



Уральский
федеральный
университет

имени первого Президента
России Б.Н. Ельцина

Институт радиоэлектроники
и информационных
технологий

П. Ф. ЧЕРНАВИН
Н. П. ЧЕРНАВИН
Ф. П. ЧЕРНАВИН

ПРАКТИЧЕСКИЙ КУРС
КЛАССИЧЕСКОГО МАШИННОГО
ОБУЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ
МОДЕЛЕЙ МАТЕМАТИЧЕСКОГО
ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие

Министерство науки и высшего образования
Российской Федерации

Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина

П. Ф. Чернавин, Н. П. Чернавин, Ф. П. Чернавин

**ПРАКТИЧЕСКИЙ КУРС
КЛАССИЧЕСКОГО МАШИННОГО ОБУЧЕНИЯ
С ИСПОЛЬЗОВАНИЕМ МОДЕЛЕЙ
МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ**

Учебно-методическое пособие

Рекомендовано методическим советом
Уральского федерального университета
для студентов вуза, обучающихся по направлениям подготовки
09.03.01, 09.04.01 — Информатика и вычислительная техника,
09.03.02, 09.04.02 — Информационные системы и технологии,
09.03.03, 09.04.03 — Прикладная информатика,
09.03.04, 09.04.04 — Программная инженерия

Екатеринбург
Издательство Уральского университета
2023

УДК 004.85 (075.8)

ББК 22.18я73

Ч-49

Рецензенты:

кафедра естественно-научных дисциплин Уральского государственного университета путей сообщения (завкафедрой д-р физ.-мат. наук, проф. Г. А. Тимофеева);

канд. физ.-мат. наук, ст. науч. сотр. Института математики и механики им. Н. Н. Красовского УрО РАН Д. Г. Ермаков

Научный редактор — канд. физ.-мат. наук, доц. А. С. Кощеев

Чернавин, Павел Федорович.

Ч-49 Практический курс классического машинного обучения с использованием моделей математического программирования : учебно-методическое пособие / П. Ф. Чернавин, Н. П. Чернавин, Ф. П. Чернавин ; М-во науки и высшего образования РФ. — Екатеринбург : Изд-во Урал. ун-та, 2023. — 124 с. — ISBN 978-5-7996-3767-5. — Текст : непосредственный.

ISBN 978-5-7996-3767-5

Учебно-методическое пособие посвящено изучению базовых методик машинного обучения для решения задач регрессии, классификации и кластеризации. Основной акцент делается на приобретении навыков решения практических задач различными методами, поэтому пособие содержит большое количество примеров, для решения которых приведены программы в кодах Wolfram Mathematica, Python, IBM ILOC CPLEX. Настоящее пособие может быть интересно и полезно как начинающим, так и опытным специалистам в области машинного обучения и может быть использовано для изучения (преподавания) данного курса. Приобретенные навыки в дальнейшем могут быть применены для решения задач из области исследования операций.

Табл. 23. Рис. 31. Прил. 10.

УДК 004.85 (075.8)
ББК 22.18я73

ISBN 978-5-7996-3767-5

© Уральский федеральный
университет, 2023

Оглавление

Введение.....	5
Глава 1. Математическое программирование и машинное обучение.....	10
1.1. Происхождение терминов «математическое программирование» и «машинное обучение»	10
1.2. Основные понятия машинного обучения и математического программирования.....	13
1.3. Признаки и их геометрическая интерпретация	15
Глава 2. Задачи регрессии	18
2.1. Линейная регрессия	18
2.2. Проблема мультиколлинеарности признаков.....	21
2.3. Задачи нелинейной регрессии	26
2.4. Применение линейной регрессии для моделирования кривых второго порядка	28
2.5. Немного философии	30
2.6. Общая постановка задачи математического программирования.....	31
2.7. Представление задач регрессии как ЗМП	32
2.8. Задачи регрессии повышенной сложности	35
Глава 3. Задачи классификации	38
3.1. Линейно разделимые множества	38
3.2. Линейно не разделимые множества и метод комитетов.....	44
3.3. Разновидности комитетных конструкций.....	48
3.4. Разделение множеств нелинейными функциями	62
Глава 4. Классические методы классификации.....	65
4.1. Метод опорных векторов	65
4.2. Логистическая регрессия	70
4.3. Основные идеи отнесения объектов наблюдения к конкретному классу	71

4.4. Метод ближайших соседей	72
4.5. Метод потенциальных функций (МПФ)	77
4.6. Логические методы.....	82
4.7. Ансамбли	84
4.8. Наивный метод Байеса (НМБ)	86
4.9. Нейронные сети	91
4.10. Градиентный спуск.....	93
4.11. Принципы построения эффективных ансамблей и оценка качества решения	97
4.12. Переобучение.....	99
4.13. Обобщенный взгляд на задачи регрессии, классификации, кластеризации и методы их решения.....	100
Глава 5. Решение задач регрессии и классификации на основе Python	103
5.1. Библиотеки и клише программ	103
5.2. Методы отбора, создания и нормализации признаков.....	104
5.3. Краткое описание программ для задач регрессии	107
5.4. Сравнение качества классификаторов	108
5.5. Использование других оптимизаторов для решения ЗМП	109
Заключение	114
Тестовые задания	116
Ответы к тестовым заданиям	119
Список библиографических ссылок	120

Введение

Период с 2012 по 2022 г. можно смело назвать десятилетием расцвета анализа больших данных на основе искусственного интеллекта (ИИ). Успехи в этом направлении достаточно хорошо описаны в [1–3], поэтому не будем повторяться. Более того, мы, как и ряд других специалистов в данной области [4; 5], считаем, что аббревиатуру ИИ более правильно расшифровывать как *имитация интеллекта*. Звучит, конечно, не так впечатляюще, но более точно отражает состояние дел в этой области на текущий момент, а опережающий маркетинг успехов ИИ приведет, скорее, к очередной его «зиме», чем к развитию.

Решающую роль ИИ в экономической и политической конкуренции отмечают практически все лидеры крупнейших стран мира [6–9]. Вкратце суть их высказываний состоит в следующем: кто будет лидером в этой гонке, тот и будет управлять миром. В настоящий момент есть две сверхдержавы в области ИИ. Об этом свидетельствует число патентных заявок в этой области, зарегистрированных в различных странах. По данным ВОИС (Всемирная организация интеллектуальной собственности) [1] в США патентных заявок 152 881, в Китае 137 010, во всех остальных странах 10 019.

К сожалению, Россия в числе остальных, и эту ситуацию надо менять. На наш взгляд, в первую очередь надо начинать с системы образования. В свое время Англия, Франция и Германия стали лидерами промышленной революции не только потому, что их великие математики Исаак Ньютон, Жозеф Луи Лагранж, Готфрид Лейбниц заложили основы математического анализа, а потому, что результаты их деятельности стали преподавать обычным инженерам, и они начали использовать эти знания в своих расчетах.

К тому же перед нами есть живой пример Билла Гейтса. В 1968 г., в первые годы становления вычислительных технологий, школа, где учился Билл Гейтс, купила компьютерное время у компании General Electric. Более того, им было предоставлено самое лучшее технологическое решение того времени на рынке вычислительных машин. В даль-

нейшем школе удалось договориться с Computer Center Corporation о предоставлении неограниченного компьютерного времени. Таким образом, можно говорить, что своевременное введение инноваций в учебный процесс позволило сформироваться Биллу Гейтсу как высококвалифицированному программисту, что напрямую повлияло на становлении корпорации Microsoft.

Билл Гейтс достиг своего успеха на заре цифровой революции в развитии вычислительных технологий. В свою очередь, по мнению многих экспертов, сегодня мы также находимся на заре новой информационной революции, основанной на технологиях анализа данных.

В данном учебно-методическом пособии мы сосредоточимся только на одном из направлений ИИ — машинном обучении (МО). Это направление мы предпочитаем называть «обучение по прецедентам», но аббревиатура МО (ML) уже крепко укоренилась в языке специалистов, поэтому ее мы тоже будем использовать.

Суть нашего предложения состоит в том, что основы МО необходимо преподавать не только узкому классу специалистов по анализу данных, но и широкому классу практических специалистов: медикам, металлургам, химикам, физикам, экономистам, банковским работникам, социологам, психологам, то есть практически всем — и, может, даже начиная со школы.

Глава крупнейшего российского банка Герман Греф, выступая в МГУ на конгрессе «Инновационная практика: наука плюс бизнес» [10] в числе недостатков современного российского образования отметил, что многие выпускники вузов не имеют нужных навыков, хотя и переполнены знаниями. При этом работодатели готовы им платить именно за навыки, а не за обширность знаний.

Мы также считаем, что обучение любому ремеслу или искусству надо начинать с приобретения простых и понятных обучаемому навыков, основанных на предыдущем опыте. Поэтому мы старались сделать изложение материала максимально простым и везде, где это возможно, приводить геометрические интерпретации и параллели с принятием решений в социуме, чтобы процесс освоения методов МО проходил естественным образом, как в обычной жизни.

Данное учебно-методическое пособие в первую очередь написано для тех практических специалистов, кто хочет идти в ногу со временем и техническим прогрессом, создавая свои собственные решения на рынке анализа данных.

Цели данного учебно-методического пособия состоят в следующем:

- научить построению решающих правил для различных задач машинного обучения в максимально доступной форме, с использованием геометрических и содержательных интерпретаций;
- дать практические навыки написания программ в кодах Wolfram Mathematica, Python, IBM ILOG CPLEX для решения сформулированных задач;
- показать, что приобретенные навыки в дальнейшем могут быть использованы и для решения задач из области исследования операций.

Учебно-методическое пособие устроено следующим образом. Все переменные приведены так, как они отражаются в программах (в прямом начертании шрифта), в том числе и математические знаки (например, знак умножения приведен через (*), как в программе Excel и др., а десятичные дроби оформлены через точку). Все примеры, используемые в пособии, представлены в электронном виде — файлах Excel, клише программ — в кодах Wolfram Mathematica, Python, IBM ILOG CPLEX в папках с соответствующим названием. Данные и программы можно найти в приложениях к данному пособию, размещенных на лазерных дисках.

По возможности мы старались соблюдать такую последовательность действий:

- 1) общие рассуждения о том, как возникает некоторая ситуация;
- 2) конкретный пример в удобном для использования виде;
- 3) геометрическая интерпретация;
- 4) математическая модель;
- 5) программа в кодах Wolfram Mathematica, Python, CPLEX;
- 6) результаты расчетов, их геометрическая и смысловая интерпретация, практические области, где эти навыки могут быть применены;
- 7) проверка результатов расчетов (примеры проверки приведены в файле Excel на отдельных листах);
- 8) примеры для самостоятельной работы с одним из возможных вариантов решения, геометрической интерпретацией, но без пояснений.

Данный курс будет построен по аналогии с практическим обучением вождению автомобиля. Главное — реально научить ездить, соблюдать правила дорожного движения и избегать ошибок при движении. Для этого в начале надо объяснить:

- 1) что такое руль, тормоз, газ (акселератор), коробка передач и куда заливать топливо (какого вида и какой марки);

2) как включаются фары, что означают показания приборов и зна-
ки дорожного движения.

Этих двух пунктов многим достаточно, чтобы успешно пользовать-
ся автомобилем всю жизнь. Более дотошные научатся менять колесо
в случае прокола (3). А если кто-то захочет сконструировать свой ав-
томобиль, то придется понять, как работает двигатель, АБС и прочие
агрегаты (4). Поэтому мы начнем с обучения самым простым поня-
тиям и навыкам, но ко многим вопросам будем возвращаться вновь
по ходу изложения, для того чтобы возникало более глубокое пони-
мание того или иного подхода к построению решающих правил. Для
решения задач мы будем использовать различные программные сред-
ства, следуя принципу «от простого к сложному».

Приведем забавный случай из нашей преподавательской практи-
ки. Даём студентам достаточно простую задачу регрессии небольшой
размерности и просим дать решение в аналитическом виде и на его ос-
нове разметить тестовую выборку. Большинство студентов успешно
ее решают средствами Excel, но один студент тестовую выборку раз-
метил (причем некачественно), а решения в виде уравнения регрес-
сии не представил. На вопрос «В чем причина?» он с гордостью отве-
тил, что решал задачу нейросетью из библиотеки Keras и прогнал аж
20 эпох. Оказалось, он просто толком не умеет пользоваться Excel —
только нейросетями.

Конечно, нейросети — мощнейший инструмент для решения раз-
личных задач МО и без них был бы невозможен наблюдаемый нами
прогресс в развитии ИИ, но надо понимать, при решении каких задач
они работают хорошо, а где — не очень.

Приведем цитату: «Организации все охотнее используют пере-
довые технологии глубокого обучения, но „классическое“ машин-
ное обучение по-прежнему будет играть важную роль, как сообщает
Gartner. Иллюстрируя этот тренд, Gartner дает прогноз, что в период
до 2022 года более чем три четверти организаций будут использовать
глубокие нейронные сети там, где вполне достаточно обычного ма-
шинного обучения. Это значит, что нужно понимать все доступные
варианты машинного обучения и не переусложнять путь к решению,
стремясь к тому, чтобы выбранный подход соответствовал сложности
решаемой проблемы» [11].

На наш взгляд, специалист по анализу данных должен уметь ис-
пользовать различные программные средства и не стрелять из пушки

по воробьям. Поэтому, если задачу можно успешно решить средствами Excel, мы покажем, как это делается. Если необходимо использовать более сложные программы, то приведем клише программ средствами Wolfram Mathematica и программами из библиотеки Scikit-learn. Если структура задачи еще более сложная, то покажем, как ее свести к задаче математического программирования и решить ее средствами IBM ILOC CPLEX или пакета MIP.

Несколько слов о сведении задач МО к задачам математического программирования (ЗМП). Мы часто применяем этот подход при решении реальных практических задач [12–20], и он дает хорошие результаты, так как позволяет вводить дополнительные ограничения и использовать различные функции цели. Данный подход достаточно подробно описан нами в [21]. Поэтому в данном учебно-методическом пособии мы не будем повторяться и дадим ссылки с указанием страниц. Аналогичным образом мы поступим и с рядом понятий из МО, потому что основная наша цель — дать практические навыки решения задачи МО.

В конце учебно-методического пособия приведены тестовые задания для самоконтроля с указанием главы, в которой есть ответ. Если ответ на вопрос вызовет у вас затруднение, то рекомендуем прочитать указанную главу еще раз.

Глава 1.

МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ И МАШИННОЕ ОБУЧЕНИЕ

1.1. Происхождение терминов «математическое программирование» и «машинное обучение»

Говоря о жизни, нельзя не согласиться, что она состоит из бесконечной череды принимаемых каждым отдельным человеком решений. Даже самые простые из них (например, во сколько встать с постели завтра утром) зависят от ряда факторов или параметров. В нашем примере — от состояния здоровья, от того, рабочий завтра день или выходной, назначены с утра встречи или нет, сколько необходимо времени на дорогу и т. п. Причем ряд параметров может быть конкретизирован более точно. Например, при оценке состояния здоровья можно положиться на самочувствие (хорошее или не очень) или измерить температуру и давление, то есть придать абстрактной оценке конкретные значения и на их основе оценить состояние здоровья, а затем решить: принять лекарство, лечь спать или вызвать скорую.

Оценка параметров и принятие решения происходят на основе собственного опыта или мнения экспертов (родителей, жены, друзей, справочников, рекламы). При этом собственный опыт может быть формализованным (любым способом, например, даже методом проб и ошибок) или неформализованным (интуитивным или случайным).

Все последующие действия (делать зарядку или нет; брать с собой зонтик или нет; каким видом транспорта добираться до работы; покупать или продавать; строить или ломать; делать операцию или нет; ставить оценку «хорошо» или «плохо» и т. д.) укладываются в простую схему, рис. 1.1.

Кроме того, на процесс принятия решения влияет система ограничений (наличие денег; мнение начальства, собственников; ограниче-

ние по химическому составу или физическим параметрам) и целевая установка (максимизация прибыли, минимизация рисков, стоимости изделия, соблюдение сроков строительства и т. п.).



Рис. 1.1. Схема процесса принятия решения

Таким образом, можно утверждать, что процесс выработки решения есть некоторая задача оптимизации целевой установки или установок при системе ограничений. Естественным является стремление к формализации процесса принятия решений, то есть к записи его в понятной для себя и других форме — такой, чтобы существовали методы решения поставленных задач за разумное количество времени и можно было оценить качество полученного решения.

Иными словами, необходимо описать данную задачу как задачу математического программирования.

Математическое программирование — это раздел математики, разрабатывающий теорию и численные методы решения многомерных задач нахождения оптимума целевой функции при заданной системе ограничений. Для ведения дальнейшего повествования необходимо дать определения основным понятиям в данной области знаний.

В рамках решения задач математического программирования нами изучается множество *объектов исследования*, описываемых совокупностью некоторых входных параметров. Например, в предыдущих примерах при принятии решения конкретным человеком параметром могла являться совокупность его знаний, опыта и мнений окружающих. В рамках математического программирования входные параметры объекта исследования называют *переменными*.

В свою очередь, для принятия решения необходима некоторая целевая установка, которая позволяет нам сравнивать решения и однозначно говорить, что одно из них лучше или хуже другого. На языке математического программирования такие целевые установки носят название *целевая функция*.

Из множества соотношений между переменными, системой ограничений и целевой функцией складывается *математическая модель задачи математического программирования* (ЗМП).

В рамках математического программирования выделяют обширный класс задач *линейного программирования*, в рамках которого математическая модель описывается исключительно линейными соотношениями. В дальнейшем в данной работе инструментарий линейного программирования будет использоваться нами достаточно часто для решения задач машинного обучения. Но прежде необходимо также определить, что представляет собой термин «машинаное обучение».

Машинное обучение (МО) — это обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных по мере накопления опыта относительно некоторого класса задач увеличивать качество их решения относительно выбранной целевой функции [21].

Конечно, приведенное выше определение не является единственным. Другие приведены нами в [21]. Вообще термин *машинаное обучение* (МО) — не совсем удачный перевод с английского на русский, из которого неспециалисту трудно понять, человек обучает машину или машина человека. Рассмотрим этот термин с гносеологической точки зрения.

В гносеологии (теории познания) принято разделять методы познания на индуктивные и дедуктивные:

- 1) индукция — переход от частного знания к общему (наиболее известным философом, развивающим этот подход, был Фрэнсис Бэкон);
- 2) дедукция — переход от общего знания к частному (наиболее яркими представителями этого подхода были Аристотель, Рене Декарт, Готфрид Лейбниц).

Естественно, эти два подхода диалектически взаимосвязаны и взаимно дополняют друг друга. Приведем примеры.

Пример индукции: Иоганн Кеплер на основе данных наблюдений за движением Марса в начале XVII в. открыл законы движения планет в Солнечной системе.

Пример дедукции: астрономическим примером будет вывод о существовании Нептуна (1846 г.) исходя из общего закона движения планет в Солнечной системе.

Другим ярким примером индуктивного подхода является периодический закон, открытый Д. М. Менделеевым в 1869 г., а дедуктивного — открытие предсказанных им элементов: галлия (Поль Лекок де Буабодран, 1875 г.), скандия (Ларс Нильсон, 1879 г.) и германия (Клеменс Винклер, 1886 г.) — экаалюминия, экабора и экасилиция соответственно.

С гносеологической точки зрения различают два вида обучения:

- 1) *обучение по прецедентам*, или *индуктивное обучение* — основано на выявлении эмпирических закономерностей в данных;
- 2) *дедуктивное обучение* — предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний (экспертные системы).

Обучение по прецедентам, общая постановка. Имеется множество *входных параметров* (ситуаций, результатов наблюдений) и множество возможных *выходных параметров* (ответов, результатов, откликов, реакций). Существует некоторая неизвестная нам зависимость между входными и выходными параметрами. Известна только конечная совокупность *прецедентов* — пар «объект — ответ», называемая *обучающей выборкой*. На основе этих данных требуется построить алгоритм, способный для любого возможного набора входных параметров выдавать достаточно точный прогноз выходных.

Термины «машинное обучение» и «обучение по прецедентам» можно считать синонимами.

Основные типы задач машинного обучения:

- 1) *обучение с учителем* (классификация, регрессия) — множество ответов существует;
- 2) *обучение без учителя* (кластеризация, фильтрация, сокращение размерности и иные) — множество ответов отсутствует.

В рамках данного учебно-методического пособия наше внимание будет сосредоточено на задачах регрессии, классификации, кластеризации и связанных с ними задачами информативности и генерации признаков. Поэтому дадим определение этим понятиям.

1.2. Основные понятия машинного обучения и математического программирования

Регрессия (лат. regressio — обратное движение, отход) в теории вероятностей и математической статистике — математическое выражение, отражающее зависимость зависимой переменной y от независимых переменных x при условии, что это выражение будет иметь статистическую значимость.

Этот термин в статистике впервые был использован Фрэнсисом Гальтоном (1886 г.) в связи с исследованием влияния роста отцов на рост сыновей. Было обнаружено, что в целом сыновья высоких отцов были более высокими, чем сыновья отцов с низким ростом. Гальтон обратил внимание, что разброс в росте сыновей был меньшим, чем разброс в росте отцов, то есть проявлялась тенденция возвращения роста сыновей к среднему (*regression to mediocrity*). С его легкой руки такие задачи стали называть задачами регрессии.

Классификация — один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов, разделенных некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Классовая принадлежность остальных объектов не известна. Требуется получить одну или несколько функций (решающих правил), способных классифицировать произвольный объект из исходного множества. Такие функции называются *дискриминантными*.

В соответствии с каноническим уравнением дискриминантной функции выделяются следующие виды переменных:

- 1) *классифицирующая (ависимая) переменная*, представляющая собой значение класса объекта;
- 2) *дискриминационные (независимые) переменные*, по которым выявляются различия между объектами с разной классовой принадлежностью.

В машинном обучении задачи, при которых нам известна классифицирующая переменная для части исследуемых объектов, относятся к разделу обучения с учителем. Стоит отметить, что существуют также задачи обучения без учителя, когда разделение объектов обучающей выборки на классы не задается и требуется классифицировать объекты на основе их сходства друг с другом. Такие задачи называются *задачами кластеризации или таксономии*, и классы называются, соответственно, *кластерами или таксонами*. Однако в рамках данной работы будут рассмотрены только задачи обучения с учителем.

И в качестве последнего определения рассмотрим понятие *искусственная нейронная сеть*. Если не касаться математической сути данного термина, то можно сказать, что это имитация функционирования биологических нейронных сетей живых организмов. Являясь, по сути, лишь одним из методов машинного обучения, искусственная нейронная сеть стала современным синонимом не только всего машинного

обучения, но и искусственного интеллекта в целом. Однако, несмотря на реально значимые практические результаты, достигнутые с применением нейронных сетей, не стоит забывать, что по своей математической сути нейронная сеть — задача математического программирования. Представленные в следующих главах комитетные конструкции будут примером простейшей однослойной нейронной сети.

Выборка — это конечный набор наблюдений, извлеченный из множества всех существующих наблюдений (*генеральной совокупности*). В рамках проведения машинного обучения выделяют два основных вида выборок:

- 1) *обучающая* — выборка данных, на которой обучается модель;
- 2) *контрольная (тестовая)* — выборка, по которой оценивается качество построенной модели.

Такое разделение генеральной совокупности позволяет избежать проблемы переобученности модели. Модель называется *переобученной*, когда средняя ошибка обученного алгоритма на наблюдениях из контрольной выборки значительно выше, чем на обучающей выборке.

Тест — это операция, при которой на основе материала обучения ведется распознавание объектов, ранее не участвовавших в обучении, однако принадлежность их к определенным классам заранее известна. Такой материал обучения называется *тестовой (контрольной) выборкой*.

Что означает термин *обучить*? Обучить значит подобрать коэффициенты в формуле (формулах) решающего правила:

- регрессия и классификация = обучение с учителем;
- кластеризация = обучение без учителя (классы не указаны, на них надо разбить обучающую выборку).

1.3. Признаки и их геометрическая интерпретация

Теперь, определив основные термины и понятия, начнем изложение материала с наглядных геометрических интерпретаций. Любой объект изучения (человек, ответственная деталь самолета, месторождение полезных ископаемых, фондовый рынок, состояние погоды и т. п.) может быть охарактеризован конечным набором параметров.

Начнем с человека. Каждый из нас имеет рост и вес. Если рассматривать только эти параметры, то математически любой человек может быть представлен как точка в двухмерном пространстве. При одном и том же росте люди могут иметь разный вес, и наоборот. Для полноты картины мы можем наращивать количество параметров. Соответственно, добавим возраст — и будет точка в трехмерном пространстве (рис. 1.2).

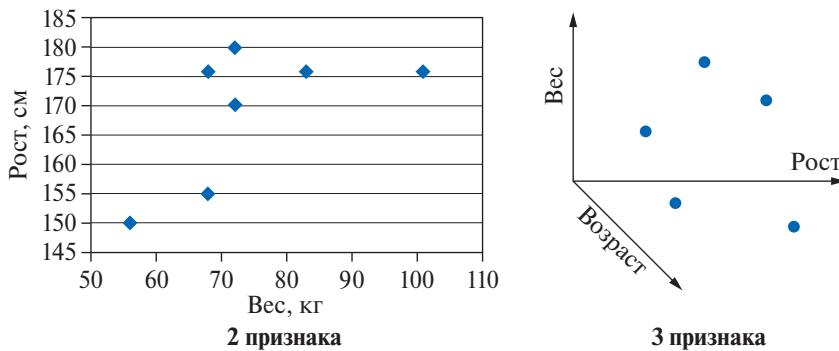


Рис. 1.2. Отображение человека как точки в пространстве признаков

Однако если мы попробуем двигаться дальше, например добавить пол и представить точку в четырехмерном пространстве, то уже возникают проблемы с визуализацией, так как пространства размерностью выше трех затруднительно даже мысленно представить.

Иными словами, начиная с четырехмерного пространства, понятная геометрическая интерпретация невозможна, но по сути это не имеет значения, так как математические методы не зависят от размерности пространства (числа рассматриваемых параметров). Поэтому попытаемся интерпретировать предлагаемый подход для случая двухмерного пространства, имея при этом в виду, что реальные задачи будут иметь размерности значительно более двух. Поэтому там, где двухмерной интерпретации недостаточно, будут следовать дополнительные пояснения.

Дальнейший набор параметров будет естественно зависеть от области изучения. Если нам необходимо принять решение (или сделать прогноз) о состоянии здоровья, то целесообразно добавить температуру тела, систолическое/диастолическое давление крови, возможно, химический состав крови и другие медицинские параметры.

Если нам необходимо принять решение о выдаче человеку кредита, то дополнительными параметрами могут быть: размер заработной платы; наличие в собственности квартиры, машины, дачи; состоит ли человек в браке; количество детей в семье; частота смены мест работы; сколько раз до этого человек брал кредит и были ли случаи невозврата кредита или задержки в уплате процентов. Могут быть также и другие социальные параметры в зависимости от желаемой глубины исследования и точности принимаемого решения.

После того как набор изучаемых параметров определен, необходимо сформировать два множества изучаемых объектов: одно для построения решающего правила (обучающая выборка) и другое для проверки решающего правила (контрольная выборка). Кроме того, нужно определиться, по какому признаку мы будем разделять множества: здоровый/больной человек, хороший/плохой заемщик,— и разделить их по этому признаку.

Различные виды признаков, способы их кодировки и приемы первоначальной обработки данных приведены нами в [21].

Глава 2.

ЗАДАЧИ РЕГРЕССИИ

2.1. Линейная регрессия

Начнем с простейших понятий. На рис. 2.1 отображены две точки в пространстве. Необходимо между ними провести прямую.

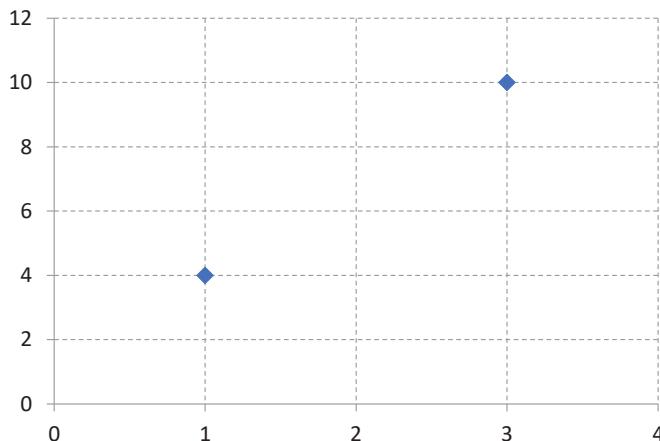


Рис. 2.1. Задача проведения прямой между двумя точками

Формула прямой записывается как $y = K \cdot x + b$. Для нахождения неизвестных нам K и b необходимо решить систему уравнений:

$$\begin{aligned} 4 &= K + b \\ (\text{или } b &= 4 - K) \rightarrow 10 = 3 \cdot K + 4 - K \rightarrow 6 = 2 \cdot K \rightarrow K = 3; b = 1 \\ 10 &= 3 \cdot K + b \end{aligned}$$

Значит, формулой прямой, соединяющей эти точки, является

$$y = 3 \cdot x + 1.$$

Теперь рассмотрим случай, когда точек несколько. Например, следующий (данные приведены в прил. 1):

P	10	20	30	40	50	60	70	80	90	100	110	120
Y	6	10	17.5	21	23.5	34	36	37	50.5	51	61.5	67

Здесь P — входные параметры, Y — выходные параметры. Графически это выглядит следующим образом (рис. 2.2).

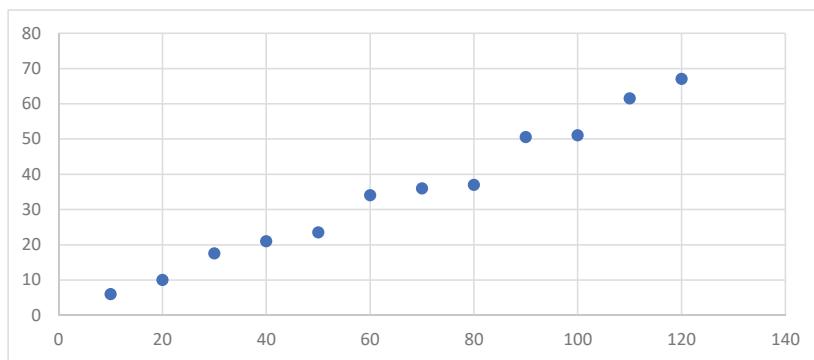


Рис. 2.2. Линейная регрессия: двухмерный случай

Очевидно, что провести прямую линию, которая проходила бы через все точки, невозможно, но можно провести прямую, которая в некотором смысле будет близка ко всем точкам. Мы не случайно использовали довольно расплывчатое определение «в некотором смысле», так как в зависимости от специфики задачи смысл может быть различным. Поэтому не будем пока вдаваться в нюансы, а используем pragматический подход: есть задача — ее надо решить, и желательно самым простым способом. При всем богатстве выбора, наверное, самый простой — решить ее в Excel.

Способ 1: выбираем в меню Данные/Анализ данных/Регрессия и указываем границы параметров. На отдельном листе получим решение задачи:

Коэффициенты	Стандартная ошибка	t-статистика	P-значение	Нижние 95%	Верхние 95%	Нижние 95.0%	Верхние 95.0%
-0,848484848	1,716938887	-0,494184653	0,631850644	-4,674063089	2,977093392	-4,674063089	2,977093392
0,545104895	0,023328606	23,36637274	4,6672E-10	0,493125523	0,597084268	0,493125523	0,597084268

Другими словами, уравнение линейной регрессии будет

$$Y = 0.545105 * x - 0.84848.$$

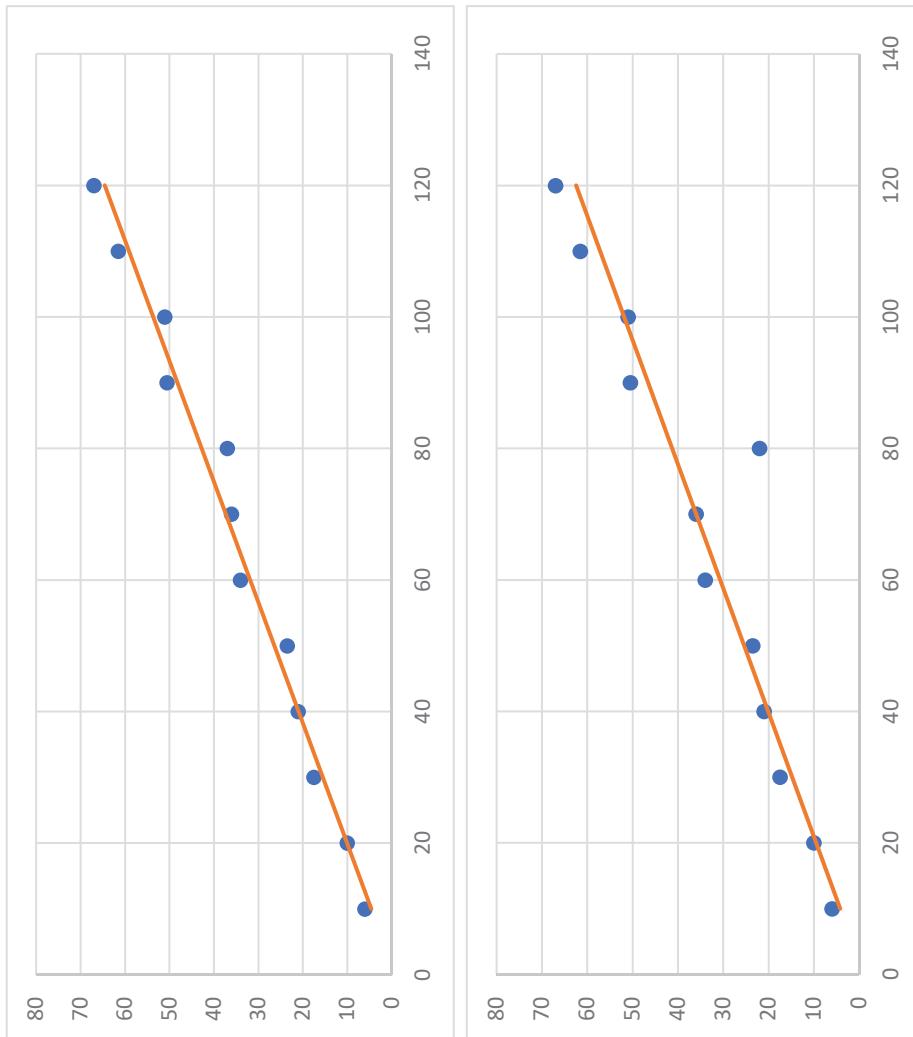


Рис. 2.3. Влияние случайного выброса на уравнение регрессии

Способ 2: используем оператор ЛИНЕЙН. При использовании данного оператора надо помнить, что если параметров несколько, то значение коэффициентов регрессии выдается, начиная с последнего параметра. Свободный член выдается в самом конце. Поэтому оператор ЛИНЕЙН лучше использовать в сочетании с оператором ИНДЕКС. Например, следующим образом:

	A	B
1	К	b
2	наклон	пересечение
3	=ИНДЕКС(ЛИНЕЙН(B5:B16;A5:A16);1)	=ИНДЕКС(ЛИНЕЙН(B5:B16;A5:A16);2)

Способ 1, конечно, проще, но он предполагает ручной режим работы. Способ 2 сложней, но он позволяет автоматизировать процесс вычисления уравнений регрессии и использовать их в дальнейших расчетах.

В прил. 1 приведены два примера. Они отличаются всего одной точкой, которая значительно отклоняется от других — такое достаточно часто случается в практических задачах. Случайные выбросы происходят по разным причинам, но они возможны, и надо понимать, как они влияют на решающее правило и как демпфировать их отрицательное влияние. Поэтому предлагаем описанными выше способами решить случай_2, сделать геометрическую интерпретацию и проанализировать полученное решение.

Графически это будет выглядеть следующим образом (рис. 2.3).

Из приведенного решения видно, что в данном случае изменился угол наклона уравнения регрессии и прямая, отображающая уравнение регрессии, сдвинулась в сторону случайного выброса.

2.2. Проблема мультиколлинеарности признаков

Начнем рассуждать последовательно. Если два признака абсолютно одинаковы во всех наблюдениях, то один из них лишний. А если один признак строго пропорционален другому? Например, среди признаков есть рост в сантиметрах и дюймах или вес в килограммах и фун-

так. Исходя из здравого смысла, надо оставить один, причем любой. Потому что если мы будем, например, искать уравнение регрессии, то влияние любого входного признака на выходной можно отрегулировать через изменение коэффициента у данного входного признака в уравнении регрессии.

На здравый смысл мы будем ссылаться и далее, но, исходя только из него, невозможно объяснить, что плохого в том, что в данных будут присутствовать два линейно зависимых признака (именно так, строго математически, правильно называть описываемую ситуацию). Поэтому для дальнейших пояснений необходимо вспомнить, как решаются системы линейных уравнений.

Рассмотрим два простых примера. Пусть необходимо найти решение двух систем уравнений:

система 1 (столбцы линейно независимы):

$$2*x_1 + 5*x_2 + 3*x_3 = 8,$$

$$3*x_1 + 1*x_2 + 7*x_3 = 11,$$

$$5*x_1 + 2*x_2 + 4*x_3 = 7;$$

система 2 (второй столбец = столбец 1*2):

$$2*x_1 + 4*x_2 + 3*x_3 = 8,$$

$$3*x_1 + 6*x_2 + 7*x_3 = 11,$$

$$5*x_1 + 10*x_2 + 4*x_3 = 7.$$

Для простоты пояснения рассматриваем случай, когда число переменных равно числу уравнений. В матричном виде задача может быть записана следующим образом: $A*x = b$.

Если обе части этого матричного уравнения умножить на обратную матрицу A^{-1} , то получим $E*x = A^{-1}*b$, где E — единичная матрица.

Таким образом, для того чтобы решить данные системы уравнений, необходимо вычислить обратные матрицы и умножить их на вектор b . Сделаем эти вычисления средствами Excel для системы 1 (табл. 2.1).

В ячейке G2 стоит формула =МОБР (A2:C4), в ячейке K2 — формула =МУМНОЖ (G2:I4; E2:E4), в ячейке L2 =МУМНОЖ (A2:C4; K2:K4). Не забываем, что поскольку в данном случае мы работаем с массивами, то первоначально выделяем место под массив, а ввод формулы завершаем Shift-Ctrl-Enter.

Таблица 2.1

Решение системы 1

	A	B	C	D	E	F	G	H	I	J	K	L
1	Прямая матрица A			b		Обратная матрица				решение	проверка	
2	2	5	3	8	-0.10204	-0.14286	0.326531		-0.10204	8		
3	3	1	7	11	0.234694	-0.07143	-0.05102		0.734694	11		
4	5	2	4	7	0.010204	0.214286	-0.13265		1.510204	7		

В данном случае решение задачи не вызывает никаких сложностей. Попробуем аналогичным образом решить систему 2 (табл. 2.2).

Таблица 2.2

Решение системы 2

Прямая матрица A			b	Обратная матрица		
2	4	3	8	#ЧИСЛО!	#ЧИСЛО!	#ЧИСЛО!
3	6	7	11	#ЧИСЛО!	#ЧИСЛО!	#ЧИСЛО!
5	10	4	7	#ЧИСЛО!	#ЧИСЛО!	#ЧИСЛО!

Из приведенной таблицы видно, что вычислить обратную матрицу не получается. Это происходит потому, что при наличии линейно зависимых столбцов или строк матрица становится вырожденной. Вырожденная матрица не имеет обратной, значит, задача не имеет решения.

Конечно, в задачах машинного обучения квадратные матрицы встречаются редко, и требуется более сложное пояснение, какие проблемы создают линейно зависимые признаки, но суть от этого не меняется: присутствие в исходных данных линейно зависимых признаков может привести к абсурдным решениям. Более того, если признаки линейно независимы, но сильно коррелируются, то возникает та же проблема.

Настала пора дать строгие определения. Если один из признаков может быть представлен как линейная комбинация других признаков, в которой как минимум один из коэффициентов не равен нулю, то такие признаки называются *линейно зависимыми*. Если признаки строго линейно зависимы, их называют *коллинеарными*. Если признаки сильно коррелируют друг с другом, то их называют *мультиколлинеарными*.

Поскольку коллинеарность и мультиколлинеарность признаков сильно влияет на качество решающих правил, попробуем пояснить

данную проблему по-другому. Пусть в результате решения регрессионной задачи мы получили следующее уравнение регрессии:

$$Y = a_1 * x_1 + a_2 * x_2 + a_3 * x_3 + b. \quad (2.1)$$

Допустим, что признак x_3 линейно зависит от остальных. Для простоты пусть он равняется их сумме, то есть $x_3 = x_1 + x_2$. Сделаем преобразование, скорректируем коэффициенты уравнения на некоторую произвольную константу с следующим образом:

$$Y = (a_1 - c) * x_1 + (a_2 - c) * x_2 + (a_3 + c) * x_3 + b, \quad (2.2)$$

$$Y = a_1 * x_1 + a_2 * x_2 + a_3 * x_3 + c * (x_3 - x_1 - x_2) + b. \quad (2.3)$$

Поскольку $x_3 - x_1 - x_2 = 0$, то, исключив эту составляющую из уравнения (2.3), мы вернемся к (2.1), то есть для прогнозирования Y можно использовать как уравнение (2.1), так и уравнение (2.2). Тогда возникают вопросы:

1. Коэффициенты какого из уравнений являются истинными?
2. Если коэффициенты могут быть произвольно большими по абсолютной величине, то как это скажется на точности счета?

К проблеме мультиколлинеарности мы еще вернемся в разд. 2.8, посвященном более сложным задачам регрессии, а пока возьмем за правило по возможности избавляться от сильно коррелирующих признаков и научимся это делать простыми средствами.

Вычисление коэффициентов корреляции в Excel

До сих пор мы решали только небольшие демонстрационные примеры. Настала пора оттачивать навыки решения задач машинного обучения на данных, максимально приближенных к реальным. Для этого надо научиться получать их из различных источников. Пока будем тренироваться на данных, предоставляемых очень популярным репозиторием UCI Machine Learning Repository. Для этого надо зайти в него и скачать заинтересовавший вас набор данных. В приведенном ниже примере нам необходим набор Energy Efficiency (Энергоэффективность). Мы не будем здесь подробно описывать содержание данных, так как оно есть в репозитории.

Исходные данные выглядят следующим образом (табл. 2.3).

Найдем коэффициенты корреляции всех признаков. Для этого выбираем в меню Данные/Анализ данных/Корреляция и указываем границы параметров. На отдельном листе получим следующее (табл. 2.4).

Таблица 2.3

Исходные данные задачи «Энергоэффективность зданий»

X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2	
0.98	514.50	294.00	110.25	7.00	2	0.00	0	15.55	21.33	X1 Относительная компактность
0.98	514.50	294.00	110.25	7.00	3	0.00	0	15.55	21.33	X2 Площадь поверхности
0.98	514.50	294.00	110.25	7.00	4	0.00	0	15.55	21.33	X3 Площадь стены
0.98	514.50	294.00	110.25	7.00	5	0.00	0	15.55	21.33	X4 Площадь крыши
0.90	563.50	318.50	122.50	7.00	2	0.00	0	20.84	28.28	X5 Общая высота
0.90	563.50	318.50	122.50	7.00	3	0.00	0	21.46	25.38	X6 Ориентация
0.90	563.50	318.50	122.50	7.00	4	0.00	0	20.71	25.16	X7 Площадь остекления
0.90	563.50	318.50	122.50	7.00	5	0.00	0	19.68	29.60	X8 Распределение площади остекления
0.86	588.00	294.00	147.00	7.00	2	0.00	0	19.50	27.30	y1 Нагревательная нагрузка
0.86	588.00	294.00	147.00	7.00	3	0.00	0	19.95	21.97	y2 Охлаждающая нагрузка
0.86	588.00	294.00	147.00	7.00	4	0.00	0	19.34	23.49	

Таблица 2.4

Коэффициенты корреляции исходных данных

A	B	C	D	E	F	G	H	I	J	K
Столбец 1	Столбец 1	Столбец 2	Столбец 3	Столбец 4	Столбец 5	Столбец 6	Столбец 7	Столбец 8	Столбец 9	Столбец 10
Столбец 1	1									
Столбец 2	-0.99190146	1								
Столбец 3	-0.20378168	0.195501633	1							
Столбец 4	-0.86882341	0.880719517	-0.29231647	1						
Столбец 5	0.827747317	-0.85814767	0.280975743	-0.97251224	1					
Столбец 6	0	0	0	0	0	1				
Столбец 7	1.28399E-17	1.31836E-16	-7.9697E-19	-1.3818E-16	1.86142E-18	0	1			
Столбец 8	1.76462E-17	-3.5586E-16	0	-1.0791E-16	0	0	0.212964221	1		
Столбец 9	0.622271936	-0.65811992	0.455671365	-0.86182805	0.889430464	-0.00258676	0.269841685	0.08736846	1	
Столбец 10	0.634339066	-0.67299893	0.427116998	-0.8625466	0.895785169	0.014289598	0.207504991	0.050525119	0.975861739	1

Из таблицы видно, что зависимость между входными параметрами 1, 2 и 4, 5 близка к линейной, поэтому исключаем 1 и 4. Зависимость между выходными параметрами 9, 10 близка к линейной, но так как нам необходимы уравнения регрессии для нагревательной и охлаждающей нагрузки, то оставляем оба выходных параметра. Строим уравнения регрессии описанным ранее способом для каждого выходного параметра и сравниваем результаты (табл. 2.5).

Два параметра были исключены, поэтому первоначальная и конечная нумерация параметров не совпадают.

Попутно заметим, что коэффициенты корреляции показывают: шестой входной признак очень слабо коррелируется с выходными признаками и, хотя это не создает особых вычислительных сложностей, он может быть исключен как неинформативный. Рекомендуем сделать это самостоятельно и построить новые уравнения регрессии.

Таблица 2.5

Коэффициенты уравнения регрессии

СРАВНЕНИЕ	Y2	Y1	первоначальные обозначения
Y-пересечение	-31.7154	-33.9909	X2 Площадь поверхности
Переменная X 1	0.023961	0.015384	X3 Площадь стены
Переменная X 2	0.018598	0.036945	X5 Общая высота
Переменная X 3	5.77062	5.530411	X6 Ориентация
Переменная X 4	0.12151	-0.02333	X7 Площадь остекления
Переменная X 5	14.71707	19.93274	X8 Распределение площади остекления
Переменная X 6	0.040697	0.203777	

2.3. Задачи нелинейной регрессии

Рассмотрим ситуацию, когда зависимость между входными и выходными параметрами носит явно нелинейный характер. Как всегда, начнем с простых примеров. Пусть, например, данные выглядят следующим образом (рис. 2.4).

Зависимость между входным и выходным параметрами носит явно нелинейный характер. В этом случае можно выдвинуть гипотезу, что данное множество точек можно аппроксимировать параболой. Значит, нам необходимо найти коэффициенты следующего уравнения регрессии:

$$Y = a_1 * x^2 + a_2 * x + b.$$

Заметим, что неизвестными в данном выражении являются a_1 , a_2 и b . Значит, относительно этих величин уравнение линейно. Поэтому если мы введем дополнительный параметр r^2 , то будем решать обычную задачу линейной регрессии, но в другом пространстве признаков — r и r^2 . Заметим, что для входных параметров мы сознательно используем обозначение r . Довольно часто для этого используют обозначение X . Математически нет никакой разницы, но, на наш взгляд, исходя из опыта преподавания, наша система обозначений удобней для понимания. Исходные данные дополняем новым признаком и решаем (рис. 2.5).

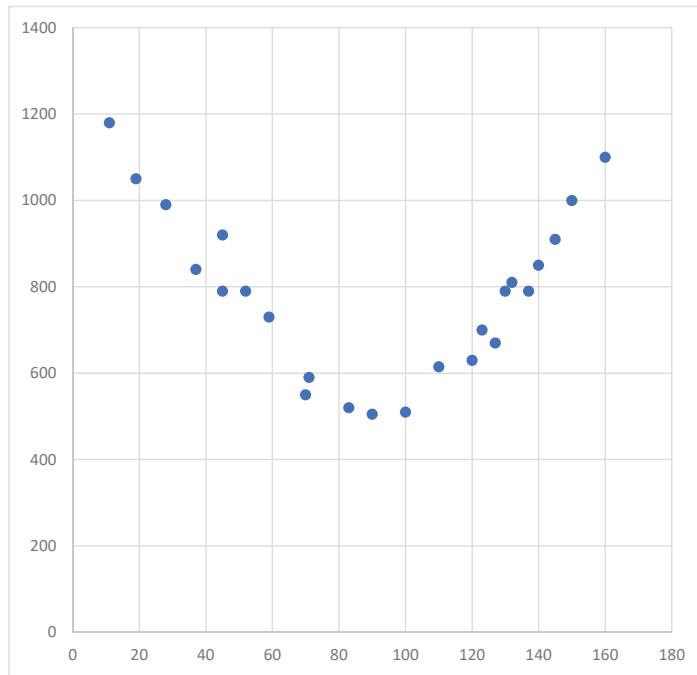


Рис. 2.4. Данные для нелинейной регрессии

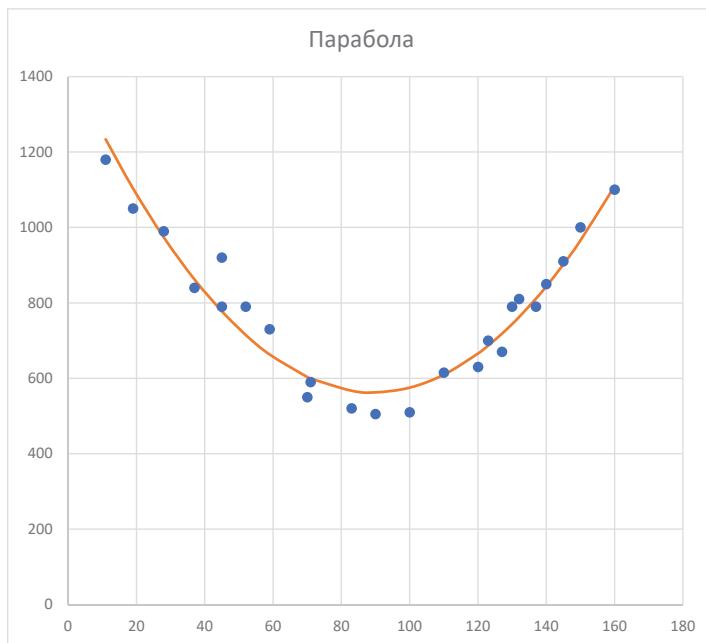


Рис. 2.5. Решение задачи нелинейной регрессии

Парабола является полиномом второй степени. Если множество точек необходимо аппроксимировать полиномами более высоких степеней, то поступаем абсолютно аналогичным образом, просто число входных параметров будет равняться степени полинома.

2.4. Применение линейной регрессии для моделирования кривых второго порядка

Рассмотрим другую, на первый взгляд, более сложную задачу (даные в прил. 1). Заданы координаты окружности (сечения трубы) — множество точек (x, y) . Требуется найти центр (c_1, c_2) и радиус r окружности. Для решения этой задачи можно применить инструментарий линейной регрессии. Заметим, что она не является задачей обучения с учителем — нахождения для объекта соответствующего ответа (в данной задаче (x, y) являются равнозначимыми признаками объекта). Эту задачу можно характеризовать как задачу обучения без учителя — нахождение зависимостей по данным.

Решим задачу определения параметров окружности. Уравнение окружности:

$$(x - c_1)^2 + (y - c_2)^2 = r^2.$$

Преобразуем:

$$2x*c_1 + 2y*c_2 + (r^2 - c_1^2 - c_2^2) = x^2 + y^2.$$

Переходим в новое пространство признаков и решаем (табл. 2.6).

У читателя может создаться впечатление, что приведен достаточно абстрактный пример и вряд ли такие задачи встретятся на практике. Это не так. Подобного рода задачи возникают при оценке состояния трубопроводов неразрушающими методами контроля, и их надо уметь решать.

Таблица 2.6

Определение центра и радиуса окружности

Входные параметры		Выходной				
2x	2y	x^2+y^2				
26	28	365	$r^2-c1^2-c2^2$	Y-пересечение	-572.672	
27	28.8038476	389.665409	c1	Переменная X 1	19.93197	
28	37.2111026	542.166538	c2	Переменная X 2	14.98586	
29	19.339746	303.756443				
30	37.797959	582.171426	r	радиус	7.013385	
31	17.2761947	314.8666726				
32	43.4891253	728.826005				
33	21.8756443	391.885954				
34	40.6491106	702.087549				
35	14.9233032	361.926244				
36	45.4164079	839.662526				
37	16.3252057	408.878085				
38	45.8564065	886.702503				
39	14.03576	429.500639				
40	42	841				
41	16.03576	484.536399				
42	41.8564065	878.98969				
43	20.3252057	565.528496				
44	43.4164079	955.246118				
45	16.9233032	577.849548				
46	40.6491106	942.087549				
47	15.8756443	615.259021				
48	43.4891253	1048.826				
49	21.2761947	713.419115				
50	41.797959	1061.76734				
51	23.339746	786.435935				
52	37.2111026	1022.16654				
53	24.8038476	856.057714				
54	32	985				

2.5. Немного философии

Теперь те, кто самостоятельно решал рекомендуемые задачи, уже имеют некоторые базовые навыки построения линейных и нелинейных уравнений регрессии и могут для конкретной задачи построить как те, так и другие. Конечно, когда мы видим рисунок на плоскости, то высказывать гипотезы о виде зависимости между входным и выходным параметром гораздо проще, чем когда входных признаков несколько, так как мы не умеем видеть в многомерном пространстве. Но даже на плоскости далеко не всегда очевидно, какая зависимость подходит лучше — например, какая зависимость лучше подходит для случая на рис. 2.6.

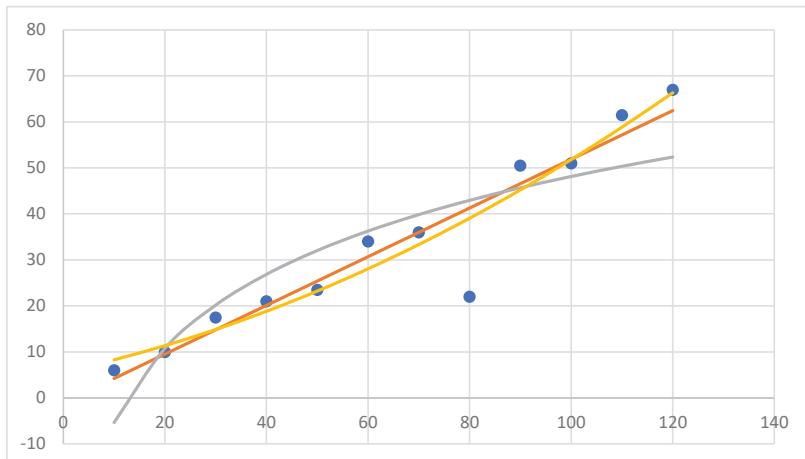


Рис. 2.6. Различные виды уравнений регрессии

Вообще есть два диалектически взаимосвязанных философских принципа, которые довольно часто применяют в машинном обучении. Это бритва Оккама и принцип Эпикура.

Принцип Эпикура: если более одной теории согласуются с наблюдениями, то следует оставить все теории.

Принцип бритвы Оккама в общем философском понимании: не следует множить сущее без необходимости (то есть чем проще, тем лучше).

Данные принципы применимы к различным задачам машинного обучения — от выбора конкретной модели до отбора признаков. С этой точки зрения в задаче анализа энергоэффективности зданий исключ-

чить входной признак, слабо коррелирующий с выходным, следовало, руководствуясь бритвой Оккама.

Однако даже если вы определились с видом зависимости, то существуют разные критерии качества решения. На наш взгляд, для пояснения сути различных критериев качества решения целесообразно представлять задачи машинного обучения как задачи математического программирования (ЗМП).

2.6. Общая постановка задачи математического программирования

Как считал великий математик Леонард Эйлер, если тщательно взглянуться в любую задачу, то можно обнаружить, на минимум она или на максимум. Действительно, жизнь устроена так, что всегда приходится добиваться некоторой цели, укладываясь при этом в некоторую систему ограничений. Если вам удалось формализовать функцию цели и систему ограничений, то вы уже сформулировали ЗМП. Если функция цели и система ограничений линейны, то это задача линейного программирования (ЗЛП), которая имеет очень эффективные алгоритмы решения задач большой размерности. Если это ЗЛП, но ряд переменных целочисленные, то это ЗЛП с частично целочисленными переменными. Причем если целочисленные переменные булевые, то такие задачи, естественно, решаются медленней, чем чистые ЗЛП, но тоже за приемлемое время. Наконец, если функция цели или система ограничений нелинейны, то это задачи нелинейного программирования. Данные задачи подразделяются на два класса: выпуклого программирования и невыпуклого. В классе задач выпуклого программирования есть задачи квадратичного программирования, которые тоже имеют эффективные алгоритмы решения. Так что, если вы практик и смогли сформулировать задачу как ЗЛП или ЗЛП с частично целочисленными переменными либо задачу квадратичного программирования, то вы можете воспользоваться алгоритмами их решения, даже не вникая в их суть. Данные алгоритмы создавались лучшими математиками мира начиная с середины 1940-х гг. Если у вас есть желание понять и суть самих алгоритмов, вы можете ознакомиться с ними отдельно

в рамках курсов по математическому программированию или самостоятельно читая труды классиков в этой области [22; 23]. Алгоритмы решения ЗМП оформлены в виде некоторых программ, которые называются *оптимизаторами* (CPLEX, MIP, PULP и др.). Для того чтобы ими воспользоваться, необходимо модель и данные привести к воспринимаемому ими виду.

2.7. Представление задач регрессии как ЗМП

Начнем с более формального определения: *математическое программирование* — это математическая дисциплина, в которой разрабатываются методы отыскания экстремальных значений целевой функции среди множества ее возможных значений, определяемых ограничениями.

Принципы математической записи задач регрессии как ЗМП и решение их в кодах IBM ILOG CPLEX достаточно подробно изложены нами в [21, с. 20–30]. Для начала работы с CPLEX вы можете взять клише программ из прил. 2 и потренироваться на данных условного примера. Решение задач регрессии в кодах Python с использованием пакета MIP с достаточно подробными комментариями приведены нами в прил. 10. Поэтому перейдем к рассмотрению практической задачи, которая на первый взгляд не выглядит как задача регрессии. На ее примере мы хотим показать взаимосвязь МО и исследования операций.

Взаимосвязь МО и исследования операций на примере задачи регрессии

Рассмотрим в несколько упрощенном виде реальную задачу, которую авторам довелось решать на практике (табл. 2.7).

Таблица 2.7

Вводные данные по группам оборудования

Вводные данные	Группа 1	Группа 2	Группа 3
Количество однотипных станков (шт.)	12	15	16
Максимальный фонд рабочего времени 1 станка за период (ч)	22	22	22
Общий фонд рабочего времени группы станков за период (ч)	264	330	352

По всем группам оборудования были избыточные мощности, поэтому хронометраж удельных затрат времени на производство каждого изделия не велся. Кроме того, заказы были эпизодическими, и вести специально хронометраж не имело смысла. Учитывались только объемы производства по каждому изделию и общие затраты времени по каждой группе оборудования. Сотрудниками отдела маркетинга и сбыта были сформулированы границы производства для потенциально большого заказа. Экономисты посчитали маржу по каждому изделию (табл. 2.8).

Таблица 2.8

Границы производства и маржа по каждому изделию

Границы/Маржа	Изд. 1	Изд. 2	Изд. 3	Изд. 4	Изд. 5
Минимум	0	100	0	0	0
Максимум	300	300	150	200	300
Маржа	5	2	10	7	3

Требовалось быстро составить план производства и сбыта — времени на хронометраж не было. Хотелось максимизировать суммарную маржу, но, так как штрафы за недопоставку были огромные, объемы производства необходимо было выбрать выполнимые. Кроме приведенной выше информации, в наличии была только информация по объемам производства изделий и общие затраты времени по группам оборудования в различные периоды времени (табл. 2.9).

Таблица 2.9

Объем производства изделий и затраты времени по группам*

Период	Объемы производства изделий (шт.)					Затраты времени (ч)		
	Изд. 1	Изд. 2	Изд. 3	Изд. 4	Изд. 5	Группа 1	Группа 2	Группа 3
1	50	8	82	118	73	140	139	192
2	20	84	6	20	50	58	142	81
3	84	84	106	89	100	176	274	248
4	99	29	71	5	65	107	160	179
5	91	115	103	89	82	167	301	265
6	100	73	88	115	35	132	218	281
...
94	12	73	76	61	45	99	175	105

*Полностью данные приведены в прил. 3.

В математическом программировании существует большое количество классических задач. Одна из них — это составление оптимального плана производства. Сформулируем ее сразу в кодах CPLEX (через математические символы для тренировки запишите самостоятельно):

```
/* план производства */
int g=...; //число групп оборудования
range j = 1..g;// индекс группы оборудования
int p=...; // число видов продукции
range i = 1..p; // индекс вида продукции
float U [j] [i] =...; // удельные затраты времени при производстве i-го вида
продукции на j-ом оборудовании
float F [j] =...; // фонды времени
float M [i] =...; // маржа i-го вида продукции
int NG [i] =...; // нижняя граница производства i-го вида продукции
int VG [i] =...; // верхняя граница производства i-го вида продукции
/* искомые переменные */
dvar int+ x [i]; // выпуск i-го вида продукции
dvar float+ summarg; // суммарная маржа
maximize summarg;
subject to {
sum (i in i) M [i] *x [i] ==summarg;
forall (j in j) sum (i in i) U [j] [i] *x [i] <=F [j];
forall (i in i) x [i] <=VG [i];
forall (i in i) x [i] >=NG [i];
};
```

Давайте проанализируем, каких данных нам не хватает, чтобы решить данную задачу. Очевидно, что это $U [j] [i] = \dots$; — то есть удельные затраты. Теперь подумайте, каким образом мы можем их найти? Ответ простой: необходимо решить три уравнения регрессии в Excel или как 3МП. Все решения приведены в прил. 3.

Естественно, реальная задача была намного сложнее. Для раздумья сформулируем несколько вопросов, касающихся рассмотренной задачи:

1. Как Вы проинтерпретируете свободные члены в задачах регрессии?
2. Можно ли в задачах регрессии избавиться от свободных членов?
3. Что вы будете делать, если опытный технолог, посмотрев результаты по удельным затратам, скажет, что у ряда изделий они явно нереальные и должны быть в определенных границах?

2.8. Задачи регрессии повышенной сложности

Описание задач регрессии повышенной сложности приведены нами в [21, с. 30–35]. Всеми примерами, приведенными в [21], мы хотим показать, что если вы научитесь представлять задачи регрессии как ЗМП, то перед вами откроются широчайшие возможности по решению задач, которые нельзя решить стандартными способами.

Вообще сведение задач регрессии к ЗМП позволяет шире взглянуть на сами задачи регрессии. Например, в ряде случаев интересно выяснить, каким количеством уравнений регрессии можно описать обучающую выборку с заданной точностью, то есть сколько неравенств потребуется, чтобы число наблюдений, не удовлетворяющих ни одному неравенству, было меньше некоторой пороговой величины. Такая постановка может быть описана следующей моделью:

$$\min \sum_{j \in J} \omega_j, \quad (2.4)$$

$$(1-d)^* y_j - L^* z'_j \leq \sum_{i \in I} p_{ij}^* a'_i + b' \leq (1+d)^* y_j + L^* z'_j \quad t \in T, j \in J, \quad (2.5)$$

$$\sum_{t \in T} z'_j \leq m - 1 + \omega_j \quad j \in J, \quad (2.6)$$

где m — количество уравнений регрессии; t — индекс уравнения регрессии $t = 1, \dots, m$; z'_j — булева переменная (признак непопадания в диапазон t -го уравнения); ω_j — булева переменная (признак непопадания в диапазон всех уравнений).

Отметим, что кусочно-линейная регрессия является частным случаем данной модели. Модель вида (2.4)–(2.6) позволяет выделять практически любые закономерности между входными и выходными параметрами, от описываемых параллельными гиперплоскостями до описываемых гиперплоскостями, пересекающимися крест-накрест. Такие ситуации не являются чем-то экзотическим, а скорее всего будут свидетельствовать о том, что не учтены какие-то существенные входные параметры.

В кодах CPLEX данная модель выглядит следующим образом:

```
/* несколько уравнений регрессии */
float d=0.1;//допустимый диапазон (доля от выходного параметра)
```

```
int m=...;//число линий
range t = 1..m;//индекс линии
int L=10000;
int k=...;//число элементов в множестве
range j = 1..k;//индекс элемента 1-го множества
int n=...;//число переменных (параметры наблюдений)
range i = 1..n;//индекс параметра
float P [j] [i] =...;//множество входных параметров объектов
float Y [j] =...;//множество выходных параметров объектов
dvarfloat a [i] [t];//коэффициенты t-й линии регрессии
dvarfloat b [t];//свободный член t-й линии регрессии
dvarboolean z [j] [t];//признак непопадания в диапазон t-й линии регрессии
dvarboolean w [j];//признак непопадания ни в один из диапазонов
minimizesum (j in j) w [j];//сумма непопаданий ни в один диапазон
subject to {
forall (t in t) forall (j in j) sum (i in i) P [j] [i] *a [i] [t] +b [t] <= (1+d) *Y [j] +L*z [j] [t];
forall (t in t) forall (j in j) sum (i in i) P [j] [i] *a [i] [t] +b [t] >= (1-d) *Y [j]-L*z [j] [t];
forall (jin j) sum (t in t) z [j] [t] <= (m-1) + w [j];//нормально, если наблюдение попало хотя бы в один диапазон
};
```

Попробуйте приведенную выше модель на данных листа «Кусочно-линейная» прил. 1.

Согласитесь, что по двухмерным проекциям (рис. 2.7) достаточно сложно сказать, какой вид регрессии соответствует этим данным. Попробуйте порешать данную задачу при различных значениях d , и вы увидите, что, начиная с определенного диапазона, число выбросов стабилизируется на уровне шести и останется таковым до $d = 0.016$. При таком диапазоне данные соответствуют двум уравнениям регрессии:

$$Y = -3X_1 + 4X_2 + 100 \text{ и } Y = 5X_1 + X_2 + 9$$

$$\text{или } Y = -2,83X_1 + 3,96X_2 + 89,4 \text{ и } Y = 5.18X_1 + 1.01X_2 + 5.12.$$

Дальнейшее снижение диапазона будет приводить к росту числа наблюдений, не попадающих в него, но уравнения останутся прежними. Поэтому остановимся на этих уравнениях. Естественно, возникает вопрос: каким из них прогнозировать выходной показатель для конкретного тестового наблюдения? Вопрос интересный и непростой,

поэтому предлагаем читателям самостоятельно рассмотреть различные варианты ответов.

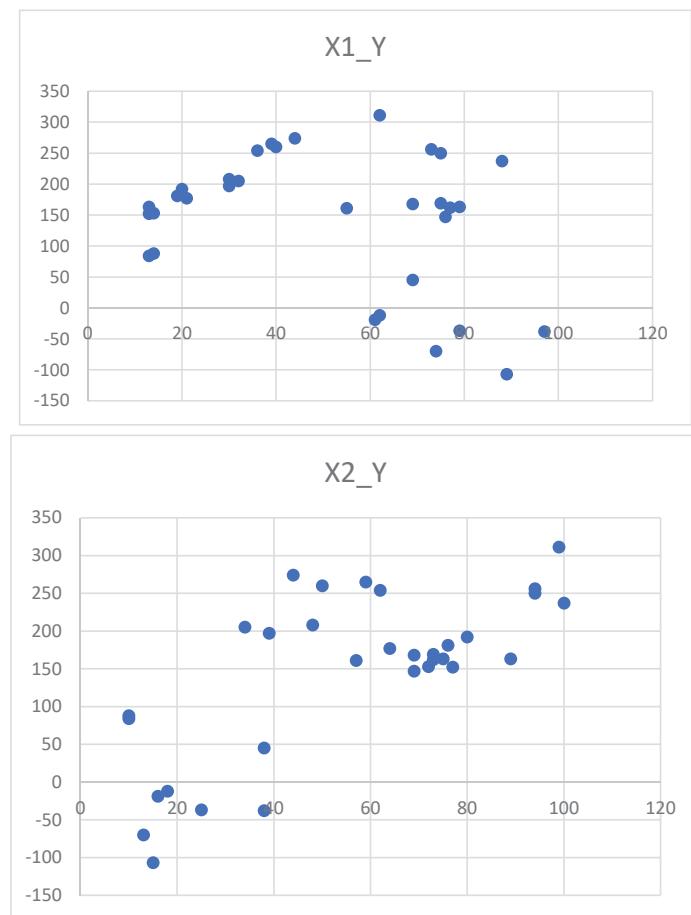


Рис. 2.7. Двухмерные проекции для кусочно-линейной аппроксимации

Глава 3.

ЗАДАЧИ КЛАССИФИКАЦИИ

3.1. Линейно разделимые множества

Имея представление об уравнении прямой и умея решать задачи регрессии, можно переходить к изучению методов классификации. Скажем сразу, что методов классификации достаточно много — мы рассмотрим основные из них. Но для сохранения логики изложения начнем с линейно разделимых множеств. Перед рассмотрением определения данного понятия попробуем для начала решить конкретную задачу.

Пусть в двухмерном пространстве (то есть на плоскости) заданы два множества MP1 и MP2, содержащие координаты некоторых точек. Пример множеств в виде, удобном для использования в IBM ILOG CPLEX, можно найти в прил. 4 на листе «Множества» в столбце «Данные для CPLEX» в строках под номером 1.

Графически множества MP1 и MP2 представлены на рис. 3.1.

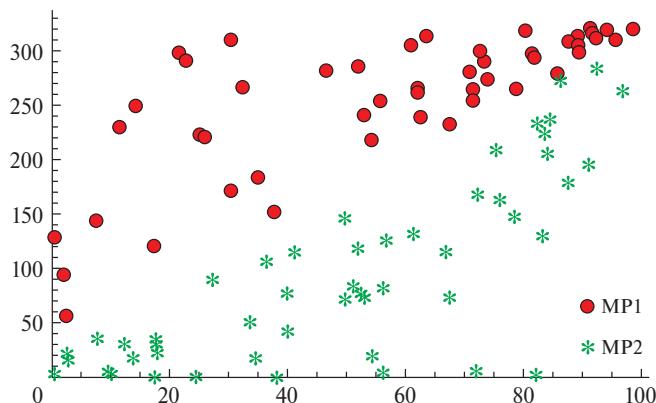


Рис. 3.1. Пример линейно разделимых множеств

В дальнейшем под *линейно разделимыми множествами* мы будем понимать два множества, между которыми в двухмерном пространстве можно провести линию, в трехмерном пространстве — плоскость, в n -мерном пространстве — гиперплоскость, такие, что точки разных множеств будут расположены по разные стороны линии, плоскости или гиперплоскости.

На рис. 3.1 видно, что такую линию провести можно. Если это сделать вручную, то получим графическое решение задачи. Попробуем вычислить формулу линии формализованно, а не при помощи линейки и карандаша.

Для удобства математической записи определимся с используемыми символами:

J_1 и J_2 — множества, которые необходимо разделить;

$J = J_1 \cup J_2$ — множество наблюдений;

I — множество параметров наблюдений;

i, j — индексы соответствующих множеств;

p_{ij} — i -й параметр j -го наблюдения (константы);

a_i — коэффициенты гиперплоскости (искомые переменные);

b — свободный член гиперплоскости (искомая переменная);

L — очень большое число;

ε — очень малое число (используем для строгости ограничений);

z_j — булевые переменные для фиксации нарушений условий разделения множеств.

В принципе, для решения поставленной задачи достаточно найти допустимое решение системы неравенств:

$$\sum_{i \in I} p_{ij} * a_i + b < 0 \quad j \in J_1, \quad (3.1)$$

$$\sum_{i \in I} p_{ij} * a_i + b > 0 \quad j \in J_2, \quad (3.2)$$

где p_{ij} — параметры множеств (числа); a_i — коэффициенты разделяющей линии (в общем случае — гиперплоскости), переменные значения, которые надо определить; b — свободный член разделяющей линии, значение которого тоже надо определить. На практике строгие неравенства используются редко, но если их записать как нестрогие в виде

$$\sum_{i \in I} p_{ij} * a_i + b \leq 0 \quad j \in J_1, \quad (3.3)$$

$$\sum_{i \in I} p_{ij} * a_i + b \geq 0 \quad j \in J_2, \quad (3.4)$$

то допустимое решение очевидно (все переменные равны нулю), но совершенно бесполезно. Поэтому неравенства обычно записывают в виде

$$\sum_{i \in I} p_{ij} * a_i + b \leq -\varepsilon \quad j \in J_1, \quad (3.5)$$

$$\sum_{i \in I} p_{ij} * a_i + b \geq \varepsilon \quad j \in J_2. \quad (3.6)$$

Возможным также является подход, когда одна система неравенств задается строго, а другая — нет, например можно использовать (3.3) и (3.6) или (3.4) и (3.5). Форма записи системы неравенств определяется требованиями, которые предъявляет исследователь к желаемому результату.

В нашем примере $i = 2$, множества J_1, J_2 содержат по 50 элементов. Множества могут быть линейно не разделимы. В этом случае система ограничений будет противоречивой и допустимое решение будет отсутствовать. Поэтому правильнее будет записать систему ограничений в следующем виде:

$$\sum_{i \in I} p_{ij} * a_i + b \leq w_j \quad j \in J_1, \quad (3.7)$$

$$\sum_{i \in I} p_{ij} * a_i + b \geq -w_j + \varepsilon \quad j \in J_2, \quad (3.8)$$

$$w_j \geq 0 \quad j \in J, \quad (3.9)$$

где w_j — это возможные корректировки отдельных неравенств.

Вполне естественным является стремление избавиться от таких корректировок. Поэтому добавим целевую функцию C , например:

$$\min \sum_{j \in J} w_j. \quad (3.10)$$

Если множества линейно разделимы, то $C = 0$, иначе $C > 0$.

Если $C > 0$, то исследователь может продолжить работу с корректировками, например потребовать, чтобы максимальная из них была

минимальной (критерий минимакса). Для этого необходимо ввести условия: $w_j \leq u \quad j \in J$, а целевую функцию записать в виде $\min u$.

Поскольку в данном примере нас интересуют не величины w_j , а сам факт — выполняется условие разделения множеств или нет, то правильным будет ввести следующие условия:

$$w_j \leq L * z_j \quad j \in J, \quad (3.11)$$

где z_j — булева переменная, принимающая значение 0, если условие разделения множеств выполняется, и 1 в противоположном случае. Таким образом, переменная z_j будет фиксировать факт нарушения условия разделения. Соответственно, целевая функция будет

$$\min \sum_{j \in J} z_j. \quad (3.12)$$

Данную модель на языке IBM ILOG CPLEX можно найти в файле Excel на листе «Модели» в столбце «Модель линейного разделения множеств 1».

В более компактном виде модель может быть представлена как

$$\sum_{i \in I} p_{ij} * a_i + b - L * z_j \leq 0 \quad j \in J_1, \quad (3.13)$$

$$\sum_{i \in I} p_{ij} * a_i + b + L * z_j \geq \varepsilon \quad j \in J_2, \quad (3.14)$$

$$\min \sum_{j \in J} z_j. \quad (3.15)$$

Данную модель на языке IBM ILOG CPLEX можно найти в файле Excel на листе «Модели» в столбце «Модель линейного разделения множеств 2».

Форма записи практически совпадает с приведенной математической моделью. Для удобства анализа результатов решения введены условия суммирования z_j по каждому множеству. Кроме того, введены следующие условия:

```
a [n] ==g-q; g+q==1;
dvar boolean g [t];//техническая переменная
dvar boolean q [t];//техническая переменная
```

Благодаря этим условиям одна из переменных $a[n]$ принимает значения -1 или 1 . Пока будем считать, что сделано это чисто для удобства, так как обычно запись прямой $y = a*x + b$ воспринимается и интерпретируется лучше, чем $a_1*x - a_2*y + B = 0$, хотя по сути это одно и то же.

Теперь, чтобы получить результаты расчетов, необходимо загрузить данные по множествам MP1 и MP2 в файл данных, файлы модели и данных внести в Конфигурацию запуска и запустить на счет эту конфигурацию.

Результаты расчетов:

$$sz_1 = 0; sz_2 = 0,$$

то есть множества линейно разделимы:

$$a_1 = -2.6123; a_2 = 1; b = -48.999,$$

смысл каждой переменной дается в комментариях к модели.

Значит, линией, разграничающей множества, может быть

$$y = 2.6123*x + 48.999.$$

Нами не случайно использован оборот «может быть», так как обычно, если множества линейно разделимы, то линий таких можно провести бесконечно много, но в определенном диапазоне. Неоднозначность решения надо воспринимать не как угрозу, а как возможность дальнейшего исследования и поиска наиболее красивых с точки зрения исследователя решений. Например, можно ввести ограничение: коэффициенты должны быть целыми. Для этого в модели необходимо заменить `dvar float a[i];` на `dvar int a [i];`. Тогда решение будет $a_1 = -3; a_2 = 1; b = -19$.

Допустим, что мы остановились на этом решающем правиле. В результате новых наблюдений мы имеем новую точку $[79.2, 250.7]$. Подставляем данные значения в уравнение гиперплоскости: $3*79.2 + 1*250.7 - 19 = -5.9$, значит, новая точка лежит ниже и поэтому относится к множеству MP1.

Теперь рассмотрим задачу, где число параметров более двух, например четыре (файл Excel, лист «Множества» № 2). Естественно, представлять такую задачу графически бессмысленно. Однако для получения решения достаточно изменить в модели условие `int n = 4`, ввести

новые данные в файл данных, сформировать новую Конфигурацию и запустить ее.

В результате получим

$$sz_1 = 0; sz_2 = 0,$$

то есть множества линейно разделимы;

$$a_1 = -3.2596; a_2 = 1.0941; a_3 = -6.418; a_4 = 1; b = 0.$$

Если добавим условие целочисленности $a[i]$ и b , то

$$sz_1 = 0; sz_2 = 0,$$

то есть множества линейно разделимы:

$$a_1 = -2; a_2 = 2; a_3 = -6; a_4 = 1; b = -140.$$

Таким образом, формула разделяющей гиперплоскости может быть

$$x_4 = 2*x_1 - 2*x_2 + 6*x_3 - 140.$$

Имея данное решающее правило, определим, к какому множеству принадлежит, например, точка [49.1, 42.5, 98.8, 659.2]:

$$-2*49.1 + 2*42.5 - 6*98.8 + 1*659.2 - 140 = -86.8.$$

Значит, точка лежит ниже гиперплоскости и относится к множеству MP1.

Проверим полученное решающее правило для всех точек MP1 и MP2. Для этого перенесем исходные данные из IBM ILOG CPLEX в таблицу Excel, добавим уравнение гиперплоскости и проанализируем полученные результаты (файл Excel, листы «Проверка линейной разделимости», «Проверка комитет единогласия», «Проверка комитет большинства» и «Проверка комитет старшинства»). В колонке «Результаты решающего правила» у всех точек множества MP1 результат отрицательный, а у точек множества MP2 положительный, это и есть решающее правило.

Таким образом, суть подхода не изменяется в зависимости от размерности пространства признаков. Поэтому все дальнейшие пояснения будем давать для случая двух признаков и дополнять модели геометрическими интерпретациями.

Попробуем дать не геометрическую, а другую, содержательную, интерпретацию. Представим, что существует некий уникум, способный видеть, как расположены точки в пространстве любой размерности. Тогда его мнения, к какому из множеств отнести любую точку, будет достаточно. Данный случай можно трактовать как существование комитета, состоящего из одного члена, единолично принимающего решения. Параллели в обществе — диктатор, никому не доверяющий собственник и т. п. Их роль в данном случае выполняет гиперплоскость.

Отметим также, что если исследуется только вопрос линейной разделимости множеств, то для одного из множеств система неравенств может быть задана без корректировок, то есть $w_1[j_1]$ или $w_2[j_2]$ и связанные с ними переменные в модели будут отсутствовать. В дальнейшем мы будем поступать таким образом при построении комитета старшинства итерационным способом, а пока рекомендуем видоизменить и посчитать предыдущие примеры этим способом самостоятельно.

3.2. Линейно не разделимые множества и метод комитетов

Как правило, в реальной жизни приходится работать в условиях линейно не разделимых множеств, когда в n -мерном пространстве между множествами нельзя провести гиперплоскость. Чтобы проиллюстрировать данный случай, сделаем небольшое изменение в множестве MP2. Пусть последним элементом множества вместо [85.916, 278.119] будет [85.916, 27.119]. Вполне реальная ситуация: например, при вводе данных была сделана ошибка. В этом случае множества становятся линейно не разделимыми. Поэтому решением будет

$$sz_1 = 0; sz_2 = 1; z_2 [50] = 1,$$

то есть множества линейно не разделимы, но мы тем не менее имеем коэффициенты разграничитывающей линии, при которой число таких нарушений минимально: $x_1 = -3; x_2 = 1; b = -16$.

Далее уже исследователь должен разобраться, что это: ошибка ввода данных, сбой аппаратуры при исследовании, случайный выброс и т. п. Естественно, такие выводы можно делать только в том случае, когда

подобных нарушений немного, и исследователь считает, что данными отклонениями можно пренебречь. Рассмотрим два других множества — MP1 и MP2 (файл Excel, лист «Множества» № 3).

Попробуем разделить множества линейно, то есть в модели меняем только

```
int K1=42;//число элементов во множестве 1
int K2=108;//число элементов во множестве 2
```

и используем новый файл данных. Получаем

$$sz_1 = 2; sz_2 = 1; a_1 = 2.586; a_2 = 1; b = -188.9.$$

Множества линейно не разделимы. Графически это показано на рис. 3.2.

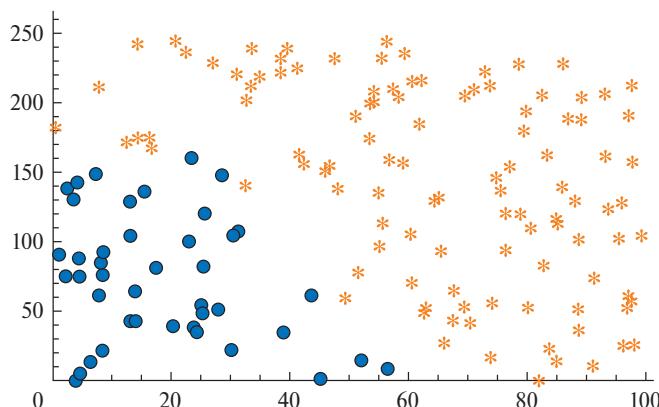


Рис. 3.2. Пример линейно не разделимых множеств

Пока не будем рассматривать возможность разделения множеств нелинейными гиперповерхностями, так как такие решения могут быть излишне сложными. Естественно, если нельзя разделить одной гиперплоскостью, то, вероятно, можно разделить несколькими. Такой метод называется *методом комитетов*.

Метод комитетов относится к методам классификации с учителем, применяемым для решения проблемы отнесения того или иного объекта к одному из двух классов объектов. Наличие учителя обозначает, что предварительно существует некоторая обучающая выборка, на которой нам известно, какой объект к какому классу относится. Сама логика работы метода напоминает логику работу обычного комитета

как коллегиального руководящего органа, где итоговое решение принимается на основании решений нескольких экспертов. В случае метода комитетов роль экспертов исполняют несколько разделяющих гиперплоскостей, называемых членами комитета, каждая из которых «голосует» за решение по-своему. Итоговое решение принимает комитет, в целом учитывая решения всех отдельных членов за счет использования логики комитетов.

Для лучшего понимания указанного определения вернемся к рис. 3.2. Как уже было сказано, одной линией разграничить множества действительно нельзя. Если бы мы использовали алгоритм линейного разделения, то получили бы результат с тремя ошибками, как показано на рис. 3.3.

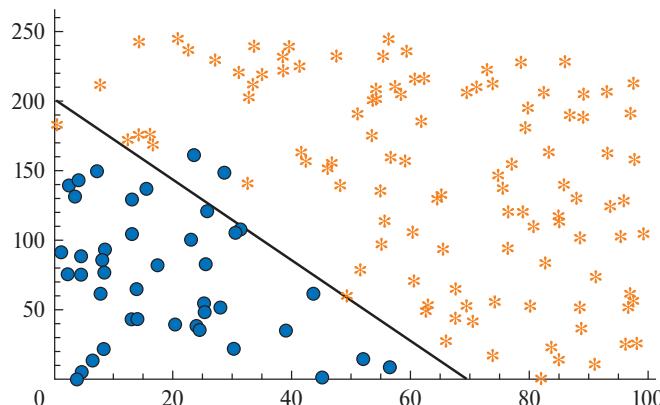


Рис. 3.3. Линейное разделение линейно неразделимых множеств

Однако возможно разделение и без ошибок. Для этого только и нужно, что воспользоваться двумя линейными классификаторами. Для начала сделаем такое разграничение с помощью карандаша и линейки (рис. 3.4). Обратим внимание, что все точки одного из множеств лежат ниже каждой из линий.

Если опять провести параллели с обществом, то данный случай можно интерпретировать как наличие комитета из двух членов. Очевидно, что если существует несколько членов комитета, то решение будет однозначным только в случае заранее четко прописанной процедуры голосования. Опять же, на примере общества мы знаем, что голоса их могут быть равнозначными и неравнозначными, то есть существуют старший и младший члены комитета. Если их голоса равнозначны, то

решение может быть принято, например, в случае их единогласного голосования за или против. Поэтому данный комитет принято называть *комитетом единогласия* (далее по тексту — КЕ). В реальной жизни примером такого комитета может служить принцип парламентского устройства в Речи Посполитой, который позволял любому депутату сейма прекратить обсуждение вопроса и работу сейма вообще, выступив против. Данный принцип действовал более 200 лет. Из современных примеров можно привести процедуру принятия решения по не-процедурным вопросам в Совете Безопасности ООН. Решение здесь требует также единогласия со стороны постоянных членов Совета Безопасности.

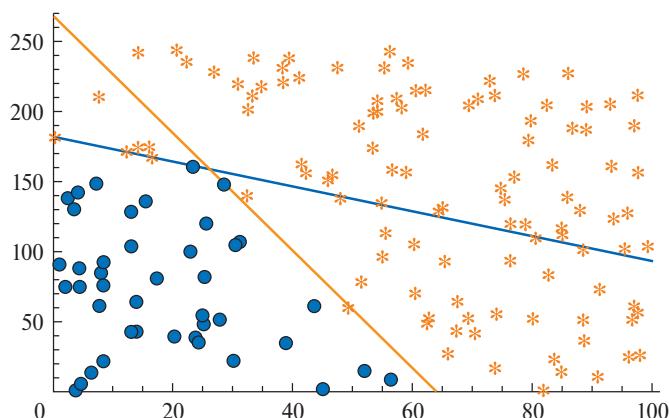


Рис. 3.4. Разделение линейно не разделимых множеств композицией линейных разделителей

В данном случае решающее правило будет звучать следующим образом: точка относится к множеству МР1, если оба члена комитета проголосовали за это одинаково. Геометрически это означает, что точка лежит ниже каждой из линий (гиперплоскостей).

Достаточно часто при равнозначности голосов разделить множества при помощи КЕ не представляется возможным. В таком случае вместо единогласного решения можно嘅 пытаться найти «коллектив наблюдений» — такой, что за удовлетворение каждого ограничения будет голосовать большинство членов комитета. Он будет называться *комитетом большинства* (КБ). Самый известный пример КБ из реальной жизни — это выборы президента РФ. Если рассматривать выборы в терминологии метода комитетов, то данная задача представляет собой классификацию множества кандидатов в президенты на множе-

ства «президент» и «непрезидент». Соответственно, в данной терминологии каждый избиратель предстает как отдельный член комитета, голосующий за того или иного кандидата на основании своего жизненного опыта и известных ему параметров кандидатов.

В случае, когда голоса неравнозначны, процедура может быть следующей. Если старший член проголосовал за, решение принято. Если старший член воздержался, то он делегировал принятие решения младшему члену. Аналог ситуации — когда шеф, не желая вникать в мелкий для него вопрос, говорит подчиненному: «Примите решение самостоятельно». Такой комитет называется *комитетом старшинства* (далее по тексту — КС), и именно с него мы начнем рассмотрение комитетных конструкций. Такой выбор сделан по очень простой причине: на основе навыков по линейному разделению множеств читатель легко может применить итерационные алгоритмы построения КС, а мы ему в этом поможем.

3.3. Разновидности комитетных конструкций

Итерационный алгоритм разделения на основе комитета старшинства (КС)

Если внимательно посмотреть на предыдущий рисунок, то естественным образом возникает желание, используя гиперплоскости, отсечь элементы одного множества от другого. Такая операция может быть представлена IBM ILOG CPLEX в виде программного кода — как в файле Excel на листе «Модели» в столбце «Модель построения КС итерационным способом».

Вся последовательность действий с иллюстрациями подобно изложена нами в [21, с. 52–63]. Для лучшего понимания сути КС рекомендуем повторить ее самостоятельно. Очевидно, что такой процесс всегда сходится за конечное число шагов, то есть КС существует всегда. Отметим, что итерационный алгоритм не гарантирует, что КС будет состоять из минимального числа членов. Это несколько другая задача.

Естественным образом возникает вопрос: обязательно ли осуществлять действия последовательно и можно ли построить математические модели, где члены комитета ищутся одновременно в рамках одной модели? Ответ смотрите далее.

Модель комитета единогласия

Рассмотрение вопроса о построении единой модели комитета мы начнем с примера для КЕ, построенного на рисунках, приведенных в предыдущем разделе. Дополним множество обозначений:

- T — множество членов комитета (гиперплоскостей);
- t — индекс множества T ;
- p_{ij} — параметры наблюдения j ;
- a_i^t — коэффициенты гиперплоскостей (искомые переменные);
- b^t — свободные члены гиперплоскостей (искомые величины);
- z_j^t — булевы переменные для фиксации нарушений условий разделения множеств;
- h — число членов комитета (гиперплоскостей, нейронов).

Напомним, что в случае КЕ за элементы одного из множеств все члены комитета голосуют единогласно, а против элементов другого множества выступает хотя бы один член комитета. Поэтому по аналогии с итерационным методом построения КС на начальном этапе для одного из множеств условия зададим жестко и строго. В символьном виде это можно записать как КЕ с жесткими условиями для одного из множеств:

$$\sum_{i \in I} p_{ij} * a_i^t + b^t \leq -\varepsilon \quad j \in J_1, t \in T, \quad (3.16)$$

$$\sum_{i \in I} p_{ij} * a_i^t + b^t + L * z_j^t \geq \varepsilon \quad j \in J_2, t \in T, \quad (3.17)$$

$$\sum_{t \in T} z_j^t \leq 0 \quad j \in J_1, \quad (3.18)$$

$$\sum_{t \in T} z_j^t \leq h - 1 \quad j \in J_2, \quad (3.19)$$

$$\min \sum_{t \in T} \sum_{j \in J_2} z_j^t. \quad (3.20)$$

В кодах IBM ILOG CPLEX данную модель можно записать, как в файле Excel на листе «Модели» в столбце «Модель КЕ с жестким условием для одного из множеств» [условие (3.18) можно не записывать, так как выполнение (3.19) автоматически означает выполнение (3.18)].

Решение:

- первый член комитета: $a_1 = 0.88895$; $a_2 = 1$; $b = -182.16$;
- второй член комитета: $a_1 = 5.5526$; $a_2 = 1$; $b = -322.93$.

В общем случае нам неизвестно, для какого из множеств условия должны быть записаны без корректировок. Попробуем построить модель, где такой выбор осуществляется автоматически в виде КЕ с выбором множества с жесткими условиями:

$$\sum_{i \in I} p_{ij} * a_i^t + b^t - L * z_j^t \leq -\varepsilon \quad j \in J_1, t \in T, \quad (3.21)$$

$$\sum_{i \in I} p_{ij} * a_i^t + b^t + L * z_j^t \geq \varepsilon \quad j \in J_2, t \in T, \quad (3.22)$$

$$\sum_{t \in T} z_j^t \leq (h-1) * Z \quad j \in J_1, \quad (3.23)$$

$$\sum_{t \in T} z_j^t \leq (h-1) * (1-Z) \quad j \in J_2, \quad (3.24)$$

$$\min \sum_{t \in T} \sum_{j \in J} z_j^t, \quad (3.25)$$

где Z — булева переменная для выбора множества (0 — условия единогласия выполняются для множества J_1 ; 1 — условия единогласия выполняются для множества J_2).

Отметим, что, начиная с этого примера, мы не будем специально оговаривать условие строгости для ограничений, полагая, что в случае необходимости добавить малые величины в правые части ограничений не составит труда в файле Excel на листе «Модели» в столбце «Модель выбора множества условия без корректировок».

Решение:

- первый член комитета: $a_1 = 5.5518$; $a_2 = 1$; $b = -322.89$;
- второй член комитета: $a_1 = 0.90838$; $a_2 = 1$; $b = -183.88$.

Некоторое различие в коэффициентах по сравнению с предыдущим решением не должно смущать исследователя, так как мы уже поясняли, что это не угроза, а возможность, и в дальнейшем на таких различиях останавливаться не будем. Проверку результатов можно просмотреть в файле Excel на листе «Модели» в столбце «Модель выбора множества условия без корректировок».

Итоги голосования можно интерпретировать следующим образом:

1. Для точек множества МР1 оба члена комитета не возражали за отнесение их к множеству МР1.
2. Для точек множества МР2 хотя бы один член комитета возражал за отнесение их к множеству МР2.

Естественно, для разных множеств число членов КЕ (если он существует) будет различным. Рекомендуем для тренировки построить КЕ для множеств, указанных в файле Excel, лист «Множества» № 6 (MP1 — 15 элементов; MP2 — 135 элементов).

Лучше для практики попробовать оба подхода, так как автоматический выбор множества более комфортен для пользователя, но более расточителен по числу переменных, а значит, и по времени счета на больших задачах.

Один из вариантов решения:

- первый член комитета: $a_1 = 0,58386$; $a_2 = 1$; $b = -183,04$;
- второй член комитета: $a_1 = -1,4087$; $a_2 = -1$; $b = 157,38$;
- третий член комитета: $a_1 = 2,6802$; $a_2 = 1$; $b = -280,37$.

Далее рассмотрим различные виды КЕ и предложим читателю самостоятельно свести их к виду, удобному для вычислений, и найти решающие правила. Для примера будем давать одно из них. Все графические изображения комитетов будут содержать стрелки, показывающие, в какую сторону голосует каждый член комитета. В каждом случае голосование будет проходить за выбор множества кружков. В файле Excel на листе «Множества» под № 7 представлены множества MP1 (8 элементов) и MP2 (92 элемента).

Указанные множества можно разделить КЕ из следующих трех членов (рис. 3.5):

$$a_2 = 200 - 7a_1; \quad a_2 = 47 + 5a_1; \quad a_2 = 10 + 17a_3.$$

В файле Excel на листе «Множества» № 8 представлены множества MP1 (6 элементов) и MP2 (94 элемента).

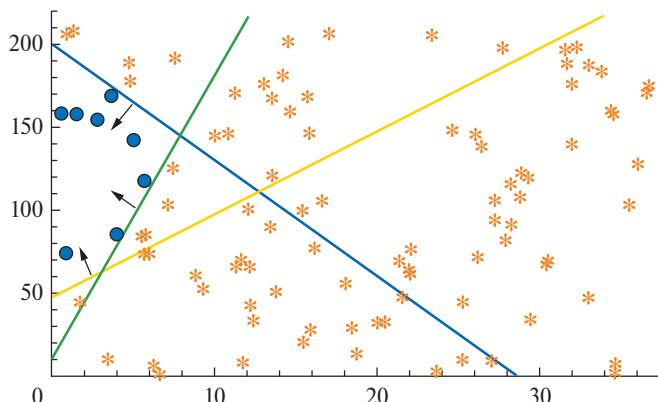


Рис. 3.5. Первый пример разделения множеств КЕ

Указанные множества можно разделить КЕ из следующих трех членов (рис. 3.6):

$$y_1 = 300 - 7a; y_2 = 67 + 5a; y_3 = 10 + 17a.$$

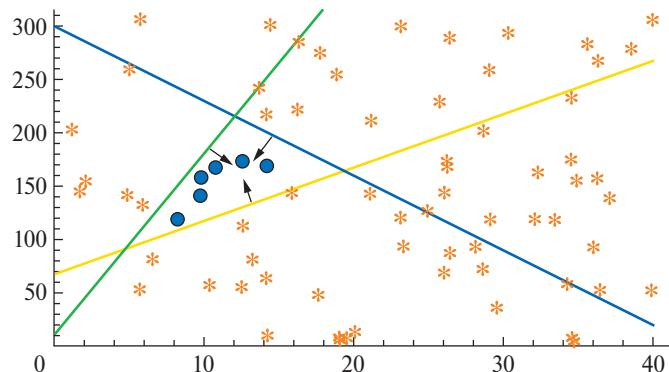


Рис. 3.6. Второй пример разделения множеств КЕ

Модель комитета большинства

Достаточно часто разделить множества КЕ не представляется возможным (пример: файл Excel, лист «Множества» под № 9). Как это может выглядеть графически, показано на рис. 3.7.

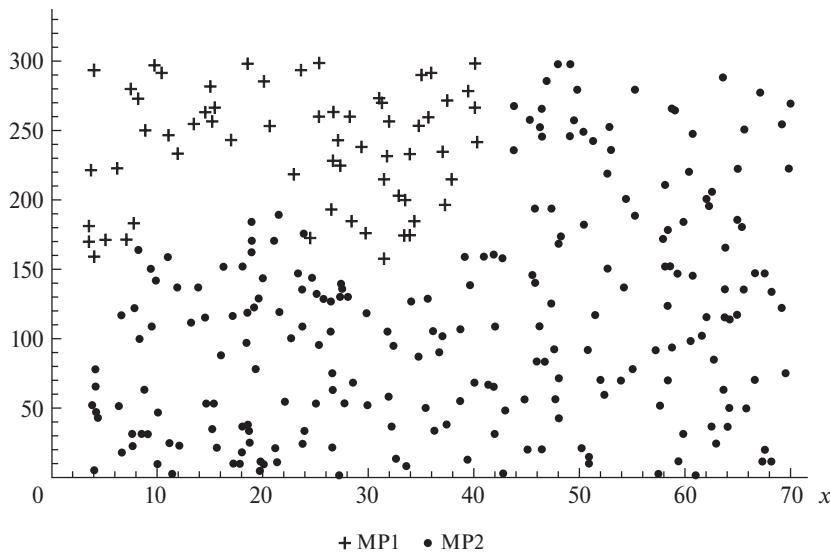


Рис. 3.7. Пример множеств, не разделимых КЕ

Очевидно, что линейно или КЕ эти множества не разграничить. Но для практики рекомендуем попробовать. Воспользуемся опять идеологией комитетного подхода. Пусть голоса членов комитета равнозначны, а количество членов (гиперплоскостей) нечетно. Необходимо найти такой набор гиперплоскостей, чтобы для каждой точки множества MP1 простое большинство (больше половины) из них голосовало за, а для каждой точки множества MP2 — против.

Математически это можно записать как КБ, заменив в модели КЕ с выбором множества с жесткими условиями ограничения (3.23) и (3.24) предыдущей главы на (3.26) и (3.27) ниже, соответственно:

$$\sum_{t \in T} z_j^t \leq m \quad j \in J_1, \quad (3.26)$$

$$\sum_{t \in T} z_j^t \leq h - m - 1 \quad j \in J_2, \quad (3.27)$$

где m — меньшинство (задаваемая константа), h — количество членов комитета.

В виде, удобном для вычислений, для комитета простого большинства (за большинство принимается более половины членов комитета) это будет выглядеть как в файле Excel на листе «Модели» в столбце «Модель КБ 3 члена».

Решение:

- 1-й член комитета: $a_1 = 3.36119; a_2 = -1; b = 136.92;$
- 2-й член комитета: $a_1 = 11.2154; a_2 = -1; b = -223.1;$
- 3-й член комитета: $a_1 = -5.49795; a_2 = -1; b = 308.5.$

В данном примере мы задали меньшинство как константу $m = 1$. Проведем небольшой эксперимент. Уберите это условие, определите m как переменную (dvar int m) и минимизируйте ее, то есть функция цели будет minimize m;.

Отправив задачу на счет, вы убедитесь, что решения почти совпадают.

Решение:

- 1-й член комитета: $a_1 = 3.36119; a_2 = -1; b = 136.92;$
- 2-й член комитета: $a_1 = 11.08; a_2 = -1; b = -217.39;$
- 3-й член комитета: $a_1 = -9.9142; a_2 = -1; b = 413.483.$

Значит, квалифицированное меньшинство может быть вычисляемо. Для случая трех членов это не имеет смысла, а, например, при 9 членах оно может быть как 5, так и 7. Эти рассуждения понадобятся нам в дальнейшем.

Наверное, глядя на эту модель, хочется сказать: «Да, это, скорее, комитет меньшинства». Поэтому поясним, что приводимый нами способ подсчета голосов и форма записи условий комитета не являются единственными возможными. По сути, мы математически записали, что если член комитета согласен, то он воздерживается ($z_j^t = 0$), а если не согласен, то голосует против ($z_j^t = 1$). Иногда кажется, что более естественным будет сделать наоборот. Это легко достигается введением дополнительного условия $\alpha'_j = 1 - z_j^t \quad j \in J$, а условия комитета принимают вид

$$\sum_{t \in T} \alpha'_j \geq \beta \quad j \in J_1, \quad (3.28)$$

$$\sum_{t \in T} \alpha'_j \leq \beta \quad j \in J_2, \quad (3.29)$$

где β — большинство.

Конечно, можно решать задачу и в таком виде. Более того, можно даже согласиться, что так она более интерпретируема, но проигрывает по числу переменных, а значит, и по времени счета на задачах большой размерности. Кроме того, если вы сделаете подстановку в условия (3.28) и (3.29), то получите условия (3.26) и (3.27).

Отметим также, что таким же естественным является подход, когда за — это 1, а против —1. Достигается это тоже довольно легко, введением дополнительного условия $h_j^t = 1 - 2 * z_j^t \quad j \in J$. Как записать условия комитетов, предлагаем сообразить самостоятельно. Мы не оговорились, именно «комитетов», то есть и для единогласия, и всех последующих. Поэтому каждый должен определиться, какая форма записи ему ближе, и в дальнейшем использовать ее.

Для самостоятельной работы приведем различные примеры комитетов большинства.

Посмотрим множества MP1 (68 элементов) и MP2 (100 элементов) в файле Excel на листе «Множества» под № 10.

Данные множества можно разделить комитетом большинства, состоящим из трех членов: $Y1 = 5 + 59a$; $Y2 = 241 - 9a$; $Y3 = 12a$ (рис. 3.8).

Наблюдательный читатель заметит, что все коэффициенты целые. Измените программу в кодах CPLEX так, чтобы получить именно это решение.

Следующий пример в файле Excel на листе «Множества» под № 11 содержит множества MP1 (23 элемента) и MP2 (100 элементов). Данные

множества можно разделить КБ из трех членов: $Y_1 = 200 - 13a$; $Y_2 = 47 + 5a$; $Y_3 = -100 + 17a$ (рис. 3.9).

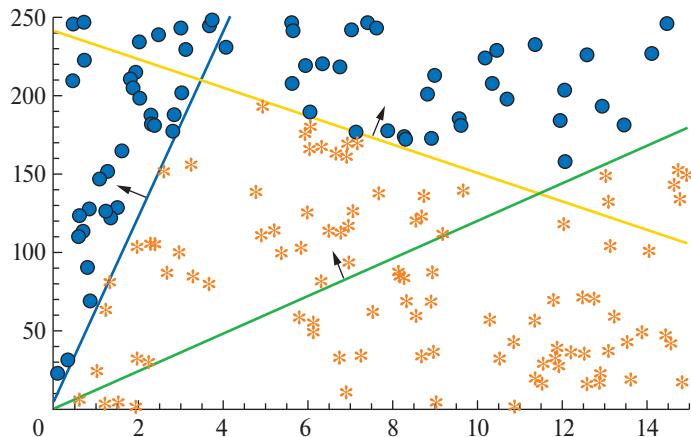


Рис. 3.8. Первый пример разделения множеств КБ

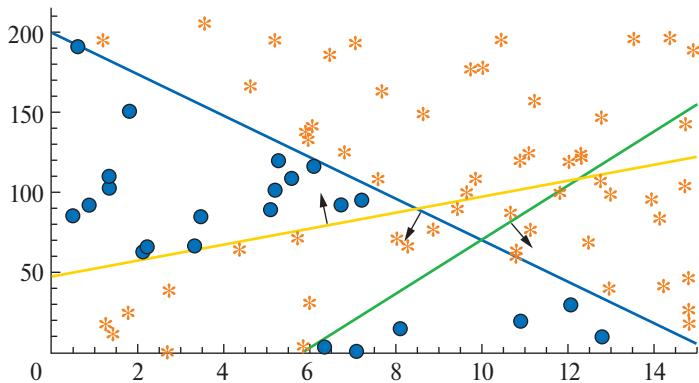


Рис. 3.9. Второй пример разделения множеств КБ

В файле Excel на листе «Множества» под № 12 представлен дополнительный пример, содержащий множества MP1 (83 элемента) и MP2 (100 элементов). Данные множества можно разделить комитетом большинства, состоящим из пяти членов: $Y_1 = 5 + 59a$; $Y_2 = 57 + 2a$; $Y_3 = -7 + 23a$; $Y_4 = 241 - 9a$; $Y_5 = 12a$ (рис. 3.10).

Надеемся, пытливый читатель заметит, что в данном примере одна из линий лишняя, исключит ее и проинтерпретирует это, используя идеологию метода комитетов.

Ответ: это КБ из четырех членов с квалифицированным большинством 3.

Далее рассмотрим еще пример, содержащий множества МР1 (118 элементов) и МР2 (100 элементов), соответствующие множествам из файла Excel, лист «Множества» № 13.

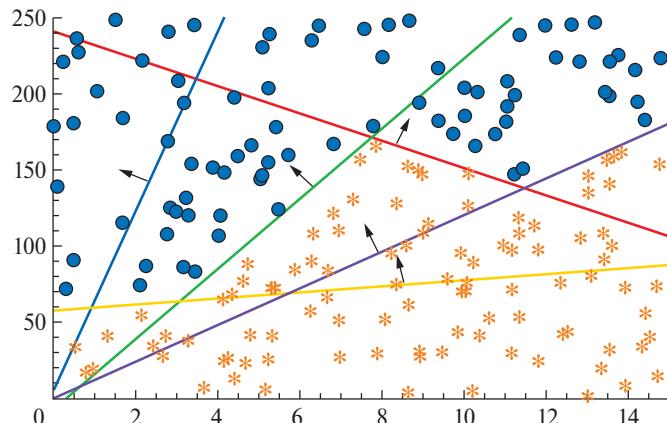


Рис. 3.10. Третий пример разделения множеств КБ

Данные множества можно разделить КБ, состоящим из пяти членов:

$$y_1 = 5 + 59a; y_2 = 97 + 2a; y_3 = -7 + 23a; y_4 = 241 - 9a; y_5 = 12a \text{ (рис. 3.11).}$$

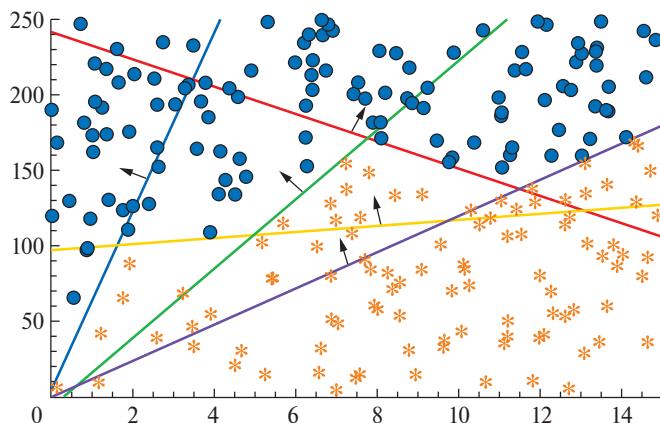


Рис. 3.11. Четвертый пример разделения множеств КБ

Рисунок 3.11 очень похож на предыдущий: те же линии, но точки расположены чуть по-другому. Соответственно, всплывают следующие вопросы:

1. В данном случае достаточно четырех членов или нет?
2. Какие точки выпадут из рассмотрения при четырех членах?

Модель комитета старшинства

До сих пор при разделении множеств гиперплоскостями и интерпретации их в качестве членов комитета, голосующих за или против, мы исходили из того, что голоса всех членов комитета имеют одинаковый вес. Для иллюстрации подхода и сравнительного анализа воспользуемся множествами из примеров, которые мы использовали при описании итерационного подхода.

Отметим, что, следуя данному правилу, после получения ответа «да» на любом шаге можно принимать решение. Это напоминает комитет, где вес голоса старшего члена больше всех остальных вместе взятых. Если он воздержался, то решение принимает следующий по старшинству и т. д. Поэтому такие комитеты и называются комитетами старшинства (КС).

Для составления КС с фиксированными весами достаточно в модели преобразовать (3.26) и (3.27) из предыдущего раздела в (3.30) и (3.31) соответственно, как показано ниже, и поменять целевую функцию:

$$\sum_{t \in T} z_j^t * V^t \leq m \quad j \in J_1, \quad (3.30)$$

$$\sum_{t \in T} (z_j^t * V') \leq \sum_{t \in T} V^t - m - 1 \quad j \in J_2, \quad (3.31)$$

$$\min m, \quad (3.32)$$

где V^t – веса членов комитета (заранее задаваемые величины по принципу «вес старшего больше суммы всех остальных», для этого удобно использовать степени 2 или 10, таким образом можно оцифровать все сектора); m – ограничитель по весу (переменная).

Попробуем разделить КС множества, используемые нами при построении разделения итерационным методом (рис. 3.12). Для этого сначала присвоим каждому члену комитета веса, равные степеням 2, то есть 4, 2 и 1. Модель и данные можно найти в файле Excel соответственно на листе «Модели» в столбце «Модель КС 3 члена» и листе «Множества» в столбце «Данные для CPLEX» в строках под № 14.

Кроме того, попробуем ужесточить условия, требуя целочисленности:

```
dvar int a [i] [t];//коэффициенты гиперплоскостей
dvar int b [t];//свободные члены гиперплоскостей
```

Другими словами, проверим, есть ли целочисленные решения. В этом случае решением будет $y_1 = 251 - 3a$, $V = 4$; $y_2 = -51 + 15a$, $V = 2$; $y_3 = 122 + a$, $V = 1$ (рис. 3.12).

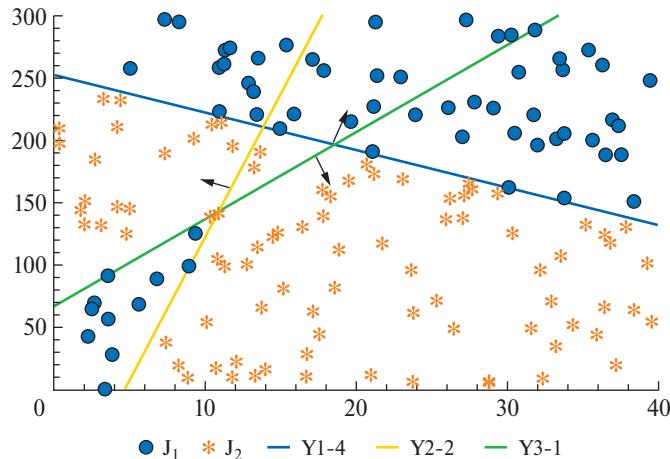


Рис. 3.12. Первый пример разделения множеств КС

Чтобы понять, как указанное решение классифицирует объекты, на рис. 3.12 последовательно представлено присвоение весов от старшего члена комитета к младшему. Исходя из графика 4 на рис. 3.12, можно говорить, что объект принадлежит к кружкам, если область, в которой он находится, имеет вес не более 4, иначе он принадлежит к звездочкам. В конечном виде последовательность шагов приведена на рис. 3.13.

Теперь, как нами было обещано, дадим пояснения, почему удобно, чтобы одна из переменных принимала значения 1 или -1 . Запишем уравнения гиперплоскостей из рис. 3.12 немного в другом виде:

$$-3*a_1 - 1*a_2 + 251 = 0; 15*a_1 - 1*a_2 - 51 = 0; -1*a_1 + 1*a_2 - 122 = 0.$$

Именно так наиболее удобно анализировать результаты расчетов в таблицах Excel и определять направление голосования. Гиперплоскости с -1 при p_2 голосуют в одну сторону, с 1 — в другую. Тогда относительно множества MP1: y_1 и y_2 — «вверх»; y_3 — «вниз». Относительно множества MP2 — наоборот.

Сравним найденное решение с решением, полученным итерационным способом. Все они правильно разделяют множества, только при решении в рамках одной задачи понадобилось членов комитета на один меньше. Таким образом, мы видим, что итерационный метод

выигрывает по размерности задачи (число переменных и ограничений), а значит, и по времени счета, но проигрывает по возможности находить минимальный комитет. Минимальный комитет, конечно, не надо фетишизировать и на практике совсем не обязательно использовать именно его, но возможность его находить открывает дополнительные возможности при анализе данных. Иначе говоря, если существует комитет из трех членов в данной задаче, то он будет найден.

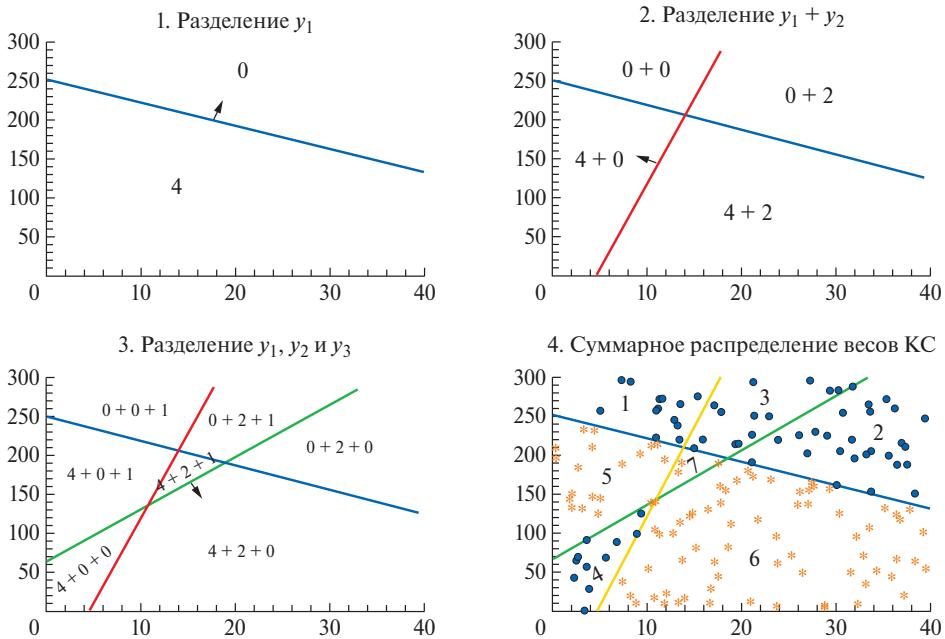


Рис. 3.13. Логика работы КС для примера с рис. 3.12

Мы использовали оборот «если существует». А что же делать, если не существует? Ответ дадим чуть позже, а пока предлагаем читателям самостоятельно поэкспериментировать с данными предыдущей задачи — попытаться сделать задачу противоречивой, понять, как в этом случае ведет себя модель, и самим ее видоизменить. Кроме того, рекомендуем решить следующие задачи.

Приведем различные примеры КС. В файле Excel на листе «Множества» под № 15 представлены множества MP1 (180 элементов) и MP2 (100 элементов).

Данные множества можно разделить КС, состоящим из четырех членов при $m = 10$: 1) $a_2 = 101 + 13a_1$, $V = 8$; 2) $a_2 = 57 + 7a_1$, $V = 4$; 3) $a_2 = 13 + 5a_1$, $V = 1$; 4) $a_2 = -51 + 11a_1$, $V = 2$ (рис. 3.14).

Все результаты для каждого наблюдения представлены в файле Excel, лист «Результаты КС 4 членов».

Рассмотрим множества MP1 (384 элемента) и MP2 (100 элементов) в файле Excel, лист «Множества» под № 16. Данные множества можно разделить КС, состоящим из пяти членов при $m = 28$: 1) $a_2 = -10a_1 + 266$, $V = 16$; 2) $a_2 = 13a_1 + 101$, $V = 8$; 3) $a_2 = 7a_1 + 57$, $V = 2$; 4) $a_2 = 5a_1 + 13$, $V = 1$; 5) $a_2 = 14a_1 - 74$, $V = 4$ (рис. 3.15).

Все результаты для каждого наблюдения представлены в файле Excel, лист «Результаты КС 5 членов».

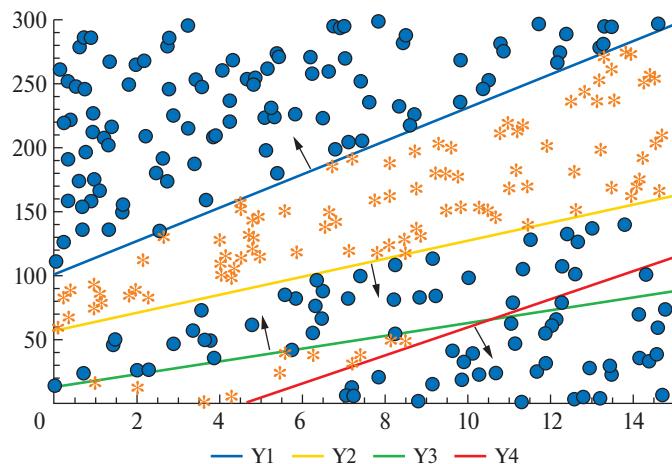


Рис. 3.14. Второй пример разделения множеств КС

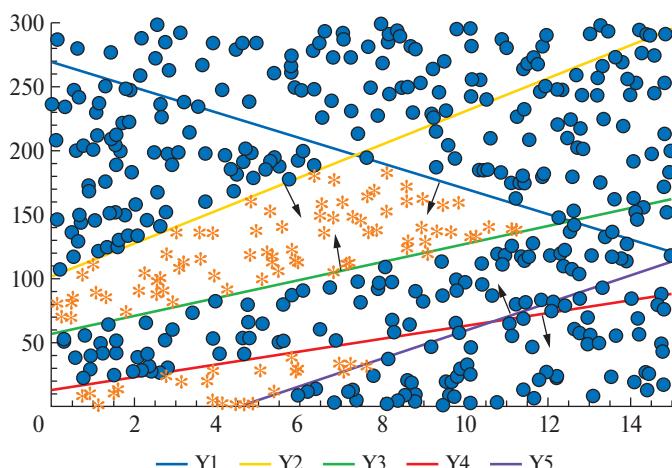


Рис. 3.15. Третий пример разделения множеств КС

Чуть выше, до приведения примеров, мы обещали рассмотреть случай, когда комитет из заданного числа членов не существует, то есть задача противоречива, и вместо конкретного решения вы получите ответ «значения отсутствуют». Означает ли это, что ситуация тупиковая? Конечно, нет. Просто условия для комитета надо записать в нежестком виде и потребовать минимизации их нарушений. Составим такую модель КС с фиксированными весами и корректировкой условий комитета в случае противоречивости условий задачи, преобразовав в предыдущей модели (3.30)–(3.32) в (3.33)–(3.35) соответственно:

$$\sum_{t \in T} z_j^t * V^t \leq m + L * E_j \quad j \in J_1, \quad (3.33)$$

$$\sum_{t \in T} (z_j^t * V^t) \leq \sum_{t \in T} V^t - m - 1 + L * E_j \quad j \in J_2, \quad (3.34)$$

$$\min \sum_{j \in J} E_j, \quad (3.35)$$

где V^t — веса членов комитета (заранее задаваемые величины по принципу «вес старшего больше суммы всех остальных», удобно использовать степени 2); m — меньшинство (переменная); E_j — булевы переменные для фиксации нарушений классификации.

Решая задачу в таком виде, в случае непротиворечивости условий вы получаете допустимое решение, в случае противоречивости — оптимальную коррекцию условий. А далее выбор за вами:

- согласиться с коррекцией, если нарушений немного;
- увеличить число членов комитета;
- попробовать модель с автоматическим подбором весов.

Дополнительно анализируя модель КС, можно прийти к интересному выводу: указанная модель также подходит для КЕ и КБ. Для доказательства обозначенного вывода сначала вспомним, какие ограничения используются в каждой из моделей комитета (табл. 3.1).

Рассмотрим КС при $V^t = 1 \forall t \in T$:

- если $m = 0$ или $m = h - 1$, то (3.30) и (3.31) могут быть соответственно преобразованы в (3.18) и (3.19), что соответствует КЕ;
- если $0 < m < (h - 1)$, то (3.30) и (3.31) могут быть соответственно преобразованы в (3.26) и (3.27), что соответствует КБ.

Таблица 3.1

Ограничения в моделях комитетов

Логика	$j \in J_1$	$j \in J_2$
КЕ	$\sum_{t \in T} z_j^t \leq 0 \quad (3.18)$	$\sum_{t \in T} z_j^t \leq h - 1 \quad (3.19)$
КБ	$\sum_{t \in T} z_j^t \leq m \quad (3.26)$	$\sum_{t \in T} z_j^t \leq h - m - 1 \quad (3.27)$
КС	$\sum_{t \in T} (z_j^t * V^t) \leq m \quad (3.30)$	$\sum_{t \in T} (z_j^t * V^t) \leq \sum_{t \in T} V^t - m - 1 \quad (3.31)$

Рассмотрим КС при $V^t = 2^t$:

- если $m = 0$ или $m = sv - 1$, то (3.30) и (3.31) могут быть соответственно преобразованы в (3.18) и (3.19), что соответствует КЕ;
- если $0 < m < (sv - 1)$, то (3.30) и (3.31) не могут быть преобразованы к другому виду — значит, КС.

В прил. 4 будут приведены клише программ в кодах IBM ILOG CPLEX. Для лучшего усвоения материала можно воспользоваться схемой соответствия модели КС (без корректировок условий комитета) ее программному коду [21, с. 75–76].

3.4. Разделение множеств нелинейными функциями

Ранее нами были рассмотрены примеры разделения множеств линейными функциями, однако в жизни можно встретить примеры, для которых эффективное разделение достигается при применении нелинейных функций. В качестве примера допустим, что в результате исследований были получены два множества, указанные в файле Excel на листе «Множества» под № 19 (рис. 3.16).

Множества линейно не разделимы. Исследователь хочет проверить гипотезу, что множества разделимы функцией $y = a_1 * x^2 + a_2 * x + b$. Необходимо найти: a_1 , a_2 , b .

В принципе, в данном подходе нет ничего нового для читателя. Сведение нелинейных зависимостей к линейным мы подробно разбира-

рали, когда рассматривали задачи регрессии. В файле Excel на листе «Множества» под № 20 содержатся исходные множества, преобразованные к виду, удобному для вычислений в формате IBM ILOG CPLEX, с добавлением параметра, равного квадрату первого.

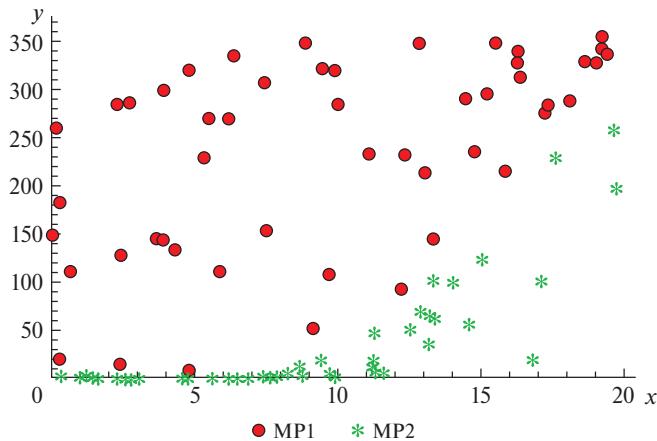


Рис. 3.16. Пример множества, разделимого нелинейной функцией

Теперь для получения решения достаточно указать в модели линейного разделения множеств $n = 3$ и создать конфигурацию с новыми данными.

Решение:

$$sz_1 = 0; sz_2 = 0,$$

значит, множества разделимы полиномом степени 2,

$$a_1 = 1.0754; a_2 = -6.5327; a_3 = -1; b = 8.7221,$$

то есть разделяющий полином имеет вид

$$y = 1.0754 * x^2 - 6.5327 * x + 8.7221.$$

Попробуем усложнить решение. Например, потребуем целочисленности а и б. Решение получим более красивое:

$$y = x^2 - 6 * x + 9.$$

Но $sz_1 = 0; sz_2 = 1$, значит, точного разделения не произошло, и исследователь должен решить, может он этим пренебречь или нет.

Отметим, что аналогичным образом можно осуществлять разделение множеств полиномами более высоких степеней и любыми сепарабельными функциями. Только надо понимать, что вид функции (или функций) должен задаваться исследователем, а коэффициенты перед ними определяются в ходе решения.

Хотелось бы также отметить, что работа с нелинейными функциями требует определенного опыта, а попытки линеаризовать несепарабельные функции часто заканчиваются абсурдными результатами.

Глава 4.

КЛАССИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ

4.1. Метод опорных векторов

Мы достаточно подробно рассмотрели метод комитетов, который позволяет строить решающие правила даже в случае отсутствия линейной разделимости множеств. Достаточно большое число авторов внесли свою лепту в развитие и практическое применение данного подхода. На наш взгляд, наиболее системно данный подход развивался в математической школе И. И. Еремина и В. Д. Мазурова [24].

Выше мы уже отмечали, что в случае линейной разделимости множеств не все так просто: если множества линейно разделимы, то в общем случае число гиперплоскостей, их разделяющих, равно бесконечности. И возникает закономерный вопрос: какое решение лучше? Начнем, как всегда, с простого примера на плоскости. Пусть мы имеем следующие множества (табл. 4.1, см. также прил. 5). Графически это выглядит так, как показано на рис. 4.1.

Очевидно, что между синими и оранжевыми точками существует некоторый зазор, через который можно провести большое количество гиперплоскостей. Тогда лучшей гиперплоскостью, разделяющей множества, можно считать, например, равноудаленную от каждого. Алгоритм построения такой гиперплоскости был предложен В. Вапником и А. Червоненкисом [25] в 1963 г. По сути, они свели данную задачу к задаче квадратичного программирования и решили ее на основе теоремы Куна — Таккера. В последствии в 1992 г. В. Вапник, Б. Босер и И. Гийон предложили способ создания нелинейного классификатора на основе перехода от скалярных произведений к произвольным ядрам, то есть разделение линейным классификатором в другом пространстве признаков. Вам этот прием уже известен, мы рассматривали его при решении задач нелинейной регрессии и разделении классов нелинейными гиперповерхностями.

Таблица 4.1

Данные для иллюстрации метода опорных векторов

ОБУЧАЮЩЕЕ 1		ОБУЧАЮЩЕЕ 2		ТЕСТОВОЕ		
p1	p2	p1	p2	p1	p2	
1.8	11.2	1	5	1.5	12	S1
3	20	1.5	2	1.8	4.9	S2
7	21	2	2.3	4.1	14	S1
2	18	2.5	3	5.5	8	S2
6.5	22	4	7	7.7	11	S2
4	17	5	5			
5	14	5.5	10.1			
6	18.5	6	10			
2	17	7	13			
1.5	20	8	15			
3	12					
4.5	21					
3.5	22					
2.5	21					

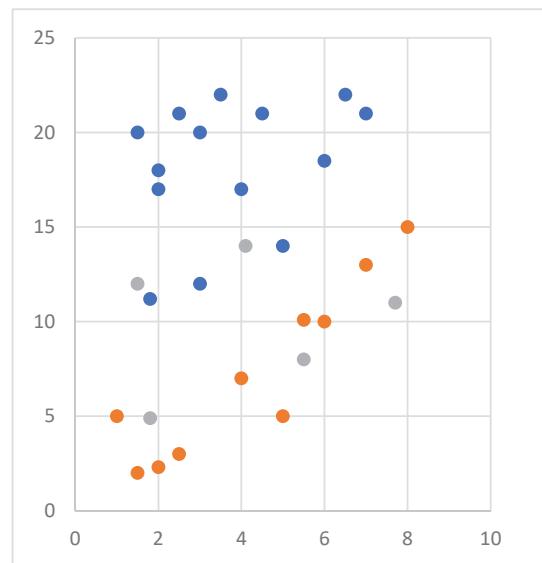


Рис. 4.1. Обучающие и тестовое множества для метода опорных векторов

Описанный выше подход называется *метод опорных векторов*, по-английски Support Vector Machine (SVM). Графически это выглядит следующим образом (рис. 4.2).

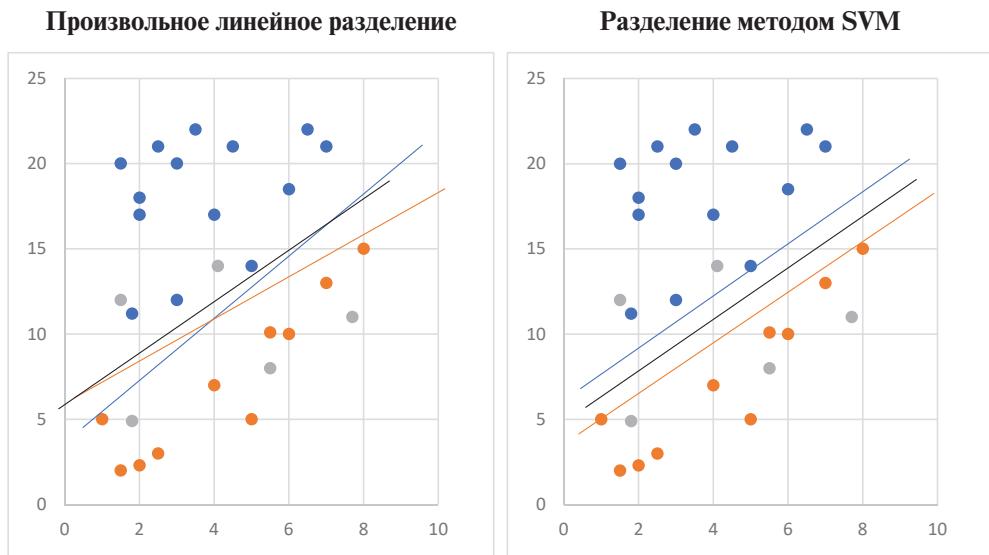


Рис. 4.2. Разделяющие гиперплоскости
(произвольные и по методу SVM)

При разделении методом SVM разделяющая гиперплоскость (черная линия) пройдет ровно посередине между синей и красной линиями.

Программы для решения задач машинного обучения данным методом есть в различных языках высокого уровня и программных комплексах. Мы воспользуемся Wolfram Mathematica, в которой для решения задач машинного обучения есть очень удобный оператор `Classify`.

Давайте создадим или скопируем из прил. 5 таблицу Excel с названием `data_for_SVM.xlsx` и каждое множество разместим на отдельном листе без заголовков, то есть только данные. Данную таблицу мы сохраним в директории `D:\ML` и запустим следующую программу:

```
(* считываем обучающие и тестовое множества *)
set1=Import ["D:\\ML\\data_for_examples.xlsx", {"Data", 1}];
Print ["Training set 1 = ", Length [set1]];
set2 = Import ["D:\\ML\\data_for_examples.xlsx", {"Data", 2}];
Print ["Training set 2 = ", Length [set2]];
test= Import ["D:\\ML\\data_for_examples.xlsx", {"Data", 3}];
```

```
Print ["Test set = ", Length [test]];
(* ищем указанным методом решающее правило *)
decisionrule=Classify [<|"S1"->set1,"S2"->set2|>, Method->
"SupportVectorMachine", PerformanceGoal->"Quality"];
(* применяем решающее правило *)
decisionrule [set1]
decisionrule [set2]
decisionrule [test]
```

Мы специально показываем, как взять определенные данные, присвоить им определенные метки (S1 и S2), выбрать метод решения "SupportVectorMachine", целевую функцию "Quality", записать решающее правило в decisionrule и применить его к обучающим и тестовому множествам. Результаты решения будут следующими:

```
Training set 1 = 14
Training set 2 = 10
Test set = 5
{S1, S1, S1}
{S2, S2, S2, S2, S2, S2, S2, S2, S2}
{S1, S2, S1, S2, S2}
```

Если в качестве разметки множеств будет написано

```
decisionrule=Classify [<|"1"->set1,"0"->set2|>, Method->
"SupportVectorMachine", PerformanceGoal->"Quality"];
```

то и разметка множеств будет соответствующей:

```
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
{1,0,1,0,0}
```

В данном месте, зная, что конкретным специалистам не терпится поскорее опробовать все возможные методы для анализа их набора данных, и несколько забегая вперед, приведем вначале программу, которая позволяет это сделать, а потом поясним суть каждого из методов.

```
(* считываем обучающие множества *)
set1=Import ["D:\\ML\\data_for_examples.xlsx", {"Data", 1}];
```

```

Print ["Training set 1 = ", Length [set1]];
set2 = Import ["D:\\ML\\data_for_examples.xlsx", {"Data", 2}];
Print ["Training set 2 = ", Length [set2]];
test= Import ["D:\\ML\\data_for_examples.xlsx", {"Data", 3}];
Print ["Test set = ", Length [test]];
methods= {"SupportVectorMachine", "LogisticRegression", "NearestNeighbors",
"RandomForest", "NaiveBayes",
"NeuralNetwork"};
(*"Запоминаем количество методов"*)
nom=Length [methods];
Print ["Количество используемых методов = ", nom];
(* ищем решающее правило каждым методом и тестируем его*)
For [i=1, i <= nom, i++,
decisionrule=Classify [<|"S1"->set1,"S2"->set2|>, Method-> methods [[i]],
PerformanceGoal-> "Quality"];
(* применяем решающее правило *)
Print [methods [[i]], " ", decisionrule [test]];
];

```

И в результате получим:

```

Training set 1 = 14
Training set 2 = 10
Test set = 5
Количество используемых методов = 6
SupportVectorMachine {S1, S2, S1, S2, S2}
LogisticRegression {S1, S2, S1, S2, S2}
NearestNeighbors {S1, S2, S1, S2, S2}
RandomForest {S1, S2, S1, S2, S2}
NaiveBayes {S2, S2, S2, S2, S2}
NeuralNetwork {S1, S2, S1, S2, S2}

```

Конечно, в этом месте у читателя может возникнуть вопрос: «Так я могу применять разные методы, даже не понимая их сути?». Самое странное, наш ответ будет: «Да!» И в этом, по большому счету, состоит концепция AUTOML, суть которой в следующем: «Зачем специалисту в конкретной области (медицинскому работнику высокой квалификации, технологу, химику, маркетологу, риелтору и т. п.) знать до тонкостей, как работает каждый метод? Для практической работы

ему достаточно найти решающее правило, подавая на вход которого конкретные параметры, он будет получать соответствующий ответ». Скажем более того: опыт работы с профессионалами в конкретных областях показывает, что они успешно применяют данное клише для своих исследований буквально на следующий день после пояснения, как им пользоваться, и получают впечатляющие результаты, не вникая в суть методов. И они благодарны, что мы не «гружим» их лишней математикой — им важен результат и его интерпретация.

Данное клише полностью отвечает простоте применения, не претендуя на использование всех мыслимых методов — только основных, и далее мы поясним их суть. Эти методы называются *логистическая регрессия, ближайшие соседи, случайный лес, наивный Байес и нейронные сети*. Именно в такой последовательности мы и будем их рассматривать.

4.2. Логистическая регрессия

Мы по-прежнему продолжаем рассматривать случай линейной разделимости множеств. Допустим, это так, и на обучающей выборке мы нашли решающее правило, но в качестве результата решения нас интересует не расстояние от гиперплоскости, а вероятность правильного отнесения к тому или иному множеству. Согласитесь, если совершенно случайно новое наблюдение точно попало на разделяющую множества гиперплоскость, то вы не сможете точно сказать, к какому из множеств оно относится. Более того, если оно немного выше или ниже гиперплоскости, сможете ли вы уверенно отнести его к конкретному множеству? Наверно, в случае четкого попадания наблюдения на гиперплоскость будет правильно сказать, что вероятность отнесения его к любому из множеств равна 0.5, а в случае отклонения в ту или другую сторону вероятность изменяется в зависимости от расстояния от разделяющей гиперплоскости. Иначе говоря, нам нужна функция, которая смогла бы расстояние переводить в вероятность. И такая функция есть — это *логистическая функция*.

Данная функция идеально подходит для перевода расстояния от точки до гиперплоскости в вероятность отнесения к тому или иному

классу. Обозначим расстояние от конкретной точки до гиперплоскости как $w(j)$. Тогда формула логистической функции будет $P(j) = 1 / (1 + \exp^{-w(j)})$, а ее график будет выглядеть следующим образом (рис. 4.3).

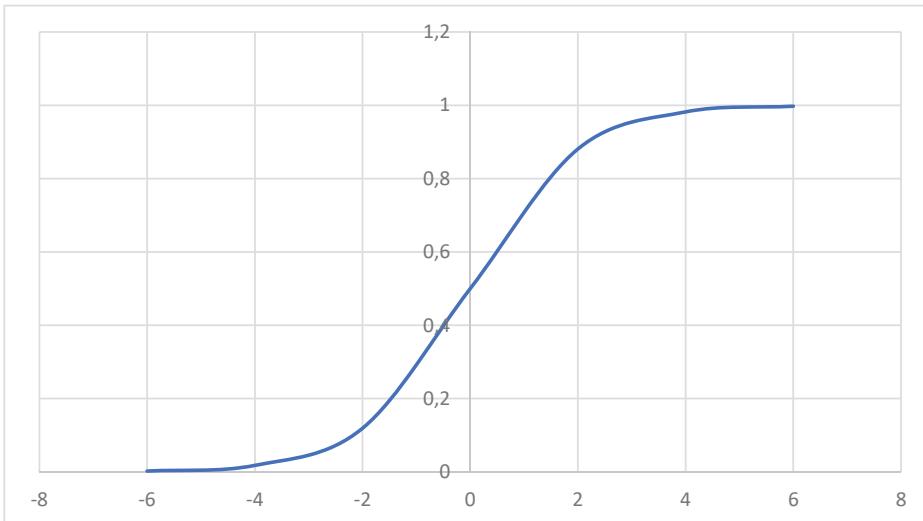


Рис. 4.3. Логистическая функция

Очевидно, что при больших отрицательных значениях w функция будет стремиться к нулю, при $w = 0 P = 0.5$, при больших положительных значениях w функция стремится к единице.

На тонкостях типа максимального правдоподобия и т. п. мы не будем останавливаться, так как практическим специалистам это и так понятно.

4.3. Основные идеи отнесения объектов наблюдения к конкретному классу

В предыдущих главах мы достаточно подробно рассмотрели методы классификации, базирующиеся на идее, что различные классы можно отделить друг от друга некоторой гиперповерхностью (линейной или нелинейной). Настало время рассмотреть другие идеи классификации и методы, построенные на их основе.

1. *Идея компактности (близости).* Объекты каждого класса расположены ближе к объектам своего класса, чем к объектам другого класса. Значит, надо рассчитать расстояние между объектами и на основе расстояний построить классификатор. На этой идеи базируется метод ближайших соседей и потенциальных функций.

2. *Логическая идея.* Решающее правило можно построить на основе логических деревьев.

3. *Статистическая идея.* Классификатор строится исходя из частоты повторения определенного события в зависимости от сочетания факторов.

4. *Нейросетевая идея.* Решающее правило строится по аналогии с работой мозга животных и человека.

На основе этих идей возникли различные школы машинного обучения. П. Домингос [26] в своей книге «Верховный алгоритм» выделяет следующие направления:

- 1) символизм — поиск логических закономерностей;
- 2) коннекционизм — обучаемые нейронные сети;
- 3) эволюционизм — адаптивная оптимизация сложных моделей;
- 4) байесианизм — оценивание распределений над параметрами;
- 5) аналогизм — «близким объектам — близкие ответы»;
- 6) композиционизм — кооперация моделей.

Рассмотрим суть различных методов.

4.4. Метод ближайших соседей

Как уже говорилось, данный метод базируется на идеи компактности. В соответствии с данным методом точки тестовой выборки можно относить к тому или иному множеству в зависимости от того, к какому множеству тестируемая точка ближе. Поскольку точек обучающей выборки с одинаковым расстоянием от тестируемой может быть несколько, то берется некоторое количество точек с минимальным расстоянием и принимается решение, к какому множеству отнести тестируемую точку. Поэтому такой подход и называется *метод ближайших соседей* (Nearest Neighbors, k-NN). Он применим как для классификации, так и для регрессии. В случае регрессии объекту присваивается

среднее значение k ближайших соседей. В случае классификации — наиболее часто встречающийся класс среди k ближайших соседей.

Метод ближайших соседей относится к метрическим методам. В случае количественных признаков обычно используется евклидово расстояние:

$$d = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \dots + (x_n - x'_n)^2}.$$

В случае бинарных признаков используется расстояние Хэмминга — это мера различия векторов одинаковой размерности. Так, между векторами 10111 и 11101 расстояние Хэмминга равно 2.

Если признаки количественные и их размерности сильно отличаются, то их необходимо нормализовать. Одним из часто применяемых методов нормализации является следующий:

$$p_{ij}^n = \frac{p_{ij} - \min(p_{ij})}{\max(p_{ij}) - \min(p_{ij})},$$

где p_{ij}^n — нормализованное значение i -го параметра j -го объекта; p_{ij} — ненормализованное значение; $\min(p_{ij})$ — минимальное значение параметра; $\max(p_{ij})$ — максимальное значение параметра.

При такой нормализации значение нормализованных параметров будет от нуля до единицы.

Можно использовать другой способ:

$$p_{ij}^n = \frac{p_{ij} - \text{среднее значение}(p_{ij})}{\text{стандартное отклонение}(p_{ij})}.$$

В Excel для вычисления среднего значения p_{ij} можно использовать оператор СРЗНАЧ, стандартного отклонения p_{ij} — оператор СТАНДОТКЛОН.

Среднее значение нормализованных равно 0, дисперсия равна 1.

Надо понимать, что нормализация может иметь и отрицательные моменты, например ухудшать интерпретируемость результатов решения. Выбор — нормализовать или нет — всегда остается за исследователем, и общие рекомендации давать сложно.

Естественно, существует много программных реализаций данного метода. Рассмотрим условный пример из прил. 6 (табл. 4.2).

Таблица 4.2

Данные для метода ближайших соседей

	X	Y			X	Y
	Обучающие выборки			Тестовая		
1	28	25			1	18
2	24	20			2	15
3	19	37			3	80
4	25	40			4	60
5	16	35			5	43
6	22	39			6	10
7	23	18				
8	13	26				
9	19	24				
10	11	18				
11	75	16				
12	73	11				
13	66	10				
14	75	10				
15	72	14				
1	43	32				
2	63	42				
3	18	47				
4	18	52				
5	57	21				
6	40	3				
7	58	54				
8	90	18				
9	57	32				
10	59	20				
11	89	2				
12	35	25				
13	81	38				
14	50	38				
15	89	1				
16	30	40				
17	31	55				
18	28	33				
19	76	30				
20	27	35				

Решим данный пример средствами Excel (табл. 4.3, см. также прил. 7) и Wolfram.

Таблица 4.3

Решение методом ближайших соседей средствами Excel

	X	Y		Расстояния до тестовых					
Обучающие выборки				T1	T2	T3	T4	T5	T6
1	28	25		11.18033989	13.34166406	53.60037313	35.34119409	21.21320344	30.8058436
2	24	20		11.66190379	9.219544457	56.56854249	41.18252056	21.47091055	33.10589071
3	19	37		7.071067812	15.5241747	65.9241989	41.10960958	36.12478374	15.8113883
4	25	40		12.20655562	20.59126028	61.7170965	35	34.98571137	18.02775638
5	16	35		5.385164807	13.03840481	68.00735254	44.28317965	36.79673899	16.15549442
6	22	39		9.848857802	18.38477631	63.97655821	38.01315562	35.80502758	16.2788206
7	23	18		13	8.94427191	57.31491952	43.0464865	21.54065923	34.53983208
8	13	26		6.403124237	4.472135955	68.44705983	49.04079934	34	24.18677324
9	19	24		6.08276253	4.472135955	62.16912417	44.01136217	27.78488798	27.51363298
10	11	18		13.89244399	5.656854249	69.26037828	53.71219601	32.984845	32.01562119
11	75	16		58.69412236	60.29925373	6.403124237	28.3019434	32.55764119	73.35529974
12	73	11		58.1893461	59.03388857	7.071067812	31.78049716	30.01666204	74.09453421
13	66	10		52	52.39274759	14.14213562	30.59411708	23	68.81860214
14	75	10		60.40695324	61.18823416	5.385164807	33.54101966	32	76.32168761
15	72	14		56.32051136	57.55866572	8.246211251	28.63564213	29.27456234	71.69379332
1	43	32		25.07987241	29.73213749	42.05948169	18.78829423	22	37.58989226
2	63	42		46.57252409	52	34.4818793	3.605551275	37.73592453	53.60037313
3	18	47		17	25.17935662	71.19691005	42.57933771	44.65422712	8.544003745
4	18	52		22	30.14962686	73.78346698	43.68065934	48.87739764	8.246211251
5	57	21		40.02499219	42.01190308	24.69817807	19.23538406	17.80449381	55.22680509
6	40	3		34.82814953	31.40063694	41	42.05948169	7.615773106	55.75840744
7	58	54		46.64761516	53.60037313	47.41307836	14.14213562	46.4865572	48.16637832
8	90	18		72.99315036	75.10659092	11.66190379	37.20215048	47.67598976	86.16263691
9	57	32		39.05124838	43.17406629	30.47950131	8.544003745	26.07680962	50.32891813
10	59	20		42.20189569	44.04543109	22.47220505	20.02498439	18.86796226	57.45432969
11	89	2		76.32168761	76.65507159	13.45362405	47.80167361	46.69047012	92.43916919
12	35	25		17.72004515	20.22374842	46.84015371	29.15475947	17	35.35533906
13	81	38		63.50590524	67.91170739	26.01922366	21.09502311	47.20169488	72.00694411
14	50	38		32.984845	38.48376281	39.69886648	10.19803903	28.86173938	41.76122604
15	89	1		76.69419796	76.92203845	14.2126704	48.60041152	46.87216658	92.96235797
16	30	40		15.62049935	23.43074903	57.30619513	30	32.69556545	22.36067977
17	31	55		28.17800561	36.67424164	65.19202405	32.64965543	46.57252409	21.58703314

Окончание табл. 4.3

	X	Y	Расстояния до тестовых					
Обучающие выборки			T1	T2	T3	T4	T5	T6
18	28	33	10.44030651	17.02938637	56.08029957	32.75667871	27.45906044	24.75883681
19	76	30		58	61.52235366	18.43908891	18.86796226	38.58756276
20	27	35	10.29563014	17.69180601	57.7754273	33.37663854	29.68164416	22.6715681
Тестовая								
1	18	30	5.385164807					
2	15	22		4.472135955				
3	80	12			5.385164807			
4	60	40				3.605551275		
5	43	10					7.615773106	
6	10	50						8.246211251

В Excel для вычисления суммы квадратов разности есть специальный оператор СУММКВРАЗН.

Программа на Wolfram выглядит следующим образом:

```
Print ["Указываем место нахождения исходных данных"];
file="D:\\ML\\data_for_examples.xlsx"
(* считываем обучающие и тестовые множества *)
set1= Import [file, {"Data", 1}];
Print ["Training set 1 = ", Length [set1]];
set2 = Import [file, {"Data", 2}];
Print ["Training set 2 = ", Length [set2]];
test= Import [file, {"Data", 3}];
Print ["Test set = ", Length [test]];
(* формируем решающее правило *)
decisionrule=Classify  [<|"1"->set1,"2"->set2|>, Method-
>"NearestNeighbors", PerformanceGoal-> "Quality"];
(* применяем решающее правило *)
result=decisionrule [test];
Print [result];
Print ["Расчеты закончены"]
```

Результаты работы программы Wolfram:

Указываем место нахождения исходных данных

D:\ML\data_for_examples.xlsx

Training set 1 = 15

Training set 2 = 20

Test set = 6

{1,1,1,2,2,2}

Расчеты закончены

4.5. Метод потенциальных функций (МПФ)

Часто пишут, что метод потенциальных функций является частным случаем метода ближайших соседей. Наверно, более правильно будет сказать, что данный метод является дальнейшим развитием метода ближайших соседей. Идея метода пришла, скорее всего, из физики. Для понимания данного подхода вспомним ряд физических законов. Начнем с законов Ньютона, открытых еще в 1666 г.— закона всемирного тяготения и принципа суперпозиции, которые звучат следующим образом.

Сила гравитационного притяжения между двумя материальными объектами с массами m_1 и m_2 , разделенными расстоянием r , действует вдоль соединяющей их прямой, пропорциональна обеим массам и обратно пропорциональна квадрату расстояния между ними:

$$F = \gamma * \frac{m_1 * m_2}{r^2},$$

где γ — гравитационная постоянная, то есть константа.

Принцип суперпозиции: результат воздействия на объект нескольких внешних сил есть векторная сумма этих сил.

Вспомним закон Кулона из области электростатики, открытый им в 1785 г.: модуль силы взаимодействия двух точечных зарядов в вакууме прямо пропорционален произведению модулей этих зарядов и обратно пропорционален квадрату расстояния между ними:

$$F = k \cdot \frac{q_1 \cdot q_2}{r^2},$$

где k — коэффициент пропорциональности, то есть константа.

Внешне закон всемирного тяготения и закон Кулона совпадают один в один. Мы не будем продолжать перечисление других физических законов, но отметим, что во многих из них сила воздействия объектов друг на друга убывает пропорционально квадрату расстояния между ними.

В наиболее общем виде функция называется *потенциальной*, если она резко убывает с ростом аргумента. Например, $F = \frac{1}{r^2}$, где F — функция, r — аргумент функции. Данную функцию неудобно использовать, так как при $r = 0$ она стремится к бесконечности. Поэтому на практике обычно используют функцию $F = \frac{1}{1+r^2}$.

Тогда если $r = 0$, то $F = 1$, F резко убывает с ростом r . Если значения r большие и с очень маленькими F работать неудобно, то в числителе вместо 1 можно использовать некоторую положительную константу.

Основная идея МПФ состоит в следующем. Считаем, что каждая точка любого множества генерирует некоторый потенциал на все остальные точки, как в законе всемирного тяготения или Кулона. Суммарный потенциал в каждой точке определяется как сумма потенциалов, исходящих от всех точек (принцип суперпозиции). Необходимо подобрать весовые коэффициенты для каждой точки таким образом, чтобы суммарные потенциалы для точек одного множества были положительными, а для другого — отрицательными.

Обычно в научных статьях для решения задач МПФ предлагаются различного рода итерационные алгоритмы, и сами авторы отмечают, что эти алгоритмы просты в реализации, но имеют плохую сходимость. Потому мы предлагаем представить данную задачу как ЗМП и решить стандартными методами. Более того, для решения задач относительно небольшой размерности нам даже не понадобятся пакеты математического программирования, так как задача может быть представлена как задача линейной алгебры, для решения которой требуется только вычислить обратную матрицу. Покажем, как решить предыдущий пример в Excel методом потенциальных функций. Выполним следующую последовательность действий:

1. Вычисляем квадраты евклидовых расстояний между всеми точками обучающей выборки (корень извлекать не надо).
2. На основе квадратов евклидовых расстояний вычисляем потенциалы.

3. Вычисляем обратную матрицу или решаем задачу линейного программирования для определения весовых коэффициентов всех точек.

4. Делаем разметку тестового множества.

Расчеты приведены в прил. 7. Дадим ряд пояснений. Очевидно, что матрица потенциалов квадратная, поэтому система уравнений $F^*a = b$ (где F — матрица потенциалов, a — вектор весовых коэффициентов наблюдений, b — некоторый произвольный ненулевой вектор) всегда имеет решение, если существует матрица F^{-1} , то есть обратная к F . Поэтому если векторе b элементам одного множества будет присвоено значение -1 , а другого 1 , то система будет иметь решение, если существует F^{-1} . Конечно, данную задачу лучше решать как ЗМП. Поэтому приведем ее запись сразу в кодах IBM ILOG CPLEX:

```
/*для каждой точки обучающей выборки необходимо подобрать вес таким образом, чтобы суммарный потенциал для каждой точки одного множества был положительным, а для каждой точки другого — отрицательным */
int k =...;//число наблюдений (объектов) в множестве 1
range i = 1..k;//индекс наблюдения (объекта)
int m =...;//число наблюдений (объектов) в множестве 2
range j = 1..m;//индекс наблюдения (объекта)
int n =...;//суммарное число наблюдений (объектов)
range s = 1..n;//индекс наблюдения (объекта)
float F1 [i] [s] =...;//потенциалы для точек множества 1 (C / (1+R^2 [i] [s])), //
где С некоторая положительная константа, R — евклидово расстояние
float F2 [j] [s] =...;//потенциалы для точек множества 2

/* искомые переменные */
dvarfloat a [s];//вес точки в суммарном потенциале
dvar float+ v [i];//невязка
dvar float+ w [j];//невязка
minimize sum (i in i) v [i] +sum (j in j) w [j];
subject to {
forall (i in i) sum (s in s) F1 [i] [s] *a [s] <= -1+v [i];
forall (j in j) sum (s in s) F2 [j] [s] *a [s] >= 1-w [j];
};
```

Теперь из прил. 7 матрицу потенциалов достаточно загрузить в CPLEX и сделать расчеты. Однако, прежде чем приводить результаты решения, давайте порассуждаем. Если мы используем k-NN, то

считать ли каждый объект соседом самому себе? Очевидно, что нет. Тогда, используя МПФ, надо ли учитывать потенциал, который объект генерирует сам на себя? Скорее всего, тоже нет. Давайте сделаем расчеты и сравним результаты. Для этого надо просто обнулить потенциалы на диагонали матрицы потенциалов и сделать расчеты. Приведем результаты решения и проанализируем их (табл. 4.4).

Таблица 4.4

Данные для МПФ

	X	Y	k_NN	МПФ Диагональ=С		МПФ Диагональ=0	
				коэффициент	суммарный	коэффициент	суммарный
Обучающие выборки				потенциал		потенциал	
1	28	25		-5.565481509	-54603.63922	0	-1.447116405
2	24	20		0	-973.3040936	0	-44.83069209
3	19	37		-0.925308676	-8794.815155	0	-59.86875202
4	25	40		-0.075997089	-1	0	-1
5	16	35		0	-420.6881612	0	-133.3403822
6	22	39		0	-1	0	-21.04909129
7	23	18		0	-484.4977689	0	-46.6149053
8	13	26		0	-48.39958404	-1.477587919	-1
9	19	24		0	-293.2093067	0	-337.7838619
10	11	18		0	-49.35714666	-0.275755069	-176.741808
11	75	16		-0.007276865	-71.54544363	0	-1
12	73	11		0	-1.743716989	-2.347810768	-1
13	66	10		0	-1	14.84528866	-678.9563614
14	75	10		0	-1	-1.744547644	-2106.145789
15	72	14		0	-4.098272995	0	-8.885073176
1	43	32		0	95.72188326	0	99.99434822
2	63	42		0	12.74852733	0	100.8560271
3	18	47		0	44.85514502	0	1
4	18	52		0	41.36172717	0.012451194	5.471791776
5	57	21		0	1	0	618.2969371
6	40	3		0.002711335	1	0	156.6316879
7	58	54		0	15.84629635	0	49.0815477
8	90	18		0	1.116856475	0	99.21162916
9	57	32		0	15.48958624	0	198.9255915
10	59	20		0	1.113117123	0	848.8954435
11	89	2		0.000122964	1.045899936	0	110.3050372
12	35	25		0.052083538	1	0	66.2526547

Окончание табл. 4.3

	X	Y	k_NN	МПФ Диагональ=С		МПФ Диагональ=0	
				коэффициент	суммарный	коэффициент	суммарный
	Обучающие выборки			потенциал		потенциал	
13	81	38		0	3.216452881	0	92.51334674
14	50	38		0	43.54524192	0	100.540956
15	89	1		0	1	0	112.1358785
16	30	40		0	966.5111317	0	20.82511532
17	31	55		0	59.21210776	0	21.80990852
18	28	33		7.053861523	69579.94609	0.002249238	1
19	76	30		0	3.297306633	0	185.7894845
20	27	35		0	11054.99839	0	1
Тестовая			Разметка тестовой выборки				
1	18	30			14.42713056		-322.3802622
2	15	22			-107.1799756		-745.0427068
3	80	12			-0.702920583		-306.2204567
4	60	40			15.53391222		112.9894402
5	43	10			-36.00215388		222.2564715
6	10	50			17.79165711		-2.11742122

Из данной таблицы можно сделать следующие выводы:

1. Результаты МПФ с обнуленной диагональю матрицы потенциалов ближе к методу ближайших соседей.
2. При использовании МПФ весовые коэффициенты некоторых наблюдений равны нулю, то есть в дальнейшем эти наблюдения можно не учитывать.

Вообще, строго говоря, в методе ближайших соседей нет никакого обучения, так как надо хранить всю обучающую выборку и каждый раз считать расстояние от тестируемой точки до всех точек обучающей выборки. Кроме того, всегда возникает вопрос: на сколько ближайших соседей надо ориентироваться? Это и есть его основные недостатки, но k-NN достаточно надежен и хорошо интерпретируем. Поэтому им часто пользуются практические работники.

В отличие от k-NN, МПФ позволяет исключать неинформативные наблюдения. Поэтому он более экономен по объему хранимой информации и времени на разметку контрольной выборки.

4.6. Логические методы

Изложение начнем с загадки, фокуса, принципа решения задач — подать это можно по-разному. Допустим, это загадка с простым вопросом: «Вы загадываете число от 1 до 1000. На мои вопросы отвечаете только „Да“ или „Нет“. Какое минимальное количество вопросов я должен задать, чтобы точно угадать загаданное число?» Люди, не знакомые с принципом дихотомии, обычно называют сильно завышенные цифры. Пусть это фокус или пари, Вы говорите кому-то: «Загадайте число от 1 до 1000. Я буду задавать вопросы, на которые вы будете отвечать „Да“ или „Нет“ — за 10 вопросов я точно назову задуманное число». В этом случае надо точно знать степени числа 2 и уметь быстро считать в уме.

Допустим, задумано число 651. Вопросы:

1. Больше или равно 512? Ответ: да.
2. Больше или равно $640 (512 + 512/2)$? Ответ: нет.
3. Больше или равно $576 [(512 + 640) / 2]$? Ответ: да.

И так далее. Поскольку $2^{10} = 1024$, то максимально необходимое число вопросов — 10.

Самое интересное, что, если будет несколько участников фокуса или пари, не знакомых с принципом дихотомии, и каждый загадает число, отличное от других, им будет сложно понять, что вы делаете. Потому что в вопросах вы будете называть разные цифры, при этом используя каждый раз один и тот же алгоритм, а именно — построение нового логического дерева решений.

Такой принцип решения задач настолько естественен, что многие его применяют, ничего не зная о дихотомии и машинном обучении. Например, вы позвонили в скорую помощь. Вам сразу зададут вопросы:

1. Температура есть?
2. Голова болит?
3. Кашель есть? И т. п.

О более сложных задачах можно сказать, что метод деревьев реализует принцип рекурсивного деления, представляя решающие правила в иерархической, последовательной структуре.

Продемонстрируем, как данный метод может быть применен к задаче об ирисах Фишера. Надеемся, что вы уже скачали необходимые данные из репозитория UCI Machine Learning Repository. Если нет,

то воспользуйтесь данными из прил. 10, папка ДАННЫЕ, таблица FROM_UCI, лист irises. В таблице Excel просто найдите границы каждого параметра. Это выглядит так (табл. 4.5).

Таблица 4.5
Данные для разделения ирисов Фишера деревом решений

Границы	p1	p2	p3	p4	ТИП
min	4.3	2.3	1	0.1	setosa
max	5.8	4.4	1.9	0.6	setosa
min	4.9	2	3	1	versicolor
max	7	3.4	5.1	1.8	versicolor
min	4.9	2.2	4.5	1.4	virginica
max	7.9	3.8	6.9	2.5	virginica
1	5.1				setosa
2	4.9				setosa
3	4.7				setosa
4	4.6				setosa
5	5				setosa
6	5.4				setosa
7	4.6				setosa
8	5				setosa
9	4.4				setosa
10	4.9				setosa

Легко увидеть, что по параметру p3 ирис setosa не пересекается с virginica и versicolor, причем зазор от 1.9 до 3. Для надежности берем середину и считаем, что построили первую веточку нашего логического дерева. Посмотрим, как это выглядит графически (рис. 4.4).

Деревья можно строить, рассматривая все параметры или беря за основу несколько из них. Например, на основе p3 и p2 может выглядеть следующим образом:

- 1) $p3 \leq 2.45 \rightarrow \text{setosa};$
- 2) $p3 > 5.1 \rightarrow \text{virginica}.$

Остальные ветви дерева предлагаем построить читателю самостоятельно, исходя из рис. 4.4.

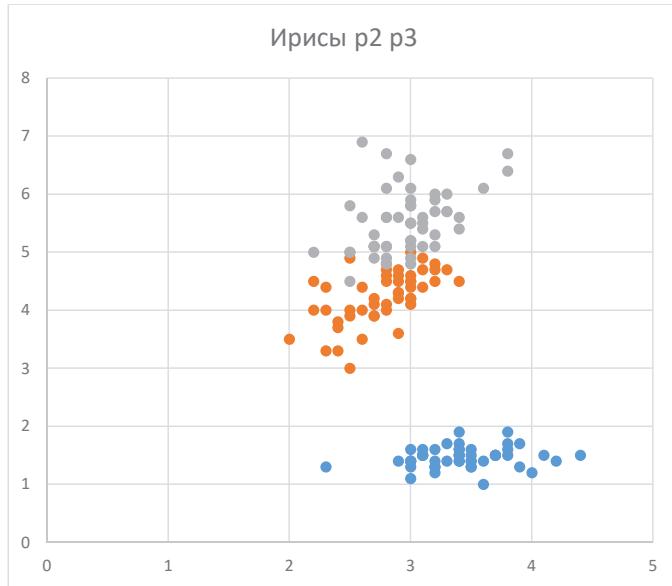


Рис. 4.4. Графическая иллюстрация ирисов Фишера

Деревьев может быть построено много, поэтому возникают вопросы:

1. Если существует несколько решений, приводящих к одинаковому результату (например, все объекты обучающей выборки идентифицированы), то какое решение выбрать?

2. Что делать, если ни одно дерево не дает надежный результат?

При ответе на первый вопрос можно руководствоваться бритвой Оккама и выбрать дерево с минимальным числом ветвлений. Специалисты из конкретных областей, скорее всего, предпочтут наиболее интерпретируемое с их точки зрения.

Для ответа на второй вопрос нам надо понять принципы создания эффективных ансамблей (комитетов) из одного или нескольких методов.

4.7. Ансамбли

Вспомним басни Крылова. Ансамбль из лебедя, рака и щуки явно не получился, то же самое произошло с мыртышкой, ослом, козлом и косолапым мишкой. С другой стороны, такие ансамбли, как

The Beatles, уже на протяжении нескольких десятилетий с удовольствием слушают многие поколения людей. В чем секрет и на чем базируется построение ансамблей? Возможны различные варианты объяснений. Попробуем рассмотреть ряд из них.

Еще Аристотель говорил: «Когда в процессе обсуждения участвуют многие, каждый может внести свою лепту добродетели и благородства... один понимает одну деталь, другой — другую, и все вместе понимают всё». Главное, что разные люди обращают внимание на разные детали, и, если их правильно соединить, коллективное знание будет обширнее (и подробнее) знания любого отдельного человека.

В качестве наглядного примера обычно приводят эксперимент Гальтона, который он провел на ярмарке в 1906 г. Народу поставили задачу: на взгляд определить вес быка. Кто будет ближе к истинному весу, получит приз. Отклонения каждого прогноза от истинного веса были значительными, но, когда посчитали среднее значение прогнозов, выяснилось, что толпа ошиблась на 1 фунт. Феномен называли «мудрость толпы».

Есть более строгое объяснение — теорема Кондорсе, сформулированная им в 1784 г.: «Если каждый член жюри присяжных имеет независимое мнение и если вероятность правильного решения члена жюри больше 0.5, то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри и стремится к единице. Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных».

Хорошее и эмоциональное объяснение дал Владимир Маяковский: «Плохо человеку, когда он один. Горе одному, один не воин — каждый дюжий ему господин, и даже слабые, если двое. А если в партию сгрудились малые — сдайся, враг, замри и ляг!»

Если говорить вообще, то, конечно, бывают эффективными и монархии (единоличное решение), но опыт человечества все же показывает, что на большом промежутке времени коллективные решения эффективней. В социуме мы их называем парламентом, думой, советом директоров и т. п. В машинном обучении они называются *ансамбль*, *комитет*, *аркинг*. Суть от этого не меняется — это коллективные решения. Вспомним: не можем разделить одной гиперплоскостью — пробуем разделить несколькими. Не устраивает прогноз по одному бли-

жайшему соседу — давайте будем рассматривать k ближайших соседей. Вернемся к логическим деревьям: если одно не дает правильного решения с большой вероятностью, так давайте построим лес из этих деревьев, который даст нам удовлетворяющее решение.

Этот метод называется *случайный лес* (Random Forest). Как пользоваться им в Wolfram, вы уже понимаете. Надо просто в операторе Classify указать это имя. Данный метод был создан Лео Брейманом в 2001 г. Последовательность действий следующая:

1. Из обучающей выборки генерируется случайная подвыборка (50–70 % от обучающей).
2. Строится дерево решений, причем в каждом новом узле дерева переменная для разбиения выбирается не из всех признаков, а из случайно выбранного их подмножества небольшой мощности.
3. Таких деревьев создается достаточно много, например несколько сотен.
4. Из построенных деревьев по определенным правилам создается ансамбль.

При создании ансамблей, конечно, возникает много вопросов. К ним мы еще вернемся. А пока рассмотрим оставшиеся идеи классификации.

4.8. Наивный метод Байеса (НМБ)

Данный метод относится к статистическим и, наверное, исторически возник намного раньше других. Вероятно, поэтому он считается очень легким для восприятия. Давайте его рассмотрение мы начнем с простой задачи. Обязательно попытайтесь решить ее сами, не заглядывая в ответ.

«Вот ситуация, с которой часто сталкиваются врачи: 1 % женщин в возрасте 40 лет, участвовавших в регулярных обследованиях, имеют рак груди; 80 % женщин с раком груди имеют положительный результат маммографии; 9.6 % здоровых женщин также получают положительный результат (маммография, как любые измерения, не дает на 100 % точных результатов). Женщина-пациент из этой возрастной группы получила положительный результат на регулярном обследо-

вании. Какова вероятность того, что она фактически больна раком груди?»

Это цитата из книги Элиезера Юдковски «Наглядное объяснение теоремы Байеса» [28].

Мы специально приводим данную цитату, так как сами имеем достаточно большой опыт общения со студентами и практическими работниками из различных отраслей, и другая часть цитаты полностью совпадает с нашим мнением. «Хотя в Сети есть несколько (немного!) онлайновых объяснений теоремы Байеса, мой опыт с попытками познакомить кого-либо с байесианским мышлением говорит о том, что все они слишком абстрактны. Байесианская мысль очень *контр-интуитивна*. Люди не могут применять байесианское мышление автоматически, находят его весьма трудным в изучении и быстро забывают байесианские методы после его окончания» [27].

Почему же так? Ведь теорема Байеса давно известна и звучит вроде бы просто и понятно:

$$P(A|B) = P(B|A) * P(A) / P(B),$$

где $P(A)$ — априорная вероятность гипотезы A ; $P(A|B)$ — вероятность гипотезы A при наступлении события B ; $P(B|A)$ — вероятность наступления события B при истинности гипотезы A ; $P(B)$ — вероятность наступления события B .

Вербальное объяснение наивного Байесовского метода, которое обычно приводят, состоит в следующем.

«В основе данного классификатора лежит теорема Байеса, позволяющая определить вероятность какого-либо события при условии, что произошло другое статистически взаимозависимое с ним событие.

Соответственно, данный классификатор использует оценку апостериорного максимума для определения наиболее вероятного класса с предположением о независимости признаков математической модели друг от друга. Именно из-за последнего условия данный метод называют наивным, так как зачастую в математических моделях признаки имеют зависимость между собой».

Вопрос: у вас есть задача по маммографии, есть теорема Байеса, есть определение НМБ. Вы продвинулись в решении задачи? Каков Ваш ответ?

Продолжим цитирование: «Теперь предположим, что я скажу вам: большинство докторов дают неверный ответ — обычно лишь около

15 % врачей способны решить задачу правильно. („В самом деле? 15 %? Это реальные данные или городская легенда, основанная на опросах в Интернете?“ Да, это реальные данные. См. Casscells, Schoenberger, and Grayboys 1978; Eddy 1982; Gigerenzer and Hoffrage 1995 и многие другие статьи. Это удивительный результат, который, однако, легко воспроизводится и потому воспроизводится в широких масштабах.)

Ошибкой, которая постоянно совершается в этой задаче, является игнорирование наличия двух групп — действительно больных раком и здоровых, получающих ложноположительные результаты теста, и фокусирование только на группе больных и получивших положительные результаты. Например, абсолютное большинство врачей в уже упоминавшихся исследованиях полагают, что если около 80 % женщин с раком груди имеют положительные маммограммы, то и вероятность для женщины с положительной маммограммой быть больной раком тоже около 80 %.

Правильное вычисление окончательного ответа требует *всех трех* условий задачи — процента женщин с раком груди, процента здоровых женщин с ложноположительными результатами теста и процента женщин с раком груди, получивших истинно положительные маммограммы» [27].

На наш взгляд, понять, как решать данную задачу, исходя из теоремы Байеса, действительно сложно. Поэтому попробуем пояснить без нее в таблице Excel. Для простоты пояснения будем считать, что обследовано какое-то конкретное большое количество женщин — пусть сто тысяч. Занесем данные в ячейки табл. 4.6.

Таблица 4.6

Решение примера средствами Excel

	A	B	C	D	E	F	G
1	Всего обследованных	Результат теста	доля здоровых с положительным результатом	доля больных с положительным результатом			
2	100000		0.096	0.8			
3			здоровые	больные			вероятность %
4		да	9504	800	10304	7.763975155	
5		нет	89496	200	89696		
6	% здоровых	0.99 ИТОГО	99000	1000			

Окончание табл. 4.6

	A	B	C	D	E	F	G
1	Всего обследованных	Результат теста	доля здоровых с положительным результатом	доля больных с положительным результатом			
2	100000		0.096	0.8			
3		здоровые	больные		вероятность %		
4	да	=ОКРУГЛ(Д2*D6;0)	=ОКРУГЛ(Е2*Е6;0)	=D4+E4	=E4/F4*100		
5	нет	=D6-D4	=E6-E4	=D5+E5			
6	% здоровых	0.99 ИТОГО	=B2*B6	=B2-D6			

Проанализируйте данные таблицы самостоятельно — и вы убедитесь, что вероятность действительно примерно 7.76 %.

Учитывая сложность понимания байесовского подхода, распишем подробно логику вычислений.

- Если было обследовано 100 000 женщин (в общем случае S), то 99 000 ($0.99 * S$) из них будут здоровы и только 1000 — больны ($0.01 * S$).
- Значит, 9504 ($0.096 * 0.99 * S$) здоровых женщин получат ложно-положительный диагноз, то есть подозрение на то, что они больны.
- Из 1000 больных женщин 800 ($0.8 * 0.01 * S$) получат тоже положительный диагноз, и они действительно больны.
- Значит, получив положительный диагноз, не стоит сразу паниковать, так как реальная вероятность, что вы действительно больны $800 / (800 + 9504) = 0,0776 [0.8 * 0.01 * S / (0.8 * 0.01 * S + 0.096 * 0.99 * S)]$.
- Выносим S скобки в знаменателе, сокращаем S и получаем:

$$\frac{\text{Действительно больные}}{\text{Действительно больные} + \text{Ложно больные}} = \\ = \frac{0.8 * 0.01}{0.8 * 0.01 + 0.096 * 0.99} = 0.0776.$$

Контринтуитивность наивного байесовского метода и излишняя академичность его объяснения приводят к сложности его понимания, что, в свою очередь, приводит к различным курьезам. Например, наберите в любом поисковике «байесианский заговор». Ответом, скорее всего, будет что-то типа: «Байесианский Заговор — многонациональная, междисциплинарная и действующая в тени группа ученых, контролирующая публикации, гранты, пожизненные профессорские должности и незаконный ввоз аспирантов. Лучший способ быть принятым в Байесианский Заговор — присоединиться к Университетскому

Таблица 4.7

Формулы для условного примера

Крестовому Походу за Байеса еще в школе или в колледже и постепенно повышать градус посвящения. Ходят слухи, что на верхних уровнях Байесианского Заговора существуют девять молчаливых фигур, известных как Байесианский Совет».

Конечно, это шутка, но так как мы хотим добиться практического понимания НМБ, приведем еще несколько примеров. Мы сознательно будем давать отличающиеся по форме, но одинаковые по сути формулировки. Решите следующую задачу.

Тест на болезнь имеет вероятность успеха 95 %, то есть вероятность ошибки первого рода (ложного срабатывания, *false positive*) и ошибки второго рода (пропуск больного, *false negative*) — 5 %. Болезнь распространена и имеется у 1 % респондентов. Определите условные вероятности исходов.

Решение приведем в табличной форме (табл. 4.7).

Данная таблица приведена в прил. 8. Расчет по формуле: $0.95 * 0.01 / (0.95 * 0.01 + 0.05 * 0.99) = 0.1610$.

4.9. Нейронные сети

В последнее время в анализе данных широкое распространение получили нейронные сети (НС).

НС — это модель, имитирующая работу головного мозга (отсюда и название). Нейрон — это некоторая конструкция, на вход которой подается множество сигналов, которые внутри нейрона преобразуются в один выходной сигнал. Данный сигнал по синапсам (межнейронным связям) с определенным весом, соответствующим синапсу, передается другим нейронам. Нейроны обычно располагаются слоями. Архитектура сети зависит от типа нейронов, количества слоев и типа связей между нейронами. Нейронные сети можно обучать. Что-нибудь полезное поняли? Наверно, только источник названия.

Скажем по-другому. В математическом смысле *нейронная сеть* — это иерархически упорядоченный, хорошо подобранный набор формул различной сложности, в которых результаты одних являются входными параметрами для других. «Обучить» означает подобрать такие коэффициенты в формулах, чтобы входные сигналы правильно преобразовывались в выходные.

Стоп! А разве не то же мы делали, строя различные комитеты? Да, в нашем случае все комитеты — это двухслойные нейронные сети с нейронами линейного типа и выходным сигналом нейрона (гиперплоскости) 0 или 1. Поэтому в случае использования комитета можете эти три понятия — *член комитета, гиперплоскость и линейный нейрон* — использовать как слова-синонимы.

В методе комитетов вместо линейных функций (гиперплоскостей) можно использовать нелинейные гиперповерхности, но ограниченного типа. В НС нелинейные нейроны используются довольно часто, поэтому НС позволяют отслеживать более тонкие зависимости между входными и выходными параметрами. Правда, если при использовании метода комитетов есть хоть какие-то надежды на содержательную интерпретацию, то НС — это всегда черный ящик.

Приведем графическую интерпретацию нейронных сетей (рис. 4.5).

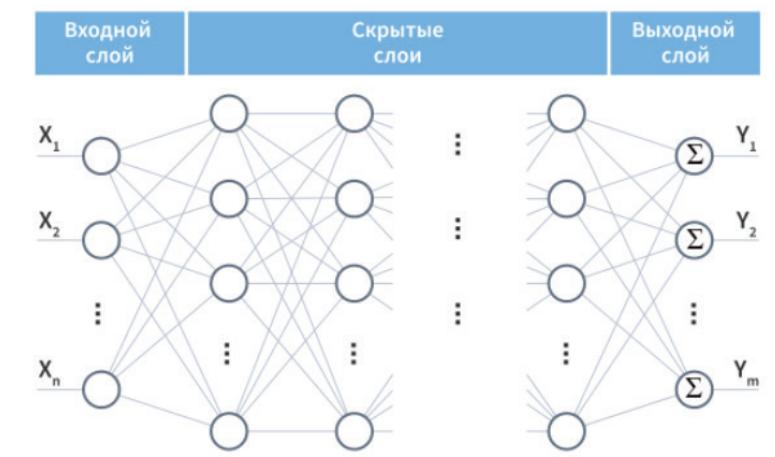


Рис. 4.5 Графическая интерпретация нейронной сети [28]

Математическая модель нейрона была предложена Уорреном Мак-Каллоком и Уолтером Питтсом в 1943 г. [29]. Первый нейрокомпьютер *Mark-1* был создан Фрэнком Розенблаттом в 1960 г. Однако в 60-е годы XX в. данный метод не нашел широкого применения. Более того, в 1969 г. Марвин Минский опубликовал доказательство ограниченности применения перцептрона, и интерес к нейросетям резко упал. В 1986 г. несколько авторов независимо друг от друга опубликовали статьи по методу обратного распространения ошибки, и интерес к нейросетям снова резко возрос.

Поскольку данный метод требует больших вычислительных ресурсов, в XX в. были некоторые сложности с его широким использованием. Ситуация изменилась коренным образом в XXI в., так как необходимые ресурсы появились. В настоящее время существуют различные разновидности нейронных сетей: перцептрон Розенблатта, сплайн-модель Хакимова, многослойный перцептрон Розенблатта, многослойный перцептрон Румельхарта, сеть Джордана, сеть Элмана, сеть Хэмминга и т. д.

В данном случае мы привели названия сетей по фамилиям их авторов. Возможна типологизация сетей в зависимости от их архитектуры, типа входных данных, характера обучения, характера связей между нейронами и т. п.

Однако с математической точки зрения всё равно все виды сетей — это многопараметрическая задача нелинейной оптимизации.

Существуют различные методы решения таких задач. Самым простым в реализации является метод градиентного спуска — поясним его суть на простых примерах.

4.10. Градиентный спуск

Градиентный спуск — метод нахождения локального экстремума функции с помощью движения в направлении градиента.

Градиент — вектор, указывающий направление максимального прироста функции, то есть это производная по пространству.

Сделаем небольшое пояснение. Если в функции одна переменная, то и производная одна, то есть скаляр. Если в функции несколько переменных, то и производных несколько, значит, вектор. Поясним сказанное простым примером (прил. 9). Допустим, нам требуется найти локальный экстремум функции $f(x, y) = x^3 + 2y^2 - 3x - 5y$. Вычисляем частные производные:

$$\frac{df}{dx} = 3x^2 - 3, \quad \frac{df}{dy} = 4y - 5.$$

Дальнейшие расчеты с пояснениями приведены в табл. 4.8.

Таблица 4.8

Расчеты методом градиентного спуска с шагом 0.1

		Функция	Произв. по х	Произв. по У	шаг (задаем)	
		$x^3+2y^2-3x-5y$	$3x^2-3$	$4y-5$	0.1	
		$f(x,y)$	df/dx	df/dy		корень((df/dx) $^2 + (df/dy)^2$) <-- будет убывать, останавливаем счет при
итерация						модуль градиента
1	0	0	-3	-5		
2	0.3	0.5	-2.873	-2.73	-3	5.831 На начальном шаге значения х и у задаем
3	0.573	0.8	-4.2509	-2.015	-1.8	4.0562 $x(2)=x(1)- шаг * df/dx$
4	0.7745	0.98	-4.8381	-1.2004	-1.08	2.7019 $y(2)=y(1)- шаг * df/dy$
5	0.89454	1.088	-5.0403	-0.5994	-0.648	1.6147
6	0.954448	1.1528	-5.1	-0.2669	-0.3888	0.8827
7	0.98117	1.19168	-5.1171	-0.1119	-0.2333	0.4716
8	0.99236	1.21501	-5.1224	-0.0457	-0.14	0.2587
9	0.99693	1.22901	-5.1241	-0.0184	-0.084	0.1473
10	0.99877	1.23741	-5.1247	-0.0074	-0.0504	0.086
11	0.99951	1.24245	-5.1249	-0.0029	-0.0302	0.0509
12	0.9998	1.24547	-5.125	-0.0012	-0.0181	0.0303
13	0.99992	1.24728	-5.125	-0.0005	-0.0109	0.0181
14	0.99997	1.24837	-5.125	-0.0002	-0.0065	0.009
						0.0065

Что произойдет, если мы увеличим шаг до 0.2 (табл. 4.9)?

Таблица 4.9

Расчеты методом градиентного спуска с шагом 0.2

			Функция	Произв. по x	Произв. по Y	шаг (задаем)
			$x^3+2y^2-3x-5y$	$3x^2-3$	$4y-5$	0.2
итерация	x	y	f(x,y)	df/dx	df/dy	корень((df/dx)^2+(df/dy)^2) модуль градиента
1	0	0	0	-3	-5	5.831
2	0.6	1	-4.584	-1.92	-1	2.1648
3	0.984	1.2	-5.1192	-0.0952	-0.2	0.2215
4	1.00304	1.24	-5.1248	0.0183	-0.04	0.044
5	0.99938	1.248	-5.125	-0.0037	-0.008	0.0088
6	1.00012	1.2496	-5.125	0.0007	-0.0016	0.0017
7	0.99998	1.24992	-5.125	-0.0001	-0.0003	0.0003
8	1	1.24998	-5.125	0	-0.0001	0.0001
9	1	1.25	-5.125	0	0	0

Количество итераций резко уменьшилось, и мы даже достигли ситуации, когда модуль градиента равен нулю, то есть минимум функции достигается в точке $x = 1$, $y = 1.25$.

Так, может, следует еще увеличить шаг? Сделаем его равным 0.5 (табл. 4.10).

Таблица 4.10

Расчеты методом градиентного спуска с шагом 0.5

			Функция	Произв. по x	Произв. по Y	шаг (задаем)
			$x^3+2y^2-3x-5y$	$3x^2-3$	$4y-5$	0.5
итерация	x	y	f(x,y)	df/dx	df/dy	корень((df/dx)^2+(df/dy)^2) модуль градиента
1	0	0	0	-3	-5	5.831
2	1.5	2.5	-1.125	3.75	5	6.25
3	-0.375	0	1.0723	-2.5781	-5	5.6255
4	0.91405	2.5	-1.9785	-0.4935	5	5.0243
5	1.1608	0	-1.9183	1.0424	-5	5.1075
6	0.6396	2.5	-1.6571	-1.7727	5	5.3049
7	1.52595	0	-1.0246	3.9856	-5	6.3941
8	-0.4669	2.5	1.2988	-2.3462	5	5.5231
9	0.70625	0	-1.7665	-1.5036	-5	5.2212
10	1.45805	2.5	-1.2745	3.3777	5	6.034
11	-0.2308	0	0.6801	-2.8402	-5	5.7504
12	1.1893	2.5	-1.8857	1.2433	5	5.1523
13	0.56765	0	-1.52	-2.0333	-5	5.3976
14	1.5843	2.5	-0.7763	4.53	5	6.7469
15	-0.6807	0	1.7267	-1.6099	-5	5.2528
16	0.12425	2.5	-0.3708	-2.9537	5	5.8073
17	1.6011	0	-0.6988	4.6906	-5	6.8558
18	-0.7442	2.5	1.8204	-1.3385	5	5.1761
19	-0.075	0	0.2244	-2.9831	-5	5.8223
20	1.4166	2.5	-1.407	3.0203	5	5.8414
21	-0.0936	0	0.2798	-2.9737	-5	5.8175
22	1.3933	2.5	-1.4751	2.8239	5	5.7423
23	-0.0187	0	0.0559	-2.999	-5	5.8304

Таблица 4.11

Другой вариант метода градиентного спуска

		Функция		Произв. по х		Произв. по У		шаг (задаем)			
итерация	x	y	f(x,y)	df/dx	df/dy	корень((df/dx)^2+(df/dy)^2)	модуль градиента	cos(x)	cos(y)		
1	0	0	0	-3	-5	5.831	0.545	-0.8413	0.5354	-0.5405	-0.8575
2	0.05145	0.085749	-0.5682	-2.9921	-4.657	5.2409	0.566	-0.8413	0.5354	-0.5405	-0.8575
3	0.1055	0.16988	-1.107	-2.9666	-4.3205	4.9456	0.5907	-0.8444	0.5209	-0.5405	-0.8575
4	0.16211	0.252318	-1.6163	-2.9212	-3.9907	4.9456	0.5907	-0.8069	0.5209	-0.5405	-0.8575
5	0.22117	0.333301	-2.096	-2.8532	-3.668	4.647	0.614	-0.7893	0.5209	-0.5405	-0.8575
6	0.28257	0.411943	-2.5455	-2.7605	-3.3572	4.3425	0.6357	-0.772	0.5209	-0.5405	-0.8575
7	0.34614	0.489138	-2.9641	-2.6406	-3.0434	4.0293	0.6553	-0.7553	0.5209	-0.5405	-0.8575
8	0.41168	0.564669	-3.3509	-2.4916	-2.7413	3.7044	0.6726	-0.74	0.5209	-0.5405	-0.8575
9	0.47894	0.638671	-3.7045	-2.3919	-2.4453	3.3652	0.6887	-0.7266	0.5209	-0.5405	-0.8575
10	0.54764	0.711335	-4.0234	-2.1003	-2.1547	3.009	0.698	-0.7161	0.5209	-0.5405	-0.8575
11	0.61744	0.782943	-4.3056	-1.8563	-1.8682	2.6336	0.7049	-0.7094	0.5209	-0.5405	-0.8575
12	0.68792	0.853381	-4.5494	-1.5803	-1.5845	2.2379	0.7062	-0.708	0.5209	-0.5405	-0.8575
13	0.75854	0.924684	-4.7525	-1.2739	-1.3013	1.821	0.6996	-0.7146	0.5209	-0.5405	-0.8575
14	0.8285	0.996144	-4.9129	-0.9408	-1.0154	1.3842	0.6797	-0.7336	0.5209	-0.5405	-0.8575
15	0.89646	1.069501	-5.0288	-0.5891	-0.722	0.9318	0.6322	-0.7748	0.5209	-0.5405	-0.8575
16	0.95968	1.146385	-5.099	-0.237	-0.4121	0.4754	0.4985	-0.8668	0.5209	-0.5405	-0.8575
17	1.00954	1.23367	-5.1242	0.0575	-0.0653	0.087	0.6609	-0.7506	0.5209	-0.5405	-0.8575
18	0.94345	1.308728	-5.1087	-0.3297	0.2349	0.4048	0.8145	0.5803	0.5209	-0.5405	-0.8575
19	1.02489	1.250699	-5.1231	0.1512	0.0028	0.1512	1	0.0185	0.5209	-0.5405	-0.8575
20	0.92489	1.248847	-5.1085	-0.4337	-0.0046	0.4337	-1	-0.0106	0.5209	-0.5405	-0.8575
21	1.02489	1.249908	-5.1231	0.1512	-0.0004	0.1512	1	0.0026	0.5209	-0.5405	-0.8575
22	0.92489	1.250172	-5.1085	-0.4337	0.0007	0.4337	-1	0.0016	0.5209	-0.5405	-0.8575
23	1.02489	1.250011	-5.1231	0.1512	0	0.1512	1	0	0.5209	-0.5405	-0.8575
24	0.92489	1.250011	-5.1085	-0.4337	0	0.4337	-1	0	0.5209	-0.5405	-0.8575
25	1.02489	1.250011	-5.1231	0.1512	0	0.1512	1	0	0.5209	-0.5405	-0.8575
26	0.92489	1.250011	-5.1085	-0.4337	0	0.4337	-1	0	0.5209	-0.5405	-0.8575

Вместо уменьшения числа итераций мы получили несходящийся процесс, потому что шаг слишком большой, и мы просто проскакиваем точку локального минимума.

У метода градиентного спуска существуют различные варианты. Например, представленный в табл. 4.11.

Отличие методов в следующем. В первом случае

$$x_{t+1} = x(t) - \text{шаг} * df/dx, \quad y_{t+1} = y(t) - \text{шаг} * df/dy,$$

где t — номер итерации.

Во втором случае

$$x_{t+1} = x(t) - \text{шаг} * \cos(x), \quad y_{t+1} = y(t) - \text{шаг} * \cos(y).$$

Из приведенной таблицы видно, что в нашем случае второй способ не подходит.

Приведенные в данной главе таблицы находятся в прил. 9.

4.11. Принципы построения эффективных ансамблей и оценка качества решения

Теперь, когда вы уже умеете использовать различные методы для решения задач машинного обучения, вернемся к теме ансамблей. В разделе 4.7 вы уже познакомились с теоремой Кондорсе, которая является математическим обоснованием того, что создание эффективных ансамблей возможно. Более того, вы уже умеете строить комитеты из различных гиперповерхностей, пользоваться случайным лесом и методом ближайших соседей, то есть создавать ансамбли на основе многократного использования одного классификатора. Каждый классификатор имеет некоторую ошибку обучения. Естественно, если ошибка обучения $\geq 50\%$, алгоритм считать классификатором нельзя. Алгоритм обучения, имеющий ошибку обучения менее 50% , далее будем называть *слабым классификатором*, а *сильным классификатором* — алгоритм обучения, позволяющий добиться произвольно малой ошибки обучения. Теперь надо разобраться, как на основе слабых классификаторов создать сильный. Существует несколько подходов.

1. **Беггинг (Bootstrap Aggregating)**: один алгоритм обучается много раз на случайных выборках из исходных данных. Таким образом, снижается зависимость классификаторов друг от друга. Благодаря различности подмножеств, ошибки, возникающие на них, взаимно компенсируют друг друга. Объекты-выбросы могут не попадать в некоторые обучающие подвыборки.

Расчеты не зависят друг от друга, поэтому их можно вести параллельно (особенно если есть видеокарты). Поэтому беггинг превосходит по скорости нейросети.

2. **Бустинг (Boosting)**: обучает алгоритмы последовательно, обращая особое внимание на случаи, где ошибся предыдущий алгоритм. В новую выборку обязательно берется часть данных, на которых ошибся предыдущий, то есть происходит постоянное «дообучение». Минус данного подхода: расчеты нельзя распараллелить; плюс — высокая точность.

Особенность бустинга заключается в том, что он строится только на слабых классификаторах. Причины: сильный классификатор, давая нулевую ошибку на обучающих данных, не адаптируется, и композиция будет состоять из одного классификатора; один, даже сильный, классификатор может дать «плохое» предсказание на данных тестирования, давая «хорошие» результаты на обучающих данных.

3. **Стекинг (Stacking)**: независимо обучаются несколько слабых алгоритмов, предсказания которых затем используют для обучения метамодели. Например, обучают линейную регрессию, лес и ближайших соседей, а затем их результаты объединяют нейронной сетью.

Допустим, мы решили задачу классификации несколькими методами и надо выбрать наилучший. Для того чтобы это можно было сделать, необходимы формализованные оценки качества решения (метрики). Хотя основой проверки качества решающего правила является тестовая выборка, способы оценки качества решения применимы как к тестовой, так и к обучающей выборкам. Теоретические основы качества решений и наиболее применяемые показатели достаточно подробно рассмотрены нами в [21, с. 42–48]. Клише программ классификации с оценкой качества решения приведены в прил. 10.

4.12. Переобучение

В предыдущих разделах мы показали, что из слабых классификаторов можно сделать, в принципе, сколько угодно сильный классификатор. Например, при разделении классов гиперповерхностями, наращивая число членов комитета, всегда можно добиться стопроцентного разделения на обучающей выборке, так как комитеты большинства и старшинства всегда существуют [24]. Аналогично в задачах регрессии, увеличивая степень полинома, мы каждый раз будем точнее аппроксимировать обучающую выборку. Возникает закономерный вопрос: когда следует остановиться?

Мы специально подчеркнули, что улучшение обязательно будет на обучающей выборке. При этом на контрольной выборке качество решения будет до определенного предела расти, а потом начнет снижаться. Этот момент очень важен, и означает он, что наступило переобучение и дальнейшие попытки улучшить решения одновременно на обучающей и контрольной выборках не только бессмысленны, но и вредны.

Геометрическая интерпретация может быть следующая (рис. 4.6).

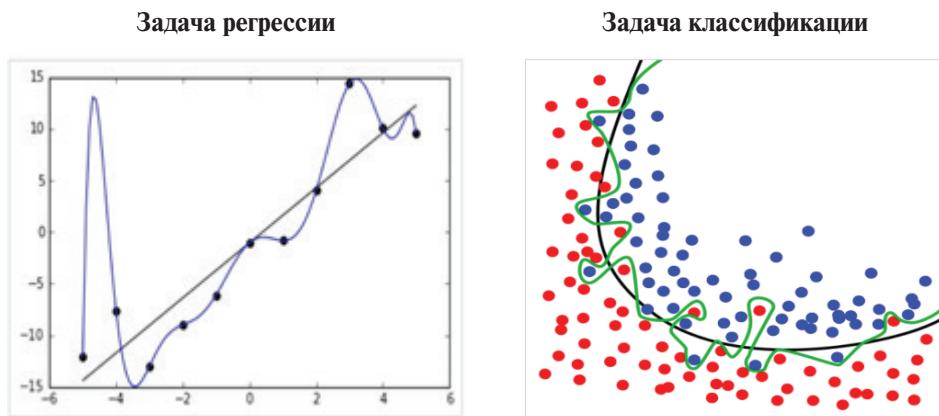


Рис. 4.6. Переобучение [30]

Конечно, в первую очередь внимание обращают на результаты на контрольной выборке, так как если вы провалили экзамен, то бессмысленно убеждать, что вы усердно готовились. Более того, если вы будете участвовать в конкурсах, то тестовую выборку вообще не увидите в полном объеме.

На практике обычно пользуются принципом двойной бритвы Оккама:

- 1) из двух моделей с одинаковыми ошибками тестирования предпочтительней простейшая;
- 2) из двух моделей с одинаковыми ошибками обучения предпочтительней простейшая.

4.13. Обобщенный взгляд на задачи регрессии, классификации, кластеризации и методы их решения

До сих пор задачи регрессии и классификации мы рассматривали отдельно, как и методы их решения. На практике, имея конкретный набор данных, не всегда сразу понятно, какую задачу придется решать и какими методами. Вернемся к задаче энергоэффективности зданий. Если кто-то сформулирует вопрос: «Я покупаю здание со следующими параметрами... Какими будут величины нагревательной и охлаждающей нагрузки?», то вам надо будет решать задачи регрессии.

Если же последует задание подобрать допустимые границы параметров здания так, чтобы нагревательная и охлаждающая нагрузка были меньше некоторых пороговых величин, то решать придется задачу классификации. Поэтому в UCI Machine Learning Repository эта задача и указана как задача классификации и регрессии.

Приведем еще ряд примеров из этого репозитория:

Название задачи	Тип задачи
Качество вина	Классификация, регрессия
Семена	Классификация, кластеризация
Статистика матчей главного теннисного турнира	Классификация, кластеризация, регрессия
Оценки уровня ожирения на основе привычек питания и физического состояния	Классификация, кластеризация, регрессия
Сеть метаболических отношений KEGG	Классификация, кластеризация, регрессия

Как видно из приведенной таблицы, задача может быть даже одновременно трех типов — все зависит от того, какой конечный результат необходим. Примерно такая же ситуация и с методами: в принципе, для решения любого типа задач подходят все методы, просто одни лучше, а другие хуже.

В данном пособии мы еще не рассматривали кластеризацию. Как уже было сказано, она может строиться на любом из перечисленных методов. Конечно, задачи кластеризации имеют свою специфику. Методы их решения можно объединить в следующие группы: статистические методы, сети Кохонена, иерархическая кластеризация. Для решения практических задач можем рекомендовать следующее клише на Wolfram Mathematica:

```
file="D:\\ML\\data_for_examples.xlsx"
(* считываем обучающее и контрольное множества *)
set1=Import[file, {"Data", 5}];
Print ["Training set = ", Length [set1]];
set2 = Import [file, {"Data", 6}];
Print ["Test set = ", Length [set2]];
(* ищем указанным методом решающее правило *)
decisionrule=ClusterClassify [set1];
(* применяем решающее правило *)
decisionrule [set1]
decisionrule [set2]
(* задаем число классов *)
classes=2
decisionrule=ClusterClassify [set1, classes];
(* применяем решающее правило *)
decisionrule [set1]
decisionrule [set2]
```

Набор данных приведен в прил. 4. Результат будет следующим:

```
Training set = 24
Test set 2 = 5
{1,1,1,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2}
{2,2,2,1,1}
2
{2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1}
{1,1,1,2,2}
```

Для кластеризации используется оператор ClusterClassify [].

В одном случае в клише он в виде ClusterClassify [set1]. В данном случае число кластеров не указано, поэтому программа разбила обучающее множество по своему усмотрению на три кластера.

В другом — в виде Cluster Classify [set1, classes = 2]. Поскольку classes = 2, то разбиение сделано на два кластера.

Глава 5.

РЕШЕНИЕ ЗАДАЧ РЕГРЕССИИ И КЛАССИФИКАЦИИ НА ОСНОВЕ PYTHON

5.1. Библиотеки и клише программ

Существуют различные программы решения задач МО. До сих пор мы в основном использовали Excel, Wolfram Mathematica или сводили решение к ЗМП. На наш взгляд, первые два инструмента легко воспринимаются практиками из конкретных областей знаний (медицинскими работниками, технологами, экономистами и т. п.), а навыки сведенияя к ЗМП позволяют им решать одним инструментарием задачи различной сложности без написания программ на языках высокого уровня. Конечно, такой подход страдает некоторой однобокостью.

Профессионалы в области машинного обучения обычно используют либо Python, либо язык R. Причем, на наш взгляд, Python в последнее время стал явно вырываться вперед по популярности среди дата-сайентистов. Перечислим основные библиотеки программ, написанных на Python для решения задач машинного обучения, и их возможности:

NumPy	Поддержка многомерных массивов и математических функций для работы с ними
Pandas	Группировка, комбинирование, фильтрация данных. Анализ временных рядов
Scikit-learn	Решение задач классическими алгоритмами МО
Matplotlib	Визуализация изображений

Поэтому, не вдаваясь в подробности языка Python, в прил. 10 приведем различные клише программ на этом языке. Сразу предупредим, что многие действия могли бы быть записаны более компактно.

Читатели, имеющие опыт программирования на Python, могут сделать это самостоятельно. Мы ориентировались на начинающих пользователей этого мощнейшего языка программирования. Поэтому для нас была более приоритетной понятность кода, а не его компактность. В клише достаточно много операторов превращены в комментарии, то есть начинаются со знака #. Эти операторы предназначены либо для отладки программы, либо для принятия решения после выполнения определенного блока. На начальном этапе # перед операторами можно убрать. Тогда будет легче отследить и понять последовательность действий. В определенный момент вы сами придете к выводу, нужны эти операторы вам или нет, и либо вернете #, либо исключите их из кода вообще.

Клише программ мы старались расположить в последовательности, которой следует придерживаться при решении большинства практических задач:

1. Отбор признаков исходя из их собственных свойств или вычислительных проблем, которые они могут создать.
2. Построение решающих правил различными методами.
3. Оценка качества решающих правил.
4. Оценка информативности признаков с точки зрения различных методов машинного обучения.

Для того чтобы воспользоваться данными программами, достаточно загрузить их в Пользователи из прил. 10, папка ПРОГРАММЫ. Данные для этих программ находятся в прил. 10 в папке ДАННЫЕ. Все эти данные рекомендуем скопировать в папку D:\ML, так как именно этот адрес указан в большинстве программ, в противном случае надо будет в программах просто изменить этот адрес на действительный.

5.2. Методы отбора, создания и нормализации признаков

Различные способы отбора признаков средствами Excel мы привели в гл. 2. К числу простых приемов исключения признаков средствами Excel, конечно, еще следует добавить исключение признаков, у которых величина дисперсии или величина, равная разнице между

максимальным и минимальным значением, меньше некоторого порогового значения. Логика здесь очень простая: если значение признака — одна и та же константа во всех наблюдениях, то этот признак явно лишний. Теперь если отклонения от некоторой константы незначительны, то это могут быть просто шумы, и их учет ничего, кроме проблем, не создаст.

Сейчас стало модным называть признаки на английский манер фичами (Feature). По этому поводу нам нравится высказывание крупного российского специалиста в области машинного обучения К. В. Воронцова: «Как только вы называете признаки фичами, знайте, что все многолетние труды российских ученых в области машинного обучения прошли зря» [4]. Поэтому мы будем давать русскоязычные названия, а в скобках — их английские синонимы.

Вообще работа с признаками предполагает как их отбор (Selection), так и их создание (Engineering), причем порядок следования действий может быть любой, в зависимости от ситуации. Оба подхода могут применяться как к входным, так и к выходным признакам.

Приемы создания признаков:

- 1) распространенный пример: при оценке стоимости недвижимости — замена длины и ширины здания на площадь;
- 2) превращение чисто категориальных признаков в балльные;
- 3) округление чисел или разбивка их на диапазоны;
- 4) разбивка дат на составляющие или превращение их в дни, годы, сезоны, периоды и т. п.;
- 5) извлечение дополнительной информации (например, по сокращениям Mr, Mrs и Miss можно легко извлечь половой признак);
- 6) добавление результатов других алгоритмов (например, разбить объекты на кластеры и добавить номер кластера);
- 7) агрегирование признаков (например, покупатель выбирает различные товары, но в определенном диапазоне — замена наименований на диапазон переведет категориальные признаки в количественные и сократит размерность).

Мы перечислили только те случаи, упоминания о которых встречались в статьях по машинному обучению. Конечно, создание признаков — это, скорее, искусство, а не наука, и рекомендовать что-то в конкретной ситуации достаточно сложно. На практике нам довелось столкнуться со следующим случаем. Необходимо было построить уравнение регрессии для прогнозирования качественных показателей агломерата в за-

висимости от технологических параметров. Результаты нас долгое время не устраивали, пока мы не сообразили, что два признака — скорость движения ленты агломашины и высоту спекаемого слоя — надо заменить на их произведение, то есть на объем шихты, спекаемой в единицу времени. Сразу увеличилась точность прогнозирования, и количество параметров в уравнении регрессии уменьшилось на один.

Понятно, что генерация новых признаков необходима в случаях, когда в исходном пространстве признаков не получается решить поставленную задачу. Рассмотрим причины избавления от признаков. Ранее мы уже упоминали, что следует избавляться от признаков, если они плохие формально или по сути, мультиколлинеарные, имеют нулевую или близкую к нулю дисперсию. Другие причины уменьшения числа признаков:

- 1) уменьшение ресурсоемкости и сложности вычислений;
- 2) увеличение скорости обучения;
- 3) упрощение и удешевление сбора информации в дальнейшем;
- 4) увеличение интерпретируемости результатов;
- 5) снижение риска переобучения.

Список можно продолжить. В данном случае хорошо подходит принцип бритвы Оккама: при примерно одинаковых результатах чем меньше признаков, тем лучше.

Вообще методы отбора признаков обычно делят на четыре категории: фильтры, обертки, встроенные и гибридные.

Фильтры — методы, основанные на теории вероятности и статистике (IG-индексирование, хи-квадрат, mRmR). Важность признаков оценивается без привлечения моделей обучения, только на основе их собственных характеристик. Достоинство — быстрота, недостаток — не учитывается влияние признаков друг на друга и на целевую переменную.

Обертки: оценивается финальный результат (обычно прирост точности решения) применения конкретного (любого) алгоритма обучения при использовании конкретного подмножества признаков. Существует два подхода: включение (Forward selection) и исключение (Backward selection) признаков. Методы включения начинают с одного признака и постепенно добавляют другие. В случае исключения метод стартует с некоторого исходного множества признаков, оценивает результат, удаляет малоинформативные признаки. Критерием продолжения и прекращения процесса включения или исключения признаков служит изменение качества решающего правила.

Встроенные методы: процессы обучения модели и отбора признаков совмещены (Lasso, гребневая регрессия, различные виды регуляризации).

Гибридные методы: фильтры и обертки объединяют в двухфазный процесс.

Таким образом, можно сказать, что оценка информативности признака зависит от его собственных свойств и от того, в каком методе машинного обучения он используется. Один и тот же признак может быть информативным для одного метода и неинформативным для другого.

Поскольку ряд методов машинного обучения чувствителен к величине признаков (метод ближайших соседей, нейросети), то для дальнейшей работы требуется нормализация признаков. Существуют различные методы нормализации. Программа для одного из них называется **НОРМАЛИЗАЦИЯ ДАННЫХ МИН=0 МАХ=1**.

Для отбора признаков различными методами можете воспользоваться программой **ОТБОР ПРИЗНАКОВ В ЗАДАЧАХ КЛАССИФИКАЦИИ**.

5.3. Краткое описание программ для задач регрессии

Исходные данные для этих задач были взяты из репозитория UCI, для удобства использования размещены в таблице FROM_UCI на отдельных листах.

1. **ЛИНЕЙНАЯ РЕГРЕССИЯ** (расчет коэффициентов корреляции признаков, удаление мультиколлинеарных, построение уравнения регрессии).

2. **СРАВНЕНИЕ РЕГРЕССОРОВ ОПРЕДЕЛЕНИЕ ВАЖНОСТИ ПРИЗНАКОВ** (**Регрессоры:** LinearRegression, LogisticRegression, SVR, KNeighborsRegressor, RandomForestRegressor. **Метрики качества:** mean_absolute_error, mean_squared_error, median_absolute_error, r2_score).

3. **СРАВНЕНИЕ РЕГУЛЯРИЗАТОРОВ** (**Регуляризаторы:** LinearRegression, HuberRegressor, Lasso, Ridge, BayesianRidge, ElasticNet. **Метрики качества:** mean_absolute_error, mean_squared_error, median_absolute_error, r2_score).

5.4. Сравнение качества классификаторов

Сравнение качества классификаторов будем делать на множествах ирисов Фишера. Поскольку среди классификаторов мы будем рассматривать и метод ближайших соседей, то работать будем с ранее нормализованными данными. Нам известно, что множество setosa линейно отделимо от остальных. Поэтому отделим множество setosa от остальных и оценим качество работы различных классификаторов по распознаванию данной ситуации. Для этого воспользуемся программой SETOSA, СРАВНЕНИЕ КЛАССИФИКАТОРОВ.

Классификаторы: LinearDiscriminantAnalysis (), # линейное разделение
SVC (kernel='linear'), # метод опорных векторов с линейным ядром
#LogisticRegression (), # логистическая регрессия
KNeighborsClassifier (n_neighbors=6), # метод ближайших соседей
DecisionTreeClassifier (), # дерево решений
RandomForestClassifier (n_estimators=10), # случайный лес
GaussianNB () # наивный байесовский метод

Метрики: accuracy_score# аккуратность
balanced_accuracy_score# сбалансированная аккуратность
precision_score# точность
roc_auc_score# площадь под ROC-кривой

По результатам решения мы видим, что в данном случае все качественные показатели равны единице, то есть все классификаторы безошибочно распознали ситуацию линейной отделимости. Поэтому далее мы оценим качество распознавания множеств versicolor и virginica. Для этого достаточно видоизменить только второй блок программ:

```
# рассматриваем данные без setosa
df=dataset.iloc [50:]
# делаем перекодировку
df.loc [df ['тип'] =='versicolor','тип'] =0
df.loc [df ['тип'] =='virginica','тип'] =1
```

Все остальные блоки можно оставить в прежнем виде (в прил. 10 мы приводим полный текст программы).

В этом случае большинство качественных показателей отличны от единицы, но все имеют достаточно высокое значение. Данный случай не стоит обобщать для всех возможных ситуаций. Просто сейчас у вас в руках есть инструментарий, при помощи которого вы сможете оценить качество классификаторов для различных наборов данных и различных значений параметров настроек классификаторов (kernel, n_neighbors, n_estimators и т. п.). С полным списком параметров настроек классификаторов вы можете самостоятельно познакомиться в документации к библиотеке Scikit-learn.

5.5. Использование других оптимизаторов для решения ЗМП

Существует большое число различных оптимизаторов. Достаточно подробные обзоры оптимизаторов приведены в [31; 32]. Поэтому мы не будем сравнивать их достоинства и недостатки. Лично мы, кроме IBM ILOC CPLEX, достаточно часто используем MIP и PULP. В данной главе приведем и объясним ряд клише для MIP (все они приведены в прил. 10). На их основе вы всегда сможете сделать программу, которая нужна именно вам. Пояснения к клише даны с максимально подробными комментариями в тексте программ. Поэтому в данной главе мы дадим только краткие дополнительные пояснения по использованию и изменению базовых клише.

В качестве базового клише следует использовать программу ЗМП с критерием MAE.

1. Если вы хотите использовать другую модель на тех же данных, то вам для этого достаточно видоизменить блоки, начиная с блока # Создаем модель. Например, если необходимо решение, в котором максимальная невязка будет минимальна по абсолютной величине (чебышевское приближение), то модель примет следующий вид:

```
# Создаем модель
model_regress_mm=Model ()
# Переменные
a= [model_regress_mm.add_var (var_type="C", lb=-float ('inf'), ub=float
('inf')) for i in I]
```

```
w= [model_regress_mm.add_var (var_type="C") for j in J]
v= [model_regress_mm.add_var (var_type="C") for j in J]
b=model_regress_mm.add_var (var_type="C", lb=-float ('inf')) # так как переменная входит во все ограничения, то #без квадратных скобок
ch=model_regress_mm.add_var (var_type="C") #чебышевский минимакс

# Условия
for j in J:
    model_regress_mm+=xsum (P [j] [i] *a [i] for i in I) ==Y [j] +w [j]-v [j] +b
    model_regress_mm+=w [j] <= ch
    model_regress_mm+=v [j] <= ch
# Целевая функция
model_regress_mm.objective=minimize (ch)
# Запуск модели на счет
status=model_regress_mm.optimize (max_seconds=1000)

# Выводим статус оптимизации
if  (status!=OptimizationStatus.NO SOLUTION FOUND)  and
(status!=OptimizationStatus.INFEASIBLE):
    res_a= []

    res_b= [b.x]
    res_w= []
    res_v= []
    res_ch= [ch.x]

    for i in I:
        res_a.append (a [i].x)

    for j in J:
        res_w.append (w [j].x)
        res_v.append (v [j].x)
        print ("a:", res_a)
        print ("b:", res_b)
        print ("w:", res_w)
        print ("v:", res_v)
        print ("ch:", res_ch)
```

```
else:  
    print ("Не разрешима")  
  
# Запись модели в файл (можно открыть в блокноте и сверить запись)  
#model_regress_mm.write ('model_mm.lp')  
#Сохраняем полученное решение  
#Важно! Таблица Excel "параметры_minmax.xlsx" должна быть закрыта, иначе будет ошибка!  
pd.DataFrame (res_a+res_b+res_ch).to_excel ('параметры_minmax.xlsx')  
#КОНЕЦ ПРОГРАММЫ
```

Естественно, что в результате расчетов вы получите другие значения переменных модели:

```
a: [-1.2160898035547263, 1.1945743685687549, 0.14686623012161046]  
b: [-240.02572497661404]  
w: [3.6927034611786818, 0.0, 0.0, 0.0, 3.0603367633302434,  
0.0, 3.2623947614593076, 8.40271281571571, 8.402712815715711,  
8.402712815715711, 0.0, 0.0]  
v: [0.0, 8.402712815715711, 3.995790458372405, 7.689897100093585,  
0.0, 8.40271281571571, 0.0, 0.0, 0.0, 0.0, 1.3372310570626738,  
5.163236669784939]  
ch: [8.40271281571571]
```

2. Если вы хотите использовать модель с минимизацией суммы непопаданий в заданный диапазон, то она будет выглядеть следующим образом:

```
# Создаем модель  
d=0.05#диапазон  
  
L=10000#штраф за непопадание в диапазон  
model_regress_d=Model ()  
  
# Переменные  
a=[model_regress_d.add_var (var_type="C", lb=-float ('inf'), ub=float ('inf'))  
for i in I]  
z=[model_regress_d.add_var (var_type="B") for j in J] #булевая переменная
```

```
b=model_regress_d.add_var (var_type="C", lb=-float ('inf')) # так как переменная входит во все ограничения, то без квадратных скобок
```

```
# Условия
for j in J:
    model_regress_d+=xsum (P [j] [i] *a [i] for i in I) <= (1+d) *Y [j] +L*z [j] +b
    model_regress_d+=xsum (P [j] [i] *a [i] for i in I) >= (1-d) *Y [j]-L*z [j] +b

# Целевая функция
model_regress_d.objective=minimize (xsum (z [j] for j in J))

# Запуск модели на счет
status=model_regress_d.optimize (max_seconds=1000)

# Выводим статус оптимизации
if (status!=OptimizationStatus.NO SOLUTION FOUND) and (status!=OptimizationStatus.INFEASIBLE):

    res_a=[]
    res_b=[b.x]
    res_z=[]

    for i in I:
        res_a.append (a [i].x)

    for j in J:
        res_z.append (int (z [j].x))

    print ("a:", res_a)
    print ("b:", res_b)
    print ("z:", res_z)

else:
    print ("Не разрешима")
```

Результаты расчетов по этой модели будут:

```
a: [-1.1012554585152847, 0.9686681222707438, 0.07576419213973769]
```

```
b: [241.29530567685592]
z: [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
# Запись модели в файл (можно открыть в блокноте и сверить запись)
model_regress_d.write ('model_D.lp')
#Сохраняем полученное решение
#Важно! Выводные таблицы Excel должны быть закрыты, иначе будет
ошибка!
pd.DataFrame (res_a+res_b).to_excel ('параметры_регрессии_диапазон.
xlsx')
pd.DataFrame (res_z).to_excel ('индикаторы_попадания_в_диапазон.xlsx')
#КОНЕЦ ПРОГРАММЫ
```

Заключение

На этом мы заканчиваем наше учебно-методическое пособие. Конечно, рассмотрены далеко не все ситуации, которые могут возникать при решении реальных задач, так как это просто невозможно. Для дальнейшего совершенствования навыков построения решающих правил рекомендуем следующее.

1. Обязательно внимательно изучите курсы других авторов, так как любые специалисты, в том числе и мы, подобны флюсу (однобоки). В силу образования, жизненного опыта, места работы и прочих факторов конкретные специалисты — как бы ни пытались быть объективными — все равно более подробно опишут одни аспекты машинного обучения и гораздо менее подробней — другие. Создайте в своей голове из различных курсов эффективный ансамбль, в котором достоинства каждого курса будут компенсировать недостатки других. Курсы и статьи, которые мы вам рекомендуем: [33–39].

2. Познакомьтесь с открытыми источниками информации и сформулируйте для себя задачу МО, которую вам было бы интересно решить самостоятельно. Открытых источников много, перечислим ряд из них:

- портал «Открытые данные России», <http://data.gov.ru>;
- конкурс «Открытые данные РФ», <http://opendatacontest.ru>;
- Росстат, <http://www.gks.ru/opendata>;
- центральный Банк РФ, <http://www.cbr.ru/statistics>;
- Министерство финансов РФ, <http://minfin.ru/opendata>;
- Министерство транспорта РФ, <http://www.mintrans.ru/opendata>;
- Министерство образования и науки, <http://открытые-данные.минобрнауки.рф/opendata>.

3. Обязательно участвуйте в различных конкурсах по МО. Это увлекательно, интересно и, самое главное, полезно. Приводим список известных нам конкурсов:

- www.NetflixPrize.com (2006–2009) — первый крупный конкурс;
- www.kaggle.com — самая известная платформа;

- DataRing.ru — отечественная конкурсная платформа;
- «Цифровой прорыв», <https://hacks-ai.ru/>.

Не расстраивайтесь, если не все будет получаться сразу. Успех приходит с опытом, а опыт появляется только при постоянном тренинге.

Успехов в увлекательном мире машинного обучения!

Тестовые задания

Предлагаем проверить полученные знания. На любой из вопросов может быть как один, так и несколько вариантов ответов.

Глава 1

1. К какому методу познания относится машинное обучение?
 - а) дедукция;
 - б) абстрагирование;
 - в) индукция;
 - г) анализ;
 - д) синтез.
2. Какие из приведенных ниже задач относятся к задачам обучения с учителем?
 - а) классификация;
 - б) кластеризация;
 - в) фильтрация;
 - г) регрессия;
 - д) сокращение размерности.
3. Модель называют переобученной, если:
 - а) в ней большое количество наблюдений;
 - б) в ней большое количество признаков;
 - в) средняя ошибка алгоритма на наблюдениях контрольной выборки значительно выше, чем на обучающей;
 - г) размер тестовой выборки значительно выше обучающей.

Глава 2

1. Какие признаки называют мультиколлинеарными?
 - а) имеющие одинаковое значение;
 - б) линейно зависимые;
 - в) сильно коррелирующие друг с другом;
 - г) имеющие низкую дисперсию.

2. Зависимость между входными и выходными признаками имеет полиномиальный характер. Как данную задачу свести к задаче линейной регрессии?

- а) попытаться аппроксимировать полином касательными;
- б) создать новые признаки на основе существующих;
- в) попытаться аппроксимировать полином отрезками, соединяющими точки полинома.

3. Кто из философов сказал: «Не следует множить сущее без необходимости»? То есть чем проще, тем лучше.

- а) Аристотель;
- б) Эпикур;
- в) Оккам;
- г) Байес.

Глава 3

1. Какие множества являются линейно не разделимыми?

- а) выпуклые;
- б) невыпуклые;
- в) не разделяемые гиперплоскостью;
- г) не разделяемые любой гиперповерхностью.

2. Какой из комитетов является универсальным?

- а) единогласия;
- б) большинства;
- в) старшинства;
- г) единодушия.

3. Возможно или нет в методе комитетов использовать нелинейные функции?

- а) да;
- б) нет.

Глава 4

1. В методе опорных векторов разделяющая гиперплоскость:

- а) опирается на одно из множеств;
- б) находится на одинаковом расстоянии от ближайших точек обоих множеств;
- в) находится на одинаковом расстоянии от всех точек обоих множеств.

2. Какие из перечисленных слов не имеют никакого отношения к названиям научных школ машинного обучения?
 - а) символизм;
 - б) абстракционизм;
 - в) коннекционизм;
 - г) эволюционизм;
 - д) экзистенциализм;
 - е) байесианизм;
 - ж) аналогизм;
 - з) композиционизм.
3. Какие из перечисленных методов являются ансамблевыми?
 - а) логистическая регрессия;
 - б) метод комитетов;
 - в) дерево решений;
 - г) наивный байесовский;
 - д) нейросети;
 - е) ближайших соседей;
 - ж) случайный лес;
 - з) потенциальных функций.

Глава 5

1. В каких моделях регрессии совмещено обучение модели и отбор признаков?
 - а) с критерием MAE;
 - б) с критерием MSE;
 - в) гребневая;
 - г) лассо.
2. Какие из методов можно использовать для решения задач регрессии?
 - а) ближайших соседей;
 - б) случайный лес;
 - в) нейросети.

Ответы к тестовым заданиям

Глава 1

1. в
2. а, г
3. в

Глава 3

1. в
2. в
3. а

Глава 5

1. в, г
2. а, б, в

Глава 2

1. в
2. б
3. в

Глава 4

1. б
2. б, д
3. б, д, е, ж

Список библиографических ссылок

1. Марр Б., Уорд М. Искусственный интеллект на практике. 50 кейсов успешных компаний. М. : Манн, Иванов и Фербер, 2020. 319 с.
2. Кай-Фу Ли. Сверхдержавы искусственного интеллекта. Китай, Кремниевая долина и новый мировой порядок. М. : Манн, Иванов и Фербер, 2019. 240 с.
3. Тополь Э. Искусственный интеллект в медицине. М. : Альпина Паблишер, 2022. 398 с.
4. Воронцов К. В. Машинальное обучение : [курс лекций] // MachineLearning.ru : проф. информ.-аналит. ресурс. URL: <https://clck.ru/JF9R> (дата обращения: 30.08.2022).
5. «Это имитация интеллекта»: Константин Воронцов — о настоящем и будущем машинного обучения // Системный Блокъ : сайт. URL: <https://clck.ru/36o5pQ> (дата обращения: 30.08.2022).
6. Шваб К. Четвертая промышленная революция. М. : Эксмо, 2016. 138 с.
7. Путин: монополист в сфере искусственного интеллекта может стать властелином мира // ТАСС : информ. агентство России : сайт. URL: https://tass.ru/ekonomika/6489864?utm_source=google.com&utm_medium=organic&utm_campaign=google.com&utm_referrer=google.com (дата обращения: 30.08.2022).
8. Си Цзиньпин: технологии искусственного интеллекта станут новым драйвером экономики Китая // ТАСС : информ. агентство России : сайт. URL: <https://tass.ru/ekonomika/5745558> (дата обращения: 30.08.2022).
9. Ковачич Л. Зачем Трамп вступил в войну с китайским искусственным интеллектом // Фонд Карнеги за международный мир : сайт. URL: <https://carnegiemoscow.org/commentary/76102> (дата обращения: 30.08.2022).
10. 7 декабря 2016 года в Москве состоялся III Конгресс «Иновационная практика: наука плюс бизнес» // Иннопрактика : сайт. URL: <https://innopraktika.ru/news/688/> (дата обращения: 30.08.2022).

11. Новые тренды в сфере ИИ и машинного обучения в этом году // IT Channel News : сайт. URL: <https://www.novostiitkanala.ru/news/detail.php?ID=143082> (дата обращения: 30.08.2022).
12. Чернавин Ф. П. Применение метода комитетов для решения задач классификации // Российские регионы в фокусе перемен : сб. материалов XII Междунар. конф. Екатеринбург : Изд-во УМЦ УПИ, 2018. С. 437–447.
13. Чернавин Ф. П. Применение нейронных сетей к задачам оценки вероятности дефолта по потребительским кредитам // Журнал научных публикаций аспирантов и докторантов. 2013. № 7. С. 21–25.
14. Чернавин Ф. П. Применение комитетных конструкций для принятия решений по потребительским кредитам // Экономика и предпринимательство. 2015. № 12. С. 14–149.
15. Чернавин Н. П., Чернавин Ф. П. Применение метода комитетов к прогнозированию движения фондовых индексов. М. : РусАльянс СОВА, 2015. С. 307–320.
16. Чернавин Н. П., Чернавин Ф. П. Применение метода комитетов в техническом анализе инструментов финансовых рынков // Современные научные исследования в сфере экономики : сб. результатов науч. исследований. Киров : Изд-во МЦИТО, 2018. С. 1052–1062.
17. Чернавин П. Ф., Чернавин Ф. П., Чернавин Н. П. Сведение задач регрессии к задачам линейного целочисленного программирования // Анализ, моделирование, управление, развитие социально-экономических систем : сб. науч. трудов XIV Всероссийской с международным участием школы-симпозиума АМУР-2020. Симферополь, 2020. С. 383–386.
18. Анализ и прогнозирование выхода годного и прочности агломерата на основе модели математического программирования / П. Ф. Чернавин [и др.] // Черные металлы. 2021. № 12. С. 20–24.
19. Имитационная модель подбора технологических параметров для получения агломерата с высокими потребительскими свойствами на основе метода комитетов / П. Ф. Чернавин [и др.] // Черные металлы. 2022. № 3. С. 10–14.
20. Машинное обучение на основе задач математического программирования / П. Ф. Чернавин [и др.]. М. : Наука, 2021, 128 с.
21. Саймон Х. Нейронные сети: полный курс. М. : Вильямс, 2006. 1104 с.
22. Гасс С. Линейное программирование (методы и приложения). М. : ФИЗМАТЛИТ, 1961. 304 с.

23. Данциг Дж. Б. Линейное программирование, его применения и обобщения. М. : Прогресс, 1963. 590 с.
24. Мазуров В.Д. Математические методы распознавания образов : учеб. пособие. 2-е изд., перераб. и доп. Екатеринбург : Изд-во Урал. ун-та, 2010. 101 с.
25. Вапник В. Н. Восстановление зависимостей по эмпирическим данным. М. : Наука, 1979. 447 с.
26. Домингос П. Верховный алгоритм. М. : Манн, Иванов и Фербер, 2016. 336 с.
27. Юдковски Э. Наглядное объяснение теоремы Байеса // LessWrong.ru : сайт. URL: https://lesswrong.ru/w/Наглядное_объяснение_теоремы_Байеса (дата обращения: 30.08.2022).
28. Нейронные сети, перцептрон // Викиконспекты : сайт. URL: https://neerc.ifmo.ru/wiki/index.php?title=Нейронные_сети,_перцептрон (дата обращения: 30.08.2022).
29. Модель Маккалока-Питса // MachineLearning.ru : проф. информ.-аналит. ресурс. URL: http://www.machinelearning.ru/wiki/index.php?title=Модель_Маккалока-Питса (дата обращения: 30.08.2022).
30. Переобучение // Википедия : сайт. URL: <https://ru.wikipedia.org/wiki/Переобучение> (дата обращения: 30.08.2022).
31. Comparison of Open-Source Linear Programming Solvers / Gerarhart J. L. [et al.] // SANDIA REPORT. 2013. P. 62.
32. A review and comparison of solvers for convex MINLP / Kronqvist J. [et al.] // Optimization and Engineering. 2019. Vol. 20. P. 397–455.
33. Минский М., Пейпарт С. Персептроны. М. : Мир, 1971. 262 с.
34. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. М. : ДМК Пресс, 2015. 400 с.
35. Базилевский М. П., Носков С. И. Формализация задачи построения линейно-мультиплекативной регрессии в виде задачи частично-булевого линейного программирования // Современные технологии. Системный анализ. Моделирование. 2017. Т. 55, № 4. С. 101–105.
36. Базилевский М. П. Сведение задачи отбора информативных регрессоров при оценивании линейной регрессионной модели по методу наименьших квадратов к задаче частично-булевого линейного программирования // Моделирование, оптимизация и информационные технологии. 2018. Т. 6, № 1 (20). С. 108–117.

37. Горелик В. А., Трембачева О. С. Решение задачи линейной регрессии с использованием методов матричной коррекции в метрике // Журнал вычислительной математики и математической физики. 2016. Т. 56, № 2. С. 202–207.
38. Чемезов Д. С., Тюкачев Н. А. Обзор основных методов классификации и кластеризации // Вестник ВГУ. 2009. № 2. С. 25–29.
39. Мандель И. Д. Кластерный анализ. М. : Финансы и статистика, 1988. 176 с.

Учебное издание

**Чернавин Павел Федорович
Чернавин Николай Павлович
Чернавин Федор Павлович**

**ПРАКТИЧЕСКИЙ КУРС
КЛАССИЧЕСКОГО МАШИННОГО ОБУЧЕНИЯ
С ИСПОЛЬЗОВАНИЕМ МОДЕЛЕЙ
МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ**

Редактор *T. E. Мериц*
Верстка *Ю. В. Ершовой*

Подписано в печать 05.12.2023. Формат 70×100 1/16.
Бумага офсетная. Цифровая печать. Усл. печ. л. 10,0.
Уч.-изд. л. 8,4. Тираж 30 экз. Заказ 168.

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ ЦСД
620049, Екатеринбург, ул. С. Ковалевской, 5
Тел.: 8 (343) 375-48-25, 375-46-85, 374-19-41
E-mail: rio@urfu.ru

Отпечатано в ИПЦ УрФУ ЦСД
620083, Екатеринбург, ул. Тургенева, 4
Тел.: 8 (343) 358-93-06, 350-58-20, 350-90-13
Факс: 8 (343) 358-93-06
<http://print.urfu.ru>



9 785799 637675

A standard linear barcode is positioned above a series of numbers. The numbers are: 9, 785799, and 637675.