Clasificadores de Texturas

Flores-Silva P.

Abstract—En el presente trabajo se obtuvieron las características que definen a la textura de una imagen a partir del cálculo de la matriz de coocurrencia, para múltiples imágenes, por dos métodos: empleando ventanas cuadradas que recorren la imagen y superpíxeles. A partir de los datos obtenidos para cada método se entrenaron cuatro clasificadores, a saber: Bayesiano ingenuo, Soporte de vectores, k vecinos cercanos y k medias, con el fin de clasificar una imagen de entrada y construir una imagen de clasificación de acuerdo a las clases predichas. Para el caso de las ventanas cuadradas se encontró que los clasificadores Soporte de vectores, k vecinos cercanos ofrecen un rendimiento de 100% al considerar únicamente las características de Homogeneidad, ASM, Energía y Correlación, mientras que los "mejores" resultados obtenidos por el clasificador Bayesiano ingenuo se encuentran al considerar todas las características siendo su rendimiento bajo. Finalmente, al considerar superpíxeles se obtuvieron resultados similares que preservan los bordes de la imagen de prueba al considerar al menos nueve clases para el caso de los clasificadores Soporte de vectores, k vecinos cercanos. El clasificador k-medias se ajusto a tres clusters obteniéndose una imagen de validación que no preserva algunos bordes y bajo rendimiento, sin embargo, esto se debe a los cambios notables de los niveles de gris en las imágenes de entrenamiento y prueba.

Index Terms—Visión computacional, Imágenes, Aprendizaje de Maquina, Texturas, Superpíxeles, Clasificador Bayesiano Ingenuo, Clasificador Soporte de Vectores, Clasificador k vecinos más cercanos, Clasificador k-medias.

1 Introducción

Con la creciente industria social electrónica, tales como facebook, TikTok, foros en internet entre otros, en años recientes, la cantidad de datos generados en esta industria crece de manera acelerada cada día. El creciente poder computacional y el avance de las ciencias computacionales permite manejar todo tipo de datos en tiempos aceptables para una gran cantidad de los mismos. La modernización de la industria tiene como objetivo la automatización adquisición de datos de forma electrónica, esto debido a que las nuevas tecnologías proveen métodos que permiten el fácil manejo y análisis de los datos.

La importancia de los datos puede resumirse en que gracias a ellos es posible extraer conocimiento o la deducción de ciertos eventos. Existen muchos métodos para realizar dicha tarea, entre ellos se encuentra la estadística inferencial.

El teorema de Bayes es uno de los grandes pilares de la estadística inferencial, es también, importante en una gran variedad de modelos de aprendizaje de maquina. El razonamiento de Bayesiano es un enfoque lógico para actualizar la probabilidad de una hipótesis o evento a la luz de nueva evidencia (como los datos) y, por lo tanto, juega un papel fundamental en la ciencia [1].

La mayoría de las veces los datos que se desean analizar requieren un preprocesamiento y/o análisis para poder realizar una clasificación. En el caso de las imágenes, para obtener un clasificador es necesario caracterizar las imágenes. La caracterización de una imagen puede hacerse a través de una gran cantidad de métodos, que proveen diferentes atributos de la imagen. En el presente trabajo nos enfocaremos en el método estadístico de segundo orden para caracterizar las texturas de una cantidad de imágenes. Mediante las cuales será posible construir tres clasificadores y posteriormente evaluarlos.

• *P. Flores-Silva estudiante del PCIC en la UNAM. E-mail: flosipan@ciecias.unam.mx*

1.1 Textura

La textura es un concepto vago, regularmente atribuido a la percepción humana. Cada persona tiene su propia interpretación de la naturaleza de la textura. No existe una definición matemática de la textura [1]. Por lo que si nos referimos al diccionario *Oxford Concise English Dictionary* encontramos una definición de textura como (traducida del ingles) *Textura: sust. disposición de hilos etc. en tejido textil. sensación característica debido a esto; disposición de pequeñas partes constituyentes, estructura percibida (de piel, roca, tierra, tejido orgánico, obra literaria, etc.); representación de estructura y detalle de objetos en el arte*. Si cambiamos hilos por píxeles, la definición podría aplicarse a imágenes. Esencialmente, la textura puede ser lo que nosotros mismos definimos como puede ser (*texture can be what we define it to be*) [1]. Como definicion podemos considerar a la textura como definida por la base de datos de imágenes que los investigadores utilizan para probar sus algoritmos. Muchos investigadores de texturas han utilizado una base de datos de imágenes de texturas producidas por artistas y diseñadores [1], un ejemplo de ello es la base de datos de Brodatz [2]. De una manera alternativa se define la textura como una cantidad para la cual los algoritmos de extracción de texturas proporcionan resultados significativos. Un extracto interesante de Karru [3] postula: *La respuesta a la pregunta ¿Hay alguna textura en la imagen? depende no solamente de la imagen estudiada, si no que también del objetivo para el que se utiliza la textura de la imagen y los rasgos de textura que se extraen de la misma.*.

Debido a que la textura no tiene una definición única, existen muchas formas de describirla, estas formas pueden dividirse en básicamente tres grandes ramas: *Estructurales*, *Estadísticas*, *Combinaciones*. Como ya se mencionó, este trabajo se centrará únicamente en uno los métodos estadísticos.

1.1.1 Enfoque Estadístico: Matriz de Coocurrencia

El enfoque estadístico más famoso es la *matriz de coocurrencia* o GLCM por sus siglas en inglés (Gray Level Coocurrence Matrix). Este enfoque fue el primer método para describir y clasificar la textura de imágenes propuesto por Haralick [1]. La matriz de coocurrencia contiene elementos que son cuentas del número de pares de píxeles para niveles específicos de gris cuando están separados por alguna distancia y con alguna inclinación relativa. Para dos niveles de gris b_1 y b_2, la matriz de coocurrencia C se define como:

$$C_{b_1,b_2} = \sum_{x=1}^{N}\sum_{y=1}^{N}(P_{x,y} = b_1) \wedge (P_{x',y'} = b_2) \quad (1)$$

donde $P_{x,y}$ es la entrada en la posición (x, y). x', y' es el offset definido por la distancia d y la inclinación θ dada por:

$$x' = x + dcos(\theta) \quad (2)$$
$$y' = y + dsen(\theta) \quad (3)$$

donde $\theta \in (0, 2\pi)$.
Cuando se aplica la ecuación (1) a una imagen, se obtiene una matriz cuadrada, simétrica, cuyas dimensiones corresponden a la cantidad de niveles de gris en la imagen.
Esencialmente, la matriz de coocurencia mide las relaciones espaciales entre los niveles de gris. Para generar resultados de rápido cálculo, comúnmente, se reduce el número de los niveles de gris a través del escalado del brillo de toda la imagen, reduciendo así las dimensiones de la matriz de coocurrencia, pero esto reduce la habilidad discriminatoria [1].
Para describir las propiedades de la textura Haralick y sus colaboradores definieron un conjunto de 28 características que pueden obtenerse a partir de la matriz de coocurrencia. Posteriormente, Conners y Harlow encontraron que de esas 28 características solo cinco de ellas son suficientes para describir la textura [4]. Éstas cinco son:

- Energía: provee información de la homogeneidad de la imagen; tiene valores pequeños cuando los pares de niveles de gris son bastante similares y valores altos en caso contrario. Definida matemáticamente como
$$\sum_{i=1}^{G}\sum_{j=1}^{G}P(i,j|d,\theta)^2$$
donde esta notación es análoga con (1) y G son los niveles de gris.
- Entropía: mide el desorden de la GLCM. Definida matemáticamente como
$$-\sum_{i=1}^{G}\sum_{j=1}^{G}P(i,j|d,\theta)log_2(P(i,j|d,\theta))$$
- Correlación: mide la dependencia lineal del nivel de gris entre píxeles (relativo a cada uno de ellos) en las posiciones especificadas; tiene valores altos cuando los valores están distribuidos uniformemente en el GLCM y valores bajos en caso contrario. Definida matemáticamente como
$$\sum_{i=1}^{G}\sum_{j=1}^{G}P(i,j|d,\theta)\left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{\sigma_i^2\sigma_j^2}}\right]$$
- Homogeneidad local o momento de diferencia inversa: Es alto cuando se encuentra los mismos pares de píxeles. Definida matemáticamente como
$$\sum_{i=1}^{G}\sum_{j=1}^{G}\frac{P(i,j|d,\theta)}{1+(i-j)^2}$$
- Inercia o Contraste: Mide las variaciones locales presentes en la imagen. Definida matemáticamente como
$$\sum_{i=1}^{G}\sum_{j=1}^{G}(i-j)^2P(i,j|d,\theta)$$

Una de las grandes desventajas del enfoque GLCM es la alta dimensionalidad de la matriz y la alta correlación de las características de Haralick.
El método GLCM ofrece resultados buenos para situaciones simples donde las texturas son visualmente separables de una manera sencilla. Además, el algoritmo GLCM es fácil de implementar y ha sido empleado en una gran variedad de aplicaciones con buenos resultados. Sin embargo, debido a su gran dimensionalidad, la GLCM es muy sensible al tamaño de las muestras de textura que son procesadas [4].

1.2 Clasificadores

En general, dado un conjunto de datos donde cada dato corresponde a una clase conocida. Entonces, se quiere predecir la clase de un nuevo punto con base en las clasificaciones conocidas de las observaciones en la base de datos. Por esta razón, la base de datos se conoce como conjunto de Entrenamiento. El proceso de elegir la clasificación de una nueva observación es conocido como un problema de clasificación y existen múltiples formas de tratar este problema. Consideramos la elección de la clasificación de la nueva observación basada en las clasificaciones de las observaciones en la base de datos a las que es "más similar". Sin embargo, decidir si dos observaciones son similares o no es una pregunta abierta. Por lo que antes de podamos decidir si dos objetos son similares, es necesario encontrar una forma de comparar objetos. El principal problema de esto es que nuestra base de datos puede ser de diferentes tipos o formas. De esta manera siempre será necesario transformar los datos de tal forma que se obtengan formas de medir la similitud entre objetos de la misma base de datos. De esta forma se esta creando un conjunto de características definido en un espacio representativo cuya dimensión dependerá de las mismas características. Algunos algoritmos de clasificación usan métricas apropiadas en este espacio para definir la distancia entre el conjunto de características como una medida de similitud y a partir de la misma medida determinar la similitud de un nuevo dato con respecto a las clases de nuestra base de datos.

1.2.1 Clasificador k vecinos más cercanos (kNN)

El algoritmo kNN considera que cada una de las características de nuestro conjunto de entrenamiento corresponde a una dimensión en algún espacio métrico, y toma el valor que tiene una observación para que esta característica sea su coordenada en dicha dimensión, obteniendo así un conjunto de puntos en el espacio. Entonces es posible considerar que la similitud de dos puntos es la distancia entre ellos en este

espacio bajo alguna métrica apropiada.
La forma en que el algoritmo decide cuáles de los puntos del conjunto de entrenamiento son lo suficientemente similares como para ser considerados al elegir la clase para predecir un nuevo dato es elegir los puntos de datos k más cercanos a la nueva observación, y tomar la clase más común entre ellos. Por eso se llama el algoritmo de k vecinos más cercanos.
El algoritmo kNN se resume como sigue [5]:

- Se especifica un entero positivo k y una nueva muestra o dato
- Selecciona las k entradas en la base de datos que están más cerca de la nueva muestra
- Encuentra la clasificación más común de estas entradas
- Clasifica al nuevo dato como el valor encontrado en el punto anterior

1.2.2 Clasificador k medias (k-Means)

El algoritmo de k medias se basa en el concepto de clusterizacion o clustering, que consiste en la organización de datos no etiquetados dentro de grupos de similitud llamados clusters. Un cluster es una colección de objetos que son similares entre sí. La medida de similitud usada por este algoritmo es también la distancia en el espacio definido por las características. A diferencia de los k vecinos más cercanos el algoritmo de k-medias depende de los centroides de los k clusters que el usuario debe definir de acuerdo a las base de datos disponible. Un conjunto de datos, que puede ser vacío, pertenecerá a un cluster j si y solo sí la distancia del punto al centroide del cluster j es la mínima distancia entre el punto y otro centroide de un cluster diferente a j. Inicialmente los centroides de los k clusters se colocan en posiciones aleatorias del espacio de características y se procede a calcular la pertenencia de cada dato a cada centroide. Una vez finalizado, los centroides se actualizan usando la media de las posiciones de sus elementos. Este proceso se realiza de forma iterativa hasta cumplir un cierto criterio de paro. Los tres criterios de paro son:

- No o mínima reasignación de los puntos (datos) a diferentes clusters
- No o mínimo cambio en los centroides
- Decrecimiento mínimo de la función de suma de error cuadrático medio (SSE)

La SSE se define como:

$$SSE = \sum_{j=1}^{k} \sum_{\vec{x} \in C_j} d(\vec{x}, \vec{m}_j)^2 \tag{4}$$

donde C_j es el j-ésimo cluster, $\vec{m}_j$ es el centroide del cluster C_j (la media de todas las posiciones de los puntos en C_j) y $d(\vec{x}, \vec{m}_j)$ es la distancia entre un punto $\vec{x}$ y el centroide $\vec{m}_j$. La métrica para la distancia debe ser elegida según los datos.
El algoritmo k-means se resume como sigue [6]:

- Se especifica un entero positivo k y se eligen aleatoriamente k puntos que serán los centroides
- Asigna cada punto al centroide más cercano
- Recalcula los centroides usando los elementos de cada cluster
- Si el criterio de paro no se cumple repite los últimos dos pasos

Para este caso la clasificación es mucho más sencilla. Dado un nuevo dato. Se calcula la distancia de este punto a los k centroides y se asigna la clase según la distancia mínima obtenida.

1.2.3 Máquinas de Vectores de Soporte (SVM)

Si se tienen L puntos de entrenamiento, donde cada entrada $\vec{x}_i$ tiene D atributos y pertenece a una de las dos clases $y_i = -1$ o $+1$ los datos de entrenamiento son de la forma:

$$\{\vec{x}_i, y_i\} \tag{5}$$

donde $i = 1, 2, ...L$, $y \in \{-1, 1\}$ y $x \in \mathbb{R}^D$.
Se asume que los datos son linealmente separables, es decir, es posible dibujar un hyperplano en las gráficas de $x_1, x_2, ..., x_D$ que separe las dos clases. Este hiperplano puede ser descrito por $\vec{w} \cdot \vec{x} + b = 0$. Donde $\vec{w}$ es el vector normal al hiperplano y $\frac{b}{||\vec{w}||}$ es la distancia perpendicular del hiperplano al origen.
Los Vectores de soporte son las muestras más cercanas al hiperplano de separación y el objetivo de las SVM es orientar este hiperplano de tal manera que esté lo más lejos posible de los miembros más cercanos de ambas clases. La figura 1 muestra muestra un ejemplo en dos dimensiones.

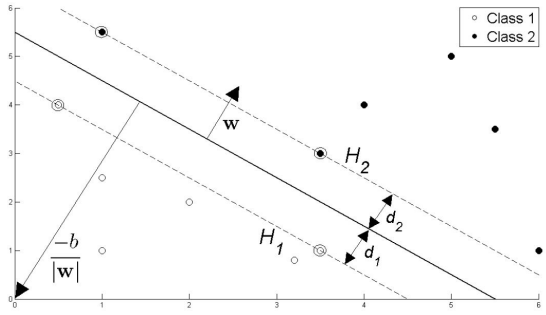

Fig. 1. Hiperplano divisor de dos clases linealmente separables. Tomada de [7]

La figura 1 prueba que implementar una SVM se reduce a encontrar las variables $\vec{w}$ y b tales que nuestro conjunto de entrenamiento pueda ser descrito por:

$$\vec{x_i} \cdot \vec{w} + b \geq +1 \quad si y_i = +1 \tag{6}$$
$$\vec{x_i} \cdot \vec{w} + b \leq -1 \quad si y_i = -1 \tag{7}$$

Las dos ecuaciones anteriores se reducen a:

$$y_i(\vec{x_i} \cdot \vec{w} + b) - 1 \geq 0 \tag{8}$$

Si se consideran los puntos más cercanos al hiperplano (vectores soporte, son los círculos en la figura 1). Entonces, los dos planos H_1 y H_2 y los puntos pueden ser descritos por las ecuaciones:

$$\vec{x_i} \cdot \vec{w} + b = +1 \quad H_1 \tag{9}$$
$$\vec{x_i} \cdot \vec{w} + b = -1 \quad H_2 \tag{10}$$

De la figura definimos d_1 y d_2 como la distancia del plano H_1 y H_2 al hiperplano separador, donde el hiperplano es equidistante a H_1 y H_2, es decir $d_1 = d_2$. A esta cantidad

se le conoce como margen SVM. Con el fin de orientar el hiperplano de tal forma que se encuentre los mas lejos posible de los vectores de soporte, es necesario maximizar el margen. De la figura, es fácil demostrar que el margen es igual a $\frac{1}{||\vec{w}||}$, por lo que la tarea consiste en maximizar este ultimo valor bajo la constricción dada por la ecuación (8) o equivalentemente, minimizar el valor de $||\vec{w}||$ bajo las condiciones de (8). A partir de la minimización de $||\vec{w}||$ es posible predecir el valor óptimo de b. La deducción de todas las ecuaciones se encuentra en [7]. En este trabajo no se incluyen ya que no es el objetivo.
Las SVM pueden extenderse a modo que se consideren más de dos clases. Esta extensión es muy natural; únicamente se ajusta un plano considerando la existencia de únicamente dos clases: Una clase separada contra el resto. Este proceso se sigue hasta haber analizado por separado cada clase.

1.3 Superpíxeles

Muchos de los algoritmos en visión computacional usan una malla simétrica o cuadrícula de píxeles como representación adyacente de una imagen. Por ejemplo, los modelos estocásticos de imágenes, tales como los campos aleatorios de Markov [8]. Regularmente la simetría de dichas mallas se ajusta a una simetría cuadrangular sin cambios en sus dimensiones.
Sin embargo, una cuadricula de imágenes no es una representación natural de las escenas del mundo real, es más bien un artefacto con el cual es posible procesar una imagen de una manera muy sencilla. Debido a esto se busca una forma más natural y eficiente para representar objetos significativos dentro de la imagen y éstos, al igual que con las cuadrículas, deben ser agrupadas mediante algún proceso que las identifique correctamente. A estos objetos representativos los llamamos superpíxeles.
De una manera más compacta definimos un superpíxel como un grupo de píxeles similares en color y otras propiedades de bajo nivel, donde este grupo es un conjunto de una sola partición [9]. Estos superpíxeles deben tener las siguientes propiedades [8], [9]:

- El cálculo de los superpíxeles debe ser computacionalmente eficiente de tal forma que reduzca la complejidad de las imágenes de cientos de miles de píxeles a solo unos cientos de superpixeles.
- Deben ser eficientes en cuanto a la representación de la imagen.
- Deben ser perceptualmente representativos: cada elemento de un superpíxel debe ser parecido con los otros elementos del mismo conjunto. Esta similitud puede ser por color, textura, entre otros.
- El conjunto de superpíxeles debe ser compacto,completo y disjunto, es decir, debido a que los superpíxeles son el resultado de la segmentación de una imagen, cada píxel de la imagen debe estar contenido en un solo elemento del conjunto. Todos los superpíxeles sin translaparse (este hecho es obvio al considerar la constricción anterior) deben cubrir toda la imagen.
- La cantidad de superpíxeles generados debe ser ajustable por el usuario.
- Adherencia de frontera: los superpíxeles deben preservar las fronteras o bordes de la imagen. La definicion de frontera dependerá de la imagen.
- Conectividad: Los superpíxeles deben representar conjuntos conectados de píxeles.

Existen al menos 28 algoritmos para obtener los superpíxeles de una imagen dada que pueden dividirse en siete diferentes categorías según la metodología que se emplee para obtener los resultados deseados [9]. De una manera breve se explicara el algoritmo Simple Linear Iterative Clustering (SLIC) perteneciente a la categoría basada en clusterizacion.

1.3.1 Algoritmo SLIC

El algoritmo SLIC se basa en la técnica de clusterizacion ya que está inspirado en los algoritmos como k-means. Debido a que este tipo de algoritmos es iterativo es necesario realizar un post-procesado de la imagen con el fin de reforzar la conectividad de los superpíxeles.
Este algoritmo genera superpíxeles al clusterizar píxeles basados en la similitud y proximidad del plano de la imagen. Esto se realiza en el espacio (l, a, b, x, y) definido por el vector de color (l, a, b) del espacio de color CIELAB y el vector de posiciones (x, y).
El espacio de color CIELAB es usado normalmente para describir todos los colores que percibe el ojo humano. Los tres parámetros representan la luminosidad (l), la posición entre valores del rojo y verde (a) y la posición entre el amarillo y el azul (b). El algoritmo puede resumirse de la siguiente manera [10]:

- Inicializa los centros de los clusters $C_k = (l_k, a_k, b_k, x_k, y_k)^T$ mediante el muestreo de píxeles en pasos regulares S de la cuadrícula
- Perturba los centros de los cluster en una vecindad de $n \times n$, a la posición del gradiente mínimo
- De manera iterativa hasta cumplir cierta condición de paro: Para cada centro de cluster C_k asigna los píxeles mas similares de una vecindad cuadrada de tamaño $2S \times 2S$ al rededor de el centro del cluster usando una métrica previamente definida. Y redefine los centros de los clusters según lo calculado anteriormente, calcula un error residual definido como la distancia entre los centros anteriormente calculados y los nuevos

2 PROCEDIMIENTO EXPERIMENTAL

El desarrollo experimental se divide en dos partes. La primera de ellas consistirá en obtener las cinco características que describen a la textura para cuatro imágenes diferentes. Dichas imágenes pertenecen a la base de datos de [2]. Las características se obtendrán a través del participando de la imagen, es decir recorriendo una ventana cuadrada que contenga los valores de la imagen en la posición en la que se encuentra la ventana y al obtener la GLCM de los valores contenidos en la misma, para así obtener las características de cada ventana. Al recorrer toda la imagen se obtendrá una pequeña base de datos de dimensiones $(L \times 6)$ para cada una de las imágenes. El valor de L dependerá de las dimensiones de la ventana. Los resultados

que se presentarán serán únicamente para dos tamaños de ventana obteniendo un valor de L igual a 25 y 100 respectivamente. Estos valores resultaron de la inspección visual de la red simétrica y periódica que representa cada imagen. Cabe mencionar que el tamaño de la ventana fue ajustada a priori según la información visual provista por la red de cada imagen y debido a que la resolución espacial de las imágenes era la misma, se decidió mantener el mismo tamaño de ventana para todas las imágenes, resultado del mismo análisis visual de las redes.
Una vez que se obtienen las cuatro bases de datos se agrega una columna a cada una de ellas para representar la clase, una para cada imagen. Posteriormente a partir de las cuatro bases de datos con la columna de clase se genera una base de datos general. Esto con el fin de entrenar tres clasificadores: Bayesiano ingenuo, SVM, y kNN.
Para validar los clasificadores se usó la imagen que se muestra en la figura 2. En esta imagen se tienen las cuatro texturas que se usaron para entrenar los clasificadores dispuestas cada una en un cuarto de la imagen de prueba. Para validar dicha imagen se sigue un procedimiento análogo al del entrenamiento: se recorre la imagen con una ventana de dimensiones determinadas, se obtiene la matriz de coocurrencia para los píxeles dentro de la misma y se determinan las características de textura. Estas propiedades se almacenan en un vector de dimensión (1×6) que para cada clasificador se predice la clase usando este vector, dependiendo de la clase asignada, en una matriz de validación se almacenan los valores de 0, 63, 127 o 255 aquellos píxeles que cubre la ventana. Dependiendo del desempeño del clasificador la imagen de validación será muy parecida a cuatro cuadrados de distinto color en el caso ideal.

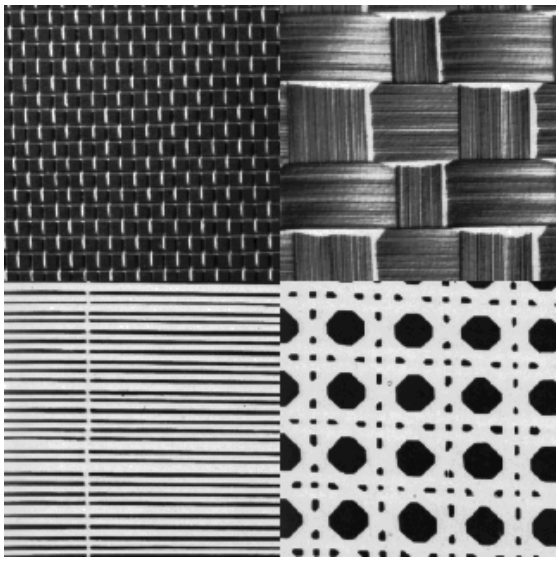

Fig. 2. Imagen empleada para la validación. Las cuatro texturas diferentes que se muestran son una porción de las cuatro imágenes empleadas para el entrenamiento

La segunda parte consiste en una sola imagen de entrenamiento que contenga diversas texturas y obtener los superpíxeles usando el algoritmo SLIC, donde la cantidad de éstos será ajustada según la divisibilidad de las texturas. Posteriormente se obtienen la matriz de coocurrencia y las características que se calculan a partir de la misma para cada superpíxel. Los datos obtenidos se guardan en una matriz de dimensiones $(N \times 6)$, donde N es la cantidad de superpíxeles. Notemos que cada superpíxel representará una zona de la imagen a una de las clases de interés, sin embargo, debido a la naturaleza de la imagen (fotografía de un paisaje con un objeto al frente), existen una gran cantidad de superpíxeles que comparten la misma clase pero se encuentran separados. Y debido a que el Algoritmo SLIC enumera de una forma no ordenada cada superpíxel es sumamente complicado encontrar un conjunto de conjuntos tales que cada uno de ellos contenga a los superpíxeles que representan a una clase, de esta forma tenemos no podemos asignar a cierto conjunto de superpixeles a una salida $y \in \{clases\}$. Es necesario conocer y para poder hacer clasificación. Para "resolver" este problema consideramos que existen tantas clases como superpixeles. Dado que sabemos que las clases son menores que los superpíxeles podemos asumir que si las clases no están fuertemente correlación entre sí y dado que los superpíxeles que comparten una clase en común estan altamente correlacionados, entonces de forma general un clasificador que considera la misma cantidad de clases como superpíxeles podrá diferenciar de manera aceptable entre las clases originales con pequeñas variaciones debido a la sobrepoblación de clases. En general se espera que al entrenar un clasificador con este método sea posible detectar algunas aristas de la imagen y que el color de las clases correlacionadas entre sí no varíe demasiado en la imagen de validación. Bajo esta premisa se van a entrenar y validar los clasificadores SVM y kNN para poder comparar los resultados con los superpíxeles y una imagen de prueba (claramente la imagen de prueba es similar a la imagen de entrenamiento). A partir de la validación se obtendrá una imagen de validación tal y como se describe en la primera parte, cambiando las ventanas por superpíxeles.
Un método más coherente para resolver el problema de la sobre población de clases consiste en usar el algoritmo de k-medias, dado que se conocen las clases que se desean entonces se procede a clusterizar los datos obtenidos por cada superpíxel, de esta manera tenemos un aproximado (ya que k-medias no garantiza una separación perfecta) del conjunto de datos que corresponden a cada una de las clases deseadas. Una vez que se realiza este método podemos predecir la clases para un conjunto de características de validación. Análogo a los otros dos procedimientos se obtendrá una imagen de validación empleando este método considerando los superpíxeles de la imagen de prueba.

3 Resultados y discusión

3.1 Ventanas cuadradas

Para la obtención de características de cada imagen de entrenamiento se desarrollo una función general que recibe como argumentos el tamaño deseado de la ventana y la imagen. A esta función se le llamó características, debido al tamaño del documento no se pondrá ninguna linea de código que sea lo suficientemente grande ya que si se llegase a colocar, visualmente el código es difícil de leer, por esta razón se invita al lector consultar el archivo Texturas.ipynb. La función características arroja una matriz de dimensiones $(L \times 6)$ donde L depende del valor de tamaño de la ventana. Después de crear la base de datos general y entrenar los

clasificadores con los datos de la base de datos, se define otra función especial que llamamos validacion1, en el archivo Texturas.ipynb se encuentra otra función llamada validación, pero esta ultima es caso particular de la primera, que recibe como argumentos el tamaño de la ventana que recorrerá la imagen, la imagen en forma de matriz, el nombre de la instancia de un solo clasificador, y dos variables inicio y fin cuya finalidad es limitar las características que contendrá el vector que servirá al clasificador para determinar la clase. Dentro de está función se produce la imagen de validación. Limitar los valores contenidos en el vector para clasificación nos permite omitir algunas de las características de textura y predecir una clase sin considerarlas. Esta función nos permitió estudiar el desempeño de los clasificadores con distintas características.
Como primer resultado se obtuvieron bases de datos de dimensión (25×6) para cada imagen, siendo la base de datos general de dimensión (100×6). La clasificación de esta base de datos considerando Bayesiano ingenuo y SVM se muestra en la imagen 3

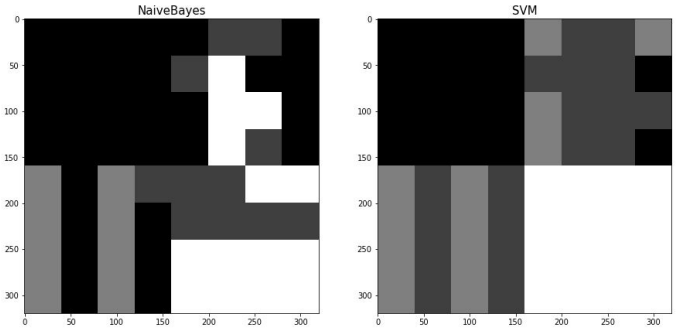

Fig. 3. Clasificación de la imagen de prueba de los clasificadores entrenados con 25 datos por clase considerando todas las características

Los resultados obtenidos para esta base de datos no fue muy buena, en particular para el caso de Bayes. Intuitivamente observamo que las clases correspondientes a la esquina inferior izquierda y esquina superior derecha pueden considerarse similares, ya que la mayoría de las equivocaciones en ambos clasificadores se concentran en estas dos clases. Para el caso de la SVM los datos de las clases de la esquina superior izquierda e inferior derecha están muy bien caracterizadas. A pesar de ello consideramos que la cantidad de datos es insuficiente como para asegurar que las clases estan suficientemente caracterizadas. Por esta razón se decidió obtener una nueva base de datos general de dimensión (400×6), es decir 100 datos por cada clase.
Antes de proceder con la presentación de resultados introduciremos la siguiente convención. La ventana superior izquierda será la clase 1, la superior derecha clase 2, inferior izquierda clase 3 e inferior derecha clase 4.
La figura 4 muestra la validación de ambos clasificadores considerando las seis características.

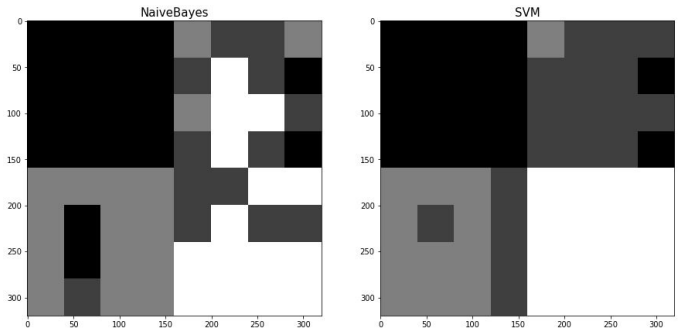

Fig. 4. Clasificación de la imagen de prueba de los clasificadores entrenados con 100 datos por clase considerando todas las características

Aumentar la cantidad de datos mejora el desempeño de ambos clasificadores. Sin embargo, aún se observan ocho errores de clasificación para el SVM, mientras que para el caso de Bayes aun se tiene una gran cantidad de errores. Nuevamente podemos decir que las clases 1 y 4 están muy bien caracterizadas.
Al dejar fuera las características de contraste y disimilitud tanto para entrenamiento y validación obtenemos las siguientes imágenes de validación

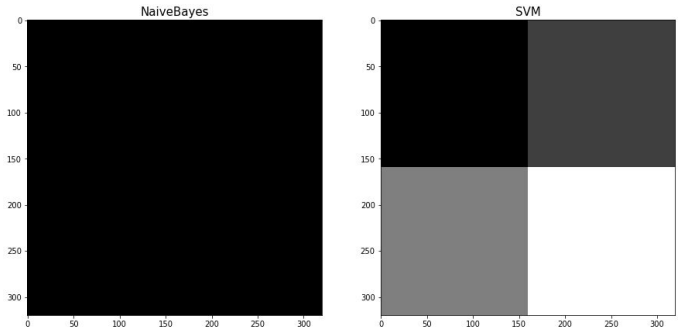

Fig. 5. Clasificación de la imagen de prueba de los clasificadores entrenados con 100 datos por clase sin considerar las características de contraste y disimilitud

Para este caso observamos que el SVM arroja el resultado ideal de clasificación, mientras que el clasificador Bayesiano comete demasiados errores. Esto puede deberse a que el clasificador bayesiano considera las probabilidades de cada textura. Estas probabilidades dependen de los valores de las características. Dado que las clases son uniformemente distribuidas el clasificador Bayesiano dará preferencia a aquellas clases cuyos valores de las características sean mayores. De aquí podemos deducir que la clase 1 es dominante con respecto a esto.
Para los experimentos donde se consideran únicamente tres características y se dejan fuera las características correlación, energía y ASM se obtienen las mismas imágenes de validación que la figura 4. Lo mismo sucede al dejar fuera las características correlación, energía, ASM y homogeneidad. El hecho de obtener resultados iguales incluso al retirar cuatro características nos indica que las características contraste y disimilitud son consideradas como atributos de gran peso para los clasificadores. Este hecho no es sorprendente al considerar que los valores que corresponden a estas características son mucho mayores a uno mientras que las

otras cuatro características son menores que uno. Para evitar este hecho se propone la regularización de los datos como forma de preprocesamiento. Cabe mencionar que esto no se realizó ya que la mayoría de métodos de regularización arrojaba una base de datos con algunos valores negativos y en el caso de querer analizarlas el clasificador Bayesiano arrojaba errores ya que la implementación que se utilizó no acepta valores negativos.
Al dejar fuera las características contraste, disimilitud, homogeneidad se obtienen las imágenes de validación que se muestran en la figura 6. La mismas imágenes se obtienen al dejar fuera las características contraste, asm, homogeneidad y disimilitud. Nuevamente Bayes muestra que la clase 1 es dominante.

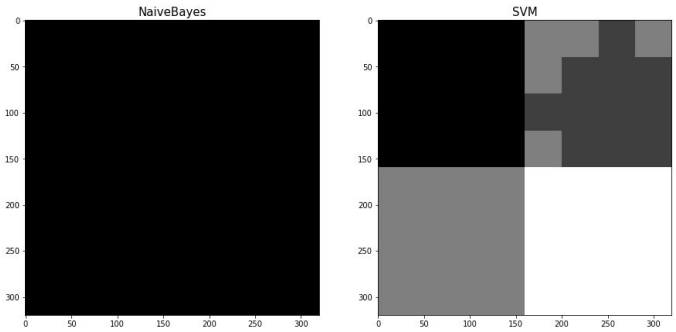

Fig. 6. Clasificación de la imagen de prueba de los clasificadores entrenados con 100 datos por clase sin considerar las características de contraste, disimilitud, asm y homogeneidad

Para esta configuración vemos que el SVM comete solo 5 errores por lo que lo consideramos como el segundo mejor resultado.
Al dejar fuera las características de contraste, correlación, energía y disimilitud obtenemos las siguientes imágenes de validación.

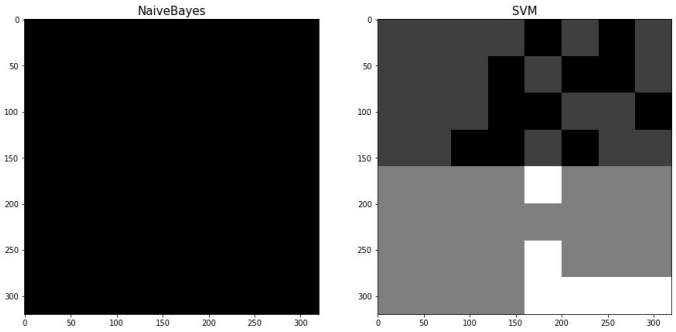

Fig. 7. Clasificación de la imagen de prueba de los clasificadores entrenados con 100 datos por clase sin considerar las características de contraste, disimilitud, energía y correlación

Para este caso el SVM no clasifica bien las muestras, siendo el peor resultado obtenido. Con esto podemos decir que, para el caso del SVM, las características asm y homogeneidad no aportan gran información para distinguir entre las clases.
Para la mayoría de las variaciones observamos que el clasificador Bayesiano da la clasifica las texturas como una sola clase. Esto puede deberse a lo ya discutido anteriormente del tamaño de los valores de cada característica. Sin embargo esto es solamente una hipotesis, que para comprobarse debería analizarse a fondo los subconjuntos y la implementación de este clasificador.
Finalmente la figura 8 muestra las imágenes de validación para el clasificador kNN considerando todos los datos y sin considerar las características de contraste y similitud.

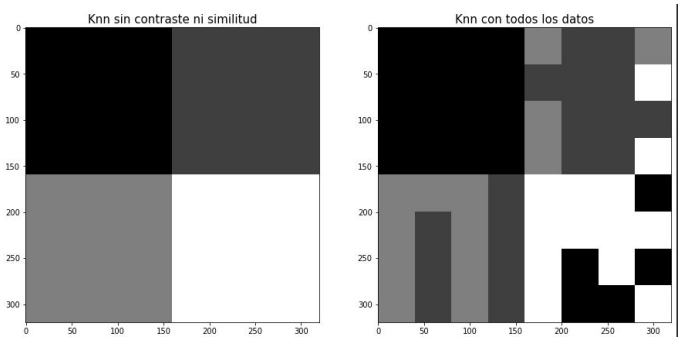

Fig. 8. Clasificación de la imagen de prueba del clasificador kNN entrenado con 100 datos por clase sin considerar las características de contraste y similitud (izquierda) y considerando todos los datos (derecha)

con kNN y sin considerar las características de contraste y similitud obtenemos el resultado de clasificación ideal. Por otro lado al considerar todas las características el clasificador kNN también falla al caracterizar determinadas regiones de la imagen. Notemos que al comparar las imágenes de validación producidas por los tres clasificadores al considerar todas las características cada imagen difiere entre sí, esto visualmente nos da una pista de que los tres clasificadores funcionan de manera diferente.

3.2 Superpíxeles

La imagen de entrenamiento se muestra en la figura 9. De una inspección visual encontramos que hay tres clases de interés: el centro café de la flor, los pétalos y el cielo. Note que las clases difieren entre sí de acuerdo a la textura.

Fig. 9. Imagen de entrenamiento para la parte de superpíxeles

Al aplicar el algoritmo SLIC con aproximadamente 750 particiones (aproximadamente ya que el algoritmo SLIC no garantiza que las particiones sean exactamente las que el usuario define) se obtiene una mascara de segmentado.

Dicha mascara se muestra sobrepuesta en la imagen original en la figura 10.

Fig. 10. Superpíxeles de la imagen de entrenamiento

Una vez obtenidos los superpíxeles se procede a convertir la imagen de entrenamiento a escala de grises, esto para calcular la GLCM. Para este caso el calculo de la GLCM es especial, dado un superpíxel, definimos una matriz auxiliar de ceros del mismo tamaño de la imagen de entrenamiento y en la posición definida por el superpíxel en cuestión se colocan los valores de gris en sus respectivas posiciones. Esto es sencillo de aplicar mediante operaciones matriciales básicas. El resultado de esta operación para un superpíxel dado se muestra en la figura 11.

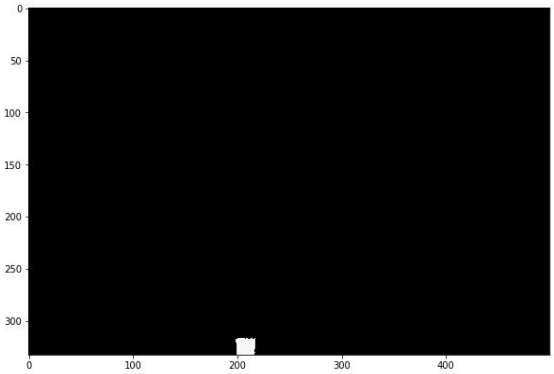

Fig. 11. Extracción de un superpíxel (parte inferior) usando el tamaño de la imagen de entrenamiento

A partir de la imagen 11 se calcula una GLCM. Dado que la GLCM considera los valores del nivel de gris cero (negro) y ajustamos una imagen cuyos valores de gris son cero si no están en la posición del superpíxel en cuestión, entonces para evitar el ruido producido por está región se modifica la GLCM de tal manera que se eliminan la primera columna y la primera fila, recordemos que estas contienen información de la relación del nivel de grises cero con los otros niveles de gris. Para esta parte se hace la suposición de que la imagen de entrenamiento no contiene ningún píxel cuyo nivel de gris es cero. Posteriormente se calculan las propiedades de la GLCM modificada y se guardan en una fila de un array de datos. Este proceso se realiza para cada uno de los superpíxeles obteniéndose un conjunto de datos de tamaño $(L \times 6)$ donde L depende del número de superpíxeles calculado por el algoritmo SLIC. La implementación de este proceso fue en forma de función llamada características del archivo Superpixel.ipynb. Esta función tiene como argumentos la imagen de entrenamiento y el conjunto de superpíxeles calculado por SLIC.
Como ya se mencionó para entrenar los clasificadores SVM y kNN fue necesario crear un array y de clases donde cada clase varía de 1 a la cantidad de superpíxeles. Antes de proceder a entrenar los clasificadores se decidió reescalar el conjunto de datos usando la función scale de sikit-learn.
La imagen de prueba también requiere de la obtención de superpíxeles, esto para obtener las características de cada región y poder predecir la clase usando alguno de los dos clasificadores. La imagen de validación con aproximadamente 50 superpíxeles se muestra en 12.

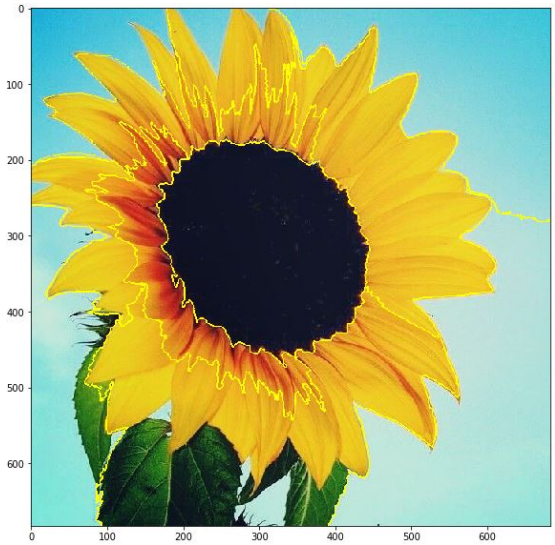

Fig. 12. Imagen de prueba con aproximadamente 50 superpíxeles

Usando la imagen de prueba superpíxeles se obtienen las características de cada superpíxel usando la función características. Con estas características se procede a predecir la clase usando los clasificadores previamente entrenados, obteniéndose así un vector de dimensiones $(L \times 1)$.
Para crear la imagen de validación se desarrollo una función llamada creaImagen cuyos argumentos son la imagen de validacion, los superpíxeles de la misma y el vector de predicciones. En esta función se encuentra iterativamente la posición de un superpíxel determinado y en una matriz auxiliar, de las mismas dimensiones de la imagen de prueba, se asigna el valor de la clase predicha a los píxeles en la posición del superpíxel. Después de iterar sobre todos los píxeles la función regresa la matriz auxiliar.
La imagen de validación generada por la función creaImagen y las predicciones de la SVM se muestran en la figura 13

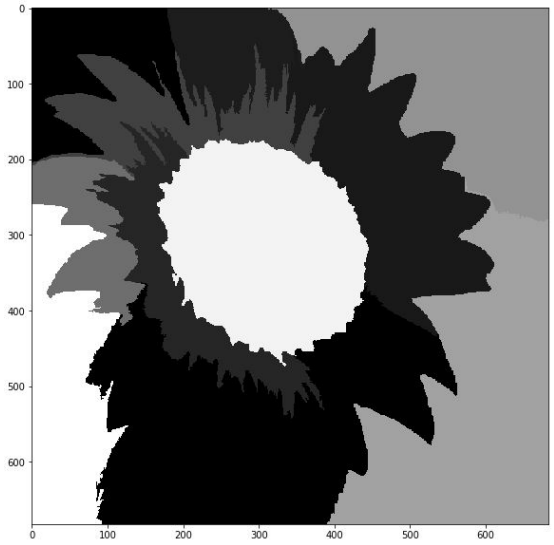

Fig. 13. Imagen de validación para el clasificador SVM

Notemos que la imagen de validación para el caso de SVM contiene al menos nueve clases diferentes. Observamos que el fondo (azul) de la imagen original contiene cuatro clases, donde una de ellas fue clasificada como el centro del girasol. Por otro lado el tallo podría considerarse clasificado como fondo o parte los pétalos. Los pétalos contiene cinco clases, cuyos niveles de gris aparentan no ser muy lejanos entre sí. Lo mismo sucede con lo que podríamos considerar fondo (gris claro de la parte derecha de la imagen). A pesar de obtener una imagen con más clases que las originales y con gran cantidad de errores, consideramos este resultado como bueno ya que el resultado preserva los bordes de las clases inicialmente buscadas.
La imagen de validación generada por la función creaImagen y las predicciones del kNN se muestran en la figura 14.

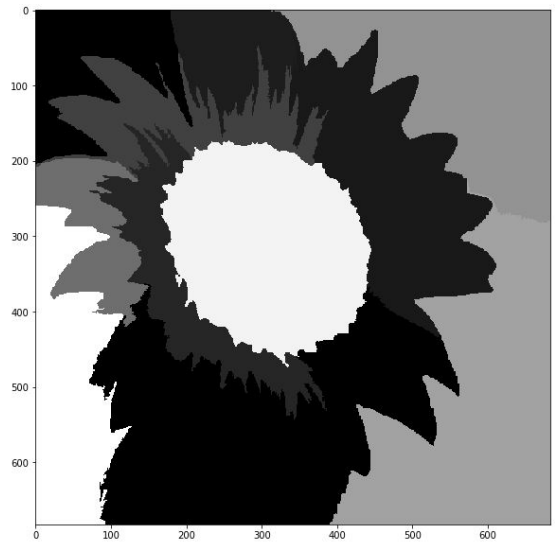

Fig. 14. Imagen de validación para el clasificador kNN

La imagen de validación es muy similar a la obtenida por SVM, por consiguiente tenemos la misma discusión.
Para el caso de k-medias, como ya se mencionó, se aplica el algoritmo de k-medias a los datos de entrenamiento con 3 clusters y se predicen los datos obtenidas mediante la función características y finalmente empleamos la función creaImagen para obtener una clasificación con k-medias. El resultado de se muestra en la figura 14.

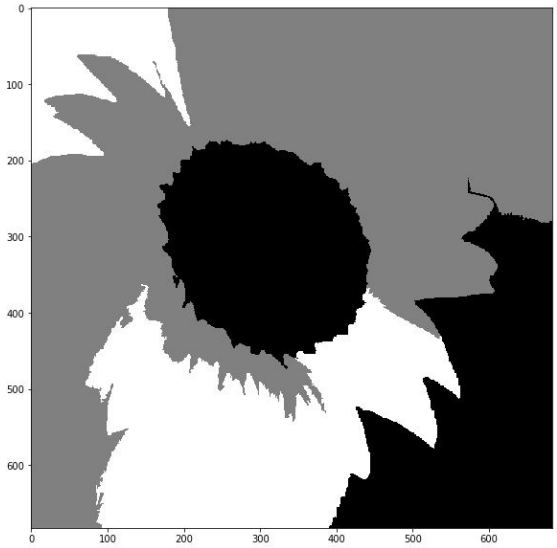

Fig. 15. Imagen de validación para el clasificador k-medias

Visualmente el resultado ofrecido por el algoritmo k-medias tiene un rendimiento menor comparado a los clasificadores kNN y SVM, sin embargo, debe tomarse en cuenta que el fondo de la imagen de prueba y entrenamiento difieren en color y por ende en el valor de escala de grises, lo mismo sucede con el centro y los pétalos y bajo la constricción de únicamente tener tres clases esta variación afecta significativamente el rendimiento de k-medias se esperaría que sí no existieran variaciones de color muy marcadas en las imágenes de entrenamiento y prueba, entonces el rendimiento será mejor comparado a los otros clasificadores. A pesar de ello, obtenemos una clasificación muy buena en la clase del centro del girasol pero perdemos algunos bordes de las clases de interés.
Finalmente cabe destacar que los grados de libertad presentes en los clasificadores SVM y kNN y la variación de color en ambas imagenes permiten conservar los bordes obteniéndose así una imagen de validación visualmente mejor comparada con k-medias.

4 Conclusiones

4.1 Ventanas cuadradas

Para el caso de los clasificadores SVM y kNN se encontró que las características que definen correctamente las cuatro clases son homogeneidad, asm, energía y correlación, al entrenar estos clasificadores con dichas características y validar la imagen de prueba obtuvimos una precisión del 100% (imagen ideal) para ambos clasificadores. Por otro lado se encontró que reducir las características el clasificador Bayesiano clasificó todas las clases como si solo pertenecieran a una sola, por lo que de esta manera concluimos que el clasificador Bayesiano requiere de las seis características para alcanzar un rendimiento regular.

Para los otros dos clasificadores se obtuvo que al considerar todos los datos el rendimiento de los mismos puede considerarse aceptable. De acuerdo a nuestros resultados encontramos que el clasificador Bayesiano no debería ser una opción a considerar para la clasificación de texturas. Para esta parte, la reproducibilidad de los resultados para diferentes características se debe a que los datos no fueron estandarizados. Esto es claramente un error de diseño, sin embargo al intentar estandarizar o regularizar los datos no fue posible hacer funcionar el clasificador Bayesiano y sin éste una comparación directa entre los tres clasificadores no habría sido posible.

4.2 Superpíxeles

Las imágenes de validación para el caso de los clasificadores SVM y kNN preserban los bordes de las clases deseadas, siendo estas dos imágenes muy similares entre sí. Se encontró que la clase más propensa a errores corresponde al fondo azul, esto se debe a la variación de color entre las imágenes de entrenamiento y prueba. Así mismo debido a que los clasificadores SVM y kNN fueron entrenados con la misma cantidad de clases que superpíxeles se obtiene que la imagen de validación contiene al menos 9 clases, permitiendo así un resultado visualmente mejor comparado al algoritmo de k-medias. Por otro lado debido a las variaciones de color y la constricción a tres clases, el rendimiento de este clasificador es bajo, produciéndose así la perdida de algunos bordes de clases de interés. Como hipótesis se propone que el rendimiento del clasificador tendría mejor rendimiento con respecto a los otros dos si ambas imágenes no presentaran una variación de color muy marcada.

REFERENCES

[1] M. Nixon, A. Aguado *Feature Extranction and Image Processing for Computer Vision*, 3rd ed. Elsevier Academic Press, 2012.

[2] P. Brodatz, *A Photographic Album for Artist and Designers*. Reinhold, USA.

[3] K. Karru, A. Jain, R. Bolle *Is there any texture in an image?*, Pattern Recog. 29 (9), 1437-1446.

[4] A. Humeau-Heurtier *Texture Feature Extraction Methods: A Survey*, IEEE Access, 7, 8975-9000, 2019.

[5] O. Sutton *Notas: Introduction to k Nearest Neighbour Classification and Condensed Neares Neighbour Data Reduction*, University of Leicester, 2012. http://www.math.le.ac.uk/people/ag153/homepage/KNN/OliverKNN_Talk.pdf. Consultado el 25.04.20

[6] S. Ullman *Notas: Unsupervised learning: Clustering*, MIT, 2014. http://www.mit.edu/~9.54/fall14/slides/Class13.pdf. Consultado el 25.04.20

[7] T. Fletcher *Notas: Support Vector Machines Explained*, UCL University, 2008. https://cling.csd.uwo.ca/cs860/papers/SVM_Explained.pdf. Consultado el 25.04.20

[8] A. Anónimo *Superpixel: Empirical Studies and Aplications*, Chigago University, sin año. https://ttic.uchicago.edu/~xren/research/superpixel/. Consultado el 26.04.20

[9] D. Stutz, A. Hermans, B. Leibe *Superpixels: An Evaluation of the State-of-the-Art*, Computer Vision and Image Understanding, 2017.

[10] J. Darshita *Superpixels and SLIC*, Medium, 2019. https://medium.com/@darshita1405/superpixels-and-slic-6b2d8a6e4f08. Consultado el 26.04.20