INVESTIGACIÓN SOBRE EL ALGORITMO CLASSIFIER CHAINS

Diagrama, Tabla

Descripción generada automáticamente

Jan. 14, 2025

Luis Mellado Díaz

melladodiazluis@gmail.com

Universidad de Sevilla – Grado en Ingeniería del Software

# Contenido

[Investigacion 1](#_Toc187775302)

[Transformación de Problemas 1](#_Toc187775304)

[Binary Relevance (BR) 1](#_Toc187775305)

[Label Powerset (LP) 2](#_Toc187775306)

[Classifier Chain 2](#_Toc187775307)

[Conjunto de Clasificadores - Ensembles 2](#_Toc187775308)

[Dudas 3](#_Toc187775309)

[Bibliografía 3](#_Toc187775310)

# Tabla de Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Autor | Fecha | Cambios |
| V1.0 | Luis Mellado Díaz | 14/01/2024 | * Creación del documento. * Investigación Algoritmo Classifier Chain |
| V1.1 | Luis Mellado Díaz | 15/01/2024 | * Índice * Investigación Algoritmos Clustering Difuso |

# Investigación Algoritmo Classifier Chain

# Classifier chains es un método de aprendizaje automático para la transformación de problemas en la clasificación multietiqueta. Combina la eficiencia computacional del método de relevancia binaria, al mismo tiempo que es capaz de tener en cuenta las dependencias entre etiquetas para la clasificación. (Read, Pfahringer, Holmes, & Frank, 2009)

## Transformación de Problemas

### Binary Relevance (BR)

Dado un conjunto de etiquetas **L** y un conjunto de datos con instancias de la forma **(x,Y)**, donde **x** es un vector de características e **Y⊆L** es un conjunto de etiquetas asignadas a la instancia. El método BR (Relevancia Binaria) transforma el conjunto de datos en **∣L|** conjuntos de datos y entrena **∣L|** clasificadores binarios **H:X→{l,¬l}** para cada **l∈L**.

Durante este proceso, la información sobre las dependencias entre etiquetas no se conserva. Esto puede conducir a situaciones donde se asigna un conjunto de etiquetas a una instancia, aunque esas etiquetas nunca hayan ocurrido juntas en el conjunto de datos. Por lo tanto, la información sobre la co-ocurrencia de etiquetas puede ayudar a asignar combinaciones de etiquetas correctas. La pérdida de esta información puede, en algunos casos, provocar una disminución en el rendimiento de la clasificación. (Dembczynski, Waegeman, Cheng, & Hüllermeier, 2010)

### Label Powerset (LP)

Cada combinación de etiquetas en un conjunto de datos se considera como una única etiqueta. Después de la transformación, se entrena un clasificador de etiqueta única **H:X→P(L)**, Donde **P(L)** es el conjunto potencia de todas las etiquetas en **L**.

El principal inconveniente de este enfoque es que el número de combinaciones de etiquetas crece exponencialmente con el número de etiquetas (10 etiquetas caben a 2^10 combinaciones). Esto aumenta el tiempo de ejecución del proceso de clasificación.

## Classifier Chain

Para un conjunto dado de etiquetas **L**, el modelo **Classifier Chain (CC)** aprende **∣L∣** clasificadores, al igual que en el método de Relevancia Binaria. Todos los clasificadores están enlazados en una cadena a través del espacio de características.

Dado un conjunto de datos donde la **i-ésima** instancia tiene la forma **(xi​,Yi​)**, donde **Yi**​ es un subconjunto de etiquetas y **xi**​ es un conjunto de características. El conjunto de datos se transforma en **∣L∣** conjuntos de datos, donde las instancias del **j-ésimo** conjunto de datos tienen la forma **((xi​,l1​,...,lj−1​),lj​),lj​∈{0,1}**. Si la **j-ésima** etiqueta fue asignada a la instancia, entonces **lj​=1**, de lo contrario, **lj​=0**. Así, los clasificadores forman una cadena donde cada uno aprende la clasificación binaria de una sola etiqueta. Las características dadas a cada clasificador se amplían con valores binarios que indican cuáles de las etiquetas anteriores fueron asignadas a la instancia.

Al clasificar nuevas instancias, las etiquetas se predicen nuevamente construyendo una cadena de clasificadores. La clasificación comienza con el primer clasificador **C1**​ y avanza hasta el último **C∣L∣**​, pasando la información sobre las etiquetas entre los clasificadores a través del espacio de características. De este modo, se conserva la dependencia entre etiquetas. Sin embargo, el resultado puede variar dependiendo del orden de la cadena. Por ejemplo, si una etiqueta suele co-ocurrir con otra etiqueta, entonces solo las instancias de la etiqueta que aparece más tarde en la cadena tendrán información sobre la otra en su vector de características. Para solucionar este problema y aumentar la precisión, es posible utilizar un **conjunto de clasificadores**. (Rokach, 2010)

### Conjunto de Clasificadores - Ensembles

En el **Ensemble of Classifier Chains (ECC)**, varios clasificadores CC pueden ser entrenados con un orden aleatorio de cadenas (es decir, un orden aleatorio de etiquetas) sobre un subconjunto aleatorio del conjunto de datos. Las etiquetas de una nueva instancia son predichas por cada clasificador por separado. Después, se cuenta el número total de predicciones o "votos" para cada etiqueta. La etiqueta se acepta si fue predicha por un porcentaje de clasificadores que supera un valor umbral determinado.

### Ejemplo

Tienes un conjunto de datos con varias instancias. Cada instancia tiene:

* **Características (x)**: Son las variables que describen la instancia (por ejemplo, color, tamaño, etc.).
* **Etiquetas (Y)**: Son las etiquetas que están asociadas con la instancia, en un formato multietiqueta (es decir, una instancia puede tener más de una etiqueta).

Por ejemplo, una instancia podría ser:

* **Características (x)**: [color: rojo, tamaño: grande]
* **Etiquetas (Y)**: [etiqueta A, etiqueta B, etiqueta C]

En lugar de entrenar un clasificador que prediga todas las etiquetas al mismo tiempo, el modelo **Classifier Chains (CC)** divide el problema en varios clasificadores, uno por cada etiqueta.

En nuestro ejemplo, **Classifier Chains** entrenará tres clasificadores:

* Clasificador 1 predice si **A** está presente o no.
* Clasificador 2 predice si **B** está presente o no, pero también usa la predicción de **A** como una característica adicional.
* Clasificador 3 predice si **C** está presente o no, usando las predicciones de **A** y **B** como características adicionales.

# Investigación Algoritmos Clustering Difuso

El agrupamiento difuso es una clase de algoritmos de agrupamiento donde cada elemento tiene un grado de pertenencia difuso a los grupos.

Este tipo de algoritmos surge de la necesidad de resolver una deficiencia del agrupamiento exclusivo, que considera que cada elemento se puede agrupar inequívocamente con los elementos de su cluster y que, por lo tanto, no se asemeja al resto de los elementos.

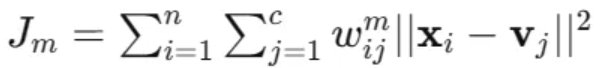
## Conjunto Difuso

Un conjunto difuso es una clase de objetos con un continuo de grados de pertenencia. Dicho conjunto se caracteriza por una función de pertenencia (característica) que asigna a cada objeto un grado de pertenencia que varía entre cero y uno. (Zadeh, 1965)

Sea **X** un espacio de puntos (objetos), con un elemento genérico de **X** denotado por x. Así, **X={x}**

Un conjunto difuso (clase) **A** en **X** se caracteriza por una función de pertenencia (característica) **fA(x)** que asocia a cada punto **x** en **X** un número real en el intervalo **[0,1]**, donde el valor de **fA(x)** en **x** representa el "grado de pertenencia" de **x** en **A**. Así, cuanto más cercano sea el valor de **fA(x)** a la unidad, mayor será el grado de pertenencia de **x** en **A**.

## Algoritmo Fuzzi c-Means

El **objetivo** es obtener la partición difusa óptima para **X**. El método más popular hasta el momento está asociado con la función de error mínimo cuadrático: (Bezdek, Ehrlich, & Full, 1984)

Donde:

*n = numero de puntos*

*c = numero de clusters*

*x = ‘i’ data point*

*v = centroide del cluster ‘j’*

*w = membresía del punto ‘i’ al cluster ‘j’*

*m = fuzziness (m>1)*

El funcionamiento del algoritmo Fuzzi c-Means es el siguiente:

* 1. **Inicialización:** Elegir e inicializar aleatoriamente los centroides de los clusters a partir del conjunto de datos y especificar un parámetro de difusividad (m) para controlar el grado de difusividad en la agrupación.
  2. **Actualización de membresía:** Calcular el grado de pertenencia de cada punto de datos a cada cluster basado en su distancia a los centroides de los clusters utilizando una métrica de distancia (por ejemplo: distancia euclidiana).

Diagrama

Descripción generada automáticamente

* 1. **Actualización de centroides:** Actualizar el valor del centroide y recalcular los centroides de los clusters en función de los valores de pertenencia actualizados.

Imagen que contiene Texto

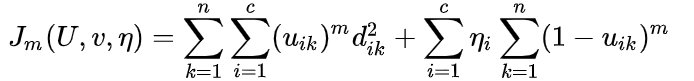
Descripción generada automáticamente

* 1. **Verificación de convergencia:** Repetir los pasos 2 y 3 hasta que se alcance un número especificado de iteraciones o los valores de pertenencia y centroides converjan a valores estables.

## Algoritmo Possibilistic c-Means

Los algoritmos Possibilistic c-Means pretenden resolver el mal comportamiento de los algoritmos Fuzzy c-Means cuando se aplican sobre datasets con mucho ruido. (Krishnapuram & Keller, 1993)

Se caracteriza por interpretar **w** como grado de compatibilidad con el grupo, en vez de probabilidad de pertenencia. Una de las funciones objetivo más empleadas es: (Krishnapuram & Keller, 1993)



El vector  **ηiη** es un vector de valores positivos, donde **ηi** denota la distancia desde el centro del grupo i a la que el grado de pertenencia de un elemento es 0.5.

Texto

Descripción generada automáticamente

Donde en la mayoría de los casos, K = 1.

El procedimiento general para estos algoritmos suele ser:

1. Fijar ***c***, ***m***, ***A*** y **||*k*||A**. Elegir una matriz inicial U(0)∈Mfcy estimar los valores de η**η**.
2. Calcular los centros de los grupos.

Diagrama, Esquemático

Descripción generada automáticamente

1. Actualizar la matriz de partición difusa U = [uik]

Gráfico

Descripción generada automáticamente con confianza baja

1. Si se alcanzó el criterio de parada, terminar. En caso contrario, regresar al paso 2.

# Implementación

[6.3. Preprocessing data — scikit-learn 1.6.1 documentation](https://scikit-learn.org/stable/modules/preprocessing.html)

<https://scikit-learn.org/1.6/auto_examples/multioutput/plot_classifier_chain_yeast.html>

<https://www.youtube.com/watch?v=WOUMZatSiLM>

<https://github.com/IbraDje/PFCM>

# Dudas

* Escribo en Ingles o español
* Porque necesito escalar los datos, que significa escalar el dataset
* OneVSRest classifier

# Bibliografía

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The Fuzzy c-Means Clustering Algorithm. *Computers & Geosciences 10 (2-3)*, 191-203.

Dembczynski, K., Waegeman, W., Cheng, W., & Hüllermeier, E. (2010). On label dependence in multi-label classification. *Workshop Proceedings of Learning from Multi-Label Data*, 5-12.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2009). Classifier Chains for Multi-label Classification. En W. G.-T. Buntine, *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2009* (pág. Volumen 5782). Springer, Berlin, Heidelberg.

Rokach, L. (2010). Ensemble-based classifiers. *Artif. Intell. Rev. 33*, 1-2.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 338-353.