Práctica 3: Caracterización de Texturas

Reconocimiento de Patrones - 0757

Facultad de Ingeniería
Universidad Nacional Autónoma de México

Ceballos Equihua C. N.
Ingeniería en Computación
Facultad de Ingeniería, UNAM
Ciudad de México, México
ceballos.equihua@gmail.com

Muñoz Marbán J.
Ingeniería en Mecatrónica
Facultad de Ingeniería, UNAM
Ciudad de México, México
juanmzmb@gmail.com

Murrieta Villegas A.
Ingeniería en Computación
Facultad de Ingeniería, UNAM
Ciudad de México, México
alfonsomvmx@gmail.com

Solano González F. J.
Ingeniería en Mecatrónica
Facultad de Ingeniería, UNAM
Ciudad de México, México
felipe.solano.gos@gmail.com

Profesores:
Dr. Boris Escalante Ramírez
Dra. Olveres Montiel Jimena
I.I.M.A.S. - UNAM

Resumen— **Esta práctica consite en encontrar patrones a través de la caracterización y clasificación de texturas mediante entidades denominadas "texeles", además de otros elementos como la matriz GLCM, por otro lado, con el uso de diferentes modelos de aprendizaje supervisado como SVM o KNN es como se realiza la clasificación de dichas texturas.**

I. Objetivo

Desarrollar métodos de caracterización de texturas para posteriormente emplear modelos supervisados como medio de clasifación.

II. Introducción

Uno de los recusos más utilizados en el campo de la visión artificial y en el reconocimiento de patrones en conjunto de datos de tipo blob como es el caso de imágenes es el uso de texturas.

A través de aspectos como las diferentes propiedades estadísticas que cada píxel nos ofrece o la dispersión que esta información tiene respecto a sus pixeles vecinos es como podemos denotar los patrones de comportamiento de cada textura involucrada en nuestra imagen.

Además, como parte fundamental del procesamiento de imágenes y específicamente de clasificación de texturas, no solamente debemos considerar elementos como matrices GLCM para entender las texturas sino a su vez modelos de aprendizaje para poder realizar una posterior clasificación o incluso predicción.

A continuación, se explican algunos modelos propuestos para la clasificación de texturas en la presente práctica.

SVM

Una Máquina de Vectores de Soporte o en inglés SVM se basa en la extracción de la funcionalidad mediante la aplicación de aproximaciones del gradiente, dicho modelo se basa en una función de distribución la cual se utiliza para determinar la probabilidad de que un elemento de una clase sea la clase correcta.

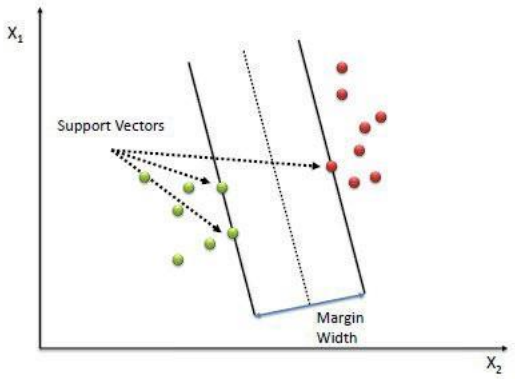

Fig. 1 Máquina de Vectores de soporte.

Precisamente la distribución de probabilidad de una SVM tiene la forma de una función de densidad de probabilidad, por lo que el valor de la función de densidad de probabilidad puede ser evaluado para cada elemento conjunto de la clase, o para cada elemento del conjunto de entrada.

A su vez es necesario destacar que un elemento importante en las SVM son el kernel o nucleo, el cuál se encarga de poder contemplar la forma en que se trabajará el hiperplano creado para nuestro conjunto de datos.

KNN

K-Nearest-Neighbor o KNN es un algoritmo que sirve principalmente para encontrar o agrupar elementos. Precisamente este algoritmos se basa primero en considrar los atributos que se encuentran en las muestras conocidas los cuales se definirán como los distintos números de atributo.

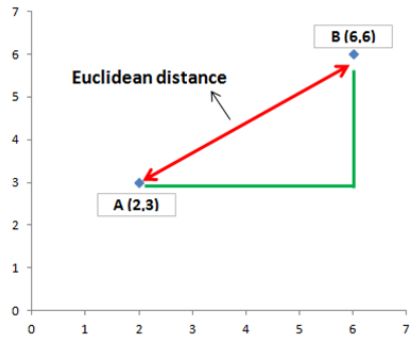

Fig. 2 Distancia euclidiana.

De esta manera, se puede comparar la distancia entre la muestra nueva y las muestras conocidas, como aspecto importante es necesario denotar que existen disintos métodos para poder obtener la distancia entre los elementos como es el caso de la distancia Euclideana u otras más.

III. Desarrollo

Para la presente práctica tenemos 2 principales secciones, la primera consiste en varias funciones asociadas a la separación de las imágenes originales en "texeles" además de contemplar aspectos estadísticos de estas entidades, por otro lado, y como segunda parte de la práctica, la sección corresponde a los modelos de aprendizaje automático, los cuales nos ayudarán con los datos de la primera sección a clasificar las texturas de cada imagen.

Sección 1: Funciones para previas a la clasificación

Lo primero que se realizó fue trabajar directamente con los elementos asociados directamente a las texturas como son 2 principales aspectos:

1. La matriz GLCM
2. Los aspectos o características estadísticas asociadas a cada texel

Es por ello que para esta parte se realizaron 3 funciones encargadas de este apartado:

Función de obtención de la matriz GLCM:
Como argumento se pasa a una matriz asociada a los texeles de una imagen para posteriormente obtener la matriz GLCM resultante.

Función de obtención de características
Esta función tiene como principal objetivo que al pasarle la matriz GLCM obtener todos los aspectos característicos de cada matriz asociada.

Función de principal para características:
Se desarrolló una función que asociara las 2 anteriores funciones con el objetivo de recopilar toda la información necesaria para posteriormente guardarla en un vector con todas las características estadísticas del texel.

Sección 2: Funciones y modelos para la clasificación

Para este aparto lo primero que se trabajó fue la llamada de las funciones principales además de pasar de cambiar de RGB a B&W cada una de las imágenes

A su vez, contemplando que este ejercicio estaba pensando para clasificación se decidió tomar un tamaño del 80 por ciento del total de la muestra para el entrenamiento y un 20 por ciento para la parte de prueba o test.

Clasificador basado en KNN

Para el uso de este modelo, lo primero que se realizó fue obtener el coeficiente K del algoritmo para lo que se procedió a emplear el score asociado a la predicción respecto a el número asociado a K, a continuación, se muestra el diagrama obtenido:

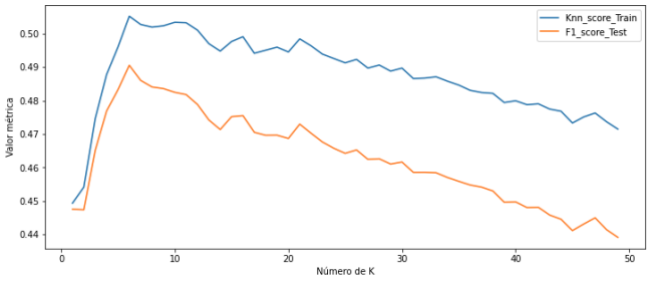

Fig. 3 Validación del número K para su posterior uso en el mismo modelo

Como observamos de la figura anterior, el número K con mejor resultado de entrenamiento es 6.

Posteriormente para ello se entró el modelo obteniendo un resultado de score en el Train de alrededor de 0.50, recordemos que si bien el resultado a priori parece ser relativamente pequeño, sabemos que el verdadero valor a conocer es el score o resultado de validación en el conjunto o dataset de prueba:

```
[ ] KNN_MODEL = KNeighborsClassifier(n_neighbors=F1_k_max)
    KNN_MODEL.fit(X_train,Y_train)
    Y_pred = KNN_MODEL.predict(X_test)
    #Metricas finales
    print("KNN_SCORE final: ",KNN_MODEL.score(X_test,Y_test))
    print("F1_SCORE final:  ",f1_score(Y_test,Y_pred,average='macro'))

    KNN_SCORE final:  0.5052083333333334
    F1_SCORE final:   0.4905494288645969
```

Como bien se menciona anteriormente, la forma de poder validar el resultado obtenido con nuestro clasificador mediante KNN es empleando una matriz de confusión donde sabemos que el mayor peso de esta debe estar en la diagonal principal, a continuación, el resultado obtenido:

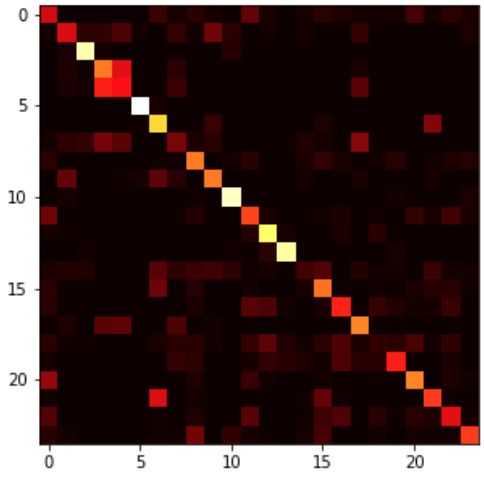

Fig. 4 Matriz de confusión obtenida con los datos de prueba en el clasificador basado en KNN

Observamos que hay una gran presencia en la diagonal principal de nuestra matriz lo cual a priori es un buen indicio del resultado obtenido, sin embargo, es necesario denotar que existen algunos resultados volatiles que si bien no son la mayor cantidad de los eventos si llegan a ser en algunos casos ser notables.

Clasificador basado en SVM

Para este clasificador al igual que el KNN se procedió a encontrar el kernel o núcleo que mejor se ajustara a nuestros datos. A continuación, se muestra el resultado obtenido:

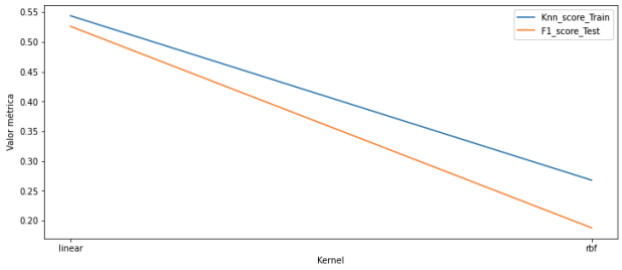

Fig. 5 Determinación del tipo de kernel asociado a nuestra máquina de soporte vectorial

Como podemos observar en el resultado anterior, notamos que se sugiere emplear un núcleo de tipo lineal.

A continuación se muestra el resultado obtenido del entrenamiento empleando un Kernel Lineal:

Como bien se mencionó en el caso anterior con el clasificador basado en KNN, el resultado obtenido de la parte de prueba no muestra el resultado o accuracy de nuestro modelo, para este caso también se realizó su respectiva matriz de confusión:

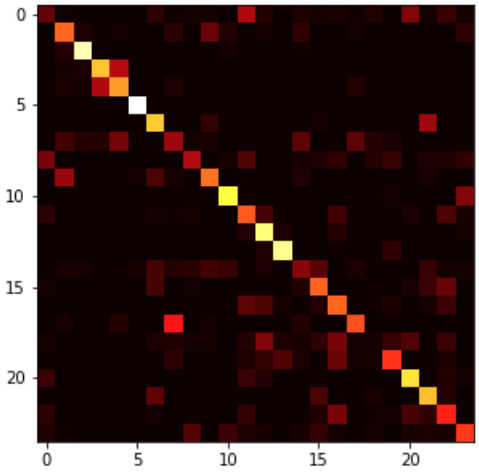

Fig. 6 Matriz de confusión obtenida con los datos de prueba en el clasificador basado en SVM

Observamos que el resultado a diferencia del clasificador KNN tiene menor dispersión de los datos, sin embargo, también notamos una presencia de datos volátiles no asociados a la diagonal principal.

Resultados obtenidos con KNN y SVM

Una de las mayores diferencias de la presente práctica respecto a la del clasificador Bayesiano basado en colores es que tanto para el clasificador KNN como para el SVM empleamos los vectores característicos asociados a los texeles de nuestras imágenes, es decir, para este caso mientras más se parezcan los texeles en el resultado obtenido en la comparación de la imagen original respecto a la final es como podremos observar que tan buen resultado obtenemos en nuestro clasificador, a continuación se muestran algunos resultados obtenidos mediante ambos clasificadores:

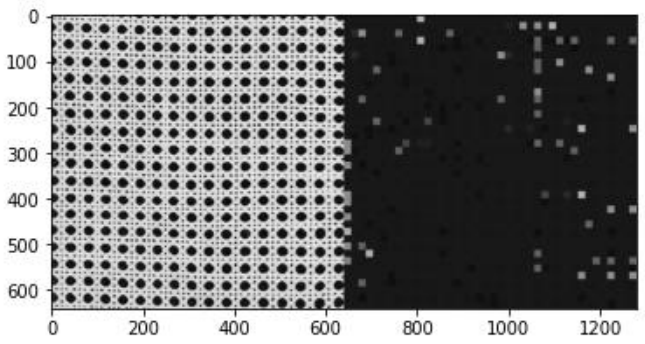

Fig. 7 Imagen clasificada con el modelo basado en SVM

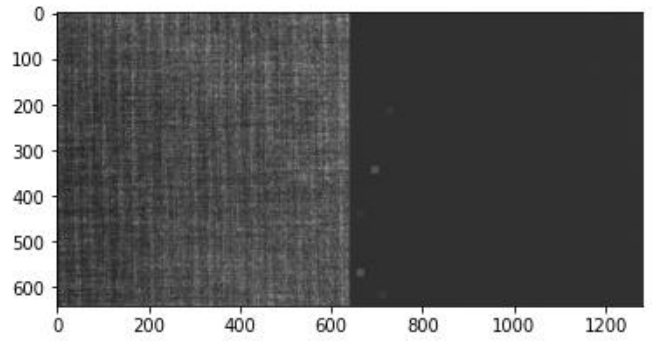

Fig. 8 Imagen clasificada con el modelo basado en SVM

Como podemos observar en las figuras 7 y 8 al ser texturas que sonbastante homogenas observamos un buen resultado dado por nuestro clasificador, cabe destacar que también esto se debe a que los texeles dentro de nuestras texturas nov arian mucho respecto a otros caso que a continuación se mostrarán:

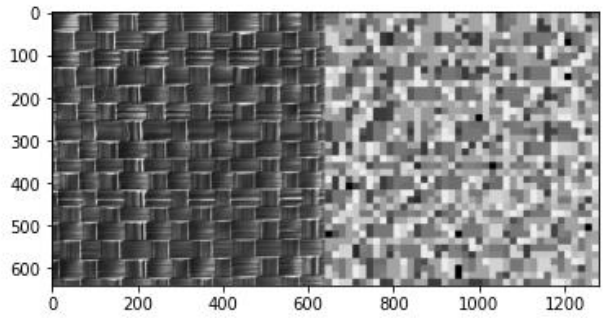

Fig. 9 Imagen no tan bien clasificada con el modelo basado en SVM

A diferencia de los casos anteriores, es notable el problema que encontramos en esta imagen, si bien existen varios casos o texeles que pueden ser bien clasificados a su vez observamos un problema en la dispersión y asociación de texeles con el modelo de predicción .

IV. Conclusión

En la presente práctica aprendimos y manejamos varios conceptos importantes dentro del procesamiento de imágenes como son la matriz GLCM y sobre todo texel. Precisamente a través de estos 2 principales conceptos es como encontramos patrones de similitud mediante herramientas estadísticas para posteriormente poder emplearlas como recurso principal dentro de 2 principales clasificadores, uno basado en el algoritmo de KNN y otro en una máquina de soporte vectorial basada en un núcleo lineal.

A su vez si bien nuestros resultados no fueron los esperados, notamos que principalmente sufrimos de un probable problema de sobre entrenamiento, debido a que obtuvimos casos con una asertividad extrema mientras que a su vez tuvimos casos relativamente dispersos respecto a otros casos particulares.

Como parte final de esta práctica queremos mencionar que una posible solución al resultado obtenido con ambos modelos podría ser el poder particularizar el texel respecto a cada imagen, es decir, asociar un ángulo, y tamaño particular a cada texel dentro de cada imagen para de esa forma poder determinar apropiadamente en el modelo de predicción las texturas de las imágenes del conjunto de prueba-

V. NOTAS

Como parte extra de la práctica decidimos agregar una nueva prueba con un tamaño de texel mucho mayor, a continuación se muestra la dispersión de los resultados en la matriz de confusión:

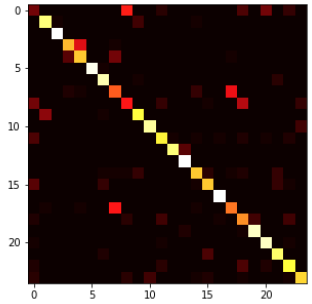

Fig. 10 Matriz de confusión obtenida con el nuevo tamaño de texel

Podemos observa un notable cambio respecto a las matrices anteriores, incluso nos confirma el accuracy del 0.75 dado por el modelo basado en SVM. Sin duda, aquí notamos que la importancia y tamaño del tamaño del texel es fundamental para poder determinar de mejor o peor manera las texturas.

Para terminar, también adjuntamos un ejemplo claro de lo asertivo que resulta el modelo contemplando un tamaño de texel de 40 en vez de 16 pixeles.

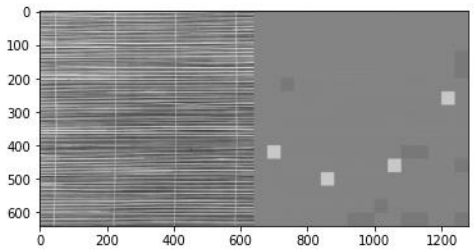

Fig. 10 Clasificación de la textura mediante el segundo tamaño de texel

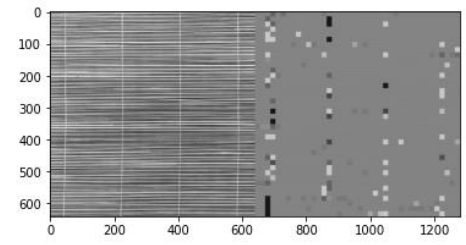

Fig. 11 Clasificación de la textura mediante el primer tamaño de texel (16)

En las figuras 10 y 11 podemos observar el cambio significativo de la clasificación de la textura considerando un tamaño mayor de texel, lo cual no solo nos garantizó particularmente la sertividad en una textura sino a su vez un buen resultado respecto a las demás imágenes o texturas.

Referencias

[1] A. Yadav, «SUPPORT VECTOR MACHINES (SVM),» Towards Data Science, 2018. [En línea]. Available: https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589. [Último acceso: 19 noviembre 2021].

[2] S. Yildirim, «K-Nearest Neighbors (kNN) — Explained,» Towards Data Science, 2020. [En línea]. Available: https://towardsdatascience.com/k-nearest-neighbors-knn-explained-cbc31849a7e3. [Último acceso: 19 noviembre 2021].