UNIVERSIDAD DE GRANADA

Algoritmos de discretización

Autores
Juan Manuel Castillo Nievas
Francisco José González García
Víctor Torres de la Torre

Máster Profesional en Ingeniería Informática

—

Granada, 6 de enero de 2021

Índice

1. IEM (Information Entropy Maximization)

Este algoritmo es un **algoritmo supervisado**. Para poder entender este algoritmo es necesario conocer dos conceptos importantes:

- **Entropía:** la entropía hace referencia al desorden o a la incertidumbre de un conjunto de datos. Cuanto más grande es la entropía, más incretidumbre hay en los datos. Una entropía igual a 0 significa que los datos son 100 % predecibles y que no tienen incertidumbre. La entropía se mide siguiendo la siguiente ecuación:

$$H(x) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

  donde $p(x_i)$ se refiere a la probabilidad.

- **Ganancia de información:** mide la cantidad de información que nos da un atributo con respecto a la clase. El algoritmo debe maximizar la ganancia de información. Lo que se pretende con este algoritmo es que los conjuntos no estén "mezclados", es decir, que su entropía sea la más baja posible y de esta forma maximizar la ganancia de información.

1.1. Entendiendo el concepto de entropía

Para entender bien el concepto de entropía se propone un ejemplo bastante sencillo [1]. Imaginemos un conjunto de datos muy sencillo en el se describe si una persona tiene un salario anual de más de 50.000€ (ver Tabla 1).

Se calcula la entropía de los dos intervalos (salario menor o mayor a 50.000€):

$H(salario) = -(\frac{4}{10}\log_2(\frac{4}{10}) + \frac{6}{10}\log_2(\frac{6}{10})) = 0{,}529 + 0{,}442 = 0{,}971$

La entropía es muy cercana a 1, con lo cual indica que los datos están muy mezclados o desordenados.

	Salario <= 50.000€	Salario >50.000€
Edad <25	4	6

Tabla 1: Ejemplo 1 del salario

Imaginemos el mismo ejemplo pero con otros datos. En este ejemplo (ver Tabla 2) tenemos que 9 personas ganan menos de 50.000€ y solamente una persona gana más de 50.000€. Se calcula la entropia:

$$H(salario) = -(\frac{9}{10}\log_2(\frac{9}{10}) + \frac{1}{10}\log_2(\frac{1}{10})) = 0{,}137 + 0{,}332 = 0{,}469$$

Se obtiene una entropía de 0.469 que es mucho menor que la entropía que se obtuvo en el anterior ejemplo. Esto quiere decir que estos datos nos proporcionan más información acerca de lo que queremos saber, que es el salario.

	Salario <= 50.000€	Salario >50.000€
Edad <25	9	1

Tabla 2: Ejemplo 2 del salario

Otro ejemplo sencillo sería el siguiente: imaginemos que tenemos una bolsa con 3 bolas negras y 3 bolas rojas y debemos coger una al azar. La entropía es 1 puesto que la incertidumbre es máxima. En el supuesto de que tuviéramos una bolsa con 6 bolas negras, la entropía sería 0, es decir, no hay incertidumbre posible puesto que sabemos que la bola va a ser negra.

1.2. Algoritmo IEM

El objetivo de este algoritmo es maximizar la ganancia de información asociada a una división de un intervalo. Para la discretización se utiliza lo que se conoce como **quanta matrix** (matriz de frecuencias), en la que se representan los intervalos en los que se divide la variable en las columnas y las diferentes clases en las filas. La intersección contiene el número de instancias de esa clase que están en ese intervalo (ver Figura 1). La fórmula de la ganancia de información se puede ver en la Figura 2.

La matriz de frecuencias viene dada por:

- **Clase C:** con S posibles valores o etiquetas
- **Atributo continuo** A_j: con rango de valores en $[a, b]$
- **Partición D_j on N_j intervalos**: $[d_0, d_1], (d_2, d_3], ..., (d_{n-1}, d_n]$
- Q_j: conjunto de frecuencias de aparición en cada etiqueta de la clase en un intervalo $[d_{r-1}, d_r]$
- M: total de instancias evaluadas
- M_{+r}: muestras en el intervalo $[d_{r-1}, d_r]$

- M_{i+}: muestras de la clase C_i

Class	Interval					Class Total
	$[d_0, d_1]$	…	$(d_{r-1}, d_r]$	…	$(d_{n-1}, d_n]$	
C_1	q_{11}	…	q_{1r}	…	q_{1n}	M_{1+}
⋮	⋮	…	⋮	…	⋮	⋮
C_i	q_{i1}	…	q_{ir}	…	q_{in}	M_{i+}
⋮	⋮	…	⋮	…	⋮	⋮
C_S	q_{S1}	…	q_{Sr}	…	q_{Sn}	M_{S+}
Interval Total	M_{+1}	…	M_{+r}	…	M_{+n}	M

Figura 1: **Quanta matrix**

$$INFO(C, D|F) = \sum_{i=1}^{S} \sum_{r=1}^{n} p_{ir} \log_2 \frac{p_{+r}}{p_{ir}} \qquad \begin{cases} p_{+r} = p(D_r|F) = \dfrac{M_{+r}}{M} \\ p_{ir} = p(C_i, D_r|F) = \dfrac{q_{ir}}{M} \end{cases}$$

Figura 2: Fórmula de la ganancia de información

Este algoritmo busca el **menor número de intervalos** en los que se puede dividir el atributo y maximiza el número de ejemplos de la misma clase en el mismo intervalo. El algoritmo se puede describir de la siguiente forma:

1. Ordenar los valores del atributo
2. Inicializar intervalo al intervalo total, $D \leftarrow [d_0, d_n]$
3. Inicializar la entropía global a 0, $GlobalIEM = 0$
4. Mientras no se cumpla el criterio de parada:
   a) Calcular la ganancia de información para la división de intervalos generada
   b) $LocalIEM =$ ganancia de información calculada
   c) Si $LocalIEM > GlobalIEM$
      1) $D \leftarrow D'$
      2) $GlobalIEM = LocalIEM$

El criterio de parada debe ser seleccionado antes de inicializar el algoritmo, pero hay 2 criterios de parada que se suelen usar frecuentemente:

- **Número de intervalos:** el algoritmo termina cuando el atributo se ha dividido en n intervalos.
- **Ganancia de información mínima:** termina cuando la ganancia de información baja de un límite preestablecido.

1.3. Ejemplo

Para mostrar de manera visual cómo funciona el algoritmo, se ha usado el conjunto de datos que se vio en clase (ver Figura 3). El atributo que se va a discretizar ya está ordenado. En este ejemplo, el algoritmo parará cuando se llegue a un máximo de 3 intervalos.

Atributo	Clase
1.3	A
1.4	A
1.4	A
1.5	A
4	B
4.5	B
4.7	B
4.9	B
5.1	C
5.6	C
5.9	C
6	C

Figura 3: Conjunto de datos

1.3.1. Primera iteración

En la primera iteración se divide el atributo en los dos intervalos que se muestran en la Figura 4. La **quanta matrix** se encuentra en la Figura 6.

La ganancia de información se calcula de la siguiente forma:

$$INFO(C,D|F) = \frac{1}{12}\log_2\left(\frac{\frac{1}{12}}{\frac{1}{12}}\right)+\frac{0}{12}\log_2\left(\frac{\frac{1}{12}}{\frac{0}{12}}\right)+\frac{0}{12}\log_2\left(\frac{\frac{1}{12}}{\frac{0}{12}}\right)+\frac{3}{12}\log_2\left(\frac{\frac{11}{12}}{\frac{3}{12}}\right)+\frac{4}{12}\log_2\left(\frac{\frac{11}{12}}{\frac{4}{12}}\right)+$$
$$\frac{4}{12}\log_2\left(\frac{\frac{11}{12}}{\frac{4}{12}}\right) = 0+0+0+0{,}467+0{,}486+0{,}486 = 1{,}44$$

Atributo	Clase
1.3	A
1.4	A
1.4	A
1.5	A
4	B
4.5	B
4.7	B
4.9	B
5.1	C
5.6	C
5.9	C
6	C

Figura 4: Intervalos de la primera iteración

	[1.3,1.4)	[1.4,6]	TOTAL CLASES
A	1	3	4
B	0	4	4
C	0	4	4
TOTAL INTERVALOS	1	11	12

Figura 5: Quanta matrix de la primera iteración

1.3.2. Segunda iteración

En esta segunda iteración se prueba con un nuevo intervalo, pero sigue habiendo 2 intervalos en total (ver Figura 4). La **quanta matrix** de esta iteración se puede ver en la Figura 6.

La ganancia de información se calcula de la siguiente forma:

$$INFO(C,D|F) = \frac{3}{12}\log_2\left(\frac{\frac{3}{12}}{\frac{3}{12}}\right)+\frac{0}{12}\log_2\left(\frac{\frac{3}{12}}{\frac{0}{12}}\right)+\frac{0}{12}\log_2\left(\frac{\frac{3}{12}}{\frac{0}{12}}\right)+\frac{3}{12}\log_2\left(\frac{\frac{9}{12}}{\frac{3}{12}}\right)+\frac{4}{12}\log_2\left(\frac{\frac{9}{12}}{\frac{4}{12}}\right)+\frac{4}{12}\log_2\left(\frac{\frac{9}{12}}{\frac{4}{12}}\right) = 1{,}044$$

Atributo	Clase
1.3	A
1.4	A
1.4	A
1.5	A
4	B
4.5	B
4.7	B
4.9	B
5.1	C
5.6	C
5.9	C
6	C

Figura 6: Intervalos de la segunda iteración

	[1.3,1.5)	[1.5,6]	TOTAL CLASES
A	3	1	4
B	0	4	4
C	0	4	4
TOTAL INTERVALOS	3	9	12

Figura 7: Quanta matrix de la segunda iteración

1.3.3. Siguientes iteraciones y resultado

El proceso de las siguientes iteraciones es similar pero probando diferentes intervalos. Separando el atributo en únicamente dos intervalos, la ganancia de entropía que se obtiene se muestra en la Figura 8. Se puede ver que la mayor ganancia de información se obtiene con los intervalos [1.3, 4.9) y [4.9, 6].

Atributo	Clase	
1.3	A	1.44
1.4	A	1.044
1.4	A	1.044
1.5	A	0.92
4	B	0.88
4.5	B	0.92
4.7	B	0.88
4.9	B	0.67
5.1	C	1.044
5.6	C	1.27
5.9	C	1.44
6	C	1.58

Figura 8: Ganancia de información con dos intervalos

Pero el algoritmo debe seguir porque el criterio de parada es que pare cuando se hayan probado hasta un número de intervalos de 3. Como son muchas operaciones y muchas divisiones de intervalos las que se deben realizar, se va a proceder a probar con la división que hace que la ganancia de información sea 0 y, por lo tanto, el resultado que va a dar este algoritmo.

En algún momento el algoritmo probará con la división de intervalos que se muestra en la Figura 9 y cuya **quanta matrix** se muestra en la Figura 10.

La ganancia de información se calcula de la siguiente manera:

$$INFO(C, D|F) = \frac{4}{12}\log_2\left(\frac{\frac{4}{12}}{\frac{4}{12}}\right) + \frac{4}{12}\log_2\left(\frac{\frac{4}{12}}{\frac{4}{12}}\right) + \frac{4}{12}\log_2\left(\frac{\frac{4}{12}}{\frac{4}{12}}\right) = 0$$

El resultado es que el algoritmo discretiza el atributo con los siguientes intervalos:

[1.3, 4), [4, 5.1), [5.1, 6]

Atributo	Clase
1.3	A
1.4	A
1.4	A
1.5	A
4	B
4.5	B
4.7	B
4.9	B
5.1	C
5.6	C
5.9	C
6	C

Figura 9: División de intervalos de la iteración con mayor ganancia de información

	[1.3, 4)	[4, 5.1)	[5.1, 6]	TOTAL CLASES
A	4	0	0	4
B	0	4	0	4
C	0	0	4	4
TOTAL INTERVALOS	4	4	4	12

Figura 10: Quanta matrix de la iteración con mayor ganancia de información

2. CADD (Class-Attribute Dependence Discretizer)

Este método, cuyas siglas son **CADD**, está basado en el concepto de la dependencia Clase-Atributo. Propuesto en 1995 por J.Y. Ching; A.K.C [2]. La idea principal de este algoritmo es la de maximizar la dependencia mutua que existe entre los intervalos discretos y las etiquetas de la clase, medido por su redundancia de interdependencia. De esta forma se puede determinar el número óptimo de intervalos para una aplicación de aprendizaje inductivo, por ejemplo, que son aquellas que mejor se adaptan a los intervalos discretos generados por este algoritmo.

Usa una matriz de 2 dimensiones con las frecuencias de cada instancia de cada clase en cada intervalo, también conocida como **quanta matrix** (la misma matriz que se usó en el algoritmo IEM, Figura 1). Se usa para calcular la probabilidad conjunta de que un objeto pertenezca a una clase determinada cuando el valor de un atributo cae en un determinado rango.

2.1. Características del algoritmo

Podemos clasificar el algoritmo con las siguientes características [3]:

- **Algoritmo supervisado:** tiene en cuenta las etiquetas de las clases como se ha estado comentando.
- **Estático:** realiza la discretización antes que cualquier otro proceso de DM. (Clasificación, por ejemplo).
- **Univariado:** trabaja con un solo atributo para realizar la discretización.
- Parte de una partición inicial y se va iterando para mejorar/reducir esos intervalos, por lo que también es **incremental**.
- **Paramétrico:** Requiere partir de un número de intervalos indicado por el usuario.

2.2. El criterio de discretización

La idea de este algoritmo es maximizar la función de la Figura 11 conocido como **Redundancia de interdependencia clase-atributo**,

$$R_{CAj} = \frac{I(C; A_j)}{H(C, A_j)}$$

Figura 11: Función a maximizar

donde:

$$I(C; A_j) = \sum_{C} \sum_{A_j} p_{sr} \log \frac{p_{sr}}{p_{s+} \cdot p_{+r}}$$

Es la información mutua entre clase y atributo. Y,

$$H(C, A_j) = -\sum_{C} \sum_{A_j} p_{sr} \log p_{sr}$$

es la entropía conjunta de ambos valores, siendo p_{sr} la probabilidad de que una determinada clase s pertenezca al intervalo r, que se puede calcular como $p_{sr} = \frac{q_{sr}}{M}$.

Con este método se busca maximizar la relación de dependencia entre la clase y un atributo continuo. Por lo que tendríamos que comprobar el **valor de redundancia de interdependencia CA** (clase-atributo) para todos los posibles intervalos que se puedan crear en el rango $[a, b]$ y quedarnos con aquellos que maximicen este valor.

El problema de este algoritmo es que es altamente combinatorio y es impracticable implementarlo directamente en un ordenador para todo el rango de valores y probabilidades del atributo ha discretizar en términos de coste computacional. Por lo que para su implementación se propone una una implementación heurística que es efectiva y eficiente a partes iguales. Esta implementación se divide en tres grupos principalmente: **inicialización de intervalos, mejora de intervalos y reducción de intervalos.**

El primer paso es ordenar los valores continuos en orden creciente. Posteriormente se selecciona un número de intervalos, que puede ser seleccionado por el usuario, o calculado por $\frac{M}{NxS}$ donde N es un parámetro, M el número de muestras y S el número de clases. Una vez establecidos los

intervalos, se procede a construir la primera quanta matrix y a calcular la primera medida R_{CAj} para las etiquetas de las clases y los intervalos.

La segunda fase tiene como objetivo mejorar los intervalos creados en la primera, mejorando la medida de redundancia de interdependencia inicial. Para ello se realizan pequeñas perturbaciones entre los intervalos. Para ello, se puede ajustar tanto el extremo superior como el extremo inferior del intervalo.

Para asegurar una buena estimación de una interdependencia global óptima, el algoritmo modifica cada límite hacía arriba y luego hacía abajo de cada intervalo comenzando por el primero. En cada modificación se guarda el cambio de interdependencia. Después de realizar todas las posibles combinaciones y calcular sus respectivos valores de interdependencia, se determina que ajuste causa la mayor ganancia de interdependencia y se modifica los intervalos, así como la quanta-matrix conforme a estos datos. Todo el proceso se repite hasta que no se produzcan nuevas mejoras.

La última parte del algoritmo, la reducción de intervalos, como su nombre indica se trata de buscar intervalos que estadísticamente no sean significantes o sean redundantes y puedan combinarse. Para ello, el algoritmo extrae dos pares de intervalos adyacentes y le aplica un test estadístico de interdependencia [2]. Si dos intervalos adyacentes no contribuyen en la dependencia clase-atributo son combinados en un único intervalo. El algoritmo es aplicado a todos los pares de algoritmos hasta que todos ellos pasan el test.

2.3. Ejemplo

Debido a la naturaleza de este algoritmo es impracticable realizar un ejemplo paso a paso del proceso que sigue, debido a que es eminentemente heurístico y tiene muchas causístcas, por lo que hasta para el ejemplo más pequeño pueden generarse una gran cantidad de pasos y cálculos hasta dar con la discretización final. De esta forma, se va a realizar un ejemplo hablando solamente de los pasos generales.

En este caso práctico tenemos un conjunto de datos con 85 muestras. Estas se dividen en una clase con 6 etiquetas. Siguiendo el algoritmo, en el primer paso generamos X intervalos iniciales calculados como $\frac{85}{3+6} = 4$. *3 es un valor usual para el parámetro N. Se partirá el atributo continuo en 4 partes con una distribución lo más parecida posible. Esta partición se puede ver en la Figura 12.

La $IC(C; A_j)$ inicial es 0.161820 y la $R_{CAj} = 0.055825$. Después de varias iteraciones se modifican algunos intervalos y se incrementan ambos valores de información mútua CA y redundancia de interdependencia a $IC(C; A_j) = 0.239157$ y $R_{CAj} = 0.09579$. La quanta matrix asociada a este punto se puede ver en la Figura 13.

Clases	Intervalos				Total
	0-4.7	4.8-4.9	5-5	5.1-7.1	
1	7	2	1	5	15
2	2	6	6	5	19
3	0	7	4	6	17
4	0	5	3	7	15
5	0	4	3	4	11
6	0	3	2	3	8
Total	9	27	19	30	85

Figura 12: Valores iniciales

Clases	Intervalos				Total
	0-4.7	4.8-4.8	4.9-5.3	5.4-7.1	
1	7	1	4	3	15
2	2	2	12	3	19
3	0	3	14	0	17
4	0	2	13	0	15
5	0	3	8	0	11
6	0	2	6	0	8
Total	9	13	57	6	85

Figura 13: Quanta matrix

Por último, se puede comprobar como los intervalos 2 y 3 tienen una distribución parecida y realizando el test estádistico de interdependencia se puede comprobar que ambos intervalos se pueden combinar sin perder interdependencia CA. Después de la reducción, se vuelve a pasar el algoritmo de modificación de intervalos y se comprueba que ya no obtenemos mejoras, consiguiendo de esta manera la discretización final que vemos en la Figura 14.

Clases	Intervalos			Total
	0-4.7	4.8-5.3	5.4-7.1	
1	7	5	3	15
2	2	14	3	19
3	0	17	0	17
4	0	15	0	15
5	0	11	0	11
6	0	8	0	8
Total	9	70	6	85

Figura 14: Discretización final

3. CAIM (Class Attribute Interdependence Maximization)

El algoritmo **CAIM**, propuesto en 2004 por Lukasz A. Kurgan y Krzysztof J. Cios [4], discretiza un atributo continuo en el menor número de intervalos discretos y máximiza la interdependencia clase atributo. El algoritmo selecciona el número óptimo de intervalos sin necesitar supervisión del usuario. Parte de la base de lo que veíamos en el algoritmo **CADD** [2], pero incluye diversas mejoras que lo hacen más eficiente y óptimo. También usa la información de interdependencia clase atributo como criterio para una discretización óptima, pero ligeramente modificado llamado **Class-Attribute Interdependence Uncertainty** (CAIU).

Usa básicamente todos los conceptos descritos en el algoritmo anterior, tales como la matriz de frecuencias (quanta matrix) y las dependencias clase atributo, pero reduce considerablemente el coste computacional al partir de un único intervalo inicial e ir dividiéndolo en sub-intervalos en cada iteración usando un criterio que, aunque siga siendo heurístico, es menos costoso que que el anterior.

El criterio que sigue es el siguiente:

$$CAIM(C, D|F) = \frac{\sum_{r=1}^{n} \frac{max_r^2}{M_{+r}}}{n}$$

donde n es el número de intervalos, r la iteración sobre todos los intervalos ($r = 1, 2, \ldots, n$), max_r es el máximo valor de todos los q_{ir} (columna r de la matriz ; $i = 1, 2, \ldots, S$) y M_{+r} es el número total de valores del atributo a discretizar en el intervalo $(d_{r-1}, d_r]$.

3.1. Características del algoritmo

- **Supervisado:** tiene en cuenta las etiquetas de las clases como se ha estado comentando.
- **Estático:** realiza la discretización antes que cualquier otro proceso de DM. (Clasificación, por ejemplo).
- **Univariado:** trabaja con un solo atributo para realizar la discretización.
- No parte de ninguna partición, o si se quiere ver de otra forma, parte de una única partición global que comprende el extremo superior y el extremo inferior del rango de valores del atributo continuo, $[a, b]$. En cada iteración va dividiendo los intervalos de la iteración anterior en sub-intervalos (**Top-down**), por lo que es **incremental**.

3.2. Algoritmo

En la Figura 15 podemos ver el pseudocódigo del algoritmo. Básicamente partimos de un conjunto de valores continuos del atributo a discretizar ordenados. Creamos el primer intervalo como [d0, dm] y establecemos el $GlobalCAIM$ a 0. Para todos los posibles subintervalos de D calculamos, su valor CAIM. Una vez calculados todos los posibles valores cogemos aquel que sea de un resultado mayor y si supera al $GlobalCAIM$ lo actualizamos y actualizamos los intervalos D. El algoritmo termina cuando no obtenemos ninguna mejora al realizar más particiones en los intervalos.

Algorithm 1 CAIM

Input: S conjunto de clases con l clases
Input: M conjunto de ejemplos clasificados cada uno a una única clase de S
Input: F = un atributo continuo para discretizar
Output: D, discretización de F

1: Ordenar valores de F, $F = \{d_0, d_1, \ldots, d_m\}$.
2: $D \leftarrow [d_0, d_n]$
3: GlobalCAIM $\leftarrow 0$
4: **for** $0 < k \leq l$ **do**
5: Calcular el valor CAIM para todas las particiones en subintervalos de D añadiendo un intervalo más
6: CAIMmax $\leftarrow$ valor máximo de CAIM para una partición D'
7: **if** CAIMmax $>$ GlobalCAIM **then**
8: GlobalCAIM $\leftarrow$ CAIMmax
9: $D \leftarrow D'$
10: **else** Terminar bucle
11: **return** D

Figura 15: Algoritmo CAIM [5]

4. Comparativa CADD vs CAIM

Examinando la comparativa realizada en [6] (ver Figura 16) podemos ver las diferencias entre estos dos métodos de discretización. Se han usado los siguientes datasets para realizar la comparativa:

1. Statlog Project Heart Disease dataset (hea)
2. Pima Indians Diabetes dataset (pid)
3. Thyroid Disease dataset (thy)
4. Waveform dataset (wav)

Criterio	Algoritmo	Dataset			
		thy	wav	hea	pid
CAIR	CADD	0.03	0.07	0.09	0.06
	CAIM	0.17	0.13	0.14	0.08
Número de intervalos	CADD	80	627	56	96
	CAIM	18	63	12	16
Tiempo (s)	CADD	628.64	8287.80	2.03	27.86
	CAIM	103.47	1143.91	0.31	4.95

Figura 16: Comparación de algoritmos

Los criterios elegidos para comparar el rendimiento de los métodos han sido el valor CAIR, el número de intervalos y el tiempo de ejecución. El primer criterio seguido para comparar está basado en el valor CAIR, que evalua la redundancia en la interdependencia atributo-clase, que es uno de los criterios que siguen estos algoritmos para crear los intervalos realmente.

Como ya se hablaba en la propia descripición de los algoritmos, CAIM resuelve muchos de los problemas que tenía CADD en términos de coste computacional y eficiencia, algo que se puede comprobar en los experimentos, ya que si nos fijamos en la tabla, supera a este en los tres criterios de comparación, tanto en el valor CAIR, como el número de intervalos (menor) y el tiempo de ejecución.

5. PKID

La idea principal de este método de discretización es conseguir que el número de intervalos coincida con el número de valores que contiene cada intervalo, es decir, suponiendo que tenemos 9 valores, hablamos de tres intervalos en los que tendremos 3 valores dentro de cada uno de estos intervalos.

Esto nos lleva a entender que, si tenemos un atributo con N valores, entonces el número de intervalos k será igual a la raíz cuadrada de N, $k = \sqrt{N}$

Encontramos la posibilidad de tener valores repetidos para el atributo. Todos los intervalos tendrán k valores distintos de forma que todos los valores iguales se incluirán en el intervalo que se colocó el primero. Esto nos lleva a entender que los únicos casos en los que se permite tener un número de valores superiores a k es por valores repetidos o para ajustar el último intervalo, ya que no siempre k es divisor de N.

5.1. Ejemplo sin valores repetidos

Supondremos un atributo "Nota" que representa las notas finales de 10 alumnos de la asignatura TID. Los datos serían los siguientes: (ver Figura 17).

Nota	4,75	5,6	7,2	6,8	4,25	8,9	10	5,75	9,75	6,3
Índice	1	2	3	4	5	6	7	8	9	10

Figura 17: Notas finales de 10 alumnos

Lo primero que haremos será encontrar el valor de K. Como nuestra muestra es de 10 elementos, nuestro número de intervalos será $k = \sqrt{10} = 3{,}1622 = 3$

Con esto sabemos que, en el caso ideal de que 3 fuese divisor de 10 y no teniendo repetidos, tendríamos 3 intervalos con 3 valores cada uno. Esto no es así ya que 3 no es divisor de 10 por lo que esperamos dos conjuntos de 3 elementos y uno de 4.

El siguiente paso será ordenar los elementos de forma ascendente (ver Figura 18).

Ahora se tendrán los siguientes intervalos de discretización (ver Figura 19).

Nota	4,25	4,75	5,6	5,75	6,3	6,8	7,2	8,9	9,75	10
Índice	5	1	2	8	10	4	3	6	9	7

Figura 18: Elementos ordenados de forma ascendente

Nota	4,25	4,75	5,6	5,75	6,3	6,8	7,2	8,9	9,75	10
Índice	5	1	2	8	10	4	3	6	9	7
Valor discretizado	1			2			3			

Figura 19: Intervalos de discretización

El último valor, con índice 7, será denominado **remanente**, y es un valor que propicia un número de elementos o valores mayor a k, en el último intervalo, por reajuste de los mismos como se ha explicado anteriormente.

5.2. Ejemplo con valores repetidos

En la Figura 20 se muestran las notas finales de 10 alumnos, pero esta vez con valores repetidos.

Nota	4,75	9,5	6	5,6	7,75	6	9,5	6	9,5	6
Índice	1	2	3	4	5	6	7	8	9	10

Figura 20: Notas finales de 10 alumnos

Como en el caso anterior, lo primero será calcular k de la misma forma: $k = \sqrt{10} = 3{,}1622 = 3$

El siguiente paso será ordenar los elementos de forma ascendente (ver Figura 21).

Nota	4,75	5,6	6	6	6	6	7,75	9,5	9,5	9,5
Índice	1	4	3	6	8	10	5	2	7	9

Figura 21: Datos ordenados de forma ascendente

Vemos los intervalos generados, teniendo en cuenta que se esperan 3 intervalos de 3 elementos cada uno (ver Figura 22). Vemos que, debido al número de valores repetidos, tenemos 2 intervalos y no 3 como se esperaba. El motivo es que sabemos que todos los valores repetidos tendrán que ir en un mismo intervalo.

Nota	4,75	5,6	6	6	6	6	7,75	9,5	9,5	9,5
Índice	1	4	3	6	8	10	5	2	7	9
Valor discretizado	1						2			

Figura 22: Intervalos discretizados

6. FFD

Es habitual cuando utilizamos técnicas de discretización pedir a la persona que va a usar el algorítmo que proporcione el número de intervalos que tendrá la discretización. En el caso del el algoritmo **FFD** esto no ocurre. El usuario, por el contrario, tendrá que decidir el número de valores o elementos que quiere que tenga cada intervalo.

Sabiendo esto podemos deducir que para obtener k, siendo k el número de intervalos, tendremos que dividir el tamaño del conjunto a discretizar N entre la frecuencia proporcionada por el usuario f, quedando la formula de la siguiente manera y teniendo en cuenta que siempre redondearemos el valor de k a la alza en caso de aparecer decimales en su cálculo: $k = \frac{N}{f}$

En este caso también tendremos un número fijo de elementos para cada intervalo, pero a diferencia del método anterior, el último intervalo siempre tendrá un número de elementos menor o igual a k.

Tras tener los valores de f, para el posterior cálculo de k, se ordena el conjunto de datos a discretizar de forma ascendente y se clasifican en intervalos del tamaño resultante.

6.1. Ejemplo sin valores repetidos

Como hemos hecho anteriormente, en este caso, para comprobar que la existencia de valores repetidos no supone ninguna modificación en el algoritmo, intentaremos discretizar los conjuntos de notas del método de discretización anterior.

Nos encontramos con los siguientes datos: (ver Figura 23).

Nota	4,75	5,6	7,2	6,8	4,25	8,9	10	5,75	9,75	6,3
Índice	1	2	3	4	5	6	7	8	9	10

Figura 23: Notas finales de 10 alumnos

Vamos a realizar este ejemplo con $f = 3$, obteniendo el siguiente valor para k: $k = \frac{10}{3} = 3{,}3333 = 4$

Viendo el resultado, redondeando al alza como se dijo anteriormente, deducimos que tendremos 4 intervalos de tamaño 3, pudiendo ser menor a 3 el tamaño del último intervalo. Ordenamos ahora

el conjunto (ver Figura 24).

Nota	4,25	4,75	5,6	5,75	6,3	6,8	7,2	8,9	9,75	10
Índice	5	1	2	8	10	4	3	6	9	7

Figura 24: Elementos ordenados de forma ascendente

En la Figura 25 se muestra cómo ha quedado la discretización de la característica nota.

Nota	4,25	4,75	5,6	5,75	6,3	6,8	7,2	8,9	9,75	10
Índice	5	1	2	8	10	4	3	6	9	7
Valor discretizado	1			2			3			4

Figura 25: Intervalos de discretización

6.2. Ejemplo con valores repetidos

En la Figura 26 es muestran los datos con valores repetidos. En este caso se definirá el valor de $f = 2$.

Nota	4,75	9,5	6	5,6	7,75	6	9,5	6	9,5	6
Índice	1	2	3	4	5	6	7	8	9	10

Figura 26: Notas finales de 10 alumnos

El valor de k que se obtiene aplicando la fórmula es $k = \dfrac{10}{2} = 5$.

Sabemos que habrá 5 intervalos con 2 elementos, además, como 2 es múltiplo de 10, el último intervalo también será de 2 elementos. Pasemos a ordenar el conjunto (ver Figura 27).

Nota	4,75	5,6	6	6	6	6	7,75	9,5	9,5	9,5
Índice	1	4	3	6	8	10	5	2	7	9

Figura 27: Elementos ordenados

En la Figura 28 se muestran los intervalos de discretización que se han obtenido.

Nota	4,75	5,6	6	6	6	6	7,75	9,5	9,5	9,5
Índice	1	4	3	6	8	10	5	2	7	9
Valor discretizado	1		2		3		4		5	

Figura 28: Intervalos de discretización

Referencias

[1] Natalie Meurer. A simple guide to entropy-based discretization. `https://natmeurer.com/a-simple-guide-to-entropy-based-discretization/`, 2015.

[2] John Y. Ching, Andrew K. C. Wong, and Keith C. C. Chan. Class-dependent discretization for inductive learning from continuous and mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):641–651, 1995.

[3] Salvador Garcia, Julian Luengo, José Antonio Sáez, Victoria Lopez, and Francisco Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):734–750, 2012.

[4] Lukasz A Kurgan and Krzysztof J Cios. Caim discretization algorithm. *IEEE transactions on Knowledge and Data Engineering*, 16(2):145–153, 2004.

[5] G. Navarro. Preparación de datos - tratamiento inteligente de datos. 2020.

[6] Lukasz Kurgan and Krzysztof Cios. Discretization algorithm that uses class-attribute interdependence maximization. *Proceedings of the 2001 International Conference on Artificial Intelligence (IC-AI 2001)*, 06 2001.