УДК 621.8:681.5

М.Э. КУССУЛЬ

ГРАФЫ МОДУЛЬНЫХ НЕЙРОННЫХ СЕТЕЙ

Abstract: *The article is devoted to theoretical problems of modular neural networks representation with directed graphs. There are given definitions of cycles and the classification of cycles by proposed graph representation. Characteristics of most practically interesting types of cycles are considered. The proposed representation is meant for general analysis of modular neural networks architectures and in the first place for the analysis with automatic systems.*
Key words: *neural networks, modular neural networks, graphs of neural networks.*

Анотація: *Стаття присвячена теоретичним питанням опису модульних нейронних мереж за допомогою направлених графів. На основі запропонованого опису подані визначення циклів і проведена їх класифікація. Розглянуті властивості найбільш цікавих з практичної точки зору циклів. Запропонований опис призначений для загального аналізу архітектур модульних мереж і, в першу чергу, для аналізу архітектур автоматичними системами.*
Ключові слова: *нейронні мережі, модульні нейронні мережі, графи нейронних мереж.*

Аннотация: *Статья посвящена теоретическим вопросам описания модульных нейронных сетей при помощи направленных графов. На основе предложенного представления даны определения циклов и проведена их классификация. Рассмотрены свойства наиболее интересных с практической точки зрения типов циклов. Предложенное представление предназначено для общего анализа архитектур модульных сетей и, в первую очередь, для анализа архитектур автоматическими системами.*
Ключевые слова: *нейронные сети, модульные нейронные сети, графы нейронных сетей.*

1. Введение

В последнее десятилетие в области использования искусственных нейронных сетей активно развивается модульный подход [1]. Модульные сети, в частности, позволяют разбивать сложные задачи распознавания на простые подзадачи, решаемые отдельными модулями или подсетями, и конструировать решение проблемы в целом как совокупность решенных подзадач. Под модульной нейронной сетью мы понимаем произвольный набор алгоритмов обработки данных, включая алгоритмы искусственных нейронных сетей, объединенных для решения некоторой задачи.

Применение модульных нейронных сетей требует решения нескольких проблем как частного, так и общего характера. К проблемам частного характера относятся вопросы выбора архитектуры модульной сети для решения конкретной задачи. Например, в случае разделения на подзадачи возникает вопрос наиболее приемлемого разбиения задачи с точки зрения конечного результата ее решения. К проблемам общего характера относятся способы объединения разнотипных алгоритмов в единую модульную сеть, порядок пересчета и методы обучения модульных нейронных сетей как целого. Порядком пересчета модульной сети будем называть последовательность применения алгоритмов для обработки входных данных.

Под способами объединения разнотипных алгоритмов в единую сеть мы подразумеваем, в первую очередь, правила передачи данных между модулями сети. Использование правил обмена данными актуально, например, в тех случаях, когда разрешенный диапазон выходных значений предыдущего модуля не совпадает с разрешенным диапазоном входных значений последующего модуля. Это также относится к преобразованиям дискретных величин в непрерывные и обратно. Проблемы передачи данных решаются, например, с помощью масштабирования, смещения и аналогичных линейных преобразований, которые могут быть представлены самостоятельными модулями и включены в сеть между соответствующими алгоритмами (модулями). Большинство

вопросов, связанных с передачей данных, решено в программном симуляторе модульных нейронных сетей САПР МНН [2].

Обучение модульных нейронных сетей представляет собой самостоятельную задачу, решение которой зачастую зависит от выбранной модульной архитектуры. В настоящее время общепринятым подходом является обучение различных модулей (нейронных сетей) по отдельности, а затем конструирование общего выхода модульной сети, используя выходы обученных модулей. Альтернативные подходы к обучению модульных сетей рассматриваются в [3] и [4].

Модульная сеть может включать в себя как модули (алгоритмы), параллельно обрабатывающие входные данные, так и модули, обрабатывающие данные последовательно, друг за другом. Кроме того, возможны архитектуры модульных сетей, когда на вход модуля поступают данные, уже обработанные этим же модулем. Иными словами, архитектуры модульных сетей допускают наличие циклов.

При отсутствии циклов порядок пересчета модульной сети очевиден. В случае использования циклов, если порядок работы модулей не определен условиями решаемой задачи, возможно возникновение противоречий. Примером такого противоречия может служить триггерная схема соединения модулей, показанная на рис. 1. Результат работы такой сети будет различным в зависимости от того, какой из алгоритмов обработки данных b или c был применен первым, либо они применялись параллельно.

Определение и разрешение противоречий в модульной архитектуре является важной задачей использования модульных нейронных сетей. Особое значение эта задача приобретает при разработке интерактивных средств моделирования модульных сетей, таких как САПР МНН. Система проектирования должна автоматически определять и запрещать создание противоречивых архитектур или требовать разрешения противоречий пользователем.

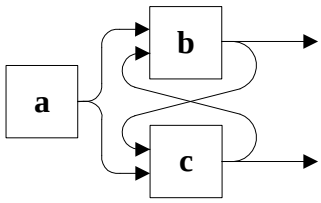

Рис. 1. Триггерная схема соединения модулей

Главной целью представленной работы является попытка предложить такой способ описания модульных нейронных сетей, который позволил бы строить непротиворечивые архитектуры и автоматически определять последовательность работы алгоритмов (модулей) сети.

В данной статье предлагается описание модульных нейронных сетей при помощи направленных графов. На основе предложенного описания даны определения и рассмотрены свойства наиболее интересных, с практической точки зрения, типов циклов в модульных сетях. Графы модульных сетей и свойства рассмотренных циклов используются для анализа архитектур нейронных сетей, в частности, автоматическими системами. Вопросам применения предлагаемого подхода будут посвящены последующие статьи по этой теме.

2. Представление модульных сетей в виде направленных графов

Модульная сеть может быть описана с помощью некоторого ориентированного графа $G(V,E)$, вершины $v \in V$ которого соответствуют модулям сети, а ребра $e \in E$ – связям между модулями.

Будем считать, что ребро графа $e(v_1,v_2)$ соответствует всем связям между модулями v_1 и v_2. То есть, если модуль v имеет n входов, причем некоторые k входов соединены с выходами модуля v_1, а остальные $n-k$ с выходами модуля v_2, то в v будут входить два ребра: $e_1(v_1,v)$ и $e_2(v_2,v)$. Аналогично, если выходы модуля v соединены со входами модулей v_1 и v_2, то из v будут выходить два ребра: $e_1(v,v_1)$ и $e_2(v,v_2)$. Путь и элементарный путь из вершины v_1 в вершину v_2 будем обозначать $w(v_1,v_2)$.

Введем в модульной сети два особых «виртуальных» модуля, представляющих собой входы и выходы всей модульной сети. Все входы модульной сети будем считать модулем входов, а все выходы – модулем выходов сети. Модули входов и выходов виртуальны в том смысле, что, в отличие от остальных модулей сети, не реализуют алгоритмов обработки данных.

По аналогии с сетями в теории графов входом или истоком (source) графа G будем считать вершину I, соответствующую виртуальному модулю входов сети. Вход сети определяется следующим условием: $s_{in}(I)=0$, где $s_{in}(I)$ – число входящих в вершину ребер. Выходом или стоком (sink) графа будем называть вершину O, соответствующую виртуальному модулю выходов. Для него справедливо утверждение $s_{out}(O)=0$, где $s_{out}(O)$ – число выходящих из вершины ребер. Следует отметить, что при этом количество входов данного модуля, собственно, выходов модульной нейронной сети, может быть произвольным.

Для того чтобы ориентированный граф G мог рассматриваться как некоторая модульная сеть, он должен удовлетворять следующим условиям:

1) граф G – слабо связный, то есть неориентированный граф, соответствующий G – связный;

2) граф G не имеет кратных ребер;

3) в графе G существует единственная вершина вход $I \in V$ и любая вершина достижима из входа, то есть: $\forall v \in V : v \neq I \Rightarrow \exists\, w(I,v)$ и, следовательно, $\forall v \in V : v \neq I \Rightarrow s_{in}(v) \geq 1$;

4) в графе G существует единственная вершина выход $O \in V$ и выход достижим из любой вершины, то есть $\forall v \in V : v \neq O \Rightarrow \exists\, w(v,O)$ и, следовательно, $\forall v \in V : v \neq O \Rightarrow s_{out}(v) \geq 1$.

Последние два условия очевидны, если принять во внимание, что модульная нейронная сеть строится для обработки данных. Если сеть не удовлетворяет условию 3, то существуют модули, которые не получают входные данные. Условие 4 гарантирует, что результаты работы любого алгоритма, прямо или после обработки другими модулями, поступят на выход, то есть такой алгоритм не является излишним. Заметим, что с точки зрения графов, последние два условия гарантируют отсутствие «висящих» вершин.

В отличие от принятого в теории графов определения, будем рассматривать путь w из вершины a в вершину b как множество вершин и соединяющих их ребер, не включая конечную вершину. Согласно такому определению, на рис. 2 можно выделить шесть путей: $w_1(a,b)=\{a;e_1\}$; $w_2(a,c)=\{a;e_1;b;e_3\}$; $w_3(a,c)=\{a;e_1;b;e_2;b;e_3\}$; $w_4(b,b)=\{b;e_2\}$;

$w_5(b,c)=\{b;e_3\}$ и $w_6(b,c)=\{b;e_2;b;e_3\}$. Заметим, что путь $w(a,b)=\{a;e_1;b;e_2\}$ не существуют, поскольку вершина b не может принадлежать пути по принятому определению. Аналогично, цепью будем называть элементарный путь (без самопересечений), не включающий конечную вершину. Соответственно, на рис. 2 можно выделить только четыре цепи: w_1, w_2, w_4 и w_5. Цепь w_4 существует, поскольку начальная вершина включается.

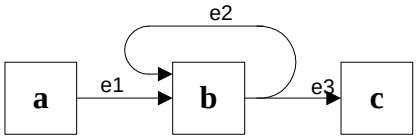

Рис. 2. Пример циклов

Как будет видно из дальнейшего, уточнение определения пути преследует следующую цель: если рассматривать подграфы модульной сети как множества вершин и дуг, то исключение конечной вершины в определении пути позволяет использовать операцию разности без явного указания принадлежности вершин результирующему множеству. То есть, если имеется путь, представленный множеством вершин и дуг $w(a,c)=\{a;e_1;b;e_2\}$, то такой путь может быть представлен двумя непересекающимися подмножествами $w_1(a,b)=\{a;e_1\}$ и $w_2(b,c)=\{b;e_2\}$, каждое из которых также является путем согласно принятому определению.

Для дальнейшего изложения нам необходимо ввести два понятия: проекции одного модуля на другой P и степень неопределенности, или неопределенность модуля по входам U.

Определение 1: **Проекцией** $P(b,a)$ (projection) вершины (модуля) $a\in V$ на вершину (модуль) $b\in V$ будем называть такой ориентированный подграф графа $G(V,E)$, который включает все возможные цепи $w(b,a)$ из b в a:

$$P(b,a)=\bigcup w(b,a);\quad \forall w(b,a)=\{v_1,v_2,\ldots,v_n\}: v_1=b, v_{n+1}=a;\ \ v_i\neq v_j\quad \forall i,j=\overline{1,n},$$

то есть в подграф входят все вершины и соответствующие ребра всех цепей из b в a, кроме вершины a, согласно принятому нами определению путей.

Одной из наиболее важных проекций является проекция произвольного модуля a на входы модульной сети I, которая, в соответствии с Определением 1, записывается как $P(I,a)$.

Определение 2: Если каждой вершине $v\in V$ поставить в соответствие параметр $t(v)$ такой, что $t(v)=1$ в том случае, если выходы соответствующего модуля сети на данном шаге еще не вычислены, и равен нулю, если модуль уже посчитан, то неопределенностью (uncertainty) вершины a будем называть сумму

$$U(a)=\sum_{v\in P(I,a)} t(v).$$

При этом неопределенность модуля входов $U(I)$ всегда равна нулю. Поскольку модуль входов виртуальный, то считаем, что его выходы всегда определены (вычислены).

Необходимость введения понятия «неопределенность» непосредственно следует из поставленных целей. Для проверки модульных архитектур на непротиворечивость и построение

последовательности работы алгоритмов (модулей) сети необходимо ввести критерий сравнения, какой из модулей должен быть посчитан раньше. Таким критерием может служить введенное понятие неопределенности.

Аксиома 1: Из двух модулей сети, при прочих равных условиях и если порядок счета модулей не определен специальными требованиями, первым должен быть посчитан модуль с меньшей неопределенностью.

Проиллюстрируем требование данной аксиомы на примере последовательного соединения модулей, показанного на рис. 3. В начале работы модульной сети выходы модуля I известны по определению, а выходы модуля a не определены. Тогда, согласно Определению 2, неопределенность модуля a равна нулю, а неопределенность модуля b равна единице. Следовательно, согласно Аксиоме 1, модуль a должен быть посчитан раньше модуля b, что для последовательного соединения модулей очевидно.

Рис 3. Схема последовательного соединения модулей

Необходимость Аксиомы 1 становится еще более очевидной для обучения модульной нейронной сети. Если по пути от входов к некоторой нейронной сети (модулю) b встречается необученный модуль (нейронная сеть) a, то обучение модуля b не имеет смысла, пока не обучена нейронная сеть a.

3. Циклы в модульных сетях

С точки зрения прикладных задач, циклы представляют особый интерес. Если исходные данные организованы в виде временных последовательностей, можно говорить о «рекуррентных» циклах. Кроме того, если входящие в цикл модули реализуют ассоциативные поля или другие алгоритмы, для работы которых требуется несколько итераций с одним входным вектором данных, то можно говорить об итерационных циклах. Таким образом, кроме различия в архитектурах, цикл может быть «рекуррентным» либо «итерационным».

Для описания рекуррентных циклов необходимо вводить понятия такта работы сети, для итерационных достаточно говорить о количестве «обходов» цикла, где под обходом цикла подразумевается однократный пересчет всех модулей, входящих в этот цикл. Обсуждение работы модульных сетей с временными последовательностями требует введения дополнительных определений и правил. Поэтому в данной работе ограничимся обсуждением общих определений типов циклов, возможных при разработках архитектур модульных сетей. Подробное исследование свойств таких циклов будет обсуждаться в последующих статьях. В дальнейшем везде, где рассматриваются вопросы построения очередей для установления порядка пересчета модулей в сети и алгоритмы построения очередей, предполагается, что речь идет об итерационных циклах.

3.1. Основные типы циклов и базовые определения

В первую очередь нам понадобится общее определение цикла в модульных сетях.

Определение 3: Циклом с начальной вершиной a будем называть такой ориентированный подграф $C_a(V_a,E_a)$ графа модульной сети $G(V,E)$, который включает все возможные пути $w(a,a)$ из вершины a в саму себя:

$$C_a(V_a,E_a)=\bigcup w(a,a);\quad \forall w(a,a)=\{v_1,v_2,\ldots,v_n\}: v_1=a, v_{n+1}=a.$$

В отличие от определения проекций, в определении циклов фигурируют пути, а не цепи. Введенное определение отличается от определения, принятого в теории графов, где циклом называют путь, для которого начальная вершина совпадает с конечной [5]. Определение 3 обобщает понятие цикла в том смысле, что оно включает все возможные пути вида $w(a,a)$. Не трудно заметить, что циклы, согласно Определению 3, соответствуют сильно связанным компонентам в орграфе и обладают очевидным свойством: $C_a \bigcap C_b = \varnothing$.

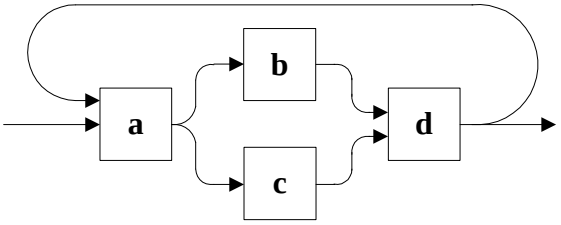

Рис. 4. Схема простого цикла

Необходимость определения цикла как набора всех путей иллюстрируется архитектурой, представленной на рис. 4. Очевидно, что при пересчете модулей, входящих в цикл $C_a: V_a=\{a,b,c,d\}$, прежде чем считать модуль d, требуется посчитать оба модуля b и c. Заметим, что такой же порядок счета требует и Аксиома 1.

Из определения цикла непосредственно следует, что виртуальные модули (входы и выходы) никогда не входят ни в один цикл, поскольку модуль входов не имеет входящих ребер, а модуль выходов – выходящих. Кроме того, согласно условиям, накладываемым на граф, представляющий модульную сеть, любой цикл обязательно имеет как входящие, так и выходящие внешние ребра. То есть такие ребра, которые связывают вершины цикла с вершинами, не принадлежащими этому циклу.

Для любого цикла в модульной нейронной сети можно выделить три группы особых вершин. Вершины, которые имеют входящие ребра от вершин, не принадлежащих этому циклу. Вершины, имеющие выходящие ребра, которые входят в вершины за пределами цикла. И вершины, выходящие ребра которых являются входящими ребрами начальной вершины цикла. Особый интерес представляют некоторые из перечисленных вершин. Например, из всех возможных циклов для анализа и работы с модульными сетями интерес представляют те, у которых начальная вершина является первой, в которую поступают данные от входов модульной сети.

Дадим формальное определение вершин, которые будут использоваться в дальнейшем. Для этого воспользуемся понятием длины пути $d(w)$, которое в данном случае будем считать просто как число ребер в этом пути.

Определение 4: Начальной вершиной f (first) цикла $C(V,E)$ будем называть такую вершину, для которой $d_f=\min\limits_{v\in V}\ \min\limits_{w=w(I,v)} d(w)$. Конечной вершиной l (last) цикла будем называть такую

вершину, для которой $d(w(l,f))=1$. Граничной вершиной t (terminal) будем называть такую вершину, для которой выполняются условия

$$t:\left(\exists\ e(t,p),\ p\notin V\right)\&\left(\forall\ v\in V:P(f,t)\subset P(f,v)\Rightarrow\overline{\exists}\,e(v,g),\ g\notin V\right),$$

то есть t – самая дальняя от начальной вершина цикла, имеющая связь (связи) с не принадлежащими циклу вершинами.

Заметим, что приведенные определения не гарантируют единственность каждой из вершин. Тем не менее такое определение особых вершин позволяет провести дальнейшую классификацию циклов.

Введем определения для двух типов циклов, принципиально важных с точки зрения счета модульных сетей и рассмотрим некоторые из свойств таких циклов.

Определение 5: Обычным или обыкновенным циклом (ordinary cycle) с начальной вершиной a будем называть такой цикл $Co_a(V_a,E_a)$, для которого не существует ни одной пары вершин таких, что они принадлежат проекциям друг друга на вход, то есть

$$\forall\ v_1,v_2\in V_a:v_1\in P(I,v_2)\quad\Rightarrow\quad v_2\notin P(I,v_1).$$

Перекрестным циклом (crossed cycle) с начальной вершиной b будем называть такой цикл $Cc_b(V_b,E_b)$, который содержит хотя бы одну пару вершин, входящих в проекции друг друга на вход, то есть

$$\exists\ v_1,v_2\in V_b:v_1\in P(I,v_2)\quad\&\quad v_2\in P(I,v_1).$$

Введение понятия обыкновенных циклов необходимо для установления порядка пересчета модулей в сети и в, первую очередь, чтобы отделить все возможные варианты триггерных схем (рис. 1). Нетрудно проверить, что цикл, изображенный на рис. 1, является перекрестным и содержит две начальные вершины, удовлетворяющие Определению 4.

Если обозначить множество всех возможных циклов с начальными вершинами, удовлетворяющими Определению 4, через $\Omega\{C_f\}$, то для множеств обыкновенных и перекрестных циклов очевидны соотношения:

$$\Omega\{Cc\}\cap\Omega\{Co\}=\varnothing;\qquad\Omega\{Cc\}\cup\Omega\{Co\}=\Omega\{C_f\}. \tag{1}$$

Обыкновенный цикл может быть представлен только единственным образом, в отличие от циклов общего вида, которые тождественны с точностью до выбора начальной вершины. Перекрестные циклы этим свойством не обладают.

3.2. Свойства циклов и дополнительные определения

Разработка правил для анализа архитектур модульных сетей автоматической системой существенно упрощается, если строить правила анализа на основе свойств, присущих графам модульных сетей. Общие свойства циклов будут подробно рассмотрены в следующей статье. Здесь же рассматриваются определения и свойства некоторых частных случаев, имеющих особое значение при проектировании модульных нейронных сетей, содержащих циклы. Рассматриваемые

частные случаи представляют самостоятельный интерес с точки зрения построения правил для пересчета таких сетей.

Определение 6: Простым циклом будем называть такой обыкновенный цикл, для всех вершин которого любой замкнутый путь проходит через начальную вершину этого цикла. То есть цикл $Co_f(V_f, E_f)$ простой, если $\forall\, w(v,v): v \in V_f \quad \Rightarrow \quad f \in w(v,v)$.

Заметим, что простой цикл может содержать более одного пути, как, например, на рис. 4.

Утверждение 1: Для простых циклов $Co_f(V_f, E_f) \equiv P(f, f)$. То есть простой цикл совпадает с проекцией начальной вершины на саму себя.

Доказательство: Покажем, что для простого цикла все пути из начальной вершины в саму себя являются элементарными, то есть цепями. Пусть найдется хотя бы один не элементарный путь $w(f, f)$. Это означает, что существует вершина v, которая входит в этот путь хотя бы два раза: $w(f, f) = \{f, ..., v, ..., v, ...\}$. Следовательно, в цикле существует путь из этой вершины в нее же, который не содержит начальную вершину $\exists\, w(v,v) \subset Co_f : f \notin w$, что противоречит определению простого цикла. Таким образом, в простом цикле все пути из начальной вершины в нее же являются элементарными и, по определению, образуют проекцию. То есть, простой цикл совпадает с проекцией начальной вершины на саму себя.

Определение 7: Триггерным циклом (trigger cycle) будем называть такой цикл, который содержит хотя бы две вершины a и b, имеющие равные кратчайшие пути от входов. То есть цикл $C_T(V, E)$ триггерный, если $\exists\, a, b \in V : \quad \min_{w(I,a)}(d_a) = \min_{w(I,b)}(d_b)$.

Тривиальный пример такого цикла представлен на рис. 1.

3.2.1. Пересечения простых циклов

Самостоятельный интерес представляют некоторые виды циклов, которые можно представить как совокупность пересекающихся простых циклов. Такое представление существует, если вершины и ребра исходного цикла можно разбить на два или более множества, удовлетворяющих четырем условиям:

1) каждая вершина и ребро исходного цикла принадлежит хотя бы одному из этих множеств;

2) каждое множество вершин и ребер образует простой цикл, если удалить все ребра исходного цикла, которые не принадлежат данному множеству;

3) каждое множество имеет непустое пересечение не менее чем с одним другим множеством (имеет общие вершины и ребра);

4) ни одно из этих множеств не является подмножеством другого.

Условие, что каждое из этих множеств вершин и ребер должно представлять собой цикл, является принципиальным с точки зрения анализа модульной сети автоматической системой. При несоблюдении данного условия и использовании методики замещения циклов «метавершинами» можно получить непредсказуемые результаты. Методика замещения обсуждается в последующих статьях.

Ориентированный подграф, образованный общими вершинами и ребрами, будем называть пересечением циклов. Таким образом, если цикл C представим в виде множества пересекающихся простых циклов $Co_1,...,Co_N$, то

$$C = Co_1 \bigcup ... \bigcup Co_N; \quad \forall i \in [1,N] \quad \exists\, j \neq i : \{Co_i \bigcap Co_j \neq \varnothing \mid Co_i \not\subset Co_j \;\&\; Co_j \not\subset Co_i\}.$$

Пример цикла, который можно представить в виде пересекающихся простых циклов, показан на рис. 5, где 5.1 исходный цикл, а 5.2 и 5.3 иллюстрируют два возможных разделения на простые циклы, объединение которых дает исходный цикл. Пересечения циклов выделены жирными линиями.

Не всякий цикл можно представить в виде пересечения простых циклов. Например, тривиальный триггерный цикл (рис. 1) не имеет такого представления. Кроме того, в некоторых случаях возможно только единственное представление в виде пересечения простых циклов. Пример такого цикла показан на рис. 6, где жирными линиями выделены общие вершины и ребра.

Простые циклы на рис. 6 будут представлены вершинами $V_1 = \{a,d,e,f\}$ и $V_2 = \{b,d,e,f\}$. Очевидно, что при любом другом выделении вершин в получаемые простые циклы не попадут одно или два ребра исходного цикла, что не удовлетворяет условиям представления в виде пересекающихся простых циклов.

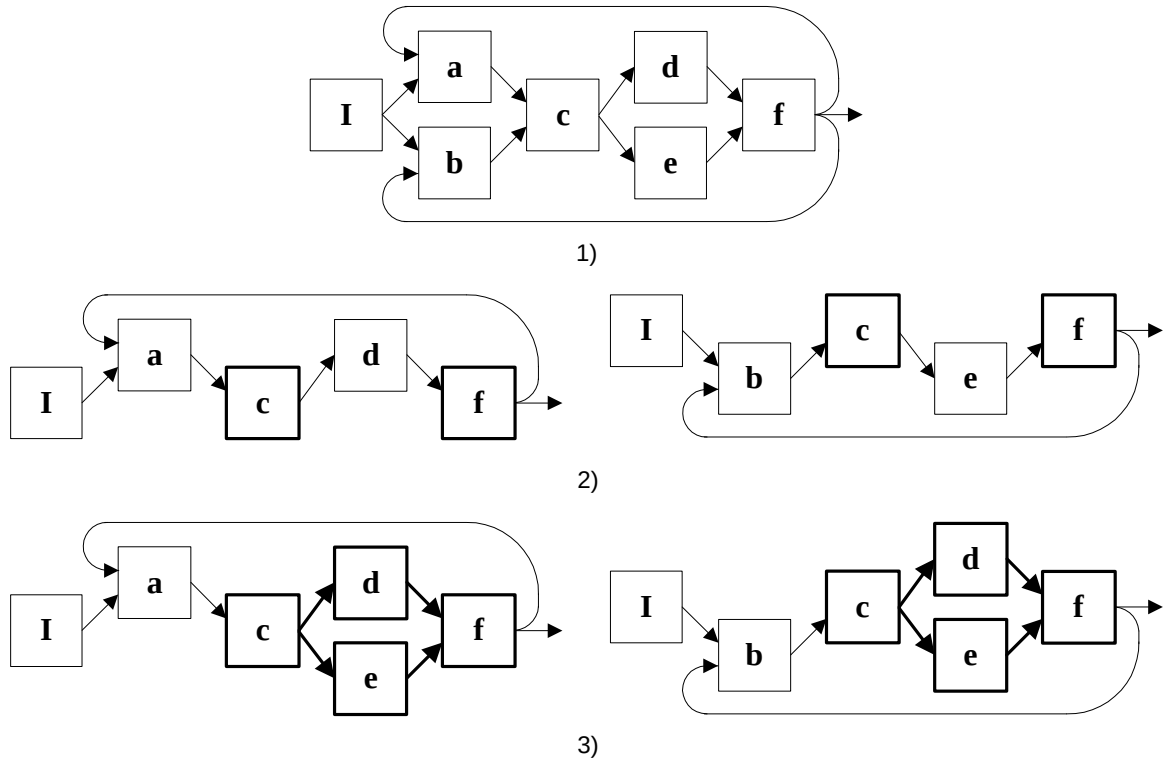

Рис. 5. Выделение простых циклов

Не любые пересечения простых циклов представляют интерес с точки зрения анализа модульных нейронных сетей. Рис. 5.2 иллюстрирует случай, когда произвольное разбиение на простые циклы приводит к нарушению Аксиомы 1. Действительно, с точки зрения выполнения Аксиомы 1 неважно, который из модулей a или b будет посчитан раньше. Однако прежде чем

считать модуль f, должны быть посчитаны модули d и e. Чтобы конкретизировать полезные виды пересечений простых циклов нам понадобятся определения «сцепленных» и «вложенных» циклов.

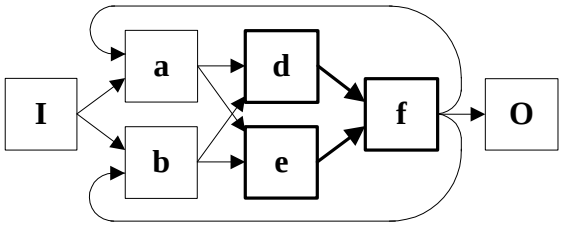

Рис. 6. Цикл, имеющий единственное представление в простых циклах

Определение 8: Сцепленными циклами будем называть такие простые циклы Co_1 и Co_2, которые имеют одну или более общих вершин. Причем, если для двух общих вершин a и b вершина a принадлежит проекции вершины b на начальную вершину любого из этих циклов, то проекция $P(a,b)$ целиком принадлежит каждому из циклов Co_1 и Co_2. То есть:

$$\forall a,b : a,b \in Co_1,Co_2 \quad a \in P(f,b) \;\Rightarrow\; P(a,b) \subset Co_1 \;\&\; P(a,b) \subset Co_2,$$

где f – начальная вершина любого из циклов Co_1 или Co_2.

Данное определение легко обобщается на случай пересечения любого количества простых циклов.

Определение 9: Вложенными циклами будем называть такие сцепленные циклы Co_1 и Co_2, для которых начальная f_2 и граничная t_2 вершины цикла Co_2 принадлежат циклу Co_1, при этом цикл Co_1 удобно называть внешним, а Co_2 – внутренним циклом.

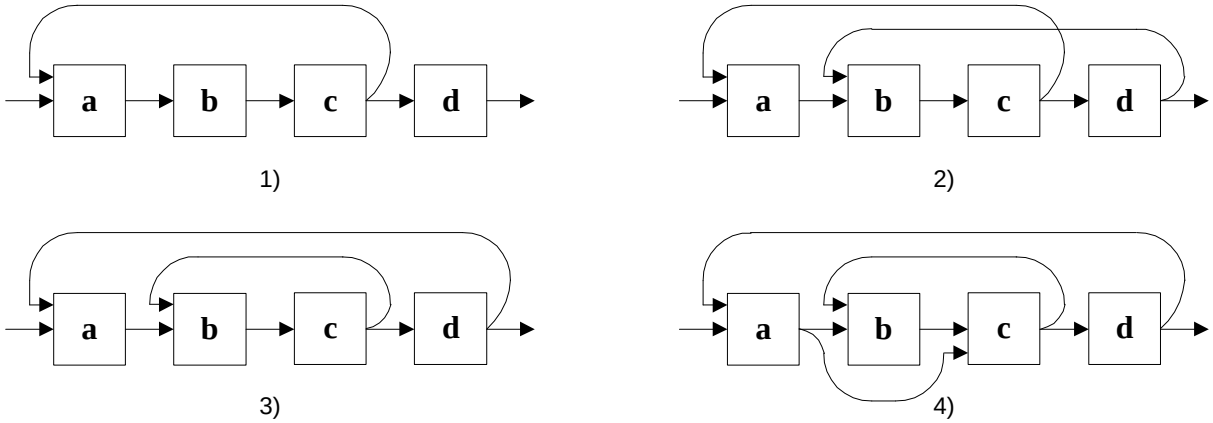

Рис. 7. Типы циклов: 1-3 обыкновенные, 1 - простой, 2-3 сцепленные, 3 - вложенные, 4 - триггерный

Примером сцепленных циклов является цикл, показанный на рис. 5. Однако чтобы рассматривать данный цикл как случай сцепленных простых циклов согласно Определению 8, допустимо лишь разделение, показанное на рис. 5.3, то есть сцепленными являются простые циклы, содержащие вершины $V_1 = \{a,c,d,e,f\}$ и $V_2 = \{b,c,d,e,f\}$.

Как для сцепленных, так и для вложенных циклов, не важно, в проекцию какой из множества возможных граничных вершин попадают общие вершины пересекающихся циклов. Рис. 7 иллюстрирует простейшие архитектуры различных типов циклов.

Введение понятия сцепленных циклов в первую очередь связано с возможностью представления Ассоциативно-Проективных Нейронных Сетей (АПНС) в виде модульной нейронной сети, где каждый модуль представляет собой ассоциативное или проективное поле [6]. Поскольку в АПНС предусмотрена как возможность композиции, так и декомпозиции, то сцепленные циклы и предоставляют возможность описать распространение возбуждения нейронной сети в обе стороны.

В свою очередь вложенные циклы позволяют моделировать, например, рекуррентные циклы, включающие итеративные ассоциативные сети.

3.2.2. Основные свойства сцепленных и вложенных циклов

Теорема 1: Если начальная вершина одного из сцепленных циклов принадлежит другому, то такие циклы образуют обыкновенный цикл.

Доказательство: Чтобы цикл был обыкновенным, необходимо показать, что в цикле не существует никаких двух вершин, принадлежащих проекциям друг друга на входы.

Пусть существует цикл C_f, образованный двумя сцепленными простыми циклами $Co_{f1}(V_{f1},E_{f1})$ и $Co_{f2}(V_{f2},E_{f2})$, причем начальная вершина второго цикла принадлежит первому, и $G(V_g,E_g)=Co_{f1}\bigcap Co_{f2}$ – их пересечение. Очевидно, что $f2\in G$ и вершина $f1$ является начальной вершиной цикла C_f, поскольку имеет минимальное расстояние от входов.

В сцепленных циклах C_{f1} и C_{f2} можно выделить два набора вершин: вершины, принадлежащие пересечению, и вершины, принадлежащие только одному из этих циклов. Пусть пересечение содержит N вершин g_n, имеющих выходящие ребра к вершинам циклов C_{f1} и C_{f2}, не принадлежащим пересечению. Поскольку (см. Утверждение 1) простые циклы представляют собой набор цепей, то циклы C_{f1} и C_{f2} можно представить в виде

$$Co_{f1}=P(f1,f1)=\left(\bigcup_{n=1}^{N}P(g_n,f1)\right)\cup\left(\bigcup_{n=1}^{n}P(f1,g_n)\right)\cup P'(f1,f1);$$

$$Co_{f2}=P(f2,f2)=\left(\bigcup_{n=1}^{N}P(g_n,f2)\right)\cup\left(\bigcup_{n=1}^{N}P(f2,g_n)\right).$$

Проекция P' соответствует случаю, когда цикл C_{f1} содержит замкнутые цепи, не проходящие через пересечение. Цикл C_{f2} не может содержать таких цепей, поскольку в простом цикле все цепи проходят через начальную вершину по определению, а $f2$ принадлежит пересечению G.

Проекцию P' можно не рассматривать, поскольку, если она существует, то цикл C_{f1}, в свою очередь, можно представить как пересечение простых циклов P' и P'', и результаты,

полученные для пересечения P'' с C_{f2}, распространить на все случаи пересечений. Так как $f2 \in G$, то по определению сцепленных циклов, проекции $P(f2, g_n)$ целиком принадлежат пересечению. Тогда верно следующее:

$$\left(\bigcup_{n=1}^{N} P(f2, g_n)\right) \equiv G \subseteq \left(\bigcup_{n=1}^{N} P(f1, g_n)\right).$$

Причем множества равны только в том случае, если начальные вершины совпадают.

Тогда сцепленный цикл C_f можно представить в виде

$$C = \left(\bigcup_{n=1}^{N} P(g_n, f2)\right) \cup \left(\bigcup_{n=1}^{N} P(g_n, f1)\right) \cup \left(\bigcup_{n=1}^{N} P(f1, g_n)\right). \quad (2)$$

Причем

$$\left(\bigcup_{n=1}^{N} P(g_n, f2)\right) \cap \left(\bigcup_{n=1}^{N} P(g_n, f1)\right) \cap \left(\bigcup_{n=1}^{N} P(f1, g_n)\right) = \varnothing.$$

Каждая из этих скобок описывает подмножество цепей, целиком принадлежащих простому циклу. По определению, простой цикл является обыкновенным и не содержит ни одной пары вершин, принадлежащих проекции друг друга на вход, а, следовательно, и подмножества этих циклов не содержат таких вершин. То есть сцепленные циклы образуют обыкновенный цикл, если начальная вершина одного из них принадлежит другому.

Следствие 1.1: Если начальные вершины сцепленных циклов совпадают, то такие циклы представляют собой один простой цикл.

Доказательство: По определению простого цикла, все замкнутые цепи должны проходить через начальную вершину. Если в сцепленных циклах начальные вершины совпадают $f1 = f2 = f$, то, согласно (2), сцепленные циклы представимы в виде

$$C = \left(\bigcup_{n=1}^{N} P1(g_n, f)\right) \cup \left(\bigcup_{n=1}^{N} P2(g_n, f)\right) \cup \left(\bigcup_{n=1}^{N} P(f, g_n)\right),$$

где последняя скобка описывает участки цепей, принадлежащие пересечению G, $P1$ описывает участки цепей, принадлежащие только циклу C_{f1}, и $P2$ – только циклу C_{f2} соответственно. То есть любая замкнутая цепь включает хотя бы одну вершину пересечения, а, следовательно, и общую начальную вершину.

Следствие 1.2: Если пересечение двух простых циклов содержит цикл, то этот цикл простой и начальные вершины пересекающихся циклов совпадают:

$$C_g \subseteq Co_{f1} \cap Co_{f2} \quad \Rightarrow \quad g = f1 = f2 \quad \& \quad f \in w(v,v) \quad \forall v \in V_f.$$

Доказательство: Пусть пересечение G простых циклов Co_{f1} и C_{f2} содержит цикл $C_g \subseteq Co_{f1} \cap Co_{f2}$, который является подмножеством вершин и ребер каждого из них. Поскольку Co_{f1} простой, то все замкнутые пути проходят через начальную вершину $f1$, и эта вершина

является начальной и для цикла C_g, поскольку имеет минимальное расстояние до входов, то есть $g = f1$. Аналогично, для цикла C_{f2} имеем $g = f2$, и, следовательно, начальные вершины пересекающихся циклов совпадают $f2 = f1$.

Согласно Следствию 5.1, пересечение циклов с совпадающими начальными вершинами образует простой цикл, тогда и его подмножество C_g является простым циклом.

Утверждение 2: Если два простых цикла сцеплены и ни одна из начальных вершин не принадлежит пересечению этих циклов, то такие сцепленные циклы образуют перекрестный цикл.

Доказательство этого утверждения следует из общих свойств обыкновенных циклов и будет приведено в следующей статье. Заметим, что из Теоремы 1 непосредственно следует, что вложенные циклы образуют один обыкновенный цикл.

4. Выводы

В статье приведены новые теоретические результаты, полученные на основе представления модульных нейронных сетей в виде направленных графов. Особое значение предлагаемое описание имеет для систем автоматического анализа архитектур нейронных сетей и, в частности, систем автоматического проектирования. Наиболее интересные результаты относятся, к архитектурам модульных сетей, содержащих циклы. Очевидно, что для счета приведенных архитектур требуются различные допущения о правилах и порядке обхода циклов. В качестве основы таких правил могут быть использованы свойства предложенных типов циклов.

В модульных нейронных сетях циклы различаются не только по типам архитектур, но также по типам обрабатываемых данных и используемым алгоритмам нейронных сетей. Рассмотренные в статье виды циклов позволяют учесть специфику обрабатываемых данных и нейросетевых алгоритмов в виде соответствующих модульных архитектур.

Предложенная теория была успешно использована в подсистеме автоматического анализа модульных архитектур САПР МНН [2]. Общие свойства циклов, алгоритмы обнаружения и анализа циклов, алгоритмы построения очередей счета модулей в сетях и другие вопросы практического применения предложенной теории будут изложены в последующих публикациях.

СПИСОК ЛИТЕРАТУРЫ

1. Галинская А.А. Модульные нейронные сети: обзор современного состояния разработок // Математические машины и системы. – 2003. – № 3, 4. – С. 87 – 102.
2. Резник А.М., Куссуль М.Э., Сычев А.С., Садовая Е.Г., Калина Е.А. Система проектирования модульных нейронных сетей САПР МНС // Математические машины и системы. – 2002. – № 3. – С. 28 – 37.
3. Kussul M.E., Galinskaya A.A. Comparative study of modular classifiers and its training // Proc. of the X-th International Conference “Knowledge – Dialog – Solution” KDS 2003. – Varna, Bulgaria. – 2003. – 16-26 June. – P.168 – 174.
4. Галинская А.А. Архитектура и обучение модульных классификаторов для прикладных задач // Математические машины и системы. – 2003. – № 2. – С. 77 – 86.
5. Дистель Р. Теория графов. – Новосибирск: Изд-во Ин-та математики, 2002. – 336 с.
6. Куссуль Э.М. Ассоциативные нейроподобные структуры. – Киев: Наукова думка, 1992. – 143 с.