Construcción de un sistema de clasificación basado en reglas difusas

Julen Rodríguez Meneses

Índice

Introducción y objetivos

En este trabajo vamos a realizar un sistema clasificador de datos basados en reglas difusas. Para ello vamos a trabajar sobre el conjunto de datos "***Iris***". Este está basado en un conjunto de 4 variables (*largo sépalo, ancho sépalo, largo pétalo, ancho pétalo*) y 3 clases distintas (*iris setosa, iris versicolor, iris virginica*).

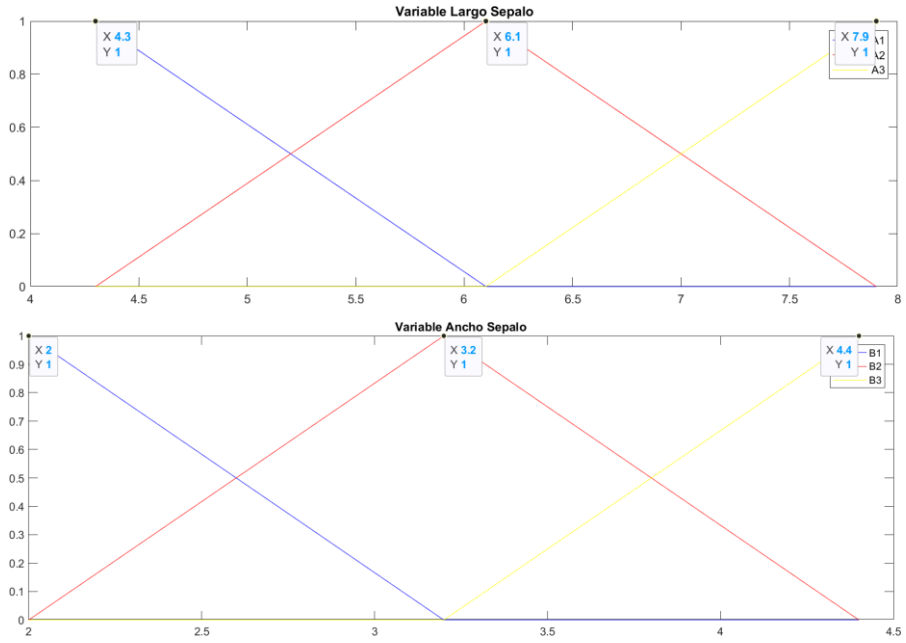
Lo que queremos conseguir finalmente es construir un programa que, dando unos datos sobre las distintas variables, este nos clasifique en una u otra clase. Para ello podemos tenemos varias posibilidades, pero nos hemos decantado finalmente por usar un tipo de regla difusa la cual la construiremos a nuestra conveniencia.

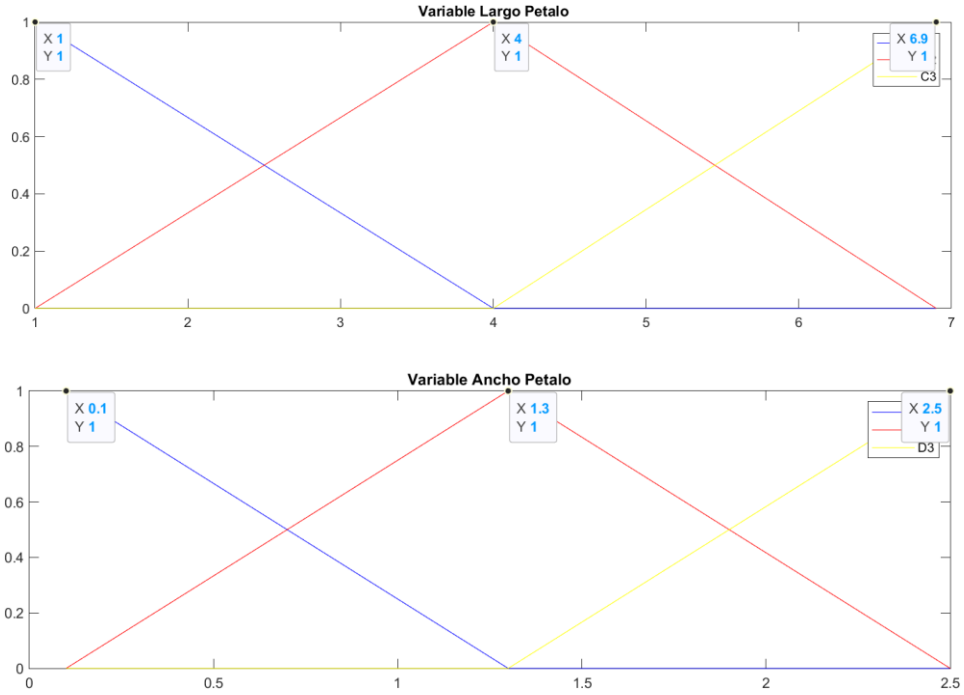
Generación de reglas

Lo primero a realizar sería una partición de datos en varios conjuntos, tanto de *train* como de *test*. Hemos realizado una división del 50%, una mitad de cantidad de datos en cada uno de los conjuntos (estamos seguros de que existen todas las clases en ambos). Vamos a realizar prácticamente la mayoría del trabajo sobre *train*.

Conjuntos difusos

Para construir las reglas difusas necesitamos realizar un trabajo previo sobre el conjunto de datos *train*. Construiremos **4 conjuntos difusos**, uno para cada tipo de variable. El referencial de cada uno transcurre desde el elemento mínimo al máximo con un paso de 0'1. Igualmente, **cada uno** de ellos está compuesto por **3 pertenencias triangulares** (con rango 0-1). Cada uno de ellos representando si su valor es **bajo** ('*X'1*), **medio** ('*X'2*) o **alto** ('*X'3*). El conjunto total de etiquetas lingüísticas construidos son los siguientes:





Conjunto de reglas

A continuación, construiremos el **conjunto de reglas**. Para ello, lo primero ha sido recorrer todo el conjunto de *Xtrain* (al igual que *Ytrain*) y mediante el referencial de cada variable, encontramos el argumento (índice del array *conjuntos*) donde se encuentran los valores de cada una en las diversas etiquetas. Una vez obtenidos estos índices, obtenemos las distintas pertenencias (bajo, medio o alto) sobre los distintos **atributos** y solo nos quedamos el de **mayor valor y su pertenencia**. Finalmente, con estos datos junto a la clase del elemento (el cual está incluido en el mismo orden el array de *Ytrain*), obtendríamos una regla definida. Conseguiríamos algo así:

Pert. A	Pert. B	Pert. C	Pert. D	Clase
1	2	1	1	1
1	2	1	1	1
1	2	1	1	1
1	2	1	1	1
1	2	1	1	1
:	:	:	:	:
2	2	3	3	3
2	2	2	3	3
3	2	3	3	3
2	2	2	2	3
2	2	3	3	3

Ejemplos en orden del conjunto train

Se puede comprobar perfectamente como en cada variable se presenta su pertenencia más alta y a que clase pertenece. No es muy difícil comprobar que existen varias reglas repetidas. Necesitamos quedarnos solamente con una y **eliminar las reglas redundantes**. La única manera de diferenciarlas es mediante el grado de certeza de cada una de las reglas. Para ello usaremos como **T-norma** el **producto**. Lo único que tenemos que hacer es multiplicar la pertenencia de cada variable de cada ejemplo. A un valor mayor, es mejor esa regla. Finalmente nos quedaríamos con aquellas que poseen mayor valor.

Pert. A	Pert. B	Pert. C	Pert. D	Grado Certeza	Clase
1	2	1	1	0.80556	1
2	3	1	1	0.48611	1
2	2	1	1	0.3579	1
2	2	2	2	0.66667	2
2	1	2	2	0.59028	2
1	1	2	2	0.38194	2
2	2	3	3	0.56194	3
3	2	3	3	0.41507	3
2	2	3	2	0.21456	3
3	2	3	2	0.23132	3
2	1	3	2	0.1408	3
2	2	2	3	0.38314	3
2	1	2	3	0.1734	3
3	1	3	3	0.37037	3

Conjunto final de reglas

Mediante esta forma obtenemos unas reglas distintas y definitivas con el mayor grado de certeza en cada una.

Estas reglas, una vez ya construidas, poseen una la siguiente estructura. **[TERMINA]**

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

Clasificación

En cuanto al proceso de clasificación de los datos, lo primero de todo es clasificar los ejemplos del conjunto *Xtrain* e *Ytrain* para entrenar el algoritmo y más tarde hacer ejemplos con los conjuntos de *test*.

Grado de compatibilidad

El primer paso por realizar es buscar el **grado de compatibilidad** entre cada ejemplo del conjunto con cada una de las reglas. Para ello usaremos la **T-norma** del **producto**. Lo único a realizar es comparar cada uno de regla en regla de la siguiente manera:

1. Clasificar cada atributo del ejemplo con su correspondiente pertenencia según marque la regla a seguir.
2. Una vez clasificados, multiplicarlos y conseguir la compatibilidad final.

Conseguiremos un array de una dimensión con cada columna representando el grado de compatibilidad con cada una de las reglas. Todo en el mismo orden de los conjuntos originales.

Grado de asociación

Seguidamente, necesitaremos calcular el **grado de asociación** de cada ejemplo con cada una de las reglas. El procedimiento por realizar es simplemente realizar una **multiplicación** (la cual es una **T-norma**) entre el grado de compatibilidad (calculado anteriormente) y el grado de certeza de cada una de las reglas. Al realizar los pasos anteriores en orden, esta operación no tendría ningún error ya que la compatibilidad calculada es la referida a la certeza de la regla operada.

Finalmente obtendremos un array de dos dimensiones, siendo cada fila cada una de las reglas y dos columnas siendo una la operación misma y la otra la clase representante. Se obtendrá un array por cada ejemplo calculado.

Grado de asociación por clases

Por tercer lugar, lo que tendríamos que hacer es utilizar el array de asociaciones para, como bien dice el nombre, averiguar la **asociación por clases**. Para ello lo que tenemos que realizar es algo tan simple como realizar una **función de agregación**, la cual nosotros usaremos la **media aritmética**, entre cada clase.

Lo que obtendremos finalmente es un array de tres elementos, cada uno de ellos representando la agregación con cada clase.

Clasificación

El último paso a realizar en cuanto a la clasificación de nuestro algoritmo en cada ejemplo es simplemente elegir el **índice máximo del array de agregación** por clases.

Con esto finalmente tendríamos un ejemplo clasificado con nuestro algoritmo propio.

Métricas de evaluación

Lo que resta es simplemente comparar nuestros resultados con los originales para calcular el porcentaje de acierto. Para ello, hemos calculado dos tipos de variables las cuales son **acc** y una matriz 3x3 **MC (Matriz de Confusión)**. La primera de ellas es simplemente la cantidad de acierto del algoritmo calculada como $\frac{Número\ de\ ejemplos\ clasificados\ correctamente}{Número\ total\ de\ ejemplos}$.
Esta operación nos da un resultado final de **84%** de acierto.

Por otro lado, mediante la matriz obtenemos un resultado como el siguiente:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	25	0	0
	Clase 2	0	25	0
	Clase 3	0	12	13

Gracias a estos datos podemos observar de donde proviene el porcentaje de acierto final. Se trata de un tanto porciento bastante alto por lo que podemos pensar que acierta muchos de los ejemplos a predecir, lo cual es cierto hasta cierto punto. Se pueden determinar el 100% de ejemplos tanto de la clase 1 como de la clase 2. Esto indica que hemos creado un gran modelo. En cambio, el problema aparece cuando se quiere predecir un ejemplo el cual pertenece a la **clase 3**. La mitad de ellos se predicen como clase 2, dando un **52% de acierto**. Este porcentaje no es muy alto por lo que no podemos estar en ningún momento seguros que lo que se haya dictaminado finalmente mediante el algoritmo sea cierto o no.

Como conclusión podemos dictaminar que el porcentaje final de aciertos general es bastante alto como para darlo como un buen predictor del conjunto *Iris*. En cambio, esto solo es una confianza ciega, una vez vista la matriz de confusión podemos darnos cuenta de que no tiene un gran acierto como se pensaba. Debido al problema prediciendo cualquier ejemplo de la clase 3, podemos dictaminar finalmente que el algoritmo **NO es un buen predictor**.

Modificaciones

**Entre modificaciones usaremos los datos originales y solo cambiaremos lo mencionado dentro de las mismas.*

1ª Modificación: Redistribuir partición *train-test*

Como primera modificación para poder conseguir un mayor porcentaje de acierto, como puede ser mas claro, modificamos el porcentaje de datos que se encuentran en cada conjunto. Un **primer intento** hemos realizado lo siguiente:

- ***Train***: Un total de 100/150 ejemplos, un 66,67%.
- ***Test***: Un total de 50/150 ejemplos, un 33,33%

Hemos conseguido la siguiente matriz de confusión:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	25	0	0
	Clase 2	0	25	0
	Clase 3	0	0	0

Como es obvio, y se puede intuir, hemos conseguido un **100%** de acierto, lo cual parece perfecto. El problema es que esta más que claro donde se encuentra el fallo, no había ningún valor de la clase 3 a predecir. Hemos hecho **una mala distribución de datos**.

La **segunda distribución** realizada es la siguiente:

- ***Train***: Un total de 120/150 ejemplos, un 80%.
- ***Test***: Un total de 30/150 ejemplos, un 20%

En este caso hemos elegido usar un mayor porcentaje de datos para el conjunto de *train* y así poder generar unas reglas mucho más precisas. La matriz obtenida es la siguiente:

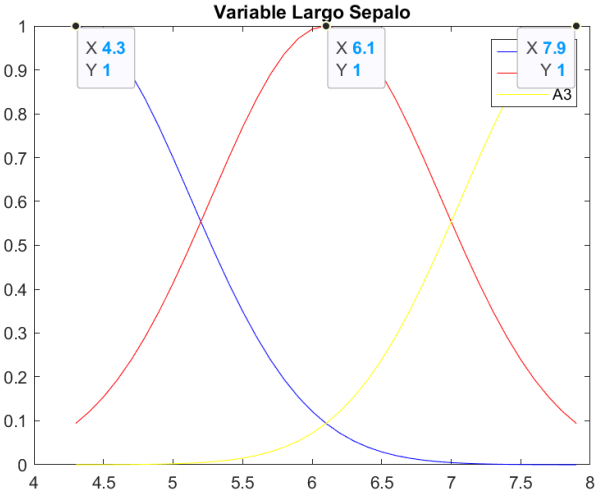
		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	10	0	0
	Clase 2	0	10	0
	Clase 3	0	4	6

Como podemos ver, nos hemos asegurado tener la misma cantidad de datos de cada clase, en este caso 10 de cada uno. Se ha obtenido un **86%** de acierto. Ciertamente es mayor que el obtenido originalmente. La verdadera cuestión es que el problema persiste de la misma manera. Los ejemplos acertados de la clase 3 son del **60%**, lo cual sigue siendo un porcentaje malo.

Damos por sabido que el problema no se ha corregido modificando los conjuntos. Vamos a intentar con otro tipo de modificaciones.

2ª Modificación: Modificación conjuntos difusos

Uno de los posibles grandes fallos puede ser la construcción de los conjuntos difusos. Para ello hemos hecho algunos cambios en su construcción. Primeramente, hemos probado a **construirlos de forma gaussiana**. Todos ellos nos han quedado de la siguiente manera:



Como se puede observar, los *"puntos"* más altos, séase los valores ciertos a un tipo de dato dentro del conjunto siguen siendo los mismos, la única diferencia viene siendo la construcción de la misma, no siendo tan drástico y dando un poco más de importancia a los valores vecinos. Finalmente, con este método hemos obtenido la siguiente matriz de confusión:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	25	0	0
	Clase 2	0	25	0
	Clase 3	0	10	15

Se obtiene un total de **86,7% de acierto**. Una clara mejoría sin duda, sin decir por supuesto que la predicción de la clase 3 también ha aumentado, ahora siendo de un **60%**. Pero todo esto sigue sin ser suficiente, vamos a probar con otro tipo de modificaciones.

3ª Modificación: Cambio de *T-norma* en la construcción de las reglas

Para ello vamos a probar 3 *T-normas* distintas. Estas, junto a sus resultados finales son las siguientes:

- **Mínimo**: Posible calcularla mediante a la asociatividad de las *T-normas*. Calculando de la siguiente manera $min(min(valA, valB), min(valC, valD))$. Obtenemos los siguientes resultados:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	25	0	0
	Clase 2	0	25	0
	Clase 3	0	11	14

85,33% acierto

- **Lukasiewicz**: Igual que la calculada anteriormente, posible gracias a la regla asociativa. La operación realizada es la siguiente $max(0, max(0, valA + valB - 1) + max(0, valC + valD - 1) - 1)$. Los resultados son:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
_ Val. real -	Clase 1	25	0	0
	Clase 2	0	25	0
	Clase 3	0	13	12

82,67% acierto

- **Drástica**: Como en todas las anteriores, la asociatividad nos ayuda bastante. En este caso hemos construido la siguiente función para poder realizar la *T-norma* drástica:

```
function resultado = drastico(x,y)
    if(x==1)
        resultado = 1;
    elseif(y==1)
        resultado = 1;
    else
        resultado = 0;
    end
end
```

Igualmente, realizamos la siguiente función $drastico(drastico(valA, valB), drastico(valC, valD))$. Hemos obtenido lo siguiente:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	25	0	0
	Clase 2	0	25	0
	Clase 3	0	7	18

90.67% acierto

Tras haber visto 4 tipos de *T-normas* distintas (La original más 3 nuevas realizadas en esta modificación). Nos damos cuenta de que en todas las clases 2 y 3 se aciertan siempre, en cambio, sin ningún tipo de duda, podemos decir que la peor de todas es la *T-norma* de Lukasiewicz, teniendo en la clase 3 un **48% de acierto**. En cambio, la mejor, y el mejor resultado obtenido hasta ahora es usando la *T-norma* drástica, dando en la clase 3 un **72% de acierto**.

4ª Modificación: Cambiar la función de agregación entre clases

En esta modificación vamos a realizar una serie de cambios en las agregaciones entre clases. Vamos a probar 4 tipos de cambios distintos:

- **Integral Choquet**: Esta la hemos realizado de tal manera que realizamos esta función al momento de agregar las distintas clases:

```
function agregacion = integralChoquet(conjunto)
    agregacion = 0;
    aux = 0;
    conjunto = sort(conjunto,'ascend');
    for i = 1:length(conjunto)
        agregacion = agregacion+((conjunto(i)-aux)*(size(conjunto,2)-aux));
        aux = aux+1;
    end
end
```

Se han obtenido los siguientes resultados:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	0	20	5
	Clase 2	18	1	6
	Clase 3	21	4	0

13,3% acierto

- **OWA**: Se ha realizado mediante la función:

```
function agregacion = owa(conjunto,a,b)
    agregacion = 0;
    conjunto= sort(conjunto,'descend');
    for i = 1:length(conjunto)
        agregacion = agregacion+conjunto(i)*(operacionOWA(a,b,i/length(conjunto))-operacionOWA(a,b,(i-1)/length(conjunto)));
    end
end

function res = operacionOWA(a,b,x)
    if(x < a)
        res = 0;
    elseif(x > b)
        res = 1;
    else
        res = (x-a)/(b-a);
    end
end
```

Como se puede observar no hemos especifiado que variable *a* y *b* usar. Esto es debido a que usamos las 3 posibilidades que sabemos:

- **Al menos la mitad (*a*=0 *b*=0,5)**:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	25	0	0
	Clase 2	0	25	0
	Clase 3	0	13	12

82,67% acierto

- **La mayor parte (*a*=0,3 *b*=0.8)**:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	25	0	0
	Clase 2	0	25	0
	Clase 3	0	17	8

77,33% acierto

- **La mayor cantidad posible (*a*=0.5 *b*=1)**:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	25	0	0
	Clase 2	0	25	0
	Clase 3	0	18	7

76% acierto

Con todos estos datos podemos darnos cuenta que los datos obtenidos con la *Integral Choquet* son ciertamente nefastos, clasifica mal incluso las clases 1 y 2. Por otro lado, el *OWA* realizando "*Al menos la mitad*" obtiene un resultado decente, pero no el mejor, obteniendo de la clase 3 un total de **48% de acierto** de la **clase 3**.

5ª Modificación: Mezclar todo lo realizado anteriormente

En esta última modificación no vamos ha realizar otra cosa que mezclar las que se han realizado anteriormente. Para ello hemos realizado un nuevo código en el que introducimos que modificación querer realizar y obtener los datos predichos. Mediante esta función los máximos valores obtenidos han sido los siguientes:

		Valor predicho		
		Clase 1	Clase 2	Clase 3
Val. real	Clase 1	25	0	0
	Clase 2	0	25	0
	Clase 3	0	6	19

92% acierto

Esto ha sido obtenido usando los siguiente:

- Usar la **distribución original.**
- Creación de **conjuntos difusos Gaussianos.**
- Usar una ***T-norma* Drástica.**
- **Agregar** mediante un **OWA (Al menos la mitad)**

Ya no es que tenemos un muy buen porcentaje de acierto, sino que además acertamos un **76%** de la **clase 3**, lo cual es una gran mejora respecto al 52% original.

Conclusión del trabajo

Finalmente, podemos decir que la **mejora** realizada a base de las modificaciones es cuanto menos **grandiosa**. Esto se debe **principalmente** al **trabajo** en cuanto a los **conjuntos difusos**. La forma de construirlos influye bastante, pero sobre todo la manera de realizar las asociaciones entre ellas, tener cuidado que tipo de *T-norma* usar. En este caso, como se ha demostrado, usar la *T-norma* Drástica, como bien dice su nombre, influye obteniendo una mejora drástica de los resultados.