УДК 006.72 **DOI** 10.15622/sp.54.9

А.А. Воевода, Д.О. Романников

СИНТЕЗ НЕЙРОННОЙ СЕТИ ДЛЯ РЕШЕНИЯ ЛОГИКО-АРИФМЕТИЧЕСКИХ ЗАДАЧ

Воевода А.А., Романников Д.О. **Синтез нейронной сети для решения логико-арифметических задач.**

Аннотация. Одна из основных проблем, стоящих перед разработчиком системы с нейронной сетью — выбор структуры нейронной сети, которая могла бы решать поставленные задачи. В настоящее время нет однозначных рекомендаций по выбору такой структуры и таких параметров, как: количество слоев, количество нейронов в слое, тип нелинейности нейрона, метод обучения, параметры метода обучения и другие.

В статье рассматривается подход к синтезу нейронной сети для класса логико-арифметических задач, основанный на формировании сети из предпостроенных элементарных функций. Новизна предлагаемого подхода заключается в формировании нейронной сети по известному алгоритму с использованием предварительно построенных функций. Таким образом, в статье изначально построены элементарные логико-арифметические функции, такие как «и», «или», «исключающее или», «и-не», «или-не», «≥», «≤», «>», «<», которые можно использовать для решения более сложных задач. Также приведен пример решения задачи построения функции по выбору максимального числа из четырех чисел, представленных в бинарном виде тремя разрядами. Синтез нейронной сети вышеприведенным способом выполняется с дальнейшей целью получения обобщенной структуры нейронной сети.

Ключевые слова: нейронные сети, машинное обучение, логико-арифметические задачи, синтез нейронной сети.

1. Введение. В настоящее время нейронные сети широко распространены в решении различных задач: распознавание изображений [1] и речи [2, 3], задачах классификации [4-7], кластеризации [4-6, 8], задачах управления [9] и других.

При решении вышеприведенных задач при помощи нейронных сетей возникает ряд проблем, среди которых можно отметить отсутствие однозначных рекомендаций по выбору структуры нейронной сети, избыточность (*redundancy*) [4-7], недообучение (*underfitting*) и переобучение (*overfitting*) моделей [5, 6], выбор начальных условий [6] и другие.

В частности, при рассмотрении классической задачи машинного обучения — XOR [4, 5] — видно, что избыточность нейронных сетей проявляется в том, что задача решается при различных значениях весовых коэффициентов и значениях смещения нейронов. В [10] приводится решение XOR задачи при помощи нейронной сети с одним скрытым нейронным слоем (рисунок 1).

В практически решаемых задачах, где число параметров может превышать десятки миллионов обучаемых параметров [1-3, 9, 11] и нет однозначных рекомендаций для выбора числа слоев, числа «фильтров», значений ядра свертки, избыточность может привести к тому, что система

«попадает» в локальный минимум или в лучшем случае значительно увеличивается время и затрачиваемые ресурсы на обучение сети.

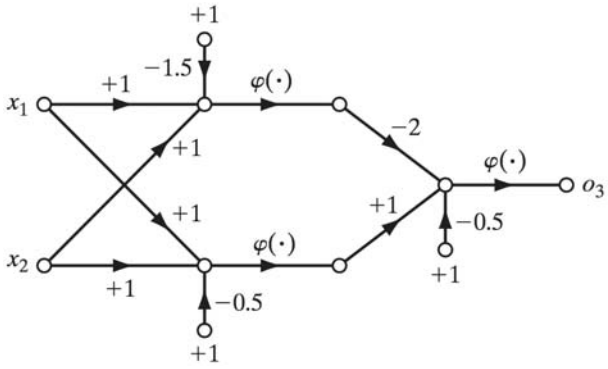

Рис. 1. Схема нейронной сети с тремя нейронами для решения *XOR* [17]

Также стоит отметить неоднозначность при выборе метода обучения, которая выражается в выборе самого метода обучения, значения параметра обучения (*learning rate*), количества эпох обучений (*epoch*), начальных условий параметров обучения (от их выбора зависит к какому из многих минимумов «сойдется» нейронная сеть при обучении).

Все вышеприведенные недостатки, заключающиеся в неоднозначности выбора параметров системы, на практике решаются при помощи проведения множества экспериментов и сравнении полученных результатов. Среди самых распространенных техник и методик можно выделить обнуление случайно выбранных весов при обучении (*dropout и dropConnect)* [12], ранний останов (*early stopping*) [13, 14], регуляризация [4, 6], кадрирование (*cropping*) и другие.

В отличие от используемого на практике подхода, основанного на выборе одного из известных вариантов структуры нейронной сети и дальнейшего обучения, в данной работе предлагается построение структуры нейронной сети исходя из анализа алгоритма решения рассматриваемой задачи. Иллюстрация предлагаемого подхода выполнена на примере логико-арифметической задачи, а именно рассмотрены примеры построения функций логических операций «и», «или», «не», исключающего или» и функции выбора максимального числа из массива чисел. При решении данной задачи все входные и выходные данные представлены в виде чисел, где под «единицей» подразумевается

число из диапазона 0.9-1.1, под «нулем» — число из диапазона -0.1-0.1. Это условие необходимо для того, чтобы при прохождении через цепочку нейронов выходное значение выходных нейронов не смещалось.

В разделе 2 приводится реализация основных логико-арифметических функций при помощи нейронных сетей, которые затем могут быть использованы при синтезе нейронной сети для решения более сложных логико-арифметических задач. В частности, в разделе 3 приведен пример решения задачи выбора максимального числа из массива из четырех трехразрядных чисел, представленных в бинарном виде. В разделе 4 приведен сравнительный анализ полученной нейронной сети с нейронными сетями прямого распространения с четырьмя скрытыми слоями для различного числа нейронов в слое. Статья заканчивается разделом с выводами.

2. Реализация основных логико-арифметических функций. Для решения логико-арифметических задач существует необходимость использовать такие базовые функции, как «и», «или», «исключающее или», «и-не», «или-не», «≥», «≤», «>», «<» (например, $x_1 > x_2$, если $x_1 : 1,\ x_2 : 0$).

Для того чтобы нейронные сети по вычислительным свойствам не вырождались (обусловленность — *condition number* — неудовлетворительная), весовые коэффициенты и смещения в среднем должны не слишком отличаться от единицы по модулю. Это следует из того, что при возрастании размеров системы с вычислительной точки зрения ее свойства существенно ухудшаются. Одним из возможных способов ограничения значений коэффициентов является изменение наклона (сжатие) нелинейной функции (измененный наклон нелинейной функции нейронной сети может негативно сказаться при обучении сети, но так как предлагаемый синтез нейронной сети основан на использовании уже пред-рассчитанных функций, то данным недостатком можно пренебречь). В статье в качестве базовой функции используется сигмоидальная (*sigmoid*) функция $f(x) = 1/(1+e^{(-x)})$ как наиболее распространенная в нейронных сетях, график которой приведен на рисунке 2. Далее для различных примеров она будет изменяться в соответствии со спецификой задачи. Также можно использовать и другие монотонно изменяющиеся функции.

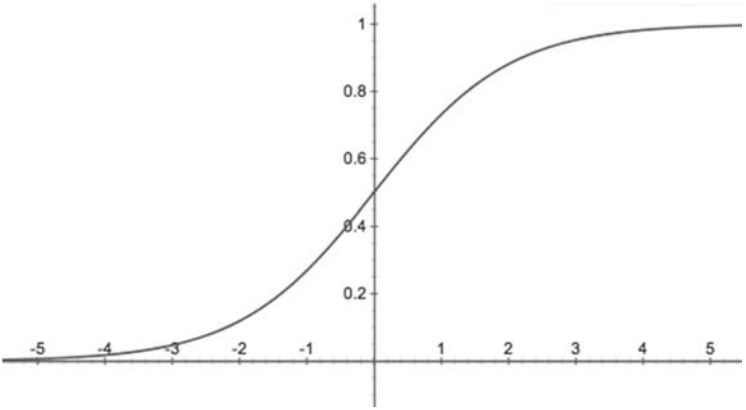

Рис. 2. График сигмоидальной нелинейной функции $f(x) = 1 / (1 + e^{-x})$

Рассмотрим процесс синтеза элементарных функций с использованием нейронный сетей на *примере синтеза логической функции «или»*. Первым шагом необходимо составить таблицу значений входов и выходов, расширенную дополнительной колонкой с суммой значений x_1, x_2, для рассматриваемой функции (таблица 1).

Таблица 1. Значения входов и выходов для функции «или»

x1	x2	y	Сумма входных значений
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	2

Основной идеей задачи реализации функции на основе нейронов является подбор такой нелинейности, которая подходит под заданный критерий точности. Например, при заявленной ошибке в 10% (далее будет использоваться абсолютное значение 0.1) для соответствия требуемому выходу, функция должна проходить через точки $x = 0.2,\ y = 0.1;$ $x = 0.8,\ y = 0.9.$ Эти значения обуславливаются тем, что в худшем случае значение «нуля» будет равно 0.2, а «единицы» — 0.8. Функция, проходящая через эти точки, представлена на рисунке 3.

Далее необходимо убедиться, что выбранная функция действительно удовлетворяет заданным критериям. Значения функции «или» представлены в таблице 2. В таблице приведена только часть значений входных сигналов с погрешностями, потому что остальные заведомо укладываются в значение заявленной ошибки.

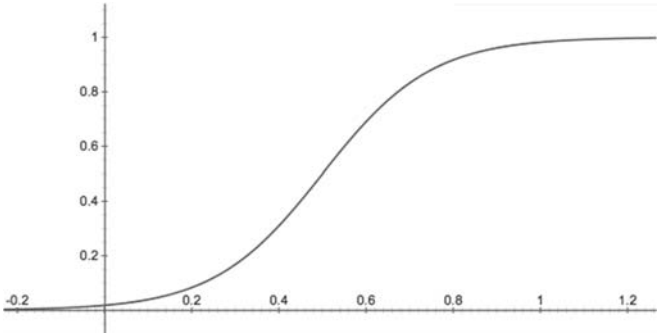

Рис. 3. График сигмоидальной нелинейной функции $f(x) = 1/(1+e^{-8*x+4})$

Таблица 2. Значения функции «или» при различных значениях входных сигналов

x1	x2	Сумма входных значений	y
0	0	0	0.01
0	1	1	0.98
1	0	1	0.98
1	1	2	0.99
0.1	0.1	0.2	0.08
-0.1	0.9	0.8	0.92
0.9	-0.1	0.8	0.92
0.9	0.9	1.8	0.99

Нейронная схема полученной функции приведена на рисунке 4.

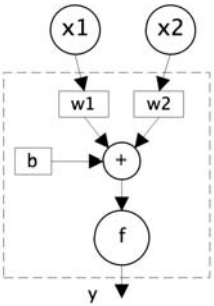

Рис. 4. Операция логического «или», реализованная при помощи одного нейрона

В полученной функции значение смещения будет b равно нулю, значение w_1 и w_2 равняются единицам.

Реализация функции «и» выполняется схожим образом. Изначально необходимо составить таблицу значений входов и выходов, расширенную дополнительной колонкой с суммой значений x_1, x_2 для функции «и» (таблица 3).

Таблица 3. Значения входов и выходов для функции «и»

x1	x2	y	Сумма входных значений
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	2

Следующим шагом является получение точек, через которые должна проходить функция. Такими точками будут: $x = 1.2,\ y = 0.1;\ x = 1.8,\ y = 0.9$. Функция, проходящая через эти точки представлена на рисунке 5.

Результаты значений функции «и» представлены в таблице 4.

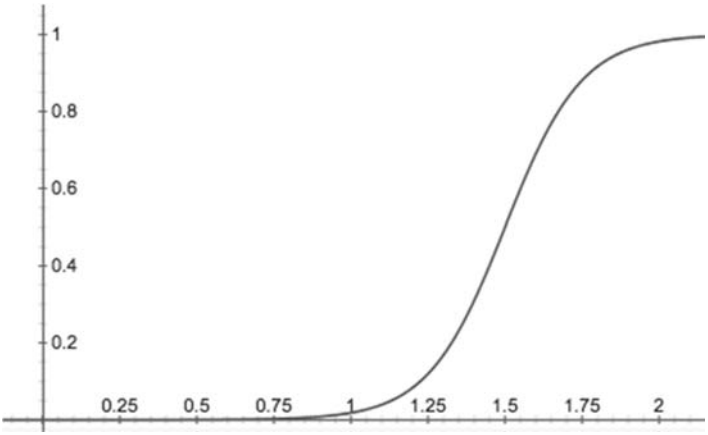

Рис. 5. График сигмоидальной нелинейной функции $f(x) = 1/(1+e^{-8*x+12})$

Таблица 4. Значения функции «и» при различных значениях входных сигналов

x1	x2	Сумма входных значений	y
0	0	0	0.01
0	1	1	0.02
1	0	1	0.02
1	1	2	0.98
0.1	0.1	0.2	0.01
0.1	1.1	1.2	0.08
1.1	0.1	1.2	0.08
0.9	0.9	1.8	0.91

Нейронная схема полученной функции также, как и структуры нейронных сетей для *функций «и-не», «или-не» и функций сравнения,* совпадает со схемой на рисунке 4. Все вышеприведенные функции могут быть реализованы при помощи нейронной сети, состоящей из одного нейрона. Это обусловливается тем, что разделение выходного множества результатов нейронной сети может быть достигнуто при помощи прямой.

Рассмотрим реализацию *функции XOR*. Согласно порядку синтеза вышеприведенных функций, необходимо составить таблицу значений входов и выходов, расширенную дополнительной колонкой с суммой значений x_1, x_2 (таблица 5).

Таблица 5. Значения входов и выходов для функции «исключающее или»

x1	x2	y	Сумма входных значений
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	2

Исходя из таблицы 5 получим точки, через которые должна проходить функция нелинейности: $x = 0.2$, $y = 0.1$; $x = 0.8$, $y = 0.9$; $x = 1.2$, $y = 0.9$; $x = 1.8$, $y = 0.1$. Очевидно, что сигмоидальная функция не может пройти через все указанные точки. Тогда можно либо отказаться от сигмоидальной функции, либо составить результирующую функцию из двух сигмоидальных функций, то есть использовать схему из нескольких нейронов. Стоит отметить, что результирующая функция может быть представлена как сумма двух функций: 1) перевернутой функции на рисунке 3 ($f(x) = -1/(1+e^{-8*x+12})$); 2) функции на рисунке 3. Тогда итоговое уравнение полученной функции будет иметь вид: $f(x) = -1/(1+e^{-8*x+12}) + 1/(1+e^{-8*x+4})$. График ее представлен на рисунке 6.

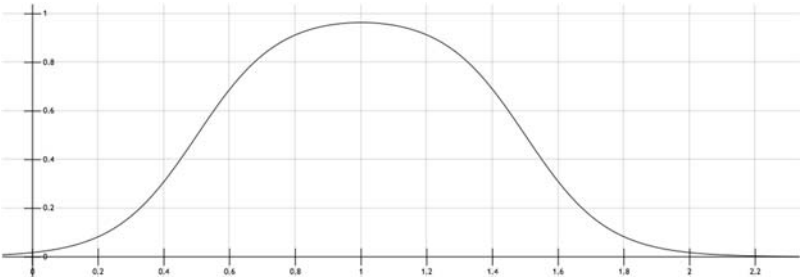

Рис. 6. График сигмоидальной нелинейной функции

$$f(x) = -1/(1+e^{-8*x+12}) + 1/(1+e^{-8*x+4})$$

Реализация вышеприведенной функции будет выполняться путем разделения данной функции на слагаемые, где каждая из частей будет включена в состав нейрона. После исполнения каждого из нейронов их результат будет суммироваться на третьем нейроне, где в качестве функции нелинейности будет константное значение единицы. Итоговая схема приведена на рисунке 7.

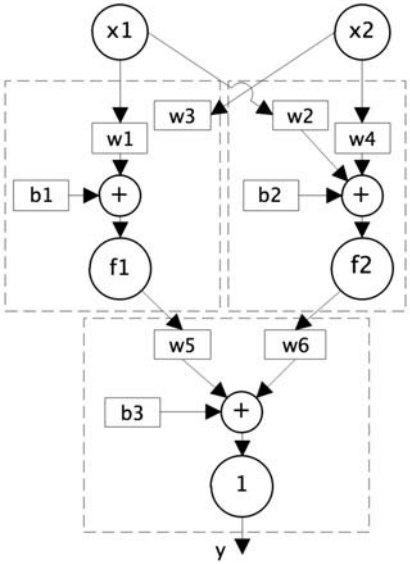

Рис. 7. Операция «исключающего или» (XOR),реализованная при помощи нейронной сети

Получившееся решение для функции XOR совпадает по структуре с решениями, приведенными в [4, 5], но различается по выбранной нелинейности и значениям весов.

Построение других функций выполняется аналогичным образом. Стоит также отметить, что изменение количества входов для разных функций приводит к разным результатам. Например, функции «или» с любым количеством входов может быть реализована с помощью одного нейрона. А для реализации функции XOR с тремя входами требуется большее количество нейронов. В вышеприведенных таблицах помимо колонки с суммой может быть выбрана другая функция, например, разность для реализации функции сравнения «>».

3. Пример. Рассмотрим решение логико-арифметических задач на примере реализации функции выбора максимального числа из трех чисел, представленных в бинарном виде четырьмя разрядами.

Идея, на которой базируется структура нейронной сети (рисунок 8) схожа с идей, представленной в [15-17], и заключается в том, что числа сравниваются попарно, начиная со старших разрядов и далее к младшим. При сравнении разрядов возможны несколько вариантов: 1) значения разрядов равны; 2) значение разрядов не равны; 3) значения разрядов не сравниваются, если при сравнении более старших разрядов было выявлено, что они не равны. При реализации сравнения может возникнуть ситуация, когда по старшим разрядам можно закончить сравнение, но так как в нейронных сетях, в отличие от сетей Петри, отсутствует возможность задать порядок управления, то необходимо продолжать сравнение с учетом предыдущих результатов.

На рисунке 8 представлена схема сравнения старших разрядов $(x_1,\ x_2)$ двух чисел. Если значения старших разрядов x_1 и x_2 являются единицами, то на выходе *and* также будет единица (на рисунках 8, 9, 10, 11 прямоугольниками представлены функции, составленные при помощи нейронов), если оба значения равняются нулю, то на выходе функции *and_not* будет единица. Если значения разрядов не равны, то на выходе двух вышеприведенных функций будут нули. Выявление какой из сравниваемых разрядов больший выполняется функциями *cmp1*, *cmp2 (compare)*, которые на выходе выдают единицу, если $x_1 = 1,\ x_2 = 0$ и $x_1 = 0,\ x_2 = 1$ соответственно. В итоге в результате работы нейронной сети на рисунке 8 может быть единица либо на выходе eq_0, либо на $x_1 > x_{2_0}$, либо на $x_1 < x_{2_0}$.

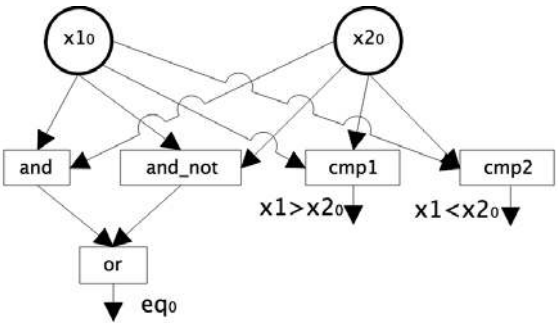

Рис. 8. Сравнение старших разрядов двух чисел

На рисунке 9 приведена схема сравнения последующих разрядов. В отличие от предыдущей схемы (рисунок 8), на этой схеме пунктирными линиями представлены связи нейронов с результатами сравнения предыдущих слоев.

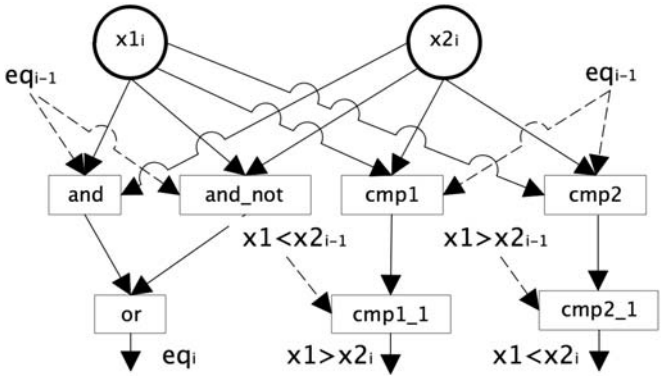

Рис. 9. Сравнение *i*-тых разрядов двух чисел

Функция *and* выдает единицу на выходе при двух единицах в x_{1_i} и x_{2_i}, а также если в результате прошлого сравнения *i-1*-ые разряды были равны. Если eq_{i-1} – ноль (т.е. значения чисел не равны), то выходом функции *and* будет ноль, то есть числа не равны. Аналогично и для функции *and_not*. Функции *cmp1* и *cmp2* работают также, как и в предыдущем слое при условии, что в предыдущем сравнении разряды были равны (значение eq_{i-1} — единица). Выходное значение $x1 > x2_i$ (функция *cmp1_1*) формируется с учетом значения выхода $x_1 < x_{2_{i-1}}$: если его значением является единица, то на выходе $x_1 > x_{2_i}$ будет ноль, так как сравнение более старших разрядов приоритетно, если $x_1 < x_{2_{i-1}}$ является нулем, то выход функции *cmp1_1* определяется значением выхода функции *cmp1*. Аналогично работает и функция *cmp2_1*.

Итоговая схема сравнения всех разрядов двух чисел представлена на рисунке 10. Данная схема может быть легко расширена на произвольное количество разрядов путем добавления повторяющихся участков.

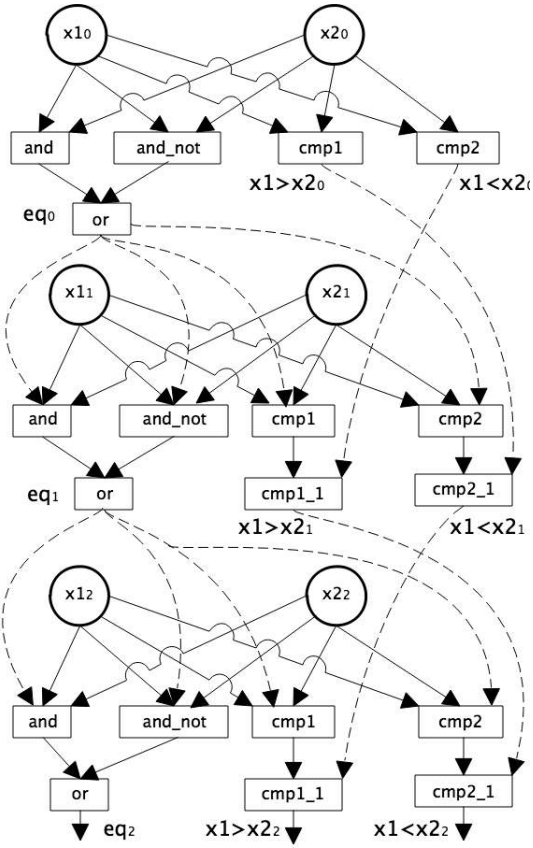

Рис. 10. Сравнение разрядов двух трехразрядных чисел

На рисунке 11 представлена часть нейронной сети, в которой выполняется формирование выходных разрядов двух сравниваемых чисел. Изначально формируются признаки сравнения двух чисел: *eq* — будет равной единице, если результат сравнения младших разрядов eq_2 будет равен единице (в противном случае — ноль). Выходы $x_1 > x_2$, $x_1 < x_2$ формируются в результате операций «или» для всех $x_1 > x_{2_i}$ и $x_1 < x_{2_i}$ соответственно. Только один из функций eq, $x_1 > x_2$, $x_1 < x_2$ может содержать значение единицы. Далее значения разрядов и признаков, что соответствующее число больше, поступают на функции «и» (отмечены как *and*). Для первого числа таким признаком является или выход eq,

или $x_1 > x_2$, для второго только $x_1 < x_2$. После исполнения функций «и» значение разряда меньшего числа будет содержать ноль, а значение разряда старшего числа само значение разряда. Далее из выходов функций «и» формируется итоговые значения разрядов максимального числа сравниваемых чисел.

Пример получения значений разрядов может быть расширен для произвольного количества разрядов путем добавления повторяющихся участков схемы выбора соответствующего разряда.

Результатом работы нейронной сети на рисунке 11 является максимальное число из двух сравниваемых чисел. Таким образом, для выбора максимального числа из массива чисел необходимо продолжить сравнение текущего максимального числа (т.е. результата нейронной сети на рисунке 11) с оставшимися числами массива. В итоге максимальное число будет выбрано в результате сравнения последнего числа и текущего максимального значения.

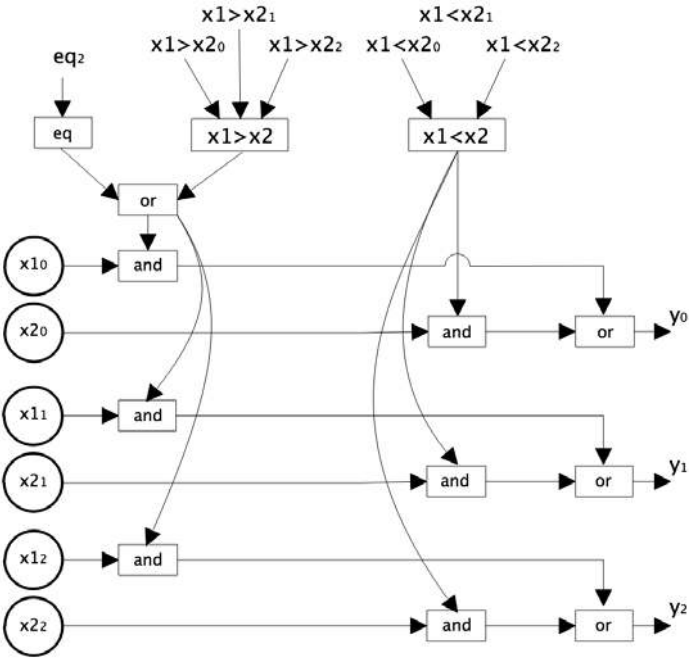

Рис. 11. Формирование разрядов числа

Общая итоговая схема представлена на рисунке 12. Каждая из секций сравнения, представленная на рисунке прямоугольником

comparison, включает в себя сравнение двух поступающих на вход схемы чисел, представленных в бинарном виде тремя разрядами. Результатом работы каждой секции сравнения будет максимальное из двух число, также представленное в бинарном виде (для чисел i_1 и i_2 максимальным будет $max_{1\text{-}2}$). Затем максимальное выбранное число сравнивается со следующим и далее происходит до тех пор, пока не будут сравнены все числа. Итоговое выбранное максимальное число на рисунке 12 обозначено как *max*.

Также стоит отметить, что приведенная схема похожа по своей работе на однопоточный выбор максимального числа в массиве чисел, который можно реализовать при помощи любого языка программирования. Но потенциально работу представленной схемы можно ускорить за счет распараллеливания выполняемых операций. При этом синхронизация работы параллельно выполняющихся операций будет выполняться на любом нейроне, который использует результаты вычислений предыдущих слоев.

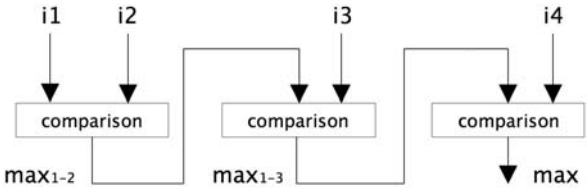

Рис. 12. Формирование *i*-ого разряда числа

4. Проанализируем рассмотренный в предыдущем разделе пример. Получившаяся система содержит 314 обучаемых параметров в отличие от нейронной сети, например, прямого распространения с четырьмя скрытыми слоями по 12 нейронов в каждом, которая содержит 676 обучаемых параметров. Нейроны обучаемой сети также используют сигмоидальную функцию активации. Обучение выполняется при помощи библиотек для работы с нейронными сетями keras v2.0.2 с использованием tensorflow v1.0.0 в качестве вычислительной среды. Начальные условия задаются равномерным распределением из диапазона {-1, 1}, непосредственно само обучение выполняется за 500 эпох с использованием метода среднеквадратического распространения ошибки (*RMSprop*) с шагами обучения 0.1, 0.01 и бинарной перекрестной энтропией (*binary crossentropy*) в качестве стоимостной функции.

Кроме того, рассматривался пример выбора максимального числа из пяти чисел, представленных в бинарном виде восьмью разрядами (вместо данного примера в работе приведен более компактный, но схожий по логике работы). Данная система была синтезирована аналогичным образом и содержит 1300 обучаемых параметров. В нейронной сети прямого распространения из четырех скрытых слоев по 40 нейронов содержится 6888 обучаемых параметров (остальные параметры обучения такие же, как и у сети выше).

Стоит отметить, что предлагаемый в статье способ построения нейронной сети не потребовал обучения, что связано с простотой схемы и незначительным числом входов каждого из элементов (таблица 4). Достоинством предлагаемого подхода является то, что начальные значения для обучаемых параметров, по крайней мере для простых схем, можно рассчитать, а для более сложных систем — они будут близки к оптимальным.

Отдельно стоит сравнить скорость обучения нейронной сети прямого распространения с 12 нейронами в скрытых слоях и нейронной сети, представленной на рисунках 8-12. Графики стоимостных функций (*loss function*) приведены на рисунках 13-16.

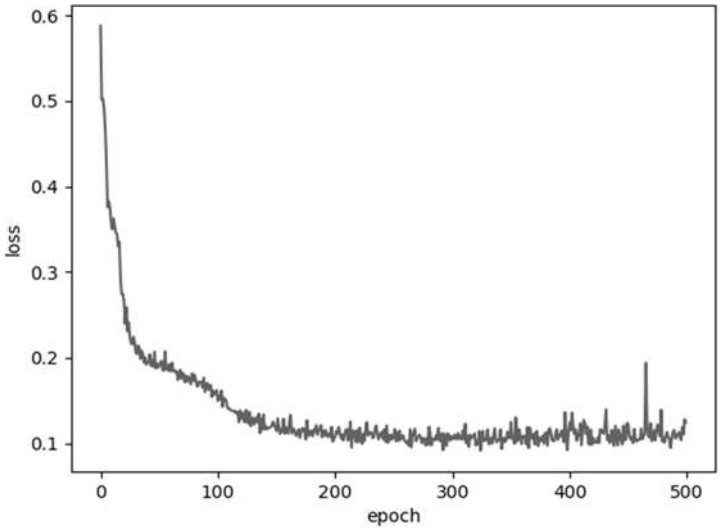

Рис. 13. График стоимостной функции нейронной сети прямого распространения с параметром обучения равным 0.1

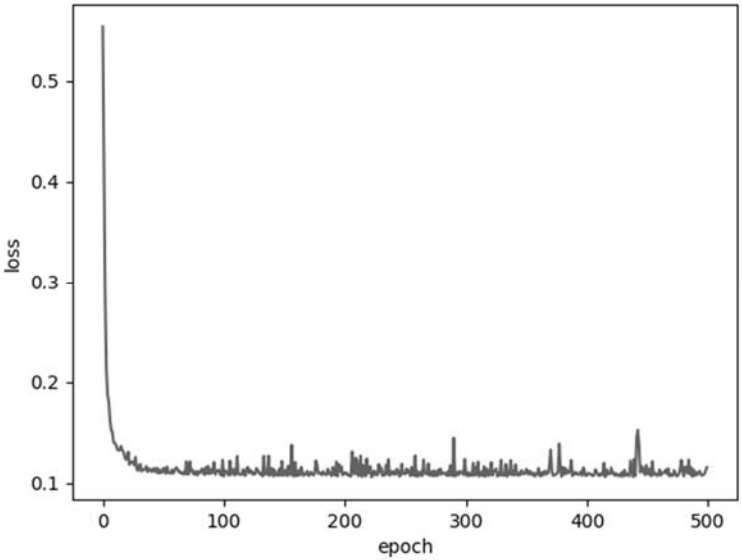

Рис. 14. График стоимостной функции нейронной сети (рисунки 8-12) с параметром обучения равным 0.1

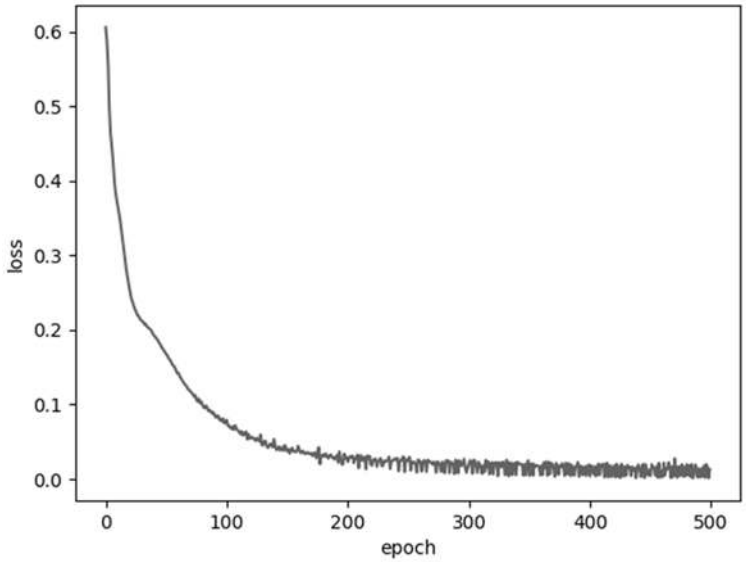

Рис. 15. График стоимостной функции нейронной сети прямого распространения с параметром обучения равным 0.01

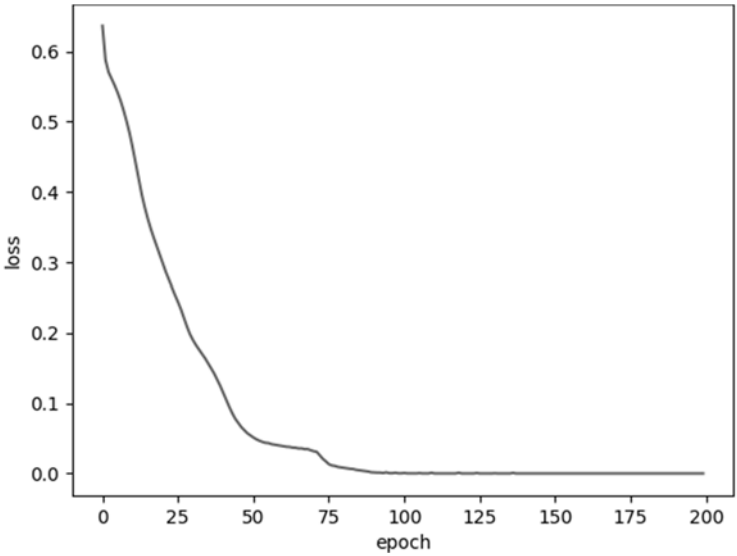

Рис. 16. График стоимостной функции нейронной сети (рисунки 8-12) с параметром обучения равным 0.01

Из приведенных графиках (рисунки 13-16) видно, что обучение выполняется быстрее в синтезированной нейронной сети, что может быть объяснено меньшим количеством параметров обучения.

На рисунке 14 стоимостная функция сходится быстрее и достигает асимптотического значения, но само значение больше, чем у сравниваемой функции на рисунке 13. Это может быть объяснено большим значением параметра обучения (0.1), так как при уменьшенном до 0.01 значение стоимостной функции практически достигает нуля за первые 100 эпох в отличии от рисунка 15, где значение нуля достигается после 400 эпох.

5. Заключение. В статье предложен способ синтеза нейронных сетей для логико-арифметических задач, который заключается в предварительном построении элементарных функций таких как: «и», «или», «не», «или-не», «≥», «≤», «>», «<» и других. На основе вышеприведенных функций может конструироваться нейронная сеть, что иллюстрируется на примере построения функции выбора максимального числа из трех чисел, представленных в бинарном виде четырьмя разрядами. Синтез нейронной сети вышеприведенным способом выполняется с дальнейшей целью получения обобщенной структуры нейронной сети. Кроме того, полученная нейронная сеть не потребовала обучения, так как она основывается на предварительно построенных функциях, которые уже содержат нужные значения коэффициентов. А для более сложных систем, начальные значения будут близки к оптимальным, что также положительно скажется на скорости обучения.

С другой стороны, если рассматривать только структуру нейронной сети (без значений весов), то ее обучение выполняется значительно быстрее, чем у сравниваемой нейронной сети прямого распространения.

Литература

1. *Krizhevsky A., Sutskever I., Hinton G.E.* ImageNet Classification with Deep Convolutional Neural Networks // Proceedings of Neural Information Processing Systems (NIPS). 2012. pp. 1090–1098.
2. *Graves A., Mohamed A., Hinton G.* Speech recognition with deep recurrent neural networks // Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2013. pp. 6645–6649.
3. *Deng L., Hinton G.E., Kingsbury B.* New types of deep neural network learning for speech recognition and related applications: An overview // Proceedings of IEEE International Conference on Acoustic Speech and Signal (ICASSP 2013). 2013. pp. 8599–8603.
4. *Haykin S.* Neural networks and learning machines: 3rd edition // Pearson Education. 2009. 938 p.
5. *Bishop C.* Pattern Recognition and Machine Learning // Springer. 2006. 738 p.
6. *Goodfellow I., Bengio Y., Courville A.* Deep Learning // MIT Press. 2016. 781 p.
7. *Haykin S., Deng C.* Classification of radar clutter using neural networks // IEEE Transactions on Neural Networks. 1991. vol. 2. pp. 589–600.
8. *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning: Data Mining, Inference, and Prediction // Springer. 2001. 552 p.
9. *Hagan M., Demuth H., Jesús O.* A neural network predictive control system for paper mill wastewater treatment // Engineering Applications of Artificial Intelligence. 2003. vol. 16. no. 2. pp. 121–129.
10. *Touretzky D.S., Pomerleau D.A.* What is hidden in the hidden layers? // Byte. 1989. vol. 14(8). pp. 227–233.
11. *LeCun Y., Bengio Y., Hinton G.* Deep learning // Nature. 2015. vol. 521. no. 7553. pp. 436–444.
12. *Srivastava N. et al.* Dropout: A Simple Way to Prevent Neural Networks from Overfitting // Journal of Machine Learning Research. 2014. vol. 15(1). pp. 1929–1958.
13. *Mahsereci M., Balles L., Lassner C., Hennig P.* Early Stopping without a Validation Set // ArXiv e-prints. no.1703.09580. 2017.
14. *Prechelt L.* Early Stopping — But When? // Neural Networks: Tricks of the Trade. 2012. pp. 53–67.
15. *Воевода А.А., Романников Д.О.* Асинхронный алгоритм сортировки массива чисел с использованием ингибиторных сетей Петри // Труды СПИИРАН. 2016. Вып. 48. С. 198–213.
16. *Воевода А.А., Полубинский В.Л., Романников Д.О.* Сортировка массива целых чисел с использованием нейронной сети // Научный Вестник НГТУ. 2016. № 2(63). С. 151–157.
17. *Voevoda A.A., Romannikov D.O.* A Binary Array Asynchronous Sorting Algorithm with Using Petri Nets // Journal of Physics: Conference Series. 2017. vol. 803. no. 1. pp. 012178.

Воевода Александр Александрович — д-р техн. наук, профессор, профессор кафедры автоматики, Новосибирский государственный технический университет (НГТУ). Область научных интересов: полиномиальный синтез, сети Петри, UML диаграммы. Число научных публикаций — 200. voevoda@ucit.ru; пр. Карла Маркса 20, Новосибирск, 630073; р.т.: +79139223092.

Романников Дмитрий Олегович — к-т техн. наук, доцент кафедры автоматики, Новосибирский государственный технический университет (НГТУ). Область научных интересов: машинное обучение, нейронные сети, сети Петри. Число научных публикаций — 51. dmitry.romannikov@gmail.com; пр. Карла Маркса 20, Новосибирск, 630073; р.т.: +7 961 223 8567.

DOI 10.15622/sp.54.9

A.A. VOEVODA, D.O. ROMANNIKOV

SYNTHESIS OF NEURAL NETWORK FOR SOLVING LOGIC-ARITHMETIC PROBLEMS

Voevoda A.A., Romannikov D.O. **Synthesis of Neural Network for Solving Logical-Arithmetic Problems.**

Abstract. One of the main problems facing the developer of a system with a neural network is the choice of the structure of a neural network that could solve the tasks. At present there are no unambiguous recommendations for choosing such a structure and such parameters as: the number of layers, the number of neurons in the layer, the type of neuron nonlinearity, the training method, the parameters of the training method, and others.

The article considers an approach to the synthesis of a neural network for a class of logical-arithmetic problems, based on the formation of a network of pre-constructed elementary functions. The novelty of the proposed approach is the formation of a neural network using a well-known algorithm using pre-built functions. Thus, in the article elementary logical-arithmetic functions such as "and", "or", "exclusive or", "and-not", "or-not", "≥", "≤", ">", "<" are built, which can be used to solve more complex problems. An example of a solution to the problem of constructing a function for selecting the maximum number of four numbers represented in binary form in three digits is given. Synthesis of a neural network in the manner described above is performed with the further goal of obtaining a generalized structure of the neural network.

Keywords: neural networks, machine learning, logical-arithmetic problems, synthesis of a neural network.

Voevoda Alexandr Aleksandrovich — Ph.D., Dr. Sci., professor, professor of automation department, Novosibirsk State Technical University (NSTU). Research interests: polynomial synthesis, UML diagrams, Petri nets. The number of publications — 200. voevoda@ucit.ru; 20, Karl Marx Avenue, Novosibirsk, 630073; office phone: +79139223092.

Romannikov Dmitry Olegovich — Ph.D., associate professor of automation department, Novosibirsk State Technical University (NSTU). Research interests: machine learning, neural networks, Petri nets.. The number of publications — 51. dmitry.romannikov@gmail.com; 20, Karl Marx Avenue, Novosibirsk, 630073; office phone: +7 961 223 8567.

References

1. Krizhevsky A., Sutskever I., Hinton G.E. ImageNet Classification with Deep Convolutional Neural Networks. Proceedings of Neural Information Processing Systems (NIPS). 2012. pp. 1090–1098.
2. Graves A., Mohamed A., Hinton G. Speech recognition with deep recurrent neural networks. Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2013. pp. 6645–6649.
3. Deng L., Hinton G.E., Kingsbury B. New types of deep neural network learning for speech recognition and related applications: An overview. Proceedings of IEEE International Conference on Acoustic Speech and Signal (ICASSP 2013). 2013. pp. 8599–8603.
4. Haykin S. Neural networks and learning machines: 3rd edition. Pearson Education. 2009. 938 p.
5. Bishop C. Pattern Recognition and Machine Learning. Springer. 2006. 738 p.
6. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press. 2016. 781 p.

7. Haykin S., Deng C. Classification of radar clutter using neural networks. *IEEE Transactions on Neural Networks*. 1991. vol. 2. pp. 589–600.
8. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer. 2001. 552 p.
9. Hagan M., Demuth H., Jesús O. A neural network predictive control system for paper mill wastewater treatment. *Engineering Applications of Artificial Intelligence*. 2003. vol. 16. no. 2. pp. 121–129.
10. Touretzky D.S., Pomerleau D.A. What is hidden in the hidden layers? *Byte.* 1989. vol. 14(8). pp. 227–233.
11. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. vol. 521. no. 7553. pp. 436–444.
12. Srivastava N. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 2014. vol. 15(1). pp. 1929–1958.
13. Mahsereci M., Balles L., Lassner C., Hennig P. Early Stopping without a Validation Set. ArXiv e-prints. no. 1703.09580. 2017.
14. Prechelt L. Early Stopping — But When? Neural Networks: Tricks of the Trade. 2012. pp. 53–67.
15. Voevoda A.A., Romannikov D.O. [Asynchronous Sorting Algorithm for Array of Numbers With the use of Inhibitory Petri Nets]. *Trudy SPIIRAN — SPIIRAS Proceedings.* 2014. vol. 3(34). pp. 218–232. (In Russ.).
16. Voevoda A.A., Polubinskij V. L., Romannikov D.O. [Array of integers sorting with a using of a neural network]. *Nauchnyj Vestnik NGTU — Science Bulletin of NSTU.* 2016. vol. 2(63). pp. 151–157. (In Russ.).
17. Voevoda A.A., Romannikov D.O. A Binary Array Asynchronous Sorting Algorithm with Using Petri Nets. *Journal of Physics: Conference Series*. 2017. vol. 803. no. 1. pp. 012178.