УДК 004.8

И.А. Ходашинский, И.В. Горбунов, Д.С. Синьков

Алгоритмы генерации структур двухкритериальных Парето-оптимальных нечетких аппроксиматоров

Предложены алгоритмы генерации структуры нечетких аппроксиматоров, оптимизированных по следующим двум критериям: точность и сложность. Приведены результаты исследований полученных аппроксиматоров на реальных данных из репозитория KEEL, проведено сравнение полученных результатов с аналогами.

Ключевые слова: нечеткий аппроксиматор, Парето-оптимальность, генерация структуры, метаэвристики.

Традиционно разработка нечетких систем, основанных на правилах, направлена на оптимизацию показателей качества, например, максимизацию точности в задачах классификации или минимизацию ошибки в задачах аппроксимации. Указанные цели часто достигаются в ущерб понятности проектируемой нечеткой системы (НС). Решение проблемы нахождения компромисса между точностью и понятностью является предметом исследования настоящей работы.

Если в НС каждая входная переменная представлена небольшим числом нечетких термов, каждому терму можно придать осмысленное лингвистическое значение, например: «очень малое», «малое», «среднее», «большое», «очень большое». Кроме того, когда уменьшается число правил и число условий в правилах, интерпретируемость НС улучшается. Отметим также, что несложная и хорошо интерпретируемая НС более проста в настройке, требует меньше памяти и времени вывода, чем более сложная НС.

Не существует универсального способа измерения сложности или интерпретируемости моделей [1–3]. В нашей работе сложность определена как сумма числа правил и числа термов нечеткой системы. Кроме того, на нечеткую систему наложены следующие ограничения:

- число термов, которыми описывается каждая входная переменная, находится в разумных пределах (как правило, от 2 до 9, за исключением НС с одним входом);
- функции принадлежности (ФП) нечетких термов выпуклы и нормализованы, т.е. каждая ФП имеет значение, равное единице по крайней мере в одной точке в области определения;
- область определения полностью покрыта функциями принадлежности, т.е. по крайней мере, одна ФП получает значение, не равное нулю в любой точке области определения;
- ФП различимы, т.е. две ФП не принимают очень близких значений на области определения;
- в работе используются глобально определенные функции принадлежности, это означает, что определенные один раз функции принадлежности используются во всех правилах;
- в базе нет правил, у которых одинаковые антецеденты, но различные консеквенты.

Поскольку точность и сложность являются противоречивыми критериями, генерируется не одна оптимальная НС, а набор таких систем. Из сгенерированного набора выбирается множество недоминируемых решений, обозначенное как Парето-множество, характерным свойством которого является оптимальное соотношение между критериями точности и сложности**.**

Постановка задачи. Нечеткий аппроксиматор задается своей базой правил; i-е правило имеет следующий вид:

$$\text{ЕСЛИ } x_1{=}A_{1i} \text{ И } x_2{=}A_{2i} \text{ И } \ldots \text{ И } x_n{=}A_{ni} \text{ ТО } y = r_i,$$

где A_{ij} – лингвистический терм, которым оценивается переменная x_i; r_i – действительное число, которым оценивается выход y.

Нечеткий аппроксиматор осуществляет отображение $f:\Re^n \to \Re$:

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{R} \mu_{A1i}(x_1)\cdot\mu_{A2i}(x_2)\cdot\ldots\cdot\mu_{Ani}(x_n)\cdot r_i}{\sum_{i=1}^{R} \mu_{A1i}(x_1)\cdot\mu_{A2i}(x_2)\cdot\ldots\cdot\mu_{Ani}(x_n)},$$

где **x** – входной вектор, R – число правил; n – число входных переменных; μ_{Aij} – функция принадлежности, определяемая набором своих параметров, например, треугольная – тремя параметрами, трапециевидная – четырьмя, гауссова и параболическая – двумя.

Нечеткий аппроксиматор может быть представлен как

$$y = f(\mathbf{x}, \boldsymbol{\theta}),$$

где y – скалярный выход аппроксиматора; $\boldsymbol{\theta} = ||\theta_1,\ldots,\theta_N||$ – вектор параметров; $N = \sum_{i=1}^{n} a \times b_i$; a – число. Пусть дано множество обучающих данных (таблица наблюдений) $\{(\mathbf{x}_p; t_p), p = 1 ,..., m\}$, тогда среднеквадратическая функция ошибки ($RMSE$), являющаяся численным критерием адекватности модели, вычисляется по следующей формуле:

$$E(\boldsymbol{\theta}) = \frac{1}{m}\sqrt{\sum_{p=1}^{m}(t_p - f(\mathbf{x}_p, \boldsymbol{\theta}))^2}.$$

Проблема построения Парето-оптимального множества нечетких аппроксиматоров с разными соотношениями между значениями их точности и сложности основано на оптимизации заданной функции в многомерном пространстве, координаты которого соответствуют параметрам **θ** нечеткого аппроксиматора. Схематично технология построения Парето-оптимального множества нечетких аппроксиматоров приведена на рис. 1.

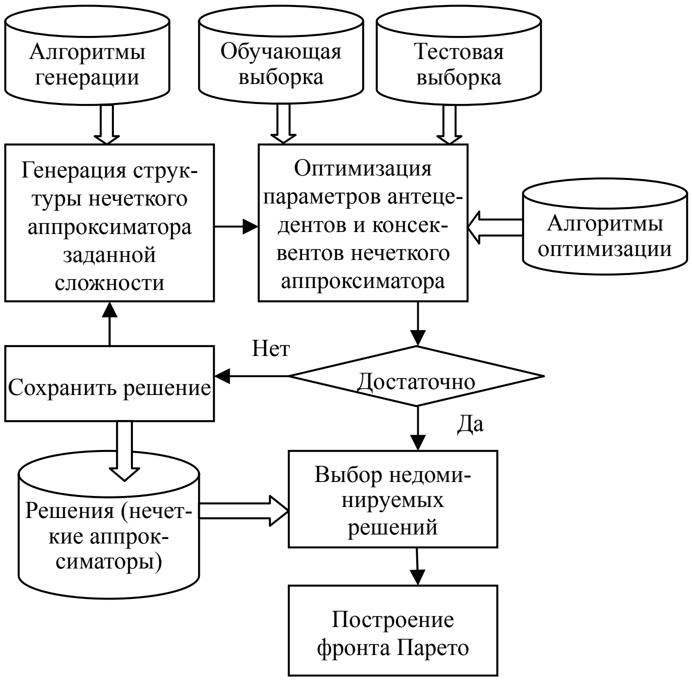

Рис. 1. Схема построения Парето-оптимального множества нечетких аппроксиматоров

Генерация структуры предполагает определение количества нечетких правил и нечетких термов (функций принадлежности). Для этих целей предлагается использовать следующие методы и алгоритмы: 1) модифицированный Gath-Geva метод кластеризации; 2) алгоритм генерации базы правил нечеткой системы равномерным разбиением и перебором; 3) алгоритм генерации базы правил нечеткой системы исключением неэффективных правил; 4) алгоритм генерации базы правил с заданной структурой; 5) алгоритм генерации базы правил делением.

Для оптимизации антецедентов правил в работе используются следующие алгоритмы: 1) классический и модифицированный алгоритмы роящихся частиц [4, 5]; 2) непрерывный и прямой алгоритмы муравьиной колонии [6–9]; 3) алгоритм пчелиной колонии [10, 11]; 4) алгоритм перемещения бактерий [12, 13]. Для оптимизации консеквентов правил применяется метод наименьших квадратов [14].

Оптимизация консеквентов правил в предлагаемой технологии выполняется следующими методами: 1) методом поиска ближайшего соседа в таблице наблюдений; 2) адаптированным методом наименьших квадратов.

Конечный пользователь может выбрать наиболее подходящее решение из окончательного фронта Парето путем поиска компромисса между сложностью и точностью в зависимости от своих предпочтений.

Сильные стороны предлагаемого подхода:

1) пользователь заранее может определить желаемый уровень сложности, все получаемые нечеткие системы будут отвечать данному требованию;

2) поскольку сложность не является оптимизируемым параметром, это позволяет использовать алгоритмы однокритериальной оптимизации.

Недостаток данного подхода заключается в его ориентации только на дискретное описание критерия сложности нечеткой системы.

Алгоритмы генерации структуры нечеткого аппроксиматора

Алгоритм генерации структуры на основе Gath-Geva метода кластеризации

Рассмотрим алгоритм генерации структуры на основе модифицированного Gath-Geva метода кластеризации [15]. Кластеры, полученные применением алгоритма, преобразуются в нечеткие термы, количество таких термов, относящихся к каждой переменной, фиксировано и равно количеству кластеров. Собственно алгоритм приведен ниже.

Вход: Таблица наблюдений $\{\mathbf{x}_p, t_p\}$, количество входных параметров l, количество кластеров c, экспоненциальный вес m, требуемая точность работы алгоритма кластеризации ε.

Выход: $\boldsymbol{\theta}$ – база правил аппроксиматора.

$\mathbf{u}$ – матрица разбиения содержит c – строк, n – колонок, ограничения: $\sum_{i=1}^{c}\mathbf{u}_{ie}=1$, $0<\sum_{e=1}^{n}\mathbf{u}_{ie}<n$;

$\mathbf{v}_i$ – вектор координат центра i-го кластера;

$\mathbf{d}$ – матрица расстояний между кластерами и элементами выборки содержит c – строк, n – колонок;

$\mathbf{A}_i$ – матрица размером $l\times l$ нечеткой ковариации между i-м кластером и выборкой $\mathbf{x}$;

$\mathbf{S}$ – матрица дисперсий элементов таблицы наблюдения относительно центров векторов по каждому из входных параметров. Матрица содержит строк по количеству кластеров c, столбцов по количеству входных параметров l.

Шаг 1. Произвольное заполнение матрицы разбиения $\mathbf{u}$, $\sum_{e=1}^{n}\mathbf{u}_{ie}=1$.

Шаг 2. Расчет векторов центров кластеров $\mathbf{v}_i$, $\mathbf{v}_i=\dfrac{\sum_{e=1}^{n}\mathbf{u}_{ie}^{m}x_e}{\sum_{e=1}^{n}\mathbf{u}_{ie}^{m}}$.

Шаг 3. Вычисление расстояний между кластером и каждым элементом выборки

$$\mathbf{A}_i=\frac{\sum_{e=1}^{n}\mathbf{u}_{ie}{}^{m}(\mathbf{x}_e-\mathbf{v}_j)(x_e-\mathbf{v}_j)^T}{\sum_{e=1}^{n}\mathbf{u}_{ie}{}^{m}},\quad p_i=\frac{1}{p}\left(\sum_{e=1}^{n}\mathbf{u}_{ie}\right),\quad \mathbf{d}_{ie}^2=\frac{\det(\mathbf{A}_i)^{\frac{1}{2}}}{p_i}\exp\left(\frac{1}{2}(\mathbf{v}_i-\mathbf{x}_e)^T(\mathbf{A}_i)^{-1}(\mathbf{v}_i-\mathbf{x}_e)\right).$$

Шаг 4. Расчет новой матрицы разбиения;

Шаг 4.1. **ЕСЛИ** $\mathbf{d}_{ie}=0$ **ТО** $\mathbf{u}^*_{ie}=1$, переход на шаг 5.

Шаг 4.2. **ЕСЛИ** $\mathbf{d}_{ie}<0$ **ТО** $\mathbf{u}^*_{ie}=0$, переход на шаг 5.

Шаг 4.3. **ЕСЛИ** $\mathbf{d}_{ie}>0$ **ТО** $\mathbf{u}^*_{ie}=\left[\sum_{k=1}^{c}\left(\frac{\mathbf{d}_{ie}}{\mathbf{d}_{ke}}\right)^{\frac{2}{m-1}}\right]^{-1}$, переход на шаг 5.

Шаг 5. **ЕСЛИ** $\left\|\mathbf{u}^*-\mathbf{u}\right\|<\varepsilon$ **ТО** переход на *шаг* 7, **ИНАЧЕ** переход на шаг 6.

Шаг 6. $\mathbf{u}=\mathbf{u}^*$ переход на *шаг* 2.

Шаг 7. Расчет размеров проекции кластера для определения параметров ФП

$$S_{ij}=\frac{\sum_{k=1}^{n}(\mathbf{u}_{ki})^m\cdot(x_{kj}-v_{ij})^2}{\sum_{k=1}^{n}(\mathbf{u}_{ki})^m}.$$

Шаг 8. Построение функций принадлежности A_{iq}, соответствующих каждому нечеткому терму:

$$A_{ij}=\mathrm{Gauss}(\mathbf{v}_{ij},\sqrt{\mathbf{S}_{ij}}).$$

Шаг 9. Формирование нечетких правил вида

R_i: **ЕСЛИ** $x_1=A_{i1}$ **И** $x_2=A_{i2}$ **И** $x_3=A_{i3}$ **И** … **И** $x_n=A_{in}$ **ТО** $r_i = none$,

описывающих каждый кластер.

Шаг 10. Инициализация консеквента r_i правила R_i методом поиска ближайшего соседа в таблице наблюдений.

Алгоритм генерации базы правил нечеткой системы равномерным разбиением и перебором

Вход: таблица наблюдений $\{\mathbf{x}_p, t_p\}$, *Type* – тип функции принадлежности, вектор **Count_terms** с указанием количества термов, требуемых по каждому признаку.

Выход: **θ** – начальная база правил аппроксиматора.

Алгоритм:

Шаг 1. Для каждого *i*-го входного параметра **делать:**

Шаг 1.1. Рассчитать длину одного терма как

$distance = \max(x_{pi}) - \min(x_{pi}) / (Count\_terms_i - 1)$.

Шаг 1.2. Задать базовую точку построения терма $term\_base = \min(x_{pi})$.

Шаг 1.3. **Пока** $term\_base <= \max(x_{pi})$.

Шаг 1.3.1. Создание терма A_{iq} типа *Type,* накрывающего интервал

$[term\_base - distance, term\_base + distance]$.

Шаг 1.3.2. $term\_base = term\_base + distance$.

End Пока;

End делать (*i*).

Шаг 2. Формирование базы правил.

Шаг 2.1. Создание правил путем перебора термов каждого с каждым по всем входным параметрам *i*

R_q: **ЕСЛИ** $x_1=A_{1q}$ **И** $x_2=A_{2q}$ **И** $x_3=A_{3q}$ **И** … **И** $x_n=A_{nq}$ **ТО** $\mathrm{r}_q = none$.

Шаг 2.2. Инициализация консеквента r_q правила R_q методом ближайшего соседа;

Шаг 2.3. $\boldsymbol{\theta} := \boldsymbol{\theta} \cup R_q$.

Достоинство алгоритма заключается в простоте его реализации, полученная база правил хорошо поддается оптимизации.

Недостатком алгоритма является подверженность его эффекту «проклятия размерности».

Алгоритм генерации базы правил нечеткой системы исключением неэффективных правил

Вход: таблица наблюдений $\{\mathbf{x}_p, t_p\}$, *Shrink_rule* – количество правил, которые необходимо исключить, *Type* – тип функции принадлежности, **Count_terms** – вектор с указанием количества термов для каждого признака.

Выход: **θ** – усечённая база правил аппроксиматора.

Шаг 1. Построить базу правил **θ** алгоритмом генерации базы правил нечеткой системы равномерным разбиением и перебором ($\{\mathbf{x}_p, t_p\}$, *Type,* **Count_terms**).

Шаг 2. Сгенерировать единичный вектор **Init_rule_struct** размером, равным количеству правил в базе правил **θ**

$Init\_rule\_struct_i=1$, $i=1..|\,\boldsymbol{\theta}.\mathbf{R}|$.

Шаг 3. Заменить *Shrink_rule* единиц в начале вектора **Init_rule_struct** на нули.

Шаг 4. Сгенерировать все различные варианты перестановок с повторением **Case_of_rule_struct** из вектора **Init_rule_struct** методом лексикографического генератора.

Шаг 5. Скопировать **θ** в **θ’**;

Шаг 6. Для всех *j*-х вариантов структуры правил в

Case_of_rule_struct делать:

Шаг 6.1. Для всех *k*-х элементов структуры

Case_of_rule_struct$_j$ **делать:**

Шаг 6.1.1. ЕСЛИ **Case_of_rule_struct**$_{jk}$=0 ТО

$\boldsymbol{\theta}'.\mathbf{R} = \boldsymbol{\theta}'.\mathbf{R} \setminus \boldsymbol{\theta}'.\mathbf{R}_k$;

End делать (*k*).

Шаг 6.2. Для всех *l*-х термов в **θ’ Do:**

Шаг 6.2.1. ЕСЛИ $\boldsymbol{\theta}'.\mathrm{term}_l \notin \boldsymbol{\theta}'.\mathbf{R}_k$. $k=1..|\,\boldsymbol{\theta}'.\mathbf{R}|$ ТО $\boldsymbol{\theta}'.\mathrm{term} = \boldsymbol{\theta}'.\mathrm{term} \setminus \boldsymbol{\theta}'.\mathrm{term}_l$;

End делать (*l*).

Шаг 6.3. **θ’ =**

Метод_геометрической_коррекции_параметров_ФП$(\{\mathbf{x}_p,t_p\}, \boldsymbol{\theta}')$**;**

Шаг 6.4. $\mathbf{Case\_FS}_j = \boldsymbol{\theta}'$;

Шаг 6.5. Рассчитать ошибку для каждого варианта базы правил **Case_FS**$_j$

Error_of_FS$_j$= *RMSE*({$\mathbf{x}_p$,t_p}, **Case_FS**$_j$));

End делать (*j*).

Шаг 7. Вернуть такой **Case_FS**$_p$ что $p = \arg\min_p(\mathbf{Error\_of\_FS}_p)$.

Алгоритм формирует нечеткие системы с малой ошибкой при невысокой сложности, но при больших сложностях системы алгоритм практически бесполезен.

Алгоритм генерации базы правил с заданной структурой

Вход: таблица наблюдений {$\mathbf{x}_p$, t_p}, *Type* – тип функции принадлежности, **Count_terms** – вектор с указанием количества термов для каждого признака.

Выход: **θ** – начальная база правил аппроксиматора**.**

Шаг 1. Отсортировать **Count_terms** в лексикографическом порядке по возрастанию.

Шаг 2. Сгенерировать все различные варианты перестановок с повторением **Case_of_struct** из вектора **Count_terms** методом лексикографического генератора;

Шаг 3. Для *j*-го варианта перестановки **Case_of_struct делать:**

Шаг 3.1. Сгенерировать варианты начальной базы правил

Case_of_FS*j* = *алгоритм генерации базы правил нечеткой системы перебором*({$\mathbf{x}_p$,t_p}; *Type;* **Case_of_struct**$_j$).

Шаг 3.2. **Error_of_FS**$_j$=*RMSE* ({$\mathbf{x}_p$,t_p }, **Case_of_FS**j)).

End делать (*j*);

Шаг 4. Вернуть такой **Case_of_FS**$_p$,что $p = \arg\min_p(\mathbf{Error\_of\_FS}_p)$.

Алгоритм генерирует аппроксиматор с ошибкой, сравнимой с алгоритмом генерации базы правил нечеткой системы перебором, при меньшем количестве нечетких термов и правил в базе. В меньшей степени, по сравнению с алгоритмом генерации базы правил нечеткой системы перебором, но подвержен влиянию эффекта «проклятия размерности».

Алгоритм генерации базы правил делением

Суть алгоритма заключается в следующем. На начальном этапе генерируется терм, накрывающий все пространство определения входной переменной, и формируются правила из этого набора термов. Данный шаг позволяет отказаться от методов коррекции базы правил, потому что не менее одного правила будет использовано для работы аппроксиматора. Далее используется алгоритм деления пополам. Этот этап повторяется, пока не будет сгенерировано нужное количество правил. Рекурсивное описание алгоритма представлено ниже.

Вход: таблица наблюдений {$\mathbf{x}_p$, t_p}, **min_input** – вектор минимумов входных параметров, **max_input** – вектор максимумов входных параметров**,** *Type* – тип функции принадлежности, база правил аппроксиматора **θ,** требуемое количество правил *need_rules*.

Выход: **θ** – база правил аппроксиматора.

count_rules – счетчик правил в базе правил **θ**.

Шаг 1. **ЕСЛИ** *count_rules* <= *need_rules*, **ТО** шаг 2, **ИНАЧЕ Выход**.

Шаг 2. Для каждого *i*-го входного параметра создать терм A_{iq} типа *Type,* накрывающего интервал [**min_input**$_i$, **max_input**$_i$].

Шаг 3. Создать правило

R_q: **ЕСЛИ** $x_1 = A_{1q}$ **И** $x_2 = A_{2q}$ **И** $x_3 = A_{3q}$ **И** … **И** $x_n = A_{nq}$ **ТО** r_q = *none.*

Шаг 4. Инициализация консеквента r_q правила R_q *методом_ближайшего_соседа*({$\mathbf{x}_p$,t_p}, R_q).

Шаг 5. $\boldsymbol{\theta} := \boldsymbol{\theta} \cup \{R_q\}$.

Шаг 6. Задать вектор **center_input**, таким образом, что его элементы равны среднему арифметическому соответствующих элементов **min_input** и **max_input**

center_input$_i$= (**min_input**$_i$ + **max_input**$_i$)/2, *i*=1..|**min_input**|.

Шаг 7. Вызов алгоритма_генерации_базы_правил_делением(**min_input; center_input**; Type; **θ**; need_rules).

Шаг 8. Вызов алгоритма_генерации_базы_правил_делением(**center_input; max_input**; Type; **θ**; need_rules).

Сравнение с аналогами. Были проведены исследования различных вариантов использования алгоритмов генерации и оптимизации. Каждый из алгоритмов генерации работал с алгоритмом оптимизации. Исследование алгоритмов проводилось при решении задач аппроксимации идеальных данных и данных, описывающих реальные процессы, представленных в репозитории KEEL (Knowledge Extraction Evolutionary Learning, http://www.keel.es). Характеристики данных представлены в табл. 1. Все входные и выходные переменные – вещественные числа. Каждая выборка разделена на пять наборов, из которых строится обучающая и тестовая выборки, содержащие 80 и 20% данных соответственно. Разделение проводилось таким образом, чтобы каждый набор попал во все тестовые выборки ровно один раз.

Таблица 1

Описание данных

Название данных	Количество образцов	Количество входных переменных	Краткое описание
Diabetes	43	2	Прогнозирование развития сахарного диабета у инсулинозависимых детей
ELE2	1066	4	Проблема оценки стоимости обслуживания городских электрических сетей

В работах [17, 18] ошибка представлена как MSE/2, поэтому в таблице сравнений оригинальное значение умножено на 2. Из таблицы видно, что оптимизация нашими алгоритмами дает лучшие результаты на всех выборках, кроме тестовой выборки по набору данных Diabetes.

Заключение. Интерпретируемость и транспорентность нечетких систем являются их важнейшим преимуществом в интерактивных приложениях. Для обеспечения этих свойств в работе предложены алгоритмы генерации, накладывающие определенные ограничения на проектируемые нечеткие системы. Алгоритмы позволяют находит компромисс между сложностью, выраженной через число нечетких правил и нечетких термов, и точностью модели. Сравнительный анализ разработанных алгоритмов с аналогами показал их высокую эффективность.

Таблица 2

Сравнение эффективности алгоритмов

Алгоритм	Правила		Обучающая выборка		Тестовая выборка	
	Число	СКО	*MSE*	СКО	*MSE*	СКО
Diabetes						
Наши алгоритмы	25	0	**0,02502**	0,0116	0,5963	0,1918
Wang–Mendel [16]	18,6	1,4	0,2284	0,0425	1,4024	0,6890
COR-BWAS [16]	18,6	1,4	0,1750	0,0250	1,4587	0,7091
Thrift [16]	46,2	0,7	0,0745	0,0098	0,8783	0,3575
Pittsburgh [16]	15	2,9	0,1040	0,0182	0,9509	0,7881
Fuzzy-GAP [16]	10	0	0,1429	0,0376	0,5014	0,3014
Pitts-DNF min [16]	**1,6**	0,5	0,4162	0,1231	0,4539	0,1288
Pitts-DNF med [16]	5,4	0,5	0,1296	0,0136	**0,3213**	0,1922
Pitts-DNF max [16]	9,6	1,2	0,1066	0,0150	0,6340	0,5276
ELE 2						
Наши алгоритмы	37,3	12,3	**13510**	2611	**17235**	2420
Wang–Mendel [16]	65	0	112270	1498	112718	4685
COR-BWAS [16]	65	0	102664	1080	102740	4321
Thrift [16]	524,6	6,4	146305	12991	168472	20135
Pittsburgh [16]	240	21,1	210717	32027	265130	30161
Fuzzy-GAP [16]	33	0	279166	90017	290062	89155
Pitts-DNF min [16]	**12,2**	0,7	202943	43684	212018	44616
Pitts-DNF med [16]	18,6	1,4	86930	3955	99310	12996
Pitts-DNF max [16]	32,4	6,6	70207	1658	88017	8968
DynMO GFS [17]	25	–	18732	–	20858	–
NSGA–II_RB [18]	29	–	34230	–	39668	–
NSGA–II_KB[18]	29	–	26272	–	31174	–
PAES_RB [18]	30	–	15453	–	30906	–
PAES_KB [18]	30	–	22086	–	25212	–

Работа выполнена при финансовой поддержке РФФИ (проект №12-07-00055), РГНФ (проект №12-06-12008) и в соответствии с Госзаданием 7.701.2011.

Литература

1. Gacto M.J. Interpretability of linguistic fuzzy rule–based systems: An overview of interpretability measures / M.J. Gacto, R. Alcala, F. Herrera // Information Sciences. – 2011. – Vol. 181. – P. 4340–4360.

2. Cordon O. A historical review of evolutionary learning methods for Mamdani–type fuzzy rule–based systems: Designing interpretable genetic fuzzy systems // International Journal of Approximate Reasoning. – 2011. – Vol. 52. – P. 894–913.

3. Interpretability assessment of fuzzy knowledge bases: A cointension based approach / C. Mencar, C. Castiello, R. Cannone, A.M. Fanelli // International Journal of Approximate Reasoning. – 2011. – Vol. 52. – P. 501–518.

4. Ходашинский И.А. Идентификация параметров нечетких моделей типа синглтон на основе алгоритма роящихся частиц // Информационные технологии. – 2009. – № 6. – С. 8–11.

5. Ходашинский И.А. Идентификация параметров нечетких систем на основе адаптивного алгоритма роящихся частиц / И.А. Ходашинский, Д.С. Синьков // Информационные технологии. – 2011. – № 8. – С. 2–5.

6. Ходашинский И.А., Дудин П.А. Параметрическая идентификация нечетких моделей на основе гибридного алгоритма муравьиной колонии // Автометрия. – 2008. – Том 44, № 5. – С. 24–35.

7. Ходашинский И.А. Идентификация нечетких систем на основе прямого алгоритма муравьиной колонии / И.А. Ходашинский, П.А. Дудин // Искусственный интеллект и принятие решений. – 2011. – № 3. – С. 26–33.

8. Ходашинский И.А. Идентификация нечетких систем на основе непрерывного алгоритма муравьиной колонии / И.А. Ходашинский, П.А. Дудин // Автометрия. – 2012. – Т. 48, № 1. – С. 63–71.

9. Khodashinskii I.A. Identification of fuzzy systems using a continuous ant colony algorithm / I.A. Khodashinskii, P.A. Dudin // Optoelectronics, Instrumentation and Data Processing. – 2012. – Vol. 48, № 1. – P. 54–61.

10. Ходашинский И.А. Алгоритмы муравьиной и пчелиной колонии для обучения нечетких систем / И.А. Ходашинский, И.В. Горбунов, П.А. Дудин // Доклады Том. гос. ун-та систем управления и радиоэлектроники. – 2009. – № 2 (20). – С. 157–161.

11. Ходашинский И.А. Оптимизация параметров нечетких систем на основе модифицированного алгоритма пчелиной колонии / И.А. Ходашинский, И.В. Горбунов // Мехатроника, автоматизация, управление. – 2012. – №10. – С. 15–20.

12. Ходашинский И.А. Построение нечетких аппроксиматоров на основе метода перемещения бактерий / И.А. Ходашинский, Н.Н. Земцов, Р.В. Мещеряков // Известия высших учебных заведений. Физика. – 2012. – Т. 55, № 3. – С. 57–61.

13. Hodashinskii I.A. Construction of fuzzy approximators based on the bacterial foraging method / I.A. Hodashinskii, N.N. Zemtsov, R.V. Meshcheryakov // Russian Physics Journal. – 2012. – Vol. 55, № 3. – P. 301–305.

14. Ходашинский И.А. Идентификация нечетких систем на базе алгоритма имитации отжига и методов, основанных на производных // Информационные технологии. – 2012. – № 3. – С. 14–20.

15. Gath I. Unsupervised optimal fuzzy clustering / I. Gath, A.B. Geva // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1989. – Vol. 7. – P. 773–781.

16. Casillas J. Learning consistent, complete and compact sets of fuzzy rules in conjunctive normal form for regression problems / J. Casillas, P. Martınez, A.D. Benıtez // Soft Computing. – 2009. – Vol. 13. – P. 451–465.

17. Pulkkinen P. Dynamically Constrained Multiobjective Genetic Fuzzy System for Regression Problems / P. Pulkkinen, H.A. Koivisto // IEEE Transactions on Fuzzy Systems. – 2010. – Vol. 18. – P. 161–177.

18. A Multiobjective Evolutionary Approach to Concurrently Learn Rule and Data Bases of Linguistic Fuzzy–Rule–Based Systems / R. Alcala, P. Ducange, F. Herrera et al. // IEEE Transactions on Fuzzy Systems. – 2009. – Vol. 17. – P. 1106–1122.

Ходашинский Илья Александрович
Д-р техн. наук, профессор каф. комплексной информационной безопасности
электронно-вычислительных систем (КИБЭВС) ТУСУРа
Тел.: 8 (382-2) 41-34-26
Эл. почта: hodashn@rambler.ru

Горбунов Иван Викторович
Аспирант каф. КИБЭВС ТУСУРа
Тел.: 8 (382-2) 41-34-26
Эл. почта: noby.Ardor@gmail.com

Синьков Дмитрий Сергеевич
Аспирант каф. КИБЭВС ТУСУРа
Тел.: 8 (382-2) 41-34-26
Эл. почта: express@sibmail.com

Hodashinsky I.A., Gorbunov I.V., Sinkov D.S.
Algorithms for generating structures of two-criterion Pareto optimal fuzzy approximators

This paper proposes an approach, enabling one to build Pareto optimal fuzzy approximators. We present algorithm generation of fuzzy approximator structure. Simulation results confirmed that algorithms are a useful tool for tuning fuzzy systems to achieve better performance.
Keywords: fuzzy approximators, Pareto optimality, generating structure, metaheuristics.