А. Л. Серебренников

СТАНДАРТНЫЕ И НОВЫЕ ПОДХОДЫ К АРХИТЕКТУРЕ И МЕТОДАМ ОБУЧЕНИЯ В СРЕДЕ SIGNIFICO, ОСНОВНЫЕ НАПРАВЛЕНИЯ РАЗВИТИЯ СРЕДЫ[*]

1. ВВЕДЕНИЕ

В данной работе описывается проектируемая интегрированная среда создания, обучения и диагностики нейросетей под названием Significo. Потенциальных пользователей среды Significo можно разделить на две категории. Первая категория пользователей — это исследователи как нейросетей, так и методов их обучения. Вторая категория — это разного рода аналитики.

В данное время на рынке программных продуктов широко представлены пакеты для использования нейросетевых технологий, но эти пакеты, как правило, предназначены для финансового прогнозирования. Доминирующее число существующих пакетов использует традиционные подходы и методы, имея только лишь продуманный и удобный интерфейс. Учитывая вышесказанное, становится явной востребованность в интегрированной среде, которая позволила бы не только специалистам в области нейросетевых технологий удобно проводить исследования, но и аналитикам анализировать различные предметные области.

Эта статья содержит шесть основных разделов.

- Задачи, решаемые с помощью нейросетевых технологий.
- Преобработка данных.
- Концепции передачи данных.
- Новые подходы к задаче проектирования структуры сети.
- Новые подходы к задаче обучения нейросетей.
- Интерфейсная часть.

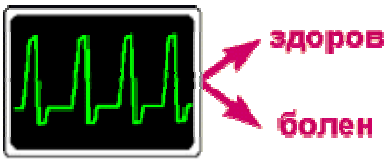
В этих разделах сначала будут рассматриваться стандартные подходы, а затем сравниваться с новыми по временным и качественным характеристикам.

[*] Работа выполнена при финансовой поддержке Министерства образования РФ (грант № E02-1.0-42).

2. ЗАДАЧИ, РЕШАЕМЫЕ С ПОМОЩЬЮ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ

Классификация

Отметим, что задачи классификации (типа распознавания букв) очень плохо алгоритмизуются. Если в случае распознавания букв верный ответ очевиден для нас заранее, то в более сложных практических задачах обученная нейросеть выступает как эксперт, обладающий большим опытом и способный дать ответ на трудный вопрос.



Примером такой задачи служит медицинская диагностика, где сеть может учитывать большое количество числовых параметров (энцефалограмма, давление, вес и т.д.). Конечно, «мнение» сети в этом случае нельзя считать окончательным.

Классификация предприятий по степени их перспективности — это уже привычный способ использования нейросетей в практике западных компаний. При этом сеть также использует множество экономических показателей, сложным образом связанных между собой.

Нейросетевой подход особенно эффективен в задачах экспертной оценки по той причине, что он сочетает в себе способность компьютера к обработке чисел и способность мозга к обобщению и распознаванию. Говорят, что у хорошего врача способность к распознаванию в своей области столь велика, что он может провести приблизительную диагностику уже по внешнему виду пациента. Можно согласиться также, что опытный трейдер чувствует направление движения рынка по виду графика. Однако в первом случае все факторы наглядны, т. е. характеристики пациента мгновенно воспринимаются мозгом как «бледное лицо», «блеск в глазах» и т.д. Во втором же случае учитывается только один фактор, показанный на графике, — курс за определенный период времени. Нейросеть позволяет обрабатывать ог-

ромное количество факторов (до нескольких тысяч) независимо от их наглядности; это универсальный «хороший врач», который может поставить свой диагноз в любой области.

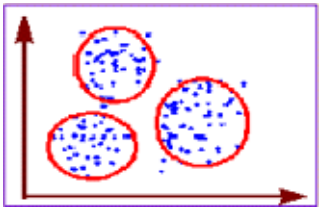
Кластеризация и поиск зависимостей

Помимо задач классификации, нейросети широко используются для поиска зависимостей в данных и кластеризации.

Например, нейросеть на основе методики МГУА (метод группового учета аргументов) позволяет на основе обучающей выборки построить зависимость одного параметра от других в виде полинома. Такая сеть может не только мгновенно выучить таблицу умножения, но и найти сложные скрытые зависимости в данных (например, финансовых), которые не обнаруживаются стандартными статистическими методами.

$$y = x_1^3 - 4x_3^2x_8^5 + x_3^2$$

Кластеризация — это разбиение набора примеров на несколько компактных областей (кластеров), причем число кластеров заранее неизвестно. Кластеризация позволяет представить неоднородные данные в более наглядном виде и использовать далее для исследования каждого кластера различные методы. Например, таким образом можно быстро выявить фальсифицированные страховые случаи или недобросовестные предприятия.



Прогнозирование

Задачи прогнозирования особенно важны для практики, в частности, для финансовых приложений, поэтому поясним способы применения нейросетей в этой области более подробно.

Рассмотрим практическую задачу, ответ в которой неочевиден, — задачу прогнозирования курса акций на 1 день вперед.

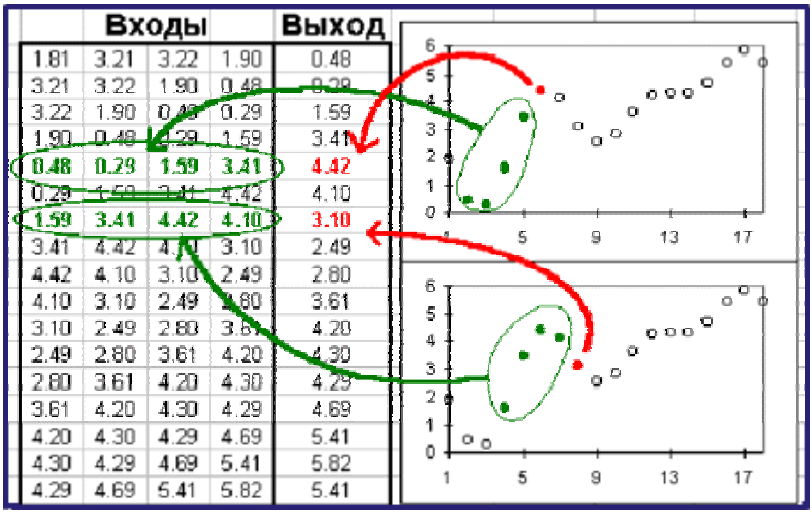
Пусть у нас имеется база данных, содержащая значения курса за последние 300 дней. Простейший вариант в данном случае — попытаться построить прогноз завтрашней цены на основе курсов за последние несколько дней. Понятно, что прогнозирующая сеть должна иметь всего один выход и столько входов, сколько предыдущих значений мы хотим использовать для прогноза, например, 4 последних значения. Составить обучающий пример

очень просто: входными значениями будут курсы за 4 последовательных дня, а желаемым выходом — известный нам курс в следующий день за этими четырьмя.

Если нейросеть совместима с какой-либо системой обработки электронных таблиц (например, Excel), то подготовка обучающей выборки состоит из следующих операций.

1. Скопировать столбец данных значений котировок в 4 соседних столбца.

2. Сдвинуть второй столбец на 1 ячейку вверх, третий столбец — на 2 ячейки вверх и т.д.



Смысл этой подготовки легко увидеть на рисунке: теперь каждая строка таблицы представляет собой обучающий пример, где первые 4 числа — входные значения сети, а пятое число — желаемое значение выхода. Исключение составляют последние 4 строки, где данных недостаточно, — эти строки не учитываются при тренировке. Заметим, что в четвертой снизу строке заданы все 4 входных значения, но неизвестно значение выхода. Именно к этой строке мы применим обученную сеть и получим прогноз на следующий день.

Как видно из этого примера, объем обучающей выборки зависит от выбранного нами количества входов. Если сделать 299 входов, то такая сеть

потенциально могла бы строить лучший прогноз, чем сеть с 4 входами, однако в этом случае мы имеем всего 1 обучающий пример, и обучение бессмысленно. При выборе числа входов следует учитывать это, выбирая разумный компромисс между глубиной предсказания (число входов) и качеством обучения (объем тренировочного набора).

3. ПРЕОБРАБОТКА ДАННЫХ

Преобработка данных имеет важнейшее значение, от возможностей этой части системы зависит в целом гибкость среды. Предполагается, что преобработка будет содержать 3 раздела.

Раздел символьной обработки конвертирует буквенно-численные выражения в числовой вид, который уже может быть обработан нейросетью. Например, понятия «холодно», «тепло», «жарко» можно представить числами 0, 1 и 2 соответственно.

Раздел логической обработки конвертирует данные числового вида, руководствуясь правилами пользователя, здесь имеются в виду правила типа «если—то—иначе».

Раздел спектральной обработки позволяет убрать из зависимостей помехи, например, различные циклически встречающиеся изменения в зависимостях.

4. НОВЫЕ ПОДХОДЫ К РЕШЕНИЮ ЗАДАЧ ПРОЕКТИРОВАНИЯ СТРУКТУРЫ НЕЙРОСЕТИ

Рассмотрим основные архитектуры нейросетей, имеющиеся на данный момент. В этой статье выделено несколько нейросетевых архитектур. Каждая архитектура наилучшим образом подходит для решения специфического круга задач.

Стандартные сети

Это стандартный тип сетей с обратным распространением, в котором каждый слой связан только с непосредственно предшествующим слоем. Это самая простая архитектура, и используется она в основном при решении задач распознавания образов.

Сети с обходными соединениями

В сетях с обратным распространением этого типа каждый слой связан со всеми предшествующими слоями. Задачи, выполняемые сетями с обходными путями, сходны с задачами, решаемыми стандартными сетями, но в некоторых случаях данные сети более помехоустойчивы.

Рекуррентные сети

Рекуррентные сети с обратным распространением ошибки очень хорошо подходят для решения задач прогнозирования. Например, часто применяются для финансовых биржевых предсказаний, поскольку эти сети могут запоминать последовательности. Благодаря этому свойству, такие сети являются прекрасным инструментом для работы с данными, представляющими собой временные серии.

Обычные сети с обратным распространением дают всегда в точности тот же самый ответ на один и тот же предъявляемый образ, тогда как ответ рекуррентной сети в подобных случаях будет зависеть от того, какие образы предъявлялись сети перед этим. Рекуррентные сети обладают долговременной памятью, построенной на внутренних нейронах.

Сети Ворда

Сети этого типа способны выделять различные свойства в данных, благодаря наличию в скрытом слое нескольких блоков, каждый из которых имеет свою передаточную функцию. Передаточные функции (обычно сигмоидного типа) служат для преобразования внутренней активности нейрона. Когда в разных блоках скрытого слоя используются разные передаточные функции, нейросеть оказывается способной выявлять новые свойства в предъявляемом образе.

Все вышеуказанные архитектуры являются архитектурами с обратным распространением ошибки и могут использовать разные способы подстройки весов: Простой, С Моментом и др. При Простом способе тренировки при предъявлении каждого примера происходит подстройка весов с заданной скоростью обучения. Подстройка С Моментом означает, что при модификации весов учитывается не только заданная скорость обучения, но и предыдущее изменение веса. TurboProp — это такой способ тренировки, при котором веса модифицируются после предъявления всех примеров (пакетная подстройка весов). Этот метод иногда работает гораздо быстрее, чем другие алгоритмы обратного распространения ошибки.

Все сети с обратным распространением ошибки используют Калибровку, что предотвращает возможность «переучивания» сети (и тем самым сильно сокращает время тренировки), а также улучшает способность сети обобщать информацию.

Сети Кохонена

Данные сети называют Саморганизующейся Картой Кохонена. Эта архитектура призвана решать задачи классификации. Сетям этого типа в процессе обучения не нужен «учитель», т.е. в процессе тренировки не требуется сообщать сети правильный ответ при предъявлении примера. Эти сети способны находить распределение данных на различные категории (классы).

Вероятностные нейронные сети

Вероятностные Нейронные Сети (ВНС) известны своей способностью обучаться на ограниченных наборах данных, причем для обучения нейросети достаточно однократного предъявления тренировочного набора! ВНС разделяет данные на указанное количество выходных категорий. Сеть ВНС зачастую способна работать уже после предъявления ей всего двух примеров из тренировочного набора, поэтому тренировка может осуществляться поэтапно.

Нейронные сети с общей регрессией

Подобно сетям ВНС, Нейронные Сети с Общей Регрессией (НСОР) известны своей способностью обучения в результате однократного предъявления тренировочных данных. Однако в отличие от сетей ВНС, которые классифицируют данные, сети НСОР способны предсказывать выходы с непрерывной амплитудой.

НСОР особенно полезны для аппроксимации непрерывных функций и могут строить многомерные поверхности, аппроксимирующие данные. Обнаружено, что для многих типов задач сети НСОР делают предсказания значительно лучше, чем сети с обратным распространением ошибки. Обращаем Ваше внимание: НСОР не использует приемов регрессионного анализа!

Для сетей ВНС и НСОР термин «Калибровка» означает оптимизацию параметра сглаживания, который используется в момент применения сетей. Модуль Калибровки с Генетическим Поиском использует генетический алгоритм для нахождения индивидуальных параметров сглаживания для каж-

дой входной переменной, что дает возможность получить сеть с очень хорошими обобщающими свойствами.

Полиномиальные сети

Полиномиальные сети представляют собой мощную архитектуру, называемую Методом группового учета аргумента (МГУА) или полиномиальными сетями. На самом деле, сеть МГУА непохожа на обычные сети с прямой связью, и изначально эта архитектура обычно не представлялась в виде сети. Сеть МГУА содержит в связях полиномиальные выражения и использует в некотором смысле аналогичный генетическим алгоритмам механизм принятия решения о том, сколько слоев необходимо построить. Результатом тренировки является возможность представить выход как полиномиальную функцию всех или части входов.

Новые подходы к проектированию архитектуры сети заключаются в том, что сеть должна быть разделена на отдельные взаимодействующие блоки с различным функциональным назначением. Как правило, аналитик, работающий над какой-либо предметной областью, может разделить поставленную перед ним задачу на функциональные блоки и определить между ними связь. При данной архитектуре функционального разделения общая сеть обладает двумя бесспорными плюсами. Первый плюс — это то, что одна большая сеть делится на несколько, это даёт выигрыш во времени обучения и потребляемых ресурсах.

В этом можно убедиться, рассмотрев график зависимости времени обучения нейросети на тестовом наборе обучающих пар от количества нейронов в нейросети (рис. 4.1).

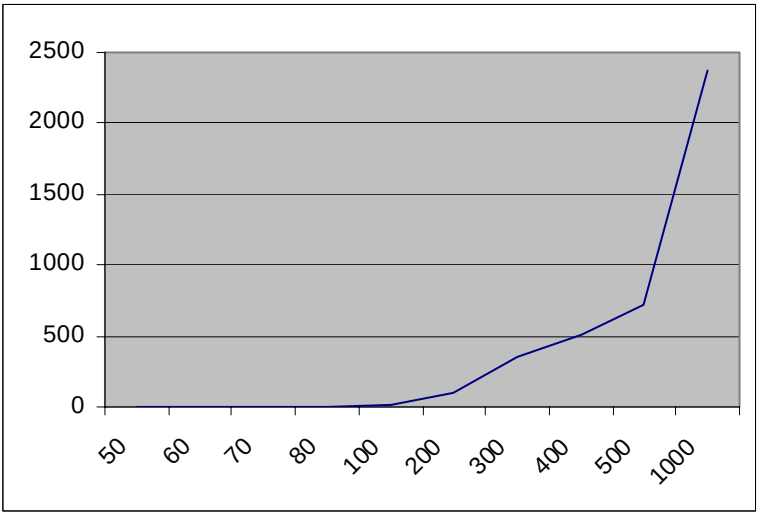


Рис. 4.1. График зависимости времени обучения нейросети от количества нейронов в ней

Второй плюс — это то, что отдельные блоки сети могут состоять из сетей различной архитектуры, что позволяет решать различные функциональные задачи сетями соответствующих архитектур.

5. НОВЫЕ ПОДХОДЫ К ОБУЧЕНИЮ НЕЙРОСЕТЕЙ

Обучение сети — это самый ресурсоёмкий процесс в работе с нейросетями. Оптимизация методов обучения позволяет открыть новые возможности нейросетей. Рассмотрим стандартные методы обучения нейросетей обратного распространения. Эти методы делятся в общем случае на 2 категории: рекурсивные и стохастические.

Долгое время не было теоретически обоснованного алгоритма для обучения многослойных искусственных нейронных сетей. А так как возможности представления с помощью однослойных нейронных сетей оказались весьма ограниченными, то и вся область в целом пришла в упадок.

Разработка алгоритма обратного распространения сыграла важную роль в возрождении интереса к искусственным нейронным сетям. Обратное распространение — это систематический метод для обучения многослойных

искусственных нейронных сетей. Он имеет солидное математическое обоснование. Несмотря на некоторые ограничения, процедура обратного распространения сильно расширила область проблем, в которых могут быть использованы искусственные нейронные сети, и убедительно продемонстрировала свою мощь.

Сетевые конфигурации

На рис. 5.1 показан нейрон, используемый в качестве основного строительного блока в сетях обратного распространения. Подается множество входов, идущих либо извне, либо от предшествующего слоя. Каждый из них умножается на вес, и произведения суммируются. Эта сумма, обозначаемая NET, должна быть вычислена для каждого нейрона сети. После того как величина NET вычислена, она модифицируется с помощью активационной функции и получается сигнал OUT.

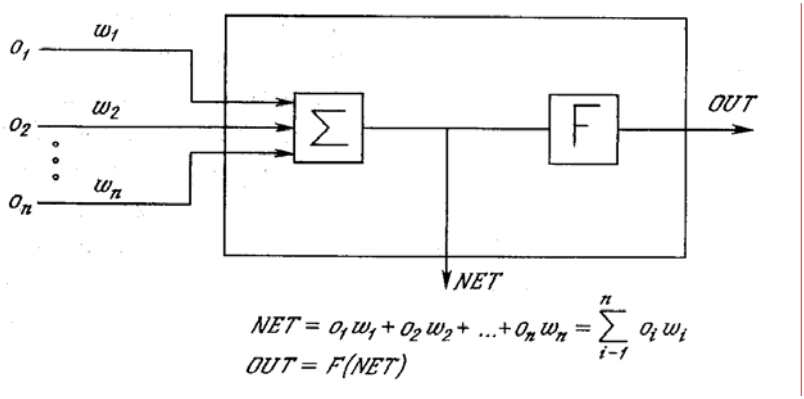


Рис. 5.1. Искусственный нейрон с активационнной функцией

На рис. 5.2 показана активационная функция, обычно используемая для обратного распространения.

$$\mathrm{OUT} = \frac{1}{1+\mathrm{e}^{-\mathrm{NET}}}. \qquad (5.1)$$

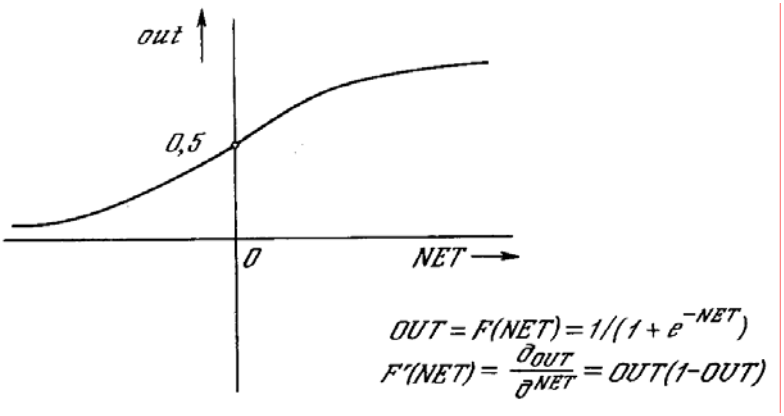


Рис. 5.2. Сигмоидальная активационная функция

Как показывает уравнение (5.2), эта функция, называемая *сигмоидом,* весьма удобна, так как имеет простую производную, что используется при реализации алгоритма обратного распространения.

$$\frac{\partial \mathrm{OUT}}{\partial \mathrm{NET}} = \mathrm{OUT}(1-\mathrm{OUT}). \qquad (5.2)$$

Сигмоид, который иногда называется также *логистической* или *сжимающей функцией,* сужает диапазон изменения NET так, что значение OUT лежит между нулем и единицей. Как указывалось выше, многослойные нейронные сети обладают большей представляющей мощностью, чем однослойные, только в случае присутствия нелинейности. Сжимающая функция обеспечивает требуемую нелинейность.

В действительности имеется множество функций, которые могли бы быть использованы. Для алгоритма обратного распространения требуется лишь, чтобы функция была всюду дифференцируема. Сигмоид удовлетворяет этому требованию. Его дополнительное преимущество состоит в автоматическом контроле усиления. Для слабых сигналов (величина NET близка к нулю) кривая вход-выход имеет сильный наклон, дающий большое усиление. Когда величина сигнала становится больше, усиление падает. Таким образом, большие сигналы воспринимаются сетью без насыщения, а слабые сигналы проходят по сети без чрезмерного ослабления.

Многослойная сеть

На рис. 5.3 изображена многослойная сеть, которая может обучаться с помощью процедуры обратного распространения (для ясности рисунок упрощен). Первый слой нейронов (соединенный с входами) служит лишь в качестве распределительных точек, суммирования входов здесь не производится. Входной сигнал просто проходит через них к весам на их выходах. А каждый нейрон последующих слоев выдает сигналы NET и OUT, как описано выше.

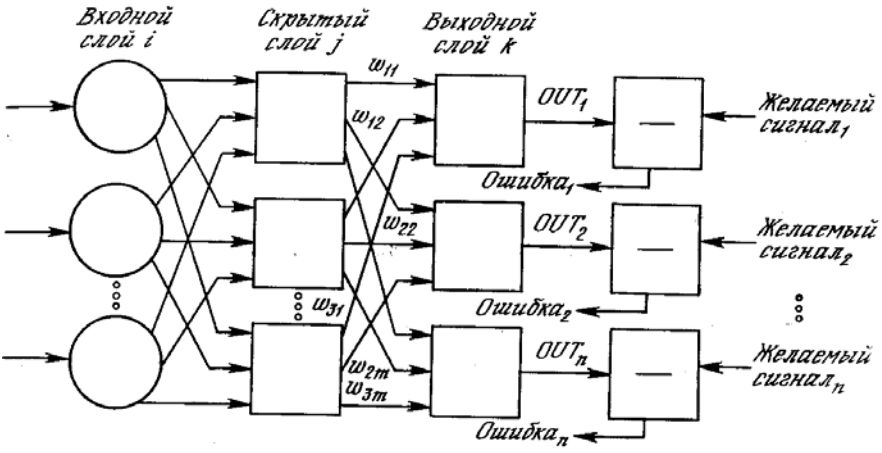


Рис. 5.3. Двухслойная сеть обратного распространения (ε — желаемый сигнал)

В литературе нет единообразия относительно того, как считать число слоев в таких сетях. Одни авторы используют число слоев нейронов (включая несуммирующий входной слой), другие — число слоев весов. Так как последнее определение функционально описательное, то оно будет использоваться на протяжении всей статьи. Согласно этому определению, сеть на рис. 5.3 рассматривается как двухслойная. Нейрон объединен с множеством весов, присоединенных к его входу. Таким образом, веса первого слоя оканчиваются на нейронах первого слоя. Вход распределительного слоя считается нулевым слоем.

Процедура обратного распространения применима к сетям с любым числом слоев. Однако для того чтобы продемонстрировать алгоритм, достаточно двух слоев. Сейчас будут рассматриваться лишь сети прямого действия, хотя обратное распространение применимо и к сетям с обратными связями.

Обучение сети обратного распространения требует выполнения следующих операций.

1. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети.
2. Вычислить выход сети.
3. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).
4. Подкорректировать веса сети так, чтобы минимизировать ошибку.
5. Повторять шаги с 1 по 4 для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

Операции, выполняемые шагами 1 и 2, сходны с теми, которые выполняются при функционировании уже обученной сети, т. е. подается входной вектор и вычисляется получающийся выход. Вычисления выполняются послойно. На рис. 5.3 сначала вычисляются выходы нейронов слоя *j*, затем они используются в качестве входов слоя *k*, вычисляются выходы нейронов слоя *k*, которые и образуют выходной вектор сети.

На шаге 3 каждый из выходов сети, которые на рис. 5.3 обозначены OUT, вычитается из соответствующей компоненты целевого вектора, чтобы получить ошибку. Эта ошибка используется на шаге 4 для коррекции весов сети, причем знак и величина изменений весов определяются алгоритмом обучения (см. ниже).

После достаточного числа повторений этих четырех шагов разность между действительными выходами и целевыми выходами должна уменьшиться до приемлемой величины, при этом говорят, что сеть «обучилась». Теперь сеть используется для распознавания, и веса не изменяются.

На шаги 1 и 2 можно смотреть как на «проход вперед», так как сигнал распространяется по сети от входа к выходу. Шаги 3, 4 составляют «обратный проход», здесь вычисляемый сигнал ошибки распространяется обратно по сети и используется для подстройки весов. Эти два прохода теперь будут детализированы и выражены в более математической форме.

Проход вперед. Шаги 1 и 2 могут быть выражены в векторной форме следующим образом: подается входной вектор X, и на выходе получается вектор Y. Векторная пара вход-цель X и T берется из обучающего множества. Вычисления проводятся над вектором X, чтобы получить выходной вектор Y.

Как мы видели, вычисления в многослойных сетях выполняются слой за слоем, начиная с ближайшего к входу слоя. Величина NET каждого нейрона первого слоя вычисляется как взвешенная сумма входов нейрона. Затем активационная функция F «сжимает» NET и дает величину OUT для каждого нейрона в этом слое. Когда множество выходов слоя получено, оно является входным множеством для следующего слоя. Процесс повторяется слой за слоем, пока не будет получено заключительное множество выходов сети.

Этот процесс может быть выражен в сжатой форме с помощью векторной нотации. Веса между нейронами могут рассматриваться как матрица W. Например, вес от нейрона 8 в слое 2 к нейрону 5 слоя 3 обозначается $w_{8,5}$. Тогда NET-вектор слоя N может быть выражен не как сумма произведений, а как произведение X и W. В векторном обозначении N = XW. Покомпонентным применением функции F к NET-вектору N получается выходной вектор O. Таким образом, для данного слоя вычислительный процесс описывается следующим выражением:

$$O = F(XW). \tag{5.3}$$

Выходной вектор одного слоя является входным вектором для следующего, поэтому вычисление выходов последнего слоя требует применения уравнения (5.3) к каждому слою от входа сети к ее выходу.

Обратный проход. *Подстройка весов выходного слоя.* Так как для каждого нейрона выходного слоя задано целевое значение, то подстройка весов легко осуществляется с использованием модифицированного дельта-правила. Внутренние слои называют «скрытыми слоями», для их выходов не имеется целевых значений для сравнения. Поэтому обучение усложняется.

На рис. 5.4 показан процесс обучения для одного веса от нейрона p в скрытом слое j к нейрону q в выходном слое k. Выход нейрона слоя k, вычитаясь из целевого значения (Target), дает сигнал ошибки. Он умножается на производную сжимающей функции [OUT(1 — OUT)], вычисленную для этого нейрона слоя k, что дает величину δ:

$$\delta = OUT(1 — OUT)(Target — OUT). \tag{5.4}$$

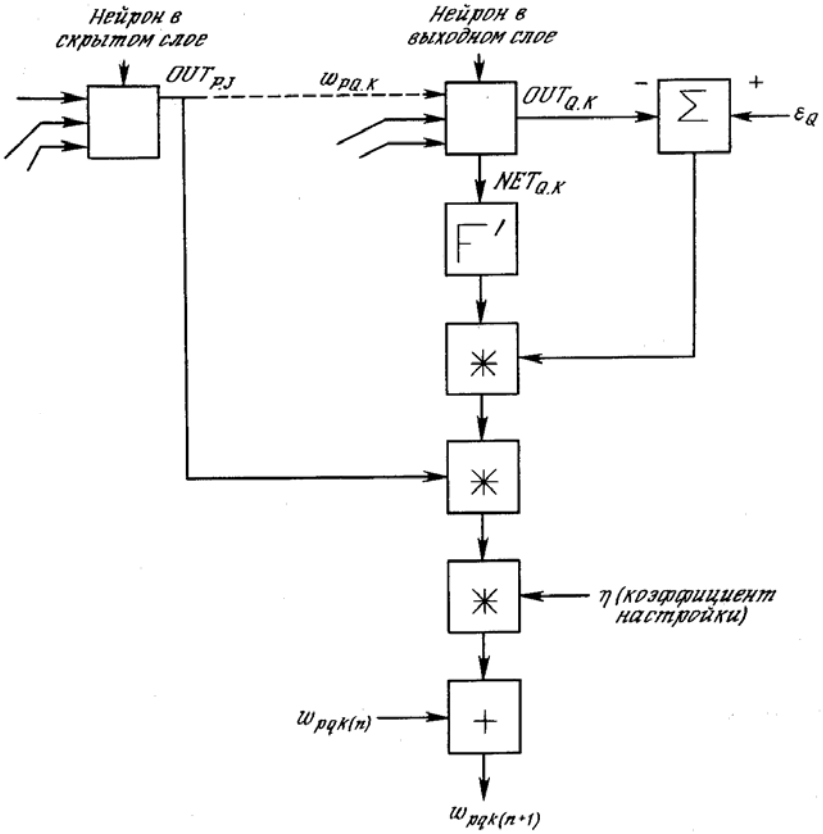


Рис. 5.4. Настройка веса в выходном слое

Затем δ умножается на величину OUT нейрона *j*, из которого выходит рассматриваемый вес. Это произведение, в свою очередь, умножается на коэффициент скорости обучения η (обычно от 0,01 до 1,0), и результат прибавляется к весу. Такая же процедура выполняется для каждого веса от нейрона скрытого слоя к нейрону в выходном слое.

Следующие уравнения иллюстрируют это вычисление:

$$\Delta w_{pq,k} = \eta\, \delta_{q,k}\, \mathrm{OUT} \tag{5.5}$$

$$w_{pq,k}(n+1) = w_{pq,k}(n) + \Delta w_{pq,k}, \tag{5.6}$$

где $w_{pq,k}(n)$ — величина веса от нейрона p в скрытом слое к нейрону q в выходном слое на шаге n (до коррекции); отметим, что индекс k относится к слою, в котором заканчивается данный вес, т. е. согласно принятому здесь соглашению, с которым он объединен; $w_{pq,k}(n+1)$ — величина веса на шаге n + 1 (после коррекции); $\delta_{q,k}$ — величина δ для нейрона q в выходном слое k; $OUT_{p,j}$ — величина OUT для нейрона p в скрытом слое j.

Подстройка весов скрытого слоя. Рассмотрим один нейрон в скрытом слое, предшествующем выходному слою. При проходе вперед этот нейрон передает свой выходной сигнал нейронам в выходном слое через соединяющие их веса. Во время обучения эти веса функционируют в обратном порядке, пропуская величину δ от выходного слоя назад к скрытому слою. Каждый из этих весов умножается на величину δ нейрона, к которому он присоединен в выходном слое. Величина δ, необходимая для нейрона скрытого слоя, получается суммированием всех таких произведений и умножением на производную сжимающей функции (см. рис. 5.5):

$$\delta_{q,k} = OUT_{p,j}(1 - OUT_{p,j})\left[\sum_{q} \delta_{q,k} w_{pq,k}\right] \qquad (5.7)$$

Когда значение δ получено, веса, питающие первый скрытый уровень, могут быть подкорректированы с помощью уравнений (5.5) и (5.6), где индексы модифицируются в соответствии со слоем.

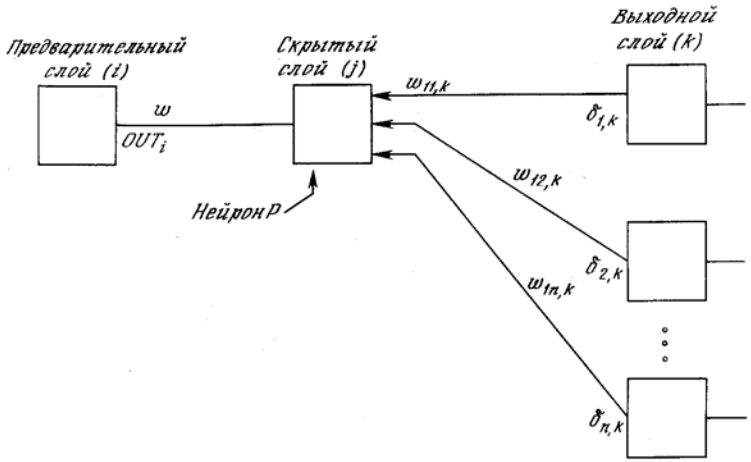


Рис. 5.5. Настройка веса в скрытом слое

Для каждого нейрона в данном скрытом слое должно быть вычислено δ и подстроены все веса, ассоциированные с этим слоем. Этот процесс повторяется слой за слоем по направлению к входу, пока все веса не будут подкорректированы

Стохастические методы

Стохастические методы полезны как для обучения искусственных нейронных сетей, так и для получения выхода от уже обученной сети. Стохастические методы обучения приносят большую пользу, позволяя исключать локальные минимумы в процессе обучения. Но с ними также связан ряд проблем.

Использование стохастических методов для получения выхода от уже обученной сети рассматривалось в работе [2].

Искусственная нейронная сеть обучается посредством некоторого процесса, модифицирующего ее веса. Если обучение успешно, то предъявление сети множества входных сигналов приводит к появлению желаемого множества выходных сигналов. Имеются два класса обучающих методов: детерминистский и стохастический.

Детерминистский метод обучения шаг за шагом осуществляет процедуру коррекции весов сети, основанную на использовании их текущих значений, а также величин входов, фактических выходов и желаемых выходов.

Стохастические методы обучения выполняют псевдослучайные изменения величин весов, сохраняя те изменения, которые ведут к улучшениям. Чтобы увидеть, как это может быть сделано, рассмотрим рис. 5.6, на котором изображена типичная сеть, в которой нейроны соединены с помощью весов. Выход нейрона является здесь взвешенной суммой его входов, которая преобразована с помощью нелинейной функции. Для обучения сети может быть использована следующая процедура.

1. Выбрать вес случайным образом и подкорректировать его на небольшое случайное значение. Предъявить множество входов, и вычислить получающиеся выходы.

2. Сравнить эти выходы с желаемыми выходами и вычислить величину разности между ними. Общепринятый метод состоит в нахождении разности между фактическим и желаемым выходами для каждого элемента обучаемой пары, возведение разностей в квадрат и нахождение суммы этих квадратов. Целью обучения является минимизация этой разности, часто называемой *целевой функцией.*

3. Выбрать вес случайным образом и подкорректировать его на небольшое случайное значение. Если коррекция помогает (уменьшает целевую функцию), то сохранить ее, в противном случае вернуться к первоначальному значению веса.

4. Повторять шаги с 1 до 3 до тех пор, пока сеть не будет обучена в достаточной степени.

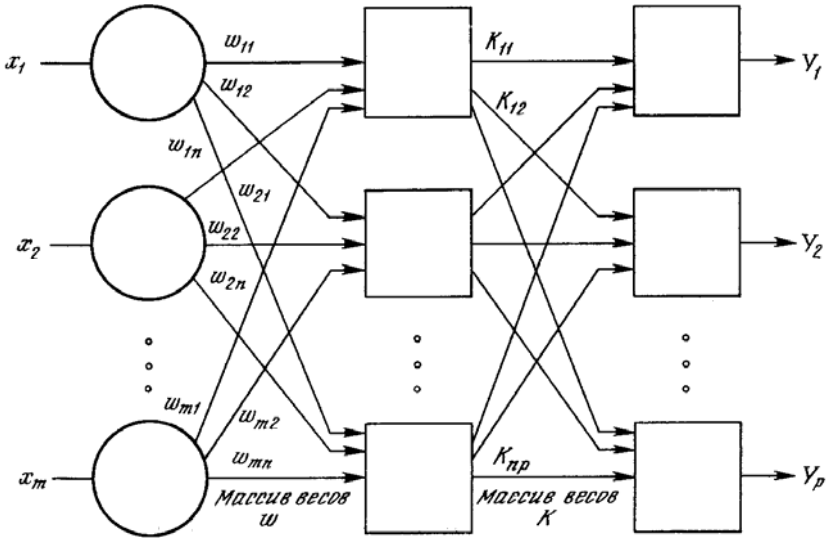


Рис. 5.6. Двухслойная сеть без обратных связей

Этот процесс стремится минимизировать целевую функцию, но может попасть, как в ловушку, в неудачное решение. На рис. 5.7 показано, как это может иметь место в системе с единственным весом. Допустим, что первоначально вес взят равным значению в точке А. Если случайные шаги по весу малы, то любые отклонения от точки А увеличивают целевую функцию и будут отвергнуты. Лучшее значение веса, принимаемое в точке В, никогда не будет найдено, и система будет поймана в ловушку локальным минимумом, вместо глобального минимума в точке В. Если же случайные коррекции веса очень велики, то как точка А, так и точка В будут часто посещаться, но то же самое справедливо и для каждой другой точки. Вес будет меняться так резко, что он никогда не установится в желаемом минимуме.

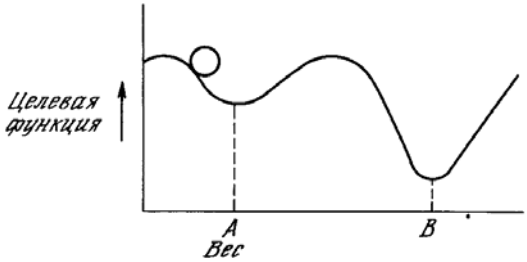


Рис.5.7. Проблема локальных минимумов

Полезная стратегия для избежания подобных проблем состоит в больших начальных шагах и постепенном уменьшении размера среднего случайного шага. Это позволяет сети вырываться из локальных минимумов и в то же время гарантирует окончательную стабилизацию сети.

Ловушки локальных минимумов досаждают всем алгоритмам обучения, основанным на поиске минимума, включая персептрон и сети обратного распространения, и представляют серьезную и широко распространенную трудность, которой часто не замечают. Стохастические методы позволяют решить эту проблему. Стратегия коррекции весов, вынуждающая веса принимать значение глобального оптимума в точке B, возможна.

В качестве объясняющей аналогии предположим, что на рис. 5.7 изображен шарик на поверхности в коробке. Если коробку сильно потрясти в горизонтальном направлении, то шарик будет быстро перекатываться от одного края к другому. Нигде не задерживаясь, в каждый момент шарик будет с равной вероятностью находиться в любой точке поверхности.

Если постепенно уменьшать силу встряхивания, то будет достигнуто условие, при котором шарик будет на короткое время «застревать» в точке B. При еще более слабом встряхивании шарик будет на короткое время останавливаться как в точке A, так и в точке B. При непрерывном уменьшении силы встряхивания будет достигнута критическая точка, когда сила встряхивания достаточна для перемещения шарика из точки A в точку B, но недостаточна для того, чтобы шарик мог передвинуться из B в A. Таким образом, окончательно шарик остановится в точке глобального минимума, когда амплитуда встряхивания уменьшится до нуля.

Искусственные нейронные сети могут обучаться по существу таким же образом посредством случайной коррекции весов. Вначале делаются большие случайные коррекции с сохранением только тех изменений весов, ко-

торые уменьшают целевую функцию. Затем средний размер шага постепенно уменьшается, и глобальный минимум в конце концов достигается.

Это сильно напоминает отжиг металла, поэтому для ее описания часто используют термин «имитация отжига». В металле, нагретом до температуры, превышающей его точку плавления, атомы находятся в сильном беспорядочном движении. Как и во всех физических системах, атомы стремятся к состоянию минимума энергии (единому кристаллу в данном случае), но при высоких температурах энергия атомных движений препятствует этому. В процессе постепенного охлаждения металла возникают все более низкоэнергетические состояния, пока в конце концов не будет достигнуто низшее из возможных состояний, глобальный минимум. В процессе отжига распределение энергетических уровней описывается следующим соотношением:

$$P(e) = \exp(-e/kT), \tag{5.8}$$

где $P(e)$ — вероятность того, что система находится в состоянии с энергией e; k — постоянная Больцмана; T — температура по шкале Кельвина.

При высоких температурах $P(e)$ приближается к единице для всех энергетических состояний. Таким образом, высокоэнергетическое состояние почти столь же вероятно, как и низкоэнергетическое. По мере уменьшения температуры вероятность высокоэнергетических состояний уменьшается по сравнению с низкоэнергетическими. При приближении температуры к нулю становится весьма маловероятным, чтобы система находилась в высокоэнергетическом состоянии.

Больцмановское обучение

Этот стохастический метод непосредственно применим к обучению искусственных нейронных сетей.

1. Определить переменную T, представляющую искусственную температуру. Придать T большое начальное значение.
2. Предъявить сети множество входов, и вычислить выходы и целевую функцию.
3. Дать случайное изменение весу, и пересчитать выход сети и изменение целевой функции в соответствии со сделанным изменением веса.
4. Если целевая функция уменьшилась (улучшилась), то сохранить изменение веса.

Если изменение веса приводит к увеличению целевой функции, то вероятность сохранения этого изменения вычисляется с помощью распределения Больцмана:

$$P(c) = \exp(-c/kT), \tag{5.9}$$

где $P(c)$ — вероятность изменения c в целевой функции; k — константа, аналогичная константе Больцмана, выбираемая в зависимости от задачи; T — искусственная температура.

Выбирается случайное число r из равномерного распределения от нуля до единицы. Если $P(c)$ больше, чем r, то изменение сохраняется, в противном случае величина веса возвращается к предыдущему значению.

Это позволяет системе делать случайный шаг в направлении, портящем целевую функцию, позволяя ей тем самым вырываться из локальных минимумов, где любой малый шаг увеличивает целевую функцию.

Для завершения больцмановского обучения повторяют шаги 3 и 4 для *каждого* из весов сети, постепенно уменьшая температуру T, пока не будет достигнуто допустимо низкое значение целевой функции. В этот момент предъявляется другой входной вектор и процесс обучения повторяется. Сеть обучается на всех векторах обучающего множества, с возможным повторением, пока целевая функция не станет допустимой для всех них.

Величина случайного изменения веса на шаге 3 может определяться различными способами. Например, подобно тепловой системе весовое изменение w может выбираться в соответствии с гауссовским распределением:

$$P(w) = \exp(-w^2/T^2), \tag{5.10}$$

где $P(w)$ — вероятность изменения веса на величину w, T — искусственная температура.

Такой выбор изменения веса приводит к системе, аналогичной [3].

Так как нужна величина изменения веса Δw, а не вероятность изменения веса, имеющего величину w, то метод Монте-Карло может быть использован следующим образом.

1. Найти кумулятивную вероятность, соответствующую $P(w)$. Это есть интеграл от $P(w)$ *в* пределах от 0 до w. Так как в данном случае $P(w)$ не может быть проинтегрирована аналитически, она должна интегрироваться численно, а результат необходимо затабулировать.

2. Выбрать случайное число из равномерного распределения на интервале (0,1). Используя эту величину в качестве значения $P(w)$, найти в таблице соответствующее значение для величины изменения веса.

Свойства машины Больцмана широко изучались. В работе [1] показано, что скорость уменьшения температуры должна быть обратно пропорциональна логарифму времени, чтобы была достигнута сходимость к глобальному минимуму. Скорость охлаждения в такой системе выражается следующим образом:

$$T(t) = \frac{T_0}{\log(1+t)}, \tag{5.11}$$

где $T(t)$ — искусственная температура как функция времени; T_0 — начальная искусственная температура; t — искусственное время.

Этот разочаровывающий результат предсказывает очень медленную скорость охлаждения (и данные вычисления). Этот вывод подтвердился экспериментально. Машины Больцмана часто требуют для обучения очень большого ресурса времени.

Обучение Коши

В работе [6] развит метод быстрого обучения подобных систем. В этом методе при вычислении величины шага распределение Больцмана заменяется на распределение Коши. Распределение Коши имеет, как показано на рис. 5.8, более длинные «хвосты», тем самым увеличивается вероятность больших шагов. В действительности распределение Коши имеет бесконечную (неопределенную) дисперсию. С помощью такого простого изменения максимальная скорость уменьшения температуры становится обратно пропорциональной линейной величине, а не логарифму, как для алгоритма обучения Больцмана. Это резко уменьшает время обучения. Эта связь может быть выражена следующим образом.

$$T(t) = \frac{T_0}{1+t}. \tag{5.12}$$

Распределение Коши имеет вид

$$P(x) = \frac{T(t)}{T(t)^2 + x^2}, \tag{5.13}$$

где $P(x)$ есть вероятность шага величины x.

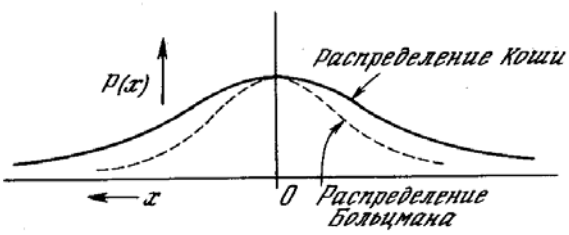


Рис. 5.8. Распределение Коши и распределение Больцмана

В уравнении (5.13) $P(x)$ может быть проинтегрирована стандартными методами. Решая относительно x, получаем:

$$x_c = \rho\, T(t)\, \mathrm{tg}(P(x)), \qquad (5.14)$$

где ρ — коэффициент скорости обучения; x_c — изменение веса.

Теперь применение метода Монте Карло становится очень простым. Для нахождения x в этом случае выбирается случайное число из равномерного распределения на открытом интервале $(-\pi/2, \pi/2)$ (необходимо ограничить функцию тангенса). Оно подставляется в формулу (5.13) в качестве $P(x)$, и с помощью текущей температуры вычисляется величина шага.

Метод искусственной теплоемкости

Несмотря на улучшение, достигаемое с помощью метода Коши, время обучения может оказаться все еще слишком большим. Способ, полученный на основе законов термодинамики, может быть использован для ускорения этого процесса. В этом методе скорость уменьшения температуры изменяется в соответствии с искусственной «теплоемкостью», вычисляемой в процессе обучения.

Во время отжига металла происходят фазовые переходы, связанные с дискретными изменениями уровней энергии. При каждом фазовом переходе может иметь место резкое изменение величины, называемой теплоемкостью. *Теплоемкость* определяется как скорость изменения температуры с энергией. Изменения теплоемкости происходят из-за попадания системы в локальные энергетические минимумы.

Искусственные нейронные сети проходят аналогичные фазы в процессе обучения. На границе фазового перехода искусственная теплоемкость может скачкообразно измениться. Эта псевдотеплоемкость определяется как

средняя скорость изменения температуры с целевой функцией. В примере шарика в коробке сильная начальная встряска делает среднюю величину целевой функции фактически не зависящей от малых изменений температуры, т. е. теплоемкость близка к константе. Аналогично при очень низких температурах система замерзает в точке минимума, так что теплоемкость снова близка к константе. Ясно, что в каждой из этих областей допустимы сильные изменения температуры, так как не происходит улучшения целевой функции.

При критических температурах небольшое уменьшение температуры приводит к большому изменению средней величины целевой функции. Возвращаясь к аналогии с шариком, при «температуре», когда шарик обладает достаточной средней энергией, чтобы перейти из А в В, но недостаточной — для перехода из В в А, средняя величина целевой функции испытывает скачкообразное изменение. В этих критических точках алгоритм должен изменять температуру очень медленно, чтобы гарантировать, что система не замерзнет случайно в точке А, оказавшись пойманной в локальный минимум. Критическая температура может быть обнаружена по резкому уменьшению искусственной теплоемкости, т. е. средней скорости изменения температуры с целевой функцией. При достижении критической температуры скорость изменения температуры должна замедляться, чтобы гарантировать сходимость к глобальному минимуму. При всех остальных температурах может без риска использоваться более высокая скорость снижения температуры, что приводит к значительному снижению времени обучения.

В процессе работы с рекурсивным методом было сделано добавление к нему в виде модуля управления скоростью обучения и модуля калибровки. При обучении сети при каждой итерации вычисляется значение калибровочной ошибки на всём наборе векторов обучения, далее в зависимости от градиента ошибки вычисляется значение скорости. При использовании данного оптимизационного метода общая скорость обучения выросла, и в то же время уменьшилась вероятность попадания процесса обучения в локальный минимум. Это можно заключить из проведённых опытов, состоящих в следующем: берутся две одинаковые нейросети и два одинаковых набора обучающих пар, производится обучение с одинаковым количеством рекурсий. В итоге опыта на стандартной процедуре обучения уровень ошибки получился равным 0,41, при обучении усовершенствованным методом при тех же условиях ошибка составила 0,992. Также (что не маловажно) у пользователя появилась возможность контролировать степень облученности сети. В дальнейшем предполагается автоматически определять количество итераций, необходимое для обучения сети с определённой точностью.

6. ИНТЕРФЕЙСНАЯ ЧАСТЬ

Так как предполагается использование среды как исследователями нейросетей так и различного рода аналитиками, интерфейс должен включать в себя как простоту и наглядность происходящих процессов, так и многофункциональность.

Наглядность происходящих процессов предполагает развитую систему визуализации с применением трехмерного моделирования. Также необходимо снабдить интерфейс удобным инструментом отслеживания процесса обучения с отображением истории различного рода ошибок в процессе обучения и других параметров.

Гибкость интерфейса предполагает, помимо возможности его изменения, возможность легкого манипулирования результатами различных операций с нейросетями. Предполагается использование OLE технологии для передачи результатов в пакет MS Office, что позволит пользователю доработать результаты (например, составить диаграмму или график) или передать данные для дальнейшей обработки в другие программы (например, MathCAD).

Среда Significo должна поддерживать несколько стандартов результатных файлов. Это файлы, содержащие результаты работы отдельных модулей среды. Планируется наличие проектных файлов, в которых будут связываться все поддерживаемые файлы среды.

7. ЗАКЛЮЧЕНИЕ

Практическим результатом данной работы является описание проектируемой интегрированной среды Significo и описание применяемых в ней технологий, видов сетей, алгоритмов обучения. Приведены новые подходы в области проектирования архитектуры сетей с практическим обоснованием выигрыша по времени обучения. Подробно представлены стандартные методы обучения нейросетей и наряду с ними новые подходы к обучению сетей и усовершенствованию стандартных методов обучения. Представлен результат проведённых опытов, которые показали, что средняя скорость обучения нейросети увеличилась примерно в 4 раза.

СПИСОК ЛИТЕРАТУРЫ

1. Geman S., Geman D. Stohastic relaxation, Gibbs distribution and Baysian restoration of images // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 1984. — Vol. 6. — P. 721–741.
2. Hinton G. E., Sejnowski T. J. Learning and relearning in Boltzmann machines // Parallel distributed processing. — Cambridge, MA: MIT Press. — 1986. — Vol. 1. — P. 282–317..
3. Metropolis N., Rosenbluth A. W-.Rosenbluth M. N., Teller A. N., Teller E. Equations of state calculations by fast computing machines // J. of Chemistry and Physics. — 1953. — Vol. 21. — P. 1087–1091.
4. Материалы сайта `www.neuroproject.ru`
5. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика / Пер. на русский язык Ю. А. Зуев, В. А. Точенов. — М.: Мир. 1992. — 66 с.
6. Szu H., Hartley R. Fast Simulated annealing // Physics Letters. — 1987. — Vol. 1222, N 3,4. — P. 157–162.