

Clasificación de 15 tipos hojas suecas con un enfoque del aprendizaje de maquina clásico

Flores-Silva P.

Abstract—La clasificación *in situ* de objetos orgánicos es tópico de gran importancia para la industria agrícola. En el presente trabajo se resolvió un problema de clasificación de hojas cuyo entorno era de densidad nula. Si bien es cierto este problema se aleja de la realidad, este trabajo puede servir como base para la construcción de un clasificador con densidad no nula. Para la clasificación se empleó un conjunto de 15 clases de hojas suecas con 75 imágenes por cada clase. A partir del segmentado de las hojas de cada imagen se obtuvieron 14 características de textura empleando superpíxeles, el operador LBP y la matriz de coocurrencia. El análisis exploratorio reveló la existencia de correlación entre algunas clases para los datos obtenidos. Se entrenó una maquina de soporte vectorial considerando todos las características y se encontró que el valor del parámetro de regularización óptimo C fue de 50. Con este parámetro se obtuvieron las curvas ROC de cada clase así como su matriz de confusión y su reporte de clasificación obteniéndose puntajes promedio de precisión de 0.91 u.a., recall de 0.90 u.a. y f1 de 0.91 u.a. Por lo que se considera que las características obtenidas y el clasificador en general son muy buenos. Finalmente se realizó clasificación *in situ* considerando una restricción extra de las imágenes de prueba, debido a que el conteo de error se hizo manual solo fue posible obtener el puntaje de precisión que resultó ser de 0.88 u.a. que se considera, también, muy bueno.

NOTA: El código desarrollado para este artículo y las instrucciones para obtener las imágenes puede encontrarse en la siguiente dirección web: <https://github.com/Pedri0/Leafs-ProyectoAprendizajeDeMaquina>.

Index Terms—Aprendizaje de Maquina, Visión computacional, Detector de hojas, SVM, Superpíxeles, Textura.

1 INTRODUCCIÓN AL PROBLEMA

Recientemente el reconocimiento de patrones junto con la inteligencia artificial, en particular el aprendizaje de maquina y el aprendizaje profundo, se han consolidado como una rama conjunta que ha atraído un enorme interés debido a la gran cantidad de problemas desafiantes cuya solución u aproximación podría deberse al conjunto de éstas [1], [2]. Dentro de este gran conjunto de problemas encontramos al que a este proyecto concierne; la clasificación de plantas de forma automática es un problema vigente en la industria agrícola. La necesidad de clasificar *in situ* una cierta cantidad de clases de plantas, a partir de imágenes, es trascendental para la recolección automatizada de dichas, ahorro de costos en la recolección entre otros. Muchos otros objetivos pueden plantearse para este sector, tales como la detección de enfermedades y/o plagas dentro de un sembradío específico, sin embargo, el problema que será atacado, desde el punto de vista de la visión computacional y aprendizaje de maquina, será, justamente, el primero de ellos.

2 DELIMITACIÓN DEL PROBLEMA Y OBJETIVOS

Plantear una solución al problema propuesto dependerá del conjunto de imágenes disponibles, ya que un conjunto de imágenes cuyo entorno presenta una gran variedad de objetos que no son de interés particular requiere de un procesado y segmentado muy sofisticados. La solución propuesta por Olson A. et Al. [3] consiste en el empleo de una red neuronal convolucional para la segmentación de imágenes con un entorno de densidad alta. Otros trabajos similares [4], [5], [6] cuyo objetivo primordial consiste en la segmentación de imágenes solucionan este mismo problema empleando aprendizaje profundo. La aplicación de aprendizaje profundo se encuentra fuera de

nuestro alcance debido a que nuestro objetivo es aplicar algunos de los métodos provistos por el aprendizaje automático clásico. Teniendo en cuenta la complejidad del problema y el problema en sí, nuestro objetivo consiste en el desarrollo de un clasificador de hojas comunes en Suecia, cuyo conjunto de datos (imágenes) es de muy baja densidad con respecto a los objetos presentes.

Si bien es cierto que este conjunto de datos se aleja drásticamente de la realidad, ya que es prácticamente inverosímil esperar obtener imágenes de baja densidad en espacios naturales, la solución obtenida mediante dicho conjunto permite sentar las bases de análisis mucho más complejos como aquellos que emplean el aprendizaje profundo para la segmentación de imágenes.

De una manera muy sintetizada la solución propuesta para este problema consiste en extraer las características de textura de cada imagen, para, posteriormente, entrenar una maquina de soporte vectorial con dichos datos. Para entender la solución completa, expuesta en la sección desarrollo experimental, es necesario entender los conceptos básicos de textura, maquinas de soporte vectorial, entre otros. Estos conceptos se explican de una manera muy breve en la sección introducción teórica.

3 INTRODUCCIÓN TEÓRICA

3.1 Texturas

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 [8]. Por lo que si nos referimos al diccionario *Oxford Concise English Dictionary* encontramos una definición de textura como (traducida del inglés) *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*

• P. Flores-Silva estudiante del PCIC en la UNAM.
E-mail: flosipan@ciencias.unam.mx
Link misterioso: <https://doi.org/10.1016/j.physen.2020.114282>

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*) [8]. Como definición 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 [8], un ejemplo de ello es la base de datos de Brodatz [9]. 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 Karrru [10] 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*. En este trabajo se centrará únicamente en el enfoque estadístico.

3.1.1 Enfoque Estadístico: Matriz de Coocurrencia

El enfoque estadístico más famoso es la *matriz de coocurrencia* o GLCM por sus siglas en inglés (Gray Level Cooccurrence Matrix). Este enfoque fue el primer método para describir y clasificar la textura de imágenes propuesto por Haralick [8]. 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:

$$x' = x + d\cos(\theta) \quad (2)$$

$$y' = y + d\sin(\theta) \quad (3)$$

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 coocurrencia 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 [8].

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, Connors y Harlow encontraron que de esas 28 características solo cinco de ellas son suficientes para describir la textura [11]. 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 encuentran 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

$$\sum_{i=1}^G \sum_{j=1}^G (i - j)^2 P(i, j|d, \theta)$$

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 [11].

3.1.2 Patrones binarios Locales (LBP)

Otro método para describir texturas consiste en el operador LBP que etiqueta los píxeles de una imagen con números decimales de [0 a 255] llamados códigos LBP que describen la estructura local alrededor de cada píxel dentro de una vecindad cuadrada de 3×3 píxeles. Cada píxel es comparado con sus ocho vecinos. Dado un valor de gris $I(x_c)$ del píxel central de la vecindad se define el operador LBP sobre la vecindad de la siguiente manera.

$$LBP(x_c) = \sum_{p=1}^{P=8} \phi(I(x_p) - I(x_c)) \times 2^{p-1} \quad (4)$$

donde ϕ es la función de Heaviside. No necesariamente la vecindad debe ser de dichas dimensiones y puede extenderse a mayores valores, sin embargo, para obtener las propiedades de textura se recomienda usar ventanas proporcionalmente pequeñas comparadas con el tamaño de la imagen [12]. Al aplicar este operador sobre una imagen determinada se obtiene una imagen con diferentes valores decimales, donde el histograma de ésta última permite describir la textura de la imagen original. El método descrito es únicamente invariante ante la translación, por lo que para agregar invarianza ante la escala y rotación se consideran el pesado de los valores de las

intensidades y puntos a distancias mayores determinados por un área circular respectivamente [12].

3.2 Maquinas de Soporte Vectorial (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:

$$\{\vec{x}_i, y_i\} \quad (5)$$

donde $i = 1, 2, \dots, L$, $y \in \{-1, 1\}$ y $x \in \mathbb{R}^D$.

Se asume que los datos son linealmente separables, es decir, es posible dibujar un hiperplano en las gráficas de x_1, x_2, \dots, 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.

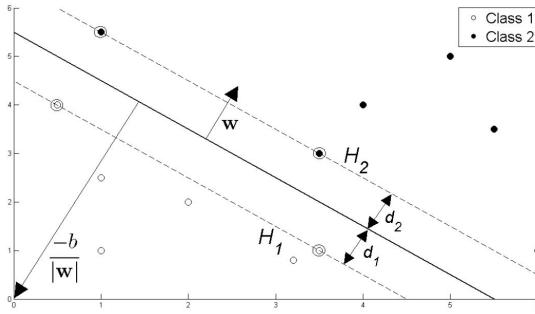


Fig. 1. Hiperplano divisor de dos clases linealmente separables. Tomada de [13]

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:

$$\vec{x}_i \cdot \vec{w} + b \geq +1 \quad si y_i = +1 \quad (6)$$

$$\vec{x}_i \cdot \vec{w} + b \leq -1 \quad si y_i = -1 \quad (7)$$

Las dos ecuaciones anteriores se reducen a:

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 0 \quad (8)$$

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:

$$\vec{x}_i \cdot \vec{w} + b = +1 \quad H_1 \quad (9)$$

$$\vec{x}_i \cdot \vec{w} + b = -1 \quad H_2 \quad (10)$$

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 restricción 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 [13]. 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.

3.3 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 [14]. Regularmente la simetría de dichas mallas se ajusta a una simetría cuadrangular sin cambios en sus dimensiones.

Sin embargo, una cuadrícula 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 [15]. Estos superpíxeles deben tener las siguientes propiedades [14], [15]:

- 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 superpíxeles.
- 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 traslaparse (este hecho es obvio al considerar la restricción 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 definición 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 [15]. De una manera breve se explicará el algoritmo Simple Linear Iterative Clustering (SLIC) perteneciente a la categoría basada en clusterización.

3.3.1 Algoritmo SLIC

El algoritmo SLIC se basa en la técnica de clusterización 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 [16]:

- 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

3.4 Momentos de Imagen

Un momento de imagen es una media ponderada en función de las intensidades de los píxeles de la imagen. Estos momentos son útiles para determinar y describir algunos objetos después de segmentarlos [17]. Los momentos más sencillos son los llamados espaciales definidos como

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y) \quad (11)$$

donde $I(x, y)$ es la función que define las intensidades de los píxeles de una imagen en escala de grises en las posiciones x e y . A partir de esta definición puede obtenerse el área de una imagen al calcular M_{00} .

Los siguientes momentos son los momentos centrales, que son una generalización de los momentos espaciales, dados por:

$$\mu_{pq} = \sum_x \sum_y (x - \hat{x})^p (y - \hat{y})^q f(x, y) \quad (12)$$

donde $\hat{x} = M_{10}/M_{00}$, $\hat{y} = M_{01}/M_{00}$ y $f(x, y)$ es una imagen digital. Debido a su definición los momentos centrales son invariantes ante translaciones. Para extender la invarianza ante la escala se construye los momentos invariantes a escala a partir de los momentos centrales. Estos momentos invariantes a escala se definen como:

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{1+(i+j)/2}} \quad (13)$$

Finalmente, para extender los momentos a modo que sean invariantes ante la escala, translación y rotación se definen los momentos de Hu dados por:

$$I_1 = \eta_{20} + \eta_{02} \quad (14)$$

$$I_2 = \eta_{11}^2 + 4\eta_{11}^2 \quad (15)$$

$$I_3 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (16)$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} - \eta_{03})^2 \quad (17)$$

$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (18)$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \quad (19)$$

$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (20)$$

4 DESARROLLO EXPERIMENTAL Y RESULTADOS

Como ya se mencionó se empleará una conjunto de imágenes de hojas comunes en Suecia. Este conjunto puede obtenerse en [7]; consiste de quince clases diferentes de hojas con 75 imágenes por clase con un total de 1125 imágenes. La figura 2 muestra cuatro imágenes del conjunto de cuatro clases distintas con su respectiva clase. Dado que el nombre de las clases o especies no es relevante se decidió enumerarlas del 0 al 14. Debido a que las imágenes fueron capturadas, aparentemente, con una cámara fotográfica a diferentes distancias, el fondo puede presentar alguna especie de ruido por lo que fue necesario realizar una segmentación previa de cada imagen y además se trabajaron, la mayor parte del tiempo, en escala de grises ya que los algoritmos de textura requieren este tipo de imágenes. De esta forma se dividió el desarrollo experimental en cuatro etapas: *segmentado, obtención de datos, análisis exploratorio de datos con pruebas basadas en valores numéricos y clasificación on-line*, para cada una de éstas se desarrolló una libreta de jupyter con los nombres **LeafsDoMasks.ipynb**, **LeafsGetData.ipynb**, **SVM.ipynb** y **OnlineClassification.ipynb** respectivamente. Las siguientes subsecciones presentan el desarrollo de cada libreta así como sus resultados más destacables.



Fig. 2. Imágenes de cuatro clases diferentes del conjunto de datos original.

4.1 Segmentado

Una inspección visual rápida del conjunto de imágenes reveló que el fondo o zonas no pertenecientes a la hoja contenía colores amarillezcos, dado que nuestro interés reside en la hoja y los algoritmos de textura permiten definir un umbral de análisis para intensidades de gris fue necesario uniformizar el fondo a un único valor, que se definió como cero o color negro. Para ello fue necesario binarizar la imagen en escala de grises. Esta binarización es posible gracias al método de Otsu que consiste en el análisis del histograma de la imagen para definir y separar dos clases en la imagen. El resultado de la binarización para las primeras 25 imágenes de la clase 0 se muestra en la imagen 3. Dado que las imágenes binarias obtenidas contienen valores 0 o 1, donde 0 corresponde al fondo, es posible obtener imágenes segmentadas con el fondo uniforme en escala de gris al multiplicar entrada por entrada la imagen original con su respectiva imagen (tomando en cuenta que una imagen en escala de gris es en realidad una matriz cuyas entradas tienen valores entre 0 y 255). El resultado de este segmentado para una imagen de la clase 9 se muestra en la figura 4.

Una vez obtenidas las imágenes binarias y segmentadas se procede a guardarlas en carpetas llamadas máscaras y binarios respectivamente. Donde cada una de ellas contiene 15 carpetas correspondientes a cada clase con sus respectivas imágenes. Esto se hace para tener un control sobre las imágenes y clases. Cabe mencionar que las imágenes binarias son necesarias para los algoritmos de textura, ya que estos admiten analizar regiones de interés dictadas por un parámetro que corresponde a imágenes binarias, donde el algoritmo analizará zonas de una imagen de acuerdo a los valores de dicho parámetro. Esta función es muy útil ya que nos evita agregar valores que no correspondan a la hoja en sí.

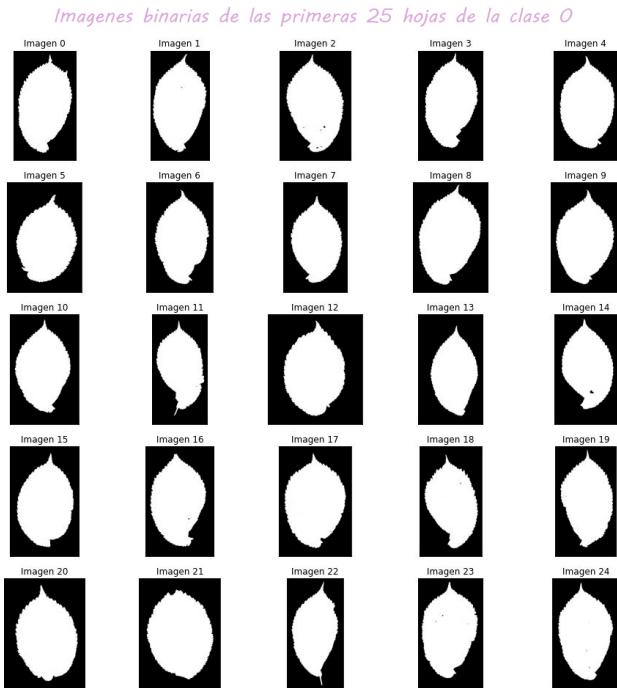


Fig. 3. 25 Imágenes binarias de la clase cero, note que en algunas zonas dentro de la hoja se observan islas negras, esto se debe a que las hojas en las imágenes originales contenían hoyos



Fig. 4. Imagen segmentada con fondo uniforme en escala de gris correspondiente a la clase 9

4.2 Obtención de datos

Para esta parte se comienza leyendo las imágenes binarias y segmentadas obtenidas en el paso anterior. Y se dividen en dos conjuntos de entrenamiento y prueba, siendo el 80% de las imágenes totales para el primer conjunto y el otro 20% para el segundo. Para que el conjunto de entrenamiento estuviera equilibrado con respecto a las clases se forzó a que cada clase contenga 60 imágenes, mientras que para el conjunto de prueba se forzó a que cada clase contenga 15 imágenes. Posterior a esto se procedió a obtener los superpíxeles de cada imagen en cada conjunto mediante el algoritmo SLIC, forzando a que los superpíxeles calculados se encuentren únicamente en zonas que corresponden a la hoja a través de las imágenes binarias. El algoritmo SLIC arroja una matriz de las mismas dimensiones que la imagen analizada donde cada píxel contiene un valor de cero a el número de superpíxeles deseados y un píxel corresponde a un superpíxel según su valor. El valor óptimo en cuanto a tiempo de computo se encontró que fue de 10 superpíxeles por cada imagen. La figura 5 muestra cuatro imágenes del conjunto de entrenamiento de la clase 7 con sus superpíxeles superpuestos (líneas amarillas). Visualmente un superpíxel lo consideramos como una región amarilla cerrada.

Posteriormente, a cada imagen segmentada se le aplica el operador LBP, con una distancia de vecindad de 32 y un radio de curvatura de valor 3 y se calcula el histograma de cada una de las imágenes. Dado que el histograma completo describe las propiedades de textura pero tiene una alta dimensionalidad (256 valores) se decide normalizar el histograma y calcular el área truncada del mismo. Este algoritmo, con los parámetros indicados, regresa una imagen LBP con una intensidad máxima de gris de 32+3 y para nuestro caso, la imagen cuyos valores son mayores a 32 corresponden a zonas de la imagen como el fondo o partes de la hoja que no aportan, visualmente, gran información de la textura. Con esto en mente es posible reducir la dimensionalidad que presenta el almacenar el histograma y el área truncada nos garantiza que extraemos información relevante de la textura de cada clase. La imagen 6 es la imagen resultante al aplicar el operador LBP a una imagen de la clase 1. Note que el fondo y ciertas regiones de la hoja comparten el mismo color, estas zonas son las que no se consideran en el área truncada.



Fig. 5. Cuatro imágenes segmentadas y sus superpíxeles superpuestos (regiones limitadas por líneas amarillas)

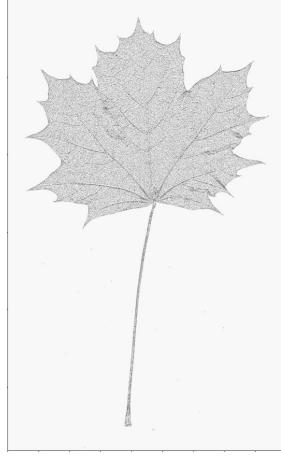


Fig. 6. Imagen resultante al aplicar el operador LBP a una imagen de la clase 1

Una vez obtenidos los superpíxeles y el área de las imágenes LBP se procede a obtener sus características de textura a través de la matriz de coocurrencia y los momentos de Hu. Las propiedades de textura que se obtuvieron fueron contraste, disimilitud, homogeneidad, asm, energía y correlación. Para realizar esto se itera con tres ciclos for sobre las clases, las imágenes y los superpíxeles de cada imagen (función datos-Textura de la libreta correspondiente). En el ciclo donde se considera una imagen de una determinada clase se calculan sus momentos de Hu, mientras que en el ciclo anterior donde se considera cada superpíxel se crea una imagen auxiliar del mismo tamaño que la original con valor cero en cada píxel para posteriormente en la región del superpixel determinada por la matriz arrojada por el algoritmo SLIC se configura en dichas posiciones un valor de 1 para posteriormente multiplicar, entrada a entrada, la imagen original con esta imagen auxiliar, es decir se crea una imagen segmentada del superpíxel en turno. A este superpíxel segmentado se le calcula su matriz

de coocurrencia con distancia uno, un ángulo de cero radianes y todos los niveles de gris ([0,255]). Dado que el superpíxel segmentado contiene valores en el color negro (valores cero) la matriz de concurrencia tendrá un valor de las cuentas para este color muy alto, sin embargo, dicha matriz contiene un orden y se sabe que la primera fila y primera columna corresponden a las cuentas de los valores cero, por lo que al eliminar esta columna y fila solo se consideran las cuentas de los niveles de gris de 1 a 255. Aquí se está realizando la suposición de que el área correspondiente a las hojas no contiene ningún nivel cero o su cantidad es mucho menor con relación a los otros. Esta suposición es coherente, al menos, visualmente. Con la matriz de coocurrencia recortada se calculan las propiedades de textura. Se encontró que los valores de dichas propiedades variaban en un rango de 1×10^{-3} a 200 unidades. Por lo que se procedió a aplicar una transformación logarítmica según su rango de variación. Las transformaciones empleadas fueron:

$$t(x) = \ln(|x|) \quad (21)$$

Para las propiedades de contraste, disimilitud y los momentos de Hu.

$$t(x) = -1\text{sign}(x)\ln(|x|) \quad (22)$$

Para las propiedades de homogeneidad, asm y energía. Dado que los valores de la correlación se encuentran entre 0 y 1 a esta propiedad no se le aplicó ninguna transformación.

Para cada superpíxel de cada imagen se guardaban, en un data frame, las propiedades de textura contraste, disimilitud, homogeneidad, asm, energía, correlación, momentos de Hu y el área correspondiente a la imagen previamente calculados y su respectiva clase. Siendo en total 14 características más el valor de su clase. Note que para una imagen dado que se tienen 10 superpíxeles se obtendrán 10 filas con 15 columnas de datos, dado que los momentos de Hu y el área tienen un único valor esas 10 filas de datos para una imagen en particular tendrán los mismos valores de los momentos y el área. Para este caso los valores de la textura son los que variarán. Cabe mencionar que se eligió analizar las propiedades de textura a través de los superpíxeles ya que estas propiedades son sensibles a la escala y el hecho de considerar la hoja como un todo podría estarse obteniendo características de textura generales que serían aplicables a todo el conjunto ya que, visualmente, algunas clases aparentan tener la misma textura global. El caso contrario consiste en considerar una enorme cantidad de superpíxeles, sin embargo, no se estaría obteniendo una buena generalización y el coste en tiempo computacional es muy alto. De esta forma se óptimo por un punto intermedio. Al terminar este ciclo, se obtuvieron dos dataframes, uno por cada conjunto (entrenamiento y prueba). Para el caso del data frame de prueba se obtuvieron 2196 filas con 15 columnas, mientras que el data frame de entrenamiento consiste de 8800 filas con 15 columnas. Esta cantidad de datos es considerablemente buena y podría pensarse que sería un conjunto suficientemente grande tal que pueda generalizar los datos. Estos data frames se guardan en archivos .csv separados llamados **texturaTest10y10.csv** y **texturaTrain10y10.csv** para los conjuntos de prueba y entrenamiento respectivamente.

4.3 Análisis exploratorio de datos con pruebas basadas en valores numéricos

Para la visualización de los datos, se comenzó aplicando análisis de componentes principales a dos y tres dimensiones, las gráficas obtenidas se muestran en las figuras 7 y 8, debido a que ninguno de los dos métodos mostró, visualmente, una clara separación de los datos la idea de aplicar PCA sobre el

conjunto de entrenamiento fue descartada inmediatamente y es por esta razón por la que la introducción teórica no agrega este tema en particular.

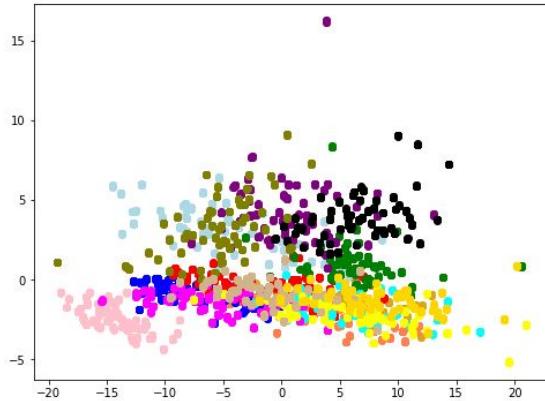


Fig. 7. Análisis de componentes principales en dos dimensiones. Cada color representa una de las 15 clases

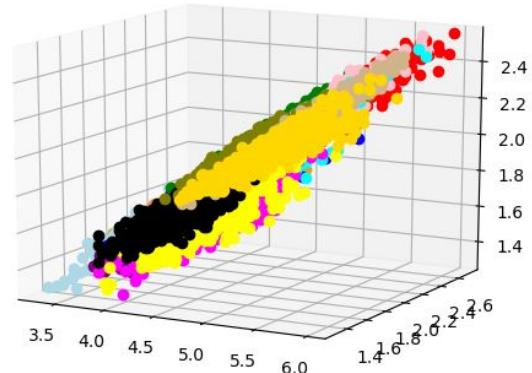


Fig. 9. Gráfica en 3D de las características contraste disimilitud y homogeneidad

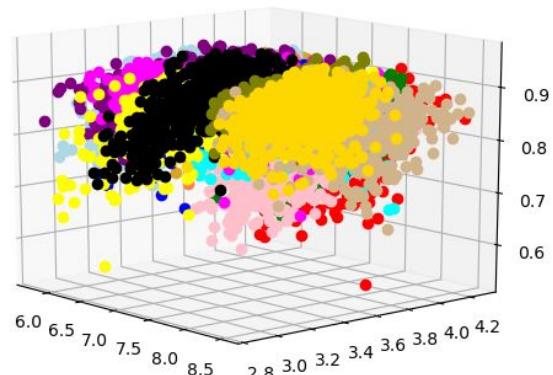


Fig. 10. Gráfica en 3D de las características asm, energía y correlación

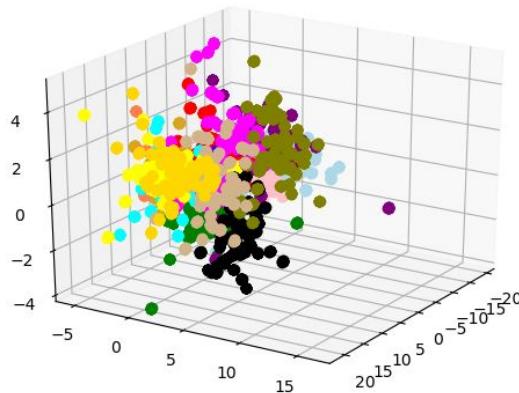


Fig. 8. Análisis de componentes principales en tres dimensiones. Cada color representa una de las 15 clases

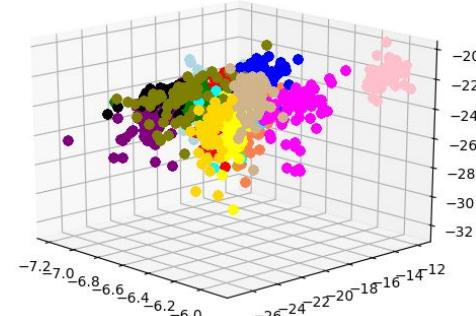


Fig. 11. Gráfica en 3D de los primeros tres momentos de Hu

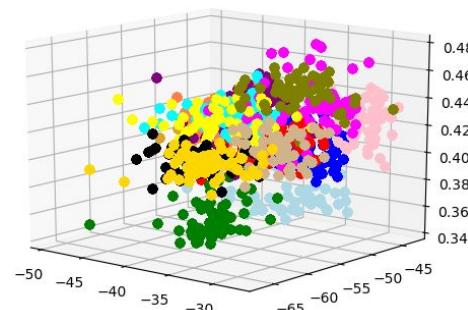


Fig. 12. Gráfica en 3D de los últimos dos momentos de Hu y el área

Se graficaron en tres dimensiones tres clases por separado dejando a las otras fuera esto con el fin de observar si es posible usar únicamente tres de ellas. Sin embargo, los resultados muestran que existe una alta correlación entre clases en tres dimensiones. Note que pueden realizarse una gran cantidad de combinaciones. Para nuestro caso solo se decidió tomar las características vecinas en el data frame por simplicidad. Los resultados se muestran en las figuras 9, 10, 11 y 12.

Al considerar todos los datos se realizó una gráfica en seaborn de densidad bivariada, univariada y de dispersión. El resultado se muestra en la imagen 13. h_j representa el j -ésimo momento de Hu.

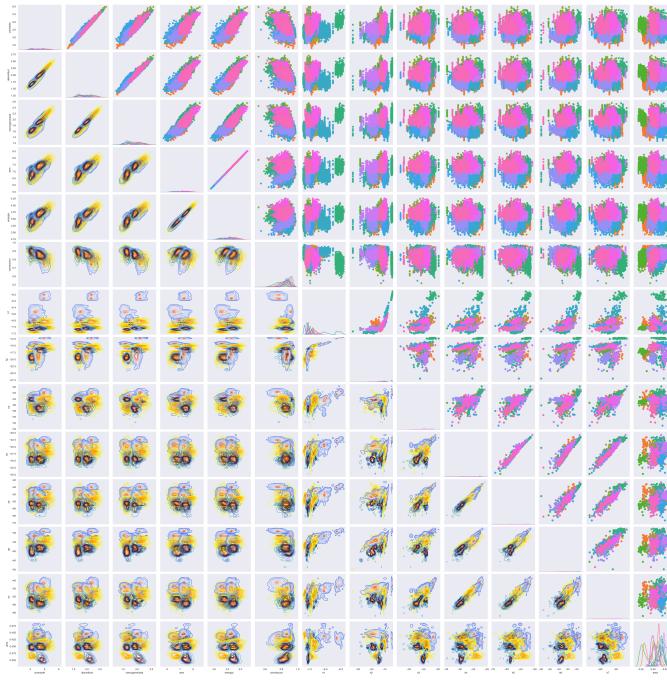


Fig. 13. Gráfica de dispersión, univariada y bivariada de todos los datos considerando dos clases. De arriba a abajo: contraste, disimilitud, homogeneidad, asm, energía, correlación, h_1 , h_2 , h_3 , h_4 , h_5 , h_6 , h_7 y área. Se sigue el mismo orden de izquierda a derecha

Se analizó la correlación por pares mediante un mapa de calor (heatmap) el cual se muestra en la figura 14.

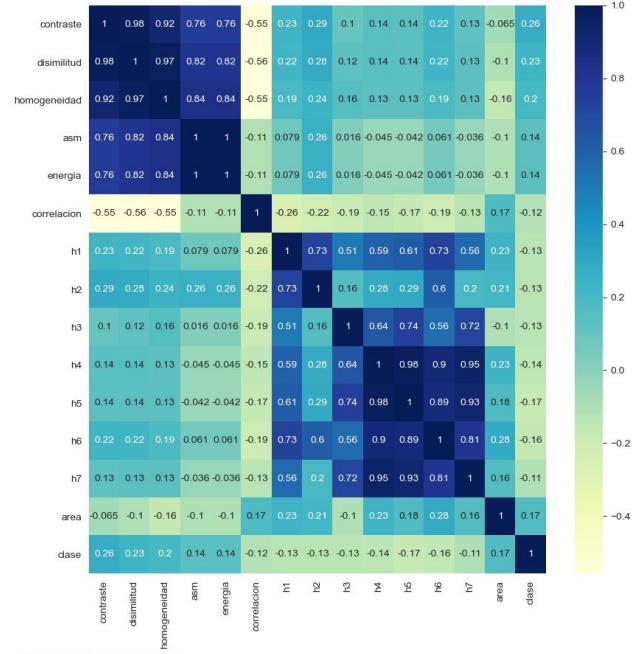


Fig. 14. mapa de calor que analiza la correlación por pares de todo nuestro conjunto de entrenamiento

Para finalizar el análisis exploratorio se despliega la información más relevante del conjunto de entrenamiento mediante

la función de pandas describe(). Este análisis exploratorio permite intuir el desempeño que tendrá nuestro clasificador y la posibilidad, de que después de un análisis conjunto de estos resultados, de la eliminación de características con el fin de reducir la dimensionalidad y el costo computacional.

El análisis exploratorio indica, visualmente, que una máquina de soporte vectorial podría ser suficiente para este problema. Por lo que se entrenaron con el conjunto de entrenamiento 7 SVM's con diferentes kernels y se calculó su puntaje de precisión con los datos de prueba. Los resultados fueron: Para el kernel radial basis function (rbf) se obtiene un puntaje de 0.58 u.a., para el kernel sigmoide: 0.0 u.a., para el kernel polinomial de grado 3: 0.67 u.a., para el kernel polinomial de grado 5: 0.74 u.a., para el kernel polinomial de grado 8: 0.77 u.a., para el kernel polinomial de grado 19: 0.82 u.a y para kernel lineal: 0.85 u.a. Cabe aclarar que el único parámetro que se cambió fue el kernel, dejando los parámetros que ofrece scikit learn por defecto. Debido a la relativamente baja cantidad de datos, el entrenamiento no tardaba más de diez minutos.

Tomando en cuenta estos resultados se seleccionó la SVM con kernel lineal y se estudió el impacto de la constante de regularización C sobre el puntaje de precisión, por lo que al variar C de dos en dos unidades hasta un valor de 199 se obtiene una gráfica del puntaje de regularización en función de la constante C . Dicha gráfica se muestra en la figura 15. Se encontró que la curva del puntaje de precisión es asintótica a la constante 0.91 y además al incrementar el valor de C el tiempo de entrenamiento aumentaba. Con esto en mente se seleccionó como valor óptimo $C = 50$

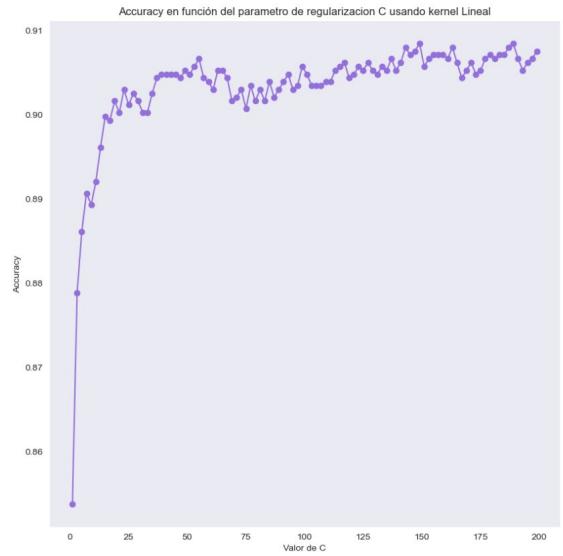


Fig. 15. Curva del puntaje de precisión en función del parámetro de regularización C para una SVM de kernel lineal

Debido a que el puntaje de precisión para $C = 50$ es aproximadamente de 0.905 u.a. se consideró que este valor es lo suficientemente bueno para el problema a considerar ya que recordemos el conjunto de datos considera superpíxeles y no la imagen total por lo que se decidió no explorar más otros hiperparámetros relativos a la SVM de kernel lineal. Por lo que se procedió a calcular la curva ROC y su respectiva área bajo la curva del problema considerando un ajuste uno contra el resto, el recuento de falsos positivos y verdaderos positivos se hace de la misma manera. La curva ROC obtenida se muestra en la figura 16

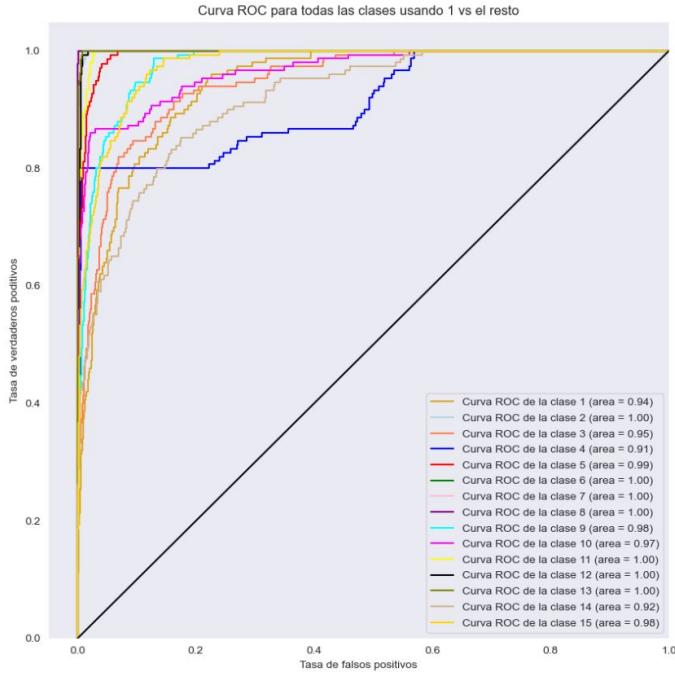


Fig. 16. Curvas ROC para todas las clases empleando un ajuste uno contra el resto. El área bajo cada curva ROC se muestra en la esquina inferior derecha de la imagen

Finalmente, se graficó la matriz de confusión y se reprodujo el reporte de clasificación. Ambos se muestran en las figuras 17 y 18 respectivamente.

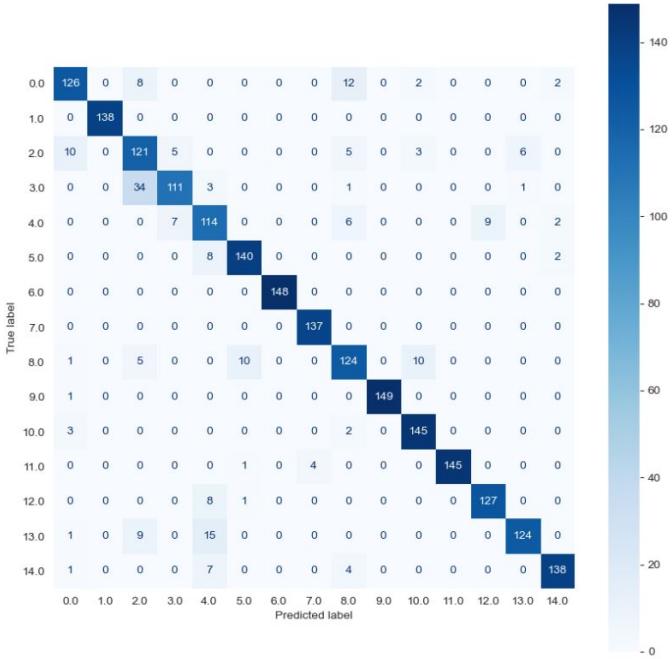


Fig. 17. Matriz de confusión para los datos de prueba de una SVM con kernel lineal y parámetro de regularización $C = 50$

	precision	recall	f1-score	support
clase 0	0.88	0.84	0.86	150
clase 1	1.00	1.00	1.00	138
clase 2	0.68	0.81	0.74	150
clase 3	0.90	0.74	0.81	150
clase 4	0.74	0.83	0.78	138
clase 5	0.92	0.93	0.93	150
clase 6	1.00	1.00	1.00	148
clase 7	0.97	1.00	0.99	137
clase 8	0.81	0.83	0.82	150
clase 9	1.00	0.99	1.00	150
clase 10	0.91	0.97	0.94	150
clase 11	1.00	0.97	0.98	150
clase 12	0.93	0.93	0.93	136
clase 13	0.95	0.83	0.89	149
clase 14	0.96	0.92	0.94	150
accuracy			0.90	2196
macro avg	0.91	0.91	0.91	2196
weighted avg	0.91	0.90	0.91	2196

Fig. 18. Reporte de clasificación para los datos de prueba de una SVM con kernel lineal y parámetro de regularización $C = 50$

4.4 Clasificación On-Line

Debido a que en la sección anterior se reducieron las imágenes a un conjunto de datos no es posible llevar un conteo de que imágenes en particular son problemáticas por lo que se desarrolló esta parte. El objetivo de este desarrollo consiste en entrenar una SVM con kernel lineal y parámetro $C = 50$ con los datos de entrenamiento (archivo .csv) tal y como se hizo en la sección anterior pero cambiando el modelo de clasificación por uno tal que una vez entrenada la SVM se le pasara las mismas imágenes de prueba y se decidiera la clase a la que corresponde y escribiera sobre la misma imagen la clase predicha. Al pasar cada imagen se calcula *in situ* sus propiedades de textura a partir de sus superpíxeles con los mismos parámetros que antes. Esta imagen reescrita se guardaba en una carpeta que contenía la clase real sin importar si el clasificador predecía una clase diferente. El método de predicción es muy sencillo y consiste en a partir de todas las propiedades de textura así como en los momentos de Hu y el área se guarden en una lista las predicciones de la SVM de cada superpíxel que corresponde a dicha imagen. Una vez termine de analizar todos los superpíxeles se crea una función de decisión que de acuerdo a las clases predichas almacenadas en la lista clasifique la imagen como aquella clase que se repite más en la lista, esto es sencillo de realizar al calcular la moda de la lista. Lamentablemente para este proceso no fue posible obtener curvas ROC ni un reporte de clasificación ni una matriz de confusión ya que la revisión de las imágenes se hace manualmente. Dado que sabemos que la carpeta debe contener una sola clase y conocemos que clase es basta con abrir las imágenes y observar el texto que se escribió en la clasificación. De esta manera es posible contar los errores por clase y proveer un puntaje de precisión. Dos ejemplos de estas imágenes con texto se muestran en las figuras 19 y 20. Mientras que el reporte realizado manualmente se muestra en la tabla 1.

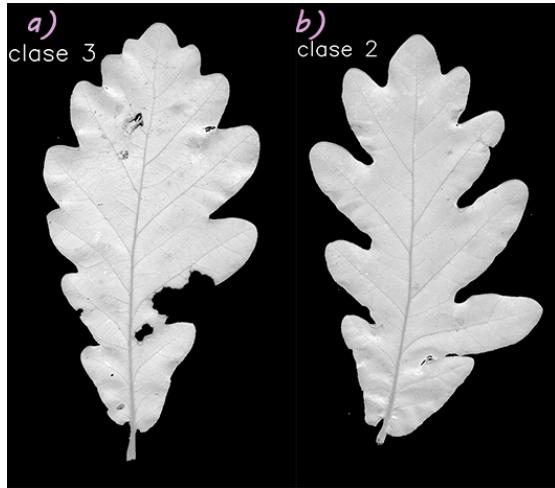


Fig. 19. Imágenes de clasificación en situ. a) Imagen con una correcta clasificación b) Imagen con una clasificación incorrecta

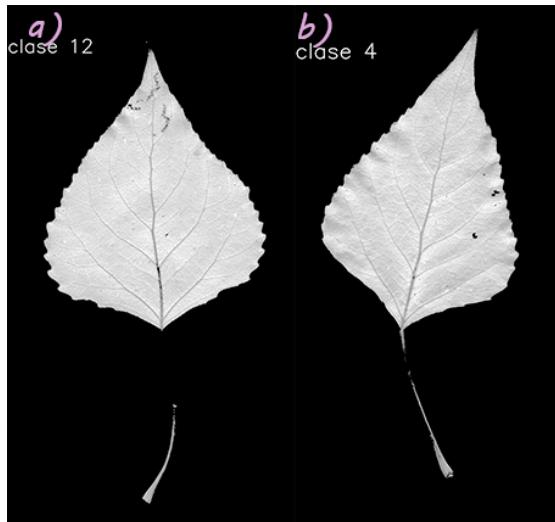


Fig. 20. Imágenes de clasificación en situ. a) Imagen con una correcta clasificación b) Imagen con una clasificación incorrecta

Clase	Errores cometidos
0	2
1	0
2	4
3	5
4	2
5	1
6	0
7	0
8	2
9	0
10	1
11	1
12	2
13	3
14	3
Puntaje de precisión Total	0.88

TABLE 1

Errores cometidos por cada clase. Total: 26 errores. Cada clase contiene 15 imágenes por lo que hay 225 imágenes en total

5 ANÁLISIS Y DISCUSIÓN

Se observó que las imágenes de segmentación y las imágenes binarias contenían puntos que no correspondían a zonas de las hojas. Sin embargo, estos puntos eran una cantidad muy pequeña al igual que sus dimensiones. Observaciones sobre los superpíxeles demostraron que el algoritmo SLIC consideraba estos puntos como un superpíxeles, por lo que decimos que nuestro conjunto de datos contiene ruido proveniente de estas zonas. Este problema no pudo resolverse ya que me fue imposible detectar y eliminar dichas zonas.

Considerar una cantidad mayor de superpíxeles, a saber 40 por imagen, si mejoraba el valor del puntaje de precisión en aproximadamente 0.02 u.a. a cambio de tener que esperar aproximadamente una hora y media para obtener únicamente los superpíxeles. Esta ganancia no fue relevante y se optó por un tiempo de computo menor empleando únicamente 10 superpíxeles. El caso contrario es similar, se encontró que el puntaje de precisión disminuía en aproximadamente 0.02 u.a al considerar solo 4 superpíxeles. Debido a que el tiempo de computo para 10 superpíxeles es razonable y la sensibilidad de las propiedades de textura en cuanto a la escala se decidió mantener un total de 10 superpíxeles. Como puede observarse en algunas clases la hoja presenta un tallo, que en algunas ocasiones el algoritmo de segmentación conectaba completamente con la hoja y en otras se consideraba como un objeto por separado, para ambos casos, visualmente, el tallo no presenta una gran variación, en cuanto a textura se refiere, para todas las clases, siendo este factor otra variable más que puede considerarse como ruido dentro de las propiedades de textura. Para el primer caso el ruido se ve disminuido debido a que dichas propiedades se ponderan al compartir espacio con zonas de la hoja que si son relevantes para las propiedades de textura. Para el segundo caso el ruido aportado por partes del tallo aislado es considerable debido a estas razones. La eliminación del tallo puede considerarse para solventar estos problemas, sin embargo, debe considerarse cuidadosamente ya que este objeto da un grado de caracterización al considerar el primer momento de Hu, ya que este momento es análogo al momento de inercia, que, en caso de retirar el tallo, el momento de inercia de toda la imagen se vería afectado.

El área obtenida por el histograma al aplicar el operador LBP es un artilugio matemático sin un fundamento sólido y se cree que que debe encontrarse una mejor característica que describa la textura a partir del histograma provisto por el mismo operador. Estudios realizados al considerar esta característica mostraron que el puntaje de precisión aumentaba en una cantidad considerablemente pequeña debido a esta característica. Si bien es cierto que considerar el área reduce drásticamente la dimensionalidad parece que esta característica no es suficientemente descriptiva como para mejorar el puntaje de precisión de la SVM.

Las transformaciones definidas por las ecuaciones (21) y (22) se tomaron con base en conocimiento a priori y se probó que éstas proveen un mejor puntaje de precisión. Para llegar a probar esto fue necesario ajustar manualmente las transformaciones según los rangos de cada característica. De esta manera es posible que puedan encontrarse transformaciones que separen de una mejor manera las características.

El análisis exploratorio de los datos reveló que el análisis de componentes principales no es útil para nuestro conjunto de datos, al menos para la reducción a dos y tres dimensiones ya que las gráficas por separado en tres dimensiones, la gráficas de dispersión, bivariada y univariada así como el mapa de calor muestran que algunas de las características se encuentran

muy correlacionadas, este hecho da un buen indicador de la posibilidad de aplicar PCA, sin embargo es necesario un buen análisis para seleccionar las características que pueden reducirse. Debido a la gran cantidad de tiempo que puede llevarse realizar este análisis se decidió considerar todas las características.

Al observar las gráficas de dispersión y el mapa de calor uno esperaría que un kernel no lineal obtendría mejores resultados comparados a su contraparte lineal, sin embargo, esto no fue así considerando únicamente el puntaje de precisión. Si bien es cierto que para esta prueba los distintos hiperparámetros de las SVM no se variaron puede estudiarse también una variación de los mismos para seleccionar de una mejor manera el kernel de la SVM pero al igual que los otros casos esto no fue posible realizar debido a la gran cantidad de tiempo que pudiese tomar por lo que se consideró el puntaje de precisión como el único indicador del desempeño global de las SVM para seleccionar el mejor kernel.

La variación de la constante de regularización C mejora el puntaje de precisión en aproximadamente 0.05 u.a. una mejora, que como ya se discutió no se considera muy relevante. A pesar de ello se consideró la constante C que fuera óptima en cuanto a tiempo de computo y puntaje de precisión.

Como ya se mencionó existe la posibilidad de estudiar otros hiperparámetros sin embargo el puntaje de precisión se consideró lo suficientemente bueno como para no seguir ésta linea. Las áreas obtenidas bajo las curvas ROC para cada clase revelan que el test de clasificación se encuentra en los rangos muy bueno a excelente a pesar de no haber seguido aquellas líneas que requerían una investigación mayor. También podemos obtener que la clase más difícil de detectar es la clase 3. Al observar las imágenes de dicha clase y compararlas con el resto del conjunto de datos se encuentra que la textura aparente es muy similar. Esta clase presenta una forma muy particular con respecto a las otras por lo que debido a que los momentos de Hu capturan propiedades de forma podemos decir que existe una variación significativa de la misma en cada imagen, este hecho se confirma a través de un análisis visual. Seguido de esta clase encontramos la clase 13 que presenta las mismas variaciones de forma. A pesar de dichas variaciones el test es muy bueno para cada clase.

La visualización de la matriz de confusión se interpreta como un mapa de calor donde un buen clasificador concentrará la mayor energía en su diagonal (tasa de verdaderos positivos) y un clasificador perfecto consistiría en todos aquellos elementos fuera de la diagonal con un valor de energía de cero. Para nuestro caso se observa que la diagonal contiene una densidad de energía alta (valores mayores a 120) y los elementos fuera de la diagonal tienen valores bajos de energía salvo quizás la clase real 3 que contiene una cantidad significativa de errores al clasificar las imágenes de dicha clase como clase 2. Este hecho ya lo conocímos al inspeccionar la curva ROC de dicha clase, pero la matriz de confusión aporta más información ya que podemos conocer la similitud entre clases. El reporte de clasificación complementa la matriz de confusión y encuentra que el promedio tanto del puntaje de precisión, recall y f1 es de 0.91 u.a., siendo 1.0 u.a. un clasificador perfecto. De esta manera consideramos nuestro modelo como muy bueno.

Finalmente, para la clasificación in situ obtenemos que la clase con mayores errores es la clase 3, tal y como lo fue en la clasificación a través de archivos .csv, seguida de la clase 2. Para esta parte se encuentra que los resultados de los errores difieren ligeramente con respecto al caso anterior ya que se agregó una restricción más que consiste en considerar la moda para la clasificación de la imagen. Dado que este caso es más restrictivo

se espera que los puntajes disminuyan su valor. Este hecho se prueba al calcular, manualmente, el puntaje de precisión, cuyo valor fue de 0.88 u.a. que se sigue considerando bastante bueno.

6 CONCLUSIONES

Las características de textura, los momentos de Hu y el área truncada obtenida a partir del operador LBP consideradas en este trabajo y el entrenamiento de una máquina de soporte vectorial con kernel lineal y parámetro de regularización $C = 50$ sobre dichas características produjo un clasificador muy bueno con puntajes promedio de precisión de 0.91 u.a., recall 0.90 u.a y f1 de 0.91 u.a. Se graficaron las curvas ROC de cada clase usando el método uno contra el resto obteniéndose un área mínima debajo de la misma de 0.91 u.a. para la clase 3. El ajuste del parámetro $C = 50$ condujo a la obtención de dichos resultados. Se realizó clasificación on-line de las imágenes agregando una restricción de clasificación. Para esta clasificación se obtiene un puntaje de precisión de 0.88 u.a. resultando, nuevamente, la clase 3 como la clase con mayor error. A pesar de ello se considera, también, que el clasificador es bueno.

El análisis exploratorio de los datos da buenos indicios para la reducción de características, ya que se encontró que algunas de ellas estaban muy correlacionadas entre sí. Sin embargo esta línea de investigación no se siguió. Si bien es cierto existieron una cantidad considerable de casos que no se estudiaron los resultados obtenidos fueron muy buenos y estos análisis pueden ayudar a futuros mejoramientos del clasificador.

Finalmente, las características empleadas en este trabajo son muy sencillas de calcular y las más conocidas, sin embargo, existen una gran cantidad de técnicas que pueden complementar los datos.

REFERENCES

- [1] Doulamis, N., Doulamis, A., Protopapadakis, E. et al., *Deep Learning for Computer Vision: A Brief Review*. Computational Intelligence and Neuroscience, Special Issue (2018).
- [2] Princeton University, *Machine Learning in Computer Vision*. Página Web <https://www.cs.princeton.edu/courses/archive/spring07/cos424/lectures/li-guest-lecture.pdf>. Consultado el 28.02.20.
- [3] Olsen, A., Konovalov, D.A., Philippa, B. et al., *DeepWeeds: A Multi-class Weed Species Image Dataset for Deep Learning*. Sci Rep 9, 2058 (2019).
- [4] Bargoti, S. And Underwood, J., *Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards*. Journal of Field Robotics 6, 34 (2017).
- [5] Mohanty, S., Hughes, D And Salathe, M. *Using Deep Learning for Image-Based Plant Disease Detection*. Front. Plant Sci., 22 September 2016.
- [6] Z. Fan, J. Lu, M. Gong, H. Xie and E. D. Goodman. *Automatic Tobacco Plant Detection in UAV Images via Deep Neural Networks*. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 3, 11 (2018).
- [7] Linkoping University (CVL), *Swedish Leaf Dataset*. Página Web <https://www.cvl.isy.liu.se/en/research/datasets/swedish-leaf/>. Consultado el 28.02.20.
- [8] M. Nixon, A. Aguado *Feature Extraction and Image Processing for Computer Vision*, 3rd ed. Elsevier Academic Press, 2012.
- [9] P. Brodatz, *A Photographic Album for Artist and Designers*. Reinhold, USA.
- [10] K. Karru, A. Jain, R. Bolle *Is there any texture in an image?*, Pattern Recog. 29 (9), 1437-1446.
- [11] A. Humeau-Heurtier *Texture Feature Extraction Methods: A Survey*, IEEE Access, 7, 8975-9000, 2019.
- [12] M. Nixon, A. Aguado *Feature Extraction and Image Processing for Computer Vision*, Elsevier, 3ra Edición, 2012.
- [13] T. Fletcher Notas: *Support Vector Machines Explained*, UCL University, 2008. https://clug.cs.uwo.ca/cs860/papers/SVM_Explained.pdf. Consultado el 07.06.20

- [14] A. Anónimo *Superpixel: Empirical Studies and Applications*, Chigago University, sin año. <https://ttic.uchicago.edu/~xren/research/superpixel/>. Consultado el 07.06.20
- [15] D. Stutz, A. Hermans, B. Leibe *Superpixels: An Evaluation of the State-of-the-Art*, Computer Vision and Image Understanding, 2017.
- [16] J. Darshita *Superpixels and SLIC*, Medium, 2019. <https://medium.com/@darshita1405/superpixels-and-slic-6b2d8a6e4f08>. Consultado el 26.04.20
- [17] wikipedia. Enciclopedia *Image moment*, Wikipedia, 2020. https://en.wikipedia.org/wiki/Image_moment. Consultado el 28.06.20