Lab 5 Textones

Nathan Mateo Marín
Universidad Los Andes
nm.marin10@uiandes.edu.co

Abstract

En este laboratorio se desarrolla un algoritmo capaz de clasificar imágenes por su textura. El algoritmo es entrenado y probado con la base de datos de 25 texturas diferentes de "The ponce group" [1]. Se desarrollan dos algoritmos distintos de clasificación uno de ellos es un bosque aleatorio y el otro clasifica usando el kernel de la intersección para determinar la distancia entre histogramas. Cada caso se ejecuta con diversos valores de k (El número de textones), de tamaño de las imágenes observadas (Se deben recortar las imágenes pues si se toma su resolución original de 480 X 640 el disco duro no tiene suficiente espacio para guardar todos los datos), de número de arboles por bosque y de la profundidad de cada árbol. Final mente se llega a la conclusión de que el método de bosque aleatorio es significativamente mejor para este problema que el método de clasificar eligiendo la clase de histograma más cercano al histograma de la imagen a clasificar. En promedio los bosques tienen un ACA (Average Classification Accuracy) de 60% *mientras que la clasificación por vecino más cercano tiene un ACA promedio de* 40%*. El método óptimo obtenido es un bosque aleatorio con 100 árboles de profundidad 100, con 175 textones e imagenes de 80 X 80, el ACA del método es* 64.8%*.*

1. Introducción

Lo primero es mencionar que el código y todos sus resultados se encuentran en https://github.com/mateomarin97/IBIO4680/tree/master/05-Textons/Answer , ahí usted encontrará una carpeta que dice código en la cual esta el código usado para entrenar y probar cada uno de estos clasificadores, dicho código llama clasificadormultiple.py. También encontrará un archivo con el diccionario de los textones obtenidos para el caso óptimo así como la matriz de confusión de ese caso. Dentro de esta carpeta también se encuentra la base de datos de entrenamiento en la carpeta Data (Los datos están debidamente segmentados en Training y Test, así como por categorías de texturas).

En la carpeta de Answer hay una carpeta más llamada Resultados donde se encuentran guardados en archivos .dat los ACA de cada algoritmo probado, en la carpeta matrices de confusion se encuentran las matrices de confusión del Training y el Test, en la carpeta textones y tiempos se encuentran consignados los diccionarios de textones obtenidos en cada caso y el tiempo en segundos que tomo armar cada uno de ellos.

Para acabar con las formalidades sólo basta mencionar las características de la base de datos. Ella tiene 25 categorías diferentes de Texturas, cada categoría tiene 40 imágenes de resolución 480 X 640, las primeras 30 de dichas imágenes constituyen el grupo de Training de la textura, mientras que las 10 últimas forman el grupo del Test.

2. Métodos

Empecemos con una breve explicación de como clasificar imágenes usando textones. Lo primero que se hace es convolucionar la imagen con múltiples filtros de derivadas (en este caso 12 orientaciones y 2 diferentes tamaños) el resultado de cada convolución se guarda. Luego para un píxel dado de la imagen tenemos 24 valores que lo representan en el espacio de las derivadas (uno por cada filtro), así colocamos en este espacio cada uno de los píxeles estudiados de las imágenes del Training y utilizamos el método de K-means para obtener una partición, que minimiza el desorden, de dicho espacio en K regiones. El centroide de cada región es llamado texton y son guardados para poder clasificar, dichos textones dan una idea del tipo de patrones que hay en una región de la imagen y así una distribución de ellos da una idea de la textura con la que se esta lidiando. Una vez se tiene el diccionario de Textones se asigna a cada píxel su texton más cercano y sobre una imagen se hace un conteo de Textones para crear el histograma de Textones de la imagen. Hasta aquí el método del vecino más cercano y del bosque aleatorio confluyen pero ahora para cada caso se toma un camino diferente.

2.1. Vecino más cercano

En este caso para cada categoría tomamos los 30 histogramas del Training y los ponderamos, este histograma

final será el histograma que representa la categoría. Luego sacamos para la imagen a clasificar su histograma y utilizando el kernel de la intersección vemos a cual de los 25 histogramas de categoría se parece más, a la categoría a la que se parezca más es a la que se asigna la imagen.

2.2. Bosque aleatorio

Entrenamos el bosque con los 30*25 histogramas del Training y los labels correspondientes de la clase de textura a la que pertenece cada uno. Una vez el bosque está entrenado simplemente se le pasa el histograma de textones de la imagen a clasificar y el la clasifica. Este método puede ser mejorado posteriormente al añadir al bosque las dimensiones de histogramas de color de las imágenes, sin embargo esto no se hace aquí.

2.3. Evaluación

Para evaluar todos los métodos se calcula la matriz de confusión de los datos del Test, luego se normalizan sus columnas y se promedia la diagonal de la matriz de confusión para obtener el ACA. Así un ACA de 0 implica que no se clasificó un sólo elemento bien mientras que uno de 1 implica que todos los elementos fueron clasificados correctamente.

2.4. Observaciones

Ya que los métodos en general han sido explicados es hora de mencionar un par de detalles del algoritmo. Como se habrá dado cuenta para crear el diccionario de Textones debemos especificar el número de ellos que queremos, no existe forma de determinar a priori cual número genere una mejor clasificación razón por la cual se deben probar múltiples opciones y con el desempeño del método determinar la óptima. Uno podría creer que entre más textones se use mejor funcionará el algoritmo pero lo que suele suceder es que al colocar un número ridículamente grande de Textones se está haciendo un overfit y además entre más Textones tenga más tiempo tomará calcular el diccionario.

Otro valor de interés es el número de píxeles que se van a tomar de las imágenes, idealmente uno querría tener toda la imagen para maximizar la información que tiene el algoritmo para entrenarse, empero esto es imposible con las maquinas a las que tengo acceso pues no tienen la suficiente memoria para almacenar tanta información. Así pues, se deciden tratar dos casos, uno con imágenes de 78X78 y otro con imágenes de 88x88. La región que se toma de las imágenes es siempre la del centro.

Los últimos dos hiperparametros a optimizar competen sólo a los bosques aleatorios y son el número de árboles por bosque y la profundidad máxima de cada árbol. El número de árboles por bosque la verdad no requiere mayor explicación. La profundidad máxima de los árboles hace que el algoritmo cambie significativamente pues pone un límite a la cantidad de divisiones que un árbol puede hacer al espacio de trabajo para clasificar, usualmente si la profundidad es muy baja se tiene un caso subfiting y sí la profundidad es muy alta empieza a haber overfiting y el árbol no está aprendiendo a clasificar sino que se está memorizando los datos del Training.

Una última observación, todas las imágenes fueron convertidas a escalas de grises para cerciorarse de que las derivadas dependan sólo de la luminosidad y no de los colores.

2.5. Resultados

Al analizar los valores de ACA arrojados por los diversos algoritmos se llega a que el algoritmo con el mayor ACA es un bosque aleatorio con 100 árboles de profundidad 100, con 175 textones e imágenes de 80 X 80, el ACA del método es 64.8%. Entonces, un podría apresurarse a decir que estos son los parámetros óptimos pero hay que tener en cuenta que al ser estos bosques aleatorios es posible que parte del éxito del método se deba a la forma aleatoria en que se construyo el bosque en esa ocasión y no tanto a los parámetros en sí. Para poder analizar los comportamientos de estos parámetros veamos las siguiente gráficas.

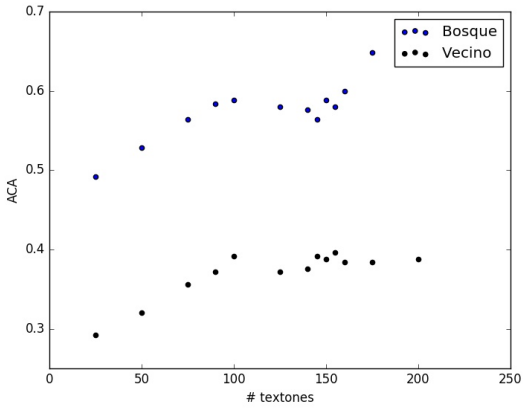

Figure 1. En este gráfico la resolución de las imágenes es de 78 X 78, la profundidad máxima de los árboles es 100 y el número de árboles en un bosque es 100 también. Se compara el desempeño del método de bosques aleatorios con el del vecino más cercano

De Figure 1 se puede concluir que: Primero, el método de bosques aleatorios es significativamente mejor al del vecino más cercano. Segundo, el desempeño de los algoritmos aumenta con el número de textones al menos hasta el límite de 200 textones.

De Figure 2 se puede concluir al contar cual de las dos resoluciones se comporta mejor en promedio que tener una resolución de 88 X 88 es mejor que tener una de 78 X 78, aunque en ocasiones la diferencia es nimia.

De Figure 3 se observa que hay un tipo de valle, uno podría creer que esto implica que entre 50 y 125 arboles el

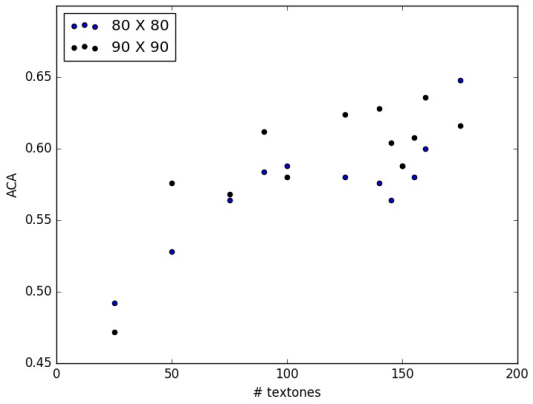

Figure 2. En este gráfico la profundidad máxima de los árboles es 100 y el número de árboles en un bosque es 100 también. Se compara el desempeño del método de bosques aleatorios con imágenes de resolución 78 X 78 y con imágenes de 88 X 88.

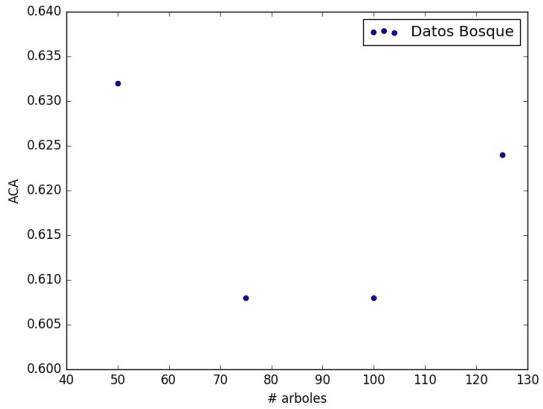

Figure 3. En este gráfico la profundidad máxima de los árboles es 100, el número de textones 155 y la resolución de las imágenes 88 X 88. Se compara el desempeño del método de bosques aleatorios a diferentes números de arboles por bosques.

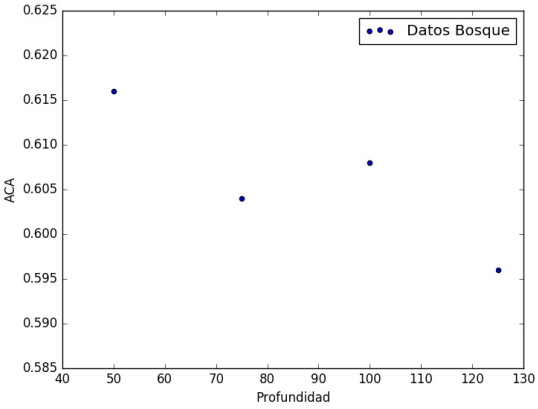

Figure 4. En este gráfico el número de árboles es 100, el número de textones 155 y la resolución de las imágenes 88 X 88. Se compara el desempeño del método de bosques aleatorios a diferentes profundidades máximas.

Table 1. Tiempo t en minutos que toma crear un diccionario con k Textones, al trabajar con imágenes de resolución 88 X 88.

t (min)	k
5.6	25
9.9	50
14.6	75
17.3	90
18.9	100
23.7	125
26.4	140
27.8	145
28.7	150
28.6	155
29.7	160
34.6	175

algoritmo se comporta ineficientemente pero la verdad esto se debe más que todo a la forma en que se estructuran los arboles en cada caso y no al número de arboles en sí. Aún así, elegir 125 arboles por bosque parece una buena opción.

De Figure 4 se concluye que una menor profundidad da mejores resultados, lo cual sugiere que al pasar de unos 100 arboles se está haciendo un overfit.

Tras hacer estas observaciones se concluye que el algoritmo óptimo (uno que posiblemente puede superar al mejor encontrado en este laboratorio) se debe buscar exhaustivamente con k=200, resolución de 88 X 88, 125 arboles por bosque y una profundidad máxima de 75.

De Table 1 se observan unas cuantas cosas curiosas. Primero, el tiempo de formar el diccionario y el número de textones tienen una relación lineal. Segundo, si observa los casos k=150,155 verá que se demoró más tiempo hacer el diccionario de 150 textones que el de 155, esto se debe a que el tiempo de ejecución de k-means también depende de donde se inicialicen los centroides, este efecto aleatorio hace que los tiempos puedan fluctuar un poco. Por último si comparamos los tiempos de generar el diccionario con el tiempo que le toma al método del vecino más cercano correr (33.8 minutos con 125 textones) vemos que son del mismo orden de magnitud, así que guardar los diccionarios de textones para no tener que calcularlos cada vez puede reducir significativamente el tiempo de ejecución.

La razón por la cual calcular los textones tarda tanto es porque estamos particionando en k un espacio de 24 dimensiones al obtener iterativamente k centros de masas de 88 X 88 X 30 X 25=5808000 vectores que en cada paso deben ser asignados a una partición temporal al observar a cual centroide esta más cerca cada uno. Si aumentamos la resolución de las imágenes el número de vectores va a aumen-

tar rápidamente y el tiempo de computo también al igual que la cantidad de memoria requerida para almacenar toda la información.

Por último se analiza la matriz de confusión del algoritmo óptimo. Tristemente esta matriz de 25 x 25 no cabe cómodamente en este informe así que para verla deberá ir al repositorio de Github en la carpeta de Code el nombre de la matriz es ConfuTest-bosque_k=175_Pixel=80n_estimators=100max_depth=100.dat.

Es interesante buscar las categorías que el algoritmo clasifica mejor y peor. En este orden de ideas, se descubre que hay una categoría en particular la cual el algoritmo clasifica a la perfección y es la textura del primer tipo de alfombra, lo cual es bastante impresionante al tener en cuenta lo mucho que las imágenes de esta categoría se parecen a las de la categoría del segundo tipo de alfombra. Es de esperarse, entonces, que el algoritmo también clasifique muy bien las imágenes del segundo tipo de alfombra, pero curiosamente sólo clasifica bien el 40% de estas imágenes.

Por otro lado, las imágenes peor clasificadas son las de las categorías de Granito, Agua y Plaid (es un tejido de estilo escoses), en estos casos sólo 2 de las 10 imágenes se clasificaron en la categoría adecuada. De las 10 imágenes de Agua 4 fueron clasificadas como pelo, de las 10 imágenes de granito 5 fueron clasificadas como el segundo tipo de corteza, de las 10 imágenes de Plaid 5 fueron clasificadas como pelo. Es muy curioso que muchas de estas imágenes mal clasificadas se clasificaran como pelo, ya que el pelo parece ser clasificado muy bien, 9 de las 10 imágenes de pelo se clasificaron apropiadamente.

Que el granito y el segundo tipo de corteza sean fácilmente confundidos tiene sentido pues de hecho se parecen bastante, a excepción de unos bordes muy delgados que el granito tiene. De hecho las imágenes de pelo también tienen bordes delgados, es posible que si aplicamos filtros de derivadas más compactas podamos detectar mejor estos bordes y clasificar de forma más apropiada.

3. Conclusiones

De momento el algoritmo no es lo suficientemente confiable para clasificar las texturas, tan sólo lo hace de forma correcta el 65% de las veces. Hay varios pasos a seguir que se deben tomar para mejorarlo significativamente:

- Se debe hacer un histograma de colores en adición al histograma de texturas para tener en cuenta la información del color a la hora de clasificar.
- Se deben aplicar a las imágenes filtros de derivadas más compactas para detectar apropiadamente los bordes delgados.
- Una vez se tengan estas dos modificaciones el algoritmo a entrenar debe ser un bosque aleatorio con unos 200 textones, 125 arboles, una profundidad máxima de 75 y la mayor resolución posible que la maquina pueda soportar (unos 90 x 90 píxeles).

Una vez se tenga el algoritmo es necesario guardar el diccionario de textones así como el bosque de clasificación para no tener que perder tiempo calculándolos de nuevo.

References

[1] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, Aug. 2005.