamente en un cluster.

Dado que el agente debe implementarse en una caja de supermercado y en este entorno el tiempo de ejecución y el bajo costo son características cruciales, se recomienda el uso de *k-means*.

Referencias

[1] Ming-Kuei Hu. "Visual pattern recognition by moment invariants". En: *Information Theory, IRE Transactions on* 8.2 (feb. de 1962), págs. 179-187. DOI: 10.1109/TIT.1962.1057692.

[2] Zhihu Huang y Jinsong Leng. "Analysis of Hu's Moments invariants on Image Scaling and Rotation". En: *2010 2nd International Conference on Computer Engineering and Technology* (abr. de 2010). DOI: 10.1109/TIT.1962.1057692.

[3] Horea Muresan. "Fruit recognition from images using deep learning". En: *Acta Univ. Sapientiae, Informatica* 10 (2018), págs. 26-42. URL: https://www.researchgate.net/publication/321475443_Fruit_recognition_from_images_using_deep_learning.

[4] Stuart Russell y Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in Artificial Intelligence. 2010.

[5] Stéfan van der Walt y col. "scikit-image: image processing in Python". En: *PeerJ* 2 (jun. de 2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: https://doi.org/10.7717/peerj.453.

[6] Joshua Warner, Juan Nunez-Iglesias y Stéfan van der Walt. *Image Analysis in Python with SciPy and Scikit-image*. SciPy 2019 - Austin, Texas. 2019. URL: https://github.com/scikit-image/skimage-tutorials.