Características por textura

José María Campo Viñas
Universidad de los Andes
Código: 201412002
jm.campo11@uniandes.edu.co

Mariana Franky
Universidad de los Andes
Código: 201313944
m.franky10@uniandes.edu.co

Abstract

Los textones son patrones locales que se repiten aproximadamente en una imagen.En este laboratorio se utilizó los textones como estrategía para representar imágenes. Creando asi, un diccionario de textones. Para clasificar la imagen basandose en su representación de textones, se utilizaron como clasificadores Random forest y Nearest neighbour,como método de evaluación del clasificador se utilizó una matriz de confusión y se determinó que el clasificador Nearest neighbour fue el mejor método para clasificar imágenes a partir de textones.

1.. Introducción

Una representación de textones nos indica aquellos patrones repetitivos en una sección local de la imagen, al tener en cuenta la variación de textura se puede modelar la orientación de una superficie, y hasta se puede generar una nueva imagen partiendo del patron actual sin tener que repetirlo.
El estudio de textones tiene implicaciones importantes en una serie de problemas. En primer lugar, la descomposición de una imagen en sus componentes constituyentes reduce la redundancia de información y conduce así a mejores algoritmos de codificación de imágenes. En segundo lugar, la representación de imagen descompuesta tiene a menudo dimensiones reducidas y menor dependencia entre variables (coeficientes), por lo que facilita el modelado de imagen que son necesarios para la segmentación y reconocimiento de imágenes. [1] El objetivo de este laboratorio es clasificar imágenes a partir de características de su textura. Inicialmente, se creó un diccionario de textones basados en la imagenes de entrenamiento, luego se entrenaron dos clasificadores Radom forest y Nearest neighbour, para la validación se utilizó la funcion de matlab predict en los dos modelos y para comprabar la eficacia de los clasificadores se utilizó una matriz de confusion y su ACA.

2.. Métodos y materiales

2.1.. Materiales

Para este laboratorio, se usó la base de datos del grupo ponce, grupo de investigación en visión y robótica del Instituto de Beckman de la universidad de Illinois. Este base de datos contiene 1000 imágenes de distintas representaciones de texturas desde madera, vidrio, tejidos, agua entre otras. Estas 1000 imágenes se dividen en dos grupos : entrenamiento y validación. En entrenamiento se encuentran 750 imágenes repartidas en 25 tipos de textura con 30 imágenes cada una. En validación, hay 250 imágenes repartidas en 25 tipos de textura con 10 imágenes cada una.

Inicialmente se propusieron dos tipos de submuestreo para la base de datos :

- *Recortar cada imagen de la base de datos:*Esta se pensó con el objetivo de reducir el tiempo de procesamiento del algoritmo , y además, eliminar detalles redundantes dentro de cada imagen.
- *Escoger aleatoriamente de cada categoria una muestra significativa de imágenes:* Este se pensó con el fin de obtener una muestra pequeña de cada categoría pero sin reducir la información de la imagen.

Finalmente, para el procesamiento se escogió el segundo submuestreo, *Escoger aleatoriamente de cada categoria una muestra significativa de imágenes*, se escogió 2 imágenes aleatoriamente por cada categoría.

2.2.. Métodos

Ahora bien, el objetivo principal de este laboratorio es poder clasificar un grupo de imágenes a partir de sus textones. Para clasificar una imagen utilizando textones: la metodoliga se divide en dos la representación de una imagen y clasificación.
Para la etapa de representación de una imagen, inicialmente, se realiza un diccionario, luego es necesario hallar

para cada pixel el centroide más cercano y asignarle su etiqueta, luego se realiza un histograma para cada imagen. Los histogramas resultantes para cada imagen van hacer los datos de entrada para los clasificadores.
Después de la etapa de representación de una imagen , comienza la etapa de clasificación para esto se utilizan clasificadores como Random Forest y Nearest neighbour, estos clasificadores tienen dos etapas una de entrenamiento y otra de validación . La etapa de entranamiento es aquella que sirve para entrenar al modelo , se da como parámetro de entrada las características y las etiquetas, como parámetro de salida me da una estructura con valores que definen las distintas categorías. En la etapa de validación, es la ultima etapa y sirve para clasificar una imagen de acuerdo a sus características de textones. Como parámetro de entrada , esta el histograma de textones de la imagen. Como parámetro de salida arrojara la etiqueta correspondiente a la imagen.

1. Caracterización de la imagen.
   Para la creación del diccionario se utilizó como código base el que se encuentra en la guía. Este cuenta con distintas funciones, se describirán cada una de ellas a continuación :

   - *fbCreate:* Esta función crea un banco de filtros, con distintos orientaciones , numero de escalas , escalamiento y elongación. Como parámetro de salida da un arreglo de celdas , donde las filas son el tipo de filtro y en las columnas el mismo filtro a diferentes escalas.
   - *fbRun:* Esta función corre el filtro de bancos en una imagen, como parámetro de entrada recibe el banco filtros y la imagen y asi calcula la respuesta a cada tipo de filtro. Como parámetro de salida da las respuestas de cada uno de los filtros. Como parámetro de salida se obtiene un arreglo de celdas y cada celda contiene la respuesta de la imagen a un filtro en específico.
   - *computeTextons:*Esta función realiza k-means entre la respuesta de los filtros y numero de agrupaciones en que se quiere dividir la imagen. Como parámetros de salida muestra un mapa con las etiquetas que corresponden a cada pixel y muestra el diccionario de textones.
   - *assignTextons:* esta función calcula la distancia entre las respuestas de los filtros y el diccionario de textones, después encuentra los mínimos en cada fila. Las posiciones donde se encuentren los mínimos corresponderán a las etiquetas de nuestro diccionario de textones. Como parámetro de salida, sale una matriz del tamaño de la imagen en la cual para cada pixel se obtiene el valor del centroide al que se encuentra más cercano.

   Ya explicada cada función del código que se utilizó se procederá a explicar la creación de diccionario de textones: El diccionario de textones fue creado usando distintas texturas provenientes de imágenes con patrones definidos , se tomó una submuestra aleatoria de la base de datos. El algoritmo utilizado se basó en el código de la guía. Inicialmente se crea un banco de filtros, se lee todas las imágenes de la submuestra y luego estas se concatenan resultando una sola imagen. Esta imagen resultante, se sometió a distintos filtros de detección de bordes o contraste y se realizó la búsqueda de 32 centroides empleando el algoritmo de k-means sobre la respuesta de cada imagen a dichos filtros, obteniendo así un diccionario de textones global. Se utilizaron alrededor de 25 textones, esto se debe a la organización de la base de datos que está clasificada en 25 texturas.
   Cabe resaltar, que es necesario contar con filtros que sean más discriminativos que otros, esto se debo a que muchos filtros pueden servir para varias imágenes lo que genera que diferentes texturas puedan ser clasificadas como una sola. Al tener filtros discriminantes ayuda que se pueda diferenciar las texturas de otras. Ya teniendo el diccionario de textones global, a cada imagen de entrenamiento se le asignó los textones y se realizó el histograma a cada imagen, los cuales son los datos de entrada que reciben cada clasificador.

2. Etapa de clasificación.
   Como clasificadores se utilizaron Random Forest y Nearest neighbour, para entrenar , valiadar y llegar a la clasificación de imágenes apartir de sus textones.

   - *K-vecinos más cercanos:* se fundamenta en la clasificación con respecto a sus vecinos más cercanos, es decir, va a clasificar en la clase más frecuente a la que pertenecen sus K-vecinos más cercanos.vecindad son fundamentalmente dependientes de la distancia y en consecuencia poseen características propias de ésta como la cercanía, la lejanía y la magnitud de longitud, entre otras.
     básicamente consiste en comparar la nueva instancia a clasificar con los datos k más cercanos conocidos, y dependiendo del parecido entre los atributos el nuevo caso se ubicaráen la clase que más se acerque al valorde sus propios atributos.
     [2] Dentro de los hiperparametros se encuentra la distancia , esta puede ser : 'doublelogit', 'invlogit', 'ismax', 'none' or 'identity' , 'sign' ,'symmetric' ,'symmetriclogit', 'symmetri-

cismax'.También se encontró 'Exponent', 'NumNeighbors','Standardize'.
Cabe tener en cuenta , que en los métodos de vecindad el principal criterio de comparación es la distancia, es por esto que es necesario mencionar algunas de las diferentes métricas usadas: distancia euclidiana , distacia de Chebychev, distancia del coseno .[3]

La función que se utilizó para entrenar los Nearest neighbour es fitcknn, la cual devuelve un modelo de clasificación k -nearest vecino basado en las variables predictoras en la tabla Tbl y la matriz de respuesta Y. *Mdl = fitcknn(Tbl , Y).*
[3] Implementada en el código : *KNNModel=fitcknn(TrainMat,Anot)*; TrainMat : parámetro que corresponde a los histogramas de cada imagen de entrenamiento. Anot: son las etiquetas referentes a cada imagen. La métrica euclidiana fue la que se utilizó ya que es la que trae por defecto.

- *Bosques aleatorios:*son una combinación de predictores de árboles de tal manera que cada árbol depende de los valores de un vector aleatorio muestreado independientemente y con la misma distribución para todos los árboles en el bosque. [4]Es una modificación sustancial de bagging que construye una larga colección de árboles no correlacionados y luego los promedia.
[5] La función que se utilizó para entrenar Random Forest es TreeBagger , la cual devuelve un modelo de clasificación por arboles. Se utilizó la función : *B = TreeBagger(NumTrees,Tbl,ResponseVarName).*
Los parámetros que se pueden utilizar en esta función son :
  - *Numtrees:* determina de cuántos arboles va hacer la funcion ;
  - *Tbl* los datos de entrada a los que le asigno unas etiquetas.
  - *ResponseVarName:* Las etiquetas de los datos de Tbl. [5]

  Adicionalmente , la función permite otros parámetros tales como : 'InBagFraction','Cost', 'SampleWithReplacement', 'OOBPrediction', 'OOBPredictorImportance', 'Method', 'NumPredictorsToSample', 'NumPrint','MinLeafSize','Options', 'Prior', 'CategoricalPredictors'. [5]

  Función implementada en el código : *'TREEModel=TreeBagger(CantTree,TrainMat,Anot,'Method', 'Classification')'*;
  La cantidad de arboles que se utilizaron fueron 7 , este parámetro se escogió aleatoriamente.
  TrainMat: es el parámetro que corresponde a los histogramas de cada imágen de entrenamiento.
  Anot: es el parámetro que corresponde a las etiquetas de cada imagen.
  Method: corresponde al método que quiero utilizar clasificación o regresión, en este caso se utilizó como clasificador.

Una vez realizada la parte de entrenamiento, se utilizó la función predit de Matlab para cada clasificador, la cual devuelve un vector de respuestas predichas para los datos del predictor en la tabla o matriz X . Para comprobar la eficiencia del algoritmo se utilizó una matriz de confusión por cada clasificar y se calculó precisión de clasificación promedio (ACA).

3.. Resultados

En las matrices de confusión de ambos métodos se observa que existe una clasificación precisa para los materiales que poseen cualidades similares entre sí, como se puede denotar en la Figura 1, un alto puntaje en la clasificación por K-Nearest Neighbours (KNN) para los materiales de madera (wood) y corteza de árbol (bark), mientras que en la Figura 2, la mayor cantidad de clasificación por Random Forest (RF) de madera y corteza, se encuentra entre sus vecinos similares al ver un cuadro de alto puntaje en la esquina superior izquierda.

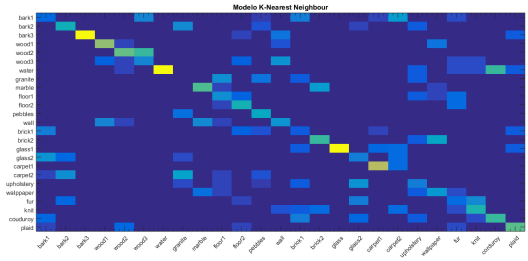

Figura 1. Matriz de confusión para modelo K-Nearest Neighbour.

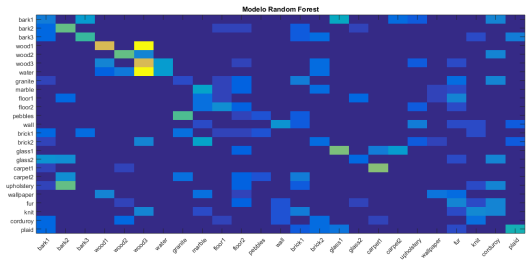

Figura 2. Matriz de confusión para modelo Random Forest.

La precisión de clasificación promedio (ACA) para KNN es 42.97 %, mientras que para RF es 29.79 %. Ésta métrica es la más común para probar la eficiencia del algoritmo en problemáticas basadas en detección [6].

4.. Discusión y Conclusiones

El clasificador más efectivo fue KNN. Probablemente se debe a la poca información suministrada a las matrices de entrenamiento; factores influenciadores como la ausencia de filtros Laplacianos, ruido y la no convergencia de centroides, pudieron afectar el entrenamiento del algoritmo. RF es un método de clasificación que requiere de un alto número de datos para separar binariamente las categorías, en este caso se tenía una baja proporción de datos u observaciones con respecto a la cantidad de categorías de imágenes.

El diccionario de textones en éste caso de estudio tomó aproximadamente dos horas para encontrar los centroides estimados, ya que estos no convergieron y alcanzaron el máximo de iteraciones por defecto. La razón por la cual el algoritmo es computacionalmente lento, se debe a la cantidad de datos con los cuales se computan las distancias respecto a los centroides en cada iteración.

Generar los modelos de clasificación y obtener sus predicciones tomó aproximadamente de 15 minutos de ejecución.

Las categorías de con menor acierto de clasificación fueron granito mostrado en la Figura 3 y ladrillo1 mostrado en la Figura 4.

La confusión de la clase de granito es dirigida hacia la clase tapicería como se muestra en la Figura 5, ambas imágenes poseen gran cantidad componentes de alta frecuencia, sin embargo, en el banco de filtros únicamente se tienen aquellos para detección de contornos (filtros Sobel) y no de contraste (filtros Laplaciano), por ende, los contrastes puntuales no serán tenidos en cuenta en el granito y sólo se tomará la información de alta frecuencia, la cual también existe en la tapicería, pero esta última posee un mayor peso al denotar los contornos horizontales presente por toda la imagen.

Por otra parte, la confusión de la clase ladrillo1 no es dirigida hacia otra clase en específico, pero sí se reparte en categorías con ruido de baja frecuencia por toda la imagen como en tapicería, a pesar de tener contornos prominentes en sentido diagonal, pero estos son confundidos con la respuesta a los filtros de la clase ladrillo2, la cual no posee ruido. Una de las limitaciones de nuestro método es que no hay una cantidad considerable de imágenes, dado que sólo se tomo 2 imágenes por cada categoría, es decir, equivale solo al 6 % de la base de datos de entrenamiento.

Para mejorar el método implementado en este laboratorio, se puede implementar un algoritmo que requiera menor procesamiento computacional que k-means y encuentre los centroides de la librería de textones de una manera más efectiva, así mismo, aumentar la variedad de filtros para que tengan en cuenta ruido de baja frecuencia y los contrastes de una región delimitada.

También se debe aumentar el número de imágenes por categoría y precisar en los aspectos fundamentales para un efectivo manejo por parte de la clasificación RF.

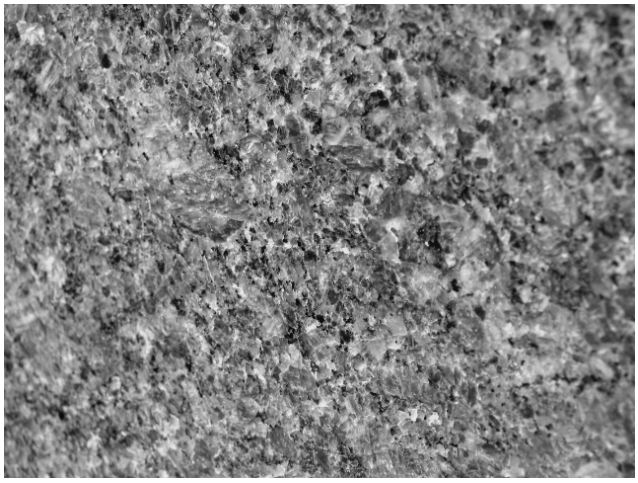

Figura 3. Muestra de textura granito.

Figura 4. Muestra de textura ladrillo1.

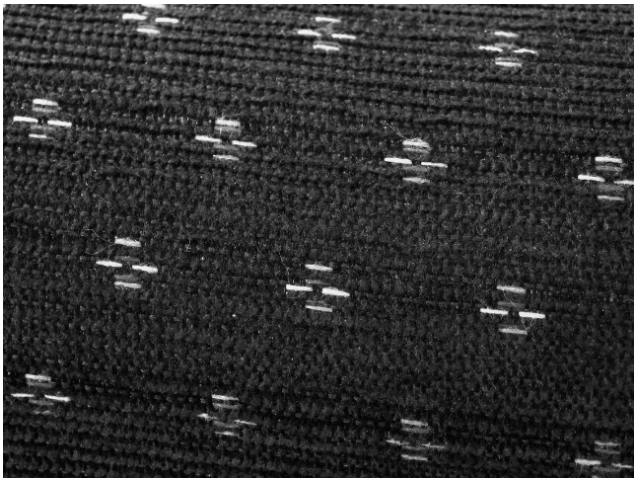

Figura 5. Muestra de textura tapicería.

Referencias

[1] S.-C. Zhu, C.-E. Guo, Y. Wang, and Z. Xu, "What are textons?," *International Journal of Computer Vision*, vol. 62, no. 1-2, pp. 121–143, 2005.

[2] J. Rodríguez, "Clasificación de datos usando el método k-nn," Master's thesis, Universidad Distrital Francisco José de Caldas, Diciembre 2008.

[3] MathWorks, "Classification knn class." `https://www.mathworks.com/help/stats/classificationknn-class.html`.

[4] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] MathWorks, "Treebagger." `https://www.mathworks.com/help/stats/treebagger.html`.

[6] P. Arbeláez, "Datasets." `https://sicuaplus.uniandes.edu.co/bbcswebdav/pid-1757212-dt-content-rid-18138914_1/courses/201710_IBIO4680_01/01b_datasets%281%29.pdf`, 2016.