Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

П. П. Урбанович, М. Д. Плонковски, М. Долецки

НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ В КРИПТОГРАФИЧЕСКИХ ПРИЛОЖЕНИЯХ

Минск 2024

УДК 004.032.26(07)
БКК 32.818.1я73
У69

Рассмотрена и рекомендована к изданию редакционно-издательским советом Белорусского государственного технологического университета.

Р е ц е н з е н т ы:
профессор, доктор технических наук,
заведующий кафедрой интеллектуальных информационных технологий УО «Брестский государственный технологический университет» *В. А. Головко*;
профессор, доктор технических наук,
главный научный сотрудник лаборатории идентификации систем ГНУ «Объединенный институт проблем информатики НАН Беларуси» *В. В. Старовойтов*

Урбанович, П. П.
У69 Нейросетевые технологии в криптографических приложениях : монография / П. П. Урбанович, М. Д. Плонковски, М. Долецки. – Минск : БГТУ, 2024. – 223 с.
ISBN 978-985-897-160-1.

В монографии рассмотрены и проанализированы достижения авторов книги и других исследователей в двух направлениях нейрокриптографии: использование и сопоставительный анализ эффективности и безопасности нейросетевых архитектур в виде древовидных машин четности (TPM) на основе различных алгебр (действительных и комплексных чисел, кватернионов и октонионов) и применение нейронных сетей с хаотическими функциями отображения для операций над хеш-функциями.

Книга предназначена для специалистов, а также аспирантов, магистрантов и студентов, научные и прикладные интересы которых лежат в указанной предметной области.

УДК 004.032.26(07)
БКК 32.818.1я73

ISBN 978-985-897-160-1

ПРЕДИСЛОВИЕ

Различные виды тайнописи обнаружены во всех цивилизациях, от индейцев в 400 году до н. э. или древних греков до более известного шифра Цезаря. С тех пор криптология, объединяющая криптографию и криптоанализ, прошла долгий путь. В XXI веке, с наступлением эпохи киберпространства, кибератак, киберпреступлений, оказывающей все возрастающее влияние не только на информационные, сетевые ресурсы, но и на иные аспекты повседневной жизни людей, проблема защиты цифрового контента от несанкционированного доступа или использования из чисто технической трансформировалась в общегосударственную. Основные методы и средства решения этой проблемы связаны как раз с криптографией.

Примерно до конца 70-х годов прошлого века криптография, или шифрование информации, основывалась на использовании ключа, общего для двух участников обмена данными. Две основные проблемы, связанные с симметричной криптографией (хранение и транспортировка ключей, которые составляли тайну для третьей стороны), стимулировали поиск нового решения. И это решение было найдено У. Диффи и М. Хеллманом (W. Diffie, M. Hellman) в виде известного протокола Диффи – Хеллмана (ДХ), положившего начало криптосистемам с публичным ключом. Математическую основу таких систем составляет алгебраическая теория чисел. Но упомянутый протокол, при его использовании для согласования между двумя абонентами общего тайного ключа, также имеет слабые места, связанные с некоторыми видами атак.

В качестве альтернативы ДХ-протоколу И. Кантером (I. Kanter) и В. Кинцелем (W. Kinzel) в 2002 году предложено решать проблему генерации и передачи общего ключа с помощью двух взаимодействующих нейронных сетей (НС), названных древовидными машинами четности (TPM, Tree Parity Machine). Концепция построения и функционирования (обучения) таких и подобных НС основана на использовании моделей и методов статистической механики. Обучение означает, что синаптические веса двух TPM по определенным правилам адаптируются к парам вход-выход. После процедуры взаимного обучения сети формируют одинаковые наборы весовых коэффициентов, которые принимаются в качестве совместного тайного ключа. Синхронизация этих двух машин

аналогична синхронизации двух хаотических осцилляторов в хаотической связи. Однако, как оказалось, и здесь существует множественность подходов, решений и проблем. Все это рассматривается и анализируется в возрастающем потоке публикаций, составляющих один из сегментов нового направления, объединившего криптографию и нейронные сети, – *нейрокриптографию*.

Следует особо отметить, что доступ к основному содержанию большинства публикаций (не только по проблемам нейрокриптографии) для многих исследователей (особенно начинающих: студентов, магистрантов и аспирантов) носит ограниченный характер. Основная проблема – коммерциализация (в пользу издателей и издательств) и все более отчетливо проявляющаяся клановость (это вопросы для отдельного исследования).

Авторы настоящей монографии, трудовая деятельность которых связана с академической средой, имея определенный опыт в подготовке научных кадров и багаж знаний в анализируемой предметной области, решились на подготовку и издание книги с целью обобщения собственных, а также достаточно разрозненных материалов в доступных публикациях других исследователей. Многие результаты получены при выполнении совместных проектов сотрудниками Белорусского государственного технологического университета и Люблинского католического университета имени Иоанна Павла II (Польша) и затрагивают анализ достижений в двух направлениях нейрокриптографии: использование и сопоставительный анализ эффективности и безопасности архитектур TPM на основе различных алгебр и применение НС для операций над хеш-функциями.

Мы не стремились к полному и детальному анализу и обобщению всех известных методов и инструментальных средств. Мы также отдаем себе отчет в том, что, несмотря на все наши старания, помощь и советы уважаемых редакторов, рецензентов и коллег (за что мы им искренне признательны и благодарны), в тексте могут встречаться отдельные неточности. Все авторы в равной степени ответственны за содержание материала книги. Соавтором главы 2 является также доцент БГТУ Шутько Н. П. Заранее приносим уважаемым читателям свои извинения и будем благодарны за обратную информацию о выявленных ошибках.

С нашей точки зрения помощью в более глубоком изучении рассматриваемых в книге вопросов может стать обширная библиография, а также некоторые коды программ, которые приведены в приложениях и использовались авторами при проведении исследований.

1. ОСНОВНЫЕ ПРИНЦИПЫ ПОСТРОЕНИЯ И ФУНКЦИОНИРОВАНИЯ НЕЙРОННЫХ СЕТЕЙ

1.1. Искусственный интеллект и нейросетевые технологии

В своих изобретениях человек неоднократно применял идеи, заимствованные из законов и механизмов функционирования биологических существ. Используя свой интеллект, он адаптировал принципы, которые узнал, для решения проблем своей повседневной жизни. Понимание законов, регулирующих природу, часто приводило к попыткам применить эти правила на практике. Примеры включают использование подъемной силы для создания самолетов способом, подобным тому, который используют птицы, производство бумаги из дерева, как это делают осы, и принятие методов скрещивания и мутации в генетических алгоритмах, которые обрабатывают битовые строки как кодовые строки ДНК живых организмов.

Одна из наиболее интересных задач – воссоздание в машине или компьютерной программе элементов и механизмов разумного поведения человека. Идея создания такой машины была высказана профессором Стэнфордского университета Дж. Маккарти (J. McCarthy) [1] в 1956 году на конференции в Дартмутском университете. Описанный им Advice Taker должен был стать устройством, работу которого можно было бы определить без какого-либо языка программирования, используя только данные, описывающие среду, в которой он работает, и ожидаемые результаты этой операции. Такая машина должна была функционировать необычным, похожим на механизмы мыслительного процесса человека образом. Эта задача стала одной из целей зарождающейся области науки – *искусственного интеллекта* (ИИ; Artificial Intelligence, AI).

В русскоязычной литературе встречаются не менее 5–6 определений ИИ. С одной стороны, искусственный интеллект – это направление в информатике и информационных технологиях, с другой – это способность системы правильно интерпретировать внешние данные, извлекать сведения из анализа таких данных и

использовать эти сведения для достижения конкретных целей в решении задач при помощи гибкой адаптации.

Определение 1.1. ***Искусственным интеллектом*** будем называть область исследований, цель которых – создание технических систем, способных решать задачи невычислительного характера и выполнять действия, требующие переработки содержательной информации (что считается прерогативой человеческого мозга).

Общая сущность интеллекта характеризуется способностью в процессе мышления к генерации и выбору способа действий, адекватно отражающих решаемую проблему. Естественный интеллект возник и развивается в ходе биологической эволюции, адаптируясь к внешней среде. В разной мере он присущ биологическим формам жизни. ИИ характеризует способность к мышлению искусственных систем.

Само название было предложено Дж. Маккарти. Эта новая область заинтересовала многих исследователей. И их открытия продолжают находить новые и интересные приложения в различных областях, таких как робототехника, медицина и др., например автомобиль, управляемый искусственной нейронной сетью. Отметим также, что иногда понятие «искусственный интеллект» отождествляют с электронными устройствами, способными автоматически выбирать заложенный разработчиками режим функционирования. Слово «искусственный» при этом означает, что система не сможет найти новый режим работы в ситуации, не предусмотренной разработчиками.

В развитии ИИ можно выделить две тенденции: формальную и естественную.

Формальный подход – это операция, в которой новые методы конструируются на основе синтеза математики, логики, алгебры и т. д. Одним из таких методов может быть нечеткая логика. Разработанные подобным образом инструменты имеют тщательную теоретическую основу и могут быть относительно легко проанализированы с использованием классических методов, полученных из математики, таких как, например, математический анализ или исчисление вероятностей.

Естественный подход базируется на изучении механизмов, встречающихся в природе, и попытках воспроизвести их в машинах и информационных алгоритмах. Методы, основанные на этой тенденции, включают *искусственные нейронные сети* (ИНС либо

НС; Artificial Neural Networks, ANN), генетические и эволюционные алгоритмы и т. д. [2].

В 1950 году один из пионеров в области вычислительной техники английский ученый А. Тьюринг (A. Turing) пишет статью под названием «Может ли машина мыслить?», где описывает процедуру, с помощью которой можно будет определить момент, когда машина сравняется в плане разумности с человеком [3]. Впоследствии эту процедуру стали называть *тестом Тьюринга*.

В 60–80-х годах XX века приоритетным направлением исследований в области ИИ были экспертные системы. Они достаточно хорошо себя зарекомендовали, но только в узкоспециализированных областях. Для создания более универсальных интеллектуальных систем потребовался другой подход.

Вероятно, это обстоятельство привело к тому, что исследователи искусственного интеллекта обратили внимание на биологические НС, которые лежат в основе человеческого мозга.

У НС много важных свойств, но ключевое из них – это способность к обучению: способность ИИ самостоятельно получать знания в процессе работы тесно связана с проблематикой *машинного обучения* (Machine Learning, ML). Это направление стало центральным с самого начала развития ИИ.

Перечисленные факторы, их взаимосвязи и особенности, текущий уровень и тенденции развития информационных технологий привели к появлению новых направлений использования систем ИИ на основе НС – *нейросетевых технологий* (НСТ; Neural Network Technology) в следующих предметных областях:

– интернет вещей (Internet of Things, IoT) и промышленный интернет вещей (Industrial Internet of Things, IIoT);

– обработка естественного языка (Natural Language Processing, NLP);

– машинное зрение (Machine Vision, MV);

– глубинное обучение (Deep Learning, DL);

– распознавание текстов, речи, изображений (Character, Speech, Image Recognition);

– бизнес-аналитика (Business Analysis, BA);

– машинный перевод (Machine Translation, MT);

– интеллектуальные системы информационной безопасности (Smart Systems of Information Security);

– анализ медицинских исследований (Health-Service and Medical Analysis);

– прогнозирование временных рядов (Time Series Prediction);

– криптографические приложения (Cryptographic Applications);

– другие области.

Рассмотрение и анализ некоторых важных теоретических и прикладных аспектов использования НС в обеспечении информационной безопасности систем составляют содержание настоящей книги.

Для лучшего понимания основного материала книги (начинающими исследователями) мы посчитали целесообразным кратко изложить основные теоретические аспекты НСТ в следующих пунктах данного раздела.

1.2. Биологические и искусственные нейроны

Невозможно говорить об ИНС в отрыве от естественных структур, из которых состоит мозг. Именно он является основой для создания моделей ИНС. Поэтому наиболее общая характеристика ИНС – это модель человеческого мозга. Конечно, эта модель значительно упрощена, и ее размер покрывает только фрагмент реального мозга. Но она позволяет получать удивительные результаты.

Знакомство со структурой и особенностями функционирования мозга важно для построения соответствующих математических моделей и их реализации. Проведенные исследования позитронно-эмиссионной томографии (ПЭТ) указывают на стимуляцию определенных участков этого органа при выполнении соответствующих задач [4]. Это позволило создать так называемую функциональную карту коры (ответственности отдельных частей) головного мозга. Первым определением этих областей мы обязаны исследованию У. Пенфилда (W. Penfield) [5].

Вся нервная система человека представляет собой сложную структуру, которая имеет фундаментальное значение для жизни и развития человека, и может быть разделена на центральную нервную систему и периферическую нервную систему. Центральная нервная система включает головной и спинной мозг. Периферическ-

ская нервная система состоит из *нейронов*, которые связаны с мышцами и железами. Они контролируют работу гладких и поперечнополосатых мышц, кровообращение, движение, функции пищеварения и секрецию различных ферментов. Эти нейроны связываются с головным мозгом через спинной мозг, который отвечает за основные безусловные рефлексы.

Мозг состоит примерно из 10^{10} нейронов и 10^{12} клеток. Количество связей между клетками составляет примерно 10^{15}. Каждый нейрон посылает и принимает импульсы с частотой от 1 до 100 Гц. Длительность одиночного импульса – около 1,5 мс. Основываясь на приведенной выше информации, можно оценить, что скорость, с которой работает мозг, составляет 10^{18} операций в секунду. Дополнительно стоит отметить, что эти операции выполняются параллельно.

1.2.1. Клетка биологического нейрона

Основными элементами мозга являются нервные клетки, называемые *нейронами*. Каждый нейрон имеет тело клетки, дендриты и аксон (см. рис. 1.1). Тело клетки (сома) отвечает за выполнение основных жизненных процессов и обработку электрических импульсов, достигающих нейрона. Эти сигналы поступают в клетку через специальные выступы, называемые дендритами, и один нейрон может иметь множество дендритов.

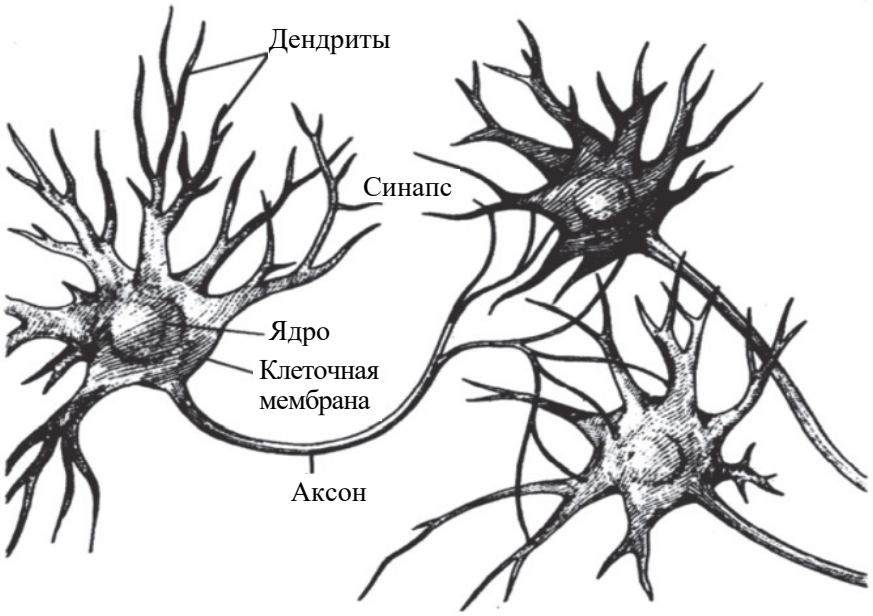

Рис. 1.1. Биологический нейрон [6]

Эта классическая модель была недавно уточнена на основе исследований мозга мышей [7], которые показали, что дендриты не просто пассивные передатчики нервных импульсов, но могут также модифицировать эти импульсы.

Нервные импульсы достигают клетки через дендриты. Выходом клетки является аксон, электрические импульсы через который передаются другим нейронам. Аксон имеет множество ответвлений, на конце каждого из которых находится область, называемая *синапсом*.

Посредством синапсов осуществляется связь между различными нейронами. Соединение аксона одной клетки с дендритом другой называется *синаптическим соединением*. Электрические импульсы передаются между нейронами с помощью катионов кальция.

На участках контакта между нейронами (синапсы) электрические импульсы превращаются в химические сигналы, которые стимулируют проникновение в клетку нейрона положительных зарядов. Когда достигается критическое значение потенциала, называемое *пороговым*, в ядре нейрона возникает электрический импульс, распространяемый как волна по аксону на следующий нейрон. Вклад одного синапса в установление соответствующего потенциала на выходе нейрона очень маленький. Для возникновения электрического импульса необходимо, чтобы нейрон непрерывно интегрировал множество синаптических входов [6].

Таким образом, в процессе психической деятельности в коре головного мозга распространяются нервные импульсы, которые активизируют соответствующие области нейронов. Совокупность нейронов и связей между ними образуют НС, от функционирования которой зависят эмоциональные реакции, сознательная деятельность и память человека.

Скорость распространения сигналов в нервных волокнах намного меньше, чем скорость распространения сигналов в электрических схемах. Однако параллельная обработка нейронами образной информации и организация взаимодействия между нейронами позволяют нейтрализовать этот недостаток.

1.2.2. Искусственный нейрон

Существующие в естественных структурах нервные клетки вдохновили на создание их эквивалентов в виде математических

моделей. Одной из самых популярных является модель, разработанная в 1943 году У. МакКаллоком (W. McCulloch) и У. Питтсом (W. Pitts) [8]. Это модель *искусственного нейрона*, выполняющего операцию нелинейного преобразования (задаваемого оператором f) скалярного произведения вектора $X = \{x_i\}$ входных сигналов x_i ($i = 1, 2, \ldots, n$; $n \in \mathbb{N}$) на вектор $W = \{w_i\}$ весовых коэффициентов w_i (синапсы), которые характеризуют силу *синаптической связи* по аналогии с биологическим нейроном:

$$y = f(XW) = f\left(\sum_{i=1}^{n} x_i w_i\right). \tag{1.1}$$

Как видим, математически нейрон представляет собой *взвешенный сумматор*. Оператор нелинейного преобразования f называют *функцией активации* нейронного элемента.

Длины входного и весового векторов вычисляются через их координаты:

$$|X| = \sqrt{\sum_{i=1}^{n}\left(x_i^2\right)},\ |W| = \sqrt{\sum_{i=1}^{n}\left(w_i^2\right)}. \tag{1.2}$$

Основными элементами (блоками) искусственного нейрона (рис. 1.2) являются *сумматор* ($\sum$) с выходным сигналом φ и *блок активации*, описываемый здесь *функцией активации* $f(\varphi)$ (ФА).

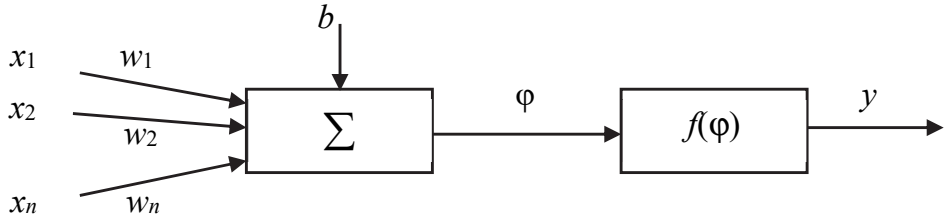

Рис. 1.2. Модель нейрона МакКаллока – Питтса

Функционал модели заключается в приеме в определенный момент времени t входных импульсов, а также некоторого смещения b, являющегося *уровнем (порогом) чувствительности* нейрона. В некоторых исследованиях роль смещения играет первый сигнал (x_0) с присвоенным ему фиксированным весом $w_0 = 1$ [2], а иногда это сигнал $x_0 = 1$ с переменным весом w_0 [9].

Отметим, что w_i – веса, соответствующие сигналам x_i, b – порог чувствительности нейрона (bias); x_i, w_i, $b \in \mathbb{N}$.

Определенный таким образом сигнал φ называется *постсинаптическим потенциалом* в момент времени t и вычисляется в соответствии с формулой

$$\varphi^{(t)} = \sum_{i=1}^{n} w_i^{(t)} x_i^{(t)} + b^{(t)}. \qquad (1.3)$$

Сигнал φ обрабатывается блоком активации, и определенная в нем функция f преобразует его в выходной сигнал y нейрона.

Как видим, в общем случае функция $y = f(\varphi)$ определяет зависимость сигнала на выходе нейрона от взвешенной суммы входных сигналов нейрона.

В некоторых исследованиях модель нейрона описывается с использованием теории графов. В частности, работа [9] содержит анализ такой модели на основе *переходных графов сигналов* (Signal-Flow Graph), определение которых сделаны С. Мэйсоном (S. Mason) в статье [10].

Определение 1.2. ***Переходный граф сигналов*** (ПГС) – это ориентированный граф, состоящий из определенных точек, называемых узлами (не вершинами, как в классических графах), которые соединены направленными ветвями (в отличие от классических ребер).

Типичный узел j связан с сигналом узла x_j. Направленная линия связи, начинающаяся в j и заканчивающаяся в k, имеет *передаточную функцию*, которая определяет взаимосвязь между сигналом x_j, исходящим из узла j, и сигналом y_k, поступающим в узел k.

С точки зрения использования графов для описания ИНС можно выделить два типа узловых соединений (см. рис. 1.3 [10]):

1) синаптические ветви, в которых передаточная функция соответствует умножению сигналов на соответствующие веса в нейронных сетях и имеет линейный характер (см. рис. 1.3, *а*);

2) ветви активации, в которых передаточная функция соответствует функции активации в нейронных сетях и, следовательно, в общем случае не обязательно является линейной (рис. 1.3, *б*).

Поток сигналов в любом ПГС определяется двумя основными правилами.

Правило 1. Сигнальное состояние узла определяется как сумма сигналов на его входе (рис. 1.3, *в*).

Правило 2. Сигнальное состояние узла передается каждому последующему узлу независимо от типа передаточной функции, связывающей эти узлы (рис. 1.3, *г*).

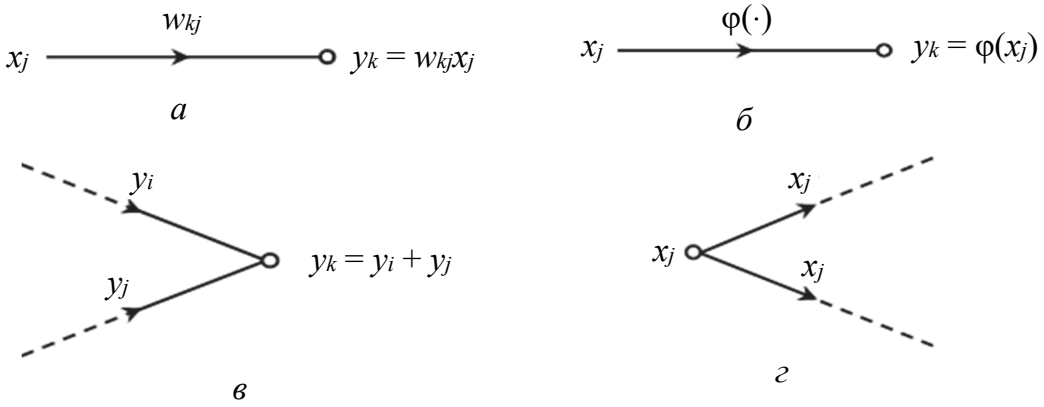

Рис. 1.3. Виды связей между узлами в ПГС

На рис. 1.4 показан одиночный искусственный нейрон с m входными сигналами (импульсами) и со смещением, выраженным сигналом $x_0 = 1$ с некоторым присвоенным ему весом b, на основе сформулированных понятий ПГС [10].

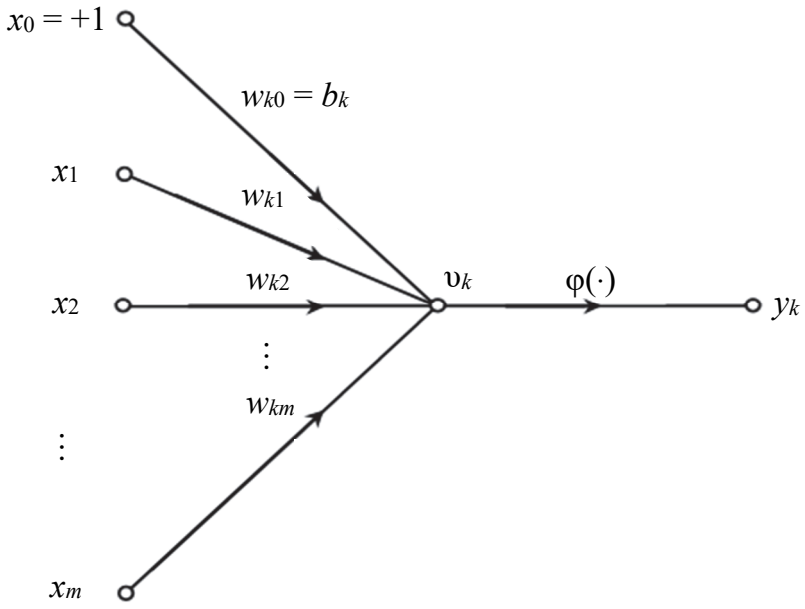

Рис. 1.4. Искусственный нейрон на основе ПГС

1.2.3. Функции активации нейрона

Функции активации (ФА), или операторы нелинейного преобразования, (φ) (см., например, [6, 11, 12] и др.) можно разделить в зависимости от их характера на *непрерывные* и *дискретные*.

Среди дискретных функций различают *униполярные* и *биполярные* – в зависимости от знака их значений. Существуют также похожие непрерывные функции (*сигмоид* и *тангенсоид*).

Простейшие ФА – это прерывистые, ступенчатые функции, которые принимают неотрицательные (униполярные) значения или могут принимать отрицательные (биполярные) значения. Функции этого типа используются в том числе в нейронных сетях для согласования криптографических ключей.

Примером дискретной ФА является униполярная пороговая функция, известная как функция Хевисайда. Она определяется следующей формулой:

$$f(\varphi)=\begin{cases}0, & \text{если } \varphi<0,\\ 1, & \text{если } \varphi\geq 0.\end{cases} \tag{1.4}$$

Данная функция имеет графическое изображение, показанное на рис. 1.5, *а*. Ее биполярный аналог – сигнум-функция (sgn(*x*) или sign(*x*)), определяемая формулой (1.5):

$$f(\varphi)=\begin{cases}-1, & \text{если } \varphi<0,\\ 0, & \text{если } \varphi=0,\\ 1, & \text{если } \varphi>0.\end{cases} \tag{1.5}$$

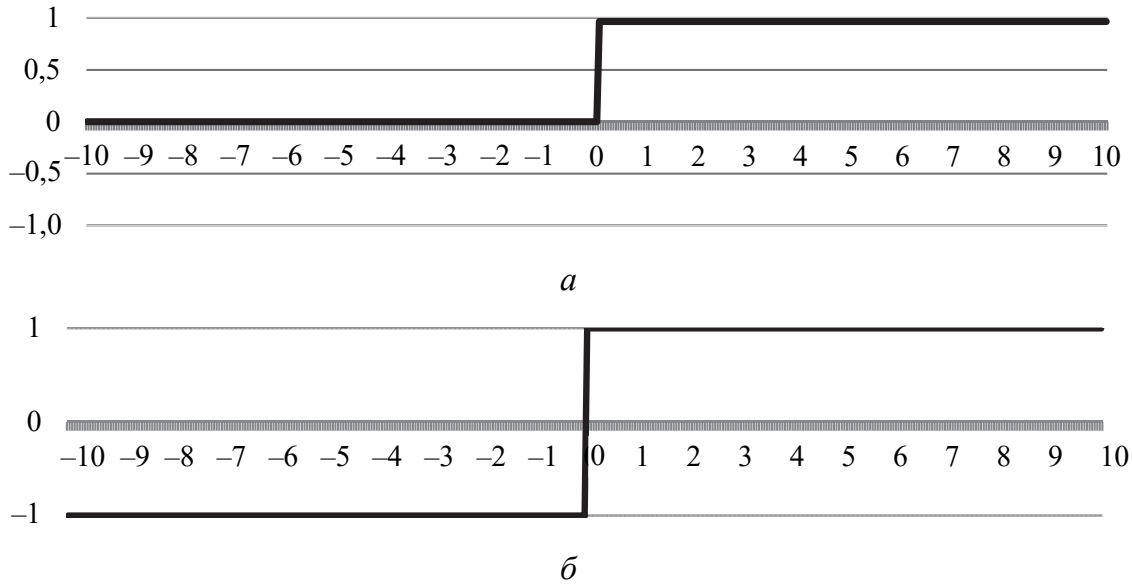

Рис. 1.5. Пороговые функции активации:
униполярная (*а*) и биполярная (*б*)

Пороговое значение (0) в обоих случаях (на рис. 1.5) обозначено нами в основном в связи с тем, что такое значение используется в ИНС, подверженных процессу синхронизации. В наиболее

общем случае пороговым значением является некоторе число k. Ступенчатые функции являются разрывными и поэтому недифференцируемыми.

1.3. Искусственные нейронные сети

Модель нейрона, разработанная У. МакКаллоком и У. Питтсом, положила начало развитию ИНС. В 1959 году Ф. Розенблатт (F. Rosenblatt) предложил модель нейронной сети, которую он назвал *персептроном* (иногда пишут «перцептрон»; Perceptron) [13]. Формальное графическое представление персептрона Ф. Розенблатта приведено на рис. 1.6.

Персептрон – это сеть, состоящая из нейронных элементов I, II и III уровня.

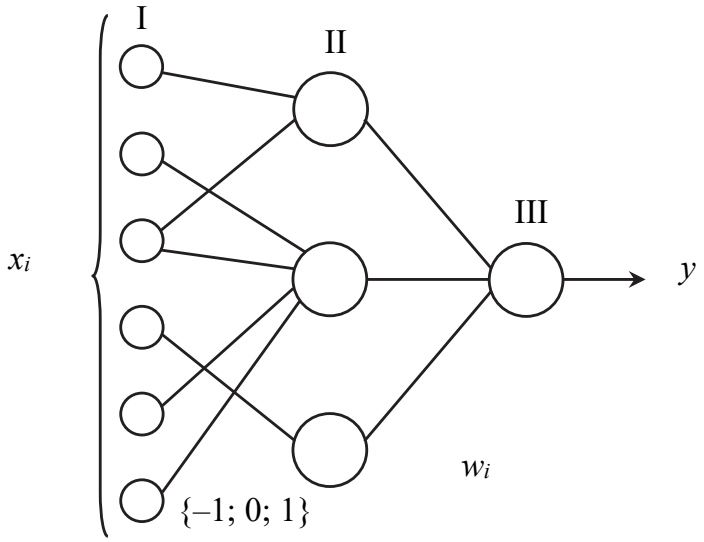

Рис. 1.6. Логическая схема элементарного персептрона

Элемент уровня II на рис. 1.6 активизируется при условии, что количество сигналов от элементов уровня I на его входе превысило некоторый порог. Элемент уровня III выполняет суммирующие функции примерно так, как это было рассмотрено выше. Все перечисленные элементы называются простыми, так как они реализуют скачкообразные (пороговые) функции. Ф. Розенблатт описал иные, отличные от представленной на рис. 1.6, структуры и ввел для них соответствующие определения. Нас далее будет интересовать

в основном приведенная на рис. 1.6 схема. Такую логическую схему называют *персептроном с одним скрытым слоем* (Perceptron with one Hidden Layer или Multilayer Perceptron, MLP): слой элементов уровня II расположен между слоями элементов уровней I и III.

Определение 1.3. ***Слой нейронной сети*** – это множество нейронных элементов, на которые в каждый такт времени параллельно поступает информация от других нейронных элементов сети.

Определение 1.4. ***Искусственная нейронная сеть*** – совокупность нейронов, связанных между собой и функционирующих по определенным правилам. Основные из этих правил позволяют характеризовать ИНС следующим образом [14]:

– как параллельную систему, поскольку в любой момент времени в активном состоянии могут находиться несколько процессов;

– как распределенную систему, поскольку каждый из процессов может независимо обрабатывать локальные данные;

– как адаптивную систему, наделенную свойствами самообучения и подстройки своих параметров при изменении профиля данных.

Основным типом ИНС является *многослойный персептрон* (MLP), представляющий собой структуру, состоящую из искусственных нейронов, расположенных слоями, без обратной связи.

Многослойный персептрон по Розеблатту – персептрон, у которого имеется более одного слоя *A*-элементов. Многослойный персептрон Д. Румельхарта (D. Rumelhart) является частным случаем персептрона Ф. Розенблатта. Отличия заключаются в применяемых алгоритмах обучения (см., например, [15, 16]).

Функционирование сети основано на передаче импульсов между последовательными слоями. Схема такой ИНС показана на рис. 1.7. Топология создаваемых сетей выбирается так, чтобы наилучшим образом выполнять поставленные перед ней конкретные задачи.

До начала 1980-х годов исследования в области ИИ можно охарактеризовать как «в спящем режиме». Развитие технологии производства интегральных микросхем привлекло внимание исследователей к схемам параллельной обработки, включая ИНС. Вехой стала работа Дж. Хопфилда (J. Hopfield) [17], который описал НС с обратными связями. Такие НС могут быть использованы для решения широкого спектра задач (упоминались выше).

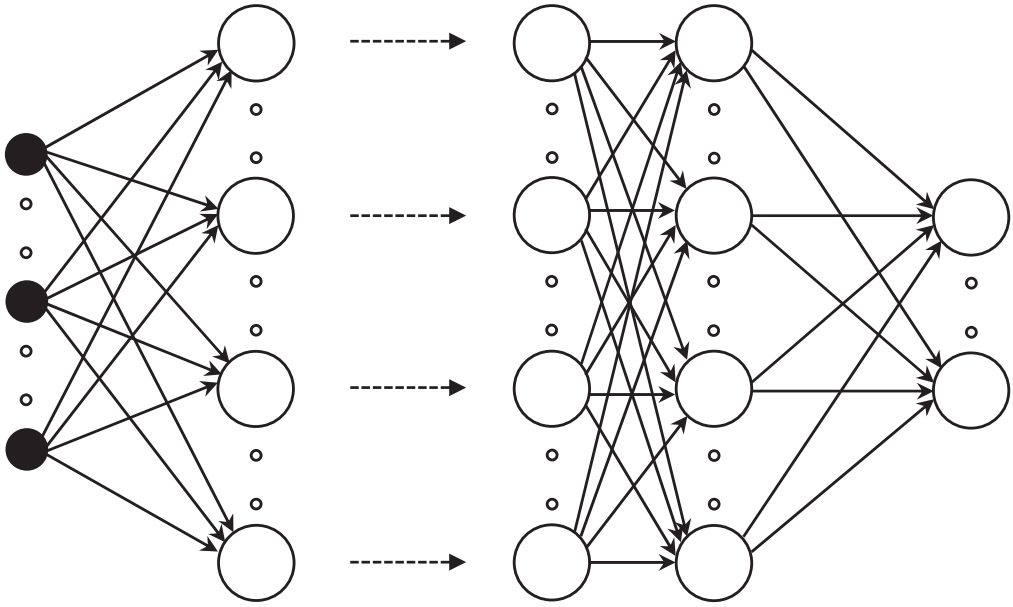

Рис. 1.7. Структура ИНС без обратных связей в общем виде

Количество слоев ИНС характеризует способность сети к разбиению входного пространства образов на подпространства меньшей размерности. Однослойный же персептрон с помощью *гиперплоскости* разбивает входное пространство образов на классы [6].

1.4. Обучение искусственных нейронных сетей

Способность к обучению является фундаментальным свойством мозга. В контексте НС обучение заключается в изменении «силы» синаптических связей между нейронами.

Определение 1.5. ***Обучение нейронной сети*** (Neural Network Training или Neural Network Learning) – процесс корректировки весовых коэффициентов связи таким образом, чтобы в результате при поступлении на вход сети определенного сигнала она выдавала нам правильный ответ.

Канадский нейропсихолог Д. Хэбб (D. Hebb) сформулировал [18] своеобразный нейрофизиологический постулат, лежащий в основе процедур обучения НС: «*Когда аксон одной клетки находится достаточно близко, чтобы возбудить другую клетку, и неоднократно или постоянно участвует в ее возбуждении (takes part in firing), в одной или обеих клетках происходит некоторый процесс роста (some growth process) или метаболические изменения, так*

что эффективность первой клетки как одной из клеток, возбуждающих вторую, возрастает». Основной смысл постулата Хэбба заключается в том, что если изначально наблюдается причинно-следственная связь между активациями пресинаптического и постсинаптического нейрона, то эта связь имеет тенденцию к усилению. Кроме того, согласно этому постулату для усиления связи между пре- и постсинаптическим нейронами необходимо совпадение их активности во времени.

Основополагающая процедура обучения персептрона, являющегося основой НС, описанная Ф. Розенблаттом [13], характеризуется тем, что весовые коэффициенты НС изменяются только в том случае, когда выходная реакция сети не совпадает с эталонной.

Как и в биологических системах, в зависимости от вида взаимодействия обучающегося объекта (ИНС) с внешней средой можно условно выделить *обучение с учителем* (Supervised Learning) и *обучение без учителя* (Unsupervised Learning).

В первом случае известно выходное пространство решений ИНС, а также предполагается, что имеются входные сигналы и эталонные реакции на них. В процессе обучения происходит целенаправленная модификация синаптических связей НС для достижения требуемого уровня соответствия (обычно – наилучшего из возможных) между реальными выходными значениями сети и их эталонными значениями. Во втором – ИНС формирует выходное пространство решений только на основе входных воздействий. Такие сети называют *самоорганизующимися* (Self Organizing Maps, SOM) или *самоорганизующимися картами Кохонена* [19].

Определение 1.6. ИНС с подстраиваемыми в процессе обучения весовыми коэффициентами называют ***сетями с динамическими связями***.

Известен метод обучения, называемый *обучением с критиком* [20] или *обучением с подкреплением* [21] (Reinforcement Learning, RL) [22]. *Системой подкрепления* Розенблатт называл любой набор правил, на основании которых можно изменять с течением времени матрицу взаимодействия внешней среды с персептроном [13]. По совокупности основных характеристик такие методы можно также отнести к обучению без учителя.

В настоящее время для обучения НС, в том числе глубоких, применяется алгоритм *обратного распространения ошибки* (Error Back Propagation Algorithm), основанный на методе *градиентного*

спуска (Gradient Descent) [23]. Алгоритм использует обучение с учителем. Для этого применяется обучающая выборка с заранее известными правильными ответами. Вводится мера ошибки, которая определяет, насколько сильно выходные значения сети отличаются от правильных ответов. Затем мера ошибки минимизируется с помощью метода градиентного спуска.

Методы обучения зависят также от типа функции активации нейрона. Для обучения персептрона с пороговой функцией активации могут использоваться: правило Хэбба, метод Уидроу – Хоффа (Widrow – Hoff), называемый также правилом персептрона [24–26], и др. Модификация весов на очередном шаге обучения $(t + 1)$ осуществляется по формуле

$$w_i^{(t+1)} = w_i^{(t)} + \Delta w_i^{(t)}, \qquad (1.6)$$

согласно которой каждый i-й весовой коэффициент (вес) изменяется на определенное значение $\Delta w_i^{(t)}$, называемое *приращением веса*. Точная форма этого компонента зависит от выбранного метода обучения сети и часто – от дополнительного фактора (обозначим его символом η), определяемого *параметром обучения*, который влияет на скорость и точность обучения.

1.4.1. Обучение искусственной нейронной сети с учителем

Цель рассматриваемого обучения состоит в том, чтобы в определенный момент времени (или на шаге t) для заданных значений входных импульсов $x_1, x_2, \ldots, x_n$ и известного ожидаемого значения $d^{(t)}$ выходного сигнала этот выходной сигнал $y^{(t)}$ был равен $d^{(t)}$. Такое состояние достигается благодаря процессу обучения сети, т. е. соответствующему выбору весов и смещения посредством соответствующего процесса оптимизации, а в некоторых сетях – также и входных сигналов [2].

Определение 1.7. В варианте обучения с учителем используется набор, содержащий шаблоны предполагаемого функционирования обученной сети. Данные x_i в виде входных импульсов и соответствующих ожидаемых значений называются ***обучающей выборкой***.

На каждом шаге обучения, таким образом, выступает «*обучающая пара*»: входной вектор $X^{(t)}$ и соответствующее ему ожидаемое значение выходного сигнала $d^{(t)}$. Зная образцовый сигнал $d^{(t)}$ и

полученную выходную величину $y^{(t)}$, мы можем подсчитать погрешность в виде *среднеквадратической ошибки* ε:

$$\varepsilon = \left(y^{(t)} - d^{(t)} \right)^2. \tag{1.7}$$

Цель процесса обучения – минимизация среднеквадратической погрешности, которая реализуется посредством модификации каждого вектора весов. Процесс обучения заканчивается, когда величина ошибки не превышает некоторое принятое допустимое значение.

Если обучающее множество имеет больше чем одну обучающую пару, то полная ошибка – это сумма ошибок всех p пар:

$$Q = \sum_{m=1}^{p} \varepsilon_m . \tag{1.8}$$

При определении среднеквадратической ошибки сети необходимо выполнить дополнительное суммирование ошибок по числу нейронов в сети (или в соответствующем слое сети).

Цель процесса обучения – минимизация среднеквадратической погрешности, которая реализуется посредством модификации каждого вектора весов. Процесс обучения заканчивается, когда величина ошибки не превышает какое-то принятое допустимое значение.

Если предположения относительно структуры сети и случайных входных векторов выполняются, обучение сети может быть записано с помощью соответствующих дифференциальных уравнений [23].

Перейдем теперь к рассмотрению сущности процесса обучения, т. е. модификации векторов весов каждого из нейронов. Изменение весовых коэффициентов должно привести к уменьшению ошибки, допускаемой всей сетью. Для определения направления изменения вектора весов используется метод градиентного спуска. Значение j-го внутреннего веса i-го нейрона на $(t + 1)$-м шаге зависит от данного значения на t-м шаге, а также определяется значением градиента функции ошибки:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \mu \left(-\frac{\partial Q(t)}{\partial w_{ij}^{(t)}} \right). \tag{1.9}$$

В приведенном выражении параметр $Q(t)$ определяется в соответствии с выражениями (1.7) и (1.8) на t-м шаге. Коэффициент μ показывает величину изменений, происходящих на каждом шаге алгоритма обучения.

На практике процесс обучения начинается с инициализации случайных величин весовых коэффициентов. Затем на вход сети подается входной сигнал X, на основании которого вычисляются величины выходного вектора сети и ошибки. Зная значение ошибки сети, вычисляется *градиент функции ошибок* и изменяются величины весовых коэффициентов в соответствии с последним выражением. Этот алгоритм повторяется вплоть до момента, когда величина ошибки не будет превышать какое-то принятое допустимое значение.

Определение 1.8. Один проход в процессе обучения НС по всей обучающей выборке называется ***эпохой обучения***.

Методика обучения ИНС с учителем требует определения механизма проверки правильности работы обучаемой сети. Классически для этой цели используется обучающий набор, элементы которого представляют собой пары, состоящие из входного вектора для сети и ожидаемого результата для всей сети. Этот набор создается перед началом обучения сети и должен храниться, например, в памяти компьютера на протяжении всего обучения. На практике ИНС учатся классифицировать входные импульсы (векторы входных сообщений) в одну из нескольких групп. Обучающий набор – это набор типовых классификаций, которые можно использовать в качестве эталона при обучении сети и для оценки ее производительности.

Пусть обучающая выборка содержит k ($k \in N$) элементов. Обучение сети проводится по следующей схеме:

1) установить индекс $i = 1$;

2) определить i-й входной вектор (X_i) и i-й ожидаемый результат (d_i; либо вектор, D_i);

определить результат работы сети для этого вектора X_i;

сравнить текущий результат с ожидаемым: если эти значения отличаются, запомнить ошибку работы сети;

3) использовать выбранный алгоритм обучения для изменения вектора весов (W_i) сети;

4) проверить условие окончания обучения и, если оно соблюдается, завершить весь процесс.

Если в обучающей выборке все еще есть неиспользованные элементы, увеличить i и вернуться к шагу 2.

После прохождения всей выборки установить ошибку равной нулю и вернуться к шагу 1, чтобы повторно запустить процесс обучения.

Условие завершения процесса обучения может быть заранее определено. Это так называемое *априорное условие*. Например, обучение может длиться определенное заданное количество циклов независимо от степени эффективности сети. Условие окончания обучения также может принимать другую форму и может быть связано с ходом процесса обучения. Тогда это называется *апостериорным условием*. Например, оно может быть построено на основе отношения значений вектора весов на последовательных шагах или на основе вычисленной сетевой ошибки. Схематично концепция такого алгоритма обучения ИНС представлена на рис. 1.8.

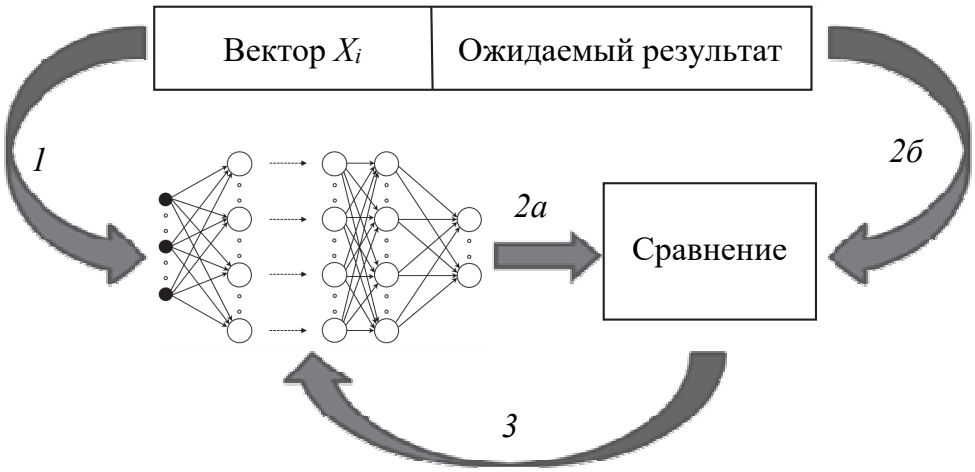

Рис. 1.8. Схематическое представление концепции обучения с учителем

На этом рисунке использованы следующие обозначения процессов: *1* – определение i-го входного вектора (X_i) на шаге 2 вышеприведенного алгоритма; *2а* – определение результата работы сети для этого вектора (X_i) на том же шаге; *2б* – сравнение текущего результата с ожидаемым (на шаге 2); *3* – шаг 3 алгоритма.

1.4.2. Обучение без учителя. Правила Хэбба и анти-Хэбба

Самым важным свойством обучения с учителем является необходимость наличия «стороннего наблюдателя», имеющего определенное множество правильных ответов. Естественно, благодаря этому обучение проходит в заданном направлении. Однако

не всегда известно множество правильных выходных ответов. Такая ситуация может иметь место в биологическом эквиваленте нейронной сети или в мозгу. Именно он часто учится без помощи учителя, распознавая мир непосредственно с помощью чувственных ощущений.

Сеть, обучающаяся без учителя, лишена априорных знаний. Она распознает (без посторонней помощи) свойства, образцы и взаимосвязи, а затем передает эту информацию в закодированном виде на выход. Такого типа обучение возможно только тогда, когда мы имеем дело с избыточностью входных данных. Без этого было бы невозможно отделение образцов из обучающего множества и распознавание их свойств. Вдобавок в обучающем множестве должна существовать возможность деления этого множества на непересекающиеся классы по хорошо определенным свойствам. Сеть, обученная методом без учителя, должна быть способна идентифицировать вышеупомянутые свойства в любом подаваемом на вход наборе параметров.

Первый алгоритм, обучающий нейронные сети такого типа, был разработан Д. Хэббом на основе упомянутых выше биологических исследований [18] и доработан Р. С. Саттоном (R. S. Sutton, см., например, [27]). Он известен как «правило Хэбба» (Hebbian rule), или просто «обучение без учителя». В настоящее время это правило является одним из наиболее часто используемых в обучении искусственных нейронов.

Как отмечалось, Д. Хэбб использовал наблюдение того, что веса межнейронных соединений при активации нейронов могут возрастать. В соответствии с этим рассматриваемое правило основано на том, что вес нейронной связи увеличивается, когда оба нейрона активны, и уменьшается в остальных случаях. Значение весов на шаге $t + 1$ вычисляется по формуле (1.6), где прирост веса определяется следующим образом:

$$\Delta w_i^{(t)} = F\left(x_i^{(t)}, y^{(t)}\right). \tag{1.10}$$

Функция F зависит от входного сигнала $x_i^{(t)}$ – *пресинаптического*, и выходного ($y^{(t)}$) – *постсинаптического*. В классическом варианте правила Хэбба функция F имеет вид [2, 11, 20, 29]:

$$F\left(x_i^{(t)}, y^{(t)}\right) = \eta x_i^{(t)} y^{(t)}, \tag{1.11}$$

где η – коэффициент скорости обучения.

Правило Хэбба может быть использовано в варианте сетевого обучения с учителем и без него. В первом случае используется искомое значение $d^{(t)}$ (здесь обычно параметр обучения $\eta = 1$), а во втором – выход нейрона $y^{(t)}$. Следовательно, функция F в выражении (1.10) в случае обучения с учителем определяется по формуле

$$F\left(x_i^{(t)}, y^{(t)}\right) = \eta x_i^{(t)} d^{(t)}. \qquad (1.12)$$

В табл. 1.1 приведен пример использования правила Хэбба для решения задачи по реализации операций на основе логической функции ИЛИ (OR) (и на тех же условиях: $\eta = 1$).

Таблица 1.1

Пошаговое обучение искусственного нейрона по правилу Хэбба для реализации операций на основе логической функции ИЛИ

t	x_1	x_2	d	w_1	w_2	F	y	ε
1	0	0	0	–1	1	0	0	0
2	0	1	1	–1	1	1	1	0
3	1	0	1	–1	2	–1	0	1
4	1	1	1	–1	2	1	1	0
5	0	0	0	0	3	0	0	0
6	0	1	1	0	3	3	1	0
7	1	0	1	0	4	0	0	1
8	1	1	1	0	4	4	1	0
9	0	0	0	1	5	0	0	0
10	0	1	1	1	5	5	1	0
11	1	0	1	1	6	1	1	0
12	1	1	1	2	6	8	1	0

Одним из недостатков классического правила Хэбба является то, что значения веса экспоненциально увеличиваются, когда один и тот же входной сигнал используется несколько раз. Как видно из табл. 1.1, значения весов обученного нейрона продолжают увеличиваться. Чтобы избежать этой проблемы, в классическое правило Хэбба вносятся различные модификации. Одна из них – добавление в формулу так называемого *веса коэффициента забывания* (Forgetting Factor) γ [2, 11, 28, 29]:

$$F\left(x_i^{(t)}, y^{(t)}\right) = \eta x_i^{(t)} d^{(t)} - \gamma w_i^{(t)} d^{(t)}. \qquad (1.13)$$

Значение γ выбирается, как правило, из интервала [0, 1] и часто составляет некоторый процент скорости обучения η. Большие

значения γ приведут к забыванию большей части того, чему нейрон обучался на ранних выборках. Рекомендуемые значения: γ < 0,1, которые позволяют сохранить опыт обучения ИНС на ранних выборках и стабилизировать значения весов на определенном уровне.

Для сигнала $x_i < \frac{\gamma}{\eta} w_i$ значение функции $F(x_i^{(t)}, y^{(t)})$, вычисленной в соответствии с формулой (1.11), может быть отрицательным, поэтому важно правильно выбрать коэффициент забывания [2, 11, 29].

Процесс обучения методом Хэбба без учителя заключается в следующем. Элементы вектора весовых коэффициентов $W = \{w_i\}$ на каждом шаге $t + 1$ равны сумме значения элемента вектора весовых коэффициентов на шаге t и произведения элементов входного сигнала $x_i^{(t)}$ на значения выходного сигнала $y^{(t)}$ обучаемого нейрона. Математически это можно записать с помощью следующего уравнения [6, 11, 12, 18]:

$$w_i^{(t+1)} = w_i^{(t)} + cx_i^{(t)} y^{(t)}, \qquad (1.14)$$

где c – это положительная константа, управляющая процессом обучения.

Анализируя вышеизложенное, мы видим, что модификации подвергаются те весовые коэффициенты, которые более активны в ситуациях, когда их нейрон возбужден. Следовательно, сеть, обученная методом Хэбба, *автоассоциативна*. Такого типа сеть (называемая также сетью Хопфилда [17]) учится распознавать образцы, последовательно группируя их в классы, содержащие элементы с похожими свойствами. Итак, один нейрон, обученный распознавать некоторый образец *Z*, будет также способен к распознаванию образцов, похожих на *Z* (принадлежащих этому же классу).

Представленный метод Хэбба называется также *корреляционным обучением* (Correlation Learning), так как он ведет к такому приспособлению весов, которые позволяют получить самую лучшую корреляцию между входным сигналом и вектором весов (того нейрона, который был больше всех возбужден в отношении вектора *X*). Благодаря этому мы имеем некоторое обобщение опыта, полученного сетью в процессе обучения, в результате чего сеть способна распознавать объекты, которых никогда прежде «не видела».

Нейронная сеть, обученная правилом Хэбба, достигает, в общем, довольно хороших успехов и полностью автоматически группирует входные сигналы в классы, похожие на образцы. Конечный результат, однако, не всегда получается положительным, так как он зависит от начального состояния сети (начальных величин вектора весов, выбираемых случайным способом). Существует небольшая вероятность, что разные классы входных образцов «найдут свои нейроны». Поэтому нужно также считаться с тем, что несколько нейронов могут указывать на сам класс. Вдобавок невозможно определить априори, какой нейрон научится распознавать данный класс. Следовательно, количество нейронов этого типа нейронной сети должно быть больше, чем ожидаемое число различных классов.

Правило Хэбба имеет еще один недостаток. Оно ведет к расхождению процесса, так как значения весовых коэффициентов нейронов увеличиваются в процессе обучения. Можно предотвратить эту ситуацию путем изменения правила, заключающегося в ограничении увеличения величины вектора весов. Хорошее решение этого вопроса предложил Э. Оя (E. Oja) [29]:

$$w_i^{(t+1)} = w_i^{(t)} + cy^{(t)}\left(x_i^{(t)} - w_i^{(t)}y^{(t)}\right). \qquad (1.15)$$

Противоположный правилу Хэбба вариант сетевого обучения – это анти-Хэббовское обучение, или *правило анти-Хэбба* (Anti-Hebbian Learning), при котором важность (вес) связей между нейронами возрастает, когда один возбужден, а другой «отдыхает». Или иными словами: правило обучения по Хэббу определяется как коррелированная активация пре- и постсинаптических нейронов, приводящая к усилению связи между двумя нейронами, в то время как в правиле анти-Хэбба упрощенно упомянутые связи следует понимать как предписывающее снижение силы синаптических связей между нейронами. Последнее означает также, что вес соединения между двумя нейронами (устройствами) уменьшается всякий раз, когда возникает ситуация, в которой один нейрон активен, а другой – нет. Это также означает, что алгоритм анти-Хэбба может создавать тормозящие связи, уменьшая их веса ниже нуля.

Знак функции F для алгоритма анти-Хэбба меняется:

$$F\left(x_i^{(t)}, y^{(t)}\right) = -\eta x_i^{(t)} d^{(t)}. \qquad (1.16)$$

Это правило никогда не приводит к неограниченному увеличению веса и обеспечивает стабильность без дополнительных модификаций. Представленные методы обучения НС с учителем применимы только к одному нейрону, и для него можно напрямую определить ожидаемый результат.

Функция (1.16) изначально применялась авторами протокола согласования криптографических ключей с использованием ИНС [30–35]. Недостатком использования метода обучения анти-Хэбба для синхронизации НС является то, что сети достигают состояния синхронизации с противоположными значениями веса [36].

В сетях, состоящих из большего количества слоев, необходимо применять более сложные алгоритмы для обучения (весов) нейронов из скрытых слоев, для которых не указаны ожидаемые результаты работы. Одним из таких методов является *обратное распространение ошибок* (Backpropagation или Backpropagation Through Structure, BPTS) [9, 11], основанное на дифференцировании сложных функций. Это связано с тем, что, зная ожидаемый результат работы всей сети, невозможно напрямую определить, какие результаты должны возвращать нейроны скрытого слоя. Метод BPTS является модификацией классического метода градиентного спуска.

1.4.3. Правило персептрона

Самая простая модель однонаправленной сети – единичный персептрон. Основной его элемент – вектор весов. Обучение персептрона (Perceptron Learning Rule) является базовой операцией для обучения сети с пороговой функцией активации в варианте с учителем. Кроме того, это частный случай метода Уидроу – Хоффа [20, 26] для двоичных выходных значений.

Выходной сигнал линейной части персептрона (φ на рис. 1.2) вычисляется в соответствии с выражением (1.3). Выход сети (персептрона) определяется по известной формуле:

$$y = \text{sign}(\varphi). \qquad (1.17)$$

Функция знака sign разделяет пространство входных векторов X пополам. Линия деления определяется уравнением $WX = 0$. Изменение положения линии деления (процесс обучения) заключается в подаче на вход персептрона соответствующих входных величин, а также связанных с ними определенных выходных значений,

к которым сеть должна стремиться. Следовательно, обучающий вектор мы можем представить в виде уже рассмотренного пространства пар: y и d.

Персептрон может возвращать значение, равное, меньшее или большее по отношению к ожидаемому значению. Каждый из этих трех случаев приводит к разному изменению веса обучаемой сети согласно анализируемому правилу. Соответствующее приращение веса (см. формулу (1.6)) определяется следующим образом:

$$\Delta w_i^{(t)} = \begin{cases} -x_i^{(t)}, \text{ если } y^{(t)} > d^{(t)}, \\ 0, \text{ если } y^{(t)} = d^{(t)}, \\ x_i^{(t)}, \text{ если } y^{(t)} < d^{(t)}. \end{cases} \qquad (1.18)$$

При изменении смещения следует также заменить на единицы значения входных импульсов.

Пошаговый алгоритм, реализующий рассматриваемый метод, может быть представлен следующим образом.

Шаг 1. На основе произвольно выбранных начальных весов $w_{1,1}$, $w_{2,1}$, …, $w_{n,1}$ на первом шаге вычисляются результирующий сигнал сумматора φ_1 и значение выходного сигнала нейрона $f_1(\varphi_1)$ (или y_1, см. рис. 1.2).

Шаг 2. На этом шаге значение выходного сигнала нейрона $f_1(\varphi_1)$ сравнивается с ожидаемым результатом работы сети d_1 на первом шаге, и на основе результата этого сравнения модифицируются весовые коэффициенты и смещение в соответствии с формулами (1.8) и (1.9):

а) если $f_1(\varphi_1) = d_1$, то значения весовых коэффициентов и смещение остаются без изменений: $w_{i,2} = w_{i,1}$ ($i = 1, 2, \ldots, n$), $b_2 = b_1$;

б) если $f_1(\varphi_1) > d_1$, веса уменьшаются на величины соответствующих им входных сигналов: $w_{i,2} = w_{i,1} - x_{i,1}$ ($i = 1, 2, \ldots, n$), и смещение уменьшается на единицу: $b_2 = b_1 - 1$;

в) если же $f_1(\varphi_1) < d_1$, то соответствующие преобразования примут формальный вид: $w_{i,2} = w_{i,1} + x_{i,1}$ ($i = 1, 2, \ldots, n$) и $b_2 = b_1 + 1$.

Шаг (k + 1)-й. На этом шаге значение выходного сигнала нейрона $f_k(\varphi_k)$ сравнивается с ожидаемым результатом работы сети d_k на k-м шаге:

а) если $f_1(\varphi_1) = d_k$, то значения весовых коэффициентов и смещение остаются без изменений: $w_{i,k+1} = w_{i,k}$ ($i = 1, 2, \ldots, n$), $b_{k+1} = b_k$;

б) если $f_1(\varphi_1) > d_k$, веса уменьшаются на величины соответствующих им входных сигналов: $w_{i,k+1} = w_{i,k} - x_{i,k}$ ($i = 1, 2, \ldots, n$), и смещение уменьшается на единицу: $b_{k+1} = b_k - 1$;

в) если же $f_1(\varphi_1) < d_k$, то соответствующие преобразования примут формальный вид: $w_{i,k+1} = w_{i,k} + x_{i,k}$ ($i = 1, 2, \ldots, n$) и $b_{k+1} = b_k + 1$.

Далее операции повторяются на основе тех же принципов.

Пример применения рассмотренного метода обучения для отдельного искусственного нейрона, обучаемого для реализации операций на основе логической функции ИЛИ (OR), представлен в табл. 1.2. Параметр обучения η был установлен на 1. Кроме того, нейрон не имеет порога чувствительности (смещения). В первом столбце таблицы указан номер t шага обучения сети. Далее следуют столбцы, взятые из обучающей выборки: входные импульсы и ожидаемые результаты работы сети. Столбцы, обозначенные w_1 и w_2, содержат значения веса нейрона на каждом шаге обучения, столбцы, обозначенные f и y, – соответственно функция активации нейрона и результат его работы. Последний столбец содержит *квадратичную ошибку* работы нейрона [6].

Таблица 1.2

Пошаговое обучение искусственного нейрона, обучаемого для реализации операций на основе логической функции ИЛИ по правилу персептрона

t	x_1	x_2	d	w_1	w_2	f	y	ε
1	0	0	0	–2	1	0	0	0
2	0	1	1	–2	1	1	1	0
3	1	0	1	–2	1	–2	0	1
4	1	1	1	–1	1	0	0	1
5	0	0	0	0	2	0	0	0
6	0	1	1	0	2	2	1	0
7	1	0	1	0	2	0	0	1
8	1	1	1	1	2	3	1	0
9	0	0	0	1	2	0	0	**0**
10	0	1	1	1	2	2	1	**0**
11	1	0	1	1	2	1	1	**0**
12	1	1	1	1	2	3	1	**0**

Как видим, обучение одного нейрона с помощью простой линейной логической функции происходит очень быстро. Понятно,

что время, необходимое для определения требуемых значений весов, зависит от начальных значений весов и параметра обучения η.

Нейрон в нашем примере полностью обучен, поскольку он возвращает ожидаемые результаты для каждого элемента обучающей выборки. В табл. 1.2 это видно из последовательности четырех нулей в столбце ошибок. В случае нелинейных функций, например XOR, один нейрон не может выполнить правильную классификацию. Для решения подобных задач строятся многослойные сети. Обучение такой сети, т. е. определение требуемых значений весов, подразумевает соответственно применение иных методов обучения: например, на основе *обратного распространения ошибок*, что в свою очередь требует дифференцируемой функции активации, например *сигмоидной* или *тангенсоидной* [6, 11].

1.5. Взаимодействие нейронных сетей

Система взаимодействующих друг с другом НС – это множество НС, в основе строения которых чаще всего лежит персептрон. Такая система сети также получает на вход вектор входных значений, на основе которого вычисляет значения выхода. Выходное значение – это основная информация для взаимного обучения НС.

Определение 1.9. Взаимное обучение НС, основанное на том, что сети «учатся» друга у друга, используя полученные от другой сети выходные сигналы и изменяя определенным образом собственные векторы весов, называется ***синхронизацией сетей*** или ***синхронизацией векторов весов обеих сетей***.

Абстрагируясь от основного объекта нашего исследования, отметим, что процесс и состояние синхронизации свойственны и биологическим, и физическим системам. Классическим примером, относящимся к последним, является синхронизация двух колебательных систем в электронике (генераторов или осцилляторов) на основе их фазового взаимодействия. При достижении состояния синхронизации динамические параметры синхронизированных систем становятся одинаковыми. Хаотические же системы могут синхронизироваться с помощью некоторого дополнительного источника или на основе взаимодействия между собой [37]. В большинстве ситуаций не имеет значения, является ли такое взаимодействие однонаправленным или двунаправленным. Таким обра-

зом, обычно нет разницы между компонентами, которые активно влияют друг на друга, и теми, на которые пассивно влияет динамика других систем. В работе [38] описаны некоторые общие принципы интерактивного взаимодействия НС. Очевидной разницы между однонаправленным и двунаправленным взаимодействием сетей в случае простых персептронов нет [31, 32].

В контексте дальнейшего анализа особенностей синхронизации ИНС для нас интерес представляют модели «*учитель – ученик»* (каждая из двух сетей выполняет соответствующую роль) и *взаимообучающие* сети. Кратко охарактеризуем их.

1.5.1. Модель «учитель – ученик»

В этой модели взаимодействие двух сетей похоже на обучение сети с учителем, которое описано в п. 1.4.1.

Как было подчеркнуто, самая простая модель однонаправленной НС – это единичный персептрон. Его основной элемент – это вектор весов $W = \{w_i\}$, $i = 1, 2, \ldots, N$; $N \in \mathbb{N}$. Выходной сигнал линейной части нейрона (сумматора) вычисляется в соответствии с формулой (1.1):

$$\varphi = \sum_{i=1}^{N} w_i x_i \,. \tag{1.19}$$

Выход сети (персептрона) вычисляется по приведенной выше формуле (1.17).

Обучающий вектор мы можем представить в виде известного нам пространства следующих пар:

$$\left(x_i^{(t)}, d_i^{(t)}\right), i = 1, \ldots, n. \tag{1.20}$$

Формально алгоритм обучения в наиболее общем виде можно представить как поиск функции g: $X \rightarrow Y$, где $X(x_i^{(t)})$ – пространство входов модели, $Y(y_i^{(t)})$ – пространство выходов. Функция g является элементом пространства функций G, которое называют также *пространством гипотез*.

В принятых нами ранее терминах и обозначениях обучение сети на основе модели «учитель – ученик» может быть представлено следующей общей формулой:

$$w^{(t+1)} = w^{(t)} + \eta x^{(t)} \left(d^{(t)} - y^{(t)}\right). \tag{1.21}$$

На основании описанных выше элементов персептрона рассмотрим систему из двух персептронов, где один исполняет роль учителя, а второй – ученика (сети *A* и *B*). Идея этой модели заключается в том, что вектор весов «учителя» остается постоянным и не меняется в процессе обучения. Вектор весов сети «ученика» изменяется на каждом шаге обучения или по требованию каждой обучающей пары. Отсюда вытекает, что процесс модификации вектора весов «ученика» в существенной мере зависит от времени. По завершении процесса изменения вектора весов «ученика» он располагается определенным образом в *n*-мерном пространстве. Процесс изменения вектора весов «ученика» должен быть направлен на соответствие вектору весов «учителя».

Расстояние между векторами весов обеих сетей (W^A и W^B) измеряется следующим образом [5, 20, 29]:

$$R = \frac{W^A W^B}{\left|W^A\right|\left|W^B\right|}, \tag{1.22}$$

где R – значение угла α между векторами весов сетей: W^A и W^B, говоря точно, $R = \cos\alpha$; $|W|$ – длина вектора.

Используя понятие *меры расхождения*, можно определить *ошибку обучения*:

$$\varepsilon = \frac{\arccos R}{\pi}. \tag{1.23}$$

Величина ошибки – это число из диапазона [0, 1], следовательно, ее можно интерпретировать как вероятность того, что сеть «ученика» для данного входного сигнала сгенерирует выходной сигнал, отличающийся от значения, указанного сетью «учителя».

Динамика изменений расхождения может быть подсчитана аналитическим способом следующим уравнением [29, 39, 40]:

$$\frac{dR}{d\alpha} = -\frac{R}{2\pi}\arccos R + \frac{1}{\pi}\left(1 - \frac{R^2}{2}\right)\sqrt{1 - R^2}\ . \tag{1.24}$$

Величина функции R уменьшается с увеличением значения α, а также в процессе обучения. Для больших значений α величина ошибки приближается к нулю. Это означает, что сеть «ученика» становится идентичной сети «учителя».

Уменьшение значения R, очевидно, влияет на величину ошибки ε. Скорость уменьшения в существенной мере зависит от выбранного правила обучения.

В процессе обучения расхождение R уменьшается и в конечном итоге, когда ученик получит «знание», содержащееся в весах «учителя», R достигнет величины 0.

1.5.2. Взаимное обучение нейронных сетей

Взаимное обучение касается простейшей однонаправленной модели, в которой участвуют два персептрона. Роли таких НС заранее не определены, каждая может выполнять функции как учителя, так и ученика (в зависимости от этапа обучения). Это значит, что сети учатся друг у друга, используя для этого полученные результаты и стремясь к общей цели, находят общие элементы в результате своих вычислений.

Далее будет показано, что эта модель обучения, благодаря своим свойствам, может быть использована в криптографии, а именно в определении совместного для двух сторон (двух сетей: A и B) криптографического ключа (аналогично протоколу Диффи – Хеллмана (W. Diffie, M. Hellman) [41]).

Пусть даны два персептрона, каждый из которых получает на вход случайно выбранный вектор входных значений: X^A и X^B. Оба персептрона преобразовывают свои внутренние векторы весов (W^A и W^B) в соответствии с принятым правилом обучения, также учитывая собственные полученные выходные величины y. Вычисляется эта величина следующим образом (см. соотношения (1.17), (1.19)):

$$y = \operatorname{sign}(WX). \tag{1.25}$$

В вышеуказанной формуле X – это нормированный N-мерный входной вектор: $WW = 1$.

Начальное состояние векторов весов W^A и W^B в обеих сетях – случайное. Затем на каждом шаге обучения на вход обеих сетей подается один (случайно сгенерированный) входной вектор X, в соответствии с которым сети вычисляют сигналы на выходе (y^A и y^B). На каждом шаге обучения весовые коэффициенты подвергаются следующему преобразованию [42]:

$$\begin{cases} w^{A(t+1)} = w^{A(t)} + \dfrac{\eta}{N} x y^{B(t)} \Theta\left(-y^{A(t)} y^{B(t)}\right), \\ w^{B(t+1)} = w^{B(t)} + \dfrac{\eta}{N} x y^{A(t)} \Theta\left(-y^{A(t)} y^{B(t)}\right). \end{cases} \tag{1.26}$$

В соответствии с вышеуказанными формулами функция Θ (*функция шага* или функция Хевисайда, см. выражение (1.4)) определяется следующим образом:

$$\Theta(a)=\begin{cases}1, & a\geq 0,\\ 0, & a<0.\end{cases} \tag{1.27}$$

Из формулы (1.27) следует, что функция Θ(·) возвращает ненулевую величину для неотрицательных входных величин. Таким образом, веса активизируются только тогда, когда выходы обеих сетей противоположны. После каждого шага обучения производится нормализация активизированных векторов весовых коэффициентов.

Степень схождения векторов весов двух НС вычисляется по формуле

$$R(t)=W^{A(t)}W^{B(t)}. \tag{1.28}$$

Динамика процесса взаимного обучения или синхронизации весовых коэффициентов может быть подсчитана с использованием параметра $\beta = t/N$ по аналогии с формулой (1.24):

$$\frac{dR}{d\beta}=(R+1)\left(\sqrt{\frac{2}{\pi}}\eta(1-R)-\eta^2\frac{\alpha}{\pi}\right), \tag{1.29}$$

где α – это угол между векторами W^A и W^B, а величина $R = \cos\alpha$.

Расчеты показывают, что для малых величин коэффициента η оба персептрона стремятся к состоянию, близкому к полному схождению (что означает, что угол $\alpha = 0°$). Вместе с увеличением η векторы весов обоих персептронов отдаляются друга от друга, стремясь к увеличению угла до $\alpha = 133°$ (при $\eta = 1{,}818$). При превышении этой величины векторы полностью не соответствуют друг другу (угол между ними $\alpha = 180°$).

2. КРИПТОГРАФИЯ И НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ

Математической основой описания и анализа процессов в информационных системах в широком смысле является теория информации. Возникновение этой теории связывают обычно с появлением фундаментальных работ К. Шеннона (C. Shannon, см., например, [43]).

Говоря об исторических аспектах научных исследований в области криптографии и криптоанализа, объединенных общим названием «криптология», необходимо отметить тот факт, что весь период с древних времен до 1949 года можно назвать донаучным периодом, когда средства «закрытия» информации не имели строгого математического обоснования. Поворотным моментом, придавшим криптографии научность и выделившим ее в отдельное направление математики, явилась публикация работы К. Шеннона «Теория связи в секретных системах» [44].

В настоящее время существует обширнейшая библиографическая база научных изданий (например, [45–48]) и учебной литературы (например, [49–51]) в рассматриваемой предметной области.

Одна из основных задач в области криптографии (помимо обеспечения целостности передаваемых данных и аутентификации сущностей) – обеспечение конфиденциальности передаваемых по каналам связи данных. Этот аспект связи может быть выполнен путем шифрования сообщения таким образом, чтобы неавторизованные субъекты системы не узнали содержимое передаваемой информации. Используя шифры, отправитель преобразует открытый текст сообщения M в *криптограмму* или *шифртекст* C. Он использует *функцию зашифрования*, которая принимает открытый текст сообщения в качестве аргумента и возвращает зашифрованный текст в качестве выходного значения. Функция зашифрования также зависит от некоторых дополнительных данных, называемых *криптографическими ключами*. Получатель, используя соответствующие алгоритмы, преобразует полученную криптограмму обратно в открытый текст. *Функция расшифрования* на стороне получателя тоже зависит от дополнительных параметров, являющихся криптографическим ключом.

Помимо защиты данных, обеспечения их целостности и выполнения операций аутентификации с помощью методов криптографии решаются и иные прикладные задачи:

– электронная цифровая подпись (ЭЦП; Electronic Digital Signature, или просто Digital Signature) [45–49];

– электронное голосование (E-Voting) [52, 53];

– доказательство с нулевым разглашением (Zero-Knowledge Proof) [45, 46, 54–56];

– разделение секрета (Secret Sharing) [45, 46, 48, 49, 57];

– протоколы конфиденциального (многостороннего) вычисления (Secure Multi-Party Computation) [45–49, 57, 58].

В вопросах криптографии мы можем выделить два основных направления: симметричные и асимметричные криптосистемы.

2.1. Формальное описание криптографической системы

В наиболее общем виде процессы в криптографической системе можно представить в виде рис. 2.1.

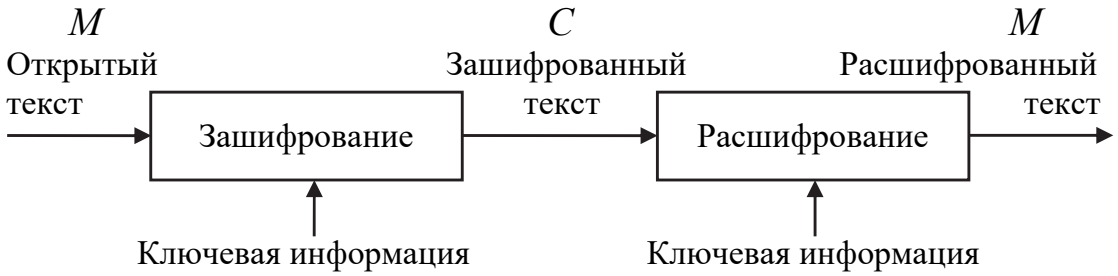

Рис. 2.1. Общее представление криптографической системы

Используются также иные термины для описания криптографической системы: *исходное сообщение* (Message, *M*), или *открытый текст* (Plaintext); *зашифрованное сообщение – шифртекст* или *шифрограмма* (Ciphertext, *C*).

После обратного преобразования (расшифрования) получаем исходный (или приближенный к нему, *M'*) документ.

Функция зашифрования (Encryption) *E* в математическом виде представляется следующим образом:

$$E(M) = C. \qquad (2.1)$$

Расшифрованием (Decryption, функция D) восстанавливается исходное сообщение M:

$$D(C) = M. \tag{2.2}$$

Поскольку смысл зашифрования и последующего расшифрования сообщения заключается в восстановлении исходного открытого текста, справедливо следующее равенство:

$$D(E(M)) = M. \tag{2.3}$$

В этом кратком анализе прямого и обратного процессов криптографического преобразования шифр отождествляется с криптографическим алгоритмом, представляющим собой математическую функцию, которая используется для зашифрования и расшифрования информации.

Определение 2.1. Основой криптографических преобразований является ***ключевая информация***, или ключи (Keys, K). Множество возможных ключей называют ***пространством ключей***.

Ключ – секретный параметр, управляющий ходом преобразования и определяющий криптостойкость (устойчивость к взлому) шифра. В соответствии с этим выражения (2.1)–(2.3) могут быть представлены следующим образом:

– функция зашифрования E:

$$E_{Ke}(M) = C, \tag{2.4}$$

– функция расшифрования D:

$$D_{Kd}(C) = M \tag{2.5}$$

или

$$D_{Kd}(E_{Ke}(M)) = M. \tag{2.6}$$

Соотношения (2.4)–(2.6) описывают симметричные криптосистемы, если $K_e = K_d$, и асимметричные – в противном случае.

Основные недостатки симметричных систем связывают с двумя проблемами:

1) синхронизация (согласование) двумя сторонами (A и B) общего ключа;

2) хранение и защита ключей, принадлежащих некоторому абоненту.

В основном для решения этих проблем У. Диффи и М. Хеллман придумали упомянутый алгоритм (Диффи – Хеллмана, ДХ) [41], а И. Кантер и У. Кинцель – альтернативу алгоритму ДХ, основанную на ИНС [31–33].

2.2. Алгоритм Диффи – Хеллмана

Два абонента, желающие передавать друг другу конфиденциальные сообщения, должны согласовать криптографический ключ для организации связи. В силу различных ограничений наиболее целесообразным является согласование общего ключа с использованием открытого канала связи. В этом варианте важно, чтобы потенциальный злоумышленник, перехватывающий сообщения, отправляемые во время согласования ключей, не мог определить значения, генерируемые двумя сторонами.

Алгоритм (протокол) ДХ использует вычислительно сложную задачу определения *дискретного логарифма* [41, 49–51, 59–61] в конечных циклических группах F_q^* или конечных полях $GF(q)$ (или F_q), состоящих из q элементов; q – простое число (обычно принимается, что $q = 2^p - 1$, p – целое число).

Установленными и общеизвестными параметрами описания системы является конечное поле F_q, а также элемент g, принадлежащий этому полю ($g \in F_q$). Хорошо, если элемент g – генератор группы F_q^*, однако это условие не является необходимым. В результате использования метода генерирования ключа могут появиться только те элементы F_q^*, которые являются степенями g. Однако если требуется, чтобы каждый элемент группы F_q^* имел одинаковую вероятность выпадения, то g должен быть генератором этой группы.

Определение 2.2. ***Дискретное логарифмирование*** – задача обращения функции g^x в некоторой конечной мультипликативной группе. Наиболее часто задачу дискретного логарифмирования рассматривают в мультипликативной группе кольца вычетов или конечного поля, а также в группе точек эллиптической кривой над конечным полем [49–51].

Предположим, что два пользователя системы (A и B) хотят согласовать тайный ключ, используя протокол ДХ. В категориях теории чисел это будет случайный элемент F_q^*, установленный на основании нижеследующего алгоритма.

Пользователь A выбирает случайное число X_A, которое является секретным элементом и известно только лицу A. Затем вычисляет функцию

$$Y_A = g^{X_A},\ Y_A \in F_q. \tag{2.7}$$

Полученный результат пересылается пользователю B по публичному каналу.

Сторона B выполняет аналогичную операцию: выбирает случайный (тайный) элемент X_B ($X_B \in \{1, 2, \ldots, q-1\}$), а затем вычисляет

$$Y_B = g^{X_B},\ Y_B \in F_q. \tag{2.8}$$

Результат пересылает пользователю A (также по публичному каналу). Далее каждая из сторон соответственно выполняет вычисление:

$$K_A = \left(Y_B\right)^{(X_A)} = g^{((X_B)(X_A))}, \tag{2.9}$$

$$K_B = \left(Y_A\right)^{(X_B)} = g^{((X_A)(X_B))}. \tag{2.10}$$

Таким образом, обе стороны получают одинаковый результат (общий ключ):

$$K_{AB} = K_A = K_B = g^{((X_A)(X_B))}. \tag{2.11}$$

Трудность взлома рассмотренной системы основывается на следующем утверждении ДХ: задача вычисления $g^{((X_A)(X_B))}$, когда известны только g^{X_A} и g^{X_B}, является вычислительно трудной.

В случае, когда F_q является мультипликативной группой кольца вычетов по модулю q $(\mathbb{Z}/q\mathbb{Z})^*$, $((\mathbb{Z}/q\mathbb{Z})^* = \{1, 2, \ldots, q-1\})$, решение называют также индексом числа K по основанию q. Индекс числа K_{AB} по основанию q гарантированно существует, если g является первообразным корнем по модулю q.

Рассмотрим процедуру согласования ключа по Диффи – Хеллману на основе операций для мультипликативной группы кольца

вычетов $(\mathbb{Z}/q\mathbb{Z})^*$. Следуя правилам вычисления в соответствии с формулами (2.7)–(2.11), легко определить числа Y_A и Y_B:

$$Y_A = g^{(X_A)} \bmod q, \tag{2.12}$$

$$Y_B = g^{(X_B)} \bmod q. \tag{2.13}$$

При этом числа X_A ($X_A \in \{1, 2, \ldots, q-1\}$) и X_B ($X_B \in \{1, 2, \ldots, q-1\}$) также выбираются абонентами соответственно A и B, никому не передаются и не разглашаются; $1 \le (Y_A, Y_B) \le q-1$.

Каждая из сторон передает другой стороне вычисленные на основе выражений (2.12) и (2.13) числа. После этого A и B вычисляют соответственно:

$$K_A = \left(Y_B\right)^{(X_A)} = g^{((X_B)(X_A))} \bmod q, \tag{2.14}$$

$$K_B = \left(Y_A\right)^{(X_B)} = g^{((X_A)(X_B))} \bmod q. \tag{2.15}$$

В результате обе стороны получают одинаковое число: $K_A = K_B = K_{AB}$, и могут его использовать в качестве общего тайного ключа. Третья сторона, имея открытый доступ к числам q, g, Y_A и Y_B, но не зная значений чисел X_B и X_A, должна найти решение уравнений:

$$X_A = \log_g Y_A \bmod q, \tag{2.16}$$

$$X_B = \log_g Y_B \bmod q, \tag{2.17}$$

и далее – найти K_{AB}.

Определение 2.3. ***Проблемой дискретного логарифма*** в $(\mathbb{Z}/q\mathbb{Z})^*$ называют задачу назначения для данного $y \in (\mathbb{Z}/q\mathbb{Z})^*$ такого натурального числа x, что $y = g^x \bmod q$.

Анализ показывает, что наиболее эффективные алгоритмы поиска решения указанных уравнений требуют около $g^{1/2}$ вычислительных операций [41]. В этом заключается сложность решения проблемы дискретного логарифмирования. Даже самые современные алгоритмы вычисления дискретного логарифма имеют очень высокую сложность, которая сравнима со сложностью наиболее быстрых алгоритмов разложения чисел на множители.

Предположение ДХ априори такое же сильное, как и проблема вычисления дискретного логарифма. Это значит, что если дискретный логарифм в данной группе может быть легко подсчитан, то

утверждение ДХ невыполнимо. Предполагается, что верно и обратное утверждение. Но это только пока предположения.

Сегодня пока не существует эффективных методов решения рассматриваемой задачи дискретного логарифмирования. Но не исключено, что такие методы будут найдены в будущем. Таким образом, формально не доказана равнозначность проблемы взлома системы обмена ключами ДХ и проблемы вычисления дискретного логарифма. Итак, протокол обмена ключами ДХ можно считать безопасным до тех пор, пока проблема дискретного логарифма остается вычислительно трудной. Возможность эффективного решения этой задачи связана с *квантовыми вычислениями* [62]. Теоретически доказано, что с их помощью дискретный логарифм можно вычислить за полиномиальное время. Если полиномиальный алгоритм вычисления дискретного логарифма будет реализован, это будет означать практическую непригодность криптосистем на его основе. Однако алгоритмов, способных вычислить дискретный логарифм за полиномиальное время, нет.

Задача дискретного логарифмирования является одной из основных задач, на которых базируется криптография с открытым ключом [61].

2.3. Криптографическая хеш-функция

Практически все криптографические протоколы зависят от безопасности хеш-функций.

Определение 2.4. ***Хеширование сообщений*** *M* является важным видом криптографических преобразований, которые называются *хеш-функциями* *h*(*M*) или *функциями свертки*, а их результаты называют *хешем*, *хеш-кодом* или *хеш-образом*. В соответствии с последним хешируемое сообщение *M* называют *прообразом* [45–51, 58, 63, 64].

Если хеш-функции удовлетворяют некоторым дополнительным криптографическим условиям, то их можно использовать для защиты целостности информации. Другими криптографическими приложениями, в которых полезны хеш-функции, являются оптимизация схем цифровой подписи, защита фраз-паролей и передача строки без ее раскрытия.

Дадим формальное определение хеш-функции.

Определение 2.5. Пусть $\{0, 1\}^l$ – множество всех двоичных строк некоторой фиксированной длины l ($l > 0$), $\{0, 1\}^*$ – множество всех двоичных строк конечной длины. Тогда ***хеш-функцией*** $h(X)$ называется преобразование вида

$$h(X):\{0, 1\}^* \to \{0, 1\}^l, \tag{2.18}$$

где l – разрядность хеш-образа.

Наиболее распространенный способ построения хеш-функции носит название *конструкции Меркла – Дамгара* (Merkle – Damgard) [45–49, 63, 64], которая основана на использовании *функции сжатия G*:

$$G:\{0, 1\}^l \times \{0, 1\}^{l'} \to \{0, 1\}^l, \tag{2.19}$$

где l' – длина блока m_i ($i = 1, 2, \ldots, t$) сообщения M (в битах; $M = X$), дополненного (или расширенного: $M \to M'$) в соответствии с используемым алгоритмом хеширования (класса MD – Message Digest или класса SHA – Secure Hash Algorithm) и разделенного на r ($r \geq 1$, $r \geq t + 1$) блоков одинаковой длины l': $M' = m_1, \ldots, m_r$ (см., например, [51, с. 121–134]).

Преобразования на основе функции G (2.19) можно представить следующим образом:

$$\begin{gathered} h_0 = IV, \\ h_i = G(h_{i-1}, m_i), \\ h(M) = h_r, \end{gathered} \tag{2.20}$$

где IV – l-разрядный фиксированный вектор инициализации (Initialization Vector) процесса хеширования, m_i – i-й блок дополненного сообщения, M'; $h_1, h_2, \ldots, h_{r-1}$ – промежуточные результаты вычисления хеша.

На i-м итеративном шаге функция сжатия G принимает результат предыдущего шага (h_{i-1}) и i-й блок данных (m_i), а также формирует результат: $h_i = G(h_{i-1}, m_i)$. Результат r-го шага объявляется хеш-образом исходного сообщения M: $h(M) = h_r$.

Исследования хеш-функций были сосредоточены на поиске и формулировании требований, которые следует наложить на функцию G, чтобы h гарантированно удовлетворяло определенным

свойствам. Речь идет о двух основных результатах исследований по этой проблеме. Первый из таких результатов сформулировали независимо И. Дамгар [65] и Р. Меркл [66]. Этот результат касается устойчивости хеш-функции к коллизиям.

Определение 2.6. ***Хеш-функция, устойчивая к коллизиям,*** – это функция $h(X = M)$, удовлетворяющая следующим условиям:

– аргумент X может иметь произвольную длину, а результат $h(X)$ имеет фиксированную длину в l битов ($l \geq 128$);

– хеш-функция должна быть *однонаправленной*.

Первое неформальное определение однонаправленной хеш-функции (One-Way Hash Function, OWHF) было дано, по-видимому, Р. Мерклом в [66].

Определение 2.7. ***Однонаправленная хеш-функция*** – это функция h, удовлетворяющая следующим условиям:

– аргумент $X = x_1, x_2, \ldots, x_r$ ($X = M$) функции h может иметь произвольную длину, а результат $h(X)$ имеет фиксированную длину в l битов;

– хеш-функция должна быть однонаправленной в том смысле, что при заданном Y в образе h вычислительно «трудно» найти сообщение X такое, что $h(X) = Y$, и при заданных X и $h(X)$ «трудно» найти сообщение $X_1 \neq X_2$ такое, что $h(X_2) = h(X_1)$.

Хеш-функция устойчива к коллизиям, если «трудно» найти два разных сообщения ($X_1 \neq X_2$), хеш которых одинаков: $h(X_1) = h(X_2)$.

Определение 2.8 [65]. Степень ***устойчивости хеш-функции*** семейства H ***к коллизиям*** основана на том, что для любого вероятностного алгоритма Δ с полиномиальным временем и любого полинома P существует подмножество экземпляров h размера l, для которых алгоритм Δ с вероятностью не менее $1/P(l)$ находит такие $X_1 \neq X_2$, что $h(X_2) = h(X_1)$.

Результат, полученный С. Лаем и Дж. Месси (X. Lai, J. Massey) [67], дает необходимые и достаточные условия для G, чтобы хеш-функция h считалась «идеально безопасной». Эти условия сформулированы следующим образом: предположим, что хеш сообщения имеет длину l и что сообщение $X = M$ (без расширения) содержит не менее 2 блоков. Тогда нахождение второго прообраза для h с фиксированными IV требует 2^l операций тогда и только тогда, когда нахождение второго прообраза для G с произвольно выбранным h_{i-1} требует 2^l операций.

2.4. Нейронные сети и криптография

Как известно, конфиденциальность информации обеспечивается в основном за счет использования алгоритмов зашифрования/расшифрования. Среди возможных способов реализации существуют такие, которые реализуются на основе ИНС, составляющих основу искусственного интеллекта. Эти методы объединены в один раздел криптографии, называемый *нейрокриптографией* (Neuro-Cryptography или Neural Cryptography). Это название было предложено в 1995 году С. Дурленсом (S. Dourlens) в работе по криптоанализу DES [68].

Иерархическая структура экосистемы нейрокриптографии представлена на рис. 2.2.

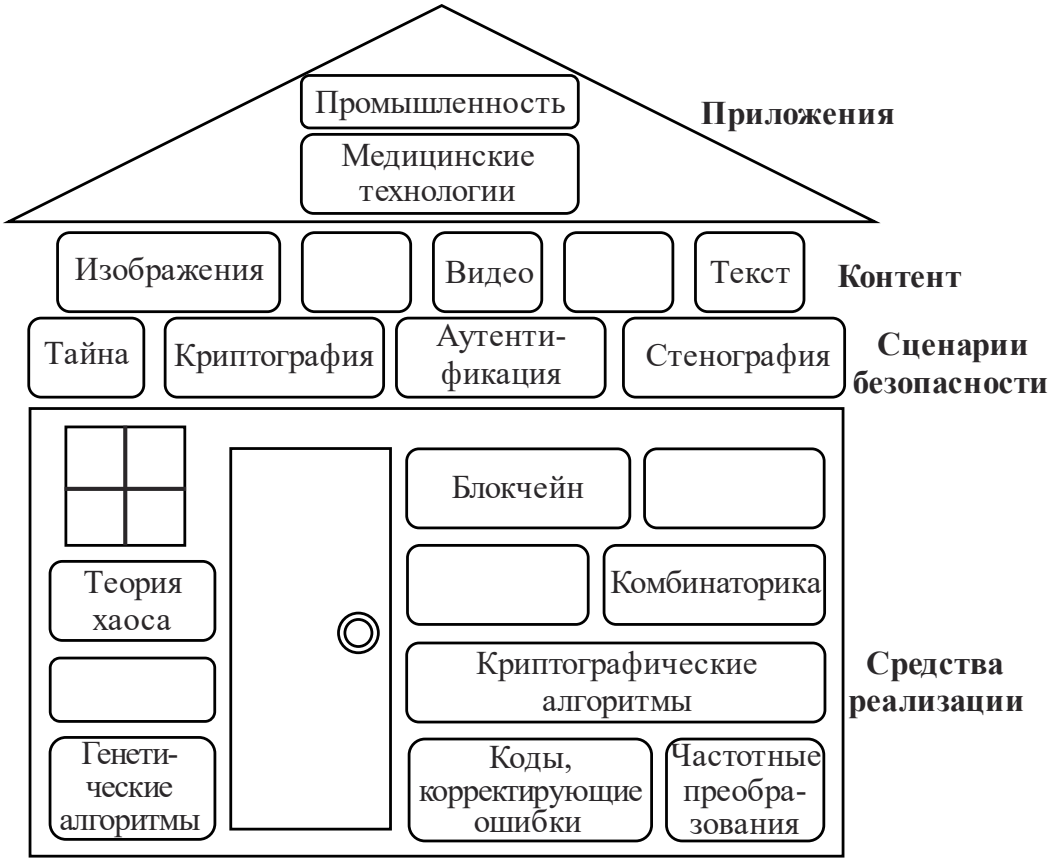

Рис. 2.2. Иерархическая структура экосистемы нейрокриптографии [69]

Криптография и нейросетевые технологии с течением времени становятся все более тесно связанными между собой. Авторы статьи [69] характеризуют эту связь состоянием «войны и мира». «Во-

енный» аспект объясняется ролью НС в решении задач криптоанализа, т. е. взлома шифров, «мирный» – использованием криптографии для повышения безопасности НС, а также применением НС в криптографических приложениях, о чем говорилось выше.

В контексте такого дихотомического представления рассматриваемых предметных областей целесообразно упомянуть следующие исследования: анализ применения *клеточных НС* (Cellular NN) для реализации операций отображения над данными логического типа и применение клеточных автоматов для шифрования данных [70]; использование НС в системах с закрытым ключом, в алгоритмах шифрования изображений, генерации псевдослучайных чисел, анализа и генерации цифровых водяных знаков [71]; в тот же период было предложено использовать специальную модель НС для стандарта беспроводной телекоммуникационной сети IEEE 802.16 с целью обмена закрытыми ключами по общедоступному каналу посредством обучающей синхронизации [72], что явилось дополнением к основополагающим результатам, известным из [30–33]; достаточно глубокие исследования сравнительного анализа известных методов использования ИНС в криптографии [73–75].

По мере того как взаимосвязь между ИНС и криптографией становится все более прочной, взаимное влияние между ними усложняется. Таким образом, существующие публикации и авторы парадигмы «войны и мира» во взаимосвязи и взаимоотношении «криптография – нейронные сети» [69] свидетельствуют о наличии следующих аспектов анализируемой дихотомии:

– применение НС в криптоанализе криптографических схем и атак на них (война);

– применение НС для повышения безопасности и эффективности криптосистем (мир);

– использование криптографии для обеспечения конфиденциальности НС (мир).

3. СТРУКТУРА И ПРИНЦИПЫ ВЗАИМОДЕЙСТВИЯ НЕЙРОННЫХ СЕТЕЙ НА ОСНОВЕ АЛГЕБРЫ ДЕЙСТВИТЕЛЬНЫХ ЧИСЕЛ

3.1. Общая характеристика двух взаимодействующих нейронных сетей

Синхронизация, или взаимное обучение сетей, основана на общих принципах, рассмотренных выше в подгл. 1.4, 1.5. Классический способ обучения или синхронизации сетей описан в подгл. 1.5. В этом подходе необходимо создать обучающую выборку, содержащую пары символов: входной вектор и ожидаемый результат работы сети. С помощью такого набора весá обученной сети модифицируются таким образом, чтобы для заданного входного вектора результат работы сети был максимально близок к ожидаемому результату.

Расширением этого классического подхода может быть случай, когда имеется механизм, формирующий входной вектор и ожидаемый результат работы сети на постоянной основе. Данный механизм будет своего рода «учителем», а обученная сеть – «учеником». Если бы учителем была другая НС с постоянными во времени весовыми коэффициентами, но построенная на основе той же топологии, что и сеть ученика, то после обучения сеть-ученик должна была бы работать точно так же, как сеть-учитель. Таким образом, можно получить функциональную копию сети-учителя. На рис. 3.1 показана схема такого обучения сети. В этом случае нет необходимости создавать или хранить какой-либо обучающий набор, потому что входные векторы и ожидаемые значения генерируются во время обучения сети.

Алгоритм обучения сети выглядит следующим образом.

1. Генерируется новый общий входной вектор X_i, $i = 1, 2, \ldots$:

а) сеть-учитель получает этот вектор в качестве входных данных;

б) сеть-ученик получает тот же входной вектор.

2. Сравниваются результаты работы сетей:

а) сеть-учитель вычисляет свой результат и подает его для сравнения с результатом обучаемой сети;

б) сеть-ученик также вычисляет результат операции и представляет его для сравнения.

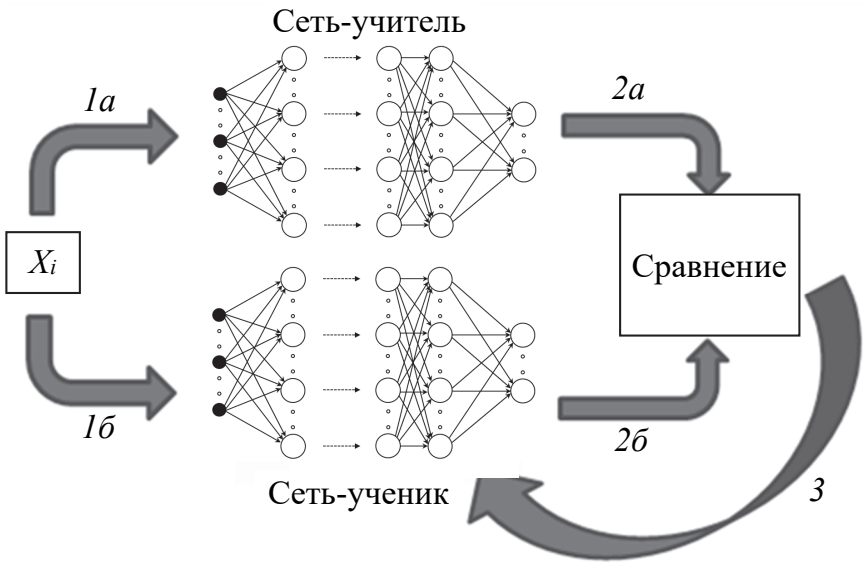

Рис. 3.1. Концепция обучения сети-ученика сетью учителем

3. В зависимости от результата сравнения изменяются весовые коэффициенты обучаемой сети, при этом далее используется выбранный алгоритм обучения сети-ученика.

Шаги с 1-го по 3-й повторяются до тех пор, пока результаты работы обеих сетей не совпадут в течение заданного количества шагов. В результате обученная сеть должна прийти в состояние, в котором она будет имитировать работу сети-учителя.

Можно проанализировать случай, когда работа сети-учителя меняется с течением времени и для одних и тех же входных векторов в разное время обучения она может возвращать разные ожидаемые результаты. В связи с этим важно, с какой скоростью сеть-ученик будет адаптироваться к изменениям параметров сети-учителя.

Наиболее интересным является случай, когда две сети изучают друг друга. Каждая из сетей действует и как учитель, и как ученик. Схема такой процедуры показана на рис. 3.2.

1. Генерируется новый общий входной вектор X_i, $i = 1, 2, \ldots$:

а) сеть *1* получает этот вектор в качестве входных данных;

б) сеть *2* получает тот же входной вектор.

2. Сравниваются результаты работы сетей:

а) сеть *1* вычисляет свой результат и представляет его (как ожидаемый) для сравнения с текущим результатом работы сети *2*;

б) сеть *2* также вычисляет результат операции и представляет его (как ожидаемый) для сравнения с текущим результатом работы сети *1*.

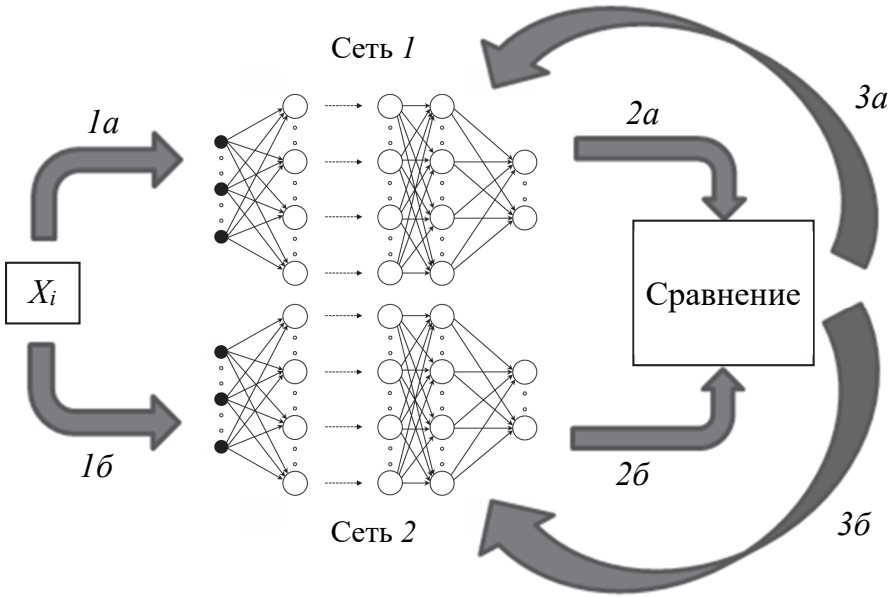

Рис. 3.2. Концепция взаимного обучения сетей

3. В зависимости от результата сравнения используется выбранный алгоритм обучения сети:

а) модифицируются соответствующие весовые коэффициенты сети *1*;

б) модифицируются соответствующие весовые коэффициенты сети *2*.

Шаги с 1-го по 3-й повторяются до тех пор, пока не совпадут входные векторы для обеих НС в течение заданного количества шагов.

На основе рассмотренных вариантов взаимодействия двух НС базируется одно из основных направлений изучения и анализа применения НС в криптографии: согласование двумя удаленными абонентами тайного криптографического ключа, осуществляемого по незащищенному каналу. Здесь решается задача, аналогичная реализуемой на основе алгоритма ДХ [41, 42, 46–49].

Идея использования ИНС для согласования криптографического ключа принадлежит, как уже отмечалось выше, И. Кантеру (I. Kanter) и В. Кинцелю (W. Kinzel), а также присоединившемуся к их исследованиям Е. Кантеру (E. Kanter) [31–34]. Общее значение векторов весов двух персептронов, отличающееся только знаком, может быть использовано в качестве секретного ключа двух сторон (сетей). Такие сети называют *древовидными машинами четности* (Tree Parity Machine, TPM).

В целом схема протокола выглядит следующим образом.

1. Две стороны коммуникации, *A* и *B*, создают сети с одинаковой топологией и устанавливают общий метод их обучения. И топология, и метод обучения сетей доступны потенциальному злоумышленнику. Векторы начальных весов выбираются независимо для обеих сетей, и эти значения хранятся в секрете.

2. На каждом этапе обучения генерируется новый общедоступный входной вектор, который используется обеими сторонами для расчета выходного сигнала сетей.

3. Стороны обмениваются этими выходными результатами через открытый канал связи. Каждый рассматривает результат сети другой стороны как ожидаемый результат и обучает свою сеть в соответствии с выбранным методом обучения.

4. Если обе сети возвращают одни и те же результаты в течение достаточно длительного времени, можно считать, что в итоге обучения они имеют согласованные значения весов. При этом обе стороны завершают процесс обучения. В противном случае генерируется новый входной вектор, и процедура повторяется с шага 2.

5. Значения весов обученных сетей в установленной последней итерации используются в качестве совместного криптографического ключа.

Таким образом, как подтвердили дальнейшие исследования создателей метода и их коллег, такое взаимное обучение сети приводит к установлению одинаковых значений весовых векторов в обеих сетях [31, 35, 40, 42]. Это состояние называется *синхронизацией* НС (см. определение 1.9).

При этом исследователи отмечали, что, если сети продолжат обучение, изменения веса будут такими же, поэтому их значения останутся согласованными. В отличие от классического обучения нейронной сети, нет фиксированных значений веса, к которым стремятся обе сети.

Рассмотренный алгоритм имеет некоторые особенности. Первая из них касается безопасности всей системы. Поскольку если у третьей стороны (*E*), наблюдающей за процессом обмена, появится возможность получить значения внутреннего состояния вектора весов одного из персептронов (*A* или *B*), участвующих в процессе обмена ключами, то она будет иметь возможность синхронизироваться с ним. Вторая особенность касается практической реализации всей системы. Обмен информацией в компьютерных системах

базируется на пространстве битов, или на дискретных величинах. Поэтому обоснованным является вопрос: будет ли правильным (или сходящимся к общему вектору) процесс обмена ключами для векторов весов с дискретными (битовыми) величинами?

Рассмотренный выше процесс синхронизации сходится только тогда, когда векторы весов нормализованы на каждом шаге обучения. Этап нормализации налагает некоторые ограничения на величины векторов весов. Этого типа ограничение и используется в системе, основанной на дискретных величинах векторов весов. Каждая величина вектора весов остается в границах следующего множества [31–33, 42, 76–80]:

$$w_i^{A/B} \in \{-L, -L+1, ..., 0, ..., L-1, L\}, \tag{3.1}$$

где L – это установленное целое число ($L \in \mathbb{Z}$). Следовательно, каждая величина вектора весов персептрона может принять одну из $2L + 1$ возможных величин.

Исходя из практических соображений, рассмотренное выше правило обучения (1.26) может быть модифицировано. Это следует из того факта, что векторы весов обоих персептронов лучше синхронизируются к параллельному состоянию, чем непараллельному (или параллельному, но с противоположным знаком). Таким образом, модифицированное правило обучения формально можно представить следующим образом:

$$\left.\begin{aligned} W^{A(t+1)} &= W^{A(t)} - Xy^A\Theta\left(y^A y^B\right), \\ W^{B(t+1)} &= W^{B(t)} - Xy^B\Theta\left(-y^A y^B\right). \end{aligned}\right\} \tag{3.2}$$

Входная величина для обоих персептронов – это случайный вектор X_i, $X_i \in \{-1, 1\}$. Из формул (3.2) следует, что если выходы обеих сетей идентичны, т. е. $y^A = y^B$, то векторы весов персептронов производят шаг в следующем направлении: $X_i y^A$. Учитывая ограничение (3.1), налагаемое на величины векторов весов, по выполнении каждого шага нужно проверить, находятся ли элементы каждого вектора весов в заданном промежутке. Если такая величина превышает заданные границы $|w_i| > L$, то она должна быть скорректирована, чтобы вернуться к данному ограничению. Этот этап довольно простой и заключается в присваивании величины, соответствующей нарушенной границе: $|w_i| = L$ или $|w_i| = -L$ (в зависимости от стороны, с которой наступило нарушение границы промежутка).

Векторы весов персептронов активизируются согласно формулам (3.2). К тому же, из того факта, что элементы входного вектора X являются дискретными числами ($\{-1, 1\}$), вытекает, что величины их активизированных весов могут принимать три значения: 1, 0 или –1. Случай получения нулевой величины означает либо несоответствие выходных величин обоих персептронов, либо выход за пороговую величину данного ограничения. Второй случай, когда не происходит активизации величины вектора весов вследствие нарушения границы $|L|$, представляет собой единственную возможность для величин вектора весов приблизиться друг к другу на единицу [76]. Говоря точнее, величины последующих элементов векторов весов движутся по прямой линии в пространстве $[-L, L]$. Если веса подчиняются процессу обучения, то соответствующие им величины движутся в одном направлении. Если одна из величин нарушит границу ($-L$ либо L), то ей будет возвращено значение нарушенной границы, и она в действительности не выполнит никакого движения (останется неизмененной). Это означает, что если соответствующая величина второго вектора весов производит движение по направлению к упомянутой границе, то расстояние между двумя величинами сократится на 1. Следовательно, этот случай является сущностью процесса синхронизации, так как обеспечивает сокращение расстояния между соответствующими величинами векторов весов.

Для двух персептронов с N-мерными векторами весов существует N двухэлементных групп (пар величин), которые движутся в промежутке $[-L, L]$. Анализируя движения внутри каждой последующей пары, можно определить экспоненциальную зависимость вероятности сходимости от времени [42, 76]:

$$P(t) \approx P(0) \exp(-t / r);\ r = O(L^2). \tag{3.3}$$

Следовательно, для определения общего времени процесса синхронизации можно использовать зависимость $NP(t) \approx 1$, из которой следует, что время синхронизации персептронов можно оценить так:

$$t_{\text{synch}} \approx r \ln(N). \tag{3.4}$$

В действительности опыты показывают, что для набора параметров $N = 100$ и $L = 3$ два персептрона синхронизируют свои внутренние векторы весов приблизительно за 100 шагов обучения.

Кроме того, время синхронизации логарифмически увеличивается с ростом величины N. Нужно также добавить, что интруз, наблюдающий за процессом обмена данных двух сетей, способен синхронизироваться с их векторами весов. Поэтому единичный персептрон не может быть использован как безопасный источник обмена секретным ключом (какой-нибудь криптографической системы).

К вопросу безопасности рассматриваемой системы мы вернемся в подгл. 3.9. Однако следует отметить, что процесс синхронизации, основанный на использовании двух одиночных персептронов, передает слишком много информации, что является слабым местом этой системы. Оппонент (E), наблюдающий за обменом информацией между двумя персептронами (следовательно, знающий и значения входного вектора, и выходные величины обоих персептронов), в состоянии синхронизироваться (согласоваться) с наблюдаемыми сетями A и B.

3.2. Структура и функционирование сети на основе древовидной машины четности

Как мы отметили, необходимо использовать такую систему взаимодействующих между собой НС, чтобы скрыть как можно больше информации. Иначе говоря, чтобы оппонент E, наблюдающий за процессом синхронизации, не был в состоянии синхронизировать свои векторы весов с сетями A и/или B. С другой стороны, степень скрытности должна давать возможность синхронизировать оба вектора и, таким образом, не может быть слишком высокой. Как будет показано далее, именно степень скрытности внутренней информации в существенной мере влияет на безопасность всей системы.

Этим условиям удовлетворяет многослойная НС с внесенными в ее строение некоторыми модификациями, которые позволяют реализовать данные требования. Это уже упомянутая выше сеть, называемая *древовидной машиной четности* (TPM). Нейроны представляют собой персептроны с дискретными векторами весов.

Сеть TPM – это односторонняя двухуровневая (нередко ее называют трехуровневой) сеть [30–35, 39, 42, 76–95]. Результат работы всей сети не влияет на входные сигналы, используемые на последующих этапах ее работы, поэтому обратная связь отсутствует. Первый уровень этой сети состоит из типичных искусственных

нейронов, построенных в соответствии с моделью МакКаллока – Питтса [7] без смещения (порог активации нейрона). Во втором слое всегда используется только один определенный нейрон, выполняющий операцию умножения результатов работы нейронов первого слоя. Характерной чертой TPM-сетей являются отдельные *рецептивные поля* (Receptive Field) для каждого нейрона. Это означает, что каждый нейрон первого слоя получает на входе «свой собственный» вектор входного воздействия, который является фрагментом (частью) всего входного вектора, что и придает сети древовидную структуру. Схема построения такой сети представлена на рис. 3.3. На каждом шаге взаимного обучения сетей их входные сигналы генерируются на основе случайного выбора.

Если представим описанную структуру в соответствии со стандартными моделями ИНС, то можем сказать, что входом для сети является один вектор X, т. е. существуют связи всех входных импульсов со всеми нейронами первого слоя. Однако использование *неперекрывающихся рецептивных полей* (Non-Overlapping Receptive Field) означает, что соединения, не отмеченные на рис. 3.3, имеют веса, постоянно равные нулю. Это позволяет легче понять топологию сети TPM и анализировать эту сеть с помощью инструментов, созданных для стандартных ИНС. Непересекающиеся поля должны дополнительно затруднять (а на практике – предотвращать) возможности для определения результатов работы нейронов первого слоя на основе известного результата работы всей сети и параметров входных векторов [42, 76–95]. Это важно для применения данных сетей в криптографии.

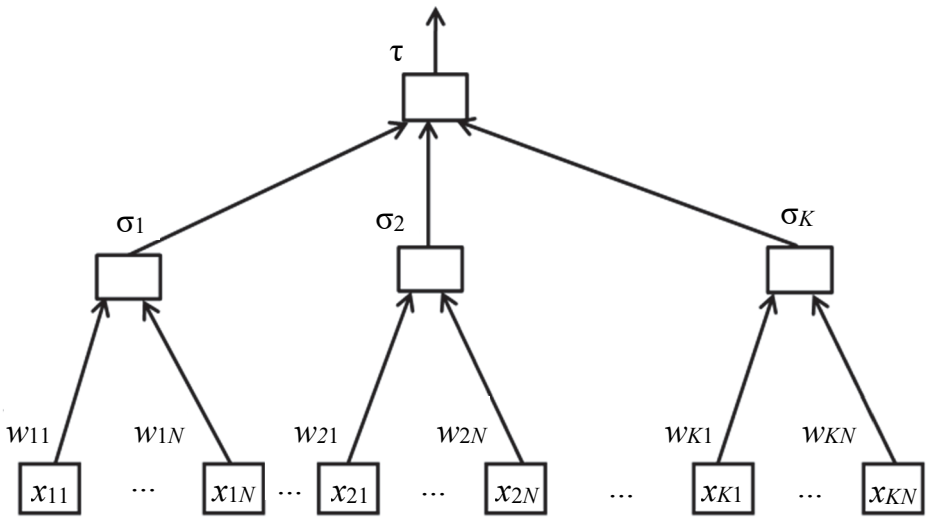

Рис. 3.3. Структура сети TPM

Топология сети TPM описывается двумя параметрами: K – количество нейронов в первом скрытом слое и N – количество входных сигналов для каждого нейрона.

Работа сети TPM основана на целочисленной арифметике, что делает этот механизм понятным для человека, простым в реализации и эффективным при использовании аппаратных ресурсов компьютера. Входные векторы – это входные сигналы для сети TPM: $X \in \{-1, 1\}^{KN}$. Каждому нейрону соответствует фрагмент входного вектора: $X_i \in \{-1, 1\}^{N}$, где i = 1, 2, …, K. Вес нейронов первого слоя – целые числа от $-L$ до L, как было отмечено выше. Следовательно, длина вектора входных значений всей сети равна KN.

Входные сигналы от каждого нейрона умножаются на соответствующие веса и затем суммируются (см. формулы (1.19), (1.25)). Если сумма (φ_i) положительна, нейрон передаст сигнал, равный 1. В противном случае – –1. Таким образом, нейроны первого слоя возвращают результат либо –1, либо +1, т. е. значения, которые попадают в ту же область, что и входные сигналы.

Нейрон, расположенный во втором слое сети, умножает приходящие на него импульсы, но не умножая при этом их на какие-либо веса (на практике мы можем принимать веса, постоянно равные 1). Результат работы последнего нейрона является результатом работы всей сети. Если он равен 1, это означает, что во внутреннем слое было четное количество отрицательных сигналов. В случае, когда конечный результат равен –1, на первом уровне должно быть нечетное количество отрицательных результатов. Данные свойства и определили название рассматриваемой сети: *машина (контроля) четности*.

Формальное определение сети TPM. Топология сети соответствует деревьям из теории графов. Пусть K ($K \in \mathbb{N}$) – количество искусственных нейронов в первом скрытом слое некоторой сети TPM, а N ($N \in \mathbb{N}$) – количество входов каждого нейрона в отдельности. Кроме того, пусть $X = \{-1, 1\}$ будет набором возможных значений, принимаемых каждым сигналом, достигающим сети (входного слоя), и пусть $x_{ij} \in X$, где $1 \le i \le K$ и $1 \le j \le N$, будет j-м входным сигналом для i-го нейрона первого слоя. Импульсы, достигающие K нейронов первого слоя сети, образуют K последовательностей по N элементов в каждой, что и образует KN-разрядную последовательность на входе сети.

Это также позволяет определить количество возможных комбинаций входных сигналов: 2^{KN} или 2^N независимых входных комбинаций для каждого нейрона.

Пусть $L \in \mathbb{N}$ – фиксированное натуральное число, и пусть $w_{ij} \in \mathbb{Z}$, где $1 \le i \le K$ и $1 \le j \le N$, – вес, связанный с входным импульсом x_{ij}. В сетях TPM каждый вес $w_{ij} \in \{-L, -L+1, \ldots, L-1, L\}$, поэтому L – абсолютное максимальное значение для каждого веса. Каждый вес принимает одно из $2L + 1$ значений, что означает, что все KN весов сети принимают одно из $(2L+1)^{KN}$ состояний.

В некоторых работах для анализа процесса взаимодействия TPM вводится понятие *нейрокриптографической сети* (НКС) [96–98].

Определение 3.1. Две НС (A и B) на основе архитектуры TPM, характеризующиеся одинаковыми параметрами (K, N, L или K-N- L) и предназначенные для согласования криптографических ключей, будем называть ***нейрокриптографическими сетями*** (НКС) и обозначать соответственно $\langle \mathrm{TPM}(K,N,L)^A \rangle$, $\langle \mathrm{TPM}(K,N,L)^B \rangle$.

Будем далее рассматривать обозначения $\langle \mathrm{TPM}(K,N,L)^{A/B} \rangle$ или просто TPM как равнозначные.

Работа нейронов первого слоя сети заключается в вычислении суммы входных сигналов, умноженных на соответствующие веса, по формуле

$$\varphi_i = \sum_{j=1}^{N} x_{ij} w_{ij}, \tag{3.5}$$

где $1 \le i \le K$ – номер искусственного нейрона, а $1 \le j \le N$ – номер входного импульса (см. выражения (1.1), (1.3), (1.19)). Результат, вычисленный по формуле (3.5), является выходом сумматора. Мы предполагаем, что нейроны сети TPM не имеют дополнительного импульса смещения. Можно заметить, что поскольку входной импульс $x_{ij} \in \{-1, 1\}$, то в суммах произведений сомножитель x_{ij} влияет только на знак суммированных значений веса, потому что

$$x_{ij} w_{ij} = \begin{cases} -w_{ij}, \text{ если } x_{ij} = -1, \\ w_{ij}, \text{ если } x_{ij} = 1. \end{cases}$$

Сумма произведений соответствующих весов и входных сигналов φ_i является аргументом функции активации данного нейрона.

Блок активации использует биполярную ступенчатую (пороговую) переходную функцию (см. п. 1.2.3), определяемую формулой

$$\sigma_i = f(\varphi_i) = \begin{cases} -1, \text{ если } \varphi_i \le 0, \\ 1, \text{ если } \varphi_i > 0, \end{cases}$$

или, в соответствии с выражениями (1.17), (1.19), (1.25) и (3.5), а также с учетом нормирования по N, выходной сигнал σ_i i-го элемента скрытого слоя сети можем представить в следующем виде [42]:

$$\sigma_i = \operatorname{sign}(\mu_i) = \operatorname{sign}\left(\left(\frac{1}{\sqrt{N}}\right)\sum_{j=1}^{N} w_{ij}x_{ij}\right), \qquad (3.6)$$

где $1 \le i \le K$ – номер искусственного нейрона.

Отметим, что при $\varphi_i = 0$ функция f принимает значение –1 (в некоторых исследованиях при $\varphi_i = 0$ принимается, что $f(\varphi_i) = 0$).

Как мы уже отметили, входные импульсы нейрона второго слоя являются результатом работы нейронов предыдущего слоя. Для сети $\langle \text{TPM}(K,N,L)^{A(B)} \rangle$ характерно то, что эти импульсы не умножаются на какие-либо веса. Нейрон последнего слоя выполняет операцию умножения поступающих на него сигналов, а результатом его работы является результат работы всей сети:

$$\tau = \prod_{i=1}^{K} \sigma_i. \qquad (3.7)$$

Результат работы каждого нейрона скрытого слоя: $\sigma_i \in \{-1, 1\}$, следовательно, $\tau \in \{-1, 1\}$. Формула (3.7) показывает, что $\tau = -1$ достигается при нечетном количестве отрицательных входных импульсов нейронов первого уровня и $\tau = 1$ – при четном. Стандартная комбинаторика показывает, что для K нейронов в первом слое существует 2^K комбинаций выходных сигналов этого слоя. Точно так же для каждого из двух возможных значений τ выхода всей сети TPM существует 2^{K-1} комбинаций выходных сигналов нейронов первого слоя, которые генерируют данный выход.

Общая схема процесса синхронизации двух сетей $\langle \text{TPM}(K,N,L)^{A(B)} \rangle$ (взаимного обучения сетей) в графическом виде представлена на рис. 3.4.

Неоднозначность и большое количество возможных вариантов представления результата функционирования всей сети на основе результатов функционирования нейронов внутреннего слоя имеют

решающее значение для безопасности протокола согласования криптографических ключей на основе синхронизации сетей TPM.

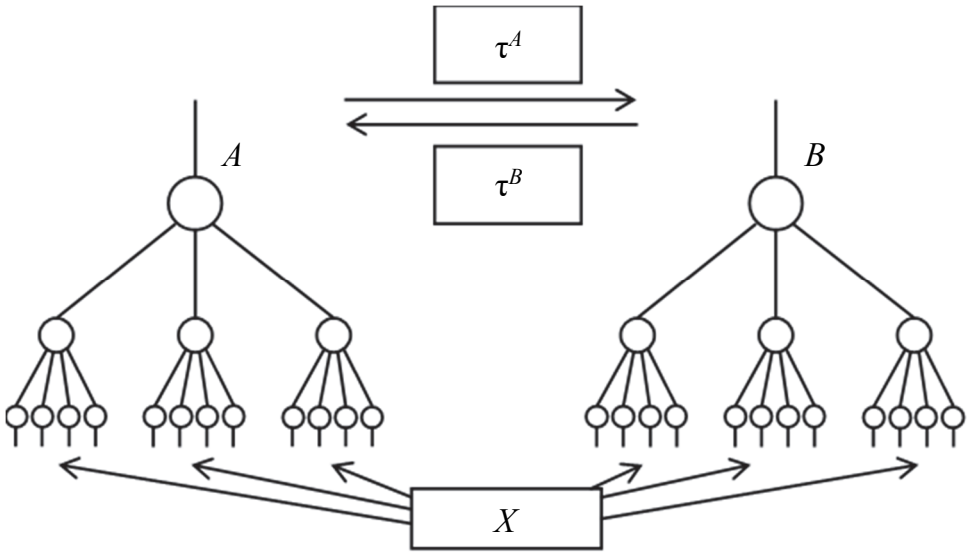

Рис. 3.4. Иллюстрация процесса синхронизации двух сетей: $\langle \mathrm{TPM}(K,N,L)^A \rangle$ и $\langle \mathrm{TPM}(K,N,L)^B \rangle$

Синхронизирующиеся сети A и B обмениваются своими выходными сигналами по открытому каналу связи, и важно, чтобы злоумышленник (E), перехватив эти данные, не смог определить внутренние значения входных сигналов сетей по крайней мере до окончания процесса синхронизации сетей A и B (рис. 3.5). Без этих данных сторона E не сможет правильно изменять веса нейронов первого слоя при обучении своей сетей, даже если E знает топологию сети. Кроме того, количество возможных вариантов значений веса сети TPM, т. е. когда они имеют согласованные результаты, злоумышленник должен проанализировать 2^{K-1} вариантов модификации весовых коэффициентов.

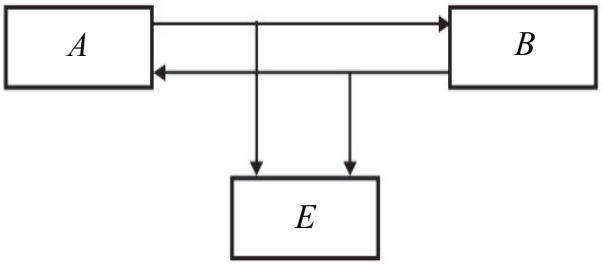

Рис. 3.5. Схема взаимодействия между двумя партнерами (A и B) с прослушиванием канала пассивным злоумышленником (E)

Обучая сети, доверенные стороны A и B изменяют веса своих сетей в зависимости от результата работы сети другой стороны. Таким образом, их сети влияют друг на друга. Потенциальный злоумышленник не знает весов синхронизирующих сетей и не меняет их веса, что затрудняет его задачу. Злоумышленнику также необходимо определить динамику изменения весов сетей A и B.

Рассмотренная архитектура TPM и принципы взаимодействия трех сторон (A, B, E) могут несущественно изменяться при детализации основополагающей цели построения и использования сетей. Но, как правило, эти изменения связаны с особенностями процесса синхронизации сетей A и B [31–38, 82–86].

3.3. Особенности процесса синхронизации сетей TPM

Понятно, что именно процесс синхронизации оказывает решающее влияние на два важнейших момента:

а) будет ли в итоге (на последнем шаге взаимного обучения сетей) принято «правильное» решение – обе сети ($\langle \mathrm{TPM}(K, N, L)^{A(B)} \rangle$) в результате характеризуются полностью совпадающими векторами весов;

б) будет ли общая длительность процесса синхронизации такой, что этот параметр позволит минимизировать возможности третьей стороны (E) для выполнения необходимых процедур согласования весовых коэффициентов сети E с результатом синхронизации сетей A и B.

Подчеркнем существенную деталь: с точки зрения безопасности анализируемых процессов параметр (число) L синхронизируемых сетей играет такую же роль, как и длина ключа в классической криптографии. Детали и результаты исследования влияния L на безопасность (время) синхронизации сетей будут рассмотрены ниже.

В общем случае, как было отмечено выше, обучение ИНС заключается в модификации ее весовых коэффициентов таким образом, чтобы конечный результат работы всей сети был наилучшим, т. е. в определенном смысле оптимальным. Текущее значение вектора весов сети в момент времени t изменяется на определенную величину Δw, которая называется *приращение веса*. Полученный результат будет определяться значением этого веса в следующий

момент времени, $t + 1$. Это можно записать таким образом (см. формулу (1.6)):

$$W_i^{(t+1)} = W_i^{(t)} + \Delta W, \tag{3.8}$$

где $W_i^{(t)}$ – значение вектора весов i-го персептрона сети в момент времени t. Изменения весов относятся к обеим сетям одинаково, поэтому в рамках рассмотрения данного вопроса индексы j будут опущены.

При взаимном обучении сетей TPM изменение весов происходит, предположим, в соответствии с алгоритмами, основанными на правиле Хэбба в варианте обучения с учителем (см. п. 1.4.1.2). Это правило реализуется в соответствии с формулой (3.8), в которой параметр ΔW можно представить следующим образом (см. формулы (1.10)–(1.12)):

$$\Delta W = \eta X d\ . \tag{3.9}$$

На основе приращения ΔW, определяемого выражением (3.9), значение веса может возрастать до бесконечности, поэтому применяются различные модификации правила Хэбба, использование которых описано, в частности, в [2, 11, 20, 29, 42, 78–80, 87–94] и др.

Используя анти-Хэббовское обучение, веса изменяются только тогда, когда сети возвращают противоположные результаты.

Сети A и B получают в качестве входных данных один и тот же случайный вектор X, который должен оставаться секретным на протяжении всего процесса обучения, и вычисляют соответствующие выходные величины: τ^A и τ^B. Стороны обмениваются выходными значениями, как это показано на рис. 3.4.

Модификация вектора весов обеих сетей может осуществляться на основе одного из рассмотренных правил, а также правила «случайного блуждания» (Random-Walk Learning Rule) [34, 42, 81–95] (см. также соотношение (3.2)).

Проанализируем перечисленные правила обучения сетей с формальной точки зрения.

Правило (метод) Хэбба. Веса изменяются, если на данном шаге результаты работы двух сетей не отличаются ($\tau^A = \tau^B$), при этом

$$\tau^{A/B} = \prod_{j=1}^{K} y_j^{A/B} = \prod_{j=1}^{K} \sigma\left(\alpha_j^{A/B}\right) = \prod_{j=1}^{K} \sigma\left(\sum_{i=1}^{N} w_{ij}^{A/B} x_{ij}\right). \tag{3.10}$$

Индекс A/B обозначает, что данная операция касается обеих сетей: A и B, а единичный индекс A или B – что операция касается соответственно только одной из сетей.

В формуле (3.10) используется модифицированная функция знака σ, определяемая следующим образом:

$$\sigma\left(\alpha_j^{A/B}\right) = \begin{cases} 1, \alpha_j^{A/B} \geq 0, \\ -1, \alpha_j^{A/B} < 0. \end{cases} \tag{3.11}$$

Изменяются соответствующие параметры только тех нейронов, которые возвращают тот же результат, что и вся сеть:

$$W_j^{A/B} = \begin{cases} w_{ij}^{A/B} + \tau^{A/B} x_{ij}, \tau^A = \tau^B \wedge \tau^{A/B} = y_j^{A/B}, \\ w_{ij}^{A/B}, \text{ в противном случае.} \end{cases} \tag{3.12}$$

Изменение параметров веса обеих сетей на каждом шаге синхронизации осуществляется в соответствии со следующей общей формулой:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + x_{ij}\sigma_i. \tag{3.13}$$

Отметим также, что векторы весов на каждом шаге обучения должны быть подвержены процессу нормализации. Для персептронов, у которых величины весов являются целыми числами, процесс нормализации связан с необходимостью наложения ограничений на них.

Как было уже проанализировано ранее, величина каждого весового коэффициента не может выйти за границы промежутка $[-L, L]$. Следовательно, каждый шаг процесса обучения (активизации векторов весов) требует выполнения следующей операции:

$$w_{ij}^{A/B} = \begin{cases} \text{sign}(w_{ij}^{A/B})L, \left|w_{ij}^{A/B}\right| > L, \\ w_{ij}^{A/B}, \text{ в противном случае.} \end{cases} \tag{3.14}$$

Правило (метод) анти-Хэбба. Веса изменяются, если на данном шаге результаты работы двух сетей отличаются ($\tau^A \neq \tau^B$), при этом

$$W_j^{A/B} = \begin{cases} w_{ij}^{A/B} - \tau^{A/B} x_{ij}, \tau^A = \tau^B \wedge \tau^{A/B} = y_j^{A/B}, \\ w_{ij}^{A/B}, \text{ в противном случае} \end{cases} \tag{3.15}$$

или

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - x_{ij}\sigma_i. \qquad (3.16)$$

Правило (метод) случайного блуждания [42, 95]. Установленное значение выхода сети не важно для синхронизации, если оно одинаково для обеих НС. Правило, построенное на учете этой особенности, предусматривает модификацию весов при совпадении результатов работы сетей, как и в методе Хэбба. Однако в рассматриваемом методе изменение весов зависит не от выходного сигнала нейрона из скрытого слоя, а только от входного импульса. Здесь изменяются только веса, которые удовлетворяют равенству $\sigma_i = \tau$. Поскольку в этой ситуации невозможно сказать, какие веса обновлены, не зная внутреннего параметра $(\sigma_1, \sigma_2, \ldots, \sigma_K)$, метод особенно ценен для криптографического применения.

Общее выражение, связывающее параметры весов на двух соседних шагах процесса синхронизации $\langle \mathrm{TPM}(K, N, L)^{A(B)} \rangle$, запишем так:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + x_{ij}. \qquad (3.17)$$

Приведенные правила в виде формул (3.13), (3.16) и (3.17) удовлетворяют выражению (3.8) и отличаются только параметром Δw_{ij}. Поэтому можно упростить запись алгоритма модификации весов, используя лишь приращение веса для каждого из рассмотренных методов следующим образом:

$$\Delta w_{ij} = \begin{cases} -x_{ij}\sigma_i, \text{ для метода анти-Хэбба,} \\ x_{ij}\sigma_i, \text{ для метода Хэбба,} \\ x_{ij}, \text{ для метода случайного блуждания.} \end{cases} \qquad (3.18)$$

Рассмотренные условия, при которых происходит модификация соответствующего весового коэффициента, можно включить в (3.18) с помощью *функции перехода* (назовем ее так) $\Theta(x)$, которая будет определять соответствие результатов работы сетей и результата работы отдельной сети с результатом работы каждого нейрона первого слоя этой сети.

Пусть $\Theta(\cdot)$ – функция перехода, определенная следующим образом (см. для сравнения функцию шага, формально задаваемую соотношением (1.30)):

$$\Theta(x)=\begin{cases}0, \text{ если } x\le 0,\\ 1, \text{ если } x>0.\end{cases} \tag{3.19}$$

Выходы сетей τ^A и $\tau^B \in \{-1, 1\}$, если $\tau^A = \tau^B$, то эти сигналы одновременно равны либо –1, либо 1. В обоих случаях произведение $\tau^A\tau^B = 1 > 0$. Если же $\tau^A \ne \tau^B$, то либо $\tau^A = -1 \wedge \tau^B = 1$, либо $\tau^A = 1 \wedge \tau^B = -1$. В этих случаях произведение $\tau^A\tau^B = -1 < 0$. Используя функцию (3.19), а также произведение результатов работы обеих сетей, можно записать:

$$\Theta\left(\tau^A\tau^B\right)=\begin{cases}0, \text{ если } \tau^A \ne \tau^B,\\ 1, \text{ если } \tau^A = \tau^B.\end{cases} \tag{3.20}$$

Подобным образом можно проверить, одинаковы ли выходные сигналы *i*-го нейрона первого слоя $(1 \le i \le K)$ и всей сети. Используя функцию перехода и соответствующие произведения, получим следующее соотношение для обеих сетей *A* и *B*:

$$\Theta\left(\tau^{A/B}\sigma_i^{A/B}\right)=\begin{cases}0, \text{ если } \tau^{A/B} \ne \sigma_i^{A/B},\\ 1, \text{ если } \tau^{A/B} = \sigma_i^{A/B}.\end{cases} \tag{3.21}$$

Алгоритмы модификации (прироста) весов с учетом соотношений (3.20) и (3.21) можно определить следующим образом (последовательность методов, как и в формуле (3.18)):

$$\Delta w_{ij}=\begin{cases}-\sigma_i^{A/B}x_{ij}\Theta\left(\tau^{A/B}\sigma_i^{A/B}\right)\Theta\left(-\tau^{A/B}\tau^{B/A}\right),\\ \sigma_i^{A/B}x_{ij}\Theta\left(\tau^{A/B}\sigma_i^{A/B}\right)\Theta\left(\tau^{A/B}\tau^{B/A}\right),\\ x_{ij}\Theta\left(\tau^{A/B}\sigma_i^{A/B}\right)\Theta\left(\tau^{A/B}\tau^{B/A}\right).\end{cases} \tag{3.22}$$

Аргументы функции $\Theta(x)$ реализуют условия, которые необходимо выполнить, чтобы изменить весовые коэффициенты сети. В анти-Хэббовском обучении изменение весов происходит тогда, когда результаты работы обеих сетей противоположны, поэтому для сети *A* используется множитель $\Theta(-\tau^A\tau^B)$, а для сети *B* – $\Theta(-\tau^B\tau^A)$. Оба эти условия для сетей *A* и *B* объединены в формуле (3.22). Если результаты работы сетей различны, то произведение $\tau^A\tau^B = -1 < 0$ и функция $\Theta(x) = 1$. Для одинаковых результатов работы сетей *A* и *B* функция $\Theta(x)$ примет значение 0, что обнулит прирост веса Δw_{ij},

поэтому на данном шаге весовые параметры не изменятся. В методах Хэбба и случайного блуждания веса меняются при одинаковых выходных сигналах обеих сетей ($\tau^A = \tau^B$), поэтому согласованность результатов проверяется параметром $\Theta(\tau^A\tau^B)$ для сети *A* и $\Theta(\tau^B\tau^A)$ для сети *B*. Если результаты сетей согласуются, функция перехода принимает значение 1, а веса могут быть изменены, в противном случае функция принимает значение 0, поэтому прирост веса также равен 0, т. е. вектор весов не меняется.

Множитель $\Theta\left(\tau^{A/B}\sigma_i^{A/B}\right)$ функционально «проверяет», согласуется ли результат работы *i*-го нейрона первого слоя с результатом работы всей сети. Верхние индексы указывают, что это относится к обеим сетям. Аналогично сравнению результатов обеих сетей, разные значения результата сети и результата нейрона внутреннего слоя уменьшат прирост веса, а согласованные результаты позволят модифицировать соответствующие веса. Если оба условия выполняются совместно, веса данного нейрона первого слоя модифицируются.

Выход σ_i каждого нейрона первого слоя и каждый входной импульс x_{ij} принадлежат множеству $\{-1, 1\}$, а значения функции $\Theta(x)$ – множеству $\{0, 1\}$. Если веса изменены, то $\Theta(\tau^A\tau^B) = 1 \wedge \Theta(\tau^{A/B}\sigma_i^{A/B}) = 1$, и приращение веса $\Delta w_{ij} = x_{ij}\sigma_i$; $\Delta w_{ij} \in \{-1, 1\}$. Таким образом, модификация веса означает его изменение на 1.

Примеры состояний на выходах элементов сетей при их обучении по методам Хэбба и случайного блуждания приведены в табл. 3.1–3.4. Факт модификации весов *i*-го нейрона (столбец mod 1) обозначен символом «+» и символом «–» – в противоположном случае.

Таблица 3.1

Состояние элементов сети TPM при $K = 1$

σ_1	τ	mod 1
–1	–1	+
1	1	+

В тривиальной сети, состоящей из одного нейрона скрытого слоя, результат работы сети всегда согласуется с выходом этого нейрона. Каждый шаг обучения изменяет веса нейрона. Такая сеть не обеспечивает никакой безопасности, потому что злоумышленник знает, какие веса следует менять на каждом шаге синхронизации.

Таблица 3.2

Состояние элементов сети TPM при $K = 2$

σ_1	σ_2	τ	mod 1	mod 2
–1	–1	1	–	–
–1	1	–1	+	–
1	–1	–1	–	+
1	1	1	+	+

Для $K = 2$ возможен случай, когда выходы обоих нейронов – в состоянии «1», и тогда вся сеть генерирует такой же выходной сигнал. Если результаты работы первого слоя сети (σ_i, $i = 1, 2$) отличаются от результата работы всей сети (τ), то ни один нейрон не меняет веса. В других случаях изменяются веса хотя бы одного нейрона.

Таблица 3.3

Состояние элементов сети TPM при $K = 3$

σ_1	σ_2	σ_3	τ	mod 1	mod 2	mod 3
–1	–1	–1	–1	+	+	+
–1	–1	1	1	–	–	+
–1	1	–1	1	–	+	–
–1	1	1	–1	+	–	–
1	–1	–1	1	+	–	–
1	–1	1	–1	–	+	–
1	1	–1	–1	–	–	+
1	1	1	1	+	+	+
–1	–1	–1	–1	+	+	+
–1	–1	1	1	–	–	+
–1	1	–1	1	–	+	–
–1	1	1	–1	+	–	–

Как видно из табл. 3.3, при наличии 3 нейронов в скрытом слое сети веса изменяются в каждом случае, но в большинстве случаев – только в одном нейроне.

Таблица 3.4

Состояние элементов сети TPM при $K = 4$

σ_1	σ_2	σ_3	σ_4	τ	mod 1	mod 2	mod 3
–1	–1	–1	–1	1	–	–	–
–1	–1	–1	1	–1	+	+	+
–1	–1	1	–1	–1	+	+	–
–1	–1	1	1	1	–	–	+

Окончание табл. 3.4

σ_1	σ_2	σ_3	σ_4	τ	mod 1	mod 2	mod 3
–1	1	–1	–1	–1	+	–	+
–1	1	–1	1	1	–	+	–
–1	1	1	–1	1	–	+	+
–1	1	1	1	–1	+	–	–
1	–1	–1	–1	–1	–	+	+
1	–1	–1	1	1	+	–	–
1	–1	1	–1	1	+	–	+
1	–1	1	1	–1	–	+	–

В последнем примере (табл. 3.4) среднее число модифицированных весов нейронов равно половине K. Для сетей с числом нейронов первого слоя $K < 4$ обычно модифицируются только веса одного нейрона. Лишь для сетей с четным числом нейронов в первом слое может случиться так, что никакие веса не изменятся, несмотря на согласованные результаты работы обеих сетей. Такая ситуация возникает, когда все нейроны первого слоя возвращают –1, тогда вся сеть возвращает 1. Значит, нет ни одного нейрона, результат работы которого соответствовал бы результату работы всей сети. Для всех остальных случаев в сетях с четным числом нейронов, а также в сетях с нечетным числом нейронов первого слоя меняются веса хотя бы одного нейрона.

Как отмечено выше (см. формулу (3.14)), если вновь вычисленное значение веса больше параметра L, оно заменяется на L, аналогично для значений ниже $-L$, которые заменяются на $-L$. Для учета этого ограничения вводится функция $g(x)$:

$$g(x) = \begin{cases} -L, \text{ если } x < -L, \\ L, \text{ если } x > L, \\ x, \text{ если } |x| \le L, \end{cases} \tag{3.23}$$

где $x \in \mathbb{Z}$.

Используя функцию (3.23), общую процедуру модификации вектора весов можно представить в виде следующего выражения:

$$w_{ij}^{A/B(t+1)} = g\left(w_{ij}^{A/B(t)} + \Delta w_{ij}^{A/B}\right), \tag{3.24}$$

которое является расширением формулы (3.8) с учетом характеристик обучения сети TPM. Приращение веса $\Delta w_{ij}^{A/B}$ в выражении (3.24) окончательно (с учетом (3.22)) определяется следующим образом:

$$\Delta w_{ij}^{A/B} = \begin{cases} -\sigma_i^{A/B} x_{ij} \Theta\left(\tau^{A/B}\sigma_i^{A/B}\right)\Theta\left(-\tau^A\tau^B\right), \text{ метод анти-Хэбба,} \\ \sigma_i^{A/B} x_{ij} \Theta\left(\tau^{A/B}\sigma_i^{A/B}\right)\Theta\left(\tau^A\tau^B\right), \text{ метод Хэбба,} \\ x_{ij} \Theta\left(\tau^{A/B}\sigma_i^{A/B}\right)\Theta\left(\tau^A\tau^B\right), \text{ метод случайного блуждания.} \end{cases} \quad (3.25)$$

Формулы (3.24) и (3.25) являются математическим представлением алгоритмов обучения (синхронизации) сетей $\langle \text{TPM}\left(K, N, L\right)^{A/B} \rangle$ [42]. Целесообразно также подчеркнуть, что при $N \to \infty$ рассмотренные правила обучения сводятся к «случайному блужданию» [96].

3.4. Динамика процесса синхронизации сетей TPM

3.4.1. Характеристика шагов синхронизации

Важность и сущность процесса синхронизации НС заключается в возможности применения его в криптографических системах.

Нейронная синхронизация – это стохастический процесс, состоящий из дискретных шагов, в котором веса сетей корректируются в соответствии с правилами, представленными выше. Понятно, что однонаправленное обучение и двунаправленная синхронизация дают в результате разные эффекты. Это различие связано с динамикой процессов, происходящих в указанных системах несмотря на то, что их состояния полностью определяются начальными весовыми векторами W_i сети TPM и последовательностью случайных входных векторов X_i.

Сходимость весовых коэффициентов синхронизируемых сетей $\langle \text{TPM}(K, N, L)^{A/B} \rangle$ (формально – уменьшение разницы между весами сетей A и B), а также продолжительность этого процесса до достижения состояния синхронизации, т. е. равенства векторов весов определяют безопасность процесса. И именно на анализе двух указанных характеристик синхронизации TPM, архитектура которых может соответствовать разным параметрам N и K, сосредоточены основные усилия исследователей как в научной, так и в прикладной областях [30–34, 36, 42, 79–81, 85–110].

В системе обмена ключами, основанной на взаимном обучении (как и в любой другой криптографической системе), содержатся как

элементы явные (пересылаемые по публичному каналу между сетями), так и секретные (не подлежащие обмену), к которым имеют доступ только пользователи системы. Явные элементы – это вся конструкция системы и ее параметры. Кроме того, в процессе взаимного обучения по публичным каналам пересылаются и входные значения (для каждой из сетей), и их выходные величины. Секретная информация же – это начальное состояние вектора весов и его величина на каждом этапе обучения.

При взаимодействии сетей $\langle \mathrm{TPM}(K,N,L)^{A/B} \rangle$ обучаются обе сети (A и B), которые одновременно выступают в роли ученика и учителя. С формальной стороны обе сети определяются одними и теми же параметрами K-N-L. Каждая из сетей изменяет «свои» веса так, чтобы наилучшим образом согласовать свою работу с работой другой сети.

Если для обучения используются правила Хэбба или случайного блуждания, то веса в обеих сетях имеют одинаковые параметры:

$$\forall_{1\le i\le K}\forall_{1\le j\le N} w_{ij}^{A} = w_{ij}^{B}, \tag{3.26}$$

для обучения же по анти-Хэббу – противоположные:

$$\forall_{1\le i\le K}\forall_{1\le j\le N} w_{ij}^{A} = -w_{ij}^{B}. \tag{3.27}$$

В подавляющем большинстве публикаций, посвященных процессам синхронизации TPM, время синхронизации и количество шагов (итераций) этого процесса определяются на основе соответствующих симуляторов, позволяющих сравнивать между собой текущие векторы весовых коэффициентов обеих сетей. При практической же реализации явления синхронизации в криптографии невозможно сравнить оба весовых вектора, поскольку они являются секретными. Единственная информация, из которой можно сделать вывод о синхронизации сети, это равенство результатов обеих сетей. Таким образом, условие окончания сетевой синхронизации состоит в том, чтобы обе сети обменивались совместимыми, но потенциально изменяющимися в последующих итерациях результатами в течение достаточно длительного времени. При этом следует отметить, что, несмотря на обмен совместимыми результатами, сети могут иметь разные веса.

Процесс взаимного обучения двух сетей приводит к установлению непротиворечивых значений весов (3.26) и (3.27) обеих сетей

во времени в зависимости от изначально сгенерированных весов и динамики процесса обучения. На рис. 3.6 для примера представлены гистограммы распределения количества итераций (шагов) до достижения синхронизации сетями с разными значениями параметра N при фиксированном K [30].

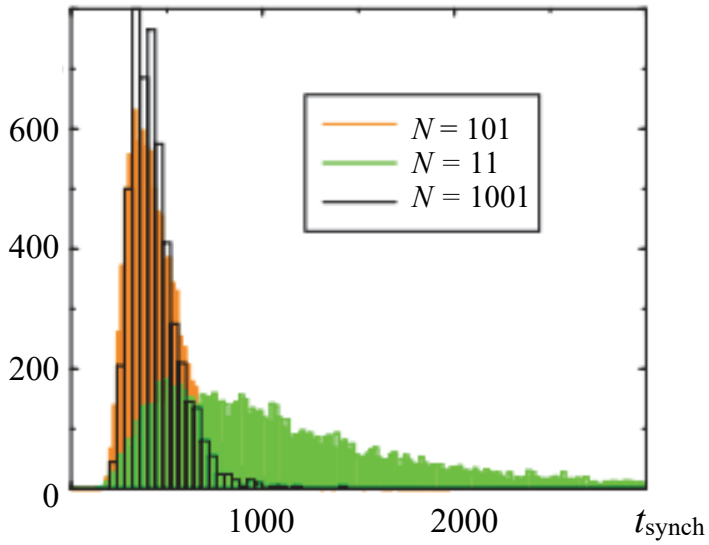

Рис. 3.6. Распределения времен синхронизации двух ТРМ

Другой пример [92]. Исследовались два случая: компьютерная программа симуляции работы сетей «имеет» доступ к весовым коэффициентам обеих сетей и может остановить (на основе анализа этих параметров) процесс синхронизации; во втором случае решение о наступлении состояния синхронизации принимается после определенного числа шагов неизменных выходных значений. Кроме того, изучению и анализу подвергались длительности периодов (максимальное число циклов), в течение которых состояние синхронизации не нарушается. При этом предполагалось, что один цикл состоит из следующих операций:

1) генерирование входного вектора X для сетей $\langle \text{ТРМ}(K,N,L)^A \rangle$ и $\langle \text{ТРМ}(K,N,L)^B \rangle$;

2) вычисление выходных значений: τ^A и τ^B;

3) обмен выходными значениями: τ^A – к сети B и τ^B – к A;

4) адаптация (обучение) сетей.

Для каждой пары сетей выполнено не менее 1000 опытов. При этом сети характеризовались следующими параметрами: $K = 3$, $N = 4$, $L \in [5; 50]$. Для каждого сочетания K-N-L получены распреде-

ления количества опытов от числа циклов (t), после которых наступило состояние синхронизации. На рис. 3.7 представлена одна из таких гистограмм, которая характеризует сети с параметрами 3-4-5.

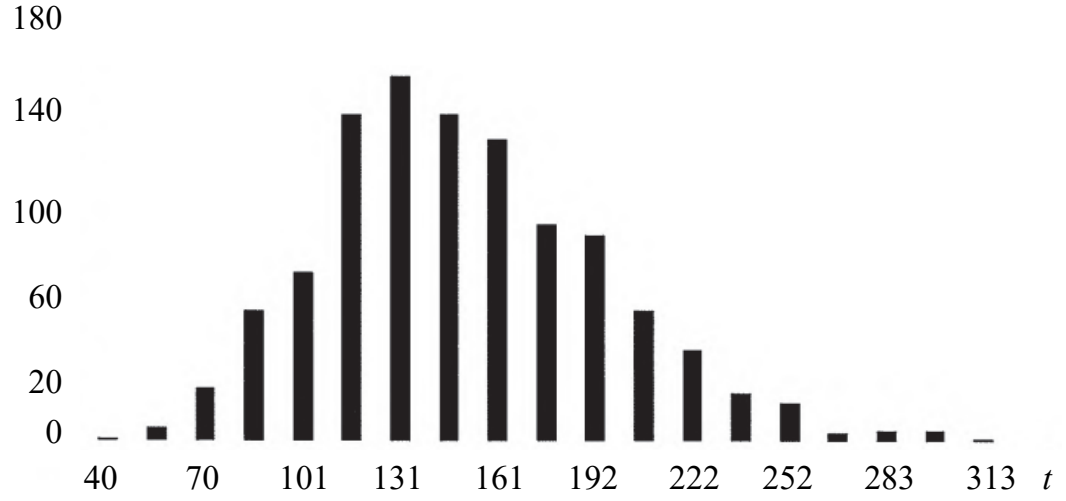

Рис. 3.7. Распределение количества пар сетей по минимальному числу циклов обучения до наступления состояния синхронизации при $K = 3$, $N = 4$, $L = 5$

Вторая из проанализированных ситуаций касается исследования наибольшего времени, в течение которого сети A и B обмениваются постоянной и неизменной информацией до наступления состояния полной синхронизации (сети находятся в состоянии обучения). Эта часть эксперимента выполнена для сетей с параметрами $K = 3$, $N = 4$, $L \in [4; 103]$. Обобщенные статистические характеристики выполненных исследований в этой части приведены в табл. 3.5. Подтверждена почти очевидная закономерность: длительность максимального периода обмена неизменной информацией (колонка «НИ» в табл. 3.5) возрастает с увеличением значений весовых коэффициентов.

К вопросу анализа подобных распределений мы еще многократно вернемся.

В контексте рассмотрения динамических характеристик процесса синхронизации ТРМ отметим, что в соответствии с принятыми правилами взаимного обучения сетей отдельно взятый шаг обучения сети можно отнести к одной из трех категорий [42, 111]:

1) «тихий шаг» (Quiet Step) – это шаг, на котором веса нейронов обеих сетей не меняются;

2) «притягивающий шаг» (Attractive Step) – это шаг, на котором веса обоих нейронов изменяются одновременно таким образом,

который зависит от общего входного вектора X_i; если на этом шаге один из весов выходит за пределы $[-L, L]$, то согласно функции g, определяемой выражением (3.24), он будет уменьшен до L (соответственно – увеличен до $-L$), а другой весовой коэффициент останется без изменений, что уменьшит расстояние между этими весами;

3) «отталкивающий шаг» (Repulsive Step) – на этом шаге меняется значение весов только одной сети, в то время как соответствующие веса другой сети остаются неизменными; как правило, этот шаг приводит к увеличению расстояния между векторами весов.

Таблица 3.5

Статистические результаты моделирования процесса синхронизации сетей TPM

Сеть, *K*-*N*-*L*	Среднее число циклов синхронизации	НИ, число циклов
3-4-4	91,17	52
3-4-10	539,55	154
3-4-20	2112,65	355
3-4-40	8330,13	1022
3-4-60	18 277,36	2014
3-4-80	32 589,64	2859
3-4-100	49 642,65	6653

Пусть сети A и B являются сетями $\langle \text{TPM}(K, N, L)^{A/B} \rangle$ с одинаковой топологией, и пусть i обозначает номер определенного нейрона. Предположим далее, что обучение сетей основано на правилах Хэбба или случайного блуждания. На каждом шаге обучения сетей для каждого из нейронов этих сетей может иметь место один из следующих случаев:

а) $\tau^A \neq \tau^B$, имеет место «тихий шаг»;

б) $\tau^A = \tau^B \wedge \sigma_i^A = \tau^A \wedge \sigma_i^B = \tau^B$, векторы весов изменяются в соответствии с принятым правилом обучения сети, это соответствует «притягивающему шагу» процесса;

в) $\tau^A = \tau^B \wedge \left[\left(\sigma_i^A = \tau^A \wedge \sigma_i^B \neq \tau^B \right) \vee \left(\sigma_i^A \neq \tau^A \wedge \sigma_i^B = \tau^B \right) \right]$, выходы сетей одинаковы; выполняется одна из следующих операций:

– выход i-го нейрона и сети A согласуется с результатом работы всей сети A, в то время как на выходе i-го нейрона сети B сформирован сигнал, отличный от уровня выходного сигнала сети B;

– выход i-го нейрона и сети A отличается выходного сигнала всей сети A, а выходы i-го нейрона B и всей сети B одинаковы; в этих двух частных случаях изменяется только вектор весов нейрона, выход которого соответствует результату работы всей сети, а элементы вектора весов второго нейрона остаются неизменными; эти ситуации соответствуют «отталкивающему шагу»;

г) $\tau^A = \tau^B \wedge \sigma_i^A \neq \tau^A \wedge \sigma_i^B \neq \tau^B$, векторы весов остаются без изменений («тихий шаг»).

При обучении сетей TPM вероятность наступления события, соответствующего «притягивающему шагу» процесса (обозначим эту вероятность p_a), увеличивается до 1, что соответствует состоянию синхронизированных сетей.

Для объяснения явления сетевой синхронизации при взаимном обучении рассмотрим случай взаимодействия TPM с 2 нейронами в первом слое (K = 2) [35], параметры N и L – произвольные: $\langle \text{TPM}(K,N,L)^{A/B} \rangle$. Обучение – по Хэббу и правилу случайного блуждания.

Обозначим $p_i = P\left[\sigma_i^A = \sigma_i^B\right]$ – вероятность того, что в некоторый момент времени t для заданного входного вектора X_i и $i \in [1, K]$ на выходах соответствующих нейронов сетей A и B сформируются одинаковые выходные значения. Конечно, по мере обучения сети вероятность p_i увеличивается до 1 для полностью синхронизированных сетей. Предположим далее, что $p_i = p_1 = p_2$. Тогда может произойти один из четырех случаев:

а) $\sigma_1^A = \sigma_1^B$ и $\sigma_2^A = \sigma_2^B$ – с вероятностью p^2 это будет соответствовать ситуациям, указанным в табл. 3.6;

Таблица 3.6

Возможные состояния выходов нейронов и сетей в целом при K = 2 (для вероятности равной p^2)

Сеть A			Сеть B		
σ_1	σ_2	τ^A	σ_1	σ_2	τ^B
1	1	1	1	1	
1		1	1		1
	1	1		1	1

б) $\sigma_1^A = \sigma_1^B$ и $\sigma_2^A \neq \sigma_2^B$ – с вероятностью $p(1-p)$ это будет соответствовать ситуациям, указанным в табл. 3.7;

Таблица 3.7

Возможные состояния выходов нейронов и сетей в целом при $K = 2$ (для вероятности равной $p(1 - p)$)

Сеть A			Сеть B		
σ_1	σ_2	τ^A	σ_1	σ_2	τ^B
1	1		1		1
1		1	1	1	
	1	1			
				1	1

в) $\sigma_1^A \neq \sigma_1^B$ и $\sigma_2^A = \sigma_2^B$ – с вероятностью $(1-p)p$ это будет соответствовать ситуациям, указанным в табл. 3.8;

Таблица 3.8

Возможные состояния выходов нейронов и сетей в целом при $K = 2$ (для вероятности равной $(1 - p)p$)

Сеть A			Сеть B		
σ_1	σ_2	τ^A	σ_1	σ_2	τ^B
1	1			1	1
1		1			
	1	1	1	1	
			1		1

г) $\sigma_1^A \neq \sigma_1^B$ и $\sigma_2^A \neq \sigma_2^B$ – с вероятностью $(1-p)^2$ это будет соответствовать ситуациям, указанным в табл. 3.9.

Таблица 3.9

Возможные состояния выходов нейронов и сетей в целом при $K = 2$ (для вероятности равной $(1 - p)^2$)

Сеть A			Сеть B		
σ_1	σ_2	τ^A	σ_1	σ_2	τ^B
1	1				
1		1		1	1
	1	1	1		1
			1	1	

Сравнивая τ^A с τ^B во всех проанализированных случаях, отмечаем, что веса могут меняться только в первом и последнем вариантах, так как в остальных – результаты работы сетей различны. Таким образом, мы можем наблюдать 4 возможных случая изменения

вектора весов с вероятностью $(1-p)^2$ и 4 иных подобных случая – с вероятностью p^2, но только в первом варианте изменение происходит в том же направлении. Таким образом, определена вероятность p_a выполнения «неотталкивающего» шага модификации весов в зависимости от вероятности p:

$$p_a = \frac{p^2}{p^2 + (1-p)^2}. \tag{3.28}$$

Аналогичные рассуждения можно провести и для более сложных сетей. На рис. 3.8 показана зависимость p_a от p для сетей TPM с числом нейронов первого слоя $K = 2$ (линия со знаками «×»), $K = 3$ (линия со знаками «+»), $K = 4$ (линия со знаками «*»).

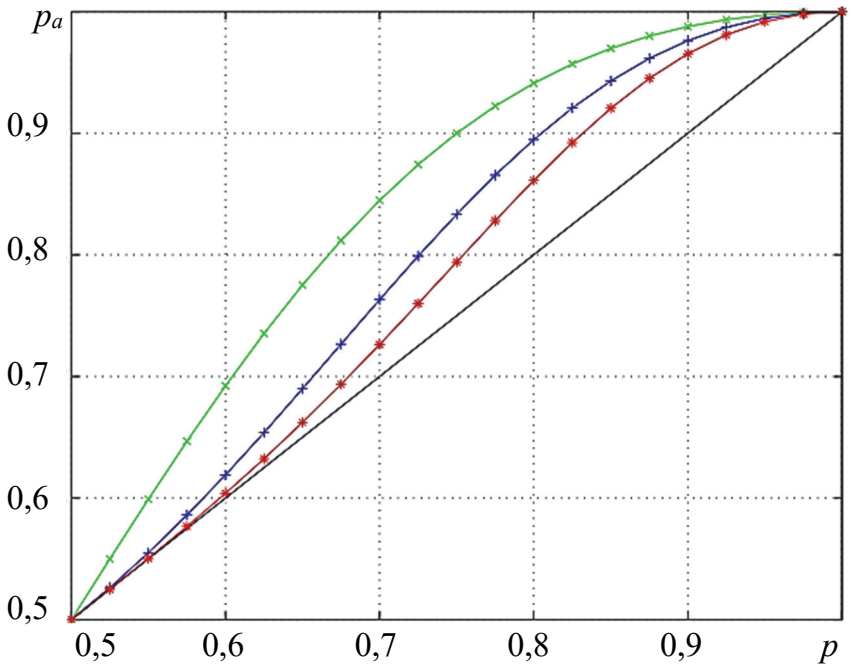

Рис. 3.8. Вероятность совершения согласованной динамики процесса модификации весовых коэффициентов двух сетей TPM [35]

Понятно, что при взаимном обучении сетей реализуются разные типы шагов, но в целом векторы весов обеих сетей сходятся до тех пор, пока через достаточно долгое время не достигнут состояния «согласия». Имея доступ к весам обеих сетей, можно показать, как выглядят эти изменения в процессе синхронизации.

Можно также дополнительно отметить, что при рассмотрении синхронизации может использоваться параметр $\mathrm{dist}(W_i^A, W_i^B)$ – расстояние между векторами весов при $t = i$; $i = 0, 1, \ldots$. Векторы

весов, а также вектор X_i рассматриваются в N-мерном евклидовом пространстве. Гиперплоскость, определяемая вектором X_i, делит это пространство на две части. Точки, соответствующие векторам весом, могут находиться в одной части либо в разных. Как мы видим из табл. 3.6–3.9, при $K = 1, 2$ процесса синхронизации $\langle \text{TPM}(K, N, L)^A \rangle$ и $\langle \text{TPM}(K, N, L)^B \rangle$ как такового нет.

3.4.2. Оценка взаимного соответствия векторов весовых коэффициентов двух TPM

Модификации весов в процессе обучения направлены на то, чтобы сблизить (синхронизировать) векторы весов сетей W^A и W^B до тех пор, пока векторы не сравняются для полностью синхронизированных сетей $\langle \text{TPM}(K, N, L)^{A/B} \rangle$. Поэтому ключевым параметром в процессе синхронизации является взаимное перекрывание этих векторов [42], выраженное как косинус угла между ними (см. формулу (1.22)) [5, 20, 29]:

$$\rho_i = \frac{(W_i^A)(W_i^B)}{\| W_i^A \| \| W_i^B \|}, \tag{3.29}$$

здесь в числителе значится операция скалярного произведения, а $\| \cdot \|$ означает связанную с ним норму.

Для случайных значений весов параметр ρ_i может принимать небольшие значения, а во время обучения он обычно увеличивается до 1. Поэтому угол между векторами весов уменьшается до 0. Тот факт, что сеть синхронизирована, приводит к тому, что косинус достигает значения 1. Следствие в обратном направлении неверно. Рассмотрим два вектора с одинаковым направлением, но разной длиной. Метод позволяет только показать изменчивость перекрытия весовых векторов. К сожалению, тот факт, что $\rho_i = 1$, не означает, что веса согласованы и сети синхронизированы. Мера (3.29) также неприменима при практической реализации сетевой синхронизации для согласования криптографических ключей, поскольку в этом случае векторы весов являются секретными, поэтому ни одна из сторон ($\langle \text{TPM}(K, N, L)^A \rangle$ и $\langle \text{TPM}(K, N, L)^B \rangle$) не «знает», как мы уже подчеркивали, текущих параметров весов обеих сетей.

Процесс сходимости векторов W^A и W^B рассмотрим на графическом примере [112].

Пусть W_i является точкой в N-мерном пространстве. Когда эта точка в процессе обучения совершает свободное движение, ее координаты изменяются. Однако если точка находится на границе указанного пространства, т. е. хотя бы одна координата точки равна L или $–L$, то она не имеет возможности свободно перемещаться и, следовательно, изменять все свои координаты. Это как раз и означает, что точки с различными координатами в процессе обучения могут сойтись, и значения их координат совпадут.

Предположим, что некоторая ⟨TPM(K, N, L)⟩ характеризуется следующими параметрами: $K = 3$, $N = 2$, $L = 3$. В начальный момент времени все пространство заполняется точками. Пусть входной вектор на каждом из 5 шагов обучения последовательно принимает следующие значения: (1, –1), (1, 1), (1, 1), (–1, –1), (1, –1). Тогда движение точек будет происходить таким образом, как показано на рис. 3.9.

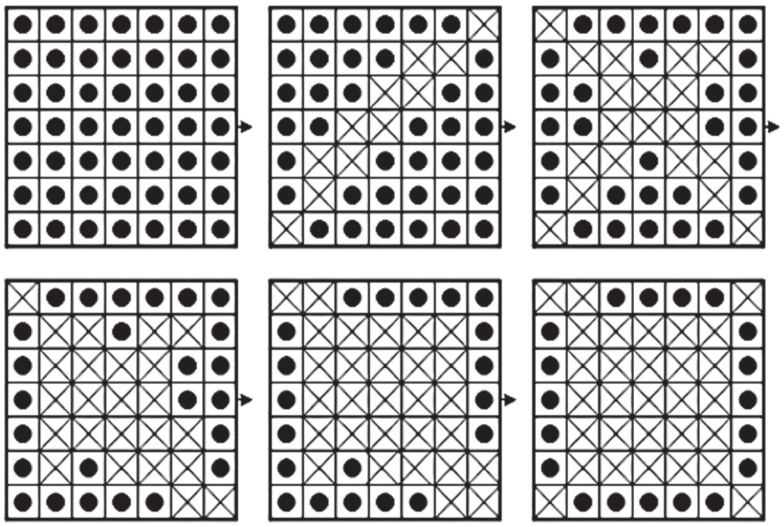

Рис. 3.9. Графическое представление сходимости векторов весов ⟨TPM($K = 3, N = 2, L = 3$)⟩ после 5 шагов обучения

Из анализа рисунка видно, что после каждого шага образуются «пустые» промежутки в нашем пространстве, которые можно исключить из множества возможных решений. Точки движутся из середины к границам пространства. После 13 шагов обучения пространство можно представить рис. 3.10.

На рис. 3.11 приведены примеры осциллограмм изменения взаимного перекрывания весовых векторов обеих сетей ⟨TPM($K = 3$, $N = 2, L = 3$)⟩ при обучении методом случайного блуждания [81, 102, 103]. Время синхронизации (t) составило около 530 (синий график), 410 (желтый график) и 230 (серый график) циклов.

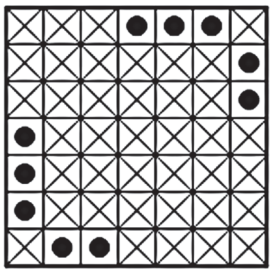

Рис. 3.10. Графическое представление возможного результата сходимости векторов весов $\langle \mathrm{TPM}(K=3, N=2, L=3)\rangle$ после 13 шагов обучения

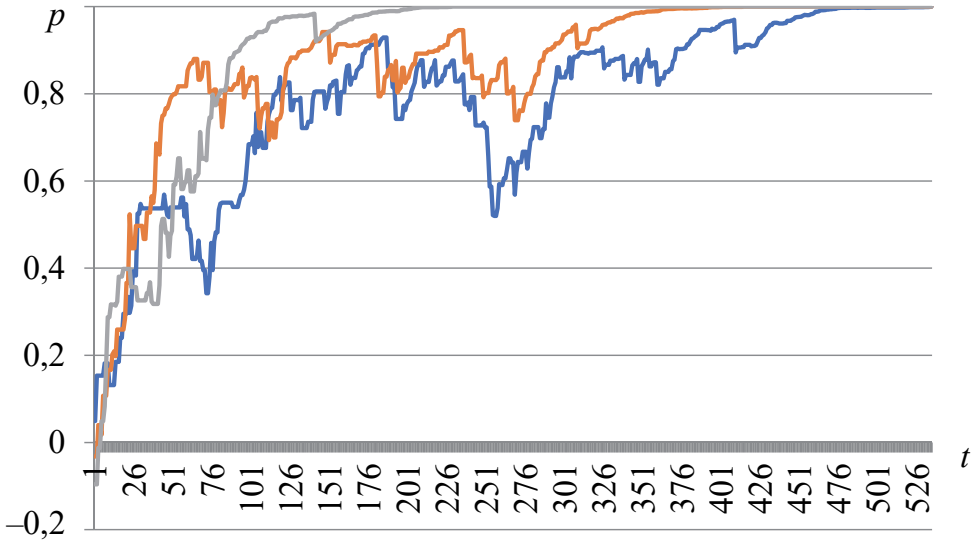

Рис. 3.11. Характер изменения сходимости векторов весов для $\langle \mathrm{TPM}(K=3, N=16, L=5)\rangle$

Как видно, процесс синхронизации характеризуется разными типами шагов: и тихими, и притягивающими, и отталкивающими.

3.4.3. Параметры весов синхронизированных сетей TPM

Правило случайного блуждания, как мы знаем, модифицирует веса, используя только значения входного вектора X, рандомизированные из равномерного распределения, благодаря чему полученные значения весов после синхронизации имеют характер распределения, более похожий на равномерное, чем распределение весов, полученное в результате использования правила Хэбба.

На рис. 3.12 показано сравнение распределений весовых коэффициентов после синхронизации сетей $\langle \mathrm{TPM}(K=3, N=101, L=3)\rangle$ по правилам Хэбба и случайного блуждания после 1000 шагов синхронизации для каждого метода [81, 103].

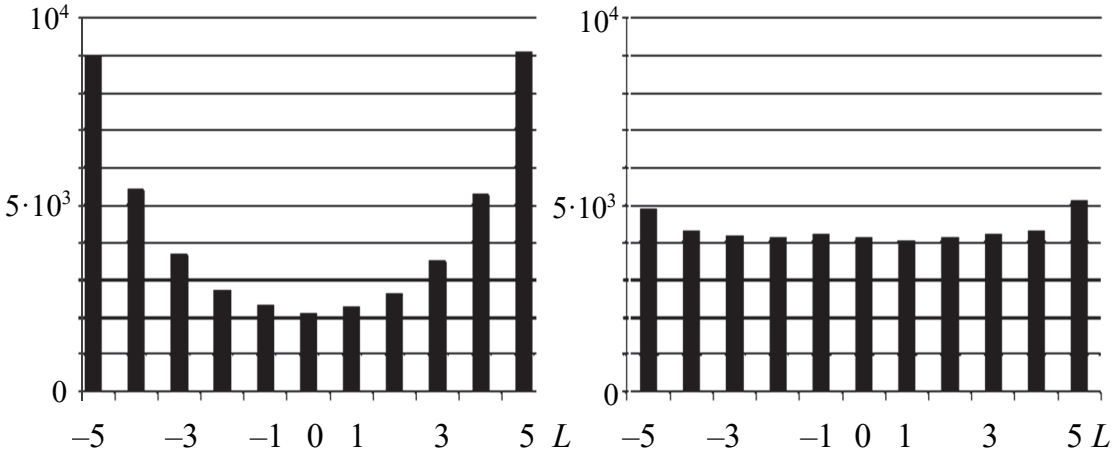

Рис. 3.12. Распределение весов для сетей $\langle \mathrm{TPM}(K = 3, N = 16, L = 5) \rangle$, обученных по правилам Хэбба (слева) и случайного блуждания (справа)

Однажды синхронизированные сети TPM должны оставаться в этом состоянии вне зависимости от времени их дальнейшего обучения. Сети характеризуются одинаковыми парами весов в каждый момент времени $t \geq t_{\text{synch}}$.

При общем входном векторе X_i все нейроны первого слоя и вся сеть возвращают одинаковые результаты. Таким образом, в каждый момент времени t мы получаем:

а) для обучения по методам Хэбба и случайного блуждания:

– для каждого нейрона первого слоя: $\sigma_i^A = \sigma_i^B$;

– для сетей: $\tau^A = \tau^B$;

б) для анти-Хэббовского обучения:

– для каждого нейрона первого слоя: $\sigma_i^A = -\sigma_i^B$;

– для сетей: $\tau^A = -\tau^B$.

Однако эти изменения коррелированы. На каждом последующем шаге сети могут характеризоваться новыми векторами весов, поэтому они должны дополнительно синхронизироваться. Если K нечетно, веса хотя бы одного нейрона в обеих сетях изменятся. Для четных значений K возможен только один случай, когда веса любого нейрона не изменятся – все нейроны скрытого слоя вернут –1. Таким образом, синхронизированные сети TPM при продолжении процесса синхронизации обычно меняют веса, но каждый раз в обеих сетях эти изменения должны носить симметричный характер. Эту особенность можно выполнять в последующих итерациях взаимообучения сетей для получения новых значений криптографических ключей.

Возвращаясь на 2 абзаца вверх по тексту, отмечаем, что «…синхронизированные сети TPM должны оставаться в этом состоянии вне зависимости от времени их дальнейшего обучения». Однако мы получили ряд примеров, когда это утверждение не подтверждалось. Подобные результаты, в частности, содержатся в магистерской диссертации И. А. Бирюка (без возможности получения доступа к ее электронной версии)[1]. В этой работе описаны примеры взаимодействия двух TPM, размещенных на разных, связанных компьютерной сетью машинах. В данном контексте можно лишь сослаться на некоторые из доступных работ упомянутого автора в рассматриваемой предметной области [106, 113].

Упомянутые примеры показывают, что после наступления состояния синхронизации между $\langle \mathrm{TPM}(K,N,L)^A \rangle$ и $\langle \mathrm{TPM}(K,N,L)^B \rangle$ после дополнительных циклов (от нескольких до нескольких десятков) сети рассинхронизируются. Одной из вероятных причин такого явления может быть влияние помех на канал, связывающий TPM.

3.5. Детализация процесса синхронизации двух TPM

Результаты исследований, представленные в этом разделе, касаются важных особенностей процесса синхронизации сетей TPM. Мы старались при этом избегать увязки самого явления синхронизации с криптографией в расчете на то, что это поможет выявить и проанализировать новые взаимосвязи между параметрами сетей и ходом процесса определения согласованных весов. Основные сведения, составляющие содержание раздела, можно найти также в [42, 79–81, 89, 90, 92, 102, 103].

3.5.1. Шаги притягивающие и неотталкивающие

В п. 3.4.1 охарактеризованы содержание и результаты выполнения операций в ходе основных шагов (циклов) процесса синхронизации сетей $\langle \mathrm{TPM}(K,N,L) \rangle$. К таким шагам относятся: тихий, притягивающий и отталкивающий.

[1] Бирюк И. А. Моделирование и анализ устойчивости процесса синхронизации искусственных нейронных сетей: дис. … магистра техн. наук: 1-40 80 02. Минск, 2017. 98 л.

Ниже (подгл. 3.9) будут проанализированы важнейшие аспекты, касающиеся безопасности рассматриваемой технологии, которая определяется эффективностью атак третьей стороны (интруза – сети E или $\langle \mathrm{TPM}(K,N,L)^E \rangle$, см. рис. 3.5) с целью синхронизации «своих» весовых коэффициентов с весами сетей A и B. Для лучшего понимания особенностей притягивающих и неотталкивающих шагов нам нужно сейчас на базовом уровне проанализировать некоторые моменты взаимодействия сетей $\langle \mathrm{TPM}(K,N,L)^A \rangle$, $\langle \mathrm{TPM}(K,N,L)^B \rangle$ и $\langle \mathrm{TPM}(K,N,L)^E \rangle$.

Самый простой способ атаки[2] основывается на том, что оппонент знает архитектуру обеих сетей, а также всю информацию, которой обмениваются между собой сети. Как показывают тесты, через определенный промежуток времени, t_{learn}, сеть оппонента достигнет состояния синхронизации с наблюдаемыми сетями. Но время обучения t_{learn} гораздо больше, чем время t_{synch} синхронизации сетей A и B:

$$t_{\mathrm{learn}} > t_{\mathrm{synch}}. \tag{3.30}$$

Знак неравенства в выражении (3.30) имеет решающее значение для безопасности всего протокола. Сети A и B, принадлежащие доверенным коммуникационным сторонам, изменяют свои веса тогда и только тогда, когда они дают согласованные результаты. Владелец сети E, подслушивающий общение, не может всегда выбирать требуемый момент для изменения «своих» весов, чтобы усилить соответствие с атакуемой сетью. Неравенство (3.30) дополнительно связано с тем, что злоумышленник не может сосредоточиться на одном искомом состоянии (A или B), а должен выяснить, каким образом меняются пары весов атакуемых сетей.

Для количественной оценки уровня безопасности процесса синхронизации сетей A и B можно использовать параметр (коэффициент) $r = t_{\mathrm{learn}}/t_{\mathrm{synch}}$ [80]. Среднее время обучения (оппонента E с наблюдаемыми сетями), время синхронизации (наблюдаемых сетей), а также величина коэффициента r представлены в сравнительной таблице (табл. 3.10) на основе выполненных экспериментов.

Как видно из таблицы, для параметра $L = 1$ величина коэффициента r может упасть даже ниже 1. Это означает, что величина

[2] Более подробное рассмотрение атак см. в подгл. 3.9.

времени обучения t_{learn} будет меньше, чем время синхронизации t_{synch}, следовательно, параметр $L = 1$ не соответствует величине, гарантирующей даже минимальный уровень безопасности. Кроме того, очевидно, что вместе с ростом параметра L растет величина коэффициента r, что говорит о повышении уровня безопасности алгоритма согласования ключевой информации.

Таблица 3.10

Среднее время обучения и синхронизации, коэффициент безопасности для проанализированных 2000 опытов в зависимости от параметра *L*

L	t_{synch}	t_{learn}	r
1	61–10	$1{,}1\cdot10^{2}$–$0{,}2\cdot10^{2}$	1,8–0,6
2	188–26	$1{,}5\cdot10^{3}$–$0{,}5\cdot10^{3}$	8,0–2,9
3	376–51	$4{,}5\cdot10^{4}$–$1{,}3\cdot10^{4}$	120–51
4	673–95	$6{,}9\cdot10^{7}$–$5{,}7\cdot10^{7}$	$1{,}04\cdot10^{5}$–$1{,}02\cdot10^{5}$

В контексте использования временно́го фактора для оценки процесса синхронизации сетей интересными представляются результаты, изложенные в статье [101]. В работе приводится анализ взаимодействия сетей TPM (отдельных компьютеров с параметрами: core2 duo T6400, 3Gb ddr3)), обменивающихся информацией на основе клиент/серверной архитектуры. При этом зафиксированное среднее время синхронизации при общей длина сформированного совместного ключа (*KN*) длиной 48 символов ($L \in [-3, 3]$) составило 22,18 секунды (с); при 66-символьном ключе – 28,40 с, при 132-символьном – 40,30 с. Кроме того, после наступления состояния синхронизации весовых коэффициентов на каждом последующем шаге сети *A* и *B* генерировали новые, но одинаковые ключи.

Для обоснования неравенства (3.30) использовался такой параметр, как вероятность совершения притягивающего шага в зависимости от вероятности получения согласованных результатов нейронами первого слоя сетей *A* и *B* [35]. Этот подход мы проанализировали выше, в п. 3.4.1. Формула (3.28) выражает точнее вероятность наступления неотталкивающего шага, который авторы [35] назвали вероятностью притягивающего шага. Авторы упомянутой публикации не анализировали возникновение тихих шагов при синхронизации, которые не изменяют веса. Ниже представлены результаты вероятностного анализа с учетом тихих шагов.

В табл. 3.11 показаны все возможные комбинации, соответствующие выходным сигналам первого слоя TPM, состоящего из

двух нейронов ($K = 2$), и выходным сигналам сетей для двух синхронизирующих TPM. Кроме того, были определены типы шагов обучения. Таблица содержит только выборку случаев, в которых обе сети возвращают согласованные результаты ($\tau^A = \tau^B$). Три столбца таблицы (с 7-го по 9-й – «Равенство») содержат оценку равенства выходов нейронов первого слоя и выходов всей сети, где «1» означает равенство выходов, а «0» – разные значения. В следующих двух столбцах («Вероятность») приведены вероятности наступления события, соответствующего указанному равенству для выходов нейронов первого слоя, которые определяют формулу из следующего столбца. Вероятность получения согласованного результата указана в столбце «Выражение» и рассчитывается в соответствии с приведенным в п. 3.4.1 выражением:

$$p_i = P\left[\sigma_i^A = \sigma_i^B\right]. \qquad (3.31)$$

Таблица 3.11

Возможные состояния сигналов и соответствующие им вероятности при синхронизации сетей при $K = 2$

Сеть A			Сеть B			Равенство			Вероятность		Выражение	Тип шага	
σ_1	σ_2	τ^A	σ_1	σ_2	τ^B	σ_1	σ_2	τ	1	2		1	2
–1	–1	1	–1	–1	1	1	1	1	p	p	p^2	Q	Q
–1	–1	1	1	1	1	0	0	1	$1-p$	$1-p$	$(1-p)^2$	R	R
–1	1	–1	–1	1	–1	1	1	1	p	p	p^2	A	Q
–1	1	–1	1	–1	–1	0	0	1	$1-p$	$1-p$	$(1-p)^2$	R	R
1	–1	–1	–1	1	–1	0	0	1	$1-p$	$1-p$	$(1-p)^2$	R	R
1	–1	–1	1	–1	–1	1	1	1	p	p	p^2	Q	A
1	1	1	–1	–1	1	0	0	1	$1-p$	$1-p$	$(1-p)^2$	R	R
1	1	1	1	1	1	1	1	1	p	p	p^2	A	A

Последние два столбца содержат описание типов шагов для каждого нейрона: A – притягивающий, R – отталкивающий и Q – тихий.

Для каждого нейрона имеются два набора притягивающих шагов – с вероятностью p^2, два набора тихих шагов – с вероятностью p^2 и четыре отталкивающих – с вероятностью $(1-p)^2$. Таким образом, можно определить вероятность наступления неотталкивающего шага, которая будет соответствовать выражению (3.28). Однако для определения вероятности совершения строго притягивающего шага следует исключить из анализа тихие шаги, что, следовательно, приведет к следующей формуле:

$$\overline{p_a} = \frac{p^2}{2[p^2 + (1-p)^2]}. \qquad (3.32)$$

Сравнение вероятностей по выражениям (3.28) и (3.32) приводит к выводу, что строго притягивающие шаги могут встречаться вдвое чаще, чем неотталкивающие, поэтому справедливо выражение

$$\overline{p_a} = \frac{1}{2} p_a. \qquad (3.33)$$

Зависимость обеих этих вероятностей от вероятности p наступления ситуации, при которой результаты работы анализируемой пары нейронов первого слоя TPM оказываются одинаковыми, показана на рис. 3.13: кривая 1 – неотталкивающий шаг, 2 – притягивающий шаг.

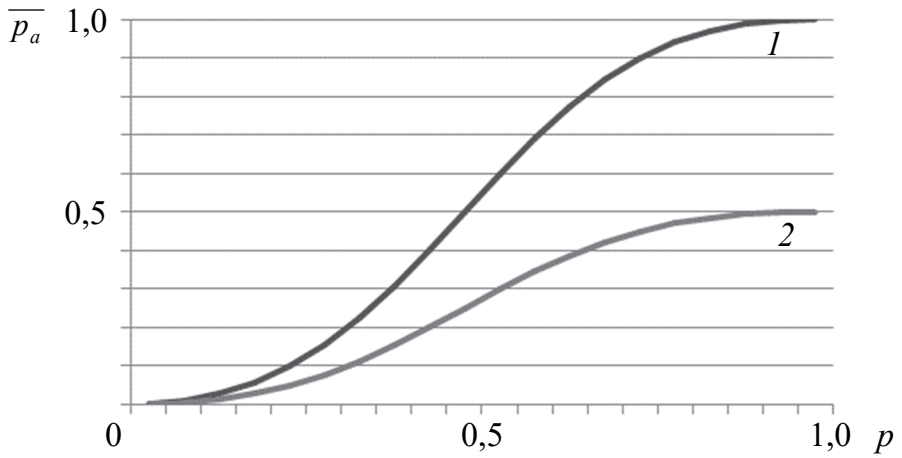

Рис. 3.13. Изменение вероятностей наступления неотталкивающего (*1*) и притягивающего (*2*) шагов синхронизации TPM

Перенося приведенные рассуждения на сети с бóльшим количеством нейронов, можно получить аналогичные зависимости. Пример таких зависимостей, выраженных количеством шагов $I(\cdot)$ разных типов и соответствующих вероятностей, приведен в табл. 3.12.

С помощью табл. 3.12 можно определить вероятности наступления неотталкивающих и только согласованных шагов в зависимости от вероятности p. Соответствующие выражения для сетей с разным количеством скрытых нейронов представлены в табл. 3.13. Согласно выражению (3.33) вероятность совершения строго притягивающего шага составляет половину вероятности совершения неотталкивающего шага, поэтому для вычисления этой вероятности полученные результаты следует разделить на 2.

Таблица 3.12

Количество и вероятности для разных типов шагов обучения для сетей с разным количеством нейронов (*K*) в первом слое

K	Шаги типа A		Шаги типа Q		Шаги типа R	
	Число, *I*(A)	Вероятность	Число, *I*(Q)	Вероятность	Число, *I*(R)	Вероятность
2	2	p^2	2	p^2	4	$(1-p)^2$
3	4	p^3	4	p^3	16	$p(1-p)^2$
	4	$p(1-p)^2$	4	$p(1-p)^2$	–	–
4	8	p^4	8	p^4	48	$p^2(1-p)^2$
	24	$p^2(1-p)^2$	24	$p^2(1-p)^2$	16	$(1-p)^4$
5	16	p^5	16	p^5	128	$p^3(1-p)^2$
	96	$p^3(1-p)^2$	96	$p^3(1-p)^2$	128	$p(1-p)^4$
	16	$p(1-p)^4$	16	$p(1-p)^4$	–	–
6	32	p^6	32	p^6	320	$p^4(1-p)^2$
	320	$p^4(1-p)^2$	320	$p^4(1-p)^2$	640	$p^2(1-p)^4$
	160	$p^2(1-p)^4$	160	$p^2(1-p)^4$	64	$(1-p)^6$

Таблица 3.13

Формулы для определения вероятности наступления неотталкивающего шага синхронизации TPM с разным количеством скрытых нейронов (*K*)

K	Выражение для расчета p_a
2	$\dfrac{p^2}{p^2+(1-p)^2}$
3	$\dfrac{p^3+p(1-p)^2}{p^3+3p(1-p)^2} = 1-\dfrac{2p(1-p)^2}{p^3+3p(1-p)^2}$
4	$\dfrac{p^4+3p^2(1-p)^2}{p^4+6p^2(1-p)^2+(1-p)^4}$
5	$\dfrac{p^5+6p^3(1-p)^2+p(1-p)^4}{p^5+10p^3(1-p)^2+5p(1-p)^4}$
6	$\dfrac{p^6+10p^4(1-p)^2+5p^2(1-p)^4}{p^6+15p^4(1-p)^2+15p^2(1-p)^4+(1-p)^6}$

Для обычно анализируемого различными авторами случая при $K = 3$ соответствующая формула представлена в двух вариантах.

По формулам из приведенной таблицы построены графики, показывающие зависимость вероятности совершения неотталкивающих шагов от вероятности получения одинаковых результатов соответствующими нейронами обеих сетей (см. рис. 3.14).

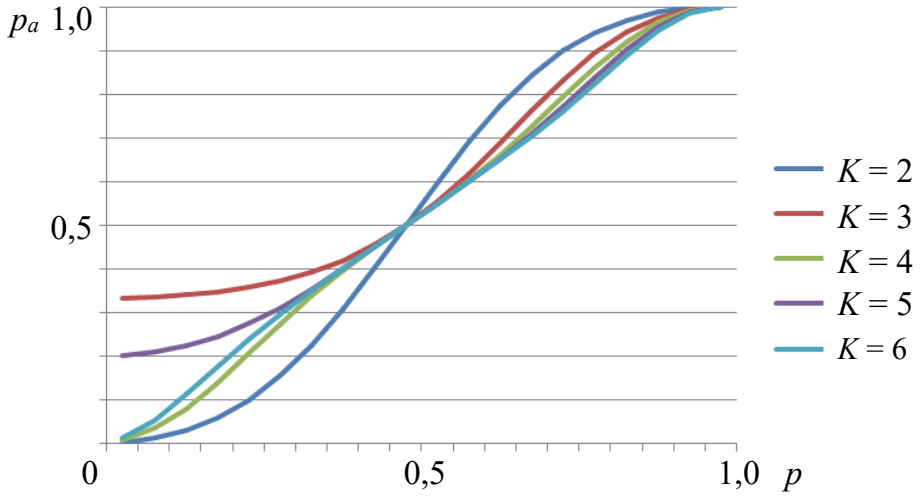

Рис. 3.14. Сравнение вероятности выполнения неотталкивающего шага для сетей TPM с разным количеством K нейронов в первом слое

Количество возможных вариантов наступления тихих и притягивающих шагов одинаково для каждого параметра K. Исключая из анализируемой формулы тихие шаги, согласно зависимости между выражениями (3.31) и (3.28), полученные результаты следует дополнительно разделить на 2. График вероятности наступления притягивающего шага можно получить, используя простое аффинное преобразование относительно оси OX со шкалой 1/2: $(x, y) \mapsto (1/2)x, y$; обозначим ее $P_{OX}^{k=1/2}$. Для примера на рис. 3.15 показаны графики изменения вероятностей наступления неотталкивающих (непрерывная линия) и притягивающих (штриховая линия) шагов для сетей $\langle \mathrm{TPM}(K = 2, N, L)^{A/B} \rangle$ и $\langle \mathrm{TPM}(K = 3, N, L)^{A/B} \rangle$.

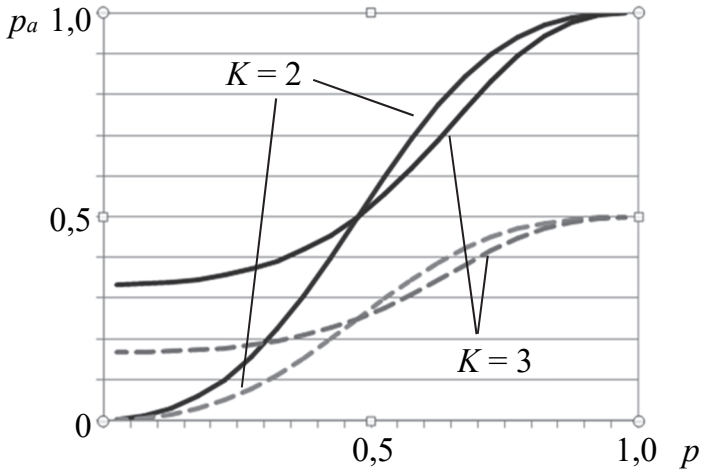

Рис. 3.15. Вероятности наступления неотталкивающих и притягивающих шагов синхронизации сетей $\langle \mathrm{TPM}(K = 2, N, L)^{A/B} \rangle$ и $\langle \mathrm{TPM}(K = 3, N, L)^{A/B} \rangle$

Следует различать строго притягивающие шаги и неотталкивающие, тем самым допуская возникновение притягивающих и тихих шагов. Соответствие выходных сигналов нейронов скрытого слоя сетей оказывает непосредственное влияние на выполнение нейронами синхронизируемых сетей шагов, не отталкивающих или притягивающих векторы весов.

Из представленных выше графиков следует, что сети с четным и нечетным числом нейронов скрытого слоя ведут себя по-разному. Анализируя рис. 3.14, видим, что для сети с четным числом скрытых нейронов и низкой вероятностью получения согласованных результатов ($p < 0{,}5$) вероятность выполнения неотталкивающего или притягивающего шага ниже вероятности получения таких согласованных результатов. Если такие сети перед началом процесса синхронизации «выбирают» несогласованные весовые векторы, то фаза сближения этих весовых векторов для двух TPM может длиться относительно долго.

Ситуация для сетей с нечетным числом скрытых нейронов выглядит несколько иначе. В этом случае вероятность неотталкивающего шага даже больше или равна вероятности получения согласованных результатов при $p = 0{,}5$. Отбрасывая тривиальный случай сети с одним скрытым нейроном, отмечаем, что, в частности, сети $\langle \text{TPM}(K = 3, N, L)^{A/B} \rangle$ имеют наибольшую вероятность синхронизации весов в зависимости от согласованности результатов работы нейронов скрытого слоя. С другой стороны, для всех сетей точка с абсциссой $x = 0{,}5$ является точкой изменения характера кривых, отражающих изменение вероятности того, что сети сделают правильный, т. е. сближающий секторы весов шаг.

3.5.2. Частота наступления притягивающих шагов

Скорость синхронизации TPM зависит от частоты наступления шагов притягивания и отталкивания векторов весов W^A и W^B. Возможные случаи, характеризующие определенные шаги для нескольких K, представлены в нижеследующих таблицах. При этом символ A соответствует притягивающему шагу, а R – отталкивающему. Здесь анализировалось количество шагов A – I(A) и R – I(R), но не учитывалось, какие именно нейроны выполняют шаг данного типа. Поэтому ссылка на нейрон выполнена безусловно (m, n, v, u), чтобы не предполагалась привязка к конкретной паре нейронов обеих

сетей. В следующих столбцах показаны возможные относительные результаты работы скрытого слоя (σ) и всей сети (τ).

Мы уже неоднократно отмечали, что при $K = 1$ результат работы сети такой же, как результат работы нейрона скрытого слоя, поэтому каждое изменение весов является притягивающим.

Типы возможных шагов для $\langle \mathrm{TPM}(K=2,N,L)^{A/B} \rangle$ представлены в табл. 3.14.

Таблица 3.14

Типы шагов для $\langle \mathrm{TPM}(K=2,N,L)^{A/B} \rangle$

Нейрон m	Нейрон n	σ	τ
A	A	+ – + –	– –
A	R	+ – + +	– +
R	R	+ – – +	– +

Первая и третья строки табл. 3.14 соответствуют шагам с модификацией векторов весов. Количество таких шагов, притягивающих, и количество шагов, отталкивающих весовые коэффициенты сетей, одинаково: $I(\mathrm{A}) = 2$ и $I(\mathrm{R}) = 2$, т. е. $I(\mathrm{A})/I(\mathrm{R}) = 1$.

Подобные результаты для $\langle \mathrm{TPM}(K=3,N,L)^{A/B} \rangle$ показаны в табл. 3.15. Эти данные согласуются с результатами анализа, представленного в [30, 78].

Таблица 3.15

Типы шагов для $\langle \mathrm{TPM}(K=3,N,L)^{A/B} \rangle$

Нейрон m	Нейрон n	Нейрон v	σ	τ
A	A	A	+ – + + – +	– –
A	A	R	+ + – + + +	– +
A	R	R	+ + – + – +	– –
R	R	R	– – + + + –	+ –

Как видим, здесь $I(\mathrm{A})/I(\mathrm{R}) = 2$, так как $I(\mathrm{A}) = 4$, $I(\mathrm{R}) = 2$.

В табл. 3.16 содержатся данные для $\langle \mathrm{TPM}(K=4,N,L)^{A/B}\rangle$, в табл. 3.17 – для $\langle \mathrm{TPM}(K=5,N,L)^{A/B}\rangle$.

Таблица 3.16

Типы шагов для $\langle \mathrm{TPM}(K=4,N,L)^{A/B}\rangle$

Нейрон m	Нейрон n	Нейрон v	Нейрон u	σ	τ
A	A	A	A	+ – + + + – + +	– –
A	A	A	R	+ + + – + + + +	– +
A	A	R	R	+ + – + + + + –	– –
A	R	R	R	+ – + – + + – +	+ –
R	R	R	R	–+ – – + – + +	– –

Для $K = 4$ получилось $I(\mathrm{A}) = 6$, $I(\mathrm{R}) = 6$; $I(\mathrm{A})/I(\mathrm{R}) = 1$.

Таблица 3.17

Типы и число шагов для $\langle \mathrm{TPM}(K=5, N, L)^{A/B}\rangle$

I(A)	I(R)	Равенство τ^A и τ^B
5	0	+
4	1	–
3	2	+
2	3	–
1	4	+
0	5	–

Для $K = 5$ $I(\mathrm{A}) = 9$, $I(\mathrm{R}) = 6$; $I(\mathrm{A})/I(\mathrm{R}) = 1{,}5$.

Продолжая эту линию рассуждений, аналогичные таблицы могут быть построены для более крупных сетей, при $K > 5$. В анализе существует закономерность, согласно которой для четных K отношение $I(\mathrm{A})/I(\mathrm{R}) = 1$. Для нечетных K соотношение $I(\mathrm{A})/I(\mathrm{R})$ имеет иной характер, что видно из табл. 3.15 и 3.17, а также табл. 3.18.

Таблица 3.18

Соотношение притягивающих и отталкивающих шагов синхронизации TPM для нечетных K

K	3	5	7	…	$2k+1$
$I(\mathrm{A})/I(\mathrm{R})$	$2 = 1+\frac{1}{1}$	$1\frac{1}{2} = 1+\frac{1}{2}$	$1\frac{1}{3} = 1+\frac{1}{3}$	…	$1+\frac{1}{k}$

Графическое отображение изменения зависимости $I(\text{A})/I(\text{R})$ от K (нечетные числа) представлено на рис. 3.16.

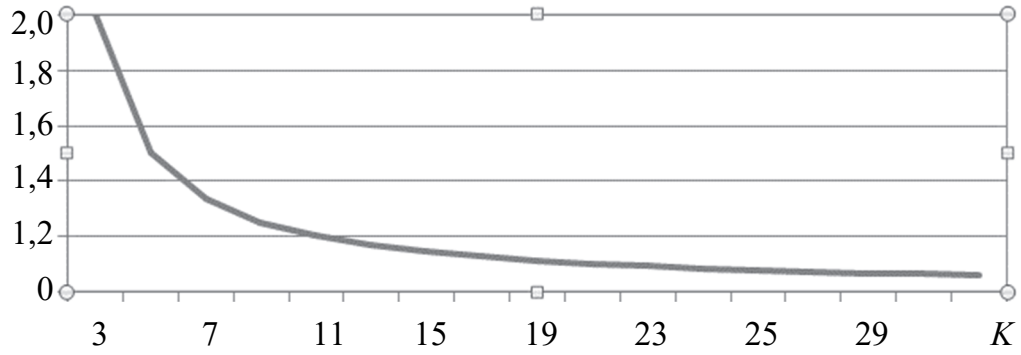

Рис. 3.16. Зависимость $I(\text{A})/I(\text{R})$ от K

3.5.3. Характер изменения расстояния между векторами весов

Для анализа будем использовать ранее принятые обозначения: w_{ij}^A и w_{ij}^B – вес j-го сигнала в i-м нейроне соответственно сети A и сети B; $1 \le i \le K$, $1 \le j \le N$. Начальные значения векторов весов W^A и W^B случайны, поэтому может случиться, что существуют индексы i и j такие, что $w_{ij}^A = w_{ij}^B$. Последнее означает, что сгенерированы равные значения соответствующих весов обеих сетей: $W^A = W^B$. Обычно $W^A \neq W^B$, т. е. $w_{ij}^A \neq w_{ij}^B$.

При взаимном обучении сетей TPM могут возникать ситуации, когда веса соответствующих нейронов обеих сетей изменяются в одном направлении, либо не изменяются, либо изменяются веса только одной из них. Эти ситуации соответствуют притягивающему (A), тихому (Q) и отталкивающему (R) шагам соответственно [42, 111].

Веса изменяются при условии $\tau^A = \tau^B$. Если выходы соответствующих нейронов обеих сетей равны выходным сигналам всей («своей») сети $(\sigma_i^A = \tau^A$ и $\sigma_i^B = \tau^B;\ 1 \le i \le K)$, то векторы весов этих нейронов изменяются в одном направлении – шаг типа A. Поскольку веса изменяются в одном направлении, расстояние между ними не изменится, если только один из них не выйдет за границы $[-L, L]$, а другой не примет любое иное значение. В данном конкретном случае, действительно, расстояние между этими весами будет уменьшаться. Для конкретного интервала $[-L, L]$, которому принадлежат веса $w_{ij}^{A/B}$, существует $(2L + 1)$ возможных комбинаций значений весов обеих сетей. Расстояние между двумя весами w_{ij}^A и w_{ij}^B может быть уменьшено, если выполняются условия:

1) перед процедурой модификации весов они разные: $w_{ij}^A \neq w_{ij}^B$; на шаге притяжения оба веса движутся в одном направлении, поэтому, если бы они были равны, их расстояние не изменилось бы;

2) перед процедурой модификации они удовлетворяют одному из следующих дополнительных условий:

а) один из весов равен $–L$, а соответствующий импульс возбуждения (x) равен –1;

б) один из весов равен L, а импульс возбуждения – 1.

Значение веса второй сети произвольное, но оно должно удовлетворять условию 1.

В результате применения правила обучения, урезающего значение элементов векторов весов до диапазона $[–L, L]$, веса, удовлетворяющие условию 2а или 2б, не изменятся, а вес второй сети приблизится к предельному значению, что уменьшит расстояние между этими весами.

Возбуждающие импульсы X рандомизированы с равной вероятностью, а это значит, что для всех i и j справедливо

$$P[x_{ij} = -1] = P[x_{ij} = 1] = \frac{1}{2}. \tag{3.34}$$

Среди $(2L + 1)^2$ возможных вариантов весов для каждого из случаев 2а и 2б имеется $4L$ вариантов, гарантирующих уменьшение расстояния между весами. Например, для $L = 2$ и $x = –1$ такими вариантами будут следующие пары весов: (–2, –1), (–2, 0), (–2, 1), (–2, 2), (–1, –2), (0, –2), (1, –2) и (2, –2). В начале синхронизации веса обеих сетей распределяются случайным образом, поэтому каждое значение равновероятно (см. равенство (3.34)). Это означает, что на шаге А вероятность уменьшения расстояния между одиночными значениями веса выражается формулой

$$p = \frac{4L}{(2L+1)^2}. \tag{3.35}$$

С увеличением значения параметра L вероятность уменьшения расстояния между весами снижается. Табл. 3.19 содержит анализируемые параметры для $L = 1, 2, \ldots, 20$.

Представленный анализ относится только к одному элементу вектора весов, но поскольку все веса рандомизированы независимо, этот анализ можно легко распространить на множественные значения, используя схему Бернулли [114] и соответствующую

вероятность. Более того, оказывается, что на отталкивающем шаге (R) можно еще и уменьшить расстояние между весами. Этот шаг наступает при следующих условиях: $\sigma_i^A = \tau^A$ и $\sigma_i^B \neq \tau^B$ или $\sigma_i^A = \tau^A$ и $\sigma_i^B = \tau^B$; $1 \le i \le K$. В такой ситуации веса нейрона с согласованным выходным результатом меняются, а соответствующие веса нейрона другой сети остаются неизменными.

Таблица 3.19

Вероятность (*p*) уменьшения расстояния между векторами весов на шаге притяжения

L	$4L$	$(2L+1)^2$	p
1	4	9	0,4444
2	8	25	0,3200
3	12	49	0,2449
4	16	81	0,1975
5	20	121	0,1653
10	40	441	0,0907
15	60	961	0,0624
20	80	1681	0,0476

Расстояния между w_{ij}^A и w_{ij}^B на шаге R могут уменьшаться, если:

1) перед модификацией весов они разные: $w_{ij}^A \neq w_{ij}^B$; на этом шаге один из весов меняется, а другой – остается прежним; если бы указанные элементы перед модификацией были бы одинаковы, то после модификации они должны были бы стать разными, что, конечно, увеличило бы расстояние между векторами весов;

2) не изменяя общности рассуждений, полагаем, что $\sigma_i^A = \tau^A$ и $\sigma_i^B \neq \tau^B$;это означает, что анализируемый нейрон сети *A* меняет свои веса, а веса соответствующего нейрона сети *B* остаются неизменными; для модификаций параметра веса должно дополнительно выполняться одно из следующих условий:

а) $w_{ij}^A > w_{ij}^B$ и соответствующий импульс возбуждения $x = -1$;

б) $w_{ij}^A < w_{ij}^B$ и $x = 1$.

Для произвольного L существует $(2L + 1)L$ вариантов весов, удовлетворяющих последним условиям 2а или 2б. Так как оба эти условия зависят от уровня входного сигнала, а он рандомизируется независимо для каждого веса, то вероятность сокращения расстояния между весами на шаге R выражается формулой

$$p = \frac{(2L+1)L}{(2L+1)^2} = \frac{L}{2L+1}. \qquad (3.36)$$

Зависимость вероятности (3.36) от L представлена в табл. 3.20.

Таблица 3.20

Вероятность (p) уменьшения расстояния между векторами весов на шаге отталкивания векторов весов

L	$(2L+1)L$	$(2L+1)^2$	p
1	3	9	0,3333
2	10	25	0,4000
3	21	49	0,4286
4	36	81	0,4444
5	55	21	0,4545
10	210	441	0,4762
15	465	961	0,4839
20	820	1681	0,4878

Как видно, и на отталкивающем шаге можно уменьшить расстояние между W^A и W^B и, как это ни парадоксально, при $L > 1$ оно даже более вероятно, чем на притягивающем шаге. В силу характера изменения весов обеих сетей шаги можно назвать *общими*, *связными* или *согласованными* вместо притягивающих, также *раздельными* или *разными* – вместо отталкивающих. С другой стороны, название тихого шага полностью отражает характер случая, когда веса не меняются. Как видно из рис. 3.17, увеличение параметра L снижает вероятность сближения весов за фиксированное число шагов рассмотренных типов, что удлиняет процесс синхронизации.

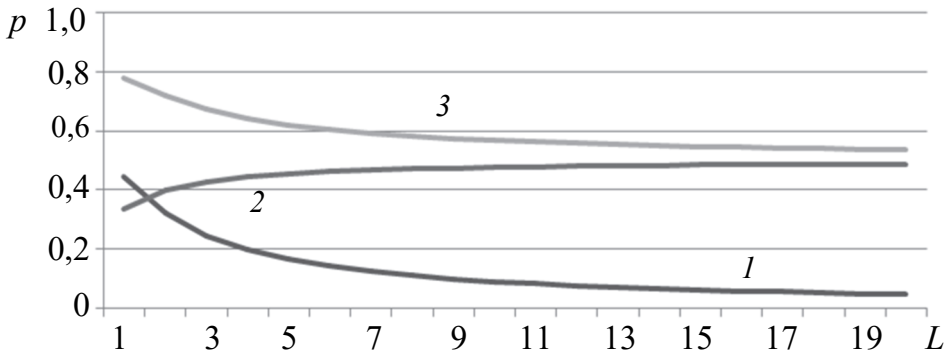

Рис. 3.17. Зависимость вероятности сближения весов сетей $\langle \mathrm{TPM}(K,N,L)^{A/B} \rangle$ при изменении параметра L

Однако важно, что это повышает устойчивость протокола согласования ключей между сетями $\langle \text{TPM}(K=2,N,L)^A \rangle$ и $\langle \text{TPM}(K=2,N,L)^B \rangle$ к попыткам атаки со стороны $\langle \text{TPM}(K=2,N,L)^E \rangle$ [42].

На рис. 3.17 кривая *1* соответствует выражению (3.35), кривая *2* – (3.36), кривая *3* – сумме вероятностей, вычисленных по (3.35) и (3.36).

3.5.4. Влияние параметров веса на распределение битов в формируемом криптографическом ключе

Параметр L, определяющий максимальное абсолютное значение каждого веса, не только влияет на вероятность уменьшения расстояния между весами сетей A и B, но и определяет количество битов, необходимых для двоичной формы записи веса, соответствующей формируемому криптографическому ключу, общему для A и B. Если значения весов, сгенерированные в ходе сетевой синхронизации, используются непосредственно в качестве ключей, то параметр L при разных кодировках отрицательных чисел влияет на распределение битов (1 и 0) в ключе.

Сетевые веса TPM, согласованные в результате синхронизации, можно использовать непосредственно в качестве значений криптографических ключей либо для генерации псевдослучайных чисел, которые будут использованы как ключи. В первом случае важно равномерно распределить биты 0 и 1 по всему ключу (стараемся получить ключ, характеризующийся максимальной энтропией).

В этой связи далее рассмотрены два случая:

1) веса смещены так, что они принадлежат диапазону от 0 до $2L$;

2) используется форма записи, позволяющая работать с отрицательными числами, характеризующими параметры весов.

Количество битов, необходимых для хранения значения веса, определяется как $\log_2(2L+1)$, но на самом деле нам нужно использовать $[\log_2(2L+1)]$, где внешние скобки означают использование функции целочисленных значений. Поэтому важно минимизировать разницу между этими значениями, т. е. выбрать параметр L, минимизирующий эту разницу [103]:

$$b(L) = [\log_2(2L+1)] - \log_2(2L+1). \qquad (3.37)$$

Рис. 3.18 показывает изменение функции $b(L)$ (3.37) для разных L. Как видно, наименьшее значение функции соответствует значениям L, максимально близким, но не превышающим степень двойки.

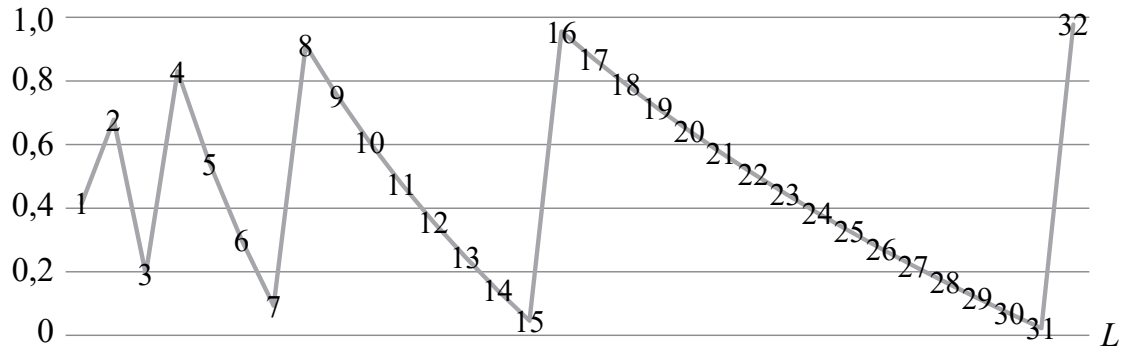

Рис. 3.18. Характер изменения функции $b(L)$

В приведенном выше случае 1 генерации и использования ключа (веса смещены так, что они принадлежат диапазону от 0 до $2L$) ключ должен быть целым числом от 0 до $2L$. Поэтому требовалось смещение весов, полученных в результате синхронизации, на $+L$. Это позволяет использовать $2L + 1$ состояний, которые представляются в двоичном коде. Для примера характер изменения некоторых из параметров отражен в табл. 3.21.

Таблица 3.21

Некоторые параметры весовых коэффициентов синхронизированных TPM

L	$2L + 1$	Количество битов в числе $2L + 1$	$b(L)$	Количество 0	Количество 1	$p(0)$	$p(1)$
1	3	2	0,41504	4	2	0,6667	0,3333
2	5	3	0,67807	10	5	0,6667	0,3333
3	7	3	0,19265	12	9	0,5714	0,4286
4	9	4	0,83007	23	13	0,6389	0,3611
5	11	4	0,54057	27	17	0,6136	0,3864
6	13	4	0,29956	30	22	0,5769	0,4231
7	15	4	0,09311	32	28	0,5333	0,4667
8	17	5	0,91254	52	33	0,6118	0,3882
9	19	5	0,75207	58	37	0,6105	0,3895
14	29	5	0,14202	78	67	0,5379	0,4621
15	31	5	0,04580	80	75	0,5161	0,4839
16	33	6	0,95561	117	81	0,5909	0,4091
17	35	6	0,87072	125	85	0,5952	0,4048
30	61	6	0,06926	190	176	0,5191	0,4809
31	63	6	0,02272	192	186	0,5079	0,4921
32	65	7	0,97763	262	193	0,5758	0,4242

Отметим, что произведение чисел во втором и в третьем столбцах строк таблицы равно сумме чисел в пятом и в шестом столбцах той же строки.

При таком кодировании вероятность бита «0» ($p(0)$) выше, особенно для небольших L, при этом вероятности $p(0)$ и $p(1)$ становятся ближе для L, равного $2^n - 1$ для некоторого n, что является прямым следствием минимизации функции (3.37). Несколько иное распределение ключевых битовых значений получается при использовании дополнительного представления для отрицательных весов ($-L$). Соответствующие результаты представлены в табл. 3.22.

Таблица 3.22

Некоторые параметры весовых коэффициентов синхронизированных TPM при учете отрицательных значений *L*

L	$2L + 1$	Количество битов в числе $2L + 1$	$b(L)$	Количество 0	Количество 1	$p(0)$	$p(1)$
1	3	2	0,41504	3	3	0,5000	0,5000
2	5	3	0,67807	8	7	0,5333	0,4667
3	7	3	0,19265	10	11	0,4762	0,5238
4	9	4	0,83007	19	17	0,5278	0,4722
5	11	4	0,54057	22	22	0,5000	0,5000
6	13	4	0,29956	26	26	0,5000	0,5000
7	15	4	0,09311	29	31	0,4833	0,5167
8	17	5	0,91254	44	41	0,5176	0,4824
9	19	5	0,75207	48	47	0,5053	0,4947
14	29	5	0,14202	72	73	0,4966	0,5034
15	31	5	0,04580	76	79	0,4903	0,5097
16	33	6	0,95561	101	97	0,5101	0,4899
17	35	6	0,87072	106	104	0,5048	0,4952
30	61	6	0,06926	182	184	0,4973	0,5027
31	63	6	0,02272	187	191	0,4947	0,5053
32	65	7	0,97763	230	225	0,5055	0,4945

При таком кодировании только для небольших значений L существуют минимальные различия в вероятностях $p(0)$ и $p(1)$.

При выборе значения параметра L для синхронизируемых сетей TPM целесообразно иметь в виду, что соотношение $L = 2^n - 1$ позволяет обеспечить относительно равномерное распределение битов в сгенерированном ключе, что особенно важно при синхронизации небольших сетей.

3.6. Статистический анализ параметров процесса синхронизации TPM

Измеряя время, необходимое для синхронизации сетей $\langle \text{TPM}(K=3, N=101, L=3)^{A/B} \rangle$, в циклах вместо обычных единиц времени, например секунд, можно сравнивать результаты моделирования (или симуляции), проведенного на компьютерах с разной вычислительной мощностью. Количество циклов, необходимых для достижения согласованных весовых векторов, не зависит от скорости, с которой они будут анализироваться на конкретной машине.

Рассмотрим в связи с этим результаты, полученные на основе обучения сетей по методу случайного блуждания (в силу обобщающего характера этого метода при $N \to \infty$ [96]). Каждая сеть имела доступ только к «своим» весам. Программа-симулятор имела доступ к векторам весов обеих сетей, что позволяло останавливать процесс обучения ровно в момент определения согласованных весов.

Гистограммы распределений количества TPM, синхронизировавшихся за определенное количество шагов, имеют схожий вид для сетей с разными топологиями, определяемыми параметрами *K*-*N*-*L* (см., например, рис. 3.6 и 3.7 на с. 68–69). Основное отличие заключается в количестве шагов, необходимых для синхронизации сетей. По форме гистограммы имеют близкое сходство с распределением Пуассона. Проверка этой гипотезы была показана на примере сети с количеством нейронов в скрытом слое $K = 3$ и количеством входных сигналов к каждому из них $N = 16$. Были проведены тысячи опытов для сетей с параметром $L \in \{1, 2, \ldots, 5, 10, 15, \ldots, 50\}$ [102, 103].

Время, необходимое для согласования весов сетей, зависит от начальных весов W^A и W^B и входных импульсов X^A и X^B. Общая картина, характеризующая распределение для 15 000 (n) опытов синхронизации сетей $\langle \text{TPM}(K=3, N=101, L=3)^{A/B} \rangle$, обученных по правилу случайного блуждания, понятна из рис. 3.19.

При некотором условном разделении результатов опытов на классы гистограммы для сетей с иной структурой выглядят примерно так же. Количество классов c определялось по формуле Д. Ханстбергера (D. Huntsberger) [115, 116]:

$$c = 1 + 3{,}32 \log n, \tag{3.38}$$

где n – количество опытов. Формула (3.38) определяет количество интервалов, на которые разбиты результаты экспериментов. Полученное число было удвоено для лучшей читаемости диаграммы.

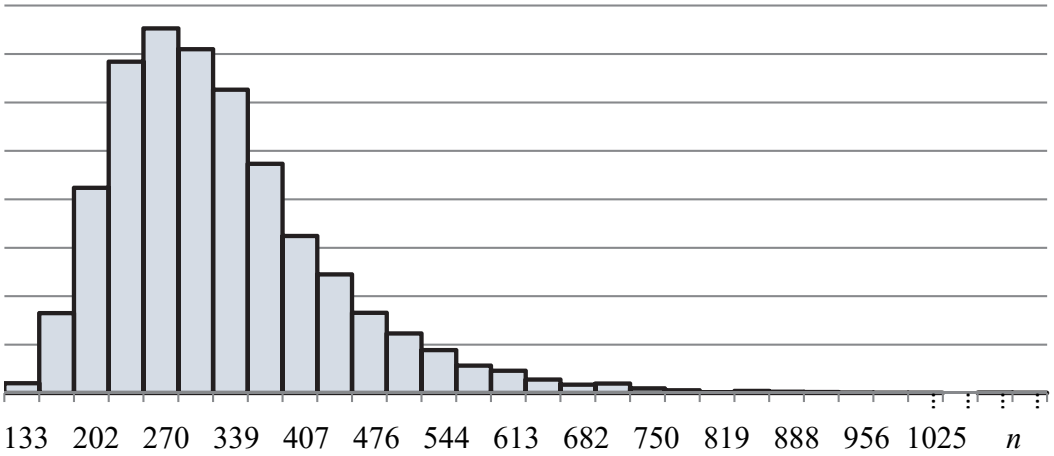

Рис. 3.19. Гистограмма распределения количества сетей по числу шагов до наступления состояния синхронизации

В работе [95] в результате подобного исследования для сетей $\langle \text{TPM}(K=3, N=101, L=3)^{A/B} \rangle$ получено распределение, показанное на рис. 3.20.

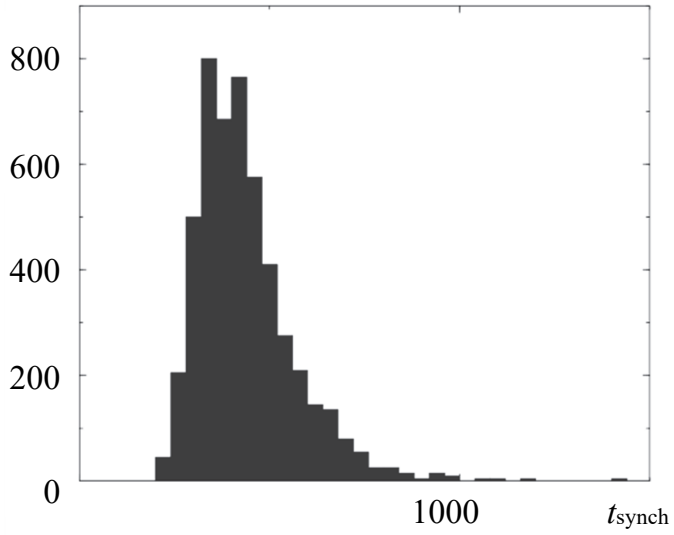

Рис. 3.20. Распределение TPM по числу шагов до наступления синхронизации [95]

Оно достигает пика при выполнении примерно 400 шагов синхронизации ($t_{\text{synch}} \approx 400$). При этом установлено, что среднее время

синхронизации практически не зависит от размера N сетей, по крайней мере до $N = 10\,000$. Распределение на рис. 3.20 также визуально можно отнести к пуассоновскому.

Наконец, табл. 3.23 содержит данные о количестве шагов самой короткой («Число шагов, мин.»), самой длинной («Число шагов, макс.») наблюдаемой синхронизации и среднее значение («Число шагов, средн.»). Представленные гистограммы сохраняют характерную форму, но нужно отметить, что интервалы (по обеим осям) для разных топологий сетей имеют разные предельные значения.

Таблица 3.23

Данные о продолжительности процесса синхронизации сетей $\langle \mathrm{TPM}(K, N, L)^{A/B} \rangle$

K-N-L	Число шагов, мин.	Число шагов, макс.	Число шагов, средн.
3-16-1	7	132	33,4
3-16-2	37	394	118,3
3-16-3	80	702	264,2
3-16-4	160	1469	469,7
3-16-5	263	2849	755,6
3-16-10	1159	8579	3239,3
3-16-15	2991	20 263	7657,9
3-16-20	5044	46 057	14 444,6
3-16-25	8242	62 486	23 033,2
3-16-30	14 520	92 569	34 104,3
3-16-35	19 209	124 496	48 443,2
3-16-40	23 919	161 440	64 018,9
3-16-45	26 191	251 386	82 656,0
3-16-50	42 856	288 219	102 382,7

Построение нескольких гистограмм на одной оси с использованием одних и тех же предельных значений можно увидеть также на рис. 3.6 (с. 68).

На основании этих данных определялись границы (размах) каждой выборки, количество интервалов для построения гистограммы, ширина и размер каждого интервала. Путем деления размера интервала на размер выборки были определены эмпирические вероятности наступления состояния синхронизации сетей TPM в каждом временном интервале. Например, для сетей $\langle \mathrm{TPM}(K = 4, N = 40, L = 5)^{A/B} \rangle$ эти результаты выглядят следующим образом:

– объем выборки n = 1000;
– диапазон выборки – 357;
– количество классов (диапазонов) c =11;
– ширина интервала – 32,6.

Интервалы, необходимые для построения гистограммы, определялись по формуле (3.38). Обобщенные данные для указанных ТРМ представлены в табл. 3.24.

Таблица 3.24

Характеристики синхронизации сетей $\langle \mathbf{TPM}(K=3, N=16, L=2)^{A/B} \rangle$

c_i	Границы диапазона		Количество	Вероятность
1	37,0	69,6	96	0,096
2	69,6	102,1	338	0,338
3	102,1	134,7	292	0,292
4	134,7	167,3	137	0,137
5	167,3	199,9	75	0,075
6	199,9	232,4	38	0,038
7	232,4	265,0	12	0,012
8	265,0	297,6	8	0,008
9	297,6	330,2	2	0,002
10	330,2	362,7	1	0,001
11	362,7	395,3	1	0,001

Одна из гипотез, которых мы придерживались, состояла в том, что времена синхронизации сетей ТРМ подчиняются упомянутому распределению Пуассона с функцией вероятности, задаваемой формулой

$$P[X = c] = \frac{\zeta^c e^{-\zeta}}{c!}.$$

Параметр ζ – средневзвешенное значение, определялся эмпирически и принят равным ζ = 2,979. Гипотеза проверялась тестом на совместимость с χ^2 путем вычисления суммы

$$\chi^2 = \sum_{i=1}^{r} \frac{(O_i - E_i)^2}{E_i}. \qquad (3.39)$$

В выражение (3.39) входят классические параметры: O_i, E_i – соответственно результаты опыта и рассчитанные согласно методу χ^2 данные для каждого из r диапазонов.

В соответствии с критерием χ^2 [117] размер каждого класса должен быть не менее 8, поэтому диапазоны 8–11 из табл. 3.24 были объединены для создания диапазона размером 12 и с вероятностью 0,012. В итоге количество диапазонов равно 8 ($r = 8$), а сумма равна $\chi^2 \approx 0{,}12$. Из таблицы распределения χ^2 было рассчитано критическое значение для 7 степеней свободы и для уровня значимости 0,999; оно равно 0,59849, а так как полученная сумма $0{,}12 < 0{,}59849$, то оснований отвергнуть гипотезу нет. Следовательно, анализируемую статистику можно классифицировать как подчиняющуюся распределению Пуассона.

Подобный анализ был проведен [102] для остальных сетей из табл. 3.24, и в табл. 3.25 показаны результаты этого анализа.

Таблица 3.25

Параметры распределения Пуассона и результаты применения критерия χ^2

K-N-L	Число шагов, мин.	Ширина диапазона	Число диапазонов	ζ	χ^2	Критическое значение
3-16-1	7	11,4	7	2,81	0,13	0,381
3-16-2	37	32,6	8	2,98	0,12	0,598
3-16-3	80	56,8	10	3,74	0,09	0,152
3-16-4	160	119,4	9	3,09	0,08	0,857
3-16-5	263	236	7	2,58	0,13	0,381
3-16-10	1159	677	9	3,57	0,06	0,857
3-16-15	2991	1576	9	3,47	0,04	0,857
3-16-20	5044	3742,1	8	3,00	0,13	0,598
3-16-25	8242	4949,3	10	3,48	0,08	1,152
3-16-30	14 520	7121,3	9	3,24	0,07	0,857
3-16-35	19 209	9606,5	10	3,55	0,09	1,152
3-16-40	23 919	12 547,5	9	3,70	0,09	0,857
3-16-45	26 191	20 547	8	3,27	0,15	0,598
3-16-50	42 856	22 387,1	8	3,16	0,08	0,598

Для каждой из анализируемых сетей указано минимальное наблюдаемое количество шагов, необходимых для синхронизации сетей, примерная ширина интервала, количество интервалов, используемых для теста χ^2. Что касается примера, описанного выше, то последние диапазоны были объединены в одну группу для выполнения требования минимального количества элементов группы в тесте χ^2, отсюда и некоторые колебания их количества для разных

сетей. В следующих столбцах показано эмпирически определенное среднее значение, которое использовалось в качестве параметра ζ в распределении Пуассона, сумма χ^2 (3.39) и его критическое значение для уровня значимости 0,999. Полученное суммарное значение ниже значения распределения χ^2 для всех табличных уровней значимости, поэтому в табл. 3.25 внесены значения наиболее ограничительного уровня значимости.

Используя данные, содержащиеся в табл. 3.25, можно ответить на вопрос о вероятности синхронизации TPM с заданной структурой за заданный период времени. Например, пусть сеть имеет параметры 3-16-4. Для расчета вероятности того, что сеть синхронизируется через 600 циклов обучения, нужно проверить, к какому интервалу относится данное время синхронизации, используя кратчайшее время и соответствующий размер интервала. Первый интервал – [160, 274], а четвертый – [519, 637] – содержит число 600, которое является искомой продолжительностью процесса синхронизации. Поскольку в таблице ширина интервалов, полученная в результате разделения выборки, указана с десятичными долями, границы интервалов округлены до целых, поскольку нельзя говорить о части цикла обучения сети. Далее, пользуясь таблицей распределения Пуассона, для $c = 4$ и $\zeta = 3{,}09$ получаем вероятность, равную 0,1728. Можно также рассчитать значение для $c \leq 4$, ответив на вопрос о вероятности синхронизации сетей за время, составляющее до 600 шагов, или, используя вероятность обратного события, рассчитать вероятность наступления синхронизации сетей за большее количество шагов.

3.7. Анализ условий окончания процесса синхронизации сетей

Достижение состояния синхронизации сетей основано на использовании целых чисел в качестве весовых коэффициентов обеих сетей, принадлежащих ограниченному диапазону $[-L, L]$, и соответствующим образом измененных правил обучения сетей. При модификации весов их значения усекаются, чтобы они не выходили за пределы установленного диапазона. Синхронизированные сети также возвращают согласованный результат для одного и того же входного

вектора. Дальнейшее обучение сетей $\langle \mathrm{TPM}(K,N,L)^{A/B} \rangle$ приводит к последующим изменениям их весов. Мы отмечали выше, что такие изменения не должны влиять на синхронизированное состояние.

Используя явление синхронизации в криптографии для определения криптографических ключей, необходимо скрыть весовые векторы обеих сетей. Абоненты (A и B) не могут легко определить, достигнуто ли состояние синхронизации. Вместе с тем потенциальному злоумышленнику нужно синхронизировать свою сеть с сетями A и B (см. рис. 3.5 на с. 57), что также непросто. Понятно, что чем дольше длится процесс синхронизации весовых коэффициентов сетей $\langle \mathrm{TPM}(K,N,L)^{A/B} \rangle$, тем больше шансов у злоумышленника.

Таким образом, сети $\langle \mathrm{TPM}(K,N,L)^{A/B} \rangle$ должны иметь некоторый статистический анализ времени синхронизации с выбранными параметрами. Уменьшение времени синхронизации усиливает протокол согласования ключей, а также сокращает время, доступное потенциальному злоумышленнику.

В настоящем подразделе проанализированы некоторые важные особенности, относящиеся к определению момента завершения синхронизации сетей и влияющие на безопасность этого процесса.

3.7.1. Статистическая оценка распределения времени синхронизации сетей на основе квартилей

Проанализируем использование *квартилей* для оценки времени, необходимого для достижения сетевой синхронизации. В нашем случае квартили [114, 117] – это предельные значения, определяемые на основе наблюдений за временами синхронизации. Нулевой квартиль определяет самую короткую по времени наблюдаемую синхронизацию; первый – значение, ниже которого попадают 25% наблюдаемых времен; второй квартиль определяет граничное значение, ниже которого попадает половина времен синхронизации. Особенно интересен третий квартиль, который определяет время, необходимое для синхронизации 75% сети. В результате моделирования оказывается, что она близка к половине самой продолжительной наблюдаемой синхронизации. Это означает, что 3/4 сетей синхронизируется за половину времени наихудшей наблюдаемой синхронизации. Такая оценка дает основание для прерывания длительных синхронизаций с использованием известных длин наихудших (по продолжительности) синхронизаций для сетей с заданной топологией.

Анализ времени сетевой синхронизации [81, 89, 90, 92, 102, 103] проводился на модели, имитирующей работу двух ТРМ, работающих на одном компьютере. Сети обменивались только выходными импульсами. Ни одна из сетей не имела доступа к весовому вектору другой сети. Программа-симулятор после каждого шага синхронизации проверяла, идентичны ли векторы весов.

Моделирование проводилось для сетей $\langle \text{TPM}(K=3, N=11, L=2)^{A/B} \rangle$ и $\langle \text{TPM}(K=3, N=11, L)^{A/B} \rangle$, где L = 1, 2, …, 10. Сети обучались по правилу случайного блуждания по 2000 раз для каждой топологии. Обработанные статистические результаты для $\langle \text{TPM}(K=3, N=11, L=2)^{A/B} \rangle$ представлены в табл. 3.26. Таблица содержит количество циклов самой короткой и самой продолжительной наблюдаемой синхронизации, диапазон тестируемой выборки, значения квартилей и среднее количество циклов синхронизации.

Таблица 3.26

Статистические характеристики синхронизации сетей $\langle \text{TPM}(K=3, N=11, L=2)^{A/B} \rangle$

Параметр	Символ	Точное значение
Количество опытов	n	2000
Минимальное время синхронизации, циклов	*MIN_ts*	26
Максимальное время синхронизации, циклов	*MAX_ts*	318
Диапазон	*R_ts*	292
Квартиль 0	$Q0$	26
Квартиль 1	$Q1$	76
Квартиль 2	$Q2$	98
Квартиль 3	$Q3$	128,25
Квартиль 4	$Q4$	318
Среднее значение	*AVG*	106,467

На основе табл. 3.26 рассчитаны дополнительные характеристики (табл. 3.27) и построена гистограмма распределения времен (рис. 3.21). Здесь используются следующие расчетные параметры: количество классов Ханстбергера – 12 (точное значение – 11,959), ширина диапазона – 25 (точное значение – 24,416).

Таблица 3.27

Дополнительные статистические характеристики синхронизации сетей $\langle \text{ТРМ}(K=3, N=11, L=2)^{A/B} \rangle$

Класс	*ts* (от)	*ts* (до)	Количество	Вероятность, *p*
1	26	50,4159	83	0,0415
2	50,4159	74,8318	375	0,1875
3	74,8318	99,2477	561	0,2805
4	99,2477	123,6636	406	0,203
5	123,6636	148,0795	273	0,1365
6	148,0795	172,4954	152	0,076
7	172,4954	196,9113	86	0,043
8	196,9113	221,3272	29	0,0145
9	221,3272	245,7431	19	0,0095
10	245,7431	270,159	9	0,0045
11	270,159	294,5749	2	0,001
12	294,5749	318,9908	5	0,0025

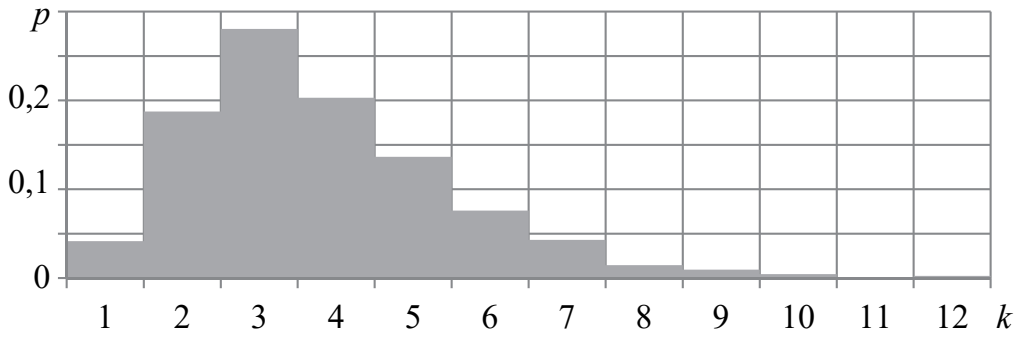

Рис. 3.21. Распределение времен (выражены через *p*) синхронизации сетей $\langle \text{ТРМ}(K=3, N=11, L=2)^{A/B} \rangle$

Обобщенные результаты моделирования содержит табл. 3.28. В ней представлены среднее, минимальное, максимальное время и третий квартиль для сетей с разными параметрами *L* после 2000 попыток синхронизации для каждой топологии.

Таблица 3.28

Итоговые статистические характеристики синхронизации сетей $\langle \text{ТРМ}(K=3, N=11, L)^{A/B} \rangle$

ТРМ			Время синхронизации, *ts*			
K	*N*	*L*	*AVG*	*MIN*	*MAX*	*Q*3
3	11	1	31	7	101	38
		2	106	26	318	128
		3	236	54	860	282

Окончание табл. 3.27

TPM			Время синхронизации, *ts*			
K	*N*	*L*	*AVG*	*MIN*	*MAX*	*Q*3
3	11	4	429	120	1230	520
		5	670	215	2723	789
		6	958	273	3353	1142
		7	1330	359	4524	1601
		8	1790	484	5801	2139
		9	2250	727	8462	2664
		10	2874	878	10 483	3423

В табл. 3.29 представлены те же результаты, но под иным углом зрения.

Таблица 3.29

Сравнение среднего значения и третьего квартиля времени синхронизации сетей

TPM			*ts*			
K	*N*	*L*	*AVG*/*MAX*, %	*Q*3/*MAX*, %	Δ = *Q*3 – *AVG*, %	Δ/*MAX*, %
3	11	1	30,5	37,6	7	7
		2	33,5	40,3	22	7
		3	27,5	32,8	46	5
		4	34,9	42,3	91	7
		5	24,6	29,0	119	4
		6	28,6	34,1	184	5
		7	29,4	35,4	271	6
		8	30,9	36,9	349	6
		9	26,6	31,5	414	5
		10	27,4	32,7	548	5
Среднее значение			29,4	35,2	–	5,9

Как видно из приведенных данных, в среднем наблюдаемые сети синхронизировались за время, составляющее примерно 30% наиболее пессимистического варианта. Глядя на третий квартиль, можно сказать, что он составляет в среднем 35% самой продолжительной синхронизации. Эти значения для любой топологии не превышают 50%. Это означает, что 75% синхронизированных сетей достигают согласованных весов менее чем за половину самой продолжительной наблюдаемой синхронизации.

Далее в табл. 3.30–3.33 и на рис. 3.22 представлены аналогичные статистические результаты для сетей $\langle \mathrm{TPM}(K=3, N=101, L)^{A/B} \rangle$ и $L \in [1;\ 10]$.

Таблица 3.30

Статистические характеристики синхронизации сетей $\langle \mathbf{TPM}(K=3, N=101, L=3)^{A/B} \rangle$

Параметр	Символ	Точное значение
Количество опытов	n	14 212
Минимальное время синхронизации, циклов	*MIN_ts*	133
Максимальное время синхронизации, циклов	*MAX_ts*	1156
Диапазон	*R_ts*	1023
Квартиль 0	$Q0$	133
Квартиль 1	$Q1$	268
Квартиль 2	$Q2$	325
Квартиль 3	Q3	398
Квартиль 4	$Q4$	1156
Среднее значение	*AVG*	345,9187

Таблица 3.31

Дополнительные статистические характеристики синхронизации сетей $\langle \mathbf{TPM}(K=3, N=101, L=3)^{A/B} \rangle$

Класс	*ts* (от)	*ts* (до)	Количество	Вероятность, *p*
1	133	202,1833	565	0,039755137
2	202,1833	271,3665	3200	0,225161835
3	271,3665	340,5498	4197	0,295313819
4	340,5498	409,733	3088	0,217281171
5	409,733	478,9163	1558	0,109625668
6	478,9163	548,0995	827	0,058190262
7	548,0995	617,2828	394	0,027723051
8	617,2828	686,466	202	0,014213341
9	686,466	755,6493	103	0,007247397
10	755,6493	824,8325	42	0,002955249
11	824,8325	894,0158	19	0,001336898
12	894,0158	963,1991	11	0,000773994
13	963,1991	1032,382	2	0,000140726
14	1032,382	1101,566	2	0,000140726
15	1101,566	1170,749	2	0,000140726

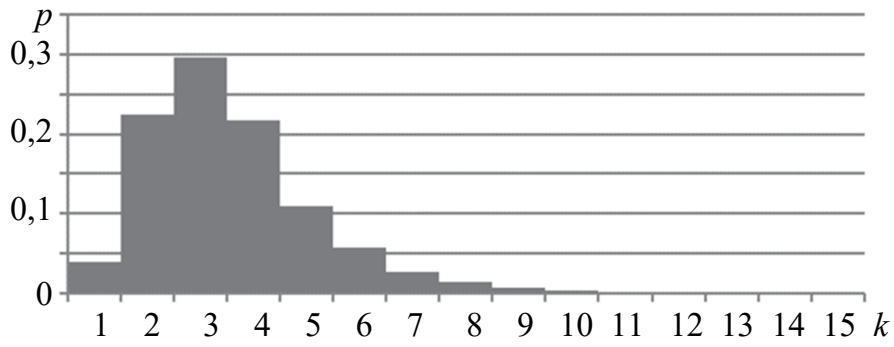

Рис. 3.22. Распределение времен синхронизации сетей
$\langle \mathrm{TPM}(K = 3, N = 101, L = 3)^{A/B} \rangle$

Таблица 3.32

Итоговые статистические характеристики синхронизации сетей $\langle \mathbf{TPM}(\boldsymbol{K = 3, N = 101, L})^{\boldsymbol{A/B}} \rangle$

TPM			*n*	Время синхронизации, *ts*			
K	*N*	*L*		*AVG*	*MIN*	*MAX*	*Q*3
3	101	1	6100	45	12	146	53
		2	6525	153	54	483	179
		3	14 212	346	133	1156	398
		4	1500	643	275	1843	744
		5	11 000	1022	418	2949	1172
		6	5123	1517	652	4483	1745
		7	11 498	2121	875	6479	2419
		8	4899	2841	1077	8234	3250
		9	8398	3681	1602	9060	4191
		10	6420	4638	2114	12 498	5301

В табл. 3.32 представлены результаты моделирования для сетей с тремя скрытыми нейронами, каждый из которых имел по 101 входу, значение весов $L \in [1; 10]$, что соответствует размеру системы $2L + 1$ из диапазона от 3 до 21. Представлено количество синхронизаций, среднее время синхронизации, измеренное в циклах, самая короткая и самая длинная синхронизация, а также третий квартиль.

Таблица 3.33

Сравнение среднего значения и третьего квартиля времени синхронизации сетей $\langle \mathbf{TPM}(\boldsymbol{K = 3, N = 101, L})^{\boldsymbol{A/B}} \rangle$

TPM			*ts*			
K	*N*	*L*	*AVG*/*MAX*, %	*Q*3/*MAX*, %	$\Delta = Q3 - AVG$, %	Δ/*MAX*, %
3	101	1	30,5	36,3	8	6
		2	31,7	37,1	26	5

Окончание табл. 3.33

TPM			*ts*			
K	*N*	*L*	*AVG/MAX*, %	*Q3/MAX*, %	$\Delta = Q3 - AVG$, %	Δ/MAX, %
3	101	3	29,9	34,4	52	5
		4	34,9	40,4	101	5
		5	40,4	46,3	150	6
		6	44,9	51,7	227	7
		7	37,8	43,2	298	5
		8	46,0	52,6	410	7
		9	49,5	56,3	508	7
		10	44,6	51,1	669	6
Среднее значение			39,0	44,9	–	5,9

По значению третьего квартиля можно определить количество шагов, после которых должна произойти синхронизация, а если сети еще не синхронизированы, то процесс следует запустить заново со случайными значениями весов. Подробное описание того, как завершить потенциально продолжительную синхронизацию, дано в следующем подразделе.

3.7.2. Продолжительность обмена согласованными результатами синхронизируемых сетей

Сети *A* и *B* должны «полагаться» на определенные критерии, чтобы определить окончание процесса синхронизации. В работах [30, 42, 77] и др. указывается, что достаточно длительный обмен непротиворечивыми результатами обеих сетей означает их синхронизацию. Однако, как мы уже неоднократно подчеркивали, такое толкование не всегда подтверждается.

Следует, вероятно, признать открытым вопрос определения длительности обмена уже согласованными результатами синхронизации сетей (см. также, например, [118, 119]). Это и является предметом нашего исследования, результаты которого представлены далее.

На первом этапе моделировалась синхронизация небольших сетей, что позволило точно отследить зависимость продолжительности обмена согласованными результатами несинхронизированными сетями от размера диапазона весовых коэффициентов. Анализу подверглись сети $\langle \mathrm{TPM}(K=3, N=4, L)^{A/B} \rangle$ для $L \in \{4, 10, 20, 30, \ldots, 100\}$.

Сети были синхронизированы 1000 раз. Для модификации весов применялся метод случайного блуждания. Самый длинный обмен непротиворечивыми результатами работы сетей увеличивается по мере возрастания L, вместе с тем при сравнении этой длины со значением третьего или четвертого квартиля (самая продолжительная синхронизация) наблюдалось соотношение, показанное на рис. 3.23.

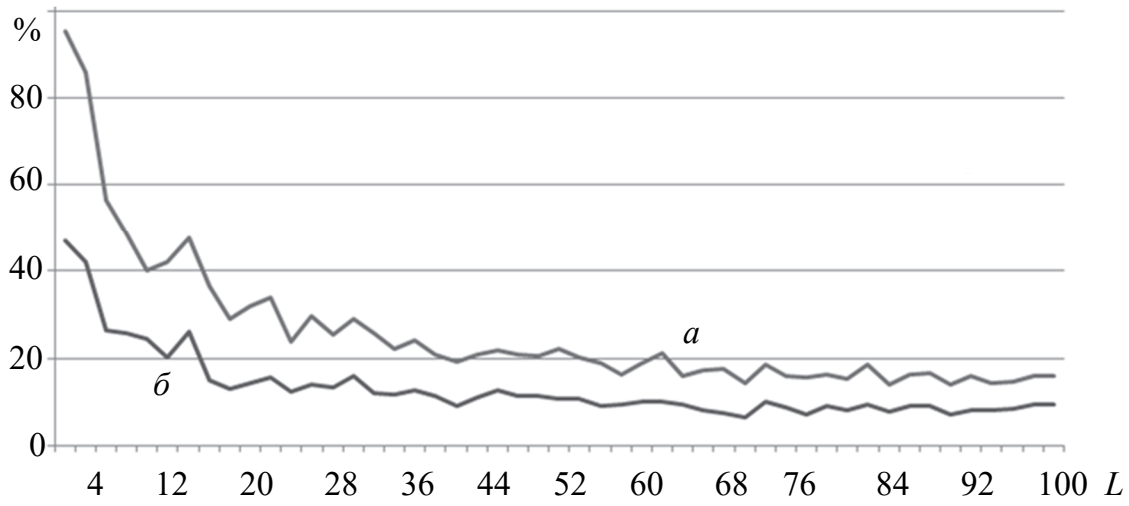

Рис. 3.23. Отношение длины самого продолжительного обмена согласованными весами к третьему (*а*) и четвертому (*б*) квартилям

Выбор третьего и четвертого квартилей для представления результатов на рис. 3.23 связан, в основном, с тем, что третий квартиль составляет примерно половину самой продолжительной синхронизации.

Более точные данные сведены в табл. 3.34. Таблица содержит наблюдаемое среднее время синхронизации, третий и четвертый квартили, длительность наиболее продолжительного обмена согласованными результатами несинхронизированными сетями ($ts_{макс}$), выраженную в количестве циклов, а также в процентах по отношению к третьему ($ts_{макс}$/3 кв.) и четвертому ($ts_{макс}$/4 кв.) квартилям.

Таблица 3.34

Результаты моделирования наиболее продолжительного времени синхронизации сетей $\langle \mathbf{TPM}(K = 3, N = 4, L)^{A/B} \rangle$

K-N-L	Средняя продолжительность синхронизации	Третий квартиль	Четвертый квартиль	$ts_{макс}$	$ts_{макс}$/3 кв., %	$ts_{макс}$/4 кв., %
3-4-4	91,17	109,00	222	52	23,4	47,7
3-4-10	539,55	633,00	1194	154	13	24
3-4-20	2112,65	2459,00	5520	355	6	14

Окончание табл. 3.34

K-N-L	Средняя продолжительность синхронизации	Третий квартиль	Четвертый квартиль	$ts_{макс}$	$ts_{макс}$/3 кв., %	$ts_{макс}$/4 кв., %
3-4-30	4557,41	5346,50	10 286	683	7	13
3-4-40	8330,13	9801,00	17 772	1022	6	10
3-4-50	12 906,70	15 003,75	26 935	1549	6	10
3-4-60	18 277,36	21 230,25	40 005	2014	5	9
3-4-70	24 564,41	28 231,00	62 152	2035	3	7
3-4-80	32 589,64	37 747,00	71 199	2859	4	8
3-4-90	40 335,44	46 884,00	93 895	3287	4	7
3-4-100	49 642,65	58 387,00	97 859	4653	5	8

Видно, что при небольших значениях L получен согласованный обмен результатами с несинхронизированными сетями, который длился (при $L = 4$) 52 цикла при третьем квартиле, равном 109, и наибольшем времени синхронизации продолжительностью 222 цикла. Это означает, что согласованный обмен результатами несинхронизированных сетей может занимать до 23,4% самого длительного времени синхронизации и 47,7% третьего квартиля. Следовательно, чтобы определить, синхронизированы ли уже сети, необходимо было бы получать обмен согласованными выходными результатами ($\sigma_i^A = \sigma_i^B$) примерно на 50% дольше, чем результат третьего квартиля. Это означает, что в 75% случаев синхронизации таких сетей процесс должен продолжаться в течение относительно длительного времени. При увеличении L наблюдалось снижение этого отношения.

На основании проведенных экспериментов можно сделать вывод, что основным критерием определения окончания процесса синхронизации может быть превышение определенной длительности обмена сетями согласованными результатами, соответствующими состоянию синхронизации. Выбор подходящего значения параметра L позволяет поддерживать разумную пропорцию между временем, необходимым для синхронизации сетей, и временем, необходимым для подтверждения этого факта.

Полученные нами и приведенные выше результаты не противоречат общему выводу о зависимости времени синхронизации двух сетей TPM от параметра L, сделанному в исследовании [93]: для неизменных K и N с увеличением L указанное время возрастает в отношении, примерно равном L^2. Зависимость времени синхронизации от N можно найти также, например, в работе [30].

3.7.3. Расчет и анализ взаимного соответствия весовых коэффициентов синхронизируемых сетей

3.7.3.1. Анализ расстояния между векторами весов. Базовым параметром, определяющим степень синхронизации сетей, является перекрытие векторов весов синхронизируемых сетей (см. п. 3.4.2, а также соотношения (1.22), (1.28), (3.14), (3.26), (3.27)), вычисляемое для каждого нейрона скрытого слоя по формуле (3.29). Перепишем ее еще раз и для удобства дальнейшего анализа обозначим здесь номером (3.40):

$$\rho_i^{A/B} = \frac{W_i^A W_i^B}{\sqrt{W_i^A W_i^A}\sqrt{W_i^B W_i^B}}, \qquad (3.40)$$

где i – номер нейрона; W_i^A и W_i^B – весовые векторы i-го нейрона сетей A и B соответственно. Другими словами, вычисляемый параметр $\rho_i^{A/B}$ представляет собой косинус угла между векторами весов i-го нейрона скрытого слоя сетей.

Вычисленное в соответствии с формулой (3.40) значение $\rho_i^{A/B}$ в начале процесса синхронизации зависит от рандомизированных значений весов и чем оно больше 0, тем благоприятнее был начальный выбор соответствующих параметров процесса, так как такое начальное состояние ближе к ожидаемому конечному результату. Здесь следует отметить, что состояние синхронизации, т. е. согласованные весовые векторы, подразумевает, что косинус достигает нуля, а обратное следствие неверно. В качестве простого довода достаточно задать весовые векторы с одинаковым направлением, но разной длиной, например (0; 1) и (0; 2).

Представленные далее результаты исследования [89] основывались на том, что анализировался один вектор весов, представляющий собой комбинацию векторов весов всех нейронов скрытого слоя сети: W^A и W^B. Такой вектор состоит из целых чисел KN от $-L$ до L. Зная W^A и W^B, можно по аналогии с формулой (3.40) определить взаимное перекрытие векторов весов сетей:

$$\rho^{A/B} = \cos\theta = \frac{W^A W^B}{\sqrt{W^A W^B}\sqrt{W^A W^B}} = \frac{W^A W^B}{\| W^A \| \| W^B \|}. \qquad (3.41)$$

Однако метод вычисления (3.41), как и (3.40), не позволяет делать выводы о соответствия векторов весов обеих сетей, поскольку

импликация в этом направлении неверна. Другим способом определения соответствия W^A и W^B может быть вычисление расстояния dist(·) между ними с помощью евклидовой метрики (см. п. 3.4.1). Расстояние между весовыми векторами обеих сетей определим выражением

$$\mathrm{dist}(W^A, W^B) = \| W^A - W^B \| = \sqrt{\sum_{k=1}^{KN} (w_k^A - w_k^B)^2}. \qquad (3.42)$$

В этом случае индекс k указывает на значение конкретного веса в последовательности всех весов сети. Связь между косинусом угла между весовыми векторами и расстоянием между ними задается законом косинусов для унитарных пространств:

$$(\| W^A - W^B \|)^2 = (\| W^A \|)^2 + (\| W^B \|)^2 - 2 \| W^A \| \| W^B \| \cos\theta. \quad (3.43)$$

Расстояние между векторами, выраженное формулой (3.43), в отличие от (3.41) и (3.42), позволяет сделать вывод о синхронизации сетей на основе вычисленного значения. Важным отличием косинуса угла между векторами от расстояния между ними является также интервал, которому принадлежат значения обеих функций, и изменение этих значений при синхронизации сетей. Косинус для начальных весовых векторов принимает значения, близкие к 0, и сохраняет возрастающий тренд, чтобы принять значение, равное 1 для полностью синхронизированных сетей. Евклидово расстояние в начале синхронизации принимает бóльшие значения в зависимости от размера сети и диапазона, к которому относятся веса, а для синхронизированных сетей – принимает значение 0. Для сравнения этих обеих величин на шаге t необходимо свести их к общему набору значений. Для этого используем формулу, обозначив через t_{synch} время (количество шагов) до наступления состояния синхронизации:

$$\mathrm{dist}(W^A, W^B) = 1 - \frac{\mathrm{dist}(W^{A(t)}, W^{B(t)}) - \min\limits_{1 \le a \le t_{\mathrm{synch}}} (\mathrm{dist}(W^{A(a)}, W^{B(a)}))}{\max\limits_{1 \le a \le t_{\mathrm{synch}}} (\mathrm{dist}(W^{A(a)}, W^{B(a)}) - \min\limits_{1 \le a \le t_{\mathrm{synch}}} (\mathrm{dist}(W^{A(a)}, W^{B(a)})}. \quad (3.44)$$

Проведено 2000 симуляций синхронизации сетей $\langle \mathrm{TPM}(K = 3, N = 40, L = 5)^{A/B} \rangle$. Полученные времена синхронизации находятся в диапазоне от 150 до 985 циклов, в среднем – 348 циклов. Для примера на рис. 3.24 показано, как косинус (3.41) – верхний график, и

нормированное евклидово расстояние (3.44) – нижний график, изменяются при синхронизации сетей с указанной топологией.

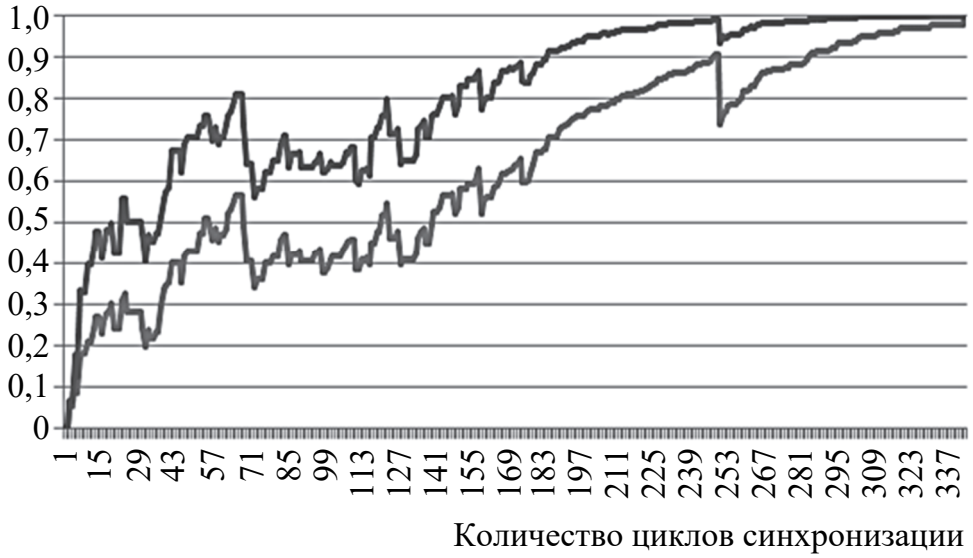

Рис. 3.24. Характер изменения мер согласования весовых коэффициентов сетей в процессе их синхронизации

Предлагаемое нормализованное евклидово расстояние для всех анализируемых сетей сильно коррелирует со значениями косинуса. Коэффициент корреляции для сетей (табл. 3.35) превышает 0,95.

Таблица 3.35

Корреляция между результатами вычисления по формулам (3.41) и (3.44)

ts	Коэффициент корреляции
150	0,952
200	0,969
250	0,962
300	0,958
350	0,966
400	0,952
450	0,962
500	0,967

Это означает, что оба рассмотренных метода обеспечивают примерно одинаковую точность. Важным преимуществом использования евклидова расстояния является возможность его применения для вывода о том, что сети синхронизированы.

Основываясь на полученных результатах взаимодействия сетей, можно рассмотреть такой параметр, как частота, с которой сети

обмениваются согласованными результатами синхронизации. Используя эти частоты, можно сделать вывод о соответствии весовых векторов сети.

Представленный ниже метод позволяет вычислить частоту обмена результатами согласования весов на s предыдущих шагах, где s – заданное натуральное число. Оказывается, это значение коррелирует с мерами совместимости весовых векторов, описанными выше. Это позволяет оценить степень соответствия векторов весов синхронизируемых сетей еще до завершения процесса синхронизации.

Рассмотрим далее последовательность $(a_t)_{t=1}^{t_{\text{synch}}}$, для которой $a_t = 1$ тогда и только тогда, когда $\tau_t^A = \tau_t^B$, и $a_t = 0$ тогда и только тогда, когда $\tau_t^A \neq \tau_t^B$. Индекс t – это номер цикла синхронизации сетей и $t = 1, 2, \ldots, t_{\text{synch}}$. Задана последовательность b_t, элемент которой является средним арифметическим элементов последовательности a_t с индексами $t, t-1, \ldots, t-s+1$. Если $t < s$, т. е. в последовательности a_t нет s элементов, предшествующих t-му элементу, то вычисляется среднее арифметическое доступных элементов последовательности, начиная от первого элемента. Создание такой последовательности можно записать как

$$b_t = \frac{1}{l} \sum_{j=t-l+1}^{t} a_j, \tag{3.45}$$

где $l = \min(t, s)$.

Проведенные исследования указывают на то, что в большинстве сетей существует устойчивая линейная зависимость между анализируемыми параметрами. Для сетей с более длительным временем синхронизации корреляция ниже и указывает на умеренную связь. Это означает, что сети A и B, анализируя указанную частоту, могут «сделать вывод» о равенстве (или неравенстве) векторов весов.

При анализе тех же 2000 симуляций синхронизации сетей $\langle \text{TPM}(K = 3, N = 101, L = 3)^{A/B} \rangle$ использовались соответствующие характеристики сетей со временем синхронизации от 150 до 500 циклов с шагом 50. Для этих сетей рассчитана частота появления согласованных результатов в предыдущих циклах, количество которых составляет $s = 25, 75, \ldots, 225$. В табл. 3.36 приведены коэффициенты корреляции между этими частотами и элементами

последовательности частот, заданными формулой (3.41), а также последовательностью частот и нормированными расстояниями, заданными выражением (3.44).

Таблица 3.36

Коэффициенты корреляции между частотами, рассчитанными по формулам (3.41) и (3.45) – cos; по формулам (3.44) и (3.45) – dist

ts	Корреляция	*s*				
		25	75	125	175	225
150	dist	0,9334	0,9476	0,9188	0,9131	0,9131
	cos	0,9434	0,9298	0,9366	0,9461	0,9461
200	dist	0,8903	0,8220	0,7547	0,7472	0,7475
	cos	0,7954	0,6801	0,6134	0,6182	0,6212
250	dist	0,8910	0,9340	0,9344	0,9345	0,9351
	cos	0,8132	0,8527	0,8579	0,8635	0,8901
300	dist	0,8411	0,8329	0,8008	0,7823	0,7907
	cos	0,7121	0,6845	0,6515	0,6407	0,6599
350	dist	0,8082	0,8308	0,7708	0,6977	0,6170
	cos	0,7001	0,6877	0,6088	0,5214	0,4325
400	dist	0,7374	0,7830	0,7386	0,7538	0,7497
	cos	0,5926	0,6373	0,6283	0,6748	0,6612
450	dist	0,7014	0,6563	0,5552	0,4631	0,3683
	cos	0,5187	0,4462	0,3270	0,2241	0,1231
500	dist	0,6920	0,7016	0,6790	0,6447	0,6232
	cos	0,5788	0,5660	0,5322	0,4927	0,4730

Таким образом, анализируя частоту обмена результатами обеими сетями, можно оценить степень совпадения векторов весов этих сетей.

Принимая во внимание, не превышает ли частота обмена совместимыми результатами определенный порог (расстояние между векторами весов), можно принять решение о разрыве синхронизации, которая не гарантирует быстрого ее окончания. Среди протестированных сетей $\langle \mathrm{TPM}(K = 3, N = 101, L = 3)^{A/B} \rangle$ в итоге была выбрана одна пара сетей, которые быстро синхронизировались, и одна пара, обучение которых было близко к самому длительному наблюдаемому процессу. Было выявлено, что в случае длительной синхронизации существует порог около 0,8, превышение которого в сравнительно небольшом количестве последующих циклов приведет

к синхронизации весов. Поэтому для всех анализируемых частот были проанализированы различные пороговые значения от 0,6 (на рис. 3.25 – LV_0,6000) до 0,85 (LV_0,8500).

В случае сети, которая синхронизируется быстро, наблюдается превышение порогового значения уже на ранних этапах синхронизации. Этот факт показан на диаграммах на рис. 3.25, которые были получены на частоте, испытанной в предыдущих 75 циклах.

Частота обычно увеличивается по мере обучения сетей, что согласуется с увеличением степени соответствия векторов весов синхронизирующих сетей.

По мере увеличения циклов обучения порог, которого достигают расчетные частоты, увеличивается. Графики (рис. 3.26) показывают, что частота превышает пороговые значения для относительно больших времен синхронизации. Здесь также подсчитывалась частота за предыдущие 75 циклов.

Рис. 3.25. Демонстрация превышения порогового уровня при относительно небольшом числе циклов синхронизации сетей

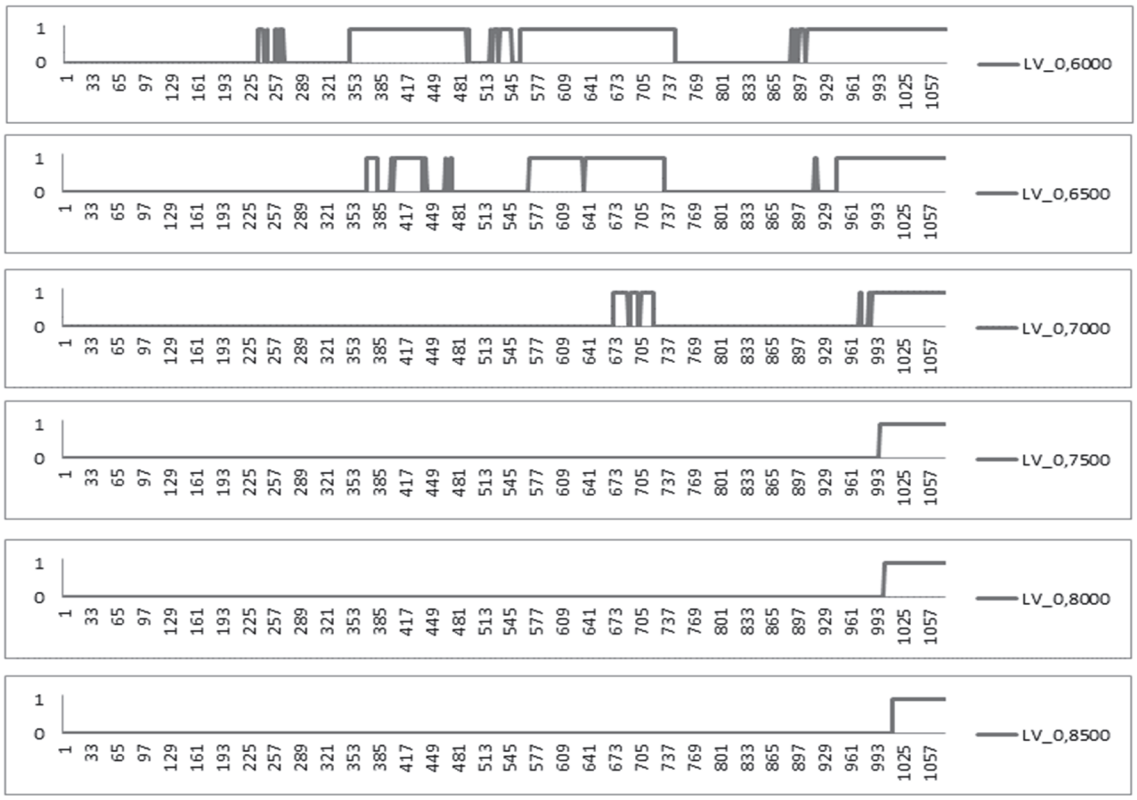

Рис. 3.26. Демонстрация превышения порогового уровня
при относительно большом числе циклов синхронизации сетей

Видно, что частоты, а также расстояния между векторами весов существенно флуктуируют. Учитывая, что среднее время синхронизации этой сети составляет 346 циклов, использование относительно низкого порога 0,65 указывает на то, что быстрой синхронизации не будет. Этот порог не превышается примерно до 360 циклов и, кроме того, через короткое время частота падает ниже порогового значения.

3.7.3.2. Вероятностная оценка взаимного соответствия весов. Взаимное соответствие (перекрытие) весовых коэффициентов сетей можно проанализировать с вероятностной точки зрения [97–99, 112].

Множество начальных значений весов ТРМ (в соответствии с определением 3.1) задается декартовым произведением K одинаковых сомножителей, каждое из которых является множеством $([-L, L])^N$.

Не уменьшая общности задачи, можно заняться оценкой размерности пространства $([-L, L])^N$ только для одного нейрона.

В работе [112] рассматривается процесс образования «пустых точек», т. е. точек, исключенных из множества потенциальных решений, или точек, принадлежащих множеству невозможных решений

или исходов (см. п. 3.4.2). С каждым шагом, на котором происходит процесс синхронизации, количество точек, принадлежащих множеству потенциальных решений, сокращается или (в частном случае) остается неизменным. Отсюда следует, что с ростом числа шагов размерность рассматриваемого пространства уменьшается (рис. 3.9 и 3.10 на с. 75–76). Поскольку размерность пространства изменяется в процессе синхронизации, полезно определить его меру.

На начальном и каждом последующем шаге синхронизации весовые коэффициенты являются элементами множества, определяемого декартовым произведением множеств $(\{-L, L\})^N$, число которых равно K. Всем возможным состояниям (исходам) на каждом шаге синхронизации, ассоциированным с точками пространства, соответствуют вероятности, начальные значения которых для каждого нейрона можно определить как $1/(2L + 1)^N$. Если в процессе синхронизации (при свободном движении точек в пространстве возможных решений; рис. 3.9 и 3.10) происходит наложение точек, то соответствующие им вероятности суммируются.

Интерес представляет анализ размерности такого пространства для $\langle \text{TPM}(K, N, L)^{A/B} \rangle$, удовлетворяющего критерию безопасного размера (KN) генерируемого криптографического ключа при достижении состояния синхронизации. Как было отмечено выше, относительно безопасной принято считать сеть TPM, в которой $N \geq 100$. При таком значении N простыми вычислениями очень трудно точно определить размерность анализируемого пространства (в разумные сроки). Поэтому следует применить другой способ определения этого параметра.

Вводится вероятностная оценка пространства решений по поиску состояний $W^A = W^B$ на основе вероятности $|\langle \text{TPM}(K, N, L)^{A/B} \rangle|_p$, которая означает, что случайно взятые (на определенном шаге синхронизации) значения векторов весовых коэффициентов будут одинаковыми. Величину $|\langle \text{TPM}(K, N, L)^{A/B} \rangle|_p$ можно использовать как меру размерности пространства. В соответствии с этим можно осуществлять выбор параметров (K, N, L) сетей A и B по определенному критерию. Кроме того, такой анализ может дать, например, ответ на вопрос, какая сеть лучше: $\langle \text{TPM}(K = 20, N = 20, L = 5)^{A/B} \rangle$ или $\langle \text{TPM}(K = 3, N = 15, L = 10)^{A/B} \rangle$.

Выполнено 100 тестовых опытов синхронизации сетей A и B [99]. В ходе каждого опыта случайным образом выбирались и анализировались пары весовых коэффициентов. Такая выборка продолжалась

до тех пор, пока не были найдено 10 правильных решений, т. е. точек, соответствующих состоянию синхронизации сетей.

Результаты экспериментов представлены на рис. 3.27–3.30 в виде гистограмм (по оси Y – количество тестов, по оси X – вероятность $|\cdot|_p$).

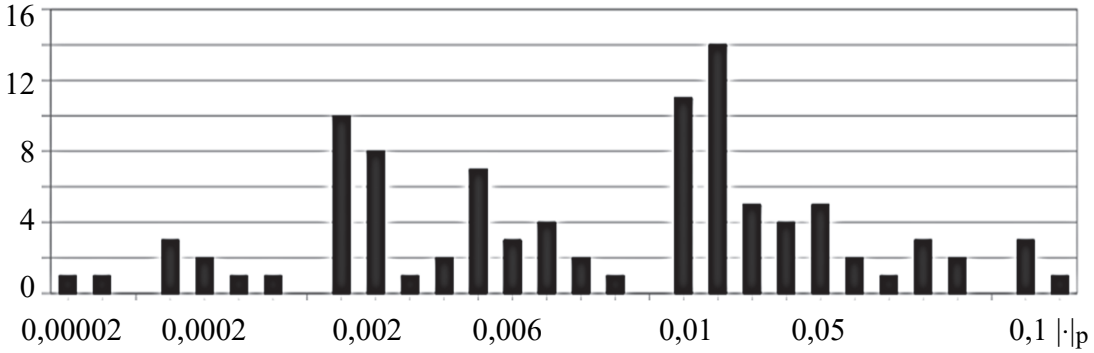

Рис. 3.27. Вероятностная оценка сходимости векторов весов в процессе синхронизации для $\langle \mathrm{TPM}(K=3, N=40, L=5)^{A/B} \rangle$

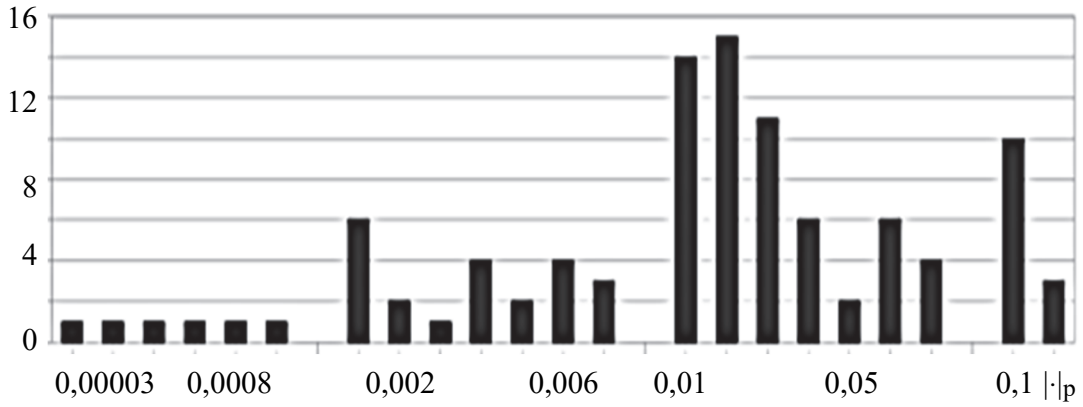

Рис. 3.28. Вероятностная оценка сходимости векторов весов в процессе синхронизации для $\langle \mathrm{TPM}(K=4, N=40, L=5)^{A/B} \rangle$

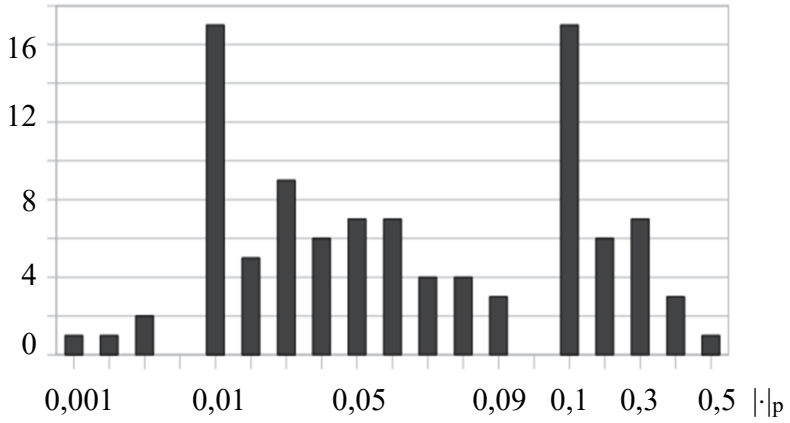

Рис. 3.29. Вероятностная оценка сходимости векторов весов в процессе синхронизации для $\langle \mathrm{TPM}(K=5, N=40, L=5)^{A/B} \rangle$

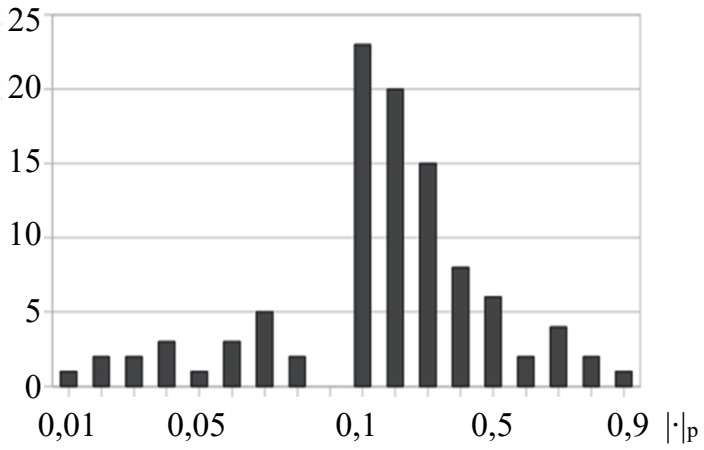

Рис. 3.30. Вероятностная оценка сходимости векторов весов в процессе синхронизации для $\langle \mathrm{TPM}(K = 6, N = 40, L = 5)^{A/B} \rangle$

В табл. 3.37 приведены усредненные результаты, полученные на основе 1000 опытов для одного (из K) нейронов и для всей сети.

Таблица 3.37

Средние показатели вероятностной оценки пространства состояний синхронизации TPM

K-L-M	Средняя вероятность $\lvert\langle \mathrm{TPM}(K, N, L)^{A/B} \rangle\rvert_p$	
	Для одного нейрона	Для всей сети
3-40-5	0,025100000206	0,000015625
4-40-5	0,042396622279	0,000003231
5-40-5	0,110485	0,000016463
6-40-5	0,28698	0,000558611
3-60-5	0,008316544567	0,000000575930368
3-80-5	0,006224653975	0,000000240641848
3-100-5	0,003071858138	0,000000028934443

Приведенные результаты показывают, что размерность анализируемого пространства значительно возрастает с увеличением N. Кроме того, размерность пространства различается даже при одинаковых параметрах сетей.

Это подтверждает тот факт, что сближение весовых коэффициентов может характеризоваться различной скоростью при тех же параметрах *K-L-M*.

3.8. Классификация периодов общего времени синхронизации сетей

Анализ частот обмена согласованными результатами, описанный выше, позволяет оценить степень соответствия весов обученных сетей и указать на потенциально долговременные периоды синхронизации. Ниже представлен метод использования результатов указанного анализа для отнесения параметров некоторой текущей синхронизации к одной из заданных временных групп или периодов [88–90, 102, 103]. Это позволяет предсказывать не только длительность синхронизации сетей, но и приблизительный период этого процесса.

С помощью статистического анализа можно разделить времена синхронизации на определенные классы: например, указать короткие, типичные и длительные периоды. Определив частоту обмена согласованными сетевыми результатами, можно выяснить, относится ли текущая синхронизация к одному из этих классов. Представленные ниже результаты относятся к сетям со следующими параметрами *K*-*N*-*L*: 3-16-3, 3-11-2, 3-11-3, 3-101-2, 3-101-3.

Все сети обучались по правилу случайного блуждания. Процесс синхронизации повторялся 1000 раз, каждый раз с новыми рандомизированными начальными значениями весов и входных импульсов.

Время, необходимое сетям для достижения согласованных векторов весов, зависит от топологии сети и диапазона, к которому принадлежат веса ([–*L*, *L*]). Результаты статистического анализа для сетей $\langle \text{TPM}\ (K = 3, N = 16, L = 3)^{A/B} \rangle$ после 1000 симуляций показаны в табл. 3.38. Она содержит квартили и среднее значение времени (*AVG*) синхронизации (циклов). Здесь синхронизации длительностью 200 циклов были определены как типичные, а продолжительностью более 600 циклов – как длительные.

Как видно из табл. 3.38, периоды синхронизации условно разделены на 3 класса, каждый из которых соотнесен с квартилями: 1-й класс (*короткий* период синхронизации) – *Q*0–*Q*1; 2-й класс (*типичный* период) – *Q*2–*Q*3; 3-й класс (*длительный* период) – *Q*4. В табл. 3.39 представлены соответствующие результаты для других TPM.

Таблица 3.38

Статистические характеристики процесса синхронизации сетей $\langle \mathbf{TPM}(\boldsymbol{K}=3,\boldsymbol{N}=16,\boldsymbol{L}=3)^{A/B} \rangle$

Квартиль	Продолжительность синхронизации	Характеристика периода
$Q0$	92	Короткий
$Q1$	192	Короткий
$Q2$	245	Типичный
$Q3$	310	Типичный
$Q4$	843	Длительный
AVG	261	–

Таблица 3.39

Статистические характеристики классификации продолжительностей процесса синхронизации сетей

TPM (K-N-L)	Класс	ts
3-11-2	1	75
	2	129–130
	3	230–334
3-11-3	1	169–170
	2	279–280
	3	550–810
3-101-2	1	119
	2	178–179
	3	315–407
3-101-3	1	264–266
	2	396–399
	3	704–1221

Описанный подход к классификации основан на анализе евклидовых расстояний между векторами весов сетей. По мере того как сети взаимно обучаются, их веса приближаются друг к другу, пока не достигнут одинаковых значений. На рис. 3.31 для примера показаны графики нормализованных расстояний между векторами для сетей $\langle \mathrm{TPM}(K = 3,\ N = 16,\ L = 3)^{A/B} \rangle$. Представлены 14 типичных синхронизаций, закончившихся после 200 циклов, и 7 – для длительных периодов синхронизаций (более 600 циклов). Как видно, можно отметить пороговое значение на уровне примерно 0,8, превышение которого свидетельствует о скором окончании процесса

синхронизации. Если же такой порог не превышается, целесообразно возобновить процесс синхронизации. Вероятность успеха после возобновления составляет примерно 0,75.

В каждом классе было отмечено не менее десяти опытов, в результате которых сети синхронизировались во временном интервале, близком к характеристике квартиля.

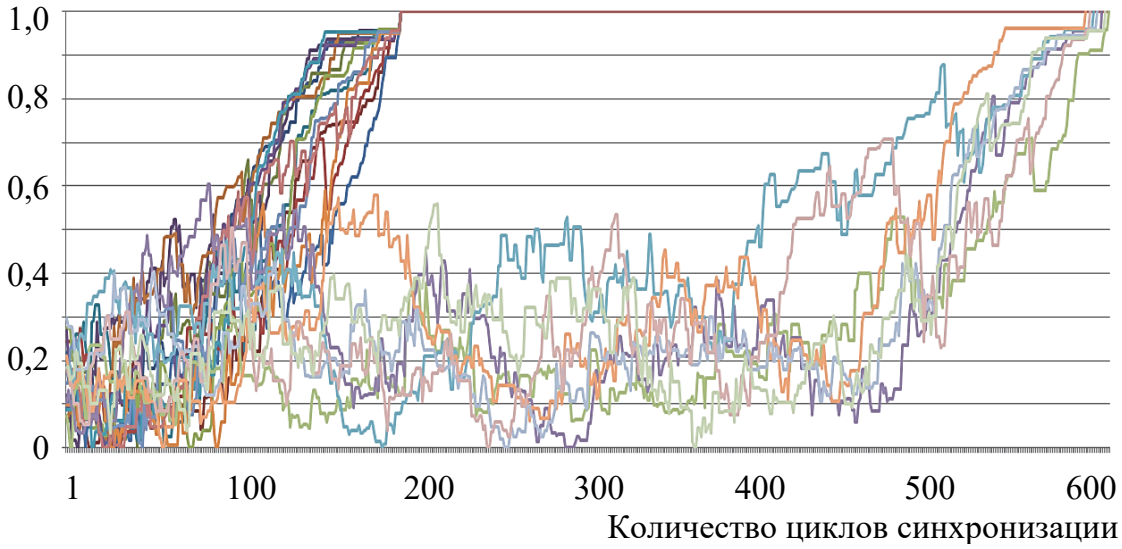

Рис. 3.31. Динамика типичных и длительных синхронизаций сетей $\langle \mathrm{TPM}(K=3, N=16, L=3)^{A/B} \rangle$

Воспользовавшись тем фактом, что евклидово расстояние между векторами тесно связано с частотой обмена согласованными результатами работы обеих сетей, был проведен анализ на основе использования определенных частот.

Детально анализировался каждый из выбранных (табл. 3.39) процессов синхронизаций, закончившихся за число циклов, близкое к характеристике квартиля. При этом оценивались значения расстояний, рассчитанных в соответствии с формулами (3.41) – cos, (3.44) – dist_mod, а также частоты обмена согласованными результатами, отобранными в последних 40, 50, …, 80 циклах (соответствуют обозначениям кривых на рис. 3.32–3.37: *pr_XX*), рассчитанные по формуле (3.45).

Все синхронизации, относящиеся к 1-му классу, имеют схожий характер изменчивости частоты обмена согласованными результатами. На рис. 3.32 показаны анализируемые параметры при стандартном ходе синхронизации с коротким периодом.

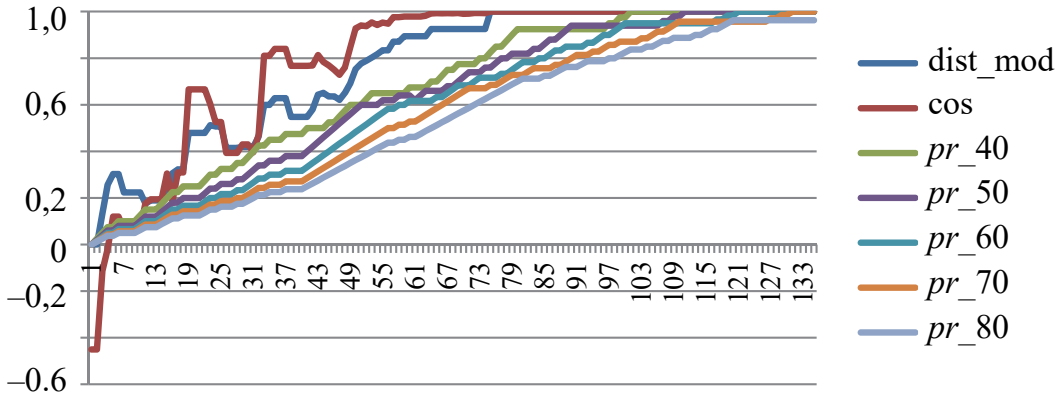

Рис. 3.32. Характеристики синхронизации сетей $\langle \mathrm{TPM}(K=3, N=11, L=2)^{A/B} \rangle$ за короткий период

Даже случайно значительные, но кратковременные уменьшения расстояния между векторами не влияют на частоту появления согласованных результатов. Этот эффект достигается за счет того, что частота вычисляется на относительно большем количестве предыдущих шагов. На рис. 3.33 показан как раз такой случай, когда косинус уменьшился с 0,8 до 0,4, а расстояние было соответственно изменено почти до 0. Однако характер изменения частот, взятых за время от 40 до 80 периодов, очень похож на стандартный.

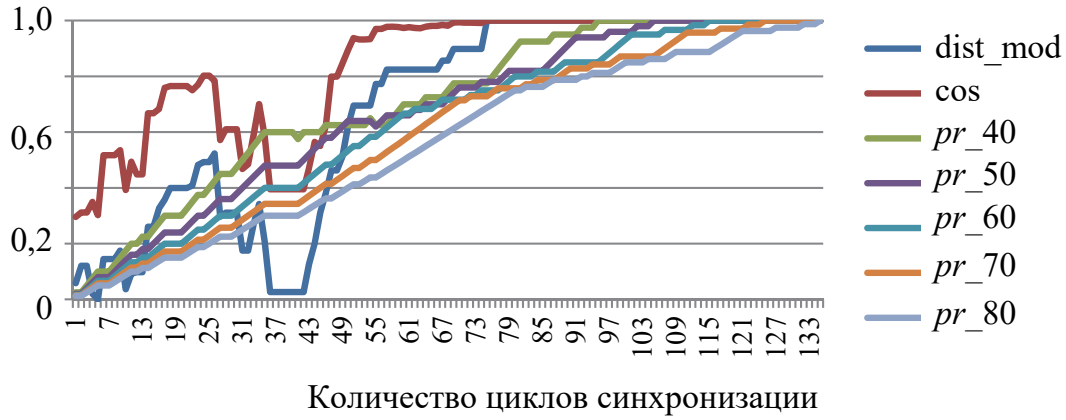

Рис. 3.33 Характеристики синхронизации сетей $\langle \mathrm{TPM}(K=3, N=11, L=2)^{A/B} \rangle$ за типичный период

Синхронизация может быть долговременной, если процесс характеризуется появлением большого числа отталкивающих шагов, приводящих к снижению совместимости весовых векторов. Это влечет за собой снижение частоты обмена согласованными результатами обучаемых сетей (рис. 3.34 и 3.37).

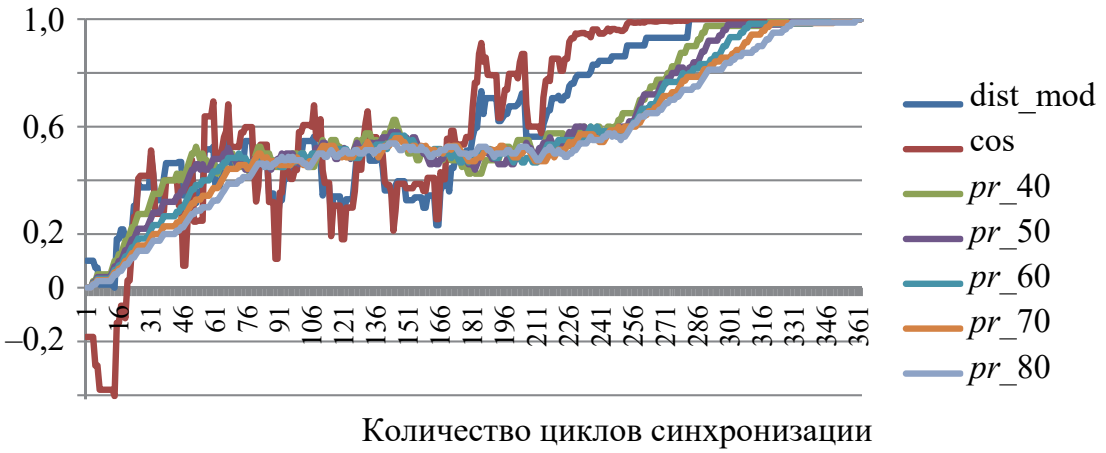

Рис. 3.34. Характеристики синхронизации сетей $\langle \mathrm{TPM}(K = 3, N = 11, L = 2)^{A/B} \rangle$ за длительный период

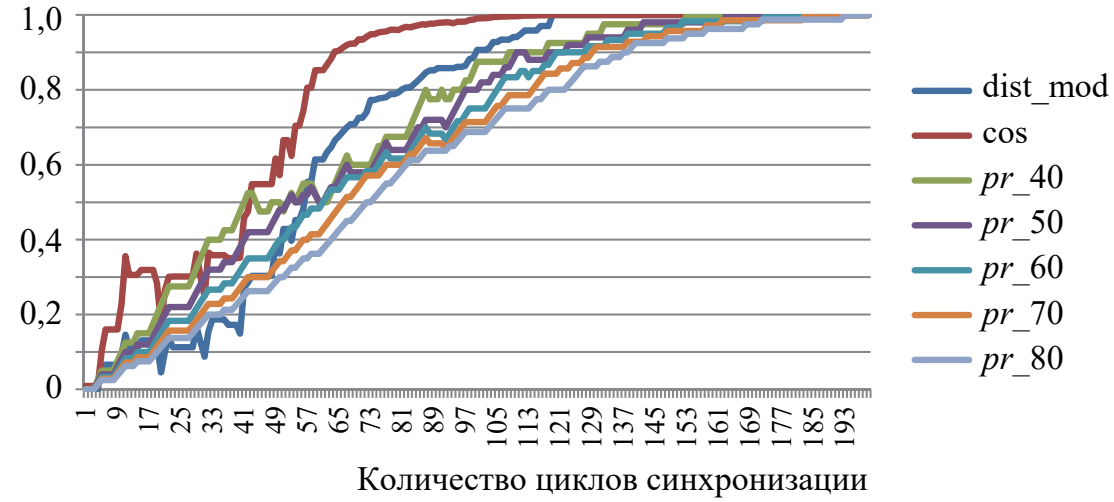

Рис. 3.35. Характеристики синхронизации сетей $\langle \mathrm{TPM}(K = 3, N = 101, L = 2)^{A/B} \rangle$ за короткий период

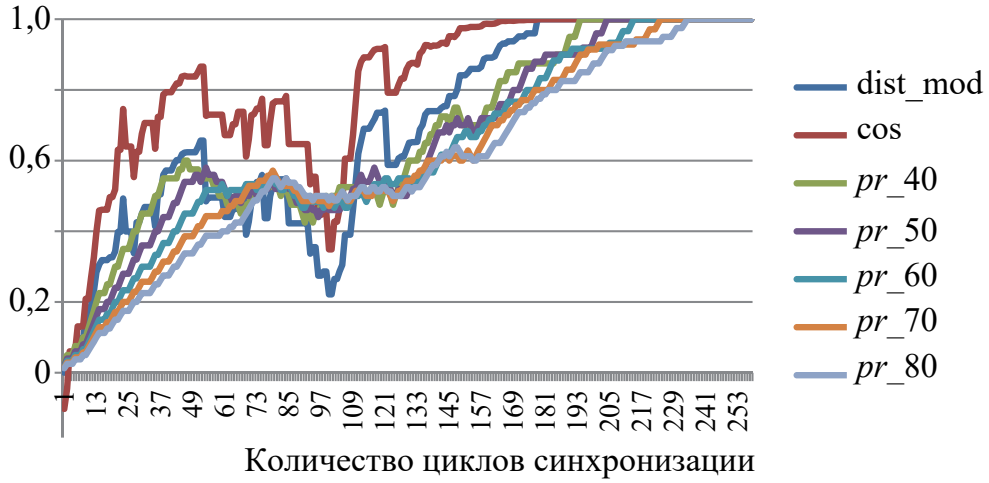

Рис. 3.36. Характеристики синхронизации сетей $\langle \mathrm{TPM}(K = 3, N = 101, L = 2)^{A/B} \rangle$ за типичный период

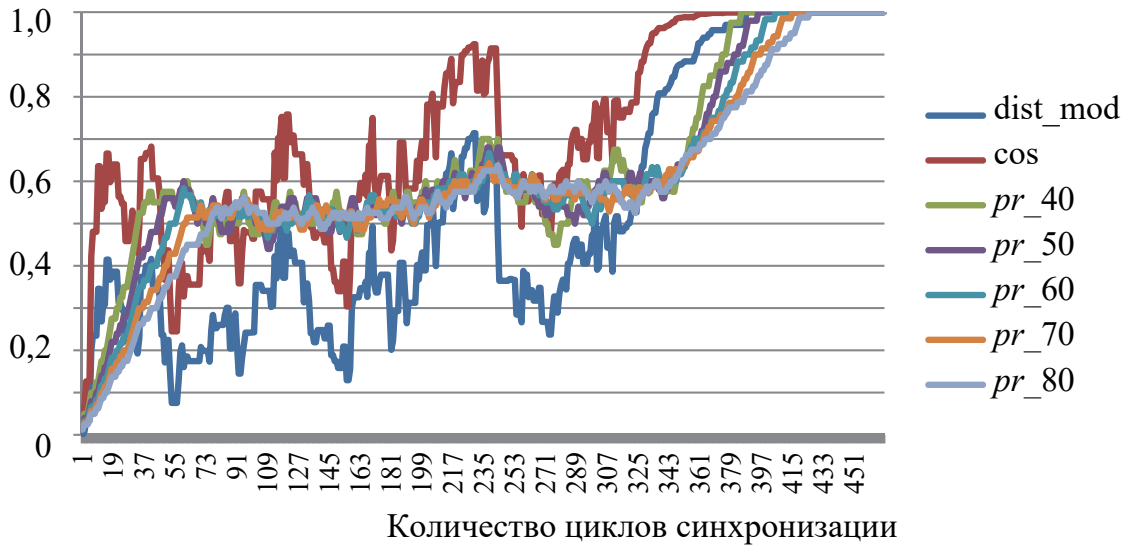

Рис. 3.37. Характеристики синхронизации сетей $\langle \text{TPM}(K = 3, N = 101, L = 2)^{A/B} \rangle$ за долговременный период

Подобными характеристиками (рис. 3.32–3.34 и 3.35–3.37) можно оценить длительности (периодичности) процессов синхронизации и других сетей TPM.

Используя представленные результаты в протоколе согласования криптографического ключа, стороны A и B могут определить параметры сетей TPM и выбрать соответствующий метод обучения. На основе статистического анализа для выбранной архитектуры сетей можно определить первый, третий или четвертый квартили (в зависимости от того, какой класс по длительности процесса синхронизации выбирается в качестве основного критерия), к которому далее будет отнесена сама процедура. Можно выбрать и иное время, за которое стороны ожидают окончания процесса синхронизации. Пусть это время будет обозначено t_z. Во время синхронизации в каждый момент времени t должна определяться частота (f_e) обмена согласованными выходными результатами. Если $t < t_z$ и был превышен порог частоты 0,8, т. е. $f_e^{(t-1)} \le 0{,}8$ и $f_e^{(t)} > 0{,}8$, синхронизацию сетей следует продолжить, сравнивая длительность обмена согласованными результатами обеих сетей.

3.9. Атаки на TPM

Основой безопасности процесса синхронизации двух сетей TPM: $\langle \text{TPM}(K, N, L)^A \rangle$ и $\langle \text{TPM}(K, N, L)^B \rangle$, и, соответственно, основной проблемой злоумышленника E (см. рис. 3.5 на с. 57), т. е. сети (или –

в общем случае – сетей) $\langle \mathrm{TPM}(K,N,L)^E \rangle$ является то, что последней не известны внутренние представления (σ_1, σ_2, …, σ_K) TPM A и B. Поскольку направление изменения векторов весов сетей A и B в процессе их синхронизации зависит от σ_i каждой из этих сетей, для успешной атаки со стороны $\langle \mathrm{TPM}(K,N,L)^E \rangle$ атакующей стороне важно правильно угадать состояние скрытых нейронов сетей $\langle \mathrm{TPM}(K,N,L)^A \rangle$ и $\langle \mathrm{TPM}(K,N,L)^A \rangle$. Большинство известных атак используют именно такой метод угадывания. Возможно, в перспективе будет найден «умный» метод атаки, который полностью нарушит безопасность нейронной криптографии.

Некоторые важные особенности процесса синхронизации сетей A и B на вероятностном уровне проанализированы в подгл. 3.5. В этой связи целесообразно вспомнить, что в большинстве проанализированных результатов исследований вероятность успеха атакующей стороны уменьшается экспоненциально с ростом L, а среднее время синхронизации сетей A и B растет примерно пропорционально квадрату L^2 [79, 93, 96, 102, 103, 105, 107, 120, 121] (см. также п. 3.5.1 и 3.5.3), что является важнейшим фактором, определяющим относительно невысокую эффективность атак на TPM.

В текущем подразделе проанализируем особенности реализации и важнейшие факторы, влияющие на эффективность наиболее известных в настоящее время атак на легитимно взаимодействующие TPM.

К основным видам атак относятся: простая, геометрическая, атака большинства, а также генетическая и вероятностная атаки. В литературе описаны некоторые модификации или комбинации перечисленных атак, а также средства защиты от них [34, 42, 101, 105, 107, 122–127].

Поскольку нас для анализа эффективности атак интересуют изменения векторов весов, целесообразно представить формальные записи правил обучения сетей в следующем виде (с учетом соотношений (3.8)–(3.25))[3] [42]:

$$w_{ij}^{(t+1)} = \begin{cases} g(w_{ij}^{(t)} - x_{ij}\tau^{(t)}\Theta(\tau^{(t)}\sigma_i^{(t)})\Theta(\tau^{A(t)}\tau^{B(t)})), \text{ метод анти-Хэбба,} \\ g(w_{ij}^{(t)} + x_{ij}\tau^{(t)}\Theta(\tau^{(t)}\sigma_i^{(t)})\Theta(\tau^{A(t)}\tau^{B(t)})), \text{ метод Хэбба,} \\ g(w_{ij}^{(t)} + x_{ij}\Theta(\tau^{(t)}\sigma_i^{(t)})\Theta(\tau^{A(t)}\tau^{B(t)})), \text{ метод случайного блуждания.} \end{cases} \quad (3.46)$$

[3] Принадлежность параметра к сети A или B без необходимости не указывается.

3.9.1. Простая атака

Простая атака (Simple Attack) реализуется на основе простейшего алгоритма однонаправленного обучения сети [30, 42, 73, 79, 80, 123]. Злоумышленник E пытается синхронизироваться с $\langle \mathrm{TPM}(K,N,L)^A \rangle$ или $\langle \mathrm{TPM}(K,N,L)^B \rangle$ обучив $\langle \mathrm{TPM}(K,N,L)^E \rangle$ с использованием параметров x_i и, например, τ^A. Сеть $\langle \mathrm{TPM}(K,N,L)^E \rangle$, построенная на основе архитектуры сетей A и B, пытается угадать вектор весов сети A, но, в отличие от простого метода «учитель – ученик», веса учителя ($\langle \mathrm{TPM}(K,N,L)^A \rangle$) в этом случае зависят от времени, поэтому злоумышленник должен использовать некоторую стратегию атаки, чтобы следовать шагам учителя.

В случае простой атаки выходы σ_i^E сети E не корректируются до применения принятого правила обучения, а обновление весов происходит независимо от τ^E, поскольку взаимодействие между A и E невозможно. Сеть E использует то же правило обучения, что и партнеры, за исключением того, что τ^A заменяется на τ^E. С учетом этих особенностей правила обучения сети E можно представить в формальном виде, модифицировав выражение (3.46):

$$w_{ij}^{E(t+1)} = \begin{cases} g(w_{ij}^{E(t)} - x_{ij}\tau^{A(t)}\Theta(\tau^{A(t)}\sigma_i^{E(t)})\times \\ \times\Theta(\tau^{A(t)}\tau^{B(t)})), \text{ метод анти-Хэбба}, \\ g(w_{ij}^{E(t)} + x_{ij}\tau^{A(t)}\Theta(\tau^{A(t)}\sigma_i^{E(t)})\times \\ \times\Theta(\tau^{A(t)}\tau^{B(t)})), \text{ метод Хэбба}, \\ g(w_{ij}^{E(t)} + x_{ij}\Theta(\tau^{A(t)}\sigma_i^{E(t)})\times \\ \times\Theta(\tau^{A(t)}\tau^{B(t)})), \text{ метод случайного блуждания}. \end{cases} \quad (3.47)$$

В приведенных выражениях $g(x)$ подчиняется формуле (3.23). В соответствии с (3.47) сеть E использует внутреннее представление выходных сигналов собственных нейронов скрытого слоя (σ_i^E; $i = 1, 2, \ldots, K$) для оценки состояния сети A, даже если выходные данные сетей A и E отличаются.

Положим, оппонент E, наблюдающий за обменом информацией, знает, когда происходит изменение векторов весов обеих сетей (при $\tau^A = \tau^B$), однако ему неизвестно, какой «вклад» в конечный результат вносят соответствующие нейроны, и, соответственно, какие векторы весов подвергаются изменению. Происходит это потому, что одна выходная величина (1 или –1) может быть результатом многих разных конфигураций выходных величин $\sigma_i^{A/B}$.

Полагаем, что используется обучение по правилу Хэбба. Как показывают тесты, через определенный промежуток времени (t_{learn}) сеть E достигнет состояния синхронизации с атакуемыми сетями: A и B. Все же время обучения t_{learn} обычно гораздо больше, чем время синхронизации t_{synch} ($t_{\text{learn}} > t_{\text{synch}}$; см. табл. 3.10 на с. 80).

Возникает вопрос: почему $t_{\text{learn}} > t_{\text{synch}}$? Говоря точнее, почему время обучения сети оппонента с атакуемыми сетями значительно больше, чем время синхронизации обеих сетей, хотя интруз обладает той же самой информацией? Это объясняется тем, что сети влияют на процесс синхронизации друг друга, тогда как оппонент имеет возможность только пассивного наблюдения. Когда сети генерируют свои выходные величины ($\tau^A = \tau^B$), и к тому же выходная величина оппонента E согласуется с выходными величинами наблюдаемых сетей A и B ($\tau^A = \tau^B = \tau^E$), то все сети производят шаг в одном направлении (с одинаковой вероятностью). Проблема возникает тогда, когда $\tau^A = \tau^B$, но $\tau^A = \tau^B \neq \tau^E$. В этом случае сеть E или пропускает шаг обучения, или все-таки пробует изменить свои векторы весов, рискуя принять ошибочное решение. Как первый, так и второй варианты приводят к задержке в процессе обучения сети оппонента времени синхронизации наблюдаемых сетей. Это отставание является главным элементом безопасности процесса синхронизации. Если этот процесс завершится на довольно раннем этапе, то оппонент будет не в состоянии синхронизировать свои векторы весов с наблюдаемыми сетями. Следовательно, чем больше эта задержка, тем выше уровень безопасности.

Для примера рассмотрим сети, состоящие только из двух нейронов в скрытом слое: $K = 2$. Анализ будет касаться вероятности совершения сетями B и E того же самого движения, что и в сети A. Определим вероятность того, что весовые коэффициенты сетей A и B совпадут [35]:

$$p = P(\tau^A = \tau^B), \tag{3.48}$$

и для простоты предположим, что вероятность одинакова для обоих персептронов: $p_1 = p_2 = p$. Возможны следующие варианты:

$$\sigma_1^A = \sigma_1^B, \sigma_2^A = \sigma_2^B, \tag{3.49}$$

$$\sigma_1^A \neq \sigma_1^B, \sigma_2^A = \sigma_2^B, \tag{3.50}$$

$$\sigma_1^A = \sigma_1^B, \sigma_2^A \neq \sigma_2^B, \tag{3.51}$$

$$\sigma_1^A \neq \sigma_1^B, \sigma_2^A \neq \sigma_2^B, \tag{3.52}$$

которым соответствуют вероятности p^2, $(1-p)p$, $p(1-p)$ и $(1-p)^2$.

Случаи (3.50) и (3.51) относятся к ситуации, где $\tau^A \neq \tau^B$, и веса обеих сетей не будут модифицированы. И все же в этом случае оппонент E и наблюдаемая сеть A могут иметь одинаковую выходную величину: $\tau^A = \tau^E$. В такой ситуации сеть E должна выполнить движение вместе с сетью A, что, однако, не произойдет, поскольку сеть A в этой ситуации не произведет никаких модификаций, так как $\tau^A \neq \tau^B$.

На основании приведенных формул (3.48)–(3.52) мы найдем вероятность того, что векторы весов сетей A и B производят шаг в одном направлении. Эта величина равна выражению (3.28):

$$\frac{p^2}{p^2 + (1-p^2)}.$$

В то же время вероятность такого совпадения между сетями A и E составляет p. Рассмотрение крайних случаев ($p = 0$ и $p = 1$) можно опустить; получаем условие $0 < p < 1$. Отсюда вытекает следующее неравенство:

$$\frac{p^2}{p^2 + (1-p^2)} > p. \tag{3.53}$$

Следовательно, вероятность совершения согласованного шага между сетями $\langle \text{TPM}(K,N,L)^A \rangle$ и $\langle \text{TPM}(K,N,L)^B \rangle$ больше, чем та же вероятность в отношении сетей $\langle \text{TPM}(K,N,L)^A \rangle$ и $\langle \text{TPM}(K,N,L)^E \rangle$ (для каждого p). Это доказывает установленный на основании многих тестов вывод: время синхронизации сетей $\langle \text{TPM}(K,N,L)^A \rangle$ и $\langle \text{TPM}(K,N,L)^B \rangle$ меньше, чем время обучения, необходимое атакующей стороне: $\langle \text{TPM}(K,N,L)^E \rangle$.

Более подробную информацию об особенностях реализации простой атаки можно найти в диссертации [42]. В частности, в указанной работе приведена зависимость вероятности успешной простой атаки ($t_{\text{learn}} \leq t_{\text{synch}}$) P_E как функции от K и L при $N = 1000$ (рис. 3.38).

Не обнаружено ни одной успешной простой атаки за 1000 симуляций с использованием параметров $K = 3$ и $L = 3$. Следовательно, простой атаки недостаточно, чтобы нарушить безопасность протокола обмена нейронными ключами на основе $\langle \text{TPM}(K,N,L)^A \rangle$ при $K \leq 3$.

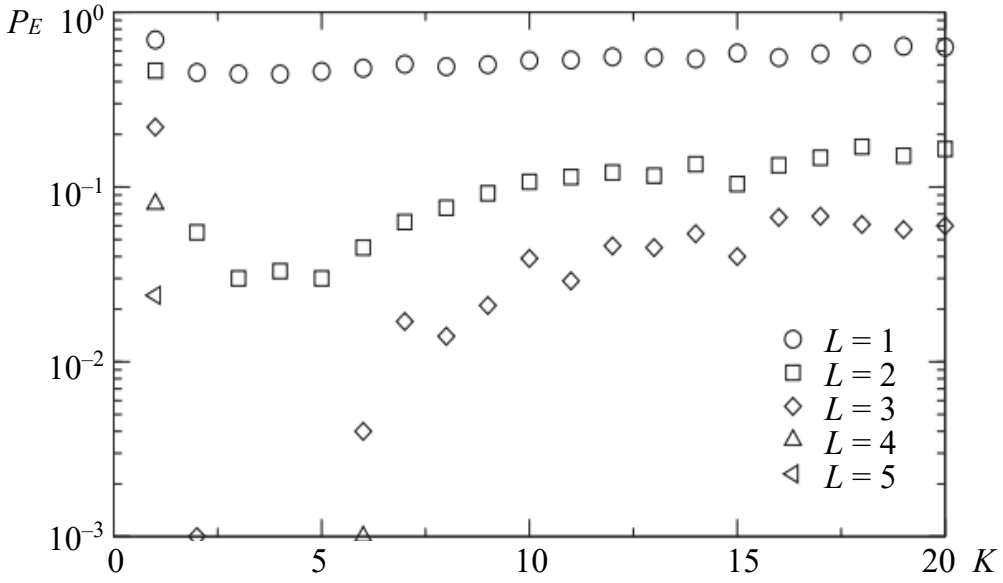

Рис. 3.38. Зависимость вероятности успешной простой атаки от K и L при $N = 1000$ [42]

3.9.2. Геометрическая атака

Геометрическая атака (Geometric Attack) [35, 42, 88, 93, 112, 121, 124] работает лучше, чем простая атака, потому что атакующая сеть $\langle \text{TPM}(K, N, L)^E \rangle$ учитывает τ^E, а также уровни сигналов на выходах элементов скрытого слоя. Данная атака считается наиболее эффективной для злоумышленника, использующего только одну TPM [42].

Как и в простой атаке, E пытается имитировать действия сети B (при этом в основе используются известные правила обучения (3.46)), не имея возможности взаимодействовать с A. При $\tau^E = \tau^A$ атаку можно реализовать, просто применяя уравнения (3.47), так как соответствующие правила обучения, записанные в виде формул (3.46) и (3.47), дают практически одинаковый эффект. Однако в случае $\tau^E \neq \tau^A$ атакующая сеть E не может остановить обновление весов сети A. Учитывая это, злоумышленник использует дополнительную информацию, определяемую локальными значениями параметра μ (3.6):

$$\mu_i^E = \frac{1}{\sqrt{N}} W_i^E X_i \,, \tag{3.54}$$

чтобы скорректировать τ^E. Эти данные можно использовать для определения уровня достоверности сигналов на выходах каждого

элемента скрытого слоя [42, 121, 125]. Поскольку малое абсолютное значение $|\mu_i^E|$ указывает на высокую вероятность того, что $\sigma_i^A \neq \sigma_i^E$, атакующая сеть изменяет выход элемента σ_i^E с минимальным значение $|\mu_i^E|$ до применения соответствующего шага используемого правила обучения. Во всех случаях веса w_{ij}, выходящие за пределы разрешенного диапазона $[-L, L]$, сбрасываются до ближайшего граничного значения: $\text{sign}(w_{ij})L$.

Для оценки эффективности атаки рассматриваемого типа разработано специализированное программное средство (NCS), интерфейс которого показан на рис. 3.39 [112].

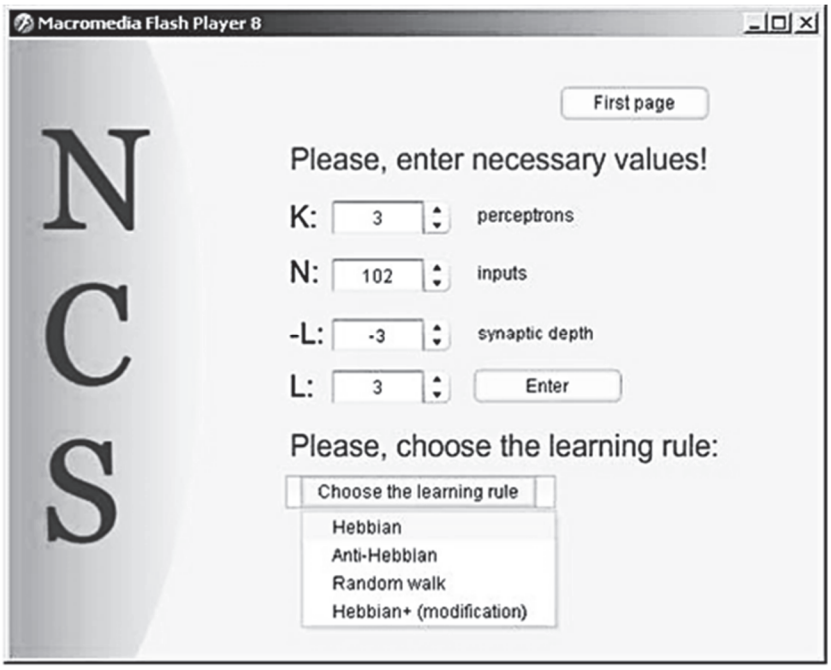

Рис. 3.39. Основное окно программного средства NCS для исследования процесса синхронизации $\langle \text{TPM}(K, N, L)^{A/B} \rangle$, а также простой и геометрической атак

Результаты экспериментов с использованием данного средства показали, что при $K = 3$, $N = 5$, $L = 3$ геометрическая атака на основе правила Хэбба результативна ($t_{\text{learn}} \leq t_{\text{synch}}$) в 30% случаев. Однако при той же архитектуре $\langle \text{TPM}(K = 3, N = 5, L = 7)^{A/B} \rangle$ использование обучения сети E по правилу анти-Хэбба заметно снижает эффективность атаки.

Результаты экспериментов также подтвердили наличие экспоненциальной зависимости вероятности успешной атаки от синаптической глубины L: с ее увеличением вероятность успешной атаки

понижается по экспоненте (см. график в окне программного средства на рис. 3.40). Индексы «C» в нижней части рис. 3.40 относятся к атакующей сети *E*.

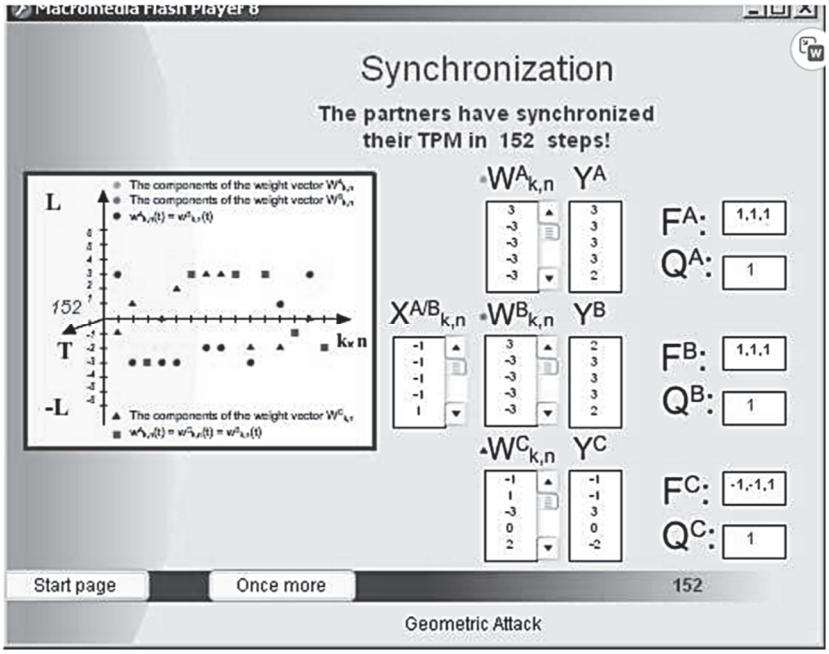

Рис. 3.40. Отображение результатов геометрической атаки в окне (в левой части) программного средства NCS

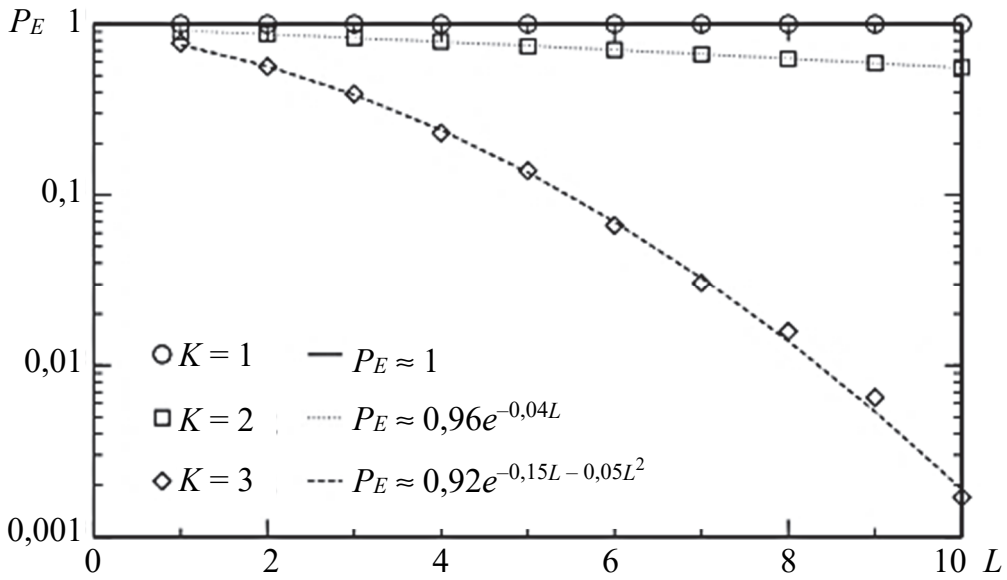

Рис. 3.41. Зависимость вероятности успешной геометрической атаки при различных параметрах сетей TPM [42][4]

[4] Размеры и начертания (шрифты) букв и чисел на данном и некоторых других подобных рисунках изменены в соответствии с форматами, установленными редколлегией издателя.

В статье [35] отмечается, что для анализа эффективности геометрической атаки сеть злоумышленника инициировалась 100 различными, случайно сгенерированными начальными состояниями. Оказалось, что по крайней мере в среднем одно из начальных состояний завершалось успешной атакой с вероятностью $P_E > 0{,}9$.

В работе [42] также приведена зависимость вероятности P_E успешной геометрической атаки от K и L при $N = 1000$ (рис. 3.41) для 10 тысяч симуляций при обучении сетей по правилу случайного блуждания.

Линии на рис. 3.41 соответствуют аппроксимациям численных результатов эксперимента соответствующими функциями, указанными в нижней части рисунка.

На рис. 3.42 [42] показаны зависимости P_E от K при фиксированных L при $N = 1000$ для правила случайного блуждания.

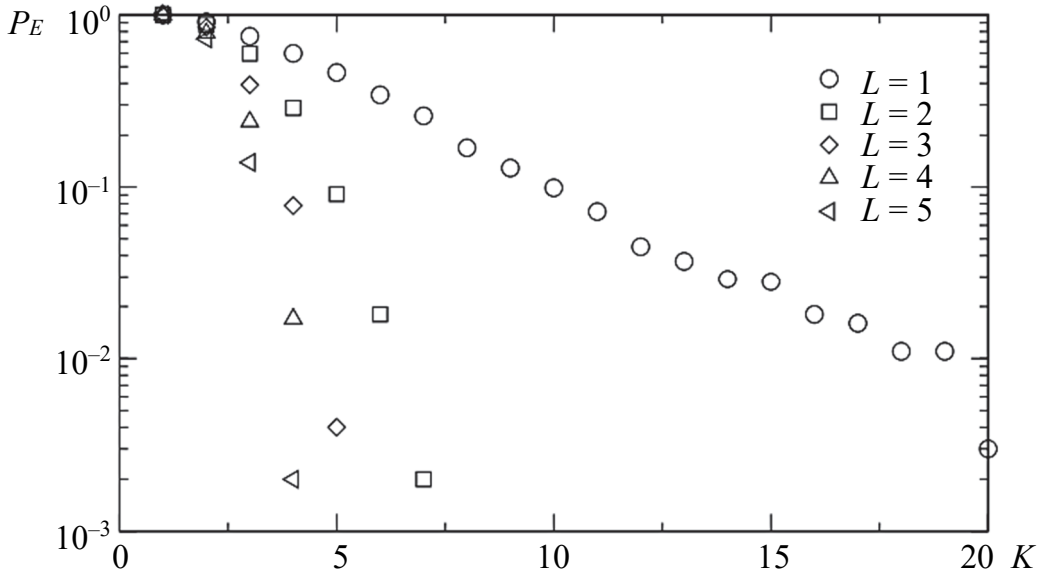

Рис. 3.42. Зависимость вероятности успешной геометрической атаки от K при фиксированных L

Как следует из анализа рис. 3.41, вероятность успеха геометрической атаки зависит не только от синаптической глубины L, но и от количества персептронов K. Этот эффект, приводящий к разным значениям коэффициентов в аппроксимирующих функциональных зависимостях, вызван особенностями используемых алгоритмов обучения сети E: на каждом шаге корректируется выход не более одного элемента в скрытом слое σ_i^E. Этого достаточно, чтобы избежать всех отталкивающих (см. п. 3.4.1) шагов для $K = 1$, однако при $K > 1$ выходы многих указанных элементов скрытого слоя TPM

могут соотноситься как $\sigma_i^E \neq \sigma_i^E$. Это означает, что вероятность данного события растет с увеличением K, так что в $\langle \mathrm{TPM}(K,N,L)^E \rangle$ возникает все больше и больше отталкивающих шагов.

Следовательно, сети A и B могут добиться защиты от этой атаки не только за счет увеличения синаптической глубины L, но и за счет использования большего числа нейронов K. Конечно, при $K > 3$ практически реализовать и использовать структуру TPM для больших значений L чрезвычайно трудно, так как процесс синхронизация в этом случае носит флуктуационный, порой неустойчивый характер, что мы ранее уже отмечали. Тем не менее рис. 3.42 показывает, что вероятность успеха для геометрической атаки быстро падает с увеличением K даже в случае $L = 1$.

Таким образом, характеризуя геометрическую атаку, можем сделать два важных вывода:

1) безопасность системы $\langle \mathrm{TPM}\ (K, N, L)^{A/B} \rangle$ определяет синаптическая глубина L машин: вероятность успеха P_E атаки экспоненциально уменьшается с ростом L;

2) время синхронизации t_{synch} увеличивается пропорционально L^2 (см., например, [93]). Следовательно, любой желаемый уровень защиты $\langle \mathrm{TPM}\ (K, N, L)^{A/B} \rangle$ от этой атаки может быть достигнут путем увеличения L (по крайней мере теоретически).

3.9.3. Атака большинства

Улучшенной версией геометрической атаки является атака большинства (Majority Attack). Ее особенность заключается в том, что атакующая сеть $\langle \mathrm{TPM}(K,N,L)^E \rangle$ для повышения эффективности атаки использует не одну TPM, а $Y\,(Y > 1)$ таких машин, которые в процессе реализации атаки взаимодействуют между собой. Большинство таких машин из числа Y определяют внутреннее представление (σ_1^E, σ_2^E, …, σ_K^E), которое используется на соответствующем шаге обучения машин атакующей сети.

Для описания состояний сети E используются два параметра [42]:

– среднее значение коэффициента $\rho_i^{A/E}$, обозначающего соответствие (перекрытие; overlap) весов соответствующих скрытых нейронов в сетях A и E:

$$\rho_i^{A/E} = \frac{1}{Y}\sum_{y=1}^{Y} \frac{W_i^A W_i^{E,y}}{\left\|W_i^A\right\|\left\|W_i^{E,y}\right\|}\ , \qquad (3.55)$$

который показывает степень синхронизации E с A. В выражении (3.55) индекс y относится к y-й ТРМ сети E; началу атаки соответствует $\rho_i^{A/E} = 0$, успешному завершению атаки – $\rho_i^{A/E} = 1$;

− среднее значение коэффициента $\rho_i^{E/E}$, обозначающего соответствие (перекрытие) весов соответствующих скрытых нейронов в двух ТРМ сети E:

$$\rho_i^{E/E} = \frac{1}{Y(Y-1)} \sum_{y=1}^{Y} \sum_{n \neq y} \frac{W_i^{E,y} W_i^{E,n}}{\left\| W_i^{E,y} \right\| \left\| W_i^{E,n} \right\|}, \tag{3.56}$$

нулевое и единичное численное значение которого имеет тот же смысл в отношении двух ТРМ сети E, что в формуле (3.55) относится к A и E.

В знаменателе обоих последних выражений используются нормированные векторы весов в соответствующем векторном пространстве. При больших значениях Y выход σ_i элемента скрытого слоя некоторой ТРМ (обозначим ее номером s) сети E будет таким же, как и у соответствующих элементов большинства ТРМ атакующей сети. Вектор весов этой машины можно нормализовать по всему ансамблю машин сети E:

$$W_i^{E,s,y} = \frac{1}{Y} \sum_{y=1}^{Y} \frac{W_i^{E,y}}{\left\| W_i^{E,y} \right\|}. \tag{3.57}$$

Нормализация по формуле (3.57) означает, что каждая ТРМ из общего числа Y в составе сети E при «голосовании по принципу большинства» имеет только один «голос».

При $Y \to \infty$ соответствие весов сети E весам сети A можно представить соотношением вида

$$\rho_i^{s,y} \sim \frac{\rho_i^{A/E}}{\sqrt{\rho_i^{E/E}}}. \tag{3.58}$$

Характер изменения и взаимосвязи векторов весов всех сетей, взаимодействующих при реализации атаки большинства (t соответствует количеству шагов) на основе соотношений (3.55)–(3.58) при $Y = 100$, $K = 3$, $L = 5$ и $N = 1000$ для обучения по методу случайного блуждания, показан на рис. 3.43 [42].

На рис. 3.44 [42] показаны усредненные (для 10 тысяч опытов) зависимости вероятности успешной атаки от L при $Y = 100$, $K = 3$, $N = 1000$ для трех правил обучения.

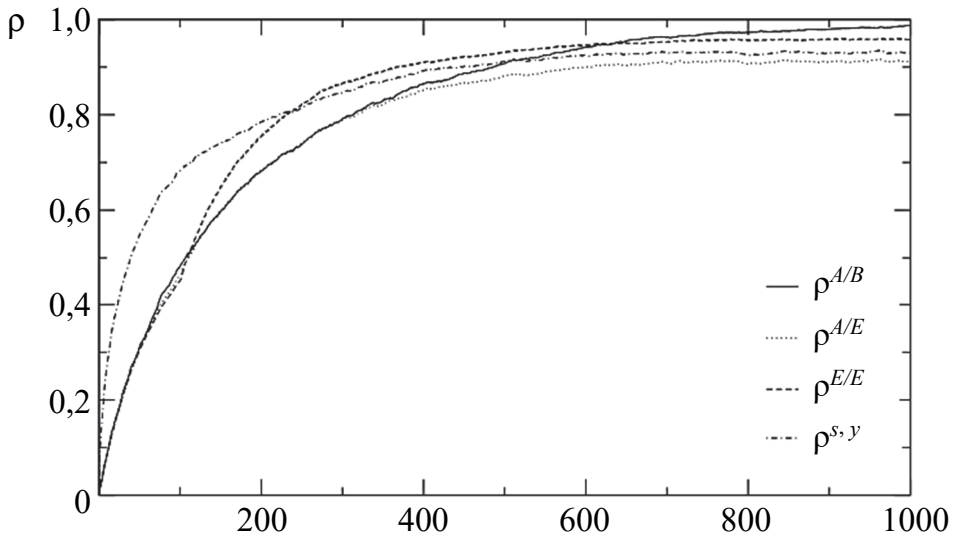

Рис. 3.43. Степень соответствия векторов весов всех участников атаки большинства с ростом числа шагов синхронизации сетей A и B

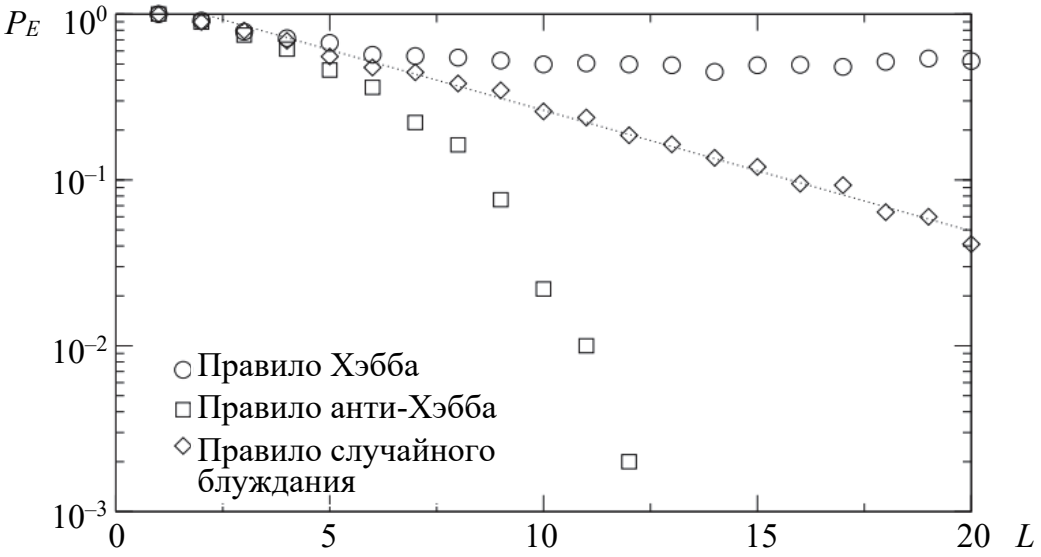

Рис. 3.44. Зависимость вероятности успешной атаки большинства от L при использовании различных правил обучения сетей

В целом эффективность атаки большинства считается более высокой по сравнению с геометрической атакой. Если $\langle \mathrm{TPM}(K,N,L)^A \rangle$ и $\langle \mathrm{TPM}(K,N,L)^B \rangle$ используют правило обучения Хэбба, P_E достигает постоянного отличного от нуля значения при $L \to \infty$ (рис. 3.44). По-видимому, изменения распределения веса, вызванного обучением по Хэббу, достаточно, чтобы нарушить безопасность протокола согласования ключей. Следовательно, сети A и B не могут использовать это правило обучения в криптографических целях.

3.9.4. Генетическая атака

Идея генетического алгоритма атаки (Genetic Attack) основана не на предсказании, как вышерассмотренные атаки, а на моделировании популяции виртуальных организмов и введении эволюционных правил, которые отдают предпочтение организмам с определенными желательными свойствами [35, 42, 96].

Атака реализуется с использованием большого количество машин ($\langle \text{TPM}(K,N,L)^E \rangle$) с одинаковой структурой, повторяющей структуру $\langle \text{TPM}(K,N,L)^A \rangle$ и $\langle \text{TPM}(K,N,L)^B \rangle$, и с обучением атакующих сетей по тем же правилам. На каждом шаге обучения около половины атакующих TPM получают выходной сигнал $\tau^E = +1$, а другая половина – $\tau^E = -1$. Сети E, результаты работы которых имитируют результаты сетей A и B, размножаются, остальные сети «умирают», т. е. исключаются из общего процесса.

Общее число сетей в составе E не превышает некоторый установленный предел Y. Атака начинается с использования одной TPM E со случайно выбранными весами. На каждом шаге обучения «популяция» атакующих ИНС развивается по трем возможным сценариям:

1) $\tau^A \neq \tau^B$ и, следовательно, сети A и B не меняют свои веса; при этом все атакующие TPM также остаются в прежнем состоянии;

2) $\tau^A = \tau^B$, а общее количество атакующих TPM ($Y/2^{K-1}$) в составе E не превышает установленный предел. Сеть E определяет все 2^{K-1} внутренних параметров (σ_1^E, σ_2^E, …, σ_K^E) каждой из атакующих TPM, которые воспроизводят выходной сигнал соответствующей TPM, совпадающий с выходом сети A: τ^A. Впоследствии эти совпадения используются для обновления весов в атакующих сетях в соответствии с применяемым правилом обучения. Такие манипуляции со стороны E создают 2^{K-1} вариантов каждой TPM на этом шаге *мутации* (Mutation Step);

3) $\tau^A = \tau^B$, но общее количество атакующих TPM ($I/2^{K-1}$) в составе E превышает установленный предел. В этом случае злоумышленник вычисляет выходные данные всех TPM, удаляет «неуспешные» ($\tau^E \neq \tau^A$) и обновляет веса в «успешных» TPM, используя стандартное правило обучения при фактических выходных данных скрытых персептронов.

После того как сети A и B синхронизируются, злоумышленник проверяет, имеет ли какая-либо из его сетей в составе E те же веса, что и A. Для TPM с $K = 3$, $N = 101$, $L = 3$ и $Y = 2500$ (сетей E) в более

чем 50% тестов хотя бы одна из атакующих TPM синхронизировалась с A еще до того, как сами A и B стали полностью синхронизированы, т. е. здесь $t_{synch} > t_{learn}$. В этих исследованиях применялись различные правила обучения. Атака оказалась особенно эффективной для вариантов, характеризующихся небольшими K (например, $K = 2$) [35].

Эффективность генетической атаки во многом зависит от алгоритма выбора TPM с наилучшим «предсказанием». В идеальном случае TPM атакующей сети, которая имеет ту же последовательность внутренних представлений (σ_1^E, σ_2^E, …, σ_K^E), что и сеть A, никогда не отбрасывается.

Дополнительную информацию можно получить из анализа рис. 3.45 [42], на котором изображены численные характеристики зависимости вероятности успешной генетической атаки от L при различных фиксированных значениях параметра I, полученные для 1000 опытов при $K = 3$ и $N = 1000$ и при использовании обучения по правилу случайного блуждания. На этом же рисунке приведены и соответствующие различным Y (на рисунке M соответствует Y) аппроксимирующие функции. Как видно, и для рассматриваемого вида атак их эффективность имеет логарифмическую зависимость от L.

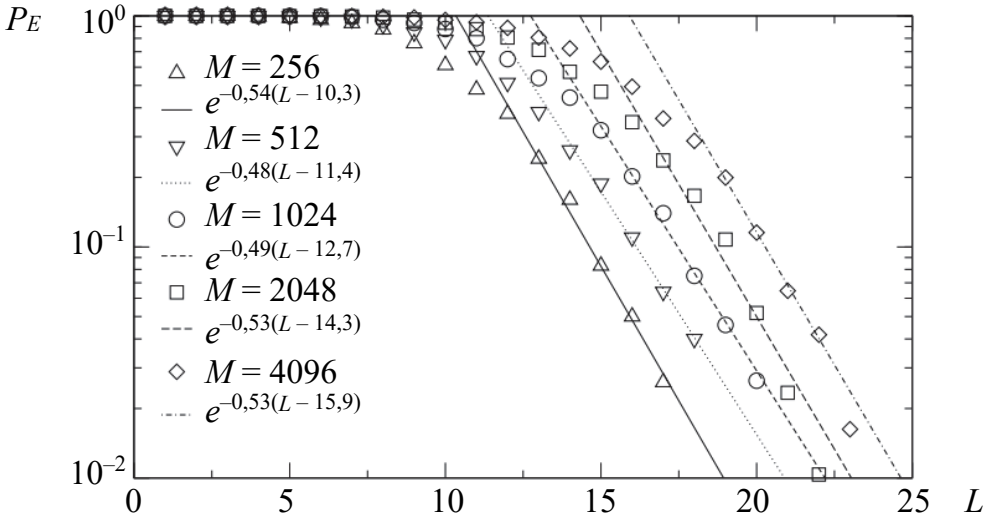

Рис. 3.45. Зависимость вероятности успешной генетической атаки от L при фиксированных Y

Очевидный вывод: уровень защиты протокола генерации ключевой информации на основе TPM от генетической атаки возрастает с ростом L и в пределе ($L \to \infty$) обеспечивается абсолютная устойчивость к атакам (и не только генетическим). Злоумышленники

должны экспоненциально увеличивать количество своих нейронных сетей, чтобы компенсировать более высокие значения L. Этот метод достигает наилучшей вероятности успеха из всех известных методов, только если синаптическая глубина L не слишком велика. При более высоких значениях L атакующий получает больше, используя атаку большинства. Для сравнения на рис. 3.46 показаны зависимости, характеризующие эффективность нескольких видов атак.

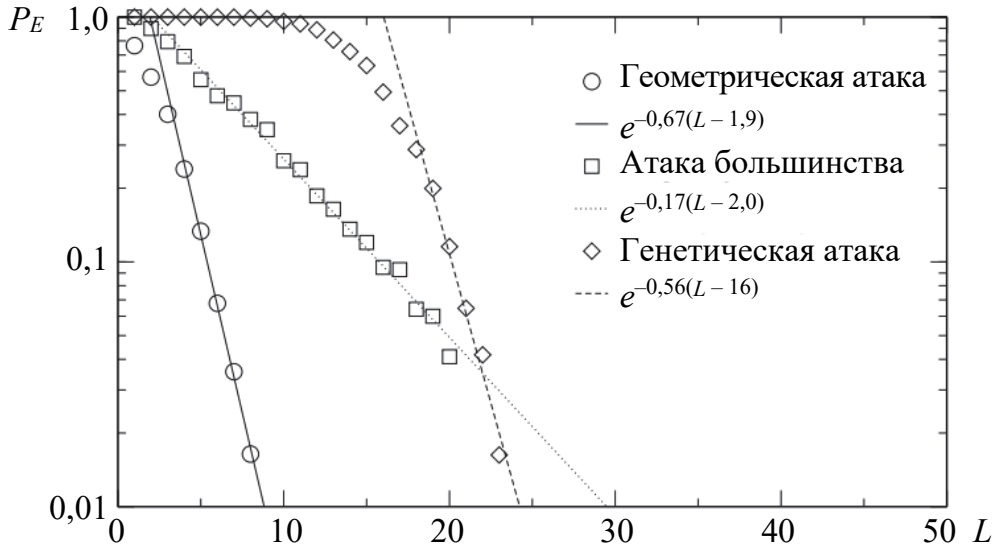

Рис. 3.46. Зависимости, характеризующие эффективность геометрической и генетической атак, а также атаки большинства при $K = 3$ и $N = 1000$

Знаки обозначают результаты, полученные для 1000 симуляций с использованием обучения по правилу случайного блуждания при $K = 3$ и $N = 1000$. Количество атакующих сетей составляет $Y = 4096$ – для генетической атаки и $Y = 100$ – для атаки большинства [42].

3.9.5. Вероятностная атака

Как было описано в подгл. 3.4–3.5 (см. также [34, 35, 97–99]), проще с некоторой вероятностью предсказать положение точки, некоторым случайным образом блуждающей в ограниченном многомерном пространстве и соответствующей значению вектора весовых коэффициентов сети $\langle \mathrm{TPM}(K,N,L)^{A/B} \rangle$ на $(t + 1)$-м шаге обучения, чем определить истинное значение этого вектора или истинное положение точки в указанном пространстве. Эта достаточно очевидная идея лежит в основе вероятностной атаки (Probabilistic Attack) сети $\langle \mathrm{TPM}(K,N,L)^{E} \rangle$ на синхронизируемые сети A и B, предложенной в работе [35] и позднее более детально проанализированной в статье [126].

Формально указанная идея состоит в том, чтобы рассмотреть каждую координату пространства решений отдельно и связать ее с каждым возможным значением ψ из интервала $[-L, \ldots, L]$ через вероятность $p_t(\psi) = P(x_t = \psi)$.

Предполагаем, что изначально $\forall\psi$ вероятность $p_0(\psi) = 1/(2L + 1)$ и после каждого шага (см. подгл. 3.2, п. 3.5.3, 3.7.3.2)

$$p_{t+1}(\psi) = \sum \beta p_t(\beta), \tag{3.59}$$

где β таковы, что если $x_t = \beta$, то $x_{t+1} = \psi$ [35].

Проблема для злоумышленника E заключается в том, что он не знает, какие персептроны в наблюдаемой сети (например, A) обновляют свои веса в каждом раунде.

Предлагается [126] представлять $w_{i,j}$ элементами из *вектора состояния* S ТРМ, состоящего из G элементов ($G >> KN$); $s_i \in \{0, 1\}$. Определяется матрица π размером $N \times K$, содержащая числа $\pi_{ij} \in \{1, \ldots, G\}$, такие что $w_{i,j} = s_{\pi i,j}$. В этом случае процесс синхронизации ТРМ (A и B) с одинаковой архитектурой (N, K и G) будет успешным после нескольких внутренних и внешних раундов (шагов).

Каждый *внутренний раунд* состоит из общеизвестных операций: элементы матрицы π и входные векторы X_j выбираются случайным образом. Затем и A, и B вычисляют τ^A и τ^B, и, если они совпадают ($\tau^A = \tau^B$), состояния σ_1^A и σ_1^B сохраняются в соответствующем буфере. Внутренние раунды повторяются до тех пор, пока буфер не достигнет размера G.

Затем выполняется *внешний раунд*: каждый буфер хранит новый вектор состояния S в соответствующей машине, заменяя старый; после каждого внешнего раунда векторы состояния S^A и S^B становятся ближе, достигая в конечном итоге полной синхронизации: $S^A = S^B$.

ТРМ E также построена на основе (N, K, G)-архитектуры. Злоумышленник использует вероятностный вектор состояния $P_E = (p^1, \ldots, p^G)$ для описания информации о векторе состояния A, S^A. Каждый элемент p_i является аппроксимацией предельной вероятности $P(s_j^A = 0|D)$ того, что i-й бит S^A равен 0, учитывая содержимое всех данных D, наблюдаемых E ранее, т. е. входных и выходных сигналов сетей A и B, которые уже были переданы по общедоступному каналу.

В начале вероятностной атаки предыдущая информация о S^A недоступна. Следовательно, действия E начинаются с нейтральной

гипотезы, и все p_i инициализируются с априорной вероятностью $P(s_j^A = 0) = 1/2$. Открытая информация о состоянии A позволяет E обновлять P_E на основе наблюдаемых данных X, π и τ^A. Для этого апостериорная вероятность

$$P(s_i = 0|(P_E, X, \pi, \tau^A)) = p_i(s_i = 0) \qquad (3.60)$$

оценивается с использованием известного метода Монте-Карло. Соответственно,

$$P(s_i = 1|(P_E, X, \pi, \tau^A)) = 1 - p_i(s_i = 0). \qquad (3.61)$$

Поскольку пространство всех матриц весов W ($W = (W_1, \ldots, W_K)$) имеет размер 2^{NK}, примерно 2^{NK-1} из них совместимы с определенным набором X, π и τ^A. Таким образом, если используемый алгоритм выборки генерирует $Y \geq 2^{NK-1}$ векторов состояния, это будет похоже на *атаку перебором* (грубой силы). Но выбор большого параметра Y возможен только для очень небольшого числа весов.

Определим далее

$$s_i^{E^*} = \begin{cases} 0, p_i > 1/2, \\ 1, p_i \leq 1/2 \end{cases} \qquad (3.62)$$

как наиболее вероятное состояние при условии, что вероятность успеха атаки соответствует P_E; при этом атака считается успешной, как только начнет выполняться условие $S^{E^*} = S^A$.

На рис. 3.47 показана эффективность вероятностной атаки на TPM A и B в зависимости от N при некоторых фиксированных G и $K = 2$.

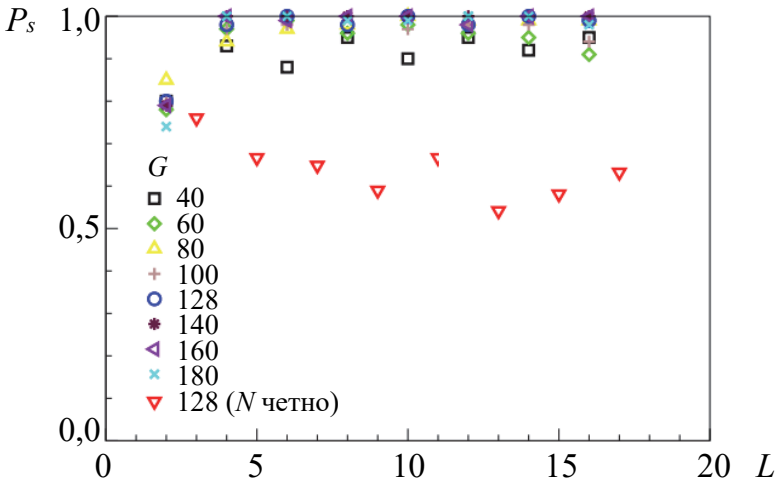

Рис. 3.47. Зависимость эффективности вероятностной атаки от N при фиксированных G (приведены числовые значения) и $K = 2$ [126]

Эта эффективность выражена через вероятность успеха P_s в реализации атаки. Символы обозначают процент успешных атак, обнаруженных в 100 симуляциях. Для нечетных N было выполнено около 25 из 100 опытов. Для четных значений N процесс обучения сети E был остановлен после 30 внешних раундов без получения четкого результата. Эти симуляции не учитывались для расчета вероятности успеха P_s. Что касается случаев с четным N, эффективность алгоритма обычно увеличивается по мере увеличения N или G. Почти для всех показанных здесь конфигураций вероятность успеха P_s атаки превышает 80% и во многих ситуациях фактически достигает 100%. Нечетное N дает вероятность успеха около 50%.

На основании приведенных результатов можно сделать вывод: нейронный протокол обмена криптографическими ключами защищен от всех известных до сих пор атак. Кроме того, при $L/\sqrt{N} \to 0$ эффект от использования трех рассматриваемых правил обучения сходится к результативности применения правила случайного блуждания. Но, как и в случае с другими криптографическими алгоритмами, всегда существует вероятность того, что будет найден хитрый метод, который полностью разрушит безопасность нейронной криптографии.

4. ДРЕВОВИДНЫЕ МАШИНЫ ЧЕТНОСТИ НА ОСНОВЕ АЛГЕБР ГИПЕРКОМПЛЕКСНЫХ ЧИСЕЛ

Развитие архитектур ТРМ должно идти по направлению увеличения сложности вычислительной алгебры, что гарантировало бы больший уровень безопасности. Подавляющее большинство исследований ТРМ-сетей основано на использовании алгебры действительных чисел. Подтверждением тому служат содержание и библиография 3-й главы настоящей книги.

Стоит отметить, что при решении ряда прикладных задач используются различные расширения действительных чисел – *гиперкомплексные числа*, которые позволяют описать положение точки в многомерном пространстве на основе операций над векторами [128]. По теореме Фробениуса единственные гиперкомплексные числа, для которых можно ввести умножение и деление (без делителей нуля), – это *комплексные числа*, *кватернионы* и *числа Кэли* (*октонионы*).

Далее проанализируем использование соответствующих алгебр для согласования весовых коэффициентов между двумя машинами ТРМ, а также оценим безопасность таких систем.

4.1. ТРСМ – ТРМ на основе алгебры комплексных чисел

4.1.1. Математическая модель архитектуры ТРСМ

Нейронные сети на основе алгебры комплексных чисел (Complex-Valued Neural Network, CVNN) начали применять достаточно давно в области телекоммуникаций, робототехнике, биоинформатике, обработке изображений, распознавании речи, машинном обучения и др. [129–132].

Естественно, в соответствии с определением ТРМ, действительные и мнимые части комплексных чисел (каноническая форма записи: $z = a_1 + \boldsymbol{i}a_2$; где $\boldsymbol{i}^2 = -1$, $a_1 = \mathrm{Re}\, z$, $a_2 = \mathrm{Im}\, z$)[5], применяемых в данной архитектуре, должны быть целыми числами.

[5] Далее по тексту будут встречаться двойные значения символа «*i*»: как индекс (использовался по ходу предыдущего материала) и как мнимая единица (символ используется традиционно). Полагаем, у читателя не возникнут проблемы с толкованием и смыслом выражений, содержащих указанный символ. Для ясности мнимые компоненты гиперкомплексных чисел будем выделять «жирным» шрифтом.

Эта архитектура изначально получила условное название TPCM (Tree Parity Complex Machine, древовидная машина четности на основе комплексных чисел) [79, 80, 100, 112, 124, 133, 134].

В более поздней[6] публикации [135] подобная архитектура названа как CVTPM (Complex-Valued Tree Parity Machine).

Сама архитектура сети TPCM, как и идея ее функционирования, схожа с архитектурой TPM. Изменения касаются методов применения правила обучения и модификации функции знака. Таким образом, архитектура TPCM состоит из двух уровней. Элементы первого уровня – это персептроны, имеющие N-элементные векторы весов ($[w_{i1}, w_{i2}, \ldots, w_{iN}]$, где $1 \le i \ge K$), величины которых – это комплексные числа. Как и в случае архитектуры TPM, вышеуказанные элементы ограничены диапазоном $[-L, L]\times[-L, L]$ (который является естественным расширением диапазона $[-L, L]$).

Определение 4.1. Две НС (A и B) на основе архитектуры TPCM, характеризующиеся одинаковыми параметрами (K, N, L) и предназначенные для согласования криптографических ключей на основе алгебры комплексных чисел, будем обозначать соответственно $\langle \mathbf{TPCM}\ (\boldsymbol{K}, \boldsymbol{N}, \boldsymbol{L})^{\boldsymbol{A}} \rangle$, $\langle \mathbf{TPCM}\ (\boldsymbol{K}, \boldsymbol{N}, \boldsymbol{L})^{\boldsymbol{B}} \rangle$.

Вход персептронов составляют K N-элементных векторов $[x_{i1}, x_{i2}, \ldots, x_{iN}]$, где $1 \le i \ge K$ и $x_{ij} = (a_1)_x + \boldsymbol{i}(a_2)_x$, часто отождествляемых с одним NK-элементным вектором $[x_1, x_2, \ldots, x_{kN}]$ четырехвалентных комплексных чисел, выбранных из множества $\{(1, 1), (-1, 1), (-1, -1), (1, -1)\}$, а каждый вес $w_{ij} = (a_1)_{ij} + \boldsymbol{i}(a_2)_{ij}$; $w_{ij} \in [-L, L]\times[-L, L]$; в графическом представлении – это все точки с целочисленными координатами по обеим осям, расположенные в квадрате с угловыми координатами: $(-L, L)$, (L, L), $(L, -L)$, $(-L, -L)$.

Выходы персептронов – это четырехвалентные комплексные числа, принадлежащие множеству $\{(1, 0), (0, 1), (-1, 0), (0, -1)\}$ и обозначенные через $\sigma_1, \sigma_2, \ldots, \sigma_K$.

Модель рассматривается нами в традиционном ее толковании: как зависимость выходных значений от входных с учетом некоторых преобразований, условно обозначаемых в общем случае «черным ящиком».

---

[6] Вероятно, авторы, ограничившись анализом лишь англоязычных источников, определили свою статью как открывающую новое направление: TPM на основе алгебры комплексных чисел («…In this technical note, a neural cryptography based on the complex-valued tree parity machine network (CVTPM) is proposed. …»).

Выход τ архитектуры TPCM вычисляется аналогично архитектуре TPM – на основе формул (3.7) и (3.10) отдельно для каждой части комплексного числа: действительной ($\boldsymbol{R}$) и мнимой ($\boldsymbol{I}$)[7]:

$$\tau^{A/B} = \prod_{i=1}^{K} \boldsymbol{R}(\sigma(\alpha_i^{A/B})) + \boldsymbol{i}\prod_{i=1}^{K} \boldsymbol{I}(\sigma(\alpha_i^{A/B})). \tag{4.1}$$

Ввиду специфики алгебры комплексных чисел, как расширения действительных чисел, любая производимая с ними математическая операция сохраняет свой смысл и вычислительную корректность. Изменению же будет подвергаться функция знака. В случае целых чисел мы определили эту функцию как модифицированную: она может принимать два значения: (–1, 1). В случае комплексных чисел это будут четыре возможных варианта: (1, 0), (0, 1), (–1, 0), (0, –1). Следовательно, учитывая размещение этих величин, нужно произвести соответствующее разделение плоскости [80]. Таким образом, модифицированная функция знака вычисляется по следующей системе:

$$\sigma(\alpha_i) = \begin{cases} (1,0), \dfrac{7\pi}{4} < \arg(\alpha_i) \le \dfrac{\pi}{4}, \\ (0,1), \dfrac{\pi}{4} < \arg(\alpha_i) \le \dfrac{3\pi}{4}, \\ (-1,0), \dfrac{3\pi}{4} < \arg(\alpha_i) \le \dfrac{5\pi}{4}, \\ (0,-1), \dfrac{5\pi}{4} < \arg(\alpha_i) \le \dfrac{7\pi}{4}. \end{cases} \tag{4.2}$$

Рассматриваемая модель должна строиться с учетом некоторых ограничивающих параметров. Основа такого ограничения в данном случае связывается с правилом обучения сети. Положим, что используется правило Хэбба на основе формулы (4.3):

$$w_{ij}^{A/B} = \begin{cases} w_{ij}^{A/B} + \tau^{A/B} x_{ij}, \ \tau^A = \tau^B \wedge \tau^{A/B} = \sigma_i^{A/B}, \\ w_{ij}^{A/B}, \text{ в противном случае.} \end{cases} \tag{4.3}$$

Изменений же требует действие, налагающее ограничения на величины элементов вектора весов. Оно будет происходить в два

[7] Далее действительную и мнимые части гиперкомплексных чисел будем обозначать одним символом: Re – $\boldsymbol{R}$, Im – $\boldsymbol{I}$ и т. д. (см., например, (4.13)–(4.16) и далее).

этапа, отдельно для каждой части комплексного числа согласно следующим формулам:

$$R(w_{ij}^{A/B}) = \begin{cases} \text{sign}(R(w_{ij}^{A/B}))L, \left|R(w_{ij}^{A/B})\right| > L, \\ R(w_{ij}^{A/B}), \text{ в противном случае,} \end{cases} \quad (4.4)$$

$$I(w_{ij}^{A/B}) = \begin{cases} \text{sign}(I(w_{ij}^{A/B}))L, \left|I(w_{ij}^{A/B})\right| > L, \\ I(w_{ij}^{A/B}), \text{ в противном случае.} \end{cases} \quad (4.5)$$

Процесс обучения сети на основе архитектуры TPCM происходит таким же образом, как и на основе TPM. Этот процесс, однако, учитывает и вышеуказанные формулы, и тот факт, что входные величины (вектор X) принадлежат следующим множествам: {(1, 1), (–1, 1), (–1, –1), (1, –1)}. В графическом виде эти числа представлены четырьмя точками, размещенными по углам квадрата (рис. 4.1).

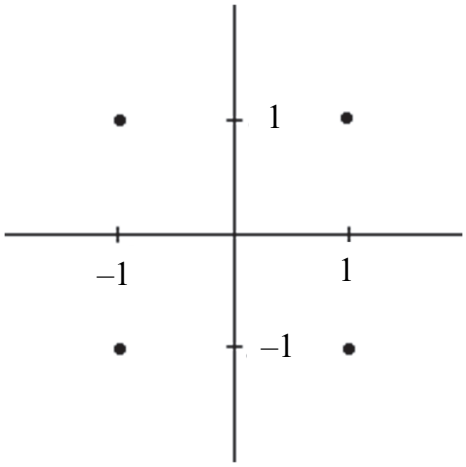

Рис. 4.1. Графическое представление входных величин вектора X

Величины векторов весов – это комплексные числа, заключенные в квадрате $[-L, L]\times[-L, L]$. Следовательно, их графическое представление будет иметь вид, показанный на рис. 4.2.

Выходы персептронов – это комплексные числа, размещенные по углам квадрата (повернутого на 45°). Выходные сигналы TPCM, как произведение выходных величин персептронов, также принадлежат множеству, представленному на рис. 4.3.

Модифицированная функция знака σ упрощенно представлена на рис. 4.4, где уровни выходных сигналов обозначены точками, находящимися в соответствующих четвертях.

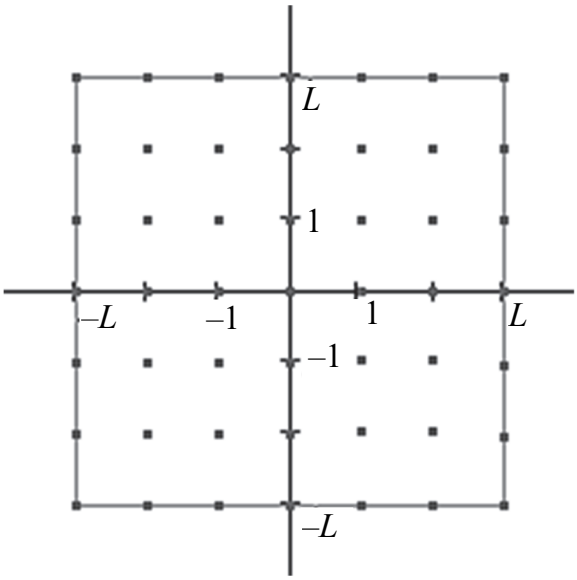

Рис. 4.2. Точки, которые могут составлять элементы вектора весов

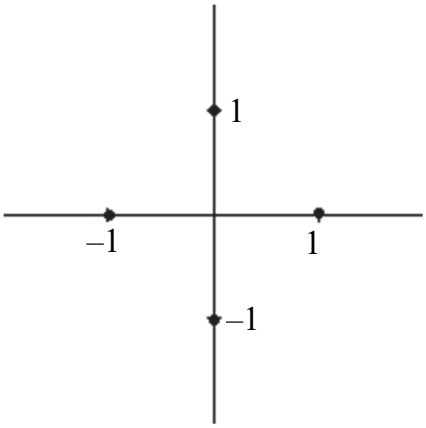

Рис. 4.3. Точки, отображающие выходные сигналы персептронов

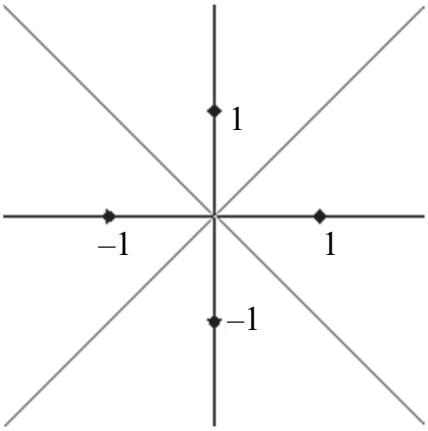

Рис. 4.4. Упрощенное графическое представление функции знака

Целесообразно представить формально соотношения, характеризующие более подробно анализируемое в отношении сети ТРСМ

правило обучения по Хэббу с использованием действительной ($\boldsymbol{R}$) и мнимой ($\boldsymbol{I}$) частей этих чисел (см. также формулу (3.25)):

$$\left.\begin{aligned}
&\boldsymbol{R}(w_{ij}^{(t+1)}) = g[\boldsymbol{R}(w_{ij}^{(t)}) + (\boldsymbol{R}(x_{ij}\tau^{(t)}))\Theta(\boldsymbol{R}(\tau^{(t)})\boldsymbol{R}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{R}(\tau^{A(t)})\boldsymbol{R}(\tau^{B(t)}))], \\
&\boldsymbol{I}(w_{ij}^{(t+1)}) = g[\boldsymbol{I}(w_{ij}^{(t)}) + (\boldsymbol{I}(x_{ij}\tau^{(t)}))\Theta(\boldsymbol{I}(\tau^{(t)})\boldsymbol{I}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{I}(\tau^{A(t)})\boldsymbol{I}(\tau^{B(t)}))].
\end{aligned}\right\} \qquad (4.6)$$

В формулах (4.6) $\Theta(x)$ – функция Хевисайда (см. соотношения (1.4), (1.27)), которая принимает нулевое значение при $x < 0$ и единичное – в остальных случаях; $g(x)$ также определяется стандартным образом (см. формулу (3.23)).

Главным элементом, влияющим на безопасность процесса синхронизации машин A и B на основе TPCM-архитектуры, является тот факт, что выходная величина не определяется напрямую через выходные величины отдельных персептронов σ_i. Например, в классической TPM с $K = 3$ для каждой выходной величины τ существуют четыре возможные комбинации сигналов на выходах персептронов. Например, для $\tau = 1$ это (1, 1, 1), (–1, –1, 1), (1, –1, –1), (–1, 1, –1).

В TPCM-архитектуре количество таких комбинаций сигналов на выходах элементов внутреннего слоя, формирующих один из выходных сигналов всей сети, возрастает квадратично. Например, для $K = 3$ существует 16 возможных комбинаций: при $\tau = 1$ – (1, 1, 1), (–1, –1, 1), (1, –1, –1), (–1, 1, –1), (***i***, ***i***, –1), (***i***, –1, ***i***), (–1, ***i***, ***i***), (–***i***, ***i***, 1), (–***i***, 1, ***i***), (***i***, 1, –***i***), (***i***, –***i***, 1), (1, ***i***, –***i***), (1, –***i***, ***i***), (–***i***, –***i***, –1), (–***i***, –1, –***i***), (–1, –***i***, –***i***)[8] (для упрощения мы использовали каноническую запись комплексных чисел). Следовательно, можно полагать, что системы обмена криптографическими ключами, основанные на архитектуре $\langle \text{TPCM}(K, N, L)^{A/B}\rangle$, будут характеризоваться более высоким уровнем безопасности, чем их эквиваленты, использующие архитектуру $\langle \text{TPM}(K, N, L)^{A/B}\rangle$.

Дополнительным плюсом архитектур TPCM является бóльшая свобода в выборе конструктивных параметров, определяющих процесс синхронизации [136]. Именно в классической архитектуре TPM область, ограничивающая рост величины весов, может быть представлена только с помощью диапазона $[-L, L]$. Это сопряжено

[8] Здесь ***i*** – элемент комплексного числа.

со спецификой алгебры целых чисел, которые, как мы ранее отмечали, ввиду взаимного однозначного отображения могут быть отождествлены с точками на числовой оси. Такие ограничения, налагаемые на величины вектора весов в машинах ТРСМ, являются более гибкими. Это сопряжено с тем фактом, что комплексные числа через взаимно однозначное отображение могут быть отождествлены с точками на плоскости. Отсюда вытекает, что любая двумерная фигура может быть множеством, ограничивающим порядок величин, формирующих вектор весов. Пример такого множества («колесо») представлен на рис. 4.5.

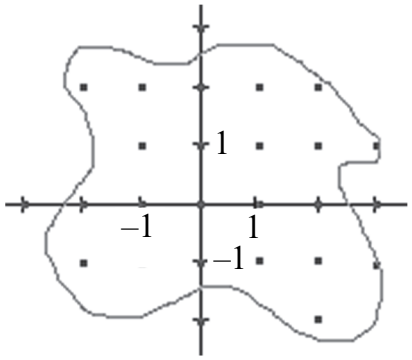

Рис. 4.5. Примерная область вероятных (приемлемых) точек для вектора весов ТРСМ

Выясним теперь, почему оппонент *E*, который наблюдает за обменом информацией между двумя сетями *A* и *B*, не в состоянии синхронизировать свои векторы весов с наблюдаемыми сетями.

Вероятность того, что сети ТРМ (*A* и *B*) активизируют свои векторы весов в одном направлении, определяется соотношением (3.28); если вероятность наступления согласованного движения между сетями *A* и *E* равна *p*, можно воспользоваться формулой (3.53).

Несколько иначе обстоит ситуация в сетях ТРСМ, хотя базовые положения практически повторяются. Рассмотрим это на примере. Для простоты возьмем архитектуру ТРСМ с $K = 2$ и определим по аналогии с формулой (3.48) следующую вероятность:

$$p_i = P[\sigma_i^A = \sigma_i^B], \; i = 1, 2. \qquad (4.7)$$

Для простоты примем также, что $p_1 = p_2 = p$. Получим четыре возможных варианта: (3.49)–(3.52).

Им соответствуют следующие вероятности: p^2, $p(1-p)$, $(1-p)p$ и $(1-p)^2$. Вариант (3.49) возникает в ситуации, когда $\tau^A = \tau^B$. Варианты (3.50) и (3.51) – в ситуациях, где $\tau^A \neq \tau^B$, следовательно, веса обеих ТРСМ не активизируются. Однако вариант (3.52) в 1/3 случаев возможен, если $\tau^A = \tau^B$, в остальных 2/3 случаев – при $\tau^A \neq \tau^B$. Следовательно, вероятность того, что $\langle$ТРСМ $(K, N, L)^A\rangle$ и $\langle$ТРСМ $(K, N, L)^B\rangle$ активизируют свои векторы весов в одном направлении, равна

$$\frac{p^2}{p^2 + (1/3)(1-p)^2}, \tag{4.8}$$

если вероятность наступления согласованного движения между сетями $\langle$ТРСМ $(K, N, L)^A\rangle$ и $\langle$ТРСМ $(K, N, L)^E\rangle$ равна p. Легко доказать, что

$$\frac{p^2}{p^2 + (1/3)(1-p)^2} > \frac{p^2}{p^2 + (1-p)^2}. \tag{4.9}$$

Отсюда следует, что сети A и B на основе архитектуры ТРСМ должны быстрее достигать состояния синхронизации относительно оппонента E, чем сети A, B и E, основанные на архитектуре ТРМ.

4.1.2. Анализ процесса синхронизации ТРСМ и его безопасности

4.1.2.1. Особенности компьютерной реализации имитационной модели ТРСМ. Ввиду того что архитектура ТРСМ не является классической реализацией нейронной сети, реализована ее приблизительная модель. Первый уровень архитектуры ТРСМ составляет двухслойный персептрон.

Входные сигналы относятся к первому слою сети. Этот подход удобен ввиду логики представления нейронных сетей как в математической форме, так и в графической. И все же такой способ представления, встречающийся во многих программных приложениях (имплементации нейронных сетей), не является лучшим. Это связано с эффективностью кода, поскольку каждый элемент в виде функции или объекта существенно замедляет производительность программы. Поэтому в программной реализации также используется один слой нейронов.

Программа (*RCQO*) основывается на использовании объектно-ориентированного подхода. В настоящее время это доминирующее

течение, которое очень хорошо применяется в больших проектах, реализуемых многими программистами. Его главным плюсом является конструктивная логика, которая облегчает написание кода, делая его простым, легким для понимания и дальнейших модификаций.

Для программной реализациях рассматриваемой модели выбран язык C++ ввиду многих его достоинств. Используется шаблонный класс *Complex* (определенный в стандартном пространстве), воспроизводящий комплексные числа (в данном продукте реализованы также операции над иными гиперкомплексными числами; особенности и результаты использования будут рассмотрены ниже). Ввиду применяемых операторов эти числа можно использовать аналогично действительным числам. Определение класса комплексных чисел с целочисленными коэффициентами выглядит следующим образом:

```
typedef std::complex<int> Cint;
```

Основной элемент каждой из моделируемых нейронных сетей – это нейрон, которому в контексте рассматриваемой технической реализации соответствует персептрон. Код класса *CPerceptron* имеет следующий вид:

```
class CPerceptron
{
 Cint *Weights;
 Cint *Inputs;
 Cint Output;
 void RandomWeights();
public:
 CPerceptron();
 ~CPerceptron();
 Cint GetOutput(Cint *AInputs);
 void AktualizeWeights(Cint OutputTPM);
 int Distance(const CPerceptron &p);
};
```

Метод определения функции знака имеет общий вид, адаптирующий ее к специфике комплексных чисел:

```
if (abs(Output.real())>=abs(Output.imag()))
 {
  if (Output.real()>=0)
   Output._M_re = 1;
  else
```

```
   Output._M_re = -1;
  Output._M_im = 0;
 }
else
 {
 if (Output.imag()>=0)
   Output._M_im = 1;
  else
   Output._M_im = -1;
  Output._M_re = 0;
 }
```

Процедура ограничения коэффициентов вектора весов должна быть реализована отдельно для каждой координаты (комплексного числа):

```
if (Weights[i].real() > L) Weights[i]._M_re = L;
 else
if (Weights[i].real() < -L) Weights[i]._M_re = -L;

if (Weights[i].imag() > L) Weights[i]._M_im = L;
 else
if (Weights[i].imag() < -L) Weights[i]._M_im = -L;
```

Остальные изменения менее существенны, но также необходимы. Так, например, расстояние в двумерном пространстве определено с помощью функции *norm*.

Таким образом, на основе модифицированного персептрона образован класс *TPCM*. Процесс синхронизации векторов весов обеих архитектур происходит с использованием алгебры комплексных чисел на основе следующего кода:

```
const int NC = N/2;
class TPCM
{
 CPerceptron CPerceptrons[K];
 Cint Output;
public:
 Cint GetOutput(Cint AInputs[K][NC]);
 void Synchronize();
 void ModifyOutput(Cint AOutput){Output = AOutput;};
 int Distance(const TPCM &ATPCM);
};
```

Полный программный код реализованной модели TPCM в составе приложения *RCQO* представлен в приложении А. Для примера

или сравнения можно обратиться к работе [137], где размещен код для реализации модели на основе TPM.

Другое программное средство (*RCQOv.2*), использованное, в частности, при проведении исследований, результаты которых излагаются, например, в работах [106, 113], предназначено для реализации математических операций, а также систематизации и анализа полученных результатов моделирования процессов синхронизации на основе алгебр гиперкомплексных чисел. Предметное обсуждение использования приложения *RCQOv.2* будет приведено далее.

4.1.2.2. Сравнительный анализ процессов синхронизации архитектур TPM и TPCM. Статистические параметры и свойства распределений, относящихся к синхронизации сетей TPM, достаточно подробно проанализированы в главе 3. В настоящем подразделе рассмотрим некоторые важные практические аспекты, относящиеся к процедуре синхронизации сетей $\langle \mathrm{TPSCM}(K=3, N=10, L=6)^{A/B} \rangle$, $\langle \mathrm{TPCM}(K, N, L)^{B} \rangle$, а также оценим полученные результаты в сравнении с сетями $\langle \mathrm{TPM}(K, N, L)^{A/B} \rangle$.

В табл. 4.1 приведены данные о результатах симуляций процесса синхронизации TPCM, полученные с использованием приложения *RCQOv.2*.

Таблица 4.1

Статистические результаты симуляций процесса синхронизации TPCM

N	K	$\pm L$	Количество симуляций	Количество успешных синхронизаций	t_{synch}, шагов (среднее)	Среднеквадратичное отклонение	Индекс Брея – Кертиса
6	7	6	1000	993	227,4	76,2	0,180
6	6	7	1000	972	318,6	162,0	0,254
5	6	5	1000	944	186,1	65,2	0,185
6	5	5	1000	925	149,2	55,5	0,200
5	5	6	1546	1285	230,4	80,7	0,182
5	5	5	1668	1409	176,2	64,3	0,204
6	6	6	1079	1061	212,1	73,4	0,185
7	7	7	1011	970	280,8	91,6	0,174
7	6	6	1000	929	211,1	68,6	0,175

Дополнительно на рис. 4.6 приведена гистограмма распределения числа успешно синхронизировавшихся машин TPCM (по горизонтали) по числу шагов до наступления состояния синхронизации t_{synch} сетей $\langle \mathrm{TPCM}(K=5, N=5, L=5)^{A/B} \rangle$.

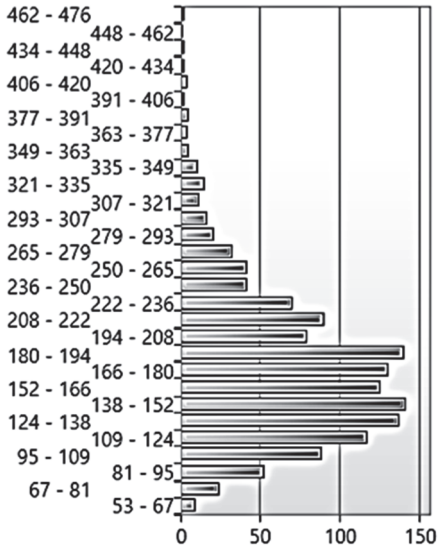

Рис. 4.6. Распределение количества успешно синхронизировавшихся машин ТРСМ по числу шагов до синхронизации

Это же распределение с дополнительными его характеристиками в окне приложения *RCQOv.2* приведено на рис. 4.7.

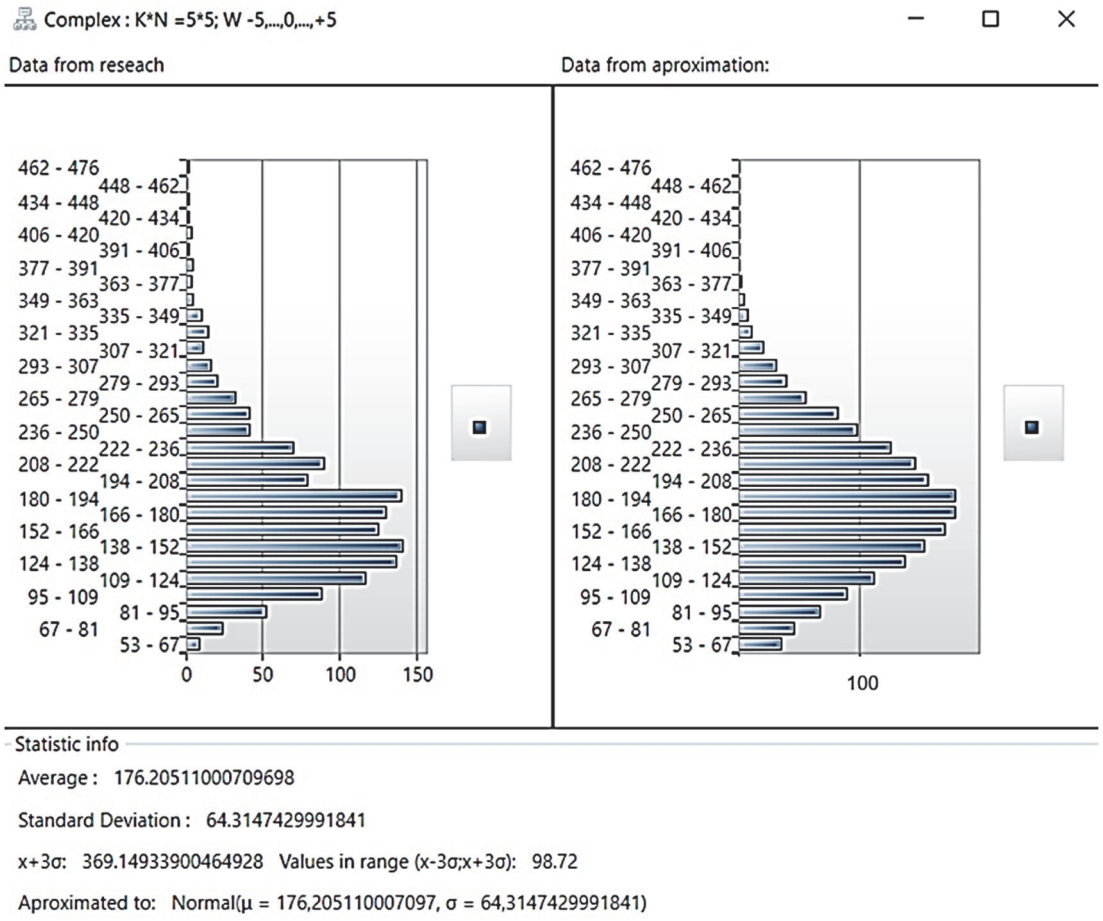

Рис. 4.7. Окно приложения *RCQOv.2*

Даже визуально представленное распределение похоже на нормальное (представлено в правой части окна на рис. 4.7). Чтобы оценить степень визуальной схожести, использовалась оценка в виде расстояния, или *несходства Брея – Кертиса* (Bray – Curtis) [106, 113, 137]. Эта характеристика представляет собой статистику, используемую для количественной оценки несходства между двумя разными выборками, и находится в диапазоне от 0 до 1: 0 означает полное совпадение (сходство), 1 – полное несходство.

Данные в крайнем правом столбце табл. 4.1 свидетельствуют о высокой степени соответствия полученных распределений нормальному распределению. Дополнительно установлено, что в диапазон «трех сигм» попадает 98,72% успешно синхронизировавшихся нейронных сетей (вторая строка снизу на рис. 4.7); для нормального распределения этот показатель составляет 99,72%. Высокую степень соответствия нормальному закону мы отмечали выше, анализируя подобные распределения для сетей TCP.

Обратимся к сравнению временно́й характеристики рассматриваемого типа распределения для машин TPM и TPCM с одинаковой архитектурой. Для примера сети характеризуются следующими параметрами: $\langle \text{TPM } (K = 3,\ N = 20,\ L = 3)^{A/B} \rangle$, $\langle \text{TPCM } (K = 3,\ N = 10,\ L = 3)^{A/B} \rangle$ [80]. Различие в значении N возникает вследствие того, что комплексное число состоит из двух целых чисел. Следовательно, целые числа из общего, входного вектора X будут сгруппированы в пары. Установлено, что отношение времени синхронизации t_{synch} сетей $\langle \text{TPCM } (K = 3, N = 10, L = 3)^{A/B} \rangle$ ко времени синхронизации (обучения) t_{learn} сетей $\langle \text{TPCM } (K = 3, N = 10, L = 3)^{A/E} \rangle$ составляет 0,0602, а для сетей на основе действительных чисел – 0,207. Понятно, что это отношение существенно влияет на уровень безопасности рассматриваемого процесса. Однако, с другой стороны, увеличивается и время синхронизации (количество шагов): соответственно 14 812 и 214 (см. табл. 3.10 на с. 80).

Интерес представляет также сравнение времен синхронизации t_{synch} TPM и TPCM при различной длине формируемого ключа (длине вектора весов). В табл. 4.2 представлена соответствующая статистика, полученная на основе правила Хэбба с использованием приложения *RCQO* [101] (для TPM-архитектуры подобная информация приведена в п. 3.5.1) при $L \in [-3, 3]$. Во второй строке заголовка таблицы указана длина формируемого ключа (в символах).

Таблица 4.2

Сравнительная статистика синхронизации TPM и TPCM

Архитектура	t_{synch}		
	48	66	112
TPM	22,2	28,4	40,3
TPCM	1800,3	2270,5	3287,9

Как мы отмечали, архитектура TPCM отличается большей свободой в выборе ограничения, налагаемого на величины вектора весов: в случае TPM данное ограничение представляет собой простой промежуток, а в случае TPCM им может быть любая двумерная фигура. Мы проследим за скоростью процесса синхронизации для двух указанных типов ограничений, наложенных на вектора весов в архитектуре TPCM. Эти параметры мы также сравним по отношению к архитектуре TPM.

Для тестирования используем три разных типа архитектур. Первый тип – классическая архитектура TPM, второй – TPCM. Третий – также основан на TPCM, но с той разницей, что он устанавливает разные типы ограничения вектора весов (TPCM1 и TPCM2).

В первоначальной модели TPCM (TPCM1) коэффициенты вектора весов ограничены в диапазоне (квадрате) $[-L, L]\times[-L, L]$. Формальное представление анализируемых сетей: $\langle$TPCM ($K = 3$, $N = 6$, $L = 3)^{A/B}\rangle$ и $\langle$TPM ($K = 3$, $N = 12$, $L = 3)^{A/B}\rangle$. Разница в величине параметра N обусловлена тем, что комплексное число представляется парой целых чисел. Следовательно, входные векторы X для TPCM будут вдвое короче относительно векторов для TPM. Кроме этого, для второго типа архитектуры TPCM (TPCM2) параметр $L = 4$, благодаря чему окружность будет содержать столько же точек, что и квадрат $[-3, 3]\times[-3, 3]$.

Усредненные результаты моделирования результатов взаимодействия сетей A, B и E по 1000 опытам приведены в табл. 4.3 [136].

Таблица 4.3

Сравнение времени синхронизации архитектур TPM и TPCM (с двумя типами ограничения весовых коэффициентов)

Архитектура	t_{synch}, шагов	t_{learn}, шагов	t_{synch}/t_{learn}
TPM	278,1	1209,3	0,230
TPCM1	2695,8	30 428,2	0,089
TPCM2	17 668,1	408 931	0,043

Из табл. 4.3 следует, что архитектура TPCM позволяет существенно (примерно в 2,5–3,0 раза) увеличить уровень безопасности системы (данный уровень измерен отношением, представленным в четвертом столбце таблицы).

Кроме этого, в самой TPCM существуют различия, являющиеся следствием выбора соответствующего ограничения, наложенного на коэффициенты вектора весов. Как видно, даже простое изменение ограничения существенным образом влияет на безопасность системы.

4.1.2.3. Анализ эффективности геометрической атаки на TPCM.

В п. 3.9.2 описаны особенности и эффективность геометрической атаки на сети TPM. Этот тип атаки является одним из простейших в реализации. Оценим теперь результативность реализации такой атаки на архитектуры TPCM. Обучение основано на правиле Хэбба. Реализация рассматриваемого типа атаки на основе алгебры комплексных чисел диктует необходимость изменения проанализированного в п. 3.9.2 алгоритма и его адаптации к специфике архитектуры TPCM.

Так как функция знака σ возвращает четыре величины, принадлежащие множеству $\{1, \boldsymbol{i}, -1, -\boldsymbol{i}\}$, то она делит плоскость на четыре части, представленные выше на рис. 4.3 и 4.4.

В анализируем алгоритме мы ищем вектор W_i, соответствующая которому выходная величина σ_i близка не к 0, а к одной из линий деления плоскости. В случае геометрической атаки для архитектуры TPM было достаточно заменить величину σ_i найденного вектора W_i на противоположную по знаку. Архитектура TPCM требует выполнения более сложных преобразований: модифицированная выходная величина вектора W_i не всегда ведет к равенству $\tau^A = \tau^E$. Следовательно, надо ограничить область искомых внутренних векторов только теми, для которых изменение данной величины будет гарантировать равенство $\tau^A = \tau^E$.

Алгоритм геометрической атаки для архитектур TPCM выглядит следующим образом.

1. Если $\tau^A \neq \tau^B$, то сеть E, так же как A и B, не производит активизацию внутреннего вектора весов.

2. Если $\tau^A = \tau^B$ и $\tau^A = \tau^E$, то оппонент E активизирует свой внутренний вектор весов в соответствии с выбранным правилом обучения.

3. Если $\tau^A = \tau^B$ и $\tau^A \neq \tau^E$, тогда E находит вектор W_i, длина которого $\|Re(\alpha_i)\| - \|Im(\alpha_i)\|$ ◘ наименьшая (или близка к одной из

линии деления плоскости). Область поиска ограничивается только теми векторами W_i, для которых изменение или перенесение величины в соседнюю четверть гарантировано приведет к $\tau^A \neq \tau^E$. Этот процесс переноса обозначает умножение σ_i (подчеркнем: здесь i – индекс) на $\boldsymbol{i}$ или $-\boldsymbol{i}$ (здесь $\boldsymbol{i}$ – символ канонической формы записи комплексного числа) в зависимости от направления переноса найденной величины. Затем, благодаря этой операции, сеть E активизирует свой внутренний вектор весов.

По результатам экспериментов вероятность успеха геометрической атаки ($P_E = t_{\text{synch}}/t_{\text{learn}}$) на сети $\langle \text{TPM } (K = 3, N = 12, L = 3)^{A/B} \rangle$ составляет 0,58, а на сети $\langle \text{TPCM } (K = 3, N = 6, L = 3)^{A/B} \rangle$ – $P_E < 0{,}002324$ (см. также табл. 4.4).

В табл. 4.4 приведены для сравнения результаты компьютерного моделирования геометрической атаки с использованием приложения *RCQO*.

Таблица 4.4

Сравнение эффективности геометрической атаки в контексте безопасности архитектур TPM и TPCM

Архитектура	t_{synch}	t_{learn}	P_E
TPM	222,9	387,6	0,58
TPCM	2324,4	1 000 000[9]	<0,002324

Как показали тесты (табл. 4.3), геометрическая атака очень опасна в процессе синхронизации сетей на основе архитектуры TPM (см. рис. 3.42 (с. 133) и 3.46 (с. 139)). В то же время модель на основе TPCM характеризуется очень высокой степенью устойчивости к данным атакам. Даже значительное количество шагов (1 000 000) не ведет к синхронизации вектора весов сети интруза с сетями A и B [124, 133].

В работах [139, 140] рассматриваются особенности применения *сплит-комплексных чисел* (Split-Complex Number) при реализации сетей TPM. Такие сети авторы определили как $\langle \text{TPSCM } (K, N, L)^{A/B} \rangle$ (Tree Parity Split-Complex Machine).

Сплит-комплексные (*разделенные комплексные* или *двойные*) числа, как и комплексные числа, имеют две компоненты вещественного числа и формально записываются одинаково: $z = a_1 + \boldsymbol{j}a_2$. Основная

[9] 1 000 000 – максимальное произведенное количество шагов, при котором оппонент E не смог синхронизироваться с наблюдаемыми сетями A и B.

разница состоит в том, что $\boldsymbol{j}^2 = 1$. Кроме того, для двойных чисел определены только правила сложения, вычитания и умножения [128].

Входами сети являются векторы четырехзначных двойных чисел, выбранных из множества {(1, 1), (–1, 1), (–1, –1), (1, –1)}. Выходы σ_1, σ_2, …, σ_K нейронов представляют собой комплексные числа, принадлежащие множеству {(1, 0), (0, 1), (–1, 0), (0, –1)}.

Выход архитектуры TPSCM рассчитывается так же, как и для TPM, и для TPCM – по формуле (4.1), а модификация функции знака – так же, как и для TPCM: на основе выражения (4.2).

Сравнительная оценка безопасности TPSCM может быть получена на основе соотношений (4.8) и (4.9). В работе [140] приведены данные, характеризующие сравнение сетей $\langle \text{TPM}\,(K = 3,\ N = 20,\ L = 3)^{A/B} \rangle$ и $\langle \text{TPSCM}(K = 3, N = 10, L = 6)^{A/B} \rangle$, обученных по правилу Хэбба (табл. 4.5).

Таблица 4.5

Сравнение безопасности архитектур TPM и TPSCM

Архитектура	t_{synch}	t_{learn}	P_E
TPM	1156,6	25 721,9	0,045
TPSCM	18 631,3	798 235,9	0,023

4.2. TPQM – TPM на основе алгебры кватернионов

4.2.1. Структура и математическая модель сетей TPQM

Кватернионы были введены в математику У. Гамильтоном (W. Hamilton). С формальной точки зрения алгебра на основе этих чисел сформирована на «размещении» комплексных чисел и комплексной плоскости в трехмерном пространстве с добавлением к $\boldsymbol{i}$ еще двух мнимых чисел (обычно их обозначают символами $\boldsymbol{j}$ и $\boldsymbol{k}$) [141].

Кватернионы можно определить в виде следующей суммы:

$$q = a_1 + a_2\boldsymbol{i} + a_3\boldsymbol{j} + a_4\boldsymbol{k};\ \ a_1, a_2, a_3, a_4 \in \mathbb{R}. \qquad (4.10)$$

Здесь символы $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ – мнимые единицы[10]. Отношения между $\boldsymbol{i}$, $\boldsymbol{j}$ и $\boldsymbol{k}$ очень похожи на правила векторного умножения единичных декартовых векторов: кватернион – это вектор 4-мерного векторного

[10] Также общепринятые обозначения.

пространства над базисом $\{1, \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$[11]. Особенность кватернионов состоит в правилах их умножения.

Заметим дополнительно, что каждый кватернион вида (4.10) представляет собой формально сумму действительного числа a_1 с вектором $a_2\boldsymbol{i} + a_3\boldsymbol{j} + a_4\boldsymbol{k}$ (мнимая часть).

Некоторые важные детали использования алгебры кватернионов для решения прикладных задач можно найти, например, в работах [142, 143].

Идея, структура и принципы взаимодействия TPM на основе кватернионов изложены в статье [144], а некоторые результаты их использования и характеристика безопасности – в работах [106, 113, 140]; в них сеть названа TPQM (Tree Parity Quaternion Machine). Достаточно подробный и всесторонний анализ TPM на основе кватернионов выполнен в статье [145]; здесь архитектура получила название Quaternion-Valued TPM (QVTPM).

Далее для анализа будем использовать более раннее название и обозначение: TPQM.

Определение 4.2. Две НС (A и B) на основе архитектуры TPQM, характеризующиеся одинаковыми параметрами (K, N, L) и предназначенные для согласования криптографических ключей на основе алгебры кватернионов, будем обозначать соответственно $\langle\textbf{TPQM}\,(\boldsymbol{K}, \boldsymbol{N}, \boldsymbol{L})^{\boldsymbol{A}}\rangle$ и $\langle\textbf{TPQM}\,(\boldsymbol{K}, \boldsymbol{N}, \boldsymbol{L})^{\boldsymbol{B}}\rangle$.

Архитектура сети TPQM состоит из двух слоев. Первый уровень составляют персептроны, содержащие N-элементные векторы весов ($[w_{i1}, w_{i2}, \ldots, w_{iN}]$, где $1 \le i \le K$), величины которых являются кватернионами. Каждый вес ω_{ij} является кватернионом с коэффициентами (в соответствии с формулой (4.9)) $(a_1)_{ij}$, $(a_2)_{ij}$, $(a_3)_{ij}$ и $(a_4)_{ij}$, тогда

$$w_{ij} = (a_1)_{ij} + (a_2)_{ij}\boldsymbol{i} + (a_3)_{ij}\boldsymbol{j} + (a_4)_{ij}\boldsymbol{k}, \qquad (4.11)$$

где $\{(a_1)_{ij}, (a_2)_{ij}, (a_3)_{ij}, (a_4)_{ij}\} \in \{-L, -L+1, \ldots, L\}$. Элементы входного вектора также являются кватернионами:

$$x_{ij} = (a_1)_x + (a_2)_x\boldsymbol{i} + (a_3)_x\boldsymbol{j} + (a_4)_x\boldsymbol{k}.$$

Как и во всех моделях, на веса налагаются определенные ограничения. В модели TPQM это ограничение будет промежутком $[-L, L]\times[-L, L]\times[-L, L]\times[-L, L]$. Такой способ представления ограничительной области весовых коэффициентов возможен ввиду того

[11] Если в формуле (4.9) $a_3 = 0$ и $a_4 = 0$, получим комплексное число; при $a_2 = 0$ и $a_4 = 0$ – сплит-комплексное.

факта, что между кватернионами и точками пространства $\mathbb{R}^4$ существует взаимно однозначная зависимость. Величины весов являются элементами множества, состоящего из шестнадцати (2^4) разных величин: {(1, 1, 1, 1), (–1, 1, 1, 1), (1, –1, 1, 1), (1, 1, –1, 1), (1, 1, 1, –1), (–1, –1, 1, 1), (–1, 1, –1, 1), (–1, 1, 1, –1), (1, –1, –1, 1), (1, –1, 1, –1), (1, 1, –1, –1), (–1, –1, –1, 1), (–1, –1, 1, –1), (–1, 1, –1, –1), (1, –1, –1, –1), (–1, –1, –1, –1)}.

Выходы нейронов скрытого слоя σ_i, а также выход $\tau^{A/B}$ ⟨TPQM (K, N, L)^A⟩ или ⟨TPQM (K, N, L)^B⟩ также представляются в виде кватернионов: $\sigma_i \in$ {(1, 0, 0, 0), (–1, 0, 0, 0), (0, 1, 0, 0), (0, –1, 0, 0), (0, 0, 1, 0), (0, 0, –1, 0), (0, 0, 0, 1), (0, 0, 0, –1)} или через {1, –1, $\boldsymbol{i}$, –$\boldsymbol{i}$, $\boldsymbol{j}$, –$\boldsymbol{j}$, $\boldsymbol{k}$, –$\boldsymbol{k}$}:

$$\sigma_i = (a_1)_{\sigma_i} + (a_2)_{\sigma_i}\boldsymbol{i} + (a_3)_{\sigma_i}\boldsymbol{j} + (a_4)_{\sigma_i}\boldsymbol{k}, \qquad (4.12)$$

где $(a_1)_{\sigma_i}$, $(a_2)_{\sigma_i}$, $(a_3)_{\sigma_i}$, $(a_4)_{\sigma_i}$ – функция знака для соответствующего компонента кватерниона.

Как известно, кватернионы вместе с операциями сложения и умножения создают кольцо. Следовательно, каждая математическая операция сохраняет свою корректность так же, как и в случае с действительными числами (за исключением коммутативности умножения). Все же функция знака будет подвергаться существенной модификации. Анализируя TPM, мы подчеркивали, что данная функция делит пространство на два непересекающихся подмножества, а в архитектуре TPCM – на четыре. По отношению к TPQM деление будет еще более сложным – имеем восемь непересекающихся подмножеств пространства $\mathbb{R}^4$. В случае кватернионов и пространства $\mathbb{R}^4$ возможности графического представления функции знака не существует. Математически функцию знака σ_i можно представить в следующем виде:

$$\sigma_i = \begin{cases} (1,0,0,0),\ a_1 = \max(\{a_1,a_2,a_3,a_4\}) \wedge a_1 \geq 0), \\ (-1,0,0,0),\ a_1 = \max(\{a_1,a_2,a_3,a_4\}) \wedge a_1 < 0), \\ (0,1,0,0),\ a_2 = \max(\{a_1,a_2,a_3,a_4\}) \wedge a_2 \geq 0), \\ (0,-1,0,0),\ a_2 = \max(\{a_1,a_2,a_3,a_4\}) \wedge a_2 < 0), \\ (0,0,1,0),\ a_3 = \max(\{a_1,a_2,a_3,a_4\}) \wedge a_3 \geq 0), \\ (0,0,-1,0),\ a_3 = \max(\{a_1,a_2,a_3,a_4\}) \wedge a_3 < 0), \\ (0,0,0,1),\ a_4 = \max(\{a_1,a_2,a_3,a_4\}) \wedge a_4 \geq 0), \\ (0,0,0,-1),\ a_4 = \max(\{a_1,a_2,a_3,a_4\}) \wedge a_4 < 0). \end{cases} \qquad (4.13)$$

Такая функция знака будет легко применима в коде любого языка программирования.

Как видим, отдельные точки выполняют роль аттракторов, «притягивая» точки, лежащие ближе всего к одному из возможных

выходных состояний системы. Схематично это может быть представлено рис. 4.8.

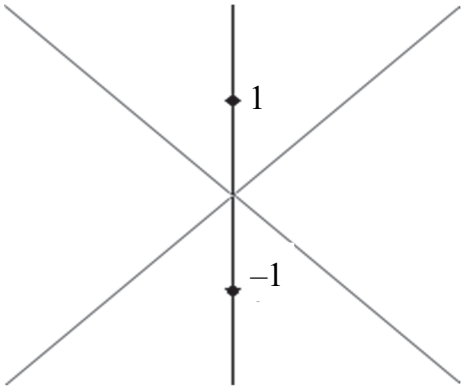

Рис. 4.8. Схематичное представление пары аттракторов в функции знака σ

Эта одна пара (из четырех встречающихся в нашем множестве) вызывает деление пространства на точки, «притягиваемые» к каждому из двух элементов.

Обучение сетей TPQM происходит по аналогии с процессом в модели TPM в соответствии с выбранным правилом. Как отмечено выше, в конструктивной части TPQM ограничением для величины вектора весов будет промежуток $[-L, L]\times[-L, L]\times[-L, L]\times[-L, L]$. Отсюда формально формулы ограничения увеличения значения весов в TPQM можно представить выражениями (4.4), (4.5), к которым следует присоединить соотношения, относящиеся к двум правым частям кватерниона[12]:

$$\left.\begin{aligned} \boldsymbol{J}(w_{ij}^{A/B}) &= \begin{cases} \operatorname{sign}(\boldsymbol{J}(w_{ij}^{A/B}))L, \left|\boldsymbol{J}(w_{ij}^{A/B})\right| > L, \\ \boldsymbol{J}(w_{ij}^{A/B}), \text{ в противном случае,} \end{cases} \\ \boldsymbol{K}(w_{ij}^{A/B}) &= \begin{cases} \operatorname{sign}(\boldsymbol{K}(w_{ij}^{A/B}))L, \left|\boldsymbol{K}(w_{ij}^{A/B})\right| > L, \\ \boldsymbol{K}(w_{ij}^{A/B}), \text{ в противном случае.} \end{cases} \end{aligned}\right\} \qquad (4.14)$$

Однако в анализируемой модели используется тот факт, что входные величины (x_{ij}) принадлежат множеству {(1, 1, 1, 1), …, (–1, –1, –1, –1)}. Как видно, оно заключает все четырехэлементные комбинации векторов на базе двух символов: {1, –1}. Упрощенно положение соответствующих точек отображает рис. 4.9.

[12] В силу использования одинаковых обозначений соответствующих мнимых частей в комплексных числах и кватернионах.

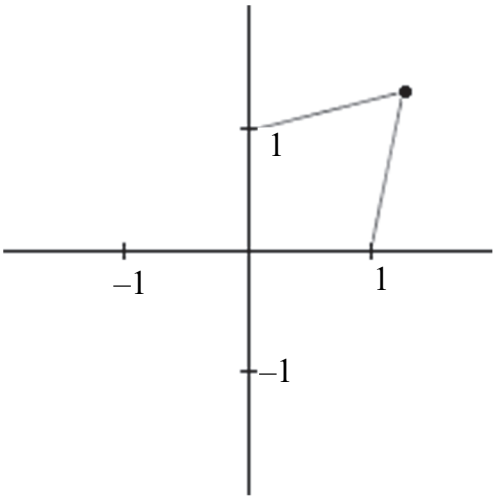

Рис. 4.9. Схематичное представление положения точки, соответствующей входу нейрона

В дополнение к этому еще раз отметим, что величины вектора весов – это кватернионы в коэффициентах, являющихся целыми числами, которые содержатся в гиперкубе. Элементарная ограничительная модель величин весов представлена на рис. 4.10.

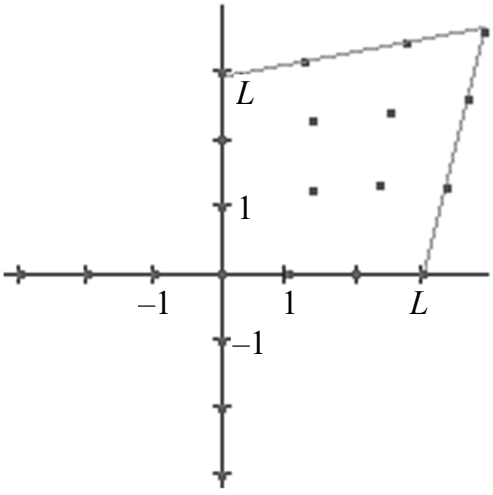

Рис. 4.10. Графическое представление ограничительной модели величин весов в TPQM

Выход сети TPQM определяется в соответствии с формулой

$$\tau^{A/B} = \prod_{i=1}^{K} \boldsymbol{R}\left(\sigma_i^{A/B}\right) + \boldsymbol{i}\prod_{i=1}^{K} \boldsymbol{I}\left(\sigma_i^{A/B}\right) + \boldsymbol{j}\prod_{i=1}^{K} \boldsymbol{J}\left(\sigma_i^{A/B}\right) + \boldsymbol{k}\prod_{i=1}^{K} \boldsymbol{K}\left(\sigma_i^{A/B}\right), \quad (4.15)$$

где символы $\boldsymbol{R}$, $\boldsymbol{I}$, $\boldsymbol{J}$, $\boldsymbol{K}$ (в отличие от K) обозначают соответственно действительную и три мнимые компоненты кватернионов, а $\boldsymbol{i}$ (выделено жирным, в отличие от i), $\boldsymbol{j}$, $\boldsymbol{k}$ – мнимые числа.

В приложении Б приведен листинг программного кода для реализации сети TPQM.

4.2.2. Анализ процесса синхронизации TPQM и его безопасности

Как и в предыдущих архитектурах, начальные значения веторов весов W^A и W^B (см. формулу (4.10)) сетей выбираются случайным образом и хранятся в тайне.

На каждом шаге синхронизации используется соответствующий входной вектор, а векторы весов изменяются в соответствии с используемым правилом обучения.

Для правила обучения по анти-Хэббу по аналогии с формулами (4.6) (см. для сравнения выражение (3.46)):

$$\left.\begin{aligned}
&R(w_{ij}^{(t+1)}) = g[R(w_{ij}^{(t)}) - (R(x_{ij}\tau^{(t)}))\Theta(R(\tau^{(t)})R(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(R(\tau^{A(t)})R(\tau^{B(t)}))],\\
&I(w_{ij}^{(t+1)}) = g[I(w_{ij}^{(t)}) - (I(x_{ij}\tau^{(t)}))\Theta(I(\tau^{(t)})I(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(I(\tau^{A(t)})I(\tau^{B(t)}))],\\
&J(w_{ij}^{(t+1)}) = g[J(w_{ij}^{(t)}) - (J(x_{ij}\tau^{(t)}))\Theta(J(\tau^{(t)})J(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(J(\tau^{A(t)})J(\tau^{B(t)}))],\\
&K(w_{ij}^{(t+1)}) = g[K(w_{ij}^{(t)}) - (K(x_{ij}\tau^{(t)}))\Theta(K(\tau^{(t)})K(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(K(\tau^{A(t)})K(\tau^{B(t)}))];
\end{aligned}\right\} \quad (4.16)$$

для обучения по Хэббу:

$$\left.\begin{aligned}
&R(w_{ij}^{(t+1)}) = g[R(w_{ij}^{(t)}) + (R(x_{ij}\tau^{(t)}))\Theta(R(\tau^{(t)})R(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(R(\tau^{A(t)})R(\tau^{B(t)}))],\\
&I(w_{ij}^{(t+1)}) = g[I(w_{ij}^{(t)}) + (I(x_{ij}\tau^{(t)}))\Theta(I(\tau^{(t)})I(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(I(\tau^{A(t)})I(\tau^{B(t)}))],\\
&J(w_{ij}^{(t+1)}) = g[J(w_{ij}^{(t)}) + (J(x_{ij}\tau^{(t)}))\Theta(J(\tau^{(t)})J(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(J(\tau^{A(t)})J(\tau^{B(t)}))],\\
&K(w_{ij}^{(t+1)}) = g[K(w_{ij}^{(t)}) + (K(x_{ij}\tau^{(t)}))\Theta(K(\tau^{(t)})K(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(K(\tau^{A(t)})K(\tau^{B(t)}))];
\end{aligned}\right\} \quad (4.17)$$

для обучения по методу случайного блуждания:

$$\left.\begin{aligned}
&\boldsymbol{R}(w_{ij}^{(t+1)}) = g[\boldsymbol{R}(w_{ij}^{(t)}) + \boldsymbol{R}(x_{ij})\Theta(\boldsymbol{R}(\tau^{(t)})\boldsymbol{R}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{R}(\tau^{A(t)})\boldsymbol{R}(\tau^{B(t)}))], \\
&\boldsymbol{I}(w_{ij}^{(t+1)}) = g[\boldsymbol{I}(w_{ij}^{(t)}) + \boldsymbol{I}(x_{ij})\Theta(\boldsymbol{I}(\tau^{(t)})\boldsymbol{I}(\sigma_i^{A/B(t)}))\times \\
&\times\ \Theta(\boldsymbol{I}(\tau^{A(t)})\boldsymbol{I}(\tau^{B(t)}))], \\
&\boldsymbol{J}(w_{ij}^{(t+1)}) = g[\boldsymbol{J}(w_{ij}^{(t)}) + \boldsymbol{J}(x_{ij})\Theta(\boldsymbol{J}(\tau^{(t)})\boldsymbol{J}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{J}(\tau^{A(t)})\boldsymbol{J}(\tau^{B(t)}))], \\
&\boldsymbol{K}(w_{ij}^{(t+1)}) = g[\boldsymbol{K}(w_{ij}^{(t)}) + \boldsymbol{K}(x_{ij})\Theta(\boldsymbol{K}(\tau^{(t)})\boldsymbol{K}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{K}(\tau^{A(t)})\boldsymbol{K}(\tau^{B(t)}))].
\end{aligned}\right\} \qquad (4.18)$$

Как и ранее, в формулах (4.16)–(4.18) $\Theta(x)$ – функция Хевисайда; $g(x)$ определяется стандартным образом (см., например, выражение (3.23)).

В табл. 4.6 (см. также [106]) приведены данные о результатах симуляций процесса синхронизации TPQM по правилу Хэбба, полученные с использованием уже упоминавшегося выше специального приложения *RCQOv.2*.

Таблица 4.6

Статистические результаты симуляций процесса синхронизации TPQM

N	K	$\pm L$	Количество симуляций	Количество успешных синхронизаций	t_{synch}, шагов (среднее)	Среднеквадратичное отклонение	Индекс Брея – Кертиса
5	5	5	1357	1354	445,2	212,3	0,25
6	6	6	1070	1068	417,1	147,7	0,18
6	7	7	1000	1000	708,9	273,1	0,19
7	6	7	1000	1000	858,5	385,8	0,23
7	7	7	1000	997	735,7	284,9	0,20
8	8	8	1000	999	780,7	273,6	0,19
9	9	9	1000	1000	1550,6	605,6	0,20

Сравнивая результаты, представленные в табл. 4.6 (TPQM) и в табл. 4.1 (TPCM) (с. 153), отмечаем, что сети на основе кватернионов и сети на основе комплексных чисел характеризуются примерно одинаковой степенью соответствия полученных распределений числа

успешно синхронизировавшихся сетей от количества затраченных шагов нормальному распределению.

В работе [145] приведены результаты сравнения сетей TPM, TPCM (CVTPM) и TPQM (QVTPM) при изменении синаптической глубины L для сетей с сопоставимыми K и N (параметр $KN = 4000$ – для TPM, 2000 – для TPCM, 1000 – для TPM; это связано с особенностями, которые мы обсудили выше). Рис. 4.11 и 4.12 повторяют эти результаты (знак в виде квадрата соответствует TPM, треугольника – TPCM, круга – TPQM).

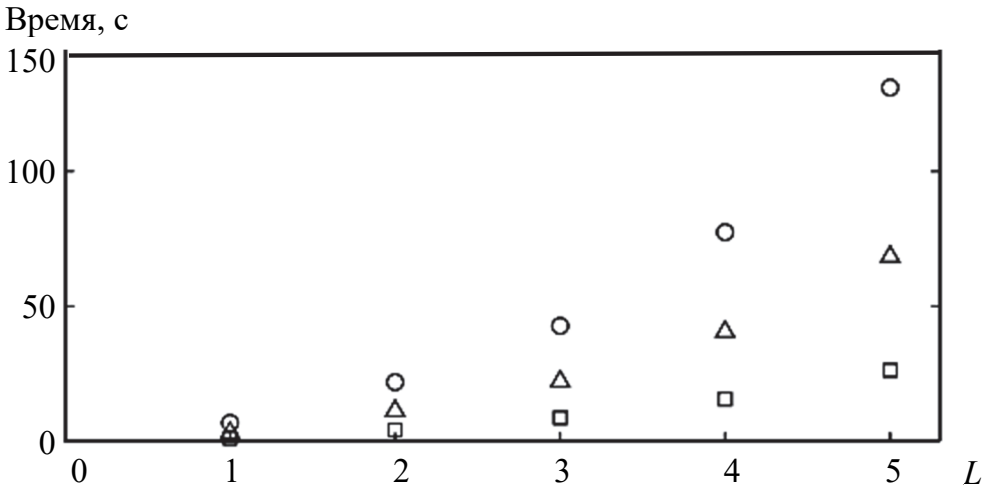

Рис. 4.11. Сравнение сетей TPM, TPCM (CVTPM) и TPQM (QVTPM) по времени (в секундах) достижения состояния синхронизации при изменении L [145]

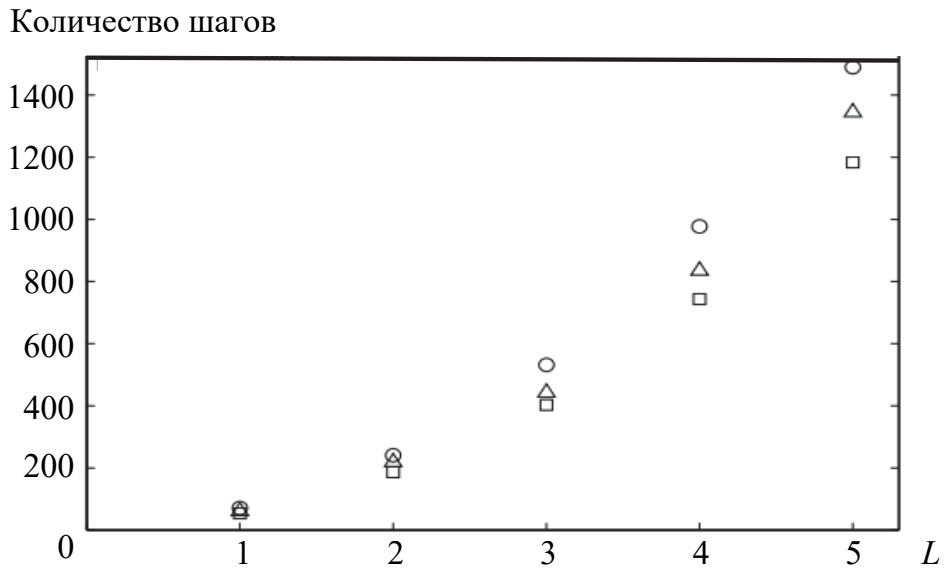

Рис. 4.12. Сравнение сетей TPM, TPCM (CVTPM) и TPQM (QVTPM) по времени (количество шагов) достижения состояния синхронизации при изменении L [145]

Авторы [145] подчеркивают, что время работы и количество итераций QVTPM также растут полиномиально с увеличением L.

Вероятность же P_E успеха простой (рис. 4.13) и геометрической атаки (рис. 4.14) является самой низкой для сетей TPQM (QVTPM) в сравнении с сетями TPM и TPCM (CVTPM) при фиксированных L. Здесь атака считалась успешной, если злоумышленнику (сеть E) удавалось узнать не менее 90% весов.

В дополнение к рассматриваемому вопросу безопасности TPQM отметим, что отношение $t_{\text{learn}}/t_{\text{synch}}$ для данного типа архитектуры в другом исследовании [144] установлено на уровне 0,012.

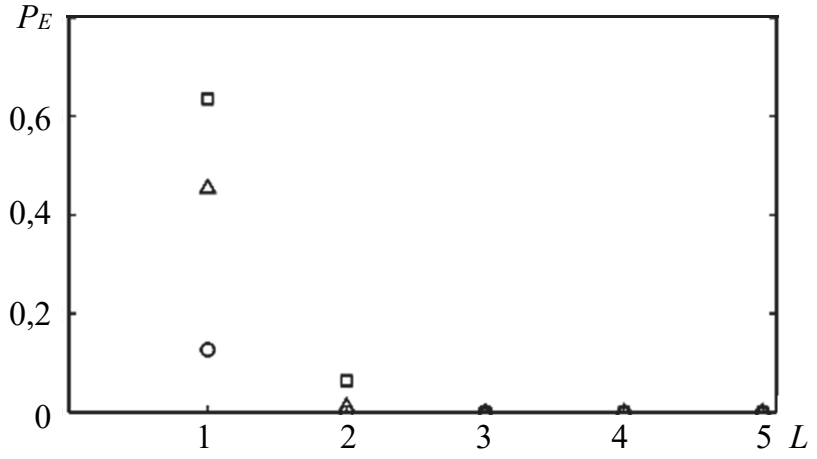

Рис. 4.13. Вероятности успеха простой атаки на сети TPM, TPCM (CVTPM) и TPQM (QVTPM) при изменении L [145]

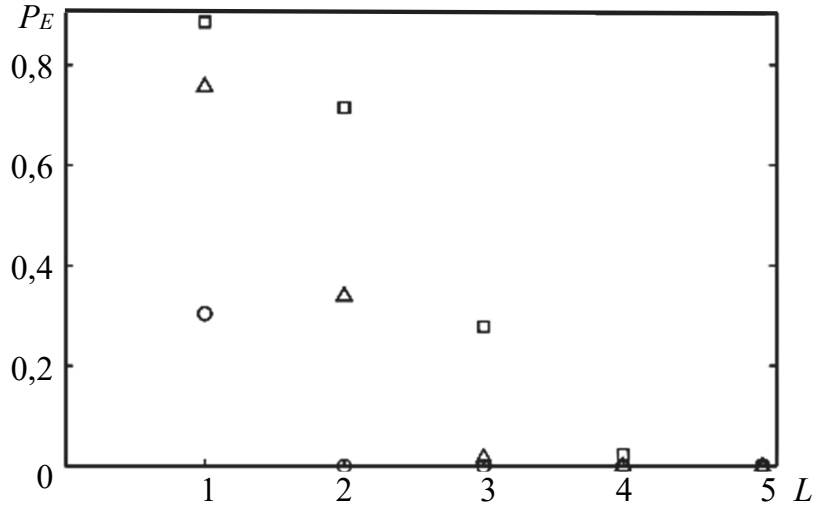

Рис. 4.14. Вероятности успеха геометрической атаки на сети TPM, TPCM (CVTPM) и TPQM (QVTPM) при изменении L [145]

Подводя общий итог, отметим, что сети на основе архитектуры $\langle \text{TPQM}(K, N, L)^{A/B} \rangle$ характеризуются более продолжительным

периодом процесса синхронизации в сравнении с сетями $\langle \text{TPM}(K, N, L)^{A/B} \rangle$ и $\langle \text{TPCM}(K, N, L)^{A/B} \rangle$, однако более эффективны в противодействии атакам злоумышленника.

4.3. TPOM – TPM на основе алгебры октонионов

4.3.1. Общая характеристика октонионов

Рассмотренные выше кватернионы являются заключительным расширением тела действительных чисел, выполняющим условие ассоциативности операции умножения. Напомним, что условие ассоциативности умножения определяется в общем случае следующим образом: $u(vw) = (uv)w$ для любых элементов u, v, w. Однако если ввести ослабленное условие ассоциативности, а именно: $(uv)v = u(vv)$, $(uu)v = u(uv)$ для любых элементов u, v, то можно вывести обобщение теоремы Фробениуса.

Обобщение теоремы Фробениуса [146]. Если $\mathbb{R}^n$ выполняет все аксиомы тела, кроме коммутативности умножения, где ассоциативность заменена альтернативностью, то для $n = 1, 2, 4, 8$ мы получаем четыре множества $\mathbb{R}$, $\mathbb{C}$, $\mathbb{H}$, $\mathbb{O}$: $\mathbb{R}$ – действительные числа, $\mathbb{C}$ – комплексные числа и $\mathbb{H}$ – кватернионы. Последнее множество $\mathbb{O}$ – это октонионы (Octonions).

Алгебра октонионов была разработана в 1843 году параллельно двумя математиками: Дж. Грейвзом (J. Graves) и А. Кэли (A. Cayley). Октонионы (числа Кэли) – третье (после комплексных чисел и кватернионов) множество, которое возникло благодаря применению конструкции Кэли – Диксона (Cayley – Dickson Construction) к действительным числам.

Хотя октонионы не так хорошо известны, как кватернионы и комплексные числа, они находят применение во многих приложениях [147, 148], в том числе – на основе ИНС (см., например, [149–150]).

Особенности и результаты использования октонионов как математической основы TPM обсуждались в материалах [106, 139][13].

[13] Достаточно подробный и всесторонний анализ использования октонионов, кватернионов и комплексных чисел в архитектурах на основе TPM представлен в диссертационном исследовании М. Плонковского (M. Płonkowski) на соискание ученой степени кандидата технических наук «Модели передачи и криптографического преобразования информации на основе нейросетевых технологий и расширения поля используемых чисел» (Бел. гос. технол. ун-т, 2009).

Таким образом, если мы допустим ослабление условия ассоциативности умножения, то можем использовать октонионы наравне с остальными кольцами (действительных чисел, комплексных чисел и кватернионов) для построения ТРМ. Это означает, что все действия сохраняют свой смысл и математическую корректность.

Октонион o по аналогии с комплексным числом и кватернионом запишем в следующей форме:

$$o = a_1 + a_2\boldsymbol{i} + a_3\boldsymbol{j} + a_4\boldsymbol{k} + a_5\boldsymbol{l} + a_6\boldsymbol{m} + a_7\boldsymbol{n} + a_8\boldsymbol{p}, \qquad (4.19)$$

где a_1–$a_8 \in \mathbb{R}$. Каждый октонион вида (4.19) представляет собой формально сумму действительного числа a_1 с вектором (мнимая часть):

$$a_2\boldsymbol{i} + a_3\boldsymbol{j} + a_4\boldsymbol{k} + a_5\boldsymbol{l} + a_6\boldsymbol{m} + a_7\boldsymbol{n} + a_8\boldsymbol{p}.$$

С алгебраической точки зрения числа Кэли – это множество 8-размерного линейного пространства над телом действительных чисел. Отсюда следует, что они могут быть представлены как 8-элементные векторы с действительными коэффициентами. В соответствии с формулой (4.19) эти числа представляют собой линейную комбинацию из 8 мнимых единиц, создающих стандартную базу пространства. Эти единицы обозначим как 1, $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$, $\boldsymbol{l}$, $\boldsymbol{m}$, $\boldsymbol{n}$, $\boldsymbol{p}$.

Действие сложения – это обыкновенное сложение векторов 8-размерного пространства. Действие умножение более сложное. Оно схематически представлено на рис. 4.15.

	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$	$\boldsymbol{l}$	$\boldsymbol{m}$	$\boldsymbol{n}$	$\boldsymbol{p}$
1	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$	$\boldsymbol{l}$	$\boldsymbol{m}$	$\boldsymbol{n}$	$\boldsymbol{p}$
$\boldsymbol{i}$	$\boldsymbol{i}$	–1	$\boldsymbol{l}$	$\boldsymbol{p}$	$-\boldsymbol{j}$	$\boldsymbol{n}$	$-\boldsymbol{m}$	$-\boldsymbol{k}$
$\boldsymbol{j}$	$\boldsymbol{j}$	$-\boldsymbol{l}$	–1	$\boldsymbol{m}$	$\boldsymbol{i}$	$-\boldsymbol{k}$	$\boldsymbol{p}$	$-\boldsymbol{n}$
$\boldsymbol{j}$	$\boldsymbol{j}$	$-\boldsymbol{l}$	–1	$\boldsymbol{m}$	$\boldsymbol{i}$	$-\boldsymbol{k}$	$\boldsymbol{p}$	$-\boldsymbol{n}$
$\boldsymbol{k}$	$\boldsymbol{k}$	$-\boldsymbol{p}$	$-\boldsymbol{m}$	–1	$\boldsymbol{n}$	$\boldsymbol{j}$	$-\boldsymbol{l}$	$\boldsymbol{i}$
$\boldsymbol{l}$	$\boldsymbol{i}$	$\boldsymbol{j}$	$-\boldsymbol{i}$	$-\boldsymbol{n}$	–1	$\boldsymbol{p}$	$\boldsymbol{k}$	$-\boldsymbol{m}$
$\boldsymbol{m}$	$\boldsymbol{m}$	$-\boldsymbol{n}$	$\boldsymbol{k}$	$-\boldsymbol{j}$	$-\boldsymbol{p}$	–1	$\boldsymbol{i}$	$\boldsymbol{l}$
$\boldsymbol{n}$	$\boldsymbol{n}$	$\boldsymbol{m}$	$-\boldsymbol{p}$	$\boldsymbol{l}$	$-\boldsymbol{k}$	$-\boldsymbol{i}$	–1	$\boldsymbol{j}$
$\boldsymbol{p}$	$\boldsymbol{p}$	$\boldsymbol{k}$	$\boldsymbol{n}$	$-\boldsymbol{i}$	$\boldsymbol{m}$	$-\boldsymbol{l}$	$-\boldsymbol{j}$	–1

Рис. 4.15. Схема умножения базисных векторов октонионов

Для удобства обозначим мнимые числа от $\boldsymbol{i}$ до $\boldsymbol{p}$ соответственно от e_1 до e_7; элементы e_1, …, e_7 – это квадратные корни числа –1. Умножение последовательно происходит от строки (e_x) к столбцу (e_y). Отсюда вытекают следующие математические зависимости:

– если $x \neq y$, то $e_x e_y = -e_y e_x$;

– если $e_x e_y = e_k$, то $e_{x+1} e_{y+1} = e_{k+1}$ и $e_{2x} e_{2y} = e_{2k}$.

4.3.2. Особенности архитектуры и модель сети ТРОМ

Идея, структура и принципы взаимодействия ТРМ на основе алгебры октонионов в сравнении с использованием ранее рассмотренных алгебр отличаются лишь количеством чисел, формирующих выходные сигналы нейронов скрытого слоя и выходной сигнал всей сети [151].

Определение 4.3. Две НС (A и B) на основе архитектуры ТРОМ, характеризующиеся одинаковыми параметрами (K, N, L) и предназначенные для согласования криптографических ключей на основе алгебры октонионов, будем обозначать соответственно $\langle \textbf{ТРОМ } (\boldsymbol{K}, \boldsymbol{N}, \boldsymbol{L})^{\boldsymbol{A}} \rangle$ и $\langle \textbf{ТРОМ } (\boldsymbol{K}, \boldsymbol{N}, \boldsymbol{L})^{\boldsymbol{B}} \rangle$.

Архитектура сети ТРОМ состоит из двух слоев. Первый уровень составляют персептроны, содержащие N-элементные векторы весов ($[w_{i1}, w_{i2}, \ldots, w_{iN}]$, где $1 \leq i \leq K$), элементами которых являются октонионы. Каждый вес w_{ij} является октонионом с коэффициентами $(a_1)_{ij}$, $(a_2)_{ij}$, $(a_3)_{ij}$, $(a_4)_{ij}$, $(a_5)_{ij}$, $(a_6)_{ij}$, $(a_7)_{ij}$ и $(a_8)_{ij}$:

$$w_{ij} = (a_1)_{ij} + (a_2)_{ij}\boldsymbol{i} + (a_3)_{ij}\boldsymbol{j} + (a_4)_{ij}\boldsymbol{k} + (a_5)_{ij}\boldsymbol{l} + \\ + (a_6)_{ij}\boldsymbol{m} + (a_7)_{ij}\boldsymbol{n} + (a_8)_{ij}\boldsymbol{p}, \qquad (4.20)$$

при этом $\{(a_1)_{ij}, (a_2)_{ij}, (a_3)_{ij}, (a_4)_{ij}\,(a_5)_{ij}, (a_6)_{ij}, (a_7)_{ij}, (a_8)_{ij}\} \in [-L, L]$.

Ограничение, налагаемое на величины вектора весов, будет расширено до следующего множества из пространства $\mathbb{R}^8$: $[-L, L]^8$. Такое представление аналогично предыдущим конструкциям ввиду того, что между октонионами и точками пространства $\mathbb{R}^8$ существует взаимно однозначное отображение.

Величинами весов каждого из нейронов являются октонионы, выбираемые из множества 2^8 разных величин: {(1, 1, 1, 1, 1, 1, 1, 1), (–1, 1, 1, 1, 1, 1, 1, 1), …, (1,– 1, –1, –1, –1, – 1, –1, –1) (–1, – 1, –1, –1, –1, –1, –1, –1)}. Однако множество всех возможных 256 октонионов может быть представлено только схематически. А представление в виде программного кода аналогично случаю с кватернионами (или же с комплексными числами). Пример кода, реализующего ТРОМ, представлен в приложении В.

Проанализируем далее особенности математического описания сети ТРОМ. Выходные величины персептронов σ_i принадлежат закрытому множеству ввиду особенности операции умножения: {(1, 0, 0, 0, 0, 0, 0, 0), (–1, 0, 0, 0, 0, 0, 0, 0), …, (0, 0, 0, 0, 0, 0, 0, 1), (0, 0, 0, 0, 0, 0, 0, –1)}.

Как отмечалось, октонионы создают альтернативное кольцо (с ослабленным, альтернативным условием ассоциативности). При этом математические операции аналогичны операциям с действительными числами. В случае октонионов функция знака ($\sigma_i(o)$) принимает более сложный вид в сравнении с предыдущими архитектурами:

$$\sigma_i(o) = \begin{cases} (1,0,0,0,0,0,0,0),\ a_1 = \max(\{a_1 : 1 \le i \le 8\}) \wedge a_1 \ge 0); \\ (-1,0,0,0,0,0,0,0),\ a_1 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_1 < 0); \\ (0,1,0,0,0,0,0,0),\ a_2 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_2 \ge 0); \\ (0,-1,0,0,0,0,0,0),\ a_2 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_2 < 0); \\ (0,0,1,0,0,0,0,0),\ a_3 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_3 \ge 0); \\ (0,0,-1,0,0,0,0,0),\ a_3 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_3 < 0); \\ (0,0,0,1,0,0,0,0),\ a_4 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_4 \ge 0); \\ (0,0,0,-1,0,0,0,0),\ a_4 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_4 < 0); \\ (0,0,0,0,1,0,0,0),\ a_5 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_5 \ge 0); \\ (0,0,0,0,-1,0,0,0),\ a_5 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_5 < 0); \\ (0,0,0,0,0,1,0,0),\ a_6 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_6 \ge 0); \\ (0,0,0,0,0,-1,0,0),\ a_6 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_6 < 0); \\ (0,0,0,0,0,0,1,0),\ a_7 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_7 \ge 0); \\ (0,0,0,0,0,0,-1,0),\ a_7 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_7 < 0); \\ (0,0,0,0,0,0,0,1),\ a_8 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_8 \ge 0); \\ (0,0,0,0,0,0,0,-1),\ a_8 = \max(\{a_i : 1 \le i \le 8\}) \wedge a_8 < 0). \end{cases} \quad (4.21)$$

Именно эта функция будет делить пространство на 16 непересекающихся подпространств (вспомним, что в случае кватернионов деление было 8-элементным).

Итак, функция $\sigma_i(o)$ определена в 8-размерном пространстве, где встречаются только 16 (8×2) конечных величин. Они исполняют роль аттракторов, притягивая к ним самые близкие точки.

Следующим этапом в конструировании системы обмена ключами, основанной на архитектуре ТРОМ, является определение метода обучения. Предположим, будет использоваться правило Хэбба.

Обучение будет происходить согласно формулам, являющихся расширением выражений (4.17):

$$\left.\begin{aligned}
&\boldsymbol{R}(w_{ij}^{(t+1)}) = g[\boldsymbol{R}(w_{ij}^{(t)}) + (\boldsymbol{R}(x_{ij}\tau^{(t)}))\Theta(\boldsymbol{R}(\tau^{(t)})\boldsymbol{R}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{R}(\tau^{A(t)})\boldsymbol{R}(\tau^{B(t)}))],\\
&\boldsymbol{I}(w_{ij}^{(t+1)}) = g[\boldsymbol{I}(w_{ij}^{(t)}) + (\boldsymbol{I}(x_{ij}\tau^{(t)}))\Theta(\boldsymbol{I}(\tau^{(t)})\boldsymbol{I}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{I}(\tau^{A(t)})\boldsymbol{I}(\tau^{B(t)}))],\\
&\boldsymbol{J}(w_{ij}^{(t+1)}) = g[\boldsymbol{J}(w_{ij}^{(t)}) + (\boldsymbol{J}(x_{ij}\tau^{(t)}))\Theta(\boldsymbol{J}(\tau^{(t)})\boldsymbol{J}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{J}(\tau^{A(t)})\boldsymbol{J}(\tau^{B(t)}))],\\
&\boldsymbol{K}(w_{ij}^{(t+1)}) = g[\boldsymbol{K}(w_{ij}^{(t)}) + (\boldsymbol{K}(x_{ij}\tau^{(t)}))\Theta(\boldsymbol{K}(\tau^{(t)})\boldsymbol{K}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{K}(\tau^{A(t)})\boldsymbol{K}(\tau^{B(t)}))],\\
&\boldsymbol{L}(w_{ij}^{(t+1)}) = g[\boldsymbol{L}(w_{ij}^{(t)}) + (\boldsymbol{L}(x_{ij}\tau^{(t)}))\Theta(\boldsymbol{L}(\tau^{(t)})\boldsymbol{L}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{L}(\tau^{A(t)})\boldsymbol{L}(\tau^{B(t)}))],\\
&\boldsymbol{M}(w_{ij}^{(t+1)}) = g[\boldsymbol{M}(w_{ij}^{(t)}) + (\boldsymbol{M}(x_{ij}\tau^{(t)}))\Theta(\boldsymbol{M}(\tau^{(t)})\boldsymbol{M}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{M}(\tau^{A(t)})\boldsymbol{M}(\tau^{B(t)}))],\\
&\boldsymbol{N}(w_{ij}^{(t+1)}) = g[\boldsymbol{N}(w_{ij}^{(t)}) + (\boldsymbol{N}(x_{ij}\tau^{(t)}))\Theta(\boldsymbol{N}(\tau^{(t)})\boldsymbol{N}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{N}(\tau^{A(t)})\boldsymbol{N}(\tau^{B(t)}))],\\
&\boldsymbol{P}(w_{ij}^{(t+1)}) = g[\boldsymbol{P}(w_{ij}^{(t)}) + (\boldsymbol{P}(x_{ij}\tau^{(t)}))\Theta(\boldsymbol{P}(\tau^{(t)})\boldsymbol{P}(\sigma_i^{A/B(t)}))\times \\
&\times\Theta(\boldsymbol{P}(\tau^{A(t)})\boldsymbol{P}(\tau^{B(t)}))].
\end{aligned}\right\} \quad (4.22)$$

Ограничение, целью которого является контроль над увеличением значений величины весов, будет использовать следующий гиперкуб: $[-L, L]\times[-L, L]\times[-L, L]\times[-L, L]\times[-L, L]\times[-L, L]\times[-L, L]\times[-L, L]$. Таким образом, формулы для вычисления ограничений состоят из выражений (4.4), (4.5), (4.14)[14] со следующим дополнением (следует обратить внимание на различный физический смысл одинаковых по начертанию символов – эта разница отмечалась выше):

[14] В силу использования одинаковых обозначений соответствующих мнимых частей в комплексных числах, кватернионах и октонионах.

$$
\left.
\begin{aligned}
\boldsymbol{L}(w_{ij}^{A/B}) &= \begin{cases} \operatorname{sign}(\boldsymbol{L}(w_{ij}^{A/B}))L, \left|\boldsymbol{L}(w_{ij}^{A/B})\right| > L, \\ \boldsymbol{L}(w_{ij}^{A/B}), \text{ в противном случае;} \end{cases} \\
\boldsymbol{M}(w_{ij}^{A/B}) &= \begin{cases} \operatorname{sign}(\boldsymbol{M}(w_{ij}^{A/B}))L, \left|\boldsymbol{M}(w_{ij}^{A/B})\right| > L, \\ \boldsymbol{M}(w_{ij}^{A/B}), \text{ в противном случае;} \end{cases} \\
\boldsymbol{N}(w_{ij}^{A/B}) &= \begin{cases} \operatorname{sign}(\boldsymbol{N}(w_{ij}^{A/B}))L, \left|\boldsymbol{N}(w_{ij}^{A/B})\right| > L, \\ \boldsymbol{N}(w_{ij}^{A/B}), \text{ в противном случае;} \end{cases} \\
\boldsymbol{P}(w_{ij}^{A/B}) &= \begin{cases} \operatorname{sign}(\boldsymbol{P}(w_{ij}^{A/B}))L, \left|\boldsymbol{P}(w_{ij}^{A/B})\right| > L, \\ \boldsymbol{P}(w_{ij}^{A/B}), \text{ в противном случае.} \end{cases}
\end{aligned}
\right\} \qquad (4.23)
$$

Вероятно, это самое простое ограничительное множество. Все же, как было сказано выше, возможны и другие (часто более эффективные) ограничения.

Учитывая рассмотренную конструкцию (вместе с введенными модификациями), мы можем использовать ее в протоколе обмена ключами. Все остальное аналогично случаю, представленному при анализе предыдущих архитектур (TPM, TPCM, TPQM).

4.3.3. Анализ процесса синхронизации TPOM

Методика исследований временны́х и вероятностных характеристик процесса синхронизации сетей $\langle \text{TPOM}(K, N, L)^A \rangle$ и $\langle \text{TPOM}(K, N, L)^B \rangle$, а также безопасности этого процесса практически ничем не отличается от описанных выше в отношении иных архитектур TPM.

На рис. 4.16 представлена гистограмма распределения числа синхронизировавшихся сетей $\langle \text{TPOM}(K = 6, N = 6, L = 6)^{A/B} \rangle$ по количеству шагов (горизонтальная ось) до наступления состояния синхронизации при общем числе симуляций – 1168. Результаты получены с помощью программного средства *RCQOv.2* (см. п. 4.1.2.2).

В табл. 4.7 приведена более подробная информация о результатах синхронизации сетей TPOM с разными параметрами и с оценкой соответствия полученных распределений нормальному распределению по индексу Брея – Кертиса. Представляет интерес сравнение

данных из табл. 4.7 с данными из табл. 4.1 (TPCM) (с. 153) и табл. 4.6 (TPQM) (с. 165), поскольку все эти результаты получены по абсолютно одинаковой методике и на одной и той же аппаратно-программной платформе.

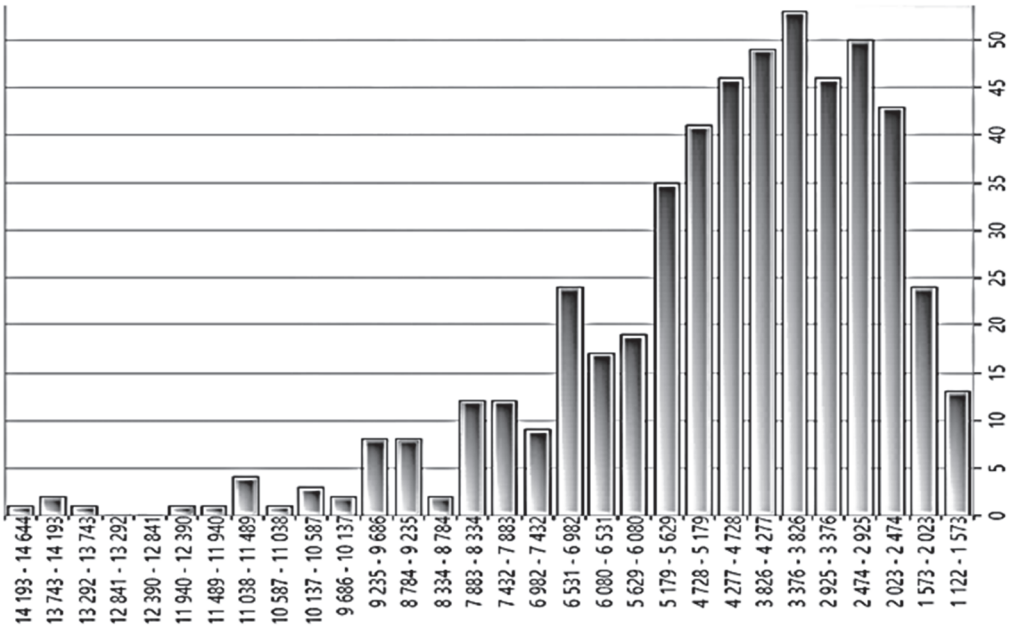

Рис. 4.16. Гистограмма распределения числа синхронизировавшихся сетей ⟨TPOM $(K, N, L)^{A/B}$⟩ по количеству шагов до наступления состояния синхронизации

Таблица 4.7

Статистические результаты симуляций процесса синхронизации TPOM

N	*K*	*L*	Количество симуляций	Количество успешных синхронизаций	t_{synch}, шагов (среднее)	Среднеквадратичное отклонение	Индекс Брея – Кертиса
5	5	5	1233	1231	1258,2	607,7	0,25
5	5	7	1000	1000	12 698,3	6471,3	0,25
5	7	7	645	644	12 923,6	5753,7	0,24
6	5	5	1000	1000	674,6	291,4	0,23
6	6	6	1168	1167	1604,6	711,7	0,23
7	5	5	1000	1000	568,2	254,9	0,23
7	7	7	653	651	5825,1	2779,1	0,23

Из анализа последней таблицы следует, что статистика для TMOM с различными параметрами (*N*-*K*-*L*) практически в одинаковой степени соответствует нормальному распределению.

Дополнительно в табл. 4.8 для сравнения приведены отношения $t_{\text{learn}}/t_{\text{synch}}$ для TPM с одинаковыми параметрами, но при использовании различных алгебр.

Таблица 4.8

Отношения $t_{\text{learn}}/t_{\text{synch}}$ для TPM при использовании различных алгебр

Архитектура	$t_{\text{learn}}/t_{\text{synch}}$
TPM	0,33553
TPCM	0,10752
TPQM	0,01193
TPOM	0,00022

Как видно, из всех рассмотренных архитектур TPM сети $\langle \text{TPOM}\ (K, N, L)^{A/B} \rangle$ обеспечивают самый высокий уровень безопасности при их использовании в качестве средства генерации общего для сторон A и B криптографического ключа.

5. ИСПОЛЬЗОВАНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ОПЕРАЦИЙ НАД ХЕШ-ФУНКЦИЯМИ

5.1. Детерминированность и хаотичность процессов в нейронных сетях

Рассмотренные в предыдущих главах преобразовательные свойства (перемежение и рассеивание или сжатие) НС, а также односторонность этих преобразований, заключающаяся в сложности определения входного воздействия по значению (значениям) выходного, позволяют использовать указанные свойства НС для хеш-преобразований.

Нейронные сети, по сути являющиеся детерминированными системами, чувствительны к минимальным изменениям входного воздействия, что в совокупности с вышеуказанными свойствами перемешивания, рассеивания и сжатия данных означает реализацию в НС хаотических процессов.

В работах [152, 153] отмечается, что ИНС подходят для использования в криптографических хеш-функциях благодаря своим важным свойствам:

1) нелинейная структура: сложные взаимосвязи между входными и выходными данными и, следовательно, обеспечение свойства перемешивания за счет использования нелинейной функции в качестве передаточной функции;

2) свойство диффузии: процесс перемешивания применяется к каждому нейрону, в то время как выходной сигнал соотносится со всеми элементами входного сигнала (см. рис. 1.7 на с. 17);

3) реализация операции параллельных вычислений;

4) гибкость: размер ввода/вывода (n элементов) может быть изменен, что позволяет обеспечить гибкий размер блока данных на входе и выходе;

5) реализация функция одностороннего сжатия.

Структурная и функциональная специфика НС, используемых для операций над хеш-функциями и иными криптографическими или стеганографическими преобразованиями, определяет такие нейронные сети как *хаотические* (ХНС; Chaotic Neural Networks, CNN).

Можно сказать, что архитектура хаотической сети похожа на архитектуру стандартной нейронной сети, но основное различие между ними заключается в том, что в слоях ХНС мы находим хаотические отображения.

Вероятно, впервые определение и математическое описании ХНС дано К. Айхара (K. Aihara) в работе [154]. В более поздних публикациях в данной предметной области авторы придерживаются, в основном, использования таких же обозначений соответствующих элементов модели ХНС, которые ввел К. Айхара.

В основе модели ХНС используется соотношение, которое К. Айхара называет моделью Нагумо – Сато (Nagumo – Sato):

$$\sigma(t+1) = u(A(t) - \alpha \sum_{d=0}^{t} k^d x(t-d) - \theta), \tag{5.1}$$

где $\sigma(t+1)$ – выход нейрона в момент времени $t+1$, который может быть активным (1) либо неактивным (0); $u(y)$ – функция шага (см. аналог: (1.27)):

$$u(y) = \begin{cases} 1, \text{ при } y \geq 0, \\ 0, \text{ при } y < 0; \end{cases}$$

$A(t)$ – мощность входного сигнала (аналог длины входного вектора X сети, в наших вышеприведенных рассуждениях – KN) в момент времени t; α – некоторое положительное число; $k \in [0, 1]$; θ – порог срабатывания нейрона.

Введя новую переменную:

$$y(t+1) = A(t) - \alpha \sum_{d=0}^{t} k^d x(t-d) - \theta),$$

соответствующую внутреннему состоянию нейрона, формулу (5.1) можно представить в виде двух связанных выражений:

$$y(t+1) = ky(t) - \alpha u(y(t)) + \alpha(t), \tag{5.2}$$

$$\sigma(t+1) = u(t+1)), \tag{5.3}$$

где

$$\alpha(t) = A(t) - kA(t-1) - \theta(1-k). \tag{5.4}$$

В общем случае, моделируя ХНС наиболее общего вида, необходимо принять во внимание наличие двух типов входных сигналов

нейрона: входы обратной связи от нейронов, таких как *сети Хопфилда* (см., например, [17]), и внешние входные данные, такие как *сети обратного распространения ошибки* (см., например, [23]).

Заменив ступенчатую функцию $u(\cdot)$ на непрерывную $f(\cdot)$, динамику i-го хаотического нейрона в НС, состоящей из K хаотических нейронов, можно смоделировать уравнением

$$\sigma_i(t+1) = f_i\Big(\sum_{j=1}^{N} x_{ij} \sum_{d=0}^{t} k_e^d A_j(t-d) +$$

$$+\sum_{j=1}^{K} w_i j \sum_{d=0}^{t} k_f^d h_j(x_j(t-d)) - \alpha \sum_{d=0}^{t} k_r^d g_i(x_i(t-d)) - \theta_i\Big). \qquad (5.5)$$

В уравнение (5.5) входят, помимо уже перечисленных, следующие параметры: x_{ij} имеет тот же физический смысл, что и в ТРМ; w_{ij} – весовой коэффициент, связывающий i-й и j-й нейроны сети (как видим, этот параметр имеет практически тот же смысл, что и в ТРМ); N – количество внешних входов сети; $A_j(t - d)$ – сила сигнала на j-м внешнем входе в момент времени $t - d$ (можно принять равной произведению $x_j w_j$); h_j – передаточная функция аксона в j-м хаотическом нейроне (примерно соответствует сигналу, определяемому формулой (1.3)); g_i – *рефракторная функция* (по определению К. Айхара – Refractory Function) i-го хаотического нейрона, означает взаимосвязь выхода нейрона с последующим входным воздействием, более точно – устойчивость или восприимчивость ко входным воздействиям, для упрощения предлагается считать, что $g_i(x) = x$ [154]; k_e, k_f, k_r – параметры, определяющие степень затухания сигналов на внешних входах, входах обратной связи и рефрактерности соответственно.

При $k_e = k_f = k_r = k$ получим:

$$\sigma_i(t+1) = f_i(y_i(t+1)), \qquad (5.6)$$

$$y_i(t+1) = k y_i(t) + \sum_{j=1}^{N} x_{ij} A_j(t) + \sum_{j=1}^{N} w_i j h_j(f_j(y_j(t)))) -$$

$$- \alpha g_i(f_i(y_i(t))) - \theta_i(1-k). \qquad (5.7)$$

ХНС можно рассматривать как рекуррентную сеть, в которой элементы связаны «каждый с каждым», без образования связи «сам на себя».

Пример рекуррентной сети представлен на рис. 5.1.

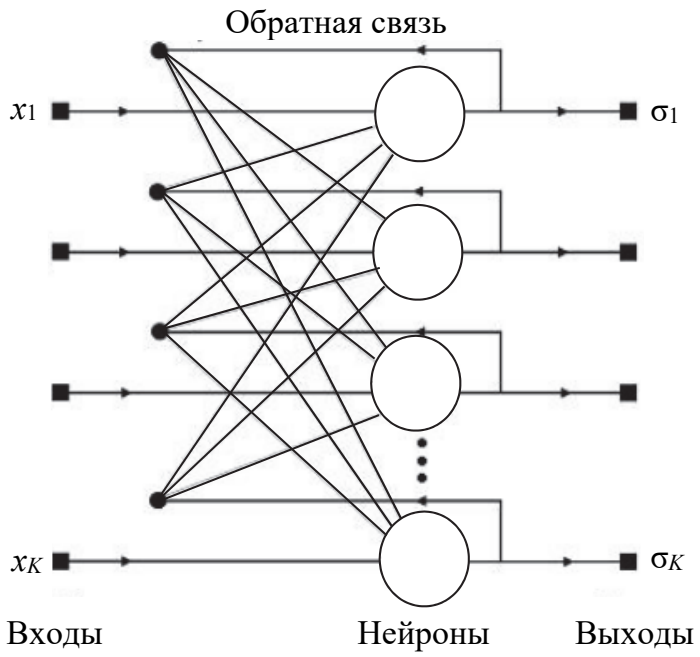

Рис. 5.1. Пример рекуррентной сети

К числу рекуррентных относится *рекуррентный многослойный перцептрон*, содержащий более одного скрытого слоя, с обратными связями между каждым из расчета слоев и входным слоем. Сигнал с выходных нейронов или нейронов скрытого слоя частично передается обратно на входы нейронов входного слоя. Частным случаем рекуррентных сетей являются двунаправленные сети. В таких сетях между слоями существуют связи как в направлении от входного слоя к выходному, так и в обратном.

5.2. Архитектура и особенности использования хаотических нейронных сетей для хеширования сообщений

В статье [153] предложена и позднее проанализирована в работах [156, 157] архитектура НС для вычисления хеша. Эта сеть состоит из трех слоев: входного, скрытого и выходного (рис. 5.2). Данные слои соответственно реализуют перемежение, рассеивание и компрессию пространств входных битов. Входные и выходные векторы каждого из слоев равны соответственно:

$$P = [P_0P_1\ldots P_{31}],\ C = [C_0C_1\ldots C_7],\ D = [D_0D_1\ldots D_7],\ H = [H_0H_1\ldots H_3].$$

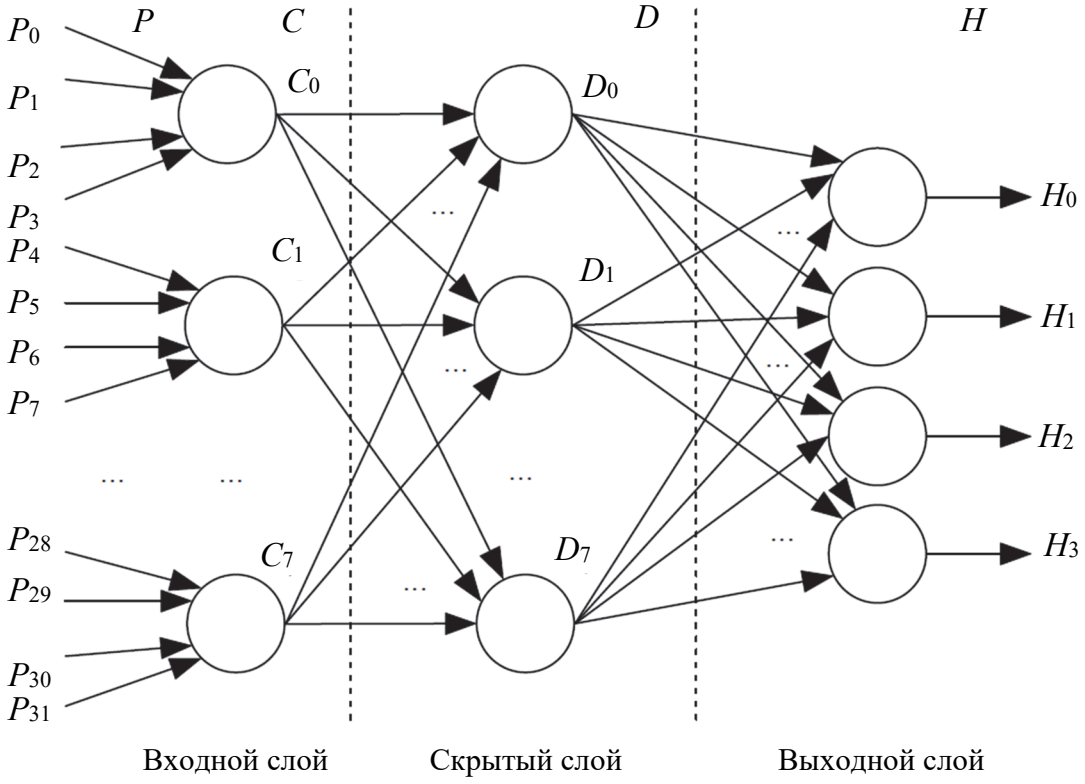

Рис. 5.2. Архитектура трехслойной ХНС
для вычисления хеша [153]

Выход H сети (выходной слой) определяется следующим способом:

$$H = f_2(W_2 D + B_2) = f_2(W_2 f_1(W_1 C + B_1) + B_2) =$$
$$= f_2(W_2 f_1(W_1 f_0(W_0 P_0 + B_0) + B_1) + B_2), \qquad (5.8)$$

где f_i – функция перехода (кусочно-линейное хаотическое отображение); W_i – элемент вектора весов; B_i – биас i-го нейрона соответствующего слоя.

Функция перехода f_i определяется следующим образом:

$$X(t+1) = f(X(t), Q) = \begin{cases} X(t)/Q, & 0 \le X(t) < Q, \\ X(t) - Q)/(0{,}5 - Q), & Q \le X(t) < 0{,}5, \\ (1 - Q - X(t))/(0{,}5 - Q), & 0{,}5 \le X(t) < 1 - Q, \\ (1 - X(t))/Q, & 1 - Q \le X(t) \le 1, \end{cases} \qquad (5.9)$$

где Q – это параметр управления, удовлетворяющий условию: $0 < Q < 0{,}5$; при этом условии $f(\cdot)$ соответствует свойствам хаотичности.

Если вычисление на основе хаотической функции повторяется T раз (T достаточно велико; обычно $T \geq 50$), то даже небольшая разница в начальном значении $X(t)$ или параметре Q приводит к большим различиям в значении $X(t + T)$.

Кусочно-линейное хаотическое отображение (Piecewise Linear Chaotic Map, PWLCM) $f(\cdot)$ используется не только в качестве передаточной функции, но и как генератор ключей для инициализации параметров нейронной сети. На основе хаотического отображения функционал входного слоя определяется в следующем виде:

$$C = f^T\left(\begin{bmatrix} \sum_{i=0}^{3} w_{0,i}P_i + b_{0,0} \\ \sum_{i=4}^{7} w_{0,i}P_i + b_{0,1} \\ \vdots \\ \sum_{i=28}^{31} w_{0,i}P_i + b_{0,7} \end{bmatrix}, Q_0\right) = \begin{bmatrix} f^T(\sum_{i=0}^{3} w_{0,i}P_i + B_{0,0}, Q_0) \\ f^T(\sum_{i=4}^{7} w_{0,i}P_i + B_{0,1}, Q_0) \\ \vdots \\ f^T(\sum_{i=28}^{31} w_{0,i}P_i + B_{0,7}, Q_0) \end{bmatrix} = \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_7 \end{bmatrix}, \quad (5.10)$$

где $W_0 = [w_{0,0}, w_{0,1}, \ldots, w_{0,31}]$.

Также принимаем, что входные величины функции f принадлежат промежутку [0, 1], и любая арифметическая операции mod 1 определяется следующим способом:

$$a \bmod 1 = \begin{cases} a, \ 0 \leq a < 1, \\ a - 1, \ 1 \leq a < 2. \end{cases}$$

По аналогии определяются также функции, вычисляющие выходные величины двух других слоев сети:

$$D = f_1(W_1C + B_1) = f(W_1C + B_1, Q) =$$

$$= \begin{bmatrix} f(\sum_{i=0}^{7} w_{1,0,i}C_i + B_{1,0}, Q_1) \\ f(\sum_{i=0}^{7} w_{1,1,i}C_i + B_{1,1}, Q_1) \\ \vdots \\ f(\sum_{i=0}^{7} w_{1,7,i}C_i + B_{1,7}, Q_1) \end{bmatrix} = \begin{bmatrix} D_0 \\ D_1 \\ \vdots \\ D_7 \end{bmatrix}, \quad (5.11)$$

$$H = f_2(W_2 D + B_2) = f^T(W_2 D + B_2, Q_2) =$$

$$= \begin{bmatrix} f^T(\sum_{i=0}^{7} w_{2,0,i} D_i + B_{2,0}, Q_2) \\ f^T(\sum_{i=0}^{7} w_{2,1,i} D_i + B_{2,1}, Q_2) \\ \vdots \\ f^T(\sum_{i=0}^{7} w_{2,3,i} D_i + B_{2,3}, Q_2) \end{bmatrix} = \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_3 \end{bmatrix}. \qquad (5.12)$$

Вес W_1 представляется в виде размера 8×8, B_1 – 8×1, W_2 – 4×8, B_2 – 4×1.

Структурная схема, реализующая соотношения (5.8)–(5.12) на основе ХНС, показана на рис. 5.3 [153].

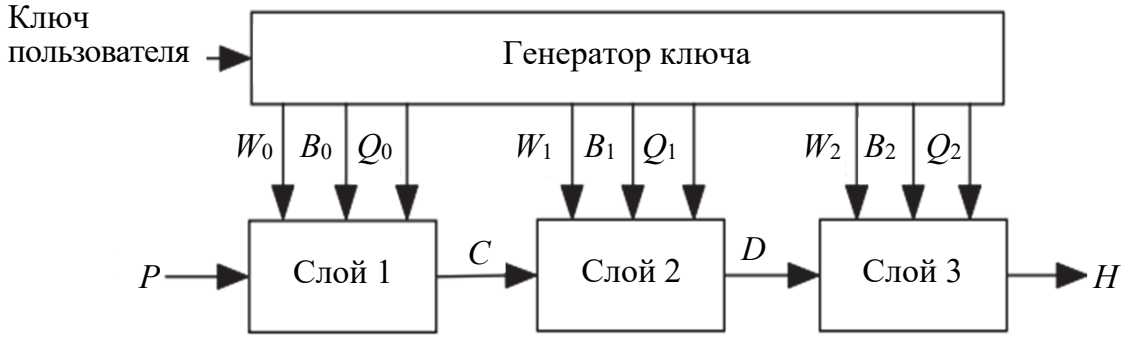

Рис. 5.3. Структурная схема, реализующая хеш-функцию на основе ХНС

Генератор ключей используется для создания дополнительных ключей: W_0, B_0, Q_0, W_1, B_1, Q_1, W_2, B_2 и Q_2. Ключ $K = k_0 k_1 \ldots k_{127}$ разделен на четыре подключа: $K_0 = k_0 k_1 \ldots k_{31}$, $K_1 = k_{32} k_{33} \ldots k_{63}$, $K_2 = k_{64} k_{65} \ldots k_{95}$ и $K_3 = k_{96} k_{97} \ldots k_{127}$.

Сеть сворачивает входной вектор размером 32 пикселя (1024 бита) до выходного вектора размером 4 пикселя (128 битов). Элементы входного вектора (которые представляют собой 32-битные пиксели) делятся на 2^{32}, благодаря чему они нормализуются на интервал [0, 1].

Рассмотренная ХНС требует выполнения относительно большего количества операций, чем традиционные схемы. Например, при длине входного сообщений 1024 бита и $T = 50$ указанное отношение составляет примерно 3,7 для алгоритма MD5 и примерно

2,95 для алгоритма SHA-1. Указанный недостаток связан с тем, что процесс хеширования производится в последовательном режиме: обработка текущего блока сообщения не может начаться, пока не будет обработан предыдущий.

В процессе исследования алгоритма на основе рассмотренной в статье [153] ХНС были выявлены и иные его особенности. Так как НС способна работать только с числами с плавающей запятой (типы данных *float* и *double*) в диапазоне [0, 1], возникает необходимость в предварительном преобразовании входного значения к последовательности вещественных чисел.

Универсальный алгоритм состоит в том, чтобы задать исходный алфавит и уже на его основе произвести преобразование. Однако если сообщение состоит из символов достаточно большого алфавита, или же исходный алфавит неизвестен, то такой подход становится малоэффективным. Поэтому базовый алгоритм хеширования на основе ХНС целесообразно несколько видоизменить:

1) сконвертировать входную строку в массив значений типа *byte* на основе кодировки, которая достоверно включает все символы, сообщения;

2) преобразовать каждое значение в массиве к типу с плавающей запятой путем поэлементного деления значения на $2^8 – 1$.

Аналогичное преобразование необходимо произвести и для выхода нейросети.

В программной реализации рассматриваемой сети [158] применена следующая схема сжатия:

1) весовые коэффициенты сети конфигурируются в соответствии с заданным ключом пользователя (K);

2) преобразованный в последовательность вещественных чисел блок сообщения подается в нейронную сеть и вычисляется ее выход – хэш переданного блока;

3) полученное значение хэша используется в качестве ключа пользователя: с его помощью сеть реконфигурируется и алгоритм возвращается к шагу 2;

4) выход нейросети для последнего блока представляет собой хэш всего сообщения.

Приведенная схема сжатия делает алгоритм достаточно чувствительным к передаваемому сообщению и гарантирует достаточную сложность обратного восстановления сообщения по хэшу. Однако процесс реконфигурации сети (изменения весовых коэффициентов

при помощи функции хаоса в соответствии с ключом пользователя) требует больших временных затрат, что приводит к необходимости поиска альтернативной функции сжатия.

В [159–161] предложены новые архитектуры ХНС, позволяющие повысить эффективность работы сети.

5.3. Использование алгебры комплексных чисел в архитектурах хаотических нейронных сетей

Нейронные сети, основанные на алгебре комплексных чисел (CNNHF – Complex Neural Network Hash Function), благодаря специфике этих чисел обеспечивают, как мы убедились ранее, более высокий уровень безопасности, чем их классические эквиваленты. Второй существенный плюс – это бóльшая свобода при определении разного рода вспомогательных компонентов данной архитектуры. Они также позволяют увеличить безопасность системы [156, 157, 162].

Как было показано выше, вся структура нейронной сети, основанной на комплексных числах, подобна структуре, основанной на действительных числах. В случаях хеширования сообщений изменение будет касаться пороговой функции (функции перехода), которая должна гарантировать односторонность. Функция перехода, определяемая формулой (5.9), трудна для обратимости (необратима), что также гарантирует в существенной степени безопасность хеш-функции.

5.3.1. Модель сети CNNHF

Будем считать, что модель, архитектура и основной функционал ХНС соответствуют их описанию в подгл. 5.2. Главное отличие состоит в используемой функции перехода.

Функция перехода на основе множества Жюлиа (Julia Set, обычно обозначается $J(f)$). Рассматриваемое множество (или функция перехода Жюлиа) – такое подмножество множества комплексных чисел, для каждой точки которого поведение функции под действием итераций является хаотичным, т. е. небольшие изменения в начальных условиях в некоторой небольшой окрестности начальной точки значительно влияют на траекторию этой точки.

С математической точки зрения в определении $J(f)$ с квадратичной функцией

$$f(z) = z^2 + \omega$$

содержится следующее рекурсивное пространство:

$$\left.\begin{aligned} z_0 &= \kappa, \\ z_{t+1} &= z_t^2 + \omega, \end{aligned}\right\} \qquad (5.13)$$

где z_i, κ, $\omega \in \mathbb{C}$ ($\mathbb{C}$ – множество комплексных чисел).

Количество корней второй степени (в теле комплексных чисел) равно двум. Следовательно, количество итераций T должно быть увеличено до 100 (в оригинальной модели количество равно 50); в выражении (5.13) κ – это входной параметр (число пикселей), роль параметра Q будет исполнять переменная ω.

Графическое отображение $J(f)$ представлено на рис. 5.4.

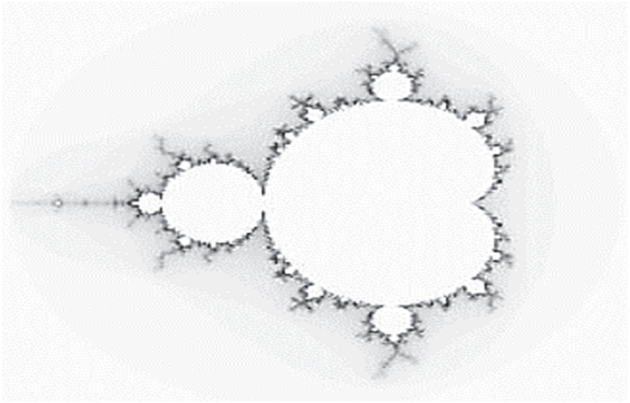

Рис. 5.4. Графическое отображение множества Жюлиа

Множество Жюлиа для $f(z)$ симметрично относительно горизонтальной оси. Учет этого обстоятельства позволяет существенно упростить программную реализацию модели: значительно сокращается объем вычислений.

Функция перехода на основе уравнения (осциллятора) Дуффинга (Duffing Oscillator). Особенностью осциллятора Дуффинга является возможность получения хаотической динамики. Уравнения движения для осциллятора Дуффинга имеют вид:

$$\left.\begin{aligned} x_{t+1} &= y_t, \\ y_{t+1} &= -bx_t + ay_t - y_t^3. \end{aligned}\right\} \qquad (5.14)$$

В них действительные числа изменяются, а пара (x_i, y_i) отождествляется с точкой на плоскости (рис. 5.5). Из соображения взаимно однозначной зависимости между точками плоскости и комплексными числами пара (x_i, y_i) может быть отождествлена с комплексным числом (где действительная часть – x_i, а мнимая часть – y_i). Параметром этой функции будет пара (a, b), также являющаяся комплексным числом.

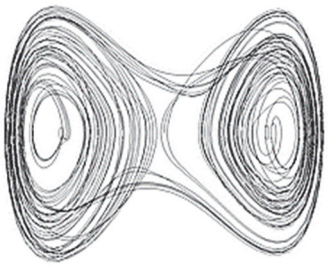

Рис. 5.5. Отображение хаотического поведения точек, задаваемых выражениями (5.14)

Функция перехода на основе уравнения (аттрактора) Энона (Henon Attractor). Отображение Энона, иногда называемое аттрактором/отображением Энона – Помо, представляет собой динамическую систему с дискретным временем. Это один из наиболее изученных примеров динамических систем, демонстрирующих хаотическое поведение. Отображение Энона берет точку (x_t, y_t) на плоскости и отображает ее в новую точку (x_{t+1}, y_{t+1}):

$$\left.\begin{aligned} x_{t+1} &= y_t + 1 - ax_t^2, \\ y_{t+1} &= bx_t. \end{aligned}\right\} \qquad (5.15)$$

Рассматриваемое отображение зависит от двух параметров: a и b, которые для классического отображения Энона (хаотично) имеют значения: a = 1,4 и b = 0,3 (рис. 5.6).

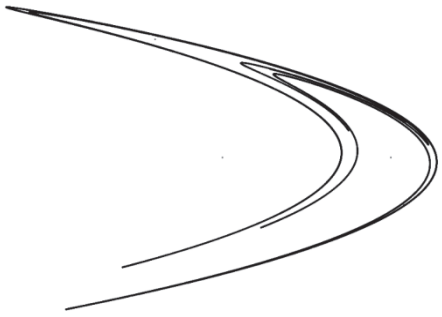

Рис. 5.6. Аттрактор Энона для a = 1,4 и b = 0,3

Физический смысл параметров, входящих в уравнения (5.15), аналогичен смыслу соответствующих параметров из равенств (5.14).

Как уже отмечалось в подгл. 2.3, от хеш-функции требуется соблюдение двух основных свойств. Во-первых, восприимчивость к изменению даже единичного бита хешируемого сообщения. Во-вторых, функция должна быть устойчивой к коллизиям. Проанализируем выполнение данных требований при хешировании сообщений на основе алгебры комплексных чисел.

Возьмем следующее сообщение (записанное в шестнадцатеричной системе): 42F10225DF8B1B79A5B3E79E3A5DC3660CD06 8F097442E391FB4CC788AA1E65D.

Оно имеет длину 64 символа. После сжатия этого сообщения, сгенерированного с помощью нейронной сети CNNHF, оно будет иметь вид: 84751E8B. Длина хеша – 8 символов. Если теперь мы изменим один из символов входного сообщения (в нашем примере первую цифру 4 на цифру 3): 32F10225DF8B1B79A5B3E79E3 A5DC3660CD068F097442E391FB4CC788AA1E65D, то его хеш существенно изменится: 542A5CE0.

На основании рассмотренного выше примера мы убеждаемся, что изменение даже единичного символа полностью преобразовывает выходную последовательность сети.

Таким образом, архитектура и модель сети CNNHF может выполнять роль хеш-функции в криптографических приложениях.

5.3.2. Устойчивость к коллизиям хеш-функции, основанной на CNNHF

Устойчивость к коллизиям означает возможность существования двух разных входных сообщений ($M_1 \neq M_2$), генерирующих одно и то же выходное сообщение или хеш: $h(M_1) = h(M_2)$; см. п. 3 определения 2.5.

Устойчивость к коллизиям проанализирована на основании отклонения числа одинаковых хешей от условной средней величины (при отсутствии коллизий эта средняя величина равна 1). Анализ проводился для всех возможных входных M согласно формуле

$$\delta = \frac{\sum_{i=1}^{n} |e_i - 1|}{N}, \qquad (5.16)$$

где δ – это мера устойчивости к коллизиям данной функции хаоса; n – это число всех возможных входных величин M; e_i – это количество выступлений элемента i.

Результаты опытов, проведенных с использованием трех функций хаоса, представлены в таблице.

Устойчивость к коллизиям при использовании различных функций хаоса

Функция хаоса	δ
Уравнение, описывающее множество Жюлиа	0,36
Уравнение Дуффинга	0,99
Уравнение Энона	1,43

На основании полученных результатов можно утверждать, что наилучшей устойчивостью характеризуется уравнение, описывающее множество Жюлиа, наихудшей – уравнение Энона. Кроме того, уравнение (5.13) проще остальных с точки зрения объема вычислении, что положительно влияет на эффективность его использования в архитектуре CNNHF.

Мы получили подтверждение тому, что выбор соответствующей функции хаоса, исполняющей роль функции перехода, может существенно повлиять на безопасность всей системы, а также на ее эффективность.

Одной из возможных атак на функцию перехода является создание карты переходов, хранимой в массиве. Она позволяет определить для каждой выходной величины $h(M)$ множество возможных входных величин M. Единственная проблема – это размер массива. При операциях только с действительными числами размерность массива составляет $2^{32}\times2^{32}$, чтобы сохранить все возможные состояния функции перехода. В случае же функции перехода, оперирующей комплексными числами, размерность такого массива: $2^{32}\times2^{32}\times2^{32}\times2^{32}$. Это гигантский по объему требуемой памяти размер, и его хранение практически невозможно ни в одной из доступных компьютерных систем.

5.4. Использование кватернионов в операциях над хеш-функциями

Некоторые особенности использования кватернионов (как и октонионов) при хешировании сообщений на основе ХНС мы здесь рассматриваем, прежде всего, с точки зрения перспективы.

Сеть QNNHF (Quaternion Neural Network Hash Function – нейронная сеть на основе кватернионов для вычисления хеш-функции) должна использовать в своей конструкции хеш-функцию – нейронную сеть, основанную на кольце кватернионов.

В случае этой модели, как и в предыдущей модели (CNNHF), единственной существенной особенностью будет установление новой функции перехода, являющейся функцией хаоса. Мы можем использовать функции перехода, рассмотренные в отношении алгебры комплексных чисел. Речь может идти о tent-отображении, использованном в классической модели и приспособленном к специфике комплексных чисел. Данная функция имеет настолько простую конструкцию, что может быть применена как в комплексных числах, так и в кватернионах.

Кватернионы (см. формулу (4.10)) могут быть представлены как пара комплексных чисел в виде $q = (z_1, z_2)$, где $q \in \mathbb{H}$, $z_1 \in \mathbb{C}$, $z_2 \in \mathbb{C}$. Они могут быть использованы во всех конструкциях на основе хаотических отображений, применяющих (первоначально) пары действительных чисел, являющихся, по сути, комплексными числами.

Например, в уравнении Дуффинга (5.14) используем два кватерниона: $q_1 = (x_1, y_1)$ и $q_2 = (a, b)$. Таким образом, математический вид уравнения останется неизмененным. К тому же кватернионы создают кольцо (операция умножения не коммутативна), а комплексные числа, из которых оно состоит, сохраняют все значения (следовательно, сохраняют также коммутативность умножения). Анализируя уравнение, убеждаемся, что каждая координата вычисляется независимо от операции, произведенной в действительности над комплексными числами. Таким образом, проблема, связанная с коммутативностью операции умножения, отсутствует.

Аналогично мы могли бы использовать уравнение Энона (5.15) с такими же коэффициентами: $q_1 = (x_1, y_1)$ и $q_2 = (a, b)$.

Множество Жюлиа может быть использовано как функция перехода и в модели QNNHF. Именно в упомянутом множестве встречаются две операции: сложение и возведение в квадрат. Сложение кватернионов аналогично сложению комплексных чисел. Зато умножение не сохраняет коммутативности. Обратим внимание на тот факт, что возведение в степень, являющееся умножением одинаковых элементов, также не вызывает трудностей, связанных с некоммутативностью умножения в кольце кватернионов.

Приложение А
ЛИСТИНГ ПРОГРАММНОГО КОДА ДЛЯ ИССЛЕДОВАНИЯ АРХИТЕКТУРЫ TPCM

UCPerceptron.h

```
#ifndef UCPerceptronH
#define UCPerceptronH
#include <complex>
typedef std::complex<int> Cint;
class CPerceptron
{
  Cint *Weights;
  Cint *Inputs;
  Cint Output;
  void RandomWeights();
public:
  CPerceptron();
  ~CPerceptron();
  Cint GetOutput(Cint *AInputs);
  void AktualizeWeights(Cint OutputTPM);
  int Distance(const CPerceptron &p);
};
#endif
//---------------------------------------------------------------------------
```

UCPerceptron.cpp

```
#pragma hdrstop
#include "UCPerceptron.h"
#include "UTPCM.h"
#pragma package(smart_init)
CPerceptron::CPerceptron()
{
  Weights = new Cint[NC];
  RandomWeights();
```

```
}
CPerceptron::~CPerceptron()
{
  delete []Weights;
}
Cint CPerceptron::GetOutput(Cint *AInputs)
{
  Inputs = AInputs;
  Output = Cint(0, 0);
  for (int i = 0; i < NC ; i++)
    Output += Inputs[i]*Weights[i];
  if (abs(Output.real())>=abs(Output.imag()))
    {
      if (Output.real()>=0)
        Output._M_re = 1;
      else
        Output._M_re = -1;
     Output._M_im = 0;
    }
  else
    {
    if (Output.imag()>=0)
        Output._M_im = 1;
      else
        Output._M_im = -1;
     Output._M_re = 0;
    }
  return Output;
}

void CPerceptron::RandomWeights()
{
  for (int i = 0; i < NC; i++)
  {
```

```
    Weights[i]._M_re = L - random(2*L);
    Weights[i]._M_im = L - random(2*L);
  }
}
void CPerceptron::AktualizeWeights(Cint OutputTPM)
{
if (OutputTPM == Output)
for (int i = 0; i < NC; i++)
  {
   Weights[i] += Output*Inputs[i];

   if (Weights[i].real() > L) Weights[i]._M_re = L;
     else
     if (Weights[i].real() < -L) Weights[i]._M_re = -L;

   if (Weights[i].imag() > L) Weights[i]._M_im = L;
     else
     if (Weights[i].imag() < -L) Weights[i]._M_im = -L;
  }
}
int CPerceptron::Distance(const CPerceptron &P)
{
  int Result = 0;
   for (int i = 0; i < NC; i++)
    Result += norm(Weights[i] - P.Weights[i]);
  return Result;
}
//---------------------------------------------------------------------------
```

UTPCM.h

```
#ifndef UTPCMH
#define UTPCMH
#include "UTPM.h"
#include "UCPerceptron.h"
const int NC = N/2;
```

```
class TPCM
{
  CPerceptron CPerceptrons[K];
  Cint Output;
public:
  Cint GetOutput(Cint AInputs[K][NC]);
  void Synchronize();
  void ModifyOutput(Cint AOutput){Output = AOutput;};
  int Distance(const TPCM &ATPCM);
};
#endif
//---------------------------------------------------------------------------
```

UTPCM.cpp

```
#pragma hdrstop
#include "UTPCM.h"
#pragma package(smart_init)
Cint TPCM::GetOutput(Cint AInputs[K][NC])
{
  Output = Cint(1);
  for (int i = 0; i < K ; i++)
    Output *= CPerceptrons[i].GetOutput(AInputs[i]);
  return Output;
}
void TPCM::Synchronize()
{
  for (int i = 0; i < K; i++)
    CPerceptrons[i].AktualizeWeights(Output);
}
int TPCM::Distance(const TPCM &ATPCM)
{
  int Result = 0;
   for (int i = 0; i < K; i++)
    Result += CPerceptrons[i].Distance(ATPCM.CPerceptrons[i]);
  return Result;
}
//---------------------------------------------------------------------------
```

Приложение Б
ЛИСТИНГ ПРОГРАММНОГО КОДА ДЛЯ ИССЛЕДОВАНИЯ АРХИТЕКТУРЫ TPQM

```
UQuaternion.h
#ifndef UQuaternionH
#define UQuaternionH
class Qint
{
  int Elements[4];
public:
  Qint(int Aa = 0, int Ab = 0, int Ac = 0, int Ad = 0);
  int GetElement(int i) {return Elements[i];};
  void SetElement(int i, int v);
  friend Qint operator + (const Qint& Q1, const Qint& Q2);
  const Qint& operator += (const Qint& Q);
  friend Qint operator * (const Qint& Q1, const Qint& Q2);
  const Qint& operator *= (const Qint& Q);
  friend Qint operator - (const Qint& Q1, const Qint& Q2);
  friend bool operator == (const Qint& Q1, const Qint& Q2);
  friend int Norm(Qint Q);
};
#endif
//---------------------------------------------------------------------------
UQuaternion.cpp
#pragma hdrstop
#include "UQuaternion.h"
#pragma package(smart_init)
Qint::Qint(int Aa, int Ab, int Ac, int Ad)
{
  Elements[0] = Aa;
  Elements[1] = Ab;
  Elements[2] = Ac;
```

```
  Elements[3] = Ad;
}
void Qint::SetElement(int i, int v)
{
  Elements[i] = v;
}
Qint operator + (const Qint &Q1, const Qint &Q2)
{
  Qint Q;
  Q.Elements[0] = Q1.Elements[0] + Q2.Elements[0];
  Q.Elements[1] = Q1.Elements[1] + Q2.Elements[1];
  Q.Elements[2] = Q1.Elements[2] + Q2.Elements[2];
  Q.Elements[3] = Q1.Elements[3] + Q2.Elements[3];
  return Q;
}
const Qint& Qint::operator += (const Qint &Q)
{
  Elements[0] += Q.Elements[0];
  Elements[1] += Q.Elements[1];
  Elements[2] += Q.Elements[2];
  Elements[3] += Q.Elements[3];
  return *this;
}
Qint operator * (const Qint &Q1, const Qint &Q2)
{
  Qint Q;
  Q.Elements[0] = Q1.Elements[0]*Q2.Elements[0] - (Q1.Elements[1]*Q2.Elements[1] + Q1.Elements[2]*Q2.Elements[2] + Q1.Elements[3]*Q2.Elements[3]);
  Q.Elements[1] = Q1.Elements[0]*Q2.Elements[1] + Q2.Elements[0]*Q1.Elements[1] + Q1.Elements[2]*Q2.Elements[3] - Q1.Elements[3]*Q2.Elements[2];
  Q.Elements[2] = Q1.Elements[0]*Q2.Elements[2] + Q2.Elements[0]*Q1.Elements[2] + Q1.Elements[3]*Q2.Elements[1] - Q1.Elements[1]*Q2.Elements[3];
  Q.Elements[3] = Q1.Elements[0]*Q2.Elements[3] + Q2.Elements[0]*Q1.Elements[3] + Q1.Elements[1]*Q2.Elements[2] - Q1.Elements[2]*Q2.Elements[1];
  return Q;
}
const Qint& Qint::operator *= (const Qint &Q)
```

```
{
  Elements[0] = Elements[0]*Q.Elements[0] - (Elements[1]*Q.Elements[1] + Elements[2]*Q.Elements[2] + Elements[3]*Q.Elements[3]);
  Elements[1] = Elements[0]*Q.Elements[1] + Q.Elements[0]*Elements[1] + Elements[2]*Q.Elements[3] - Elements[3]*Q.Elements[2];
  Elements[2] = Elements[0]*Q.Elements[2] + Q.Elements[0]*Elements[2] + Elements[3]*Q.Elements[1] - Elements[1]*Q.Elements[3];
  Elements[3] = Elements[0]*Q.Elements[3] + Q.Elements[0]*Elements[3] + Elements[1]*Q.Elements[2] - Elements[2]*Q.Elements[1];
  return *this;
}
Qint operator - (const Qint &Q1, const Qint &Q2)
{
  Qint Q;
  Q.Elements[0] = Q1.Elements[0] - Q2.Elements[0];
  Q.Elements[1] = Q1.Elements[1] - Q2.Elements[1];
  Q.Elements[2] = Q1.Elements[2] - Q2.Elements[2];
  Q.Elements[3] = Q1.Elements[3] - Q2.Elements[3];
  return Q;
}
bool operator == (const Qint& Q1, const Qint& Q2)
{
  for (int i = 0; i < 4; i++)
    if (Q1.Elements[i] != Q2.Elements[i])
      return false;
  return true;
}
int Norm(Qint Q)
{
int Result = 0;
  for (int i = 0; i < 4; i++)
   {
     Result += Q.Elements[i]*Q.Elements[i];
   }
return Result;
```

```
}
//---------------------------------------------------------------------------

#ifndef UTPQMH
#define UTPQMH
//---------------------------------------------------------------------------
#include "UTPM.h"
#include "UQPerceptron.h"
//---------------------------------------------------------------------------
UTPQM.h
const int NQ = N/4;
class TPQM
{
  QPerceptron QPerceptrons[K];
  Qint Output;
public:
  Qint GetOutput(Qint AInputs[K][NQ]);
  void Synchronize();
  int Distance(const TPQM &ATPQM);
  void ModifyOutput(Qint AOutput){Output = AOutput;};
};
#endif
//---------------------------------------------------------------------------
UTPQM.cpp
#pragma hdrstop
#include "UTPQM.h"
#pragma package(smart_init)
Qint TPQM::GetOutput(Qint AInputs[K][NQ])
{
  Output = Qint(1);
  for (int i = 0; i < K ; i++)
    Output *= QPerceptrons[i].GetOutput(AInputs[i]);
  return Output;
}
```

```
void TPQM::Synchronize()
{
  for (int i = 0; i < K; i++)
    QPerceptrons[i].AktualizeWeights(Output);
}
int TPQM::Distance(const TPQM &ATPQM)
{
  int Result = 0;
   for (int i = 0; i < K; i++)
    Result += QPerceptrons[i].Distance(ATPQM.QPerceptrons[i]);
  return Result;
}
//---------------------------------------------------------------------------
```

Приложение В
ЛИСТИНГ ПРОГРАММНОГО КОДА ДЛЯ ИССЛЕДОВАНИЯ АРХИТЕКТУРЫ ТРОМ

UOctonion.h

```
#ifndef UOctonionH
#define UOctonionH
class Oint
{
  int Elements[8];
public:
  Oint(int Aa = 0, int Ab = 0, int Ac = 0, int Ad = 0, int Ae = 0, int Af = 0, int Ag
= 0, int Ah = 0);
  int GetElement(int i) {return Elements[i];};
  void SetElement(int i, int v) {Elements[i] = v;};
  friend Oint operator + (const Oint& O1, const Oint& O2);
  const Oint& operator += (const Oint& O);
  friend Oint operator * (const Oint& O1, const Oint& O2);
  const Oint& operator *= (const Oint& O);
  friend Oint operator - (const Oint& O1, const Oint& O2);
  friend bool operator == (const Oint& O1, const Oint& O2);
  friend int Norm(Oint O);
};
#endif
//---------------------------------------------------------------------------
```

UOPerceptron.cpp

```
#pragma hdrstop
#include "UOctonion.h"
#pragma package(smart_init)
Oint::Oint(int Aa, int Ab, int Ac, int Ad, int Ae, int Af, int Ag, int Ah)
{
  Elements[0] = Aa;
  Elements[1] = Ab;
```

```
  Elements[2] = Ac;
  Elements[3] = Ad;
  Elements[4] = Ae;
  Elements[5] = Af;
  Elements[6] = Ag;
  Elements[7] = Ah;
}
Oint operator + (const Oint &O1, const Oint &O2)
{
  Oint O;
  O.Elements[0] = O1.Elements[0] + O2.Elements[0];
  O.Elements[1] = O1.Elements[1] + O2.Elements[1];
  O.Elements[2] = O1.Elements[2] + O2.Elements[2];
  O.Elements[3] = O1.Elements[3] + O2.Elements[3];
  O.Elements[4] = O1.Elements[4] + O2.Elements[4];
  O.Elements[5] = O1.Elements[5] + O2.Elements[5];
  O.Elements[6] = O1.Elements[6] + O2.Elements[6];
  O.Elements[7] = O1.Elements[7] + O2.Elements[7];
  return O;
}
const Oint& Oint::operator += (const Oint &O)
{
  Elements[0] += O.Elements[0];
  Elements[1] += O.Elements[1];
  Elements[2] += O.Elements[2];
  Elements[3] += O.Elements[3];
  Elements[4] += O.Elements[4];
  Elements[5] += O.Elements[5];
  Elements[6] += O.Elements[6];
  Elements[7] += O.Elements[7];
  return *this;
}
Oint operator * (const Oint &O1, const Oint &O2)
{
```

```
    Oint O;
    O.Elements[0] = O1.Elements[0]*O2.Elements[0]-O1.Elements[1]*O2.Elements[1]-O1.Elements[2]*O2.Elements[2]-O1.Elements[3]*O2.Elements[3]-O1.Elements[4]*O2.Elements[4]-O1.Elements[5]*O2.Elements[5]-O1.Elements[6]*O2.Elements[6]-O1.Elements[7]*O2.Elements[7];
    O.Elements[1] = O1.Elements[0]*O2.Elements[1]+O1.Elements[1]*O2.Elements[0]+O1.Elements[2]*O2.Elements[3]-O1.Elements[3]*O2.Elements[2]+O1.Elements[4]*O2.Elements[5]-O1.Elements[5]*O2.Elements[4]-O1.Elements[6]*O2.Elements[7]+O1.Elements[7]*O2.Elements[6];
    O.Elements[2] = O1.Elements[0]*O2.Elements[2]-O1.Elements[1]*O2.Elements[3]+O1.Elements[2]*O2.Elements[0]+O1.Elements[3]*O2.Elements[1]+O1.Elements[4]*O2.Elements[6]+O1.Elements[5]*O2.Elements[7]-O1.Elements[6]*O2.Elements[4]-O1.Elements[7]*O2.Elements[5];
    O.Elements[3] = O1.Elements[0]*O2.Elements[3]+O1.Elements[1]*O2.Elements[2]-O1.Elements[2]*O2.Elements[1]+O1.Elements[3]*O2.Elements[0]+O1.Elements[4]*O2.Elements[7]-O1.Elements[5]*O2.Elements[6]+O1.Elements[6]*O2.Elements[5]-O1.Elements[7]*O2.Elements[4];
    O.Elements[4] = O1.Elements[0]*O2.Elements[4]-O1.Elements[1]*O2.Elements[5]-O1.Elements[2]*O2.Elements[6]-O1.Elements[3]*O2.Elements[7]+O1.Elements[4]*O2.Elements[0]+O1.Elements[5]*O2.Elements[1]+O1.Elements[6]*O2.Elements[2]+O1.Elements[7]*O2.Elements[3];
    O.Elements[5] = O1.Elements[0]*O2.Elements[5]+O1.Elements[1]*O2.Elements[4]-O1.Elements[2]*O2.Elements[7]+O1.Elements[3]*O2.Elements[6]-O1.Elements[4]*O2.Elements[1]+O1.Elements[5]*O2.Elements[0]-O1.Elements[6]*O2.Elements[3]+O1.Elements[7]*O2.Elements[2];
    O.Elements[6] = O1.Elements[0]*O2.Elements[6]+O1.Elements[1]*O2.Elements[7]+O1.Elements[2]*O2.Elements[4]-O1.Elements[3]*O2.Elements[5]-O1.Elements[4]*O2.Elements[2]+O1.Elements[5]*O2.Elements[3]+O1.Elements[6]*O2.Elements[0]-O1.Elements[7]*O2.Elements[1];
    O.Elements[7] = O1.Elements[0]*O2.Elements[7]-O1.Elements[1]*O2.Elements[6]+O1.Elements[2]*O2.Elements[5]+O1.Elements[3]*O2.Elements[4]-O1.Elements[4]*O2.Elements[3]-O1.Elements[5]*O2.Elements[2]+O1.Elements[6]*O2.Elements[1]+O1.Elements[7]*O2.Elements[0];
    return O;
  }
  const Oint& Oint::operator *= (const Oint &O)
  {
    Elements[0] = Elements[0]*O.Elements[0]-Elements[1]*O.Elements[1]-Elements[2]*O.Elements[2]-Elements[3]*O.Elements[3]-Elements[4]*O.Elements[4]-Elements[5]*O.Elements[5]-Elements[6]*O.Elements[6]-Elements[7]*O.Elements[7];
    Elements[1] = Elements[0]*O.Elements[1]+Elements[1]*O.Elements[0]+Elements[2]*O.Elements[3]-Elements[3]*O.Elements[2]+Elements[4]*O.Elements[5]-
```

```
Elements[5]*O.Elements[4]-Elements[6]*O.Elements[7]+Elements[7]*O.Ele-
ments[6];

      Elements[2] = Elements[0]*O.Elements[2]-Elements[1]*O.Elements[3]+Ele-
ments[2]*O.Elements[0]+Elements[3]*O.Elements[1]+Elements[4]*O.Ele-
ments[6]+Elements[5]*O.Elements[7]-Elements[6]*O.Elements[4]-Ele-
ments[7]*O.Elements[5];

      Elements[3] = Elements[0]*O.Elements[3]+Elements[1]*O.Elements[2]-Ele-
ments[2]*O.Elements[1]+Elements[3]*O.Elements[0]+Elements[4]*O.Elements[7]-
Elements[5]*O.Elements[6]+Elements[6]*O.Elements[5]-Elements[7]*O.Ele-
ments[4];

      Elements[4] = Elements[0]*O.Elements[4]-Elements[1]*O.Elements[5]-Ele-
ments[2]*O.Elements[6]-Elements[3]*O.Elements[7]+Elements[4]*O.Ele-
ments[0]+Elements[5]*O.Elements[1]+Elements[6]*O.Elements[2]+Ele-
ments[7]*O.Elements[3];

      Elements[5] = Elements[0]*O.Elements[5]+Elements[1]*O.Elements[4]-Ele-
ments[2]*O.Elements[7]+Elements[3]*O.Elements[6]-Elements[4]*O.Ele-
ments[1]+Elements[5]*O.Elements[0]-Elements[6]*O.Elements[3]+Ele-
ments[7]*O.Elements[2];

      Elements[6] = Elements[0]*O.Elements[6]+Elements[1]*O.Elements[7]+Ele-
ments[2]*O.Elements[4]-Elements[3]*O.Elements[5]-Elements[4]*O.Ele-
ments[2]+Elements[5]*O.Elements[3]+Elements[6]*O.Elements[0]-Ele-
ments[7]*O.Elements[1];

      Elements[7] = Elements[0]*O.Elements[7]-Elements[1]*O.Elements[6]+Ele-
ments[2]*O.Elements[5]+Elements[3]*O.Elements[4]-Elements[4]*O.Elements[3]-
Elements[5]*O.Elements[2]+Elements[6]*O.Elements[1]+Elements[7]*O.Ele-
ments[0];

      return *this;

    }

    Oint operator - (const Oint &O1, const Oint &O2)

    {

      Oint O;

      O.Elements[0] = O1.Elements[0] - O2.Elements[0];

      O.Elements[1] = O1.Elements[1] - O2.Elements[1];

      O.Elements[2] = O1.Elements[2] - O2.Elements[2];

      O.Elements[3] = O1.Elements[3] - O2.Elements[3];

      O.Elements[4] = O1.Elements[4] - O2.Elements[4];

      O.Elements[5] = O1.Elements[5] - O2.Elements[5];

      O.Elements[6] = O1.Elements[6] - O2.Elements[6];

      O.Elements[7] = O1.Elements[7] - O2.Elements[7];

      return O;
```

```
}
bool operator == (const Oint& O1, const Oint& O2)
{
  for (int i = 0; i < 8; i++)
    if (O1.Elements[i] != O2.Elements[i])
      return false;
  return true;
}
int Norm(Oint O)
{
int Result = 0;
  for (int i = 0; i < 8; i++)
   {
     Result += O.Elements[i]*O.Elements[i];
   }
return Result;
}
//---------------------------------------------------------------------------
```

UTPOM.h

```
#ifndef UTPOMH
#define UTPOMH
#include "UTPM.h"
#include "UOPerceptron.h"
const int NO = N/8;
class TPOM
{
  OPerceptron OPerceptrons[K];
  Oint Output;
public:
  Oint GetOutput(Oint AInputs[K][NO]);
  void Synchronize();
  int Distance(const TPOM &ATPOM);
  void ModifyOutput(Oint AOutput){Output = AOutput;};
};
```

```
#endif
//---------------------------------------------------------------------------
```

UTPOM.cpp

```
#pragma hdrstop
#include "UTPOM.h"
#pragma package(smart_init)
Oint TPOM::GetOutput(Oint AInputs[K][NO])
{
  Output = Oint(1);
  for (int i = 0; i < K ; i++)
    Output *= OPerceptrons[i].GetOutput(AInputs[i]);
  return Output;
}
void TPOM::Synchronize()
{
  for (int i = 0; i < K; i++)
    OPerceptrons[i].AktualizeWeights(Output);
}
int TPOM::Distance(const TPOM &ATPOM)
{
  int Result = 0;
   for (int i = 0; i < K; i++)
    Result += OPerceptrons[i].Distance(ATPOM.OPerceptrons[i]);
  return Result;
}
```

ЛИТЕРАТУРА

1. McCarthy J. Programs with common sense // Proceedings of the Teddington conference on the mechanization of thought processes. London, 1959. P. 75–91.

2. Rutkowski L. Metody i techniki sztucznej inteligencji. Warszawa: PWN, 2006. 350 s.

3. Turing A. Computing machinery and intelligence // Mind. 1950. Vol. LIX, iss. 236. P. 433–460.

4. Grabowska A., Królicki L. Emisyjna tomografia pozytronowa (PET) i jej organizacji mózgu // Kosmos. 1997. T. 46, Nr 3. S. 393–403.

5. Penfield W., Jasper H. Epilepsy and the functional anatomy of the human brain. Boston: Little Brown and Co., 1954. 899 p.

6. Головко В. А., Краснопрошин В. В. Нейросетевые технологии обработки данных: учеб. пособие. Минск: БГУ, 2017. 263 с.

7. Dendritic spikes enhance stimulus selectivity in cortical neurons in vivo / S. Smith [et al.] // Nature. 2013. Vol. 503. P. 115–120. DOI: 10.1038/nature12600.

8. McCulloch W., Pitts W. A logical calculus of the ideas immanent in nervous activity // Bulletin of Mathematical Biophysics. 1943. Vol. 5. P. 115–133.

9. Хайкин С. Нейронные сети: полный курс: пер. с англ. 2-е изд. М.: И. Д. Вильямс, 2016. 1104 с.

10. Mason S. Feedback Theory – further properties of signal flow graphs // Proceedings of the Institute of Radio Engineers. 1956. Vol. 44, no. 7. P. 920–926.

11. Hassoun M. Fundamentals of Artificial Neural Networks. MIT Press, 1995. 511 p.

12. Кан К. А. Нейронные сети. Эволюция. [Б. м.]: Литрес Самиздат, 2018. 380 с.

13. Розенблатт Ф. Принципы нейродинамики: персептрон и теория механизмов мозга: пер. с англ. М.: Мир, 1965. 480 с.

14. Дюк В., Самойленко А. Data Mining. Учебный курс. СПб.: Питер, 2003. 368 с.

15. Галушкин А. И. Синтез многослойных систем распознавания образов. М.: Энергия, 1974. 367 с.

16. Rumelhart D. E., Hinton G. E., Williams R. J. Learning representations by back-propagating errors // Nature. 1986. Vol. 323. P. 533–536. DOI: org/10.1038/323533a0.

17. Hopfield J. Neural networks and physical systems with emergent collective computational abilities // Proceedings of the National Academy of Sciences of the United States of America, April 1, 1982. 1982. Vol. 79, no. 8. P. 2554–2558. DOI: 10.1073/pnas.79.8.2554.

18. Hebb D. O. The organization of behavior: a neuropsychological theory. New York, 2002. 378 p. DOI: org/10.4324/9781410612403.[15]

19. Kohonen T. Sell-organised formation of topologically correct feature maps // Biological Cybernetics. 1982. Vol. 43, no. 1. P. 59–69. DOI: 10.1007/BF00337288.

20. Osowski S. Sieci neuronowe w ujęciu algorytmicznym. Warszawa: WNT, 1996. 348 s.

21. Саттон Р. С., Барто Э. Г. Обучение с подкреплением = Reinforcement Learning. М.: БИНОМ. Лаборатория знаний, 2017. 399 с.

22. Yamakawa H., Okabe Y. A neural network-like critic for reinforcement learning // Neural networks. 1995. Vol. 8, iss. 3. P. 363–373. DOI: org/10.1016/0893-6080(94)00086-2.

23. Werbos P. J. Applications of advances in nonlinear sensitivity analysis // System modeling and optimization. Lecture notes in control and information sciences, vol. 38 / R. F. Drenick, F. Kozin (eds). Berlin; Heidelberg: Springer, 1982. P. 762–770. DOI: org/10.1007/BFb0006203.

24. Biehl M., Caticha N. Statistical mechanics of online learning and generalization // The handbook of brain theory and neural networks / ed. by M. A. Arbib. Berlin: MIT Press, 2001. P. 1–22.

25. Hui S., Zak S. H. The Widrow – Hoff algorithm for McCulloch – Pitts type neurons // IEEE Transactions on Neural Networks. 1994. Vol. 5, no. 6. P. 924–929. DOI: 10.1109/72.329689.

26. Widrow B., Hoff M. Adaptive switching circuits. New York: IRE, 1960. P. 96–104.

27. Barto A. G., Sutton R. S., Anderson C. W. Neuronlike adaptive elements that can solve difficult learning control problems // IEEE Transactions on Systems, Man, and Cybernetics. 1983. Vol. SMC-13, iss. 5. P. 834–846. DOI: 10.1109/TSMC.1983.6313077.

28. Fausett L. V. Fundamentals of neural networks: architectures, algorithms and applications. Pearson Education India, 2006. 480 p.

29. Oja E. A simplified neuron model as a principal component analyzer // J. Math. Biology. 1982. Vol. 15. P. 267–273.

[15] Оригинальное издание – 1949 год.

30. Kanter I., Kinzel W., Kanter E. Secure exchange of information by synchronization of neural networks // Europhysics Letters. 2002. P. 141–147. DOI: 10.1209/epl/i2002-00552-9.

31. Kinzel W., Kanter I. Neural cryptography // Proceedings of the 9th International conference on neural information processing: ICONIP'02. 2002. Vol. 3. P. 1351–1354. DOI: org/10.48550/arXiv.cond-mat/0208453.

32. Kanter I., Kinzel W. The theory of neural networks and cryptography // Proceeding of the XXII solvay conference on physics. The physics of communication. 2003. P. 631–644. DOI: org/10.1142/9789812704634_0044.

33. Kanter I., Kinzel W. The theory of neural networks and cryptography // Quantum computers and computing. 2005. Vol. 5, no. 1. P. 130–140. DOI: org/10.1142/9789812704634_0044.

34. Synchronization of neural networks by mutual learning and its application to cryptography / E. Klein [et al.] // Advances in neural information processing systems 17 (NIPS 2004). MIT Press Cambridge, 2005. P. 689–696.

35. Klimov A., Mityagin A., Shamir A. Analysis of neural cryptography // Advances in cryptology – ASIACRYPT 2002 / Y. Zheng (eds). Springer, 2003. P. 288–289. DOI: org/10.1007/3-540-36178-2_18.

36. Kim C.-M., Rim S., Kye W.-H. Sequential synchronization of chaotic systems with an application to communication // Physical Review Lett. 2002. Vol. 88, no. 1. Art. 014103. DOI: 10.1103/PhysRev Lett.88.014103.

37. Kinzel W., Metzler R., Kanter I. Dynamics of interacting neural networks // J. Phys. A: Math. Gen. 2000. Vol. 33, no. 14. P. L141–L147. DOI: 10.1088/0305-4470/33/14/101.

38. Thompson K. The brain. An introduction to neuroscience. 3th ed. Worth Publishers, 2000. 480 p.

39. Zhang G. P. Neural networks in business forecasting. Information Science Publishing, 2003. 350 p.

40. McNelis P. D. Neural networks in finance: gaining predictive edge in the market. Elsevier Academic Press, 2004. 256 p.

41. Diffie W., Hellman M. New directions in cryptography // IEEE Transactions on Information Theory. 1976. Vol. IT-22, no. 6. P. 644–654. DOI: 10.1109/TIT.1976.1055638.

42. Ruttor A. Neural synchronization and cryptography, Dissertation zur Erlangung des naturwissenschaftlichen Doktorgrades der Bayerischen Julius-Maximilians-Universitat Wurzburg. Würzburg, 2006. 121 p.

43. Шеннон К. Работы по теории информации и кибернетике. М.: Изд-во иностранной литературы, 1963. 830 с.

44. Shannon C. E. Communication theory of secrecy systems // Bell System Tech. J. 1949. Vol. 28. P. 657–715.

45. Фергюсон Н., Шнайер Б. Практическая криптография: пер. с англ. М.: Издательский дом «Вильямс», 2004. 432 с.

46. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке СИ. М.: Триумф, 2012. 420 с.

47. Введение в криптографию / под общ. ред. В. В. Ященко. 4-е изд., доп. М.: МЦНМО, 2012. 384 с.

48. Pieprzyk J., Hardjono T., Seberry J. Fundamentals of computer security. Berlin; Heidelberg: Springer-Verlag, 2003. 520 p.

49. Криптология: учебник / Ю. С. Харин [и др.]. Минск: БГУ, 2013. 511 с.

50. Урбанович П. П. Защита информации методами криптографии, стеганографии и обфускации: учеб.-метод. пособие. Минск: БГТУ, 2016. 218 с.

51. Урбанович П. П., Шутько Н. П. Лабораторный практикум по дисциплинам «Защита информации и надежность информационных систем» и «Криптографические методы защиты информации». В 2 ч. Ч. 2. Криптографические и стеганографические методы защиты информации: учеб.-метод. пособие. Минск: БГТУ, 2020. 226 с.

52. Benaloh J., Yung M. Distributing the power of a government to enhance the privacy of the voters // ACM symposium on principles of distributed computing, November 1986. Calgary, Alberta, New York, 1986. P. 52–62. DOI: org/10.1145/10590.10595.

53. Jafar U., Aziz M. J. A., Shukur Z. Blockchain for electronic voting system – review and open research challenges // Sensors. 2021. Vol. 21. P. 5874. DOI: org/10.3390/s21175874.

54. Feige U., Fiat A., Shamir A. Zero-knowledge proofs of identity // Journal of Cryptology. 1988. Vol. 1, no. 2. P. 77–94. DOI: org/10.1007/BF02351717.

55. Peng K. Attack against a batch zero-knowledge proof system // IET Information Security – IET. 2012. Vol. 6, iss. 1. P. 1–5. DOI: 10.1049/iet-ifs.2011.0290.

56. Thaler J. Proofs, arguments, and zero-knowledge. URL: https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf (date of access: 15.05.2023).

57. Ben-David A., Nisan N., Pinkas B. Fairplay MP: a system for secure multi-party computation // CCS'08: Proceedings of the 15th ACM

conference on computer and communications security – 2008. Vol. ACM. P. 257–266. DOI: org/10.1145/1455770.1455804.

58. Secure multi-party computation on blockchain: an overview / H. Zhong [et al.] // Parallel architectures, algorithms and programming. PAAP 2019. Communications in computer and information science / H. Shen, Y. Sang (eds). Singapore: Springer, 2020. P. 1163. DOI: org/10.1007/978-981-15-2767-840.

59. Browkin J. Wybrane zagadnienia z algebry. Warszawa: PWN, 1968. 267 s.

60. Pohlig S., Hellman M. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (Corresp.) // IEEE Transactions on Information Theory. 1978. Vol. 24, no. 1. P. 106–110. DOI: 10.1109/TIT.1978.1055817.

61. ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms // IEEE Transactions on Information Theory. 1985. Vol. 31, no. 4. P. 469–472. DOI: 10.1109/ TIT.1985.1057074.

62. Альбов А. Квантовая криптография. СПб.: ООО «Страта», 2015. 248 с.

63. Иванов М. А., Стариковский А. В. Хеш-функции. Теория, применение и новые стандарты (часть 1). URL: http://www.aha.ru/~msa/papers7.pdf (дата обращения: 12.04.2023).

64. Preneel B. Cryptographic hash functions: an overview. URL: https://www.esat.kuleuven.be/cosic/publications/article-289.pdf (date of access: 20.04.2023).

65. Damgard I. B. A design principle for hash functions // Advances in cryptology – CRYPTO'89 Proceedings. Santa Barbara, California, USA, August 20–24, 1989. Lecture Notes in Computer Science / G. Brassard (eds). New York: Springer-Verlag, 1990, Vol. 435. P. 416–427. DOI: org/10.1007/0-387-34805-0_39.

66. Merkle R. One way hash functions and DES // Advances in cryptology – CRYPTO'89 Proceedings. Santa Barbara, California, USA, August 20–24, 1989. Lecture Notes in Computer Science / G. Brassard (eds). New York: Springer-Verlag, 1990, Vol. 435. P. 428–446. DOI: org/10.1007/0-387-34805-0_40.

67. Lai X., Massey J. L. Hash functions based on block ciphers // Advances in cryptology – EUROCRYPT'92. Lecture Notes in Computer Science / R. A. Rueppel (eds). Berlin, Heidelberg: Springer, 1993. Vol. 658. P. 55–70. DOI: org/10.1007/3-540-47555-9_5.

68. Dourlens S. Applied neuro-cryptography // Maîtrise Informatique, Department of Microcomputers and Microelectronics, University of Paris, 1995. DOI: 10.13140/RG.2.2.35476.24960.

69. Zolfaghari B., Koshiba T. The dichotomy of neural networks and cryptography: war and peace // Applied System Innovation. 2022. Vol. 5, no. 4. P. 61. DOI: org/10.3390/asi5040061.

70. Vandewalle J., Preneel B., Csapodi M. Data security issues, cryptographic protection methods, and the use of cellular neural networks and cellular automata // Proceedings of the 5th IEEE international workshop on cellular neural networks and their applications, Proceedings (Cat. No. 98TH8359), London, UK, April 14–17, 1998. London, 1998. P. 39–44. DOI: 10.1109/CNNA.1998.685326.

71. Schmidt T., Rahnama H., Sadeghian A. A review of applications of artificial neural networks in cryptosystems // Proceedings of the world automation congress, Waikoloa, HI, USA, September 28 – October 2, 2008. Waikoloa, 2008. P. 1–6.

72. Hu D., Wang Y. Security research on WiMAX with neural cryptography // Proceedings of the international conference on information security and assurance, Busan, Korea, April 24–26, 2008. Busan, 2008. P. 370–373. DOI: 10.1109/ISA.2008.17.

73. Применение искусственных нейронных сетей и системы остаточных классов в криптографии / Н. И. Червяков [и др.]. М.: ФИЗМАТЛИТ, 2012. 280 с.

74. Hadke P. P., Kale S. G. Use of neural networks in cryptography: a review // Proceedings of the world conference on futuristic trends in research and innovation for social welfare (Startup Conclave), Coimbatore, India, February 29 – March 1, 2016. Coimbatore, 2016. P. 1–4. DOI: 10.1109/STARTUP.2016.7583925.

75. CryptoNets: applying neural networks to encrypted data with high throughput and accuracy / N. Dowlim [et al.] // Proceedings of the 33rd international conference on machine learning, New York, June 20–22, 2016. New York, 2016. Vol. 48. P. 201–210.

76. Metzler R., Kinzel W., Kanter I. Interacting neural networks // Physical review E. 2000. Vol. 62, no. 2. P. 2555–2565. DOI: org/10.48550/arXiv.cond-mat/0003051.

77. Volkmer M., Wallner S. Tree parity machine rekeying architectures // IEEE Transactions on Computers. 2005. Vol. 54, no. 4. P. 421–427. DOI: 10.1109/TC.2005.70.

78. Volkmer M., Wallner S. Tree parity machine rekeying architectures for embedded security // Cryptology ePrint Archive, Inst. Comput. Technol., Hamburg Univ. Technol., Hamburg, Germany, Rep. 2005/235. 12 p.

79. Плонковски М., Урбанович П. П. Использование нейронных сетей в системах криптографического преобразования информации // Известия Белорусской инженерной академии. 2004. № 1 (17). С. 13–15.

80. Плонковски М., Урбанович П. П. Криптографическое преобразование информации на основе нейросетевых технологий // Труды БГТУ. Сер. VI, Физ.-мат. науки и информатика. 2005. Вып. XIII. С. 161–164.

81. Dolecki M. Statistical analysis of tree parity machine synchronization time // Труды БГТУ. 2012. № 6 (153): Физ.-мат. науки и информатика. С. 149–151.

82. Malzahn D., Engel A. Correlations between hidden units in multilayer neural networks and replica symmetry breaking // Physical Review. 1999. Vol. 60 (2 Pt B). P. 2097–2104. DOI: 10.1103/PhysRevE. 60.2097.

83. Abadi M., Andersen D. Learning to protect communications with adversarial neural cryptography. URL: arXiv preprint arXiv:1610.06918 (date of access: 15.07.2023).

84. Dorokhin S. E., Fuertes W., Lascano E. On the development of an optimal structure of tree parity machine for the establishment of a cryptographic key // Secur. Commun. Networks. 2019. P. 1–10. DOI: org/10.1155/ 2019/8214681.

85. Sarkar A., Khan M. Z., Alahmadi A. Mutual learning-based group synchronization of neural networks // Sadhana. 2022. Vol. 47. P. 243. DOI: org/10.1007/s12046-022-02010-1.

86. Neural cryptography based on generalized tree parity machine for real-life systems / S. Jeong [et al.] // Secur. Commun. Networks. 2021. Art. 6680782. DOI: org/10.1155/2021/6680782.

87. Sarkar A. Mutual learning-based efficient synchronization of neural networks to exchange the neural key // Complex & Intelligent Systems. 2022. Vol. 8. P. 307–321. DOI: org/10.1007/s40747-021-00344-7.

88. Лисица Е. В., Урбанович П. П. Моделирование криптографических систем на основе нейронных сетей // Автоматический контроль и автоматизация производственных процессов: материалы Международной научно-технической конференции, Минск, 28–29 окт. 2009 г. Минск, 2009. С. 79–81. URL: https://elib.belstu.by/handle/123456789/25734 (дата обращения: 15.04.2023).

89. Dolecki M., Kozera R., Lenik K. The evaluation of the TPM synchronization on the basis of their outputs // Journal of Achievements in Materials and Manufacturing Engineering. 2013. Vol. 57, no. 2. P. 91–98.

90. Urbanowicz P., Kozera R., Dolecki M. Zastosowanie sztucznych sieci neuronowych do uwzględniania kluczy kryptograficznych. Księga pomiątkowa ku czci Księdza Profesora Andrzeja Szostka MIC. Lublin: KUL, 2016. S. 489–496. URL: https://elib.belstu.by/handle/123456789/ 28915 (дата обращения: 15.05.2023).

91. Устройство обмена конфиденциальной информацией на основе нейросетевых технологий: пол. модель к патенту 7114 Республика Беларусь, МПК H 04L 9/00. / П. П. Урбанович, М. Д. Плонковски, К. В. Чуриков; заявитель и патентообладатель Белорусский государственный технологический университет. № u 20100500; заявл. 27.05.2010; опубл. 30.04.2011. URL: https://elib.belstu.by/handle/123456789/28896 (дата обращения: 20.04.2023).

92. Урбанович П. П., Долецки М. Моделирование и анализ процесса синхронизации нейронных сетей для обмена критической информацией // Комплексная защита информации. Безопасность информационных технологий: материалы XVII Международной научно-технической конференции, Суздаль, 18 мая 2012 г. Суздаль, 2012. С. 255–257. URL: https://elib.belstu.by/handle/123456789/26744 (дата обращения: 20.04.2023).

93. Secure key-exchange protocol with an absence of injective functions / R. Mislovaty [et al.] // Physical Review E. 2002. Vol. 66. Art. 066102. DOI: 10.1103/PhysRevE.66.066102.

94. Kinzel W. Theory of interacting neural networks. 2002. 21 p. DOI: org/10.1002/3527602755.ch9.

95. Kinzel W., Kanter I. Neural cryptography // Proceedings of the 9th International conference on neural information processing: ICONIP'02. Singapore, 2002. Vol. 3. P. 1351–1354. DOI: 10.1109/ICONIP.2002. 1202841.

96. Genetic attack on neural cryptography / A. Ruttor [et al.] // Physical Review E, Statistical, Nonlinear, and Soft Matter Physics. Vol. 73. P. 036121. PMID 16605612. DOI: 10.1103/Physreve.73.036121.

97. Урбанович П. П., Плонковски М., Карчмарски Д. Вероятностная оценка пространства решений для двух нейронных сетей TPM при их синхронизации // Информационные технологии в промышленности, логистике и социальной сфере: тезисы докладов XI Международной научно-технической конференции, Минск,

26–27 мая 2021 г. Минск: ОИПИ НАН Беларуси, 2021. С. 150–153. URL: https://elib.belstu.by/handle/123456789/44493 (дата обращения: 27.05.2023).

98. Урбанович П. П., Плонковски М., Карчмарски Д. Анализ сходимости весовых коэффициентов двух нейронных сетей при их синхронизации // Информационные технологии: материалы 85-й научно-технической конференции профессорско-преподавательского состава, научных сотрудников и аспирантов (с международным участием), Минск, 1–13 февр. 2021 г. Минск: БГТУ, 2021. С. 227–229. URL: https://elib.belstu.by/handle/123456789/40837 (дата обращения: 27.05.2023).

99. Urbanovich P., Karczmarski D., Płonkowski M. Probabilistic measure of space for neurocryptographic system solutions // Proceedings of 11th International conference NEET'2019, Zakopane, Poland, June 25–28, 2019. Lublin University of Techn., 2019. P. 32. URL: https://elib.belstu.by/handle/123456789/31444 (дата обращения: 06.06.2023).

100. Плонковски М., Урбанович П. П. Синхронизация криптографических ключей на основе нейронных сетей и в системах криптопреобразования на основе XML // Труды БГТУ. Сер. VI, Физ.-мат. науки и информатика. 2006. Вып. XIV. С. 152–155. URL: https://elib.belstu.by/handle/123456789/37670 (дата обращения: 06.06.2023).

101. Урбанович П. П., Чуриков К. В. Сравнительный анализ методов взаимообучения нейронных сетей в задачах обмена конфиденциальной информацией // Труды БГТУ. Сер. VI, Физ.-мат. науки и информатика. 2010. Вып. XVIII. С. 163–166. URL: https://elib.belstu.by/handle/123456789/24402 (дата обращения: 01.06.2023).

102. Dolecki M., Kozera R. Distribution of the tree parity machine synchronization time // Advances in Science and Technology. 2013. Vol. 7, no. 18. P. 20–27. DOI: 10.5604/20804075.1049490.

103. Dolecki M., Kozera R. Threshold method of detecting long-time TPM synchronization // Computer information systems and industrial management: CISIM 2013. Lecture notes in computer science, vol. 8104 / K. Saeed [et al.] (eds). Berlin, Heidelberg: Springer, 2013. P. 241–252. DOI: org/10.1007/978-3-642-40925-7_23.

104. Neural synchronization based secret key exchange over public channels: a survey / S. Chakraborty [et al.] // 2014 International conference on signal propagation and computer technology (ICSPCT 2014). July 12–13, 2014. 2015. P. 368–375.

105. Голиков В. Ф., Радюкевич М. Л. Формирование общего секрета с помощью искусственных нейронных сетей // Системный анализ и прикладная информатика. 2019. № 2. С. 49–56. DOI: org/10.21122/2309-4923-2019-2-49-56.

106. Урбанович П. П., Бирюк И. А., Плонковски М. Д. Анализ синхронизации нейронных сетей в прикладной криптографии // Информационные технологии и системы 2019 (ИТС 2019): материалы Международной научной конференции, Минск, 30 октября 2019 г. = Information Technologies and Systems 2019 (ITS 2019): Proceeding of the International Conference, Minsk, October 30, 2019 / редкол.: Л. Ю. Шилин [и др.]. Минск: БГУИР, 2019. С. 278–279. URL: https://elib.belstu.by/handle/123456789/33008 (дата обращения: 10.07.2023).

107. Радюкевич М. Л., Голиков В. Ф. Усиление секретности криптографического ключа, сформированного с помощью синхронизируемых искусственных нейронных сетей // Информатика. 2020. Т. 17, № 1. С. 102–108. DOI: org/10.37661/1816-0301-2020-17-1-102-108.

108. Sarkar A. Generative adversarial network-based efficient synchronization of group of neural networks to exchange the neural key // J. Ambient Intell. Human Comput. 2023. Vol. 14. P. 6463–6488. DOI: org/10.1007/s12652-021-03521-1.

109. Synchronization analysis of a class of neural networks with multiple time delays / C. Wang [et al.] // Journal of Mathematics. 2021. Art. ID 5573619. DOI: org/10.1155/2021/5573619.

110. Sarkar A. Gravitational search-based efficient multilayer artificial neural coordination // Neural Process Lett. 2023. Vol. 55. P. 8509–8530. DOI: org/10.1007/s11063-023-11165-9.

111. Mutual learning in a tree parity machine and its application to cryptography / M. Rosen-Zvi [et al.] // Physical review E, Statistical, Nonlinear, and Soft Matter Physics. 2002. Vol. 66 (6 Pt 2). Art. 066135. DOI: 10.1103/PhysRevE.66.066135.

112. Карчмарски Д., Плонковски М., Лисица Е. В. Методы и алгоритмы моделирования систем криптопреобразования информации на основе нейросетевых // Труды БГТУ. Сер. VI, Физ.-мат. науки и информатика. 2008. Вып. XVI. С. 137–140. URL: https://elib. belstu.by/handle/123456789/40532 (дата обращения: 12.06.2023).

113. Бирюк И. А. Моделирование и анализ процесса синхронизации искусственных нейронных сетей // Информационные технологии:

тезисы докладов 81-й научно-технической конференции профессорско-преподавательского состава, научных сотрудников и аспирантов (с международным участием), Минск, 1–12 февр. 2017 г. Минск: БГТУ, 2017. С. 10–11. URL: https://elib.belstu.by/handle/123456789/21384 (дата обращения: 12.06.2023).

114. Zubrzycki S. Wykłady z rachunku prawdopodobieństwa i statystyki matematycznej. Warszawa: PWN, 1970. 334 s.

115. Huntsberger D. V. Elements of statistical inference. Boston: Allyn and Bacon, 1967. 248 p.

116. Gardiner V., Gardiner G. Analysis of frequency distributions, concepts and techniques in modern geography. University of East Anglia, 1979. 68 p. URL: https://alexsingleton.files.wordpress.com/2014/09/19-analysis-of-frequency-distributions.pdf (date of access: 15.07.2023).

117. Greń J. Statystyka matematyczna. Modele i zadania. Warszawa: PWN, 1978. 112 s.

118. Liu P., Zeng Z., Wang J. Global synchronization of coupled fractional-order recurrent neural networks // IEEE Transactions on Neural Networks and Learning Systems. 2019. Vol. 30, no. 8. P. 2358–2368. DOI:10.1109/TNNLS.2018.2884620.

119. Sarkar A. Mutual learning-based efficient synchronization of neural networks to exchange the neural key // Complex Intell. Syst. 2022. Vol. 8. P. 307–321. DOI: org/10.1007/s40747-021-00344-7.

120. Ruttor A., Reents G., Kinzel W. Synchronization of random walks with reflecting boundaries // Journal of Physics A: Mathematical and General. 2004. Vol. 37, no. 36. P. 8609–8619. DOI: org/10.48550/arXiv.cond-mat/0405369.

121. Ruttor A., Kinzel W., Kanter I. Dynamics of neural cryptography // Physical Review E 75. 2007. Vol. 75, iss. 5. Art. 056104. URL: https://www.semanticscholar.org/reader/54e4836f9781498c702bccd9a07628c2a3d985ec (date of access: 14.10.2023). DOI: org/10.1103/PhysRevE.75.056104.

122. Cooperating attackers in neural cryptography / L. N. Shacham [et al.] // Physical Review E, Statistical, Nonlinear, and Soft Matter Physics 69. 2003. Vol. 69, iss. 6. Art. 066137. DOI: org/10.1103/ PhysRevE.69.066137

123. Martínez Padilla J., Meyer-Baese U., Foo S. Security evaluation of tree parity re-keying machine implementations utilizing side-channel emissions // EURASIP Journal on Information Security. 2018. Vol. 3. DOI: org/10.1186/s13635-018-0073-z.

124. Урбанович П. П., Плонковски М. Д., Чуриков К. В. Эффективность геометрической атаки на компьютерные сети // Автоматический контроль и автоматизация производственных процессов: материалы Международной научно-технической конференции, Минск, 28–29 окт. 2009 г. Минск, 2009. С. 40–42. URL: https://elib.belstu.by/ handle/123456789/ 25730 (дата обращения: 15.07.2023).

125. Ein-Dor L., Kanter I. Confidence in prediction by neural networks // Physical Review A, Atomic, Molecular, and Optical Physics. 1999. Vol. 60, no. 1. P. 799–802. DOI: 10.1103/physreve.60.799.

126. Seoane L. F., Ruttor A. Successful attack on permutation-parity-machine-based neural cryptography // Physical review E, Statistical, Nonlinear, and Soft Matter Physics. 2012. Vol. 85, iss. 2. Art. 025101. DOI: 10.1103/PhysRevE.85.025101.

127. Rakin A. S., He H., Fan D. Bit-flip attack: crushing neural network with progressive bit search // 2019 IEEE/CVF International conference on computer vision (ICCV), October 27, 2019, Seoul, Korea. 2019. P. 1211–1220. DOI: org/10.48550/arXiv.1903.12269.

128. Кантор И. Л., Солодовников А. С. Гиперкомплексные числа / ред. В. В. Донченко. М.: Наука, 1973. 144 с.

129. Hirose A. Applications of complex-valued neural networks to coherent optical computing using phase-sensitive detection scheme // Information Sciences – Applications. 1994. Vol. 2. P. 103–117. DOI: org/10.1016/1069-0115(94)90014-0.

130. Hirose A., Yoshida S. Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence // IEEE Transactions on Neural Networks and Learning Systems. 2012. Vol. 23, no. 4. P. 541–551. DOI: 10.1109/TNNLS.2012.2183613.

131. Bassey J., Qian L., Li X. Complex-valued neural networks: a comprehensive survey // IEEE/CAA Journal of Automatica Sinica. 2022. Vol. 9, no. 8. P. 1406–1426. DOI: 10.1109/JAS.2022.105743.

132. Theory and implementation of complex-valued neural networks / J. A. Barrachina [et al.] URL: arXiv:2302.08286v1 (date of access: 30.08.2023). DOI: org/10.48550/arXiv.2302.08286.

133. Płonkowski M. Wykorzystanie ataku geometrycznego do kryptoanalizy procesu synchronizacji architektur TPCM // Materiały XII środowiskowej konferencji matematyczno-informatycznej, Rzeszów – Lublin – Chełm – Łuck, 2–5.07.2006. Lublin, 2006. S. 34–36.

134. Урбанович П. П., Чуриков К. В. Сравнительный анализ взаимообучения двух нейронных сетей при обмене ключевой

информацией // Современные информационные компьютерные технологии mcIT-2010: материалы II Международной научно-практической конференции. Гродно, 2010. 995 с. URL: https://elib.belstu.by/handle/123456789/37340 (дата обращения: 15.08.2023).

135. Dong T., Huang T. Neural cryptography based on complex-valued neural network // IEEE Transactions on Neural Networks and Learning Systems. 2020. Vol. 31, no. 11. P. 4999–5004. DOI: 10.1109/TNNLS.2019.2955165.

136. Płonkowski M. Algebraic aspects of mutual learning of neural networks // New electrical and electronic technologies and their industrial implementation, Zakopane, Poland, June 21–24, 2005. Zakopane, 2005. P. 125–127.

137. Плонковски М. Д., Урбанович П. П. Листинг программного кода метода TPM. Минск: БГТУ, 2020. 3 с. URL: https://elib.belstu.by/handle/123456789/35340 (дата обращения: 15.08.2023).

138. Bray J. R., Curtis J. T. An ordination of upland forest communities of southern Wisconsin // Ecological Monographs. 1957. Vol. 27. P. 325–349. DOI: org/10.2307/1942268.

139. Płonkowski M., Urbanovich P. Split-complex numbers in neural cryptography // 7th International conference “New electrical and electronic technologies and their industrial implementation – NEET’2011”, Zakopane, Poland, June 28 – July 1, 2011. Lublin, 2011. P. 146. URL: https://elib.belstu.by/handle/123456789/ 25716 (дата обращения: 05.08.2023).

140. Płonkowski M., Urbanowicz P. Liczby podwójne i ich modyfikacje w neurokryptografii // Przegląd Elektrotechniczny. 2012. T. 88. Nr 11b. S. 340–341.

141. Conway J. H., Smith D. A., Peters A. K. On quaternions and octonions. Massachusetts, 2003. 159 p.

142. Greenblatt A. B., Agaian, S. S. Introducing quaternion multi-valued neural networks with numerical examples // Information Sciences. 2018. Vol. 423. P. 326–342. DOI: 10.1016/j.ins.2017.09.057.

143. Quaternion convolutional neural networks / X. Zhu [et al.] // European conference on computer vision, Munich, September 8–14, 2018. Munich, 2018. DOI: org/10.48550/arXiv.1903.00658.

144. Plonkowski M., Urbanowicz P., Lisica, E. Wykorzystanie kwaternionów w protokole uzgadniania klucza kryptograficznego, opartym na architekturach sieci neuronowych TPQM // Przeglad Elektrotechniczny. 2010. T. 86, Nr 7. S. 90–91. URL: https://elib.belstu.by/handle/123456789/31439 (дата обращения: 10.08.2023).

145. Zhang Y., Wang W., Zhang H. Neural cryptography based on quaternion-valued neural network // International Journal of Innovative Computing, Information and Control. 2022. Vol. 6, no. 22. P. 1871–1883. DOI: 10.24507/ijicic.18.06.1871.

146. Conway J. H., Smith D. A. On quaternions and octonions. New York: A K Peters/CRC Press, 2003. 172 p. DOI: org/10.1201/9781439864180.

147. Baez J. C. The octonions // Bull. Amer. Math. Soc. 2002. P. 145–205.

148. Ujang B., Took C., Mandic D. Quaternion-valued nonlinear adaptive filtering // IEEE Transactions on Neural Networks. 2011. Vol. 22, no. 8. P. 1193–1206. DOI: 10.1109/TNN.2011.2157358.

149. Deep octonion networks / J. Wu [et al.] // Neurocomputing. 2019. Vol. 397. P. 179–191. DOI: org/10.48550/arXiv.1903.08478.

150. Takahashi K., Fujita M., Hashimoto M. Remarks on octonion-valued neural networks with application to robot manipulator control // 2021 IEEE International Conference on Mechatronics (ICM), Kashiwa, Japan, March 7–9, 2021. Kashiwa, 2021. P. 1–6. DOI: 10.1109/ICM46511.2021.9385617

151. Urbanovich P. P., Shutko N. P. Usage of hypercomplex numbers in a cryptographic key agreement protocol based on neural networks // Журнал Белорусского государственного университета. Математика. Информатика. 2024. № 1 (в печати).

152. Hunt K. J., Sbarbaro D. Neural networks for nonlinear internal model control // IEE Proceedings D (Control theory and applications). 1991. Vol. 138, no. 5. P. 431–438. DOI: 10.1049/ip-d.1991.0059.

153. Secure hash function based on neural network / S. Lian [et al.] // Neurocomputing. 2006. Vol. 69, no. 16–18. P. 2346–2350. DOI: org/10.1016/j.neucom.2006.04.003.

154. Aihara K. Chaotic neural networks. 1989. URL: https://www.kurims.kyotou.ac.jp/~kyodo/kokyuroku/contents/pdf/0710-12.pdf (date of access: 20.08.2023).

155. Ерофеева В. А. Обзор теории интеллектуального анализа данных на базе нейронных сетей // Сайт СПбГУ. URL: https://math.spbu.ru/user/gran/soi11_3/p3-17.pdf (дата обращения: 20.09.2023).

156. Плонковски М. Использование нейронных сетей в операциях над хеш-функциями // Труды БГТУ. Сер. II, Физ.-мат. науки и информатика. 2007. Вып. XV. С. 169–171.

157. Płonkowski M. Analiza funkcji chaosu w funkcjach skrótu opartych na sieciach neuronowych // Przegląd Elektrotechniczny. 2008. T. 84, Nr 3. S. 102–104.

158. Анкуда А. Д. Проектирование и анализ хэш-функций на базе нейронных и ячеистых сетей // Труды БГТУ. 2013. № 6 (162): Физ.-мат. науки и информатика. С. 127–130.

159. Li Y., Deng S., Xiao D. A novel hash algorithm construction based on chaotic neural network // Neural Computing & Applications. 2011. Vol. 20. P. 133–141. DOI: org/10.1007/s00521-010-0432-2.

160. A method for designing hash function based on chaotic neural network / B. He [et al.] // International workshop on cloud computing and information security (CCIS). 2013. P. 229–233. DOI: org/10.2991/ccis-13.2013.53.

161. Abdoun N. Design, implementation and analysis of keyed hash functions based on chaotic maps and neural networks. These de Doctorat DE. Universite de Nantes. Nantes, 2019. 157 p. URL: https://theses.hal.science/tel-02271074/ (date of access: 30.08.2023).

162. Urbanovich P., Plonkowski M., Churikov K. The appearance of conflict when using the chaos function for calculating the hash code // Przegląd Elektrotechniczny. 2012. T. 88, Nr 11b. S. 346–347.

ОГЛАВЛЕНИЕ

Научное издание

Урбанович Павел Павлович
Плонковски Мартин Даниель
Долецки Михал

НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ В КРИПТОГРАФИЧЕСКИХ ПРИЛОЖЕНИЯХ

Монография

Редактор *О. П. Приходько*
Компьютерная верстка *О. А. Солодкевич*
Дизайн обложки *Д. А. Полешова*
Корректор *О. П. Приходько*

Подписано в печать 09.04.2024. Формат 60×84 1/16.
Бумага офсетная. Гарнитура Таймс. Печать цифровая.
Усл. печ. л. 13,0. Уч.-изд. л. 13,4.
Тираж 100 экз. Заказ .

Издатель и полиграфическое исполнение:
УО «Белорусский государственный технологический университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/227 от 20.03.2014.
Ул. Свердлова, 13а, 220006, г. Минск