Práctica 6: Gestión de la incertidumbre

Sistemas de Información Distribuidos

Pedro Allué Tamargo (758267) Juan José Tambo Tambo (755742)

1 de mayo de 2021

Índice

1. Esfuerzos

- **Pedro**: Realización de la práctica, realización de la memoria.
  - **Tiempo invertido:** 4 horas.
- **Juan José**: Realización de la práctica, realización de la memoria.
  - **Tiempo invertido:** 4 horas.

2. Realización de la práctica

Para la realización de la práctica se ha creado un proyecto Java y se han importado las librerías de *JFML* y *javax.xml.bind* que son necesarias para la realización de la práctica. Se ha copiado el código que se adjunta al enunciado.
En primer lugar se ha definido la base de conocimiento y el sistema de inferencia mediante las clases `KnowledgeBaseType` y `FuzzyInferenceSystem`. Se puede osbervar su inicialización y asignación en el siguiente código:

```
FuzzyInferenceSystem fis = new FuzzyInferenceSystem("Pendulo invertido");
KnowledgeBaseType kb = new KnowledgeBaseType();
fis.setKnowledgeBase(kb);
```

En segundo paso se van a proceder a definir las variables de entrada. Estas variables son el *ángulo* y la *velocidad angular*. Se van a definir con la nomenclatura de las diapositivas de teoría. El código de definición de estas variables es el siguiente:

```
// Variable de entrada: angulo
FuzzyVariableType angulo = new FuzzyVariableType("Angulo", -180, 180);
angulo.setType("input");
kb.addVariable(angulo);
// Terminos linguisticos de Angulo
FuzzyTermType angNG = new FuzzyTermType("Negativo Grande",
    FuzzyTermType.TYPE_leftLinearShape, (new float[]{-30f, -15f}));
FuzzyTermType angNP = new FuzzyTermType("Negativo Pequeno",
    FuzzyTermType.TYPE_triangularShape, (new float[]{-30f, -15f, 0f}));
FuzzyTermType angZ = new FuzzyTermType("Cero",
    FuzzyTermType.TYPE_triangularShape, (new float[]{-15f, 0f, 15f}));
FuzzyTermType angPP = new FuzzyTermType("Positivo Pequeno",
    FuzzyTermType.TYPE_triangularShape, (new float[]{0f, 15f, 30f}));
FuzzyTermType angPG = new FuzzyTermType("Positivo Grande",
    FuzzyTermType.TYPE_rightLinearShape, (new float[]{0f, 30f}));
angulo.addFuzzyTerm(angNG);
angulo.addFuzzyTerm(angNP);
angulo.addFuzzyTerm(angZ);
angulo.addFuzzyTerm(angPP);
angulo.addFuzzyTerm(angPG);

// Variable de entrada velocidad angular
FuzzyVariableType velAngular = new FuzzyVariableType("Velocidad Angular", -1, 1);
velAngular.setType("input");
kb.addVariable(velAngular);
[...]
```

Se ha definido la variable de salida *velocidad de la plataforma* de forma análoga a las anteriores. La diferencia es que el tipo de variable es `output` y se debe indicar un mecanismo de agregación (mediante `setAccumulation()`) y un método de deffuzicación (mediante `setDefuzzifierName()`).

Por último se han definido las 11 reglas indicadas en las diapositivas de teoría. Las reglas se almacenan dentro de una colección de tipo `MamdaniRuleBaseType`, que es añadida al `FuzzyInferenceSystem` mediante `fis.addRuleBase(<rulebase>`

Para todas las reglas se ha seguido el mismo patrón. En primer lugar, se crea la regla con el constructor `FuzzyRuleType`, tal y como se indica en el guión de la práctica. Seguidamente, se debe indicar un antecedente y un consecuente a esat regla. Para ello, se crean estos objetos con `AntecedentType` y `ConsequentType` respectivamente. Las clausulas necesarias son indicadas mediante `addClause(new ClauseType(<VARIABLE>, <ETIQUETA>))`. Por útlimo, el antecedente y consecuente se ligan a la regla mediante `setAntecedent() y setConsequent()`.

Un ejemplo de creación de regla sería el siguiente:

```
// Regla 1  (Z, NG) -> NG
FuzzyRuleType r1 = new FuzzyRuleType("Regla 1", "and", "MIN", 1.0f);
AntecedentType ant1 = new AntecedentType();
ant1.addClause(new ClauseType(velAngular, velAngZ));
ant1.addClause(new ClauseType(angulo, angNG));
ConsequentType con1 = new ConsequentType();
con1.addThenClause(velocidad, velPlaNG);
r1.setAntecedent(ant1);
r1.setConsequent(con1);
rb.addRule(r1);
```

Para la realización del **Ejercicio 2**, se ha utilizado el mismo código que en el ejercicio anterior exceptuando que los valores iniciales de las variables *angulo* y *velAngular* se obtienen de la entrada estándar del programa mediante la clase *scanner*. Para ello, se ha añadido lo siguiente:

```
Scanner scanner = new Scanner(System.in);
System.out.print("Introduzca la velocidad angular del pendulo: ");
Float velocidadAngularEntrada = scanner.nextFloat();

System.out.print("Introduzca el angulo del pendulo: ");
Float anguloEntrada = scanner.nextFloat();
```

Por último, para el **Ejercicio 3**, se ha utilizado la clase *scanner* de nuevo para obtener la entrada del usuario sobre los métodos de agregación y deffuziación. Cabe destacar que se realiza una comprobación de que la entrada del usuario coincide con uno de los métodos válidos para ello (Por ejemplo, para el método de agregación, el usuario solo puede introducir “MAX” o “PROBOR”).

3. Urls de interés

- Documentación de *JFML* (enlace)
- Descarga del fichero *JAR* de *JFML* (enlace)
- Descarga del fichero *JAR* de `javax.xml.bind` (enlace)