Predicción taxonómica de muestras de microplancton usando técnicas de Aprendizaje Automático*

Pablo González[a], Jorge Díez[a]
Juan José del Coz[a]
Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
{pgonzalez,jdiez,juanjo}@aic.uniovi.es

Eva Álvarez[b], Rafael González-Quirós[b]
Enrique Nogueira[b], Angel López-Urrutia[b]
Centro Oceanográfico de Gijón
Instituto Español de Oceanografía
{eva.alvarez,rgq,enrique.nogueira,alop}@gi.ieo.es

Resumen

A la hora de realizar un estudio biológico del plancton marino es muy importante analizar la distribución de los diferentes organismos presentes en el mismo. Debido a la aparición en el mercado de nuevos instrumentos de monitorización de organismos en el plancton, capaces de captar y segmentar sus imágenes de forma automática, es imposible analizar manualmente toda esa información. En estas circunstancias, sería de gran ayuda contar con un sistema de clasificación automática de estos organismos, que fuese capaz, a su vez, de estimar la concentración de biomasa de los diferentes grupos taxónomicos presentes. Con el fin de construir un clasificador automático que cubra estas necesidades, usaremos Máquinas de Vectores Soporte con un conjunto de datos formado por 5145 imágenes pertenecientes a 5 tipos de especies de microplacton. Dado que nuestro objetivo es predecir la cantidad total de biomasa para cada una de esas clases, hemos desarrollado un método de clasificación sensible al coste que consigue tasas de acierto cercanas al 94 % en términos de biomasa.

1. Introducción

Debido a su importancia científica, se están dedicando muchos esfuerzos al estudio de la clasificación automática del plancton. Por un lado, el plancton representa el nivel más bajo de la cadena alimenticia que sostiene la vida en los océanos. Por otro lado, los ecosistemas de plancton desempeñan un rol importante en varios ciclos bioquímicos, incluyendo el ciclo del carbono. Con el objetivo de hacer este proceso automático o semiautomático, necesitamos al menos tres elementos básicos. Primero, un aparato que sea capaz de muestrear el plancton automáticamente, obteniendo imágenes digitales con alta resolución. En este campo, ha habido bastantes avances en los últimos años que han dado lugar al desarrollo de dispositivos tales como el Video Plankton Recorder [2] o la FlowCam [13]. Ésta última ha sido empleada en este estudio.

En segundo lugar, se necesita extraer de las imágenes aquellos atributos importantes para poder clasificar correctamente los organismos. El análisis de imágenes digitales ha sido y es hoy en día, una disciplina muy estudiada. Existe una gran cantidad de literatura sobre el tema y varias de las técnicas expuestas en la misma son aplicables a nuestro problema. En este caso, nos hemos centrado en el análisis de la forma, la textura y el color. Usando estos métodos, hemos obtenido 170 atributos de cada una de las imágenes analizadas.

En tercer y último lugar, se necesita un algoritmo de clasificación para predecir la clase de cada ejemplo. También aquí tenemos varias alternativas, tales como árboles de decisión o redes neuronales, entre otras. En este trabajo, nos hemos decantado por las Máquinas de

*Este trabajo ha sido financiado por el proyecto TIN2008-06247 del Ministerio de Ciencia e Innovación y por el proyecto B09-059-C2 de la FICYT (Asturias).

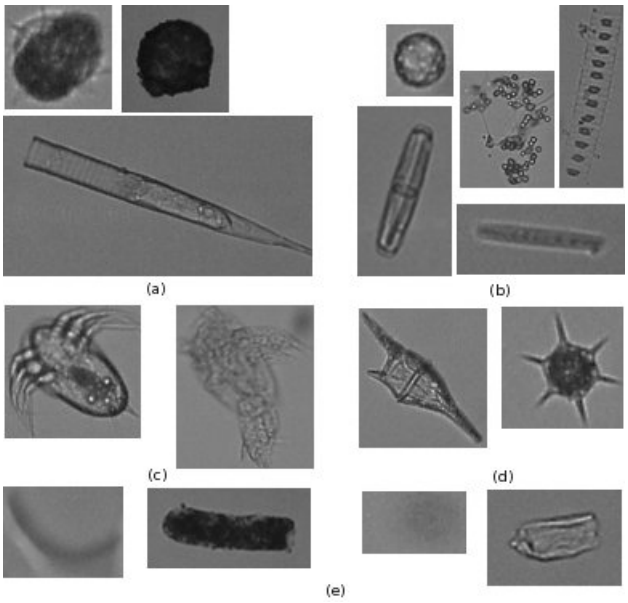

Figura 1: Ejempo de imágenes pertenecientes a 5 clases de microplancton. (a) Ciliados; (b) Diatomeas; (c) Crustáceos; (d) Flagelados; (e) Otros

Vectores Soporte (SVM), un conocido algoritmo de clasificación propuesto por Vapnik [16].

Como explicaremos más adelante, la formulación original del SVM fue diseñada para resolver problemas binarios. Posteriormente aparecieron nuevas aproximaciones con el objetivo de construir clasificadores que pudiese manejar un número arbitrario de clases. Principalmente existen dos maneras de conseguir que las SVM sean capaces de resolver problemas multi-clase. La primera alternativa considera todos los datos como un único problema de optimización [1, 17], mientras que la segunda descompone el problema original en una serie de problemas binarios. Éste es el enfoque utilizado por algoritmos como: *one-versus-all* (OVA) [16], *one-versus-one* (ovo) [8], o DDAG, que usa grafos acíclicos dirigidos [12]. Dado que ninguno de los métodos anteriores ha probado ser mejor que el resto, éste sigue siendo un tema abierto para el estudio. En este artículo usaremos dos de los métodos nombrados anteriormente: el algoritmo *one-versus-one* [8] y la propuesta de Crammer y Singer [1], extendidos en ambos casos para que tengan en cuenta el coste que supone clasificar incorrectamente cada ejemplo. Ese coste vendrá dado por su biomasa.

Es obvio que conseguir una tasa de acierto más elevada en la clasificación de los ejemplos conllevará, a su vez, una predicción más precisa de la concentración de biomasa por cada clase. Sin embargo, nuestro clasificador no será capaz de tener en cuenta el coste de cada clasificación errónea. Parece claro que el fallo de un ejemplo con una cantidad importante de biomasa, afectará al sistema en un mayor grado que el fallo de un ejemplo con una biomasa menor. Ésta es la razón por la que hemos desarrollado una extensión para dos implementaciones de SVM diferentes, con el fin de hacer que sean sensibles al coste. Esta forma de aprendizaje se conoce como *aprendizaje sensible al coste* (cost-sensitive learning)[3]. La idea principal radica en intentar que el algoritmo aprenda a clasificar mejor los ejemplos con

un coste más alto. Procediendo de esa forma, el clasificador será capaz de aproximar mejor el volumen de biomasa de cada clase.

En la siguiente sección describiremos el conjunto de datos utilizado. En la Sección 3 se presentará formalmente el aprendizaje sensible al coste y posteriormente discutiremos los algoritmos de aprendizaje utilizados. Después, en la Sección 5, mostraremos los resultados experimentales que hemos obtenido y finalizaremos presentando algunas conclusiones.

2. Datos

2.1. Conjunto de datos

La FlowCam [13] es un instrumento que permite la monitorización de partículas en un fluido. Combinando la citometría de flujo y el uso de un microscopio, es capaz de contar y analizar automáticamente los organismos de microplancton existentes en un fluido continuo. El sistema captura una imagen de cada organismo por separado, obteniendo de esta manera, un conjunto de imágenes a partir de una muestra de fluido (véase Figura 1).

Nuestro conjunto de datos fue obtenido a partir de muestras extraídas en el mar Cantábrico. Una vez que las muestras fueron procesadas por la FlowCam, un experto taxónomo clasificó las imágenes obtenidas, separándolas en las 5 categorías de microplacton relevantes para nuestro estudio: Ciliados, Diatomeas, Crustáceos, Flagelados y una categoría Otros, formada por los ejemplos de otras especies, incluyendo los no vivos. De todo este proceso se obtuvo un conjunto de datos formado por 5145 imágenes. La Figura 2 muestra la distribución de ejemplos por cada clase.

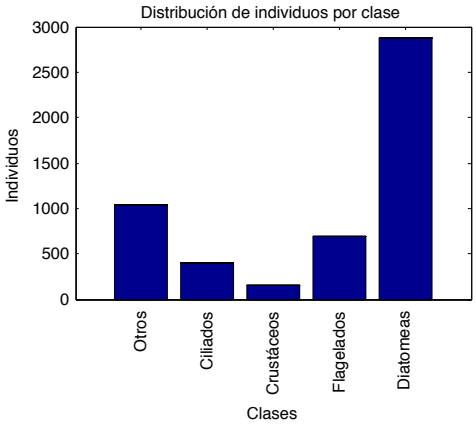

Figura 2: Ejemplos por clase

2.2. Extracción de atributos

Con el objetivo de obtener un vector de atributos de cada una de las imágenes, hemos aplicado diferentes métodos pertenecientes al campo del análisis de imágenes. No hemos centrado nuestros esfuerzos en ningún método en particular, ya que hemos preferido aplicar el enfoque de combinar diferentes técnicas para construir un vector de características lo más amplio posible. Esperamos que cada una de estas técnicas aplicadas sea capaz de representar una parte de las características de la imagen. Las técnicas utilizadas van desde los tradicionales métodos de análisis de formas, como los descriptores de Fourier, combinados con atributos de textura, como las matrices de co-ocurrencia, hasta información acerca del color de la imagen. En esta sección vamos a intentar resumir las principales técnicas aplicadas en la extracción de atributos.

En primer lugar, hemos usado los descriptores de Fourier elípticos (EDF) propuestos por Kuhl and Giardina [9]. Estos descriptores son capaces de delinear cualquier tipo de forma con un contorno bidimensional y han sido aplicados con éxito en la evaluación de diferentes formas biológicas en animales y plantas. Para calcularlos se utilizan los coeficientes de Fourier cuando la función que representa el contorno es aproximada por la Serie de Fourier. Después de algunos experimentos, decidimos que 15 armónicos es un número adecuado para describir, con suficiente precisión, la forma de los organismos que estamos estudiando.

En segundo lugar, hemos trabajado con los momentos de las imágenes. Los momentos son atributos estadísticos ampliamente usados en estudios morfológicos de imágenes. Son bastante estables respecto al ruido y no dependen

excesivamente del contorno de la imagen. El cálculo de los mismos se realiza a partir de las intensidades de los píxeles de las imágenes. Con el fin de que estos momentos sean invariantes ante un movimiento de translación del objeto dentro de la imagen, se usan los momentos centrales, calculados a partir del centro de gravedad del objeto. A partir de estos momentos centrales, Hu definió 7 características, denominadas momentos de Hu [7] . Los momentos de Hu son invariantes a la traslación, rotación y escalamiento. Esto quiere decir que dos regiones que tengan la misma forma pero que sean de distinto tamaño y que estén ubicadas en posiciones y orientaciones distintas en la imagen, tendrán momentos de Hu iguales.

En tercer lugar, hemos calculado los momentos de Zernike que están basados en la teoría de los polinomios ortogonales y fueron propuestos por Teague [15]. Los momentos de Zernike tienen muy buenas propiedades cuando hablamos de la tolerancia al ruido, el manejo de información redundante, la rotación y las capacidades de reconstrucción de la imagen original. Es importante comentar que una de las principales desventajas de los momentos en general es que estudian las imágenes de manera global y no local, por lo que no funcionan bien cuando el objeto de la imagen aparece parcialmente obstruido. En nuestro caso hemos calculado 49 momentos de Zernike usando el centro de gravedad de los objetos para asegurar que sean invariantes ante la translación.

Para finalizar con los atributos de forma, se ha realizado un análisis granulométrico [10] de las imágenes, que nos permite extraer información de un orden mayor, para complementar la información de bajo orden proporcionada por los momentos de Hu. Estudios anteriores sobre el reconocimiento automático de imágenes de plancton llevados a cabo por Tang [14], dieron como resultado que este tipo de atributos eran muy útiles para la clasificación de las imágenes. De esta manera, hemos calculado 8 atributos granulométricos para cada imagen.

Para completar esta sección, hablaremos de los atributos de textura. En primer lugar, hemos utilizado las matrices de co-ocurrencia, utilizadas ampliamente en el estudio de la textura en imágenes. Se construyen asignando a cada elemento $[i, j]$ el número de veces que un píxel con valor i es adyacente a un píxel con valor j. Cuando manejamos una escala de niveles de gris grande, estas matrices son muy grandes y dispersas, lo que hace que sean más difíciles de usar como atributos directamente. Debido a esto, se obtienen diferentes medidas a partir de ellas para conseguir un conjunto de atributos más útil. El conjunto de atributos más conocido cuando hablamos de matrices de co-ocurrencia son los atributos de Haralick [5], que han sido aplicados en numerosos análisis de imágenes digitales con buenos resultados.

Otro método para analizar la textura es mediante la transformada de Wavelet. Las Wavelet son un tipo de funciones que permiten descomponer una imagen de manera jerárquica, realizando al mismo tiempo un estudio de la imagen en diferentes resoluciones y escalas. Son capaces de representar la imagen desde una forma general, hasta sus pequeños detalles. En nuestro estudio hemos usado como función madre una Daubechies de orden 4. Hemos realizado un estudio con 4 escalas y 3 bandas de detalle para cada una de las mismas, obteniendo de esta manera 12 sub-bandas de detalle por imagen. De cada sub-banda se calculó su energía de la siguiente forma:

$$E_n^k = \frac{1}{N \times N} \sum_{i,j=1}^{N} (s_n^k(i,j))^2, \qquad (1)$$

donde, s_n^k es la sub-banda de detalle k, con escala n y tamaño $N \times N$.

Además de los métodos anteriores, hemos incluido las medidas morfológicas de las partículas que son calculadas directamente por la FlowCam [13], al mismo tiempo que obtiene las imágenes. Algunos ejemplos son el perímetro de la partícula, su área, las coordenadas de su centro geométrico, la distancia media del perímetro al centro, etc. En total, 26 atributos morfológicos fueron añadidos a nuestro vector de características, que junto a todas las anteriormente descritas, dieron como resultado final un conjunto de 170 atributos preparado para ser usado en el proceso de aprendizaje.

2.3. Cálculo de la biomasa

El contenido de carbono (al que nos referiremos como biomasa) de los organismos planctónicos es un parámetro fundamental a la hora de realizar modelos que representen ecosistemas. Debido a ello, se han realizado varios estudios con el objetivo de estimar la biomasa estableciendo una relación con el volumen del organismo. El volumen puede ser calculado (aproximadamente), a partir del diámetro de la partícula obtenido de las imágenes. En el estudio desarrollado por Menden [11], en el cual se estudia la relación entre biomasa y volumen, se proponen tres formas de realizar este cálculo dependiendo del volumen del ejemplo y de la clase de organismo con la que estemos trabajando. La ecuación 2 nos muestra cómo calcular la biomasa a partir de su volumen (V) para ejemplos con $V < 3000um^3$:

$$\log_{10} C = -0{,}583 + 0{,}86 \log_{10} V. \quad (2)$$

La ecuación siguiente se aplica para ejemplos que no pertenezcan a la clase de las Diatomeas y que cumplan que $V > 3000um^3$,

$$\log_{10} C = -0{,}665 + 0{,}939 \log_{10} V. \quad (3)$$

Por último, para Diatomeas con $V > 3000um^3$ se aplica:

$$\log_{10} C = -0{,}933 + 0{,}881 \log_{10} V. \quad (4)$$

Las fórmulas descritas nos permiten calcular de una manera sencilla la biomasa aproximada de cada ejemplo. La Figura 3 nos muestra (arriba) cómo se distribuye la biomasa entre las cinco clases de nuestro problema, y (abajo) la media de biomasa de los ejemplos de cada una de las clases. Podemos ver cómo los Crustáceos, aún no siendo relevantes en número, sí son muy importantes en términos de la biomasa, relegando a un segundo plano a las Diatomeas, la clase más abundante.

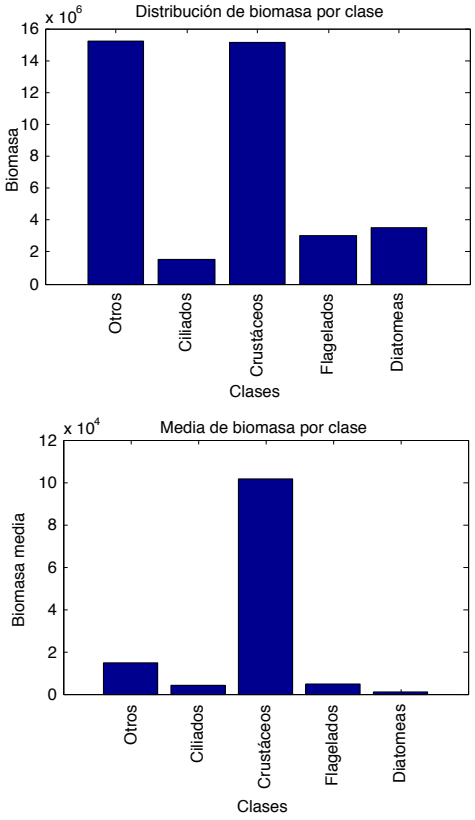

Figura 3: Distribución de biomasa por clase (arriba) y biomasa media por clase (abajo)

3. Aprendizaje sensible al coste

Sea $\mathcal{X}$ el espacio de entrada e $\mathcal{Y} = \{1, ..., k\}$ un conjunto finito de clases. Consideramos una tarea de clasificación multi-clase sensible al coste dada por el conjunto de entrenamiento $\mathcal{S} = \{(\boldsymbol{x}_1, y_1, c_1), \ldots, (\boldsymbol{x}_n, y_n, c_n)\}$, obtenido a partir de una distribución desconocida $Pr(\mathcal{X}, \mathcal{Y}, \mathbb{R}^+)$. En el aprendizaje sensible al coste, el valor positivo c_i asociado con cada ejemplo $\boldsymbol{x}_i$ representa el coste de clasificar incorrectamente dicho ejemplo. En nuestra aplicación, c_i será la biomasa del ejemplo $\boldsymbol{x}_i$. Indicar que en otras aproximaciones sensibles al coste, ver por ejemplo [3], los costes están asociados a las clases y no a los ejemplos. En cualquier caso, quedan englobadas dentro del marco aquí descrito.

En este contexto, el objetivo de estas tareas de aprendizaje es encontrar una hipótesis h del espacio de entrada al de salida, en

símbolos, $h : \mathcal{X} \longrightarrow \mathcal{Y}$, que optimice el *riesgo esperado* en muestras $\mathcal{S}'$ independientes e idénticamente distribuidas de acuerdo con la distribución $Pr(\mathcal{X}, \mathcal{Y}, \mathbb{R}^+)$. Esto se representa habitualmente por $\Delta(h, \mathcal{S}')$, y en el caso del aprendizaje sensible al coste se calcula aplicando la expresión

$$\Delta(h, \mathcal{S}') = \frac{1}{\sum\limits_{\boldsymbol{x}_i \in \mathcal{S}'} c_i} \sum_{\boldsymbol{x}_i \in \mathcal{S}'} \delta(h(\boldsymbol{x}_i), y_i, c_i), \quad (5)$$

donde $\delta(h(\boldsymbol{x}), y, c)$ es la función que mide la pérdida debido a la predicción $h(\boldsymbol{x})$ cuando la clase real es y y el coste de dicho error es c. En este tipo de aprendizaje se desea favorecer las decisiones correctas de h sobre ejemplos con un coste más alto.

En este artículo proponemos una medida, que llamaremos *Global Biomass Loss*, para estimar el error en biomasa. Sigue la forma general de las funciones de pérdida en el aprendizaje sensible al coste, esto es,

$$\delta_{GB}(h(\boldsymbol{x}_i), y_i, c_i) = c_i \left[h(\boldsymbol{x}_i) \neq y_i\right], \quad (6)$$

donde [ ] es 1 cuando el predicado interno es cierto y 0 en otro caso. Aplicando esta expresión en (5), podemos obtener la ecuación para Δ_{GB} que considera los ejemplos incorrectamente clasificados por h:

$$\Delta_{GB}(h, \mathcal{S}') = \frac{\sum\limits_{\boldsymbol{x}_i \in \mathcal{S}'} c_i [h(\boldsymbol{x}_i) \neq y_i]}{\sum\limits_{\boldsymbol{x}_i \in \mathcal{S}'} c_i}. \quad (7)$$

Δ_{GB} mide la proporción de biomasa incorrectamente clasificada. Uno de los métodos de aprendizaje descritos en la Sección 4.2 optimiza esta función de pérdida.

4. Métodos de aprendizaje

Para abordar la aplicación descrita en el artículo, aplicaremos dos tipos de métodos de aprendizaje. Por un lado, clasificadores multiclase conocidos basados en Máquinas de Vectores Soporte, y por otro lado, extenderemos dichos métodos haciéndolos sensibles al coste. El resultado esperado es que estos últimos mejoren el rendimiento de los primeros cuando la medida de rendimiento considerada sea Δ_{GB}.

4.1. Algoritmos de clasificación multiclase

Como indicamos en la Introducción, existen dos tipos de aproximaciones para tratar los problemas de clasificación multi-clase usando SVM. Una se basa en descomponer el problema original en un conjunto de SVM binarios, mientras que la segunda considera todos los datos en un único problema de optimización. En este estudio aplicaremos un método de cada una de las dos clases. Entre los primeros, seleccionamos el algoritmo *one-versus-one* (OVO) [8] dado que obtiene mejores resultados experimentales [6], y del segundo grupo, el método propuesto por Crammer y Singer [1] porque puede implementarse más eficientemente.

En el método *one-versus-one* se definen $k(k-1)/2$ problemas binarios, donde el problema *l-versus-m* consiste en separar los subconjuntos de ejemplos $\mathcal{S}_l$ y $\mathcal{S}_m$ de las clases l y m respectivamente. Empleando SVMs binarios, OVO resuelve, para cada par de clases, el siguiente problema de optimización[1]:

$$\begin{aligned} \min_{\boldsymbol{w}_{lm}, \boldsymbol{\xi}^{lm}} \quad & \frac{1}{2}\langle \boldsymbol{w}_{lm}, \boldsymbol{w}_{lm} \rangle + C \sum_{y_i \in \{l,m\}} \xi_i^{lm}, \quad (8) \\ \text{s.t.} \quad & \langle \boldsymbol{w}_{lm}, \boldsymbol{x}_i \rangle \geq +1 - \xi_i^{lm}, \quad \text{if } \boldsymbol{x}_i \in \mathcal{S}_l, \\ & \langle \boldsymbol{w}_{lm}, \boldsymbol{x}_i \rangle \leq -1 + \xi_i^{lm}, \quad \text{if } \boldsymbol{x}_i \in \mathcal{S}_m, \\ & \xi_i^{lm} \geq 0, \quad \forall \boldsymbol{x}_i \in \mathcal{S}_l \cup \mathcal{S}_m, \end{aligned}$$

donde la constante C controla la cantidad de regularización y ξ_i son las variables de holgura empleadas por las formulaciones de margen blando para evitar el sobreajuste. La salida de cada modelo inducido $\boldsymbol{w}_{lm}$ para un ejemplo $\boldsymbol{x}$ se cuenta como un voto para la clase predicha l o m; finalmente, la clase más votada será retornada como predicción.

En el caso de la formulación propuesta por Crammer y Singer, se induce un modelo $\boldsymbol{w}_l$ para cada clase l, siguiendo una aproximación *one-versus-all*. Sin embargo, en este caso todos ellos $\{\boldsymbol{w}_l : l = 1, \ldots, k\}$ son aprendidos al

[1] Para facilitar la lectura, se han omitido en todos los casos los términos independientes b. Podrían incluirse fácilmente añadiendo un atributo adicional constante a cada ejemplo $\boldsymbol{x}_i$

mismo tiempo,

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\sum_{l=1}^{k}\langle \boldsymbol{w}_l, \boldsymbol{w}_l\rangle + C\sum_{i=1}^{n}\xi_i, \qquad (9)\\ \text{s.t.} \quad & (\langle \boldsymbol{w}_{y_i}, \boldsymbol{x}_i\rangle - \langle \boldsymbol{w}_r, \boldsymbol{x}_i\rangle) \geq e_i^r - \xi_i,\\ & \forall i = 1,\ldots,n \quad \forall r \in \{1,\ldots,k\}, \end{aligned}$$

donde e_i^r es 1 cuando $r \neq y_i$ y 0 en otro caso. En este problema, no es necesario incluir una restricción adicional para asegurar que las variables de holgura son no negativas, ya que lo garantiza la restricción de cada ejemplo cuando $r = y_i$. Es importante recalcar que el número de restricciones de este problema de optimización puede ser muy grande, especialmente para problemas con muchas clases. Sin embargo, la eficiencia se consigue gracias a que la mayor parte de las restricciones están inactivas, dado que el conjunto de restricciones de cada ejemplos $\boldsymbol{x}_i$ comparte una única variable de holgura ξ_i. De hecho, solamente una de ellas estará activa: la correspondiente a $r = y_i$ cuando el ejemplo este bien clasificado con el suficiente margen ($\xi_i = 0$), o la de la clase predicha incorrectamente o que no se encuentra separada de la verdadera con el suficiente margen ($\xi_i > 0$).

Finalmente, la clase retornada por el algoritmo será determinada siguiendo el esquema *winners-takes-all*:

$$h(\boldsymbol{x}_i) = \operatorname*{argmax}_{l\in\{1,\ldots,k\}} \langle \boldsymbol{w}_l, \boldsymbol{x}_i\rangle.$$

La principal ventaja de esta aproximación sobre la anterior es que se puede optimizar de manera concreta una función de pérdida. En nuestro caso, esto resulta particularmente interesante dado que podremos optimizar directamente la función Δ_{GB} (7).

4.2. Algoritmos sensibles al coste

Los métodos de aprendizaje descritos en la sección anterior, pueden extenderse fácilmente al paradigma del aprendizaje sensible al coste. De hecho, la versión sensible al coste del algoritmo OVO fue presentada en [4]. El problema de optimización es prácticamente idéntico al de la ecuación (8), incluyendo el coste c_i de fallar cada ejemplo:

$$\begin{aligned} \min_{\boldsymbol{w}_{lm},\boldsymbol{\xi}^{lm}} \quad & \frac{1}{2}\langle \boldsymbol{w}_{lm}, \boldsymbol{w}_{lm}\rangle + C\sum_{y_i\in\{l,m\}} c_i\xi_i^{lm}, \quad (10)\\ \text{s.t.} \quad & \langle \boldsymbol{w}_{lm}, \boldsymbol{x}_i\rangle \geq +1 - \xi_i^{lm}, \quad \text{if } \boldsymbol{x}_i \in \mathcal{S}_l,\\ & \langle \boldsymbol{w}_{lm}, \boldsymbol{x}_i\rangle \leq -1 + \xi_i^{lm}, \quad \text{if } \boldsymbol{x}_i \in \mathcal{S}_m,\\ & \xi_i^{lm} \geq 0, \quad \forall \boldsymbol{x}_i \in \mathcal{S}_l \cup \mathcal{S}_m, \end{aligned}$$

En la ecuación anterior podemos apreciar como el número de restricciones de ambos problemas es el mismo. El problema dual puede derivarse mediante las técnicas de Lagrange:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\sum_{\boldsymbol{x}_i,\boldsymbol{x}_j\in\mathcal{S}_l\cup\mathcal{S}_m} \alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle + \sum_{\boldsymbol{x}_i\in\mathcal{S}_l\cup\mathcal{S}_m} \alpha_i\\ \text{s.t.} \quad & 0 \leq \alpha_i \leq c_i C, \quad \forall \boldsymbol{x}_i \in \mathcal{S}_l \cup \mathcal{S}_m. \end{aligned}$$

Por tanto, la única diferencia respecto a un problema SVM binario está en el límite superior impuesto a las variables de holgura [16].

La principal desventaja de cualquier método de descomposición es que el modelo global aprendido, formado por el conjunto de modelos binarios, no ha sido inducido optimizando ninguna función de pérdida. En este artículo presentamos una extensión del método de Crammer y Singer que permite optimizar la función de error de la ecuación (7), asignando un coste a cada ejemplo. Es importante indicar, que esta aproximación incluye el caso en el que los costes se asocian con las clases en lugar de con los ejemplos. Para ello, sería suficiente asignar el mismo coste a todos los ejemplos de una clase. Hasta donde alcanza nuestro saber, este método nunca ha sido extendido antes. La formulación es bastante simple, añadiendo los costes a la función objetivo,

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\sum_{l=1}^{k}\langle \boldsymbol{w}_l, \boldsymbol{w}_l\rangle + C\sum_{i=1}^{n} c_i\xi_i, \qquad (11)\\ \text{s.t.} \quad & (\langle \boldsymbol{w}_{y_i}, \boldsymbol{x}_i\rangle - \langle \boldsymbol{w}_r, \boldsymbol{x}_i\rangle) \geq e_i^r - \xi_i,\\ & \forall i = 1,\ldots,n \quad \forall r \in \{1,\ldots,k\}. \end{aligned}$$

La consecuencia más importante, como se dijo anteriormente, es que ahora podemos controlar el error en biomasa durante la optimización. De hecho podría probarse que el segundo

término de la función objetivo constituye una cota superior del error calculado mediante (7).

A pesar de resultar en una derivación matemática más compleja que en el caso del método anterior, el problema dual puede derivarse usando las técnicas habituales. El problema de optimización dual resulta ser:

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{r=1}^{k}\alpha_i^r\alpha_j^r\langle \boldsymbol{x}_i,\boldsymbol{x}_j\rangle - \sum_{i=1}^{n}\sum_{r=1}^{k}e_i^r\alpha_i^r$$

$$\text{s.t. } \sum_{r=1}^{k}\alpha_i^r = 0, \quad \forall i = 1,\ldots,n,$$

$$\alpha_i^r \le c_i C, \quad i = 1,\ldots,n, r = 1,\ldots,k.$$

Resaltar que ahora tenemos por cada ejemplo un vector de variables duales, α_i^r. La suma de todos ellos debe ser 0 para cada ejemplo y ninguna de las variables individuales deben ser mayores que $c_i \cdot C$.

Implementamos ambos algoritmos[2], extendiendo el trabajo presentado en [6]. En el caso del último método, se trata de un algoritmo de tipo SMO (Sequential Minimal Optimization), pero en lugar de optimizar en cada paso un par de variables duales, como se hace en las implementaciones de SMO para SVM binarios, se optimiza el conjunto de variables duales de un mismo ejemplo, respetando la primera restricción que hace que su suma deba ser cero.

5. Resultados experimentales

En los experimentos realizados comparamos el rendimiento, sobre el conjunto de datos descrito en la Sección 2, de los cuatro algoritmos analizados en la sección anterior: OVO (8), C&S (9), cs-OVO (10) y cs-C&S (11). Todos los resultados fueron estimados por medio de validaciones cruzadas estratificadas de 5 particiones y 2 repeticiones. Para cada algoritmo se probó tanto un kernel lineal como uno gaussiano. Con el objetivo de seleccionar los valores más adecuados para la constante C y el parámetro g, en el caso del kernel gaussiano, se aplicó una búsqueda en dos fases. La primera, con valores de C desde 10^{-6} hasta 10^{2} y valores de g desde 10^{-3} hasta 10, y en la segunda, se realizó una búsqueda más fina utilizando para ello diez valores distribuidos uniformemente entre el anterior y el siguiente al mejor valor obtenido en la primera fase. En esta selección de parámetros, los algoritmos multi-clase (OVO y C&S) trataron de optimizar el error 0/1 ($\Delta_{0/1}$), mientras los algoritmos sensibles al coste (cs-OVO y cs-C&S) usaron la función de pérdida Δ_{GB}. Todas las estimaciones para este ajuste de parámetros se realizaron mediante un validación cruzada estratificada de 2 particiones y 3 repeticiones.

[2]El código fuente puede descargase desde http://www.aic.uniovi.es/~juanjo/cs.zip

Kernel	Algoritmo	$\Delta_{0/1}$	Δ_{GB}
Lineal	OVO	0.10923	0.09255
	cs-OVO	0.18115	0.08642
	C&S	0.11419	0.11757
	cs-C&S	0.18309	0.10170
Gauss.	OVO	0.06395	0.09461
	cs-OVO	0.10972	0.07995
	C&S	0.06531	0.06508
	cs-C&S	0.07221	0.06144

Tabla 1: Resultados en error $\Delta_{0/1}$ y Δ_{GB} con kernel lineal y gaussiano

En la Tabla 1 se muestran los resultados obtenidos utilizando tanto el kernel lineal com el gaussiano. El mejor algoritmo en error $\Delta_{0/1}$ fue OVO, mientras que cs-OVO obtuvo mejores resultados cuando aplicamos la función de pérdida Δ_{GB}. En este caso particular, parece que los algoritmos basados en descomposición se comportan mejor. Obsérvese además, cómo los algoritmos sensibles al coste cometen un error $\Delta_{0/1}$ mucho más alto que sus versiones no sensibles al coste correspondientes, pero sin embargo tienen menos error Δ_{GB}.

Los resultados anteriores pueden mejorarse utilizando un kernel gaussiano. El mejor algoritmo optimizando el error $\Delta_{0/1}$ sigue siendo OVO con una tasa de acierto superior al 93 %. Por su parte, los mejores resultados en términos de error Δ_{GB} corresponden al algoritmo cs-C&S, que obtiene un acierto en la predicción de la biomasa cercano al 94 %. De nuevo, en ambos casos los algoritmos sensibles al coste superan a sus versiones correspondientes cuan-

Clase	Otros	Ciliados	Crustáceos	Flagelados	Diatomeas	Precisión (%)
Otros	13900676	153003	177742	107379	60525	96.54 %
Ciliados	195126	1171286	0	51015	17051	81.65 %
Crustáceos	531984	36881	14880994	37046	2998	96.07 %
Flagelados	96876	70100	0	2652396	34322	92.95 %
Diatomeas	461022	62354	55050	201292	3421264	81.44 %
Precisión (%)	91.54 %	78.42 %	98.46 %	86.99 %	96.75 %	

Tabla 2: Matriz de confusión para la biomasa

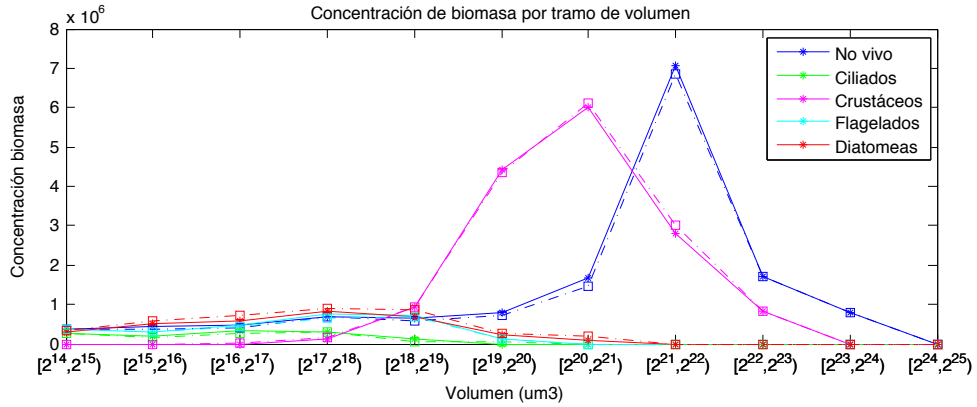

Figura 4: Concentración de biomasa por tramo de volumen

do se analiza el error Δ_{GB}, en mayor medida en el caso de cs-OVO. Uno de los motivos, es que con el algoritmo C&S obtenemos un error bastante bajo, por lo que el margen de mejora no es muy grande.

La Tabla 2 muestra la matriz de confusión correspondiente al algoritmo con mejores resultados cuando consideramos la función de pérdida Δ_{GB}: cs-C&S. Es interesante estudiar con atención dicha matriz para analizar el rendimiento del algoritmo respecto a cada clase. A la hora de interpretarla debemos tener en cuenta que en las columnas se muestra la clase real, mientras que en las filas tenemos la clase que ha predicho el clasificador; en la diagonal están las clasificaciones correctas. En la última columna y en la última fila se muestran, respectivamente, el porcentaje de biomasa predicho por el clasificador que realmente es de esa clase, y el porcentaje de la biomasa real de esa clase que ha predicho el clasificador. Así por ejemplo, para la clase Crustáceos, el 98,46 % de la biomasa total que corresponde a esa especie ha sido etiquetada como de esa clase. Por el contrario, el 96,07 % de la biomasa que el clasificador asigna a los Crustáceos es realmente de esa clase. Como se puede apreciar, las mayores dificultades las presenta la clase Ciliados, en la que en ambos casos el acierto ronda solamente el 80 %.

Para finalizar, la Figura 4 muestra la predicción de biomasa por tamaño del ejemplo. La biomasa real de la clase viene dada por las lineas de puntos, mientras que la biomasa predicha por el clasificador aparece representada por las líneas continuas. Hay que destacar que el área bajo ambas líneas es muy similar en todo el espectro, por lo que podemos concluir que se obtiene una buena predicción de la biomasa para todos los intervalos de volumen.

6. Conclusiones y trabajo futuro

A lo largo de este estudio se ha presentado una aplicación interesante que permite automatizar el muestreo y la estimación de la biomasa para 5 especies de microplancton. Con el fin de optimizar la predicción de la biomasa de cada una de las especies consideradas, se han desarrollado dos algoritmos sensibles al coste. Ambos algoritmos mejoran dicha predicción con respecto a los algoritmos tradicionales que optimizan el error 0/1. Las técnicas aplicadas en esta investigación no son solo válidas para muestras de microplancton. Nuestro objetivo es extender su aplicación a otros tipos de plancton, como por ejemplo el mesoplancton, o a diferentes subgrupos taxonómicos que puedan ser relevantes desde un punto de vista científico.

Referencias

[1] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2:265–292, 2001.

[2] Cabell Davis, Scott Gallager, and Andrew Solow. Microaggregations of oceanic plankton observed by towed video microscopy. *Science*, 257(5067):230–232, 1992.

[3] Charles Elkan. The foundations of cost-sensitive learning. In *IJCAI*, pages 973–978. Morgan Kaufmann, 2001.

[4] Hongjian Fan and Kotagiri Ramamohanarao. A weighting scheme based on emerging patterns for weighted svm. In *IEEE International Conference on Granular Computing*, pages 435– 440, 2005.

[5] Robert Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610–621, 1973.

[6] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[7] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, 8(2):179–187, 1962.

[8] U Kreßel. Pairwise classification and support vector machines. In *Advances in Kernel Methods – Support Vector Learning*, pages 255–268. MIT Press, 1999.

[9] F. Kuhl and C. Giardina. Elliptic fourier features of a closed contour. *Computer Graphics and Image Processing*, 18:236–258, 1982.

[10] G. Matheron. *Random sets and integral geometry*. Wiley New York,, 1974.

[11] Susanne Menden-Deuer and Evelyn J. Lessard. Carbon to volume relationships for dinoflagellates, diatoms and other protist plankton. *Limnology and Oceanography*, 45(3):569–579, 2000.

[12] John C. Platt, Nello Cristianini, and John Shawe-taylor. Large margin dags for multiclass classification. In *NIPS*, pages 547–553. MIT Press, 2000.

[13] C. K. Sieracki, M. E. Sieracki, and C. S. Yentsch. An imaging-in-flow system for automated analysis of marine microplankton. *Marine Ecology Progress Series*, 168:285–296, 1998.

[14] Xiaoou Tang, W. Kenneth Stewart, He Huang, Scott M. Gallager, Cabell S. Davis, Luc Vincent, and Marty Marra. Automatic plankton image recognition. *Artificial Intelligence Review*, 12(1-3):177–199, 1998.

[15] M. R. Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America (1917-1983)*, 70:920–930, August 1980.

[16] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, NY, 1998.

[17] J. Weston and C. Watkins. Multi-class support vector machines. In *ESANN*, 1999.