УДК 004.652+004.89
ББК 32.973.26-018.2.75

ОРГАНИЗАЦИЯ ХРАНЕНИЯ НЕЙРО-НЕЧЕТКИХ МОДЕЛЕЙ В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ[1]

Алексеев В. А.[2], Сараев П. В.[3], Домашнев П. А.[4], Назаркин О. А.[5]
(*Липецкий государственный технический университет, Липецк*)

Рассматривается организация хранения нейро-нечетких моделей типа ANFIS в реляционных базах данных. Разработаны две схемы хранения: универсальная, основанная на представлении модели ANFIS как функции многих переменных в виде дерева вычисления, и специфическая, отражающая слоистую структуру модели. Показаны способы организации вычислений по моделям ANFIS для двух вариантов схемы хранения. Обоснована возможность вычислений по модели ANFIS стандартными средствами языка SQL.

Ключевые слова: нейро-нечеткая модель ANFIS, дерево вычислений, реляционные базы данных, математические вычисления на SQL.

---

[1] *Исследование выполнено при финансовой поддержке РФФИ и Администрации Липецкой области в рамках научного проекта №14-47-03611-р_центр_а.*

[2] *Владимир Александрович Алексеев, кандидат технических наук* (*alexeev48@gmail.com*).

[3] *Павел Викторович Сараев, доктор технических наук, доцент* (*psaraev@yandex.ru*).

[4] *Павел Алексеевич Домашнев, кандидат технических наук* (*pdomashnev@gmail.com*)

[5] *Олег Александрович Назаркин, кандидат технических наук* (*nazarkino@mail.ru*)

1. Введение

Нейро-нечеткие модели находят применение в системах моделирования и управления техническими, социальными и другими системами [1, 4, 7, 8]. Практическое применение нейро-нечетких моделей в таких системах требует разработки схемы их постоянного хранения. Однако работы по данной тематике сосредоточены в основном на применении класса нейро-нечетких моделей к решению конкретных прикладных задач [7] или синтезу методов обучения таких моделей [4]. Разрабатываемые программные комплексы включают базы моделей [7] или базы данных и знаний [4], однако схема и средства их хранения не раскрываются, а вычисления по моделям реализуются прикладным приложением. В [8] предлагается оригинальная методика автоматической генерации программного кода, реализующего вычисления по построенной модели, для встраивания его в прикладные приложения.

В связи с этим представляется актуальным исследование и разработка схемы постоянного хранения нейро-нечетких моделей в базе данных. При этом цели могут быть различными – от сохранения некоторого текущего состояния модели, которая затем будет улучшаться, и сопоставления нескольких моделей, построенных с применением различных алгоритмов, до многократного использования построенной модели для прогнозирования и управления техническими, социальными и другими системами. Общими требованиями к схеме постоянного хранения в системах моделирования является простота использования и эффективность вычислений по модели. Простота использования подразумевает минимальные трудозатраты на интеграцию нейро-нечетких моделей в прикладные системы. Эффективность вычислений, главным образом, связана с производительностью таких систем. В данной работе в качестве средства хранения рассматриваются реляционные базы данных, как наиболее доступные и распространенные системы, для которых имеются

хорошо отработанные стандартные средства доступа к данным и язык запросов SQL[1].

Целью данного исследования является разработка вариантов схемы данных для хранения нейро-нечетких моделей в реляционных базах данных и способов организации вычислений для хранимых в базе данных моделей.

2. Структура модели ANFIS

Нейро-нечеткая модель ANFIS реализует обобщенную схему вывода Такаги – Сугено – Канга (TSK) [1, 13, 14]. Пример структуры модели для двух входных переменных представлен на рис. 1.

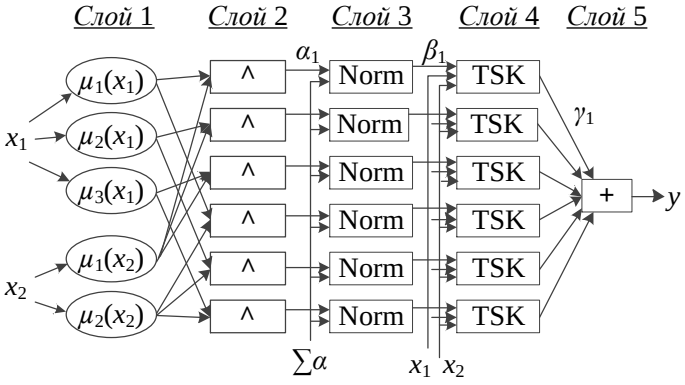

Рис. 1. Пример структуры нейро-нечеткой модели ANFIS для двух входных переменных

[1] *Авторы, тем не менее, понимают преимущества, которые дает использование объектно-ориентированных СУБД для решения поставленных задач, и планируют рассмотреть особенности хранения нейро-нечетких моделей в таких СУБД и возможности организации вычислений на их основе.*

Рассмотрим подробнее функции нейронов каждого уровня модели (N – количество правил в модели, M – количество входных переменных) [1, 13, 14]:

Слой 1. Выходы нейронов этого слоя представляют собой степени принадлежностей входных значений нечетким множествам, ассоциированным с нейронами. Обычно используется гауссовская функция принадлежности:

$$(1)\quad \mu_k(x_j) = G(a_{jk}, b_{jk}, x_j) = \exp\left[-\frac{1}{2}\left(\frac{x_j - a_{jk}}{b_{jk}}\right)^2\right],$$

где $j = 1, \ldots, M$ – номер переменной; $k = 1, \ldots, K_j$ – номер фаззификатора для переменной j; a_{jk}, b_{jk} – множество параметров.

Слой 2. Каждый нейрон этого слоя вычисляет уровень истинности правила i по формуле

$$(2)\quad \alpha_i = \mu_{k[1,i]}(x_1) \wedge \ldots \wedge \mu_{k[M,i]}(x_M),$$

где оператор $k[j, i]$ определяет номер используемого в правиле i фаззификатора переменной j, а для моделирования логической связки «И» может использоваться любая дифференцируемая t-норма, в частности, произведение, т.е.

$$(3)\quad \alpha_i = \prod_{j=1}^{M} \mu_{k[j,i]}(x_j).$$

Слой 3. Реализует нормализацию уровней истинности каждого правила по формуле

$$(4)\quad \beta_i = \frac{\alpha_i}{\sum_{i=1}^{N} \alpha_i}.$$

Слой 4. Выходы нейронов представляют произведение нормализованных значений уровней истинности на соответствующие выходы правил TSK:

$$(5)\quad \gamma_i = \beta_i\left(c_{i0} + \sum_{j=1}^{M} c_{ij} x_j\right).$$

Слой 5. Выполняет суммирование выходов нейронов предыдущего слоя, формируя выход модели ANFIS:

$$(6) \quad y = \sum_{i=1}^{M} \gamma_i \ .$$

С учетом (1)–(6) модель ANFIS реализует следующую функциональную зависимость:

$$(7) \quad F(x) = \frac{1}{\sum_{i=1}^{N} \left[\prod_{j=1}^{M} \mu_{k[j,i]}(x_j) \right]} \sum_{i=1}^{N} \left(\left[\prod_{j=1}^{M} \mu_{k[j,i]}(x_i) \right] \times \left[c_{i0} + \sum_{j=1}^{M} c_{ij} x_j \right] \right).$$

Схема хранения модели ANFIS может основываться на универсальной модели данных для представления математических выражений или учитывать особенности слоистой структуры модели. Целью работы является разработка и исследование подходов к построению схемы хранения и организации вычислений по модели ANFIS.

3. Представление математических выражений в системах моделирования

Для математического выражения общего вида можно построить дерево вывода на основе трех множеств [2, 10]:

1. Множество операндов математического выражения, включающее константы и переменные *L*.

2. Множество операций математического выражения, куда входят все операторы и функции данного математического выражения *A*.

3. Множество узлов дерева вывода математического выражения, содержащее множество результатов промежуточных математических операций (вектор результатов промежуточных операций) *N*.

Частным случаем такого представления является вычисляемое выражение для скалярной функции *n* переменных. Так, например, функция трех переменных

$$(8) \quad f(x) = 5\ln(x_1) + \frac{6e^{x_2}}{\sqrt{x_1 x_3}}$$

может быть представлена в виде дерева, структура которого приведена на рис. 2.

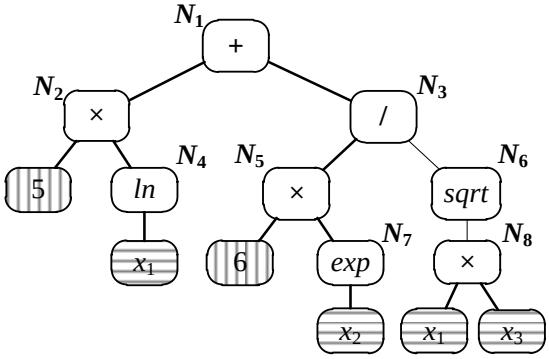

Рис. 2. Пример структуры дерева вычисления скалярной функции трех переменных

Для данного примера множество операндов:

$$L = \{5;6;x_1;x_2;x_3\},$$

множество операций:

$$A = \{"+";"\times","/";"\exp";"\ln";"sqrt"\},$$

множество узлов:

$$N = \{N_1;N_2;N_3;N_4;N_5;N_6;N_7;N_8\}.$$

Множество A для данного примера включает унарные и бинарные элементарные функции, в результате выражение представляется в виде двоичного дерева. В общем случае могут использоваться функции нескольких (более двух) переменных, соответственно математическое выражение будет представлено N-арным деревом.

При условии, что численно определены значения операндов, вектор результатов промежуточных операций будет содержать численный результат выполнения каждой операции. В этом случае для расчета значения выражения достаточно применить алгоритм, аналогичный вычислению по обратной польской (постфиксной) записи [10], с сохранением результатов промежуточных вычислений в узлах дерева. Преимуществами

такого алгоритма вычисления является возможность проведения расчетов путем однократного просмотра дерева снизу вверх и поддержка операций с любым количеством операндов.

Системы моделирования, как правило, работают с известной структурой моделей, (например, линейной, полиномиальной, нейросетевой, нейро-нечеткой [3, 11, 13, 14]), и в каждом конкретном случае использование для расчетов универсального представления математических выражений является избыточным и может привести к снижению производительности вычислений. Однако в исследовательских системах, реализующих комплекс методов моделирования и, соответственно, работающих с моделями различной структуры, представляется целесообразным разработка единого, универсального механизма вычислений и хранения моделей.

В системах, ориентированных на вычисления по математическим моделям, целесообразно учесть типовые особенности этих моделей в структуре хранения и алгоритме вычисления для сокращения количества узлов и уровней в дереве вычисления.

Так, например, регрессионные модели представляются в виде [3]

(9) $f(x) = \alpha_0 + \alpha_1\varphi_1(x_1) + \alpha_2\varphi_2(x_2) + ... + \alpha_N\varphi_N(x_N)$,

где α_i, $i = 1, \ldots, N$, – параметры модели; $\varphi_i(x_i)$, $i = 1, \ldots, N$, – функции-преобразователи; x_i, $i = 1, \ldots, N$, – независимые переменные. В частном случае регрессионная модель линейная, т.е. $\varphi_i(x_i) = x_i$. Для моделей такой структуры целесообразно включить в множество операций A параметризованные функции вида $\alpha_i\varphi_i(\cdot)$ и оператор суммирования N переменных Σ.

Для нейро-нечетких моделей типа ANFIS [1, 13, 14] с гауссовой функцией фаззификации в множество операций A целесообразно включить параметризованную функцию Гаусса (1), операторы суммирования и произведения N переменных Σ, Π.

Вообще использование параметризованных функций во многих случаях позволяет сократить количество уровней дерева вычислений, не нарушая общего принципа построения дерева. Сложность здесь представляет адекватное формирование множества параметризованных операций с учетом структуры ис-

пользуемых моделей. В качестве общей рекомендации можно предложить использование параметра-мультипликатора для всех операций, что позволяет сократить количество уровней дерева выражений как в регрессионных моделях (9), так и в нейро-нечетких (7) за счет правил TSK (5).

Расширим понятие дерева вывода выражения [2, 10], добавив в него множество параметров P, связанных с соответствующими элементами множества операций A. С позиций вычислений по математическим моделям представляется логичным относить параметры к тому же уровню, что и соответствующий параметризованный оператор выражения, так как значения параметров в модели не являются независимыми, не имеют самостоятельного смысла и не могут существовать вне контекста этого оператора. На рис. 3 представлена модифицированная структура дерева вычисления для выражения (8) с применением параметров-мультипликаторов. Как видно, сократилось количество узлов дерева вывода с 8 до 6. В то же время общее количество уровней для данного дерева вычислений осталось неизменным (сократилось лишь для некоторых ветвей дерева).

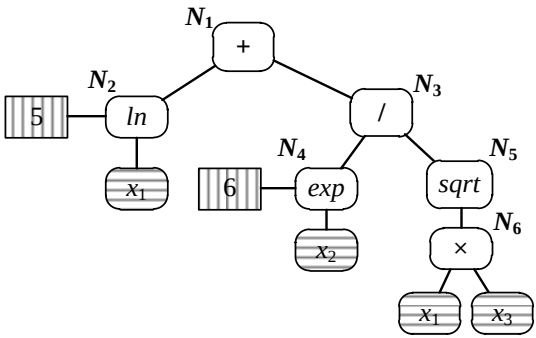

Рис. 3. Модифицированная структура дерева вычисления с применением параметров-мультипликаторов

4. Универсальная схема хранения математических выражений в реляционной БД

Хранение дерева математического выражения может быть осуществлено в базе данных под управлением любой современной СУБД. Концептуальная модель данных [5] для хранения математического выражения может быть представлена следующей диаграммой сущность–связь (Entity-Relationship, ER) в нотации Crow's Foot [5]:

Рис. 4. ER-диаграмма концептуальной модели данных для хранения математических выражений

В представленной модели данных количество параметров для каждого узла дерева вычислений принято равным не более двух (в зависимости от используемой функции), что в большинстве случаев достаточно. При необходимости количество параметров может быть увеличено без принципиального изменения

модели данных – за счет добавления нужного количества атрибутов для сущности «элемент выражения» или введения дополнительной сущности «параметр». Атрибут «значение параметра 1» также используется для хранения значений операндов-констант выражения.

Данная работа направлена на разработку схемы хранения моделей в реляционной базе данных, поэтому приведем диаграмму логической реляционной модели данных, основываясь на правилах преобразования концептуальной модели [5], без привязки к какой-либо конкретной реляционной СУБД:

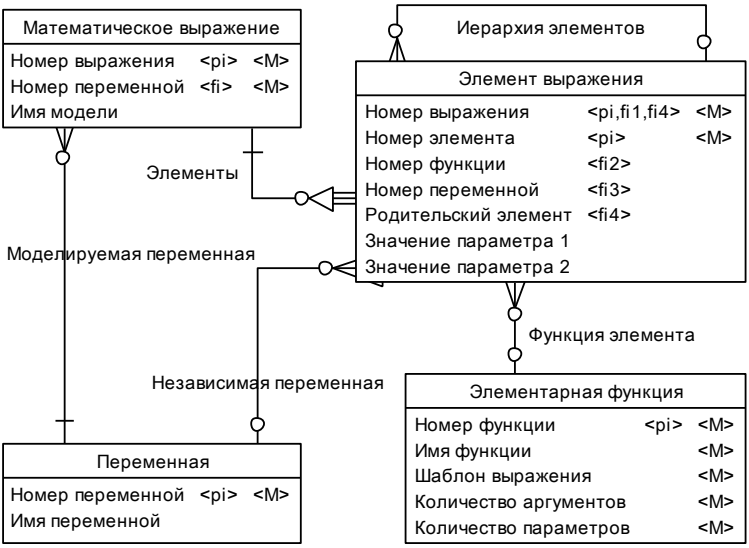

Рис. 5. Диаграмма логической модели данных для хранения математических выражений

Таким образом, четырех реляционных отношений достаточно, чтобы хранить дерево вычислений для модели, представленной скалярной функцией *n* переменных. Так, например, функция (8) может быть представлена в реляционной БД в виде следующих таблиц (таблицы 1–4):

Таблица 1. Пример содержания таблицы «Переменная»

Номер переменной	Имя переменной
1	x1
2	x2
3	x3
4	y1

Таблица 2. Пример содержания таблицы «Математическое выражение»

Номер выражения	Имя модели	Моделируемая переменная
1	F(x)	4

Таблица 3. Пример содержания таблицы «Элементарная функция»

Номер функции	Имя функции	Шаблон выражения	Количество аргументов	Количество параметров
1	x	{p0}×{x0}	1	1
2	+	{p0}× ({x0}+{x1})	2	1
3	*	{p0}×({x0})×({x1})	2	1
4	/	{p0}× ({x0})/({x1})	2	1
5	ln(x)	{p0}×ln({x0})	1	1
6	sqrt(x)	{p0}×sqrt({x0})	1	1
7	exp(x)	{p0}×exp({x0})	1	1
8	1 / x	{p0}/({x0})	1	1

Таблица 4. Пример содержания таблицы «Элемент выражения»

Номер элемента	Номер выражения	Родительский элемент	Номер функции (имя функции[1])	Номер переменной	Значение параметра 1
1	1	–	2 (+)	–	1
2	1	1	5 (ln(x))	1	5
3	1	1	4 (/)	–	6
4	1	3	7 (exp(x))	2	1
5	1	3	6 (sqrt(x))	–	1
6	1	5	3 (*)	–	1
7	1	6	–	3	1
8	1	6	–	1	1

Представленная модель данных позволяет хранить в реляционной БД широкий класс математических моделей, представимых в виде скалярной функции *n* переменных, включая нейро-нечеткие модели типа ANFIS.

5. Организация вычислений по модели, представленной в универсальной схеме хранения

Вычисления по математическому выражению, сохраненному в представленной выше схеме (рис. 5), могут быть проведены:

а) средствами хранимых процедур, реализованных в СУБД (например, на языке PL/SQL при хранении моделей в БД под управлением Oracle);

б) в прикладном приложении, реализованном на языке высокого уровня.

[1] *Имя функции указывается только для удобства восприятия*

Первый способ требует детального рассмотрения проблем, связанных с математическими вычислениями в СУБД средствами процедурных расширений языка SQL (таких как Oracle PL/SQL или Microsoft Transact-SQL), в частности полноты языков в части реализованных математических функций, производительности вычислений, и выходит за рамки данной статьи.

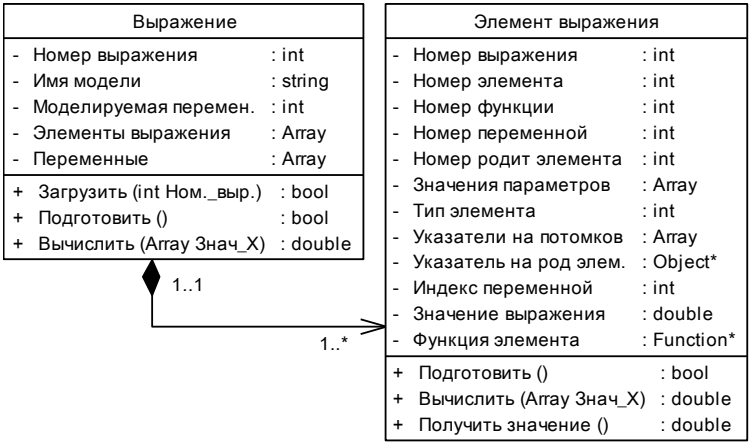

Рис. 6. Структура классов для вычисления математического выражения

При обработке математического выражения средствами объектно-ориентированных языков программирования достаточно реализовать 2 класса, состав атрибутов и операций которых в нотации UML представлен на рис. 6. При этом должно быть обеспечено однозначное соответствие номеров элементарных функций в базе данных и их реализаций в коде программы.

Применение иерархического дерева для вычисления выражений требует его обход «снизу вверх», когда при посещении очередного узла все его потомки оказываются пройденными. Поэтому массив элементов выражения индексируется так, что потомки всегда имеют больший номер по сравнению с рассматриваемым элементом. Тогда остается вычислить значение вложенного выражения снизу вверх по упорядоченным значениям элементов массива, сохраняя эти значения в поле «Значение

выражения». Следует отметить, что индексирование и формирование массивов указателей на потомков производится один раз после загрузки выражения и его элементов из базы данных.

Недостаток данного подхода к вычислению математического выражения состоит в том, что процесс организован в режиме «интерпретатора»: считанные номера элементарных функций сопоставляются с известными вариантами, и осуществляется вызов соответствующей программной реализации функции. Повысить эффективность вычислений можно за счет предварительной обработки номеров элементарных функций с сохранением в элементах выражения адресов программной реализации данных функций. Предварительная обработка может осуществляться один раз при подготовке выражения к вычислениям.

При этом набор реализуемых «элементарных» функций будет расширяемым и может быть реализован в виде «словаря», устанавливающего соответствие между идентификатором функции (совпадающем с номером функции в базе данных) и ее программной реализацией, например:

```
1   //бинарный оператор суммирования с параметром
2   private double FunctionAdd(double[] _PVals, double[] _XVals)
3   {   return _PVals[0]*(_XVals[0]+_XVals[1]); }
4   //бинарный оператор умножения с параметром
5   private double FunctionMultiply(double[] _PVals, double[]
    _XVals)
6   {   return _PVals[0]*(_XVals[0]*_XVals[1]); }
7   private delegate double BasicFunction (
8   double _PVals[],            //значения параметров
9   double _XVals[] );          //значения переменных
10  private Dictionary<int, BasicFunction> ExpressionOperations;
11  ExpressionOperations = new Dictionary<int, BasicFunction>
12  {   { 2, FunctionAdd }, } };
13      { 3, FunctionMultiply
```

Листинг 1. Вариант реализации «словаря» элементарных функций на языке C#

Возможный алгоритм загрузки выражения из реляционной базы данных может быть представлен следующим образом:

Алгоритм 1. Загрузка математического выражения из реляционной базы данных

1: Получить информацию о требуемом математическом выражении *M* из базы данных:
SELECT * FROM [Математическое выражение]
WHERE [Номер выражения] = *M*
2: Установить значения переменных объекта «Выражение»: «Номер выражения», «Имя модели», «Моделируемая переменная».
3: Получить информацию о независимых переменных, используемых в выражении *M*:
SELECT DISTINCT [Номер переменной]
FROM [Элемент выражения]
WHERE [Номер выражения] = *M*
AND [Номер переменной] IS NOT NULL
ORDER BY [Номер переменной]
4: Занести номера независимых переменных в массив «Переменные» объекта «Выражение».
5: Получить информацию об элементах выражения из базы данных в порядке возрастания их идентификаторов:
SELECT * FROM [Элемент выражения]
WHERE [Номер выражения] = *M*
ORDER BY [Номер элемента]
6: Для каждой строки результирующей таблицы:
7: Создать объект «Элемент выражения» *A*, установив значения переменных объекта *A* равными значениям соответствующих атрибутов из базы данных: номер выражения, номер элемента, номер функции, номер переменной, номер родительского элемента, значения параметров.
8: Добавить элемент *A* в массив элементов выражения.

Возможный алгоритм подготовки загруженного выражения *B* к вычислениям:

Алгоритм 2. Подготовка загруженного выражения *B* к вычислениям

1: Для каждого элемента *A* из массива *B*.[Элементы выражения], начиная с первого, выполнить:
2: Если *A*.[Номер функции] > 0, то:
3: Установить указатель *A*.[Функция элемента] из словаря функций по идентификатору *A*.[Номер функции].
4: Если *A*.[Номер переменной] > 0, то:
5: Занести в *A*.[Индекс переменной] индекс переменной *A*.[Номер переменной] в массиве *B*.[Переменные].
6: Просмотреть каждый последующий элемент *C* из массива *B*.[Элементы выражения]:
7: Если *C*.[Номер родит элемента] = *A*.[Номер элемента], то:
8: *C*.[Указатель на родит элемент] := *A*
9: Добавить *C* в массив *A*.[Указатели на потомков].
10: Если есть элементы в *A*.[Указатели на потомков], то:
11: *A*.[Тип элемента] := «промежуточный узел»
12: Иначе если Если *A*.[Номер переменной] > 0, то:
13: *A*.[Тип элемента] := «операнд-переменная»
14: Иначе:
15: *A*.[Тип элемента] := «операнд-константа»
16: *A*.[Значение выражения] := *A*.[Значения параметров][0]

Нерекурсивный алгоритм вычисления по подготовленному выражению *B* может быть реализован таким образом:

Алгоритм 3. Вычисление значения выражения *B*

1: Сформировать массив значений *X* независимых переменных, установив значения элементов массива *X* по исходным данным на основании массива *B*.[Переменные].

Окончание алгоритма 3

2: Для каждого элемента *A* из массива *B*.[Элементы выражения], начиная с последнего, выполнить функцию Calculate(*X*):
3: Если *A*.[Тип элемента] = «операнд-константа», то значение элемента уже содержится в *A*.[Значение выражения].
4: Если *A*.[Тип элемента] = «операнд-переменная», то:
5: Если задана функция-преобразователь, то:
6: В массив *Z*[] занести значение *X*[*A*.[Индекс переменной]]
7: *A*.[Значение выражения] := *A*.[Функция элемента](Значения параметров, *Z*)
8: иначе:
9: *A*.[Значение выражения] := *X*[*A*.[Индекс переменной]]
10: Если *A*.[Тип элемента] = «промежуточный узел», то:
11: Для каждого потомка элемента *A* из массива [Указатели на потомков] занести значение выражения в массив *Z*[].
12: *A*.[Значение выражения] := *A*.[Функция элемента] (Значения параметров, *Z*)
13: Значение выражения для 1-го элемента выражения *B* содержит результат расчетов.

6. Представление модели ANFIS в универсальной схеме хранения

Универсальная реляционная схема хранения математических выражений может быть применена для хранения нейронечетких моделей типа ANFIS, если рассмотреть данную модель в виде (7).

Для уменьшения количества уровней в дереве выражения целесообразно введение в состав элементарных функций следующих операторов:

1. Суммирования N переменных Σ.
2. Вычисления произведения N переменных Π.

3. Функция Гаусса (1).

Дерево вычислений для функциональной зависимости (7), реализующей модель типа ANFIS с N правилами и M входными переменными, может быть представлено следующим образом:

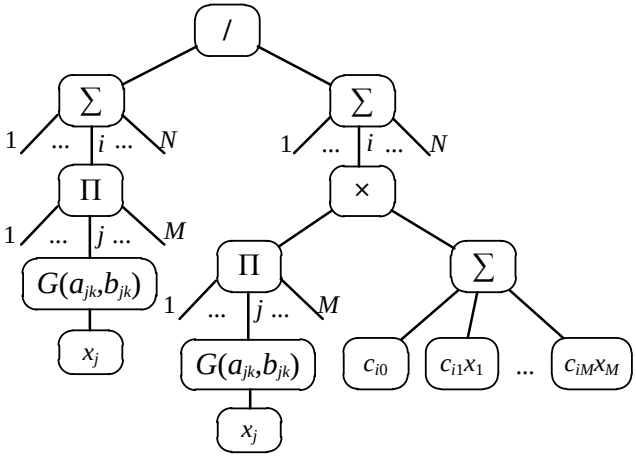

Рис. 7. Структура дерева вычислений для модели ANFIS

Для исключения дублирования вычислений при нормировании (третий слой ANFIS) представляется целесообразным обобщить схему представления, преобразовав ее к ориентированному графу общего вида (рис. 8). Аналогично могут быть исключены повторные вычисления выходов нейронов-фаззификаторов первого слоя, если они подаются на вход нескольких правил (нейронов второго слоя).

Изменение схемы представления не требует модификации алгоритма вычисления выражения, так как в нем гарантируется, что к моменту вычисления элемента слоя i все вычисления в слоях $k > i$ уже проведены, а промежуточные результаты вычислений – известны. Однако требуется внести изменения в схему хранения. На концептуальной модели (рис. 4) связь «Иерархия элементов» должна быть преобразована к типу «многие-ко-многим», что приведет к созданию дополнительного реляционного отношения в логической модели (рис. 5). Соответствую-

щие изменения должны быть внесены и в структуру классов, представленную на рис. 6.

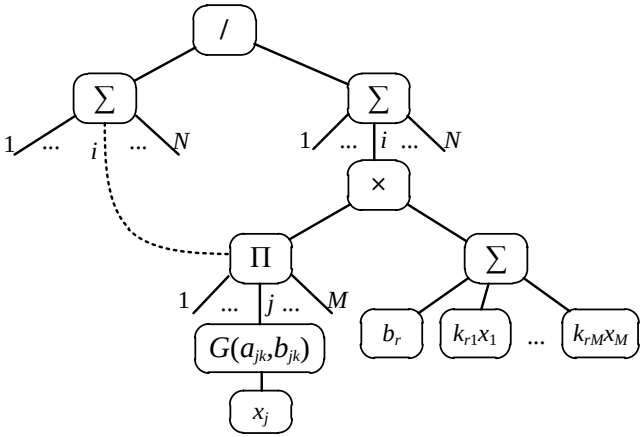

Рис. 8. Обобщенная структура графа вычислений для модели ANFIS

7. Схема данных для хранения ANFIS

Альтернативный вариант хранения модели ANFIS предполагает разработку специфической схемы данных. ER-диаграмма возможного варианта концептуальной модели данных [5] представлена на рис. 9.

Эта схема позволяет хранить нейро-нечеткие модели типа ANFIS для заданной выходной переменной и множества входных переменных. Модель может включать произвольное количество правил и нейронов-фаззификаторов в первом слое. Количество нейронов-фаззификаторов может быть одинаковым или различным для отдельных переменных.

Полагая, что используется гауссова функция принадлежности, в схеме данных для нейронов первого слоя предусмотрено по 2 значения параметров. При использовании другой функции принадлежности количество параметров может быть скорректировано.

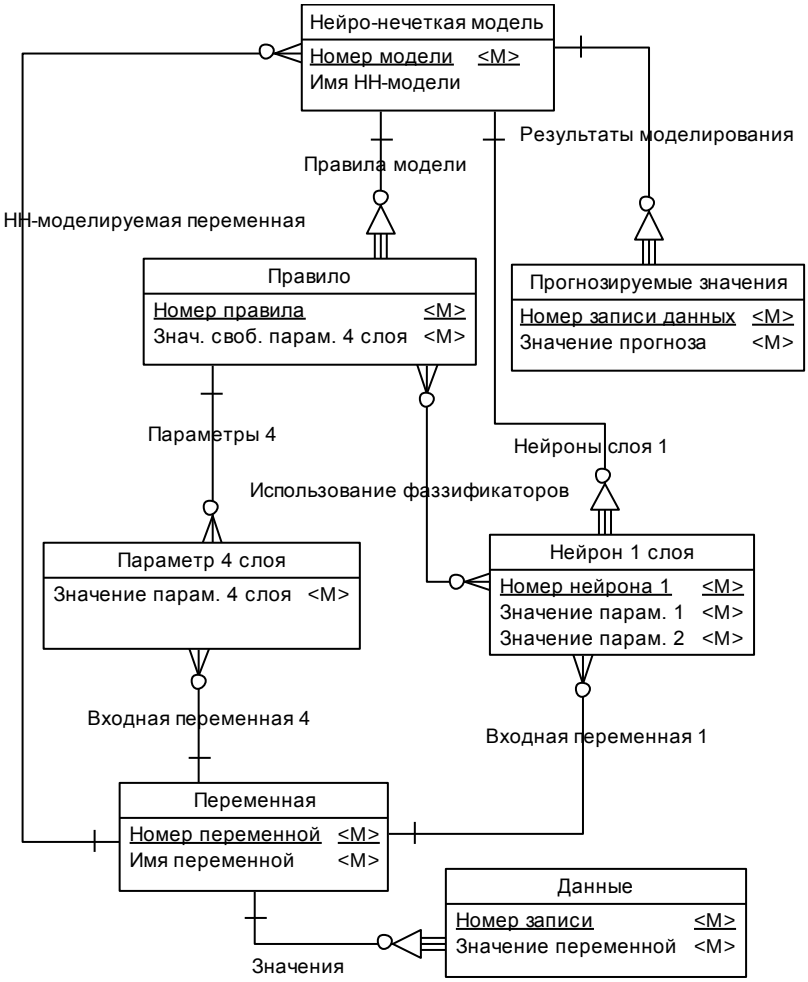

Рис. 9. ER-диаграмма концептуальной модели данных для хранения ANFIS

Диаграмма логической реляционной модели данных, построенная для приведенной концептуальной модели, приведена на рис. 10.

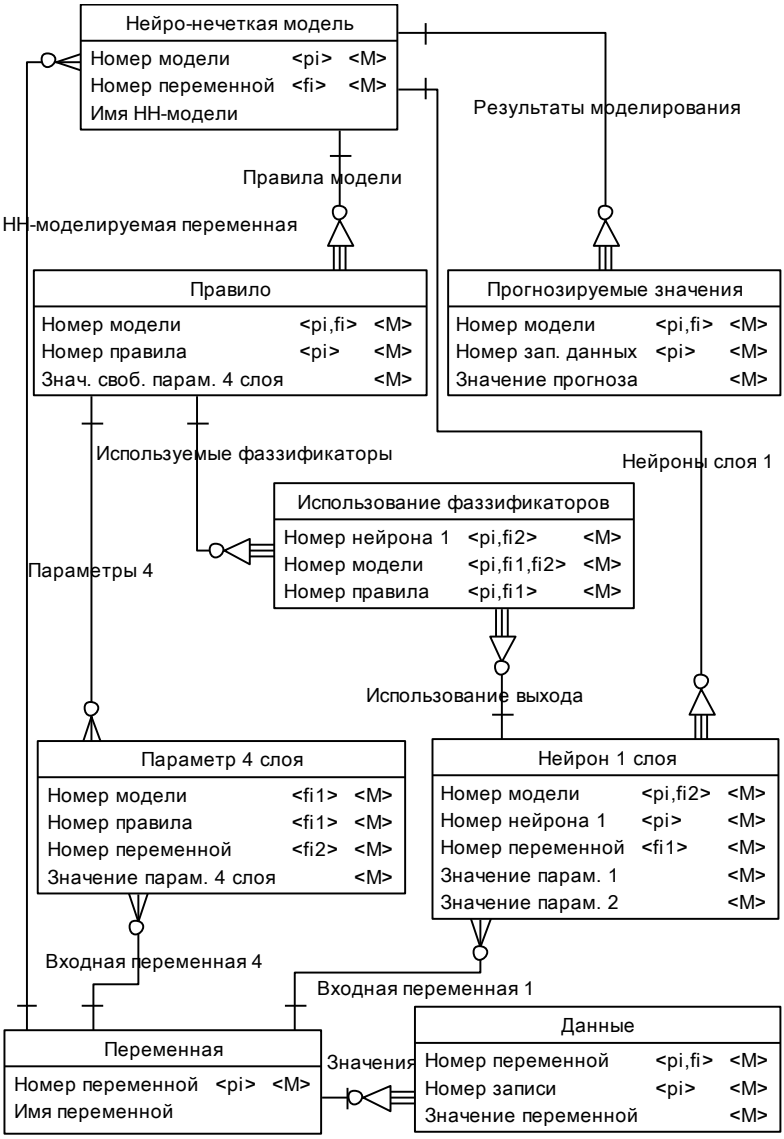

Рис. 10. Диаграмма логической модели данных для хранения ANFIS

Таким образом, хранение нейро-нечеткой модели ANFIS в реляционной БД может быть организовано с использованием следующего набора реляционных отношений (используются соответствующие англоязычные наименования отношений и атрибутов):

VARIABLE (VAR_ID, VAR_NAME)
ANFIS (ANFIS_ID, ANFIS_NAME, VAR_ID)
RULE (ANFIS_ID, RULE_NUM, C0_VAL)
NEURON1 (ANFIS_ID, NEU_NUM, VAR_ID, A_VAL, B_VAL)
RULE_INPUT (ANFIS_ID, RULE_NUM, NEU_NUM)
PARAM4 (ANFIS_ID, RULE_NUM, VAR_ID, C_VAL)
DATA (VAR_ID, REC_NUM, VAR_VAL)
FORECAST (ANFIS_ID, REC_NUM, FORECAST_VAL)

8. Организация вычислений по модели ANFIS

Вычисления по модели ANFIS, представленной в специфической схеме данных (рис. 10), могут быть реализованы в прикладном приложении на языке высокого уровня с предварительной загрузкой из базы данных структуры и параметров модели. Больший интерес представляет, однако, исследование возможности реализации вычисления прогнозируемого значения по модели ANFIS средствами языка SQL. При разработке прикладных программных систем это дает ряд преимуществ, которые будут обсуждаться ниже.

Рассмотрим процесс вычислений поэтапно, в соответствии со слоистой структурой ANFIS. Значения выходов нейронов первого слоя для модели с идентификатором *M* и записи данных с номером *R* могут быть рассчитаны следующим образом:

```
1    T1 = SELECT N.NEU_NUM,
2        EXP(-POW((D.VAR_VAL-N.A_VAL)/N.B_VAL, 2))
                 AS N1_OUT
3    FROM NEURON1 N INNER JOIN VARIABLE V
                              ON N.VAR_ID=V.VAR_ID
4        INNER JOIN DATA D ON V.VAR_ID=D.VAR_ID
```

```
5    WHERE N.ANFIS_ID=M AND D.REC_NUM=R
```

Листинг 2. SELECT-запрос для расчета значений выходов нейронов первого слоя нейро-нечеткой модели

SELECT-запрос для расчета значений выходов нейронов второго слоя для той же модели приведен в листинге 3. На данном этапе для простоты понимания запроса используется предположение, что в языке SQL существует агрегирующая функция MUL для реализации произведения.

SELECT-запросы для расчета значений выходов нейронов слоев с третьего по пятый для той же модели и той же записи данных приведены в листинге 4.

```
1    T2 = SELECT R.RULE_NUM,
2        MUL(T1.NEU1_OUT) AS N2_OUT
3    FROM RULE R INNER JOIN RULE_INPUT RI
4                        ON R.RULE_NUM=RI.RULE_NUM
5        INNER JOIN T1 ON RI.NEU_NUM=T1.NEU_NUM
6    WHERE R.ANFIS_ID=M AND RI.ANFIS_ID=M
7    GROUP BY R.RULE_NUM
```

Листинг 3. Предварительный SELECT-запрос для расчета значений выходов нейронов 2-го слоя нейро-нечеткой модели

```
1    T3 = SELECT T2.RULE_NUM,
2        T2.N2_OUT/(SELECT SUM(T.N2_OUT) FROM T2 T)
                AS N3_OUT
3        FROM T2
4        GROUP BY T2.RULE_NUM

5    T4 = SELECT R.ANFIS_ID, T3.RULE_NUM,
6    T3.N3_OUT*(R.C0_VAL+SUM(P4.C_VAL*D.VAR_VAL))
                AS N4_OUT
7    FROM T3 INNER JOIN RULE R
8                        ON T3.RULE_NUM=R.RULE_NUM
9        INNER JOIN PARAM4 P4
10                       ON T3.RULE_NUM=P4.RULE_NUM
```

```
11      INNER JOIN DATA D ON P4.VAR_ID=D.VAR_ID
12  WHERE R.ANFIS_ID=M AND P4.ANFIS_ID=M
13      AND D.REC_NUM=R

14  T5 = SELECT ANFIS_ID, SUM(N4_OUT)
                AS ANFIS_OUT
15  FROM T4
```

Листинг 4. SELECT-запрос для расчета значений выходов нейронов 3-5-го слоев нейро-нечеткой модели

Для завершения обоснования возможности вычислений по модели ANFIS средствами языка SQL необходимо уточнить этап 2, заменив несуществующую в языке SQL агрегирующую функцию MUL другими конструкциями. Прямая «арифметическая» замена дает:

(10) $MUL(x_i) = EXP(SUM(LN(x_i)))$.

В нашем случае $x_i > 0$, следовательно, такая замена всегда действительна. После замены функции MUL по формуле (10) в запросах для первого и второго слоя выполняются избыточные вычисления значений функций EXP(), LN(), которых можно избежать. В результате этих преобразований имеем:

```
1   T1 = SELECT N.NEU_NUM,
2       -POW((D.VAR_VAL-N.A_VAL)/N.B_VAL,2)
                    AS N1_OUT
3   FROM NEURON1 N INNER JOIN VARIABLE V
4                               ON N.VAR_ID=V.VAR_ID
5       INNER JOIN DATA D ON V.VAR_ID=D.VAR_ID
6   WHERE N.ANFIS_ID=M AND D.REC_NUM=R

7   T2 = SELECT R.RULE_NUM,
8       EXP(SUM(T1.N1_OUT)) AS N2_OUT
9   FROM RULE R INNER JOIN RULE_INPUT RI
10                  ON R.RULE_NUM=RI.RULE_NUM
11      INNER JOIN T1 ON RI.NEU_NUM=T1.NEU_NUM
12  WHERE R.ANFIS_ID=M AND RI.ANFIS_ID=M
```

```
13  GROUP BY R.RULE_NUM
```

Листинг 5. SELECT-запросы для расчета значений выходов нейронов 1-2-го слоев нейро-нечеткой модели

Таким образом, прогнозируемое по модели ANFIS значение может быть вычислено средствами языка SQL без использования процедурных расширений языка. Такой подход представляется оправданным в случае, когда необходимо реализовать вычисление прогнозируемых значений для множества записей данных. Приведенные выше запросы легко обобщить на случай вычислений для множества записей данных (например, последовательность записей с номера R1 по номер R2). Для этого вычисления по первому слою модели модифицируются так:

```
1   T1 = SELECT D.REC_NUM, N.NEU_NUM,
2       -POW((D.VAR_VAL-N.A_VAL)/N.B_VAL,2)
                        AS N1_OUT
3   FROM NEURON1 N INNER JOIN VARIABLE V
4                               ON N.VAR_ID=V.VAR_ID
5       INNER JOIN DATA D ON V.VAR_ID=D.VAR_ID
6   WHERE N.ANFIS_ID=M
7       AND D.REC_NUM>=R1 AND D.REC_NUM<=R2
```

Листинг 6. SELECT-запрос для расчета значений выходов нейронов первого слоя нейро-нечеткой модели в случае множества записей данных

Остальные запросы могут быть модифицированы аналогичным образом. На завершающем этапе (пятый слой) имеем:

```
1   T5 = SELECT ANFIS_ID, REC_NUM,
        SUM(N4_OUT) AS ANFIS_OUT
2   FROM T4
3   GROUP BY REC_NUM
```

Листинг 7. SELECT-запрос для расчета значений выходов нейронов пятого слоя нейро-нечеткой модели в случае множества записей данных

Результаты вычислений могут быть сохранены в таблицу прогнозов для последующей обработки:

```
1    INSERT INTO FORECAST (ANFIS_ID, REC_NUM,
                                   FORECAST_VAL)
2        SELECT ANFIS_ID, T5.REC_NUM, T5.ANFIS_OUT
3        FROM T5
```

Листинг 8. INSERT-запрос для сохранения результатов расчетов по нейро-нечеткой модели

Технологические преимущества вычислений по нейро-нечетким моделям ANFIS средствами СУБД понятны и являются следствием использования здесь централизованной модели вычислений и стандартных средств языка SQL:

1. Вычисления на языке SQL средствами СУБД позволят избавиться от накладных расходов на передачу этих данных в клиентское приложение (по сети или локально, между процессами).

2. Использование стандартных конструкций языка SQL позволяет использовать все возможности СУБД по оптимизации запросов для повышения производительности вычислений.

3. Логика вычислений существует в единственном экземпляре, что гарантирует непротиворечивость ее реализации и облегчает разработку прикладных программных модулей.

4. Упрощается интеграция построенных моделей в системы управления производством и другие корпоративные информационные системы.

Кроме того, организация вычислений на стороне сервера дает некоторые «логические» преимущества при построении систем анализа и управления различными процессами (социальными, техническими и т.п.), основанных на нейро-нечетких моделях:

1. Вычисления в режиме онлайн. Результаты вычисления по моделям совмещаются в БД с исходными данными; они могут быть получены и сохранены в БД непосредственно после поступления соответствующих исходных данных.

2. Готовность результатов моделирования. Исключается необходимость повторного расчета прогнозируемых по модели

значений в различных прикладных модулях, различных контекстах решения задачи и т.п.

3. Целостность результатов моделирования. Стандартными средствами реляционных СУБД (триггерами) может быть обеспечена актуализация результатов моделирования (прогнозируемых значений) при изменении структуры модели или значений ее параметров.

Вычисления в режиме онлайн позволяют, в частности, реализовывать логику обработки данных непосредственно при поступлении, что особенно актуально при работе с большими данными: их анализ постфактум может требовать слишком больших вычислительных ресурсов [9]. Такая возможность может быть востребована, например, в системах анализа сетевого трафика, служебных журналов сетевых сервисов, web-сайтов для обнаружения/предотвращения вторжений за счет выявления нестандартного поведения участников информационного обмена [6, 12].

Основной недостаток такой схемы является классическим для централизованной модели вычислений – производительность схемы определяется сервером БД.

9. Заключение

Представлены подходы к организации хранения нейронечетких моделей типа ANFIS в реляционных базах данных и способы организации вычислений по хранимым моделям. Задача является актуальной как для исследовательских систем моделирования, так и в прикладных системах анализа, прогнозирования и управления техническими, социальными и другими системами. Предложено две схемы хранения: первая – универсальная схема – основана на представлении математического выражения модели ANFIS в виде дерева вычислений; вторая – специфическая схема – построена на основе слоистой структуры модели. Рассмотрены достоинства и недостатки каждой из схем, способы организации вычислений по моделям для этих двух схем. Показано, что при использовании специфической схемы хранения вычисления могут быть организованы стандартными средствами языка SQL, без привлечения процедурных расширений языка. Таким образом

задействуются все средства оптимизации производительности, имеющиеся в СУБД, облегчается интеграция нейро-нечетких моделей в прикладные системы, обеспечивается возможность организации вычислений и принятия решений в режиме онлайн, по мере поступления исходных данных.

Литература

1. БЛЮМИН С.Л., ШУЙКОВА И.А., САРАЕВ П.В. И ДР. *Нечеткая логика: алгебраические основы и приложения*. – Липецк: ЛЭГИ, 2002. – 111 с.
2. ДМИТРИЕВ В.М., ГАНДЖА Т.В. *Алгоритм формирования и вычисления математических выражений методом компонентных цепей* // Математические машины и системы. – 2010. – №3, том 1. – С. 9–21.
3. ДРЕЙПЕР Н., СМИТ Г. *Прикладной регрессионный анализ*. Кн. 1. – М.: Финансы и статистика, 1986. – 366 с.
4. КЛИМЕНКО А.В., СТОЯНОВА О.В., ДЛИ М.И. и др. *Нейро-нечеткий метод построения моделей сложных объектов* // Прикладная информатика. – 2007. – №3(9). – С. 119–127.
5. КОННОЛЛИ Т., БЕГГ К. *Базы данных. Проектирование, реализация и сопровождение. Теория и практика*. – М.: Издательский дом «Вильямс», 2003. – 1440 с.
6. КОРНЕВ П.А., ПЫЛЬКИН А.Н., СВИРИДОВ А.Ю. *Применение инструментария искусственного интеллекта в системах обнаружения вторжений в вычислительные сети* // Современные проблемы науки и образования. – 2014. – №6. – [Электронный ресурс] – URL: www.science-education.ru/120-16906 (дата обращения: 19.05.2015).
7. КРИВЕЦКИЙ И.Ю., ПОПОВ Г.И. *Возможности применения технологии нейро-нечетких сетей в некоторых видах спорта* // Информатика и системы управления. – 2013. – №4(38). – С. 80–87.
8. МИХАЙЛЮК П.П. *Программный комплекс синтеза нейро-нечетких моделей технологических процессов* // Системы управления и информационные технологии. – 2007. – №1.3(27). – С. 365–369.

9. НОВИКОВ Д.А. *Большие данные: от Браге к Ньютону* // Проблемы управления. – 2013. – Вып. 6. – С. 15–23.
10. ПРАТТ Т., ЗЕЛКОВИЦ М. *Языки программирования: разработка и реализация* / Под общей ред. А. Матросова. – СПб.: Питер, 2002. – 688 с.
11. САРАЕВ П.В. *Идентификация нейросетевых моделей*. – Липецк: ЛГТУ, 2011. – 94 с.
12. ЧАСТИКОВА В.А., ВЛАСОВ К.А., КАРТАМЫШЕВ Д.А. *Обнаружение DDOS-атак на основе нейронных сетей с применением метода роя частиц в качестве алгоритма обучения* // Фундаментальные исследования. – 2014. – №8–4. – С. 829–832. – [Электронный ресурс] – URL: www.rae.ru/fs/?section=content&op=show_article&article_id=10003946 (дата обращения: 19.05.2015).
13. FULLER R. *Introduction to Neuro-Fuzzy Systems*. – Berlin/Heildelberg, Springer-Verlag, 2000. – 289 p.
14. JYH-SHING ROGER JANG. *ANFIS: Adaptive-Network-Based Fuzzy Inference* // IEEE Transactions on Systems, Man and Cybernetics. – 1993. – Vol. 23, Issue 3. – P. 665–685.

ORGANIZING THE STORAGE OF NEURO-FUZZY MODEL IN RELATIONAL DATABASES

Vladimir Alexeev, Lipetsk State Technical University, Lipetsk, Cand.Sc. (alexeev48@gmail.com).
Pavel Saraev, Lipetsk State Technical University, Lipetsk, Doctor of Science, assistant professor (psaraev@yandex.ru).
Pavel Domashnev, Lipetsk State Technical University, Lipetsk, Cand.Sc. (pdomashnev@gmail.com).
Oleg Nazarkin, Lipetsk State Technical University, Lipetsk, Cand.Sc. (nazarkino@mail.ru).

Abstract: *The paper considers the organization of the storage of ANFIS neuro-fuzzy models in relational databases. Two storage schemes were developed: the universal schema based on the ANFIS model representation as a function of several variables in a computation tree and the specific schema, which reflects layered model structure. Algorithms for computing ANFIS model output for two variants of the storage are described. The possibility of computing ANFIS model output by standard means of SQL is substantiated.*

Keywords: ANFIS neuro-fuzzy model, computation tree, relational databases, SQL-based calculations.

Статья представлена к публикации членом редакционной коллегии Н.И. Базенков.

Поступила в редакцию 15.06.2015.
Опубликована 31.05.2016.