対応しているファジィ制御

R_1 :

IF $P_{11}: x_{11} \in X_{11}$ is $A_{11} \subset X_{11}$ and ... $P_{1M(1)}: x_{1M(1)} \in X_{1M(1)}$ is $A_{1M(1)} \subset X_{1M(1)}$
THEN $y \in Y$ is $B_1 \subset Y$

というファジィ制御則が $R = \{R_1, R_2, ..., R_N\}$ だけあるようなもの。

基本的な用語と対応しているクラス、メソッド、フィールド

R	ファジィ制御	FuzzyInterface1
R_i	ファジィ制御則	Rule
P_{ij}	前件部のファジィ命題	AntecedentThesis
x_{ij}	入力値	InputValue
A_{ij}, B_j	ファジィ集合	FuzzySet の継承クラス
X_{ij}, Y	ファジィ集合の全体集合	FuzzySet.Range
y	結論値	FuzzyInterface1.getConsequent()
$\mu_A(x)$	メンバーシップ関数	FuzzySet.membershipFunction(x)
$\mu_R(x, y)$	ファジィ関係関数	RelationModel の継承クラス

ファジィ推論の流れ

IF $x_1 \in X_1$ is $A_1 \subset X_1$ and $x_2 \in X_2$ is $A_2 \subset X_2$ THEN $y \in Y$ is $B \subset Y$ を計算する。

- この1文は1つのRuleで表す。
- $x_1 x_2$ はInputValueで定義する。
- $X_1 X_2 Y$ はFuzzySet.Rangeで定義する。
- $A_1 A_2 B$ はFuzzySetから継承されたクラスで定義する。
- x_1 is A_1 はAntecedentThesisで定義する。
- FuzzyInterface1が複数のRuleを持つ。Ruleは前件部にあたる複数のAntecedentThesisと後件部にあたる1つのFuzzySetを持つ。AntecedentThesisは1つのInputValueへの参照と1つのFuzzySetを持つ。
- 後件部にあたる1つのFuzzySet(ファジィ集合 B)はconsequentFuzzySetとの名称でRuleで管理されている。

$A_1 A_2 B$はファジィ集合で、各メンバーシップ関数は$\mu_{A_1} : X_1 \to [0,1]$、$\mu_{A_2} : X_2 \to [0,1]$、$\mu_B : Y \to [0,1]$となる。

- ファジィ集合のメンバーシップ関数はFuzzySet.membershipFunctionが表している。このメソッドは、定義域(Xなど)と値域($[0,1]$)の違反をしていないか監視するFuzzySet.invokeMembershipFunctionによって実行される。

$x_1 = x_1^*$、$x_2 = x_2^*$との実測値が得られた時、$\omega = \mu_{A_1}(x_1^*) \wedge \mu_{A_2}(x_2^*)$となる。

- 上の式でのωはallAntecedentPartGoodnessとの名称でRuleで管理されている。
- InputValue.setValueにより実測値が代入されたら、後々の計算効率向上のためにも各Ruleはこの時点で上の計算をRule.updateAllAntecedentPartGoodnessにて行う。
- Rule.updateAllAntecedentPartGoodnessはFuzzyInterface1.inputValueChangedによって一斉実行が可能である。

ファジィ関係関数$\mu_R(x,y)$を用いて$\mu_B(y)$を予測する式は$\mu_B(y) = \bigcup_{x \in X}(\mu_A(x) \wedge \mu_R(x,y))$。

この式に実数値x^*を導入しそれに見合ったファジィ集合のメンバーシップ関数$\mu_{B^*}(y)$を求めるには、xを要素に持つクリスプ集合A^*を考えればよいので

$\mu_{B^*}(y) = \bigcup_{x \in X}(\mu_{A^*}(x) \wedge \mu_R(x,y)) = \mu_R(x^*, y)$となる。ファジィ関係関数は多種あるが、例えばMamdaniのファジィ関係モデルに従うと$\mu_R(x,y) = \mu_A(x) \wedge \mu_B(y)$。$x$については既に実測値が分かっていて、前件部の各ファジィ命題$A_1 A_2$をまとめ上げた結果ωがあるので、これを代入すると$\mu_R(\bullet, y) = \omega \wedge \mu_B(y)$となり、結論としては$\mu_{B^*}(y) = \mu_R(\bullet, y) = \omega \wedge \mu_B(y)$。

- ファジィ関係関数には多種あるためRelationModelという抽象クラスを作っている。Mamdamiのファジィ関係モデルに限らず、ほとんどのファジィ関係モデルは結論となる関数$\mu_{B^*}(y)$を求めるためにはωと$\mu_B(y)$しか必要ではない。
- よって、RelationModelはそれを所有するRuleからallAntecedentPartGoodnessとconsequentFuzzySetのみが提供される。
- RelationModelは複数のRuleに対して使いまわしてはいけない。allAntecedentPartGoodnessが各Ruleによって異なる値になるから。RelationModelFactoryを使うと便利。
- $\mu_{B^*}(y)$の関数はRelatonModel.getConsequentValueとRule.getConsequentValueとの名称となっている。

ルールが複数ある場合を考える。各ルールが示す結論$\mu_{B_i^*}(y)$を$i = 1, \ldots, N$まで結合する必要がある。この時、使用したファジィ関係モデルがMamdamiの場合はブール和結合し、そ

れ以外の場合はブール積結合することが多い。こうしてできあがった関数 $\mu_{B^*}(y)$ を Y の全区間において積分して重心を決め、非ファジイ化とする。 $f(y) = \mu_{B^*}(y)$ とすると重心 $y = \frac{\int f(y)ydy}{\int f(y)dy}$ で求まる。

- コンピューター上では連続的な積分はできないので離散的な総和によって擬似的な積分を実現する。これをするために、全ての数値が Y に入っているようなリストyValuesをFuzzyInterface1のコンストラクタによって提供する。
- 積分を離散化して総和 $\sum$ にする際は、積分の dy を Δy にしなければならないが、計算の高速化のために Δy を考えていない。よって、yValuesの各要素の値の差は同じでなければならない。
- FuzzyInterface1でも各Rulesでの結論となる関数 $\mu_{B^*}(y)$ の結合方法を指定する必要もあるが、RelationModelFactory.getPreferredCombinationModeBetweenRulesでも取得できる。