Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Южно-Уральский государственный университет
Кафедра «Прикладная математика»

681.327(07)
К885

Б.М. Кувшинов

РАСПОЗНАВАНИЕ ОБРАЗОВ

Учебное пособие

Челябинск
Издательство ЮУрГУ
2008

УДК 519.71(075.8)+681.327.12(075.8)
К885

Одобрено учебно-методической комиссией
механико-математического факультета

Рецензенты:
В.И. Ухоботов, Е.М. Сартасов

Кувшинов, Б.М.
К885 Распознавание образов: учебное пособие / Б.М. Кувшинов. – Челябинск: Изд-во ЮУрГУ, 2008. – 55 с.

В учебном пособии рассматриваются математические методы распознавания образов, вопросы применения этих методов при решении задач машинного анализа знаний и поддержки принятия решений с помощью вычислительной техники.

Рассмотрены существующие постановки задачи распознавания и эвристические принципы, на которых построены методы распознавания образов. Подробно рассматривается реализация этих принципов в рамках наиболее распространенных подходов к решению задач распознавания: детерминированных линейных и нелинейных классификаторов, вероятностных классификаторов. Обсуждаются вопросы предварительной обработки данных, позволяющих свести задачи выбора, прогнозирования, поддержки принятия решений к задачам распознавания образов.

Пособие предназначено для студентов, обучающихся по специальностям кафедры «Прикладная математика». Материал пособия может также использоваться студентами других специальностей, изучающими вопросы применения методов распознавания образов в решении задач поддержки принятия решений.

УДК 519.71(075.8)+681.327.12(075.8)

ВВЕДЕНИЕ

Вводные замечания

1. Это не визуальные образы! Анализ изображений часто упоминается рядом с РО, но это лишь частный случай, причем весьма специфический. РО – подход к численному решению прикладных задач, по уровню аналогичный, например, имитационному моделированию.

2. В двух словах: распознавание образов – это один из принципиальных подходов к использованию разнообразного математического аппарата в решении реальных задач (с реальными исходными данными и предположениями о происходящих процессах, в постановках, которые реально интересуют ЛПР).

Т.е. линейная алгебра, методы оптимизации, теория вероятностей, комбинаторика, теория графов, функциональный анализ, алгебра логики и т.д. – это набор математических инструментов для работы над абстрактными объектами в абстрактных постановках задачи (в соответствующих курсах технология перехода от реальной задачи к математической умалчивалась), а РО – способ применения этих инструментов в реальных задачах (прогнозирование, выбор наилучшего решения, оценка ситуации и т.п.)

3. Аналог курса ИИ (на 5 курсе 2204ф, 0102) – это разные аспекты задач ***обнаружения закономерностей (data mining)***, но не «в картинках» (в структурных схемах), а «в формулах».

4. План действий на лекциях: сначала (в главе 1) кратко рассмотрим возникающие постановки задач, идеи, лежащие в основе их решения и примеры прикладных задач из разных областей, которые сводятся к задачам РО. Затем начнем более подробно разбирать отдельные методы – сколько успеем в рамках лекций (1+0,5 пары) – я читаю первый раз.

5. Лабораторные (предположительно): реализация отдельных (простых) методов РО на Matlab. Не в готовом пакете – чтобы были ясны механизмы. Исходные данные – от соседа по бригаде (чтобы разгадать его замысел).

Литература

Отечественных много, но все стараются подогнать многообразие возникающих задач под свой подход.

Иностранные (на английском – свежие, на русском – старые) – есть общепризнанные стандартные книги.

1) Duda R.O., Hart P.E., Stork D.G. Pattern classification. N.Y.: Wiley Intersciences, 2002. 680 p. – выложу в «Учебные материалы\Книги\РО»
2) Ту Дж., Гонсалес Р. Принципы распознавания образов /Пер. с англ. /Под ред.Ю.И. Журавлева.-М.:Мир,1978. 411с.
3) Фор А. Восприятие и распознавание образов /Пер. с фр. /Под ред. Г.П. Катыса. М.: Машиностроение, 1989. 272 с.
4) Горелик А.Л., Скрипкин В.А. Методы распознавания. М.: Высшая школа, 1989. 222 с. – только вероятностные
5) Серия сборников. Распознавание. Классификация. Прогноз. Вып. 1–4. М.: Наука, 1989–1994. – разные наши подходы.

Глава 1. ЗАДАЧИ ОБУЧЕНИЯ ПО ПРЕЦЕДЕНТАМ

1.1. Основные понятия и постановка задачи распознавания образов

Классическое и информационное моделирование

Особенность многих прикладных задач – невозможность использования адекватной математической модели. Это может быть вызвано следующими причинами:

– природа процессов, которые подвергаются анализу, неизвестна;

– адекватный процессам вид модели известен, но эта модель слишком сложна с точки зрения вычислительной сложности, требует использования данных (измерений), которые сложно получить, требует слишком большой точности измерения исходных данных и т.д.

Эта ситуация возникает регулярно в таких трудно формализуемых областях, как медицина, геология, психология, социология, экономика. Даже в технических системах создание и применение классической физической модели зачастую оказывается невозможным, несмотря на полное знание внутреннего устройства системы.

На практике часто можно отказаться от всестороннего изучения системы или явления. Основная цель – научиться регулярно решать небольшой круг задач, связанных с прогнозированием и принятием управленческих решений в этой системе. В таких случаях имеет смысл моделировать не саму систему, а лишь некоторые ее проявления с точки зрения взаимодействия с лицом, принимающим решения, и с внешней средой.

Вообще, процесс построения математических моделей в прикладных задачах можно разделить на два этапа.

Первый этап – формализация экспертных знаний о предметной области, в результате которой формируется структура модели. При построении классических («физических») моделей этот этап наиболее важен. В хорошей физической модели остается, как правило, небольшое число свободных параметров, имеющих четкую содержательную интерпретацию. Эти модели узко специализированы и имеют фиксированные границы применимости.

Второй этап – настройка (идентификация) параметров модели по эмпирическим (экспериментальным) данным. Он существенно более важен для информационных моделей, когда опереться можно только на данные, поскольку суть процессов неясна. Информационные модели имеют значительно больше свободных параметров, зачастую не поддающихся содержательной интерпретации. Однако конкретная информационная модель как правило применима к достаточно широкому классу прикладных задач.

Одна и та же информационная модель, например, нейронная сеть, может быть использована для решения совершенно разных прикладных задач: классификации заемщиков банка и диагностики болезни. Многие информационные модели ис-

пользуют идеологию «чёрных ящиков» – они способны неплохо решать практические задачи, но внутренняя логика этих решений остается неизвестной даже для экспертов в данной прикладной области.

Четкого различия между физическими и информационными моделями нет. Чем больше знаний о предметной области удаётся привлечь на первом этапе построения модели, тем более она физична.

Основные понятия распознавания образов

Распознавание образов – один из подходов к построению информационных моделей. Основные положения этого подхода задаются следующим соотношением понятий.

Пусть имеются множество ***объектов*** *X*, множество ***ответов*** *Y*, и существует ***целевая функция*** $y^* : X \to Y$, значения которой $y_i = y^*(x_i)$ известны только на конечном подмножестве объектов $\{x_1,...,x_K\} \subset X$. Пары «объект–ответ» (x_i, y_i) называются ***прецедентами***. Совокупность пар $X^K = \{(x_i, y_i), i = \overline{1,K}\}$ называется ***обучающей выборкой***.

Задача распознавания образов заключается в том, чтобы восстановить функциональную зависимость между объектами и ответами, то есть построить отображение $a : X \to Y$, удовлетворяющее следующей совокупности требований.

1. Отображение *a* должно допускать эффективную численную реализацию. По этой причине будем называть его ***алгоритмом***.
2. Алгоритм *a(x)* должен воспроизводить на объектах выборки заданные ответы: $a(x_i) = y_i, i = \overline{1,K}$. Равенство здесь может пониматься как точное или как приближенное, в зависимости от конкретной задачи.
3. На алгоритм *a(x)* могут накладываться разного рода априорные ограничения, например, требования непрерывности, гладкости, монотонности, и т. д., или их сочетание. В некоторых случаях может задаваться модель алгоритма, т.е. функциональный вид отображения *a(x)*, определенный с точностью до набора параметров.
4. Алгоритм должен обладать обобщающей способностью, то есть достаточно точно приближать целевую функцию *y*(x)* не только на объектах обучающей выборки, но и на всем множестве *X*.

Это весьма неформальная постановка задачи: основные понятия нуждаются в уточнении, а некоторые из обозначенных выше требований можно ослабить, как это будет показано далее.

Простейшая постановка задачи распознавания

Сначала уточним понятие алгоритма в рамках задачи распознавания образов. ***Моделью алгоритмов*** называется параметрическое семейство отображений *A*, из которого выбирается искомый алгоритм:

$$A = \{a(x,\gamma) \mid a : X \times \tilde{A} \to Y\},$$

где Γ – множество допустимых значений параметра γ.

γ может представлять целый вектор параметров $\gamma = (\gamma_1,...,\gamma_L)$, поэтому Γ в общем случае называется ***пространством решений***.

Процесс подбора параметров модели по обучающей выборке называют ***настройкой*** или ***обучением*** алгоритма. В результате настройки выбирается единственный алгоритм $a \in A$, который должен приближать целевую зависимость y^*.

Замечание. В английской терминологии четко различается, что алгоритм является обучаемым, учеником (learning machine), а выборка данных – обучающей, учителем (training sample).

Методом обучения называется отображение μ, которое произвольной конечной выборке X^K ставит в соответствие конкретный алгоритм $a \in A$, т.е. подбирает значения параметров.

Итак, задача распознавания образов включает в себя две основные подзадачи:

– на этапе обучения метод μ по выборке X^K строит алгоритм $a = \mu(X^K)$;

– на этапе применения построенному алгоритму a подаются на вход новые объекты x, в общем случае отличные от обучающих; по этим объектам он должен выдать ответы $y = a(x)$.

Этап обучения наиболее сложен. Как правило, он сводится к поиску параметров модели, доставляющих оптимальное значение заданному *функционалу качества* (см. ниже).

Разновидности постановки задачи распознавания

В зависимости от природы множества объектов X и множества ответов Y задачи распознавании образов делятся на следующие типы.

1. В задаче ***классификации*** на M непересекающихся классов $Y = \{1,...,M\}$ и требуется построить алгоритм, относящий любой объект $x \in X$ к одному из этих классов. Классы называют также ***образами (pattern)***, откуда и происходит термин ***распознавание образов (pattern recognition)***.

2. Задача ***кластеризации*** на M непересекающихся классов отличается от задачи классификации тем, что в ответы $y_i = y^*(x_i)$ заранее не известны. Известны только сами объекты x_i, и требуется разбить выборку X^K на непересекающиеся подмножества ***(кластеры)***, т.е. сгенерировать ответы $y_i(x_i), i = \overline{1,K}$ так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались. Число кластеров может быть известно заранее, но чаще требуется определить и его. Механизм обучения здесь заменяется на механизм ***самообучения***.

3. В задаче классификации или кластеризации на M пересекающихся классов $Y = \{0,1\}^M$, т.е. каждый объект может принадлежать сразу к нескольким классам.

4. $Y = R^N$ – задача ***восстановления регрессии***. Задачи классификации на непересекающиеся классы являются частным случаем задач восстановления регрессии. Методы их решения существенно отличаются от общего случая, поэтому они и выделяются в самостоятельный класс.

5. Задача ***прогнозирования*** также является частным случаем задачи классификации или восстановления регрессии. В данном случае X – описание прошлого поведения объекта, Y – описание некоторых характеристик его будущего поведения.

Замечание. При рассмотрении алгоритмов классификации часто ограничиваются случаем двух классов, более удобным для теоретического анализа. Существует несколько стандартных способов свести задачу с M классами к задаче с двумя классами.

Признаковое описание объектов

Признаком называется отображение $f : X \to D_f$, описывающее результат измерения некоторой характеристики объекта, где D_f – заданное множество.

В зависимости от множества допустимых значений D_f признаки делятся на следующие типы:

– бинарный признак: $D_f = \{0,1\}$ (например, наличие или отсутствие какого-либо свойства);

– номинальный признак: D_f – конечное множество, порядок на котором не определен (например, цвет);

– порядковый признак: D_f – конечное упорядоченное множество (например, размер – маленький, средний или большой);

– количественный признак: D_f – множество вещественных чисел.

В прикладных задачах встречаются и более сложные случаи: значениями признаков могут быть числовые последовательности, растровые изображения, функции, результаты запросов к базе данных и т. д.

Пусть имеется набор признаков $f_1,...,f_N$. Вектор $(f_1(x),...,f_N(x))$ называют ***признаковым описанием объекта*** x. В дальнейшем не будем различать объекты из X и их признаковые описания, полагая $X = D_{f1} \times ... \times D_{fN}$. Совокупность признаковых описаний всех объектов выборки X^K, записанную в виде таблицы размером $K \times N$, называют ***матрицей объектов–признаков***:

$$F = \begin{pmatrix} f_1(x_1) \dots f_N(x_1) \\ \dots \\ f_1(x_K) \dots f_N(x_K) \end{pmatrix}.$$

Матрица объектов–признаков является стандартным и наиболее распространенным способом представления исходных данных. Однако на практике встречаются задачи, в которых данные устроены сложнее: например, описания объектов могут иметь переменную длину. В таких случаях часто предпочитают привести исходные данные к стандартному виду, вычисляя новые характеристики объектов по исходным описаниям. Таким образом, признаки в задаче распознавания – это характеристики объектов, которые либо измеряются непосредственно, либо вычисляются по «сырым» исходным данным. Преобразование исходных измеряе-

мых признаков к стандартному виду, позволяющему сформировать матрицу объектов-признаков, называется ***синтезом признаков***.

Замечание. Любое отображение из множества X можно также рассматривать как признак, в частности, любой алгоритм $a: X \to Y$, полученный в результате решения задачи распознавания также можно рассматривать как признак для задачи распознавания более высокого уровня.

Задачи классификации и кластеризации имеют наглядную геометрическую интерпретацию для случая двумерного пространства количественных признаков (рис. 1.1). Каждому объекту обучающей выборки можно поставить в соответствие точку на плоскости с координатами, соответствующими значениям признаков. Объекты, относящиеся к одному классу, занимают на этой плоскости некоторую область, которой соответствует один из распознаваемых образов. Алгоритм распознавания каким-либо способом описывает эту область в рамках выбранной модели алгоритмов.

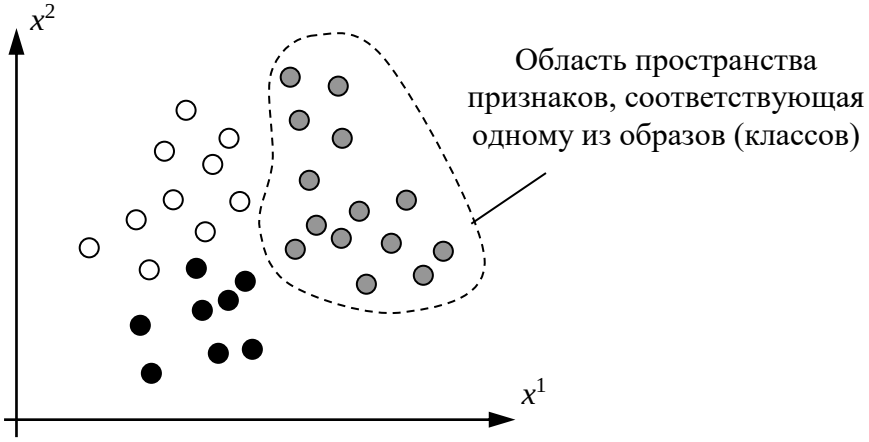


Рис. 1.1. Геометрическая интерпретация задач классификации и кластеризации для случая $X=R^2$

Вероятностная постановка задачи

До сих пор рассматривалась постановка задачи, в которой упускается из виду, что элементы множества X – это не реальные объекты, а лишь их описания, содержащие доступную часть информации об объектах. Полные описания практически никогда не бывают известны: невозможно исчерпывающим образом охарактеризовать, например, человека, геологический район, производственное предприятие или экономику страны. Поэтому одному и тому же описанию x могут соответствовать различные объекты, а, значит, и целое «облако значений» $y^*(x)$.

Для формализации этих соображений вводится вероятностная постановка задачи. Теперь вместо существования неизвестной целевой функции $y^*(x)$ будем предполагать существование неизвестного вероятностного распределения $p(x,y)$ на множестве $X \times Y$, согласно которому сгенерирована выборка пар $X^K = (\{ x_i, y_i), i = \overline{1,K} \}$.

Вероятностная постановка задачи считается более общей, так как функциональную зависимость $y^*(x)$ можно представить в виде вероятностного распределения

$$p(x,y) = p(x)p(y|x),$$

положив $p(y|x) = \delta(y - y^*(x))$, где $\delta(z)$ – дельта-функция.

Однако при этом приходится вводить дополнительную гипотезу о существовании на множестве X неизвестного распределения $p(x)$. Функциональная постановка задачи, вообще говоря, не связана с вероятностными предположениями, поэтому называть ее частным случаем вероятностной не вполне корректно. Адекватна ли гипотеза о существовании распределений $p(x)$ и $p(y|x)$ практическому опыту – вопрос скорее философский, и здесь он останется за рамками обсуждения.

1.2. Примеры прикладных задач распознавания

Исторически одними из первых задач, в решении которых стали применяться методы распознавания образов, были задачи медицинской и технической диагностики. В последнее время количество приложений методов распознавания (технических, экономических, ИТ, медицинских) стремительно возрастает. В данном параграфе отмечены некоторые из приложений, для которых методы распознавания успели зарекомендовать себя в качестве эффективного инструмента поддержки принятия решений.

При рассмотрении каждого примера будем выделять четыре основных момента:

– что является объектами (в терминах задачи распознавания образов)?
– что является признаками, описывающими объект?
– откуда берутся объекты для формирования обучающей выборки?
– как формулируется задача распознавания (в частности, в задачах классификации – что является классами)?

Пример 1. Задачи медицинской диагностики. В роли объектов в данном случае выступают пациенты. Признаки характеризуют результаты обследований, симптомы заболевания и применявшиеся методы лечения. Примеры бинарных признаков: пол, наличие симптомов; порядковых признаков: тяжесть состояния (удовлетворительное, средней тяжести, тяжелое, крайне тяжелое); количественных признаков: возраст, пульс, артериальное давление, доза препарата.

Обучающая выборка формируется по результатам детального обследования пациентов квалифицированным экспертом, а также по фактическим данным о дальнейшем развитии болезни.

Возможные постановки задачи: только по значениям признаков (без привлечения эксперта) классифицировать вид заболевания (дифференциальная диагностика); определять наиболее целесообразный способ лечения; предсказывать длительность и исход заболевания; оценивать риск осложнений.

Пример 2. Задачи технической диагностики при управлении сложными техническими комплексами. В задачах этого типа важно своевременно обнаруживать предаварийные ситуации, чтобы остановить технологический процесс и произвести ремонт или перенастройку оборудования. Признаки в таких задачах – это показания датчиков технологических параметров. В роли объектов для задачи распознавания выступают моменты времени, в которые производится регистрация показаний датчиков (для одного и того же технического объекта).

Задача классификации – своевременное обнаружение ситуаций, требующих изменения режима функционирования объекта. Обучающая выборка – те ситуации, которые были выявлены и правильно классифицированы экспертами.

Пример 3. Задача предсказания свойств конечной продукции по свойствам исходных компонент и параметрам технологического процесса. Современные химические, металлургические, текстильные и другие производства настолько сложны, что построить достаточно точную математическую модель зависимости качества продукции от множества влияющих на него факторов практически невозможно – поэтому для оценки качества конечной продукции при использовании тех или иных вариантов ее изготовления используются методы распознавания образов.

Объекты в данном случае – это партии продукции, которые можно получить при разных условиях изготовления и свойствах исходных материалов. Признаками являются параметры исходных компонент и технологического процесса. Обучающую выборку формируют экспериментальные партии продукции, изготовленные при фиксированных технологических условиях.

Задача распознавания состоит в том, чтобы оценить значение показателей качества готовой продукции (по потребительским свойствам и по затратам на производство) для разных наборов технологических условий. На основе этой задачи можно поставить и более сложную – задачу оптимизации. В этом случае необходимо выбрать параметры, которые обеспечат оптимальные качества товаров.

Пример 4. Классификация месторождений полезных ископаемых. Объектами в такой задаче являются разные территории, на которых ищутся полезные ископаемые. Признаки, описывающие состояние объектов – это данные геологической разведки: наличие или отсутствие разных пород на территории района (бинарные признаки), свойства этих пород (количественные или качественные признаки).

Обучающая выборка составляется из прецедентов двух классов: районов известных месторождений и районов, в которых интересующее ископаемое обнаружено не было. Задача состоит в том, чтобы обнаружить новые районы с залежами полезных ископаемых по данным геологической разведки.

Основная сложность, возникающая при решении этой задачи, заключается в том, что количество объектов может оказаться намного меньше, чем количество признаков. В этой ситуации классические статистические методы неработоспособны, и задача решается путем поиска скрытых логических закономерностей в

имеющемся массиве данных. В процессе решения выделяются короткие наборы признаков, обладающие наибольшей информативностью. По аналогии с медицинскими задачами, можно сказать, что с использованием методов распознавания отыскиваются «синдромы» месторождений.

Пример 5. Автоматическое распознавание спама. В роли объектов выступают сообщения, поступающие на электронную почту. Задача состоит в том, чтобы разделить поток сообщений на два класса: обычные и нежелательные (спам) – выявив представления конкретного пользователя об этих категориях. Обучающую выборку формируют те сообщения, которые конкретный пользователь классифицировал самостоятельно, тем самым неявно задав свои предпочтения.

Естественного исходного признакового описания, в отличие от предыдущих задач, здесь не существует. Поэтому в качестве признаков приходится использовать различные статистические характеристики поступающих сообщений. Примеры бинарных признаков – наличие определенного ключевого слова или фразы в тексте письма; количественных – размер письма, размер вложения, число адресатов в рассылке, число ранее полученных писем от данного отправителя.

Сложность состоит в том, что система распознавания должна самостоятельно генерировать признаки, не имея того багажа общих знаний, благодаря которому человек легко классифицирует письма и текстовые документы.

Пример 6. Рубрикация текстовых документов. Эту задачу можно рассматривать как обобщение предыдущей. Она возникает при работе с большим количеством текстовых документов, особенно, если они собираются и используются большим коллективом людей, например, в рамках компании. В некоторых случаях от среднего времени поиска документов зависит эффективность работы всей компании.

Основная сложность задачи в том, что каждому сотруднику нужна своя рубрикация обрабатываемых документов, причем, возможно, не единственная. В результате создается единое хранилище документов, снабженное множеством персональных рубрикаторов. При этом возникает потребность автоматизировать процесс раскладывания и поиска документов, для обеспечения которой и применяются методы распознавания.

Объектами для задачи распознавания являются документы, с которыми работают в компании. Обучающий прецедент создается пользователем в тот момент, когда он самостоятельно помещает документ в нужную рубрику.

Первая задача – автоматическое помещение нового документа в нужную рубрику. Алгоритм классификации должен уловить логику пользователя и в дальнейшем классифицировать тексты по тем же принципам (при этом от самого пользователя не требуется в явном виде формулировать эти принципы, более того, он может даже не подозревать об их существовании, пользуясь ими интуитивно). Сложность задачи состоит в том, что классов-рубрик очень много и они могут пересекаться. Роль признаков здесь, как правило, выполняют ключевые слова.

Документы считаются схожими, если множества их ключевых слов существенно пересекаются.

Следующая задача в рамках рубрикации – поиск документов. Запрос пользователя при этом, как правило, имеет вид: найти все документы, похожие на данный. В этом случае имеется два класса объектов: подходящие документы и остальные документы. Правильные ответы для объектов обучающей выборки в этом случае уже неизвестны, т.е. речь идет о задаче кластеризации (самообучения).

Пример 7. Задача распознавания рукописного текста. Такая задача возникает при автоматической обработке бланков, заполняемых от руки (бланки переписи населения, налоговые декларации и т.д). В роли классов выступают допустимые символы.

Первая задача – распознать символы по их изображению. Обучающая выборка – это тексты, к которым прилагается распознанный человеком электронный вариант. Первоначальными признаками является сканированное растровое изображение символа. Как правило, по растру строится контур изображения в виде ломаной линии, и в качестве признаков используются характеристики этой ломаной (положение, наклон, длина каждого звена).

Близкая задача – идентификация подписей, которая может решаться в двух постановках:

1) off-line идентификация – по растровому изображению подписи;

2) on-line идентификация, при которой подпись вводится с помощью электронного пера и в качестве исходных данных, помимо собственно изображения, выступают характеристики динамики движения пера.

Существенно более сложная задача из той же области – распознавание слитной речи. Для ее решения методы распознавания образов необходимо применять в комплексе со знаниями из акустики, фонетики, теории марковских цепей.

Пример 8. Задача оценивания надежности заемщиков при выдаче кредитов. В роли объектов выступают заемщики, претендующие на получение кредита. Признаки формируются на основе анкетные данные заявителей и дополнительная информации о них, которую собирает кредитующая организация (банк). Примеры бинарных признаков – пол, наличие телефона; номинальных – место проживания, профессия, работодатель; порядковых – образование, занимаемая должность; количественных – возраст, стаж работы, доход семьи, размер текущих обязательств по кредитам, сумма кредита.

Обучающую выборку формируют данные о заемщиках с известной кредитной историей. Задача в простейшем случае состоит в классификации заемщиков на два класса: тех, которым следует выдать кредит и тех, которым следует отказать в выдаче кредита. Более сложный вариант постановки задачи – оценить уровень надежности заемщика (задача кредитного скоринга). Сложность – в обучающей выборке класс ненадежных имеет гораздо меньшую долю, чем в тестовой (т.к. нет экспериментальных данных о клиентах, которым отказали в выдаче кредита)

Пример 9. Задача оценивания привлекательности инвестиционных проектов. Эта задача схожа с предыдущей и возникает у инвестиционных компаний, банков, государственных структур, которые финансируют исследовательские, инновационные и рискованные бизнес-проекты. Объектами для распознавания являются заявки на проекты, поданные участниками конкурса. Признаки формируются на основе содержания этих заявок. Как и в других задачах анализа текстов (обнаружения спама, рубрикации), наиболее сложным здесь является этап предварительной обработки данных, в ходе которого создаются числовые признаки заявки.

Основная сложность – низкая степень формализованности заявок. Наиболее важная с точки зрения экспертов информация заключается, как правило, в нескольких текстовых полях, содержательно описывающих суть проекта. Полная автоматизация процесса рассмотрения заявок невозможна, однако с помощью методов распознавания образов можно решить задачу предварительного отсева заведомо плохих заявок, на рассмотрение которых экспертам нецелесообразно тратить время. Обучающую выборку при этом формируют ранее классифицированные заявки.

Пример 10. Задача прогнозирования потребительского спроса в торговых сетях. Задача ставится следующим образом: спрогнозировать объемы продаж для каждого товара на заданное число дней вперед, чтобы оптимизировать план закупок, ценовую политику, график рекламных кампаний. Так как количество товаров может исчисляться десятками тысяч, принятие решений по каждому товару «вручную» невозможно, что и обуславливает необходимость привлечения процедур автоматизации поддержки принятия решений. Специфическая особенность такой задачи – несимметричность функции потерь: потери от избытка товара на прилавках существенно меньше потерь от его нехватки в таком же количестве.

Признаки формируют временные ряды цен и объемов продаж по товарам и по отдельным магазинам. Для увеличения точности прогнозов используются и различные внешние факторы, влияющие на спрос: уровень инфляции, погодные условия, рекламные кампании, социально-демографические условия, активность конкурентов. В роли объектов, в зависимости от целей анализа, могут выступать либо товары, либо магазины, либо пары магазин–товар. Обучающая выборка – это фактические данные прошлых продаж.

Пример 11. Принятие инвестиционных решений на финансовом рынке. Большой популярностью пользуются автоматические торговые стратегии – алгоритмы, принимающие торговые решения без участия человека, в том числе с использованием методов распознавания образов. Объектами здесь является рыночная ситуация в разные моменты времени, а признаками – предыстория изменения цен и объемов торгов, зафиксированная к данному моменту.

Задача состоит в том, чтобы выбрать по данному виду ценных бумаг одно из 3 возможных решений: купить, продать или выждать. Обучающую выборку состав-

ляют исторические данные о движении цен и объемов продаж за некоторый промежуток времени.

Критерий качества в такой задаче существенно отличается от стандартного функционала средней ошибки прогнозов, поскольку трейдера интересует не точность прогнозирования, а максимизация итоговой прибыли.

Пример 12. Задачи анализа клиентских сред. Клиентская среда – это совокупность клиентов некоторой компании или сервиса, регулярно пользующихся фиксированным набором услуг или ресурсов. Можно говорить о клиентских средах торговых сетей, операторов связи, эмитентов пластиковых карт, библиотек, электронных магазинов, интернет-порталов, и т. д. Современные информационные технологии позволяют детально протоколировать действия клиентов в электронном виде. Эти протоколы содержат информацию, необходимую для решения широкого спектра задач, объединяемых термином «управление взаимоотношениями с клиентами» (Customer Relationship Management, CRM). Оно подразумевает выявление целевых групп клиентов, персонализацию работы с клиентами, в частности, формирование персональных предложений, прогнозирование возможного оттока клиентов (churn prediction), выявление мошенничества (fraud detection) или необычного поведения клиентов.

Пример задачи такого рода – предсказание рейтингов отдельных товаров (например, в интернет-магазинах). Приобретая через сайт некоторый товар (книгу, фильм, и т. д.), клиент имеет возможность выразить свое отношение к нему, выставив рейтинг – обычно целое число от 1 до 5. Система использует информацию о всех выставленных рейтингах для персонализации предложений: когда очередной клиент видит на сайте страницу с описанием товара, ему показывается также список всех схожих товаров, получивших высокий рейтинг у схожих клиентов.

Задача системы распознавания состоит в том, чтобы выявить множества схожих клиентов и схожих товаров, и затем спрогнозировать их рейтинги для данного клиента. Матрица объектов-признаков – это матрица клиентов-товаров, заполненная значениями рейтингов. Как правило, она сильно разрежена и имеет более 90% пустых ячеек. Алгоритм классификации должен давать возможность предсказать рейтинг для любой незаполненной ячейки этой матрицы.

1.3. Организация решения задачи распознавания

Оценка качества обучения

Как уже отмечалось ранее, обучение системы распознавания образов сводится к поиску параметров алгоритма, доставляющих оптимальное значение заданному *функционалу качества*. Формализуем это понятие.

Функция потерь – это неотрицательная функция $L(a, x)$, характеризующая величину ошибки алгоритма a на объекте x. Если $L(a, x)=0$, то ответ $a(x)$ называется корректным.

Функционал качества алгоритма a на выборке X^K:

$$Q(a, X^K) = \frac{1}{K}\sum_{i=1}^{K} L(a, x_i). \quad (1.1)$$

При самой общей постановке задачи, когда $Y = R^N$, наиболее употребительны следующие функции потерь:

– $L(a, x)=[a(x)\neq y^*(x)]$ – индикатор несовпадения с правильным ответом, где функция [ ] принимает значение 1 при выполнении условия и 0 в противном случае;

– $L(a, x)= [|a(x)-y^*(x)|>\varepsilon]$ – индикатор существенного отклонения от правильного ответа, где ε – заданный порог точности;

– $L(a, x)=|a(x)-y^*(x)|$ – величина отклонения от правильного ответа; функционал *Q* называется средней ошибкой алгоритма *a* на выборке *X*.;

– $L(a, x)=(a(x)-y^*(x))^2$ – квадрат отклонения от правильного ответа; функционал *Q* называется средней квадратичной ошибкой алгоритма *a* на выборке *X*. .

– $Lw(a, x)=w(x)L(a, x)$ – взвешенная функция потерь, где $w(x)$ – неотрицательная весовая функция, характеризующая степень важности объекта x или величину потери от ошибки на данном объекте; $L(a, x)$ – некоторая стандартная функция потерь, например, любая из перечисленных выше.

Функционал качества *Q* называют также эмпирическим риском, так как он может быть непосредственно вычислен только по эмпирическим данным $\{(x_i, y_i), i = \overline{1,K}\}$.

Простейший метод обучения, называемый ***минимизацией эмпирического риска***, заключается в том, чтобы найти в заданной модели *A* алгоритм *a*, доставляющий минимальное значение функционалу качества *Q* на заданной обучающей выборке X^K:

$$\mu(X^K) = \underset{a\in A}{\operatorname{argmin}}\ Q(a, X^K). \quad (1.2)$$

Задачу (1.2) можно было бы решить стандартными методами оптимизации, однако возникает следующая проблема: если алгоритм *a* доставляет минимум функционалу $Q(a,X^K)$ на заданной обучающей выборке X^K, это еще не гарантирует, что он будет хорошо приближать целевую зависимость *y** на произвольной ***контрольной выборке*** $X^M = \{(x_i', y_i'), i = \overline{1,M}\}$.

Когда качество работы алгоритма на новых объектах, не вошедших в состав обучения, оказывается существенно хуже, чем на обучающей выборке, говорят об эффекте ***переобучения или переподгонки***.

По сути, задача обучения – это задача аппроксимации функциональной зависимости y* по данным обучающей выборки. Дадим геометрическую интерпретацию этому утверждению.

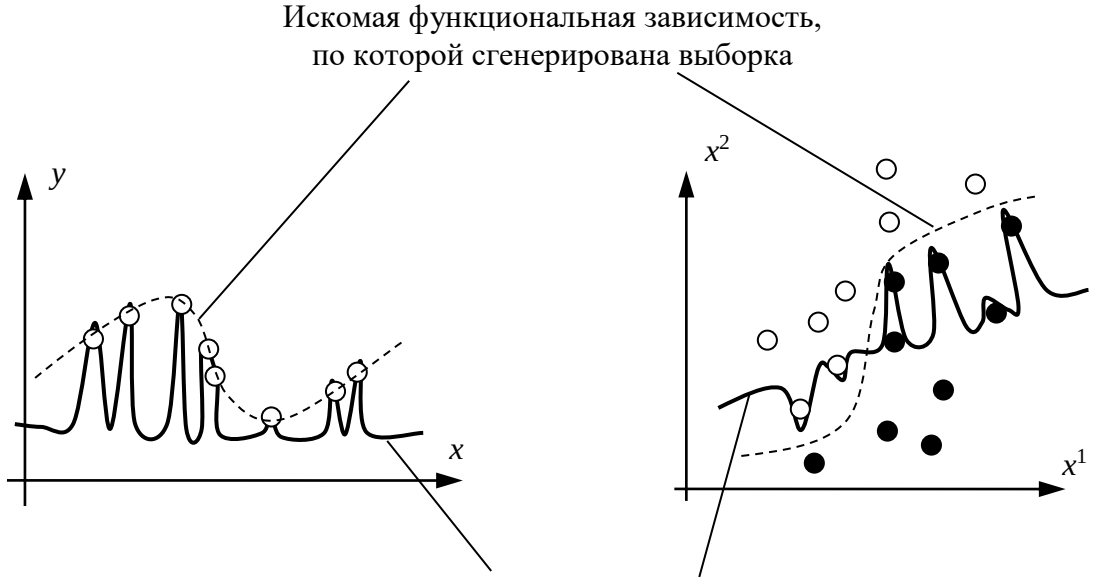
1. Если объекты характеризуются единственным параметром *x* и *Y=R*, то легко указать метод, который минимизирует эмпирический риск до нуля, но при этом абсолютно не способен обучаться (рис. 1.2, а): получив обучающую выборку *X*, запомнить ее и построить алгоритм, который сравнивает предъявляемый объект *x* с обучающими объектами x_i из X^K. В случае совпадения $x = x_i$ алгоритм выдает правильный ответ y_i, иначе выдаётся произвольный ответ. Эмпирический риск для

такого алгоритма принимает наименьшее возможное значение, равное нулю. Однако этот алгоритм не способен восстановить зависимость вне объектов обучения (на контрольной выборке значение эмпирического риска будет очень велико).

2. При $x \in R^2$ изобразить общую задачу регрессии на плоскости уже не представляется возможным, однако можно изобразить задачу классификации (рис. 1.2, б). Метод обучения строит границу между подмножествами объектов обучающей выборки, принадлежащими к разным классам (как некоторую кривую в R^2). Алгоритм определяет класс объекта $x \in R^2$ в зависимости от того, с какой стороны от границы находится соответствующая точка. Для сильно перемешанных классов минимизация эмпирического риска, по аналогии со случаем (1), дает нулевой эмпирический риск на обучающей выборке, однако искомая функциональная зависимость по-прежнему восстановлена очень плохо.

3. При большей размерности пространства признаков поставить задаче распознавания в соответствие геометрический образ уже невозможно, но проблема переобучения остается той же самой.

Т.е. для успешного решения задачи обучения необходимо не только запоминать, но и обобщать информацию. Вообще, обобщающая способность метода μ характеризуется величиной $Q(\mu(X^K),X^M)$, при условии, что выборки X^K и X^M являются представительными, и не подогнаны специально под наихудший случай.



а) б)

Рис. 1.2. Проблема переобучения
(а – для задачи регрессии при $x \in R$, б – для задачи классификации при $x \in R^2$)

Обобщающую способность метода μ можно характеризовать ***функционалом среднего риска***, т.е. математическим ожиданием ошибки:

$$R(\mu) = M_{X^K,x} L(\mu(X_K),x) = M_{X^K,X^M} Q(\mu(X_K),X^M). \qquad (1.3)$$

Эмпирические оценки риска. Оценки обобщающей способности необходимы для предсказания качества алгоритмов и разработки надежных методов обучения. Однако характеристики контрольной выборки X^M заранее не известны и вывод точных количественных оценок обобщающей способности пока остается открытой проблемой. Эмпирические оценки обобщающей способности применяются, когда нет адекватных теоретических оценок.

Пусть имеется выборка $X^L = \{(x_i, y_i), i = \overline{1,L}\}$. Разобьем ее N различными способами на две части: $X^L = X_n^K \cup X_n^M$, $n = \overline{1,N}$ – обучающую подвыборку длины K и контрольную длины M, где $K+M=L$. Вычислим среднее по всем разбиениям качество на контрольной выборке:

$$CV = \frac{1}{N}\sum_{n=1}^{N} Q_n(\mu),\ Q_n(\mu) = Q(\mu(X_n^K), X_n^M). \qquad (1.4)$$

Эта величина называется ***оценкой скользящего контроля (CV*** *– от англ.* ***cross validation)***. В зависимости от способа разбиения выборки X^L различают несколько разновидностей скользящего контроля.

– ***Контроль по отдельной выборке.*** Формируется одно разбиение N=1, как правило, случайным образом при достаточно большом M.

– ***Контроль по отдельным объектам.*** Формируются $N=L$ разбиений с контрольными выборками единичной длины, k=1.

– ***Контроль по q блокам.*** Выборка разбивается на q одинаковых блоков (обычно 10–20 блоков), и формируются $N=q$ различных разбиений. Каждый блок поочередно становится контрольной подвыборкой, остальные q–1 блоков образуют обучающую подвыборку.

– ***Контроль по случайным разбиениям.*** Формируется заданное число N случайных разбиений.

– ***Контроль по всем*** $\boldsymbol{N = C_L^K}$ ***разбиениям*** при K =1 в точности соответствует контролю по отдельным объектам, а при больших значениях K требует огромного объема вычислений.

Скользящий контроль признан де-факто стандартной методикой сравнения качества различных методов обучения на реальных данных. Основной недостаток скользящего контроля – значительные вычислительные затраты, связанные с необходимостью решить задачу обучения N раз на разных обучающих выборках.

Свойства реальных данных

В прикладных задачах исходные данные отражают сложность и разнообразие реальных процессов и явлений. Поэтому данные могут обладать рядом специфических свойств, усложняющих поиск решения. Рассмотрим основные из них.

Неточность данных. Значения признаков $f_j(x_i)$ и целевой переменной y_i могут измеряться с погрешностями. В некоторых случаях возможны грубые ошибки, приводящие к появлению редких, но больших отклонений – ***выбросов***. Опасность зашумленных данных в том, что обучаемые алгоритмы могут настраиваться на восстановление не только целевой зависимости, но и шума. Для корректной обра-

ботки выбросов существуют методы как предварительной фильтрации данных, так и ***робастные методы***, в которых задачи обучения и фильтрации решаются одновременно.

Неполнота данных. Значения признаков $f_j(x_i)$ для некоторых объектов могут вообще отсутствовать. Например, в медицинских задачах пропуски являются скорее правилом, чем исключением, поскольку здесь невозможно рассчитывать, что у каждого пациента будут проверены все возможные симптомы, применены все возможные виды обследований и лечения.

В таких случаях применяют предварительную обработку данных – заполняют пропуски спрогнозированными значениями. Идея заключается в том, чтобы для каждого признака, имеющего пропущенные значения, решить вспомогательную задачу обучения по прецедентам. В качестве обучающей выборки берутся все объекты, для которых значение данного признака известно. Затем строится алгоритм, восстанавливающий зависимость данного признака от остальных. Этот алгоритм применяется к оставшимся объектам для заполнения пропусков.

Противоречивость данных. Прецеденты i и k могут противоречить друг другу. Например, $x_i = x_k$, но $y_i \neq y_k$. Они также могут противоречить априорным ограничениям. Например, может быть заранее известно, что зависимость y^* является монотонной, но в обучающей выборке $(x_i < x_k)$ и $(y_i > y_k)$ для некоторой пары прецедентов (i, k). Противоречивость данных может быть следствием их неточности.

Разнородность данных. Признаки $f_1,...,f_N$ могут иметь различные типы (измеряться в разных шкалах). Как правило, все признаки предварительно приводятся к одному типу, однако такой переход часто приводит к потере некоторой части информации, содержавшейся в исходных признаках. Например, в логических алгоритмах классификации все признаки в конечном итоге преобразуются к бинарному типу).

Сложная структура данных. Данные могут быть представлены в более сложной форме, чем стандартная матрица объектов-признаков. Это могут быть изображения, сигналы, функции, таблицы базы данных, и т.д. В этом случае должна решаться задача ***генерации признаков***. В частности, в задачах распознавания изображений и речи, классификации текстовых документов генерация признаков является наиболее трудным этапом и в значительной степени предопределяет эффективность работы всей процедуры распознавания.

Недостаточность данных (проблема малых выборок). Объектов может оказаться существенно меньше, чем признаков. В этом случае многие классические методы статистики и аппроксимации становятся неустойчивыми или вообще неприменимыми.

Избыточность данных (проблема сверхбольших выборок). В системах с автоматическим сбором и накоплением данных возникает противоположная проблема: данных настолько много, что обычные методы обрабатывают их крайне медленно. Для решения задач распознавания в таких условиях могут применяться различные методы фильтрации или агрегирования данных.

Этапы решения задачи распознавания

Приведенная ниже последовательность действий описывает наиболее типичный цикл исследований, выполняемых при использовании методов распознавания образов. При этом выполнение каждого этапа может выявить противоречия, которые вызывают необходимость вернуться к предыдущим этапам.

Постановка задачи. На этом этапе заказчик и разработчик определяют цели решения задачи, договариваются о составе данных, фиксируют критерии качества решения.

Сбор данных. Данный этап может меняться местами с предыдущим. Нередки ситуации, когда задачи анализа данных ставятся после того, как данные собраны.

Предварительная обработка данных. Каждый алгоритм предъявляет свои требования к исходным данным, например: нормированность, отсутствие пропусков, однородность (однотипность), линейная независимость, и т. д. Для приведения данных к требуемому виду применяются различные методы предварительной обработки. Если данные имеют слишком «сырую», сложную структуру, выполняется генерация новых признаков. Возможно, при этом в признаковое описание будет включено большое количество избыточных, неинформативных признаков. Тогда на следующем этапе необходимо использовать специальные методы, способные отбирать наиболее информативные признаки.

Разработка моделей алгоритмов и методов их настройки. Необходимость создания принципиально новых моделей и методов возникает довольно редко. В то же время, некоторая адаптация готовой модели, как правило, позволяет повысить качество решения конкретной задачи.

Оценка качества алгоритмов. На практике качество алгоритмов принято оценивать по контрольным данным с помощью процедур скользящего контроля (см. выше).

Опытная эксплуатация предполагает тестирование построенного алгоритма на новых данных. Эффект переподгонки может проявиться даже в тех случаях, когда качество алгоритма подтверждается тестами на контрольных данных, но разработчик (явно или неявно) использовал результаты этих тестов для отбора наилучшего алгоритма.

Автоматизированное принятие решений без участия человека-эксперта, как правило, является конечной целью построения обучаемых алгоритмов.

1.4. Эвристические принципы обучения по прецедентам

Все эвристические принципы обучения по прецедентам в том или ином виде уточняют или модифицируют исходную задачу (1.2) минимизации эмпирического риска. Несмотря на большое разнообразие применяемых в рамках методов распознавания образов моделей алгоритмов и методов их настройки, общих принципов их построения немного. Многие модели совмещают в себе сразу несколько принципов. В рамках данного параграфа кратко рассматриваются наиболее употребительные из существующих эвристических принципов. По ходу изложения приводятся примеры простейших моделей, использующих эти принципы, а весь после-

дующий материал связан с более детальным рассмотрением некоторых из перечисленных принципов.

Принцип сходства

Принцип сходства предполагает, что на множестве X можно ввести такую функцию расстояния между объектами, что близким объектам будут, как правило, соответствовать близкие ответы:

а) для задач восстановления регрессии – это равносильно требованию гладкости целевой функции;

б) для задач классификации и кластеризации – это означает, что схожие объекты, как правило, лежат в одном классе. Граница классов может быть довольно сильно изрезана, но она не может проходить везде.

В «хорошей» задаче классы представляют собой области, компактно расположенные в пространстве признаков. Это предположение называют гипотезой компактности.

Принцип сходства лежит в основе метрических алгоритмов классификации. Пусть $\rho(x,x')$ – метрика в пространстве признаков. Метод ближайшего соседа относит объект x к тому классу, которому принадлежит ближайший объект обучающей выборки X^K:

$$a(x) = y_{n(x)},$$
$$n(x) = \underset{x_i \in X^K}{\operatorname{argmin}} \rho(x, x_i). \qquad (1.5)$$

Это, самой простой из всех алгоритмов классификации: в нем нет никаких настраиваемых параметров. Этап обучения сводится к элементарному запоминанию обучающей выборки (рис. 1.3).

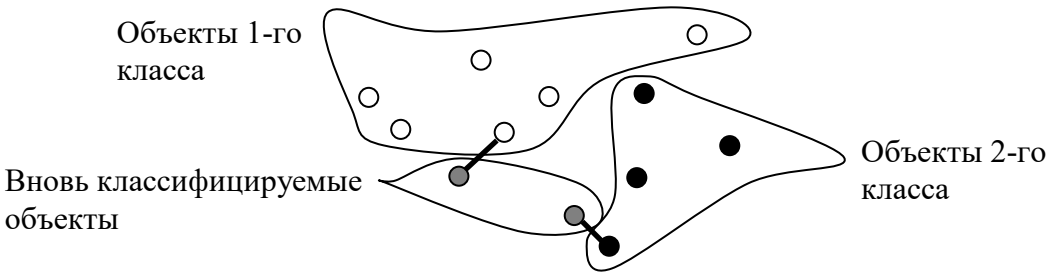


Рис. 1.3. Пример работы алгоритма классификации по методу ближайшего соседа

Наиболее тонкий вопрос для всех метрических алгоритмов анализа данных – как построить метрику. Если объекты представлены своими признаковыми описаниями, то можно взять евклидово расстояние между объектами:

$$\rho(x,x') = \sqrt{\sum_{i=1}^{N} (f_i(x) - f_i(x'))^2}, \qquad (1.6)$$

однако это далеко не единственный вариант.

Принцип регуляризации

Проблема переобучения часто проявляет себя при использовании чрезмерно сложных моделей алгоритмов. Модели, обладающие избыточным числом свободных параметров, позволяют воспроизводить точные ответы на материале обучения. При этом они улавливают не только черты восстанавливаемой зависимости, но и ошибки измерения признаков, и погрешность самой модели.

Один из способов ограничения сложности – добавить к функционалу $Q(a,X^K)$ штрафное слагаемое, наказывающее чрезмерно сложные модели:

$$\mu(X^K) = \underset{a \in A}{\operatorname{argmin}} (Q(a, X^K) + \tau C(a)), \qquad (1.7)$$

где число τ – параметр регуляризации; $C(a)$ – функционал, характеризующий сложность алгоритма a.

В общем виде принцип регуляризации можно сформулировать следующим образом: если решение существует, но оно не единственно, то из множества возможных решений следует выбрать такое, которое минимизирует дополнительный критерий регуляризации $C(a)$.

Простейшая эвристика, задающая слагаемое $C(a)$ – число настраиваемых параметров алгоритма a. Например, в задаче регрессии, если алгоритм выбирается в классе ступенчатых функций, в роли такой эвристики может выступать количество ступеней в используемой аппроксимации (рис. 1.4).

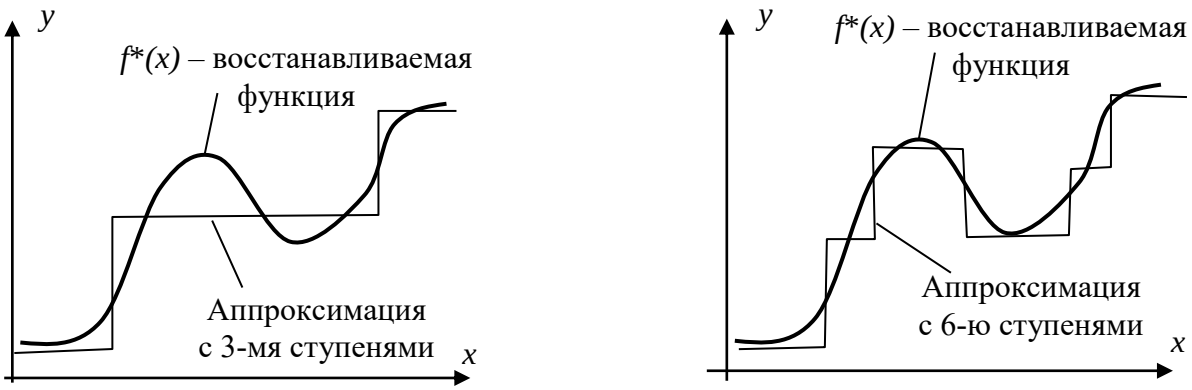


Рис. 1.4. Пример использования принципа регуляризации в задаче одномерной регрессии

Принцип разделимости

Принцип разделимости относится к задачам классификации. Он предполагает, что объекты разных классов в пространстве признаков могут быть разделены некоторой поверхностью так, что (для случая двух классов) все точки первого класса лежат по одну сторону, а все точки второго класса – по другую сторону от этой поверхности (рис. 1.5).

Рис.1.5. Принцип разделимости в задаче классификации

Пусть $Y = \{-1, +1\}$и объекты описываются признаками $f_1,..., f_N$. Линейным решающим правилом называется алгоритм классификации вида:

$$a(x) = \text{sign}(\alpha_1 f_1(x) + ... + \alpha_N f_N(x)), \quad (1.8)$$

где sign(x)=1, если $x>0$, sign(x)=–1, если $x<0$; $\alpha_1,...,\alpha_N$ – весовые коэффициенты, которые являются параметрами алгоритма и настраиваются по обучающей выборке X^K.

Самый простой метод настройки – так называемый персептронный алгоритм:

1) веса инициализируются случайными значениями в интервале $(-1/n, +1/n)$;

2) все объекты обучающей выборки в случайном порядке подаются на вход алгоритма a;

3) если объект $x_i \in X^K$. классифицируется правильно, то веса не меняются, иначе все коэффициенты модифицируются по правилу:

$$\alpha_n = \alpha_n + f_i(x_i)y_i,\ n = \overline{1,N}. \quad (1.9)$$

Доказано, что если выборка изначально линейно разделима, то этот алгоритм сходится за конечное число шагов, т.е. обучающая выборка будет разделена на два класса без ошибок.

Если предположить, что параметры объектов в каждом классе описываются некоторым вероятностным распределением, то вид оптимальной разделяющей поверхности определяется видом этих распределений. Например, если каждый класс описывается многомерным нормальным распределением, то разделяющая поверхность будет квадратичной. Для распределений с одинаковыми значениями параметров приходим к вырожденному случаю – линейному решающему правилу (рис. 1.6).

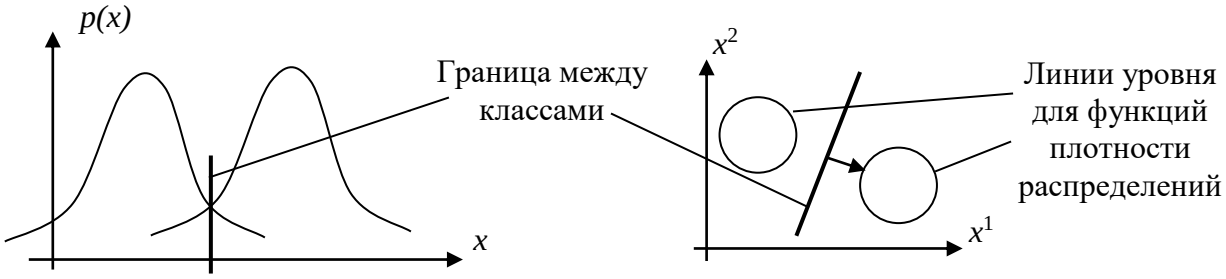


Рис. 1.6. Реализация принципа разделимости для объектов двух классов с двумерным нормальным распределением

Принцип покрытия

Принцип покрытия заключается в том, чтобы строить области в пространстве признаков, каждая из которых отделяла бы объекты относящиеся к одному классу (возможно, не все) от всех остальных. Такие области называют эталонами. Как правило, эталоны имеют простую геометрическую форму (шары, гиперплоскости, гиперпараллелепипеды).

Эталон класса $y \in Y$ – это предикат $\varphi_y : X \to \{0,1\}$. Если $\varphi_y(x) = 1$, то говорят, что эталон накрывает объект *x* и относит его к классу *y*. В противном случае эталон не дает никакой информации о классовой принадлежности объекта *x*. Алгоритм распознавания строит множество эталонов, каждый из которых отделяет лишь небольшую часть своего класса, но вместе они должны накрыть всю выборку (рис. 1.7).

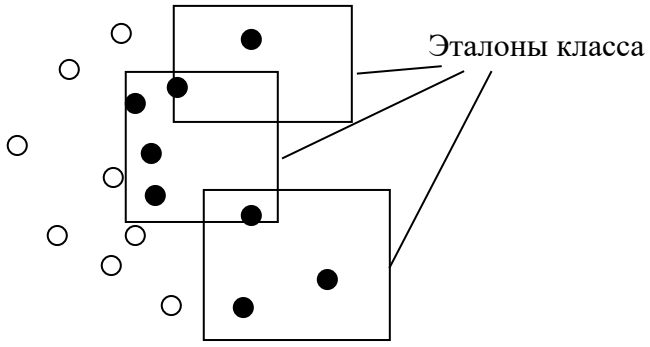


Рис. 1.7. Принцип покрытия – накрытие обучающей выборки совокупностью эталонов

Более сложный вариант алгоритма – использование пересекающихся эталонов – когда каждый эталон может накрывать некоторые объекты «чужого» класса. При этом отдельные эталоны строятся в соответствии с общим правилом (требование закономерности): каждый эталон должен накрывать достаточно много объектов «своего» класса и достаточно мало объектов «чужого» класса. Конкретные параметры этого правила зависят от задачи.

Для того, чтобы собрать алгоритм классификации из набора пересекающихся эталонов, как правило, используется ***модель голосования***: объект *x* относится к тому классу, за который голосует наибольшее число эталонов:

$$a(x) = \operatorname{argmax}_{y \in Y} \sum_{t=1}^{T} \varphi_{y_t}(x). \qquad (1.10)$$

Допустим, в качестве эталонов выбраны шары вида:

$$\varphi(x) = [\rho(x, x_0) \le r_0],$$

где x_0 –центр шара, r_0 – его радиус, $\rho(x,x_0)$ – заданная метрика (например, (1.6)).

Перебираются возможные положения центра – совпадающие с объектами выборки:

$$x_0 \in \{ x_i \in X^K \}$$

Затем при каждом x_0 перебираются возможные значения радиуса:

$$r_0 \in \{ \rho(x_i, x_0) + \varepsilon,\ x_i \in X^K \},$$

т.к. имеет смысл рассматривать только такие значения радиусов, при которых шары делят выборку по-разному.

Общее количество таких шаров – $O(K^2)$. Среди них отбираются только те, которые удовлетворяют требованию закономерности. Наконец, из них выбирается минимальное количество шаров, в совокупности накрывающих все объекты выборки по модели голосования (1.10) – это задача о покрытии множества системой его подмножеств, которая решена в рамках теории дискретной оптимизации.

Замечание. Простота эталонов обеспечивает интерпретируемость алгоритма классификации. Например, результат работы алгоритма можно представить в виде утверждения: «алгоритм $a(x)$ отнес прецедент x к классу y потому, что он достаточно близок к прецеденту x_0 – типичному представителю класса y. Среди объектов, столь же близких к x_0, классу y принадлежат 90% объектов». Прецедентная логика работы алгоритма хорошо понятна экспертам в таких предметных областях, как медицина, страхование рисков, криминалистика.

Принцип самоорганизации

Большой проблемой является выбор модели алгоритмов A, и далеко не всегда этот выбор обоснован какими-либо содержательными соображениями. Часто применяется линейный классификатор или линейная регрессия – просто потому, что их легче строить, и соответствующий метод обучения легко доступен.

Принцип самоорганизации предполагает, что структура модели $a(x,\gamma)$ не известна заранее и выбирается из некоторого множества альтернатив, настолько богатого, что с его помощью можно описать практически любую зависимость. Метод обучения на основе самоорганизации должен одновременно решать две разные задачи: выбирать структуру модели и настраивать вектор параметров γ в выбранной модели.

Как правило, эти задачи решаются по двум самостоятельным критериям: внешнему и внутреннему. Метод обучения порождает целое множество алгоритмов a_t, $t = 1,...,T$, имеющих разную структуру. Внутренний критерий используется для настройки параметров каждого из этих алгоритмов (в простейшем случае – функционал эмпирического риска $Q(a_t, X^K)$). Если все алгоритмы a_t корректные, то выбрать лучший по этому критерию не удастся, так как все значения эмпирического риска $Q(a_t, X^K)$ одинаковы и равны нулю. Для выбора лучшей из T моделей необходимо привлекать внешние данные, которые не были задействованы в процессе обучения, например с использованием критерия скользящего контроля (1.4) – это внешний критерий.

Принцип композиции

При решении сложных задач классификации, регрессии и прогнозирования часто возникает следующая ситуация. Одна за другой предпринимаются попытки

построить алгоритм, восстанавливающий искомую зависимость, однако качество работы всех построенных алгоритмов недостаточно. В таких случаях имеет смысл объединить несколько алгоритмов в композицию, в надежде на то, что погрешности этих алгоритмов взаимно скомпенсируются.

Основные механизмы композиции – усреднение и специализация. Простейший пример усреднения – взвешенное среднее:

$$a(x) = \frac{1}{T}\sum_{t=1}^{T} w_t a_t(x), \qquad (1.11)$$

где w_t – весовые коэффициенты, характеризующие качество работы отдельного алгоритма a_t в составе композиции.

Для настройки весов можно применять, например, стандартные линейные методы классификации и регрессии, рассматривая векторы $a_1(x),...,a_T(x)$ как признаковые описания объектов.

При использовании ***механизма специализации*** пространство признаков делится на T областей, в каждой из которых настраивается свой алгоритм a_t. Формально алгоритм представляется также в виде линейной комбинации (1.11), однако теперь весовые коэффициенты w_t не постоянны, а зависят от положения объекта в пространстве признаков, и называются ***функциями компетентности***. Таким образом, эти функции задают нечеткие аналоги эталонов.

Источники тестовых данных

Пока не создано универсального обучаемого алгоритма, способного решать любые практические задачи одинаково хорошо, для выбора наиболее адекватного метода приходится пользоваться эмпирическими оценками. Для построения таких оценок необходимы тестовые исходные данные. Рассмотрим потенциальные источники таких данных.

1. Эксперименты на реальных данных. Специально для тестирования и сравнения алгоритмов создаются репозитории прикладных задач. Один из наиболее известных – репозиторий задач классификации UCI4, собранный в университете Ирвина (Калифорния, США). Он содержит около сотни задач, которые охватывают различные варианты постановки (задачи классификации, кластеризации, регрессии), используют различные типы признаков объектов (логические, номинальные, порядковые, числовые), содержат данные с различными свойствами (неточность, неполнота, противоречивость, избыточность, недостаточность). Для каждой из таких задач для сравнения приводятся показатели качества работы известных алгоритмов и методов обучения.

2. Неслучайные модельные данные создаются с различными целями.

Во-первых, они позволяют наглядно продемонстрировать, в каких случаях одни методы работают лучше других. Один из классических примеров – две спирали (рис. 1.8). Эта выборка хорошо классифицируется методом ближайших соседей, но непреодолимо трудна для линейных разделяющих правил. Если витки спиралей расположить ближе друг к другу, задача станет трудна и для метода

ближайших соседей. Некоторые кусочно-линейные разделители справляются с задачей и в этом случае.

Во-вторых, модельные выборки легко параметризовать так, чтобы при изменении параметра задача постепенно переходила из разряда простых в разряд очень трудных. Это позволяет исследовать границы применимости метода. На рис. 1.9 показана серия модельных задач «пила», обладающая таким свойством относительно метода ближайших соседей и некоторых других метрических алгоритмов: чем меньше шаг зубцов пилообразной границы между классами по отношению к их длине, тем сложнее решить задачу распознавания.

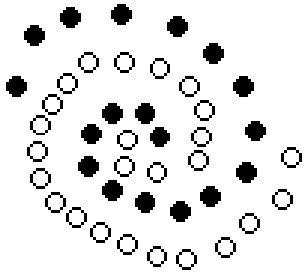


Рис. 1.8. Модельная выборка «две спирали»

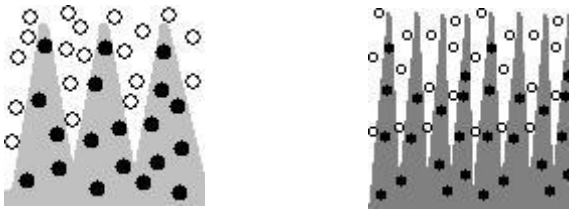


Рис. 1.9. Модельная выборка «пила»

Глава 2. ЛИНЕЙНЫЙ КЛАССИФИКАТОР И ЕГО ОБОБЩЕНИЯ

2.1. Линейное решающее правило

Определение линейного и аффинного решающего правила

В данной главе рассматриваются методы распознавания образов, в основе которых лежат:

1) предположение о том, что вероятностные характеристики распределения значений параметров объектов *p(x)* и распределения объектов по классам *p(y|x)* неизвестны;

2) принцип разделимости: объекты разных классов в пространстве признаков могут быть разделены некоторой поверхностью, так что (для случая двух классов) все точки первого класса лежат по одну сторону, а все точки второго класса – по другую сторону от этой поверхности.

Сначала рассмотрим методы построения поверхностей простейшего вида – линейных, а во второй части главы рассмотрим, как можно расширить класс алгоритмов классификации, основанных на таких простых конструкциях, за счет принципа композиции.

Определение. Аффинным решающим правилом называется алгоритм классификации вида:

$$a(x) = \text{sign}\,(< w; x > + w_0\,), \tag{2.1}$$

где $w \in R^N$ – коэффициенты линейной комбинации, $w_0 \in R$, $\text{sign}\,(u) = \begin{cases} 1, u > 0; \\ 0, u = 0; \\ -1, u < 0. \end{cases}$.

Рассмотрим основные свойства аффинного решающего правила (рис. 2.1).

Если аффинное решающее правило корректно относительно выборки $\{(x_i, y_i) \in R^N \times \{-1,1\},\ i = \overline{1,M}\}$, т.е.

$$a(x_i) = y_i,\ i = \overline{1,M}, \tag{2.2}$$

то в пространстве признаков оно задает гиперплоскость, которая удовлетворяет принципу разделимости.

Если точки $x_1, x_2 \in R^N$ лежат на этой гипреплоскости, то: $< w; x_1 > + w_0 =< w; x_2 > + w_0$. Поэтому *w* представляет собой вектор, который перпендикулярен к разделяющей гиперплоскости.

Пусть x_p – проекция произвольной точки на гиперплоскость (рис. 2.2), r – длина проекции с учетом знака, т.е. $r > 0$, если *x* лежит с «положительной» стороны от гиперплоскости (с противоположной от направления вектора), иначе $r < 0$. Тогда $r = \dfrac{< w, x > + w_0}{|w_0|}$ – это глубина погружения точки x вглубь класса. В частности, $\dfrac{w_0}{|w|}$ – расстояние от гиперплоскости до начала координат.

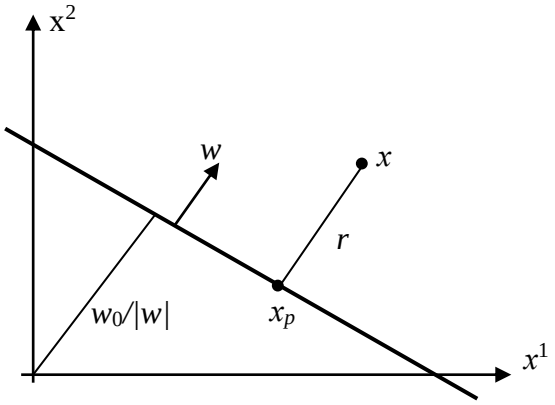


Рис. 2.1. Аффинное решающее правило

Пополненный вектор параметров $x'=(x,1)\in R^{N+1}$ и пополненный вектор весовых коэффициентов $w'=(w,w_0)$ позволяют свести алгоритм (12) к линейному (рис. 2.2):

$$a(x')=\text{sign}(<x',(w,w^0)>). \qquad (2.3)$$

Поэтому решающие правила вида (2.1) также обычно называют линейными.

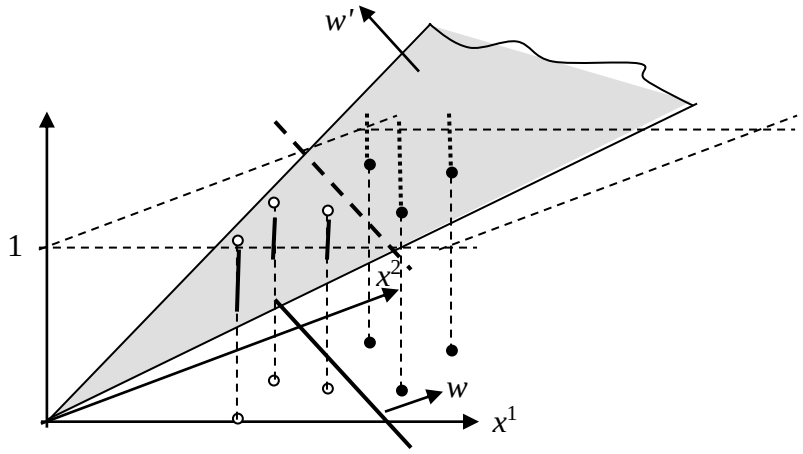


Рис. 2.2. Соотношение линейного и аффинного решающего правила

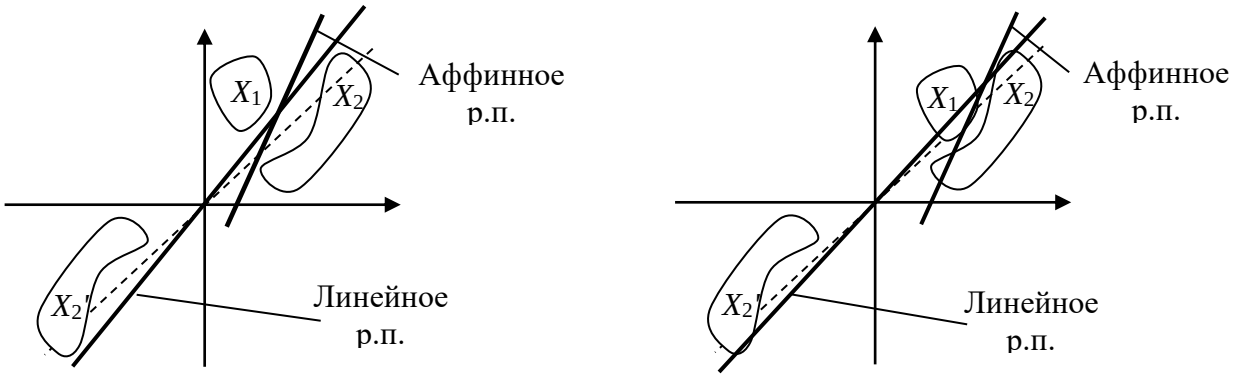
Метод обучения, использующий линейное решающее правило для разделения объектов на два класса, должен решать задачу поиска значений (w,w_0), которые являются решением системы уравнений (2.2), т.е.:

$$\text{sign}(<w;x_i>+w_0)=y_i,\ i=\overline{1,M}. \qquad (2.4)$$

Утверждение. Множества X_1 и X_2 разделимы аффинным решающим правилом тогда и только тогда, когда множества X_1 и $X_2'=\{-x: x\in X_2\}$ могут быть отнесены к одному классу линейным решающим правилом (2.3) (рис. 2.3).

Тогда, если $X_1=\{x_i: y_i=1,\ i\in 1,M\}$, $X_2=\{x_i: y_i=-1,\ i\in 1,M\}$, то заменив $x_i'=(x_i,1)$ на $x_i''=(-x_i,-1)$ для всех объектов 2-го класса (у которых $y_i=-1$) и на $x_i''=(x_i,1)$ для всех объектов 1-го класса задачу (2.4) можно свести к виду:

$$<w';x_i''>>0,\ i=\overline{1,M}. \qquad (2.5)$$



а) разделимые множества б) неразделимые множества

Рис. 2.3. Аффинная и линейная разделимость множеств

Для (2.5) можно дать двойственную геометрическую интерпретацию (рис. 2.4): w' – точка в пространстве решений, x_i'' – нормальные вектора гиперплоскостей, задающих ограничения на положение решения (оно должно находиться с «отрицательной» стороны от всех гиперплоскостей, заданных объектами) (соответственно, точки x_i'' – с отрицательной стороны от гиперплоскости, заданной w''). В совокупности они задают область решения – множество, в которое должна попасть точка w', являющаяся решением системы (2.5). Для 3-мерного случая можно рассмотреть проекцию из (M+1)-мерного пространства в M-мерное (см. рис. 2.4).

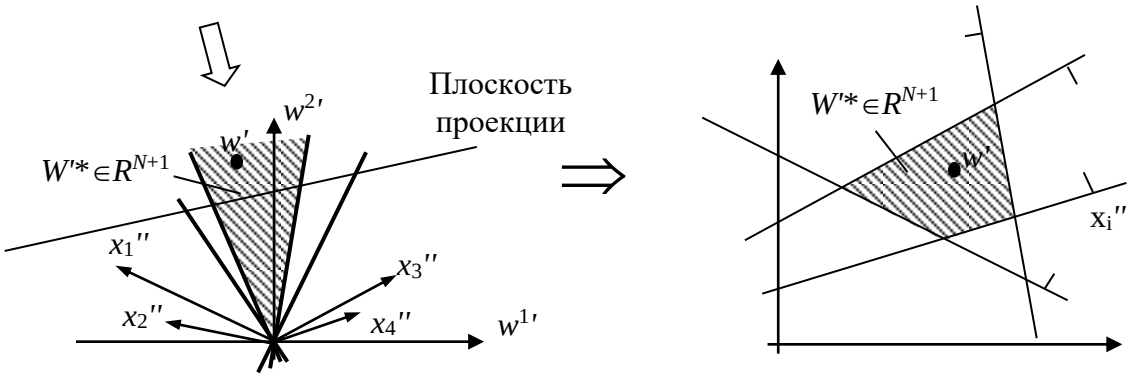


Рис. 2.4. Двойственная геометрическая интерпретация линейного решающего правила (пространство решений)

На конечной выборке объектов задача (2.5) может иметь бесконечное множество решений (если вообще их имеет). Для обеспечения максимально возможной обобщающей способности решающего правила желательно получить решение «в глубине» множества W^*, а не на его границе. Тогда разделяющая гиперплоскость пройдет на существенном расстоянии от всех объектов обучающей выборки и можно будет надеяться на правильную классификацию ею новых объектов. Этого можно добиться, усилив задачу (2.5) путем замены ее на следующую задачу:

$$<w'; x_i''> > b,\ i=1,M, \qquad (2.6)$$

где b – зазор между классами (рис 2.5).

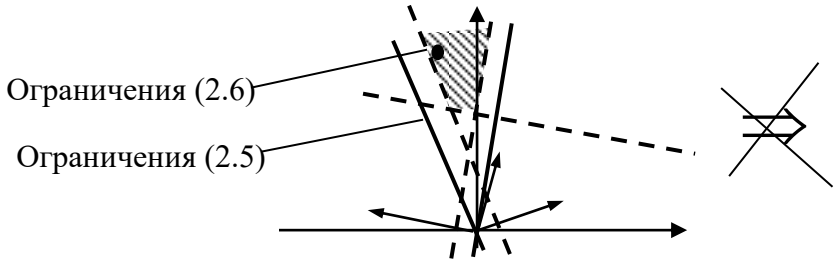


Рис. 2.5. Влияние зазора между классами на множество решений w'

Замечание. Многие алгоритмы построения линейных решающих правил позволяют получить только какое-нибудь решение из числа возможных, не гарантируя его оптимальность. Такое произвольное решение для задачи (2.6) при достаточной величине зазора автоматически будет являться близким к оптимальному для задачи (2.5).

Алгоритмы построения линейного решающего правила

К задаче построения линейного решающего правила можно подойти с двух позиций:

1) как к оптимизационной задаче – в пространстве решений для w' (см. рис. 2.4), где все ограничения сведены к одинаковому виду (2.5);

2) как к комбинаторной задаче – анализируя перекрытие выпуклых оболочек в пространстве признаков для x', где ограничения имеют вид (2.4), т.е. представлены в двух вариантах.

Оптимизационный алгоритм (перцептронный). Можно воспользоваться стандартной оптимизационной процедурой градиентного спуска (рис. 2.6):

$$w'(k+1) = w'(k) - \eta(k)\nabla Q(w'(k)),\ k = 1,2,\ldots, \qquad (2.7)$$

где $\eta(k)$ – коэффициент обжига (коэффициент обучаемости), который масштабирует размер очередного итерационного шага, $Q(w')$ – функционал качества работы алгоритма $a(w')$ на обучающей выборке, ∇ – оператор градиента, $w'(1) \in R^{N+1}$ – произвольное начальное приближение.

Процедура (2.7) повторяется до тех пор, пока не будет достигнут критерий останова:

а) $Q(w'(k)) = 0$ – т.е. построена разделяющая гиперплоскость;

б) $\nabla Q(w'(k)) = 0$ – т.е. достигнут минимум (локальный) для $Q(w')$.

Если возникла ситуация, когда условие (б) выполнено, а условие (а) – не выполнено, то обучающая выборка признается линейно неразделимой (множество W'^* пусто).

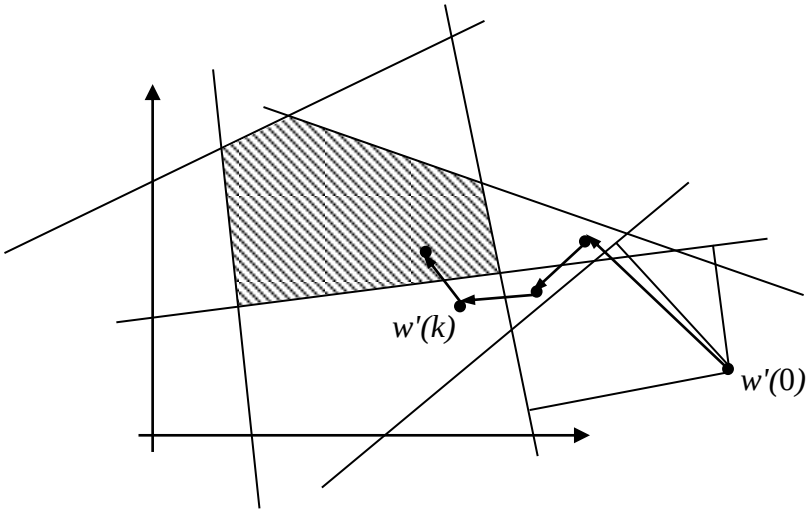


Рис. 2.6. Оптимизационных поиск в пространстве решений

Обычно $\eta(k) = \text{const}$, либо $\eta(k) \to 0$ при $k \to \infty$ – это позволяет избежать переобучения (чем больше объектов обучающей выборки уже задействовано для определения значений w', тем меньший вес придается каждому новому прецеденту). На рис. 2.7 показаны ситуации, когда значение η выбрано неудачно.

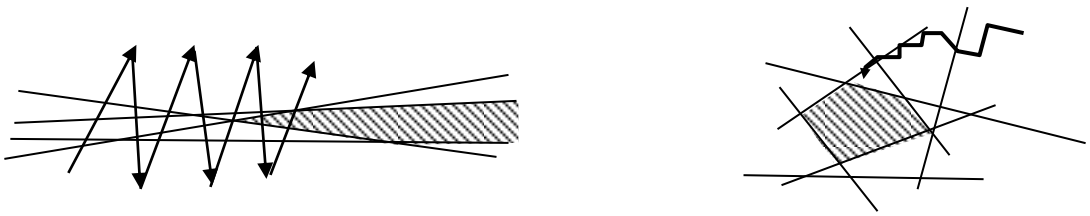


Рис. 2.7. Неудачный выбор значения коэффициента обжига

Выбор функционала качества. Функционал качества вида (1.1) (количество неправильно классифицированных объектов) – это ступенчатая функция в пространстве решений. Она не имеет градиента, поэтому использовать ее в данном случае не удастся. Вместо этого используем перцептронный функционал качества:

$$Q(w') = \sum_{i \in I} - < w'; x_i'' >, \tag{2.8}$$

где $I = \{ i \in \overline{1,M} : y_i \neq a(x_i) \}$ – множество объектов обучающей выборки, классифицированных неправильно.

Тогда $\nabla Q(w') = \sum_{i \in I} - x_i''$, поэтому метод обучения принимает вид, аналогичный (1.9):

$$\mu : w'(k+1) = w'(k) + \eta(k) \sum_{i \in I} - x_i''. \tag{2.9}$$

По теореме Новикова процесс (2.9) сходится к решению. Геометрически $Q(w')$ – это сумма векторов, являющихся перпендикулярами из текущей точки $w'(k)$ на гиперплоскости, соответствующие невыполненным ограничениям (рис. 2.7).

Комбинаторный алгоритм основан на следующем утверждении.

Утверждение. Два множества разделимы аффинным решающим правилом тогда и только тогда, когда их выпуклые оболочки не пересекаются.

Для того, чтобы построить разделяющую гиперплоскость, необходимо найти ближайшие точки выпуклых оболочек для точек из двух классов. Множество серединных перпендикуляров к отрезку, соединяющему эти точки, и будет искомой гиперплоскостью.

Выпуклую оболочку каждого класса можно задать в виде линейной комбинации входящих в него точек:

$$\forall x \in \mathrm{Conv}(X^M): x = \sum_{i=1}^{M} \alpha_i x_i, \sum_{i=1}^{M} \alpha_i = 1.$$

Две искомые точки должны лежать на гранях выпуклой оболочки, причем по крайней мере одна из них – в вершине. Поэтому задача поиска пары ближайших точек двух классов сводится к комбинаторной задаче вида:

$$\alpha = \mathrm{argmin} \left| \sum_{i:y_i=-1} \alpha_i x''_i - \sum_{j:y_i=1} \alpha_j x''_j \right| \in R^N.$$

Проблема состоит в том, что выпуклые оболочки множеств могут иметь очень много вершин: их количество оценивается величиной $O(e(M))$. Поэтому полный перебор всех вершин вычислительно сложен.

Сделаем общее замечание, относящееся ко всем алгоритмам распознавания (не только к линейным). Чем больше данных представлено в обучающей выборке, тем уже становится множество допустимых решений и решение искать сложнее. Однако чем сложнее искать решение, тем больше уверенности в том, что оно будет качественным с точки зрения обобщения информации. Обычно полагают $M > N^2$.

2.2. Алгоритмические композиции

Основная идея композиций – объединить несколько более простых алгоритмов (как правило, однотипных), каждый из которых в отдельности не может решить задачу, в расчете на то, что погрешности этих алгоритмов взаимно скомпенсируются. При построении композиции возникают следующие вопросы:

– При каких условиях качество композиции окажется лучше, чем у отдельных базовых алгоритмов?

– Как настраивать базовые алгоритмы, учитывая, что они будут работать в составе композиции?

– Как обойтись минимальным числом базовых алгоритмов?

Пространство оценок и корректирующие операции

Будем рассматривать алгоритмы, имеющие вид:

$$a(x) = C(b(x)),$$

где $b: X \to R$; $C: R \to Y$.

Многие алгоритмы классификации имеют именно такую структуру: сначала вычисляются оценки принадлежности объекта классам, затем решающее правило переводит эти оценки в номер класса. Значение оценки, как правило, характеризует степень уверенности классификации.

Определение. Алгоритмической композицией, составленной из ***алгоритмических операторов*** b_t: $X \to R$, t=1, ..., T, ***корректирующей операции*** F: $R^T \to R$ и ***решающего правила*** $C : R \to Y$ называется алгоритм a: $X \to Y$ вида:

$$a(x) = C\left[F(b_1(x),...,b_T(x))\right]. \qquad (2.10)$$

При этом функции $a_t(x) = C(b_t(x))$, t=1, ..., T называются ***базовыми алгоритмами***.

Примеры решающих правил в новой трактовке:

1) $C(b) = sign(b)$;

2) $C(b) = (b > d)$ (d – порог) – пороговое решающее правило;

3) решающее правило для задачи классификации с K непересекающимися классами:

$$C(b_1,...,b_K) = \arg\max_{k\in\overline{1,K}} b_k(x), \qquad (2.11)$$

где $b_k(x)$ – степень уверенности алгоритмического оператора в том, что объект x относится к классу k.

Перечислим основные виды корректирующих операций и в каждом случае в качестве примера рассмотрим вариант, когда базовыми алгоритмами являются линейные решающие правила.

1. ***Простое голосование***. Простейшим примером корректирующей операции является среднее арифметическое:

$$b(x) = F(b_1(x),...,b_T(x)) = \frac{1}{T}\sum_{t=1}^{T} b_t(x). \qquad (2.12)$$

Вариант при $b_t(x) \in \{-1,1\}$ – ***голосование по большинству***: объект относится к классу, к которому его относит большинство алгоритмических операторов (рис. 2.8).

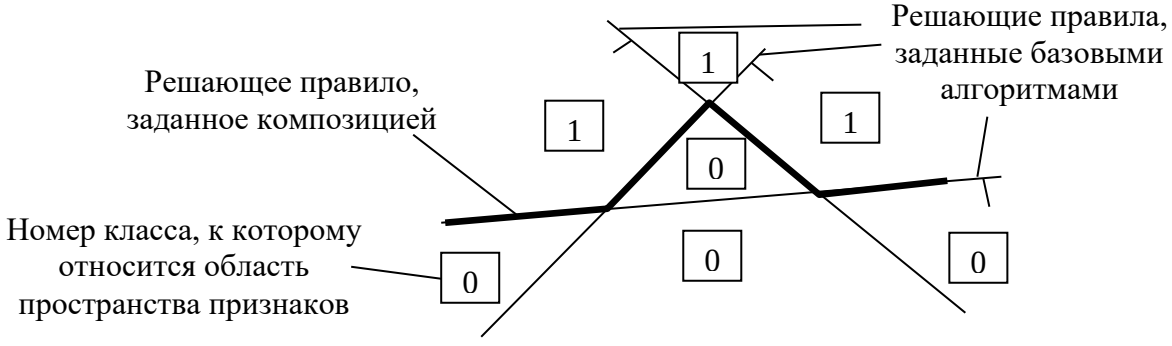


Рис. 2.8. Голосование по большинству

2. ***Взвешенное голосование***. Корректирующая операция F может иметь свободные параметры, которые необходимо настраивать по обучающей выборке наряду с параметрами базовых алгоритмов.

Пример – линейная комбинация базовых алгоритмов:

$$b(x) = F(b_1(x),...,b_T(x)) = \frac{1}{T}\sum_{t=1}^{T}\alpha_t b_t(x). \qquad (2.13)$$

3. ***Голосование по старшинству***. Корректирующая операция $F: R \to Y$, где $R \in \{0,1\}$, вычисляется по следующему алгоритму:

для $t=1,\ldots,T$

если $b_t(x)=1$, то объект x относится к классу c_t,

иначе $t=t+1$ и право голоса передается следующему по старшинству алгоритму b_{t+1}.

Таким образом, каждый базовый алгоритм в составе композиции либо указывает класс, к которому относится объект, либо отказывается от классификации и передает право голоса следующему по старшинству алгоритму. Если какой-либо базовый алгоритм указал класс для объекта, то решение младших базовых алгоритмов уже не принимается во внимание (рис. 2.9).

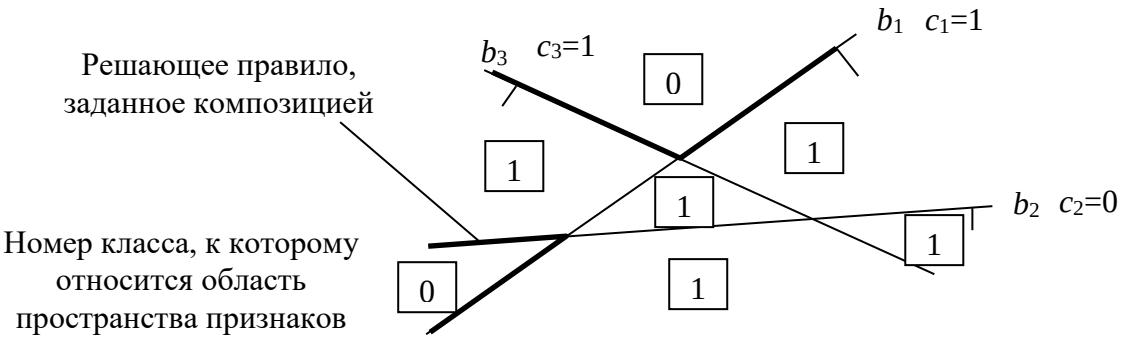


Рис. 2.9. Голосование по старшинству

4. ***Смесь алгоритмов***. Предполагает, что веса базовых алгоритмов не постоянны и зависят от положения объекта x в пространстве признаков. Квазилинейная комбинация базовых алгоритмов $b_t(x)$ с функциями компетентности $g_t(x)$, $t=1, \ldots, T$:

$$b(x) = \sum_{t=1}^{T} g_t(x) b_t(x). \qquad (2.14)$$

Если функция $g_t(x)$ принимает только два значения $g_t(x) \in \{0, 1\}$, то множество всех $x \in X$, для которых $g_t(x)=1$, называется областью компетентности базового алгоритма $b_t(x)$. Если в некоторой области пространства признаков все $g_t(x)$ меньше некоторого порога, то композиция вообще отказывается от классификации. Как правило, причина состоит в том, что в этой области находится слишком мало объектов обучающей выборки. Таким образом, смеси позволяют выявлять новые ситуации, в которых имеющиеся обучающие данные не дают оснований для выбора класса объекта.

Замечание. Композиции, построенные с использованием перечисленных корректирующих операций, можно рассматривать как базовые алгоритмы для композиции второго порядка. Это можно использовать, например, для решения задачи многоклассовой классификации: совмещая композиции с логикой большинства для задания решающих правил первого порядка и композиции с логикой старшинства, в которых вышеуказанные решающие правила выступают в качестве базовых алгоритмов).

Алгоритмы построения композиций

Можно указать основные стратегии, используемые при построении алгоритмических композиций.

1. ***Последовательная оптимизация***. Базовые алгоритмы строятся по очереди, и каждый следующий старается компенсировать недостатки предыдущих (например, для голосования старшинства – классифицирует только те объекты обучающей выборки, которые неправильно классифицированы предыдущими алгоритмами). Такая стратегия не гарантирует построения оптимальной композиции, но имеет практический смысл, поскольку почти всегда позволяет найти хотя бы какое-нибудь решение. Примеры методов обучения, использующих эту стратегию: простейшие методы построения комитетов большинства и старшинства, бустинг, метод монотонной коррекции.

2. ***Параллельная оптимизация***. Базовые алгоритмы настраиваются независимо друг от друга. Чтобы они не получались слишком похожими друг на друга, настройка производится по различным частям обучающей выборки, либо по различным частям признакового описания, либо при различных начальных приближениях. Примеры методов обучения, использующих эту стратегию: бэггинг, метод случайных подпространств, генетические методы.

3. ***Глобальная оптимизация***. Решается задача одновременного поиска значений параметров всех базовых алгоритмов, доставляющих оптимум некоторому функционалу качества работы всей композиции. Если такую задачу удается решить, то полученное решение имеет очень высокое качество, но найти его сложно по целому ряду причин:

– задача является многоэкстремальной даже при использовании алгоритмических операторов самого простого вида;

– задача вычислительно сложна (как правило, NP-сложна);

– для построения оптимизационной процедуры необходимо знать «внутреннее устройство» базовых алгоритмов, что затрудняет применение стандартных методов обучения.

Последовательный алгоритм (общая схема). Будем считать, что все базовые алгоритмы в составе композиции имеют одинаковый вид и известен некоторый стандартный метод обучения μ, который позволяет построить базовый алгоритм (т.е. определить его параметры) по заданной обучающей выборке.

На каждом шаге в композицию добавляется новый алгоритм. На t-м шаге базовый алгоритм b_t и корректирующая операция F оптимизируются при фиксированных $b_1, ..., b_{t-1}$:

$$b_t = \underset{b,F}{\operatorname{argmin}}\ Q(F(b_1,...,b_{t-1},b),X^M,Y^M). \qquad (2.15)$$

Во многих случаях для решения задачи (2.15) удается приспособить стандартные методы обучения, решающие задачу более простого вида:

$$b_t = \underset{b,F}{\operatorname{argmin}}\ Q(F(b),X^M,Y^M). \qquad (2.16)$$

Для этого функционал качества обучения выбирается в форме:

$$Q = \sum_{i=1}^{M} w_i L(a(x_i),y_i).$$

На вход стандартного метода обучения $\mu(X^M,Y^M,W^M)$ подаются модифицированные векторы весов W^M и ответов Y^M. Модификация весов, как правило, сводится к увеличению весов у наиболее «трудных» объектов, на которых чаще ошибались предыдущие базовые алгоритмы, а модификация ответов – к их замене на невязку $y_i - a(x_i)$.

Замечание. Базовый алгоритм b_t, оптимальный на t-м шаге, может оказаться не оптимальным после добавления следующих алгоритмов. Поэтому процесс (2.15)–(2.16) можно обобщить, чередуя добавление новых алгоритмов с перенастройкой предыдущих:

$$b_k = \underset{b,F}{\operatorname{argmin}}\ Q(F(b_1,...,b_{k-1},b,b_{k+1},...,b_t),X^M,Y^M). \qquad (2.17)$$

Критерии останова для процесса (2.15)–(2.17):

– построено заданное количество базовых алгоритмов T;

– достигнута заданная точность на обучающей выборке:

$$Q(F(b_1, ..., b_t), X^M, Y^M) < \varepsilon;$$

– достигнутую точность на контрольной выборке X^K не удается улучшить на протяжении последних d шагов:

$$t - t^* > d,$$

где d – параметр алгоритма, $t^* = \underset{s\in\overline{1,t}}{\operatorname{argmin}}\ Q(F(b_1,...,b_s),X^K,Y^K)$.

Параллельный алгоритм – бэггинг. Из исходной обучающей выборки длины M формируются различные обучающие подвыборки той же длины M с помощью бутстрепа – случайного выбора с повторениями. При этом некоторые объекты попадают в подвыборку по несколько раз, некоторые – ни разу. Базовые алгоритмы, обученные по подвыборкам, объединяются в композицию с помощью простого голосования.

Эффективность бэггинга объясняется двумя обстоятельствами:

1) благодаря различности базовых алгоритмов, их ошибки взаимно компенсируются при голосовании;

2) объекты-выбросы могут не попадать в некоторые обучающие подвыборки. Тогда алгоритм, построенный по подвыборке, может оказаться точнее алгоритма, построенного по полной выборке.

Другие популярные алгоритмы, реализующие параллельный поиск – это метод случайных подпространств, генетические методы.

2.3. Нелинейные обобщения линейного классификатора

Принцип разделимости, который эксплуатируется в линейных решающих правилах, позволяет представить задачу обучения, как задачу решения системы неравенств:

– параметры отдельных неравенств задаются объектами обучающей выборки;

– решение соответствует параметрам искомого решающего правила (точнее – алгоритмического оператора).

Эта идея естественным образом обобщается на случай, когда неравенства являются нелинейными. При этом разделяющая поверхность $\{x \in R^N : b(x) = 0\}$ становится нелинейной и может аппроксимировать более сложные функциональные зависимости $y^*(x)$. Нелинейность можно ввести двумя способами: за счет нелинейного алгоритмического оператора *b(x)*, либо за счет нелинейного решающего правила *C(b)*. Рассмотрим оба варианта.

Нелинейное преобразование пространства признаков

В качестве нелинейного алгоритмического оператора можно, например, выбрать полиномиальный алгоритмический оператор:

$$b(x) = w_0 + \sum_{i=1}^{N} w_i x_i + \sum_{i=1}^{N}\sum_{j=i}^{N} w_{ij} x_i x_j + \sum_{i=1}^{N}\sum_{j=i}^{N}\sum_{k=j}^{N} w_{ijk} x_i x_j x_k + \ldots . \qquad (2.18)$$

Чем больше слагаемых используется в этом выражении, тем шире класс разделяющих поверхностей. В частности, для квадратичного решающего правила форма поверхности – эллипсоид, либо гиперболоид (в зависимости от значений коэффициентов *w*).

Любой нелинейный алгоритмический оператор, в т.ч. (2.18), можно представить в альтернативной форме:

$$a(x) = \sum_{i=1}^{N'} w_i \varphi_i(x), \qquad (2.19)$$

где $\varphi_i(x), i = \overline{1, N'}$ – некоторые нелинейные функции.

Т.е. любое решающее правило, нелинейное в пространстве признаков $X = R^N$, можно представить как совокупность нелинейных функций $\varphi_i(x), i = \overline{1, N'}$, преобразующих параметры объектов в новое пространство размерности N', и линейного решающего правила в этом преобразованном пространстве.

Рассмотрим пример (рис. 2.10). Пусть нелинейные преобразования для параметров объектов имеют вид $\varphi^1(x) = x^1$, $\varphi^2(x) = x^2$, $\varphi^3(x) = \alpha x^1 x^2$ – т.е. нелиней-

ным, фактически, является преобразование только одного из трех параметров. Добавление этого параметра позволило перевести задачу из 2-мерного пространства признаков в 3-мерное. Объекты, которые раньше располагались на плоскости, теперь располагаются на криволинейной поверхности в преобразованных координатах (см. рис. 2.10). Линейное решающее правило в преобразованном пространстве может задавать границу между классами, образ которой в исходном пространстве признаков – нелинейный (см. рис. 2.10).

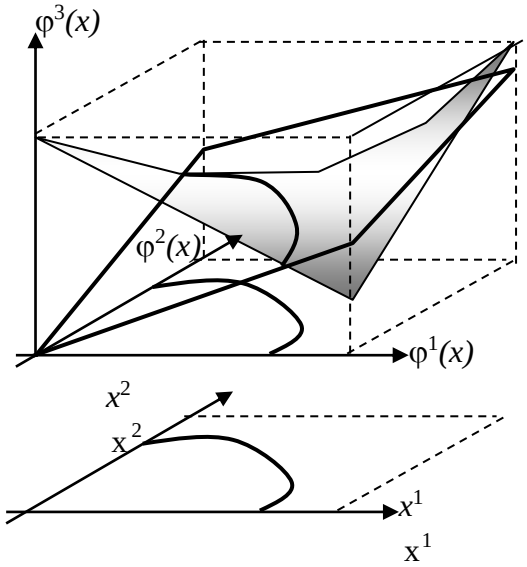


Рис. 2.10. Нелинейное преобразование пространства признаков

Рассмотрим один из популярных методов обучения, основанных на нелинейном преобразовании – метод опорных векторов SVM (Support Vector Machine).

В преобразованном пространстве для каждого объекта можно определить зазор, т.е. расстояние до разделяющей гиперплоскости:

$$m_i = \frac{y_i \operatorname{sign} < w; z_i >}{|w|},\ i = \overline{1,M}\ , \tag{2.20}$$

где $z_i = (\varphi^1(x_i),...,\varphi^{N'}(x_i))$.

В методе опорных векторов ставится задача максимизации зазора по всем объектам обучающей выборки (чтобы обеспечить максимальную обобщающую способность алгоритма):

$$\mu:\ w = \arg\max_{w} (\min_{i=\overline{1,M}} m_i)\ . \tag{2.21}$$

Опорными векторами, называются объекты обучающей выборки, расстояние от которых в преобразованном пространстве признаков до разделяющей гиперплоскости, соответствующей решению задачи (2.21), минимально (рис. 2.11). Именно они определяют положение разделяющей гиперплоскости, т.е. дают большую часть информации о свойствах обучающей выборки с точки зрения решения задачи классификации.

Доказано, что чем меньше доля опорных векторов в выборке, тем выше обобщающая способность построенного алгоритма. Поэтому задача обучения – подбор

параметров нелинейного преобразования $\varphi : R^N \to R^{N'}$, обеспечивающих минимизацию количества опорных векторов. Для полиномиальных преобразований эта задача сводится к задаче математического программирования соответствующей размерности.

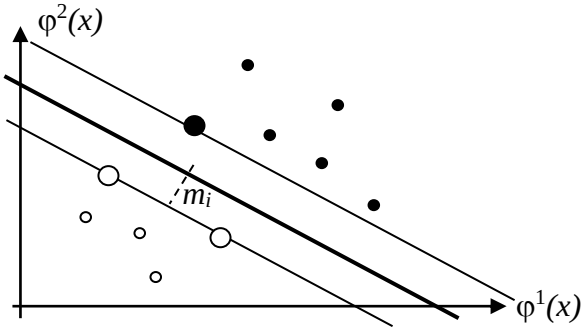


Рис. 2.11. Опорные вектора

Многослойные нейронные сети

Другая возможность обобщения линейных решающих правил – алгоритмический оператор *b(x)* оставить линейным, а решающее правило вида $C(b) = \text{sign}(b) \in \{-1, 1\}$ заменить на нелинейную функцию с непрерывным множеством значений $C(b) \in D \subset R$. Как правило, такая функция представляет собой некоторую гладкую аппроксимацию для sign(*b*). В этом случае алгоритмическую композицию можно строить не на алгоритмических операторах $b_1(x),...,b_T(x)$, т.е.:

$$a(x) = C(F(b_1(x),...,b_T(x))),$$

а непосредственно на базовых алгоритмах $C(b_1(x)),...,C(b_T(x))$, т.е.:

$$a(x) = C(F(C(b_1(x)),...,C(b_T(x)))). \qquad (2.22)$$

Тогда результаты работы базовых алгоритмов в составе композиции можно рассматривать:

а) как нелинейное преобразование исходного пространства признаков:

$$\varphi^t(x) = C(b_t(x)),\ t = \overline{1,T}, \qquad (2.23)$$

приводящее к задаче построения алгоритма вида (2.19);

б) как результат работы искусственного нейрона (рис. 2.12).

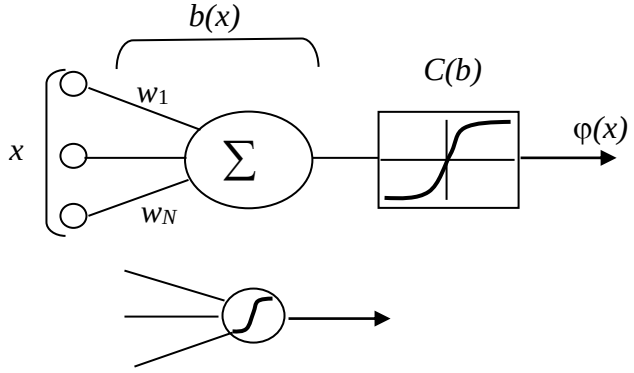


Рис. 2.12. Искусственный нейрон в терминах решающих правил

Объединив такие базовые алгоритмы в композицию, получим однослойную нейронную сеть (рис. 2.13).

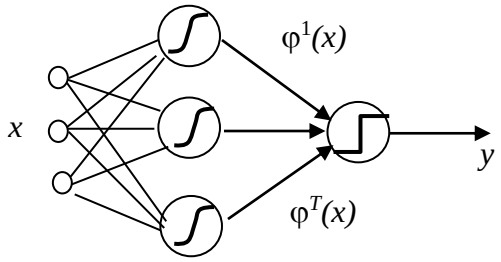


Рис. 2.13. Однослойная нейронная сеть для задачи классификации

Многослойная сеть прямого распространения (рис. 2.14) соответствует композициям более высокого порядка. Чем больше нейронов (базовых алгоритмов) и связей между ними, тем более сложный класс поверхностей описывает нейронная сеть (композиция). Как правило, нелинейные преобразования для всех нейронов выбираются однотипными, а архитектура сети определяется числом нейронов на каждом слое и полнотой связей между ними.

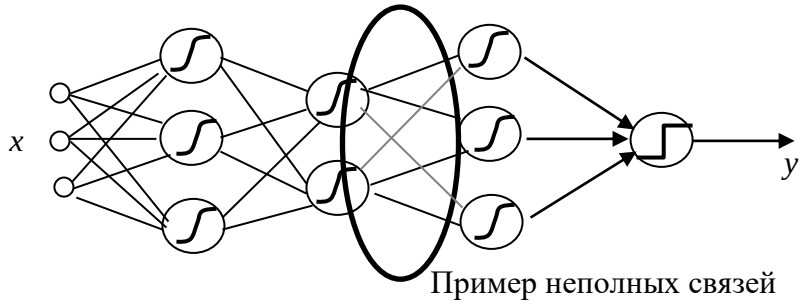


Рис. 2.14. Многослойная нейронная сеть прямого распространения

Замечание. Существуют и другие типы нейронных сетей, помимо сетей прямого распространения, которые задают иные модели преобразования информации, нежели решающие правила.

Глава 3. ВЕРОЯТНОСТНЫЕ КЛАССИФИКАТОРЫ

При вероятностной постановке задачи распознавания множество прецедентов $X \times Y$ является вероятностным пространством с функцией распределения

$$p(x,y) = P(y)p(x|y),$$

где $P(y)$ – ***априорные вероятности классов***, т.е. вероятности появления объектов каждого из классов; $p_y(x) = p(x|y)$ – ***функции правдоподобия классов***.

Алгоритм $a(x)$ должен минимизировать *вероятность* ошибочной классификации. Это вспомогательная постановка задачи распознавания. Вместо обучающей выборки здесь задается функция распределения $p(x,y)$. Позднее рассмотрим методы оценивания функции $p(x,y)$ по обучающей выборке.

Замечание. Вероятностный подход предполагает, что известная некоторая априорная информация о функции распределения $p(x,y)$. Это предположение позволяет построить алгоритм, который будет гарантированно оптимальным, т.е. даст минимально возможную вероятность ошибок. В действительности распределение приходится оценивать (восстанавливать) по имеющейся обучающей выборке конечной длины, поэтому алгоритм перестает быть строго оптимальным. Качество работы вероятностных алгоритмов существенно зависит от того, насколько адекватными были сделанные вероятностные предположения.

Можно выделить три группы вероятностных методов распознавания, в зависимости от того, какие предположения о функции распределения были приняты в качестве априорной информации:

1. Параметрические методы. Предполагается, что известен параметрический вид плотности распределения. Тогда значения параметров можно оценить по выборке, исходя из принципа максимума правдоподобия. Наиболее развиты методы восстановления многомерных нормальных распределений.

2. Методы построения смеси распределений. Функция плотности неизвестна, но предполагается, что ее можно аппроксимировать смесью конечного числа распределений заранее заданного вида.

3. Непараметрические методы. Признается невозможным единообразно описать функцию распределения на всем пространстве $X \times Y$, поэтому стоится локальная аппроксимация плотности для каждого объекта обучающей выборки.

3.1. Параметрические методы

Функционал среднего риска

Рассмотрим произвольный алгоритм $a: X \to Y$. Он разбивает множество X на непересекающиеся области:

$$A_y = \{ x \in X : a(x) = y \}.$$

Вероятность появления объекта класса y, который будет отнесен алгоритмом к классу s, равна $P_y P(A_s|y)$. Если $y=s$, то это вероятность правильной классификации, если $y \neq s$, то это вероятность ошибочной классификации. В зависимости от конкретной задачи потери от ошибок разного рода могут быть различны, поэтому

каждой паре $(y,s) \in Y \times Y$ ставится в соответствие величина потери λ_{ys} при отнесении объекта класса y к классу s.

Пример 1. В задаче радиолокационной разведки класс $y = 1$ – самолеты противника, класс $y = 0$ – ложные цели. Наибольшая потеря возникает в том случае, когда объект класса 1 принимается за объект класса 0 (ошибка I-го рода или «пропуск цели»). Когда объект класса 0 принимается за объект класса 1, говорят об ошибке II-го рода или «ложной тревоге». В данном случае $\lambda_{01} < \lambda_{10}$.

Пример 2. В задаче обнаружения спама класс y = 1 – нежелательные сообщения, класс $y = 0$ – обычные сообщения. Здесь, наоборот, пропуск спама является менее существенной потерей, чем «ложная тревога» поэтому $\lambda_{01} > \lambda_{10}$.

Как указывалось ранее, вероятность неправильной классификации объекта при использовании метода обучения μ оценивается функционалом среднего риска:

$$R(\mu) = M_{X^K,x} L(\mu(X_K), x) = M_{X^K,X^M} Q(\mu(X_K), X^M).$$

Теорема. Если известны априорные вероятности P_y и функции правдоподобия $p_y(x)$, и, кроме того, $\lambda_{yy} = 0$ и $\lambda_{ys} \equiv \lambda_y$ для всех y, $s \in Y$, то минимум среднего риска доставляется алгоритмом:

$$a(x) = \operatorname*{argmax}_{y \in Y} \lambda_y P_y p_y(x). \qquad (3.1)$$

Замечание. Согласно (3.1), уравнение

$$\lambda_t P_t p_t(x) = \lambda_s P_s p_s(x)$$

задает разделяющую поверхность между классами t и s.

Восстановление плотности распределения

Задача обучения состоит в том, чтобы, имея конечную выборку данных X^L, найти плотность вероятностного распределения $P_y p_y(x)$, ее сгенерировавшего.

Оценка для P_y строится тривиальным образом:

$$\overline{P}_y = |\{ y_i : y_i = y, i \in \overline{1:L} \}| / L.$$

Согласно закону больших чисел, случайная величина $\overline{P}_y$ сходится по вероятности к P_y при $L \to \infty$.

Задача восстановления функции правдоподобия классов $p_y(x)$ является некорректно поставленной, поскольку многие распределения могли бы сгенерировать одну и ту же выборку. Для обеспечения единственности решения при параметрическом подходе решение ищется в заранее заданном классе функций плотности распределения $\varphi(x,\theta)$.

Метод максимума правдоподобия сводит эту задачу к стандартной оптимизационной задаче:

$$\overline{\theta} = \operatorname{argmax} \sum_{i=1}^{L} \ln \varphi(x_i, \theta). \qquad (3.2)$$

Иногда решение удается выписать в явном виде, в частности для многомерного нормального распределения.

При равновероятных классах с одинаковыми параметрами распределений и независимых переменных вид параметров распределений стандартный:

$$\overline{M} = \frac{1}{L}\sum_{i=1}^{L} x_i ; \qquad \bar{\sigma} = \frac{1}{L-1}\sum_{i=1}^{L}(x_i - \overline{M})^2 . \tag{3.3}$$

В случае (3.3) разделяющая гиперплоскость проходит посередине между классами, ортогонально линии, соединяющей центры классов. Нормаль гиперплоскости обладает оптимальным свойством: в одномерной проекции на нормаль классы разделяются наилучшим образом (рис. 3.1).

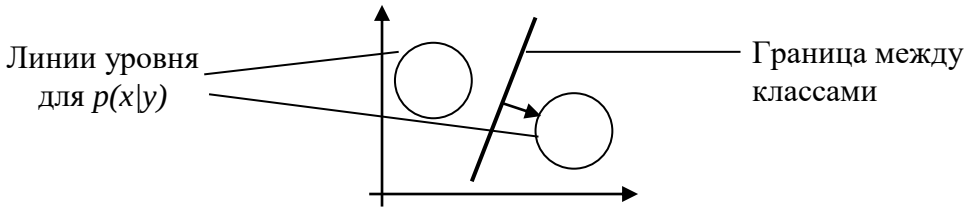


Рис. 3.1. Граница между классами в простейшем случае

Рассмотрим ряд усложнений такой задачи, позволяющих перейти к построению разделяющих поверхностей более сложного вида.

Усложнение 1: признаки коррелированны. Свойство ортогональности исчезает, однако разделяющая гиперплоскость по-прежнему проходит посередине между классами, касательно к линиям уровня обоих распределений (рис. 3.2).

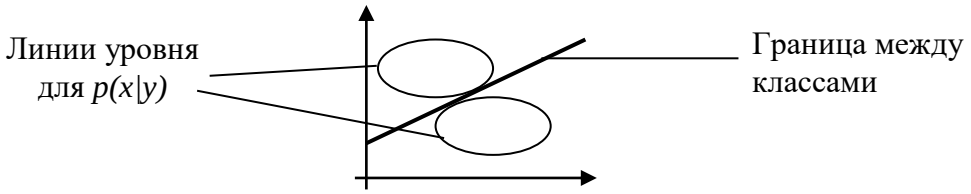


Рис. 3.2. Граница между классами в случае коррелированных признаков

Усложнение 2: классы не равновероятны или не равнозначны. Разделяющая гиперплоскость располагается дальше от более значимого класса (рис. 3.3).

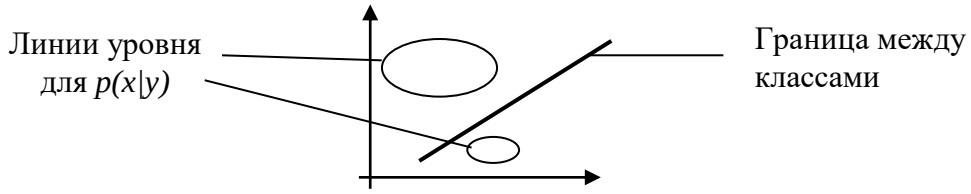


Рис. 3.3. Граница между классами при не равновероятных классах

Усложнение 3: ковариационные матрицы общего вида (не диагональны) и не равны. Тогда разделяющая поверхность становится квадратичной и прогибается так, чтобы менее плотный класс охватывал более плотный (рис. 3.4).

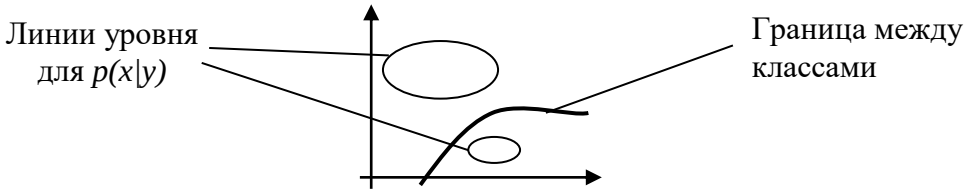


Рис. 3.4. Граница между классами для случая ковариационных матриц общего вида

Усложнение 4: Если число классов превышает 2, то разделяющая поверхность является кусочно-квадратичной, а при равных ковариационных матрицах – кусочно-линейной (рис. 3.5).

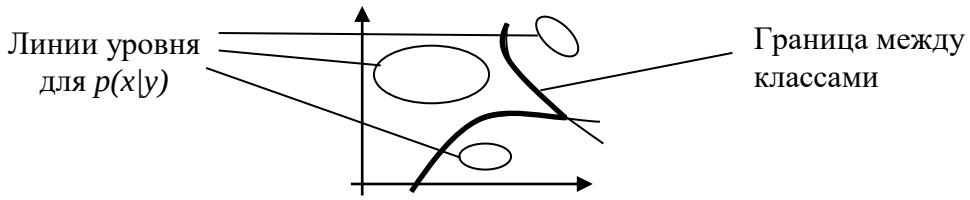


Рис. 3.5. Граница между классами при многоклассовой классификации

Замечание. Матрица $\bar{\sigma}$, по которой работает алгоритм классификации, в ряде случаев может оказаться вырожденной или плохообусловленной (обратная матрица не существует или неустойчива), что существенно снижает качество классификации, т.е. положение границ между классами будет неустойчиво относительно значений параметров классифицируемого объекта. Это может произойти в следующих случаях:

1) если длина выборки меньше размерности пространства;

2) если признаки оказываются линейно зависимыми.

Например, в базу данных по заемщикам наряду с признаками «доход заемщика» и «доход семьи» мог войти признак «доход остальных членов семьи». Таких тривиальных ситуаций легко избежать на этапе подготовки данных, однако на практике встречаются также скрытые линейные зависимости между большим числом признаков, которые обнаружить гораздо сложнее.

Робастные методы оценивания

Оценки, устойчивые относительно редких больших выбросов, связанных с малыми загрязнениями плотности, называются робастными (от англ. robust – здравый). Простейший метод робастного оценивания параметра θ плотности $\varphi(x_i, \theta)$ по заданной выборке X^L состоит в следующем.

1. Оценка параметра $\bar{\theta}$ вычисляется по всей выборке X^L, исходя из принципа максимума правдоподобия.

2. Для каждого объекта $x_i \in X^L$ вычисляется правдоподобие $\pi_i = \varphi(x_i, \bar{\theta})$. Если $\pi_i < P_0$, то объект x_i считается нетипичным (выбросом) и удаляется из выборки.

3. Оценка параметра $\bar{\theta}$ вычисляется по отфильтрованной выборке.

Пороговое значение P_0 является параметром метода. Можно также задавать пороговое значение доли удаляемых объектов.

Шаги 2 и 3 можно повторять итерационно, так как после уточнения оценки $\bar{\theta}$ некоторые объекты могут перейти в разряд нетипичных. В большинстве случаев итерационный процесс сходится за 1–2 итерации.

Тестирование методов обучения на модельных данных

Благодаря свойству оптимальности решающее правило, полученное параметрическим методом, удобно использовать в качестве эталона при тестировании различных методов обучения (не вероятностных). Методика тестирования заключается в следующем.

1. С помощью заранее известных функций правдоподобия и априорных вероятностей классов генерируются модельные выборки: обучающая и контрольная.

2. По обучающей выборке тестируемым методом настраивается алгоритм *a*.

3. Вычисляется частота ошибок алгоритма *a* и частота ошибок параметрического алгоритма на контрольной выборке. Тестируемый метод обучения считается пригодным, если эмпирическая оценка риска для алгоритма *a* оказывается не намного хуже байесовской.

3.2. Непараметрические методы

Непараметрические методы классификации основаны на локальном оценивании плотностей распределения классов $p_y(x)$ в окрестности каждого классифицируемого объекта. Такой подход не требует знания функционального вида плотностей. Вместо этого в каждой точке пространства плотность распределения класса *y* оценивается по выражению

$$p_y(x) = \sum_{i=1}^{L} \rho_i(x), \tag{3.4}$$

где $\rho_i(x) = h\rho(\theta(x_i))$ – частичная оценка плотности распределения, построенная на объекте обучающей выборки $x_i \in X^L$; $\rho(\theta(x))$, $\int_X \rho(\theta(x)) = 1$ – ***ядро плотности***, т.е. произвольная функция, параметры которой θ зависят только от того, на каком объекте она построена; *h* – ***ширина окна***, заданного оценкой $\rho_i(x)$.

Как правило, единственный параметр ядра, который зависит от объекта – это математическое ожидание, причем $M(\rho(\theta(x))) = x$. Т.е. $\rho(\theta(x_i))$ – это вклад объекта x_i в суждение о том, что данная точка пространства относится к классу y_i, а *h* – радиус окрестности объекта x_i, в которой вклад оценки $\rho_i(x)$ оказывается существенным (рис. 3.6). Типичные функции ядра плотности представлены на рис. 3.7 и в таблице.

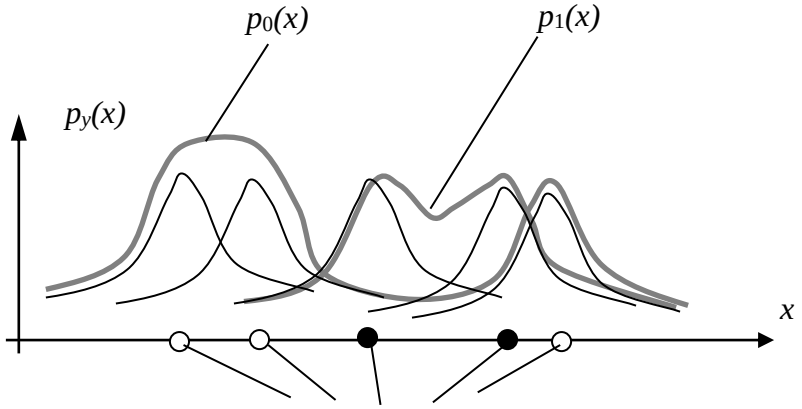


Рис. 3.6. Аппроксимация плотности распределения $p_y(x)$ с помощью частичных оценок, построенных на отдельных объектах обучающей выборки

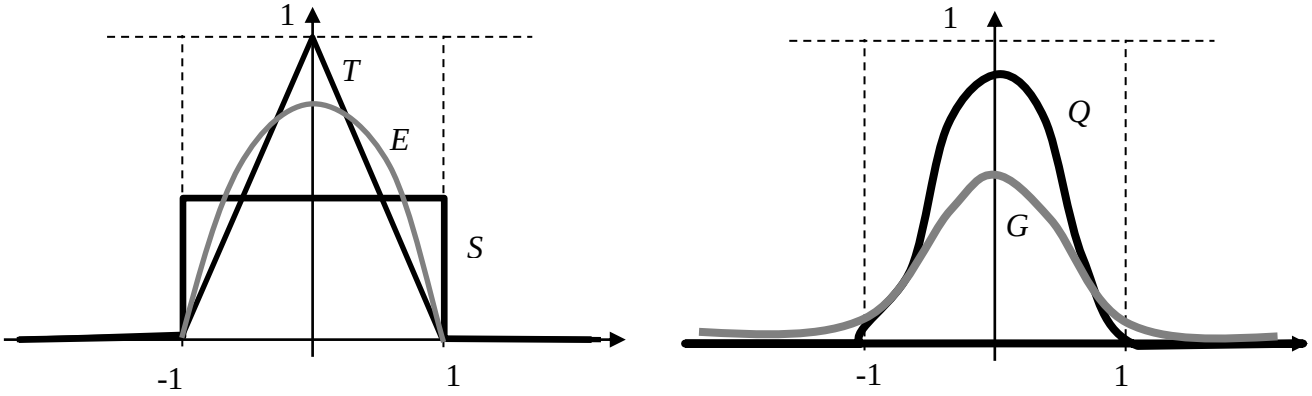


Рис. 3.7. Часто используемые функции ядра плотности:
S – прямоугольное; T – треугольное; E – Епанечникова;
Q – квадратическое; G – гауссовское

Часто используемые функции ядра плотности

Епанечникова	$E(x)=0{,}75(1-x^2)[\lvert r\rvert\le 1]$
Квадратическое	$Q(x)=15/16(1-x^2)^2[\lvert r\rvert\le 1]$
Треугольное	$T(x)=(1-\lvert x\rvert)[\lvert r\rvert\le 1]$
Гауссовское	$G(x)=(2\pi)^{-1/2}\exp(-x^2/2)$
Прямоугольное	$S(x)=0{,}5[\lvert r\rvert\le 1]$

Ширина окна h решающим образом влияет на качество восстановления плотности. При слишком узком окне ($h \to 0$) плотность концентрируется вблизи обучающих объектов и функция $p_y(x)$ претерпевает резкие скачки. При слишком широком окне плотность чрезмерно сглаживается и в пределе при $h \to \infty$ вырождается в константу. Таким образом, оптимальное значение ширины окна h – это компромисс между точностью описания конкретных данных и гладкостью

эмпирической плотности $p_y(x)$. В качестве критерия для выбора h чаще всего применяется оценка скользящего контроля с исключением объектов по одному.

Другой вариант – переменная ширина окна:

$$h_i = f(\rho(x_i, x_{(k)})),$$

где $x_{(1)}, x_{(2)}, ..., x_{(L)}$ – объекты обучающей выборки, отсортированные в порядке возрастания расстояний до x_i.

Применение такого правила обеспечивает высокое качество классификации при наличии локальных сгущений – когда распределение объектов в пространстве признаков сильно неравномерно, и одно и то же значение ширины окна h приводит к чрезмерному сглаживанию плотности в одних местах, и недостаточному сглаживанию в других.

ГЛАВА 5. ЛОГИЧЕСКИЕ АЛГОРИТМЫ РАСПОЗНАВАНИЯ

Логические алгоритмы распознавания опираются на ту же идею, что и алгоритмически композиции: объединение нескольких простых решателей в один более сложный (и более гибкий). Однако в логических алгоритмах иной баланс: исходные признаки для отдельных решателей предполагаются очень простыми (бинарные, порядковые), а количество и логика согласования решений таких решателей может быть достаточно сложной.

Основное преимущество, которое дает такой подход – интерпретируемость результатов в терминах понятной эксперту логики рассуждений. В то же время исходные данные здесь существенно загрубляются и часть доступной информации об объектах теряется.

В главе 1 уже был рассмотрен принцип покрытия, который используется во многих логических алгоритмах. Предикат $\varphi_y: X \to \{0,1\}$ покрывает объект x и относит его к классу y, если $\varphi_y(x)=1$. В противном случае эталон не дает никакой информации о классовой принадлежности объекта x. Задача логических методов:

– найти набор предикатов, каждый из которых покрывает достаточно много объектов заданного класса и как можно меньше объектов чужих классов;

– объединить эти предикаты, чтобы получить корректное решение задачи распознавания.

При этом нужно ответить на 3 вопроса:

1. Как породить предикат на исходной информации об объектах, включающей разнотипные (логические, номинальные, порядковые, числовые) признаки

2. Каков будет критерий информативности для отбора предикатов

3. Как построить решающее правило из набора наиболее информативных предикатов

1. Оценка информативности предиката

Относительно предиката класса y всю обучающую выборку можно разбить на 4 части:

$p_c(\varphi)$		$n_c(\varphi)$	
$\varphi_c(x)=1$ $y_i=c$	$\varphi_c(x)=0$ $y_i=c$	$\varphi_c(x)=1$ $y_i\neq c$	$\varphi_c(x)=0$ $y_i\neq c$
P_c		N_c	

Информативный предикат должен решать задачу:

$$p_c(\varphi)\to\max,\ n_c(\varphi)\to\min, \qquad (4.1)$$

Практика показывает, что адекватную свертку для решения единой задачи оптимизации на основе (4.1) предложить непросто, особенно при малом количестве объектов, покрываемых предикатом (т.е. когда вероятностные допущения не работают) – а для алгоритмов логической классификации это как раз распространенный случай.

В частности, такие варианты как: p/n, p/(n+p), w_1p-w_2n, p/P-n/N дают плохие результаты.

Реально применяемые критерии:

1) Эвристический критерий

Пусть $E_c(\varphi,X^L)=n_c(\varphi)/(p_c(\varphi)+n_c(\varphi))$ – доля покрываемых предикатом негативных объектов, $D_c(\varphi,X^L)=(p_c(\varphi)+n_c(\varphi))/L$ – доля всех покрываемых предикатом объектов.

Тогда от задачи (4.1) можно перейти к задаче:

$$E_c(\varphi,X^L)<\varepsilon,\ D_c(\varphi,X^L)>\delta,\ \varepsilon,\ \delta\in(0,1),\ \varepsilon\to 0,\ \delta\to 1 \qquad (4.2)$$

2) Статистический критерий:

$$I_c(\varphi,X^L)=-\ln\Gamma(p,n,P,N)=-\ln\frac{C_P^p C_N^n}{C_{P+N}^{p+n}} \qquad (4.3)$$

где $C_m^k=\frac{m!}{k!(m-k)!}$, Γ – гипергеометрическое распределение.

Смысл: если y*(x) и φ(x) – независимые случайные величины, то Γ – вероятность реализации пары (p,n) на каком-то реально существующем предикате. Если эта вероятность мала, а пара (p,n) тем не менее реализовалась – то гипотеза о независимости была неверна, т.е. предикат действительно зависим от y*(x), чего мы и добивались.

2. Бинаризация параметров объектов

Логические алгоритмы работают на предикатах. Бинарные параметры состояния объектов являются предикатами как таковые. Для параметров состояния других видов необходимо провести предварительную бинаризацию.

Пусть $f(x)\in D_f$. Необходимо разбить это множество на зоны:

а) для номинального признака $b(x)=[f(x)=d]$, $d\in D_f$

б) для порядкового или количественного признака $b(x)=[d_1<f(x)<d_2]$, $d_1,d_2\in D_f$

В случае количественных признаков имеет смысл рассматривать только такие значения d_i, которые дают разные варианты разбиения обучающей выборки на зоны по оси признака f(x). Вариантов разбиения очень много, тем более что на следующем шаге полученные бинарные признаки будут в разных вариантах входить в логические формулы, поэтому для сокращения перебора имеет смысл отдельно решить задачу оптимального разбиения значений признака на зоны.

Пусть $d_1,\ldots, d_r$ – возрастающая последовательность порогов. Зонами значений признака называются предикаты вида:

$\gamma_s(x)=[d_s\le f(x)\le d_{s+1}]$, s=1,…,r–1 (+крайние зоны)

Начнем с тривиального приближения (рис. 4.1) и будем сливать соседние пары зон с помощью жадного алгоритма: т.е. до тех пор, пока информативность слитой зоны превышает информативность исходных зон, каждый раз выбирая ту пару, при слиянии которой достигается максимальный выигрыш в информативности.

1) упорядочить выборку по возрастанию значений $f(x_i)$
2) для i=1,…,L
3) если $[y_{i-1}=c]\neq[y_i=c]$, то добавить порог в конец последовательности D, равный $(f(x_{i-1})+f(x_i))/2$
4) повторять
5) для $d_i \in D$
6) вычислить выигрыш от слияния зон γ_{i-1} и γ_i:
   $\delta I_i=I_c(\gamma_{i-1}\vee\gamma_i)-\max(I_c(\gamma_{i-1}),I_c(\gamma_i))$
7) $i=arg\max_s \delta I_s$.
8) слить зоны γ_{i-1}, γ_i, удалив из последовательности порог d_i
9) пока $\delta I_i>0$

Замечания.

1. Можно использовать более мягкое условие в (6), чтобы увеличить «сливаемость» зон.
2. Для разных классов с результат работы алгоритма может существенно отличаться

3. Составные предикаты

Бинарные параметры или результаты бинаризации параметров других видов являются элементарными предикатами, построенными на отдельных параметрах объектов. Имеет смысл сначала построить более сложные предикаты, а уже потом объединять такие предикаты в общий алгоритм.

Наиболее распространенный вариант – коньюнкции небольшого числа элементарных предикатов:

$$\varphi_c(x)=b_{c1}(x^1)\wedge\ldots\wedge b_{cK}(x^K) \qquad (4.4)$$

Как правило, коньюнкция может включать 3–7 аргументов, поскольку такие закономерности могут интерпретироваться экспертом.

Полный перебор всевозможных коньюнкций при поиске наиболее информативного варианта вычислительно сложен, поэтому необходима некоторая стратегия сокращенного перебора.

Общая идея «градиентного спуска» при поиске коньюнкции:

$\varphi_{t+1}(X^L)=\text{argmax}(\varphi,I(\varphi(X^L)))$: $\varphi\in V(\varphi_t)=\{\varphi_j: \varphi_j=O(\varphi_t)\}$

где $\varphi_0(X^L)$– произвольное начальное приближение, I – оценка информативности предиката,O( ) – множество некоторых элементарных операций над предикатом, V(φ) – окрестность предиката, т.е. множество предикатов, которые могут быть получены из φ путем совершения элементарных операций.

Конкретные варианты множества операций:

1) добавление термов, начальное приближение – пустая коньюнкция (жадный алгоритм)
2) удаление и замена термов, начальное приближение – результат работы алгоритма (1)
3) генетические алгоритмы

С геометрической точки зрения коньюнкция вида (4.4) соответствует прямоугольному параллелепипеду в пространстве параметров объектов, причем по некоторым измерениям он может быть бесконечным.

Другие распространенные варианты:

1) шары – рассматривали в главе 1: $\varphi_c(x)=[\rho(x,x_0)<r]$
2) полуплоскости – рассматривали в главе 2: $\varphi_c(x)=[<x,w> < w_0]$ – трудно интерпретировать.

4. Построение решающего правила на предикатах

Рассмотрим 2 варианта логических алгоритмов.

1. Решающий список

Решающий список состоит из упорядоченной последовательности предикатов $\varphi_1(x),\ldots, \varphi_T(x)$, каждому из которых приписан класс $c_1,\ldots,c_T \in Y$. Эта конструкция в главе 2 обозначалась как голосование по старшинству.

Часто используется дополнительное ограничение: все предикаты, относящие объекты к одному классу, должны идти в списке подряд. Это упрощает и алгоритм построения решающего правила и интерпретацию его результатов.

Жадный алгоритм построения решающего списка:

1) $U=X^L$
2) для $t=1,\ldots,T_{max}$
3) выбрать класс c, для которого будет строиться очередной предикат
4) найти наиболее информативное правило при ограничении на долю ошибок:
$\varphi_t(x)=\text{argmax}(\varphi\in\Phi,I_c(\varphi,U))$: $\Phi=\{\varphi: E_c(\varphi,U)\le E_{max}\}$
5) если $I_c(\varphi_t,U)<I_{min}$ break
6) исключить из выборки объекты, покрытые предикатом:
$U=\{x\in U: \varphi_t(x)=0\}$
7) если $|U|<L_0$ break

Достоинства решающих списков

– интерпретируемость: для предикатов стандартного вида это последовательность коньюнкций – логика, близкая к человеческой

– возможность обработки пропусков в данных: если для предиката не хватает данных об объекте, он может передать решение следующему элементу списка

Недостатки решающих списков:

– возможность построения эффективного алгоритма сильно зависит от выбора множества допустимых предикатов

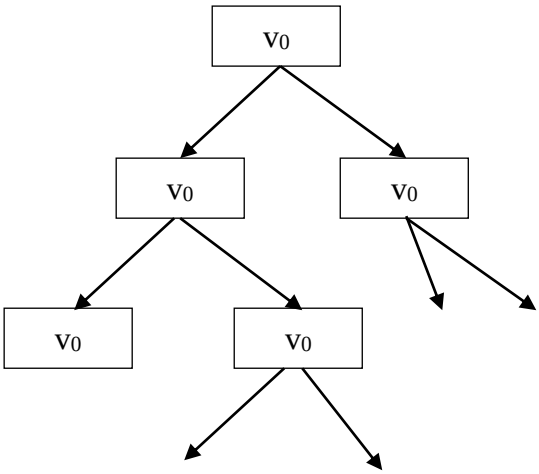
– каждый объект в итоге классифицируется только одним предикатом, что не позволяет правилам компенсировать неточность друг друга

2. Решающее дерево

Рассмотрим бинарный вариант.

В бинарном решающем дереве каждый узел, не являющийся листом, ассоциирован с предикатом $b_v(x) \in\{0,1\}$, принимающем решение о передаче объекта в

одно из своих поддеревьев, а каждый лист ассоциирован с классом, который присваивается объекту.



Т.е. объект доходит до определенного листа тогда и только тогда, когда выполняется коньюнкция из предикатов, ассоциированных с вершинами, включенными в пройденный путь от корня.

В отличии от списка, все предикаты при построении дерева ищутся одновременно. Алгоритм ID3 (Induction of Decision Tree) рассмотрим в курсе ИИ.

Достоинства:

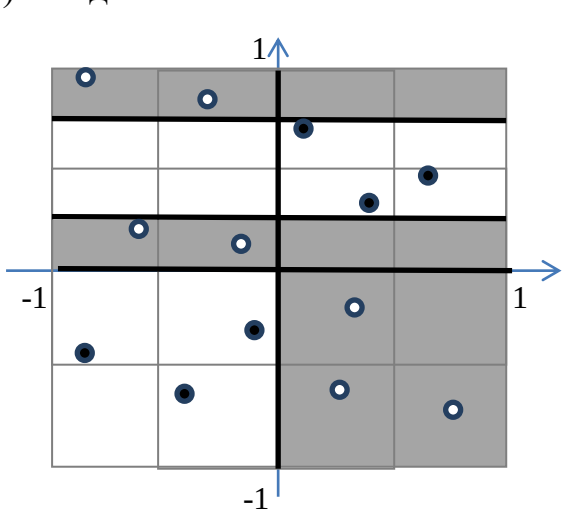
– можно гарантировать корректную классификацию даже для достаточно простого множества, из которого выбираются предикаты

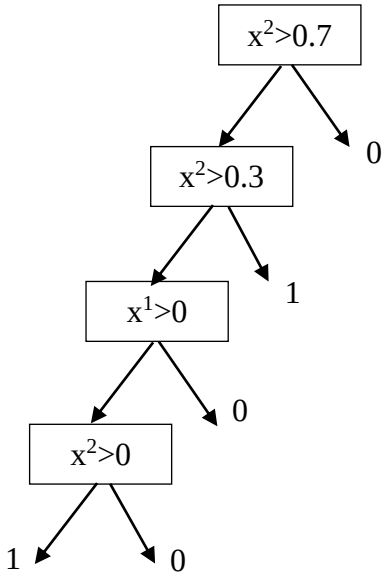
Недостатки:

– предикат для листа строится только по небольшой части обучающей выборки, которая добралась до него через предыдущие уровни

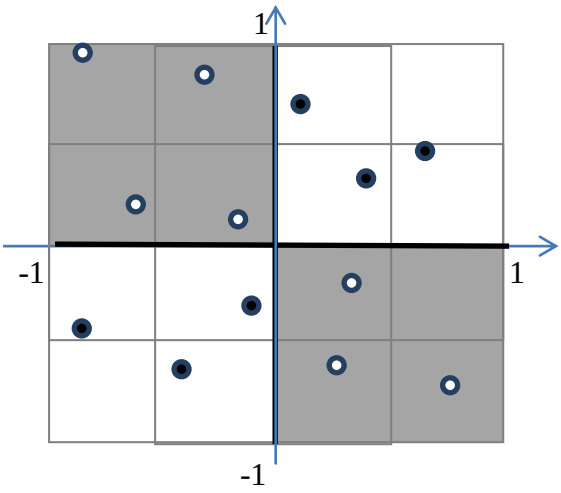
Пример.

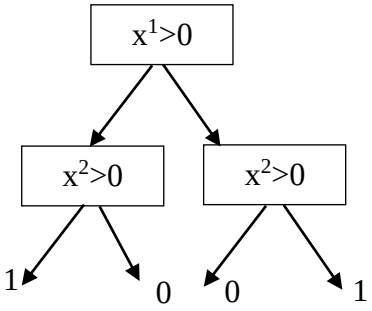
а) в виде списка





б) в виде дерева





ГЛАВА 4. ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ

Рассмотренные выше методы распознавания предполагают, что информация об объектах была предварительно преобразована в форму матрицы объектов-признаков

$$F = \begin{pmatrix} f_1(x_1) \dots f_N(x_1) \\ \dots \\ f_1(x_K) \dots f_N(x_K) \end{pmatrix}.$$

В рамках этого преобразования предполагается решение трех задач:

1) ***нормализация признаков*** – приведение полученных признаков к единому масштабу, инвариантному относительно шкал, в которых они измеряются;

2) ***селекция признаков*** – отбор признаков, которые будут наиболее информативны при решении задачи распознавания;

3) ***синтез признаков*** – построение новых признаков, являющихся функциями от исходных данных.

Нормализация – наиболее простой этап. Стандартное преобразование нормализации обеспечивает нулевое матожидание и единичное среднеквадратическое отклонение для каждого признака:

$$s_j = \frac{1}{M} \sum_{i=`}^{M} x_i^j, \quad j = \overline{1,N},$$

$$x_i^j* = (x_i^j - s_j) \Big/ \frac{1}{M-1} \sum_{k=`}^{M} (x_k^j - s_j)^2, \quad i = \overline{1,M},$$

где $x_i \in R^N$ – исходное описание объектов, $x_i* \in R^N$ – нормализованное описание.

4.1. Селекция признаков

Пусть исходное описание объектов имеет вид $\{ x_i \in R^N, i = \overline{1,M} \}$. Задача селекции признаков состоит в том, чтобы выбрать из N исходных признаков L наиболее информативных $(L<N)$. Причины, вынуждающие проводить селекцию признаков, состоят в следующем:

1) необходимо ограничить затраты на получение исходных данных (если измерение дополнительных признаков связано с временными или финансовыми затратами);

2) необходимо избавиться от сильно коррелированных признаков – это позволяет уменьшить вычислительную сложность метода обучения и повысить качество классификации.

Основная идея селекции – необходимо выбрать такое подпространство признаков, в котором расстояния между объектами разных классов будут максимальны, а расстояния между объектами одного и того же класса – минимальны.

Возможны следующие подходы к решению задачи селекции признаков.

1. Скалярная селекция: информативность каждого признака оценивается независимо, а затем выбираются L признаков с наибольшим уровнем информативности;

2. Векторная селекция: оценивается информативность целых групп признаков. Эту задачу можно решить полным перебором всевозможных комбинаций признаков, решением задачи распознавания на каждой комбинации и сравнением оценок качества полученных решений. Однако вычислительно полный перебор, как правило, невозможен, поэтому необходима некоторая стратегия сокращенного (направленного) перебора комбинаций признаков.

Скалярная селекция

При скалярной селекции признаков необходима следующая последовательность действий:

1) формируется сокращенная обучающая выборка, в которой известны значения всех N признаков для каждого объекта;

2) информативность каждого признака оценивается по этой выборке и выделяется L наиболее информативных признаков;

3) формируется полная обучающая выборка, в которой измеряются значения только L выбранных признаков;

4) по сформированной полной выборке производится обучение, т.е. настройка параметров алгоритма;

5) новые объекты, по которым измерены значения L признаков, предъявляются алгоритму для распознавания.

Возможные подходы к оценке информативности признака при скалярной селекции:

1) по статистическим характеристикам распределения значений признака для объектов разных классов;

2) с использованием оценки количества информации, добавляемой за счет использования значений признака.

Рассмотрим первый вариант. Пусть $\overline{E}_{jk}$ и $\bar{s}_{jk}^{\,2}$ – оценки математического ожидания и среднеквадратического отклонения для значений j-го признака у объектов k-го класса сокращенной обучающей выборки. Рассмотрим случайные величины, принимающие значения $\overline{E}_{jk}$ и $\bar{s}_{jk}^{\,2}$ на множестве классов $k=\overline{1,K}$. Их статистические характеристики:

$$E^*(\overline{E}_{jk})=\sum_{k=1}^{K}P_k\overline{E}_{jk},\ E^*(\bar{s}_{jk}^{\,2})=\sum_{k=1}^{K}P_k\bar{s}_{jk}^{\,2},\ D^*(\overline{E}_{jk})=E\left((\overline{E}_{jk}-E^*(\overline{E}_{jk}))^2\right).$$

Если $E^*(\bar{s}_{jk}^{\,2})<E^*(\bar{s}_{vk}^{\,2})$, то при прочих равных условиях j-й признак более информативен, чем v-й признак, т.к. объекты каждого класса располагаются компактней вдоль оси, соответствующей j-му признаку (рис. 4.1).

Если $D^*(E_{jk})>D^*(E_{vk})$, то при прочих равных условиях j-й признак более информативен, чем v-й признак, т.к. объекты разных классов располагаются на

большем удалении друг от друга вдоль оси, соответствующей *j*-му признаку (рис. 4.2).

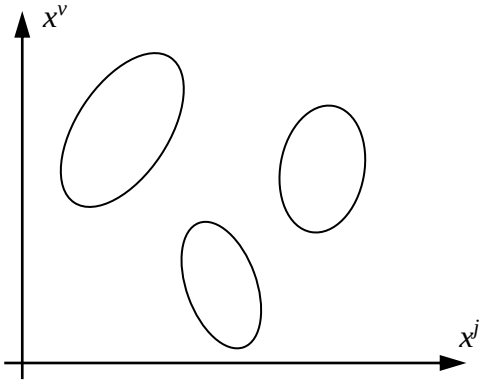


Рис. 4.1. Сравнение информативности признаков по математическому ожиданию дисперсии классов

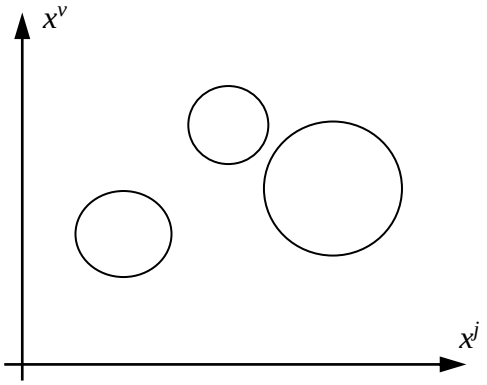


Рис. 4.2. Сравнение информативности признаков по дисперсии математических ожиданий классов

Векторная селекция

Задача векторной селекции имеет вид

$$G^L = \underset{G^L \subset G^N}{\operatorname{argmax}}\ S(G^L), \tag{4.1}$$

где G^N – исходное множество признаков, G^L – сокращенное множество признаков, $S(G^L)$ – мера отделимости классов на множестве признаков G^L.

Мера отделимости может иметь форму:

– некоторой эвристической оценки разделимости классов по описанию объектов обучающей выборки в выбранном подпространстве признаков G^L (например, аналогично рассмотренным оценкам для скалярной селекции);

– оценки качества решения задачи распознавания на выбранном подмножестве признаков G^L (для этого нужно решить задачу распознавания и оценить качество решения, например, по методу скользящего контроля (1.4)).

Стратегия векторной селекции – это некоторая процедура, позволяющая заменить полный перебор всевозможных $G^L \subset G^N$ при решении задачи (4.1) на сокращенный перебор. На практике распространены следующие стратегии.

1. Стратегия наращивания вектора признаков: выбрать $G^T = \varnothing$ и поочередно добавлять к множеству G^T по одному признаку, дающему решение задачи (4.1), пока $T < L$.

2. Стратегия сокращения вектора признаков: выбрать $G^T = G^M$ и поочередно исключать из множества G^T по одному признаку, дающему решение задачи (4.1), пока $T > L$.

Обе стратегии являются «жадными», т.е. не допускают возврата и пересмотра уже полученной ранее части решения.

3. Алгоритм плавающего поиска (не жадный) имеет следующий вид:

1) принять $G^T = g^i$, где $g^i, i \in \overline{1,N}$ – произвольный признак исходного множества признаков G^N;
2) решить перебором задачу $g^k = \underset{g \in C^N \setminus G^L}{\operatorname{argmax}} S(\{ G^T \cup g \})$ и принять $G^{T+1} = G^T \cup g^k$;
3) определить признак в составе G^{T+1}, дающий наименьший вклад в увеличение меры отделимости классов: $g^v = \underset{g \in G^{T+1}}{\operatorname{argmax}} S(G^{T+1} \setminus g)$;
4) если $g^v \neq g^k$, то исключить признак g^v из подпространства: $G^{T+1} = G^{T+1} \setminus g^v$;
5) принять $T = T + 1$; если $|G^T| < L$, то перейти к шагу (2), иначе G^T – искомое подмножество признаков.

4.2. Синтез признаков

Пусть исходное описание объектов имеет вид $\{ x_i \in R^N, i = \overline{1,M} \}$. Задача синтеза признаков: задать отображение $f : R^N \to R^L$, которое обеспечит максимальное качество решения задачи распознавания по обучающей выборке $\{ x_i^* = f(x_i), i = \overline{1,M} \}$.

Замечание. В ряде прикладных задач исходное описание объекта представлено не в форме матрицы признаков-объектов. Например, если образы – это изображения, звуковые сигналы, структуры и т.д. В таких случаях методы синтеза признаков существенно зависят от характера этого исходного представления.

Представленная постановка задачи синтеза признаков весьма неформальна – ее конкретное содержание зависит от следующих обстоятельств:

– что понимается под задачей распознавания;

– как определяется качество ее решения;

– какой класс алгоритмов распознавания и метод его обучения используется.

Рассмотрим одну из распространенных конкретизации этой постановки задачи – метод главных компонент. Задача формулируется следующим образом. Необходимо построить линейное преобразование обучающей выборки:

$$X^* = XA,\ X = \{ x_i \in R^N \},\ X^* = \{ x_i^* \in R^L \},\ \operatorname{size}(A) = N \times L,$$

при котором

$$A = \underset{A}{\operatorname{argmin}} \sum_{i=1}^{M} \rho(x_i, F(A)), \qquad (4.2)$$

где $A = \{ a_j \in R^N, j = \overline{1,L} \}$ – ***главные компоненты***, $F(A) = \{ x \in R^N : x = \sum_{j=1}^{L} \alpha_j a_j \}$ – ***линейное многообразие***, построенное на главных компонентах, $\rho(x, F(A))$ – метрика, задающая расстояние от точки $x \in R^N$ до линейного многообразия $F(A) \subset R^N$.

Если ρ – евклидова метрика, то решение задачи (4.2) известно в аналитическом виде:

1) построить матрицу $B = \sum_{i=`}^{M} (x_i - E(X^M))^2$;

2) найти ее собственные значения $v_i(B) \in R, i = \overline{1,N}$ и собственные вектора $w_i(B) \in R^N, i = \overline{1,N}$;

3) составить матрицу A из собственных векторов, которым соответствуют L наибольших по модулю собственных значений: $A = \{ v_{i_k}(B), k = \overline{1,L} \}^T$, где $i_1,...,i_L$ – индексы L наибольших по модулю собственных значений $v_i(B) \in R, i = \overline{1,N}$.

Геометрический смысл конструкций, используемых в методе главных компонент:

1) столбцы матрицы A – взаимно ортогональные вектора в исходном пространстве признаков R^N;

2) линейное многообразие $F(A)$ – подпространство, натянутое на эти вектора (в частности, при $L=N-1$ – гиперплоскость, при $L=1$ – прямая);

3) содержание задачи (4.2) – положение линейного многообразия в исходном пространстве признаков должно быть таким, чтобы сумма расстояний от объектов обучающей выборки до него была минимально возможной, т.е. искомое линейное многообразие аппроксимирует обучающую выборку (рис. 4.3);

4) преобразование $X^* = XA$– ортогональная проекция объектов обучающей выборки на построенное линейное многообразие (см. рис. 4.3).

Идея такого преобразования состоит в том, что чем меньше размерность пространства, тем:

– с одной стороны, в нем меньше информации, позволяющей разделить объекты разных классов;

– с другой стороны, меньше параметров может быть у алгоритма классификации (поэтому строить его легче).

Таким образом, синтез признаков по методу главных компонент приводит к тому, что часть информации о свойствах объектов теряется, но зато сама постановка задачи обучения упрощается.

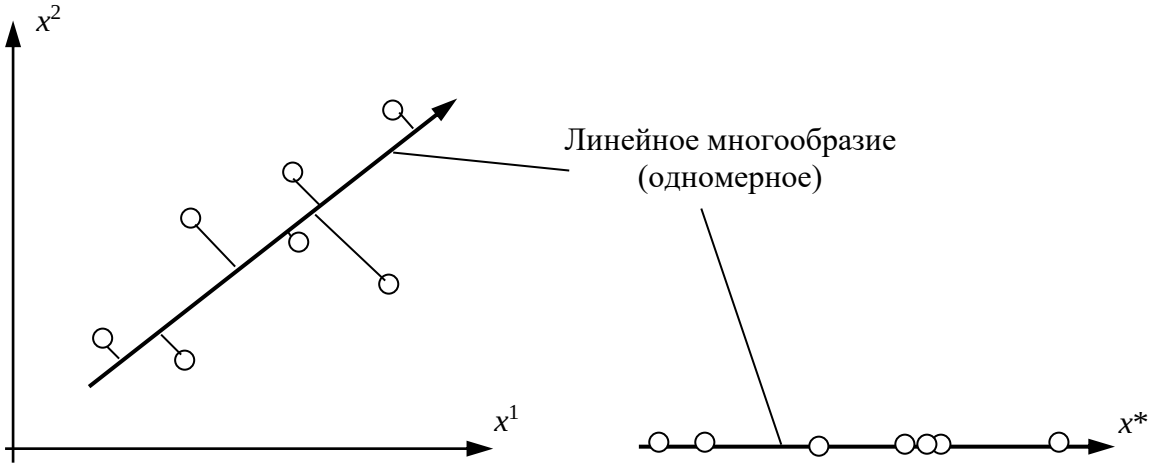


Рис. 4.3. Преобразование пространства признаков по методу главных компонент (случай $X=R^2$)

Замечание. Возможны и другие подходы к задаче синтеза признаков, где $L>N$ – в этом случае используется иная идеология синтеза.

Замечание. Любое отображение из множества X можно рассматривать как признак, поэтому любой алгоритм a: $X \to Y$, полученный в результате решения задачи распознавания также можно рассматривать как признак для задачи распознавания более высокого уровня.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Вапник, В.Н. Восстановление зависимостей по эмпирическим данным / В.Н. Вапник. – М.: Наука, 1979.
2. Воронцов, К.В. Математические методы обучения по прецедентам: курс лекций / К.В. Воронцов. – http://www.ccas.ru/voron/download/Introduction.pdf.
3. Горелик, А.Л. Методы распознавания / А.Л. Горелик, В.А. Скрипкин. – М.: Высшая школа, 1989.
4. Загоруйко, Н.Г. Прикладные методы анализа данных и знаний / Н.Г. Загоруйко. – Новосибирск: ИМ СО РАН, 1999.
5. Казанцев, В.С. Задачи классификации и их программное обеспечение / В.С. Казанцев. – М.: Наука, 1990.
6. Классификация и снижение размерности / С.А. Айвазян, В.М. Бухштабер, И.С. Енюков, Л.Д. Мешалкин. – М.: Финансы и статистика, 1989.
7. Мазуров, В.Д. Метод комитетов в задачах оптимизации и классификации / В.Д. Мазуров. – М.: Наука, 1990.
8. Местецкий, Л. М. Математические методы распознавания образов: курс лекций / Л.М. Местецкий. – www.ccas.ru/frc/papers/mestetskii04course.pdf.
9. Распознавание. Классификация. Прогноз / под ред. Ю.И. Журавлева. – М.: Наука, 1989–1994. – Вып. 1–4.
10. Ту, Дж. Принципы распознавания образов: пер. с англ. / Дж. Ту, Р. Гонсалес; под ред. Ю.И. Журавлева. – М.: Мир, 1978.
11. Фу, К. Структурные методы в распознавании образов: пер. с англ. / К. Фу; под ред. М.А. Айзермана. – М.: Мир, 1977.
12. Duda, R.O. Pattern classification / R.O. Duda, P.E. Hart, D.G. Stork. – N.Y.: Wiley Intersciences, 2002. p.

ОГЛАВЛЕНИЕ