Universidad de Oviedo

Departamento de Informática

Diseño de métodos de cuantificación aplicados a la estimación de la distribución de grupos taxonómicos presentes en muestras de plancton

Tesis presentada por Pablo González González
Doctorado en informática

Directores de la Tesis
Dr. Juan José del Coz Velasco
Dr. Jorge Díez Peláez

Oviedo, 2019

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

RESUMEN DEL CONTENIDO DE TESIS DOCTORAL

1.- Título de la Tesis	
Español/Otro Idioma: Español/Otro Idioma: Diseño de métodos de cuantificación aplicados a la estimación de la distribución de los grupos taxonómicos presentes en muestras de plancton.	Inglés: Design of quantification methods applied to the distribution estimation of taxonomic groups present in plankton samples.

2.- Autor	
Nombre: Pablo González González	
Programa de Doctorado: Doctorado en Informática	
Órgano responsable: Universidad de Oviedo (Departamento de Informática)	

RESUMEN (en español)

En las últimas décadas se han desarrollado nuevos dispositivos automáticos para la captura de muestras de plancton como por ejemplo la Imaging FlowCytobot o la FlowCam. Estos dispositivos son capaces de capturas miles de imágenes de plancton de manera automática y desatendida. El procesamiento manual de estas muestras se convierte en una tarea imposible y por tanto se necesitan desarrollar nuevos métodos automáticos y eficientes para el procesamiento de dichas muestras.

En la actualidad existen multitud de estudios que han abordado este problema desde el punto de vista de la clasificación automática y la visión artificial. El enfoque usado generalmente es analizar las imágenes con técnicas de visión artificial para luego aprender un modelo utilizando un conjunto de entrenamiento etiquetado manualmente. Este modelo será utilizado para clasificar los ejemplos de nuevas muestras capturadas por los dispositivos citados anteriormente. En este tipo de enfoques, los algoritmos utilizados para construir el modelo son algoritmos de clasificación cuyo objetivo principal es clasificar entre diferentes clases cada una de las imágenes de plancton presentes en la muestra. Utilizando este enfoque surgen tres cuestiones principales:

1. Los biólogos marinos muchas veces están interesados en la distribución de las clases de los organismos presentes en una muestra de plancton, no siendo interesante la clasificación individual de los organismos, sino el resultado agregado.
2. La distribución del plancton cambia enormemente de manera temporal y espacial. Los algoritmos de clasificación no están pensados para manejar esta variabilidad y es necesario el uso de algoritmos pensados específicamente para este tipo de problemas.
3. Debido al punto anterior, suele ocurrir que los métodos de validación tradicionales como la validación cruzada sobrestiman la eficacia de los modelos obtenidos, y cuando estos sistemas se aplican en producción, se obtengan unos resultados mucho peores que los estimados a priori.

Los puntos anteriores nos llevan a explicar las principales contribuciones de esta tesis:

1. Revisión de todos los métodos de cuantificación existentes y sus principales usos y aplicaciones.
2. Establecimiento de métodos de validación adecuados para evaluar correctamente los métodos de cuantificación y obtener medidas reales y que además nos permitan evaluar de manera precisa el sistema.
3. Aplicación de métodos de cuantificación y las técnicas de validación desarrolladas al problema real del plancton, usando para ello las últimas técnicas disponibles en visión artificial (redes neuronales convolucionales) y con un conjunto de datos real perteneciente al Woods Hole Oceanographic Institution de Massachusets.

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

RESUMEN (en Inglés)

In the last decades, new devices for the automatic capture of plankton samples have been developed. Some examples of these are the Imaging FlowCytobot or the FlowCam; they are able to capture thousands of plankton images automatically and unattended. Due to the elevated number of examples to process, the manual processing of these samples is then an unachievable task; thus, it is critical to deploy new automatic and efficient methods to process these samples.

Researchers have been trying to tackle this problem from the automatic classification and artificial vision perspectives. The approach commonly used is to analyze each plankton image with artificial vision techniques to then learn a model using a manually labeled training dataset. This model is then put to use classifying the examples of an unknown sample previously captured. In this kind of approach, the algorithms used for building the model are classification algorithms, being the main objective to place each plankton image in different classes. Using this method, three main discussion points arise:

1. More often than not, marine biologists are interested in the distribution of the examples of each class in a sample. This means that the individual classification of each example is not a priority but the aggregate result is.
2. The plankton distribution varies a lot temporally and spatially. Classification algorithms are not designed to handle this variability and it is necessary to use algorithms specifically designed for these problems.
3. Consecuentely, validation methods as cross validation usually overestimate the effectiveness of the obtained models and when those are taken into production environments, results are much worse than those estimated in the building phase.

The previous discussion points lead us to explain the main contributions of this thesis:

1. Revision of all the existing quantification methods, their main uses and applications.
2. Definition of validation methods suitable for properly evaluating quantification methods and obtaining real performance data, which will allow us to evaluate with more precision the system built.
3. Application of quantification methods and the developed validation methods to a real life problem (marine plankton) using the latest techniques in the field of artificial vision (convolutional neural networks) and with a real dataset belonging to the Woods Hole Oceanographic Institution from Massachusets.

SR. PRESIDENTE DE LA COMISIÓN ACADÉMICA DEL PROGRAMA DE DOCTORADO EN INFORMÁTICA

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

FORMULARIO RESUMEN DE TESIS POR COMPENDIO

1.- Datos personales solicitante	
Apellidos: González González	Nombre: Pablo

Curso de inicio de los estudios de doctorado	2014

	SI	NO
Acompaña acreditación por el Director de la Tesis de la aportación significativa del doctorando	X	
Acompaña memoria que incluye		
Introducción justificativa de la unidad temática y objetivos	X	
Copia completa de los trabajos *	X	
Resultados/discusión y conclusiones	X	
Informe con el factor de impacto de las publicaciones	X	
Se acompaña aceptación de todos y cada uno de los coautores a presentar el trabajo como tesis por compendio	X	
Se acompaña renuncia de todos y cada uno de los coautores a presentar el trabajo como parte de otra tesis de compendio	X	

* Ha de constar el nombre y adscripción del autor y de todos los coautores asi como la referencia completa de la revista o editorial en la que los trabajos hayan sido publicados o aceptados en cuyo caso se aportará justificante de la aceptación por parte de la revista o editorial

FOR-MAT-VOA-033

Artículos, Capítulos, Trabajos

Trabajo, Artículo 1

Titulo (o título abreviado)	Validation methods for plankton image classification systems
Fecha de publicación	27 de noviembre de 2016
Fecha de aceptación	7 de noviembre de 2016
Inclusión en Science Citation Index o bases relacionadas por la CNEAI (indíquese)	JCR
Factor de impacto	2.015

Coautor2 x Doctor □ No doctor . Indique nombre y apellidos	Eva Álvarez
Coautor3 x Doctor □ No doctor . Indique nombre y apellidos	Jorge Díez
Coautor4 x Doctor □ No doctor . Indique nombre y apellidos	Ángel López-Urrutia
Coautor5 x Doctor □ No doctor . Indique nombre y apellidos	Juan José del Coz
Coautor6 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor7 □ Doctor □ No doctor . Indique nombre y apellidos	

Trabajo, Artículo 2

Titulo (o título abreviado)	A Review on Quantification Learning
Fecha de publicación	5 de noviembre 2017
Fecha de aceptación	4 de mayo de 2017
Inclusión en Science Citation Index o bases relacionadas por la CNEAI (indíquese)	JCR
Factor de impacto	5.55

Coautor2 □ Doctor x No doctor . Indique nombre y apellidos	Alberto Castaño
Coautor3 x Doctor □ No doctor . Indique nombre y apellidos	Nitesh V. Chawla
Coautor4 x Doctor □ No doctor . Indique nombre y apellidos	Juan José del Coz
Coautor5 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor6 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor7 □ Doctor □ No doctor . Indique nombre y apellidos	

Trabajo, Artículo 3

Titulo (o título abreviado)	Automatic plankton quantification using deep features
Fecha de publicación	Pendiente de publicación (se adjunta doi y carta de aceptación)
Fecha de aceptación	10 de mayo de 2019
Inclusión en Science Citation Index o bases relacionadas por la CNEAI (indíquese)	JCR
Factor de impacto	1.897

Coautor2 □ Doctor x No doctor . Indique nombre y apellidos	Alberto Castaño
Coautor3 □ Doctor x No doctor . Indique nombre y apellidos	Emily E. Peacock
Coautor4 x Doctor □ No doctor . Indique nombre y apellidos	Jorge Díez
Coautor5 x Doctor □ No doctor . Indique nombre y apellidos	Juan José del Coz
Coautor6 x Doctor □ No doctor . Indique nombre y apellidos	Heidi M. Sosik
Coautor7 □ Doctor □ No doctor . Indique nombre y apellidos	

En caso de compendio de un número de artículos superior a seis, se incorporarán hojas suplementarias conforme a este modelo

Agradecimientos

Escribir una tesis doctoral es un trabajo que requiere una tener una serie de cualidades para las que uno normalmente no está preparado antes de empezar. Requiere esfuerzo y constancia, además saber afrontar periodos de tiempo en el que el avance puede ser mínimo o casi inexistente. En mi caso, este trabajo no habría sido posible sin la inestimable ayuda de mis directores de tesis, Juanjo y Jorge, que han sabido darme un empujón en las horas bajas y me han ayudado de esta manera a superar los momentos más difíciles de todo el proceso.

Me gustaría también agradecer el apoyo incondicional de mi compañera Laura, que ha sabido apoyarme en los momentos en los que las cosas no salían como estaba previsto.

Mi especial agradecimiento a cada uno de los autores que aparecen en cada uno de los tres artículos que forman esta tesis, tanto a los del Centro de Inteligencia Artificial de Gijón como al resto. En especial me gustaría agredecer a la Dra. Heidi M. Sosik y Emily Peacock del Woods Hole Oceanographic Institution por haberse involucrado en este trabajo y haber estado siempre dispuestas a colaborar y ayudar.

Por último, agradecer a mi familia por el apoyo durante todo el proceso y por el ánimo para seguir estudiando y trabajando en lo que te gusta aunque sea en tu tiempo libre.

Índice general

Capítulo 1

Introducción

En las últimas décadas han aparecido numerosos dispositivos de captura automática de muestras de plancton, tales como la FlowCam (Sieracki et al., 1998) o más recientemente la FlowCytobot (Olson y Sosik, 2007). Este tipo de máquinas son capaces de recoger una muestra de agua y, a partir de la misma, capturar imágenes de todos los microorganismos planctónicos presentes en la muestra mediante un proceso automático y transparente para el usuario. En consecuencia, cuando estos dispositivos se ponen a funcionar en condiciones reales, obtienen una cantidad de datos imposible de manejar de manera manual. Por poner un ejemplo concreto, la FlowCytobot situada en el Martha's Vineyard Coastal Observatory (MVCO) ha obtenido desde el año 2006 cerca de un billón de imágenes de organismos de plancton.

Es evidente que una cantidad de datos como ésta no se puede procesar manualmente y hace que sea imprescindible el desarrollo de métodos automáticos para su análisis. Durante los últimos años, han existido numerosos estudios que han trabajado en resolver el problema de procesar imágenes de plancton de manera automática (Benfield et al., 2007). Estos trabajos caen dentro del ámbito de la visión artificial y la inteligencia artificial y, a día de hoy, son capaces de obtener resultados asombrosamente buenos dentro de la complejidad que entraña el problema.

La dificultad intrínseca del problema viene dada por varios factores. En primer lugar, el tipo de imágenes con las que se trabaja suelen tener una calidad y resolución limitadas. En segundo lugar, los organismos a estudiar pueden aparecer en cualquier ángulo y con condiciones de iluminación diferentes. En tercer lugar, los organismos, aún perteneciendo a la misma clase, presentan innumerables variaciones que hace que incluso para un experto humano (taxónomo) sea un problema realmente complejo. Todos estos factores sumados a la enorme cantidad de especies y taxonomías existentes implica que no se puedan descartar errores humanos en los conjuntos de entrenamiento etiquetados manualmente (Culverhouse et al., 2003).

Tradicionalmente, la manera de describir las imágenes de plancton obtenidas por un sistema de captura automático son técnicas de visión artificial en los que a partir de una imagen en mapa de bits, se calculan una serie de valores como los descriptores de Fourier (Kuhl y Giardina, 1982), atributos de textura de Haralick (Haralick et al., 1973) o momentos de Hu (Hu, 1962). Actualmente estas técnicas parecen dar paso a las redes neuronales profundas (Schmidhuber, 2015), que son capaces de aprender automáticamente características de las imágenes con un gran poder discriminatorio. Este tipo de redes neuronales han supuesto un revulsivo en la clasificación de imágenes (He et al., 2016; Krizhevsky et al., 2012; Simonyan y Zisserman, 2014; Szegedy et al., 2015) y están empezando a aplicarse a la clasificación de plancton (Dai et al., 2016; Lee et al., 2016; Orenstein y Beijbom, 2017) con resultados muy buenos y superiores a los métodos tradicionales.

Además de la dificultad de obtener una descripción adecuada de las imágenes de plancton, existe también el problema de que los ecosistemas marinos son altamente cambiantes en su composición de especies. Esto quiere decir que la distribución del plancton varia radicalmente espacial y temporalmente. Es muy común encontrar grandes variaciones según la estación del año, observando que hay especies que prácticamente desparecen y luego se vuelven muy abundantes. Es importante recordar aquí que los algoritmos de clasificación trabajan con la premisa de que la distribución de los datos no varía entre el conjunto de entrenamiento y el conjunto de prueba (Duda et al., 2012). Es evidente que esta premisa no se cumple en el problema de plancton.

Ante unos cambios drásticos en la distribución de los datos, los algoritmos de clasificación tradicionales dejan de ser un método óptimo para la resolución del problema. Además, en muchas ocasiones los biólogos marinos están interesados en predecir la distribución resultante de una muestra de plancton desconocida, no siendo ni relevante ni necesaria la clasificación individual de cada uno de los especímenes. Este tipo de problemas encajan perfectamente en una rama del aprendizaje automático que se está desarrollando recientemente: la cuantificación.

1.1 Aprendiendo a cuantificar

Cuantificar consiste en aplicar métodos cuyo objetivo es predecir la distribución de una muestra sin tener en cuenta las clasificaciones individuales. El primero en definir la cuantificación y abordarla desde un punto de vista formal fue Forman (Forman, 2005) y la definió como "una tarea de aprendizaje automática en la que: dado un conjunto etiquetado de entrenamiento, se aprende un cuantificador que toma un conjunto sin anotar de prueba como entrada y devuelve la mejor estimación de la distribución de las clases".

Una de las maneras más intuitivas de resolver el problema de cuantificar es clasificar los ejemplos individuales y a continuación contar los ejemplos de cada clase. Este método, llamado *Clasificar y Contar* (CC), no es óptimo ya que no tiene en cuenta los cambios en la distribución que existen en el problema en cuestión. Entender los cambios en la distribución es importante de cara a comprender el funcionamiento de los cuantificadores. Dado un ejemplo x y su clase y, un cambio en la distribución entre los conjuntos de entrenamiento (E) y de test (T) ocurren cuando la probabilidad conjunta de ambos entre las descripciones de los ejemplos y su clase varía entre el conjunto de entrenamiento y el de prueba $P_E(x, y) \neq P_T(x, y)$. No todos los problemas son iguales y según las características del mismo podemos esperar que ciertos componentes de $P(x, y)$ se mantengan y otros no. Concretamente, en los problemas de cuantificación al menos $P(y)$ cambia por la propia definición del problema.

Hasta el momento de la elaboración de esta tesis doctoral se habían desarrollado varios métodos para afrontar el problema de la cuantificación. Uno de los principales problemas de esta técnica tan reciente es la gran diversidad en términos de nomenclatura presentes en los diferentes trabajos publicados hasta la fecha. Así pues, el primer artículo que constituye esta tesis (ver Sección 5.1) incluye una recopilación de los principales métodos de cuantificación utilizando una nomenclatura común. Se abordan temas como la cuantificación binaria (el caso más simple), pero también la cuantificación multiclase o la cuantificación basada en costes. Además de analizar los principales métodos, se estudian sus principales aplicaciones, las medidas de error utilizadas en todos estos trabajos, así como una posible taxonomía de los mismos según su forma de funcionamiento.

1.2 Aplicando la cuantificación a un caso real: el plancton marino

Uno de los principales problemas que tiene la cuantificación es la necesidad de conjuntos de datos mayores comparado con un problema de clasificación normal ya que, en el caso de la cuantificación, la unidad mínima del conjunto de datos manejado pasará a ser la muestra y no un ejemplo individual. Recordemos aquí que nuestro objetivo final es obtener la distribución de las clases de una nueva muestra desconocida durante la fase de entrenamiento. Resulta evidente que la creación de este tipo de conjuntos etiquetados es muy costoso ya que cada muestra puede tener miles de individuos que deberemos de etiquetar manualmente antes de construir el sistema.

El primer conjunto de datos que manejamos para esta tesis doctoral es un conjunto de 60 muestras obtenidas en el Mar Cantábrico (Álvarez et al., 2012) (ver Sección 5.2). En este trabajo se utiliza como unidad mínima para todas las pruebas del sistema la muestra y se comparan estos resultados con los obtenidos utilizando el individuo como unidad mínima en las pruebas (como se realiza habitualmente en los trabajos relacionados con la clasificación automática de plancton). En este caso en particular, el número de muestras es insuficiente si queremos aplicar la técnica de dejar un porcentaje de las muestras separadas para validación del sistema, utilizando el resto de muestras como conjunto de entrenamiento para crear el modelo. Para evitar este inconveniente, desarrollamos un método que llamamos *validación cruzada por muestra* , en el que aplicamos el concepto de la validación cruzada y la aplicamos con muestras completas en lugar de con ejemplos individuales. Las muestras tienen que cumplir además, que sean completas (no se haya eliminado ningún ejemplo manualmente) y que no tengan ningún proceso posterior al de su obtención que no sea automático. De esta forma se garantiza que la validación realizada se ajuste lo más posible a los datos reales de eficacia del sistema cuando éste sea puesto en producción. Gracias a la utilización de muestras completas para realizar las pruebas se consigue obtener una estimación precisa del error futuro del sistema. La *validación cruzada por muestra* permite seguir esta misma filosofía en las situaciones en las que el tamaño del conjunto es reducido. Además hemos demostrado que la validación cruzada por individuo no es un método adecuado para evaluar los modelos creados en este tipo de problemas ya que normalmente sobreestima los resultados obtenidos.

En el tercer y último artículo que compone esta tesis (ver Sección 5.3) se utiliza un conjunto de datos mucho más grande, con un mayor número de muestras y de ejemplos (3.4 millones de ejemplos organizados en 964 muestras) (Sosik et al., 2015). Este conjunto de datos pertenece al Woods Ocean Institute de Masachussetts y fue recogido en el Observatorio costero de Martha's Vineyard. Con un conjunto de datos de este tipo, se deben seguir las recomendaciones propuestas en el segundo artículo de la tesis (ver Sección 5.2) a la hora de diseñar la validación del sistema, pero no es necesario (y sería muy costoso computacionalmente) utilizar una *validación cruzada por muestra*, ya que el número de muestras es suficientemente grande. Como veremos posteriormente, con este conjunto de datos se utilizaron técnicas de extracción de atributos basadas en redes neuronales convolucionales (CNN), con el objetivo de compararlas con las características que son usadas normalmente en este tipo de problemas (atributos de forma y textura). Se realizó además un número importante de experimentos utilizando varios tipos de atributos extraídos de las imágenes usando diferentes redes neuronales profundas, combinados con diferentes algoritmos de cuantificación con el objetivo de extraer conclusiones interesantes sobre el funcionamiento tanto de los métodos de cuantificación, como de los atributos calculados con las CNNs.

Este documento se desarrolla como sigue: en el Capítulo 2 se desarrollan los objetivos planteados para esta tesis doctoral; en el Capítulo 3 se analizan los resultados obtenidos; a continuación en el Capítulo 4 se extraen unas conclusiones generales sobre el trabajo; en el Capítulo 5 se incluyen los tres artículos publicados y que constituyen esta tesis doctoral; por último, en el Capítulo 6 se analiza el índice de impacto de las revistas en la que se han publicado los artículos.

Capítulo 2

Objetivos

En esta tesis doctoral los objetivos han sido desarrollados en cada una de las publicaciones que la componen. De esta manera, cada uno de los trabajos incide claramente sobre unos subobjetivos diferentes que convergen en la consecución de un objetivo más general, que da sentido a esta tesis.

El objetivo general lo podemos describir como el diseño y uso de métodos de cuantificación aplicados al problema de plancton. Este objetivo trata de afrontar este complicado problema desde una perspectiva diferente a la tradicional, utilizando para ello técnicas de cuantificación. Se desarrollan además métodos de validación adecuados para el problema en cuestión y se realizan experimentos con dos conjuntos de datos reales (Álvarez et al., 2012; Sosik et al., 2015), de cara a verificar y probar los aspectos claves de esta tesis.

Debido a que esta tesis se desarrolla a través de tres publicaciones diferentes, se describen a continuación los objetivos desarrollados individualmente en cada una de éstas.

2.1 Objetivos en *A Review on Quantification Learning*

En el primer trabajo publicado para esta tesis doctoral (ver sección 5.1) el objetivo principal ha sido realizar una recopilación de los principales trabajos y la teoría desarrollada en el campo de la cuantificación. Uno de los principales problemas encontrados en esta fase ha sido la diferencia de nomenclaturas utilizada en los diferentes trabajos. Esta diferencia es debida a lo reciente que es la cuantificación en la literatura, confundiéndose muchas veces con otros problemas del campo del aprendizaje automático. La primera definición teórica viene dada por Forman (Forman, 2005) hace poco más de una década y desde entonces no han sido pocos los trabajos que han hecho mella en este campo (Beijbom et al., 2015; Forman, 2006; Milli et al., 2015; Tasche, 2014),etc.

En este primer artículo se incide principalmente en este punto, realizando una recopilación y unificación de lo publicado hasta el momento. Aquí se da una definición clara de cuantificación y se señalan las características más importantes de esta tarea de aprendizaje. Además, se hace énfasis en la diferencia entre cuantificación y clasificación y se justifica porque la cuantificación es un problema en sí mismo, diferente de la clasificación.

Otro de los objetivos desarrollados en este trabajo es ofrecer una visión global de la cuantificación desde un punto de vista teórico y se propone una unificación de la notación matemática usada para definir cada uno de los métodos de cuantificación y funciones de pérdida utilizadas. Por último, otro objetivo complementario importante es que este trabajo sirva como guía de referencia para todos los investigadores que quieran introducirse en el campo de la cuantificación.

2.2 Objetivos en *Validation methods for plankton image classification systems*

El segundo trabajo del que se compone esta tesis doctoral (ver sección 5.2) tiene el objetivo principal de establecer una metodología de validación adecuada para problemas como el del plancton, en los que la distribución de los organismos varía tan radicalmente entre las diferentes muestras obtenidas. La mayor parte de los trabajos realizados hasta el momento en este campo suelen utilizar conjuntos de datos creados artificialmente en los que las imágenes son seleccionadas manualmente, cogiendo generalmente los especímenes más interesantes. Para realizar la validación de los métodos de clasificación desarrollados se usan normalmente validaciones cruzadas tradicionales sobre dichos conjuntos (González et al., 2013; Gorsky et al., 2010; Lisin et al., 2005; Luo et al., 2004; Zhao et al., 2010), o también el método de retención (hold out), en el que se utiliza para entrenar un porcentaje de los ejemplos y para hacer las pruebas el resto (Dai et al., 2016; Gorsky et al., 1989; Hu y Davis, 2005; Jeffries et al., 1984; Simpson et al., 1991). En este segundo artículo se defiende que la unidad mínima para realizar las pruebas no puede ser el individuo sino la muestra. De esta manera y debido al bajo número de muestras presentes en el conjunto de datos utilizado, se desarrolla un método denominado *validación cruzada dejando uno fuera por muestra* y se definen las características que deben cumplir dichas muestras. De esta manera se garantiza que los resultados obtenidos en la fase de validación serán lo más reales posibles dado que se replican de manera muy fiel las condiciones que se van a encontrar con el sistema en producción. También se incluye una comparación sobre los resultados utilizando la validación cruzada tradicional y la *validación cruzada por muestra* y se demuestra cómo la poca variabilidad encontrada en las hojas de la validación cruzada tradicional resulta en una subestimación de las medidas de error obtenidas.

El conjunto de datos utilizado en este trabajo fue recogido utilizando una FlowCam (Sieracki et al., 1998) en diferentes puntos del Mar Cantábrico (Álvarez et al., 2012). El conjunto está formado por 60 muestras totalmente etiquetadas donde cada ejemplo individual puede pertenecer a ocho clases diferentes. Aunque este conjunto de datos es un buen conjunto, teníamos la impresión de que era necesario un conjunto mayor y con más muestras para poder realizar unos experimentos más concluyentes. De esta manera, llegamos al conjunto de datos recopilado por el *Woods Hole Oceanographic Institution* de Massachusetts y que daría forma al tercer y último artículo publicado de esta tesis.

2.3 Objetivos en *Automatic plankton quantification using deep features*

Para este trabajo (ver Sección 5.3) contamos con un conjunto de datos mucho mayor que para el artículo anterior, denominado WHOI-Plankton (Sosik et al., 2015). Este conjunto de datos está disponible públicamente en internet y está formado por 964 muestras totalmente etiquetadas además de tener disponible más de 1 billón de ejemplos individuales (sin etiquetar) capturados durante los últimos años. El objetivo en este caso fue aplicar los principales algoritmos de cuantificación existentes y compararlos con el método más básico utilizado cuando se desea conocer la composición de una muestra, el método *Clasificar y Contar o Classify & Count (CC)*. Los métodos de cuantificación comparados con CC fueron *Adjusted Count (AC)*, (Forman, 2008), *Probabilistic Classify Count (PCC), Probabilistic Adjusted Count (PAC)* (Bella et al., 2010) y HDy (González-Castro et al., 2013), que es un método de cuantificación basado en la distancia de Hellinger.

El objetivo era demostrar cómo estos métodos podrían suponer una mejora sobre el método CC. Como objetivo añadido en este artículo (y que no estaba previsto a la hora de empezar esta tesis) se utilizaron como datos de entrada para los algoritmos de cuantificación características computadas por redes neuronales convolucionales (CNN), en este caso *Resnets* (He et al., 2016). El principal objetivo en este caso fue calcular cuál es la mejora en eficacia de este tipo de características con respecto a las características obtenidas por técnicas de visión artificial tradicionales (como los atributos de forma o textura). También se exploró la transferencia de aprendizaje (*transfer learning*) (Pan y Yang, 2010) con el objetivo de utilizar redes neuronales preentrenadas en conjuntos de otro dominio (por ejemplo ImageNet (Deng et al., 2009)) y comprobar el grado de utilidad de este preentrenamiento a la hora de utilizar estas mismas redes para extraer características utilizables para cuantificar plancton.

Capítulo 3

Discusión de resultados

La parte principal de la que se extraen los resultados de esta tesis doctoral son el segundo y tercer artículos publicados, ya que el primero es de un contenido más teórico y en el que la experimentación no es la parte relevante.

La premisa básica de esta tesis ha sido garantizar la reproducibilidad de los experimentos de cara a que cualquiera pueda volver a ejecutarlos, ya sea con los mismos datos o con unos diferentes. De esta forma, todo el código fuente desarrollado se ha mantenido en una cuenta pública de GitHub[1] bajo una licencia GPLv3. Además, todos los experimentos realizados han sido utilizando lenguajes de programación y librerías libres.

3.1 Resultados: *Validation methods for plankton image classification systems*

En este artículo (ver sección 5.2) se presenta una comparación entre los métodos de validación tradicionales, como la validación cruzada, y el método desarrollado en este trabajo: la *validación cruzada por muestra*. Para realizar la comparación se utilizan dos algoritmos de clasificación muy conocidos como son Support Vector Machines (SVM) (Vapnik y Vapnik, 1998) y Random Forest (RF) (Breiman, 2001). En los resultados presentados en la Tabla 2 del artículo se puede observar cómo el acierto tanto para SVM como para RF es superior al 80 % utilizando

[1] `https://github.com/pglez82/IFCB_quantification`

una validación cruzada, pero no llegan al 72 % o al 78 % al utilizar la *validación cruzada por muestra* dependiendo de la forma de promediar los errores. Al analizar cada partición en la validación cruzada estándar, vemos cómo la variación en el acierto es mínima (ver Figura 4). Por el contrario, al utilizar la validación cruzada por muestra, se observa cómo el clasificador tiene problemas en ciertas muestras y existe una variación mucho mayor en el acierto cuando se analizan muestra a muestra. En la Figura 5 del artículo se puede observar en un gráfico de cajas una medida de esta variabilidad del acierto por muestra en ambos métodos clase por clase.

Un segundo análisis realizado en este artículo se basa en medidas agregadas en lugar del acierto utilizado en clasificación. Las medidas utilizas en este caso son SMAPE (Symmetric Mean Relative Error) (Armstrong, 1978) y la disimilitud de Bray-Curtis (Bray y Curtis, 1957), que es una medida muy utilizada en el ámbito de la ecología para analizar los datos de abundancia en diferentes localizaciones de muestreo. Los resultados usando ambas medidas fueron bastante pobres para ambos clasificadores (ver Tabla 3), ya que hay que tener en cuenta que el método de cuantificación utilizado era el clasificar y contar (CC). De todas maneras, salieron a la luz aspectos interesantes, como por ejemplo que un alto acierto en clasificación no conllevaba directamente un buen resultado cuantificando. La razón principal, y que justifica una parte importante de esta tesis, es que los problemas a resolver son diferentes. Un modelo que es óptimo para clasificar no tiene porque serlo para cuantificar y viceversa.

Por último, se descubrió también otra característica interesante de la validación cruzada por muestra. Este método permite un análisis mucho más pormenorizado de los resultados y puede ayudarnos a detectar problemas en nuestro conjunto de datos. Por ejemplo, podemos llegar a la conclusión de que debemos aumentar el número de muestras para obtener una estimación correcta del error.

3.2 Resultados: *Automatic plankton quantification using deep features*

En este artículo (ver Sección 5.3) se ha realizado una labor de experimentación muy amplia utilizando para ello el mayor y más completo conjunto de plancton disponible hasta el momento, denominado WHOI-Plankton (Sosik et al., 2015).

Los experimentos realizados comparan el uso de características tradicionales (atributos de forma y textura) con características computadas a partir de redes neuronales profundas (CNNs). Las redes neuronales utilizadas son una familia de redes de última generación denominadas Resnets (He et al., 2016). Estas redes han conseguido resultados realmente impresionantes sobre el conjunto de datos ImageNet (Deng et al., 2009), ganando la competición ImageNet ILSVRC de 2015 con una tasa de error del 3.6 % en un trabajo en el que los humanos normalmente tienen un error entre el 5 % y el 10 %.

Las redes Resnet se pueden encontrar con diferentes profundidades (desde 18 a 152 capas), aumentando con el número de capas tanto la complejidad de la red como su tiempo de entrenamiento y memoria utilizada. En este trabajo se han hecho experimentos con redes desde 18 capas hasta 101 capas.

Entrenar este tipo de redes desde cero es muy costoso desde el punto de vista computacional y uno de los enfoques utilizados hoy en día para acelerar este proceso es utilizar redes preentrenadas con datos de un dominio diferente. Esta técnica, denominada *transfer learning* (Pan y Yang, 2010), aprovecha la capacidad adquirida por la red para calcular características sobre imágenes diferentes a las usadas en la fase de entrenamiento. En este caso en particular, las redes estaban preentrenadas para resolver el problema ImageNet y las imágenes que presentamos a la red son imágenes de organismos de plancton.

Además de comparar diferentes conjuntos de características se compararon también diferentes algoritmos de cuantificación, partiendo como base del algoritmo CC (clasificar y contar). Todos los resultados se han plasmado en una página web interactiva en la que se pueden explorar todos los experimentos realizados[1].

La primera conclusión que podemos sacar de este trabajo es que las *resnets*, entrenadas con imágenes de otro dominio, funcionan de una manera muy robusta a la hora de describir imágenes de plancton. En las Tablas 2 y 3 del artículo se puede observar cómo incluso la red más pequeña (18 capas), sin reentrenar, mejora los resultados de las características tradicionales. Según las redes se van haciendo más complejas y aumentamos el número de capas se comprueba cómo los resultados mejoran aunque llegan a un punto donde la mejora es muy pequeña. Así vemos que hay saltos importantes de las 18 a las 34 capas y de las 34 a las 50 capas pero al pasar a 101 capas la mejora ya es muy pequeña.

[1]`http://trasgu.aic.uniovi.es/~pgonzalez/whoiresultsv2/`

Se ha realizado también una comparación entre redes sin reentrenar, es decir, redes que nunca han visto una imagen de plancton, y redes reentrenadas con imágenes de plancton. Los resultados desvelan que por norma general, el reentrenamiento de la red es beneficioso a la hora de mejorar el rendimiento de la red. Así pues, los mejores resultados en error absoluto medio (AE) se dan para la red resnet de 101 capas (0.0035), mientras que utilizando características normales (NF) se obtiene un rendimiento peor (0.0149). Es interesante observar cómo para la red de 50 capas el error absoluto medio es de 0.0036, lo que supone un rendimiento casi exacto a la versión de 101 capas.

Otra conclusión que se puede extraer de los resultados es que los algoritmos de cuantificación mejoran significativamente los resultados cuando el clasificador que funciona por debajo no es muy bueno. Según el clasificador va mejorando, los resultados del CC y el resto de algoritmos pasan a ser muy similares. Así podemos ver cómo utilizando NF+CC se obtiene un AE de 0.0149 y cambiando el algoritmo de cuantificación por AC el resultado mejora hasta un AE de 0.0084. Por otro lado cuando el clasificador está construido con las características extraídas de una red neuronal profunda, por ejemplo de la versión de 101 capas, la diferencia entre CC y AC pasa a ser tan solo de 0.0035 a 0.0031 en error absoluto medio.

Las redes neuronales profundas son costosas computacionalmente hablando. Requieren hardware específico para ser entrenadas y normalmente se utilizan equipos con GPUs dedicadas para esta tarea. A la hora de realizar los experimentos se tomaron datos de los tiempos de entrenamiento de los modelos y de los tiempos de ejecución de todos los algoritmos utilizados, de manera que otros investigadores puedan ver claramente cuáles son los requisitos de hardware para un sistema de este tipo y cuáles son los tiempos esperados para construir los modelos (ver Tabla 4).

Capítulo 4

Conclusiones

El término plancton viene del griego y significa *lo que va errante*, un nombre apropiado para un conjunto de organismos, en su mayoría microscópicos que flotan a la deriva por todos los océanos del mundo. Su distribución varía según las condiciones ambientales, la concentración de nutrientes, las horas de sol, la profundidad, entre otros factores. Son sistemas cruciales para la ecología del planeta y es un campo de estudio muy importante científicamente hablando. De estos organismos depende toda la cadena trófica de los océanos.

El análisis automático de plancton es una técnica que aunque lleva ya más de dos décadas estudiándose, sigue sin ponerse en marcha completamente. No existen muchos sitios en el mundo donde haya dispositivos de captura instalados permanentemente recogiendo datos para luego ser analizados de manera continua. Uno de los principales problemas es que los algoritmos que se encargan de procesar el plancton y clasificarlo en las diferentes especies todavía necesitan ser perfeccionados de cara a ser usados con garantías en estudios ecológicos. Evidentemente de nada sirve tener dispositivos de captura automáticos de muestras de plancton si a continuación estas muestras tienen que ser procesadas por taxónomos de forma manual.

Este trabajo ha hecho un esfuerzo en mejorar las técnicas usadas para cuantificar automáticamente muestras de plancton con unos errores similares a los obtenidos por un taxónomo humano y de una manera totalmente autónoma sin necesidad de ninguna intervención humana, avanzando un paso más en la precisión obtenida por los sistemas resultantes. Además, el estudio y desarrollo de técnicas de validación adecuadas para problemas de este tipo ha mejorado la manera en la que se estudian y validan los modelos construidos, de forma que se tenga una idea clara de cuál es el rendimiento que podemos esperar de los mismos cuando estos se desplieguen en condiciones de trabajo reales. Estas técnicas de validación no solo nos dan estimaciones más reales del rendimiento de los algoritmos, sino que nos desvelan detalles importantes sobre la calidad y la idoneidad del tamaño del conjunto de datos utilizado.

Otro aspecto importante de este trabajo ha sido la reivindicación la cuantificación como un problema de aprendizaje automático en sí mismo y diferente de la clasificación. Además de hacer un estudio teórico de los diferentes algoritmos de cuantificación existentes y de aplicarlos experimentalmente al problema del plancton, se ha establecido un marco de trabajo para problemas de cuantificación, incluyendo algoritmos, medidas de error, técnicas de validación y análisis de resultados.

Una tesis doctoral es un trabajo que se realiza durante un periodo de tiempo que comprende varios años. En este caso, desde su comienzo ha habido una verdadera revolución en el campo de la visión artificial con los avances realizados en el campo del aprendizaje profundo (deep learning) y el desarrollo de sus técnicas como son las redes convolucionales (CNNs). Aunque no estaba previsto en un primer momento, no hemos querido perder la oportunidad de utilizar estas nuevas técnicas y aplicarlas al problema de plancton. De esta forma hemos conseguido avanzar un paso más en la precisión de los sistemas obtenidos, demostrando que las características obtenidas con las CNNs funcionan mucho mejor que las características tradicionales. Estos resultados son muy prometedores y dan lugar a sistemas mucho más robustos que son capaces de obtener muy buenos resultados incluso en muestras complicadas en las que tenemos clases muy poco numerosas donde las características tradicionales tenían verdaderos problemas para obtener resultados coherentes. Esta mejora es muy esperanzadora de cara al futuro ya que cada día están saliendo nuevas redes neuronales más potentes. Con la misma metodología que la expuesta en este trabajo, podemos utilizar estas nuevas redes para llevar un paso más allá los sistemas creados. De todas maneras, nunca debemos olvidar que al final, siempre estaremos limitados por la calidad del conjunto de entrenamiento del que dispongamos, tanto por su tamaño como por la ausencia de errores en el etiquetado manual de las muestras. Así pues, el esfuerzo en el desarrollo de nuevas técnicas y algoritmos debe de ir paralelo a la creación de conjuntos de datos de calidad.

Una visión de futuro optimista consistiría en tener una red de estaciones de captura de muestras distribuidas por todos los océanos y mares del mundo (incluso en alta mar) capaces de capturar y analizar in-situ las muestras obtenidas, haciendo que los datos generados estén disponibles de manera instantánea para su consulta por toda la comunidad científica. Un sistema como este permitiría monitorizar en tiempo real la salud de los océanos y realizar estudios ecológicos de una manera impensable hasta este momento.

Para llegar a este ambicioso e ilusionante objetivo todavía es necesario que el hardware mejore ya que las necesidades computacionales de un sistema de este tipo son todavía altas. De todas formas, los avances que se están produciendo con nuevos procesadores de muy bajo consumo con capacidades para la inteligencia artificial hacen que esto pueda ser posible antes de lo esperado. Hay que recordar que lo verdaderamente costoso computacionalmente hablando es la creación inicial del modelo y no el procesado posterior de una muestra concreta. Realmente no sería un problema entrenar un modelo durante semanas (ya que solo hay que hacerlo una vez) pero que luego al desplegarlo sea capaz de procesar una nueva muestra en cuestión de minutos y con un coste energético bajo.

Por otro lado, es importante destacar que sería también necesaria la creación de conjuntos de datos mayores y de mayor calidad. Probablemente este trabajo deba ser llevado a cabo por varios centros de investigación de manera colaborativa, de forma que cada uno de ellos no cree su propio conjunto de datos individual sino que se trabaje conjuntamente para crear un conjunto de datos lo más grande y mejor posible.

Capítulo 5

Copia de las publicaciones

En este capítulo se incluye una copia de las tres publicaciones que componen esta tesis doctoral. El orden de las publicaciones es el relevante para esta tesis doctoral y no el orden de publicación estricto. En el Capítulo 6 se adjunta el índice de impacto de las revistas en las que se han publicado estos trabajos.

5.1 A Review on Quantification Learning

Artículo publicado en la revista **ACM Computing Surveys**, Volumen 50 Edición 5, Artículo Nº74. Noviembre de 2017.

Autores: *Pablo González* [1], *Alberto Castaño* [1], *Nitesh Chawla* [2], *Juan José del Coz* [1].

Doi: `https://doi.org/10.1145/3117807`

Resumen

La tarea de cuantificar consiste en devolver una estimación agregada (por ejemplo, la distribución de clases en un problema de clasificación o un número real en un problema de regresión) para un conjunto de prueba, usando un conjunto de entrenamiento que puede tener una distribución sustancialmente diferente. Varias

[1] Centro de Inteligencia Artificial de Gijón, Universidad de Oviedo
[2]Universidad de Notre Dame

aplicaciones reales requieren este tipo de métodos que no necesitan predicciones para ejemplos individuales y solo se centran en obtener estimaciones precisas a nivel agregado. Durante los últimos años, se han propuesto diferentes métodos de cuantificación enfocados desde diferentes perspectivas y con diferentes objetivos finales. Este artículo presenta una revisión unificada de todos los enfoques principales con el objetivo de servir como un tutorial introductorio para nuevos investigadores en este campo.

A Review on Quantification Learning

PABLO GONZÁLEZ and ALBERTO CASTAÑO, University of Oviedo
NITESH V. CHAWLA, University of Notre Dame
JUAN JOSÉ DEL COZ, University of Oviedo

The task of quantification consists in providing an aggregate estimation (e.g., the class distribution in a classification problem) for unseen test sets, applying a model that is trained using a training set with a different data distribution. Several real-world applications demand this kind of method that does not require predictions for individual examples and just focuses on obtaining accurate estimates at an aggregate level. During the past few years, several quantification methods have been proposed from different perspectives and with different goals. This article presents a unified review of the main approaches with the aim of serving as an introductory tutorial for newcomers in the field.

CCS Concepts: • **Theory of computation** → **Machine learning theory**;

Additional Key Words and Phrases: Class distribution estimation, prevalence estimation, quantification

ACM Reference format:
Pablo González, Alberto Castaño, Nitesh V. Chawla, and Juan José Del Coz. 2017. A Review on Quantification Learning. *ACM Comput. Surv.* 50, 5, Article 74 (September 2017), 40 pages.
https://doi.org/10.1145/3117807

1 INTRODUCTION

Quantification is a relatively new supervised learning task that has emerged in the last decade. Forman (2005) gave the first formal definition for the quantification task for machine learning: "Given a labelled training set, induce a quantifier that takes an unlabelled test set as input and returns its best estimate of the class distribution." While this is a definition of quantification for classification problems, quantification methods have also been developed for other learning tasks, such as regression, ordinal classification, and cost-sensitive problems. The goal is the same in all cases: to obtain an aggregate estimate for a test set without requiring predictions for its individual instances. A prototypical application is to automatically estimate the proportions of positive, neutral, and negative comments about a product or a service during a concrete period of time in a social network like Twitter or Facebook. The idea is that classifying individual comments is unnecessary for some applications that only require an estimation of the percentage of each class—that is, how many of the reviewers are positive or negative or neutral.

The article was written during the stay of Juan Jos del Coz at the University of Notre Dame, supported by the Fulbright Commission and the Salvador de Madariaga Program, Grant PRX15/00607. This research has been also funded by MINECO and FEDER, Grant TIN2015-65069-C2-2-R. N. V. Chawla was supported in part by the NSF Grant IIS-1447795.
Authors' addresses: P. González, A. Castaño, and J. J. del Coz, Artificial Intelligence Center, University of Oviedo at Gijón, Spain; emails: pgonzalez@aic.uniovi.es, {castanoalberto, juanjo}@uniovi.es, N. V. Chawla, Interdisciplinary Center for Network Science and Applications, University of Notre Dame, USA; email: nchawla@nd.edu.

While there are several possible applications of quantification learning, as we discuss below, quantification learning is still relatively unknown even to several machine-learning experts. The main reason is the mistaken belief of it being a trivial task that can be solved using a straightforward approach, the popular Classify & Count method (see Section 6.1) based on off-the-shelf classifiers. However, quantification requires more sophisticated methods if the goal is to obtain optimal models. Quantification may seem to be an easy task, but it is as difficult as other learning problems. In fact, it is a challenging learning problem, because, by its own definition, the data distribution changes between the training phase and the testing phase. This connects quantification with other problems that deal with changes in the distribution, such as concept drift (Gama et al. 2014), domain adaptation (Daume III and Marcu 2006), and transfer learning (Pan and Yang 2010).

Another issue with respect to quantification learning is that the literature on quantification related-methods is somehow disconnected. On the one hand, there are methods designed for quantification, but on the other hand, some of the methods that can be used as quantifiers have been devised for other purposes, mainly to improve classification accuracy when the domain changes. Methods of both groups are usually not compared; in fact, the performance of the latter group has been normally studied just in terms of the improvement on classification tasks but not as quantifiers. Unsurprisingly, given this scenario, algorithms that can be applied for quantification tasks appear on articles that use different keywords and names, such as *prior probability shift* (Storkey 2009), *posterior probability estimation* (Alaiz-Rodríguez et al. 2011), *class prior estimation* (Du Plessis and Sugiyama 2014a), *class-prior change* (Du Plessis and Sugiyama 2014b), *prevalence estimation* (Barranquero et al. 2013), or *class ratio estimation* (Asoh et al. 2012), to cite only some of them.

The aim of this article is not to necessarily enumerate all the quantification algorithms proposed so far but to present a comprehensible discussion of some of the main approaches under a common framework. In this sense, one of the contributions of the article is to introduce a taxonomy of the quantification methods, unifying the algorithms devised for different goals. Then, the most important methods of each group are explained with sufficient detail to serve as an introductory tutorial for new researchers in quantification learning. Besides, the article does not just focus on the most popular problems (binary and multi-class quantification) but also other quantification tasks are reviewed, namely cost quantification, ordinal quantification, quantification for regression, and network quantification.

The structure of the article is as follows. Sections 2, 3, and 4, respectively, provide a formal description of different quantification learning tasks, the strategies to evaluate the performance of quantification methods, and the loss functions used for different quantification problems. Section 5 introduces a possible taxonomy of quantification methods. Then, the remaining sections are devoted to discussing the characteristics of the main approaches proposed in the literature for each quantification task. The article is closed drawing some brief conclusions.

2 QUANTIFICATION LEARNING

2.1 Binary and Multi-Class Quantification

Supervised learning tasks typically require a dataset that can be defined as follows: $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$, in which $\boldsymbol{x}_i$ is the representation of an individual example in the input space $\mathcal{X}$ and y_i is its corresponding class or label, $y_i \in \mathcal{Y} = \mathscr{L} = \{\ell_1, \ldots, \ell_l\}$. The typical goal of supervised classification then is to induce a hypothesis or model, h, from D,

$$h : \mathcal{X} \longrightarrow \{\ell_1, \ldots, \ell_l\}, \tag{1}$$

that correctly predicts the class of unlabelled query instances. When the number of classes is greater than 2, that is, $l > 2$, we have a multi-class classification problem, and it is a binary classification task when the classes are just two, that is, $l = 2$. In the binary classification case, the classes are usually denoted as $\{+1, -1\}$, representing the positive and the negative classes, respectively. Whether a binary- or multi-class classification problem, the learning task at hand is to induce h from corresponding D to predict classes on the unseen data.

The main difference between quantification and classification is that a quantification model, or quantifier, does not make predictions for individual instances but for samples, that is, groups of instances. This also occurs in multi-instance learning (Foulds and Frank 2010) in which the model is designed to assign a class label for a given *bag*[1] of instances, representing usually that at least one of the individual examples of the bag belongs to the assigned class. The goal of quantification learning is to predict an aggregate magnitude for the whole bag or sample. For instance, in *multi-class quantification*, the model predicts the prevalence of each class in the sample. Formally,

$$\bar{h} : \mathbb{N}^{\mathcal{X}} \longrightarrow [0, 1]^l. \tag{2}$$

$\mathbb{N}^{\mathcal{X}}$ denotes a multi-set of examples from $\mathcal{X}$. A multi-set is represented by the number of times that each $\boldsymbol{x} \in \mathcal{X}$ appears in that set or sample. Each element of the predicted vector, $\hat{p}_j$, represents the probability of the class j in the sample. Thus, their sum is 1, $\sum_{j=1}^{l} \hat{p}_j = 1$, and the quantifier returns the probability distribution of the classes. The goal is to predict a class probability distribution $[\hat{p}_1, \ldots, \hat{p}_l]$ that is as close as possible to the true one $[p_1, \ldots, p_l]$. As it occurs in supervised classification, multi-class quantification ($l > 2$) is a more complex learning task than binary quantification ($l = 2$), because it requires a multivariate predictive model. In *binary quantification*, the quantifier produces univariate predictions, typically the estimated prevalence of the positive class, denoted as $\hat{p}$,

$$\bar{h} : \mathbb{N}^{\mathcal{X}} \longrightarrow [0, 1], \tag{3}$$

because the prevalence of the negative class can be trivially obtained using $\hat{p}$: $\hat{n} = 1 - \hat{p}$. It is worth mentioning that some loss functions compare the whole predicted distribution $[\hat{p}, \hat{n}]$ and the true one $[p, n]$, but other loss functions only compare the predicted prevalence for the positive class, $\hat{p}$, with the actual one, p (see Section 4.1).

Most of the quantification methods are designed for binary quantification. Forman (2008) proposes to use the one-versus-all approach to adapt binary quantifiers to multi-class quantification. The implementation of one-versus-all for multi-class quantification is quite straightforward:

(1) Training: Given a multi-class training set with l classes, l binary quantifiers, $\{\bar{h}_1, \ldots, \bar{h}_l\}$, must be learned. In the training set for binary quantifier $\bar{h}_j$, D_j, all the examples that belongs to class j are labelled as positives, with the rest being negatives.
(2) Testing: Given a multi-class test set T, each binary quantifier, $\bar{h}_j$ is applied over T to obtain $\hat{p}_j^0$, an initial estimation of the prevalence for class j.
(3) Final Prediction: The initial prevalences computed, $\hat{p}_j^0$, are normalized so they sum to 1:

$$\hat{p}_j = \frac{\hat{p}_j^0}{\sum_{k=1}^{l} \hat{p}_k^0}. \tag{4}$$

Forman (2008) argues that the normalization process may compensate for imperfect quantification due to class imbalance.

2.2 Connections and Differences Between Classification and Quantification

Although quantification represents a new learning problem, it bears some similarity with the classification task. First, they share the same input—a labelled set of training examples. Second, quantification tasks can generally be solved using classifiers. Notice that the quantifier $\bar{h}$ defined in

[1]The term *bag* is more commonly used in multi-instance learning than in quantification articles.

Equation (2) can be obtained using the classifier h in Equation (1). Given a test sample $T \in \mathbb{N}^{\mathcal{X}}$, its class probability distribution can be estimated using:

$$\bar{h}(T) = \left[\frac{\sum_{\boldsymbol{x} \in T} I(h(\boldsymbol{x}) = \ell_1)}{|T|}, \ldots, \frac{\sum_{\boldsymbol{x} \in T} I(h(\boldsymbol{x}) = \ell_l)}{|T|} \right], \tag{5}$$

with $I(\cdot)$ as the indicator function. This approach is called Classify & Count (see Section 6.1).

On the other hand, quantification also substantially differs from classification. First, the respective learning tasks (2) and (1) are notably different. Second, the empirical measures to evaluate performances differ. Classification tasks target performance measures such as accuracy or the area under the ROC curve (AUC), while in quantification learning the goal is to optimize metrics that compare probability distributions or metrics taken from regression for those quantification algorithms (e.g., binary quantifiers) that predict single (class distribution) scores (see Section 4). Classification methods also generally assume that training data and test data are independent and identically distributed, that is, the well-known i.i.d. assumption. However, quantification learning does not make the i.i.d. assumption—the very task of quantification learning to predict class proportional estimates in the testing data implies that the the distribution of the testing samples is different than the distribution of the training data (Hofer and Krempl 2013; Storkey 2009).

2.3 Other Quantification Problems

Although the study of quantification has been mainly focused on quantification for classification domains, quantification also appears in other kind of learning problems such as regression, ordinal classification, cost-sensitive learning, and network quantification.

Martino et al. (2016a) study the *ordinal quantification* tasks. The form of this learning problem is the same as multi-class quantification, Equation (2), but there is a total order ($\prec$) defined on the set of classes $\{\ell_1, \ldots, \ell_l\}$ in ordinal quantification problems, that is, $\ell_i \prec \ell_j, \forall i, j : i < j$. The goal is to predict the distribution of the ordered classes as accurately as possible. The interest of ordinal quantification is that it has many potential applications. For instance, training a model that learns how to quantify the number of consumers' opinions that fall inside a particular step in a predefined scale, for example, {VeryNegative, Negative, Fair, Positive, VeryPositive}. This kind of problem arises not only in e-commerce applications but also in many other contexts, especially those in which human preferences play a major role. Similarly to ordinal classification, ordinal quantification requires loss functions that take into account the order among the classes (see Section 4.3) and learning algorithms able to optimize such functions. Applying algorithms designed to deal with multi-class quantification tasks will lead to suboptimal models. Section 9 describes the algorithm proposed by Martino et al. (2016a).

Cost quantification (see Section 10) is the adaptation of cost-sensitive learning for quantification. Two different settings are considered for cost-sensitive learning in classification tasks. On the one hand, there are problems with class-dependent costs: $C_{i,j}$ is the cost of predicting the class ℓ_i when the actual class is ℓ_j. In the case of binary classification and assuming that $C_{i,i} = 0$, the problem only requires two costs: $C_{-,+}$ and $C_{+,-}$ representing the cost of misclassifying a positive example and a negative example, respectively. On the other hand, some problems have example-dependent costs: Each example has attached a cost attribute value, thus the training set has the form of $D = \{(\boldsymbol{x}_1, y_1, c_1), \ldots, (\boldsymbol{x}_m, y_m, c_m)\}$, where $c_i \in \mathbb{R}^+$ is the cost of resolving case $\boldsymbol{x}_i$. Only the latter setting has been considered for cost quantification with two different goals to estimate the total or average cost for each class. One example of application of cost quantification is to predict the cost of repairing a particular model of mobile phone over time. The classes in this problem are the breakdown types considered (e.g., cracked screens, battery faults), and we try to estimate the total or average cost of repairing all breakdowns that occurred over a period of time. The reason to use example-dependent costs is because the cost associated to each case can change depending on multiple factors (e.g., suppliers, labour costs). The first articles that deal with cost quantification were by Forman (2006, 2008). The author focuses on determining the total cost of positive examples on *binary cost quantification* tasks. However, most of the proposed methods in these articles estimate the average cost of the positive examples $\widehat{C}^{+}$ first and then multiply it by the prevalence of the positive class estimated using a binary quantifier.

Multi-class cost quantification problems can be formally solved by learning

$$\bar{h} : \mathbb{N}^{\mathcal{X}} \longrightarrow \mathbb{R}^{l}, \tag{6}$$

in which each element of the predicted vector, $\widehat{C}_{\ell_j}$, represents the average cost associated to class j in the test sample. To obtain the total cost is as simple as computing $|T| \sum_{j=1}^{l} \hat{p}_j \widehat{C}_{\ell_j}$. Another possible real application of multi-class cost quantification is in the automatic plankton recognition domain. In this field, researchers are sometimes not interested in the distribution of the plankton species in a sample but in total biomass weight of each of the species. The problem is defined by assigning an individual cost to each training example (its biomass) and the model must be able to predict the total biomass of each species.

Bella et al. (2014) focus on *quantification for regression* problems (see Section 11) and propose several novel techniques based on discretization. In this case, the examples in the training set D have attached a real value, $y_i \in \mathcal{Y} = \mathbb{R}$. The authors distinguish between two types of regression quantification tasks: (i) the estimation of the expected value for the given sample and (ii) the estimation of the whole distribution. One of the running examples of the article deals with a birthing center that collects data about each mother, including her medical history, and y is the baby weight at birth. With this training data, we may train a model to predict the average weight of the births that will take place for the current set of pregnant women that the center is monitoring for future deliveries. This is an example of the first group of tasks that can be formally described as

$$\bar{h} : \mathbb{N}^{\mathcal{X}} \longrightarrow \mathbb{R}. \tag{7}$$

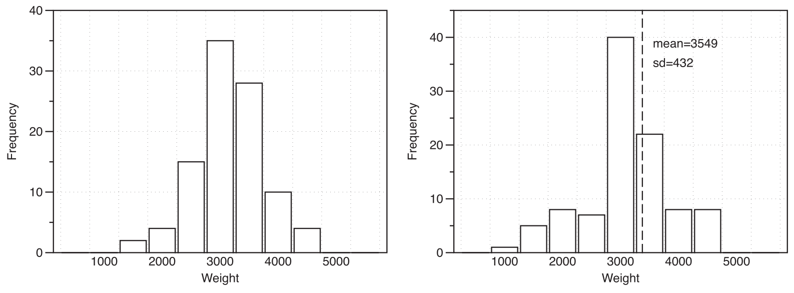
The model predicts a single indicator that represents the expected value of the output (the mean). In some cases, this value can be supplemented with another value that expresses the confidence of the such prediction (e.g., using the standard error).

However, other applications need to estimate the distribution of the output value. For instance, the birthing center may require a regression quantification model able to quantify how many low-weight births (weight lower than 2,500g) will take place. This corresponds to the second group of tasks, because we want to estimate a tail of the distribution. Formally, this problem can be represented in the following way:

$$\bar{h} : \mathbb{N}^{\mathcal{X}} \times \mathbb{R} \longrightarrow [0, 1]. \tag{8}$$

Given a test sample T and a value v, $\bar{h}(T, v) = P(\hat{y}_i < v | \boldsymbol{x}_i \in T)$, in which $\hat{y}_i$ is the predicted value for $\boldsymbol{x}_i$. Figure 1 depicts an example of quantification for regression. The graph on the left represents the actual distribution, while the graph on the right shows an estimation of the whole distribution and the expected value represented with a vertical line (the mean).

In all previous quantification applications described before, data are represented in the conventional format of attribute-value pairs. But with the growth of social networks, many applications demand *network quantification* methods able to deal with data presented in relational networks. Networks offer additional information, as the relationships between the nodes of the network (e.g., friendship or following/follower relations in social networks) that can be exploited to model the collective behavior. While collective classification seeks to classify the interlinked objects of the



(a) Actual Distribution (b) Estimation

Fig. 1. Quantification for regression. (a) The actual distribution. (b) Two type of predictions can be made: the expected value (represented with the vertical line) and the whole distribution.

network, the goal of network quantification is to quantify them. Network data can be represented as an undirected graph $G(V, E, \mathscr{L})$, in which the network G is composed by a set of nodes or vertices V, a set of edges, E, and a set of classes, $\mathscr{L}$. The objective of the methods described in Section 12 is to exploit the information in G to obtain better prevalence estimates for the classes in $\mathscr{L}$.

2.4 Applications

Despite quantification learning is a relatively novel learning problem, several applications have been solved using quantification methods according to published studies. The group of applications described here comprise very different problems, including opinion mining, collective behavior in social networks, ecosystems evolution, portfolio credit risk, and customer support. We excluded those applications in which the final goal is to improve classifiers' accuracy by means of estimating the class priors effectively. The main reason is that those methods were not tested as quantifiers. This is, for instance, the case of Chan and Ng (2006) in which Vucetic and Obradovic (2001) and Saerens et al. (2002) were applied to boost the accuracy of word sense disambiguation systems.

Maybe the most popular application of quantification is related to opinion mining and sentiment analysis in social networks, mainly Twitter (Giachanou and Crestani 2016). Gao and Sebastiani (2015) and González et al. (2017) argue that several sentiment analysis articles follow a suboptimal approach based on predicting the class label of each individual comment. They point out that those studies aimed at estimating the percentage of the different classes (Positive/Neutral/Negative) should use quantification algorithms. Their arguments are in line with those in Section 6.1. In their experiments, quantification methods outperform state-of-the-art classification algorithms commonly used to tackle sentiment classification tasks. In Amati et al. (2014a), the authors propose a methodology that provides estimates of sentiment classes proportions in real time for huge volumes of data. Their approach is based on King and Lu (2008) and Hopkins and King (2010) and does not require manual intervention. The test dataset used is composed of about 3.2M tweets.

Sentiment analysis was also tackled by Martino et al. (2016a, 2016b) using ordinal quantification (see Section 9) in Subtask E of the SemEval 2016 Task 4. The problem here was, given a set of tweets known to be about certain topic, estimate the distribution of the tweets across the five classes of a five-point scale. Using their novel approach, they ranked first in a group of 10 participant systems.

Forman et al. (2006) present an application to automatically analyze technical-support call logs to improve customer services. The goal is to quantify the most frequent issues for every product. The authors present an efficient approach that it is applicable at industrial scale, because it does not require manual coding of calls. The accurate quantification of frequent issues can be useful for different purposes: to identify rising problems before they become epidemics, to efficiently target engineering resources to solve some issues, to provide consumers with the best diagnosis and support for most frequent problems, or reduce the cost associated with the customer service in general (e.g., optimizing human resources and reducing the total number of issues/calls).

Alaiz-Rodríguez et al. (2008) address the problem of quantifying the percentage of damaged cells in a sample. This kind of problem emerges in different biological applications, for instance, cancer screening tests and fertility studies, among others. In the latter case, developing tools able to assess semen viability is crucial in the veterinary insemination field. Several aspects can be analyzed to evaluate the quality of a semen sample, including hyperactivation, motility, and concentration of sperm cells. But it has been shown that a large proportion of spermatozoa with an intact membrane is also critical for fertilizing purposes. Traditionally, determining the proportion of intact (and damaged) membrane sperm cells was carried out manually by human experts. The goal of Alaiz-Rodríguez et al. (2008) is to design an automatic process to quantify cells proportions. The authors use a combination of computer vision techniques and quantification algorithms (Saerens et al. 2002; Forman 2005), see Sections 6.2 and 8.1. The experimental results reported with boar sperm samples using such techniques outperform previous approaches based on classification in terms of several measures, including mean absolute error, KL divergence, and mean relative error.

Tasche (2014) generalizes Probabilistic Adjusted Count (Bella et al. 2010), see Section 6.4, to the multi-class quantification case. This proposal is motivated by the problem of forecasting credit default rate of portfolios during the coming year. The idea is to interpret the problem in a collective way: Instead of forecasting the probability of default of single borrowers, the goal is to predict the total percentage of defaulting borrowers. This connects this problem to quantification in general and in particular to cost quantification.

In the field of epidemiology, there is a procedure called *verbal autopsies* where the idea is that the cause of a death can be predicted using a predefined survey of dichotomous questions (the verbal autopsy) about deceased's symptoms, prior to death, provided by a relative of the deceased. To build a training dataset, some of these cases are labelled with the cause of the death determined by an actual autopsy. Then, given a set of deceases in a population without medical death certification, the goal is to predict the distribution of the causes of such deaths. This problem was tackled by King and Lu (2008), who propose a method (see Section 8.5) to produce approximately unbiased and consistent estimates, outperforming previous approaches, including physician reviews (which are far more expensive), and methods based of expert systems and other statistical models.

Another real-world application in which quantification methods have been applied is the problem of plankton analysis. This problem has been routinely tackled using classification algorithms in the past (González et al. 2013); however, it represents a typical quantification problem. The first effort to apply quantification methods was due to Solow et al. (2001). To estimate the taxonomic composition of a sample, the authors propose the same idea found on Gart and Buck (1966), Levy and Kass (1970), and Forman (2008). Sosik and Olson (2007) also use this method to estimate the abundance of different taxonomic groups of phytoplankton sampled with imaging-in-flow cytometry. More recently, Beijbom et al. (2015) present an application to analyze morphological groups in marine samples. The study deals with two large datasets: a survey of coral reefs from Caribbean sea and a time-series dataset of plankton samples. The authors investigate several quantification methods from literature showing, for instance, that the EM algorithm proposed in Saerens et al. (2002) outperform other approaches for the plankton dataset. For the same type of applications, González et al. (2017) propose a methodology to assess the efficacy of quantifiers that takes into account the fact that the data distribution may vary between the training phase and once the model is deployed. The authors argue that, in some plankton recognition tasks, the base unit to correctly estimate the error is the sample, not the individual. Thus, model assessment processes require groups of samples with sufficient variability to provide precise error estimates.

Esuli and Sebastiani (2015) employ a quantifier based on optimizing multivariate loss functions, see Section 7.3, in the context of text classification. In their experiments, this approach outperforms other quantification algorithms in a multi-class dataset with a large number of classes (99). The method seems to be not only more accurate but also more stable, which is also a desirable property.

Tang et al. (2010) study quantification in social networks. The idea is to predict the proportion of certain classes in a network given some labelled nodes from such network. The authors show that baseline approaches underperform when labelled samples are few and imbalanced. Then, they present two quantification approaches: The first one is based on classification with a posterior correction, see Section 6.2, while the second one relies on link analysis, without requiring classification and prediction (Section 12). This method provides faster predictions, but its accuracy is not comparable to the one of Median Sweep (Section 6.5). Milli et al. (2015) also introduce two quantification methods for complex social networks. The goal is again to quantify collective behavior in the network, for instance, the distribution of the users that take part in a particular activity or that they share similar preferences or behaviors. Their proposals are able to quantify the percentage of individuals with specific characteristics, allowing the analysis of many social aspects. For example, this tool can estimate the political orientation distribution or the level of education of a target population. These methods are described in Section 12.

2.5 Changes in Data Distribution

Lately, there has been an increasing interest in the real applications that present challenges arising from data distribution changes, also named *dataset shift* problems. Several machine-learning problems fall in this category, and probably the most popular are concept drift (Gama et al. 2014), domain adaptation (Daume III and Marcu 2006), and transfer learning (Pan and Yang 2010; Weiss et al. 2016). All these problems share that the distribution used for training our model is different than the data the model will face later. Different taxonomies for these kinds of problems have been proposed according to the drift type, see Moreno-Torres et al. (2012), Kull and Flach (2014), and Webb et al. (2015). These articles characterize the changes that occur in the joint probability distribution of the covariates and the class variables, $P(\mathcal{X} \times \mathcal{Y})$. Quantification is one of those problems but its dataset-shift setting appears under different names, with *prior probability shift* being the most common (Moreno-Torres et al. 2012).

For some authors, the causal relationship between $\boldsymbol{x}$ and y determines the type of expected changes in the distribution. In this sense, Fawcett and Flach (2005) distinguish two types of problems: $\mathcal{Y} \rightarrow \mathcal{X}$ problems, in which the covariates $\boldsymbol{x}$ are causally determined by the class y, and $\mathcal{X} \rightarrow \mathcal{Y}$ problems, where y causally depends on the covariates $\boldsymbol{x}$. Spam detection is an example of the latter group, because an email is considered spam or not depending on its content, and a medical diagnosis problem is a paradigmatic example of the first group: to be affected by a disease y will produce a series of symptoms $\boldsymbol{x}$.

The joint probability of $\boldsymbol{x}$ and y, $P(\boldsymbol{x}, y)$, can be written as $P(y|\boldsymbol{x})P(\boldsymbol{x})$ in $\mathcal{X} \rightarrow \mathcal{Y}$ problems and $P(\boldsymbol{x}|y)P(y)$ in $\mathcal{Y} \rightarrow \mathcal{X}$ problems. Dataset shift occurs when at least one of these four terms changes between the training data and the test sets, and, thus, the joint distribution also varies, in symbols $P_{tr}(\boldsymbol{x}, y) \neq P_{tst}(\boldsymbol{x}, y)$. Therefore, there are several types of dataset-shift problems depending on the elements that change. In the case of quantification tasks, it is obvious that $P(y)$ changes, otherwise the desired quantifier would be trivial: It returns always a set of constants equal to the prior probabilities, $P_{tr}(y)$. Also, some authors restrict quantification tasks to $\mathcal{Y} \rightarrow \mathcal{X}$ problems; see, for instance, the definition of prior probability shift tasks in Moreno-Torres et al. (2012). We think that this is not correct; one may find examples of quantification tasks for both group of problems. For instance, quantifying consumers' opinion belongs to $\mathcal{X} \rightarrow \mathcal{Y}$ problems, while quantifying plankton morphological groups belongs to $\mathcal{Y} \rightarrow \mathcal{X}$ problems. However, one of the main reason to identify quantification tasks with $\mathcal{Y} \rightarrow \mathcal{X}$ problems is because almost all quantification algorithms (Saerens et al. 2002; Forman 2008; González-Castro et al. 2013) assume that $P(\boldsymbol{x}|y)$ does not change. This learning assumption seems easier to be fulfilled in $\mathcal{Y} \rightarrow \mathcal{X}$ problems, because y determines $\boldsymbol{x}$. In our opinion, the property that $P(\boldsymbol{x}|y)$ remains unaltered must be analysed for each particular application, independently that it belongs to $\mathcal{X} \rightarrow \mathcal{Y}$ or $\mathcal{Y} \rightarrow \mathcal{X}$ problems. Only a few methods assume that $P(\boldsymbol{x}|y)$ may change, for instance that in Hofer (2015).

3 EVALUATION OF QUANTIFICATION METHODS

3.1 Testing Sets

The evaluation of quantification models/methods is more complicated than for other learning problems. In most supervised learning tasks, performance is measured using the accuracy/precision at an individual level. Basically, we estimate the probability of predicting an individual example accurately. For instance, the experimental evaluation in a classification task tries to estimate the probability of classifying an unseen random example correctly. However, the actual performance of a given model in quantification learning should be assessed in terms of the precision for an unseen random set of examples or sample. Thus, the main difference is that the unit of evaluation is not the instance but the sample. This implies that we need a collection of samples to accurately assess the performance of a model or method. Given a model $\bar{h}$, a target loss (or score) function $L(\cdot,\cdot)$, like the ones discussed in Section 4, and a set of testing samples $\{T_1,\ldots,T_s\}$, the performance of $\bar{h}$ will be the average over the testing sets

$$Performance(\bar{h}, L, \{T_1,\ldots,T_s\}) = \frac{1}{s}\sum_{j=1}^{s} L(\bar{h}, T_j). \tag{9}$$

Notice that computing the loss of a model over a testing set, $L(\bar{h}, T_j)$, does not require one to average over the individual examples. For instance, in binary quantification, only the actual prevalence, p, and the predicted prevalence, $\hat{p}$, of the whole testing set are compared. On the other hand, the collection of testing sets should be as large as possible to obtain a precise estimate. An estimation computed over a few number of testing sets will probably be biased. However, collecting labelled testing samples is expensive for some applications, so this can be an issue in such cases.

The second problem regarding evaluation has to do with the key characteristic of quantification, which is that the data distribution changes between the model training phase and the model deployment phase. The collection of testing samples should contain sufficient variability to better evaluate the future performance of the model. Otherwise, the estimation will be, again, biased. It is worth noting that variability in terms of individuals, which is the requirement for other learning problems such as classification or regression, is easier to achieve than variability at sample level. Ideally, testing should be carried out with different samples presenting different distributions, covering the expected variations in the distribution for the target application.

3.2 Experimental Designs

Quantification, like other novel learning tasks, suffers from a lack of benchmark datasets. In the literature, we can find only a few examples that might qualify as “quantification” datasets. The problem, however, is that sometimes the datasets do not fulfill the desired requirements discussed previously. For instance, Gao and Sebastiani (2016) employ 11 tweet sentiment classification datasets, but the problem is that only one test set is available for each of them. The scores of the studied quantifiers are averaged across 11 datasets to analyze their performance.

In other applications, the datasets are more adequate. Pérez-Gállego et al. (2017) use the Sentiment140 dataset (Go et al. 2009). This dataset is composed of more than a million Twitter messages labelled as positive or negative comments. The tweets were collected between April 6, 2009, and June 16, 2009. The authors train the quantification models using the tweets from the two first days and the rest (37 days) as testing data. Maybe the number of testing sets is still low in this case.

The scenario in Esuli and Sebastiani (2015) is better. They employ RCV1-v2, a multi-label text classification benchmark. In the experimental design, the authors use the news stories of the first week as training data, and the news of the other 52 weeks for testing, considering only those classes (99) that have at least one training example. So the experiments consist on $99 \times 52 = 5,148$ testing experiments, so the number of testing sets is reasonable.

As we can see, some of these quantification datasets present the problem that the number of testing sets is limited. The authors have to merge the results from different classes that is usually more reasonable than averaging the results across non-related datasets. The other problem that some of these datasets have is that the drift is reduced, which means low variability between training and testing, at least for some classes.

For this reason, most of the experiments reported in literature employ datasets taken from other problems, like classification or regression, depending on the quantification learning task studied, see, for instance, Forman (2008), Bella et al. (2010), and Barranquero et al. (2015). In all these cases, the authors create drifted testing sets artificially. This approach has the advantage that the amount of drift can be controlled to study the behavior/performance of the models under different situations. But the drawback is that the generated testing samples may be artificial with respect to the actual data distribution of the problem.

4 PERFORMANCE MEASURES

This section contains a complete review of the performance measures that have been considered in published quantification articles. As we explained above, the performance across different testing sets is averaged using Equation (9), so the mathematical definition of each measure just focus on the computation of the score for a single test set.

4.1 Performance Measures for Binary Quantification

From the point of view of evaluation, binary quantification presents maybe more choices than the other quantification problems to select our target performance measure. The reason is that binary quantification is one of the simplest quantification problems and it can be reduced to the prediction of the percentage of the positive class, p. Thus, for a model, $\bar{h}$, and a given testing set T, we just need to compare the prevalence predicted, $\hat{p}$, with the actual one, p. Most of the measures discussed here shall be extended for multi-class quantification in the next section.

Perhaps the most popular (Tang et al. 2010; Alaiz-Rodríguez et al. 2008; Barranquero et al. 2013, 2015) performance measure for binary (and multi-class) quantification is *Kullback-Leibler (KL) Divergence*, also known as discrimination information, relative entropy, or normalized cross-entropy (see Esuli and Sebastiani (2010) and Forman (2008)). KL Divergence is a special case of the family of f-divergences and it can be defined for binary quantification as

$$KLD(\bar{h}, T) = p \cdot \log\left(\frac{p}{\hat{p}}\right) + (1-p) \cdot \log\left(\frac{1-p}{1-\hat{p}}\right). \tag{10}$$

This metric, ranging from 0 (optimum) to $+\infty$, measures the divergence between the predicted distribution, $[\hat{p}, \hat{n} = 1 - \hat{p}]$, and the actual distribution, $[p, n = 1 - p]$. The main advantage of KL divergence is that it seems more suitable for averaging over different test distributions than the regression-based metrics described below. However, KL divergence also has some drawbacks: It is not a true metric (it does not obey the triangle inequality and it is not symmetric) and it is less interpretable than other performance metrics. Besides, KLD is undefined when $\hat{p}$ is 0 or 1. To resolve these situations, it can be normalized via the logistic function (Gao and Sebastiani 2016):

$$NKLD(\bar{h}, T) = 2 \cdot \frac{e^{(KLD(\bar{h}, T))}}{1 + e^{(KLD(\bar{h}, T))}} - 1. \tag{11}$$

NKLD ranges between 1 (no divergence, $p = \hat{p}$) and 0 (total divergence).

Other choice in binary quantification is to employ regression performance measures. The regression metrics used in quantification literature are

— *Bias* measures when a binary quantifier tends to overestimate or underestimate the proportion of positives:

$$Bias(\bar{h}, T) = \hat{p} - p. \tag{12}$$

When a method overestimates p, the bias is positive, and negative otherwise. Bias is used in Forman (2005), Forman (2006), and Tang et al. (2010).

— *Absolute Error* (AE) is just the difference between both magnitudes:

$$AE(\bar{h}, T) = |\hat{p} - p|. \tag{13}$$

This is one of the most employed measure, because it is simple and easily interpretable. The averaged version over a group of test sets, *Mean Absolute Error* (MAE) is used in Tang et al. (2010), Alaiz-Rodríguez et al. (2008), and Barranquero et al. (2013).

— *Square Error* (SE) penalizes large mistakes:

$$SE(\bar{h}, T) = (\hat{p} - p)^2. \tag{14}$$

Mean Squared Error (MSE) is preferred for some authors (Bella et al. 2010; Amati et al. 2014b; Asoh et al. 2012) over MAE. The differences between both is that MAE is more robust to outliers and it is more intuitive and easier to interpret than MSE, and the advantage of MSE is that it does not assign equal weight to all mistakes, emphasizing the extreme values whose consequences may be much bigger than the equivalent smaller ones for a particular application. However, the problem of MAE and MSE is that averaging across testing samples with different prevalences presents certain issues that must be taken into account. For instance, an absolute error of 0.1 produced when the actual prevalence is 1 is not the same as when the actual value is 0.2. In the latter case, the error can be considered worse. This leads to the group of relative measures:

— *Relative Absolute Error* (RAE) is the relation between the absolute error and p,

$$RAE(\bar{h}, T) = \frac{|\hat{p} - p|}{p}. \tag{15}$$

The advantage of RAE is that it diminish the importance of errors to true class prevalence. On the other side, RAE presents some problems: Its range of values can be awkward depending on the value of p and, more importantly, it is asymmetric, unbounded, and undefined when $p = 0$. The common solution to solve this last issue is to smooth the expression, adding a small constant, ϵ, to the numerator and the denominator. Gao and Sebastiani (2016) propose $\epsilon = \frac{1}{2|T|}$. Other solution is to smooth p and $\hat{p}$. *Mean Relative Absolute Error* (MRAE), also called Mean Absolute Percentage Error (MAPE) in other contexts (Tofallis 2014), is used for instance in Alaiz-Rodríguez et al. (2008); Amati et al. (2014a); Gao and Sebastiani (2016).

— *Symmetric Absolute Percentage Error* (SAPE) is a symmetric version of RAE:

$$SAPE(\bar{h}, T) = \frac{|\hat{p} - p|}{\hat{p} + p}. \tag{16}$$

SAPE tries to solve some of the issues of RAE. A recent article (Tofallis 2014) has shown that MRAE prefers those models that systematically under-forecast when it is used in model selection processes. The log of the accuracy ratio, that is, $ln(\hat{p}/p)$, has been introduced to select less biased models. However, this measure presents the same problem as MRAE: it is undefined when p is 0. *Symmetric Mean Absolute Percentage Error* (SMAPE) (Armstrong 1978), that is, the mean of (16) over a group of testing sets applying (9), seems the best alternative considering all the above factors: it is a percentage, it is always defined and its reliability for model selection purposes is comparable to that of $ln(\hat{p}/p)$ according to Tofallis (2014). However, SMAPE is less popular than MRAE, it only appears in González et al. (2017).

Finally, the last group of performance measures for binary quantification are those that provide normalized values (Barranquero et al. 2015; Narasimhan et al. 2016). The goal is to obtain score functions that range between 1 (the optimum) and 0. Analyzing equations (13) and (14), we can observe that their respective upper bounds are $\max(p, 1-p)$ and $\max(p, 1-p)^2$. Both can be normalized using those values. Besides, these loss functions can be redefined as their counterpart score functions. Considering both factors, two normalized measures can be derived as:

— *Normalized Absolute Score* (NAS)

$$NAS(\bar{h}, T) = 1 - \frac{|\hat{p} - p|}{\max(p, 1-p)}. \tag{17}$$

— *Normalized Squared Score* (NSS)

$$NSS(\bar{h}, T) = 1 - \left(\frac{\hat{p} - p}{\max(p, 1-p)}\right)^2. \tag{18}$$

4.2 Performance Measures for Multi-Class Quantification

The first performance measures in this section are the counterpart versions of some of the measures discussed for binary quantification in the previous section. Some of them are simple averages over the set of labels, so they shall be barely described. Recall again that the performance across different testing sets is obtained using Equation (9), so the expression of each measure just computes the score for a single test set. Only those measures that have been used in some articles, for instance, Gao and Sebastiani (2016) are enumerated:

— *KL divergence*

$$KLD(\bar{h}, T) = \sum_{j=1}^{l} p_j \cdot \log\left(p_j/\hat{p}_j\right). \tag{19}$$

KL divergence is, again, not defined when the predicted prevalence $\hat{p}_j$ of any class is 0 or 1, which is even more common than in binary quantification, especially if the number of classes is large. In these cases, Equation (11) can be applied.

— *Absolute Error*

$$AE(\bar{h}, T) = \frac{1}{l}\sum_{j=1}^{l} |\hat{p}_j - p_j|. \tag{20}$$

Gao and Sebastiani (2016) proposes a normalized version based in the fact that AE ranges between 0 and $2 \cdot (1 - min_{j=1..l}\ p_j)/l$:

— *Normalized Absolute Error* (NAE) is a loss function, ranging from 0 (best) and 1 (worst)

$$NAE(\bar{h}, T) = \frac{\sum_{j=1}^{l} |\hat{p}_j - p_j|}{2 \cdot (1 - min_{j=1..l}\ p_j)}. \tag{21}$$

— *Relative Absolute Error* (RAE)

$$RAE(\bar{h}, T) = \frac{1}{l} \sum_{j=1}^{l} \frac{|\hat{p}_j - p_j|}{p_j}. \tag{22}$$

RAE is undefined when any $p_j = 0$. One way to solve it is to smooth both groups of prevalences, p_j and $\hat{p}_j$ (Gao and Sebastiani 2016; Martino et al. 2016a):

$$\frac{\epsilon + p_j}{\epsilon \cdot l + \sum_{j=1}^{l} p_j}, \tag{23}$$

in which the denominator normalizes the set of prevalences. The same expression must be applied for each $\hat{p}_j$ and then compute Equation (22). Gao and Sebastiani (2016) also introduce a normalized version of RAE:

— *Normalized Relative Absolute Error* (NRAE)

$$NRAE(\bar{h}, T) = \frac{\sum_{j=1}^{l} \frac{|\hat{p}_j - p_j|}{p_j}}{l - 1 + \frac{1 - min_{j=1..l} \ p_j}{min_{j=1..l} \ p_j}}. \tag{24}$$

In the context of plankton quantification systems, Beijbom et al. (2015) and González et al. (2017) propose *Bray-Curtis dissimilarity* (Bray and Curtis 1957). BC dissimilarity is commonly used to analyze abundance data collected at different locations in ecological studies and is defined as

$$BC(\bar{h}, T) = 1 - 2\frac{\sum_{j=1}^{l} min(p_j, \hat{p}_j)}{\sum_{j=1}^{l} p_j + \hat{p}_j} = \frac{\sum_{j=1}^{l} |p_j - \hat{p}_j|}{\sum_{j=1}^{l} p_j + \hat{p}_j}. \tag{25}$$

The Bray-Curtis dissimilarity is bound between 0 and 1, with 0 meaning that the prediction is perfect. Although the Bray-Curtis dissimilarity metric is able to quantify the difference between samples, it is not a true distance, because it does not satisfy the triangle inequality axiom.

4.3 Performance Measures for Other Quantification Problems

Only a few loss functions have been proposed for other quantification problems. Forman (2008) employs absolute error in cost quantification tasks to measure the error of the normalized cost. To the best of our knowledge, no other loss functions have been applied for cost quantification.

Given that networked data quantification is in many cases a binary or multi-class quantification problem, the performance measures used are the most classic measures for binary and multi-class quantification. For instance, Tang et al. (2010) employs bias (12), absolute error (13), and KL Divergence (10). Milli et al. (2015) just uses KL Divergence.

Earth Movers Distance (EMD) (Rubner et al. 2000) is the only measure proposed for ordinal quantification (Martino et al. 2016a). EMD is a routinely used in computer vision and is defined as

$$EMD(\bar{h}, T) = \sum_{j=1}^{l-1} \left| \sum_{a=1}^{j} \hat{p}_a - \sum_{b=1}^{j} p_b \right|. \tag{26}$$

EMD ranges between 0 (best) and $l - 1$ (worst).

Quantification algorithms for regression demand two types of performance measures quite different between them. On the one hand, classical regression metrics to measure the precision of single numerical predictions (typically the mean) are required. On the other hand, metrics for comparing distributions are also needed.

The main article on quantification for regression problems, Bella et al. (2014), uses *Relative Squared Error* (RSE). RSE compares the squared error between the estimated mean and the actual one, normalizing such error with the variance observed in the test set

$$RSE(\bar{h}, T) = \left(\frac{p - \hat{p}}{\sigma_T}\right)^2. \tag{27}$$

Besides that, divergences are usually employed for comparing empirical distributions using their cumulative distribution functions. The two-sample *Klomogorov-Smirnoff statistic* is commonly used for comparing two empirical cumulative distributions, F_n and F, and is defined as

$$KS = sup_x |F_n(x) - F(x)|. \tag{28}$$

The problem of KS statistic is that it only takes into account the point in which the two distributions differ most, which is an unreliable metric, because it ignores the shapes of the distributions. A more reliable option is to employ the average instead of the maximum. For that reason, Bella et al. (2014) prefer the *Cramér-von Mises statistic* for two samples. The U value is calculated from the empirical data sorted in increasing order $Y_V = v_1, v_2, \ldots, v_n$ and $Y_W = w_1, w_2, \ldots, w_m$. Then Y_{VW} is defined as $Y_V \cup Y_W$ with $r_1, r_2, \ldots, r_n$ being the ranks of the elements of Y_V in Y_{VW} and $s_1, s_2, \ldots, s_m$ the ranks of the elements of Y_W in Y_{VW}. Then, the U value is computed as

$$U = n \sum_{i=1}^{n} (r_i - i)^2 + m \sum_{j=1}^{m} (s_j - j)^2. \tag{29}$$

Finally, the U value is normalized as $u = \frac{U}{n \cdot m \cdot (n+m)}$ and it ranges between 0 (most similar distributions) and 2 (4/3 when n is large).

5 A TAXONOMY OF QUANTIFICATION METHODS

Before describing the main binary and multi-class quantification methods proposed in the literature, which are by far the most numerous, a possible taxonomy is introduced to better understand the differences between the approaches. This taxonomy can also be applied to other quantification problems, like ordinal quantification or regression quantification, but it is explained for binary and multi-class classification to facilitate understanding.

We may distinguish three different groups of quantification methods:

— *Classify, Count, & Correct.* This group of methods are based on classifying the examples of a sample and counting them to obtain a first estimate of the class distribution. This estimate is then corrected to obtain the final prediction.

— *Methods based on adapting traditional classification algorithms to quantification learning.* The difference with respect to the first group is that these methods adapt algorithms to the problem of quantification, for instance optimizing a quantification loss function, while the methods in the first group employ off-the-shelf classifiers. In fact, this is the only difference, because these methods also classify and count the examples and the posterior correction may be applied, too.

— *Methods based on distribution matching.* In these approaches the Classify & Count idea is used neither in any step, nor in the correction phase. Nevertheless, a classifier can be used (for instance, to represent the distribution), but examples are not classified in the sense that they are not assigned to a particular class for counting them. The idea of these methods is to modify the training distribution, typically varying $P(y)$, to match the test distribution. Several of these methods are designed for unsupervised domain adaptation problems (the class information is unknown in the testing set), and their main goal is to improve classification performance when the distribution of classes changes. The estimation of the class distribution is simply a by-product. In fact, the quantification accuracy is not analyzed in several of these articles, for example, Vucetic and Obradovic (2001) and Saerens et al. (2002).

6 METHODS BASED ON CLASSIFY, COUNT, & CORRECT

6.1 Classify & Count

The straightforward method to build a quantifier for binary and multi-class classification is to apply the *Classify & Count* (CC) approach (Forman 2008). It is as simple as this: (i) to learn a classifier, and then (ii), using such classifier, to classify the instances of the test sample, counting the proportion of each class. It is obvious that we can obtain a perfect quantifier if the classifier is also perfect. The problem is that obtaining a perfect classifier is almost impossible in real-world applications. The main issue of this approach is that the quantifier inherits the bias of the classifier. This aspect is analyzed in several articles both from a theoretical and practical perspective, see for example, Forman (2008). For instance, in binary quantification problems, if the underlying classifier tends to misclassify some examples of the positive class, then the resulting CC quantifier will underestimate the proportion of positives. Notice that, in such case, when the percentage of the positive class increases in the test set, then the number of misclassified positive examples also goes up, resulting in that the CC quantifier will underestimate the percentage of positives even more. This problem is worse in a changing environment, where the data distribution changes between the training and the testing phases. The theoretical analysis of this issue for binary quantification is based on the assumption that $P(\boldsymbol{x}|y)$ does not change (Forman 2008). Under such an assumption, the estimate obtained by the CC approach, $\hat{p}$, depends only on the characteristics of the classifier, defined by its true-positive rate (tpr) and false-positive rate (fpr), and, more importantly, on the actual prevalence (p):

$$\hat{p}(p) = p \cdot tpr + (1 - p) \cdot fpr. \tag{30}$$

This expression represents that only the tpr fraction of the positives examples (p) are counted as positives by the CC quantifier ($tpr \cdot p$), and the same happens, incorrectly, for the fpr fraction of the negatives cases $(1 - p)$, that is, $(fpr \cdot (1 - p))$. The prevalence predicted, $\hat{p}$, is the sum of both terms.

Using this relationship between the prevalence estimated by the CC approach and the actual prevalence, we can demonstrate the poor performance of CC, especially when the classes proportion significantly changes. This is proved by the following theorem:

Theorem 6.1 (Forman's Theorem). *(Forman 2008, p. 169) "For an imperfect classifier, the CC method will underestimate the true proportion of positives p in a test set for $p > p^*$, and overestimate for $p < p^*$, where p^* is the particular proportion at which the CC method estimates correctly; that is, the CC method estimates exactly p^* for a test set having p^* positives."*

The demonstration (see all the details in Forman (2008)) assumes that the CC quantifier produces a perfect prediction for a concrete prevalence,[2] denoted as p^*, and studies its behavior when the prevalence slightly changes. Assuming that $\hat{p}(p^*) = p^*$, when the prevalence changes by a quantity $\Delta \neq 0$, $p^* + \Delta$, the estimate of the CC method in such case will be

$$\begin{aligned} \hat{p}(p^* + \Delta) &= tpr \cdot (p^* + \Delta) + fpr \cdot (1 - (p^* + \Delta)) \\ &= \hat{p}(p^*) + (tpr - fpr) \cdot \Delta = p^* + (tpr - fpr) \cdot \Delta. \end{aligned} \tag{31}$$

[2]The actual value of p^* can be easily derived from Equation (30): $p^* = fpr/(1 - tpr + fpr)$. This equation can be used to compute the exact point in which each CC model in Figure 2 produces perfect estimates.

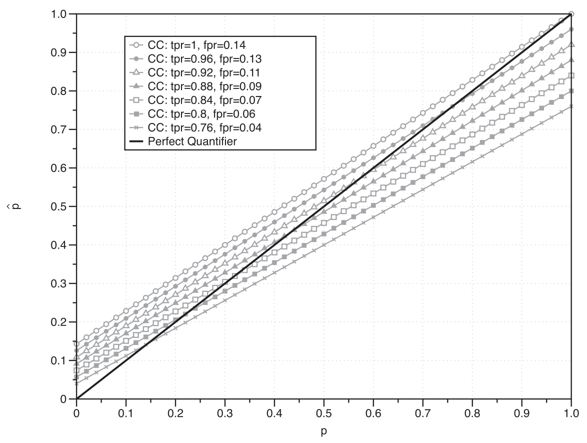


Fig. 2. The black line represents the perfect quantifier. The other lines show the theoretical estimates provided by CC method applying Equation (30) and varying the values of tpr and fpr. Despite that all the classifiers have an accuracy of 0.9, the absolute errors of the prevalence estimates are large sometimes.

Notice that the prediction of the CC method will be perfect, $\hat{p}(p^* + \Delta) = p^* + \Delta$, only when the classifier is also perfect ($tpr = 1$, $fpr = 0$ and therefore $tpr - fpr = 1$). But for the usual case, in which the classifier is imperfect ($0 \leq tpr - fpr < 1$), when the prevalence increases ($\Delta > 0$), CC underestimates it ($\hat{p} < p^* + \Delta$), and when the prevalence decreases ($\Delta < 0$) CC overestimates it ($\hat{p} > p^* + \Delta$).

An example of this behavior is illustrated in Figure 2. All the classifiers depicted have an accuracy of 0.9, but sometimes their (theoretical) absolute errors reach 0.24. The key conclusion is that an imperfect classifier (the common case in real-world applications) underestimates the prevalence of the positive class when such prevalence increases and overestimates it when decreases. This can be easily seen in the extremes of the range: when $p = 0$ any classifier with $fpr > 0$ will overestimate p, and the opposite, if $p = 1$ any classifier with $tpr < 1$ will underestimate p.

6.2 Adjusted Count

Maybe one of the most interesting and relevant quantification methods is the Adjusted Count (AC) approach proposed by Forman (2005, 2008). The main interest lies in the fact that the AC method is a theoretically perfect method when its learning assumptions are fulfilled. It is based on correcting the estimate provided by the CC approach, depending on the characteristics of the underlying classifier. An AC model is composed by two elements: a classifier (like the CC method) but also an estimation of its tpr and fpr, obtained using, for instance, cross-validation or a separated validation set. In the prediction phase, AC counts the number of examples in the testing sample predicted as positives by the classifier (following the CC approach described before) getting a first estimate, $\hat{p}_0$, that it is afterwards adjusted applying the following formula:

$$\hat{p} = \frac{\hat{p}_0 - fpr}{tpr - fpr}. \tag{32}$$

This expression is obtained by solving for the true prevalence, p, in Equation (30). $\hat{p}$ represents the adjusted prevalence of the positives in the test set, with $\hat{p}_0$ being the estimate of CC. Sometimes, this expression leads to an infeasible value of $\hat{p}$ that needs to be clipped into the range $[0, 1]$ in a final step.

Theoretically, AC returns perfect predictions, independently of the classifier's accuracy, whenever (1) $P(\boldsymbol{x}|y)$ is constant and (2) the estimations of tpr and fpr are perfect. Notice that when

$P(\boldsymbol{x}|y)$ does not vary, it ensures that tpr and fpr are independent of a change in the distribution of the classes (Fawcett and Flach 2005). Tasche (2017) proves that AC is Fisher consistent under such a perfect scenario. Unfortunately, it is rare to fulfill both conditions in real-world applications: $P(\boldsymbol{x}|y)$ shows, at least, small variations and it is difficult to obtain perfect estimations for tpr and fpr in some domains, because only small samples are available and/or they are highly imbalanced. But even in these cases, the performance of the AC method is usually better than that of the CC approach.

Actually, the correction procedure of the AC method has a long history in epidemiology (Gart and Buck 1966; Levy and Kass 1970). This same idea was already presented in the 1970s for the estimation of class proportion using screening tests. Many clinical tests are not 100% accurate, and, thus, their results are corrected taking into account the sensitivity ($sens = tpr$) and the specificity ($spec = 1 - fpr$) of the test observed in the past. The prevalence when the clinical test is applied to a given population is corrected using the following expression:

$$\hat{p} = \frac{\hat{p}_0 - (1 - spec)}{sens - (1 - spec)}. \tag{33}$$

6.3 Multi-Class AC

The AC method can be easily extended to multi-class quantification (King and Lu 2008; Hopkins and King 2010). Given a multi-class classifier h and a test set T, the probability of predicting that a random example $\boldsymbol{x}$ belongs to class ℓ_i can be written as

$$P_T(h(\boldsymbol{x}) = \ell_i) = \sum_{j=1}^{l} P(h(\boldsymbol{x}) = \ell_i | y = \ell_j) P_T(\ell_j), \tag{34}$$

in which $P(h(\boldsymbol{x}) = \ell_i | y = \ell_j)$ is the probability that h predicts class ℓ_i when the example actually belongs to class ℓ_j and $P_T(\ell_j)$ represents the actual prevalence of class ℓ_j in T. The key element is $P(h(\boldsymbol{x}) = \ell_i | y = \ell_j)$, twhich can be estimated from the training set using, for instance, cross-validation. This same expression can be written for all classes resulting in a system of l equations with l unknowns, the values of $P_T(\ell_j)$. Actually, $l - 1$ unknowns, because $\sum_{j=1}^{l} P_T(\ell_j) = 1$. The solution of this system gives us the predicted prevalence for each class, $[\hat{p}_1, \ldots, \hat{p}_l]$.

6.4 Probabilistic Adjusted Count

Bella et al. (2010) propose two probabilistic variants of the CC/AC methods. The first one, called *Probability Average* (PA), is the counterpart of CC and will be denoted here as Probabilistic CC (PCC). The only difference with respect to CC is that PCC learns a probabilistic classifier instead of a crisp one. If the classifier returns the probability that an example belongs to the positive class, then the prevalence is computed as the average of such probability for all the examples in the test sample:

$$\hat{p}_0 = \frac{1}{m} \sum_{i=1}^{m} P(y_i = +1 | \boldsymbol{x}_i). \tag{35}$$

The expected behavior of PCC should be similar to that of CC. PCC will underestimate or overestimate the actual prevalence when the distribution of the classes changes between the training and the testing phase, see Tasche (2014) (Corollary 6, p. 157). Then, the authors introduce an improved version of PCC, denoted as *Scaled Probability Average* (SPA). Here we called it *Probabilistic Adjusted Count* (PAC) to establish a parallelism with AC. The estimation obtained by PCC using

Equation (35) is adjusted applying the following formula:

$$\hat{p} = \frac{\hat{p}_0 - FP^{pa}}{TP^{pa} - FP^{pa}}, \tag{36}$$

in which TP^{pa} (*TP probability average*) and FP^{pa} (*FP probability average*) are estimated using the training data and are respectively defined as the averaged probability of the positive examples,

$$TP^{pa} = \frac{\sum_{i \in D^+} P(y_i = +1|\boldsymbol{x}_i)}{|D^+|}, \tag{37}$$

and the averaged probability of the negative examples,

$$FP^{pa} = \frac{\sum_{i \in D^-} P(y_i = +1|\boldsymbol{x}_i)}{|D^-|}, \tag{38}$$

with D^+ and D^- being the set of examples in the training data belonging, respectively, to the positive and to the negative class. The expression defined in Equation (36) is a probabilistic version of Formans correction formula (Equation (32)). The experimental performance of PCC/PAC is not as conclusive as that of CC/AC. PAC outperforms PCC in the experiments reported in Bella et al. (2010) and Tang et al. (2010) but underperforms PCC in the experimental study presented by Gao and Sebastiani (2016).

6.5 Quantification via Threshold Selection Policies

The AC quantifier produces accurate estimations for many tasks, but, according to several studies, its performance degrades, especially when the training data are highly imbalanced. Forman (2006, 2008) proposes a group of methods based on calibrated threshold classifiers to overcome this issue: (i) a threshold classifier is learned (for instance using SVM) and (ii) the threshold is adjusted. The only difference between them is the strategy to select the threshold. These methods were mainly designed for binary quantification but can be applicable for any other quantification problem that employs an underlying threshold classifier, for instance, in cost quantification (Section 10).

When the training data are imbalanced, usually because the positives examples are scarce, the performance of AC degrades severely (Forman 2006). The problem is that, in such cases, the classifier tends to predict the majority (negative) class, reducing the number of false positives, but at the cost of a low tpr, see Fawcett (2004). Mathematically, this implies a small denominator in Equation (32) that produces bigger corrections, being more vulnerable to small errors in the estimates of tpr and fpr. Formally, these methods are based on selecting a threshold that reduces the variance of the estimations for tpr and fpr. The idea is to select a threshold that increments the number of true positives but usually at the cost of incrementing the rate of false positives. Whenever $\upuparrows tpr \ggg \upuparrows fpr$ the denominator in Equation (32) increases, resulting in an adjusting method that produces smaller corrections and is more resistant to small errors in the estimates for tpr and fpr. Following this idea, Forman proposes different threshold selection policies:

- The *Max* method selects the threshold that maximizes $tpr - fpr$. This gives the biggest possible denominator in Equation (32) for the trained classifier, smoothing the posterior corrections.
- The *X* method tries to obtain a fpr equal to $1 - tpr$ to avoid the tails of both curves.
- The *T50* method picks the threshold with $tpr = 0.5$, assuming that the positives are the minority class. The idea is again to avoid the tails of the tpr curve.
- The *Median Sweep* (MS) method follows a kind of ensemble approach, computing the prevalence for all possible thresholds. MS returns the median of these values as the final prediction. Notice that this strategy requires to estimate the tpr and the fpr for such thresholds during training.

7 METHODS BASED ON QUANTIFICATION LEARNERS

All the methods described in the previous section address quantification by learning a classifier followed sometimes by a correction process aimed at compensating the bias of the classifier, underestimating or overestimating the proportion of the target class. The learning algorithms in this section are explicitly devised for quantification. The basic premise of these methods is to take into account, during training, that the model is going to be used as a quantifier.

7.1 Quantification Decision Tress

Milli et al. (2013) present a multi-class quantification method called quantification trees, based on popular decision trees. The learning process is designed to optimize the model not to be used for individual classification but for aggregated quantification. Like all decision trees learners, these algorithms have two main operations: the definition of the measure for selecting the best split on an attribute at a decision node and the stopping criteria to finish the tree growing procedure.

The most important idea of the algorithm is to employ a quantification measure for selecting the best split. Two different measures are considered:

— *Classification Error Balancing*: For every possible split and every class ℓ_j the algorithm computes:
$$E_{\ell_j} = |FP_{\ell_j} - FN_{\ell_j}|, \tag{39}$$
where FP_{ℓ_j} and FN_{ℓ_j} are the number of false positives and the number false negatives for class ℓ_j, respectively. The optimum value is $E_{\ell_j} = 0$, because the quantification of class ℓ_j will be perfect (this occurs when FP_{ℓ_j} and FN_{ℓ_j} cancel each other).

— *Classification-Quantification Balancing*: In addition to the quantification performance, this measure considers also classification performance:
$$\bar{E}_{\ell_j} = |FP_{\ell_j} - FN_{\ell_j}| \times |FP_{\ell_j} + FN_{\ell_j}|. \tag{40}$$
The second term is a classification measure, because the optimum is reached when $FP_{\ell_j} = FN_{\ell_j} = 0$, that is, there are no classification mistakes.

Notice that both measures are calculated for each class. All the values are aggregated in a single vector, $QE_1 = [E_{\ell_1}, \ldots, E_{\ell_l}]$ and $QE_2 = [\bar{E}_{\ell_1}, \ldots, \bar{E}_{\ell_l}]$, respectively, and the final score is the L2-norm of such vector, $||QE_1||_2$ or $||QE_2||_2$.

The stopping criterion also depends on the QE measure: The growing process stops when the difference between the quantification accuracy at the parent and at the child,
$$\Delta = ||QE_r^{parent}||_2 - ||QE_r^{child}||_2, \tag{41}$$
is $\Delta \leq 0$. r takes the values 1 or 2, depending on the performance measure selected.

Additionally, the authors analyze two different decision trees models: (i) a single quantification tree and (ii) a random forest, that is, a collection of k quantification trees in which each tree is built using $log_2(d + 1)$ random input features, being d the dimension of the input space. In the later case, the final prevalence predicted is the average of the predictions made by the k quantification trees.

7.2 Instance-Based Quantification

Barranquero et al. (2013) study the application of nearest neighbor (NN) methods for binary quantification. This idea is based on two main reasons: (i) weighted-based NN algorithms are well suited for learning from highly imbalanced datasets, one of the requirements of some quantification tasks, and (ii) NN algorithms present a significant advantage to build a quantifier based on the AC correction, because the method applied to estimate tpr and fpr can be more efficient and more precise. NN approaches can apply *leave-one-out* (LOO), which, theoretically, should provide more accurate estimates. Notice that LOO only computes the distance matrix once and it can be used for all folds. Other learners, like SVM, are limited to use cross-validation with a reduced number of folds.

The authors introduce two k-NN algorithms (PWK and PWK$^\alpha$) based on this weighted NN rule: $\hat{y}_i = sign(\sum_{j \sim i} w_{y_i} y_j)$, where $j \sim i$ refer to the k-nearest neighbors of the instance $\boldsymbol{x}_i$ and w_{y_i} is the class weight used to balance the relevance of the classes. The difference between them is the way in which w_{y_i} is computed. The weighting policies depend basically on class proportions and their goal is to reduce the bias in favor of the majority class. PWK$^\alpha$ uses the expression

$$w_{\ell_j}^{(\alpha)} = \left(\frac{m_{\ell_j}}{M}\right)^{-1/\alpha}, \text{with } \alpha \geq 1. \tag{42}$$

The weight of each class ℓ_j is the adjusted ratio between the cardinality of that class (m_{ℓ_j}) and the cardinality of the minority class (M) in the training set. Therefore, the weight decreases when the cardinality of the class increases. The expression for PWK is even simpler:

$$w_{\ell_j} = 1 - \frac{m_{\ell_j}}{m}. \tag{43}$$

It is easy to see that the weight w_{ℓ_j} is inversely proportional to the size of class ℓ_j with respect to the total size of the training data, represented by m.

The key benefit of PWK$^\alpha$ over PWK is that PWK$^\alpha$ is able to adapt the quantifier to each domain using the parameter α. Usually, when α grows precision is increased and recall decreases. The drawback is that PWK$^\alpha$ requires to calibrate α during training. Interestingly, PWK$^\alpha$ defines a general framework that encapsulates both KNN and PWK: When α tends to infinity the weight of both classes is 1 (KNN), and when $\alpha = 1$, PWK$^\alpha$ is equivalent to PWK (see Barranquero et al. (2013)). Thus, the parameter α defines a tradeoff between KNN and PWK. The experiments reported in the article suggest that there is no statistical difference between PWK$^\alpha$ and PWK.

7.3 Optimization Algorithms for Quantification

The methods belonging to this group are based on optimization and were designed for binary quantification. However, they can be easily extended to multi-class quantification. Following one of the most formal approach for learning, the idea initially proposed by Esuli and Sebastiani (2010) is to select a target quantification performance measure and to train an optimization algorithm to build the optimal model according to the training data available and the selected performance measure. Three different methods have been proposed following this approach. The differences between them are due to (i) the performance measure selected and (ii) the optimization algorithm used for training. All of them use advanced learning machines, because it is not straightforward to design an algorithm able to optimize a quantification measure. The difficulty arises because a quantification error cannot be decomposed as a combination of individual errors.

Esuli and Sebastiani (2015) propose to optimize the well-known KL Divergence (Equation (10)), using SVM^{perf} (Joachims 2005) as the learning algorithm. SVM^{perf} is able to optimize any non-linear measure computed using the contingency table, like KL Divergence. Barranquero et al. (2015) also use SVM^{perf} but with a different target measure. The authors argue that KL divergence, or any other *pure* quantification measure, does not take into account the accuracy of the underlying classifier, and thus the resulting quantifier could be built over a poor classifier. The key issue is that quantification measures produce several optimum points within the hypothesis search space (any that fulfills $FP = FN$). However, some of these hypotheses correspond to better (or more reliable) classifiers. To select a model that is at the same time a good quantifier and a good classifier, the authors introduce a family of measures, called the Q-measure, because it is inspired by the F-measure family. Q-measure combines a quantification metric (Q_{perf}) with a classification metric

(C_{perf}):

$$Q_\beta = (1+\beta^2) \cdot \frac{C_{perf} \cdot Q_{perf}}{\beta^2 \cdot C_{perf} + Q_{perf}}. \tag{44}$$

To be appropriately combined, C_{perf} and Q_{perf} must be bounded, for instance, between 0 and 1. The β parameter allows to tradeoff between them. The selection of the two metrics mainly depends on the problem at hand. For instance, the authors select *NAS* (17) as Q_{perf} and *recall* as C_{perf} in their experiments. Notice that combining quantification and classification measure is also employed in quantification trees, see Section 7.1.

SVM^{perf} suffers from two drawbacks: (i) the structural SVM surrogate is not necessarily a tight surrogate for all performance measures, which can lead to a suboptimal model, and (ii) it scales poorly when the amount of training data increases. To alleviate these issues, Narasimhan et al. (2016) devised two online stochastic optimization algorithms that are able to optimize quantification performance measures. Regarding this, two families of multivariate performance measures are considered: nested concave and pseudo-concave measures. For the first group, the authors analyze three nested concave functions: (1) $-KLD$, (2) $BAKLD = C \cdot BA + (1-C) \cdot -KLD$, in which C is a constant and BA stands for Balanced Accuracy, $BA = \frac{1}{2}(tpr + tnr)$, being tnr the true negative rate, and (3) Q-measure (44), using BA as C_{perf} and NSS (18) as Q_{perf}. In the group of pseudo-concave performance measures, two metrics are considered: (1) $CQReward = BA/(2-NSS)$ and (2) $BKReward = BA/(1+KLD)$. Their methods offer much faster and accurate quantification performance on a group of datasets.

Recently, Tasche (2016) investigates whether these approaches actually work without applying the AC correction. After an interesting theoretical discussion, the study concludes that quantification without adjustments *a priori* may work if proper calibration on the training dataset is ensured and the drift in class distribution is small.

7.4 Ensembles for Quantification Learning

Pérez-Gállego et al. (2017) introduce the first ensemble algorithm designed for binary quantification. The main idea of this method is that we can learn an ensemble with a proper diversity that takes into account the characteristics of our quantification task. This kind of ensemble will be better adapted to deal with the changes in data distribution of the quantification problem because its design depends on the expected variation of $P(y)$. The algorithm has three main processes:

(1) Sample generation. This step generates the samples for training the models of the ensemble. Each training sample has the same size that the training dataset, but its prevalence is different. The process to generate each sample works as follows. First, p_j, the sample prevalence, is fixed randomly in a predefined range $[p_{min}, p_{max}]$ that depends on the expected drift in the distribution. Then, the examples for the sample are selected using random sampling with replacement (to guarantee that $P(\boldsymbol{x}|y)$ is constant) until the distribution $[p_j, 1-p_j]$ is obtained.

(2) Training phase. The quantifiers of the ensemble, $\{\bar{h}_1, \ldots, \bar{h}_k\}$, are learned by training the selected quantification algorithm over each generated training sample. Two different quantification methods are considered in the article: AC (Equation (32)) (see Section 6.2) and HDy (see Section 8.4).

(3) Prediction. Each model $\bar{h}_j$ of the ensemble is applied to obtain the prevalence estimate, $\hat{p}_j$, of the unlabeled test set T. The proposed aggregation function is the arithmetic mean, thus the final prediction is just $mean(\hat{p}_1, \ldots, \hat{p}_k)$.

The authors argue that the application of an ensemble approach to quantification tasks should produce more advantages than its use in other learning problems, for example, classification. The advantage of using a collection of models instead of a single one is common to all learning problems where ensemble algorithms are applied: an individual model presents the risk of performing poorly in some situations; theoretically, an ensemble reduces such risk. However, there are additional benefits in the quantification case: (i) The knowledge about the learning problem helps to introduce an adequate diversity into the ensemble and (ii) using multiple models reduces the issues of the base quantification algorithm, especially for those methods that correct their estimates. The AC correction (Equation (32)) is sometimes unstable, despite it theoretically returns perfect estimations. When the estimates of tpr and fpr are inaccurate, the correction will cause inappropriate changes (Forman 2008; Pérez-Gállego et al. 2017) (this is in part the motivation of the methods in Section 6.5). Using ensembles reduces both risks: the risk of having a bad single classifier and the risk of a poor posterior correction.

8 METHODS BASED ON DISTRIBUTION MATCHING

This section describes several methods that are based on matching the training distribution and the testing distribution. These methods do not need to classify individual examples and count them. The idea of most of them is to adjust some parameter to modify the training distribution trying to fit the testing distribution. Typically, this parameter is the predicted class distribution ($[\hat{p}, 1-\hat{p}]$ in a binary problem). In other words, searching for the value of $\hat{p}$ that makes a modified version of the training distribution most similar to the testing distribution. The effectiveness of this idea depends on a key constraint that is the support condition, that is, $\forall \boldsymbol{x}, P_{tr}(\boldsymbol{x}) = 0 \rightarrow P_{tst}(\boldsymbol{x}) = 0$ (Huang et al. 2007). Roughly speaking, this means that the training set must be richer than the test set; otherwise, these methods cannot match well both distributions for regions in which $P_{tst}(\boldsymbol{x}) \neq 0$, but $P_{tr}(\boldsymbol{x}) = 0$. However, several articles described in this section do not make clear this assumption.

These methods usually have two main components: (i) a mechanism to represent distributions or, more formally, a density estimation method and (ii) a metric or a criterion to compare two given distributions. In fact, Firat (2016) unifies several quantification methods under a common framework defined by a constrained multi-variate regression model for quantification defined as

$$\mathbf{T} = \mathbf{D}\hat{\mathbf{p}} + \epsilon, \qquad \text{s.t.} \quad \hat{p}_j \geq 0, \quad \sum_{j=1}^{l} \hat{p}_j = 1. \tag{45}$$

$\mathbf{T}$ is a representation of the testing distribution, $\mathbf{D}$ the training distribution, and $\hat{\mathbf{p}}$ the class distribution. The goal is to obtain the values of $\hat{\mathbf{p}}$ that minimize the difference between both distributions given a loss function. Therefore, the framework comprises the same two elements enumerated before. At least three of the approaches described in this section can be derived from this framework: the Mixture Model (see Section 8.3), the methods based on the Hellinger Distance (see Section 8.4), and King and Lu's method (2008) (see Section 8.5).

Following this same idea, there are algorithms that also estimate the class distribution but their final goal is not to accurately predict such estimate but to improve the classifier accuracy. There is a lot of literature on this line of research, see, for instance, (Provost and Fawcett 2001) and Fawcett (2004) for some seminal works. The algorithms included here are designed for unsupervised domain adaptation problems. These kind of techniques are aimed at tackling those situations in which the domain changes, that is, when a drift in the distribution is expected or detected. In such cases, characterizing the new distribution can help to boost the robustness and precision of the classifier.

8.1 An Algorithm Based on EM

This is probably the most popular algorithm of this group, at least in quantification articles. Saerens et al. (2002) introduce a classification method that is designed to deal with those situations in which the distribution of the classes change. The main goal is not to estimate the class distribution but to adjust the classifier according to the class distribution drift without the need of re-training it. A minor limitation of this approach is that it can only be applied to probabilistic classifiers. Interestingly, Peters and Coberly (1976) studied this very same iterative approach, using densities instead of conditional probabilities for maximum likelihood estimate of mixture proportions.

Given a training set D, the algorithm learns a probabilistic classifier that it is able to estimate $P_{tr}(y{=}{+}1|x)$. Based on probability theory, the optimal Bayes classifier for the testing data can be modeled using the classifier obtained with the training data, so there is no need to re-train the classifier. The learning assumptions are the same than in other quantification methods, that is, $P_{tr}(y) \neq P_{tst}(y)$, but $P_{tr}(x|y) = P_{tst}(x|y)$. Taking into account that $P_{tr}(y) \neq P_{tst}(y)$, the model for the testing set, $P_{tst}(y{=}{+}1|x)$, should be different than that obtained from the training data, $P_{tr}(y{=}{+}1|x)$. The aim is to obtain $P_{tst}(y{=}{+}1|x)$ using $P_{tr}(y{=}{+}1|x)$.

Applying Bayes's theorem for the training and the testing distributions, we have that

$$P_{tr}(x|y{=}{+}1) = \frac{P_{tr}(y{=}{+}1|x)P_{tr}(x)}{P_{tr}(y{=}{+}1)}, \quad P_{tst}(x|y{=}{+}1) = \frac{P_{tst}(y{=}{+}1|x)P_{tst}(x)}{P_{tst}(y{=}{+}1)}. \tag{46}$$

But given that $P_{tr}(x|y) = P_{tst}(x|y)$,

$$P_{tst}(y{=}{+}1|x) = f(x)\frac{P_{tst}(y{=}{+}1)}{P_{tr}(y{=}{+}1)}P_{tr}(y{=}{+}1|x), \tag{47}$$

with $f(x) = P_{tr}(x)/P_{tst}(x)$. Since $P_{tst}(y{=}{+}1|x){+}P_{tst}(y{=}{-}1|x){=}1$, then

$$f(x) = \frac{1}{\frac{P_{tst}(y=+1)}{P_{tr}(y=+1)}P_{tr}(y{=}{+}1|x) + \frac{P_{tst}(y=-1)}{P_{tr}(y=-1)}P_{tr}(y{=}{-}1|x)}. \tag{48}$$

Finally, the optimal model for the testing distribution is

$$P_{tst}(y = +1|x) = \frac{\frac{P_{tst}(y=+1)}{P_{tr}(y=+1)}P_{tr}(y = +1|x)}{\frac{P_{tst}(y=+1)}{P_{tr}(y=+1)}P_{tr}(y = +1|x) + \frac{P_{tst}(y=-1)}{P_{tr}(y=-1)}P_{tr}(y = -1|x)}. \tag{49}$$

The denominator ensures that *a posteriori* probabilities sum to 1. Notice that the *a posteriori* probabilities depend nonlinearly on the *a priori* probabilities. This expression implies that, even if the classifier is an optimal Bayesian model for the training set, it will not be optimal for the testing set when the *a priori* probabilities change. However, if we know the new *a priori* probabilities, then it is possible to recover an optimal model for the distribution of the testing data applying the Bayess rule. The final conclusion is that to update the classifier, we just need to estimate the class distribution in the testing data $(P_{tst}(y{=}{+}1), P_{tst}(y{=}{-}1))$. The solution proposed by Saerens et al. (2002) is based on the Expected-Maximization (EM) algorithm. Given a test set $T = \{x_1, \ldots, x_{m'}\}$, the goal is to maximize the likelihood:

$$L(x_1, \ldots, x_{m'}) = \prod_{i=1}^{m'}(P_{tst}(x_i|y = +1)P_{tst}(y = +1) + P_{tst}(x_i|y = -1)P_{tst}(y = -1)). \tag{50}$$

Since the within-class densities remain the same, we have just to estimate $(P_{tst}(y{=}{+}1), P_{tst}(y{=}{-}1))$ to maximize the likelihood. The authors propose to apply the EM algorithm arguing that there is not a closed-form solution to this problem. The steps of the algorithm are defined as follows:

$$\text{Initial} : P^0(y{=}{+}1) = P_{tr}(y{=}{+}1), \tag{51}$$

$$\text{E Step}: P^s(y=+1|x_i) = \frac{\frac{P^s(y=+1)}{P_{tr}(y=+1)} P_{tr}(y=+1|x_i)}{\frac{P^s(y=+1)}{P_{tr}(y=+1)} P_{tr}(y=+1|x_i) + \frac{P^s(y=-1)}{P_{tr}(y=-1)} P_{tr}(y=-1|x_i)}, \tag{52}$$

$$\text{M Step}: P^{s+1}(y=+1) = \frac{1}{m'} \sum_{i=1}^{m'} P^s(y=+1|x_i). \tag{53}$$

Recently, Tasche (2017) presents a nice theoretical study about this EM approach. Based on the work by Peters and Coberly (1976) that provided conditions for the convergence of the algorithm (it converges for the binary case), and on more recent articles, namely (Tasche 2014) and Du Plessis and Sugiyama (2014b), the author proves that the EM algorithm finds the true values of the class prevalences under prior probability shift, that is, when $P_{tr}(y) \neq P_{tst}(y)$ but $P_{tr}(\boldsymbol{x}|y) = P_{tst}(\boldsymbol{x}|y)$ and it is Fisher consistent (unbiased).

8.2 Iterative Methods

Vucetic and Obradovic (2001) present an iterative procedure aimed at improving the classification accuracy. The method starts by training a classifier with the original training set that is applied to obtain an estimation of the *a priori* class probabilities using a bootstrap process. According to these probabilities, the training dataset D is sampled (with replacement) to obtain a new training set, D^{new}, which is used for re-training the classifier. These two steps are repeated until convergence. The most interesting step from the point of view of quantification is the bootstrap process to estimate the class distribution. This process requires two sets: the labelled training dataset D (or D^{new}) and an unlabelled test set, T. As p_j is the actual probability of class j in T, $P(k,j) = P(h(x_i) = k|y_i = j)$ is the probability of predicting the class k when the actual class is j and $\hat{p}_k$ is the probability of predicting class k in T. Using the law of total probability, $\hat{p}_k$ is

$$\hat{p}_k = \sum_{\ell_j \in \mathcal{L}} P(k|j) \cdot p_j. \tag{54}$$

This expression can be represented as $\hat{\mathbf{p}} = \mathbf{P} \times \mathbf{p}$ in matrix form, and the actual class probability can be obtained solving $\mathbf{p} = \mathbf{P}^{-1} \times \hat{\mathbf{p}}$. The bootstrap algorithm produces a large-enough number of samples to compute the bootstrap replicates of $\hat{p}_k$ and $P(k|j)$. Finally, Equation (54) is used for calculating the bootstrap replicate of p_j. Only the valid values of p_j (those in $[0,1]$) are averaged to compute the final prevalence.

Interestingly, this method anticipates the PAC approach described in Section 6.4. In fact, Equation (36) is simply the binary case of Equation (54). For two classes, Equation (54) can be rewritten as $\hat{p}_0 = P(\hat{y} = +1|y = +1) \cdot p + P(\hat{y} = +1|y = -1) \cdot (1-p) = TP^{pa} * p + FP^{pa} * (1-p)$. The main difference between both methods is the way they compute $\hat{p}_k$ in Equation (54) and $\hat{p}_0$ in Equation (36). In the former case, the authors propose to employ bootstrapping (thus, it is solved several times) and in the latter the complete test set is used just once.

Xue and Weiss (2009) present also an iterative method, called *CDE-Iterate*. In each iteration of this method, a classifier is trained to estimate a new class distribution. The classifier is always trained with the original training dataset but using a cost-sensitive algorithm, so the cost of the false positives changes in each iteration depending on the estimate of the distribution of the classes. Initially, the cost of the false positives and false negatives is the same for the first classifier. For the next classifiers, the cost of the false positives is the distribution mismatch ratio (*dmr*), that is, the ratio between the original class distribution and the new class distribution:

$$dmr : (p/n) : (\hat{p}/\hat{n}), \tag{55}$$

in which $[p, n]$ represents the true distribution of the training set and $[\hat{p}, \hat{n}]$ the predicted distribution computed over the unlabeled testing set. The algorithm finishes when a predefined number of iterations is reached. The authors propose another version of this algorithm, called *CDE-AC*. The only difference is that the estimate class distribution is adjusted using the AC correction (Equation (32)). Tasche (2017), using some numerical examples, shows that CDE-Iterate is not Fisher consistent under prior probability shift.

8.3 Forman's Mixture Model Approach

The main idea behind this method proposed by Forman (2005) is to employ the scores produced by a binary classifier (for instance, the scores produced by a SVM model to decide when an example is positive or negative). The distribution of the scores over the testing examples is modeled using a mixture of two distributions obtained using the training examples, one from the positives and the other from the negatives. The method, called *Mixture Model* (MM), trains a classifier but just to represent the three distributions manipulated by the algorithm. The first step is to obtain the distributions of the positives and the negatives, D^+ and D^-, respectively. These distributions are computed using the scores provided by binary classifiers. Instead of training just one classifier using the whole training data, and then obtaining the scores of those training examples to compute D^+ and D^-, the method performs a many-fold cross-validation (CV); the author uses 50-CV to obtain the scores. This reduces the risk of overfitting. When the number of folds is large, the obtained classifiers will be similar, because they have in common the most part of their training examples.

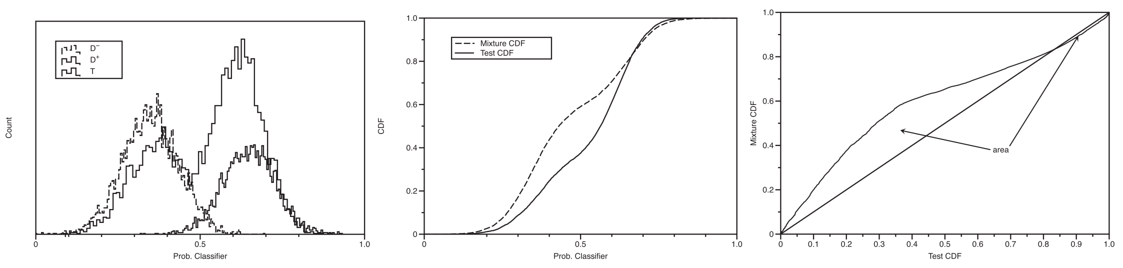
The next step is to represent the distribution of the testing examples. For this, the MM method learns a classifier (using the whole training dataset) that is applied to estimate such distribution, T. The key idea is to model this distribution as a mixture of D^+ and D^-,

$$T \approx \hat{p} \cdot D^+ + (1 - \hat{p}) \cdot D^-, \tag{56}$$

Notice that this model assumes that $P(\boldsymbol{x}|y)$ is constant, because D^+ and D^- change uniformly. The method returns the value of $\hat{p}$ that better matches the distribution of T and the mixture distribution.

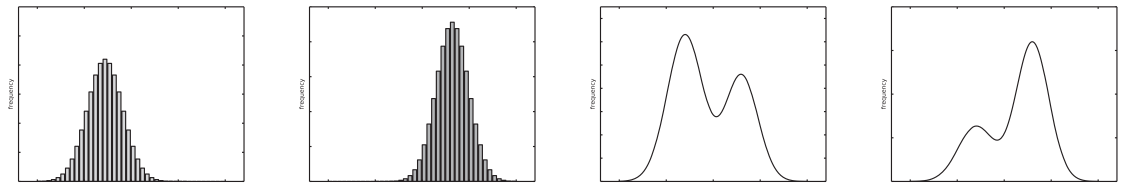
One important issue for this group of algorithms, and MM is not an exception, is how to describe the different distributions involved. Forman considers several choices, including the cumulative distribution function (CDF), the empirical probability density function (PDF), or a parametric model. Finally, the author selects CDF, arguing that PDF requires (i) discretizing the scores into artificial bins (especially if a non-probabilistic classifier is used) and (ii) additional parameters for tuning. The last reason is also used to discard a parametric model.

The other key element of MM is the procedure to measure the quality of the fit between the mixture model and the distribution of T. The Kolmogorov-Smirnov statistic is commonly used to measure the difference between two CDFs. It computes the maximum difference among all pairwise values of two CDFs. However, the author prefers another difference metric, called PP-Area. The metric is inspired in a visual comparison between the plot of the two CDFs (see Figure 3(c)). Plotting both, one versus the other, a Probability-Probability (PP) curve is drawn. A perfect 45 line is obtained when both CDFs give the same probability for each input. The level of agreement between the two distributions can be measured comparing the perfect case with the PP curve obtained. There are, again, several options to compare these two functions, like the mean error or the mean-squared error. However, the author prefers to use the area between the PP curve and the 45 line (see Figure 3). This metric presents the advantage of being commutative and monotonic. Minimizing the PP-Area corresponds to minimizing the L1-norm (Firat 2016).



(a) Negatives, Positives & Test (b) CDFs (c) PP-Area

Fig. 3. Mixture Model. First, MM computes (a) the scores distribution for the positives D^+, the negatives D^-, and the testing set T. (b) CDFs of both the test distribution and the mixture of positives and negatives examples before the fitting process. (c) PP-area before the fitting process.



(a) Negatives (b) Positives (c) $\hat{p} = 0.4$ (d) $\hat{p} = 0.7$

Fig. 4. HDx and HDy compute first the probability density functions of (a) the negatives and (b) the positives. Equation (57) combines both distributions for different values of $\hat{p}$: (c) $\hat{p} = 0.4$ and (d) $\hat{p} = 0.7$.

8.4 Matching Distributions Using the Hellinger Distance

González-Castro et al. (2013) present two algorithms based also on the idea of comparing and matching distributions. They difference between them is how to represent the distributions. However, both algorithms share a key ingredient that is the use of the Hellinger distance as the metric to assess the difference between the distributions.

The first method, called *HDx*, represents the training distribution and the testing distribution using the input feature space ($\mathcal{X}$). The authors employ discrete probability density functions (PDFs) with a parameter b to define the number of bins used to discretize each attribute f. Then, HDx computes, using the training set D, a binned distribution for each class probability density functions, that is, $P(\boldsymbol{x}|y = +1)$ and $P(\boldsymbol{x}|y = -1)$. For the test set, T, only one probability density function is computed, because no class labels are available. The strategy followed to match both distributions is to modify the combination of $P(\boldsymbol{x}|y = +1)$ and $P(\boldsymbol{x}|y = -1)$ obtained from D by means of the estimated prevalence for the positive class, that is, modifying $P(y)$ (Figure 4). The value of the pair (feature f, ith bin) for the combined distribution is computed as:

$$\frac{|D_{f,i}|}{|D|} = \frac{|D^+_{f,i}|}{|D^+|} \cdot \hat{p} + \frac{|D^-_{f,i}|}{|D^-|} \cdot (1 - \hat{p}), \tag{57}$$

where $|D^+|$ is the cardinality of the subset formed by the positive examples in D and $|D^+_{f,i}|$ is the number of positives that belong to the ith bin of the feature f. $|D^-|$ and $|D^-_{f,i}|$ refer to the same values for the negatives. Again, Equation (57) assumes that $P(\boldsymbol{x}|y)$ remains constant when the prevalence changes. In fact, both methods described in this section make this learning assumption, similarly to other methods already discussed. The similarity between both probability density functions is measured with the Hellinger distance averaged across the set of features:

$$HD(D,T) = \frac{1}{nf} \sum_{f=1}^{nf} HD_f(D,T), \tag{58}$$

in which nf is the number of features of the input space, $\mathcal{X}$, and the Hellinger Distance for a single attribute, HD_f, is defined as (discrete case):

$$HD_f(D,T) = \sqrt{\sum_{i=1}^{b} \left(\sqrt{\frac{|D_{f,i}|}{|D|}} - \sqrt{\frac{|T_{f,i}|}{|T|}} \right)^2}. \tag{59}$$

The authors use a simple linear search in which $\hat{p}$ moves over the range [0,1] in small steps to compute $D_{f,i}$ using Equation (57). The value that minimizes the Hellinger distance with respect to the testing distribution is returned as the predicted prevalence for the positive class $\hat{p}$. It is worth to note that the HDx method does not use any classifier. However, this approach is inappropriate for large input spaces, because the computational complexity of the algorithm increases with the size of $\mathcal{X}$. A similar but more complex approach is followed by Hofer and Krempl (2013). The differences with respect to *HDx* are twofold: the training and testing densities are estimated by normed Bspline basis functions and the testing prevalences are finally computed using unweighted and weighted least squares.

The second algorithm, called *HDy*, is simpler. It uses the outputs (y) of a classifier to represent the distributions, and, thus, a classifier is trained in the first step. Typically a probabilistic classifier is used to obtain a bounded output range. Then, the set of predictions for both the examples in D and the examples in T are employed to represent both distributions. Notice that these representations comprise only one dimension, so HDy can tackle applications with high-dimensional input spaces. HDy applies the same equations explained above to compare and match the distributions.

Notice that HDx/HDy methods are conceptually similar to the Mixture Model in the previous section. The differences between them are (i) the selection of the method to represent distributions, cumulative distribution function (MM) versus probability distribution function (HDx/HDy), and (ii) the metric used to compare distributions, PP-Area (or the L1-norm) versus the Hellinger distance.

8.5 King and Lu's Method

King and Lu (2008) present a non-parametric approach especially designed to estimate the cause-of-death distribution without medical death certification, using verbal autopsies (see Section 2.4). The method is clearly inspired by the AC correction (Section 6.2) and traditional methods used in epidemiology for prevalence estimation from screening tests. This approach neither requires a classifier, nor makes individual classifications. This method was popularized later by Hopkins and King (2010), where it was used to estimate document category proportions. In fact, this method is named as the Hopkins & King method in several articles.

Using the problem of estimating document category proportion as the running example, the input space, $\mathcal{X}$, is defined by a set of K word stems. Given the ith document, $x_{i,k}$ is equal to 1 if word stem k appears at least once in the document and 0 otherwise. Thus, $\boldsymbol{x}_i$ is in this case a vector that summarizes all the word stems used in the ith document, named as word stem profile. Instead of using the training data to estimate $P(h(\boldsymbol{x}_i) = \ell_j)$ and $P(h(\boldsymbol{x}_i) = \ell_j | y_i = \ell_k)$ using a classifier h and the correction via Equation (34), the content analysis (X) replaces h in that equation. The idea is that the content information is a function of the class of the document, and, thus, it is simpler to use it directly. Denoting $\boldsymbol{W}$ as any of the 2^K possible word stem profiles, we have that

$$P_{tst}(\boldsymbol{W}) = \sum_{j=1}^{l} P_{tst}(\boldsymbol{W}|\ell_j)P_{tst}(\ell_j). \tag{60}$$

This expression is applied for the 2^K possible word stem profiles, in which $P_{tst}(\boldsymbol{W})$ is estimated by direct tabulation using the testing set and $P_{tst}(\boldsymbol{W}|\ell_j)$ can be estimated using the training data by making the classical assumption that $P_{tst}(\boldsymbol{W}|\ell_j) = P_{tr}(\boldsymbol{W}|\ell_j)$, that is, $P(\boldsymbol{x}|y)$ does not change. $P_{tst}(\ell_j)$ are the l unknowns that can be obtained via constrained regression (all of them must be in $[0, 1]$ and sum to 1). This approach has two main issues: (i) the size of (60), 2^K, may be huge, especially for text classification domains in which the number of words used to describe the documents is usually large, and (ii) there are unique word stem profiles, that is, word stem profiles that appear in the training or in the testing set but not in both; the number of unique word stem profiles increases with K. To alleviate both issues, the authors propose to use a subset of few words ($\ll K$) and to apply the algorithm with several of these subsets in a kind of ensemble approach. The final estimation is obtained averaging the results for each subset of words. Thus, this process not only alleviates the aforementioned issues but also reduces the bias of the method.

8.6 Direct Estimation Methods

Many of the methods in this group correspond to the proposals of the group leaded by Sugiyama (Sugiyama et al. 2007; du Plessis and Sugiyama 2012; Sugiyama et al. 2013; Du Plessis and Sugiyama 2014a; Kawakubo et al. 2016). Instead of using the two-step procedure followed by the previous methods, where (i) two probability densities are separately estimated and then (ii) their difference is computed, these authors propose a direct estimation approach. They argue that a two-step procedure may fail, because both steps are performed without considering the other. For instance, a small estimate error in the first step can produce large mistakes in the final estimation. They propose a set of methods for directly estimating the density ratio or the density difference without estimating two densities individually. These methods can be applied to quantification tasks, despite the fact that these authors never use this term, preferring others such as class-prior estimation or even class balance change.

Given a labelled training set and a unlabelled testing set, the direct strategy to estimate $P_{tst}(y)$, which is different than $P_{tr}(y)$, is to fit a mixture of classwise densities using the training set to the test input density $P_{tst}(\boldsymbol{x})$:

$$P'_{tst}(\boldsymbol{x}) = \sum_{j=1}^{l} \hat{p}_j P_{tst}(\boldsymbol{x}|y = \ell_j) = \sum_{j=1}^{l} \hat{p}_j P_{tr}(\boldsymbol{x}|y = \ell_j). \tag{61}$$

The parameter vector $\hat{\boldsymbol{p}}$ is computed considering a particular divergence metric. Notice that the final expression is due to the classic learning assumption in quantification learning: $P_{tst}(y) \neq P_{tr}(y)$ but $P_{tst}(\boldsymbol{x}|y) = P_{tr}(\boldsymbol{x}|y)$.

Previous approaches, like those of Forman (2005) (Section 8.3) and González-Castro et al. (2013) and Hofer and Krempl (2013) (Section 8.4) estimate first $P_{tr}(\boldsymbol{x}|y)$ and $P_{tst}(\boldsymbol{x})$, using, respectively, the training set and the testing set, and, finally, they try to approximate $P'_{tst}(\boldsymbol{x})$ and $P_{tst}(\boldsymbol{x})$ given some metric. The difference between the following methods is precisely the target divergence measure.

Probably, one of the most interesting approaches of this group is presented in du Plessis and Sugiyama (2012). This article introduces a framework for class-prior estimation by means of direct divergence minimization. The KL divergence is computed between $P'_{tst}(\boldsymbol{x})$ and $P_{tst}(\boldsymbol{x})$:

$$KL(P'_{tst}||P_{tst}) = \int P_{tst}(\boldsymbol{x})\log\frac{P_{tst}(\boldsymbol{x})}{P'_{tst}(\boldsymbol{x})}\mathrm{d}\boldsymbol{x} \tag{62}$$

$$= \int P_{tst}(\boldsymbol{x})\log P_{tst}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \int P_{tst}(\boldsymbol{x})\log\left(\sum_{j=1}^{l}\hat{p}_j P_{tr}(\boldsymbol{x}|y)\right)\mathrm{d}\boldsymbol{x}. \tag{63}$$

The aim is to compute the class probability distribution, $\hat{\boldsymbol{p}}$, which minimizes the KL divergence, subject to the usual constraints, that is, $\sum_{j=1}^{l}\hat{p}_j = 1,\ \hat{p}_j \geq 0$.

This optimization problem can be solved using several methods, but the authors propose a common framework based on direct divergence minimization: Given an estimator of a target divergence from P_{tst} to P'_{tst}, for instance, KL divergence or Person (PE) divergence, learn the coefficients $\hat{\boldsymbol{p}}$ that minimizes such divergence. An f-divergence is a general divergence measure defined as

$$D_f(P_{tst}||P'_{tst}) = \int P'_{tst}(\boldsymbol{x}) f\left(\frac{P_{tst}(\boldsymbol{x})}{P'_{tst}(\boldsymbol{x})}\right)\mathrm{d}\boldsymbol{x}, \tag{64}$$

in which f is convex function such that $f(1) = 0$. Several approaches can be used to approximate the f-divergence. The first solution is following the aforementioned two-step approach to estimate $P'_{tst}(\boldsymbol{x})$ and $P_{tst}(\boldsymbol{x})$ separately and then plug these estimations into Equation (64). But as it was state above, the authors claim that this approach can be improved using a direct divergence minimization. We omit here some mathematical details (see Du Plessis and Sugiyama (2014b)), but via convex duality we can obtain the following estimator for Equation (64):

$$D_f(P_{tst}||P'_{tst}) \geq \max_r \int P_{tst}(\boldsymbol{x})r(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \int P'_{tst}(\boldsymbol{x})f^*(r(\boldsymbol{x}))\mathrm{d}\boldsymbol{x}, \tag{65}$$

This expression has nice properties because only contains expectations that can be approximated by sample averages. The maximum for a continuous f is obtained for a function r such that $P_{tst}(\boldsymbol{x})/P'_{tst}(\boldsymbol{x}) = \partial f^*(r(\boldsymbol{x}))$, which means that f-divergence is directly estimated in terms of the density ratio. This present some advantages, because the estimation of densities is a more complex problem than the estimation of a density ratio (Sugiyama et al. 2012). Using the Gaussian kernel to define a model for the density ratio $r(\boldsymbol{x})$:

$$r_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{k=0}^{b}\alpha_k\psi_k(\boldsymbol{x}), \tag{66}$$

being $\boldsymbol{\alpha}$ the parameters and $\boldsymbol{\psi}$ the basis functions: $\psi_k(\boldsymbol{x}) = \exp(-\frac{||\boldsymbol{x}-\boldsymbol{x}_k||^2}{2\rho^2})$ and $\psi_0(\boldsymbol{x}) = 1$. The constant basis function is included to deal with two equal distributions ($r(\boldsymbol{x}) = 1$). The final element is a regularizer to avoid overfitting that it is defined as $\lambda\boldsymbol{\alpha}^\top\boldsymbol{R}\boldsymbol{\alpha}$, being $\boldsymbol{R}$ a squared matrix of size $b+1$:

$$\begin{bmatrix} 0 & \boldsymbol{0}_{1\times b} \\ \boldsymbol{0}_{b\times 1} & \boldsymbol{I}_{b\times b} \end{bmatrix}.$$

Finally, the model for the density ratio is obtained solving this optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{k=0}^{b}\frac{\alpha_k}{m'} - \sum_{j=1}^{l}\frac{\hat{p}_j}{m_{\ell_j}}\sum_{i:y_i=y} f^*\left(\sum_{k=0}^{b}\alpha_k\psi_k(\boldsymbol{x}_i)\right) - \lambda\sum_{k=0}^{b}\sum_{k'=0}^{b}\alpha_k\alpha_{k'}R_{k,k'}. \tag{67}$$

To apply this framework to a f-Divergence, like KL Divergence or PE Divergence, we have to chose the function f and its convex conjugate f^*. The method presents the additional advantage that the parameters, for example, the regularization parameter or the Gaussian width, can

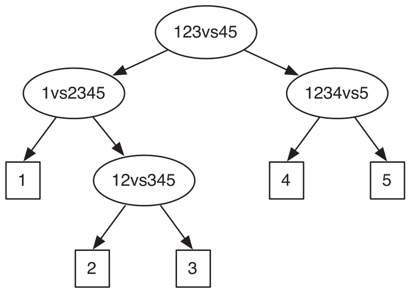


Fig. 5. Example of an Ordinal Quantification Tree (OQT) for a problem with five classes. Classes ℓ_j are represented by numbers for clarity and brevity.

be adjusted by cross-validation. The main drawback is the computational complexity, because the computation of the divergence estimator requires that $\boldsymbol{\alpha}$ must to be optimized for each $\hat{p}_j$.

The rest of the methods conceptually follow a similar strategy. The differences between them are basically twofold: the way for matching the distributions and the loss function used. In Sugiyama et al. (2013), the method is aimed at estimating the difference, in terms of the $L2$-distance, between two probability densities, $P_{tr}(\boldsymbol{x}) - P_{tst}(\boldsymbol{x})$. In Sugiyama et al. (2013), the idea is to obtain an estimate of the density ratio, $P_{tst}(\boldsymbol{x})/P_{tr}(\boldsymbol{x})$, and it is adapted for two different measures (KL and PE). Kawakubo et al. (2016) introduce the energy distance and propose to use it in class-prior estimation. This last method is related to Zhang et al. (2013) and Iyer et al. (2014), which analyze the use of the Maximum Mean Discrepancy (MMD) with kernels for class ratio estimation. The energy distance is a special case of MMD, and the advantage of Kawakubo et al. (2016) is that no parameters need to be tuned, and thus it is highly computationally efficient.

9 ORDINAL QUANTIFICATION

To the best of our knowledge, only one algorithm has been proposed to tackle ordinal quantification problems. Recently, Martino et al. (2016a) propose the idea of exploiting the apparently good performance of PCC (see Section 6.4) for sentiment quantification (Gao and Sebastiani 2015) to build a ordinal quantification model composed by a collection of binary PCC's arranged in a decision tree. This method, called *Ordinal Quantification Trees* (OQT), is also based on the decomposition proposed by Frank and Hall (2001) for ordinal classification.

Given a ordinal quantification training set, with a set of clases $\{\ell_1, \ldots, \ell_l\}$, OQT learns $l-1$ binary PCC, in which each probabilistic classifier j separates two group of classes: $\{\ell_1, \ldots, \ell_j\}$ from $\{\ell_{j+1}, \ldots, \ell_l\}$. For instance, if $l = 5$, then four binary probabilistic classifiers are learned: (i) $\{\ell_1\}$ vs. $\{\ell_2, \ell_3, \ell_4, \ell_5\}$, (ii) $\{\ell_1, \ell_2\}$ vs. $\{\ell_3, \ell_4, \ell_5\}$, (iii) $\{\ell_1, \ell_2, \ell_3\}$ vs. $\{\ell_4, \ell_5\}$, and (iv) $\{\ell_1, \ell_2, \ell_3, \ell_4\}$ vs. $\{\ell_5\}$. The method has two main procedures:

(1) To arrange the binary probabilistic classifiers in the tree. OQT follows the classical divide-and-conquer strategy for building a decision tree: OQT places at the root of the tree the binary PCC model with best quantification accuracy, measured in terms of KLD using a separate validation set. The process is repeated recursively for the left branch and for the right branch until all $l-1$ PCC models are arranged. Figure 5 shows an example with five classes.
(2) To compute the prevalence for each class. Given a test set, T, for each example, $\boldsymbol{x}_i$, we compute first the probabilities of belonging to each class, and then we compute the average probability for each class over all the examples $\boldsymbol{x}_i \in T$. The key step is how to compute, for a testing example, the probability of each class. The posterior probability for a class is

calculated in a recursive, hierarchical way: We have to multiply the posterior probability returned for all the models that lie in the path from the root to the leaf representing such class. For instance, to compute the probability of belonging to class ℓ_3 in the OQT in Figure 5, we just need to multiply

$$P_{h_{123vs45}}(y_i = 123|\boldsymbol{x}_i) \cdot P_{h_{1vs2345}}(y_i = 2345|\boldsymbol{x}_i) \cdot P_{h_{12vs345}}(y_i = 345|\boldsymbol{x}_i).$$

Notice that the order of the models on the tree is crucial in this computation. The same models arranged differently in the tree produce a different set of probabilities for the set of classes.

10 COST QUANTIFICATION

Forman (2006, 2008) introduces several methods for cost quantification. Most of them are adaptations of his methods for binary quantification. He focuses on estimating the total cost of the positive class, denoted as S, and the approach followed is the one described in Section 2.3, that is, first the method estimates the average cost of the positive class $\widehat{C}^+$, and then multiply it by the number of positives examples predicted using a binary quantifier/classifier over a test set T:

$$S = \widehat{C}^+ \cdot \hat{p} \cdot |T|. \tag{68}$$

Forman divides his proposal in two groups: basic methods and precision correction methods.

10.1 Basic Methods

The basic methods are the following:

— *Classify & Total* (CT) is the counterpart of Classify & Count. It consists on training a classifier and then computing the cost associated with those examples predicted as positives. Given a test sample T, the total cost S is

$$S = \sum_{\boldsymbol{x}_i \in T} c_i \cdot I(h(\boldsymbol{x}_i) = +1), \tag{69}$$

in which h is a binary classifier. Notice that this expression is equivalent to

$$S = \left(\frac{1}{|T|} \sum_{\boldsymbol{x}_i \in T, I(h(\boldsymbol{x}_i)=+1)} c_i \right) \sum_{\boldsymbol{x}_i \in T} I(h(\boldsymbol{x}_i) = +1), \tag{70}$$

The first term is the average cost of the examples predicted as positives, $\widehat{C}^+$, and the second term is the number of such cases, just the strategy explained above. As it occurs with CC method, the predictions made by CT are generally biased. If h tends to produce false positives, then CT overestimates the actual cost, and the other way around. The issues of CT are even worse than those of CC, because there are two sources of error: (i) $\widehat{C}^+$ is a bad estimate of C^+ and (ii) the estimation of positives cases is biased. In binary quantification, a false positive and a false negative compensate each other because they have the same cost. But in cost quantification their costs are usually different,[3] typically $C^+ > C^-$. The next methods try to reduce both sources of error.

— *Grossed-Up Total* (GUT). After computing S using the CT method, S is adjusted according to the ratio between the estimate $\hat{p}$ provided by a quantifier and the number of examples

[3]The approaches that optimize quantification measures are less applicable in these problems for the same reason.

classified as positives by the binary quantifier. GUT estimates the total cost as

$$S' = S \cdot \frac{\hat{p}}{\frac{1}{|T|}\sum_{\boldsymbol{x}_i \in T} I(h(\boldsymbol{x}) = +1)}. \tag{71}$$

For example, if the binary classifier predicts 40% of the examples in T as positives and our favorite quantifier gives an estimation of $\hat{p} = 0.5$, then the total cost, S, will be increased by 25% (because it is multiplied by 1.25). The ratio represents the percentage of examples that were missed by the binary classifier. Obviously, the performance of GUT will improve if the estimate of the quantifier is more accurate than the one provided by the binary classifier. This tries to correct one of the problems of CT. However, GUT still has two main drawbacks: (i) the estimated average cost $\widehat{C}^+$ may be biased and (ii) S' is undefined when the number of positives predicted by the binary classifier is zero. Even when it is nearly zero, the ratio is large, producing significant variations on the initial estimate S making the whole method less stable.

— *Conservative Average Quantifier* (CAQ). The core idea of the CAQ method is to reduce the number of false positives to obtain a better estimate of C^+. Following a similar strategy wereby the quantifiers are based on threshold selection (see Section 6.5), the number of false positives is reduced by selecting a more conservative threshold: Fewer instances are predicted as positives but with higher precision, obtaining a more accurate estimate of the average cost $\widehat{C}^+$. Then, we multiply it by a good estimation of the prevalence, applying the same ratio used by the GUT method.

  The selected threshold would have 100% precision in an ideal case, but sometimes there is no such threshold. The strategy suggested by the author is to use the top K examples predicted as positives. This avoids to select a very conservative threshold, predicting only a few instances as positives, especially when positive class is scarce. The uncertainty related to the estimation of $\widehat{C}^+$ will be large when the method selects very few examples to average over. In the experiments performed in Forman (2008), two variant were evaluated: CAQ30, that takes only the top 30 predicted examples (the experiments were designed to ensure that the test sample has more than 30 positive examples) and CAQhalf, which uses the top half of the instances predicted as positives by the classifier. In the experiments, the performance of both methods is similar; however, CAQhalf seems slightly better in terms of absolute error.

10.2 Precision Correction Methods

The difference of this group of methods with respect to the previous one is that they focus on improving the classifier precision. The precision is the fraction of positive instances classified as positives by the classifier. Improving precision allows to obtain a better estimate of C^+. The goal is to characterize and correct the bias of the precision and then to compute $\widehat{C}^+$ more accurately. The methods discussed here are based on the same idea used to design the AC method (Section 6.2). Three different approaches belong to this group:

— *Precision-Corrected Average*Quantifier* (PCAQ). The straightforward approach for correcting the precision is to estimate it over a validation or test set. The problem is the distribution shift that occurs on cost quantification problems. The author propose to use a multi-step process to overcome this issue:

(1) The first step is to estimate tpr and fpr of the classifier, using typically cross-validation over the training set, as the AC method does.

(2) To estimate the proportion of positives, $\hat{p}$, in the test set using a quantification algorithm, for instance, the AC method.
(3) Finally, the precision, Pr, of the classifier over the test set is estimated by a scaling formula using tpr and fpr, similar to the AC correction,

$$\widehat{Pr} = \frac{\hat{p} \cdot tpr}{\hat{p} \cdot tpr + (1 - \hat{p}) \cdot fpr}. \tag{72}$$

Using this adjusted estimate of the precision, it is possible to obtain the average cost $\widehat{C_0^+}$ of the instances predicted as positives by the classifier:

$$\widehat{C_0^+} = \widehat{Pr} \cdot C^+ + (1 - \widehat{Pr}) \cdot C^-. \tag{73}$$

In this equation, the value of $\widehat{C_0^+}$ is computed as $\sum_{\boldsymbol{x}_i \in T, I(h(\boldsymbol{x}_i)=+1)} c_i$. The problem is that both the true average cost of the positives, C^+, and of the negatives, C^-, are unknowns. To compute them, especially C^+, which is our goal, we can obtain a similar equation using the test set T:

$$C_{all} = \hat{p} \cdot C^+ + (1 - \hat{p}) \cdot C^-, \tag{74}$$

in which $C_{all} = \sum_{\boldsymbol{x}_i \in T} c_i$. Solving Equations (73) and (74) to obtain the estimate of C^+, we have

$$\widehat{C^+} = \frac{(1 - \hat{p})\widehat{C_0^+} - (1 - \widehat{Pr})C_{all}}{\widehat{Pr} - \hat{p}}. \tag{75}$$

The total cost is then computed using Equation (68).

However, this method has the same problem for imbalanced datasets. This affects not only the estimation of tpr like it occurs in binary quantification but also the computation of $\widehat{C_0^+}$. The idea proposed is again the same than that for binary classification: to select a threshold that gives a worse precision (providing a greater number of predicted positives) but that it is more stable to characterize the bias of the precision. Any of the policies discussed in Section 6.5 may be applicable, but the author selected the X method in Forman (2008).

—*Median Sweep of PCAQ* (MS-PCAQ). The implementation of this method is quite straightforward, similar to that of the MS method for binary quantification. Rather than use just a single threshold to build a PCAQ model, the process can be repeated using a collection of thresholds, obtaining an ensemble of PCAQ models. The aggregation rule proposed is again the median.

However, for this method, the author suggests three possible policies to discard some of the potential thresholds: (i) the number of predicted positives is lower than a predefined minimum (the value suggested is 30), (ii) the confidence interval for $\widehat{C^+}$ is too large, and (iii) the estimation for the precision $\widehat{Pr}$ was computed using fewer than 30 instances predicted as positives.

—*Mixture Model Average*Quantifier* (MMAQ). This algorithm is based on the Mixture Model explained in Section 8.3. MMAQ considers all possible thresholds for modeling the shape of C_t curve, applying Equation (56) and solving for C^+ and C^- using linear algebra,

$$\frac{C_t}{Pr_t} = C^- \cdot \left(\frac{1 - Pr_t}{Pr_t}\right) + C^+. \tag{76}$$

Finally, C^+ can be computed using basic linear regression over a set of data points, $(\boldsymbol{x}_t, y_t)$.

11 QUANTIFICATION FOR REGRESSION

Most of the regression quantification methods are from Bella et al. (2014) using the learning setting described in Section 2.3. These methods adapt some Forman's methods (e.g., CC, AC) and are based on some heuristics devised for regression quantification problems. They can be divided into two groups: (i) approaches based on the Regress & Sum method and (ii) methods based on discretization. The predictions obtained with these models can be: a single indicator (e.g., the mean) or the form of the distribution (e.g., shape, dispersion).

11.1 Regress & Sum, Regress & Splice

The first method studied was in fact introduced (and discarded) by Forman (2008) for cost quantification and it is the counterpart of Classify & Count for classification and Classify & Total for cost quantification. The *Regress & Sum* (RS) method first trains a regression model, using its predictions over the test set to compute the mean,

$$\hat{\rho}_T^{RS} = \frac{\sum_{\boldsymbol{x}_i \in T} \hat{y}_i}{|T|}. \tag{77}$$

RS presents the same problems as CC and CT for their respective learning tasks. It mainly depends on the precision of the regression model.

The same method takes a different name, *Regress & Splice* (RS), when it is used to make predictions about the whole distribution. Both methods have deliberately the same acronym, because they are in fact the same algorithm able to predict a single indicator (Sum) or the distribution (Splice). In the latter case, RS predicts the probability that the output for a random example from the test set, T, is less than a given value v:

$$\widehat{P}_T^{RS}(v) = \frac{\sum_{\boldsymbol{x}_i \in T} I(\hat{y}_i < v)}{|T|}. \tag{78}$$

However, using this equation directly with predictions $\hat{y}_i$ obtained from the regressor produces estimated distributions very compact (low dispersion) with a highly peaked shape. The authors solve this issue, which is even more severe for the methods based on discretization (Section 11.2), altering $\hat{y}_i$ by a random (normal) jitter:

$$\hat{y}_i' = \hat{y}_i + \sigma_{\widehat{Y}_T} \cdot rnorm_{0,1}, \tag{79}$$

in which $\sigma_{\widehat{Y}}$ is the standard deviation of the set of predictions of test set T and $rnorm_{0,1}$ is a random number generated from a typified normal distribution. This equation is applied for all the methods described here in the experiments reported in Bella et al. (2014).

Based on the RS method, Bella et al. (2014) introduce the *Adjusted Regress & Sum* (ARS) method. Following a similar idea that AC and PAC methods, ARS adjusts the prediction of RS. First, ARS computes the Quantification Error (QE) of the regression over the training set D:

$$QE_D = \rho_D - \hat{\rho}_D. \tag{80}$$

This value is used to adjust RS using the following expression:

$$\hat{\rho}_T^{ARS} = \hat{\rho}_T^{RS} + \alpha \cdot QE_D. \tag{81}$$

The degree of the adjustment depends on the value of the hyper-parameter α, and it can be seen as a way to calibrate the prediction made by the RS method. The same expression can be used to estimate the whole distribution, applied to Equation (78).

11.2 Regression Methods Based on Discretization

RS-based methods suffer when the shift in the distribution is significant. In regression problems, the output value y may notably change between the training set and testing sets. For instance, some values that appears in the testing set are infrequent in the training set. To overcome this issue, Bella et al. (2014) propose to use local corrections instead of the global adjustment made by the ARS method. In the methods based on discretization, the correction of the predicted value for a testing instance will only depend on the training examples that have a similar y value.

This group of methods works as follows:

(1) As $Y = \{y_1, \ldots, y_m\}$ is the set of outputs in the training dataset, Y is discretized into k bins.
(2) The average of each bin is computed, obtaining a sequence of prototypes $\{m_1, \ldots, m_k\}$.
(3) The prediction of each testing instance, $\hat{y}_i$, is replaced by its corresponding prototype value. For instance, if $\hat{y}_i$ belongs to bin j, then the final prediction for this example will be m_j.
(4) RS is applied over the modified predictions for the whole testing set.
(5) Optionally, the prediction obtained can be adjusted using a local version of ARS (LARS):

$$\hat{y}_i^{LARS} = m_j + \alpha \cdot QE_j, \tag{82}$$

that is, the mean value of bin j is corrected using its quantification error. QE_j is computed using Equation (80) with the training examples that belong to bin j.

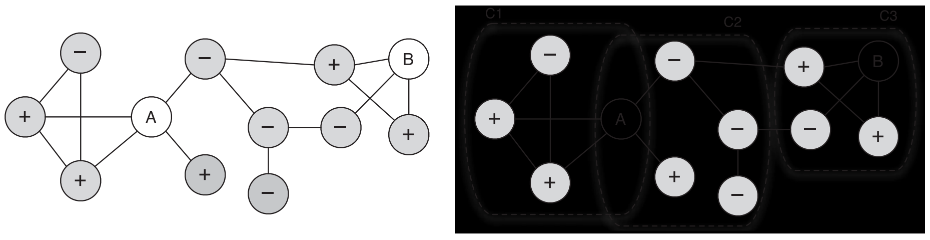
One may think that this method can suffer a lack of granularity. Regarding this aspect, several considerations should be made: (i) the method is designed for quantification purposes, not for traditional, instance regression; (ii) when the goal is to predict the mean is just the contrary, this method eliminates outliers; and (iii) only when the goal is to make predictions on the distribution the lack of granularity can be an issue but is easily solved using a large value of k. Also notice that k can be fixed when the range of y values is known. It is worthwhile to note that this approach is independent of the regression learner and the discretization method used. In their experiments, the authors employ three different discretization methods (equal width, equal frequency, and k-means) and two possibles values for k ($\sqrt{m}$ and $log_2(m+1)$). This gives a total of 12 methods due to the use or not of the local adjustment (LARS). In the experiments, the best combination employs equal frequency as the discretization method and the local adjustment procedure.

12 NETWORK QUANTIFICATION

Given a network G represented by an indirect graph, $G(V, E, \mathscr{L})$, defined by the set of nodes (V), the set of edges (E), and the set of classes ($\mathscr{L}$), the objective of network quantification methods is to exploit this information to estimate the prevalence of the classes in $\mathscr{L}$. Tang et al. (2010) presents the first method for network quantification. Besides applying most of the approaches based on Classify, Count, & Correct (see Section 6), the authors introduce a *Link-Based Quantifier* (LBQ) to exploit the homophily (i.e., "love of the same") effect that can be observed in many social networks. Inspired by the main relation of binary quantification (30), the connections of each network node are modeled using a mixture of two distributions conditioned on both classes. The probability that a node in V is connected to a given node v can be estimated as

$$P(v) = P(v|+) \cdot p + P(v|-) \cdot (1 - p), \tag{83}$$

in which $P(v|+)$ and $P(v|-)$ are the probability of a connection to v from a positive and a negative node, respectively. Note that $P(v)$, $P(v|+)$, and $P(v|-)$ can be easily obtained from G. Solving for



(a) Network (b) Communities

Fig. 6. Network Quantification. (a) A network with two classes $\mathcal{L} = \{+1, -1\}$. The goal is to quantify the prevalence of the unlabeled (white) nodes. (b) Three communities detected by a community discovery algorithm. C1 and C2 are overlapped.

p, we have that

$$p = \frac{P(v) - P(v|-)}{P(v|+) - P(v|-)}, \tag{84}$$

if $P(v|+) \neq P(v|-)$. Based on this expression, LBQ algorithm works as follows: (i) compute (84) for all unlabelled nodes in V; (ii) discard those nodes in which the value of p falls outside the range $[0, 1]$; and (iii) the estimate of the prevalence, $\hat{p}$, is the median of the valid values obtained. For instance, in the example depicted in Figure 6(a), the computation of Equation (84) for node A is $p = \frac{4/12-1/5}{3/5-1/5} = 0.33$.

To solve the lack of information due to a low connectivity in G, steps 1 and 2 are repeated for several k-hop neighborhoods. The k-hop neighborhood of node v is the set of vertices that are reachable from v in k hops or fewer. According to the authors, $k \leq 3$ generates enough valid estimates due to the small-world effect observed in many networks. The computation of Equation (84) with $k = 2$ for node A is $p = \frac{4/12-3/5}{4/5-3/5} = -1.33$, and the value is discarded because it falls outside $[0, 1]$.

The major issue of the LBQ method (Milli et al. 2015) is that there is no guarantee that estimates for all classes sum to 1. In the experiments in Tang et al. (2010), the prevalence is computed only for the positive class, assigning the complementary to the negative class, $[\hat{p}, 1-\hat{p}]$. However, the class distribution may vary if the algorithm is applied to the negative class, $[1-\hat{n}, \hat{n}] \neq [\hat{p}, 1-\hat{p}]$.

Milli et al. (2015) propose also two groups of techniques for network quantification to exploit the homophily effect. The main idea is to divide the network into several sub-networks to better bound homophily. Two different partitioning strategies are studied: community discovery and k-hop neighborhoods, also called *ego-networks* in this article.

The methods based on community discovery have two steps: (i) to find a group of communities and (ii) to assign a class to unlabeled nodes. Any community discovery algorithm can be applied for the first step. The class assigned is the one with highest frequency in the community to which the unlabeled node belongs. However, some community discovery algorithms return overlapping communities. In such cases, the authors propose two strategies to select the class for a given node: (i) frequency based: assigning the class with the highest relative frequency considering all the communities of the node; and (ii) density based: assigning the highest frequency class of the node's denser community (in terms of graph connectivity).

In the example of Figure 6(b), node B only belongs to community C3 with class frequencies $[\frac{2}{3}, \frac{1}{3}]$, so the label assigned is +1. In the case of node A that belongs to communities C1 and C2, the class assigned depends on the strategy: Using frequency based, the class selected is −1, because its frequency in C2 is $\frac{3}{4}$, while the frequency of class +1 in C1 is $\frac{2}{3}$. Applying a density-based strategy, the class assigned is +1, because the density in C1 is $\frac{5}{6}$ and it is just $\frac{4}{10}$ in C2. The method based on k-hop neighborhoods works as follows: (i) compute the k-hop neighborhood for each unlabelled node, and (ii) the label assigned is the one with highest frequency in the neighborhood. In the example of Figure 6(a), the method based on k-hop neighborhoods with $k = 3$ labels node A with class +1, because the frequencies in its three-hop neighborhood are $[\frac{4}{7}, \frac{3}{7}]$.

Both approaches assign class labels to isolated nodes following the distribution of the classes in the training set. For example, if the prevalence in the training set is $p = 0.7$, 70% of isolated nodes will be positive and the rest negatives. After all nodes have been labelled, the final prevalence can be obtained simply counting the nodes labelled as positives (CC approach) or applying the AC correction.

13 CONCLUSIONS

This article reviews the main approaches for quantification learning, trying to unify the contributions made so far. As this is a very new field of study, we think that it is important to gather together all the work that have been done in this area, most importantly when some of the research has been conducted aiming at improving classification algorithms, but the ideas and methods are still applicable and relevant to quantification problems.

The first conclusion that can be drawn from this article is that quantification is a discipline in its own right that should be treated separately from classification. After all, objectives for both problems are related but differ in their basis. Besides, it has been shown, both theoretically and experimentally, that the straightforward Classify & Count approach can be outperformed, at least, when the class probability distributions substantially change.

Although this survey discusses several quantification algorithms, much work has still to be done to improve quantification methods. First, more solid theoretical analyses are needed to better understand not only the behavior of these algorithms but also the learning problem in general. Complementarily, experimental studies must be improved. There is a lack of proper benchmark datasets for quantification. Most of the published experiments have been performed using classification datasets in which the different testing samples are generated artificially. In other cases, for instance, in SemEval competitions (Martino et al. 2016b), only one testing set or a small collection is available. This is clearly insufficient. Quantification experimental studies require several tens of samples, showing the actual drift in the distribution, to obtain meaningful results and conclusions.

Finally, most of the efforts have been made to tackle binary quantification, but there are other types of quantification applications that demand specific methods to find optimal solutions. For instance, many quantification problems (e.g., plankton abundance problems) have thousands of classes, and other tasks present ordinal relationships among the classes (e.g., sentiment quantification). This makes quantification learning a very interesting learning problem with many open lines of research.

REFERENCES

Rocío Alaiz-Rodríguez, Enrique Alegre-Gutiérrez, Víctor González-Castro, and Lidia Sánchez. 2008. Quantifying the proportion of damaged sperm cells based on image analysis and neural networks. In *Proceedings of the WSEAS International Conference on Simulation, Modelling and Optimization (SMO'08)*. WSEAS Press, 383–388.

Rocio Alaiz-Rodríguez, Alicia Guerrero-Curieses, and Jesús Cid-Sueiro. 2011. Class and subclass probability re-estimation to adapt a classifier in the presence of concept drift. *Neurocomputing* 74, 16 (2011), 2614–2623.

Giambattista Amati, Simone Angelini, Marco Bianchi, Luca Costantini, and Giuseppe Marcone. 2014a. A scalable approach to near real-time sentiment analysis on social networks. In *Proceedings of the International Workshop on Information Filtering and Retrieval*. 12–23.

Giambattista Amati, Marco Bianchi, and Giuseppe Marcone. 2014b. Sentiment estimation on twitter. In *Proceedings of the 5th Italian Information Retrieval Workshop (2014)*. 39–50.

Jon Scott Armstrong. 1978. *Long-range Forecasting: From Crystal Ball to Computer*. Wiley: New York.

Hideki Asoh, Kazushi Ikeda, and Chihiro Ono. 2012. A fast and simple method for profiling a population of twitter users. In *Proceedings of the 3rd International Workshop on Mining Ubiquitous and Social Environments*. 19–26.

Jose Barranquero, Jorge Díez, and Juan José del Coz. 2015. Quantification-oriented learning based on reliable classifiers. *Pattern Recogn.* 48, 2 (2015), 591–604.

Jose Barranquero, Pablo González, Jorge Díez, and Juan José del Coz. 2013. On the study of nearest neighbour algorithms for prevalence estimation in binary problems. *Pattern Recogn.* 46, 2 (2013), 472—482.

Oscar Beijbom, Judy Hoffman, Evan Yao, Trevor Darrell, Alberto Rodriguez-Ramirez, Manuel Gonzalez-Rivero, and Ove Hoegh Guldberg. 2015. Quantification in-the-wild: Data-sets and baselines. In *Proceedings of the Workshop on Transfer and Multi-Task Learning (NIPS'15)*.

Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. 2010. Quantification via probability estimators. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'10)*. IEEE, 737–742.

Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. 2014. Aggregative quantification for regression. *Data Min. Knowl. Discov.* 28, 2 (2014), 475–518.

J. Roger Bray and John T. Curtis. 1957. An ordination of the upland forest communities of southern Wisconsin. *Ecol. Monogr.* 27, 4 (1957), 325–349.

Yee Seng Chan and Hwee Tou Ng. 2006. Estimating class priors in domain adaptation for word sense disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 89–96.

Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *J. Artif. Intell. Res.* 26 (2006), 101–126.

Marthinus Christoffel du Plessis and Masashi Sugiyama. 2012. Semi-supervised learning of class balance under class-prior change by distribution matching. In *Proceedings of the International Conference on Machine Learning (ICML'12)*.

Marthinus Christoffel Du Plessis and Masashi Sugiyama. 2014a. Class prior estimation from positive and unlabeled data. *IEICE Trans. Inf. Syst.* 97, 5 (2014), 1358–1362.

Marthinus Christoffel Du Plessis and Masashi Sugiyama. 2014b. Semi-supervised learning of class balance under class-prior change by distribution matching. *Neur. Netw.* 50 (2014), 110–119.

Andrea Esuli and Fabrizio Sebastiani. 2010. Sentiment quantification. *IEEE Intell. Syst.* 25, 4 (2010), 72–75.

Andrea Esuli and Fabrizio Sebastiani. 2015. Optimizing text quantifiers for multivariate loss functions. *ACM Trans. Knowl. Discov. Data* 9, 4 (2015), 27:1–27:27.

Tom Fawcett. 2004. ROC graphs: Notes and practical considerations for researchers. *Mach. Learn.* 31 (2004), 1–38.

Tom Fawcett and Peter A. Flach. 2005. A response to webb and ting's on the application of ROC analysis to predict classification performance under varying class distributions. *Mach. Learn.* 58, 1 (2005), 33–38.

Aykut Firat. 2016. Unified framework for quantification. *arXiv preprint arXiv:1606.00868* (2016).

George Forman. 2005. Counting positives accurately despite inaccurate classification. In *Proceedings of the European Conference on Machine Learning (ECML'05)*. 564–575.

George Forman. 2006. Quantifying trends accurately despite classifier error and class imbalance. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'06)*. ACM, 157–166.

George Forman. 2008. Quantifying counts and costs via classification. *Data Min. Knowl. Discov.* 17, 2 (2008), 164–206.

George Forman, Evan Kirshenbaum, and Jaap Suermondt. 2006. Pragmatic text mining: Minimizing human effort to quantify many issues in call logs. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'06)*. ACM, 852–861.

James Foulds and Eibe Frank. 2010. A review of multi-instance learning assumptions. *Knowl. Eng. Rev.* 25, 01 (2010), 1–25.

Eibe Frank and Mark Hall. 2001. A simple approach to ordinal classification. In *Proceedings of the European Conference on Machine Learning*. Springer, 145–156.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Comput. Surv.* 46, 4 (2014), 44.

Wei Gao and Fabrizio Sebastiani. 2015. Tweet sentiment: From classification to quantification. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM'15)*.

Wei Gao and Fabrizio Sebastiani. 2016. From classification to quantification in tweet sentiment analysis. *Soc. Netw. Anal. Min.* 6, 1 (2016), 1–22.

John J. Gart and Alfred A. Buck. 1966. Comparison of a screening test and a reference test in epidemiologic studies ii. A probabilistic model for the comparison of diagnostic tests. *Am. J. Epidemiol.* 83, 3 (1966), 593–602.

Anastasia Giachanou and Fabio Crestani. 2016. Like it or not: A survey of twitter sentiment analysis methods. *Comput. Surv.* 49, 2 (2016), 28:1–28:41.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Rep. Stanford* 1 (2009), 12.

Pablo González, Eva Álvarez, Jose Barranquero, Jorge Díez, Rafael González-Quirós, Enrique Nogueira, Angel López-Urrutia, and Juan José del Coz. 2013. Multiclass support vector machines with example-dependent costs applied to plankton biomass estimation. *IEEE Trans. Neur. Netw. Learn. Syst.* 24, 11 (2013), 1901–1905.

Pablo González, Eva Álvarez, Jorge Díez, Ángel López-Urrutia, and Juan José del Coz. 2017. Validation methods for plankton image classification systems. *Limnol. Oceanogr. Methods* 15, 3 (2017), 221–237.

Pablo González, Jorge Díez, Nitesh Chawla, and Juan José del Coz. 2017. Why is quantification an interesting learning problem?*Progr. Artif. Intell.* 6, 1 (2017), 53–58.

Víctor González-Castro, Rocío Alaiz-Rodríguez, and Enrique Alegre. 2013. Class distribution estimation based on the hellinger distance. *Inf. Sci.* 218 (2013), 146–164.

Vera Hofer. 2015. Adapting a classification rule to local and global shift when only unlabelled data are available. *Eur. J. Operat. Res.* 243, 1 (2015), 177–189.

Vera Hofer and Georg Krempl. 2013. Drift mining in data: A framework for addressing drift in classification. *Comput. Stat. Data Anal.* 57, 1 (2013), 377–391.

Daniel J. Hopkins and Gary King. 2010. A method of automated nonparametric content analysis for social science. *Am. J. Polit. Sci.* 54, 1 (2010), 229–247.

Jiayuan Huang, Alex J. Smola, Arthur Gretton, Karsten Borgwardt, and Bernhard Schölkopf. 2007. Correcting sample selection bias by unlabeled data. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'07).* The MIT Press, 601–608.

Arun Iyer, Saketha Nath, and Sunita Sarawagi. 2014. Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection. In *Proceedings of the International Conference on Machine Learning (ICML'14).* 530–538.

Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning (ICML'05).* ACM, 377–384.

Hideko Kawakubo, Marthinus Christoffel Du Plessis, and Masashi Sugiyama. 2016. Computationally efficient class-prior estimation under class balance change using energy distance. *Trans. Inf. Syst.* 99, 1 (2016), 176–186.

Gary King and Ying Lu. 2008. Verbal autopsy methods with multiple causes of death. *Statist. Sci.* 23, 1 (2008), 78–91.

Meelis Kull and Peter Flach. 2014. Patterns of dataset shift. In *Proceedings of the 1st International Workshop on Learning over Multiple Contexts (LMCE'14) at ECML-PKDD.*

Paul S. Levy and Edward H. Kass. 1970. A three-population model for sequential screening for bacteriuria. *Am. J. Epidemiol.* 91, 2 (1970), 148–154.

Giovanni Da San Martino, Wei Gao, and Fabrizio Sebastiani. 2016a. Ordinal text quantification. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval.* 937–940.

Giovanni Da San Martino, Wei Gao, and Fabrizio Sebastiani. 2016b. QCRI at SemEval-2016 Task 4: Probabilistic methods for binary and ordinal quantification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval'16).* Association for Computational Linguistics, A, 58–63.

Letizia Milli, Anna Monreale, Giulio Rossetti, Fosca Giannotti, Dino Pedreschi, and Fabrizio Sebastiani. 2013. Quantification trees. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'13).* 528–536.

Letizia Milli, Anna Monreale, Giulio Rossetti, Dino Pedreschi, Fosca Giannotti, and Fabrizio Sebastiani. 2015. Quantification in social networks. In *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics.* 1–10.

José G. Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. 2012. A unifying view on dataset shift in classification. *Pattern Recogn.* 45, 1 (2012), 521–530.

Harikrishna Narasimhan, Shuai Li, Purushottam Kar, Sanjay Chawla, and Fabrizio Sebastiani. 2016. Stochastic optimization techniques for quantification performance measures. (unpublished).

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* 22, 10 (2010), 1345–1359.

Pablo Pérez-Gállego, José Ramón Quevedo, and Juan José del Coz. 2017. Using ensembles for problems with characterizable changes in data distribution: A case study on quantification. *Inf. Fusion* 34 (2017), 87–100.

Charles Peters and William A Coberly. 1976. The numerical evaluation of the maximum-likelihood estimate of mixture proportions. *Commun. Stat.-Theory. Methods* 5, 12 (1976), 1127–1135.

Foster Provost and Tom Fawcett. 2001. Robust classification for imprecise environments. *Mach. Learn.* 42, 3 (2001), 203–231.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 2000. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vis.* 40, 2 (2000), 99–121.

Marco Saerens, Patrice Latinne, and Christine Decaestecker. 2002. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neur. Comput.* 14, 1 (2002), 21–41.

Andrew Solow, Cabell Davis, and Qiao Hu. 2001. Estimating the taxonomic composition of a sample when individuals are classified with error. *Mar. Ecol.: Prog. Ser.* 216 (2001), 309–311.

Heidi M. Sosik and Robert J. Olson. 2007. Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. *Limnol. Oceanogr.: Methods* 5, 6 (2007), 204–216.

Amos J. Storkey. 2009. *Dataset Shift in Machine Learning.* The MIT Press, 3–28.

Masashi Sugiyama, Takafumi Kanamori, Taiji Suzuki, Marthinus Christoffel du Plessis, Song Liu, and Ichiro Takeuchi. 2013. Density-difference estimation. *Neur. Comput.* 25, 10 (2013), 2734–2775.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. 2007. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'07)*.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. 2012. *Density Ratio Estimation in Machine Learning*. Cambridge University Press.

Masashi Sugiyama, Makoto Yamada, and Marthinus Christoffel du Plessis. 2013. Learning under nonstationarity: Covariate shift and class-balance change. *Wiley Interdisc. Rev.: Comput. Stat.* 5, 6 (2013), 465–477.

Lei Tang, Huiji Gao, and Huan Liu. 2010. Network quantification despite biased labels. In *Proceedings of the 8th Workshop on Mining and Learning with Graphs (MLG'10) at ACM SIGKDD'10*. ACM, 147–154.

Dirk Tasche. 2014. Exact fit of simple finite mixture models. *J. Risk Financ. Manag.* 7, 4 (2014), 150–164.

Dirk Tasche. 2016. Does quantification without adjustments work?*arXiv preprint arXiv:1602.08780* (2016).

Dirk Tasche. 2017. Fisher consistency for prior probability shift. *arXiv preprint arXiv:1701.05512* (2017).

Chris Tofallis. 2014. A better measure of relative prediction accuracy for model selection and model estimation. *J. Operat. Res. Soc.* 66, 8 (2014), 1352–1362.

Slobodan Vucetic and Zoran Obradovic. 2001. Classification on data with biased class distribution. In *Proceedings of the European Conference on Machine Learning (ECML'01)*. Springer-Verlag, 527–538.

Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. 2015. Characterizing concept drift. *Data Min. Knowl. Discov.* (2015), 1–31.

Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *J. Big Data* 3, 1 (2016), 1–40.

Jack Chongjie Xue and Gary M. Weiss. 2009. Quantification and semi-supervised classification methods for handling changes in class distribution. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'09)*. ACM, 897–906.

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. 2013. Domain adaptation under target and conditional shift. In *Proceedings of the International Conference on Machine Learning (ICML'13)*. 819–827.

Received December 2016; revised June 2017; accepted June 2017

5.2 Validation methods for plankton image classification systems

Artículo publicado en la revista **Limnology and Oceanography: Methods**, Volumen 15, Edición 3. Noviembre de 2016.

Autores: *Pablo González* [1], *Eva Álvarez* [2], *Jorge Díez* [1], *Ángel-López-Urrutia* [2], *Juan José del Coz* [1]

Doi: `https://doi.org/10.1002/lom3.10151`

Resumen

En la últimas décadas, el estudio automático y análisis de muestras de plancton utilizando técnicas de visión artificial ha avanzado de manera significativa. La efectividad de estos métodos automáticos ha mejorado, alcanzando niveles de efectividad buenos. De todas maneras, los biólogos marinos se encuentran a menudo con que los sistemas de clasificación desarrollados en el laboratorio no funcionan tan bien como esperaban cuando se aplican a nuevas muestras. Este artículo propone una metodología para evaluar la eficacia de los modelos de clasificación desarrollados que tiene en cuenta el hecho de que la distribución de los datos por especie puede variar entre la fase de construcción del modelo y la fase de producción. En contraposición a la mayoría de los métodos de validación que consideran los organismos individuales con la unidad de validación, nuestro enfoque usa una validación-por-muestra, que es más apropiada cuando el objetivo es estimar la abundancia de diferentes tipos de plancton. Demostramos que en estos casos la unidad base para estimar el error correctamente es la muestra y no el individuo. De esta forma, los métodos de evaluación desarrollados requerirán grupos de muestras con la suficiente variabilidad de manera que se pueda calcular de manera precisa la eficiencia del modelo en cuestión.

[1] Centro de Inteligencia Artificial de Gijón, Universidad de Oviedo
[2] Instituto Español de Oceanografía, Gijón

LIMNOLOGY
and
OCEANOGRAPHY: METHODS

Limnol. Oceanogr.: Methods 00, 2016, 00–00
© 2016 Association for the Sciences of Limnology and Oceanography
doi: 10.1002/lom3.10151

Validation methods for plankton image classification systems

Pablo González,[1] Eva Álvarez,[2] Jorge Díez,[1] Ángel López-Urrutia,[2] Juan José del Coz[1]*
[1]Artificial Intelligence Center, University of Oviedo, Gijón, Spain
[2]Centro Oceanográfico de Gijón, Instituto Español de Oceanografía, Gijón, Asturias, Spain

Abstract

In recent decades, the automatic study and analysis of plankton communities using imaging techniques has advanced significantly. The effectiveness of these automated systems appears to have improved, reaching acceptable levels of accuracy. However, plankton ecologists often find that classification systems do not work as well as expected when applied to new samples. This paper proposes a methodology to assess the efficacy of learned models which takes into account the fact that the data distribution (the plankton composition of the sample) can vary between the model building phase and the production phase. As opposed to most validation methods that consider the individual organism as the unit of validation, our approach uses a validation-by-sample, which is more appropriate when the objective is to estimate the abundance of different morphological groups. We argue that, in these cases, the base unit to correctly estimate the error is the sample, not the individual. Thus, model assessment processes require groups of samples with sufficient variability in order to provide precise error estimates.

Since the advent of plankton-imaging systems, there has been a clear need to automate the classification of these images into taxonomic and functional categories. Despite the complexity of the problem from a learning perspective, automatic plankton classification seems to be quite good in terms of accuracy and close to that achieved by professional taxonomists (Benfield et al. 2007). The methods used when building automatic plankton recognition systems differ in many aspects, including the capture device used, image preprocessing, the considered taxonomy, the construction of the training, and test sets, the algorithm used for learning and the validation methods applied to estimate the accuracy of the overall approach. It is therefore virtually impossible to compare the results from different studies and it is not easy to extract general conclusions, except some obvious ones, like the conclusion that accuracy tends to decrease when the number of classes increases. For example, Tang et al. (1998) report accuracies of up to 92% when classifying between six classes, while other authors, like Culverhouse et al. (1996), report 83% accuracy using neural networks and classifying between 23 classes. Table 1 summarizes the diversity of methods used.

However, most of the authors of the papers listed in Table 1 would probably agree with respect to a worrying fact: the performance of plankton recognition systems degrade when they are deployed and have to work in real conditions (Bell and Hopcroft 2008). This means that the model assessment strategies employed are not able to correctly estimate the future performance of these systems. Yet, the techniques applied are those proposed in the statistical literature, like cross-validation. Acknowledging that solving this issue is difficult, the present paper exhaustively discusses it from a formal point of view and proposes a validation methodology that may help to mitigate the problem, suggesting further directions of research. Our proposal is designed to deal with the particular characteristics of plankton recognition systems, focusing on those cases in which the goal is to obtain estimates for complete samples, e.g., the abundance of different groups in unseen samples.

Why do traditional model assessment methods not work in plankton recognition systems? In our opinion, there are two main reasons why the performance of plankton recognition systems is not accurately estimated by model assessment methods.

The first has to do with an imprecise definition of what the actual prediction task is from the learning point of view and how its performance should be assessed. In many cases, error estimates during learning are provided in terms of the classification accuracy at an individual level. Basically, they estimate the probability of classifying an individual example correctly. However, many of these studies are designed to predict the total abundance of the different taxonomic or functional groups. Hence, the actual performance of the model/algorithm should be assessed in terms of the estimated abundance for each group in a sample. We believe that this dichotomy in evaluating the learned model at an

*Correspondence: juanjo@uniovi.es

Table 1. Summary of the training sets and validation methodologies used in several papers. Note that results may not be comparable due to the variety of datasets and methods used in the experiments. The abbreviations used are the following: manually selected (man. sel.) examples (ex.), classes (cl.), samples (sa.), phytoplankton (phyto.), zooplankton (zoo.), cross-validation over training sets (CV), Hold-out applied over testing sets (HO), Resubstitution (R), Accuracy (ACC), Precision (P), Recall (RE), True Positives (TP), False Positives (FP), Confusion Matrices (CM), Abundance estimate (AE), Abundance comparison with graphics (AC), Regression analysis (RA) and Kullback–Leibler Divergence (KLD).

Paper	Datasets	Validation method	Performance metrics
Jeffries et al. (1984)	315 man. sel. ex., 8 cl. (zoo.)	HO (265 ex. for training and 50 ex. for testing)	ACC (89%)
Gorsky et al. (1989)	3 cl. (phyto). 30 mL of each cl. for testing	HO (50 ex./cl. for training)	AE
Simpson et al. (1991)	100 man. sel. ex., 2 cl. (phyto.)	HO	ACC (90%)
Boddy et al. (1994)	42 cl. (phyto.) (200 man. sel. ex./cl.)	HO (100 ex./cl. for testing)	ACC (half of the cl. over 70%)
Culverhouse et al. (1996)	5000 man. sel. ex., 23 cl. (phyto.)	HO (100 ex. for training, rest for testing)	ACC (83%)
Frankel et al. (1996)	6000 man. sel. ex., 6 cl. (phyto.)	R, HO (1,000 extra ex. for testing)	ACC (98%), CM
Tang et al. (1998)	2000 man. sel. ex., 6 cl. (zoo. and phyto.)	HO (1/2 training, 1/2 testing)	ACC (95%)
Boddy et al. (2000)	1st) 61 cl. (phyto.) 2nd) 52 cl. (phyto.)	HO (500 ex./cl. for training and 500 ex./cl. for testing)	ACC (77% 1st. dataset, 73% 2nd. dataset)
Embleton et al. (2003)	235 ex., 4 cl. (phyto.)	HO (235 for training, 500 ex. for testing)	CM, AC
Luo et al. (2003)	1st) 1,258 man. sel. ex., 5 cl. 2nd) 6,000 man. sel. ex., 6 cl. (zoo. and phyto.)	10-fold CV	ACC (90% 1st. dataset, 75% 2nd. dataset), CM
Beaufort and Dollfus (2004)	4150 man. sel. ex., 11 cl. (150 ex./cl. + 2500 ex. in class others)	HO (50 ex./cl. for testing)	AC (91%), RA
Davis et al. (2004)	D_1 1,920(5cl.) D_2 1,527(7) D_3 1,671(7) D_4 1,400(7) 200ex/cl T_1 19,521(7) T_2 20,000(7) T_3 time series (zoo. and phyto.)	R, CV, HO	ACC (93% for R, 84% for CV and 63% for HO), CM, AC, RA
Grosjean et al. (2004)	1st) 1,035 man. sel. ex, 8 cl. (zoo.) 2nd) 1,127 man. sel. ex., 29 cl. (zoo.)	HO (2/3 training, 1/3 testing, 100 repetitions)	ACC (1st 85%, 2nd 75%)
Luo et al. (2004)	1st) 1,285 man. sel. ex., 5 cl. 2nd) 6,000 man. sel. ex., 6 cl. (phyto. and zoo.)	10-fold CV over both datasets	ACC (1st 90%, 2nd 75.6%), CM
Blaschko et al. (2005)	982 man. sel. ex., 13 cl. (zoo. and phyto.)	10-fold CV	ACC (71%)
Hu and Davis (2005)	20,000 man. sel. ex., 7 cl. (zoo. and phyto.)	HO (200 ex. for training, 200 ex. for testing)	ACC (72%), KLD
Lisin et al. (2005)	1826 man. sel. ex., 14 cl. (phyto. and zoo.)	10-fold CV	ACC (65.5%), CM
Luo et al. (2005)	8440 man. sel. ex., 5 cl. (phyto. and zoo.)	HO (7,440 ex. for training, 1,000 ex. for testing)	ACC (88%)
Hu and Davis (2006)	20,000 man. sel. ex., 7 cl. (zoo. and phyto.)	HO	TP, FP, CM, AC
Tang et al. (2006)	3147 man. sel. ex., 7 cl. (phyto. and zoo.)	R	ACC (91%), CM
Sosik and Olson (2007)	D 3,300 ex. (22 cl.) 150 ex./cl. T_1 3,300(22) T_2 19,000 (phyto.)	HO T_3 15 sa.	ACC (88%), R, P, AC (T_3)
Bell and Hopcroft (2008)	63 cl. 10-30 ex./cl. (zoo.)	CV, HO	ACC (82%), CM, AE, RA

TABLE 1. Continued

Paper	Datasets	Validation method	Performance metrics
Gislason and Silva (2009)	D_1 1,135 ex. (34 cl.), D_2 1,139 ex. (25 cl.), D_3 1,174 ex. (19 cl.), *T* 17sa. (zoo.)	10-fold CV, HO	ACC, CM, P, AE, RA (*T*)
Gorsky et al. (2010)	5–35 cl. (phyto. and zoo.), 300 ex./cl.	CV	TP, FP, CM, AC
Zhao et al. (2010)	3119 man. sel. ex., 7 cl. (phyto. and zoo.)	10-fold CV	ACC (93.27%), CM
Ye et al. (2011)	154,289 ex., 26 cl. (zoo.)	HO (50% for training, 50% for testing)	ACC (69%), AC
Álvarez et al. (2012)	526 sa., 86 sa. for training, 17 sa. for testing (61,700 ex.), 6 cl. (phyto. and zoo.)	HO	ACC (86%), CM, P, RE, AC
Vandromme et al. (2012)	14 cl. 9668 ex. for training (zoo.)	HO (26,027 ex. in 22 sa. for testing)	CM, AC, RA
González et al. (2013)	5145 man. sel. ex., 5 cl. (phyto. and zoo.)	Fivefold CV (repeated twice)	ACC (93,6%), P, RE
Lindgren et al. (2013)	50 sa., 5 depths, 17 cl. (zoo.)	CV	ACC (81.6%), P, AE (1, 5 sa.)
Ellen et al. (2015)	725,516 ex. (46 sa.), 24 cl. (phyto. and zoo.)	HO (80% for training, 20% for testing	RE (88% with 8 cl.), CM
Orenstein et al. (2015)	3.4 million ex., 70 cl. (phyto. and zoo.)	HO (20% for training, 80% for testing)	ACC (93.8%)
Dai et al. (2016)	9460 man. sel. ex., 13 cl. (zoo.)	HO (80% for training, 20% for testing)	ACC (93.7%)
Faillettaz et al. (2016)	1.5 million ex., 14 cl. (phyto. and zoo.)	HO (5,979 man. sel. ex. for training)	ACC (56.3%), P. for biological groups (84%), AC

individual level during training, but using it to estimate total group abundance per sample during "production" should be considered when validating plankton recognition systems. When the goal is to obtain an accurate estimate of the abundance per class, the learning problem is different to when the goal is to classify each image correctly. The former is not a classification task, as the model should return simply an estimate for the whole sample. The performance at an individual level is secondary in such cases.

There are, in fact, some methods whose final estimations are not just based on the number of examples classified for each class (Solow et al. 2001; Lindgren et al. 2013). Unfortunately, most experimental studies focus only on obtaining error estimates for individual predictions. Only a few papers analyze the performance of the model when a global magnitude, typically abundance, is predicted. Different techniques are applied in these papers:

- Confusion matrix. The abundance of each group can be estimated from a confusion matrix (Gislason and Silva 2009; Vandromme et al. 2012; Lindgren et al. 2013). The problem is that the information of the confusion matrix comprises just one sample, the complete testing set. This is equivalent to estimating the classification accuracy at an individual level using only one example.
- Graphically. Some papers use graphs to compare the actual and the predicted magnitude for a set of samples (Davis et al. 2004; Sosik and Olson 2007; Lindgren et al. 2013). The problem is that performance cannot be measured numerically using only graphs.
- Regression analysis. This is carried out to analyze the relationship between both values, observing whether they are well correlated; see, for instance, (Davis et al. 2004; Bell and Hopcroft 2008; Gislason and Silva 2009). R^2 is a good measure to assess fit accuracy, but does not measure prediction accuracy so well.

In addition of these techniques, a precise estimate of the error for the target magnitude should be provided using a group of samples. This estimate will be more useful once the model is deployed. Therefore, our unit in the model assessment process is not the individual example, but the sample, i.e., a group of individual examples. The most important element in our proposal is that the datasets should be composed of a collection of actual complete samples.

The second problem arises from another intrinsic property of plankton recognition problems: changes in data distribution (Haury et al. 1978), also called dataset shift (Moreno-Torres et al. 2012). This drift occurs when the joint distribution of inputs (description of the individuals) and outputs (classes) differs between training and test stages. For instance, when the probability of a class (e.g., diatoms) changes or when the characteristics of the individuals of such class change (e.g., the size distribution of diatoms varies) or both things together (e.g., the proportion of diatoms changes and also their size distribution). Data drifts occur in many practical applications for a number of different reasons. However, there are two well documented situations: (1) the sample selection bias introduced in the dataset used during training and/or the validation process, for instance, when the training set is manually built without representing the true underlying probability distribution, and (2) because it is impossible to reproduce the testing conditions at training time, mainly because the testing conditions vary over time and are unknown when the training set is built. Both situations may be found, at different levels, in plankton recognition studies. Focussing on the latter, plankton composition shows natural variability. The concentration of different morphological groups usually varies over space and time and this variation depends on numerous causes. However, this is precisely what the model must capture. In order to achieve this goal and also to assess its future performance, the collection of samples that compose the dataset should contain sufficient variability. So once again, variability in terms of individuals, which is the current trend, should shift to variability in terms of samples. Otherwise, it is impossible to obtain accurate estimates.

This paper makes two main contributions to the literature. The first is that of studying how changes in distribution affect the performance of classifiers and assessment strategies. The second is to put forward some guidelines and propose an appropriate model assessment methodology designed to deal with the characteristics of the aforementioned plankton recognition tasks. A relatively large dataset, composed of 60 different samples and 39,613 examples, was used to analyze both aspects. The dataset was captured using a FlowCAM (Sieracki et al. 1998) in the Bay of Biscay and off the northern coast of the Iberian Peninsula.

Material and methods

Learning task

Supervised classification tasks require as input a dataset $D=\{(x_i,y_i) : i=1\ldots n\}$, in which x_i is the representation of an individual in the input space $\mathcal{X}$ and $y_i \in \mathcal{Y}=\{c_1,\ldots,c_l\}$ is its corresponding class. The goal of a classification task is to induce from D a hypothesis or model

$$h : \mathcal{X} \rightarrow \mathcal{Y}=\{c_1,\ldots,c_l\}, \tag{1}$$

that correctly predicts the class of unlabeled query instances, x. A typical example of this kind of learning problem is the prediction of a disease. The input space, $\mathcal{X}$, would be the symptoms of the patient and h returns the most probable disease from $\mathcal{Y}$. Obviously, patients are interested in knowing how accurate h is. Hence, the assessment strategy must estimate the probability that h correctly predicts the disease of a random patient, x.

Most approaches solve plankton recognition tasks using a classifier, including those aimed at returning aggregate estimates. For instance, predicting the abundance per unit of volume for class c_j of a dataset, D, can be computed using a classifier, h:

$$\bar{h}(D,c_j)=\frac{1}{v}\sum_{x_i\in D} I(h(x_i)=c_j), \tag{2}$$

where v is the volume and $I(p)$ is the indicator function that returns 1 if p is true and 0 otherwise. This approach is called "classify and count" in the context of quantification learning (Forman 2008) as individual instances are first classified by h and then counted to compute the estimate for the whole sample, D. Formally, the aforementioned learning task takes the form $\bar{h} : \mathcal{X}^n \times \mathcal{Y} \rightarrow \mathbb{R}$, if we wish to predict one magnitude for a given class, or the form $\bar{H} : \mathcal{X}^n \rightarrow \mathbb{R}^l$, if we wish to make a prediction for all classes together. Notice that $\bar{H}$ can be computed using $\bar{h}$ in (2) because $\bar{H}(D)=(\bar{h}(D,c_1),\ldots,\bar{h}(D,c_l))$. Notice that both, $\bar{h}$ and $\bar{H}$, do not require an individual example as input, but a sample denoted as $\mathcal{X}^n$ representing a set of a variable number of instances from the original input space $\mathcal{X}$.

There are two reasons why the classify and count approach is so popular. First, it is a straightforward solution using any off-the-shelf classifier. However, it is not the only possible approach; there exist other alternatives whose analysis falls outside the scope of this paper. One such method was proposed by Solow et al. (2001) and applied by Lindgren et al. (2013). In fact, the classify and count approach is outperformed by other methods according to the quantification literature (Forman 2008; Barranquero et al. 2013).

The second reason is the false belief that if you build the best possible classifier, then you will also have the most accurate estimates at an aggregated level too. This is simply not true (Forman 2008). The only case when it is true is when you have a perfect classifier (accuracy 100%), but this never occurs in real-world applications as difficult as plankton recognition problems. Imagine, for instance, a two-class problem (positive class and negative class) with 200 examples, 100 of each class, and two classifiers, h_1 and h_2. h_1 produces 0 false positives and 20 false negatives, while for h_2, these values are 20 false positives and 20 false negatives. Classifier h_1 has an accuracy of 90%, but it does not estimate the abundance of both classes exactly. While h_2 is a worse classifier, with an accuracy of 80%, the abundance estimates are perfect. Several examples can also be found in plankton recognition papers. For instance, in Lindgren et al. (2013), according to Table 2, page 77, the precision classifying Nonionella examples is 96.7%, with a 3% of error in estimating abundance, while the precision classifying examples of the Multiparticles class is just 68.4% but the error in estimating the abundance is only 1%. In the experiments, we shall see similar examples in the case study (see Fig. 6).

Optimizing precision at an individual level does not mean improving precision at an aggregate level. The performance metrics for both problems are different, so the optimal model for one of them, is rarely also optimal for the other. The perfect classifier is simply an exception. The performance measures for samples require a kind of compensation among the whole sample, as occurs for classifier h_2 in the previous example. There are classifiers that select this kind of model for binary quantification; see, for instance, (Barranquero et al. 2015).

Our advice is that the experiments should focus on estimating the error at an aggregate level at which the ecological question is posed, typically analyzing samples for a target region. This, of course, requires datasets composed of several samples taken for such region. Recall that most often, the ecological unit of analysis is the sample and therefore the classification accuracy at an individual level should be somewhat secondary.

Datasets: representing the underlying probability distribution

One factor that has a major influence on the validation process is the way in which datasets are constructed. Learning theory establishes that training (and validation) datasets must be generated independently and identically according to the probability distribution, $P(x, y)$, on $\mathcal{X}\times\mathcal{Y}$. This is the so-called independent and identically distributed (i.i.d.) assumption, which is the main assumption made for the learning processes of most algorithms (Duda et al. 2012). When this assumption is not fulfilled, the model obtained is suboptimal with respect to the true underlying distribution, $P(x, y)$, and the performance function optimized by the algorithm (for instance, accuracy). Unfortunately, the datasets used in many plankton studies are biased. Several authors *design* their training sets, selecting *ideal* examples or fixing the number of examples manually for each class in an attempt to improve the overall accuracy, especially when some morphological groups are scarce. This is a clear case of *sampling bias* and the training set does not represent the underlying probability distribution.

Although these kinds of databases are sometimes only used for training the models, which are subsequently validated using a different testing set, sampling bias is still dangerous for the training process. Learning a model is in fact a searching process in which the algorithm selects the best model from a model space according to: (1) the training dataset, which is the representation of the probability distribution, and (2) a target performance measure, including some regularization mechanism to avoid overfitting. If the training set is biased, then the learning process is ill-posed. Furthermore, if the same dataset is also employed in the validation phase (for instance, when a cross-validation is performed), then the estimate of the error obtained is clearly biased. Thus, the first thing to bear in mind is the golden rule of building an unbiased dataset.

A frequent practice in plankton recognition studies is to look for the *best* training dataset. This is partly motivated by the scarcity of labeled examples and imbalanced classes; there are groups that have much fewer examples than others. Researchers usually build training datasets with the same number of individuals for each class to avoid this issue. This is a bad practice.

It is true that imbalanced situations can make some classifiers misclassify the examples of the minority classes. Nonetheless, the solution from a formal point of view is not to select the examples for the training dataset manually, thereby biasing the sampling process. The correct procedure is just the opposite. In supervised learning, the training data comes first. It is the most important element and should be obtained obeying the i.i.d. assumption as far as possible, without introducing sampling bias of any kind. Then, we may work with three elements to boost the performance of our model: (1) enhancing the representation of the input objects (e.g., using advanced computer vision techniques robust to rotation or obstruction), (2) selecting a classifier well tailored to the characteristics of the training data and the learning task (e.g., using algorithms for imbalanced data (Chawla et al. 2004) if required), and (3) tuning the parameters of the learning algorithm (e.g., algorithms usually have a regularization parameter to avoid overfitting, like parameter C in the case of Support Vector Machines).

Selecting the training dataset manually (Culverhouse et al. 1996; Luo et al. 2003; Grosjean et al. 2004; Hu and Davis 2005) is counterproductive for a number of reasons. Balancing the number of training examples for all classes may mean that a large class does not have sufficient diversity, for instance, when such class is complex and it has different types of individuals. Limiting the number of examples for such classes reduces the desired diversity of the training data. The i.i.d. assumption guarantees that, if the sample is large enough, all the individuals will be represented in the training set, making the learning process more reliable.

The previous argument is supported by statistical learning theory. Over the past few years, learning theory papers have established generalization error bounds for different classifiers, including Support Vector Machines (SVM) and ensemble methods like Boosting (Bartlett and Shawe-Taylor 1999; Schapire and Singer 1999; Cristianini and Shawe-Taylor 2000; Vapnik and Chapelle 2000). These bounds decrease (i.e., the probability of error is lower) when the number of examples in the training set increases, among other factors. This is a quite intuitive result; when the model has been trained with more information (examples), its ability to classify unseen examples is greater. Hence, if the total number of examples is reduced in order to balance all classes, the risk of the generalization error of the classifier increases.

Basically, supervised learning requires a large collection of examples, one that is as large as possible, sampled without bias from the underlying population. This in turn guarantees the required diversity of the examples. Note that diversity in this context refers to the different types of objects that the model has to work with. This includes not only the different types of individuals from a biological or morphological point of view, but also the diversity produced by the capturing device or any other element of the processing system that may mean that the same type of individual is represented differently. This general principle has the drawback that obtaining a large collection of examples is usually expensive. Our case is even worse, because, if the unit is the sample and not the individual, a large collection of diverse samples is required. The problem is that obtaining sufficient diversity at the sample level is difficult, but makes diversity at an individual level less problematic.

Data distribution drift

Understanding data distribution drift is important to obtain a better solution to the plankton recognition problem. Formally, this occurs when the joint distribution of inputs and outputs changes. Given two datasets, D and T, captured at different times or places, drift occurs when their joint probability distributions differ; in symbols, $P_D(x,y) \neq P_T(x,y)$. Several factors can be the cause of this drift and the joint probability can be expressed in different ways depending on the type of learning problem. Fawcett and Flach (2005) proposed a taxonomy to classify learning problems according to the causal relationship between class labels and covariates (or inputs). The interest of this taxonomy lies in the fact that it determines the kind of changes in the distribution that a particular task may experience. The authors distinguished between two different kinds of problems: $\mathcal{X} \rightarrow \mathcal{Y}$ problems, in which the class label is causally determined by the values of the inputs; and $\mathcal{Y} \rightarrow \mathcal{X}$ problems, where the class label causally determines the covariates. Spam detection constitutes an example of the first type of problem; the content of the mail and other characteristics determine whether the mail is spam or not. On the other hand, a medical diagnosis task is a typical example of $\mathcal{Y} \rightarrow \mathcal{X}$ problems; suffering from a particular disease, y, causes a series of symptoms, x, to appear, and not the other way around. Plankton recognition is a $\mathcal{Y} \rightarrow \mathcal{X}$ problem. An individual will have some characteristics because it belongs to a particular species or morphological group. These characteristics are a consequence of its class.

In this type of problem, the joint distribution, $P(x, y)$, can be written as $P(x,y)=P(x|y)P(y)$, in which $P(y)$ represents the probability of a class and $P(x|y)$ is the probability of an object x, although knowing that the class is y. We know that $P(x, y)$ changes, so we need to determine whether both terms in the expression change or just one of them.

In abundance related problems, it is evident that $P(y)$ changes, because it is precisely the magnitude that must be estimated for the model. However, does $P(x|y)$ change? The answer to this question is more complex. Lets imagine that we represent each individual using only one characteristic: the particle physical size. If $P(x|y)$ remains constant, it means that the distribution of sizes in each class does not change. Notice that this is a quite strong condition that depends on several factors, basically the representation of the input space, the taxonomy and the classes considered in each particular plankton recognition problem. If we only have a small number of top-level classes, it is almost certain that $P(x|y)$ changes as these classes are formed by different subclasses, whose probabilities will not change in proportion to the main class. Conversely, if the problem distinguishes between classes at the bottom of a taxonomy, then $P(x|y)$ changes are less probable.

Knowledge of all these factors for a given problem, mainly the behavior of $P(x|y)$, is crucial in order to design new algorithms that are robust against the expected changes in the joint probability distribution. For instance, the algorithm proposed by Solow et al. (2001) is based on the assumption that $P(x|y)$ is constant. This is also the main assumption made by several quantification algorithms (Forman 2008).

A way to measure changes in the distribution between two datasets is the Hellinger distance (HD). This measure has been used in classification methods to detect failures in classifier performance due to shifts in data distribution (Cieslak and Chawla 2009). In this paper, HD will be used to study the dataset shift between training and test datasets and how this relates to the accuracy of the classifier and the corresponding validation methods. The Hellinger distance is a type of f-divergence initially proposed to quantify the similarity between two probability distributions. Based on the continuous case formulation, the Hellinger distance can also be computed for the discrete case. Given two datasets, D and T, from the same input space, $\mathcal{X}$, their HD is calculated as

$$\mathrm{HD}(D,T)=\frac{1}{d}\sum_{f=1}^{d}\mathrm{HD}_f(D,T)=\frac{1}{d}\sum_{f=1}^{d}\sqrt{\sum_{k=1}^{b}\left(\sqrt{\frac{|D_{f,k}|}{|D|}}-\sqrt{\frac{|T_{f,k}|}{|T|}}\right)^2}, \quad (3)$$

in which d is the dimension of the input space (the number of attributes or features), $\mathrm{HD}_f(D,T)$ represents the Hellinger distance for feature f, b is the number of bins used to construct the histograms, $|D|$ is the total number of examples in dataset $|D|$, and $|D_{f,k}|$ is the number of examples whose feature f belongs to the k-th bin (the same definitions apply to dataset T).

Performance measures

In order to compare the performance of several methods over a group of samples, two different types of results can be studied. First, the goal may be to analyze the error for a particular class across all samples and obtain the error rate. However, it may also be necessary to calculate the precision for all classes across all samples, a kind of general error of the model.

To compute the error rate for a particular class, we need to compare the predicted count data or frequencies, $\{n'_{c_i,j} : j=1,\ldots,m\}$, with the ground-truth count of class c_i over m labeled samples $\{n_{c_i,j} : j=1,\ldots,m\}$. Three performance measures are usually employed in similar regression problems:

- Bias: $Bias(c_i)=\frac{1}{m}\sum_{j=1}^{m} n_{c_i,j}-n'_{c_i,j}$
- Mean Absolute Error: $MAE(c_i)=\frac{1}{m}\sum_{j=1}^{m} |n_{c_i,j}-n'_{c_i,j}|$
- Mean Square Error: $MSE(c_i)=\frac{1}{m}\sum_{j=1}^{m} (n_{c_i,j}-n'_{c_i,j})^2$

The drawback of *Bias* is that negative and positive biases are neutralized. For instance, a method that guesses the same number of units as too high or too low will have zero bias on average. *MAE* and *MSE* are probably the most widely used loss functions in regression problems, although *MAE* is more intuitive and easier to interpret than *MSE*. Nonetheless, all these metrics present some issues in this case. First, if the frequencies are expressed in terms of another variable, typically volume, the density must be similar in order to average the errors across samples, otherwise the samples with a higher density have a greater influence on the final score. More importantly, these metrics do not allow us to determine the magnitude of the errors. Averaging across samples with different frequencies has certain implications that should be carefully taken into account. For instance, an error of 10 units produced when the actual value is 100 is not the same as when the actual value is 20. In the latter case, the error can be considered worse. The problem of these measures in the context of plankton studies is that it is quite commonplace for a given sample not to contain examples for some classes. Any error in theses cases is high in relative terms, even when the absolute error is low. These factors decrease the usefulness of these performance metrics.

There are several measures for evaluating the importance of errors. Mean Absolute Percentage Error, $MAPE=1/m\sum_{i=1}^{m}(|n_{c_i,j}-n'_{c_i,j}|/n_{c_i,j})$, also called Mean Relative Error *MRE*, is probably the most popular. However, this measure presents some issues: it is asymmetric, unbounded and undefined when $n_{c_i,j}=0$. Moreover, recent papers (Tofallis 2015) have shown that *MAPE* prefers those models that systematically under-forecast when it is used in model selection processes. The log of the accuracy ratio, i.e., $\ln(n'_{c_i,j}/n_{c_i,j})$, has been introduced to select less biased models. However, this measure presents the same problem as *MAPE*: it is undefined when $n_{c_i,j}$ is 0 for one sample, which it is quite common when the number of classes is large. Symmetric MAPE (Armstrong 1978) seems the best alternative considering all the above factors:

$$SMAPE(c_i)=\frac{1}{m}\sum_{i=1}^{m}\frac{|n_{c_i,j}-n'_{c_i,j}|}{n_{c_i,j}+n'_{c_i,j}}. \tag{4}$$

It is a percentage, it is always defined and its reliability for model selection purposes is comparable to that of $\ln(n'_{c_i,j}/n_{c_i,j})$ according to Tofallis (2015).

In order to compute a kind of overall result, an initial performance metric that can be applied is Bray–Curtis dissimilarity (Bray and Curtis 1957). This is commonly used to analyze abundance data collected at different sampling locations in ecological studies. The Bray–Curtis dissimilarity is defined as:

$$\mathrm{BC}=1-2\frac{\sum_{i=1}^{l}\min(n_{c_i},n'_{c_i})}{\sum_{i=1}^{l} n_{c_i}+n'_{c_i}}=\frac{\sum_{i=1}^{l}|n_{c_i}-n'_{c_i}|}{\sum_{i=1}^{l} n_{c_i}+n'_{c_i}}, \tag{5}$$

where l is the number of classes. A good thing here is that both samples obviously have the same total size. This means that the score it is the same whether counts or frequencies are used. The Bray–Curtis dissimilarity is bound between 0 and 1, with 0 meaning that the prediction is perfect. Although the Bray–Curtis dissimilarity metric is able to quantify the difference between samples, it is not a true distance because it does not satisfy the triangle inequality axiom.

A possible alternative to the Bray–Curtis dissimilarity is the Kullback–Leibler Divergence, also known as normalized cross-entropy. In this case, it is usual to compare a set of frequencies:

$$\mathrm{KLD}(n,n')=\sum_{i=1}^{l}\frac{n_{c_i}}{\sum_{i=1}^{l} n_{c_i}}\cdot\log\left(\frac{n_{c_i}}{n'_{c_i,j}}\right). \tag{6}$$

The main advantage of KLD is that it may be more suitable for averaging over different test prevalences. However, a drawback of KLD is that it is less interpretable than other measures, such as the Bray–Curtis dissimilarity or *MAE*. Moreover, it is not defined when a frequency is 0 or 1, which is quite common in plankton recognition, particularly when the number of classes is large. In order to resolve these situations, KLD can be normalized via the logistic function: $\mathrm{NKLD}(n,n')=2/(1+\exp(\mathrm{KLD}(n,n')))$.

Assessment methods for a collection of samples

As stated previously, several studies have found it difficult to expose their algorithms to changes in the distribution of plankton populations, reaching the conclusion that traditional assessment methods significantly overestimate models accuracy. Our goal is to propose an assessment methodology that ensures that training and testing datasets change, introducing the data distribution variations that will occur under real conditions. Moreover, the test should not be carried out only with one test set. Ideally, testing should be carried out with different samples presenting different distributions, covering the actual variations due to seasonal factors or the location of sampling stations as much as possible. Here, we shall discuss how to extend traditional assessment methods, namely hold-out and cross-validation, to the case of working with a group of samples, highlighting both their drawbacks and strengths.

The extension of hold-out is fairly straightforward. As always, we need a training dataset, obtained without sampling bias, that represents the probability distribution of the study. Additionally, a collection of samples must be collected to constitute the testing set. The performance of the model is assessed just in this collection, computing a sample-based measure, like the ones discussed previously.

The drawback of hold-out is that the effort involved in collecting data is doubled because we need two separate datasets. This is much most costly when working with samples. The labeled data is usually limited, so in some studies one of the datasets will be smaller than it should be. If the training dataset is reduced in size, useful information to build the model is lost. If we limit the size of the testing dataset, the assessment of the model will be poor. It seems that shifting the unit of the study from the individual to the sample makes hold-out less suitable.

The other alternative is to apply cross-validation (CV). The difference with respect to traditional CV is that the folds are composed of a number of complete samples. The key parameter in CV is the number of folds. Selecting a low number of folds once again means that the training dataset for each run is smaller, with the same drawbacks as mentioned previously. Thus, the best way to have the maximum amount of training data is to conduct a leave-one-out (LOO) cross-validation of samples (see Fig. 1). Given a set of m samples, m training and test iterations are performed ($E_1 \dots E_m$). In each iteration, all but one sample (the gray sample in each iteration in Fig. 1), is selected as the testing set, performing training with the remaining samples (the white samples in each iteration in Fig. 1).

Notice that this kind of LOO is computationally less expensive than in the case of LOO at the individual level, because the number of individuals is much larger than the number of samples ($n \gg m$); in the case under study, 39,613 examples vs. 60 samples. The other main advantage is that the method operates under similar conditions to real ones when the model is deployed: it has been trained using a group of samples, then it has to make a prediction for a new, unseen sample. It also guarantees a realistic degree of variation between training and test sets. We shall analyze this factor in the experiments.

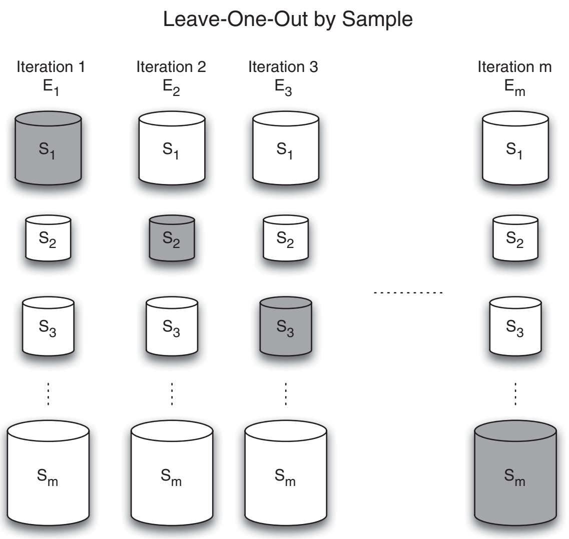


Fig. 1. Given m samples, m training and test iterations are performed ($E_1 \dots E_m$). In each iteration, the gray sample is selected as the testing set, while white samples are used for training. The samples may have different sizes, as represented in the figure.

The advantages of LOO over hold-out are twofold: (1) LOO uses as many training examples as possible, and (2) the estimate of the error is theoretically more precise. However, it also presents an important drawback: it cannot be applied when the samples present some kind of correlation among themselves; for instance, when they come from a series of samples obtained in a short period of time. In such cases, hold-out is the best option: the model is trained with a separate training set and tested on such collection of testing samples. Any sort of cross-validation using this collection of samples will over-estimate the performance of the model. In order to apply cross-validation to a collection of correlated samples, the division in folds must guarantee that those samples correlated among themselves should belong to the same fold. This may, however, be impossible in some cases; for instance, when the size of the fold is just one sample, which is the case of LOO.

Finally, if the collection of samples is large, which is the ideal situation, then instead of using LOO, which can be computationally expensive in such situations, a CV by sample can be applied. The number of folds selected should be as large as possible depending on the computational resources available in order to obtain a more precise estimate. Another important aspect is that, in order to compute the performance measures, the actual samples of the dataset must be considered, without aggregating those that belong to the same fold. Testing samples must not be aggregated because this procedure will create new artificial samples. The error should be measured for each sample separately and then averaged. For instance, if we have a dataset with 1000 samples, carrying out a LOO by sample will be computationally expensive, as 1000 training and testing operations will be required, each one with a large number of individual examples. This can be reduced to our liking, selecting a number of folds and performing a CV by sample. For instance, with 10-folds, 900 samples will be used for each training process, using the other 100 samples for testing. This process will be carried out 10 times (vs. 1000 iterations of LOO), thus saving in training time. Note that we cannot evaluate error using a test set of 100 samples together, as it would be an artificial sample, suffering the same problems as standard cross validation. Instead, the error should be measured for each test sample separately and then averaged.

Case under study

A relatively large dataset of samples was collected to study the behavior of plankton recognition systems and model assessment methods. Specifically, the images obtained correspond to 60 different samples obtained at different places and different times. This dataset was captured using a FlowCAM (Sieracki et al. 1998) in the Bay of Biscay and off the northern coast of Spain and Portugal between August 2008 and April 2010 (Álvarez et al. 2012). Images were captured using 100X magnification with the aim of analyzing organisms with an equivalent spherical diameter (ESD) between 20 μm and 100 μm. Each of the captured images was segmented using the intensity-based method proposed by Tang et al. (1998). Once segmented, the images were classified by an expert taxonomist into eight categories (Artefacts, Diatoms, Detritus, Sillicoflagellates, Ciliates, Dinoflagellates, Crustaceans and the category Others, for other living objects which could not be classified among the previous categories). A crucial aspect is that all the organisms within each sample were analyzed and labeled without exception to avoid sampling bias.

The sample stations are located at different geographical points and at different depths, as shown in Fig. 2. This results in high variation in the concentration of species, can be seen in Fig. 3, since large regions have been covered in both temporal and spatial terms. For example, the concentration of diatoms in Sample 57 is large (over 75%) compared to Sample 54, in which there are almost no diatoms (less than 1%). More examples like this one can be found in the dataset.

The features vector for each image, x, for each image was calculated using the EBImage R package (Pau et al. 2010). Standard descriptors, including shape and texture features (Haralick et al. 1973), were computed with this package. Furthermore, features computed by the FlowCAM software, such as particle diameter and elongation, were also included

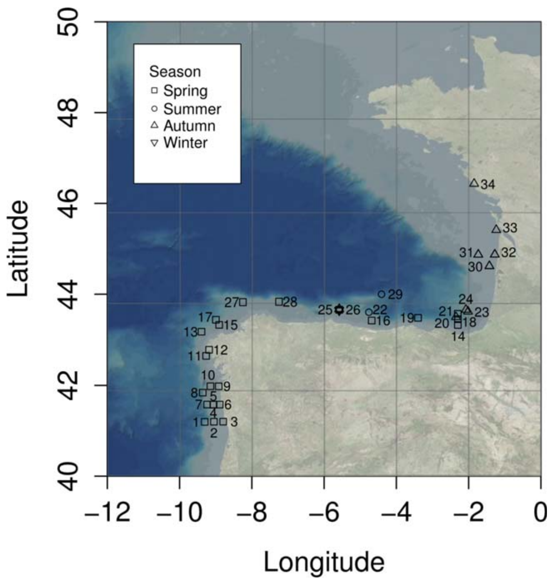


Fig. 2. Geographical location and sampling season. The numbers shown in the figure correspond to the station (see Fig. 3) to allow the identification of the place where each sample was collected).

in the feature vector. In all, a vector with 64 characteristics was computed for each image.

To summarize, a total of $m = 60$ samples were captured and processed, resulting in a total of $n = 39,\ 613$ images manually labeled in eight different classes.

All experiments were performed using the caret R package (Kuhn 2008). Results were extracted using two different learning algorithms, Support Vector Machines (SVM) (Vapnik and Vapnik 1998) and Random Forest (Breiman 2001), to confirm that the results do not depend on a particular classifier. These classifiers are the most popular in plankton recognition papers. A Gaussian kernel was used to train the SVM models, using a grid search in order to find the best parameters for just the training dataset of each run (regularization parameter C values from 1 to 1×10^3, and sigma values from 1×10^{-6} to 1×10^{-1}). In the case of Random Forest, each model is composed of 500 trees. A grid search was also used to estimate the number of random features selected (values 4,8,16). Tuning parameters is essential in order to avoid overfitting and to obtain better results. This process must be carried out using only the training data in each run of the learning algorithm.

Results

The goal of the experiments was not to build the best classifier or analyze various learning approaches. The experiment was simply designed to compare model assessment

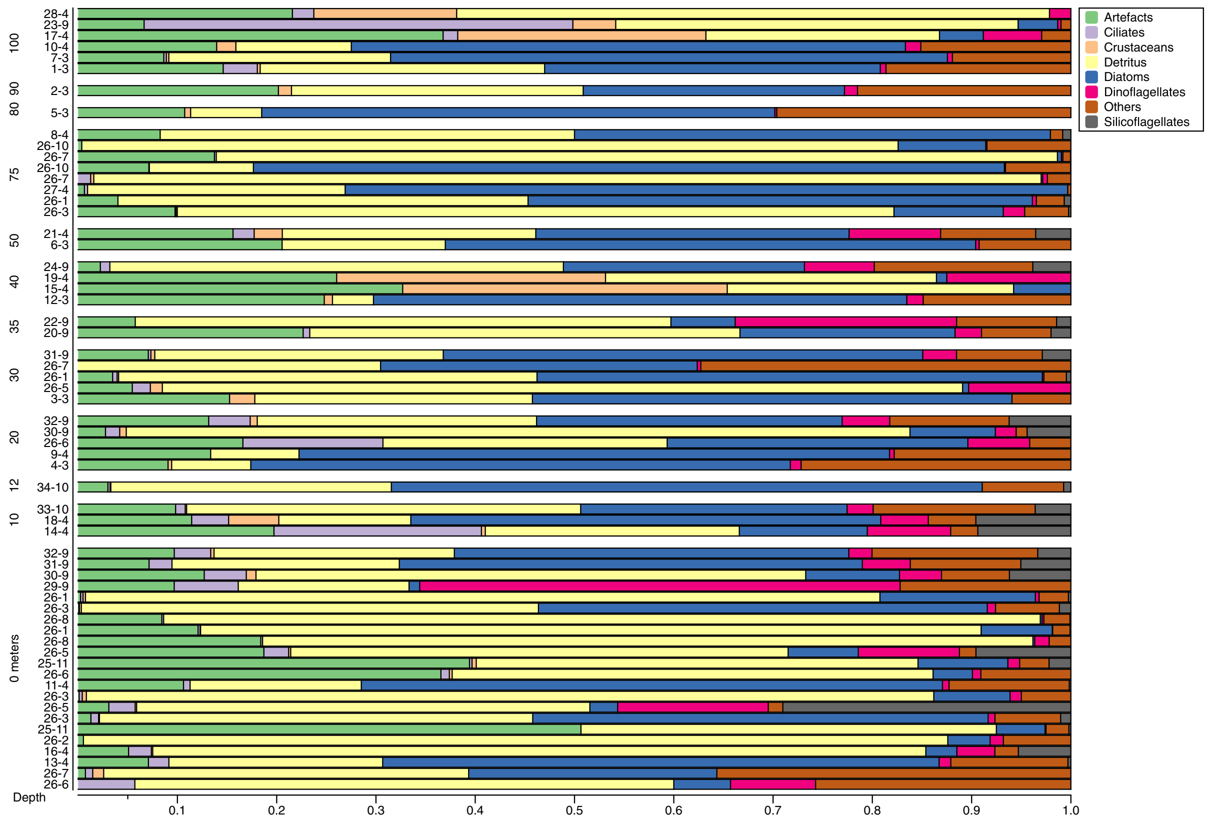


Fig. 3. Distribution of samples by classes. Samples are grouped by depth (in meters) and labeled using the station number and the month in which they were taken.

methods, focusing on the differences between those based on the performance at an individual level and those based on samples. We thus compared standard cross-validation (CV), which works at an individual level, with the proposed leave-one-out (LOO) by sample. Note that in the former case the whole dataset is merged and the samples are not taken into account to obtain the folds. Thus, individual examples from the same sample may belong to different folds. Specifically, we compared three methods: 10-fold CV, 60-fold CV (both working at an individual level) and LOO by sample. We selected 10-fold CV because it is a quite common experimental procedure in many studies (see Table 1) and 60-fold CV to match up the number of samples in the dataset, providing a fair comparison to the experiment using the LOO by sample method.

Table 2 presents the results for both algorithms (SVM and RF) using the three different validation techniques discussed previously. The results for SVM are slightly better, although both algorithms show the same trend. SVM achieves a reasonable degree of accuracy of 82.88% using a standard 10-fold CV. There is no significant difference when the number of folds is increased to 60. Similar results are also obtained with RF using 10-fold CV and 60-fold CV. We can thus conclude that the number of folds has no influence over the obtained estimate. This is mainly due to the fact that the number of examples in the dataset is quite large. Therefore, the probability distributions represented by the training datasets used in each trial are similar because they are large (35,652 examples in a 10-fold CV vs. 38,953 in a 60-fold CV)

Table 2. Accuracy (in percentage) and standard error using different validation methods.

	10 CV	60 CV	LOO by sample	
	Acc	*Acc*	*Acc*	$Acc_{\mathbf{sample}}$
SVM	82.88 ± 0.191	83.10 ± 0.179	77.74	71.78 ± 1.679
RF	82.06 ± 0.180	82.16 ± 0.183	77.05	70.36 ± 1.912

and cross-validation tends to produce similar folds, so the learned models should be approximately equal in each run. In fact, exploring results fold by fold, it was found that the variation in accuracy between folds was small (1.5%/1.4% for 10-fold CV and 6.1%/6.2% for 60-fold CV using SVM/RF, respectively).

However, the accuracy estimate is lower when LOO by sample is used. Notice that we can compute accuracy in two different ways here: (1) summing up the number of correct predictions on each sample and dividing by the total number of examples (this corresponds to the probability of correctly classifying a single unseen instance), and (2) averaging the accuracy per sample (the estimate is the average accuracy of a given unseen sample). Although the scores are different, they are computed from the same individual predictions, the difference arising from the way of averaging the predictions.

In the case of standard CV, both values, Acc_{byfold} and Acc are approximately equal. This is because all the folds have approximately the same size. Being n the number of examples in training set D, NF the number of folds and n_{F_j} the number of examples in fold F_j, we have that:

$$Acc_{byfold}=\frac{1}{NF}\sum_{j=1}^{NF}\frac{1}{n_{F_j}}\sum_{x_i\in F_j}I(h_j(x_i)=y_i)\approxeq\frac{1}{n}\sum_{j=1}^{NF}\sum_{x_i\in F_j}I(h_j(x_i)=y_i)=Acc, \tag{7}$$

because $\frac{n}{NF}\approxeq n_{F_j}$ for all j. In contrast, the samples have different sizes in a LOO by sample experiment and hence the two values differ.

Comparing the accuracy estimate at an individual level obtained by means of CV and LOO by sample, the question that has to be answered here is why they are different. First, in our opinion, standard CV is optimistic because, as stated previously, individual examples from the same sample are placed in different folds. Thus, the learner uses examples for training from the same samples as those in the testing set. The estimate is optimistic because these examples are correlated and tend to be similar. This will not occur when the model is deployed and it classifies a new unseen sample, which is in fact the conditions that LOO by sample simulates.

On the other hand, the estimate provide by LOO could be seen as pessimistic because one particular sample used as the test sample may be very different from the rest. This obviously will not occur the same number of times if the training set is composed of a larger collection of samples. Actually, when the number of samples tends to infinity, both methods will return the same estimate (which will be the true accuracy). However, bear in mind that we are estimating the accuracy for the model computed with a limited dataset, not with an infinite number of samples. Hence, in our case, if the test sample is not very well classified in one iteration of LOO using the model learned with the other 59 samples (nearly 40,000 examples), we may infer that the same will occur with other unseen samples when we train our model with the complete 60-sample training set. This is, in fact, the goal of the validation process, making estimates of the future performance. Moreover, these cases show us that the training dataset is possibly not large enough, and that more samples are required.

For all the above reasons, we do think that the estimate computed by means of LOO by sample is more realistic than the one computed by means of standard CV, even though the latter may be somewhat pessimistic, which is better than being optimistic, and it is probably more accurate for a finite collection of samples, which is our goal. Another interesting aspect is that the difference between both measures can serve as an estimator of the completeness of the training set.

On the other hand, the average accuracy at a sample level is especially useful once we have trained our model and we wish to apply it to classify new, unseen single samples. Recall that it measures the expected accuracy when the model only classifies one finite sample and hence it is different to the one previously discussed. First, it is logical for the accuracy in this case to be lower due to size of the samples; the accuracy tends to be lower for smaller samples because any mistake represents a higher percentage. Furthermore, for the same reason, it is always more variable than in the case of large samples. For very large samples, the accuracy at a sample level will tend to be the same as that estimated at an individual level. However, several ecological studies work with relatively small samples.

The second aspect to consider is that the variability in terms of samples can be huge with respect to several features, like size, difficulty and class distribution, among others. We can find small samples and large ones, samples that contain individuals that are particularly difficult to classify and other samples that are composed of easy examples, samples with a different class distribution, etc. Thus, the accuracy can dramatically differ in all of these situations. For instance, in the LOO experiment in Table 2, the accuracy for the worst sample is as low as 30.1% (93 examples); in the contrary case, this value rises to 96.4% for the best sample (2731 individuals). The standard deviation of the estimate is 13.01, showing the great variability in accuracy when different samples are considered. For the sake of comparison, the standard deviation in 10-fold CV is 0.60 and 1.39 in 60-fold CV. The standard deviation of LOO seems excessively high in this experiment, suggesting that we should probably add new samples to the training data to increase the stability of the model. Nonetheless, it is far more realistic than the one provided by standard CV.

In conclusion, we need as large a collection as possible of actual samples in order to estimate the accuracy, or any other magnitude, at a sample level. This is the reason for proposing LOO by sample. Note that these measures cannot be computed using traditional CV because the folds that CV generates: (1) are artificial samples that do not represent

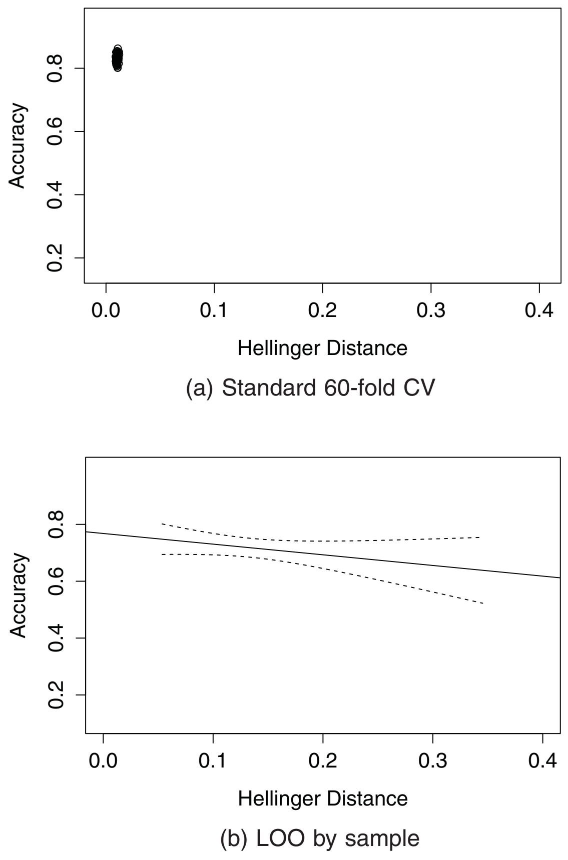


Fig. 4. Relationship between HD and accuracy for each fold/sample using SVM as the classifier (R^2=0.0341, *p*-value = 0.1577 and *S* = 0.1289 for the LOO experiment). Dotted lines: 95% confidence interval. A similar graph is obtained for the Random Forest classifier. (**a**) Standard 60-fold CV, (**b**) LOO by sample.

actual ones, and (2) they do not have the required variability. In fact, the folds of a CV are a sort of average of the underlying population. Figure 4 shows this feature, comparing the training and testing distributions of both experiments. The figure shows the Hellinger Distance (HD) of both sets, computed used 30 bins, and the corresponding accuracy estimate of the sample/fold. Both distributions are approximately equal in the case of CV (Fig. 4a). It is worth noting that the accuracy and HDs remain practically constant throughout each of the folds. In contrast, the distributions of the training and testing sets differ when LOO by sample is applied; the HDs are significantly different and the accuracy between samples also varies notably (see Fig. 4b). Notice that the minimum HD in the LOO experiment is greater than all the HDs for the CV experiment.

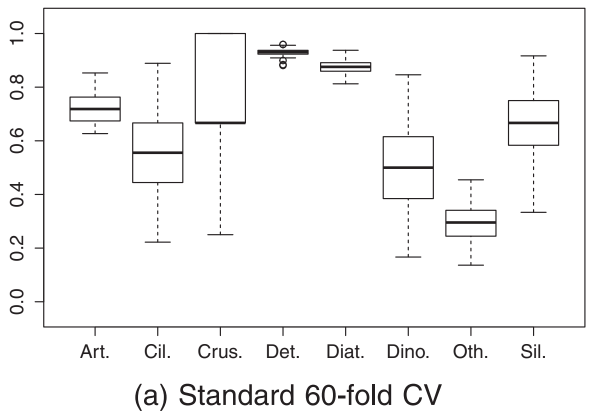
The box plot in Fig. 5 shows the accuracies of LOO by sample and 60-fold CV using SVM for the different classes across the iterations of the experiment. Samples or folds with less than 10 examples for a given class were omitted (three for Crustaceans). This filter was applied to exclude situations that do not produce representative results. For instance, if there is only a single example of a class in a given test set and the classifier fails to recognize it, it will yield an accuracy of 0% for that particular iteration and class, when in fact the classifier has only failed to classify one example, which is insignificant.

Analyzing the box plot in detail, major differences can be seen in classification stability for some classes, especially Detritus and Diatoms. When using standard CV (Fig. 5b), the success rate by class once again remains much more constant throughout all folds because their variability is small. The classes with more variability are those with a limited number of examples per fold (Ciliates, 10.3 examples; Crustaceans, 3.2; Dinoflagellates, 12.6; and Silicoflagellates, 12.3). Classes with a large number of examples (Detritus and Diatoms) do not show any variability. It is thus impossible to study the robustness and stability of the model for each class. In contrast, when LOO by sample is used (Fig. 5a), the classifier accuracy for each class in different iterations tends to be more variable, allowing researchers to analyze these cases in order to improve their models. Once more, this is because these measures are estimates obtained at a sample level, in which LOO by sample is a more appropriate validation method.

The second part of these experiments is devoted to analyzing the behavior of the proposed performance metrics to estimate the abundance. Our purpose is to show a comparison between two algorithms, in this case SVM and Random Forest. Two performance measures are considered: *SMAPE* and Bray–Curtis dissimilarity. The former is applied to study the precision for each class individually and the latter to obtain a global measure. In all cases, we use the predictions obtained in the LOO by sample experiments. Table 3 contains the results for both SVM and RF.

Analyzing the SMAPE results, the errors are excessively high, except for Detritus and Diatoms. Comparing SVM and RF, the scores obtained by SVM are better than those of RF for most classes, except for Others and Silicoflagellates. This is also confirmed by the Bray–Curtis dissimilarity value, which is lower in the case of SVM. These results seem to suggest that the better the accuracy of a model, the better the estimates at an aggregated level.

Figure 6 shows the relationship between accuracy and Bray–Curtis dissimilarity when SVM and RF are used in a LOO by sample experiment. Each point represents both scores for a sample. In both cases, the correlation between the two measures are lower than expected, confirming that better accuracy at an individual level does not mean better performance when an aggregated magnitude is predicted. For instance, in sample 31 the accuracy is just 0.62 but BC score is relatively low, 0.09, while sample 39 has a much



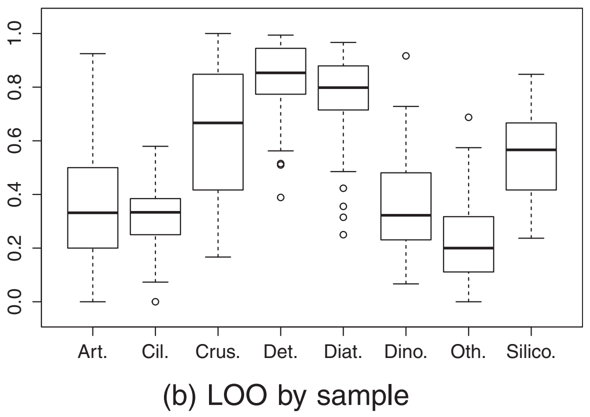


Fig. 5. Accuracy by class and iteration/fold using SVM as the classifier. Only classes with 10 (three for the Crustaceans class) or more examples in the iteration/fold are represented. The number of these cases for LOO are: Artefacts (46), Ciliates (13), Crustaceans (19), Detritus (59), Diatoms (49), Dinoflagellates (24), Others (49), and Silicoflagellates (18). For CV, there are always 60 values. (**a**) Standard 60-fold CV, (**b**) LOO by sample.

Table 3. SMAPE scores and Bray–Curtis dissimilarity in the LOO by sample experiment.

SMAPE scores	SVM	Random Forest
Artefacts	0.41 ± 0.03	0.53 ± 0.04
Ciliates	0.47 ± 0.05	0.63 ± 0.05
Crustaceans	0.32 ± 0.05	0.33 ± 0.05
Detritus	0.10 ± 0.01	0.12 ± 0.01
Diatoms	0.19 ± 0.03	0.25 ± 0.03
Dinoflagellates	0.41 ± 0.04	0.57 ± 0.05
Others	0.45 ± 0.04	0.42 ± 0.04
Silicoflagellates	0.38 ± 0.05	0.38 ± 0.05
	SVM	Random Forest
Bray–Curtis	0.15 ± 0.01	0.19 ± 0.01

better accuracy, 0.83, but a higher BC dissimilarity, 0.13. In fact, the two problems are different from a learning point of view, as discussed previously, and the optimal model for one of them is not optimal for the other, except in the trivial case of obtaining a perfect classifier, which is unrealistic.

Discussion

We can distinguish between two major groups of studies in which image classification tools are useful. Abundance is a primary ecological currency and many studies require automatic methods able to predict the abundance of the different planktonic groups for samples collected in plankton surveys. However, the aim of other studies is to understand properties of the plankton community in addition to abundance (for example, calculating the size structure composition of each classification category) and hence require precise classification of each individual image. Although both cases seem the same learning problem (both classify plankton images), these two types of applications likely require different learning algorithms and surely call for different model assessment and validation methods. The most important difference between both types of studies is that, in the former the aim is to minimize the error per sample, while in the latter, the learning algorithm should also seek to minimize the error for each individual image. This paper focuses mainly on the analysis of the validation techniques required for those studies that require predictions for complete samples.

It is important to stress the great variability found in the performance rates depending on the classified sample. A great disparity in results is also observed in intra-class accuracy. In this respect, the proposed methodology can show us aspects of the capabilities of the models that would remain hidden using other validation strategies. One of these aspects is the significant variability in the results found for certain classes (e.g., diatoms and ciliates). In difficult problems like the one addressed in this paper, it is important to try to look beyond the overall accuracy rate. Very useful information can thus be found which may be valuable in order to build better automatic recognition systems.

One of the most interesting features of performing sample-oriented experiments, like LOO by sample, is that it helps researchers to draw conclusions about the adequacy of the whole learning process. High variations in performance between samples, or for certain classes, may reflect a need to increase the size of the dataset, adding new labeled samples. Eventually, the system may face a new sample, that contains examples that seldom appear in the rest of the training dataset, causing a high error rate for that sample and class and hence high variability in the results. Adding more samples will make the results and the system more robust and more stable.

An illustrative example can be seen in Fig. 5a. There is at least one sample for which the hit rate is 0 for the class

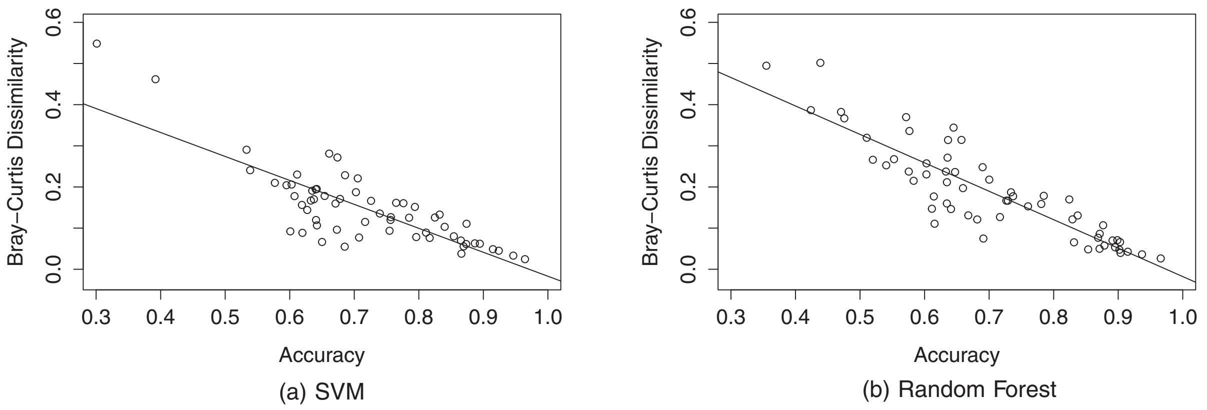


Fig. 6. Relationship between Accuracy and Bray–Curtis Dissimilarity in LOO experiments (R^2=0.6510, p-value = 0, S = 0.0560 for SVM and R^2=0.7798, p-value = 0, S = 0.0548 for RF). (**a**) SVM, (**b**) Random Forest.

Ciliates. The sample in question is Sample 13, which has 140 examples labeled as Ciliates. All of these examples are misclassified by the classifier when LOO is applied. Investigating more deeply, it turned out that this group of examples was actually a subspecies of ciliates, oligotrichs. There are only 142 examples of this subspecies in our dataset, 140 of which are in Sample 13. Obviously, when excluding Sample 13 from the training set, the classifier does not have enough information to learn how to classify this subtype. Such situations cannot be detected using standard CV during the experiments, but will occur once the model is deployed. This is another important reason why the validation strategy should cover such cases, in order to detect them and, if necessary, improve both the dataset and the classification algorithm. It may be considered that sufficient samples are taken when LOO results are good enough not only in terms of overall accuracy, but also with respect to other aspects, like the variability in inter-sample and intra-class performance. The ultimate goal is to obtain a more robust final model and its corresponding accurate performance estimate.

An interesting open question is whether we can somehow anticipate the reliability of the prediction for a new sample. In this respect, Fig. 4 seems to suggest that when a sample is far from the training set in terms of Hellinger distance, any prediction made is less reliable. This is partially true, although there are other factors that also exert an influence. The most important is the difficulty in classifying the instances of the sample: classifiers make most mistakes in those examples near the frontiers between classes. Actually, to detect whether the sample is strange, we could compute the minimum distance between the sample and all of those in the training set. A large distance implies that the new sample is so different to the samples in the training set, thus making the prediction less reliable.

The main drawback of the methodology proposed here is that it requires a large collection of samples to be absolutely precise. In some studies, this is impossible due to the cost of labeling individual examples. A possible alternative in these situations is to generate artificial, yet biologically plausible samples. This technique is used in quantification learning and is based on the fact that we are dealing with a $\mathcal{Y} \rightarrow \mathcal{X}$ problem and the class causally determines the values of the inputs. We know that $P(y)$ changes in abundance-related problems, and in some studies we can make the further assumption that $P(x|y)$ remains constant. Given an actual sample, we can generate a new artificial sample following these two steps: (1) varying the proportions of the classes of the original sample, generating random values for $P(y)$ possibly using predefined thresholds, and (2) performing a random sampling with replacement (to ensure that $P(x|y)$ does not change) in the original sample, until the number of examples required for each class is obtained. This process has to be carried out using knowledge about the actual study in order to generate plausible samples with the expected distribution of classes, $P(y)$. This allows us to study whether the model is able to correctly predict the abundance for a wide range of expected distributions. In some problems, the assumption that $P(x|y)$ remains constant is too strong; for instance, in the case under study, in which only top-level classes are considered. However, in problems with a large taxonomy, the assumption is probably true for the classes at the bottom of the taxonomy and the procedure could be applicable.

Conclusions

This paper analyzes different validation techniques used in plankton recognition problems, comparing the common methods used in the field. Although studies apply different approaches, most present similar issues. Results can be very different when different validation strategies are employed, leading to results which are not directly comparable. Even more importantly, when they are used in "production," the models learned are likely to provide less satisfactory results than those estimated in the experimental phase applying traditional model assessment methods. The reason is that these techniques, such as standard cross-validation, are devised for other kinds of learning tasks.

After discussing the shortcomings of these validation strategies for those problems in which the goal is to predict a magnitude given new samples, we propose to change the basic unit of these studies, using the sample as the basic unit. In keeping with this idea, the present paper proposes an extension of the well-known leave-one-out method as a good alternative to obtain accurate estimates at a sample level. The method is able to estimate classifier performance more realistically, taking into account the variety of samples the classifier will face. Using this model assessment method and applying the Hellinger distance, it has been found that the difference between the training and test sets exerts a certain influence over model performance.

Another important conclusion is that it is necessary to focus efforts on designing new learning algorithms which are more robust to the differences between training and test sets. This does not mean to increasing the overall classifier accuracy (which already may be high enough), but making the methods more robust to the changes that occur under real world conditions. These algorithms could be applied in domains like the one studied here, in which, due to a variety factors, the data used to train the model does not accurately represent the final data it will predict. From the point of view of machine learning researchers, this validation strategy allows them to test whether their ideas and the algorithms they have developed to address plankton classification problems work well when there are changes in data distribution.

In the era of Big Data, in which large collections of data are obtained for different applications, plankton recognition also needs to build large datasets for different types of analytic studies. In this respect, using software and hardware tools that allow taxonomists to classify instances quickly can help to obtain these datasets, thereby reducing costs. Ultimately, machine learning requires data, particularly for difficult learning problems like plankton recognition. Lack of data leads to poor models, to poor model assessments or, often, to both.

References

Álvarez, E., Á. López-Urrutia, and E. Nogueira. 2012. Improvement of plankton biovolume estimates derived from image-based automatic sampling devices: Application to FlowCAM. J. Plankton Res. **34**: 454–469. doi:10.1093/plankt/fbs017

Armstrong, J. S. 1978. Long-range forecasting: From crystal ball to computer. Wiley.

Barranquero, J., P. González, J. Díez, and J. J. del Coz. 2013. On the study of nearest neighbour algorithms for prevalence estimation in binary problems. Pattern Recognit. **46**: 472–482. doi:10.1016/j.patcog.2012.07.022

Barranquero, J., J. Díez, and J. J. del Coz. 2015. Quantification-oriented learning based on reliable classifiers. Pattern Recognit. **48**: 591–604. doi:10.1016/j.patcog.2014.07.032

Bartlett, P., and J. Shawe-Taylor. 1999. Generalization performance of support vector machines and other pattern classifiers, p. 43–54. In B. Schölkopf, C. J. C. Burges and A. J. Smola [eds.], Advances in kernel methods—support vector learning. The MIT Press, Cambridge, Massachusetts.

Beaufort, L., and D. Dollfus. 2004. Automatic recognition of coccoliths by dynamical neural networks. Mar. Micropaleontol. **51**: 57–73. doi:10.1016/j.marmicro.2003.09.003

Bell, J. L., and R. R. Hopcroft. 2008. Assessment of ZooImage as a tool for the classification of zooplankton. J. Plankton Res. **30**: 1351–1367. doi:10.1093/plankt/fbn092

Benfield, M., and others. 2007. RAPID: Research on automated plankton identification. Oceanography **20**: 172–187. doi:10.5670/oceanog.2007.63

Blaschko, M., G. Holness, M. Mattar, D. Lisin, P. Utgoff, A. Hanson, H. Schultz, and E. Riseman. 2005. Automatic in situ identification of plankton, v. 1, p. 79–86. *In* IEEE Workshop on Application of Computer Vision, Breckenridge, Colorado.

Boddy, L., C. W. Morris, M. F. Wilkins, G. A. Tarran, and P. H. Burkill. 1994. Neural network analysis of flow cytometric data for 40 marine phytoplankton species. Cytometry **15**: 283–293. doi:10.1002/cyto.990150403

Boddy, L., C. W. Morris, M. F. Wilkins, L. AlHaddad, G. A. Tarran, R. R. Jonker, and P. H. Burkill. 2000. Identification of 72 phytoplankton species by radial basis function neural network analysis of flow cytometric data. Mar. Ecol. Prog. Ser. **195**: 47–59. doi:10.3354/meps195047

Bray, J. R., and J. T. Curtis. 1957. An ordination of the upland forest communities of southern Wisconsin. Ecol. Monogr. **27**: 325–349. doi:10.2307/1942268

Breiman, L. 2001. Random forests. Mach. Learn. **45**: 5–32. doi:10.1023/A:1010933404324

Chawla, N. V., N. Japkowicz, and A. Kotcz. 2004. Editorial: Special issue on learning from imbalanced data sets. SIGKDD Explor. Newsl. **6**: 1–6. doi:10.1145/1007730.1007733

Cieslak, D. A., and N. V. Chawla. 2009. A framework for monitoring classifiers' performance: When and why failure occurs? Knowl. Inf. Syst. **18**: 83–108. doi:10.1007/s10115-008-0139-1

Cristianini, N., and J. Shawe-Taylor. 2000. An introduction to support vector machines and other kernel-based learning methods. Cambridge Univ. Press.

Culverhouse, P. F., and others. 1996. Automatic classification of field-collected dinoflagellates by artificial neural network. Mar. Ecol. Prog. Ser. **139**: 281–287. doi:10.3354/meps139281

Dai, J., R. Wang, H. Zheng, G. Ji, and X. Qiao. 2016. ZooplanktoNet: Deep convolutional network for zooplankton classification, p. 1–6. *In* OCEANS 2016-Shanghai. IEEE, Shanghai, China.

Davis, C. S., Q. Hu, S. M. Gallager, X. Tang, and C. J. Ashjian. 2004. Real-time observation of taxa-specific plankton distributions: An optical sampling method. Mar. Ecol. Prog. Ser. **284**: 77–96. doi:10.3354/meps284077

Duda, R. O., P. E. Hart, and D. G. Stork. 2012. Pattern classification. John Wiley & Sons.

Ellen, J., H. Li, and M. D. Ohman. 2015. Quantifying California current plankton samples with efficient machine learning techniques, p. 1–9. *In* OCEANS 2015-MTS/IEEE Washington. IEEE, Washington D.C., USA.

Embleton, K. V., C. E. Gibson, and S. I. Heaney. 2003. Automated counting of phytoplankton by pattern recognition: A comparison with a manual counting method. J. Plankton Res. **25**: 669–681. doi:10.1093/plankt/25.6.669

Faillettaz, R., M. Picheral, J. Y. Luo, C. Guigand, R. K. Cowen, and J. O. Irisson. 2016. Imperfect automatic image classification successfully describes plankton distribution patterns. Methods Oceanogr. **15–16**: 60–77. doi:10.1016/j.mio.2016.04.003

Fawcett, T., and P. Flach. 2005. A response to Webb and Ting's on the application of ROC analysis to predict classification performance under varying class distributions. Mach. Learn. **58**: 33–38. doi:10.1007/s10994-005-5256-4

Forman, G. 2008. Quantifying counts and costs via classification. Data Min. Knowl. Discov. **17**: 164–206. doi:10.1007/s10618-008-0097-y

Frankel, D. S., S. L. Frankel, B. J. Binder, and R. F. Vogt. 1996. Application of neural networks to flow cytometry data analysis and real-time cell classification. Cytometry **23**: 290–302. doi:10.1002/(SICI)1097-0320(19960401)23:4<290::AID-CYTO5>3.0.CO;2-L

Gislason, A., and T. Silva. 2009. Comparison between automated analysis of zooplankton using ZooImage and traditional methodology. J. Plankton Res. **31**: 1505–1516. doi:10.1093/plankt/fbp094

González, P., E. Alvarez, J. Barranquero, J. Díez, R. Gonzalez-Quiros, E. Nogueira, A. Lopez-Urrutia, and J. J. del Coz. 2013. Multiclass support vector machines with example-dependent costs applied to plankton biomass estimation. IEEE Trans. Neural Netw. Learn. Syst. **24**: 1901–1905. doi:10.1109/TNNLS.2013.2271535

Gorsky, G., P. Guilbert, and E. Valenta. 1989. The Autonomous Image Analyzer–enumeration, measurement and identification of marine phytoplankton. Mar. Ecol. Prog. Ser. **58**: 133–142. doi:10.3354/meps058133

Gorsky, G., and others. 2010. Digital zooplankton image analysis using the ZooScan integrated system. J. Plankton Res. **32**: 285–303. doi:10.1093/plankt/fbp124

Grosjean, P., M. Picheral, C. Warembourg, and G. Gorsky. 2004. Enumeration, measurement, and identification of net zooplankton samples using the ZOOSCAN digital imaging system. ICES J. Mar. Sci. **61**: 518–525. doi:10.1016/j.icesjms.2004.03.012

Haralick, R. M., K. Shanmugam, and I. H. Dinstein. 1973. Textural features for image classification. IEEE Trans. Syst. Man Cybern. **6**: 610–621. doi:10.1109/TSMC.1973.4309314

Haury, L., J. McGowan, and P. Wiebe. 1978. Patterns and processes in the time-space scales of plankton distributions, p. 277–327. In J. H. Steele [ed.], Spatial pattern in plankton communities. Woods Hole Oceanographic Institution, Woods Hole, Massachusetts.

Hu, Q., and C. Davis. 2005. Automatic plankton image recognition with co-occurrence matrices and support vector machine. Mar. Ecol. Prog. Ser. **295**: 21–31. doi:10.3354/meps295021

Hu, Q., and C. Davis. 2006. Accurate automatic quantification of taxa-specific plankton abundance using dual classification with correction. Mar. Ecol. Prog. Ser. **306**: 51–61. doi:10.3354/meps306051

Jeffries, H. P., M. S. Berman, A. D. Poularikas, C. Katsinis, I. Melas, K. Sherman, and L. Bivins. 1984. Automated sizing, counting and identification of zooplankton by pattern recognition. Mar. Biol. **78**: 329–334. doi:10.1007/BF00393019

Kuhn, M. 2008. Building predictive models in R using the caret package. J. Stat. Softw. **28**: 1–26. doi:10.18637/jss.v028.i05

Lindgren, J. F., I. M. Hassellöv, and I. Dahllöf. 2013. Analyzing changes in sediment meiofauna communities using the image analysis software ZooImage. J. Exp. Mar. Biol. Ecol. **440**: 74–80. doi:10.1016/j.jembe.2012.12.001

Lisin, D., M. Mattar, M. Blaschko, E. Learned-Miller, and M. Benfield. 2005. Combining local and global image features for object class recognition, p. 47–47. *In* CVPR Workshops, San Diego, CA, USA.

Luo, T., K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, and T. Hopkins. 2003. Learning to recognize plankton, v. 1, p. 888–893. *In* IEEE International Conference on Systems, Man, and Cybernetics, Washington, D.C., USA.

Luo, T., K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, and T. Hopkins. 2004. Recognizing plankton images from the shadow image particle profiling evaluation recorder. IEEE Trans. Syst. Man Cybern. Part B Cybern. **34**: 1753–1762. doi:10.1109/TSMCB.2004.830340

Luo, T., K. Kramer, D. B. Goldgof, L. O. Hall, S. Samson, A. Remsen, and T. Hopkins. 2005. Active learning to recognize multiple types of plankton. J. Mach. Learn. Res. **6**: 589–613. http://www.crossref.org/jmlr_DOI.html

Moreno-Torres, J. G., T. Raeder, R. Alaiz-RodríGuez, N. V. Chawla, and F. Herrera. 2012. A unifying view on dataset shift in classification. Pattern Recognit. **45**: 521–530. doi:10.1016/j.patcog.2011.06.019

Orenstein, E. C., O. Beijbom, E. E. Peacock, and H. M. Sosik. 2015. Whoi-plankton-a large scale fine grained visual recognition benchmark dataset for plankton classification. arXiv Preprint arXiv **1510**: 00745.

Pau, G., F. Fuchs, O. Sklyar, M. Boutros, and W. Huber. 2010. EBImage—an R package for image processing with applications to cellular phenotypes. Bioinformatics **26**: 979–981. doi:10.1093/bioinformatics/btq046

Schapire, R. E., and Y. Singer. 1999. Improved boosting algorithms using confidence-rated predictions. Mach. Learn. **37**: 297–336. doi:10.1023/A:1007614523901

Sieracki, C. K., M. E. Sieracki, and C. S. Yentsch. 1998. An imaging-in-flow system for automated analysis of marine microplankton. Mar. Ecol. Prog. Ser. **168**: 285–296. doi: 10.3354/meps168285

Simpson, R., P. Culverhouse, R. Ellis, and B. Williams. 1991. Classification of euceratium Gran. in neural networks, p. 223–229. *In* IEEE Conference on Neural Networks for Ocean Engineering, Washington, D.C., USA.

Solow, A., C. Davis, and Q. Hu. 2001. Estimating the taxonomic composition of a sample when individuals are classified with error. Mar. Ecol. Prog. Ser. **216**: 309–311. doi: 10.3354/meps216309

Sosik, H. M., and R. J. Olson. 2007. Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr.: Methods **5**: 204–216. doi:10.4319/lom.2007.5.204

Tang, X., W. K. Stewart, L. Vincent, H. Huang, M. Marra, S. M. Gallager, and C. S. Davis. 1998. Automatic plankton image recognition, p. 177–199. In S. Panigrahi [ed.], Artificial intelligence for biology and agriculture, North Dakota State University, Fargo, USA.

Tang, X., F. Lin, S. Samson, and A. Remsen. 2006. Binary plankton image classification. IEEE J. Oceanic Eng. **31**: 728–735. doi:10.1109/JOE.2004.836995

Tofallis, C. 2015. A better measure of relative prediction accuracy for model selection and model estimation. J. Oper. Res. Soc. **66**: 1352–1362. doi:10.1057/jors.2014.103

Vandromme, P., L. Stemmann, C. Garcìa-Comas, L. Berline, X. Sun, and G. Gorsky. 2012. Assessing biases in computing size spectra of automatically classified zooplankton from imaging systems: A case study with the ZooScan integrated system. Methods Oceanogr. **1**: 3–21. doi: 10.1016/j.mio.2012.06.001

Vapnik, V. N., and V. Vapnik. 1998. Statistical learning theory, v. **1**. Wiley.

Vapnik, V., and O. Chapelle. 2000. Bounds on error expectation for support vector machines. Neural Comput. **12**: 2013–2036. doi:10.1162/089976600300015042

Ye, L., C. Y. Chang, and C. Hsieh. 2011. Bayesian model for semi-automated zooplankton classification with predictive confidence and rapid category aggregation. Mar. Ecol. Prog. Ser. **441**: 185–196. doi:10.3354/meps09387

Zhao, F., F. Lin, and H. S. Seah. 2010. Binary SIPPER plankton image classification using random subspace. Neurocomputing **73**: 1853–1860. doi:10.1016/j.neucom.2009.12.033

Acknowledgments

This research has been supported by MINECO (the Spanish Ministerio de Economía y Competitividad) and FEDER (Fondo Europeo de Desarrollo Regional), grant TIN2015-65069-C2-2-R. Juan José del Coz is also supported by the Fulbright Commission and the Salvador de Madariaga Program, grant PRX15/00607.

Conflict of Interest

None declared.

Submitted 23 June 2016
Revised 05 October 2016
Accepted 07 November 2016

Associate editor: Paul Kemp

5.3 Automatic plankton quantification using deep features

Artículo aceptado para publicación en **Journal of Plankton Research** el 10 de mayo de 2019.

Autores: Pablo González [1], *Alberto Castaño* [1], *Emily E. Peacock* [2], *Jorge Díez* [1], *Juan José del Coz* [1], *Heidi M. Sosik* [2]

Doi: https://doi.org/10.1093/plankt/fbz023

Resumen

El estudio de datos de plancton marino es de vital importancia para la monitorización de la salud de los océanos. En las últimas décadas, los sistemas de reconocimiento automático de plancton han demostrado ser capaces de procesar enormes cantidades de datos recogidos por dispositivos especialmente diseñados para recoger imágenes digitales directamente de las aguas del océano. Al inicio, estos sistemas fueron desarrollados y puestos en marcha utilizando sistemas tradicionales de clasificación. Estos sistemas utilizaban para describir las imágenes descriptores muy comunes en el campo de la visión artificial como los descriptores de Fourier, con resultados bastante satisfactorios. En los últimos años, ha habido muchos avances en el campo de la visión artificial con el resurgimiento de las redes neuronales. En este artículo se estudia el rendimiento de los descriptores calculados utilizando redes neuronales convolucionales profundas (CNNs) preentrenadas con datos de otro dominio (ImageNet) de cara a reemplazar los descriptores tradicionales en la tarea de estimar la prevalencia de cada clase de plancton en una muestra de agua. Para conseguir este objetivo, hemos diseñado un amplio conjunto de experimentos que muestran cómo este nuevo tipo de características obtienen unos resultados excelentes cuando son combinadas con los algoritmos de cuantificación más avanzados.

[1] Centro de Inteligencia Artificial de Gijón, Universidad de Oviedo
[2] Woods Hole Oceanographic Institute, Massachusetts

Automatic plankton quantification using deep features

Pablo González[a], Alberto Castaño[a], Emily E. Peacock[b], Jorge Díez[a], Juan José del Coz[a], Heidi M. Sosik[b]

[a]*Artificial Intelligence Center, University of Oviedo at Gijón, Spain*

[b]*Biology Department, Woods Hole Oceanographic Institution, Woods Hole, MA 02543*

Abstract

The study of marine plankton data is vital to monitor the health of the world's oceans. In recent decades, automatic plankton recognition systems have proved useful to address the vast amount of data collected by specially engineered in situ digital imaging systems. At the beginning, these systems were developed and put into operation using traditional automatic classification techniques, which were fed with hand-designed local image descriptors (such as Fourier features), obtaining quite successful results. In the past few years, there have been many advances in the computer vision community with the rebirth of neural networks. In this paper, we leverage how descriptors computed using Convolutional Neural Networks (CNNs) trained with out-of-domain data are useful to replace hand-designed descriptors in the task of estimating the prevalence of each plankton class in a water sample. To achieve this goal, we have designed a broad set of experiments that show how effective these deep features are when working in combination with state-of-the-art quantification algorithms.

Keywords: Abundance estimation, quantification, deep learning, convolutional neural networks, phytoplankton

1. Introduction

Phytoplankton play a vital role in marine ecosystems. Since the creation of automatic plankton imaging systems, many efforts have been devoted to the development of automatic techniques for processing all the data captured to optimize conclusions from temporally dense data sets (Benfield et al., 2007).

In the last few years, attention has turned to Convolutional Neural Networks (CNNs) that push the limit

Email addresses: `gonzalezgpablo@uniovi.es` (Pablo González), `castanoalberto@uniovi.es` (Alberto Castaño), `epeacock@whoi.edu` (Emily E. Peacock), `jdiez@uniovi.es` (Jorge Díez), `juanjo@uniovi.es` (Juan José del Coz), `hsosik@whoi.edu` (Heidi M. Sosik)

of computer vision techniques, but before that, systems designed to automatically classify plankton were trained with hand-designed descriptors. These descriptors were a reduced representation of each image, that were used for training and testing machine learning algorithms such as Random Forest or Support Vector Machines (SVM). These well studied descriptors included shape and texture features, such as Fourier descriptors (Kuhl and Giardina, 1982), Haralick features (Haralick and Shanmugam, 1973), invariant moments (Hu, 1962), etc. When using CNNs, these descriptors no longer need to be computed for each image. Convolutional layers in the CNN serve as feature extractors, gaining in complexity as we move forward into the network, while pooling layers are designed to reduce the spatial resolution of the feature maps, obtaining then invariance to translations and distortions. The network itself learns a representation of the images when adjusting the weights of its different layers. This approach has been shown to be superior to hand-designed descriptors (Sharif Razavian et al., 2014) and it was applied by most of the teams participating in the National Data Science Bowl (NDSB) competition (**?**), where participants had to classify plankton images, and 81.5% accuracy was reached across 121 different categories.

While CNNs can be used to build a classifier for a specific dataset, as the teams of the NDSB competition did, they can also be used to extract a numerical representation of a given image (as an alternative to hand-designed descriptors). After feeding a CNN with a plankton image and evaluating all the activations of the network, activations in fully connected layers are compressed representations of the image and can be used as image descriptors. These descriptors, called deep features, have been applied successfully to many computer vision problems (Oquab et al., 2014; Chatfield et al., 2014).

One of the main problems with CNNs is that they are computationally expensive. They need special hardware to be trained (powerful GPUs) and the time needed for training a big CNN with a respectable amount of data is usually counted in weeks. One possible solution for this issue is to use a CNN already trained with a set of images belonging to a different domain, a technique known as transfer learning (Pan and Yang, 2010). With this approach, a pre-trained CNN can be used to compute deep features of plankton images. To improve the results, the network can be fine-tuned with labeled plankton images, so the network weights are adjusted better to the plankton domain.

Transfer learning is a technique that has been around for a few years and that has increased in importance since the growth in popularity of CNNs. It has been applied to the WHOI-plankton dataset (Sosik et al., 2015) with promising results in terms of classification accuracy (Orenstein et al., 2015). Recently, increasingly more powerful CNNs have been developed with larger numbers of layers (He et al., 2016), leading to astonishing results over the ImageNet dataset (Deng et al., 2009). These very deep pre-trained networks are usually openly available, presenting us with the opportunity to test their performance in challenging problems such as the WHOI-plankton dataset where the objective is to estimate the prevalence of plankton taxa in a water sample.

The task of predicting the prevalence of each taxon in a given sample has often been tackled with image classification techniques. The most basic approach uses a classifier to assign a class to each plankton image and then counts them. We shall call this approach "Classify & Count". Although this method has some efficacy, it is suboptimal and can be improved with methods specifically designed for quantification (González et al., 2017c), such that the aggregated underlying distribution is considered rather than individual classifications.

We are interested in estimating the prevalence of each class in an unknown water sample. To that end, we have used quantification algorithms with deep features as their input, and we have analyzed their performance with a rigorously designed validation methodology (González et al., 2017a) where the sample is the minimum test unit.

In recent years, different deep learning algorithms have been applied for plankton classification, see Moniruzzaman et al. (2017) for a small survey of some of these applications. However, to the best of our knowledge, only one other paper (Beijbom et al., 2015) has studied the use of deep learning for plankton abundance estimation using quantification algorithms before. In the majority of the papers published on the topic, authors use CNNs as classifiers rather than as quantifiers. For instance, Py et al. (2016) and Luo et al. (2018) describe two systems based on CNNs able to automatically classify 121 and 108 types of plankton, respectively. Dunker et al. (2018) and Lloret et al. (2018) use CNN classifiers for identifying phytoplankton species. Dai et al. (2016a) present a similar approach in the design of a zooplankton classifier. Other authors combine CNNs with different machine learning techniques, including active learning (Bochinski et al., 2018), hybrid systems (Dai et al., 2016b), parallel networks (Wang et al., 2018), imbalance learning (Lee et al., 2016) or different forms of information fusion (Cui et al., 2018; Lumini and Nanni, 2019). It should be noted that the improvements on plankton classification described in these papers may not be directly transferable to quantification systems, as classification and quantification are two different tasks. Importantly, capturing the changes in the distribution between training data and test samples (González et al., 2017b) is crucial when dealing with quantification. All proper quantification algorithms have some mechanism to detect and deal with such changes (see Section 2.4). Furthermore, classification and quantification use different target performance measures. While classification requires performance metrics that measure classification accuracy at the individual image level (e.g., how likely it is that an image of a given taxon will be classified correctly), quantification focuses on sample-level errors instead (e.g., how precise is the estimated concentration of a given taxon). The correlation between both performances is lower than expected, see González et al. (2017a) for further details. Thus, plankton quantification should be properly studied through a well-designed set of experiments, different from those commonly used in plankton classification papers.

Beijbom et al. (2015) apply four quantification algorithms (Forman, 2008; Saerens et al., 2002) based on CNNs classifiers to automatically estimate the abundance of 33 classes over 21 test samples. The present paper expands such study in several directions:

1. Applying standard CNNs, with and without fine-tuning, as feature extractors. The goal is to analyze whether fine-tuning helps to significantly improve quantification performance.

2. Employing CNNs to obtain deep features, rather than as classifiers. Since training CNN classifiers may be complex for some users, this paper proposes the use of deep features provided by already trained CNNs in combination with easy-to-train quantification algorithms.
3. Comparing deep features with hand-crafted features (e.g. shape and texture features) that were the standard until the emergence of deep learning algorithms.
4. Performing more exhaustive experiments. In Beijbom et al. (2015) only 21 test samples were used, a number that we consider to be too low for analyzing quantification performance, and those classes with less than 1000 examples were removed. Our study comprises 764 test samples considering all the 49 classes present in those samples. Additionally, the computational cost of the compared approaches is also analyzed.

Our aim is to show that an approach that combines standard CNNs with basic quantification algorithms outperforms traditional machine learning methods over the WHOI-plankton dataset.

For the sake of reproducibility, all relevant source code used to run experiments has been made available for download[1], along with the full results drawn from all the experiments that were not included in this paper[2].

2. Material and Methods

2.1. Dataset

The WHOI-Plankton dataset (Sosik et al., 2015) was used for all of the experiments. This dataset is publicly available and it has been used by a few papers on this topic (Beijbom et al., 2015; Lee et al., 2016; Orenstein and Beijbom, 2017). The WHOI-Plankton data was collected with a multi-year series of Imaging FlowCytobot (IFCB) (Olson and Sosik, 2007) deployments at the Martha's Vineyard Coastal Observatory (MVCO), which is a facility operated by Woods Hole Oceanographic Institution (WHOI). The MVCO site is a component of the Northeast U.S. Shelf Long-Term Ecological Research (NES-LTER) program where the IFCB time series contributes critical information to characterize and understand the roles of plankton in ecosystem function. At MVCO, IFCB automatically draws in a 5-ml sample of seawater every 20 minutes. The seawater sample is pumped through a cytometric system, where particles that contain chlorophyll and are in the approximate size range 10 to 150 μm are imaged. Regions of interest (ROIs) containing plankton targets are extracted from the camera frame in realtime during a sample run. These ROIs are stored onboard the IFCB and transmitted to shore over an Ethernet connection. At MVCO, IFCB has captured nearly 1 billion images since 2006.

[1]`https://github.com/pglez82/IFCB_quantification`
[2]`https://pglez82.github.io/IFCB_quantification`

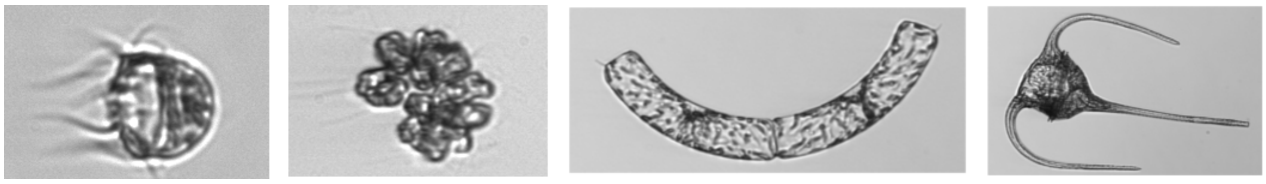


Figure 1: Examples of ROIs from the WHOI annotated dataset.

From this huge quantity of images, NES-LTER researchers at WHOI have annotated 3.5 million ROIs belonging to more than 5000 different samples. They have manually sorted images into 103 different classes, which can be grouped together into 51 more generic categories. Notably, in these experiments we have used these image categories as determined by NES-LTER researchers working with the image data to address unresolved ecological questions. An example of IFCB images from MVCO can be seen in Figure 1. The size of the images varies depending on the size of the organism, typically ranging from 100 to 100,000 pixels. All images are gray-scale.

The WHOI-Plankton dataset has a highly unbalanced distribution in which over 90% of images belong to only 5 classes and the most prevalent class (miscellaneous nanoplankton) represents 75% of all ROIs in the dataset.

It is important to highlight that every sample has a different plankton distribution, due to temporal variations in the natural community.

2.2. Data preprocessing

The WHOI-plankton dataset images are distributed in more than 5000 samples. From all these samples only the fully annotated ones (all the individuals in the sample have been annotated) were considered for the experiment in order to properly test quantification methods. The resulting dataset contains 3.4 million images organized in 964 samples collected between 2006 and 2014. The categories considered were the ones suggested in Sosik et al. (2015), which comprises 51 different classes. From these 964 samples, the dataset was split into training and test sets taking the first 200 samples (in temporal sequence) as the training set and the rest as the test set. Because two classes contained no examples in the first 200 samples, this split resulted in a 49-class training set and resulting 49-class quantifiers evaluated in this work.

This experimental setting follows the guidelines suggested in (González et al., 2017a) trying to simulate a realistic case in which: 1) training data is collected during a sufficiently long period of time, 2) a model is learned using these training data and 3) finally such model is deployed to automatically process subsequent samples. Notice that only 20% of all available data has been chosen for the training set but these first 200 samples represent all data collected from 2006 to 2008, a length of time which should assure the classes of interest are represented. It is likely that using a larger number of samples in the training set could result in improved performance for the whole system, but the trade-off is increased time required to learn each model. Our choice balances adequate performance with training time that is fast enough to run all the experiments reported in this paper (see Table 4).

All plankton images have been resized to match the inputs of the CNNs. In this case, images had to be 224x224 pixels. As most of the IFCB images were not square, each one was resized keeping its original aspect ratio and so that its longest dimension is 224 pixels; then the resulting scaled ROI was placed in the middle of the image and the two lateral gaps were filled with the value of the average pixel, computed from 30.000 plankton images randomly selected from the data set.

Hand-coded features were downloaded from the publicly available MVCO IFCB Dashboard website[3], where standard computations are provided for the entire dataset (Sosik and Olson, 2007; Sosik et al., 2016). For each image a vector with 227 features was downloaded, including shape and texture features. The computation process for these features is carefully documented in Sosik (2017). From now on, we will refer to this feature set as *normal features* (NF) for which performance will be compared against the features computed using Convolutional neural networks (CNNs), also known as *deep features*.

2.3. Deep features

Convolutional networks (CNNs) have recently enjoyed great success in large-scale image recognition tasks. This has been made possible by the existence of large public image repositories, such as ImageNet (Deng et al., 2009), and the increase of computing capacity. CNNs used by the computer vision community have been growing deeper and deeper since AlexNet (Krizhevsky et al., 2012) was proposed in 2012 with only 8 layers. Nowadays, deep residual networks (resnets) (He et al., 2016) contain more than one hundred layers. The resnet architecture solves the notorious vanishing gradient problem (Hochreiter et al., 2001) that emerges when training very deep networks by introducing short cut connections that skip one or more layers. Notably, residual networks were successful in winning the ImageNet ILSVRC 2015 competition with an incredible error rate of 3.6% (humans generally hover around a 5-10% error rate).

When a CNN is trained on images, like those in ImageNet, to perform image classification, it automatically learns features that will vary in complexity depending on the layer depth. On the first layers, features similar to Gabor filters that act as edge and contour detectors are learned. These features are not specific to a particular dataset. Deeper layers in the network learn more complex features, usually from a combination of features from early layers, that resemble shapes or forms, that are also more specific to the dataset in hand. Nonetheless, this specificity is not a problem since the ImageNet dataset is sufficiently varied. When a new image (in our case, a plankton image) is presented to the CNN, this set of learned features will be computed in each of the network layers. These deep features, can then be used as the input for the different quantification algorithms.

[3]`http://ifcb-data.whoi.edu/mvco`

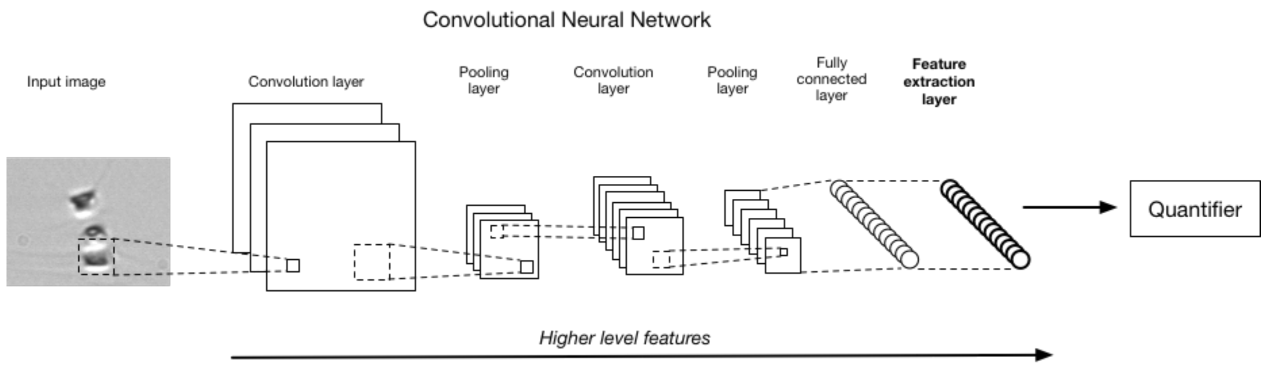


Figure 2: CNN architecture used for deep feature extraction. Layers on the right contain higher level features. The network input is a raw image resized to fit the input layer.

For the problem at hand, we have used resnets pre-trained on the ImageNet dataset, even though other network architectures could be considered (see for example, Huang et al. 2016). ImageNet contains images totally different from plankton images containing only macroscopic images such as animals, landscapes, etc. We tested different pre-trained versions of this network, varying the number of layers from 18 (RN-18) to 101 (RN-101). Our goal was to determine the degree of complexity necessary to obtain satisfactory results. To compute deep features for each plankton image, activations were pulled from the final fully connected layer of a network during a forward pass of each IFCB plankton image (see Figure 2). The number of deep features obtained for each image depends on the network used, varying between 512 for RN-18 and RN-34 to 2048 for RN-50 and RN-101.

Even though off-the-shelf CNN deep features have good discriminative power (Sharif Razavian et al., 2014), results can be improved by fine-tuning the networks to actual plankton images. To fine-tune the CNNs and adapt them to plankton images, we replaced the last fully connected layer of the CNNs (which is designed for classifying ImageNet) with an output layer matching the number of classes in our dataset. The network was then trained with the labeled images from the training set in order to adapt its weights to plankton images. In our experiments, we used 30 epochs, being an epoch a full pass of the whole training dataset (half a million images), with a learning rate of 0.01, which was decreased by an order of magnitude after completing the first 15 epochs.

All fine-tuning and deep feature computing was done with the R deep learning package MXNet (Chen et al., 2015) on 2xNVIDIA K80 GPUs. Times needed for fine-tuning the networks and computing the deep features are shown in Table 4.

2.4. Quantification algorithms

In the wide variety of problems that machine learning faces, there are tasks in which the individual class predictions are not as important as predicting the proportion of each class in a concrete sample or set of examples. This problem is called quantification (Forman, 2008) in machine learning and data mining communities. Quantification is a learning problem on its own because it requires specific approaches, and not just using classification methods. In fact, many experiments (Barranquero et al., 2013, 2015; González et al., 2017c) have shown that off-the-shelf classifiers are often suboptimal when applied directly to quantification tasks. For that reason, several quantification algorithms have been proposed during the past few years (Firat, 2016; Narasimhan et al., 2016; Pérez-Gállego et al., 2017, 2019). A review of quantification learning can be found in González et al. (2017b).

Given a dataset $D = \{(x_1, y_1), ..., (x_n, y_n)\}$, in which x_i is a representation of an individual example in the input space $\mathcal{X}$ and $y_i \in \mathcal{Y} = \{c_1, ..., c_l\}$ is the corresponding class label, the goal in supervised classification is learning a model:

$$h : \mathcal{X} \longrightarrow \{c_1, ..., c_l\}, \tag{1}$$

able to assign a class label for a new unseen example. In quantification, the learning task is totally different from a formal point of view; it can be defined as follows:

$$\bar{h} : Sample \longrightarrow [0, 1]^l. \tag{2}$$

In this case the model $\bar{h}$ returns a l-dimensional vector in which each element, $\hat{p}_j$, represents the predicted prevalence for class j for the input sample, such that

$$\begin{aligned} &\sum_{j=1}^{l} \hat{p}_j = 1, \\ s.t. \quad &0 \leq \hat{p}_j \leq 1, \forall j = 1, \ldots, l. \end{aligned} \tag{3}$$

That is, $\bar{h}$ predicts the class probability distribution of a sample. Despite the fact that most quantifiers have been designed for binary problems ($l = 2$), multiclass quantification ($l > 2$) can be solved combining the results of l binary quantifiers. In this paper, we use the well-known one-vs-all approach that learns a collection of binary quantifiers:

$$\bar{h}_j : Sample \longrightarrow [0, 1]. \tag{4}$$

Each $\bar{h}_j$ just returns the proportion of examples of class j in the sample. The initial predictions of all the binary models, $\{\, \hat{p}^0_j \mid j = 1, \ldots, l\}$, are finally normalized in order to satisfy (3):

$$\hat{p}_j = \frac{\hat{p}^0_j}{\sum_{j=1}^{l} \hat{p}^0_j}. \tag{5}$$

In this study, we have evaluated a set of quantification algorithms developed in the literature for application to binary quantifiers. These approaches were mainly selected because their implementation is very simple and any practitioner with a little knowledge of machine learning could implement these algorithms. We briefly describe each of them here.

The Classify & Count (CC) approach follows the most intuitive way to tackle a quantification problem: build a classifier and count the examples falling into each class. We shall consider this method as a baseline as it is not formally a quantification method, even though it is used in the evaluation of many automatic plankton recognition systems (González et al., 2017a). The problem with the CC method is that its performance degrades when there are significant changes in class distributions (González et al., 2017b).

The Adjusted Count (AC) algorithm, proposed by Forman (Forman, 2008), is theoretically well founded and based on making a correction to the prevalence estimated by the CC method, $\hat{p}_j^{CC}$, using the classifier true positive rate (tpr) and false positive rate (fpr) for the target class j:

$$\hat{p}_j^{AC} = \frac{\hat{p}_j^{CC} - fpr}{tpr - fpr} \tag{6}$$

which would lead to a perfect prediction given that the estimation of tpr and fpr is perfect and $P(x|y)$ is constant [see further details in Forman (2008)]. Even when these two conditions are not completely fulfilled, AC usually works better than CC (see Section 3). This approach has been previously used in the plankton domain for correcting abundance estimates with a high degree of success (Sosik and Olson, 2007).

The methods referred to as Probabilistic Classify & Count (PCC) and Probabilistic Adjusted Count (PAC) (Bella et al., 2010) work with an underlying probabilistic classifier instead of a crisp one. The prevalence is then computed as the average of the probability of belonging to class j for all the examples in a test sample T:

$$\hat{p}_j^{PCC} = \frac{1}{|T|} \sum_{x \in T} P(y = c_j|x) \tag{7}$$

In the PAC method, this result is adjusted in an analogous way as in the AC method.

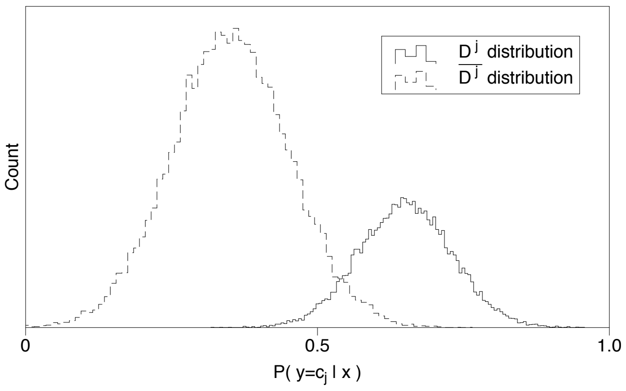
$$\hat{p}_j^{PAC} = \frac{\hat{p}_j^{PCC} - FP^{pa}}{TP^{pa} - FP^{pa}} \tag{8}$$

in which TP^{pa} (*TP probability average*) and FP^{pa} (*FP probability average*), are estimated from the training dataset, and are defined as:

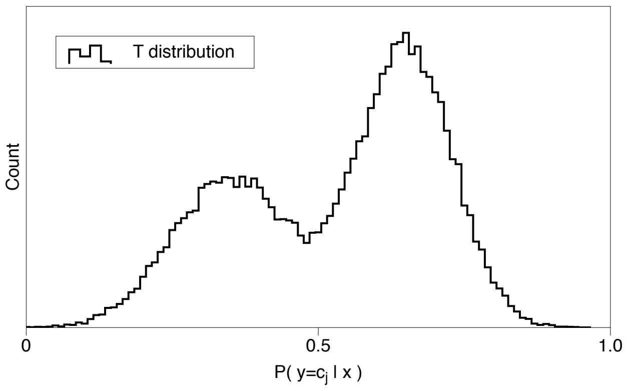
$$TP^{pa} = \frac{\sum_{x \in D^j} P(y = c_j|x)}{|D^j|} \quad \text{and} \quad FP^{pa} = \frac{\sum_{x \in \overline{D^j}} P(y = c_j|x)}{|\overline{D^j}|} \tag{9}$$

where D^j is the set of training examples in class j and $\overline{D^j}$ is the rest of the training examples in D.

The HDy method (González-Castro et al., 2013) is based on matching probability distributions where the Hellinger Distance (HD) is the metric to compute the difference between such distributions. It uses the outputs of a binary classifier to represent the distributions for D^j, $\overline{D^j}$ and the test set T (with binning to approximate the integral in the definition of HD), see Figure 3. The idea is to combine the distributions



(a) D^j and $\overline{D^j}$ distributions



(b) Test sample distribution

Figure 3: HDy computes first the probability density functions of (a) the examples from D^j and $\overline{D^j}$. Eq.(10) combines both distributions for different values of $\hat{p}$, to approximate (b) the distribution of the test set sample.

D^j and $\overline{D^j}$, by means of their prevalences, to approximate the observed distribution in T:

$$\hat{p}_j^{HDy} = \min_{\hat{p}_j \in [0,1]} \sqrt{\sum_{k=1}^{bins} \left(\sqrt{\frac{|T_k|}{|T|}} - \sqrt{\frac{|D_k^j|}{|D^j|} \cdot \hat{p}_j + \frac{|\overline{D^j}_k|}{|\overline{D^j}|} \cdot (1 - \hat{p}_j)} \right)^2} \ . \tag{10}$$

For instance, in the example depicted in Figure 3, the prevalence of class j is 0.4 in the training set (a) and 0.6 in T (b). To match the distribution of T, we need to increase $\hat{p}_j$ to give more importance to D^j distribution. A simple linear search in which $\hat{p}_j$ moves over the range [0,1] in small steps is used to select the predicted prevalence for class j that minimizes the HD.

These quantification algorithms have been implemented in Python and are publicly available as a Python module called PyQuan[4]. PyQuan is able to tackle multi-class quantification problems with n binary quantifiers using a one-vs-all approach. As explained above, a binary quantifier is trained for each class j considering examples of this class as the positive class and the rest as the negative. We trained only one model per class and used it for each of the quantification algorithms described above (CC, AC, PCC, PAC and HDy). This guarantees that the differences in performance between them are only due to the way in which each method employs the predictions made by the binary classifiers. We use linear regression as the underlying binary classifier as it is simple and fast enough to be trained with this dataset in reasonable time and it provides probabilistic outputs, which are needed for the PCC and PAC methods. The regularization parameter C was adjusted for each model with a grid search over the values $(0.1, 1, 10)$.

To run the quantification algorithms, we used a machine with 2 Haswell 2680v3 processors, 24 cores, and 120Gb of RAM. All experiment times are shown in Table 4.

[4]`https://github.com/albertorepo/quantification`

2.5. Performance measures

To compare the different methods and descriptors used in this paper, we need to define the evaluation method and performance measures. Given the characteristics of this dataset and its inherent properties (see Section 2.1), we chose a validation method that ensures that results are transferable to production like conditions. That is, testing should be carried out with different samples presenting different plankton distributions, covering the actual variations due to seasonalilty or any other factors. The dataset comprises 964 samples, where the first 200 (ordered from oldest to newest) are used for training, having 764 remaining samples for the test set. In this work, the evaluation guidelines proposed in González et al. (2017a) have been followed. Thus, given the high number of samples, the evaluation method chosen has been a hold-out by sample, where the model to evaluate is used to predict the distribution of all the samples in the test set. An important precondition for properly evaluating quantification algorithms is that samples must be complete, meaning that all examples present in a sample have to be annotated and placed in one class and no example can be manually discarded.

Performance measures included in this paper are Mean Absolute Error (MAE) and Mean Relative Absolute Error (MRAE). Given the true prevalences $\{p_{j,s} : s = 1, \ldots, m\}$ of class c_j over m labelled samples, $\{T_1, \ldots, T_m\}$, and the predicted prevalences $\{\hat{p}_{j,s} : s = 1, \ldots, m\}$, these performance measures can be defined as:

- Mean Absolute Error: $MAE(c_j) = \frac{1}{m} \sum_{s=1}^{m} |p_{j,s} - \hat{p}_{j,s}|$
- Mean Relative Absolute Error: $MRAE(c_j) = \frac{1}{m} \sum_{s=1}^{m} \frac{\epsilon + |p_{j,s} - \hat{p}_{j,s}|}{\epsilon + p_{j,s}}$, where ϵ is a small constant that prevents the function from being undefined when $p_{j,s} = 0$.

We do not compare classification accuracy for individual images for several reasons: 1) our goal is to tackle the abundance problem in which predictions at the individual level are not relevant, 2) in fact, some of the compared algorithms (e.g. AC, PAC and HDy) do not provide such individual classifications, and 3) it has been shown that the correlation between classification accuracy and quantification accuracy is much lower than expected; see González et al. (2017a) for a complete analysis on this issue.

3. Results

All experiment results are fully available online[5] as an interactive web application, allowing the user to compare between different feature sets and quantification methods, for each class in the dataset.

The annotated dataset described in Section 2.1 was used for the experiments. Hand-coded descriptors (”normal” features, NF) downloaded from the MVCO IFCB Dashboard (see Section 2.2) were used as a baseline to compare with results from descriptors obtained with CNNs. The CNN-derived descriptors

[5] `https://pglez82.github.io/IFCB_quantification/`

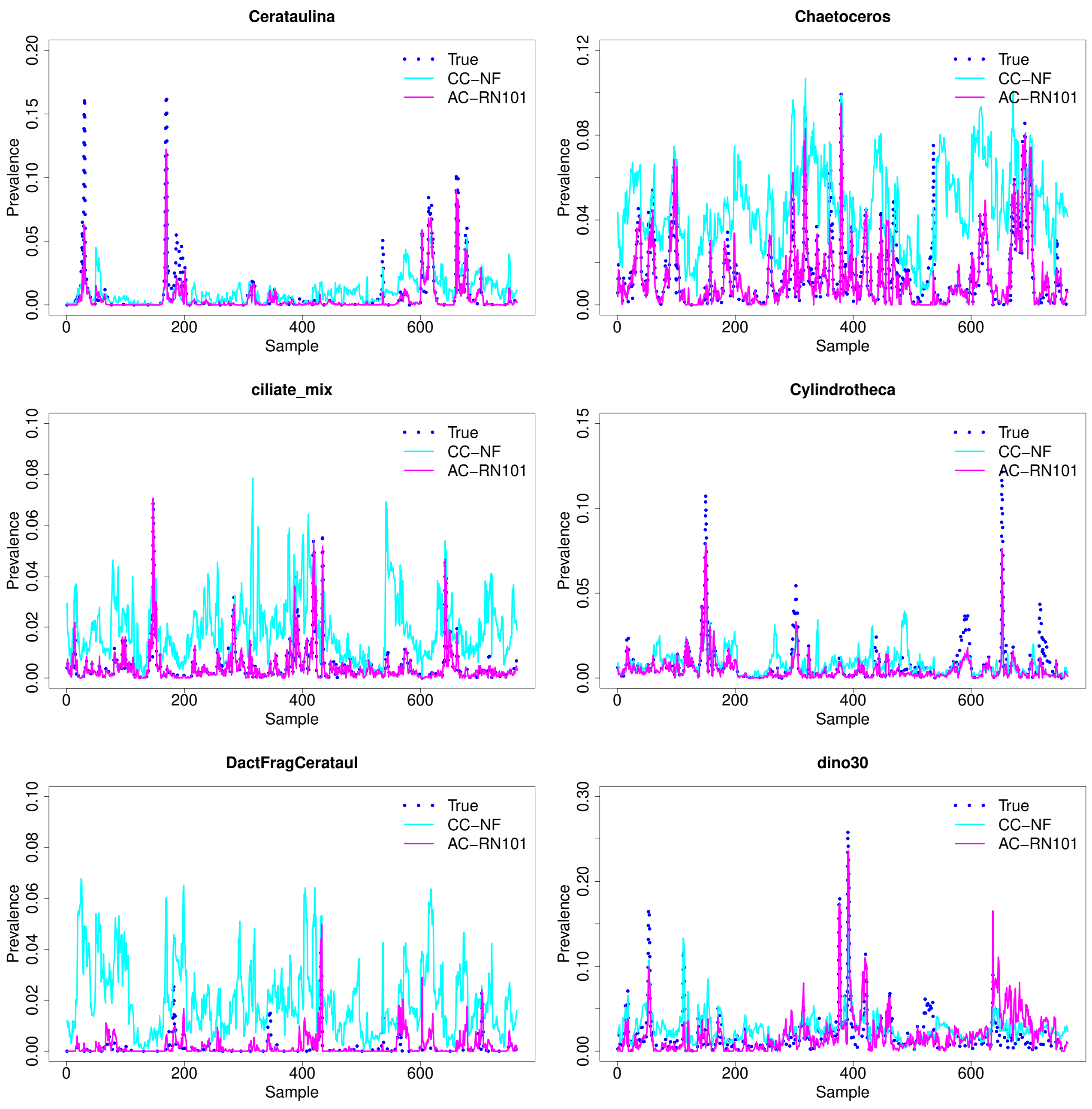


Figure 4: (Part 1) Results for each sample comparing true prevalence and model outputs using NF (normal features) with CC method and RN (fine-tuned RN-101 features) using AC method.

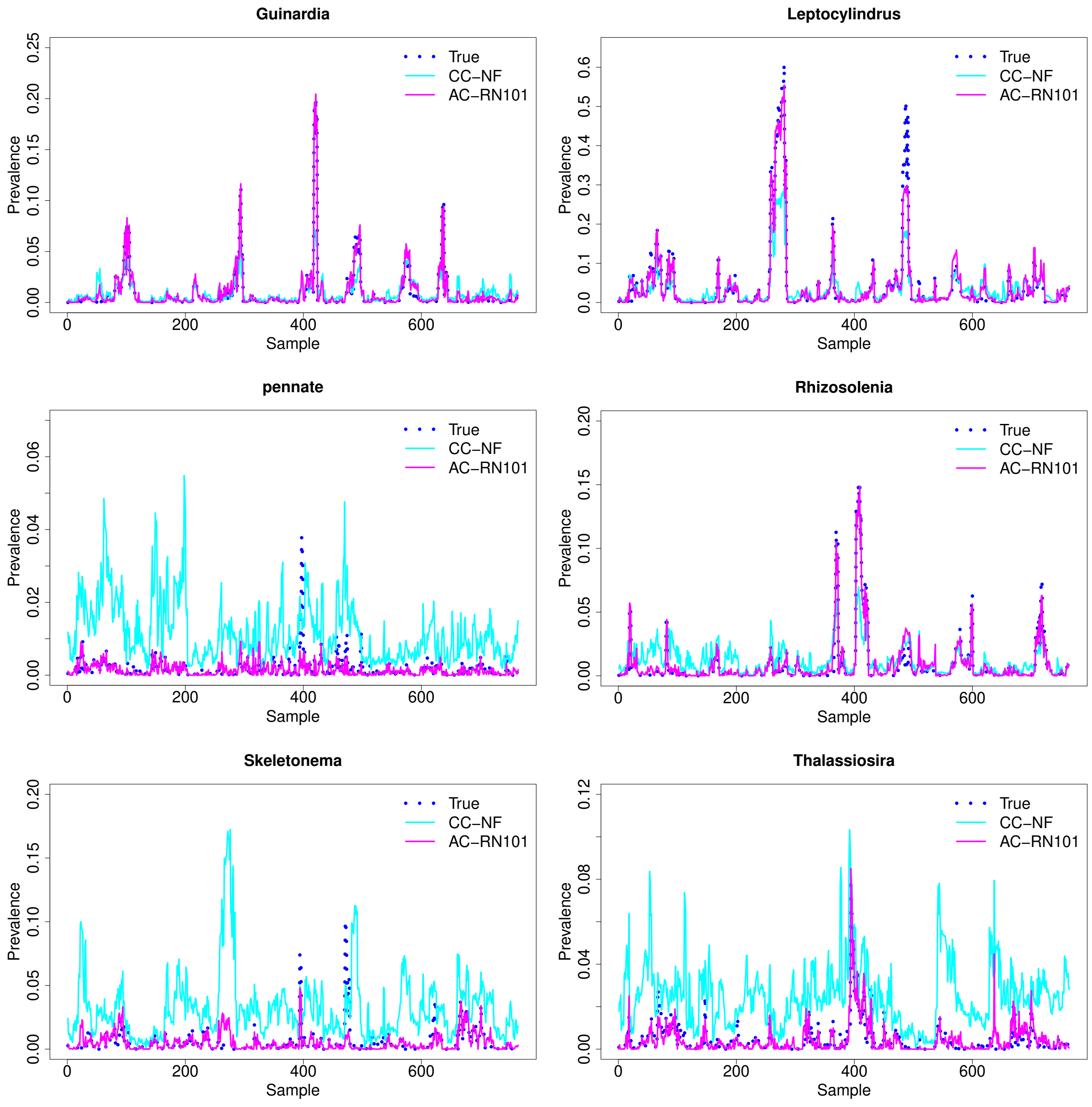


Figure 5: (Part 2) Results for each sample comparing True prevalence and model outputs using NF (normal features) with CC method and RN (fine-tuned RN-101 features) using AC method.

	NF		**RN-101**	
Class	**CC** (10^{-4})	**AC** (10^{-4})	**CC** (10^{-4})	**AC** (10^{-4})
Cerataulina	89.25 / 4.39	61.97 / 4.61	28.44 / 2.87	**27.52** / 2.79
Chaetoceros	334.28 / 6.17	136.26 / 5.47	42.49 / 1.86	**38.81** / 1.97
ciliate_mix	148.76 / 3.96	71.13 / 4.29	6.56 / 0.40	**6.47** / 0.36
Cylindrotheca	58.14 / 2.82	63.26 / 3.00	23.18 / 1.98	**22.50** / 1.93
DactFragCerataul	192.36 / 4.97	75.18 / 5.29	20.19 / 1.06	**14.69** / 1.04
dino30	148.90 / 4.82	128.98 / 4.78	138.26 / 5.05	**121.86** / 5.20
Guinardia	62.17 / 4.42	50.89 / 3.97	20.88 / 1.27	**20.48** / 1.28
Leptocylindrus	217.77 / 16.64	184.05 / 12.26	91.86 / 7.48	**87.28** / 7.17
pennate	101.97 / 2.83	30.42 / 2.28	**8.05** / 0.79	8.35 / 0.79
Rhizosolenia	82.21 / 3.95	61.50 / 3.55	20.99 / 1.41	**20.00** / 1.39
Skeletonema	272.94 / 9.43	229.31 / 13.01	**28.52** / 2.65	28.57 / 2.66
Thalassiosira	210.93 / 4.94	75.49 / 5.07	19.93 / 0.98	**19.32** / 0.99

Table 1: Absolute Errors (AE) Mean / Standard Error for 12 classes over all test samples (full table can be found in the supplemental material). Results for CC and AC methods using normal features (NF) and RN-101 features. Lowest errors per class shown in bold.

were computed with residual deep networks (see Section 2.3). These deep features were used for training and testing the quantification algorithms in the same way as the hand-coded descriptors. All experiment times have been logged with the aim of giving a general view of the computing time needed to apply these methods (see Table 4). With the class prevalences calculated for each quantification method, Absolute Errors and Relative Absolute Errors were computed (see Table 1).

We found that deep features perform better than normal features resulting in a lower absolute error for all classes. This difference is illustrated in Figures 4 and 5. For a given class, it is important to observe how the true prevalence varies from sample to sample, sometimes very abruptly. Predictions made with deep features get a superior level of adjustment compared to traditional features. There are some classes like mix_elongated or ciliate_mix where predicted prevalences from deep features are almost perfect. It is interesting to note how true prevalences vary over samples. For instance in the class *Leptocylindrus*, true prevalence goes from less than 0.1 to 0.61 in sample 280. Similar changes can be observed for most classes. These variations make this problem challenging and suitable for quantification techniques.

Differences between the CC and AC method are very small when we deal with very low absolute errors. For RN-101 features, AC is almost equivalent to CC. The mean absolute error by class for CC is 0.0035 where the same value for AC is 0.0031. This difference is greater when features do not work as well. For instance, with normal features, error decreases from 0.0149 with CC to 0.0.0084 with AC, a 43% decrease in absolute error. On the one hand, when the CC method already gets very good results, the margin for improvement is too low to be noticeable. On the other hand, when dealing with a complex quantification problem like this one, the conditions for a perfect adjustment are only met to a certain degree (tpr and fpr estimations are not perfect and $P(x|y)$ varies across the dataset).

For other quantification methods, it is interesting to see that adjustments usually work better than the CC

	NF	**RN-18***	**RN-18**	**RN-34***	**RN-34**	**RN-50**	**RN-101**
CC	0.0149	0.0111	0.0103	0.0110	0.0070	0.0036	0.0035
AC	0.0084	0.0208	0.0074	0.0211	0.0268	0.0031	0.0031
PCC	0.0171	0.0133	0.0118	0.0130	0.0082	0.0045	0.0044
PAC	0.0089	0.0077	0.0082	0.0072	0.0058	0.0035	0.0034
HDy	0.0075	0.0063	0.0071	0.0057	0.0055	0.0054	0.0062

Table 2: Mean Absolute Error (AE) by class for all the CNN tested. CNNs with * have not been fine-tuned to plankton images.

	NF	**RN-18***	**RN-18**	**RN-34***	**RN-34**	**RN-50**	**RN-101**
CC	17.64	9.15	9.69	8.53	4.45	2.13	2.08
AC	9.99	67.03	7.87	69.85	94.16	1.88	1.87
PCC	20.34	11.96	12.18	11.11	6.67	3.71	3.37
PAC	10.24	7.07	8.95	5.95	4.09	2.50	2.41
HDy	7.9	5.71	8.19	5.16	4.67	11.31	13.97

Table 3: Mean Relative Absolute Error (MRAE) by class for all the CNN tested. CNNs with * have not been fine-tuned to plankton images.

method. For instance, AC improves the results in four out of seven experiments. Taking a closer look at experiments where AC has underperformed CC (RN-18*, RN-34* and RN-34), the problem is caused by a few classes (such as *Stephanopyxis*) with very few training examples and nearly zero prevalence over all samples. With such a low number of examples for a class, it is possible for AC to compute a tpr and fpr almost zero. In this case, it is easy to see how a very small denominator in Equation 6 can lead to a high error in the adjustment. Since Table 2 and Table 3 errors are averaged by class, giving the same importance to every class in the dataset, the errors can be misleading without considering each class individually.

Similar conclusions can be drawn looking at the Mean Relative Absolute Errors (see Table 3). The same problem is observed with class *Stephanopyxis*, where the relative error is very high for the AC method in three experiments. In the rest of the experiments, values are equivalent to those in the Absolute Error table and show how well RN-50 and RN-101 with AC work, with an error lower than 2%.

Another interesting conclusion is that PAC outperforms PCC in all seven experiments (both in absolute and relative errors). The adjustment in PAC works similarly to AC (see Equation 8), but seems more resistant than AC to the problems due to very infrequent classes, mainly because PAC does not use a threshold when deciding if an example belongs or not to a certain class, as this method works with the raw probabilities returned by the classifier. Also, HDy appears as a very solid method, giving very good results even with normal features.

Mean absolute errors by class (see Table 2) show errors for shallower residual networks. It is important to note than even the smaller network with 18 layers and without fine-tuning (RN-18*) outperforms

	NF	RN-18	RN-34	RN-50	RN-101
Fine-tuning CNN	X	11	17	36	65
Compute deep features	X	27	30	32	45
Quantification	72	96	96	224	226

Table 4: Comparative of times (in hours) needed for making the experiments using 2 GPUs NVIDIA Tesla K80 for GPU tasks (fine-tuning and compute deep features) and 2 processors Haswell 2680v3, 24 cores with 120Gb RAM memory for CPU tasks (quantification).

normal features. This result leads to the conclusion that the process of fine-tuning is desirable but not required. Previous studies (Sharif Razavian et al., 2014) have shown how off-the-shelf deep features work better than hand-coded features and this claim is confirmed by our work. Nonetheless, it is important to note that fine-tuning is a process that is done only once in the model building phase and it is not very computationally expensive (65 hours of GPU computing for the biggest network tested: RN-101). Fine-tuning leads to an improvement over 7% in absolute error for RN-18 and a 36% improvement for RN-34.

It is important to notice how errors decrease with deeper networks. The largest difference takes place from 34 layers to 50 layers, where absolute error for the CC method decreases from 0.0070 to 0.0036 (49% improvement). From there, even doubling the number of network layers (from 50 to 101) only results in a 2% decrease in absolute error. This improvement also has a drawback in computation time. In addition, the number of deep features computed with RN-50 is four times higher than for RN-34 (2048 vs. 512). Table 4 shows how time increases from 96 hours to more than 200 hours for building the model and applying quantification algorithms. Part of this time increment has its origin in the memory requirements to fit a dataset four times bigger. With a machine with 120Gb of RAM, we were able to build up to twelve binary models at the same time for 512 deep features, but only four parallel models for 2048 deep features.

4. Discussion

We have evaluated how well deep features perform when trying to estimate the abundances of plankton species in a water sample. Conforming with current computer vision literature, deep features have proven far more powerful than traditional hand-designed descriptors. Even the smallest networks, pre-trained with out-of-domain data, are able to compete against traditional features.

Deep features are applicable to most problems in the computer vision field. Nowadays they should be considered above hand-designed descriptors given their robust performance, as shown by this and many other recent studies. Even if a dataset is not big, and fine-tuning is not an option, pre-trained CNNs should still remain as a viable alternative.

The power of the improved quantification accuracy achieved with deep features is most evident when we consider the implications for understanding important ecological problems. A major objective of IFCB deployments at MVCO is to characterize taxon-specific bloom occurrences and temporal changes in community structure in the plankton. The number of images in the multi-year dataset (nearly 1 billion) makes full expert validation of image classifications prohibitive. Traditional hand-descriptor based classification has been employed with some success, but even low overall false positive rates can interfere with the ability to separate critical species and provide adequate estimates of bloom trajectories through time.

We highlight the relative strengths of quantification with deep features with several contrasting plankton taxa in the MVCO records, fully analyzed to reflect absolute concentration in the environment (count estimate scaled to seawater volume) and known sample date and time (Figure 6). The chain-forming diatom species *Guinardia delicatula* commonly dominates the phytoplankton biomass at MVCO, with large wintertime blooms occurring in many years. While traditional features and a random forest classification approach have previously been used to study bloom dynamics in this species (Peacock et al., 2014), quantification from AC coupled with RN-101 provides fewer cases of incorrectly predicted small peaks during non-bloom periods. For the less abundant diatom, *Ditylum brightwellii* the improvement is even more evident, with AC-RN101 estimates almost entirely removing the false bloom events and overestimates that plague quantification with CC and traditional feature-based classification. Appropriately interpreted random forest classification has also been useful for studying temporal dynamics in the ciliated micrograzer *Laboea strobila* (Brownlee et al., 2016) but, as for the low concentration diatom, quantification with deep features provides striking fidelity even during period of very low concentration ($\ll 1ml^{-1}$). Notably, quantification with deep features also works extremely well for some challenging cases, such as heterogeneous groupings of small ciliated protozoan taxa with a range of morphologies and relatively small cell types including the nanoflagellate *Pyraminomnas longicauda*, that have proven difficult to distinguish reliably from other nanoplankton on the basis of traditional features.

New CNN architectures are emerging rapidly, with innovations that are expected to lead to even better results going forward(Huang et al., 2016). The availability of these models pre-trained with a dataset such as ImageNet makes it relatively easy for researchers from different domains to take advantage of transfer learning and apply these models to their problems. Even a low-end computer, equipped with an inexpensive GPU, would be able to compute deep features for an automatic plankton system in real time. For instance, with a GPU GTX 1080, and a 100-layer resnet, deep features for all the images in an IFCB sample (we have taken 3500 images as the average sample size), can be computed in less than five minutes.

The methods described in this work are fully applicable to other plankton capture systems capable of obtaining and processing water samples containing plankton data. Parameters such as the period between samples, the number of ROIs in each sample or the plankton classes to be identified, are not determining as long as the system is trained and validated with a sufficiently high number of samples. This number

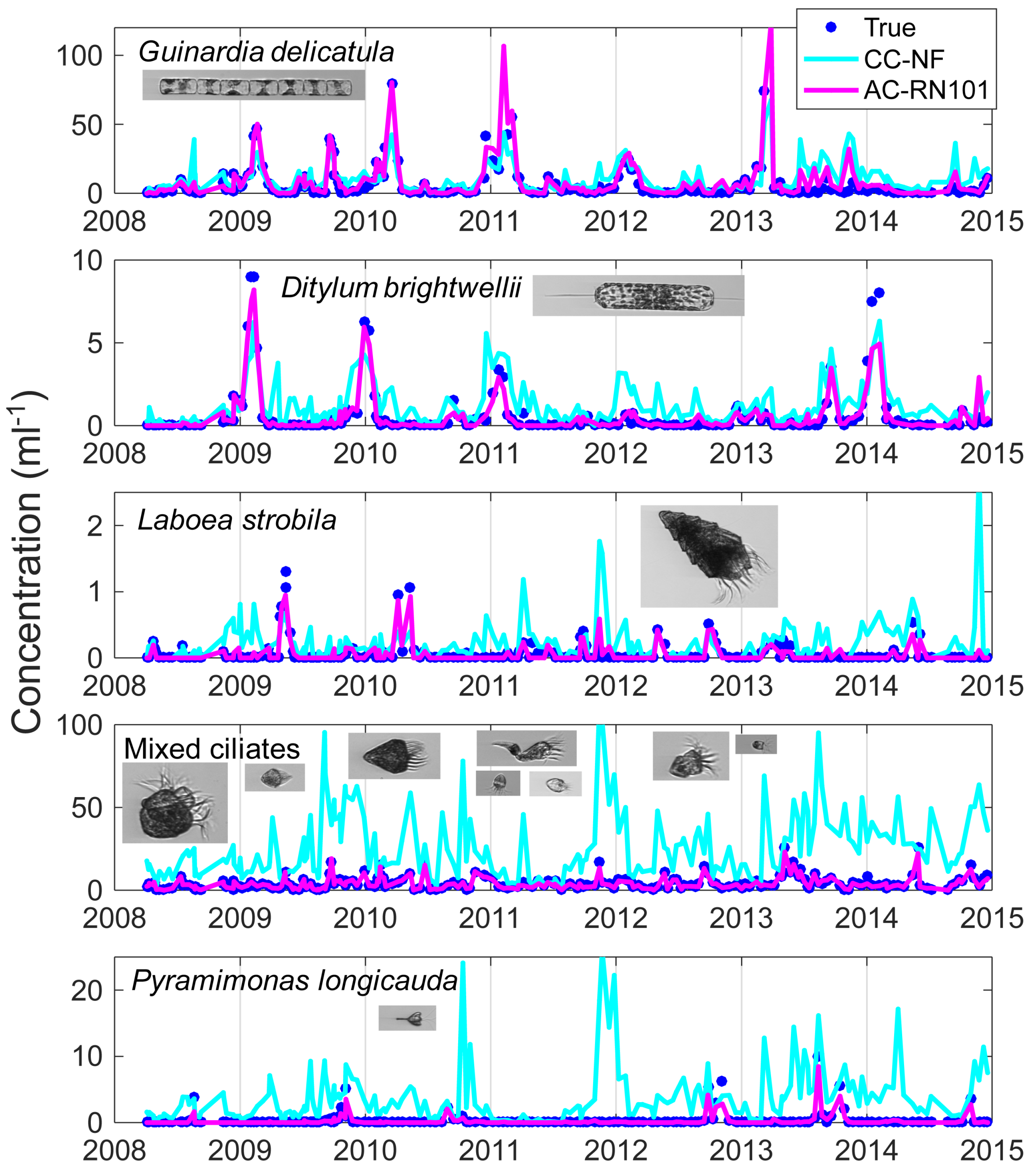


Figure 6: Daily resolved estimates of plankton concentration in the ocean at MVCO for several taxa that exhibit a range of concentrations and patterns of temporal variability during a 7-year period after collection of the images used for classifier training. True concentration (manually verified by experts) is shown along with estimates contrasting the simple CC method with traditional features (CC-NF) with the AC method with fully trained 101-layer network (AC-RN101). *Guinardia delicatula* and *Ditylum brightwellii* are diatoms. *Laboea strobila* is a distinctive species of ciliated protozoa, while ”Mixed ciliates” corresponds to a heterogeneous grouping of smaller-sized ciliates of unknown identity. *Pyramimonas longicauda* is a small ($< 10\mu m$) flagellate.

is a parameter that should be analyzed carefully while validating the system and that will depend on the complexity of the data to which the system is exposed.

In this work, quantification algorithms have been tested against the traditional Classify and Count (CC) method. Our results show that the improvement of quantification methods as AC, PAC or HDy over CC is typically small. Nonetheless, results also indicate that these quantification methods make the biggest difference when the underlying classifier performs less well, as the adjustments made are bigger and have a greater impact on the final result.

The set of experiments conducted in this work were carried out following a very thorough method (González et al., 2017a). Quantification algorithms were tested in the most similar way to actual working conditions. Also, the error measures used, are appropriate for abundance estimation problems, and allow us to detect potential problems in the built system. It is very interesting to note that all numerical and graphical data generated during the experiments are available online, favouring the detailed analysis of the system performance and its refinement.

Finally, it is important to highlight the importance of the effort of making public and available for download a dataset as the WHOI-Plankton dataset. Researchers from the Woods Hole Oceanographic Institution have made public not only the annotated data used in this paper, but also all data captured by IFCB since 2006. On the one hand, the use of publicly available dataset is important to guarantee experimental reproducibility in studies such as the one described in this paper. On the other hand, the fact of having all this raw data accessible will enable future exploration of different approaches such as autoencoders (Hinton and Salakhutdinov, 2006). The idea behind autoencoders is to build a neural network where the input and output layers are fed with the pixel values of the images. Thus, the network learns how to reconstruct each image in the dataset and the activations in the internal layers can be considered as a compressed representation of the image. Future work with the full IFCB image dataset will make it possible to assess the utility of this unsupervised method that can exploit the huge amount of unlabelled data available.

Acknowledgments

Our thanks to Angel López-Urrutia for his comments and insights during this research. Contributions from HMS and EEP were supported in part by the Simons Foundation, by the National Oceanic and Atmospheric Administration (NOAA) through the Cooperative Institute for the North Atlantic Region (CINAR) under Cooperative Agreement NA14OAR4320158, and by the National Science Foundation (NSF; Grants CCF-1539256, OCE-1655686).

References

Barranquero, J., Díez, J., and del Coz, J. J. (2015). Quantification-oriented learning based on reliable classifiers. *Pattern Recognition*, 48(2):591–604.

Barranquero, J., González, P., Díez, J., and del Coz, J. J. (2013). On the study of nearest neighbour algorithms for prevalence estimation in binary problems. *Pattern Recognition*, 46(2):472—482.

Beijbom, O., Hoffman, J., Yao, E., Darrell, T., Rodriguez-Ramirez, A., Gonzalez-Rivero, M., and Guldberg, O. H. (2015). Quantification in-the-wild: data-sets and baselines. *arXiv preprint arXiv:1510.04811*.

Bella, A., Ferri, C., Hernández-Orallo, J., and Ramirez-Quintana, M. J. (2010). Quantification via probability estimators. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 737–742. IEEE.

Benfield, M. C., Grosjean, P., Culverhouse, P. F., Irigoien, X., Sieracki, M. E., Lopez-Urrutia, A., Dam, H. G., Hu, Q., Davis, C. S., and Hansen, A. (2007). RAPID: research on automated plankton identification. *Oceanography*, 20(2):172–187.

Bochinski, E., Bacha, G., Eiselein, V., Walles, T. J., Nejstgaard, J. C., and Sikora, T. (2018). Deep active learning for in situ plankton classification. In *International Conference on Pattern Recognition*, pages 5–15. Springer.

Brownlee, E. F., Olson, R. J., and Sosik, H. M. (2016). Microzooplankton community structure investigated with imaging flow cytometry and automated live-cell staining. *Marine Ecology Progress Series*, 550:65–81.

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. (2015). Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

Cui, J., Wei, B., Wang, C., Yu, Z., Zheng, H., Zheng, B., and Yang, H. (2018). Texture and shape information fusion of convolutional neural network for plankton image classification. In *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, pages 1–5. IEEE.

Dai, J., Wang, R., Zheng, H., Ji, G., and Qiao, X. (2016a). Zooplanktonet: Deep convolutional network for zooplankton classification. In *OCEANS 2016-Shanghai*, pages 1–6. IEEE.

Dai, J., Yu, Z., Zheng, H., Zheng, B., and Wang, N. (2016b). A Hybrid Convolutional Neural Network for Plankton Classification. In *Computer Vision – ACCV 2016 Workshops*, Lecture Notes in Computer Science, pages 102–114. Springer, Cham.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Dunker, S., Boho, D., Wäldchen, J., and Mäder, P. (2018). Combining high-throughput imaging flow cytometry and deep learning for efficient species and life-cycle stage identification of phytoplankton. *BMC ecology*, 18(1):51.

Firat, A. (2016). Unified framework for quantification. *arXiv preprint arXiv:1606.00868*.

Forman, G. (2008). Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17(2):164–206.

González, P., Álvarez, E., Díez, J., López-Urrutia, Á., and del Coz, J. J. (2017a). Validation methods for plankton image classification systems. *Limnology and Oceanography: Methods*, 15(3):221–237.

González, P., Castaño, A., Chawla, N. V., and del Coz, J. J. (2017b). A Review on Quantification Learning. *ACM Computing Surveys (CSUR)*, 50(5):74.

González, P., Díez, J., Chawla, N., and del Coz, J. J. (2017c). Why is quantification an interesting learning problem? *Progress in Artificial Intelligence*, 6(1):53–58.

González-Castro, V., Alaiz-Rodríguez, R., and Alegre, E. (2013). Class distribution estimation based on the Hellinger distance. *Information Sciences*, 218:146–164.

Haralick, R. M. and Shanmugam, K. (1973). Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hu, M.-K. (1962). Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187.

Huang, G., Liu, Z., and Weinberger, K. Q. (2016). Densely connected convolutional networks. *CoRR*, abs/1608.06993.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kuhl, F. P. and Giardina, C. R. (1982). Elliptic Fourier features of a closed contour. *Computer graphics and image processing*, 18(3):236–258.

Lee, H., Park, M., and Kim, J. (2016). Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3713–3717.

Lloret, L., Heredia, I., Aguilar, F., Debusschere, E., Deneudt, K., and Hernandez, F. (2018). Convolutional neural networks for phytoplankton identification and classification. *Biodiversity Information Science and Standards*, 2:e25762.

Lumini, A. and Nanni, L. (2019). Ocean ecosystems plankton classification. In *Recent Advances in Computer Vision*, pages 261–280. Springer.

Luo, J. Y., Irisson, J.-O., Graham, B., Guigand, C., Sarafraz, A., Mader, C., and Cowen, R. K. (2018). Automated plankton image analysis using convolutional neural networks. *Limnology and Oceanography: Methods*, 16(12):814–827.

Moniruzzaman, M., Islam, S. M. S., Bennamoun, M., and Lavery, P. (2017). Deep learning on underwater marine object detection: A survey. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 150–160. Springer.

Narasimhan, H., Li, S., Kar, P., Chawla, S., and Sebastiani, F. (2016). Stochastic optimization techniques for quantification performance measures. *stat*, 1050:13.

Olson, R. J. and Sosik, H. M. (2007). A submersible imaging-in-flow instrument to analyze nano-and microplankton: Imaging FlowCytobot. *Limnology and Oceanography: Methods*, 5(6):195–203.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724.

Orenstein, E. C. and Beijbom, O. (2017). Transfer Learning and Deep Feature Extraction for Planktonic Image Data Sets. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1082–1088.

Orenstein, E. C., Beijbom, O., Peacock, E. E., and Sosik, H. M. (2015). Whoi-plankton-a large scale fine grained visual recognition benchmark dataset for plankton classification. *arXiv preprint arXiv:1510.00745*.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Peacock, E. E., Olson, R. J., and Sosik, H. M. (2014). Parasitic infection of the diatom *Guinardia delicatula*, a recurrent and ecologically important phenomenon on the new england shelf. *Marine Ecology Progress Series*, 503:1–10.

Pérez-Gállego, P., Castaño, A., Quevedo, J. R., and del Coz, J. J. (2019). Dynamic ensemble selection for quantification tasks. *Information Fusion*, 45:1 – 15.

Pérez-Gállego, P., Quevedo, J. R., and del Coz, J. J. (2017). Using ensembles for problems with characterizable changes in data distribution: A case study on quantification. *Information Fusion*, 34:87–100.

Py, O., Hong, H., and Zhongzhi, S. (2016). Plankton classification with deep convolutional neural networks. In *Information Technology, Networking, Electronic and Automation Control Conference, IEEE*, pages 132–136. IEEE.

Saerens, M., Latinne, P., and Decaestecker, C. (2002). Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1):21–41.

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

Sosik, H. M. (2017). Ifcb-analysis wiki. `https://github.com/hsosik/ifcb-analysis/wiki`.

Sosik, H. M., Futrelle, J., Brownlee, E. F., Peacock, E., Crockford, T., and Olson, R. J. (2016). hsosik/ifcb-analysis: Ifcb-analysis software system, initial formal release at v2 feature stage.

Sosik, H. M. and Olson, R. J. (2007). Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. *Limnology and Oceanography: Methods*, 5(6):204–216.

Sosik, H. M., Peacock, E. E., and Brownlee, E. F. (2015). Annotated plankton images data set for developing and evaluating classification methods.

Wang, C., Zheng, X., Guo, C., Yu, Z., Yu, J., Zheng, H., and Zheng, B. (2018). Transferred parallel convolutional neural network for large imbalanced plankton database classification. In *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, pages 1–5. IEEE.

Appendix A. Plankton Dataset Labels

Short quantifier label	**WHOI-Plankton dataset labels**	**Taxonomic range in class, other description**
Asterionellopsis	*Asterionellopsis*	*Asterionellopsis spp.*
Cerataulina	*Cerataulina pelagica*	*Cerataulina pelagica*
Ceratium	*Ceratium*	*Ceratium spp.*
Chaetoceros	*Chaetoceros, Chaetoceros didymus*	*Chaetoceros spp., Chaetoceros didymus*
Corethron	*Corethron hystrix*	*Corethron hystrix*
Coscinodiscus	*Coscinodiscus*	*Coscinodiscus spp.*
Cylindrotheca	*Cylindrotheca*	*Cylindrotheca spp.*
DactFragCerataul	*Dactyliosolen fragilissimus*	*Dactyliosolen fragilissimus*
Dactyliosolen	*Dactyliosolen blavyanus*	*Dactyliosolen blavyanus*
Dictyocha	*Dictyocha*	*Dictyocha spp.*
Dinobryon	*Dinobryon*	*Dinobryon spp.*
Dinophysis	*Dinophysis*	*Dinophysis spp.*
Ditylum	*Ditylum brightwellii*	*Ditylum brightwellii*
Ephemera	*Ephemera*	*Ephemera spp.*
Eucampia	*Eucampia*	*Eucampia spp.*
Euglena	Euglenia (subclass)	Euglenia (subclass)
Guinardia	*Guinardia delicatula*	*Guinardia delicatula*
Guinardia_flaccida	*Guinardia flaccida*	*Guinardia flaccida*
Guinardia_striata	*Guinardia striata*	*Guinardia striata*
Gyrodinium	*Gyrodinium, Amphidinium, Katodinium, Torodinium, Proterythropsis*	*Gyrodinium spp., Amphidinium spp., Katodinium spp., Torodinium spp., Proterythropsis spp.*
Laboea	*Laboea strobila*	*Laboea strobila*
Lauderia	*Lauderia*	*Lauderia spp.*
Leptocylindrus	*Leptocylindrus*	*Leptocylindrus spp.*
Licmophora	*Licmophora*	*Licmophora spp.*
Myrionecta	*Mesodinium_sp*	*Mesodinium spp.*
Odontella	*Odontella*	*Odontella spp.*
Paralia	*Paralia*	*Paralia spp.*

Phaeocystis	*Phaeocystis, Parvicorbicula socialis*	*Phaeocystis globosa, Parvicorbicula socialis*
Pleurosigma	*Pleurosigma*	*Pleurosigma spp.*
Prorocentrum	*Prorocentrum*	*Prorocentrum spp.*
Pseudonitzschia	*Pseudonitzschia*	*Pseudonitzschia spp.*
Pyramimonas	*Pyramimonas longicauda*	*Pyramimonas longicauda*
Rhizosolenia	*Rhizosolenia*	*Rhizosolenia spp.*
Skeletonema	*Skeletonema*	*Skeletonema spp.*
Stephanopyxis	*Stephanopyxis*	*Stephanopyxis spp.*
Thalassionema	*Thalassionema*	*Thalassionema spp.*
Thalassiosira	*Thalassiosira*	*Thalassiosira spp.*, other similar centric diatom species
Thalassiosira_dirty	*Thalassiosira* with external detritus	Thalassiosira spp.with external detritus
bad	contains only camera field background	contains only camera field background
ciliate_mix	*Didinium*, *Euplotes*, *Leegaardiella ovalis*, *Pleuronema*, *Strobilidium*, *Tiarnia*, *Tontonia*, and unidentified ciliates	*Didinium spp.*, *Euplotes spp.*, *Leegaardiella ovalis*, *Pleuronema spp.*, *Strobilidium spp.*, *Tiarnia spp.*, *Tontonia spp.*, and unidentified ciliates
clusterflagellate	*Corymbellus*	*Corymbellus spp.*
detritus	detritus	detritus
dino30	ameoba, *Akashiwo*, *Hetercapsa triquetra*, *Karenia*, *Protoperidinium*, *Vicicitus globosus*, unidentified dinoflagellates	*Akashiwo spp.*, *Hetercapsa triquetra*, *Karenia spp.*, *Protoperidinium spp.*, *Vicicitus globosus*, unidentified dinoflagellates and amoeba
kiteflagellates	*Chrysochromulina lanceolata*	*Chrysochromulina lanceolata*
mix	*Cryptophyta*, *Pyramimonas*, *Chrysochromulina*, *Heterocapsa rotundata*, unidentified nanoplankton	*Cryptophyta, Pyramimonas spp., Chrysochromulina spp., Heterocapsa rotundata, unidentified nanoplankton*
mix_elongated	miscellaneous diatom fragments	miscellaneous diatom fragments

na	dino10, other, diatome_flagellate, other_interaction, *Leptocylindrus mediterraneus*, pennates on diatoms, *Delphineis*, *Bacillaria*, *Bidulphia*, *Cochlodinium*, *Emiliania huxleyi*, *Pseudochattonella farcimen*, bead, bubble, pollen, spore, zooplankton	Other rare and/or unidentified taxa
pennate	miscellaneous pennate diatoms	miscellaneous pennate diatoms
tintinnid	Tintinnida	Tintinnida

Appendix B. Mean absolute error by class

	NF		RN-101	
Class	**CC** (10^{-4})	**AC** (10^{-4})	**CC** (10^{-4})	**AC** (10^{-4})
Asterionellopsis	56.97 / 1.60	43.16 / 2.50	2.64 / 0.25	**2.61** / 0.26
bad	148.35 / 15.04	162.01 / 18.44	**40.43** / 5.16	40.78 / 5.31
Cerataulina	89.25 / 4.39	61.97 / 4.61	28.44 / 2.87	**27.52** / 2.79
Ceratium	0.97 / 0.07	0.95 / 0.07	0.32 / 0.05	**0.31** / 0.05
Chaetoceros	334.28 / 6.17	136.26 / 5.47	42.49 / 1.86	**38.81** / 1.97
ciliate_mix	148.76 / 3.96	71.13 / 4.29	6.56 / 0.40	**6.47** / 0.36
clusterflagellate	80.33 / 3.14	67.71 / 5.56	0.61 / 0.09	**0.57** / 0.09
Corethron	75.28 / 2.32	39.16 / 3.01	2.04 / 0.21	**1.89** / 0.19
Coscinodiscus	2.42 / 0.17	2.61 / 0.29	**0.82** / 0.11	0.87 / 0.12
Cylindrotheca	58.14 / 2.82	63.26 / 3.00	23.18 / 1.98	**22.50** / 1.93
DactFragCerataul	192.36 / 4.97	75.18 / 5.29	20.19 / 1.06	**14.69** / 1.04
Dactyliosolen	72.64 / 2.14	60.30 / 3.11	**9.49** / 0.85	9.61 / 0.87
detritus	555.91 / 20.27	652.58 / 24.07	409.26 / 23.09	**387.54** / 23.79
Dictyocha	20.72 / 1.02	14.98 / 1.22	1.13 / 0.19	**1.13** / 0.20
dino30	148.90 / 4.82	128.98 / 4.78	138.26 / 5.05	**121.86** / 5.20
Dinobryon	27.72 / 0.92	18.52 / 1.04	3.54 / 0.33	**3.30** / 0.32
Dinophysis	18.10 / 0.83	15.44 / 1.49	**1.59** / 0.16	1.66 / 0.17
Ditylum	7.91 / 0.43	6.11 / 0.52	1.86 / 0.22	**1.67** / 0.20
Ephemera	3.99 / 0.20	2.29 / 0.23	**0.57** / 0.11	0.58 / 0.11
Eucampia	29.93 / 1.02	25.28 / 1.66	**2.01** / 0.26	2.09 / 0.27
Euglena	17.63 / 0.60	4.60 / 0.51	4.08 / 0.37	**3.91** / 0.37
Guinardia	62.17 / 4.42	50.89 / 3.97	20.88 / 1.27	**20.48** / 1.28
Guinardia_flaccida	4.56 / 0.23	3.49 / 0.30	**0.72** / 0.08	0.72 / 0.08
Guinardia_striata	12.56 / 1.28	8.64 / 1.22	4.25 / 1.24	**4.18** / 1.24
Gyrodinium	33.73 / 1.60	25.01 / 2.14	4.89 / 0.34	**4.81** / 0.34
kiteflagellates	14.35 / 0.79	16.26 / 1.13	**0.75** / 0.21	0.75 / 0.22
Laboea	2.02 / 0.15	1.85 / 0.17	0.50 / 0.09	**0.47** / 0.08
Lauderia	2.22 / 0.30	7.61 / 1.14	**0.19** / 0.04	0.22 / 0.05
Leptocylindrus	217.77 / 16.64	184.05 / 12.26	91.86 / 7.48	**87.28** / 7.17
Licmophora	3.61 / 0.16	2.45 / 0.22	0.42 / 0.06	**0.42** / 0.06
mix	3135.64 / 29.01	1296.29 / 39.10	583.80 / 23.05	**459.72** / 22.77
mix_elongated	305.75 / 7.28	177.42 / 9.49	119.55 / 6.54	**111.25** / 6.81
Myrionecta	65.59 / 1.92	25.91 / 2.15	**2.33** / 0.17	2.38 / 0.16
na	484.27 / 7.93	125.58 / 9.29	19.33 / 0.88	**18.90** / 0.90
Odontella	0.68 / 0.06	1.13 / 0.14	**0.10** / 0.02	0.12 / 0.03
Paralia	22.52 / 0.93	17.57 / 1.49	**0.60** / 0.06	0.60 / 0.06
pennate	101.97 / 2.83	30.42 / 2.28	**8.05** / 0.79	8.35 / 0.79
Phaeocystis	13.62 / 0.58	20.35 / 1.28	1.98 / 0.32	**1.94** / 0.31
Pleurosigma	2.43 / 0.17	2.40 / 0.21	0.76 / 0.09	**0.75** / 0.09
Prorocentrum	15.38 / 0.58	10.92 / 0.66	**4.35** / 0.49	4.36 / 0.48
Pseudonitzschia	69.37 / 2.54	28.30 / 2.34	**11.66** / 0.74	11.80 / 0.75
Pyramimonas	24.49 / 0.82	15.42 / 1.14	1.12 / 0.27	**1.04** / 0.25
Rhizosolenia	82.21 / 3.95	61.50 / 3.55	20.99 / 1.41	**20.00** / 1.39
Skeletonema	272.94 / 9.43	229.31 / 13.01	**28.52** / 2.65	28.57 / 2.66
Stephanopyxis	0.38 / 0.04	0.51 / 0.09	**0.04** / 0.01	0.04 / 0.01
Thalassionema	10.25 / 0.37	5.55 / 0.37	**1.48** / 0.11	1.48 / 0.11
Thalassiosira	210.93 / 4.94	75.49 / 5.07	19.93 / 0.98	**19.32** / 0.99
Thalassiosira_dirty	48.70 / 2.29	25.04 / 2.61	6.13 / 0.83	**5.45** / 0.79
tintinnid	8.71 / 0.43	6.09 / 0.57	**1.75** / 0.17	1.77 / 0.17

Table B.1: Absolute Errors (AE) Mean / Standard Error by class over all test samples. Results for CC and AC methods using normal features (NF) and RN-101 features. Lowest errors per class shown in bold.

Capítulo 6

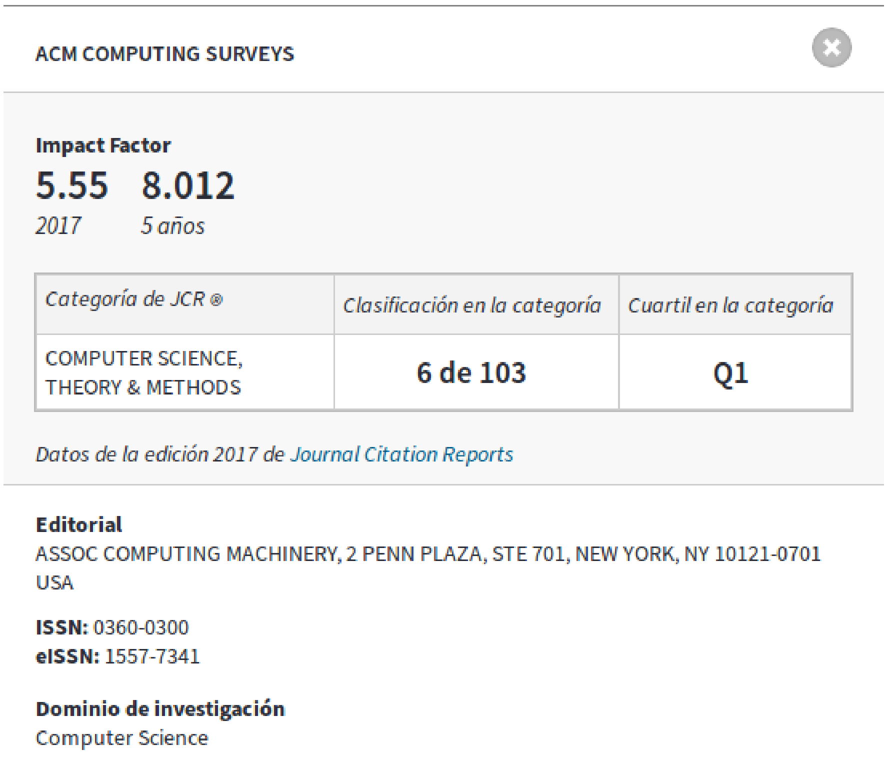
Análisis del factor de impacto

A continuación, se incluye la información referente al Factor de Impacto de las revistas en las que se han publicado los trabajos que componen esta tesis. Todas ellas se encuentran incluidas en JCR (Journal Citation Reports). De este modo, para obtener el Factor de Impacto de cada una de las mismas, se consideró la información aportada por la Web of Sciences. Concretamente, se empleó el año 2017 como referencia, al ser la última anualidad recogida hasta el momento.

1. Revista *ACM Computing Surveys*. Esta revista pertenece al primer cuartil en la categoría *Computer Science, Theory & Methods*. Su factor de impacto en 2017 es de 5.55 puntos (Figura 6.1)

2. Revista *Limmology and Oceanography: Methods*. Esta revista pertenece a dos categorías. En la categoría *Limnology* se encuentra en el segundo cuartil y en la categoría *Oceanography* pertence también al segundo cuartil. Su factor de impacto en 2017 es de 2.015 puntos (Figura 6.2)

3. Revista *Journal of Plankton Reearch*. Esta revista pertenece a las categorías *Oceanography* y *Marine & Freshwater Biology*, en las que se encuentra en el segundo cuartil. Su factor de impacto en 2017 es de 1.897 puntos (Figura 6.3)



ACM COMPUTING SURVEYS

Impact Factor

5.55 **8.012**

2017 *5 años*

Categoría de JCR ®	*Clasificación en la categoría*	*Cuartil en la categoría*
COMPUTER SCIENCE, THEORY & METHODS	**6 de 103**	**Q1**

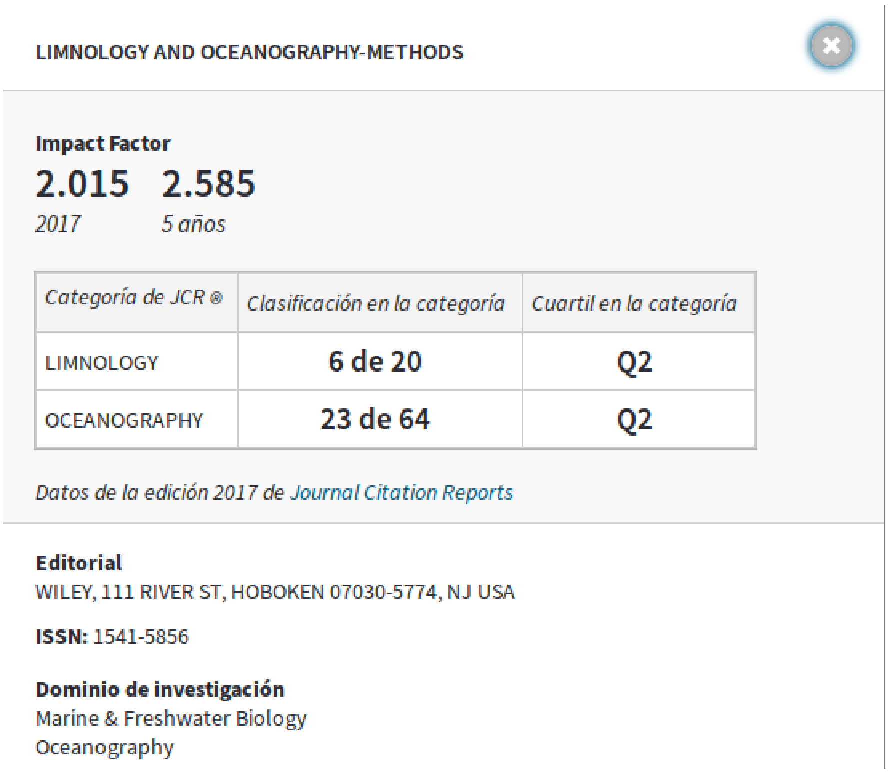
Datos de la edición 2017 de Journal Citation Reports

Editorial
ASSOC COMPUTING MACHINERY, 2 PENN PLAZA, STE 701, NEW YORK, NY 10121-0701 USA

ISSN: 0360-0300
eISSN: 1557-7341

Dominio de investigación
Computer Science

Figura 6.1: Factor de impacto de la revista ACM Computing Surveys



LIMNOLOGY AND OCEANOGRAPHY-METHODS

Impact Factor

2.015 **2.585**
2017 *5 años*

Categoría de JCR ®	*Clasificación en la categoría*	*Cuartil en la categoría*
LIMNOLOGY	6 de 20	Q2
OCEANOGRAPHY	23 de 64	Q2

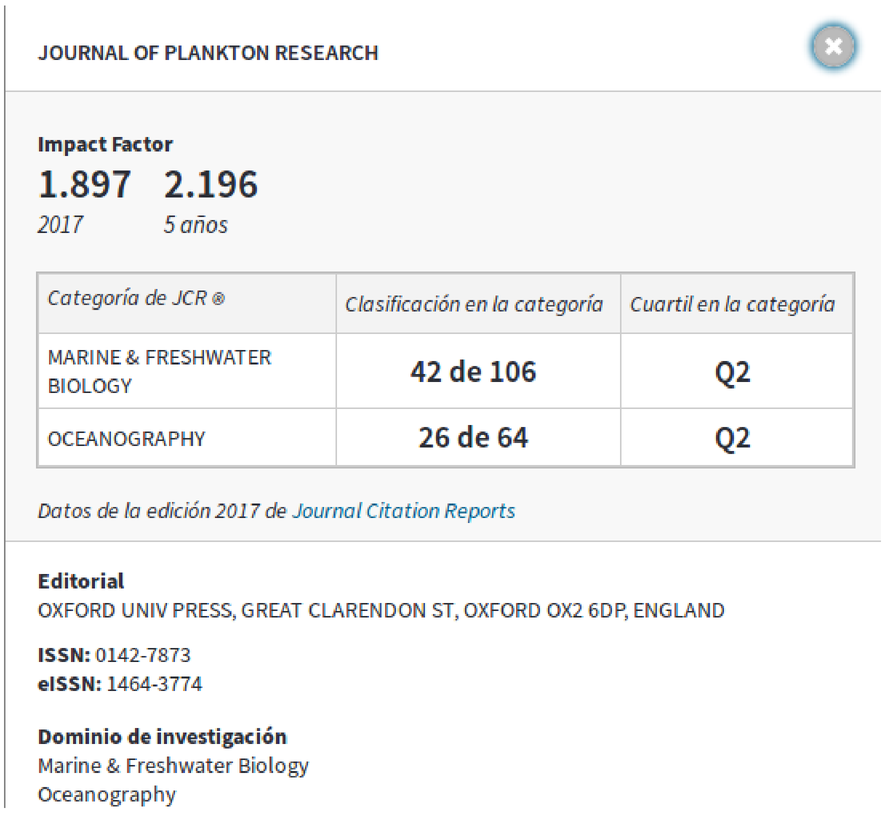
Datos de la edición 2017 de Journal Citation Reports

Editorial
WILEY, 111 RIVER ST, HOBOKEN 07030-5774, NJ USA

ISSN: 1541-5856

Dominio de investigación
Marine & Freshwater Biology
Oceanography

Figura 6.2: Factor de impacto de la revista Limmology and Oceanography: Methods



JOURNAL OF PLANKTON RESEARCH

Impact Factor

1.897	2.196
2017	5 años

Categoría de JCR ®	Clasificación en la categoría	Cuartil en la categoría
MARINE & FRESHWATER BIOLOGY	42 de 106	Q2
OCEANOGRAPHY	26 de 64	Q2

Datos de la edición 2017 de Journal Citation Reports

Editorial
OXFORD UNIV PRESS, GREAT CLARENDON ST, OXFORD OX2 6DP, ENGLAND

ISSN: 0142-7873
eISSN: 1464-3774

Dominio de investigación
Marine & Freshwater Biology
Oceanography

Figura 6.3: Factor de impacto de la revista Journal of Plankton Research

Bibliografía

Eva Álvarez, Ángel López-Urrutia, y Enrique Nogueira. Improvement of plankton biovolume estimates derived from image-based automatic sampling devices: application to FlowCAM. *J. Plankton Res.*, 34(6):454–469, 2012.

Jon Scott Armstrong. *Long-range Forecasting: From Crystal Ball to Computer.* Wiley: New York, 1978.

Oscar Beijbom, Judy Hoffman, Evan Yao, Trevor Darrell, Alberto Rodriguez-Ramirez, Manuel Gonzalez-Rivero, y Ove Hoegh Guldberg. Quantification in-the-wild: data-sets and baselines. *arXiv preprint arXiv:1510.04811*, 2015.

Antonio Bella, Cesar Ferri, José Hernández-Orallo, y Maria Jose Ramirez-Quintana. Quantification via probability estimators. En *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, págs. 737–742. IEEE, 2010.

Mark C. Benfield, Philippe Grosjean, Phil F. Culverhouse, Xabier Irigoien, Michael E. Sieracki, Angel Lopez-Urrutia, Hans G. Dam, Qiao Hu, Cabell S. Davis, y Allen Hansen. RAPID: research on automated plankton identification. *Oceanography*, 20(2):172–187, 2007.

J Roger Bray y John T Curtis. An ordination of the upland forest communities of southern wisconsin. *Ecol. Monogr.*, 27(4):325–349, 1957.

Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

Phil F. Culverhouse, Robert Williams, Beatriz Reguera, Vincent Herry, y Sonsoles González-Gil. Do experts make mistakes? A comparison of human and machine identification of dinoflagellates. *Mar. Ecol.: Prog. Ser.*, 247(17-25):5, 2003.

Jialun Dai, Ruchen Wang, Haiyong Zheng, Guangrong Ji, y Xiaoyan Qiao. ZooplanktoNet: Deep convolutional network for zooplankton classification. En *OCEANS 2016-Shanghai*, págs. 1–6. IEEE, 2016.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, y L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. En *CVPR09*. 2009.

Richard O Duda, Peter E Hart, y David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

George Forman. Counting positives accurately despite inaccurate classification. En *Proceedings of ECML'05*, págs. 564–575. 2005.

George Forman. Quantifying trends accurately despite classifier error and class imbalance. En *Proceedings of ACM SIGKDD'06*, págs. 157–166. ACM, 2006.

George Forman. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17(2):164–206, 2008. ISSN 1384-5810.

Pablo González, Eva Álvarez, Jose Barranquero, Jorge Díez, Rafael González-Quirós, Enrique Nogueira, Angel López-Urrutia, y Juan José del Coz. Multiclass support vector machines with example-dependent costs applied to plankton biomass estimation. *IEEE Transactions on Neural Networks and Learning Systems*, 24(11):1901–1905, 2013.

Víctor González-Castro, Rocío Alaiz-Rodríguez, y Enrique Alegre. Class distribution estimation based on the hellinger distance. *Information Sciences*, 218:146–164, 2013.

G. Gorsky, P. Guilbert, y E. Valenta. The Autonomous Image Analyzer– enumeration, measurement and identification of marine phytoplankton. *Marine ecology progress series. Oldendorf*, 58(1):133–142, 1989.

Gaby Gorsky, Mark D. Ohman, Marc Picheral, Stéphane Gasparini, Lars Stemmann, Jean-Baptiste Romagnan, Alison Cawood, Stéphane Pesant, Carmen García-Comas, y Franck Prejger. Digital zooplankton image analysis using the ZooScan integrated system. *J. Plankton Res.*, 32(3):285–303, 2010.

Robert M. Haralick, Karthikeyan Shanmugam, y Its' Hak Dinstein. Textural features for image classification. *IEEE Trans. Sys. Man and Cybern.*, 6:610–621, 1973.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, y Jian Sun. Deep residual learning for image recognition. En *Proceedings of the IEEE conference on computer vision and pattern recognition*, págs. 770–778. 2016.

Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.

Qiao Hu y Cabell Davis. Automatic plankton image recognition with co-occurrence matrices and support vector machine. *Mar. Ecol.: Prog. Ser.*, 295:21–31, 2005.

H. P. Jeffries, M. S. Berman, A. D. Poularikas, C. Katsinis, I. Melas, K. Sherman, y L. Bivins. Automated sizing, counting and identification of zooplankton by pattern recognition. *Marine Biology*, 78(3):329–334, 1984. ISSN 0025-3162, 1432-1793.

Alex Krizhevsky, Ilya Sutskever, y Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. En *Advances in neural information processing systems*, págs. 1097–1105. 2012.

Frank P. Kuhl y Charles R. Giardina. Elliptic Fourier features of a closed contour. *Computer graphics and image processing*, 18(3):236–258, 1982.

H. Lee, M. Park, y J. Kim. Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning. En *2016 IEEE International Conference on Image Processing (ICIP)*, págs. 3713–3717. 2016. doi:10.1109/ICIP.2016.7533053.

D.A Lisin, M.A Mattar, M.B. Blaschko, E.G. Learned-Miller, y M.C. Benfield. Combining Local and Global Image Features for Object Class Recognition. En *CVPR Workshops*, págs. 47–47. 2005.

Tong Luo, K. Kramer, D.B. Goldgof, L.O. Hall, S. Samson, A Remsen, y T. Hopkins. Recognizing plankton images from the shadow image particle profiling evaluation recorder. *IEEE Trans. Sys. Man and Cybern., Part B Cybern.*, 34(4):1753–1762, 2004.

Letizia Milli, Anna Monreale, Giulio Rossetti, Dino Pedreschi, Fosca Giannotti, y Fabrizio Sebastiani. Quantification in social networks. En *IEEE Int. Conf. on Data Science and Advanced Analytics,*, págs. 1–10. 2015.

Robert J. Olson y Heidi M. Sosik. A submersible imaging-in-flow instrument to analyze nano-and microplankton: Imaging FlowCytobot. *Limnology and Oceanography: Methods*, 5(6):195–203, 2007.

E. C. Orenstein y O. Beijbom. Transfer Learning and Deep Feature Extraction for Planktonic Image Data Sets. En *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, págs. 1082–1088. 2017. doi:10.1109/WACV.2017.125.

Sinno Jialin Pan y Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Christian K. Sieracki, Michael E. Sieracki, y Charles S. Yentsch. An imaging-in-flow system for automated analysis of marine microplankton. *Mar. Ecol.: Prog. Ser.*, 168:285–296, 1998.

Karen Simonyan y Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

R. Simpson, P. Culverhouse, R. Ellis, y B. Williams. Classification of euceratium gran. in neural networks. En *IEEE Conf. Neural Networks for Ocean Eng.*, págs. 223–229. 1991.

H M Sosik, E E Peacock, y E F Brownlee. Annotated plankton images data set for developing and evaluating classification methods. 2015. doi:10.1575/1912/7341. URL `http://hdl.handle.net/10.1575/1912/7341`.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, y Andrew Rabinovich. Going deeper with convolutions. En *Proceedings of the IEEE conference on computer vision and pattern recognition*, págs. 1–9. 2015.

Dirk Tasche. Exact fit of simple finite mixture models. *J. of Risk and Financial Management*, 7(4):150–164, 2014.

Vladimir Naumovich Vapnik y Vlamimir Vapnik. *Statistical learning theory*, tomo 1. Wiley New York, 1998.

Feng Zhao, Feng Lin, y Hock Soon Seah. Binary SIPPER plankton image classification using random subspace. *Neurocomputing*, 73(10–12):1853–1860, 2010. ISSN 0925-2312.