ESCUELA POLITÉCNICA
SUPERIOR DE CÓRDOBA
Universidad de Córdoba

PROPUESTA DE TEMA DE TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Módulo Python de gestión de datos multi-instancia multi-etiqueta

Autor
Damián Martínez Ávila

DNI: 46271668S

Directora
Dª. Eva Lucrecia Gibaja Galindo

Octubre, 2023

UNIVERSIDAD D CORDOBA

CONTENIDO

1 INTRODUCCIÓN 3

2 ANTECEDENTES 5

3 OBJETIVOS 6

4 FASES DE DESARROLLO DEL PROYECTO 7

5 RECURSOS 8
5.1 Recursos Humanos 8
5.2 Recursos Software 8
5.3 Recursos Hardware 8

BIBLIOGRAFÍA 9

1. INTRODUCCIÓN

La clasificación es una tarea de aprendizaje supervisado que consiste en estudiar la correspondencia entre un conjunto de variables de entrada, X, y una variable de salida, Y, para predecir las salidas de datos no observados [1].

En la **clasificación tradicional** o convencional (ver Figura 1), cada patrón está representado por una única instancia con un número determinado de atributos y una sola etiqueta de salida. Si la etiqueta solo puede tomar los valores 0,1 hablamos de clasificación binaria, y en caso de que la etiqueta pueda tener varios valores (ej. *iris versicolor*, *iris setosa*, *iris virginica*), hablamos de clasificación multi-clase.

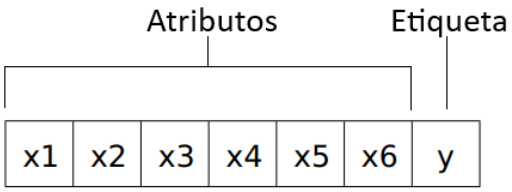

Figura 1: Ejemplo de patrón para clasificación convencional

Aunque la formalización anterior prevalece y tiene éxito, hay muchos problemas del mundo real que no se ajustan bien a esta representación, en los que un objeto del puede estar asociado a varias etiquetas binarias simultáneamente [2]. Este grupo de problemas representa un área conocida como **clasificación multi-etiqueta** (*multi-label*, ML)[3][4] (ver Figura 2). Dicha clasificación permite una mayor flexibilidad en la representación del espacio de salida.

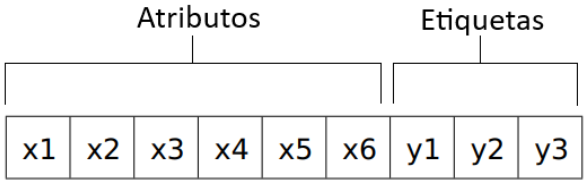

Figura 2: Ejemplo de patrón para clasificación ML

Por otro lado, la **clasificación multi-instancia** (*multi-instance*, MI) es una variante del aprendizaje supervisado que consiste en aprender un concepto a partir de bolsas de instancias positivas y negativas (ver Figura 3). Cada bolsa puede contener un número variable de instancias, todas ellas con el mismo número de atributos y generalmente se asume la hipótesis estándar de Dietterich que estable que una bolsa se etiqueta como positiva si alguna de las instancias que contiene corresponda es positiva y la bolsa se etiqueta como negativa si todas las instancias de la bola son

negativas [5]. Este marco de trabajo permite una mayor flexibilidad en la representación del espacio de entrada que el aprendizaje convencional.

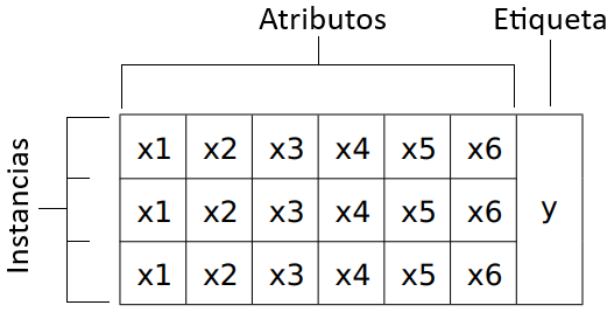

Figura 3: Ejemplo de patrón para clasificación MI

Finalmente, la **clasificación multi-instancia multi-etiqueta** (*multi-instance multi-label*, MIML) combina aprendizaje multi-etiqueta y multi-instancia para introducir una mayor flexibilidad tanto en la representación de la entrada, como en las salidas. Cada objeto es representado por una bolsa de instancias y se le permite tener múltiples etiquetas binarias de clase (ver Figura 4).

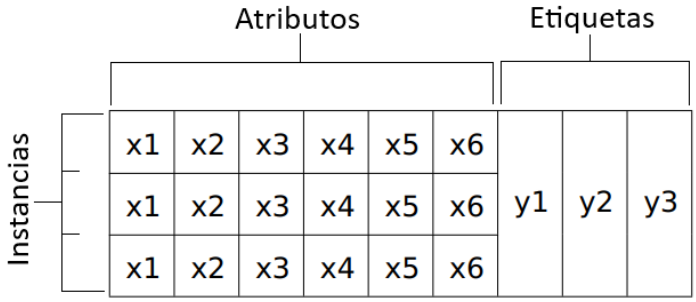

Figura 4: Ejemplo de patrón para clasificación MIML

Este aprendizaje permite formalizar objetos en diferentes problemas complejos. Por ejemplo, una imagen contiene de forma natural diferentes regiones (instancias) y la imagen completa puede representar diferentes clases, tales como *nubes*, *leones*, y *paisajes*. En categorización de textos, cada documento normalmente consiste en diferentes secciones o párrafos (instancias), y cada documento puede ser asignado a diferentes categorías, tales como *deportes*, *política*, y *ocio*.

Como el aprendizaje MIML se basa tanto en el aprendizaje MI como en el aprendizaje ML, uno de los enfoques más directos consiste en aplicar transformaciones a los datos MIML para obtener un problema MI o ML y resolver el problema transformado mediante algoritmos MI o ML (ver Figura **??**).

En este TFG se pretende profundizar en el marco de aprendizaje MIML desarrollando una librería en Python que permita trabajar con este tipo de problemas. En concreto nos centraremos en el manejo básico de conjuntos de datos (datasets) con información MIML y en desarrollar un marco de trabajo básico que pueda servir de punto de arranque para futuras ampliaciones de la librería.

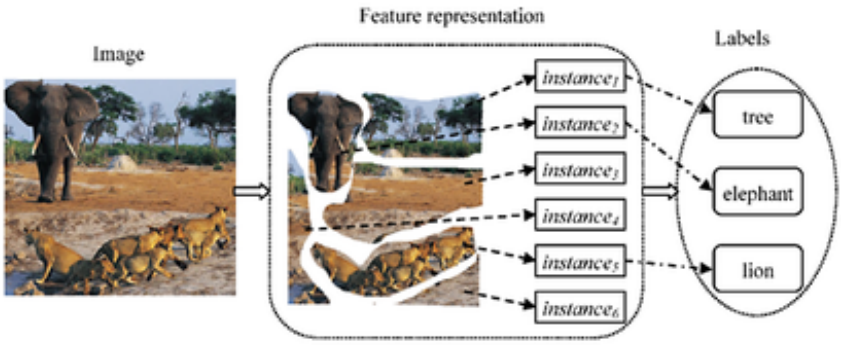

Figura 5: Clasificación MIML

2. ANTECEDENTES

Actualmente, existen herramientas que permiten resolver problemas utilizando el aprendizaje MI. Una de las de uso más extendido es Weka [6], basado en Java, que incluye un paquete específico con algoritmos de aprendizaje MI. También se han desarrollado librerías para resolver problemas utilizando el aprendizaje ML. Destacan Meka y Mulan [7], basadas en Weka y desarrolladas en Java, y `scikit-learn` [8], desarrollado en Python, que incluye algorimos básicos de clasificación ML.

Para resolver problemas MIML se han desarrollado algunas implementaciones en matlab por parte del grupo de investigación LAMDA [9] en forma de paquetes independientes para la ejecución de un único algoritmo. También hay que citar la librería MIML [10], escrita en Java, con funcionalidad para el tratamiento de datos MIML y el desarrollo y prueba de nuevos algoritmos MIML. Incluye 43 algoritmos de clasificación MIML, procedimientos de validación cruzada y de *train-test*, métricas estándar para la evaluación del rendimiento, así como la generación de informes.

El lenguje de programación Python, aun cuando fue creado como lenguaje de programación de uso general, se ha posicionado, junto con Java y R, como uno de los lenguajes más extendidos para aprendizaje automático. `scikit-learn` es una de las librerías más populares en Python para aprendizaje automático ya que contiene gran variedad de módulos y algoritmos y herramientas de procesamiento y análisis de datos. Además es compatible con otras liberías como Numpy, SciPy, Matplotlib o Pandas. Su carácter *open source* y su facilidad de aprendizaje han llevado a convertirla en una herramienta de uso extendido.

Aunque hay disponibles algunos módulos desarrollados para aprendizaje ML [11][12], hasta la fecha no incluye ningún módulo que realice aprendizaje MIML. Por este motivo, en este TFG pretendemos crear un paquete basado en `scikit-learn` con funcionalidad básica para aprendizaje MIML.

3. OBJETIVOS

El objetivo principal de este Trabajo Fin de Grado es desarrollar una librería en Python que pueda ser usada como punto de partida para el desarrollo y comparación de nuevos algoritmos MIML. Este objetivo general puede desglosarse en los siguientes objetivos específicos:

- Realizar un manejo básico de datasets MIML que permita realizar tareas entre las que se incluyen:
  - Diseñar un formato apropiado para el almacenamiento en disco de datos MIML.
  - Diseñar un formato apropiado para el almacenamiento en memoria de datos MIML.
  - Cargar los datasets en memoria para su posterior procesamiento.
  - Realizar particionado para validación cruzada de k iteraciones sobre los datos.
  - Obtener métricas sobre las características más relevantes de los datasets (número de patrones, número medio de instancias por patrón, número medio de etiquetas por patrón, etc.).
- Implementar las transformacines aritmética, geométrica y minimax de datos MIML para poder resolver el problema mediante algoritmos ML.
- Implementar las transformacines LP y BR de datasetS MIML para poder resolver el problema mediante algoritmos MI.
- Diseño y desarrollo de las clases necesarias para poder desarrollar clasificadores basados en aprendizaje MIML.
- Implementación de un clasificador MIML sencillo.
- Prueba del módulo mediante el desarrollo de un paquete tutorial que ilustrará las principales funcionalidades de la librería.

4. FASES DE DESARROLLO DEL PROYECTO

El Trabajo Fin de Grado se desarrollará en las siguientes etapas:

- Estudio e investigación: Se estudiarán las distintas tecnologías necesarias para el desarrollo del Trabajo Fin de Grado y se investigará en profundidad acerca del tema de aprendizaje MIML.
- Análisis y definición de requisitos: Se analizará lo anteriormente investigado para obtener los requisitos del proyecto.
- Diseño: Se definirá la estructura para el software asegurandose de que cumplirá los requisitos previamente establecidos.
- Implementación: Se traducirá el diseño y la planificación previos en código ejecutable y funcional, en el lenguaje de programación Python.
- Pruebas: Comprobación de la calidad, fiabilidad y funcionamiento correcto de la librería.
- Documentación: Se documentará cada una de las etapas a lo largo de todo el desarrollo del trabajo.

El Trabajo Fin de Grado ha de tener una duración de 300 horas (correspondientes a 12 créditos ECTS correspondientes por completo al alumno). Para ello hemos realizado una distribución en 15 semanas, con 20 horas de trabajo planificadas para cada semana que se detalla en la Tabla 1.

Semana	Tarea
1-2	Estudio aprendizaje Multi-Etiqueta y aprendizaje Multi-Instancia
3-4	Estudio de Python y las diferentes librerías a utilizar
5-6	Análisis y definición de requisitos
7-9	Diseño y desarrollo de las clases de la librería que se va a implementar
10-11	Implementación de la librería
12-13	Pruebas del software de la librería a realizar
14-15	Reunificación de la documentación realizadas en cada una de las fases

Cuadro 1: Distribución temporal del proyecto

5. RECURSOS

5.1. Recursos Humanos

El Trabajo Fin de Grado será realizado por Damián Martínez Ávila con la supervisión de la profesora Eva Lucrecia Gibaja Galindo.

5.2. Recursos Software

Para llevar a cabo la implementación del módulo desarrollado se utilizará el lenguaje Python. De este modo se aprovecharán las funcionalidades de librerías como Scikit Learn, Numpy y Pandas proporcionan para el procesamiento de los datos.

El entorno de programación que se utilizará será Visual Studio Code.

La documentación se realizará con Latex.

5.3. Recursos Hardware

Como equipo hardware para el desarrollo se ha utilizado un ordenador portátil personal con las siguientes características:

- Procesador Ryzen 7 a 1.8 GHz
- Memoria RAM 16 GB
- Disco Duro SSD de 512 GB
- Sistema operativo: Windows 11 de 64 bits

BIBLIOGRAFÍA

[1] P. Cunningham, M. Cord, and S. J. Delany, *Supervised Learning*, pp. 21–22. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[2] A. R. Rout, "A deep learning model for image classification," *IRJET*, 2017.

[3] E. Gibaja and S. Ventura, "Multilabel learning: A review of the state of the art and ongoing research," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11 2014.

[4] E. Gibaja and S. Ventura, "A tutorial on multi-label learning," *ACM Computing Surveys*, vol. 47, 04 2015.

[5] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997.

[6] Weka 3: Machine Learning Software in Java, "www.cs.waikato.ac.nz/ml/weka," 5 de Octubre de 2023.

[7] Mulan: A Java Library for Multi-Label Learning, "https://mulan.sourceforge.net/," 5 de Octubre de 2023.

[8] scikit-learn: Machine Learning in Python, "https://scikit-learn.org/stable/," 5 de Octubre de 2023.

[9] Lamda: Learning and mining from data, "https://www.lamda.nju.edu.cn/data.ashx," 5 de Octubre de 2023.

[10] E. G. Álvaro Belmonte and A. Zafra, "Miml library, https://github.com/kdis-lab/MIML," 2023.

[11] scikit-multilearn: Python module for multi-label learning tasks, "https://github.com/scikit-multilearn/scikit-multilearn," 5 de Octubre de 2023.

[12] mil: multiple instance learning library for Python, "https://github.com/rosasalberto/mil/tree/master/mil," 5 de Octubre de 2023.