Práctica 3B (Práctica Final): Construcción de un sistema de clasificación basado en reglas difusas

Ingeniería del Conocimiento

Curso académico 2022-2023

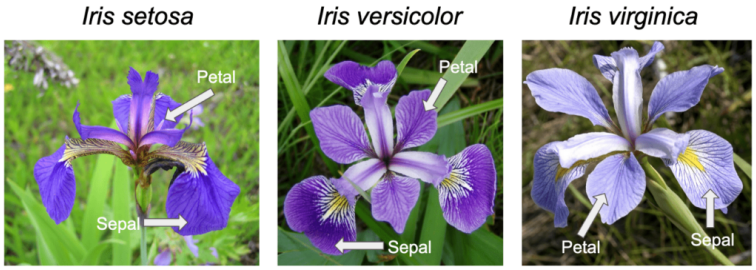
1 Introducción

En esta práctica final vamos a construir un sistema de clasificación basado en reglas difusas. Para ello, seguiremos los conceptos explicados en teoría y la práctica constará de dos secciones diferenciadas así como una última de comparativa y razonamiento. De esta manera las partes de la práctica serán:

1. Construcción del algoritmo de generación de reglas
2. Construcción del algoritmo de clasificación
3. Comparativa de resultados

1.1 Conjunto de datos

Antes de la construcción del algoritmo, lo primero que deberemos realizar será la carga del conjunto de datos. En esta práctica vamos a utilizar un conjunto de datos pequeño y estándar, llamado Iris [1]. Se trata de un conjunto de datos de 150 ejemplos (flores) donde cada uno de ellos está conformado por cuatro variables numéricas (largo sepalo, ancho sepalo, largo pétalo, ancho pétalo). Cada ejemplo corresponde a una de las tres clases (denominadas setosa, versicolor y virginica).



1.2 Particionado del conjunto de datos

Se utilizará todo el conjunto de datos para la generación de las particiones de las etiquetas lingüísticas. Para la generación de reglas se utilizará el conjunto train 50% del conjunto de datos (teniendo en cuenta que exista el mismo número de ejemplos para cada clase). Y el resto del conjunto de datos (50%) para el conjunto test. El particionado viene hecho en el código entregado.

[1] R. A. Fisher, "The use of multiple measurements in taxonomic problems". Annals of Eugenics 7 (2): 179-188. https://onlinelibrary.wiley.com/doi/10.1111/j.1469-1809.1936.tb02137.x

2 Generación de reglas

La parte de generación de reglas la podemos dividir en tres subapartados:

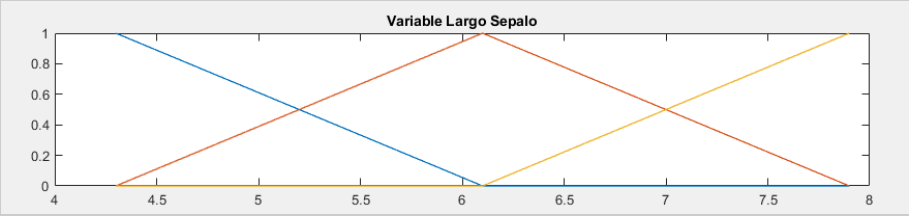
1. Generación de las particiones de etiquetas lingüísticas
2. Generación de una regla para cada ejemplo
3. Eliminación de reglas redundantes

2.1 Generación de las particiones de etiquetas lingüísticas

Para cada una de las variables utilizaremos las letras A, B, C, D en función de la ordenación dada en la introducción, que es la misma en la que se encuentran en el conjunto de datos. De esta manera, las etiquetas lingüísticas para cada uno serán A_1 para el bajo, A_2 para el medio y A_3 para el alto. De forma análoga con los B, C, D. Para cada variable generaremos:

- Un referencial distinto para cada uno, al que llamaremos R_A, R_B, R_C y R_D. Los referenciales, para cada variable irán desde el valor del elemento mínimo hasta el máximo con incrementos de 0.1. Las variables de entrada siempre tendrán un decimal. Se recomienda redondear los referenciales a un decimal mediante la función `round(·, 1)` para evitar problemas en el uso de la función `find()` para encontrar la posición de un valor concreto en el vector.
- Tres etiquetas lingüísticas (bajo, medio, alto). De esta manera, las etiquetas lingüísticas para cada uno serán A_1 para el bajo, A_2 para el medio y A_3 para el alto. De forma análoga con los B, C, D.
- Al igual que en la Práctica 3A, cada variable difusa (A, B, C, D), como está formada por tres etiquetas lingüísticas, se almacenará como una matriz de 3 filas (una por etiqueta: bajo, medio, alto) y tantas columnas como tenga el referencial concreto. Por ejemplo, la matriz `A` tendrá dimensiones $3 \times longitud R_A$. Para ello se pueden construir cada una de las filas (conjuntos) como `conjuntos.A(1,:) = ...` para la primera, y así sucesivamente, o bien hacer tres vectores y concatenarlos mediante `;`.
- Cada una de ellas estará definida por un conjunto difuso cuya función de pertenencia sea triangular. Los puntos donde la pertenencia sea 1 (es decir, el pico del triángulo, el punto más alto) será de la siguiente manera para cada uno: para el conjunto bajo (A_1), el elemento mínimo del conjunto de datos; para el conjunto medio (A_2), la media aritmética entre el elemento mínimo y el elemento máximo; y para el conjunto alto (A_3), el elemento máximo.

Ejemplo. Si para la variable largo sepalo el mínimo es 4.3, el máximo 7.9 y la media entre estos dos 6.1, las particiones que obtendremos serán las siguientes:



Construir una función llamada `construirConjuntos` que tome como entrada todo el conjunto de datos (es decir, `dataset.X`), y devuelva como salida la estructura `conjuntos`, formada por las variables `R_A, R_B, R_C, R_D, A, B, C, D`.

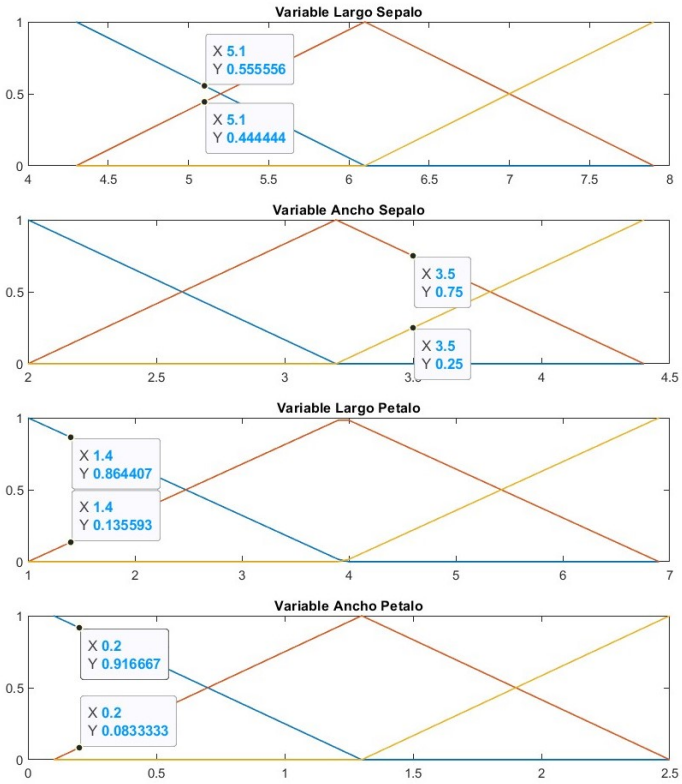
NOTA: Para generar una estructura con variables internas en MATLAB únicamente hay que escribir el nombre de la estructura, un punto y a continuación el nombre de la variable. Por ejemplo, si queremos generar una estructura llamada `conjuntos` y dentro la variable `R_A`, escribiremos `conjuntos.R_A = ...`

2.2 Generación de reglas

Para la construcción de la base de reglas, realizaremos en primer lugar una regla para cada ejemplo. De esta manera, los pasos a seguir serán los siguientes.
Para cada ejemplo. Pongamos el ejemplo $(a, b, c, d) = (5.1, 3.5, 1.4, 0.2)$, de la clase 1.

1. Calcular los grados de pertenencia asociado a las etiquetas. Cada ejemplo tiene cuatro valores, uno asociado a cada variable difusa. Es decir,

$$\begin{array}{lll} A_1(5.1) = 0.5556 & A_2(5.1) = 0.4444 & A_3(5.1) = 0.0000 \\ B_1(3.5) = 0.0000 & B_2(3.5) = 0.7500 & B_3(3.5) = 0.2500 \\ C_1(1.4) = 0.8644 & C_2(1.4) = 0.1356 & C_3(1.4) = 0.0000 \\ D_1(0.2) = 0.9167 & D_2(0.2) = 0.0833 & D_3(0.2) = 0.0000 \end{array}$$



NOTA: para utilizar el índice del referencial se recomienda utilizar la función `find()`. Por ejemplo, si queremos buscar el valor `5.1` dentro del referencial que lleva como nombre de variable `conjuntos.R_A`, haremos `find(conjuntos.R_A==5.1)` y nos devolverá el valor del referencial, para posteriormente pasárselo a cada etiqueta lingüística de A.

2. Asignar el ejemplo a la etiqueta con mayor grado para cada variable:

$$A_1(5.1) = 0.5556, \quad B_2(3.5) = 0.7500, \quad C_1(1.4) = 0.8644, \quad D_1(0.2) = 0.9167$$

3. Estudiar el grado de certeza de cada regla. Calcular una t-norma de los grados máximos a cada variable.
   - Utilizaremos como T-norma el producto: $0.5556 * 0.7500 * 0.8644 * 0.9167 = 0.3302$.
   - También podemos utilizar otro tipo de T-normas, como el mínimo: $\min\{0.5556, 0.7500, 0.8644, 0.9167\} = 0.5556$.
   - Por tanto, **generar la regla**: SI largo sepalo BAJO (A_1) Y ancho sepalo MEDIO (B_1) Y largo pétalo BAJO (C_1) Y ancho pétalo BAJO (D_1), ENTONCES clase 1 con $r = 0.3302$.
   - En código: `[1 2 1 1 0.3302 1]`

2.3 Eliminación de reglas redundantes

Hasta ahora se ha generado una matriz de reglas con 6 columnas (las primeras 4, una para cada variable, la quinta el grado de la regla y la sexta la clase) y 120 filas (el número de ejemplos del conjunto de train). Sin embargo, como podremos observar, al generarse una regla por ejemplo, se generan reglas repetidas, cuya única variación es el peso de la regla, teniendo por tanto, repetidos los elementos `1:4` de las filas. Por tanto, eliminaremos todas las reglas redundantes iguales, quedándonos con la de mayor peso.

NOTA: La función de MATLAB `ismember(A,B,'rows')`, devuelve una vector booleano de las filas de `A` que sean iguales a `B` [2].

Construir la función `construirReglas` que reciba de entrada el conjunto de datos de train (es decir, `partition.X.train` y `partition.Y.train`) y la estructura `conjuntos` y que obtenga como salida la matriz de `reglas` La matriz de reglas tendra dimensiones $numReglas \times 6$.

3 Clasificación

A partir de la base de reglas construida, el proceso de clasificación de los ejemplos de tets consta de 5 fases:

1. Calcular el grado de compatibilidad
2. Calcular el grado de asociación
3. Calcular el grado de asociación por clases
4. Clasificación
5. Medir la clasificación: medidas de evaluación

[2] ismember - Array elements that are members of set array https://www.mathworks.com/help/matlab/ref/double.ismember.html

3.1 Grado de compatibilidad

En primer lugar, calcularemos para cada ejemplo, el grado de compatibilidad que tiene para cada regla. Pongamos que tenemos un $ejemplo = (a, b, c, d)$, como antes, el grado de compatibilidad será:

$$compatibilidad(ejemplo, R_k) = T\Big(A_k(a), B_k(b), C_k(c), D_k(d)\Big) \tag{1}$$

siendo R_k cada una de las reglas $k \in \{1, \ldots, numReglas\}$ de la base de reglas, y A_k, B_k, C_k, D_k cada una de las etiquetas lingüísticas de la regla k asociadas a las variables A, B, C, D. Es decir, cada regla R_k está formada por $(A_k, B_k, C_k, D_k, certeza_{R_k}, i(\text{clase}))$. Y como T-norma utilizaremos el producto:

$$compatibilidad(ejemplo, R_k) = A_k(a) \cdot B_k(b) \cdot C_k(c) \cdot D_k(d) \tag{2}$$

Ejemplo. Pongamos que tenemos el $ejemplo = (6.9, 31., 4.9, 1.5)$ de la clase 2.
Para la regla R_1 (asociada a la clase 1) que es la que hemos construido anteriormente:

$$compatibilidad(ejemplo, R_1) = A_1(6.9) \cdot B_2(3.1) \cdot C_1(4.9) \cdot D_1(1.5) = 0 \cdot 0.92 \cdot 0 \cdot 0 = 0$$

Pongamos que existen otras dos reglas y que las compatibilidades para las mismas R_2 (asociada a la clase 2) y R_3 (asociada a la clase 3) son:

$$compatibilidad(ejemplo, R_2) = 0.3$$

$$compatibilidad(ejemplo, R_3) = 0.1$$

Para cada ejemplo, la variable `compatibilidad` será un vector de $numReglas$ elementos.

3.2 Grado de asociación

En esta fase, calcularemos el grado de asociación entre el grado de compatibilidad y el grado de certeza de la regla, de la siguiente manera:

$$asociacion(ejemplo, R_k, i) = h\Big(compatibilidad(ejemplo, R_k, i), certeza_{R_k,i}\Big) \tag{3}$$

siendo R_k cada una de las reglas $k \in \{1, \ldots, numReglas\}$ y $i \in \{1, \ldots, numClases\}$. Nosotros como función h utilizaremos una implicación difusa, como por ejemplo, el producto.

$$asociacion(ejemplo, R_k, i) = compatibilidad(ejemplo, R_k) \cdot certeza_{R_k,i} \tag{4}$$

Ejemplo. Continuamos con el ejemplo anterior. Para cada regla, calculamos el grado de asociación. Pongamos que las reglas R_2 y R_3 tienen grado de certeza 0.8 y 0.3 respectivamente

$$asociacion(ejemplo, R_1, 1) = compatibilidad(ejemplo, R_1) \cdot certeza_{R_1,1} = 0 \cdot 0.3302 = 0$$

$$asociacion(ejemplo, R_2, 2) = compatibilidad(ejemplo, R_2) \cdot certeza_{R_2,2} = 0.3 \cdot 0.8 = 0.24$$

$$asociacion(ejemplo, R_3, 3) = compatibilidad(ejemplo, R_3) \cdot certeza_{R_3,3} = 0.1 \cdot 0.3 = 0.03$$

Para cada ejemplo, la variable `asociacion` será una matriz de $numReglas \times 2$, donde la primera columna será el grado de asociación y la segunda la clase asociada a cada regla.

3.3 Grado de asociación por clases

En esta fase, una vez calculados los grados de asociación para cada regla (recordemos que cada regla está asociada a una clase), vamos a agrupar todos los grados de asociación que sean mayores de 0 para cada una de las clases mediante una función de agregación f:

$$asocClase_i = f\Big(asociacion_{R_k,i}(ejemplo) \;\Big|\; asociacion_{R_k,i}(ejemplo) > 0\Big) \tag{5}$$

siendo R_k cada una de las reglas $k \in \{1, \ldots, numReglas\}$ y $i \in \{1, \ldots, numClases\}$. Como función de agregación f, utilizaremos la media aritmética.

Ejemplo. Siguiendo con el ejemplo anterior, tenemos un ejemplo para cada clase, por lo que no tendríamos que agregar.

$$(asocClase_1, asocClase_2, asocClase_3) = (0.00, 0.24, 0.03)$$

En el caso de que tuviéramos más de un ejemplo en cada clase agregaríamos mediante la media aritmética aquellos valores mayores que 0.

Para cada ejemplo, la variable `asocClase` será un vector de $= 3$ elementos.

3.4 Clasificación

Una vez tenemos cuál es el grado de asociación para cada clase, seleccionaremos aquella clase que mayor grado de asociación tenga, es decir:

$$clase = \arg \max_{i=1}^{numClases} asocClase_i \tag{6}$$

Ejemplo. El argumento del elemento máximo del vector $asocClase$ es 2, por tanto, clasificaríamos el ejemplo en la clase 2.

3.5 Métricas de evaluación

Una vez tenemos las clasificaciones realizadas, así como las clases reales de los ejemplos, mediremos cómo de bien ha clasificado los ejemplos nuestro clasificador basado en reglas difusas. Para ello, la medida más sencilla es el **accuracy**, la cual se calcula como

$$accuracy = \frac{\text{número de ejemplos clasificados correctamente}}{\text{número de ejemplos totales a clasificar}} \tag{7}$$

Sin embargo, el accuracy nos muestra simplemente cuántos se han clasificado bien, pero no nos da más información, como por ejemplo que ejemplos de que clase se han clasificado como ejemplos de

otra clase. Para ello, en el ámbito de la clasificación se puede utilizar la **matriz de confusión**. Consiste en una matriz donde cada columna de representa el número de predicciones de cada clase, mientras que cada fila representa a los ejemplos de la clase real.

Ejemplo. Un sistema de clasificación nos da una accuracy de 0.8 (80%). Podemos pensar que es un valor relativamente bueno sin más información. Sin embargo, si obtenemos la matriz de confusión

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	10	0	0
	Clase 2	0	10	0
	Clase 3	0	6	4

Podemos observar que más de la mitad de los ejemplos reales de la clase 3 han sido incorrectamente clasificados, y concretamente todos como de la Clase 2. A partir de la información que nos da la matriz de confusión, podemos extraer más conclusiones así como formas para mejorar nuestro sistema de clasificación.

Construir la función `clasificador` que tenga como entrada el conjunto de datos de test (es decir, `partition.X.test` y `partition.Y.test`) así como la matriz de `reglas` y la estructura `conjuntos`. Esta función devolverá el valor de accuracy (`acc`) y la matriz de confusión (`MC`). La variable `acc` será un número entre 0 y 1 (o entre 0% y 100% si mostramos el porcentaje) y la matriz `MC`, una matriz de 3×3.

4 Modificaciones y comparación

Una vez construido el algoritmo de generación de reglas así como el de clasificación, se pide implementar al menos dos modificaciones al algoritmo y realizar un estudio comparativo de los resultados para cada una de las combinaciones. Como ayuda, algunas de las modificaciones pueden ser:

- Utilización de otro tipo de funciones de pertenencia (gaussianas, trapezoidales, etc.).
- Generación de las funciones de pertenencia triangulares de otras formas.
- Modificación de la función de agrupación de las pertenencias máximas para calcular la certeza de las reglas.
- Utilización de otro conjunto de datos.
- Utilización de otro tipo de T-normas en el grado de compatibilidad.
- Modificar la función del grado de asociación
- Modificar la función de agregación utilizada en el grado de asociacion por clases
- Cualquier otra modificación que valore el/la alumno/a.

5 Entrega, memoria y evaluación

5.1 Memoria y código

Se entregarán dos ficheros.

(Fichero 1) Fichero comprimido `3C-Apellido1Apellido2.zip` con los ficheros `iris.mat`, `construirConjuntos.m`, `construirReglas.m`, `clasificador.m` y un `main.m` que sea un script en el que se ejecute el programa sin ningún parametro que fijar manualmente.

(Fichero 2) La memoria: `3C-Apellido1Apellido2.pdf`, que constará de distintas secciones.

1. Introducción y objetivos
2. Generación de reglas
   - Explicar brevemente cómo se han construido/implementado los conjuntos y reglas.
   - Mostrar las gráficas de las etiquetas lingüísticas construidas.
   - Mostrar la base de reglas construidas con la estructura siguiente o similar:

```
R1:  IF Largo_Sepalo IS BAJO AND Ancho_Sepalo IS MEDIO AND
       Largo_Petalo IS BAJO AND Ancho_Petalo IS BAJO
             THEN Clase 1 con Certeza = 0.50
```

3. Clasificación
   - Explicar brevemente cómo se han realizado el proceso de clasificación.
   - Mostrar los resultados de accuracy y matriz de confusion. Valoración de resultados.
4. Modificaciones del algoritmo
   - Explicar concisamente cuáles son las modificaciones que se van a realizar sobre el algoritmo.
   - Mostrar las figuras/tablas necesarias para realizar la comparación de resultados.
   - Valoración y conclusión de los resultados en comparativa.
5. Conclusión final del trabajo

NOTA: No introducir explicaciones copiadas directamente del guión de la práctica. Se valorará la claridad de la explicación y de los resultados así como su acompañamiento con tablas y figuras.

5.2 Entrega y evaluación

Se abrirá una tarea en la plataforma **MiAulario** con **fecha límite** el **30 de diciembre de 2023** . El trabajo consistirá el 60% de la parte práctica de la asignatura.

Código

A continuación se muestra la estructura del fichero `main.m`:

```
%% Cargar datos
load('iris.mat')
dataset.X = round(iris.features,1);
dataset.Y = (1:3)*[strcmp(iris.label,'Iris-setosa');
                   strcmp(iris.label,'Iris-versicolor');
                   strcmp(iris.label, 'Iris-virginica')];

%% Particiones de datos
partition.indexes.train = [1:25 51:75 101:125];
partition.X.train = dataset.X(partition.indexes.train,:);
partition.Y.train = dataset.Y(partition.indexes.train);

partition.indexes.test = [26:50 76:100 126:150];
partition.X.test = dataset.X(partition.indexes.test,:);
partition.Y.test = dataset.Y(partition.indexes.test);

%% Construir conjuntos difusos
conjuntos = construirConjuntos(dataset.X);

%% Construir Reglas
reglas = construirReglas(partition.X.train,
                         partition.Y.train, conjuntos);

%% Clasificar
[acc, MC] = clasificador(partition.X.test,
                         partition.Y.test, reglas, conjuntos);
```

NOTA: al realizar las funciones en otros ficheros, para evitar errores en MATLAB, las variables de entrada en la cabecera de la función no deberá tener puntos (es decir, llamar `XTrain` por ejemplo, en lugar de `partition.X.train`).