Information Sources module is a WEB-based document collection regarding various fields of interest. An expert, who will be the owner of the source, can add an information source.

The Forum Module provides further discussion regarding an information source where members can share information, experience and opinions.

Non-members of the centre will be able to get information about this Competence Centre, browse the categories available, but not access them, or the discussion forum. Of course, they will be able to register on-line and get access to all the information.

The community members are divided into three categories:

- Administrators: They add other administrators and experts, remove other users from the community, to change the owners of an information source, delete information sources and entries in any discussion forum).

- Experts: They can add new information sources, they are considered to be the "owners" of the information source they have added (they can modify or delete it), and they can share the ownership of an information source with other experts. They are also allowed to create test question (and modify and delete them) for the training materials that they own. If they post an entry in a discussion forum regarding an information source that they own, their entry is shown in a different, high contrast font, being considered as an "authorized entry".

- Users: This category of community members represents the "knowledge seekers". They can access here information sources, access the discussion forums, and contact the "owners" of an information source, or the poster of an entry in a discussion forum, for additional information.

Because the environment uses a modular approach, it is easy to maintain, update, develop or reuse it. The object-oriented facilities offered by the language PHP allows that also the modules have this orientation. They are easier and faster even a module-level upg.

CONCLUSION

Virtual forms of co-operation and communication like VCCs which support processes of KM and Internet-based ODL have the future before them.

Within our EU-co-operation we would like to work out didactical principles of Internet-based ODL and to expand our VCC for other European countries.

REFERENCES

[1] *Brodner P., Hamburg I.* (Eds.) Strategische Wissensnetze: Wie Unternehmen die Resource Wissen nutzen, Schneider, Gelsenkirchen, 1999.

[2] *Brodner P. Helmstadter E. Widmaier B.* (Hg.) Wissensteilung - zur Dynamik von Innovation und kollektivem Lernen, Munchen/Mering. 1999.

[3] Probst, G. J. B. Wissen managen: wie Unternehmen ihre wertvollste Ressource optimal nutzen, Frankfurt am Main. New York. 1998.

[4] HART H., 1999: Knowledge transfer and knowledge management in a global organisation // P. Brodner, I. Hamburg (Hrsg.), Strategische Wissensnetze: Wie Unternehmen die Ressource Wissen nutzen, Schneider, Gelsenkirchen, 27-46.

[5] *Engert S., Hamburg I., Terstriep J.* Kompetenznetzwerke zur Kontextsteuerung von Wissensteilung: Ein Beispiel. 1999.

[6] *Engert S., Hamburg I., Terstriep J.* Web based Training: Experience within a German Project. // Supporting Learning Communities in Education, Paper presented at the i3 Spring Days 2000, Athens.

[7] *Pallof R., Pratt K.* Building Learning Communities in Cyberspace, San Francisco, 1999.

[8] *Covey S.*, The Seven Habits of Highly Effective People: Powerful Lessons in Personal Change, Fireside, New York 1989.

[9] *Schmidtmann A.,Felser W.* Virtuelle Kompetenzzentren. // WIRTSCHAFTSINFORMATIK , 1999, 41 6, S. 554-560.

УДК 681.32:007.58

ОБЗОР СОВРЕМЕННЫХ ТИПОВ НЕЙРОННЫХ СЕТЕЙ

М.В.Аникеев, Л.К.Бабенко, О.Б.Макаревич

В работе рассматривается современное состояние теории нейронных сетей. Проведен обзор некоторых наиболее часто применяемых в настоящее время нейросетевых архитектур. Рассмотрены различные подходы к обучению, проектированию и настройке нейронных сетей. Особое внимание уделено различным вариантам метода обратного распространения ошибки.

The current theory of neural networks is considered in this paper. Some of the most widely used architectures of neural networks are reviewed. Various methods of training and design are analyzed. Particular attention is paid to the backpropagation method of training.

ВВЕДЕНИЕ

Возможности современных компьютеров позволяют производить различные вычисления со скоростью на десятки порядков превышающей возможности человеческого мозга. Однако ряд даже тривиальных для человека задач, не связанных с вычислениями, остается весьма сложным для вычислительной техники. Способность человека к ассоциативному хранению информации, обучению, обобщению и обработки информации с учетом контекста остается непревзойденной даже для современных суперкомпьютеров. Целью проектирования искусственных нейронных сетей (ИНС) является построение вычислительной структуры или алгоритма, работающего по принципам естественного интеллекта. К основным можно отнести следующие свойства нейронных сетей.

1.Нейронные сети, по аналогии с мозгом человека и животных, строятся из множества простых элементов, выполняющих элементарные действия и соединенных между собой различными связями.

2.Нейронные сети способны совершенствовать свою работу (обучаться или адаптироваться), используя примеры.

3.Нейросетевое решение задачи не требует от разработчика формулировки алгоритма решения поставленной задачи и его программирования. Нейронные сети, как правило, используют примеры "правильной" работы для построения своего метода решения задачи. При этом существует возможность выявления сетью скрытых закономерностей в задаче, неизвестных разработчику.

1. ОСНОВНЫЕ ПОНЯТИЯ АРХИТЕКТУРЫ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Основным элементом искусственных нейронных сетей является аппаратный или программный узел, имитирующий работу нейрона. Обобщенная схема базовой модели нейрона представлена на рис. 1.1.

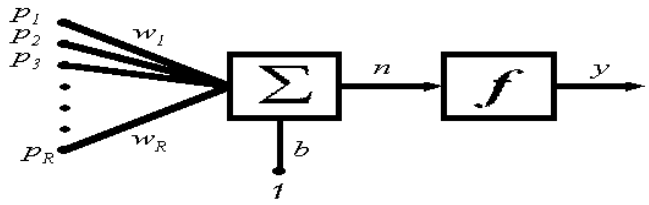

Рисунок 1.1

Здесь $\{p_1, p_2, \ldots, p_R\} \equiv \mathbf{p}$ - сигнал, подаваемый на вход нейрона, $\{w_1, w_2, \ldots, w_R\} \equiv \mathbf{w}$ - вектор весовых коэффициентов нейрона, b - коэффициент смещения (порог) нейрона, f - функция активации, y - выходной сигнал. Таким образом, нейрон реализует функцию $y = f(n)$, где $n = \sum_{j=1}^{\kappa} p_j w_j + b$. Обобщенная функция нейрона в векторной форме записывается следующим образом

$$y = f((\mathbf{w} \cdot \mathbf{p}) + b). \qquad (1.1)$$

Перечислим основные типы активационных функций, используемые в ИНС.

1.*Пороговые* функции. Используются при проектировании сетей персептронного типа, предназначенных для решения задач линейной классификации. Пороговая функция существует в двух модификациях: асимметричная, описываемая формулой

$$f(n) = \begin{cases} 0, & \text{если } n < 0 \\ 1, & \text{если } n \geq 0 \end{cases}, \qquad (1.2)$$

и симметричная

$$f(n) = \begin{cases} -1, & \text{если } n < 0 \\ 1, & \text{если } n \geq 0 \end{cases}. \qquad (1.3)$$

2.*Линейная функция*. Используется в различных классах сетей для линейной аппроксимации линейных и нелинейных функций. Выражается формулой

$$f(n) = n. \qquad (1.4)$$

3.*Сжимающие функции*. Существуют в двух основных модификациях: сигмоидальной (логистической)

$$f(n) = \frac{1}{1 + e^{-n}} \qquad (1.5)$$

и тангенциальной

$$f(n) = \mathrm{th}(n) = \frac{e^{2n} - 1}{e^{2n} + 1}. \qquad (1.6)$$

Смысл использования сжимающих функций состоит в преобразовании входного сигнала из бесконечного диапазона в конечный: [0, 1] в первом, и [-1, 1] во втором случае. Кроме того, сжимающие функции обладают полезным свойством - они непрерывны и дифференцируемы на всей области действительных чисел, что позволяет обучать сети нейронов со сжимающими активационными функциями методом обратного распространения ошибки. В [1] предложена функция (1.7), обладающая всеми достоинствами логистической функции (дифференцируемость, конечный интервал значений, возрастание, простая производная), но просчитываемая быстрее на вычислительной технике.

$$f(n) = \frac{n}{1 + |n|}. \qquad (1.7)$$

4.*Радиально-базисная* функция (Гауссиана). Используется в радиально-базисных сетях и описывается формулой

$$f(n) = e^{-n^2}.$$

Эта функция достигает своего максимума при *n=0* и поэтому применяется для определения степени близости двух векторов - входного **p** и вектора весовых коэффициентов **w**.

Применяемые в нейронных сетях активационные функции не ограничиваются приведенным выше списком. Однако представленные четыре класса функций охватывают подавляющее число нейронных сетей.

Слоем нейронов называется совокупность параллельно подключенных нейронов, обрабатывающих в один и тот же момент времени один входной вектор. Выход нейронного слоя выражается формулой

$$\bar{y}_{S\times 1}(\bar{p}_{R\times 1}) = f(W_{S\times R}\bar{p}_{R\times 1} + \bar{b}_{S\times 1}). \qquad (1.8)$$

Матрица **W** имеет размерность $S \times R$, где S - число нейронов в слое, а R - размерность входного вектора.

Все нейронные сети можно разделить на два больших

класса - сети прямого распространения и сети с обратными связями. Сети прямого распространения представляют собой упорядоченные последовательности нейронных слоев, в которых сигналы передаются последовательно от входного слоя к выходному. Слои, расположенные между входным и выходным слоями называются скрытыми (иногда входной слой называют первым скрытым). В сетях с обратными связями, как видно из их названия, сигналы могут передаваться от последующих слоев к предыдущим, а также между нейронами одного и того же слоя.

Для успешного функционирования нейронной сети необходимо соответствующим образом настроить ее весовые коэффициенты. Для некоторых типов сетей (например, для адаптивных линейных сетей, описанных в гл. 3) существуют методы построения обученных конфигураций из постановки задачи. Однако подавляющее большинство нейронных сетей необходимо обучать после построения. Существуют две основные парадигмы обучения: обучение "с учителем" (supervised manner) и обучение "без учителя" (unsupervised manner) [2-4].

Обучение "с учителем" характерно для сетей прямого распространения. В процессе обучения "с учителем" сети последовательно предъявляются пары $\langle \mathbf{p}_i, \mathbf{d}_i \rangle$, где $\mathbf{p}_i$ - входной вектор, $\mathbf{d}_i$ - желаемый выходной вектор. При этом сеть пытается настроить свои веса таким образом, чтобы при предъявлении на вход вектора $\mathbf{p}_i$, получать на выходе вектор максимально близкий к $\mathbf{d}_i$. После окончания процедуры обучения можно надеяться, что нейронная сеть правильно отреагирует на входные вектора, не входящие в обучающую выборку. Способ обучения "без учителя" характерен для сетей с обратными связями. Получая входное воздействие, сеть подстраивает свои веса согласно своим внутренним целям. Например, соревновательные сети (гл. 6) в процессе работы учатся относить близко расположенные входные вектора к одному классу.

2. ОДНОСЛОЙНЫЕ ПЕРСЕПТРОНЫ

Однослойными персептронами называются однослойные сети прямого распространения, содержащие нейроны, которые активизируются только пороговыми функциями. Правило обучения однослойного персептрона формулируется следующим образом. Пусть имеется множество пар $\{\langle \mathbf{p}_1, \mathbf{d}_1 \rangle, \langle \mathbf{p}_2, \mathbf{d}_2 \rangle, \ldots, \langle \mathbf{p}_Q, \mathbf{d}_Q \rangle\}$, называемое обучающей выборкой. Здесь $\mathbf{p}_Q$ - входной сигнал, $\mathbf{d}_Q$ - желаемый отклик персептрона на соответствующее входное воздействие, называемый целью ($q = 1 \ldots Q$). Поскольку все нейроны активизируются пороговой функцией, координаты векторов $\mathbf{d}_q$ должны принимать два устойчивых значения (обычно 0 и 1). Первоначально, веса и смещения принимают любые случайные значения, соизмеримые с величинами координат входных векторов из обучающей выборки, или нулевые значения. Процесс обучения состоит в последовательном предъявлении сети векторов $\mathbf{p}_q$. Введем для каждого нейрона понятие ошибки $e = d - y$, где d - цель, y - фактический выход нейрона. Правило обучения однослойного персептрона можно записать следующим образом:

$$\Delta W_{S\times R} = (\bar{d}_{S\times 1} - \bar{y}_{S\times 1})(\bar{p})^T_{1\times R} = \bar{e}_{S\times 1}(\bar{p})^T_{1\times R}. \quad (2.1)$$

В процесс обучения можно включить также смещения нейронов, принимая их за веса, на вход которых всегда подается 1. Для смещений формулируется следующее правило обучения:

$$\Delta \bar{b}_{S\times 1} = \bar{d}_{S\times 1} - \bar{y}_{S\times 1} = \bar{e}_{S\times 1}. \quad (2.2)$$

Итерацию алгоритма обучения, включающую в себя последовательное предъявление сети всех пар обучающей выборки, принято называть эпохой (epoch). Процесс модификации весов и смещений останавливается на той эпохе, которая не вызвала никаких изменений в настройке сети. Если образцы, предъявляемые однослойному персептрону, линейно разделяемы, то такая процедура обучения гарантирует, что персептрон построит классификатор за конечное число итераций. Для персептронов весьма важной является возможность настройки смещений в нейронах, поскольку при фиксированных смещениях (в частности при $b=0$ для всех нейронов) гиперплоскости классификатора привязываются к определенной точке (к началу координат). Такое ограничение не гарантирует сходимость процедуры обучения, даже если задача классификации линейно разрешима. Это происходит из-за того, что при исключении смещений из рассмотрения, сеть ограничивается классификаторами вида $a_1p_1 + a_2p_2 + \ldots + a_np_n = 0$ в базисе $\{p_1, p_2, \ldots, p_n\}$, а все множество гиперплоскостей n-мерного пространства описывается уравнениями вида

$$a_1p_1 + a_2p_2 + \ldots + a_np_n + a_{n+1} = 0.$$

Каждый нейрон в однослойном персептроне описывает одну гиперплоскость в пространстве входных векторов и своим выходным значением, определяет по какую сторону от этой гиперплоскости находится поданный на вход вектор. Таким образом, при проектировании однослойных персептронов необходимо выбирать количество нейронов, равное количеству гиперплоскостей, достаточному для классификации всего пространства входного воздействия. Кроме того, персептроны не могут решать задачи классификации, если классы не могут быть разделены между собой при помощи линейных функций. Например, из множества всех возможных булевых функций двух переменных, однослойный персептрон не может реализовать "эквивалентность" и "исключающее ИЛИ". Поскольку однослойные персептроны могут распознавать только линейно разделяемые образы, они используются только в простейших случаях классификации. Для более сложных случаев обычно применяются другие типы нейронных сетей - адаптивные линейные сети и сети, обучаемые методом обратного распространения ошибки.

3. СЕТИ АДАПТИВНЫХ ЛИНЕЙНЫХ ЭЛЕМЕНТОВ

Адаптивными линейными элементами (ADALINE = ADAptive LINear Element) называются нейроны с линейной функцией активации [2]. Однослойные сети прямого распространения, состоящие из нескольких адаптивных линейных элементов, иногда объединяют под термином MADALINE (Multi ADALINE).

Сети ADALINE, как и персептроны, способны решать линейно разделяемые задачи классификации образов. Однако диапазон выходных значений нейронов в сетях ADALINE не ограничен только двумя выходными значениями. Это свойство позволяет обучать такие сети методом наименьших среднеквадратических отклонений, известным в литературе как алгоритм LMS (Least Mean Square) [2, 5], называемым также алгоритмом Уидроу-Хоффа или дельта-правилом [6]. Этот метод строит из сети более точный линейный классификатор, чем правило обучения однослойного персептрона. Метод LMS заключается в минимизации среднеквадратической ошибки сети. Функция ошибки (mse = Mean Square Error) выражается формулой

$$mse = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q} (d(k)-y(k))^2 . \qquad (3.1)$$

Доказано, что функция ошибки в сетях ADALINE с любой обучающей выборкой представляет собой R-мерную параболу, где R - число входных линий сети. Поэтому поверхность ошибки имеет не более одного минимума. Изменение весов и смещений сети согласно методу LMS выражается следующими формулами:

$$W(k+1)_{S\times R} = W(k)_{S\times R} + 2\eta\bar{e}(k)_{S\times 1}\bar{p}^T(k)_{1\times R}; \quad (3.2)$$

$$b(k+1)_{S\times 1} = W(k)_{S\times 1} + 2\eta\bar{e}(k)_{S\times 1}.$$

Здесь η - скорость обучения. Чем больше η, тем быстрее происходит обучение, однако слишком большое значение ведет к нестабильности алгоритма. Несмотря на то, что алгоритм LMS работает только для однослойных линейных сетей, это его свойство не снижает универсальности, поскольку добавление других линейных слоев к сетям ADALINE не улучшает характеристик сетей и, следовательно, бесполезно.

Адаптивные линейные сети дают хорошие результаты, только в случае, если зависимость между входными и выходными значениями сети линейна или линейно аппроксимируема. Если линейные сети не способны решать задачу классификации, аппроксимации или предсказания, поставленную перед ними, то для решения этой задачи могут быть использованы нелинейные сети прямого распространения, обучаемые методом обратного распространения ошибки.

4. ОБРАТНОЕ РАСПРОСТРАНЕНИЕ ОШИБКИ

Метод обратного распространения ошибки (ОРО, Backpropagation) был разработан как обобщение метода Уидроу-Хоффа для многослойных сетей с нелинейными дифференцируемыми функциями активации. В современном виде метод был впервые формализован в [7].

4.1. Базовый алгоритм ОРО [3, 8, 9]

Принцип ОРО основан на настройке сети таким образом, чтобы минимизировать среднеквадратическую ошибку на обучающей выборке. Для этого используется понятие градиента функции ошибки. Сети последовательно предоставляются входные вектора из обучающей выборки, и, начиная с последнего слоя, вычисляются градиенты функции ошибки для каждого нейрона. Веса и смещения изменяются в направлении, противоположном градиенту. Поэтому базовый алгоритм ОРО еще часто называют методом градиентного спуска (gradient descent).

Приведем базовый алгоритм ОРО в виде последовательности шагов:

1.Все веса и смещения инициализируются маленькими ненулевыми величинами обычно из интервала от -1 до +1.

2.Представление входного вектора **p** и желаемого выходного вектора **t**.

3.Расчет фактического выхода сети **a** при подаче на вход вектора **p**.

4.Настройка весов и смещений по формуле:

$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) + \eta\delta_i^{(k)}y_j^{(k-1)}. \qquad (4.1)$$

Здесь $\delta_i^{(k)}$ - вычисляется для выходного слоя по формуле:

$$\delta_i^{(K+1)} = f'(y_i^{(K+1)})(d_i^{(K+1)} - y_i^{(K+1)}), \qquad (4.2)$$

а для скрытых слоев по рекуррентной формуле:

$$\delta_i^{(k)} = f'(y_i^{(k)})\sum_j \delta_j^{(k+1)}w_{ji}^{(k+1)} \quad (k = 1, \ldots, K). \quad (4.3)$$

Здесь η - скорость обучения (небольшая положительная постоянная величина, обычно берущаяся из интервала от 0 до 1), $f'(y_i^{(k)})$ - производная активационной функции i-го нейрона, находящегося на k-м слое. Если используются нейроны с сигмоидной функцией активации, то для них

$$f'(y_i^{(k)}) = y_i^{(k)}(1-y_i^{(k)}). \qquad (4.4)$$

Процедура настройки весов происходит в направлении от выходного слоя к первому, отсюда и название метода.

5.Если не удовлетворено условие окончания обучения, то переход к шагу 2.

Существует множество различных модификаций базово-

го алгоритма ОРО (градиентного спуска). В [10] они условно разделены на пять групп:

1) алгоритмические изменения (ОРО с моментом [2, 11], Quickprop [10, 11], методы второго порядка [2, 4, 10], адаптация скорости обучения [10, 12, 13], RPROP [10, 14]);

2) эвристики, применяемые к алгоритму (эвристические методы инициализации весов [2], оценка оптимальной скорости обучения);

3) регулировки (падение весов [11, 15]);

4) модификации сетей (нестандартные нейроны и активационные функции, рекуррентные узлы в сетях прямого распространения);

5) сети с динамически изменяющейся структурой (каскадная корреляция [6]).

Рассмотрим некоторые наиболее популярные модификации алгоритма ОРО.

4.2. Пакетная модификация [2, 11]

В стандартном (пошаговом) алгоритме ОРО веса и смещения модифицируются после каждого предъявления обучающего вектора. При использовании пакетной модификации, веса и смещения обновляются только после окончания эпох. То есть после предъявления очередного обучающего вектора изменения весов вычисляются, но они не влияют на конфигурацию сети, а суммируются. Только после предъявления последнего вектора, полученные суммы прибавляются к текущим весам. Пакетная модификация не гарантирует ускорения сходимости метода, но избавляет процесс обучения от его зависимости от порядка предъявления обучающих пар. Если алгоритм ОРО реализуется в пошаговом режиме, то желательно на каждой эпохе случайным образом менять порядок предъявления обучающих векторов.

4.3. Градиентный спуск с моментом [2, 10, 11]

Стандартный метод ОРО часто сходится очень медленно, двигаясь вдоль плоских участков поверхности ошибки. Для ускорения сходимости алгоритма предлагается ввести понятие взвешенного момента. Выражение для изменения весов в этой модификации будет следующим:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) + \alpha \Delta w_{ij}(t-1). \qquad (4.5)$$

Здесь t - номер очередной итерации обучения, w_{ij} - рассматриваемый вес (или смещение), Δw_{ij} - изменение веса (смещения), предлагаемое стандартным алгоритмом ОРО, α - константа момента. Значение константы момента должно лежать в диапазоне от 0 до 1 (обычно ? выбирается равным 0,9).

4.4. Разрушение весов [16]

При обучении часто возникает проблема недопустимо низкого уровня обобщающей способности нейронных сетей. То есть сеть может давать на выходе низкий уровень ошибок для данных обучающей выборки, но для других данных значения ошибок будут недопустимо высоки. Такая ситуация обычно вызывается двумя причинами: переобученностью или избыточным количеством нейронов скрытых слоев. Переобученность сети (overfitting) возникает из-за стремления обучающего алгоритма максимально приблизить значение среднеквадратической ошибки к нулю на обучающей выборке. При этом сеть может потерять способность получать правильные выходные данные для данных, не входящих в обучающую выборку. Если низкая обобщающая способность возникает из-за избыточного количества нейронов, то достаточно сложно точно оценить от какого числа нейронов можно отказаться, чтобы сеть сохранила в себе способность решать требуемую задачу.

Универсальный подход к решению проблемы низкой обобщающей способности, называемый разрушением весов (weight decay) был предложен в [16]. Метод заключается в вычитании из каждого весового коэффициента незначительной части перед модификацией веса по алгоритму ОРО, согласно формуле (4.6).

$$w_{ij}(t) = (1-h)w_{ij}(t-1) - \eta \delta_i y_i. \qquad (4.6)$$

Здесь h - небольшая константа. В [10] утверждается, что даже значение $h = 10^{-5}$, как правило, позволяет существенно улучшить обобщающую способность сети. В [15] используются значения h, равные $1 \cdot 10^{-3}$ и $5 \cdot 10^{-3}$. В [11] метод разрушения весов используется для устранения переполнения разрядной сетки, иногда возникающего при обучении алгоритмом Quickprop.

4.5. Эластичная модификация (RPROP) [14]

Эластичный алгоритм ОРО (Resilient backpropagation = RPROP) был предложен Ридмиллером и Брауном в 1993 году. В отличие от других алгоритмов, основанных на градиентном спуске, RPROP не использует величину градиента, а только его знак. Каждому весу сопоставляется так называемая величина обновления (update value), которая инициализируется небольшим числом. После каждой итерации алгоритма значения этих параметров увеличиваются на коэффициент увеличения, если знак текущего градиента совпадает со знаком предыдущего, или уменьшаются на коэффициент уменьшения, если эти знаки не совпадают. Коэффициент увеличения принято брать равным 1,2, а коэффициент уменьшения - 0,5. Если знак (направление) градиента отрицателен, то величина обновления прибавляется к весу, а если знак положителен, то величина обновления вычитается из веса.

Алгоритм RPROP часто дает более быструю сходимость, чем большинство других алгоритмов, и поэтому он является одной из наиболее популярных модификаций ОРО из известных в настоящее время. Однако в [11] отмечается, что для некоторых особых случаев RPROP не может найти оптимальную настройку сети, попадая в ложные локальные минимумы на поверхности ошибки.

4.6. Методы второго порядка [2, 10]

Базовый алгоритм ОРО двигается к минимуму функции ошибки в направлении наискорейшего спуска. Такой принцип не всегда означает максимально быструю сходимость поиска. Методы второго порядка пытаются найти более короткий путь к минимуму, более тщательно исследуя кривизну поверхности ошибки. Все методы второго порядка основаны на методе Ньютона, использующем первые три члена разложения функции ошибки в ряд Тейлора. Правило для изменения весовых коэффициентов в методе Ньютона следующее:

$$w(t+1) = w(t) - A^{-1}(t)\eta\frac{\partial E}{\partial w}. \qquad (4.7)$$

Здесь $A(t)$ - гессиан (матрица вторых производных) функции ошибки по текущим значениям весов и смещений. В теории метод Ньютона должен ускорять сходимость поиска в несколько раз, но вычисление гессиана настолько трудоемкая операция, что практические реализации метода Ньютона находят минимум ошибки намного медленнее стандартного метода ОРО. В силу своей трудоемкости, метод Ньютона на практике не применяется, но существует множество эффективных алгоритмов, использующих основную идею метода Ньютона.

Одним из них является алгоритм Левенберга-Маркардта (Levenberg-Marquardt), который использует нетрудоемкую аппроксимацию гессиана для метода Ньютона. В этом алгоритме, гессиан оценивается по формуле $A^{-1} \approx J^T \cdot J$, а градиент рассчитывается как $\mathbf{g} = J^T \cdot \mathbf{e}$, где J - якобиан (матрица первых производных) функции ошибки, $\mathbf{e}$ - вектор ошибки сети. Так как активационные функции в нейронных сетях прямого распространения имеют простые производные, якобиан рассчитывается намного быстрее матрицы Гесса, тем более он и так вычисляется в стандартном методе ОРО.

На методе Ньютона основаны также так называемые алгоритмы сопряженных градиентов (conjugate gradient algorithms). В алгоритмах сопряженных градиентов направления изменения весов определяются по следующей рекуррентной формуле:

$$V(t) = -G(t) + z(t, t-1)V(t-1), \qquad (4.8)$$

где $G = \frac{\partial E}{\partial w}$ - градиент функции ошибки, параметр $z(t, t-1)$ определяется, в зависимости от выбранной конкретной модификации алгоритма, по формуле (4.9) (алгоритм Флетчера-Ривса) или (4.10) (алгоритм Полака-Рибьера).

$$z(t, t-1) = \frac{|G(t)|^2}{|G(t-1)|^2}, \qquad (4.9)$$

$$z(t, t-1) = \frac{|G(t)|^2 - |G(t)\cdot G(t-1)|}{|G(t-1)|^2}. \qquad (4.10)$$

После выбора направления поиска, находится величина изменения веса по формуле $\Delta w = \lambda V(t)$, где коэффициент λ вычисляется, исходя из условия минимума функции $E(w(t-1) + \lambda V)$.

4.7. Алгоритм Quickprop [10, 11]

Алгоритм Quickprop, предложенный Фальманом [11] в 1988 году, является одной из наиболее популярных модификаций обратного распространения ошибки. Quickprop сочетает в себе идеи алгоритмов второго порядка с простотой реализации и низкой трудоемкостью. Правило изменения весов в методе Quickprop сводится к следующему уравнению.

$$\Delta w(t) = -\eta\frac{\partial E}{\partial w} + \frac{\frac{\partial E}{\partial w}(t)}{\frac{\partial E}{\partial w}(t-1) - \frac{\partial E}{\partial w}(t)}\Delta w(t-1). \qquad (4.11)$$

Фальман также вводит в алгоритм коэффициент максимального роста (μ), который используется как верхний предел изменения весов для предотвращения колебаний сети в окрестности минимума ошибки. Без использования этого коэффициента весовые коэффициенты могут постоянно расти, не увеличивая общую производительность сети. Для большинства практических задач значение коэффициента максимального роста рекомендуется брать из интервала от 1,75 до 2,25. Полностью алгоритм Quickprop представлен в [10] в виде псевдокода.

4.8. Модификации ОРО с переменной скоростью обучения [2, 10]

В классическом методе ОРО скорость обучения (η) инициализируется и в течение всего процесса обучения остается постоянной. Существует несколько модификаций ОРО, которые подразумевают изменение параметра η во время обучения.

В 1981 году Саттоном [12] был предложен метод адаптивного изменения скорости обучения, называемый методом Delta-Delta. Согласно этому методу каждому весу сопоставляется своя скорость обучения, которая изменяется по следующей формуле

$$\Delta\eta_i(t) = \gamma\frac{\partial E(t)}{\partial w_i(t)}\frac{\partial E(t-1)}{\partial w_i(t-1)}. \qquad (4.12)$$

Здесь γ - небольшой положительный коэффициент, являющийся аналогом параметра η в алгоритмах ОРО с постоянной скоростью обучения.

В 1988 Джейкобс [13] указал на недостатки алгоритма

Саттона и модифицировал его, предложив следующую формулу для изменения скорости обучения:

$$\Delta\eta_i(t) = \begin{cases} k, \text{ если } \bar{\delta}(t-1)\delta(t) > 0 \\ -\Phi\eta(t), \text{ если } \bar{\delta}(t-1)\delta(t) < 0 \\ 0, \text{ если } \bar{\delta}(t-1)\delta(t) = 0 \end{cases}, \qquad (4.13)$$

где $\delta(t) = \frac{\partial E(t)}{\partial w(t)}$, $\bar{\delta}(t) = (1-\theta)\delta(t) + \theta\bar{\delta}(t-1)$. Здесь $\delta(t)$ - частная производная функции ошибки по весу, $\bar{\delta}(t)$ - экспоненциальное среднее текущей и ранее вычисленных производных. Джейкобс использовал для коэффициента линейного роста k значения от 0,35 (для сложных задач) до 5,0 (для простых задач); коэффициент экспоненциального падения Φ от 0,1 до 0,35 и θ в районе 0,7. Алгоритм Джейкобса получил название Delta-Bar-Delta. Его достоинством является то, что в отдалении от минимума функции ошибки скорость обучения растет линейно, ускоряя сходимость, а при попадании в окрестность минимума, она падает экспоненциально, повышая точность поиска.

5. РАДИАЛЬНО-БАЗИСНЫЕ СЕТИ

Радиально-базисные сети относятся к сетям прямого распространения, но они не обучаются методом ОРО. Свое название эти сети получили из-за использования нейронов с радиально-базисными функциями. На входе нейрона вычисляется расстояние между векторами **w** и **p**, затем полученное значение умножается на смещение нейрона и от результата берется радиально-базисная функция (гауссиана). Радиально-базисная функция выражается формулой $H(x) = \exp(-x^2)$. Таким образом, нейроном реализовывается функция $a = H(\|w-p\|b)$. Поскольку гауссиана достигает максимума при значении аргумента, равном нулю, рассмотренный нейрон выдает значение, показывающее насколько близко расположен входной вектор к весовому вектору.

Основная разновидность радиально-базисных сетей представляет собой двухслойную сеть прямого распространения, первый слой которой состоит из радиально-базисных нейронов, а второй слой представляет собой линейную сеть (гл. 4). В отличие от сетей, обучаемых методом ОРО, радиально-базисные сети не обучаются, а разрабатываются под конкретную задачу. Например, известен алгоритм разработки радиально-базисной сети, дающей нулевую ошибку на любой обучающей выборке (функция newrbe в [2]). Но такой алгоритм может построить сеть с избыточным числом нейронов. Поэтому, как правило, при разработке радиально-базисных сетей берется максимально допустимое значение ошибки, большее нуля, и находится минимальная конфигурация системы, которая на обучающей выборке не превышает эту ошибку. Кроме основной разновидности радиально-базисных сетей существуют и другие, например сети обобщенной регрессии и вероятностные сети [2, 4].

У радиально-базисных сетей имеется масса достоинств, но и есть существенный недостаток. Радиально-базисные нейроны используют больше вычислений для своего функционирования, поэтому радиально-базисные сети работают медленнее, чем другие сети прямого распространения.

6. САМООРГАНИЗУЮЩИЕСЯ СЕТИ

Самоорганизующиеся сети относятся к сетям с обратными связями. В отличие от всех рассмотренных ранее классов нейронных сетей, у самоорганизующихся сетей нет ярко выраженного разделения на фазу обучения и фазу функционирования. Самоорганизующиеся сети получают вектора входного воздействия, выдают на выходе свою реакцию и, возможно, корректируют свои настройки в зависимости от поступившего входного вектора. Самоорганизующиеся сети включают в себя соревновательные сети и самоорганизующиеся карты признаков (Self-Organizing Feature Map = SOFM) Кохонена.

6.1. Соревновательные сети

Соревновательные сети - это разновидность однослойных самоорганизующихся сетей. На входе сеть вычисляет вектор, содержащий отрицательные расстояния между вектором **p** и элементами матрицы **W**, затем прибавляет к нему вектор смещений **b**. На выход сети, соответствующий максимальному полученному значению подается 1, а на остальные выходы - 0. Если все смещения равны 0, то слой выбирает тот весовой вектор, который находится ближе всех к входному вектору. Это происходит потому, что расстояния между входным вектором и весовыми векторами берутся с противоположными знаками, и поэтому наименьшее расстояние между этими векторами дает наибольшее значение на выходе.

Кохонен [17] сформулировал следующее правило обучения соревновательных нейросетей. Пусть поданный на вход вектор активизировал i-й выход соревновательной сети. Тогда i-ая строка весовой матрицы меняется по формуле:

$$_iW_{11}(q) = {}_iW_{11}(q-1) + \eta(p(q) - {}_iW_{11}(q-1)). \qquad (6.1)$$

В результате увеличивается вероятность того, что рассматриваемый нейрон при предъявлении сети похожего вектора также активизируется. В то же время, снижается вероятность активизации этого же нейрона, если сети будет предъявлен непохожий вектор. Часто правило Кохонена дополняется специальной процедурой, заключающейся в периодическом увеличении смещений нейронов, которые редко выигрывают (или никогда не выигрывают) соревнование с другими нейронами. Необходимость в такой процедуре возникает из-за того, что при инициализации системы некоторые нейроны оказываются заведомо далеко от центров кластеров и, как бы долго ни шел процесс обучения, такие нейроны не активизируются и, следовательно, не участвуют в процессе кластеризации.

Если соревновательная сеть содержит достаточное количество нейронов, то в конечном итоге эта сеть должна выделить кластеры в пространстве входных сигналов, к которым она будет относить поступающие входные вектора.

6.2. Самоорганизующиеся карты признаков (SOFM)

Самоорганизующиеся карты признаков (Self-Organizing Feature Maps = SOFM) Кохонена, как и соревновательные сети обучаются кластеризации векторов по принципу близости их расположения во входном пространстве. Сети SOFM отличаются тем, что в них близкие по расположению нейроны учатся распознавать близкие области. Таким образом, карты признаков обучаются не только распределению входных векторов, но и их топологии. Самоорганизующиеся карты можно представить в виде взвешенного графа, в вершинах которого расположены нейроны, а взвешенные ребра характеризуют расстояния между нейронами. Правило Кохонена в самоорганизующихся картах применяется не только для победившего нейрона, но и для соседних с ним. В некоторых случаях, для соседних нейронов используется меньший коэффициент скорости обучения, чем для победившего нейрона (например, в два раза). Архитектура карт Кохонена похожа на архитектуру соревновательных сетей, за исключением того, что в картах не используются нейронные смещения.

6.3. Соревновательные сети с квантованием обучающего вектора (LVQ)

Квантование обучающего вектора (Learning Vector Quantization = LVQ) - это метод обучения соревновательных сетей "с учителем". Стандартные соревновательные сети обучаются классификации входных векторов "без учителя" в процессе работы, используя в качестве единственного признака функцию расстояния. Если два входных вектора очень похожи, соревновательный слой, вероятно, поместит их в один класс. При этом не существует механизма, позволяющего разработать соревновательный слой, относительно которого можно было бы с уверенностью сказать: попадут ли два произвольно заданных входных вектора в один и тот же класс или нет. При использовании метода квантования обучающего вектора, к соревновательному слою достраивается линейный слой. В этом случае, первый (соревновательный) слой обучается классифицировать входные образы по расстоянию между векторами, а второй (линейный) приводит получающиеся подклассы к классам, определяемым пользователем.

7. РЕКУРРЕНТНЫЕ СЕТИ

Рекуррентные сети являются подклассом нейронных сетей с обратными связями. В отличие от самоорганизующихся сетей, в рекуррентных сетях обратные связи располагаются не между нейронами одного и того же слоя, а идут от выхода слоев к входам. Рассмотрим две разновидности рекуррентных сетей: сети Эльмана и сети Хопфилда.

7.1. Сети Эльмана

Сеть Эльмана представляет собой двухслойную сеть с первым тангенциальным и вторым линейным слоем. Обратная связь проходит от выхода первого слоя на его вход через элемент задержки и матрицу весов. Обратная связь позволяет сети Эльмана распознавать не только пространственные, но и временные образы. Не смотря на то, что сети Эльмана относятся к рекуррентным сетям, их можно обучать слегка модифицированным методом обратного распространения ошибки. Если с помощью сети Эльмана требуется распознать временные образы, то обучающие входные вектора необходимо предъявлять последовательно, как они будут появляться при работе сети. При обучении сети Эльмана, весовые коэффициенты можно корректировать как после предъявления каждого входного вектора, так и после предъявления последовательности входных векторов (в пакетном режиме). В [2] рассмотрен пример практического применения сети Эльмана для адаптивного определения амплитуды произвольного сигнала.

7.2. Сети Хопфилда

Сети Хопфилда представляют собой однослойные рекуррентные сети с линейно-пороговой функцией активации и элементом задержки на обратной связи. Под линейно-пороговой активационной функцией понимается функция вида:

$$f(n) = \begin{cases} n, & \text{если } |n| < 1 \\ \text{sign}(n), & \text{если } |n| \geq 1 \end{cases}. \qquad (7.1)$$

Принцип функционирования сети состоит в хранении некоторого множества точек устойчивого равновесия (аттракторов) в пространстве выходного сигнала. При предъявлении входного вектора **p**, сеть Хопфилда, через некоторое число итераций переходит в устойчивое состояние, соответствующее ближайшему аттрактору для вектора **p**. Таким образом, основное применение сети Хопфилда - реализация ассоциативной памяти, способной восстановить весь вектор по вектору с утраченными или зашумленными фрагментами. Сети Хопфилда не обучаются, а проектируются для заданного множества аттракторов. В [2] реализован алгоритм проектирования сети Хопфилда с минимизацией количества ложных аттракторов. К сожалению, не известен алгоритм разработки сетей Хопфилда, гарантирующий полное отсутствие ложных и неустойчивых точек равновесия для любой задачи.

ЗАКЛЮЧЕНИЕ

В результате работы был проведен обзор основных типов современных архитектур нейронных сетей и нейросетевых алгоритмов. Работа проводилась в рамках хоздоговора 11711 Таганрогского радиотехнического университета для разработки нейросетевой программы биометрической идентификации пользователя по клавиатурному почерку.

ПЕРЕЧЕНЬ ССЫЛОК

1. *Elliot D. L.*, "A Better Activation Function for Artificial Neural Networks", Institute for System Research: Maryland, 1993 (ftp://ftp.isr.umd.edu/pub/TechReports/1993/TR_93-8.pdf).
2. *Demuth H.,* and *M. Beal*e "Neural network toolbox for use with Matlab. User's guide", MathWorks, Inc., 1997 (http://www.mathworks.com).
3. Учебно-методическое пособие для самостоятельной работы по курсам "Системы искусственного интеллекта", "Методы распознавания образов"/ Сост. *Вишняков Ю. М., Кодачигов В. И., Родзин С. И.- Таганрог: ТРТУ, 1999.-132 с.*
4. *Hagan M. T., H. B. Demuth, M. H. Beale* Neural Networks Design. Boston, MA: PWS Publishing, 1996.
5. *Widrow B.*, and *M. E. Hoff*, Adaptive switching circuits, 1960 IRE WESCON Convention Record, New York IRE, 1960, pp. 96-104.
6. *Fahlman S. E.* and *C. Lebiere* The Cascade-Correlation Learning Architecture, Pittsburgh, PA: Carnegie Mellon University, 1991 (ftp://ftp.cs.cmu.edu/afs/cs/project/connect/tr/cascor-tr.ps.Z).
7. *Rumelhart D. E., Hinton, G. E., and Williams, R. J.* (1986) Learning Internal Representations by Error Propagation// Rumelhart D. E. and McClelland, J. L., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, 1986.
8. *Вороновский Г. К.*, и др. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. - Харьков: ОСНОВА, 1997. - 112 с.
9. *Wasserman P.* Neurocomputing. Theory and practice, Nostram Reinhold, 1990. (Рус. пер. Ф. Уоссермен. Нейрокомпьютерная техника. М. Мир, 1992).
10. *Gibb, J.* Back Propagation Family Album, NSW, Australia: Macquarie University, 1996, 72 p. (ftp://ftp.mpce.mq.edu.au/pub/comp/techreports/96C005.gibb.ps).
11. *Fahlman S. E.* An Empirical Study of Learning Speed in Back-Propagation Networks, Pittsburgh, PA: Carnegie Mellon University, 1988 (ftp://ftp.cs.cmu.edu/afs/cs/project/connect/tr/qp-tr.ps.Z).
12. *Barto A. G.* and *R. S. Sutton* Goal seeking components for adaptive intelligence. Technical Report AFWAL-TR-81-1070, Wright-Patterson Air Force Base, Ohio, 1981.
13. *Jacobs R. A.* Increased rates of convergence through learning rate adaptation// Neural Networks , N1, 1988, pp. 295-307.
14. *Riedmiller M.*, and *H. Braun*, A direct adaptive method for faster backpropagation learning: The RPROP algorithm// Proceedings of the IEEE International Conference on Neural Networks, 1993 (ftp://i11s16.ira.uka.de/pub/neuro/papers/riedml.icnn93.ps.Z).
15. *Tveter D. R.*, Backpropagator's Review, 2000 (http://www.dontveter.com/bpr/bpr.html).
16. *Krogh A.*, and *J. A. Hertz*. A simple weight decay can improve generalization. Technical report. Niels Bohr Institute / Nordita, Blegdamsvej 17, DK-2100 Copenhagen, Denmark, 1990.
17. *Kohonen T.*, Self-Organization and Associative Memory, 2nd Edition, Berlin: Springer-Verlag, 1987.

УДК 519.873 + 519.7.24/25

ЛОГИКО-АЛГЕБРАИЧЕСКИЕ ОСНОВЫ МАТЕМАТИЧЕСКОЙ ТЕОРИИ НАДЕЖНОСТИ

Л.И.Волгин

На базе логико-алгебраического аппарата аддитивно-мультипликативной (АМ) алгебры разработаны логические основы теории надежности изделий. На основе изоморфизма логической АМ-модели $P = P_{\text{АМ}}(p_1, p_2, \ldots, p_n)$ *и топологической модели (логической схемы надежности - ЛСН)* $\text{ЛСН} = P_{\text{АМ}}(E_1, E_2, \ldots, E_n)$ *построена единая совмещенная структурно-параметрическая модель надежности, где* $p_i \in [0, 1]$ *- вероятность безотказной работы элементов* E_i *ЛСН со временем безотказной работы* t_i.

The logical bases of production reliability were developed on the basis of the additive-multiplicative algebra.

ВВЕДЕНИЕ

Любой технический объект может быть представлен триадой (S,P,Q), где $S = S(E_1, E_2, \ldots, E_n)$ - структура объекта, содержащая множество $\{E_1, E_2, \ldots, E_n\}$ взаимодействующих элементов E_i, каждому из которых ставится в соответствие параметр $p_i = p(E_i)$, Q - критерий оптимизации [1]. Задача синтеза заключается в построении структуры объекта, задача анализа (при заданной структуре) заключается в определении результирующего параметра $P = p[S(E_1, E_2, \ldots, E_n)]$ структуры объекта. Оптимизация сводится к обеспечению экстремального значения функционала $Q = Q(S, P, D) \to \text{ext}$ (максимум или минимум), где D - условия оптимизации.

Если ввести символы топологических операций параллельного "$\oplus$" и последовательного "$\cdot$" соединения элементов топологической модели (схемы) объекта, то для классов параллельно-последовательных схем (ПП-схемы) топологическая модель может быть представлена в аналитической форме записи $S = S(E_1, E_2, \ldots, E_n)$. В частности, на рис.1 представлены логические (структурные) схемы надежности (ЛСН) при последовательном и параллельном (горячее резервирование) соединении элементов в ЛСН, аналитическое представление которых имеет вид:

$$S = E_1 \cdot E_2 \cdot \ldots \cdot E_n, \; S = E_1 \oplus E_2 \oplus \ldots \oplus E_n. \qquad (1)$$

Математическая модель объекта записывается в базисе операций арифметической алгебры (А-алгебра).

В настоящее время для описания объекта проектирования повсеместно используется раздельное представление топологической и математической моделей. Топологическая модель дает наглядно-образное представление структуры объекта и служит в качестве исходных данных для построения математической модели, которая необходима для расчета результирующего (выходного) параметра объекта $P = P_A(p_1, p_2, \ldots, p_n)$.