УДК 681.306

МЕТОД ЭВОЛЮЦИОННОГО НАКОПЛЕНИЯ ПРИЗНАКОВ ДЛЯ АВТОМАТИЧЕСКОГО ПОСТРОЕНИЯ НЕЙРОННЫХ СЕТЕЙ

В. В. Тютерев[1]

Предлагается новый подход к автоматическому построению нейронных сетей, который объединяет положительные стороны конструктивного и эволюционного подходов. Основная идея алгоритма заключается в том, что на каждом этапе работы нейронная сеть достраивается блоками нейронов, максимально уменьшающими ошибку. Поиск блоков ведется при помощи генетических алгоритмов. Процесс построения нейронной сети прекращается при достижении заданной величины ошибки. Эффективность предложенного алгоритма исследуется на ряде практических задач.

Ключевые слова: нейронные сети, построение нейронных сетей, генетические алгоритмы, нейроинформатика.

1. Введение. Нейронные сети представляют собой основную структуру для обработки информации в математической дисциплине, носящей название *нейроинформатика.* Формально искусственную нейронную сеть можно определить как *систему, имеющую входы и выходы, состоящую из большого числа простых параллельных вычислителей, в произвольном порядке соединенных односторонними каналами передачи сигнала. В процессе работы нейронная сеть преобразует поданный сигнал от входов в свои выходы.* В 1957 г. Колмогоровым была доказана теорема, которая позволяет говорить о том, что для решения любой задачи возможно построить нейронную сеть [1].

Начиная с конца 80-х годов по настоящее время теория нейронных сетей испытывает настоящий бум по количеству научных публикаций. Нейронные сети специальных типов все чаще применяются для решения разнообразных практических задач. Устройства, построенные на нейронных сетях, вошли в нашу жизнь и используются в промышленности, финансовой системе и в повседневной жизни.

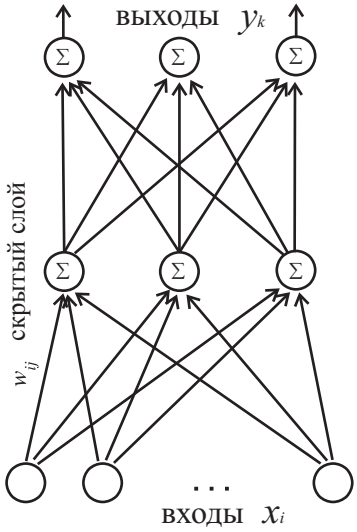

Рис. 1. Многослойный персептрон

В данной статье мы будем рассматривать один из самых распространенных типов нейронных сетей — *многослойный персептрон* (*multi-layer perceptron*). Многослойный персептрон с одним скрытым слоем реализует следующую сложную функцию:

$$y_k = g_k\left(w_0 + \sum_{j=1} w_{jk} g\left(w_{0j} + \sum_{i=1} w_{ij} x_i\right)\right). \qquad (1)$$

Здесь $W = \{w_{ij} : i = 1, \ldots, p\}$ называются *весами сети*; $g()$ — нелинейные пороговые функции, обычно *логический сигмоид* $g(a) = 1/(1 + e^{-a})$ или *тангенс гиперболический* $g(a) = \tanh \equiv (e^a - e^{-a})/(e^a + e^{-a})$; g_k может быть линейной функцией, сигмоидом или tanh; w_0 и w_{0j} — *пороговые значения нейронов*, которые обычно переносятся под знак суммы следующей модификацией входного вектора: $x = (1, x_1, \ldots, x_d)$. Число нейронов, их связность и количество слоев будут определять возможности конкретной нейронной сети. Настройку весов многослойного персептрона обеспечивает специальная *процедура обучения.* В 1986 г. Румельхарт (Rumelhart), Хинтон (Hinton) и Вильямс (Williams) предложили для этого эффективный алгоритм *обратного распространения ошибки.*

Основной задачей в теории нейронных сетей является нахождение способов получения нейронной сети, наилучшим образом решающей определенный класс задач. Эта проблема включает в себя определение структуры нейронной сети и настройку ее параметров — обучение, включающее в себя определение эффективности готовой сети. В настоящее время существует достаточное количество эффективных методов обучения нейронных сетей и разработаны совершенные критерии оценки, но не решена окончательно важная проблема определения структуры сети, т.е. количества и способа взаимной организации нейронов.

[1] Томский государственный университет, механико-математический факультет, пр. Ленина, д. 36, 634050, г. Томск; e-mail: vlad@math.tsu.ru

Непараметрические модели, к которым можно отнести нейронные сети, в силу своей гибкости характеризуются большим количеством всех возможных моделей. Из всего класса возможных моделей выбирается такой элемент, который лучше "настраивается" на обучающую выборку. Определим формально критерий выбора такой модели.

Пусть мы имеем некоторую обучающую выборку D из N пар наблюдений вида $D = \{(\vec{x_i}, y_i) : i = 1, \ldots, N\}$, определяющих отображение $f : x \to y$. Будем считать, что выборка D была сгенерирована по следующему правилу:

$$y_i = f(x_i) + \xi(x). \tag{2}$$

Здесь $f(x)$ — некоторая неизвестная функция; входы x_i подаются независимо с неизвестной стационарной функцией плотности $p(x)$; $\xi(x)$ — независимые случайные величины с нулевым средним $\overline{\xi} = 0$ и неизвестной дисперсией σ_ξ^2; y_i — полученные выходные значения.

Задача *обучения*, или задача регрессии, заключается в том, чтобы по выборке D среди класса возможных моделей $f_\Delta(x)$ выбрать модель $\hat{f}(x, D)$, как можно точнее оценивающую вид $f(x)$. В терминах теории нейронных сетей задачу регрессии можно сформулировать как поиск среди множества всех возможных нейронных сетей Δ нейронной сети, преобразующей входные векторы x_i в свои выходы y_i и реализующей оценку $\hat{f}(x, D)$.

Если зафиксировать класс нейронных сетей наиболее распространенным классом — сетями прямого распространения, все множество нейронных сетей Δ можно разбить на счетное множество подклассов A, определяемых выбранной топологией сети (числом уровней L, количеством нейронов сети γ). В пределах каждого класса $A_i \subset A$ нейронные сети будут характеризоваться дополнительным набором параметров — набором весов связей между нейронами сети $W = \{w^i, i = 1, \ldots, p\}$. Таким образом, мы имеем вектор настраиваемых параметров нейронной сети $\theta \equiv \{L, \gamma, W\}$.

Естественным критерием для выбора нейронной сети, реализующей оценку $\hat{f}(x)$, будет функция, заданная среднеквадратическим отклонением для произвольных входных данных:

$$P(x) = \int_x \left[f(x) - \hat{f}(x)\right]^2 p(x)dx + \sigma_\xi^2. \tag{3}$$

При конечном числе примеров, для значения $P(x)$ можно привести оценку

$$\widehat{P}(x) = E\left\{\frac{1}{N}\sum_{i=1}^{N}[y_i^* - \hat{f}(x_i^*)]^2\right\}. \tag{4}$$

Здесь $(\vec{x}_i^*, y_i^*)$ — точки *тестовой выборки* или новые измерения, полученные по (2) и не участвовавшие при конструировании $\hat{f}(x)$. Функция $E\{\cdot\}$ определяет дисперсию аргумента. Оценка $\widehat{P}(x)$ характеризует обобщающую способность конкретной нейронной сети.

Задача построения нейронной сети и получения хорошей оценки (4) существенно усложнена по причине зашумленности и ограниченного объема обучающей выборки D. Подобный ограниченный объем данных приводит, помимо всего прочего, к проблеме, известной в научной литературе как *проблема стабильности–пластичности* (*biase–variance trade-off*) [2]. Рассмотрим эту проблему на простом примере.

В качестве способа построения $\hat{f}(x)$ мы выберем полином степени M:

$$\hat{f}(x) = w_0 + w_1 x + \cdots + w_M x^M. \tag{5}$$

Подбирая некоторым образом веса w_i, можно добиться минимального значения $\widehat{P}(x)$. На рис. 2 (а) приведен результат, полученный для $M = 1$ (линейной функции). Можно видеть, что такой полином дает очень плохое приближение для $f(x)$ из-за своей ограниченной пластичности. Лучшие результаты мы получим, наращивая степень полинома, т.е. увеличивая число *свободных переменных* модели. На рис. 2 (б) показана оценка для $M = 12$. Хотя полином достигает наилучшего приближения к данным обучающей выборки D, тем не менее оценка $\widehat{P}(x)$ будет далека от минимальной из-за высокой степени осцилляции $\hat{f}(x)$. Только для варианта на рис. 2 (в) можно сказать, что при $M = 3$ мы получим лучшую оценку $\widehat{P}(x)$.

Таким образом, обобщающая способность модели будет кардинально зависеть от ее сложности: модель с малым числом свободных переменных ($M = 1$) имеет высокую *стабильность*, в то время как модель с высокой степенью свободы ($M = 12$) обладает большей *пластичностью*. Основной задачей регрессии будет определение модели, реализующей некий компромисс между этими крайностями.

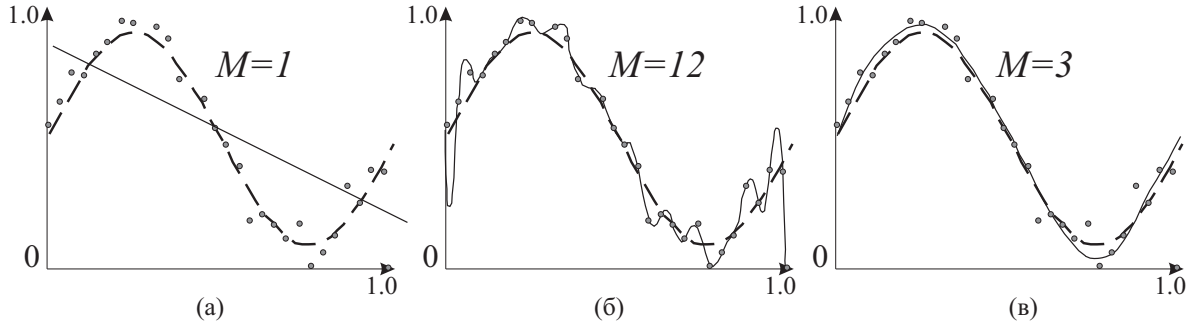

Рис. 2. Пояснение к проблеме стабильности-пластичности

Приведенная выше проблема зависимости эффективности модели от ее сложности послужила толчком для появления нескольких более совершенных критериев оценки лучшей модели. Ряд новых критериев (таких как *информационный критерий Акаика* (*AIC*), *байесовский информационный критерий* (*BIC*), *сетевой информационный критерий* (*NIC*), *обобщенная прогнозируемая ошибка* (*GPE*), *описание минимальной длины* (*MDL*) и др.) принимают во внимание сложность модели при построении целевой функции [3].

К сожалению, рассмотренные критерии (как общего назначения, так и разработанные специально для нейронных сетей) предлагают только способы оценки эффективности модели в пределах семейства моделей, но *не предоставляют механизма поиска* моделей-кандидатов. Возвращаясь к нейронным сетям, можно сказать, что подавляющее число исследователей для определения эффективной нейронной сети используют метод эмпирического подбора, когда вручную перебирается несколько вариантов нейронной сети различной архитектуры. Каждый из вариантов проверяется по одному из перечисленных критериев; выбор останавливается на архитектуре, наиболее точно реализующей оценку. С увеличением сложности задач подобный подход становится чрезвычайно трудоемким; в этом случае качество решения проблемы полностью зависит от опыта и интуиции исследователя.

Решением поставленной проблемы автоматического нахождения нейронной сети, наилучшим образом реализующей $\hat{f}(x)$, является отказ от жесткой фиксации топологии нейронной сети до начала этапа обучения. Точнее говоря, происходит *объединение процесса создания нейронной сети с алгоритмом ее обучения.* Таким образом, реализуется следующая общая схема: перебор (в соответствии с некоторым критерием) нейронных сетей возможных архитектур одновременно с оценкой эффективности каждой полученной нейронной сети.

Согласно [4], все способы автоматического построения нейронных сетей можно достаточно условно разделить на два направления — *конструктивные алгоритмы* и *методы глобального поиска.*

Исторически более ранние, конструктивные методы объединяет то, что нейронная сеть оптимальной топологии получается за счет построения нейронной сети по шагам за счет наращивания или усечения архитектуры. По способу конструирования сети выделяют, соответственно, два подхода: *роста* (*growing*) и *усечения* (*pruning*) нейронной сети [5]. Подход *роста* предполагает начало построения с простейшей нейронной сети с последующим добавлением элементов архитектуры сети. Направление *усечения* последовательно упрощает уже имеющуюся обученную нейронную сеть, поэтапно отсекая фрагменты архитектуры. В качестве варьируемых элементов архитектуры могут выступать, например, количество нейронов в скрытом слое для сетей прямого распространения или число центров в сетях радиально базисных функций.

Решение о способе модификации нейронной сети принимается в соответствии с фиксированным алгоритмом построения. Алгоритм построения задается таким образом, чтобы добавление новых нейронов (связей) на каждом этапе *гарантированно уменьшало* (*в случае усечения не увеличивало*) значение ошибки работы сети. Как правило, схема работы алгоритмов *роста* носит характер локализации неверно решаемых нейронной сетью примеров обучающей выборки D, фиксирование уже построенной сети для правильно решаемых примеров и добавление архитектуры нейронами (связями), корректирующими работу сети для неверно решаемых примеров [6, 7].

Методика работы алгоритмов *усечения* предполагает определение степени влияния отдельных элементов архитектуры нейронной сети на значение ошибки и отбрасывание наименее значимых. В отечественной литературе техника усечения весов носит название *процедура контрастирования*, которая впервые была

предложена в работе [8].

Основным преимуществом конструктивного подхода следует признать небольшое время работы. Это непосредственно вытекает из алгоритма построения, при котором всякий раз мы имеем дело с нейронной сетью уже, хотя бы частично, решающей задачу. Каждое дальнейшее усложнение архитектуры только улучшает результат. Еще одним положительным результатом подобного подхода к построению сетей является иерархическая организация архитектур созданных нейронных сетей. В ряде случаев (при использовании некоторых эффективных критериев оценок обобщающей способности нейронной сети) для корректного сравнения различных построенных нейронных сетей свойство вложенности будет важным [3, 9].

Недостатком конструктивного подхода, точнее направления роста, является то, что архитектура построенной нейронной сети может значительно отличаться от оптимальной. Подобная организация поиска затрагивает фактически только смежные классы архитектур A_i в пространстве архитектур A. В таком случае мы можем только рассчитывать на попадание в ближайший локальный минимум. Более того, большая часть конструктивных алгоритмов жестко фиксирует способ наращивания архитектуры (например, добавлением нейронов в единственный скрытый слой сети или же, наоборот, присоединением нового скрытого слоя с единственным нейроном, что приводит к дополнительному раздуванию размеров сети). В плане критики направления усечения можно сказать только то, что единственным неясным моментом является способ получения исходной усекаемой сети. Очевидно, что такую сеть мы сможем предварительно иметь не для всех случаев.

Методы глобального поиска, как следует из названия, основаны на поиске в пределах всех возможных моделей в пространстве архитектур. Среди всех возможных топологий нейронных сетей целенаправленно ищется единственная сеть, которая будет формировать наилучшую оценку $\hat{f}(x)$. Современным способом построения нейронных сетей является использование эволюционного подхода к поиску по пространству возможных архитектур. Исторически сложились три методологии эволюционного поиска [10]: *эволюционное программирование* (*evolutionary programming*) Фогела (1966, Fogel), *эволюционные стратегии* (*evolution strategies*) Реченберга (1973, Rechenberg) и *генетические алгоритмы* (*genetic algorithms*) Холланда (1975, Holland). Эти подходы имеют схожую идею, но различаются в выборе представления индивидуумов, механизмах селекции, форме генетических операторов и способах оценки приспособленности. Наиболее универсальными являются генетические алгоритмы (ГА) Холланда [11, 12]. Идея ГА основана на принципе эволюции биологических существ: из всей совокупности особей данного вида выживают индивидуумы, наиболее приспособленные к сложившимся условиям окружающей среды. Полезные свойства закрепляются и накапливаются на генетическом уровне в последующих поколениях путем обмена генами. Возвращаясь к нейронным сетям, можно сказать, что в данном случае наборы нейронных сетей будут выступать в качестве популяций, архитектура сети (число уровней, нейронов, связей) будет генетическим кодом, критерием приспособленности будет наилучшая точность результатов и малый размер сети.

Поиск оптимальной топологии сети начинается с выбора способа кодирования архитектуры сети в *генетическую строку*, описывающую отдельную особь [13]. На каждом этапе эволюции из популяции выбирается несколько лучших особей, порождающих с помощью генетических операций (мутации, кроссинговера) новые экземпляры, которые, в свою очередь, перейдут на следующий этап. Выбор особей производится на основе оценочной *фитнесс-функции*. Процесс порождения эволюций завершается при получении особи (нейронной сети) с искомыми свойствами. Выделяют два подхода к кодированию нейронных сетей: *прямое кодирование* (*direct encoding*) и *порождающее кодирование* (*recipe encoding*) [14].

Эволюционный подход обеспечивает нам глобальный охват пространства поиска; очевидно, в процесс поиска будут вовлекаться как простые, так и сложные архитектуры. Как было показано выше, сложность модели будет существенно влиять на точность построенной оценки $\hat{f}(x)$. Для достижения наилучшего результата и избежания эффекта *перенастройки* (*overfitting*) способы задания фитнесс-функций предполагают наличие параметров, регулирующих размеры сети, поощряющих получение наименьшего адекватного задаче варианта сети. Необходимо отметить, что критерии оценки, используемые в конструктивных алгоритмах, зачастую не принимает во внимание сложность модели.

К положительным сторонам глобального эволюционного поиска нейронных сетей можно отнести охват всех классов архитектур A, а не только смежных. Поиск по всему пространству дает большие шансы для нахождения глобального минимума оценочной функции. Кроме этого, способ организации работы генетических алгоритмов дает нам возможность находить решение при произвольно сложном виде оценочной фитнесс-функции.

Основным недостатком эволюционного поиска можно считать очень длительное время нахождения решения. Это объясняется тем, что в процессе поиска оптимального решения на каждом этапе эволюции производится сначала построение, а затем перебор большого числа индивидуумов. В некоторых гене-

тических алгоритмах для уменьшения времени вычислений искусственно сужается пространство поиска — в генетической строке ограничивается размер представимой нейронной сети. Это делает алгоритмы эволюционного получения нейронных сетей оптимальной топологии малопривлекательными для задач, требующих получения сложных нейронных сетей.

Оба приведенных выше подхода к автоматическому построению нейронных сетей имеют свои достоинства и свои недостатки. Конструктивные алгоритмы за короткий срок получают субоптимальное решение, а эволюционный поиск находит, как правило, лучший результат, но за более длительное время.

2. Предлагаемая организация поиска архитектуры. Стремясь объединить лучшие стороны этих подходов, мы постарались уйти от ограничений, связанных с жесткой итеративной схемой построения, и от больших временных затрат, вызванных охватом сразу всего пространства. Практические исследования нейронных сетей показывают, что в условиях зашумленных наборов данных ограниченной размерности совершенно различные нейронные сети обеспечивают для многих задач практически одинаковые результаты [3]. Таким образом, при поиске архитектуры нейронной сети нам не обязательно охватывать все пространство целиком. Для нахождения приемлемого результата достаточно будет реализовать стратегию перемещения по пространству небольшими скачками с подробным поиском в пределах некоторой области.

В своей работе мы предлагаем следующую стратегию организации поиска архитектуры. Основополагающим остается принцип, согласно которому сложность нейронной сети в процессе поиска итеративно увеличивается. Изменения, которые необходимо внести в первоначальную модель, определяются при помощи генетических алгоритмов. Для этого нами разработан специальный способ кодировки архитектуры нейронной сети в генетическую строку, учитывающий наличие постоянной части сети.

Формально эта процедура будет выглядеть следующим образом. Имеется некоторая базовая нейронная сеть с архитектурой A. На каждом этапе алгоритма мы будем получать новую нейронную сеть $\tilde{A}$ путем добавления к сети некоторого количества нейронов ($\gamma_\Delta > 1$). Способ добавления допускает образование как новых уровней L_Δ, так и дополнительных связей p_Δ. Фрагмент Δ будем называть *наращивающим блоком* нейронной сети. Способ добавления достраивающего блока Δ мы будем определять при помощи эволюционной процедуры

$$\tilde{A} = A + \Delta. \tag{6}$$

С другой стороны, на данный способ организации поиска можно взглянуть как на разновидность конструктивного алгоритма. Здесь фактически реализован аналогичный итеративный принцип построения сети за единственным исключением: вместо жесткого способа построения сети, например каскадированием или по слоям, способ надстройки на каждом этапе будет выбираться произвольно в зависимости от потребности задачи. Необходимость выбора будет оцениваться специальной эволюционной процедурой, реализующей целенаправленный перебор и оценку вариантов. Более того, условие возможного добавления сразу нескольких нейронов допускает реализацию одновременно и каскадного, и послойного наращивания.

Следует заметить, что многие эффективные алгоритмы конструктивного направления для сглаживания ограничений, связанных с жестким способом построения, реализуют переборную стратегию в неявном виде. Так, во всех модификациях алгоритма каскадной корреляции для выбора нейронов-кандидатов используется так называемый пул из нескольких нейронов. Нейроны всего пула поочередно оцениваются; из всего пула выбирается тот, который будет влиять на ошибку сети больше всех.

Как было сказано выше, эволюционная процедура, предлагаемая в данном подходе, имеет специальный способ кодировки архитектуры нейронной сети в генетическую строку, учитывающий наличие постоянной части сети. Фактически это будет означать разделение генетической строки на *постоянную и варьируемую* части. В терминах *теоремы шим* [11] положительный эффект от подобной организации можно обосновать следующим образом.

Как известно, по мере сходимости эволюционной процедуры алгоритм формирует генетическую строку, отдельные участки которой изменяются все реже и реже по мере образования строительных шим. Такой эффект свидетельствует о близости конечного решения. Постоянная часть генетической строки будет обусловлена конкретной архитектурой нейронной сети, полученной на предыдущей итерации. Из способа построения предполагается, что сформированная подобным образом постоянная часть даст лучшие частичные решения для решаемой нейронной сетью задачи. Можно считать, что постоянная часть нашей генетической строки фактически является строительной шимой, зафиксированной принудительно. Таким образом, скорость сходимости генетической процедуры значительно увеличивается. Необходимо добавить, что при этом также сокращается длина генетической строки, что дополнительно повышает скорость работы.

Организация постоянной и переменной частей переложена на разработанный нами метод кодирования

нейронной сети в генетическую строку, который подробно рассматривается в следующем разделе. Заметим также, что постоянная часть не будет содержать в себе полное описание архитектуры базовой сети; в ней содержатся только те ее элементы, которые существенны для переменной части.

Выделяя постоянную часть в генетическом коде, мы реализуем следующую, на наш взгляд, более естественную концепцию эволюционного процесса, когда основная часть генетической информации через поколения сохраняется неизменной. Классические генетические операции, такие как кроссинговер и мутация, могут создавать потомков, радикально отличающихся от своих родителей по всему набору признаков. Очевидно, что подобные революционные изменения в генотипе не характерны для живой природы. Если на микроуровне быстрая смена генотипа микроорганизмов еще возможна, то на макроуровне некий базовый набор признаков не изменяется или меняется медленно. Таким образом, выделение постоянной и переменной частей в генетическом коде можно считать более естественным способом для воссоздания эволюционного процесса получения решения.

3. Реализация кодирования путями. Сравнивая порождающий способ кодирования нейронной сети с прямым, мы пришли к выводу, что для предложенной схемы поиска эффективнее было бы использование прямой схемы. Исследования показывают, что порождающее кодирование для ряда задач оказывается значительно менее эффективным [15, 16] и требующим больше времени для достижения результата [17].

Нами использовано развитие варианта прямого кодирования, основанного на путях передачи сигнала в нейронной сети, предложенного в [18]. Данный вариант прямого кодирования предполагает сохранение в генетическом коде не информации о количестве уровней, нейронов или связях в нейронной сети, а информации об условных путях передачи сигнала в сети от входов к выходам.

В нашей работе метод кодировки путями подвергся существенным изменениям применительно к поставленной задаче. Все требования к модифицированному построению нейронной сети путями можно изложить следующим формальным образом.

(1) До начала формирования генетической строки мы имеем некоторую базовую нейронную сеть прямого распространения, состоящую из нейронов $\Gamma = \{h_1, \ldots, h_\gamma\}$ и имеющую k входов и 0 выходов. Зададим некоторое количество *строящих нейронов* $U = \{h_{\gamma+1}, \ldots, h_{\gamma+u}\}$. В формировании путей будут участвовать все нейроны $\{\Gamma, U\}$, то есть новые нейроны будут равноправно участвовать при построении путей наряду с нейронами базовой сети.

(2) Архитектура нейронной сети задается набором множеств путей продвижения сигнала $\{P_i\}$. Путь продвижения сигнала в нейронной сети задается упорядоченным списком нейронов из Γ и U; тем самым моделируется расположение нейронов последовательно по всем слоям, начиная от слоя расположения начального и заканчивая слоем расположения конечного. Каждый путь P может начинаться только с нейрона из набора *допустимых стартовых нейронов* S и может заканчиваться одним нейроном из набора *допустимых выходных нейронов* E. Повторения нейронов в пределах одного пути не допускается. Никакого другого ограничения на порядок нейронов внутри пути не накладывается.

(3) Возможно несколько способов формирования наборов S и E. При практической реализации данного способа кодирования мы остановились на следующем. Набор допустимых стартовых нейронов S формируется из всех нейронов базовой нейронной сети Γ за исключением выходов сети и последнего скрытого слоя. Набор допустимых выходных нейронов E состоит из всех нейронов базовой сети Γ за исключением входов сети и первого скрытого слоя. Естественно, что в оба набора не попадают *строящие нейроны* U.

(4) Непосредственно в генетической строке нейроны представляются своими индексами. Каждый из путей отделяется друг от друга специальным служебным символом $\#$.

Предложенная схема иллюстрируется на рис. 3.

В исходном методе к построенному пути применяются следующие генетические операции: оператор кроссинговера с двумя точками разбиения и четыре варианта оператора мутации [18]. В нашей модификации метода путей для работы с генетическими строками мы использовали следующие генетические операции: *кроссинговер* с одной точкой разбиения и два варианта операции *мутации*. Операция кроссинговера, или пересечения, служит для закрепления полезных для решения поставленной задачи свойств генетической строки. Кроссинговер выбирает случайным образом точку между путями в генетической строке двух индивидуумов. Результатом является два потомка, составленных из путей, принадлежащих обоим родителям (см. рис 4).

Как было уже сказано, мы используем два способа внесения мутаций в генетический код. Первый вариант оператора мутации производит случайным образом замещение нейронов в произвольном числе путей, составляющих генетическую строку индивидуума. Второй вариант оператора мутации предусмат-

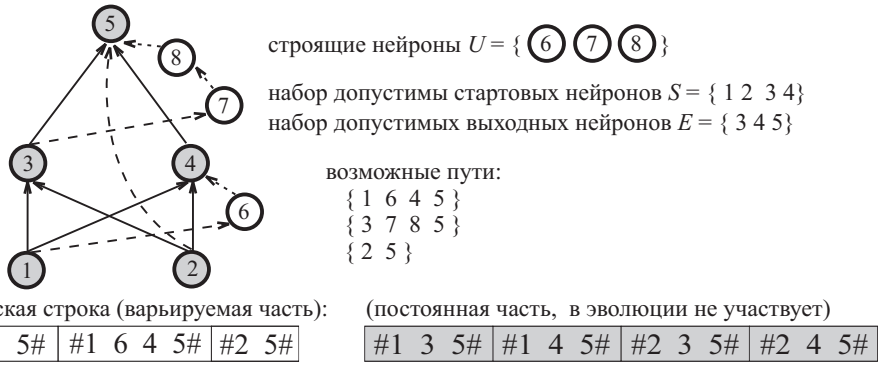

Рис. 3. Предложенный вариант кодировки путями

ривает создание случайным образом генетической строки полностью. Внесение случайных изменений (мутаций) позволяет производить скачки в пространстве архитектур и выбираться из локальных экстремумов, куда нас может привести поиск оптимальной архитектуры. Необходимо отметить, что применение генетических операций может привести к появлению некорректных архитектур для нейронной сети прямого распространения. Распознавание и исправление генетического кода подобных индивидуумов производится на этапе восстановления нейронной сети из генетического кода.

В работе Груо [19] были впервые сформулированы свойства, присущие эффективному методу кодирования нейронной сети. Попытаемся выяснить соответствие предложенной нами схемы кодирования этим основным свойствам.

1) *Полнота* (*completeness*): способ кодирования считается полным по отношению к классу архитектур, если любая возможная архитектура класса может быть представлена этим способом.

Если определить класс возможных архитектур как сети прямого распространения (многослойный персептрон), то предлагаемая здесь кодировка будет обладать свойством полноты в зависимости от способа построения. Всякий раз в качестве исходной мы имеем нейронную сеть прямого распространения. Для преобразования архитектуры этой сети в любую другую нам необходимо увеличивать число слоев и нейронов в слоях, изменять связность нейронов. Для добавления нейронов в любой слой достаточно сформировать путь из двух нейронов: одного из предыдущего, другого из последующего слоев с новым нейроном посредине. Аналогично поступаем для добавления нового слоя, только граничные нейроны берем из смежных слоев. Для создания связи между нейронами достаточно сформировать путь с последовательностью этих нейронов. Если учесть, что исходная нейронная сеть в простейшем виде может не содержать скрытых слоев (т.е. состоять из входов и выходов), можно смело сделать вывод о том, что предлагаемый способ кодировки обладает свойством *полноты*.

2) *Компактность* (*compactness*): способ кодирования A считается более компактным по сравнению с B, если для любой архитектуры сети длина кода A меньше длины кода B.

В качестве меры длины кода естественнее всего использовать количество байт, необходимых для формирования генетической строки. В своей работе [19] Груо в качестве базового способа кодирования B приводит простейший метод прямого кодирования по Миллеру [20]. В этом одном из самых простых способов кодирования все соединения сети из γ нейронов кодируются матрицей связности нейронов размерностью $\gamma * (\gamma + 1)$. Покажем, что методика, предложенная нами, будет генерировать генетический код меньшей длины.

Рассмотрим представление полносвязной нейронной сети при помощи путей. Для каждого нейрона нам потребуется составить цепочки, определяющие связность с прочими нейронами. В зависимости от способа построения, в цепочку будут попадать только нейроны, действительно связанные с данным. Причем отсутствующие и заведомо некорректные обратные связи не будут присутствовать совсем. Более того, за счет построения длинных путей будет экономиться по байту для каждого промежуточного нейрона. Кроме этого, в зависимости от количества цепочек потребуется некоторое количество байт для обозначения служебного символа-разделителя # по числу цепочек. Для прямого кодирования по Миллеру в матрице будет фиксироваться место для *всех* возможных связей, в том числе и некорректных. Из сказанного можно сделать вывод о том, что генетическая строка, сформированная при помощи кодировки путями, будет короче матричной формы кодирования. Таким образом, можно утверждать, что эта кодировка обладает свойством *компактности* по отношению к простейшему методу прямого кодирования по Миллеру.

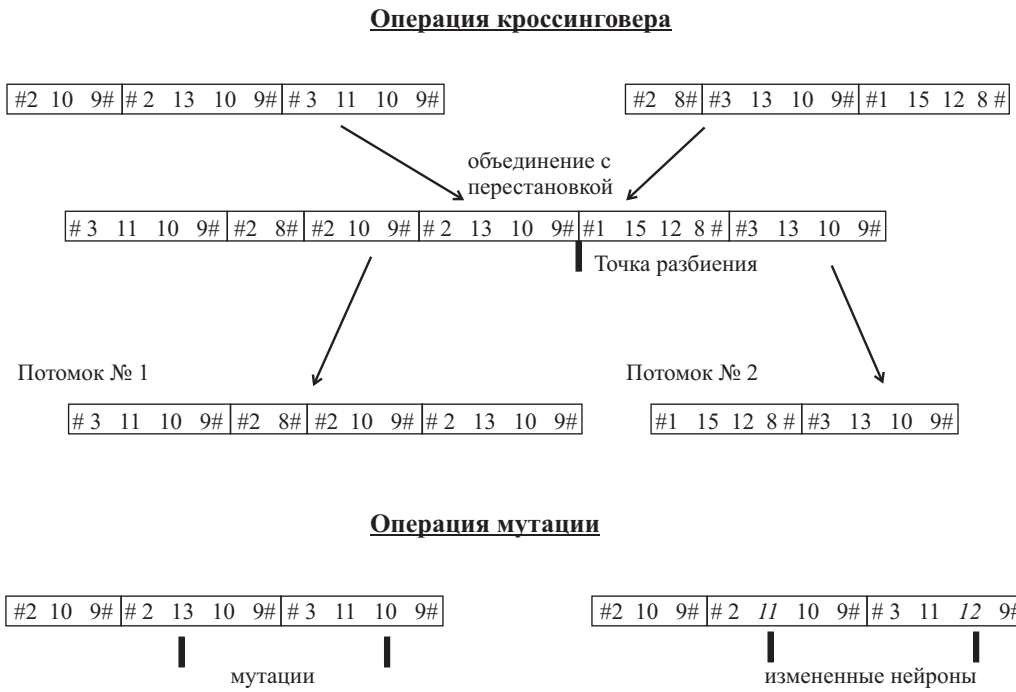

Рис. 4. Генетические операции

3) *Замкнутость* (*closure*): это свойство в отношении класса архитектур нейронных сетей существует тогда, когда любая возможная строка кода восстанавливается в архитектуру данного класса.

Данное свойство можно рассматривать как некоторую идеализацию по отношению к методам кодировки. Дело в том, что даже если способ кодировки исключает возможность создания некорректной архитектуры, то использование операторов мутации (и в некоторых случаях кроссинговера) может подвергнуть генетическую строку преобразованиям, в результате которых восстановить можно будет только некорректную нейронную сеть (например, с обратными связями для сетей прямого распространения или сеть, состоящую из нескольких несвязных частей). Для большей части методов кодирования можно говорить в лучшем случае лишь о частичном соответствии свойству замкнутости. Корректность работы алгоритмов генетического поиска поддерживается путем отбрасывания “уродов” на этапе восстановления нейронной сети.

Кодировка путями может гарантированно строить нейронные сети без разделения на несколько несвязных компонент и без “висячих” скрытых нейронов. Исходя из алгоритма формирования путей в соответствии с правилом (3) на стр. 93, когда начальный и конечный нейроны пути берутся из специально сформированных наборов, можно всегда гарантировать отсутствие “висячих” вершин, т.е. нейронов на скрытых уровнях, не передающих сигнал на другие нейроны или выходы сети. Генетические операции мутации реализованы таким образом, что мутационные изменения производятся также в соответствии с данным правилом. Кроссинговер же совсем не производит манипуляций над содержимым путей.

Сохранение связности закодированных сетей обеспечивают правила (1) и (3) на стр. 93. Всякий раз мы формируем цепочку относительно некоторой корректной нейронной сети; создание цепочки, не имеющей общих нейронов с базовой сетью, невозможно. Генетические операции, также не будут разделять нейронную сеть.

Предложенная кодировка не ограничивает направления движения сигнала на “только вперед”; так как мы ограничились в наших исследованиях нейронными сетями прямого распространения, то существования обратных связей в сети нам неприемлемо. Мы посчитали неэффективным введение механизма, ограничивающего обратные связи. Сложность кодировки при этом существенно возросла бы. Гораздо удобнее отсекать такие связи на этапе восстановления нейронной сети и передавать в этап эволюционного уже корректный генетический код.

Таким образом, с приведенными оговорками можно считать, что предлагаемый способ кодировки обладает свойством *замкнутости*.

4. Фитнесс-функция. В генетических алгоритмах под “приспособленностью” (fitness) отдельного

индивидуума популяции к среде рассматривается его способность к эффективной реализации некоторой заданной целевой функции (построение эффективной нейронной сети в нашем случае). Существует множество вариантов задания фитнесс-функции для построения нейронных сетей эволюционными методами. Далее мы предлагаем свой вариант определения фитнесс-функции. По форме наше задание фитнесс-функции попадает под общий принцип, известный как *принцип описания минимальной длины* [24]. В своих исследованиях мы пользовались фитнесс-функцией F следующего вида:

$$F = \frac{1}{E_{\text{Train}}} + \frac{1}{E_{\text{Valid}}} + C + S. \tag{7}$$

Здесь E_{Train} — среднеквадратическая ошибка сети, восстановленной из генетической строки, которая задана на обучающей выборке; E_{Valid} — среднеквадратическая ошибка сети на проверочной выборке; C — оценка сложности топологии нейронной сети; S — скорость уменьшения ошибки нейронной сети на проверочных данных.

Величина C служит для определения сложности топологии сети и оценивается по следующей эмпирической формуле:

$$C = \begin{cases} F_1(\dfrac{1}{N} + c_1\dfrac{1}{\sigma}), & \sigma > 0, \\ F_1(\dfrac{1}{N} + c_1), & \sigma \leqslant 0. \end{cases} \tag{8}$$

Здесь N — общее число нейронов в сети, $\sigma = \left[\frac{1}{N}\sum\limits_{i=1}^{N}(n_i - \overline{n_i})\right]^{1/2}$ — стандартное отклонение, характеризующее степень разброса количества нейронов в скрытых слоях сети от некоторого среднего числа нейронов $\overline{n_i}$ по слоям; n_i — число нейронов в скрытом слое i. Согласно (8), большую величину C будут иметь нейронные сети, которые имеют малое число нейронов, расположенных максимально равномерно по скрытым слоям. Именно последнее слагаемое в C способствует созданию нейронных сетей без резких отличий числа нейронов на скрытых слоях.

Скорость уменьшения ошибки сети на проверочных данных S определяется в (9) и характеризует архитектуру нейронной сети:

$$\frac{\sum\limits_{i=1}^{\delta}\left(E_{t-\delta_{i+1}} - E_{t-\delta_i}\right)^2}{\delta} < \Omega, \quad t \geqslant t_0, \quad \delta_i \in \overline{1,\delta}. \tag{9}$$

Здесь $E_{t-\delta_{i+1}}$ и $E_{t-\delta_i}$ — значения ошибок на интервале обучения δ; Ω — некоторая заданная минимально возможная скорость изменения ошибки, при превышении которой принимается решение об изменении сети; δ — количество эпох-циклов обучения сети перед выполнением оценки скорости ошибки.

Можно считать, что сложность сети не соответствует поставленной задаче, если ошибка в числе эпох обучения уменьшается недостаточно быстро либо не уменьшается совсем. Способ оценки скорости взят из предложенного нами ранее метода мониторинга динамики изменения ошибки [25]. При данной оценке скорости ошибки принимается во внимание динамика изменения ошибки на некотором интервале обучения δ. Здесь $E_{t-\delta_i}$ представляют собой значения ошибки на интервале обучения δ в момент времени i. Заметим, что при оценке скорости по (9) учитывается возможность наличия периодов временного роста ошибки на интервале δ, причем за счет эффекта усреднения отсекаются локальные провалы производительности, вызванные временным замедлением скорости уменьшения ошибки.

5. Алгоритм эволюционного накопления признаков. Выполненный анализ организации поиска нейронной сети, а также способов кодирования и реализации фитнесс-функции положен в основу предлагаемого в нашей работе нового алгоритма для построения нейронных сетей, названного *алгоритмом эволюционного накопления признаков* (ЭНП) [26]. Опишем формально схему работы алгоритма.

Основная идея алгоритма эволюционного наращивания нейронной сети заключается в том, чтобы строить нейронную сеть в процессе обучения. Сеть строится на основе исходной сети, состоящей в общем виде только из входов и выходов. Сеть достраивается новыми нейронами и связями как между существующими, так и новыми нейронами. В качестве новых добавочных блоков нейронов используются подсети нейронов, полученные эволюционным способом. Подсеть выбирается эволюционным способом таким образом, чтобы уменьшить общую ошибку нейронной сети на предложенной обучающей выборке. Алгоритм можно условно разбить на два этапа. Основной этап работы алгоритма заключается в реализации механизма поиска наращивающих блоков. Второй этап алгоритма — надстройка нейронной сети и проверка эффективности ее работы. Рассмотрим алгоритм эволюционного наращивания нейронной сети по этапам.

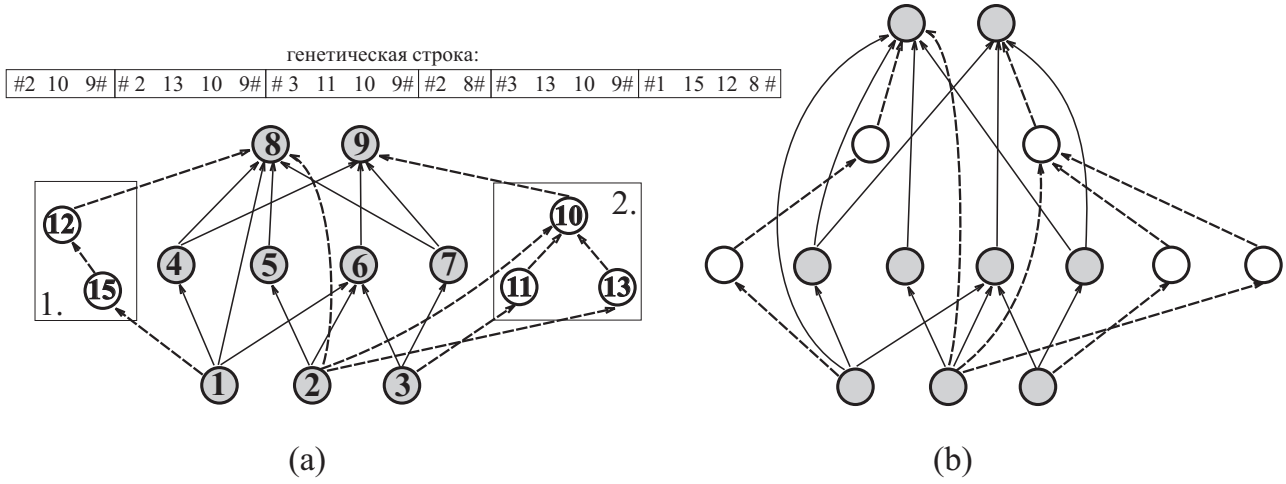

Рис. 5. Схема алгоритма *эволюционного накопления признаков*

На этапе инициализации эволюционного поиска предполагается, что у нас уже имеется некоторая нейронная сеть, которую необходимо улучшить. В начале работы алгоритма берется сеть, состоящая только из входов и выходов — без скрытого слоя нейронов. Задается некоторое максимально возможное количество нейронов, которые разрешено добавить к сети в результате эволюционного поиска. На основе нейронов этой сети и полученной информации о дополнительных нейронах случайным образом генерируется некоторая начальная популяция. Затем выполняется этап генетического поиска: для каждого индивидуума из текущей популяции производится вычисление оценочной фитнесс-функции. Данная функция в нашем случае будет определять эффективность и точность работы нейронной сети, восстановленной из генетической строки при решении поставленной задачи. Все индивидуумы ранжируются в соответствии со своим значением фитнесс-функции. В нашей работе для определения лучших представителей популяции мы применяли так называемый метод "колеса рулетки". Для получения нового поколения, генетически более приспособленного к среде, мы применяем к выбранным индивидуумам генетические операции мутации и кроссинговера. После этого опять производится вычисление фитнесс-функции и выбор новых кандидатов в родители. Весь процесс эволюционного порождения останавливается, когда не происходит существенного увеличения величины лучшего значения фитнесс-функции для популяции. На этом эволюционный этап работы алгоритма заканчивает свою работу. Среди лучших представителей популяции выбирается тот, который даст способ наращивания нейронной сети. При извлечении информации из данной генетической строки получаются два наращивающих блока (рис. 5 (a)). После нахождения строительных блоков для нейронной сети эволюционный этап считается завершенным.

Надстройка нейронной сети является очевидным следующим шагом, логично завершающим этап эволюционного поиска. Строительные блоки, полученные на предыдущем шаге, добавляются к сети. На рис. 5 (a) показана фаза добавления полученных надстроечных блоков 1 и 2 к текущей нейронной сети. Светлым цветом выделены новые добавленные нейроны, пунктирной линией отмечены новые связи между нейронами. Хотелось бы отметить, что в принципе добавление новых нейронов при наращивании сети не обязательно. Нейронная сеть может развиваться только за счет добавления новых синаптических связей между существующими нейронами.

На рис. 5 (b) показана получившаяся в результате модификации новая нейронная сеть. Полученную нейронную сеть мы проверяем на обучающей и на проверочной выборках. В случае если значение ошибки работы сети нас устраивает, работа всего алгоритма считается завершенной, а текущая сеть считается искомой нейронной сети оптимальной архитектуры для данной задачи. В противном случае мы рассматриваем полученную сеть как базу для наращивания и передаем управление обратно на эволюционный этап.

Предложенный нами алгоритм сочетает в себе лучшие качества конструктивного и эволюционного подходов. От конструктивного направления взят принцип, согласно которому сложность нейронной сети в процессе поиска итеративно увеличивается. Изменения, которые необходимо внести в первоначальную модель, определяются при помощи генетических алгоритмов. Для этого разработан специальный способ кодировки архитектуры нейронной сети в генетическую строку, учитывающий наличие постоянной части

сети.

Представленный алгоритм (за счет применения генетических алгоритмов) позволяет реализовать более широкий охват пространства архитектур по сравнению с конструктивным направлением. Результатом будет улучшенное качество решения. Итеративный принцип увеличения сложности позволяет значительно уменьшить временные затраты сравнительно с большей частью методов построения нейронных сетей эволюционного направления.

Как способ дальнейшего развития предложенного алгоритма эволюционного наращивания нейронной сети можно использовать распараллеливание эволюционного этапа алгоритма на системах многопроцессорной обработки информации. Сделать это позволяет наличие нескольких индивидуумов в популяции и независимая реализация вычисления фитнесс-функции каждого индивидуума. В дополнение к этому можно применить к построенной нашим алгоритмом нейронной сети любой из эффективных методов усечения.

6. Исследование алгоритма. В настоящее время в области, связанной с исследованиями искусственных нейронных сетей, сложилась практика использования некоторых стандартных наборов тестовых проблем и методик для сравнения тех или иных характеристик различных алгоритмов и методов тренировки нейронных сетей. Часть из этих тестовых проблем составляют искусственно созданные наборы данных — *синтетические тесты*. Для того чтобы наиболее полно и точно исследовать свойства предлагаемого алгоритма, нами был использован сложный синтетический тест *"проблема двух спиралей"*, часто используемый авторами для оценки работы различных алгоритмов построения нейронных сетей [7, 27, 28].

Помимо этого мы в нашей работе будем использовать тесты из коллекции тестовых задач PROBEN1, основанных на данных, взятых из реального мира. Эта коллекция была сформирована Л. Пречелтом (L.Prechelt) в [29] из нескольких типичных задач, для решения которых применяются нейронные сети. Она содержит 14 наборов данных из 12 различных областей. Тесты из PROBEN1 широко используются нейросетевым сообществом для сравнения характеристик алгоритмов построения нейронных сетей. В частности, имеются публикации по исследованиям одного из самых эффективных алгоритмов — алгоритма каскадной корреляции и его разнообразных вариаций [28, 30 – 34]. Существуют данные о сравнительных характеристиках и для других основных методов эволюционного поиска нейронных сетей [20, 35 – 40]. Ориентируясь на существующие публикации, мы отобрали для тестирования нашего алгоритма и сравнения с прочими шесть различных по сложности наборов данных.

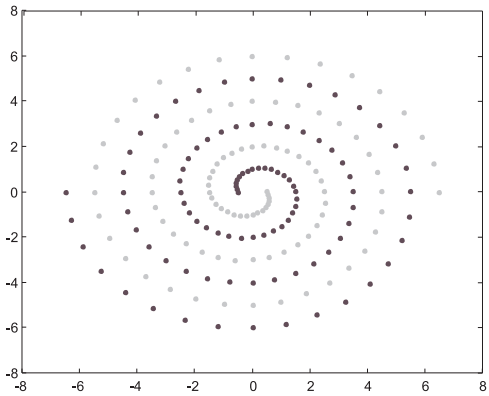

Рис. 6. Обучающий набор данных для проблемы двух спиралей

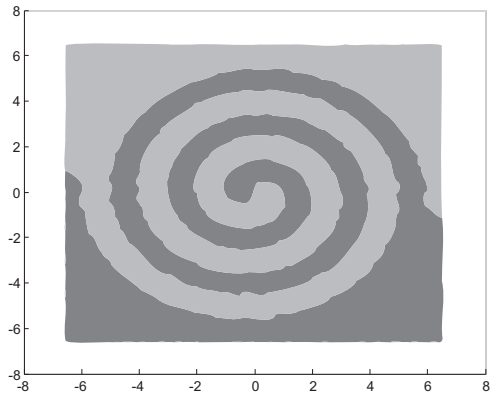

Рис. 7. Тестовый набор для проблемы двух спиралей

Проблема *двух спиралей* (*the Two Spirals test*) была впервые предложена А. Виеландом (A.Wieland) из корпорации MITRE как довольно сложная проблема для нейронных сетей. Набор данных был взят без изменений в том виде, в котором он был предложен в работе [28]. Обучающий набор данных состоит из 194 значений (x, y), половина из которых определяет точки одной спирали, а другая половина — точки второй спирали. Спирали плотно переплетаются друг с другом, как показано на рис. 6. Функцией нейронной сети будет задача классификации, состоящая в том, чтобы научиться различать принадлежность точек плоскости одной из спиралей. Сложность задачи двух спиралей определяется тем, что для подобных данных граница решения, разделяющая представителей разных классов, будет иметь сложный вид и для ее построения потребуется, как минимум, многослойная нейронная сеть.

Для оценки работы обученной нейронной сети была использована тестовая выборка из 17 161 точки (см. рис. 7), равномерно распределенных на плоскости в интервале $[-6.5, 6.5]$. Точки тестового набора

были классифицированы с использованием алгоритма ближайшего соседа с обучающим набором в качестве образца. Проверочная выборка данных была составлена из двух частей. Первая часть состоит из 192 точек, задающих спирали, слегка повернутых относительно исходных спиралей. Вторая часть проверочного набора данных включает в себя 200 точек, сгенерированных случайным образом и классифицированных с использованием того же метода случайного соседа. Все три набора данных доступны по адресу http://www.cse.unsw.edu.au/-nickt/datasets/ts.net.gz.

Проведем сравнительный анализ полученных результатов для предложенного алгоритма и известных из публикаций данных о прочих алгоритмах и методах построения нейронных сетей. Для проблемы двух спиралей мы имеем данные для исходного алгоритма каскадной корреляции [7] и алгоритма каскадной корреляции с ограничением числа слоев [27], а также для семейства алгоритмов КасПер [28]. Из эволюционного направления к построению нейронных сетей был выбран самый известный метод клеточного кодирования (cellular encoding) [41].

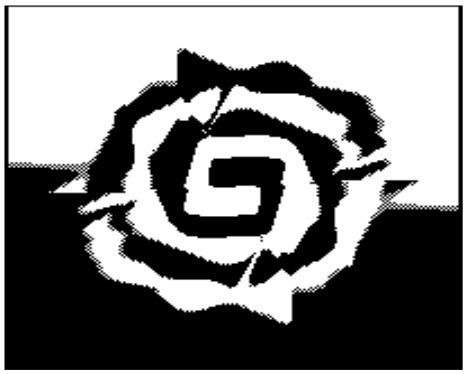

Рис. 8. Результат работы сети, полученной методом каскадной-корреляции, приведенный в работе [7]

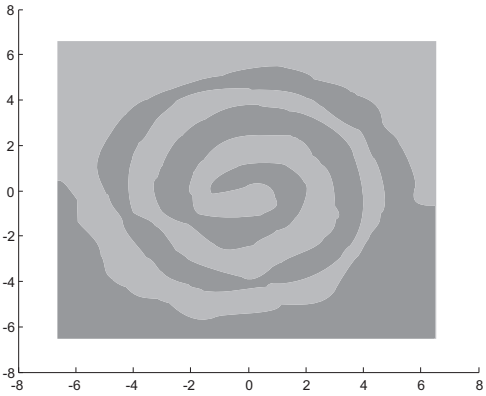

Рис. 9. Результат работы сети на проверочной выборке для ЭНП

Таблица 1

Сводная таблица по всем алгоритмам для проблемы двух спиралей

Алгоритм	**запусков**	**ОК тестовая**	**Количество нейронов**	**Число слоев**	**Степень компактности**
эволюционное наращивание (ЭН)	50	14,89%	23,88	3,68	6,49
каскадная-корреляция(КК)	100	>20%	15,2	15,2	1,00
КК с ограничением числа слоев	100	–	34,87	6,37	5,47
КасПер (КП)	50	15,50%	12	12	1,00
АКасПер	50	16,40%	12	12	1,00
Метод клеточного кодирования	1	42,70%	18	6	3,00
метод победитель	–	ЭН	КП	ЭН	ЭН

В работе [7] приведены данные о том, что в построенной методом каскадной корреляции нейронной сети число нейронов будет варьироваться от 12 до 19 и в среднем будет состоять из 15.2 нейронов для 100 запусков. Исходя из способа построения “один нейрон в слое”, мы будем иметь число слоев, равное числу нейронов. К сожалению, в [7] не приведены результаты проверки работы построенной нейронной сети на независимой тестовой выборке, но зато имеется рисунок, иллюстрирующий работу сети с 12 нейронами (рис. 8). Легко видеть, что эта сеть гладко интерполирует первые 1.5 оборота спиралей, но дальше, с расхождением точек спирали, становится неровной и смазанной. Это свидетельствует о том, что для данной сети процент ошибки классификации будет довольно высок.

Опираясь на полученные данные, можно с уверенностью сказать, что предложенный алгоритм эволюционного накопления признаков превосходит исходный алгоритм каскадной корреляции. Существенное

Таблица 2

Основные характеристики наборов данных PROBEN1

Тестовая проблема из набора PROBEN1	Количество		Число примеров наборов		
	входов	выходов	обучающего	проверочного	тестового
Cancer1	9	2	350	175	174
Card1	51	2	345	173	172
Diabetes3	8	2	384	192	192
Glass1a	9	6	107	54	53
Heart1a	35	2	460	230	230
Thyroid2	21	3	3600	1800	1800

преимущество достигается по двум основным характеристикам: обобщающей способности сети (ошибка классификации равна 11 % на проверочной выборке) и компактности построения сети (рис. 9). Метод эволюционного накопления признаков дает более компактные нейронные сети, проигрывая по среднему числу нейронов; однако реализуется существенно меньшее число слоев (в среднем 3.68) по сравнению с каскадной структурой из 15.2 слоев по одному нейрону.

Для алгоритма каскадной корреляции с ограничением числа слоев в работе [27] были получены следующие результаты, усредненные по 100 запускам: число нейронов сети в среднем 34.87 с минимумом, равным 27, среднее число слоев 6.37 с минимумом, равным 5. В качестве ошибки классификации на тестовой выборке приведено среднее значение, равное 7.3 %, но не приведена ни сама выборка, ни ее объем. Значительный отрыв значения ошибки на тестовой выборке среди *всех* рассмотренных конструктивных алгоритмов, примененных для проблемы двух спиралей, позволяет говорить о том, что объем тестовой выборки был, по-видимому, сильно уменьшен, поэтому в данном отношении алгоритм невозможно сравнивать с остальными.

Большего внимания среди всех конструктивных алгоритмов заслуживает семейство алгоритмов КасПер. В [28] исследованы несколько вариантов алгоритма КасПер для проблемы двух спиралей и несколько задач из PROBEN1. Если сравнивать полученные результаты с известными из [28] данными по алгоритму КасПер, можно видеть, что предложенный алгоритм эволюционного накопления признаков дает в среднем меньший процент ошибки (на 0.6 %), а минимальное значение ошибки на тестовой выборке 10.91 % против 14.6 %. С точки зрения компактности построенной нейронной сети алгоритму КасПер свойственны те же недостатки, которые присущи алгоритмам, основанным на каскадной корреляции — большое число слоев сети только с одним нейроном в каждом. Следовательно, по степени компактности сети КасПер уступает алгоритму эволюционного наращивания, но превосходит его по общему числу нейронов.

Для эволюционного направления нам доступны результаты решения проблемы двух спиралей одним из расширений широко распространенного метода — метода клеточного кодирования [37, 38]. Стандартный набор команд клеточного кодирования был расширен несколькими специальными командами. Лучшая нейронная сеть, полученная модифицированным методом клеточного кодирования, имеет 18 нейронов, расположенных в 6 слоях [38]. Статистика по нескольким запускам алгоритма отсутствует. Ошибка такой сети на тестовой выборке всего из 584 точек была довольно высока — 42.7 %.

В качестве лучшей сети, полученной методом эволюционного накопления признаков, можно выбрать сеть, для которой 17 нейронов в 3 слоях дают нам ошибку в 15.32 % для тестовой выборки из 17 161 точки. Можно видеть, что разработанный нами алгоритм в ряде случаев может создавать нейронные сети с лучшими характеристиками, чем сети, создаваемые алгоритмом КасПер.

Суммируя информацию по результатам сравнения предложенного алгоритма эволюционного накопления признаков и всех выше приведенных алгоритмов, можно сделать вывод о том, что разработанный нами алгоритм показал себя лучше прочих для довольно сложной проблемы двух спиралей, а именно создавал более компактные нейронные сети с лучшей обобщающей способностью (табл.1).

7. Практические задачи из набора PROBEN1. PROBEN1 — это коллекция задач, предназначенных специально для тестирования нейронных сетей на реальных данных, плюс набор правил и соглашений о единообразии проведения тестирования и предоставления результатов. В целом PROBEN1 состоит из 10 задач классификации и 4 задач интерполяции. В большей части тестовых наборов имеются пропущенные

Таблица 3

Результаты по всем алгоритмам построения нейронных сетей для задачи *Cancer1* из набора PROBEN1. Пояснения 1) – 4) см. на стр. 103

	название алгоритма	построение сети	лучший вариант	запусков	ошибка классификации, %						число		структура сети по слоям	CPU - время, чч:мм	число эволюций	размер популяции
					обучение	std. откл	проверка	std. откл	тестовая	std. откл	нейронов	слоев				
	Proben1	ручн.		50	2,87	0,27	1,96	0,25	1,47	0,60	6,00	2,00	**4-2**	–		
1	*предлагаемый метод ЭННС*	эвол.		50	0,90	0,28	1,64	0,47	1,73	0,64	7,40	1,50	**7**	00:09	5	30
2	*Миллер*	эвол.	да	30	3,54	0,50	2,76	0,58	2,01	0,68	12,00	7,00	**1-2-1-2-2-3-1**	1)	100	50
3	*Китано*	эвол.	да	30	2,56	0,39	3,03	0,78	1,99	0,56	19,00	17,00	**1(10)-2-2-1(5)**	1)	100	50
4	*Нолфи*	эвол.	да	30	3,80	0,41	1,73	0,42	2,32	0,54	12,00	4,00	**4-2-1-5**	1)	100	50
5	*Канджелози*	эвол.	да	30	3,51	0,21	2,86	0,15	2,70	0,66	11,00	4,00	**3-1-3-4**	1)	100	50
6	*Фридрих*	эвол.	да	1	–	–	–	–	–	–	20,00	5,00	**6-6-4-1-3**	12:00 2)	100	50
7	*АКасПер*	конст.		50	–	–	–	–	1,89	0,80	4,86	4,86	**1-1-1-1**	–		
8	*КасПер*	конст.		50	–	–	–	–	1,95	0,38	5,18	5,18	**1-1-1-1-1**	–		
9	*SCGA*	эвол.	да	1	–	–	–	–	7,50	–	13,00	2,00	**6-7**	–	70	20
10	*MBP* 3)	усеч.	да	11	–	–	–	–	2,30	–	15,00	2,00	**7-8**	–		
11	*OBS* 3)	усеч.	да	11	–	–	–	–	2,90	–	11,00	2,00	**7-4**	–		
12	*unit-OBS* 3)	усеч.	да	1	–	–	–	–	2,30	–	7,00	2,00	**5-2**	–		
13	*ENZO* 3)	эв.ус.	да	1	–	–	–	–	1,70	–	9,00	2,00	**5-4**	–	–	30
14	*ENZO-MI* 3)	эв.ус.	да	1	–	–	–	–	1,10	–	7,00	2,00	**5-2**	–	–	30
15	*Cand*	конст.		42	2,52	0,35	2,21	0,21	1,44	0,65	15,76	1,00	**16**	232 4)		
16	*Cascade*	конст.		42	2,75	0,45	2,24	0,29	1,60	0,52	15,29	15,29	**1(15)**	561 4)		
17	*Cascor*	конст.		45	2,35	0,14	2,20	0,09	1,95	0,38	5,18	5,18	**1-1-1-1-1**	532 4)		
18	*Kogi2*	конст.		42	2,83	0,45	2,19	0,28	1,72	0,65	15,48	15,48	**1(15)**	553 4)		
19	*Kogi3*	конст.		39	2,52	1,04	2,20	0,39	1,61	0,69	17,62	1,00	**18**	842 4)		
20	*Kogi9*	конст.		41	2,92	0,68	2,20	0,36	1,58	0,76	11,50	2,00	**7-4**	627 4)		
21	*Lprune*	усеч.		30	2,59	0,15	1,80	0,19	1,49	0,58	2,93	2,00	**2-1**	309 4)		
22	*Tprune*	усеч.		30	2,80	0,15	1,89	0,22	1,57	0,71	3,30	2,00	**2-1**	210 4)		

данные. Для всех наборов данных значения векторов входов и выходов нормализованы к интервалу $[0, 1]$. Для классификационных задач с несколькими классами C выходные значения кодируются C выходами по схеме 1 из C, когда выход, соответствующий классу принадлежности, устанавливается равным 1, а прочие 0. Данные набора PROBEN1 см. в [29]. В табл. 2 приведены основные характеристики наборов данных, используемых для исследования.

Помимо создания базы проверочных наборов, Л. Пречелт в [29] приводит подробнейшую информацию об исследованиях работы нейронных сетей на данных из PROBEN1. Для каждого теста по 50 запускам мы имеем усредненную информацию об архитектуре нейронной сети, полученной методом эмпирического подбора, лучшим образом решающей данную проблему, и о различных характеристиках сети.

Задачи из набора PROBEN1 широко используются для проверки алгоритмов и методов работы с нейронными сетями. По результатам исследований приведенных в табл. 2 задач имеется обширнейшая коллекция работ, которая охватывает все основные методы: 15 конструктивных и 8 эволюционных алгоритмов построения нейронных сетей.

Конструктивные методы построения представлены алгоритмом каскадной корреляции (*Cascor*) с пятью модификациями (*CasCade, Cand, Kogi2, Kogi3, Kogi9*) [30], а также двумя версиями алгоритма КасПер: *CasPer* и *aCasPer* [28]. Методы усечения (*усечение меньшего веса* (*magnitude based pruning, MBP*), *OBS*, *unit-OBS*, *ENZO* и *ENZO-MI*) исследованы в [40]. Алгоритмы *ENZO* и *ENZO-MI* реализуют усечение весов и нейронов с использованием генетических алгоритмов. Алгоритм усечения *Tprune* описан в [42]; он реализует усечение 35 % всех соединений на первоначальном этапе работы по 10 % оставшихся весов на каждом последующем шаге. Алгоритм усечения *Lprune* является модификацией *Tprune*: в нем заложены адаптационные правила, автоматически определяющие процент весов для усечения [30].

Эволюционный подход к построению нейронной сети представлен алгоритмом прямого кодирования нейронной сети *BP-Generator* [35] и алгоритмом прямой кодировки по *Миллеру* [20]. Порождающее кодирование представлено алгоритмами *Semantic Changing Genetic Algorithm* (*SCGA*) [36], двумя вариантами метода *клеточного кодирования* (*cellular encoding*): Фридриха [37, 38] и *ANKKA* [39], алгоритмом, реа-

Таблица 4

Результаты по всем алгоритмам построения нейронных сетей для задачи *Card1* из набора PROBEN1. Пояснения 4) и 6) приведены в тексте (см. стр. 103)

	название алгоритма	построение сети	лучший вариант	запусков	ошибка классификации, %						число		структура сети по слоям	CPU - время, чч:мм	число эволюций	размер популяции
					обучение	стд. откл	проверка	стд. откл	тестовая	стд. откл	нейронов	слоев				
	Proben1	ручн.		50	8,92	0,54	8,89	0,59	13,64	0,85	32,00	1,00	**32**	–		
1	*предлагаемый метод ЭННС*	эвол.		50	6,18	2,16	12,77	2,25	15,70	2,59	8,32	1,56	**8**	01:02	5	30
2	*АКасПер*	конст.		50	–	–	–	–	13,72	0,59	0,12	0,12	**0**	–		
3	*КасПер*	конст.		50	–	–	–	–	13,58	0,43	1,07	1,07	**1**	–		
4	*ANKKA* [6]	эвол.	да	1	–	–	–	–	12,80	–	14,00	11,00	**2-1-1-2-1-1-2-1(4)**	–	100	50
5	*Cand*	конст.		42	8,91	0,97	8,78	0,11	13,52	0,60	1,55	1,00	**1**	145[4]		
6	*Cascade*	конст.		42	9,11	0,43	8,78	0,14	13,58	0,56	1,45	1,45	**1**	108[4]		
7	*Cascor*	конст.		45	8,69	0,29	8,81	0,15	13,58	0,43	1,07	1,07	**1**	437[4]		
8	*Kogi2*	конст.		42	9,18	0,54	8,81	0,13	13,61	0,48	1,60	1,60	**1-1**	157[4]		
9	*Kogi3*	конст.		39	8,77	0,86	8,78	0,15	13,80	0,56	1,67	1,00	**2**	96[4]		
10	*Kogi9*	конст.		41	8,89	0,56	8,77	0,13	13,88	0,64	1,21	2,00	**1-1**	80[4]		
11	*Lprune*	усеч.		30	7,39	0,54	8,27	0,21	13,51	0,90	6,90	1,00	**6**	2136[4]		
12	*Tprune*	усеч.		30	8,75	0,37	8,20	0,20	13,51	0,68	4,50	1,00	**4**	1695[4]		

лизующим *грамматику графовой генерации* (*graph generation grammar*) по Китано [20], а также двумя методами построения с помощью *аксоновых деревьев* по Нолфи и Канджелози [20].

Рассмотрим по порядку все тестовые задачи из табл. 2. Ввиду большого количества алгоритмов, для эффективного сравнения будем использовать табличную форму представления основных результатов их работы.

Cancer — это набор данных, связанных с диагностикой рака грудной железы. Задача заключается в классификации опухоли (доброкачественной или злокачественной) на основе полученном с помощью микроскопа наборе характеристик клетки. В качестве входных характеристик мы подаем информацию о диаметре группы клеток опухоли, единообразии размера и формы клетки, доле нуклеина, доле хроматина и др. Всего имеется 699 примеров выборки: 9 входов и 2 выхода. Все входы — вещественные значения, 65.5 % всей выборки определяют доброкачественную опухоль. Мы использовали первый вариант разбиения всего набора на выборки.

Оценим эффективность предлагаемого алгоритма ЭНП, полагаясь на сводную таблицу табл. 3 по всем алгоритмам построения нейронных сетей для задачи *Cancer1*. Однако вследствие большого количества работ и разнообразия подачи материала в них сначала необходимо сделать несколько комментариев относительно алгоритмов и организации результатов в таблице табл. 3 и последующих таблицах.

Кратко прокомментируем структуру таблицы табл. 3. Все значения, которые должны были быть приведены авторами алгоритмов, но по каким-то причинам не указанные, обозначены прочерком. Данные, которые имеются в наличии, но которые невозможно поместить в приемлемом виде в табличную часть, прокомментированы ниже по тексту в сносках 1) – 4).

В графе таблицы **построение сети** мы кратко обозначили тип алгоритма: *ручн.* — результат был получен методом эмпирического подбора (взят из PROBEN1 для сравнения); *конст.* — алгоритм относится к конструктивному направлению и реализует построение нейронной сети; *усеч.* — алгоритм выполняет усечение некоторой исходной нейронной сети; в большинстве алгоритмов в качестве исходной сети, как правило, используется сеть из PROBEN1; *эвол.* — реализует эволюционный способ построения нейронной сети; *эв.ус.* — выполняет усечение некоторой исходной нейронной сети с применением эволюционного поиска.

В графе **лучший вариант** мы выделили те алгоритмы, для которых авторы предлагали результаты только для одной, *как правило лучшей*, полученной архитектуры. В этом случае графа **число запусков** обозначает число запусков этой архитектуры по различной первоначальной инициацией весов. Во всех остальных случаях число запусков — это количество построений алгоритмом различных нейронных сетей.

Графа **ошибка классификации** дает нам процент ошибочно определенных значений для трех различных выборок данных. Ошибка на тестовой выборке, не участвующей в процессе построения нейронной

Таблица 5

Результаты по всем алгоритмам построения нейронных сетей для задачи *Diabetes3* из набора PROBEN1. Пояснения 4), 5) и 6) приведены в тексте (см. стр. 103)

	название алгоритма	построение сети	лучший вариант	запусков	ошибка классификации, %						число		структура сети по слоям	CPU - время, чч:мм	число эволюций	размер популяции
					обучение	std. откл	проверка	std. откл	тестовая	std. откл	нейронов	слоев				
	Proben1	ручн.		50	13,34	1,11	17,98	0,62	23,06	1,91	32,00	1,00	**32**	–		
1	*предлагаемый метод ЭННС*	эвол.		50	16,60	2,04	25,87	1,68	23,63	1,54	10,54	1,52	**10**	00:31	8	30
2	*Фридрих*	эвол.	да	5	15,00	–	22,40	–	23,10	–	11,00	5,00	**2-3-2-3-1**	[5]	100	100
3	*АКасПер*	конст.		50	–	–	–	–	23,14	1,26	3,02	3,02	**1-1-1**	–		
4	*КасПер*	конст.		50	–	–	–	–	24,53	1,44	9,78	9,78	**1-1-1-1-1-1-1-1-1**	–		
5	*ANKKA* [6]	эвол.	да	1	–	–	–	–	22,90	–	4,00	1,00	**4**	–	100	50
6	*Cand*	конст.		31	11,80	4,26	17,51	0,62	22,87	1,65	22,55	1,00	**22**	483[4]		
7	*Cascade*	конст.		40	13,94	0,84	17,74	0,59	22,01	1,56	8,55	8,55	**1-1-1-1-1-1-1-1**	845[4]		
8	*Cascor*	конст.		44	13,47	0,92	17,76	0,47	22,17	1,66	8,73	8,73	**1(8)**	434[4]		
9	*Kogi2*	конст.		38	14,36	0,87	17,81	0,72	22,53	2,22	16,45	16,45	**1(16)**	449[4]		
10	*Kogi3*	конст.		29	11,74	2,85	17,00	1,09	23,58	2,04	7,82	1,00	**7**	176[4]		
11	*Kogi9*	конст.		32	12,53	1,81	17,07	1,23	24,30	2,01	13,19	2,00	**7-6**	538[4]		
12	*Lprune*	усеч.		30	13,00	0,57	17,46	0,23	22,10	1,24	13,13	1,00	**13**	1259[4]		
13	*Tprune*	усеч.		30	13,07	0,74	17,82	0,24	22,83	1,01	21,73	1,00	**21**	646[4]		

сети, характеризует обобщающую способность сети.

Число нейронов и слоев также дается в виде, усредненном по количеству запусков. Графа **структура сети по слоям** предназначена для уточнения архитектуры нейронной сети. Запись **4-2-1** обозначает наличие четырех нейронов на первом скрытом слое, двух — на втором и одного — на третьем. В графе **CPU-время** указано среднее время работы алгоритма. Графы **число эволюций** и **размер популяции** относятся только к эволюционным алгоритмам, **число эволюций** показывает количество произведенных эволюций до достижения результата.

Для отдельных алгоритмов в сводных таблицах приведены следующие комментарии.

1) Точного значения по времени для каждого из набора алгоритмов *Миллер, Китано, Нолфи* и *Канджелози* в [36] не приведено. Известно только, что длительность построения нейронных сетей лежит в интервале от 14 минут до 70 часов на рабочей станции Ultra Sparc 167-296MHz и алгоритмы *Нолфи* и *Канджелози* самые медленные.

2) Задача выполнялась на рабочей станции SparcStation-10.

3) Данные тестовой задачи были взяты из PROBEN1, но были подвергнуты дополнительной предобработке таким образом, чтобы иметь нулевое среднее и стандартное отклонение, равное единице.

4) В [30] время работы указано без единиц измерения; скорее всего, это секунды. Кроме того, для тестирования использовалось несколько компьютеров: однопроцессорный MasPar MP-1 16384, многопроцессорный KSR-1 32 и шесть рабочих станций Sun.

5) Общее время выполнения — 20 дней на рабочей станции SparcStation-10; было просмотрено около 120000 архитектур.

6) Тестовые наборы были подготовлены авторами самостоятельно по данным первоисточников, а не из PROBEN1. Проверочные задачи взяты: Card, Diabetes, Heart из Stat Log Datasets (ftp://ftp.ics.uci.edu-/pub/machine-learning-databases/), Glass из University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Выборки не отличаются по общему числу примеров, но число атрибутов несколько отлично, поскольку в PROBEN1 Пречелт добавлял атрибуты, обозначающие пропущенные значения. Кроме того, разбиение на тестовую выборку в [39] было отведено только 10 % от набора, а вместо валидационной выборки была реализована кросс-валидация.

7) Тестовый набор был взят из Stat Log Datasets и по всем параметрам практически совпадает с PROBEN1. В процессе обучения использовалось только 713 специально отобранных примеров, но значения ошибки классификации считалось на наборе из 3 428 примеров. Задача выполнялась на шести Sun4 и четырех NeXT рабочих станциях.

Основываясь на результатах, полученных алгоритмом ЭНП, и на сводных данных из табл. 3, можно

Таблица 6

Результаты по всем алгоритмам построения нейронных сетей для задачи *Glass1* из набора PROBEN1. Пояснения 1), 4) и 6) приведены в тексте (см. стр. 103)

	название алгоритма	построение сети	лучший вариант	запусков	ошибка классификации, %						число		структура сети по слоям	CPU - время, чч:мм	число эволюций	размер популяции
					обучение	std. откл	проверка	std. откл	тестовая	std. откл	нейронов	слоев				
	Proben1	ручн.		50	7,68	0,79	9,48	0,24	39,03	8,14	32,00	2,00	**16-8**	–		
1	*предлагаемый метод ЭННС*	эвол.		50	19,14	4,13	38,66	3,14	35,73	2,88	9,70	1,56	**9**	00:31	4	30
2	*Миллер*	эвол.		30	33,33	8,54	38,21	6,47	38,18	9,13	12,00	9,00	**2-1-1-2-2-1(3)**	[1]	100	50
3	*Китано*	эвол.		30	28,13	5,87	34,81	2,72	32,08	5,25	23,00	1,00	**1(23)**	[1]	100	50
4	*Нолфи*	эвол.		30	37,85	14,6	43,40	12,2	49,69	9,37	30,00	6,00	**9-2-2-4-9-4**	[1]	100	50
5	*Канджелози*	эвол.		30	62,68	14,0	59,81	15,9	63,46	14,8	13,00	5,00	**2-1-2-1-7**	[1]	100	50
6	*АКасПер*	конст.		50	–	–	–	–	30,68	2,61	4,18	4,18	**1-1-1-1**	–		
7	*КасПер*	конст.		50	–	–	–	–	34,76	5,88	8,07	8,07	**1-1-1-1-1-1-1-1**	–		
8	*ANKKA* [6]	эвол.	да	1	–	–	–	–	35,00	–	12,00	2,00	**8-4**	–	100	40
9	*Cand*	конст.		42	6,17	0,44	9,18	0,25	36,39	3,24	8,60	1,00	**8**	160[4]		
10	*Cascade*	конст.		42	6,49	0,87	9,25	0,36	36,97	3,91	12,36	12,36	**1(12)**	287[4]		
11	*Cascor*	конст.		45	5,79	1,05	8,99	0,54	34,76	5,88	8,07	8,07	**1-1-1-1-1-1-1-1**	352[4]		
12	*Kogi2*	конст.		42	6,62	0,76	9,25	0,31	36,84	3,15	11,88	11,88	**1(11)**	289[4]		
13	*Kogi3*	конст.		42	5,34	0,55	8,45	0,28	34,23	2,55	10,38	1,00	**10**	195[4]		
14	*Kogi9*	конст.		40	5,45	0,70	8,29	0,38	34,20	4,04	9,63	2,00	**6-3**	283[4]		
15	*Lprune*	усеч.		30	6,43	0,71	8,99	0,32	34,40	7,92	3,03	2,00	**2-1**	258[4]		
16	*Tprune*	усеч.		30	7,74	0,51	9,34	0,20	37,67	7,79	3,03	2,00	**2-1**	166[4]		

сказать, что алгоритм ЭНП ведет себя для задачи диагностики рака грудной железы ожидаемым образом. Объединение положительных сторон конструктивного и эволюционного подходов действительно позволило создавать эффективные нейронные сети очень компактной архитектуры. В целом, по 50 запускам ЭНП строил сети со средним значением ошибки ниже среднего по всем методам на 1.87 % (если отбросить явно неудачные результаты для алгоритма *SCGA*). В этом отношении алгоритм ЭНП уступает только методам, реализующим усечение. Реализация в ЭНП эффективного эволюционного подхода для данной задачи позволила получать самые компактные нейронные сети. Время получения результатов было также приемлемым. По количеству затраченных эволюций, а также по времени выполнения, ЭНП был в несколько раз быстрее эволюционных алгоритмов и, если придерживаться соображений, изложенных в комментарии 4) на стр. 103, не слишком отстает от алгоритмов конструктивного направления.

Набор данных **Card** используется для предсказания подтверждения или не подтверждения платежеспособности кредитной карты. Каждый пример выборки содержит некоторую информацию о кредитной истории карты, а выходное значение показывает, доверяет или нет банк данной кредитной карте. Значения атрибутов приводится без описания по конфиденциальным соображениям. Всего имеется 690 примеров выборки: 51 вход и 2 выхода. Вся выборка содержит около 5 % пропущенных атрибутов. Этот набор имеет хорошее сочетание атрибутов: вещественные, целые перечисляемые с малым и с большим числом вариантов, 44 % выборки подтверждают платежеспособность карты. Мы использовали вариант разбиения набора Card1.

Помимо ЭНП для решения данной проблемы применялись, в основном, конструктивные алгоритмы и только один эволюционный. Все результаты приведены в табл. 4.

Как видно из табл. 4, данная задача оказалась довольно простой с практически линейными зависимостями. Для решения задачи многим алгоритмам потребовалось всего один–два нейрона на одном скрытом слое, а некоторые (*АКасПер*) сформировали нейронные сети совсем без скрытых слоев. Алгоритм ЭНП сумел построить для данной задачи не самые лучшие нейронные сети и обошел только другой эволюционный алгоритм. Скорее всего, данная задача оказалась слишком простой для решения, и ее можно эффективно решать без использования сложного эволюционного подхода.

Diabetes — это набор данных, который содержит информацию о диагностике диабета для индейцев Пима (Pima). Опираясь на персональные данные (возраст, число беременностей) и результаты медицинского обследования (кровяное давление, вес, содержание глюкозы и др.), принимается решение, болен индеец диабетом или нет. В выборке содержится 768 примеров, 8 входов и 2 выхода. Все входы — ве-

Таблица 7

Результаты по всем алгоритмам построения нейронных сетей для задачи *Heart1* из набора PROBEN1. Пояснения 1), 4) и 6) приведены в тексте (см. стр. 103)

	название алгоритма	построение сети	лучший вариант	запусков	ошибка классификации, %						число		структура сети по слоям	CPU - время, чч:мм	число эволюций	размер популяции
					обучение	стд. откл	проверка	стд. откл	тестовая	стд. откл	нейронов	слоев				
	Proben1	ручн.		50	9,25	1,07	13,22	1,32	19,89	2,27	32,00	1,00	**32**	–		
1	*предлагаемый метод ЭННС*	эвол.		50	11,60	0,91	17,80	1,03	18,86	0,91	4,24	1,16	**4**	00:11	3	30
2	*Миллер*	эвол.		30	15,42	1,45	17,14	1,27	20,32	1,25	11,00	9,00	**1(3)-3-1(5)**	[1]	100	50
3	*Китано*	эвол.		30	14,86	0,31	17,49	0,49	19,97	0,43	4,00	3,00	**1-2-1**	[1]	100	50
4	*Нолфи*	эвол.		30	22,70	9,6	24,52	9,4	26,84	7,10	32,00	6,00	**15-1-3-3-1-9**	[1]	100	50
5	*Канджелози*	эвол.		30	32,41	12,2	33,80	12,6	34,16	10,0	20,00	5,00	**5-2-2-2-9**	[1]	100	50
6	*АКасПер*	конст.		50	–	–	–	–	19,21	0,44	0,10	0,10	**0**	–		
7	*КасПер*	конст.		50	–	–	–	–	19,89	1,58	2,64	2,64	**1-1-1**	–		
8	*ANKKA* [6]	эвол.	да	1	–	–	–	–	14,80	–	5,00	1,00	**5**	–	100	70
9	*Cand*	конст.		42	8,24	1,11	13,00	0,23	20,08	1,35	5,07	1,00	**5**	574[4]		
10	*Cascade*	конст.		42	9,22	0,84	13,02	0,22	20,09	1,17	4,86	4,86	**1-1-1-1-1**	580[4]		
11	*Cascor*	конст.		45	9,03	0,75	13,20	0,16	19,99	1,58	2,64	2,64	**1-1-1**	1050[4]		
12	*Kogi2*	конст.		42	8,76	1,07	13,01	0,21	20,27	1,31	5,98	5,98	**1-1-1-1-1-1**	706[4]		
13	*Kogi3*	конст.		42	7,83	1,29	12,84	0,30	19,73	1,50	7,24	1,00	**7**	634[4]		
14	*Kogi9*	конст.		40	7,85	1,20	12,82	0,31	20,10	1,45	7,00	2,00	**5-2**	1069[4]		
15	*Lprune*	усеч.		30	8,44	0,55	12,78	0,27	19,51	1,15	13,90	1,00	**1(13)**	2117[4]		
16	*Tprune*	усеч.		30	9,09	0,46	12,72	0,24	20,14	0,94	3,43	1,00	**3**	2297[4]		

щественные значения, для 65.1 % всей выборки зафиксировано отсутствие диабета. Мы использовали вариант разбиения набора Diabetes3.

Для данной задачи результаты по алгоритму ЭНП оказались лучше, чем для предыдущей. По значению тестовой ошибки были показаны средние результаты, зато построенные нейронные сети имели более компактный вид, чем у остальных подходов. Единственный метод, который оказался эффективнее, — это эволюционный алгоритм *ANKKA*. Заметим, что для *ANKKA* у нас имеется информация только о его наилучших результатах, полученных на немного отличающихся данных (см. стр. 103). Со временем выполнения алгоритма ЭНП ситуация остается неизменной: продолжительнее большинства конструктивных алгоритмов, но в несколько раз быстрее эволюционных.

Набор данных **Glass** используется для определения качества стекла по его химическому составу. Результат химического анализа осколка стекла (процентное содержание следующих элементов: RI, Na, Mg, Al, Si, K, Ca, Ba и Fe) плюс преломляющие свойства используются для отнесения стекла к одному из 6 классов: оконное стекло с пузырьками, оконное стекло без пузырьков, транспортное стекло, стекло для тары, стекло для посуды, ламповое стекло. Подобная задача возникает в случае экспертизы в судебном расследовании. В выборке содержится 214 примеров, 9 входов и 6 выходов. Все входные значения вещественные, два из них, по-видимому, мало влияют на результат. Размеры 6 классов соответственно 70, 76, 17, 13, 9 и 29 примеров. Мы использовали вариант разбиения набора Glass1.

Задача определения типа стекла оказалась достаточно сложной, если судить по большому проценту ошибок и необходимому размеру нейронных сетей. В этом случае ЭНП дает ошибку, стабильно меньшую среднего значения по всем алгоритмам, и одну из самых малых архитектур для нейронных сетей. Ближайший конкурент по архитектуре — *Cand* — имеет на один процент большее значение ошибки. Примечательно то, что практически все алгоритмы автоматического построения нейронных сетей получили лучший результат, чем вручную подобранный в PROBEN1.

Heart — это набор данных, который применяется для предсказания сердечных заболеваний. Делается вывод о том, что по крайней мере одна из четырех основных сердечных камер (правое и левое предсердия, правый и левый желудочки) уменьшилась в диаметре больше чем на 50 %. Прогнозирование вида "да/нет" основано на различной персональной информации о больном (возраст, пол, курение, субъективное описание самочувствия и др.) и на результатах медицинской диагностики (кровяное давление, электрокардиограмма и др.). В выборке содержится 920 примеров, 35 входов и 2 выхода. Многие атрибуты имеют пропущенные значения. Для атрибутов 10, 12 и 11 мы имеем соответственно 309, 611 и 486

Таблица 8

Результаты по всем алгоритмам построения нейронных сетей для задачи *Thyroid2* из набора PROBEN1. Пояснения 3), 4) и 7) приведены в тексте (см. стр. 103)

	название алгоритма	построение сети	лучший вариант	запусков	ошибка классификации, %						число		структура сети по слоям	CPU - время, чч:мм	число эволюций	размер популяции
					обучение	std. откл	проверка	std. откл	тестовая	std. откл	нейронов	слоев				
	Proben1	ручн.		50	0.59	0.24	0.88	0.19	1.86	0.41	12.00	2.00	**8-4**	–		
1	*предлагаемый метод ЭННС*	эвол.		50	1.11	0.24	1.48	0.29	1.89	0.20	4.08	1.36	**4**	02:52	2	30
2	*АКасПер*	конст.		50	–	–	–	–	1.67	0.26	4.64	4.64	**1-1-1-1**	–		
3	*КасПер*	конст.		50	–	–	–	–	3.03	1.15	25.04	25.04	**1(25)**	–		
4	*BP-Generator* [7]	эвол.	да	1	0.50	–	–	–	1.80	–	48.00	–	**–**	72:00	3000	–
5	*MBP* [3]	усеч.		11	–	–	–	–	1.10	–	10.00	2.00	**7-3**	–		
6	*OBS* [3]	усеч.		11	–	–	–	–	1.20	–	10.00	2.00	**6-4**	–		
7	*unit-OBS* [3]	усеч.		1	–	–	–	–	1.10	–	6.00	2.00	**4-2**	–		
8	*ENZO* [3]	эв.ус.		1	–	–	–	–	1.20	–	8.00	2.00	**5-3**	–	–	30
9	*ENZO-MI* [3]	эв.ус.		1	–	–	–	–	1.20	–	7.00	2.00	**5-2**	–	--	30
10	*Cand*	конст.		30	1.61	2.72	2.04	3.18	3.51	4.57	36.50	1.00	**36**	15629 [4]		
11	*Cascade*	конст.		43	2.13	1.23	2.16	1.09	4.43	2.07	14.88	14.88	**1(14)**	21326 [4]		
12	*Cascor*	конст.		40	0.61	0.99	1.41	0.55	2.53	0.87	22.00	22.00	**1(22)**	87454 [4]		
13	*Kogi2*	конст.		36	2.33	1.17	2.22	1.09	4.83	2.37	12.63	12.63	**1(12)**	10792 [4]		
14	*Kogi3*	конст.		24	2.34	4.12	2.92	5.02	5.17	8.07	32.00	1.00	**32**	16225 [4]		
15	*Kogi9*	конст.		30	2.04	2.21	2.23	2.96	4.03	5.10	16.07	2.00	**10-6**	9385 [4]		
16	*Lprune*	усеч.		30	0.50	0.09	0.81	0.10	1.65	0.20	18.33	2.00	**13-5**	33132 [4]		
17	*Tprune*	усеч.		30	0.59	0.15	0.86	0.09	1.80	0.23	17.73	2.00	**12-5**	39835 [4]		

пропущенных значений; для остальных полей их около 60. Дополнительное бинарное значение используется для обозначения пропущенного значения. По всей выборке 45 % пациентов не имеют уменьшенных сердечных камер. Мы использовали вариант разбиения набора Heart1.

Для данной задачи ЭНП показал отличные результаты. Значение ошибки было самым малым, если учитывать, что алгоритм *ANKKA*, для которого приводится только лучший результат, а не средний по нескольким запускам. Кроме того, для *ANKKA* использовалась несколько отличающаяся выборка данных (см. стр. 103). Очень компактно формируются и нейронные сети, результаты лежат близко к лучшим (конструктивные алгоритмы) и превышают все, полученные эволюционным путем по степени компактности.

По времени выполнения ЭНП показал интересные результаты: если придерживаться соображений, изложенных в комментарии 4) на стр. 103, то для большинства конструктивных методов время выполнения будет существенно больше. Конечно, мы не можем учесть разницу в производительности машин, но мы можем сравнивать отношения времени выполнения для разных проблем из PROBEN1. В случае ЭНП соотношение трех эволюций к 100 для эволюционных алгоритмов говорит само за себя.

Набор данных **Thyroid** используется для диагностики состояния щитовидной железы. Основываясь на персональных данных пациента и результатах медицинского обследования, делается суждение о повышении функции (гипертиреоз), нормальном состоянии или недостаточности функции (гипотиреоз) щитовидной железы. В выборке содержится 7 200 примеров, 21 вход и 3 выхода. Для различных атрибутов имеются пропущенные значения. Размеры каждого класса соответственно 5.1 %, 92.6 % и 2.3 %. Мы использовали вариант разбиения набора Thyroid2.

На наш взгляд, наилучших результатов алгоритм ЭНП достигает в задаче диагностики состояния щитовидной железы. По обобщающей способности сети (значению проверочной ошибки) он оказался эффективнее практически всех конструктивных алгоритмов. Лучшие результаты по значению ошибки получены только для методов усечения. По компактности архитектуры построенной сети в случае ЭНП получен удивительно большой отрыв от всех прочих алгоритмов: всего четыре нейрона! Время выполнения было сравнимо со временем, затраченным конструктивными подходами, в то время как для остальных тестовых проблем ЭНП по этому параметру существенно проигрывал. Видимо, это как-то связано с очень большим объемом данных в выборках.

Достаточно подробно был исследован алгоритм эволюционного наращивания на сложном синтетиче-

ском тесте и семи практических задачах из области медицины, финансов и судебной экспертизы. Проведен сравнительный анализ полученных на этих задачах результатов с имеющимися данными по 8 конструктивным, 7 методам усечения и 8 эволюционным алгоритмам построения нейронных сетей, которые охватывают все основные подходы к их автоматическому созданию.

Полученные результаты позволяют говорить, что ЭНП успешно справился с решением всех проверочных задач. Предложенный метод сопоставим на данном наборе практических задач со всеми алгоритмами семейства каскадной корреляции по размерам построенной сети, уступает им только по времени построения и воспроизводит все результаты лучше всех прочих алгоритмов, основанных на эволюционном подходе. Как и следовало ожидать, ЭНП уступает по степени компактности полученной нейронной сети некоторым алгоритмам, которые реализуют усечение уже существующей сети.

Подводя итоги, можно сказать, что предложенный нами метод эволюционного наращивания достаточно эффективен и может применяться для решения практических задач.

СПИСОК ЛИТЕРАТУРЫ

1. *Горбань А.Н., Дунин-Барковский В.Л., Кирдин А.Н.* Нейроинформатика. Новосибирск: Наука, 1998 (http://www.neuropower.de/rus/).
2. *Bishop C.M.* Neural network for pattern recognition. Oxford: Oxford University Press, 1997.
3. *Moody J.E., Utans J.* Architecture selection strategies for neural networks: application to corporate bond rating prediction // Neural Networks in the Capital Markets. 277–300. Chichester: Wiley, 1995 (ftp://cse.ogi.edu/pub/tech-reports/1994/94-036.ps.gz).
4. *Ripley B.D.* Pattern recognition and neural networks. Cambridge: Cambridge University Press, 1997.
5. *Hassibi B., Stork D.G.* Second order derivatives for network pruning: optimal brain Surgeon // Neural Information Processing Systems. 1992 (ftp://archive.cis.ohio-state.edu/pub/neuroprose/stork.obs.ps.gz).
6. *Torres-Moreno J.M.* Apprentissage et généralisation par des réseaux de neurones : étude de nouveaux algorithmes constructifs: PhD Thesis. De l'Institut National Polytechnique de Grenoblre. Grenoblre, 1997 (ftp://archive.cis.ohio-state.edu/pub/neuroprose/torres.thesis.ps.Z).
7. *Fahlman S.E., Lebiere C.* The cascade-correlation learning architecture // Advances in Neural Information Processing II. 1990. 524–532 (ftp://archive.cis.ohio-state.edu/pub/neuroprose/fahlman.cascor-tr.ps.gz).
8. *Горбань А.Н.* Обучение нейронных сетей. М.: СП "ParaGraph", 1990.
9. *Murata N., Yoshizawa S., Amari S.* Network information criterion — determining the number of hidden units for an artificial neural network model // IEEE Transactions on Neural Networks. 1994. **5**, N 6. 865–872 (http://www.islab.brain.riken.go.jp/ mura/paper/mura94_nic.ps.gz).
10. *Spears W.M., de Jong A., Back T., Fogel D.B., de Garis H.* An overview of evolutionary computation // Proceedings of the 1993 European Conference on Machine Learning. 1993 (http://www.aic.nrl.navy.mil/ spears/papers/ecml93.ps).
11. *Holland J.H.* Adaptation in natural and artificial systems. Univ. of Michigan Press, 1992.
12. *Дмитрович А.И.* Интеллектуальные информационные системы. Минск: Тетрасистемс, 1997.
13. *Branke J.* Evolutionary algorithms for neural network design and training // 1st Nordic Workshop on Genetic Algorithms and its Applications. 1995 (ftp://ftp.aifb.uni-karlsruhe.de/pub/jbr/Vaasa.ps.gz).
14. *Koehn Ph.* Combining genetic algorithms and neural networks: the encoding problem: PhD Thesis. The University of Tennessee. Knoxville, 1994 (ftp://archive.cis.ohio-state.edu/pub/neuroprose/koehn.encoding.ps.gz).
15. *Тютерев В.В., Шевелев О.Г.* Использование L-грамматик для определения нейронной сети оптимальной топологии методом генетических алгоритмов // IV межвузовская конференция студентов, аспирантов и молодых ученых "Наука и образование". Томск: ТГПУ, 2000.
16. *Siddiqi A.A., Lucas S.M.* A comparison of matrix rewriting versus direct encoding for evolving neural networks // Proceedings of Intern. Joint Conf. on Neural Networks'98. Anchorage, 1998.
17. *Gruau F., Whitley D., Pyeatt L.* A comparison between cellular encoding and direct encoding for genetic neural networks // Proceedings of the First Genetic Programming Conference. 1996. 81–89.
18. *Jacob W., Rehder M.* Evolution of neural net architectures by a hierarchical grammar-based genetic system // Proceedings of the International Joint Conference on Neural Networks and Genetic Algorithms. Innsbruck, 1993.
19. *Gruau F.* Cellular encoding of genetic neural networks. Tech. Rep. 92-21, Laboratoire de l'Informatique du Parallelisme, Ecole Normale Superieure de Lyon. Lyon, 1992 (http://www.cwi.nl/ gruau/gruau/RR92-21.ps.Z).
20. *Grönroos M.* Evolutionary design of neural networks: PhD Thesis. Department of Mathematical Sciences, University of Turku. Turku, 1998 (http://magi.yok.utu.fi/-Emagi/opinnot/gradu/mscthesis.ps.gz).
21. *Hussain T.S.* Modularity within neural networks: PhD Thesis. Department of Computing and Information Sciences, 1995.
22. *Gruau F.* Automatic definition of sub-neural networks. Tech. Rep. RR94-28. Laboratoire de l'Informatique du Parallelisme, Ecole Normale Superieure de Lyon. Lyon, 1994.
23. *Boers E., Kuiper H., Happel B., Sprinkhuizen-Kuyper I.* Designing modular artificial neural networks. Tech. Rep. 93-24. Department of Computer Science, Leiden University. Leiden, 1993.

24. *Hinton E., van Camp D.* Keeping neural networks simple by minimizing the description length of the weights // Proceedings of the Workshop on Computational Learning Theory. Toronto: Morgan Kaufmann Publishers, 1993 (http://www.cs.utoronto.ca/ drew/colt93.ps).
25. *Тютерев В.В., Новосельцев В.Б.* Метод динамического наращивания узлов как способ построения нейронных сетей эффективного размера // Нейрокомпьютеры: разработка, применение. М.: Радиотехника, 2001. № 2. 3–8.
26. *Тютерев В.В., Новосельцев В.Б.* Исследования алгоритма автоматического построения нейронной сети // Исследования по анализу и алгебре. Томск: Томский гос. ун-т. 2001. № 3. 269–281.
27. *Phatak D.S., Koren I.* Connectivity and performance tradeoffs in the cascade correlation learning architecture. Tech. Rep. TR-92-CSE-27. ECE Dept., UMASS. Amherst, 1994 (ftp://archive.cis.ohio-state.edu/pub/neuroprose/phatak.layered-cascor.ps.gz).
28. *Treadgold N.K., Gedeon T.D.* Exploring constructive cascade networks // IEEE Transactions on Neural Networks. 1999 (http://www.cse.unsw.edu.au/ nickt/doc/acasper.ps).
29. *Prechelt L.* Proben1 — a set of neural network benchmark problems and benchmarking rules. Tech. Rep. 21/94. Fakultät für Informatik, Universität Karlsruhe. Karlsruhe, 1994 (ftp://ftp.ira.uka.uka.de/pub/neuron/proben1.tar.gz).
30. *Prechelt L.* Investigation of the CasCor family of learning algorithms // Neural Networks. 1997. **10**, N 5. 885–896 (ftp://ftp.ira.uka.de/pub/neuron/neurnetw97.ps.gz).
31. *Treadgold N.K., Gedeon T.D.* Exploring architecture variations in constructive cascade networks // Proceedings Int. Joint Conf. on Neural Networks. Anchorage, 1998. 343–348 (http://www.cse.unsw.edu.au/ nickt/doc/tower.ps).
32. *Treadgold N.K., Gedeon T.D.* A Cascade network algorithm employing progressive RPROP // Int. Conf. on Artificial and Natural Neural Networks. Lanzarote, 1997. 733–742 (http://www.cse.unsw.edu.au/ nickt/doc/casper.ps).
33. *Treadgold N.K., Gedeon T.D.* Extending and benchmarking the CasPer algorithm // Australian Conference on Artificial Intelligence. Perth, 1997. 398–406 (http://www.cse.unsw.edu.au/ nickt/doc/casperp_class.ps).
34. *Treadgold N.K., Gedeon T.D.* Extending CasPer: a regression survey // Int. Conf. on Neural Information Processing. Dunedin, 1997. 310–313 (http://www.cse.unsw.edu.au/ nickt/doc/casperp_reg.ps).
35. *Schiffmann W., Joost M., Werner R.* Application of genetic algorithms to the construction of topologies for multilayer perceptrons // Proceedings of Artificial Neural Networks and Genetic Algorithms. Innsbruck, 1993. 675–682.
36. *Salustowicz R.* A genetic algorithm for the topological optimization of neural networks: PhD Thesis. Technische Universitat Berlin. Berlin, 1995 (ftp://archive.cis.ohio-state.edu/pub/neuroprose/salustowicz.evnn.ps.gz).
37. *Friedrich C.M., Moraga C.* An evolutionary method to find good building-blocks for architectures of artificial neural networks // Proceedings of the Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU96). Granada, 1996. 951–956.
38. *Friedrich C.M., Moraga C.* Using genetic engineering to find modular structures and activation functions for architectures of artificial neural networks // Computational Intelligence, Theory and Applications. 1997. 150–161.
39. *Fredriksson K.* Genetic algorithms and generative encoding of neural networks for some benchmark classification problems // Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications. Turku, 1997. 123– 34.
40. *Ragg T., Gutjahr S., Hai Ming Sa.* Automatic determination of optimal network topologies based on information theory and evolution // Euromicro '97, Track on Computational Intelligence. Springbank, 1997 (ftp://springbank.ira.uka.de/pub/neuro/ragg/euromicro97.ps.gz).
41. *Gruau F.* Automatic definition of modular neural networks // Adaptive Behavior. 1995. N 3. 151–183.
42. *Finnoff W., Hergert F., Zimmermann H.G.* Improving model selection by nonconvergent methods // Neural Networks. 1993. N 6. 771–783.

Поступила в редакцию
15.09.2001