МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИБМП. 1 КАФЕДРА

ОЦЕНКА РЕФЕРАТА
РУКОВОДИТЕЛЬ

Л. П. Вершинина

РЕФЕРАТ

НЕЧЕТКИЕ ЭВМ И НЕЧЕТКОЕ ПО

по дисциплине: ОБРАБОТКА НЕЧЕТКОЙ ИНФОРМАЦИИ В СИСТЕМАХ ПРИНЯТИЯ РЕШЕНИЙ

РЕФЕРАТ ВЫПОЛНИЛ
СТУДЕНТ ГР.№ **M810M** Д. А. Горбачев

Санкт-Петербург
2018

НЕЧЕТКИЕ ЭВМ И НЕЧЕТКОЕ ПО

Горбачев Д. А.,

Санкт-Петербургский государственный
университет аэрокосмического приборостроения
Санкт-Петербург, 2018 г.

1 Поколения ЭВМ

От электронных ламп к нечетким схемам и нейронным ядрам

Специалисты, изучающие историю развития ЭВМ, делят все компьютеры на несколько поколений. Разбиение на группы происходит по ряду параметров, таких как архитектурные особенности, программное обеспечение, быстродействие, количество оперативной памяти и т.д. Но главное отличие машин из разных поколений состоит в элементной базе, на базе которой формируется определенная архитектура.

Для первого поколения ЭВМ (1946–1959 г.) характерно использование электронных (или электрических) ламп. Во втором поколении (1960–1969 г.) уже использовались полупроводники (транзисторы). Начало третьего поколения (1970–1979 г.) ознаменовал приход в употребление интегральных схем (плат). А в четвертом (1980–2000 г.) – инженеры встали на путь оптимизации и уменьшали размеры компонентов, тем самым размещая большую "вычислительную мощь" на схемах того же размера. Такие платы стали называть микропроцессорами.

В последствии развитие машин свелось к переходам к меньшим технологическим процессам изготовления. Известно, что закон Мура (количество транзисторов, размещенных на кристалле интегральной схемы, удваивается каждые 24 месяца) не выполняется уже примерно с 2012 года, так как производить ЦП по технологии менее 12 нм не выгодно. Поэтому в наше время все больше внимания приковано к альтернативам современного цифрового построения компьютера Фон Неймана. В следствии чего был осуществлен окончательный переход к многоядерной децентрализованной структуре ЭВМ, были разработаны новые способы ввода-вывода информации, добавлен искусственный интеллект и автоматизация процессов решения задач.

Еще в конце прошлого века (1970–1990 г.) можно было наблюдать за попытками реализации компьютеров "пятого поколения" в Японии и СССР. Основной идей были децентрализованные параллельные вычисления с прежней цифровой архитектурой. Не смотря на успех советских инженеров, проекты "Кронос" и "МАРС" [2] были скоро забыты и не возымели большой популярности, так как интегральные схемы в то время быстро заменялись новыми – более производительными, а технологии разработки ПО менялись в след за новой архитектурой. В наше время параллельные вычисления применяются повсеместно, начиная с персональных компьютеров, заканчивая серверными ЭВМ.

С развитием искусственного интеллекта, встала потребность в нечетких вычислениях и составлении нейронных сетей на основе элементной базы. Хоть цифровые компьютеры и универсальны, но операции с нечеткими переменными они выполняют не оптимально. Поэтому появилось желание иметь специальные аппаратные средства для осуществления нечетких выводов – нечеткие компьютеры. Аналогичная ситуация и с компьютерами, реализующими архитектуру нейронной сети – нейронными компьютерами. Использование нейронных процессоров уже дошло не только до персональных компьютеров, но и до телефонов, так Apple iPhone X (2017 г.) в своем ЦП A11 Bionic имеет нейронное ядро Neural Engine. Все такие компьютеры можно называть ЭВМ шестого поколения.

2 Нечеткие выводы и идея нечеткой ЭВМ

Работа с нечеткими величинами и идея нечетких компьютеров

Нечеткий вывод – получение нового логического заключения из правил вывода, которые хранятся в виде базы знаний и заданных фактов. В отличие от традиционных выводов все переменные в суждениях являются нечеткими переменными. Это можно представить следующим образом [1]:

$$\begin{array}{l}\text{(Знание) Если } x \text{ есть } A\text{, то } y \text{ есть } B\\ \text{(Факт) } x \text{ есть } A'\\ \hline \text{(Вывод) } y \text{ есть } B'\\ \textit{(восходящий нечеткий вывод)}\end{array}. \qquad (1)$$

Можно также рассмотреть и следующий вывод:

$$\begin{array}{l}\text{(Знание) Если } x \text{ есть } A, \text{ то } y \text{ есть } B\\ \text{(Факт) } y \text{ есть } B'\\ \hline \text{(Вывод) } x \text{ есть } A'\\ \textit{(нисходящий нечеткий вывод)}\end{array}. \qquad (2)$$

Здесь A, A', B, B' – нечеткие множества. Делать выводы по формулам (1, 2) с помощью классической логики крайне трудно. Где A может быть представлена как:

$$A = \{(x_i, a_i)\},\ x_i \in X,\ a_i \in [0, 1],$$

где X – полное множество (базовая шкала), a_i – степень принадлежности x_i множеству A. Очевидно, что определив базовые шкалы, два нечетких множества можно рассматривать как вектор:

$$A = (a_1, a_2, a_3, \ldots, a_i, \ldots, a_m),$$
$$B = (b_1, b_2, b_3, \ldots, b_j, \ldots, b_n),$$

где $a_i, b_j \in [0, 1]$. Если A – причина, а B – результат, то можно определить матрицу, отражающую причиненные отношения между A и B. Она называется нечетким отношением R из A в B:

$$R = \begin{bmatrix} r_{11} & \ldots & r_{1n} \\ \vdots & r_{ij} & \vdots \\ r_{m1} & \ldots & r_{mn} \end{bmatrix}.$$

Заде определил следующим образом результат вывода B' по формуле (1):

$$\begin{aligned} B' &= (b'_1, b'_2, b'_3, \ldots, b'_j, \ldots, b'_n) = \\ &= (a'_1, a'_2, a'_3, \ldots, a'_j, \ldots, a'_m) \circ R = \\ &= A' \circ B, \end{aligned}$$

или

$$b'_j = \bigvee_i a'_j * r_{ij}, \qquad (3)$$

где $*$ – например, операция MIN, алгебраическое произволение или другая операция. Мамдани предложил следующее нечеткое отношение и показал его уместность:

$$r_{ij} = a_i \wedge b_j.$$

Основная идея нечеткого компьютера заключается в том, что он, в силу своей элементной базы, мог бы параллельно выполнять восходящие и нисходящие нечеткие выводы по формулам с помощью анализа сигналов, а не цифровых значений, соответственно задавая степени принадлежности для конкретных нечетких множеств, например, силой тока или же напряжением.

Такие сигналы можно передавать по m сигнальным линиям. В принципе нечеткий компьютер состоит из "нечеткой памяти", позволяющей хранить вышеописанные сигналы, "машины нечетких выводов", а также, при необходимости, блока преобразования в достоверные данные (дефадзификатора). Для работы этих трех блоков необходимо спроектировать специальные электрические схемы [1].

3 Проектирование нечеткой ЭВМ

Стандартные блоки и архитектура нечеткой ЭВМ

Функции нечеткой логики в отличие от двухзначной или многозначной логики в принципе можно определить многими способами. В настоящее время известно несколько десятков таких функций. Но достаточно часто используют функции нечеткой логической суммы (MAX) и нечеткого логического произведения (MIN). В зависимости от типа схема (в режиме тока, в режиме напряжения) появляются преимущества и недостатки. Так первые хороши для операций реализации операций сложения и вычитания через малое количество транзисторов, но не терпят высокие нагрузки (нагрузочная способность равна 1). Вторые же не подходят для вышеизложенных операций, но имеют высокую нагрузочную способность [1].

Устройство хранения – нечеткая память, может представлять собой плату с u ($u = \operatorname{card} P$, где P – функция принадлежности) ячейкой [1]. По сути нечеткая память должна уметь сохранять функции принадлежности.

Машина нечетких выводов представляет собой комбинацию MIN-схемы, MAX-схемы. Она позволяет осуществлять вывод (выражение 1) с использованием формулы (3). Например, можно спроектировать машину нечетких выводов (рисунок 1) с m параллельно работающими двухвходными схемами MIN, а MAX с m входами и одним выходом. При этом схема усечения – это n параллельно работающих двухвходных схем MIN. Один из двух входов каждой схемы подключен к входу a. Если ввести A, B из знания "если x есть A, то y есть B", полученного из базы знаний, а также A' из фактической информации "X есть A'", то на выходе B как факт "y есть B'" появятся напряжения в виде дискретной функции принадлежности. При этом выход B' представлен n сигнальными минами (или выходными контактами).

Прочитать подробнее о реализации электрических схем можно в книге "Прикладные нечеткие системы" [1].

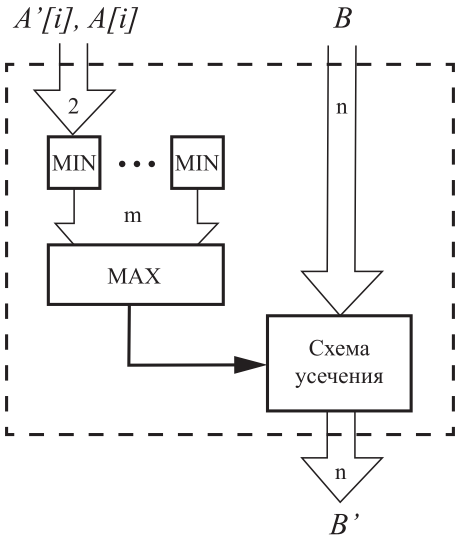

Рис. 1: Блок-схема машины нечетких выводов

Для проектирования быстродействующей нечеткой ЭВМ на основе стандартных нечетких блоков, рассмотренных ранее, необходимо использовать принцип параллелизма (рисунок 2): для одновременного выполнения r нечетких выводов, установим r машин нечетких выводов и будем вводить в них параллельно нечеткие слова. Кроме того, все нечеткие слова в каждом выводе распределим в виде напряжений по m или n сигнальным шинам (сигнальным линиям) и сигналы на каждой шине будем обрабатывать независимо и параллельно [1]. В конце, для r правил определим связь. Если использовать связь “а также”, то правильно будет разместить блок MAX на выходе. Далее нужно лишь отправить сигналы в дефадзификатор для получения четкого значения.

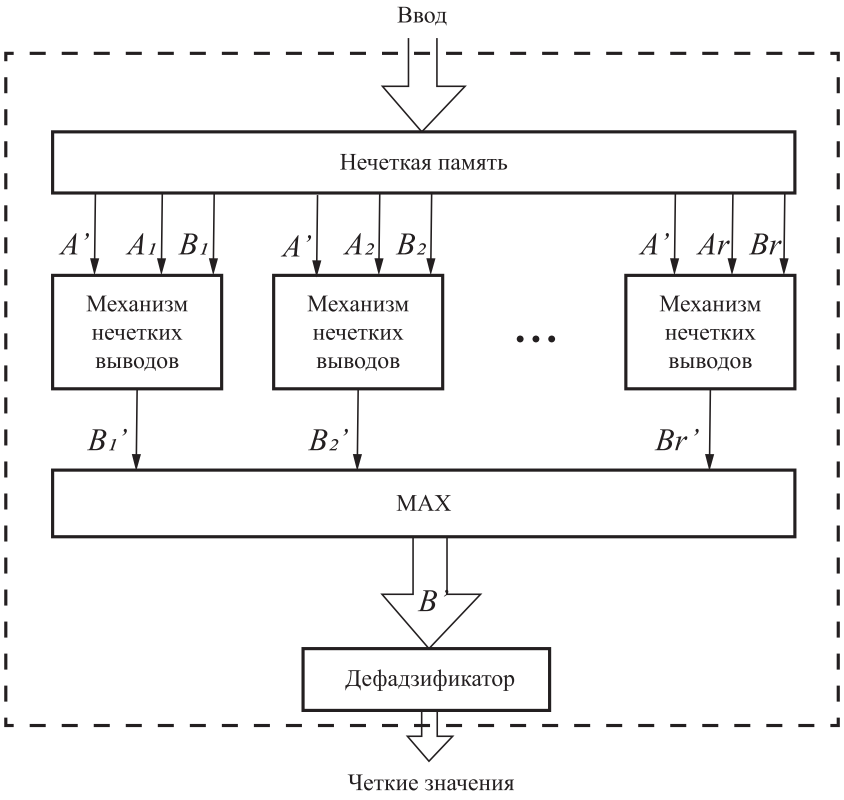

Рис. 2: Базовая архитектура нечеткого компьютера

4 Нечеткое ПО

Инструментарий и запуск контрольного примера

Под нечетким ПО подразумевают программы, выполняемые на цифровых компьютерах. Рассмотрев устройство архитектуры нечеткой ЭВМ, становится очевидным тот факт, что нечеткий компьютер непригоден для большинства задач, даже статистических, поэтому он выполняет роль некоторого "регистра".

В книге "Прикладные нечеткие системы" [1] рассказывается про Нечеткий пролог. К сожалению, в данное время, 15 лет спустя, Нечеткий пролог, да и другие многие разработки в сфере нечеткой математики недоступны, или же некорректно запускаются на современных ОС. Большим недостатком этого "нечеткого ПО" является и то, что оно в большинстве своем написано под 32-битную архитектуру x86.

Благодаря статье "A Survey of Fuzzy Systems Software: Taxonomy, Current Research Trends and Prospects" [3] (2016 г.) удалось найти более "новое" ПО, написанное под современную архитектуру x64. Но на практике оказалось, что разработкой библиотек для реализации нечетких вычислений чаще всего занимаются младшие научные сотрудники различных университетов по всему миру. Поэтому эти библиотеки сложнодоступны и плохо поддерживаются. Из-за этого список ПО сильно сократился:

Название	Тип	Язык
CIRG	Код	Java
NNandFS	Код	C++, Java
DotFuzzy	Библиотека	C# (.NET)
FINK-FLTO	Библиотека	Octave
FlouLib	Библиотека	Matlab
FLP	Библиотека	C++
FRBS	Библиотека	R
fugeR	Библиотека	R
FuzzyLite	Библиотека	C++, Java
pyfuzzy	Библиотека	Python
sets	Библиотека	R
Matlab-FLT	Плагин	Matlab
FuzzyLogicTools	Пакет	C++
fuzzyTECH	Пакет	C++, C, Jav
WEKA	Пакет	Java

Особое внимание имеет смысл обратить на плагин для Matlab. Он позволяет в графическом интерфейсе на основе блок-схем моделировать экспертную систему, а так же "программировать" контроллеры на основе нечеткой логики. Пакет для искусственного анализа WEKA [4] известен широким спектром возможностей. С помощью пакета можно также осуществить нечеткий вывод. Но в целях установить полноценный диалог с ЭВМ, принято решение отбросить готовые пакеты в сторону и осуществить запуск тестовых нечетких вычислений, используя библиотеку. В списке присутствует библиотека с названием sets (в пер. с англ. "множества"). Она хорошо документирована [6]. Поэтому всяко-разные проблемы с ней исключены.

Рассмотрим sets [5]. Для установки библиотеки в консоли нужно вставить команду:

```
install.packages("sets")
```

Для того, чтобы использовать библиотеку sets, в начале исполняемого файла, нужно подключить библиотеку

```
library(sets)
```

Если вы получите ошибку при запуске файла, значит библиотека sets не установлена. После подключения библиотеки, нужно произвести настройки нечеткой системы, а именно указать универсальную базовую шкалу, к которой будут приведены остальные базовые шкалы.

```
sets_options("universe", seq(from = 0,
to = 40, by = 0.1))
```

Вышеизложенная команда установит в качестве универсальной базовой шкалу шкалу с началом в 0, концом в 40, шагом (делением) 0.1. Если ваши переменные будут иметь другие базовые шкалы, то они будут нормализованы относительно универсальной шкалы.

Выбрать аксиоматику можно командой:

```
fuzzy_logic("Zadeh")
```

Задавать термы можно с помощью уже готовых функций, например, "треугольная функция" (fuzzy_cone), или трапецевидная функция (fuzzy_trapezoid) или s-образная функция (fuzzy_sigmoid) и т.д.

```
service <- fuzzy_partition(varnames =
c(poor = 0, good = 5, excellent = 10),
sd = 1.5)
```

Правила задаются следующим образом:

```
fuzzy_rule(service %is% poor ||
food %is% rancid, tip %is% cheap)
```

Строку "a %is% b" можно перевести как "a есть b" (как в выражении 1).

Ниже представлен полный код примера, с расчетом модели по которой вычисляются чаевые. Три переменных: качество еды (food), качество сервиса (service), количество чаевых (tip).

```
## Задание базовой шкалы
sets_options("universe", seq(from = 0, to = 25, by = 0.1))
## Задание нечетких переменных
variables <-
  set(service =
        fuzzy_partition(varnames =
                          c(poor = 0, good = 5, excellent = 10),
                        sd = 1.5),
      food =
        fuzzy_variable(rancid =
                         fuzzy_trapezoid(corners = c(-2, 0, 2, 4)),
                       delicious =
                         fuzzy_trapezoid(corners = c(7, 9, 11, 13))),
      tip =
        fuzzy_partition(varnames =
                          c(cheap = 5, average = 12.5, generous = 20),
                        FUN = fuzzy_cone, radius = 5)
  )
## Задание правил
rules <-
  set(
    fuzzy_rule(service %is% poor || food %is% rancid,
               tip %is% cheap),
    fuzzy_rule(service %is% good,
               tip %is% average),
    fuzzy_rule(service %is% excellent || food %is% delicious,
               tip %is% generous)
  )
## Объединение в систему
system <- fuzzy_system(variables, rules)
print(system)
plot(system)
## Вывод графиков переменных
## Осуществление вывода
fi <- fuzzy_inference(system, list(service = 3, food = 8))
## Вывод графика с результирующим нечетким множеством
plot(fi)
## Дефадзификатор
gset_defuzzify(fi, "centroid")
## Сброс базовой шкалы
sets_options("universe", NULL)
```

Список литературы

[1] Асаи К. Прикладные нечеткие системы: Пер. с япон. / К. Асаи, Д. Ватада, С. Иваи; под редакцией Т. Тэрано, К. Асаи, М. Сугэно – М.: Мир, 1993. – 368 с.

[2] Богатырев Р. Язык как основа архитектуры. Проект «Кронос» и путь к технологиям XDS – 1998 // Режим доступа: http://computer-museum.ru/histussr/kronos.htm

[3] Alcalá-Fdez J. A Survey of Fuzzy Systems Software: Taxonomy, Current Research Trends and Prospects / J. Alcalá-Fdez, José M. Alonso – IEEE Transactions on Fuzzy Systems 24:1, 2016 – p. 40-56.

[4] Baldwin J. F. Simple fuzzy logic rules based on fuzzy decision tree for classification and prediction problem / J. F. Baldwin, D. Xie – IFIP International Federation for Information Processing – p. 175–184.

[5] Heaton J. Fuzzy Logic in R / J. Heaton – Forecasting & Futurism July 2014 – Issue 9 – p. 35–38.

[6] Meyer D. Package 'sets' – 2017 // Access mode: https://cran.r-project.org/web/packages/sets/sets.pdf