# СОЗДАНИЕ И ПРИМЕНЕНИЕ БАЗЫ ЗНАНИЙ НА ОСНОВЕ АППРОКСИМАТИВНЫХ НЕЧЕТКИХ ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ ДЛЯ ПРОГНОЗИРОВАНИЯ INTERNET ТРАФФИКА

**Хмелевой С.В.** ✓
Донецкий национальный технический университет, г. Донецк
кафедра автоматизированных систем управления
E-mail: hmelevoy_sergey@ukr.net

***Abstract***
***Khmilovyy S.V. Creating and using approximative fuzzy logic controller knowledge base for Internet traffic forecasting****. The purpose of this article is a creating an approximative fuzzy logic controller knowledge base tuned by genetic algorithms and using this knowledge base to the problem of the time series forecasting with reference to, in particular, internet traffic volume.*

**Введение**

Задача прогнозирования временных рядов является достаточно распространенной задачей. Существует множество тестовых задач, число практических применений данной задачи также весьма значительно. Но классические методы прогнозирования не дают достаточной точности в случае, если в данных присутствуют нелинейные взаимосвязи. Достаточную точность в таких случаях дают нейронные сети, но для данных сетей существует проблема анализа полученных результатов. Обе эти проблемы пытаются решать базы знаний на основе нечетких логических контроллеров, настраиваемые с помощью генетических алгоритмов [1], [2], [3]. Но данные системы для задач прогнозирования временных рядов пока не применялись.

Целью данной статьи является применение базы знаний на основе аппроксимативных нечетких логических контроллеров применительно к задаче прогнозирования временных рядов, в частности, объема internet траффика [4].

В статье рассмотрены вопросы нечетких FLC-контроллеров, генетических алгоритмов.

**Описание системы для получения базы знаний.**

База знаний, применяемая в данной работе использует нечеткие логические контроллеры Mamdani. Она состоит из набора правил треугольного вида на входные и выходные переменные. Данные правила состоят из двух частей – условия на входные переменные (antecedent) и правила – вывода (consequent).

*Кодировка особей в популяции.* Для начальной работы с правилами используется мичиганский подход с некоторыми модификациями:

Хромосома состоит из двух частей. Первая часть включает в себя номера лингвистических переменных из некоторого первичного нечеткого множества, на основе которых строится набор правил, вторая – более точный вид правила и имеет вид $C=C_1C_2$, где $C_1= C_{11},C_{21},\ldots,C_{n1}$ – выбор конкретной лингвистической переменной для каждой входной, $C_2=C_{a12}\ C_{b12}\ C_{c12}\ C_{a22}\ C_{b22}\ C_{c22}\ldots\ C_{an2}\ C_{bn2}\ C_{cn2}\ C_{ay2}\ C_{by2}\ C_{cy2}$ – точная настройка нечетких функций правила (координаты абсцисс левого края треугольника, его центра и его правого края). С помощью части хромосомы $C_1$ выполняется грубая настройка правила, с помощью части $C_2$ – точная.

*Определение качества правила для конкретного набора данных и получение прогноза.* Входными данными для хромосомы-правила является точка из обучающего множества

$$N = (N_{x_1}, N_{x_2}, ..., N_{x_n}, N_y)$$

Для каждого значения входной переменной $Nx_i$ получаем её степень принадлежности треугольной функции принадлежности $A_i$ (получение ординаты функции по конкретному

значению абсциссы, фуззификация). Из полученных степеней принадлежности различных переменных для одного правила получаем одно значение с помощью одной из функций, удовлетворяющих условиям t-норм. Чаще всего это функция минимум.

$R^i = *(A(N_x^i), B(N_y^i))$, i=1..p

$A(N_x^i) = *(A_1(N_{x_1}^i), ..., A_1(N_{x_n}^i))$, где

* - t-норма.

Где $R^i$-степень соответствия правила примеру. Хорошие правила должны иметь высокую степень соответствия большому количеству примеров. Для получения прогноза используют значение $A(N_x^i)$. По этому значению из правила на выходную переменную производят отсечение части фигуры. При наличии нескольких правил в базе знаний получаем такое же количество выходных значений, как и правил в популяции. Затем из этих выходных фигур получается одна прогнозная фигура с помощью функции, удовлетворяющей t-конорме (дефуззификация). Чаще всего это функция максимум. В данной работе для дефуззификации используют центр тяжести фигуры.

$$Z = \frac{\int_{-\infty}^{\infty} y * B(y) dy}{\int_{-\infty}^{\infty} B(y) dy}$$, или в дискретной форме

$$Z = \frac{\sum y * B(y)}{\sum B(y)}.$$

Подробно этот процесс описан в [6].

*Фитнесс-функция* для продукционных ГА состоит из нескольких составляющих и объединяются мультипликативно. В описываемой системе используются такие составляющие:

- Высокое значение частоты правила. Частота нечеткого правила $R_i$, вычисляемая по множеству примеров $E_p$, определяется как:

$$\Psi_{E_p}(R_i) = \frac{\sum_{l=1}^{p} R_i(e_l)}{p}$$, где

p - количество правил в обучающей выборке

$e_l$ – l-й пример из обучающей выборки

- Большая степень покрытия положительных примеров. Множество положительных примеров, отобранных $R_i$, со степенью покрытия точек больше ω, определяется как:

$$E_\omega^+(R_i) = \{e_l \in E_p / R_i(e_l) \geq \omega\},$$

обозначив количество положительных примеров, отобранных правилом, через $n_\omega^+ = \left|E_\omega^+(R_i)\right|$, можно определить среднюю степень покрытия положительных примеров как:

$$G_\omega(R_i) = \sum_{e_l \in E_\omega^+(R_i)} R_i(e_l) / n_\omega^+(R_i)$$

- Малая степень покрытия отрицательных примеров. Отрицательным считается такой пример из обучающей выборки, который отбирается правилом, но не покрывается им со степенью ω. Правила, которые дают много таких оценок, должны штрафоваться.

$$g_n(R_i^-) = \begin{cases} 1 \quad ,если \quad n_\omega^-(R_i) \le k * n_\omega^+(R_i) \\ \dfrac{1}{n_\omega^-(R_i) - k * n_\omega^+(R_i) + \exp(1)}, в \quad остальных \quad случаях \end{cases}$$

Если степень покрытия положительных примеров можно считать не на всем множестве данных, а только на тех, которые в настоящий момент надо покрыть, то степень покрытия отрицательных примеров всегда считают по всему множеству данных.

- Малая степень взаимного перекрытия с другими правилами множества. Имеем центры правил $N_i$=($N_i$x, $N_i$y), определенных ранее в процессе работы алгоритма (i=1..d), где d-количество шагов алгоритма по генерации правил. Степень взаимного перекрытия правил можно определить как

$$LNIR(R) = 1 - NIR(R)$$

$$NIR(R) = \bigcup_i R^i$$

$$R^i = *(A(N_x^i), B(N_y^i)), \text{ i=1..p}$$

$$A(N_x^i) = *(A_1(N_{x_1}^i), ..., A_1(N_{x_n}^i)),$$

$R^i$: ЕСЛИ $x_1 \in A_1$, И…И $x_n \in A_n$, ТО $y \in B$.

где * - t-норма

Малые значения LNIR получаются, если в популяции уже созданы правила, подобные данному. Если существуют аналогичные правила, значение LNIR=0.

*Генетические операторы*

Для каждой из частей хромосомы (грубой и точной настройки) разработаны свои генетические операторы рекомбинации, взятые из дескриптивного и ассоциативного подходов соответственно. В зависимости от того, одинаковая ли первая часть у двух родителей, выполняется один из двух аналогичных операторов рекомбинации:

Классический кроссинговер для данной кодировки особи работает и применяется при грубой настройке правил. Он выполняется на первой части хромосомы (C1), точки из C2 переносятся из предварительно сформированного эталонного нечеткого набора (primary fuzzy set), предоставляющего эталонный вид множества C2 для каждого возможного вида C1. При двух одинаковых частях C1 кроссинговер выполнять нельзя.

Мин-максный кроссинговер применяется, когда у родителей одинаковые первые части (C1), т.е. для точной настройки правил, он работает над частями C2.

Для родителей $C_w^t = (c_1, \ldots, c_k, \ldots, c_{3*(n+1)})$ и $C_v^t = (c_1', \ldots, c_k', \ldots, c_{3*(n+1)}')$ получаются 4 потомка

$$C_1^{t+1} = \alpha * C_w^t + (1-\alpha) * C_v^t,$$

$$C_2^{t+1} = \alpha * C_v^t + (1-\alpha) * C_w^t,$$

$$C_3^{t+1} = \min\{c_k, c_k'\},$$

$$C_4^{t+1} = \max\{c_k, c_k'\}.$$

Выбираются два лучших из этих четырех потомков.

Мутация на C1 (мутация Thrift-a) случайно меняет одну лингвистическую переменную для данного правила на соседнюю в большую или меньшую сторону.

Мутация на C2 (предложена Z.Michalewicz). Из особи случайно выбирается точка $c_k$ (с разрешенными пределами изменения [$c_{kl}$ $c_{kr}$]). Точка меняется на

$$c_k' = \begin{cases} c_k + \Delta(t, c_{kr} - c_k), при \quad a = 0, \\ c_k - \Delta(t, c_k - c_{kl}), при \quad a = 1, \end{cases} \text{ где}$$

a – случайно выбранное направление изменения,

$\Delta(t, y)$ - функция, возвращающая случайную величину в пределах [0..y] таким образом, что при увеличении t среднее возвращаемое значение увеличивалось:

$\Delta(t, y) = y(1 - r^{(1-\frac{t}{T})^b})$, где

r – случайная величина на интервале [0..1]

t – текущая эпоха работы генетического алгоритма

T – общее разрешенное число эпох алгоритма

b – задаваемый пользователем параметр, определяющий степень зависимости от числа эпох.

*Организация полноты покрытия правилами обучающего множества.*

Для того, чтобы правила полностью покрывали обучающее множество, используется понятие покрытия (covering). Величина покрытия для обучающей точки $e_l$ считается по выражению:

$CV_R(e_l) = \sum_{i=1}^{T} R_i(e_l)$, где

$R_i(e_l)$ - степень покрытия i-м правилом l-го примера из обучающего набора,

T – общее число правил.

Поскольку база знаний состоит из множества правил, необходимо добиться, чтобы это множество покрывало все точки обучающего множества с достаточной степенью покрытия. Для этого после создания каждого нового правила величина покрытия каждой точки ОВ пересчитывается. При достижении величиной определенного порога точку удаляют из подмножества, участвующего в составлении правил. Для оценки качества вновь создаваемых правил их уже не используют.

*Итеративный процесс создания правил*

Правила генерируются частично случайно, частично используя некоторое множество из ОВ (около 1/3 точек) для максимального их покрытия. Затем запускается генетический алгоритм для достижения максимальной степени покрытия сгенерированным множеством правил обучающей выборки. Из популяции берется одно лучшее правило, оно добавляется в итоговое множество правил. Величина покрытия для ОВ пересчитывается, ОВ сокращается и снова происходит генерация новой популяции на основе сокращенной ОВ с отбором лучшей хромосомы. Процесс итеративно повторяется до полного покрытия ОВ.

*Дальнейшее сокращение избыточного множества правил (мультисимплификация)*

После полного покрытия правилами обучающей выборки имеем множество правил, которое избыточно покрывает данные (и не всегда дает точный прогноз благодаря своей избыточности). Это множество необходимо сократить для увеличения точности прогноза. Это производится еще одним генетическим алгоритмом.

Хромосома имеет T двоичных генов, где T – количество правил. Применяемые генетические операторы – классические. Фитнесс – функцией является MSE прогнозных и реальных значений ОВ:

$E(C_j) = \frac{1}{2|E_{TDS}|} \sum_{e_l \in E_{TDS}} (ey^l - S(ex^l))^2$, где

$|E_{TDS}|$ - величина ОВ,

$S(ex^l)$ - прогноз системы для подаваемых входных значений $ex^l$.

Имеет смысл использовать понятие величины покрытия для проверки полноты использования набором правил обучающей выборки.

$C_{R(C_j)}(e_l) = \bigcup_{j=1..T} R_j(e_l) \geq \tau, \forall e_l \in E_{TDS} \; u \quad R_j \in R(C_j)$,

где τ - минимальная величина покрытия всеми правилами примера из ОВ. Находим величину покрытия для всех правил:

$$TSCD(R(C_j), E_{TDS}) = \bigcap_{e_l \in E_{TDS}} C_{R(C_j)}(e_l),$$

при превышении общей величины покрытия порога τ считается, что система хорошо подстроена под данную ОВ, иначе в фитнесс-функцию необходимо ввести штрафную составляющую:

$$F(C_j) = \begin{cases} E(C_j), если \quad TSCD(R(C_j), E_{TDS}) \geq \tau, \\ \frac{1}{2}\sum_{e_l \in E_{TDS}} (ey^l)^2, иначе. \end{cases}$$

Один запуск данного генетического алгоритма (процедуры симплификации) даст одно субоптимальное решение. Для нахождения наилучшего решения необходимо определить несколько таких решений (мультисимплификация). Для этого выполняется несколько запусков генетического алгоритма, при каждом запуске лучшая особь записывается в массив лучших особей, а фитнесс-функция каждой особи популяции модифицируется для направления поиска в другие регионы, удаленные от уже найденных решений:

$F'(C_j) = F(C_j) * G(C_j, S)$, где

$S = \{s_1, s_2, ...s_k\}$ – множество уже найденных субоптимальных решений,

$$G(C_j, S) = \begin{cases} \infty, & если\, d = 0 \\ 2 - \left(\frac{d}{r}\right)^{\beta}, & если\, 0 < d < r \\ 1, & если\, d \geq r \end{cases}, \text{ где}$$

d – минимальное расстояние по Хэммингу между хромосомой $C_j$ и уже найденными решениями $s_j$, включенными в S, т.е. $d = Min_i \{H(C_j, s_i)\}$,

r – радиус ниши, которую занимает каждое из уже найденный решений,

β - величина, определяемая пользователем.

*Окончательное уточнение формы правил (тюнинг)*

После удаления из множества правил излишние, форму остальных правил необходимо подстроить для максимально точного описания данных. Это производится на заключительном этапе тюнинга системы еще одним генетическим алгоритмом.

Входной информацией являются результаты работы мультисимпликационного генетического алгоритма. Сокращенную там популяцию используют для кодирования 1-й хромосомы на 3 этапе. Из особей оставляются только части хромосом $C_2$. Все они склеиваются в одну мультихромосому. Таким образом, при количестве особей в сокращенной популяции = m, длина мультихромосомы равна m*3*(n+1). Для каждого гена хромосомы по первой хромосоме определяется разрешенный интервал. Он получается таким образом:

Если (t mod 3)=1, где mod – операция остатка от деления, то $c_t$ – самый левый ген из одной нечеткой функции принадлежности. Тогда разрешенные интервалы будут выглядеть так:

$$c_t \in [c_t^l, c_t^r] = \left[c_t - \frac{c_{t+1} - c_t}{2}, c_t + \frac{c_{t+1} - c_t}{2}\right],$$

$$c_{t+1} \in [c_{t+1}^l, c_{t+1}^r] = \left[c_{+1t} - \frac{c_{t+1} - c_t}{2}, c_{t+1} + \frac{c_{t+2} - c_{t+1}}{2}\right],$$

$$c_{t+2} \in [c^l_{t+2}, c^r_{t+2}] = \left[ c_{t+2} - \frac{c_{t+2} - c_{t+1}}{2}, c_{t+2} + \frac{c_{t+3} - c_{t+2}}{2} \right].$$

Используя эти интервалы, все особи популяции, кроме первой, создаются случайно. Далее запускаем стандартный генетический алгоритм. Используемые в нем операторы рекомбинации: мин-максный кроссинговер, мутация Michalewicz-a. Фитнесс-функция - стандартная MSE.

Таким образом, всего система состоит из трех этапов – первоначального создания правил, удаления излишних и окончательной подстройки их формы. Для этого используются три различных генетических алгоритма и три различных представления хромосомы.

Все вышесказанное может быть проиллюстрировано рис. 1.

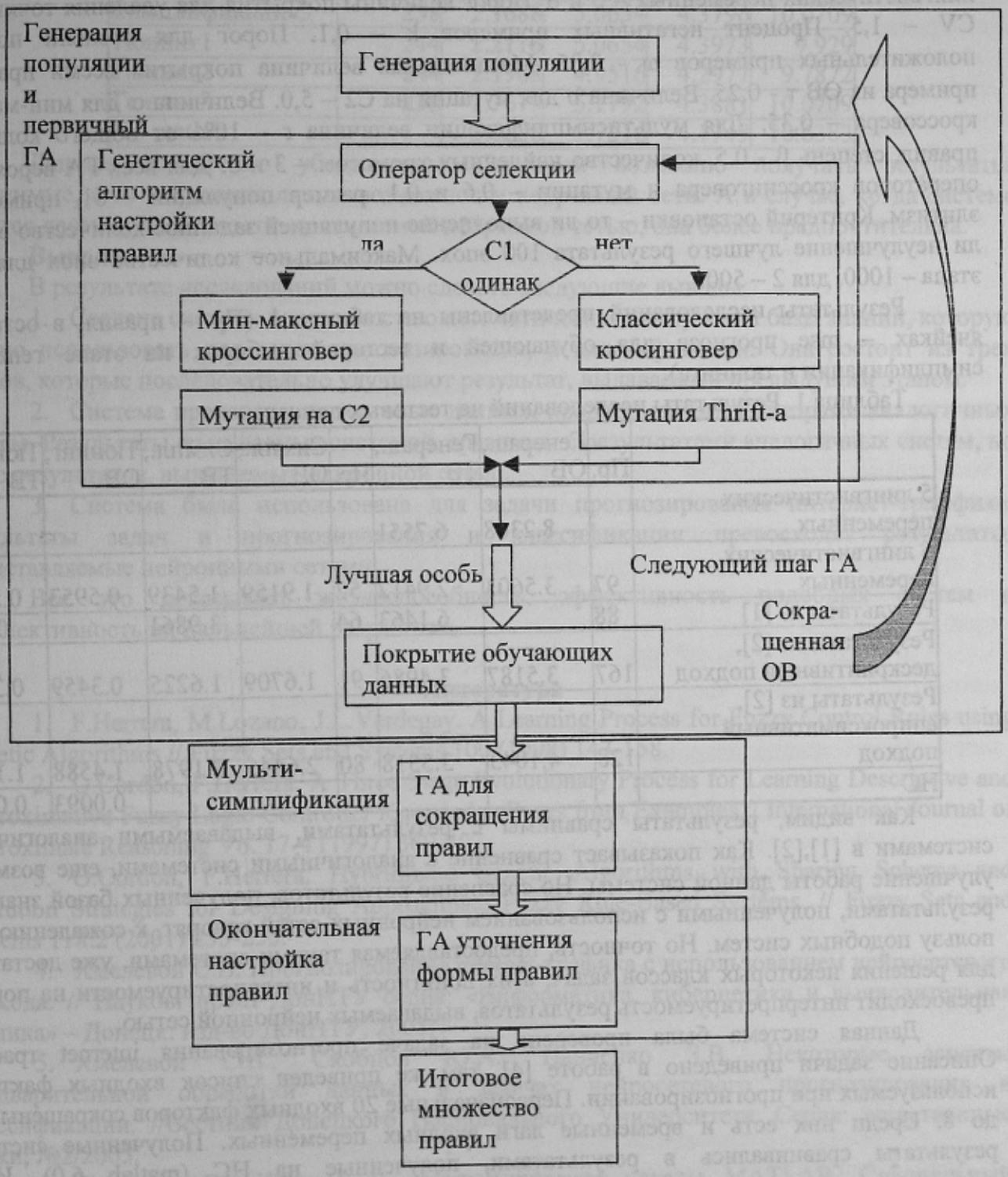


Рисунок 1 - Система получения базы знаний с помощью ГА.

**Практическое применение данной системы**

В качестве тестовой задачи использовалась задача spherical model. Это достаточно простая унимодальная функция

$F(x_1, x_2) = x_1^2 + x_2^2,\ x_1, x_2 \in [-5..5], F(x_1, x_2) \in [0..50]$.

Для составления ОВ пространство поиска каждой входной переменной разбили на 41 равный интервал, в итоге получено 1681 точка. Для составления ТВ (тестовой выборки) входные точки были взяты с помощью генератора случайных чисел, в объеме 10% от ОВ (168 точек) [1],[2]. Полученные результаты сравнены с результатами из исходных источников, а также с результатами, выдаваемыми нейронной сетью.

Параметры алгоритма использовались следующие: количество базовых лингвистических переменных – 5 и 6. Порог величины покрытия для удаления точки из ОВ CV – 1,5. Процент негативных примеров k – 0,1. Порог для степени покрытия положительных примеров ω - 0,05. Минимальная величина покрытия всеми правилами примера из ОВ τ - 0,25. Величина b для мутации на C2 – 5,0. Величина α для мин-максного кроссовера – 0,35. Для мультисимплификации величина r – 10% от общего количества правил, степень β - 0.5, количество найденных хромосом – 3 и 5. Для всех ГА вероятности операторов кроссинговера и мутации – 0,6 и 0,1, размер популяции – 61, применяется элитизм. Критерий остановки – то ли выполнение популяцией заданное количество эпох, то ли неулучшение лучшего результата 100 эпох. Максимальное количество эпох для 1 и 3 этапа – 1000, для 2 – 500.

Результаты исследований представлены на таблице 1 (Пр. – правил, в остальных ячейках – mse прогноза для обучающей и тестовой выборок на этапе генерации, симплификации и тюнинга).

Таблица 1. Результаты исследований на тестовых данных.

| | Пр. | Генерац., ОВ | Генерац., ТВ | Пр. | Симпл., ОВ | Симпл., ТВ | Тюнинг, ОВ | Тюнинг, ТВ |
|---|---|---|---|---|---|---|---|---|
| 5 лингвистических переменных | | 8.2368 | 6.7551 | | | | | |
| 6 лингвистических переменных | 97 | 3.5603 | 2.9412 | 52 | 1.9159 | 1.5439 | 0.5953 | 0.5339 |
| Результаты из [1] | 88 | | 6.1463 | 64 | | 3.9861 | | 0.768 |
| Результаты из [2], дескриптивный подход | 167 | 3.5187 | 3.4986 | 91 | 1.6709 | 1.6225 | 0.3459 | 0.3174 |
| Результаты из [2], аппроксимативный подход | 128 | 4.1043 | 3.5518 | 80 | 2.6445 | 2.1978 | 1.4588 | 1.1293 |
| НС | | | | | | | 0.0093 | 0.0047 |

Как видим, результаты сравнимы с результатами, выдаваемыми аналогичными системами в [1],[2]. Как показывает сравнение с аналогичными системами, еще возможно улучшение работы данной системы. Но сравнение результатов, полученных базой знаний, с результатами, полученными с использованием нейронных сетей говорят, к сожалению, не в пользу подобных систем. Но точность, предоставляемая такими системами, уже достаточна для решения некоторых классов задач, а их понятность и интерпретируемости на порядок превосходит интерпретируемость результатов, выдаваемых нейронной сетью.

Данная система была проверена на задаче прогнозирования internet трафика. Описание задачи приведено в работе [4], там же приведен список входных факторов, используемых при прогнозировании. Первоначальные 20 входных факторов сокращены в [5] до 8. Среди них есть и временные лаги входных переменных. Полученные системой результаты сравнивались в результатами, полученные на НС (matlab 6.0). Итоги

представлены в таблице 2 (Пр. – количество правил, mse*100 для прогнозирования на ОВ и ТВ, % погрешности классификации для ОВ и ТВ).

Таблица 2. Результаты исследований для задачи прогнозирования объема интернет-траффика

| | Правил | ОВ | % Класс. ОВ | ТВ | % Класс. ТВ |
|---|---|---|---|---|---|
| Случайные правила | 300 | 4.8 | 10.53 | 6.82 | 15.18 |
| Генерация | 457 | 5.8576 | 6.0899 | 8.129 | 12.9078 |
| Симплификация1 | 244 | 2.2394 | 5.1792 | 4.4271 | 9.9291 |
| Симплификация2 | 244 | 2.2331 | 5.1224 | 4.5003 | 10.6383 |
| Симплификация3 | 238 | 2.1688 | 5.0654 | 4.3796 | 10.0709 |
| Тюнинг1 | 244 | 2.2116 | 5.0654 | 4.3973 | 9.929 |
| Тюнинг2 | 244 | 2.1988 | 4.9516 | 4.2977 | 9.7872 |
| Тюнинг3 | 238 | 2.1619 | 5.1792 | 4.3881 | 10.0709 |
| НС matlab | | 3.0286 | 7.7216 | 4.1749 | 10.6383 |

Данные достаточно убедительны. Системой возможно получить результаты, сравнимые, а то и превосходящие возможности нейронной сети. А в случае, когда системе удастся достигнуть точности, достигаемой нейронной сетью, она более предпочтительна.

**Выводы**

В результате исследований можно сделать следующие выводы:

1. Создана система для полностью автоматического получения базы знаний, которую можно использовать для задач прогнозирования, и классификации. Она состоит из трех этапов, которые последовательно улучшают результат, выдаваемый предыдущим этапом.
2. Система проверена на тестовых данных, предоставляемых авторами аналогичных систем. Результаты, выдаваемые системой, сравнимы с результатами аналогичных систем, но хуже результатов, выдаваемых нейронной сетью.
3. Система была использована для задачи прогнозирования интернет-траффика. Результаты задач и прогнозирования и классификации превосходят результаты, представляемые нейронными сетями.

Все это доказывает жизнеспособность, эффективность подобных систем и перспективность их дальнейшей разработки.

**Литература**

1. F.Herrera, M.Lozano, J.L.Verdegay. A Learning Process for Fuzzy Control Rules using Genetic Algorithms // Fuzzy Sets and Systems 100 (1998) 143-158.
2. O.Cordon, F.Herrera. A Three-Stage Evolutionary Process for Learning Descriptive and Approximative Fuzzy Logic Controller Knowledge Bases from Examples.// International Journal of Approximate Reasoning Vo. 17-4 (1997) 369-407.
3. O.Cordon, F.Herrera. Hybridizing Genetic Algorithms with Sharing Scheme and Evolution Strategies for Designing Approximate Fuzzy Rule-Based Systems. // Fuzzy Sets and Systems 118:2 (2001) 235-255.
4. Хмелевой С.В. Прогнозирование internet-трафика с использованием нейросетевого подхода. // Наукові праці ДонНТУ серия: «Информатика, кибернетика и вычислительная техника» – Донецк: изд-во ДонНТУ, 2005г.
5. Хмелевой С.В., Скобцов Ю.А., Панченко З.В. Некоторые аспекты предварительной обработки данных в задачах нейросетевого прогнозирования и классификации. //Вестник Донецкого Национального Университета. Серия: естественные науки, №2/2005.
6. Дьяконов В., Круглов В.. Математические пакеты MATLAB. Специальный справочник. – СПб.: Питер, 2001. – 480с.: ил.