Нейронные сети в мехатронике

Ю. Ф. ГОЛУБЕВ
Московский государственный университет им. М. В. Ломоносова
e-mail: golubev@keldysh.ru

УДК 621

Ключевые слова: нейронные сети, искусственный интеллект, мехатроника, прогнозирование, ассоциативная память, поведение, управление.

Аннотация

Даётся элементарное введение в теорию искусственных нейронных сетей. Представлены принципы их структурной организации. Сформулированы и обоснованы методы обучения нейронных сетей, применяемые для различных уровней интеллектуального управления мехатронными системами. Изложены нейросетевые подходы к решению типовых задач классификации, цифровой обработки сигналов, сжатия информации, интерполяции и экстраполяции функций, ассоциативного поведения, оптимизации.

Abstract

Yu. F. Golubev, Neuron networks in mechatronics, Fundamentalnaya i prikladnaya matematika, vol. 11 (2005), no. 8, pp. 81—103.

An elementary introduction to the theory of artificial neuron networks is given. Principles of their structural composition are presented. Methods for the neural networks training commonly used for different levels of intellectual control of mechatronic systems are formulated and substantiated. Neuron networks approaches to typical problems of classification, digital signal processing, data compression, function interpolation and extrapolation, associative behavior, and optimization are stated.

1. Введение

Понятие «искусственные нейронные сети», оформилось в 1940-х годах в основополагающих работах МакКаллога и Питтса [3], которые показали, что сети, состоящие из искусственных нейронов, способны в принципе вычислить любую арифметическую или логическую функцию.

В 1949 г. Дональд Хебб [13] предположил, что классический условный рефлекс, открытый И. П. Павловым, возникает вследствие способности отдельных нейронов к установлению ассоциаций, и сформулировал соответствующее правдоподобное правило обучения биологических нейронов.

Первое практическое приложение искусственных нейронных сетей относится к концу 1950-х годов и связано с изобретением Розенблаттом перцептрона и соответствующего правила его обучения [18]. Розенблатт и его коллеги построили

Фундаментальная и прикладная математика, 2005, том 11, № 8, с. 81—103.

перцептрон и продемонстрировали его способность к обучению и решению задач классификации. Этот первый успех вызвал волну интереса к исследованию нейронных сетей.

Приблизительно в то же время Видров и Хофф [19] предложили другой обучающий алгоритм для настройки адаптивных линейных нейронных сетей. Обучающий алгоритм Видрова—Хоффа применяется и в наше время.

Однослойные сети Розенблатта и Видрова имеют сходные ограничения, сужающие их область применения, что и было выявлено в книге Минского и Пайперта [4]. После выхода в свет этой книги Розенблатт и Видров разработали многослойные сети, свободные от выявленных недостатков [8, 20]. Однако Розенблатту и Видрову не удалось модернизировать свои обучающие алгоритмы так, чтобы эти более сложные сети можно было автоматически настраивать.

Из-за книги Минского и Пайперта многие поверили, что дальнейшие исследования в области нейронных сетей не имеют никакой перспективы. Пессимизм усугублялся и тем, что тогда ещё не было достаточно мощных компьютеров для экспериментирования с нейронными сетями. Приблизительно на десять лет исследования в области нейронных сетей практически заглохли.

Тем не менее в 1970-х годах разработки нейронных сетей всё же продолжались. В 1972 г. Кохонен и Андерсон [10, 15, 16] независимо предложили новый тип нейронных сетей, способных функционировать в качестве памяти, обеспечивающей самоорганизующиеся отображения состояний с сохранением топологии сенсорного пространства (карта признаков). В это же время над созданием самоорганизующихся сетей активно работал Гроссберг [11].

К 1980-м годам мощные персональные компьютеры и рабочие станции стали широко доступными. Возникли также новые концепции по структурам нейронных сетей. Всё это привело к заметному всплеску исследований.

Возрождение интереса к нейронным сетям обязано, в основном, двум идеям. Первая из них состояла в применении методов механики для объяснения работы некоторого специального класса рекуррентных сетей. Соответствующие результаты были опубликованы Хопфилдом в 1982 г. [14].

Вторая идея 1980-х годов — это алгоритм обратного распространения для настройки многослойных сетей, который был открыт независимо несколькими различными исследователями. Наиболее популярной по алгоритму обратного распространения оказалась вышедшая в 1986 г. статья Рюмельхарта и МакКлеланда [17]. Этот алгоритм позволил преодолеть критические замечания Минского и Пайперта.

С тех пор нейронные сети нашли много приложений. Достаточно подробный обзор соответствующих работ можно найти в [6, 7, 9, 12], а также в материалах ежегодных Всероссийских конференций «Нейрокомпьютеры и их применение».

По существу, многослойная нейронная сеть представляет собой вычислительную среду параллельного действия с адаптацией на параметрическом, алгоритмическом и структурном уровне управляемых процессов. Такая среда имеет много потенциальных возможностей. Существующие в настоящее время микропроцессорные средства, вообще говоря, могут реализовать функции нейронных

сетей при создании для них соответствующего программного обеспечения. Однако более перспективным представляется применение нейрочипов, архитектура которых специально ориентирована на выполнение нейросетевых операций. Число приложений нейронных сетей, денежные инвестиции в их разработку, глубина и широта интереса к таким устройствам растут очень быстро. Информацию о технической реализации нейронных сетей можно найти, например, в [1].

Многие задачи, для решения которых используются нейронные сети, могут рассматриваться как частные случаи следующих основных проблем: классификация объектов, аппроксимация функции по конечному набору её значений, оптимизация, построение отношений на множестве объектов, смысловой поиск информации и ассоциативная память, фильтрация, сжатие информации, управление динамическими системами, нейросетевая реализация алгоритмов вычислительной математики.

2. Основные определения

Не вдаваясь в подробности технической реализации искусственных нейронов, приведём формальное определение.

Искусственный нейрон — устройство, обеспечивающее вычисление функции $a = f(\mathbf{W}\mathbf{p} + b)$, где a — скалярный выход нейрона, $\mathbf{W}$ — весовая $(1 \times m)$-матрица-строка, $\mathbf{p} \in \mathbb{R}^m$ — вектор-столбец входных сигналов, b — скаляр, называемый *смещением*, f — функция активации, скаляр $z = \mathbf{W}\mathbf{p} + b$ — *чистый вход* (дискриминантная функция) нейрона. В аппаратном исполнении функцию чистого входа вычисляет *адаптивный сумматор*.

Указанные элементы образуют основной стандарт нейроинформатики. Вместе с тем существует много дополнений и вариаций. Например, в качестве чистого входа может применяться квадратичная функция

$$z = \sum_{ij} v_{ij} p_i p_j + \sum_i w_i p_i + b.$$

Часто использующиеся функции активации приведены в таблице 1. В её последней колонке даны краткие обозначения функций активации. Нейрон с функцией активации в виде ступеньки или симметричной ступеньки называется *перцептроном*. Кроме перечисленных, могут применяться и другие функции, например $f(z) = \operatorname{arctg} z$ или $f(z) = z/(\alpha + |z|)$, $\alpha > 0$.

Число входов в нейрон зависит от существа решаемой проблемы. Например, если требуется определить, лётная или нелётная стоит погода, то входами в нейрон целесообразно выбрать температуру воздуха, скорость ветра и влажность. Поэтому нейрон в данном случае будет иметь три входа.

Линейка сигнала — устройство, преобразующее входной сигнал, заданный в виде функции $f(\tau)$, в m-мерный вектор, состоящий из значений входного сигнала в текущий момент времени и в моменты времени, отстоящие от текущего

Таблица 1. Стандартные функции активации

Смысл	Формула	Иконка	Имя
Ступенька	$\begin{cases} a = 0, & z < 0, \\ a = 1, & z \geqslant 0 \end{cases}$		hardlim
Симметричная ступенька	$\begin{cases} a = -1, & z < 0, \\ a = 1, & z \geqslant 0 \end{cases}$		hardlims
Линейная	$a = z$		purelin
Линейная с насыщением	$\begin{cases} a = 0, & z < 0, \\ a = z, & 0 \leqslant z \leqslant 1, \\ a = 1, & z \geqslant 1 \end{cases}$		satlin
Симметричная линейная с насыщением	$\begin{cases} a = -1, & z < -1, \\ a = z, & -1 \leqslant z \leqslant 1, \\ a = 1, & z \geqslant 1 \end{cases}$		satlins
Положительная линейная	$\begin{cases} a = 0, & z < 0, \\ a = z, & 0 \leqslant z \end{cases}$		poslin
Лог-сигмоид	$a = \dfrac{1}{1 + e^{-z}}$		logsig
Гиперболический тангенс	$a = \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$		tansig
Соревновательная	$\begin{cases} a = 1, & \text{нейрон с } \max z, \\ a = 0, & \text{другие нейроны} \end{cases}$		compet

последовательно на $1, 2, \ldots, m-1$ временны́х шагов, т. е.

$$\mathbf{p} = (p_1, p_2, \ldots, p_m)^{\mathrm{T}}, \quad p_1 = f(k), \quad p_2 = f(k-1), \ldots, \quad p_m = f(k-m+1).$$

Системы с одним нейроном имеют значительную область применения, поскольку вычислять линейные функции вектора данных требуется во многих

приложениях. Среди них кодирование, декодирование, фильтрация (отделение сигнала от шума), выделение контуров на изображениях, статистическая обработка. Линейные функции применяются также в простейших задачах классификации для двух классов. Если классифицируемых типов объектов больше двух, то одного нейрона будет недостаточно.

Слой нейронов — это множество нейронов, имеющих один и тот же входной вектор. Слой включает весовую матрицу, сумматоры, вектор смещений $\mathbf{b}$, блоки, реализующие функции активации, выходной вектор $\mathbf{a}$. Каждый элемент входного вектора $\mathbf{p} \in \mathbb{R}^m$ соединён с каждым нейроном через весовую матрицу $\mathbf{W}$ размерности $n \times m$, где n — число нейронов в слое.

Работу слоя нейронов можно выразить формулой $\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b})$. Здесь $\mathbf{f}$ — $(n \times 1)$-вектор-функция. Её элементами служат функции активации отдельных нейронов. Аргументом отдельной функции активации служит только соответствующий по номеру элемент вектора чистого входа. Слой из нескольких нейронов может, например, решать задачи многокомпонентной классификации.

Многослойной нейронной сетью прямого распространения называется сеть, включающая набор слоёв нейронов, для которых выход каждого предыдущего слоя служит входом для последующего. Первый слой называется *входным*, а последний — *выходным*. Остальные слои называются *скрытыми*.

Пусть сеть имеет L слоёв и l — номер слоя. Тогда работа последовательной сети даётся формулами

$$\mathbf{a}^{(l)} = \mathbf{f}^{(l)}(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1, \ldots, L, \quad \mathbf{a}^{(0)} = \mathbf{p}, \quad \mathbf{a} = \mathbf{a}^{(L)}. \qquad (1)$$

Здесь вектор $\mathbf{p}$ — задаёт вход в нейронную сеть, вектор $\mathbf{a}$ представляет собой выход или результат работы сети. По-прежнему аргументом отдельной компоненты вектор-функции активации служит соответствующий ей по номеру элемент вектора чистого входа для слоя, который содержит данный нейрон.

Можно показать, что многослойная нейронная сеть с линейными функциями активации эквивалентна однослойной линейной нейронной сети. Как будет видно из дальнейшего, многослойные сети с нелинейными функциями активации оказываются более мощными, чем однослойные.

Блок зацикливания, реализованный в виде отдельного устройства, формирует входной сигнал $\mathbf{p}(\tau)$ в соответствии с формулой

$$\mathbf{p}(\tau + 1) = \mathbf{a}(\tau).$$

Здесь предполагается, что счётчик τ представлен дискретными шагами и может быть только целым. Этому блоку требуется при $\tau = 0$ задать начальные данные в виде $\mathbf{a}(0)$.

Рекуррентная нейронная сеть — это сеть с обратной связью. Некоторые из её выходов с помощью блоков зацикливания снова подаются на её входы.

Рекуррентная сеть обладает возможностями, которых не было у описанных выше последовательных сетей. Приведём несколько примеров.

Рекуррентная хэммингова сеть. Она предназначена для того, чтобы решать задачи классификации в бинарном пространстве, где элементы векторов могут

принимать значения только 1 или -1. Эта сеть включает два слоя. Первый слой работает в соответствии с формулой

$$\mathbf{a}^{(1)} = \mathbf{purelin}(\mathbf{W}^{(1)}\mathbf{p} + \mathbf{b}^{(1)}).$$

Второй слой рекуррентный:

$$\mathbf{a}^{(2)}(0) = \mathbf{a}^{(1)}, \quad \mathbf{a}^{(2)}(\tau+1) = \mathbf{poslin}(\mathbf{W}^{(2)}\mathbf{a}^{(2)}(\tau)),$$

$$\mathbf{W}^{(2)} = \begin{pmatrix} 1 & -\varepsilon & \dots & -\varepsilon \\ -\varepsilon & 1 & \dots & -\varepsilon \\ \vdots & \vdots & \ddots & \vdots \\ -\varepsilon & -\varepsilon & \dots & 1 \end{pmatrix},$$

причём число нейронов первого слоя совпадает с числом нейронов второго слоя и $0 < \varepsilon < 1/(n^{(2)} - 1)$.

Цель работы хэмминговой сети состоит в том, чтобы установить, какой прототип (строка весовой матрицы $\mathbf{W}^{(1)}$) ближе всего к входному вектору. Ответ получается на выходе рекуррентного слоя. Каждому прототипу отвечает один нейрон выхода. Когда рекуррентный слой сходится, остаётся только один ненулевой выходной нейрон (по принципу «победитель соревнования получает всё»). Этот ненулевой нейрон указывает на прототип, ближайший к входному вектору.

Решение систем линейных уравнений. Дана система линейных уравнений $\mathbf{W}\mathbf{p} = \mathbf{b}$, $\mathbf{W}$ — квадратная матрица размерности $m \times m$, $\mathbf{b}$ — вектор размерности m. Требуется найти значение вектора $\mathbf{p}$, при котором все уравнения обращаются в тождества.

Поиск решения осуществляется рекуррентной сетью

$$\mathbf{p}(0) = 0, \quad \mathbf{a}(k) = \mathbf{f}(\mathbf{W}\mathbf{p}(k) - \mathbf{b}), \quad \mathbf{p}(k+1) = \mathbf{p}(k) - 2h\mathbf{W}^{\mathrm{T}}\mathbf{a}(k),$$

где $f(z) = \mathrm{poslin}(z - \delta) - \mathrm{poslin}(-z - \delta)$. Эта сеть ищет

$$\arg\min_{\mathbf{p}} F(\mathbf{p}), \quad F(\mathbf{p}) = \|\mathbf{a}\|^2.$$

Решение систем неравенств $\mathbf{W}\mathbf{p} \leqslant \mathbf{b}$, $\mathbf{p} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, вполне аналогично решению системы линейных уравнений. Различие состоит лишь в выборе функции активации. Здесь $\mathbf{a} = \mathrm{poslin}(\mathbf{W}\mathbf{p} - \mathbf{b})$. Сеть найдёт некоторое значение вектора $\mathbf{p}$, при котором все неравенства системы будут удовлетворены. Если система несовместна, полученное значение вектора $\mathbf{p}$ будет обеспечивать минимум функционала F.

Перейдём к способам обучения нейронных сетей. *Правило обучения* определяет закон изменения весов и смещений в нейронной сети в соответствии с целью обучения.

Из совокупности известных правил обучения нейронных сетей можно выделить три достаточно широких класса: супервизорное обучение, ассоциативное обучение и несупервизорное обучение.

Правило супервизорного обучения предполагает набор примеров (*обучающая выборка*) правильной работы сети:

$$\mathcal{T} = \{(\mathbf{p}_k, \mathbf{t}_k),\ k = 1, \ldots, N\}, \tag{2}$$

где $\mathbf{p}_k$ — вход в нейронную сеть и $\mathbf{t}_k$ — соответствующий ему правильный выход (*цель*). Когда входы предъявляются сети, её выходы сравниваются с целями. Правило обучения используется для уточнения весов и смещений нейронов сети, чтобы приблизить выход сети к соответствующим целям.

Ассоциативное обучение подразумевает наличие условного и безусловного стимулов. Оно похоже на супервизорное обучение за исключением того, что правильный вектор выхода формируется сетью в ответ на безусловный стимул, чтобы при предъявлении сети условного стимула возникал такой же ответ сети.

При *несупервизорном обучении* производится модификация весов и смещений на основе анализа лишь входов в сеть. Целевые выходы не предъявляются. Многие из несупервизорных алгоритмов настраивают сеть на выполнение кластерных операций, т. е. обучают сеть выделять в множестве входов конечное число классов. Это полезно в таких приложениях, как векторная квантизация.

3. Правила обучения однослойных сетей

Рассмотрим несколько правил супервизорного обучения. Начнём с процедуры обучения перцептрона, который решает задачу классификации, т. е. отнесения входного вектора к тому или иному множеству из числа заданных в пространстве входных векторов, и функционирует в соответствии с формулой

$$\mathbf{a} = \mathbf{hardlim}(\mathbf{W}\mathbf{p} + \mathbf{b}).$$

Для уточнения весовой матрицы и вектора смещений перцептрона используется заданная обучающая выборка (2), где компоненты векторов $\mathbf{t}_k$ могут принимать значения либо 0, либо 1 в соответствии со следующей рекуррентной процедурой:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \boldsymbol{\delta}(k)\mathbf{p}^{\mathrm{T}}(k), \quad \mathbf{b}(k+1) = \mathbf{b}(k) + \boldsymbol{\delta}(k), \tag{3}$$

где $\boldsymbol{\delta}(k) = \mathbf{t}(k) - \mathbf{a}(k)$. Здесь k — номер итерации, $[\mathbf{p}(k), \mathbf{t}(k)] \in \mathcal{T}$, $\mathbf{a}(k)$ — ответ перцептрона на вход $\mathbf{p}(k)$. Процедура (3) носит название «правило обучения перцептрона».

Справедлива следующая теорема.

Теорема. *Если весовая матрица и вектор смещений, которые правильно классифицируют все N заданных входных векторов, существуют, то при неограниченном предъявлении перцептрону всех элементов обучающей выборки процедура* (3) *сходится за конечное число шагов.*

Заметим, что если классы входных векторов плохо разделимы (множества прототипов обучающей выборки близки друг к другу), то для сходимости алгоритма потребуется много итераций. Перцептрон может применяться лишь для

классификации линейно разделимых множеств входных векторов. Существует много задач, которые не удовлетворяют условию линейной разделимости.

Линейный ассоциатор — это однослойная сеть, выходной вектор которой определяется по формуле

$$\mathbf{a} = \mathbf{purelin}(\mathbf{W}\mathbf{p}) = \mathbf{W}\mathbf{p}.$$

Линейный ассоциатор иначе называется *ассоциативной памятью*. Другими словами, если сеть на входе получает $\mathbf{p}_k$, то на выходе она должна выдавать в точности $\mathbf{a} = \mathbf{t}_k$ для всех индексов $k = 1, 2, \ldots, N$. Эта сеть была предложена независимо Андерсоном и Кохоненом в 1972 г.

Потребуем, чтобы весовая матрица минимизировала следующий критерий работы сети:

$$F(\mathbf{W}) = \sum_{k=1}^{N} \|\mathbf{t}_k - \mathbf{a}(\mathbf{p}_k)\|^2. \tag{4}$$

Функция, стоящая в правой части, дифференцируема и всегда положительна. Следовательно, она имеет минимум. Условие минимальности состоит в том, что частные производные от функции $F(\mathbf{W})$ по переменным компонентам весовой матрицы должны быть равны нулю:

$$\mathbf{W}(\mathbf{P}\mathbf{P}^{\mathrm{T}}) = \mathbf{T}\mathbf{P}^{\mathrm{T}}, \quad \mathbf{T} = (\mathbf{t}_1\mathbf{t}_2\ldots\mathbf{t}_N), \quad \mathbf{P} = (\mathbf{p}_1\mathbf{p}_2\ldots\mathbf{p}_N).$$

Решение выражается формулами

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{+}, \quad \mathbf{P}^{+} = \begin{cases} \mathbf{P}^{\mathrm{T}}(\mathbf{P}\mathbf{P}^{\mathrm{T}})^{-1} & \text{при } \det(\mathbf{P}\mathbf{P}^{\mathrm{T}}) \neq 0, \\ (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}} & \text{при } \det(\mathbf{P}^{\mathrm{T}}\mathbf{P}) \neq 0. \end{cases}$$

Матрица $\mathbf{P}^{+}$ называется *псевдоинверсной*, а соответствующее правило обучения — *псевдоинверсным*.

В качестве приложения линейных ассоциаторов отметим задачу перекодировки цифр в двоичный код. Прямоугольное поле цифры разбивается на квадратики, цифра в этом поле пишется заданным способом. При чтении по правилу телевизионной развёртки строится многомерный входной вектор. Выходной вектор — та же цифра в двоичном коде.

В 1960 г. Видров и его аспирант Хофф предложили сеть АДАЛИН (ADAptive LInear NEuron: ADALINE) и разработали правило обучения, названное ими алгоритмом LMS (List Mean Square). Выход сети АДАЛИН выражается формулой

$$\mathbf{a} = \mathrm{purelin}(\mathbf{W}\mathbf{p} + \mathbf{b}) = \mathbf{W}\mathbf{p} + \mathbf{b}.$$

Алгоритм LMS обучает сеть по заданному множеству образцов её правильной работы (2).

Вместо полного критерия качества (4) Видров и Хофф предложили взять

$$\hat{F} = \|\mathbf{t}(k) - \mathbf{a}(\mathbf{p}(k))\|^2 = \boldsymbol{\delta}^2(k), \tag{5}$$

где пара $[\mathbf{p}(k), \mathbf{t}(k)]$ выбирается из множества (2). Таким образом, роль критерия качества теперь выполняет среднеквадратичная ошибка для k-го образца. Метод

градиентного спуска (*обучающий алгоритм Видрова—Хоффа*) принимает вид

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\boldsymbol{\delta}(k)\mathbf{p}^{\mathrm{T}}(k), \quad \mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\boldsymbol{\delta}(k),$$

где $\alpha > 0$ — параметр, задающий величину шага итераций. Эту последовательность операций называют также *алгоритм LMS* или *дельта-правило*.

Предъявляемые на вход сети образцы не обязаны все быть различными. Они могут повторяться сколько угодно раз, и процесс обучения сети может быть продолжен неограниченно.

Справедлива следующая теорема.

Теорема. *При неограниченном предъявлении сети некореллированной последовательности случайных равновероятных реализаций заданных образцов* (2) *и достаточно малом* α *обучающий алгоритм Видрова—Хоффа приводит к тому, что математическое ожидание весов и смещений сходится к аргументу минимума среднеквадратичного критерия качества* (4).

Сеть АДАЛИН — одна из наиболее широко распространённых в практических приложениях. Основная сфера её приложений — адаптивная фильтрация, где эта сеть до сих пор интенсивно применяется.

Чтобы сеть АДАЛИН работала как экстраполятор, нужна линейка сигнала с m выходами.

Экстраполяция функций. Пусть требуется выполнить экстраполяцию функции $f(x)$ по её заданным значениям

$$f_1 = f(x_1), \ldots, \; f_m = f(x_m).$$

Рассмотрим нейросетевую реализацию некоторых стандартных методов экстраполяции. Если узловые точки лежат равномерно на оси абсцисс, $x_{i+1} - x_i = h$, $i = 1, \ldots, m-1$, а спрогнозировать значение функции требуется для $x = x_m + kh$, то многочлен Лагранжа даёт

$$a_k = f(x_m + kh) = \sum_{i=1}^{m} w_{ki} f_i,$$

где

$$w_{ki} = \frac{(-1)^{m-i}(k+m-1)!}{(k-1)!\,(i-1)!\,(m-i)!\,(k+m-i)}.$$

Если прогноз требуется осуществить для k точек, расположенных равномерно с шагом h, то сеть будет иметь один слой с выходным вектором $\mathbf{a} = (a_1, \ldots, a_k)$.

Если прогнозируемая функция периодическая, $f(t) = F(t+T)$, $T \neq 0$, то её лучше представить отрезком ряда Фурье. Тогда

$$f(t) \approx \sum_{i=1}^{m} w_i f(t_i),$$

где

$$w_1 = \frac{1}{2(m-1)}\left[1 + \frac{\sin \tilde{N}\tau_1}{\sin}\tau_1\right], \quad w_m = \frac{1}{2(m-1)}\left[1 + \frac{\sin \tilde{N}\tau_m}{\sin}\tau_m\right],$$

$$w_i = \frac{1}{m-1}\left[1 + \frac{\sin \tilde{N}\tau_i}{\sin}\tau_i\right], \quad i = 2, \ldots, m-1,$$
$$\tilde{N} = N - 0{,}5, \quad \tau_j = \frac{\pi(t - t_j)}{T}, \quad j = 1, \ldots, m.$$

Видим, что если разность $t - t_1$ фиксирована, то и для экстраполяции по методу Фурье веса w_j оказываются не зависящими от конкретной экстраполируемой функции. Вместе с тем они будут отличаться от весов, получаемых с помощью полиномов Лагранжа. Возникает вопрос о поиске наилучшего класса функций для экстраполяции. Ответ на этот вопрос можно искать путём подбора значений весовых коэффициентов w_j, $j = 1, \ldots, m$, минимизирующих ошибку экстраполяции на множестве различных векторов, получаемых последовательно на линейке сигнала.

Адаптивный фильтр есть комбинация линейки сигнала и сети АДАЛИН, так что выход фильтра есть

$$a_k = \text{purelin}(\mathbf{w}_1^{\mathrm{T}}\mathbf{p} + b) = \sum_{i=1}^{m} w_{1i} f(k - i + 1) + b,$$

а весовые коэффициенты настраиваются алгоритмом LMS при каждом обновлении линейки. Адаптивный фильтр можно использовать во многих сферах деятельности.

Правило Хебба несупервизорного обучения предполагает увеличение соответствующего веса, если входной сигнал и ответ на него совпадают по знаку. Если знаки разные, то вес должен уменьшаться. Конкретизация правила Хебба зависит от решаемой задачи. Приведём два варианта конкретизации правила Хебба, пригодные для установления требуемых ассоциаций в поведении роботов.

Правило инстар (instar) имеет вид

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \alpha\, \mathbf{a}(k)[\mathbf{p}^{\mathrm{T}}(k) - \mathbf{W}(k)], \tag{6}$$

где α — скалярный обучающий коэффициент, $\mathbf{p}(k)$ — очередной входной вектор-столбец, $\mathbf{a}(k)$ — соответствующий ему вектор-столбец выхода.

Когда некоторый выходной нейрон активен, соответствующая весовая вектор-строка передвигается к входному вектору вдоль линии, соединяющей исходную весовую вектор-строку и входной вектор. Расстояние, на которое передвигается весовой вектор, зависит от значения α.

Для установления ассоциаций входы в нейрон делятся на две категории: безусловный стимул — входной вектор, для которого весовая матрица фиксирована так, что при предъявлении безусловного стимула сеть всегда отвечает заданным образом, и условный стимул — входной вектор, для которого весовая матрица не определена. Правило инстар обучения весовой матрицы для условного стимула позволяет установить такую ассоциативную связь между безусловным и условным стимулом, что сеть отвечает одинаково как на условный, так и на безусловный стимул.

Правило аутстар (outstar) выражается формулой

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \alpha\,[\mathbf{a}(k) - \mathbf{W}(k)]\mathbf{p}^{\mathrm{T}}(k).$$

Свойства правила аутстар являются дополнительными к правилу обучения инстар. Обучение происходит, когда p_j (вместо a_i) не равно нулю. Когда происходит обучение, j-й вектор-столбец весовой матрицы приближается к выходному вектору.

В качестве иллюстрации применения правила аутстар рассмотрим следующий пример. В лифте, обслуживающем только трёх главных руководителей фирмы, установлена нейронная сеть. Лифт имеет кнопки с 2-го по 5-й этажи. Когда руководитель входит на первый этаж, лифт с помощью видеодатчика определяет, кто вошёл, и использует настраиваемую сеть, чтобы выбрать этаж, на котором данный человек обычно выходит. Если выбор лифта неверен, человек может нажать в любое время другую кнопку, изменив при этом настройку сети. Функция вход/выход сети имеет вид

$$\mathbf{a} = \mathbf{hardlims}(\mathbf{W}^0\mathbf{p}^0 + \mathbf{W}\mathbf{p} + \mathbf{b}).$$

Безусловный стимул $\mathbf{p}^0$ представляет собой код этажа, возникающий при нажатии кнопки:

$$\mathbf{p}_2^0 = (-1;-1)^{\mathrm{T}}, \quad \mathbf{p}_3^0 = (1;-1)^{\mathrm{T}}, \quad \mathbf{p}_4^0 = (-1;1)^{\mathrm{T}}, \quad \mathbf{p}_5^0 = (1;1)^{\mathrm{T}}.$$

Если не нажата никакая кнопка, то выдаётся нулевой код:

$$\mathbf{p}_0^0 = (0;0)^{\mathrm{T}}.$$

Безусловный стимул взвешивается с диагональной матрицей, а смещения полагаются равными $-0{,}5$, так что если кнопка нажата, то сеть отвечает её кодом:

$$\mathbf{W}^0 = 2\mathbf{E}, \quad \mathbf{b} = (-0{,}5;-0{,}5)^{\mathrm{T}}.$$

Условный стимул всегда имеется. Он состоит из трёх компонент и представляет трёх руководителей: президента, вице-президента и директора соответственно:

$$\mathbf{p}_1 = (1;0;0)^{\mathrm{T}}, \quad \mathbf{p}_2 = (0;1;0)^{\mathrm{T}}, \quad \mathbf{p}_3 = (0;0;1)^{\mathrm{T}}.$$

Сеть учится вызывать этаж, предпочитаемый обычно руководителем, посредством изменения второго множества весовых коэффициентов с использованием правила аутстар. Если бы президенту был всегда нужен 5-й этаж, вице-президенту — 4-й, а директору — 2-й, то матрица $\mathbf{W}$, решающая задачу, могла бы иметь вид

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

Применяя правило обучения, сеть может менять эту матрицу, например при изменении обычного места присутствия какого-нибудь руководителя.

4. Обратное распространение

Отметим некоторые полезные свойства многослойных сетей, а затем представим алгоритм обратного распространения.

Многослойный перцептрон устроен так, что выход его первого слоя служит входом для второго слоя, выход второго слоя — входом третьего и так далее. Каждый слой может иметь различное число нейронов.

С помощью многослойного перцептрона можно решать проблему классификации с произвольными разрешающими границами. Надо только позаботится о достаточном числе нейронов в скрытых слоях. Идея состоит в том, чтобы на первом слое создать необходимое число линейных границ, которые потом можно будет объединить, используя операции «и» на втором слое и «или» на третьем. Разрешающие границы на втором слое будут выпуклыми, а на третьем слое область решений может иметь произвольную форму.

Помимо проблем классификации и фильтрации помех нейронные сети могут применяться для аппроксимации функций. Рассмотрим несколько примеров.

Интерполяция функции одного переменного. Пусть требуется вычислить значение непрерывной функции $f(x)$, $x \in [a,b] \subset \mathbb{R}$. Функцию $f(x)$ будем аппроксимировать кусочно-линейными функциями, проходящими через точки разбиения $\{x_i, f(x_i)\}$, $i = 0, 1, \ldots, N$, $x_0 = a$, $x_N = b$, $x_i < x_{i+1}$. Функция

$$\varphi_i(x) = \operatorname{sutlin}\left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right) + \operatorname{sutlin}\left(\frac{x_{i+1} - x}{x_{i+1} - x_i}\right) - 1$$

равна нулю при $x < x_{i-1}$ и $x > x_{i+1}$. При $x_{i-1} \leqslant x \leqslant x_i$ она линейно возрастает от нуля до единицы. При $x_i \leqslant x \leqslant x_{i+1}$ эта функция линейно убывает от единицы до нуля. Если взять линейную комбинацию

$$z(x) = \sum_{i=0}^{N} f(x_i)\varphi_i(x),$$

добавив какие-нибудь точки $x_{-1} < a$ и $x_{N+1} > b$, то тогда $z(x_i) = f(x_i)$, а в промежутках между узловыми точками функция $z(x)$ будет линейной и обеспечит требуемую линейную интерполяцию.

Нейронная сеть, реализующая указанный метод, имеет два слоя. Первый слой содержит $2N + 2$ нейрона, вычисляющих функции

$$a_{i,i-1} = \operatorname{sutlin}\left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right), \quad a_{i,i+1} = \operatorname{sutlin}\left(\frac{x_{i+1} - x}{x_{i+1} - x_i}\right).$$

Второй слой содержит 1 нейрон, выход которого даётся формулой

$$a = \operatorname{purelin}\left(\sum_{i=0}^{N}[f(x_i)a_{i,i-1} + f(x_i)a_{i,i+1}] - \sum_{i=0}^{N} f(x_i)\right),$$

и служит выходом сети.

Вместо функции purelin в интерполирующих сетях можно применять функцию logsig. Тогда интерполяция не будет линейной, но по своим свойствам будет напоминать линейную.

По аналогии с функцией $\varphi(x)$ применяется

$$\phi(z) = \operatorname{logsig}(z+c) + \operatorname{logsig}(-z+c) - 1 = \frac{1-e^{-2c}}{1+e^{-c}(e^{-z}+e^{z})+e^{-2c}}, \quad c>0.$$

Эта функция имеет максимум при $z=0$. Величину c можно подобрать так, чтобы значение $\phi(0)$ было сколь угодно близким к 1. Для первого слоя нейронов будем иметь

$$a_{2i}^{(1)} = \operatorname{logsig}(z_i+c), \quad a_{2i+1}^{(1)} = \operatorname{logsig}(-z_i+c), \quad z_i = \frac{x-x_i}{d}, \quad i=0,\ldots,N,$$

где величина d характеризует быстроту убывания логсигмоидных функций. Второй слой выполняет свёртку

$$f(x) \approx \sum_{i=0}^{N} \bigl(w_i^2 a_{2i}^{(1)} + w_i^2 a_{2i+1}^{(1)} - w_i^2\bigr).$$

Значения w_i^2 должны быть близки к значениям $f(x_i)$, но будут отличаться от них из-за влияния соседних узловых точек. Такая сеть будет нуждаться в некоторой дополнительной настройке весовых коэффициентов.

Триангуляция поверхностей. Пусть в аффинном пространстве $\mathbb{R}^3$ выбран прямоугольный репер $Oxyz$ и задана поверхность $z=f(x,y)$ на прямоугольнике $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$. Выберем узловые точки на пересечении прямых, проходящих через точки x_μ, $\mu=0,1,\ldots,N$, $x_0=a$, $x_N=b$, $x_\mu < x_{\mu+1}$, параллельно оси Oy, с прямыми, проходящими через точки y_ν, $\nu=0,1,\ldots,M$, $y_0=c$, $y_M=d$, $y_\nu < y_{\nu+1}$, параллельно оси Ox. Всего получим NM прямоугольников, образованных указанными прямыми. Обозначим $S_{\mu\nu}$ прямоугольник с вершинами в точках $(x_\mu, y_\nu, 0)$, $(x_\mu, y_{\nu+1}, 0)$, $(x_{\mu+1}, y_{\nu+1}, 0)$, $(x_{\mu+1}, y_\nu, 0)$. Соединив точки $(x_\mu, y_\nu, 0)$, $(x_{\mu+1}, y_{\nu+1}, 0)$ отрезком прямой, получим два треугольника: $T_{\mu,\nu+1}$ с вершинами $(x_\mu, y_\nu, 0)$, $(x_\mu, y_{\nu+1}, 0)$, $(x_{\mu+1}, y_{\nu+1}, 0)$ и $T_{\mu+1,\nu}$ с вершинами $(x_\mu, y_\nu, 0)$, $(x_{\mu+1}, y_\nu, 0)$, $(x_{\mu+1}, y_{\nu+1}, 0)$. Всего имеем $2NM$ таких треугольников. Для каждой узловой точки вычислим z-координату поверхности $z_{\mu\nu} = f(x_\mu, y_\nu)$, $\mu=0,\ldots,N$, $\nu=0,\ldots,M$.

По заданным координатам (x,y) точки на плоскости составим выходы первого слоя:

$$a_{2i}^{(1)} = \operatorname{purelin}(w_{2i,x}x + w_{2i,y}y + b_{2i}), \quad a_{2i+1}^{(1)} = \operatorname{purelin}(w_{2i+1,x}x + w_{2i+1,y}y + b_{2i+1}),$$

где

$$w_{2i,x} = \frac{z_{\mu+1,\nu+1} - z_{\mu,\nu+1}}{x_{\mu+1}-x_\mu}, \quad w_{2i,y} = \frac{z_{\mu,\nu+1} - z_{\mu\nu}}{y_{\nu+1}-y_\nu},$$

$$w_{2i+1,x} = \frac{z_{\mu+1,\nu} - z_{\mu,\nu}}{x_{\mu+1}-x_\mu}, \quad w_{2i+1,y} = \frac{z_{\mu+1,\nu+1} - z_{\mu+1\nu}}{y_{\nu+1}-y_\nu},$$

$$b_{2i} = z_{\mu\nu} - w_{2i,x}x_\mu - w_{2i,y}y_\nu, \quad b_{2i+1} = z_{\mu\nu} - w_{2i+1,x}x_\mu - w_{2i+1,y}y_\nu,$$
$$i = \mu + N\nu, \quad \mu = 0, \ldots, N-1, \quad \nu = 0, \ldots, M-1.$$

Если точка с координатами (x, y) попадает в треугольник $T_{\mu,\nu+1}$, то выход сети должен совпасть с $a^{(1)}_{2i}$. Если же точка (x, y) попадает в треугольник $T_{\mu+1,\nu}$, то выход сети должен совпасть с $a^{(1)}_{2i+1}$.

К $2NM$ нейронам первого слоя, вычисляющим значения z, добавим $6NM$ нейронов, определяющих положение точки (x, y) относительно сторон опорных треугольников:

$$a^{(1)}_{2NM+6i} = \operatorname{hardlim}(x - x_\mu), \quad a^{(1)}_{2NM+6i+1} = \operatorname{hardlim}(-y + y_{\nu+1}),$$
$$a^{(1)}_{2NM+6i+2} = \operatorname{hardlim}\left(-\frac{y_{\nu+1} - y_\nu}{x_{\mu+1} - x_\mu}x + y + \frac{y_{\nu+1} - y_\nu}{x_{\mu+1} - x_\mu}x_\mu - y_\nu\right),$$
$$a^{(1)}_{2NM+6i+3} = \operatorname{hardlim}(-x + x_{\mu+1}), \quad a^{(1)}_{2NM+6i+4} = \operatorname{hardlim}(y - y_\nu),$$
$$a^{(1)}_{2NM+6i+5} = \operatorname{hardlim}\left(\frac{y_{\nu+1} - y_\nu}{x_{\mu+1} - x_\mu}x - y - \frac{y_{\nu+1} - y_\nu}{x_{\mu+1} - x_\mu}x_\mu + y_\nu\right).$$

Выходы $a^{(1)}_{2NM+6i}$, $a^{(1)}_{2NM+6i+1}$, $a^{(1)}_{2NM+6i+2}$ одновременно равны единице, если точка (x, y) принадлежит треугольнику $T_{\mu,\nu+1}$, и хотя бы один из них равен нулю в противоположном случае. Выходы $a^{(1)}_{2NM+6i+3}$, $a^{(1)}_{2NM+6i+4}$, $a^{(1)}_{2NM+6i+5}$ одновременно равны единице, если (x, y) принадлежит треугольнику $T_{\mu+1,\nu}$, и хотя бы один из них равен нулю в противоположном случае. Второй слой содержит $4NM$ нейронов, из которых первые $2NM$ сохраняют выходы первых $2NM$ нейронов первого слоя:

$$a^{(2)}_{2i} = \operatorname{purelin}\big(a^{(1)}_{2i}\big), \quad a^{(2)}_{2i+1} = \operatorname{purelin}\big(a^{(1)}_{2i+1}\big), \quad i = 1, \ldots, NM,$$

тогда как остальные $2NM$ нейронов определяют принадлежность точки (x, y) тому или иному треугольнику:

$$a^{(2)}_{2NM+2i} = \operatorname{hardlim}\big(a^{(1)}_{2NM+6i} + a^{(1)}_{2NM+6i+1} + a^{(1)}_{2NM+6i+2} - 2, 1\big),$$
$$a^{(2)}_{2NM+2i+1} = \operatorname{hardlim}\big(a^{(1)}_{2NM+6i+3} + a^{(1)}_{2NM+6i+4} + a^{(1)}_{2NM+6i+5} - 2, 1\big),$$
$$i = 1, \ldots, NM.$$

Наконец, единственный нейрон третьего слоя осуществляет свёртку выходов второго слоя по формуле

$$z = a^{(3)} = \sum_{i=1}^{NM}\big[a^{(2)}_{2i}a^{(2)}_{2NM+2i} + a^{(2)}_{2i+1}a^{(2)}_{2NM+2i+1}\big].$$

Приведённая нейронная триангуляционная сеть содержит $8NM$ нейронов в первом слое, $2NM$ нейронов во втором (скрытом) слое и один нейрон в третьем (выходном) слое. Указанные числа нейронов обусловлены требованием точной триангуляции.

Для создания виртуальных моделей правдоподобных поверхностей аппроксимация $f(x,y)$ может быть выполнена более экономично следующим образом. Выходы первого слоя выразим формулами

$$a_i^{(1)} = \begin{cases} \left(1 - \frac{(x-x_\mu)^2+(y-y_\nu)^2}{R^2}\right)^q, & (x-x_\mu)^2+(y-y_\nu) \leqslant R, \\ 0, & (x-x_\mu)^2+(y-y_\nu) > R, \end{cases} \quad i = \nu + N\mu,$$

где R — максимальный радиус, в пределах которого ещё учитывается влияние узловых точек, а q — целочисленный показатель степени. Второй слой имеет один нейрон и осуществляет свёртку

$$a = \sum_{i=0}^{NM} w_i^2 a_i^{(1)},$$

где первоначально принимается $w_i^2 = f(x_\mu, y_\nu)$ для $i = \nu + N\mu$. В итоге сеть получается двухслойной, а число нейронов сокращается до $NM+1$ за счёт уменьшения точности аппроксимации в узловых точках. Такая сеть будет нуждаться в дополнительной настройке.

Многослойная сеть прямого распространения (1), настраиваемая алгоритмом обратного распространения, широко применяется в настоящее время. Алгоритм работает с набором примеров правильного поведения сети (2). Он должен найти такие параметры сети, для которых минимизируется среднеквадратичная ошибка на k-й итерации:

$$\hat{F}(\mathbf{x}) = (\mathbf{t}_k - \mathbf{a}_k)^{\mathrm{T}}(\mathbf{t}_k - \mathbf{a}_k) = \mathbf{e}_k^{\mathrm{T}}\mathbf{e}_k.$$

Алгоритм обратного распространения итерационный. Одна итерация градиентного спуска для функции $\hat{F}(\mathbf{x})$ имеет включает следующие этапы.

1. Задаётся входной вектор, который проходит через сеть в прямом направлении:

$$\mathbf{a}^0 = \mathbf{p}, \quad \mathbf{a}^{l+1} = \mathbf{f}^{l+1}(\mathbf{W}^{l+1}\mathbf{a}^l + \mathbf{b}^{l+1}), \quad l = 0, 1, \ldots, L-1, \quad \mathbf{a} = \mathbf{a}^L.$$

2. Векторы $\mathbf{s}^l$, $l = 1, \ldots, L$, распространяются через сеть в обратном направлении:

$$\mathbf{s}^L = -2\dot{\mathbf{F}}^L(\mathbf{z}^L)(\mathbf{t} - \mathbf{a}), \quad \mathbf{s}^l = \dot{\mathbf{F}}^l(\mathbf{z}^l)(\mathbf{W}^{l+1})^{\mathrm{T}}\mathbf{s}^{l+1}, \quad l = L-1, \ldots, 2, 1.$$

3. Вычисляются очередные значения весовых коэффициентов и смещений по правилу

$$\mathbf{W}^l(k+1) = \mathbf{W}^l(k) - \alpha\,\mathbf{s}^l \cdot (\mathbf{a}^{l-1})^{\mathrm{T}}, \quad \mathbf{b}^l(k+1) = \mathbf{b}^l(k) - \alpha\,\mathbf{s}^l.$$

Предполагается, что диагональные матрицы

$$\dot{\mathbf{F}}^l(\mathbf{z}^l) = \operatorname{diag}\left(\frac{df_i^{(l)}}{dz_i^l}\right), \quad l = 1, \ldots, L,$$

существуют для любого слоя. Поэтому функции активации типа hardlim здесь не годятся. Их следует заменить подходящими сигмоидными функциями. Рациональные методы организации вычислений градиентов представлены в [2]. Проиллюстрируем работу алгоритма обратного распространения на следующем примере.

Сжатие изображений. Исходное двумерное изображение представляется в виде совокупности $\mathbf{x}(k) = (x_1(k), \ldots, x_m(k))$, $k = 1, \ldots, N$, $m = \mu_1\mu_2$, непересекающихся фрагментов. Фрагменты представляют собой образцы, предъявляемые сети для обучения. Суть рассматриваемого алгоритма сжатия состоит в представлении фрагментов $\mathbf{x}(k)$ элементами метрического пространства размерности m. Пусть в этом пространстве существуют $n < m$ линейно независимых векторов $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n$, таких что

$$\mathbf{x}(k) = \hat{\mathbf{x}}(k), \quad \hat{\mathbf{x}}(k) = \hat{x}_1(k)\boldsymbol{\beta}_1 + \ldots + \hat{x}_n(k)\boldsymbol{\beta}_n, \quad \text{для каждого } k.$$

Приведённому разложению векторов $\mathbf{x}(k)$ отвечает параметр сжатия изображения $\gamma = n/m$. В реальных ситуациях указанное разложение следует рассматривать как приближённое, а векторы $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n$ — как наиболее информативную составляющую часть изображения. В качестве критерия информативности можно взять, например,

$$F = \sum_{j=1}^{m} \left[x_j(k) - \sum_{i=1}^{n} \hat{x}_i(k)\beta_{ij} \right]^2.$$

Для достижения наилучшего результата следует найти минимум этого критерия по векторам $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n$ и по компонентам $\hat{x}_1(k), \ldots, \hat{x}_n(k)$.

Возьмём двухслойную сеть. Первый слой содержит n нейронов. Входом в него служит вектор $\mathbf{p} = \mathbf{x}(k)$, а выходом — вектор $\mathbf{a}^{(1)}$ с компонентами

$$a_i^{(1)}(k) = \sum_{j=1}^{m} w_{ij}^{(1)} x_j(k), \quad i = 1, \ldots, n.$$

Этот слой выполняет роль кодера. Для каждого фрагмента $\mathbf{x}(k)$ вычисляется вектор коэффициентов кодирования $\mathbf{a}^{(1)}(k)$. Роль декодера выполняет второй слой сети, который по вектору $\mathbf{a}^{(1)}(k)$ пытается восстановить фрагмент $\mathbf{x}(k)$:

$$a_j^{(2)}(k) = \sum_{i=1}^{n} w_{ji}^{(2)} a_i^{(1)}(k), \quad j = 1, \ldots, m.$$

Значения коэффициентов весовых матриц $\mathbf{W}^{(1)}$ и $\mathbf{W}^{(2)}$ не зависят от фрагментов и определяются условием достижения минимума ошибки

$$\min_{w^{(1)}, w^{(2)}} \hat{F}, \quad \hat{F} = \left(\sum_{j=1}^{m} \left[x_j(k) - a_j^{(2)}(k) \right]^2 \right).$$

Для определения весовых коэффициентов применяется базовый вариант алгоритма обратного распространения. Входной образец выбирается случайным образом из множества фрагментов.

Описанная методика сжатия применялась для обработки изображений спектрометрической системы, входившей в состав научной аппаратуры модуля «Природа» орбитальной станции «Мир». Эти сканерные изображения представлялись в виде фрагментов размера $2^7 \times 1$, так что $m = 128$. При исследовании методики сжатия варьировалось число n нейронов первого слоя и исследовалось качество сжатия в зависимости от $\gamma = n/m$. Обучающий коэффициент выбирался равным $\alpha = 0{,}005$. Число итераций от n не зависело и было приблизительно $5 \cdot 10^4$. Рассматривались значения $\gamma = 1/32,\ 1/16,\ 1/8$. Оказалось, что описанная методика даёт наилучшие результаты по сравнению с известными методами сжатия, основанными на базисах Фурье и Уолша.

5. Кластерный анализ

Кластером называется область входных векторов, относящихся к одному и тому же классу. Рассмотрим некоторые несупервизорные методы обучения многослойных сетей решению задач выделения классов. Определим функцию активации, которая на выходе даёт тот же результат, что соревновательный слой. Функция активации $\mathbf{a} = \mathbf{compet}(\mathbf{z})$ (см. табл. 1) работает в соответствии с формулой

$$a_i = \begin{cases} 1, & i = i^*, \\ 0, & i \neq i^*, \end{cases} \quad \text{где } z_{i^*} \geqslant z_i \text{ для каждого } i \text{ и } i^* \leqslant i \text{ для каждого } z_i = z_{i^*}.$$

Для того чтобы соревновательная сеть решала задачи кластеризации, необходимо приблизить строки матрицы $\mathbf{W}$ к векторным прототипам, образующим ядро класса. Одно из возможных правил обучения сети в этом направлении есть правило инстар (6). В соревновательном слое компоненты вектора $\mathbf{a}$ не равны нулю только для победившего нейрона ($i = i^*$). Тот же результат можно получить с помощью *правила Кохонена*

$$\mathbf{w}_i(k) = \begin{cases} (1-\alpha)\mathbf{w}_i(k-1) + \alpha\mathbf{p}(k), & i = i^*, \\ \mathbf{w}_i(k-1), & i \neq i^*. \end{cases}$$

Чтобы промоделировать способность биологических систем классифицировать информацию, Кохонен предложил следующее упрощение. Его сеть, работающая по принципу *самоорганизующейся карты кластеров*, сначала определяет нейрон-победитель i^* с помощью той же процедуры, какая применена в соревновательном слое. Затем весовые векторы для всех нейронов внутри некоторой окрестности победившего нейрона изменяются в соответствии с правилом Кохонена

$$\mathbf{w}_i(k) = (1-\alpha)\mathbf{w}_i(k-1) + \alpha\mathbf{p}(k), \quad i \in N_{i^*}(d),$$

где окрестность $N_{i^*}(d)$ содержит индексы всех нейронов, которые лежат внутри круга радиуса d с центром в победившем нейроне i^*:

$$N_i(d) = \{j\colon d_{ij} \leqslant d\}.$$

Когда сети предъявляется обучающий вектор $\mathbf{p}$, весовые векторы победившего нейрона и его соседей смещаются по направлению к концу вектора $\mathbf{p}$. В результате после многих предъявлений одного и того же входного вектора соседствующие нейроны будут иметь весовые векторы близкими друг к другу. Множество нейронов в самоорганизующейся карте кластеров не обязаны иметь двумерную структуру. После обучения нейроны классифицируют приблизительно равные области в пространстве входов. Число получаемых кластеров зависит, очевидно, от размеров области притяжения победившего нейрона.

Квантизационная сеть гибридная. Она включает два слоя и использует как супервизорное, так и несупервизорное обучение в задаче классификации.

В сетях векторной квантизации каждый нейрон первого слоя приписывается часто вместе с несколькими другими нейронами к некоторому подклассу. Каждый подкласс затем приписывается одному нейрону второго слоя, объединяющему подклассы в классы. Число нейронов первого слоя $m^{(1)}$ должно, следовательно, быть по крайней мере таким же, как число нейронов второго слоя $m^{(2)}$, а в регулярном случае оно должно быть больше.

Так же как в любой соревновательной сети, каждый нейрон первого слоя настраивается на вектор прототипа, который позволяет ему классифицировать область пространства входов. Чтобы определить меру близости входного и весового векторов, можно вычислять расстояние напрямую. Первый слой работает в соответствии с формулами

$$\mathbf{z}^{(1)} = \left[z_1^{(1)}, \ldots, z_n^{(1)}\right]^{\mathrm{T}}, \quad \mathbf{a}^{(1)} = \mathbf{compet}\left(\mathbf{z}^{(1)}\right), \quad z_i^{(1)} = -\left\|\mathbf{w}_i^{(1)} - \mathbf{p}\right\|.$$

Следовательно, нейрон, весовой вектор которого ближе всего к входному вектору, выдаст значение 1, а остальные нейроны выдадут 0. Здесь победивший нейрон отмечает подкласс. Может быть несколько различных нейронов (подклассов), принадлежащих каждому классу.

Второй слой сети векторной квантизации применяется, чтобы скомбинировать подклассы в единый класс. Это делается с помощью матрицы $\mathbf{W}^{(2)}$. Столбцы $\mathbf{W}^{(2)}$ представляют подклассы, а строки представляют классы. Матрица $\mathbf{W}^{(2)}$ имеет единственную единицу в каждом столбце, тогда как все другие компоненты столбца равны нулю. Строка, в которой присутствует 1, определяет класс, к которому данный подкласс имеет отношение: $\mathbf{a}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)}$. Другими словами,

$$\left(w_{ij}^{(2)} = 1\right) \implies \text{подкласс } j \text{ принадлежит классу } i.$$

Процесс комбинирования подклассов для формирования классов позволяет сети векторной квантизации создавать сложные границы классов. Стандартный соревновательный слой может разделять только те области, границы которых являются выпуклыми. Сеть векторной квантизации преодолевает указанные трудности.

Обучение сети векторной квантизации комбинирует соревновательное обучение с супервизорным. Необходим набор образцов правильного поведения сети (2). При этом какая-нибудь одна компонента любого целевого вектора долж-

на быть равна 1, а остальные его компоненты должны быть равны нулю. Номер ненулевой компоненты целевого вектора определяет класс, к которому относится входной вектор.

Прежде чем начнётся обучение, каждый нейрон второго слоя определяется как выходной нейрон. Это формирует матрицу $\mathbf{W}^{(2)}$. Обычно равные числа нейронов скрытого слоя приписывается каждому выходному нейрону, так что каждый класс может быть сформирован одним и тем же числом выпуклых областей. Ненулевые элементы матрицы $\mathbf{W}^{(2)}$ определены следующим образом:

$$w_{ki}^{(2)} = 1, \quad \text{если скрытый } i\text{-й нейрон приписывается к классу } k.$$

Матрица $\mathbf{W}^{(2)}$, будучи однажды сформированной таким образом, впоследствии не меняется. Весовая матрица $\mathbf{W}^{(1)}$ скрытого слоя настраивается с помощью некоторой модификации правила Кохонена.

Правило обучения сети векторной квантизации, известное под названием LVQ1, работает следующим образом. На каждой итерации входной вектор $\mathbf{p}$ предъявляется сети, и вычисляется расстояние между $\mathbf{p}$ и каждым вектором-строкой весовой матрицы первого слоя. В скрытом слое происходит соревнование, и нейрон i^* оказывается победителем, а i^*-я компонента вектора $\mathbf{a}^{(1)}$ становится равной 1. Затем вектор $\mathbf{a}^{(1)}$ умножается на матрицу $\mathbf{W}^{(2)}$, которая содержит только один ненулевой элемент с индексом k^*, показывающим, что вектор $\mathbf{p}$ относится к классу k^*.

Правило Кохонена используется, чтобы откорректировать первый слой, следующим образом. Во-первых, если вектор $\mathbf{p}$ классифицирован правильно, то весовая вектор-строка $\mathbf{w}_{i^*}^{(1)}$ победившего скрытого нейрона приближается к вектору $\mathbf{p}$:

$$\mathbf{w}_{i^*}^{(1)}(k) = \mathbf{w}_{i^*}^{(1)}(k-1) + \alpha(\mathbf{p}(k) - \mathbf{w}_{i^*}^{(1)}(k-1)), \quad \text{если} \quad a_{k^*}^{(2)} = t_{k^*} = 1.$$

Во-вторых, если вектор $\mathbf{p}$ классифицирован неверно (неправильный скрытый нейрон выиграл соревнование), то весовая вектор-строка $\mathbf{w}_{i^*}^{(1)}$ отодвигается от вектора $\mathbf{p}$:

$$\mathbf{w}_{i^*}^{(1)}(k) = \mathbf{w}_{i^*}^{(1)}(k-1) - \alpha(\mathbf{p}(k) - \mathbf{w}_{i^*}^{(1)}(k-1)), \quad \text{если} \quad a_{k^*}^{(2)} = 1 \neq t_{k^*} = 0.$$

В итоге весовая вектор-строка каждого скрытого нейрона движется к векторам, относящимся к требуемому классу, и уходит от векторов, которые попадают в другие классы.

6. Сеть Хопфилда

Непрерывный по времени вариант сети Хопфилда описывается системой обыкновенных дифференциальных уравнений первого порядка в векторной форме

$$\varepsilon \frac{d\mathbf{z}(t)}{dt} = -\mathbf{z}(t) + \mathbf{W}\mathbf{a}(t) + \mathbf{b},$$

где ε — скалярный параметр, $\mathbf{z}, \mathbf{b} \in \mathbb{R}^m$, $\mathbf{a}(t) = (a_1(t), \ldots, a_m(t))$, $a_i = f(z_i)$, f — сигмоидальная функция активации, обладающая свойством симметрии $f(-z) = -f(z)$, $\mathbf{W}$ — симметричная $(m \times m)$-матрица. Как сама функция f, так и обратная к ней f^{-1} предполагаются монотонно возрастающими в строгом смысле. Начальные условия задаются равенством $\mathbf{z}(0) = \mathbf{p}$.

Возьмём функцию

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{W}\mathbf{a} + \sum_{i=1}^{m}\Big[\int_0^{a_i} f^{-1}(u)\,du\Big] - \mathbf{b}^{\mathrm{T}}\mathbf{a}, \quad \frac{dV}{da_i} = -\varepsilon\frac{d[f^{-1}(a_i)]}{da_i}\frac{da_i}{dt}.$$

Видно, что точки покоя, для которых производная $d\mathbf{a}/dt$ обращается в нуль, служат также стационарными точками функции $V(\mathbf{a})$, а в области, где функция $f^{-1}(\mathbf{a})$ приблизительно линейна, уравнения сети обеспечивают градиентный спуск к какой-нибудь из этих точек. Из-за свойств интегральных членов имеем

$$V \to +\infty \quad \text{при} \quad |a_i| \to 1 \quad \text{для каждого } i.$$

Отсюда заключаем, что в гиперкубе $|a_i| < 1$, $i = 1, \ldots, m$, существует по крайней мере один минимум функции V.

Представим функцию активации в виде $f(z) = \varphi(\gamma z)$, где $\varphi(x)$ — стандартная сигмоидальная функция, например $\mathrm{tansig}(x)$ (см. табл. 1). Параметр γ характеризует наклон функции активации в точке $z = 0$. При $\gamma \to \infty$ получим $f(z) \to \mathrm{hardlims}(z)$, а интегральные члены функции V в области $-1 < a_i < 1$, $i = 1, \ldots, m$, стремятся к нулю. Это даёт основание для больших значений γ принять

$$V \approx \tilde{V}, \quad \tilde{V} = -\frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{W}\mathbf{a} - \mathbf{b}^{\mathrm{T}}\mathbf{a}, \quad -1 < a_i < 1, \quad i = 1, \ldots, m.$$

Пусть задан набор векторов-прототипов $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$. Каждый из этих векторов имеет размерность m, а его компоненты принимают либо значение 1, либо значение -1. Предполагается, что $N \ll m$, т. е. пространство состояний достаточно велико. Для того чтобы сеть сошлась к вектору-прототипу, ближайшему к входному вектору, векторы-прототипы должны быть аргументами минимумов функции $\tilde{V}$ в области $-1 \leqslant a_i \leqslant 1$, $i = 1, \ldots, m$.

Пусть $\mathbf{b} = 0$. В качестве весовой возьмём матрицу оператора проектирования вектора $\mathbf{a}$ на линейную оболочку $\mathrm{lin}\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$:

$$\mathbf{W} = \mathbf{P}(\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}}, \quad \mathbf{W} = \mathbf{W}^{\mathrm{T}},$$

где $\mathbf{P}$ — матрица, составленная из столбцов-векторов $\mathbf{p}_1, \ldots, \mathbf{p}_N$. Очевидно, что $\mathbf{W}\mathbf{p}_k = \mathbf{p}_k$, $k = 1, \ldots, N$, а если $\mathbf{a} \perp \mathbf{p}_k$ для каждого k, то $\mathbf{W}\mathbf{a} = 0$.

Функция $\tilde{V}$ неположительна:

$$\tilde{V} = -\frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{W}\mathbf{a} = -\frac{1}{2}(\mathbf{W}\mathbf{a})^{\mathrm{T}}(\mathbf{W}\mathbf{a}).$$

Она обращается в нуль (достигает максимума) для любого вектора $\mathbf{a}$, ортогонального $\mathrm{lin}\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$. Координаты концов векторов $\mathbf{p}_k$ равны ± 1, поэтому

они лежат в вершинах куба $-1 \leqslant a_i \leqslant 1$. В точках $\mathbf{p}_k$ функция $\tilde{V}$ должна достигать минимума. Для этого необходимо, чтобы проекции всех других прототипов на любой вектор $\mathbf{p}_k$ (с учётом их знака) не превышали длины этого вектора. Достаточным условием минимальности будет условие, чтобы проекция (с учётом её знака) любой линейной комбинации всех других прототипов, лежащей в кубе $-1 \leqslant a_i \leqslant 1$, не превышала длины вектора $\mathbf{p}_k$. Самый хороший в этом смысле вариант — когда все векторы-прототипы попарно ортогональны или близки к этому.

Каждый вектор-прототип есть собственный вектор матрицы $\mathbf{W}$, и все они соответствуют общему собственному значению $\lambda = 1$. Пространство X собственных векторов матрицы $\mathbf{W}$ есть $\operatorname{lin}\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$. Всё пространство $\mathbb{R}^m$ можно представить как сумму двух непересекающихся множеств: $\mathbb{R}^m = X \cup X^{\perp}$, где $X^{\perp}$ есть ортогональное дополнение для X и ему отвечает собственное значение $\lambda = 0$.

Траектории сети Хопфилда будут стремиться попасть в те углы гиперкуба $\mathbf{a}\colon\ -1 < a_i < 1$, которые принадлежат пространству X, если только начальная точка не попала в пространство $X^{\perp}$. В этом смысле такая сеть Хопфилда может применяться как ассоциативная компьютерная память. Вместе с тем все точки из пространства $X^{\perp}$ будут стационарными точками (положениями неустойчивого равновесия) для сети. Если траектория сети попала точно на это множество, то самостоятельно выбраться оттуда она уже не сможет.

Для целей управления интерес представляют аналого-цифровые преобразователи, реализованные на базе аналоговых сетей с обратными связями (сетей Хопфилда) [5]. Пусть необходимо преобразовать напряжение u, $0 < u \leqslant u_m$, в n-разрядный двоичный код $\mathbf{a} = (a_{n-1}, a_{n-2}, \ldots, a_1, a_0)$. Тогда

$$u = m \sum_{j=0}^{n-1} 2^j a_j \pm \Delta u,$$

где m — масштаб, имеющий размерность $[v/\text{ед.}]$, $a_j \in \{0; 1\}$ — двоичный знак, принимающий значение 0 или значение 1, Δu — абсолютная погрешность преобразования, n — число двоичных знаков.

Для того чтобы каждому напряжению u соответствовал единственный код $\mathbf{a}$, разобьём диапазон максимального напряжения на 2^n отрезков. Значение n и величина Δu связаны соотношением $\Delta u = u_m/(2^{n+1} - 1)$.

Каждый двоичный знак порождается нейроном с пороговой функцией активации вида

$$a_i = \operatorname{hardlim}\left(u - m \sum_{j=i+1}^{n-1} 2^j a_j - b_i\right), \quad b_i = m \sum_{j=0}^{i-1} 2^j - \Delta u.$$

С учётом запаздывания сети Хопфилда на время τ получим следующую динамическую систему:

$$\mathbf{a}(t) = \mathbf{hardlim}\big(\mathbf{u} - 2\Delta u \mathbf{W}_1 \mathbf{a}(t - \tau) - 2\mathbf{W}_2 \Delta \mathbf{u} - \Delta \mathbf{u}\big),$$

где $\mathbf{u} = (u, u, \ldots, u)^{\mathrm{T}}$, $\Delta\mathbf{u} = (\Delta u, \Delta u, \ldots, \Delta u)^{\mathrm{T}}$,

$$\mathbf{W}_1 = \begin{pmatrix} 0 & 0 & \ldots & 0 & 0 \\ 2^{n-1} & 0 & \ldots & 0 & 0 \\ 2^{n-1} & 2^{n-2} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 2^{n-1} & 2^{n-2} & \ldots & 2^1 & 0 \end{pmatrix}, \quad \mathbf{W}_2 = \begin{pmatrix} 0 & 2^{n-2} & \ldots & 2^1 & 1 \\ 0 & 0 & \ldots & 2^1 & 1 \\ 0 & 0 & \ldots & 2^1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & 0 \end{pmatrix}.$$

Такая рекуррентная сеть Хопфилда сходится за n шагов.

7. Заключение

Мехатронные системы синтезируют механическую часть, способную совершать активные действия, и электронное оборудование, управляющее системой в целом. В данной работе в простейшей форме представлены принципы работы некоторых структур из искусственных нейронов, позволяющих за счёт распараллеливания вычислений существенно ускорить решение типичных вычислительных задач, возникающих при выработке управления, а также придать мехатронным системам некоторые полезные для автономной деятельности свойства искусственного интеллекта, такие как распознавание сложившейся ситуации в пространстве признаков, способность к обучению и прогнозу событий, формирование ассоциативной памяти и др. Элементы пространства признаков формируются из показаний сенсоров. Обработка этих показаний также есть типичная задача для нейроподобных структур, способных параллельно препарировать информацию на большом поле и осуществлять наилучшим способом её кодирование и декодирование при передаче более высоким уровням обработки. Для систем управления конкретными мехатронными системами методы решения указанных задач, приведённые в настоящей работе, могут быть реализованы существующими в настоящее время специализированными нейрочипами, взятыми в достаточном наборе.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проекты 04-01-00065, 04-01-00105), гранта НШ-1835.2003.1 и Федеральной целевой программы «Интеграция» (проекты Т0267, Б0053).

Литература

[1] Галушкин А. И. Нейрокомпьютеры. Кн. 3: Учеб. пособие для вузов / Общ. ред. А. И. Галушкина. — М: ИПРЖР, 2000.

[2] Горбань А. Н., Россиев Д. А. Нейронные сети на персональном компьютере. — Новосибирск: Наука; Сибирская издательская фирма РАН, 1996.

[3] МакКаллок У. С., Питтс В. Логическое исчисление идей, относящихся к нервной активности // Нейрокомпьютер. — 1992. — № 3, 4. — С. 40—53.

[4] Минский М., Пайперт С. Персептроны. — М.: Мир, 1971.

[5] Нейрокомпьютеры в системах обработки сигналов. Кн. 9 / Под ред. Ю. В. Гуляева, А. И. Галушкина. — М.: Радиотехника, 2003.

[6] Нейронные сети: история развития теории. Кн. 5: Учеб. пособие для вузов / Под общей ред. А. И. Галушкина, Я. З. Цыпкина. — М: ИПРЖР, 2001.

[7] Омату С., Халид М., Юсоф Р. Нейроуправление и его приложения / Под ред. А. И. Галушкина, В. А. Птичкина. — М.: ИПРЖР, 2000.

[8] Розенблатт Ф. Принципы нейродинамики. Перцептрон и теория механизмов мозга. — М.: Мир, 1965.

[9] Терехов В. А., Ефимов Д. В., Тюкин И. Ю. Нейросетевые системы управления. Кн. 8: Учеб. пособие для вузов / Общ. ред. А. И. Галушкина. — М: ИПРЖР, 2002.

[10] Anderson J. A. A simple neural network generating an interactive memory // Math. Biosci. — 1972. — Vol. 14. — P. 197—220.

[11] Grossberg S. Adaptive pattern classification and universal recoding. I. Parallel development and coding of neural feature detectors // Biol. Cybernet. — 1976. — Vol. 23. — P. 121—134.

[12] Hagan M. T., Demuth H. B., Beal M. H. Neural Network Design. — Boston: PWS, 1995.

[13] Hebb D. O. The Organization of Behavior: A Neuropsychological Theory. — New York: Wiley, 1949.

[14] Hopfield J. J. Neural networks and phisical systems with emergent collective computational abilities // Proc. Nat. Acad. Sci. U.S.A. — 1982. — Vol. 79. — P. 2554—2558.

[15] Kohonen T. Correlation matrix memories // IEEE Trans. Comput. — 1972. — Vol. 21. — P. 353—359.

[16] Kohonen T. Self-Organization and Associative Memory. — Berlin: Springer, 1987.

[17] Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1 / Rumelhart D. E., McClelland J. L., eds. — Cambridge: MIT Press, 1986.

[18] Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain // Psychological Review. — 1958. — Vol. 65. — P. 386—408.

[19] Widrow B., Hoff M. E. Adaptive switching circuits // 1960 IRE WESCON Convention Record. Part 4. — New York: IRE, 1960. — P. 96—104.

[20] Widrow B., Lehr M. A. 30 years of adaptive neural networks: Perceptron, Madaline, and backpropagation // Proc. IEEE. — 1990. — Vol. 78. — P. 1415—1441.