1. **Задание**

1. Программная реализация частного случая вывода Мамдани.

* 1. Прочитать справочную информацию об алгоритме Мамдани.
  2. Реализовать вывод Мамдани в соответствии с вариантами задания.
  3. Реализовать FIS – типа Мамдани в соответствии с вариантом используя Fuzzy Tool Box.
  4. Сравнить выходные значения программной реализации вывод Мамдани и полученные в Matlab в 50 точках, распределенных на всей области терм-множества с постоянным шагом.

Таблица 1 – Правила нечеткого вывода

|  |  |
| --- | --- |
| Вар. | Правила |
| 3 | Если T21 и T13 и T32 То X1;  Если T12 или T23 или T31 То X2;  Если T11 и T22 То X3; |

Таблица 2 – Указания к значениям T1, T2 и т.д из таблицы 1

|  |  |
| --- | --- |
| Вар. | Функции принадлежности термов |
| 3 | T1x – PI-S, T2x – Trian, T3x – Trian, X1 – Trian, X2 – Trian,  X3 – Trian. |

1. **Реализация с использованием Matlab**

На рисунках 1-6 представлена реализация варианта 2 в среде Matlab.

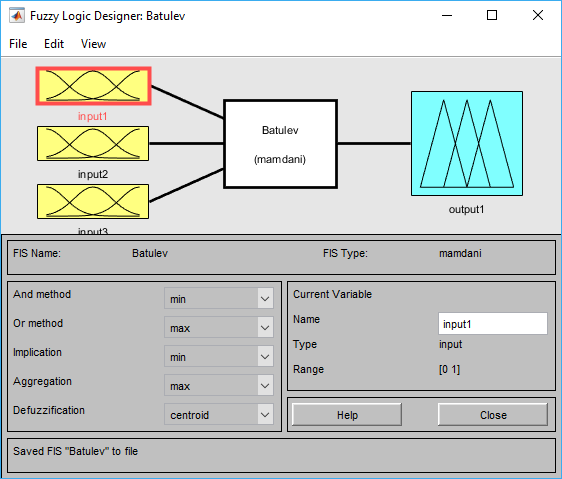


Рисунок 1 – Окно Fuzzy Logic Designer

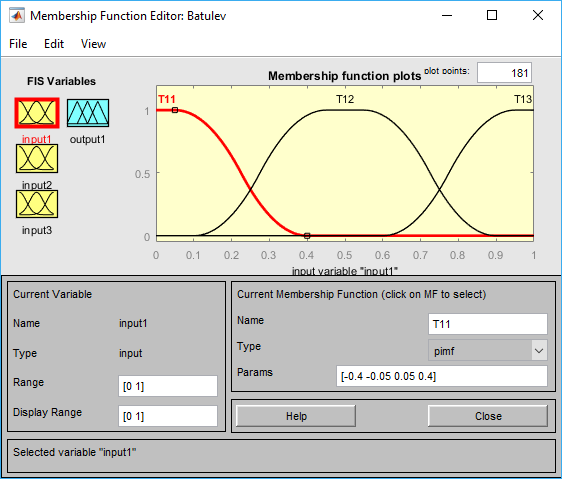
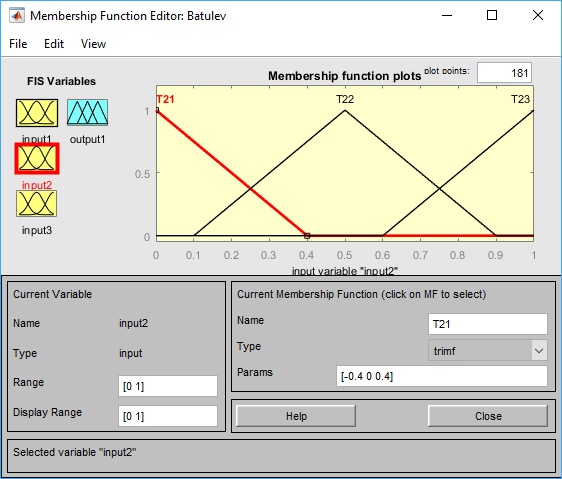
 

Рисунок 2 – Термы входной переменной input1

Рисунок 3 – Термы входной переменной input2

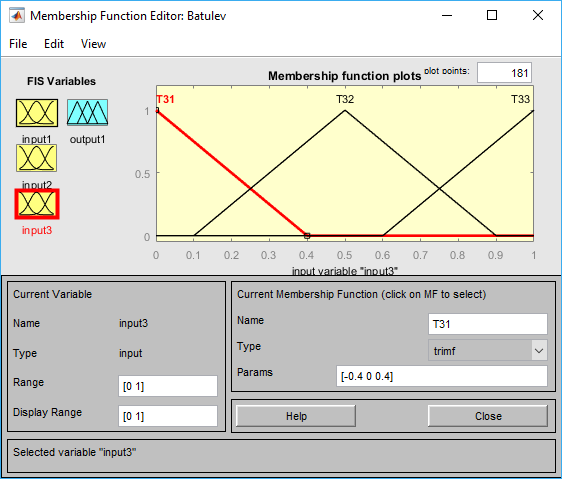
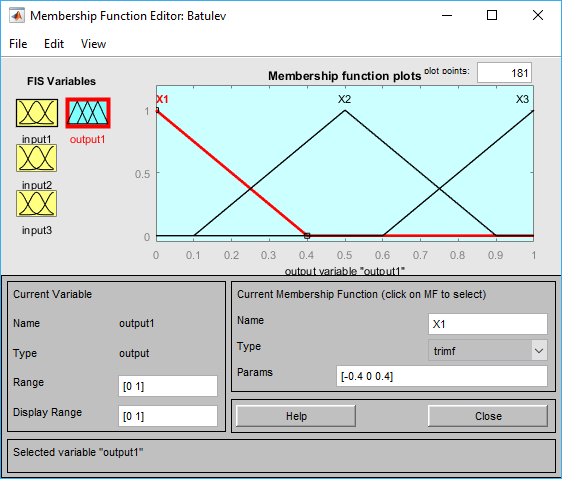
 

Рисунок 4 – Термы входной переменной input3

Рисунок 5 – Выходная

переменная output1

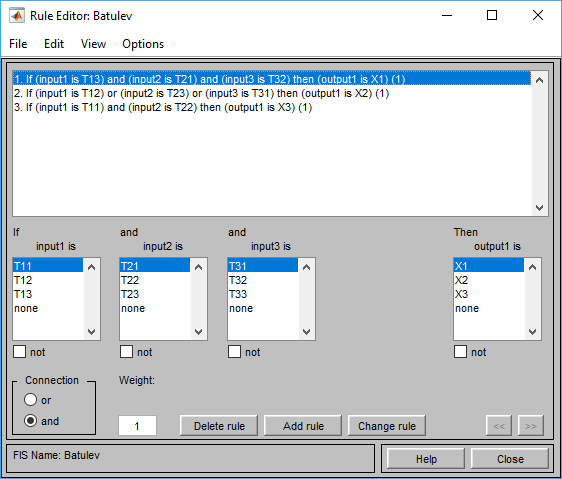
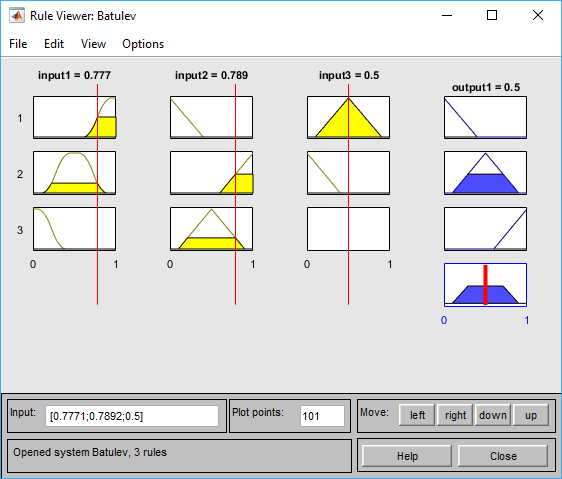
 

Рисунок 6 – Правила

Рисунок 7 – Логический вывод

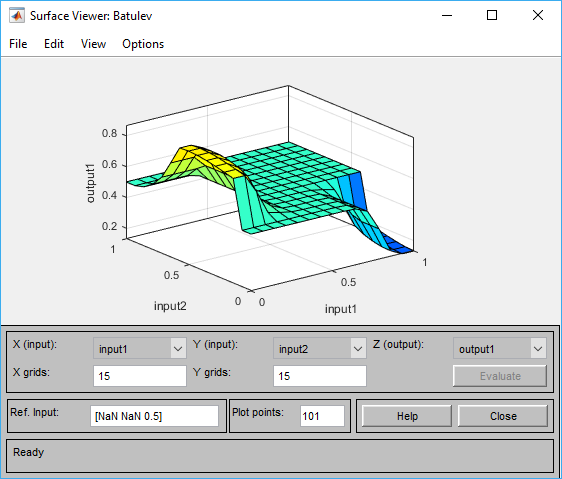


Рисунок 8 – Поверхность вывода

Таблица 3 – input1

|  |  |  |
| --- | --- | --- |
| name | type | params |
| T11 | pimf | [-0.4 -0.05 0.05 0.4] |
| T12 | pimf | [0.1 0.45 0.55 0.9] |
| T13 | pimf | [0.6 0.95 1.05 1.4] |

Таблица 4 input2

|  |  |  |
| --- | --- | --- |
| name | type | params |
| T21 | trimf | [-0.4 0 0.4] |
| T22 | trimf | [0.1 0.5 0.9] |
| T23 | trimf | [0.6 1 1.4] |

Таблица 5 input3

|  |  |  |
| --- | --- | --- |
| name | type | params |
| T31 | trimf | trimf |
| T32 | trimf | trimf |
| T33 | trimf | trimf |

Таблица 6 – output1

|  |  |  |
| --- | --- | --- |
| name | type | params |
| X1 | trimf | [-0.4 0 0.4] |
| X2 | trimf | [0.1 0.5 0.9] |
| X3 | trimf | [0.6 1 1.4] |

1. **Программная реализация вывода**

Программа выполнялась в среде разработки Visual Studio C#. На рисунке 9 представлен результат выполнения программы.

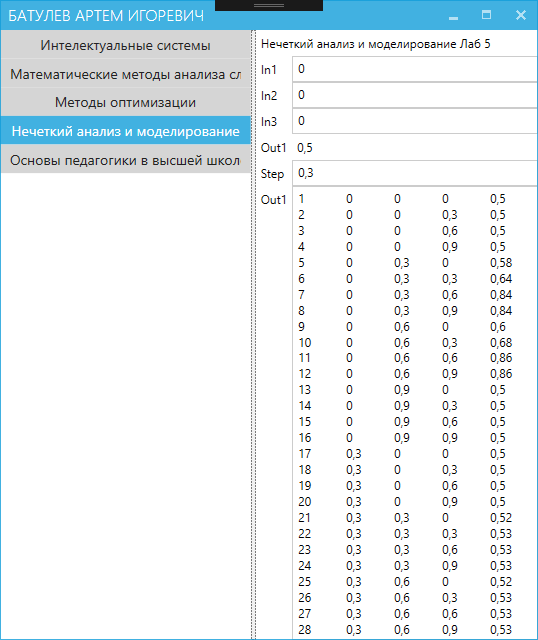


Рисунок 9 – Программная реализация

1. **Листинг программы**

Листинг программной реализации вывода Мамдани представлен в приложении А.

1. **Выходные значения**

Выходные значения программной реализации и с использованием Matlab представлены в таблице 7.

Таблица 7 – Сравнения выходных значений MATLAB и C#

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | input1 | unput2 | input3 | Matlab | C# |
| 1 | 0 | 0 | 0 | 0,5 | 0,5 |
| 2 | 0 | 0 | 0,3 | 0,5 | 0,5 |
| 3 | 0 | 0 | 0,6 | 0,5 | 0,5 |
| 4 | 0 | 0 | 0,9 | 0,5 | 0,5 |
| 5 | 0 | 0,3 | 0 | 0,578312 | 0,58 |
| 6 | 0 | 0,3 | 0,3 | 0,645676 | 0,64 |
| 7 | 0 | 0,3 | 0,6 | 0,847049 | 0,84 |
| 8 | 0 | 0,3 | 0,9 | 0,847049 | 0,84 |
| 9 | 0 | 0,6 | 0 | 0,603411 | 0,6 |
| 10 | 0 | 0,6 | 0,3 | 0,679881 | 0,68 |
| 11 | 0 | 0,6 | 0,6 | 0,86281 | 0,86 |
| 12 | 0 | 0,6 | 0,9 | 0,86281 | 0,86 |
| 13 | 0 | 0,9 | 0 | 0,5 | 0,5 |
| 14 | 0 | 0,9 | 0,3 | 0,5 | 0,5 |
| 15 | 0 | 0,9 | 0,6 | 0,5 | 0,5 |
| 16 | 0 | 0,9 | 0,9 | 0,5 | 0,5 |
| 17 | 0,3 | 0 | 0 | 0,5 | 0,5 |
| 18 | 0,3 | 0 | 0,3 | 0,5 | 0,5 |
| 19 | 0,3 | 0 | 0,6 | 0,5 | 0,5 |
| 20 | 0,3 | 0 | 0,9 | 0,5 | 0,5 |
| 21 | 0,3 | 0,3 | 0 | 0,523159 | 0,52 |
| 22 | 0,3 | 0,3 | 0,3 | 0,526554 | 0,53 |
| 23 | 0,3 | 0,3 | 0,6 | 0,526554 | 0,53 |
| 24 | 0,3 | 0,3 | 0,9 | 0,526554 | 0,53 |
| 25 | 0,3 | 0,6 | 0 | 0,523159 | 0,52 |
| 26 | 0,3 | 0,6 | 0,3 | 0,526554 | 0,53 |
| 27 | 0,3 | 0,6 | 0,6 | 0,526554 | 0,53 |
| 28 | 0,3 | 0,6 | 0,9 | 0,526554 | 0,53 |
| 29 | 0,3 | 0,9 | 0 | 0,5 | 0,5 |
| 30 | 0,3 | 0,9 | 0,3 | 0,5 | 0,5 |
| 31 | 0,3 | 0,9 | 0,6 | 0,5 | 0,5 |
| 32 | 0,3 | 0,9 | 0,9 | 0,5 | 0,5 |
| 33 | 0,6 | 0 | 0 | 0,5 | 0,5 |
| 34 | 0,6 | 0 | 0,3 | 0,5 | 0,5 |
| 35 | 0,6 | 0 | 0,6 | 0,5 | 0,5 |
| 36 | 0,6 | 0 | 0,9 | 0,5 | 0,5 |
| 37 | 0,6 | 0,3 | 0 | 0,5 | 0,5 |
| 38 | 0,6 | 0,3 | 0,3 | 0,5 | 0,5 |
| 39 | 0,6 | 0,3 | 0,6 | 0,5 | 0,5 |
| 40 | 0,6 | 0,3 | 0,9 | 0,5 | 0,5 |
| 41 | 0,6 | 0,6 | 0 | 0,5 | 0,5 |
| 42 | 0,6 | 0,6 | 0,3 | 0,5 | 0,5 |
| 43 | 0,6 | 0,6 | 0,6 | 0,5 | 0,5 |
| 44 | 0,6 | 0,6 | 0,9 | 0,5 | 0,5 |
| 45 | 0,6 | 0,9 | 0 | 0,5 | 0,5 |
| 46 | 0,6 | 0,9 | 0,3 | 0,5 | 0,5 |
| 47 | 0,6 | 0,9 | 0,6 | 0,5 | 0,5 |
| 48 | 0,6 | 0,9 | 0,9 | 0,5 | 0,5 |
| 49 | 0,9 | 0 | 0 | 0,5 | 0,5 |
| 50 | 0,9 | 0 | 0,3 | 0,354324 | 0,36 |
| 51 | 0,9 | 0 | 0,6 | 0,13719 | 0,14 |
| 52 | 0,9 | 0 | 0,9 | 0,195 | 0,2 |
| 53 | 0,9 | 0,3 | 0 | 0,5 | 0,5 |
| 54 | 0,9 | 0,3 | 0,3 | 0,422982 | 0,43 |
| 55 | 0,9 | 0,3 | 0,6 | 0,173662 | 0,18 |
| 56 | 0,9 | 0,3 | 0,9 | 0,195 | 0,2 |
| 57 | 0,9 | 0,6 | 0 | 0,5 | 0,5 |
| 58 | 0,9 | 0,6 | 0,3 | 0,5 | 0,5 |
| 59 | 0,9 | 0,6 | 0,6 | 0,5 | 0,5 |
| 60 | 0,9 | 0,6 | 0,9 | 0,5 | 0,5 |
| 61 | 0,9 | 0,9 | 0 | 0,5 | 0,5 |
| 62 | 0,9 | 0,9 | 0,3 | 0,5 | 0,5 |
| 63 | 0,9 | 0,9 | 0,6 | 0,5 | 0,5 |
| 64 | 0,9 | 0,9 | 0,9 | 0,5 | 0,5 |

**Приложение А –листинг программы**

FAAMMainViewModel.cs

using System;

using System.Linq;

using FuzzyLogic.FuzzySystem;

using FuzzyLogic.FuzzySystem.Mamdani;

using FuzzyLogic.LinguisticVariables;

using FuzzyLogic.Terms;

using Prism.Mvvm;

namespace FuzzyAnalysisAndModeling.ViewModel

{

public class FAAMMainViewModel : BindableBase

{

private IFuzzySystem \_fuzzySystem;

private string \_in1;

private string \_in2;

private string \_in3;

private string \_step;

public FAAMMainViewModel()

{

var input1 = new InputLinguisticVariable("in1", 0, 1);

var input2 = new InputLinguisticVariable("in2", 0, 1);

var input3 = new InputLinguisticVariable("in3", 0, 1);

input1.Terms.Add(new PiTerm("T11", -0.4, -0.05, 0.05, 0.4));

input1.Terms.Add(new PiTerm("T12", 0.1, 0.45, 0.55, 0.9));

input1.Terms.Add(new PiTerm("T13", 0.6, 0.95, 1.05, 1.4));

input2.Terms.Add(new TrianTerm("T21", -0.4, 0, 0.4));

input2.Terms.Add(new TrianTerm("T22", 0.1, 0.5, 0.9));

input2.Terms.Add(new TrianTerm("T23", 0.6, 1, 1.4));

input3.Terms.Add(new TrianTerm("T31", -0.4, 0, 0.4));

input3.Terms.Add(new TrianTerm("T32", 0.1, 0.5, 0.9));

input3.Terms.Add(new TrianTerm("T33", 0.6, 1, 1.4));

var output1 = new OutputLinguisticVariable("out1", 0, 1);

output1.Terms.Add(new TrianTerm("X1", -0.4, 0, 0.4));

output1.Terms.Add(new TrianTerm("X2", 0.1, 0.5, 0.9));

output1.Terms.Add(new TrianTerm("X3", 0.6, 1, 1.4));

var mamdani = new MamdaniFuzzySystem(new[] { input1, input2, input3 }, new[] { output1 });

mamdani.AddRule("if (in1 is T13) and (in2 is T21) and (in3 is T32) then (out1 is X1)");

mamdani.AddRule("if (in1 is T12) or (in2 is T23) or (in3 is T31) then (out1 is X2)");

mamdani.AddRule("if (in1 is T11) and (in2 is T22) then (out1 is X3)");

\_fuzzySystem = mamdani;

}

public string In1

{

get { return \_in1; }

set

{

\_in1 = value.ReplacePoint();

OnPropertyChanged(nameof(Out1));

}

}

public string In2

{

get { return \_in2; }

set

{

\_in2 = value.ReplacePoint();

OnPropertyChanged(nameof(Out1));

}

}

public string In3

{

get { return \_in3; }

set

{

\_in3 = value.ReplacePoint();

OnPropertyChanged(nameof(Out1));

}

}

public string Step

{

get { return \_step; }

set

{

\_step = value.ReplacePoint();

OnPropertyChanged(nameof(Results));

}

}

public string Results

{

get

{

double step;

if (!double.TryParse(\_step, out step))

{

step = 0.25;

}

if (step <= 0)

{

return string.Empty;

}

string result = String.Empty;

int iterator = 0;

for (double i = 0; i <= 1; i = i + step)

{

for (double j = 0; j <= 1; j = j + step)

{

for (double k = 0; k <= 1; k = k + step)

{

iterator++;

var r = String.Join(",", \_fuzzySystem.Evaluate(i, j, k).Select(x => x.Result));

result += $"{iterator}\t{i}\t{j}\t{k}\t{r}\r\n";

}

}

}

return result;

}

}

public string Out1 => Evaluate();

private string Evaluate()

{

double in1;

double in2;

double in3;

if (string.IsNullOrWhiteSpace(\_in1) || string.IsNullOrWhiteSpace(\_in2) || string.IsNullOrWhiteSpace(\_in3) ||

!double.TryParse(\_in1, out in1) || !double.TryParse(\_in2, out in2) || !double.TryParse(\_in3, out in3) ||

\_fuzzySystem == null)

{

return String.Empty;

}

return String.Join(",", \_fuzzySystem.Evaluate(in1, in2, in3).Select(x => x.Result));

}

}

}

MamdaniFuzzySystem.cs

using System.Collections.Generic;

using System.Linq;

using FuzzyLogic.FuzzySystem.Mamdani.Parser;

using FuzzyLogic.FuzzySystem.Mamdani.Rules;

using FuzzyLogic.LinguisticVariables;

namespace FuzzyLogic.FuzzySystem.Mamdani

{

public class MamdaniFuzzySystem : IFuzzySystem

{

private readonly InputLinguisticVariable[] \_inputLinguisticVariables;

private readonly OutputLinguisticVariable[] \_outputLinguisticVariables;

private readonly List<MamdaniFuzzyRule> \_rules;

public MamdaniFuzzySystem(InputLinguisticVariable[] inputLinguisticVariables, OutputLinguisticVariable[] outputVariables)

{

\_inputLinguisticVariables = inputLinguisticVariables;

\_outputLinguisticVariables = outputVariables;

\_rules = new List<MamdaniFuzzyRule>();

}

public IOutputLinguisticVariable[] Evaluate(params double[] inputVariables)

{

for (int i = 0; i < inputVariables.Length; i++)

{

\_inputLinguisticVariables[i].InputValue = inputVariables[i];

}

Fuzzify(\_inputLinguisticVariables)

.EvaluateConditions(\_rules, AndMethod.Min, OrMethod.Max)

.Implicate(ImplicationMethod.Min)

.Aggregate(AggregationMethod.Max, \_outputLinguisticVariables)

.Defuzzify(DefuzzificationMethod.Centroid);

return \_outputLinguisticVariables;

}

/// <summary>

/// Фаззификация входных переменных.

/// </summary>

/// <param name="inputVariables"></param>

/// <returns></returns>

private FuzzifyResult Fuzzify(params InputLinguisticVariable[] inputVariables)

{

return new FuzzifyResult(inputVariables);

}

public void AddRule(string rule)

{

\_rules.Add(RuleParser.Parse(rule, \_inputLinguisticVariables.ToList(), \_outputLinguisticVariables.ToList()));

}

}

}

LinguisticVariable.cs

using System.Collections.Generic;

using FuzzyLogic.Terms;

namespace FuzzyLogic.LinguisticVariables

{

public abstract class LinguisticVariable

{

private readonly List<Term> \_terms;

private readonly string \_name;

protected LinguisticVariable(string name, double min, double max)

{

\_terms = new List<Term>();

\_name = name;

Range = new Range

{

Min = min,

Max = max

};

}

public string Name => \_name;

public Range Range { get; set; }

public List<Term> Terms => \_terms;

}

}

InputLinguisticVariable.cs

namespace FuzzyLogic.LinguisticVariables

{

public class InputLinguisticVariable : LinguisticVariable

{

public InputLinguisticVariable(string name, double min, double max)

: base(name, min, max)

{

}

public double InputValue { get; set; }

}

}

OutputLinguisticVariable.cs

using System;

using FuzzyLogic.FuzzySystem;

namespace FuzzyLogic.LinguisticVariables

{

public class OutputLinguisticVariable : LinguisticVariable, IOutputLinguisticVariable

{

private double \_result;

public OutputLinguisticVariable(string name, double min, double max)

: base(name, min, max)

{

}

public double Result

{

get

{

return Math.Round(\_result, 2);

}

set { \_result = value; }

}

}

}

Range.cs

namespace FuzzyLogic.LinguisticVariables

{

public class Range

{

public double Min { get; set; }

public double Max { get; set; }

}

}

Term.cs

namespace FuzzyLogic.Terms

{

public abstract class Term

{

private readonly string \_name;

protected Term(string name)

{

\_name = name;

}

public string Name => \_name;

public abstract double GetValue(double x);

}

}

BellTerm.cs

using System;

namespace FuzzyLogic.Terms

{

public class BellTerm : Term

{

public BellTerm(string name)

: base(name)

{

}

public BellTerm(string name, double a, double b, double c)

: this(name)

{

this.a = a;

this.b = b;

this.c = c;

}

public double a { get; set; }

public double b { get; set; }

public double c { get; set; }

public override double GetValue(double x) => 1 / (1 + (Math.Pow((Math.Abs((x - c) / a)), 2 \* b)));

}

}

GaussTerm.cs

using System;

namespace FuzzyLogic.Terms

{

public class GaussTerm : Term

{

public GaussTerm(string name)

: base(name)

{

}

public GaussTerm(string name, double sigm, double c)

: this(name)

{

this.sigm = sigm;

this.c = c;

}

public double sigm { get; set; }

public double c { get; set; }

public override double GetValue(double x) => Math.Exp((-1 \* Math.Pow(x - c, 2)) / (2 \* Math.Pow(sigm, 2)));

}

}

PiTerm.cs

using System;

namespace FuzzyLogic.Terms

{

public class PiTerm : Term

{

public PiTerm(string name)

: base(name)

{

}

public PiTerm(string name, double a, double b, double c, double d)

: this(name)

{

this.a = a;

this.b = b;

this.c = c;

this.d = d;

}

public double a { get; set; }

public double b { get; set; }

public double c { get; set; }

public double d { get; set; }

public override double GetValue(double x)

{

if (x <= a)

{

return 0;

}

if (a <= x && x <= (a + b) / 2)

{

return 2 \* Math.Pow((x - a) / (b - a), 2);

}

if ((a + b) / 2 <= x && x <= b)

{

return 1 - 2 \* Math.Pow((x - b) / (b - a), 2);

}

if (b <= x && x <= c)

{

return 1;

}

if (c <= x && x <= (c + d) / 2)

{

return 1 - 2 \* Math.Pow((x - c) / (d - c), 2);

}

if ((c + d) / 2 <= x && x <= d)

{

return 2 \* Math.Pow((x - d) / (d - c), 2);

}

if (x >= d)

{

return 0;

}

throw new InvalidOperationException("Ни одно из условий не выполнилось.");

}

}

}

TrianTerm.cs

using System;

namespace FuzzyLogic.Terms

{

public class TrianTerm : Term

{

public TrianTerm(string name)

: base(name)

{

}

public TrianTerm(string name, double a, double b, double c)

: this(name)

{

this.a = a;

this.b = b;

this.c = c;

}

public double a { get; set; }

public double b { get; set; }

public double c { get; set; }

public override double GetValue(double x)

{

if (x <= a)

{

return 0;

}

if (a <= x && x <= b)

{

return (x - a) / (b - a);

}

if (b <= x && x <= c)

{

return (c - x) / (c - b);

}

if (c <= x)

{

return 0;

}

throw new InvalidOperationException("Ни одно из условий не выполнилось.");

}

}

}

FuzzifyResult.cs

using System;

using System.Collections.Generic;

using System.Linq;

using FuzzyLogic.FuzzySystem.Mamdani.Parser;

using FuzzyLogic.FuzzySystem.Mamdani.Rules;

using FuzzyLogic.LinguisticVariables;

using FuzzyLogic.Terms;

namespace FuzzyLogic.FuzzySystem.Mamdani

{

internal class FuzzifyResult

{

private readonly InputLinguisticVariable[] \_inputVariables;

private AndMethod \_andMethod;

private OrMethod \_orMethod;

public FuzzifyResult(params InputLinguisticVariable[] inputVariables)

{

\_inputVariables = inputVariables;

}

/// <summary>

/// Агрегирование подусловий.

/// </summary>

/// <param name="fuzzyRules"></param>

/// <param name="andMethod"></param>

/// <returns></returns>

public EvaluateConditionsResult EvaluateConditions(List<MamdaniFuzzyRule> fuzzyRules, AndMethod andMethod, OrMethod orMethod)

{

\_andMethod = andMethod;

\_orMethod = orMethod;

var result = new Dictionary<MamdaniFuzzyRule, double>();

var fuzzifiedInput = new Dictionary<LinguisticVariable, Dictionary<Term, double>>();

foreach (var inputLinguisticVariable in \_inputVariables)

{

fuzzifiedInput[inputLinguisticVariable] = inputLinguisticVariable.Terms.ToDictionary(t => t, term => term.GetValue(inputLinguisticVariable.InputValue));

}

foreach (MamdaniFuzzyRule rule in fuzzyRules)

{

result.Add(rule, EvaluateCondition(rule.Condition, fuzzifiedInput));

}

return new EvaluateConditionsResult(result);

}

protected double EvaluateCondition(ICondition condition, Dictionary<LinguisticVariable, Dictionary<Term, double>> fuzzifiedInput)

{

if (condition is Conditions)

{

double result = 0.0;

Conditions conds = (Conditions)condition;

if (conds.ConditionsList.Count == 0)

{

throw new Exception("Inner exception.");

}

else if (conds.ConditionsList.Count == 1)

{

result = EvaluateCondition(conds.ConditionsList[0], fuzzifiedInput);

}

else

{

result = EvaluateCondition(conds.ConditionsList[0], fuzzifiedInput);

for (int i = 1; i < conds.ConditionsList.Count; i++)

{

result = EvaluateConditionPair(result, EvaluateCondition(conds.ConditionsList[i], fuzzifiedInput), conds.Op);

}

}

if (conds.Not)

{

result = 1.0 - result;

}

return result;

}

else if (condition is FuzzyCondition)

{

FuzzyCondition cond = (FuzzyCondition)condition;

double result = fuzzifiedInput[(LinguisticVariable)cond.Var][(Term)cond.Term];

switch (cond.Hedge)

{

case HedgeType.Slightly:

result = Math.Pow(result, 1.0 / 3.0); //Cube root

break;

case HedgeType.Somewhat:

result = Math.Sqrt(result);

break;

case HedgeType.Very:

result = result \* result;

break;

case HedgeType.Extremely:

result = result \* result \* result;

break;

default:

break;

}

if (cond.Not)

{

result = 1.0 - result;

}

return result;

}

else

{

throw new Exception("Internal exception.");

}

}

private double EvaluateConditionPair(double cond1, double cond2, OperatorType op)

{

if (op == OperatorType.And)

{

if (\_andMethod == AndMethod.Min)

{

return Math.Min(cond1, cond2);

}

else if (\_andMethod == AndMethod.Production)

{

return cond1 \* cond2;

}

else

{

throw new Exception("Internal error.");

}

}

else if (op == OperatorType.Or)

{

if (\_orMethod == OrMethod.Max)

{

return Math.Max(cond1, cond2);

}

else if (\_orMethod == OrMethod.Probabilistic)

{

return cond1 + cond2 - cond1 \* cond2;

}

else

{

throw new Exception("Internal error.");

}

}

else

{

throw new Exception("Internal error.");

}

}

}

}

EvaluateConditionsResult.cs

using System;

using System.Collections.Generic;

using FuzzyLogic.FuzzySystem.Mamdani.Enums;

using FuzzyLogic.FuzzySystem.Mamdani.Rules;

namespace FuzzyLogic.FuzzySystem.Mamdani

{

internal class EvaluateConditionsResult

{

private Dictionary<MamdaniFuzzyRule, double> result;

private ImplicationMethod \_implMethod;

public EvaluateConditionsResult(Dictionary<MamdaniFuzzyRule, double> result)

{

this.result = result;

}

/// <summary>

/// Активизация подзаключений.

/// </summary>

/// <returns></returns>

public ImplicateResult Implicate(ImplicationMethod implicationMethod)

{

\_implMethod = implicationMethod;

return new ImplicateResult(Implicate(result));

}

private Dictionary<MamdaniFuzzyRule, Func<double, double>> Implicate(Dictionary<MamdaniFuzzyRule, double> conditions)

{

var conclusions = new Dictionary<MamdaniFuzzyRule, Func<double, double>>();

foreach (var rule in conditions.Keys)

{

MfCompositionType compType;

switch (\_implMethod)

{

case ImplicationMethod.Min:

compType = MfCompositionType.Min;

break;

case ImplicationMethod.Production:

compType = MfCompositionType.Prod;

break;

default:

throw new Exception("Internal error.");

}

var resultMf = new CompositeMembershipFunction(

compType,

new ConstantMembershipFunction(conditions[rule]).GetValue,

rule.Conclusion.Term.GetValue);

conclusions.Add(rule, resultMf.GetValue);

}

return conclusions;

}

}

public class ConstantMembershipFunction

{

private double \_constValue;

/// <summary>

/// Constructor

/// </summary>

/// <param name="constValue">Constant value</param>

public ConstantMembershipFunction(double constValue)

{

if (constValue < 0.0 || constValue > 1.0)

{

throw new ArgumentException();

}

\_constValue = constValue;

}

/// <summary>

/// Evaluate value of the membership function

/// </summary>

/// <param name="x">Argument (x axis value)</param>

/// <returns></returns>

public double GetValue(double x)

{

return \_constValue;

}

}

}

ImplicateResult.cs

using System;

using System.Collections.Generic;

using FuzzyLogic.FuzzySystem.Mamdani.Enums;

using FuzzyLogic.FuzzySystem.Mamdani.Rules;

using FuzzyLogic.LinguisticVariables;

namespace FuzzyLogic.FuzzySystem.Mamdani

{

internal class ImplicateResult

{

private Dictionary<MamdaniFuzzyRule, Func<double, double>> \_dictionary;

private OutputLinguisticVariable[] \_output;

private AggregationMethod \_aggregationMethod;

public ImplicateResult(Dictionary<MamdaniFuzzyRule, Func<double, double>> dictionary)

{

this.\_dictionary = dictionary;

}

/// <summary>

/// Аккумулирование заключений

/// </summary>

/// <returns></returns>

public AggregateResult Aggregate(AggregationMethod aggregationMethod, OutputLinguisticVariable[] output)

{

\_aggregationMethod = aggregationMethod;

\_output = output;

return new AggregateResult(Aggregate(\_dictionary));

}

public Dictionary<OutputLinguisticVariable, Func<double, double>> Aggregate(Dictionary<MamdaniFuzzyRule, Func<double, double>> conclusions)

{

var fuzzyResult = new Dictionary<OutputLinguisticVariable, Func<double, double>>();

foreach (var variable in \_output)

{

var mfList = new List<Func<double, double>>();

foreach (MamdaniFuzzyRule rule in conclusions.Keys)

{

if (rule.Conclusion.Var == variable)

{

mfList.Add(conclusions[rule]);

}

}

MfCompositionType composType;

switch (\_aggregationMethod)

{

case AggregationMethod.Max:

composType = MfCompositionType.Max;

break;

case AggregationMethod.Sum:

composType = MfCompositionType.Sum;

break;

default:

throw new Exception("Internal exception.");

}

fuzzyResult.Add(variable, new CompositeMembershipFunction(composType, mfList).GetValue);

}

return fuzzyResult;

}

}

}

AggregateResult.cs

using System;

using System.Collections.Generic;

using FuzzyLogic.LinguisticVariables;

namespace FuzzyLogic.FuzzySystem.Mamdani

{

internal class AggregateResult

{

private Dictionary<OutputLinguisticVariable, Func<double, double>> dictionary;

private DefuzzificationMethod \_defuzzMethod;

public AggregateResult(Dictionary<OutputLinguisticVariable, Func<double, double>> dictionary)

{

this.dictionary = dictionary;

}

/// <summary>

/// Дефаззификация выходных переменных.

/// </summary>

public void Defuzzify(DefuzzificationMethod defuzzMethod)

{

\_defuzzMethod = defuzzMethod;

Defuzzify(dictionary);

}

public Dictionary<OutputLinguisticVariable, double> Defuzzify(Dictionary<OutputLinguisticVariable, Func<double, double>> fuzzyResult)

{

var crispResult = new Dictionary<OutputLinguisticVariable, double>();

foreach (var variable in fuzzyResult.Keys)

{

var result = Defuzzify(fuzzyResult[variable], variable.Range.Min, variable.Range.Max);

crispResult.Add(variable, result);

variable.Result = result;

}

return crispResult;

}

private double Defuzzify(Func<double, double> mf, double min, double max)

{

if (\_defuzzMethod == DefuzzificationMethod.Centroid)

{

int k = 50;

double step = (max - min) / k;

double ptLeft = 0.0;

double ptCenter = 0.0;

double ptRight = 0.0;

double valLeft = 0.0;

double valCenter = 0.0;

double valRight = 0.0;

double val2Left = 0.0;

double val2Center = 0.0;

double val2Right = 0.0;

double numerator = 0.0;

double denominator = 0.0;

for (int i = 0; i < k; i++)

{

if (i == 0)

{

ptRight = min;

valRight = mf(ptRight);

val2Right = ptRight \* valRight;

}

ptLeft = ptRight;

ptCenter = min + step \* ((double)i + 0.5);

ptRight = min + step \* (i + 1);

valLeft = valRight;

valCenter = mf(ptCenter);

valRight = mf(ptRight);

val2Left = val2Right;

val2Center = ptCenter \* valCenter;

val2Right = ptRight \* valRight;

numerator += step \* (val2Left + 4 \* val2Center + val2Right) / 3.0;

denominator += step \* (valLeft + 4 \* valCenter + valRight) / 3.0;

}

if (denominator == 0)

{

return 0.5;

}

return numerator / denominator;

}

else if (\_defuzzMethod == DefuzzificationMethod.Bisector)

{

// **TODO:**

throw new NotSupportedException();

}

else if (\_defuzzMethod == DefuzzificationMethod.AverageMaximum)

{

// **TODO:**

throw new NotSupportedException();

}

else

{

throw new Exception("Internal exception.");

}

}

}

}