|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика, искусственный интелект и системы управления» (ИУ)

КАФЕДРА «Системы обработки информации и управления» (ИУ5)

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

**«Исследование нечётких алгоритмов»**

Студент группы ИУ5-34М **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Н.С. Грунин

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Ю.Е. Гапанюк

*2022 г.*

**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУ5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.И. Терехов

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме «Исследование нечётких алгоритмов»

Студент группы ИУ5-34М

Грунин Никита Сергеевич

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

исследовательская\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к \_\_\_ нед., 50% к \_\_\_ нед., 75% к \_\_ нед., 100% к \_\_\_ нед.

Техническое задание Произвести разработку модуля обработки запросов в метрагафовом хранилище

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на \_\_\_\_ листе формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**Руководитель НИР**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Е. Гапанюк

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Н.С. Грунин

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Оглавление**

[**Введение** 3](#_Toc121741306)

[**Нечеткие выводы** 4](#_Toc121741307)

[**Алгоритм Mamdani** 9](#_Toc121741308)

[**Алгоритм Tsukamoto** 16](#_Toc121741309)

[**Алгоритм Sugeno** 18](#_Toc121741310)

[**Алгоритм Larsen** 20](#_Toc121741311)

[**Упрощенный алгоритм нечеткого вывода** 22](#_Toc121741312)

[**Методы приведения к четкости** 24](#_Toc121741313)

[**Нисходящие нечеткие выводы** 26](#_Toc121741314)

[**Заключение** 34](#_Toc121741315)

[**Список использованных источников** 35](#_Toc121741316)

**Введение**

В наши дни многие понятия и явления невозможно описать с помощью математических методов, использующих классическую двузначную логику. Здесь на помощь приходит нечеткая логика, позволяющая решать подобные проблемы. Нечеткий логический вывод активно развивается, с каждым годом появляются новые подходы к решению задач. В связи с этим было принято решение реализовать некоторые алгоритмы нечеткого логического вывода. Их реализация дает возможность исследовать возможность проведения обратного вывода. А также оценить применимость данных алгоритмов в рамках возможных задач, при помощи оценки по точности, алгоритмической сложности и занимаемой памяти.

**Нечеткие выводы**

Понятие **нечеткого вывода** занимает важнейшее место в нечеткой логике Алгоритм Mamdani, Алгоритм Tsukamoto, Алгоритм Sugeno, Алгоритм Larsen, Упрощенный алгоритм нечеткого вывода, Методы приведения к четкости.

Используемый в различного рода экспертных и управляющих системах механизм нечетких выводов в своей основе имеет базу знаний, формируемую специалистами предметной области в виде совокупности нечетких предикатных правил вида:

П1: если *х*есть A1, тогда *у*есть B1,

П2: если *х*есть А2, тогда *у*есть В2,

·················································

П*n*: если *х* есть *Аn,*тогда *у*есть В*n*, где *х*— входная переменная (имя для известных значений данных), *у* — переменная вывода (имя для значения данных, которое будет вычислено); А и В — функции принадлежности, определен­ные соответственно на *x*и *у*.

*Пример подобного правила*

Если *х* — низко, то *у* — высоко.

Приведем более детальное пояснение. Знание эксперта А → В отражает нечеткое причинное отношение предпосылки и заключе­ния, поэтому его можно назвать нечетким отношением и обозна­чить через *R:*

*R =*А → В,

где «→» называют нечеткой импликацией.

Отношение *R*можно рассматривать как нечеткое подмножество прямого произведения *Х×У* полного множества предпосылок *X* и заключений *Y.* Таким образом, процесс получения (нечеткого) результата вывода В' с использованием данного наблюдения *А'* и знания А → В можно представить в виде формулы

В' = А' ᵒ*R*= А' (А → В),

где «о» — введенная выше операция свертки.

Как операцию композиции, так и операцию импликации в алгебре нечетких множеств можно реализовывать по-разному (при этом, естественно, будет разниться и итоговый получаемый результат), но в любом случае общий логический вывод осуществляется за следующие четыре этапа.

1. *Нечеткость* (введение нечеткости, фазификация, fuzzification). Функции принадлежности, определенные на входных пере­менных применяются к их фактическим значениям для определе­ния степени истинности каждой предпосылки каждого правила.

2. *Логический вывод.* Вычисленное значение истинности для предпосылок каждого правила применяется к заключениям каждого правила. Это приводит к одному нечеткому подмножеству, которое будет назначено каждой переменной вывода для каждого правила. В качестве правил логического вывода обычно используются только операции min(МИНИМУМ) или prod(УМНОЖЕНИЕ). В логическом выводе МИНИМУМА функция принадлежно­сти вывода «отсекается» по высоте, соответствующей вычислен­ной степени истинности предпосылки правила (нечеткая логика «И»). В логическом выводе УМНОЖЕНИЯ функция принадлежности вывода масштабируется при помощи вычисленной степени истинности предпосылки правила.

3. *Композиция.*Все нечеткие подмножества, назначенные к каждой переменной вывода (во всех правилах), объединяются вместе, чтобы формировать одно нечеткое подмножество для каждой переменной вывода. При подобном объединении обычно используются операции max(МАКСИМУМ) или sum(СУММА). При ком­позиции МАКСИМУМА комбинированный вывод нечеткого подмножества конструируется как поточечный максимум по всем нечетким подмножествам (нечеткая логика «ИЛИ»). При композиции СУММЫ комбинированный вывод нечеткого подмножества конструируется как поточечная сумма по всем нечетким подмножествам, назначенным переменной вывода правилами логического вывода.

4. В заключение (дополнительно) —*приведение к четкости*(дефазификация, defuzzification), которое используется, когда по­лезно преобразовать нечеткий набор выводов в четкое число. Име­ется большое количество методов приведения к четкости, некото­рые из которых рассмотрены ниже.

**Пример**.Пусть некоторая система описывается следующими нечет­кими правилами:

П1: если *х*есть А, тогда *ω* есть D,

П2: если *у*есть В, тогда *ω* есть Е,

П3: если *z*есть С, тогда *ω е*сть F, где *х, у* и *z*— имена входных переменных, *ω* — имя переменной вывода, а А, В, С, D, Е, F— заданные функции принадлежности (треугольной формы).

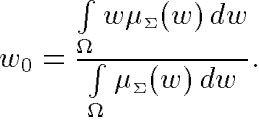
Предполагается, что входные переменные приняли некоторые кон­кретные (четкие) значения — *хо, yо*и *z*о.

В соответствии с приведенными этапами, на этапе 1 для данных зна­чений и исходя из функций принадлежности А, В, С, находятся степени истинности *α*(*хо*)*, α*(*уо*)и *α*(*zo*)для предпосылок каждого из трех при­веденных правил (см. рис. 1.9).

На этапе 2 происходит «отсекание» функций принадлежности за­ключений правил (т.е. D, Е, F) на уровнях *α*(*хо*)*, α*(*уо*) и *α*(*zo*)*.*

На этапе 3 рассматриваются усеченные на втором этапе функции при­надлежности и производится их объединение с использованием операции max, в результате чего получается комбинированное нечеткое подмноже­ство, описываемое функцией принадлежности μ∑(ω) и соответствующее логическому выводу для выходной переменной *ω.*

Наконец, на 4-м этапе — при необходимости — находится четкое значение выходной переменной, например, с применением центроидного метода: четкое значение выходной переменной определяется как центр тяжести для кривой μ∑(ω), т.е.



Рассмотрим следующие наиболее часто используемые модифи­кации алгоритма нечеткого вывода, полагая, для простоты, что базу знаний организуют два нечетких правила вида:

П1: если *х* есть A1и *у* есть B1, тогда *z*есть C1,

П2: если *х* есть А2 и *у* есть В2, тогда *z*есть С2, где *x* и *у* — имена входных переменных, *z*— имя переменной вы­вода, A1, А2, B1, В2, C1, С2 — некоторые заданные функции при­надлежности, при этом четкое значение *z*0необходимо определить на основе приведенной информации и четких значений *x*0и *у*0.

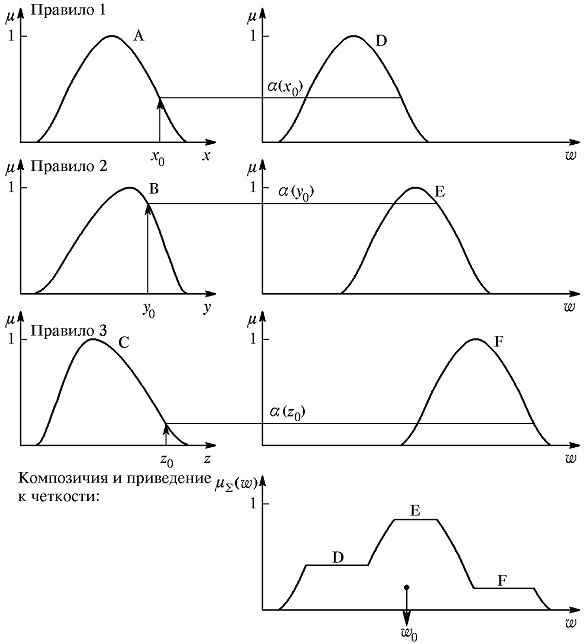


Рисунок 1 - Иллюстрация к процедуре логического вывода

**Алгоритм Mamdani**

Данный алгоритм соответствует рассмотренному примеру и рис. 1.9. В рассматриваемой ситуации он математически может быть описан следующим образом.

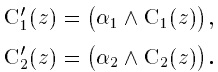
1. Нечеткость: находятся степени истинности для предпосылок каждого правила: А1(*x*0), А2(*x*0), B1(*y*0), В2(*y*0).

2. Нечеткий вывод: находятся уровни «отсечения» для пред­посылок каждого из правил (с использованием операции МИНИМУМ)

α1= A1(*x*0) ˄ B1(*y*0)

α2 = A2(*x*0) ˄ B2(*y*0)

где через «˄» обозначена операция логического минимума (min), затем находятся «усеченные» функции принадлежности



3. Композиция: с использование операции МАКСИМУМ (max, далее обозначаемой как «˅») производится объединение найден­ных усеченных функций, что приводит к получению **итогового**не­четкого подмножества для переменной выхода с функцией принад­лежности

Нечеткие выводы

4. Наконец, приведение к четкости (для нахождения *z*0*)*прово­дится, например, центроидным методом.

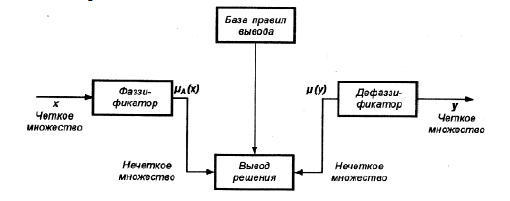


Рисунок 2 - Структура нечеткой системы с фаззификатором и дефаззификатором

Данный алгоритм описывает несколько последовательно выполняющихся этапов (рисунок 3). При этом каждый последующий этап получает на вход значения, полученные на предыдущем шаге.

https://habrastorage.org/r/w1560/storage/habraeffect/d3/96/d3963d1f04f1e048aca0db4ed759cdb2.png  
Рисунок 3 - Диаграмма деятельности процесса нечеткого вывода

Алгоритм примечателен тем, что он работает по принципу «черного ящика». На вход поступают количественные значения, на выходе они же. На промежуточных этапах используется аппарат нечеткой логики и теория нечетких множеств. В этом и состоит элегантность использования нечетких систем. Можно манипулировать привычными числовыми данными, но при этом использовать гибкие возможности, которые предоставляют системы нечеткого вывода.

Диаграмма (рисунок 4) показывает наиболее существенные связи и отношения между классами, задействованными в алгоритме.

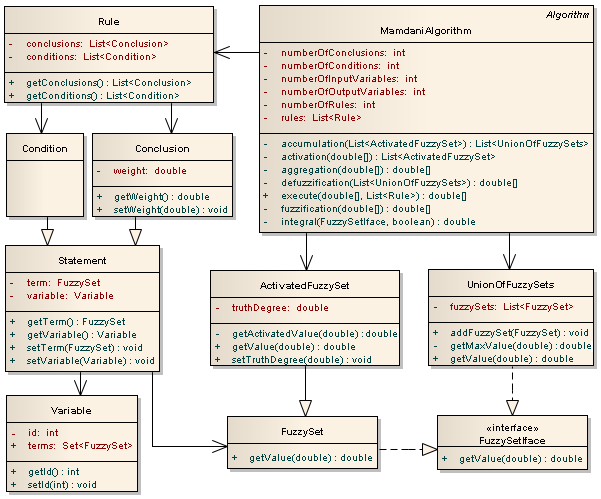


Рисунок 4 - Диаграмма классов реализации алгоритма Мамдани

Правила (Rule) состоят из условий (Condition) и заключений (Conclusion), которые в свою очередь являются нечеткими высказываниями (Statement). Нечеткое высказывание включает в себя лингвистическую переменную (Variable) и терм, который представлен нечетким множеством (FuzzySet). На нечетком множестве определена функция принадлежности, значение которой можно получить с помощью метода getValue(). Это метод определенный в интерфейсе FuzzySetIface. При выполнении алгоритма необходимо будет воспользоваться «активизированным» нечетким множеством (ActivatedFuzzySet), которое некоторым образом переопределяет функцию принадлежности нечеткого множества (FuzzySet). Также в алгоритме используется объединение нечетких множеств (UnionOfFuzzySets). Объединение также является нечетким множеством, и поэтому имеет функцию принадлежности (определенную в FuzzySetIface).  
  
Алгоритм Мамдани (MamdaniAlgorithm), включает в себя все этапы (рисунок 3) и использует базу правил (List<Rule>) в качестве входных данных. Также алгоритм предполагает использование «активизированных» нечетких множеств (ActivatedFuzzySet) и их объединений (UnionOfFuzzySets).  
  
Итак, этапы нечеткого вывода выполняются последовательно. И все значения, полученные на предыдущем этапе, могут использоваться на следующем.

**1. Формирование базы правил**

*База правил* — это множество правил, где каждому подзаключению сопоставлен определенный весовой коэффициент.  
База правил может иметь следующий вид (для примера используются правила различных конструкций):  
RULE\_1: IF «Condition\_1» THEN «Conclusion\_1» (*F*1) AND «Conclusion\_2» (*F*2);  
RULE\_2: IF «Condition\_2» AND «Condition\_3» THEN «Conclusion\_3» (*F*3);  
…  
RULE\_n: IF «Condition\_k» THEN «Conclusion\_(q-1)» (*F*q-1) AND «Conclusion\_q» (*F*q);  
  
Где *F*i — весовые коэффициенты, означающие степень уверенности в истинности получаемого подзаключения (*i* = 1..*q*). По умолчанию весовой коэффициент принимается равным 1. Лингвистические переменные, присутствующие в условиях называются *входными*, а в заключениях *выходными*.  
  
Обозначения:  
*n* — число правил нечетких продукций (numberOfRules).  
*m* — кол-во входных переменных (numberOfInputVariables).  
*s* — кол-во выходных переменных (numberOfOutputVariables).  
*k* — общее число подусловий в базе правил (numberOfConditions).  
*q* — общее число подзаключений в базе правил (numberOfConclusions).  
  
**2. Фаззификация входных переменных**

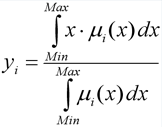
Этот этап часто называют приведением к нечеткости. На вход поступают сформированная база правил и массив входных данных А = {a1, ..., am}. В этом массиве содержатся значения всех входных переменных. Целью этого этапа является получение значений истинности для всех подусловий из базы правил. Это происходит так: для каждого из подусловий находится значение *b*i = μ(*a*i). Таким образом получается множество значений *b*i (*i* = 1..*k*).  
  
**3. Агрегирование подусловий**

Как уже упоминалось выше, условие правила может быть составным, т.е. включать подусловия, связанные между собой при помощи логической операции «AND». Целью этого этапа является определение степени истинности условий для каждого правила системы нечеткого вывода. Упрощенно говоря, для каждого условия находим минимальное значение истинности всех его подусловий. Формально это выглядит так:  
  
*c*j =min{*b*i}.  
  
Где:  
*j* =1..*n*;  
*i* — число из множества номеров подусловий в которых участвует *j*-ая входная переменная.  
  
**4. Активизация подзаключений**

На этом этапе происходит переход от условий к подзаключениям. Для каждого подзаключения находится степень истинности *d*i = *c*i\**F*i, где *i* = 1..*q*. Затем, опять же каждому *i*-му подзаключению, сопоставляется множество *D*i с новой функцией принадлежности. Её значение определяется как минимум из *d*i и значения функции принадлежности терма из подзаключения. Этот метод называется min-активизацией, который формально записывается следующим образом:  
μ'i(*x*)=min{*d*i,μi(*x*)}.  
Где:  
μ'i(*x*)—«активизированная» функция принадлежности;  
μi(*x*)—функция принадлежности терма;  
*d*i —степень истинности *i*-го подзаключения.  
Итак, цель этого этапа — это получение совокупности «активизированных» нечетких множеств *D*i для каждого из подзаключений в базе правил (*i* = 1..*q*).  
  
**5. Акумуляция заключений**

Целью этого этапа является получение нечеткого множества (или их объединения) для каждой из выходных переменных. Выполняется он следующим образом: *i*-ой выходной переменной сопоставляется объединение множеств *E*i = ∪ *D*j. Где *j* — номера подзаключений в которых участвует *i*-aя выходная переменная (*i* = 1..*s*). Объединением двух нечетких множеств является третье нечеткое множество со следующей функцией принадлежности:  
  
μ'i(*x*) = max {μ1(*x*), μ2(*x*)}, где μ1(*x*), μ2(*x*) — функции принадлежности объединяемых множеств.

**6. Дефаззификация выходных переменных**

Цель дефаззификациии получить количественное значение (crisp value) для каждой из выходных лингвистических переменных. Формально, это происходит следующим образом. Рассматривается *i*-ая выходная переменная и относящееся к ней множество *E*i (*i* = 1..*s*). Затем при помощи метода дефаззификации находится итоговое количественное значение выходной переменной. В данной реализации алгоритма используется метод центра тяжести, в котором значение *i*-ой выходной переменной рассчитывается по формуле:  
  
Где:  
μi(*x*) — функция принадлежности соответствующего нечеткого множества *E*i;  
*Min* и *Max* — границы универсума нечетких переменных;  
*y*i — результат дефаззификации.

**Алгоритм Tsukamoto**

 Исходные посылки — как у пре­дыдущего алгоритма, но в данном случае предполагается, что функ­ции C1(*z*), С2(*z*) являются монотонными.

1. Первый этап — такой же, как в алгоритме Mamdani.

2. На втором этапе сначала находятся (как в алгоритме Mam­dani) уровни «отсечения» α1 и α2, а затем — посредством решения уравнений

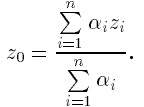
α1 = C1(*z*1), *α2 =* C2(*z*2)

— четкие значения (*z1*и *z2*)для каждого из исходных правил.

3. Определяется четкое значение переменной вывода (как взве­шенное среднее *z1*и *z2*)*:*

Нечеткие выводы

в общем случае (дискретный вариант центроидного метода)



Пример. Пусть имеем A1(*x*0) = 0,7, A2(*x*0) = 0,6, B1(*y*0) = 0,3, В2(*y*0) = 0,8, соответствующие уровни отсечения

*a1 =*min (A1(*x*0), B1(*y*0)) = min(0,7; 0,3) = 0,3,

*a2 =*min (А2(*x*0), В2(*y*0)) = min (0,6; 0,8) = 0,6

и значения *z1*= 8 и *z*2= 4, найденные в результате решения уравнений

C1(*z*1) = 0,3 , C2(*z*2) = 0,6.

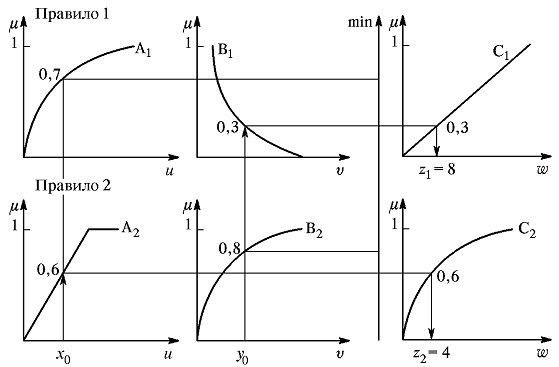


Рисунок 6 - Иллюстрации к алгоритму Tsukamoto

При этом четкое значение переменной вывода (см. рис. 1.10)

z0 = (8·0,3 + 4·0,6) / (0,3 + 0,6) = 6.

**Алгоритм Sugeno**

Sugeno и Takagi использовали набор правил в следующей форме (как и раньше, приводим пример двух правил):

П1: если *х* есть A1и *у* есть B1, тогда *z1*= *а*1*х* + *b*1*у,*

П2: если *х* есть A2 и *у* есть В2, тогда *z*2= *a2x*+ *b*2*y*.

Представление алгоритма

1. Первый этап — как в алгоритме Mamdani.

2. На втором этапе находятся *α*1= A1(*x*0) ˄ B1(*y*0), α2 = А2(*x*0) ˄ В2(*у*0) и индивидуальные выходы правил:

Нечеткие выводы

З.      На третьем этапе определяется четкое значение переменной вывода:

Нечеткие выводы

Иллюстрирует алгоритм рисунок 7

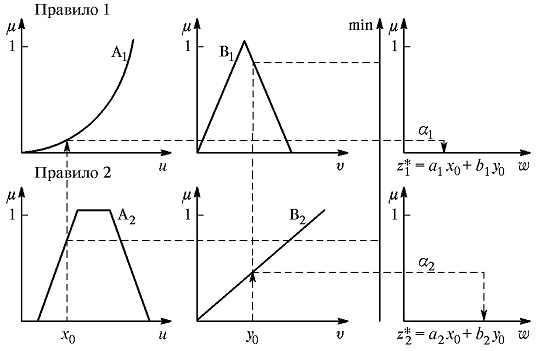


Рисунок 7 - Иллюстрация к алгоритму Sugeno

**Алгоритм Larsen**

В алгоритме Larsen нечеткая импли­кация моделируется с использованием оператора умножения.

Описание алгоритма

1. Первый этап — как в алгоритме Mamdani.

2. На втором этапе, как в алгоритме Mamdani вначале нахо­дятся значения

α1= A1(*x*0) ˄ B1(*y*0),

α2 = А2(*x*0) ˄ В2(*y*0),

а затем — частные нечеткие подмножества

α1C1(*z*), *a2C2*(*z*)*.*

3. Находится итоговое нечеткое подмножество с функцией при­надлежности

*μs*(*z*)*= С*(*z*)= (a1C1(*z*)) ˅ (*a2C2*(*z*))

(в общем случае *n* правил).

4. При необходимости производится приведение к четкости (как в ранее рассмотренных алгоритмах).

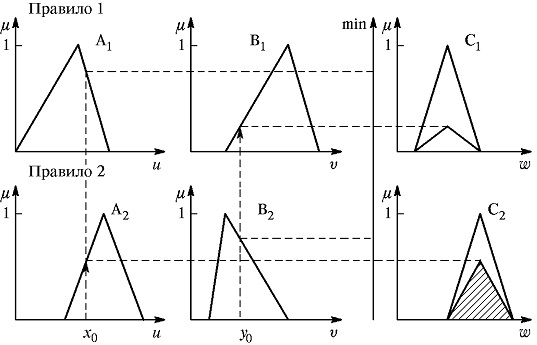


Рисунок 8 - Иллюстрация алгоритма Larsen

**Упрощенный алгоритм нечеткого вывода**

Исходные пра­вила в данном случае задаются в виде:

П1: если *х* есть A1и *у* есть B1, тогда *z*1= *c*1,

П2: если *х* есть А2 и *у* есть В2, тогда *z*2= *с*2*,*где *c*1и *с2* — некоторые обычные (четкие) числа.

Описание алгоритма

1. Первый этап — как в алгоритме Mamdani.

2. На втором этапе находятся числа α1= A1(*x*0) ˄ B1(*y*0), α2= A2(*x*0) ˄ B2(*y*0).

3. На третьем этапе находится четкое значение выходной пе­ременной по формуле

Нечеткие выводы

или — в общем случае наличия *n* правил — по формуле

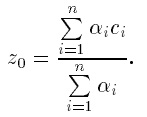


Иллюстрация алгоритма приведена на рисунке 9.

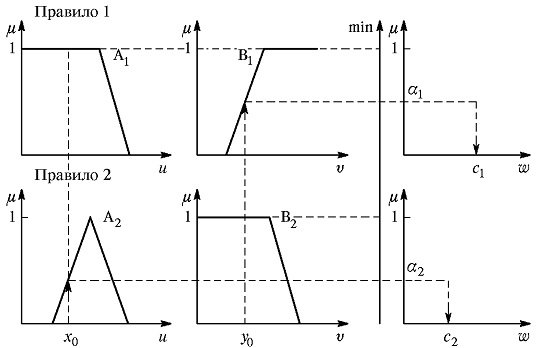
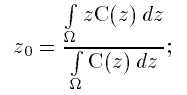


Рисунок 9 - Иллюстрация упрощенного алгоритма нечеткого вывода

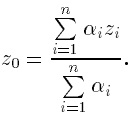
**Методы приведения к четкости**

1. Выше уже был рассмотрен один из данных методов — троидный. Приведем соответствующие формулы еще раз.

Для непрерывного варианта:



для дискретного варианта:



2. Первый максимум (First-of-Maxima). Четкая величина пере­менной вывода находится как наименьшее значение, при котором достигается максимум итогового нечеткого множества, т.е.

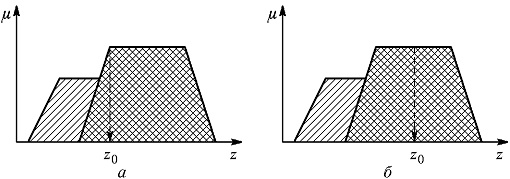
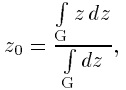


Рисунок 10 - Иллюстрация к методам приведения к четкости: α — первый максимум; б — средний максимум

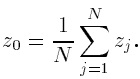
Нечеткие выводы

3.          Средний максимум (Middle-of-Maxima). Четкое значение находится по формуле



где G — подмножество элементов, максимизирующих С.

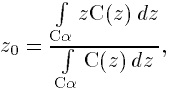
Дискретный вариант (если С — дискретно):



4. Критерий максимума (Max-Criterion). Четкое значение вы­бирается произвольно среди множества элементов, доставляющих максимум С, т. е.

https://neuronus.com/images/theory/nechetkaya_logika/5_ne4etkie_vivod/19.jpg

5. Высотная дефазификация. Элементы области определения Ω для которых значения функции принад­лежности меньше, чем некоторый уровень *α* в расчет не принима­ются, и четкое значение рассчитывается по формуле



где Сα — нечеткое множество *α*-уровня (см. выше).

**Нисходящие нечеткие выводы**

Рассмотренные до сих пор нечеткие выводы представляют собой восходящие выводы от предпосылок к заключению. В последние годы в диагностических нечетких системах начинают применяться нисходящие выводы. Рассмотрим механизм подобного вывода на примере.

Возьмем упрощенную модель диагностики неисправности ав­томобиля с именами переменных:

*х*1— неисправность аккумулятора;

*x*2— отработка машинного масла;

*y*1— затруднения при запуске;

*y*2— ухудшение цвета выхлопных газов;

*y*3— недостаток мощности.

Между *xi*и *yj*существуют нечеткие причинные отношения *rij*= *xi*→ *yj,*которые можно представить в виде некоторой ма­трицы **R**с элементами *rij*ϵ [0, 1]. Конкретные входы (предпо­сылки) и выходы (заключения) можно рассматривать как нечет­кие множества А и В на пространствах *X*и *Y.* Отношения этих множеств можно обозначить как

*В***=***А* ᵒ **R,**

где, как и раньше, знак «о» обозначает правило композиции не­четких выводов.

В данном случае направление выводов является обратным к направлению выводов для правил, т.е. в случае диагностики име­ется (задана) матрица **R**(знания эксперта), наблюдаются выходы *В* (или симптомы) и определяются входы *А* (или факторы).

Пусть знания эксперта-автомеханика имеют вид

https://neuronus.com/images/theory/nechetkaya_logika/5_ne4etkie_vivod/21.jpg

а в результате осмотра автомобиля его состояние можно оценить как

*В*= 0,9/*y*1+ 0,1/*у*2 + 0,2/*у*3.

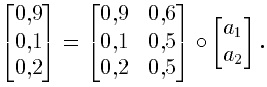
Требуется определить причину такого состояния:

*А = a*1/*x*1+ *a*2/*x*2.

Отношение введенных нечетких множеств можно представить в виде

https://neuronus.com/images/theory/nechetkaya_logika/5_ne4etkie_vivod/22.jpg

либо, транспонируя, в виде нечетких векторов-столбцов:



При использовании (max-mix)-композиции последнее соотно­шение преобразуется к виду

0,9 = (0,9 ˄ α1) ˅ (0,6 ˄ α2),

0,1 = (0,1 ˄ α1) ˅ (0,5 ˄ α2),

0,2 = (0,2 ˄ α1) ˅ (0,5 ˄ α2).

При решении данной системы заметим прежде всего, что в первом уравнении второй член правой части не влияет на правую часть, поэтому

0,9 = 0,9 ˄ α1, α1≥ 0,9.

Из второго уравнения получим:

0,1 ≥ 0,5 ˄ α2, α2 ≤ 0,1.

Полученное решение удовлетворяет третьему уравнению, та­ким образом имеем:

0,9 ≤ α1≤ 1,0,     0 ≤ α2 ≤ 0,1,

т.е. лучше заменить аккумулятор (α1— параметр неисправности аккумулятора, α2— параметр отработки машинного масла).

На практике в задачах, подобных рассмотренной, количество переменных может быть существенным, могут одновременно ис­пользоваться различные композиции нечетких выводов, сама схема выводов может быть многокаскадной. Общих методов решения по­добных задач в настоящее время, по-видимому, не существует.

*Таблица 1. Расчет сложности нечётких алгоритмов.*

|  |  |
| --- | --- |
| **Итог** | **Функция** |
| O(1) + O(N) \* O(N) \* (O(1) + O(1)) =  = O(1) + O(N2) =  =  ***O(N2)*** | private double[] fuzzification (double[] inputData) {  int i = 0;  double[] b = new double[numberOfConditions];  for (Rule rule: rules) {  for (Condition condition: rule.getConditions()) {  int j = condition.getVariable().getId()  FuzzySet term = condition.getTerm();  b[i] = term.getValue (inputData[j]);  i++;  }  }  return b;  } |
| O(1) + O(1) +O(N) \* O(N) \* (O(1) + O(1)) =  = O(1) + O(N2) =  ***= O(N2)*** | private double[] aggregation (double[] b) {  int i = 0;  int j = 0;  double[] c = new double[numberOfInputVariables];  for (Rule rule: rules) {  double truthOfConditions = 1.0;  for (Condition condition: rule.getConditions()) {  truthOfConditions = Math.min (truthOfConditions, b[i]);  i++;  }  c[j] = truthOfConditions;  j++;  }return c;  } |
| O(1) + O(1) +O(N) \* O(N) \* (O(1) + O(1)) =  = O(1) + O(N2) =  ***= O(N2)*** | private List<ActivatedFuzzySet> activation (double[] c) {  int i = 0;  List<ActivatedFuzzySet> activatedFuzzySets = new ArrayList<ActivatedFuzzySet>();  double[] d = new double[numberOfConclusions];  for (Rule rule: rules) {  for (Conclusion conclusion: rule.getConclusions()) {  d[i] = c[i]\*conclusion.getWeight();  ActivatedFuzzySet activatedFuzzySet = (ActivatedFuzzySet) conclusion.getTerm();  activatedFuzzySet.setTruthDegree (d[i]);  activatedFuzzySets.add(activatedFuzzySet);  i++;  }  }  return activatedFuzzySets;  }  private double getActivatedValue (double x) {  return Math.min (super.getValue(x), truthDegree);  } |
| O(N) + O(N2) + O(N2) + O(N) =  **= *O(N2)*** | private List<UnionOfFuzzySets> accumulation (List<ActivatedFuzzySet> activatedFuzzySets) {  List<UnionOfFuzzySets> unionsOfFuzzySets =  new ArrayList<UnionOfFuzzySets>(numberOfOutputVariables);  for (Rule rule: rules) {  for (Conclusion conclusion: rule.getConclusions()) {  int id = conclusion.getVariable().getId();  unionsOfFuzzySets.get(id).addFuzzySet (activatedFuzzySets.get(id));  }  }  return unionsOfFuzzySets;  }  private double getMaxValue (double x) {  double result = 0.0;  for (FuzzySet fuzzySet: fuzzySets) {  result = Math.max (result, fuzzySet.getValue(x));  }  return result; |

*Алгоритм Мамдани реализация*

|  |
| --- |
| fcltClusters =  **FOR\_EACH** f **IN** closedFacilitiesList:  fcltClusters.add (new Cluster({f}))  **WHILE** есть существенный прогресс **AND** время не истекло:  **FOR\_EACH** clstr **IN** fcltClusters:  activEnemies = [empty] ==!! fuzzy set ==  **FOR\_EACH**enemy **IN** openFacilitiesList:  **IF** enemy хорошо заменяется на clstr **OR**  цена enemy велика **THEN**:  activEnemies.add(enemy)  goodChange IS false  **FOR\_EACH** c **IN** citiesList:  **FOR\_EACH** f **IN** clstr:  **IF** fwtn[c] ? activEnemies **AND**  f дешевле для c чем fwtn[c] **THEN**:  goodChange IS true  **IF** goodChange **OR**  стоимость activEnemies больше стоимости clstr  **THEN**:  open(clstr)  fcltClusters.remove(clstr)  close(activEnemies)  **FOR\_EACH** aEnm **IN** activEnemies:  fcltClusters.add (new Cluster({aEnm}))  **FOR\_EACH** clstr1 **IN** fcltClusters:  **FOR\_EACH** clstr2 **IN** fcltClusters:  **IF** clstr1 похож на clstr2 **AND**  (стоимость clstr1 не слишком велика **OR**  стоимость clstr2 не слишком велика) **THEN**:  join (clstr1, clstr2)  **IF <fuzzy condition> THEN var\_out IS val\_out**  **<fuzzy condition>**-  **<fuzzy condition> =**  **<fuzzy check>**  **<fuzzy condition> <operator> <antecedent>**  **<antecedent> = <fuzzy variable> IS <linguistic variable>**  **<operator> = OR AND**  **<fuzzy variable<linguistic variable>**  **public interface INegotiatingAgent {**  **public void onNegotiationStart (INegotiationContext context);**  **/\*\***  **\* returns true if wants to continue negotiation**  **\*/**  **public boolean negotiate (INegotiationContext context);**  **}**  **public** **interface** INegotiationContext {  **public** <ServT **extends** IService> ServT getService (  Class<ServT> clazz  );  **public** <ServT **extends** IService> **void** registerService (  Class<ServT> clazz,  ServT service  );  }  **private** **void** analiseUtilities (  INegotiationContext context,  List<INegotiatingAgent> agents  ) {  IUtilityFuncNegotiationAgent minUtilAgent = **null**;  **double** minUtility = Double.*MAX\_VALUE*;  List<IUtilityFuncNegotiationAgent> ufagents =  extractUFAgents(agents);  **for** (IUtilityFuncNegotiationAgent agent: ufagents) {  **double** utility = agent.computeUtility(context);  **if** (utility < minUtility) {  minUtility = utility;  minUtilAgent = agent;  }  }  **if** (minUtilAgent == **null**) {  **return**;  }  minUtilAgent.negotiatePrivileged(context);  }  **private** **void** extractUFAgents (  List<INegotiatingAgent> agents)  {End} |

В результате подсчетов получили O(N2) - квадратичную сложность алгоритма.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Квадратичная сложность алгоритма | Память | Точность |
| Алгоритм Mamdani | O(N2) | 9 блоков | 99% |
| Алгоритм Tsukamoto | O(N2) | 5 блоков | 96% |
| Алгоритм Sugeno | O(N2) | 8 блоков | 98% |
| Алгоритм Larsen | O(N2) | 9 блоков | 99% |

Несмотря на равную оценку квадратичной сложности алгоритмов, можно выбрать алгоритм Mamdani, если требуется высокая точность, при относительно высоких затратах по памяти. Однако при наличии ограничения на память необходимо использовать алгоритм Tsukamoto или же Sugeno, в зависимости от ограничений.

**Заключение**

Гибридизация методов интеллектуальной обработки информации - девиз, под которым прошли 90-е годы у западных и американских исследователей. В результате объединения нескольких технологий искусственного интеллекта появился специальный термин - «мягкие вычисления» (soft computing), который ввел Л. Заде в 1994 году. В настоящее время мягкие вычисления объединяют такие области как: нечеткая логика, искусственные нейронные сети, вероятностные рассуждения и эволюционные алгоритмы. Они дополняют друг друга и используются в различных комбинациях для создания гибридных интеллектуальных систем.

Влияние нечеткой логики оказалось, пожалуй, самым обширным. Подобно тому, как нечеткие множества расширили рамки классической математическую теорию множеств, нечеткая логика «вторглась» практически в большинство методов Data Mining, наделив их новой функциональностью. Ниже приводятся наиболее интересные примеры таких объединений.

Триумфальное шествие нечеткой логики по миру началось после доказательства в конце 80-х Бартоломеем Коско знаменитой теоремы FAT (Fuzzy Approximation Theorem). В бизнесе и финансах нечеткая логика получила признание после того как в 1988 году экспертная система на основе нечетких правил для прогнозирования финансовых индикаторов единственная предсказала биржевой крах. И количество успешных фаззи-применений в настоящее время исчисляется тысячами.

# **Список использованных источников**

1. Леоненков А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH / А. Леоненков. - СПб: БХВ-Петербург, 2003. - 736 с.
2. Штовба С.Д. Проектирование нечетких систем средствами MATLAB / С. Штовба. - М: Горячая линия-Телеком, 2007. - 288 с.
3. В. Дьяконов, В. Круглов. Математические пакеты расширения MATLAB. Специальный справочник. - Санкт-Петербург: Питер, 2001 - 480 с.
4. Р.А. Алиев. Управление производством при нечёткой исходной информации, М.: Энергоатомиздат, 1991.
5. А. Гультяев. Визуальное моделирование в среде MATLAB: учебный курс. - Санкт-Перербург: Питер, 2000.
6. Н.Н. Мартынов, А.П. Иванов. MATLAB 5.x. Вычисление, визуализация, программирование. - М.: КУДИЦ-ОБРАЗ, 2000.
7. Орлов А.И. Теория принятия решений. Учебное пособие / А.И. Орлов. - М.: Издательство «Март», 2004. - 656 с.
8. Нейронные сети, генетические алгоритмы и нечеткие системы. Д. Рутковская, M. Пилиньский, Л. Рутковский. 1999.
9. Понятие лингвистической переменной и его применение к принятию приближенных решений. - Заде Л.А.М.: Мир, 1976.
10. Борисов, В. В. Нечеткие модели и сети / В. В. Борисов, В. В. Круглов, А. С. Федулов. – М. : Горячая линия. − Телеком, 2007. – 284 с. 2. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И. Д. Рудинского / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М. : Горячая линия − Телеком, 2006. – 452 с.