Trabajo Final Inteligencia Artificial 1

Año 2020: Visión Artificial

Profesor:

Prof. Titular Dra. Ing. Selva S. Rivera

Autor:

Bruno, Simón

Resumen

En este informe se hará una breve exposición del desarrollo del Proyecto Final de Inteligencia Artificial 1, donde se mostrará un poco cómo se llegó al resultado final y analizando cada parte afrontada. También se explicará en qué consiste la implementación real de cada una de las distintas etapas del plan trabajo junto con sus principales complicaciones. También se incluye un ejemplo de aplicación del código desarrollado, que junto con las imágenes adjuntadas, ayudan a visualizar mejor los resultados. Al final hay conclusiones obtenidas directamente de cada etapa de trabajo y una breve explicación de su relevancia.

Introducción

El problema a resolver es la construcción de un sistema de reconocimiento de frutas a través de visión artificial por computadora. Esto consiste en realizar un código que reconozca ciertas imágenes de entrenamiento, luego aplicarles ciertos filtros y prepararlas, para formar una base de datos con la que comparar cada una de las imágenes de prueba y así reconocer la naturaleza de cada fruta. Esto fue posible con la implementación de los métodos Knn y Kmeans estudiados durante el cursado. Fue utilizada la librería OpenCV de Python en Spyder de manipulacion de imagenes.

La consigna consistía en reconocer frutas en las cajas de un supermercado para agilizar el proceso de cobro. Se desarrolló un agente prototipo que puede reconocer bananas, naranjas, limones y tomates a partir de fotografías:

Figura(1)

El principal problema que hubo que enfrentar fue la gran variación de las características de las imágenes, por lo que según la base de datos que se utilizara, variaba más o menos la dispersión de datos, y consecuentemente, la eficiencia del código. Entonces para sacar conclusiones reales y constructivas sobre los resultados del código, hubo que invertir bastante tiempo en aprender cómo adecuar y procesar las imágenes correctamente.

Según Wikipedia: *"La visión artificial, también conocida como visión por computadora o visión técnica, es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un ordenador."* Para la tarea propuesta fue necesario investigar sobre la visión artificial,

manipulación de imágenes, utilización de filtros y así poder preparar las imágenes para ser utilizadas correctamente. En este caso, se utilizó una base de datos de archivos de fotos, pero el paso siguiente de complejización de este tipo de proyectos consiste en utilizar una cámara en tiempo real para la obtención de imágenes.

Especificación del agente

REAS

Agente	Medidas de rendimiento	Entorno	Actuadores	Sensores
. Visión artificial de frutas	Maximizar: . efectividad . constancia en los resultados . distinción entre objetos y entorno Minimizar: . errores en el reconocimiento de los objetos	. caja de supermercado	. cinta transportadora . caja registradora	. camaras en vivo

Propiedades del entorno de trabajo

Entorno de trabajo	Observable	Determinista	Episódico	Estático	Discreto	agentes
contenedor lleno de frutas	parcialmente	estocástico	episódico	dinámico	discreto	individual

Diseño del agente

Investigación sobre OpenCV

La primera etapa consistió en investigar sobre las distintas librerías disponibles para la realización del proyecto. Se eligió OpenCV por ser de las más comunes y utilizadas. Luego, hubo que analizar cómo implementar las funciones de la librería propias para las tareas a realizar sobre cada imagen. Por ej, resize, threshold, blur, etc.

Realizar código de preprocesing

Utilizando alguna imagen de prueba, se empezó a construir el código para adecuar las imágenes. La etapa de investigación continua aquí para ir descubriendo distintas posibilidades o formas de alcanzar algún objetivo en específico.

Sacar fotos y adecuarlas

Una parte importante del proyecto es utilizar imágenes propias, pero tuve que adecuarlas a un marco cuadrado y recortarles el fondo para obtener mejores resultados. Si no, las posibilidades de dispersión de los datos aumentan mucho por información propia del fondo de cada imagen. Este proceso se hizo 2 o 3 veces porque las frutas y las imágenes no aportaban datos semejantes.

Esta etapa terminó después de alrededor de 2 días de trabajo, pero se siguió ajustando según los nuevos parámetros que comencé a utilizar o cambié.

Realizar código de Knn

Ya con imágenes un poco más estandarizadas, fue posible comenzar el análisis propio de la Inteligencia Artificial y la creación del código de clasificación. Elegí el Knn porque me pareció un poco más sencillo de programar y sus conclusiones me iban a ayudar para facilitar el Kmeans. El desarrollo entero duró aproximadamente 1 dia pero tuvo 1 dia mas de ajuste de imágenes y parámetros para ajustar a resultados

coherentes y corregir errores que afectan directamente el funcionamiento del código.

Realizar código de Kmeans

El código del Kmeans tambien tardó 1 dia entero, pero como ya tenía los resultados verificados del Knn fue un poco más fácil la etapa de revisión y lo pude terminar más rápido. Al igual que en el código anterior, al final también tuve que corregir errores del programa para que devuelva datos, y después analizarlos para corregir parámetros e incoherencias.

Corrección de parámetros

Luego de verificar la funcionalidad de todo el código, corresponde la variación de ciertos números para "jugar" con los distintos valores de entrada para las funciones y analizar los distintos resultados. Se trata de variar los parámetros y tratar de obtener mejores rendimientos y sacar nuevas conclusiones.

Realización del informe

Una vez corregido el código y habiendo obtenido una serie de conclusiones, llega la etapa de plasmar todo eso en un informe que será presentado durante la instancia de examen. Aquí se muestran las gráficas pertinentes y las distintas dificultades que hubo que superar para llegar a la solución del problema.

Código

El código consiste en 4 módulos, separados siguiendo una lógica de realización de tareas. Así tenemos el código que se encarga del procesamiento de las imágenes y la obtención de los parámetros; un módulo para el Knn y uno para el Kmeans, que reciben los datos de las imágenes de entrenamiento y devuelven resultados que luego se comparan con los datos de cada una de las imágenes de prueba; y por último tenemos el main que centraliza la distribución de los parámetros y la obtención de valores. Se encarga de llamar a los módulos y mostrar en pantalla las conclusiones del código.

El código con todos sus archivos se encuentra adjunto en la carpeta del Proyecto Final.

Ejemplo de aplicación

El código obtiene todos los datos necesarios dentro del *main* y luego, en un *for* llama a cada método según la cantidad de imágenes a comprar y printea los resultados. También guarda los aciertos en una variable y al final calcula el porcentaje de efectividad.

```
for i in range(len(tests[0])):
    distances = knn.get_distances(dataset,tests[:,i])
    neighbours, fruit_neighbours, result_knn = knn.knn(distances, fruits,3)
    print('La fruta segun el knn es ' + result_knn)
    print('La fruta analizada es ' + tested_fruits[i])
    print('----------------------------------------')
    if(result_knn == tested_fruits[i]):
        precision_knn +=1

    centroids = kmeans.centroides(dataset)
    result_kmeans, distance_kmeans = kmeans.kmeans(tests[:,i], centroids, fruits_kmeans)
    print('La fruta segun el kmeans es ' + result_kmeans )
    print('La fruta analizada es ' + tested_fruits[i])
    print('----------------------------------------')
    if(result_kmeans == tested_fruits[i]):
        precision_kmeans +=1
```

Figura(2)

Entonces cada imagen se compara con cada uno de los métodos y devuelve un vector de frutas ordenadas del cual va a obtener el resultado correcto y muestra si acertó o no.

```
----------------------------------------
['naranja', 'tomate', 'naranja']
La fruta segun el knn es naranja
La fruta analizada es naranja
----------------------------------------
['naranja', 'limon', 'tomate', 'banana']
La fruta segun el kmeans es naranja
La fruta analizada es naranja
----------------------------------------
```

Figura(3)

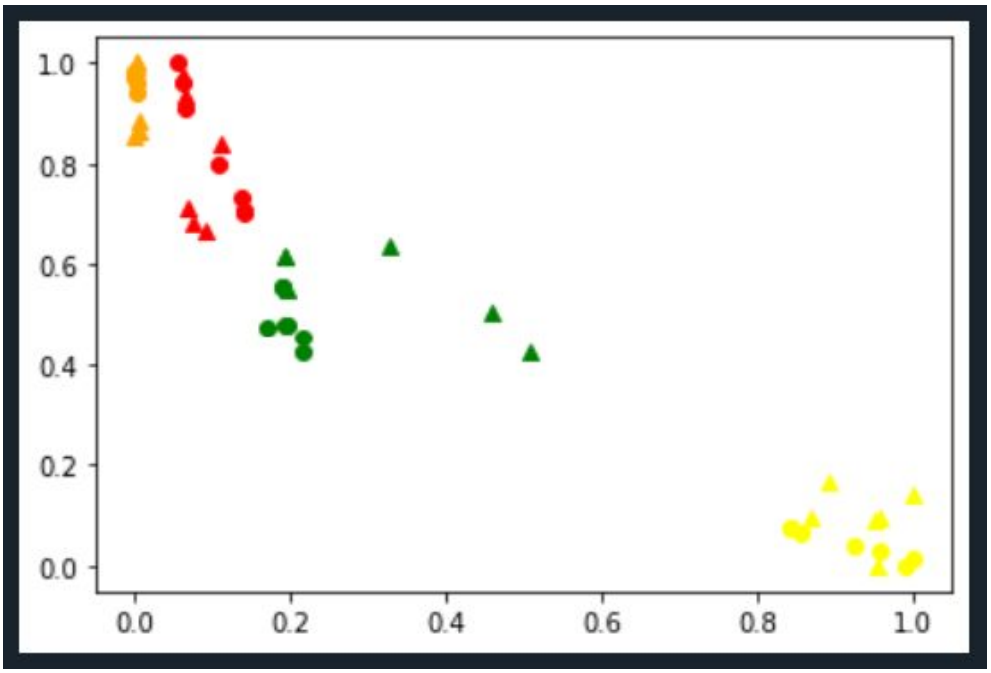
Luego muestra la precisión de cada método.

```
------------------------------------
La precision del knn es de: 100.0
La precision del kmeans es de: 75.0
```

Figura(4)

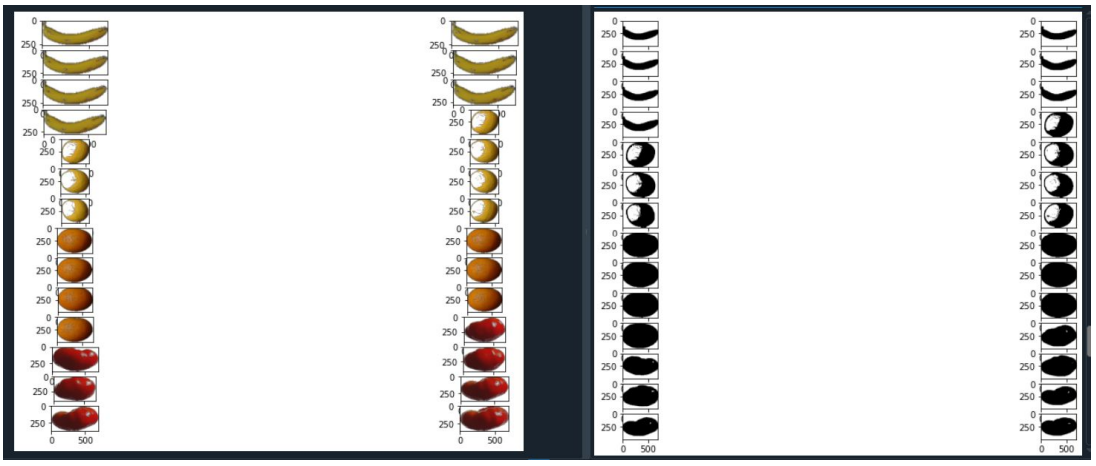
También se incluye un scatter de los datos para visualizar un poco cómo serán los resultados del código. En amarillo se muestran las bananas, en verde los limones, en naranaja las naranjas y en rojo los tomates. El eje x representa la suma de los momentos hu 1 y 3 sumandos, y el eje y es un promedio del histograma de colores de cada imagen.

Esta imagen fue de gran utilidad para analizar la variación de los centroides iniciales del Kmeans, ya que forzar unos centroides más adecuados a cada grupo ayudaba a conseguir mejores resultados. Sin embargo, el Kmeans tiene problemas para distinguir entre naranjas y tomates por la cercanía de los datos. Según donde se inicien los centroides correspondientes, varía la eficiencia del código.

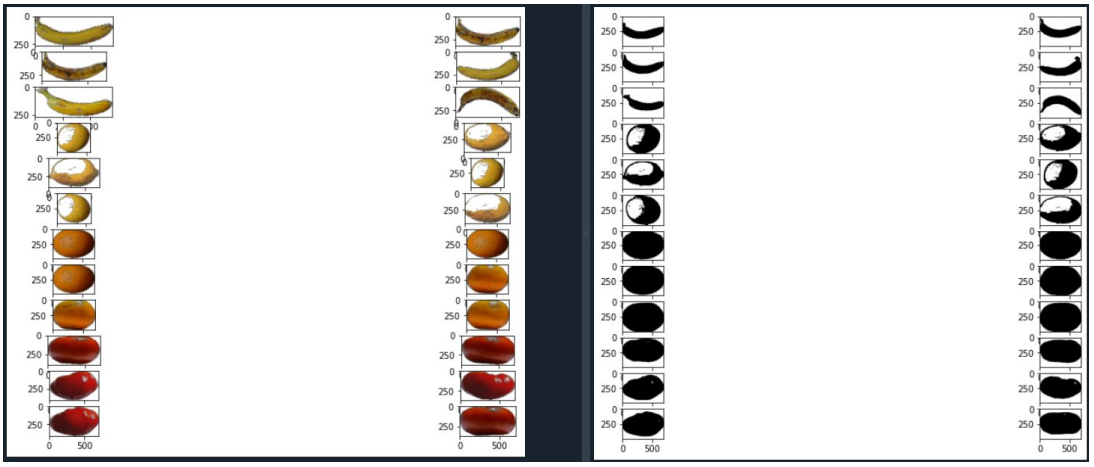


Figura(5)

También se incluye ploteos de las imágenes utilizadas, tanto de entrenamiento como de prueba, en sus versiones a color y en binario.



Figura(6)



Figura(7)

Resultados

Los resultados del código son de una eficiencia mayor para el método del Knn que para el Kmeans. Estos varían según los momentos hu que se tomen, ya que con uno solo de los momentos, ambos métodos bajan la efectividad. Así se refuerza la elección de la suma de momentos.

Las imágenes muestran un correcto reconocimiento en la mayor parte, salvo en los limones, pero no afecta a los resultados del código. También se puede ver que gracias a la eliminación del fondo de cada imagen, no hay problema con las sombras producidas sobre la superficie de apoyo, lo que muchas veces trae problemas a la efectividad. Todo esto ayudó bastante a que las imágenes formarán grupos paramétricos compactos y así mantener constancia en los resultados.

También se puede analizar los resultados analizando el plot del scatter, y así tratar de predecir los resultados de cada método, según se varíe la cantidad de vecinos a considerar o la ubicación inicial de los centroides.

Conclusiones

A lo largo de la realización del proyecto, llegué a varias conclusiones importantes:

- Las imágenes utilizadas influyen mucho más de lo que uno pueda llegar a pensar en un principio, ya que la dispersión de datos que se produce puede ser grande. Por esta razón, tomarse el tiempo para sacar fotos adecuadas ayuda bastante. También se puede concluir que resultados bajos de efectividad del código no significan necesariamente algún error conceptual o de programación.
- Eliminar el fondo de las imágenes ayuda bastante a conseguir mejores resultados.
- Contrariamente a lo que se puede creer inicialmente, una base de datos de entrenamiento demasiado grande y variada también produce dispersión de datos, arruinando la efectividad del código. Por esto es mejor utilizar imágenes parecidas para analizar los alcances de cada método.
- Es fundamental eliminar las unidades de cada dato, para poder compararlos correctamente.
- Hay que buscar parámetros que nos separen los grupos de datos, para poder acceder a ellos más efectivamente.
- Elegir muchos parámetros complica la visualización de los grupos de datos según las frutas, ya que no se puede plotear graficos de mas de 3 dimensiones, por lo que se complica encontrar las fallas del código. Es mejor utilizar menos parámetros y plotear datos.
- Para poca cantidad de datos, el Knn se comporta mejor, ya que el Kmeans depende del promedio entre los datos, y esto mejora con mayor cantidad de imágenes de entrada.

Bibliografía y/o Referencias

https://es.stackoverflow.com/

https://docs.opencv.org/2.4/modules/refman.html

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html

https://numpy.org/