ESCUELA POLITÉCNICA
SUPERIOR DE CÓRDOBA
Universidad de Córdoba

ANTEPROYECTO

Grado en Ingeniería Informática

Desarrollo de los algoritmos de clasificación ordinal KDLOR, SVC1V1 y SVC1VA para ORCA-Python

Autor
Cristian Torres Pérez de Gracia

Directores
David Guijo Rubio
Pedro Antonio Gutiérrez Peña

Octubre, 2023

UNIVERSIDAD D CORDOBA

Índice

1. Datos del Proyecto

Título: Desarrollo de los algoritmos de clasificación ordinal KDLOR, SVC1V1 y SVC1VA para ORCA-Python.

Autor:

- **Nombre:** Cristian Torres Pérez de Gracia
- **Email:** i52topec@uco.es
- **Titulación:** Grado en Ingeniería Informática
- **Mención:** Computación

Directores del Proyecto:

- David Guijo Rubio
- Pedro Antonio Gutiérrez Peña

2. Introducción

La convergencia entre la informática y la comunicación ha creado una sociedad que se nutre de la información. A la información que se encuentra sin tratar se la conoce como datos. Si los datos se caracterizan como hechos registrados, entonces la información es el conjunto de patrones que subyacen de los datos. En la actualidad, hay una gran cantidad de información almacenada en las bases de datos, siendo esta de suma importancia por lo que cada vez se usan más procedimientos de análisis de datos. La minería de datos se encarga de extraer esa información que resulta de utilidad a partir del conjunto de datos generado por procedimientos estándares. En este punto es importante introducir el concepto de aprendizaje automático, que es el área que usa estos datos extraídos de numerosos procedimientos previos para obtener conclusiones más o menos generales a partir de casos reales que han ocurrido previamente.

El aprendizaje automático, o *machine learning*, proporciona mecanismos que permiten al ordenador aprender por sí mismo a resolver un determinado problema. Según el tipo o forma de adquisición de conocimiento, el aprendizaje automático puede dividirse, principalmente, en: supervisado y no supervisado. En el caso del aprendizaje supervisado, los ejemplos de entrenamiento van acompañados de la salida que el sistema debería ser capaz de reproducir. Mientras que en el aprendizaje no supervisado, se construyen descripciones, hipótesis o teorías a partir de un conjunto de hechos u observaciones, sin que exista información adicional acerca de cómo deberían agruparse los ejemplos del conjunto de entrenamiento. Dentro del aprendizaje automático supervisado hay dos tipos de problemas a los que nos podemos enfrentar principalmente: clasificación y regresión. Los algoritmos de clasificación predicen etiquetas o clases que se conocen a priori. El resultado deseado es un valor discreto. Por otro lado, los algoritmos de regresión son aquellos que se utilizan en problemas donde la salida es un valor numérico. Una vez definidos los dos tipos de tareas principales de aprendizaje automático, introducimos el concepto de regresión ordinal, que es un área que se encuentra a medio camino entre la clasificación y la regresión.

La regresión ordinal o clasificación ordinal [1] es el área del aprendizaje automático, en el que las etiquetas muestran un orden natural entre sí. Un claro ejemplo de estos mismos es cuando se evalúa la satisfacción de un servicio entre 1 y 5 en escala de Likert, siendo 1 muy deficiente y 5 excelente. Cuando se trabaja con este tipo de problemas hay que tener en cuenta que los errores de clasificación no pueden tener el mismo peso, es decir, no se

debe penalizar de la misma forma un servicio clasificado como muy deficiente siendo este excelente, a si cometiéramos el error de clasificarlo como deficiente o decente.

En este Trabajo de Fin de Grado vamos a centrarnos en la clasificación ordinal, más concretamente, en la implementación de 3 algoritmos (`KDLOR`, `SVC1V1` y `SVC1VA`) que estaban previamente desarrollados en `ORCA` [2] y para los cuales debemos obtener resultados similares para unos problemas utilizando el *framework* `ORCA-Python` [3, 4]. `ORCA` es un *framework* de `MATLAB` [5] que implementa e integra una amplía gama de métodos de regresión ordinal y de métricas de rendimiento. La implementación se va a realizar en `ORCA-Python`, que es un *framework* escrito en `Python` integrado con los módulos de `Scikit-Learn` [6] y `sacred` [7].

3. Objetivos

Se desarrollará una biblioteca escrita en `Python` que implemente los algoritmos de clasificación ordinal propuestos. Para ello, se tomará como base el *framework* de `ORCA-Python` [3, 4], el cual consiste en una adaptación y ampliación del *framework* `ORCA` [2] que esta escrito en `MATLAB/Octave`. `ORCA-Python` es un *framework* que incluye implementaciones previas realizadas por otros compañeros a partir de Trabajos Fin de Grado.

Estos algoritmos ya se encuentran implementados en el *framework* `ORCA` [2], desarrollado por el grupo AYRNA [8], por lo que uno de los objetivos será profundizar en estos algoritmos de clasificación ordinal para su posterior adaptación e implementación en `ORCA-Python` [3, 4].

A continuación, se citan los puntos que se van a abordar en este trabajo:

- Implementación del algoritmo de `KDLOR` [9] (*Kernel Discriminant Learning for Ordinal Regression*): reformulación del aprendizaje discriminante que permite el cálculo del óptimo mapeo unidimensional de los datos, mediante el análisis de dos objetivos: la maximización de la distancia entre clases, y la disminución de la distancia intraclase. Para reformular el algoritmo de regresión ordinal, una restricción de orden sobre las clases contiguas es impuesta sobre el promedio de los patrones proyectados de cada clase.
- Implementación del algoritmo de `SVC1V1` [10] (*Support Vector Classifier using one-vs-one*): máquina de vectores soporte nominal que realiza la formulación *OneVsOne*. Es considerado como un enfoque ingenuo para los problemas de regresión ordinal, ya que ignora el orden de la información. Este método construye $K(K-1)/2$ clasificadores donde cada uno es entrenado con datos de dos clases.
- Implementación del algoritmo de `SVC1VA` [10] (*Support Vector Classifier using one-vs-all*): máquina de vectores soporte nominal que realiza la formulación *OneVsAll*. Es considerado como un enfoque ingenuo para los problemas de regresión ordinal, ya que ignora el orden de la información. Este método construye K clasificadores, uno por cada clase. Ese clasificador se entrena tomando como etiquetas positivas las muestras pertenecientes a esa clase y como negativas las del resto de clases.
- Comprobación del correcto funcionamiento de cada algoritmo a través de diversas pruebas.

4. Antecedentes

4.1. Python

`Python` [11] es un lenguaje de programación de propósito general muy poderoso y flexible, a la vez que sencillo y fácil de aprender, que fue creado a principios de los noventa por Guido Van Rossum. Es un lenguaje de alto nivel, que permite procesar fácilmente todo tipo de estructuras de datos, tanto numéricas como de texto.

4.2. ORCA

`ORCA` [2] (*Ordinal Regression and Classification Algorithms*) es un *framework* de `MATLAB` [5] que implementa e integra una amplía gama de métodos de regresión ordinal y de métricas de rendimiento. Fue desarrollado por el grupo de investigación AYRNA [8], que pertenece al Departamento de Informática y Análisis Numérico de la Universidad de Cordoba (UCO). A continuación se explicaran con más detalles los algoritmos de regresión ordinal que se van a desarrollar en `ORCA-Python` [3, 4].

El aprendizaje discriminante se ha reformulado para abordar regresión ordinal. Esta metodología es conocida como `KDLOR` [9] (*Kernel Discriminant Learning for Ordinal Regression*). El análisis discriminante no suele ser considerado como una técnica de clasificación por si misma, sino más bien como una reducción de la dimensionalidad supervisada. En general, permite el calculo del óptimo mapeo unidimensional de los datos, mediante el análisis de dos objetivos: la maximización de la distancia entre clases, y la disminución de la distancia intraclase, usando para ello las matrices de varianza-covarianza y el coeficiente de Rayleigh. Para tener en cuenta los problemas de regresión ordinal, una restricción de orden sobre las clases contiguas es impuesta sobre el promedio de los patrones proyectados de cada clase, lo que lleva al algoritmo a ordenar los patrones proyectados según sus etiquetas. Esto preservará la información ordinal y evitará errores graves de clasificación.

Dentro de los enfoques ingenuos tenemos dos formulaciones *OneVsOne* y *OneVsAll* (`SVC1V1` y `SVC1VA`) [10] que son las principales formulaciones en SVM cuando se trabaja con problemas multiclase. Aunque estos métodos consideran la descomposición binaria, han sido incluidos dentro de los grupos nominales de clasificación debido a que no se toma en cuenta el orden.

El método *OneVsOne* construye $K(K-1)/2$ clasificadores donde cada uno es entrenado con datos de dos clases. Se pueden considerar diferentes métodos para la fase de generalización una vez que se han construido los $K(K-1)/2$ clasificadores. Un ejemplo de estos métodos sería el siguiente: si $\text{sin}((w^{ij})^T\phi(x) + b^{ij})$ dice que x pertenece a la clase i entonces se aumenta en un voto la clase i, en caso contrario, se aumenta en un voto la clase j. X pertenecerá a la clase con más votos. El enfoque de votación descrito es llamado como la estrategia de *Max Win*.

El método *OneVsAll* construye K clasificadores, uno por cada clase. Ese clasificador se entrena tomando como etiquetas positivas las muestras pertenecientes a esa clase y como negativas las del resto de clases. Después de resolver el problema, tendremos k funciones de decisión y x pertenecerá a la función de decisión con el mayor valor:

$$clase\ de\ x \equiv \underset{x=1,\ldots,k}{\text{argmax}}((w^i)^T\phi(x) + b^i)$$

4.3. ORCA-Python

`ORCA-Python` [3, 4] es un *framework* escrito en `Python` integrado con los módulos de `Scikit-Learn` [6] y `sacred` [7], que busca automatizar la ejecución de los experimentos de *machine learning* mediante ficheros de configuración fáciles de entender. Fue desarrollado como Trabajo Fin de Grado del alumno Iván Bonaque Muñoz y posteriormente ampliado por el alumno Ángel Heredia Pérez y se trata de una ampliación y adaptación del *framework* ORCA [2] al lenguaje de progamación `Python`.

Este *framework* es compatible con cualquier algoritmo que se encuentre implementado en `Scikit-Learn` o bien creado por el propio usuario siempre que siga las reglas de compatibilidad de esta biblioteca. Si bien, como ya se ha mencionado, este está escrito en `Python`, mientras que algunos algoritmos ya implementados en `ORCA-Python`, como es el caso de `REDSVM` o `SVOREX`, están implementados en un lenguaje de programación diferente, `C++` y `C`, respectivamente.

La correcta ejecución del *framework* requiere de la instalación de las siguientes dependencias de `Python`:

- `NumPy`
- `Pandas`

- `Scikit-Learn`
- `SciPy`
- `Sacred`

Por último, se muestra una tabla con las diferentes aportaciones de los alumnos al proyecto:

Alumno	TFG
Iván Bonaque Muñoz	Diseño e implementación del framework ORCA-Python
Ángel Heredia Pérez	Implementación de REDVSM y SVOREX
Manuel Jesús Cabrera Delgado	Implementación de CSSVC, wrapper de regresión
Adrián López Ortíz	Implementación de NMPOM y NNOP
Javier Lozano Rojas	Implementación de SVMOP Y POM
Cristian Torres Pérez de Gracia	Implementación de SVC1V1, SVC1VA y KDLOR

Tabla 4.1: Tabla comparativa

4.3.1. NumPy

`Numpy` [12] es una biblioteca para `Python` que facilita el trabajo con *arrays*. Esta biblioteca permite declarar *arrays* con distintas dimensiones capaces de albergar gran cantidad de datos del mismo tipo y relacionados entre sí. Además, provee numerosos métodos para manipular los *arrays*; y para acceder a la información y procesarla de forma muy eficiente.

4.3.2. Pandas

`Pandas` [13] es una biblioteca para `Python`, que proporciona una serie de estructuras de datos y operaciones para manipulación y el análisis de los mismos. `Pandas` permite importar varios formatos de archivos (`csv, excel, sql, json, parquet, etc`).

4.3.3. Scikit-Learn

`Scikit-Learn` [6] es una biblioteca de software libre para `Python`. Cuenta con varios algoritmos como máquina de vectores soporte, *random forest* y *K*-

neighbours. También admite bibliotecas numéricas y científicas como `NumPy` y `SciPy`.

4.3.4. SciPy

`Scipy` [14] es un software de código abierto que nació a partir de la colección original de Travis Oliphant y que consistía en módulos de extensión para `Python`. La biblioteca de `SciPy` depende de `NumPy`, proporcionando una manipulación de vectores N-dimensionales de una forma cómoda y rápida. Además contiene diversos módulos para optimización, álgebra lineal, integración y otras tareas relacionadas con la ciencia e ingeniería.

4.3.5. Sacred

`Sacred` [7] es una herramienta que permite configurar, organizar, registrar y reproducir experimentos computacionales. Está diseñado para introducir la mínima sobrecarga, y, al mismo tiempo, fomentar la modularidad y facilitar la configuración de de los experimentos.

La capacidad para hacer experimentos configurables es el corazón de `Sacred` y permite, entre otros:

- Realizar un seguimiento de todos los parámetros de tu experimento.
- Guardar la versión del código que se ha lanzado, para evitar problemas de saber que código has utilizado para un determinado experimento.
- Ejecutar tus experimentos con diferentes configuraciones.
- Guardar configuración para ejecuciones individuales en ficheros o en una base de datos.
- Reproducir tus resultados.

5. Fases de desarrollo

5.1. Estudio y análisis del problema

Se realizará un análisis del problema para alcanzar una cierta comprensión sobre el mismo. Para ello se deberá profundizar en los algoritmos de clasificación ordinal ya implementados en el *framework* `ORCA` así como acostumbrarse al *framework* `ORCA-Python`. Además, se necesita aprender a utilizar todas las herramientas y librerías mencionadas en la sección 4.3.

5.2. Diseño

Una vez obtenidos los requisitos, resultados del análisis, se empezará la etapa de diseño donde se han de estudiar todas las posibles alternativas de implementación para nuestro problema. El diseño deberá cumplir los objetivos y requisitos extraídos durante la fase de análisis.

5.3. Codificación

Se procederá a su codificación utilizando `Python` y las librerías mencionadas en la sección 4.3.

5.4. Pruebas

La etapa de pruebas tiene como objetivo detectar los errores que se hayan podido cometer en las etapas anteriores del proyecto, para posteriormente corregirlas. Por lo tanto, se realizaran una serie de pruebas para verificar el correcto funcionamiento de los algoritmos implementados.

5.5. Documentación

Durante el desarrollo del proyecto se redactará el manual de usuario y el manual técnico con el fin de que cualquier persona ajena al proyecto se capaz de entenderlo y sea capaz de utilizarlo con un mínimo esfuerzo. Además, el Trabajo de Fin de Grado se irá subiendo a Github.

6. Recursos

6.1. Recursos humanos

Autor: Cristian Torres Pérez de Gracia
Alumno de 4º curso de Ingeniería informática.

Director: David Guijo Rubio
Investigador postdoctoral en la Universidad East Anglia, Norwich, Reino Unido y miembro investigador del grupo AYRNA.

Director: Pedro Antonio Gutiérrez Peña
Profesor Titular de la Universidad de Córdoba del Dpto. Informática y Análisis Numérico y miembro investigador del grupo AYRNA.

6.2. Recursos materiales

A continuación se citarán los distintos recursos software y hardware que se utilizarán en dicho Trabajo de Fin de Grado.

6.2.1. Recursos software

En el desarrollo de dicho proyecto se utilizarán los siguientes recursos software:

- Se utilizará el lenguaje de programación `Python`.
- `Ubuntu`.
- Para la documentación se utilizará Overleaf (editor online de LaTeX).

6.2.2. Recursos hardware

Como recursos hardware se utilizará el portátil del proyectista cuyas características se especifican a continuación:

- Procesador Intel (R) Core (TM) i7-8750H CPU @ 2.20GHz
- Memoria RAM 16GB DDR4
- Controlador gráfico GeForce® GTX 1060 6GB GDDR5
- Disco duro HGST Travelstar 7K1000 1TB + SSD de 256GB

7. Planificación temporal

La planificación a seguir de las fases detalladas en la sección 5 se especifican en la Tabla 7.2:

Fase de desarrollo	**Año 2021 - 2022**				**Horas estimadas**
	Mes 1	**Mes 2**	**Mes 3**	**Mes 4**	
Estudio y análisis del problema	20	10	-	-	30
Diseño	30	20	10	-	60
Implementación	-	30	35	35	100
Pruebas	-	15	15	20	50
Documentación	5	10	20	25	60
Total	55	85	80	80	300

Tabla 7.2: Planificación horaria

Referencias

[1] P.A. Gutiérrez, M. Pérez-Ortiz, J. Sánchez-Monedero, F. Fernandez-Navarro y C. Hervás-Martínez. «Ordinal regression methods: survey and experimental study». En: *IEEE Transactions on Knowledge and Data Engineering* 28.1 (2016), págs. 127-146. URL: http://dx.doi.org/10.1109/TKDE.2015.2457911.

[2] Javier Sánchez-Monedero, Pedro A. Gutiérrez y María Pérez-Ortiz. «ORCA: A Matlab/Octave Toolbox for Ordinal Regression». En: *Journal of Machine Learning Research* 20.125 (2019), págs. 1-5. URL: http://jmlr.org/papers/v20/18-349.html.

[3] Iván Bonaque Muñoz, Pedro Antonio Gutiérrez Peña y Javier Sánchez Monedero. *Trabajo de Fin de Grado. Framework en Python para problemas de clasificación ordinal.* 2019.

[4] Ángel Heredia Pérez, Pedro Antonio Gutiérrez Peña y Juan Carlos Fernández Caballero. *Trabajo de Fin de Grado. Desarrollo de una biblioteca para algoritmos de clasificación ordinal en Python.* 2020.

[5] MATLAB. *version 9.0 (R2016a).* The MathWorks Inc., Natick, Massachusetts, 2016.

[6] *Sitio oficial de Scikit-learn.* URL: https://scikit-learn.org/stable/. [Online. Última consulta: 23-09-2023].

[7] *Sitio oficial de Sacred.* URL: https://sacred.readthedocs.io/en/stable/. [Online. Última consulta: 23-09-2023].

[8] *Departamento de Informática y Análisis Numérico de la Universidad de Córdoba. Aprendizaje y redes neuronales artificiales.* URL: https://www.uco.es/grupos/ayrna/index.php/es. [Online. Última consulta: 23-09-2023].

[9] B.-Y. Sun, D. D. Wu J. Li, X.-M. Zhang y W.-B. Li. «Kernel discriminant learning for ordinal regression». En: *IEEE Trans. Knowl. Data Eng* 22.6 (2016), págs. 906-910. URL: https://www.researchgate.net/publication/224569314_Kernel_Discriminant_Learning_for_Ordinal_Regression.

[10] C.-W. Hsu y C.-J. Lin. «A comparison of methods for multiclass support vector machines». En: *IEEE Trans. Neural Netw.* 13.2 (2002), págs. 415-425.

[11] *Sitio oficial de Python.* URL: https://www.python.org/. [Online. Última consulta: 23-09-2023].

[12] *Sitio oficial de NumPy*. URL: `https://numpy.org/`. [Online. Última consulta: 23-09-2023].

[13] *Sitio oficial de Pandas*. URL: `https://pandas.pydata.org/`. [Online. Última consulta: 23-09-2023].

[14] *Sitio oficial de Scipy*. URL: `https://www.scipy.org/`. [Online. Última consulta: 23-09-2023].