

УДК 004.032.26, 519.766.23

К.А. Кузнецов, А.В. Додатко

## МЕТОДЫ АДАПТИВНОГО ПРОЕКТИРОВАНИЯ НЕЙРОННЫХ СЕТЕЙ ПРЯМОГО РАСПРОСТРАНЕНИЯ С КАСКАДНОЙ АРХИТЕКТУРОЙ

*Аннотация.* Данная работа посвящена разработке инструмента для автоматического подбора архитектуры нейронных сетей с помощью эволюционных алгоритмов. Рассмотрены альтернативные способы представления архитектуры и алгоритмы обучения нейронных сетей каскадного типа.

*Ключевые слова:* Нейронные сети, L-системы, эволюционные алгоритмы, автоматический подбор архитектуры, задача о двух спиралях.

### Постановка задачи

Нейронные сети с неполиномиальными функциями активации являются широко используемым инструментом при решении задач аппроксимации функций. В работе [1] их универсальность была доказана для ограниченных и непрерывных функций, а впоследствии эти результаты были обобщены [2] и для произвольных функций. Традиционно для решения задач аппроксимации используют сети прямого распространения (Feedforward Neural Network, FNN).

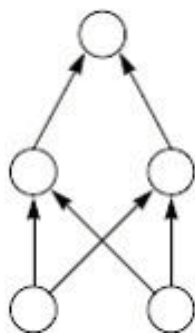


Рис. 1 – Классическая архитектура для задачи XOR

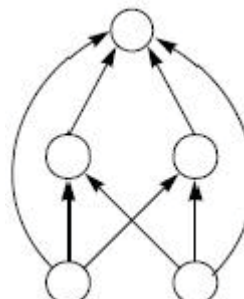


Рис. 2 – Архитектура для задачи XOR с каскадными элементами

В [3] показано, что для обучения обычной FNN сети (рис. 1) на задаче XOR требуется в 55 раз больше эпох, чем для сети с дополнительными прямыми связями между входными и выходным нейронами (рис. 2). Таким образом, наличие прямых связей между

несмежными слоями сети может существенно повысить скорость и качество её обучения. Существует гипотеза, что такие связи устраняют плато в поверхности ошибки.

Следовательно, очень важно найти наилучшую архитектуру для решения каждой конкретной задачи. Для небольших задач или задач с известными решениями архитектуру сети можно разработать и вручную. Однако, когда речь идет о сложных задачах, не имеющих аналитического решения, найти оптимальную топологию сети вручную становится практически невозможно.

Задача нахождения оптимальной архитектуры сети состоит из

- нахождения оптимальной топологии сети;
- подбора активационных функций нейронов в рамках данной топологии.

В данной работе мы ограничились рассмотрением только первой задачи.

### **Анализ существующих публикаций**

Традиционно для автоматического подбора топологии сети используются конструктивные и деструктивные алгоритмы. Одним из таких методов является метод каскадной корреляции [4]. Сеть, обучаемая по этому методу, имеет ряд преимуществ по сравнению с FNN:

- отсутствие обратного распространения повышает скорость и качество обучения;
- сеть сохраняет усвоенную ранее информацию при повторном обучении, тем самым обеспечивая большую устойчивость;
- сеть автоматически определяет собственные размеры при обучении, тем самым предельно упрощая процесс проектирования.

Однако метод каскадной корреляции имеет и определенные недостатки. Топология сети имеет фрактальную природу, а решаемая задача определяет только её размер. Таким образом, топология сети, построенная по методу каскадной корреляции, почти всегда неоптимальна.

Использования как конструктивных, так и деструктивных методов обучения не достаточно для решения сложных проблем. В дальнейших исследованиях [5] были попытки применить для автоматического подбора архитектуры эволюционные, итерационные и др. алгоритмы.

### Цели статьи

Целью данного исследования является разработка инструмента для автоматического подбора оптимальной топологии нейронной сети. В данной статье описана используемая нами схема кодирования, операторы мутации, метод обучения сети и сравнение с аналогичными подходами.

### Основная часть

При автоматическом проектировании топологий сетей общей рекомендацией является уменьшение поискового пространства с помощью как можно более компактных схем кодирования топологий. С этой точки зрения, представление графа сети в виде матрицы смежности не является приемлемым. В [3] был предложен альтернативный способ представления топологии сети, основанный на L-системах. Он позволяет получить топологию сети, описывая правила построения графа в виде формальной грамматики. Очевидно, что это наиболее компактный способ описания графов фрактальной природы.

В грамматике, предложенной [3], алфавит определяется следующим образом.

1. Символы латинского алфавита используются для обозначения вершин. Они не обязаны быть уникальными – наличие идентификатора просто свидетельствует о наличии вершины.

2. Целые числа используются для кодирования дуг ориентированного графа. Началом дуги является вершина, которая кодируется ближайшей буквой слева от обозначения этой дуги в выводе грамматики. Концом дуги является вершина, отстоящая в выводе грамматики от её начала на соответствующее число вершин.

3. Квадратные скобки [ ]. Все, что находится между ними, считается изолированным подграфом, эквивалентным единственной вершине.

Назовем **входными слотами** подграфа вершины, имеющие входящие дуги из вершин, не принадлежащих этому подграфу. Соответственно, **выходными слотами** назовем вершины, в которых начинаются дуги, исходящие из этого подграфа.

Рассмотрим граф, закодированный следующим правилом вывода: *A12[C1D]01-1B*. При использовании описанной выше грамматики

возникает неопределенность в выборе входных и выходных слотов подграфа  $[CID]$ . Ниже приведены наиболее распространенные варианты разрешения этой неоднозначности :

1. все вершины подграфа считаются входными слотами;
2. все вершины подграфа считаются выходными слотами;
3. все вершины подграфа, не имеющие входящих дуг из вершин данного подграфа, считаются входными слотами;
4. все вершины подграфа, не имеющие исходящих дуг в вершины данного подграфа, считаются выходными слотами;

Также эта грамматика имеет ограничение в один символ на длину идентификатора вершины.

Для разрешения этих противоречий и ограничений нами были введены новые символы :

1. Подчеркивание «\_». Идентификатор вершины начинается с этого символа, за которым следует произвольное количество букв латинского алфавита.

2. Двоеточие «:» отделяет обозначения типа слота от идентификатора его вершины.

3. Символы «i», «o», следующие за разделителем «:», используются для кодирования входных и выходных слотов вершины соответственно.

В модифицированной грамматике, цепочка вывода будет иметь вид  $\_A:i+1+2[_C:io+1\_D:o]+0+1-1\_B:o$ . В ней отсутствуют противоречия и она однозначно описывает следующий граф (рис. 3).

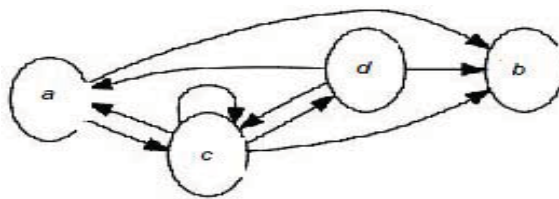


Рис. 3 – Граф, соответствующий цепочке вывода

Сеть с каскадной архитектурой [4] может быть описана такой L-системой:

$$G = \langle N, T, P, S \rangle$$

$$T = \{c, d, b, \_ , :, i, o, [, ]\}$$

$$N = \{INPUTS, LAYERS, OUTPUTS, A\}$$

$$S = \{ \_ INPUTS: i + 1 + 2 \_ LAYERS + 1 \_ OUTPUTS: o \}$$

$$P = \{ \_ INPUTS \rightarrow [ \_ c: io\_d: io ]$$

$$\_ LAYERS \rightarrow [ [ \_ A: io ]: io + 1 \_ b: io ]$$

$\_A: io \rightarrow \_b: io + 1 \ [\_A: io]: io\}$

После трехкратного применения правил к аксиоме получаем цепочку вывода, описывающую граф на рис. 4:

$[\_c: io \ \_d: io]: i \ 1 \ 2 \ [[\_b: io + 1 \ \_A: io]: io]: io + 1 \ \_b: io] + 1 \ \_OUTPUTS: o$

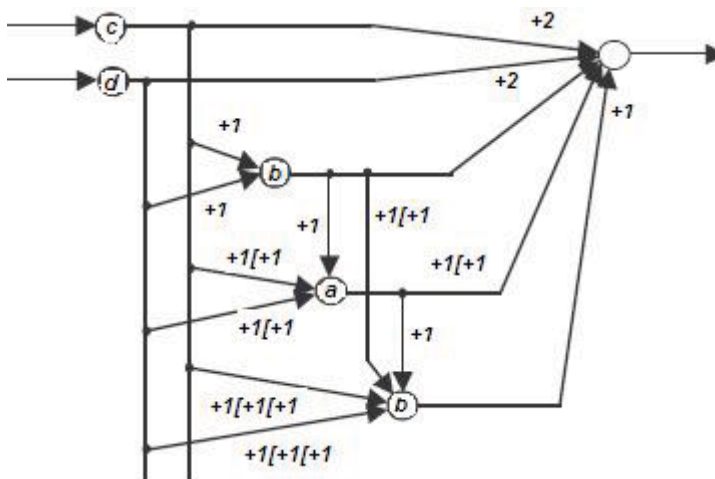


Рис. 4 – Каскадная сеть

Вслед за [3], мы будем использовать эволюционные алгоритмы для автоматического подбора топологии нейронных сетей. В приведенной ниже таблице представлены эквивалентные значения терминов, которые используются для описания подбора архитектуры сети и генетических алгоритмов.

Таблица 1

Соответствие между генетическими алгоритмами и L-системами

Генетические алгоритмы	Подбор архитектуры сети
Хромосома	L-система
Ген	Правило L-системы, Аксиома L-системы
Приспособленность (fitness)	Ошибка на выходе сети (MSE)
Скращивание (crossover)	Не используется
Мутация (mutation)	Замена вершины на слой нейронов, бинарное дерево, цепочку или каскад

Для использования данной схемы кодирования топологии в генетическом алгоритме мы предлагаем ввести следующие операторы мутации:

- 1. Замена вершины на слой нейронов.** В L-систему вводится правило вида:

$$Padd = (\_b \rightarrow [\_x1: io \ \_x2: io \ \_x3: io])$$

Количество нейронов, на которые будет заменена вершина, выбирается случайно. С целью сокращения размерности пространства поиска, мы ограничили максимальное количество вершин в правой части этого правила пятью.



Рис. 5 – до применения правила      Рис. 6 – после применения правила

**2. Замена вершины на каскад.** В L-систему вводится правило вида:

$$\text{Padd} = (\_b \rightarrow [[_x: io]: i + 1 \_y: o] \\ \_x: io \rightarrow \_y: io + 1 [\_x: io]: io)$$

**3. Замена вершины на изолированную цепочку**

В L-систему вводится правило вида:

$$\text{Padd} = \_b \rightarrow [\_b1:i+1\_b2:o]$$

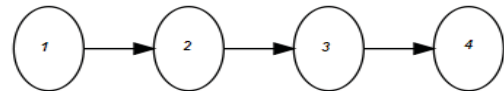
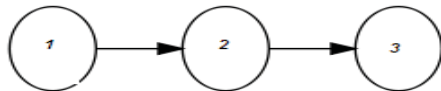


Рис. 7 – до применения правила      Рис. 8 – после применения правила

**4. Замена вершины на открытую цепочку**

В L-систему вводится правило вида:

$$\text{Padd} = \_b \rightarrow [\_b1:io+1\_b2:io]$$

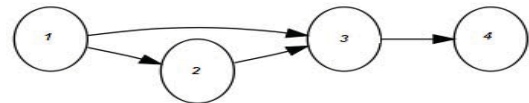
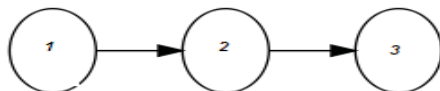


Рис. 9 – до применения правила      Рис. 10 – после применения правила

**5. Замена вершины на бинарное дерево.** В L-систему вводится правило вида:  $\text{Padd} = (\_b \rightarrow [\_x1: io \ 1 \ 2 \ 3 \ \dots + N \_b1: io\_b2: io\_b3: io \ \dots \_bN: io])$

Аналогично оператору (1), количество нейронов, на которые будет заменена вершина, также случайно. Максимальная арность дерева, вводимого в систему за одно применение оператора, составляет 3.

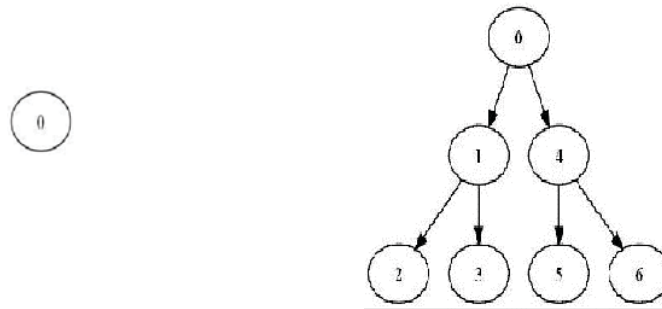


Рис. 11 – до применения правила Рис. 12 – после применения правила

Описанные выше операторы мутации могут применяться только к терминалам грамматики.

Для обучения сетей, имеющих прямые связи между несмежными слоями, был использован алгоритм случайного поиска [6]. Его высокая скорость работы обеспечивается адаптацией размера пробного шага к локальному характеру поверхности ошибки. Этот метод дал приемлемые результаты на простых задачах (задача XOR) но оказался недостаточно эффективным для более сложных задач.

Поэтому мы попытались применить *Matlab Neural network toolbox* для обучения сетей с нестандартной топологией. Он имеет возможность работы с сетями, имеющими каскадную архитектуру, в частности, функцию `newcf` для создания каскадной сети. Для обучения таких сетей целесообразно использовать алгоритм *LMA (Levenberg-Marquardt backpropagation algorithm)*. Однако функция `newcf` всегда создает полносвязную архитектуру, то есть, к каждому слою сети подсоединены все предыдущие слои. Следовательно, использование этой функции неприемлемо в рамках рассматриваемой задачи. Таким образом, возникает потребность самостоятельно конструировать необходимую архитектуру (*custom network*) в терминах объекта *Matlab Neural Network* (для более подробной информации см. [7]).

#### Особенности *Matlab Neural network toolbox*.

1. Если два слоя нейронов соединены между собой, то настраиваются все связи между их нейронами. Сети, рассмотренные в данной работе, не нуждаются в такой полной связанности. Наоборот, при их проектировании мы стараемся ее избежать.

2. Невозможно задать свойства (активационную функцию, "замораживание" весов и т.д.) для одного конкретного нейрона в слое. Это можно сделать только для слоя в целом. Поэтому мы

предлагаем использовать сеть, каждый слой которой состоит из единственного нейрона.

3. Функция инициализации слоев *"initnw"* не работает для сети, имеющей хотя бы один слой нейронов с пороговой (*hardlim, hardlims*) функцией активации. Более того, даже сеть с удачной топологией может оказаться неэффективной при плохом выборе начальных весов связей.

4. Нейроны входного слоя (*net.inputs*) не описываются в структурах (*net.layers*) и (*net.layerConnect*). Эти нейроны вынесены в отдельный слой с линейными активационными функциями.

Таким образом, при проектировании в **Matlab** сетей с неслоистой архитектурой, мы предлагаем использовать матрицу смежности графа в качестве структуры **net.layerConnect**. При этом необходимо инициализировать элементы матрицы **net.IW**, соответствующие входным нейронам, значениями "1". Также следует запретить дальнейшее изменение этой матрицы, задав **net.inputWeights{i}.Learn = 0** для индекса *i* каждого входного нейрона.

Для оценки качества алгоритма автоматического подбора топологии нейронной сети будем использовать задачу «*Two Spiral problem*». Данная задача широко используется для сравнения эффективности алгоритмов обучения сетей [4], так как считается что она сложна для FNN сетей.

На рис. 13 представлены точки, составляющие обучающую выборку, построенную по формулам (1) и (2). Точки первого класса на нем обозначены звездочкой, а точки второго класса – кружочком. Тестовая выборка для задачи представляет собой совокупность всех точек плоскости.

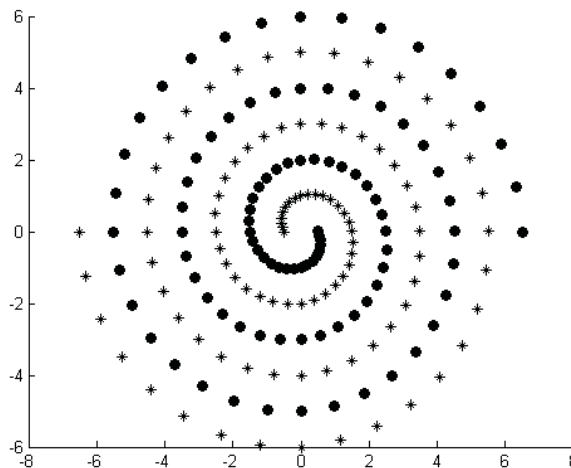


Рис. 13 – Исходные данные для задачи "Two spiral problem"



Испытания проводились как на архитектурах, заданных вручную, так и на архитектурах, полученных в результате автоматического подбора. Для решения этой задачи [8] была предложена топология 2-2-8-1. При этом каждый слой соединен со всеми следующими слоями. На рис. 14, рис.15 приведены результаты обучения этой сети и каскадной сети [4] соответственно с помощью алгоритма LMA из Matlab Neural Network toolbox.

В результате применения автоматического подбора архитектуры и последующем сравнении их со статически заданными архитектурами были получены следующие результаты :

Таблица 2

## «Сравнительный анализ эффективности обучения»

Сеть	Количество нейронов	Ошибка обучения (MSE)	Количество попыток мутации
Каскадная архитектура (рис. 15)	16	0.000001	N/A
Ручная архитектура 1 (рис. 14)	13	0.058941	N/A
Автоматическая архитектура 1 (рис. 16)	20	0.103743	34

В дальнейшем в иллюстрациях результатов будут использованы следующие цвета и обозначения :

Таблица 3

## Условные обозначения в иллюстрациях

Цвет	Значения
Тёмно-серый	Изображения тестовой выборки, отнесенные сетью ко второму классу
Светло-серый	Изображения тестовой выборки, отнесенные сетью к первому классу
Кружок	Изображения обучающей выборки, отнесенные сетью ко второму классу
Звездочка	Изображения обучающей выборки, отнесенные сетью к первому классу

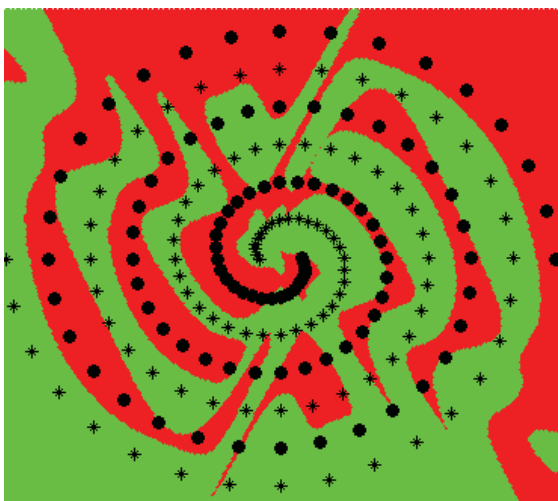


Рис. 14 – Сеть из 4х слоев

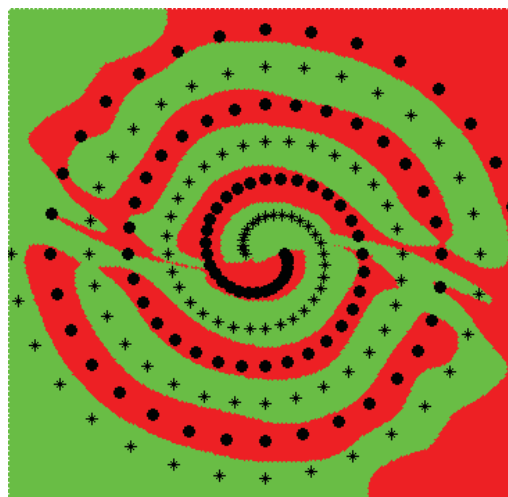


Рис. 15 – Каскад из 16 слоёв

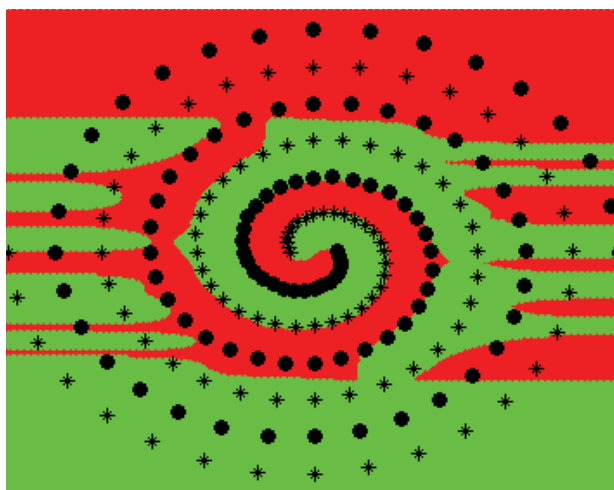


Рис. 16 – Автоматический подбор

### Выводы

Таким образом, в результате данной работы был получен инструмент для построения и дальнейшего обучения нейронных сетей прямого распространения с перекрестными связями. Для представления таких сетей была разработана компактная схема кодирования, основанная на L-системах, пригодная для дальнейшего применения в эволюционных алгоритмах. Также была создана библиотека для преобразования нейросетей, представленных в виде G0L грамматики или матрицы смежности графа топологии сети в формат *Matlab neural network toolbox*.

Результаты, полученные в результате обучения сети с автоматически подобранной топологией, несколько уступают результатам работы предложенных ранее сетей. Мы считаем, что это

обусловлено наличием ограничений максимального количества нейронов в сети и количества переписываний. Результаты автоматического подбора также могут быть улучшены за счет более детального изучения архитектуры, предложенной данной системой, более удачного выбора начальной архитектуры сети в зависимости от решаемой задачи или использованием более сложных и эффективных эволюционных алгоритмов.

Упомянутые недостатки компенсируются тем, что обучение и проектирование сети для произвольной задачи осуществляется без участия человека. Рассмотренный в данной работе подход позволяет существенно ускорить наиболее трудоемкую стадию разработки нейронных сетей – проектирование их топологий.

### ЛИТЕРАТУРА

1. Hornik K., Stinchcombe M., White H, Multilayer feedforward networks are universal approximators, 1989
2. Leshno M., Schocken S, Multilayer feedforward networks with non-polynomial activation functions can approximate any function, 1993
3. Boers E. J. W., Sprinkhuizen-Kuyper I, Combined Biological Metaphors, 2001
4. Fahlman S. E., Lebiere C., The Cascaded-correlation learning architecture, 1990
5. Montana D., Talib. S. Hussain , Adaptive reconfiguration of data networks using genetic algorithms, 2004
6. Yuret D., From Genetic Algorithms To Efficient Optimization, 1994
7. <http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/>
8. WU Youshou, ZHAO Mingsheng, A neuron model with trainable activation function and its MFNN supervised learning, 2001