ПРОГРАММИРОВАНИЕ, 2018, № 5, с. 43–56

ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

УДК 004.42

АДАПТАЦИЯ ОБЩИХ КОНЦЕПЦИЙ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ К НЕЙРОННЫМ СЕТЯМ *

© 2018 г. Ю.Л. Карпов[1,*], Л.Е. Карпов[2,3,], Ю.Г. Сметанин[4,5,***]**

[1]*ООО Люксофт Профешнл*
123060 Москва, 1-й Волоколамский проезд, 10
[2]*Институт системного программирования им. В.П. Иванникова РАН*
109004 Москва, ул. Солженицына, 25
[3]*Московский Государственный университет имени М.В. Ломоносова*
119991 Москва, Ленинские горы, 1
[4]*Федеральный исследовательский центр "Информатика и управление" РАН*
119333 Москва, ул. Вавилова, д. 44, кор. 2
[5]*Московский Физико-Технический институт (технический университет)*
141701 Московская область, г. Долгопрудный, Институтский пер., 9
E-mails: y.l.karpov@yandex.ru, mak@ispras.ru**, ysmetanin@rambler.ru****

Поступила в редакцию 10.05.2018

Рассмотрена задача тестирования и отладки обучающихся нейросетевых систем. Указаны отличия этих систем от программно реализованных алгоритмов с точки зрения тестирования. Определены требования к системам тестирования. Проведён анализ специфики различных нейросетевых моделей с точки зрения выбора методики тестирования и определения тестируемых параметров. Рассмотрены вопросы устранения замеченных недостатков систем. Приведён иллюстративный пример.
DOI: 10.31857/S013234740001214-0

1. ВВЕДЕНИЕ

Искусственные нейронные сети вызвали в последние годы очередной всплеск интереса исследователей, что привело к огромному потоку публикаций по этой теме и появлению значительного числа специализированных журналов, проведению многочисленных конференций и других научных мероприятий. Основные причины, которые привели к этому буму, связаны с надеждой на создание эффективных средств имитационного моделирования человеческого мозга как с целью изучения его функций, так и с попыткой реализации возможностей способности к ассоциативному обучению в задачах обработки и анализа больших объёмов сложноструктурированной информации. Нейронные сети — попытка перейти от алгоритмического подхода к эвристическому, в котором система будет "сама" решать поставленную задачу, по возможности, без вмешательства человека, который может даже не знать, что эта сеть на самом деле делает. Преимущества нейронных сетей проявляются в тех задачах, где трудно выделять правила и велика размерность объектов, зато можно выбирать много обучающих примеров.

Оживление интереса к искусственным нейронным сетям вызвано появлением глубоких нейронных сетей, отличие которых от известных ранее систем заключается в наличии значительно большего числа слоёв, каждый из которых выполняет свою задачу. Появление глубоких нейронных сетей связано с результатами исследований биологических нейронных сетей, в ходе которых обнаружено, что увеличение числа слоёв приводит к повышению эффективности решения сложных задач, а также с увеличением мощно-

*Работа выполнена при поддержке Российского фонда фундаментальных исследований, грант № 18-07-00697_а).

сти вычислительных средств, позволяющим реализовать многослойные системы.

Основное направление работ в настоящее время — создание систем глубокого обучения (deep learning), позволяющих решать такие задачи, как обработка и анализ изображений в режиме реального времени [1–3], распознавание речи [4], машинный перевод [5, а также многочисленные публикации в Интернете, преимущественно исследовательских групп Google], биоинформатика и медицина [6–9], рекомендации пользователям и потребителям [10] и т. д. Появляются работы, посвящённые реалистичному имитационному моделированию работы мозга [11–13].

К настоящему времени разработано много различных моделей нейронных сетей, довольно успешно решающих конкретные задачи. В этих моделях сети отличаются архитектурой и методами обучения. Вопрос о преимуществах и недостатках разных моделей в конкретной области применения достаточно сложен. На практике выбор производится практически наугад, на основе опыта пользователей. При построении глубоких нейронных сетей это может привести к выбору, сильно отличающемуся от наилучшего.

С точки зрения тестирования и отладки, отличие нейронных сетей от программных систем, реализующих алгоритмические методы решения задач, достаточно существенно. При программной реализации алгоритмов можно в принципе проследить движение данных и обнаружить ошибки. Нейронная сеть работает по принципу чёрного ящика, известны только пары (*вход, выход*), вывод о правильности работы надо делать именно на основании этих пар, а коррекция ошибок работы — принципиально эвристическая процедура модификации параметров сети.

2. НЕЙРОННЫЕ СЕТИ И ДЕЯТЕЛЬНОСТЬ ПО ИХ ТЕСТИРОВАНИЮ

К настоящему времени развитие теории алгоритмов решения задач привело к получению большого числа важных и интересных результатов. Разработаны и виртуальные, и вполне реальные инструменты, позволяющие по алгоритмам решать практически полезные задачи. Разработаны методы преобразования алгоритмов, позволяющие получать решения за приемлемое время, причём приемлемость времени (как и точность решения) может определяться при формулировании конкретного варианта задачи в интересах конкретной прикладной области и конкретной прикладной системы.

Особенно быстро развиваются методы решения задач интеллектуального анализа данных, задач распознавания образов, задач оптимизации, которые отличаются необходимостью обработки больших объёмов данных. При их решении часто возникают ситуации, когда необходимо исследовать и спрогнозировать поведение объектов, для которых отсутствуют формализованные модели. Даже в тех задачах, для которых известны алгоритмы, дающие хорошие результаты, при решении практических задач возникают трудности, связанные с требованиями точности решения и приемлемости времени его получения.

Трудности порождаются наличием большого числа вариантов и необходимостью их перебора. Множество задач решаются переборными методами, в которых перебор возможных вариантов либо полон, либо ограничен с той или иной степенью обоснованности. Для ограничения перебора могут использоваться разные подходы, которые, как правило, связаны с построением эвристик, то есть методов, для которых нет строгих обоснований корректности. Эвристики основаны на интуитивных соображениях, поэтому искусственные нейронные сети, которые с самого начала были направлены на моделирование работы мозга, естественно, относятся к области применения эвристических методов.

Элементы переборных методов можно увидеть и при применении нейронных сетей, для которых выписывание точных формульных зависимостей выходных сигналов от входных практически никогда не производится. Нейросетевые методы решения задач анализа данных, распознавания и комбинаторной оптимизации относятся к классу эвристических. Многослойные нейронные сети с прямым распространением информации достаточно давно используются в решении задач распознавания образов. Для повышения эффективности таких сетей разработано много вспомогательных процедур. Эти процедуры особенно важны в глубоких нейронных сетях. Рекуррентные (однослойные) нейронные сети используются в качестве ассоциативной памяти. В глубоком

обучении их можно использовать для инициализации данных.

Несмотря на отступление от строгих математических постулатов, эвристические методы показали прекрасные результаты, с успехом используемые в реальных промышленных системах.

И наиболее известным эвристическим подходом является такая человеческая деятельность, как *тестирование*. Эвристика тестирования определяется прежде всего его конечностью. Возможное количество ситуаций, возникающих при тестировании, ограничивается временем и ресурсами, которыми обладают тестировщики, оценками достигаемого качества тестирования.

В силу естественной ограниченности тестирование никогда не даёт гарантий правильности построения тестируемой установки. В особенности строго это указывается в стандартах, описывающих деятельность по тестированию программного обеспечения [14–18]. Являясь частью более общего процесса по верификации программного обеспечения, технический процесс тестирования, как и процесс реализации программного средства имеет целью демонстрацию того, что программный продукт удовлетворяет спецификациям и готов для применения в заданном окружении или интеграции с системой, для которой он предназначен, способствуя тем самым достижению результатов процесса верификации. Чтобы тестирование стало возможным, необходимо заранее создать тестовые примеры и программные данные, а также разработать процедуры тестирования (способы выполнения одного или нескольких тестовых вариантов и их составных элементов — отдельных шагов и проверок), уточнив условия проведения тестирования.

Сам по себе процесс тестирования заключается в выборе модели тестирования, в последовательной отработке тестовых сценариев с привлечением заранее заготовленных тестовых данных и в последующем сравнении полученных результатов с заранее сформированными эталонами. Эталоны составляются для каждого отдельного сценария и каждого отдельного набора входных данных путём применения наиболее пригодных и наиболее теоретически выверенных средств. Иногда для этого создаются специальные программные инструменты [19–26], которые автоматизируют и процесс подготовки сценариев (специально созданных ситуаций) и данных, и собственно тестирование, и оценку полноты проведенного тестирования, и последующий анализ результатов тестирования, то есть сличения полученных выходных данных с эталонами. Некоторые разработки с самого начала строятся на основе тестирования (test-driven development, [27–28]). При этом тесты пишутся самими разработчиками, а реализация ведётся так, чтобы тестирование дало положительный ответ о готовности программных процедур, модулей, компонентов.

В этой последовательности действий наибольший интерес представляет процесс разработки сценариев тестирования, как наиболее творческий и требующий от его исполнителей максимального напряжения сил. Крайне редко удаётся подготовить исчерпывающий набор сценариев, охватывающий все возможные комбинации входных данных и системных реакций на них. Гораздо чаще приходится сталкиваться с ситуациями, в которых необходимо заранее классифицировать данные, выявлять наиболее вероятные комбинации параметров, выбирать и сравнивать критерии проведения тестирования. Вот тут и проявляется творческий характер деятельности по тестированию. При этом, чем сложнее объект тестирования, тем больше опыта и умения приходится вкладывать в его тестирование. В этом отношении нейронные сети представляют собой уникальные структуры, которые создают огромное поле деятельности для разработчика сценариев.

3. ЭВРИСТИКА В СИСТЕМАХ НЕЙРОННЫХ СЕТЕЙ

Нейронная сеть является совокупностью некоторого конечного количества взаимодействующих нейронов (процессорных элементов) N^1, N^2,..., N^n. Формальный нейрон N^i — упорядоченная тройка $(w^i = (w_{i1} \ldots, w_{in}); w_{i0}; F_i)$, где w^i — вектор весов синапсов нейрона (связей, по которым поступает информация от всех нейронов в N^i), F — функция активации нейрона (решающее правило, определяющее зависимость выхода нейрона от его входов), w_{i0} — порог

нейрона (нейрон активизируется, если значение функции активации не меньше порога).

На вход нейрона N^i поступают сигналы $v_1, \ldots v_n$, после чего нейрон вычисляет своё новое значение $v = F(w_{i0}, w_{i1}, \ldots, w_{in}, v_1, \ldots, v_n)$. Наиболее распространённая функция активации —

$$F = F\left(\sum_{j=1}^{n} w_{ij} v_j - w_0\right).$$

Как правило, если не для всех нейронов в сети, то по крайней мере для больших групп нейронов используется одна и та же функция активации. В качестве важных частных случаев функции активации используются такие варианты:

1. Пороговые правила, $F(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}$ (нейрон МакКаллоха–Питтса);
2. "Сигмоидальные" функции (сигмоиды) $F(x) = \sigma(x)$[1]; примером которых является экспоненциальная функция активации $F(x) = (1 + \exp(-x))^{-1}$;
3. Линейные функции.

В первом случае нейроны являются двоичными и принимают значения $(+1, -1)$ ("биполярные" нейроны)[2], во втором нейроны принимают значения из непрерывного интервала.

Линейные нейроны, как правило, используются в неоднородных моделях с элементами разных типов.

Нейронная сеть работает как единый набор нейронов, в котором выходы одних нейронов поступают на входы других, соединённых с ними синапсами, после чего происходит пересчёт состояний по описанным выше правилам.

Информация может поступать также из внешнего мира. Формально её можно описать дополнительными "входными" нейронами, значения которых не зависят от значений остальных нейронов.

В конкретных применениях размерность (количество входных и выходных нейронов) и тип входных и выходных данных определяются спецификой задачи и конкретной системой, в рамках которой задача решается. Опираясь на эти исходные данные и руководствуясь опытом предшествующих разработок, создатели нейросетевых систем выбирают архитектуру нейронной сети, то есть

- количество и типы нейронов,
- их разбиение на слои нейронов,
- виды слоев нейронов,
- способы взаимодействия между нейронами,
- веса синапсов,
- функции активации и
- другие характеристики.

Все эти параметры являются объектами тестирования. Поясним этот посыл на примере.

Перед началом основной работы сети нейронов с входной информацией может выполняться вспомогательный, но для некоторых задач обязательный процесс обучения с учителем, который заключается в построении матрицы весов связей элементов сети $W = W(w_{ij})$, при которой заданный набор входных векторов $\overline{X}$ (обучающая выборка) преобразуется в заданный набор соответствующих выходных векторов $\overline{Y}$:

$$\overline{Y} = W\overline{X}.$$

Обучение проводится подбором элементов w_{ij}, причём начальные значения этих элементов часто выбираются случайным образом.

Непосредственно при обучении подбором элементов стараются найти такую матрицу W, которая каждый начальный вектор $\overline{X^i}$ будет преобразовывать в заданный выходной вектор $\overline{Y^j}$:

$$\overline{Y^j} = W\overline{X^i}.$$

[1] Функция a(x) называется сигмоидальной, если она непрерывно дифференцируема, монотонно возрастает и ограничена сверху и снизу.

[2] В двоичных сетях иногда используют значения нейронов $(1, 0)$. В этом случае функция активации — хевисайдовская,

$$\mathcal{Q}(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}.$$

Переход от таких значений к стандартным значениям $(1, -1)$ тривиален: $y = 2x - 1$.

Иногда из обучающей выборки выделяют часть, называемую тестовой, на которой проверяют правильность работы обученной нейронной сети. При этом, однако, следует учитывать, что исключение из процесса обучения части образцов может приводить к снижению точности обучения.

Чтобы проиллюстрировать особенности тестирования архитектуры нейронной сети, рассмотрим вкратце основные модели нейронных сетей.

Разбиение нейросетевых моделей на классы может производиться по разным признакам; наиболее важны с точки зрения организации тестирования дихотомии нейросетевых моделей такие:

- по рекуррентному или нерекуррентному режиму работы;
- по детерминированному или вероятностному режиму работы;
- по обучению с учителем и без учителя (так называемые, смешанные стратегии).

Возможны также дихотомии по области значений параметров (цифровые и аналоговые), по времени (дискретное и непрерывное), по режиму обновления параметров (с синхронизацией и без нее), но с точки зрения тестирования различия по этим критериям менее важны.

В нерекуррентных нейронных сетях (с прямым распространением информации) каждый нейрон задействован только единожды: после получения входов он передаёт свой выход нейронам, которые ещё не участвовали в работе сети, и сам больше не участвует в работе. Таким образом, сеть можно разбить на слои, работающие последовательно. Известна базовая модель этого типа — это многослойный персептрон [29–31].

В рекуррентных нейронных сетях допускается наличие обратных связей, и некоторые нейроны сети могут включаться в работу неоднократно. Окончанием работы (получением решения или отказом) можно считать состояние, в котором все нейроны перестают изменять свои состояния после поступления входов. Самый известный пример такой нейронной сети — сеть Хопфилда [32].

Обе упомянутые модели — детерминированные и обучаемые с учителем.

В вероятностных нейронных сетях обновление параметров происходит с использованием случайных параметров. Классический пример вероятностной сети — машина Больцмана [33]. Изначально она была использована в сети Хопфилда для обхода локальных оптимумов, сильно отличающихся от глобального. В глубоком обучении применяется ограниченный вариант машины Больцмана.

В обучении без учителя нет обучающих пар (*вход, выход*), есть только набор входных векторов, которые нейронная сеть разбивает на кластеры по некоторым признакам сходства. Классический пример такого подхода — самоорганизующаяся карта Кохонена [34].

Глубокие нейронные сети основаны на сочетании различных моделей.

Впервые методы, которые можно считать прообразами глубокого обучения (без использования этого термина), появились в работах [35–36]. Другой пример успешного применения методов обучения, которые впоследствии стали общепринятыми в глубоких нейронных сетях — неокогнитрон [37]. В [38] метод обратного распространения ошибок был применён в машинном зрении для распознавания ZIP-кодов с помощью нейронной сети со многими слоями и дал хорошие результаты, но тогда для обучения ему потребовалось три дня. Впервые современные методы глубокого обучения описаны в [39].

В 2007 было впервые показано, что нейронную сеть со многими слоями и прямым (без обратных связей) распространением информации можно эффективно обучать слой за слоем, причем каждый слой является обучаемой без учителя вероятностной системой — ограниченной машиной Больцмана; после этого производится уточнение обучения методом обратного распространения (то есть с учителем) [40].

Далее рассмотрены некоторые нейросетевые модели, которые можно считать в определенном смысле базовыми и на которых будет удобно иллюстрировать предлагаемый эвристический подход. Для каждой из моделей, описанных в этом разделе, приведено их краткое описание, показаны принципы определения тестируемых параметров и построения тестовой выборки с учетом этих параметров.

3.1. *Многослойный перцептрон*

Простейший пример многослойной нейронной сети, работающей в нерекуррентном режиме — многослойный персептрон. Граф связей между нейронами в этой модели имеет иерархическую структуру, в которой связаны только соседние уровни, а сигналы передаются только в одну сторону. Поступающие входные сигналы последовательно обрабатываются слоями сети, начиная от входного и кончая выходным. Все остальные (промежуточные) слои называются скрытыми. Обучение с учителем основано на предъявлении пар входных и выходных векторов, соответствия между которыми должна реализовать нейронная сеть. Активное развитие таких сетей началось после создания метода обучения на основе обратного распространения ошибок [41].

Идея метода заключается в присвоении произвольных начальных значений весам сети, подсчёте выхода при обучающем входном образце и последовательном определении производных отклонения этого выхода от желаемого по весам, начиная от выходного слоя. При экспоненциальной функции активации зависимость этих производных от самой функции очень проста, и вычисление их значений не требует сложных операций.

Для иллюстрации опишем случай нейронной сети с одним скрытым слоем и с экспоненциальной функцией активации. Обобщение на случай большего числа слоев очевидно.

Пусть $X = (x_i)$ — входной вектор, $i = 1, 2, \ldots, I$, $Z = (z_j)$ — значения скрытых нейронов (принадлежащих промежуточному слою), $j = 1, 2, \ldots, J$, $Y = (y_k)$ — значения выходных нейронов, $k = 1, 2, \ldots, K$, w_{ij} — веса синапсов от входного слоя к скрытому, w'_{jk} — веса синапсов от скрытого слоя к выходному, $T = (t_k)$ — эталон выхода, который хотелось бы получить при входном векторе X, $E_k = \frac{1}{2}\sum_k (t_k - y_k)^2$ — минимизируемое в ходе обучения отклонение истинного выхода от эталона.

Алгоритм обратного распространения ошибок выполняется по следующим шагам:

Шаг 1. Выбрать случайные значения всех весов;

Шаг 2. Выбрать обучающую пару (вход X, выход T);

Шаг 3. Вычислить

$$net_j = \sum_i w'_{ij} x_j;$$

Шаг 4. Вычислить $z_j = \frac{1}{1+\exp(-net_j)}$;

Шаг 5. Вычислить

$$net_k = \sum_j w_{jk} z_j;$$

Шаг 6. Вычислить $y_k = \frac{1}{1+\exp(-net_k)}$;

легко видеть, что $\frac{\partial E_k}{\partial w_{jk}} = y_k(t_k - y_k)(1 - y_k)z_j$;

Шаг 7. Вычислить $\delta_k = y_k(t_k - y_k)(1 - y_k)$;

Шаг 8. Вычислить

$$\delta_j = z_j(1 - y_k)\sum_k \delta_k w_{jk};$$

Шаг 9. Модифицировать веса по правилу

$$w_{jk}(new) = w_{jk}(old) + \eta\delta_k z_j,$$
$$w'_{jk}(new) = w'_{jk}(old) + \eta\delta_j x_i,$$

(параметр η выбирается пользователем, он одинаков для всех весов);

Шаг 10. Вернуться к шагу 2 и продолжить работу до получения приемлемого результата.

Метод обратного распространения ошибок может давать хорошие результаты, но у него есть один существенный недостаток. Как и все градиентные методы, он в принципе приводит к локальному минимуму ошибки, который может сильно отличаться от глобального минимума (minimum minimorum).

Даже отвлекаясь от необходимости/возможности варьировать число слоев нейронной сети, многослойный перцептрон содержит огромное количество параметров сети, тестирование которых и последующая модификация которых может привести к существенному повышению качества решения основной задачи, для которой многослойный перцептрон проектировался.

Прежде всего, к параметрам, влияние которых на результирующую сеть следует тестировать, относятся начальные значения коэффициентов матриц весов синапсов (в показанном примере это матрицы w и w'). От этих значений зависит многое — и скорость сходимости алгоритма обратного распространения ошибок, и (главное) близость найденного минимума ошибок к глобальному минимуму.

Тестирование может также помочь правильному выбору параметра η.

3.2. Нейронная сеть Хопфилда

Нейронной сетью Хопфилда называется однослойная рекуррентная сеть с симметричными синапсами. Каждый нейрон получает информацию о состояниях всех связанных с ним нейронов и обновляет свое состояние. Этот процесс продолжается до тех пор, пока сеть не достигнет стационарного состояния, в котором состояния нейронов перестают изменяться. Если пороги равны нулю (нет непосредственной связи нейрона с самим собой), а обновление нейронов происходит асинхронно, это состояние будет достигнуто; известны оценки скорости сходимости [42]. Доказательство основано на том, что для любой сети такого вида существует энергетическая функция вида

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}v_iv_j - \sum_{i=1}^{n} b_iv_i,$$

уменьшающаяся на каждом шаге работы сети; здесь v_i — значения нейронов, b_i — их пороги, w_{ij} — веса синапсов.

Сеть Хопфилда может использоваться в качестве ассоциативной памяти: считается, что любой входной вектор является версией (возможно, искажённой) одного из стационарных состояний.

Чтобы использование сети Хопфилда было оправданно, метод обучения должен быть простым. Одни из самых распространённых методов обучения — метод внешних произведений, а также некоторые его модификации. Метод основан на построении матрицы весов W по следующим правилам:

1. Матрица W представляется в виде суммы матриц, каждая из которых соответствует одному из обучающих примеров $\overline{X^m}$ ($1 \leq \leq m \leq n$, где n — число обучающих примеров): $W = W^1 + W^2 \cdots + W^n$.

2. Все элементы на главных диагоналях всех матриц W^m равны 0.

3. Остальные элементы каждой из матриц W^m вычисляются так: $w_{ij}^m = x_i^m x_j^m$ ($1 \leq i,\ j \leq p$, где p — размерность входного вектора $\overline{X^m}$).

В нейронных сетях с дискретными значениями нейронов $(+1, -1)$ правила вычисления новых значений имеют вид $y_i = \mathrm{sgn}(\sum_{j=1}^{n} w_{ij}x_j)$.

Легко убедиться, что для обучающих примеров $\overline{X^m}$ в сделанных определениях знак y_i^m всегда будет совпадать со знаком x_i^m.

Подача на вход такой сети произвольного вектора, не входящего в обучающий набор, может привести к возникновению одной из трёх ситуаций, из которых первая и вторая могут возникать как в сетях с прямым распространением (после прохождения всей сети), так и в рекуррентных сетях (после попадания за конечное число шагов в состояние равновесия), а третья — только в рекуррентных, причем с синхронизированным обновлением параметров:

1. На выходе сети будет получено значение, в точности совпадающее с одним из откликов, встречавшихся при обучении. Распознавание можно считать успешным.

2. На выходе сети будет получено значение, не совпадающее ни с одним из обучающих выходных образцов. Такое решение необходимо считать ложным.

3. При повторении шагов сеть не достигает состояния равновесия, а возникает цикл повторяющихся выходных значений. Длина цикла в симметричной нейронной сети без петель не может быть больше 2 [43]. Выявление циклов не представляет собой алгоритмически сложную задачу, однако оно требует достаточно много памяти: для каждого входного вектора необходимо хранить результаты всех шагов его преобразования на основе матрицы весов связей. Если некоторый шаг преобразования приводит к тому, что выходной вектор совпадёт с одним из запомненных ранее его значений, кроме предпоследнего, а это предпоследнее значение будет отличаться от последнего, это и будет означать обнаружение цикла. Необходимо будет констатировать, что построенный распознаватель некорректен и требует модификации. Нейронные сети, в которых требовалось запомнить именно циклы, не получили широкого распространения.

Чтобы найти наилучшую для конкретного применения сеть Хопфилда, необходимо провести тестирование различных вариантов

значений сразу нескольких её параметров. Начальные значения коэффициентов матрицы весов, векторы порогов срабатывания нейронов, метрики, позволяющие сравнивать выходы сети с обучающими выходными образцами, вид сигмоидальной функции, используемой при формировании выходных векторов на основе матрицы весов и значений входных векторов — все эти характеристики могут и должны быть протестированы перед началом реального использования сети для решения прикладной задачи.

3.3. Стохастические методы обучения

Стохастические методы обучения применяются как для обучения искусственных нейронных сетей, так и для возмущения выходов от уже обученных сетей с целью исключения локальных минимумов энергии, соответствующих ложным состояниям равновесия. Основная идея этих методов заключается в псевдослучайном изменении величин весов и сохранении тех изменений, которые ведут к улучшению работы нейронной сети.

Наиболее широко известный стохастический алгоритм обучения, называемый, в соответствии с отмечаемой физической аналогией, больцмановским обучением, состоит из следующих шагов:

Шаг 1. Определить переменную T, представляющую собой искусственную температуру, присвоить T большое начальное значение.

Шаг 2. Предъявить сети вектор из обучающего множества, вычислить соответствующий ему выход сети и значение целевой функции.

Шаг 3. Дать случайное изменение весу синапса "вход–выход", пересчитать выход сети и изменение целевой функции, соответствующее произведенной коррекции веса.

Шаг 4. Если изменение веса, привело к уменьшению целевой функции, то сохранить его, если же имеет место увеличение целевой функции, то вычислить вероятность сохранения изменения в соответствии с распределением Больцмана $P(e) = \exp(-e/kT)$, где $P(e)$ — вероятность изменения значения коэффициента e в целевой функции; k — константа, аналогичная константе Больцмана, выбираемая в соответствии с конкретной задачей; T — управляющий положительный параметр, называемый (искусственной) температурой.

Из равномерного распределения на интервале $(0;1)$ выбирается случайное число r. Если $P(e) > r$, то изменение сохраняется, в противном случае величина веса возвращается к предыдущему значению.

Эта схема дает системе возможность делать случайный шаг в направлении, увеличивающем штрафную функцию, позволяя ей тем самым вырываться из ловушек локальных минимумов.

Указанная последовательность операций выполняется для каждого из весов сети и сопровождается постепенным уменьшением искусственной температуры до тех пор, пока не будет достигнуто допустимо низкое значение целевой функции. Далее сети предъявляется следующий эталонный набор, и процесс обучения повторяется. Таким образом, сеть обучается на всех эталонах, с возможным повторением, пока целевая функция не станет допустимой для всех них. Величина случайного изменения веса может определяться различными способами.

На каждом шаге этого процесса открываются собственные возможности для использования эвристик и проверки их эффективности тестированием вариантов.

Во-первых, на шаге 1 можно целенаправленно подбирать начальное значение искусственной температуры T. Выбор начального значения может существенно повлиять на время решения основной задачи, поставленной перед нейронной сетью. Если выбранное значение оказывается недостаточно большим, алгоритм может отдавать "плохим" решениям предпочтение перед гораздо более "правильными".

Во-вторых, на шаге 2 различными способами можно выделять то подмножество обучающих векторов, которое будет использовано при настройке сети на решение поставленной задачи. Одним из возможных (явно не всегда) вариантов является полный перебор всех вариантов обучения, однако такой перебор может в некоторых ситуациях оказаться слишком затратным по времени.

В-третьих, выбор метода изменения весов и целевых функций, выполняемый на шаге 3, непо-

средственно и одновременно влияет на скорость сходимости процесса (количество шагов, которое выполняется сетью) и качество решения (близость найденного локального минимума целевой функции к ее глобальному минимуму).

Наконец, на шаге 4 решающее значение может оказывать способ определения необходимости коррекции веса исследуемого синапса. Вариации этого способа, определяемые при тестировании эвристическими методами, могут помочь поиску действительно подходящих для конкретных условий задачи параметров нейронной сети. В глубоком обучении часто используется ограниченный вариант машины Больцмана, структура связей между нейронами в которой описана не полным, а двудольным графом.

3.4. Самоорганизующиеся нейронные сети

Самоорганизация — самый распространённый вид обучения нейронной сети без учителя. При этом на стадии обучения сети предъявляется исходная совокупность объектов без информации о принадлежности объекта к какому-либо классу. Эта информация заменяется набором правил, в соответствии с которыми сеть вырабатывает метод разбиения набора имеющегося набора входных векторов на классы. В соответствии с определёнными правилами предъявленный набор объектов разбивается на классы, каждый из которых включает сходные векторы. Наиболее известный метод самоорганизации — карта самоорганизации Кохонена. Она реализуется на рекуррентной структуре, поэтому её тестирование можно считать сходным с тестированием модели Хопфилда.

3.5. Глубокое обучение

Глубокое обучение, появившееся в последние годы, основано на заимствованиях из результатов исследования процессов анализа информации в биологических нейронных сетях [44–47]. В исследованиях биологических систем зрения было обнаружено, что увеличение числа слоёв в многослойной сети повышает точность анализа сложных входных данных. Повышение вычислительных возможностей компьютерных систем сделало возможным реализовать сети с очень большим числом слоёв. Например, в системе ResNet удалось реализовать 1001 слой [48].

Отличительной чертой глубокого обучения является то, что оно направлено на обучение представлению данных, тогда как все описанные ранее подходы предполагают, что признаки, описывающие входные данные, заданы ещё до начала обучения. Глубокое обучение может проводиться с учителем, без учителя, оно также может иметь смешанный характер. Глубокая нейронная сеть состоит из каскада многих слоёв, использует нелинейную обработку для выделения новых признаков с возрастающим по мере перехода от слоя к слою уровнем абстракции. Таким образом, формируется иерархия концепций, высшие уровни которой лучше соответствуют специфике решаемой задачи анализа данных и распознавания.

Основные вызовы в области глубоких нейронных сетей — возникновение “переобучения” и относительно долгое время обучения. Под переобучением понимается чрезмерно точное обучение на заданной выборке, когда в обучающей выборке обнаруживаются некоторые случайные закономерности, что приводит к сложному описанию данных и плохой способности к обобщению. Оно может возникать, если при обучении принимаются во внимание посторонние признаки обучающих данных, не имеющие прямого отношения к рассматриваемой задаче. Из-за этого по мере обучения анализ обучающей выборки уточняется, но за её пределами может становиться менее точным. Для устранения переобучения, как правило, используется регуляризация. Разработано и успешно применяется несколько методов регуляризации [49–52 и многочисленные методы построения дополнительных обучающих образцов на основе изменения имеющихся], включая многочисленные методы построения дополнительных обучающих образцов на основе изменения имеющихся. Для решения проблемы вычислительной сложности используются мощные вычислительные устройства, в том числе с графическими процессорами. Для выбора начальных параметров при обучении слоёв нейронной сети эффективной оказывается ограниченная машина Больцмана [53], число слоёв в которой может быть больше двух.

4. ПРИЛОЖЕНИЕ МЕТОДОВ ТЕСТИРОВАНИЯ К НЕЙРОННЫМ СЕТЯМ

Эвристический подход к моделированию структур нейронных сетей также не обязательно будет приводить к получению оптимальных структур, к идеальным нейронным сетям, как не приводит он в обычном своём применении к идеальному программному обеспечению, свободному от всех возможных ошибок. Однако его использование даст уверенность в том, что на некотором пуле задач при некоторых входных данных (построенных не самым идеальным образом) сеть будет давать вполне приемлемые результаты.

В отличие от тестирования программного обеспечения, тестирование нейронных сетей лишь в малой степени связано с проверкой выполненной программной реализации выбранной абстрактной модели. Такую проверку тоже необходимо проводить, но она не должна занимать чрезмерно много времени, к тому же, иногда её можно выполнить на самых упрощённых модельных примерах. В частности, почти всегда можно ограничиться небольшим количеством входных векторов, малыми размерами матриц весов связей. При этих проверках совсем не требуется проводить слишком глубокое обучение сетей. Все это должно способствовать минимизации затрат на построение тестовых сценариев, имеющих в виду проверку принятых программных решений.

Совсем иначе обстоит дело с тестированием правильных программных моделей, которое должно проводиться с целью выбора наиболее подходящих для решения некоторого класса задач сетевых структур.

Как уже было показано, выбору подлежат самые разные параметры сетей: количество обучающих примеров, их разнообразие, степень их влияния на формируемые выходные векторы, количество слоев нейронной сети, необходимое для решения выбранного набора задач, метрики близости входных и выходных векторов, используемые на разных слоях сети, алгоритмы построения матриц весов связей элементов сети и другие ее структурные характеристики.

Технологии тестирования достигли в настоящее время такого развития, что выбор технологии, наиболее подходящей для тестирования нейронных сетей, также становится трудной задачей. Обычно для построения тестов используются следующие варианты методов [54]:

- вероятностные методы (вероятностная генерация тестовых воздействий в соответствии с определёнными распределениями), полнота тестирования которых оказывается показателем случайным и непредсказуемым;
- методы, нацеленные на полное покрытие, требующие полной информации о внутреннем устройстве системе и больших ресурсов для систематического тестирования;
- комбинаторные методы, связанные с группированием входных воздействий, занимающие промежуточное положение между первыми двумя подходами, не требующие исчерпывающих знаний о тестируемых системах;
- автоматные методы, с помощью которых удаётся добиваться достаточно высоких показателей полноты тестирования, но требующих полной информации о системе;
- алгебраические методы, которые редко находят себе применение из-за жёсткой необходимости описать тестируемую систему в виде набора абстрактных типов данных с полным набором аксиом.

Ни один из перечисленных подходов, по-видимому, не может быть применён к тестированию нейронных сетей в “чистом” виде. Наиболее продуктивным представляется подход, организованный на базе моделирования нейронной сети в виде обобщённого иерархического недетерминированного автомата, допускающего спонтанные изменения своего состояния.

5. ПРИМЕР ПРИМЕНЕНИЯ МЕТОДОВ ТЕСТИРОВАНИЯ К НЕЙРОННОЙ СЕТИ ПРИ РЕШЕНИИ ПРОСТОЙ ЗАДАЧИ

Здесь рассмотрен условный пример рекуррентной сети Хопфилда, который иллюстрирует основные положения предлагаемого подхода к тестированию.

Пусть мы хотим построить ассоциативную память, запоминающую заданную выборку двоичных векторов-образцов определённой размерности n и трактующую любой входной вектор размерности n как вектор из этой выборки, возможно, с искажениями типа замены символов. Для решения этой задачи будем использовать нейронную сеть Хопфилда. Выбор этой сети для иллюстрации объясняется тем, что при всей простоте модели определение вида областей притяжения для положений равновесия в сети — трудная задача [55].

Построение точной матрицы весов связей, которая не допускала бы возникновения циклов и ложных решений ни при каких значениях входных векторов, требует очень большого числа вычислительных операций, поэтому на практике используются эвристические подходы, эффективные с вычислительной точки зрения. Приведём простейший (и необязательно успешный) пример подхода к тестированию нейронной сети.

Пусть известно ложное решение — вектор z. Построим по правилу внешнего произведения матрицу $W^z : w^z_{ij} = z_i z_j$. Если бы вектор y был требуемым решением, матрица W^y вошла бы в W как одно из слагаемых. Если бы метод внешнего произведения работал абсолютно правильно, то по набору векторов x была бы построена матрица W, которая после добавления вектора z была бы заменена матрицей $W + W^z$. После вычитания из этой матрицы W^z в сети вновь останутся только нужные решения.

Авторам представляется полезной такая модификация исходной матрицы весов:

$$W' = W - \alpha_{z1}W^{z1} - \cdots - \alpha_{zi}W^{zi}\ldots,$$
$$\text{где все } |\alpha_{zi}| < 1.$$

Вычитаемые матрицы соответствуют посторонним решениям (ложным решениям и циклам), а значения соответствующих параметров α_{zi} приходится подбирать эмпирически, используя модифицированную матрицу весов и проверяя, возникают ли при таком использовании циклы и/или ложные решения, препятствующие успешному распознаванию тестовых примеров.

Чтобы сделать пример наглядным, мы рассматриваем пример очень малой размерности, который, естественно, не имеет практического смысла, но позволяет проследить возникновения посторонних решений и возможности их устранения.

Пусть в качестве входных векторов используются последовательности чисел 1 и -1 длиной 3. В качестве обучающего вектора выберем последовательность:

$$\overline{X^6} = (1, -1, 1).$$

Матрица весов связей, строящаяся по формулам $w^m_{ij} = x^m_i\ x^m_j,\ w^m_{ii} = 0 (1 \le m \le 3)$, будет такой:

$$W = \begin{pmatrix} 0 & -1 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix}.$$

Применим преобразование с этой матрицей ко всем возможным входным векторам $\overline{X^i} (1 \le i \le \le 8)$, которые могут быть только такими, как это показано в таблице 1. Верными решениями будем считать только те выходные векторы, которые совпадают во всех своих элементах с обучающими векторами. Совпадение не обязательно означает тождества. Для определения совпадения следует использовать подходящую для этого метрику. Остальные решения будем рассматривать как ложные.

Как видно из таблицы 1, в двух случаях из 8 распознавание не может дать никакого определённого ответа ни за какое число шагов, так как на выходе сети значения выходного вектора циклически повторяются. Остальные 6 входных векторов дают, пусть за разное количество шагов, чёткий ответ, один из которых построен на ложном решении, а другие — решение, на получение которого было нацелено обучение. Выходов из сложившейся ситуации может быть несколько. Один из них — продолжить обучение, проведя более точный выбор матрицы весов связей на основе более длительной предварительной подготовки сети к основной работе в качестве распознавателя.

Однако можно модифицировать матрицу весов, немного изменив её недиагональные элементы. При модификации необходимо найти возможность избавиться от ложного решения (z) и циклов (c_1, c_2), для которых строятся такие матрицы весов связей:

Таблица 1

i	$\overline{X^i}$	$\overline{Z_1^i} = W \cdot \overline{X^i}$	$\overline{Y_1^i} = \mathrm{sgn}(\overline{Z_1^i})$	$\overline{Z_2^i} = W \cdot \overline{Y_1^i}$	$\overline{Y_2^i} = \mathrm{sgn}(\overline{Z_2^i})$	Комментарий
1	$(-1,-1,-1)$	$(0,2,0)$	$(1,1,1)$	$(0,-2,0)$	$(1,-1,1)$	
2	$(-1,-1,1)$	$(2,0,0)$	$(1,1,1)$	$(0,-2,0)$	$(1,-1,1)$	
3	$(-1,1,-1)$	$(-2,2,-2)$	$(-1,1,-1)$	$(-2,2,-2)$	$(-1,1,-1)$	
4	$(-1,1,1)$	$(0,0,-2)$	$(1,1,-1)$	$(-2,0,0)$	$(-1,1,1)$	
5	$(1,-1,-1)$	$(0,0,2)$	$(1,1,1)$	$(0,-2,0)$	$(1,-1,1)$	
6	$(1,-1,1)$	$(2,-2,2)$	$(1,-1,1)$	$(2,-2,2)$	$(1,-1,1)$	
7	$(1,1,-1)$	$(-2,0,0)$	$(-1,1,1)$	$(0,0,-2)$	$(1,1,-1)$	
8	$(1,1,1)$	$(0,-2,0)$	$(1,-1,1)$	$(2,-2,2)$	$(1,-1,1)$	

i	$\overline{Z_3^i} = W \cdot \overline{Y_2^i}$	$\overline{Y_3^i} = \mathrm{sgn}(\overline{Z_3^i})$	$\overline{Z_4^i} = W \cdot \overline{Y_3^i}$	$\overline{Y_4^i} = \mathrm{sgn}(\overline{Z_4^i})$	Комментарий
1	$(2,-2,2)$	$(1,-1,1)$	$(2,-2,2)$	$(1,-1,1)$	Решение
2	$(2,-2,2)$	$(1,-1,1)$	$(2,-2,2)$	$(1,-1,1)$	Решение
3	$(-2,2,-2)$	$(-1,1,-1)$	$(-2,2,-2)$	$(-1,1,-1)$	Ложное решение
4	$(0,0,-2)$	$(1,1,-1)$	$(-2,0,0)$	$(-1,1,1)$	Цикл
5	$(2,-2,2)$	$(1,-1,1)$	$(2,-2,2)$	$(1,-1,1)$	Решение
6	$(2,-2,2)$	$(1,-1,1)$	$(2,-2,2)$	$(1,-1,1)$	Решение
7	$(-2,0,0)$	$(-1,1,1)$	$(0,0,-2)$	$(1,1,-1)$	Цикл
8	$(2,-2,2)$	$(1,-1,1)$	$(2,-2,2)$	$(1,-1,1)$	Решение

$$W^z = \begin{pmatrix} 0 & -1 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix},$$

$$W^{c1} = \begin{pmatrix} 0 & -1 & -1 \\ -1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix},$$

$$W^{c2} = \begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix}.$$

Именно здесь и могут оказаться полезными подходы и методы, выработанные для такой области деятельности, как тестирование программного обеспечения. Алгебраический подход, использованный для моделирования нейронных сетей (включая рекуррентные и многослойные сети, решение задач в которых основано на глубоком обучении), не может скрыть того факта, что рядом с математической моделью практически всегда имеется модель программная. Методы тестирования и громадный научно-методический задел, аккумулированный в том числе в большом числе международных стандартов, разработанных для применения при проведении тестирования, должны по нашему мнению дать новый толчок развитию идей, положенных в основу нейронных сетей.

СПИСОК ЛИТЕРАТУРЫ

1. *Ciresan, D., Meier, U., Masci, J., and Schmidhuber, J.* (August 2012). Multi-column deep neural network for traffic sign classification. Neural Networks. Selected Papers from IJCNN 2011, vol. 32, pp. 333–338.
2. CES 2015: Nvidia Demos a Car Computer Trained with “Deep Learning”, A commercial device uses powerful image and information processing to let cars interpret 360° camera views, David Talbot, January 6, 2015, MIT Technology Review; Schmidt.
3. *Roth, S.* Shrinkage Fields for Effective Image Restoration (PDF). IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.
4. *Deng, L. and Yu, D.* Deep Learning: Methods and Applications, Foundations and Trends in Signal Processing, 2014, vol. 7, no. 3–4, pp. 1–19.
5. *Gao, J., He, X., Yih, S. W-t., and Deng, L.* Learning Continuous Phrase Representations for Translation Modeling, 2014, Microsoft Research, www.aclweb.org/anthology/P14-1066.
6. *Chicco, D., Sadowski, P., and Baldi, P.* Deep autoencoder neural networks for gene ontology annotation predictions, Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, pp. 533–540.
7. *Sathyanarayana, A., Joty, S., Fernandez-Luque, L., Ofli, F., Srivastava, J., Elmagarmid, A.,*

Arora, T., and Taheri, S. Sleep Quality Prediction From Wearable Data Using Deep Learning. JMIR Mhealth Uhealth, 2016, vol. 4, no. 4, p. 125.

8. *Movahedi, F., Coyle, J.L., and Sejdic, E.* Deep belief networks for electroencephalography: A review of recent contributions and future outlooks, IEEE J. Biomed Health Inform, 2018, May 22, vol. 3, pp. 642–652.
9. *Choi, E., Schuetz, A., Stewart, W.F., and Jimeng, S.* Using recurrent neural network models for early detection of heart failure onset, Journal of the American Medical Informatics Association, 2016.
10. *Elkahky, A.M., Song, Y., and He, X.* A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. Microsoft Research. http://sonyis.me/paperpdf/ /frp1159-songA-www-2015.pdf
11. *Yamins, D.L.K. and Di Carlo, J.J.* Using goal-driven deep learning models to understand sensory cortex. Nature Neuroscience, 2016, vol. 19, no. 3, pp. 356–365.
12. *Zorzi, M. and Testolin, A.* An emergentist perspective on the origin of number sense, Phil. Trans. R. Soc. B, 2018, vol. 373, no. 1740.
13. *Morel, D., Singh, C., and Levy, W.B.* Linearization of excitatory synaptic integration at no extra cost, Journal of Computational Neuroscience, April 2018, vol. 44, no. 2, pp. 173–188.
14. IEEE 829. Standard for Software Test Documentation. IEEE 1008. Standard for Software Unit Testing. https://www.twirpx.com/file/1615980/
15. ISO/МЭК 12119. Пакеты программ. Требования к качеству и тестированию. http://docs.cntd.ru/document/1200025075
16. ГОСТ Р 56920-2016, ГОСТ Р 56921–2016, ГОСТ Р 56922–2016. https://allgosts.ru.
17. ISO/IEC 29119-2013 1-5. Software testing. http://files.stroyinf.ru/Data2/1/4293754/ /4293754866.pdf
18. ГОСТ Р 12207-2010, ISO/IEC 12207:2008. http://docs.cntd.ru/document/1200082859
19. *Бейзер Б.* Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. СПб.: Питер, 2004.
20. *Dusting, E., Rashka, J., and Paul, J.* Automated Software Testing. Introduction, Management and Performance. Addison Wesley, 1999.
21. *Tamres, L.* Introducing Software Testing, Addison Wesley, 2002, русский перевод: Тампе Л. Введение в тестирование программного обеспечения, М.: Издательский дом "Вильямс", 2003.
22. *Кулямин В.В., Петренко А.К., Косачев А.С., Бурдонов И.Б.* Подход UniTesK к разработке тестов // Программирование. 2003. № 6. С. 25–43.
23. *Бурдонов И.Б., Косачев А.С., Кулямин В.В.* Теория соответствия для систем с блокировками и разрушением. М.: "Физ-мат лит" Наука, 2008, 411 с.
24. *Иванников В.П., Петренко А.К., Кулямин В.В., Максимов А.В.* Опыт использования UniTESK как зеркало развития технологий тестирования на основе моделей // Труды Института системного программирования РАН. 2013. Т. 24. С. 207–218.
25. *Kulyamin V.V., Petrenko, A.K.* Evolution of the UniTESK test development technology. Programming and Computer Software, 2014, vol. 24, no. 5, pp. 296–304.
26. *Yenigun, H., Kushik, N., Lopez, J., Yevtushenko, N., and Cavalli, A.R.* Decreasing the complexity of deriving test suites against nondeterministic finite state machines. Proceedings of 2017 IEEE East-West Design and Test Symposium, Proc. of East-West Design & Test Symposium (EWDTS), 2017, IEEE Xplore, pp. 1–4.
27. *Beck, K.* Test-Driven Development: By Example, Addison-Wesley, 2003.
28. *Astels, D.* Test-Driven Development. A Practical Guide, Prentice Hall, 2003.
29. *Rosenblatt, F.* Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961.
30. *Rumelhart, D.E., Hinton, G.E., and Williams, R.J.* Learning Internal Representations by Error Propagation, 1986. https://dl.acm.org/citation.cfm?id=104293.
31. *Rumelhart, D.E. and McClelland, J.L.*, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: Foundation. MIT Press, 1986.
32. *Hopfield, J.J.* Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences of the USA, April 1982, vol. 79, no. 8, pp. 2554–2558.

33. *Ackley, D.H., Hinton, G.E., and Sejnowski, T.J.* A Learning Algorithm for Boltzmann Machines. Cognitive Science, 1985, vol. 9, no. 1, pp. 147–169.

34. *Kohonen, T.* Self-Organized Formation of Topologically Correct Feature Maps, Biological Cybernetics, 1982, vol. 43, no. 1, pp. 59–69.

35. *Ивахненко А.Г., Лапа В.Г.* Кибернетические предсказывающие устройства. К.:"Наук. думка", 1965.

36. *Ivakhnenko, A.G. and Lapa, V.G.* Cybernetics and forecasting techniques. New York: Elsevier Publishing Company, Inc., 1967.

37. *Fukushima, K.* Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybern., 1980, vol. 36, pp. 193–202.

38. *Lecun, Y., Bosser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., and Jacket, L.D.* Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1989, vol. 1, no. 4, pp. 541–551.

39. *Hinton, G.E., Osindero, S., Teh, Y.W.* A Fast Learning Algorithm for Deep Belief Nets. Neural computation, 2006, vol. 18, pp. 1527–1554. http://dx.doi.org/10.1162/neco.2006.18.7.1527

40. *Hinton, G.E.* Learning multiple layers of representation, Trends in Cognitive Sciences, 2007, vol. 11, no. 10, pp. 428–434.

41. *Rumelhart, D.E., Hinton, G.E., and Williams, R.J.* Learning internal representations by backpropagating errors. Nature, 1986, vol. 323, pp. 533–536.

42. *Floreen, P.* Worst-Case Convergence Times for Hop-field Memories, IEEE Trans. Neural Networks, 1991, vol. 2, no. 5, pp. 533–535.

43. *Floreen, P.* The Convergence of Hamming Memory Networks, IEEE Trans. Neural Networks, 1991, vol. 2, no. 4, pp. 449–457.

44. *Utgoff, P.E. and Stracuzzi, D.J.* Many-layered learning. Neural Computation, 2002, vol. 14, pp. 2497–2529.

45. *Jeffrey, L., Elman, J.L, Bates, E.A., Johnson, M.H., Karmiloff-Smith, A., Parisi, D., and Plunkett, K.* Rethinking Innateness: A Connectionist Perspective on Development. Cambridge, MIT Press, 1996.

46. *Shrager, J. and Johnson, M.H.* Dynamic plasticity influences the emergence of function in a simple cortical array. Neural Networks, 1996, vol. 9, no. 7, pp. 1119–1129.

47. *Quartz, S.R. and Sejnowski, T.J.* The neural basis of cognitive development: A constructivist manifesto. Behavioral and Brain Sciences, 1997, vol. 20, no. 4, pp. 537–556.

48. *He, K., Zhang, X., Ren, S., and Sun, J.* Identity Mappings in Deep Residual Networks, European Conference on Computer Vision, 2016, pp. 630–645.

49. *Ivakhnenko, A.* Polynomial theory of complex systems. IEEE Transactions on Systems, Man and Cybernetics, 1971, vol. 4, no. 1, pp. 364–378.

50. *Bengio, Y., Boulanger-Lewandowski, N., and Pascanu, R.* Advances in optimizing recurrent networks, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 8624–8628. arXiv:1212.0901v2 [cs.LG]

51. *Dahl, G., Sainath, T., and Hinton, G.* Improving DNNs for LVCSR using rectified linear units and dropout, Proc. of International Conference on Acoustics, Speech and Signal Processing, 2011, pp. 8609–8613.

52. *Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R.R.* Improving neural networks by preventing co-adaptation of feature detectors, 2012, arXiv:1207.0580.

53. *Hinton, G.E. and Salakhutdinov, R.R.* Reducing the Dimensionality of Data with Neural Networks, Science, 2006, vol. 313, no. 5786, pp. 504–507.

54. *Кулямин В.В.* Технологии программирования. Компонентный подход. М. Интернет-университет информационных технологий. БИНОМ. Лаборатория знаний, 2007.

55. *Floreen, P. and Orponen, P.* Attraction Radii in Binary Hopfield Nets Are Hard to Compute, Neural Comput., 1993, vol. 5, pp. 812–821.