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.

З А Д А Н И Е
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

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 *x*есть A_1, тогда *y*есть B_1,

П2: если *x*есть A_2, тогда *y*есть B_2,

……

$П_n$: если *x* есть A_n,тогда *y*есть B_n, где *x*— входная переменная (имя для известных значений данных), *y* — переменная вывода (имя для значения данных, которое будет вычислено); A и B — функции принадлежности, определенные соответственно на *x*и *y*.

Пример подобного правила

Если *x* — низко, то *y* — высоко.

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

R =A → B,

где «→» называют нечеткой импликацией.

Отношение *R*можно рассматривать как нечеткое подмножество прямого произведения $X \times Y$ полного множества предпосылок X и заключений Y. Таким образом, процесс получения (нечеткого) результата вывода B' с использованием данного наблюдения A' и знания A → B можно представить в виде формулы

$$B' = A' \circ R = A' (A \rightarrow B),$$

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 *x*есть A, тогда ω есть D,

П2: если *y*есть B, тогда ω есть E,

П3: если *z*есть C, тогда ω *е*сть F, где *x, y* и *z*— имена входных переменных, ω — имя переменной вывода, а A, B, C, D, E, F— заданные функции принадлежности (треугольной формы).

Предполагается, что входные переменные приняли некоторые конкретные (четкие) значения — x_o, y_o и z_o.

В соответствии с приведенными этапами, на этапе 1 для данных значений и исходя из функций принадлежности A, B, C, находятся степени истинности $\alpha(x_o)$, $\alpha(y_o)$и $\alpha(z_o)$для предпосылок каждого из трех приведенных правил (см. рис. 1.9).

На этапе 2 происходит «отсекание» функций принадлежности заключений правил (т.е. D, E, F) на уровнях $\alpha(x_o)$, $\alpha(y_o)$ и $\alpha(z_o)$.

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 $\mu_\Sigma(\omega)$ 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 $\mu_\Sigma(\omega)$, т.е.

$$w_0 = \frac{\int\limits_{\Omega} w\mu_\Sigma(w)\,dw}{\int\limits_{\Omega} \mu_\Sigma(w)\,dw}.$$

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

П1: если x есть A_1 и y есть B_1, тогда zесть C_1,

П2: если x есть A_2 и y есть B_2, тогда zесть C_2, где x и y — имена входных переменных, z — имя переменной вывода, A_1, A_2, B_1, B_2, C_1, C_2 — некоторые заданные функции принадлежности, при этом четкое значение z_0 необходимо определить на основе приведенной информации и четких значений x_0 и y_0.

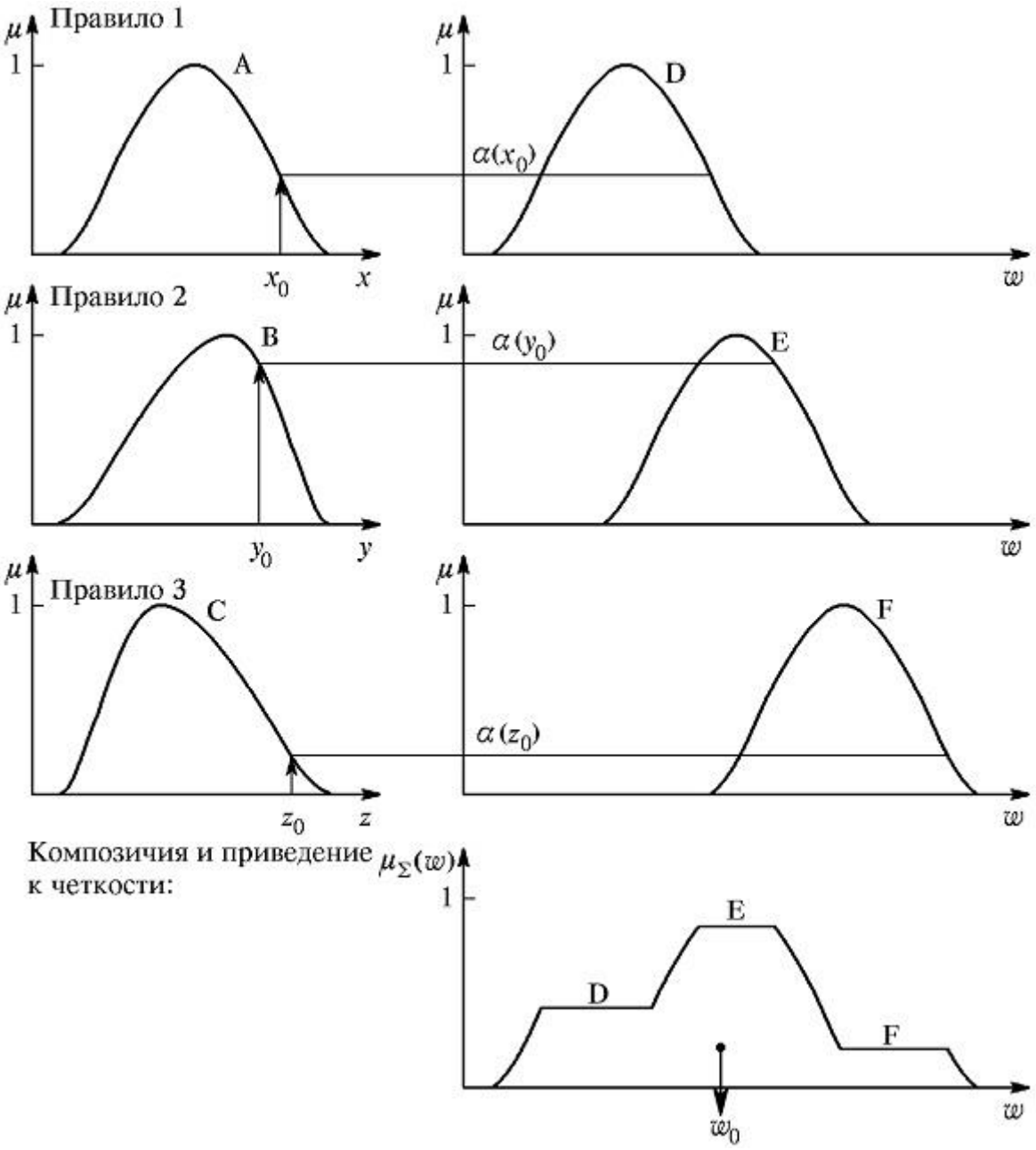

Рисунок 1 - Иллюстрация к процедуре логического вывода

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: $A_1(x_0)$, $A_2(x_0)$, $B_1(y_0)$, $B_2(y_0)$.

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

$\alpha_1 = A_1(x_0) \wedge B_1(y_0)$

$\alpha_2 = A_2(x_0) \wedge B_2(y_0)$

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

$$C_1'(z) = (\alpha_1 \wedge C_1(z)),$$
$$C_2'(z) = (\alpha_2 \wedge C_2(z)).$$

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

$$\mu_\Sigma(z) = C(z) = C_1'(z) \vee C_2'(z) = (\alpha_1 \wedge C_1(z)) \vee (\alpha_2 \wedge C_2(z)).$$

4. Наконец, приведение к четкости (для нахождения z_0) проводится, например, центроидным методом.

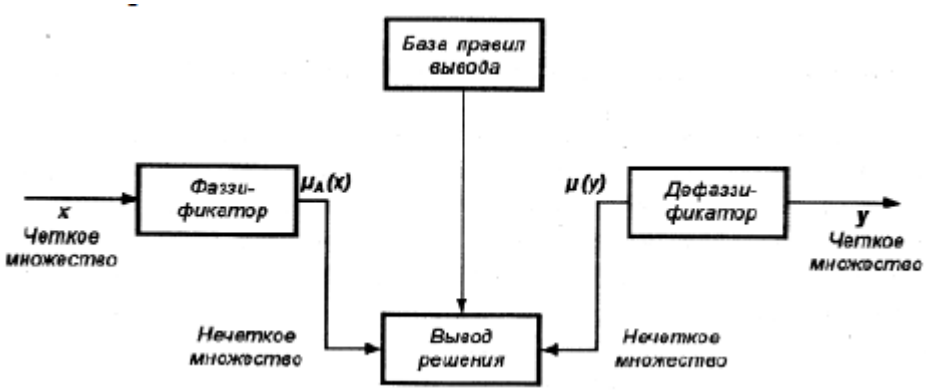

Рисунок 2 - Структура нечеткой системы с фаззификатором и дефаззификатором

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

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.

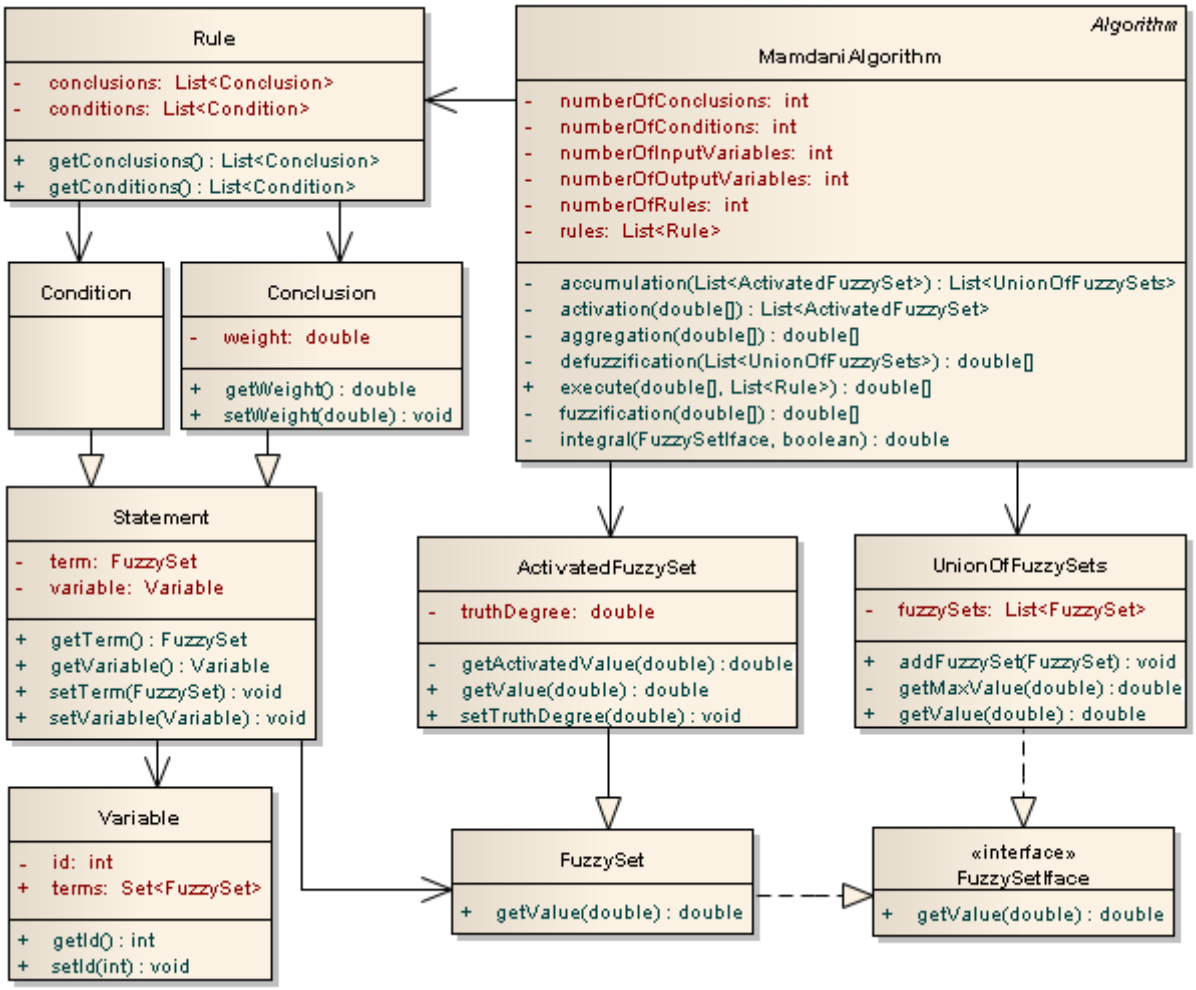

Рисунок 4 - Диаграмма классов реализации алгоритма Мамдани

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 $A = \{a_1, ..., a_m\}$. 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 $b_i = \mu(a_i)$. 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:

$\mu'_i(x)=\min\{d_i,\mu_i(x)\}$.

Где:

$\mu'_i(x)$—«активизированная» функция принадлежности;

$\mu_i(x)$—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 $E_i = \cup D_j$. Где j — номера подзаключений в которых участвует i-ая выходная переменная ($i = 1..s$). Объединением двух нечетких множеств является третье нечеткое множество со следующей функцией принадлежности:

$\mu'_i(x) = \max\{\mu_1(x), \mu_2(x)\}$, где $\mu_1(x)$, $\mu_2(x)$ — 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:

$$y_i = \frac{\int_{Min}^{Max} x \cdot \mu_i(x)\,dx}{\int_{Min}^{Max} \mu_i(x)\,dx}$$

Где:

$\mu_i(x)$ — 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 $C_1(z)$, $C_2(z)$ являются монотонными.

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

2. На втором этапе сначала находятся (как в алгоритме Mamdani) уровни «отсечения» α_1 и α_2, а затем — посредством решения уравнений

$\alpha_1 = C_1(z_1)$, $\alpha_2 = C_2(z_2)$

— четкие значения (z_1 и z_2)для каждого из исходных правил.

3. Определяется четкое значение переменной вывода (как взвешенное среднее z_1 и z_2):

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2};$$

в общем случае (дискретный вариант центроидного метода)

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_i z_i}{\sum_{i=1}^{n} \alpha_i}.$$

Пример. Пусть имеем $A_1(x_0) = 0{,}7$, $A_2(x_0) = 0{,}6$, $B_1(y_0) = 0{,}3$, $B_2(y_0) = 0{,}8$, соответствующие уровни отсечения

$a_1 = \min(A_1(x_0), B_1(y_0)) = \min(0{,}7; 0{,}3) = 0{,}3$,

$a_2 = \min(A_2(x_0), B_2(y_0)) = \min(0{,}6; 0{,}8) = 0{,}6$

и значения $z_1 = 8$ и $z_2 = 4$, найденные в результате решения уравнений

$C_1(z_1) = 0{,}3$, $C_2(z_2) = 0{,}6$.

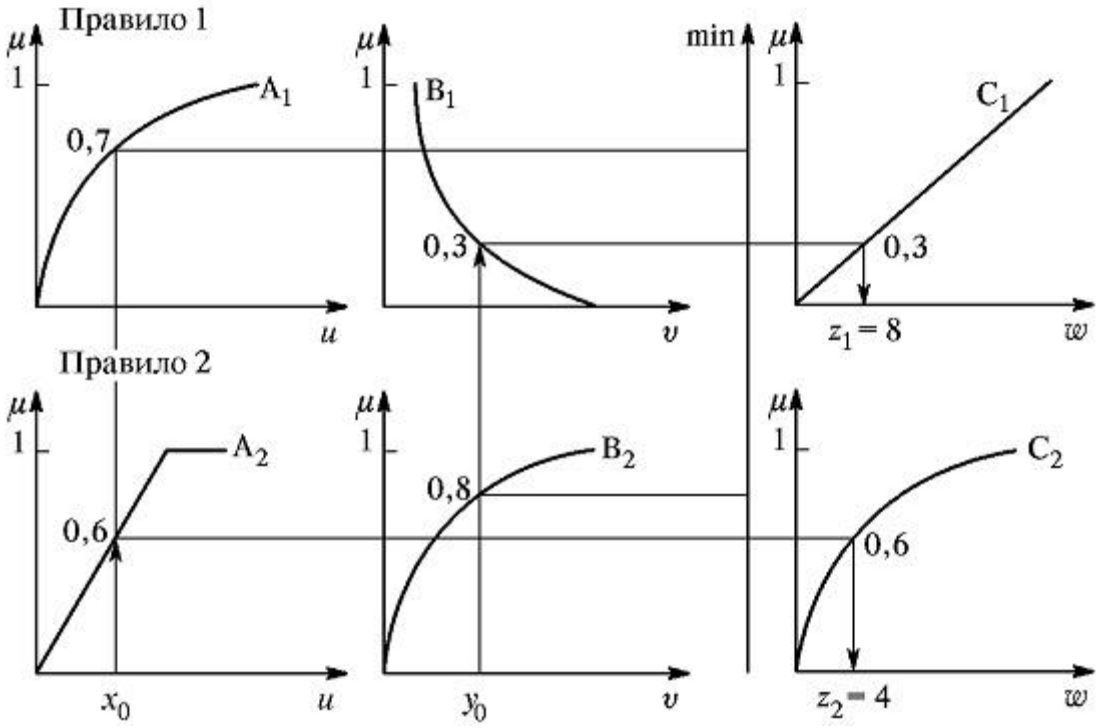

Рисунок 6 - Иллюстрации к алгоритму Tsukamoto

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

$z_0 = (8 \cdot 0{,}3 + 4 \cdot 0{,}6) / (0{,}3 + 0{,}6) = 6.$

Алгоритм Sugeno

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

$П_1$: если x есть A_1 и y есть B_1, тогда $z_1 = a_1x + b_1y$,

$П_2$: если x есть A_2 и y есть B_2, тогда $z_2 = a_2x + b_2y$.

Представление алгоритма

1. Первый этап — как в алгоритме Mamdani.

2. На втором этапе находятся $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$ и индивидуальные выходы правил:

$$z_1^* = a_1 x_0 + b_1 y_0,$$
$$z_2^* = a_2 x_0 + b_2 y_0.$$

3. На третьем этапе определяется четкое значение переменной вывода:

$$z_1^* = a_1 x_0 + b_1 y_0,$$
$$z_2^* = a_2 x_0 + b_2 y_0.$$

Иллюстрирует алгоритм рисунок 7

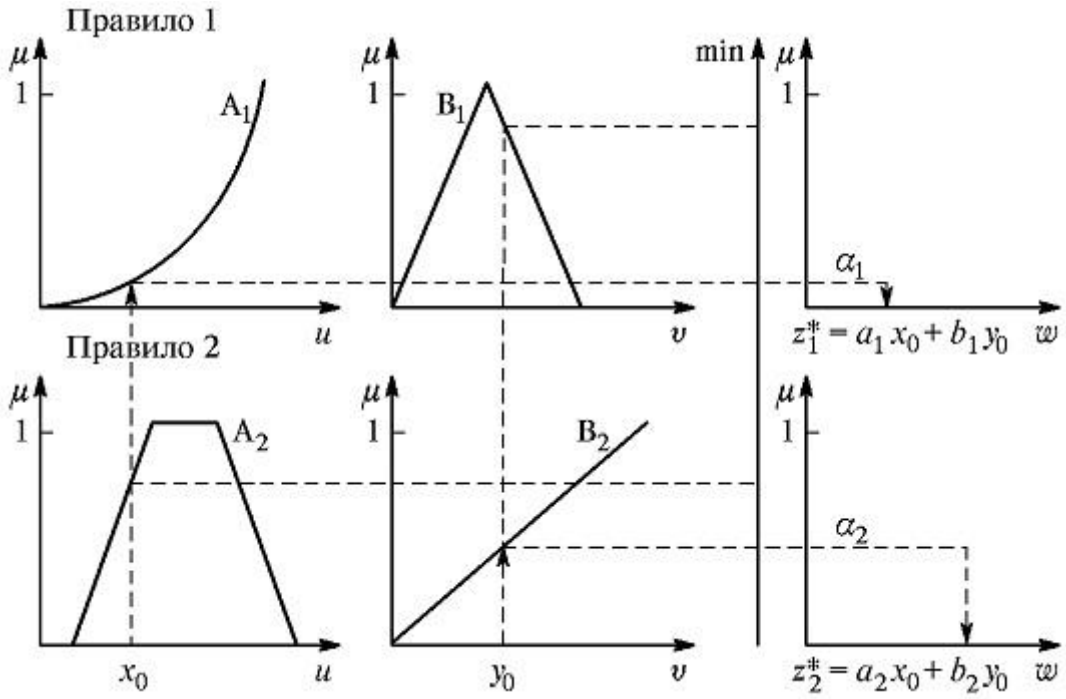

Рисунок 7 - Иллюстрация к алгоритму Sugeno

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

$\alpha_1 = A_1(x_0) \wedge B_1(y_0)$,

$\alpha_2 = A_2(x_0) \wedge B_2(y_0)$,

а затем — частные нечеткие подмножества

$\alpha_1 C_1(z)$, $a_2 C_2(z)$.

3. Находится итоговое нечеткое подмножество с функцией принадлежности

$\mu_s(z) = C(z) = (a_1 C_1(z)) \vee (a_2 C_2(z))$

(в общем случае n правил).

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

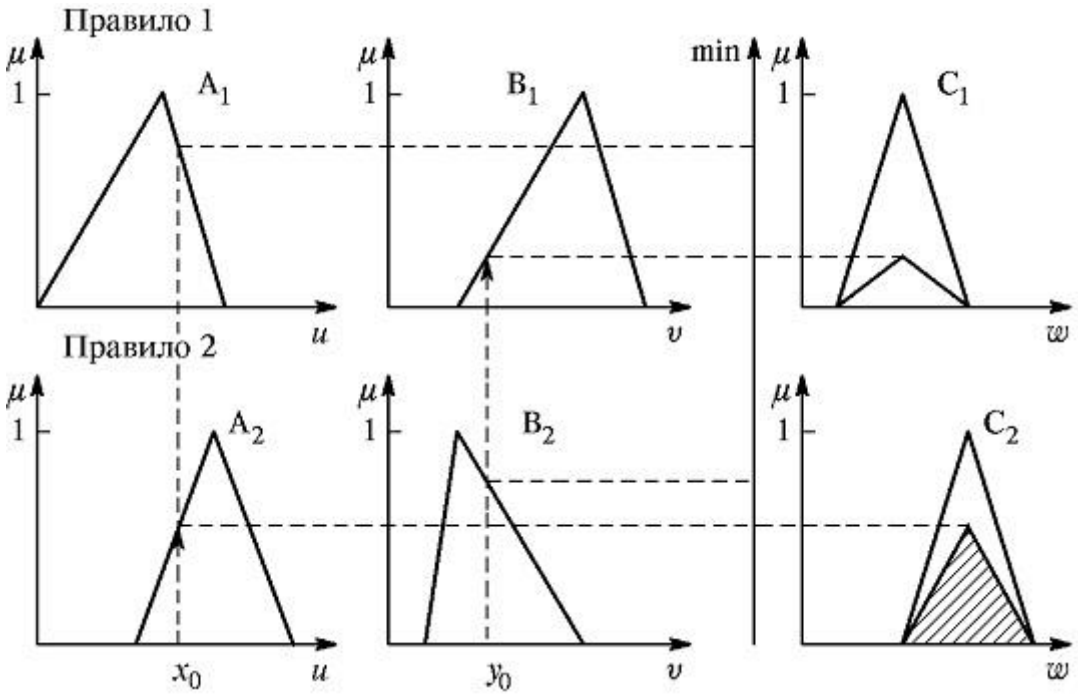

Рисунок 8 - Иллюстрация алгоритма Larsen

Упрощенный алгоритм нечеткого вывода

Исходные правила в данном случае задаются в виде:

$П_1$: если x есть A_1 и y есть B_1, тогда $z_1 = c_1$,

$П_2$: если x есть A_2 и y есть B_2, тогда $z_2 = c_2$, где c_1 и c_2 — некоторые обычные (четкие) числа.

Описание алгоритма

1. Первый этап — как в алгоритме Mamdani.

2. На втором этапе находятся числа $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$.

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

$$z_0 = \frac{\alpha_1 c_1 + \alpha_2 c_2}{\alpha_1 + \alpha_2},$$

или — в общем случае наличия n правил — по формуле

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_i c_i}{\sum_{i=1}^{n} \alpha_i}.$$

Иллюстрация алгоритма приведена на рисунке 9.

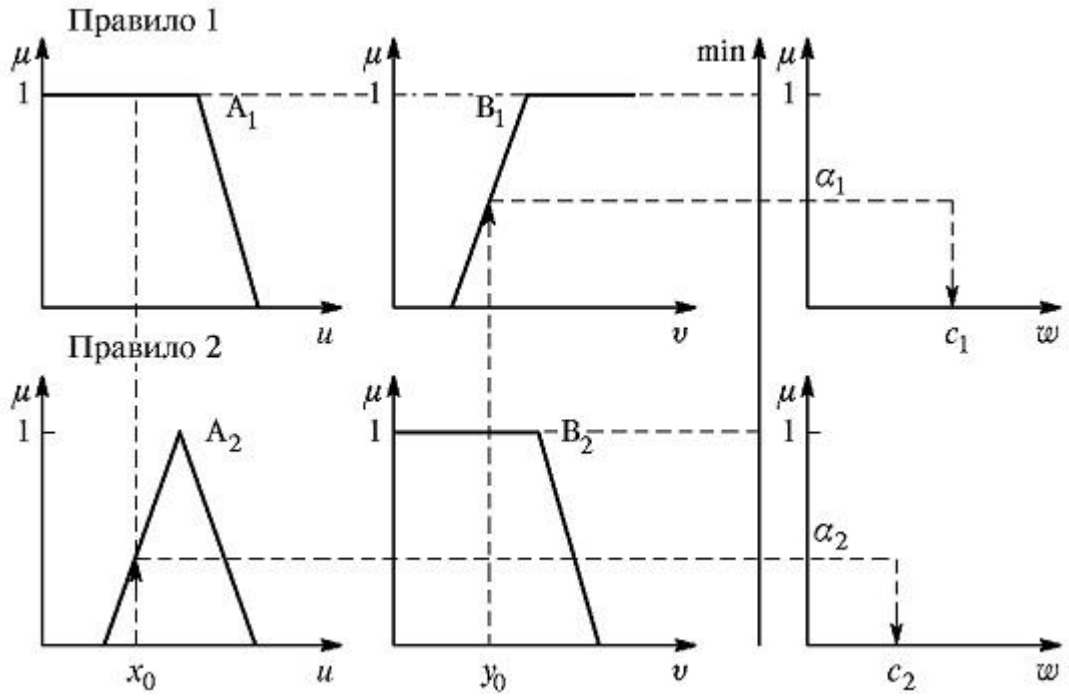

Рисунок 9 - Иллюстрация упрощенного алгоритма нечеткого вывода

Методы приведения к четкости

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

Для непрерывного варианта:

$$z_0 = \frac{\int\limits_{\Omega} zC(z)\,dz}{\int\limits_{\Omega} C(z)\,dz};$$

для дискретного варианта:

$$z_0 = \frac{\sum\limits_{i=1}^{n} \alpha_i z_i}{\sum\limits_{i=1}^{n} \alpha_i}.$$

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

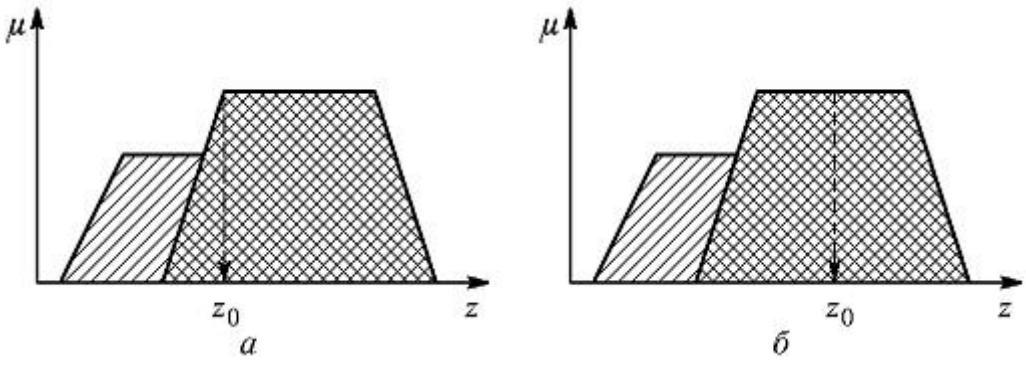

Рисунок 10 - Иллюстрация к методам приведения к четкости: α — первый максимум; б — средний максимум

$$z_0 = \min\left(z \middle| C(z) = \max_u C(u)\right).$$

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

$$z_0 = \frac{\int\limits_{\mathrm{G}} z\,dz}{\int\limits_{\mathrm{G}} dz},$$

где G — подмножество элементов, максимизирующих C.

Дискретный вариант (если C — дискретно):

$$z_0 = \frac{1}{N}\sum_{j=1}^{N} z_j.$$

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

$$z_0 \in \left\{ z | \mathrm{C}(z) = \max_u \mathrm{C}(u) \right\}.$$

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

$$z_0 = \frac{\int\limits_{\mathrm{C}\alpha} z\mathrm{C}(z)\,dz}{\int\limits_{\mathrm{C}\alpha} \mathrm{C}(z)\,dz},$$

где Cα — 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:

x_1— неисправность аккумулятора;

x_2— отработка машинного масла;

y_1— затруднения при запуске;

y_2— ухудшение цвета выхлопных газов;

y_3— недостаток мощности.

Между x_iи y_jсуществуют нечеткие причинные отношения $r_{ij} = x_i \rightarrow y_j$,которые можно представить в виде некоторой матрицы **R** с элементами $r_{ij} \in [0, 1]$. Конкретные входы (предпосылки) и выходы (заключения) можно рассматривать как нечеткие множества A и B на пространствах *X* и *Y*. Отношения этих множеств можно обозначить как

$$B = A \circ \mathbf{R},$$

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 *B* (или симптомы) и определяются входы *A* (или факторы).

Пусть знания эксперта-автомеханика имеют вид

$$\mathbf{R} = \begin{bmatrix} 0{,}9 & 0{,}1 & 0{,}2 \\ 0{,}6 & 0{,}5 & 0{,}5 \end{bmatrix},$$

а в результате осмотра автомобиля его состояние можно оценить как

$B = 0{,}9/y_1 + 0{,}1/y_2 + 0{,}2/y_3$.

Требуется определить причину такого состояния:

$A = a_1/x_1 + a_2/x_2$.

Отношение введенных нечетких множеств можно представить в виде

$$[0{,}9 \;\; 0{,}1 \;\; 0{,}2] = [a_1 \;\; a_2] \circ \begin{bmatrix} 0{,}9 & 0{,}1 & 0{,}2 \\ 0{,}6 & 0{,}5 & 0{,}5 \end{bmatrix},$$

либо, транспонируя, в виде нечетких векторов-столбцов:

$$\begin{bmatrix} 0{,}9 \\ 0{,}1 \\ 0{,}2 \end{bmatrix} = \begin{bmatrix} 0{,}9 & 0{,}6 \\ 0{,}1 & 0{,}5 \\ 0{,}2 & 0{,}5 \end{bmatrix} \circ \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}.$$

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

$$0{,}9 = (0{,}9 \wedge \alpha_1) \vee (0{,}6 \wedge \alpha_2),$$

$$0{,}1 = (0{,}1 \wedge \alpha_1) \vee (0{,}5 \wedge \alpha_2),$$

$$0{,}2 = (0{,}2 \wedge \alpha_1) \vee (0{,}5 \wedge \alpha_2).$$

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

$$0{,}9 = 0{,}9 \wedge \alpha_1,\ \alpha_1 \geq 0{,}9.$$

Из второго уравнения получим:

$$0{,}1 \geq 0{,}5 \wedge \alpha_2,\ \alpha_2 \leq 0{,}1.$$

Полученное решение удовлетворяет третьему уравнению, таким образом имеем:

$$0{,}9 \leq \alpha_1 \leq 1{,}0, \quad 0 \leq \alpha_2 \leq 0{,}1,$$

т.е. лучше заменить аккумулятор (α_1 — параметр неисправности аккумулятора, α_2 — 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) + $O(N^2)$ = = ***O(N^2)***	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) + $O(N^2)$ =	private double[] aggregation (double[] b) {
	int i = 0;
	int j = 0;

= ***O(N^2)***	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) + $O(N^2)$ = = ***O(N^2)***	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);
	}

$O(N) + O(N^2) + O(N^2) + O(N) =$
 $= \underline{\boldsymbol{O(N^2)}}$
	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 $O(N^2)$ - квадратичную сложность алгоритма.

	Квадратичная сложность алгоритма	Память	Точность
Алгоритм Mamdani	$O(N^2)$	9 блоков	99%
Алгоритм Tsukamoto	$O(N^2)$	5 блоков	96%
Алгоритм Sugeno	$O(N^2)$	8 блоков	98%
Алгоритм Larsen	$O(N^2)$	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, М. 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.