Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Ижевский государственный технический университет имени М. Т. Калашникова»

Кафедра «Программное обеспечение»

ПРОГРАММИРОВАНИЕ НЕЙРОННЫХ СЕТЕЙ

Учебно-методическое пособие по дисциплинам
«Методы оптимизации. Нейронные сети»,
«Нейрокомпьютерные системы» и
«Нечеткая логика и генетические алгоритмы»

Ижевск
ИжГТУ имени М. Т. Калашникова
2013

С о с т а в и т е л ь : *А. В. Коробейников*, канд. техн. наук, доц. кафедры «Программное обеспечение» ИжГТУ

Рекомендовано к использованию на заседании кафедры «Программное обеспечение» ИжГТУ (протокол №37 от 15 октября 2013 г.).

Программирование нейронных сетей : учебно-методическое пособие по дисциплинам «Методы оптимизации. Нейронные сети», «Нейрокомпьютерные системы» и «Нечеткая логика и генетические алгоритмы» / сост. А. В. Коробейников. – Ижевск: Изд-во ИжГТУ, 2013. – 44 с.

Учебно-методическое пособие предназначено для дисциплин «Методы оптимизации. Нейронные сети», «Нейрокомпьютерные системы» и «Нечеткая логика и генетические алгоритмы». Приведена теоретическая основа и варианты заданий для выполнения работ по дисциплинам. Рассмотрены возможные проблемы, возникающие при выполнении заданий; примеры решения некоторых задач; алгоритмы, которые необходимы для программной реализации заданий.

Пособие может быть использовано для проведения практических занятий и выполнения самостоятельных работ магистрантов, обучающихся по направлению подготовки 231000.68 «Программная инженерия» по программам «Разработка программно-информационных систем» и «Системы мультимедиа и компьютерная графика» при изучении дисциплин «Нейрокомпьютерные системы» и «Нечеткая логика и генетические алгоритмы».

Пособие может быть использовано для проведения практических занятий и выполнения лабораторных работ студентов, обучающихся по направлению подготовки 230100.62 «Информатика и вычислительная техника» при изучении дисциплины «Методы оптимизации. Нейронные сети».

Пособие может быть использовано для проведения практических занятий и выполнения лабораторных работ студентов, обучающихся по специальности 230105.65 «Программное обеспечение вычислительной техники и автоматизированных систем» при изучении дисциплины «Нейрокомпьютерные системы».

Оглавление

1. Теоретические основы …… 4
1.1. Модель искусственного нейрона …… 4
1.2. Функции активации …….. 5
1.3. Персептроны ……. 7
1.3.1. Однослойный персептрон ……. 7
1.3.2. Обучение по дельта-правилу …… 8
1.3.3. Проблема линейной разделимости …….. 11
1.3.4. Двухслойные и трехслойные персептроны ……. 12
1.4. Двухслойные сигмоиды ……... 13
1.5. Методы обучения многослойных нейросетей ……... 14
1.5.1. Обратное распространение ошибки …… 15
1.5.2. Тепловая машина …….. 19
1.5.3. Генетический алгоритм ……. 23

2. Работа № 1 «Персептроны»……..... 28

3. Работа № 2 «Двухслойные сигмоиды» ……..... 32

4. Требования к содержанию и оформлению отчетов ……. 42

Приложение. Форма титульного листа отчета…..……. 43

1. Теоретические основы

1.1. Модель искусственного нейрона

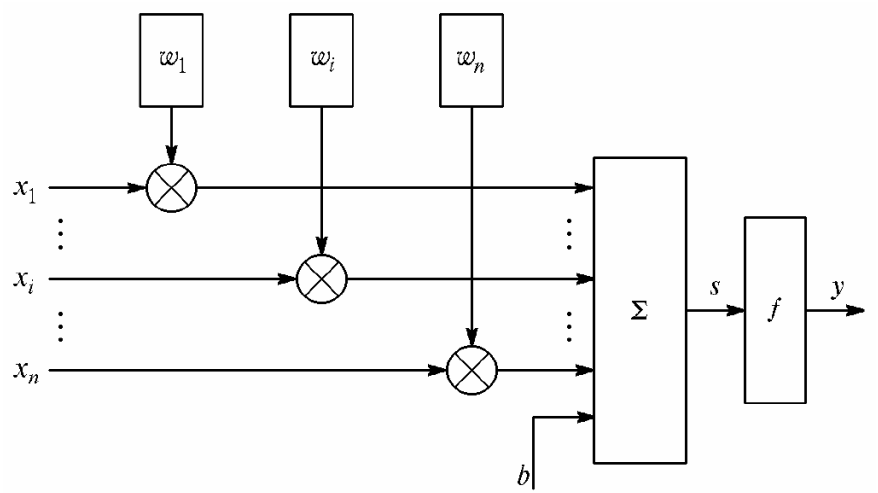


Рис. 1.1. Модель искусственного нейрона

Искусственный нейрон (рис. 1.1) имитирует в первом приближении свойства биологического нейрона. Хотя нейросетевые парадигмы разнообразны, в основе почти всех их лежит эта конфигурация.

В модели искусственного нейрона следующие обозначения:

- x_i – вход нейрона;
- w_i – весовой коэффициент по входу (*weight*), синапс;
- b – смещение (*bias*);
- Σ – операция (блок) суммирования;
- s – сумма, взвешенная сумма входов;
- y – выход нейрона;
- $f(s)$ – функция активации нейрона.

На вход искусственного нейрона поступает множество сигналов. Каждый вход x_i умножается на соответствующий вес w_i, и все произведения суммируются, определяя уровень активации нейрона. Входные сигналы образуют входной вектор X, а синапсы образуют вектор весов W. Каждый вес соответствует силе одной биологической синаптической связи. Суммирующий блок, соответствующий телу

биологического элемента, складывает взвешенные входы алгебраически, формируя значение s. В векторных обозначениях это может быть компактно записано: $s = X \cdot W$. Смещение b – вклад в сумму s, не зависящий от входов. Функция $f(s)$ – функция активации нейрона – определяет зависимость выхода нейрона y от значения суммы s. Характер функции активации может быть различным. Математически работа нейрона описывается выражениями:

$$s = \sum_{i=1}^{n} x_i \cdot w_i + b\,; \qquad y = f(s).$$

Для удобства описания работы нейрона смещение b в некоторых источниках задают с помощью дополнительного фиктивного входа $x_0 = 1$ и соответствующего синапса $w_0 = b$. Получают формулы:

$$s = \sum_{i=0}^{n} x_i \cdot w_i\,; \qquad y = f(s).$$

1.2. Функции активации. Наиболее популярные функции активации представлены в табл. 1.1 и рис. 1.2. При выполнении работ потребуется использование пороговой и сигмоидальной функций.

Пороговая функция активация исторически появилась одной из первых. Она является бинарной (цифровой), так как имеет только 2 варианта значений выхода нейрона $y = \{0, 1\}$. Функция имеет глобальный характер. Порог переключения нейрона в большинстве случаев $\theta = 0$. При необходимости порог можно задать с помощью смещения нейрона. В этом случае порог равен $-b = -w_0$, согласно выражениям:

$$s = \sum_{i=1}^{n} x_i \cdot w_i + b = 0\,, \qquad \sum_{i=1}^{n} x_i \cdot w_i = -b = w_0\,.$$

Сигмоидальная функция активации (s-образная) наряду с пороговой часто используется в искусственных нейросетях (ИНС). По своему поведению она похожа на пороговую, однако имеет гладкий вид переходного участка и поэтому является аналоговой ($0 < y < 1$). Формула сигмоида в общем виде:

$$f(s) = \frac{1}{1 + \exp(-\alpha \cdot s)}\,,$$

где α – параметр, влияющий на крутизну перехода (обычно $\alpha = 1$).

Дополнительно отметим особенности сигмоидальной функции:

- функция имеет непрерывную первую производную;
- производная функции выражается через саму себя:

$$(f(s))'_s = \alpha \cdot f(s) \cdot (1 - f(s)),$$

что используется в некоторых алгоритмах обучения ИНС;

- функция усиливает малые входные значения и ограничивает выходы при больших входных значениях ($0 < y < 1$).

Таблица 1.1. **Функции активации нейронов**

Название	Формула	Область значений
Пороговая	$f(s) = \begin{cases} 0, s < \theta \\ 1, s \geq \theta \end{cases}$	{0, 1}
Пороговая знаковая	$f(s) = \begin{cases} -1, s < \theta \\ 1, s \geq \theta \end{cases}$	{0, 1}
Линейная	$f(s) = s$	(-∞, ∞)
Линейная с насыщением	$f(s) = \begin{cases} -1, s < -1 \\ s, -1 \leq s < 1 \\ 1, s \geq 1 \end{cases}$	[-1, 1]
Полулинейная с насыщением	$f(s) = \begin{cases} 0, s < 0 \\ s, 0 \leq s < 1 \\ 1, s \geq 1 \end{cases}$	[0, 1]
Сигмоидальная	$f(s) = \frac{1}{1 + \exp(-\alpha \cdot s)}$	(0, 1)
Сигмоидальная знаковая	$f(s) = \frac{\exp(s) - \exp(-s)}{\exp(s) + \exp(-s)}$	(-1, 1)
Треугольная базисная	$f(s) = \begin{cases} 1 - s, \lvert s \rvert \leq 1 \\ 0, \lvert s \rvert \geq 1 \end{cases}$	[0, 1]
Радиальная базисная	$f(s) = \exp(-s^2)$	[0, 1]

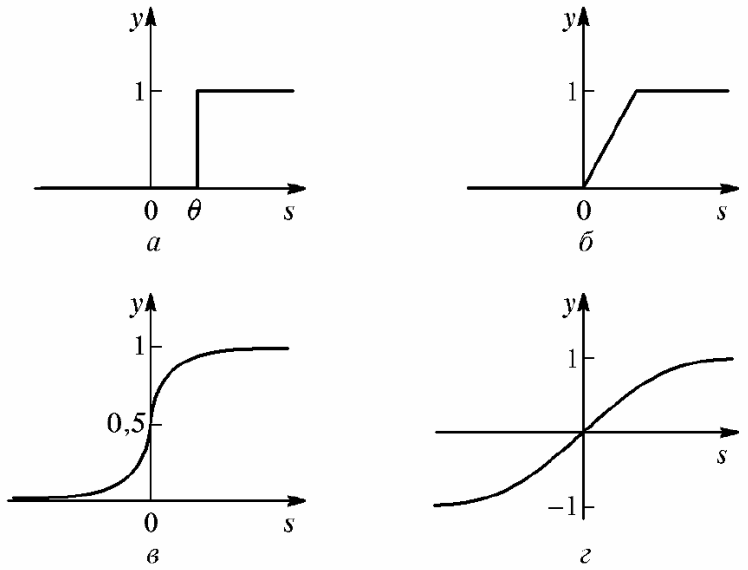


Рис. 1.2. Функции активации нейронов:
а – пороговая; *б* – полулинейная с насыщением;
в – сигмоидальная; *г* – сигмоидальная знаковая

1.3. Персептроны

1.3.1. Однослойный персептрон. В качестве научного предмета искусственные нейронные сети впервые заявили о себе в 40-е годы XX века. Стремясь воспроизвести функции человеческого мозга, исследователи создали простые аппаратные (а позже программные) модели биологического нейрона и системы его соединений.

Персептрон (*to percept* – воспринимать, представлять) представляет собой нейрон с пороговой функцией активации (рис. 1.3).

Однослойная сеть из персептронов (рис. 1.4) представляет собой набор персептронов, подключенных к одному вектору входов $X = (x_1 \ldots x_n)$ и выходы которых составляют вектор выходов $Y = (y_1 \ldots y_m)$. Как видно из структуры сети, работа каждого персептрона не зависит от других нейронов, а только от входов и значений своих синапсов. Таким образом, каждый персептрон решает свою подзадачу y_i задачи Y.

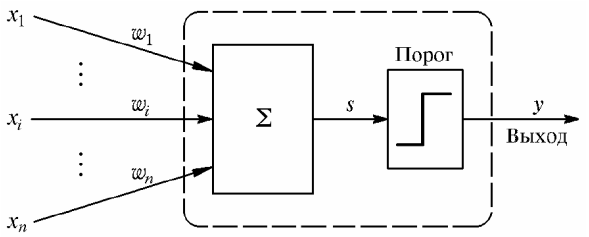


Рис. 1.3. Структура персептрона

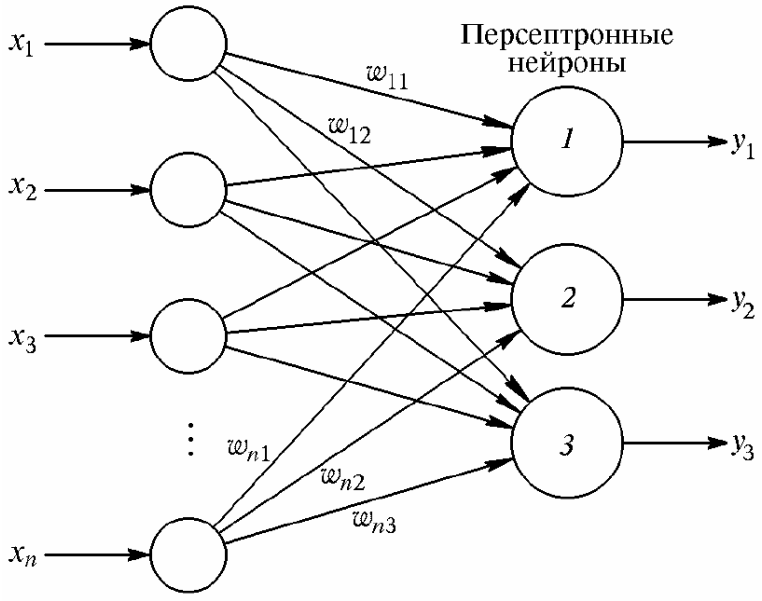


Рис. 1.4. Структура однослойной сети

1.3.2. Обучение по дельта-правилу. Персептроны традиционно связывают с работами Розенблатта 60-х годов, хотя модель персептрона существовала и ранее. Он предложил метод обучения персептрона с учителем. Доказательство теоремы обучения персептрона показало, что персептрон способен научиться всему, что он способен представлять. Важно при этом различать представляемость и обучаемость. Понятие представляемости относится к способности персептрона моделировать определенную функцию. Обучаемость же требует наличия систематической процедуры настройки весов сети для реализации этой функции.

Для иллюстрации проблемы представляемости допустим, что у нас есть множество карт, помеченных цифрами от 0 до 9. Допустим также, что мы обладаем гипотетической машиной (персептроном), способной отличать карты с нечетным номером от карт с четным номером и зажигающей индикатор на своей панели при предъявлении карты с нечетным номером (рис. 1.5).

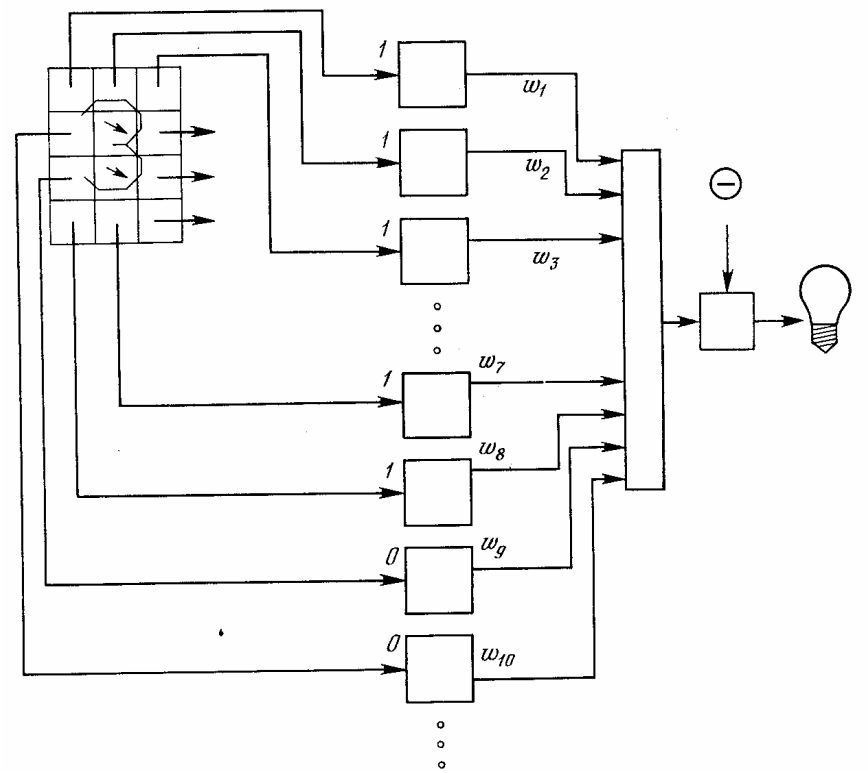


Рис. 1.5. Распознавание изображений персептроном

Для обучения персептрона нужен внешний учитель (обучение с учителем), который бы сообщал системе требуемые (верные) ответы t (*target* – цель) на выходе персептрона. При обучении с учителем необходимо задать обучающую выборку – пары X^k и t^k, где k – номер обещающего образца. K – общее число обучающих образцов.

Персептрон обучают, подавая множество образов по одному на его вход и подстраивая веса до тех пор, пока для всех образов не будет достигнут требуемый выход. Допустим, что входные образы нанесены на демонстрационные карты. Каждая карта разбита на квадраты (точки, пиксели), и от каждого квадрата на персептрон подается вход. Если в квадрате имеется линия, то от него подается $x_i^k = 1$, в противном случае $x_i^k = 0$. Множество квадратов на карте задает, таким образом, множество входов персептрона. Цель состоит в том, чтобы научить персептрон включать индикатор (выход y_i^k) при подаче на него вектора входов, задающих нечетное число, либо не включать при подаче четного числа.

Вектор $\boldsymbol{X^k}$ является образом распознаваемого изображения. Каждый вход x_i^k умножается на соответствующий вес w_i вектора синапсов W. Вычисляется сумма s^k. Если сумма превышает порог θ, то выход нейрона $y^k = 1$ (индикатор зажигается), в противном случае $y^k = 0$.

Для обучения сети образ X^k подается на вход и вычисляется выход y^k. Если y^k верен, то синапсы не меняются. Иначе веса, присоединенные к входам, имеющим $x_i^k = 1$, модифицируются, чтобы уменьшить ошибку, так как вклад в s^k вносят только входы $x_i^k <> 1$.

Алгоритм обучения:

1. Подать входной образ X^k и вычислить y^k.
2. Модификация весов (w_i):
   - если выход правильный ($y^k = t^k$), то синапсы не изменяются;
   - если выход неправильный и равен нулю ($y^k <> t^k$ и $y^k = 0$), то увеличить веса w_i, подключенные к входам $x_i^k = 1$;
   - если выход неправильный и равен единице ($y^k <> t^k$ и $y^k = 1$), то уменьшить веса w_i,подключенные к входам $x_i^k = 1$.
3. Если есть ошибки при работе сети на образцах обучающей выборки, то перейти к пункту 1.

Эпоха обучения – цикл, в котором сеть обучается для каждого образца обучающей выборки по 1 разу. За конечное число эпох персептрон обучается разделять карточки на четные и нечетные.

Дельта-правило. Обобщение алгоритма обучения персептрона называется дельта-правилом. Для этого используются формулы:

$$\delta^k = (t^k - y^k), \quad \Delta_i{}^k = \eta \cdot \delta^k \cdot x_i, \quad w_i := w_i + \Delta_i{}^k,$$

где η – скорость обучения, константа ($0 < \eta < 1$).

Дельта-правило модифицирует веса в соответствии с требуемым и действительным значениями выхода каждой полярности и соответствует приведенному выше алгоритму.

1.3.3. Проблема линейной разделимости. Один из самых пессимистических результатов работы Минского показал, что однослойный персептрон не может воспроизвести такую простую функцию, как *XOR* или «ИСКЛЮЧАЮЩЕЕ ИЛИ» (табл. 1.2).

Попытаемся реализовать эту функцию на персептроне. Составим уравнение для порога переключения такого нейрона

$$s^k = w_0 + x_1^k \cdot w_1 + x_2^k \cdot w_2 = 0.$$

Преобразуем это выражение к виду

$$x_2^k = -(x_1^k \cdot w_1 + w_0) / w_2,$$

что представляет собой уравнение прямой в плоскости (x_1, x_2).

Цель обучения – разделить обучающие образцы (рис. 1.6), так чтобы с верными ответами $t^k = 1$ (квадраты) лежали по одну сторону от этой прямой, а с $t^k = 0$ (точки) – по другую. Этого сделать невозможно.

Линейная разделимость. К сожалению, приведенный пример логической функции не единственный. Такие функции являются линейно неразделимыми, и они не реализуемы с помощью однослойных сетей.

Персептрон с n двоичными входами может иметь 2^n различных входных образов. Всего имеется $m_F = 2^{2^n}$ функций от n переменных. В табл. 1.3 приведено число линейно разделимых функций m_L. Видно, что доля линейно разделимых функций при увеличении n стремится к нулю. По этой причине использование однослойных персептронов на практике ограничено простыми задачами.

Таблица 1.2. **Функция *XOR***

k	X^k		$t_{XOR}{}^k$
	x_1^k	x_2^k	
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Таблица 1.3. **Линейно разделимые функции**

n	m_F	m_L
1	4	4
2	16	14
3	256	104
4	65536	1 882
5	$4{,}3 \cdot 10^9$	94 572
6	$1{,}8 \cdot 10^{19}$	15 028 134

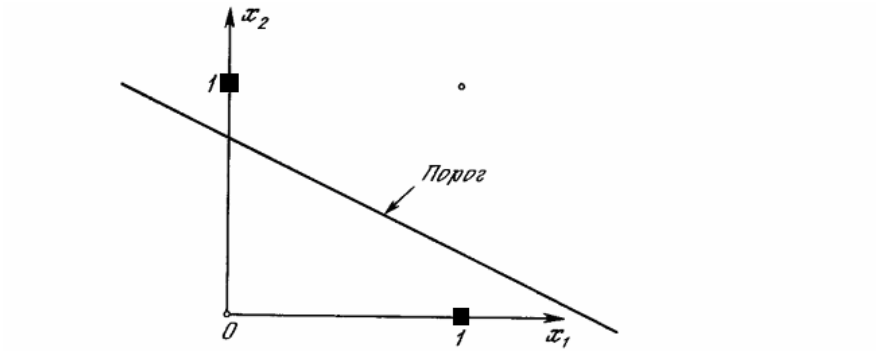


Рис. 1.6. Проблема функции *XOR*

1.3.4. Двухслойные и трехслойные персептроны. К концу 60-х годов проблема линейной разделимости была хорошо понята. Было известно, что ограничение представляемости однослойных сетей можно преодолеть, добавив дополнительные слои.

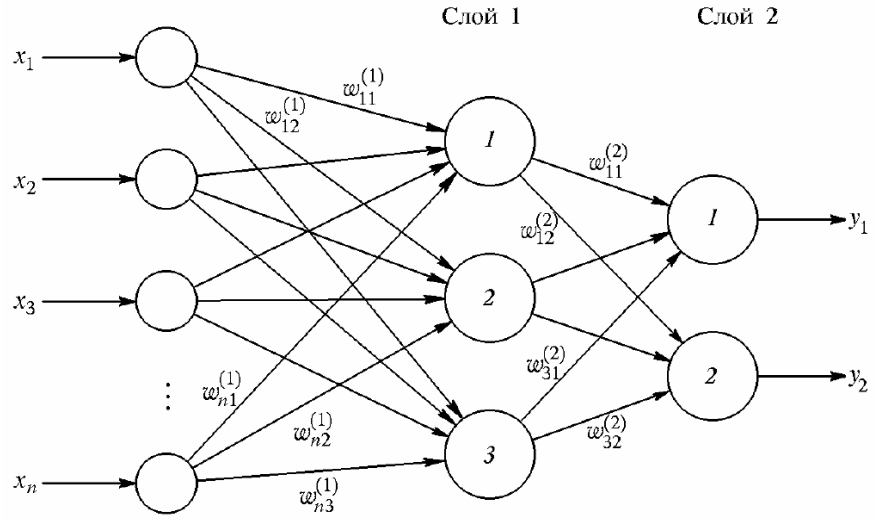


Рис. 1.7. Двухслойная сеть

Двухслойные сети можно получить каскадным соединением двух однослойных сетей (рис. 1.7). Они способны выполнять более общие классификации. Результат их работы – комбинация результа-

тов работы однослойной сети (полуплоскостей). В результате можно получить выпуклые ограниченные или неограниченные области. Область называется выпуклой, если для любых двух ее точек соединяющий их отрезок целиком лежит внутри области (рис. 1.8).

Трехслойная сеть решает еще более общие задачи. Результат ее работы – комбинация результатов работы двухслойной сети (выпуклых областей). В результате можно получить невыпуклую область (рис. 1.9) или область, состоящую из отделенных областей. Это позволяет аппроксимировать область любой формы в плоскости (x_1, x_2).

Несмотря на то, что возможности многослойных сетей были известны давно, в течение многих лет не было теоретически обоснованных алгоритмов для настройки их весов (обучения). Алгоритм обучения по дельта-правилу пригоден для обучения только выходного слоя нейросети, но не для остальных (скрытых).

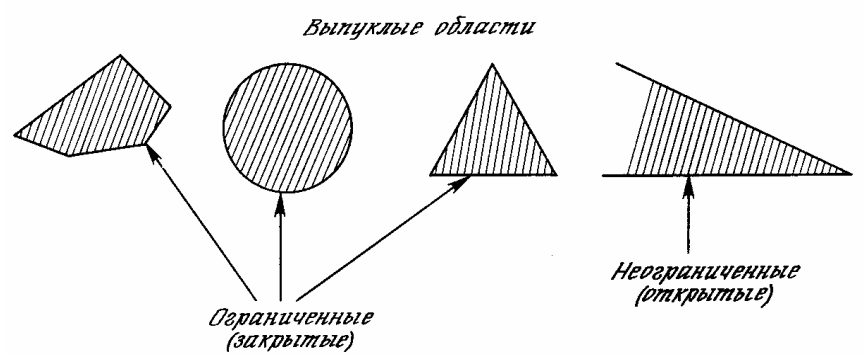


Рис. 1.8. Выпуклые области

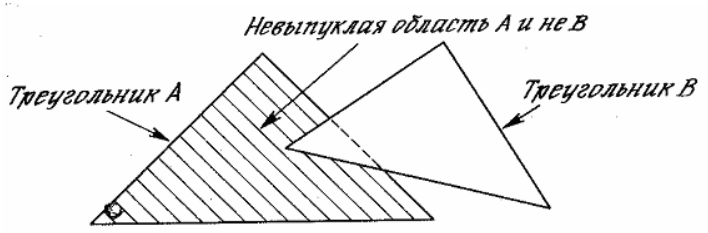


Рис. 1.9. Невыпуклые области

1.4. Двухслойные сигмоиды. Решение вопроса о необходимых и достаточных свойствах ИНС для решения того или иного рода задач

представляет собой целое направление. Так как проблема синтеза ИНС сильно зависит от решаемой задачи, дать общие подробные рекомендации сложно. В литературе приведены доказательства того, что для любого алгоритма существует ИНС, которая может его реализовать.

Многие задачи сводятся к следующей математической постановке. Необходимо построить отображение $X \to T$ такое, чтобы на каждый возможный входной сигнал X^k формировался правильный выходной сигнал T^k (верный, целевой выход, *target*). Отображение задается конечным набором пар (вход, верный выход). Число таких пар (обучающих образцов) существенно меньше общего числа возможных сочетаний значений входных и выходных сигналов.

Для любого множества обучающих образцов $\{(X^k, T^k), k = 1..K\}$ существует двухслойная однородная ИНС с последовательными связями, с сигмоидальными функциями активации и с конечным числом нейронов, которая для каждого входного вектора X^k формирует соответствующий ему выходной вектор Y^k такой, что $Y^k = T^k$.

Структура такой ИНС соответствует рассмотренному ранее двухслойному персептрону (рис. 1.9), однако функции активации всех нейронов необходимо заменить на сигмоидальные.

При практическом использовании ИНС требуют некоторого приближенного соответствия $Y^k \sim T^k$. Таким образом, ИНС решает требуемую задачу с определенной ошибкой.

Оценка числа нейронов в скрытом слое (в первом от входа) может быть выполнена последующей эмпирической формуле:

$$0.1 \cdot K - n - h \le m \le 0.5 \cdot K - n - h,$$

где K – число образцов в обучающей выборке; n – число входов; h – число выходов (число нейронов выходного слоя); m – число нейронов скрытого (первого, входного) слоя.

1.5. Методы обучения многослойных нейросетей. Обучение ИНС заключается в определении таких значений синапсов, которые обеспечат решение требуемой задачи посредством ИНС заданной архитектуры. Различают два подхода:

1) обучение с учителем; на обучающей выборке $\{(X^k, T^k), k = 1..K\}$;

2) обучение без учителя; известны только входы $\{X^k, k = 1..K\}$.

Минимизация ошибки ИНС. Наиболее изучено и более часто применяется практически обучение с учителем. При этом задача

обучения сводится в задаче оптимизации (минимизации) многопараметрической нелинейной функции, где параметрами служат значения всех синапсов ИНС, а сама функция – ошибка сети, разница между верным ответом T^k и реальным ответом ИНС Y^k.

Методы оптимизации. Для решения задачи оптимизации могут быть использованы следующие итерационные алгоритмы:

1) алгоритмы локальной оптимизации на основе частных производных (градиентный алгоритм – обратное распространение ошибки);

2) стохастические алгоритмы оптимизации (тепловая машина);

3) алгоритмы на базе эволюционного подхода (генетический алгоритм);

4) алгоритмы глобальной оптимизации.

Алгоритмы локальной оптимизации (градиентные алгоритмы) выполняют движение в сторону антиградиента частных первых производных функции ошибки по синапсам. Поэтому они имеют проблему остановки процесса оптимизации в локальных минимумах.

Алгоритм глобальной оптимизации работает на основе полного перебора возможных вариантов параметров (значений синапсов). Его использование требует значительных затрат вычислительных ресурсов.

Тепловые машины и генетические алгоритмы реализуют так называемый направленный перебор, что в большинстве задач оказывается наиболее эффективным.

1.5.1. Обратное распространение ошибки. Долгое время не было теоретически обоснованного алгоритма для обучения многослойных искусственных нейронных сетей. А так как возможности представления с помощью однослойных нейронных сетей оказались весьма ограниченными, то и вся область в целом пришла в упадок. Разработка (в середине 1980-х годов) алгоритма обратного распространения сыграла важную роль в возрождении интереса к искусственным нейронным сетям. Обратное распространение – это систематический метод для обучения многослойных искусственных нейронных сетей. Он имеет солидное математическое обоснование. Несмотря на некоторые ограничения, процедура обратного распространения сильно расширила область проблем, в которых могут быть использованы искусственные нейронные сети, и убедительно продемонстрировала свою мощь. Целью обучения сети является такая подстройка ее весов, чтобы приложение некоторого множества входов приводило к требуемому множеству

выходов. Перед началом обучения всем весам должны быть присвоены небольшие начальные значения, выбранные случайным образом. Это гарантирует, что в сети не произойдет насыщения большими значениями весов, и предотвращает ряд других патологических случаев.

При наличии у ИНС нескольких выходов обучение по каждому из них производится независимо, поэтому в дальнейшем будем рассматривать ИНС с единственным выходом (t^k, y^k).

Этапы обучения. Обучение сети обратного распространения требует выполнения следующих этапов:

1. Выбрать очередную обучающую пару (X^k, t^k) из обучающего множества; подать входной вектор на вход сети.
2. Вычислить реальный выход сети y^k.
3. Вычислить разность y^k и t^k.
4. Изменить синапсы сети так, чтобы минимизировать ошибку.
5. Повторять этапы 1…4 для каждого вектора обучающего множества, пока ошибка на всем множестве не достигнет приемлемого уровня.

Операции этапов 1, 2 – это операции работы обученной ИНС (проход вперед). Операции этапов 3, 4 – это собственно обучение (обратный проход – ошибка распространяется обратно по сети и используется для подстройки весов). После достаточного числа повторений этих этапов разность между действительными и целевыми выходами должна уменьшиться до приемлемой величины, при этом говорят, что сеть обучилась. Теперь сеть применяется, и веса не изменяются.

Выражения для коррекции синапсов. Для коррекции синапсов используется градиентный подход. При этом ошибка для текущего k-го образца вычисляется по формуле

$$E^k = \frac{1}{2} \cdot (t^k - y^k)^2 .$$

Ошибка ИНС по текущему образцу при фиксированной архитектуре ИНС и фиксированной обучающей выборке может быть снижена только путем изменения синапсов. Коррекция синапсов при подаче очередного образца в градиентном методе пропорциональна антиградиенту ошибки. Таким образом, изменение синапсов происходит в сторону уменьшения ошибки на основе производной (касательной) ошибки по синапсу:

$$w_{ij} := w_{ij} - \eta \cdot \frac{dE^k}{dw_{ij}}, \quad v_i := v_i - \eta \cdot \frac{dE^k}{dv_i},$$

где η – скорость обучения ($0 < \eta < 1$); $w_{ij} = w^{(1)}{}_{ij}$ – синапсы входного (первого) слоя; $v_i = w^{(2)}{}_i$ – синапсы выходного (второго) слоя; E^k – ошибка ИНС по текущему (k-му) образцу.

Изменение синапсов на каждом шаге будем производить в сторону уменьшения ошибки (рис. 1.10).

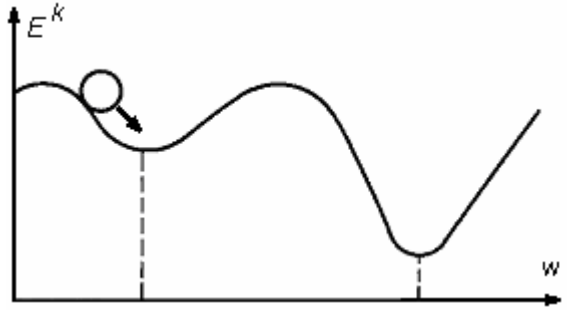


Рис. 1.10. Изменение синапса в сторону антиградиента ошибки

После подробной записи y^k и дифференцирования получим:

$$\delta^k = y^k \cdot (1 - y^k)(t^k - y^k),$$

$$v_i := v_i + \eta \cdot \delta^k \cdot y^{(1)k}{}_i,$$

$$w_{ij} := w_{ij} + \eta \cdot \delta^k \cdot v_i \cdot y^{(1)k}{}_i \cdot (1 - y^{(1)k}{}_i) \cdot x^k{}_j,$$

где $y^{(1)\,k}{}_i$ – выход первого (входного) слоя; $x^k{}_i$ – вход сети.

Отметим, что при дифференцировании используется представление производной сигмоида через сам сигмоид (см. п. 1.2).

Коррекция синапсов начинается от выходного слоя к входному (обратное распространение ошибки), причем новое значение v_i используется для коррекции w_{ij}.

Условия окончания процесса обучения

1. Получение суммарной ошибки с требуемой величиной:

$$E^\Sigma = \sum_{k=1}^{K} E^k = \sum_{k=1}^{K} \frac{1}{2}(t^k - y^k)^2 \le E^\Sigma{}_{\min}.$$

При выполнении этого условия ИНС считается обученной.

2. Слишком большое число эпох обучения (e):

$$e \ge e_{\max}.$$

Это условие блокирует возможность бесконечного обучения ИНС при невозможности достижения требуемой суммарной ошибки.

3. Снижение эффективности обучения:

$$\Delta E^{\Sigma}{}_{e} = E^{\Sigma}{}_{e-1} - E^{\Sigma}{}_{e} \le \Delta E^{\Sigma}{}_{\min} .$$

Это условие останавливает обучение при слишком малом изменении ошибки за последнюю эпоху (низкая эффективность обучения).

Конкретные значения констант ($E^{\Sigma}{}_{min}$, e_{max}, $\Delta E^{\Sigma}{}_{min}$) для приведенных условий задаются исследователем перед началом обучения.

Преимущества метода. Градиентный метод является систематическим универсальным методом и может быть применен к другим подобным ИНС с большим числом слоев или к ИНС с другими архитектурами, а также в других прикладных задачах оптимизации.

Недостатки метода. Несмотря на достигнутые на основе этого метода успехи, при обучении ИНС у метода существует ряд проблем:

1. Обратное распространение использует разновидность градиентного спуска – спуск вниз по поверхности ошибки, подстраивая веса в направлении к минимуму. Поверхность ошибки сложной сети сильно изрезана и состоит из холмов, долин, складок и оврагов в пространстве высокой размерности. Сеть может попасть в локальный минимум (неглубокую долину), когда рядом имеется гораздо более глубокий минимум (рис. 1.10). В точке локального минимума все направления ведут вверх, и сеть неспособна из него выбраться.

2. В процессе обучения сети значения весов могут в результате коррекции стать очень большими величинами. Это может привести к тому, что все или большинство нейронов будут функционировать при очень больших значениях y^k в области, где производная сигмоида очень мала. Так как посылаемая обратно в процессе обучения ошибка пропорциональна этой производной, то процесс обучения может практически замереть (паралич сети). В теоретическом отношении эта проблема плохо изучена. Обычно этого избегают уменьшением скорости обучения η, но это увеличивает время обучения (время движения по плоскому участку сигмоида).

Существуют некоторые модификации метода. Например, введение смещений ($w_{i0} \neq 0$, $v_0 \neq 0$) снижает проблему локальных минимумов. Однако принципиально эти недостатки не устраняются.

1.5.2. Тепловая машина. Имеется два класса обучающих методов.

1. *Детерминистский метод* обучения шаг за шагом осуществляет процедуру коррекции весов сети, основанную на использовании их текущих значений, а также величин входов, фактических выходов и желаемых выходов. Обучение персептрона и обратное распространение ошибки являются примером подобного детерминистского подхода. При таком подходе значения синапсов на следующем шаге обучения однозначно определены некоторыми формулами.

2. *Стохастические методы* обучения выполняют псевдослучайные изменения величин весов, сохраняя те изменения, которые ведут к улучшениям. Изменения значений синапсов при таком подходе нельзя описать точно, так как присутствует фактор случайности. Для описания процесса обучения используются вероятностные формулы.

Этапы обучения. Обучение сети на основе стохастического подхода состоит из следующих этапов:

0. Начальные установки сети и процесса обучения.

1. Расчет суммарной ошибки по всей обучающей выборке $E^{\Sigma}_{(1)}$.

2. Формирование случайных изменений синапсов сети Δw_{ij} и Δv_i. Как правило, изменяют значения синапсов целых слоев или сразу всей ИНС.

3. Расчет суммарной ошибки по всей обучающей выборке после случайных изменений синапсов $E^{\Sigma}_{(2)}$.

4. Сравнение суммарных ошибок до случайного изменения $E^{\Sigma}_{(1)}$ и после $E^{\Sigma}_{(2)}$ и принятие решения о сохранении сформированных случайных изменений.

5. Повтор этапов 1…4, пока суммарная ошибка не достигнет требуемой величины – пока сеть не будет обучена.

В отличие от метода обратного распространения ошибки в данном методе обучение ведется на основе суммарной ошибки, а не ошибки по отдельному образцу:

$$E^{\Sigma} = \sum_{k=1}^{K} E^{k} = \sum_{k=1}^{K} \frac{1}{2} (t^{k} - y^{k})^{2}.$$

Ключевой этап стохастических методов (этап 4) состоит в принятии решения о сохранении сделанных случайных изменений. При этом возможны две ситуации:

1) $E^{\Sigma}_{(1)} \geq E^{\Sigma}_{(2)}$ – случайное изменение привело к уменьшению суммарной ошибки. Такое изменение синапсов всегда сохраняется,

так как оно ведет к минимизации суммарной ошибки. Это изменение синапсов можно считать действием искусственной силы тяжести.

2) $E^{\Sigma}_{(1)} < E^{\Sigma}_{(2)}$ – случайное изменение привело к увеличению суммарной ошибки. Возможны следующие варианты действий:

(1) Сохранить изменения синапсов, ведущие к ухудшению ошибки. Однако в таком случае процесс обучения не будет иметь никакой цели, так как и уменьшение и увеличение ошибки одинаково сохраняются. Этот вариант неприемлем.

(2) Не сохранять изменения синапсов, ведущие к ухудшению ошибки. Однако в таком случае в процессе обучения будут сохраняться только изменения, ведущие к улучшению суммарной ошибки. То есть будет действовать только искусственная сила тяжести. Таким образом, мы получим аналог градиентного метода с его проблемой локальных минимумов. Этот вариант неприемлем.

(3) Сохранять ухудшения суммарной ошибки, но не всегда. В стохастических методах используют именно этот вариант действий.

Искусственная температура. Как правило, для принятия решения о сохранения ухудшения суммарной ошибки используется искусственная температура T. При этом вычисляют вероятность сохранения ухудшения ошибки $P(\Delta E^{\Sigma}, T)$, которая зависит от величины искусственной температуры и величины ухудшения суммарной ошибки $\Delta E^{\Sigma} = E^{\Sigma}_{(2)} - E^{\Sigma}_{(1)}$.

По причине использования искусственной температуры T такое обучение ИНС называется тепловой машиной. Таким образом, кроме силы тяжести в процессе обучения (оптимизации) присутствует тепловое движение, имеющее случайный характер, что позволяет осуществлять изменение суммарной ошибки вверх (ухудшение).

После вычисления $P(\Delta E^{\Sigma}, T)$ выполняют генерацию псевдослучайного числа $0 \le r \le 1$. Если $r \ge P(\Delta E^{\Sigma}, T)$, то случайное изменение синапсов сохраняется, иначе – отменяется. То есть для использования вероятности при принятии решения производится испытание – генерация случайного значения числа r.

Температура в процессе обучения уменьшается, начиная с некоторого значения T_0, происходит остывание. По этой причине этот метод оптимизации называют методом отжига металла: после нагревания металл медленно остывает, что приводит к минимизации напряжения в кристаллической решетке, а это дает снижение хрупкости изделия.

При снижении искусственной температуры T различают 3 этапа:

1. Значение T велико. Величина теплового движения также велика и практически компенсирует величину силу тяжести. Происходит хаотическое изменение синапсов без направленности к минимуму. Тепловая машина перегрета. Нахождение в этом этапе нужно сокращать.

2. Значение T среднее. При этом величина теплового движения значительна, но меньше силы тяжести. Это полезный этап работы тепловой машины – направленный перебор.

3. Значение T мало. При этом величина теплового движения незначительна, то есть в процессе обучения действует только сила тяжести. Это заключительный этап работы тепловой машины, он соответствует градиентному методу. Тепловая машина остыла. Нахождение в этом этапе нужно сокращать – прекращение обучения или завершение его методом обратного распространения ошибки.

Конкретные выражения для уменьшения искусственной температуры T в процессе обучения и для вычисления вероятности сохранения ухудшения суммарной ошибки $P(\Delta E^{\Sigma}, T)$ задаются в различных модификациях алгоритма различным образом. Однако важным вопросом при этом является сходимость процесса обучения (оптимизации) к глобальному минимуму суммарной ошибки. Рассмотрим выражения для тепловых машин Больцмана и Коши, которые гарантируют сходимость к глобальному минимуму.

Машина Больцмана. Метод предложен во второй половине 1980-х годов. Использует следующие формулы из термодинамики:

$$T(e) = \frac{T_0}{\log_2(1+e)}, \qquad P(\Delta E^{\Sigma}, T(e)) = \exp\left(\frac{-\Delta E^{\Sigma}}{k \cdot T(e)}\right),$$

где T_0 – начальное значение искусственной температуры; e – номер текущей эпохи обучения; $T(e)$ – величина текущей искусственной температуры; k – постоянная Больцмана (некоторая константа).

Основной недостаток этих соотношений – медленная скорость уменьшения искусственной температуры, что приводит к замедлению обучения.

Машина Коши использует следующие формулы:

$$T(e) = \frac{T_0}{1+e}, \qquad P(\Delta E^{\Sigma}, T(e)) = \frac{(T(e))^2}{(T(e))^2 + (\Delta E^{\Sigma})^2}.$$

Приведенное выражение для текущей искусственной температу-

ры приводит к более быстрому снижению (рис. 1.11).

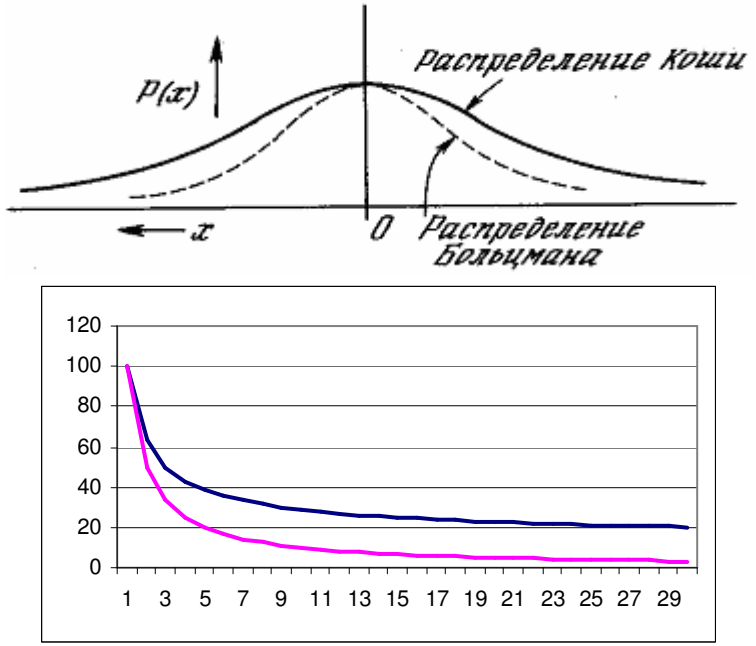


Рис. 1.11. Сравнение распределений Больцмана и Коши: $P(\Delta E^{\Sigma})$ – вверху; $T(e)$ – внизу

Однако тепловая машина Коши имеет другой существенный недостаток. На рис. 1.11 представлено сравнение распределения вероятности $P(x) = P(\Delta E^{\Sigma})$ для машин Больцмана и Коши. Недостаток машины Коши: ненулевая вероятность сохранения очень больших ухудшений суммарной ошибки, что приводит к возможному резкому ухудшению работы уже обученной ИНС.

Модификации тепловых машин

1. Ступенчатое изменение температуры. В этом методе закон изменения температуры принимает вид: $T(e \; div \; c)$, где c – количество эпох без изменения температуры.

2. Метод искусственной теплоемкости. В этом методе рассматривают соотношение

$$\Delta E^{\Sigma} \cdot \Delta T \approx const,$$

где ΔT – изменение температуры.

Требование выполнения этого соотношения приводит к тому, что в случае если в процессе обучения величина суммарной ошибки начинает значительно изменяться, то снижают скорость падения искусственной температуры. Это исключает возможность резкого падения температуры, когда синапсы ИНС попадают в локальный минимум суммарной ошибки (слишком быстрого остывания). С другой стороны, метод позволяет при незначительных изменениях ошибки увеличить скорость снижения температуры.

Преимущество метода – нахождение в процессе обучения глобального минимума суммарной ошибки ИНС.

Недостатки метода

1. Большее время обучения, по сравнению с градиентным методом. Это происходит за счет наличия в алгоритме случайных шагов (изменений синапсов), из которых только часть является полезными. Однако за счет этого достигается глобальный минимум.

2. Возможный паралич сети (слишком большие значения синапсов).

Особенность метода. Отметим также наличие в методе ряда параметров и выражений, которые влияют на эффективность обучения: T_0, $T(e \ div \ c)$, $P(\Delta E^{\Sigma}, T)$, c, k. Достижение глобального минимума возможно только при некоторых вариантах приведенных констант и выражений, которые подбирают практически.

1.5.3. Генетический алгоритм. Генетические алгоритмы (ГА) представляют собой подход к поиску наиболее оптимальной конфигурации систем на основе эволюционного подхода. Свое развитие этот подход начал в середине 70-х годов XX века. Под общим названием скрывается, скорее, подход к оптимизации (концепция), чем конкретные алгоритмы. ГА основаны на удачной аналогии, взятой из биологии вместе с соответствующими операторами и терминами.

По своей сути ГА относятся к стохастическим методам оптимизации. В них всегда присутствует элемент случайности, часто используются формулы, описывающие вероятность принятия какого-либо решения. Однако ГА имеют существенные отличия от тепловых машин:

1) биологическая аналогия и соответствующий аппарат операторов и терминов;

2) групповая оптимизация: в процессе оптимизации используется группа текущих решений (популяция).

По этой причине в литературе ГА традиционно рассматривается отдельно от стохастических методов.

Термины. Рассмотрим основные термины ГА, основанные на биологической аналогии, применительно к обучению ИНС.

Ген **–** атомарный параметр системы, кодирующий определенный признак. В биологии ген – атомарный элемент хромосомы, кодирующий признак организма. В ИНС ген – отдельный синапс сети: w_i.

Генокод – полный набор генов. В биологии – полный набор хромосом (состоят из генов) организма. В ИНС – полный набор синапсов сети: $W = (w_1 \ldots w_G)$. Синапсы всех нейронов всех слоев записываются в виде цепочки, G – общее число синапсов ИНС, длина генокода.

Особь – вариант значений генов генокода. В биологии – фенотип, организм. В ИНС – вариант решения задачи – обучения сети: W^j.

Популяция – набор особей. В биологии – группа организмов. В ИНС – группа вариантов обучения сети (группа решений задачи).

Поколение – номер популяции. Текущее поколение – текущая популяция. В биологии – смена поколений связана с ограниченным времени жизни организма. В ИНС – смена поколений связана с ограниченными вычислительными ресурсами, которые необходимы для хранения и обработки особей поколения. Можно считать, что в ИНС поколение, то же самое, что эпоха обучения в предыдущих методах обучения.

Формирование новых особей **–** генерация особей, которые составят следующее поколение. В биологии – *скрещивание* (кроссовер) родительских генокодов, а также *мутации* генокода при размножении организмов. В ИНС – генерация нового варианта обучения сети, на основе имеющихся (родительских), а также случайная модификация синапсов. Подробнее рассмотрено ниже.

Приспособленность особей – степень соответствия характеристик особи определенным требованиям. В биологии – приспособленность особи к среде обитания и к конкуренции внутри вида. В ИНС – степень приспособленности данного варианта обучения сети к поставленной задаче. В биологии эта приспособленность реализуется за счет естественного отбора: выживают и дают потомство наиболее приспособленные особи. В ИНС для продолжения обучения выбираются варианты обучения с минимальной суммарной ошибкой E^{Σ}_j для j-й особи (W^j).

Кодирование генов. При записи значений синапсов (генов) используют два варианта:

1) вещественное представление значений синапсов, например:
$W^j = (w_1, w_2, \dots, w_g) = (0.52, -15.03, \dots, 105.99)$;

2) бинарное представление, пример:
$W^j = (w_1, w_2, \dots, w_g) = (01000101, 00110100, \dots 10110001)$.

Бинарное представление получают не просто записью вещественного числа в двоичном коде, а предварительным преобразованием к целому числу, а затем кодированием этого числа кодами Грея.

При практическом применении ГА бинарное представление оказывается эффективнее вещественного, однако при бинарном представлении появляются накладные расходы на преобразование, так как при вычислении ошибки E^{Σ}_j необходимы вещественные значения.

Цикл ГА. В ходе одного цикла обучения ИНС разделяют этапы:

1. Отбор – отбор особей из текущей популяции для скрещивания.
2. Скрещивание – формирование генокода новой особи.
3. Мутации – случайные изменения генокода новой особи.
4. Замещение – формирование нового поколения на основе старых и новых особей.

Начало процесса эволюции начинается с генерации начальной популяции случайным образом. Затем многократно выполняются циклы ГА. Останов алгоритма происходит при появлении особи с требуемой суммарной ошибкой или при слишком большом числе популяций (эпох).

Отбор особей из текущей популяции для скрещивания выполняют следующими возможными способами:

1) приоритет имеют наиболее приспособленные особи;
2) приоритет имеют наименее приспособленные особи;
3) приоритет всех особей равный.

Скрещивание (*crossover*) выполняют на основе генокодов двух родительских особей. Источниками значений генов для генокода новой особи служат фрагменты родительских генокодов. Различают одноточечное и многоточечное скрещивание – по числу точек разреза родительских генокодов. Возможно случайное скрещивание: источник каждого гена выбирается случайным образом среди родителей (рис. 1.12).

Пример двухточечного скрещивания:

$$W^1 = (w^1_1, w^1_2, w^1_3, w^1_4, w^1_5, w^1_6, w^1_7, w^1_8),$$
$$W^2 = (w^2_1, w^2_2, w^2_3, w^2_4, w^2_5, w^2_6, w^2_7, w^2_8),$$
$$W^{new} = (w^2_1, w^2_2, w^1_3, w^1_4, w^1_5, w^2_6, w^2_7, w^2_8),$$

где W^1 и W^2 – родительские генокоды; W^{new} – генокод новой особи.

Мутации новой особи выполняют путем выбора нескольких случайных генов генокода, с которыми проводят случайные изменения.

В случае вещественного кодирования генов возможны действия:

1) w^{new}_i := <случайное значение> (замена);

2) Δw^{new}_i := <случайное значение> (коррекция);

3) $w^{new}_i := -w^{new}_i$ (инверсия).

В случае бинарного кодирования генов выполняют инверсию w^{new}_i := NOT w^{new}_i (рис. 1.12).

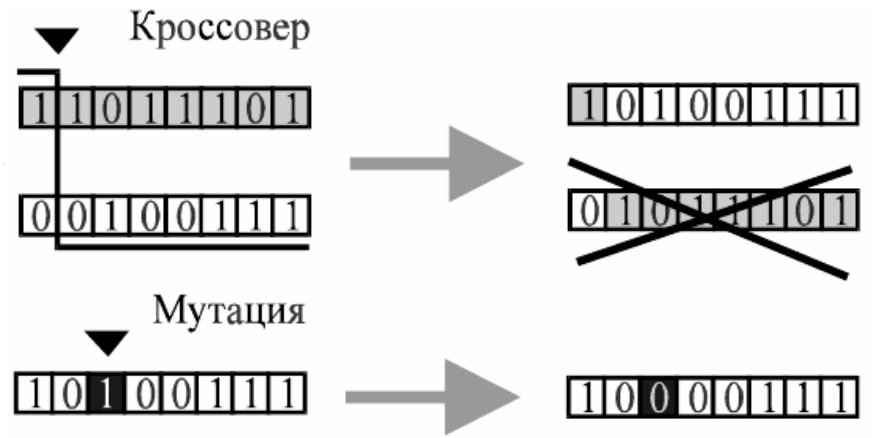


Рис. 1.12. Генетические операции

Замещение особей популяции реализуют различными способами:

1) выбор наиболее приспособленных только среди новых особей;

2) выбор особей для новой популяции наиболее приспособленных среди новых и старых особей – на основе E^{Σ}_j (стратегия элитизма);

3) замещение на основе вероятности, зависящей от приспособленности, – $P(E^{\Sigma}_j)$.

Модификации метода

1. Возможен постоянный или переменный размер популяции O_p. В случае переменного размера популяции в течение нескольких поколений размер популяции растет, а затем возвращается к минимальному значению. Таким образом, дается шанс для неприспособленных особей развить потенциал изменений.

2. Важным соотношением является число особей популяции O_p (старых) и число новых особей O_n. Как правило, при наличии большого числа новых особей эволюция идет быстрее.

3. Из проблем ГА можно отметить возможность вырождения популяции: снижение вариабельности (разнообразия) генокодов популяции. Это выражается в том, что со временем все особи становятся слишком сильно похожими друг на друга – потомками одной наиболее приспособленной особи. Для избежания такой ситуации нужно оценивать расстояние между генокодами особей W^i и W^j:

$$d_{ij} = \sqrt{\sum_{g=1}^{G} \left(w^i{}_g - w^j{}_g \right)^2} \ .$$

Если величина d_{ij} меньше порога, то одна из особей-близнецов удаляется или генерируется заново случайным образом.

Преимущества метода

1. Так как ГА можно рассматривать вариантом стохастического метода, то на основе ГА достигается глобальный минимум.

2. ГА является привлекательным методом по причине простоты концепции и реализации.

3. Если в задаче (системе) есть подзадачи (подсистемы), то за счет скрещивания возможно резкое улучшение суммарной ошибки новой особи, если каждый родитель успешно решает отдельные подзадачи.

Недостатки метода

1. ГА – это, скорее, концепция, чем конкретный алгоритм, и успешность его использования зависит от деталей реализации.

2. В ГА сложно управлять эффективностью обучения. Например, в тепловых машинах был определенный набор параметров и соотношений, которые влияли на эффективность. В ГА такого четко определенного набора нет, реализация каждого этапа повлияет на конечный результат.

2. Работа № 1 «Персептроны»

2.1. Цель работы: практическое освоение основ нейронных сетей.

Выполняется в виде программы на языке высокого уровня. В ходе работы использовать персептроны – пороговую функцию активации (порог $\theta = 0$). При необходимости использовать вход смещения $b = w_0$.

2.2. Задание на выполнение работы

Задание 1. Реализовать с помощью персептронов логическую функцию двух переменных в соответствии с номером варианта. Вид функции приведен в табл. 2.1.

Если функция для варианта линейно неразделима, то необходимо использовать число слоев сети 2 или 3. Допускается использовать ручное задание синаптических коэффициентов (без обучения сети).

Задание 2. Реализовать с помощью персептронов распознавание изображений букв, заданных в виде матриц (5×5, 8×8 или других). Выполнить в виде сети из одного слоя. Необходимо выполнить обучение сети по дельта-правилу на заданном наборе букв.

2.3. Варианты заданий

Задание 1. Варианты приведены в табл. 2.1.

Задание 2. Набор букв для распознавания определяется фамилией студента (‘Иванов‘ = ‘И‘, ‘В‘, ‘А‘, ‘Н‘, ‘О‘). Достаточно выбрать 5 букв фамилии.

Таблица 2.1. **Варианты для задания 1**

Вариант	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 10$	$x_1x_2 = 11$
2	0	0	0	1
3	0	0	1	0
5	0	1	0	0
7, 1, 16	0	1	1	0
8	0	1	1	1
9	1	0	0	0
10, 4, 6	1	0	0	1
11	1	0	1	0
12	1	0	1	1
13	1	1	0	0
14	1	1	0	1
15	1	1	1	0

2.4. Обсуждение выполнения работы. Для выполнения каждого задания необходимо определить: архитектуру сети, образцы обучающей выборки и алгоритм обучения.

Задание 1. Обучающая выборка представлена в табл. 2.2.

Таблица 2.2. **Обучающая выборка для задания 1**

K	X^k		$t_1{}^k$	$t_2{}^k$	$t_3{}^k$
	$x_1{}^k$	$x_2{}^k$			
1	0	0	0	0	0
2	0	1	1	0	1
3	1	0	0	1	1
4	1	1	0	0	0

Рассмотрим реализацию линейно разделимой функции t_1. В этом случае сеть состоит из одного персептрона (однослойная сеть).

Обучение такой сети можно проводить по дельта-правилу (смотри обсуждение задания 2) или подобрать значения синапсов вручную.

Рассмотрим пример подбора синапсов для функции t_1. Для этого нужно определить такие w_0, w_1, w_2, чтобы выполнялись условия:

1) $t_1{}^1 = 0$;
   $w_0 + w_1 \cdot x_1{}^1 + w_2 \cdot x_2{}^1 = s^1 < 0$;
   $w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0$;
   $w_0 < 0$;
2) $t_1{}^2 = 1$;
   $w_0 + w_1 \cdot x_1{}^2 + w_2 \cdot x_2{}^2 = s^2 \geq 0$;
   $w_0 + w_2 \geq 0$;
3) $t_1{}^3 = 0$;
   $w_0 + w_1 \cdot x_1{}^3 + w_2 \cdot x_1{}^3 = s^3 < 0$;
   $w_0 + w_1 < 0$;
4) $t_1{}^4 = 0$;
   $w_0 + w_1 \cdot x_1{}^4 + w_2 \cdot x_1{}^4 = s^4 < 0$;
   $w_0 + w_1 + w_2 < 0$.

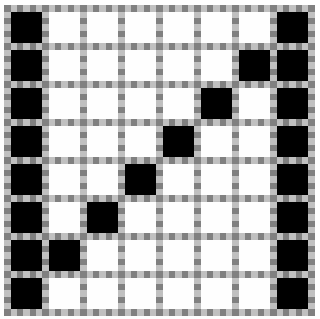
Приведенным условиям удовлетворяют значения:

$w_0 = -1$; $w_1 = -2$; $w_2 = 2$.

Для реализации линейно неразделимой функции t_3 потребуется минимум 3 нейрона (двухслойная сеть). При этом 2 нейрона первого слоя будут реализовывать функции t_1 и t_2, а третий нейрон функцию $t_3 = t_1$ ИЛИ t_2.

Задание 2. Архитектура сети представляет собой однослойную нейросеть из персептронов. Количество входов определяется размером изображения букв. Например, для изображения размером 8×8 (см. рис.) число входов составит $n = 8 \cdot 8 = 64$. Значения входов бинарные $x_i^k = \{0, 1\}$. Количество персептронов (выходов) совпадает с количеством букв фамилии, которые нужно распознавать. Нейроны должны иметь вход $x_0 = 1$ для формирования смещения за счет синапсов w_{i0}.

Количество нейронов сети и размер обучающей выборки совпадают. Таким образом, заданные изображения букв являются эталонами. Если нейрон распознал букву, поданную на вход, то на его выходе должен быть сформирован единичный выход. Обучающая выборка представлена в табл. 2.3. Каждая строка этой таблицы представляет собой обучающий образец: X^k (входной вектор, изображение буквы) и T^k (вектор верных ответов сети). Каждый столбец t_i представляет собой функцию работы каждого нейрона.



Изображение буквы «И»

Таблица 2.3. **Обучающая выборка для задания 2**

k	X^k	T^k				
		$t_1^k = t_И^k$	$t_2^k = t_В^k$	$t_3^k = t_А^k$	$t_4^k = t_Н^k$	$t_5^k = t_О^k$
1	**И**	1	0	0	0	0
2	**В**	0	1	0	0	0
3	**А**	0	0	1	0	0
4	**Н**	0	0	0	1	0
5	**О**	0	0	0	0	1

Обучение каждого персептрона требуется проводить по дельта-правилу. В результате обучения каждый персептрон должен реализовывать функцию, приведенную в табл. 2.3, и суммарная ошибка будет равна 0. Это гарантируется тем, что функция работы каждого нейрона содержит только один входной вектор X^k, вызывающий выход нейрона, равный $t_i^k = 1$, и, таким образом, проблемы линейной разделимости не возникает.

Алгоритм обучения

1. Начальные установки:
   - $e := 0$; // номер эпохи обучения;
   - $E^{\Sigma}_{min} := 0$; // требуемая суммарная ошибка сети;
   - $w_{ij} := Random \cdot 10 - 5$; // случайные значения синапсов.
2. Новая эпоха:
   - $e := e + 1$;
   - $E^{\Sigma}_{e} := 0$; // обнуление суммарной ошибки эпохи;
   - $k := 0$; // номер текущего обучающего образца.
3. Следующий образец: $k := k + 1$.
4. Работа сети: подать X^k и вычислить Y^k.
5. Коррекция синапсов: $w_{ij} := w_{ij} + \eta \cdot (t_i^k - y_i^k) \cdot x_j^k$;
   // выполнить по всем нейронам (i) и по всем входам (j).
6. Вычисление ошибки: $E^{\Sigma}_{e} := E^{\Sigma}_{e} + |t_i^k - y_i^k|$;
   // выполнить по всем нейронам (i).
7. Условие окончания эпохи: если $k = K$, то перейти к пункту 8, иначе – к пункту 3.
8. Условие окончания обучения: если $E^{\Sigma}_{e} = 0$, то перейти к пункту 9, иначе – перейти к пункту 2.
9. Конец обучения: сеть обучена.

При демонстрации работы обученной сети необходимо подавать как изображения букв из обучающей выборки, так и другие похожие на них или произвольные. В качестве результата работы сети вывести вектор значений всех выходов сети Y^k. Дополнительно можно вывести вектор суммы нейронов сети S^k.

Список литературы

1. *Осовский*, *С*. Нейронные сети для обработки информации. – М. : Финансы и статистика, 2004. – 344 с.

2. *Уоссермен*, *Ф*. Нейрокомпьютерная техника : Теория и практика. – М. : Мир, 1992. – 184 с.

3. *Круглов*, *В*. *В*. Искусственные нейронные сети / В. В. Круглов, В. В. Борисов. – М. : Горячая линия – Телеком, 2002. – 382 с.

3. Работа №2 «Двухслойные сигмоиды»

3.1 Цель работы: практическое освоение методов обучения универсальных многослойных ИНС. Выполняется в виде программы на языке высокого уровня. В ходе работы использовать двухслойную нейросеть. Нейроны с сигмоидальной функцией активации.

3.2. Задание на выполнение работы

Задание 1. Аппроксимация монотонной функции $t = f(x)$. Обучение: обратное распространение ошибки.

Задание 2. Прогнозирование $t = f(k) = F(f(k\text{-}1)... f(k\text{-}n))$. Обучение: обратное распространение ошибки.

Таблица 3.1. **Выполнение заданий по вариантам**

Вариант	Задание 1	Задание 2	Задание 3	Задание 4
1	-	+	-	+
2	-	+	+	-
3	+	-	-	+
4	+	-	+	-
5	-	+	-	+
6	-	+	+	-
7	+	-	-	+
8	+	-	+	-
9	-	+	-	+
10	-	+	+	-
11	+	-	-	+
12	+	-	+	-
13	-	+	-	+
14	-	+	+	-
15	+	-	-	+
16	+	-	+	-
17	-	+	-	+
18	-	+	+	-
19	+	-	-	+
20	+	-	+	-

Задание 3. Аппроксимация немонотонной функции $t = f(x)$. Обучение: тепловая машина.

Задание 4. Аппроксимация немонотонной функции $t = f(x)$. Обучение: генетический алгоритм.

3.3. Варианты заданий. Необходимо выполнить 2 задания в соответствии с табл. 3.1. Например: для варианта 7 выполнить задания 1 и 4.

Задание 1. Вид функции берется из табл. 3.2, в соответствии с номером варианта. Для реализации взять монотонный участок [a, b] функции.

Задание 2. Вид прогнозируемой функции $f(x)$ берется из табл. 3.3, в соответствии с номером варианта. Прогноз нужно выполнять для участка 1.5…2 периода функции (3…4 монотонных участков).

Задание 3, 4. Функция периодическая, берется из табл. 3.3. Для реализации взять немонотонный участок [a, b] функции длиной в 1.5…2 периода (3…4 монотонных участков).

Таблица 3.2. **Варианты по выполнению задания 1**

Вариант	$f(x)$	a	b
1	$0.37 \cdot exp(\sin x)$	1.5	4
2	$0.5 + x \cdot lg\ x$	0.4	4
3	$(x + 1.9) \cdot \sin (x / 3)$	-1	4
4	$(1 / x) \cdot ln (x + 2)$	0.1	4
5	$\cos (3 \cdot x) / (2 \cdot x + 1.7)$	-0.8	2.5
6	$(2 \cdot x + 0.6) \cdot \cos (x / 2)$	-0.2	4
7	$2.6 \cdot x^2 \cdot ln\ x$	0.1	4
8	$(x^2 + 1) \cdot \sin (x - 0.5)$	-1	2.6
9	$x^2 \cdot \cos (x / 4)$	-1	4
10	$\sin (0.2 \cdot x - 3) / (x^2 - 1)$	-1	4
11	$3 \cdot x + ln\ x$	0.1	4
12	$4 \cdot x \cdot exp (x^2)$	-1	4
13	$3 \cdot x^2 + tg\ x$	1.6	4
14	$(3 \cdot x^2 + \sin x) / x^2$	0.1	4
15	$3 \cdot x \cdot exp (\cos x)$	-1	1.1
16	$x^2 + tg (x / 2)$	0	3.1
17	$x^{0.5} exp (- x)$	0.5	4
18	$3.1 \cdot x \cdot (ln\ x)^2$	1	4
19	$(x - 0.8) \cdot ln (x / 2)$	1.3	4
20	$(x - 3.1) \cdot exp (tg\ x)$	0	1.4

Таблица 3.3. **Варианты по выполнению задания 2…4**

Вариант	***f*(*x*)**
1	$sin(x)$
2	$cos(x)$
3	$sin(2 \cdot x)$
4	$cos(2 \cdot x)$
5	$exp(sin(x))$
6	$exp(cos(x))$
7	$sin(x) \cdot sin(x)$
8	$cos(x) \cdot cos(x)$
9	$exp(sin(x) \cdot sin(x))$
10	$exp(cos(x) \cdot cos(x))$
11	$tg(x)$
12	$1 / tg(x)$
13	$tg(2 \cdot x)$
14	$1 / tg(2 \cdot x)$
15	$sin(x) \cdot tg(x)$
16	$cos(x) / tg(2 \cdot x)$
17	$x \; mod \; 3$
18	$(x \; mod \; 3) \cdot (x \; mod \; 3)$
19	$(10 - x) \; mod \; 3$
20	$((10 - x) \; mod \; 3) \cdot ((10 - x) \; mod \; 3)$

Примечание. Аналитические функции в задании по прогнозированию (задание 2) могут быть заменены преподавателем на практические данные, представленные виде массива значений.

3.4. Обсуждение выполнения работы. Для выполнения каждого задания необходимо определить: архитектуру сети, образцы обучающей выборки и алгоритм обучения.

Во всех заданиях используется двухслойная нейросеть с одним выходом и сигмоидальными функциями активации всех нейронов. При необходимости наличия у нейронов смещения ввести дополнительный вход $x_0 = 1$.

Масштабирование выходов и входов сети. На выходе нейросети находится нейрон с сигмоидальной функцией активации. Поэтому во всех заданиях необходимо масштабирование диапазона требуемых (верных) ответов t^k сети к диапазону сигмоида.

Например, для табл. 3.4, значения из диапазона задачи $1 \le t^k \le 100$ нужно преобразовать в диапазон $0.1 \le t^{*k} \le 0.9$ и при обучении нейросети использовать значение t^{*k} (внутреннее значение). При демонстрации работы нейросети значения выходов сети y^{*k} (внутреннее) нужно из диапазона нейросети $0.1 \le y^{*k} \le 0.9$ преобразовать во внешний диапазон (диапазон задачи) $1 \le y^k \le 100$ и это значение отобразить пользователю. Диапазон выходов 0.1…0.9 выбран вместо полного диапазона сигмоида 0…1, т. к. на участках, близких к 0 и 1, обучение происходит медленно.

Дополнительно, но не обязательно, возможно масштабирование входов из (внешнего) диапазона задачи (табл. 3.4) $1 \le x^k \le 10$ в диапазон сети (внутренний) $-5 \le x^{*k} \le +5$, это значение нужно использовать для обучения и работы сети. Данное преобразование используется, чтобы сбалансировать положительные и отрицательные значения входов, что ускоряет процесс обучения.

Задание 1. В заданиях 1, 3, 4 (аппроксимация) нейросеть имеет только один вход x. Пример обучающей выборки приведен в табл. 3.4 (число образцов $K = 10$). Обучение: метод обратного распространения ошибки.

Таблица 3.4. **Пример обучающей выборки для задания 1 ($t = x^2$)**

k	1	2	3	4	5	6	7	8	9	10
x^k	1	2	3	4	5	6	7	8	9	10
t^k	1	4	9	16	25	36	49	64	81	100

Алгоритм обучения

1. Начальные установки:
   - e:= 0; // номер эпохи обучения;
   - задать E^{Σ}_{min}; // требуемая суммарная ошибка сети;
   - задать e_{max}; // максимальное число эпох обучения;
   - задать ΔE^{Σ}_{min} // минимальное улучшение ошибки;
   - w_{ij}:= *Random* · 10 – 5; // случайные значения синапсов слоя 1;
   - v_i:= *Random* · 10 – 5; // случайные значения синапсов слоя 2.
2. Новая эпоха:
   - e:= e + 1;
   - E^{Σ}_{e}:= 0; // обнуление суммарной ошибки эпохи;
   - k:= 0; // номер текущего обучающего образца.

3. Следующий образец:

- $k := k + 1$.

4. Работа сети:

- подать X^k и вычислить y^k.

5. Коррекция синапсов:

- $w_{ij} := w_{ij} + \Delta w_{ij}$;
- $v_i := v_i + \Delta v_i$; // выполнить по всем нейронам i и по всем входам j в соответствии с формулами, приведенными в теоретической части.

6. Вычисление ошибки:

- $E^{\Sigma}_e := E^{\Sigma}_e + 0.5 \cdot (t^k - y^k)^2$.

7. Условие окончания эпохи:

- если $k = K$, то перейти к пункту 8, иначе – к пункту 3.

8. Условия окончания обучения:

- $E^{\Sigma}_e \le E^{\Sigma}_{min}$; // нейросеть обучена;
- $e \ge e_{max}$; // нейросеть не обучена, достигли предельного числа эпох;
- $\Delta E^{\Sigma}_e = (E^{\Sigma}_{e-1} - E^{\Sigma}_e) \le \Delta E^{\Sigma}_{min}$; // нейросеть не обучена, эффективность обучения меньше допустимой;
- если выполняется какое-либо из условий, то перейти к пункту 9, иначе – перейти к пункту 2.

9. Конец обучения.

При формировании обучающей выборки по данному заданию необходимо выбирать монотонный участок функции, так как в противном случае может проявиться проблема локальных минимумов.

Для визуального отображения многомерной поверхности ошибки (число измерений равно числу синапсов плюс значение ошибки) можно построить 3*D*-проекции. Для этого нужно зафиксировать значения всех синапсов кроме двух, значения которых изменяют в некотором диапазоне. Третья координата – значение ошибки E^{Σ}_e (рис. 3.1).

Задание 2. В задании 2 (прогнозирование) на входы подаются несколько последовательных значений функции (порядка 10, например: $f(k\text{-}10)\ldots f(k\text{-}1)$), которые используются для предсказании неизвестного, следующего за ними значения функции (f(k)). Пример обучающей выборки представлен в табл. 3.5.

Алгоритм обучения совпадает с алгоритмом, приведенным для задания 1. Проблема локальных минимумов в задании 2 менее выражена, так как самым грубым прогнозом является $f(k) = f(k-1)$, а это монотонная функция от $f(k-1)$.

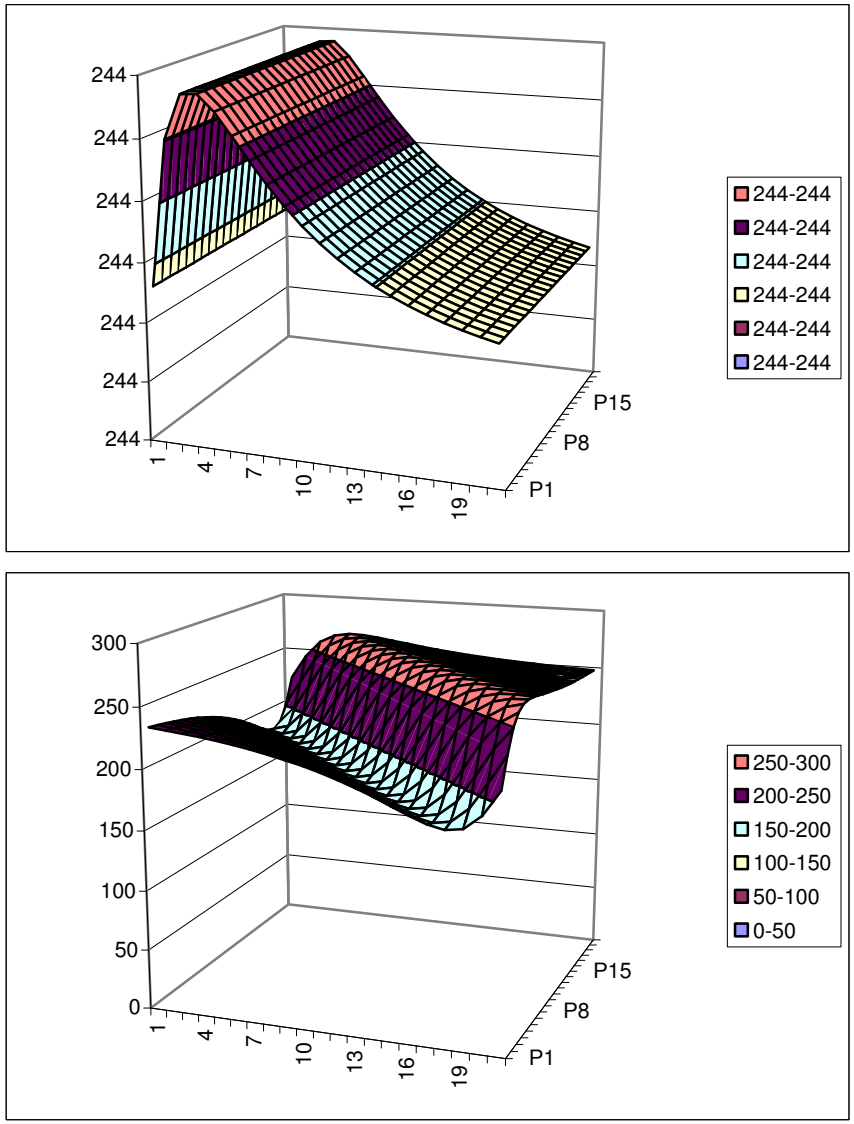


Рис. 3.1. Пример 3*D*-проекций многомерной поверхности ошибки

Таблица 3.5. **Пример обучающей выборки для задания 2**

K	1	2	3	4	5	6	7	8	9	10
X^k	$f(-9)$ … $f(0)$	$f(-8)$ … $f(1)$	$f(-7)$ … $f(2)$	$f(-6)$ … $f(3)$	$f(-5)$ … $f(4)$	$f(-4)$ … $f(5)$	$f(-3)$ … $f(6)$	$f(-2)$ … $f(7)$	$f(-1)$ … $f(8)$	$f(0)$ … $f(9)$
t^k	$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$

Задание 3 является усложнением задания 1, для решения используется более эффективный алгоритм обучения – тепловая машина.

Алгоритм обучения

1. Начальные установки:
   - $e := 0$; // номер эпохи обучения;
   - задать E^{Σ}_{min}; // требуемая суммарная ошибка сети;
   - задать e_{max}; // максимальное число эпох обучения;
   - задать начальную искусственную температуру T_0;
   - $w_{ij} := Random \cdot 10 - 5$; // случайные значения синапсов слоя 1;
   - $v_i := Random \cdot 10 - 5$; // случайные значения синапсов слоя 2.
2. Новая эпоха: $e := e+1$.
3. Определение $E^{\Sigma}_{e(1)}$ – суммарная ошибка до случайного изменения синапсов:
   - $E^{\Sigma}_{e} := E^{\Sigma}_{e} + 0.5 \cdot (t^k - y^k)^2$; // выполнить по всем K образцам (k).
4. Случайное изменение синапсов:
   - $w_{ij} := w_{ij} + \Delta w_{ij} = w_{ij} + (Random \cdot 10 - 5)$;
   - $v_i := v_i + \Delta v_i = v_i + (Random \cdot 10 - 5)$; // выполнить по всем нейронам i и по всем входам j.
5. Определение $E^{\Sigma}_{e(2)}$ – суммарная ошибка после случайного изменения синапсов.
6. Условия сохранения случайного изменения синапсов:
   - если $E^{\Sigma}_{e(1)} \geq E^{\Sigma}_{e(2)}$, то сохранить изменения, перейти к пункту 8;
   - если $E^{\Sigma}_{e(1)} < E^{\Sigma}_{e(2)}$, то перейти к пункту 7.
7. Принятие решения о сохранении ухудшения ошибки:
   - вычислить $P(\Delta E, T)$; // согласно формулам в теоретической части;
   - сгенерировать случайное число $r := Random$; // $0 \leq r \leq 1$;
   - если $P(\Delta E, T) \geq r$, случайное изменение синапсов сохраняется, иначе – отменяется;
8. Изменение искусственной температуры: $T := T(e)$; // согласно формулам в теоретической части.

10. Условия окончания обучения:

- $E^{\Sigma}_{e(1)} \le E^{\Sigma}_{min}$; // нейросеть обучена;
- $e \ge e_{max}$; // нейросеть не обучена, достигли предельного числа эпох;
- если выполняется какое-либо из условий, то перейти к пункту 10, иначе – перейти к пункту 2.

10. Конец обучения.

Задание 4 является усложнением задания 1, для решения используется более эффективный алгоритм обучения – генетический алгоритм.

Алгоритм обучения

1. Начальные установки:
   - e:= 0; // номер эпохи (поколения) обучения;
   - задать E^{Σ}_{min}; // требуемая суммарная ошибка сети;
   - задать e_{max}; // максимальное число эпох обучения;
   - задать O_p; // размер популяции;
   - задать O_n; // количество новых особей популяции;
   - сгенерировать начальную популяцию (O_p особей);
   - w_g:= $Random \cdot 10 - 5$; // синапсы нейросети; для всех G генов (синапсов) w_g генокода g.
   - определить E^{Σ}_j; // суммарная ошибка особи для всех O_p особей.
2. Новая эпоха: e:= e+1.
3. Отбор особей для формирования новых особей.
4. Формирование новых O_n особей путем:
   - скрещивания;
   - мутаций.
5. Определение суммарных ошибок новых особей: определить E^{Σ}_j; // суммарная ошибка особи j; для всех O_n особей.
6. Замещение особей:
   - сортировка всех особей ($O_p + O_n$) по суммарной ошибке E^{Σ}_j;
   - отбор лучших особей числом O_p для следующего поколения.
7. Условия окончания обучения:
   - $min(E^{\Sigma}_j) \le E^{\Sigma}_{min}$; // нейросеть обучена, если есть особь с требуемой суммарной ошибкой;
   - $e \ge e_{max}$; // нейросеть не обучена, достигли предельного числа эпох;
   - если выполняется какое-либо из условий, то перейти к пункту 8, иначе – перейти к пункту 2.
8. Конец обучения.

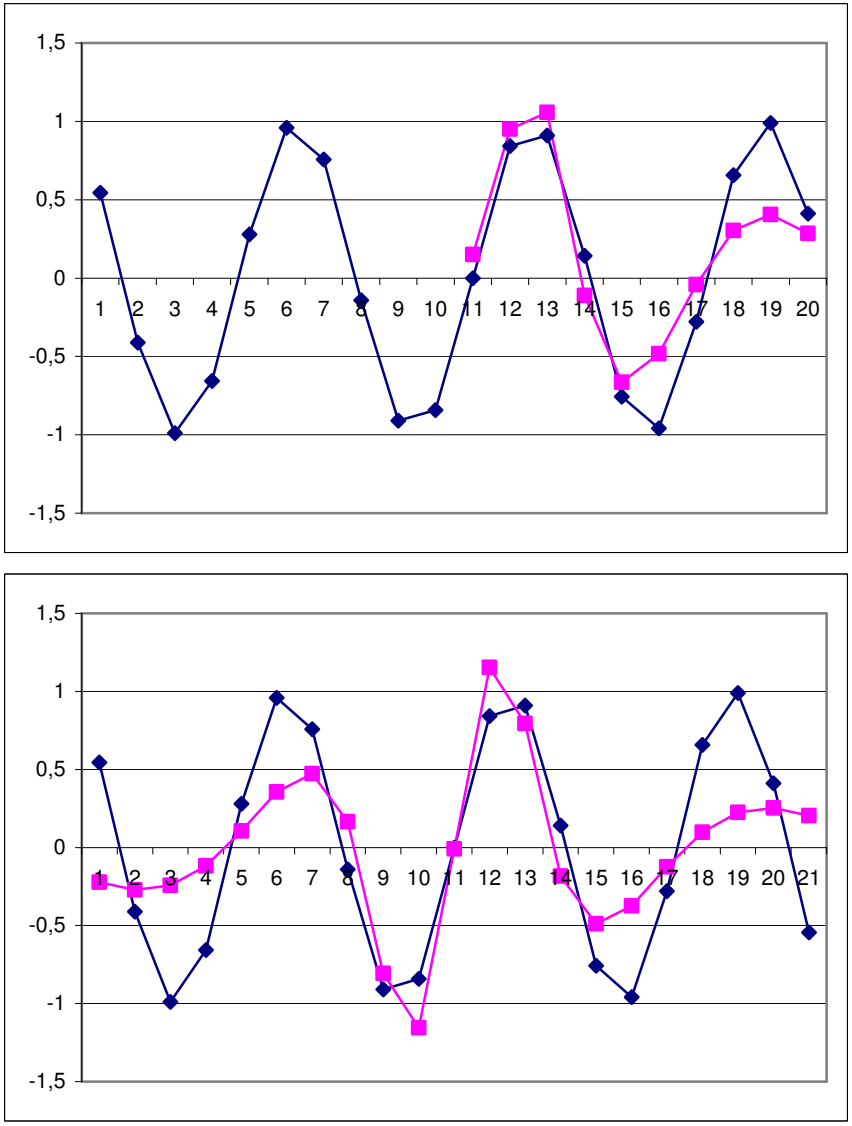


Рис. 3.2. Пример результатов выполнения заданий 2 (вверху) и 3, 4 (внизу)

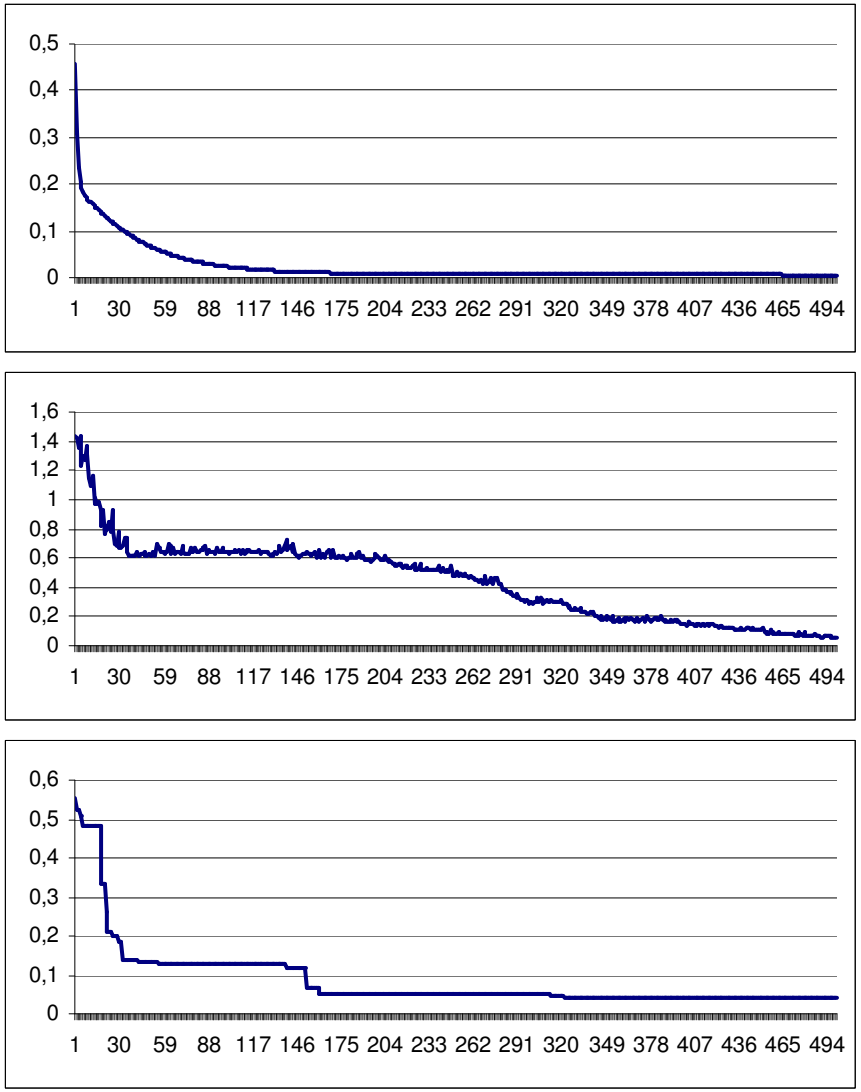


Рис. 3.3. Пример изменения суммарной ошибки в зависимости от эпохи: вверху – задание 1; в середине – задание 3; внизу – задание 4

На рис. 3.3 приведены примеры экспериментально полученных графиков изменения суммарной ошибки для трех алгоритмов обучения двухслойного сигмоида.

Список литературы

1. *Хайкин*, *С*. Нейронные сети: полный курс. – 2-е изд. – М. ; СПб. ; Киев : Вильямс, 2006. – 1104 с.

2. *Струченков*, *В*. *И*. Методы оптимизации. – М. : Солон-Пресс, 2009. – 320 с.

3. *Уоссермен*, *Ф*. Нейрокомпьютерная техника : Теория и практика. – М. : Мир, 1992. – 184 с.

4. *Круглов*, *В*. *В*. Нечеткая логика и искусственные нейросети / В. В. Круглов, М. И. Дли, Р. Ю. Голунов. – М. : Физматлит, 2001. – 224 с.

5. *Рутковская*, *Д*. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М. : Горячая линия – Телеком, 2006. – 452 с.

4. Требования к содержанию и оформлению отчетов

Отчет должен содержать следующие разделы:

1. Задания по работе с указанием номера варианта и данных по варианту.

2. Основные положения теории (2–3 страницы).

3. Алгоритм программы для каждого задания.

4. Текст программы для каждого задания (только программные модули, касающиеся логики работы нейросетей).

5. Результаты работы программы для задания.

6. Выводы по результатам выполнения задания.

7. Список литературы.

Приложение

Форма титульного листа отчета

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Ижевский государственный технический университет имени М. Т. Калашникова»

Кафедра «Программное обеспечение»

Отчет
по самостоятельной работе № 1 на тему
«Персептроны»
по дисциплине
«Нейрокомпьютерные системы»

Выполнил
магистрант группы М03-191-1 Иванов И. И.

Принял Коробейников А. В.

Ижевск
2012

ПРОГРАММИРОВАНИЕ НЕЙРОННЫХ СЕТЕЙ

Учебно-методическое пособие по дисциплинам
«Методы оптимизации. Нейронные сети»,
«Нейрокомпьютерные системы» и
«Нечеткая логика и генетические алгоритмы»

Коробейников Александр Васильевич (составление)