ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

М. Л. Кричевский

ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ В МЕНЕДЖМЕНТЕ

Учебное пособие

Санкт-Петербург
2005

УДК 658.012:681.3.01(075)
ББК 65.290-2
К82

Кричевский М. Л.

К82 Интеллектуальный анализ данных в менеджменте: Учеб. пособие / СПбГУАП. СПб., 2005. 208 с.: ил.
ISBN5-8088-0143-5

В пособии рассмотрены интеллектуальные информационные технологии, включающие нейронные сети, генетические алгоритмы, нечеткую логику. Приведены примеры использования таких технологий в различных задачах менеджмента.

Рецензенты:
кафедра вычислительной техники и информационных технологий Санкт-Петербургского государственного морского технического университета;
кандидат технических наук, доцент *Е. А. Муравьев*

Утверждено
редакционно-издательским советом университета
в качестве учебного пособия

ISBN5-8088-0143-5

Введение

Под интеллектуальными (интеллект – от лат. Intellectus – ум, рассудок, разум) методами подразумеваются такие способы решения задач, в основе которых лежат алгоритмы и действия, в большей или меньшей степени связанные с интеллектуальной деятельностью человека, его эволюцией, повседневным поведением. Термин «интеллектуальные» в области компьютерных технологий можно считать устоявшимся, и сочетание «интеллектуальные информационные технологии» не режет слух и воспринимается специалистами достаточно однозначно. Тем не менее окончательного решения, что же может быть отнесено к этой сфере, еще нет, и каждый, занимающихся такой проблемой, решает ее по-своему.

В данном учебном пособии класс интеллектуальных технологий (ИТ) включает следующие направления:

– искусственные нейронные сети (ИНС);

– генетические алгоритмы (ГА);

– нечеткая логика (НЛ).

Кратко охарактеризуем каждое направление и укажем задачи менеджмента, которые могут решаться с использованием этих методов.

Искусственные нейронные сети состоят из отдельных вычислительных элементов (формальных нейронов), которые в определенной степени подобны биологическим нейронам мозга человека. Характерная особенность ИНС заключается в том, что процесс программирования традиционного пути решения задач заменяется здесь процедурой обучения сетей. Метод обучения ИНС является одним из главных классификационных признаков сетей. Способы объединения нейронов в сеть, количество слоев нейронов, наличие или отсутствие обратных связей определяют архитектуру ИНС. В области менеджмента к проблемам, которые могут быть решены с помощью ИНС, относятся задачи классификации и ранжирования предприятий, фирм, построения рейтингов банков, прогнозирования объема продаж и изменения обменного курса валют.

Генетические алгоритмы представляют собой алгоритмы поиска оптимальных решений, построенные на принципах естественного отбора и генетики. Любое возможное решение изображается в виде строки (хромосомы) фиксированной длины, к популяции которых применяются традиционные генетические операторы: селекция, скрещивание, мутация. Селекция направляет генетический поиск в перспективные районы пространства решений, скрещивание выполня-

ет случайный поиск вблизи локального оптимума, мутация отыскивает улучшенное решение. Определяя в каждом поколении лучшие хромосомы, скрещиваем их между собой для получения потомства, которое представляет близкое к оптимальному решение задачи. К задачам в сфере менеджмента, решаемым посредством ГА, можно отнести составление плана оптимальных перевозок, определение лучшей торговой стратегии, размещение производственных мощностей.

Нечеткая логика и правила, основанные на ее концепции, представляют собой средство моделирования неопределенностей естественных понятий языка. Примерами таких неопределенностей служат лингвистические переменные *холодная, теплая, горячая* применительно к температуре воды. На вход системы с нечеткой логикой поступают четкие переменные, которые преобразуются в нечеткие подмножества. С помощью экспертов создается база правил вида «*если* ... , *то* ...», включаемая в систему. Над входными переменными с помощью базы правил производятся необходимые преобразования, после чего посредством методов дефазификации (перехода от нечетких переменных к четким) на выходе системы формируется четкая выходная переменная. Среди управленческих задач, решаемых с помощью систем НЛ, можно выделить класс проблем риск-менеджмента, где при нечетких входных переменных требуется получить количественную характеристику выходной величины.

Отметим, что есть задачи, решение которых обеспечивается использованием только одного из перечисленных методов, однако наряду с этим намечается тенденция, особенно в последнее время, применения гибридных технологий для решения поставленной проблемы. В каждом из направлений, соответствующих указанным интеллектуальным методам, имеются сложившиеся научные школы, достаточно широкий круг специалистов, издаются книги и журналы (большей частью за рубежом), проводятся международные конференции, симпозиумы. Значительно меньше внимания уделяется изучению совокупности таких методов в какой-либо конкретной области научной и практической деятельности.

ГЛАВА 1. ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Этот раздел знакомит читателей с основами нейросетевой технологии, которая аналогична работе биологического прототипа (человеческого мозга). Из большого количества нейронных архитектур, применяемых на сегодняшний день для решения задач из различных сфер деятельности, отобраны те, которые, по мнению автора, наиболее успешно применяются или могут использоваться в области менеджмента. К таким топологиям относятся многослойные прямонаправленные персептроны и конкурентные сети Кохонена. Именно этим видам сетей уделяется внимание в данной главе. Достаточно подробно рассматриваются принцип действия, алгоритмы обучения, применение в задачах менеджмента указанных типов сетей. Приводятся сведения о пакетах программного обеспечения в области нейросетей и примеры решения задач менеджмента с помощью пакета Statistica Neural Networks.

1.1. Становление нейронной доктрины

Искусственные нейронные сети используются при решении проблем, которые не могут быть точно сформулированы. Слово «нейронные», фигурирующее в названии раздела, применяется потому, что многое в теорию ИНС пришло из нейробиологии, хотя в действительности при изучении дисциплины не рассматриваются сети реальных биологических нейронов. Моделирование функций мозга – совершенно другая научная сфера, но из этой области в теорию ИНС заимствованы некоторые биологические аналогии.

В качестве определения нейронных сетей может служить такое [1]: ИНС – параллельно распределенная структура обработки информации, состоящая из отдельных элементов (нейронов), которые соединены между собой связями.

На сегодняшний день традиционные методы организации вычислительного процесса в ЭВМ оказываются неэффективными при решении ряда задач, относящихся к классам неформализуемых или случайных, которые требуют обработки больших массивов информации, проверки множества альтернативных гипотез, поиска в базах данных. Применение ИНС позволяет привлечь в вычислительную технику метод обработки информации, характерный для высокоорганизованных биологических систем, в частности, принцип параллелизма. В отличие от используемых в традиционных ЭВМ фон-Неймановской архитектуры в ИНС и компьютерах, построенных на их основе, роль программирования выполняет обучение, под кото-

рым понимается изменение состояния самих нейронов и связей между ними.

Мозг человека превосходит компьютер при решении многих задач. Удачный пример сопоставления – это обработка визуальной информации: годовалый ребенок лучше и быстрее распознает предметы, лица и т. п., чем это делает самая совершенная система искусственного интеллекта. Только в одной задаче компьютер превосходит возможности мозга – при выполнении арифметических операций.

В то же время мозг обладает рядом характерных свойств, которые могут быть привлечены для искусственных систем:

– устойчив к повреждениям, нервные клетки в мозге умирают каждый день без какого-либо ощутимого влияния на функционирование нервной системы;

– гибок, легко может настраиваться на новое окружение посредством «обучения» (не требует программирования на каком-либо стандартном языке);

– может иметь дело с информацией, являющейся вероятностной, нечеткой, искаженной помехами;

– обладает высокой степенью параллелизма;

– мал, компактен и потребляет мало энергии.

Эти свойства мозга являются реальной мотивацией для изучения ИНС, которые служат альтернативой традиционной вычислительной парадигме, введенной фон-Нейманом и используемой сегодня практически во всех компьютерах.

Искуственные нейронные сети называют также искусственными нейронными системами, параллельно распределенными системами, коннекционистскими моделями, нейрокомпьютерами. Такие системы являются попыткой имитировать, по крайней мере, частично структуру, функции мозга и нервной системы живых существ. Одна из привлекательных особенностей ИНС заключается в их способности к адаптации к внешним условиям с помощью изменения связей или структуры. ИНС иногда рассматриваются как значительно упрощенные модели человеческого мозга. Такой взгляд несколько преувеличен и вводит в заблуждение, так как мозг человека еще не полностью изучен и его поведение очень сложное.

Мозг содержит порядка 10^{11} нейронов (нервных клеток) различных типов. На рис. 1.1 показано схематическое изображение одного нейрона. Ветвящиеся сети нервных волокон, называемые дендритами, соединяются на теле клетки (соме), где расположено ее ядро. Из тела клетки выходит аксон, представляющий собой нервное волокно, который разветвляется на множество прядей. Он может быть как

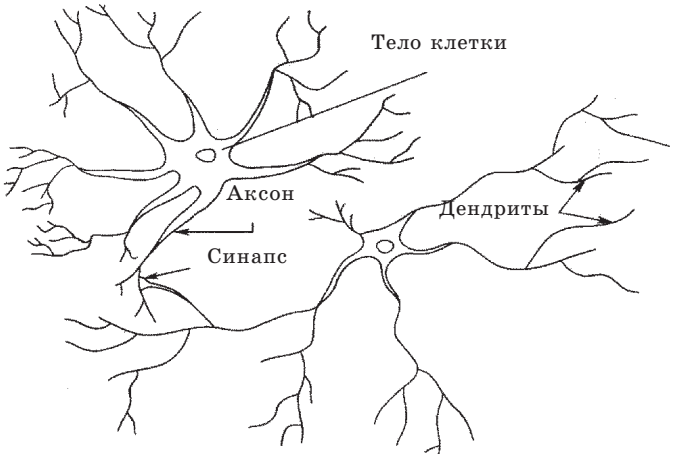


Рис. 1.1

коротким (0,1 мм), так и длинным (свыше 1м), распространяясь в другую часть тела человека. На концах своих ветвей аксон имеет передающие окончания синаптических соединений (синапсов), через которые сигнал передается в другие нейроны через дендриты, а в некоторых случаях – прямо в тело клетки. Существует огромное количество синаптических связей от аксона к аксону, от аксона к телу клетки, от дендрита к дендриту.

В отличие от электрических цепей синаптические контакты не являются физическими или электрическими соединениями. Вместо этого имеется узкое пространство, называемое синаптической щелью, отделяющее дендрит от передающего аксона. Специальные химические вещества (нейротрансмиттеры), выбрасываемые аксоном в синаптическую щель, диффундируют к дендриту, а затем улавливаются рецепторами на дендрите и внедряются в тело клетки. Определено более 30 видов нейротрансмиттеров. Некоторые из них являются возбуждающими и стремятся вызвать активацию клетки и выработать выходной импульс, другие – тормозящими и стремятся подавить такой импульс. Тело клетки суммирует сигналы, полученные от дендритов, и, в случае превышения результирующего сигнала над некоторым порогом, вырабатывает импульс, проходящий по аксону к другим нейронам. Каждый нейрон может генерировать импульс, который возбуждает или затормаживает сотни или тысячи других нейронов; последние через свои дендриты воздействуют на другие нейроны. Эта высокая степень связности обеспечивает мозгу вычислительную мощность.

Дадим краткий исторический обзор становления нейронной доктрины.

Начало эпохи нейронных сетей. В 1943 г. В. Маккаллох и В. Питтс (W. McCulloch, W. Pitts) [2] предложили общую теорию информационных вычислений, основанную на бинарных решающих элементах, которые назвали «нейронами». Каждый из этих элементов $i = 1,\ldots, n$ может принимать выходные значения только $n_i = 0; 1$, при этом $n_i = 0$ определяет состояние покоя, а $n_i = 1$ – активное состояние элементарной ячейки. Положим, что изменения состояния сети имеют место в дискретные моменты времени $t = 0, 1, 2,\ldots$. Новое состояние некоторого нейрона определяется воздействием всех других на него нейронов и выражается линейной комбинацией их выходных значений:

$$h_i(t) = \sum_j w_{ij} n_j(t). \tag{1.1}$$

Здесь w_{ij} определяет синаптические веса между нейронами i и j, а $h_i(t)$ – общий постсинаптический потенциал на нейроне i, обусловленный действием всех других нейронов. Иначе говоря, модель нейрона Маккаллоха–Питтса вычисляет взвешенную сумму своих входов от других нейронов. Далее в модели принято, что нейрон становится активным, если эта сумма (вход нейрона) превышает некоторый порог θ_i, который может значительно различаться от одного нейрона к другому. Тогда эволюция сети определяется следующим выражением:

$$n_i(t+1) = f[h_i(t) - \theta_i], \tag{1.2}$$

где $f(x)$ – единичная ступенчатая функция, равная

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

Вес w_{ij}, определяющий величину связи между нейронами i и j, может быть положительным или отрицательным, что соответствует возбуждающему или замедляющему синапсу. При нулевом значении веса связь между нейронами i и j отсутствует.

В. Маккаллох и В. Питтс показали, что сети, составленные из таких нейронов, могут выполнять вычисления подобно программируемым цифровым компьютерам. В определенном смысле сеть также содержит «программный код», который управляет вычислительным процессом, т. е. матрицей весов w_{ij}. Сеть отличается от традиционного компьютера тем, что шаги программы выполняются не последовательно, а параллельно в пределах каждой элементарной ячейки.

Однако реальные нейроны имеют ряд специфических признаков, отличающих их от упрощенной модели Маккаллоха–Питтса:

– часто не являются такими пороговыми устройствами, как описано выше. Вместо этого нейроны откликаются на воздействие входного сигнала непрерывным способом (иногда это называют градуальным откликом). Но нелинейное соотношение между входом и выходом нейрона имеет место и в биологическом прототипе;

– вырабатывают последовательность импульсов, а не простую выходную величину. Представляя активацию нейрона единственным числом n_i, игнорируется та информация, которая может содержаться в последовательности импульсов (фазовые соотношения). Однако большинство специалистов полагает, что фаза не играет значительной роли в реальных нейронных сетях;

– не все нейроны имеют одно и то же время задержки ($t \to t + 1$). Также более правдоподобной является гипотеза об асинхронном изменении состояния нейронов;

– количество нейротрансмиттерного вещества, поступающего в синаптическую щель, может меняться непредсказуемым образом. Отчасти это можно учесть, придавая динамике нейрона Маккаллоха–Питтса стохастический характер.

В целом, учитывая некоторые из вышеприведенных свойств реальных нейронов, в модель Маккаллоха–Питтса следует ввести вместо пороговой функции более общую нелинейную функцию $f(x)$, которая называется функцией активации, или передаточной функцией. На рис. 1.2 приведена схема искусственного нейрона Маккаллоха–Питтса.

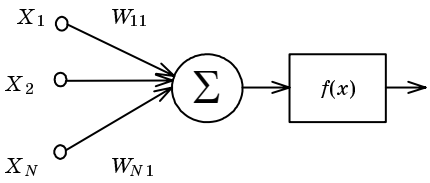


Рис. 1.2

На первом этапе становления нейронной доктрины еще одним из основателей по праву считается канадский ученый-нейропсихолог Д. Хебб (D. Hebb). В 1949 г. вышла его книга «Организация поведения», где он изложил теорию нейронных ансамблей и сформулировал используемый в этой теории нейрофизиологический постулат, который считается первым законом обучения для ИНС. Смысл этого закона заключается в том, что связь между двумя нейронами возрастает, если оба этих нейрона одновременно являются активными.

Второй этап (50-е–60-е гг.). Его можно назвать *первым золотым веком* нейронных сетей. В этот период Ф. Розенблатт (F. Rosenblatt) с коллегами интенсивно изучали специальный тип ИНС, названный ими п е р с е п т р о н , так как они рассматривали его как упрощенную модель биологического механизма определения сенсорной информации [3]. В самой простейшей форме персептрон состоит из двух различных слоев нейронов, представляющих собой входной и выходной слои, соответственно. Отметим, что даже простой персептрон с технической точки зрения является трехслойным устройством, так как слой сенсорных ячеек расположен перед первым слоем вычислительных нейронов, который здесь называется входным. Нейроны выходного слоя принимают синаптические сигналы от нейронов входного слоя, но не наоборот, и нейроны в пределах одного слоя не соединяются между собой. Таким образом, поток информации является строго направленным, а сети – прямонаправленными. Ф. Розенблатт разработал итеративный алгоритм для определения весов w_{ij}, с помощью которого конкретный входной образ трансформируется в требуемый выходной образ, и доказал сходимость этого алгоритма. Научные круги с энтузиазмом восприняли его работы, так как полагали, что такие устройства могут послужить основой для создания искусственного интеллекта.

Третий этап развития концепции ИНС начался в конце 60-х гг. и получил название *застойного периода*. В 1969 г. М. Минский (M. Minsky) и С. Паперт (S. Papert) в своей книге, посвященной персептронам, доказали, что существует ряд простых задач, которые, в принципе, не могут быть решены с помощью простого персептрона [4]. Одна из них – проблема «исключающего ИЛИ», которая не может быть представлена простым персептроном. Ф. Розенблатт разработал структуры многослойных персептронов, но отсутствие обучающего алгоритма, т. е. правила изменения весов, в то время не позволило ему показать возможности таких сетей. Научный авторитет авторов книги [4], в первую очередь М. Минского, в кругах специалистов был достаточно высок, и это послужило причиной того, что многие ученые, разочаровавшись в нейросетевой теории, переключились на другие области знаний. На протяжении почти полутора десятилетий проблемами ИНС продолжали заниматься лишь немногие ученые, верившие в перспективность этого направления.

Среди таких ученых надо, прежде всего, отметить С. Гроссберга (S. Grossberg) из Бостонского университета, который в период 1967–1988 гг. опубликовал почти 150 работ в области ИНС, и Т. Кохонена

(T. Kohonen) из Хельсинского университета технологии – основоположника самоорганизующихся сетей. Усилиями этих и других ученых в этот период застоя продолжалась разработка теоретических основ нейрокомпьютинга.

Следующий шаг в развитии теории нейронных сетей произошел в конце 70-х гг., когда В. Литтл (W. Little) выявил сходство между нейронами Маккаллоха–Питтса и системами элементарных магнитных моментов, называемых спинами. В таких системах спин s_i в каждом узле решетки может иметь только два различных направления: вверх или вниз, обозначаемые, соответственно, как $s_i = +1$ и $s_i = -1$. Аналогия с нейронной сетью следует из идентификации каждого спина с нейроном и ассоциированием ориентации спина вверх $s_i = +1$ с активным состоянием нейрона $n_i = 1$ и ориентации спина вниз $s_i = -1$ с состоянием покоя.

Эти идеи были развиты далее Д. Хопфилдом (D. Hopfield) [5], который изучил, как подобные сети или спиновые системы могут запоминать и восстанавливать информацию. Модели В. Литтла и Д. Хопфилда различаются способом, по которому изменяется состояние системы. В модели В. Литтла все нейроны (спины) изменяются синхронно в соответствии с законом (1), в то время как в модели Д. Хопфилда нейроны изменяются последовательно во времени (или в некотором фиксированном порядке, или случайным образом). Последовательное изменение имеет значительное преимущество при имитации сети на традиционном цифровом компьютере, а также для теоретического анализа свойств сети. Однако такой режим имеет существенный недостаток: в реальных нейронных сетях одновременно участвует большое число параллельных ячеек.

Аналогия со спиновыми системами становится особенно плодотворной благодаря прогрессу в понимании термодинамических свойств неупорядоченных систем спинов (спиновых стекол), достигнутому в последние годы. Для того чтобы применить эти результаты к нейронным сетям, необходимо заменить детерминированный закон эволюции (1.2) стохастическим законом, где величине $n_i(t + 1)$ присваиваются значения в соответствии с вероятностной функцией, зависящей от интенсивности входа h_i. Эта функция содержит параметр T, который играет роль температуры. Однако T здесь не является моделью физической температуры биологической сети, но служит формальной концепцией для введения элементов случайности в сеть и применения методов статистической термодинамики.

Четвертый этап (середина 80-х гг.) – последний этап развития ИНС, которой получил название *возрожденного энтузиазма*. Это

развитие было инициировано разработкой эффективного алгоритма для определения весов в многослойных сетях со скрытыми слоями. Данный метод, первоначально предложенный в 1974 г. П. Вербозом (P. Werbos) в своей докторской диссертации, но незамеченный в среде специалистов, в 1985 г. был изучен несколькими группами исследователей [6] и получил название метода обратного распространения ошибки (Back Propagation Error – BPE). Этот обучающий алгоритм основан на простом, но очень эффективном принципе: веса w_{ij} изменяются итеративно таким образом, что выходной сигнал отличается от требуемого выхода настолько мало, насколько это возможно. Такое решение достигается использованием градиентного метода, который дает требуемые изменения весов. Так как действие сети основано на нелинейном отображении между входом и выходом, то метод BPE должен быть применен многократно до тех пор, пока не будет достигнута сходимость. Появление этого метода, позволяющего решать сложные задачи с помощью многослойных персептронов, оказало существенное влияние на пути дальнейшего развития теории ИНС.

1.2. Парадигмы обучения

Возможно, самым удобным принципом классификации искусственных нейронных сетей являются парадигмы их обучения (иначе, правила изменения и регулирования весов). Существуют три основных парадигмы обучения:

– супервизорное обучение (СО);

– несупервизорное обучение (НСО);

– усиленное обучение (УО).

Супервизорное обучение – наиболее часто используемый вид обучения сетей и применяется в ИНС, предназначенных для классификации и предсказания.

Несупервизорное обучение применяется в задачах кластеризации и сегментации для поддержки принимаемого решения.

Усиленное обучение находит применение в задачах оптимизации и адаптивного управления и по сравнению с другими способами обучения используется реже.

Рассмотрим более подробно указанные виды обучения.

Супервизорное обучение

Этот вид обучения эквивалентен программированию на примерах. При таком подходе сети задается проблема, и ИНС ищет решение по известному соотношению «вход-выход». Здесь «учитель»

(тренер) указывает, каким должен быть правильный ответ. Обучающий алгоритм по разнице между правильным (требуемым) выходом и действительным выходом сети регулирует ее веса таким образом, что в следующий момент времени (на следующем проходе) выход сети будет ближе к правильному ответу. На входной слой сети подается вектор входных параметров, а на выходной – соответствующий этому вектору номер класса. Схема СО приведена на рис. 1.3. Такие примеры «вход-выход» сети должны быть предъявлены десятки, сотни и даже тысячи раз прежде, чем сеть может дать точный ответ на некоторую сложную проблему (например, отнести новый, неизвестный сети объект к одному из классов, на которые она обучена).

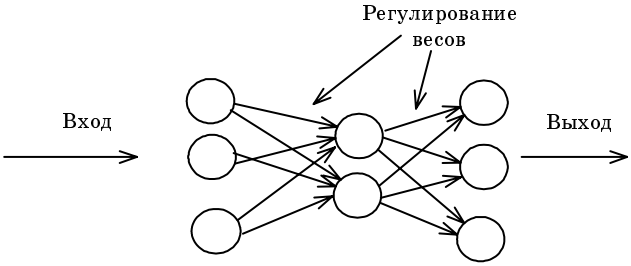


Рис. 1.3

Супервизорное обучение обычно используется, когда в наличии имеется база данных примеров, содержащих вход и выход. По этой базе ИНС может обучиться тому, как связаны вход и выход, и на основании этого принять соответствующее решение. Сеть может обучиться на сотнях (тысячах) примеров, выполненных лучшими исполнителями фирмы, и это только тренируясь на предъявляемых ей «учителем» примерах. Цель такой тренировки заключается в минимизации ошибки между правильным и реальным выходами сети.

Супервизорное обучение является полезным подходом для обучения ИНС выполнять классификацию, аппроксимацию функций, прогнозирование. Это особенно полезно в задачах, где данные доступны в форме пар «вход-выход», но неизвестно точное преобразование для обработки входного вектора с целью получения выхода. Такая ситуация имеет место для данных, полученных статистическим путем, в задачах, которые являются нелинейными и имеют сложную связь между переменными.

Несупервизорное обучение

Несупервизорное обучение используется в ситуациях, когда имеется много данных о некоторых объектах и представляет интерес оценка сходства объектов из рассматриваемой совокупности. Перед сетью здесь ставится задача кластеризации входных данных таким способом, чтобы похожие образы (объекты) попали в одну группу (кластер). ИНС, использующая НСО, может решать такую задачу с большой точностью.

Обычно сеть с таким способом обучения состоит только из входного и выходного слоев, причем количество нейронов в первом слое определяется размерностью входного вектора, а во втором – числом предполагаемых классов, на которые желательно разбить исходную совокупность. Схема НСО приведена на рис. 1.4. Ячейки

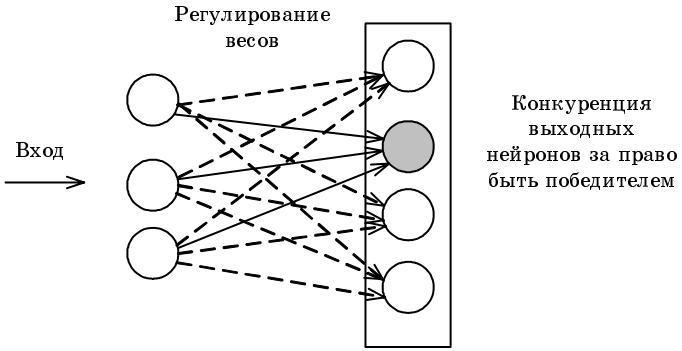


Рис. 1.4

выходного слоя соревнуются между собой за право быть победителем, которым в итоге становится нейрон выходного слоя с весами, наиболее близкими к компонентам входного вектора. В процессе обучения происходит подстройка весов, в первую очередь нейрона-победителя, с тем чтобы стать еще ближе к входному вектору. Здесь ячейка-победитель является, по существу, меткой класса, к которому отнесен входной вектор. ИНС, которые обучаются таким методом, называются *самоорганизующимися*, потому что они не получают указаний относительно требуемого выхода. При предъявлении входного образа сеть посредством конкуренции и регулирования весов относит образ к одному из существующих уже кластеров или к новому классу.

Усиленное обучение

В сетях с супервизорным обучением предполагалось, что имеется информация о правильных выходных значениях для каждого входного образа. Но в некоторых ситуациях доступна менее детальная информация. В предельном случае может быть только один бит информации: выход правильный или неверный. В таких условиях используется процедура УО.

Последнее является формой СО, так как сеть получает некоторую обратную связь из окружающей среды. Но эта обратная связь (сигнал усиления, да/нет) служит лишь оценкой, но не инструкцией к поведению сети. УО иногда называют обучением с критиком в противоположность обучению с учителем.

При использовании парадигмы УО полагают работу ИНС в определенной внешней среде. Среда формирует входы в сеть, получает выходы сети и затем определяет сигнал усиления r (рис. 1.5). Здесь есть несколько проблем, зависящих от характера окружающей среды.

1. В самом простейшем случае сигнал усиления r всегда одинаков для данной пары вход-выход. Таким образом, в этом случае имеется конкретное отображение вход-выход, которому сеть должна обучиться, или несколько таких отображений при наличии многих выходов для одного входа. Кроме того, входные образы выбираются в случайном порядке внешней средой без учета предыдущих выходов. Эта ситуация отчасти напоминает случай СО.

2. Расширение предыдущего случая – это стохастическое окружение. Здесь отдельная пара вход-выход определяет только вероятность положительного усиления. Эта вероятность является фиксированной для каждой пары вход-выход, а входная последовательность, как и в предыдущем случае, не зависит от прошлой истории. Подоб-

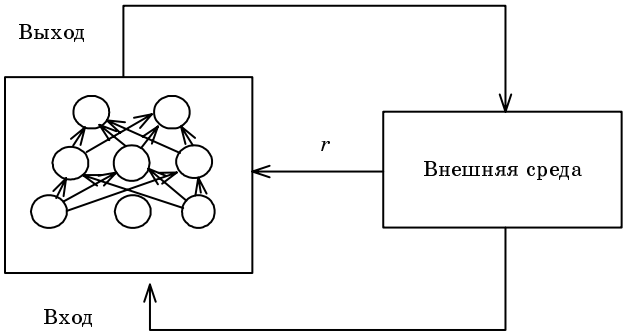


Рис. 1.5

ные проблемы часто возникают в моделировании поведения животных, экономических системах, некоторых простых играх.

3. В самом общем случае внешняя среда может сама управляться сложным динамическим процессом. Здесь сигналы усиления и входные образы могут зависеть произвольным образом от прошлой истории сетевого выхода. Классический пример – это теория игр, где природа является еще одним игроком. Например, ИНС для игры в шахматы: сеть принимает сигнал усиления (выигрыш или проигрыш) только после долгой последовательности ходов. Анализируя партию, мы можем найти, где были правы или неправы (ситуация, очень напоминающая многие жизненные ситуации). В некоторых отношениях УО – самый похожий на житейские проблемы вид обучения.

1.3. Нейросетевые топологии

Организация нейронов и их связей в определенную структуру (архитектуру) оказывает значительное влияние на вычислительные возможности ИНС. Все сети имеют некоторое количество вычислительных элементов (формальных нейронов), принимающих сигналы из внешней среды. Такие нейроны называются входными. Многие ИНС обладают одним или несколькими слоями «скрытых» вычислительных элементов, которые принимают сигналы от других нейронов. Этот слой принимает вектор входных данных или выходы предыдущего слоя и обрабатывает их параллельно. Нейроны, которые представляют окончательный результат нейросетевых вычислений, определяются как выходные. Топологии связей, которые определяют поток данных между входными, скрытыми и выходными нейронами, сводятся к двум основным группам: прямонаправленные (слоистые) и рекуррентные (полносвязные) сети. Перед рассмотрением этих базовых топологий укажем на согласованность работы различных нейронов во времени. Здесь и далее рассматриваются только нейронные сети, синхронно функционирующие в дискретные моменты времени (все нейроны срабатывают одновременно).

В слоистых сетях нейроны расположены в несколько слоев (рис. 1.6). Нейроны первого слоя получают входные сигналы, преобразуют их и передают нейронам второго слоя. Далее срабатывает второй слой и т. д. до k-го слоя, который выдает выходные сигналы для интерпретатора и пользователя. Если не оговорено противное, то каждый выходной сигнал предыдущего слоя подается на вход всех нейронов последующего. Число нейронов в каждом слое может быть любым и никак заранее не связано с количеством нейронов в других слоях. Стандартный способ подачи входных сигналов: все нейроны первого слоя

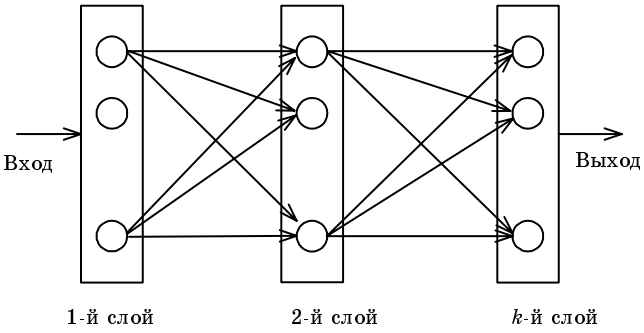


Рис. 1.6

получают каждый входной сигнал, поэтому чаще всего количество нейронов входного слоя определяется размерностью вектора входных данных.

В полносвязных сетях каждый нейрон передает свой выходной сигнал остальным нейронам, включая самого себя. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети. Входные сигналы подаются всем нейронам.

Для полносвязной сети входной сумматор нейрона фактически распадается на два: первый вычисляет линейную функцию от входных сигналов сети, второй – линейную функцию от выходных сигналов других нейронов, полученных на предыдущем шаге.

На рис. 1.7 показан пример сети рекуррентного типа.

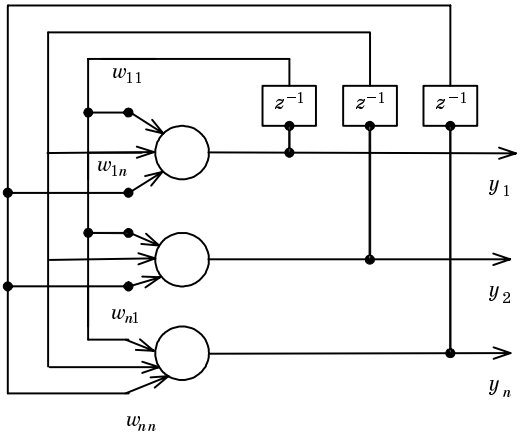


Рис. 1.7

Важную роль в искусственных нейронных сетей выполняет функция активации, под воздействием которой искусственный нейрон преобразует взвешенную сумму входных сигналов и вырабатывает выходной сигнал. Такая операция является основной. Обычно одна и та же функция активации используется во всех нейронах данного слоя сети, хотя иногда могут быть исключения из этого правила. В большинстве случаев применяется нелинейная функция активации, в качестве которой в нейросетевой технологии распространены так называемая сигмоидная функция и функция гиперболического тангенса. Применение таких функций обусловлено, в основном, двумя обстоятельствами: ограничением величины сигнала после операций умножения и суммирования и простыми соотношениями между этими функциями и их первыми производными, которые используются в методе обучения многослойных сетей.

Сигмоидная функция и ее производная имеют следующий вид:

$$f(x) = [1 + \exp(-\beta x)]^{-1},$$
$$f'(x) = \beta f(x)[1 - f(x)].$$

Входящий в формулу параметр β, величина которого влияет на форму кривой, подбирается пользователем.

Значения этой функции лежат в диапазоне от 0 до 1, поэтому ее часто применяют в ситуациях, когда требуемые выходные значения являются бинарными или находятся в указанном диапазоне. Иногда такую функцию называют бинарной сигмоидой. На рис. 1.8 изображен график сигмоидной функции при различных значениях β (кривая *1* – β =0,2; кривая *2* – β = 2; кривая *3* – β = 10).

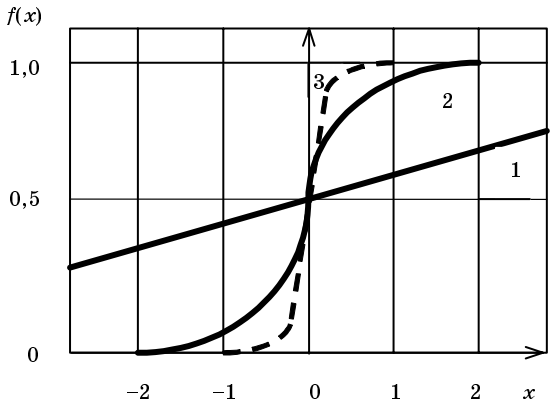


Рис. 1.8

При необходимости иметь значения выходных сигналов в диапазоне от −1 до 1 используется биполярная сигмоида в виде:

$$f(x) = [1 - \exp(-x)]/[1 + \exp(-x)],$$
$$f'(x) = [1 + f(x)][1 - f(x)].$$

Биполярная сигмоида очень близка к функции гиперболического тангенса:

$$th(x) = [1 - \exp(-2x)]/[1 + \exp(-2x)],$$
$$th'(x) = [1 + th(x)][1 - th(x)].$$

Существуют, кроме указанных, и другие виды функций активации, используемые в тех или иных ситуациях в зависимости от конкретных задач.

1.4. Алгоритмы обучения

Алгоритм обучения означает процедуру, в которой используются правила обучения для настройки весов. Известны четыре основных типа правил обучения:

– правило Хебба;
– коррекция по ошибке;
– метод конкуренции;
– машина Больцмана.

Правило Хебба является самым старым обучающим правилом и представляет собой постулат Хебба, о котором упоминалось выше. Подчеркнем еще раз смысл этого правила: если нейроны с обеих сторон синапса активизируются одновременно и регулярно, то сила синаптической связи возрастает. Важная особенность этого правила состоит в том, что изменение синаптического веса зависит только от активности нейронов, которые связаны данным синапсом.

Коррекция по ошибке используется в сетях СО. Для каждого входного примера задается требуемый выход t. Реальный выход сети y может не совпадать с требуемым, откуда следует принцип коррекции ошибки при обучении: для модификации весов, обеспечивающей постепенное уменьшение ошибки, используется сигнал $(t - y)$. Обучение проводится только в том случае, если сеть на выходе выдает сигнал, отличный от требуемого.

При обучении *методом конкуренции* нейроны выходного слоя соревнуются между собой за право стать активным. Борьба происходит под девизом: «Победитель получает все» (в английском написании – «Winner Takes All» (WTA)), в результате только нейрон-победитель (тот нейрон, весовой вектор которого ближе всех к входному вектору)

получает право на изменение своих весовых коэффициентов. Обучение посредством конкуренции позволяет кластеризовать входные данные: сходные образы группируются сетью в соответствии с корреляциями и представляются одним элементом (меткой кластера).

Машина Больцмана представляет собой стохастическое правило обучения, которое следует из информационных теоретических и термодинамических принципов. При адаптпции производится настройка весовых коэффициентов, при которой состояния нейронов удовлетворяют желаемому распределению вероятностей (в частности, распределению Больцмана). Обучение Больцмана может рассматриваться как специальный случай коррекции по ошибке, в котором под ошибкой понимается расхождение корреляций состояния в двух режимах.

Ниже подробно рассматриваются три первых алгоритма как наиболее употребительные в нейросетевой технологии.

1.5. Простые однослойные сети

Самой простой задачей, которую могут выполнять однослойные нейронные сети, является классификация образов. При решении классификационных проблем обученная ИНС должна отнести предъявленный образ (входной вектор) к одному из классов, распознавать которые она была обучена. Перед тем как перейти к описанию однослойных сетей, целесообразно рассмотреть вопрос о линейной разделимости входных образов, так как это тесно связано с потенциальными возможностями таких сетей.

Линейная разделимость

Процесс обучения сетей в задачах классификации сводится к адаптивному изменению весов таким образом, чтобы сеть могла отнести новый предъявляемый вектор к одному из обученных классов. В простейшем случае можно оценивать принадлежность образа к единственному классу. Тогда на выходе сети должен быть отклик, равный 1, если образ принадлежит к этому классу, или –1 (0 в бинарном представлении) в противном случае. В такой ситуации в качестве функции активации применим ступенчатую функцию вида:

$$\text{sign}(x) = \begin{cases} 1, \text{ если } x \geq 0, \\ 0, \text{ если } x < 0. \end{cases} \qquad (1.3)$$

В нейронную сеть иногда вводят смещение, которое действует как весовой коэффициент от ячейки с активацией, равной 1. Смещение увеличивает входное воздействие в сеть на единицу. При наличии смещения на выходную ячейку поступает сигнал:

$$h = b + \sum_i x_i w_i,$$

где x_i – входной сигнал; w_i – его вес; b – смещение.

При воздействии на выходную ячейку такого сигнала решающая граница между областью, где $h > 0$, и областью, где $h < 0$, определяется уравнением:

$$b + \sum_i x_i w_i = 0.$$

В зависимости от числа входных ячеек в сети это уравнение представляет линию, плоскость или гиперплоскость.

Вместо смещения в ряде случаев применяется фиксированный порог θ для функции активации, которая тогда в отличие от вида (3) записывается как:

$$\text{sign}(h) = \begin{cases} 1, \text{ если } h \geq \theta, \\ 0, \text{ если } h < \theta. \end{cases}$$

Если веса (и смещение) таковы, что все из обучающих входных векторов, для которых правильный отклик есть $+1$, лежат по одну сторону решающей границы, а все из обучающих векторов, для которых правильный отклик есть -1, лежат по другую сторону решающей границы, то данная ситуация приводит к линейной разделимости. Для наглядности дальнейшего анализа ограничимся двумя входными нейронами, что дает для решающей границы следующее выражение:

$$x_2 = -(w_1 / w_2)x_1 - b / w_2,$$

а полученные две области называются областями решений для сети. Выбор знака для b определяет, какая из сторон разделяющей границы соответствует отклику $+1$, а какая -1. Рассмотрим несколько иллюстративных примеров для простой однослойной сети, состоящей из двух входных и одного выходного нейронов, чтобы ответить на вопрос: существуют ли такие веса, при которых сеть будет иметь требуемый выход для каждого из обучающих входных векторов?

Пример 1.1. Области отклика для функции логического И (рис. 1.9).

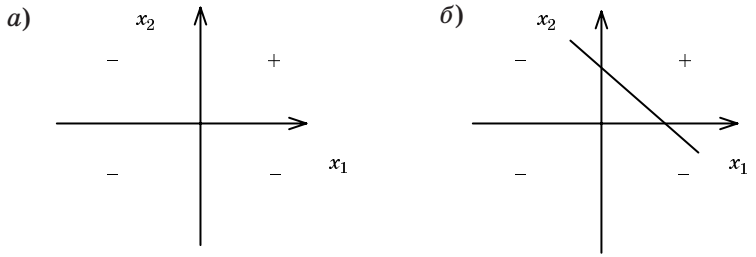


Рис. 1.9

Логическое И для биполярных входов и выхода определяется следующим образом:

Вход	Выход
(1, 1)	+1
(1, −1)	−1
(−1, 1)	−1
(−1, − 1)	−1

Требуемые отклики показаны на рис. 1.9, *а*, где +1 и −1 заменены знаками "+" и "–". Одна из возможных решающих границ приведена на рис. 1.9, *б*, которая описывается следующим уравнением:

$$x_2 = -x_1 + 1$$

со значениями весов и смещения, равных $b = -1$, $w_1 = w_2 = 1$.

Пример 1.2. Области отклика для функции логического ИЛИ (рис. 1.10).

Эта функция для биполярных входов и выхода имеет следующий вид:

Вход	Выход
(1, 1)	+1
(1, −1)	+1
(− 1, 1)	+1
(−1, −1)	−1

Веса должны быть выбраны так, чтобы обеспечить разделение указанных точек, как показано на рис. 1.10. Одно из возможных решений выбора весов есть: $b = w_1 = w_2$, которое дает решающую границу в виде:

$$x_2 = -x_1 - 1.$$

Эти два примера иллюстрируют концепцию линейно разделимого входа. Входные точки, которые должны быть классифицированы положительно, могут быть отделены от входных точек, которые должны быть оценены отрицательно, прямой линией. Полученные уравнения решающих границ не являются единственными. Отметим, что при отсутствии веса смещения в приведенных примерах решающая граница должна проходить через начало координат.

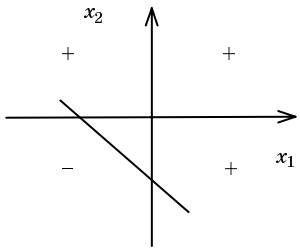


Рис. 1.10

Однако однослойная сеть, состоящая их двух входных и одного выходного нейронов, не может решить любые по-

добные задачи. Таким примером может служить проблема исключающего ИЛИ.

В заключение этого параграфа отметим разницу между представлением входных данных в биполярной (1 и −1) и бинарной (1 и 0) формах. Многие ранние модели ИНС использовали бинарное представление, хотя в большинстве случаев оно может быть трансформировано в биполярное следующим образом:

$$S_i = 2n_i - 1,$$

где $S_i = 1$ или -1 – биполярное; $n_i = 1$ или 0 – бинарное преставления.

Сеть Хебба

Правило Хебба известно как самое раннее и наиболее простое для обучения сетей. В оригинальном правиле Хебба речь идет только о нейронах, находящихся в активном (возбужденном) состоянии, и ничего не говорится о других нейронах, которые не являются активными в это же время. Более сильная форма правила обучения распространяется и на оставшиеся нейроны: вес между двумя нейронами должен быть увеличен, если оба они находятся в пассивном (невозбужденном) состоянии в одно и то же время. Исследования показали, что в такой формулировке правило Хебба обладает улучшенными вычислительными возможностями

Назовем однослойные нейронные сети, использующие такое расширенное правило, сетями Хебба. Архитектура такой сети чрезвычайно проста: один из нейронов сети служит ее входом, один из выходных – выходом. Здесь нет входных нейронов, соединенных друг с другом; то же самое относится и к нейронам выходного слоя. В случае, если данные представлены в биполярной форме, изменение веса при обучении определяется следующим выражением:

$$w_i(new) = w_i(old) + x_i y \ , \qquad (1.4)$$

где $w_i(new)$, $w_i(old)$ – новое и старое значения весов; x_i – значение i-го входного нейрона; y – значение выходного нейрона.

Если данные представлены в бинарном виде, то формула (1.4) не делает различия между обучающими парами, в одной из которых входной нейрон находится в активном состоянии и требуемое значение выхода такое же, и другой парой, в которой как входной нейрон, так и требуемое значение находятся в пассивных состояниях.

Алгоритм обучения сети Хебба представляет собой следующую последовательность шагов.

1. Инициировать все веса: $w_i = 0$, $i = 1, 2, \ldots, n$.

2. Для каждого входного обучающего вектора и требуемого выхода (s; t) делать шаги 3 – 5.

3. Установить активации для входных ячеек: $x_i = s_i$.

4. Установить активацию для выходной ячейки: $y = t$.

5. Отрегулировать веса по соотношению: $w_i(new) = w_i(old) + x_i y$, $j = 1, 2, \ldots, n$.

6. Отрегулировать смещение: $b(new) = b(old) + y$.

Отметим, что смещение регулируется аналогично весу от ячейки, чей выходной сигнал всегда равен 1. Изменение весов может быть также выражено в векторной форме следующим образом:

$$w(new) = w(old) + xy.$$

Последнее выражение можно записать через изменение веса Δw как

$$\Delta w = xy.$$

Простой персептрон

Понятие персептрона впервые было введено Ф. Розенблаттом [3], который описал несколько типов таких устройств и разработал алгоритм обучения персептрона. Его оригинальный персептрон имел три слоя нейронов: сенсорные (приемные), ассоциированные и выходные ячейки, которые в совокупности формировали искусственную модель сетчатки. Ячейки сенсорного слоя соединялись с ячейками ассоциированного слоя связями с фиксированными весами, имеющими значения +1, 0 или −1, которые рассматривались как случайные. При введении входного образа часть ячеек ассоциированного слоя, чьи пороги превышались суммарным входом ячеек сенсорного входа, становилась активной, другая – неактивной. Активные ячейки ассоциированного слоя передавали сигнал на выходной слой. Однако i-я ячейка выходного слоя посылает тормозящие сигналы обратно к каждой j-й ячейке ассоциированного слоя ($i \neq j$). В результате устройство совершает серию итераций, заставляя ячейки ассоциированного слоя быть активными или неактивными до тех пор, пока система не станет стабильной. Схема персептрона Ф. Розенблатта приведена на рис. 1.11.

Термин «персептрон» имеет неоднозначное толкование. Иногда его используют для определения сети со многими ячейками в слоях, как описано выше. Однако в большинстве определений под персептроном понимается сеть, состоящая из ассоциированного и выходного слоев, а так как ассоциированный слой не выполняет никаких вычислений и не учитывается при подсчете слоев сети, то персептрон представляет собой простую однослойную сеть.

Функция активации в таком персептроне является бинарной ступенчатой функцией с произвольным, но фиксированным порогом. При

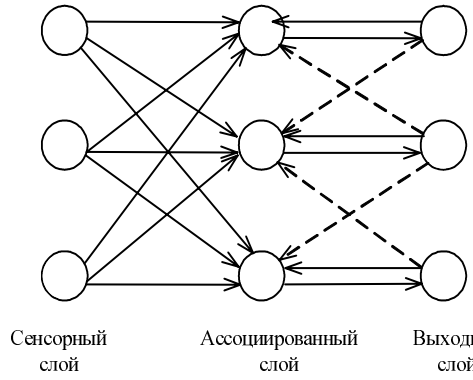


Рис. 1.11

этом сигнал, поступающий на выходную ячейку, есть бинарный (0 или 1) сигнал. Выход персептрона определяется как:

$$y = f(x),$$

где $f(x)$ – функция активации, равная

$$f(h) = \begin{cases} 1, & \text{если } h > \theta, \\ 0, & \text{если } -\theta \le h \le \theta, \\ -1, & \text{если } h < -\theta. \end{cases}$$

Веса между ассоциированным и выходным слоями вычисляются с использованием обучающего правила персептрона. Для каждого обучающего входа сеть должна найти отклик выходной ячейки. Затем сеть определяет наличие ошибки для этого образа путем сравнения вычисленного выхода с требуемой (целевой) величиной. Сеть не делает различия между ошибкой, при которой вычисленный выход 0, а целевое значение равно -1, и ошибкой, при которой вычисленный выход равен $+1$, а целевая величина равна -1. В каждом из этих случаев знак ошибки указывает, что веса должны быть изменены. Однако только те веса изменяются, нейроны которых посылают ненулевой сигнал в ячейки выходного слоя. Если ошибка имеет место для конкретного обучающего входного образа, то изменение весов осуществляется по формуле:

$$w_i(new) = w_i(old) + \eta t x_i,$$

где $t = +1$ или -1 – целевая величина; η – скорость обучения.

В том случае, если ошибки нет, веса не изменяются. Обучение будет продолжаться до тех пор, пока имеется ошибка. Теорема о сходимости обучающего правила персептрона гласит, что при существова-

нии весов, которые позволяют сети правильно откликаться на все обучающие образы, процедура определения весов будет находить такие их значения, при которых сеть должна давать правильный отклик на все предъявленные для обучения образы. Кроме того, сеть будет находить эти значения весов за конечное число шагов.

В первоначальной версии персептрона выходной сигнал от ассоциированного слоя представлял собой бинарный вектор, однако это не является необходимым условием. При рассмотрении алгоритма обучения персептрона регулируемыми весами являются веса между ассоциированным и выходным слоями. Цель сети – классифицировать каждый входной образ как принадлежащий или не принадлежащий определенному классу (категории). Принадлежность определяется значением выходной ячейки, равным +1; если образ не принадлежит этому классу – величина выхода равна –1.

Алгоритм обучения, представленный ниже, подходит как для бинарных, так и для биполярных входных векторов с биполярным целевым выходом, фиксированным порогом и регулируемым смещением. Алгоритм не чувствителен к начальным значениям весов и величине скорости обучения. Обучающий алгоритм простого персептрона, предназначенного для классификации образов, представляет следующую последовательность шагов.

1. Инициировать веса и смещение (для упрощения все веса и смещение примем равными нулю). Выбрать скорость обучения η из диапазона $0 < \eta \le 1$ (положим $\eta = 1$).

2. Если условие остановки не выполняется, делать шаги 3–7.

3. Для каждой обучающей пары $(s; t)$ делать шаги 4–6.

4. Установить активации входных ячеек: $x_i = s_i$.

5. Вычислить отклик выходной ячейки:

$$h = b + \sum_i x_i w_i;$$

$$y = \begin{cases} 1, & \text{если } h > \theta, \\ 0, & \text{если } -\theta \le h \le \theta, \\ -1, & \text{если } h < -\theta. \end{cases}$$

6. Если ошибка имеет место для этого образа, изменить веса и смещение:

$$\text{если } y \ne t, \quad \text{то } w_i(new) = w_i(old) + \eta t x_i,$$
$$b(new) = b(old) + \eta t;$$
$$\text{иначе } w_i(new) = w_i(old); \quad b(new) = b(old).$$

7. Проверить условие остановки: если на шаге 3 нет изменения весов, тогда остановка; в противном случае – продолжить.

Нужно отметить, что изменяются только те веса, которые соединяют активные входные нейроны ($x_i \neq 0$). Кроме того, веса модифицируются только для тех входных образов, которые не формируют правильное значение выхода y. Последнее означает, что увеличение числа обучающих образов, вырабатывающих правильный отклик, приводит к уменьшению времени обучения.

Ф. Розенблаттом была доказана теорема о сходимости обучающего правила персептрона. Позже были даны модифицированные версии доказательства этой теоремы [7], которая формулируется следующим образом.

Дан конечный набор из P обучающих входных векторов $x(p)$, $p = 1, 2, \ldots, P$, каждый из которых связан с целевой величиной $t(p)$, $p = 1, 2, \ldots, P$, принимающей значения $+1$ или -1.

Дана функция активации, имеющая вид, аналогичный приведенным выше примерам.

Тогда изменение весов осуществляется следующим образом:

$$\text{если } y \neq t, \text{ то } w(new) = w(old) + tx;$$

иначе никакого изменения весов не происходит.

Теорема о сходимости обучающего правила персептрона гласит: если имеется весовой вектор w^*, такой, что $f[x(p) \cdot w^*] = t(p)$ для всех p, тогда для любого начального вектора w обучающее правило персептрона будет сходиться к весовому вектору (не обязательно единственному и не обязательно w^*), который дает правильный отклик для всех обучающих образов, и это будет сделано за конечное число шагов.

Доказательство теоремы достаточно громоздко, поэтому ограничимся итоговым результатом. Максимально возможное число изменений весов определяется следующим выражением:

$$k \leq M \left\| w^* \right\|^2 / m^2,$$

где k – число изменений весов; $M = \max \{||x||^2$ для всех x в обучающем наборе$\}$; $||x||$ – норма вектора x; $m = \min \{x \cdot w^*\}$.

Обучение может занимать много времени (большое число шагов), если обучающие векторы обладают малой величиной своей нормы. Это приведет к малому значению m и, как следствие, к большой величине k. Ненулевое значение порога θ не сказывается на доказательстве теоремы, но его величина может изменить решаемую задачу. Кроме того, отличие скорости обучения от 1, как это принято при

доказательстве теоремы, не оказывает существенного влияния на итоговый результат.

1.6. Многослойные нейронные сети

Однослойные нейронные сети не могут выполнять некоторые достаточно простые задачи, к примеру проблему исключающего ИЛИ. Переход к многослойным нейронным сетям, способным решать более сложные задачи, был затруднен из-за отсутствия обучающего алгоритма для таких сетей. Только в 1974 г. П. Вербозом в докторской диссертации был предложен алгоритм обучения таких сетей, но его результаты оставались почти неизвестными для широкого круга исследователей. Метод обучения многослойных сетей, получивший название метода обратного распространения ошибки (ОРО), в английском написании – Back Propagation Error (BPE), был открыт вновь в 1986 г. [6], после чего начался интенсивный период разработки нейросетевых технологий на основе многослойных сетей в приложениях к различным сферам науки и техники. Рассмотрим метод ОРО для многослойного прямонаправленного персептрона (МПП), предназначенного для решения задач, которые могут быть выражены в виде образов (пар) вход-выход. Такие соотношения можно назвать обучающими примерами.

Метод обратного распространения ошибки

Обучение многослойного прямонаправленного персептрона состоит в адаптации всех синаптических весов таким образом, чтобы разница между действительными выходными и требуемыми сигналами, усредненная по всем обучающим примерам, была настолько мала, насколько это возможно. Обучение сети методом ОРО включает в себя три этапа:

– прямое распространение входного обучающего образа;

– вычисление ошибки и ее обратное распространение;

– регулирование весов.

После обучения сети ее использование состоит только из вычислений первой фазы. Хотя обучение сети может представлять собой медленный процесс, обученная сеть выполняет свою задачу очень быстро.

Этот метод основан на вычислении вектора градиента поверхности ошибок, который указывает направление кратчайшего спуска по поверхности из данной точки. Последовательность шагов приводит после ряда итераций к минимуму поверхности ошибок. Очевидную трудность здесь представляет выбор длины шага. При большой дли-

не сходимость более быстрая, но есть опасность «перескочить» через решение, особенно в случаях, когда поверхность отклика имеет форму узкого оврага. При малом шаге направление продвижения выбирается правильным, но требуется много итераций для достижения минимума. На практике величина шага принимается пропорциональной крутизне склона с некоторой постоянной, называемой скоростью обучения.

Для конкретности рассмотрим МПП, состоящий из входного, скрытого и выходного слоев нейронов, схема которая приведена на рис. 1.12.

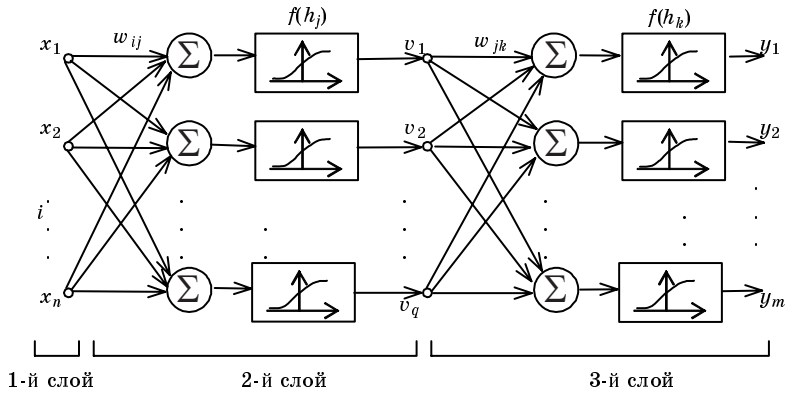


Рис. 1.12

Обозначим x_i ($i = 1, 2,..., n$) – входные нейроны; v_j ($j = 1, 2,..., q$) – нейроны скрытого слоя; y_k ($k = 1, 2,..., m$) – выходные нейроны; w_{ij} – веса от ячеек входного слоя к нейронам скрытого слоя; w_{jk} – веса от нейронов скрытого слоя к выходным нейронам. Индексом p ($p = 1, 2,..., P$) обозначим различные образы, предъявляемые на вход. Входные сигналы могут быть бинарными, биполярными или непрерывными.

Поведение сети должно быть определено на основе ряда пар вход-выход. Каждый обучающий пример состоит из n входных сигналов x_i и m соответствующих выходных сигналов t_k. Обучение МПП для конкретной задачи эквивалентно нахождению таких значений всех синаптических весов, при которых для соответствующего входа формируется требуемый выход. Таким образом, обучение многослойной сети заключается в регулировании всех весов таким образом, что

ошибка между требуемыми выходными t_k и действительными выходными y_k сигналами, усредненная по всем обучающим примерам, была бы минимальна (возможно нулевая). При предъявлении образа p на вход сети скрытая ячейка j принимает сигнал:

$$h_j^p = \sum_i w_{ij} x_i^p$$

и на своем выходе с помощью функции активации вырабатывает такой сигнал:

$$v_j^p = f(h_j^p) = f(\sum_i w_{ij} x_i^p).$$

Выходная ячейка с номером k суммирует сигналы от нейронов скрытого слоя, образуя сигнал, равный

$$h_k^p = \sum_j w_{jk} v_j^p = \sum_j w_{jk} f(\sum_i w_{ij} x_i^p),$$

и после воздействия функции активации формирует выходной сигнал:

$$y_k^p = f(h_k^p) = f(\sum_j w_{jk} v_j^p) = f[\sum_j w_{jk} f(\sum_i w_{ij} x_i^p)].$$

Для упрощения записи здесь пороговые значения отброшены; они всегда могут быть введены в сеть.

В качестве функции ошибок примем функцию вида:

$$E[w] = 0,5 \sum_p \sum_k (t_k^p - y_k^p)^2, \qquad (1.5)$$

где $t_k{}^p$ – требуемое значение выхода.

Подставляя в выражение (1.5) значение $y_k{}^p$, получим

$$E[w] = 0,5 \sum_p \sum_k \{t_k^p - f[\sum_j w_{jk} f(\sum_i w_{ij} x_i^p)]\}^2. \qquad (1.6)$$

Функция (1.6) является непрерывно дифференцируемой функцией от каждого веса, входящего в это выражение, поэтому можно воспользоваться алгоритмом градиентного спуска для нахождения весов. Большую практическую важность здесь играет форма правил изменения весов.

Для весов между скрытым и выходным слоями правило градиентного спуска дает:

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = \eta \sum_p (t_k^p - y_k^p)\ f'(h_k^p)\ v_j^p = \eta \sum_p \delta_k^p v_j^p, \qquad (1.7)$$

где

$$\delta_k^p = f'(h_k^p)\,(t_k^p - y_k^p). \qquad (1.8)$$

Для весов между входным и скрытым слоями нужно продифференцировать (1.6) по w_{jk}, для чего воспользуемся цепным правилом и получим

$$\Delta w_{ij} = \eta \frac{\partial E}{\partial w_{ij}} = -\eta \sum_p \frac{\partial E}{\partial v_j^p} \frac{\partial v_j^p}{\partial w_{ij}} = \eta \sum_p \sum_k (t_k^p - y_k^p)\, f'(h_k^p)\, w_{jk} f'(h_j^p)\, x_i^p =$$

$$= \eta \sum_p \sum_k \delta_k^p w_{jk} f'(h_j^p)\, x_i^p = \sum_p \delta_j^p x_i^p, \qquad (1.9)$$

где

$$\delta_j^p = f'(h_j^p) \sum_k w_{jk} \delta_k^p. \qquad (1.10)$$

Отметим, что уравнения (1.7) и (1.9) имеют одинаковую форму, но различаются значениями параметра δ. В общем случае при произвольном числе слоев правило изменения весов в методе ОРО имеет следующий вид:

$$\Delta w_{\alpha\beta} = \eta \sum_p \delta_{out} v_{in}, \qquad (1.11)$$

где суммирование производится по всем предъявляемым образам; выход и вход относятся к двум концам α и β синаптического соединения; v_{in} – активация от скрытой ячейки или реального входа. Значения δ зависят от рассматриваемого слоя: для последнего слоя эта величина определяется по выражению (1.8), а для других слоев – формулой, подобной (1.10).

Выражения (1.7) и (1.9), определяющие правила изменения весов, записаны в виде сумм по предъявляемым образам, однако обычно образы поступают на вход последовательно: образ p предъявляется на вход, и по окончании прохода «вперед-назад» по сети все веса изменяются перед предъявлением следующего образа. Это уменьшает функцию ошибок E на каждом шаге. Если образы выбираются в случайном порядке, то движение по пространству весов происходит стохастически, что позволяет более широко исследовать поверхность ошибок. Альтернативная версия изменения весов («групповое обучение») заключается в минимизации функции ошибки таким образом, что весовые изменения накапливаются по всем обучающим примерам, и только затем происходит модификация весов.

Алгоритм обучения сети

Алгоритм обучения сети представляет следующую последовательность шагов.

1. Инициировать веса, приняв их малыми случайными величинами.
2. Если условие остановки не выполняется, делать шаги 3–10.
3. Для каждой обучающей пары выполнять шаги 4–9.

Прямой проход по сети

4. Каждая входная ячейка x_i, $i = 1, 2,\ldots, n$ принимает входной сигнал и распространяет его ко всем нейронам следующего (скрытого) слоя.

5. Каждая ячейка скрытого слоя v_j, $j = 1, 2,\ldots, q$ суммирует свои взвешенные входные сигналы

$$h_j = \sum_i w_{ij} x_i,$$

применяет к полученной сумме функцию активации, формируя выходной сигнал:

$$v_j = f(h_j),$$

который посылается ко всем ячейкам выходного слоя.

6. Каждая выходная ячейка y_k, $k = 1, 2,\ldots, m$ суммирует взвешенные сигналы

$$h_k = \sum_j w_{jk} v_j,$$

формируя после применения функции активации выходной сигнал сети:

$$y_k = f(h_k).$$

Обратное распространение ошибки

7. Каждая выходная ячейка сопоставляет свое значение выхода с требуемой целевой величиной и вычисляет параметр δ_k

$$\delta_k = (t_k - y_k) f'(h_k),$$

после чего определяется корректировочный член для весов

$$\Delta w_{jk} = \eta \delta_k v_j,$$

а параметры δ_k посылаются в нейроны скрытого слоя.

8. Каждая скрытая ячейка v_j суммирует свои δ-входы от нейронов выходного слоя

$$h_j = \sum_k \delta_k w_{jk},$$

результат умножается на производную от функции активации для определения δ_j

$$\delta_j = f'(h_j)\sum_k \delta_k w_{jk},$$

и вычисляется поправочный член

$$\Delta w_{ij} = \eta \delta_j x_i.$$

Изменение весов

9. Веса между скрытым и выходным слоями модифицируются следующим образом:

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}.$$

Аналогичным образом изменяются веса между входным и скрытым слоями

$$w_{ij}(new) = w_{ij}(old) + \Delta w_{ij}.$$

10. Проверка условия остановки.

Вычислительная схема алгоритма ОРО показана на рис. 1.13.

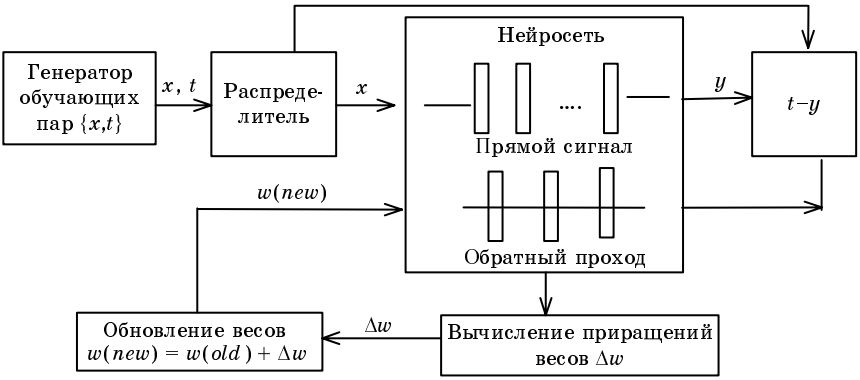


Рис. 1.13

Приведенный алгоритм осуществляет изменение весов после каждого предъявления обучающего вектора, что определяет эпоху в методе ОРО. Обычно требуется много эпох для обучения нейронной сети методом ОРО.

Укажем несколько практических рекомендаций при использовании этого алгоритма.

Выбор начальных значений весов. Этот этап выполнения алгоритма оказывает влияние на достижение сетью глобального (или локального) минимума функции ошибки и на скорость схождения к мини-

муму. С одной стороны, значения начальных весов не должны быть очень большими, иначе начальные входные сигналы в каждую скрытую или выходную ячейку попадут в диапазон, где производная сигмоидной функции активации имеет очень малую величину. С другой – если начальные значения взять достаточно малыми, то сетевой вход в скрытый или выходной нейрон будет близок к нулю, что приведет к очень медленному обучению.

Общее правило инициирования начальных весов заключается в выборе их значений из равномерно распределенных величин в интервале $-0{,}5 \ldots + 0{,}5$ (иногда этот интервал может быть несколько меньше или больше, но не превышающий ± 1). Значения весов могут быть положительными или отрицательными, поскольку окончательные веса после обучения также могут иметь любой знак.

Продолжительность обучения сети. Так как целью обучения сети является достижение баланса между правильными откликами на обучаемые образы и приемлемыми откликами на новые входные образы (т. е. равновесие между запоминанием и обобщением), поэтому нет необходимости продолжать обучение до тех пор, пока общий квадрат ошибки достигнет минимума. Было предложено использовать две серии данных во время обучения: серию обучающих образов и серию контрольных образов, которые являются раздельными. Регулирование весов основано на обучающих образах, однако во время обучения ошибка вычисляется с использованием контрольных образов. До тех пор пока ошибка для контрольных образов уменьшается, процесс обучения продолжается. При возрастании этой ошибки сеть начинает терять свою способность к обобщению, и в этот момент обучение прекращается.

Количество требуемых обучаемых пар. Для соотношения между числом обучаемых образов P, количеством регулируемых весов w и точностью классификации ε предложено использовать следующее выражение [8]:

$$w / P = \varepsilon .$$

Например, при $\varepsilon = 0{,}1$ многослойная сеть с 80 регулируемыми весами потребует 800 обучаемых образов, чтобы быть уверенным в правильной классификации 90% предъявленных контрольных образов.

Представление данных. Во многих задачах входные и выходные векторы имеют составляющие в одном и том же диапазоне величин. Так как один из членов в выражении для корректировки весов является активацией ячейки предыдущего слоя, нейроны, имеющие нулевую активацию, обучаться не будут. Это предполагает, что обуче-

ние может быть улучшено в том случае, если входной вектор представлен в биполярной форме, а в качестве функции активации используется биполярная сигмоида.

Иногда данные (входные и целевые образы) могут быть представлены или в виде непрерывных значений, или в виде набора состояний. К примеру, температура пищи может определяться действительной температурой или одним из следующих состояний: замороженная, холодная, комнатная, горячая. В первом случае используется единственный нейрон; во втором – потребуется четыре нейрона. Для нейронной сети легче обучиться набору различных состояний, чем отклику с непрерывным значением.

Введение инерционной поправки. Можно показать, что поиск минимума функции ошибок методом градиентного спуска оказывается достаточно медленным при малой скорости обучения η ; при большой скорости η имеем значительные осцилляции . В качестве примера на рис. 1.14 показаны четыре траектории пути градиентного спуска, состоящих из 20 шагов (1 – η = 0, 02; 2 – η = 0, 047; 3 – η = =0,049; 4 – η = 0,50). Отдельная траектория соответствует определенному значению скорости обучения, откуда видно, что ее увеличение приводит к значительным колебаниям траектории относительно контура постоянной ошибки (в виде эллипса).

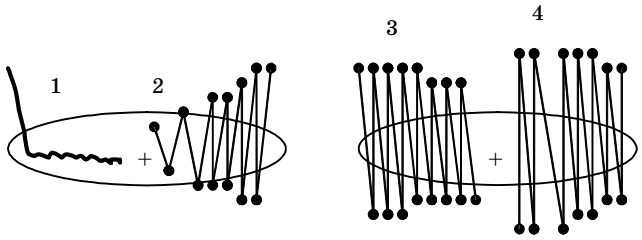


Рис. 1.14

Есть ряд путей для решения этой проблемы, включая замену градиентного спуска сложными минимизационными алгоритмами, однако существует более простое и эффективное приближение, заключающееся в добавлении инерционного (моментного) члена к правилу изменения весов. Такая добавка к весу вынуждает его изменяться в направлении “среднего” спуска, вместо того чтобы осциллировать относительно положения минимума. Подобная схема реализуется посредством вклада от предыдущего временного шага к каждому изменению веса

$$\Delta w_{jk}(t+1) = -\eta \partial E / \partial w_{jk} + \alpha \Delta w_{jk}(t),$$

из которого при замене первого слагаемого в правой части получаем:

$$\Delta w_{jk}(t+1) = \eta \delta_k v_j + \alpha \Delta w_{jk}(t).$$

Последнее равенство определяет изменение весов между скрытым и выходным слоями; аналогично добавляется поправочный член и к весам между скрытым и входным слоями. Обычно величина моментного (инерционного) параметра α находится в диапазоне от 0 до 1, наиболее часто его значение принимается равным 0,9.

На рис. 1.15 приведены примеры двух траекторий градиентного спуска, состоящего из 12 шагов при одинаковой скорости обучения η. Левая траектория соответствует случаю отсутствия инерционного члена ($\alpha = 0$), правая – соответствует значению $\alpha = 0{,}5$.

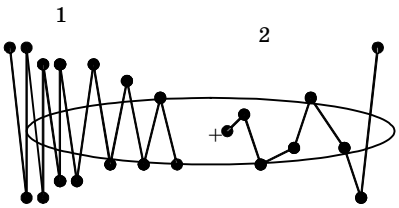


Рис. 1.15

Модификации функции активации. Диапазон функции активации должен соответствовать диапазону целевых значений конкретной задачи. Бинарная сигмоидная функция вида

$$f(x) = [1 + \exp(-x)]^{-1}$$

с производной

$$f'(x) = f(x)[1 - f(x)]$$

может быть изменена для перекрытия любого требуемого диапазона с центром при любом значении x и необходимом наклоне.

Сигмоида может иметь расширенный диапазон, чтобы отображать значения в интервале $[a, b]$ для любых a и b. Для этого нужно ввести параметры

$$\gamma = b - a; \quad \rho = -a.$$

Тогда сигмоидная функция

$$g(x) = \gamma\ f(x) - \rho$$

имеет требуемые свойства, т. е. диапазон $[a, b]$. Ее производная выражается как

$$g'(x) = \gamma^{-1}[\rho + g(x)][\gamma - \rho - g(x)].$$

Наклон сигмоидной функции может быть изменен с помощью введенного параметра β. Вид сигмоидной функции в этом случае показан на рис. 1.8.

Альтернативные функции стоимости. Квадратичная функция стоимости (1.6) является не единственно возможной. Выражение $(t_k^p - y_k^p)$ может быть заменено любой другой дифференцируемой функцией $f(t_k^p, y_k^p)$, которая достигает минимума при равных аргументах. Дифференцирование такой функции показывает, что изменяется только величина δ_k^p для весов между выходным и скрытым слоями; остальные выражения метода ОРО остаются неизменными.

Возможная альтернативная функция стоимости приведена в [8] и имеет следующий вид:

$$E = \sum_{p,k} \begin{cases} \gamma(\, t_k^p - y_k^p)^2, & \text{если } sign\ t = sign\ y, \\ (t_k^p - y_k^p)^2, & \text{если } sign\ t = -sign\ y, \end{cases}$$

с параметром γ, изменяющимся от 0 до 1. Такое выражение вначале устанавливает правильным знак t, а затем обращает внимание на величину расхождения между t и y.

Число нейронов в скрытом слое. Обычно число нейронов в скрытом слое заранее неизвестно, и для выбора количества скрытых нейронов используют метод проб и ошибок. Возможный подход при выборе скрытых нейронов заключается в том, что на первом этапе число таких нейронов берется заведомо большим, чем требуется, а затем, по мере обучения сети, излишние нейроны убираются. Все нейроны, которые не вносят вклад в решение или дают информацию, не требующуюся в последующем слое, рассматриваются как лишние и удаляются из скрытого слоя. Для того чтобы найти те нейроны, которые можно исключить, выходы всех скрытых ячеек запоминаются и анализируются по всем обучающим примерам после того, как сеть достигла сходимости. На практике считается, что сеть достигла сходимости, если разность между требуемым и действительным выходами не превышает 0,1. В случае, если выход некоторой скрытой ячейки остается примерно постоянным для всех обучающих примеров, этот нейрон может быть удален из сети, так как он не вносит существенного вклада в решение и действует как дополнительное смещение. Аналогичным образом, если два скрытых нейрона дают приблизительно одинаковый выход для всех обучающих примеров, то только один из них действительно необходим, так как оба нейрона переносят одинаковую информацию. После удаления тех ячеек, которые не дают вклада в решение, значения веса уменьшенной сети должны

быть изменены путем переобучения сети для получения требуемых характеристик.

Разновидности градиентных алгоритмов обучения

Метод сопряженных градиентов. В этом методе происходит последовательный поиск минимума по различным направлениям на поверхности отклика. Сначала, как и в методе ОРО, берется направление наискорейшего спуска, однако здесь шаг берется непропорционально скорости обучения: в методе сопряженных градиентов проводится прямая в нужном направлении, а затем вдоль нее ищется минимум (это не требует больших вычислений, так как поиск проходит в одном измерении).

Затем происходит поиск в новых направлениях (сопряженных), которые выбираются из тех соображений, чтобы не терять минимума по уже выбранным ранее направлениям.

В действительности сопряженные направления вычисляются в предположении, что поверхность ошибок является квадратичной, что не всегда справедливо. Однако это предположение работоспособно, а если алгоритм обнаруживает, что выбранное направление не ведет вниз по поверхности отклика, то он находит направление наискорейшего спуска и начинает поиск заново с этого направления.

Начальное направление поиска d_0 задается следующей формулой:

$$d_0 = -g_0,$$

где g_0 – величина градиента.

На последующих шагах направление поиска d_{j+1} корректируется следующим образом:

$$d_{j+1} = -g_{j+1} + \beta_j d_j \ .$$

Из последней формулы следует, что новое направление минимизации зависит только от значения градиента в точке решения g_{j+1} и от предыдущего направления поиска d_j, умноженного на коэффициент сопряжения β_j. Для вычисления последнего существуют различные правила [9], в частности, можно использовать и такое:

$$\beta_j = \left[g_{j+1}(g_{j+1} - g_i) \right] / g_j^2 \ .$$

Практическое применение такого подхода связано с постепенной утратой свойства ортогональности между векторами направлений минимизации.

Метод Левенберга–Маркара. Этот метод – самый быстрый и надежный метод, но его применение связано со следующими ограничениями:

1. Метод Левенберга–Маркара можно применять только для сетей с одним выходным нейроном.

2. Метод требует памяти, пропорциональной квадрату числа весов в сети, поэтому его нельзя использовать для сетей большого размера (порядка 1000 и более весов).

Этот метод предполагает, что функция, моделируемая ИНС, является линейной. В таком предположении минимум определяется за один шаг вычислений. Затем найденный минимум проверяется, и если ошибка уменьшилась, весам присваиваются новые значения. Вся процедура повторяется.

Алгоритм Левенберга–Маркара разработан так, чтобы минимизировать функцию ошибок с помощью формулы, которая предполагает, что моделируемая сетью функция является линейной. Вблизи точки минимума это предположение выполняется с большой точностью, так что алгоритм может продвигаться очень быстро. Вдали от минимума это предположение может быть неправильным. Поэтому метод Левенберга–Маркара находит компромисс между линейной моделью и градиентным спуском. Шаг делается только в том случае, если он уменьшает ошибку.

Итерации в этом методе проводятся по следующей формуле:

$$\Delta w = -(Z^T Z + \lambda I)^{-1} Z^T \varepsilon,$$

где λ – управляющий параметр; I – единичная матрица; ε – вектор ошибок на всех наблюдениях; Z – матрица частных производных от этих ошибок по весам, равная

$$(Z)_{ni} = \frac{\partial \varepsilon^n}{\partial w_i}.$$

Метод быстрого распространения. Этот метод является разновидностью метода ОРО. В методе быстрого распространения (БР) производится пакетная обработка данных. В методе ОРО веса сети корректируются после обработки каждого очередного наблюдения; здесь же вычисляется усредненный градиент поверхности ошибок по всему обучающему множеству, и веса корректируются один раз в конце каждой эпохи.

Метод БР действует в предположении, что поверхность ошибок является квадратичной. Если это так, то точка минимума на ней находится через одну-две эпохи. В общем случае такое предположение неверно, но даже если оно выполняется лишь приблизительно, алгоритм все равно быстро сходится к минимуму.

При этом допущении алгоритм БР работает так:

1. На первой эпохе веса корректируются по тому же правилу, что и в методе ОРО, исходя из локального градиента и скорости обучения.

2. На последующих этапах алгоритм использует предположение о квадратичности для более быстрого продвижения в точку минимума. Здесь изменения весов рассчитываются по следующей формуле:

$$\Delta w(t) = \frac{s(t)}{s(t-1) - s(t)} \Delta w(t-1),$$

где $s(t)$ – градиент поверхности ошибки по отношению к данному весу.

Существуют и другие виды градиентного поиска, отличающиеся способом вычисления приращения весов по поверхности ошибок. В любом случае алгоритм работает по правилу

$$w^{(n+1)} = w^{(n)} + \Delta w^{(n)},$$

где n определяет шаг итерации.

1.7. Конкурентные сети

Отличие сетей этого типа от рассмотренных ранее, которые обучались с учителем (на примерах), заключается в том, что теперь нет связи из внешнего мира, чтобы сказать, какие из выходов сети являются правильными. Сеть в такой ситуации должна сама отыскивать во входных данных образы, признаки, корреляции, категории и кодировать их на выходе. Таким образом, ячейки и связи между ними здесь выполняют роль самоорганизации (самообучения). Тип образа, который сеть данного типа обнаруживает во входных данных, зависит от архитектуры и выходов сети. Укажем следующие возможные выходы [7].

Подобие. Единственный выход сети может дать оценку сходства нового входного образа с типичным образом, показанным сети в прошлом.

Главные компоненты (ГК). Данный подход представляет собой расширение предыдущего случая на несколько ячеек или ряд осей, вдоль которых измеряется сходство с предшествующими примерами. Отметим, что ГК широко распространены в статистических приложениях при многомерном анализе.

Кластеризация. Ряд бинарных выходов (в каждый момент времени только один) позволяет оценить принадлежность входного образа к одному из нескольких классов. Приемлемые категории должны быть найдены сетью на основе корреляций во входных образах.

Кодирование. Выход может быть кодированной версией входа, сохраняющей максимальное количество информации о предъявленном входе.

Отображение признаков. При организации выходных ячеек в некоторую фиксированную конфигурацию (к примеру, двухмерное устройство) они могут отображать входные образы в различные точки этой конфигурации. При этом создается топографическая карта входа таким образом, что сходные входные ячейки всегда активируют похожие выходные ячейки.

Перечисленные случаи четко не различаются и могут комбинироваться в различных сочетаниях. В частности, кодирование может быть выполнено с использованием главных компонентов, которые сами могут применяться для снижения размерности данных перед кластеризацией или отображением.

Архитектура сетей, обучаемых без учителя достаточно проста: большинство сетей состоит из одного слоя. В основном сети являются прямонаправленными, за исключением сетей адаптивной резонансной теории. Сложность в таких сетях обусловлена только типом обучающих правил.

Правило Ойя

Рассмотрим вначале однослойную линейную сеть (рис. 1.16), состоящую из нескольких входных ячеек, число которых определяется размерностью входного вектора, и одной выходной ячейки. Примем, что на вход сети поступают поочередно векторы x с составляющими x_j, $i = 1, 2, \ldots, n$, взятыми из некоторого распределения $P(x)$. Компоненты вектора могут быть непрерывными или бинарными величинами. При каждом временном шаге очередной вектор x выбирается из распределения и поступает на вход сети. После просмотра достаточного количества таких векторов сеть должна обучиться до такой степени, чтобы по своему выходу сделать вывод, насколько хорошо предъявленный сети образ согласуется с исходным распределением.

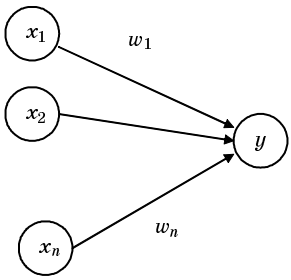


Рис. 1.16

При архитектуре сети на ее выходе имеем:

$$y = \sum_j w_j x_j.$$

При единственной выходной ячейке необходимо, чтобы этот выход стал скалярной мерой сходства. Чем ближе отдельный входной образ x к векторам, взятым из распределения $P(x)$, тем большим должно быть значение выходной ячейки (по крайней мере, в сред-

нем). В такой ситуации естественно воспользоваться простым обучением Хебба, т. е. для изменения весов принять

$$\Delta w_j = \eta y x_j, \tag{1.12}$$

где η – скорость обучения.

Такой выбор усиливает выход по очереди для каждого предъявляемого входа и будет приводить к наибольшему выходу. Однако при этом веса продолжают расти безгранично, и обучение никогда не прекратится. Рассмотрим правило (1.12) более подробно. Положим, что имеется точка равновесия для вектора w. После достаточного обучения весовой вектор будет оставаться по соседству с точкой равновесия, только флуктуируя вокруг нее на величину, пропорциональную η. Вследствие этого можно ожидать, что весовые изменения в среднем должны быть равны нулю:

$$0 = \langle \Delta w_i \rangle = \langle y x_i \rangle = \left\langle \sum_j w_j x_j x_i \right\rangle = \sum_j C_{ij} w_j = Cw, \tag{1.13}$$

где угловые скобки определяют среднее по входному распределению $P(x)$ и корреляционную матрицу C как

$$C_{ij} = \langle x_i x_j \rangle.$$

Полученная матрица C_{ij} является симметричной, т. е. собственные числа этой матрицы – действительные числа, а собственные векторы – ортогональны. Из равенства (1.13) следует, что в гипотетической точке равновесия вектор w есть собственный вектор матрицы C с нулевым собственным числом. Но такое состояние не может быть стабильным, так как у матрицы C есть некоторые положительные собственные числа; любая флуктуация, имеющая составляющую вдоль собственного вектора с положительным собственным числом, будет расти экспоненциально. Следовательно, направление, соответствующее наибольшему собственному числу $\lambda_{\max}$, будет доминировать, а весовой вектор w при этом будет постепенно приближаться к собственному вектору, соответствующему $\lambda_{\max}$, неограниченно возрастая.

Имеется несколько возможностей предотвратить рост весового вектора. Например, это можно сделать простой перенормировкой, приняв $w_i' = \alpha w_i$ для всех весов после каждого изменения, выбирая α так, чтобы $|w'| = 1$. Но финский ученый Е. Ойя предложил оригинальное решение, заключающееся в модификации самого правила Хебба таким образом, который позволяет сделать весовой вектор приближающимся к постоянной длине $|w| = 1$ без перенормировки после

каждого изменения. Кроме того, предложенный способ приближает вектор w к собственному вектору матрицы C с наибольшим собственным числом λ_{max}.

Правило Ойя соответствует добавлению спадающего члена, пропорционального y^2, в простое правило Хебба, что приводит к следующему выражению:

$$\Delta w_j = \eta y(x_j - y w_j). \qquad (1.14)$$

Как следует из формулы (1.14), правило Ойя похоже на правило обратного распространения: Δw_j зависит от разности между действительным входом и «обратным» выходом.

Е. Ойя показал, что его правило выбирает направление w, которое максимизирует средний квадрат выхода $\langle y^2 \rangle$. Этот вывод из правила Ойя приводит к заключению, что выбранное направление совпадает с направлением первого ГК для данных с нулевым средним значением. Метод ГК, широко используемый в многомерной статистике, заключается в нахождении набора из m ортогональных векторов в n-мерном исходном пространстве данных. Обычно $m<<n$, что дает возможность говорить о снижении размерности первоначального вектора наблюдений; при этом такое преобразование часто сохраняет большую долю информации о данных. В статистике для нахождения ГК нужна матрица корреляций между исходными признаками, основываясь на которой, определяются матрицы собственных векторов и чисел. Первый ГК соответствует наибольшему собственному числу и рассчитывается, используя составляющие собственного вектора, определяемого λ_{max}, которое равно наибольшей дисперсии признаков. Таким образом, первый ГК берется вдоль направления с максимальной дисперсией. Второй ГК должен лежать в подпространстве, перпендикулярном тому, где находится первый ГК. В пределах этого подпространства второй ГК берется вдоль направления с максимальной дисперсией. Третий ГК находится в направлении наибольшей дисперсии в подпространстве, перпендикулярном первым двум и т. д.

Правило (1.14) определяет первый ГК, однако иногда необходимо иметь сеть с m выходами, которая выделяет первые m ГК. Е. Ойя расширил свое правило на случай m выходных ячеек, которое в этом случае имеет вид:

$$\Delta w_{ij} = \eta y_i(x_j - \sum_{k=1}^{n} y_k w_{kj}). \qquad (1.15)$$

Выходы такой сети проектируют входной вектор x на пространство первых m ГК. Отметим, что правило, аналогичное правилу (1.15),

было разработано Т. Зангером. Единственная разница между двумя выражениями – верхний предел суммы: в правиле Зангера он составляет величину i вместо n в правиле Ойя. Правило Зангера более полезно в приложениях, так как оно выделяет ГК по порядку, и дисперсия выходов уменьшается равномерно с увеличением номера i.

Самоорганизующиеся карты Кохонена

Карты Кохонена, получившие свое название по имени разработавшего их финского ученого Т. Кохонена [10], представляют собой прямонаправленные сети, которые используют алгоритм НСО. Такие сети с помощью процесса, названного самоорганизацией, формируют выходные ячейки в топологическую карту, которая может иметь одномерную (рис. 1.4) или двухмерную структуру (рис. 1.17).

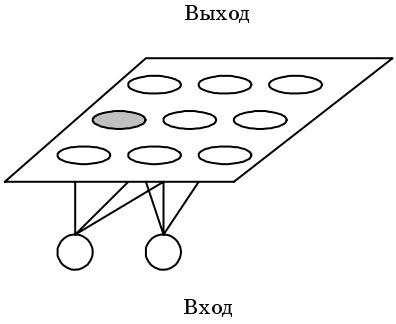


Рис. 1.17

Нейронная сеть карт Кохонена состоит из входного и выходного слоев (скрытый слой отсутствует). При предъявлении входного образа на отображающую карту нейроны в выходном слое конкурируют друг с другом за право быть «победителем». Победителем считается тот нейрон, у которого входящие веса являются самыми близкими (в смысле евклидова расстояния) к входному образу. Таким образом, каждая выходная ячейка определяет свое сходство или близость с входным предъявляемым образом, и победителем становится ячейка с номером j^*, для которой выполняется соотношение:

$$\left|w_{j^*} - x\right| \le \left|w - x\right| \text{ для всех } j.$$

Выходная ячейка, которая стала победителем, получает право на регулирование своего веса. В алгоритме Кохонена правило изменения весов имеет следующий вид:

$$\Delta w_{ij} = \eta \Lambda(j, j^*)(x_j - w_{ij}), \qquad (1.16)$$

где $\Lambda(j, j^*)$ – функция соседства, равная единице для $j = j^*$ и уменьшающаяся с расстоянием $|r_{j^*} - r_j|$ между нейронами j и j^* в выходном слое.

В итоге, карты Кохонена создают топологию путем регулирования весов не только нейрона-победителя, но и при помощи изме-

нения весов соседних выходных ячеек в непосредственной близости от победителя. Веса победителя и его соседей передвигаются в направлении входного образа на величину, определяемую скоростью обучения. Нейроны, близкие к победителю, так же, как и сам победитель, изменяют свои веса в значительной степени, в то время как удаленные от победителя ячейки испытывают меньшие изменения весов. При продолжении процесса обучения размер области соседства от нейрона-победителя уменьшается. В начале обучения значительное число выходных нейронов будут изменяться, но затем все меньше и меньше ячеек подвергается модификации своих весов. В конце обучения только победитель регулирует свой вес. Аналогичная процедура происходит и со скоростью обучения: по мере обучения скорость уменьшается, и ее снижение до нуля останавливает обучающий процесс.

Правило (1.16) перетягивает весовой вектор w_{j*}, принадлежащий победителю, к вектору x. Кроме того, это правило приближает вектора w_j ближайших соседей. Вследствие этого можно представить на выходе тип «эластичной сети», смещающейся так близко, насколько это возможно, к входному образу.

С точки зрения использования карт Кохонена представляет интерес задача разбиения объектов (признаков) на сходные группы (задача кластеризации), при решении которой индексы ячейки-победителя определяют номер категории, куда относится предъявляемый входной образ. Помимо этого, сети Кохонена широко используются при распознавании звуков речи, отпечатков пальцев, при оценке состояния механизмов.

Приведем алгоритм обучения сети Кохонена, который представляет следующую последовательность шагов.

1. Инициировать веса. Установить параметры функции соседства и скорости обучения.

2. Если условие остановки не выполняется, делать шаги 3–9.

3. Для каждого входного вектора x делать шаги 4–6.

4. Для каждого j вычислить $D(j) = \sum w_{ij} - x_i)^2$.

5. Найти индекс j^* такой, что $D(j^*) = \min$.

6. Для всех ячеек j в пределах определённого соседства от j^* и для всех i найти $w_{ij}(new) = w_{ij}(old) + \eta\Lambda(j, j^*)[x_j - w_{ij}(old)]$.

7. Изменить скорость обучения.

8. Уменьшить радиус функции соседства.

9. Проверить условие остановки.

1.8. Алгоритмы решения задач с помощью нейронных сетей

Для решения задачи независимо от ее конкретного содержания посредством ИНС необходимо выполнить следующую совокупность операций:

– определить составляющие входного вектора x;

– установить выходной вектор y таким образом, чтобы его компоненты представляли собой ответ на поставленную задачу;

– выбрать количество слоев сети и нейронов в каждом слое. При этом числа нейронов в входном и выходном слоях определяются размерностями векторов, соответственно, x и y. При использовании сети Кохонена размерность выхода определяется количеством требуемых классов (групп), на которые должна быть разделена совокупность;

– оценить диапазоны изменения входов и выходов; при необходимости произвести нормировку входных данных;

– обучить нейронную сеть тем или иным способом, т. е. подобрать параметры сети (значения весов) таким образом, чтобы минимизировать ошибку обучения;

– приступить к работе с обученной нейронной сетью, подав на вход сети значение входного вектора x.

Рассмотрим некоторые типичные задачи в области менеджмента, которые можно решать с помощью ИНС.

Задача классификации. Положим, что имеется набор данных об n объектах (фирмах, банках, товарах), каждый из которых характеризуется m-мерным вектором x, т. е. исходная матрица данных состоит из n строк и m столбцов. Необходимо посредством ИНС разделить объекты на ряд сходных групп.

Такая задача может решаться как с использованием многослойного персептрона, так и сети Кохонена. В первом случае применяется обучение с учителем, во втором – без учителя.

Количество слоев в многослойном персептроне обычно не превышает трех (входной, скрытый, выходной), что следует из теоремы А. Н. Колмогорова. В последней доказано, что аппроксимация функции, трансформирующей m-мерный входной вектор в M-мерный выходной вектор, осуществима нейросетью с одним скрытым слоем с $(2m+1)$ нейронами. По условию задачи размерность входного вектора равна m, поэтому входной слой содержит m нейронов; число нейронов в выходном слое зависит от того, на сколько классов предполагается разбить выборку данных.

В задаче сеть должна отнести каждый объект к одному из выделенных классов. Для классификации используется номинальная выходная переменная в одном из двух видов кодировки:

– бинарная;

– кодировка 1-из-M.

При бинарном представлении в выходном слое сети имеется один нейрон, принимающий одно из двух значений (0 или 1). В такой кодировке происходит разделение данных на два класса.

При кодировке вида 1 – из – M на каждое состояние выделяется один нейрон, так что каждое конкретное состояние представляется как 1 на соответствующем месте и 0 во всех других положениях, например, для четырех классов имеем такие значения выходных ячеек: (1 0 0 0), (0 1 0 0), (0 0 1 0), (0 0 0 1).

Каждый из выходных нейронов сети будет содержать числовые значения в интервале от 0 до 1. Для определения класса по набору выходных значений сеть должна решить, достаточно ли близки они к нулю или единице. При нахождении такого решения устанавливаются пороговые значения принятия и отвержения решений. Эти пороговые значения можно корректировать, чтобы заставить сеть быть более или менее решительной при объявлении класса. На практике обычно не добиваются полного совпадения выходного значения сети с 0 или 1. Расхождение между требуемым и действительным выходами в пределах 5% вполне достаточно для решения большинства практических задач, т. е. пороговые значения при бинарном представлении можно принять равными, соответственно, 0,95 и 0,05.

При решении задачи классификации необходимо обратить внимание на нежелательную возможность переобучения сети. Это явление проще всего продемонстрировать на примере аппроксимации с помощью полиномов. Графики полиномов могут иметь различную форму, и при более высокой степени многочлен имеет достаточно сложную форму. По имеющимся опытным данным (графику зависимости полинома) подбирается модель для объяснения результатов эксперимента. Можно выбрать сложную модель, которая будет точно проходить через все экспериментальные точки, но при такой модели теряется искомая зависимость. Полином низкого порядка может быть недостаточно гибким для аппроксимации данных, но способен уловить общую тенденцию требуемой зависимости. ИНС при обучении сталкивается с такой же проблемой. Сети с большим числом весов моделируют более сложные функции и, следовательно, склонны к переобучению. Вместо способности к обобщению, для чего и обучается нейронная сеть, можно получить такую нежелательную ситуацию.

Для исключения подобных явлений применяется контрольная проверка, при которой используются данные, не предъявляемые сети

при обучении. При обучении сети по методу ОРО для контроля результата обучения применяются обучающая и контрольная выборки. По мере обучения сети ее ошибка убывает, и вместе с тем уменьшается ошибка на контрольной выборке. Прекращение снижения ошибки на контрольной выборке (или даже ее рост) свидетельствует о том, что сеть начала слишком точно аппроксимировать входные данные и обучение необходимо прекратить, иначе можно оказаться в ситуации переобучения.

Как отмечалось выше, для классификации могут использоваться и самообучающиеся сети Кохонена, которые по имеющимся значениям входных переменных определяют их структуру. Такой подход может быть рекомендован для предварительного анализа данных. Сеть Кохонена может объединить данные в кластеры и тем самым установить сходство между различными объектами, участвующими в классификации. В случае уже заданных классов (сеть маркирует классы номерами нейронов выходного слоя) такая ИНС способна выявить сходство между различными классами. При поступлении на вход сети нового, непохожего на уже использованные при классификации образы объекта в виде вектора его признаков, сеть Кохонена не сможет отнести этот объект ни к одному из известных классов и сформирует для него новый класс.

Задача прогнозирования. Такая задача часто ставится с целью определения поведения финансово-экономических показателей в будущие интервалы времени, например, объема продаж некоторого товара, изменения курса акций и т. п. Здесь целью является прогноз будущих значений переменной на основе ее предыдущих значений. Обычно прогнозируемая переменная является числовой, поэтому прогнозирование временных рядов представляет собой частный случай регрессии. Очередное значение ряда прогнозируется по некоторому числу его предыдущих значений (прогноз на один шаг вперед по времени).

Общий метод идентификации таких закономерностей заключается в использовании *метода скользящих окон*. Основная идея этого метода состоит в использовании двух окон W^i и W^0 фиксированных размеров n и m, соответственно, перемещающихся (скользящих) вдоль оси абсцисс (рис. 1.18).

Для данного размера окна последовательность значений $W_0{}^i, ..., W_n{}^i$ связана с последовательностью $W_0{}^0, ..., W_m{}^0$, и эта связь полностью определяется установленными данными. Затем для нахождения двух конфигураций значений могут использоваться различные методы. В случае нейронных сетей $W^i > W^0$ принимается за обучающий вектор.

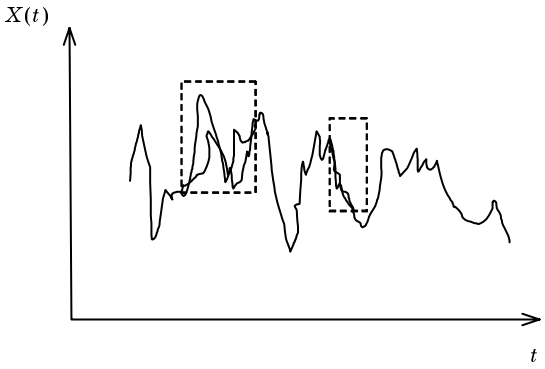


Рис. 1.18

Оба окна смещаются вдоль временного ряда, сохраняя фиксированный размер шага s. Получающаяся пара $W^i > W^0$ используется как элемент обучающей выборки сети.

Приведем пример формирования обучающей пары. Пусть после дискретизации временного ряда имеем следующие данные:

$$25,\ 30,\ 32,\ 29,\ 37,\ 34,\ 35,\ 38,\ 41.$$

Установим такие параметры окон: $n = 4$; $m = 1$; $s = 1$. Тогда для нейронной сети будут сгенерированы следующие пары:

$$25,\ 30,\ 32,\ 29 \to 37;$$
$$30,\ 32,\ 29,\ 37 \to 34;$$
$$32,\ 29,\ 37,\ 34 \to 35 \text{ и т. д.}$$

Каждый следующий вектор формируется путем сдвига окон W^i и W^0 вправо на один элемент, так как $s = 1$. Нейронная сеть, предполагая наличие скрытых зависимостей, обучается на этих наблюдениях, извлекает тенденции изменения и строит прогнозную оценку.

В такой задаче могут применяться следующие два вида прогнозирования:

Многошаговый прогноз. Используется при необходимости долгосрочного прогноза, целью которого является идентификация общих тенденций и наиболее важных поворотных точек во временном ряду. В этом случае система для предсказания временного ряда в фиксированный период использует конфигурацию текущих значений. Затем прогноз снова вводят в сеть, чтобы спрогнозировать следующий период.

Одношаговый прогноз. Здесь в отличие от многошагового прогноза повторно вводить спрогнозированные ранее данные не требуется.

Сеть предсказывает значение ряда на какой-то момент времени, при этом она для каждого последующего прогнозного значения использует фактическую величину, а не значение, предсказанное ранее.

1.9. Предварительная обработка данных

При проведении нейросетевого исследования, перед тем как приступить к построению ИНС, необходимо выполнить ряд действий с данными наблюдений. Данный этап анализа еще слабо формализован, различные авторы предлагают разные преобразования над наблюдениями [11], но основные рекомендации сводятся к следующим.

Сбор данных. Начальный этап включает извлечение высококачественных данных из доступных источников, которые содержат полную и достаточную информацию в требуемой области.

При сборе результатов наблюдений нужно учитывать:

– доступность данных;

– восстановление пропущенных данных;

– очистку данных.

Большую по объему информацию часто трудно или невозможно получить. Исторические данные значительно изменяются после опубликования, например, квартальная информация закрывает недельные или месячные результаты.

При восстановлении пропущенных наблюдений могут быть использованы два подхода. Сущность первого состоит в том, чтобы исключить строку или столбец матрицы данных, где есть пропущенные значения. Альтернативой такому подходу является допущение о том, что пропущенные данные в среднем имеют характер, аналогичный имеющимся, и могут быть восстановлены, например, средние значения из соседних элементов.

Очистка данных предполагает устранение нежелательных шумовых эффектов для получения качественных результатов наблюдений. Зашумленные данные могут значительно искажать оценочные процедуры. Кроме того, при очистке данных необходимо сравнивать поступающие значения с предполагаемым диапазоном изменений результатов наблюдений для того, чтобы отсечь выходящие за пределы диапазона выбросы.

Анализ и преобразование данных. Этот шаг – один из самых важных при нейросетевом проектировании. Анализ входных данных зависит от решаемой проблемы, например, для ИНС, предназначенной для работы на рынке акций, необходимы данные по ценам открытия, закрытия, максимума и минимума. Собранные для анализа данные представляют набор многомерных векторов, отображающих ис-

следуемый процесс. Ключевые вопросы, на которые должен быть получен ответ при отборе входных данных, сводятся к следующим:

Какие основные переменные влияют на выход?

Какие переменные предлагают или используют эксперты?

Какие преобразования данных необходимы?

Что предлагают классические статистические методы?

Какова приемлемая частота отсчета данных?

Собранные данные часто неадекватны для обучения нейронных сетей, вследствие чего анализ и преобразование данных необходимы для улучшения информации, которая способна обеспечить лучшее описание трендов или процессов в анализируемых данных. Цель анализа и преобразования данных – упростить классификацию или предсказание с помощью нейронной сети.

Преобразование данных должно помочь при:

– классификации данных;

– трансформации от нелинейных задач к линейным;

– концентрации усилий только на части входного диапазона.

Наиболее простыми видами преобразований являются процентная разность и логарифмирование; более сложными – статистические преобразования. Процентная разность и логарифмирование легко вычисляются в табличной форме. Например, лаги первого порядка цен закрытия могут быть вычислены с помощью дополнительного столбца для нахождения величины

$$x_i = \ln(P_t / P_{t-1}),$$

где P_t, P_{t-1} – соответственно, текущая цена закрытия и цена закрытия предыдущего дня.

Примером статистических преобразований служит наклон линии регрессии, которая измеряет направление трендов в потоке данных. Более сложные включают преобразования Фурье и вейвлетное, которые в ряде случаев несут больше информации, чем исходные данные, однако для применения этих преобразований от менеджеров требуется более глубокое знание математики.

Обычно нейронные сети плохо работают с величинами из широкого диапазона значений, встречающихся во входных данных. Для исключения этого нежелательного явления данные необходимо промасштабировать в диапазон [0…+1] или [−1… +1]. Нужно заметить, что используемые для формирования выхода сети функции (сигмоидные или гиперболического тангенса) приводят к трудностям получения выходных значений, близких к 1 или 0 (−1 в некоторых случаях). Вследствие этого целесообразно проводить масштабирование таким образом, чтобы выходной диапазон переменных составлял

[+0,2…+0,8] или [–0,8… +0,8]. Формула, по которой можно провести масштабирование входных данных, имеет следующий вид:

$$X_s = Sc * X_u + Of,$$

$$Sc = (T_{\max} - T_{\min}) / (R_{\max} - R_{\min}), \quad Of = T_{\min} - Sc * R_{\min},$$

где X_s, X_u – соответственно, отмасштабированные и исходные входные данные; $T_{\min} = 0$, $T_{\max} = 1$ – максимум и минимум целевой функции; $R_{\max}$, $R_{\min}$ – максимум и минимум входных данных.

Отбор переменных. Эта процедура имеет цель – снизить размерность вектора входных данных. Часто нейронная сеть с меньшим числом входных переменных показывает большую эффективность в работе, чем сеть с большим количеством входов. В случае, если определено приемлемое множество входов для решения конкретной задачи, то такие входные переменные остаются постоянными.

Существует три подхода к отбору используемых входов.

Первое направление связано с использованием статистических методов или методов нелинейной динамики для решения вопроса о частоте отсчетов (если, например, речь идет о временной зависимости).

Во втором способе для обученной сети может быть применен известный метод анализа чувствительности для определения важности входов. Подобная процедура ранжирования входов и исключения нижней части списка входов достаточно эффективна. Статистические методы могут пропустить входы, которые важны только при взаимодействии с другими переменными, но анализ чувствительности обнаруживает такие входы. В этом методе процесс целиком представляет собой последовательность таких шагов: отбор множества данных для входа, обучение сети до принятого уровня ошибки, проведение анализа чувствительности и исключение всех переменных, находящихся ниже порогового уровня.

Альтернативой способу анализа чувствительности является определение относительного воздействия данного входа на среднее значение всех входных переменных. Однако для такой оценки важности входов данные должны быть пронормированы.

Вне зависимости от того, какой из способов снижения размерности входного вектора используется, цель отбора входных переменных, укажем еще раз, заключается в их уменьшении. Это приводит к сокращению времени обучения сети и улучшению в ряде случаев характеристик нейросетевой модели.

Обучающая и тестовая выборки. Напомним, что нейронные сети, о которых идет речь в этом разделе, в первую очередь, предназначены для решения непараметрических, плохо формализуемых задач. Не-

параметрические модели противоположны параметрическим. В последних моделях обычно имеется некоторая формула, связывающая изучаемое явление или процесс с внешней средой. Непараметрические модели аппроксимируют множество соотношений между входными и выходными переменными. Например, ИНС, обучаемые методом обратного распространения ошибки, способны аппроксимировать почти любое соотношение между входными и выходными переменными, что подчеркивает мощность нейронных сетей. По существу, построение сети эквивалентно выводу математической формулы.

Перейдем к обсуждению данных, которые могут быть использованы для обучения и проверки нейронной сети. Обучающая выборка применяется для разработки нейросетевой модели, т. е. для определения значений весов, при которых достигается минимальная ошибка между требуемым и выходным сигналами (в частности, для обучения с «учителем»). Тестовая выборка необходима для того, чтобы удостовериться в обобщающих свойствах сети и в работоспособности последней с данными, не участвовавшими в обучении. (Иногда перед началом работы с обученной сетью используется еще проверочная выборка для оценки действительных характеристик нейросетевой модели в условиях окружающей среды).

Общая процедура разработки нейронных сетей заключается в расщеплении исходных данных на обучающее и тестовое множества. Обычно 2/3 данных используются для обучения, 1/3 – для тестирования. Основное требование к этим множествам состоит в том, чтобы эти выборки были репрезентативны, т. е. правильно отражали весь процесс, из которого извлекаются данные.

Процедура выборки – это снижение популяции данных таким образом, чтобы удовлетворить ряду критериев. Такие критерии включают в себя уменьшение стоимости анализа, ускорение времени обработки и т. п. Традиционные выборки удовлетворяют общим критериям изменением количества наблюдений так, что статистические свойства выборки и популяции (генеральной совокупности) остаются похожими. Однако при формировании выборки для нейронных сетей важно, чтобы данные извлекались пропорционально обучающей сложности.

Деление на обучающую и тестовую выборки требуют, прежде всего, прямонаправленные сети. Обучающая выборка – это множество, с помощью которого сеть итеративным образом снижает ошибку между реальным выходом сети и требуемым выходом. При разработке выборочной стратегии для ускорения нейросетевого моделирования создатели сети заинтересованы в снижении объема обучающей вы-

борки до 10% от всех имеющихся данных. Кроме того, выборка должна быть статистически репрезентативна используемой базе данных или подчеркивать некоторые специфические признаки данных. Выборочная методология может быть использована для определения особенностей данных. Примером служит метод, который отбирает в выборку данные, находящиеся на «хвостах» распределений. При этом предполагается, что образы, представляющие интерес, распределены симметрично.

Идентификация выходных индикаторов. В зависимости от используемой парадигмы обучения необходимо выбрать один или более выходных индикаторов. При супервизорном обучении предполагается, что требуемый выход должен сравниваться с реальным выходом сети. Выбор выходных индикаторов обусловлен также типом решаемой задачи. В случае, если целью системы является предсказание будущих цен, уровней продаж и т. п., то выходами сети должны быть те же переменные. В системах прогнозирования важно выбрать такие выходные показатели, которые соответствуют временному окну, использованному для торговых операций. Например, если система торговли основана на ежедневной отчетности, то и предсказание должно быть сделано на этот же временной период.

В ситуации, если целью нейросетевой системы является распознавание образов или классификация данных, то на выходе сети может быть любое число индикаторов. Например, образы в исторических данных могут быть связаны с образами в будущих данных. Важной особенностью при выборе выходных индикаторов является способ, который выявляет указанные образы, например, при делении исходных данных на три класса на выходе сети должно быть три индикатора. Кроме того, при выборе индикаторов нужно учитывать способность нейронной сети к обучению: сеть с ограниченной емкостью (мало нейронов в скрытом слое) не может обучиться сотням различных предъявляемых ей образов.

Полезные выходные индикаторы могут быть разработаны на основе требуемого уровня прибыли, требуемой торговой стратегии (например, длинная или короткая рыночные позиции, уровень поворота кривой цен и т. п.). Однако в ситуации, когда разработчик не знает, чему должна обучаться нейронная сеть, маловероятно, что такая сеть может достичь нужных соотношений между входами и выходами.

Оценка разработанной нейронной сети. Для оценки качества обученной ИНС могут быть использованы следующие показатели:

– эффективность модели;

– эффективность работы сети;
– устойчивость;
– стабильность.

Эффективность модели определяет качество обученной сети, основываясь на характеристике, которая достигается при требуемом или идеальном выходах сети. Для вычисления эффективности модели вначале определяют оценку поведения сети при требуемом или идеальном выходе и сравнивают результат с тем, который получен на выходе сети при подаче на ее вход элементов обучающей выборки. Например, если годовой доход на основе идеального выхода сети составляет 20%, а нейросетевой реальный выход на тех же данных дает 16%, то эффективность модели равна 80%. В итоге, этот показатель является индикатором качества обучения ИНС на предъявленной ей выборке данных, но не демонстрирует ее возможности в реальной работе.

Эффективность работы сети (по-английски – walk-forward efficiency) является мерой потенциала работы обученной сети. Этот показатель дает оценку действия нейронной сети на тестовой выборке и сравнивает полученную оценку с той, которая достигается при использовании требуемого (идеального) выхода. Например, если на основе данных, не участвовавших в обучении, идеальный выход дает 14,5%, а реальный нейросетевой выход – 11,5%, то такой показатель составляет 11,5 / 14,5 = 72%. Эффективность работы сети – это индикатор потенциала обученной нейронной сети, однако он не гарантирует, что достигнутая характеристика будет поддерживаться. Режим работы или структурные изменения могут создать различие между обучающей, тестовой выборками и реализацией работы сети в реальном времени. Вследствие этого важно проверить устойчивость ИНС во времени.

Устойчивость – характеристика обученной сети в различные временные периоды, которыми могут быть перекрывающиеся или неперекрывающиеся окна изменяющихся размеров. Например, если доступны исторические данные за прошедшие несколько лет, то возможно оценить поведение нейронной сети для всех окон шириной 3, 6, 9 или 12 месяцев. Использование скользящих окон дает распределение значений характеристики ИНС. Такие распределения можно изучать методами классического статистического анализа, определяя средние величины, дисперсии и значимость полученных по распределениям оценок.

Стабильность – характеристика обученной нейронной сети, основанной на окнах переменных размеров. Основная идея сводится к

использованию метода Монте–Карло для оценки характеристики обученной сети на многих временных интервалах, каждый из которых имеет случайные даты начала и окончания. Единственное ограничение сводится к сохранению некоторой минимальной ширины окна, например один месяц.

1.10. Нейронные сети в задачах менеджмента

Рассмотрим типовые задачи менеджмента, которые могут решаться с использованием аппарата ИНС. Одной из целей книги является демонстрация менеджерам (и будущим, и сегодняшним) возможностей «интеллектуального инструментария». Здесь укажем только разновидности задач менеджмента, а способы их решения с применением программных продуктов – в следующем подразделе.

Как отмечалось выше, классификация и регрессия (прогнозирование) – основные области практического приложения нейронных сетей. Применительно к задачам менеджмента классификации могут подвергаться фирмы, предприятия, выпускаемая продукция, поставщики и т. п. С целью сохранения общности изложения все перечисленное можно назвать объектами классификации. Ко второй области практического приложения ИНС можно отнести задачи прогнозирования объемов выпуска продукции, объемов продаж, изменения курса акций, изменения обменных курсов и т. п., которые также с целью общности назовем показателем. Здесь для получения оценки прогноза необходимо иметь ретроспективный временной ряд изменения конкретного показателя. Рассмотрим несколько типовых задач.

Управление качеством [12]. В такой задаче производителя интересует качество выпускаемой продукции. Для оценки качества можно предложить систему контроля, построенную на принципах ИНС. В системе должна быть использована нейронная сеть прямого распространения, обучаемая методом ОРО. Для увеличения вычислительных возможностей сеть должна состоять из трех слоев: входного, скрытого и выходного.

Оценим число нейронов в каждом слое. Количество нейронов входного слоя должно соответствовать размерности входного вектора, например, если объект характеризуется пятью признаками, то входной слой содержит такое же число нейронов. В качестве первого приближения для скрытого слоя установим удвоенное количество нейронов (10). Число нейронов выходного слоя будет определяться тем количеством классов, на которые предполагается разделить всю совокупность объектов. Положим, что таких

классов – два (годные и бракованные изделия). Тогда на выходе сети будем иметь два выходных нейрона с кодировкой $(1\ 0)^т$ и $(0\ 1)^т$ (т – знак транспонирования). При большем количестве выходных классов (категорий продукции) используется кодировка вида 1 – из – N. Для уменьшения времени обучения пороги принятия и отвержения можно установить равными, соответственно, $(0{,}9\ 0{,}1)^т$ и $(0{,}1\ 0{,}9)^т$.

Для обучения сети методом ОРО необходимо иметь достаточное количество обучающих пар: входной вектор – требуемый вектор. В ряде случаев в распоряжении разработчика сети таких данных мало, что приводит к невозможности обучения сети до заданного уровня ошибки. В этой ситуации следует применить метод Монте–Карло для «размножения» недостающих данных. Обученную таким образом сеть можно использовать в качестве системы контроля качества продукции.

Оценка кредитоспособности заемщика [13]. При выдаче кредита юридическому или физическому лицу банк заинтересован в возврате выданных сумм, и кредит выдается лишь тем субъектам, которые, по мнению банка, вернут деньги. К сожалению, бывают ситуации, когда выданный кредит не возвращается, несмотря на казавшуюся надежность заемщика. В такой ситуации задача банка заключается в том, чтобы на основе кредитной истории прошлых заемщиков банка найти оценку возврата или невозврата кредита, т. е. классифицировать заемщика по степени его надежности. В зарубежной литературе такого рода задачи относятся к скоринг-анализу (от англ. слова to score – считать). Поставленная задача оценки надежности заемщика также решается с применением многослойного персептрона, состоящего, к примеру, из трех слоев: входного, скрытого и выходного. На входной слой подается вектор входных параметров, определяемый через известные банку финансово-экономические характеристики заемщика; число нейронов в скрытом слое подбирается разработчиком вначале интуитивно (по крайней мере, в два раза больше размерности входного вектора); количество нейронов выходного слоя определяется числом классов, на которые требуется разбить совокупность данных.

Для обучения нейронной сети необходимо иметь достаточную выборку примеров кредитной истории, из которой формируется обучающая и тестовая серии. Большей частью таких данных в распоряжении разработчика нет, поэтому следует воспользоваться методом статистического моделирования для «размножения» данных. Размерность входного вектора влияет на архитектуру ИНС, усложняя ее при увеличении числа компонентов, вследствие чего

следует провести предобработку данных с целью снижения составляющих этого вектора. Обученная, например, на два класса, нейронная сеть будет относить нового потенциального клиента банка к разрядам «хороших» или «плохих» заемщиков. Сотрудникам банка для принятия такого решения нужно на вход сети предъявить вектор исходных данных, характеризующих финансовое состояние клиента. Сеть, ориентируясь на значение выхода и установленный порог принятия решения, отнесет нового клиента к одному из классов.

С помощью этой же сети могут быть определены риски «перепутывания» классов, т. е. решена задача риск-менеджмента при предоставлении кредита банком. Для этого нужно иметь базу данных (реальных или разыгранных) хороших и плохих клиентов. Доля неправильно расклассифицированных клиентов даст количественную оценку риска.

Кластеризация объектов. Такая задача возникает при необходимости разделить объекты на ряд групп (кластеров). При этом менеджер не обладает достаточными сведениями об объектах, чтобы сформировать требуемый выходной вектор. В такой ситуации можно воспользоваться самообучающимися сетями Кохонена. Здесь нейронная сеть сама, ориентируясь лишь на структуру подаваемых входных векторов, будет относить очередной, предъявляемый на вход объект к определенному классу.

Разработчик сети в этом случае должен определить число нейронов во входном и выходном слоях (сеть Кохонена состоит только из двух слоев), скорость обучения и критерий остановки. Количество нейронов входного слоя, как и в предыдущих задачах, устанавливается равным размерности вектора признаков объекта. В выходном слое число нейронов определяется количеством классов, на которые предполагается разделить анализируемую совокупность объектов. Скорость обучения обычно выбирается из диапазона 0,5–1, а ее уменьшение за время обучения до нуля определяет критерий остановки. Например, при выбранной начальной скорости обучения, равной 0,5, и шаге изменения скорости, составляющем 0,05, потребуется всего 10 эпох для формирования нейрона – победителя в выходном слое.

Победивший среди всех нейронов выходного слоя (тот нейрон, у которого весовой вектор наиболее близок к входному вектору объекта) определяет, в сущности, метку класса, к которому принадлежит предъявленный объект. Затем на вход сети поступает вектор признаков следующего объекта, и сеть определяет его принадлежность: если

этот объект похож на предыдущий, то сеть относит его к тому же классу; в противном случае – указывает метку другого кластера.

Одним из свойств сетей Кохонена является возможность сжатия информации, поскольку в один нейрон-победитель могут входить несколько объектов. Вследствие этого типичными задачами, решаемыми с помощью сетей Кохонена, являются разбиение совокупности предприятий, фирм на ряд классов, формирование рейтингов банков (в каждый кластер попадают несколько схожих банков), составление базы данных по продаже объектов недвижимости (квартиры разной площади, разные районы, типы домов определяют различные ценовые кластеры).

Прогнозирование изменения показателей. Подобная задача типична для деятельности менеджера в ситуациях, когда нужно оценить потребность в той или иной продукции в следующий временной промежуток, сформировать требования к запасам комплектующих деталей, выяснить характер изменения спроса на выпускаемую продукцию. Нейросетевая технология при построении прогнозных оценок здесь представляется вполне приемлемым подходом. Однако не следует забывать о существовании традиционных статистических методов прогноза и дополнять нейросетевой прогноз подходящей статистической моделью, например, класса ARIMA.

Как обычно, при разработке ИНС начнем с ее архитектуры. Метод скользящих окон, используемый при нейросетевом прогнозе, однозначным образом определяет число нейронов в каждом слое: входной слой содержит число нейронов, равное ширине первого окна; скрытый слой – количество нейронов в первом приближении, равное удвоенной ширине окна; в выходном слое находится только один нейрон, который определяет ширину второго окна. Метод обучения сети (ОРО) требует наличия обучающей пары вход – требуемый выход. Такая пара формируется из значений ряда, попавших в первое окно (вход сети), и единственное значение ряда, взятое через некоторый заранее установленный сдвиг по временной оси (требуемый выход).

Система из двух окон, передвигаясь («скользя») вдоль временной оси, обучает нейронную сеть до тех пор, пока правое окно не выйдет за пределы имеющегося временного ряда. При этом образуется так называемая проекция временного ряда, полученная как прогноз на наблюдаемом участке. На последнем, таким образом, имеем две кривых: исходный временной ряд и его проекция как результат нейросетевого прогноза. За пределами интервала наблюдения получаем только результат прогноза, выдаваемый обучен-

ной нейронной сетью. Поскольку число значений ряда в первом окне уменьшается при пересечении правой стороной окна границы интервала наблюдения, то для сохранения архитектуры сети и дальнейшего прогноза полученные очередные прогнозные величины ряда подаются на вход сети.

Примеры решения задач

Перед тем, как перейти к описанию конкретных задач менеджмента с помощью нейросетевой технологии, укажем некоторые программные пакеты в этой области, обеспечивающие возможности решения.

1. Пакет `BrainMaker` производства американской компании California Scientific Software появился на Западе в конце 80-х гг. XX в. и вскоре стал одним из лидеров продаж. В 1990 г. этот пакет получил приз в номинации «Лучший программный продукт года», а впоследствии стал самым продаваемым в США нейропакетом [14]. Отметим, что вначале пакет был разработан по заказу NASA, а затем адаптирован для коммерческих приложений. Пакет получил широкое распространение во многих промышленных и финансовых компаниях, в оборонных предприятиях США для решения задач прогнозирования, оптимизации и моделирования ситуаций.

Пакет `BrainMaker` состоит из двух компонентов: нейросетевого ядра, которое обучает и тестирует нейронные сети, и среды для создания нейросетей, анализа и подготовки исходных данных. Пакет реализует только одну парадигму: обучение с учителем методом обратного распространения ошибки, но этого оказывается достаточно для большинства приложений в области классификации и прогнозирования.

2. Пакет `AI Trilogy` американской фирмы Ward Systems Group, представляющий собой набор из трех самостоятельных приложений: `NeuroShell II`, `Neuro Windows`, `GeneHunter`. `NeuroShell II` – это средство создания, обучения и тестирования нейросетевых приложений, `Neuro Windows` – нейросетевая библиотека в исходных текстах, `GeneHunter` – система оптимизации сетей на основе генетических алгоритмов. Вместе эти компоненты образуют мощный «конструктор», позволяющий строить аналитические комплексы весьма большой сложности. Этот пакет также адаптирован для бизнес-приложений и используется в более, чем 150 крупнейших банках США. В пакете `AI Trilogy` имеются возможности обработки текстовых данных, задания правил в явном виде, работы с финансовыми индикаторами и обработки циклических событий.

3. Пакет Statistica Neural Networks относится к современным нейросетевым продуктам и поэтому более совершенен по сравнению с ранее выпущенными. Данный пакет – это универсальный пакет нейросетевого анализа американской фирмы Statsoft. Он имеет мощные алгоритмы обучения сети (включая методы сопряженных градиентов и Левенберга–Маркара), возможность создания сложных комбинаций из сетей различных архитектур. Для этого пакета характерны простота в использовании и аналитические мощности, например Automatic Network Desingner (автоматический конструктор сети) определит наилучшую архитектуру для конкретной задачи, осуществит отбор переменных.

Отметим, что сегодня нейрокомпьютерный сегмент рынка программного обеспечения бурно развивается, и появление новых пакетов представляет собой естественный процесс. Большинство фирм, работающих на рынке аналитических программ, уже заявили о подготовке или выпуске систем на основе нейросетевой технологии. У современных нейропродуктов улучшен интерфейс пользователя, включены дополнительные алгоритмы обучения сетей, реализованы возможности взаимодействия с другими приложениями. Однако плата за эти возможности – необходимость применения более мощного компьютера, рост требований к объему оперативной и дисковой памяти. Необходимо также учесть, что новый пакет вовсе не гарантирует более качественного решения пользовательской задачи.

Проиллюстрируем возможности пакета Statistica Neural Networks для решения некоторых задач. Задачу классификации рассмотрим на примере разделения совокупности предприятий на два класса: «банкроты» и «небанкроты». Такая задача решалась под руководством автора при подготовке магистерской диссертации С. А. Смирновой. Следует отметить, что механическое применение методик прогнозирования банкротства, принятых за рубежом, не годится для нашей страны, так как у нас иные темпы инфляции, другие циклы макро- и микроэкономики, отличающиеся уровни фондо-, энерго- и трудоемкости производства, производительности труда, налоговое бремя. Набор признаков, выбранных для указанной классификации предприятий, представлял совокупность четырех показателей финансовой устойчивости организации и двух показателей его платежеспособности. Первая группа показателей характеризовала финансовую устойчивость, отражающую сбалансированность денежных и товарных потоков, доходов и расходов, средств и источников их формирования. Описание используемых показателей финансовой устойчивости представлено в табл. 1.1.

Таблица 1.1

№ п/п	Коэффициент	Характеристика	Расчет
1	Коэффициент автономии ($K_а$)	Указывает на независимость от заемных средств, на долю собственных средств в общей сумме всех средств	Отношение общей суммы всех средств предприятия к источникам собственных средств
2	Коэффициент соотношения заемных и собственных средств ($K_{з.с}$)	Показывает величину заемных средств, привлеченных предприятием на 1 р. вложенных в активы собственных средств	Отношение всех обязательств к собственным средствам
3	Коэффициент маневренности ($K_м$)	Оценивает способность предприятия поддерживать уровень собственного оборотного капитала и пополнять оборотные средства за счет собственных источников	Отношение собственных оборотных средств к общей величине собственных средств предприятия
4	Коэффициент имущества производственного назначения ($K_{п.им}$)	Определяет долю имущества производственного назначения в общей стоимости всех средств предприятия	Отношение суммы внеоборотных активов и производственных запасов к итогу баланса

К показателям платежеспособности предприятий относятся коэффициенты текущей ликвидности и обеспеченности собственными средствами.

Коэффициент текущей ликвидности K_1 характеризует общую обеспеченность предприятия оборотными средствами для ведения хозяйственной деятельности и своевременного погашения срочных обязательств предприятия. Коэффициент K_1 определяется как отношение фактической стоимости, находящихся в наличии у предприятия оборотных средств к наиболее срочным обязательствам предприятия в виде краткосрочных кредитов банков, краткосрочных займов и различных кредиторских задолженностей.

Коэффициент обеспеченности собственными средствами K_2 характеризует наличие собственных оборотных средств у предприятия, необходимых для его финансовой устойчивости. Коэффициент K_2 определяется как отношение разности между объемами источников собственных средств и фактической стоимостью основных средств к фак-

тической стоимости находящихся в наличии у предприятия оборотных средств.

Показатели платежеспособности наравне с показателями финансовой устойчивости участвуют в классификации предприятий на две группы: «банкротов» и «небанкротов». Кроме того, коэффициенты K_1, K_2 выступают как самостоятельные критерии предсказания возможного банкротства предприятия путем прогнозирования их значений на будущие периоды на основании известных значений за ряд предшествующих периодов. На базе полученных спрогнозированных значений данных показателей на последующие промежутки времени можно сделать вывод о надвигающейся угрозе банкротства или, наоборот, о стабильном финансовом состоянии организации на ближайшее время.

В качестве сети для классификации выбран многослойный персептрон с алгоритмом обучения в виде метода ОРО. Входными данными для классификации предприятий на две группы «банкрот» – «небанкрот» являлись значения шести показателей платежеспособности и финансовой устойчивости предприятия, т. е. входной слой сети состоял из шести элементов. Выходным параметром служит бинарная номинальная переменная, соответствующая двум классам: «банкрот» и «небанкрот», к которым в результате обучения сеть должна научиться относить новые предприятия.

Набор данных с помощью метода статистических испытаний сформирован из 600 наблюдений – предприятий, для которых известны значения всех шести входных параметров и принадлежность к одному из заданных классов. При этом 300 наблюдений соответствует предприятиям-банкротам, а 300 – финансово-устойчивым предприятиям. Для последующего анализа результатов и качества работы сети набор данных следует разделить на два подмножества – обучающее и тестовое. Обучающее подмножество состоит из двухсот наблюдений, соответствующих фирмам-банкротам, и такого же числа для финансово-устойчивых фирм. Тестовое подмножество включает в себя по 100 примеров, принадлежащих обоим классам.

Доверительные уровни установлены достаточно жесткими: минимальное значение выхода, при котором наблюдение считается принадлежащим «положительному» классу фирм-банкротов, установлено равным 0,95 (порог принятия); максимальное значение выхода, при котором наблюдение относится к классу финансово-устойчивых организаций, соответствует 0,05 (порог отвержения).

Критериями выбора наиболее подходящей сети, демонстрирующей лучшие результаты классификации, выступают величина контрольной ошибки по обучающему и тестовому подмножествам и

итоговые статистики окна «Статистики классификации» (Classification Statistics) для тестового подмножества. Лучшие результаты продемонстрировала сеть, скрытый слой которой включал семь элементов, т. е. архитектура нейронной сети имела вид: 6–7–1 (рис. 1.19).

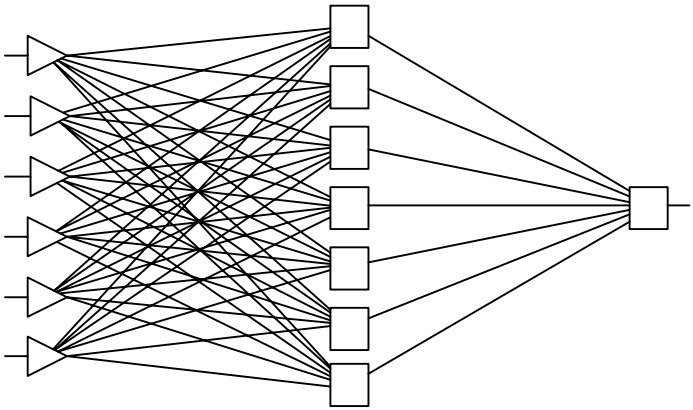


Рис. 1.19

Величина контрольной ошибки составляет 0,016 для обучающего подмножества и 0,018 – для тестового. Сеть демонстрирует самое большое среди других сетей количество правильно классифицированных наблюдений как для обучающего, так и для тестового множеств.

Результаты, полученные с помощью сети 6–7–1, приведены в табл. 1.2.

Таблица 1.2

Архитектура сети	Величина контрольной ошибки (среднеквадратическая ошибка по обучающему и тестовому подмножествам)	Число классифицированных наблюдений на тестовом подмножестве (действительный класс наблюдения равен классу, присвоенному ИНС)					
		небанкроты = небанкроты	небанкроты = банкроты	банкроты = банкроты	банкроты = небанкроты	небанкроты = не определен	банкроты = не определен
6–7–1	0,016; 0,018	99	–	96	–	1	4

Из табл. 1.2 следует, что нейронная сеть решила задачу классификации предприятий на две группы: потенциальных банкротов и финансово-устойчивых предприятий. Из ста фирм, потенциально являвшихся банкротами, сеть точно классифицировала 96; из такого же количества финансово-устойчивых предприятий 99 были определены правильно. Ни одного наблюдения сеть не классифицировала неправильно, четыре фирмы-банкрота и одну фирму-небанкрот сеть не классифицировала, что может быть отнесено за счет установления слишком жестких доверительных уровней.

На рис. 1.20 приведен график ошибки обучения (`Training Error Graph`) данной сети.

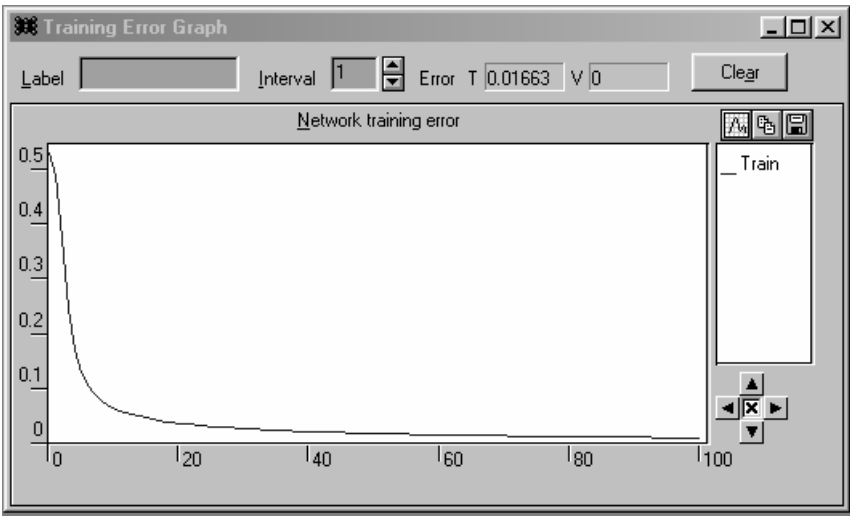


Рис. 1.20

При анализе результатов и качества классификации, а также выборе порогов принятия/отвержения очень помогает информация, содержащаяся в окне «Статистики классификации» (`Classification Statistics`), показанном на рис. 1.21.

Столбцы таблицы соответствуют классам, каждый столбец разбит на две части: суммарные статистики и статистики присваивания. Все данные отображаются отдельно для каждого множества (обучающего или тестового).

Суммарные статистики включают: общее число наблюдений в этом наборе данных; количество наблюдений из этого класса, которые сеть классифицировала правильно; количество наблюдений из этого класса, которые сеть классифицировала неправильно как относящиеся к

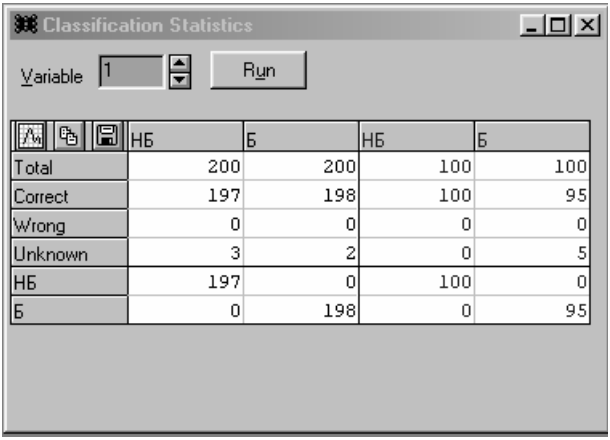


Рис. 1.21

другому классу; количество наблюдений из этого класса, которые сеть не смогла классифицировать. Статистики присваивания показывают, сколько наблюдений из каждого набора было отнесено к каждому из возможных классов, включая правильный (неясные наблюдения не отражены). Из приведенной таблицы следует вывод о высоком качестве обучения и работы нейронной сети.

Полученная нейросетевая модель имеет своей целью классификацию новых фирм, для которых известны значения шести указанных выше показателей. Работа модели была проверена на тестовом множестве данных, которые не участвовали в обучении сети, и показала хороший результат, который свидетельствует о широких возможностях практического применения данного подхода.

Воспользуемся этим же примером для построения прогнозной оценки с помощью нейронной сети. В качестве прогнозируемых параметров примем коэффициенты K_1, K_2, значения которых за период наблюдения с 1997–2001 гг. приведены в табл. 1.3.

В качестве обучающего набора данных были взяты первые 18 значений показателей платежеспособности за период с I квартала 1997 г. по II квартал 2001 года, контрольными данными являются значения показателей за III и IV кварталы 2001 года.

В ходе исследования были рассмотрены различные модификации архитектуры многослойных персептронов (параметры *Временное окно – Steps* и *Горизонт – Lookahead*, количество элементов в скрытых слоях), варианты настройки обучения (скорость обучения, инерция, количество эпох). В качестве алгоритма обучения сетей применялся метод ОРО. Для улучшения качества экстраполяции логисти-

Таблица 1.3

№ п/п	Отчетный период (квартал, год)	Коэффициент текущей ликвидности (K_1)	Коэффициент обеспеченности собственными средствами (K_2)
1	I 1997	2,58	0,61
2	II 1997	2,55	0,6
3	III 1997	2,65	0,68
4	IV 1997	2,59	0,63
5	I 1998	2,63	0,66
6	II 1998	2,51	0,56
7	III 1998	1,46	0,01
8	IV 1998	1,53	0,09
9	I 1999	1,86	0,24
10	II 1999	1,77	0,19
11	III 1999	1,87	0,24
12	IV 1999	1,92	0,27
13	I 2000	1,98	0,30
14	II 2000	2,12	0,38
15	III 2000	2,26	0,44
16	IV 2000	2,18	0,41
17	I 2001	2,27	0,45
18	II 2001	2,35	0,49
19	III 2001	2,33	0,48
20	IV 2001	2,38	0,51

ческая функция активации в выходном слое была заменена на линейную, которая не меняет уровня активации и при этом, в отличие от логистической, не насыщается, поэтому способна лучше прогнозировать. При поиске лучшего варианта сети применялись возможности автоматического конструктора сети, который реализован в пакете `Statistica Neural Networks`. Для решения задачи прогнозирования показателя текущей ликвидности K_1 лучшим оказался вариант сети с пятью входными ячейками, одним скрытым слоем, состоящим из десяти нейронов, и одной экстраполирующей ячейкой, что привело к архитектуре сети вида 5–10–1.

Величина контрольной ошибки составляет 0,18 для обучающего подмножества и 0,21 – для контрольного. С помощью обученной сети можно выполнить проекцию временного ряда показателя текущей ликвидности (рис. 1.22).

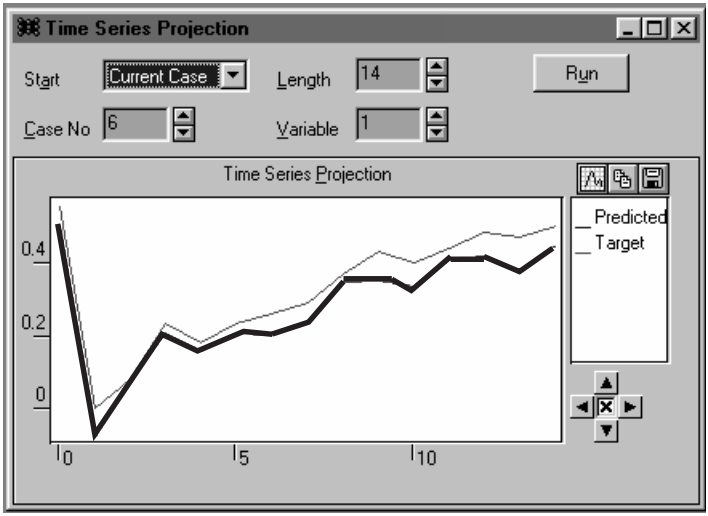


Рис. 1.22

График показывает, что прогнозируемый ряд достаточно хорошо аппроксимирует исходную последовательность и, как следствие, дает достаточно надежную оценку прогноза на два периода (19 и 20) вперед, приведенную в табл. 1.4.

Таблица 1.4

Период прогноза (квартал, год)	Значение показателей	
	K_1	K_2
III 2001	2,17	0,42
IV 2001	2,21	0,47

В целом, прогнозирование с использованием нейросетевой технологии, учитывая недостаточно большую протяженность наблюдаемого временного ряда, показало неплохие результаты по сравнению с реальными значениями этих коэффициентов.

Продемонстрируем возможности этого пакета для решения задачи кластеризации с помощью сети Кохонена. Источником информа-

ции являлись значения параметров 20 банков Российской Федерации, приведенные в журнале «Эксперт» за 2001 г., № 3 (см. табл. 1.5). Показатели являются реальными, выраженными в миллионах рублей. Данная задача решалась при выполнении магистерской диссертации А. М. Дашевской под руководством автора.

Таблица 1.5

Банки	Активы	Собственный капитал	Балансовая прибыль	Портфель коммерческих кредитов
Сбербанк РФ	550,7	40,7	12,1	221,6
Внешторгбанк	97,6	22,9	0,04	26,2
Газпромбанк	70,1	18,5	0,46	36,5
Альфа-банк	67,7	12,3	0,7	20,4
Международный промышленный банк	64,1	29,9	0,38	44,7
Сургутнефтегазбанк	4,6	0,6	0,04	15,8
Доверительный и инвестиционный банк	39,3	2,7	0,76	6,04
Росбанк	34,2	4,4	1,1	11,7
Банк Москвы	33,6	1,9	0,16	12,6
Ситибанк	26,4	2,9	1,6	13,5
Башкредитбанк	23,9	3,5	0,49	9,9
МДМ-банк	22,4	4,4	0,24	7,7
Промстройбанк	19,7	1,3	0,5	9,8
Глобэкс	17,9	7,9	0,005	7,5
Менатеп, СПб	17,8	1,6	0,3	6,8
Райффайзербанк, Австрия	16,1	1,1	0,05	5,5
Национальный резервный банк	14,9	3,6	0,12	4,7
Еврофинанс	15	1,9	0,3	4,5
Автобанк	14,7	1,6	0,08	7,8
Гута-банк	13,0	2,3	0,1	4,9

В сетях Кохонена, которые используются для решение поставленной задачи, происходит неуправляемое обучение (без учителя): сеть учится распознавать кластеры среди неразмеченных данных,

содержащих только входные значения. Кроме этого, сеть Кохонена располагает родственные кластеры поблизости друг от друга в выходном слое, формируя так называемую топологическую карту (*Topological Map*).

Работа алгоритма определяется двумя параметрами: скорость обучения – *Learning rate* и окрестность – *Neighborhood*. Процесс обучения сети происходит следующим образом: очередное наблюдение подается на вход сети, обрабатывается ею, выбирается выигравший (наиболее активный элемент второго слоя сети), и затем он и его ближайшие соседи корректируются так, чтобы лучше воспроизводить обучающее наблюдение.

Обычно работа алгоритма разбивается на два этапа: упорядочивания и тонкой настройки, на каждом из которых скорость обучения, размер окрестности постепенно меняются от своих начальных значений к конечным. В пакете «Statistica NN» можно задавать начальные и конечные значения как для скорости обучения, так и для размера окрестности. Размер окрестности определяет квадрат с центром в выигравшем элементе: «нулевой» размер соответствует одному выигравшему элементу.

Сеть Кохонена в данном пакете имеет специальные средства для решения задач кластеризации, к которым можно отнести:

– окно «Частота выигрышей» – «*Win Frequencies*», в котором показывается, где в сети формируются кластеры;

– окно «Топологическая карта» – «*Topological Map*», которое показывает, какие наблюдения отнесены к тому или иному кластеру, и помогает пользователю правильно пометить элементы и наблюдения.

Алгоритм обучения сети Кохонена корректирует положения центров в слое топологической карты таким образом, чтобы приблизить их к центрам кластеров в обучающих данных. На каждом шаге обучения алгоритм выбирает элемент, чей центр лежит ближе всего к обучающему наблюдению. Этот элемент и соседние с ним корректируются так, чтобы они больше походили на данное обучающее наблюдение.

Исключительную роль в обучении Кохонена играет окрестность обучения. Корректируя не только сам выигравший элемент, но и соседние с ним, алгоритм Кохонена относит близкие наборы данных к смежным областям топологической карты. По ходу обучения окрестность постепенно сужается, и одновременно уменьшается скорость обучения, так что поначалу выстраивается грубое отображение (при котором на одно наблюдение откликаются большие

группы элементов), а на последующих этапах достраиваются более тонкие детали (отдельные элементы в группах реагируют на небольшие различия в данных).

После обучения сети можно рассмотреть полученные результаты по кластеризации банков. Выбранная сеть имела архитектуру 6–4, т. е. во входном слое – шесть нейронов в соответствии с числом признаков и в выходном – четыре нейрона, что предполагало разделение всей совокупности на четыре группы. Схема сети Кохонена показана на рис. 1.23.

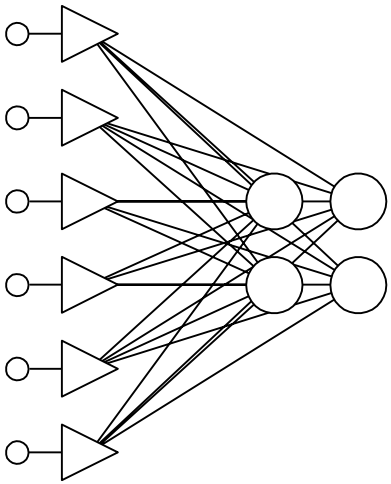


Рис. 1.23

Далее обратимся к диалоговому окну «Частота выигрышей», где можно наблюдать за тем, как на топологической карте формируются кластеры. Это окно «прогоняет» сеть по всем наблюдениям из обучающего множества и подсчитывает, сколько раз каждый элемент выигрывал (т. е. оказывался ближайшим к обрабатываемому наблюдению). Большие значения частоты выигрышей указывают на центры кластеров в топологической карте.

После того, как зафиксировано распределение центров кластеров, средствами окна «*Топологическая карта*» (рис. 1.24) можно протестировать сеть для выяснения смысла кластеров. В этом окне при обработке очередного наблюдения каждый элемент показывает степень своей близости к нему с помощью сплошного черного квадрата (чем больше размер квадрата, тем больше степень близости), а выигравший элемент помещается в квадратную рамку (рис. 1.24).

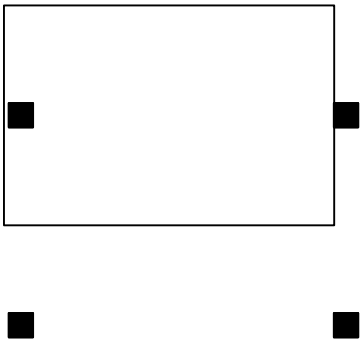


Рис. 1.24

После обучения помечаем элементы значками соответствующих им классов. Таким образом, можно сформулировать алгоритм действий, который необходимо осуществить для решения задачи кластеризации банков:

– обучить сеть Кохонена;

– идентифицировать кластеры в окне «Топологическая карта» средствами окна «Частота выигрышей»;

– пометить кластеры в топологической карте;

– протестировать наблюдения в окне «Топологическая карта» и присвоить наблюдениям имена кластеров;

– сформировать полученные кластеры.

Таким образом, были получены следующие результаты по кластеризации банков с помощью сети Кохонена (табл. 1.6).

Таблица 1.6

№ кластера	Название банка
1	Сберегательный банк РФ
2	Международный промышленный банк
3	СитибанкРосбанкГазпромбанк
4	Внешторгбанк Газпромбанк Альфа-банк Башкредитбанк Глобэкс Доверительный и инвестиционный банк Банк Москвы МДМ – Банк Менатеп, СПб Промышленный строительный банк Сургутнефтегазбанк Национальный резервный банк Автобанк Еврофинанс Райффайзербанк Гута-банк, Австрия

Как видно из табл. 1.6, наиболее крупные банки занимают отдельные кластеры; три банка, менее значимые по сравнению с первы-

ми двумя, формируют третий кластер; все остальные входят в четвертый кластер.

Подводя итоги изложенному в первой главе, отметим, что нейронные сети являются мощным средством оказания помощи в решении задач менеджмента. В частности, задачи классификации и прогнозирования, наиболее широко применяемые в сфере менеджмента, могут решаться посредством нейросетевой технологии. Увеличение числа менеджеров, свободно разбирающихся в концепции ИНС, является насущной необходимостью, поскольку знание таких технологий свидетельствует об уровне квалификации менеджера.

Библиографический список

1. *Cichoki A., Unbehauen R.* Neural Netwokrs for Optimization and Signal Processing. Chichester: J.Wiley & Sons, 1993.

2. *Мак-Каллок У. С., Питтс В.* Логическое исчисление идей, относящихся к нервной активности // Автоматы / Под ред. *К. Э. Шеннона, Дж. Маккарти*. М.: ИИЛ, 1956. С. 362–401.

3. *Розенблатт Ф.* Принципы нейродинамики. Персептроны и теория механизмов мозга. М.: Мир, 1965.

4. *Минский М., Пайперт И.* Персептроны. М.: Мир, 1971.

5. *Hopfield J. J.* Neural Networks and Physical Systems with Emergent Collective Computational Abilities// Proc. Nath. Acad. Sci. USA. 1982. Vol.79. 8. P. 2554 – 2558.

6. *Rumelhart D. E., Hinton G. E., Williams R. J.* Learning Internal Representations by Error Propagation // Parallel Distributed Processing/ Ed. D. E. Rumerhart, J. L. McClelland. Vol.1. Chap. 8. Cambridge: MIT Press, 1986.

7. *Hertz J. A., Krogh A. S., Palmer R. G.* Introduction to the Theory of Neural Computation. N.Y.: Addison-Wesley, 1991.

8. *Fausett L.* Fundamentals of Neural Networks. New Jersey: Prentice-Hall, 1994.

9. *Осовский С.* Нейронные сети для обработки информации. М.: Финансы и статистика, 2002.

10. *Kohonen T.* Self-Organization and Associative Memory. Berlun: Springer–Verlag, 1989.

11. *Klimasauskas C.* Neural Networks Techniques. / in Trading on the Edge / Ed. G.J.Deboeck. N.-Y.: J.Wiley&Sons, 1994. С. 3–26.

12. *Кричевский М. Л.* Введение в искусственные нейронные сети: Учеб. пособие/ СПб., 1999.

13. *Кричевский М. Л.* Нейронные сети в задачах риск-менеджмента // Науч. сессия проф.-преп. состава Гос. ун-та экон. и финансов: Сб. докл. СПб.: Изд-во СПбГУЭФ, 2002.

14. Базы данных. Интеллектуальная обработка информации/ *В. В. Корнеев, А. Ф. Гарев, С. В. Васютин* и *др*. М.: Нолидж, 2000.

ГЛАВА 2. НЕЧЕТКАЯ ЛОГИКА

Данная глава пособия посвящена нечеткой логике и ее использованию в задачах менеджмента. Основная идея НЛ состоит в том, что интеллектуальный способ рассуждений, опирающийся на естественный язык общения человека, не может быть описан в рамках традиционных математических формул. Формальному подходу присуща строгая однозначность интерпретации, а все, что связано с применением естественного языка, имеет многозначную интерпретацию. Например, говоря о температуре воды, мы используем термины: теплая или горячая, не определяя точно значение температуры. Основатель современной концепции нечеткой логики профессор Л. Заде (L. Zaden) построил новую математическую дисциплину, в основе которой лежит не классическая теория множеств, а теория нечетких множеств. Последовательно проводя идею нечеткости, можно описать нечеткие аналоги всех основных математических понятий и создать аппарат для моделирования человеческих рассуждений и способов решения задач. Применение термина «нечеткий» в математической теории может ввести в заблуждение, поэтому более точным названием этой дисциплины было бы «непрерывная логика». Вначале сочетание «Fuzzy Logic» переводилось на русский язык как «размытая логика» и только позже установился принятый сейчас термин «нечеткая логика». Используемый в этой дисциплине аппарат строг и точен, как и в классической теории множеств, но наряду со значениями «истина» и «ложь», принятыми в классической теории множеств, дает возможность оперировать промежуточными значениями.

2.1. Возникновение нечеткой логики

В развитии и становлении нечеткой логики можно выделить три этапа [1].

Логические парадоксы и принцип неопределенности В. Гейзенберга привели на первом этапе (1920–1930 гг.) к понятию многозначной логики. За открытие принципа неопределенности в 1932 г. этот выдающийся немецкий физик был удостоен Нобелевской премии. Были введены значения TRUE и FALSE как два предельных случая полного спектра неопределенности. Квантовые теории позволили включать среднюю часть истинной величины в бивалентную логическую основу. В конце 30-х гг. польский математик Я. Лукашевич (J. Lukasiewicz) первым формально разработал трехзначную логическую систему и затем расширил диапазон истинных величин до всех вещественных чисел в интервале [0,1].

Математики, которые использовали обобщенную истинную функцию t вида: {утверждение} $\rightarrow$ [0,1], определяли такой вид анализа как непрерывную или нечеткую логику. Несколько позже английский ученый М. Блэк (M. Black) применил непрерывную логику к множествам элементов или символов. Исторически он определил первую функцию принадлежности, которая в разработанной Л. Заде НЛ является краеугольным камнем при переходе от четких высказываний к нечеткому описанию.

С опубликования в 1965 г. работы «Fuzzy sets» («Нечеткие множества») американским ученым Л. Заде начался в т о р о й э т а п развития НЛ. Л. Заде сформулировал принцип несовместимости: чем сложнее система, тем менее мы способны дать точные и в то же время имеющие практические суждения об ее поведении. Для систем, сложность которых превосходит некоторый пороговый уровень, точность и практический смысл становятся почти исключающими друг друга характеристиками. По словам Л. Заде, «в большинстве основных задач, решаемых человеком, не требуется высокая точность. Человеческий мозг использует допустимость такой неточности, кодируя информацию, достаточную для задачи, элементами нечетких множеств, которые лишь приближенно описывают исходные данные. Поток информации, поступающей в мозг через органы зрения, слуха, осязания и др., суживается, таким образом, в тонкую струйку информации, необходимую для решения поставленной задачи с минимальной степенью точности. Способность оперировать нечеткими множествами и вытекающая из нее способность оценивать информацию является одним из наиболее ценных качеств человеческого разума, которое фундаментальным образом отличает человеческий разум от так называемого машинного разума, приписываемого существующим вычислительным машинам» [1].

Сущность НЛ, предложенной Л. Заде, сводится к следующим моментам:

– в ней используются лингвистические переменные (вместо обычных числовых) или в дополнение к ним;

– простые отношения между переменными описываются с помощью нечетких высказываний;

– сложные отношения определяются нечеткими алгоритмами.

Начало т р е т ь е г о э т а п а , хронологически начавшегося в середине 80-х гг. XX в., характеризовалось появлением коммерческих программных продуктов в области НЛ. Наиболее ярким событием стала состоявшаяся в 1987 г. международная конференция по системам на основе НЛ, на которой был продемонстрирован японский контроллер. Он был использован для управления скоростью электропо-

ездов в японском городе Сендай, и некоторые характеристики этой системы остаются недостижимыми для железнодорожников многих стран, которые не используют системы с НЛ.

Совершенно естественно, что мимо такого перспективного инструмента, как НЛ, не могли пройти военные: в начале 80-х гг. XX в. в Японии и США в обстановке глубокой секретности были развернуты комплексные работы по использованию нечеткой логики в различных оборонных проектах. Одним из самых впечатляющих результатов стало создание управляющего микропроцессора на основе НЛ, способного автоматически решать известную «задачу о собаке, догоняющей кота». Разумеется, в роли кота выступала межконтинентальная ракета противника, а в роли собаки – мобильная ракета, слишком легкая для установки на нее громоздкой традиционной системы управления. Задача о коте и собаке с той поры относится к разряду классических, обошла все учебные пособия и пакеты по НЛ.

Однако основные результаты использования НЛ в прикладных задачах были получены не военными, а промышленниками (не в США, а в Японии). Изобретенная и разработанная в США НЛ начала свой триумфальный путь на массовый рынок в далекой стране. Такое, впрочем, случалось и ранее (например, с технологиями плоских экранов для портативных компьютеров), однако обычно это было связано с непомерными по американским меркам долгосрочными инвестициями. Японцы довели практическое воплощение НЛ до совершенства. Можно много рассказывать об автоматических прокатных станах, интеллектуальных складах и «безлюдных производствах», созданных с использованием НЛ. Однако, пожалуй, более впечатляющим выглядит применение НЛ в товарах массового рынка – пылесосах, видеокамерах, микроволновых печах и т. п. В 1991 г. Япония экспортировала в общей сложности более чем на 25 млрд долларов товаров, в которых тем или иным образом использованы компоненты НЛ.

Рассмотрим кратко сущность НЛ, которая представляет удобный путь отображения входного пространства в выходное (рис. 2.1).

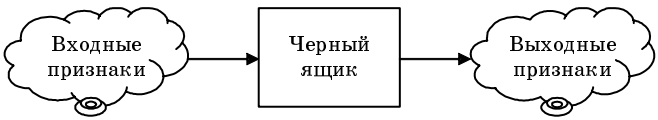


Рис. 2.1

В качестве черного ящика принципиально могут быть различные системы, в частности: нейронные сети, экспертные системы, дифференциальные уравнения и т. д. В нашем случае роль черного ящика выполняет НЛ.

В основе НЛ, отображающей входное пространство в выходное, используется механизм этого отображения в виде набора правил вида: *if – then* (*если – то*). Пример: если вода горячая, то нужно завернуть кран горячей воды.

Все правила оцениваются параллельно, их порядок не важен. Перед тем, как строить систему, описываемую правилами, необходимо определить все члены, которые будут использоваться в системе, и прилагательные для их описания (например, вода может быть холодная, горячая, теплая и т. п.).

Диаграмма, приведенная на рис. 2.2, поясняет работу системы с НЛ.

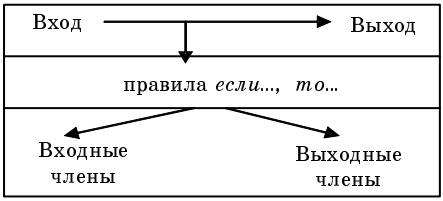


Рис. 2.2

С помощью этой диаграммы можно определить НЛ как метод, который интерпретирует значения выходного вектора и, основываясь на наборе правил, присваивает значения выходному вектору.

Укажем причины, на основании которых отдают предпочтение применению систем с НЛ:

– концептуально легче для понимания;

– гибкая система и устойчива к неточным входным данным;

– может моделировать нелинейные функции произвольной сложности;

– учитывается опыт специалистов-экспертов;

– основана на естественном языке человеческого общения.

Последнее – самое важное. Естественный язык сформирован на протяжении тысячелетий. Предложения, написанные обычным языком, представляют собой триумф эффективности коммуникаций. Вследствие того, что НЛ построена на конструкциях качественного описания, используемых в повседневной жизни, НЛ легка для применения.

2.2. Нечеткие множества

Под нечетким множеством понимается множество без четких, определенных границ. Оно может содержать элементы только с частичной степенью принадлежности. Для лучшего понимания различия между четкими и нечеткими множествами проведем их сравнение на уровне определений.

Пусть E – универсальное множество, x – элемент множества E, а R – некоторое свойство. Четкое подмножество A универсального множества E, элементы которого удовлетворяют свойству R, определяется как множество упорядоченных пар $A = \{\mu_A(x)/x\}$, где $\mu_A(x)$ – *характеристическая функция принадлежности* (или просто функция принадлежности), принимающая значения в некотором множестве M (например, $M = [0,1]$). Функция принадлежности (ФП) указывает степень (или уровень) принадлежности элемента x подмножеству A. Нечеткое подмножество отличается от обычного тем, что для элементов x из E нет однозначного ответа *"да–нет"* относительно свойства R.

Пример 2.1. Объект исследования представляет множество взрослых людей.

Обозначим через x возраст человека и введем функцию:

$$\mu(x) = \begin{cases} 1, \text{ если } x \geq 18 \\ 0, \text{ если } x < 18. \end{cases}$$

В этой записи учтено, что взрослым считается человек, достигший 18 лет. Следовательно, множество взрослых людей может быть задано в виде:

$$A = \{x \mid \mu(x) = 1\}, \; x \in X,$$

где X – множество возможных значений x.

Последнее равенство гласит: множество A образуют такие «объекты», для которых ФП $\mu(x) = 1$ (рис. 2.3).

Однако такая двузначная логика («*да–нет*») не учитывает возможного разброса мнений относительно границ множества A. Более естественна форма представления, показанная на рис. 2.4, где ФП ставит в соответствие каждому элементу $x \in X$ число $\mu(x)$ из интервала $[0,1]$, описывающее степень принадлежности X к множеству A.

Исходя из предложенного, можно для НМ указать следующую форму записи:

$$A = \left\{ [x, \mu(x)] \;\middle|\; x \in X \right\}. \qquad (2.1)$$

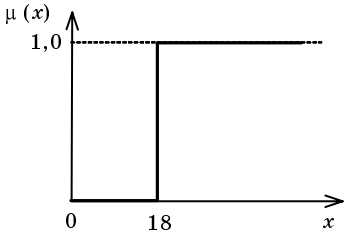


Рис. 2.3

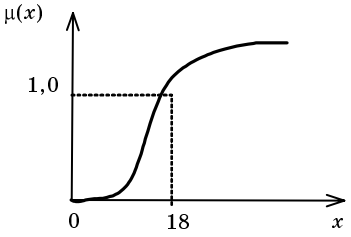


Рис. 2.4

Еще одним примером, указывающим на различие между четкими и нечетким множествами, является понятие уикенда (weekend). В четком представлении к дням уикенда относятся только суббота и воскресенье, однако из собственного опыта известно, что такие дни, особенно в летний период, включают обычно вторую половину пятницы и утренние часы понедельника. Здесь происходит столкновение с ситуацией, когда четкая логика «да-нет» перестает быть полезной, а более ценной и привычной для человека становится НЛ. В НЛ истинность любого утверждения становится частью спектра неопределенности из интервала [0,1]. НЛ противоположна двузначной, и в ней ответ на вопрос «x – элемент множества A?» может быть «да», «нет» или любой из 100 промежуточных значений между «да» и «нет» (1 и 0). Иначе говоря, x может иметь частичную принадлежность к множеству A.

Перечислим основные свойства НМ [2–4].

1. Носитель A – множество тех его элементов x, для которых $\mu(x)$ положительна:

$$\text{Носитель}\,(A) = \left\{ x \in X \;\middle|\; \mu(x) \geq 0 \right\}.$$

2. Точка перехода A – это элемент множества A, для которого $\mu(x) = 0{,}5$.

3. α-срез НМ – множество элементов x, для которых $\mu(x)$ принимает значение не меньше заданного числа α ($0<\alpha<1$):

$$A_\alpha = \left\{ x \in X \;\middle|\; \mu(x) \geq \alpha \right\}.$$

Пример 2.2. Пусть $\mu(x)$ – треугольная функция, определенная на множестве действительных чисел R (рис. 2.5). α-срез в этом случае определяется как

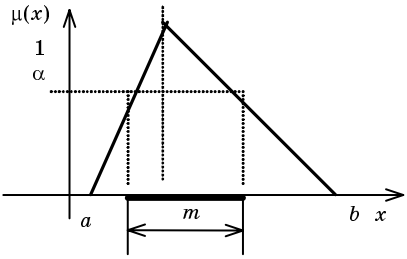


Рис. 2.5

$$A_\alpha = \begin{cases} \left[a + \alpha(m-a),\ b - \alpha(b-m)\right], & \text{если } 0 \le \alpha \le 1, \\ R, & \text{если } \alpha = 0. \end{cases}$$

4. Высота НМ A – точная верхняя грань его ФП:

$$H(A) = sup\ \mu(x),$$

$$x \in X.$$

Если $H(A) = 1$, то множество – нормализованное; если $H(A) < 1$, НМ – субнормализованное, и для приведения к нормализованному виду необходима нормировка: $\mu(x)/\sup\mu(x)$.

5. Одноточечное множество – носитель A состоит из одной точки:

$$A = \mu \Big| x,$$

где μ – степень принадлежности x множеству A.

6. Дискретное НМ – носитель A состоит из конечного числа элементов:

$$A = \mu_1 \Big| x_1 + \mu_2 \Big| x_2 + \ldots + \mu_n \Big| x_n,$$

где $\mu_i \ \left(i = \overline{1,n}\right)$ – степень принадлежности $x_i \in A$.

7. Табличный способ задания НМ имеет вид:

x_i	14	16	18	20	22
μ_i	0,1	0,3	0,5	0,8	1,0

В этом случае носитель A состоит из пяти элементов со степенями принадлежности от 0,1 до 1,0.

8. Носитель НМ A состоит из бесконечного числа точек; тогда $\mu(x)$ выражается графически или аналитически.

2.3. Операции над нечеткими множествами

Логические операции

1. Включение: НМ A содержится в НМ B ($A \subset B$) тогда и только тогда, когда $\mu_A(x) \le \mu_B(x)$.

2. Эквивалентность: два НМ A и B эквивалентны ($A = B$) тогда и только тогда, когда для всех $x \in X$ имеет место равенство: $\mu_A(x) = \mu_B(x)$.

3. Объединение (соответствует логической операции ИЛИ) двух НМ A и B ($A \cup B$) определяется как наименьшее НМ, включающее как A, так и B, с ФП следующего вида:

$$\mu_{A\cup B}(x) = \max\left[\mu_A(x), \mu_B(x)\right] = \mu_A(x) \cup \mu_B(x).$$

4. Пересечение (соответствует логической операции И) двух НМ A и B ($A \cap B$) определяется как наибольшее НМ, содержащееся одновременно в A и B, с ФП вида

$$\mu_{A\cap B}(x) = \min\left[\mu_A(x), \mu_B(x)\right] = \mu_A(x) \cap \mu_B(x).$$

5. Дополнение B (соответствует логическому отрицанию НЕ) с ФП вида

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x).$$

Для нечетких множеств можно привести графическую интерпретацию. Рассмотрим прямоугольную систему координат, на оси ординат которой откладываются значения $\mu_A(x)$, а на оси абсцисс в произвольном порядке расположены элементы **E**. Такое представление, показанное на рис. 2.6, делает наглядными простые операции над НМ.

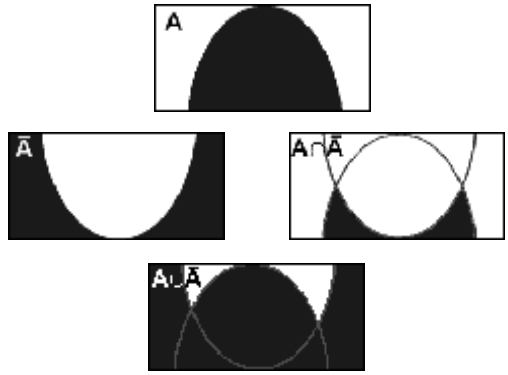


Рис. 2.6

На верхней части рисунка заштрихованная часть соответствует НМ A; ниже показаны дополнение $\bar{A}$, пересечение $A \cap B$ и объединение $A \cup B$.

Введенные операции над НМ основаны на использовании операций max и min. В теории НМ разрабатываются вопросы построения обобщенных, параметризованных операторов пересечения, объединения и дополнения, позволяющих учесть разнообразные смысловые оттенки соответствующих им связок «И», «ИЛИ», «НЕ».

Один из подходов к операторам пересечения и объединения заключается в их определении в классе треугольных норм и конорм.

Треугольной нормой (t-нормой) называется двуместная действительная функция T:[0,1] · [0,1]→[0,1], удовлетворяющая следующим условиям:

1. $T(0,0) = 0$; $T(\mu_A, 1) = \mu_A$; $T(1, \mu_A) = \mu_A$ – ограниченность;
2. $T(\mu_A, \mu_B) \le T(\mu_C, \mu_D)$, если $\mu_A \le \mu_C$, $\mu_B \le \mu_D$ – монотонность;
3. $T(\mu_A, \mu_B) = T(\mu_B, \mu_A)$, – коммутативность;
4. $T(\mu_A, T(\mu_B, \mu_C)) = T(T(\mu_A, \mu_B), \mu_C)$ – ассоциативность.

Примерами треугольных норм являются:

$$\min(\mu_A, \mu_B),$$
$$\text{произведение } (\mu_A \cdot \mu_B),$$
$$\max(0, (\mu_A + \mu_B - 1).$$

Треугольной конормой (t-конормой) называется двуместная действительная функция S:[0,1] · [0,1] → [0,1] со следующими свойствами:

1. $S(1,1) = 1$; $S(\mu_A, 0) = \mu_A$, $S(0, \mu_A) = \mu_A$ – ограниченность;
2. $S(\mu_A, \mu_B) \ge S(\mu_C, \mu_D)$, если $\mu_A \ge \mu_C$, $\mu_B \ge \mu_D$ – монотонность;
3. $S(\mu_A, \mu_B) = S(\mu_B, \mu_A)$, – коммутативность;
4. $S(\mu_A, S(\mu_B, \mu_A)) = S(S(\mu_A, \mu_B), \mu_C)$ – ассоциативность.

Примеры t-конорм:

$\max(\mu_A, \mu_B)$,
$\mu_A + \mu_B - \mu_A \cdot \mu_B$,
$\min(1, \mu_A + \mu_B)$.

Алгебраические операции

1. Алгебраическое произведение НМ A и B обозначается как $A \cdot B$ и имеет ФП следующего вида:

$$\mu_{A \cdot B}(x) = \mu_A(x)\mu_B(x).$$

2. Алгебраическая сумма этих множеств, обозначаемая $A \oplus B$, имеет ФП вида:

$$\mu_{A \oplus B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x).$$

Для операций $\{\cdot, \oplus\}$ выполняются следующие свойства:

коммутативность

$$A \cdot B = B \cdot A;$$
$$A \oplus B = B \oplus A;$$

ассоциативность

$$(A \cdot B) \cdot C = A \cdot (B \cdot C);$$
$$(A \oplus B) \oplus C = A \oplus (B \oplus C).$$

На основе операции алгебраического произведения определяется операция возведения в степень α НМ A (α – положительное число). Нечеткое множество A^α определяется следующей ФП:

$$\mu_{A^\alpha}(x) = \mu_A^\alpha(x).$$

Частными случаями возведения в степень являются:

концентрация (concentration)

$$CON(A) = A^2;$$

$$\mu_{CON(A)}(x) = \mu_A^2(x).$$

Эта операция позволяет уменьшить степень принадлежности элементов этому множеству. В естественном языке это соответствует усиливающему терму «очень»;

растяжение (dilation)

$$DIL(A) = A^{0,5};\ \mu_{DIL(A)}(x) = \sqrt{\mu_A(x)}.$$

Эта операция противоположна по своему смыслу операции концентрации и соответствует термину «довольно».

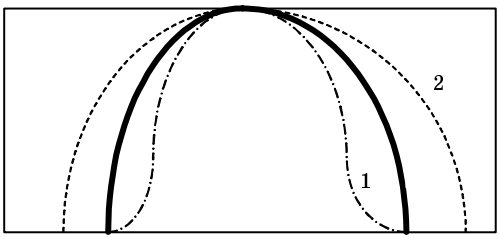


Рис. 2.7

На рис. 2.7 приведены ФП (*1* – концентрация; *2* – растяжение), иллюстрирующие операции концентрации и растяжения.

2.4. Построение функций принадлежности

Несмотря на то что выше было введено понятие функции принадлежности, конкретизируем ее определение как кривую, указывающую, каким образом каждая точка входного пространства отображается в степень принадлежности между 0 и 1.

Для нахождения ФП могут быть использованы следующие методы:

– прямой;

– косвенный;

– типовые формы;

– по данным эксперимента.

Прямой метод. Эксперт задает значения ФП $\mu(x)$ для $\forall\, x \in X$. Обычно прямые методы используются для измеримых понятий, таких

как, например, темп роста, величина дохода и т. п. или при выявлении бинарных значений какого-либо параметра. При таком подходе применяют также групповые прямые методы, когда группе экспертов предъявляется конкретный объект (напомним, что под объектом подразумевается любое предприятие, фирма, магазин), и каждый из экспертов должен дать по каждому параметру один из двух ответов такого, например, вида: «этот параметр в норме» или «этот параметр не в норме». Количество положительных ответов, деленное на число экспертов, дает величину ФП для параметра, находящегося в нормативных границах.

Косвенный метод. Используют в случаях, когда отсутствуют количественные признаки, необходимые для определения НМ. В этом случае применяют метод попарных сравнений, которые можно представить матрицей отношений A. Эксперт сам формирует матрицу A, в которой диагональные элементы равны единице, а элементы, симметричные относительно диагонали, заполняются значениями a_{ij} и $1/a_{ij}$ (a_{ij} – отношение предполагаемых экспертом значений ФП i-го и j-го признаков рассматриваемого объекта).

Типовые формы. Могут применяться различные виды ФП, в частности, треугольная, трапециевидная, гауссова, сигмоидальная и другие. Форма ФП определяется разработчиком системы, исходя из условий простоты, удобства и эффективности использования. Например, в пакете прикладного программирования `Matlab` в модуле `Fuzzy Logic`, применяемого для решения задач посредством нечеткой логики, имеется одиннадцать видов ФП. На рис. 2.8 показаны некоторые из ФП.

По данным эксперимента определяются относительные частоты проявления того или иного признака у объекта, на основании которых находятся значения ФП.

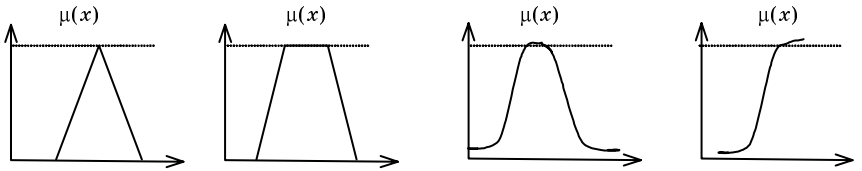


Рис. 2.8

Различные методы построения ФП НМ можно классифицировать по четырем признакам [2]:

– предполагаемый вид области определения НМ: числовая – дискретная (*a*) или непрерывная (*b*) – и нечисловая (*c*);

– применяемый способ экспертного опроса: индивидуальный (d_1), групповой (d_2);

– тип используемой экспертной информации: порядковая (e_1) или кардинальная (e_2) шкалы;

– интерпретация данных экспертного опроса: вероятностная (D), детерминированная (N).

Был предложен метод построения ФП типа $\langle a, d_1, e_2, N\rangle$ на основе количественного сравнения степеней принадлежности, проводимого экспертом. Результатом такого подхода является матрица $B = \|b_{ij}\|$ размера $n \cdot n$, где n — число точек x_i, в которых сравниваются значения ФП. Элемент b_{ij} матрицы B является субъективной оценкой отношения $\mu_A(x_i) / \mu_A(x_j)$ и показывает, во сколько раз, по мнению эксперта, $\mu_A(x_i)$ больше $\mu_A(x_j)$. Величина b_{ij} назначается в соответствии с балльной шкалой, значения которой интерпретируются экспертом на основе его опыта и знаний.

Количество вопросов, на которые должен ответить эксперт, составляет не n^2, а $(n^2 - n)/2$, так как по определению $b_{ii} = 1$ и с целью согласования экспертных оценок принимается, что $b_{ij} = 1/b_{ji}$. Значения ФП $\mu_A(x_1),\ldots, \mu_A(x_n)$ в точках $x_1,\ldots, x_n$ определяются на основе решения задачи о нахождении собственного вектора матрицы B:

$$BW^T = \nu_{\max} W,$$

где $n_{\max}$ – максимальное собственное число матрицы B; $W = (w_1,\ldots,w_n)$ – соответствующий собственный вектор; ^{T} — символ транспонирования.

Поскольку матрица B является положительной по построению, решение этой задачи всегда существует и является единственным. Можно показать [2], что в этом случае

$$\mu_A(x_i) = w_i \Big/ \sum_{i=1}^{n} w_i .$$

При этом значения ФП $\mu_A(x_i)$ оказываются измеренными в шкале отношений.

Предложенный метод обладает рядом достоинств:

– применяемая в методе процедура парных сравнений является достаточно простой для экспертов, поскольку она не навязывает ему априорных ограничений;

– метод допускает наблюдаемую на практике несогласованность оценок экспертов и позволяет учесть и оценить ее введением коэффициента несогласованности λ, равного

$$\lambda = (\nu_{\max} - n)/n,$$

$\lambda \geq 0$; $\lambda = 0$ соответствует ситуации полной согласованности суждений экспертов; чем больше значение λ, тем больше несогласованность мнений.

Кроме описанного, можно предложить метод параметрического определения ФП типа $\langle b, d_1, e_2, N\rangle$ с участием индивидуального эксперта. В соответствии с данным методом вид ФП задается аксиоматически, а ее параметры непосредственно оцениваются экспертом. Например, в случае треугольной формы эксперт указывает такие ее параметры x_1, x_2, x_3, при которых ФП принимает единичное и нулевые значения, т. е. $\mu_A(x_2) = 1$ и для всех $x \leq x_1$, $x \geq x_3$ имеет место равенство $\mu_A(x) = 0$.

Параметрическое представление ФП является компактным, обеспечивает простоту их построения, однако связано с исследованием адекватности используемых форм (треугольной, трапециевидной, гауссовой и др.) и соответствующих аналитических описаний ФП.

Рассмотрим пример построения ФП с помощью экспертов.

Пример 2.3 [5]. В банк обратились четыре предприятия с просьбой о выдаче им кредита. На основании данных бухгалтерской отчетности предприятий рассчитываются финансовые показатели, характеризующие кредитоспособность заемщиков:

– коэффициент абсолютной ликвидности (F_1);
– промежуточный коэффициент покрытия (F_2);
– общий коэффициент покрытия (F_3);
– коэффициент финансовой независимости (F_4);
– коэффициент рентабельности продукции (F_5).

Рассчитанные значения финансовых показателей приведены в табл. 2.1.

Таблица 2.1

Показатели	Значения				Норма
	A_1	A_2	A_3	A_4	
F_1	0,15	0,10	0,10	0,14	0,10–0,25
F_2	1,30	0,71	0,59	0,57	0,50–1,00
F_3	2,78	2,27	1,86	1,27	1,00–2,50
F_4	0,75	0,72	0,71	0,68	0,6
F_5	0,28	0,11	0,15	0,12	Чем больше, тем лучше

Сопоставляя рассчитанные показатели с их нормативными значениями, эксперты строят ФП по пяти критериям F_1,..., F_5, которые приведены на рис. 2.9.

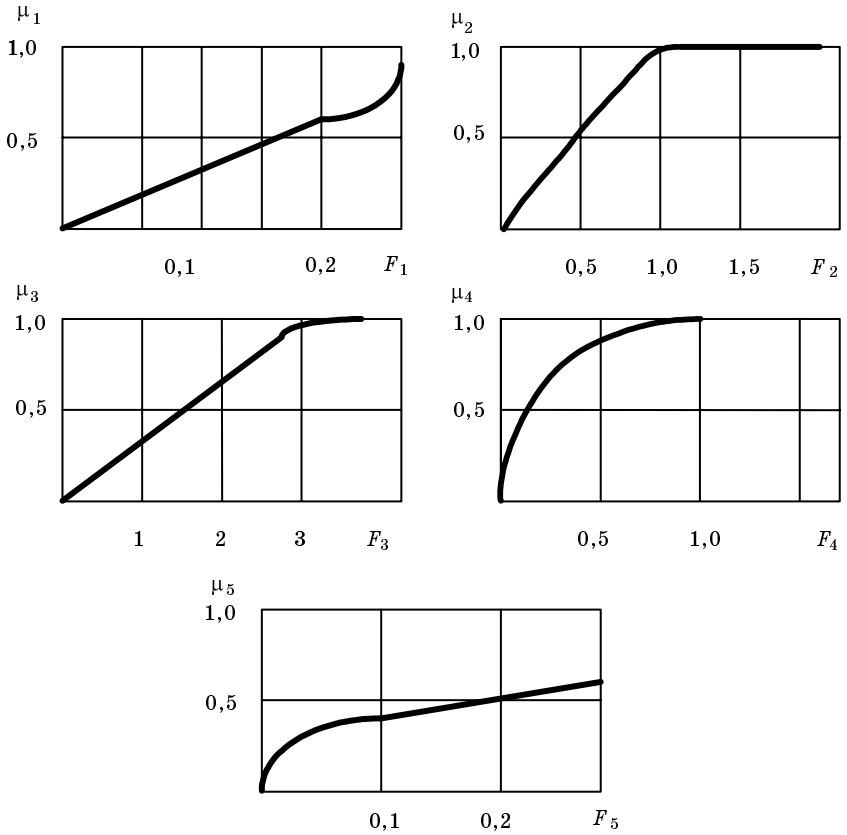


Рис. 2.9

В заключение укажем на различие между ФП и вероятностью. Вследствие того что ФП и вероятность имеют одинаковый диапазон изменения от 0 до 1, между этими понятиями часто происходит путаница. Две этих меры совершенно различны, хотя обе описывают меру неопределенности. Каждая из величин измеряет различные аспекты неопределенности.

Функция принадлежности является описанием сложного состояния; дополнительные данные не изменяют ее значения. С другой стороны, вероятности зависят от частоты события, поэтому последующие выборки могут изменить вероятность.

Рассмотрим такой пример [6]. Имеются две бутылки с водой.

Бутылка A – с ФП = 0,90 принадлежности к множеству питьевой воды, бутылка B – с 90% -й вероятностью, что в ней питьевая вода.

Какую бутылку предпочтительнее выбрать для питья?

Бутылка A не полностью пригодна для питья в соответствии с ее ФП, но похожа на питьевую воду. В ней может быть и грязная вода.

Бутылка B имеет 90% -ю вероятность того, что это питьевая вода, и 10% -ю вероятность того, что в ней находится непитьевая вода (в крайнем случае – яд). При единственной выборке в бутылке B может оказаться или питьевая вода, или яд. (Из 100 бутылок этого типа в среднем в 90 окажется питье, а в 10 – яд). Поскольку бутылка A бульшей частью содержит питьевую воду, то предпочтительный выбор останется за ней.

2.5. Нечеткие и лингвистические переменные

В нечеткой логике для описании объектов и явлений с помощью НМ вводятся понятия нечеткой и лингвистической переменных.

Нечеткая переменная характеризуется тройкой $<\alpha, X, A>$,

где α- наименование переменной; X – универсальное множество (область определения α); A – нечеткое множество на X, описывающее ограничения (т. е. $\mu_A(x)$) на значения нечеткой переменной α.

Лингвистическая переменная – это набор $<\beta, T, X, G, M>$,

где β – наименование лингвистической переменной; T – множество ее значений (терм-множество), представляющих собой наименования нечетких переменных, областью определения каждой из которых является множество X; G – синтаксическая процедура, позволяющая оперировать элементами терм-множества T, в частности, генерировать новые термы (значения); M – семантическая процедура, позволяющая превратить каждое новое значение лингвистической переменной, образуемое процедурой G, в нечеткую переменную, т. е. сформировать соответствующее нечеткое множество.

Для того чтобы избежать большого количества символов, рекомендуются следующие действия [7]:

– символ β используют как для названия самой переменной, так и для всех ее значений;

– для обозначения НМ и его названия пользуются одним и тем же символом, например терм «малый», являющийся значением лингвистической переменной β = «доход», одновременно есть и НМ M («малый»).

Пример 2.4. Для определения величины дохода введем понятия «малый», «средний» и «большой». При этом минимальный доход установим равным 2000 р., а максимальный – 10000 р.

Формализация такого описания может быть проведена с помощью следующей лингвистической переменной $<\beta, T, X, G, M>$,

где β – доход; T – {«малый доход», «средний доход», большой доход»}; X – [2000, 10000]; G – процедура образования новых термов с помощью связок «и», «или» и модификаторов типа «очень», «не», «слегка» и т. д., например, малый или средний доход» «очень большой доход» и др.; M – процедура задания на X = [2000,10000] нечетких подмножеств A_1 = «малый доход», A_2 = «средний доход», A_3 = «большой доход», а также НМ для термов из G(T) в соответствии с правилами трансляции нечетких связок и модификаторов «и», «или» и модификаторов типа «очень», «не», «слегка» и др. операций над НМ вида: $A \cap B$, $A \cup B$, $\bar{A}$, $CON\,A = A^2$, $DIL\,A = A^{0,5}$.

На рис. 2.10 приведены ФП для НМ: A_1 = «малый доход», A_2 = «средний доход», A_3 = «большой доход».

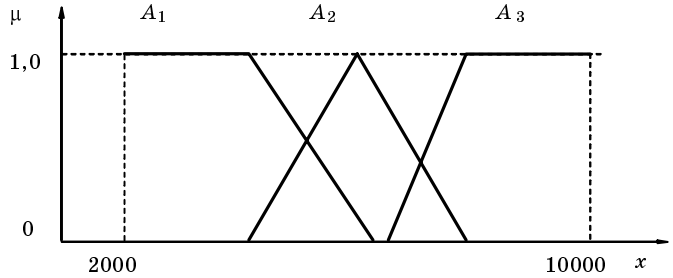


Рис. 2.10

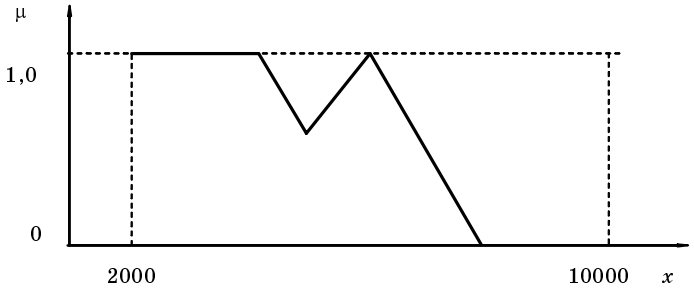


Рис. 2.11

В качестве еще одной иллюстрации на рис. 2.11 приведена ФП для НМ вида $A \cup B$.

2.6. Нечеткие алгоритмы и выводы

Нечеткий алгоритм – упорядоченное множество нечетких правил, в формулировке которых содержатся нечеткие указания.

Примерами нечетких алгоритмов могут служить такие правила:

– «x = очень малой величине»;

– «x приблизительно равно 10»;

– «если x в интервале [4;6], то выбрать y из интервала [9;10].

Последнее правило наиболее часто используется в НЛ, а совокупность таких правил образует базу знаний вида:

$$\Pi_1: \text{если } x \text{ есть } A_1, \text{ то } y \text{ есть } B_1;$$
$$\Pi_2: \text{если } x \text{ есть } A_2, \text{ то } y \text{ есть } B_2;$$
$$\Pi_3: \text{если } x \text{ есть } A_n, \text{ то } y \text{ есть } B_n,$$

где x – входная переменная (имя для известных значений данных); y – переменная вывода (имя для значений данных, которое будет вычислено).

Такое правило, называемое продукционным, состоит из двух частей:

– антецедент (предпосылка правила, после союза «если»);

– консецедент (следствие, заключение или вывод, после союза «то»).

Например, если обслуживание хорошее, то чаевые – средние. Входом в правило здесь является текущее значение входной переменной (обслуживание). Выходом служит нечеткое множество «средние», которое позже должно быть дефазифицировано (должно быть получено четкое значение выходной переменной).

Между принятым лингвистической переменной x значением A и значением B, принятым переменной y, существует отношение, которое называется импликацией и обозначается $R = A \rightarrow B$.

Операцию импликации в алгебре НМ можно реализовывать поразному, но в любом случае общий логический вывод включает следующие этапы [3, 7]:

– приведение к нечеткости;

– логический вывод;

– композиция;

– приведение к четкости.

В общем случае схема системы нечеткого логического вывода приведена на рис. 2.12.

Опишем более подробно каждый из этапов.

Приведение к нечеткости (фазификация, fuzzyfication): ФП, определенные на входных переменных, применяются к их факти-

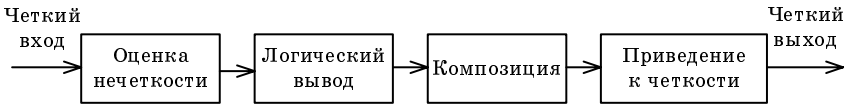


Рис. 2.12

ческим значениям для определения степени истинности предпосылки каждого правила.

Логический вывод. Вычисленное значение истинности для предпосылок каждого правила применяется к заключениям каждого правила. Это приводит к одному НМ, которое будет назначено каждой переменной вывода для каждого правила.

В качестве правил логического вывода обычно используются операции:

– минимума (min, рис. 2.13, *а*);

– произведения (prod, рис. 2.13, *б*).

а) *б*)

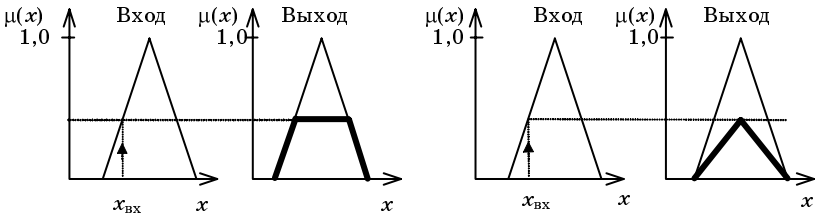


Рис. 2.13

При операции min ФП вывода (правой части правила) «отсекается» по высоте, соответствующей вычисленной степени истинности предпосылки (левой части) правила (нечеткая логика «И»).

В логическом выводе prod ФП вывода масштабируется в зависимости от степени истинности предпосылки правила.

Композиция. Выходы всех правил вычисляются отдельно, но в правой части нескольких из них может быть указана одна и та же нечеткая переменная. Нечеткие подмножества, назначенные для каждой переменной вывода (или одной переменной), объединяются вместе для формирования одного нечеткого подмножества.

При таком объединении используются следующие операции при композиции:

– максимума (max, рис. 2.13, *а*);

– суммы (sum, рис. 2.13, *б*).

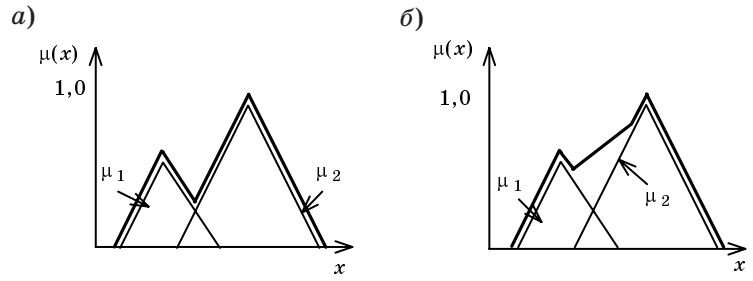


Рис. 2.14

При композиции max комбинированный вывод формируется как поточечный максимум по всем нечетким подмножествам (нечеткая логика «ИЛИ»).

При композиции sum такой вывод строится как поточечная сумма по всем нечетким подмножествам.

Приведение к четкости (дефазификация, defuzzification). Этот прием используется, когда необходимо перейти от нечеткого вывода к четкому выходному значению.

При переходе от нечеткого вывода к четкому выходу могут использоваться различные способы, в частности:

– метод центра тяжести (определяется абсцисса центра тяжести кривой под ФП);

– метод первого максимума (выбирается наименьший элемент НМ, при котором достигается максимум значения ФП);

– метод среднего максимума;

– метод наименьшего максимума (определяется из вида ФП).

Рассмотрим следующие наиболее часто используемые модификации алгоритма нечеткого вывода, полагая, что базу правил образуют два нечетких правила вида:

$$\Pi_1\text{: если } x \text{ есть } A_1 \text{ и } y \text{ есть } B_1\text{, то } z \text{ есть } C_1,$$
$$\Pi_2\text{: если } x \text{ есть } A_2 \text{ и } y \text{ есть } B_2\text{, то } z \text{ есть } C_2,$$

где x и y – имена входных переменных; z – имя переменной вывода; $A_1, A_2, B_1, B_2, C_1, C_2$ – некоторые заданные значения ФП.

При этом четкое значение z_0 необходимо определить на основе приведенной информации и четких значений x_0 и y_0.

Общий вид диаграммы потока информации от входов к выходу показан на рис. 2.15.

Рассмотренные ниже алгоритмы нечеткого вывода различаются, в основном, способом получения четкого выхода. Приведенная на

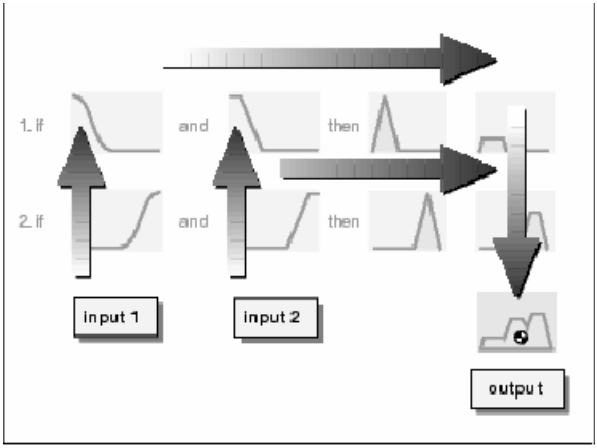


Рис. 2.15

рис. 2.15 диаграмма полностью соответствует алгоритму Мамдани, но несколько отличается от других алгоритмов.

Алгоритм Мамдани (Mamdani). Данный алгоритм математически может быть описан следующим образом.

1. Нечеткость: находятся степени истинности для предпосылок каждого правила: $A_1(x_0)$, $A_2(x_0)$, $B_1(y_0)$, $B_2(y_0)$.

2. Нечеткий вывод: находятся уровни «отсечения» для предпосылок каждого из правил (с использованием операции min):

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0),$$

$$\alpha_2 = A_2(x_0) \wedge B_2(y_0),$$

где через «$\wedge$» обозначена операция логического минимума (min). После этого находятся «усеченные» функции принадлежности:

$$C_1'(z) = (\alpha_1 \wedge C_1(z)),$$

$$C_2'(z) = (\alpha_2 \wedge C_2(z)).$$

3. Композиция: с использованием операции max, далее обозначаемой как «$\vee$», производится объединение найденных усеченных функций, что приводит к получению итогового нечеткого подмножества для переменной выхода с ФП следующего вида:

$$\mu_{\sum}(z) = C(z) = C_1'(z) \vee C_2'(z) = (\alpha_1 \wedge C_1(z)) \vee (\alpha_2 \wedge C_2(z).$$

4. Приведение к четкости для нахождения z_0 проводится одним из указанных выше способов, например центроидным методом.

Алгоритм Цукамото (Tsukamoto). Исходные посылки такие же, как у предыдущего алгоритма, но в данном случае предполагается, что функции $C_1(z)$, $C_2(z)$ являются монотонными.

Первый этап — такой же, как в алгоритме Мамдани.

Второй этап – сначала определяются (как в алгоритме Мамдани) уровни отсечения α_1 и α_2, а затем – посредством решения уравнений:

$$\alpha_1 = C_1(z_1),$$

$$\alpha_2 = C_2(z_2)$$

находятся четкие значения z_1 и z_2 для каждого из исходных правил.

Третий этап – определяется четкое значение переменной вывода как взвешенное среднее z_1 и z_2:

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2}.$$

Пример 2.5. Пусть $A_1(x_0) = 0{,}7$; $A_2(x_0) = 0{,}6$; $B_1(y_0) = 0{,}3$; $B_2(y_0) = 0{,}8$. Соответствующие уровни отсечения определяются как

$$\alpha_1 = \min(A_1(x_0), B_1(x_0)) = \min(0{,}7; 0{,}3) = 0{,}3;$$

$$\alpha_2 = \min(A_2(x_0), B_2(x_0)) = \min(0{,}6; 0{,}8) = 0{,}6.$$

После этого в результате решения уравнений $C_1(z_1) = 0{,}3$; $C_2(z_2) = 0{,}6$ находятся значения $z_1 = 8$ и $z_2 = 4$. В результате четкое значение переменной выхода составляет величину

$$z_0 = (8 \cdot 0{,}3 + 4 \cdot 0{,}6) / (0{,}3 + 0{,}6) = 6.$$

Иллюстрация алгоритма Цукамото показана на рис. 2.16.

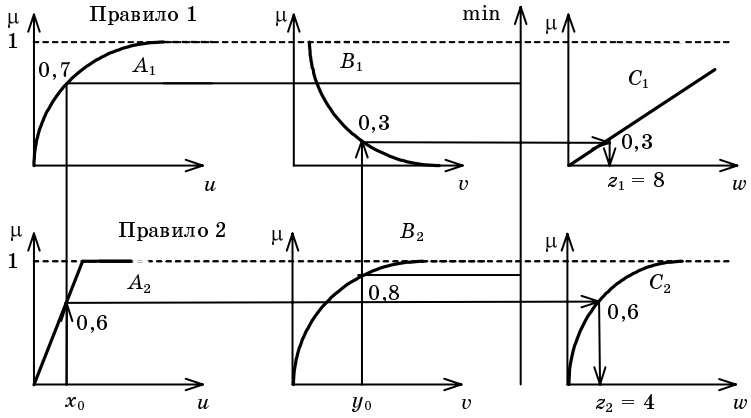


Рис. 2.16

Алгоритм Сугено (Sugeno). В этом алгоритме использован набор правил в следующей форме (как и раньше, приводим пример двух правил):

Π_1: если x есть A_1 и y есть B_1, тогда $z = a_1x + b_1y$,
Π_2: если x есть A_2 и y есть B_2, тогда $z = a_2x + b_2y$.

Первый этап выполняется так же, как в алгоритме Мамдани.

На втором этапе определяются значения α_1 и α_2:

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0);$$
$$\alpha_2 = A_2(x_0) \wedge B_2(y_0),$$

и индивидуальные выходы правил:

$$z_1^* = a_1x_0 + b_1y_0;$$
$$z_2^* = a_2x_0 + b_2y_0$$

На третьем этапе рассчитывается четкое значение переменной:

$$z_0 = \frac{\alpha_1 z_1^* + \alpha_2 z_2^*}{\alpha_1 + \alpha_2}.$$

Принцип работы алгоритма Сугено приведен на рис. 2.17.

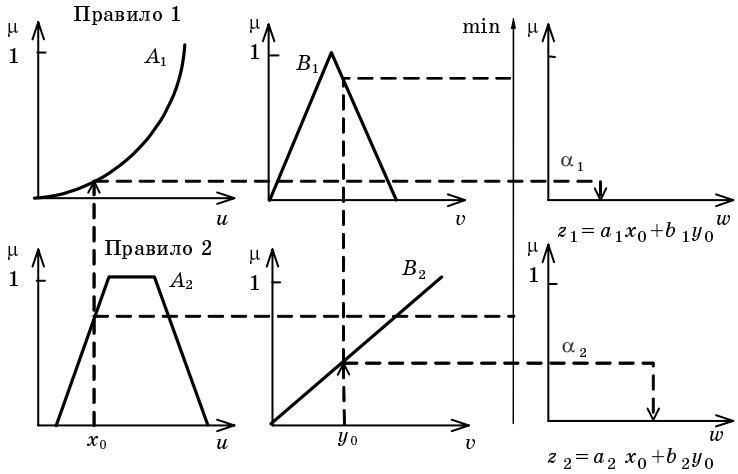


Рис. 2.17

Алгоритм Ларсена (Larsen). В этом алгоритме нечеткая импликация моделируется с использованием оператора умножения.

Первый этап выполняется так же, как в алгоритме Мамдани.

Второй этап – по аналогии с алгоритмом Мамдани вначале находятся значения

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0);$$
$$\alpha_2 = A_2(x_0) \wedge B_2(y_0),$$

а затем частные нечеткие подмножества $\alpha_1 C_1(z)$, $\alpha_2 C_2(z)$.

Третий этап – находится итоговое нечеткое подмножество с ФП вида

$$\mu_{\sum}(z) = C(z) = \alpha_1 C_{1'}(z) \vee \alpha_2 C_2(z).$$

Четвертый этап – при необходимости производится приведение к четкости (как в ранее рассмотренных алгоритмах).

Принцип работы алгоритма Ларсена показан на рис. 2.18.

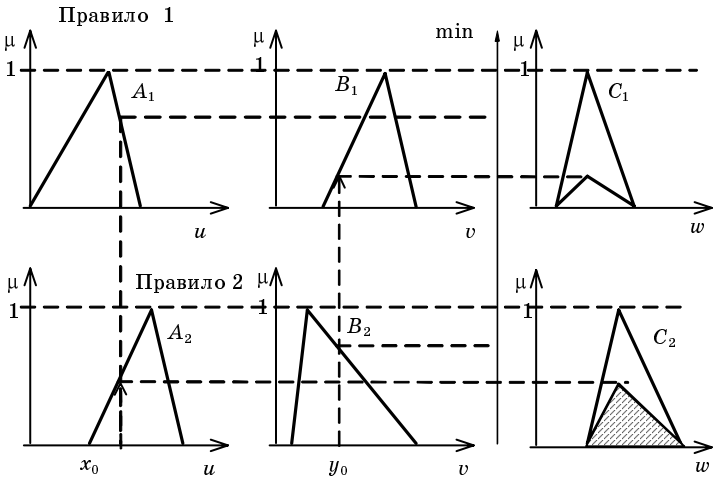


Рис. 2.18

Упрощенный алгоритм нечеткого вывода. Исходные правила в данном случае задаются в виде:

Π_1: если x есть A_1 и y есть B_1, тогда $z = c_1$,
Π_2: если x есть A_2 и y есть B_2, тогда $z = c_2$,

где c_1, c_2 – некоторые обычные четкие числа.

Первый этап – выполняется так же, как в алгоритме Мамдани.

Второй этап – находятся числа

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0);$$
$$\alpha_2 = A_2(x_0) \wedge B_2(y_0).$$

Третий этап – рассчитывается четкое значение выходной переменной по следующей формуле:

$$z_0 = \frac{\alpha_1 c_1 + \alpha_2 c_2}{\alpha_1 + \alpha_2}.$$

Принцип работы упрощенного алгоритма показан на рис. 2.19

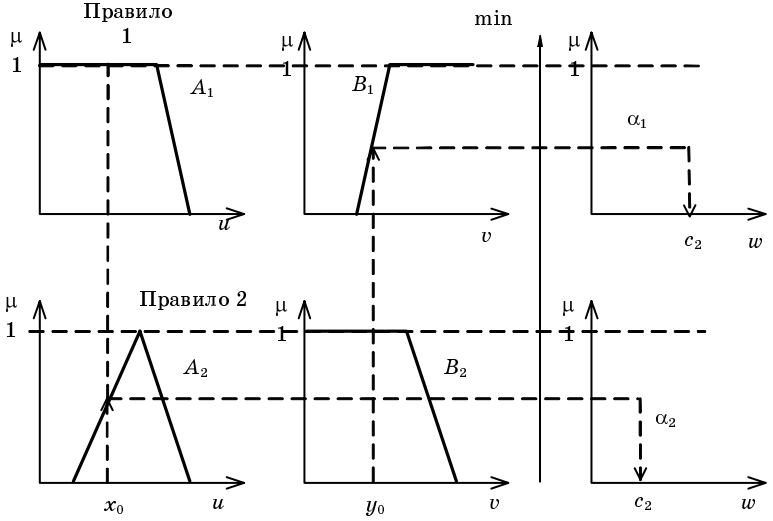


Рис. 2.19

После рассмотрения алгоритмов получения нечеткого вывода конкретизируем приведенные выше методы приведения дефазификации (приведения к четкой выходной переменной).

Для метода центра тяжести расчетные формулы имеют вид:

– для непрерывного случая:

$$z_0 = \frac{\int\limits_{\Omega} zC(z)dz}{\int\limits_{\Omega} C(z)dz};$$

– для дискретного случая:

$$z_0 = \frac{\sum\limits_{i=1}^{n} \alpha_i z_i}{\sum\limits_{i=1}^{n} z_i}.$$

Для метода первого максимума:

$$z_o = \min(z \mid C(z) = \max_u C(u)).$$

Для метода среднего максимума:

$$z_0 = \frac{\int_G z\,dz}{\int_G dz},$$

где G – подмножество элементов, максимизирующих C.

Два последних метода приведены на рис. 2.20, *а* – первый максимум; *б* – средний максимум.

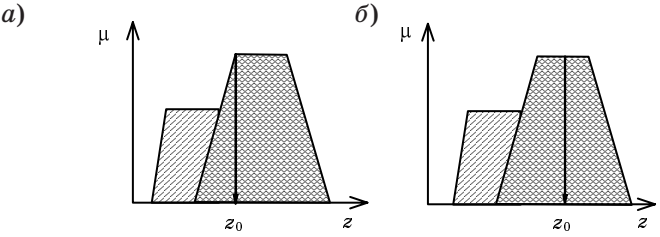


Рис. 2.20

Необходимо отметить, что из многих предложенных методов дефазификации наиболее часто используется метод центра тяжести. Рассмотренные в этом разделе алгоритмы НЛ используются в различных пакетах программного обеспечения и ниже будут продемонстрированы их возможности при решении задач менеджмента.

Приведем несколько примеров нечеткого вывода. Начнем с простого примера [3], сущность которого можно представить следующим правилом:

если *прибыль клиента Большая*,
то *кредитный рейтинг клиента Хороший*. (2.2)

Лингвистическими переменными здесь являются «большая» и «хороший». Пусть *Большая прибыль* соответствует «2000 рублей». Знание *Большая прибыль* эксперт может представить с помощью НМ, элементы которого включают значения ФП и прибыли. Например:

$$\textit{Большая прибыль} = 0,1|1,4 + 0,3|1`,6 + 0,7|1,7 + \\ + 0,8|1,8 + 0,9|1,9 + 1|2 + 1|2,2. \quad (2.3)$$

Аналогично, *Хороший рейтинг* клиента эксперт может представить, например, цифрами от 14 до 20. С помощью НМ понятие «хороший рейтинг» можно интерпретировать следующим образом:

$$Хороший\ \ рейтинг = 0,1|14 + 0,2|15 + 0,3|16 + \\ + 0,5|17 + 0,8|18 + 0,9|19 + 1|20. \qquad (2.4)$$

Таким образом, правило (2.2), представленное в словесной форме, записано в виде двух нечетких множеств (2.3) и (2.4).

Разработчик системы, используя знания эксперта и получаемые реальные результаты, корректирует значения ФП до тех пор, пока система наилучшим образом не будет моделировать конкретную ситуацию. Значения ФП можно хранить в форме базы знаний в компьютере. Таким образом, продукционные правила вида (2.2) в форме нечетких множеств (2.3) и (2.4) могут накапливаться в базе знаний.

Понятие «большая прибыль» на практике может иметь различные оттенки: «довольно большая прибыль», «очень большая прибыль» и т. д. Пусть в процессе анализа документов клиента обнаружено, что

Прибыль клиента довольно большая. (2.5)

Информацию наблюдения (2.5) можно также представить с помощью нечеткого множества:

$$Довольно\ \ большая = 0,5|1,6 + 1|1,7 + 0,8|1,8 + 0,2|1,9. \qquad (2.6)$$

Теперь требуется сделать заключение на основе двух продукционных правил:

если *Большая прибыль*, то *Хороший рейтинг.*
Довольно большая прибыль. (2.7)

Предпосылка «большая» и наблюдение «довольно большая» образуются путем сопоставления. В четкой логике сопоставление не имеет смысла, поэтому никакого логического вывода сделать нельзя. Однако человек, исходя из правил (2.7), может сделать заключение, что клиент имеет «неплохой» рейтинг. Таким образом, путем приближенного сопоставления правил (2.7) человек делает нечеткий вывод:

если *Большая прибыль*,
то *Хороший рейтинг.*
Довольно большая прибыль.
Неплохой рейтинг. (2.8)

Формула (2.8) представляет классический пример нечеткого вывода человека, рассуждающего на лингвистическом уровне. Объяснение нечеткого вывода можно провести с помощью рис. 2.21. Пусть

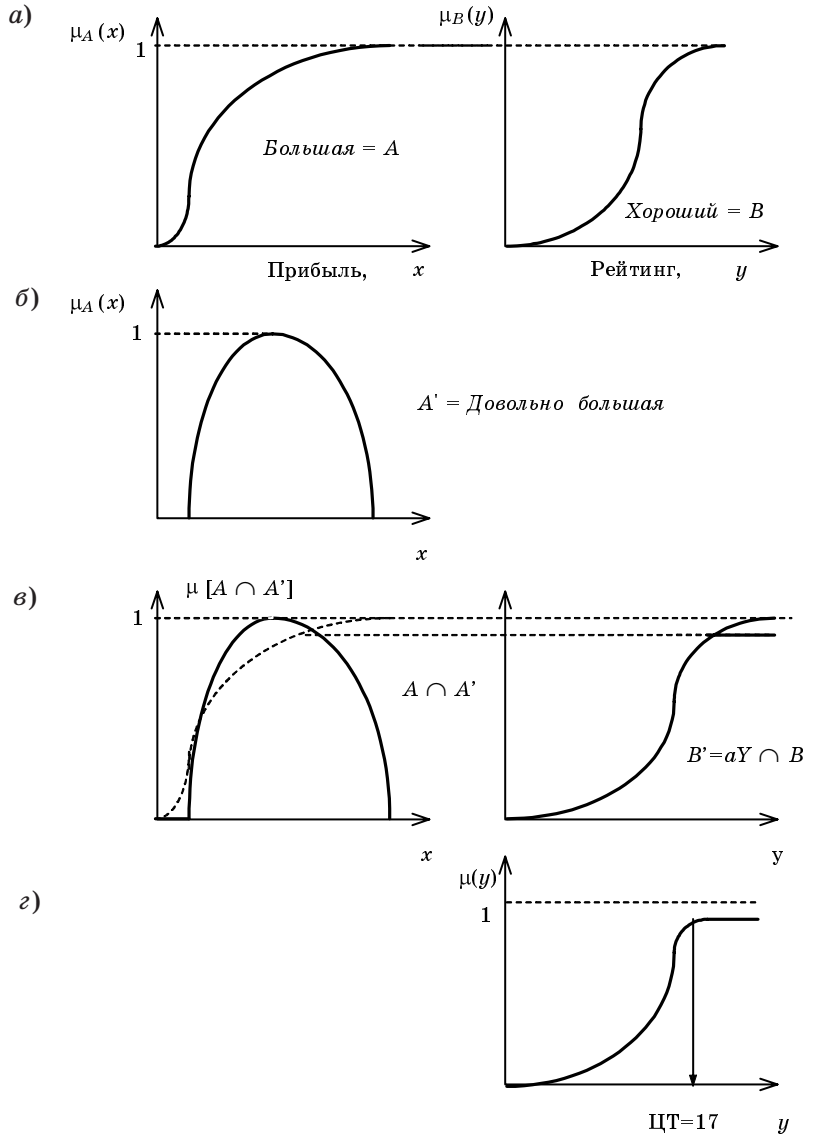


Рис. 2.21

X – полное множество значений «Большая прибыль» и Y – полное множество значений «Хороший рейтинг». Используя формулы (2.3) и (2.4) нечеткое правило (2.2.) можно графически изобразить так, как показано на рис. 2.21, *а*. Для упрощения рассуждений обозначим через A НМ «Большая» в предпосылке X и через B – НМ «Хороший» в заключении Y.

Нечеткое множество «Довольно большая» в данных наблюдения обозначим через A' (рис. 2.21, *б*). На рис. 2.21, *в* показан процесс классического нечеткого вывода.

Значение $A \cap A'$ получено в результате приближенного сопоставления предпосылки правила A и данных наблюдения A'. Затем рассмотрим максимальное значение a как некую меру сопоставления $A \cap A'$ и выполним редукцию по этой мере заключения B (рис. 2.21, *в*). В данном случае редукцией B является усечение по мере сопоставления a. На рис. 2.21, *в* величина aY означает, что

$$\mu_a Y = a. \tag{2.9}$$

Итак, для текущих данных наблюдения A' (*Довольно большая*) в результате применения правила $A \rightarrow B$ (если *Большая*, то *Хороший*) получаем B' (*Неплохой*). Здесь результат вывода B' является НМ в Y (рис. 2.21, *г*). Однако принять окончательное решение по установлению рейтинга пока нельзя. Дело в том, что на основе ФП для B необходимо получить четкое значение через процесс дефазификации (преобразованием НМ в четкое представление). На рис. 2.21, *г* для этих целей использован метод центра тяжести и определено значение выхода, составляющее примерно 17 баллов.

2.7. Формирование базы правил

Нечеткие правила конструируются из ФП входов и выходов. Такие правила обеспечивают связь между предпосылкой и заключением, выраженных в форме *если*..., *то*. Фазифицированные входы и выходы облегчают выделение правил и их обобщение. В общем случае правила могут быть получены из опыта одного или нескольких экспертов, исторических данных, комбинирования этих двух подходов, посредством использования специальных приемов.

Определение хороших лингвистических правил зависит от объема и качества знаний эксперта. Однако перевод знаний в теорию нечетких множеств не формализован, вследствие чего выбор, например формы ФП, произволен. То же самое относится и к выбору параметров ФП.

Положим, что нас интересует установление правил торговли. Пусть трейдер определяет торговое правило посредством долгой позиции, когда тренд цены равен или превышает определенную величину x или волатильность равна или ниже некоторого значения y. Такое правило в четкой (механической) интерпретации может быть записано следующим образом:

если *тренд* $\geq x$ *и волатильность* $\leq y$,
то *долгая позиция* должна быть равна z,

где x, y и z – параметры, определяемые трейдером или оцениваемые на основе имитации.

Нечеткая версия этого же правила имеет вид:

если тренд *быстрорастущий* и *волатильность низкая*,
то *торговая позиция долгая*.

Таким образом, синтаксис нечетких правил определяется как

если (нечеткая предпосылка), то (нечеткое заключение)

или в случае многих предпосылок как

если (нечеткая предпосылка 1) и (или) (нечеткая предпосылка 2)…
и (или) (нечеткая предпосылка n),
то (нечеткое заключение),

где нечеткая предпосылка имеет форму «x есть X»; x – одна из исходных скалярных переменных (например, тренд); X – НМ, связанное с этой переменной (например, отрицательный большой).

Специфика четких правил диктует частые изменения, что вызывает значительные проблемы в оценке их поведения. Такие системы с часто изменяющимися параметрами не имеют практического значения. В то же время нечеткая реализация приводит к тому, что активизация правил будет зависеть от нечетких множеств (в рассматриваемом случае тренда и волатильности акций) и от правил, связывающих свойства входа и выхода.

Набор нечетких правил образует *нечеткую ассоциативную память* (*fuzzy associative memory* – FAM). Обычно число правил, требуемых в системе, связано с числом управляемых параметров. Таким образом, в рассматриваемом примере оценка торговой позиции является единственной переменной выхода, зависящей от двух входных переменных: тренда и волатильности. Если принять, что каждый вход имеет ФП, состоящую из пяти элементов, тогда возможны 25 входных комбинаций тренда и волатильности. В большинстве слу-

чаев можно использовать меньшее количество правил. Упрощенная версия нечеткой базы правил показана на рис. 2.22.

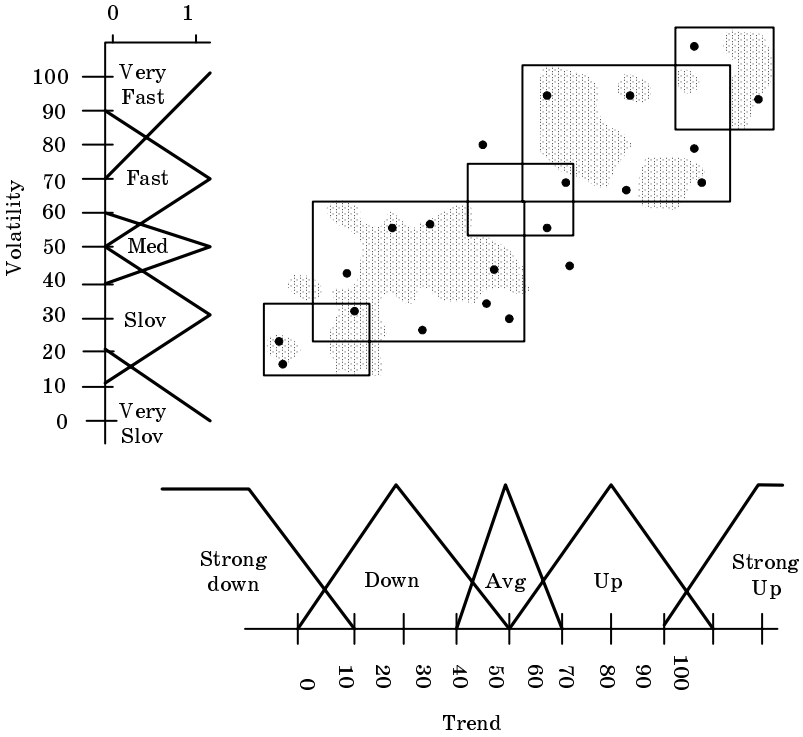


Рис. 2.22

Правила или нечеткие ассоциации представляют собой знания, которые важны для системы с точки зрения отклика на все возможные комбинации входов. Нечеткие системы запоминают наборы (банки) нечетких ассоциаций или правил. Классическим примером нечеткой ассоциативной памяти может служить FAM-матрица для задачи о балансировании перевернутого маятника [1] с двумя входными нечеткими переменными: угол θ отклонения маятника от вертикали и угловая скорость $\Delta\theta / \Delta t$ перемещения маятника и одной выходной переменной: величиной тока v электродвигателя, который регулирует отклонение маятника и возвращает его в вертикальное положение $\theta = 0$.

Каждая нечеткая переменная может принимать по п я т ь нечетких значений:

– PM (positive medium);
– PS (positive small);
– ZE (*z*ero);
– NS (negative small);
– NM (negative medium).

На рис. 2.23 показан набор значений для лингвистической переменной, задающей угол отклонения маятника θ от вертикали. Аналогичный вид имеют ФП для угловой скорости маятника $\Delta\theta / \Delta t$ и нечеткой переменной управления v.

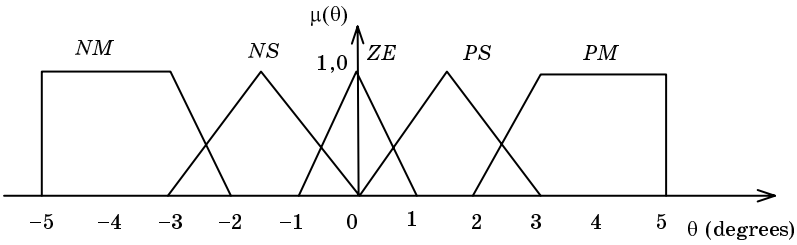


Рис. 2.23

Управляющий алгоритм для этой системы представлен набором правил вида:

$$IF\ \theta\ is\ NM\ AND\ D\theta / D\ t\ is\ ZE\ THEN\ v\ \text{is}\ PM.$$

Каждое правило содержит в левой части две переменные состояния, а в правой части переменную, характеризующую управление (действие). Данный алгоритм управления может быть представлен матрицей размерности 5×5, элементами которой являются значения нечетких переменных задачи (табл. 2.2). Пустые ячейки матрицы указывают, что никакие действия для данного состояния системы предприниматься не должны. Например, если угловая скорость NS и угол отклонения NM, то никакие воздействия на систему не оказываются.

В каждый момент времени четкие величины физических параметров системы снимаются набором датчиков. Эти значения сравниваются со всеми правилами управляющего алгоритма. Это сопоставление носит название ф а з и ф и к а ц и и (размывания) четких данных, поскольку измеряемые величины сопоставляются со значениями лингвистических переменных в левых частях правил.

Таблица 2.2

$\Delta\theta / \Delta t$	θ				
	NM	*NS*	*ZE*	*PS*	*PM*
NM			*PM*		
NS			*PS*		
ZE	*PM*	*PS*	*ZE*	*NS*	*NM*
PS		*PS*	*NS*		
PM		*PM*	*NM*		

Предположим, что закон управления маятником состоит из двух нечетких правил:

Правило 1: *IF* θ *is NM AND* $\Delta\theta / \Delta t$ *is ZE THEN v is PM*;
Правило 2: *IF* θ *is NS AND* $\Delta\theta / \Delta t$ *is ZE THEN v is PS*.

Примем, что θ_0 и $\Delta\theta_0$ – четкие показания датчиков в некоторый момент времени. Степень, с которой эти показания являются членами переменных *NM* и *ZE*, определяется величиной ФП, найденной при этих показаниях датчиков, т. е. $\mu_{NM}(\theta_0)$ и $\mu_{ZE}(\Delta\theta_0)$ для правила 1, соответственно. Выходом первого правила является минимум этих двух величин, поскольку при логическом выводе используется операция минимума (конъюнкция), поэтому для левой части правила 1 имеем:

$$d_1 = \min[\mu_{NM}(\theta_0), \mu_{ZE}(\Delta\theta_0)].$$

Аналогично для правила 2 получим:

$$d_2 = \min[\mu_{NS}(\theta_0), \mu_{ZE}(\Delta\theta_0)].$$

Вклад текущего управления в правило 1 определяется применением величины d_1 к его правой части. Величина этого вклада определяется как аргумент ФП текущего управления в следующем виде:

$$a_1 = \mu_1^{-1}(d_1).$$

То же самое справедливо для правила 2

$$a_2 = \mu_2^{-1}(d_2).$$

Вклад обоих правил определяется на этапе дефазификации посредством метода Цукамото в следующем виде:

$$a^* = \frac{\sum_{i=1}^{n} d_i a}{\sum_{i=1}^{n} d_i},$$

где n – число правил.

Дефазифицированная композиция всех правил, полученная с помощью последнего уравнения, дает четкое значение выходной переменной, которое посылается к двигателю маятника.

Выше отмечалось, что в теории нечетких множеств еще слабо разработаны формализованные методы построения базы правил (знаний). Укажем на возможности преодоления этого недостатка с помощью нейронных сетей [8,9]. Комбинация ИНС и НЛ дает шанс получить решение задачи по разработке и регулировке систем нечеткого управления. Рассмотрим более подробно несколько вариантов такого управления.

Регулировка параметров нечеткого управления нейронными сетями

Выбор конкретной функции принадлежности, представляющей лингвистический терм, более или менее произволен. Обычно эксперт имеет некоторые соображения о диапазоне изменения ФП, но не в состоянии аргументировать малые изменения в этом диапазоне. Вследствие этого регулировка ФП становится важной задачей при построении нечетких систем. Такая задача может рассматриваться как оптимизационная, поэтому ИНС дают возможность ее решения.

Прямолинейный путь заключается в принятии допущения об определенной форме для ФП, зависящей от ряда параметров, которые могут быть обучены посредством ИНС. Например, ФП симметричной треугольной формы зависит от двух параметров: абсциссы положения максимума и длины основания треугольника, значения которых получаются в результате обучения ИНС. Такой подход требует наличия обучающего множества в виде пар вход – выход и набора правил, включающих определение соответствующих ФП.

Для ФП произвольной формы метод модифицируется следующим образом. Обучающая выборка должна быть разделена на r непересекающихся кластеров $R_i,\ldots, R_r$. Каждый кластер R_i соответствует правилу v_i. Элементы кластеров представляют собой значения пар вход-выход вида (x, y), где $x = (x_1,\ldots,x_n)$ определяет вектор n входных переменных. Это означает, что правила определяются не в форме лингвистических переменных, а в виде четких пар вход-выход.

Многослойный персептрон, состоящий из n входных нейронов, скрытого слоя и r выходных нейронов, может быть использован для обучения этих кластеров. Входные данные для этой задачи – входные вектора всех кластеров, т. е. множество вида

$$\{x \mid \exists i \ \ \exists y : (x,y) \in R_i\}.$$

Требуемый выход $t_{y_i}(x)$ на выходной ячейке y_i определяется следующим образом:

$$t_{y_i}(x) = \begin{cases} 1, & \text{если } (x, y) \in R_i, \\ 0, & \text{в противном случае.} \end{cases}$$

После того как сеть обучилась путем адаптации весов, в качестве входов могут быть взяты произвольные значения x. Затем выход на выходном нейроне y_i может быть интерпретирован как степень, с которой x отвечает предпосылке правила υ_i, т. е. функция

$$x \to o_{y_i}$$

является ФП для НМ, представляющего лингвистический терм в левой части правила υ_i.

Нейронные сети для выделения нечетких правил

В случае, если доступно множество пар вход-выход, то можно попытаться выделить нечеткие правила из этого множества. Это достигается посредством обучения нейронных сетей.

Входные вектора обучающих пар рассматриваются как входы самоорганизующейся сети Кохонена, которые интерпретируются в виде лингвистических переменных. Основная идея интерпретации заключается в отказе от принципа «Победитель получает все» после того, как веса самоорганизующейся сети установлены в процессе обучения. Таким образом, каждая выходная ячейка сети соответствует предпосылке нечеткого правила. В зависимости от того, как далеко находится выходной нейрон y_i от нейрона – победителя данного входного вектора x, определяется степень $\mu_i(x)$, с которой x удовлетворяет предпосылке соответствующего правила.

Нейронные сети и нечеткое управление

В случае комбинации искусственных нейронных сетей и нечеткого управления основная цель заключается в сохранении структуры нечеткого контроллера, которая определяется нечеткими правилами. Такие правила представляются как часть структурных знаний, определяющих управляемую систему, которую можно назвать нейро-нечетким контроллером.

Рассмотрим систему, которая может управляться одной переменной C, а состояние системы описывается двумя параметрами X_1 и X_2.

Пример такой системы показан на рис. 2.24.

Модули X_1 и X_2 представляют собой входные переменные и посылают свои четкие значения к μ-модулям, которые содержат соответствующие ФП, интерпретируемые в данном случае как лингвисти-

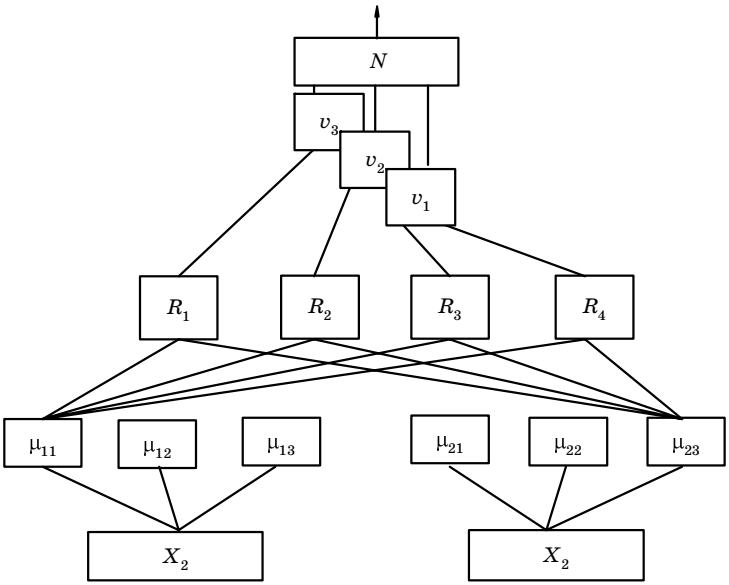


Рис. 2.24

ческие значения, присвоенные соответствующим входным переменным. μ-модули соединены с последующими *R*-модулями, являющимися нечеткими правилами вида *если*..., *то* и формирующими базу знаний. Каждый μ-модуль посылает сигналы ко всем соединенным с ним *R*-модулям. Для каждого μ-модуля возможно соединение с несколькими *R*-модулями. Последние используют *t*-норму для вычисления конъюнкции их входов и направляют полученное значение к одному из υ-модулей, которые содержат функции принадлежности, отображающие лингвистические значения выходной переменной. Проходя через υ-модули, эти значения изменяются до заключения соответствующего правила. Это означает, что выполнена импликация для получения необходимого заключения, которое в общем случае является НМ. Заключения затем проходят к *C*-модулю, где они агрегируются с использованием операции максимума, и четкий выход управляемого воздействия определяется путем дефазификации. При использовании в данной ситуации монотонной ФП, при которой дефазификация алгоритма Цукамото достигается вычислением обратной к ФП величины, υ-модули пропускают пары $(r_i, \upsilon_k^{-1}(r_i))$ к *C*-модулю, где окончательно определяется выходная величина в виде:

$$c = \frac{\sum_{i=1}^{n} r_i \upsilon_{R_i}^{-1}(r_i)}{\sum_{i=1}^{n} r_i},$$

где n – число правил; r_i – степень, с которой правило R_i активируется.

Как видно из рис. 2.24, такая система представляет собой прямонаправленную нейронную сеть. X-, R- и C-модули могут рассматриваться как нейроны сети, а μ- и υ-модули – в качестве адаптируемых весов. X- и C- слои идентифицируются как входной и выходной слои сети, соответственно; R-слой – скрытый слой сети. Тот факт, что один μ-модуль соединен более, чем с одним R-модулем, эквивалентен соединениям весов в нейронной сети.

Среди специальных приемов для формирования базы правил заслуживает внимания применение программных продуктов класса «Data Mining» («Извлечение знаний из данных») [10]. Такой подход в последнее время начинает стремительно развиваться и выделяется в отдельное направление в сфере информационных технологий. В основе использования таких приемов лежит построение деревьев решений, что является самым распространенным способом к выявлению и изображению логических закономерностей. Применяемый в таких задачах алгоритм типа ID3 (Interactive Dixotomizer – интерактивный дихотомайзер) циклически разделяет обучающие примеры на классы в соответствии с переменной, имеющей наибольшую классифицирующую силу. Каждое подмножество объектов, выделяемое такой переменной, вновь разбивается на классы с использованием следующей переменной с наибольшей классифицирующейся способностью и т. д. Разделение завершается, когда в подмножестве оказываются объекты только одного класса. В ходе процесса образуется дерево решений. Пути движения по дереву с верхнего уровня на самые нижние определяют логические правила в виде цепочек конъюнкций.

Одной из таких систем для обнаружения знаний в базах данных является программа `See-5` компании RuleQuest. Результат работы программы представляется в виде деревьев решений. Система проста в обращении и не требует от пользователя обширных знаний в области статистики и принятия решений. Демонстрационная версия этой системы ограничена количеством анализируемых объектов (до 200). Иногда полученное дерево решений оказывается слишком сложным для восприятия. В этом случае в программе предусмотрена возмож-

ность преобразования дерева в набор правил вида *если...*, *то*. Последнее обстоятельство в наибольшей мере подходит для систем НЛ при формировании базы правил и позволяет формализовать один из сложных этапов решения таких задач.

Проиллюстрируем применение программы See-5 на примере классификации кредитных карт. В рассматриваемом примере все названия признаков и их значения обозначены лишь символами для защиты конфиденциальности кредитных карт, указывается только принадлежность каждого из 15 признаков к порядковой или количественной шкалам. Применительно к данному примеру получен набор, состоящий из семи правил следующего вида:

```
Extracted rules:
Rule 1: (81/2, lift 1.3)
A15 > 225
-> class + [0.964]
Rule 2: (7, lift 1.2)
A5 = p
A14 > 311
-> class + [0.889]
Rule 3: (119/45, lift 0.8)
A15 <= 225
-> class + [0.620]
Rule 4: (12, lift 4.0)
A5 = p
A10 = f
A14 <= 311
A15 <= 225
-> class - [0.929]
Rule 5: (6, lift 3.7)
A7 = bb
A10 = f
A12 = t
-> class - [0.875]
Rule 6: (26/7, lift 3.0)
A7 = v
A10 = f
A14 > 102
A15 <= 50
-> class - [0.714]
Rule 7: (1, lift 2.8)
A7 = j
```

```
A10 = f
-> class - [0.667]
```

Каждое правило определяется такими фрагментами:

– номер правила;

– статистики $(n, lift\ x)$ или $(n/m, lift\ x)$, где n – число обучающихся объектов, перекрывающихся данным классом; m – число объектов, не принадлежащих классу, предсказанному правилом; точность правила определяется отношением $(n - m + 1) / (n + 2)$; $lift\ x$ – частное от деления точности правила на относительную частоту предсказанного класса в обучающей выборке;

– одно или больше условий, входящих в состав правила;

– класс, предсказанный правилом;

– величина, принимающая значения от 0 до 1, которая выражает степень доверия к правилу.

Кроме получения правил, можно оценить величину ошибочной классификации, которая представляется в таком виде:

```
Evaluation on training data (200 cases):
(a) (b) <-classified as
148 5 (a): class +
16 31 (b): class -
```

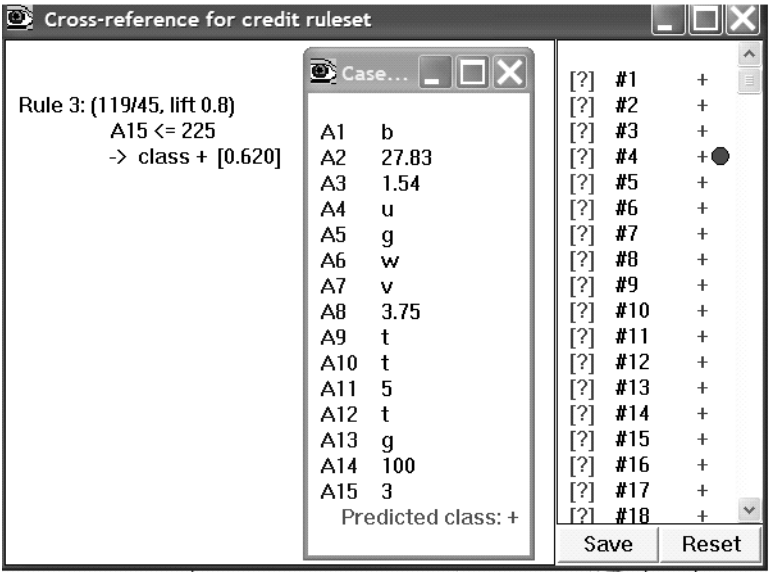


Рис. 2.25

Из приведенных данных видно, что по обучающей выборке, состоящей из 200 объектов к первому классу, обозначенному как «а», правильно отнесены 148, а ошибочно – 16. Аналогичный вывод можно сделать и относительно второго класса.

Для детального рассмотрения множества правил можно использовать окно перекрестных ссылок, как показано на рис. 2.25.

Как видно из этого рисунка, объект №4 относится к классу с меткой (+), так как для него выполняется условие $A15 \le 225$.

2.8. Фазификация временных рядов

Финансовые временные ряды трудны для анализа традиционными средствами. Основные недостатки для применения методов НЛ в финансовых приложениях сводятся к следующим:

– большинство методов не формируют ФП и не выделяют нечетких правил;

– финансовые ряды обычно не стационарны, и поэтому требуют модификации ФП в части отображения последними значений временных рядов в группы или сегменты.

Часто не ясно, сколько ФП и какой формы требуется для отображения временных рядов. Одним из путей решения этих вопросов является применение кластеризационных методов для выявления формы ФП из временных рядов. Возможные формы ФП зависят от естественного пути представления параметров. Во временных рядах необходимо уделять внимание выпуклым функциям и сконцентрироваться на неубывающих и невозрастающих функциях, так как они наиболее естественны для финансовых рядов. Несмотря на то что часто в качестве ФП используются линейная (треугольная) форма, гауссова и сигмоидальная формы, возможно, дадут лучшие результаты.

Рассмотрим два метода кластеризации данных из временных рядов [11].

П е р в ы й м е т о д состоит из двух этапов:

– формирование кластеров;

– построение ФП, отображающих каждый кластер.

Значения ФП выводятся с помощью известного метода k-средних, используемого в кластеризационных процедурах, который состоит из следующих шагов:

– выбор числа классов C;

– сортировка данных по классам;

– вычисление центров каждого класса $\mu_1,\ldots, \mu_{C,}$ допуская, что классы имеют одинаковый размер;

– классификикация каждого значения ряда x_k, используя среднеквадратичную ошибку;

– пересчет значения центров μ_i, используя результаты шага 4;

– в случае устойчивых значений μ_i – остановка алгоритма; в противном случае – переход к шагу 4.

На основе этого алгоритма была предложена следующая аппроксимация при построении ФП:

$$C = 0{,}5(S+E); \quad K = 2(1-M)/(E-S); \quad F_i = \max(0{,}1 - K|X_i - C|,$$

где M – значение ФП на границах; S – левая граница; E -правая граница; C – центральное значение; K – масштабный коэффициент; X_i – входная величина; F_i – текущее значение ФП.

Такой подход позволяет двум соседним ФП пересекаться на уровне 0,5. Однако этот метод обладает двумя недостатками:

– пользователь должен предварительно определить число ФП;

– степень перекрытия соседних сегментов ничем не обусловлена.

Второй метод находит число кластеров, которые лучше всего представляют исходные данные. Оптимальным числом кластеров является такое, при котором данные из одного и того же кластера находятся близко друг к другу, а данные из разных кластеров – далеко друг от друга. Такое число получается путем нахождения локального минимума величины

$$S(c) = \sum_{k=1}^{n}\sum_{i=1}^{c}(\mu_{ik})^m\left(\left\|X_k - \nu_i\right\|^2 - \left\|\nu_i - \overline{x}\right\|^2\right),$$

где n – число данных, подвергаемых кластеризации; c – число кластеров ($c \geq 2$); X_k – вектор данных; $\overline{x}$ – среднее значение; n_i – вектор, определяющий центр i-го кластера; μ_{ik} – степень, с которой k-е данные принадлежат i-му кластеру; m – регулируемый вес.

Этот метод минимизирует дисперсию индивидуальных значений x_k внутри каждого кластера и максимизирует дисперсию между кластерами. Перед минимизацией величины $S(c)$ должны быть сформированы c кластеров и найдены в первом приближении их ФП. Последние вводятся в матрицу U размером $n \times c$, к которой применяется метод нечеткой кластеризации. Сущность предлагаемого метода заключается в сведении ФП, содержащейся в матрице U, к трапецеидальному виду. Алгоритм метода нечеткой кластеризации, в основе которого лежит способ k-средних, состоит из следующих пяти шагов:

– определяется и фиксируется величина c; $2 \leq c \leq n$;

– инициируется матрица $U^{(l)}$ таким образом, что

$$\sum_{i=1}^{c} \mu_{ik} = 1,$$

где $l = 0, 1, 2\ldots$; $i = 1, 2,\ldots, c$; k – k-й вход;

– вычисляются центры нечетких кластеров v_i:

$$v_i = \frac{\sum_{k=1}^{c} (\mu_{ik})^m X_k}{\sum_{k=1}^{n} (\mu_{ik})}, \quad 1 \le i \le c;$$

– матрица $U^{(l)}$ изменяется следующим образом:

$$\text{если } d_{ik} = \left\| X_k - v_i \right\| \neq 0 \text{ для всех } 1 \le i \le c, \text{ то } \left[\sum (d_{ik} / d_{jk})^{2/(m-1)} \right]^{-1},$$

$$\text{в противном случае } \mu_{ik} = 0 \text{ и } \sum \mu_{ik} = 1;$$

– если $\left| U^{(l)} - U^{)l-1)} \right|$ окажется больше некоторого порога, необходимо вернуться к шагу 3; в противном случае алгоритм завершается.

Приведенный алгоритм вначале вычисляет центры каждого кластера (шаг 3). Затем определяется расстояние между каждым X_k и центром кластера. В случае, если это расстояние отлично от нуля, изменяется параметр μ_{ik} (шаг 4). Если X_k и v_i оказываются равными, то μ_{ik} имеет нулевое значение, а степень принадлежности данных, характеризуемых вектором X_k, распределяется среди других μ_{ik}. На последнем шаге осуществляется проверка сходимости матрицы U. Если матрица не происходит, то процесс повторяется.

При повторении процесса вычисляется $S(c)$ и сравнивается с $S(c+1)$. В случае, если $S(c)$ оказывается больше $S(c+1)$, то c увеличивается на единицу, и процесс повторяется, начиная с шага 2. В противном случае $S(c+1)$ достигает локального минимума, и процесс завершается.

2.9. Нейро-нечеткие системы

Гибридизация нейронных сетей с нечеткой логикой позволяет существенно повысить эффективность работы таких нейро-нечетких систем за счет того, что недостатки, присущие одной из технологий, компенсируются преимуществами другой. В частности, ИНС хорошо распознают образы, но процесс работы обученной сети сложен для понимания. В то же время системы НЛ хорошо объясняют выво-

ды, но имеют ограничения на количество входных переменных. Вследствие этого возможно построение гибридных нейро-нечетких систем, в которых выводы формируются на основе НЛ, а ФП подстраиваются с помощью ИНС. Преимущество таких систем очевидно: построенная структура является логически прозрачной.

Рассмотрим вначале обычный многослойный персептрон, в который будем вводить нечеткие величины на входной и выходной слои [12]. Левая часть правил систем НЛ (предпосылка) генерируется при обратном распространении через сетевые веса, а правая часть (заключение) – через выходы сети. Алгоритм обучения, используемый для нечеткого персептрона, является расширением традиционного метода ОРО, только вместо среднеквадратичной ошибки применяется нечеткая ошибка, которая больше подходит для данных, включающих нечеткости между классами. Входные данные фазифицируются посредством ФП.

В отличие от обычного персептрона эта модель способна иметь дело с входными переменными, выраженными в лингвистической форме. Гибридная сеть, как и обычная ИНС, проходит через две стадии: обучение и тестирование. На стадии обучения используется супервизорное обучение для присвоения выходных значений ФП из интервала [0,1] обучающим векторам. Следовательно, каждому нейрону выходного слоя может быть присвоено ненулевое значение ФП вместо выбора единственного нейрона с наибольшей активацией. Это позволяет моделировать нечеткие данные, когда признаковое пространство включает перекрывающиеся классы так, что точка образа может принадлежать более чем одному классу с ненулевой ФП. При обучении каждая ошибка в заданной ФП вводится обратно в сеть, и на обратном проходе изменяются веса сети. Ошибка обратного распространения вычисляется по отношению к каждому требуемому выходу, которым является значение ФП, определяющего степень принадлежности входного вектора к рассматриваемому классу. После ряда циклов такая сеть будет сходиться к решению с минимальной ошибкой.

Алгоритм обучения можно описать следующим образом.

Рассмотрим сеть обратного распространения, в которой общий вход x_j^{h+1}, принятый нейроном j в слое $h+1$, определяется как

$$x_j^{h+1} = \sum_i y_i^h w_{ji}^h \quad -\theta_j^{h+1},$$

где y_i^h – состояние i-го нейрона в предшествующем скрытом слое; w_{ji}^h – вес от нейрона i в слое h к j-му нейрону в слое $h+1$; θ_j^{h+1} – порог j-го нейрона в слое $h+1$.

Выход нейрона в любом слое, кроме первого, определяется монотонной нелинейной функцией в виде сигмоиды от общего входа слоя и имеет следующий вид:

$$y_j^h = 1/[1 + \exp(-x_j^h)].$$

Вследствие того что выходные нейроны сети представляют значения ФП в диапазоне от 0 до 1 и не являются бинарными, обычная функция ошибки не может быть использована. Вместо этого здесь используется функция ошибки в виде:

$$E_p = \frac{1}{2}\sum_p \sum_k \mu(x_p)(y_{pk} - \theta_{pk})^2,$$

где $\mu(x_p)$ – ФП для входного вектора x_p.

Алгоритм, как и в классическом варианте нейронных сетей, состоит из таких же шагов и принципиально не отличается от описанного ранее, за исключением, естественно, вида функции ошибки.

В противоположность описанной разновидности нейро-нечеткой системы была предложена адаптивная нечеткая нейронная система вывода (Adaptive Neuro-Fuzzy Inference System -ANFIS) [13], в которой использовался алгоритм Сугено в качестве системы вывода. При рассмотрении ANFIS для упрощения допустим, что нечеткая система имеет два входа x, y и один выход f. Пусть база правил содержит всего лишь два правила вида: *если*..., *то*:

П1: если x есть A_1 и у есть B_1, то $f_1 = p_1x + q_1y$;

П2: если x есть A_2 и у есть B_2, то $f_2 = p_2x + q_2y$.

Соответствующая системе `ANFIS` архитектура нейронной сети показана на рис. 2.26.

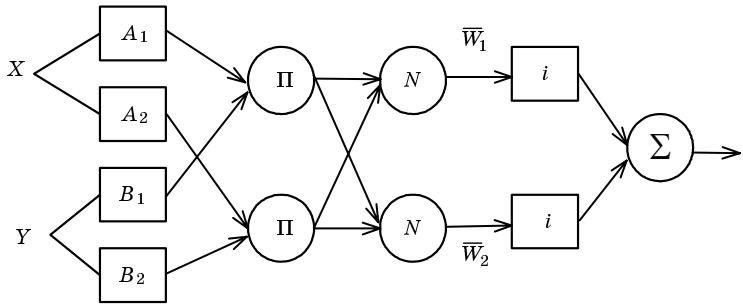


Рис. 2.26

Каждый слой сети выполняет следующие функции:

1 - й с л о й

Каждый нейрон этого слоя (обозначен квадратом на рис. 2.26) есть величина, равная

$$O_i^1 = \mu_{A_i}(x),$$

где x – вход в нейрон i; A_i – лингвистическая переменная (малая, большая и т. п.), ассоциированная с функцией нейрона; O_i^1 – ФП переменной A_i, определяющая степень, с которой x удовлетворяет переменной A_i. В первоначально предложенном варианте ФП $\mu_{A_i}(x)$ были выбраны в виде колоколообразной кривой с максимумом, равным 1, и минимумом, равным 0. Формулы, отображающие такую кривую, имеют следующий вид:

$$\mu_{A_i}(x) = \left\{1 + \left[\left(\frac{x - c_i}{a_i}\right)^2\right]^{b_i}\right\}^{-1};$$

$$\mu_{A_i}(x) = \exp\left[-\left(\frac{x - c_i}{a_i}\right)^2\right],$$

где a_i, b_i, c_i – устанавливаемые параметры.

При изменении этих параметров, соответственно, будут меняться функции $\mu_{A_i}(x)$, определяющие различные формы ФП переменной A_i. Иными словами, можно сказать, что выходы нейронов этого слоя представляют собой значения ФП при конкретных значениях входных переменных.

2 - й с л о й

Каждый нейрон этого слоя (обозначен кружком П) перемножает входящие сигналы и посылает полученное произведение в следующий слой. Например:

$$W_i = \mu_{A_i}(x) \cdot \mu_{B_i}(x),\ i = 1;\ 2.$$

Каждый выход этого слоя определяет активизацию правила.

3 - й с л о й

Нейрон этого слоя (обозначен кружком N) вычисляет отношение величин:

$$\overline{W_i} = \frac{W_i}{W_1 + W_2}, \qquad i = 1; 2.$$

Выходы этого слоя могут быть названы нормированной активацией правила.

4-й слой

Нейрон с индексом i (обозначен квадратом) данного слоя выполняет следующую операцию:

$$O_i^4 = \overline{W_i} \cdot f_i = \overline{W_i} \cdot (p_i x + q_i y).$$

5-й слой

Единственный нейрон этого слоя (обозначен кружком) вычисляет итоговый выход как сумму всех входящих сигналов:

$$O_i^5 = \sum_i \overline{W_i} \cdot f_i.$$

Такая адаптивная сеть функционально эквивалентна нечеткой системе вывода Сугено. Разработчик такой системы J. S. Jang предложил применять ее для имитации функций, прогнозирования и обработки сигналов. Корректировка параметров системы в такой сети производится посредством специально разработанного гибридного обучающего алгоритма.

Отметим, что выход нейрона во втором слое ANFIS представляет собой операцию «логического И» ФП каждой лингвистической переменной первого слоя. Каждый слой ANFIS эквивалентен шагу вычислений нечеткой системы вывода. Функция активации в каждом слое соответствует шагу нечеткого вывода. База знаний нечетких правил эквивалентна знаниям, находящимся в соответствующей ANFIS. При разработке такой системы предполагалась фиксированной ее структура, а идентификация параметров осуществлялась гибридным правилом. Другая важная проблема структурной идентификации, которая касается выбора приемлемого разделения входного пространства и количества ФП на каждом входе, еще не решена. Эффективное разделение входного пространства может уменьшить число правил и, следовательно, увеличить скорость работы при обучении и применении.

2.10. Программные пакеты в области нечеткой логики

Для решения задач с применением систем нечеткой логики существует достаточное количество программных продуктов, различающихся степенью сложности и предоставляемыми возможностями. Укажем некоторые из них.

Пакет CubiCalc, разработанный американской фирмой HyperLogic в начале 90-х гг. XX в. Следует отметить, что до 1995 г. экспорт этого программного продукта в Россию был запрещен аме-

риканским Комитетом по контролю за экспортом (COCOM), так как системы с НЛ использовались в американской программе «Стратегической оборонной инициативы». Включенная сейчас в пакет в качестве иллюстративного примера задача о собаке, догоняющей кота, есть не что иное, как задача о ракете и противоракете. Используя ограниченный набор входных переменных (координаты собаки и кошки, угол между направлениями их движений) и единственную выходную переменную (величину угла перемещения собаки), базу знаний, состоящую всего из пяти правил, программа успешно решает поставленную задачу. После того как быстро бегущая «собака» стала догонять межконтинентального «кота», скептицизм, проявляемый противниками НЛ, резко уменьшился, а системы с НЛ были включены в оборонные программы США. Сегодня CubiCalc применяется при решении различных задач: от адаптивного управления оптовыми складами до моделирования рынка фьючерсных контрактов.

Этот пакет содержит средства для ввода и представления данных, формирования правил вывода, описания нечетких множеств. В некоторые версии пакета включается модуль RuleMaker, позволяющий решать одну из основных проблем в системах с НЛ: автоматическое построение нечетких правил. Пакет позволяет контролировать каждый шаг вычислений, генерировать на языке Си тексты, содержащие алгоритмы работы нечеткой системы, которые затем могут быть встроены в приложения пользователя.

Процесс выполнения любого проекта в пакете CubiCalc состоит из нескольких шагов:

– инициализация;
– ввод данных;
– предобработка;
– выполнение правил;
– постобработка;
– вывод данных;
– моделирование процесса.

Краткая иллюстрация работы этого пакета приведена в [14].

Модуль Fuzzy Logic Toolbox из системы Matlab. Этот модуль представляет собой пакет расширения системы Matlab, которая была разработана американской фирмой MathWorks для технических приложений. Одним из достоинств системы Matlab является возможность ее расширения с целью решения новых, возникающих задач. Описываемый здесь модуль Fuzzy Logic Toolbox используется для решения задач в области НЛ.

Основные возможности этого пакета:

– построение систем нечеткого вывода (экспертных систем, аппроксиматоров зависимостей, различных регуляторов);

– формирование адаптивных нечетких систем (гибридных нейронных сетей);

– интерактивное динамическое моделирование.

Как описывалось выше, решение задач на основе НЛ сводится к выполнению ряда последовательных действий, в частности: выбор входных и выходных переменных, формирование базы правил для оцениваемой системы, построение выходного нечеткого множества и его дефазификация с целью получения четкого выхода системы. Пакет `Fuzzy Logic Toolbox` позволяет выполнить перечисленные операции, графически отображая промежуточные и итоговые результаты. Пример использования этого пакета для решения конкретной задачи по оценке потенциала балтийских портов приведен в следующем параграфе.

Пакет WINROSA. Разработан в Германии в начале 90-х гг. Работа с этим пакетом представляет собой следующую последовательность шагов:

– определение проекта;

– формирование правил;

– уменьшение количества правил;

– анализ результатов;

– экспорт результатов.

Рассмотрим эти шаги более подробно.

Для работы с этим пакетом вначале необходимо определить входные и выходные переменные. Входные переменные представляют собой предпосылки правил «*если*..., *то*», а выходные – заключения этих правил. Необходимо иметь результаты наблюдений для каждой переменной. Лингвистические переменные и ФП должны быть определены перед формированием правил.

На втором шаге пользователь должен выделить максимальное число значимых лингвистических соотношений, что позволяет оценить величину пространства поиска. При этом необязательно обобщать в одном правиле все входные переменные. Для малых пространств поиска возможен метод полного перебора, для больших – необходимо применять эволюционные процедуры поиска. На этом этапе избавляются от правил, которые полностью перекрываются лучшими правилами. Кроме того, каждому правилу придают вес (между нулем и единицей), который показывает, как хорошо данное правило подтверждается имеющимися данными. Здесь

формируются не только положительные правила, которые выражают рекомендации, но и отрицательные, определяющие запрещения и предостережения.

Результатом второго шага является множество приемлемых правил. Так как оно состоит из обобщающих правил, могут иметь место различные перекрытия правил и даже конфликты между последними. Хотя подобные явления разрешаются при дефазификации, рекомендуется на третьем шаге уменьшить, насколько это возможно, количество используемых правил. Это приводит к большей прозрачности решения и сокращению времени вычислений.

На четвертом шаге при анализе результатов правила должны быть проверены на правдоподобие. При этом важную роль играет вес правила. Высокие значения этой величины свидетельствуют о явной зависимости между переменными, в то время как малые – о противоречивости переменных в данном правиле. Для получения быстрого обзора базы правил полезно применить графическое представление. При рассмотрении, например, всех правил, которые содержат определенную входную переменную, важность этой величины может быть найдена из графического отображения. Во многих приложениях необходимо, чтобы, по крайней мере, имелось одно правило для каждой входной ситуации.

При экспорте результатов необходимо учитывать, что файл `WINROSA` сохраняется как `ASCII`-файл с расширением «`*.ini`». Для прямого использования результатов в имитационных задачах экспортируемые файлы записаны для `DORA for Windows 6.0`; `Matlab` и `fuzzyTECH`.

Рассмотрим пример использования этого пакета в задаче регулирования температуры в одном из двух сосудов, обменивающихся теплом. В распоряжении автора имелась только демоверсия этого пакета, поэтому решать конкретную задачу невозможно. Тем не менее сущность работы с этим пакетом можно продемонстрировать и на таком примере.

Имеются данные по четырем переменным: значения температуры в сосудах 1 и 2, давление в сосуде 2 и управляемая переменная для регулирования температуры в сосуде 1. Необходимо сформировать правила, которые описывают, как температура в сосуде 1 зависит от управляемой переменной для сосуда 1, температуры и давления в сосуде 2.

Определение проекта. В данной задаче измеренные величины используются без каких-либо дополнительных преобразований. Входными переменными являются:

– управляемая переменная для регулирования температуры в сосуде 1 (act1);

– температура в сосуде 2 (temp2);

– давление в сосуде 2 (press2).

Выходная переменная:

– температура в сосуде 1 (temp1).

Данные измерений включают записи за период, равный 2250 секунд, с шагом дискретизации, составляющим 2 с. Используя эти данные, можно определить входные и выходные переменные в соответствующих окнах. Пример такого окна для переменной act1 показан на рис. 2.27.

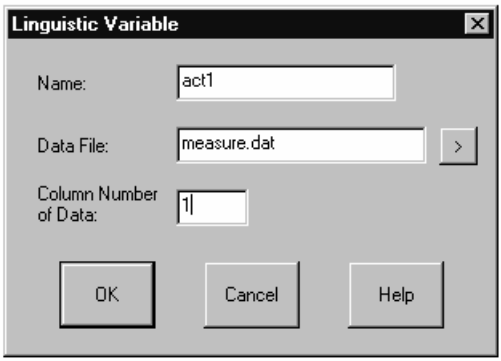


Рис. 2.27

После определения всех переменных осуществляется переход к диалоговому окну построения ФП (рис. 2.28). Для нахождения ФП выберем способ их автоматического формирования нажатием кнопки «→» справа от перечня переменных. В табл. 2.3 приведены необходимые для построения ФП величины.

Таблица 2.3

Переменная	Лингвистические значения	Левая граница	Правая граница	Форма функции принадлежности
act1	9	15	120	Треугольная
temp2	6	50	190	Треугольная
press2	4	1	3,2	Треугольная
temp1	9	60	100	Трапециевидная

На рис. 2.29 приведено окно формирования ФП для переменной act1.

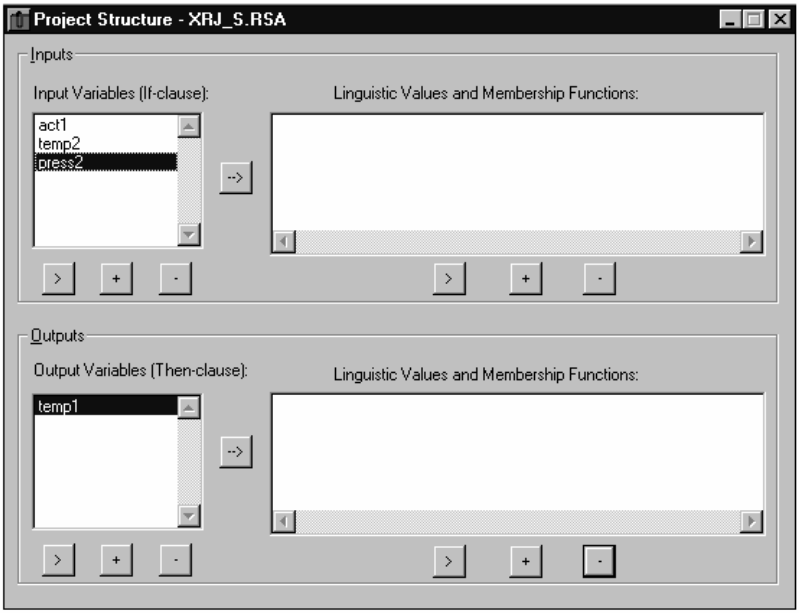


Рис. 2.28

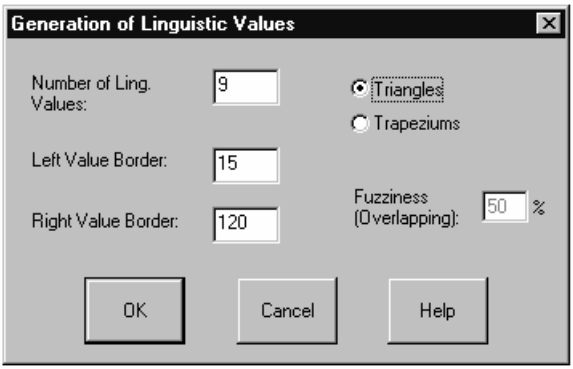


Рис. 2.29

При выборе треугольной формы средние в диапазоне ФП представляют собой равнобедренные треугольники. В случае трапециевидной формы ФП имеется параметр, определяющий степень нечеткости. С помощью этого параметра можно выразить долю диапазона со значением ФП, меньшим единицы. Если значение параметра составляет

0%, то получаем ФП с вертикальными боковыми границами; при 100% значении параметра формируются треугольные ФП.

После определения ФП для всех переменных приходим к следующему результату (рис. 2.30):

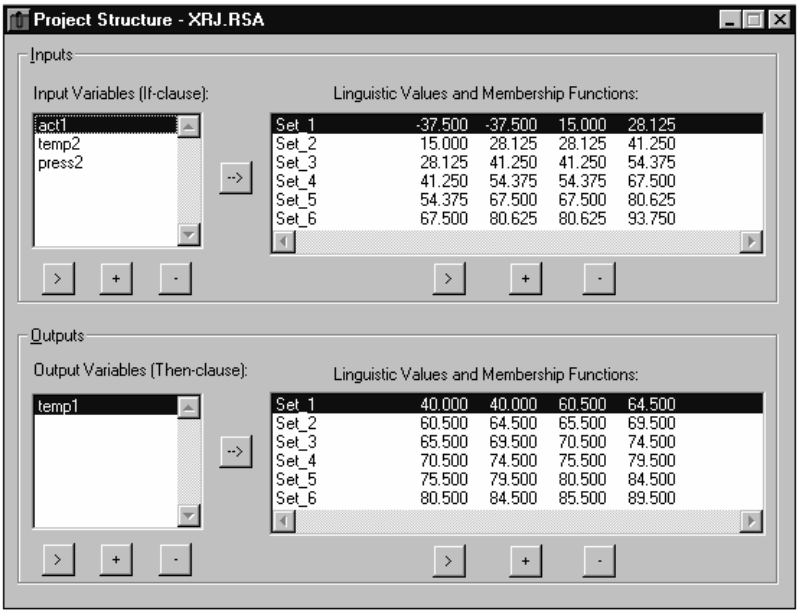


Рис. 2.30

Генерирование правил. Выбором опции «Generation» из меню или нажатием соответствующей кнопки в основном окне вызывается окно генерирования правил, и нажатием кнопки «Start» инициируется процесс. Результат формирования правил показан на рис. 2.31.

После исключения недопустимых и некачественных правил в задаче остается 83 правила. Далее можно выбрать опцию быстрого обзора «Quick Overview» из главного меню и получить набор положительных правил, число которых составляет 55 (рис. 2.32).

Уменьшение числа правил. Рассмотрение в базе правил конфликтных ситуаций, выявленных пользователем, а также обусловленных структурными и ситуационными особенностями правил, позволяет сократить количество используемых правил до 14. Диалоговое окно снижения правил показано на рис. 2.33.

Итог последовательного применения указанных в окне (рис. 2.33) опций приводит к следующему результату (рис. 2.34).

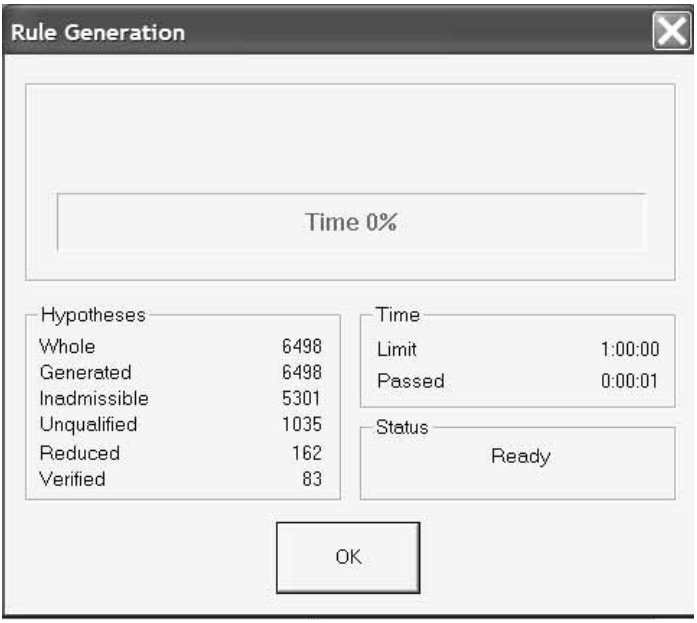


Рис. 2.31

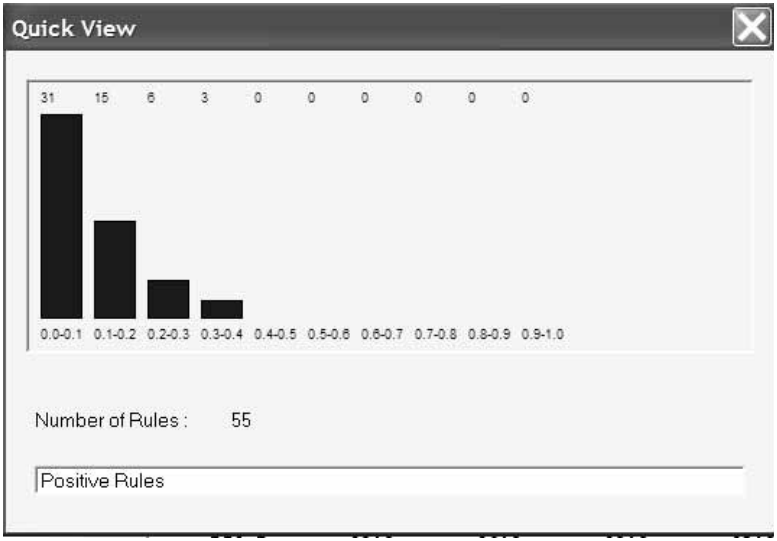


Рис. 2.32

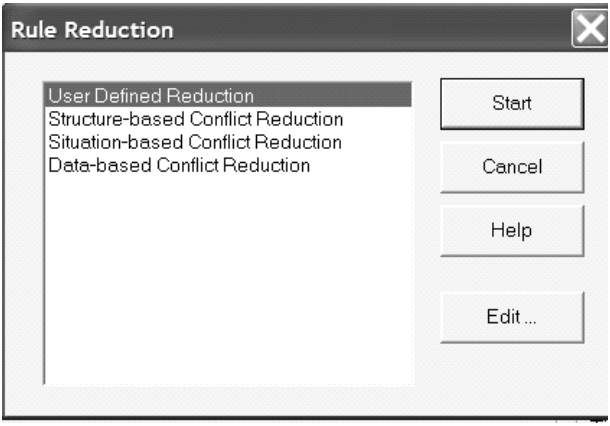


Рис. 2.33

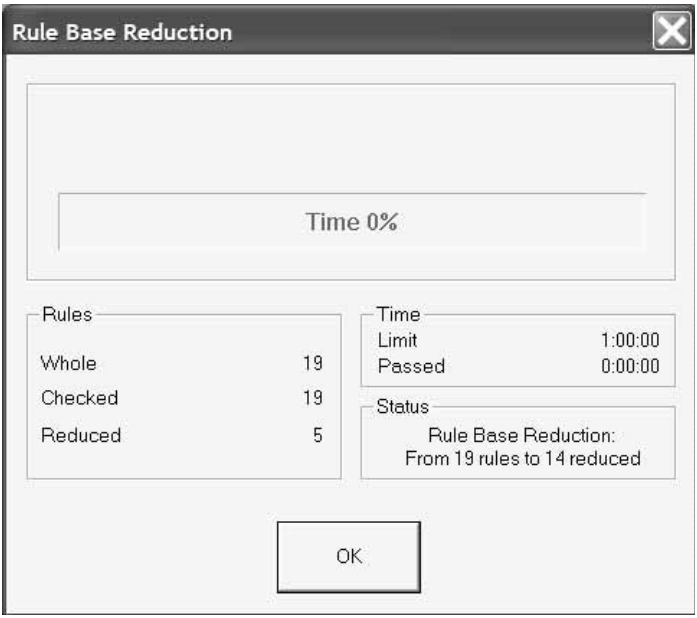


Рис. 2.34

Анализ результатов. Опция «`Analysis`» из главного меню вызывает окно методов анализа (рис. 2.35), используя которое можно перейти к просмотру правил (рис. 2.36, *а, б*).

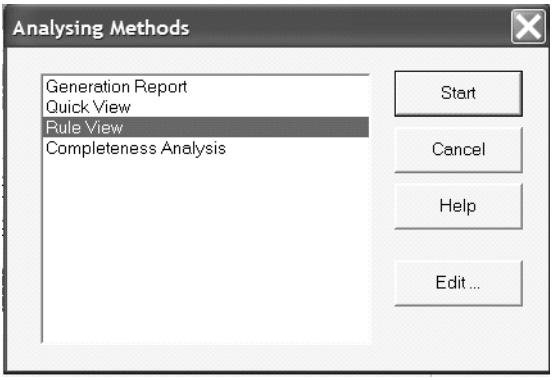
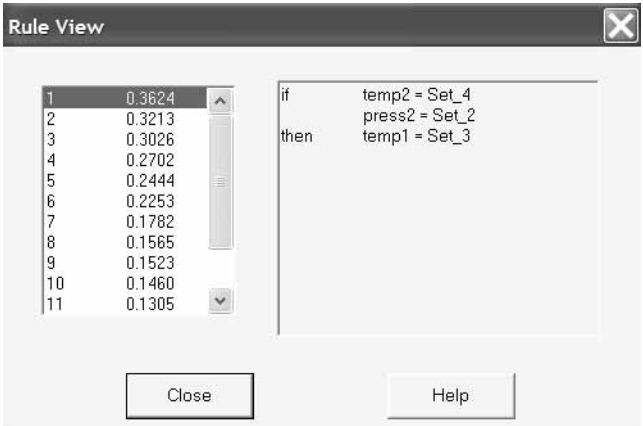


Рис. 2.35

а)



б)

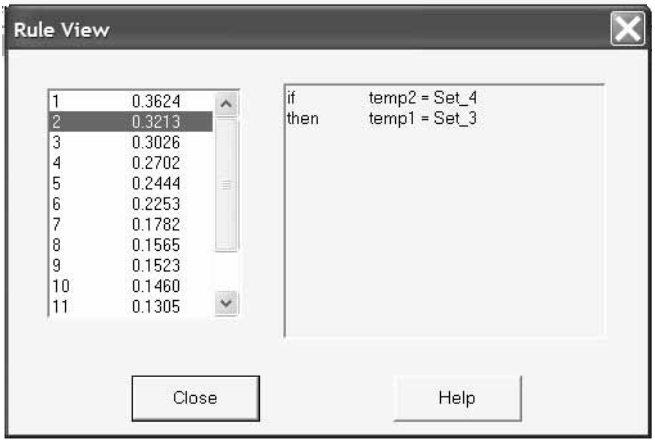


Рис. 2.36

В левой части окна вместе с порядковым номером правила указывается его вес; в правой – само правило вида «*если*..., *то*».

Экспорт результатов. В качестве примера рассмотрим экспорт результатов в систему `Matlab`. С помощью созданного файла с расширением (`*.fis`) можно связать нечеткую систему вывода с пакетом моделирования динамических систем `SIMULINK`. Результат построения такой системы показан на рис. 2.37. После моделирования нечеткой системы на имеющихся данных можно получить зависимость, отображающую ее выход (рис. 2.38).

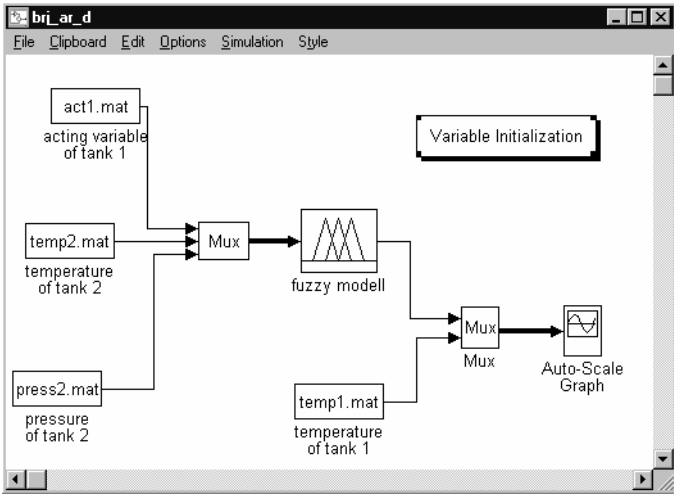


Рис. 2.37

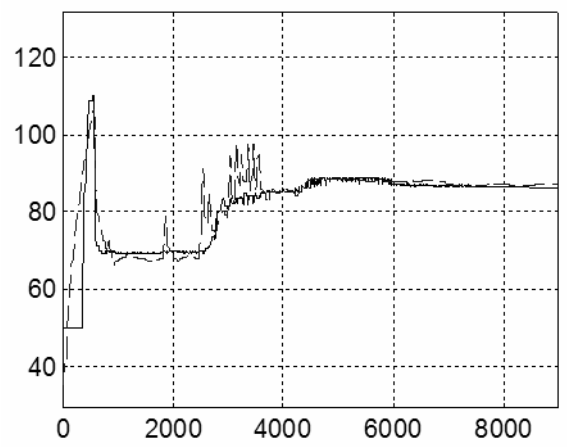


Рис. 2.38

Пакет FIDE (Fuzzy Inference Development Environment) разработки американской фирмы Aptronix представляет собой средство для создания и использования нечетких систем вывода. Ядром этой системы является компилятор, который применяет англоязычный синтаксис и обозначения для переменных, ФП и правил. В качестве входа система использует текстовой формат, что представляет значительную гибкость для пользователя. Кроме того, разработчик может воспользоваться любым текстовым редактором, включая и собственный редактор системы FIDE для создания и изменения правил. В состав пакета входит матричный редактор правил с целью более эффективной разработки правил и графический редактор отображения ФП. Система FIDE также обеспечивает полным набором средств визуализации и отладки. Пользователь имеет возможность просмотреть разрабатываемую систему нечеткого вывода в трехмерном пространстве или в любом сечении этого пространства.

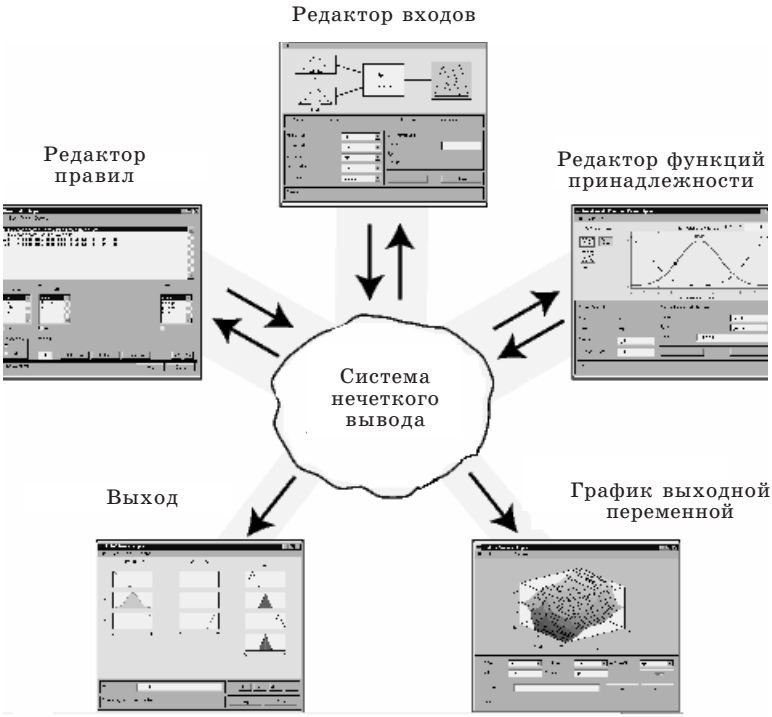


Рис. 2.39

После того как разработана отдельная ячейка нечеткого вывода, состоящая из входных, выходных переменных и набора правил, пользователь может соединить их с элементами, выполненными вне системы FIDE, для формирования интегрированной структуры. FIDE имеет в своем составе специальное средство, называемое «Compοsερ» (составитель), которое символически отображает отдельные ячейки нечеткого вывода и позволяет разработчику графически соединять входы и выходы каждой ячейки. Такой подход дает возможность осуществлять моделирование системы и графически рассматривать ее поведение.

Демонстрационная версия системы FIDE находится по адресу: http://www.aptronix.com.

Из опыта работы автора с пакетами CubiCalc и FuzzyLogic системы Matlab предпочтительнее последняя вследствие больших возможностей в части выбора вида ФП, композиции нечетких выводов и графического отображения результатов. Схема работы с этой системой показана на рис. 2.39.

2.11. Использование нечеткой логики в задачах менеджмента

Воспользуемся изложенными теоретическими положениями о нечеткой логике для решения задачи оценки потенциалов некоторых портов Балтийского моря. Такая задача возникла в связи с тем, что после распада СССР основной поток грузов из России и в Россию осуществлялся через порты бывших прибалтийских республик. Строительство новых портовых сооружений в Усть-Луге и Приморске потребовало наряду с технико-экономическим обоснованием оценить потенциал существующих портов по набору параметров. Рассматриваемая задача была решена в магистерской работе М. В. Цветковой, выполненной под руководством автора.

Зависимость от зарубежных транспортных коридоров приводит к тому, что за перевалку грузов, проходящих через порты прибалтийских республик и Финляндии, российские грузовладельцы несут дополнительные транспортные расходы, которые по разным экспертным оценкам составляют до 1 млрд USD. Помимо прямых убытков из-за недополучения налогов от компаний-экспортеров и импортеров, у которых происходит снижение прибыли за счет увеличения расходов на транспортировку, российский бюджет несет косвенные убытки.

Для оценки потенциалов портов Балтии необходимо проанализировать данные по грузообороту характерных грузов, проходящих через них (табл. 2.4).

Таблица 2.4

Порты	Виды грузов					
	нефть и нефтепродукты	металлы	химические грузы	древесина	контейнеры	roll on/ roll of
Санкт-Петербург	8146,3	7598	3684,4	7131,7	4217,2	32
Рига	26311,1	981,6	4980,9	511,3	84,9	12,3
Вентспилс	29732,4	3191,6	6557,3	6100,5	910,7	57,2
Лиепая	2831,4	1761,6	1558,9	4152,9	857,8	11,8
Клайпеда	3921,86	4203,7	2890,7	690,2	280,64	92,8
Таллинн	17181,7	5125,7	5100,1	4271,3	382,7	53

Поскольку каждый порт представляет собой многомерный (6-мерный) объект, желательно перейти к сокращенному признаковому пространству, для чего можно использовать ГК.

Метод ГК обладает рядом полезных свойств, делающих его эффективным для визуализации структуры многомерных данных. Все они касаются наименьшего искажения геометрической структуры точек (объектов) при их проектировании в пространство меньшей размерности. Первый ГК есть линейная комбинация исходных признаков, обладающая наибольшей дисперсией. Геометрически это выглядит как новая ось y_1, ориентированная вдоль направления наибольшей вытянутости эллипсоида рассеивания объектов выборки в исходном пространстве.

Наибольшую дисперсию среди всех оставшихся линейных преобразований, некоррелированных с первым главным компонентом, имеет второй ГК. Он рассматривается как направление наибольшей вытянутости эллипсоида рассеивания, перпендикулярный первому ГК и т. п. Метод ГК обладает рядом свойств, делающим его эффективным для визуализации структуры многомерных данных. Все они касаются наименьшего искажения геометрической структуры точек (объектов) при их проектировании в пространстве меньшей размерности.

Метод ГК для данных, представленных в табл. 2.4, реализуется при помощи программного пакета «Statgraphics Plus For Windows», который предназначен для решения задач с многомерными данными и на сегодняшний день является одной из наиболее эффективных систем статистического анализа данных.

Анализ ГК приведен в табл. 2.5.

Таблица 2.5

Component Number	Eigenvalue	Percent of Variance	Cumulative Percentage
1	2,49115	41,519	41,519
2	1,91592	31,932	73,451
3	1,25253	20,875	94,327
4	0,257753	4,296	98,622
5	0,0826557	1,378	100,000
6	0,0	0,000	100,000

Из последнего столбца табл. 2.5 видно, что первые три ГК в сумме дают 94,3% от общей дисперсии. Следовательно, потеря информативности при сохранении только трех ГК составляет 5,7% . Такая величина потерь вполне приемлема, так как при этом можно визуализировать многомерные данные в сокращенном трехмерном пространстве.

В табл. 2.6 приведены значения трех первых ГК для всех шести портов.

Таблица 2.6

Row	Component 1	Component 2	Component 3
1	-2,81923	0,429649	-0,391662
2	2,03748	0,2268	-0,932925
3	0,509481	1,96381	0,252101
4	-0,0856758	-1,70635	-1,29182
5	0,291603	-1,49564	1,77863
6	0,0663432	0,581729	0,585668

С использованием данных табл. 2.6 в трехмерном пространстве первых трех ГК приведены все шесть объектов – портов (рис. 2.40).

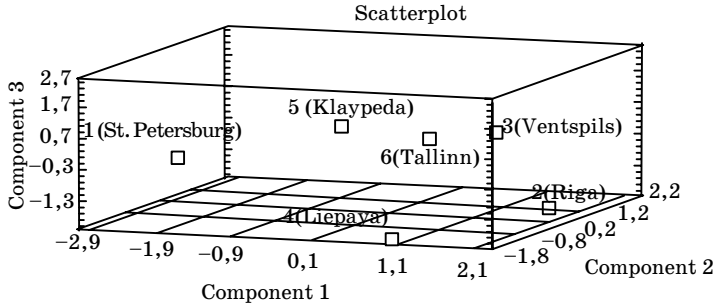


Рис. 2.40

По значениям весовых коэффициентов, участвующих в расчете ГК, следует, что на первый ГК оказывают наибольшее влияние грузооборот нефти и нефтепродуктов, на второй ГК – химические грузы, на третий ГК – грузооборот колесной техники (Roll on/Roll of, RORO). Исходя из этого, можно дать следующие названия осям трехмерного пространства: нефтяной компонент, химический компонент и компонент Roll on/Roll of.

Воспользуемся полученными результатами для построения нечеткой системы. Последняя будет иметь три входа (ГК1, ГК2, ГК3), один выход (потенциал) и базу правил типа «*если*..., *то*». Система формируется при помощи модуля Fuzzy Logic из пакета MatLab.

Модуль Fuzzy Logic позволяет строить нечеткие системы двух типов – Мамдани и Сугэно. Основное отличие между этими системами заключается в разных способах задания значений выходной переменной в правилах, образующих базу знаний. В системах типа Мамдани значения выходной переменной задаются нечеткими термами, в системах типа Сугэно – как линейная комбинация входных переменных. В этой задаче используем алгоритм вывода Мамдани (рис. 2.41).

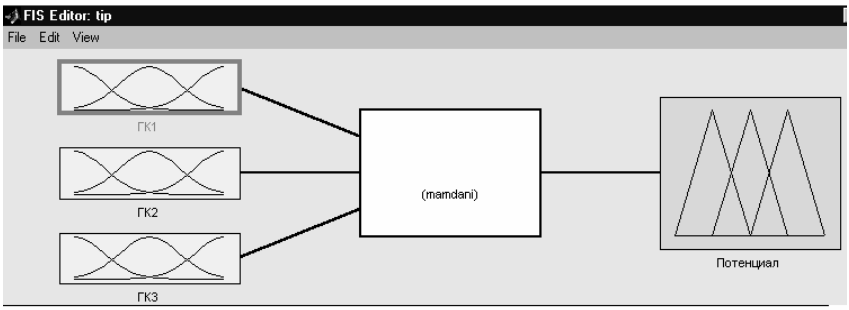


Рис. 2.41

Зададим ФП для переменных. Для этого устанавливаем диапазон изменения и отображения для переменных ГК1, ГК2, ГК3 от –3 до +3, так как именно в этом диапазоне находятся значения переменных. Для каждого ГК задаются три ФП гауссова типа, каждая из которых характеризует ГК, соответственно, как «большой», «средний» и «малый».

Для выходной переменной (потенциал порта) диапазон изменения зададим в пределах от 0 до 100% с ФП треугольной формы и присвоим им имена: «низкий», «средний», «большой». Будем считать, что низкий потенциал соответствует диапазону от 0 до 20 %,

средний потенциал – от 20 до 40 % , свыше 40 % потенциал считается большим (рис. 2.42).

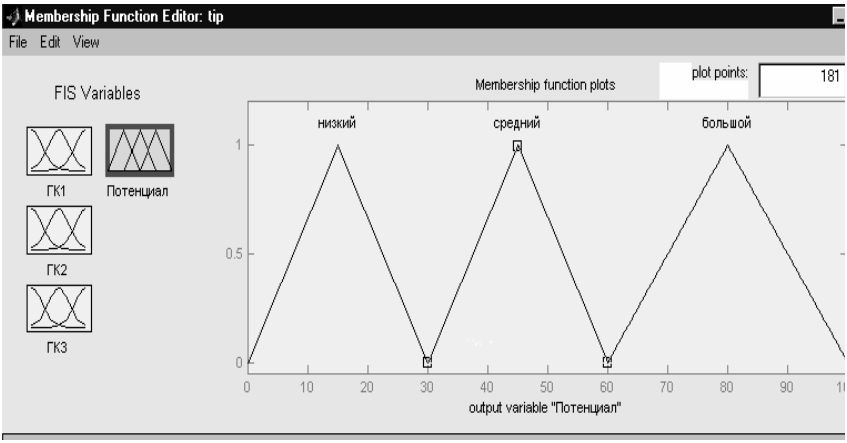


Рис. 2.42

Следующим шагом в формировании задачи является составление правил типа «*если*..., *то*». В соответствии с тем, что наиболее значимыми для грузооборотов портов Балтики являются ГК1 и ГК2, а влияние на общий грузооборот ГК3 носит по сравнению с ними второстепенный характер, создадим следующую базу правил:

Если ГК1 большой и ГК2 большой и ГК3 большой, *то* потенциал большой.

Если ГК1 большой и ГК2 средний и ГК3 средний, *то* потенциал большой.

Если ГК1 большой и ГК2 средний и ГК3 малый, *то* потенциал средний.

Если ГК1 средний и ГК2 большой и ГК3 средний, *то* потенциал большой.

Если ГК1 средний и ГК2 большой и ГК3 малый, *то* потенциал средний.

Если ГК1 малый и ГК2 малый и ГК3 малый, *то* потенциал низкий.

Если ГК1 малый и ГК2 малый и ГК3 большой, *то* потенциал средний.

Если ГК1 малый и ГК2 малый и ГК3 средний, *то* потенциал низкий.

Если ГК1 средний и ГК2средний и ГК3 средний, *то* потенциал средний.

Если ГК1 средний и ГК2 малый и ГК3 малый, *то* потенциал низкий.

Если ГК1 малый и ГК2 средний и ГК3 малый, *то* потенциал низкий.

Результаты проведенных расчетов для порта Санкт-Петербург приведены на рис. 2.43.

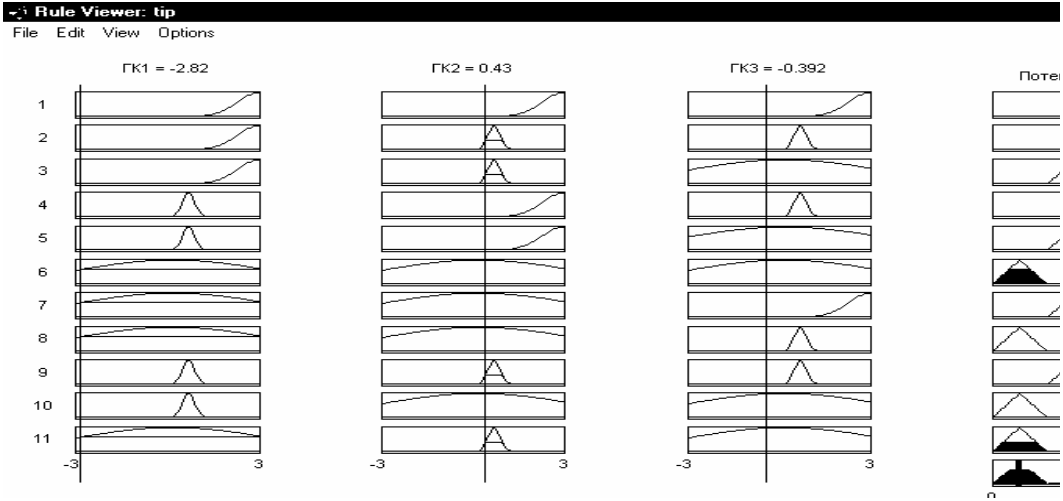

Rule Viewer: tip
File Edit View Options
ГК1 = -2.82
ГК2 = 0.43
ГК3 = -0.392
1
2
3
4
5
6
7
8
9
10
11
-3
3
-3
3
-3
3

Рис. 2.43

В итоге получили значения потенциалов для всех шести рассматриваемых портов (табл. 2.7).

Таблица 2.7

Номер	Порты	ГК1 (нефтяной)	ГК2 (химический)	ГК3 (RORO)	Потенциал, %
1	Санкт-Петербург	–2,82	0,43	–0,392	15
2	Рига	2,04	0,227	–0,933	18,2
3	Вентспилс	0,509	1,96	0,252	32,4
4	Лиепая	–0,0857	–1,71	–1,29	15
5	Клайпеда	0,292	–1,5	1,78	23,5
6	Таллинн	0,0663	0,582	0,586	15,8

Из табл. 2.7 видно, что все порты имеют недостаточно высокие потенциалы, следовательно, не используют свои мощности рациональным образом. Самый крупный нефтяной порт – Вентспилс – имеет самое высокое значение потенциала (32,4%) и попадает лишь в интервал среднего потенциала. Все остальные порты относятся к диапазону низкого потенциала. Тем не менее порты Клайпеда (23,5%) и Рига (18,2%) несколько выделяются в этой группе. Порты Санкт-Петербург, Лиепая и Таллинн имеют весьма низкий результат.

Таким образом, здесь показана возможность использования аппарата НЛ для решения нетривиальной задачи оценки потенциала различных портов.

Приведем еще один пример использования НЛ в задаче оценки кредитоспособности потенциального клиента, обратившегося в финансовое учреждение с просьбой о выдаче кредита. Такая задача решалась автором при консультировании одного из городских банков. Здесь для сокращения объема вычислений рассмотрим упрощенный вариант такой оценки, хотя никаких принципиальных ограничений для перехода к более сложному варианту расчета не имеется. Пусть в банк обратился клиент (физическое лицо) с просьбой о кредите. Соответствующее структурное подразделение банка (кредитный отдел) рассматривает заявление и принимает положительное или отрицательное решение о выдаче кредита. Поскольку в России еще отсутствуют кредитные бюро, в которых должна накапливаться финансовая предыстория потенциальных клиентов, то решение принимается на основании некоторых критериев, характеризующих финансовое состояние будущего заемщика.

Пусть в качестве таких критериев выбраны следующие параметры: возраст, доход, наличие квартиры, дачи, автомобиля. Подчеркнем еще раз, что выбор именно этих входных переменных обусловлен только логикой и здравым смыслом: при необходимости набор таких признаков клиента может быть расширен. Выходной переменной системы нечеткого вывода служит кредитоспособность клиента, выраженная в баллах от 0 до 10.

В табл. 2.8 приведены входные и выходная переменные с указанием диапазонов изменения.

Таблица 2.8

Входные переменные	Выходная переменная
1.Возраст_Age (20–60) лет 2. Доход_Return (200–2000) долл. 3. Квартира_Flat (0–10) баллов 4. Дача_Cottage (0–10)баллов 5. Машина_Auto (0–10) баллов	Кредитоспособность_Credit (0–10 баллов)

Отметим, что первые две входные переменные имеют размерность, соответственно, в годах и денежных единицах, а последние три – выражены в баллах, хотя могут быть переведены в денежные единицы. Принципиального значения выбор единиц не играет роли.

Общий вид системы нечеткого вывода приведен на рис. 2.44.

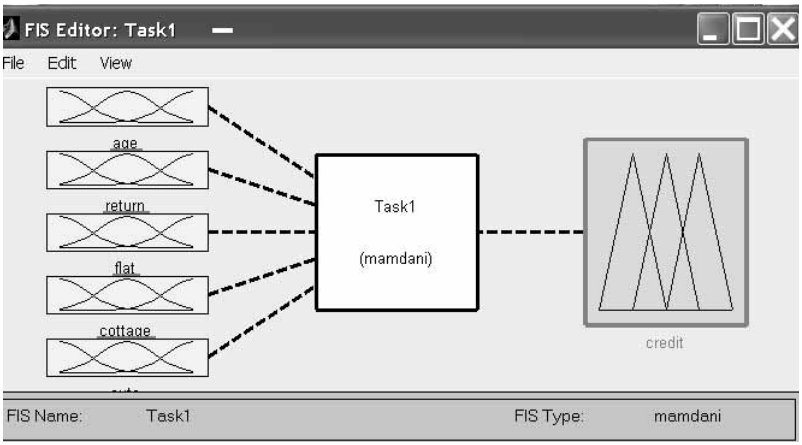


Рис. 2.44

Функции принадлежности в данном примере выбираются разными для отдельных переменных. На рис. 2.45 показана ФП для переменной «доход» (`return`).

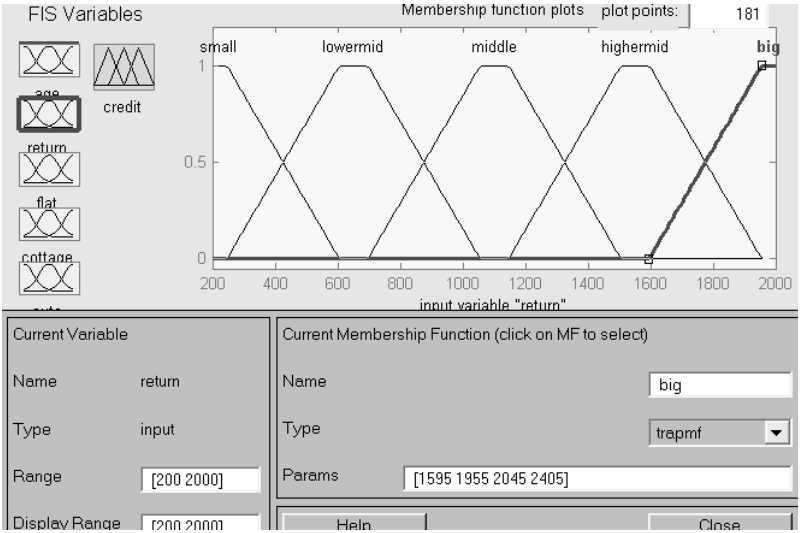


Рис. 2.45

База правил здесь составляется на основе логических умозаключений, которые приведены в диалоговом окне на рис. 2.46 (всего 9 правил).

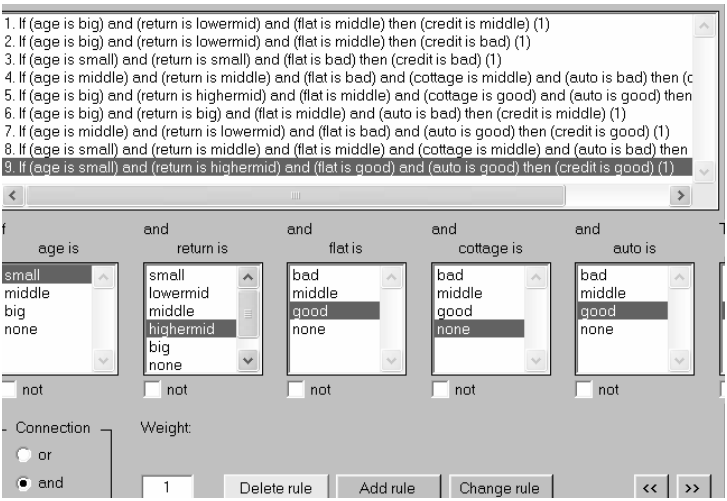


Рис. 2.46

Пример моделирования системы -1. `Credit` = 7,29 балла приведен на рис. 2.47, моделирование системы -2. `Credit` = 2,1 балла – на рис. 2.48.

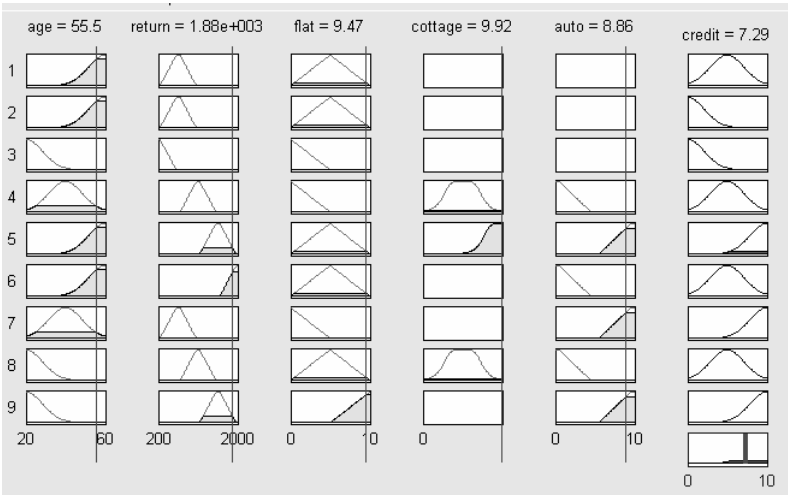


Рис. 2.47

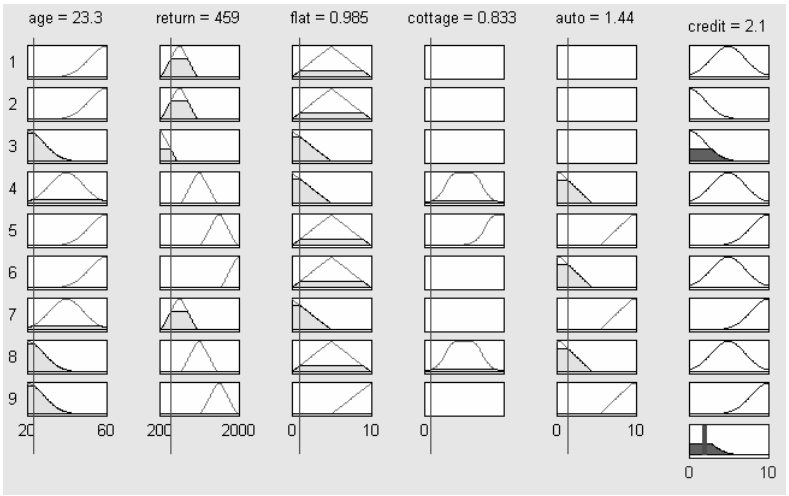


Рис. 2.48

Как видно из последних двух рисунков, система дает различные оценки кредитоспособности для двух обратившихся клиентов, от-

личающихся по всему набору входных признаков, что свидетельствует о работоспособности предложенного метода. Такая объективная оценка клиентов должна предварять окончательное принятие решения о выдаче займа кредитным отделом банка или лицом, принимающим решения.

В целом, подводя итоги рассмотрению методов НЛ и их использованию в задачах менеджмента, можно сказать, что на сегодняшний день возможности предлагаемого аппарата еще не оценены должным образом специалистами-менеджерами. В условиях становления рыночной экономики с характерными в такой ситуации неопределенностями НЛ, по своей сути, имеющая дело с «размытыми» по некоторому интервалу величинами, как нельзя лучше подходит для описания таких явлений.

Библиографический список

1. *Kosko B.* Neural Networks and Fuzzy Systems. New York: Prentice-Hall, 1992.

2. Обработка нечеткой информации в системах принятия решений/ *А. Н. Борисов, А. В. Алексеев, Г. В. Меркурьева* и *др.* М.: Радио и связь, 1989.

3. *Змитрович А. И.* Интеллектуальные информационные системы. Минск: ТетраСистемс, 1997.

4. *Тэрано Т., Асаи К., Сугэно М.* Прикладные нечеткие системы. М.: Мир, 1993.

5. *Андрейчиков А. В., Андрейчикова О. Н.* Анализ, синтез, планирование решений в экономике. М.: Финансы и статистика, 2000.

6. *Петерс Э.* Хаос и порядок на рынках капитала. М.: Мир, 2000.

7. *Круглов В. В., Дли М. И., Голунов Р. Ю.* Нечеткая логика и искусственные нейронные сети. М.: Физматлит, 2001.

8. *Nauck D., Klawonn F., Kruse R.* Fuzzy Sets, Fuzzy Controllers and Neural Networks// Scien. Journ. of Humboldt-Univer. Berlin: Series Medicine, 1992. № 41. Vol. 4. P. 99–120.

9. *Tsoukalas L. Y., Ikonomopoulos A., Uhrig R. E.* Fuzzy Neural Control// Artificial Neural Networks for Intelligent Manufacturing/ Ed. C.H. Dadli, London: Chapman and Hall, 1994.

10. *Дюк В., Самойленко А.* Data Mining. СПб.: Питер, 2001.

11. *Benachenhou D.* Smart Trading with FRET// Trading on the Edge/ Ed. G. J. Deboek. New York: J. Wiley and Sons, 1994.

12. *Wang G. Y., Fisher P. S.* Knowledge Acquisition: Neural Network Learning// Proceeding of SPIE, 2000. Vol. 4057. P. 117–128.

13. *Jang J. S.* ANFIS: Adaptive-Network-Based Fuzzy Inference System// IEEE Trans. Syst., Man, Cybern. 1993. Vol. 23. P. 665–685.

14. Базы данных. Интеллектуальная обработка информации/ *В. В. Корнеев, А. Ф. Гареев, С. В. Васютин* и *др.* М.: Нолидж, 2000.

ГЛАВА 3. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Генетические алгоритмы применяются для решения оптимизационных задач с помощью метода эволюции, т. е. путем отбора из множества решений наиболее подходящего. В типичной задаче оптимизации существует набор переменных, влияющих на процесс, и формула или алгоритм, которые используют эти переменные для построения модели этого процесса. При этом задача заключается в том, чтобы найти такие значения переменных, которые определенным образом оптимизируют модель. В случае, если моделью является формула, то обычно отыскивают максимум или минимум функции, которую данная формула представляет. Существует много математических методов, которые решают задачи оптимизации в том случае, если это задачи с «хорошим поведением». Однако традиционные методы терпят крах, если задача не принадлежит к этому классу. Имеется большой класс задач, для которых вычислительные сложности растут экспоненциально с размерностью задачи (NP-задачи, «nondeterministic polynomial») Примерами задач с таким «плохим поведением» могут служить комбинаторные задачи, а также задачи, математическое описание которых не является гладкой непрерывной функцией. Здесь минимизируется целевая функция (функция стоимости), которая зависит от порядка конечного числа объектов. Количество вариантов размещения N объектов, и, следовательно, усилий для нахождения минимума целевой функции увеличиваются экспоненциально с ростом N.

В данном разделе рассматриваются ГА и способы решения оптимизационных задач из области менеджмента посредством ГА.

3.1. Сущность эволюционных вычислений

В последние 30 лет появился интерес к задачам, решение которых основано на принципах эволюции и наследования признаков. Системы подобного рода содержат популяцию потенциальных решений, имеют определенный процесс отбора, использующий критерии пригодности индивидуумов (отдельных решений), применяют некоторые операторы рекомбинации. К таким системам относится класс эволюционных вычислений (ЭВ).

Под последними понимается термин, используемый для описания алгоритмов поиска, оптимизации или обучения, основанных на некоторых формальных признаках естественного эволюционного отбора [1]. Методы ЭВ часто применяются для описания процессов эво-

люции программ или функций (генетическое программирование), конечных автоматов (эволюционное программирование) и систем, основанных на продукционных правилах (классификационные системы). Иногда ЭВ вместе с НЛ используются для обучения нейронных сетей, что привело к новому термину «мягкие вычисления», объединившие ГА, НЛ и ИНС.

Среди методов ЭВ можно выделить следующие [2–4]:

эволюционное программирование (1963 г., Л. Фогель, А. Оуэнс, М.Уолш, (L. Fogel, A. Owens, M. Walsh)) – представляет решение задачи в виде универсальных конечных автоматов, которые реагируют на стимулы из внешней среды;

эволюционные стратегии (1973 г., И. Реченберг (I. Rechenberg)) – каждое решение находится в виде массива числовых параметров, определяющих аргумент целевой функции;

генетические алгоритмы (1975 г., Д. Холланд (J. Holland)) – каждое решение является битовой строкой (хромосомой) определенной длины в популяции фиксированного размера;

генетическое программирование (1992 г., J. Koza) – здесь применяются идеи ГА для эволюции компьютерных программ.

В России до начала 80-х гг. XX в. получили развитие два направления, близкие к методам ЭВ, но мало известные на Западе, к которым относятся методы стохастической оптимизации (1968 г., Л. А. Растригин) и группового учета аргументов (1969 г., А. Г. Ивахненко).

Каждая из этих школ взяла за основу ряд принципов, существующих в природе, и упростила их до такой степени, чтобы их можно было реализовать на вычислительной технике того времени.

В самом общем виде метод ЭВ можно описать следующим образом. Метод ЭВ представляет собой вероятностный алгоритм, который содержит популяцию индивидуумов $P(t) = \{x_1^t, x_2^t, \ldots, x_n^t\}$ на итерации t. Каждый индивидуум есть потенциальное решение рассматриваемой проблемы, и в любом из методов ЭВ реализуется как некоторая (возможно, сложная) структура данных S. Каждое решение $x^t{}_i$ оценивается для получения величины пригодности (в английском языке часто используется термин «fitness»). Затем создается новая популяция (итерация $t+1$) путем отбора наиболее пригодных индивидуумов (этап селекции). Некоторые члены новой популяции подвергаются преобразованиям (этап рекомбинации) посредством генетических операторов для формирования новых решений. Среди операторов можно выделить оператор мутации, который создает новый индивидуум путем малых изменений исходного, и оператор скрещивания, фор-

мирующий новые индивидуумы посредством комбинирования частей нескольких исходных. После ряда таких генераций программа сходится, т. е. находится лучший индивидуум, который представляет собой оптимальное решение.

Очевидно, что многие эволюционные программы (синоним эволюционных вычислений) могут быть сформулированы для решения конкретной проблемы. Такие программы могут иметь различную структуру данных для построения отдельного индивидуума, генетические операторы для трансформации индивидуумов, методы для создания начальной популяции, параметры вычислений (размер популяции, вероятности применения различных операторов и т. д.). Однако, несмотря на различия, такие программы имеют общий принцип: популяция индивидуумов подвергается некоторым преобразованиям, и в течение эволюционного процесса индивидуумы сражаются за выживание.

Следует отметить, что методы ЭВ не гарантируют обнаружения глобального оптимума за приемлемое время. Практический интерес к ним объясняется тем, что эти методы позволяют найти «хорошие» решения очень трудных задач за меньшее время, чем другими методами.

Необходимо уяснить разницу между эволюционным программированием и ГА: при первом подходе используется любая структура данных (хромосомное представление), подходящая для решения задачи, и любое множество генетических операторов; во второй ситуации применяются бинарные строки фиксированной длины (хромосомы) и два оператора (бинарные мутация и скрещивание).

Естественно, что близкое к эволюционным процессам представление потенциального решения для данной проблемы вместе с совокупностью подходящих генетических операторов может быть полезным для решения многих задач, и такой подход является весьма перспективным направлением. За последние 10–15 лет опубликовано достаточное количество работ, в которых для решения конкретных задач рассмотрены различные модификации классического представления ГА (ниже будет показано, что ГА, впервые предложенный Д. Холландом в 1975 г., является классическим), например, строки переменной длины; более богатая, чем бинарные строки, структура; модифицированные генетические операторы. Среди последних можно указать описание ГА, который использует в качестве оператора метод обратного распространения ошибки (в гибридных нейронных сетях). Многие исследователи в этой области применяют модифицированные реализации ГА или путем использования нестроковой хромосомы при представлении задачи, или посредством разра-

ботки специальных генетических операторов для возможности решения конкретной проблемы. Различные нестандартные подходы были предложены для частных задач, так как классические ГА было трудно непосредственно применить для решения подобных задач.

В связи с вышеуказанными соображениями о модификации ГА возникает вопрос: являются ли эволюционные стратегии генетическими алгоритмами? Для того чтобы избежать подобных вопросов, связанных, прежде всего, с классификаци ЭВ, назовем такие системы эволюционными программами (ЭП) [2] и рассмотрим различие между ГА и ЭП. Несмотря на то что ГА имеют достаточно серьезную теоретическую базу, они не могут обеспечить успешное применение во многих областях. Очевидно, что главный фактор неудачи ГА является тем же фактором успеха в решении задач – это область независимости. Одним из следствий четкости ГА в смысле области независимости является их неспособность к учету нетривиальных ограничений. В большинстве работ по ГА хромосомы представляют собой битовые строки (состоящие из 1 и 0), поэтому реализация ограничений в такой ситуации является достаточно сложной задачей.

В эволюционных программах проблема ограничений имеет другой характер. Здесь речь идет, скорее, о выборе «лучшего» хромосомного представления искомого решения вместе со значимыми генетическими операторами для удовлетворения всех ограничений, накладываемых данной проблемой. Любой генетический оператор должен проходить через некоторую характеристическую структуру от «родителей» к «потомкам», поэтому структура представления играет важную роль в определении генетических операторов. Кроме того, различные структуры представлений имеют различные характеристики пригодности для ограничений, что еще более усложняет задачу. Эти два компонента – представление и операторы – влияют друг на друга. Очевидно, что любая проблема требует тщательного анализа, который позволит выбрать подходящую схему представления и генетические операторы.

Основная концептуальная разница между ГА и ЭП показана на рис. 3.1 и 3.2. ГА, которые действуют на бинарных строках, требуют модификации исследуемой проблемы в подходящую для таких алгоритмов форму. Указанная модификация обычно включает в себя отображение между потенциальными решениями и бинарным представлением, изменение алгоритмов и т. п.

С другой стороны, ЭП оставляют проблему неизменной, модифицируя хромосомное представление потенциального решения и применяя подходящие генетические операторы.

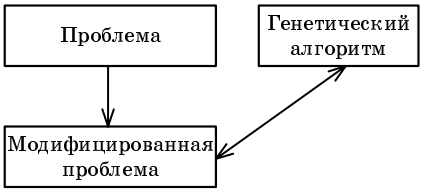


Рис. 3.1

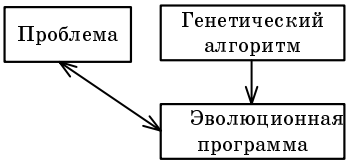


Рис. 3.2

Подводя итог сопоставлению ГА и ЭП, можно сказать, что для решения нетривиальной проблемы с применением ЭП необходимо или трансформировать задачу в форму, подходящую для ГА (рис. 3.1), или изменить ГА для решения данной проблемы (рис. 3.2).

3.2. Основные понятия генетических алгоритмов

Сосредоточим внимание на генетических алгоритмах как наиболее разработанном методе ЭВ. Кроме того, в настоящее время трудно провести четкие границы между различными модификациями эволюционных методов, и класс ГА объединяет многие из них. В англоязычной научной литературе эти методы называют «genetic algorithms» или «a genetic algorithms», но не «the genetic algorithms», так как ГА представляет широкий класс связанных процедур со многими отдельными шагами.

Эволюция и компьютеризация пришли вместе в мир ГА. Ч. Дарвин разработал свою теорию эволюции в 1830-х гг., а Ч. Беббедж (C. Babbage) – один из основателей современного компьютинга и друг Дарвина – предложил аналитическую вычислительную машину примерно в то же время. Вероятно, они были бы удивлены и восхищены их связью в этих двух областях. Идея использования ГА состоит в том, чтобы выполнять то, что делает природа. Рассмотрим один пример с популяцией кроликов, имеющейся в некоторый момент времени t. Часть из них более быстрая и проворная, чем другая. Такие быстрые кролики имеют меньше шансов быть съеденными лисами, и поэтому большинство из них выживает, чтобы делать то, что кролики хорошо делают: создавать еще больше кроликов. Выжившая популяция кроликов дает новое потомство, которое имеет хорошую смесь генетического материала: некоторые «медленные» кролики скрещиваются с «быстрыми», быстрые – с быстрыми, проворные – с неловкими и т. д. Результирующее потомство будет в среднем более быстрым и проворным, чем в исходной популяции, потому что только такие особи имеют шанс избежать гибели.

Генетические алгоритмы выполняют шаг за шагом процедуру, которая тесно связана с примером из жизни кроликов. Следовательно,

одним из возможных определений может служить следующее: ГА – это алгоритмы, чьи методы поиска моделируют частные естественные явления: генетическое наследование и борьбу за выживание. прежде чем перейти к структуре ГА, осветим кратко историю генетики.

Генетика – это изучение наследственности. Генетики изучают взаимодействие генов и их передачу от родителей к потомству. В живых организмах клетки содержат важные группы специальной материи, содержащей наследственную информацию. Такие нитевидные структуры называются хромосомами. Гены – специальные элементы, которые переносятся хромосомами. Основной процесс кодирования информации в пределах генов и хромосом достаточно прост, но возможные результаты почти безграничны по количеству. Генетическая структура каждого организма называется генотипом, который определяет и ограничивает многие аспекты развития и отбора. Отклик организма на внешние условия принципиально определяется его г е н о т и п о м . Свойства организма, которые возникают из этого столкновения с окружающей средой, определяют его ф е н о т и п .

Основной принцип естественного отбора как главного принципа эволюции был сформулирован Ч. Дарвином задолго до открытия генетических механизмов. Игнорируя основные наследственные принципы, Ч. Дарвин выдвинул гипотезу слияния, полагая, что родительские качества смешиваются вместе, подобно жидкостям, в наследственном организме. Его селекционная теория вызвала серьезную критику со стороны других ученых, однако в то время никакой более приемлемой генетической теории еще не было.

Значительный шаг вперед в период 1856–1863-х гг. в разработке генетики сделал католический священник из Чехии Г. Мендель (G. Mendel), который открыл основные принципы передачи наследственных факторов от родителей к потомству. В результате многолетних исследований он открыл законы доминирования признаков в первом поколении независимо от их распределения в последующих поколениях и их количественного соотношения. Своими исследованиями Г. Мендель опередил развитие науки на десятилетия. Его законы стали известны научному сообществу только после того, как они были независимо переоткрыты в 1900 г., когда Х. де Фриз (Y. de Vries, Нидерланды), К. Корренс (C. Correns, Германия) и К. Чермак (C. Tshermak, Австрия) опубликовали результаты исследований по скрещиванию растений.

Генетика была полностью разработана американским ученым Т. Морганом (T. Morgan) в период 1909–1914-х гг. и его коллегами, которые экспериментально доказали, что хромосомы являются

главными носителями наследственной информации, а гены, которые представляют наследственные факторы, расположены в линию на хромосомах. Т. Моргану удалось наблюдать в микроскоп процесс обмена генетического материала между разными хромосомами: две хромосомы сближались и скрещивались, обмениваясь фрагментами. Ученый представлял гены упорядоченными по длине хромосом, как бусинки в ожерелье. Экспериментальные данные привели его к замечательной идее о создании генетических карт. Очевидно, что чем дальше находятся два гена друг от друга, тем больше вероятность обрыва их связывающей нити и получения новых сочетаний генов. Открытие Т. Моргана дало мощный толчок развитию генетики; молодая наука обогатилась первыми теоретическими обоснованиями и получила признание в мире ученых, а сам Т. Морган в 1933 г. получил Нобелевскую премию за создание хромосомной теории наследственности.

В начале 1960-х гг. некоторые биологи начали экспериментировать с компьютерной имитацией генетических систем. Однако современная теория ГА, применимая для решения, в основном, оптимизационных задач, связана с именем Д. Холланда. Он начал читать курсы по теории адаптивных систем в Мичиганском университете и опубликовал к концу 60 – началу 70-х гг. много работ по этой тематике. В 1975 г. он издал книгу [5], где обобщил накопленный материал. Д. Холланд по праву считается отцом-основателем теории ГА, а его книга – библией в этой области знаний.

Генетические алгоритмы используют словарь, заимствованный из естественной генетики. Здесь необходимо отметить все еще не устоявшуюся терминологию в этой области, и поэтому в различных изданиях (особенно в сетевых источниках) можно встретить различную интерпретацию одинаковых понятий. В данном пособии будем, в основном, придерживаться следующей терминологии:

битовая строка 0101…101 (хромосома, особь, индивидуум) – определяет точку пространства поиска и представляет потенциальное решение задачи;

гены – элементы, из которых состоит хромосома (синоним генов – признаки, буквы);

популяция – набор хромосом (строк); в классе ГА размер (величина) популяции принимается фиксированной величиной;

поколение (генерация) – новое поколение после каждого шага работы ГА;

родители – хромосомы, из которых путем скрещивания и отбора формируются хромосомы-«потомки»;

качество хромосомы – функция, оценивающая пригодность данной строки по сравнению с другими (в англ. языке – fitness).

генетические операторы (отбор, скрещивание, мутация) – преобразования, которым подвергаются хромосомы в процессе эволюции и борьбы за выживание.

Подчеркнем еще раз, что каждая строка (хромосома) представляет потенциальное решение задачи. Эволюционный процесс через популяцию хромосом соответствует поиску по пространству потенциальных решений. Популяция совершает имитационную эволюцию: при каждой генерации относительно «хорошие» решения воспроизводятся, в то время как «плохие» – умирают. Для различия между плохими и хорошими решениями используется оценочная функция, определяющая качество каждой строки.

Для того чтобы выявить отличие ГА при поиске оптимальных решений от традиционных способов решения оптимизационных задач, кратко укажем принцип действия двух общепринятых методов: подъема на холм и имитационный отжиг.

Метод подъема на холм (hillclimbing method) использует итеративный улучшающий подход. Метод применим к единственной (текущей) точке в пространстве поисков. Во время одной итерации следующая точка выбирается из условия соседства с текущей точкой. В случае, если новая точка обеспечивает лучшую величину целевой функции, то она становится текущей. В противном случае выбирается другая соседняя точка, которая сравнивается с текущей. Метод поиска оптимума завершается, если дальнейшего улучшения в поведении целевой функции не наступает. Ясно, что этот метод обеспечивает только локальный оптимум, а продолжительность поиска зависит от выбора начальной точки. Кроме того, здесь отсутствует информация об относительной ошибке (по отношению к глобальному оптимуму) найденного решения. Для увеличения шансов на успех метод обычно выполняется для значительного числа начальных точек.

Имитационный отжиг (simulated annealing) основан на идее, заимствованной из статистической физики. Этот метод описывает поведение материального тела при отвердевании с применением процедуры отжига (управляемого охлаждения) при температуре, последовательно уменьшаемой до нуля. В реальных процессах кристаллизации твердых тел температура понижается ступенчатым образом. На каждом уровне она какое-то время поддерживается постоянной, что необходимо для обеспечения термического равновесия. На протяжении всего периода времени, когда температура остается выше абсо-

лютного нуля, она может как повышаться, так и понижаться. За счет удержания температуры процесса вблизи от значения, соответствующего уровню термического равновесия, удается обходить ловушки локальных максимумов. Метод имитации отжига представляет собой алгоритмический аналог физического процесса управляемого охлаждения. Предложенный Н. Метрополисом в 1953 г. и доработанный многочисленными последователями, он в настоящее время считается одним из немногих алгоритмов, позволяющих практически находить глобальный минимум функции нескольких переменных.

В описании алгоритма в качестве названия параметра, влияющего на вероятность увеличения значения целевой функции, используется термин «температура», хотя с формальной точки зрения приведенная модель оптимизации является только математической аналогией процесса отжига. Алгоритм имитации отжига выглядит концептуально несложным и логически обоснованным. В действительности приходится решать много фундаментальных проблем, которые влияют на его практическую применимость. Основной следует назвать проблему длительности имитации. Для повышения вероятности достижения глобального минимума длительность отжига (представляемая количеством циклов, повторяемых при одном и том же значении температуры) должна быть достаточно большой, а коэффициент уменьшения температуры – низким. Это увеличивает продолжительность процесса моделирования, что может дискредитировать его с позиций практической целесообразности.

Метод имитационного отжига исключает большинство недостатков метода подъема на холм: решение не зависит от начальной точки и обычно является близким к оптимальной точке. Это достигается введением вероятности p замены текущей точки новой точкой: $p = 1$, если новая точка обеспечивает лучшее значение целевой функции; однако $p > 0$ в противоположной ситуации. В последнем случае вероятность p зависит от значений целевой функции для текущей и новой точек и управляющего параметра температуры. Во время работы метода температура понижается ступенями, и алгоритм завершается при некотором малом значении температуры.

Отличия ГА от традиционных методов поиска состоят в следующем:

– для поиска оптимума используют несколько точек одновременно, а не переходят от точки к точке, как это делается в традиционных методах, что позволяет избежать опасности попадания в локальный экстремум целевой функции, если она не является унимодальной, т. е. имеет несколько таких экстремумов. Использование нескольких точек одновременно значительно снижает такую возможность;

– в процессе работы ГА не требуют никакой дополнительной информации, что увеличивает скорость работы алгоритма (единственная информация: область допустимых значений и функция пригодности в определенных точках);

– используют и детерминированные правила для перехода от одних точек к другим, и вероятностные правила для порождения новых точек анализа;

– работают с кодами, в которых представлен набор параметров, зависящих от аргументов целевой функции. Интерпретация этих кодов происходит только перед началом работы алгоритма и после завершения его работы для получения результата. В процессе работы манипуляции с кодами происходят совершенно независимо от их интерпретации, код рассматривается просто как битовая строка.

Структура ГА такая же, как и любой эволюционной программы. Во время итерации t ГА содержит популяцию потенциальных решений (хромосом, векторов) $P(t) = \{x_1^t, \ldots, x_n^t\}$. Каждое решение x_i^t оценивается для того, чтобы получить некоторую меру его пригодности (fitnεss). Затем на итерации $t+1$ формируется новая популяция путем отбора наиболее пригодных индивидуумов. Некоторые члены этой новой популяции подвергаются репродукции посредством генетических операторов: скрещивания и мутации для образования новых решений. Скрещивание комбинирует признаки двух родительских хромосом для образования двух потомков обменом соответствующих сегментов родителей. Например, если родители отображаются пятимерными векторами $(a_1, b_1, c_1, d_1, e_1)$ и $(a_2, b_2, c_2, d_2, e_2)$, то, выбрав точку скрещивания после второго гена, получим такие потомки: $(a_1, b_1, c_2, d_2, e_2)$ и $(a_2, b_2, c_1, d_1, e_1)$. Ясно, что оператор скрещивания представляет собой обмен информации между различными потенциальными решениями. Мутация произвольно изменяет один или больше генов выбранной хромосомы случайной заменой значения, равного единице, на нуль или наоборот. Очевидно, что оператор мутации вводит некоторую вариабельность в популяцию хромосом. ГА для решения любой проблемы должен содержать, как правило, следующие компоненты:

– генетическое представление потенциальных решений задачи;

– способ создания начальной популяции потенциальных решений;

– оценочную функцию, которая играет роль окружения и ранжирует решения по степени их пригодности;

– генетические операторы, изменяющие генетический состав потомства;

– значения параметров ГА (вероятности скрещивания и мутации, размер популяции, количество поколений и др.).

В качестве критериев остановки выполнения алгоритма могут использоваться такие:

– сформировано заданное число поколений;

– популяция достигла заданного качества;

– достигнут определенный уровень сходимости.

Отметим, что последние два критерия связаны с заданной величиной пригодности популяции или сходством строк в популяции.

Процесс работы ГА приведен на рис. 3.3.

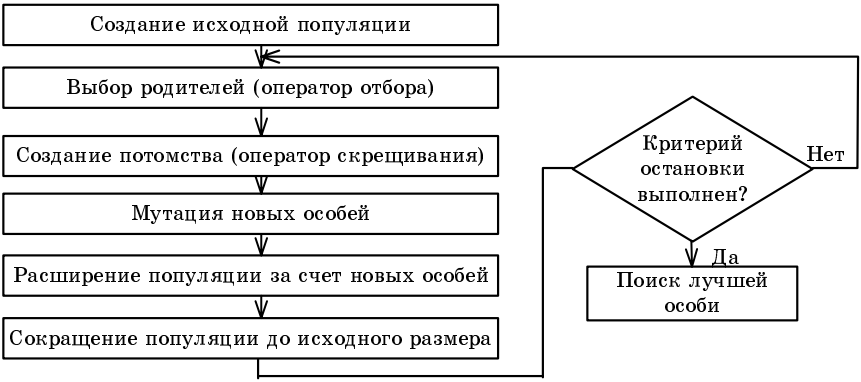


Рис. 3.3

Для иллюстрации основных понятий и принципа действия ГА приведем простой пример на поиск максимума.

Пример 3.1. Найти максимум функции $f(x) = x^2$ в диапазоне $0 < x < 31$.

Здесь в качестве функции пригодности выступает сама функция. Чем больше ее значение, тем лучше пригодность хромосомы.

Установим размер популяции, равный четырем строкам. Начальная популяция и оценка пригодности приведена в табл. 3.1.

Таблица 3.1

№ строки	Начальная популяция	x	$f(x)$	Относительная пригодность, %
1	01101	13	169	14,4
2	11000	24	576	49,2
3	01000	8	64	5,5
4	10011	19	361	30,9
			1170	100

Так как функция пригодности второй строки лучшая, отбираем две копии второй строки и оставляем первую и четвертую строки в родительском пуле. Отбор партнеров производим случайным образом: партнером первой строки служит вторая, партнером четвертой – тоже вторая. Положение точек скрещивания также случайно и выбирается следующим образом: для пары из первой и второй строк точка скрещивания – после четвертого бита; для пары из второй–четвертой строк – после второго бита (табл. 3.2).

Таблица 3.2

№ строки	Родительский пул	Парная строка	До скрещивания	После скрещивания
1	01101	2	0110[1]	01100
2	11000	1	1100[0]	11001
3	11000	4	11[000]	11011
4	10011	2	10[011]	10000

Второе поколение без мутации приведено в табл. 3.3.

Таблица 3.3

Строка	x	$f(x)$	Относительная пригодность, %
01100	12	144	8,2
11001	25	625	35,6
11011	27	729	41,5
10000	16	256	14,7
		1754	100%

Из табл. 3.3 видно, что третья строка является лучшей во втором поколении и значение $x = 27$ достаточно близко к отыскиваемому максимуму. Очевидно, что через несколько шагов оптимальное решение будет найдено даже без использования оператора мутации.

3.3. Кодирование в генетических алгоритмах

Подчеркнем еще раз различие между фенотипом и генотипом. Из биологии известно, что любой организм может быть представлен своим фенотипом, который фактически определяет, чем является объект в реальном мире (во внешней среде). Генотип содержит всю информацию об объекте на уровне хромосомного набора. Для решения задачи необходимо представить каждый признак объекта в форме, пригодной для использования в ГА. Кодирование решения задачи в хромо-

соме является ключевой проблемой применения ГА. Проблема исследована с различных сторон, например, по характеру отображения пространства генотипов в пространство фенотипов, когда отдельные особи декодируются в решения, и по свойствам преобразования, когда хромосомы регулируются с помощью операторов.

В классической работе Д. Холланда кодирование выполняется с использованием бинарных строк. Такое кодирование для задач оптимизации имеет несколько недостатков вследствие существования хеммингова сдвига для пары закодированных значений, имеющих большое хеммингово расстояние, в то время, как эти величины принадлежат к точкам с минимальным расстоянием в фенотипическом пространстве. Например, пара 01111111111 и 10000000000 принадлежит соседним точкам в фенотипическом пространстве (точки с минимальным евклидовым расстоянием), но имеют максимальное хеммингово расстояние в генотипическом пространстве. Для преодоления хеммингова сдвига все биты должны изменяться одновременно, но вероятность такого события очень мала.

Для многих задач, встречающихся в современных проблемах, достаточно трудно представить их решения с помощью только бинарного кодирования. В течение последних десяти лет были разработаны различные методы кодирования для обеспечения эффективного выполнения ГА, которые могут быть разделены на следующие классы [4]:

– бинарное кодирование;

– кодирование действительными числами;

– целочисленное кодирование;

– кодирование общей структуры данных.

Более подробно рассмотрим *бинарное кодирование*, которое используется в классическом подходе Д. Холланда. Пусть вектор допустимого решения $\vec{x} \in D$, где D – область поиска решений. Каждый компонент вектора $\vec{x} = (x_1, x_2, \ldots, x_n)$ можно закодировать с помощью целого неотрицательного числа $\beta_i \in [0, K_i]$, $i = 1, n$; (K_i – число возможных дискретных значений i-й переменной).

Введем бинарный алфавит $B_2 = \{0;1\}$. Для представления целочисленного вектора $\beta = (\beta_1, \ldots, \beta_n)$ в алфавите B_2 необходимо определить максимальное число двоичных символов θ, которое достаточно для отображения в двоичном коде любого значения β_i из области его допустимых значений $[0, K_i]$. Нетрудно видеть, что параметр θ должен удовлетворять неравенству

$$K < 2^{\theta}, \qquad (3.1)$$

где $K = \max K_i$.

$$1 \le i \le n \ .$$

Тогда для β_i ($0 \le \beta_i \le 2^{\theta}$) можно записать

$$\beta_i = \sum_{i=1}^{\theta} \alpha_i \cdot 2^{\theta-1}, \qquad (3.2)$$

где α_i – двоичное число (0 или 1); θ – длина двоичного слова, кодирующего целое число β_i.

Пример 3.2. Представим в виде хромосомной строки число $\beta_i = 19$.

С учетом формул (3.1) и (3.2) имеем: $\theta = 5$, так как $2^{\theta} = 2^5 = 32$

По формуле (3.2) получим:

$$19 = 1 \cdot 2^{5-1} + 0 \cdot 2^{5-2} + 0 \cdot 2^{5-3} + 1 \cdot 2^{5-4} + 1 \cdot 2^{5-5} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

Строка будет иметь следующий вид:

1	0	0	1	1

В качестве гена, т. е. единицы наследственного материала, ответственного за формирование признаков особи, примем бинарную комбинацию $e_{\theta}(\beta_i)$, которая определяет фиксированное значение целочисленного кода β_i управляемой переменной x_i в обычном двоичном коде. В этом случае $e_{\theta}(\beta_i)$ имеет вид:

α_1	α_2	...	α_{θ}
θ			

Одна особь $a_k{}^t$ будет характеризоваться n генами, каждый из которых отвечает за формирование целочисленного кода соответствующей управляемой переменной. Тогда хромосому можно определить в следующем виде $E(X)$:

$\alpha_i{}^1$	...	$\alpha_{\theta}{}^1$	$\alpha_i{}^2$	...	$\alpha_{\theta}{}^2$	...	...	...	$\alpha_i{}^n$	...	$\alpha_{\theta}{}^n$
$e_{\theta}(\beta_1)$			$e_{\theta}(\beta_2)$			...			$e_{\theta}(\beta_n)$		
ген 1			ген 2			...			ген n		
локус 1			локус 2			...			локус n		
Хромосома											

Местоположение определенного гена в хромосоме называется л о к у с о м , а альтернативные формы одного и того же гена, расположенные в одних и тех же локусах, называются а л л е л я м и .

Хромосома, содержащая в своих локусах конкретные значения аллелей, определяет генотип (генетический код) $E(a_k{}^t)$, который со-

держит всю наследственную генетическую информацию об особи a_k^t, получаемую от предков и передаваемую затем потомкам. Конечное множество всех допустимых генотипов образует генофонд.

Таким образом, в реализации ГА хромосома представляет собой битовую строку фиксированной длины. При этом каждому участку строки соответствует ген. Длина гена внутри строки может быть одинаковой или различной. Например, для объекта из пяти признаков, каждый из которых закодирован геном длиной в четыре элемента, общая длина хромосомы составит 5·4 = 20 битов:

При использовании N битов для бинарной строки (хромосомы) преобразование от двоичного кода строки к десятичному значению осуществляется по формуле:

$$x_i = a_i + decimal\,(100\ldots010_2)\cdot\frac{b_i - a_i}{2^N - 1},$$

где a_i, b_i – нижняя и верхняя границы изменения i-го признака; $decimal\,(100\ldots010_2)$ – десятичное значение бинарной строки.

Кодирование действительными числами является лучшим при решении задач оптимизации функций. Так как топологическая структура пространства генотипов для этого вида кодирования идентична структуре в пространстве фенотипов, то легко сформировать эффективные генетические операторы заимствованием полезных приемов у традиционных методов.

Целочисленное кодирование лучше всего подходит для комбинаторных оптимизационных задач.

В соответствии с *кодированием общей структуры данных* методы кодирования могут быть разделены на два вида:

– одномерный;

– многомерный.

В большинстве случаев используется одномерное кодирование, однако многие проблемы требуют решений в многомерном пространстве, поэтому естественно применять метод многомерного кодирования в таких ситуациях.

Генетические алгоритмы выполняются на двух типах пространств: кодирования и решения, другими словами, пространствах генотипа и фенотипа. Генетические операторы работают в пространстве генотипов, а оценка и отбор происходят в фенотипическом пространстве. Естественный отбор – это связь между хромосомами и поведением декодированных решений.

Отображение из генотипического пространства в фенотипическое оказывает значительное влияние на поведение ГА. Одна из главных проблем заключается в том, что некоторые особи соответствуют невозможным (неприемлемым) решениям данной задачи. При использовании ГА необходимо различать две базовые концепции:

– недопустимость;

– незаконность.

Эти термины часто неправильно используются в литературе. *Недопустимость* относится к ситуации, когда решение, декодированное из хромосом, лежит вне области допустимых решений данной задачи. *Незаконность* относится к такому явлению, когда хромосома не описывает решение конкретной задачи (рис. 3.4).

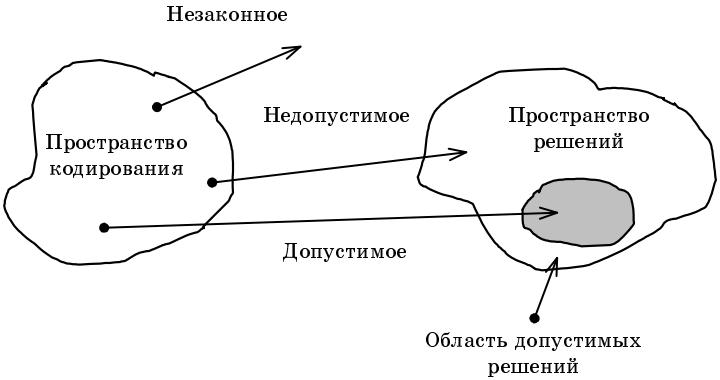


Рис. 3.4

Недопустимость хромосом следует из существа задачи оптимизации с ограничениями. Независимо от того, какой метод используется – традиционный или ГА – он должен иметь дело с ограничениями. Для многих оптимизационных задач допустимая область может быть представлена как система уравнений или неравенств. В таких случаях при работе с недопустимыми хромосомами могут быть применены штрафные методы. В задачах оптимизации оптимум обычно имеет место на границе между допустимыми и недопустимыми областями. Штрафные методы будут усиливать генетический поиск для приближения к оптимуму с двух сторон (допустимой и недопустимой).

Незаконность хромосом вытекает из природы методов кодирования. Для многих задач комбинаторной оптимизации методы кодирования обычно дают незаконное потомство посредством простой одноточечной операции скрещивания. Так как незаконная хромосома не может быть декодирована в решение, штрафные методы неприменимы в такой ситуации.

При появлении или разработке нового метода кодирования необходимо рассмотреть, можно ли провести эффективный генетический поиск этим методом. Для оценки кодирования предложено использовать следующие принципы:

– недостаточность;

– легальность;

– полноту;

– причинность.

Недостаточность означает, что отображение между кодированием и решением должно иметь вид $1–k–1$. В общем случае такое отображение может относиться к одному из следующих типов (рис. 3.5):

1. $1 – k – 1$.

2. $n – k – 1$.

3. $1 – k – n$.

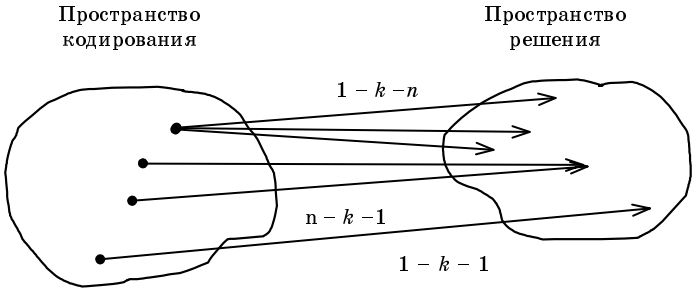


Рис. 3.5

Самый лучший способ – это отображение $1–k–1$. Если имеет место отображение $n–k–1$, то ГА затратит много времени на поиск. Самая плохая ситуация при отображении $1–k–n$, так как необходимы дополнительные процедуры на фенотипическом пространстве для определения единственного из n возможных решений.

Легальность означает, что любая перестановка кодирования соответствует решению. Это свойство гарантирует, что большинство генетических операторов могут быть легко применены к кодированию.

Полнота определяет, что любое решение имеет соответствующее кодирование и любая точка в пространстве решений доступна для генетического поиска.

Причинность связывает малые вариации в пространстве генотипов, которые появляются вследствие мутации, с малыми вариациями в фенотипическом пространстве. Это свойство было предложено,

в первую очередь, для эволюционных стратегий. Оно определяет сохранение соседних структур. Таким образом, для успешного введения новой информации посредством мутации ее оператор должен сохранять структуру соседства в соответствующем фенотипическом пространстве.

3.4. Генетические операторы

Определены три основных вида генетических операторов:

– селекция;

– скрещивание;

– мутация.

Перед тем как непосредственно перейти к изучению этих операторов, остановимся подробнее на концепции поиска и его применении в ГА.

Поиск – один из универсальных методов решения задач, в которых нельзя заранее определить последовательность шагов, приводящих к искомому результату. Обычно рассматриваются два вида поиска:

– случайный;

– локальный.

Случайный поиск исследует решение целиком и способен избежать попадания в локальный оптимум. Локальный поиск использует лучшее решение и способен подняться вверх к локальному оптимуму.

Идеальный поиск должен сочетать оба типа поиска одновременно. Но практически такой поиск невозможно осуществить традиционными методами. ГА являются классом общецелевых поисковых методов, комбинирующих элементы направленного и стохастического типов поиска, что приводит к хорошему балансу между исследованием и использованием поискового пространства.

В традиционных ГА оператор скрещивания применяется как основной, и поведение генетической системы в значительной степени зависит от него. Оператор мутации, который осуществляет спонтанные случайные изменения в различных хромосомах, отчасти выполняет роль фона. По существу, генетические операторы проводят случайный поиск и не могут гарантировать появления улучшенного потомства. Было выполнено много эмпирических исследований сравнения мутации и скрещивания, в результате чего выяснилось, что иногда мутация играет более важную роль, чем скрещивание.

Есть две гипотезы объяснения того, как ГА используют распределенную информацию для формирования хороших решений:

– гипотеза строительных блоков;

– гипотеза изменения управляемой сходимости.

Согласно первой гипотезе скрещивание комбинирует признаки от двух родителей для производства потомства. Иногда скрещивание использует лучшие признаки от обоих родителей, формируя очень хорошее потомство. Так как пригодность особи часто зависит от сложного взаимодействия простых признаков, то важно, чтобы оператор был в состоянии передать потомству те наборы признаков, которые вносили вклад в оценку пригодности родителей.

Вторая гипотеза предполагает использование сходимости популяции для ограничения поиска. При сходимости популяции вариация пригодности становится более заметной. В то время как первая гипотеза усиливает комбинацию признаков, которые выжили в родительской популяции, вторая гипотеза усиливает случайную выборку из распределения, которое является функцией текущей популяции.

Селекция

Основной принцип генетических алгороитмов, по существу, представляет собой дарвиновский естественный отбор. Селекция обеспечивает движущую силу в ГА. Чем больше эта сила, тем быстрее может завершиться генетический поиск; при меньшей величине этой силы эволюционный процесс будет медленнее, чем необходимо.

Селекция направляет генетический поиск в «обещающие» районы поискового пространства. Основной компонент ГА – это метод, используемый для перехода от одной генерации к следующей. Существует много возможных вариаций относительно выбора потенциальных родителей и способа их комбинирования для воспроизводства потомства. Селекция может меняться от очень трудоемких подходов, требующих значительных усилий, до очень легких схем. В течение последних 20 лет были предложены многие методы, среди которых наибольшее распространение получили следующие [4,6]:

Селекция с помощью пропорциональной рулетки. Метод рулетки – самый известный способ отбора. Основная идея этого подхода заключается в определении вероятности отбора (вероятности выживания) для каждой хромосомы пропорционально величине ее пригодности. Процесс отбора основан на вращении колеса рулетки столько раз, каков размер популяции, каждый раз выбирая одну хромосому для новой популяции. Колесо служит методом отбора как стохастическая выборочная процедура.

В качестве очень упрощенного примера рассмотрим популяцию из 10 индивидуумов. Пусть эта популяция содержит шесть идентич-

ных хромосом, которые обозначим как тип A и четыре идентичных хромосомы типа B. Допустим, что тип A имеет величину пригодности, равную 12, а тип B – 7. Общая пригодность составит величину:

$$6 \cdot 12 + 4 \cdot 7 = 100.$$

Тип A занимает 72% общей пригодности, а тип B – 28%. При таких входных данных «конструируем» рулетку, 72% площади которой обозначим меткой A, а 28% – меткой B. Для выбора следующей популяции даем рулетке только 10 оборотов (таков размер популяции) и создаем новую комбинацию из типов A и B. В созданной новой генерации будет содержаться, в среднем, около семи (основанном на 72%) членов типа A и трех (основанном на 28%) членов типа B. Поскольку вращение колеса рулетки и момент его остановки случайны (в компьютерной реализации используется случайный цифровой генератор), то действительное разделение между A и B может быть и 0 / 10, и 10 / 0, однако в среднем этот метод дает высокие значения пригодности.

Стохастическая остаточная селекция. Рассмотрим, как и выше, десятичленную популяцию типов A и B. Ожидаемое число копий типа A есть 7,2, а для типа B – это число составит 2,8. В этом методе селекции вначале концентрируемся на целых частях ожидаемых величин: 7 и 2. Семь копий типа A размещаются в следующей генерации так же, как и две копии типа B. Для определения идентичности 10-го члена остатки используются для «взвешивания» колеса рулетки. Здесь B получает 80% всего веса, а тип A – только 20%. Этот метод будет давать вклад в новую генерацию в соотношении 7/3 или 8/2 между типами A и B.

Стохастическая универсальная селекция. Этот метод – еще одна вариация предыдущих двух, но требует немного большего воображения для его реализации. Представим то же самое колесо рулетки, 72% площади которой отмечено как A, а 28% – как B. Дополним это колесо внешним кольцом из 10 равных интервалов (маркеров). Колесо рулетки делает один оборот. Для определения состава следующей генерации рассматриваются 10 маркеров: те, которые находятся напротив площади A, помечаются как тип A; то же самое относится и к B. Этот метод будет давать результаты, похожие на предыдущий случай: разделение в этом случае составит или 7/3, или 8/2.

Турнирная селекция. Турнирная селекция представляет собой пример процедуры, сочетающей одновременно случайный и детерминированный подход. Турниры похожи на маленькие «бои» между членами популяции, чтобы увидеть, кто примет участие в следующем

поколении. Здесь случайно определяется множество хромосом, из которого отбираются лучшие особи для репродукции. Число хромосом в этом множестве называется размером турнира, который чаще всего принимается равным 2 (бинарный турнир). После получения пары хромосомы с более высокой оценкой пригодности внедряются в новую популяцию. Процесс продолжается до тех пор, пока не заполнится вся популяция.

Ранжированная селекция. Эта процедура начинается с ранжирования популяции по величине пригодности. Далее вводится функция присваивания, которая дает каждой хромосоме вероятность включения в следующую генерацию. Хромосомы с более высоким рангом имеют большую вероятность включения. Функция присваивания может быть линейной или нелинейной (чаще – последняя). Колесо рулетки строится со слотами, определяемыми функцией присваивания. Последующая генерация n-размерной популяции определяется после n вращений колеса. Эта процедура продвигает селекцию к членам популяции, обладающими лучшими характеристиками.

Из перечисленных методов селекции наибольшее распространение получил метод пропорциональной рулетки, однако нужно указать некоторые недостатки процедуры пропорционального отбора. В этой схеме на ранних поколениях может проявиться тенденция доминировании «очень хороших» особей в процессе отбора; на более поздних генерациях конкуренция среди таких хромосом становится менее сильной, и в большей степени доминирует случайный поиск. Кроме указанных способов селекции, существуют так называемые элитные методы, которые гарантируют, что в процессе селекции будут сохраняться лучшие (в смысле пригодности) члены популяции. Наиболее часто используется процедура сохранения одной лучшей хромосомы, если она не прошла, как другие, через селекцию, скрещивание и мутацию. Элитный метод может быть применен в любую из указанных выше схем селекции.

Скрещивание

Скрещивание – это шаг, который реально усиливает генетические алгоритмы. Оно позволяет расширить поиск в различных направлениях, оценивая привлекательные решения и позволяя двум строкам «спариваться». Такой подход может привести к наследству, которое окажется более привлекательным, чем родители.

Пусть популяция $P^t = (a_1{}^t, ..., a_v{}^t)$ представляет собой репродукционную группу, т. е. совокупность v особей, любые две из которых $a_k{}^t$, $a_l{}^t \in P^t$, $k \neq 1$ могут размножаться, выступая в роли родителей ($a_k{}^t$ –

мать; a_l^t – отец). Здесь под размножением понимается свойство особей воспроизводить одного или нескольких себе подобных непосредственных потомков b_i^t, $i \geq 1$ и обеспечивать у них непрерывность и наследственную преемственность качественных признаков родителей. Таким образом, этот фактор эволюционного развития популяции приводит к получению новой генетической информации, содержащей различные комбинации аллельных форм генов родительских генотипов.

Воспроизводство себе подобных хромосом можно интерпретировать как возможность построения по заданным допустимым решениям $x_k, x_l \in D$ нового допустимого решения $x_i \in D$, а непрерывность и наследственную преемственность – как возможность использования аллельных форм в виде бинарных комбинаций $e_\theta(\beta_i)$, содержащихся в генотипах родителей $E(x_k)$ $E(x_l)$, для формирования генотипа $E(x_i)$ потомка, тем самым обеспечивая передачу наследственных признаков особей от поколения к поколению на уровне обмена генами.

Рассмотрим механизм размножения двух родительских особей $a_k^t, a_l^t \in P$ путем сигнамии (оплодотворения) их репродуктивных клеток: материнской гаметы (яйцеклетки) $E(a_k^t)$ и отцовской гаметы (сперматозоида) $E(a_l^t)$. В процессе сигнамии образуется родительская зигота (оплодотворенная клетка), способная развиваться в новую особь с передачей наследственных признаков (генетической информации) от родителей их потомкам. Зигота содержит одну пару из двух неотличимых одна от другой хромосом, которые происходят от «родительских» гамет: одна – от материнской гаметы, а другая – от отцовской гаметы. Такие хромосомы называются гомологичными хромосомами.

В результате акта сигнамии аллели родительских гамет могут поменяться местами в аллельных генах гомологичной хромосомы, что позволяет рассматривать следующие ситуация образования зигот:

– ген из отцовской хромосомы переходит в материнскую хромосому;

– ген из материнской хромосомы переходит в отцовскую хромосому;

– происходит взаимный обмен генами между материнской и отцовской хромосомами;

– отцовская и материнская хромосомы остаются без изменения.

Таким образом, при образовании зигот происходит независимое и случайное расхождение родительских генов по аллельным генам гомологичных хромосом зиготы независимо от того, у какой из родительских гамет они присутствовали до оплодотворения.

Заключительным этапом размножения особей является акт мейоза, под которым понимается процесс образования гамет из родительской зиготы путем независимого расхождения гомологичных хромосом по

дочерним гаметам, воспроизводящим потомство. Процесс размножения двух особей должен удовлетворять первому и второму законам наследственности Менделя, которые формулируются следующим образом:

– два гена, определяющие тот или иной признак, не сливаются и не растворяются один в другом, но остаются независимыми друг от друга, расщепляясь при формировании гамет;

– родительские гены, определяющие различные признаки, наследуются независимо друг от друга.

Согласно первому закону гены (или соответствующие им признаки родителей), имеющие одинаковые аллели $e_{\theta}^{OT}(\beta_i)=e_{\theta}^{M}(\beta_i)$, сохраняют свои значения в потомстве, т. е. передаются с вероятностью, равной 1, потомку по наследству. Гены родителей, имеющие разные аллели, передаются потомку по наследству с вероятностью, равной 0,5, т. е. половина гамет оказывается носителем аллели отца $e_{\theta}^{OT}(\beta_i)$, а другая половина – аллели матери $e_{\theta}^{M}(\beta_i)$.

В соответствии со вторым законом рекомбинация (обмен) генов в акте сигнамии может происходить либо в каком-то одном аллельном гене, либо в нескольких аллельных генах одновременно, т. е. передача аллелей от родителей потомству может происходить в каждом аллельном гене независимо друг от друга. При этом может оказаться, что гаметы потомков либо совпадают с родительскими гаметами, либо отличаются от них в одном или нескольких локусах.

Алгоритм скрещивания может быть описан такой последовательностью шагов.

1. Из популяции случайно выбираются два потенциальных родителя.

2. Выполняется жеребьевка (компьютерная) для определения – проводить или не проводить скрещивание.

3. При положительном варианте ответа на предыдущем шаге случайно выбирается точка расщепления.

4. Каждая исходная родительская строка перерезается в точке расщепления.

5. Производится обмен генетическим материалом слева и справа от точки расщепления и соединение полученных строк для формирования потомства.

Схема скрещивания показана на рис. 3.6.

Приведенная схема называется *одноточечным скрещиванием*, так как имеется только один разрез в каждой хромосоме.

Нужно отметить, что процедура скрещивания является весьма критичной для поиска лучших решений, поскольку хорошие с точки зрения пригодности строки могут быть сильно искажены. В случае, если скрещивание происходит достаточно часто, то поиск – хаотичен

Рис. 3.6

и неадекватно извлекает пользу из предыдущих успехов. Редкие (нечастые) скрещивания удерживают поиск в «узких» областях пространства поиска.

Наряду с одноточечным имеется двухточечное скрещивание, которое включает два разреза и может быть изображено так, как показано на рис. 3.7.

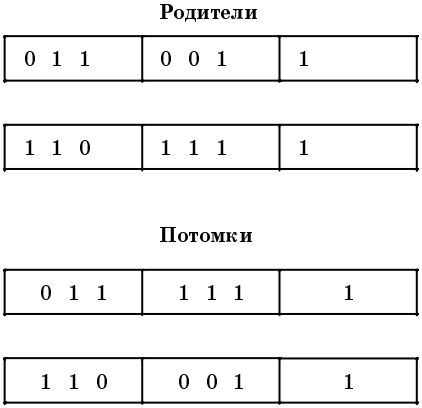


Рис. 3.7

Подобно большинству вариаций ГА двухточечное скрещивание имеет свои плюсы и минусы. Рассмотрим следующие две строки:

Строка 1: 1 * * * 0 1

Строка 2: * * 0 0 * *,

где знак * означает символ безразличия (на этом месте может находиться или 1, или 0).

При одноточечном скрещивании здесь нет возможности для строки 1 комбинироваться со строкой 2 без потери, по крайней мере, одной из единиц в концах строк. Однако при двухточечном скрещивании получится следующее:

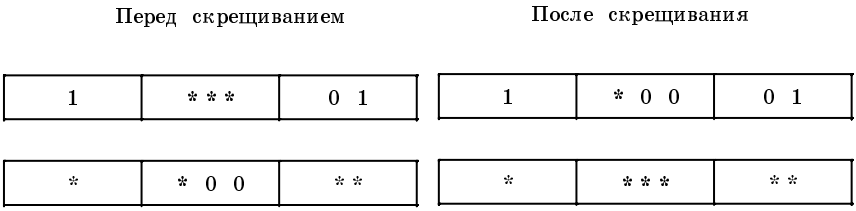


Рис. 3.8

Однако есть и другие схемы, которые даже при двухточечном скрещивании не могут комбинироваться должным образом. Для устранения таких нежелательных ситуаций были разработаны схемы многоточечного скрещивания, в частности равномерного[6]. Последнее потенциально позволяет любому образу совершать обмен и сформировать потомство. Здесь задача решается на побитовой основе (bit-by-bit basis) для выяснения того, как строки потомства соотносятся с родительскими строками.

В качестве примера рассмотрим две пятибитовые строки. Для создания скрещенного образа воспользуемся для каждой битовой позиции жребием (монетой), вместо орла и решки которого определим два исхода: «same – такой же» и «swap -обмен». Результат показан в табл. 3.4.

Таблица 3.4

Строки	Положение бита				
	1	2	3	4	5
Родитель 1	0	0	0	1	1
Родитель 2	1	1	1	0	0
Скрещенный образ	same	swap	same	swap	same
Потомок 1	0	1	0	0	1
Потомок 2	1	0	1	1	0

Далее полученный образ «same-swap-same-swap-same» используем для формирования двух потомков. Для первой битовой позиции

образ «same»; родитель 1 здесь имеет 0, поэтому потомок 1 получает также 0; родитель 2 – имеет 1, следовательно, потомок 2 получает 1. Для второй битовой позиции образ «swap», поэтому потомок 1 получает 1 от родителя 1, а потомок 2 – 0 от родителя 2. Окончательный результат приведен в последних двух строках табл. 3.4.

Равномерное скрещивание дает большую гибкость при комбинировании строк, что является привлекательным свойством при работе с ГА.

Подводя итог рассмотрению оператора скрещивания, отметим, что скрещиванию подвергается не вся популяция строк, а только ее часть, определяемая вероятностью скрещивания (например, $p_{\text{скр}} = 0{,}6$ означает, что лишь 60% строк из популяции будут образовывать родительские пары.

Мутация. При скрещивании генотипы потомков содержат новые сочетания аллельных форм генов родителей, ведущие к новым количественным признакам потомков (степени пригодности). Однако генетическая информация, содержащаяся в хромосомном наборе родителей и потомков, не меняется, так как в результате размножения особей путем сигнамии и мейоза частоты аллелей остаются постоянными, а меняются только частоты генотипов. Источником генетической изменчивости особей является мутация, под которой понимается изменение качественных признаков особей в результате появления новых аллельных форм в отдельных генах или целиком в хромосоме. Тем самым в каждом поколении мутации поставляют в хромосомный набор популяции множество различных генетических вариаций, присущих особям, которых в дальнейшем будем называть мутантами.

Процесс изменения содержания генов в хромосоме путем мутации называется мутагенезом. Самое главное заключается в том, что этот фактор эволюции популяции является источником новой генетической информации, не содержащейся ранее в генах родителей и их потомков.

Мутации происходят независимо от того, приносят ли они особи вред или пользу. Они не направлены на повышение или понижение степени приспособленности особи, а только производят структурные изменения в аллельных формах генов, что приводит к изменению количественных признаков особи. В принципе, комбинация мутаций может привести к возникновению новых форм аллелей в некоторых генах генотипа мутанта, которые обеспечивают увеличение его степени приспособленности к внешней среде.

Наиболее простым видом мутаций является точечная мутация, связанная с изменением аллеля родительского гена в одном из битов

генной информации (0 заменяется на 1 или наоборот), например, в строке вида 0 1 1 0 0 заменяется средний элемент, в результате чего получим такую строку: 0 1 0 0 0. Кроме того, мутация может быть инверсионной, при которой происходит изменение всех компонентов строки, например, строка 0 1 1 0 0 замещается строкой 1 0 0 1 1.

Оператор мутации вводит элемент вариабельности в популяцию, так как резко изменяет некоторое решение. Основным параметром мутации является ее вероятность, которую задает пользователь. Обычно ее величина достаточно мала (порядка 0,01 и ниже), чтобы, с одной стороны, расширить область поиска, а с другой – не привести к таким изменениям потомков, которые будут далеки от приемлемых решений.

3.5. Приемы выполнения генетических алгоритмов

После подробного ознакомления с принципом действия и назначением генетических операторов метод работы ГА целесообразно проиллюстрировать на конкретном примере. Основные шаги расчета при выполнении ГА были описаны выше в параграфе 3.2, где также приведена схема вычислений и действий.

Прежде чем перейти непосредственно к примеру, приведем некоторые комментарии, которые помогут лучше разобраться с сущностью ГА. Рассмотрим действие ГА на проблеме оптимизации функции многих переменных.

Отметим, что задачи минимизации и максимизации эквивалентны, так как:

$$\min f(x) = \max g(x) = \max \{-f(x)\},$$

где $g(x)=-f(x)$.

Кроме того, примем, что целевая функция f имеет положительную величину; в противном случае можно добавить некоторую положительную постоянную C:

$$\max g(x) = \max \{g(x)+C\}.$$

Пусть задача заключается в максимизации функции k переменных $f(x_1,...,x_к)$; при этом каждая переменная $x_i (i = \overline{1,k})$ принимает значения из области $D_i = [a_i, b_i]$ и $f(x_1,...,x_к) > 0$ для всех $x_i \in D_i$.

Установим требуемую точность оптимизации функции f: два знака после запятой. Ясно, что каждая область D_i должна быть разделена на $(b_i - a_i)\cdot 10^2$ равных отрезков.

Обозначим через m_i наименьшее число, удовлетворяющее неравенству

$$(b_i - a_i)\cdot 10^2 \le 2^{m_i} - 1.$$

Тогда каждая переменная x_i кодируется как бинарная строка длиной m_i, удовлетворяющая заданной точности.

Каждая хромосома (потенциальное решение) представляется бинарной строкой длиной $m = \sum_{i=1}^{k} m_i$. В этой строке первые m_1 битов отображают x_1 из диапазона $[a_1, b_1]$, вторые m_2 – из диапазона $[a_2, b_2]$ и т. д. В итоге хромосома может иметь такой вид:

$$\underbrace{01010}_{m_1}\underbrace{111}_{m_2}00\underbrace{1100}_{m_k}$$

Кроме того, зададим размер популяции M (число хромосом).

Далее работа ГА представляется вполне очевидной в соответствии с приведенным выше алгоритмом:

– в каждой генерации оценивается отдельная хромосома на предмет ее пригодности с использованием функции f на декодированном наборе переменных:

– отбирается новая популяция с учетом рассчитанной пригодности;

– с помощью операторов скрещивания и мутации хромосомы комбинируются в новую популяцию.

После некоторого числа генераций, когда не наблюдается улучшения популяции, лучшая хромосома представляет оптимальное (возможно глобальное) решение. Часто алгоритм останавливают после фиксированного числа итераций.

Рассмотрим некоторые шаги более подробно.

Для процесса селекции служит рулетка с размерами секторов, пропорциональных пригодности каждой строки. Разработка такой рулетки состоит из следующих шагов:

– вычисляется пригодность $\mu(a_j)$ для каждой хромосомы

$$a_j,\ j = \overline{1, M}\ ;$$

– находится общая функция пригодности всей популяции:

$$F = \sum_{j=1}^{M} \mu(a_j)\ ;$$

– определяется вероятность выбора p_j для каждой хромосомы a_j:

$$p_j = \mu(a_j) / F\ ;$$

– вычисляется кумулятивная (накопленная) вероятность q_j для каждой хромосомы:

$$q_j = \sum_{j=1}^{j^*} p_j\ .$$

Процесс селекции основан на вращении колеса M раз, и каждый раз отбирается одна хромосома в новую популяцию следующим образом:

– генерируется случайное число r из диапазона [0...1];

– если $r < q_1$, то выбирается первая хромосома a_1; в противном случае отбирается j-я хромосома a_j $(2 \le j \le M)$ так, чтобы $q_{j-1} < r \le q_j$.

Очевидно, что некоторые хромосомы будут выбраны больше, чем один раз. Лучшие хромосомы дают больше копий, средние – остаются неизменными, плохие – умирают. Новые решения на этом этапе не создаются.

Задается вероятность скрещивания p_c. Ожидаемое число хромосом, которые подвергаются скрещиванию, составляет $p_c \cdot M$.

Для каждой хромосомы в (новой) популяции:

– генерируется случайное число r из диапазона [0...1];

– если $r < p_c$, то данная хромосома выбирается для скрещивания.

Таким образом отбираются особи для скрещивания.

Выбор точки скрещивания тоже случаен: генерируется случайное число s из диапазона $[1...(m-1)]$ (m – длина хромосомы). Это число s определяет точку скрещивания.

В итоге две хромосомы ($b_1b_2...b_sb_{s+1}...b_m$) и ($c_1c_2...c_sc_{s+1}...c_m$) заменяются парой потомков $(b_1b_2...b_sc_{s+1}...c_m)$ и $(c_1c_2...c_sb_{s+1}...b_m)$.

Задается вероятность мутации p_m. Ожидаемое число измененных битов составит $p_m \cdot m \cdot M$. Каждый бит во всех хромосомах во всей популяции имеет равный шанс подвергнуться мутации, т. е. измениться с 0 на 1 или наоборот. Это осуществляется следующим образом:

– генерируется случайное число r из диапазона [0...1];

– если $r < p_m$, то бит изменяется.

После отбора, скрещивания и мутации новая популяция готова для дальнейшего оценивания. Полученные оценки используются для построения новой рулетки с секторами, пропорциональными текущим значениям функции пригодности. Остальная часть эволюции представляет по существу циклическое повторение процесса.

Воспользуемся приведенными пояснениями для решения следующей задачи [2]. Найти максимум функции:

$$f(x_1, x_2) = 21{,}5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2),$$

где $-3{,}0 \le x_1 \le 12{,}1$ и $4{,}1 \le x_2 \le 5{,}8$.

Кроме того, примем размер популяции $M = 20$; вероятности скрещивания $p_c = 0{,}25$; вероятности мутации $p_m = 0{,}01$.

Допустим, что требуемая точность составляет четыре цифры после запятой для каждой переменной. Тогда диапазон для переменной x_1, которой составляет 15,1, должен быть разделен на 15,1 *10000 равных отрезков. Это означает, что для первой части хромосомы потребуется 18 битов, так как

$$2^{17} \le 15100 \le 2^{18}.$$

Для второй переменной x_2 с диапазоном, равным 1,7, условие установленной точности требует, чтобы весь диапазон был разделен на 1,7*10000 равных отрезков. Таким образом, для этой переменной необходимо 15 битов, поскольку

$$2^{14} \le 17000 \le 2^{15}.$$

Общая длина хромосомы (вектор потенциального решения) составит $m = 18+15 = 33$ битов, из которых первые 18 кодируют первую переменную, а оставшиеся 15 – вторую. Рассмотрим, например, такую строку:

(010001001011010000 111110010100010).

Первые 18 битов определяют такое значение переменной x_1:

$$x_1 = -3,0 + decimal(010001001011010000_2) \cdot \frac{12,1-(-3,0)}{2^{18}-1} =$$

$$= -3,0 + 70352 \cdot \frac{15,1}{262143} = -3,0 + 4,05 = 1,05.$$

Оставшиеся 15 битов, декодированные по аналогии с вышеприведенным равенством, дают для переменной x_2 значение, равное 5,75. Таким образом, хромосома (010001001011010000 111110010100010) соответствует $(x_1, x_2) = (1,05; 5,75)$, что определяет для оптимизируемой функции следующее значение: $f(x_1, x_2) = f(1,05; 5,75) = 20,25$.

Создадим начальную популяцию, состоящую из 20 строк, в каждой из которых значения 33 битов инициируются случайным образом.

Положим, что после инициирования получена популяция, приведенная в табл. 3.5.

Теперь необходимо декодировать каждую хромосому и вычислить функцию пригодности каждой строки. Ясно, что в этом примере пригодность определяется самой оптимизируемой функцией. После декодирования получим результат, показанный в табл. 3.6.

Таблица 3.5

Номер строки	Строка-хромосома
1	10011010000000111111110100110111111
2	11100010010011011100101010011010
3	00001000001100100000101011011101
4	10001100010110100111100000111010
5	00011101100101001101011111000101
6	00010100001001010100101011111011
7	00100010000011010111101101111011
8	10000110000111010001101010100111
9	01000000010110001011000000111100
10	00000111100011000001101000011011
11	01100111111011010110000110111000
12	11010001011110110100010101000000
13	11101111101000100011000000100110
14	01001001100000101010011110010 1001
15	11101110110111000010001111101110
16	11001111000001111110000101001011
17	01101011111100111101000110111101
18	01110100000000111010011111010 1101
19	00010101001111111111000011000 1100
20	10111001011001111001100010111111 10

Из полученных данных видно, что вторая хромосома обладает наименьшей пригодностью, а хромосома a_{15} – наибольшей.

Перейдем к конструированию рулетки, необходимой для процесса селекции. Общая пригодность всей популяции составляет величину

$$F = \sum_{j=1}^{20} \mu(a_j) = 387{,}77.$$

Вероятности выбора p_j для каждой хромосомы в соответствии с вышеуказанным правилом приведены в табл. 3.7.

Кумулятивные вероятности для каждой хромосомы приведены в табл. 3.8.

Далее необходимо совершить 20 оборотов рулетки, каждый раз отбирая единственную хромосому для новой популяции. Пусть случайная последовательность 20 чисел из диапазона [0...1] имеет вид, показанный в табл. 3.9.

Таблица 3.6

Номер строки	Функция	Пригодность
1	f (6,08; 5,65)	26,01
2	f (I0,34; 4,38)	7,58
3	f (–2,51;4,39)	19,52
4	f (5,27; 5,59)	17,40
5	f (–1,25; 4,73)	25,34
6	f (–1,81; 4,39)	18,10
7	f (–0,99; 5,68)	16,02
8	f (4,91; 4,70)	17,95
9	f (0,79; 5,38)	16,12
10	f (–2,55; 4,79)	21,27
11	f (3,13; 4,99)	23,41
12	f (9,35; 4,23)	15,01
13	f (11,13; 5,37)	27,31
14	f (1,33; 5,15)	19,87
15	f (11,08; 5,05)	30,06
16	f (9,21; 4,99)	23,86
17	f (3,36; 4,57)	13,69
18	f (3,84; 5,15)	15,41
19	f (–1,74; 5,39)	20,09
20	f (7,93; 4,75)	13,66

Таблица 3.7

Номер строки	Вероятность p_j	Строка	Вероятность p_j
1	0,067	11	0,060
2	0,019	12	0,038
3	0,050	13	0,070
4	0,044	14	0,051
5	0,065	15	0,077
6	0,046	16	0,061
7	0,041	17	0,035
8	0,046	18	0,039
9	0,041	19	0,051
10	0,054	20	0,035

Таблица 3.8

Номер строки	Вероятность q_j	Строка	Вероятность q_j
1	0,067	11	0,538
2	0,086	12	0,577
3	0,137	13	0,647
4	0,181	14	0,698
5	0,247	15	0,776
6	0,293	16	0,837
7	0,335	17	0,873
8	0,381	18	0,912
9	0,423	19	0,964
10	0,478	20	1,000

Таблица 3.9

Разыгранные числа				
0,513	0,175	0,308	0,534	0,947
0,171	0,702	0,226	0,494	0,424
0,703	0,389	0,227	0,368	0,983
0,005	0,765	0,646	0,767	0,780

Первое число r_1 больше, чем q_{10} и меньше, чем q_{11}, поэтому для новой популяции выбирается хромосома a_{11}; второе число r_2 больше, чем q_3 и меньше, чем q_4. Следовательно, второй для новой популяции выбирается строка a_4 и т. д.

Окончательно новая популяция имеет вид, приведенный в табл. 3.10.

Как видно из табл. 3.10, худшая в начальной популяции строка 2 после селекции не попала в следующую генерацию, а лучшая в начальной популяции строка 15 появилась в новой популяции три раза.

Следующим шагом в проведении ГА является скрещивание, которое применим к полученной новой популяции. Заданная вероятность скрещивания составляет величину $p_c = 0{,}25$, поэтому в среднем должно подвергнуться скрещиванию 25% исходных хромосом. Здесь поступаем следующим образом: для каждой хромосомы в новой популяции генерируем случайное число r из диапазона [0...1]; если $r < 0{,}25$, то выбираем данную хромосому для скрещивания.

Допустим, что последовательность случайных чисел диапазона [0...1] получилась такая, как показано в табл. 3.11.

Таблица 3.10

Новый номер строки	Хромосома	Старый номер строки
1	01100111111011010110000110111 1000	11
2	10001100010110100111100000111 0010	4
3	00100010000011010111101101111 1011	7
4	01100111111011010110000110111 1000	11
5	00010101001111111111000011000 1100	19
6	10001100010110100111100000111 0010	4
7	11101110110111000010001111101 1110	15
8	00011101100101001101011111100 0101	5
9	01100111111011010110000110111 1000	11
10	00001000001100100000101011101 1101	3
11	11101110110111000010001111101 1110	15
12	01000000010110001011000000111 1100	9
13	00010100001001010100101011111 1011	6
14	10000110000111010001011010110 0111	8
15	10111001011001111001100010111 1110	20
16	10011010000000111111101001101 1111	1
17	00000111100011000001101000011 1011	10
18	11101111101000100011000000100 0110	13
19	11101110110111000010001111101 1110	15
20	11001111000001111110000110100 1011	16

Таблица 3.11

Разыгранные числа				
0,82	0,15	0,62	0,31	0,34
0,91	0,51	0,40	0,60	0,78
0,03	0,86	0,16	0,67	0,75
0,58	0,38	0,20	0,35	0,82

Из табл. 3.11 следует, что для скрещивания отбираются хромосомы с номерами 2, 11, 13 и 18, поскольку значения случайных чисел на этих позициях меньше, чем 0,25. Отметим, что в данном случае

число отобранных хромосом получилось четным, поэтому легко составить родительские пары. В противном случае необходимо добавить или убрать одну хромосому. Состав родительских пар также случаен, например, в качестве одной такой пары выберем строки a_2, a_{11} и другой – строки a_{13}, a_{18}. Для каждой из этих двух пар генерируем случайное число s из диапазона [1…32] (напомним, что 33 – общее число битов в хромосоме), которое определяет положение точки скрещивания. Для первой пары это число составит 9, а для второй – 20.

Первая пара хромосом

$a_2 = 100011000 \mid 101101001111000001110010$

$a_{11} = 111011101 \mid 101110000100011111011110$

после скрещивания дает такую пару потомков:

$a_2^* = 100011000 \mid 101110000100011111011110$

$a_{11}^* = 111011101 \mid 101101001111000001110010.$

Вторая пара хромосом

$a_{13} = 000101000010010101000 \mid 1010111111011$

$a_{18} = 111011111010001000110 \mid 0000001000110$

в результате скрещивания дает такую пару потомков:

$a_{13}^* = 000101000010010101000 \mid 0000001000110$

$a_{18} = 111011111010001000110 \mid 1010111111011.$

После скрещивания популяция хромосом принимает вид, показанный в табл. 3.12.

Перейдем теперь к оператору мутации, который выполняется на побитовой основе. Заданная вероятность мутации $p_m = 0{,}01$, поэтому ожидаемое число мутируемых битов составит 1% от общего числа битов в популяции. В последней имеется 33*20 = 660 битов. Следовательно, в среднем число битов-мутантов составит 6–7 единиц. Каждый бит в популяции имеет равный шанс подвергнуться мутации, поэтому для каждого бита генерируем случайное число r из диапазона [0...1]; если $r < 0{,}01$, то данный бит мутируется. В целом, необходимо разыграть 660 случайных чисел, из которых в данном случае только пять удовлетворяют требуемому условию. Положение бита и соответствующее значение случайного числа приведены в табл. 3.13.

Для определения положения мутированного бита в строках популяции воспользуемся табл. 3.14.

Таблица 3.12

Номер строки	Хромосома
1	01100111111011010110000110111000
2*	10001100010111000010001111011110
3	00100010000011010111011011111011
4	01100111111011010110000110111000
5	00010101001111111111000011000 1100
6	10001100010110100111100000111 0010
7	11101110110111000010001111011110
8	00011101100101001101011111000101
9	01100111111011010110000110111000
10	00001000001100100000101011101 1101
11*	11101110110110100111100000111 0010
12	01000000010110001011000000011111 00
13*	00010100001001010100000000100011 0
14	10000110000111010001011010110 0111
15	10111001011001111001100010111111 0
16	10011010000000111111101001101 1111
17	00000111100011000001101000011101 1
18*	11101111101000100011101011111101 1
19	11101110110111000010001111011110
20	11001111000001111110000110100101 1

Примечание. В табл. 3.12 скрещенные хромосомы отмечены звездочкой.

Таблица 3.13

Позиция бита	Случайное число
112	0,00021
349	0,00994
418	0,00880
429	0,00542
602	0,00283

Таблица 3.14

Позиция бита	Номер хромосомы	Номер бита в хромосоме
112	4	13
349	11	19
418	13	22
429	13	33
602	19	8

Из табл. 3.14 видно, что четыре хромосомы подверглись мутации, причем одна из строк с номером 13 дважды поменяла значения битов.

Окончательная популяция после операторов скрещивания и мутации приведена в табл. 3.15.

Таблица 3.15

Номер строки	Хромосома	Функция пригодности
1	01100111111011010110000110111 1000	f (3,13; 4,99) = 23,41
2*	10001100010111000010001111101 1110	f (5,27; 5,05) = 18,20
3	00100010000011010111101101111 1011	f (–0,99; 5,68) = 16,02
4**	01100111111001010110000110111 1000	f (3,13; 4,99) = 23,41
5	00010101001111111111000011000 1100	f (–1,74; 5,39) = 20,09
6	10001100010110100111100000111 0010	f (5,27; 5,59) = 17,40
7	11101110110111000010001111101 1110	f (1,08; 5,05) = 30,06
8	00011101100101001101011111100 0101	f (–1,25; 4,73) = 25,34
9	01100111111011010110000110111 1000	f (3,13; 4,99) = 23,41
10	00001000001100100000101011101 1101	f (–2,51; 4,39) = 19,52
11**	11101110110110100101100000111 0010	f (11,08; 4,74) = 33,35
12	01000000010110001011000000111 1100	f (0,79; 5,38) = 16,12
13**	00010100001001010100010000100 0111	f (–1,81; 4,20) = 22,69
14	10000110000111010001011010110 0111	f (4,91; 4,70) = 17,95
15	10111001011001111001100010111 1110	f (7,93; 4,75) = 13,66
16	10011010000000111111101001101 1111	f (6,08; 5,65) = 26,01
17	00000111100011000001101000011 1011	f (–2,55; 4,73) = 21,27
18*	11101111101000100011101011111 1011	f (11,13; 5,66) = 27,59
19**	11101111110111000010001111101 1110	f (11,05; 5,05) = 27,60
20	11001111000001111110000110100 1011	f (9,21; 4,99) = 23,86

Примечание. Мутированные строки отмечены двумя звездочками, а измененные биты выделены жирным шрифтом.

В этой же таблице в последнем столбце приведены значения функции пригодности, полученные для исходной популяции после селекции, скрещивания и мутации. Лучшая строка имеет значение функции пригодности, равное 33,35, что превышает наибольшую величину в исходной популяции. Кроме того, и общая пригодность, равная 447,04, намного превышает аналогичную величину в начале работы ГА. Таким образом, за один шаг процедуры выполнения ГА удалось значительно продвинуться вперед на пути поиска максимального значения рассматриваемой функции. Далее необходимо вновь применить селекцию, скрещивание и мутацию, оценить полученную генерацию с точки зрения ее пригодности и т. д. до тех пор, пока не будет удовлетворяться условие остановки.

3.6. Фундаментальная теорема генетических алгоритмов

Теоретические основы генетических алгоритмов базируются на бинарном представлении решений и понятии эталона-шаблона (schema), который позволяет исследовать сходство между хромосомами. (Отметим, что появление в русскоязычной научно-технической литературе термина «шима», см., например, [7] – искаженная транскрипция английского слова schema – является, по мнению автора, неверным и лишь загрязняет русский язык). Эталон создается введением *символа безразличия*, обозначаемого (*), в алфавит генов. Эталон представляет все строки, которые соответствуют ему на всех позициях, отличных от *. Например, рассмотрим строки и эталон одинаковой длины $m = 10$. Эталон (*111100100) соответствует двум строкам вида:

(0111100100), (1111100100),

а эталон (*1*1100100) – четырем строкам:

(0101100100), (0111100100), (1101100100), (1111100100).

Конечно, эталон вида (1001110001) определяет только одну такую же строку (1001110001), а эталон (**********) представляет все строки длиной $m = 10$. Ясно, что каждый эталон соответствует точно 2^r строкам, где r – число символов безразличия в эталоне.

Существуют два важных свойства эталона, которые будут использованы далее при выводе фундаментальной теоремы ГА. Первое свойство определяет *порядок эталона* $o(S)$ – число позиций 0 и 1, т. е. фиксированные позиции (не *), представленные в эталоне. Иначе, порядок эталона – это его длина минус число символов безразличия. Например, следующие три эталона одинаковой длины $m = 10$

$$S_1 = (***001*110),$$

$$S_2 = (****00**0*),$$
$$S_3 = (11101**001)$$

имеют такие значения порядка: $o(S_1) = 6$; $o(S_2) = 3$; $o(S_3) = 8$, причем эталон S_3 обладает наибольшим порядком.

Определение порядка необходимо при вычислении вероятности выживания эталона при мутации.

Второе свойство характеризуется *определяющей длиной* $\delta(S)$ эталона, которая представляет собой расстояние между первой и последней фиксированными позициями в строке. Это свойство дает оценку компактности информации, содержащейся в эталоне. Для указанных выше строк определяющая длина равна

$$\delta(S_1) = 10 - 4 = 6;\ \delta(S_2) = 9 - 5 = 4;\ \delta(S_3) = 10 - 1 = 9.$$

Понятие определяющей длины важно при вычислении вероятности выживания эталона при скрещивании.

Из материала, изложенного выше, следует, что имитацию эволюционного процесса можно представить в виде четырех последовательно повторяемых шагов:

– $t \leftarrow t + 1$;
– селекция $P(t)$ из $P(t - 1)$;
– рекомбинация $P(t)$;
– оценка $P(t)$.

Первый шаг представляет собой сдвиг эволюционных часов на одну единицу вперед; на последнем шаге производится оценка с точки пригодности текущей популяции. Основные явления эволюции имеют место на двух оставшихся шагах цикла: селекции и рекомбинации, при этом под последней подразумевается использование генетических операторов – скрещивания и мутации.

Начнем обсуждение с шага селекции и проиллюстрируем ход рассуждений на приведенном выше примере. Примем размер популяции $M = 20$; длина строки и эталона составляет $m = 33$ бита и в момент эволюции t популяции хромосом имеет вид, показанный в табл. 3.5.

Обозначим через $\xi(S,t)$ число строк в популяции в момент t, соответствующих эталону S. Пусть, например, эталон имеет вид:

$$S_0 = (****111**********************).$$

В этом случае $\xi(S_0,t) = 3$, так как в табл. 3.5 имеется три строки с номерами 13, 15 и 16, удовлетворяющих эталону S_0. Заметим, что порядок эталона $o(S_0) = 3$, а определяющая длина $\delta(S_0) = 7 - 5 = 2$.

Еще одним свойством эталона является его пригодность в момент t, которая определяется как средняя пригодность всех строк в попу-

ляции, отвечающих эталону S. Допустим, что в популяции есть p строк $\{a_1, a_2,..., a_p\}$, соответствующих эталону S в момент времени t. Тогда определяется средняя пригодность:

$$\mu(S,t) = \frac{1}{p}\sum_{j=1}^{p}\mu(a_j).$$

В процессе селекции создаются промежуточные популяции, состоящие из отдельных строк, отобранных на этой стадии. Каждая строка копируется нуль, один или больше раз в зависимости от ее пригодности. Строка имеет вероятность быть отобранной в следующую популяцию, равную $p_j = \mu(a_j)/F(t)$, где $F(t)$ – общая пригодность всей популяции в момент t.

После шага селекции ожидаем иметь $\xi(S,t+1)$ строк, соответствующих эталону S. Далее учтем следующее:

– для средней строки, определяемой эталоном S, вероятность ее отбора (при единственном выборе строки) есть $\mu(S,t)/F(t)$;

– число строк, удовлетворяющих эталону S, равно $\xi(S,t)$;

– число отборов отдельных строк определяется размером популяции M.

При указанных условиях для $\xi(S,t+1)$, соответствующих эталону S, можно записать следующее выражение:

$$\xi(S,t+1) = \xi(S,t)\cdot M\cdot\mu(S,t)/F(t). \qquad (3.3)$$

Формулу (3.3) можно преобразовать, учитывая, что средняя пригодность популяции $\overline{F}(t) = F(t)/M$, к виду:

$$\xi(S,t+1) = \xi(S,t)\cdot\mu(S,t)/\overline{F}(t). \qquad (3.4)$$

Как видно из формулы (3.4), число строк в популяции, отвечающих эталону S, растет пропорционально отношению пригодности эталона к средней пригодности популяции. Это означает, что выше средней пригодности эталон принимает увеличенное число строк в следующую генерацию, а ниже средней – уменьшенное количество строк, так что в среднем эталон остается на том же уровне. Уравнение (3.4) называется *уравнением репродуктивного роста эталона*.

Оценим долговременный эффект этого правила. Допустим, что эталон S остается выше среднего значения пригодности на ε%:

$$\mu(S,t) = \overline{F(t)} + \varepsilon\overline{F(t)}.$$

В этом случае

$$\xi(S,t) = \xi(S,0)(1+\varepsilon)^t. \qquad (3.5)$$

Формула (3.5) представляет собой геометрическую прогрессию, поэтому можно говорить, что при переходе от одной популяции к следующей эталон получает увеличенное число строк, подчиняющееся экспоненциальному закону.

Возвратимся к рассматриваемому примеру с эталоном S_0. Так как имеются три строки в популяции, удовлетворяющих эталону S_0, то оценка пригодности получается равной

$$\mu(S_0, t) = (27{,}31 + 30{,}06 + 23{,}86)/3 = 27{,}08.$$

В то же время средняя пригодность всей популяции составляет величину

$$\overline{F(t)} = M^{-1}\sum_{j=1}^{20}\mu(a_j) = 387{,}77/20 = 19{,}38,$$

и отношение пригодности эталона S_0 к средней пригодности популяции оказывается равным 27,08 / 19,38 = 1,39.

Данный результат означает, что в последующих генерациях эталон получит увеличенное по экспоненциальному закону число строк, удовлетворяющих эталону S_0. В частности, в популяции $t + 1$ ожидаемое число строк, отвечающих эталону $S_{0,}$ составит $3*1{,}39 = 4{,}17$ (наиболее вероятно между 4 и 5 строками); в популяции $t + 2$ получим: $3*1{,}39^2 = 5{,}79$ (скорее всего, между 5 и 6 строками) и т. д.

Проверим полученные предсказания на популяции, полученной в предыдущем параграфе (см. табл. 3.10). Если в исходной популяции число строк, удовлетворяющих эталону $S_{0,}$ было равно 3 (строки с номерами 13, 15 и 16), то в новой популяции $t + 1$ таких строк будет пять (строки с номерами 7, 11, 18, 19 и 20), что хорошо согласуется с предсказанным значением.

Как было сказано выше, селекция не создает новых точек (потенциальных решений) для поиска в пространстве решений. Селекция только копирует некоторые строки для создания промежуточных популяций. Вследствие этого необходим ввод в популяцию новых индивидуумов, что осуществляется посредством рекомбинации с помощью двух генетических операторов: скрещивания и мутации. Рассмотрим эффект этих двух операторов на ожидаемое число строк в популяции, удовлетворяющих эталону, поочередно.

Начнем со скрещивания и рассмотрим следующий пример. Возьмем одну строку из популяции (табл.3.10), например, номер 18, которая имеет вид:

(11101111101000100011000000100011 0).

Эта строка соответствует двум эталонам, к примеру таким:

$$S_0 = (****111**********),$$

$$S_1 = (111**********10).$$

Допустим далее, что строка с номером 18 скрещивается со строкой 13, а точка скрещивания находится после 20 позиции. Очевидно, что эталон S_0 выдержит такое скрещивание, поскольку один из потомков еще удовлетворяет S_0, так как последовательность «111» находится на пятой, шестой и седьмой позициях одного из потомков. Пара родителей из 18 и 13 строк (табл. 3.10):

$$a_{18} = (11101111101000100011 \mid 0000001000110),$$
$$a_{13} = (00010100001001010100 \mid 1010111111011)$$

дает таких потомков

$$a_{18}^* = (11101111101000100011 \mid 1010111111011),$$
$$a_{13}^* = (00010100001001010100 \mid 1010111111011).$$

С другой стороны, эталон S_1 будет разрушен, так как ни один из потомков не будет удовлетворять эталону: фиксированные позиции «111» в начале эталона и «10» в его конце находятся в различных потомках.

Из рассмотренного примера ясно, что определяющая длина эталона играет важную роль в его выживании или гибели. Заметим, что этот параметр для двух эталонов, соответственно, равен $\delta(S_0) = 2$ и $\delta(S_1) = 32$.

В общем случае точка скрещивания выбирается случайным образом из $m-1$ возможных позиций. Вследствие этого вероятность гибели (разрушения) эталона S равна

$$p_d(S) = \delta(S)/(m-1),$$

а вероятность выживания эталона S определяется как

$$p_s(S) = 1 - [\delta(S)/(m-1)].$$

В нашем примере вероятности выживания и разрушения эталонов S_0 и S_1 оказываются равными

$$p_d(S_0) = 2/32, \quad p_s(S_0) = 30/32, \quad p_d(S_1) = 32/32 = 1, \quad p_s(S_0) = 0.$$

Важно отметить, что только некоторые хромосомы подвергаются скрещиванию, количество которых определяется вероятностью скрещивания p_c. Это означает, что вероятность выживания схемы фактически есть

$$p_s(S) = 1 - p_c[\delta(S)/(m-1)]. \qquad (3.6)$$

Вновь, возвращаясь к рассматриваемому примеру, при $p_c = 0,25$ получим

$$p_s(S_0) = 1 - 0,25(2/32) = 0,98 .$$

Заметим, что даже в том случае, если точка скрещивания выбирается между фиксированными позициями эталона, еще сохраняется шанс выживания эталона. Например, если обе строки 18 и 13 начинаются с «111» и заканчиваются «10», то эталон S_1 выдержит скрещивание. Вследствие этого необходимо изменить формулу (3.6) выживания схемы следующим образом:

$$p_s(S) \geq 1 - p_c[\delta(S)/(m-1)]. \qquad (3.7)$$

Таким образом, комбинируя эффекты селекции и скрещивания, получим иную форму уравнения роста строк-представителей эталона в последующих генерациях:

$$\xi(S, t+1) \geq \xi(S, t) \bullet \mu(S, t) / \overline{F(t)}[1 - p_c \frac{\delta(S)}{m-1}]. \qquad (3.8)$$

Полученное уравнение (3.8) показывает, что ожидаемое число строк, удовлетворяющих эталону в следующей генерации, является функцией действительного количества хромосом, отвечающих эталону, относительной пригодности эталона и его определяющей длины.

Для эталона S_0 получим

$$\mu(S, t) / \overline{F(t)}[1 - p_c \frac{\delta(S)}{m-1}] = 1,39 \cdot 0,98 = 1,36.$$

Это означает, что эталон S_0 будет получать в следующих генерациях увеличивающее по экспоненциальному закону число строк: в момент $(t+1)$ ожидаем иметь $3 \cdot 1,36 = 4,08$ строк, удовлетворяющих эталону (только немного меньше, чем величина $4,17$ в случае одной селекции); в момент $(t+2) - 3 \cdot 1,37^2 = 5,63$ таких строк (слегка меньше величины $5,79$ для одной селекции).

Далее рассмотрим влияние оператора мутации на уравнение роста эталона. Как известно, этот оператор изменяет единственную позицию в пределах хромосомы с вероятностью p_m. Очевидно, что все фиксированные позиции эталона должны оставаться неизменными, если эталон выдерживает мутацию. Например, рассмотрим одну строку из популяции с номером 19:

11101110110111000010001111101111 0

и эталон S_0, равный

$$S_0 = (****111**********************).$$

Допустим, что строка 19 подверглась мутации на восьмой позиции, после чего она приняла такой вид:

$$11101111110111000010001111101111 0,$$

но еще отвечает эталону S_0. В случае, если позиции мутируемых битов находятся от 1 до 4 или от 8 до 33, то результирующий потомок еще будет удовлетворять эталону S_0. Важны только три бита, занимающие позиции с 5 по 7 в рассматриваемой строке, мутация, по крайней мере, одного из этих битов приведет к разрушению эталона S_0. Ясно, что число таких «важных» битов определяется порядком эталона, т. е. числом фиксированных позиций.

Вероятность изменения единственного бита равна p_m, вероятность выживания составляет величину $1 - p_m$. Одиночная мутация независима от других мутаций, поэтому вероятность выдержать мутацию для эталона S_0 равна

$$p_s(s) = (1 - p_m)^{o(S)}.$$

Поскольку $p_m << 1$, то последняя формула преобразуется к виду:

$$p_s(S) \approx 1 - o(S) \cdot p_m.$$

Возвращаясь к нашему примеру с эталоном S_0 при $p_m = 0{,}01$, получаем следующее значение для $p_s(S_0) = 1 - 3 \cdot 0{,}01 = 0{,}97$.

Объединяя эффекты селекции, скрещивания и мутации, получим новую запись уравнения роста строк в популяции, отвечающих эталону, в следующем виде:

$$\xi(S, t+1) \geq \xi(S, t) \cdot \mu(S, t) / \overline{F(t)} \left[1 - p_c \frac{\delta(S)}{m-1} - o(S) \cdot p_m \right]. \qquad (3.9)$$

Как и в предыдущих, более простых формулах (3.4) и (3.8), последнее выражение определяет ожидаемое число сходных с эталоном строк в последующих генерациях как функцию числа таких строк в текущей популяции, относительной пригодности эталона, его определяющей длины и порядка.

Для рассматриваемого примера с эталоном S_0 получим

$$\mu(S, t) / \overline{F(t)} \left[1 - p_c \frac{\delta(S)}{m-1} - o(S) \cdot p_m \right] = 1{,}39 \cdot 0{,}95 = 1{,}32.$$

В момент $(t + 1)$ ожидаем иметь $3 \cdot 1{,}32 = 3{,}96$ строк, определяемых эталоном (немного меньше, чем 4,17 в случае одной селекции и

4,08 при селекции и скрещивании); в момент $(t + 2) - 3 \cdot 1{,}32^2 = 5{,}22$ (также немного меньше, чем 5,79 и 5,63).

Выражение (3.9) определяет основную теорему ГА, иногда называемую теоремой эталонов, которую можно сформулировать следующим образом [2]:

Эталоны, обладающие малой определяющей длиной, низким порядком и пригодностью, выше средней в популяции, будут увеличивать число строк, сходных с эталоном, в последующих генерациях по экспоненциальному закону.

3.7. Примеры использования генетических алгоритмов в задачах менеджмента

Отметим еще раз, что генетические алгоритмы – это процедура, которая исследует пространство строк определенной длины для нахождения хромосом с относительно высокой величиной пригодности. При подготовке использования ГА к частной проблеме первый шаг состоит в определении пути представления задачи на языке, подобном хромосомам.

Первый пример связан с нахождением наилучшей бизнес-стратегии для четырех ресторанов, продающих гамбургеры [8]. В этой задаче менеджеру необходимо выбрать лучшую стратегию среди возможных трех бинарных решений:

– цена гамбургера (50 центов или 10 долл.);

– напиток (вино или кола);

– скорость обслуживания (медленная – официанты в смокингах, быстрая – официанты в униформе).

Цель заключается в нахождении комбинации этих трех решений (бизнес-стратегий), которая обеспечивает наибольший доход. Здесь имеются три переменные (цена, напиток, быстрота), каждая из которых может принимать одно из двух значений. Тогда каждую возможную бизнес-стратегию можно представить в виде строки длиной $m = 3$ с алфавитом $K = 2$. Для каждой переменной значения 0 или 1 присваиваются одному из возможных вариантов выбора. Пространство поиска в этой задаче состоит из $2^3 = 8$ возможных бизнес-стратегий. Выбор строки, состоящей из нулей и единиц, и отображения значений переменных в нули и единицы на определенных позициях строки, определяют схему представления для этой задачи.

Ограничимся четырьмя из восьми возможных бизнес-стратегий и представим их в табл. 3.16.

Положим, что при решении задачи уровень знаний менеджера в этой области бизнеса крайне низок, в частности, он не знает:

Таблица 3.16

Номер ресторана	Цена	Напиток	Обслуживание	Бинарное представление
1	Высокая	Кола	Быстрое	011
2	Высокая	Вино	Быстрое	001
3	Низкая	Кола	Медленное	110
4	Высокая	Вино	Медленное	010

– какая из трех переменных наиболее важная;

– каковы величины максимального дохода при оптимальном решении и потерь при ошибочном решении;

– какая единственная переменная дает наибольшее изменение в доходе.

Менеджер не знает также, приближается ли он к глобальному максимуму путем ступенчатого изменения одной переменной, фиксируя ее при лучшем результате, затем повторяя аналогичную процедуру с другой переменной и т. д. Возможно, переменные связаны некоторой зависимостью таким образом, что для достижения оптимума вначале необходимо идентифицировать и зафиксировать частную комбинацию двух переменных, а затем изменять оставшуюся.

У менеджера есть единственная обратная связь с внешней средой – это доход, получаемый каждым рестораном еженедельно.

Так как большей информации о внешней среде у менеджера нет, он может испытывать различные начальные случайные стратегии в каждом из четырех ресторанов. Стратегии, показанные в табл.3.16, можно использовать как начальные.

В действительности менеджер работает так же, как и ГА. Работа ГА начинается с попыток узнать что-либо об окружающей среде тестированием ряда случайно выбранных точек в пространстве поиска. Выполнение ГА начинается с нулевой генерации, т. е. с популяции, состоящей из случайно созданных особей. В нашем случае размер популяции $M = 4$.

В каждой генерации каждая строка (особь) оценивается с точки зрения приспособленности (пригодности) к внешней среде. Пригодностью в рассматриваемом примере может быть доход, прибыль, полезность и другие подобные характеристики.

В табл. 3.17 показаны значения функции пригодности $f(a_i)$ для рассматриваемого примера, где в качестве $f(a_i)$ принят десятичный эквивалент бинарной строки.

Таблица 3.17

Номер строки	Генерация 0		
	Строка a_i	$f(a_i)$	
1	011	3	0,25
2	001	1	0,08
3	110	6	0,50
4	010	2	0,17
Сумма		12	1,00
Худшее значение $f(a_i)$		1	
Среднее значение $f(a_i)$		3	
Лучшее значение $f(a_i)$		6	

Эта таблица показывает пригодность для каждой из четырех хромосом в случайной исходной популяции, разработанной для данной задачи. Анализируя табл. 3.17, менеджер узнает конкретные значения пригодности (дохода) для каждой из четырех точек (стратегий) в пространстве поиска. В частности, стратегия 110 дает доход, равный 6 долл. в неделю. Такая стратегия является лучшей в нулевой генерации, в то время как стратегия 001, создавая доход всего лишь 1 долл. в неделю, определяет худшую. Единственной информацией, используемой менеджером при выполнении ГА, служит величина меры пригодности отдельных строк, представленных в популяции. ГА, как было изложено выше, преобразует одну популяцию в другую посредством генетических операторов.

Применим к нулевой генерации вначале оператор селекции, который является аналогом дарвиновского оператора репродукции. Эту процедуру выполняем копированием строк текущей популяции в следующую генерацию с вероятностью, пропорциональной значению функции пригодности. Воспользуемся здесь алгоритмом рулетки, разделенной на четыре неравные части, которые пропорциональны относительной пригодности каждой строки (последний столбец табл. 3.17) и показаны на рис. 3.9.

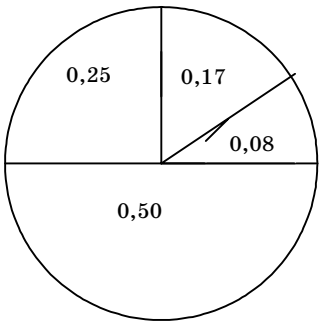


Рис. 3.9

Поясним более подробно построение и селекцию в рассматриваемом примере.

Сумма значений пригодности всех строк составляет 12 единиц, при этом лучшая строка в текущей популяции имеет пригодность, равную 6. Следовательно, часть пригодности всей популяции, вносимой строкой 110, есть 0,5. В этой схеме селекции хромосома 110 обладает вероятностью, равной 0,5, с которой она может быть отобрана на каждую из четырех позиций в новой популяции. Исходя из этого, можно ожидать, что строка 110 будет занимать две из четырех строк в новой генерации. Однако следует учесть, что ГА по своей природе носит вероятностный характер, поэтому есть шанс, что строка 110 может появиться три раза или один раз в новой популяции (есть гораздо меньший шанс появиться четыре раза или не появиться вовсе в новой генерации). Каждая строка в исходной популяции отображается сектором рулетки, чей размер пропорционален пригодности строки. Понятно, что строка 2 с наименьшей пригодностью имеет меньшие шансы оказаться выбранной, а строка 3 – наибольшие.

После четырех вращений колеса (поскольку размер популяции составляет $M = 4$) были отобраны такие строки: 3, 3, 1 и 4. В результате селекции строка 2 не вошла в новую популяцию, а строка 3 попала дважды. Родительский пул (набор строк, из которых путем скрещивания будут создаваться потомки) после селекции показан в табл.3.18.

Таблица 3.18

Номер строки	Строка a_i	$f(a_i)$
1	011	3
2	110	6
3	110	6
4	010	2
Сумма		17
Худшее значение $f(a_i)$		2
Среднее значение $f(a_i)$		4,25
Лучшее значение $f(a_i)$		6

Результатом селекции является улучшение средней пригодности популяции (4,25 вместо 3 в исходной популяции). Кроме того, здесь худшее значение пригодности составляет две единицы, а в исходной популяции – одну единицу. Такие улучшения в популяции типичны для схем селекции, потому что особи с низкой пригодностью имеют

тенденцию к исключению из популяции, в то время как строки с высокой пригодностью – к созданию дуплетов в новой генерации. Конечно, пригодность, связанная с наилучшей строкой в популяции, не может улучшиться в результате селекции, так как никаких новых хромосом в результате селекции не создается.

Перейдем к оператору скрещивания, который позволяет создать новые особи. В результате применения этого оператора в пространстве поиска формируются новые точки, которые проверяются с точки зрения их пригодности. Оператор скрещивания создает двух потомков, которые обычно отличаются от своих родителей и друг от друга. Каждый потомок содержит некоторый генетический материал, приобретенный им от каждого родителя.

Задаемся вероятностью скрещивания, например $p_c = 0{,}50$, что означает: только две строки из четырех будут подвергаться скрещиванию. Допустим, что для этой процедуры отобраны первые две особи из родительского пула: 011 и 110. Точка скрещивания определяется случайно, к примеру после второго бита, т. е. 01–1 и 11– 0.

В результате скрещивания получили два потомка: 010 и 111. В табл. 3.19 приведена первая генерация, полученная после оператора скрещивания.

Таблица 3.19

Номер строки	Строка a_i	$f(a_i)$
1	111	7
2	010	2
3	110	6
4	010	2
Сумма		17
Худшее значение $f(a_i)$		2
Среднее значение $f(a_i)$		4,25
Лучшее значение $f(a_i)$		7

Четыре строки в последней таблице представляют собой новую популяцию, созданную в результате применения селекции и скрещивания. Эти четыре особи есть первая генерация при использовании ГА.

Сравним результаты новой популяции с результатами исходной:

– средняя пригодность популяции увеличилась с 3 до 4,25;

– лучшая строка улучшилась с 6 до 7;

– худшая особь стала равной 2 вместо 1.

Лучшее решение, соответствующее строке 111, обеспечивает доход 7 долл. в неделю и является оптимальной стратегией. Таким образом, в этой задаче, не прибегая к еще одному оператору – мутации, добились оптимального результата, приводящего к такой бизнес-стратегии:

– цена гамбургера 50 центов;

– кола в качестве напитка;

– быстрое обслуживание.

Второй пример, взятый из области производственного менеджмента, связан с оптимизационной задачей поиска площади поперечного сечения определенной конструкции, схема которой показана на рис. 3.10. Конструкция представляет собой заделанную в стену с левой стороны консоль, состоящую из 10 элементов и нагруженную двумя грузами с правой стороны консоли. Цель расчета состоит в нахождении площади поперечного сечения каждого из 10 элементов для минимизации общей массы (или стоимости) материала.

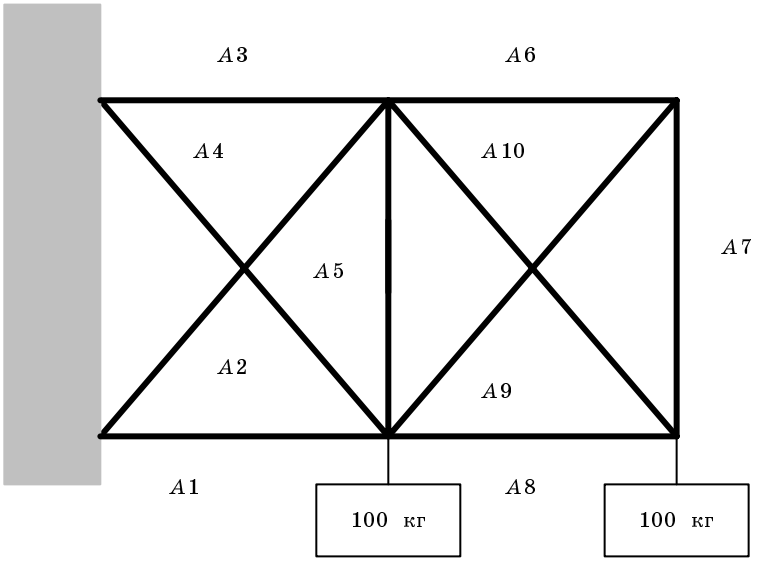


Рис. 3.10

Начальный шаг – создание схемы представления множества целых чисел как бинарной строки фиксированной длины, в которой каждое целое число ассоциируется с частью всей строки. Было предложено представить десять площадей поперечных сечений 40-битовой строкой, в которой площадь, равная 0,1 кв. дюйма, отобража-

лась четырьмя битами 0000, а площадь, равная 10,0 кв. дюймам, – битами 1111.

Каждый из оставшихся 14 битовых образов кодировался подходящими промежуточными значениями для площади поперечных сечений (так как для каждой площади используются четыре бита, а алфавит состоит из двух элементов, то всего образов $2^4 = 16$). Одна из возможных хромосом имеет такой вид:

0001	1110	0001	0011	1011	0011	1111	0011	0011	1010

Первые четыре бита в этой строке кодируют поперечную площадь $A1$, первого элемента конструкции. Каждая из оставшихся девяти площадей, соответственно, кодируется остальными наборами битов. Конечно, при построении хромосомы необходимо учитывать имеющийся в распоряжении разработчика сортамент используемого материала (балки, угольники, профили) и определять нижнюю границу площади поперечного сечения из условия допустимых нагрузок.

Другим важным моментом является выбор меры пригодности, которая должна показывать, насколько хорошо 10-элементная конструкция отображается 40-битовой строкой. В этой задаче было решено в качестве такой меры использовать общую стоимость материала для всех десяти элементов конструкции.

Кроме того, при выполнении ГА и его управлении существен выбор параметров. Двумя наиболее значимыми параметрами служат размер популяции, принятый в данной задаче равным $M = 200$, и максимальное число генераций до остановки алгоритма, установленное здесь равным $G = 40$.

Еще один пример использования ГА возьмем из области финансового прогнозирования [9]. Prediction Company (США) работает в области финансового консалтинга, и ее способность правильно (или с минимальной ошибкой) прогнозировать поведение временных рядов, описывающих рыночные показатели, является весьма привлекательной. В этой фирме для получения прогноза используются ГА. Для иллюстрации применяемого подхода предположим, что компания предсказывает поведение рынка, основываясь на 18 индикаторах. Этими индикаторами могут быть скользящие средние, алгебраические комбинации базовых компонентов, объемы продаж и т. д.

Методология применения ГА в этом случае состоит в следующем. Используемая хромосома есть перечень из 36 целых чисел. Каждые два из этих чисел ассоциированы с каждым из 18 индикаторов, при этом одно из чисел определяет нижнее значение индикатора, а другое – верхнее. Каждая хромосома оценивается определением того, насколько

хорошо был осуществлен «исторический» прогноз, когда каждое значение индикатора попадало между нижней и верхней границами, представленными в хромосоме. Результатом успешных проходов системы является множество хромосом, которые описывают комбинации значений используемых параметров. Для применения таких хромосом пользователю необходимо ожидать до тех пор, пока значения текущего индикатора не попадут в диапазон, определяемый одной из таких строк, и затем торговать акциями, основываясь на прогнозе системы.

В последние годы значительное внимание уделяется построению систем классификации (СК), основанных на применении ГА. Для большей наглядности рассмотрим вначале формирование такой СК в криминалистике (хотя она и не относится к сфере менеджмента), а затем – в области менеджмента [6].

Из многочисленных полицейских сериалов нам известно, как создается «портрет» предполагаемого преступника. Это достаточно трудоемкий и затратный по времени процесс. ПК помогают улучшить эту процедуру, используя библиотеки различных признаков лица, однако они имеют ограничения, так как полагаются на память свидетелей. Последние чаще всего в состоянии идентифицировать субъект по фотографии, но описать лицо по памяти представляется для них достаточно трудной задачей.

Система классификации с применением ГА работает с большой библиотекой основных черт лица. Пять блоков системы представляют образы лба, глаз, носа, рта, подбородка, которые позволяют генерировать 34 биллиона компонентов лица. Такие пять категорий закодированы как 35-битовая бинарная строка, состоящая из пяти семибитовых параметров. Пример такой строки может выглядеть следующим образом:

1001001	0011001	1101000	0101100	0010110

Система классификации начинает с популяции из 20 случайно сгенерированных лиц, которые отображаются на дисплее. Свидетели изучают представленные образы и присваивают каждому образу рейтинг в диапазоне от 1 до 9, который служит оценкой пригодности полученных «портретов». Вторая генерация образуется путем селекции, скрещивания и мутации. Свидетели оценивают вторую генерацию образов, и процесс продолжается аналогичным образом до установленного заранее критерия остановки или получения наиболее, по мнению свидетелей, достоверного «портрета». Описанный подход показал высокую эффективность, поскольку сходимость метода часто происходила за 20 генераций.

Система классификации в финансовом менеджменте – это ряд правил, которые выполняют определенные действия. Как пример, рассмотрим систему управления портфелем акций. Менеджер рассматривает акции многих компаний, решая, какие – купить, а какие – продать. Эти решения должны «вращаться» вокруг различных критериев, например, рост доходов, неуплаченная доля, узаконенная собственность, рост дивидендов и другие. Правила в СК могут быть закодированы так, как показано в табл. 3.20.

Таблица 3.20

Вход							Выход	
Номер примера	Рост доходов	Неуплаченная доля	Собственность	Рост дивидендов	Левередж	Цена / прибыль	Покупать	Продавать
1	1	#	#	#	1	#	1	0
2	1	#	#	#	1	0	1	0
3	#	1	#	#	#	#	0	1

Левая часть табл. 3.20 представляет собой вход, правая – выход.

Входная часть закодирована символами 0, 1 и #, которые создают триаду или трехчленный алфавит. Цифры 0 и 1 используются для определения, обладают или нет акции привлекательностью, например высоким ростом доходов. В случае, если какой-либо атрибут закодирован как «1» в рассматриваемой строке, то эта единица дает сигнал к тому, что акция обладает высоким значением указанного признака. Знак «#» в какой-либо позиции строки указывает на то, что данный признак не является определяющим фактором этого правила.

Выходная часть имеет две позиции. «1» в первом положении дает указание менеджеру покупать акции; «1» во второй позиции – указывает на продажу. Образы, которые определяют, что никаких действий нет, имеют такой вид: 00, 11, #0, 0#, # #.

С правилом, закодированным в табл.3.20, покупка определена для первого примера. Входной код 1# # # 1 # означает, что высокий рост доходов и приемлемый уровень левериджа определяют покупку, а все другие атрибуты не чувствительны к решению «покупать».

Второе правило – некоторая вариация первого. Кодировка 1 # # # 1 0 определяет, что высокий рост доходов и левередж важны, но акция должна еще иметь приемлемую величину отношения цена /

прибыль. В случае, если отношение цена / прибыль тоже высокое, то, возможно, за акцию назначена завышенная цена, и она не должна покупаться.

Третье правило с кодировкой # 1 # # # # является самым простым. Оно гласит, что при высокой неуплаченной доле наступает время продавать акцию независимо от значений других факторов.

Систему классификации можно создать следующим образом. Начальная популяция формируется случайным образом или путем отбора из данной серии правил. Процедура рандомизации работает с трехчленным алфавитом для образования, возможно, 100 начальных случайных правил. В СК правила содержат ассоциированный уровень устойчивости. В начальной точке всем правилам присваивается одинаковый уровень устойчивости.

Каждый цикл работы СК называется временным шагом. Правила предлагают в имитированном аукционе посмотреть, какие из них дают активацию на данном временном шаге. Здесь имеется много путей для создания СК в этом примере. Допустим, что каждое правило представляется в виде данных об акциях (базы данных), описанных ранее рассмотренными атрибутами. После просмотра базы данных некоторые правила могут быть применены, а некоторые – нет. Например, одно правило может «видеть» несколько акций, которые это правило советует купить, так как в этих акциях встречаются критерии покупки. Можно сказать, что база данных заставляет инициироваться некоторым правилам, которые затем могут участвовать в аукционе.

Надбавки цены на аукционе могут быть вычислены различными путями, но часто такие надбавки – определенный процент, например, 10% от уровня устойчивости. Здесь можно рассматривать предложения всех правил и выбрать предложение с наивысшей ценой. Однако СК обычно полагаются на «зашумленный» процесс предложений цены, в котором случайно генерированный шум добавляется к каждой величине предложения. Например, правило A может иметь уровень устойчивости, равный 100, и надбавку в 10 единиц. Правило B – уровень 95 и надбавку 9,5 единиц. Однако правило B может выиграть в аукционе, так как оно получает 1 единицу положительного шума, а правило A – 0,75 единиц отрицательного шума, что приводит к эффективным предложениям 9,3 и 10,5, соответственно.

В некоторых СК правила могут получать «премии», добавляемые к их эффективности. Правило-победитель может заставить покупать или продавать подходящие акции из базы данных. Кроме того, мож-

но допустить некоторый произвольный период задержки (к примеру, один год). Премия для правила тогда оказывается связанной с доходом, полученным через портфель в течение этого года, и может быть как положительной, так и отрицательной.

На втором временном шаге правила, способные конфликтовать с новыми данными, должны пройти еще один цикл: надбавка–аукцион–премия. При повторении этого процесса по многим временным шагам хорошие правила будут усиливаться, плохие – ослабляться. Новые правила вводятся в СК через ГА. Однако при построении СК имеется отличие от традиционного решения задач посредством ГА. Последние расширяют область поиска в пространстве решений, пытаясь найти оптимальное решение, в то время, как при создании СК стоит задача найти ряд адаптирующих правил, которые хорошо работают в комбинации, а не одно лучшее правило.

3.8. Генетические алгоритмы в искусственных нейронных сетях

Генетические алгоритмы могут использоваться в искусственных нейронных сетях для решения следующих задач:

– нахождение оптимальных значений весов;
– адаптация обучающего правила для ИНС;
– отбор входных переменных;
– выбор наилучшей архитектуры сети.

В первой задаче ГА выступает как альтернатива методу обратного распространения ошибки (ОРО) [10]. Обучение этим методом регулирует веса прямонаправленной ИНС, основываясь на принципах наискорейшего спуска. Один из основных недостатков этого классического алгоритма состоит в возможном попадании в локальные минимумы. Вследствие мутации в ГА последний имеет характеристики, аналогичные способу подъема на холм при поиске оптимума, что позволяет избежать опасности попадания в локальный минимум. Принцип использования ГА для обучения ИНС проиллюстрируем следующим примером.

Положим, что имеется трехслойная ИНС, схема которой показана на рис. 3.11.

В приведенной сети имеются 10 весов: $w_1 \ldots w_6$ между входным и скрытым слоями и $g_1 \ldots g_4$ между скрытым и выходным слоями. Входной и выходной образы, соответственно: $[x_1, x_2, x_3]^T$ и $[y_1, y_2]^T$; требуемый выходной вектор: $[d_1, d_2]^T$. Целевая функция в данном примере сводится к минимизации величины z, определяемой как

$$z = \left[(d_1 - y_1)^2 + (d_2 - y_2)^2 \right]^{0,5}.$$

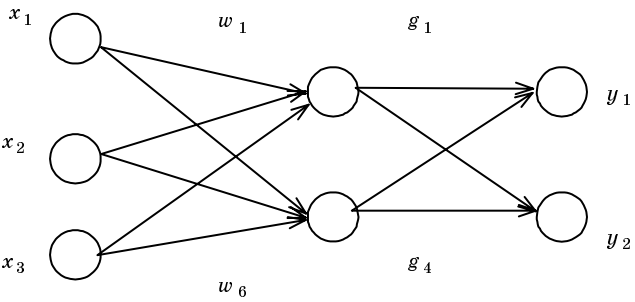


Рис. 3.11

Длина хромосомы составляет 10 битов, соответствующих весам: $w_1 \ldots w_6$, $g_1 \ldots g_4$. Каждый бит может быть представлен как действительной число в двоичной системе. Генетические операторы скрещивания и мутации применяются здесь для эволюции весов. Алгоритм завершается, когда прекращается улучшение величины z. Таким образом, для данного примера можно с помощью ГА найти требуемое множество весов.

Вторая задача – адаптация обучающего правила для НС – заключается в следующем. Супервизорная обучающая система должна генерировать требуемый выход для данных входных значений. Примем, что многослойная НС используется как обучающий агент. Пусть O_k – выход ячейки o_k в выходном слое; I_t – вход на t-м нейроне входного слоя; w_{ij} – вес от ячейки i к нейрону j.

Правило обучения в общем случае может быть записано как

$$\Delta w_{ij} = f(I_t, O_j, w_{ij}),$$

$$w_{ij}(t+) = w_{ij} + \Delta w_{ij}.$$

Допустим, что функция f(I_t, O_j, w_{ij}) имеет следующий вид

$$f(I_t, O_j, w_{ij}) = \sum_t a_t I_t + \sum_j b_j O_j + \sum_t \sum_i c_{ti} I_t O_i + \sum \sum d_{it} w_{it} I_t + \sum_i \sum_j e_{ij} O_j w_{ij}.$$

Здесь хромосомы конструируются из следующих параметров: a_t, b_i, $c_{ti,}$ d_{it}, e_{ij}. Пригодность хромосом измеряется сигналом ошибки, т. е. (целевой сигнал минус требуемый сигнал) на всех ячейках выходного слоя. Чем меньше величина этой ошибки, тем лучше качество хромосомы. После ряда генетических эволюций ГА определяет почти оптимальные значения параметров. Так как величины a_t, b_i, $c_{ti,}$ d_{it}, e_{ij} характеризуют обучающее правило, то их регулирование создает новые правила. На рис. 3.12 показан процесс адаптации обучающих правил.

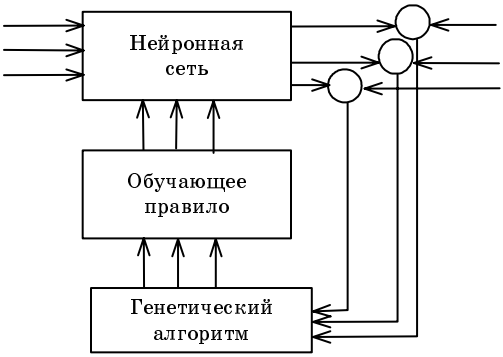


Рис. 3.12

Отбор входных переменных, являющийся целью третьей задачи, представляет собой один из самых трудных вопросов, которые приходится решать разработчику нейросетевых приложений. Качество работы сети можно улучшить путем уменьшения числа входных переменных, поскольку заранее не ясно, какие из входных параметров наиболее важны для решения задачи. Единственный способ получить гарантию, что входные данные выбраны наилучшим образом, состоит в том, чтобы перепробовать все возможные варианты входных наборов данных и выбрать наилучший. Однако практически это сделать невозможно, поэтому для решения такой проблемы можно выделить два подхода:

– преобразование всех исходных переменных в меньшее число обобщенных некоторым способом признаков, сохранив при этом максимум информации (этот прием широко используется в методе главных компонентов);

– выделение тех входных переменных, которые не вносят существенного вклада в работу сети, и их удаление.

Второй подход можно реализовать путем сочетания вероятностных нейронных сетей (ВНС) или обобщенно-регрессионных нейронных сетей (ОРНС) с ГА. Поскольку нашей целью является выполнение второго способа, то вначале укажем на характерные особенности таких сетей.

Обычные архитектуры ИНС, рассчитанные на управляемое обучение, предполагают построение параметрической модели по имеющимся обучающим данным, где в качестве параметров выступают веса. Параметрическая модель (сеть) по объему оказывается гораздо меньше, чем набор обучающих данных, и работает достаточно быстро, хотя время, затрачиваемое на обучение сети, может оказаться значи-

тельным. К задаче можно подойти с иных позиций: попытаться смоделировать искомое отображение непосредственно по обучающим данным. Преимущество такого подхода заключается в том, что здесь не требуется обучения, а недостаток в том, что в результате может получиться очень громоздкая модель, занимающая много памяти и медленно работающая.

Вероятностные нейронные сети и обобщенно-регрессионные нейронные сети как раз и представляют собой методы такого типа, замаскированные под ИНС, предназначенные, соответственно, для задач классификации и регрессии. Первый промежуточный слой в сетях этих типов состоит из радиальных элементов. Выходной сигнал радиального элемента является гауссовой функцией активации с центром в той точке, которая хранится в данном элементе. В последующих слоях из этих сигналов составляются оценки для плотностей вероятностей классов (для ВНС) или зависимой переменной регрессии (для ОРНС).

Далее для решения задачи отбора входных переменных будем рассматривать сочетание ВНС и ГА. Такого рода задача может быть реализована с помощью уже упоминавшегося в первом разделе пакета `Statistica Neural Networks`. ГА отбора входных данных, работающий со строками битов, в данной задаче с помощью таких строк создает маску. Каждая маска определяет, какие из входных переменных должны использоваться при построении ИНС: «No» означает, что эта переменная не используется, «Yes» – используется. ГА случайным образом создает популяцию таких строк, а затем применяет для отбора лучших строк генетические операторы. В качестве решения выбирается одна из строк последнего поколения.

Для иллюстрации описанного подхода воспользуемся сокращенными данными файла «`Credit`», приведенным в пакете SNN. Этот файл содержит девять входных признаков, характеризующих финансовое состояние заемщика, и одну выходную номинальную переменную «Risk», имеющую два значения: «Bad» и «Good». ИНС обучается на входных образах, формируя на выходе положительное или отрицательное заключение о состоянии потенциального заемщика. Обученная сеть, способная к обобщению, дает заключение о степени риска выдачи кредита новому клиенту. На рис. 3.13 показана часть исходных данных файла «`Credit`», которые были разделены на обучающую (150 строк) и контрольную (50 строк) выборки.

Через меню `Train-Auxiliary` (Обучение – Дополнительно) открываем диалоговое окно `Input Feature Selection` (Отбор входных переменных), которое показано на рис. 3.14.

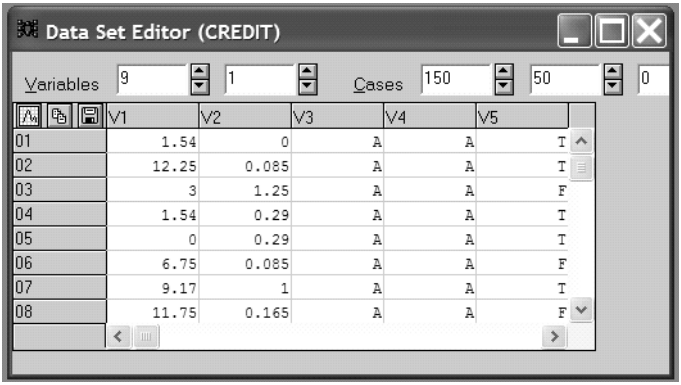


Рис. 3.13

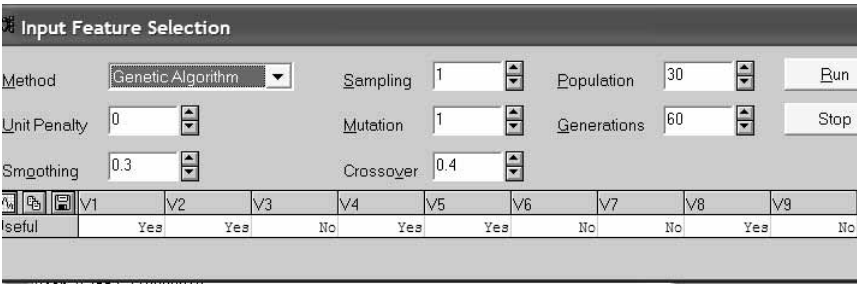


Рис. 3.14

Как видно из рис. 3.14, это окно содержит много управляющих параметров. Поясним лишь некоторые из них, остальные оставив по умолчанию.

Параметр сглаживания, равный здесь 0,3, используется при обучении ВНС и определяет ширину «колпаков» гауссовых функций с центрами в каждом обучающем наблюдении. В общем случае такие сети не слишком чувствительны к точному выбору этого параметра, и в данном случае вполне подойдет значение, взятое по умолчанию.

Параметры «Популяция» и «Генерация» определяют, соответственно, количество исходных строк, формирующих популяцию, и заданное число поколений, в течение которых выполняется эволюция.

После запуска через некоторое время (на ПК с процессором это занимает 23 с) в нижней строке окна выдается искомый результат. В итоге применения ГА для отбора входных данных получили, что в наборе используемых признаков остаются переменные: V_1, V_2, V_4, V_5, V_8; остальные четыре признака можно отбросить из рассмотрения.

Генетические алгоритмы используются также в задаче выбора архитектуры сети, т. е. при решении вопроса, сколько скрытых слоев и нейронов в них должно быть в составе сети. Начальная популяция выбирается таким образом, чтобы включить нейронные сети в широком диапазоне скрытых слоев и нейронов в них.

Генетические операторы в этой задаче имеют дело с закодированными строками различной длины, так как сети с большим числом скрытых нейронов будут иметь больше весов и, соответственно, большую длину строки. Кроме того, должна быть возможность использования нулевых весов для представления сети с меньшим числом нейронов. При эволюции ГА те сети, которые имеют наименьшие средние ошибки предсказания, будут доминировать в популяции, и последняя будет сходиться к оптимальной архитектуре сети.

3.9. Программное обеспечение генетических алгоритмов

Как отмечалось выше, одно из преимуществ использования генетических алгоритмов для решения различных задач заключается в том, что для реализации ГА не требуется создавать отдельный программный продукт. Единственное, что нужно от пользователя – это представить искомое решение в виде хромосомы и сформировать функцию пригодности. Далее реализация ГА происходит независимо от конкретики рассматриваемой задачи. Учитывая схему выполнения ГА, указанную ранее в этом разделе, очевидно, что процесс программирования не представляет значительных трудностей, так как реализация ГА есть циклическая процедура применения генетических операторов к исходной популяции хромосом. Вследствие этого можно рассмотреть некоторые доступные пакеты программного обеспечения в этой области.

Отметим, что в сети Интернет имеются в свободном доступе некоторые материалы по реализации ГА, однако здесь рассмотрим пакет `GeneHunter`, разработанной американской фирмой Ward Systems Group в 1995 г. В настоящее время этот пакет русифицирован российской компанией Нейропроект, являющейся официальным представителем Ward Systems Group на территории России.

В состав пакета `GeneHunter` входят дополнение `Microsoft Excel`, позволяющее пользователю решать оптимизационные задачи из рабочих листов `Excel`, Динамическая Библиотека (`Dynamic Link Library`) функций ГА, которые можно вызывать из таких языков программирования, как `MS Visual Basic` или СИ, и демонстрационные примеры. Поясним приемы использования этого пакета на одном из примеров, имеющихся в составе этого продукта, – в задаче оптимизации портфеля акций.

Финансовый менеджер должен пытаться решать задачу таким образом, чтобы минимизировать риски и одновременно максимизировать доход. Идея решения такой задачи заключается в том, чтобы минимизировать риск за счет создания диверсифицированного набора акций (дающих хорошую прибыль), похожего на набор акций в каком-либо из уже существующих оптимизированных портфелей. В качестве последнего можно взять доступный диверсифицированный портфель S&P500. Таким образом, задача состоит в разделении имеющихся в наличии денег на N частей с целью приобретения N акций. При этом новый портфель акций должен формироваться так, чтобы его динамика стоимости была как можно более сходна с динамикой стоимости портфеля S&P500.

Положим, что наш интерес проявляется к 15 акциям следующих компаний: IBM, GE,..., AT&T. Необходимо определить, сколько процентов $p_1, p_2,..., p_{15}$ от наличных денег должно быть вложено в каждую из них. К примеру, если $p_1 = 8\%$, то это означает, что 8% наличных денег инвестируются в акции IBM. Далее необходимо минимизировать различие между стоимостями «средней» акции в нашем портфеле и средней акции портфеля S&P500.

Средняя цена акции *OurAv* (аббревиатура *OurAv* соответствует «Нашему среднему») в нашем портфеле вычисляется как

$$OurAv = \frac{1}{100}\sum_{i=1}^{15} p_i C_i,$$

где p_i, C_i – соответственно, процент инвестированных денег и стоимость i-ой акции.

Сумма в последней формуле делится на 100, поскольку все значения p_i выражены в процентах.

Стоимость средней акции в портфеле S&P500 – это и есть индекс S&P, который обозначим $SP500$. Таким образом, нужно минимизировать различие между двумя величинами *OurAv* и $SP500$, которое в статистическом смысле рассчитывается как квадрат разности между указанными переменными:

$$(OurAv - SP500)^2.$$

Однако это квадратичное отклонение будет меняться с течением времени, поэтому лучше воспользоваться среднеквадратичным отклонением (СКО) за какой-то фиксированный промежуток времени, например 12 месяцев. Тогда целевая функция `GeneHunter` за этот период рассчитывается как

$$[(OurAv - SP500)^2]/12$$

и характеризует пригодность.

Следует заметить, что индекс *SP*500 не является простой стоимостью средней акции, и диапазон изменения *SP*500 значительно отличается от диапазона изменения *OurAv*. Вследствие этого необходимо перенормировать обе переменные в один и тот же диапазон для целей сравнения. Легче всего этого можно достичь путем нормировки каждой из величин по ее максимальному значению.

В пакете для решения данной задачи используются непрерывные целые хромосомы. Этот тип хромосом применяется в том случае, когда подбираемый параметр может принимать значения из некоторой непрерывной области, например значение 1,5 внутри диапазона от 0 до 2. Непрерывные хромосомы могут быть также целыми числами при необходимости ограничения пространства поиска. Каждая подбираемая ячейка может иметь собственную область значений.

Основное диалоговое окно `GeneHunter` приведено на рис. 3.15. Здесь указываются ячейка с целевой функцией (D12), выбираемые значения (максимум, минимум или фиксированная величина), к которым стремится алгоритм при поиске, подбираемые параметры (хромосомы) вместе с их типом, список ограничений.

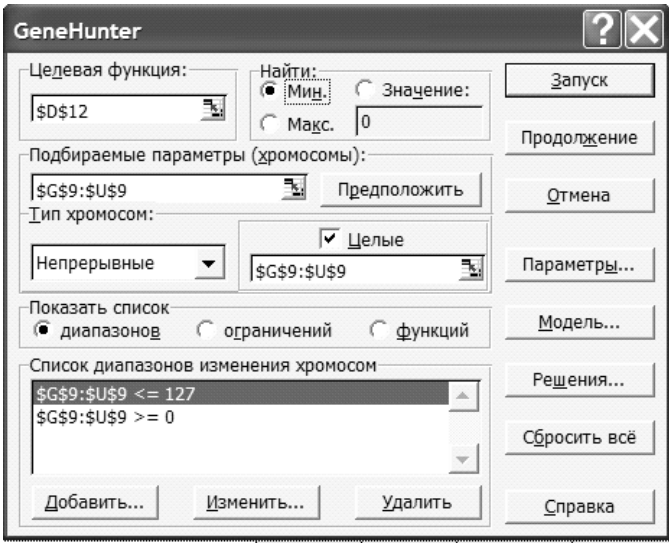


Рис. 3.15

В данной задаче используются непрерывные целые хромососы. При точности представления, равной 8 битам, диапазон изменения каж-

дой из хромосом составит от 0 до 127 единиц. Следующее диалоговое окно (рис. 3.16) позволяет установить параметры алгоритма.

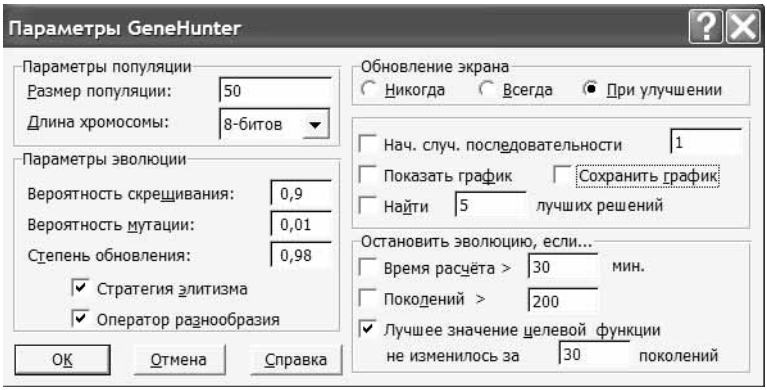


Рис. 3.16

В окне, кроме известных уже параметров вероятностей скрещивания и мутации, нужно указать степень обновления. Этот параметр определяет ту часть индивидуумов, которая не переходит в следующее поколение. Диапазон изменения этого параметра составляет величину от 0 до 1. К примеру, при значении степени обновления, равной 0,98, только 2% строк перейдут в следующее поколение, не подвергаясь скрещиванию и мутации, а 98% – погибнут. При размере популяции, составляющей 50 строк, только $50 \cdot 0{,}02 = 1$ индивидуум перейдет, не изменяясь, в следующее поколение. В случае включения стратегии элитизма (еще одна опция в окне установки параметров алгоритма) сохраненные строки являются наиболее подходящими (элита).

В некоторых случаях полезно вначале запустить `GeneHunter` с отключенным элитизмом, чтобы позволить популяции развиваться, не оказывая на процесс сильного селекционного давления. Через некоторое время можно включить элитизм, чтобы сконцентрировать процесс поиска оптимизации вокруг лучшего решения.

Включение оператора разнообразия (установки метки в левом нижнем углу) позволяет выполнить слабую форму мутации, производящую индивидуумы со слегка измененными свойствами, в противоположность сильным изменениям, которые выполняет стандартный оператор мутации. В некоторых задачах применение этого оператора дает улучшение процесса эволюции, в других – нет, однако в большинстве случаев его стоит использовать.

Часть исходных данных (базы данных за 1994 г.) с еженедельными значениями индекса S&P500 и ценами акций (показаны только шесть) приведены в табл. 3.21.

Таблица 3.21

		aig	bni	bt	dow	ed	gte
		American Intl Group	Burlington Northern	Bankers Trust	Dow Chemical	Consolidated Edison	GTE Corp
Дата	Индекс SP500	Страхова-ние	Транспорт	Банки	Химичес-кая промыш-ленность	Коммуналь-ные услуги	Связь
940107	469,9	84,875	58,5	78,75	59	31,375	34,625
940114	474,91	85,375	60	78,75	58,875	30,875	34,75
940121	474,72	89,75	62	79,625	62,875	29,75	34,5
940128	478,7	91,625	63,5	82,25	62,625	31	34,25
940204	469,81	88,125	62,875	82	64,125	30,375	33,25
940211	470,18	88,875	63	81,75	63,5	29,75	33,125
940218	467,69	88,5	63	80,5	66,375	28,875	31,75
940225	466,07	88,25	62	83,125	63,375	29,25	33,375
940304	464,74	85,75	61,75	76	64,5	30,625	32,375

Преобразование значений хромосом к процентам акций проводится по следующей формуле:

$$p = \frac{c}{\text{sum}} \cdot 100\%,$$

где p – процент акций; c – соответствующее (десятичный эквивалент) значение хромосомы; sum – сумма всех хромосом.

Такой прием гарантирует, что для каждой комбинации значений хромосом сумма процентов всегда составит величину 100%.

Таблица 3.22

	A	B	C	D
12			**Относитель-ное отклонение стоимости:**	**0,004623479**
13	Стоимость портфеля акций	Нормированная стоимость портфеля	Нормированный SP500	Квадратич-ное отклонение
14	45,65	0,9798	0,9816	3,19651E-06
15	45,96	0,9865	0,9921	3,07158E-05
16	45,86	0,9843	0,9917	5,41036E-05
17	46,59	1,0000	1,0000	0
18	45,66	0,9801	0,9814	1,86462E-06
19	45,65	0,9798	0,9822	5,5347E-06
20	45,58	0,9784	0,9770	1,94422E-06
21	45,65	0,9799	0,9736	3,89027E-05
22	45,35	0,9734	0,9708	6,58772E-06

Часть итоговых расчетов приведена в табл. 3.22, где наряду со стоимостями нашего портфеля и портфеля *SP*500 в ячейке D12 указано значение СКО за 52 недели.

Окончательно состав нашего портфеля с процентным составом приобретаемых акций приведен на рис. 3.17.

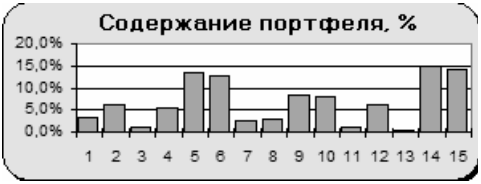


Рис. 3.17

Резюмируя результаты этой главы, можно сделать заключение о пригодности генетических алгоритмов, инспирированных биологической эволюцией, к решению таких задач менеджмента, в которых необходимо получить оптимальное решение. Осуществляя поиск по пространству параметров с помощью набора возможных решений, реализуемых хромосомами, можно, скорее, достичь оптимального (или близкого к нему) решения, чем в случае применения других методов.

Библиографический список

1. *Де Янг К.* Эволюционные вычисления: новейшие достижения и нерешенные проблемы// Обозрение прикладной и промышленной математики. 1996. Т. 3. Вып. 5.

2. *Michalewicz Z.* Genetics Algorithms + Data Structures = Evolution Programs. Berlin: Springer-Verlag, 1992.

3. Базы данных. Интеллектуальная обработка информации/ *В.В. Корнеев, А.Ф. Гареев, С.В. Васютин* и *др.* М.: Нолидж, 2000.

4. *Gen M., Cheng R.* Genetic Algorithms and Engineering Optimization. New-York, 2000.

5. *Holland J.* Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan, 1975.

6. *Bauer R.J.* Genetics Algorithms and Investment Strategies. New-York: J.Wiley&Sons, 1994.

7. *Романов В.П.* Интеллектуальные информационные системы в экономике: Учеб. пособие. М.: Экзамен, 2003.

8. *Koza J.R.* Genetic Programming. Cambridge, MA: MIT Press, 1992.

9. *Davis L.* Genetic Algorithms and Financial Applications// Trading on the Edge/ Ed. G. J. Deboek. New York: J.Wiley&Sons, 1994.

10. *Konar A.* Artificial Intelligence and Soft Computing. London: CRC Press, 2000.

Оглавление

Глава 1. Искусственные нейронные сети 5
1.1. Становление нейронной доктрины 5
1.2. Парадигмы обучения 12
1.3. Нейросетевые топологии 16
1.4. Алгоритмы обучения 19
1.5. Простые однослойные сети 20
1.6. Многослойные нейронные сети 28
1.7. Конкурентные сети 40
1.8. Алгоритмы решения задач с помощью нейронных сетей 46
1.9. Предварительная обработка данных 50
1.10. Нейронные сети в задачах менеджмента 56
Библиографический список 73
Глава 2. Нечеткая логика 74
2.1. Возникновение нечеткой логики 74
2.2. Нечеткие множества 78
2.3. Операции над нечеткими множествами 80
2.4. Построение функций принадлежности 83
2.5. Нечеткие и лингвистические переменные 88
2.6. Нечеткие алгоритмы и выводы 90
2.7. Формирование базы правил 101
2.8. Фазификация временных рядов 112
2.9. Нейро-нечеткие системы 114
2.10. Программные пакеты в области нечеткой логики .. 118
2.11. Использование нечеткой логики в задачах менеджмента 130
Библиографический список 140
Глава 3. Генетические алгоритмы 141
3.1. Сущность эволюционных вычислений 141
3.2. Основные понятия генетических алгоритмов 145
3.3. Кодирование в генетических алгоритмах 152
3.4. Генетические операторы 158
3.5. Приемы выполнения генетических алгоритмов 167
3.6. Фундаментальная теорема генетических алгоритмов 178
3.7. Примеры использования генетических алгоритмов в задачах менеджмента 185
3.8. Генетические алгоритмы в искусственных нейронных сетях 195
3.9. Программное обеспечение генетических алгоритмов 200
Библиографический список 205

Учебное издание

Кричевский Михаил Лазаревич

ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ В МЕНЕДЖМЕНТЕ

Учебное пособие

Редактор *А. М. Картухина*
Компьютерная верстка *А. Н. Колешко*

Сдано в набор 18.11.04. Подписано к печати 21.03.05. Формат 60×84 1/16. Бумага офсетная. Печать офсетная. Усл. печ. л. 12,09. Усл. кр.-отт. 12,21. Уч. -изд. л. 11,96. Тираж 100 экз. Заказ №

Редакционно-издательский отдел
Отдел электронных публикаций и библиографии библиотеки
Отдел оперативной полиграфии
СПбГУАП
190000, Санкт-Петербург, ул. Б. Морская, 67