УДК 519.87

Вестник СибГАУ
Т. 16, № 4. С. 842–848

МНОГОАГЕНТНЫЙ АЛГОРИТМ ПРОЕКТИРОВАНИЯ БАЗ НЕЧЕТКИХ ПРАВИЛ ДЛЯ ЗАДАЧ КЛАССИФИКАЦИИ

В. В. Становов*, С. С. Бежитский, Е. А. Бежитская, Е. А. Попов

Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, 660037, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31
*E-mail: vladimirstanovov@yandex.ru

Представлен многоагентный подход к организации коллектива оптимизаторов, основанный на встречах между агентами. В ходе оптимизационного процесса агенты обмениваются лучшими решениями, при этом лучшие агенты получают больше ресурсов в форме встреч. Среди агентов выбраны шесть генетических алгоритмов с различными операторами и три различных алгоритма роя частиц. Предложенный метод коллективного решения оптимизационных задач применен для формирования нечеткой базы правил. Нечеткая база правил состояла из фиксированного числа правил, для каждой переменной каждого правила функция принадлежности задавалась с помощью двух сигмоидальных функций. При этом кодируемыми параметрами были точки достижения сигмоидами 0 и 1, так что задача построения базы нечетких правил сводилась к задаче вещественной оптимизации. Число вещественных параметров при этом зависело от размерности классификационной задачи. Эффективность алгоритма сравнивалась с самонастраивающимся генетическим алгоритмом, решающим ту же задачу формирования нечеткой базы правил. При этом качество классификации оценивалось по величине точности, выборка разбивалась на обучающую и тестовую в соотношении 70 на 30. В качестве классификационных задач выбраны шесть задач с репозиторием KEEL и UCI, в их числе задачи кредитного скоринга, медицинской диагностики, распознавания банкнот и форм зёрен. Для сравнения также были выбраны два других метода классификации, в частности, машины опорных векторов (SVM) и ещё один метод формирования нечетких систем, в котором кодировались номера нечетких термов. По результатам тестирования можно отметить, что многоагентный алгоритм показал эффективность, сравнимую с другими методами при решении сложных оптимизационных задач.

Ключевые слова: эволюционные алгоритмы, алгоритмы роя частиц, нечеткая логика, машинное обучение, генетические нечеткие системы, классификация.

Vestnik SibGAU
Vol. 16, No. 4, P. 842–848

MULTIAGENT ALGORITHM FOR FUZZY RULE BASES DESIGN FOR CLASSIFICATION PROBLEM

V. V. Stanovov*, S. S. Bezhitskii, E. A. Bezhitskaya, E. A. Popov

Reshetnev Siberian State Aerospace University
31, Krasnoyarskiy Rabochiy Av., Krasnoyarsk, 660037, Russian Federation
*E-mail: vladimirstanovov@yandex.ru

In this article a multiagent approach for organizing an ensemble of optimizers, based on the meetings between algorithms is presented. During the optimization process the agents exchange with best solutions and better algorithms receive more resources in the form of meetings. Among agents the six genetic algorithms with different operators and three particle swarm optimizers have been selected. The proposed approach of ensemble-based optimization problems solving is applied to the problem of designing a fuzzy rule base. The fuzzy rule base consisted of a fixed number of rules, for every variable and every rule the membership function was defined with two sigmoidal functions. The encoded parameters were the points where sigmoids reached 0 and 1, so that the problem of designing a fuzzy rule base reduced to a real-valued optimization problem. The number of real-valued parameters depended on the dimension of the classification problem. The effectiveness of the algorithm was compared to the self-configured genetic algorithm, solving the same problem of designing a fuzzy rule base. The classification quality was estimated using the accuracy values; the sample was split with 70 and 30 ratio. As classification problems, six problems have been selected from KEEL and UCI repositories, including credit scoring problems, medical diagnostics problems, banknote recognition and seeds' forms. Two more classification methods have been selected for comparison, more precisely, support vector machines (SVM) and another fuzzy classification method, in which the term numbers were encoded. According to the testing results,

it should be mentioned that the multiagent algorithm has shown the effectiveness, comparable to other method when solving complex optimization problems.

Keywords: evolutionary algorithms, particle swarm optimization, fuzzy logic, machine learning, genetic fuzzy systems, classification.

Введение. На сегодняшний день одним из популярных методов машинного обучения являются системы на нечеткой логике. Однако формирование таких систем связано с решением сложных оптимизационных задач, характеризующихся большим числом переменных, а также наличием переменных различных типов, т. е. вещественных, целочисленных и бинарных. По этой причине для решения подобных оптимизационных задач используются эволюционные алгоритмы, в частности, генетический алгоритм (ГА) и его модификации.

Для формирования нечеткой базы правил необходимо определить нечеткие термы, т. е. положения нечетких множеств и их количество, номера классов, соответствующие каждому правилу, количество правил, а также номера нечетких термов, соответствующих каждой переменной в каждом правиле. В данной работе предлагается сведение всех приведенных задач к задачам вещественной оптимизации на единичном гиперкубе.

Такое представление задачи формирования базы правил позволяет использовать любые оптимизационные процедуры, способные решать задачи вещественной оптимизации. В частности, в данной работе для оптимизации был использован многоагентный алгоритм, комбинирующий несколько генетических алгоритмов и алгоритмов роя частиц (PSO). Многоагентный алгоритм основан на идее встреч между подпопуляциями, в ходе которых они могут обмениваться лучшими решениями. По окончании встреч агенты работают некоторое время независимо, пытаясь улучшить решения. Те агенты, чей ресурс (количество доступных встреч) заканчивается, выбывают из оптимизационной процедуры.

Многоагентый алгоритм оптимизации был сравнен по эффективности с самоконфигурируемым генетическим алгоритмом, который ранее использовался для решения данной задачи.

Остальная часть статьи организована следующим образом: «Теоретический анализ» содержит описание методики представления нечетких баз правил и формулировку задачи опимизации, а также описание самоконфигурируемого ГА и многоагентного алгоритма; «Методология» содержит описание методологии реализации и тестирования алгоритмов; «Экспериментальная часть» содержит список задач и результаты работы алгоритмов, а также их анализ; в заключение приведен список использованной литературы.

Теоретический анализ. Классификация объектов состоит в приписывании объекту F-мерного пространства R^F некоторого класса C_j из заранее заданного множества $C = \{C_1, ..., C_k\}$. Пусть $X = \{X_1, ..., X_F\}$ – набор входных переменных задачи, а U_f, $f = 1, ..., F$, – область определения f-й переменной. Для каждой переменной каждого правила задается нечеткое множество $A_{f,m}$, $f = 1, ..., F$; $m = 1, ..., M$, где M – число правил.

Нечеткое правило R_m, $m = 1, ..., M$, обычно имеет вид

$$R_m : \text{ĹŃËČ } X_1 \text{ ýňî } A_{1,m} \text{ č } ... \text{ č } X_F \text{ ýňî } A_{F,m} \text{ ŇÎ } Y \text{ ýňî } C_m,$$

где Y – выход, $C_m \in \boldsymbol{C}$ – номер класса для m-го правила, а $j_{m,f} \in [1, T_f]$ – это номер нечеткого множества из разбиения P_f, выбранный для переменной X_f.

Зададим для каждой переменной в каждом правиле базы по две функции, описывающие степени принадлежности сигмоидального типа. Зададим сигмоидальную функцию таким образом, чтобы она имела 3 аргумента x, y и z. Здесь x – текущее значение переменной, y – положение на оси X, в котором функция принимает нулевое значение, z – положение на оси X, в котором функция принимает единичное значение. Концы функции округлены до 0 и 1 после y и z. Функции представляются формулой

$$s(x,y,z) = \begin{cases} 1, \text{ Íńëč } (x \geq y \text{ č } y \geq z) \text{ čëč } (x \leq y \text{ č } z > y), \\ 0, \text{ Íńëč } (x \leq z \text{ č } y \geq z) \text{ čëč } (x \geq y \text{ č } z > y), \\ \dfrac{1}{1 + e^{\frac{-c}{y-z}(x + \frac{z-y}{2} - \min(y,z))}}, \text{ Íńëč } y > x > z, \\ \dfrac{1}{1 + e^{\frac{-c}{y-z}(x - \frac{z-y}{2} - \min(y,z))}}, \text{ Íńëč } z > x > y, \end{cases}$$

$$\text{Íńëč } y = z, \text{ ňî } y = y + \frac{k}{c};\ z = z - \frac{k}{c}.$$

Константы имеют значения $c = 15$, $k = 0{,}1$. Пример двух таких функций представлен на рис. 1.

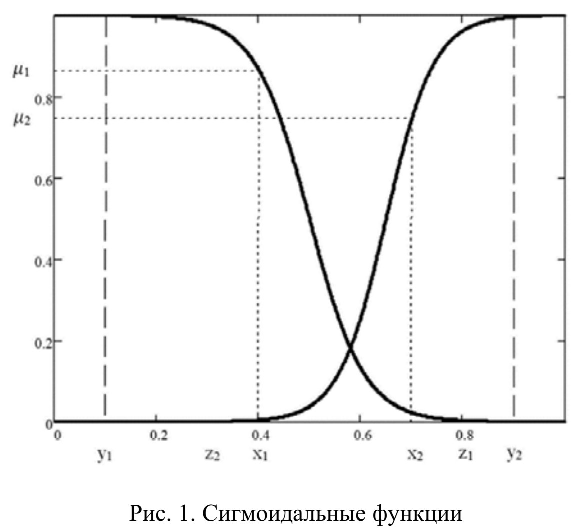

Рис. 1. Сигмоидальные функции

Функция была модифицирована для того, чтобы избежать необходимости решения задачи условной

оптимизации, так как значения y и z лежат в области определения переменной и могут быть использованы напрямую в алгоритме оптимизации.

Пара функций для каждой из переменных кодирует единственный нечеткий терм для конкретной переменной конкретного правила. Однако различные комбинации положений сигмоидальных функций позволяют получать всевозможные формы нечетких термов, обычно недостижимые при использовании других алгоритмов. При этом в процедуре нечеткого вывода используется минимум из значений двух функций принадлежности. Примеры возможных вариантов расположения двух сигмоидальных функций и соответствующие формы нечетких множеств (выделены жирным) представлены на рис. 2.

Как можно видеть, комбинация двух сигмоид позволяет получать классические формы нечетких термов, как, например, варианты 3, 4, 6, так и такие редко используемые формы, как 1 или 5.

Вариант 2 ($\mu = \min(s(x_1, y_1, z_1),\ s(x_2, y_2, z_2)) = 0$ для всех x) представляет особый интерес, так как в этом случае при любом значении переменной процедура нечеткого вывода будет получать нулевое значение функции принадлежности. Этот вариант в алгоритме рассматривается как игнорирование значений переменной, и в этом случае возвращаемое значение степени принадлежности равно единице.

В данном алгоритме также используется определение номера класса по принципу правила-победителя. В общем виде эту процедуру можно записать следующим образом:

$$R = \arg\max_{i \in M} \left\{ \min_{j \in F} \left\{ \min(s(X_j, y_{i,j}^1, z_{i,j}^1), s(X_j, y_{i,j}^2, z_{i,j}^2)) \right\} \right\},$$

где M – число правил; F – число переменных в задаче.

Для задания нечеткой базы правил кодируются значения y^1, z^1, y^2, z^2 для каждой переменой каждого терма, таким образом, общее число переменных равно $4 \cdot N \cdot M$. Стоит отметить, что в этом подходе не используется некоторый общий набор функций принадлежности, каждое правило получает свою комбинацию сигмоид. Номер класса кодируется отдельно для каждого правила, так что итоговое число переменных в задаче равно $4 \cdot N \cdot M + M$. Для кодирования номера правила используется округление в сторону ближайшего номера класса. К примеру, для 3 классов диапазон изменения переменной, кодирующей класс, задается равным [0,1], при этом значения от 0 до 0,333 означают первый класс, от 0,333 до 0,666 – второй, от 0,666 до 1 – третий. Подробнее о методах комбинирования генетических алгоритмов и нечетких систем можно узнать в [1].

При использовании генетического алгоритма для оптимизации нечеткой базы правил использовался метод самонастройки, описанный в [2; 3]. Сам генетический алгоритм использовал три типа селекции (пропорциональная, ранговая, турнирная с размером турнира 2), три типа скрещивания (одноточечное, двухточечное, равномерное) и три типа мутации (слабая, средняя, сильная). Ниже приведено краткое описание метода самонастройки.

Примененный метод самонастройки основывается на поощрении того оператора, который доставил наибольшую суммарную пригодность на данном поколении. Пусть z – число различных операторов i-го типа. Начальные значения вероятностей устанавливаются $p_i = 1/z$. Оценка эффективности каждого оператора каждого типа производится на основании усредненных значений пригодности:

$$AvgFit_i = \frac{\sum_{j=1}^{n_i} f_{ij}}{\sum_{j=1}^{n_i} 1}, i = 1,\ 2,\ ...,\ z,$$

где n_i – количество потомков, в формировании которых принял участие i-й тип оператора; f_{ij} – пригодность j-го потомка, построенного с помощью i-го оператора; $AvgFit_i$ – средняя пригодность решений, построенных при помощи i-го оператора.

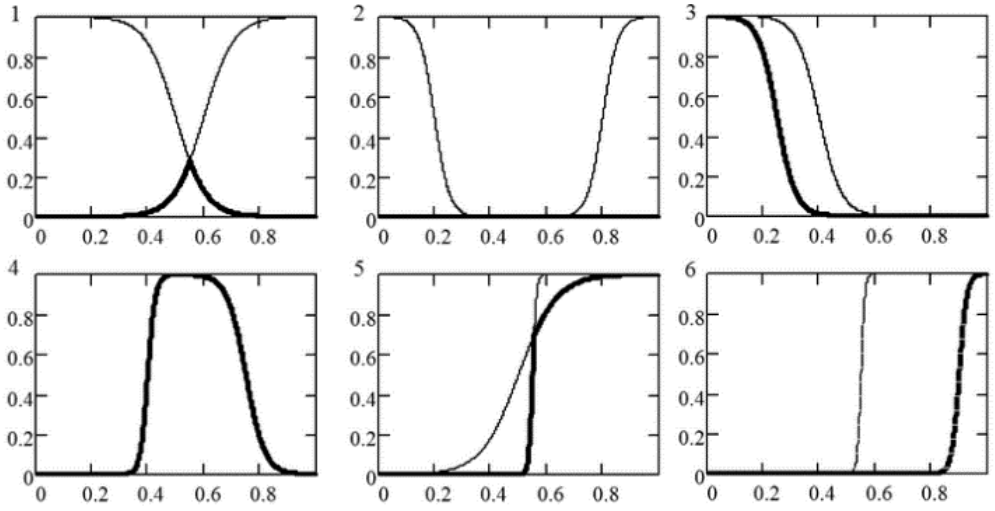

Рис. 2. Примеры возможных форм нечетких термов, получаемых при комбинации сигмоидальных функций

После этого вероятность применения оператора, чье значение $AvgFit_i$ является наибольшим среди всех операторов такого типа, увеличивается на $(z \cdot K – K)/(z \cdot N)$, а вероятности применения остальных операторов уменьшаются на $K/(z \cdot N)$, где N – заданное максимальное число поколений генетического алгоритма; K – константа, её значение устанавливалось равным 0,5.

Многоагентный алгоритм, использованный в данной работе, был реализован с учетом идей, описанных в [4] и [5]. Агентами в многоагентном алгоритме являются несколько генетических алгоритмов [6] и алгоритмов PSO.

Группа генетических агентов:

1) GA1 – турнирная селекция, побитовое скрещивание, сильная мутация;

2) GA2 – турнирная селекция, равномерное скрещивание, средняя мутация;

3) GA3 – пропорциональная селекция, двухточечное скрещивание, сильная мутация;

4) GA4 – пропорциональная селекция, побитовое скрещивание, сильная мутация;

5) GA5 – ранговая селекция, равномерное скрещивание, слабая мутация;

6) GA6 – ранговая селекция, побитовое скрещивание, сильная мутация.

Группа агентов PSO состоит:

– PSO1 – канонический PSO (c1 = c2 = 1,49, w = 0,7298) [7];

– PSO2 – FIPS [8];

– PSO3 – модифицированный канонический PSO (c1 = c2 = 1,49, w_i = random(0;1)), в котором инерция скорости – случайное индивидуальное для каждого агента число в диапазоне от 0 до 1.

Параметры агентов были одинаковыми во всех рассматриваемых задачах поисковой оптимизации. Алгоритм останавливал работу, как только достигался предел числа вычислений целевой функций, установленный для каждой задачи.

Ключевыми характеристиками агентов являются взаимодействие, время жизни и борьба за ресурсы. Поясним идею функционирования агентов. Ресурсами являются количество контактов или встреч (взаимодействия) агентов. Число встреч задается пользователем многоагентной системы. Каждая встреча представляет собой обмен опытом между агентами. Агенты встречаются парами. Пары формируются случайным образом. В каждой паре обмен опытом заключается в передаче менее удачному агенту лучшего решения от более удачного агента. Удачность агента характеризуется наличием у него лучшего решения, чем у оппонента при встрече. Следовательно, по результату встречи менее удачливый агент получает еще более лучшее решение, чем у него имеющиеся, но в обмен на решение вынужден поделиться имеющимся ресурсом, т. е. отдать одну возможную встречу. Таким образом, после расставания оба агента имеют выгоду, один получил более лучшее решение, а второй получил возможность еще дополнительной встречи в будущем, так как на текущей встрече ему улучшить свои решения не удалось. В случае, когда агенты встречаются и их лучшие решения одинаковы, число встреч сокращается на одну у обоих агентов, что говорит о бесполезном взаимодействии. Как правило, бесполезные встречи происходят к концу процесса поиска, когда значение целевой функции не удается улучшить, даже если агенты поделились своими лучшими решениями. Следовательно, время жизни агента зависит от того, насколько полезен он был в результате взаимодействия (обмена информацией) с другими агентами.

Экспериментальная часть. Качество классификации, полученное после обучения самоконфигурируемым и многоагентным алгоритмами, сравнивалось по значениям общей точности классификации. При этом выборка разбивалась на обучающую и тестовую в соотношении 70:30. Выполнялось 25 прогонов алгоритма для каждой задачи. Обучающая выборка нормировалась на интервале [0,1], на котором задавались сигмоидальные функции. Тестовая выборка нормировалась с использованием параметров нормировки, полученных для обучающей выборки.

Алгоритмы были реализованы на языке C++ в среде Embarcadero RAD Studio XE. Тестирование производилось на ноутбуке с процессором Intel Core i5-2410M (2 ядра, 2,3 ГГц) и 8 ГБ оперативной памяти.

Для тестирования было выбрано 6 задач с репозиториев UCI [9] и KEEL [10]. Среди них:

1) Banknote – задача об определении достоверности банкнот по изображениям;

2) Column 2c – задача классификации заболеваний позвоночника 2 класса;

3) Liver – задача о заболеваниях печени;

4) Seeds – задача классификации зёрен по их форме;

5) Austrlian credit – задача о подозрительных операциях с банковскими картами;

6) German credit – задача о классификации клиентов банка.

Для самоконфигурируемого генетического алгоритма устанавливались следующие параметры: число индивидов – 700, число поколений – 700, максимальное число правил – 10, минимальная вероятность применения операторов – 0,02. Для многоагентного алгоритма использовались следующие параметры: число индивидов в генетических алгоритмах – 78, число частиц в PSO – 78 (общее число решений – 702), число поколений/итераций до встречи – 10, начальный ресурс встреч – 200. Максимальное число вычислений функции пригодности ограничивалось величиной 490000.

Результаты сравнения алгоритмов представлены в табл. 1. Самоконфигурируемый генетический алгоритм обозначен SCGA, многоагентный алгоритм – Multi-agent. По результатам сравнения можно заключить, что на обучающей выборке ГА показывает лучшие результаты для 4 задач из 6, на одной задаче результаты практически неотличимы, и на одной задаче многоагентный алгоритм показывает лучшие результаты. При этом на тестовой выборке ГА показывает лучшие результаты лишь на 2 задачах из 6, причем отставание многоагентного алгоритма не столь значительно. На одной задаче многоагентный алгоритм превосходит по точности самоконфигурируемый ГА, на остальных трех задачах результаты статистически неотличимы.

Несмотря на то, что многоагентный алгоритм показывает худшие результаты в смысле общей точности, разница между точностью на обучающей и тестовой выборках для данного алгоритма меньше, что свидетельствует о том, что он меньше переобучается.

На рис. 3 приведен график изменения ресурсов агентов в многоагентном алгоритме.

В данном случае лучшим алгоритмом оказался GA2, хорошие результаты показал также PSO1. Худшие результаты были продемонстрированы алгоритмом GA3, который терял ресурсы почти всегда. Остальные алгоритмы ведут борьбу за распределяемые ресурсы. Отметим, что это графики для конкретной задачи, для других задач графики могут быть другими и имена удачных и неудачных агентов тоже могут быть другими.

Для самонастраивающегося алгоритма в соответствии с методикой, разработанной в [11], построены графики изменения вероятностей применения операторов мутации для задачи German (рис. 4).

Как можно заметить, графики для многоагентного алгоритма и самонастраивающегося алгоритма отличаются. В частности, для самонастраивающегося некоторые из операторов оказываются эффективными на тех или иных этапах работы алгоритма, в то время как в многоагентном алгоритме изначальная тенденция по обмену встречами сохраняется на протяжении почти всего процесса оптимизации.

В табл. 2 представлено сравнение многоагентного и самонастраивающегося алгоритмов на тестовой выборке с другими методами машинного обучения. Для сравнения были выбраны машины опорных векторов (SVM), реализованные в среде Statistica 10. Выборка также разбивалась на обучающую и тестовую в соотношении 70:30, производилось 25 запусков. Также для сравнения приведен другой метод построения нечетких баз правил самоконфигурируемым генетическим алгоритмом, в котором кодируются номера нечетких термов (SCGA+FL2) [12].

Таблица 1

Сравнение точности классификации самоконфигурируемого и многоагентного алгоритма

Обучающая	Multi-agent	SCGA	Тестовая	Multi-agent	SCGA
Australian	0,893	**0,909**	Australian	**0,853**	**0,852**
Banknote	**1,000**	**0,999**	Banknote	**0,993**	**0,995**
Column_2c	**0,938**	0,931	Column_2c	**0,808**	**0,807**
Liver	0,701	**0,854**	Liver	**0,673**	0,629
German	0,753	**0,806**	German	0,701	**0,709**
Seeds	0,989	**0,992**	Seeds	0,906	**0,917**

Рис. 3. Пример изменения ресурсов агентов, задача German

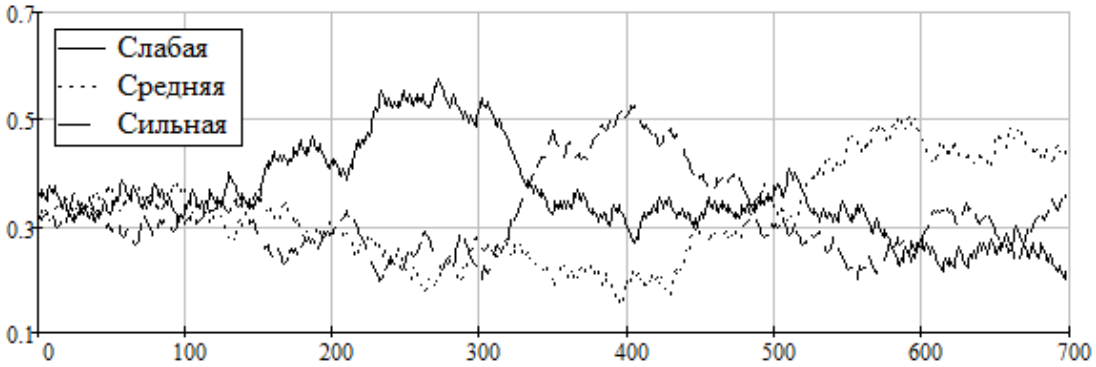

Рис. 4. Пример изменения вероятностей мутации в ГА, задача German

Таблица 2

Сравнение точности классификации с другими алгоритмами, тестовая выборка

Обучающая	Multi-agent	SCGA	SVM	SCGA+FL2
Australian	0,853	0,852	**0,864**	0,840
Banknote	0,993	**0,995**	0,987	0,942
Column_2c	**0,808**	**0,807**	0,784	0,777
Liver	0,673	0,629	**0,682**	0,568
German	0,701	0,709	**0,753**	0,707
Seeds	0,906	0,917	**0,968**	0,883

По результатам сравнения можно отметить, что машины опорных векторов показали лучшие результаты на большинстве задач. При этом на задачах Banknote и Column_2c разработанные алгоритмы показали лучшие результаты. Разница в точности классификации в основном обусловлена сложностью оптимизационной задачи построения нечеткой базы правил и, к тому же, не так уж и велика. Однако, хотя алгоритм SVM чаще показывает более высокую точность, необходимо отдавать себе отчет, что он дает намного меньше информации о решаемой задаче, чем предложенный в данной работе подход. SVM дает лишь вычислительную процедуру [13], а предложенный подход генерирует базу лингвистических правил, которая может быть интерпретирована человеком-экспертом и, тем самым, дать новые и полезные знания о причинно-следственных связях в анализируемых данных [14], чего SVM дать не могут. Незначительное снижение численной точности решений объясняется тем, что предложенным в данной работе алгоритмам выделены равные с более простыми алгоритмами вычислительные ресурсы, хотя с их помощью из базы данных извлекается намного больше информации. Более показательным является тот факт, что предложенным в данной работе подходом генерируются базы лингвистических правил, дающие более высокую точность классификации, чем базы лингвистических правил, полученные другими методами (т. е. SCGA+FL2).

Заключение. Развитие предложенного в данной работе подхода должно быть направлено на расширение возможности применения алгоритмов построения нечетких систем, в частности, использование их не только для проектирования классификаторов, но и для автоматического построения нечетких систем управления сложными объектами [15].

Благодарности. Работа поддержана грантом Президента РФ на 2014–2015 гг. (МК-5391.2014.9).

Acknowledgments. This work was supported by the grant of President of the Russian Federation for 2014–2015 (МК-5391.2014.9).

Библиографические ссылки

1. Genetic Fuzzy Systems. Evolutionary tuning and learning of fuzzy knowledge bases / O. Cordon [et al.] // Advances in Fuzzy Systems: Applications and Theory. World Scientific, 2001. 489 p.

2. Semenkin E., Semenkina M. Self-configuring genetic algorithm with modified uniform crossover operator // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 7331 LNCS (PART 1). Pp. 414–421.

3. Semenkin E., Semenkina M. Self-configuring genetic programming algorithm with modified uniform crossover // 2012 IEEE Congress on Evolutionary Computation, CEC. 2012. Pp. 1918–1923.

4. A Multi-Agent Genetic Algorithm for Global Numerical Optimization / W. Zhong [et al.] // IEEE Transactions on Cybernetics. 2004. 34(2). Pp. 1128–1141.

5. Akhmedova S., Semenkin E. Co-operation of biology related algorithms // 2013 IEEE Congress on Evolutionary Computation, CEC. 2013. Pp. 2207–2214.

6. Об эволюционных алгоритмах решения сложных задач оптимизации / А. В. Гуменникова [и др.] // Вестник СибГАУ. 2003. № 4. С. 14–24.

7. Clerc M., Kennedy J. The particle swarm – explosion, stability, and convergence in a multidimensional complex space // IEEE Transaction on Evolutionary Computation. 2002. 6(1). Pp. 58–73.

8. Mendes R., Kennedy J., Neves J. The fully informed particle swarm: Simpler, maybe better // IEEE Transactions on Evolutionary Computation. 2004. № 8 (3). Pp. 204–210.

9. Asuncion A., Newman D., UCI machine learning repository [Электронный ресурс] / University of California, Irvine, School of Information and Computer Sciences. 2007. URL: http://www.ics.uci.edu/~mlearn/MLRepository.html.

10. KEEL: A software tool to assess evolutionary algorithms for data mining problems / J. Alcalá-Fdez [et al.] // Soft Comput. 2009. Vol. 13, no. 3. Pp. 307–318.

11. Semenkin E., Semenkina M. Empirical study of self-configuring genetic programming algorithm performance and behavior // IOP Conference Series: Materials Science and Engineering. 2015. 70 (1). 012004. Pp. 1–13.

12. Stanovov V. V., Semenkin E. S. Self-adjusted evolutionary algorithms based approach for automated design of fuzzy logic systems // Вестник СибГАУ. 2013. № 4 (50). С. 148–152.

13. Akhmedova Sh. A., Semenkin E. S. SVM-based classifier ensembles design with co-operative biology inspired algorithm // Вестник СибГАУ. 2015. Т. 16, № 1. С. 22–27.

14. Становов В. В., Семенкин Е. С. Самонастраивающийся эволюционный алгоритм проектирования баз нечетких правил для задач классификации // Системы управления и информационные технологии. 2014. Т. 57, № 3. С. 30–35.

15. Бежитский С. С., Семенкин Е. С., Семенкина О. Э. Гибридный эволюционный алгоритм для задач выбора эффективных вариантов систем управления // Автоматизация. Современные технологии. 2005. № 11. С. 24.

References

1. Cordon O., Herrera F., Hoffmann F., Magdalena L. Genetic Fuzzy Systems. Evolutionary tuning and learning of fuzzy knowledge bases. *Advances in Fuzzy Systems: Applications and Theory*, World Scientific, 2001, 489 p.

2. Semenkin E., Semenkina M. Self-configuring genetic algorithm with modified uniform crossover operator. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7331 LNCS (PART 1), P. 414–421.

3. Semenkin E., Semenkina M. Self-configuring genetic programming algorithm with modified uniform crossover. *2012 IEEE Congress on Evolutionary Computation, CEC* 2012, P. 1918–1923.

4. Zhong W., Lui J., Xue M., Jiao L. A Multi-Agent Genetic Algorithm for Global Numerical Optimization. *IEEE Transactions on Cybernetics*, 2004, No. 34(2), P. 1128–1141.

5. Akhmedova S., Semenkin E. Co-operation of biology related algorithms. *2013 IEEE Congress on Evolutionary Computation*, CEC 2013, P. 2207–2214.

6. Gumennikova A. V., Emeljanova M. N., Semenkin E. S., Sopov E. A. [On evolutionary algorithms for solving complex optimization problems]. *Vestnik SibGAU*. 2003, No. 4, P. 14–24 (In Russ.).

7. Clerc M., Kennedy J. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Transaction on Evolutionary Computation*, 2002, No. 6(1), P. 58–73.

8. Mendes R., Kennedy J., Neves J. The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*. 2004, No. 8 (3), P. 204–210.

9. Asuncion A., Newman D. UCI machine learning repository. *University of California, Irvine, School of Information and Computer Sciences*, 2007, Available at: http://www.ics.uci.edu/~mlearn/MLRepository.html.

10. Alcalá-Fdez J., Sánchez L., Garcia S., del Jesus M. J., Ventura S., Garrell J. M., Otero J., Romero C., Bacardit J., Rivas V. M., Fernández J. C., Herrera F. KEEL: A software tool to assess evolutionary algorithms for data mining problems, *Soft Comput.*, 2009, Vol. 13, No. 3, P. 307–318.

11. Semenkin E., Semenkina M. Empirical study of self-configuring genetic programming algorithm performance and behavior. *IOP Conference Series: Materials Science and Engineering,* 2015, No. 70 (1), 012004, P. 1–13.

12. Stanovov V. V., Semenkin E. S. Self-adjusted evolutionary algorithms based approach for automated design of fuzzy logic systems. *Vestnik SibGAU*, 2013, No. 4 (50), P. 148–152 (In Russ.).

13. Akhmedova Sh. A., Semenkin E. S. SVM-based classifier ensembles design with co-operative biology inspired algorithm. *Vestnik SibGAU*, 2015, Vol. 16, No. 1, P. 22–27 (In Russ.).

14. Stanovov V. V., Semenkin E. S. [Self-configured evolutionary algorithm for designing fuzzy rule bases for classification problems]. *Sistemy upravleniya i informatsionnye tekhnologii*. 2014, Iss. 57, No. 3, P. 30–35 (In Russ.).

15. Bezhitskiy S. S., Semenkin E. S., Semenkina O. J. [Hybrid evolutionary algorithm for the problems of selecting effective variants of control systems]. *Avtomatizatsiya. Sovremennye tekhnologii*. 2005, No. 11, P. 24 (In Russ.).