# IX Simposio de Teoría y Aplicaciones de la Minería de Datos (IX TAMIDA)

TAMIDA 6:

PROBLEMAS NO ESTÁNDAR

# JCLAL 2.0: mejoras y nuevas funcionalidades en la herramienta Java de código abierto para el aprendizaje activo

Eduardo Pérez
Instituto Maimónides de Investigación Biomédica de Córdoba
Email: eduardo.perez@imibic.org

Luis D. González
Dpto. de Informática
Universidad de Holguín
Email: ldgonzalezo@uho.edu.cu

Luis M. Sánchez
Dpto. de Informática
Universidad de Holguín
Email: lsanchezv@uho.edu.cu

Oscar Reyes
Dpto. Informática y Análisis Numérico
Universidad de Córdoba
Email: ogreyes@uco.es

Sebastián Ventura
Dpto. Informática y Análisis Numérico
Universidad de Córdoba
Email: sventura@uco.es

***Resumen*—El aprendizaje activo tiene como objetivo la construcción de mejores clasificadores mediante el etiquetado iterativo de conjuntos de ejemplos no etiquetados. Este paradigma de aprendizaje resulta de suma importancia en la actualidad, debido principalmente al alto coste que tiene el proceso de clasificación de grandes volúmenes de datos. JCLAL es una librería Java de código abierto para la investigación en el área de aprendizaje activo, que fue creada con el objetivo de facilitar el desarrollo de algoritmos y la ejecución de experimentos en esta área de estudio. En este trabajo, se presentan las mejoras y nuevas funcionalidades que incorpora JCLAL en su versión 2.0. Esta nueva versión tiene varias características que hacen de JCLAL una herramienta más potente para la investigación en el área del aprendizaje activo. Se introduce un entorno visual que facilita aún más el uso de JCLAL por usuarios no expertos en el dominio. Además, esta nueva versión aprovecha los beneficios de la computación paralela y distribuida, posibilitando que JCLAL sea utilizado eficientemente en entornos actuales donde se manejan grandes conjuntos de datos.**

## I. Introducción

El aprendizaje supervisado tiene como objetivo la construcción de modelos a partir de datos que han sido etiquetados previamente. Sin embargo, uno de los principales problemas de este paradigma de aprendizaje es que generalmente se requiere de una considerable cantidad de datos etiquetados para lograr construir modelos con una alta precisión. El etiquetado de datos no es una tarea sencilla de hacer, ya que requiere comúnmente un esfuerzo considerable por parte de los expertos que etiquetan los datos. En consecuencia, hoy en día se pueden encontrar muchas colecciones de datos que tienen un número reducido de ejemplos etiquetados en conjunto con una cantidad enorme de ejemplos que quedaron sin etiquetar [1]. En este tipo de escenario las técnicas supervisadas son insuficientes al no ser capaces de explotar la información útil que se encuentra implícita en los datos no etiquetados. Ante este tipo de problema, surge el aprendizaje semi-supervisado y el aprendizaje activo (AL), que tienen como principal objetivo la construcción de modelos precisos usando tanto los datos etiquetados como los no etiquetados.

Los métodos de AL emplean como principal hipótesis que si el algoritmo tiene la oportunidad de elegir los datos para el proceso de aprendizaje, entonces se podrán construir mejores modelos con un menor costo. Las técnicas de AL tratan de superar los obstáculos del etiquetado de datos mediante consultas donde se intenta descubrir aquellos ejemplos no etiquetados que pueden aportar información útil en la construcción de un modelo. Los métodos de AL son iterativos, donde en cada paso los ejemplos no etiquetados más significativos son presentados a un oráculo (por ejemplo, un experto), el cual los etiqueta y posteriormente estos ejemplos son incorporados al conjunto de entrenamiento a partir del cual se construye un nuevo modelo. De esta manera se intenta constuir un modelo más preciso usando la menor cantidad posible de información [2, 3].

En la actualidad, el rápido y adecuado desarrollo de un área de investigación está condicionado a contar con las herramientas computacionales que provean, al menos, las siguientes funcionalidades: (I) la disponibilidad de un API que permita reducir el tiempo en la implementación de nuevos algoritmos; (II) la facilidad de reproducción de experimentos; y (III) que tenga implementado un número considerable de métodos del estado del arte, de manera que le ahorre tiempo a los investigadores, así como disminuya la posibilidad de cometer errores en la implementación de dichos algoritmos. En este sentido, el AL ha estado bastante limitado, ya que las heramientas computacionales para la investigación en esta área son muy escasas, y la mayoría de las existentes no proveen de las funcionalidades mencionadas anteriormente. Entre las librerías que podemos encontrar en Internet destacan LibAct [4] desarollada en Python, y JCLAL [5] que está completamente desarrollada en el lenguaje Java.

Si comparamos estas dos librerías de clases, es de destacar que JCLAL contiene todos los elementos tratados en LibAct,

pero además incorpora un amplio conjunto de funcionalidades que permiten controlar todo el ciclo de AL a través de un correcto y flexible diseño de clases e interfaces. De esta manera, la implementación de nuevas estrategias de AL, así como la definición de nuevos escenarios de consulta, tipos de oráculos, condiciones de paradas, entre otras muchas funcionalidades, resulta un proceso bastante sencillo de hacer. Por otra parte, JCLAL cuenta con una detallada documentación y un conjunto amplio de ejemplos.

Como parte de la evolución a la cual está sujeto cualquier software, la librería JCLAL se ha seguido mejorando y desarrollando desde su publicación inicial. El objetivo del presente trabajo es presentar las nuevas características y funcionalidades que podemos encontrar en la nueva versión de este framework. Si se analiza la versión 1.0 de JCLAL, esta no disponía de una interfaz de usuario, obligando a los investigadores a usar el sistema de configuración de experimentos (basado en ficheros XML) o bien a codificar directamente los procedimientos usando su API. En este sentido, la nueva versión de JCLAL dispone de una interfaz de usuario que lo convierte en un producto más asequible y usable, lo que posibilita conocer sus potencialidades sin tener que entrar en el proceso de asimilación de su API. Por otra parte, la antigua versión de JCLAL está limitada a trabajar solamente con conjuntos de datos que pueden ser cargados en la memoria de estaciones de trabajo individuales, lo cual complicaba el uso de este framework en escenarios con enormes volúmenes de datos (problemas *Big Data*). En este trabajo, se explica además como la nueva versión de JCLAL supera esta última limitación gracias al desarrollo de un módulo que se integra con el conocido framework para procesamiento de datos masivos Apache Spark [6].

El resto de este trabajo se organiza de la manera siguiente. En la Sección II se detallan los cambios y nuevas funcionalidades en el API de JCLAL, así como se describe la nueva manera en la que se gestiona la modularidad del framework. En la sección III se introduce el entorno visual de JCLAL, detallando las funcionalidades del mismo. La integración de la librería con el framework Spark se explica en la Sección IV. Finalmente, en la Sección V se presentan las conclusiones del presente trabajo.

## II. Estructura de clases y nuevas funcionalidades en el API de JCLAL 2.0

Desde su primer lanzamiento JCLAL ha continuado mejorándose y desarrollándose como parte del proceso de actualización y mantenimiento de cualquier software. En su versión 2.0, JCLAL cuenta con una nueva estructura de clases y funcionalidades que lo convierten en un framework cada vez más cercano a los estándares que se esperan de este tipo de herramienta computacional.

Una de las nuevas funcionalidades es la posibilidad de aprovechar todas las capacidades de cómputo de las máquinas modernas. El AL es un proceso iterativo, donde en cada iteración comúnmente los pasos más costosos son la construcción del modelo, la evaluación del modelo y el cálculo de la importancia de cada instancia no etiquetada según la estrategia de consulta. JCLAL brinda la posibilidad de paralelizar el experimento en una estación de trabajo individual y también a través del framework Apache Spark. El investigador además tiene la posibilidad de elaborar sus propias estrategias de paralelización a través de la interfaz *IParallelContext*. Si el usuario desea hacer la paralelización de forma personalizada deberá registrar su clase en un archivo de configuración, como se muestra en el ejemplo siguiente:

```
<query-strategy type="...EntropySamplingQueryStrategy">
 <parallel-unlabeled type="...TestUnlabeledDataSinglePC"/>
 <wrapper-classifier type="...WekaClassifier">
  <parallel-test type="...WekaTestModelSinglePC"/>
  <parallel-build type="...WekaBuildClassifierSinglePC"/>
  <classifier type="weka.classifiers.functions.SMO"/>
 </wrapper-classifier>
</query-strategy>
```

Anteriormente, JCLAL estaba restringido al uso de conjuntos de datos que siguieran el formato establecido por el popular framework Weka [7]. En esta nueva versión, se modificó la jerarquía de clases e interfaces que están relacionadas con la manipulación de los conjuntos de datos. De esta manera, se logra una mayor flexibilidad e independencia en el trabajo con los conjuntos de datos, haciendo posible integrar a JCLAL con datos en formato de cualquier otro framework. Si un investigador quiere aplicar JCLAL en conjuntos de datos no soportados nativamente, solamente tendrá que implementar las interfaces *IInstance* e *IDataset*; además estas interfaces dan soporte a los tipos de instancias densas y dispersas.

Por otra parte, en JCLAL 2.0 se le ha dado mayor soporte a los clasificadores del framework MOA [8], lo cual permite utilizar algoritmos de aprendizaje incremental en el proceso de AL. De esta manera, el modelo no necesita ser entrenado desde cero en cada iteración de AL (a diferencia de la mayoría de los clasificadores de Weka), lo cual mejora significativamente el rendimiento en escenarios donde se disponen de grandes volúmenes de datos no etiquetados o con datos en *streamming*.

Por último, en esta nueva versión se ha añadido un paquete de clases que implementan varios test estadísticos no paramétricos. Dicho paquete está inspirado en el módulo de test estadísticos de la librería Keel [9]. Los test estadísticos son de suma importancia para el análisis de resultados experimentales, y JCLAL incluye varios test que permiten realizar comparaciones múltiples entre estrategias de AL.

### *II-A. Modularidad de JCLAL*

Una de las nuevas mejoras es la modularidad del framework, lo cual facilita su portabilidad y distribución. La idea surgió al observar que JCLAL crecía en tamaño (un tamaño considerable al incluir el paquete de integración con Spark), lo cual hacía que el árbol de dependencia fuera bien extenso y además afectaba la portabilidad y usabilidad de JCLAL en aquellos usuarios que solamente querían usar las características básicas del framework.

En la nueva versión se optó por un diseño modular inspirado en el diseño de Apache Spark y Spring [10], separando las características del framework en los distintos módulos que componen el sistema. De esta manera, el proyecto fue

dividido en 3 módulos: **jclal-core** que tiene las funcionalidades básicas de JCLAL, el módulo **jclal-gui** que incluye la nueva interfaz visual y **jclal-spark** que comprende el procesamiento distribuido con Spark.

La modularidad de JCLAL se gestionó mediante Apache Maven [11], el cual es una herramienta popular para la gestión de proyectos software, que facilita la administración de dependencias y un amplio conjunto de estándares para el proceso de desarrollo del software. Si el usuario decide emplear Maven como administrador de dependencias, el *groupid* para JCLAL sería **net.sf.jclal**. El siguiente ejemplo muestra cómo instalar el núcleo de JCLAL (**jclal-core**) en una aplicación; de esta manera Maven gestiona las dependencias necesarias para usar solamente el módulo **jclal-core**.

```
<dependencies>
  <dependency>
    <groupId>net.sf.jclal</groupId>
    <artifactId>jclal-core</artifactId>
   <version>2.0</version>
  </dependency>
</dependencies>
```

## III. Entorno visual para JCLAL

Es común que los framework de aprendizaje automático dispongan de una interfaz gráfica de usuario (GUI, por sus siglas en inglés) para facilitar la configuración, ejecución y análisis de resultados de los experimentos, reduciendo así el tiempo y trabajo necesario de los investigadores. Tal es el caso de Weka, VisualJCLEL [12], Eva2 [13], OTA [14] y HeuristicLab [15] que permiten a los investigadores sin un amplio conocimiento de su funcionamiento interno, o incluso sin habilidades de programación, utilizar estos framework de forma intuitiva.

VisualJCLAL es un módulo GUI que permite trabajar con el framework de forma intuitiva y flexible, acercando JCLAL a los usuarios no expertos en el dominio. Este entorno visual está inspirado en la arquitectura de Weka y VisualJCLEC. VisualJCLAL permite aprovechar la mayoría de las funcionalidades que brinda su framework base, y hasta el momento brinda la opción de utilizar estrategias de AL diseñadas para crear modelos predictivos sobre datos clasificados con una sola etiqueta (*single-label learning*) y datos clasificados con múltiples etiquetas (*multi-label learning*) [2].

VisualJCLAL brinda también la posibilidad de incluir escenarios de AL, algoritmos y estrategias de consulta mediante *plugins* creados por el usuario, gracias al uso de una arquitectura basada en componentes y al diseño flexible de clases e interfaces de JCLAL. Para ello el usuario utilizará la opción de agregar un nuevo *plugin* al sistema donde sus clases serán añadidas a los archivos de configuración en formato **.props** y serán mostrados en la GUI aquellos componentes que implementen la interfaz *IPlugin*. Estos parámetros se incluirán automáticamente dentro de las opciones de la interfaz gráfica usando la estructura modular y el generador de interfaces del entorno de usuario.

Por otra parte, este módulo aprovecha las posibilidades de JCLAL en su versión 2.0 para la ejecución de los algoritmos en forma paralela y distribuida (esta última utilizando el framework Apache Spark), además de otras funcionalidades que resultan muy útiles para la realización de experimentos de AL.

La Figura 1 muestra la pantalla inicial que aparece tras ejecutar VisualJCLAL. A partir de ella se tienen tres opciones principales:

- Crear nuevo experimento (*New Experiment*): permite crear una nueva configuración desde cero para la ejecución de un experimento.
- Cargar la configuración de un experimento (*Make from XML*): permite cargar una configuración de un experimento creado anteriormente y que se encuentra almacenado en un archivo XML.
- Ver resultados (*View results*): permite analizar de forma gráfica los resultados de los experimentos ejecutados en JCLAL.


Figura 1. Interfaz inicial.

A la interfaz de configuración de un experimento se puede acceder a través de la opción *New Experiment* o cargando una configuración existente mediante la opción *Make from XML*. Como se puede observar en la Figura 2, en las diferentes pestañas de la interfaz se pueden configurar todos los parámetros necesarios para la ejecución del experimento.

En cuanto a la opción *View results*, esta nos permite cargar los ficheros generados por JCLAL y analizar gráficamente los resultados de un experimento, como se muestra en la Figura 3. Mediante la selección de una medida de evaluación se puede observar el comportamiento de varias curvas de aprendizaje al mismo tiempo, permitiendo realizar una comparación visual entre diferentes estrategias de AL. Además, esta interfaz permite configurar y analizar las gráficas mediante el cambio de colores, ocultar o mostrar curvas, definir rangos de evaluación, cambiar las formas de las líneas y calcular el área bajo una curva de aprendizaje.

Otra opción importante que brinda VisualJCLAL es la posibilidad de crear un estudio experimental con múltiples configuraciones a partir de una configuración básica (la configuración base debe contar con los parámetros mínimos

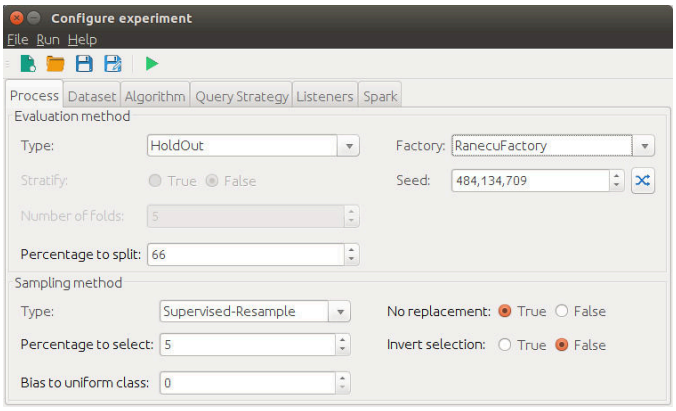



Figura 2. Interfaz para la configuración de un experimento.

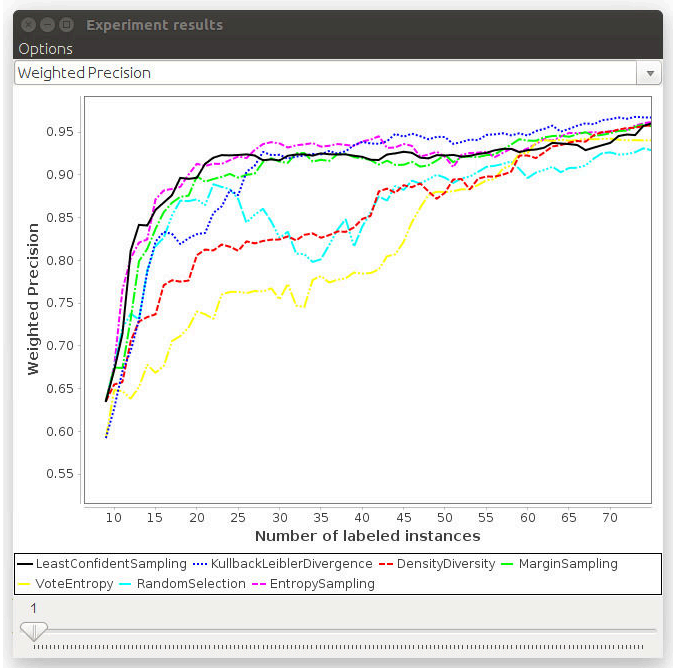



Figura 3. Interfaz de usuario para la visualización de resultados.

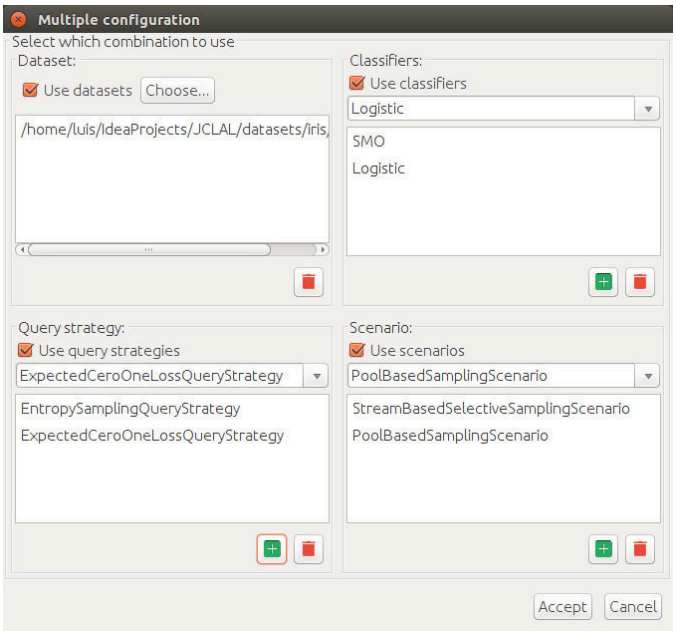



Figura 4. Creación de un estudio experimental.

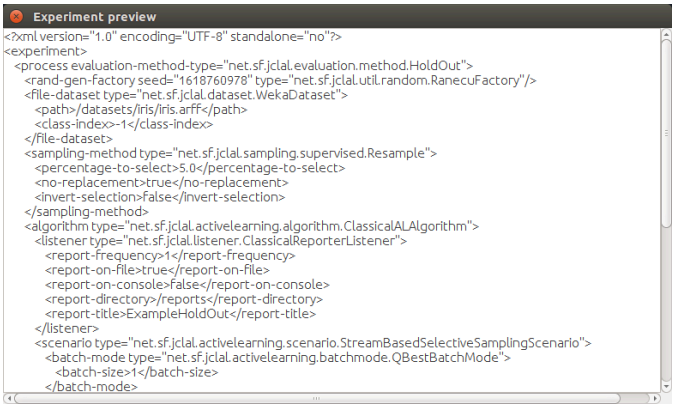



Figura 5. Vista previa del archivo de configuración.

necesarios para la ejecución de un experimento). En la Figura 4 se puede observar cómo se pueden generar $N$ configuraciones en dependencia de las opciones escogidas.

Por último, se cuenta con la opción de chequear cómo quedaría la configuración del experimento antes de ser almacenada en un fichero, como se muestra en la Figura 5.

## IV. Integración de JCLAL con el framework Spark

*Big Data* es un término muy popular en la actualidad, que hace referencia a problemas que involucran volúmenes de datos que superan la capacidad de los software clásicos y las máquinas convencionales para ser capturados, gestionados y procesados en un tiempo razonable. Apache Spark es uno de los framework más populares para el procesamiento de enormes conjuntos de datos, el cual implementa el paradigma de programación MapReduce [16] propuesto por Google. Spark almacena los resultados intermedios en memoria, logrando ejecutar aplicaciones mucho más rápido que las soluciones desarrolladas con Hadoop [17], lo cual hace de Spark una solución más adecuada para el desarrollo de algoritmos de aprendizaje automático altamente escalables. En este sentido, Spark cuenta con la librería MLlib [18] para el desarrollo de algoritmos distribuidos, la cual tiene implementaciones distribuidas de algunos de los algoritmos supervisados y no supervisados más populares, como Naive Bayes [19], árboles de decisión [20] y *k-means* [21].

La versión 1.0 de JCLAL se encontraba limitada a la hora de ejecutar experimentos con grandes conjuntos de datos, principalmente porque los algoritmos de aprendizaje que provee el framework Weka, o Mulan [22] para el caso de datos multi-etiqueta, no están diseñados para trabajar en este tipo escenarios. En la versión actual de JCLAL se ha desarrollado

un módulo que integra a JCLAL con Spark, aprovechando las ventajas que tiene este último para el procesamiento paralelo y distribuido de datos masivos. El módulo de AL distribuido (**jclal-spark**) está diseñado de forma tal que puede trabajar con los modelos de clasificación que provee la librería MLlib. La arquitectura es similar a la de **jclal-core**, siendo la estructura de paquetes **net.sf.jclal.spark.***.

Para utilizar **jclal-spark** a través de Maven hay que añadir el *artifactid* de este módulo, de esta manera se instalan automáticamente todas las dependencias necesarias para el trabajo con Spark. Además, en el fichero de configuración de un experimento, se debe incluir la etiqueta que indica la configuración para conectarse al servidor Spark. La etiqueta **master** especifica la dirección donde se encuentra el nodo maestro y en **app-name** se indica el nombre que se desea asignar al trabajo.

```
<spark>
  <master>spark://localhost:7077</master>
  <app-name>MyAlJob</app-name>
<spark>
```

Debido a que Spark provee sus propios modelos de clasificación, **jclal-spark** implementa las clases *SparkClassifier* y *SparkCommitteeClassifier* que sirven de interfaz para acceder a dichos clasificadores directamente (ver Figura 6). La clase encargada de realizar las evaluaciones de los clasificadores de Spark es *SparkSingleLabelEvaluation*, que además provee las medidas de tiempo de ejecución e información acerca de los conjuntos de datos.

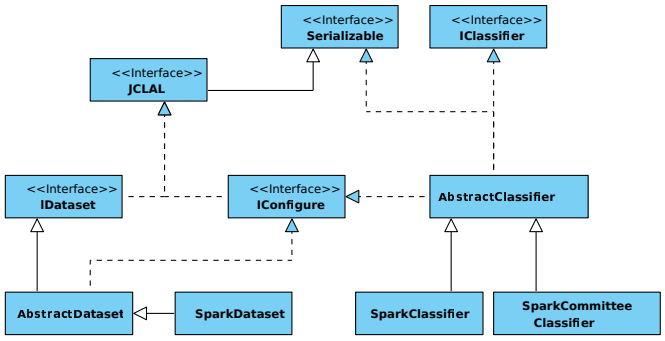



Figura 6. Diagrama de las clases principales del paquete **jclal-spark**.

Es válido destacar que el procesamiento de los conjuntos de datos en un ambiente distribuido es completamente diferente a como se hace en **jclal-core**. La clase *SparkDataset* se ha implementado para el manejo de datos distribuidos a través del conjunto de acciones y operadores que brinda la librería Spark. Los datos pueden ser cargados directamente desde disco o en un sistema de archivos distribuidos los cuales son primeramente indexados para luego ser procesados en el framework. En la nueva versión de JCLAL se añadió la etiqueta **format** para especificar el formato del conjunto de datos, logrando una mayor compatibilidad con Spark y otras librerías que se puedan incluir en el futuro. El ejemplo siguiente muestra como utilizar un conjunto de datos en formato **libsvm** empleando *SparkDataset*.

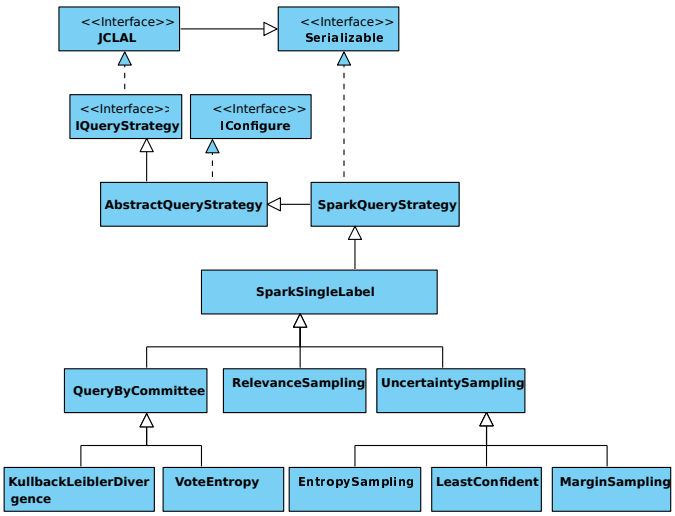



Figura 7. Diagrama de clases de las estrategias de consulta implementadas en el módulo **jclal-spark**.

```
<file-dataset type="net.sf.jclal.spark.dataset.SparkDataset">
  <path>path/to/dataset</path>
  <format>libsvm</format>
<file-dataset>
```

Respecto a las estrategias de consultas de AL implementadas en el paquete **jclal-spark**, hasta el momento están implementadas aquellas estrategias de consultas que se adaptan directamente al paradigma MapReduce, pero se espera implementar próximamente el resto de estrategias contenidas en **jclal-core**. La figura 7 muestra el diagrama de clases de las estrategias de consulta que actualmente soporta el módulo **jclal-spark**; estas son *Relevance Sampling*, y todas aquellas derivadas de *Uncertanty Sampling* (*Entropy Sampling*, *Least Confident* y *Margin Sampling*) y *Query By Committee* (*Kullback Leibler Divergence* y *Vote Entropy*).

Por último, en **jclal-spark** están implementados los métodos de evaluación clásicos: *Hold-Out*, *k-Fold Cross Validation*, *5X2 Cross Validation* y *Leave One Out Cross Validation*. En el caso de los métodos de validación cruzada se utilizaron funciones de Spark que están optimizadas para la creación de los *folds* en conjuntos de datos distribuidos.

## V. Conclusiones

En este trabajo se describieron las mejoras y nuevas funcionalidades que se incluyen en la nueva versión del framework JCLAL. En su nueva versión, el núcleo de JCLAL cuenta con una nueva estructura de clases y funcionalidades que lo convierten en un framework más potente para la investigación en el área de AL. Se logró una mayor flexibilidad e independencia en la manipulación de los conjuntos de datos, haciendo posible integrar a JCLAL con datos provenientes de cualquier otro framework. Además, se le dio un mayor soporte a algoritmos de aprendizaje incremental, lo cual posibilita una mayor eficiencia en el proceso de construcción de modelos predictivos. Por otra parte, se añadió un paquete que permite realizar contrastes de hipótesis mediantes test estadísticos no

paramétricos. Otra nueva mejora radica en la descomposición del framework en módulos independientes, lo cual facilita su portabilidad y distribución. De esta manera se puede trabajar con las funciones deseadas del framework sin tener instaladas todas y cada una de las librerías del árbol de dependencias.

JCLAL 2.0 también incorpora el entorno VisualJCLAL que permite trabajar con el framework de forma intuitiva y flexible, acercando JCLAL a los usuarios no expertos en el dominio y permitiendo un ahorro significativo de tiempo. Por último, se incorporó un nuevo e importante módulo, que permite de forma sencilla aplicar programación distribuida en JCLAL. El módulo de AL distribuido utiliza la popular librería Spark, lo cual permite el uso de JCLAL 2.0 en problemas *Big Data* y posibilita una mejor acogida por parte de la comunidad científica.

## Agradecimientos

Este trabajo ha sido financiado por el proyecto TIN2017-83445-P del Ministerio de Economía y Competitividad y Fondos FEDER. También ha sido financiado por el contrato i-PFIS no. IFI17/00015 otorgado por el Instituto de Salud Carlos III.

## Referencias

[1] O. Reyes, A. H. Altalhi, and S. Ventura, "Statistical comparisons of active learning strategies over multiple datasets," *Knowledge-Based Systems*, vol. 145, pp. 274–288, 2018.

[2] O. Reyes, C. Morell, and S. Ventura, "Effective active learning strategy for multi-label learning," *Neurocomputing*, vol. 273, pp. 494–508, 2018.

[3] O. Reyes and S. Ventura, "Evolutionary strategy to perform batch-mode active learning on multi-label data," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 4, pp. 46:1–46:26, 2018.

[4] Y. Y. Yang, S. C. Lee, Y. A. Chung, T. E. Wu, S. A. Chen, and H. T. Lin, "libact: Pool-based active learning in python," *arXiv preprint arXiv:1710.00379*, 2017.

[5] O. Reyes, E. Pérez, M. C. Rodríguez-Hernández, H. M. Fardoun, and S. Ventura, "JCLAL: a java framework for active learning," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3271–3275, 2016.

[6] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 10–10.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[8] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[9] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas *et al.*, "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

[10] R. Johnson, *Expert One-on-one J2EE Design and Development*. Wrox Press, 2002.

[11] F. P. Miller, A. F. Vandome, and J. McBrewster, "Apache maven," 2010.

[12] J. I. Jaen, J. R. Romero, and S. Ventura, "VisualJCLEC: A visual framework for evolutionary computation," in *Intelligent Systems Design and Applications (ISDA), 2012 12th International Conference on*. IEEE, 2012, pp. 119–125.

[13] M. Kronfeld, H. Planatscher, and A. Zell, "The EvA2 optimization framework," in *International Conference on Learning and Intelligent Optimization*. Springer, 2010, pp. 247–250.

[14] J. Brownlee *et al.*, "Oat: The optimization algorithm toolkit," *Centre for Information Technology Research (CITR), Swinburne University of Technology, Victoria, Australia, Technical Report A*, vol. 20071220, 2007.

[15] S. Wagner and M. Affenzeller, "Heuristiclab: A generic and extensible optimization environment," in *Adaptive and Natural Computing Algorithms*. Springer, 2005, pp. 538–541.

[16] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[17] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M. Franklin, S. Shenker, and I. Stoica, "Fast and interactive analytics over hadoop data with spark," *Usenix Login*, vol. 37, no. 4, pp. 45–51, 2012.

[18] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, "Mllib: Machine learning in apache spark," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016.

[19] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[20] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[21] S. Lloyd, "Least squares quantization in PCM," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[22] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning," *Journal of Machine Learning Research*, vol. 12, pp. 2411–2414, 2011.

# Resolviendo el problema de regresión multi-salida mediante *Gene Expression Programming*

Oscar Reyes
Dpto. Informática y Análisis Numérico
Universidad de Córdoba
Email: ogreyes@uco.es

Jose M. Moyano
Dpto. Informática y Análisis Numérico
Universidad de Córdoba
Email: jmoyano@uco.es

Jose M. Luna
Dpto. Informática y Análisis Numérico
Universidad de Córdoba
Email: jmluna@uco.es

Sebastián Ventura
Dpto. Informática y Análisis Numérico
Universidad de Córdoba
Email: sventura@uco.es

***Resumen*—En los últimos años, el estudio de problemas donde los ejemplos están asociados a múltiples variables objetivo al mismo tiempo ha ganado un creciente interés en la comunidad científica. En este trabajo se propone un método basado en *Gene Expression Programming* para darle solución al problema de regresión multi-salida. El método propuesto se enfoca en la regresión simbólica, lo cual permite crear un modelo predictivo multi-salida sin tener conocimiento previo de las relaciones existentes en los datos. La codificación de un individuo de la población se realiza mediante un cromosoma de varios genes, donde cada gen codifica una expresión matemática que produce la salida para una variable objetivo, y así cada individuo de la población representa directamente una posible solución al problema. Los operadores usados en el proceso evolutivo permiten la constante creación de nuevo material genético, y algunos de ellos favorecen la detección de las dependencias existentes entre variables objetivo. El estudio experimental realizado en 8 conjuntos de datos muestra los beneficios y efectividad del método propuesto para resolver el problema de regresión multi-salida.**

## I. Introducción

En la última década, el estudio de problemas donde cada ejemplo del conjunto de datos está asociado a múltiples variables objetivo (variables de salida) ha tenido un creciente interés en la comunidad de aprendizaje automático, debido principalmente al elevado número de aplicaciones reales que pueden ser modeladas dentro de este paradigma. El objeto de estudio de la regresión multi-salida o *multi-target regression* (MTR), se enfoca en la predicción simultánea de múltiples variables objetivo continuas usando un único conjunto de variables descriptoras (variables de entrada) [1, 2]. MTR está presente en varios dominios de aplicación, tales como en el modelado ecológico [3], procesado de señales [4], y la eficiencia energética [5].

A día de hoy, se han propuesto un número considerable de métodos para resolver el problema MTR, los cuales se pueden categorizar en dos grupos: métodos de transformación de problemas y métodos de adaptación de algoritmos [1]. Los métodos de transformación descomponen un problema de regresión multi-salida en varios problemas de regresión simple (es decir, con una sola variable objetivo), y por último mediante un proceso de agregación se obtienen las predicciones finales. En este sentido, la mayoría de los métodos de transformación para MTR han sido inspirados en aproximaciones existentes para el aprendizaje multi-etiqueta [2, 6]; en este último paradigma cada ejemplo está asociado a múltiples variables objetivo binarias. Por otro lado, la categoría de adaptación de algoritmos incluye aquellos métodos que no descomponen el problema MTR en problemas de regresión simple, sino que tratan directamente los datos multi-salida. Dentro de esta categoría se han propuesto un amplio número de métodos, tales como métodos estadísticos [7], árboles de regresión [3], algoritmos basados en reglas [8], máquinas de vectores soporte [9] y métodos locales [10].

Entre los paradigmas existentes más populares para resolver problemas de regresión se encuentra la regresión simbólica, donde el objetivo es buscar una expresión matemática que se ajuste a un conjunto de datos dado. En este tipo de técnica no se proporciona ningún modelo en particular como punto de partida, sino que las expresiones se forman por medio de la codificación de bloques matemáticos, lo cual provee de una selección de características implícita en el proceso de construcción de expresiones, y además puede beneficiar la detección de relaciones complejas en los datos. Las técnicas de regresión simbólica han sido aplicadas a un gran número de problemas reales [11], destacando el uso de algoritmos evolutivos para su resolución [12]. Entre las técnicas de computación evolutiva más adecuadas para el desarrollo de métodos de regresión simbólica se encuentra *Gene Expression Programming* (GEP) [12–14], una aproximación que se sitúa entre los algoritmos genéticos y la programación genética, aprovechando así las ventajas de ambas técnicas. [13]. GEP emplea una población de individuos, selecciona padres de acuerdo a su *fitness* y evoluciona la población utilizando operadores genéticos, tal y como hacen los algoritmos genéticos y la programación genética. Sin embargo, la principal diferencia radica en la forma en que los individuos son codificados. En este caso, los individuos se codifican como cadenas lineales de longitud fija (como en algoritmos genéticos), que luego se expresan como árboles de diferentes tamaños y formas (como en programación genética) [13].

Hasta el momento, los trabajos existentes que aplican GEP para regresión simbólica se han restringido a solucionar problemas de regresión simple. Es de destacar que al tratar de solucionar el problema MTR mediante regresión simbólica usando la técnica GEP aparecen dificultades adicionales que no existían en el contexto de la regresión simple, como es la dependencia estadística que puede existir entre las variables objetivo. En este sentido, varios estudios recientes han demostrado que es vital detectar y explotar correctamente dichas dependencias de cara a mejorar el rendimiento predictivo de los algoritmos de regresión [2, 9, 10, 15].

En este trabajo, se propone un método basado en GEP que resuelve el problema MTR mediante regresión simbólica. El método diseñado no divide el problema multi-salida en varias tareas de salida simple, sino que trata directamente con los datos multi-salida, por lo que se obtiene un método con un coste computacional aceptable en conjuntos de datos con un elevado número de variables objetivo. Los individuos se representan mediante un cromosoma con codificación multigénica, donde cada gen representa una función matemática que predice el valor de una variable objetivo; de esta manera cada individuo representa una solución completa al problema MTR. El poder creativo de GEP permite la constante creación de nuevo material genético, permitiendo una mejor exploración del espacio de búsqueda. Además, el método puede detectar las dependencias existentes entre variables objetivo por medio de los operadores de transposición o recombinación de genes entre cromosomas.

La contribución principal de este trabajo es la introducción de un método basado en GEP para el problema MTR. Este trabajo representa un estudio inicial, donde se pretende analizar los beneficios del paradigma GEP para construir modelos de regresión multi-salida. La efectividad del método desarrollado se comprobó mediante un estudio experimental en 8 conjuntos de datos, demostrando que se obtienen resultados prometedores mediante la comparación contra otros dos métodos del estado del arte.

El resto del este trabajo se organiza de la siguiente manera: en la Sección II se describen los fundamentos del método propuesto; la configuración del estudio experimental, así como la discusión de los resultados son presentados en la Sección III; finalmente, en la Sección IV se presentan las conclusiones del presente trabajo.

## II. Método Basado en GEP

En esta sección se describe el método propuesto basado en GEP para resolver el problema MTR mediante regresión simbólica. Antes de describir el método en sí, se definen algunas notaciones que son utilizadas a lo largo del trabajo.

Sea $S = \{(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2), \ldots, (\mathbf{x}^n, \mathbf{y}^n)\}$ un conjunto de datos de $n$ ejemplos de entrenamiento. Una instancia $i \in S$ se representa como una tupla $(\mathbf{x}^i, \mathbf{y}^i)$, donde $\mathbf{x}^i \in \mathcal{X}$ e $\mathbf{y}^i \in \mathcal{Y}$ son los vectores de entrada y salida de $i$, respectivamente. $\mathcal{X}$ representa el espacio de entrada que contiene $d$ variables descriptivas $x_1, x_2 \ldots, x_d$, mientras que $\mathcal{Y}$ representa el espacio de salida, que contiene $q$ variables objetivo $y_1, y_2 \ldots, y_q$. Por otro lado, $x_\ell^i$ denota el valor de la $\ell$-ésima variable descriptiva para el ejemplo $i$, mientras que $y_\ell^i$ representa el valor de su $\ell$-ésima variable objetivo. En el problema MTR el objetivo principal es construir una función $f : \mathcal{X} \rightarrow \mathcal{Y}$ que prediga un vector de variables objetivo $\hat{\mathbf{y}}$ a partir de un vector de variables descriptivas nunca antes visto $\mathbf{x}$. Por otro lado, como se hace uso de regresión simbólica, es necesaria la definición del conjunto de símbolos no terminales (funciones) $F$=$\{-, +, /, *, \ldots\}$ y el conjunto de símbolos terminales $T$=$\{x_1, x_2, \ldots, x_d\}$, donde cada símbolo terminal representa una variable descriptiva $x_\ell \in \mathcal{X}$.

En el paradigma GEP, un gen está compuesto por una cabeza y una cola. La cabeza puede contener símbolos que pertenezcan tanto a los conjuntos $F$ como $T$, mientras que la cola puede contener solo símbolos terminales. La longitud de la cabeza ($h$) se escoge de acuerdo al problema, mientras que la longitud de la cola ($t$) se calcula como $t$=$h(a-1)+1$, donde $a$ es el número de argumentos de la función con máxima aridad en $F$. Tenga en cuenta que a mayor $h$, más larga y compleja será la expresión matemática que se podrá codificar, sin embargo hay que considerar que también será mayor el espacio de búsqueda. Una desventaja de la mayoría de métodos basados en GEP reside en la continua transformación de los cromosomas en árboles de expresión (ETs, por sus siglas en inglés) y su correspondiente evaluación para obtener el *fitness* de los individuos. Para evitar esto, en este trabajo los genes se codifican mediante notación prefija, siguiendo el enfoque propuesto por Peng et al. en [12], permitiendo la evaluación de los individuos sin construir los ETs, y mejorando por tanto significativamente la eficiencia computacional de GEP. La Figura 1 representa un gen de un cromosoma, que en este caso contiene la notación prefija de la ecuación matemática $f(\mathbf{x}) = \sqrt{\left(\frac{x_1}{x_4 * x_5} - \sqrt{x_1}\right)} + x_1$.

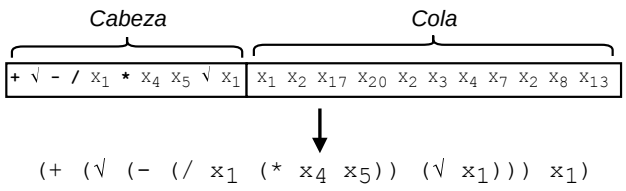



Figura 1. Un gen con $h = 10$ y su correspondiente fenotipo representado en notación prefija.

Se define una población de $p$ individuos, donde cada individuo representa una posible solución completa al problema MTR. Para ello, un individuo es representado mediante un cromosoma multi-génico, compuesto por $q$ genes de igual longitud $c = [g_1, g_1, \ldots, g_q]$, conteniendo un gen por cada variable objetivo del problema. Por tanto, el gen $\ell$-ésimo de un cromosoma codifica una función matemática que se puede utilizar posteriormente como modelo predictivo para estimar el valor de la variable objetivo $y_\ell^i$ de un ejemplo de prueba $i$.

Para calcular el nivel de adaptación de un individuo se calcula el error cuadrático medio relativo (*average relative*

*root mean square error*, aRRMSE) en promedio para todas las variables objetivo sobre el conjunto de entrenamiento $S$,

$$\frac{1}{q}\sum_{\ell=1}^{q}\sqrt{\frac{\sum_{i\in S}(y_\ell^i-\hat{y}_\ell^i)^2}{\sum_{i\in S}(y_\ell^i-\bar{y}_\ell)^2}}\,, \tag{1}$$

donde $y_\ell^i$ e $\hat{y}_\ell^i$ son los valores de la $\ell$-ésima variable objetivo en los vectores de salida conocido ($\mathbf{y}^i$) y predicho ($\hat{\mathbf{y}}^i$) para el ejemplo $i$, respectivamente. Por otro lado, $\bar{y}_\ell$ es el valor medio de la variable objetivo $\ell$-ésima en el conjunto de entrenamiento $S$. Resumiendo, dado un individuo de la población, la función de *fitness* mide el error cuadrático medio relativo (RRMSE) para la primera variable objetivo, evaluando la función matemática codificada en el primer gen del cromosoma en cada ejemplo de entrenamiento $i \in S$, y así este proceso se realiza para todas las variables objetivo obteniendo finalmente el valor promedio entre todas ellas.

La población inicial se crea de manera aleatoria, pero siempre controlando la diversidad entre los individuos generados. Un cromosoma se forma creando cada uno de sus genes de la siguiente manera: (I) los símbolos en la cabeza del gen se seleccionan aleatoriamente de $F \cup T$, asegurando que el primer elemento (raíz) de la cabeza sea un símbolo no terminal; y (II) los símbolos de la cola se seleccionan aleatoriamente del conjunto $T$. Un individuo se añade a la población inicial si tiene una similaridad respecto al resto de la población menor que un umbral específico. La similaridad entre dos individuos se calcula como

$$s(c^i, c^j) = 1 - \frac{\sum_{\ell=1}^{q} h(c_\ell^i, c_\ell^j)}{q^2(t+h)}, \tag{2}$$

donde $c^i$ y $c^j$ son los cromosomas de los individuos $i$ y $j$, respectivamente. La función $h(c_\ell^i, c_\ell^j)$ mide el número de símbolos distintos, sumados a lo largo de las $t+h$ posiciones, en el $\ell$-ésimo gen de los cromosomas $c^i$ y $c^j$. Este método permite la creación de una población inicial con suficiente diversidad, lo cual evita la convergencia temprana del método a un mínimo local.

Respecto al operador de selección, se ha utilizado una selección por torneo binario para crear la población intermedia de padres. Se empleó una baja presión selectiva para que no solo se favorezca a los mejores individuos en el proceso de selección, sino que se estimule también la diversidad entre los individuos seleccionados.

En cuanto al operador de mutación, los padres mutan con una tasa de mutación $p_m$ y se ha diseñado un operador que siempre genera individuos válidos. Se han empleado varios puntos de mutación por cromosoma ($n_{mp}$), y dicho operador realiza los siguientes pasos para cada uno de los puntos de mutación: (I) se selecciona aleatoriamente uno de los genes del individuo; (II) se selecciona aleatoriamente una posición dentro del gen; (III) si la posición pertenece a la cabeza, el símbolo en dicha posición se cambia aleatoriamente por un símbolo que pertenece a $F \cup T$, excepto en el caso de que la posición seleccionada sea la raíz, donde se reemplazará necesariamente por un símbolo en $F$; (IV) si la posición pertenece a la cola, el símbolo en la posición seleccionada se cambia por otro símbolo en $T$.

Una característica de GEP que lo diferencia de otros paradigmas evolutivos es que fragmentos del genoma se pueden activar y saltar a otro lugar en el cromosoma (elementos transponibles), y esta característica da lugar a la definición de los siguientes operadores de transposición: *insertion sequence* (IS), *root insertion sequence* (RIS) y *gene transposition* (GT). El operador IS se utiliza con una pequeña probabilidad $p_{is}$, y permite que pequeños fragmentos de un gen (fragmentos con una función o un terminal en la primera posición) se transpongan a la cabeza de los genes, excepto a la raíz; este operador se ejecuta $n_{is}$ veces sobre un cromosoma. El operador IS realiza los siguientes pasos: (I) dado un cromosoma, se selecciona un gen a partir del cual un fragmento de longitud entre $[1, l_{is}]$ ($l_{is}$-máxima longitud de un fragmento) se selecciona aleatoriamente; y (II) se selecciona un gen de destino donde insertar el fragmento copiado empezando en una posición aleatoria de la cabeza del gen (distinta de la raíz).

Por otra parte, el operador RIS se usa con una probabilidad baja $p_{ris}$, y permite que fragmentos pequeños con un símbolo no terminal en la primera posición se transpongan a la raíz de los genes; este operador se realiza $n_{ris}$ veces sobre un cromosoma. Para ello, este operador realiza los siguientes pasos: (I) dado un cromosoma, se selecciona aleatoriamente un gen; (II) se selecciona un punto aleatorio en la cabeza del gen seleccionado y se busca hacia atrás hasta encontrar un símbolo no terminal; (III) y este fragmento se copia empezando en la raíz del gen. Durante la transposición RIS, los símbolos en la cabeza se desplazan hacia la derecha hasta acomodar completamente el nuevo fragmento copiado. Al igual que en el operador de transposición IS, la cola del gen que se transpone y los genes adyacentes no se modifican.

Por último, el operador GT se usa con una pequeña probabilidad $p_{gt}$, y permite que genes completos se transpongan al principio de los cromosomas. Para ellos, este operador realiza los siguientes pasos: (I) dado un cromosoma, se selecciona aleatoriamente un gen y se elimina de su lugar de origen; (II) y el gen se copia al principio del cromosoma, mientras que los genes restantes se desplazan a la derecha, manteniéndose la longitud original del cromosoma. Los operadores IS, RIS y GT producen siempre individuos válidos, sin embargo, hay que tener en cuenta que dichos operadores pueden modificar drásticamente la expresión de un gen; cuanto más cercano a la raíz del gen ocurra la transposición, más profundo será el cambio en la expresión del gen [13].

En el método propuesto también se aplicaron tres tipos de recombinación o cruce: recombinación en un punto (OP), recombinación en dos puntos (TP), y recombinación de genes (GP), con probabilidades de aplicación $p_{op}$, $p_{tp}$, y $p_{gp}$, respectivamente. Dados dos cromosomas, el operador OP realiza el cruce sobre un punto seleccionado aleatoriamente. El operador TP seleccionan aleatoriamente dos puntos y los fragmentos entre dichos puntos se intercambian entre ambos padres. En GP se intercambian los genes, ubicados en una

posición seleccionada aleatoriamente, entre ambos padres. Es de destacar que en los tres operadores, se cruzan dos padres y se obtienen dos nuevos individuos válidos. Además, el operador TP tiene más poder de transformación que el OP en problemas más complejos, especialmente cuando se utilizan cromosomas multigénicos. Por último, el operador GP no crea genes nuevos como OP y TP, sin embargo, cabe destacar que este operador puede introducir nuevo material en la población, ya que los genes intercambiados entre padres pueden ser muy diferentes.

Finalmente, se ha seguido un esquema generacional para actualizar la población que pasa de una generación a otra, donde el peor individuo de la nueva población es reemplazado por el mejor individuo de la población anterior. El método propuesto (GEPMTR, de ahora en adelante) sigue los pasos que se muestran en la Figura 2. Estos pasos se realizan $n_g$ veces (número de generaciones), y al final del proceso se selecciona el mejor individuo encontrado a lo largo de todas las generaciones, cuyo cromosoma representa un modelo de regresión multi-salida que puede ser utilizado posteriormente para predecir las variables objetivo de conjuntos de ejemplos nunca antes visto.

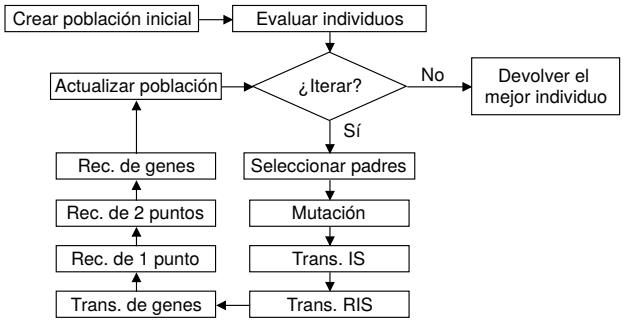



Figura 2. Diagrama de flujo del método GEPMTR.

## III. Estudio Experimental

En esta sección primero se explica la configuración de los experimentos realizados en este trabajo, y posteriormente se presenta un análisis y discusión de los resultados obtenidos.

### *III-A. Configuración de los Experimentos*

Los experimentos se han ejecutado sobre ocho conjuntos de datos de regresión multi-salida[1]. La Tabla I muestra los conjuntos de datos utilizados y sus características, como el número de instancias ($n$), variables descriptivas ($d$) y variables objetivo ($q$).

En el estudio experimental, se compara nuestra propuesta (GEPMTR) con otros dos algoritmos del estado del arte para regresión multi-salida, *Single Target* (ST) y *Regressor Chains* (RC) [2]. Tanto ST como RC usan árboles de regresión como predictores de regresión simple, tal y como recomiendan sus autores. Los parámetros utilizados para la ejecución de GEPMTR se muestran en la Tabla II; la mayoría de los valores son los recomendados por Ferreira [13]. Además, se utilizó una probabilidad global de cruce de 0.8, que corresponde con la suma de las probabilidades de los tres operadores de cruce utilizados.

[1]Los conjuntos de datos están disponibles públicamente en http://mulan.sourceforge.net/datasets-mtr.html

Tabla I
Conjuntos de datos de regresión multi-salida.

| | $n$ | $d$ | $q$ |
|---|---|---|---|
| slump | 103 | 7 | 3 |
| jura | 359 | 15 | 3 |
| sf1 | 323 | 10 | 3 |
| atp1d | 337 | 411 | 6 |
| osales | 639 | 413 | 12 |
| wq | 1060 | 16 | 14 |
| oes97 | 334 | 263 | 16 |
| oes10 | 403 | 298 | 16 |

Tabla II
Parámetros del algoritmo GEPMTR.

| **Parámetro** | **Valor** |
|---|---|
| Conjunto de funciones ($F$) | $\{-, +, /, *, \sqrt{}, sin, cos, log\}$ |
| Conjunto de terminales ($T$) | $\{x_1, x_2, \ldots, x_d\}$ |
| Tamaño de la población ($p$) | 100 |
| Número de puntos de mutación ($n_{mp}$) | 2 |
| Probabilidad de mutación ($n_{mp}$) | 0,1 |
| Número de fragmentos IS ($n_{is}$) | 3 |
| Máx. longitud de los fragmentos IS ($l_{is}$) | 3 |
| Probabilidad IS ($p_{is}$) | 0,1 |
| Número de fragmentos RIS ($n_{ris}$) | 3 |
| Probabilidad RIS ($p_{ris}$) | 0,1 |
| Probabilidad GT ($p_{gt}$) | 0,1 |
| Probabilidad OP ($p_{op}$) | 0,2 |
| Probabilidad TP ($p_{tp}$) | 0,5 |
| Probabilidad GP ($p_{gp}$) | 0,1 |

Para la evaluación de los algoritmos, se utilizó la medida aRRMSE, que ha sido ampliamente usada para la evaluación de métodos de regresión multi-salida [1, 2]. La medida aRRMSE se calcula como se mostró en la Ecuación 1. En todos los experimentos, el valor de aRRMSE se estimó realizando un *10-folds cross-validation*, midiendo el error en cada uno de los conjuntos de test y promediando finalmente los resultados.

Para realizar comparaciones múltiples entre algoritmos, se ha utilizado el test de Friedman [16]. En casos donde el test de Friedman indicase que existen diferencias significativas en el rendimiento de los algoritmos, se ha realizado el test *post-hoc* de Holm [17] para realizar una comparación múltiple con un algoritmo de control. Por otra parte, en los casos donde se compararon dos métodos independientes, se empleó el test de Wilcoxon [18].

### *III-B. Resultados Experimentales*

En el estudio experimental, primero se analizó el comportamiento de GEPMTR considerando diferentes valores del parámetro $h$= $\{5, 10, 15, 20\}$ y número de generaciones. La Tabla III muestra los resultados, donde en cada fila se han resaltado en negrita los mejores valores para cada $h$.

Se puede observar como en prácticamente todos los casos, cuando GEPMTR se ejecutó con 500 generaciones, los individuos evolucionaron hacia un mejor modelo predictivo,

Tabla III
Resultados de GEPMTR dependiendo de $h$ y del número de generaciones.

| | $h=5$ | | $h=10$ | | $h=15$ | | $h=20$ | |
|---|---|---|---|---|---|---|---|---|
| | **100 g.** | **500 g.** | **100 g.** | **500 g.** | **100 g.** | **500 g.** | **100 g.** | **500 g.** |
| slump | **0,860** | 0,872 | 0,848 | **0,772** | 0,864 | **0,767** | 0,845 | **0,752** |
| jura | 0,692 | **0,669** | 0,681 | **0,646** | 0,686 | **0,644** | 0,684 | **0,642** |
| sf1 | 1,129 | **1,083** | 1,128 | **1,128** | **1,142** | 1,164 | 1,156 | **1,141** |
| atp1d | 0,555 | **0,446** | 0,541 | **0,453** | 0,545 | **0,450** | 0,544 | **0,450** |
| osales | 0,954 | **0,860** | 0,910 | **0,861** | 0,908 | **0,881** | 0,921 | **0,888** |
| wq | 0,976 | **0,968** | 0,974 | **0,963** | 0,975 | **0,961** | 0,976 | **0,960** |
| oes97 | 0,660 | **0,589** | 0,650 | **0,601** | 0,658 | **0,598** | 0,666 | **0,623** |
| oes10 | 0,530 | **0,471** | 0,530 | **0,520** | 0,546 | **0,484** | 0,541 | **0,487** |
| *p-value* | **0,023** | | **0,022** | | **0,030** | | **0,014** | |

obteniendo un mejor rendimiento. Para cada valor diferente de $h$, se realizó el test de Wilcoxon para comparar el rendimiento de GEPMTR con 100 y 500 generaciones. Los *p-values* del test de Wilcoxon se muestran en la última fila de la Tabla III. El test estadístico rechazó la hipótesis nula en todos los casos para un nivel de significación $\alpha$=$0,05$, lo cual demuestra que el algoritmo con 500 generaciones obtiene resultados significativamente mejores que con 100 generaciones.

Una vez verificado que GEPMTR funciona mejor con un mayor número de generaciones, el estudio se centra en el valor de $h$. La Tabla IV muestra los *rankings* de GEPMTR ejecutado con diferentes valores de $h$ y con 500 generaciones sobre cada conjunto de datos. Esta tabla es un resumen de la Tabla III, pero esta se centra en los valores de *ranking* en lugar de aRRMSE. En cada fila, el algoritmo con un mejor rendimiento obtiene un *ranking* de 1, el siguiente un *ranking* de 2, y así sucesivamente. La última fila muestra el valor de *ranking* medio devuelto por el test de Friedman.

Tabla IV
Valores de ranking con diferentes valores de $h$ y 500 generaciones.

| | $h$=5 | $h$=10 | $h$=15 | $h$=20 |
|---|---|---|---|---|
| slump | 4,00 | 3,00 | 2,00 | **1,00** |
| jura | 4,00 | 3,00 | 2,00 | **1,00** |
| sf1 | **1,00** | 2,00 | 4,00 | 3,00 |
| atp1d | **1,00** | 4,00 | 2,50 | 2,50 |
| osales | **1,00** | 2,00 | 3,00 | 4,00 |
| wq | 4,00 | 3,00 | 2,00 | **1,00** |
| oes97 | **1,00** | 3,00 | 2,00 | 4,00 |
| oes10 | **1,00** | 4,00 | 2,00 | 3,00 |
| *Ranking* medio | **2,13** | 3,00 | 2,44 | 2,44 |

Se puede observar que, a pesar de que el test de Friedman no encontró diferencias significativas en el rendimiento de GEPMTR con los diferentes valores del parámetro $h$, la configuración con $h$=5 fue la que mejor *ranking* medio obtuvo, mostrando una tendencia a que cuanto más simples sean los modelos, mejor es su rendimiento predictivo.

En la segunda fase del estudio experimental, se realizó una comparación entre GEPMTR y dos algoritmos del estado del arte en MTR. La Tabla V muestra los resultados; para esta comparación, los resultados que se muestran para GEPMTR son el mínimo de las diferentes configuraciones de $h$ con 500 generaciones. La última fila de la tabla muestra los valores de *ranking* medio obtenidos por el test de Friedman. En cada fila, el mejor valor de error se resalta en negrita. Los resultados muestran que el método propuesto obtiene mejores resultados que los otros dos algoritmos en siete de los ocho conjuntos de datos considerados, obteniendo también el mejor valor de *ranking* medio.

Tabla V
Comparación de GEPMTR con dos algoritmos del estado del arte en MTR.

| | **GEPMTR** | **ST** | **RC** |
|---|---|---|---|
| slump | **0,752** | 0,814 | 0,829 |
| jura | **0,642** | 0,696 | 0,704 |
| sf1 | 1,083 | 1,127 | **1,046** |
| atp1d | **0,446** | 0,479 | 0,484 |
| osales | **0,860** | 0,925 | 0,965 |
| wq | **0,960** | 0,966 | 0,974 |
| oes97 | **0,589** | 0,716 | 0,719 |
| oes10 | **0,471** | 0,594 | 0,595 |
| *Ranking* medio | **1,125** | 2,125 | 2,750 |

El estadístico del test de Friedman fue igual a $10,75$, y la hipótesis nula fue rechazada con un *p-value* de $0,005$, existiendo diferencias significativas en el rendimiento de los algoritmos. Para detectar dónde estaban localizadas dichas diferencias se ejecutó el test de Holm, cuyos resultados se muestran en la Tabla VI, indicando que GEPMTR tiene un rendimiento predictivo significativamente mejor que el resto de algoritmos para un nivel de significación $\alpha = 0,05$.

Tabla VI
p-values ajustados calculados por el test de Holm.

| | **ST** | **RC** |
|---|---|---|
| **GEPMTR** vs. | 0,045 | 0,002 |

### *III-C. Discusión*

La representación multi-génica utilizada resultó ser una manera efectiva para representar los cromosomas, permitiendo que cada individuo constituya una solución diferente y completa para el problema MTR. La manera de crear la población inicial, así como el uso de un operador de selección con menor presión, permitió la evolución de una población con suficiente diversidad previniendo que el método quedase atrapado en un mínimo local o en valles aplanados en las primeras generaciones.

Operadores como el de transposición IS, y los de recombinación OP y TP pueden introducir material genético beneficioso. Así, buenos bloques existentes en genes más adaptados se pueden transferir a otros genes, favoreciendo el intercambio de información entre genes, y por tanto, el modelado de relaciones estadísticas entre variables objetivo.

En cuanto a la eficiencia computacional de GEPMTR, el hecho de haber usado notación prefija, y por tanto que los genes puedan ser evaluados sin construir su correspondiente árbol de expresión, hace que la evaluación de los individuos se haya acelerado significativamente. GEPMTR puede ser paralelizado fácilmente, y por tanto, se puede reducir significativamente su

tiempo de cómputo. Por otro lado, no se consume tiempo en la descomposición del problema MTR en varios problemas de regresión simple, sino que el método propuesto trata directamente con los datos multi-salida. Además, GEPMTR tiene implícito un proceso de selección de características que puede ser muy beneficioso en el proceso de aprendizaje de modelos de regresión; la expresión matemática codificada por un gen involucra un número de variables mucho menor que el total de variables descriptivas existentes. Todas las características antes mencionadas permiten la aplicación efectiva del método propuesto en conjuntos de datos de gran escala.

GEPMTR obtuvo resultados prometedores en los experimentos realizados. Sin embargo, la principal desventaja de GEPMTR radica en el análisis de un considerable número de hyperparámetros. Para futuros trabajos será importante expandir el estudio experimental, por ejemplo analizando diferentes valores para las probabilidades de los operadores.

## IV. Conclusiones

En este trabajo se ha presentado un método basado en GEP que resuelve el problema MTR mediante la técnica de regresión simbólica. Tanto la forma de codificación y evaluación de los individuos, como la posibilidad de paralelización, resultan en un método con un coste computacional aceptable y que permite ser usado en conjuntos de datos de gran escala.

Este trabajo corresponde con un estudio inicial, donde se busca analizar los beneficios del paradigma GEP para construir modelos de regresión multi-salida. En trabajos futuros, se estima mejorar algunos de los componentes del algoritmo, como un operador de selección que permita ajustar la presión selectiva. Por otro lado, sería interesante el desarrollo de *ensembles* usando GEPMTR.

## Agradecimientos

Este trabajo ha sido financiado por el proyecto TIN2017-83445-P del Ministerio de Economía y Competitividad y Fondos FEDER, y además por la ayuda FPU del Ministerio de Educación FPU15/02948.

## Referencias

[1] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216–233, 2015.

[2] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, and I. Vlahavas., "Multi-target regression via input space expansion: Treating targets as inputs," *Machine Learning*, vol. 104, no. 1, pp. 55–98, 2016.

[3] D. Kocev, S. Džeroski, M. D. White, G. R. Newell, and P. Griffioen, "Using single and multi-target regression trees and ensembles to model a compound index of vegetation condition," *Ecological Modelling*, vol. 220, no. 8, pp. 1159–1168, 2009.

[4] D. Tuia, J. Verrelst, L. Alonso, F. Pérez-Cruz, and G.Camps-Valls, "Multioutput support vector regression for remote sensing biophysical parameter estimation," *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 4, pp. 804–808, 2011.

[5] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy and Buildings*, vol. 49, no. 560-567, 2012.

[6] O. Reyes, C. Morell, and S. Ventura, "Evolutionary feature weighting to improve the performance of multi-label lazy algorithms," *Integrated Computer-Aided Engineering*, vol. 21, no. 4, pp. 339–354, 2014.

[7] T. Similä and J. Tikka, "Input selection and shrinkage in multiresponse linear regression," *Computational Statistics & Data Analysis*, vol. 52, no. 1, pp. 406–422, 2007.

[8] T. Aho, S. D. B. Ženko, and T. Elomaa, "Multi-target regression with rule ensembles," *Journal of Machine Learning Research*, vol. 373, pp. 2055–2066, 2009.

[9] G. Melki, A. Cano, V. Kecman, and S. Ventura, "Multi-target support vector regression via correlation regressor chains," *Information Sciences*, vol. 415-416, pp. 53–69, 2017.

[10] O. Reyes, A. Cano, H. Fardoun, and S. Ventura, "A locally weighted learning method based on a data gravitation model for multi-target regression," *International Journal of Computational Intelligence Systems*, vol. 11, pp. 282–295, 2018.

[11] S. Stijven, E. Vladislavleva, A. Kordon, L. Willem, and M. E. Kotanchek, *Genetic Programming Theory and Practice XIII*. Cham: Springer, 2016, ch. Prime-Time: Symbolic Regression Takes Its Place in the Real World, pp. 241–260.

[12] Y. Peng, C. Yuan, X. Qin, J. Huang, and Y. Shi, "An improved gene expression programming approach for symbolic regression problems," *Neurocomputing*, vol. 137, pp. 293–301, 2014.

[13] C. Ferreira, "Gene expression programming: A new adaptive algorithm for solving problems," *Complex Systems*, vol. 13, pp. 87–129, 2001.

[14] H. S. Lopes and W. R. Weinert, "Egipsys: An enhanced gene expression programming approach for symbolic regression problems," *International Journal of Applied Mathematics and Computer Science*, vol. 14, no. 3, pp. 375–384, 2004.

[15] J. M. Moyano, E. Gibaja, and S. Ventura, "An evolutionary algorithm for optimizing the target ordering in ensemble of regressor chains," in *IEEE Congress on Evolutionary Computation*, 2017, pp. 2015–2021.

[16] M. Friedman, "A comparison of alternative tests of significance for the problem of $m$ rankings," *Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[17] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.

[18] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

# Análisis de algoritmos de cuantificación basados en ajuste de distribuciones*

Alberto Castaño
*Centro de Inteligencia Artificial*
*Universidad de Oviedo*
Gijón, España
castanoalberto@uniovi.es

Laura Morán-Fernández
*Departamento de Computación*
*Universidade da Coruña*
A Coruña, España
laura.moranf@udc.es

Jaime Alonso
*Centro de Inteligencia Artificial*
*Universidad de Oviedo*
Gijón, España
jalonso@uniovi.es

Verónica Bolón-Canedo
*Departamento de Computación*
*Universidade da Coruña*
A Coruña, España
veronica.bolon@udc.es

Amparo Alonso-Betanzos
*Departamento de Computación*
*Universidade da Coruña*
A Coruña, España
ciamparo@udc.es

Juan José del Coz
*Centro de Inteligencia Artificial*
*Universidad de Oviedo*
Gijón, España
juanjo@uniovi.es

***Resumen*—La cuantificación consiste en predecir la proporción de las distintas clases presentes en una muestra dada de ejemplos. Un tipo de algoritmos de cuantificación se basa en estimar y ajustar las distribuciones subyacentes del conjunto de entrenamiento y de la muestra a predecir. La diferencia clave entre los métodos propuestos reside en cómo estimar las distribuciones: usando los propios atributos o las predicciones dadas por un clasificador. Este artículo analiza ambas alternativas y propone dos nuevos algoritmos. La conclusión es que los métodos basados en usar las predicciones de un clasificador no son en general peores y en ciertos casos producen mejores estimaciones.**

***Palabras clave:*—Cuantificación, Estimación de la prevalencia, Adaptación al dominio**

## I. Introducción

Existen diversas aplicaciones reales que demandan predecir la distribución de probabilidad de las clases en un conjunto de ejemplos. Formalmente, este problema se denomina cuantificación [1]. Ejemplos típicos son predecir la proporción de comentarios positivos y negativos en una red social sobre un acontecimiento o producto durante un período de tiempo [2], cuantificar la cantidad de células dañadas en un tejido [3] o predecir el porcentaje de incidencias de cada tipo en un centro de atención al usuario [1]. En estas aplicaciones no se requiere dar una predicción individual para cada ejemplo sino que basta con retornar una predicción única para toda la muestra.

Aunque a primera vista la cuantificación pueda verse como un subproducto de la clasificación, se pueden diseñar algoritmos de cuantificación que no se basen simplemente en agregar las predicciones dadas por un clasificador. Es el caso de los métodos que estiman y ajustan distribuciones de muestras de ejemplos. Si nos centramos en la cuantificación binaria, la idea consiste en estimar durante el entrenamiento la distribución de la clase positiva y negativa de alguna forma y, en el momento de predecir una nueva muestra, estimar de la misma manera su distribución y aproximarla con una combinación de la distribución de la clase positiva y negativa dependiendo de la proporción de cada clase. La proporción que más aproxime ambas distribuciones será la que retorne el método [3], [4].

Se han propuesto sistemas basados en el ajuste de distribuciones no sólo en el campo de la cuantificación [3], sino también en otro problema relacionado como es el de los algoritmos de adaptación al dominio [5]. En estos últimos la idea es calcular la probabilidad a priori de las clases para actualizar un clasificador, sin necesidad de reentrenarlo, cuando la distribución de las clases cambia. El primer objetivo de este artículo es aunar y analizar ambas líneas de trabajo. Una diferencia clave entre ambas corrientes es cómo estimar las distribuciones. Mientras los algoritmos de adaptación al dominio se basan en estimar las distribuciones usando la información con la que se describen los ejemplos, los métodos de cuantificación proponen usar también las predicciones de un clasificador, asumiendo que ejemplos similares deben obtener predicciones parecidas. Un segundo objetivo de este artículo es comparar ambas alterativas. Asimismo, proponemos dos nuevos métodos que extienden algoritmos de adaptación al dominio, pero usando las predicciones de un clasificador.

Con todo ello, las contribuciones de este trabajo son: 1) presentar dos nuevos cuantificadores basados en ajustar distribuciones usando las predicciones de un clasificador, y 2) realizar un análisis experimental riguroso de los métodos basados en ajuste de distribuciones.

## II. Cuantificación Binaria

Sea $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ un conjunto de entrenamiento, donde $\boldsymbol{x}_i$ es la representación de un ejemplo en el espacio de entrada $\mathcal{X} \subset \mathbb{R}^d$, e $y_i \in \mathcal{Y} = \{-1, +1\}$ su clase. El objetivo de la cuantificación binaria es obtener un modelo, $\bar{h}$, que dado un conjunto de ejemplos sin etiquetar, $T = \{\boldsymbol{x}_j\}_{j=1}^{m}$,

*Esta investigación ha sido financiada en parte por el Ministerio de Economía y Competitividad (MINECO) y el Fondo Europeo de Desarrollo Regional (FEDER), a través del proyecto coordinado TIN2015-65069-C2.

prediga la proporción de la clase positiva y la clase negativa. Dado que ambas son complementarias, basta con predecir la proporción o prevalencia de la clase positiva $\hat{p}$, siendo la de la clase negativa $1-\hat{p}$. En símbolos: $\bar{h} : \mathbb{N}^{\mathcal{X}} \longrightarrow [0,1]$, donde $\mathbb{N}^{\mathcal{X}}$ denota un multi-conjunto de ejemplos, representado por el número de veces que cada posible ejemplo $\boldsymbol{x} \in \mathcal{X}$ aparece en dicho multi-conjunto.

La aproximación más sencilla para producir un cuantificador es la denominada Clasificar y Contar (CC), que consiste en entrenar un clasificador binario, $h$, clasificar con él todos los ejemplos de $T$ y contar los ejemplos de cada clase: $\hat{p}_{CC} = \bar{h}(T) = \frac{1}{m}\sum_{\boldsymbol{x}_i \in T} I(h(\boldsymbol{x}_i) = +1)$, siendo $I$ la función indicatriz. Sin embargo, los estudios muestran [6] que CC produce en general resultados subóptimos, especialmente cuando la distribución de las clases cambia significativamente entre el entrenamiento y la fase de predicción. Por ese motivo se han propuesto distintos algoritmos de cuantificación. Una revisión completa de ellos puede encontrarse en [7].

La cuantificación, por su propia definición, es uno de esos problemas de aprendizaje [8] en los que se sabe de antemano que la distribución de los datos cambia, es decir, que $P_D(x,y) \neq P_T(x,y)$, ya que es obvio que al menos cambia $P_D(y) \neq P_T(y)$[1]. La cuestión es determinar qué elementos de la distribución, entre $P(x)$, $P(x|y)$ y $P(y|x)$, se asume que no cambian para facilitar el aprendizaje. De hecho, la clave y el primer paso para diseñar un algoritmo de cuantificación es definir precisamente eso. La mayoría de los algoritmos de cuantificación están diseñados bajo la asunción de que $P(x|y)$ se mantiene constante y que el resto de factores puede cambiar.

Esta asunción la sigue por ejemplo el método AC [1], que probablemente es el cuantificador más usado y que tomaremos como algoritmo básico de comparación en este estudio. La idea del método AC es que, dado que $P(x|y)$ es constante, los ratios de verdaderos positivos, $tpr$, (*true positive rate*) y de falsos positivos, $fpr$, (*false positive rate*) del clasificador usado por el método CC serán constantes también, aunque cambie la distribución de probabilidad de las clases. Eso implica que la prevalencia que predice el método CC, $\hat{p}_{CC}$, se puede escribir en función de la prevalencia real $p$:

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

Despejando $p$ calculamos la prevalencia estimada por AC:

$$\hat{p}_{AC} = \frac{\hat{p}_{CC} - fpr}{tpr - fpr}. \tag{2}$$

En base a este desarrollo matemático, los pasos del método AC consisten en, primero, entrenar un clasificador y estimar su *tpr* y *fpr*, después usar el método CC para calcular $\hat{p}_{CC}$ y finalmente aplicar (2) para obtener la predicción final, $\hat{p}_{AC}$. En teoría AC produce predicciones perfectas siempre que se cumpla que: 1) $P(x|y)$ es constante y 2) las estimaciones del *tpr* y *fpr* son perfectas. Obviamente, es difícil que ambas condiciones se cumplan totalmente en problemas reales de una cierta complejidad, pero aún así el método suele ofrecer un buen rendimiento.

[1]Nótese que si $P_D(y) = P_T(y)$ el problema de cuantificación sería trivial

## III. Métodos Basados en Ajuste de Distribuciones

El enfoque para el problema de la cuantificación en el que se centra este artículo se basa en estimar y ajustar las distribuciones de entrenamiento y test. La idea fundamental consiste en modificar la distribución de entrenamiento, que abusando de notación denotaremos por $D'$, mediante una mezcla de la distribución de los ejemplos positivos, $D^+$, y la distribución de los negativos, $D^-$, en función de la prevalencia estimada de ambas clases, es decir,

$$D' = D^- \cdot (1-\hat{p}) + D^+ \cdot \hat{p}. \tag{3}$$

El objetivo es tratar de aproximar $D'$ lo más posible a la distribución estimada para el conjunto de test $T$. La Figura 1 trata de ilustrar esta idea. En la figura izquierda se observa la distribución de los ejemplos positivos y de los negativos de entrenamiento. En la parte derecha se muestra la distribución observable en el conjunto de test. Asumiendo de nuevo que $P(x|y)$ es constante, y que por tanto las distribuciones de $D^+$ y $D^-$ cambiarán uniformemente en función de la prevalencia de las clases, el objetivo es minimizar la diferencia entre la distribución de test y la distribución modificada $D'$:

$$\min_{\hat{p}} \Delta(T, D') = \min_{\hat{p}} \Delta(T, D^- \cdot (1-\hat{p}) + D^+ \cdot \hat{p}). \tag{4}$$

En el ejemplo de la Figura 1 es evidente que la prevalencia de la clase positiva en el conjunto de test es menor que la observada en los datos de entrenamiento y que por lo tanto, tenemos que disminuir ese valor para ajustar las distribuciones.

Los métodos basados en ajustar distribuciones comparten un marco común que consta de los siguientes elementos:

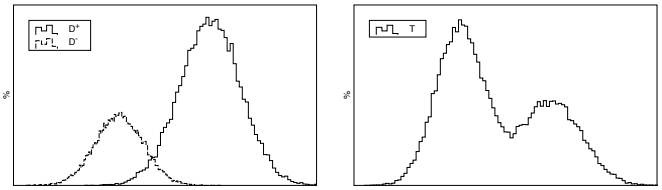
- una forma de estimar o representar las distribuciones,
- una medida $\Delta$ para compararlas, y
- un método para calcular el mínimo de (4).

La diferencia entre los métodos que se estudian en este artículo está en alguno de estos tres elementos, pero nuestro interés principal es analizar las distintas formas usadas para estimar las distribuciones y en menor medida las métricas empleadas para compararlas.

En la Figura 1 se ha dejado deliberadamente sin etiquetar el eje horizontal para indicar que la distribución se puede representar usando diferentes datos. No obstante, hay dos corrientes fundamentales: usar la propia descripción de los ejemplos (lo que algunos métodos llaman usar la información de las $X$'s), o usar las predicciones dadas por un clasificador (usar las predicciones $y$)[2]. Las distribuciones de la Figura 1 podrían perfectamente representar funciones de densidad de probabilidad tanto del valor de un atributo del espacio de entrada $\mathcal{X}$, como de la probabilidad a posteriori dada por un clasificador de que un ejemplo pertenezca a la clase positiva, $P(y=+1|x)$.

En cuanto a la medida $\Delta$, hay múltiples opciones incluyendo normas como $L1$ o $L2$ y medidas de similitud o divergencia entre distribuciones de probabilidad como la divergencia

[2]No confundir con usar las etiquetas reales presentes en $D$




(a) Distribuciones del cjto de entrenamiento (b) Distribución del cjto de test

Figura 1: Los métodos basados en comparación y ajuste estiman de alguna forma la distribución de (a) los ejemplos positivos y negativos disponibles en el entrenamiento, $D^+$ y $D^-$, y (b) de los ejemplos de la muestra a predecir, $T$. Después tratan de ajustar la combinación de las distribuciones de $D^+$ y $D^-$ usando (3) para aproximar la distribución de $T$

de Kullback-Leibler (KLD) y la distancia de Hellinger. En el artículo analizaremos las medidas que se han propuesto hasta ahora y propondremos dos nuevos métodos usando una función basada en distancias y otra en rankings.

### *III-A. Ajuste Mediante la Distancia de Hellinger*

Analizando la literatura de cuantificación, los métodos más relevantes basados en ajuste de distribuciones son los propuestos en [3]. Se trata de dos métodos basados en la distancia de Hellinger (HD) para comparar las distribuciones, donde éstas se representan mediante histogramas: en un caso del valor de los atributos (versión llamada HDX) y en el otro del valor de las predicciones de un clasificador (HDy). Usar histogramas hace que estos métodos tengan un hiperparámetro, $b$, que es el número de intervalos o *bins* usados en los histogramas.

Usando la definición de la distancia de Hellinger para el caso discreto multivariante, y aplicando (3) para representar $D'$, resulta en el siguiente problema a minimizar:

$$\min_{\hat{p}} \frac{1}{d}\sum_{l=1}^{d}\sqrt{\sum_{k=1}^{b}\left(\sqrt{\frac{|T_{k,l}|}{m}}-\sqrt{\frac{|D^-_{k,l}|}{n^-}(1-\hat{p})+\frac{|D^+_{k,l}|}{n^+}\hat{p}}\right)^2}, \tag{5}$$

donde $n^-$ y $n^+$ son el número de ejemplos en $D^-$ y $D^+$ respectivamente y $|T_{k,l}|/m$, $|D^-_{k,l}|/n^-$ y $|D^+_{k,l}|/n^+$ son la proporción de ejemplos de $T$, $D^-$ y $D^+$ que pertenece al bin $k$ en la dimensión $l$. En el caso de HDy, solamente tenemos una dimensión que se corresponde con las predicciones del clasificador, luego el primer sumatorio desaparece.

Los autores proponen una búsqueda lineal, variando $\hat{p}$ en el rango $[0,1]$ en pequeños incrementos, para resolver (5). Sin embargo, la solución se puede hallar analíticamente, con más precisión y menos coste computacional, teniendo en cuenta la equivalencia entre la distancia de Hellinger y el coeficiente de Bhattacharyya, $HD(T,D') = \sqrt{1-BC(T,D')}$ [9], y resolviendo el siguiente problema de optimización:

$$\begin{aligned} \min_{\pi} \quad & 1-\sum_k \sqrt{T_k(D'\pi)_k}, \\ s.a. \quad & \sum \pi = 1, \quad \pi_i \geq 0, \end{aligned} \tag{6}$$

donde, sobrecargando la notación, definimos:

- $T$ como una matriz con $bd \times 1$ en el caso del método HDX, con las proporciones de los bins del primer atributo en las $b$ primeras filas, y así sucesivamente con el resto de atributos, y en el caso de HDy con solamente $b$ filas y una columna con las proporciones de cada bin calculadas con las predicciones del clasificador,
- $D'$ es una matriz de $bd \times 2$ en el caso del método HDX, y $b \times 2$ en el caso del HDy, con idéntica estructura a la matriz $T$, donde las dos columnas representan, en este orden, las proporciones de la clase negativa y positiva, y
- $\pi$ una matriz $2 \times 1$ con dos variables, $\pi_1$ la prevalencia de la clase negativa, y $\pi_2$ la de la positiva, $\hat{p}$.

Este problema puede resolverse con cualquier librería de optimización convexa; nosotros empleamos `cvxpy` para Python.

En teoría, la ventaja de HDX es usar directamente la información disponible en los atributos con los que se representan los ejemplos, lo cual puede ser una desventaja en espacios de alta dimensionalidad, o cuando los problemas dependan mucho de la interacción entre los valores de los atributos, ya que estos se consideran de forma en cierto modo independiente al calcular la HD, como hemos descrito anteriormente. La ventaja del método HDy es que resume en una sola dimensión, a través de un clasificador, la distribución de los ejemplos, asumiendo que ejemplos parecidos deben obtener predicciones similares. La desventaja del método HDy es que el clasificador debe entrenarse adecuadamente y las estimaciones de las predicciones de los ejemplos deben hacerse no en reescritura, sino usando validaciones cruzadas con un alto número de particiones para evitar el sobreajuste en las predicciones.

### *III-B. Ajuste Mediante la Distancia Energy*

Varios métodos de adaptación al dominio [5], [10] se basan en calcular la nueva distribución de las clases (denominadas probabilidades a priori en este contexto). La idea es usar esas nuevas probabilidades a priori para ajustar el clasificador disponible sin necesidad de reentrenarlo. Es obvio que dichos métodos también pueden usarse como algoritmos de cuantificación, aunque su objetivo sea otro bien distinto. En la literatura de adaptación al dominio es habitual que la comparación entre las distribuciones de entrenamiento y test se haga solamente usando el valor de los atributos [4], [11]. Dichos métodos plantean los mismos problemas que comentamos anteriormente para el método HDX: a veces es complejo calcular las densidades antes de compararlas. No obstante, siguiendo ese enfoque, recientemente se ha publicado un método [12] que destaca entre los propuestos en ese campo ya que es computacionalmente muy eficiente y tiene un mejor rendimiento que otros métodos anteriores [4].

Dicho algoritmo, que denotaremos por EDX, se basa en minimizar la distancia *Energy* (ED) entre la distribución de $T$ y la distribución modificada $D'$ definida por:

$$\begin{aligned} \min_{\hat{p}} \quad & 2\cdot\mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D'}||\boldsymbol{x}_j-\boldsymbol{x}_i|| \\ & -\mathbb{E}_{\boldsymbol{x}_j,\boldsymbol{x}'_j \backsim T}||\boldsymbol{x}_j-\boldsymbol{x}'_j|| - \mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}'_i \backsim D'}||\boldsymbol{x}_i-\boldsymbol{x}'_i||, \end{aligned} \tag{7}$$

donde $||\cdot||$ representa la distancia euclídea. Nótese que el segundo término puede suprimirse al ser independiente de $\hat{p}$ debido a que solo interviene la distribución de $T$. Omitimos aquí la derivación matemática para resolver este problema ya que es similar al que detallaremos en la sección siguiente.

## IV. Métodos Propuestos

Nuestra primera propuesta consiste en hacer el método complementario al EDX, que llamaremos EDy y que emplea las predicciones de un clasificador $h$ en lugar de los atributos que definen $\mathcal{X}$. Para ello debemos minimizar la expresión:

$$\begin{aligned}\min_{\hat{p}} \quad & 2 \cdot \mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D'} ||h(\boldsymbol{x}_j) - h(\boldsymbol{x}_j)||_1 \\ & -\mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D'} ||h(\boldsymbol{x}_i) - h(\boldsymbol{x}'_i)||_1.\end{aligned} \tag{8}$$

Si desdoblamos ambos términos aplicando (3) tenemos

$$\begin{aligned}\min_{\hat{p}} \quad & 2 \cdot (1-\hat{p}) \cdot \mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D^-} ||h(\boldsymbol{x}_j) - h(\boldsymbol{x}_i)||_1 \\ & +2 \cdot \hat{p} \cdot \mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D^+} ||h(\boldsymbol{x}_j) - h(\boldsymbol{x}_i)||_1 \\ & -(1-\hat{p})^2 \cdot \mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D^-} ||h(\boldsymbol{x}_i) - h(\boldsymbol{x}'_i)||_1 \\ & -2 \cdot \hat{p} \cdot (1-\hat{p}) \mathbb{E}_{\boldsymbol{x}_i \backsim D^+, \boldsymbol{x}'_i \backsim D^-} ||h(\boldsymbol{x}_i) - h(\boldsymbol{x}'_i)||_1 \\ & -\hat{p}^2 \cdot \mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D^+} ||h(\boldsymbol{x}_i) - h(\boldsymbol{x}'_i)||_1.\end{aligned} \tag{9}$$

En la práctica podemos aproximar cada esperanza usando las correspondientes muestras de ejemplos disponibles:

$$\mathbb{E}_{\substack{\boldsymbol{x}_j \backsim T \\ \boldsymbol{x}_i \backsim D^-}} \ldots \approx \mu_{T,D^-} = \frac{1}{mn^-} \sum_{\boldsymbol{x}_j \in T} \sum_{\boldsymbol{x}_i \in D^-} ||h(\boldsymbol{x}_j) - h(\boldsymbol{x}_i)||_1. \tag{10}$$

Sustituyendo todos los términos tenemos

$$\begin{aligned}\min_{\hat{p}} \quad & 2(1-\hat{p})\mu_{T,D^-} + 2\hat{p}\mu_{T,D^+} \\ & -(1-\hat{p})^2 \mu_{D^-,D^-} - 2\hat{p}(1-\hat{p})\mu_{D^-,D^+} - \hat{p}^2 \mu_{D^+,D^+},\end{aligned} \tag{11}$$

un problema fuertemente convexo [12] en el que derivando e igualando a cero obtenemos que

$$\hat{p} = \frac{\mu_{T,D^-} - \mu_{T,D^+} - \mu_{D^-,D^-} + \mu_{D^-,D^+}}{-\mu_{D^-,D^-} + 2\mu_{D^-,D^+} - \mu_{D^+,D^+}}. \tag{12}$$

El segundo método propuesto trata de completar el tipo de medidas consideradas para comparar distribuciones. Por un lado tenemos una medida basada en histogramas (Hellinger), una segunda basada en distancias (*Energy*) y se podría considerar una tercera opción que se basase en rankings, por ejemplo usando el criterio Cramér-von Mises (CvM). Después de analizar varias alternativas que utilizaban medidas derivadas de CvM [13], observamos que planteaban ciertos problemas en el contexto de la cuantificación binaria. El principal es que los errores en los rankings dados por un clasificador no son simétricos respecto a las clases: la clase positiva tiende a dar valores más altos, por ejemplo, al usar un clasificador probabilístico que devuelva $P(y = +1|x)$, lo que hace que los errores sean mayores y la medida sea sesgada hacia esa clase. Por ese motivo, finalmente decidimos usar el enfoque propuesto en [14] y crear el método CvMy que consiste en:

1. Calcular un ranking conjunto de los ejemplos disponibles en $T$, $D^-$ y $D^+$,
2. Utilizar la ED para ajustar las distribuciones.

Es decir, aplicamos el mismo algoritmo que en el método EDy, pero en lugar de calcular la distancia entre las predicciones, calculamos la distancia entre los rankings de las predicciones. En este caso $\mu_{T,D^-}$ se calcularía como

$$\mu_{T,D^-} = \frac{1}{mn^-} \sum_{\boldsymbol{x}_j \in T} \sum_{\boldsymbol{x}_i \in D^-} ||r(h(\boldsymbol{x}_j)) - r(h(\boldsymbol{x}_i))||, \tag{13}$$

donde $r(h(\boldsymbol{x}))$ nos devuelve la posición en el ranking conjunto para todos los ejemplos disponibles, tanto del conjunto de entrenamiento como de la muestra de test a predecir. Luego la diferencia entre EDy y CvMy es simplemente que en un caso usamos las distancias entre las predicciones y en otro caso las diferencias entre los rankings de esas predicciones.

## V. Experimentos

El objetivo de los experimentos[3] era comparar los métodos descritos en las secciones III y IV sobre varios conjuntos de datos. Se prestó especial atención a aquellos pares de métodos que, empleando la misma técnica, actúan sobre $X$ ó $y$. Otro aspecto a analizar es la diferencia entre los métodos que utilizan distintas métricas para ajustar las distribuciones. Con el fin de tener una base de comparación, se incluyó el método AC. Para realizar los experimentos se emplearon $41$ conjuntos de datos. Entre ellos se encuentran conjuntos binarios y otros que son versiones binarizadas de conjuntos originalmente multiclase. Así, por ejemplo, el conjunto balance.1 es un conjunto binario en el que la clase 1 original es la clase positiva, formando el resto la clase negativa.

Los métodos que necesitan un clasificador (AC, HDy, EDy y CvMy) fueron entrenados garantizando que usen exactamente los mismos clasificadores. Como algoritmo de clasificación se optó por utilizar *Random Forest* (RF) con salida probabilística para obtener modelos no lineales. La representación usada por el método HDy se generó mediante 8 *bins* por ser el valor que da mejores resultados en estudios previos [15]. Respecto a la hiperparametrización de RF, se llevó a cabo una búsqueda donde la profundidad variaba entre $[1, 5, 10, 15, 20, 25, 30]$, el número de árboles entre $[9, 18, 27, 36, 45, 54, 63]$, y el número mínimo de ejemplos para los nodos hoja entre $[1, 2, 4, 6, 8, 10]$. Dicha búsqueda se realizó optimizando la media geométrica mediante una validación cruzada de tres particiones de tal forma que se obtuvieran buenos clasificadores incluso cuando las clases no estuvieran balanceadas.

Todos los modelos se entrenaron sobre las mismas particiones de los conjuntos de datos en subconjuntos de entrenamiento y test, usando el $70\,\%$ de los datos para entrenar y el $30\,\%$ restante para testear el modelo, haciendo $20$ repeticiones. Con cada partición de test se generaron a su vez $50$ conjuntos aleatorios con diferente prevalencia mediante muestreo con reemplazamiento. La Tabla I presenta el error absoluto medio de las $1000$ muestras totales de test, $\frac{1}{1000}\sum_{i=1}^{1000} |\hat{p}_i - p_i|$.

[3]Con el fin de facilitar la reproducibilidad de estos experimentos, los conjuntos de datos, el código y los resultados obtenidos están disponibles en http://github.com/albertorepo/analisis-cuantificacion.

Tabla I: Error absoluto medio para cada método sobre cada conjunto de datos

| conjunto | AC | CvMy | EDX | EDy | HDX | HDy |
|---|---|---|---|---|---|---|
| acute.a | **0.037108** | 0.063550 | 0.043139 | 0.045118 | 0.055545 | 0.038207 |
| acute.b | **0.032788** | 0.066146 | 0.036890 | 0.041999 | 0.041406 | 0.054910 |
| balance.1 | 0.035783 | 0.033945 | **0.023380** | 0.030977 | 0.031761 | 0.029968 |
| balance.3 | 0.036954 | 0.041549 | 0.031478 | 0.036346 | 0.038860 | **0.028690** |
| breast-cancer | **0.016127** | 0.041405 | 0.021433 | 0.019083 | 0.020661 | 0.016980 |
| cmc.1 | 0.086332 | 0.078364 | 0.078307 | 0.076136 | 0.069633 | **0.066261** |
| cmc.2 | 0.107941 | 0.119897 | **0.067238** | 0.115609 | 0.070623 | 0.108277 |
| cmc.3 | 0.123313 | 0.102359 | 0.097244 | 0.094503 | **0.089708** | 0.099939 |
| coil | **0.074547** | 0.088761 | 0.091519 | 0.084081 | 0.152487 | 0.111920 |
| ctg.1 | **0.015548** | 0.024147 | 0.029277 | 0.017414 | 0.039771 | 0.015466 |
| ctg.2 | 0.029396 | 0.034666 | 0.033221 | 0.029420 | 0.035380 | **0.027457** |
| ctg.3 | 0.036554 | 0.034602 | 0.037974 | **0.028830** | 0.072301 | 0.029866 |
| default_credit | 0.019715 | 0.022824 | 0.019998 | 0.020298 | **0.019029** | 0.022613 |
| diabetes | 0.072729 | 0.054864 | 0.059925 | **0.053669** | 0.079258 | 0.057511 |
| german | 0.087824 | 0.094541 | **0.078250** | 0.094154 | 0.100175 | 0.108456 |
| haberman | 0.243842 | 0.249899 | **0.166582** | 0.235816 | 0.263772 | 0.210012 |
| ionosphere | **0.048585** | 0.066306 | 0.060841 | 0.055436 | 0.055167 | 0.051017 |
| iris.1 | 0.002824 | 0.093470 | **0.016286** | 0.019482 | 0.019413 | 0.036264 |
| iris.2 | 0.055456 | 0.072105 | 0.083458 | **0.048620** | 0.050365 | 0.060695 |
| iris.3 | 0.056763 | 0.066537 | 0.048433 | 0.046074 | **0.040583** | 0.075169 |
| lettersH | 0.020787 | 0.026393 | 0.027549 | 0.025537 | 0.024807 | **0.016682** |
| mammographic | 0.055443 | 0.049573 | 0.047413 | 0.044584 | 0.040516 | **0.039722** |
| normtrans | 0.126085 | 0.115678 | **0.083582** | 0.111233 | 0.184914 | 0.140455 |
| normwine.1 | 0.044364 | 0.053990 | **0.031585** | 0.037656 | 0.036819 | 0.041440 |
| normwine.2 | 0.050073 | 0.052678 | **0.041247** | 0.044185 | 0.067116 | 0.052616 |
| normwine.3 | 0.043011 | 0.058478 | **0.037641** | 0.037738 | 0.058761 | 0.059418 |
| pageblocks.5 | 0.042875 | 0.045982 | 0.056530 | 0.042546 | 0.116209 | **0.037511** |
| phoneme | 0.016554 | 0.019391 | 0.020960 | 0.015883 | 0.017325 | **0.012115** |
| semeion.8 | 0.058507 | 0.058614 | 0.086534 | 0.058085 | 0.053439 | **0.042278** |
| sonar | 0.113939 | 0.108738 | **0.103082** | 0.107576 | 0.126948 | 0.110015 |
| spambase | **0.008127** | 0.015965 | 0.011639 | 0.010721 | 0.020363 | 0.008359 |
| spectf | 0.184524 | 0.162914 | 0.105445 | 0.150938 | **0.088333** | 0.134393 |
| tictactoe | **0.048306** | 0.061411 | 0.081694 | 0.056250 | 0.078846 | 0.048768 |
| transfusion | 0.107033 | 0.132669 | **0.079578** | 0.116017 | 0.162173 | 0.139883 |
| wdbc | 0.022888 | 0.030576 | 0.025309 | 0.016853 | 0.027982 | **0.016830** |
| wine-quality-red | 0.045189 | 0.037138 | 0.042694 | **0.035102** | 0.047469 | 0.037303 |
| wine-quality-white | 0.031770 | 0.030004 | **0.028205** | 0.028864 | 0.029644 | 0.025918 |
| wine.1 | 0.037356 | 0.051096 | 0.034395 | **0.033942** | 0.036904 | 0.044647 |
| wine.2 | 0.044871 | 0.056931 | 0.051078 | **0.041804** | 0.068114 | 0.051526 |
| wine.3 | 0.037569 | 0.057219 | 0.036618 | **0.035190** | 0.048123 | 0.038674 |
| yeast | 0.060799 | 0.062919 | **0.051120** | 0.063225 | 0.060161 | 0.052552 |

El método ganador para cada conjunto se destaca en negrita. Puede verse que no hay un método que gane claramente a los demás. Tampoco se observan diferencias concluyentes entre las formas de estimar las distribuciones ni entre las medidas para ajustarlas. Entre los métodos propuestos, el algoritmo EDy es superior, siendo competitivo con los algoritmos previos, lo que parece no ocurrir en el caso del método CvMy.

Para analizar los resultados desde un punto de vista estadístico hemos aplicado análisis bayesianos en lugar de los tradicionales tests de contraste de la hipótesis nula ya que estamos de acuerdo con las desventajas de estos últimos apuntadas en [16]. Previo a la aplicación de estos tests, es necesario definir la *región de equivalencia práctica* (*rope*, por sus siglas en inglés): dos métodos se consideran prácticamente equivalentes si la diferencia dada una métrica no supera un cierto umbral. En nuestro caso, consideramos dos cuantificadores equivalentes si la diferencia en error absoluto es menor del $1\,\%$. La elección del error absoluto como medida de evaluación se debe a que es una métrica acotada entre 0 y 1 y por lo tanto, *rope* puede definirse como un porcentaje de variación dentro de ese intervalo. Para otras métricas como la KLD no sería trivial la elección de tal región, dado que su valor tiene un rango infinito, además de una difícil interpretación práctica. Una vez fijado el valor de *rope* es posible realizar el análisis para cada par de métodos tanto en cada conjunto individual (Tabla II) como para todos los conjuntos usando un test jerárquico [16] (Tabla III). En el primer caso podemos calcular la probabilidad de cada una de las tres posibilidades: que gane uno de los dos cuantificadores o que los resultados caigan en la zona de equivalencia. Cuando una de esas tres probabilidades es mayor que el $95\,\%$ consideramos que hay una diferencia significativa en favor de esa alternativa, y si no, lo marcamos como un conjunto *sin decisión*. Analizando la Tabla II vemos una ligera ventaja en favor de los métodos $y$ frente a los métodos $X$, más clara en el caso de HDy frente a HDX que en la comparación EDy vs. EDX. En el caso de CvMy, es inferior a HDy y EDy, especialmente frente al segundo. En cambio EDy obtiene mejores resultados que el resto de métodos excepto HDy, pero la diferencia entre ambos es pequeña. Si analizamos los resultados a nivel global con el

Tabla II: Número de conjuntos para los que el test bayesiano decide que existe diferencia significativa ($\geq 95\,\%$)

| Par ($m1$-$m2$) | $m1$ | $rope$ | $m2$ | Sin decisión |
|---|---|---|---|---|
| EDX-AC | 16 | 7 | 16 | 2 |
| EDX-CvMy | 17 | 11 | 7 | 6 |
| EDX-EDy | 9 | 16 | 12 | 4 |
| EDX-HDX | 10 | 15 | 7 | 9 |
| EDX-HDy | 10 | 15 | 14 | 2 |
| EDy-AC | 14 | 14 | 9 | 4 |
| EDy-CvMy | 11 | 24 | 0 | 6 |
| EDy-HDX | 15 | 11 | 9 | 6 |
| EDy-HDy | 8 | 13 | 10 | 10 |
| HDX-AC | 11 | 9 | 17 | 4 |
| HDX-CvMy | 16 | 9 | 13 | 3 |
| HDX-HDy | 6 | 13 | 19 | 3 |
| HDy-AC | 11 | 19 | 11 | 0 |
| HDy-CvMy | 15 | 15 | 6 | 5 |
| AC-CvMy | 16 | 11 | 9 | 5 |

Tabla III: Resultados del test bayesiano jerárquico

| Par ($m1$-$m2$) | $P(m1 \gg m2)$ | $P(rope)$ | $P(m2 \gg m1)$ |
|---|---|---|---|
| EDX-AC | 0.537 | 0.000 | 0.463 |
| EDX-CvMy | 0.970 | 0.013 | 0.018 |
| EDX-EDy | 0.239 | 0.252 | 0.509 |
| EDX-HDX | 0.347 | 0.598 | 0.055 |
| EDX-HDy | 0.342 | 0.009 | 0.649 |
| EDy-AC | 0.523 | 0.202 | 0.275 |
| EDy-CvMy | 0.286 | 0.714 | 0.000 |
| EDy-HDX | 0.768 | 0.134 | 0.099 |
| EDy-HDy | 0.105 | 0.744 | 0.150 |
| HDX-AC | 0.162 | 0.002 | 0.837 |
| HDX-CvMy | 0.706 | 0.001 | 0.293 |
| HDX-HDy | 0.022 | 0.005 | 0.973 |
| HDy-AC | 0.451 | 0.072 | 0.477 |
| HDy-CvMy | 0.971 | 0.005 | 0.024 |
| AC-CvMy | 0.814 | 0.005 | 0.180 |

test jerárquico (Tabla III) vemos que hay pocas diferencias significativas (HDy vs. CvMy, HDy vs. HDX y EDy vs. CvMy). Se observa de nuevo la ligera ventaja de los métodos $y$, EDy y HDy, frente a EDX y HDX. Estos datos son un resumen de la distribución de las diferencias entre cada par de métodos. La forma de la distribución puede verse mediante un gráfico simplex (Figura 2). Por ejemplo, es interesante ver que en la comparación EDy-CvMy la nube de puntos está muy alejada del vértice de CvMy, a pesar de que la diferencia no resulte significativa, mostrando una superioridad clara.

## VI. Conclusiones

Este trabajo tenía como principal objetivo realizar un análisis comparativo entre diferentes algoritmos de cuantificación basados en ajuste de distribuciones. Los resultados experimentales obtenidos muestran que los métodos basados en las $y$'s, es decir, en el uso de clasificadores, no son en general peores, obteniendo mejores estimaciones en ciertas ocasiones. En cuanto a los métodos propuestos, EDy se configura como una buena alternativa, pero no así el algoritmo propuesto basado en ranking, CvMy. Como futura línea de investigación, se plantea la aplicación de algún algoritmo de selección de características en los métodos $X$ para tratar de obtener un subconjunto con los mejores atributos. Además, sería interesante realizar una búsqueda de características en los problemas

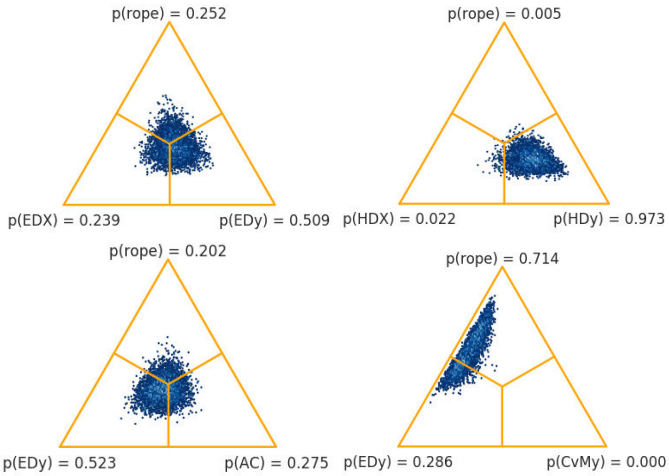



Figura 2: Gráficos simplex de la comparación mediante el test jerárquico bayesiano de varios pares de métodos

(alta dimensionalidad de entrada, presencia de ruido, etc.), que puedan implicar un mejor comportamiento en cada método.

## Referencias

[1] G. Forman, "Quantifying counts and costs via classification," *Data Mining and Knowledge Discovery*, vol. 17, no. 2, pp. 164–206, 2008.

[2] P. González, J. Díez, N. Chawla, and J. J. del Coz, "Why is quantification an interesting learning problem?" *Progress in Artificial Intelligence*, pp. 1–6, 2016.

[3] V. González-Castro, R. Alaiz-Rodríguez, and E. Alegre, "Class distribution estimation based on the hellinger distance," *Information Sciences*, vol. 218, pp. 146–164, 2013.

[4] M. Sugiyama, T. Kanamori, T. Suzuki, M. C. du Plessis, S. Liu, and I. Takeuchi, "Density-difference estimation," *Neural Computation*, vol. 25, no. 10, pp. 2734–2775, 2013.

[5] A. Margolis, "A literature review of domain adaptation with unlabeled data," University of Washington, Tech. Rep., 2011.

[6] J. Barranquero, P. González, J. Díez, and J. J. del Coz, "On the study of nearest neighbor algorithms for prevalence estimation in binary problems," *Pattern Recognition*, vol. 46, no. 2, pp. 472–482, 2013.

[7] P. González, A. Castaño, N. V. Chawla, and J. J. del Coz, "A review on quantification learning," *ACM Computing Surveys*, vol. 50, no. 5, pp. 74:1–74:40, 2017.

[8] J. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition*, vol. 45, no. 1, pp. 521–530, 2012.

[9] A. Firat, "Unified framework for quantification," *arXiv preprint arXiv:1606.00868*, 2016.

[10] H. Daume III and D. Marcu, "Domain adaptation for statistical classifiers," *JAIR*, vol. 26, pp. 101–126, 2006.

[11] A. Iyer, S. Nath, and S. Sarawagi, "Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection." in *ICML*, 2014, pp. 530–538.

[12] H. Kawakubo, M. C. Du Plessis, and M. Sugiyama, "Computationally efficient class-prior estimation under class balance change using energy distance," *IEICE Tran. on Inf. and Sys.*, vol. 99, pp. 176–186, 2016.

[13] L. Morán-Fernández, V. Bolón-Canedo, and A. Alonso-Betanzos, "A distributed approach for classification using distance metrics," in *ESANN*, 2017.

[14] J. Curry, X. Dang, and H. Sang, "A rank-based Cramér-von-Mises-type test for two samples," *arXiv preprint arXiv:1802.06332*, 2018.

[15] P. Pérez-Gállego, J. R. Quevedo, and J. J. del Coz, "Using ensembles for problems with characterizable changes in data distribution: A case study on quantification," *Information Fusion*, vol. 34, pp. 87–100, 2017.

[16] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis," *Journal of Machine Learning Research*, vol. 18, no. 77, pp. 1–36, 2017.

# $k$-Vecinos más Cercanos Difuso para Clasificación Monotónica

Sergio González, Salvador García, Francisco Herrera
*Dept. de Ciencias de la Computación e Inteligencia Artificial*
*Universidad de Granada*
Granada, España
{sergiogvz, salvagl, herrera}@decsai.ugr.es

Sheng-Tun Li
*Dept. de Gestión Industrial y de Información*
*Universidad Nacional Cheng Kung*
Tainan, Taiwan
stli@mail.ncku.edu.tw

Robert John
*Escuela de Ciencias de la Computación*
*Universidad de Nottingham*
Nottingham, Reino Unido
robert.john@nottingham.ac.uk

***Resumen*—La clasificación con restricciones monotónicas proviene de una necesidad de algunos problemas reales, en los que la variable de salida no disminuyen con un aumento de las entrada, o viceversa. Los conjuntos de datos reales con frecuencia tiene una gran cantidad de ruido de clase. Este incrementa el número de violaciones monotónicas y repercute negativamente en el rendimiento de los clasificadores. Por lo que, algunos métodos deben recurrir a técnicas de re-etiquetado que pueden alterar el conocimiento del problema. Además, la inclusión de estas restricciones obligan a los clasificadores a diversificar sus objetivos entre precisión y monotonicidad.**

**Con esta contribución, se propone un nuevo modelo basado en los $k$-Vecinos más Ceranos Difuso para la clasificación con restricciones monotónicas, MonF$k$NN. Este clasificador incorpora un nuevo cálculo de pertenencias difusas, que lo dota de robustez frente ruido monotónico sin necesidad de re-etiquetado. Asimismo, se ha diseñado para ser adaptable a las diferentes necesidades del problema. Tras varios estudios experimentales, ha demostrado ser tremendamente más preciso igualando el grado de monotonicidad de los mejores métodos de la literatura.**

***Palabras Clave*—Ciencia de datos, aprendizaje perezoso, vecinos más cercanos difuso, restricciones monotónicas, clasificación ordinal, regresión ordinal**

## I. Introducción

La evaluación de activos o individuos [1], [2] tienen como objetivo determinar los artículos más valiosos según sus virtudes, es decir, la clasificación en etiquetas ordinales según sus atributos ordinales. Además, estas aplicaciones suelen conllevar una restricción monotónica entre las entradas y la clase. Es decir, la predicción de clase de un individuo no debe disminuir con un mejor valor para una determinada variable, fijando el resto. Estos problemas se conocen como clasificación con restricciones de monotonicidad [3] y la ruptura de estas, como violaciones de monotonicidad.

Este trabajo ha contado con el apoyo del Proyecto Español TIN2017-89517-P y una beca de investigación (FPU) otorgada a S. González por el MECD.

Estas tareas de aprendizaje exigen otros requisitos además de modelos precisos, como la consistencia monótona de las predicciones o la minimización de los errores de clasificación. Sin embargo, estos otros objetivos pueden perjudicar la precisión. Por lo tanto, se debe buscar un equilibro justo entre las diferentes necesidades de cada problema.

En estos últimos años, se ha diseñado nuevos algoritmos que consideran estas restricciones [1], [3]–[6]. El aprendizaje basado en la instancia ha demostrado ser una buena aproximación para la clasificación monotónica [4], [6]. Sin embargo, algunos de estos métodos, como la variante monotónica de $k$-Vecinos más Cercanos [4] (M$k$NN), necesitan aprender de un conjunto completamente monótono para asegurar predicciones monótonas. Esta circunstancia rara vez se cumple en conjuntos reales, donde el ruido y las discrepancias son comunes. Por ello, se hace uso de estrategias de re-etiquetado que alteran las etiquetas de clase de algunos ejemplos forzando un conjunto monotónico [4].

En clasificación estándar, la versión difusa de los $k$-Vecinos más Cercanos [7] (F$k$NN) es un método muy sólido con un gran rendimiento, gracias a su alta robustez al ruido [8]. En esta contribución, se presenta un nuevo modelo diseñado sobre F$k$NN con nociones de M$k$NN para tener en cuenta las restricciones de monotonicidad, llamado MonF$k$NN. La robustez de F$k$NN se ha adaptado para reducir el impacto de las violaciones monotónicas sin la necesidad de re-etiquetado. Adicionalmente, se han añadido otras técnicas para tratar las pertenencias de instancia de una manera más adecuada según la monotonicidad. Nuestra propuesta MonF$k$NN es un clasificador flexible que cubre diferentes necesidades de monotonicidad y precisión, con buen equilibrio entre ambos, ajustando sus parámetros.

Este documento está organizado de la siguiente manera. En la Sección II, presentamos el problema de la clasificación con restricciones monotónicas y los métodos relacionados con

nuestra propuesta. La sección III está dedicada a explicar con lujo detalle la propuesta MonF$k$NN. En la Sección IV, se presenta el marco experimental utilizado en los diferentes estudios empíricos. La sección V plantea tres estudios experimentales: un análisis sobre los parámetros de nuestro método y sus dos configuraciones más relevantes, una profunda comparación con el Estado-del-Arte y un estudio de la robustez de MonF$k$NN frente al ruido. Finalmente, en la Sección VI, se presentan las principales conclusiones de este estudio.

## II. Preliminares

En esta sección, se introducen los preliminares sobre los que se construye este trabajo.

### II-A. Clasificación Monotónica

La clasificación monotónica tiene como objetivo predecir la etiqueta de la clase $y$ a partir del vector de la característica de entrada $x$, donde $y \in \mathcal{Y} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_c\}$, $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^K$. Como propiedad principal, los atributos y sus predicciones están monotónicamente restringidos por conocimiento previo del problema, es decir, $\mathbf{x} \succeq \mathbf{x}' \rightarrow f(\mathbf{x}) \geq f(\mathbf{x}')$ [9], donde $\mathbf{x} \succeq \mathbf{x}'$ implica $\forall_{j=1,\ldots,K}, \mathbf{x}_j \geq \mathbf{x}'_j$, esto es, $\mathbf{x}$ domina $\mathbf{x}'$. Por lo tanto, el objetivo principal es construir clasificadores que no violen estas restricciones, conocidos como clasificadores monótonos.

Se pueden distinguir dos tipos diferentes de clasificadores monotónicos: los modelos monotónicos aproximados, que minimizan el número de violaciones monotónicas en sus decisiones, y los clasificadores monotónicos puros, cuyas predicciones son siempre monotónicas en relación con el entrenamiento. Esto último es difícil de lograr, especialmente en aplicaciones de la vida real, donde los conjuntos de datos de formación rara vez son puramente monótonos. Para ser considerado monotónico, un conjunto de datos debe tener todos sus pares de instancias monotónicas entre sí [6]: $\mathbf{x}_i \succeq \mathbf{x}_j \rightarrow y_i \geq y_j, \forall_{i,j}$.

### II-B. k-Vecinos más Cercanos Monotónico

La aproximación de los $k$-Vecinos más Cercanos para la clasificación monotónica [4] (M$k$NN) modifica la regla estándar del vecino más cercano para evitar violaciones monotónicas en sus predicciones. Para ello, M$k$NN calcula para cada nuevo ejemplo $x_i$ el rango $r_i = [y_{min}, y_{max}]$ de etiquetas de clase válidas, que satisfagan las restricciones monotónicas. El $y_{min}$ inferior de $r_i$ se calcula como la etiqueta de clase más alta de todas las instancias por debajo del ejemplo $x_i$. Análogamente, el límite superior $y_{max}$ es la etiqueta de clase mínima de las instancias superiores a $x_i$ (véase la Ec. 1).

$$r_i = \begin{cases} y_{min} = \text{máx}\{y | (x,y) \in \mathcal{D} \wedge x_i \succeq x\} \\ y_{max} = \text{mín}\{y | (x,y) \in \mathcal{D} \wedge x \succeq x_i\} \end{cases} \quad (1)$$

Se pueden distinguir dos variantes diferentes dependiendo de cómo se extraen los vecinos para una nueva instancia $i$. La variante *enRango* considera los ejemplos $k$ más cercanos $x_j$ con sus etiquetas de clase $y_j$ en el rango $[y_{min}, y_{max}]$. La versión *fueraDeRango* extrae los vecinos, independientemente de su clases, y luego, se eliminan aquellos cuyas clases están fuera del rango $r_i$. Como en el $k$-NN estándar, la clase mayoritaria entre los vecinos de $k$ se usa como la etiqueta predicha.

M$k$NN requiere conjuntos de datos monotónicos para funcionar correctamente. Por ello, se utiliza una técnica de re-etiquetado para transformar los datos de entrenamiento. Estas técnicas pretenden identificar las violaciones de la monotonicidad y hacer los mínimos cambios posibles con la mínima diferencia de clase para eliminarlas [4].

### II-C. k-Vecinos más Cercanos Difuso

Los métodos difusos de los $k$-Vecinos más Cercanos [8] incluyen conceptos difusos en la clásica regla $k$-NN para aprender de conjuntos difusos y producir reglas de clasificación difusas. Para una nueva instancia $x_i$, el algoritmo original F$k$NN [7] extrae sus $K$ vecinos más cercanos de la misma manera que el $k$-NN estándar. Sin embargo, sus pertenecías a cada clase $l$ se calculan con la siguiente expresión:

$$u(x,l) = \frac{\sum_{j=1}^{K} u(x_j, l) * \frac{1}{||x - x_j||^{(m-1)}}}{\sum_{j=1}^{K} \frac{1}{||x - x_j||^{(m-1)}}} \quad (2)$$

Como se muestra en 2, la pertenencia $u(x_i, l) = u_{il}$ de la muestra $x_i$ a la clase $l$ se asigna con el producto de la pertenencia a la clase $u(x_j, l)$ de los vecinos $x_j$ y la inversa de sus distancias a $x_i$. Este último sirve como un peso que sesga hacia las pertenencias de las muestras más cercanas. El parámetro $m$ determina el grado de influencia de las distancias a los vecinos. La etiqueta final de clase para el ejemplo $x_i$ será aquella $l$ con el mayor grado de pertenencia $u_{il}$.

Partiendo de un conjunto de entrenamiento etiquetado, F$k$NN [7] lo transforma en un conjunto difuso con pertenencias de clase usando la regla del vecino más cercano. Para cada muestra de entrenamiento $x$, se extraen sus $k$ vecinos más cercanos y, luego, sus pertenencias se calculan de acuerdo a 3. Esta transformación ha demostrado ser útil frente a muestras ruidosas, ya que sus pertenencias se repartirán por las clases circundantes perdiendo su influencia.

$$u(x,l) = \begin{cases} 0{,}51 + 0{,}49 * (nn_l/k) & \text{if} \quad l = C(x) \\ 0{,}49 * (nn_l/k) \end{cases} \quad (3)$$

## III. $k$-Vecinos más Cercanos Difuso para clasificación monotónica

En esta sección, se explica en detalle nuestra propuesta de $k$-Vecinos más Cercanos Difuso para clasificación monotónica.

### III-A. De probabilidades a etiqueta final de clase

Es esencial elegir el mecanismo ideal según la monotonicidad para obtener una clase final a partir de una función de probabilidad. La clase más probable es el método más

común. Sin embargo, esto podría ser inapropiado para escenarios con restricciones monótonas. Por ejemplo, sea $x \leq y$ y sus probabilidades $f_x = (0{,}2, 0{,}2, 0{,}4, 0{,}2, 0{,}0)$ y $f_y = (0{,}0, 0{,}4, 0{,}3, 0{,}2, 0{,}1)$, entonces sus clases finales rompen la monotonía: $argmax(f_x) = l_3 > l_2 = argmax(f_y)$. Aunque, la instancia $y$ tenga mayor peso en etiquetas más altas que $x$. Es más, $f_y$ domina débilmente a $f_x$ según el primer grado dominancia estocástica (FSD) [10], ya que la función de distribución acumulativa $F_x$ de $x$, es mayor, elemento por elemento, que $F_y$, es decir, $f_x \preceq_{FSD} f_y \iff (\forall l \in \mathcal{Y})(F_x(l) \geq F_y(l))$. La FSD ha sido de utilidad para definir las restricciones de monotonicidad en las clasificaciones estocásticas [5], [11], con la expresión $x \leq y \implies f_x \preceq_{FSD} f_y$.

La mediana satisface dicha expresión, preservando la monotonicidad [5], [10]. Esta se calcula siguiendo la definición de percentil 50 como el rango$[l_m, l_M]$:

$$\begin{aligned} l_m &= \text{mín}\{l \in \mathcal{Y} | Prob\{X \leq l\} \geq 1/2\} \\ l_M &= \text{máx}\{l \in \mathcal{Y} | Prob\{X \geq l\} \geq 1/2\} \end{aligned} \tag{4}$$

Volviendo al ejemplo anterior, las clases para $x$ y $y$ elegidas por la mediana no rompen la monotonicidad: $med(f_x) = med(f_y) = 3$.

### *III-B. Pertenencias de clase robustas al ruido monotónico*

En esta subsección, se explica el cálculo de pertenencias de clase rediseñado para reducir la influencia de las instancias no-monotónicas. En primer lugar, se ha de tratar las violaciones monótonas más simples, es decir, instancias repetidas con diferentes clases. MonF$k$NN primero sustituye las réplicas de cualquier ejemplo $x$ por solo vector de característica $x$ y sus pertenencias $u(x)$. La probabilidad $u(x, l)$ de la instancia $x$ y la clase $l$ se calcula con frecuencia de ejemplos duplicados pertenecientes a $l$, como se muestra en la siguiente expresión:

$$u(x, l) = \frac{|\{z \in \mathcal{D} | z = x \wedge C(z) = l\}|}{|\{z \in \mathcal{D} | z = x\}|} \tag{5}$$

Posteriormente, MonF$k$NN estima las pertenencias del resto de instancias con la información de los vecinos más cercanos a cada una. Sin embargo, estos vecinos son extraídos con un M$k$NN configurado como *enRango* en lugar de uno tradicional. Una vez obtenidos los vecinos de cada ejemplo $x$, la información de clase se fusiona en pertenencias de $x$. La probabilidad $u(x, l)$ de pertenecer a la clase $l$ se calcula con la siguiente expresión:

$$u(x, l) = \begin{cases} \textit{RCr} + (nn_l/k) * (1 - \textit{RCr}) & \text{if} \quad l = C(x) \\ (nn_l/k) * (1 - \textit{RCr}) \end{cases} \tag{6}$$

donde $nn_l$ es el número de vecinos de la clase $l$, $k$ el número total de vecinos y $C(x)$ es la etiqueta original del ejemplo $x$. *RCr* es un nuevo parámetro llamado *Relevancia de la clase real"*. *RCr* puede ser visto como la probabilidad mínima asignada a la clase original $C(x)$ de la instancia $x$, en caso de que no haya vecinos etiquetados con $C(x)$.

Hay algunos valores para *RCr* en el rango $[0, 1]$ que tienen comportamientos muy interesantes y distintos. En el caso de un conjunto de datos realmente ruidoso, *RCr* podría establecerse en 0, dejando toda la responsabilidad al cálculo del rango $r_i$ de las clases válidas y los vecinos más cercanos. En presencia de solo instancias repetidas, el usuario puede optar por tratarlas con *RCr* $= 1$. Finalmente, si se desea considerar las instancias etiquetadas originalmente, se recomienda asignar *RCr* a $0{,}5$. Este valor asegura que la clase real esté dentro del conjunto de medianas. En contraste con F$k$NN y sus $0{,}51$, si todos los vecinos pertenecen a la misma clase diferente a la actual, nuestro método obliga a elegir una clase intermedia. Por lo general, este último valor ofrece un comportamiento balanceado, estable y con un mejor rendimiento.

### *III-C. Agregación flexible de las pertenencias*

Después de estimar la pertenencia de clase de cada instancia de entrenamiento, nuestro algoritmo está listo para predecir nuevos ejemplos. Esta fase de predicción ha sido diseñada con una mayor flexibilidad, permitiendo al usuario elegir entre predicciones más precisas o puramente monótonas.

Para ello, nuestro método utiliza otro M$k$NN para obtener los vecinos a utilizar en la predicción final. Este M$k$NN tiene también dos versiones, *enRango* y *fueraDeRange*. Pero son sustancialmente diferentes en comparación con las variantes originales, especialmente para la variante *fueraDeRange*.

La alternativa *enRango* se basa en la misma idea del M$k$NN original, donde los vecinos de un ejemplo deben pertenecer a un conjunto de clases monotónicamente válidas. Sin embargo, este rango de clases se obtiene utilizando las medianas adquiridas de las pertenencias de clase de las instancias de entrenamiento. Este avance mejora nuestro método hacia la robustez del ruido monotónico. La primera fase de M$k$NN se centra en la preservación de la monotonía de las medianas resultantes, para servir como una ventaja para las predicciones finales.

La versión *fueraDeRango* de nuestro método es completamente diferente de la anterior regla *fueraDeRango*. Ha sido diseñada con la intención de priorizar la precisión del clasificador sobre la monotonicidad. Con este propósito, nuestro método considera cualquier ejemplo como un vecino válido sin importar su clase. A diferencia del modelo original, no se realiza ningún filtrado o eliminación de vecinos fuera del rango válido. Sin embargo, su relevancia en la agregación de pertenencias se puede reducir, gracias a un factor de penalización introducido en la expresión de agregación.

Luego, para un nuevo ejemplo $x$, se obtienen sus vecinos más cercanos según la variante elegida. Sus pertenencias se agregan con la misma fórmula utilizada por el F$k$NN original con la adición del factor de penalización para la versión *fueraDeRange*. La siguiente expresión muestra cómo se integra este parámetro:

$$u(x,l) = \frac{\sum_{j=1}^{K} u(x_j,l) * \frac{pOR_j}{||x-x_j||^{(m-1)}}}{\sum_{j=1}^{K} \frac{pOR_j}{||x-x_j||^{(m-1)}}} \quad (7)$$

Como anteriormente, la pertenencia $u(x_i,l)$ de la nueva muestra $x_i$ a la clase $l$ es el resultado de la suma de las pertenencias a dicha clase $u(x_j,l)$ de los vecinos $x_j$ ponderadas inversamente con la distancia a $x_i$. En la versión *fueraDeRange* de nuestro método, hay otro factor de ponderación en la contribución a las probabilidades finales, el parámetro mencionado como *"penalización de fueraDeRango"* (*pOR*). El factor $pOR_j$ sólo se aplica si la clase $y_j$ del vecino $x_j$ no se encuentra en el rango de clases válidas $r_i$ de $x_i$. Se puede configurar con valores continuos de 0 a 1. Cuando se asigna a 1, no se aplica ninguna penalización. El valor 0 significa una penalización completa, es decir, los vecinos con clases inválidas no participarán en la agregación. Este último comportamiento es equivalente a la variante *fueraDeRango* del M$k$NN original. Se recomienda el uso de 0.5, ya que es un buen equilibrio entre reducir su relevancia y considerarlos en la decisión.

Finalmente, la predicción de clase del nuevo ejemplo $x$ es la mediana de las pertenencias de clase normalizadas resultantes.

MonF$k$NN se ha desarrollado para ser robusto al ruido monotónico y versátil en múltiples escenarios gracias a sus parámetros. Entre sus posibilidades, destacan dos configuraciones con un comportamiento muy distintivo: Monotónico Puro (PM) y Monotónico Aproximado (AM). La configuración monotónica pura corresponde a un valor de 0.5 para el parámetro *RCr* y el uso de la regla *enRango* para obtener las pertenencias de nuevas instancias. Este enfoque pretende dar predicciones con las mínimas violaciones de la monotonicidad.

La configuración monotónica aproximada prioriza la capacidad de predicción y relaja las restricciones monótonas. Las pertenencias del entrenamiento se obtienen mediante el tratamiento de las muestras repetidas. Los ejemplos únicos tendrán una probabilidad de 1 para la clase real y 0 para el resto. Este comportamiento se consigue con $RCr = 1$. Ya que buscamos predicciones más precisas, se consideran todas las instancias como vecinos válidos. Sin embargo, aquellas con etiquetas de clase inválidas contribuirán con la mitad de sus probabilidades ($pOR = 0{,}5$).

## IV. Metodología experimental

Esta sección introduce el marco experimental utilizado en los diferentes estudios empíricos del trabajo. En dichos experimentos, se han incluido 12 conjuntos de datos diferentes. Estos son retratados en la Tabla I, donde se detallan el número de instancias, atributos y clases de cada conjunto en las columnas Ins., At. y Cl., respectivamente. La columna Direcciones indica la dirección de la relación monotónica entre cada atributo y la clase: monotonía directa (+) o inversa (-).

Se ha llevado a cabo un esquema de validación cruzada de 10 particiones (10-fcv) para ejecutar los diferentes clasificadores sobre estos conjuntos. Sus particiones han sido extraídas del repositorio KEEL [12].

Tabla I: Descripción de los 12 conjunto de datos usados.

| Conjuntos | Ins. | At. | Cl. | Direcciones |
|---|---|---|---|---|
| *artiset* | 1000 | 2 | 10 | directas |
| *balance* | 625 | 4 | 3 | {-, -, +, +} |
| *bostonhousing4cl* | 506 | 13 | 4 | {-, +, -, +, -, +, -, +, -, -, -, +, -} |
| *car* | 1728 | 6 | 4 | directas |
| *ERA* | 1000 | 4 | 9 | directas |
| *ESL* | 488 | 4 | 9 | directas |
| *LEV* | 1000 | 4 | 5 | directas |
| *machineCPU* | 209 | 6 | 4 | {-, +, +, +, +, +} |
| *qualitative_bankruptcy* | 250 | 6 | 2 | directas |
| *SWD* | 1000 | 10 | 4 | directas |
| *windsorhousing* | 546 | 11 | 2 | directas |
| *wisconsin* | 683 | 9 | 2 | directas |

La tabla II muestra los clasificadores involucrados en la comparación experimental y sus parámetros.

Tabla II: Parámetros considerados para los algoritmos a comparar.

| Algoritmos | Parámetros |
|---|---|
| M$k$NN [4] | $k$ = 5, distancia = euclidiana, tipoVecinos = enRango |
| OSDL [5] | equilibrado = No, tipoClasificación = mediana, límiteInferior = 0, límiteSuperior = 1 ajusteInterpolación = No, ponderado = No, pasoInterpolación = 10, gradoInterpolación = 0.5 |
| OLM [6] | resolución = conservativa tipoClasificación = conservativa |
| MID [3] | R = 1, confianza = 0.25, elementosPorHoja = 2 |
| Mon Fuzzy$k$NN | $k$ = 5, $K$ = 9, distancia = euclidiana |
| Monotónico Puro | $RCr$ = 0.5, tipoVecinos = enRango |
| Monotónico Aprox. | $RCr = 1$, tipoVecinos = fueraDeRango, $pOR = 0{,}5$ |

Para evaluar la competencia de los clasificadores, se han empleado tres medidas que cubren diferentes aspectos de su rendimiento: capacidad predictiva con la precisión estándar, coste de los errores con el error absoluto medio (MAE) y monotonicidad con el Índice No Monotónico (NMI). El NMI mide la proporción de pares de muestras que rompen la monotonicidad respecto al total de pares.

## V. Resultados y análisis

Esta sección presenta los resultados de diferentes estudios empíricos y sus análisis.

### V-A. *Evaluación de las propuestas de k-Vecinos más Cercanos Difuso Monotónico*

A continuación, una comparación entre las versiones Pura y Aproximada de MonF$k$NN, destacando los diferentes comportamientos y aspectos de su rendimiento. La Tabla III contienen los resultados de las dos configuraciones de la propuesta en términos de Precisión, MAE y NMI. La fuente en negrita indica los mejores resultados obtenidos para cada conjunto de datos y métrica.

En la Tabla III, las diferencias entre ambos enfoques quedan claramente patentes. Tal como fueron diseñados, MonF$k$NN Aproximado (AM) tiene una mejor precisión en promedio en

Tabla III: Resultados de las versiones Puramente y Cuasi Monotónicas de MonF$k$NN.

| | Precisión | | MAE | | NMI | |
|---|---|---|---|---|---|---|
| | **PM** ($RCr = 0,5$) | **AM** ($pOR = 0,5$) | **PM** ($RCr = 0,5$) | **AM** ($pOR = 0,5$) | **PM** ($RCr = 0,5$) | **AM** ($pOR = 0,5$) |
| *artiset* | 0,9309 | **0,9349** | 0,0691 | **0,0651** | **0,0000** | **0,0000** |
| *balance* | **0,9307** | 0,9008 | **0,0853** | 0,1168 | **0,0000** | 0,0001 |
| *bostonhousing4cl* | 0,6561 | **0,7134** | 0,3972 | **0,3261** | **0,0000** | 0,0001 |
| *car* | 0,9740 | **0,9834** | 0,0295 | **0,0195** | **0,0000** | **0,0000** |
| *ERA* | 0,2420 | **0,2430** | **1,2813** | 1,2993 | **0,0052** | **0,0052** |
| *ESL* | 0,7036 | **0,7131** | 0,3149 | **0,3053** | 0,0004 | **0,0003** |
| *LEV* | **0,6377** | 0,6110 | **0,3927** | 0,4223 | **0,0004** | 0,0009 |
| *machineCPU* | **0,7033** | 0,6699 | **0,3158** | 0,3493 | **0,0002** | 0,0017 |
| *qualitative_bankruptcy* | **0,9960** | **0,9960** | **0,0040** | **0,0040** | **0,0000** | **0,0000** |
| *SWD* | 0,5807 | **0,5833** | **0,4370** | 0,4380 | 0,0007 | **0,0003** |
| *windsorhousing* | 0,7576 | **0,7839** | 0,2424 | **0,2161** | **0,0005** | 0,0051 |
| *wisconsin* | 0,9653 | **0,9663** | 0,0347 | **0,0337** | **0,0000** | **0,0000** |
| *Media*: | 0,7565 | **0,7583** | 0,3003 | **0,2996** | **0,0006** | 0,0012 |

el 75 % de los conjuntos utilizados. Por otro lado, MonF$k$NN Puro (PM) logra predicciones monotónicamente más confiables en 10 de los 12 conjuntos de datos utilizados, con grandes diferencias en los problemas de *Windsorhousing* y *MachineCPU*. Ambos tienen un resultado estable y bueno en términos de MAE, siendo la versión AM un poco mejor.

Ya que la monotonicidad se priorizada la mayoría de las veces en la clasificación con restricciones monotónicas, usaremos la variante Pura de MonF$k$NN en los siguientes estudios empíricos.

### *V-B. Comparación con el Estado-del-Arte*

A lo largo de este experimento, se evalúa el rendimiento de MonF$k$NN en comparación a los clasificadores monotónicos del Estado-del-Arte. En esta, se busca un equilibrio entre predicciones precisas y monotónicas.

La Tabla IV recoge los resultados de precisión para los diferentes conjuntos de datos obtenidos por los algoritmos probados. En términos de precisión, MonF$k$NN rinde bastante mejor con un amplio margen. Nuestro método logra predicciones más precisas para 7 conjuntos de datos, con casos particularmente notables, como *balance* o *ESL*.

Tabla IV: Resultados en precisión obtenidos por los algoritmos evaluados.

| | **MonF$k$NN-PM** | **M$k$NN** | **OSDL** | **OLM** | **MID** |
|---|---|---|---|---|---|
| *artiset* | **0,9309** | 0,9199 | 0,1952 | 0,7948 | 0,7237 |
| *balance* | **0,9307** | 0,8624 | 0,6352 | 0,8320 | 0,7808 |
| *bostonhousing4cl* | 0,6561 | 0,6126 | 0,2787 | 0,5277 | **0,6739** |
| *car* | **0,9740** | 0,9711 | 0,9549 | 0,9543 | 0,8027 |
| *ERA* | 0,2420 | 0,1990 | 0,2320 | 0,1690 | **0,2760** |
| *ESL* | **0,7036** | 0,6332 | 0,6721 | 0,5738 | 0,6414 |
| *LEV* | 0,6377 | 0,4630 | **0,6400** | 0,4250 | 0,6070 |
| *machineCPU* | **0,7033** | 0,6890 | 0,2919 | 0,6746 | 0,6220 |
| *qualitative_bankruptcy* | **0,9960** | **0,9960** | 0,9160 | 0,9800 | 0,9840 |
| *SWD* | 0,5807 | 0,5200 | **0,5840** | 0,4160 | 0,5540 |
| *windsorhousing* | 0,7576 | 0,5861 | 0,4927 | 0,7564 | **0,8205** |
| *wisconsin* | **0,9653** | 0,9649 | 0,9590 | 0,8873 | 0,9517 |
| *Media:* | **0,7565** | 0,7014 | 0,5710 | 0,6659 | 0,7031 |

En la clasificación con restricciones monotónicas, el coste de los errores puede ser esencial. La Tabla V muestra el error en forma de MAE cometido por cada clasificador. Al igual que el rendimiento de precisión, MonF$k$NN es claramente mejor que el resto con el error más pequeño en promedio y para 8 de los conjuntos de datos. En aquellos problemas donde otros son superiores, como *LEV* o *wisconsin*, este consigue resultados realmente cercanos.

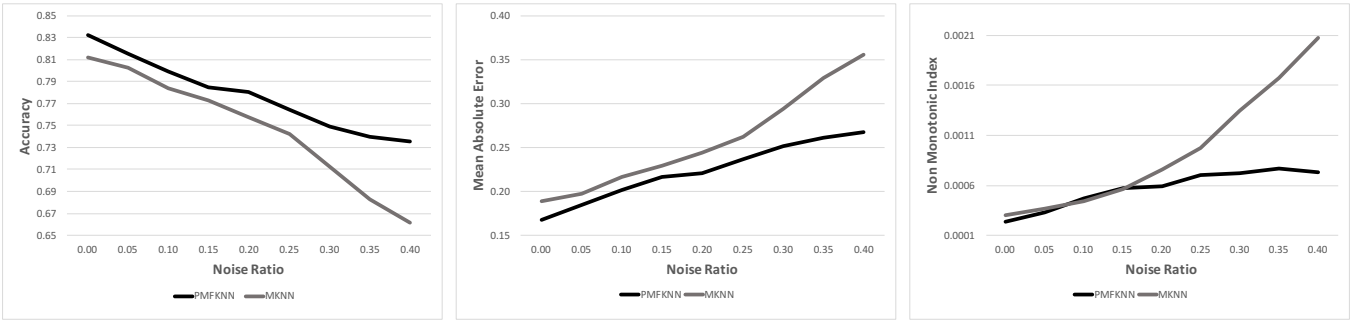
Tabla V: Resultados en MAE obtenidos por los algoritmos evaluados.

| | **MonF$k$NN-PM** | **M$k$NN** | **OSDL** | **OLM** | **MID** |
|---|---|---|---|---|---|
| *artiset* | **0,0691** | 0,0771 | 1,6897 | 0,2082 | 0,3123 |
| *balance* | **0,0853** | 0,1504 | 0,4912 | 0,1920 | 0,3360 |
| *bostonhousing4cl* | 0,3972 | 0,4901 | 0,9368 | 0,5988 | **0,3893** |
| *car* | **0,0295** | 0,0359 | 0,0475 | 0,0538 | 0,2506 |
| *ERA* | **1,2813** | 1,4270 | 1,2850 | 2,1500 | 1,2970 |
| *ESL* | **0,3149** | 0,3791 | 0,3607 | 0,4734 | 0,3934 |
| *LEV* | 0,3927 | 0,5740 | **0,3920** | 0,6680 | 0,4290 |
| *machineCPU* | **0,3158** | 0,3301 | 0,9043 | 0,3589 | 0,4211 |
| *qualitative_bankruptcy* | **0,0040** | **0,0040** | 0,0840 | 0,0200 | 0,0160 |
| *SWD* | **0,4370** | 0,4840 | **0,4370** | 0,7630 | 0,4750 |
| *windsorhousing* | 0,2424 | 0,4304 | 0,5073 | 0,2436 | **0,1795** |
| *wisconsin* | 0,0347 | **0,0337** | 0,0410 | 0,1127 | 0,0483 |
| *Media:* | **0,3003** | 0,3680 | 0,5980 | 0,4869 | 0,3790 |

En relación a la monotonicidad, la Tabla VI presenta los resultados de los NMI alcanzados. En este caso, la competencia por ser el mejor está mucho más reñida. OLM y MID obtienen predicciones claramente menos monótonas. Este último tiene el peor comportamiento considerando sólo la monotonicidad. MonF$k$NN, M$k$NN y OSDL funcionan de manera similar. MonF$k$NN y OSDL son ligeramente mejores en promedio.

Tabla VI: Resultados en NMI obtenidos por los algoritmos evaluados.

| | **MonF$k$NN-PM** | **M$k$NN** | **OSDL** | **OLM** | **MID** |
|---|---|---|---|---|---|
| *artiset* | **0,0000** | **0,0000** | **0,0000** | **0,0000** | 0,0039 |
| *balance* | **0,0000** | 0,0001 | 0,0006 | **0,0000** | 0,0017 |
| *bostonhousing4cl* | **0,0000** | **0,0000** | **0,0000** | 0,0003 | 0,0022 |
| *car* | **0,0000** | **0,0000** | **0,0000** | **0,0000** | 0,0046 |
| *ERA* | 0,0052 | 0,0056 | **0,0049** | 0,0063 | 0,0082 |
| *ESL* | **0,0004** | 0,0012 | 0,0006 | 0,0025 | 0,0021 |
| *LEV* | **0,0004** | 0,0010 | **0,0004** | 0,0043 | 0,0018 |
| *machineCPU* | 0,0002 | **0,0000** | **0,0000** | 0,0014 | 0,0037 |
| *qualitative_bankruptcy* | **0,0000** | **0,0000** | 0,0003 | **0,0000** | 0,0002 |
| *SWD* | 0,0007 | **0,0005** | 0,0009 | 0,0015 | 0,0020 |
| *windsorhousing* | 0,0005 | **0,0000** | **0,0000** | **0,0000** | 0,0030 |
| *wisconsin* | **0,0000** | **0,0000** | **0,0000** | **0,0000** | **0,0000** |
| *Media:* | **0,0006** | 0,0007 | **0,0006** | 0,0014 | 0,0028 |




(a) Efecto del ruido en la Precisión. (b) Efecto del ruido en el MAE. (c) Efecto del ruido en el NMI.

Figura 1: Rendimiento de Puro MonF$k$NN y M$k$NN con los diferentes conjuntos ruidos de *Artiset*.

### *V-C. Sobre la robustez al ruido monotónico de k-Vecinos más Cercanos Difuso Monotónico*

Con este último estudio empírico, pretendemos probar la robustez de MonF$k$NN frente a la presencia de violaciones monótonas o ruido en los sets de entrenamiento en contraste con M$k$NN. Para ello, se ha introducido diferentes cantidades de instancias ruidosas en las particiones de entrenamiento del conjunto de datos artificial *Artiset*. Luego, se han comparado el comportamiento de MonF$k$NN y M$k$NN en términos de precisión, MAE y NMI, mientras que el ratio de ruido aumenta.

La figura 1 muestra el comportamiento de MonF$k$NN y M$k$NN (líneas más oscuras y más claras, respectivamente) en base a la precisión (1a), MAE (1b) y NMI (1c), con la progresión del ruido. Como era de esperar, mientras la cantidad de ruido aumenta, el rendimiento de ambos métodos empeora. Sin embargo, hay alguna gran diferencia entre ambos clasificadores.

En primer lugar, el comportamiento de cara al ruido de MonF$k$NN es claramente mejor que el del M$k$NN en todos los aspectos probados. Las líneas negras siempre están sobre las más claras de la figura 1a, lo que indica siempre una precisión mayor, y debajo de ellas en las figuras 1b y 1c, lo que significa mejores MAE y NMI para MonF$k$NN. Por lo general, la distancia entre ambos métodos es amplia, con excepción de los resultados de NMI obtenidos para los valores más pequeños de ruido. Además, mientras que el índice de ruido aumenta, sus diferencias también aumentan.

Con estos experimentos, MonF$k$-NN ha demostrado una fuerte robustez al ruido monotónico preservando su buen rendimiento en términos de precisión, costes de error y monotonicidad.

## VI. Conclusiones

En este trabajo, se ha propuesto un modelo difuso de $k$-Vecinos más Cercanos para la clasificación con restricciones monotónicas. La transferencia de una función de probabilidades a una clase final se ha adaptado para que respete dichas restricciones. Se ha diseñado mecanismos para reducir la influencia de las violaciones monotónicas. Como demostración de su flexibilidad, se han presentados dos configuraciones diferentes del modelo con comportamientos muy distintos.

Durante el análisis experimental, se ha mostrado el gran potencial en relación a la monotonicidad de la configuración puramente monotónica y la buena precisión de la versión aproximada de MonF$k$NN. En comparación con otros métodos, MonF$k$NN es significativamente mejor en términos de precisión y coste de error, igualando los mejores resultados de NMI. Además, ha demostrado una fuerte robustez a grandes cantidades de ruido preservando su buen rendimiento.

## Referencias

[1] C.-C. Chen and S.-T. Li, "Credit rating with a monotonicity-constrained support vector machine model," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7235–7247, 2014.

[2] J.-R. Cano, N. R. Aljohani, R. A. Abbasi, J. S. Alowidbi, and S. Garcia, "Prototype selection to improve monotonic nearest neighbor," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 128–135, 2017.

[3] A. Ben-David, "Monotonicity maintenance in information-theoretic machine learning algorithms," *Machine Learning*, vol. 19, no. 1, pp. 29–43, 1995.

[4] W. Duivesteijn and A. Feelders, "Nearest neighbour classification with monotonicity constraints," in *ECML/PKDD (1)*, 2008, pp. 301–316.

[5] S. Lievens, B. De Baets, and K. Cao-Van, "A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting," *Annals of Operations Research*, vol. 163, no. 1, pp. 115–142, 2008.

[6] A. Ben-David, "Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications," *Decision Sciences*, vol. 23, pp. 1357–1372, 1992.

[7] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985.

[8] J. Derrac, S. García, and F. Herrera, "Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects," *Information Sciences*, vol. 260, pp. 98–119, 2014.

[9] W. Kotłowski and R. Słowiński, "On nonparametric ordinal classification with monotonicity constraints," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2576–2589, 2013.

[10] H. Levy, *Stochastic dominance: Investment decision making under uncertainty*. Springer, 2015.

[11] S. Lievens and B. De Baets, "Supervised ranking in the weka environment," *Information Sciences*, vol. 180, no. 24, pp. 4763–4771, 2010.

[12] I. Triguero, S. González, J. M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M. J. del Jesús, L. Sánchez, and F. Herrera, "Keel 3.0: an open source software for multi-stage analysis in data mining," *International Journal of Computational Intelligence Systems*, vol. 10, no. 1, pp. 1238–1249, 2017.

# Eventos raros, anomalías y novedades vistas desde el paraguas de la clasificación supervisada

Ander Carreño
Intelligent Systems Group (ISG)
UPV/EHU
P. Manuel Lardizabal 1
20018 San Sebastián (Spain)
Email: ander.carreno@ehu.eus

Iñaki Inza
Intelligent Systems Group (ISG)
UPV/EHU
P. Manuel Lardizabal 1
20018 San Sebastián (Spain)
Email: inaki.inza@ehu.eus

Jose A. Lozano
Intelligent Systems Group (ISG)
UPV/EHU, 28018 San Sebastián (Spain)
Baque Center For Applied Mathematics
BCAM, 48009 Bilbao (Spain)
Email: ja.lozano@ehu.eus

***Resumen*—En los últimos años, muchas áreas han contribuido a resolver problemas denominados: eventos raros (*rare events*), anomalías (*anomalies*), novedades (*novelties*) y datos atípicos (*outliers*). Estas áreas han creado una mezcla entre términos y problemas. Por un lado, problemas similares han sido descritos con diferentes términos. Por el otro, el mismo término ha sido utilizado para describir problemas diferentes. Dada esta confusión, este artículo tiene como objetivo poner orden en el área viendo todos estos problemas desde el prisma de la clasificación supervisada. En concreto, se asocia a cada término un escenario de clasificación, se muestran ejemplos, y se consideran aquellas características asociadas a cada término que comparten la mayoría de artículos de la literatura. Por lo tanto, los escenarios se formalizan desde el punto de vista de la clasificación supervisada, se revisan los objetivos de la tarea de clasificación, las métricas usadas para evaluar el rendimiento, las características de los datos de entrada y las técnicas utilizadas en cada uno de ellos.**

***Index Terms*—Eventos raros, anomalías, novedades, datos atípicos, outliers, aprendizaje automático, clasificación supervisada, aprendizaje desbalanceado.**

## I. Introducción

Una gran cantidad de aplicaciones requieren filtrar o detectar observaciones anormales en los datos. En algunas situaciones, las anormalidades son descritas como eventos raros, anomalías, novedades, outliers, excepciones, aberraciones, sorpresas, peculiaridades, ruido o contaminantes entre otras. De todas estas acepciones, las más comunes son: eventos raros, anomalías, novedades y outliers. La importancia de detectar estas anormalidades es debido a que estas pueden estar relacionadas con situaciones críticas o interesantes en una gran cantidad de dominios. Por ejemplo, en seguridad, las intrusiones son consideradas anormalidades [1]–[3]; en seguridad vial, los accidentes de tráfico son considerados anormalidades [4]; en geología la erupción de un volcán es una anormalidad [5]; en el control de alimentos, cuerpos extraños dentro de envoltorios son también anormalidades [6]; o en el caso de la neurociencia, un estímulo que no se ha experimentado anteriormente es también una anormalidad [7].

Considerando la importancia de las anormalidades, se ha desarrollado una larga lista de trabajos relacionados con estos términos. Sin embargo, se ha formado una mezcla entre los términos y los problemas. En primer lugar, se han utilizado diferentes términos para describir el mismo problema. Por ejemplo, en [8], los autores tratan un problema en el que predicen si se va a producir un desprendimiento en una zona concreta y en un periodo de tiempo acotado. Para ello, crean un mapa de zonas susceptibles a que ocurra un desprendimiento. Cada zona tiene asociado un riesgo de desprendimiento. Este riesgo se predice con un modelo aprendido con datos sobre zonas donde ha ocurrido un desprendimiento y zonas en las que no. Es decir, el modelo se aprende con datos etiquetados en dos categorías: desprendimiento y no desprendimiento. Estos desprendimientos de tierra son descritos como *eventos raros*, denotando que son eventos que ocurren pocas veces. No obstante, en [1], se trata un problema similar desde el punto de vista de la clasificación supervisada, pero que se describe con otro término. En éste, se realiza un estudio en la industria ferroviaria. Las puertas de pasajeros cuentan con diferentes sistemas para abrir y cerrar las puertas de los trenes, que aseguran el confort y la seguridad de los pasajeros abordo. En algunas situaciones, por el deterioro de estos sistemas, el sistema de apertura y cierre de las puertas falla. Por lo tanto, la tarea de predicción se enfoca en predecir si la puerta va a fallar en un periodo de tiempo acotado. Para ello, se aprende un modelo con datos etiquetados en dos categorías: fallos (puertas que fallan) y situaciones normales. En este estudio, los errores en las puertas se denominan *anomalías*, denotando también situaciones que ocurren pocas veces. Como se puede ver, los dos problemas son prácticamente idénticos desde el punto de vista de la clasificación supervisada, pero se han utilizado diferentes términos para describir las anormalidades. Por lo tanto, analizando los problemas desde el punto de vista de la clasificación supervisada, ambos problemas se pueden modelar como problemas de clasificación binarios con distribuciones de probabilidad desbalanceadas en las clases.

Por otro lado, también el mismo término ha sido utilizado para describir problemas diferentes. En los siguientes dos problemas, los autores utilizan el término *novedad*, *novelty* en inglés, para describir la anormalidades. En [3], detectan si un paciente está sufriendo un ataque epiléptico. Dada la fuerza de las convulsiones que sufren estos pacientes durante un ataque, pueden hacerse daño a ellos mismos. En consecuencia, detectar estos ataques lo más rápido posible es muy importante

para evitar los daños. Por lo tanto, el objetivo de este estudio es detectar si un paciente está sufriendo un ataque epiléptico. Para ello, se aprende un modelo predictivo con datos de movimientos de pacientes recogidos con un acelerómetro 3D. Estos datos previamente han sido grabados durante varios días en los cuales, en algún momento, los pacientes han sufrido una o varias convulsiones. Después, estos datos son segmentados en ventanas de tiempo fijas y, estas particiones son etiquetadas como ataque (porque ha sufrido un ataque en este tramo temporal) o como normal. Por lo tanto, cuando llega un nuevo caso, es decir, al recibir los datos de un paciente que está constantemente monitorizado, el clasificador detecta si está o no sufriendo un ataque epiléptico (i.e. identificado como novedad). Sin embargo en [6], se detectan cuerpos extraños dentro de contenedores de alimentos. En algunas ocasiones, ciertos cuerpos extraños como insectos, piedras o plásticos son encontrados dentro de los envoltorios de alimentos, tales como bandejas o tetrabriks. Detectar estos cuerpos extraños es muy importante para la imagen de las empresas productoras de este tipo de productos y para los consumidores finales. Por lo tanto, partiendo únicamente de imágenes de rayos X de alimentos limpios de cuerpos extraños, es decir, desde nuestro prisma, de una única clase, se aprende un modelo que es capaz de detectar estos cuerpos extraños, que son descritos como novedades en este estudio. Como se puede apreciar en ambos ejemplos, el mismo término (novedad) ha sido utilizado para describir las anormalidades. No obstante, ambos trabajos no se basan en el mismo escenario de aprendizaje, y en [3] el modelo predictivo es aprendido con tanto instancias normales como anormales mientras que en [6] únicamente se aprende el modelo con datos normales (sin cuerpos extraños).

Sin embargo, no sólo los problemas descritos anteriormente, sino la mayoría de los problemas reseñados con los términos eventos raros, anomalías y novedades, se pueden formalizar como problemas de clasificación supervisada. Por lo tanto, este artículo trata de poner orden en esta mezcla y confusión de términos y problemas, asignando un único escenario de aprendizaje a cada uno de los términos, que viene a ser el más compartido en la literatura. Después, describe cada uno de los escenarios de clasificación mediante la asignación de aquellas características que más comparten los problemas de la literatura asociados a cada término.

El artículo se organiza de la siguiente manera. Cada sección corresponde a un escenario de clasificación, siendo: la Sección II para la detección de eventos raros, la Sección III para la detección de anomalías y la Sección IV para las novedades. En la Sección V se discute sobre el término outlier y, finalmente, en la Sección VI se exponen las conclusiones del artículo.

## II. Detección de Eventos Raros

La mayoría de los artículos que utilizan el término evento raro para describir las anormalidades tienen la dimensión temporal como una característica clave. Por ejemplo en [4], se trata un problema relacionado con la seguridad vial en la autopista Attica (Grecia). Los autores dividen la autopista en diferentes secciones en las que quieren detectar si ha ocurrido un accidente en la última hora. Para ello, crean un modelo basándose en información histórica tanto de sensores terrestres como de cámaras de tráfico en el que conocen si ha ocurrido un accidente en las diferentes secciones y en intervalos de 1 hora. Por lo tanto, llegada una nueva información de 1 hora en una sección, el clasificador predice la existencia de un accidente. Otra aplicación relacionada con el término evento raro, es [5]. En este artículo se realiza un estudio sobre la erupción de dos volcanes de Sudamérica. Los autores quieren detectar si estos volcanes van a entrar en erupción en una ventana de tiempo acotada. Para ello, aprenden un modelo con datos históricos de actividad volcánica en el que eventualmente hay erupciones. Esta estimación la realizan mediante un proceso de Poisson en el que se predice si va a haber o no una erupción en un horizonte acotado. Otro ejemplo diferente relacionado es [9]. En este artículo se realiza un estudio sobre la probabilidad de fallo en cascada de una red eléctrica. Para ello, los autores realizan un estudio por simulación numérica en el que someten a la red a diferentes situaciones climatologícas extremas de manera intencionada. Como puede observarse, la dimensión temporal de todos estos problemas es clave. Sin embargo, se observan dos objetivos diferentes en todos estos trabajos. Por un lado, la estimación de la probabilidad de ocurrencia de un evento raro. Por otro, la detección de eventos raros en una ventana de tiempo fija.

En el caso de la estimación de la probabilidad de ocurrencia de un evento raro, la probabilidad es estimada por simulación. Este tema se ha tratado especialmente en la física y la ingeniería. Ejemplos de aplicaciones de estimación de la probabilidad de un evento raro son: la estimación de fallo de una infraestructura [9], la probabilidad de fallo de sistemas técnicos [10], o la probabilidad de desarrollos climáticos extremos [11].

En la clasificación de eventos raros, las instancias son series temporales [12]. El objetivo es clasificar nuevas series temporales utilizando un modelo aprendido anteriormente. Sin embargo, dada la temporalidad de las instancias, existen dos aproximaciones de clasificación diferentes. En primer lugar, se encuentra la clasificación de eventos raros completos, es decir, de series temporales completas. Por ejemplo, en [13], se hace uso del conjunto de datos SMART [1] para predecir si el disco duro va a fallar en un periodo de tiempo acotado. Los datos consisten en una colección de medidas tomadas a lo largo del tiempo por sensores del disco duro cuando este está en uso. En [14], se hace uso de un conjunto de datos de tecnología térmica en el que se ha almacenado información sobre sistemas de calefacción. El objetivo es predecir si el sistema de calefacción está fallando o no. En segundo lugar, la clasificación temprana de eventos raros se ha tratado en la literatura [15] (*early classification*). El objetivo es clasificar las nuevas observaciones lo antes posible, preferiblemente, antes de que se complete la serie temporal. Esta temprana clasificación es crítica en una variedad de aplicaciones. Por ejemplo, en

[1]El conjunto de datos SMART se compone de datos de uso de discos duros en los que se pueden ver errores como comportamiento normal.

[8] se realiza una predicción temprana de desprendimientos de tierra usando imágenes por satélite. O en [5], donde basándose en erupciones volcánicas del pasado, se predice si un volcán va a entrar en erupción o no en una ventana de tiempo fija.

De acuerdo a las distribuciones de probabilidad de los eventos raros ($\mathcal{A}$) y normales ($\mathcal{N}$), asumiendo que los datos son generados por el mecanismo generador $P(\mathbf{x}, c)$ [16], $P(C = \mathcal{A}) \ll P(C = \mathcal{N})$, los datos cuentan con una distribución desbalanceada de las clases. Por ello, la clasificación de eventos raros se puede formalizar bajo el marco de clasificación supervisada de series temporales con un (alto) desbalanceo de probabilidad sobre las clases [17], [18].

Sin embargo, en algunos problemas de clasificación de eventos raros, las instancias se transforman de tal forma que no se tiene en cuenta la información temporal en el proceso. Por esto, la clasificación de eventos raros se puede convertir en un problema de clasificación atemporal desbalanceado [4], [8].

Para evaluar el rendimiento de los clasificadores en la detección de eventos raros, métricas populares como el AUC [14], [19], [20] y el *recall* de la clase asociada al evento raro [14], [20] son utilizadas en la literatura.

### II-A. *Características de los datos de entrada*

En la mayoría de los artículos referidos a la detección de eventos raros, los datos están etiquetados en dos categorías: normal ($\mathcal{N}$) y rara ($\mathcal{A}$). Las instancias son series temporales, una secuencia ordenada de pares (tiempo, valor) de longitud fija $N$; $TS = \{(t_i, x_i), i = 1, \ldots, N\}$ [12]. Tanto en el escenario de clasificación de eventos raros, como en el escenario de detección de anomalías, existe una distribución desbalanceada de las clases donde $P(C = \mathcal{A}) \ll P(C = \mathcal{N})$. En este escenario de clasificación, tanto las instancias normales como las anormales se encuentran en el conjunto de entrenamiento del modelo.

### II-B. *Técnicas*

Entre las técnicas utilizadas en la literatura para estimar la probabilidad de ocurrencia de un evento raro se encuentran: *importance sampling*, simulaciones de Monte Carlo [21], [22], *kriging* [22] o *first order reliability methods (FORM)* [23]. De acuerdo con la clasificación de eventos raros, la técnica *rare event logistic regression*, una adaptación de la regresión logística, ha sido explícitamente propuesta en este escenario de clasificación [4], [8], [20], [24]. No obstante, técnicas populares de clasificación han sido aplicadas con éxito, como son la discriminación por divergencia de *Kullback-Leibler* [19], las redes neuronales de memoria a corto y largo plazo [14], los algoritmos genéticos [25], *naïve Bayes* para múltiples instancias [13], los procesos de Poisson [5], los *support vector data regression surrogates* [26], las redes Bayesianas [27], las máquinas de soporte vectorial [28] o el *AdaBoost* [29].

Por otro lado, teniendo en cuenta la distribución desbalanceada de las clases, se han encontrado técnicas en la literatura que tratan con este tipo de situaciones, como son *Structure Preserving Over Sampling (SPO)* [18] o una técnica de muestreo similar al SMOTE [17].

## III. Detección de Anomalías

En el escenario de detección de anomalías, comúnmente los datos son atemporales y etiquetados. El objetivo es detectar anomalías basándose en un modelo aprendido con datos históricos. El conjunto de entrenamiento se compone de tanto instancias normales como anormales (anomalías). Nótese que se espera que las instancias anómalas nuevas (no vistas) sean parecidas a aquellas provistas en el conjunto de entrenamiento del modelo. Por lo tanto, este escenario puede formalizarse bajo el marco de la clasificación supervisada con un (alto) desbalanceo de las clases, asumiendo que los datos se generan por un mecanismo generador $P(\mathbf{x}, c)$ [16], $P(C = \mathcal{A}) \ll P(C = \mathcal{N})$. Esta distribución desbalanceada de los datos hace que los clasificadores estándar estén abrumados por la clase mayoritaria (anormal) e ignoren la minoritaria (normal) [30].

Para evaluar el rendimiento de los clasificadores en este escenario, dado el (alto) desbalanceo de las clases, métricas comunes como el *accuracy* no son lo suficiente representativas. Por ello, focalizándose en la correcta clasificación de las anomalías; o lo que es lo mismo, minimizando el error de tipo II, una métrica de evaluación popular utilizada en la literatura es el *recall* de la clase minoritaria [1], [31]. Sin embargo, otras métricas de evaluación pueden ser utilizadas como las que se revisan en [32].

### III-A. *Características de los datos de entrada*

En la detección de anomalías, los datos son atemporales y, en la mayoría de las aplicaciones, están etiquetados en dos categorías: normal ($\mathcal{N}$) y anómalo ($\mathcal{A}$). En este escenario de clasificación existe una distribución desbalanceada de estas dos categorías, por lo que se tienen muchas más instancias de la clase normal que de la anormal. Sin embargo, en el conjunto de entrenamiento, se tienen tanto instancias de la clase normal como de la clase anómala.

### III-B. *Técnicas*

Entre las técnicas especificas creadas bajo el entorno de detección de anomalías, se encuentra el popular *Isolation Forest* [33]. Por otro lado, se han adaptado técnicas de clasificación para resolver problemas en este ámbito, como son: las máquinas de soporte vectorial [34], los *random forests* [35], los clasificadores Bayesianos [36], las redes neuronales [37], o los modelos basados en mixturas de gausianas [38]. Nótese que dado que la detección de anomalías se puede formalizar como un escenario de clasificación supervisada con distribuciones de probabilidad (altamente) desbalanceadas, es posible utilizar técnicas específicas para el desbalanceo, como puede ser el *Synthetic Minority Oversampling Technique* (SMOTE) [39], [40].

## IV. Detección de Novedades

En detección de novedades, comúnmente las instancias son atemporales y durante la etapa de entrenamiento, no se proporcionan instancias anormales [2], [6], [7]. El objetivo es clasificar si una nueva instancia pertenece al conjunto

normal provisto en entrenamiento o no, basándose en un modelo aprendido únicamente con instancias de una clase. Después, el modelo debe ser capaz de detectar cualquier grupo de instancias discordantes que es posible que formen una nueva clase [41]. Nótese que quizá sea necesaria la intervención de un experto para poder consolidar esta última anotación de una nueva clase. Por ejemplo, en este estudio de neurociencia [7], una novedad se considera un estímulo que no ha sido experimentado anteriormente. Este nuevo estímulo puede compartir características con otros nuevos estímulos no vistos en momento de entrenamiento, formando una clase o un nuevo tipo de estímulo. En el control alimenticio, los objetos extraños dentro de los envoltorios de los alimentos, como las piedras, los plásticos o insectos, son considerados novedades [6]. En este caso, es posible que las piedras de un alimento tengan características parecidas a otras observaciones no vistas en el conjunto de entrenamiento que contenga también una piedra.

En la detección de novedades, el conjunto de entrenamiento es generado únicamente por $P(\mathbf{X}|C = \mathcal{N})$. Consecuentemente, no se posee conocimiento alguno sobre las novedades cuando se aprende el primer modelo. Por lo tanto, en la creación del modelo, se debe asumir que las novedades estarán uniformemente distribuidas en el espacio de características. Esto es razonable dado que al no poseer ninguna información a priori, cualquier suposición es inconsistente. Sin embargo, una vez que se reconocen algunas novedades, estas pueden formar una clase nueva, por lo que el problema puede pasar a ser un problema de clasificación supervisada con distribuciones de probabilidad de clases desbalanceadas [41]. Para clarificar este último concepto, considérese un estudio médico en el que se proporcionan electrocardiogramas asociados a una enfermedad concreta como conjunto de entrenamiento. En la etapa de predicción, ciertos electrocardiogramas muestran una discordancia notable con aquellos de la etapa de entrenamiento. Por lo tanto, el clasificador los etiqueta como novedades. No obstante, se sabe que estos electrocardiogramas marcados como novedad, están relacionados con otra enfermedad. Por ello, es razonable pensar que estos electrocardiogramas comparten características entre ellos y que por tanto, crear una nueva clase para representar a los mismos sería una buena aproximación.

Por otro lado, también se ha tratado la detección de novedades en un entorno dinámico o *stream*. Por ejemplo en [41], partiendo de un conjunto de clasificación supervisada, los autores detectan nuevas clases combinando técnicas de clasificación supervisada y no supervisada. En concreto, si las instancias etiquetadas como novedades son suficientemente densas y distintas a las demás, estas son etiquetadas como una nueva clase.

Finalmente, para evaluar el rendimiento de estos clasificadores, es necesario señalar que la clasificación se centra principalmente en detectar las instancias normales proporcionadas en momento de entrenamiento [3], [33], [42]. Por lo tanto, las métricas de evaluación se basan en la minimización del error de tipo I. Una medida popular es la maximización del *recall* de la clase normal [3], [42].

### *IV-A. Características de los datos de entrada*

En el escenario de detección de novedades, no se provee de instancias nóveles en el conjunto de entrenamiento. El modelo es aprendido únicamente con instancias de una clase. Se espera que en momento de predicción puedan llegar tanto instancias de una como de otra (novel) clase. Sin embargo, también se espera una distribución de probabilidad desbalanceada sobre las clases normal y novel $P(C = \mathcal{N}) \ll P(C \neq \mathcal{N})$ [7]. Además, se asume uniformidad sobre la distribución de las instancias novel.

### *IV-B. Técnicas*

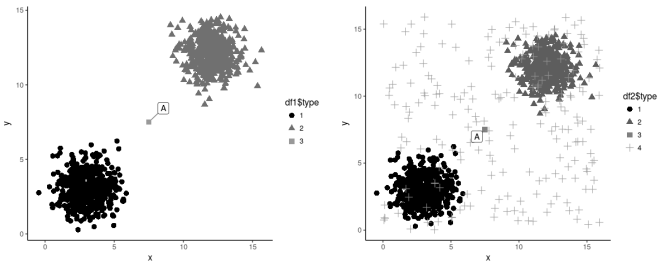
Las técnicas especialmente diseñadas para la detección de novedades tienen en cuenta que aprenden únicamente de una clase y son capaces de discernir si una nueva instancia forma parte o no de este comportamiento. Por lo tanto, las técnicas más representativas de este escenario de clasificación son aquellas relacionadas con el marco *one class*. Por ejemplo, *one class SVM* [28], [43], [44], *K-Nearest Neighbors data description* [45], *graph embeded one-class classifiers* [46] y *one-class random forests* [47] han sido propuestas en la literatura en escenarios de detección de novedades.

## V. Discusión: Detección de Outliers

El término dato atípico, *outlier* en inglés tiene características comunes a los demás términos. Por ejemplo, las instancias pueden ser tanto temporales (series temporales) [48] como atemporales [49]–[51].

No obstante, aunque similar, el término outlier tiene diferencias clave con el resto de términos analizados. La diferencia más significativa es que los datos no están etiquetados, es decir, la tarea de predicción se realiza bajo el marco de la clasificación no supervisada [49]. Por lo tanto, en contraposición al resto de escenarios, no se asume que al menos existen dos clases en el problema de predicción. En ocasiones, el término outlier también se ha relacionado con el *ruido* sugiriendo que estas observaciones pueden ser inconsistentes o incorrectas [49]. Por ejemplo, cuando se cometen errores humanos al recoger los datos, estos comportamientos erráticos son considerados outliers [52]. En cambio, en otras situaciones, el término outlier se ha relacionado con instancias con alta varianza [51], [53]. Por ejemplo en [51], los autores detectan jugadores *all-star* de la NBA en un conjunto de datos constituido por jugadores de baloncesto entre los años 1973 y 2003. Los jugadores excepcionales son considerados outliers. Para detectarlos, los autores realizan un *clustering* y aquellos puntos que se desvían significativamente son considerados outliers.

Sin embargo, la detección de los outliers es subjetiva [49]. ¿Cuánto debe desviarse una observación del resto para considerarla como atípica? En la literatura existen múltiples criterios para responder a esta pregunta [51], [54], [55]. Para clarificarla, en la Figura 1 se describen dos situaciones para la detección de outliers. En 1a el punto marcado con $A$ es considerado dato atípico dada la desviación significativa hacia uno y otro grupo. No obstante, cuando existe mucha más




(a) Sin ruido (b) Con ruido

Figura 1: Diferentes escenarios de outliers. Por [49].

Cuadro I: Resumen de las características de los diferentes escenarios de aprendizaje.

| | Eventos raros | Anomalías | Novedades | Outliers |
|---|---|---|---|---|
| Siempre relacionado con dominios temporales | ✓ | ✗ | ✗ | ✗ |
| Distribución desbalanceada de las clases | ✓ | ✓ | ✓ | - |
| Observaciones anormales durante la etapa de entrenamiento | ✓ | ✓ | ✗ | - |
| Se realiza predicción sobre clases previamente vistas | ✓ | ✓ | ✗ | ✗ |
| Clasificación supervisada | ✓ | ✓ | ✓ | ✗ |

dispersión en los datos, este concepto puede ser más difuso y complejo. Este comportamiento se puede ver en 1b. En 1b, dados los demás outliers alrededor del punto $A$, la densidad del punto $A$ es mayor y puede que quede difuminado el criterio de separación, haciendo una detección más difícil.

## VI. Conclusiones

Por un lado, los términos eventos raros, anomalías y novedades se han utilizado para describir problemas de clasificación similares en la literatura. Por el otro, problemas similares han sido descritos con diferentes términos. Además, estos términos se han utilizado indistintamente sin tener en cuenta las características que los diferencian. Por ello, en este artículo se han remarcado las características genuinas de cada uno de los términos. Para ello, se han extraído las características que más comparten los trabajos de la literatura referidos a cada término y se han revisado los objetivos, las características de los datos de entrada, y las técnicas utilizadas en los respectivos trabajos. Además, se ha argumentado que estos escenarios de aprendizaje pueden ser formalizados como problemas de clasificación supervisada con un (alto) desbalanceo entre las distribuciones de probabilidad de las clases. Consecuentemente, las técnicas específicas para tratar el desbalanceo pueden ser adaptadas y utilizadas en estos escenarios. En la Tabla I se muestra un resumen de las principales características de los diferentes escenarios de aprendizaje tratadas en este trabajo.

## VII. Agradecimientos

Este trabajo ha sido parcialmente financiado por el Gobierno Vasco (IT609-13) y por el Ministerio de Economía, Industria y Competitividad (TIN2016-78365-R). Ander Carreño posee una beca del Ministerio de Economía, Industria y Competitividad (BES-2017-080016). J.A. Lozano posee financiación tanto del programa BERC 2018-2021 (Gobierno Vasco) como del programa Severo Ochoa SEV-2017-0718 (Ministerio de Economía, Industria y Competitividad).

## Referencias

[1] R. P. Ribeiro, P. Pereira, and J. Gama, "Sequential anomalies: a study in the Railway Industry," *Machine Learning*, vol. 105, no. 1, pp. 127–153, oct 2016.

[2] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[3] S. Luca, D. A. Clifton, and B. Vanrumste, "One-class classification of point patterns of extremes," *Journal of Machine Learning Research*, vol. 17, pp. 1–21, 2016.

[4] A. Theofilatos, G. Yannis, P. Kopelias, and F. Papadimitriou, "Predicting Road Accidents: A Rare-events Modeling Approach," in *Transportation Research Procedia*, 2016.

[5] Y. Dzierma and H. Wehrmann, "Eruption time series statistically examined: Probabilities of future eruptions at Villarrica and Llaima Volcanoes, Southern Volcanic Zone, Chile," *Journal of Volcanology and Geothermal Research*, 2010.

[6] H. Einarsdóttir, M. J. Emerson, L. H. Clemmensen, K. Scherer, K. Willer, M. Bech, R. Larsen, B. K. Ersbøll, and F. Pfeiffer, "Novelty detection of foreign objects in food using multi-modal X-ray imaging," *Food Control*, vol. 67, pp. 39–47, sep 2016.

[7] A. Kafkas and D. Montaldi, "How do memory systems detect and respond to novelty?" *Neuroscience Letters*, feb 2018.

[8] M. Van Den Eeckhaut, T. Vanwalleghem, J. Poesen, G. Govers, G. Verstraeten, and L. Vandekerckhove, "Prediction of landslide susceptibility using rare events logistic regression: A case-study in the Flemish Ardennes (Belgium)," *Geomorphology*, vol. 76, no. 3-4, pp. 392–410, 2006.

[9] L. Dueñas-Osorio and S. M. Vemuru, "Cascading failures in complex infrastructure systems," *Structural Safety*, vol. 31, no. 2, pp. 157–167, mar 2009.

[10] T. Bedford and R. M. Cooke, *Probabilistic risk analysis: foundations and methods*. Cambridge University Press, 2001.

[11] S. Dessai and M. Hulme, "Does climate adaptation policy need probabilities?" *Climate Policy*, vol. 4, no. 2, pp. 107–128, jan 2004.

[12] P. Esling and C. Agon, "Time-series data mining," *ACM Computing Surveys*, vol. 45, no. 1, pp. 1–34, nov 2012.

[13] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado, "Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application," *J. Mach. Learn. Res.*, 2005.

[14] S. Zhang, S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Deep learning on symbolic representations for large-scale heterogeneous time-series event prediction," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, mar 2017, pp. 5970–5974.

[15] U. Mori, "Contributions to time series data mining departing from the problem of road travel time modeling," Ph.D. dissertation, University of the Basque Country, 2015.

[16] T. M. T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[17] S. Köknar-Tezel and L. J. Latecki, "Improving SVM classification on imbalanced time series data sets with ghost points," *Knowledge and Information Systems*, vol. 28, no. 1, pp. 1–23, jul 2011.

[18] H. Cao, X.-L. Li, Y.-K. Woon, and S.-K. Ng, "SPO: Structure Preserving Oversampling for Imbalanced Time Series Classification," in *2011 IEEE 11th International Conference on Data Mining*. IEEE, dec 2011, pp. 1008–1013.

[19] J. Xu, S. Denman, C. Fookes, and S. Sridharan, "Detecting rare events using Kullback-Leibler divergence: A weakly supervised approach," *Expert Systems with Applications*, 2016.

[20] Y. Ren, Y. Wang, X. Wu, G. Yu, and C. Ding, "Influential factors of red-light running at signalized intersection and prediction using a rare events logistic regression model," *Accident Analysis and Prevention*, 2016.
[21] M. Balesdent, J. Morio, and L. Brevault, "Rare Event Probability Estimation in the Presence of Epistemic Uncertainty on Input Probability Distribution Parameters," *Methodology and Computing in Applied Probability*, 2016.
[22] Y. Auffray, P. Barbillon, and J. M. Marin, "Bounding rare event probabilities in computer experiments," *Computational Statistics and Data Analysis*, 2014.
[23] D. Straub, I. Papaioannou, and W. Betz, "Bayesian analysis of rare events," *Journal of Computational Physics*, 2016.
[24] G. King, G. Langche Zeng, J. Fowler, E. Katz, M. Tomz for research assistance, J. Alt, J. Freeman, K. Gleditsch, G. Imbens, C. Manski, P. McCullagh, W. Mebane, J. Nagler, B. Russett, K. Scheve, P. Schrodt, M. Tanner, R. Tucker for helpful suggestions, S. Bennett, P. Huth, and R. Tucker, "Logistic Regression in Rare Events Data," *Political Analysis*, vol. 9, no. 2, pp. 137–163, 2001.
[25] G. M. Weiss and H. Hirsh, "Learning to Predict Rare Events in Event Sequences," *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, 1998.
[26] J.-M. Bourinet, "Rare-event probability estimation with adaptive support vector regression surrogates," *Reliability Engineering and System Safety*, 2016.
[27] S.-P. Cheon, S. Kim, S.-Y. Lee, and C.-B. Lee, "Bayesian networks based rare event prediction with sensor data," *Knowledge-Based Systems*, 2009.
[28] W. Khreich, B. Khosravifar, A. Hamou-Lhadj, and C. Talhi, "An anomaly detection system based on variable N-gram features and one-class SVM," *Information and Software Technology*, 2017.
[29] J. Wu, J. M. Rehg, and M. D. Mullin, "Learning a Rare Event Detection Cascade by Direct Feature Selection," *Neural Information Processing Systems (NIPS)*, vol. 16, 2003.
[30] N. V. Chawla, N. Japkowicz, and A. Ko, "Editorial: Special Issue on Learning from Imbalanced Data Sets," *Sigkdd Explorations*, vol. 6, no. 1, p. 1, 2004.
[31] A. Ogbechie, J. Díaz-Rozo, P. Larrañaga, and C. Bielza, "Dynamic Bayesian Network-Based Anomaly Detection for In-Process Visual Inspection of Laser Surface Heat Treatment," *Machine Learning for Cyber Physical Systems*, pp. 17–24, 2017.
[32] J. Ortigosa-Hernández, I. Inza, and J. A. Lozano, "Towards competitive classifiers for unbalanc classification problems: a study on the performance scores," 2016.
[33] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "iBAT: Detecting anomalous taxi trajectories from GPS traces," *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp '11)*, pp. 99–108, 2011.
[34] Y. Zhou, W. Su, L. Ding, H. Luo, and P. Love, "Predicting Safety Risks in Deep Foundation Pits in Subway Infrastructure Projects: Support Vector Machine Approach," *Journal of Computing in Civil Engineering*, vol. 31, no. 5, p. 04017052, 2017.
[35] S. Fan, G. Liu, and Z. Chen, "Anomaly detection methods for bankruptcy prediction," *2017 4th International Conference on Systems and Informatics (ICSAI)*, no. 17, pp. 1456–1460, 2017.
[36] N. A. Heard, D. J. Weston, K. Platanioti, and D. J. Hand, "Bayesian anomaly detection methods for social networks," *Annals of Applied Statistics*, vol. 4, no. 2, pp. 645–662, 2010.
[37] K. Noto, C. Brodley, and D. Slonim, "FRaC: a feature-modeling approach for semi-supervised and unsupervised anomaly detection." *Data mining and knowledge discovery*, vol. 25, no. 1, pp. 109–133, 2012.
[38] D. Reynolds, "Gaussian Mixture Models," in *Encyclopedia of Biometrics*. Boston, MA: Springer US, 2015, pp. 827–832.
[39] S. Miri Rostami and M. Ahmadzadeh, "Extracting Predictor Variables to Construct Breast Cancer Survivability Model with Class Imbalance Problem," *Shahrood University of Technology*, vol. 6, no. 2, pp. 263–276, jul 2018.
[40] M. Araujo, R. Bhojwani, J. Srivastava, L. Kazaglis, and C. Iber, "ML Approach for Early Detection of Sleep Apnea Treatment Abandonment," *Proceedings of the 2018 International Conference on Digital Health - DH '18*, pp. 75–79, 2018.
[41] M. M. Masud, Q. Chen, L. Khan, C. C. Aggarwal, J. Gao, J. Han, A. Srivastava, and N. C. Oza, "Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1484–1497, jul 2013.
[42] M. Swarnkar and N. Hubballi, "OCPAD: One class Naive Bayes classifier for payload based anomaly detection," *Expert Systems with Applications*, 2016.
[43] F. Dufrenois and J. C. Noyer, "One class proximal support vector machines," *Pattern Recognition*, vol. 52, pp. 96–112, 2016.
[44] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognition*, 2016.
[45] D. M. J. Tax, "One-class classification," 2001.
[46] V. Mygdalis, A. Iosifidis, A. Tefas, and I. Pitas, "Graph Embedded One-Class Classifiers for media data classification," *Pattern Recognition*, vol. 60, pp. 585–595, dec 2016.
[47] C. Désir, S. Bernard, C. Petitjean, and L. Heutte, "One class random forests," *Pattern Recognition*, vol. 46, no. 12, pp. 3490–3506, dec 2013.
[48] M. Gupta, J. Gao, and C. C. Aggarwal, "Outlier Detection for Temporal Data : A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 1–20, 2013.
[49] C. C. Aggarwal, *Outlier Analysis*. Springer International Publishing, 2017.
[50] L. Swersky, H. O. Marques, R. J. G. B. Campello, and A. Zimek, "On the Evaluation of Outlier Detection and One-Class Classification Methods," *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–10, 2016.
[51] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Reverse Nearest Neighbors in Unsupervised Distance-Based Outlier Detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 4347, no. OCTOBER, pp. 1–14, 2014.
[52] A. Barai and D. Lopamudra, "Outlier Detection and Removal Algorithm in K-Means and Hierarchical Clustering," *World Journal of Computer Application and Technology*, vol. 5, no. 2, pp. 24–29, 2017.
[53] X. H. Dang, I. Assent, R. T. Ng, A. Zimek, E. Schubert, Xuan Hong Dang, I. Assent, R. T. Ng, A. Zimek, and E. Schubert, "Discriminative features for identifying and interpreting outliers," *Proceedings - International Conference on Data Engineering*, pp. 88–99, mar 2014.
[54] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Kdd*, vol. 96, no. 34, pp. 226–231, 1996.
[55] S. Ramaswamy, R. Rastogi, and K. Shim KAIST, "Efficient Algorithms for Mining Outliers from Large Data Sets," *ACM Sigmod Record 2000*, vol. 29, no. 2, pp. 427–438, 2016.

# Una librería para el aprendizaje Multi-instancia Multi-etiqueta

Álvaro Belmonte, Amelia Zafra, Eva Gibaja, Sebastián Ventura
*Departamento de Informática y Análisis Numérico*
*Universidad de Córdoba*
Córdoba, España

***Resumen*—Este artículo presenta una librería para trabajar en la resolución de problemas de clasificación con múltiples instancias y múltiples etiquetas. Se describe el formato de datos, la arquitectura software, así como las diferentes propuestas algorítmicas que incorpora. La librería permite añadir nuevos algoritmos de forma sencilla, facilitando a los investigadores en esta área el desarrollo, prueba y comparación de nuevas propuestas. Además, es libre y de código abierto y está implementada en Java, usando las librerías Weka y Mulan. De este modo, los usuarios que trabajan tanto en el aprendizaje con múltiples instancias como en el aprendizaje con múltiples etiquetas, se encontrarán con un entorno de desarrollo con el que están familiarizados.**

***Index Terms*—multi-instance learning, multi-label learning, software**

## I. Introducción

A medida que los problemas de clasificación se vuelven más complejos, encontrar una representación adecuada de la información se convierte en una tarea cada vez más complicada. La experiencia demuestra que una representación precisa, que sea capaz de reflejar todas las relaciones e interacciones existentes en los datos, influye directamente en una resolución más efectiva del problema. En este contexto, el aprendizaje con múltiples instancias y múltiples etiquetas (*multi-instance multi-label learning*, MIML) se presenta como una alternativa prometedora que permite realizar una formulación natural que se adapta a las condiciones particulares de la representación de objetos complejos que suelen tener los problemas reales. Entre los problemas donde ha sido aplicado con éxito destacan la categorización de textos o imágenes [1]–[3], la detección de vídeo y audio [4] o la bioinformática [5], [6].

El aprendizaje MIML introduce una mayor flexibilidad en la representación de objetos, tanto en el espacio de entrada como en el de salida, debido a los dos paradigmas de aprendizaje que combina. Por un lado, la representación basada en el aprendizaje multi-instancia (*multi-instance*, MI) [7] ofrece una alternativa a la representación de instancias simples. En el aprendizaje MI un objeto o patrón, conocido habitualmente como *bolsa*, es representado mediante un conjunto variable de instancias, todas ellas con el mismo número de atributos. De este modo, un objeto puede ser representado con descripciones alternativas [7], con diferentes componentes [2], o bien mostrando su evolución en el tiempo [8]. Por otro lado, el aprendizaje multi-etiqueta (*multi-label*, ML) [9] introduce una mayor flexibilidad, esta vez en el espacio de salida, permitiendo que cada patrón tenga la posibilidad de pertenecer de forma simultánea a varias clases (etiquetas). Por ejemplo, para la resolución del problema de clasificación de imágenes, una imagen podría ser representada mediante múltiples instancias donde cada instancia se correspondería con diferentes regiones de la imagen, pudiendo a su vez tener diferentes etiquetas, tales como nubes, leones, y paisajes.

Actualmente, existen herramientas que permiten resolver problemas utilizando el aprendizaje MI y ML. Así, Weka [10] permite trabajar con problemas MI, y las librerías MEKA [11] y MULAN [12] permiten abordar el aprendizaje ML. No obstante, ninguna de ellas permite trabajar con el aprendizaje MIML. Hasta donde los autores conocemos, los únicos algoritmos públicamente disponibles para resolver problemas MIML han sido desarrollados por el grupo de investigación LAMDA [13]. Las principales limitaciones de estas implementaciones son que: están codificados en Matlab, con lo que es necesario disponer de licencia de este software para poder ejecutarlos y están formados por paquetes independientes, que habitualmente contienen la implementación de un único algoritmo que utiliza un formato de entrada y salida distinto al de otros paquetes, lo que dificulta la tarea de poder realizar un estudio experimental con ellos.

En este contexto, este artículo presenta una librería para aprendizaje MIML basada en las librerías Weka y MULAN, con lo que los investigadores en MI y ML, que empleen las librerías anteriores, estarán familiarizados con su estructura y formato. Entre sus características más relevantes se puede destacar: que utiliza un formato de datos diseñado específicamente para este aprendizaje, que cuenta con un conjunto de algoritmos desarrollados que directamente pueden ser ejecutados, que permite el uso de los algoritmos implementados para aprendizaje MI y ML de Weka y Mulan en un contexto MIML, que facilita el diseño y desarrollo de nuevos modelos que resuelvan problemas de clasificación con una representación MIML, que permite llevar a cabo un estudio experimental utilizando validación cruzada, y finalmente, su uso y ejecución resultan sencillos mediante la configuración de ficheros $xml$.

El resto del artículo se organiza como sigue. En la sección 2 se describe diferentes algoritmos MIML propuestos en la literatura. La sección 3, muestra la descripción de la librería, considerando el formato de los datos, su funcionalidad, su arquitectura y los principales paquetes y elementos de los que consta. En la sección 4, se describe algunos ejemplos de uso. Finalmente, la sección 5 describe las conclusiones más

relevantes del trabajo realizado.

## II. Trabajo Previo

Los algoritmos de clasificación desarrollados para aprendizaje MIML se pueden clasificar en dos enfoques [14]. Por un lado, algoritmos basados en una estrategia de transformación del problema MIML, y por otro lado, algoritmos que abordan el problema MIML directamente.

Debido a que tanto el aprendizaje MI como el aprendizaje ML son parte de MIML, se pueden usar como vía o puente para la transformación y posterior resolución de problemas MIML. Una primera aproximación consiste en aplicar alguna técnica de descomposición de etiquetas que transforme el problema MIML a un problema MI. Después de esto, puede ser utilizado cualquier algoritmo MI para resolver el problema. Otra aproximación consiste en emplear alguna técnica que permita unificar las diferentes instancias de una bolsa en una única instancia, transformando el problema MIML en un problema ML al que se le puede aplicar cualquier algoritmo de ML. En la literatura se pueden encontrar diferentes algoritmos que realizan una transformación del problema, entre ellos encontramos métodos basados en *ensembles* [15], [4], máquinas de vector soporte [5] y métodos basados en redes neuronales [1].

El rendimiento de los algoritmos anteriores puede verse limitado debido a la pérdida de información incurrida durante el proceso de simplificación/transformación. Idealmente, las conexiones entre instancias y etiquetas, así como la correlación entre las propias etiquetas deberían ser tenidas en cuenta. Por este motivo, también se han propuesto técnicas para abordar el problema MIML de forma directa basadas en redes neuronales [16], *ensembles* [6], máquinas de vector soporte [3] o vecinos más cercanos [2]. En [17] puede encontrarse una revisión más exhaustiva de algoritmos publicados en el aprendizaje MIML.

## III. Descripción de la librería

En esta sección se comentará el formato de los datos, los algoritmos y funcionalidades que se encuentran implementados, los paquetes de los que se compone y el formato de los ficheros de configuración para ejecutar cualquiera de sus algoritmos de clasificación.

### *III-A. Formato de los datos*

El formato diseñado para la representación de la estructura de un problema MIML está basado en los formatos de datos utilizados por Weka y MULAN. De este modo, un conjunto de datos estará representado mediante dos ficheros:

- Un fichero *xml* cuya función principal es identificar a los atributos del fichero *arff* que representan las etiquetas. De esta manera, las etiquetas no tienen por qué ser necesariamente los últimos atributos que se definan. Además, este fichero permite representar, anidando etiquetas, una jerarquía de clases. A continuación, se muestra un ejemplo con la definición de cuatro etiquetas.

```
<?xml version="1.0" encoding="utf-8"?>
<labels xmlns="http://mulan.sourceforge.net/labels">
  <label name="label1"></label>
  <label name="label2"></label>
  <label name="label3"></label>
  <label name="label4"></label>
</labels>
```

- Un fichero *arff* (*Attribute-Relation File Format*) cuya estructura presenta dos partes, cabecera y datos:
  - La **cabecera** contiene el nombre de la relación junto con una lista de atributos y sus tipos.
    - En la primera línea del fichero se encuentra la sentencia *@relation <relation-name>* que define el nombre del conjunto de datos. Es una cadena de caracteres que, si contiene espacios, debe de ir entrecomillada.
    - A continuación, se definen dos atributos: el primero de tipo nominal que identifica unívocamente a cada bolsa y el segundo atributo, relacional, que contiene los atributos que describen a las instancias. Para definir los atributos se utilizan las sentencias *@attribute <attribute-name><data-type>*. Habrá una línea por atributo:
      - Los tipos de dato numérico se especifican con *numeric*.
      - Si se trata de un conjunto posible de valores nominales debe ir especificado entre llaves y separado por comas. Por ejemplo: $\{valor_1, valor_2, ..., valor_N\}$.
    - Finalmente, se definen los atributos correspondientes a las etiquetas. Estos atributos tienen que ser de tipo binario (nominales con valores 0 y 1).
  - La sección de **datos** comienza con un *@data*. Cada una de las líneas siguientes representará una bolsa. Los atributos deben de estar ordenados según la declaración realizada en la cabecera. El valor de cada atributo se separa por comas y todas las filas deben tener el mismo número de columnas. Los decimales deben de separarse con el punto. El contenido del atributo relacional estará entrecomillado (comillas simples o dobles), separando cada instancia de la bolsa por un $\backslash n$. A continuación, se muestra un ejemplo de fichero *arff* donde cada bolsa está formada por tres atributos de tipo numérico y tiene cuatro etiquetas. La primera bolsa está formada por 3 instancias, y la segunda bolsa por 2 instancias.

```
@relation toy
@attribute id {bag1,bag2}
@attribute bag relational
  @attribute f1 numeric
  @attribute f2 numeric
  @attribute f3 numeric
@end bag
@attribute label1 {0,1}
@attribute label2 {0,1}
@attribute label3 {0,1}
@attribute label4 {0,1}
```

```
@data
bag1,"42,−198,−109\n42,−191,−142\n3,4,6",1,0,0,1
bag2,"12,−98,10\n42,−19,−12",0,1,1,0
```

La librería cuenta con las clases necesarias tanto para representar bolsas individuales (clase *Bag* del paquete *data*) como para representar un conjunto de datos completo (clase *MIMLInstances*). También proporciona medios para guardar un conjunto de datos en sus respectivos ficheros *arff* y *xml* y para obtener diferentes métricas que te permitan estudiarlos en más profundidad (*MIMLStatistics* en *data.statistics*).

### *III-B. Funcionalidad incluida en la librería*

La librería contiene métodos para transformar un conjunto de datos MIML a otro tipo de representación como MI o ML, algoritmos de clasificación MIML y clases para dar soporte al desarrollo de experimentos.

*III-B1. Métodos para transformar datos MIML:* se han incluido los siguientes métodos:

- Métodos para transformar a MI (formato de datos usado en Weka) [9]:
  - *Binary Relevance Transformation*: transforma un conjunto de datos MIML en tantos conjuntos de datos MI binarios como etiquetas tenga el problema.
  - *Label Powerset Transformation*: transforma un conjunto de datos MIML en uno multiclase en el que cada posible combinación de etiquetas del conjunto de datos original es considerado una clase diferente.
- Métodos para transformar a ML (formato de datos usado en MULAN) [18]:
  - *Arithmetic Transformation*: transforma cada bolsa en una única instancia donde el valor para cada atributo es su valor medio dentro de la bolsa.
  - *Geometric Transformation*: transforma cada bolsa en una única instancia donde el valor para cada atributo es el centro geométrico de sus valores máximo y mínimo dentro de la bolsa.
  - *Min-Max Transformation*: transforma cada bolsa en una única instancia que contiene, para cada atributo, sus valores mínimo y máximo dentro de esa bolsa. Cada instancia está definida por el doble de atributos de los que tenía previamente.

*III-B2. Algoritmos de clasificación:* se han incluido los siguientes algoritmos:

- *MIMLClassifierMI*: realiza una transformación del problema MIML para obtener un problema MI aplicando una transformación LP o BR. Posteriormente, obtiene un resultado resolviendo el problema con un algoritmo MI que se especifique. Al ser compatible con la librería Weka, en la tabla I se muestra un ejemplo de 15 algoritmos de Weka que podrían ser utilizados directamente este clasificador.
- *MIMLClassifierML*: realiza una transformación del problema MIML para obtener un problema ML aplicando una transformación aritmética, geométrica o min-max. Posteriormente, obtiene un resultado resolviendo el problema con un algoritmo ML que se especifique. Al ser compatible con la librería MULAN, en la tabla I se muestra un ejemplo de 15 algoritmos de MULAN que podrían ser utilizados directamente este clasificador.
- *MIML-kNN* [2]: utiliza los vecinos y referencias más cercanos a una bolsa para estimar las posibles clases a las que pertenece, trabajando directamente sobre datos MIML.

*III-B3. Otra funcionalidad:* la librería presenta un marco de trabajo para manejar conjuntos de datos MIML, obtener informes, calcular medidas de distancia, realizar experimentos utilizando validación cruzada o indicando los conjuntos de datos de entrenamiento y test, así como las clases e interfaces genéricas y necesarias para desarrollar nuevos clasificadores, que permite que el desarrollo de nuevas propuestas y la realización de un estudio experimental sea una tarea sencilla.

### *III-C. Arquitectura de la librería*

La librería, desarrollada en Java y de código abierto, hace uso de las librerías MULAN y Weka. Se encuentra estructurada en siete paquetes, algunos de ellos organizados en subpaquetes, de acuerdo con su funcionalidad:

- **core**: incluye diferentes funcionalidades que serán utilizadas por otras clases de la librería. Este paquete contiene las clases relacionadas con la configuración de los métodos utilizando ficheros *xml*.
- **core.distance**: contiene diferentes variantes de la distancia de *Hausdorff* para calcular la distancia entre bolsas.
- **data**: incluye clases para trabajar con el formato de datos detallado en la sección III-A. Entre las funcionalidades que soporta se encuentra: cargar en memoria un conjunto de datos, conocer propiedades de los datos como el número de instancias de una bolsa concreta, el número de atributos, el número de bolsas totales, el número de etiquetas, etc. Además, permite acceder tanto a una bolsa particular, como a sus instancias y etiquetas.
- **data.statistics**: incluye clases encargadas de obtener información relevante de los conjuntos de datos MIML. Se considera tanto información referente a MI (atributos por

Tabla I: Algoritmos compatibles

| Clasificadores MI (Weka) | Clasificadores ML (MULAN) |
|---|---|
| CitationKNN | BRkNN |
| MDD | DMLkNN |
| MIDD | IBLR |
| MIBoost | mLkNN |
| MIEMDD | HOMER |
| MILR | RAkEL |
| MINND | EPS |
| MIOptimalBall | ECC |
| MIRI | MLStacking |
| MISMO | BR |
| MISVM | LP |
| MITI | PS |
| MIWrapper | CC |
| SimpleMI | RPC |

bolsa, número medio de instancias por bolsa, número de instancias mínimo y máximo por bolsa, etc.), como a ML (frecuencia de los conjuntos de etiquetas, distribución de etiquetas por bolsa, co-ocurrencias entre etiquetas, etc.).

- **transformation.mimlTOml**: incluye los métodos para transformar un problema MIML en otro ML.
- **transformation.mimlTOmi**: incluye los métodos para transformar un problema MIML en otro MI.
- **mimlclassifier**: incluye las interfaces y las clases abstractas necesarias en el desarrollo de los algoritmos de clasificación para resolver problemas MIML.
- **mimclassifier.mimlTOmi**: incluye los algoritmos de clasificación que resuelven el problema MIML transformando previamente el problema a MI.
- **mimclassifier.mimlTOml**: incluye los algoritmos de clasificación que resuelven el problema MIML transformando previamente el problema a ML.
- **mimclassifier.lazy**: contiene el algoritmo MIMLkNN.
- **report**: incluye clases que permiten generar informes de resultados de los experimentos realizados. La librería incorpora un informe general con principales métricas que se pueden utilizar para evaluar un modelo MIML.
- **tutorial**: incluye ejemplos de uso de las diferentes funcionalidades de la librería: ejecutar y obtener los resultados de un algoritmo de clasificación MIML, transforma un conjunto de datos MIML a ML o MI, etc.
- **tutorial.config**: incluye ejemplos de ficheros de configuración para poder ejecutar y obtener resultados de los algoritmos de clasificación MIML incluidos en la librería.
- **miml**: incluye la clase para ejecutar cualquier clasificador de la librería mediante un fichero de configuración.

Todos los paquetes contienen las interfaces y las clases necesarias para extender su funcionalidad heredando e implementado la interfaz concreta. De este modo, es posible desarrollar de forma sencilla un nuevo método de transformación del conjunto de datos, y/o un nuevo algoritmo de clasificación. La figura 1 muestra un diagrama que representa las diferentes clases de la librería y sus relaciones.

### *III-D. Ficheros de configuración*

Para facilitar la ejecución de los diferentes algoritmos (ver la sección III-B), la librería cuenta con una clase principal *RunAlgorithm*, que junto con un fichero de configuración en formato *xml*, que especifica el clasificador que se va a utilizar, el conjunto de datos y el informe de salida que se desea obtener, ejecuta y obtiene los resultados de cualquier clasificador MIML que se desee emplear. Estos ficheros tienen que seguir el siguiente formato para que la librería, a partir de la clase *ConfigLoader*, sea capaz de leerlos y configurar todo el proceso que se debe de seguir, desde la creación del modelo hasta la generación de los informes con los resultados:

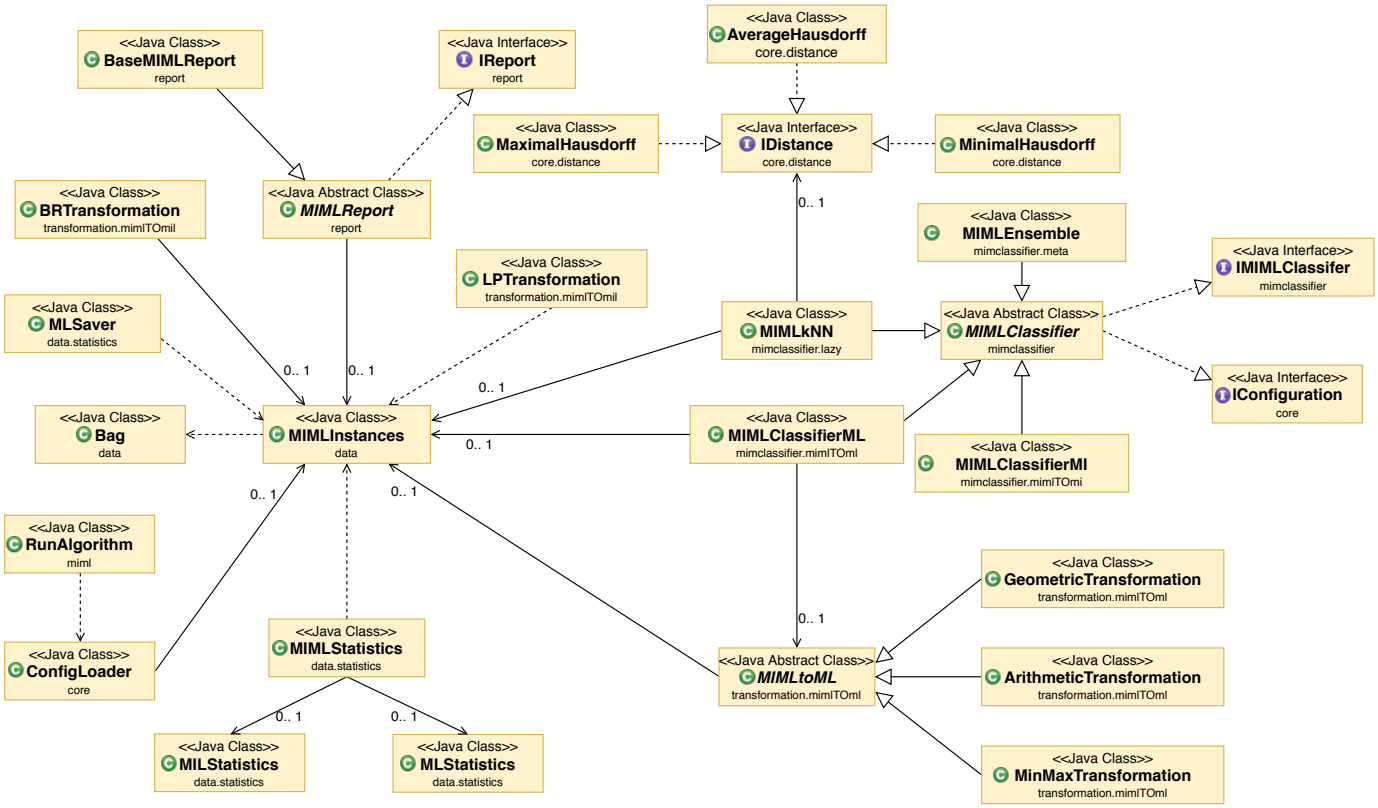



Figura 1: Diagrama de clases de la librería

```
<configuration>
  <classifier>  </classifier>
  <evaluator>  </evaluator>
  <report> </report>
</configuration>
```

Todos los ficheros deben empezar con el elemento *configuration*. En la etiqueta *classifier*, se especificará el clasificador de la librería que se quiere utilizar. Dentro del clasificador será donde se detalle su configuración: los parámetros específicos que necesite, el método de transformación a utilizar, y/o el clasificador MI o ML que se requiera en las versiones que transforman el problema. La etiqueta *evaluator* detallará el conjunto de datos a utilizar, y si se lleva a cabo una validación cruzada o se especifican directamente los ficheros *train/test*. Finalmente, la etiqueta *report*, indicará el informe que genera el fichero de salida. Esta clase puede ser extendida fácilmente para que los usuarios especifiquen el formato de salida más conveniente.

## IV. Ejemplos de uso

La experimentación mediante ficheros *xml* se puede llevar a cabo haciendo uso de la clase ejecutable *RunAlgorithm* pasándole como argumento, a través de la opción *-c*, la ruta del fichero de configuración. A continuación, se mostrarán ejemplos para un conjunto de casos ilustrativos.

### *IV-A. Ejecución de un algoritmo de clasificación MIML reduciendo el problema a MI*

El ejemplo de fichero *xml* para este caso es el siguiente:

```
<configuration>
  <classifier name="mimlclassifier.mimlTOmi.MIMLClassifierMI"
      >
    <multiInstanceClassifier name="weka.classifiers.mi.MISMO">
        <listOptions>-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V
            -1 -W 1 -K "weka.classifiers.mi.supportVector.
            MIPolyKernel -E 1.0 -C 250007"</listOptions>
        </multiInstanceClassifier>
        <transformMethod name="mulan.classifier.transformation.
            BinaryRelevance"/>
  </classifier>
  <evaluator method="train-test">
    <data>
      <trainFile>data/miml_03_data.arff</trainFile>
      <testFile>data/miml_04_data.arff</testFile>
      <xmlFile>data/miml_03_data.xml</xmlFile>
    </data>
  </evaluator>
  <report name="report.BaseMIMLReport">
      <fileName>results/results.csv</fileName>
    </report>
</configuration>
```

- El clasificador que se desea utilizar debe especificarse en el atributo *name* del elemento *classifier*. Se va a emplear el clasificador *MIMLClassifierMI*. Este clasificador necesita que se especifique en *transformMethod* el método de transformación que se va a utilizar para transformar el problema MIML en un problema MI. En el ejemplo se hace uso del método BR de la librería de Mulan.
- Como configuración de este algoritmo, se debe detallar también en la etiqueta *multiInstanceClassifier* el algoritmo MI que se utilizará una vez realizada la transformación. En este caso se ha utilizado MISMO, un clasificador de la librería Weka para problemas MI. En la etiqueta *listOptions* detallaremos los parámetros que se quieren utilizar en el algoritmo MISMO. Si no se detalla ninguno, se tomarán los valores por defecto establecidos para cada algoritmo.
- El atributo *method* de la etiqueta *evaluator* especifica el proceso de evaluación que se llevará a cabo.
- En el caso de que se vaya a realizar un método *train/test* es necesario especificar en la etiqueta *data*, que pertenece al *evaluator*, la ruta de los ficheros *arff* de entrenamiento y test, así como la del fichero *xml* con las etiquetas.
- Por último, en el elemento *report* se indicará el informe que se desea utilizar, y la configuración de cada informe. En el informe que se ha especificado en el ejemplo, solamente es necesario especificar en *fileName* la ruta donde se quiere guardar el informe de resultados.

### *IV-B. Ejecución de algoritmo degenerativo utilizando ML*

Para la realización de un experimento utilizando ML como vía, un ejemplo de fichero de configuración sería:

```
<configuration>
  <classifier name="mimlclassifier.mimlTOml.MIMLClassifierML"
      >
    <multiLabelClassifier name="mulan.classifier.lazy.MLkNN">
      <parameters>
        <classParameters>int.class</classParameters>
        <classParameters>double.class</classParameters>
        <valueParameters>1</valueParameters>
        <valueParameters>1.0</valueParameters>
      </parameters>
    </multiLabelClassifier>
    <transformMethod name="transformation.mimlTOml.
        ArithmeticTransformation"/>
  </classifier>
  <evaluator method="cross-validation">
    <numFolds>5</numFolds>
    <data>
      <file>data/miml_03_data.arff</file>
      <xmlFile>data/miml_03_data.xml</xmlFile>
    </data>
  </evaluator>
  <report name="report.BaseMIMLReport">
      <fileName>results/results.csv</fileName>
    </report>
</configuration>
```

- En este caso, el elemento *multiInstanceClassifier* es sustituido por *multiLabelClassifier*.
- En caso de que el clasificador ML que se utilice se pueda configurar, los parámetros se deben de especificar de la siguiente forma (para respetar la forma de ejecución que tiene establecida MULAN):
  - En primer lugar se pondrán tantos elementos *classParameters* como parámetros tenga el constructor del clasificador. Su contenido debe de ser el tipo al que

pertenecen dichos parámetros, tienen que estar en el orden en el que se declaren en el constructor.
    - A continuación, se especifican los valores que se quiere que tengan dichos parámetros, indicando cada uno de ellos en un elemento *valueParameters* diferente que, nuevamente, tienen que estar en orden.
- El clasificador que se ha utilizado (*MIMLClassifierML*) se adapta a todos los métodos de transformación disponibles en *transformation.mimlToml*, es necesario especificar en su configuración el elemento *transformMethod*.
- Este experimento se ha configurado para que realice una evaluación basada en validación cruzada, especificado en el atributo *method* la cadena *"cross-validation"*. Es necesario indicar el número de *folds* en el elemento *numFolds* y el fichero que contiene el conjunto de datos. La etiqueta *report* sería similar al ejemplo anterior.

### *IV-C. Ejecución de algoritmo basado en vecinos más cercanos*

Por último, se muestra un ejemplo de fichero *xml* para ejecutar el algoritmo MIMLkNN:

```
<configuration>
  <classifier name="mimlclassifier.lazy.MIMLkNN">
    <nReferences>2</nReferences>
    <nCiters>2</nCiters>
    <metric name="core.distance.AverageHausdorff">
    </metric>
  </classifier>
  <evaluator method="train-test">
    <data>
      <trainFile>data/miml_03_data.arff</trainFile>
      <testFile>data/miml_04_data.arff</testFile>
      <xmlFile>data/miml_03_data.xml</xmlFile>
    </data>
  </evaluator>
  <report name="report.BaseMIMLReport">
      <fileName>results/results.csv</fileName>
    </report>
```

- Con los algoritmos que trabajen directamente con el formato MIML tan solo es necesario especificar en su configuración los parámetros para su ejecución. En este ejemplo, el algoritmo MIML-kNN necesita configurar el número de referencias y citas con los que trabajará, así como la métrica que va a utilizar para medir la distancia entre las bolsas de un *data set*. Las etiquetas *evaluator* y *report*, serían similares a los ejemplos anteriores.

## V. Conclusiones

En este trabajo se presenta una librería Java basada en Weka y Mulan para el desarrollo, prueba y comparación de algoritmos dentro del marco de trabajo del aprendizaje MIML. Además de un conjunto de algoritmos MIML, de referencia en la temática, la estructura de la librería permite el desarrollo de nuevas propuestas de forma sencilla. Finalmente, se ha diseñado un formato de datos específico para MIML. La librería es sencilla de utilizar mediante el uso de ficheros de configuración *xml*.

## Agradecimientos

Este trabajo está financiado por el proyecto de investigación TIN2017-83445-P del Ministerio de Economía y Competitividad y el Fondo Europeo de Desarrollo Regional.

## Referencias

[1] K. Yan, Z. Li, and C. Zhang. A new multi-instance multi-label learning approach for image and text classification. *Multimedia Tools and Applications*, 75(13):7875–7890, 2016.

[2] M.L. Zhang. A k-nearest neighbor based multi-instance multi-label learning algorithm. In *Proceedings of the 22nd International Conference on Tools with Artificial Intelligence*, volume 2, pages 207–212, 2010.

[3] C. Tong-tong, L. Chan-juan, Z. Hai-lin, Z. Shu-sen, L. Ying, and D. Xin-miao. A multi-instance multi-label scene classification method based on multi-kernel fusion. In *Proceedings of the Conference on Intelligent Systems*, pages 782–787, 2015.

[4] X.S. Xu, X. Xue, and Z. Zhou. Ensemble multi-instance multi-label learning approach for video annotation task. In *Proceedings of the 19th International Conference on Multimedia*, pages 1153–1156. ACM, 2011.

[5] Y.X. Li, S. Ji, S. Kumar, J. Ye, and Z.H. Zhou. Drosophila gene expression pattern annotation through multi-instance multi-label learning. *Transactions on Computational Biology and Bioinformatics*, 9(1):98–112, 2012.

[6] J.S. Wu, S.J. Huang, and Z.H. Zhou. Genome-wide protein function prediction through multi-instance multi-label learning. *Transactions on Computational Biology and Bioinformatics*, 11(5):891–902, 2014.

[7] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31–71, 1997.

[8] S. Kotsiantis, D. Kanellopoulos, and V. Tampakas. Financial application of multi-instance learning: two greek case studies. *Journal of Convergence Information Technology*, 5(8):42–53, 2010.

[9] E. Gibaja and journal=Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery volume=4 number=6 pages=411–444 year=2014 publisher=Wiley Online Library Ventura, S. Multi-label learning: a review of the state of the art and ongoing research.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: An update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[11] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes. Meka: a multi-label/multi-target extension to weka. *Journal of Machine Learning Research*, 17(1):667–671, 2016.

[12] G. Tsoumakas, E. Spyromitros-Xioufis, Jozef Vilcek, and I. Vlahavas. Mulan: A java library for multi-label learning. *Journal Machine Learning Research*, 12:2411–2414, 2011.

[13] LAMDA Learning and Mining from DatA. http://lamda.nju.edu.cn/Data.ashx. Accessed: 2018-06-19.

[14] Z.H. Zhou, M.L. Zhang, S.J. Huang, and Y.F. Li. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320, 2012.

[15] Z.H. Zhou and M.L. Zhang. Multi-instance multi-label learning with application to scene classification. In *Proceedings of the Advances in neural information processing systems*, pages 1609–1616, 2006.

[16] C. Li and G. Shi. Weights optimization for multi-instance multi-label rbf neural networks using steepest descent method. *Neural Computing and Applications*, 22(7-8):1563–1569, 2013.

[17] F. Herrera, S. Ventura, R. Bello, C. Cornelis, A. Zafra, D. Tarragó, and S. Vluymans. *Multiple Instance Learning. Foundations and Algorithms*. Springer, 230 pages, 2016.

[18] Eibe Frank and Xin Xu. Applying propositional learning algorithms to multi-instance data. pages 1–13, *Computer Science Working Papers. University of Waikato, Department of Computer Science*. 2003.