УДК 004.032.26

SIGMA-PI-НЕЙРОН И РЕШЕНИЕ ПРОБЛЕМЫ «ПЕРЕМНОЖЕНИЯ ФАКТОРОВ» В ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЯХ

Р. Ю. Порцев*, А. В. Макаренко**

*,**Институт проблем управления им. В. А. Трапезникова РАН, г. Москва

*✉ poruss@mail.ru, **✉ avm.science@mail.ru

Аннотация. Предложена конструкция Sigma-Pi-нейрона, позволяющая на этапе обучения нейронной сети управляемо и автоматически активировать операцию перемножения факторов. Решение прозрачно интегрируется в текущие и перспективные нейросети и способно осуществлять прямое нейросетевое вычисление множества математических отношений, традиционно реализуемых нейросетями в режиме мемоизации (запоминания на этапе обучения). Прямое выполнение операции умножения формирует меньшие сетевые ошибки. В полную силу этот фактор раскрывается при работе с входными данными вне домена обучения. Доказана реализация единственным Sigma-Pi-нейроном логической булевой функции «XOR» произвольной размерности. Работоспособность и вычислительные возможности нового конструкта, его превосходство над классическими полносвязными нейросетями прямого распространения продемонстрированы на задачах нейросетевого моделирования многочленов и операций векторно-матричной алгебры. Представляется, что предложенная конструкция Sigma-Pi-нейрона найдёт своё применение как в суррогатных расчётно-моделирующих нейросетевых моделях, так и в задачах класса AI+Science в целом.

Ключевые слова: Sigma-Pi-нейрон, операция перемножения факторов, функция XOR, суррогатные модели.

ВВЕДЕНИЕ

В 1943 г. У. Мак-Каллок и У. Питтс опубликовали работу [1] в которой предложили формальную модель искусственного нейрона. В 1958 г. Ф. Розенблатт обобщил эту модель [2]. В современной нотации она имеет вид

$$s = \mathbf{w}\cdot\mathbf{x} + b,\ z = \mathrm{g}(s), \qquad (1)$$

и известна как нейрон Мак-Каллока – Питтса. В формуле (1) обозначения имеют следующий смысл: $\mathbf{x}$ – вектор входных данных размерностью N, $\mathbf{x}\in\mathbb{R}^N$; $\mathbf{w}$ – вектор обучаемых весов; b – обучаемое смещение; $\cdot$ – операция скалярного умножения; $\mathrm{g}(\circ)$ – нелинейная функция активации; z – выход. Согласно парадигме Ф. Розенблатта и современным представлениям глубокого обучения [3], нейроны вида (1) и/или их производные объединяются в регулярные слои, а объединение слоёв, в свою очередь, формирует нейронную сеть.

Анализ выражения (1) показывает, что основные вычисления и адаптация в сети с архитектурой Ф. Розенблатта сосредоточены в обучаемом взвешивающем сумматоре $\mathbf{w}\cdot\mathbf{x}$. Функция активации $\mathrm{g}(\circ)$ имеет вторичный смысл – в случае многослойной организации она защищает композицию слоёв от вырождения до тривиального векторно-матричного умножения (фактически до линейной регрессии). Следует отметить, что подавляющее большинство современных глубоких искусственных нейронных сетей (включая полносвязные, свёрточные, рекуррентные, трансформерные архитектуры и их комбинации) [3–5] функционируют на основе данной парадигмы. В относительно недавно предложенной архитектуре KAN [6] сумматор в формуле (1) является необучаемым (с фиксированными единичными весами, $\mathbf{w}\equiv\mathbf{1}$), а функции

активации $g(\circ)$ представляют собой обучаемые полиномы ограниченной степени.

Изначально функция активации $g(\circ)$ определялась по типу пороговой функции Хевисайда (нейроны оперировали бинарными сигналами):

$$z = g_{01}(s) = \begin{cases} 1, & \text{если } s > a, \\ 0, & \text{если } s \le a, \end{cases}$$

где a – порог активации, по умолчанию $a = 0$. Затем Ф. Розенблатт [2] ввёл в рассмотрение знаковую функцию

$$z = g_{11}(s) = \operatorname{sign} s,$$

а Д. Е. Румельхарт в 1986 г. [7] предложил использовать либо сигмоидальную функцию, либо гиперболический тангенс (вход сети становится непрерывнозначным):

$$z = g_{sg}(s) = \frac{1}{1+e^{-s}}, \qquad (2)$$

$$z = g_{th}(s) = \operatorname{th} s.$$

В 1989 г. Г. Цыбенко публикует [8] доказательство теоремы об универсальной аппроксимации:

Теорема 1 . *Искусственная нейронная сеть прямого распространения с одним скрытым слоем может аппроксимировать любую непрерывную функцию многих переменных с любой точностью при условии, что сеть имеет в скрытом слое достаточное количество нейронов N, имеющих сигмоидальную функцию активации* g_{sg}.

Отметим, что теорема 1 существенно дополнила теорему о сходимости перцептрона [9]. В этом ключе стоит также отметить близкую к ней теорему Хехт – Нильсена [10]. В 1991 г. К. Хорник обобщает теорему 1 на случай произвольных нелинейных активационных функций [11], см. также работу [12]. Становится ясно, что универсальные аппроксимационные свойства искусственных нейронных сетей (ИНС) – это в большей мере свойство именно сетевой структуры. В 1992 г. М. Лешно с соавторами [13] дополнительно ослабили требования на вид допустимых функций активации, что фактически сформировало теоретический базис под использование таких функций, как ReLU, Swish, GELU и т. п. Наконец, в 1999 г. В. Майоров и А. Пинкус [14] доказывают аналог теоремы 1 для случая нейросети ограниченной ёмкости: два скрытых слоя и конечное число нейронов в обоих слоях, при условии использования сигмоидальной функции активации g_{sg}.

Теорема 1 является в определённом смысле специализированным аналогом теоремы А. Н. Колмогорова и В. И. Арнольда от 1957 г. о представимости непрерывных функций нескольких переменных композицией непрерывных функций одной переменной [15–17]:

Теорема 2 [Теорема Колмогорова – Арнольда]. *Каждая непрерывная функция нескольких переменных может быть представлена в виде суперпозиции непрерывных функций одной переменной и бинарной операции сложения.*

Для функции f от N переменных представление по теореме 2 может иметь следующий вид (обобщение формулировки Досса):

$$f(\mathbf{x}) = \sum_{i=1}^{2N+1} \Phi_i \left[\sum_{j=1}^{N} \phi_{i,j}\left(x_j\right) \right]. \qquad (3)$$

В свою очередь, теорема 2 связана с аппроксимационной теоремой К. Вейерштрасса от 1885 г. [18]:

Теорема 3. *Предположим, что* f *– непрерывная функция, принимающая вещественные значения и определённая на замкнутом вещественном интервале* $[a, b]$. *Для каждого* $\varepsilon > 0$ *существует многочлен* p *такой, что для всех* $x \in [a, b]$ *имеем* $|f(x) - p(x)| < \varepsilon$.

В итоге складывается парадоксальная ситуация.

Условия теоремы 2 и формула (3) определяют, что сложение является единственной истинной многомерной операцией, в то время как другие многомерные операции (включая умножение) могут быть выражены как операции сложения в сочетании с одномерными функциями[1]. И мейнстрим нейросетевых подходов оперирует только «суммированием факторов», а все остальные операции, в том числе «перемножение факторов», реализуются благодаря режиму «мемоизации[2]».

Но в практике как математического моделирования, так и вычислений вообще результаты теоремы Колмогорова – Арнольда о представлении (3) носят весьма ограниченный характер. Можно даже утверждать, что они не реализуются при решении реальных задач: аналитическая запись формул и вычислительные реализации изоби-

[1] Следует отметить, что теорема Колмогорова – Арнольда гарантирует представление только для непрерывных функций, но ничего не утверждает по дифференцируемым функциям (см. также работу [14] и результаты А. Г. Витушкина [19]).

[2] На этапе обучения нейросети внутри неё формируется статическая память, фактически генерируется табличное сопоставление аргумента и значения запомненной функции.

луют операцией умножения. Почему так? Ответ кроется в компактности и точности формул и вычислений, в которых операция умножения реализуется в явном виде[3].

Таким образом, «честное» и прямое перемножение факторов – весьма важная и востребованная операция в нейросетях. Приведём простейшие примеры:

• Входными данными заданы длина и ширина прямоугольной излучающей площадки, а также нормируется равномерная поверхностная плотность потока излучения. Необходимо определить общий поток излучения. Естественно, «внутри» модели возникает необходимость вычислить площадь площадки.

• Входными данными заданы количество товара и его цена, необходимо вычислять общую стоимость партии товара. Естественно, модель внутри себя должна уметь перемножать количество товара на его цену.

Следует также отметить, что в настоящее время нейросетевые подходы активно применяются при построении расчётно-моделирующих суррогатных моделей[4] для оценивания / прогнозирования свойств и/или поведения сложных нелинейных (в том числе нестационарных) систем физической/технической природы. В данном классе задач модели зачастую функционируют вне домена обучения (причём существенно отходят от его границ), что приводит к существенной деградации их качества и устойчивости из-за ограниченной обобщающей способности механизма мемоизации.

Активная операция перемножения факторов в искусственных нейронных сетях потенциально может быть реализована следующими основными способами:

• **Определение функции активации в форме возведения аргумента в квадрат**, через выражение

$$(u+v)^2 = u^2 + 2uv + v^2, \qquad (4)$$

позволяет двухслойной сети выделить произведение входных факторов. Наглядно данное утверждение приведено на рис. 1. Минусы данного подхода очевидны: экстенсивный расход нейронов и слоёв, невозможность адаптивно «выращивать» блоки перемножения в необходимых местах и количествах; плохая обобщаемость подхода на случай перемножения более двух входных переменных.

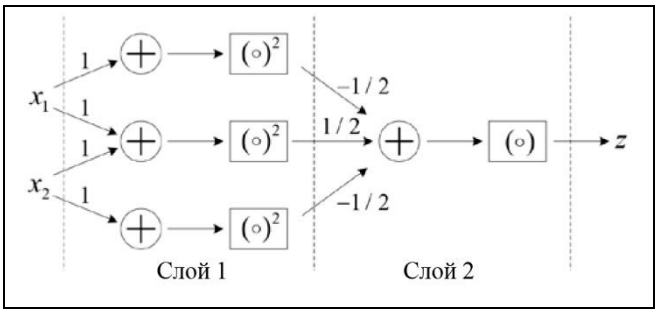

Рис. 1. Реализация умножения факторов нейронной сетью с квадратичной функцией активации

• **Функционирование нейронов в логарифмическом базисе** позволяет свести умножение к суммированию. Исходя из формулы

$$\log u + \log v = \log(uv),$$

подача на вход нейрона логарифмированных входных значений и определение функции активации в виде экспоненты позволяет выделить произведение входных факторов. У данного подхода есть дополнительный положительный момент – автоматическая реализация отношения факторов, а также смешанный режим: умножение и деление. Минусы данного подхода также очевидны: невозможность, в свою очередь, теперь уже адаптивно «выращивать» блоки суммирования в необходимых местах и количествах; плохая устойчивость при обучении сети в связи с экспоненциальной функцией активации, провоцирующей взрывной рост градиента.

• **Вентильный подход к построению функций активации**, определяемый как поэлементное произведение двух линейных преобразований входных данных, одно из которых активируется сигмоидной функцией [20] либо более современными вариантами [21]: ReLU, GeLU, Swish и т. п.; возможен также вариант с линейной функцией (билинейную форму исходно предложили А. Мних и Дж. Хинтон в работе [22] в 2007 г.). Формально данный подход может быть записан в виде:

$$\mathbf{z} = (\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \otimes \mathrm{a}(\mathbf{W}_2\mathbf{x} + \mathbf{b}_2), \qquad (5)$$

где $\otimes$ – произведение Адамара; $\mathrm{a}(\circ)$ – функция активации. Выражение (5) даёт нейросети возмож-

[3] Здесь уместна аналогия. Теорема 1 утверждает, что перцептрон с одним скрытым слоем способен решить любую задачу, но при условии экспоненциально большого числа нейронов в скрытом слое, что в действительности не реализуется. А глубокие нейросети в реальности позволяют решать сложные задачи достаточно компактными архитектурами (см. основную теорему в статье [14]). То есть операция умножения в этом контексте аналогична глубине сетевой структуры – обе делают задачи реализуемыми на практике.

[4] Это недорогие и быстрые модели, которые строятся на данных (по принципу чёрного ящика) и аппроксимируют поведение более сложных, точных и дорогих расчётных (имитационных) моделей.

ность моделировать квадратичные зависимости, что с учётом формулы (4) порождает возможность формировать произведения факторов.

Исходя из изложенного выше, следует отметить два важных фактора, существенно влияющих на характеристики нейросетевых моделей на текущий момент времени.

Во-первых, классические нейросетевые архитектуры KAN [6] реализуют механизм умножения факторов опосредованно, через экстенсивный механизм в форме (4). Расширение MultKAN [23] вводит умножение в явной форме. Однако данная операция вводится «принудительно», в виде дополнительного промежуточного слоя после слоя суммирования, что негативно сказывается на компактности сети и устойчивости её обучения. Кроме того, число перемножаемых факторов в узле сети задаётся пользователем (по умолчанию используются два сомножителя) и фиксируется перед обучением сети.

Во-вторых, современные трансформеры (и, как следствие, большие языковые и фундаментальные модели в частности) реализуют достаточно странный «способ умножения» в форме (5). Причём способ не только странный, но и весьма расточительный по ресурсам, а также неустойчивый при обучении (нейронная сеть не всегда находит при обучении требуемую для умножения комбинацию параметров).

В настоящей работе предложена альтернативная конструкция так называемого ΣΠ-нейрона, решающая проблему «перемножения факторов» в нейронных сетях «напрямую» и свободная от недостатков вышеприведённых подходов.

Работа имеет следующую структуру. В § 1 изложена конструкция ΣΠ-нейрона и приведены его основные свойства. Численным примерам, демонстрирующим работоспособность нейросетей на основе предложенного решения и его преимущества перед классическим суммирующим нейроном, посвящён § 2.

1. КОНСТРУКЦИЯ SIGMA-PI НЕЙРОНА И ЕГО ОСНОВНЫЕ СВОЙСТВА

1.1. Общие положения

Введём в рассмотрение так называемый ΣΠ-нейрон, который состоит из суммы

$$\Sigma\Pi(\mathbf{x}) = \left[1 - \mathrm{g}_{\mathrm{sg}}(\gamma)\right] s(\mathbf{x}) + \mathrm{g}_{\mathrm{sg}}(\gamma)\, p(\mathbf{x}), \qquad (6)$$

где γ – вентиль ΣΠ-нейрона, управляющий режимом его работы (суммирование/перемножение), $\mathrm{g}_{\mathrm{sg}}(\circ)$ – сигмоидальная функция (2). Слагаемое $s(\mathbf{x})$ ответственно за суммирование компонент входного вектора $\mathbf{x}$:

$$s(\mathbf{x}) = b + \sum_{i=1}^{N} w_i\, x_i, \qquad (7)$$

оно эквивалентно сумматору в выражении (1). Слагаемое $p(\mathbf{x})$ отвечает за перемножение входных данных и имеет вид:

$$p(\mathbf{x}) = m \prod_{i=1}^{N} \left[1 - \mathrm{g}_{\mathrm{sg}}(\alpha_i) + \mathrm{g}_{\mathrm{sg}}(\alpha_i)\, x_i\right], \qquad (8)$$

где m – масштабирующий коэффициент мультипликатора, предназначенный для окончательной перенормировки произведения; α_i – вентиль мультипликатора по i-му сомножителю. Отметим, что параметры γ, m, и α_i – обучаемые, также как w_i и b. Причём все обучаемые параметры принадлежат домену $\mathbb{R}$ и непрерывны.

Из формул (6)–(8) следует, что агрегирующий оператор в нейроне (сумма или произведение) выбирается автоматически в процессе обучения. В режиме чистого сумматора, при $\mathrm{g}_{\mathrm{sg}}(\gamma) \to 0$, ΣΠ-нейрон представляет собой классический нейрон Мак-Каллока – Питтса. При $\mathrm{g}_{\mathrm{sg}}(\gamma) \to 1$ ΣΠ-нейрон осуществляет селективное перемножение входов, множители выбираются посредством вентилей $\mathrm{g}_{\mathrm{sg}}(\alpha_i) \to 1$. В промежуточных случаях ΣΠ-нейрон реализует линейную комбинацию из суммы и произведения компонент входного вектора $\mathbf{x}$. Отметим, что конструкция вентилей (использование сигмоидальной функции g_{sg}) аналогична таковой в вентильном подходе [20] и LSTM-сетях [24].

Конструкцию ΣΠ-нейрона можно обобщить на случай структурной нейропластичности[5], введя адаптивную зависимость γ от текущих входных данных $\mathbf{x}$, скрытого состояния нейросети $\mathbf{h}$, и/или выхода сети $\mathbf{z}$. Схематично нейропластичность может иметь вид:

$$\gamma = \mathrm{G}(\circ),$$

[5] Нейропластичность в глубоком обучении – это адаптационный механизм, реализующийся в форме способности нейросети динамически изменять свою структуру и/или функционал в процессе эксплуатации в ответ на изменяющиеся входные данные или условия / задачи выполнения.

где G – некоторое отображение $\mathbb{R}^K \to \mathbb{R}$, настраиваемое при обучении/эксплуатации нейросети.

1.2. Основные свойства Sigma-Pi-нейрона

1.2.1. Топология разделяющих поверхностей

В отличие от классического нейрона Мак-Каллока – Питтса, ΣΠ-нейрон порождает существенно более разнообразные, качественно различные разделяющие поверхности. Так, в режиме бинарной классификации ΣΠ-нейрон порождает три качественно различных фазовых портрета разделения входных данных на классы. В общем виде данные случаи записываются в форме:

$$\mathrm{F_S} : z = \mathrm{g}_{11}\left[\sum_{i=1}^{N} x_i\right], \qquad (9)$$

$$\mathrm{F_{SP}} : z = \mathrm{g}_{11}\left[\sum_{i=1}^{N} x_i + \prod_{i=1}^{N} x_i\right], \qquad (10)$$

$$\mathrm{F_P} : z = \mathrm{g}_{11}\left[\prod_{i=1}^{N} x_i\right], \qquad (11)$$

причём случай $\mathrm{F_S}$ соответствует классическому суммирующему нейрону. Графически фазовые портреты (9)–(11) для случая двумерного входного вектора изображены на рис. 2, для трёхмерного входного вектора – на рис. 3.

Как видно из рис. 2 и 3, при включении в агрегирующий оператор нейрона операции *произведения* – топология разделения вектора входных данных на классы существенно усложняется – появляется возможность решения единственным нейроном нелинейных задач.

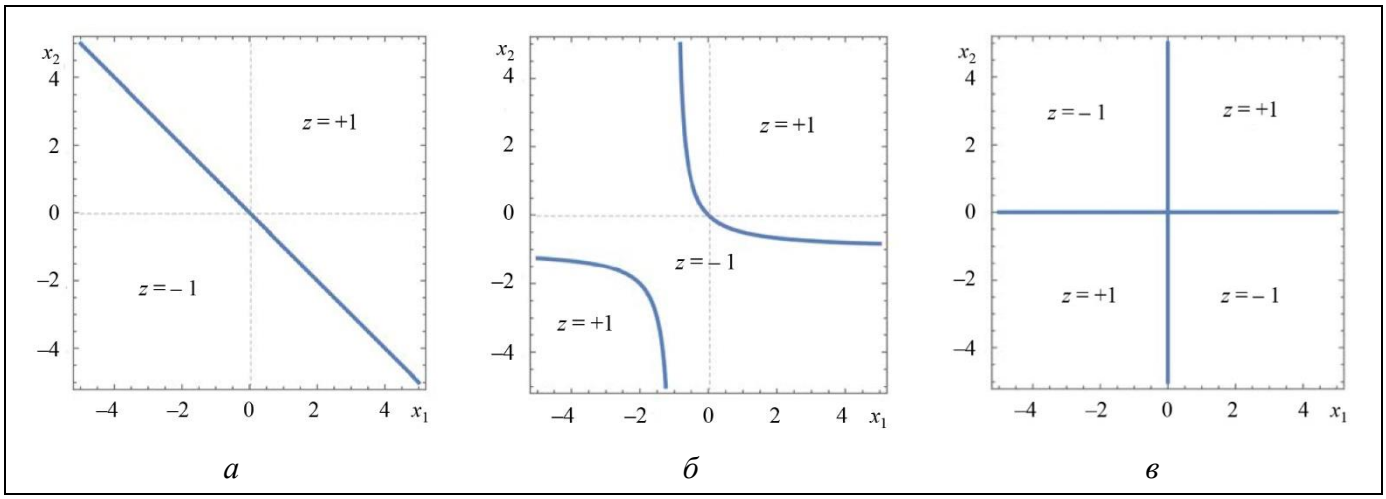

Рис. 2. Структура разделяющих поверхностей в случае бинарной классификации двумерных входных данных нейронами вида: *а* – $\mathrm{F_S}$; *б* – $\mathrm{F_{SP}}$; *в* – $\mathrm{F_P}$

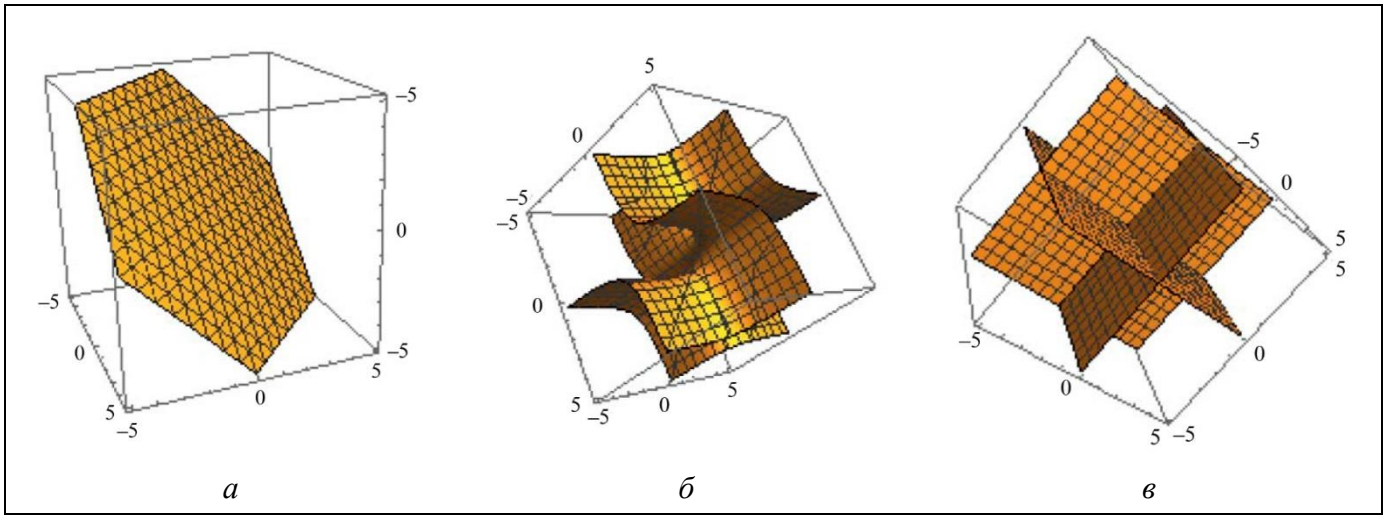

Рис. 3. Структура разделяющих поверхностей в случае бинарной классификации трёхмерных данных нейронами вида: *а* – $\mathrm{F_S}$; *б* – $\mathrm{F_{SP}}$; *в* – $\mathrm{F_P}$

Действительно, случай F_S (соответствующий классическому суммирующему нейрону) формирует одну разделяющую гиперплоскость и две области (по одной на каждый класс) при любой размерности входных данных ($\dim \mathbf{x} \geq 2$). Случай F_{SP} формирует уже нелинейную разделяющую гиперповерхность. При $\dim \mathbf{x} = 2$ образуются три изолированные области (две области на класс $z = +1$ и одна на класс $z = -1$). При $\dim \mathbf{x} \geq 3$ формируются две области (по одной на каждый класс). Случай F_P формирует $2^{\dim \mathbf{x}}$ изолированных областей (равное количество областей на каждый класс).

1.2.2. Вычисление булевой функции XOR

В работе [25] было доказано, что сети с нейронами вида (1), состоящие из единственного слоя, неспособны решать даже элементарные несепарабельные задачи, например вычислять булеву функцию XOR (исключающее «или»). В свою очередь, исходя из результатов п. 1.2.1, справедлива

Теорема 4. *Единственный* $\Sigma\Pi$*-нейрон в бинарном входном* $\mathbf{x} \in \mathbb{Z}_2 \times \ldots \times \mathbb{Z}_2$ *и выходном* $z \in \mathbb{Z}_2$ *базисах способен вычислять булеву функцию XOR* $\oplus_2$ *произвольной размерности,* $N = \dim \mathbf{x} \geq 2$.

Доказательство. Из свойств функции $\oplus_2$ известно, что результат выполнения операции будет «ИСТИНА» только тогда, когда количество аргументов, равных «ИСТИНА», составляющих текущий входной набор, – нечётное. Отсюда определим базис $\mathbb{Z}_2 = \{-1, 1\}$, а конфигурацию $\Sigma\Pi$-нейрона, реализующую $\oplus_2$, в случае нечётной размерности $\mathbf{x}$, представим в виде:

$$N \bmod 2 \neq 0: \ \Sigma\Pi = \prod_{i=1}^{N} x_i, \ z = g_{11}(\Sigma\Pi). \qquad (12)$$

При чётной размерности $\mathbf{x}$ конфигурацию $\Sigma\Pi$-нейрона выразим в виде:

$$N \bmod 2 = 0: \ \Sigma\Pi = \frac{1}{N}\sum_{i=1}^{N} x_i - 2\prod_{i=1}^{N} x_i - 1, \ z = g_{11}(\Sigma\Pi). \qquad (13)$$

Проверка выражений (12) и (13) при малых N и далее по индукции подтверждает справедливость теоремы. ♦

Для наглядности на рис. 4 приведён фазовый портрет (13) для случая двух переменных.

Следует отметить, что теорема 4 важна как с теоретической стороны (подтверждение возможности $\Sigma\Pi$-нейроном решать некоторые классы несепарабельных задач), так и с прикладной: булева функция XOR находит применение в самых разных областях, от низкоуровневого программирования и криптографии до алгоритмов поиска и обнаружения ошибок. Это потенциально способствует возможности прямого решения подобных задач нейросетями на основе $\Sigma\Pi$-нейронов.

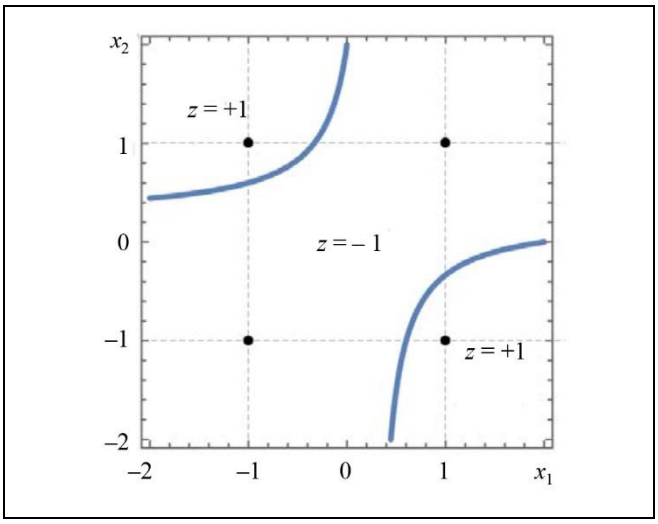

Рис. 4. Структура разделяющих поверхностей в случае вычисления булевой функции XOR (дисками выделены входные значения в базисе $\mathbb{Z}_{\{-1,1\}}$)

2. ЧИСЛЕННЫЕ ЭКСПЕРИМЕНТЫ

2.1. Общие положения

С целью всестороннего анализа свойств $\Sigma\Pi$-нейрона и отработки схем построения и обучения сетей на его основе были разработаны два фреймворка. Первый – сугубо исследовательский в среде Wolfram Mathematica, второй – более прикладной, в виде расширения PyTorch[6]. С использованием данных фреймворков в том числе проводились численные эксперименты по моделированию $\Sigma\Pi$-нейросетями различных функциональных зависимостей и сравнения результатов с классическими полносвязными сетями. Некоторые из полученных результатов представлены ниже.

Исходно была сформулирована задача: исследовать моделирующие свойства $\Sigma\Pi$-сетей в части аппроксимации простых многочленов двух переменных (п. 2.2) и скалярного умножения векторов и векторно-матричного умножения (п. 2.3) внутри и вне домена обучения.

Домен обучения D^{TR} для каждой переменной (компоненты переменной) определяется отрезком:

[6] Расширение для PyTorch написано Р. Ю. Порцевым совместно с сотрудником лаборатории № 77 ИПУ РАН В. В. Казаковым.

$$x \in \mathrm{D}^{\mathrm{TR}} \subset \mathbb{R},\ \mathrm{D}^{\mathrm{TR}} = \left[-2,2;\ 2,2\right], \qquad (14)$$

где x – входная переменная (компонента переменной) анализируемых зависимостей. Домен тестирования D^{TS} несколько превосходит по своему размеру домен обучения:

$$x \in \mathrm{D}^{\mathrm{TS}} \subset \mathbb{R},\ \mathrm{D}^{\mathrm{TS}} = \left[-3,5;\ 3,5\right].$$

Кроме того, был определён расширенный домен валидации $\overline{\mathrm{D}}^{\mathrm{VL}}$:

$$x \in \overline{\mathrm{D}}^{\mathrm{VL}} \subset \mathbb{R},\ \overline{\mathrm{D}}^{\mathrm{VL}} = \left[-110;\ 110\right]. \qquad (15)$$

Из сравнения формул (15) и (14) следует, что по амплитуде расширенный домен валидации превосходит домен обучения в 50 раз.

Внутри доменов обучения D^{TR} и тестирования D^{TS} были сформированы соответственно два непересекающихся датасета $\mathrm{DS}^{\mathrm{TR}} \bigcap \mathrm{DS}^{\mathrm{TS}} \equiv \varnothing$: $\mathrm{DS}^{\mathrm{TR}}$ – для обучения, 40 000 семплов; $\mathrm{DS}^{\mathrm{TS}}$ – для тестирования, 10 000 семплов. Их аргументы исходно представлялись псевдослучайными числами с равномерным распределением:

$$\mathrm{DS}^{\mathrm{TR}}:\ x \in \mathcal{U}\left[\mathrm{D}^{\mathrm{TR}}\right],\ \mathrm{DS}^{\mathrm{TS}}:\ x \in \mathcal{U}\left[\mathrm{D}^{\mathrm{TS}}\right].$$

Затем к значениям функций этих датасетов добавлялся аддитивный шум с нормальным распределением:

$$z \mapsto z + \varepsilon,\ \varepsilon \in \mathcal{N}\left[\mu = 0,\ \sigma = 0{,}05\right],$$

который, с одной стороны, моделировал некие ошибки измерений, с другой – выполнял регуляризующую функцию.

Дополнительно были сформированы валидирующие датасеты (по 100 000 семплов каждый):

$$\mathrm{DS}^{\mathrm{VL}}:\ x \in \mathcal{U}\left[\mathrm{D}^{\mathrm{TR}}\right],\ \overline{\mathrm{DS}}^{\mathrm{VL}}:\ x \in \mathcal{U}\left[\overline{\mathrm{D}}^{\mathrm{VL}}\right],$$

по которым делался окончательный анализ качества и устойчивости функционирования исследуемых нейросетей. Естественно, $\mathrm{DS}^{\mathrm{TR}} \bigcap \mathrm{DS}^{\mathrm{VL}} \equiv \varnothing$ и $\mathrm{DS}^{\mathrm{TS}} \bigcap \mathrm{DS}^{\mathrm{VL}} \equiv \varnothing$.

Формирование $\Sigma\Pi$-нейросетей осуществлялось исходя из условия минимальной функциональной достаточности – задавалось такое минимальное число слоёв и нейронов, которое функционально может реализовать модел ируемую зависимость. Выборочно рассматривались также более слабые и/или напротив, избыточные сети.

Решения для $\Sigma\Pi$-сетей сравнивались с подобными им классическими MLP[7] полносвязными архитектурами (далее FC-нейросетями), близкими по своей сложности (по числу слоёв и обучаемых параметров). Все исследуемые нейросети обучались во фреймворке PyTorch версии 2.1.0. При обучении сетей использовалась функция потерь L в виде среднеквадратичной ошибки MSE:

$$L_{\mathrm{MSE}} = \frac{1}{M}\sum_{i=1}^{M}\left(z_i - \hat{z}_i\right)^2,$$

где M – число семплов в наборе данных; z_i – фактическое значение i-го семпла; $\hat{z}_i$ – предсказанные значения i-го семпла. Каждая архитектура инициализировалась не менее чем пятью разными вариантами весов и каждый вариант обучался оптимизатором RMSprop в течение 200 эпох с размером минибатча в 32 семпла (минибатч формировался специальным образом в зависимости от номера эпохи).

Далее отбирались лучшие (top-1) сети для каждой функции и архитектуры по правилу

$$L_{\mathrm{MSE}} \mid \mathrm{DS}^{\mathrm{TS}} \to \min,$$

причём выбор производился на пространстве «инициализация» х «эпоха обучения».

С целью корректного сравнения свойств нейросетей на доменах $\mathrm{DS}^{\mathrm{VL}}$ и $\overline{\mathrm{DS}}^{\mathrm{VL}}$ (в том числе для междоменного анализа) применялась метрика APE (*Absolute Percentage Error*):

$$R_{\mathrm{APE}} = \left|\frac{z_i - \hat{z}_i}{z_i}\right|, \qquad (16)$$

для которой в рамках каждой сети и каждого валидационного датасета определялись её статистические квантили следующих порядков:

$$\alpha = \overline{0{,}25;\ 0{,}5;\ 0{,}75;\ 0{,}9;\ 0{,}99}. \qquad (17)$$

Кроме того, для каждого эксперимента в случае необходимости анализировались также специфические характеристики.

2.2. Простые многочлены двух переменных

Объект «многочлен» является едва ли не главным объектом исследования для классической алгебры. Подобные объекты также играют важнейшую роль в математическом моделировании. Это связано с тем, что любую непрерывную функцию на интервале можно с произвольно малой ошибкой заменить на многочлен (см. теорему 3), что позволяет приближённо выражать многочленами широкие классы функций.

[7] *Multylayer Perceptron* – многослойный перцептрон [3].

В настоящем подразделе исследуются возможности моделирования ΣΠ-нейросетями и классическими полносвязными сетями (FC-нейросетями) простых многочленов двух переменных. Для экспериментов были отобраны пять многочленов:

$$f_1 = -x^2 y^2, \quad (18)$$

$$f_2 = (3x - y)^2, \quad (19)$$

$$f_3 = x^2 - y^2, \quad (20)$$

$$f_4 = x^2 y + x y^2, \quad (21)$$

$$f_5 = x^2 + y^2. \quad (22)$$

Выбор в качестве объекта моделирования именно этих выражений объясняется, с одной стороны, требованием относительной простоты для возможности интерпретации полученных результатов; с другой стороны, необходима реализация достаточно нетривиальных зависимостей (наличие умножений и возведения в степень, приводимость/неприводимость к множителям).

Конфигурации нейросетей, использовавшихся для моделирования выражений (18)–(22) приведены в табл. 1. Во всех полносвязных нейросетях между слоями использовалась функция активации ReLU.

Отметим, что многочлен f_5 моделировался тремя вариантами нейросетей (см. эксперименты E5.1 и E5.2 в табл. 1). В первом случае использовалась двухслойная сеть, которая заведомо не реализует функцию f_5, во втором – уже трёхслойная, минимально достаточная для реализации моделируемой зависимости, в третьем – также трёхслойная, но с избыточным нейроном в первом слое.

Для отобранных top-1 нейросетей набор статистических квантилей (17) по метрике (16) на датасетах $\mathrm{DS}^{\mathrm{VL}}$ и $\overline{\mathrm{DS}}^{\mathrm{VL}}$ приведен на рис. 5.

Таблица 1

Конфигурации нейросетей для моделирования многочленов

Многочлен	ΣΠ- нейросеть		FC-нейросеть		Эксперимент
	Архитектура: слои / нейроны	Число обучаемых параметров	Архитектура слои / нейроны	Число обучаемых параметров	
$f_1 = -x^2 y^2$	2 / 2+1	21	2 / 5+1	21	E1
$f_2 = (3x - y)^2$	2 / 2+1	21	2 / 5+1	21	E2
$f_3 = x^2 - y^2$	2 / 2+1	21	2 / 5+1	21	E3
$f_4 = x^2 y + x y^2$	2 / 2+1	21	2 / 5+1	21	E4
$f_5 = x^2 + y^2$	2 / 2+1	21 ↓	2 / 5+1	21	E5.1
$f_5 = x^2 + y^2$	3 / 3+2+1	46	3 / 6+4+1	51	E5.2
$f_5 = x^2 + y^2$	3 / 4+2+1	57 ↑	3 / 7+4+1	58	E5.3

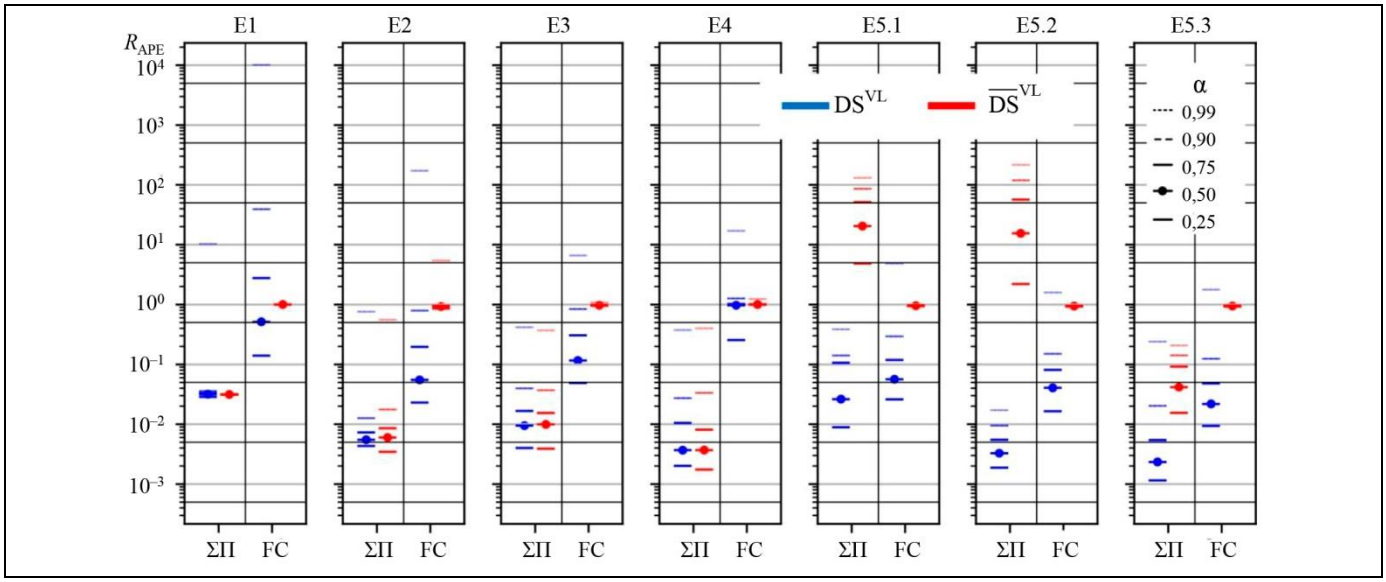

Рис. 5. Уровень ошибок обученных нейросетей для функций f_1 — f_5 и экспериментов E1 – E5.3

Анализ данных рис. 5 показывает, что ΣΠ-нейросети в экспериментах E1–E4 весьма точно аппроксимировали функции $f_1 - f_4$ соответственно. На это указывает как собственно низкое значение ошибок (медиана менее 0,04), так и практически равное значение медианы ошибок на датасетах $\mathrm{DS}^{\mathrm{VL}}$ и $\overline{\mathrm{DS}}^{\mathrm{VL}}$ в пределах одного эксперимента. Напомним, что по амплитуде расширенный датасет валидации $\overline{\mathrm{DS}}^{\mathrm{VL}}$ превосходит домен обучения в 50 раз, а по площади (две независимые переменные) в 2500 раз. В свою очередь, на датасете $\overline{\mathrm{DS}}^{\mathrm{VL}}$ ошибки FC-нейросетей по медиане существенно превосходят ошибки ΣΠ-нейросетей: более чем в 50 раз для функции f_1 и более чем в 100 раз для функций $f_2 - f_4$. При этом даже в пределах домена обучения FC-нейросети демонстрируют на порядок худшие значения ошибок относительно ΣΠ-нейросетей, а вне его они полностью деградируют. Картина результатов для экспериментов с моделированием функции f_5 выглядит сложнее. В пределах домена обучения даже функционально недостаточная ΣΠ-нейросеть (эксперимент E5.1) демонстрирует вполне приемлемую работоспособность и в 2,5 раза меньший уровень ошибки, нежели эквивалентная ей классическая FC-нейросеть. Естественно, более сложные сети (эксперименты E5.2 и E5.3) имеют лучшие результаты, и разрыв с FC-нейросетями увеличивается. Но работоспособность вне домена обучения имеет только избыточная ΣΠ-нейросеть (эксперимент E5.3), при этом эквивалентная ей FC-нейросеть полностью деградирована. Неработоспособность минимально достаточной ΣΠ-нейросети (эксперимент E5.2) за пределами домена обучения объясняется, скорее всего, её недообученностью.

Понятно, что в силу архитектурной ущербности FC-нейросети неспособны реализовывать прямое моделирование многочленов вида (18)–(22) и в настоящей серии экспериментов работают в режиме полной мемоизации. Тем интереснее исследовать реальный механизм реализации выражений (18)–(22) ΣΠ-нейросетями. Для этого рассмотрим собственно значения обучаемых параметров обученных ΣΠ-нейросетей. Запишем коэффициенты многочленов (18)–(22) компактно в виде квадратной матрицы размерности 3×3:

$$\mathcal{M}(f)=\begin{pmatrix} k^{(0,0)} & \cdots & k^{(0,2)} \\ \vdots & \ddots & \vdots \\ k^{(2,\,0)} & \cdots & k^{(2,2)} \end{pmatrix},$$

где элемент $k^{(n,m)}$ соответствует коэффициенту при $x^n y^m$. Тогда для рассматриваемых функций и обученных ΣΠ-нейросетей (после элементарных преобразований весов) матрицы коэффициентов будут выглядеть так, как они представлены в табл. 2.

Из данных табл. 2 наглядно видно, что во всех экспериментах, кроме E5.1, ΣΠ-нейросети достаточно хорошо восстанавливают теоретически обусловленные коэффициенты полинома (выделены жирным шрифтом) и имеют малое количество паразитных коэффициентов заметной величины (значения более 0,01 по модулю выделены рамкой). Это – свидетельство в пользу того, что данные ΣΠ-нейросети функционируют в режиме прямого моделирования соответствующих функциональных зависимостей. В случае же эксперимента E5.1 функционально недостаточная архитектура сети (см. табл. 1) обучается скорее в режиме мемоизации, фактически вместо теоретической функции (22) реализуя «близкую» ей внутри домена обучения функцию. При этом следует отметить, что убрать паразитные коэффициенты для архитектуры эксперимента E5.1 теоретически невозможно, так как двухслойная ΣΠ-нейросеть неспособна выразить функцию (22) в чистом виде.

В свою очередь, сравнение результатов экспериментов E5.1 и E5.2 демонстрирует повышение выразительной способности ΣΠ-нейросетей с увеличением их глубины (это также справедливо и для классических полносвязных архитектур).

2.3. Скалярное умножение

Помимо многочленов, в практике и теории математического моделирования существенную роль играют также операции между векторами и матрицами. В настоящем разделе исследуются возможности моделирования ΣΠ-нейросетями и классическими полносвязными сетями двух операций:

$$\mathbf{v}_1 = \mathbf{x}\cdot\mathbf{y}, \tag{23}$$

$$\mathbf{v}_2 = \mathbf{X}\mathbf{y}, \tag{24}$$

Таблица 2

Матрицы коэффициентов для моделируемых многочленов и Sigma-Pi-нейросетей

Эксперимент	Многочлен	$\mathcal{M}\mid f$	$\mathcal{M}\mid$ ΣП- нейросеть
E1	$f_1 = -x^2 y^2$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 4{,}159\times10^{-4} & 1{,}394\times10^{-5} & -1{,}06\times10^{-7} \\ -6{,}988\times10^{-5} & -2{,}311\times10^{-3} & -6{,}646\times10^{-4} \\ 9{,}222\times10^{-10} & 3{,}149\times10^{-4} & \mathbf{-1{,}031} \end{pmatrix}$
E2	$f_2 = (3x-y)^2$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & -6 & 0 \\ 9 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 6{,}268\times10^{-3} & 3{,}737\times10^{-3} & \mathbf{0{,}9962} \\ \boxed{1{,}252\times10^{-2}} & \mathbf{-5{,}977} & -1{,}758\times10^{-4} \\ \mathbf{8{,}946} & 5{,}262\times10^{-4} & 7{,}736\times10^{-9} \end{pmatrix}$
E3	$f_3 = x^2 - y^2$	$\begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} -8{,}082\times10^{-6} & 2{,}078\times10^{-3} & \mathbf{-1{,}011} \\ 6{,}149\times10^{-4} & 5{,}871\times10^{-3} & -1{,}702\times10^{-5} \\ \mathbf{1{,}003} & 3{,}629\times10^{-5} & 2{,}557\times10^{-10} \end{pmatrix}$
E4	$f_4 = x^2 y + x y^2$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 2{,}241\times10^{-4} & 5{,}109\times10^{-4} & -1{,}856\times10^{-4} \\ 5{,}085\times10^{-4} & 4{,}455\times10^{-3} & \mathbf{0{,}9985} \\ -6{,}544\times10^{-4} & \mathbf{0{,}9988} & 1{,}177\times10^{-4} \end{pmatrix}$
E5.1	$f_5 = x^2 + y^2$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} \boxed{-2{,}213\times10^{-2}} & \boxed{-7{,}81\times10^{-2}} & \mathbf{1{,}058} \\ \boxed{-7{,}965\times10^{-2}} & \boxed{7{,}107\times10^{-2}} & -3{,}908\times10^{-3} \\ \mathbf{1{,}061} & -3{,}95\times10^{-3} & \boxed{-2{,}526\times10^{-2}} \end{pmatrix}$
E5.2	$f_5 = x^2 + y^2$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} -7{,}277\times10^{-4} & 3{,}595\times10^{-4} & \mathbf{0{,}9991} \\ 2{,}591\times10^{-3} & -6{,}276\times10^{-3} & 3{,}717\times10^{-3} \\ \mathbf{0{,}9956} & 3{,}969\times10^{-3} & -3{,}412\times10^{-6} \end{pmatrix}$
E5.3	$f_5 = x^2 + y^2$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} \boxed{1{,}562\times10^{-2}} & -7{,}9\times10^{-4} & \mathbf{0{,}9965} \\ -1{,}295\times10^{-4} & -1{,}577\times10^{-3} & -4{,}803\times10^{-4} \\ \mathbf{0{,}9938} & -2{,}412\times10^{-4} & -2{,}721\times10^{-5} \end{pmatrix}$

– соответственно скалярного умножения векторов **x** и **y** и умножения вектора-столбца **y** на матрицу **X** слева. Выбор в качестве объекта моделирования именно этих выражений опять же объясняется, с одной стороны, требованием относительной простоты для возможности интерпретации полученных результатов, с другой стороны, эти операции – ключевые в векторно-матричной алгебре и они имеют весьма богатое прикладное применение.

Конфигурации нейросетей, использовавшихся для моделирования выражений (23) и (24), приведены в табл. 3. Во всех полносвязных нейросетях между слоями использовалась функция активации ReLU.

Для отобранных top-1 нейросетей набор статистических квантилей (17) по метрике (16) на датасетах $\mathrm{DS}^{\mathrm{VL}}$ и $\overline{\mathrm{DS}}^{\mathrm{VL}}$,приведен на рис. 6.

Из рис. 6 видно, что во всех экспериментах ΣП-нейросети существенно превосходят эквивалентные им классические FC-нейросети. Исключением является ΣП-нейросеть в эксперименте E6 на данных вне домена обучения. Аномальный рост ошибок скорее всего объясняется её недообученностью. Выяснение причин этого, а также анализ аналогичной ситуации с экспериментом E5.2 и изучение особенностей обучения ΣП-нейросетей в целом требуют дополнительных исследований.

Таблица 3

Конфигурации нейросетей для моделирования операций векторно-матричной алгебры

Операция	ΣΠ- нейросеть		FC-нейросеть		Эксперимент
	Архитектура: слои / нейроны	Число обучаемых параметров	Архитектура слои / нейроны	Число обучаемых параметров	
$\mathbf{v}_1 = \mathbf{x}\cdot\mathbf{y}$, $\dim\mathbf{x}=3$	2 / 3+1	54	2 / 7+1	57	E6
$\mathbf{v}_1 = \mathbf{x}\cdot\mathbf{y}$, $\dim\mathbf{x}=4$	2 / 4+1	87	2 / 9+1	91	E7
$\mathbf{v}_2 = \mathbf{X}\mathbf{y}$, $\dim\mathbf{X}=2\times2$	2 / 4+2	82	2 / 9+2	83	E8

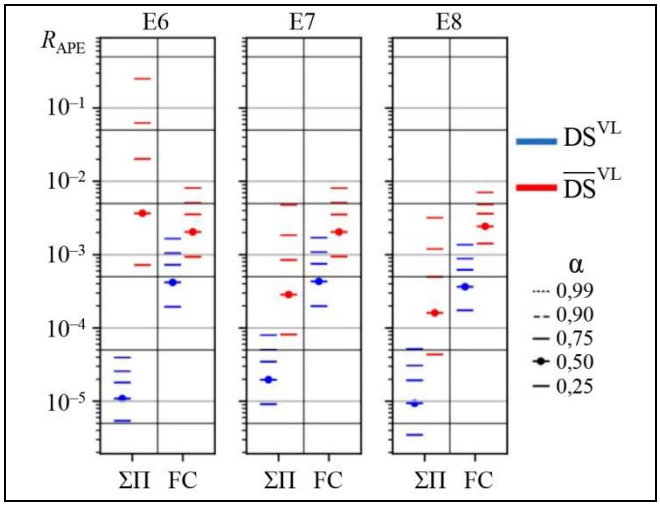

Рис. 6. Уровень ошибок обученных нейросетей для функций v_1 и v_2 и экспериментов E7 и E8

ЗАКЛЮЧЕНИЕ

В работе предложена и обоснована в первом приближении конструкция ΣΠ-нейрона, решающая достаточно органично проблему «перемножения факторов» в искусственных нейронных сетях. Несомненными достоинствами решения являются:

• Прозрачная интеграция с текущими и перспективными архитектурами нейросетей как в режиме вывода, так и в режиме обучения. Причём в силу свойств ΣΠ-нейрона в сети допустимо достаточно свободное перемешивание используемых слоёв.

• Автоматическое, управляемое на этапе обучения включение операции перемножения факторов. Кроме того, возможна реализация режима нейропластичности, когда перемножение включается при определенных входных данных и/или при определённом скрытом состоянии сети.

• Возможность решения единственным нейроном некоторых несепарабельных задач. В работе доказана реализация единственным ΣΠ-нейроном логической булевой функции «XOR» произвольной размерности.

• Выявление в данных в символьной форме композиционных структур полиномиального, векторного и матричного характеров. Данный аспект продемонстрирован в статье и является важным в задачах класса AI+Science.

• Качество и устойчивость функционирования ΣΠ-нейросетей стабильно лучше по сравнению с классическими полносвязными сетями, т. е. ΣΠ-нейросети формируют меньшие ошибки при обучении и выводе, чем классические многослойные перцептроны, при том же количестве параметров. В полную силу это раскрывается на задачах с рабочими входными данными вне домена обучения.

В тоже время текущая реализация ΣΠ-нейрона обладает явным недостатком – в нейроне не реализовано деление факторов. Кроме того, в силу математических особенностей (6)–(8) ΣΠ-нейросети для лучшего раскрытия своего потенциала требуют некоторой модернизации классического алгоритма обратного распространения ошибки и разработки соответствующей функции потерь. Решение данных вопросов является предметом текущих исследований авторов.

Таким образом, следует отметить, что предложенные в работе решения, с одной стороны, повышают «выразительность» нейросети и делают её более компактной, с другой – улучшают её интерпретируемость. Представляется, что предложенная конструкция ΣΠ-нейрона найдёт своё применение не только в суррогатных расчётно-моделирующих нейросетевых моделях, но и в задачах класса AI+Science в целом.

Авторы выражают благодарность академику Д. А. Новикову за внимание к работе, Д. С. Гаджиеву за активное обсуждение затрагиваемых в работе вопросов и ряд ценных замечаний, а также В. В. Казакову за реализацию ΣΠ-нейрона во фреймворке PyTorch.

ЛИТЕРАТУРА

1. *McCulloch, W.S., Pitts, W.* A Logical Calculus of the Ideas Immanent in Nervous Activity // The Bulletin of Mathematical Biophysics. – 1943. – Vol. 5. – P. 115–133.
2. *Rosenblatt, F.* The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain // Psychological Review. – 1958. – Vol. 65, no. 6. – P. 386–408.
3. *Макаренко А.В.* Глубокие нейронные сети: зарождение, становление, современное состояние // Проблемы управления. – 2020. – № 2. – С. 3–19. [*Makarenko, A.V.* Deep Neural Networks: Origins, Development, Current Status // Control Sciences. – 2020. – No. 2. – P. 3–19. (In Russian)]
4. *Vaswani, A., Shazeer, N., Parmar, N.,* et al. Attention Is All You Need // arXiv: 1706.03762. – 2017. – DOI: https://doi.org/10.48550/arXiv.1706.03762
5. *Dehghani, M., Gouws, S., Vinyals, O.,* et al. Universal Transformers // arXiv: 1807.03819. – 2018. – DOI: https://doi.org/10.48550/arXiv.1807.03819
6. *Liu, Z., Wang, Y., Vaidya, S.,* et al. KAN: Kolmogorov-Arnold Networks // arXiv: 2404.19756. – 2024. – DOI: https://doi.org/10.48550/arXiv.2404.19756
7. *Parallel* Distributed Processing: Explorations in the Microstructures of Cognition / Ed. by D.E. Rumelhart, J.L. McClelland. – Cambridge, MA: MIT Press, 1986.
8. *Cybenko, G.* Approximation by Superpositions of a Sigmoidal Function // Mathematics of Control Signals and Systems. – 1989. – Vol. 2, no. 4. – P. 303–314.
9. *Розенблатт Ф.* Принципы нейродинамики: Перцептроны и теория механизмов мозга. – М.: Мир, 1965. – 478 с. [*Rosenblatt, F.* Principles of Neurodynamics: Perceptions and the Theory of Brain Mechanisms. – New York: Cornell University, 1961. – 625 p.]
10. *Nielsen, R.* Kolmogorov's Mapping Neural Network Existence Theorem // Proc. of the IEEE 1st Int. Conference on Neural Networks. – San Diego, CA, 1987. – Vol. 3. – P. 11–13.
11. *Hornik, K.* Approximation Capabilities of Multilayer Feedforward Networks // Neural Networks. – 1991. – Vol. 4, no. 2. – P. 251–257.
12. *Hornik, K., Stinchcombe, M., White, H.* Multilayer Feedforward Networks Are Universal Approximators // Neural Networks. – 1989. – Vol. 2, no. 5. – P. 359–366.
13. *Leshno, M., Lin, V.Ya., Pinkus, A., Schocken, S.* Multilayer Feedforward Networks with a Non-Polynomial Activation Function Can Approximate Any Function. NYU Working Paper No. IS-92-13. – 1992.
14. *Maiorov, V., Pinkus, A.* Lower Bounds for Approximation by MLP Neural Networks // Neurocomputing. – 1999. – Vol. 25, no. 1–3. – P. 81–91.
15. *Колмогоров А.Н.* О представлении непрерывных функций нескольких переменных суперпозициями непрерывных функций меньшего числа переменных // Докл. АН СССР. – 1957. – Т. 108. – С. 179–182. [*Kolmogorov, A.N.* On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition // In: American Mathematical Society Translations: Series 2. – 1963. – Vol. 28. – P. 55–59.]
16. *Арнольд В.И.* О представлении функций нескольких переменных в виде суперпозиции функций меньшего числа переменных // Математика, ее преподавание, приложения и история, Матем. просв., сер. 2, 3. – 1958. – С. 41–61. [*Arnold, V.I.* On the Representation of Functions of Several Variables as a Superposition of Functions of a Smaller Number of Variables // In: Vladimir I. Arnold – Collected Works: Representations of Functions, Celestial Mechanics, and KAM Theory 1957–1965. – Vol. 1. – Berlin–Heidelberg: Springer, 2009. – P. 57–77.]
17. *Арнольд В.И.* О функции трёх переменных // Доклады АН СССР. – 1957. – Т. 114. – С. 679–681. [*Arnold, V.I.* On Functions of Three Variables // In: American Mathematical Society Translations. Series 2. – 1963. – Vol. 28. – P. 51–54.]
18. *Weierstrass, K.* Über die analytische Darstellbarkeit sogenannter willkürlicher Functionen einer reellen Veränderlichen // Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin. – 1885. – Bd. II. – S. 633–639 (Mitteilung 1), 789–805 (Mitteilung 2). (In German.)
19. *Витушкин А.Г.* О многомерных вариациях. – М.: Физматгиз, 1955. [*Vitushkin, A.G.* On Multidimensional Variations. – M.: Fizmatgiz, 1955. (In Russian)]
20. *Dauphin, Y.M., Fan, A., Auli, M., Grangier, D.* Language Modeling with Gated Convolutional Networks // arXiv: 1612.08083. – 2016. – DOI: https://doi.org/10.48550/arXiv.1612.08083
21. *Shazeer, N.* GLU Variants Improve Transformer // ArXiv: 2002.05202. – 2020. – DOI: https://doi.org/10.48550/arXiv.2002.05202
22. *Mnih, A., Hinton, G.* Three New Graphical Models for Statistical Language Modelling // Proceedings of the 24th International Conference on Machine learning. – Oregon, 2007. – P. 641–648.
23. *Liu, Z., Ma, P., Wang, W.,* et al. KAN 2.0: Kolmogorov-Arnold Networks Meet Science // arXiv: 2408.10205. – 2024. – DOI: https://doi.org/10.48550/arXiv.2408.10205
24. *Hochreiter, S., Schmidhuber, J.* Long Short-Term Memory // Neural Computation. – 1997. – Vol. 9, no. 8. – P. 1735–1780.
25. *Minsky, M., Papert, S.* Perceptrons: An Introduction to Computational Geometry. – Cambridge, MA: MIT Press, 1969.

Статья представлена к публикации членом редколлегии академиком РАН Д. А. Новиковым.

Поступила в редакцию 09.07.2026,
после доработки 10.07.2026.
Принята к публикации 10.07.2026.

Порцев Руслан Юрьевич – мл. науч. сотрудник,
✉ poruss@mail.ru
ORCID iD: https://orcid.org/0009-0009-4008-8725

Макаренко Андрей Викторович – канд. техн. наук,
✉ avm.science@mail.ru
ORCID iD: https://orcid.org/0000-0002-3875-7549

Институт проблем управления им. В. А. Трапезникова РАН,
г. Москва

THE SIGMA-PI NEURON AND RESOLVING THE GENERAL "FACTOR MULTIPLICATION" PROBLEM IN ARTIFICIAL NEURAL NETWORKS

R. Yu. Portsev* and A. V. Makarenko**

*,**Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia

*✉ poruss@mail.ru, **✉ avm.science@mail.ru

Abstract. This paper proposes a novel design of the Sigma-Pi neuron with the controlled automatic activation of the factor multiplication operation during the training phase of a neural network. The solution can be transparently integrated into both current and prospective neural architectures; it performs direct neural computation of a wide range of mathematical relations that are commonly implemented by neural networks in a memoization mode. The direct execution of the multiplication operation yields smaller network errors; this advantage is very clear when the network processes input data outside the training domain. As proved below, a single Sigma-Pi neuron can implement the XOR function of arbitrary dimension. The effectiveness and computational capabilities of the novel design, as well as its superiority over classical fully connected feedforward neural networks, are demonstrated for the neural modeling of polynomials and operations of vector-matrix algebra. The proposed Sigma-Pi neuron design is expected to find application both in surrogate neural network models for computation and simulation and in general AI+Science tasks.

Keywords: Sigma-Pi neuron, factor multiplication operation, XOR function, surrogate models.

Acknowledgments. The authors are grateful to Academician D.A. Novikov for his permanent attention to this work, to D.S. Gadzhiev for his active discussion of the related issues and valuable remarks, and to V.V. Kazakov for his implementation of the $\Sigma\Pi$ neuron in the PyTorch framework.