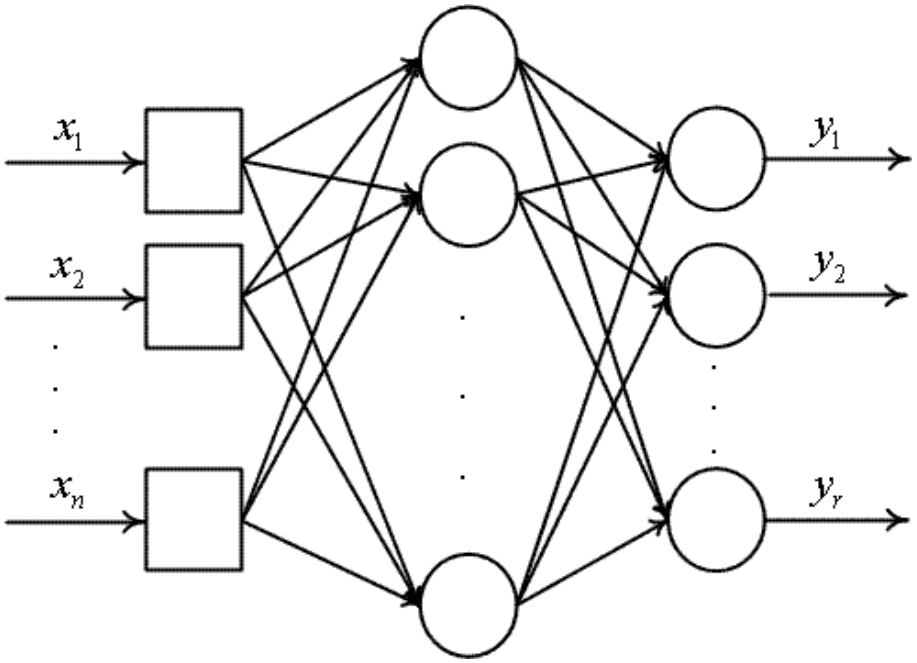
П.В. Сараев

ИДЕНТИФИКАЦИЯ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ



Липецк

Липецкий государственный технический университет

2011

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Липецкий государственный технический университет»

П.В. Сараев

Идентификация нейросетевых моделей

Монография

Липецк
Липецкий государственный технический университет
2011

УДК 004.8
С 20

Рецензенты:
д-р техн. наук, проф. Малыш В.Н.;
канд. физ.-мат. наук, доц. Соболев Б.В.

Сараев, П.В.

С 20 Идентификация нейросетевых моделей [Текст]: монография.– Липецк: Изд-во ЛГТУ, 2011.– 94 с.

ISBN 978-5-88247-516-0

В монографии представлены основные понятия и алгоритмы применения нейронных сетей прямого распространения в решении задач моделирования. Основное внимание уделено обучению нейронных сетей, приведены алгоритмы повышения эффективности обучения на основе матричного псевдообращения. Рассмотрено применение алгоритмов гарантированной глобальной оптимизации на основе методов интервального анализа в обучении нейронных сетей. Большое внимание уделено также задаче конструирования нейросетевых моделей оптимальной структуры.

Монография будет полезна научным работникам, аспирантам и студентам математических и технических специальностей, занимающихся вопросами математического моделирования и интеллектуальными системами.

Табл. 3. Ил. 21. Библиогр.: 41 назв.

ISBN 978-5-88247-516-0

Оглавление

Введение

В настоящее время нейронные сети являются мощным инструментом моделирования анализа сложных систем. Нейронные сети с успехом применяются в решении задач идентификации систем, прогнозировании поведения динамических систем, управлении сложными системами, распознавании образов, классификации и кластеризации объектов. Они могут быть эффективно применены в тех случаях, когда о моделируемом объекте известна только информация о его входных и выходных сигналах. Для возможности применения нейронных сетей их необходимо вначале настроить на описание известных вход-выходных данных, что осуществляется в процессе конструирования сетей оптимальной структуры и обучения. В теории моделирования эти задачи решаются на этапах структурной и параметрической идентификации соответственно.

Начало теории нейронных сетей положено в 1943 г. в работе «Логическое исчисление идей, относящихся к нервной деятельности» («A Logical Calculus of Ideas Immanent in Nervous Activity») МакКаллока (McCulloc) и Питтса (Pitts) и в чуть более поздней работе Хебба (Hebb) «Организация поведения» («Organization of Behavior») (1949 г.). Интерес к нейронным сетям был потерян после выхода в 1969 г. работы Минского (Minsky) и Пейперта (Papert) «Перцептроны» («Perceptrons»), в которой была показана ограниченность использования перцептронов, простейших структур нейронных сетей, не способных решить даже задачу «исключающего или». Интерес к нейронным сетям заново возник в 80-е годы XX века, чему способствовали следующие события:

— выход работы Хопфилда (Hopfield) по математическим основам динамических нейронных сетей (1982 г.);

— разработка Кохоненом (Kohonen) самоорганизующихся сетей, обучающихся без учителя (1984 г.);

— представление Румельхартом (Rumelhart) и МакКлеландом (McCleland) алгоритма обратного распространения ошибки для обучения многослойных нейронных сетей прямого распространения (1986 г.).

Данная монография посвящена вопросам организации качественного и эффективного построения моделей структуры на основе нейронных сетей прямого распространения. В первой главе рассмотрены теоретические основы построения нейросетевых архитектур, рассмотрена суперпозиционная структура нейронных сетей прямого распространения, приведена методика применения нейронных сетей для решения различных практических задач. Вторая глава посвящена постановке задачи и классификации методов обучения нейронных сетей. В третьей главе приведены результаты разработки метода обучения нейронных сетей на основе операции матричного псевдообращения, позволяющего снизить размерность пространства оптимизируемых весов за счет декомпозиции задачи обучения, показано применение блочных рекуррентно-итерационных процедур в обучении. Четвертая глава посвящена применению методов интервального анализа для гарантированного обучения нейронных сетей, позволяющих гарантировать глобальность искомого оптимума. В пятой главе рассмотрены вопросы построения нейронных сетей оптимальной структуры, основное внимание уделено конструктивному построению, предложено применение операции блочного псевдообращения для повышения эффективности алгоритмов наращивания моделей. В заключении указаны перспективные направления дальнейших исследований в нейросетевом моделировании.

Глава 1

Нейронные сети прямого распространения

1.1. Нейрон

Основным элементом нейронных сетей (НС) является *нейрон*, преобразующий векторный вход $x \in \mathbb{R}^n$ в скалярный выход $y \in \mathbb{R}$. Каждому входному сигналу x_i, $i = 1, \ldots, n$, ставится в соответствие число $w_i \in \mathbb{R}$, называемое *весовым коэффициентом* входа, *весом* или *синапсом* (рис. 1.1).

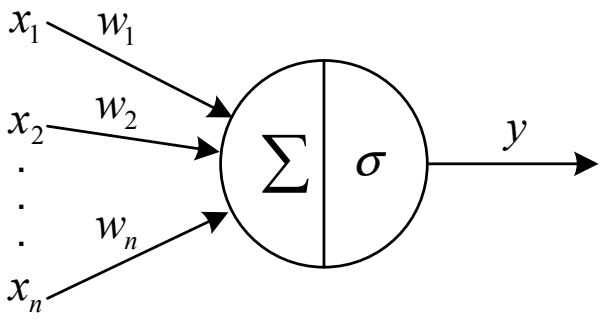


Рис. 1.1. Нейрон

Преобразование нейроном входных данных осуществляется в два этапа:

1. Вычисляется *уровень активации* нейрона net по формуле

$$net = \langle w, x \rangle = \sum_{i=1}^{n} w_i x_i, \tag{1.1}$$

где $w = \begin{bmatrix} w_1 & w_2 \ldots & w_n \end{bmatrix}^T \in \mathbb{R}^n$ — вектор весов; $x = \begin{bmatrix} x_1 & x_2 \ldots & x_n \end{bmatrix}^T \in \mathbb{R}^n$ — входной вектор; $\langle \cdot, \cdot \rangle$ — скалярное произведение векторов.

2. К значению уровня активации применяется, как правило, нелинейная *функция активации* $\sigma : \mathbb{R} \to \mathbb{R}$

$$y = \sigma(net). \tag{1.2}$$

Таким образом, нейрон осуществляет отображение (обычно нелинейное) из пространства $\mathbb{R}^n$ в пространство $\mathbb{R}$.

Для удобства работы в вектор входных сигналов вводят дополнительный, фиктивный, постоянный единичный входной сигнал $x_0 = 1$, которому соответствует вес w_0 (рис. 1.2). В результате этого формула вычисления уровня активации нейрона представляется в виде

$$net = \sum_{i=0}^{n} w_i x_i. \tag{1.3}$$

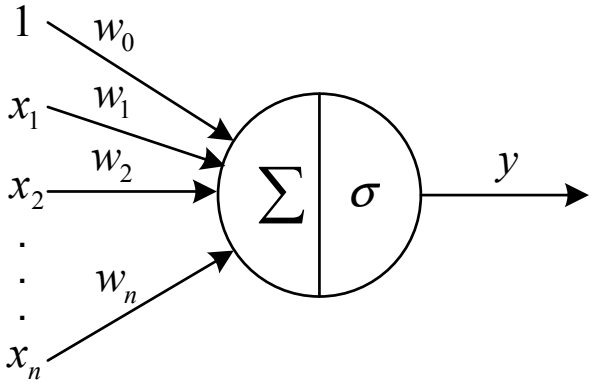


Рис. 1.2. Нейрон с фиктивным единичным входным сигналом

Формула (1.3) является частным случаем формулы вычисления уровня активации в полиномиальном нейроне k-го порядка, который осуществляет следующее преобразование:

$$net = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i_1=1}^{n}\sum_{i_2=1}^{n} w_{i_1 i_2} x_{i_1} x_{i_2} + \ldots + \\ + \sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\ldots\sum_{i_k=1}^{n} w_{i_1 i_2 \ldots i_k} x_{i_1} x_{i_2} \ldots x_{i_k}. \tag{1.4}$$

Вычисление уровня активации возможно еще одним способом:

$$net = \frac{\sum\limits_{i=0}^{n} w_i x_i}{\sum\limits_{i=0}^{n} \widetilde{w}_i x_i}, \tag{1.5}$$

где $\widetilde{w}_i$, $i = 0, \ldots, n$, также являются весами нейрона. Нейрон, использующий функцию (1.5), называется Паде-нейроном, так как аппроксимация с применением дробно-рациональных функций носит имя Паде. На практике нейроны, в которых уровень активации рассчитывается по формулам (1.4) и (1.5), практически не используются, так как они усложняют процесс построения нейросетевой модели, не внося значительных дополнительных возможностей.

В нейросетевых моделях (НСМ) в качестве функции активации $\sigma(net)$ обычно используется одна из следующих [21]:

— Униполярная сигмоидная логистическая (рис. 1.3):

$$\sigma(net) = \frac{1}{1 + e^{-net}}. \tag{1.6}$$

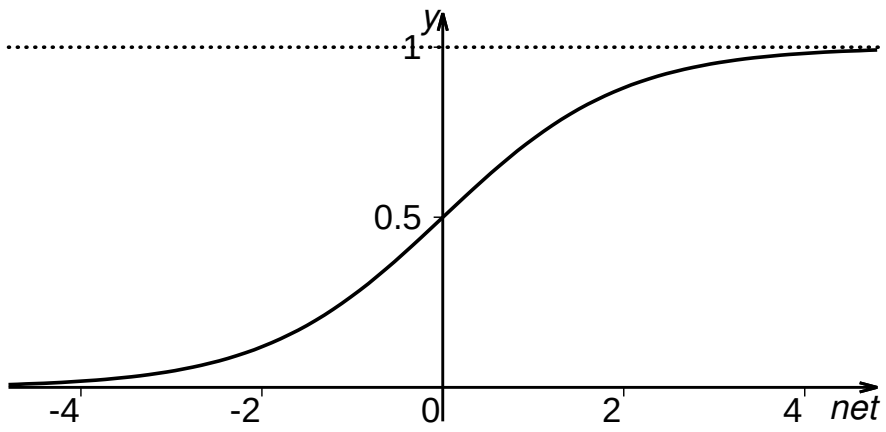


Рис. 1.3. Сигмоидная логистическая функция

— Биполярная сигмоидная логистическая – гиперболический тангенс (рис. 1.4):

$$\sigma(net) = th(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}}. \tag{1.7}$$

Сигмоидные логистические функции дифференцируемы на всей своей области определения – вещественной оси. Их производные по аргументу легко выражаются через значения самих функций:

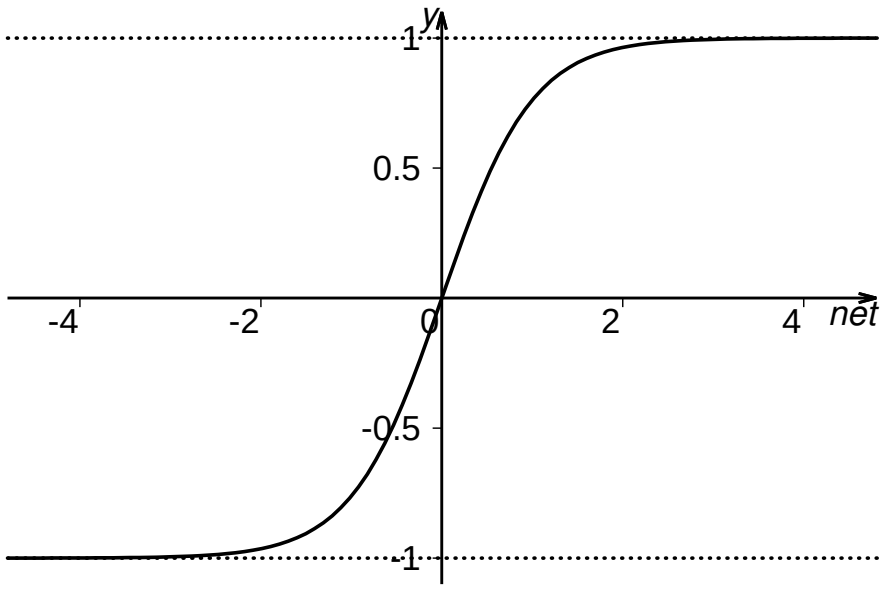


Рис. 1.4. Гиперболический тангенс

— Производная униполярной сигмоидной логистической функции (1.6):

$$\sigma(net)'_{net} = \sigma(net)(1 - \sigma(net)).$$

— Производная гиперболического тангенса (1.7):

$$\sigma(net)'_{net} = 1 + \sigma(net)^2.$$

Чаще предпочтение отдают функции (1.6), так как для ее вычисления требуется меньшее количество операций (для гиперболического тангенса кроме вычисления e^{-net} требуется находить еще и e^{net}).

Функции активации обладают свойством насыщения: при больших по модулю уровнях активации нейронов указанные функции дают близкие значения. Чтобы свойство насыщения проявлялось не так сильно, можно использовать параметризованную функцию для (1.6) вида

$$\sigma_\alpha(net) = \frac{1}{1 + e^{-\alpha \cdot net}}, \alpha > 0,$$

где α – априорно задаваемый параметр крутизны (рис. 1.5). При $\alpha \to \infty$ параметризованная сигмоидная логистическая функция стремится к пороговой функции, а при $\alpha \to 0$ – к линейной. Обычно значение α задается из диапазона от $0,1$ до $0,8$.

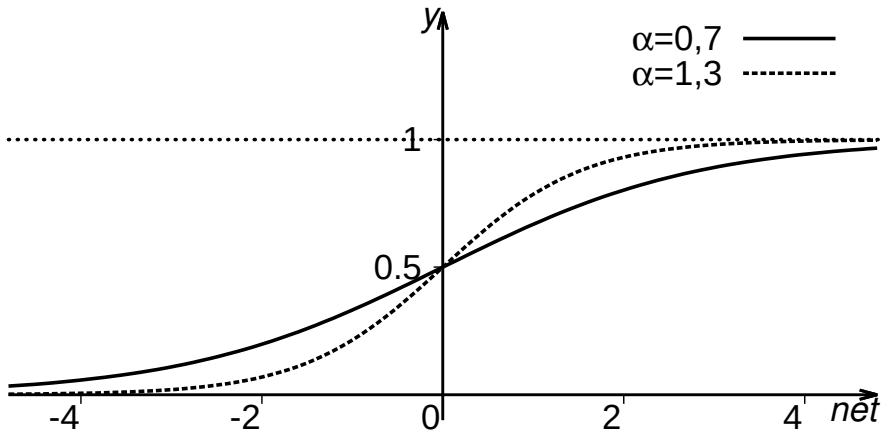


Рис. 1.5. Параметризованная сигмоидная логистическая функция

Области значений рассмотренных функций активации ограничены. Для того чтобы на выходе получать произвольные действительные значения, в некоторых нейронах активационная функция отсутствует, то есть

$$\sigma(net) = net.$$

В этом случае считается, что применяется единичная (тождественная) функция активации.

1.2. Структура нейронных сетей прямого распространения

Совокупность нейронов, состоящих из нескольких слоев и связанных друг с другом таким образом, что выходы нейронов одного слоя передают сигналы на входы нейронов следующего слоя, называется *нейронной сетью прямого распространения* (НС ПР) (рис. 1.6). Нейроны одного слоя не связаны друг с другом, также не связаны нейроны слоев, не являющихся соседними. Входы НС ПР выделяют в отдельный слой – входной. Элементы входного слоя фактически не являются нейронами, так как не преобразуют информацию, а лишь распределяют ее. В связи с этим при подсчете количества слоев НС ПР входной слой обычно (за исключением некоторых работ) не учитывают. Слой, выходы нейронов которого не передаются на входы дру-

гих нейронов, называется выходным – эти выходы образуют выходы всей НС. Все остальные слои называются *скрытыми*, или *промежуточными*. Таким образом, m-слойная НС ПР состоит из одного входного слоя, $(m-1)$ скрытых слоев и одного выходного, m-го, слоя. Вычисление результатов производится сетью последовательно, в направлении от входного слоя к выходному. Такие сети называют также *слоистыми*. Хотя существуют и другие архитектуры нейронных сетей (нейронные сети с радиальными базисными функциями – RBF-сети, саморганизующиеся карты Кохонена – SOM, сети Хопфилда и другие), в данной работе в качестве нейросетевых моделей (НСМ) рассматриваются именно НС ПР. На рис. 1.6 приведена многовыходная НС ПР.

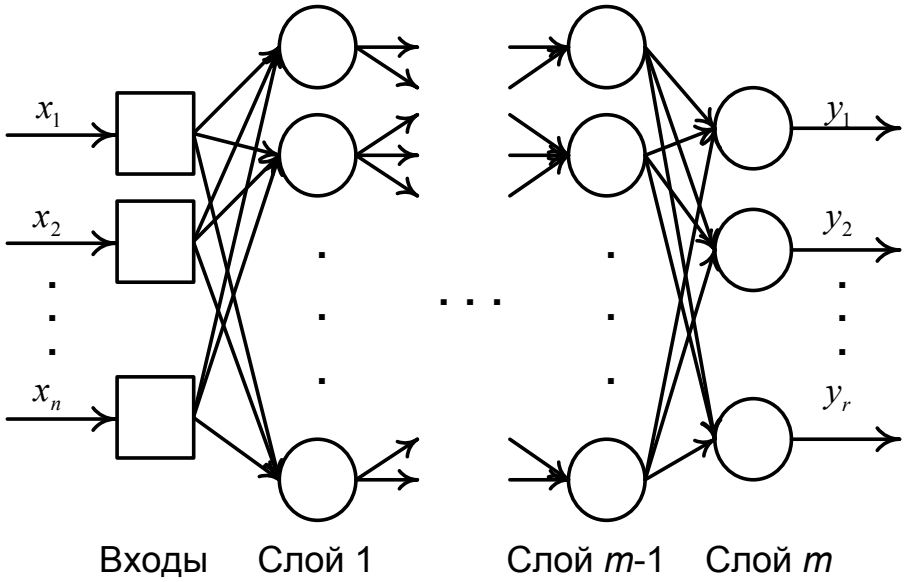


Рис. 1.6. m-слойная нейронная сеть прямого распространения

Пусть НС ПР состоит из m слоев, в l-ом слое которой находится N_l нейронов, $l = 0, \ldots, m$; $N_0 = n$ – количество входов НС ПР. Пусть $w_j^{(l,i)}$ – j-й вес i-го нейрона l-го слоя, $l = 1, \ldots, m$, $i = 1, \ldots, N_l$, $j = 0, \ldots, N_{l-1}$. Функционирование i-го нейрона l-го слоя задается формулой

$$y^{(l,i)} = \sigma^{(l,i)}\left(net^{(l,i)}\right) = \sigma^{(l,i)}\left(\sum_{j=0}^{N_{l-1}} w_j^{(l,i)} y^{(l-1,j)}\right), \qquad (1.8)$$

где $y^{(l-1,0)} = 1$ – фиктивный единичный вход нейрона; $y^{(l-1,j)}$ – выход j-го нейрона $(l-1)$-го слоя, $j = 1, \ldots, N_{l-1}$; $y^{(0,j)} = x_j$ – j-й вход НС ПР, $j = 1, \ldots, n$.

Хотя в общем случае функции активации нейронов могут отличаться друг от друга, обычно для всех нейронов скрытых слоев они берутся одинаковыми, чаще всего – униполярными сигмоидными логистическими. Нейроны выходного слоя в подавляющем большинстве случаев содержат единичную функцию активации для того, чтобы сеть могла выдавать любые действительные выходные значения, а не только ограниченные областью значений функции активации.

Выходы нейронов l-го слоя представляются вектор-столбцом $y^{(l)} \in \mathbb{R}^{N_l}$, веса нейронов этого слоя – матрицей $W^{(l)} \in \mathbb{R}^{N_l \times (1+N_{l-1})}$, в которой i-я строка состоит из весов нейрона (l, i). Матрица $W^{(l)}$ представляет собой блочную матрицу $W^{(l)} = [w_0^{(l)} \mid W_1^{(l)}]$, где $w_0^{(l)} \in \mathbb{R}^{N_l}$ – веса нейронов, соответствующие фиктивным единичным входам; $W_1^{(l)} \in \mathbb{R}^{N_l \times N_{l-1}}$ – матрица весов между нейронами $(l-1)$-го и l-го слоев. На основе введенных обозначений функционирование l-го слоя НСМ в векторно-матричной форме описывается следующей формулой:

$$y^{(l)} = \sigma^{(l)} \left(w_0^{(l)} + W_1^{(l)} y^{(l-1)} \right), \tag{1.9}$$

где $\sigma^{(l)}$ – векторная функция векторного аргумента, соответствующая применению функций активации нейронов к каждому нейрону l-го слоя. На основе данного соотношения функционирование всей НС ПР может быть представлено в виде

$$y = \sigma^{(m)} \left(w_0^{(m)} + W_1^{(m)} \sigma^{(m-1)} \left(\ldots W_1^{(2)} \sigma^{(1)} \left(w_0^{(1)} + W_1^{(1)} x \right) \ldots \right) \right), \tag{1.10}$$

где $x \in \mathbb{R}^n$ – вектор входов; $y \in \mathbb{R}^r$ – вектор выходов НСМ (выходы нейронов выходного слоя). В случае применения единичной функции активации в выходных нейронах и одинаковой функции активации во всех скрытых нейронах отображение (1.10) упрощается и принимает вид

$$y = w_0^{(m)} + W_1^{(m)} \sigma \left(\ldots + W_1^{(2)} \sigma \left(w_0^{(1)} + W_1^{(1)} x \right) \ldots \right). \tag{1.11}$$

Из формул (1.10) и (1.11) видно, что НСМ реализует сложную нелинейную функциональную зависимость между входными и выходными величинами. НС ПР имеет характерную суперпозиционную линейно-нелинейную

по весам структуру. Линейно-нелинейный характер означает, что веса состоят из двух частей – линейно входящих $W^{(m)}$ и нелинейно входящих (веса скрытых нейронов). Более того, эта «линейно-нелинейность» характерна для каждого слоя, а в контексте всей НСМ она, как слои гамбургера, суперпозиционным образом накладывается друг на друга. Указанная специфика НС ПР, в особенности ее суперпозиционный характер, лежит в основе многих алгоритмов обучения.

Многослойные сети не приводят к увеличению вычислительной мощности по сравнению с однослойными, если нейроны скрытых слоев обладают единичными активационными функциями. Действительно, в этом случае вычисление выхода слоя заключается в умножении входного вектора на первую весовую матрицу с последующим умножением результирующего вектора на вторую весовую матрицу и т.д. Это означает, что любая многослойная линейная сеть может быть заменена эквивалентной ей однослойной сетью с соответствующей матрицей весов:

$$y = w_0^{(m)} + W_1^{(m)} \left(\ldots + W^2 \left(w_0^{(1)} + W_1^{(1)} x \right) \ldots \right) = w_0 + Wx.$$

Одновыходной *стандартной НС ПР* (иногда называемой полутораслойным предиктором) называется двухслойная НС ПР с единичной функцией активации у нейронов выходного слоя. Работа одновыходной стандартной НС ПР описывается формулой

$$y = \sum_{i=1}^{q} w_i \sigma \left(w_{i0} + \sum_{j=1}^{n} w_{ij} x_j \right) = \sum_{i=1}^{q} w_i \sigma \left(\sum_{j=0}^{n} w_{ij} x_j \right), \qquad (1.12)$$

где q – количество нейронов единственного скрытого слоя; w_{ij} – вес i-го нейрона скрытого слоя, идущий от j-го входа, $j = 1, \ldots, n$; w_{i0} – вес фиктивного единичного входа; w_i – вес нейрона выходного слоя, идущий от i-го нейрона скрытого слоя. В многовыходной стандартной НС ПР ($r > 1$) соотношение (1.12) описывает работу каждого выхода. Схема многовыходной стандартной НС ПР приведена на рис. 1.7.

Теоретической основой успешного применения НС ПР на практике являются их универсальные аппроксимационные способности. Уже стандарт-

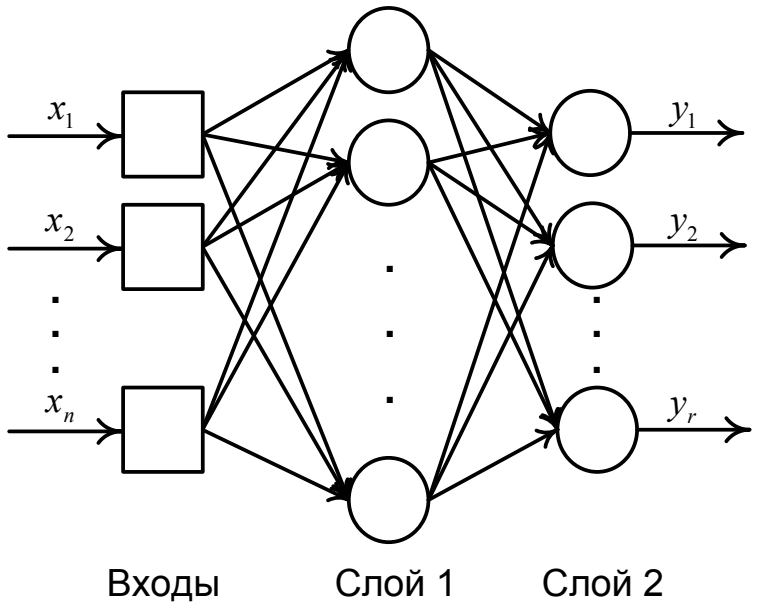


Рис. 1.7. Стандартная нейронная сеть прямого распространения

ные сети являются аппроксиматорами непрерывных функций нескольких переменных, заданных на компактном множестве. Другими словами, они способны приблизить непрерывную функцию с любой заданной точностью. Эти возможности восходят к хорошо известным теоремам из теории аппроксимации (теорема Вейерштрасса о приближении полиномами, теорема Стоуна, результаты В.И. Арнольда и А.Н. Колмогорова). Результаты исследований универсальных аппроксимационных свойств НС ПР были получены в 1989 г. одновременно несколькими авторами. Результат, приведенный Фунахаши (Funahashi), отражен в следующей теореме [29].

Теорема 1.1. (Об универсальных аппроксимационных способностях) Пусть $f^*(x_1, x_2, \ldots, x_n)$ – непрерывная функция, определенная на компактном множестве, и $\varepsilon > 0$ – точность аппроксимации. Существует такое число $q \in \mathbb{N}$ и набор чисел $w_{ij}, w_i \in \mathbb{R}$, что функция

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{q} w_i \sigma \left(w_{i0} + \sum_{j=1}^{n} w_{ij} x_j \right),$$

где σ – непостоянная ограниченная монотонно возрастающая непрерывная функция (например, сигмоидная логистическая), приближает исходную функцию $f^*(x_1, x_2, \ldots, x_n)$ с погрешностью, не превышающей ε на всей области

определения, т.е.

$$\sup_{(x_1,x_2,\ldots,x_n)\in\mathbb{R}^n} \mid f(x_1,x_2,\ldots,x_n) - f^*(x_1,x_2,\ldots,x_n) \mid \leqslant \varepsilon.$$

Эта теорема говорит о том, что любую непрерывную функцию нескольких переменных можно с любой точностью приблизить с помощью стандартной НС ПР с достаточным количеством нейронов в скрытом слое. Теорема, однако, не говорит о том, сколько нейронов и какие веса необходимы для аппроскимации функции с заданной точностью.

Хотя отображения (1.10) и (1.12) являются статическими, НС ПР можно применять и для моделирования динамических систем. В таких системах выход зависит от дискретного времени t:

$$y = y[t],$$

при этом вместо (1.10) рассматривается зависимость вида

$$y[t] = f\left(w; x[t], x[t-1], \ldots, x[t-d_1]; y[t], y[t-1], \ldots, y[t-d_2]\right), \qquad (1.13)$$

где d_1 и d_2 – порядки задержки входных и выходных сигналов соответственно. На рис. 1.8 схематично приведена структура динамической модели (1.13). Кружки на рис. 1.8 означают не отдельные нейроны, а их ансамбли, соответствующие нескольким нейронам. Для построения моделей (1.13) используются те же подходы, что и для построения НС ПР статических систем. Дополнительной сложностью является определение порядков d_1 и d_2.

1.3. Методика применения нейронных сетей

На рис. 1.9 схематично представлена методика практического применения НСМ. Рассмотрим подробнее этапы данной методики.

Этап 1. Информативное множество входных $x \in \mathbb{R}^n$ и выходных $y \in \mathbb{R}^r$ величин выбирается, исходя из поставленной задачи. Следует обратить внимание на то, что неосновательный и поспешный выбор входных и выходных переменных отрицательно влияет на качество построения моделей. Данный

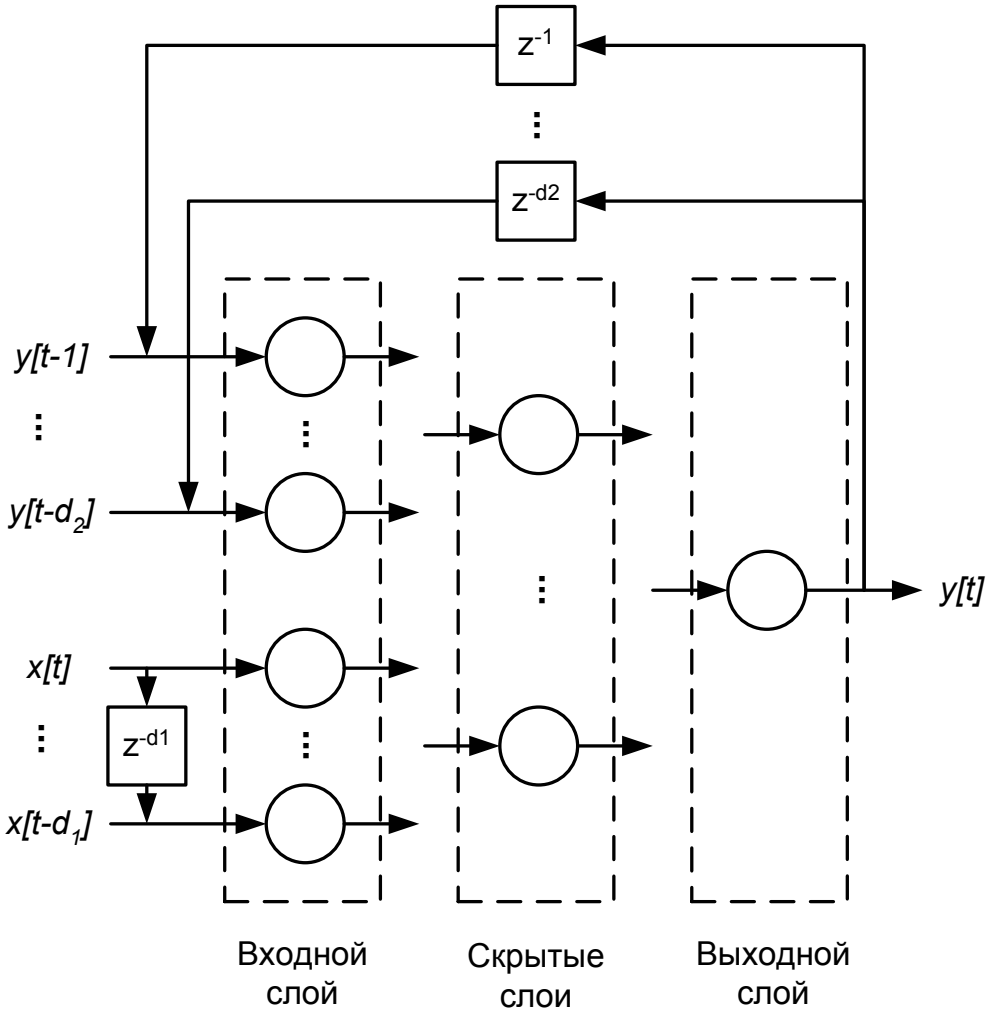


Рис. 1.8. Динамическая НС

процесс выбора неформализуем, он в сильной мере зависит от конкретной задачи, а также от опыта и интуиции эксперта. После определения входов и выходов составляется множество вход-выходных данных, еще не являющееся, однако, обучающим множеством. Эти данные могут либо собираться в ходе проведения экспериментов, либо быть заданными. Множество должно быть репрезентативным (отражать реальное поведение моделируемого объекта). Чем больше объем, тем лучше может быть настроена сеть на решение задач.

Этап 2. С целью дальнейшего использования данных в обучении НС применяется предварительная обработка (предобработка) данных. Можно выделить следующие шаги предобработки информации, которые применяются в зависимости от конкретной задачи:

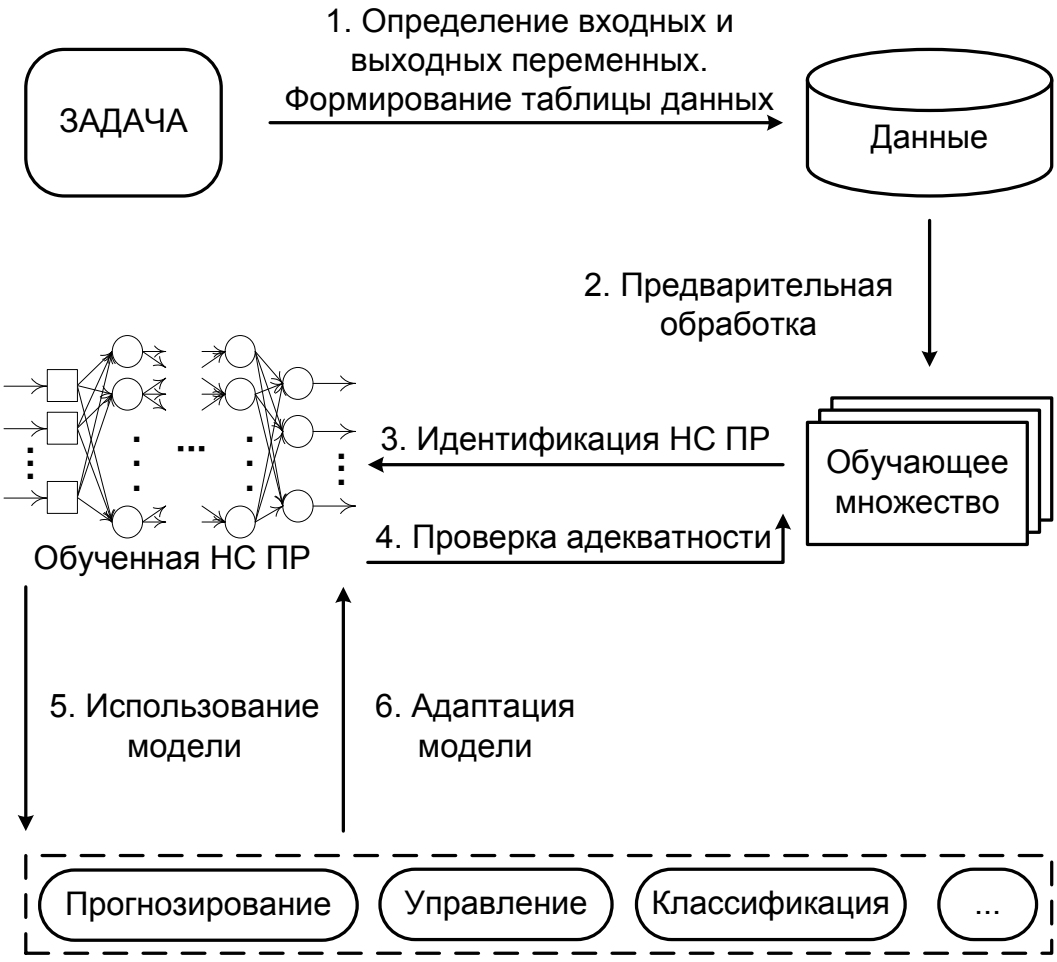


Рис. 1.9. Схема применения НС ПР

1. Преобразование данных, заданных в качественном виде (то есть в виде понятий, вербальных оценок), к числовому виду.
2. Восстановление отсутствующих данных в обучающем множестве (восстановление пробелов).
3. Удаление неестественных, ошибочных, сильно искаженных данных, не отражающих реального поведения моделируемого объекта.
4. Переход от абсолютных значений к относительным.
5. Нормировка значений переменных.

Первые четыре шага актуальны для всех задач моделирования. Восстановление данных, выявление и удаление ошибочной информации обычно производится с применением статистических процедур. Переход к относительным значениям целесообразен при моделировании динамических зависимостей (например, при прогнозировании временных рядов это позволяет

получить более стационарный ряд). Спецификой нейросетевого моделирования является нормировка значений переменных.

Как отмечалось выше, используемые в НС функции активации обладают свойством насыщения: начиная с определенных входных значений, функция будет выдавать практически одно и то же значение. Данное обстоятельство может отрицательно отразиться на обучении НС, так как при начальной инициализации весов даже в небольших пределах (например, значениями из отрезка $[-0,1; 0,1]$) входные значения могут приводить к появлению на выходе неотличимых значений. В связи с этим данные обучающего множества рекомендуется нормализовать, т.е. привести (обычно с помощью линейных преобразований) к некоторому диапазону $[a, b]$. Наиболее часто выбираются диапазоны $[0; 1]$ и $[-1; 1]$. Для каждой j-й входной переменной x_j определяется диапазон изменения значений

$$\left[x_j^{min}, x_j^{max}\right]$$

на основе вход-выходной таблицы данных:

$$x_j^{min} = \min_{i=1}^{k} \widetilde{x}_{ij}$$

и

$$x_j^{max} = \max_{i=1}^{k} \widetilde{x}_{ij},$$

где $\widetilde{x}_{ij}$ – значение j-го входа из i-й строки обучающего множества.

Нормализация значений $\widetilde{x}_{ij}$ в диапазон $[0, 1]$ осуществляется по формуле

$$\frac{\widetilde{x}_{ij} - x_j^{min}}{x_j^{max} - x_j^{min}}, \tag{1.14}$$

а в диапазон $[-1, 1]$ – по формуле

$$2\frac{\widetilde{x}_{ij} - x_j^{min}}{x_j^{max} - x_j^{min}} - 1. \tag{1.15}$$

Заметим, что нормализация выходных значений необязательна.

Этап 3. Центральным этапом решения практических задач с использованием НС ПР является идентификация НСМ. Данная задача, как и вообще

задача построения математической модели в аналитическом виде, состоит из двух компонентов:

1. Структурной идентификации.
2. Параметрической идентификации.

При построении нейросетевых моделей структурная идентификация является частично решенной, так как задана структура формирования выходных значений НС ПР. Остается открытым вопрос о количестве скрытых слоев и количестве нейронов в каждом из них. Эта задача, называемая также задачей определения оптимальной структуры НС ПР, должна решаться таким образом, чтобы НС ПР могла показывать хорошие результаты на данных, не входящих в обучающее множество. Теоретически увеличение количества слоев и нейронов в сети приводит к уменьшению ошибки обучения. Однако это с определенного момента начинает отрицательно сказываться на способности НС ПР описывать зависимость на данных, не включенных в обучающее множество. Возможны три стратегии выбора структуры НС ПР:

— Генерация набора НС, не зависимых друг от друга.
— Контрастирование НС, заключающееся в оценке значимости нейронов в текущей НС и удалении наименее значимых из них.
— Конструктивный подход, заключающийся в последовательном наращивании количества слоев и/или нейронов в скрытых слоях, начиная с сети минимальной структуры.

Каждый из подходов имеет свои преимущества и недостатки. Процедура генерации независимых структур НС проста в реализации, однако может привести к пропуску хорошей структуры. Для осуществления контрастирования необходимо иметь начальную сеть с достаточно большим количеством весов. Конструктивный подход представляется наиболее рациональным.

Параметрическая идентификация НС ПР – это, как указывалось выше, и есть обучение. Процесс обучения будет подробно рассмотрен далее.

Этап 4. Проверка адекватности модели очень важна при практическом применении. Основной подход для решения этой задачи состоит в выделении из исходной таблицы вход-выходных данных тестового множества

(около 20%). Остальная часть является обучающим множеством. Тестовое множество состоит из данных, не использующихся в процессе обучения (т.е. фактически в обучающее множество они не входят). Часто производится обучение нескольких НС ПР фиксированной структуры, далее среди которых по ошибке на тестовом множестве выбирается наилучшая. После окончания процесса обучения на обучающем множестве рассчитывается *ошибка обучения*, а на тестовом множестве – *ошибка обобщения*. Более адекватной считается та НС ПР, которая показывает меньшую ошибку обобщения (ошибка обучения может быть больше, чем для других структур).

Этап 5. Широкие возможности применения НС при решении широкого круга задач, особенно часто встречающихся в экономической и технической сферах деятельности, опираются на их способность строить зависимости произвольной нелинейной сложности. Можно выделить следующие основные способы применения НС ПР:

— Прогнозирование – получение выходных величин для заданных входных значений.

— Управление – формирование управляющих сигналов с целью получения желаемых выходов управляемой системы.

— Классификация – соотнесение входного вектора, описывающего некоторый объект, к определенному классу объектов.

Многие другие задачи могут быть сведены к одному из рассмотренных способов использования НС ПР.

Этап 6. При использовании НС ПР с течением времени встает необходимость учета новой информации о предметной области, рассматриваемой в задаче, например связанная с поступлением новой вход-выходной информации. Процесс построения новой адекватной НСМ очень трудоемок, поэтому перспективным является подход, заключающийся в подстройке используемой модели (обычно только значений весов сети).

Глава 2

Классификация методов обучения нейронных сетей

2.1. Постановка задачи обучения

Обучением НС называется процесс определения значений весов НС фиксированной структуры на основе набора известных вход-выходных данных. Эти данные задаются таблицей, называемой *обучающим множеством*, в которой отражается зависимость значений выходов от комбинации значений входных величин. Строки обучающего множества называются *примерами*, а значения выходов – *указаниями учителя*. Состоящее из k примеров обучающее множество, включающее n входов и r выходов, имеет вид:

$$\left[\begin{array}{cccc|cccc} \widetilde{x}_{11} & \widetilde{x}_{12} & \dots & \widetilde{x}_{1n} & \widetilde{y}_{11} & \widetilde{y}_{12} & \dots & \widetilde{y}_{1r} \\ \widetilde{x}_{21} & \widetilde{x}_{22} & \dots & \widetilde{x}_{2n} & \widetilde{y}_{21} & \widetilde{y}_{22} & \dots & \widetilde{y}_{2r} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \widetilde{x}_{k1} & \widetilde{x}_{k2} & \dots & \widetilde{x}_{kn} & \widetilde{y}_{k1} & \widetilde{y}_{k2} & \dots & \widetilde{y}_{kr} \end{array}\right].$$

В кратком виде обучающее множество представляется так:

$$\{\widetilde{x}_i, \widetilde{y}_i\}, i = 1, \dots, k,$$

где $\widetilde{x}_i \in \mathbb{R}^n$ и $\widetilde{y}_i \in \mathbb{R}^r$ – соответственно входной вектор и вектор указаний учителя из i-го примера. Если задан вектор весов $w \in \mathbb{R}^s$, то при подаче входного вектора НС выдаст набор r выходных значений. Степень близости вектора выходов НС на i-м примере $y_i(w)$ и указаний учителя $\widetilde{y}_i$ характеризуется мгновенным функционалом качества

$$Q(\varepsilon_i(w)) = \varepsilon_i^T(w) V \varepsilon_i(w), \tag{2.1}$$

где $\varepsilon_i(w) = y_i(w) - \widetilde{y}_i$ – вектор отклонений выходов сети от указаний учителя, $V \in \mathbb{R}^{r \times r}$ – положительно определенная матрица, задающая взвешенную норму вектора $\varepsilon_i(w)$. Обычно в роли матрицы V выступает единичная, поэтому функционал представляет собой евклидову норму вектора отклонений

$$Q_i(\varepsilon(w)) = \varepsilon_i^T(w)\varepsilon_i(w) = (y_i(w) - \widetilde{y}_i)^T(y_i(w) - \widetilde{y}_i) = \sum_{j=1}^{r}(y_{ij}(w) - \widetilde{y}_{ij})^2.$$

Интегральная степень соответствия (по всем примерам) нейросетевой модели данным из обучающего множества задается функционалом

$$J(w) = \frac{1}{2}\sum_{i=1}^{k} Q_i(w) = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{r}(y_{ij}(w) - \widetilde{y}_{ij})^2. \tag{2.2}$$

Множитель $\frac{1}{2}$ введен для удобства преобразований, на вектор оптимальных весов он не влияет.

Для случая с одним выходом ($r = 1$) функционал (2.2) упрощается и принимает следующий вид:

$$J(w) = \frac{1}{2}\sum_{i=1}^{k} Q_i(w) = \frac{1}{2}\sum_{i=1}^{k}(y_i(w) - \widetilde{y}_i)^2 = \frac{1}{2}\|y(w) - \widetilde{y}\|^2, \tag{2.3}$$

где $y(w)$ и $\widetilde{y}$ – соответственно вектор выходов НС ПР на обучающем множестве и вектор указаний учителя; $\|\cdot\|$ – евклидова норма вектора. Цель обучения НС – определение такого вектора весов w^*, чтобы функционал (2.2) или (2.3) принимал минимальное значение:

$$w^* = \arg\min_{w \in \mathbb{R}^s} J(w). \tag{2.4}$$

Фактически обучение НС – многоэкстремальная задача параметрической идентификации нелинейной модели, которая может осуществляться с помощью обширного аппарата теории оптимизации.

2.2. Классификация методов обучения нейронных сетей

Задача обучения НС (2.4) является нелинейной задачей о наименьших квадратах (НЗНК). Сущность оптимизации (в случае НС – минимизации) заключается в построении релаксационной последовательности векторов $w_0, w_1, \ldots$, на каждой итерации (шаге) уменьшающих значение целевой функции ($J(w_0) > J(w_1) > \ldots$). Процесс построения такой последовательности прекращается на t-й итерации в случае удовлетворения одного или нескольких критериев:

1. $\mid J(w_{t+1}) - J(w_t) \mid \leqslant \varepsilon_1$;
2. $\|w_{t+1} - w_t\| \leqslant \varepsilon_2$;
3. $\|\nabla_w J(w_t)\| \leqslant \varepsilon_3$,

где $\varepsilon_i, i = 1, 2, 3$, – предварительно задаваемые точности оптимизации. Наиболее часто используется 1-й критерий останова (практически перестает изменяться значение целевой функции), 2-й – реже всех (практически перестает изменяться вектор). Последний, 3-й, критерий может применяться только в случае дифференцируемых функций (он означает, что при данном векторе невозможно отыскать направление уменьшения значения целевой функции).

Для решения НЗНК, как правило, используются два типа методов оптимизации:

1. Общие, т.е. предназначенные для класса дифференцируемых функций.
2. Специальные, т.е. учитывающие квадратичный характер минимизируемого функционала.

Многовыходная НС ПР (r выходов) может быть заменена совокупностью r одновыходных НС, поэтому рассмотрим методы обучения лишь для случая одновыходных НС ПР ($r = 1$). Основная схема методов оптимизации представлена на рис. 2.1. Основными моментами при реализации алгоритма являются определение направления минимизации на текущем шаге и определение длины шага вдоль этого направления. Способ выбора направления дает название самому оптимизационному методу. Выбор длины шага – как

правило, одномерная минимизация функции вдоль выбранного направления.

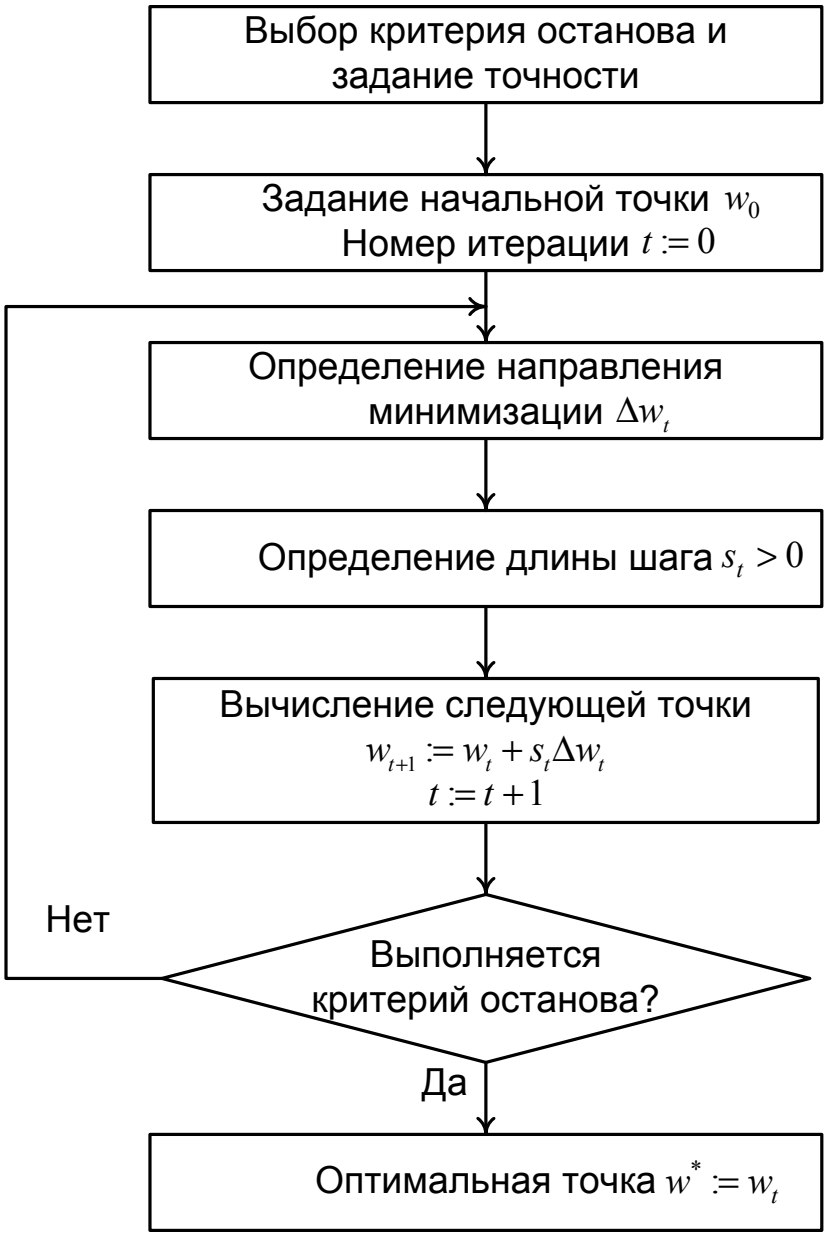


Рис. 2.1. Схема оптимизации функций

Направление оптимизации определяется исходя из поведения функции в окрестности текущей точки w. Эта информация заложена, прежде всего, в градиенте целевой функции по вектору оптимизируемых параметров. Аналогично тому, что производная показывает направление возрастания функции от одной переменной, градиент показывает направление возрастания функции от нескольких переменных.

2.2.1. Процедура обратного распространения ошибки

Эффективным способом нахождения градиента минимизируемого функционала (2.3) по вектору весов, учитывающим суперпозиционный характер сети на основе формулы производной сложной функции, является процедура *обратного распространения ошибки (ОРО)* (error backpropagation), предложенная Румельхартом. Вследствие аддитивности производной

$$\nabla_w J(w) = \nabla_w \left(\sum_{i=1}^{k} Q_i(w) \right) = \sum_{i=1}^{k} \nabla_w Q_i(w)$$

достаточно вывести формулу вычисления градиента по ошибке на одном примере обучающего множества (для мгновенного функционала качества), обозначив для упрощения записи $Q_i(w)$ через $Q(w)$. Градиент $\nabla_w Q(w)$ состоит из частных производных $Q(w)$ по весам нейронов $w_j^{(m,i)}$, которые, исходя из (1.8), рассчитываются по формуле

$$\frac{\partial Q(w)}{\partial w_j^{(m,i)}} = \frac{\partial Q(w)}{\partial y^{(m,i)}} \cdot \frac{\partial y^{(m,i)}}{\partial net^{(m,i)}} \cdot \frac{\partial net^{(m,i)}}{\partial w_j^{(m,i)}}, \tag{2.5}$$

где последние два множителя определяются легко

$$\frac{\partial y^{(m,i)}}{\partial net^{(m,i)}} = \sigma'_{net}(net^{(m,i)}), \tag{2.6}$$

$$\frac{\partial net^{(m,i)}}{\partial w_j^{(m,i)}} = y^{(m-1,j)}. \tag{2.7}$$

Для нахождения первого множителя формулы (2.5) обозначим его как

$$s^{(m,i)} = \frac{\partial Q(w)}{\partial y^{(m,i)}}.$$

Нахождение $s^{(m,i)}$ опирается на рекуррентную процедуру, которая получается из формулы производной сложной функции от нескольких переменных:

$$s^{(m,i)} = \sum_{j=1}^{N_{m+1}} \frac{\partial Q}{\partial y^{(m+1,j)}} \frac{\partial y^{(m+1,j)}}{\partial y^{(m,i)}} = \sum_{j=1}^{N_{m+1}} s^{(m+1,j)} \sigma'_{net}(net^{(m+1,j)}) w_i^{(m+1,j)}. \tag{2.8}$$

Для вычислений по формуле (2.8) необходимо начальное условие. Оно получается из формулы (2.1) с учетом множителя $\frac{1}{2}$ из функционала (2.2):

$$s^{(M,1)} = \frac{\partial Q(w)}{\partial y^{(M,1)}} = \varepsilon(w) = y(w) - \widetilde{y}, \tag{2.9}$$

где $\widetilde{y}$ – указание учителя для поданного примера. Ошибка работы сети $\varepsilon(w)$ словно распространяется в направлении, обратном функционированию сети. Это и обусловило такое название метода нахождения градиента (обратное распространение ошибки).

Пример 2.1

Рассмотрим стандартную НС ПР, содержащую один вход, один выход и два нейрона в скрытом слое. На рис. 2.2 приведена структура этой сети, обозначены входы и выходы каждого нейрона, а также веса нейронов. Внутри каждого нейрона указан его номер. Например, обозначение $(1, 2)$ показывает, что это 2-й нейрон 1-го слоя.

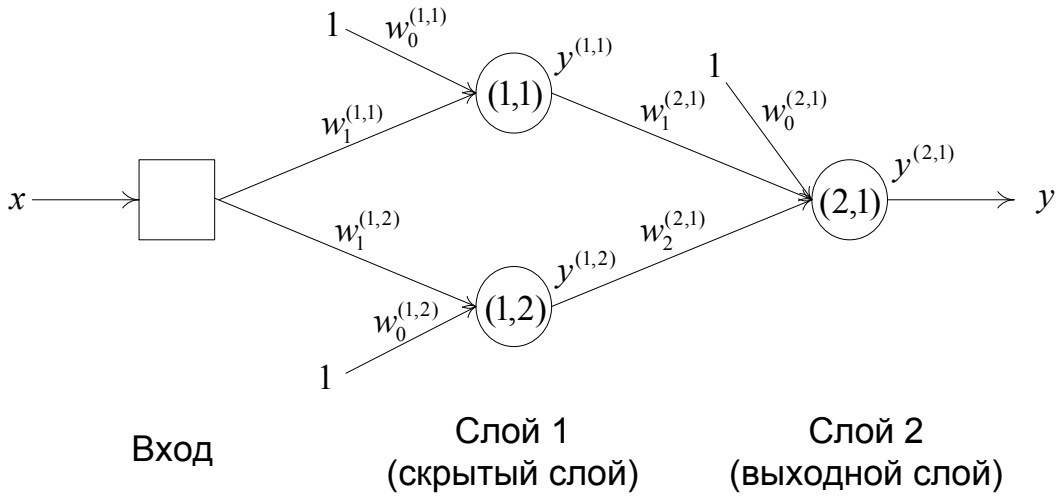


Рис. 2.2. Стандартная нейронная сеть с одним входом, одним выходом и двумя нейронами в скрытом слое

Распишем для этой сети формулы для вычисления градиента по методу

ОРО. Вектор весов состоит из семи элементов:

$$w = \begin{bmatrix} w_0^{(1,1)} \\ w_1^{(1,1)} \\ w_0^{(1,2)} \\ w_1^{(1,2)} \\ w_0^{(2,1)} \\ w_1^{(2,1)} \\ w_2^{(2,1)} \end{bmatrix}.$$

Используя начальное условие (2.9), получаем для нейрона выходного слоя

$$s^{(2,1)} = \frac{\partial Q(w)}{\partial y^{(2,1)}} = y(w) - \widetilde{y}.$$

Нейрон выходного слоя имеет единичную функцию активации, поэтому

$$y^{(2,1)} = w_0^{(2,1)} + w_1^{(2,1)} y^{(1,1)} + w_2^{(2,1)} y^{(1,2)}.$$

Тогда из (2.8) получаем следующие выражения для нейронов скрытого слоя:

$$s^{(1,1)} = s^{(2,1)} \frac{\partial y^{(2,1)}}{\partial y^{(1,1)}} = s^{(2,1)} w_1^{(2,1)};$$

$$s^{(1,2)} = s^{(2,1)} \frac{\partial y^{(2,1)}}{\partial y^{(1,2)}} = s^{(2,1)} w_2^{(2,1)}.$$

Теперь можно записать формулу для вычисления частных производных, используя формулу (2.5). Например, для веса $w_0^{(1,1)}$ в предположении, что используется в качестве функции активации нейронов скрытого слоя используется униполярная сигмоидная логистическая функция, получаем

$$\frac{\partial Q(w)}{\partial w_0^{(1,1)}} = s^{(1,1)} \cdot \frac{\partial y^{(1,i)}}{\partial net^{(1,i)}} \cdot \frac{\partial net^{(1,1)}}{\partial w_0^{(1,1)}} = s^{(1,1)} \sigma(net^{(1,1)})(1 - \sigma(net^{(1,1)})) y^{(0,0)} =$$

$$= s^{(1,1)} y^{(1,1)} (1 - y^{(1,1)}).$$

Здесь учтено, что $y^{(0,0)} = 1$. Для веса $w_1^{(1,1)}$: $y^{(0,1)} = x$, поэтому

$$\frac{\partial Q(w)}{\partial w_1^{(1,1)}} = s^{(1,1)} \sigma(net^{(1,1)})(1 - \sigma(net^{(1,1)})) y^{(0,1)} = s^{(1,1)} y^{(1,1)} (1 - y^{(1,1)}) x.$$

Аналогично получаются и остальные формулы. В итоге градиент мгновенного функционала качества будет получаться следующим образом:

$$\nabla_w Q(w) = \begin{bmatrix} s^{(1,1)}y^{(1,1)}(1-y^{(1,1)}) \\ s^{(1,1)}y^{(1,1)}(1-y^{(1,1)})x \\ s^{(1,2)}y^{(1,2)}(1-y^{(1,2)}) \\ s^{(1,2)}y^{(1,2)}(1-y^{(1,2)})x \\ s^{(2,1)}y^{(2,1)}(1-y^{(2,1)}) \\ s^{(2,1)}y^{(2,1)}(1-y^{(2,1)})y^{(1,1)} \\ s^{(2,1)}y^{(2,1)}(1-y^{(2,1)})y^{(1,2)} \end{bmatrix}.$$

2.2.2. Обучение на основе локальных методов безусловной оптимизации дифференцируемых функций

Заменим (аппроксимируем) функционал (2.3) в окрестности текущего вектора весов w_t квадратичной моделью

$$J(w) \approx J(w_t) + \nabla_w J(w_t)(w - w_t) + \frac{1}{2}(w - w_t)^T \nabla_w^2 J(w_t)(w - w_t) \qquad (2.10)$$

или линейной

$$J(w) \approx J(w_t) + \nabla_w J(w_t)(w - w_t), \qquad (2.11)$$

где $\nabla_w J(w_t)$ – градиент функции по вектору w в точке w_t, $\nabla_w^2 J(w_t)$ – матрица Гессе в точке w_t. В зависимости от того, какая модель используется, выделяют следующие методы:

— градиентные (первого порядка), использующие градиент функции;

— ньютоновского типа (второго порядка), использующие градиент функции и матрицу Гессе;

— квазиньютоновского типа (методы переменной метрики, МПМ), использующие градиент и приближения матрицы Гессе.

Зная градиент целевой функции, можно использовать метод оптимизации дифференцируемой функции, называемый методом наискорейшего спус-

ка. В этом методе функция минимизируются вдоль направления Δw_t, противоположного градиенту

$$\Delta w_t = -\frac{\nabla_w J(w_t)}{\|\nabla_w J(w_t)\|}.$$

Длина шага определяется на основе одномерной минимизации

$$s_t = \arg \min_{s_t \in \mathbb{R}, s_t > 0} J(w_t + s_t \Delta w_t).$$

Данный метод обычно показывает невысокую скорость сходимости. Следует заметить, что в некоторой литературе под ОРО понимается метод наискорейшего спуска, а не только способ вычисления градиента НС ПР, учитывающий ее суперпозиционную структуру.

Ускорение сходимости может быть достигнуто за счет использования информации о локальном поведении функции на предыдущей итерации. Распространенными методами данного типа являются методы сопряженных градиентов Флэтчера-Ривса и Полака-Рибьера. Выбор направления в этих методах осуществляется таким способом:

1. На начальном этапе ($t = 0$):

$$\Delta w_0 = -\nabla_w J(w_0).$$

2. На t-й итерации по методу Флэтчера-Ривса вычисляется значение

$$\beta_t = \frac{\|\nabla_w J(w_t)\|^2}{\|\nabla_w J(w_{t-1})\|^2},$$

а по методу Полака-Рибьера –

$$\beta_t = \frac{\nabla_w J(w_t) \left(\nabla_w J(w_t) - \nabla_w J(w_{t-1})\right)}{\|\nabla_w J(w_{t-1})\|^2}.$$

Направление минимизации находится по формуле

$$\Delta w_t = -\nabla_w J(w_t) + \beta_t \Delta w_{t-1}.$$

Эти методы эффективны и довольно просты в реализации.

Метод Ньютона основывается на расчете направления по формуле

$$\Delta w_t = -H(w_t)^{-1}\nabla_w J(w_t),$$

где $H(w_t) = \nabla_w^2 J(w_t)$ – матрица Гессе функционала качества. Сложность практического применения данного метода состоит в том, что вторые производные функционала часто сложно или совсем невозможно находить. Поэтому прибегают к использованию аппроксимаций матрицы Гессе, лежащих в основе квазиньютоновских методов оптимизации.

В квазиньютоновских методах на каждой итерации строятся положительно определенные (матрица Гессе является таковой) матрицы H_t на основе рекуррентного соотношения $H_{t+1} = H_t + \Delta H_t$ (аналогичным образом строится и аппроксимация обратной матрицы Гессе). Способ определения матрицы ΔH_t определяет название метода.

Формула пересчета обратной матрицы Гессе $W_t = H_t^{-1}$ по методу BFGS имеет вид

$$W_{t+1} = W_t + \frac{(s_t - W_t g_t)\, s_t^T + s_t\,(s_t - W_t g_t)^T}{g_t^T s_t} - \frac{(s_t - W_t g_t)\, g_t s_t s_t^T}{\left(g_t^T s_t\right)^2},$$

где $s_t = w_t - w_{t-1}$; $g_t = \nabla_w J(w_t) - \nabla_w J(w_{t-1})$. На начальной итерации $W_0 = I$ – направление оптимизации совпадает с антиградиентом. Пересчет по методу DFP опирается на формулу

$$W_{t+1} = W_t + \frac{s_t s_t^T}{s_t^T g_t} + \frac{W_t g_t g_t^T W_t}{g_t^T W_t g_t}.$$

2.2.3. Обучение на основе методов решения нелинейных задач о наименьших квадратах

Учет квадратичности минимизируемого функционала приводит к разработке алгоритмов, ориентированных на решение НЗНК. Применяя к квадратичной аппроксимации функционала (2.10) необходимое условие оптимума функции, приходим к уравнению

$$\nabla J(w_t) + \nabla^2 J(w_t)(w - w_t) = 0.$$

Для оптимизации данной модели может быть использован метод Ньютона с псевдообращением

$$\Delta w_t = -\left[\nabla^2 J(w_t)\right]^+ \nabla J(w_t), \tag{2.12}$$

где $\left[\nabla^2 J(w_t)\right]^+$ – псевдообратная матрица к исходной (матрица Мура-Пенроуза), являющаяся обобщением обратной матрицы на случай вырожденных и прямоугольных матриц (краткие сведения о псевдообратных матрицах см. в приложении). Использование псевдообращения позволяет не заботиться о невырожденности матрицы Гессе оптимизируемой функции. Минимизируемый функционал (2.3) для НЗНК можно представить в форме

$$J(w) = \frac{1}{2} R(w)^T R(w), \tag{2.13}$$

где $R(w) = y(w) - \widetilde{y}$ – вектор невязок; $y(w)$ – вектор выходов НС ПР, составленный на примерах обучающего множества; $\widetilde{y}$ – вектор указаний учителя. Тогда будут справедливы следующие формулы:

$$\nabla J(w) = R'^T(w) R(w), \tag{2.14}$$

где $R'^T(w)$ – матрица Якоби вектора невязок, и

$$\nabla^2 J(w) = R'^T(w) R'(w) + G(w), \tag{2.15}$$

где $G(w)$ – матрица, содержащая информацию о вторых частных производных элементов вектора $R(w)$. Более точно:

$$G(w) = \sum_{i=1}^{k} J_i(w) G_i(w),$$

где $J_i(w)$ – i-й элемент вектора $J(w)$, а $G_i(w)$ – матрица Гессе для элемента $J_i(w)$. Подставляя формулы (2.14) и (2.15) в (2.12), получаем ньютоновское направление минимизации с псевдообращением:

$$\Delta w_t = -\left[R'^T(w_t) R'(w_t) + Q(w_t)\right]^+ R'^T(w_t) R(w_t). \tag{2.16}$$

Матрицы $G_i(w)$ сложно рассчитываются, поэтому формула (2.16) в чистом виде не применяется.

В основе большинства алгоритмов решения НЗНК лежит предположение о том, что с каждой итерацией слагаемое $R'^T(w_t)R'(w_t)$ становится все более значимым по сравнению с $G(w_t)$. Действительно, при $\|J(w_t)\|_{t\to\infty} \to 0$ матрица $G(w_t)$ стремится к нулевой. В методе Гаусса-Ньютона с псевдообращением полагается, что $G(w) = 0$, поэтому

$$\Delta w_t = -\left[R'^T(w_t)R'(w_t)\right]^+ R'^T(w_t)R(w_t) = -\left[R'(w_t)\right]^+ R(w_t). \tag{2.17}$$

Классический метод Гаусса-Ньютона получается при полном столбцовом ранге матрицы $R'(w_t)$; в этом случае

$$\left[R'(w_t)\right]^+ = \left[R'^T(w_t)R'(w_t)\right]^{-1} R'^T(w_t)$$

и направление определяется по формуле

$$\Delta w_t = -\left[R'^T(w_t)R'(w_t)\right]^{-1} R'^T(w_t)R(w_t).$$

Если матрица $G(w_t)$ аппроксимируется на каждом шаге диагональной матрицей $\mu_t I$, $\mu_t \geqslant 0$, получается метод Левенберга-Марквардта

$$\Delta w_t = -\left[R'^T(w_t)R'(w_t) + \mu_t I\right]^+ R'^T(w_t)R(w_t). \tag{2.18}$$

При этом шаг спуска вдоль найденного направления всегда является единичным, а параметр μ_t выбирается так, чтобы минимизировать целевую функцию. В отличие от других метод Левенберга-Марквардта не попадает под рассматриваемую схему оптимизации; он относится к классу методов доверительной окрестности. При $\mu_t \to 0$ направление почти совпадает с направлением метода Гаусса-Ньютона, при $\mu \to \infty$ – с направлением антиградиента.

В задачах с большим значением целевой функции (с большой невязкой) целесообразно использовать квазиньютоновские приближения H_t матрицы $G(w_t)$. Если используется формула BFGS для нахождения приближения, то получается следующее:

$$H_{t+1} = H_t - \frac{1}{s_t^T M_t s_t} M_t s_t s_t^T M_t + \frac{1}{g_t^T s_t} g_t g_t^T,$$

где $M_t = R_t'^T R_t' + H_t$, $s_t = w_t - w_{t-1}$ и $g_t = R_t'^T J(w_t) - R_{t-1}'^T J(w_{t-1})$. На начальной итерации берется $H = 0$. Периодически для уменьшения влияния погрешностей округлений снова полагают $H = 0$. Если приближение осуществляется на основе DFP-метода, то

$$H_{t+1} = H_t + \frac{(\widetilde{g}_t - H_t s_t)\, g_t^T + g_t\, (\widetilde{g}_t - H_t s_t)^T}{g_t^T s_t} - \frac{(\widetilde{g}_t - H_t s_t)^T\, s_t g_t g_t^T}{\left(g_t^T s_t\right)^2},$$

где $\widetilde{g}_t = R_t'^T J(w_t) - R_{t-1}'^T J(w_t)$. Следует отметить, что специальные методы решения НЗНК не всегда работают лучше методов оптимизации произвольных дифференцируемых функций.

Расчет производных выходов сети по весам НС ПР возможен с помощью процедуры, учитывающей суперпозиционный характер структуры сети аналогично методу ОРО. При этом процедура построения матрицы Якоби даже несколько упрощается $\nabla_w^T y(w_t)$ в связи с тем, что подлежит вычислению производная не функционала, а выхода сети.

2.3. Глобальное обучение нейронных сетей

Функционал (2.3) имеет многоэкстремальный характер, поэтому численные методы локальной оптимизации и методы решения НЗНК в чистом виде для обучения НС ПР неэффективны. Большинство подходов глобальной оптимизации используют случайную компоненту [13, 21, 35]. Хотя имеется возможность реализации полностью случайного поиска, метода Монте-Карло, такой подход не применяется вследствие того, что он не может учитывать специфику задачи обучения. В связи с этим методы обучения НС ПР обычно представляют собой синтез детерминированных и случайных способов поиска оптимума. Наиболее популярными являются следующие алгоритмы глобального обучения:

1. Комбинация метода мультистарта и численных методов локальной оптимизации.
2. Комбинация туннельного метода и численных методов локальной оптимизации (метод «встряски» весов).

3. Генетические алгоритмы.
4. Метод имитации отжига.
5. Интервальные методы гарантированной оптимизации.

Метод мультистарта на базе метода Монте-Карло. Из множества X случайно или детерминировано выбирается некоторое подмножество из N точек. Последовательно из каждой точки запускается численный метод локальной оптимизации, и из полученного множества локальных решений выбирается наилучшее. В чистом виде метод мультистарта не является эффективным, так как одно и то же локальное решение может быть найдено не один раз. Мультистарт – обобщенный подход: большинство эффективных методов глобальной оптимизации основано на идее метода мультистарта – запуска стандартных локальных алгоритмов из множества точек, равномерно распределенных на множестве X.

Каждый шаг туннельного метода состоит из фазы минимизации, на которой улучшается значение текущего результата, и фазы туннелирования, на которой находится точка из допустимого множества, отличная от последней, где найдено значение минимума. Полученная на фазе туннелирования точка рассматривается как стартовая для очередного цикла. Другое название метода – метод «встряски». По сути, это внесение небольшой случайной добавки: $\widehat{w}_j^{(l,i)} = w_j^{(l,i)} + s_j^{(l,i)}$, где $s_j^{(l,i)}$ – некоторая случайная величина, равномерно распределенная на отрезке $[a, b]$. Обычно выбираются небольшие значения a и b, например, $a = -0,1$; $b = 0,1$.

Генетические алгоритмы предназначены для глобальной оптимизации широкого класса функций [8, 16]. В них любая альтернатива решения – значения вектора весов НС ПР – представляется в виде битовой строки фиксированной длины. Для поиска используется несколько точек пространства одновременно, что позволяет снизить вероятность попадания целевой функции в локальный минимум. Для работы генетических алгоритмов требуется только информация о целевой функции и области допустимых значений. Выполнение условий дифференцируемости и даже непрерывности оптимизируемой функции не требуется, поэтому генетические алгоритмы могут при-

меняться как для оптимизации непрерывных, так и для оптимизации дискретных, а также дискретно-непрерывных функций. Для перехода от одних точек к другим используются как вероятностные, так и детерминированные правила перехода.

В методе имитации отжига целевая функция является аналогом равновесия термодинамической системы и видоизменяется путем добавления случайных величин (условий температурного режима). Процесс повторяется достаточное число раз для каждой температуры, после чего температура понижается и весь процесс происходит снова до состояния полной заморозки. Избегание попадания в незначительные локальные минимумы (замерзание) зависит от «схемы обжига», выбора начальной температуры, количества итераций для каждой температуры и того, насколько уменьшается температура на каждом шаге процесса «охлаждения».

Все перечисленные подходы к глобальной оптимизации обладают существенным недостатком – они не могут гарантировать глобальность найденного решения. В отличие от них интервальные методы гарантированной оптимизации позволяют обеспечить достоверность результата.

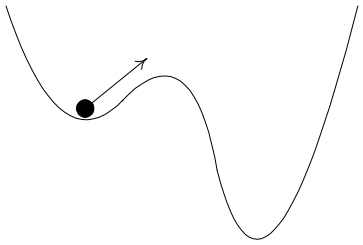


Рис. 2.3. Иллюстрация механизма встряски

Несмотря на наличие случайного компонента, качество и эффективность обучения НС ПР с использованием механизма встряски во многом определяется используемым локальным методом обучения.

Глава 3

Методы обучения нейронных сетей на основе матричного псевдообращения

3.1. Обучение на основе декомпозиции вектора весов с использованием псевдообращения

3.1.1. Псевдообратная матрица и ее свойства

Понятие *псевдообратной матрицы* является обобщением понятия обратной матрицы на случай прямоугольных матриц [1]. Пусть $A \in \mathbb{R}^{k \times n}$ – исходная матрица, тогда $A^+ \in \mathbb{R}^{n \times k}$ называется псевдообратной (или матрицей Мура-Пенроуза) к A, если выполняются четыре условия (условия Мура-Пенроуза):

1. $AA^+A = A$;
2. $A^+AA^+ = A^+$;
3. $(AA^+)^T = AA^+$;
4. $(A^+A)^T = A^+A$.

Заметим, что если матрица удовлетворяет только первым двум условиям, то это определяет более широкое понятие *обобщенно обратной* или $G-$матрицы.

Имеют место следующие важные свойства:

1. Псевдообратная матрица существует для любой матрицы A.
2. Псевдообратная матрица единственна.
3. Для квадратной невырожденной матрицы псевдообратная матрица совпадает с обратной: $A^+ = A^{-1}$.

Для нахождения псевдообратной матрицы могут быть использованы, например, такие способы:

1. Если A – матрица полного столбцового ранга, т.е. все столбцы являются линейно независимыми, то $A^+ = \left(A^T A\right)^{-1} A^T$.
2. Если A – матрица полного строкового ранга, то $A^+ = \left(AA^T\right)^{-1} A$.

К универсальным способам нахождения псевдообратной матрицы относятся рекуррентные алгоритмы Гревиля и Фадеева. В данной работе приведем алгоритм Гревиля для псевдообращения матриц.

Пусть дана матрица $A \in \mathbb{R}^{m \times n}$ и a_k – ее k-й столбец, $k = 1, \ldots, n$. Пусть A_k – матрица, составленная из k первых столбцов матрицы A:

$$A_k = \begin{bmatrix} a_1 & a_2 & \ldots & a_k \end{bmatrix}.$$

При $k = 1$: $A_1 = a_1$, а при $k = 2, \ldots, n$: $A_k = \begin{bmatrix} A_{k-1} & a_k \end{bmatrix}$; $A_n = A$. Матрица $A^+ \in \mathbb{R}^{n \times m}$ может быть вычислена с помощью рекуррентного алгоритма 3.1.

Алгоритм 3.1. Псевдообращение матрицы по Гревилю

1. Инициализация.
$$A_1^+ = \begin{cases} 0, & \text{если } a_1 = 0, \\ \dfrac{a_1^T}{\|a_1\|^2}, & \text{иначе,} \end{cases}$$
где $\|\cdot\|$ – евклидова норма вектора.
2. Цикл по $k = 2, \ldots, n$.
$$A_k^+ = \begin{bmatrix} A_{k-1}^+ (I - a_k f_k) \\ f_k \end{bmatrix},$$
где I – единичная матрица порядка m,
$$f_k = \begin{cases} \dfrac{c_k^T}{\|c_k\|^2}, \quad c_k = (I - A_{k-1} A_{k-1}^+) a_k, & c_k \neq 0, \\ \dfrac{a_k^T (A_{k-1}^+)^T A_{k-1}^+}{1 + \|A_{k-1}^+ a_k\|^2}, & c_k = 0. \end{cases}$$

Полученная на последнем шаге матрица A_n^+ и есть искомая матрица A^+.

Простейшей НС ПР является НС, не содержащая скрытых слоев, с единичной функции активации (рис. 3.1). Функция $y = \sum_{i=1}^{n} w_n x_n$, реализуемая данной сетью, представляет собой линейную по весам функцию.

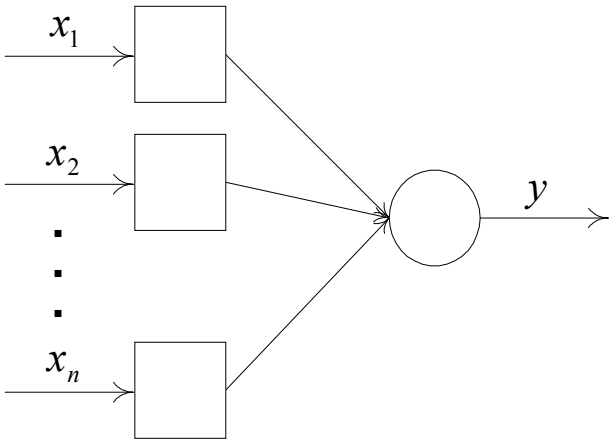


Рис. 3.1. Простейшая нейронная сеть

Ее обучение может быть произведено за один шаг и аналитически представлено в виде

$$w = \widetilde{X}^{+}\widetilde{y}, \tag{3.1}$$

где $\widetilde{X} \in \mathbb{R}^{k\times n}$ – матрица, составленная из входов обучающего множества:

$$\widetilde{X} = \begin{bmatrix} \widetilde{x}_{11} & \widetilde{x}_{12} & \dots & \widetilde{x}_{1n} \\ \widetilde{x}_{21} & \widetilde{x}_{22} & \dots & \widetilde{x}_{2n} \\ \dots & \dots & \dots & \dots \\ \widetilde{x}_{k1} & \widetilde{x}_{k2} & \dots & \widetilde{x}_{kn} \end{bmatrix}.$$

Формула (3.1) является псевдорешением уравнения

$$\widetilde{X}w = \widetilde{y}.$$

Перейдя к рассмотрению стандартной НС ПР, заметим, что ее функционирование можно представить формулой

$$y = \Psi(v)u,$$

где $\Psi(v)$ – матрица выходов нейронов скрытого слоя, зависящая от вектора нелинейно входящих весов v (веса матрицы $W^{(1)}$); u – вектор линейно входящих весов, состоящий из весов нейрона выходного слоя ($W^{(2)}$). Таким образом, вектор весов w разбит на два подвектора – v и u.

Для фиксированного вектора v аналогично (3.1) следует справедливость формулы

$$u = \Psi(v)^{+}\widetilde{y}. \tag{3.2}$$

Формулу (3.2) будем называть базовым линейно-нелинейным соотношением (ЛНС), связывающим линейно и нелинейно входящие в НС веса. Она позволяет находить оптимальный вектор u при заданном v, однако, как (3.1), полностью решить задачу обучения НС ПР не в состоянии. Для вектора v формулы, аналогичной (3.1), нет, так как неизвестно, какие значения должны выдавать нейроны скрытого слоя.

Можно записать, что НС ПР реализует функцию

$$y(u, v; x) = \sum_{j=1}^{q} u_j \psi_j(v, x), \tag{3.3}$$

где $\psi_j(v, x)$ – выход j-го нейрона скрытого слоя при входном векторе сети x. Здесь $u \in \mathbb{R}^q$, $v \in \mathbb{R}^p$, $(p + q)$ – суммарное количество весов НС. Функционал (2.3) можно переписать в следующем виде:

$$J(u, v) = \sum_{i=1}^{k} \sum_{j=1}^{q} \left(y(u, v; \widetilde{x}_i) - \widetilde{y}_i\right)^2 = \sum_{i=1}^{k} \sum_{j=1}^{q} \left(u_j \psi_j(v, \widetilde{x}_i) - \widetilde{y}_i\right)^2. \tag{3.4}$$

Функция (3.3) может быть представлена в векторно-матричной форме

$$y(u, v; x) = \psi(v, x)^T u, \tag{3.5}$$

где $\psi(v, x) = \begin{bmatrix} \psi_1(v, x) & \psi_2(v, x) & \dots & \psi_q(v, x) \end{bmatrix}^T \in \mathbb{R}^q$. На обучающем множестве рассматриваемая функция $\psi(v, x)$ образует матрицу $\Psi(v) \in \mathbb{R}^{k \times q}$:

$$\Psi = \begin{bmatrix} \psi_1(v, x_1) & \psi_2(v, x_1) & \dots & \psi_q(v, x_1) \\ \psi_1(v, x_2) & \psi_2(v, x_2) & \dots & \psi_q(v, x_2) \\ \dots & \dots & \dots & \dots \\ \psi_1(v, x_p) & \psi_2(v, x_p) & \dots & \psi_q(v, x_p) \end{bmatrix}.$$

Для НС ПР матрица $\Psi(v)$ состоит из выходов нейронов скрытого слоя на обучающем множестве.

С учетом этого задача оптимизации функционала (2.3) (или, что то же самое, (3.4)) может быть записана как

$$J(u, v) = \frac{1}{2} \|\Psi(v)u - \widetilde{y}\|^2 \to \min. \tag{3.6}$$

Из ЛНС (3.2) следует, что НЗНК (3.6) эквивалентна задаче

$$\widehat{J}(v) = \|\Psi(v)\Psi(v)^{+}\widetilde{y} - \widetilde{y}\|^2 \to \min\,. \tag{3.7}$$

Преимущество минимизации функционала в задаче (3.7) состоит в том, что оптимизация производится лишь по нелинейно входящим параметрам v_i, линейные определяются на основе соотношения (3.2). Недостатком является то, что снижение размерности задачи приводит к необходимости оптимизации более сложной функции. Данная задача впервые была поставлена Голубом (Golub) и Перейрой (Pereyra) в 1973 г.

Для вывода метода обучения НС на основе декомпозиции вектора весов с псевдообращением введем обозначение

$$H(v) = \Psi(v)\Psi(v)^{+}\widetilde{y} - \widetilde{y} = (\Psi(v)\Psi(v)^{+} - I_k)\widetilde{y}, \tag{3.8}$$

где I_k – единичная матрица порядка k. Основной задачей является определение матрицы Якоби $H'(v)$ по вектору нелинейно входящих весов v. На основе матрицы $H'(v)$ можно реализовывать алгоритмы обучения НС, базирующиеся на методах решения НЗНК. Соотношение

$$\nabla\widehat{J}(v) = H'(v)^T H(v) \tag{3.9}$$

позволяет использовать другие алгоритмы оптимизации, использующие информацию о градиенте минимизируемой функции $\nabla\widehat{J}(v)$ по вектору v.

Производная матрицы $A(B) \in \mathbb{R}^{m\times n}$ по матрице $B \in \mathbb{R}^{k\times p}$ определяется следующим образом (для упрощения записи аргумент в дальнейшем будем опускать):

$$A'_B = \frac{\partial A}{\partial B} = \begin{bmatrix} \dfrac{\partial A}{\partial b_{11}} & \dfrac{\partial A}{\partial b_{12}} & \cdots & \dfrac{\partial A}{\partial b_{1p}} \\ \dfrac{\partial A}{\partial b_{21}} & \dfrac{\partial A}{\partial b_{22}} & \cdots & \dfrac{\partial A}{\partial b_{2p}} \\ \vdots & \vdots & \vdots & \vdots \\ \dfrac{\partial A}{\partial b_{k1}} & \dfrac{\partial A}{\partial b_{k2}} & \cdots & \dfrac{\partial A}{\partial b_{kp}} \end{bmatrix} \in \mathbb{R}^{mk\times np}.$$

Матрицу Якоби $A'_v = A'$ по вектору $v \in \mathbb{R}^p$ будем рассчитывать как производную матрицы по вектор-строке v^T:

$$A' = \frac{\partial A}{\partial v^T} = \begin{bmatrix} \dfrac{\partial A}{\partial v_1} & \dfrac{\partial A}{\partial v_2} & \dots & \dfrac{\partial A}{\partial v_p} \end{bmatrix} \in \mathbb{R}^{m \times np}. \tag{3.10}$$

Такой способ вычисления связан с определением матрицы Якоби как матрицы, составленной из транспонированных градиентов.

В общем случае для производной произведения AB двух матриц $A \in \mathbb{R}^{m \times n}$ и $C \in \mathbb{R}^{n \times l}$ по третьей $C \in \mathbb{R}^{k \times p}$ справедлива следующая формула:

$$(AC)'_B = \frac{\partial (AC)}{\partial B} = \frac{\partial A}{\partial B}(I_p \otimes C) + (I_k \otimes A)\frac{\partial C}{\partial B} = A'(I_p \otimes C) + (I_k \otimes A)C',$$

где $\otimes$ – символ тензорного (кронекеровского) произведения матриц. Тензорное произведение определяется как блочная матрица следующего вида:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mk \times np}.$$

Если дифференцирование ведется по вектор-строке $v^T \in \mathbb{R}^{1 \times p}$, то производная произведения матриц вычисляется как

$$(AC)'_v = \frac{\partial (AC)}{\partial v^T} = \frac{\partial A}{\partial v^T}(I_p \otimes C) + \frac{\partial C}{\partial v^T} = A'(I_p \otimes C) + C'. \tag{3.11}$$

Пример 1.1

Проверим справедливость формулы (3.11). Пусть даны зависящие от $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$ векторы $a = \begin{bmatrix} a_1(v) & a_2(v) \end{bmatrix}$ и $c = \begin{bmatrix} c_1(v) \\ c_2(v) \end{bmatrix}$. Прямое вычисление $(ac)'_v = \dfrac{\partial (ac)}{\partial v^T}$ дает следующий вектор:

$$(ac)'_v = (a_1c_1 + a_2c_2)'_v = \begin{bmatrix} (a_1c_1 + a_2c_2)'_{v_1} & (a_1c_1 + a_2c_2)'_{v_2} \end{bmatrix} =$$

$$= \begin{bmatrix} a_1{}'_{v_1}c_1 + a_1c_1{}'_{v_1} + a_2{}'_{v_1}c_2 + a_2c_2{}'_{v_1} & a_1{}'_{v_2}c_1 + a_1c_1{}'_{v_2} + a_2{}'_{v_2}c_2 + a_2c_2{}'_{v_2} \end{bmatrix},$$

а применение формулы (3.11) приводит к

$$(ac)'_v = \begin{bmatrix} a_1{}'_{v_1} & a_2{}'_{v_1} & a_1{}'_{v_2} & a_2{}'_{v_2} \end{bmatrix} \begin{bmatrix} c_1 & 0 \\ c_2 & 0 \\ 0 & c_1 \\ 0 & c_2 \end{bmatrix} + \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} c_1{}'_{v_1} & c_1{}'_{v_2} \\ c_2{}'_{v_1} & c_2{}'_{v_2} \end{bmatrix} =$$

$$= \begin{bmatrix} a_1{}'_{v_1} c_1 + a_2{}'_{v_1} c_2 & a_1{}'_{v_2} c_1 + a_2{}'_{v_2} c_2 \end{bmatrix} + \begin{bmatrix} a_1 c_1{}'_{v_1} + a_2 c_2{}'_{v_1} & a_1 c_1{}'_{v_2} + a_2 c_2{}'_{v_2} \end{bmatrix} =$$

$$= \begin{bmatrix} a_1{}'_{v_1} c_1 + a_2{}'_{v_1} c_2 + a_1 c_1{}'_{v_1} + a_2 c_2{}'_{v_1} & a_1{}'_{v_2} c_1 + a_2{}'_{v_2} c_2 + a_1 c_1{}'_{v_2} + a_2 c_2{}'_{v_2} \end{bmatrix}.$$

Оба способа дают идентичный результат.

3.1.2. Производная псевдообратной матрицы

Для разработки метода обучения найдем способ вычисления производной псевдообратной матрицы $(A^+(v))'$, состоящей из элементов-функций. Получение в общем случае аналитической формулы для вычисления требуемой производной невозможно (прямой способ). Имеется возможность вычисления $(A^+(v))'$ косвенным способом. Вывод формулы для вычисления производной псевдообратной матрицы осуществляется в четыре этапа:

1. Нахождение производной обратной матрицы.
2. Определение производной в случае, когда дана прямоугольная матрица полного столбцового ранга.
3. Получение аналогичной формулы для прямоугольной матрицы полного строкового ранга.
4. Получение искомой формулы на основе скелетного разложения исходной матрицы.

Этап 1

Для квадратной невырожденной матрицы A с учетом (3.11) имеем

$$\left(AA^{-1}\right)' = A'\left(I_p \otimes A^{-1}\right) + A\left(A^{-1}\right)' = I' = 0,$$

откуда следует

$$\left(A^{-1}\right)' = -A^{-1}A'\left(I_p \otimes A^{-1}\right). \tag{3.12}$$

Этап 2

Пусть B – матрица полного столбцового ранга. В этом случае ее псевдообратная выражается как

$$B^+ = \left(B^T B\right)^{-1} B^T. \tag{3.13}$$

Применение формулы (3.11) дает

$$\left(B^+\right)' = \left[(B^T B)^{-1}\right]' \left(I_p \otimes B^T\right) + \left(B^T B\right)^{-1} \left(B^T\right)'.$$

Матрица $B^T B$ – квадратная невырожденная, применение к ней (3.12) приводит к формуле

$$\begin{aligned}\left(B^+\right)' = -(B^T B)^{-1}(B^T B)'(I_p \otimes (B^T B)^{-1})(I_p \otimes B^T) + \left(B^T B\right)^{-1} \left(B^T\right)' = \\ = -(B^T B)^{-1}(B^T)'(I_p \otimes B)(I_p \otimes (B^T B)^{-1})(I_p \otimes B^T) - \\ - (B^T B)^{-1} B^T B'(I_p \otimes (B^T B)^{-1})(I_p \otimes B^T) + (B^T B)^{-1}(B^T)'.\end{aligned}$$

Обычное и тензорное произведения связаны следующим образом:

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD),$$

если матрицы имеют согласованные размеры. Из этой формулы получаем важное для нас следствие

$$(I_p \otimes B)(I_p \otimes D) = (I_p I_p) \otimes (BD) = I_p \otimes (BD), \tag{3.14}$$

и поэтому

$$\begin{aligned}\left(B^+\right)' = -(B^T B)^{-1}(B^T)'(I_p \otimes B(B^T B)^{-1} B^T) - \\ - (B^T B)^{-1} B^T B'(I_p \otimes (B^T B)^{-1} B^T) + (B^T B)^{-1}(B^T)'.\end{aligned}$$

На основе (3.13) и равенства $(B^T B)^{-1} = (B^T B)^+ = B^+ B^{+T}$ получается

$$\left(B^+\right)' = -B^+ B^{+T}(B^T)'(I_p \otimes BB^+) - B^+ B'(I_p \otimes B^+) + B^+ B^{+T}(B^T)'.$$

Группируя первое слагаемое с третьим и считая, что матрица B состоит из k строк, приходим к формуле

$$\left(B^+\right)' = B^+ B^{+T}(B^T)'(I_{kp} - (I_p \otimes BB^+)) - B^+ B'(I_p \otimes B^+).$$

Операция тензорного произведения дистрибутивна относительно сложения матриц, а $I_{kp} = I_p \otimes I_k$, поэтому

$$I_{kp} - (I_p \otimes BB^+) = I_p \otimes I_k - (I_p \otimes BB^+) = I_p \otimes (I_k - BB^+),$$

на основании чего приходим к окончательной формуле для производной псевдообратной матрицы полного столбцового ранга

$$\left(B^+\right)' = B^+B^{+T}(B^T)'(I_p \otimes (I_k - BB^+)) - B^+B'(I_p \otimes B^+). \tag{3.15}$$

Этап 3

Рассмотрим случай с матрицей полного строкового ранга C. Псевдообратная такой матрицы выражается так:

$$C^+ = C^T \left(CC^T\right)^{-1}.$$

Вывод формулы для расчета $(C^+)'$ аналогичен выводу (3.15):

$$\begin{aligned}\left(C^+\right)' &= (C^T)' \left(I_p \otimes (CC^T)^{-1}\right) + C^T \left[(CC^T)^{-1}\right]' = \\ &= (C^T)' \left(I_p \otimes (CC^T)^{-1}\right) - C^T(CC^T)^{-1}(CC^T)' \left(I_p \otimes (CC^T)^{-1}\right) = \\ &= (C^T)' \left(I_p \otimes C^{+T}C^+\right) - C^+C' \left(I_p \otimes C^+\right) - C^+C \left(C^T\right)' \left(I_p \otimes C^{+T}C^+\right),\end{aligned}$$

откуда получается формула

$$\left(C^+\right)' = \left(I_q - C^+C\right)(C^T)' \left(I_p \otimes C^{+T}C^+\right) - C^+C' \left(I_p \otimes C^+\right). \tag{3.16}$$

Здесь q – число столбцов в матрице C, а $(C^T)' \neq (C')^T$. Формулы (3.15) и (3.16) обладают определенной симметрией.

Этап 4

Рассмотрим матрицу $A \in \mathbb{R}^{k\times q}$ произвольного ранга $r \leqslant \min\{k, q\}$. В этом случае существуют матрица полного столбцового ранга $B \in \mathbb{R}^{k\times r}$ и матрица полного строкового ранга $C \in \mathbb{R}^{r\times q}$ такие, что $A = BC$. Такое разложение матрицы называется скелетным. При этом справедливо равенство

$$A^+ = C^+B^+. \tag{3.17}$$

Учитывая формулы (3.11), (3.15), (3.16), (3.14), а также некоторые другие свойства псевдообратных матриц, получаем формулу для вычисления производной псевдообратной матрицы произвольного ранга:

$$\left(A^{+}\right)' = (I_q - A^{+}A)(A^T)'(I_p \otimes A^{+T}A^{+}) + \\ + A^{+}A^{+T}(A^T)'(I_p \otimes (I_k - AA^{+})) - A^{+}A'(I_p \otimes A^{+}). \quad (3.18)$$

Здесь $A' \in \mathbb{R}^{k \times qp}$ и $(A^T)' \in \mathbb{R}^{q \times kp}$. Из анализа размерностей матриц видно, что, действительно, $(A^{+})' \in \mathbb{R}^{q \times kp}$.

Вернемся к нахождению формулы для вычисления $H'(v)$. Для этого необходимо знать производную не самой матрицы Ψ, а производную произведения $\Psi\Psi^{+}$:

$$\left(\Psi\Psi^{+}\right)' = \Psi'(I_p \otimes \Psi^{+}) + \Psi(\Psi^{+})' = \\ = \Psi'(I_p \otimes \Psi^{+}) + \Psi(I_q - \Psi^{+}\Psi)(\Psi^T)'(I_p \otimes \Psi^{+T}\Psi^{+}) + \\ + \Psi\Psi^{+}\Psi^{+T}(\Psi^T)'(I_p \otimes (I_k - \Psi\Psi^{+})) - \Psi\Psi^{+}\Psi'(I_p \otimes \Psi^{+}).$$

Принимая во внимание, что

$$\Psi\Psi^{+}\Psi^{+T} = (\Psi\Psi^{+})^T\Psi^{+T} = (\Psi^{+}\Psi\Psi^{+})^T = \Psi^{+T},$$

а слагаемое

$$\Psi(I_q - \Psi^{+}\Psi) = \Psi - \Psi\Psi^{+}\Psi = 0,$$

получаем формулу

$$\left(\Psi\Psi^{+}\right)' = \Psi^{+T}(\Psi^T)'(I_p \otimes (I_k - \Psi\Psi^{+})) + (I_k - \Psi\Psi^{+})\Psi'(I_p \otimes \Psi^{+}). \quad (3.19)$$

Размеры матриц согласованы, $(\Psi\Psi^{+})' \in \mathbb{R}^{k \times kp}$.

Формула (3.19) позволяет найти производную (точнее, матрицу Якоби) $H(v)$, определенную по (3.8), по вектору нелинейно входящих весов v:

$$H' = (\Psi\Psi^{+})'(I_p \otimes \widetilde{y}) = \\ = \Psi^{+T}(\Psi^T)'(I_p \otimes (I_k - \Psi\Psi^{+})\widetilde{y}) + (I_k - \Psi\Psi^{+})\Psi'(I_p \otimes \Psi^{+}\widetilde{y}) \in \mathbb{R}^{k \times p}. \quad (3.20)$$

3.1.3. Распространение метода на многослойные сети

Метод обучения НС ПР на основе декомпозиции вектора весов с псевдообращением может быть применен и к многослойным сетям. Функционирование m-слойной НС ПР можно представить в виде

$$y = y(u, v, x) = \sum_{i=1}^{q} u_i y_i^{m-1}(v, x),$$

где $y_i^{m-1}(v, x)$ – выход i-го нейрона последнего скрытого, $(m-1)$-го, слоя. Рассмотренный выше метод обучения одновыходных стандартных НС ПР практически без изменений применяется к многослойным сетям. Изменения касаются только процесса нахождения матрицы Якоби для $\Psi(v)$ (и соответственно $\Psi^T(v)$) по вектору весов v. Для этого применяется способ, аналогичный процедуре ОРО и связанный с учетом суперпозиционного характера НС ПР. Требующая вычисления производная $\dfrac{\partial y^{(m-1,l)}}{\partial w_j^{(h,i)}}$, $l = 1, \ldots, N_{m-1}$, $h = 1, \ldots, m-2$, $i = 1, \ldots, N_h$, определяется как

$$\frac{\partial y^{(m-1,l)}}{\partial w_j^{(h,i)}} = \frac{\partial y^{(m-1,l)}}{\partial y^{(h,i)}} \cdot \frac{\partial y^{(h,i)}}{\partial net_j^{(h,i)}} \cdot \frac{\partial net_j^{(h,i)}}{\partial w_j^{(h,i)}};$$

$$s^{(l,h,i)} = \frac{\partial y^{(m-1,i)}}{\partial y^{(h,i)}} = \sum_{d=1}^{N_{h+1}} \frac{\partial y^{(m-1,l)}}{\partial y^{(h+1,d)}} \cdot \frac{\partial y^{(h+1,d)}}{\partial y^{(h,i)}} = \sum_{d=1}^{N_{h+1}} s^{(l,h+1,d)} \frac{\partial y^{(h+1,d)}}{\partial y^{(h,i)}}.$$

Начальное условие получается следующим образом:

$$s^{(l,m-1,i)} = \frac{\partial y^{(m-1,l)}}{\partial y^{(m-1,i)}} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}.$$

При реализации численного метода, основанного на декомпозиции вектора весов и псевдообращении, для корректировки весов необходимо знать матрицы $\Psi(v_c)$, $\Psi'(v_c)$ и $(\Psi^T(v_c))'$, где v_c – текущий вектор нелинейно входящих весов. $\Psi(v_c)$ – это матрица, каждая строка которой представляет вектор выходов нейронов скрытого слоя, полученный при входном векторе данной обучающей пары. У $\Psi'(v_c)$ каждая строка представляет собой составной вектор – производные выходов нейронов скрытого слоя по всем весам скрытого

слоя:

$$\Psi'(v) =$$

$$= \begin{bmatrix} \begin{bmatrix} f'(v_1,x_1) & 0 & \dots & 0 \\ f'(v_1,x_2) & 0 & \dots & 0 \end{bmatrix} & \begin{bmatrix} 0 & f'(v_2,x_1) & \dots & 0 \\ 0 & f'(v_2,x_2) & \dots & 0 \end{bmatrix} & \dots & \begin{bmatrix} 0 & 0 & \dots & f'(v_l,x_1) \\ 0 & 0 & \dots & f'(v_l,x_2) \end{bmatrix} \\ \dots & \dots & & \dots \\ \begin{bmatrix} f'(v_1,x_k) & 0 & \dots & 0 \end{bmatrix} & \begin{bmatrix} 0 & f'(v_2,x_k) & \dots & 0 \end{bmatrix} & \dots & \begin{bmatrix} 0 & 0 & \dots & f'(v_l,x_k) \end{bmatrix} \end{bmatrix}.$$

У $(\Psi^T(v_c))'$ эти элементарные блоки составляют не строку, а записываются в матрицу:

$$(\Psi^T(v))' =$$

$$= \begin{bmatrix} \begin{bmatrix} f'(v_1,x_1) & 0 & \dots & 0 \\ 0 & f'(v_2,x_1) & \dots & 0 \end{bmatrix} & \begin{bmatrix} f'(v_1,x_2) & 0 & \dots & 0 \\ 0 & f'(v_2,x_2) & \dots & 0 \end{bmatrix} & \dots & \begin{bmatrix} f'(v_1,x_k) & 0 & \dots & 0 \\ 0 & f'(v_2,x_k) & \dots & 0 \end{bmatrix} \\ \dots & \dots & & \dots \\ \begin{bmatrix} 0 & 0 & \dots & f'(v_l,x_1) \end{bmatrix} & \begin{bmatrix} 0 & 0 & \dots & f'(v_l,x_2) \end{bmatrix} & \dots & \begin{bmatrix} 0 & 0 & \dots & f'(v_l,x_k) \end{bmatrix} \end{bmatrix}.$$

В данном случае применяется «блочное» транспонирование. Матрицы $\Psi'(v)$ и $(\Psi^T(v))'$ получились весьма разреженными. При компьютерной реализации этот факт был учтен путем реализации специальных процедур умножения на указанные матрицы, чтобы уменьшить объем занимаемой памяти и сократить количество лишних операций.

При реализации разработанного численного метода возникает задача нахождения произведения вида $A(I_p \times B)$. Прямой подход состоит в предварительном определении матрицы $I_p \times B$ с помощью операции тензорного произведения, а затем умножении на нее матрицы A. Однако данный способ неэффективен: вместо него может быть реализован более эффективный способ косвенного вычисления выражения $A(I_p \times B)$ путем перемножения соответствующих блоков матрицы A на матрицу B.

3.1.4. Вычислительные эксперименты

На рис. 3.2 приведены сводные результаты попарного сравнения численных методов обучения на основе анализа качества решения каждой задачи:

традиционный градиентный метода с численным методом обучения на основе ЛНС и градиентного метода и т.п. Из диаграммы на рис. 3.2 видно, что алгоритмические реализации на базе метода, основанного на ЛНС, во многих случаях работают качественнее. Исключение составляют методы BFGS и DFP, предназначенные для решения НЗНК. Причины их превосходства над методами на основе ЛНС состоят, скорее всего, в двойном псевдообращении матриц при расчете направления минимизации и связанными с этим обстоятельством ошибками округления и т.п. Эффект этих ошибок при использовании метода Гаусса-Ньютона не проявляется. Таким образом, алгоритмические реализации разработанного численного метода обучения, основанного на ЛНС, в большинстве случаев показывают лучшие результаты по качеству обучения, чем их традиционные аналоги.

Среднее время работы методов обучения приведено на рис. 3.3. Данные на диаграмме, за исключением метода Гаусса-Ньютона, показывают, что для определения лучшего решения требуется затратить больше времени. Это справедливо и для традиционных методов обучения, так и для численного метода на основе ЛНС.

На рис. 3.4 показаны данные по среднему количеству итераций, затрачиваемых на обучение НС ПР. Этот, а также рис. 3.5 показывают, что численный метод на основе ЛНС использует для обучения значительно меньшее количество итераций, но при этом затрачивает на каждую итерацию большее время. Это свидетельствует о лучшем качестве определения численным методом обучения на основе ЛНС направления минимизации из-за учета линейно-нелинейной структуры. Большие временные затраты на расчет направления являются следствием достаточной сложности метода, основанного на ЛНС.

Результаты анализа вычислительных экспериментов можно сформулировать следующим образом:

— Качество работы алгоритмических реализаций численного метода на основе ЛНС в большинстве случаев выше (за исключением методов DFP и BFGS-методов для решения НЗНК).

— Численный метод обучения НС ПР, основанный на ЛНС, более эффек-

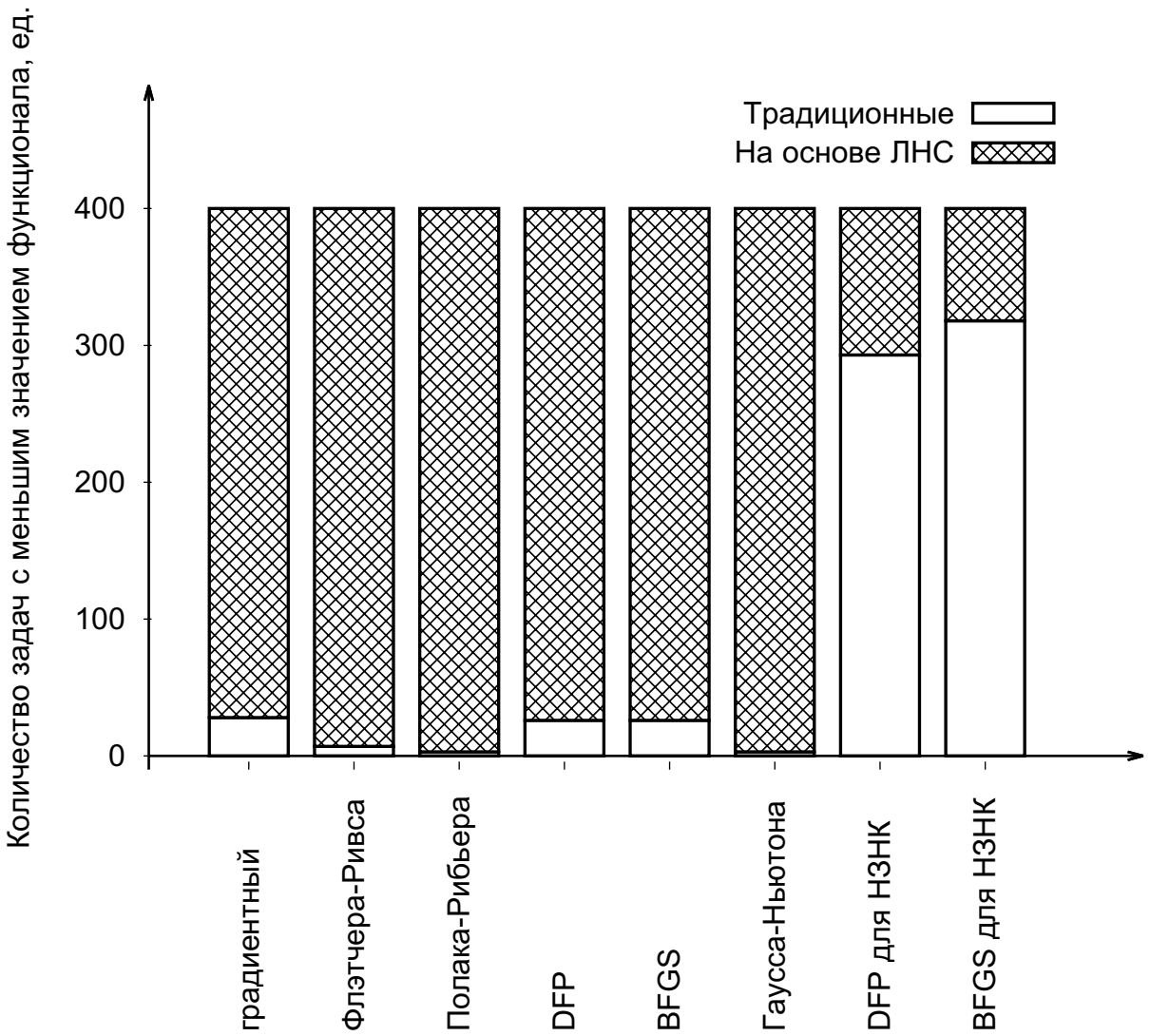


Рис. 3.2. Попарное сравнение алгоритмов по ошибке обучения

тивно реализует переход от итерации к итерации, что свидетельствует о высоком качестве определения направления оптимизации, но при этом затрачивают на одну итерацию больше времени.

— Не прослеживается определенная зависимость качества и эффективности работы численного метода на основе ЛНС по отношению к качеству работы традиционных методов обучения от количества весов НС ПР, объема обучающих множеств. Не наблюдается также монотонная зависимость качества и эффективности работы предлагаемого численного метода от доли количества линейно входящих весов от суммарного количества, хотя в целом можно говорить об увеличении эффективности работы метода при увеличении доли линейно входящих весов.

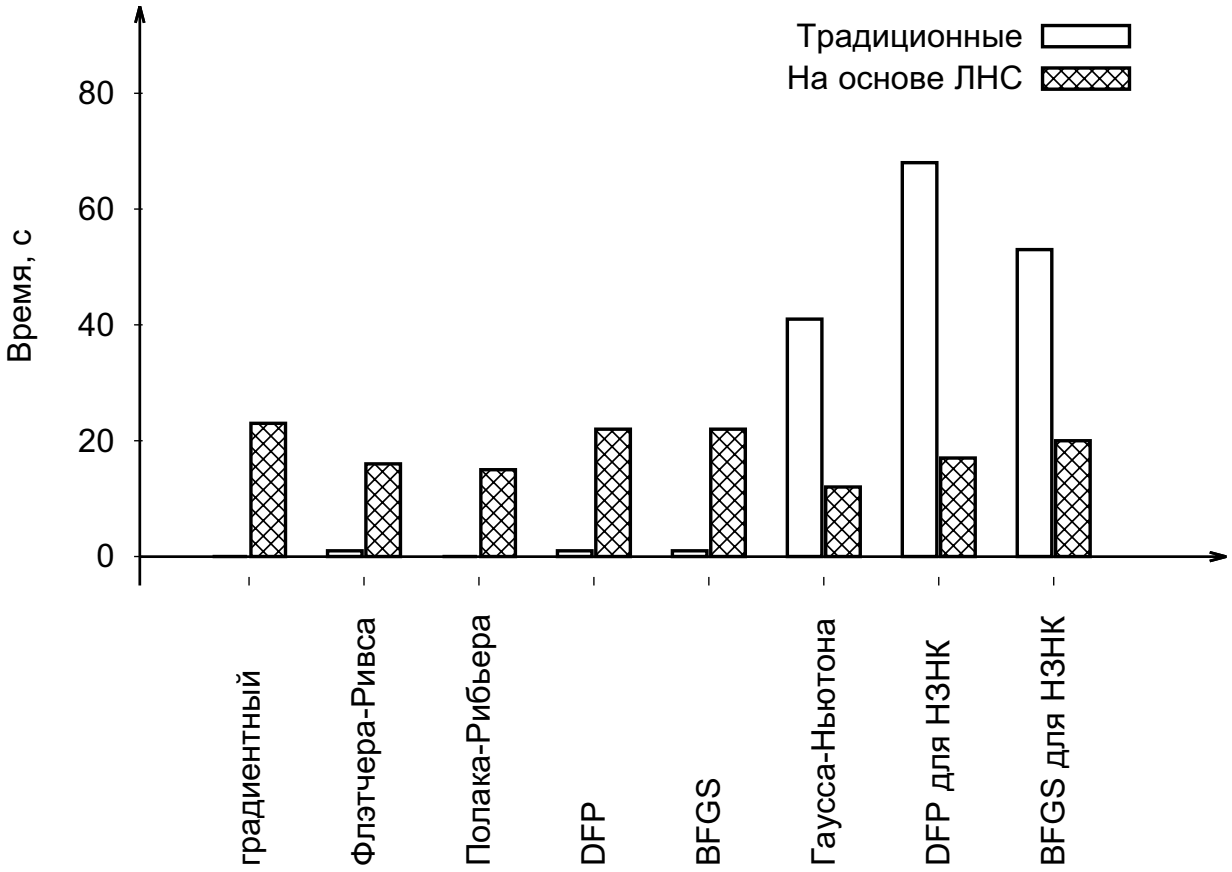


Рис. 3.3. Среднее время обучения

— Время работы алгоритмических реализаций численного метода обучения НС ПР, основанного на ЛНС, значительно возрастает с увеличением объема обучающего множества (свыше 60 примеров).

3.2. Блочные рекуррентно-итерационные процедуры в обучении

В работе [4] для решения НЗНК было предложено использовать блочные рекуррентно-итерационные процедуры (БРИП). Их суть состоит в том, что направление для минимизации ЗНК на некотором шаге итерационного процесса, определяемое с помощью метода Гаусса-Ньютона с псевдообращением, не следует вычислять непосредственно по формуле (2.17):

$$\Delta w_t = -\left[R'^T(w_t)R'(w_t)\right]^+ R'^T(w_t)R(w_t) = -\left[R'(w_t)\right]^+ R(w_t).$$

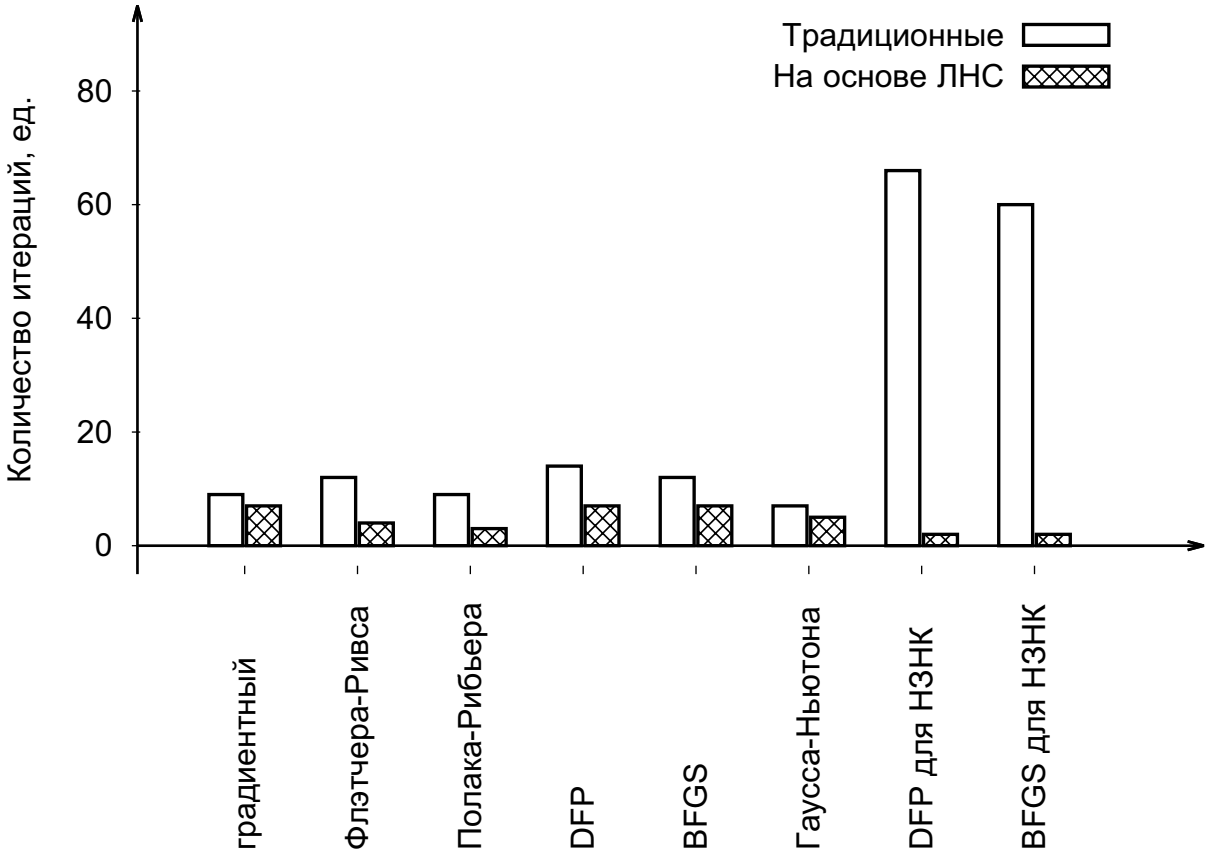


Рис. 3.4. Среднее количество итераций

Матрица A^+ может находиться с помощью алгоритмов Гревиля, Фаддеева, QR-разложения, SVD-разложения [2, 9, 19, 20]. Вместо это предлагается воспользоваться формулой Клайна для псевдообращения блочных матриц $\begin{bmatrix} A & B \end{bmatrix}$ [1]:

$$\begin{bmatrix} A & B \end{bmatrix}^+ = \begin{bmatrix} A^+(I - BL) \\ L \end{bmatrix}, \tag{3.21}$$

где

$$\begin{aligned} L = C^+ + (I - C^+C)KB^T(A^+)^TA^+(I - BC^+); \\ C = (I - AA^+)B; \\ K = (I + M^TM)^{-1}; \\ M = A^+B(I - C^+C). \end{aligned} \tag{3.22}$$

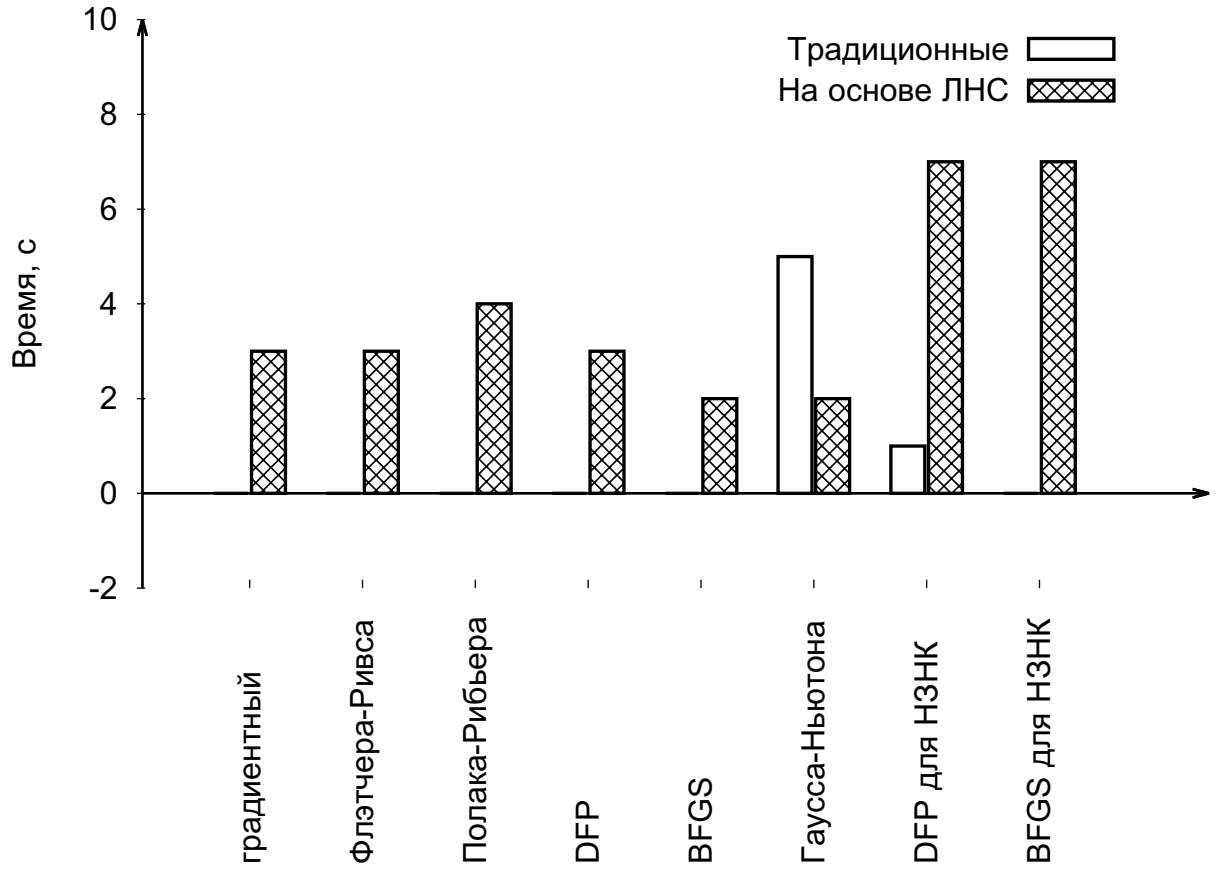


Рис. 3.5. Среднее время выполнения одной итерации

Для компьютерной реализации формулы (3.22) удобнее представить в виде

$$L = C^+ + NKB^T(A^+)^T A^+(I - BC^+);$$
$$C = (I - AA^+)B; \quad K = (I + M^T M)^{-1};$$
$$M = A^+ BN; \quad N = I - C^+ C. \quad (3.23)$$

Формула Клайна позволяет реализовать рекуррентную процедуру псевдообращения матрицы, состоящей из нескольких блоков. Данный способ может обосновываться преимуществами вычислительного характера, имеющими место при псевдообращении матриц небольших размеров.

БРИП основываются на разбиении исходного вектора параметров w на произвольное количество подвекторов $w^i, i = 1, \ldots, n$. На каждой итерации на основе рекуррентного алгоритма Клайна определяется направление минимизации вдоль каждой группы весов Δw^i.

Рассмотрим ситуацию, когда вектор w разбивается на две части: w^1 и w^2. Алгоритм Гаусса-Ньютона в соответствии с (3.21) для модели со скалярным выходом можно записать как

$$\begin{bmatrix}\Delta w^1\\ \Delta w^2\end{bmatrix} = \left[\nabla^T_{w^1} y \quad \nabla^T_{w^2} y\right]^+ (y-\widetilde{y}) = \begin{bmatrix}\left(\nabla^T_{w^1} y\right)^+ \left(I - \nabla^T_{w^2} y L\right)\\ L\end{bmatrix} (y-\widetilde{y}). \quad (3.24)$$

Это позволяет определить сначала один вектор

$$\Delta w^2 = L(y-\widetilde{y}),$$

затем на основе этого вычислить и другой:

$$\Delta w^1 = \left(\nabla^T_{w^1} y\right)^+ \left(I - \nabla^T_{w^2} y L\right)(y-\widetilde{y}) = \left(\nabla^T_{w^1} y\right)^+ \left(y - \widetilde{y} - \nabla^T_{w^2} y \Delta w^2\right).$$

БРИП могут быть применены для параметрической идентификации НСМ; при этом разбиение вектора весов становится естественным, вытекающим из самой структуры сети [17,18]. В отдельную часть могут быть выделены как всего слоя, так и отдельных нейронов; другие разбиения содержательного смысла не имеют. Рассмотрим НС ПР стандартной структуры. Пусть, как и в методе, основанном на декомпозиции задачи и псевдообращении, u – вектор линейно входящих, а v – нелинейно входящих в НСМ параметров. Так как порядок составления вектора w ($[\ u^T \quad v^T\ ]^T$ или $[\ v^T \quad u^T\ ]^T$) может быть выбран произвольным, алгоритм определения направления минимизации на основе (3.24) запишем в виде

$$\Delta v = L(y-\widetilde{y});$$

$$\Delta u = \left(\nabla^T_u y\right)^+ \left(y - \widetilde{y} - \nabla^T_v y \Delta v\right).$$

Вследствие того, что параметры u входят в НС линейно, справедливо, что

$$\nabla^T_u y = \Psi,$$

где $\Psi = \Psi(v) \in \mathbb{R}^{k\times q}$ – простым способом вычисляемая матрица, составленная из значений нелинейных функций $\psi_i(v,x), i = 1,\ldots,q$, на примерах

обучающего множества:

$$\Psi = \begin{bmatrix} \psi_1(v, x_1) & \psi_2(v, x_1) & \dots & \psi_q(v, x_1) \\ \psi_1(v, x_2) & \psi_2(v, x_2) & \dots & \psi_q(v, x_2) \\ \vdots & \vdots & \vdots & \vdots \\ \psi_1(v, x_k) & \psi_2(v, x_k) & \dots & \psi_q(v, x_k) \end{bmatrix}.$$

Итак, получили следующую формулу определения направления минимизации вдоль линейно входящих весов стандартной НС ПР:

$$\Delta u = \Psi^+ \left(y - \widetilde{y} - \nabla_v^T y \Delta v \right),$$

где матрица Якоби $\nabla_v^T y$ может быть вычислена аналогично алгоритму ОРО с учетом суперпозиционной структуры нейросетевой модели.

БРИП для параметрической идентификации НСМ могут быть распространены на многослойные сети. Пусть $A = \begin{bmatrix} A_1 & A_2 & \dots & A_n \end{bmatrix}$ и $A_{1..j} = \begin{bmatrix} A_1 & A_2 & \dots & A_j \end{bmatrix}$, тогда рекуррентный алгоритм блочного псевдообращения может быть представлен в следующем виде:

$$A^+ = \begin{bmatrix} A_{1..n-1}^+ (I - A_n L_{1..n-1}) \\ L_{1..n-1} \end{bmatrix};$$

$$A_{1..n-1}^+ = \begin{bmatrix} A_{1..n-2}^+ (I - A_{n-1} L_{1..n-2}) \\ L_{1..n-2} \end{bmatrix};$$

$$\vdots$$

$$A_{1..2}^+ = \begin{bmatrix} A_1 & A_2 \end{bmatrix}^+ = \begin{bmatrix} A_1^+ (I - A_2 L_1) \\ L_1 \end{bmatrix}.$$

Таким образом, определив матрицу A_1^+, можно рекуррентно посчитать другие части псевдообратной блочной матрицы. Выпишем формулу для нахождения псевдообратной матрицы $A_{1..i}^+$:

$$A_{1..i}^+ = \begin{bmatrix} A_{1..i-1}^+ (I - A_i L_{1..i-1}) \\ L_{1..i-1} \end{bmatrix}, \tag{3.25}$$

где матрица $L_{1..i}$ строится на основе $A_{1..i-1}^+$ и A_i.

Теперь применим данные формулы для вычисления направлений минимизации. Введя обозначение $\Delta w^{1..i} = \left[\Delta w^{1^T} \;\; \Delta w^{2^T} \;\; \dots \;\; \Delta w^{i^T}\right]^T$, получим

$$\Delta w^n = L_{1..n-1}(y - \widetilde{y});$$

$$\Delta w^{1..n-1} = A^+_{1..n-1}(I - A_n L_{1..n-1})(y - \widetilde{y}) = A^+_{1..n-1}(y - \widetilde{y} - A_n \Delta w^n);$$

$$\Delta w^{n-1} = L_{1..n-2}(y - \widetilde{y} - A_n \Delta w^n);$$

$$\Delta w^{1..n-2} = A^+_{1..n-2}(I - A_{n-1} L_{1..n-2})(y - \widetilde{y} - A_n \Delta w^n) =$$

$$= A^+_{1..n-2}(y - \widetilde{y} - A_n \Delta w^n - A_{n-1} \Delta w^{n-1});$$

$$\vdots$$

$$\Delta w^2 = L_1(y - \widetilde{y} - A_n \Delta w^n - A_{n-1} \Delta w^{n-1} - \ldots - A_3 \Delta w^3);$$

$$\Delta w^1 = A^+_1(I - A_2 L_1)(y - \widetilde{y} - A_n \Delta w^n - A_{n-1} \Delta w^{n-1} - \ldots - A_2 \Delta w^2) =$$

$$= A^+_1(y - \widetilde{y} - A_n \Delta w^n - A_{n-1} \Delta w^{n-1} - \ldots - A_2 \Delta w^2).$$

Видно, что рекуррентная формула для вычисления направления оптимизации i-го подвектора весов ($i = n, \ldots, 2$) записывается как

$$\Delta w^i = L_{1..i-1}\left(y - \widetilde{y} - \sum_{j=i+1}^{n} A_j \Delta w^j\right) \tag{3.26}$$

и для подвектора Δw^1:

$$\Delta w^1 = A^+_1\left(y - \widetilde{y} - \sum_{j=2}^{n} A_j \Delta w^j\right). \tag{3.27}$$

При $n = 2$ получается в точности (3.24). Для алгоритма Гаусса-Ньютона матрица $A_j = \nabla_{w^j} y$ находится с учетом суперпозиционной структуры на основе процедуры, аналогичной ОРО. Далее приведен алгоритм использования БРИП в параметрической идентификации НСМ.

Алгоритм 3.2. Применение БРИП в параметрической идентификации многослойных НСМ

1. Вычисление A^+_1, $D := A^+_1$.
2. Вычисление поблочно псевдообратной матрицы A^+.

(a) $i := 2$.

(b) Вычисление $L_{1..i-1}$ на основе D и A_i.

(c) Вычисление $A^+_{1..i}$, $D := A^+_{1..i}$.

(d) $i := i + 1$.

(e) Если $i < n$, возврат на шаг 2b.

3. $z := y - \widetilde{y}$.

4. Определение подвекторов Δw^i.

(a) $i := n$.

(b) $\Delta w^i := L_{1..i-1} z$.

(c) $i := i - 1$.

(d) $z := z - A_{i+1} \Delta w^{i+1}$.

(e) Если $i > 2$, возврат на шаг 4b.

(f) $\Delta w^1 := A^+_1 z$.

Заметим, что нет необходимости хранить все матрицы $A_{1..i}$ – достаточно знать эту матрицу на текущем шаге рекуррентного процесса. Кроме того, саму матрицу A^+ также нет надобности вычислять. При нахождении подвекторов для увеличения эффективности используется рекуррентно рассчитываемый вектор z.

Данный алгоритм, как и БРИП вообще, может основываться не только на базе метода Гаусса-Ньютона, но и на базе других методов ньютоновского типа с псевдообращением, определяющих направление минимизации по (2.16):

$$\Delta w_t = - \left[R'^T(w_t) R'(w_t) + Q(w_t) \right]^+ R'^T(w_t) R(w_t).$$

Это связано с тем, что общая схема формирования направления описывается формулой $\Delta w = \widehat{R}^+ \widehat{y}$, где $\widehat{R}$ – аппроксимация матрицы Гессе (своя для каждого метода); $\widehat{y}$ – вектор, содержащий информацию об указаниях учителя (в методе Гаусса-Ньютона $\widehat{y} = R(w) = y(w) - \widetilde{y}$).

Глава 4

Гарантированное обучение нейронных сетей на основе методов интервального анализа

4.1. Методы интервального анализа в глобальной оптимизации

Результат об аппроксимационных свойствах НС ПР, полученный Фунахаши, означает, что при достаточном количестве нейронов в скрытом слое и соответствующих весах этих нейронов может быть смоделирована любая непрерывная нелинейная функция.

Из этого утверждения следует, что при увеличении количества нейронов, а следовательно, и количества весов, НС ПР должна все более точно моделировать функцию, задаваемую обучающим множеством. На практике дело обстоит иным образом. В связи с тем, что нет гарантий того, что найденный оптимум является локальным, приходится либо прекращать обучение НС ПР имеющейся структуры, либо затрачивать большой объем времени на обучение. Даже если при возрастающих объемах сетей удается получать все меньшую ошибку, довольно высока вероятность того, что построенная в итоге нейросетевая структура будет обладать плохими обобщающими свойствами из-за наличия большого количества весов. Получается, что рассмотренные универсальные аппроксимационные способности НС ПР имеют в большей степени теоретическую значимость. В [13] представлен обзор существующих подходов к глобальной оптимизации.

Гарантировать нахождение глобального оптимума позволяет использование методов интервального анализа. В методах гарантированной глобальной интервальной оптимизации может быть эффективно учтен суперпози-

ционный характер нейросетевых моделей. В реальных задачах НС ПР содержат несколько сотен оптимизируемых параметров. Известно, однако, что интервальные методы оптимизации являются ресурсоемкими, что приводит, в частности, к большому времени работы даже при оптимизации целевой функции по небольшому количеству переменных. Задачи исследования возможностей применения интервальных методов параметрической идентификации в обучении НС и разработки алгоритмов повышения эффективности методов обучения являются важными.

Основоположником работ по интервальному анализу является Р.Е. Мур. За рубежом наиболее крупными учеными, занимающимися вопросами интервального анализа в применении к задачам оптимизации, являются В. Крейнович, Э. Хансен, Р.Б. Кирфот. В России с начала 80-х годов активную работу в данном направлении ведет школа академика Ю.И. Шокина в Новосибирске, в частности профессор С.П. Шарый.

4.1.1. Основы интервального анализа

Рассмотрим основы интервального анализа в соответствии с [11, 15, 23].

Интервал – односвязное подмножество множества вещественных чисел, задаваемое упорядоченной парой вещественных чисел:

$$[x] = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} : \underline{x} \leqslant x \leqslant \overline{x}\},$$

где $\underline{x}$, $\overline{x}$ – нижняя и верхняя границы интервала соответственно. Числовыми харатеристиками интервала $[x]$ являются его ширина

$$\mathrm{w}([x]) = \overline{x} - \underline{x}$$

и середина (центр)

$$\mathrm{m}([x]) = \frac{\underline{x} + \overline{x}}{2}.$$

Частным случаем интервала является вещественное число $x = [x, x]$.

Пусть X, Y, Z – множества, $* : X \times Y \to Z$ – некоторое бинарное отображение, тогда отображение $*$ расширяется для работы с множествами

следующим образом:

$$X_1 * Y_1 = \{x * y : x \in X_1 \subset X, y \in Y_1 \subset Y\}. \tag{4.1}$$

Если в (4.1) множества являются интервалами, то применение бинарного отображения определяет интервал:

$$[x] * [y] = [\min\{x * y : x \in [x], y \in [x]\}, \max\{x * y : x \in [x], y \in [x]\}]. \tag{4.2}$$

Если интервалы являются замкнутыми непустыми множествами, то арифметические операции в соответствии с (4.2) сводятся к вычислениям над границами:

— Умножение на число $a \in \mathbb{R}$:

$$a[x] = \begin{cases} [a\underline{x}, a\overline{x}], & a \geqslant 0, \\ [a\overline{x}, a\underline{x}], & a < 0. \end{cases}$$

— Сложение интервалов:

$$[x] + [y] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}].$$

— Разность интервалов:

$$[x] - [y] = [\underline{x} - \overline{y}, \overline{x} - \underline{y}].$$

— Умножение интервалов:

$$[x][y] = [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}, \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}].$$

— Деление интервалов:

$$\frac{[x]}{[y]} = [x]\left[\frac{1}{\overline{y}}, \frac{1}{\underline{y}}\right], 0 \notin [\underline{y}, \overline{y}].$$

Приведенная выше интервальная арифметика называется классической и обозначается $\mathbb{IR}$. Определенные таким образом арифметические операции являются обобщением операций для обычной арифметики вещественных чисел.

Важной положительной особенностью интервального расширения арифметических операций является их монотонность по включению:

$$[x] \subset [x'], [y] \subset [y'] \Rightarrow [x] * [y] \subset [x'] * [y'].$$

Это означает, что при переходе к более узкому диапазону изменения аргумента результат вычилений не может увеличиться.

Естественным образом арифметические операции распространяются на интервальные векторы (брусы, параллелотопы) $[x] = [x_1 \quad x_2 \quad \dots \quad x_n]^T$ и матрицы. Соответствующие пространства обозначаются $\mathbb{IR}^n$ и $\mathbb{IR}^{m \times n}$. Середина бруса – обычный вектор $\mathrm{m}([x])$ с координатами

$$\mathrm{m}([x]) = \left[\frac{\underline{x}_1 + \overline{x}_1}{2} \quad \frac{\underline{x}_2 + \overline{x}_2}{2} \quad \dots \quad \frac{\underline{x}_n + \overline{x}_n}{2}\right]^T, \tag{4.3}$$

ширина бруса

$$\mathrm{w}([x]) = \max_{i=1}^{n} \mathrm{w}\left([x_i]\right). \tag{4.4}$$

Формула (4.1) расширяется на функции, аргументы которых являются интервалами:

$$f([x]) = \{f(x) : x \in [x] \in \mathbb{IR}^n\}. \tag{4.5}$$

По сути, это задача определения образа функции f на множестве $[x]$. Интервальные функции также являются монотонными по включению:

$$[x] \subset [y] \Rightarrow f([x]) \subset f([y]).$$

Интервальная функция (4.5) определяется просто только для монотонных функций. Если $f(x)$ – возрастающая (неубывающая) функция на $[x]$, то

$$f([x]) = [f(\underline{x}), f(\overline{x})].$$

Например, для экспоненты, часто вычисляемой в НС, интервальная функция вычисляется так:

$$e^{([x])} = [e^{\underline{x}}, e^{\overline{x}}].$$

Если $f(x)$ – убывающая (невозрастающая) функция на $[x]$, то

$$f([x]) = [f(\overline{x}), f(\underline{x})].$$

Для некоторых функций могут быть получены формулы для вычисления интервальной функции с учетом областей их монотонности. Для функции возведения в квадрат расчет будет таким:

$$[x]^2 = \begin{cases} [0, \max\{\underline{x}^2, \overline{x}^2\}], & 0 \in [\underline{x}, \overline{x}], \\ [\underline{x}^2, \overline{x}^2], & \underline{x} > 0, \\ [\overline{x}^2, \underline{x}^2], & \overline{x} < 0. \end{cases}$$

Следует заметить, что расчет значения интервальной функции зависит от ее формального представления. Например, $[x] \cdot [x]$ в общем случае даст ответ, отличный от $[x]^2$. Общее правило таково: чем меньше переменная встречается в формальном выражении функции, тем более узким будет вычисленный интервал, в который попадут значения функции.

В общем случае точное вычисление интервальной функции невозможно. Вместо этого используют функции включения (внешняя оценивающая функция или интервальное расширение в терминах [23]), т.е. достаточно легко вычисляемые функции $[f]([x])$, удовлетворяющие условию

$$f([x]) \subset [f]([x]) \, \forall [x] \subset \mathbb{R}^n . \tag{4.6}$$

Существует несколько вариантов построения функций включения. Самой простой и универсальной, хотя и бесполезной, функцией включения является функция

$$[f]([x]) = \mathbb{R} \, \forall [x] \in \mathbb{IR}^n .$$

Наиболее распространенными являются естественная $[f]_n$, центрированная $[f]$ и тейлоровская функция включения второго порядка $[f]_T$. Естественная функция включения $[f]_n$ получается заменой всех переменных и операций их интервальными аналогами. Центрированная функция включения $[f]_c$ получается путем применения теоремы о среднем:

$$[f]_c([x]) = f(m) + [g^T]([x])([x] - m),$$

где $m = \mathrm{m}([x])$; g – градиент функции f. Тейлоровская функция включения

имеет вид

$$[f]_T([x]) = f(m) + [g^T](m)([x] - m) + \frac{1}{2}([x] - m)^T [H]([x])([x] - m),$$

где H – матрица Гессе.

Как правило, на небольших интервалах центрированная функция дает более точные результаты, чем естественная [15]. На больших интервалах естественная функция включения предпочтительней. Тейлоровская функция второго порядка, как правило, показывает лучшие результаты по сравнению с естественными и центрированными функциями и для малых, и для больших интервалов.

4.1.2. Интервальные алгоритмы глобальной оптимизации

На основе использования арифметических операций над интервалами и функций включения могут быть разработаны алгоритмы гарантированной глобальной идентификации. Следует помнить, что задача глобальной оптимизации является условной в том смысле, что ее рассмотрение ведется на некотором, пусть даже очень широком, брусе $[x] \in \mathbb{IR}^n$. Основная идея, имеющая больше теоретическое значение вследствие трудоемкости, предложенная Р.Е. Муром, состоит в последовательном разбиении исходного бруса, содержащего точку глобального минимума, на подбрусы и оценке на них образа целевой функции. Сходимость обеспечивается свойством монотонности функций. Вычисления прекращаются, когда ширина брусов окажется меньше заданной точности. Методы глобальной оптимизации на основе интервального анализа относятся к категории методов ветвей и границ. Рассматриваемые в данном подразделе алгоритмы приведены с точки зрения их возможного применения для задачи параметрической идентификации НСМ.

Бисекцией бруса $[x] = [\underline{x}, \overline{x}]$ по i-й координате называется его разделение на два новых бруса: $[x^1]$ и $[x^2]$, где $[x_i^1] = [\underline{x_i}, \mathrm{m}([x_i])]$, $[x_i^2] = [\mathrm{m}([x_i]), \overline{x_i}]$, остальные границы брусов остаются неизменными. Приведем алгоритм простейшего интервального метода глобальной оптимизации GlobOpt для нахождения одного глобального оптимума 4.1 [23].

Алгоритм 4.1. Простейший алгоритм глобальной интервальной оптимизации GlobOpt

Вход:

— $f(w), w \in \mathbb{R}^n$ – целевая (минимизируемая) функция.

— $[w] \in \mathbb{IR}^n$ – начальный брус.

— $\varepsilon > 0$ – минимально допустимая ширина бруса.

Выход:

— $[p]$ – брус, содержащий точку глобального минимума.

— f^* – оценка снизу глобального минимума.

Алгоритм:

1. Инициализация: $[p] := [w]$.
2. Оценка минимума: $f^* := \underline{[f]}\,([p])$.
3. Инициализация списка: $L := \{([p], f^*)\}$.
4. Пока $\mathrm{w}\,([f]\,([p])) \geqslant \varepsilon$ повторять:
   (a) Выбор компоненты l, по которой брус $[p]$ имеет наибольшую ширину:
   $$l := \arg\max_{i=1}^{n} \mathrm{w}\,([p_i])\,.$$
   (b) Бисекция $[p]$ по l-й координате на $[p']$ и $[p'']$.
   (c) Вычисление $f' := \underline{[f]}\,([p'])$, $f'' := \underline{[f]}\,([p''])$.
   (d) Удаление из списка L пары $([p], f^*)$.
   (e) Помещение пар $([p'], f')$ и $([p''], f'')$ в список L в порядке возрастания второго поля.
   (f) Обозначение первой (ведущей) записи списка L через $([p], f^*)$.

Алгоритм 4.1 GlobOpt не может быть с успехом применен к решению практических задач, так как эффект от бисекции при увеличении размерности становится все менее и менее ощутим. Основное внимание в алгоритмах уделяется разработке процедур удаления брусов из списка, без чего создание эффективных алгоритмов невозможно.

Модификации приведенного выше алгоритма заключаются в удалении из списка L брусов, точно не содержащих оптимальные значения, предложенные Хансеном в 1980 г. [30]:

— Не выполняется тест на значение в средней точке, т.е. $\underline{[f]}([w]) > f(\mathrm{m}([w]))$ для бруса $[w]$ из списка L.

— Не выполняется тест на необходимое условие оптимума (тест на монотонность), т.е. $0 \notin g([w])$ ($\underline{[g_i]} > 0$ или $\overline{[g_i]} < 0$) для некоторого $i = 1, \ldots, n$, где g – градиент функции f по вектору w.

— Не выполняется тест на выпуклость вниз, например если не для всех i $h_{ii}([w]) \geqslant 0$, $i = 1, \ldots, n$, где $h_{ii}(\cdot)$ – диагональные элементы матрицы Гессе функции f.

Вместо теста на значения в средней точке может быть тест на значение в любой другой точке, найденной с помощью локальных методов оптимизации внутри бруса. Также внимание для повышения эффективности работы методов может быть уделено:

— Построению более качественной функци включения.

— Использования алгоритмов локальной оптимизации дифференцируемых функций.

В [23, 41] предлагаются методы оптимизации, основанные на дроблении графика функции, что включает в себя задача проверки совместности уравнений. В связи с тем, что такая задача не менее легка, чем задача оптимизации, в данной работе такой подход н рассматривается.

Алгоритм интервального метода глобальной оптимизации для нахождения всех глобальных оптимумов GlobalOptimize приведен ниже, с тем отличием от [11], что не используется шаг интервального метода Ньютона-Гаусса-Зейделя.

Алгоритм 4.2. Алгоритм глобальной интервальной оптимизации для нахождения всех оптимумов GlobalOptimize

Вход:

— $f(w), w \in \mathbb{R}^n$ – целевая (минимизируемая) функция.

— $[w] \in \mathbb{IR}^n$ – начальный брус.

— $\varepsilon > 0$ – минимально допустимая ширина бруса.

Выход:

— L_{res} – список брусов, содержащих точки глобального минимума.

— $[f^*]$ – оценка глобального минимума.

Алгоритм:

1. Инициализация: $[p] := [x]$, $c := \mathrm{m}\left([p]\right)$.
2. Оценка минимума сверху: $\widetilde{f} := \overline{[f]}\left([p]\right)$.
3. Инициализация списков: $L := \{\}$, $L_{res} := \{\}$.
4. Основной цикл:
   (a) Выбор компоненты l, по которой брус $[p]$ имеет наибольшую ширину:
   $$l := \arg\max_{i=1}^{n} \mathrm{w}\left([p_i]\right).$$
   (b) Бисекция $[p]$ по l-й координате на $[p_1]$ и $[p_2]$.
   (c) Цикл по $i := 1..2$:
      i. $[g] := [f']\left([p_i]\right)$ – функция включения для градиента.
      ii. Если не выполнен тест на монотонность (функция монотонна), то переход на следующий виток цикла i.
      iii. $[f]_c := \left(f(c) + [g]\left([p_i] - c\right)\right.$ – центрированная форма функции включения.
      iv. Если $\widetilde{f} < \underline{[f]}_c$, то переход на следующий виток цикла i.
      v. $[H] := [f'']\left([p_i]\right)$ – функция включения для матрицы Гессе.
      vi. Если не выполнен тест на выпуклость (функция не является выпуклой вниз), то переход на следующий виток цикла i.
      vii. $L := L + \left([p_i], \underline{[f]}\left([p_i]\right)\right)$ – добавление в список.
   (d) ВыполнятьБисекцию := Ложь.
   (e) Цикл: Пока $(L \neq \{\})$ и (не ВыполнятьБисекцию)
      i. $\left([p], \underline{f}\right)$:= первый элемент списка L;
      $L := L - \left([p], \underline{f}\right)$ – удаление из списка;
      $c := \mathrm{m}([p])$.
      ii. $\widetilde{f} := \min\{\widetilde{f}, f(c)\}$. Удаление всех брусов из списка L, не проходящих тест на среднюю точку по сравнению с $\widetilde{f}$.
      iii. $[f^*] := [\underline{f}, \widetilde{f}]$.
      iv. Если $\left(\mathrm{w}\left([f^*]\right) \leqslant \varepsilon\right.$ и $\left(\mathrm{w}\left([p]\right) \leqslant \varepsilon\right.$, то $L_{res} := L_{res} + \left([p], \underline{f}\right)$ иначе ВыполнятьБисекцию := Истина.

ПОКА (ВыполнятьБисекцию).

5. $\left([p], \underline{f}\right) :=$ первый элемент списка L; $[f^*] := [\underline{f}, \widetilde{f}]$.

В данном алгоритме при достижении желаемой точности брус сохраняется в списке результатов L_{res}, иначе для него осуществляется возврат к шагу бисекции.

В алгоритме Мура-Скелбо указано также на модификацию теста на среднюю точку, когда предлагается вместо середины бруса c использовать точку, полученную на основе процедуры локальной оптимизации, начиная с точки c. Ратчек и Рокне показали также, что бисекцию лучше производить по координате l, выбираемой из условия

$$l := \arg \max_{i=1}^{n} \mathrm{w}\left([f']\left([p_i]\right)\right) \cdot \mathrm{w}\left([p_i]\right).$$

Хансен [30–32] предлагает также в качестве условия останова использовать комбинацию условий

$$\mathrm{w}([w]) < \varepsilon \text{ и } \mathrm{w}([f][w]) < \delta.$$

Алгоритм Хансена использует понятие сжимающего оператора.

Сжимающие операторы C – процедуры, переводящие исходную область $[w]$ в область $[w'] \subset [w]$, и множество решений исходной задачи не изменяется за счет применения операции пересечения

$$[w'] = [w] \cap C_{[w]}.$$

Не существует оптимальных операторов, их эффективность зависит от решаемой задачи. Сжимающий оператор C может вообще не привести к уменьшению исходной области. Заметим, что эффективные сжимающие операторы для решения задач глобальной оптимизации строятся в том случае, когда известна информация о вторых производных минимизируемой функции.

4.2. Применение методов интервального анализа в обучении

Универсальные аппроксимационные свойства НС ПР означают, что при достаточном количестве нейронов в скрытом слое и соответствующих весах

этих нейронов любая непрерывная нелинейная функция может быть приближена с произвольной точностью. По сути, при увеличении количества нейронов, а следовательно, и количества весов НС ПР должна все более точно моделировать функцию, задаваемую обучающим множеством. На практике дело обстоит иначе. В связи с тем, что нет гарантий того, что найденный оптимум является глобальным, приходится либо прекращать обучение НС ПР имеющейся структуры, либо затрачивать большой объем времени на обучение, используя стохастические подходы к оптимизации. Даже если при возрастающих объемах сетей удается получать все меньшую ошибку, довольно высока вероятность того, что построенная в итоге НСМ будет обладать плохими обобщающими свойствами из-за наличия большого количества весов. Получается, что универсальные аппроксимационные способности НС ПР имеют в большей степени теоретическую значимость, чем практическую.

Гарантировать нахождение глобального оптимума позволяет использование методов интервального анализа. В методах гарантированной глобальной интервальной оптимизации может быть эффективно учтен суперпозиционный характер нейросетевых моделей. Во многих реальных задачах НС ПР содержат сотни оптимизируемых весов. Известно, однако, что интервальные методы оптимизации являются ресурсоемкими, что приводит, в частности, к огромному времени работы даже при оптимизации целевой функции по небольшому количеству переменных. В [15] указано, что для применения интервальных методов задачами высокой размерности являются задачи с количеством переменных больше 10 и задачи с функциями, в которые переменные входят многократно. Задача исследования возможностей применения численного метода гарантированного обучения НСМ на основе методов интервальной глобальной оптимизации и совершенствования метода обучения за счет учета специфики решаемой задачи оптимизации является актуальной. В данном разделе анализируется применимость различных техник интервального анализа и их комбинирование с алгоритмами оптимизации к обучению НСМ.

Работы по синтезу НС и методов интервального анализа появились в начале 1990-х годов и привели к понятию интервальных НС (ИНС, Interval

Neural Networks), предназначенных для обработки неопределенных входных сигналов, представляемых в виде интервалов. НС называется интервальной, если хотя бы один из ее параметров – вход, выход или вес – является интервальным [27]. Сфера их применения в первую очередь – робастное управление [39, 40]. Применение интервальных методов для гарантированной глобальной оптимизации в обучении классических НСМ фактически не исследовалось. В работе 1998 года декларирована возможность применения интервальных методов для обучения НС ПР, а также отмечено, что эти алгоритмы до той поры не применялись [26]. В [24] отмечена сложность использования производной по вектору весов вследствие появления больших неточностей в оценке образа из-за эффекта обертывания – вхождения веса кратное количество раз в формальное выражение функции, реализуемой НС ПР. Там же предлагается для ускорения обучения НСМ на основе методов интервального анализа использовать сплайны в функциях активации. В работе [25] исследуется применимость методов интервального анализа в максимизации выходных сигналов НСМ, что связано со значительно меньшей размерностью пространства входов по сравнению с размерностью пространства весов НСМ. Применение программного обеспечения для глобальной интервальной оптимизации – пакета GlobSol и его расширение ParaGlobSol для работы в параллельных средах [33, 34] – в обучении НСМ затруднительно, так как пакет рассчитан на оптимизацию произвольных функций, учесть специфику задачи обучения НСМ невозможно.

В случае использования функции активации (1.6) все операции будут корректными, так как не будут возникать ситуации деления на интервалы, содержащие нуль. Использовать другую функцию активации, например биполярную сигмоидную логистическую – гиперболический тангенс:

$$\sigma(net) = \mathrm{th}(\mathrm{net}) = \frac{\mathrm{e}^{\mathrm{net}} - \mathrm{e}^{-\mathrm{net}}}{\mathrm{e}^{\mathrm{net}} + \mathrm{e}^{-\mathrm{net}}},$$

область значений которой представляет собой интервал $(-1; 1)$, сложнее, так как необходимо применять расширенную интервальную арифметику, при которой результатом деления на интервал, содержащий нуль, будет получаться

объединение двух интервалов. Но более важно то, что при этом будут получаться очень широкие интервалы, так как результаты будут содержать бесконечные границы. Широкие интервалы будут получаться и для функции (1.6), если нижняя граница будет близка к нулю. Учитывая свойство насыщения этой функции, подобная ситуация будет встречаться достаточно часто, так как уже при $net = -7$

$$\frac{1}{1+e^{-7}} \approx 0,0009,$$

что близко к нулю. Улучшить поведение функции, реализуемой нейронной сетью, легко – достаточно использовать модифицированные функции активации, которые отличаются от классических на некоторую константу, например вместо (1.6) применять функцию

$$\sigma_m(net) = \frac{1}{1+e^{-net}} + 1. \tag{4.7}$$

Это не приведет к необходимости изменения методов обучения, так как производная функции не изменится. При этом нужно иметь в виду, что производная (1.6) в терминах функции выглядит как

$$\sigma'(net) = \frac{e^{-net}}{(1+e^{-net})^2} = \sigma(net)\left(1-\sigma(net)\right),$$

для функции (4.7) производная

$$\sigma'_m(net) = \frac{e^{-net}}{(1+e^{-net})^2} = \left(\sigma_m(net)-1\right)\left(2-\sigma_m(net)\right).$$

Использовать в (4.7) большие значения констант нецелесообразно, так как это приведет к неотличимости входных сигналов для нейрона выходного слоя.

Работоспособность алгоритмов обучения НС может быть иллюстрировано на следующем модельном примере. На основе функции 2-х переменных [14]:

$$z = 0,5\sin(\pi x^2)\sin(2\pi y)$$

было сформировано случайным образом обучающее множество, состоящее из 500 примеров, при этом $-1 \leqslant x \leqslant 1$, $-1 \leqslant y \leqslant 1$. Результаты экспериментов приведены в табл. 4.1.

Сводные результаты вычислительных экспериментов с нейронными сетями с двумя входными и одним выходным нейронами, сравнение метода обучения на основе интервального анализа и метода обратного распространения ошибки приведены в табл. 4.1.

Таблица 4.1

Результаты применения интервальных методов в обучении

Показатель	Сеть без скрытых нейронов (3 веса)	Сеть с 1 скрытым нейроном (5 весов)	Сеть с 2 скрытыми нейронами (9 весов)
Численный метод на основе обратного распространения ошибки			
Средняя квадратическая ошибка	0,2312	1,1904	1,1704
Время обучения, мин:с	00:01	01:23	02:12
Численный метод обучения на основе интервального анализа			
Средняя квадратическая ошибка (интервал)	$[0,1602; 0,2595]$	$[0,1404; 0,3409]$	$[0,0351; 0,8796]$
Значение в средней точке	0,2063	0,2046	0,1994
Время обучения, мин:с	00:07	25:20	33:06
Ширина бруса	0,0625	0,0850	0,1000
Брусов в списке	78	23309	27028
Удалено по тесту в средней точке	46	1128	0
Удалено по тесту на необходимое условие оптимума	99	1784	48

В табл. 4.2 приведены результаты исследования эффективности интервального метода обучения при использовании различных функций включения. Из таблицы видно, что наибольшую эффективность дает применение комбинации естественной и центрированной функций включения. Эта комбинация использовалась в дальнейших экспериментах для оценки образа функционала качества.

Таблица 4.2

Время обучения при разных функциях включения, мин:с

Показатель	Сеть без скрытых нейронов (3 веса)	Сеть с 1 скрытым нейроном (5 весов)	Сеть с 2 скрытыми нейронами (9 весов)
$[f]_n$	00:09	31:09	39:13
$[f]_c$	00:12	46:35	58:14
$[f]_n \cap [f]_c$	00:07	25:20	33:06

В табл. 4.3 приведены данные исследования эффективности применения локального поиска для модификации теста в средней точке для интервальных методов обучения. В качестве метода локального поиска исследовался алгоритм обратного распространения ошибки в двух вариациях: с фиксированным шагом и с переменным шагом, основанном на одномерном методе оптимизации – методе первого приемлемого значения. Применение локальных методов поиска не показало свою эффективность, что можно объяснить неэффективностью их применения для брусов малых размеров. В дальнейших исследованиях локальный поиск не применялся.

Из вычислительных экспериментов можно сделать следующие выводы:

— Обучение простейших нейронных сетей без скрытых слоев (фактически линейных по параметрам моделей) производится очень эффективно даже при широких исходных брусах и больших объемах обучающих множеств.

— Обучение нелинейных нейронных сетей проводится медленно: несколько минут или часов в зависимости от параметров работы алгоритма.

Таблица 4.3

Время обучения при использовании локального поиска, мин:с

Показатель	Сеть без скрытых нейронов (3 веса)	Сеть с 1 скрытым нейроном (5 весов)	Сеть с 2 скрытыми нейронами (9 весов)
Без локального поиска	00:07	25:20	33:06
С локальным поиском (фиксированный шаг)	00:11	45:30	54:19
С локальным поиском (переменный шаг)	00:08	30:42	37:21

— Для оценки образа функционала качества целесообразно применение пересечение естественной и центрированной функций включения.

— Применение методов локального поиска для уточнения значения функционала в средней точке неэффективно.

— Использование информации о градиенте повышает эффективность отсеивания неперспективных брусов, а также определить отсутствие решения.

— Качество и надежность работы интервальных методов обучения нейронных сетей существенно выше, чем для метода обратного распространения ошибки, хотя и значительно затратнее по временному ресурсу.

Глава 5

Построение нейронных сетей оптимальной структуры

5.1. Анализ подходов к построению оптимальных моделей

Задача построения НСМ оптимальной структуры относится к классу задач структурной идентификации. В отношении НСМ это означает необходимость определения количества скрытых слов, количества нейронов в скрытом слое, а также функций активации нейронов. Заметим, что под НСМ оптимальной структуры понимается та НСМ, которая обеспечивает не минимум функционала качества обучения, а та, которая обладает высокими обобщающими способностями, т.е. способностями хорошо описывать данные, не входящие в обучающее множество, а составляющие отдельное тестовое множество. Теоретически увеличение количества слоев и нейронов в сети приводит к уменьшению ошибки обучения. Однако это с определенного момента начинает отрицательно сказываться на способности НСМ описывать зависимость на данных, не включенных в обучающее множество.

Задача структурной идентификации моделей является трудноформализуемой. Задача построения оптимальной структуры для НСМ является частично решенной, так как задана архитектура НС, т.е. общий алгоритм построения и функционирования НС. Это определено способом преобразования информации нейронами, а также структурой допустимых связей. Априорно определить оптимальную структуру НСМ не представляется возможным, можно дать лишь некоторые грубые оценки.

Известно, что для k точек можно построить полиномиальную модель порядка $k-1$, проходящую через каждую из k точек. Это связано с необ-

ходимостью решения системы линейных алгебраических уравнений, так как с помощью k уравнение можно определить нахождения k линейно входящих параметров, что соответствует полиному $(k-1)$-й степени. При этом построенная модель, скорее всего, будет обладать плохими обобщающими свойствами [12]. Тем самым, можно дать оценку верхней границы количество параметров НСМ:

$$N_w \leqslant k,$$

где N_w – количество параметров НСМ; k – объем обучающего множества. Количество параметров wc с учетом фиктивных единичных сигналов для каждого нейрона для полносвязных НС ПР можно рассчитать по формуле

$$N_w = \sum_{l=1}^{m} (N_{l-1} + 1) N_l,$$

где $N_0 = n$, $N_m = r$ – количество входов и выходов НСМ соответственно.

Вариант решения данной задачи могут дать результаты, полученные А.Н. Колмогоровым и В.И. Арнольдом, об аппроксимации функций с помощью функций одной переменной, линейной комбинации и суперпозиции функций. Ими показано, что для аппроксимации функции от n переменных достаточно $2n+1$ нелинейных функций одной переменной, соответствующих нейронам скрытого слоя. Тем не менее, алгоритм построения этих функций настолько сложен, что описание их аналитическим образом не представляется возможным. В связи с указанными результатами некоторые исследователи НСМ предлагают ориентироваться на такую оценку числа нейронов.

Возможны три стратегии выбора структуры НС ПР:

— Генерация набора НС, не зависимых друг от друга, и дальнейший выбор лучшей из них.

— Контрастирование НС (network pruning, редукция НС), заключающееся в оценке значимости нейронов в текущей НС и удалении наименее значимых из них.

— Конструктивный подход (constructive, network growing), заключающийся в последовательном наращивании количества слоев и/или нейронов в скрытых слоях, начиная с сети минимальной структуры.

Каждый из подходов имеет свои преимущества и недостатки. В частности, процедура генерации независимых структур НС проста в реализации, однако может привести к пропуску хорошей структуры. Рассмотрим подробнее второй и третий подходы к построению оптимальных НСМ.

5.1.1. Контрастивный подход

В соответствии с [22] контрастивный подход опирается на одну из двух идей:

1. Применение техники регуляризации.
2. Удаление незначащих связей.

При применении техники регуляризации в процессе обучения минимизируется модифицированный функционал качества обучения, включающий слагаемое, отражающее близость выходов НС указаниями учителя (2.2), и слагаемое, отражающее сложность выбранной структуры НС:

$$J(w) + \lambda E_c(w), \tag{5.1}$$

где $E_c(w)$ – штраф за сложность структуры НС ПР; λ – параметр регуляризации, определяющий относительную значимость слагаемых. Выражение (5.1) представляет собой общий вид функции из теории регуляризации Тихонова. Если $\lambda = 0$, то обучение представляет собой обычный процесс параметрической идентификации.

В качестве штрафа за сложность структуры НС ПР могут использоваться:

1. Процедура снижения значений весов:

$$E_c(w) = \|w\|^2.$$

Данный критерий не совсем описывает логику (5.1), но является очень простым в реализации и позволяет снизить количество избыточных весов, не влияющих на обобщающие способности НС.

2. Процедура исключения весов:

$$E_c(w) = \sum_{i=1}^{k} \frac{(w_i/w_0)^2}{1 + (w_i/w_0)^2},$$

где w_0 – некоторый предопределенный масштабирующий параметр. Данный способ контрастирования обобщает описанную выше процедуру снижения значений весов.

3. Процедура сглаживающей аппроксимации для одновыходной НС с одним скрытым слоем:

$$E_c(w) = \sum_{i=1}^{q} w_i^2 \|w^{(i)}\|^p,$$

где w_i – вес выходного нейрона, идущий от i-го нейрона скрытого слоя; $w^{(i)}$ – вектор весов i-го нейрона скрытого слоя; p – натуральное число, определяющее порядок сглаживания. Данная формула разделяет роли весовых коэффициентов выходного и скрытого слоев, а также отражает их взаимодействие.

4. Процедура оценки сложности НСМ на основе меры Вапника-Червоненкиса [14]:

$$\varepsilon(VCdim),$$

где $\varepsilon(\cdot)$ – убывающая функция, можно принять $\varepsilon(VCdim) = \dfrac{1}{VCdim}$. $VCdim$ – мера Вапника-Червоненкиса, которая отражает уровень сложности НС, метод ее точного определения отсутствует. Рекомендовано в качестве границ этой меры принимать для стандартной НСМ

$$2\left[\frac{q}{2}\right] n \leqslant VCdim \leqslant 2N_w \left(1 + \lg N_n\right),$$

где q – количество нейронов скрытого слоя; $[\cdot]$ – целая часть числа; N_n – количество нейронов в НСМ. Можно принять [14]:

$$VCdim \approx N_w.$$

Недостатками данного способа упрощения сети являются необходимость первоначального задания структуры НСМ, априорное задание параметра регуляризации и высокие вычислительные сложности.

Второй способ контрастирования заключается в первоначальном обучении НС избыточно большой структуры, дальнейшим оцениванием значимости весов НС и удалением незначащих весов. В качестве теоретической основы оценки значимости весов выступает анализ локального поведения функции в окрестности текущей точки на основе формулы Тейлора. Пусть w^0, w^* – текущий и возмущенный векторы весов, $\Delta w = w^* - w^0$ – возмущение. Тогда приближение функционала качества (2.10) представится в виде

$$J(w^*) \approx J(w^0) + \nabla_w^T J(w^0)(w^* - w^0) + \frac{1}{2}(w^* - w^0)^T \nabla_w^2 J(w^0)(w^* - w^0) = \\ = J(w^0) + \nabla_w^T J(w^0)\Delta w + \frac{1}{2}\Delta w^T \nabla_w^2 J(w^0)\Delta w. \quad (5.2)$$

Применение процедуры контрастирования рассмотрено в работах [10, 12]. В них рассматривается линейное приближение мгновенного функционала качества для примера j вместо (5.3):

$$Q(w^*) \approx Q(w^0) + \nabla_w^T Q(w^0)\Delta w = Q^0 + \sum_{i=1}^{N_w} \frac{\partial Q}{\partial w_i}\left(w_i^* - w_i^0\right).$$

При замене w^0 на w^* оценка изменения веса i на изменение значения мгновенного функционала качества $\Delta Q = Q(w^*) - Q(w^0)$ приближенно вычисляется как

$$\chi(i,j) = \left|\frac{\partial Q}{\partial w_i}\right| \cdot \left|w_i^* - w_i^0\right|.$$

Величина $\chi(i,j)$ – показатель чувствительности к замене w_i^0 на w_i^* для примера j. Для оценки чувствительности веса на всех примерах можно использовать формулу

$$\chi(i) = \max_{j=1}^{k} \chi(i,j).$$

Далее на основе оценок чувствительности $\chi(i)$ вычисляется номер веса

$$i^* = \arg\min_{i=1}^{N_w} \chi(i),$$

после чего производится обучение НС без веса i^*. Если при этом качество работы НС не сильно ухудшилось, вес i^* удаляется (контрастируется). Подобная процедура выполняется итерационно.

В [14] не рекомендуется применять процедуру контрастирования в процессе обучения, так как веса могут временно характеризоваться низкой оценкой чувствительности, например в связи с неудачным выбором начальной точки. В случае, когда обучение НС произведено, градиент функционала $\nabla_w J(w)$ оказывается равным нулю, и для оценки значимости весов необходимо пользоваться информацией о вторых производных. Тогда в (5.3) слагаемое $\nabla_w^T J(w^0)\Delta w = 0$ и поэтому

$$\Delta J = J(w^*) - J(w^0) \approx \frac{1}{2}\Delta w^T \nabla_w^2 J(w^0)\Delta w. \tag{5.3}$$

Наиболее эффективными методами, основанными на анализе (5.3), являются метод оптимальной степени повреждения ЛеКуна OBD (Optimal Brain Damage) и метод оптимальной хирургии мозга Б. Хассиби и Д. Шторка OBS (Optimal Brain Surgeon) [14]. Метод OBD основан на предположении, что матрица Гессе является положительно определенной с диагонально доминирующими элементами, метод OBS такие допущения не использует. В связи с этим OBS включает в себя метод OBD как частный случай.

Цель OBS – обнуление одного из весов для минимизации (5.3) [22]. Для удаления веса i необходима такая корректировка, чтобы

$$\Delta w_i = -w_i$$

или

$$e_i^T \Delta w + w_i = 0, \tag{5.4}$$

где $e_i \in \mathbb{R}^n$, составленный из нулей, кроме 1 на месте i. Получается задача условной оптимизации (5.3) при условии (5.4). Решая данную задачу с помощью метода множителей Лагранжа, получается

$$\Delta w = -\frac{w_i}{[(\nabla^2 J)^{-1})]_{ii}}(\nabla^2 J)^{-1} e_i, \tag{5.5}$$

где $\left[(\nabla^2 J)^{-1})\right]_{ii}$ – диагональный элемент обратной матрицы Гессе. Вес i^* с наименьшим значением (5.5) удаляется, если изменение ΔJ при этом намного меньше значения функционала $J(w)$. Данный процесс является также итерационным и продолжается до отсечения всех малозначащих весов.

Недостатком такого подхода является необходимость определения первоначальной структуры НСМ достаточно больших размеров.

5.1.2. Конструктивный подход

Суть конструктивного подхода состоит в начальном выборе НС минимальной структуры и дальнейшем последовательном наращивании структуры путем добавления нейронов в скрытые слои или созданием новых слоев. Теоретически это позволяет построить сеть с достаточно небольшим количеством слоев. На практике это вызывает сложности, связанные с необходимостью такого обучения новой НСМ, чтобы ошибка обучения уменьшалась. При этом возникает необходимость решения многоэкстремальной задачи оптимизации.

Одним специальным методом конструктивного обучения является подход на применении сетей каскадной корреляции (СКК), предложенный Фальманом [14,28] (рис. 5.1). Данная сеть также обладает универсальными аппроксимационными способностями [37] . В СКК скрытые нейроны добавляются по одному, добавляемый нейрон подключается ко всем входам сети и всем ранее добавленным нейронам.

Обучение СКК производится вначале без скрытых нейронов. При параметрической идентификации СКК вместо функционала (2.2) используется коэффициент корреляции между выходом добавляемого нейрона и ошибкой выхода сети:

$$S = \sum_{j=1}^{r} \left| \sum_{i=1}^{k} \left(v^{(i)} - \overline{v} \right) \left(e_j^{(i)} - \overline{e}_j \right) \right| \to \max, \tag{5.6}$$

где $v^{(i)}$ – выход добавляемого нейрона; $e_j^{(i)}$ – ошибка j-го выходного нейрона СКК на i-ом примере; $\overline{v}$ и $\overline{e}_j$ – средние значения на обучающем множестве.

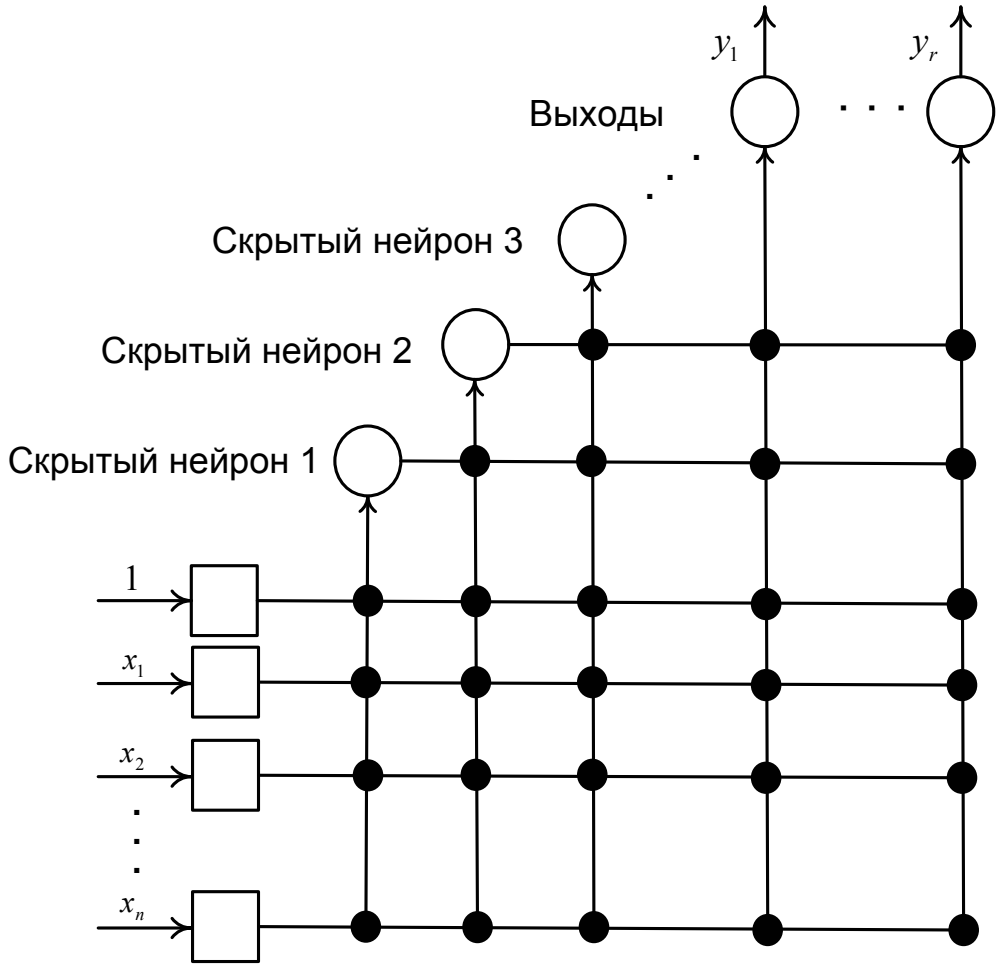


Рис. 5.1. Структура сети каскадной корреляции Фальмана

Фальманом разработана эффективная процедура максимизации (5.6) на основе расчета частных производных $\frac{S}{w_i}$. Этот алгоритм гарантирует увеличение значения коэффициента корреляции, хотя глобальность решения при этом не обеспечивается. Заметим, что данная процедура приводит к изменению весов только выходного слоя.

Процесс конструирования НСМ (впрочем, как и задача контрастирования) может быть формализован как задача поиска в пространстве состояний [36,38]. В такой интерпретации необходимо определить четыре элемента:

1. Пространство состояний.
2. Начальное состояние.
3. Критерий останова (целевое состояние).
4. Стратегию поиска.

Пространство состояний описывается множеством кортежей

$$S = \{s\} = \{< N, C, \Gamma, W >\},$$

где N – количество скрытых нейронов; C – ориентированный граф, задающий связь между входными, скрытыми и выходными нейронами; Γ – множество функциональных зависимостей скрытых нейронов (определяемых способом вычисления уровня активности и видом функции активации); W – параметры, соответствующие Γ. Таким образом, S определяет множество функций, реализуемых НСМ.

Начальным состоянием является НСМ без скрытых нейронов, т.е. линейная по параметрам модель. В отличие от конструктивного подхода в контрастивном подходе начальное состояние однозначно не определено.

Критерий останова должен сигнализировать о необходимости прекращения наращивания структуры НСМ при увеличении ошибки обобщения. Ошибка обобщения складывается из двух слагаемых:

$$\|f - f_{s,k}\| = \|f - f_s\| + \|f_s - f_{s,k}\|, \tag{5.7}$$

где f – моделируемая функция f, являющаяся в общем случае неизвестной; $f_{s,k}$ – НСМ, определяемая кортежем s, обученная на k примерах; f_s – ближайшая (идеальная) функция, реализуемая НСМ данной структуры. Слагаемое $\|f-f_s\|$ – ошибка аппроксимации; $\|f_s-f_{s,k}\|$ – ошибка оценивания. Было показано, что с увеличением количества нейронов в (5.7) ошибка аппроксимации уменьшается, а ошибка оценивания увеличивается [36]. Таким образом, необходим некоторый компромисс в критерии останова. Существуют несколько подходов к оценке ошибки обобщения, наиболее известным является использование тестового (валидационного) множества. В качестве критерия останова может использоваться останов, когда ошибка обучения при добавлении нейронов увеличивается незначительно.

Сутью конструктивных алгоритмов является стратегия поиска, т.е. алгоритм добавления новых нейронов и параметрическая идентификация новой НСМ. Пусть V – множество вершин, соответствующих НСМ; E – множество дуг, связывающих нейроны. Стратегия поиска состоит из двух компонентов:

1. Изменение пары множеств с (V_1, E_1) до (V_2, E_2).
2. Параметрическая идентификация весов для новой НСМ, определяемой парой (V_2, E_2).

Обычно $V_1 \subset V_2$, $E_1 \subset E_2$. Чаще всего $|V_2| = |V_1| + 1$, т.е. происходит добавление одного нейрона. Фактически это задача определения отображения переходов состояний

$$\Delta : S \to S.$$

В отличие от других задач поиска в простанстве состояний в данном случае ограничений на Δ в общем случае не накладывается, это может быть даже случайный переход в новое состояние, отображение может одно- и многозначным. Наиболее гибким является подход, при котором отображение является многозначным, т.е. на следующем шаге имеется несколько кандидатов структур, среди которых отбирается наилучшая. Как правило, на следующем шаге все равно должна остаться только одна структура НСМ.

Параметрическая идентификация НСМ при наращивании структуры может быть осуществлена путем независимого обучения новой НС, путем обучения весов вновь добавленных элементов, путем обучения новых весов с перерасчетом значений весов ранее имевшихся параметров. Идеология конструктивного подхода подразумевает, что обучение только новых весов непродуктивна (теряется смысл построения НСМ минимально возможной структуры), поэтому обычно корректируются и ранее добавленные веса. Классификация алгоритмов конструирования НСМ приведена на рис. 5.2 [36].

Хотя в [14] указано, что методы наращивания сети при большой размерности входного вектора имеют «относительно низкую эффективность» и «не являются серьезной альтернативой методам редукции сети», с этим можно не согласиться. Конструктивный подход представляется наиболее рациональным по следующим причинам [36]:

1. Однозначно определяется начальная структура НСМ.
2. С помощью конструктивных алгоритмов производится параметрическая идентификация НСМ сначала для сетей небольших размеров.

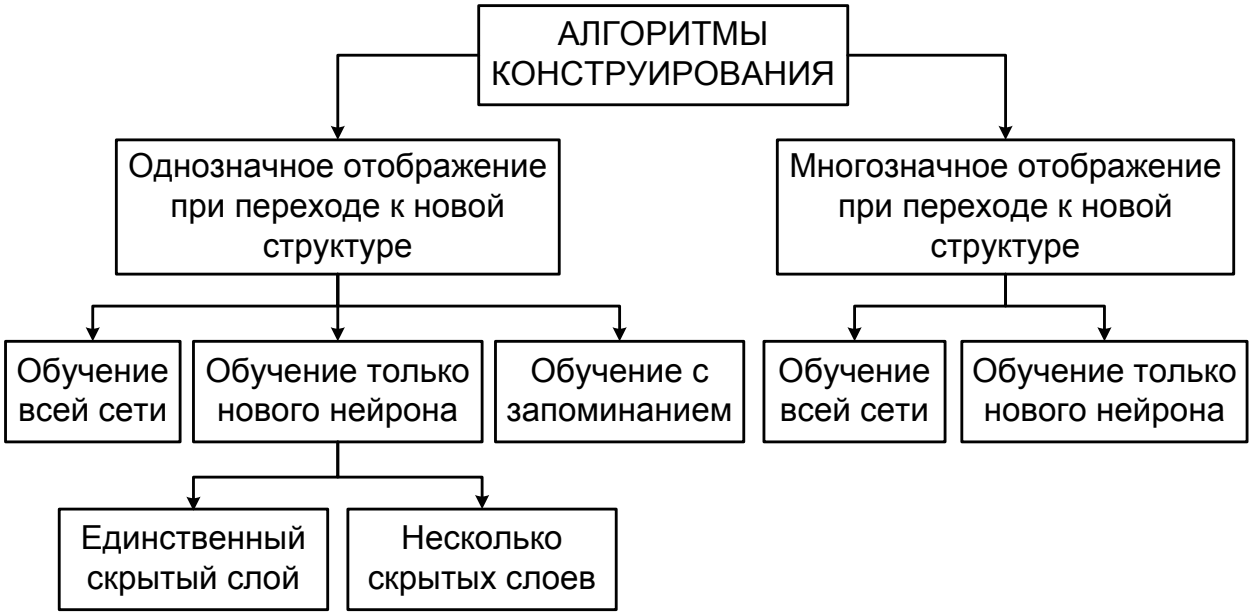


Рис. 5.2. Алгоритмы конструирования НС

3. Среди всех структур НСМ, решающих поставленную задачу, будет построена НСМ меньших размеров, чем при применении алгоритмов констрастирования.
4. Отсутствуют ошибки и вычислительная сложность в связи с тем, что нет необходимости вычислять вторые производные функционала качества для матрицы Гессе.

Низкая ошибка оценивания обеспечивается репрезентативностью обучающего множества. В отличие от СКК Фальмана при конструировании НСМ классической архитектуры отсутствуют методы, гарантирующие снижение ошибки аппроксимации при последовательном добавлении новых нейронов.

5.2. Блочные процедуры в конструктивном построении нейронных сетей

В [5] БРИП получили дальнейшее развитие – они были применены к задачам пошаговой регрессии, связанным с добавлением новых параметров в регрессионную модель $\phi(w, x)$. При этом добавление новых параметров $\widehat{w}$ производилось на основе суперпозиционного расширения исходной модели до

$\theta(\widehat{w}, w, x)$ в смысле

$$\theta(\widehat{w}, w, x) = \psi(\widehat{w}, \phi(w, x)).$$

Рассмотрим применение блочных процедур к последовательному наращиванию структуры одновыходной НСМ [6, 7]. Можно выделить два способа такого наращивания:

1. Добавление нейрона в последний скрытый слой (рис. 5.3);
2. Добавление нового слоя, состоящего из одного нейрона, перед выходным слоем (рис. 5.4).

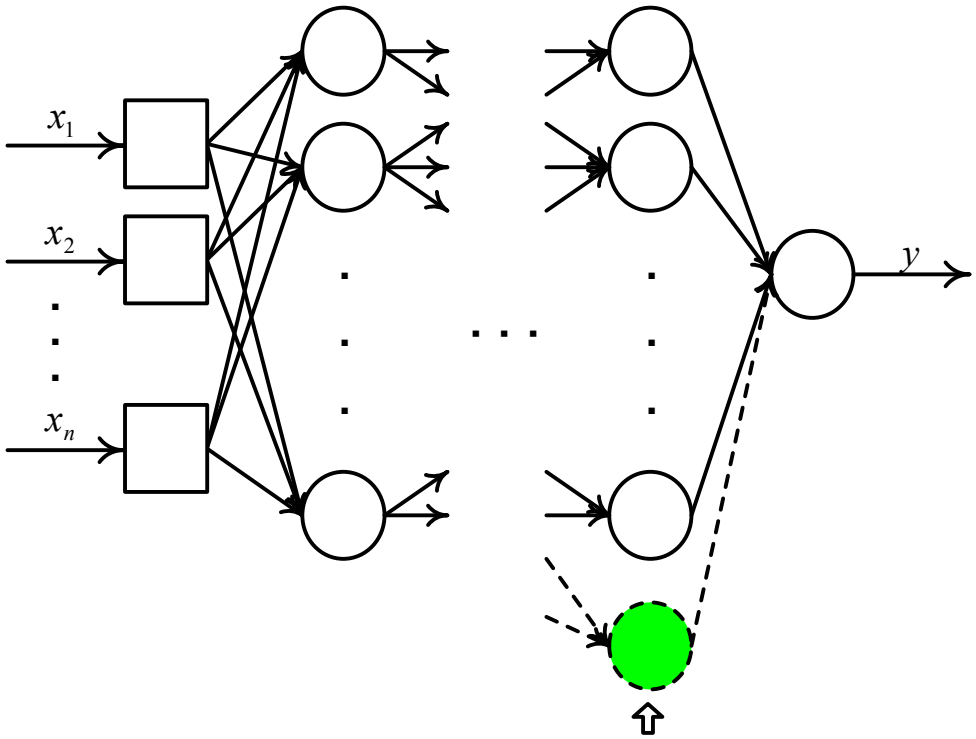


Рис. 5.3. Добавление нейрона в последний скрытый слой

Рассмотрим первый вариант (рис. 5.3). Новый нейрон приводит к появлению аддитивной добавки

$$\widehat{y}(\widehat{w}, w; x) = \widehat{w}_{q+1}^{(m)} f(\sum_{i=1}^{n_{m-2}} \widehat{w}_{0,q+1} + \widehat{w}_{i,q+1} y^{(m-2,i)}),$$

где $\widehat{w} \in \mathbb{R}^{n_{m-2}+2}$ – вектор, составленный из весов добавленного нейрона и веса от добавленного к выходному нейрону; $y^{(m-2,i)}$ – выход i-го нейрона $(m-2)$-го слоя. Здесь добавленная функция будет зависеть от части вектора весов

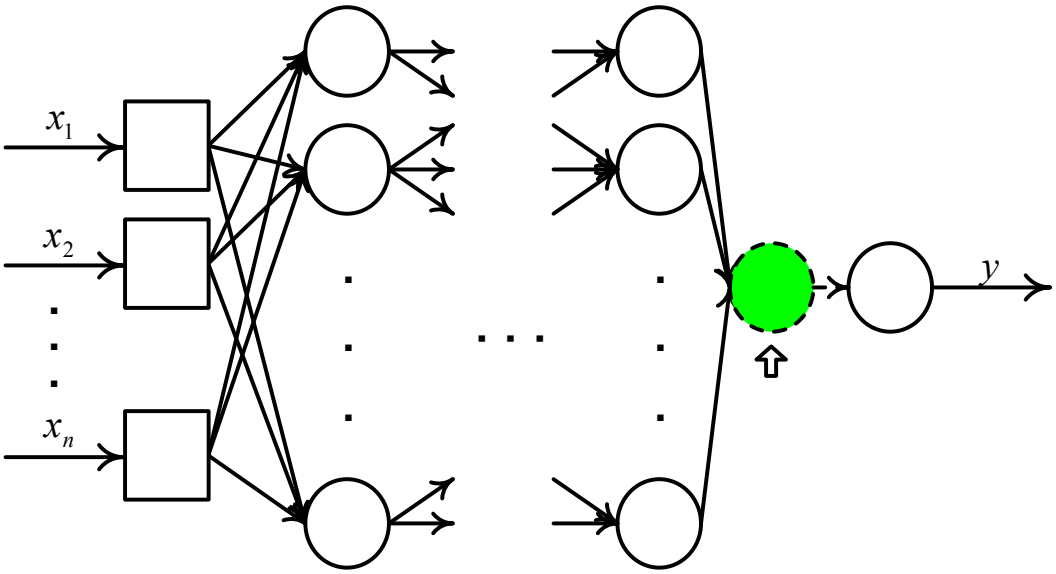


Рис. 5.4. Добавление слоя с одним нейроном перед выходным слоем

w предыдущей сети – весов нейронов скрытых слоев, находящихся с 1-го по $(m-2)$-й слои. Получившаяся сеть реализует функцию

$$y_{new}(\widehat{w}, w; x) = y(w; x) + \widehat{y}(\widehat{w}, w; x).$$

Благодаря тому, что вводится аддитивная добавка, корректировка весов может быть представлена в следующей форме:

$$\Delta\widehat{w} = L(y_{new}(\widehat{w}_t, w_t) - \widetilde{y}) = L(y(w_t) + \widehat{y}(\widehat{w}_t) - \widetilde{y});$$

$$\Delta w = \left(\nabla_w^T y_{new}\right)^+ \left(I - \nabla_{\widehat{w}}^T y_{new} L\right)(y_{new} - \widetilde{y}).$$

В связи с тем, что $\nabla_w^T y_{new} = \nabla_w^T y + \nabla_w^T \widehat{y}$ и $\nabla_{\widehat{w}}^T y_{new} = \nabla_{\widehat{w}}^T \widehat{y}$, получаем

$$\Delta w = \left(\nabla_w^T y + \nabla_w^T \widehat{y}\right)^+ \left(y + \widehat{y} - \widetilde{y} - \nabla_{\widehat{w}}^T \widehat{y} \Delta\widehat{w}\right). \tag{5.8}$$

Формула (5.8) показывает, как зависит пересчет вектора весов w в зависимости от весов добавленного нейрона.

Если нейрон добавляется в НС ПР стандартной структуры ($m = 2$), то $y^{(m-2,i)} = x_i$ – i-й вход, а добавка $\widehat{y}(\widehat{w}, w; x) = \widehat{y}(\widehat{w}; x)$, то есть не зависит от вектора w. Аналогично [5] получаем, что $\nabla_w^T y_{new} = \nabla_w^T y$:

$$\Delta w = \left(\nabla_w^T y\right)^+ \left(y + \widehat{y} - \widetilde{y} - \nabla_{\widehat{w}}^T \widehat{y} \Delta\widehat{w}\right) = (\Delta w)_y + \left(\nabla_w^T y\right)^+ \left(\widehat{y} - \nabla_{\widehat{w}}^T \widehat{y} \Delta\widehat{w}\right),$$

где $(\Delta w)_y$ – приращение, полученное для первоначальной сети $y(w,x)$, обозначенное так для отличия его от Δw в новой сети. Это соотношение показывает связь приращения вектора полученной НС в зависимости от приращения этого вектора в старой сети и приращения весов добавленного нейрона. Матрицы Якоби $\nabla_w^T y$ и $\nabla_{\widehat{w}}^T \widehat{y}$, а в случае многослойной сети и $\nabla_w^T \widehat{y}$, получаются на основе процедуры, аналогичной ОРО.

Перейдем к рассмотрению случая, когда добавляется новый слой из одного нейрона (рис. 5.4). Такой способ наращивания структуры сети реализуется суперпозицией функций

$$y_{new}(\widehat{w}, w; x) = \widehat{w} f(y(w,x)).$$

Заметим, что $\nabla_w^T y_{new} = \widehat{w} D_k \nabla_w^T y$, где $D_k = \operatorname{diag}\left\{f'_{y_i}(y_i)\right\}_{i=1}^k$ – диагональная матрица порядка k, составленная из производных функций активации по своему аргументу на обучающем множестве, а $\nabla_{\widehat{w}}^T y_{new} = \Psi$. Поэтому

$$\Delta w = \left(\widehat{w} D_k \nabla_w^T y\right)^+ (I - \Psi L)\,(y_{new} - \widetilde{y}) = \\ = \left(\nabla_w^T y\right)^+ D_k^+ \widehat{w}^+ \left(y + \widehat{y} - \widetilde{y} - \Psi \Delta \widehat{w}\right), \quad (5.9)$$

где $\widehat{w}^+$ – скаляр ($\widehat{w} = 0$ выполняется лишь в вырожденных случаях), а матрица $D_k^+ = \operatorname{diag}\left\{\left[f'_{y_i}(y_i)\right]^+\right\}_{i=1}^k$ также легко определяется.

На базе БРИП могут быть реализованы процедуры адаптивного обучения нейросетевой модели при поступлении новых данных [3, 6]. Задача адаптации модели на новых данных в некотором смысле двойственна задаче последовательного наращивания структуры НС. Действительно, в первом случае формируемая матрица Якоби пополняется новыми строками (соответствует появлению новых строк), а во втором – новыми столбцами (соответствует появление новых параметров, подлежащих оптимизации).

Заключение

Выбор метода обучения следует выбирать исходя из решаемой задачи, учитывать оперативный или аналитический характер задачи. В данной монографии предложены методы обучения, которые целесообразно применять для построения сложных аналитических зависимостей, когда имеются достаточные временные ресурсы, а наибольшее значение имеет качество построения моделей. Тем не менее только построения адекватной модели не всегда достаточно.

В ряде областей знаний, например в медицине, для максимального доверия результатам нейросетевого моделирования требуется их лингвистическая интерпретация. В других предметных областях указанное требование является необязательным, но весьма желательным. В таких случаях имеет смысл применять легко интерпретируемые нейроподобные конструкции наподобие нейро-нечетких систем или применять алгоритмы извлечения знаний из нейросетевых моделей. Существующие сегодня методы интерпретации ориентированы лишь на ограниченный класс нейросетевых моделей.

Библиографические ссылки

1. *Алберт А.* Регрессия, псевдоинверсия и рекуррентное оценивание. — М.: Наука, 1977. — 224 с.

2. *Блюмин С. Л., Миловидов С. П.* Псевдообращение: учеб. пособие. — Воронеж: ВорПИ-ЛипПИ, 1990. — 72 с.

3. *Блюмин С. Л., Миловидов С. П., Погодаев А. К.* Нелинейный метод наименьших квадратов и псевдообращение: учеб. пособие. — Липецк: ЛипПИ, 1992. — 80 с.

4. *Блюмин С. Л., Погодаев А. К.* Блочные рекуррентно-итерационные процедуры решения нелинейной задачи о наименьших квадратах // *Журнал вычислительной математики и математической физики.* — 1992. — Т. 32, № 8. — С. 1180–1186.

5. *Блюмин С. Л., Погодаев А. К.* Суперпозиционная регрессия // *Журнал вычислительной математики и математической физики.* — 1995. — Т. 35, № 10. — С. 1576–1581.

6. *Блюмин С. Л., Сараев П. В.* Адаптивное рекуррентно-итерационное обучение искусственных нейронных сетей // Современные проблемы информатизации в технике и технологиях: тр. VI Междунар. открытой науч. конф. — Воронеж: ВЭПИ, 2001. — С. 87–88.

7. *Блюмин С. Л., Сараев П. В.* Рекуррентно-итерационные процедуры для адаптивного конструирования нейронных сетей // Нейроинформатика и ее приложения: мат. IX Всерос. семинара. — Красноярск: ИПЦ КГТУ, 2001. — С. 20–21.

8. *Гаврилова Т. А., Хорошевский В. Ф.* Базы знаний интеллектуальных систем. — СПб.: Питер, 2001. — 384 с.

9. *Голуб Д., Ван Лоун Ч.* Матричные вычисления. — М.: Мир, 1999. — 548 с.

10. *Горбань А. Н., Россиев Д. А.* Нейронные сети на персональном компьютере. — Новосибирск: Наука. Сибирская издательская фирма РАН, 1996. — 276 с.

11. Достоверные вычисления. Базовые численные методы / У. Кулиш [и др.]. — М.-Ижевск: Регулярная и хаотическая динамика, 2005. — 496 с.

12. Нейроинформатика / А. Н. Горбань [и др.]. — Новосибирск: Наука. Сибирское предприятие РАН, 1998. — 296 с.

13. *Орлянская И. В.* Современные подходы к построению методов глобальной оптимизации // *Исследовано в России.* — 2002. — С. 2097–2108. Режим доступа: http://zhurnal.ape.relarn.ru/articles/2002/189.pdf.

14. *Осовский С.* Нейронные сети для обработки информации. — М.: Финансы и статистика, 2002. — 344 с.

15. Прикладной интервальный анализ / Л. Жолен [и др.]. — М.-Ижевск: Институт компьютерных исследований, 2007. — 468 с.

16. *Рутковская Д., Пилиницкий М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы. — М.: Горячая линия – Телеком, 2006. — 452 с.

17. *Сараев П. В.* Использование псевдообращения в задачах обучения искусственных нейронных сетей // *Исследовано в России: электронный журнал.* — 2001. — Т. 29. — С. 308–317. Режим доступа: http://zhurnal.ape.relarn.ru/articles/2001/029.pdf.

18. *Сараев П. В.* Обучение искусственных нейронных сетей: учет линейно-нелинейной структуры // *Вестник молодых ученых.* — 2002. — Т. 12, № 2. — С. 45–51.

19. *Сараев П. В.* Исследование псевдообращения матриц в математическом пакете maple // Системы компьютерной математики и их приложения: мат. XI Междунар. науч. конф., посвященной 70-летию профессора В.П. Дьяконова. — № 11. — Смоленск: Изд-во СмолГУ, 2010. — С. 62–65.

20. *Сараев П. В.* Обобщающая способность нейронных сетей прямого распространения // Управление большими системами: сб. тр. VII Всерос. шк.-конф. молодых ученых. — Т. 2. — Пермь: Пермский государственный технический университет, 2010. — С. 346–355.

21. *Уоссермен Ф.* Нейрокомпьютерная техника. — М.: Мир, 1992. — 184 с.

22. *Хайкин С.* Нейронные сети: полный курс. — М.: Вильямс, 2006. — 1104 с.

23. *Шарый С. П.* Конечномерный интервальный анализ. — 2010. — 603 с. Режим доступа: http://www.nsc.ru/interval.

24. *de Weerdt E., Chu Q. P., Mulder J. A.* Neural network global optimization using interval analysis // 27th Benelux Meeting on Systems and Control. — Heeze, The Netherlands: Netherlands Organisation for Scientific Research, 2008. — P. 162.

25. *de Weerdt E., Chu Q. P., Mulder J. A.* Neural network output optimization using interval analysis // *IEEE Transacions on Neural Networks.* — 2009. — Vol. 20, № 4. — P. 638–653.

26. *Duch W., Korczak J.* Optimization and global minimization methods suitable for neural networks: Tech. Rep. KMK UMK Technical Report 1/99: Neural Computing Surveys, 1998. Access mode: http://www.fizyka.umk.pl/publications/kmk/99globmin.pdf.

27. *Escarcina R. E. P., Bedregal B. R. C., Lyra A.* Interval computing in neural networks: One layer interval neural networks // *CIT.* — 2004. — P. 68–75. Access mode: http://dx.doi.org/10.1007/978-3-540-30561-3_8.

28. *Fahlman S. E., Lebiere C.* The cascade-correlation learning architecture // Advance in Neural Information Processing Systems. — Morgan-Kaufmann, Los Altos C, 1990. — P. 524–532.

29. *Fuller R.* Introduction to Neuro-Fuzzy Systems. — Berlin/Heildelberg: Springer-Verlag, 2000. — 289 p.

30. *Hansen E., Walster G. W.* Global optimization using interval analysis. — New York: Marcel Dekker, 2003. — 492 p.

31. *Hansen E. R.* Global optimization using interval analysis: The one-dimensial case // *Journal of Optimization Theory and Applications.* — 1979. — Vol. 29, № 3. — P. 331–344.

32. *Hansen E. R.* Global optimization using interval analysis – the multi-dimensial case // *Numerical Mathematics.* — 1980. — Vol. 34. — P. 247–270.

33. *Hu C.* A parallel software package for nonlinear global optimization // Proceedings of the 5th International Conference on Optimization: Techniques and Applications. — Hong Kong, 2001. — P. 1030–1037.

34. *Hu C., Xu S., Yang X.* A review on interval computation - software and applications // *International Journal of Computational and Numerical Analysis and Applications.* — 2002. — Vol. 1, № 2. — P. 149–162.

35. *Iwaarden R. J. V.* An improved unconstrained global optimization algorithm: PhD thesis. — Denver: University of Colorado, 1996. — 120 p.

36. *Kwok T.-Y., Yeung D.-Y.* Constructive algorithms for structure learning in feedforward neural networks for regression problems // *IEEE Transactions on Neural Networks.* — 1997. — № 8(3). — 630–645 p.

37. *Kwok T. Y., Yeung D. Y.* A theoretically sound learning algorithm for constructive neural networks // Proceedings of the IEEE International Symposium on Speech, Image Processing and Neural Networks. — Hong Kong, 1994. — P. 389–392.

38. *Kwok T. Y., Yeung D. Y.* Constructive neural networks: Some practical considerations // Proceedings of the IEEE International Conference on Neural Networks (ICNN). — Orlando, Florida, USA, 1994. — P. 198–203.

39. *Liu M.* Interval standard neural network models for nonlinear systems // *Journal of Zhejiang University SCIENCE A*. — 2006. — №7(4). — P. 530–538.

40. *Sanquing H., Wang J.* Global robust stability of a class of discrete-time interval neural networks // *IEEE Transactions on circuits and systems*. — 2006. — Vol. 53, № 1. — P. 129–138.

41. *Shary S. P.* A surprising approach in interval global optimization // *Reliable Computing*. — 2001. — № 7. — P. 497–505.

Научное издание

Сараев Павел Викторович

Идентификация нейросетевых моделей

Монография

Редактор М.Ю. Копытина

Подписано в печать 26.06.12. Формат 60x84 1/16. Бумага офсетная.

Ризография. Печ.л. 5,9. Тираж 100 экз. Заказ № 332.

Издательство Липецкого государственного технического университета.

Полиграфическое подразделение Издательства ЛГТУ.

398600 Липецк, ул. Московская, 30.