Российская Академия Наук
Вычислительный центр

На правах рукописи

Воронцов Константин Вячеславович

Локальные базисы в алгебраическом подходе к проблеме распознавания

01.01.09 — Математическая кибернетика

диссертация на соискание учёной степени
кандидата физико-математических наук

Москва – 1999

Содержание

Диссертация выполнена в рамках алгебраического подхода к проблеме распознавания, развиваемого школой академика РАН Ю.И. Журавлёва. Целью работы является создание специальных методов, необходимых для эффективного использования конструкций алгебраического подхода при решении прикладных задач распознавания, классификации и прогнозирования. Данные методы ориентированы на непосредственное практическое применение и обеспечивают требуемое качество распознавания при относительно невысокой сложности алгоритмов. Рассматривается новая для алгебраического подхода задача построения локальных базисов, приводящая к построению алгоритмических операторов путём решения последовательности оптимизационных задач. Предлагается общая методология решения прикладных задач, основанная на использовании специального языка для описания настраиваемых алгоритмических суперпозиций.

Введение

Методы распознавания образов и восстановления зависимостей по неполным, неточным и разнородным данным используются при создании интеллектуальных информационных и аналитических систем в самых разных прикладных областях. Предпосылкой для их применения являются сбор и первичная обработка данных, в результате которых формируются описания объектов, ситуаций или явлений предметной области. Вслед за этим возникают задачи классификации объектов, восстановления неизвестных значений некоторых их свойств, прогнозирования их состояний, и т. д. Во всех этих случаях требуется строить алгоритмы, преобразующие исходную (начальную) информацию об объектах в выходную (финальную) информацию об этих же объектах.

Как правило, в задачах такого класса наличие некоторой зависимости между начальными и финальными информациями представляется несомненным, однако она может оказаться сложной и неявной, а предметная область — недостаточно формализованной, чтобы построить её адекватную модель. В таких случаях зависимость восстанавливают по прецедентам: накапливают достаточное количество пар вида «начальная информация» → «финальная информация» и строят алгоритм, корректно воспроизводящий заданные ответы на заданном конечном множестве объектов. Процесс построения алгоритма, приближающего искомую зависимость, принято называть *обучением по прецедентам*, совокупность прецедентов — *обучающей выборкой*, а начальные информации, описывающие отдельные прецеденты, — *объектами обучения*.

За неимением лучшей модели предметной области зависимость обычно выбирают из некоторого достаточно «богатого» (универсального) семейства алгоритмов с максимально широким спектром применений. Такие универсальные семейства алгоритмов называют эвристическими информационными моделями, подчёркивая тем самым, что ими моделируется не предметная область, а общие принципы преобразования информации.

Недостаток эвристических информационных моделей состоит в том, что при их использовании не гарантируется построение корректного алгоритма. «Богатые» модели как правило представляют собой сложные многопараметрические семейства алгоритмов. Даже если такая модель содержит корректный алгоритм, мы можем не найти его по причине отсутствия эффективных численных методов для решения соответствующей оптимизационной задачи. В то же время «бедные» модели, допускающие относительно простое решение оптимизационных задач, могут не содержать корректного алгоритма.

В алгебраическом подходе к проблеме распознавания указанные недостатки устраняются путём расширения эвристической информационной модели с помощью корректирующих операций. Искомый алгоритм строится в виде суперпозиции нескольких, вообще говоря, некорректных эвристических алгоритмов и корректирующей операции над ними. При этом ставится задача построения корректного алгоритма, то есть такого алгоритма, который удовлетворял бы двум требованиям: во-первых, он не должен допускать ошибок на обучающей выборке; во-вторых, он должен удовлетворять некоторым дополнительным ограничениям [37, 38], заданным на основе содержательных представлений об искомом отображении.

Традиционно в алгебраическом подходе основное внимание уделялось изучению вопросов регулярности и полноты алгебраических замыканий моделей алгоритмов [14, 15]. При этом корректные алгоритмы строились конструктивно, но основной целью их построения было доказательство теорем существования, а не решение прикладных задач. Для упрощения доказательств при изучении класса регулярных задач корректность обеспечивалась путём максимального, в некотором смысле, расширения исходной эвристической модели. Получаемое в результате семейство алгоритмов оказывалось настолько обширным, что не нужно было применять методы оптимизации, чтобы найти в нём корректный алгоритм — он легко конструировался по явным формулам. С другой стороны, получаемые таким образом алгоритмы оказывались довольно громоздкими и на практике обычно не применялись.

В данной работе развиваются методы алгебраического подхода, ориентированные в первую очередь на решение прикладных задач. Их основное отличие заключается в том, что базисный набор алгоритмов специальным образом оптимизируется под конкретную обучающую выборку и под конкретный тип корректирующих операций. Чтобы подчеркнуть это отличие, вводятся понятия *глобального базиса* и *локального базиса* задачи. Один и тот же глобальный базис может быть использован для решения широкого класса задач, что соответствует традиционному для алгебраического подхода способу построения корректного алгоритма. В отличие от глобального, локальный базис предназначен для решения одной единственной задачи. Его построение проводится путём решения специальной последовательности оптимизационных задач. Эффект от применения локальных базисов состоит в резком сокращении сложности получаемых алгоритмов и улучшении их экстраполирующей способности. Оптимизационные задачи построения

локальных базисов названы в данной работе *проблемно-независимыми* подзадачами, так как их постановки и методы решения не зависят от предметной области и свойств исходной информации.

В то же время на практике часто возникают подзадачи принципиально иного типа. К ним, в частности, относятся: предварительная обработка данных, формирование обучающих и контрольных выборок, выбор эвристических моделей, семейств корректирующих операций, критериев настройки и параметров оптимизации. Подзадачи данного типа связаны с формированием структуры суперпозиции и учётом разного рода априорной информации. В данной работе они названы *проблемно-зависимыми*, поскольку совокупность структурных характеристик, оптимальная для одной задачи, может оказаться неудачной при другой исходной информации, не говоря уже о другой предметной области. В настоящее время неизвестны строгие формальные методы выбора структуры алгоритмической суперпозиции. Обычно структура задаётся с учётом специфических особенностей решаемой задачи, различных эвристических соображений, практического опыта и вычислительных экспериментов.

Для решения проблемно-зависимых подзадач предлагается использовать не формальные методы, а специальное инструментальное средство, позволяющее описывать, настраивать и отлаживать сложные алгоритмические суперпозиции в режиме вычислительных экспериментов на реальных данных. В качестве такого средства предлагается разработанный автором язык описания суперпозиций настраиваемых алгоритмов ASDIEL (Algorithmic Superpositions Description and Investigation: Environment and Language).

Работа состоит из трёх глав.

В первой главе описывается общий метод построения локальных базисов, основанный на решении последовательности оптимизационных задач. Итоговый алгоритм конструируется в виде суперпозиции алгоритмических операторов и корректирующих операций, в общем случае произвольного уровня вложенности. Для построения каждого базисного алгоритма предлагается решать отдельную задачу оптимизации, в которой явным образом учитывается не только исходная прецедентная информация, но и место этого алгоритма в суперпозиции, то есть вид корректирующей операции и влияние других базисных алгоритмов.

Во второй главе рассматриваются методы решения оптимизационных задач отдельно для трёх семейств корректирующих операций: линейных, полиномиальных и монотонных. Для каждого семейства изучается два типа исходных задач: классификации и восстановления регрессии. Выбор именно этих случаев продиктован тем, что они часто встречаются на практике.

В каждом из шести случаев задача оптимизации базисного алгоритма с учётом его места в суперпозиции сводится к постановке, формально эквивалентной оптимизации алгоритма без учёта его места в суперпозиции. Показывается, что для решения обеих задач можно использовать одни и те же численные методы. Это позволяет практически без изменений перенести весь арсенал методов оптимизации, разработанных для различных эвристических моделей, на задачи

построения локальных базисов.

Для каждого из шести случаев строится комбинированный функционал качества, при оптимизации которого алгоритм одновременно настраивается как на обучающую выборку, так и на компенсацию неточности остальных базисных алгоритмов. Данный функционал параметризуется вещественным числом, задающим степень компромисса между обеими стратегиями настройки.

Для линейных и полиномиальных корректирующих операций оптимизационные задачи решаются стандартными методами. Разработка новых численных методов потребовалась только при построении монотонных корректирующих операций. Основные формальные результаты, полученные в этой части работы, относятся именно к данному случаю:

— доказана сходимость суперпозиции к корректному алгоритму за конечное число шагов;

— рассмотрена задача построения суперпозиции ограниченной сложности; для неё исследовано специальное отношение предпорядка на множестве объектов обучения, что позволило сформулировать критерий оптимизации очередного базисного оператора;

— решена задача построения многомерной монотонной функции, проходящей через заданные точки и обладающей достаточной степенью гладкости.

В третьей главе рассмотрена методология решения проблемно-зависимых подзадач с использованием языка алгоритмических суперпозиций ASDIEL. Описана модель данных ASDIEL и перечислены основные возможности языка. Показано, что определения *метода* и *алгоритма*, лежащие в её основе, охватывают многие известные модели алгоритмов. Для этого в качестве обоснования достаточности модели данных приведены интерфейсы некоторых стандартных методов, принципиально отличающихся как по назначению, так и по структуре входных и выходных данных. В завершение продемонстрировано использование языка ASDIEL для описания итерационных процессов настройки алгоритмических суперпозиций, рассмотренных в предыдущих главах.

Благодарности

Автор выражает глубокую признательность члену-корреспонденту РАН Константину Владимировичу Рудакову за постановку задач, многочисленные идеи и постоянное внимание к работе; академику РАН Юрию Ивановичу Журавлёву за поддержку на всех этапах выполнения данной работы; всем своим коллегам, дискуссии с которыми способствовали решению задачи монотонной коррекции и развитию языка ASDIEL.

1 Локальные базисы и методы их построения

1.1 Задачи обучения по прецедентам

В самой общей постановке задача синтеза алгоритмов преобразования информации состоит в следующем [15]. Имеется множество начальных информаций $\mathfrak{I}_i$ и множество финальных информаций $\mathfrak{I}_f$. Требуется построить алгоритм, реализующий отображение из $\mathfrak{I}_i$ в $\mathfrak{I}_f$, удовлетворяющее заданной системе ограничений.

Одним из частных случаев является задача *обучения по прецедентам*, в которой система ограничений задаётся следующим образом. Фиксируется последовательность $I_q = \{x_k\}_{k=1}^q$ элементов множества $\mathfrak{I}_i$ и последовательность $\widetilde{I}_q = \{y_k\}_{k=1}^q$ элементов множества $\mathfrak{I}_f$. Искомый алгоритм A должен точно или приближённо удовлетворять системе из q равенств

$$A(x_k) = y_k, \quad k = 1, \ldots, q,$$

которую мы будем сокращённо записывать как $A(I_q) = \widetilde{I}_q$. Ограничения такого типа называются *локальными* или прецедентными. Кроме того, обычно требуется, чтобы искомый алгоритм удовлетворял некоторым дополнительным ограничениям, которые в общем случае выражаются условием

$$A \in \mathfrak{M}^u,$$

где $\mathfrak{M}^u$ — заданное множество отображений из $\mathfrak{I}_i$ в $\mathfrak{I}_f$. Алгоритм, удовлетворяющий локальным и дополнительным ограничениям, называют *корректным*. Итак, рассматриваемые задачи обучения по прецедентам определяется пятёркой $Z = \langle \mathfrak{I}_i, \mathfrak{I}_f, \mathfrak{M}^u, I_q, \widetilde{I}_q \rangle$.

Различие между локальными и дополнительными ограничениями заключается в том, что первые относятся к конечному набору точек и допускают эффективную проверку, в то время как вторые накладываются на всё отображение «в целом» и не допускают эффективной проверки. В частности, это могут быть ограничения непрерывности, гладкости, монотонности, унимодальности, и т. д. На практике дополнительные ограничения учитываются на этапе построения параметрического семейства алгоритмов, а локальные — при последующей настройке параметров алгоритма на заданные прецеденты.

1.2 Оптимизационный и алгебраический подходы

Оптимизационный подход к решению задачи обучения по прецедентам предполагает выполнение следующих шагов.

1. Выбирается эвристическая информационная модель алгоритмов $\mathfrak{M}$ таким образом, чтобы $\mathfrak{M} \subseteq \mathfrak{M}^u$. Тем самым гарантируется, что все алгоритмы из $\mathfrak{M}$ удовлетворяют универсальным ограничениям «по построению».

2. При произвольном фиксированном q определяется функционал качества Q — отображение вида

$$Q : \mathfrak{M}^u \times \mathfrak{I}_i^q \times \mathfrak{I}_f^q \to \mathbb{R}$$

так, чтобы значение $Q(A, I_q, \widetilde{I}_q)$ было тем меньше, чем точнее алгоритм A удовлетворяет условию $A(I_q) = \widetilde{I}_q$.
3. Решается задача оптимизации функционала качества в рамках выбранной модели $\mathfrak{M}$ и найденный алгоритм

$$A^* = \arg\min_{A \in \mathfrak{M}} Q(A, I_q, \widetilde{I}_q)$$

объявляется решением (как правило приближённым).

На практике может оказаться, что построить адекватную модель алгоритмов не удаётся, либо выбранная модель не содержит приемлемого алгоритма, либо используемый метод оптимизации не находит его. Поиски регулярного способа разрешить эти проблемы привели к возникновению алгебраического подхода [14, 15]. Основная его идея состоит в том, чтобы расширить исходную модель алгоритмов с помощью корректирующих операций. Строится некоторое количество алгоритмов и корректирующая операция над ними с тем расчётом, чтобы результирующий алгоритм гарантированно удовлетворял локальным ограничениям.

Алгебраический подход к проблеме распознавания предполагает выполнение следующих шагов.

1. Наряду с множествами $\mathfrak{I}_i$ и $\mathfrak{I}_f$ вводится пространство оценок $\mathfrak{I}_e$. Оно выбирается так, чтобы на нём можно было определить удобные алгебраические операции, приводящие к построению подходящих корректирующих операций.
2. Выбирается модель алгоритмических операторов

$$\mathfrak{M}^0 \subseteq \mathfrak{M}^0_* = \{B : \mathfrak{I}_i \to \mathfrak{I}_e\}$$

и семейство решающих правил

$$\mathfrak{M}^1 \subseteq \bigcup_{p=0}^{\infty} \{C : \mathfrak{I}_e^p \to \mathfrak{I}_f\}.$$

Всевозможные суперпозиции алгоритмических операторов и решающих правил образуют эвристическую информационную модель алгоритмов $\mathfrak{M} = \mathfrak{M}^1 \circ$ $\circ\, \mathfrak{M}^0$, см. рис. 1(а).
3. Выбирается семейство корректирующих операций

$$\mathfrak{F} \subseteq \bigcup_{p=0}^{\infty} \{F : \mathfrak{I}_e^p \to \mathfrak{I}_e\}.$$

Заметим, что суперпозиция $B(x) \equiv F(B_1(x), \ldots, B_p(x))$ осуществляет отображение из $\mathfrak{I}_i$ в $\mathfrak{I}_e$, и поэтому также является алгоритмическим оператором.

Введём для него обозначение $F(B_1, \ldots, B_p)$. Всевозможные суперпозиции алгоритмических операторов, корректирующих операций и решающих правил образуют так называемое $\mathfrak{F}$-расширение модели $\mathfrak{M}$, обозначаемое $\mathfrak{F}(\mathfrak{M})$, в рамках которого и ведётся поиск корректного алгоритма (см. рис. 1(б)). Все три семейства отображений строятся таким образом, чтобы соблюдалось требование $\mathfrak{F}(\mathfrak{M}) \subseteq \mathfrak{M}^u$. Тем самым гарантируется, что все алгоритмы удовлетворяют универсальным ограничениям «по построению». Общие подходы к построению $\mathfrak{F}$-расширений развиваются в теории универсальных и локальных ограничений [37, 38, 39].

4. Выбираются алгоритмические операторы $B_1, \ldots, B_p$ из $\mathfrak{M}^0$, корректирующая операция F из $\mathfrak{F}$ и решающее правило C из $\mathfrak{M}^1$ так, чтобы алгоритм

$$A = C \circ F(B_1, \ldots, B_p)$$

удовлетворял условию корректности, см. рис. 1(в).

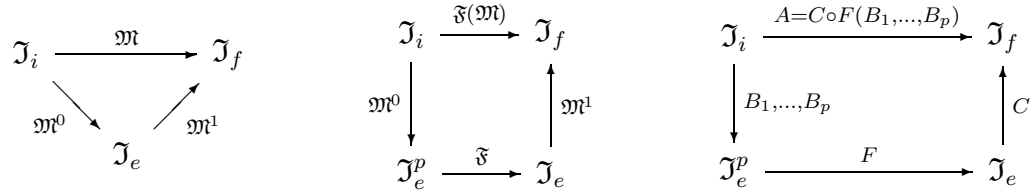
В предыдущих работах по алгебраическому подходу [14, 15, 37, 38] было показано, что построение корректного алгоритма A возможно для широкого класса задач, называемых регулярными, при условии, что $\mathfrak{M}^0$ и $\mathfrak{F}$ обладают свойством полноты. Доказательство теорем существования в этих работах основывалось на конструктивном построении корректного алгоритма, не требовавшим применения методов оптимизации. В ряде случаев — для отдельных $\mathfrak{M}^0$ и $\mathfrak{F}$ — искомый алгоритм удавалось выписать в виде явной формулы. При этом использовалось много (порой неоправданно много) алгоритмических операторов, по отдельности обладавших довольно низким качеством. Такие конструкции были удобны для теоретических рассмотрений, но не предназначались для непосредственного применения на практике. Алгоритмы, реализованные по явным формулам, получались слишком громоздкими и не давали надёжных результатов. Основная причина этого заключалась в том, что набор операторов $B_1, \ldots, B_p$ оставался одним и тем же для всех регулярных задач и никак не учитывал специфику данной конкретной задачи.

Вообще, для существования корректного алгоритма необходимо, чтобы набор операторов $B_1, \ldots, B_p$ был достаточно разнообразным, а семейство $\mathfrak{F}$ — достаточно «богатым». Для точного изучения данных требований введём понятия глобального и локального базисов задачи Z при заданных $\mathfrak{F}$ и $\mathfrak{M}^1$.

Определение 1. Набор алгоритмических операторов $B_1, \ldots, B_p$ называется *глобальным базисом* для задачи Z при заданных $\mathfrak{F}$ и $\mathfrak{M}^1$, если для любой финальной информации $\widetilde{I}'_q \in \mathfrak{I}^q_f$ найдётся корректирующая операция $F \in \mathfrak{F}$ и решающее правило $C \in \mathfrak{M}^1$ такие, что $A(I_q) = \widetilde{I}'_q$.

Определение 2. Набор алгоритмических операторов $B_1, \ldots, B_p$ называется *локальным базисом* для задачи Z при заданных $\mathfrak{F}$ и $\mathfrak{M}^1$, если найдётся корректирующая операция $F \in \mathfrak{F}$ и решающее правило $C \in \mathfrak{M}^1$ такие, что $A(I_q) = \widetilde{I}_q$.

Произвольный глобальный базис является также и локальным. Обратное в общем случае неверно.



(а) модель $\mathfrak{M}$ *(б) модель* $\mathfrak{F}(\mathfrak{M})$ *(в) структура алгоритма*

Рис. 1: *Построение суперпозиций алгоритмических операторов, корректирующих операций и решающих правил в алгебраическом подходе к проблеме распознавания.*

В предыдущих работах, носивших главным образом теоретический характер, корректный алгоритм строился на основе глобального базиса, хотя сам термин и не вводился. В настоящей работе предлагается при практическом применении алгебраического подхода использовать локальные базисы. Преимущество локального базиса заключается в том, что он обеспечивает существование корректного алгоритма только для одной задачи, а не для всех регулярных задач. Требования к базису в данном случае слабее, следовательно множество допустимых базисов шире, и допустимыми могут оказаться базисы существенно меньшей мощности p. Для построения таких базисов предлагается решать последовательность задач оптимизации, рассмотрение которой и будет нашей ближайшей целью.

1.3 Оптимизационные задачи построения локальных базисов

Пусть задан функционал качества алгоритмических операторов $Q : \mathfrak{M}_*^0 \times \mathfrak{I}_i^q \times \times \mathfrak{I}_f^q \to \mathbb{R}$, принимающий нулевое значение $Q(B, I_q, \widetilde{I}_q) = 0$ тогда и только тогда, когда существует решающее правило C из $\mathfrak{M}^1$, при котором алгоритм $A = C \circ B$ корректен на прецедентной информации $\langle I_q, \widetilde{I}_q \rangle$.

В общем случае задача построения локального базиса заключается в том, чтобы при фиксированном p найти алгоритмические операторы $B_1, \ldots, B_p$ из $\mathfrak{M}^0$ и корректирующую операцию F из $\mathfrak{F}$, при которых достигается минимум функционала $Q(F(B_1, \ldots, B_p), I_q, \widetilde{I}_q)$. В дальнейшем при записи функционала качества условимся опускать аргументы I_q и $\widetilde{I}_q$, входящие в постановку задачи Z.

Близкая к данной задача состоит в том, чтобы при заданном $\varepsilon \geqslant 0$ найти минимальное число p алгоритмических операторов $B_1, \ldots, B_p$ из $\mathfrak{M}^0$ и корректирующую операцию F из $\mathfrak{F}$, при которых $Q(F(B_1, \ldots, B_p)) \leqslant \varepsilon$.

Минимизация функционала качества одновременно по всем алгоритмическим операторам и корректирующей операции является в общем случае достаточно сложной задачей. Методы, основанные на решении таких задач, развиты только для некоторых узких классов постановок. Например, в методе комитетов [24, 30] используется одноэлементное семейство корректирующих операций,

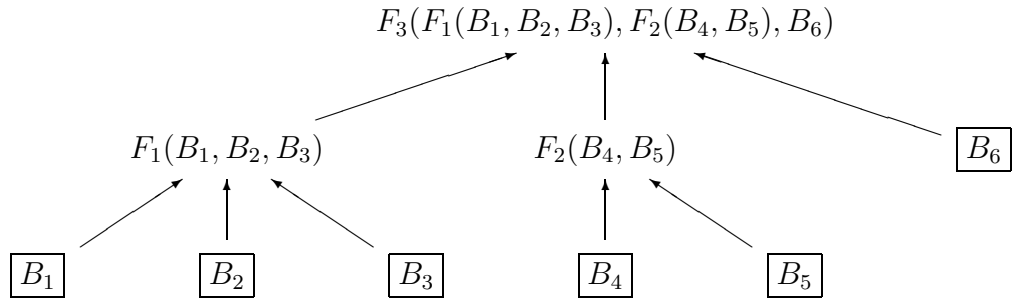


Рис. 2: *Пример. Суперпозиция алгоритмических операторов и корректирующих операций уровня вложенности 2.*

обычно основанное на принципе голосования.

В данной работе развивается общий подход, основанный на поочерёдной настройке алгоритмических операторов и корректирующей операции. Используется итерационный процесс, на каждом шаге которого решается задача минимизация функционала качества $Q(F(B_1,\ldots,B_p))$ либо по корректирующей операции F, либо по одному из операторов B_r, $1 \leqslant r \leqslant p$, при условии, что остальные операторы фиксированы.

При фиксации некоторых алгоритмических операторов предполагается, что к моменту решения данной оптимизационной задачи они уже сконструированы и надлежащим образом настроены. Разумеется, не исключается возможность их построения в виде суперпозиции других алгоритмических операторов и корректирующих операций.

В общем случае это приводит к возникновению суперпозиции алгоритмических операторов и корректирующих операций произвольной вложенности. Такую суперпозицию удобно представлять в виде дерева, терминальные вершины (листья) которого соответствуют алгоритмическим операторам, а нетерминальные — корректирующим операциям, см. рис. 2. Всю суперпозицию в целом можно рассматривать как одну корректирующую операцию F_* над операторами $B_1,\ldots,B_T$, где T — число терминальных вершин дерева. Настройка суперпозиции $F_*(B_1,\ldots,B_T)$ сводится к решению последовательности оптимизационных задач, в которой явным образом учитывается структура суперпозиции.

Использование произвольных суперпозиций в качестве корректирующих операций имеет ряд преимуществ.

1. Суперпозицию можно наращивать постепенно, пока не будет достигнуто приемлемое качество обучения $Q(B)$ для оператора $B = F_*(B_1,\ldots,B_T)$. При этом формируется набор операторов $(B_1,\ldots,B_T)$, который (за счёт использования оптимизации) оказывается локальным базисом.
2. Появляется бóльшая гибкость при настройке на решаемую задачу, так как можно варьировать структуру суперпозиции. Более того, на каждом шаге имеется возможность выбора наиболее адекватных: модели $\mathfrak{M}^0$, семейства $\mathfrak{F}$, функционала качества Q и метода его оптимизации.

3. Минимизация функционала $Q(F(B_1,\ldots,B_p))$ итеративными методами, описанными ниже, не намного сложнее традиционной оптимизации функционала $Q(B)$ в рамках отдельной эвристической модели $\mathfrak{M}^0$. Это позволяет не только применять известные методы или их незначительные модификации, но и ставить задачу одновременной настройки алгоритмического оператора как на исходные прецеденты, так и на компенсацию неточностей, допущенных другими операторами.
4. Не исключается построение одноуровневых суперпозиций. Тем самым данный подход включает многие из ранее применявшихся как частные случаи.

Рассмотрим теперь общий подход к оптимизации качества алгоритмических суперпозиций, основанный на итеративной поочерёдной настройке алгоритмических операторов и корректирующих операций [6, 7, 8, 10].

В качестве нулевого приближения возьмём оператор B_1 из модели $\mathfrak{M}^0$, найденный путём минимизации функционала $Q(B)$. Следующие базисные операторы B_2, $B_3,\ldots$ будем строить по очереди, причём после добавления очередного оператора будем повторно оптимизировать ранее построенные операторы и корректирующую операцию. При этом на каждом шаге итерационного процесса решается одна из двух задач: функционал качества минимизируется либо по оператору B_r при фиксированных F и $B_1,\ldots,B_{r-1}$, $B_{r+1},\ldots,B_p$:

$$B_r^* = \arg\min_{B_r\in\mathfrak{M}^0} Q(F(B_1,\ldots,B_r,\ldots,B_p)), \quad 1\leqslant r\leqslant p, \tag{1.1}$$

либо по корректирующей операции F при фиксированных $B_1,\ldots,B_p$:

$$F^* = \arg\min_{F\in\mathfrak{F}} Q(F(B_1,\ldots,B_p)). \tag{1.2}$$

В принципе имеется также возможность решать задачи совместной оптимизации оператора B_r и корректирующей операции F:

$$(B_r^*, F^*) = \arg\min_{B_r\in\mathfrak{M}^0,\, F\in\mathfrak{F}} Q(F(B_1,\ldots,B_r,\ldots,B_p)), \quad 1\leqslant r\leqslant p,$$

а также совместной оптимизации нескольких алгоритмических операторов. Однако в настоящей работе такие задачи не рассматриваются.

Последовательность решения задач (1.1) и (1.2), а также очерёдность настройки операторов, в общем случае не фиксируются. На практике очерёдность настройки выбирается в зависимости от различных конкретных обстоятельств, включая специфику решаемой задачи и особенности применяемых семейств $\mathfrak{F}$ и $\mathfrak{M}^0$.

Мы также не рассматриваем критерии останова данного итерационного процесса и считаем, что в зависимости от конкретной задачи используется один из стандартных способов останова. Итерации можно прекращать по достижению заданных: числа итераций, значения функционала качества или скорости его уменьшения.

Описанный процесс представляет собой вариант покоординатного спуска с тем отличием, что определение каждой «координаты» $B_1, \ldots, B_p$ и F требует решения отдельной, как правило многопараметрической, оптимизационной задачи.

В дальнейшем исследование описанного итерационного процесса в конкретных случаях проводится преимущественно по одной и той же схеме.

Сначала рассматриваются методы поиска корректирующей операции при фиксированных алгоритмических операторах (1.2). Обычно они строятся независимо от модели $\mathfrak{M}^0$ и определяются выбором семейства $\mathfrak{F}$ и функционала качества Q.

Затем изучаются оптимизационные задачи вида (1.1), в которых производится настройка одного алгоритмического оператора. Наряду с (1.1) рассматривается исходная задача оптимизации

$$B_r^* = \arg\min_{B_r \in \mathfrak{M}^0} Q(B_r). \tag{1.3}$$

Оказывается, что для наиболее широко используемых семейств корректирующих операций задачи (1.3) и (1.1) практически эквивалентны. Как правило, они принадлежат одному классу задач и отличаются только значениями параметров. Во всех случаях эти отличия легко проинтерпретировать как использование одной из двух основных идей коррекции:

- более точно настроиться на выделенном подмножестве объектов при возможном увеличении погрешностей на остальной части обучающей выборки;
- настроиться на компенсацию совокупной ошибки остальных операторов.

Технически это реализуется по-разному в зависимости от множества $\mathfrak{I}_f$ и семейства $\mathfrak{F}$, например в виде появления или изменения весовых коэффициентов объектов обучения, модификации вектора свободных членов, появления дополнительных ограничений типа неравенств, превращения однородной системы неравенств в неоднородную, и т.д.

Указанный факт позволяет в большинстве случаев не рассматривать конкретные методы оптимизации (1.1) для отдельных моделей алгоритмических операторов, а ограничиться получением постановок задач, аналогичных тем, которые возникают при решении задачи (1.3). Соответствующая корректировка численных методов становится в результате такого сведения несложным техническим упражнением.

Итак, в каждом конкретном случае мы приходим к формулированию пары схожих оптимизационных проблем (1.3) и (1.1). На их основе синтезируется комбинированная постановка задачи, сочетающая в себе оба критерия настройки. Степень близости полученной задачи к первой или второй можно параметризовать числовым параметром $\lambda \in [0, 1]$. Один из способов параметризации состоит в построении функционала качества

$$Q_\lambda(B_r) = (1 - \lambda)\, Q(B_r) + \lambda\, Q(F(B_1, \ldots, B_r, \ldots, B_p)). \tag{1.4}$$

Другой способ основан на более детальном рассмотрении постановок задач, к которым сводятся (1.3) и (1.1). Допустим, что они принадлежат одному классу задач и характеризуются векторами параметров α_1 и α_2 соответственно. Если множество допустимых значений параметров является линейным векторным пространством, то задача с вектором параметров $(1-\lambda)\alpha_1+\lambda\alpha_2$ также принадлежит данному классу, и её решение может рассматриваться в качестве компромиссной стратегии настройки.

От способа параметризации в общем случае требуется только, чтобы при граничных значениях параметра λ реализовались «чистые» стратегии настройки, а именно:

— при $\lambda = 0$ должна решаться задача (1.3), состоящая в настройке алгоритмического оператора B_r на исходные прецеденты без учёта его дальнейшего использования в качестве аргумента корректирующей операции;

— при $\lambda = 1$ должна решаться задача (1.1), в которой оператор B_r настраивается исключительно на компенсацию неточностей, допущенных остальными операторами.

Каждая из этих двух «чистых» стратегий настройки имеет свои недостатки.

В первом случае, многократно решая одну и ту же задачу, мы получаем одинаковые операторы $B_1, \ldots, B_p$. Даже если мы пытаемся варьировать метод настройки, в общем случае нет никакой гарантии, что мы не получим набор из почти одинаковых операторов. В таком случае никакая корректирующая операция не позволит построить алгоритм, качественно отличающийся от уже имеющихся. Это приведёт либо к неоправданному наращиванию числа операторов, либо вообще к отказу от идеи коррекции.

Во втором случае модель операторов $\mathfrak{M}^0$ используется для компенсации суммарной ошибки остальных операторов, но не для аппроксимации зависимости между начальными и финальными информациями. Подобное использование представляется нецелесообразным для проблемно-ориентированных моделей, структура которых «по построению» учитывает специфические особенности этой зависимости.

При промежуточных значениях параметра λ образуются комбинированные стратегии настройки, приводящие к компромиссу между настройкой на прецеденты и компенсацией неточности остальных операторов. Появление дополнительного параметра повышает гибкость и настраиваемость суперпозиции в целом. На практике этот параметр назначается из априорных соображений, учитывающих особенности выбранной модели $\mathfrak{M}^0$ и семейства $\mathfrak{F}$, их универсальность и адекватность решаемой задаче.

1.4 Проблемно-независимые и проблемно-зависимые подзадачи

При построении алгоритмической суперпозиции $F_*(B_1, \ldots, B_T)$ возникают подзадачи двух принципиально различных типов.

Подзадачи первого типа условимся называть проблемно-независимыми. Они легко формализуются, для их решения могут быть использованы различные математические методы, применимость которых не зависит от предметной области и конкретной прецедентной информации $\langle I_q, \widetilde{I}_q \rangle$. К числу проблемно-независимых относятся сформулированные выше задачи оптимизации (1.1)–(1.4).

Численные методы их решения, точные или приближённые, могут быть построены и рассмотрены после конкретизации семейства $\mathfrak{F}$, функционала качества Q и модели $\mathfrak{M}^0$. Функционал качества, в свою очередь, определяется выбором множества $\mathfrak{I}_f$ и семейства решающих правил $\mathfrak{M}^1$.

Ниже будут рассмотрены три семейства корректирующих операций $\mathfrak{F}$, широко применяемые на практике:

— линейные;

— полиномиальные;

— монотонные.

Каждое из них приводит к различным постановкам оптимизационных задач в зависимости от выбора множества $\mathfrak{I}_f$:

— $\mathfrak{I}_f = \{0, 1\}$ для задач классификации с двумя непересекающимися классами;

— $\mathfrak{I}_f = \mathbb{R}$ для задач восстановления регрессии.

Подзадачи второго типа условно назовём проблемно-зависимыми. Они решаются из априорных или эвристических соображений, в каждом конкретном случае по-разному, с учётом специфики предметной области и реальных данных $\langle I_q, \widetilde{I}_q \rangle$. К числу проблемно-зависимых подзадач относятся:

— выбор структуры суперпозиции F_*;

— выбор модели $\mathfrak{M}^0$, семейства $\mathfrak{F}$, функционала качества Q и метода оптимизации в каждой из задач (1.1)–(1.4);

— выбор очерёдности решения задач (1.1)–(1.2) и критерия останова;

— выбор параметра настройки λ в задачах вида (1.4).

Для решения подзадач такого типа разработано специальное инструментальное средство — язык описания алгоритмических суперпозиций ASDIEL. В этом языке проблемно-независимые подзадачи реализованы максимально компактно — в виде алгоритмов настройки и вычисления, задаваемых только своими входами и выходами. В то же время основные выразительные средства языка направлены на эффективную запись той части информации о суперпозиции, которую невозможно получить путём автоматической настройки. Эта информация задаётся исследователем на основе его содержательных представлений о решаемой задаче.

Язык ASDIEL позволяет подбирать оптимальную структуру суперпозиции в ходе вычислительных экспериментов на реальных прецедентных данных. Он берёт на себя функции загрузки и предварительной обработки данных, выдачи отладочной информации и визуализации, что превращает его в удобное средство экспериментальной настройки алгоритмических суперпозиций.

Таким образом, для решения проблемно-зависимых подзадач предоставляется не формальный метод, а прикладное программное обеспечение, предназначенное для проведения серий вычислительных экспериментов. Исследователь может описать какую угодно суперпозицию, и по результатам тестов определить, является ли она приемлемым решением задачи Z, и если нет — то каким образом её следует подкорректировать.

2 Решение задач оптимизации при построении локальных базисов

В этой главе рассматриваются методы построения оптимальных алгоритмических операторов и корректирующих операций. Исследуются три семейства корректирующих операций: линейные, полиномиальные и монотонные; каждое — для двух типов исходных задач: классификации и восстановления регрессии. Для каждого из шести случаев рассматриваются четыре оптимизационные задачи:

1. минимизация функционала качества $Q(B)$ без учёта дальнейшего использования полученного оператора в корректирующей операции;
2. минимизация функционала $Q(F(B_1, \ldots, B_p))$ по алгоритмическому оператору B_p при фиксированных $B_1, \ldots, B_{p-1}$;
3. построение комбинированного критерия настройки в виде комбинации двух предыдущих функционалов качества и его минимизация по оператору B_p при фиксированных $B_1, \ldots, B_{p-1}$;
4. минимизация функционала качества по корректирующей операции при фиксированных алгоритмических операторах.

При этом преследуется цель свести постановку второй и третьей задач к форме, максимально близкой к первой задаче. В рассматриваемых случаях решение всех трёх задач удаётся свести к применению одних и тех же численных методов. Таким образом, имеющийся арсенал методов, предназначенных для решения задач первого типа, путём незначительных изменений, либо вообще в неизменном виде, переносится на задачи построения локальных базисов. Этот факт позволяет обойтись без рассмотрения конкретных моделей алгоритмических операторов, ограничившись сведéнием возникающих оптимизационных задач к уже известным.

Специального рассмотрения требуют методы построения оптимальных корректирующих операций. Во всех случаях возникающие постановки задач определяются выбором семейств $\mathfrak{F}$, $\mathfrak{M}^1$ и природой множества финальных информаций, но никак не зависят от используемых моделей алгоритмических операторов. Разработка новых численных методов потребовалась только в случае монотонных корректирующих операций.

2.1 Линейные корректирующие операции

Возьмём в качестве пространства оценок множество действительных чисел, $\mathfrak{I}_e = \mathbb{R}$, а в качестве множества $\mathfrak{F}$ — семейство линейных корректирующих операций:

$$\mathfrak{F}_L = \Big\{ F(B_1, \ldots, B_p) = \sum_{i=1}^{p} \alpha_i B_i \;\Big|\; \alpha_i \in \mathbb{R},\ i = 1, \ldots, p \Big\}.$$

Каждая линейная корректирующая операция однозначно задаётся набором из p параметров $\alpha_1, \ldots, \alpha_p$. Иногда на параметры накладывают дополнительные ограничения, например требуют их неотрицательности, и/или чтобы сумма их модулей равнялась единице.

Линейные корректирующие операции над множествами некорректных алгоритмов были впервые введены Ю.И. Журавлёвым [14, 15] при рассмотрении задачи классификации.

2.1.1 Задача восстановления регрессии

Задача восстановления регрессии является частным случаем задачи обучения по прецедентам при $\mathfrak{I}_f = \mathbb{R}$.

Возьмём в качестве семейства решающих правил $\mathfrak{M}^1$ множество, состоящее из единственного тождественного отображения $C(B) \equiv B$. Фактически это означает, что решающие правила не используются.

Положим функционал качества Q равным среднеквадратичной невязке:

$$Q(B) = \sum_{k=1}^{q} (B(x_k) - y_k)^2.$$

Допустим без ограничения общности, что в наборе алгоритмических операторов $B_1, \ldots, B_p$ все операторы, кроме последнего, фиксированы, и решается задача оптимизации оператора B_p. Введём матричные обозначения:

$$\begin{aligned} Z &= [B_i(x_k)]_{k=1,q}^{i=1,p-1}, \\ z &= [B_p(x_k)]_{k=1,q}, \\ a &= [\alpha_i]_{i=1,p-1}, \\ y &= [y_k]_{k=1,q}. \end{aligned}$$

Тогда

$$Q(B_p) = \|z - y\|^2; \tag{2.1}$$

$$Q(F(B_1, \ldots, B_p)) = \|Za + z\alpha_p - y\|^2. \tag{2.2}$$

Задача минимизации функционала $Q(F(B_1, \ldots, B_p))$ по корректирующей операции F при фиксированных алгоритмических операторах сводится к нахождению вектора (a, α_p) и легко решается методом наименьших квадратов.

Рассмотрим задачу минимизации этого функционала по оператору B_p при фиксированных $B_1, \ldots, B_{p-1}$ и F. Путём элементарных преобразований он приводится к виду

$$Q(F(B_1, \ldots, B_p)) = \alpha_p^2 \, \|z - (y - Za)/\alpha_p\|^2 = \alpha_p^2 \, \|z - \widetilde{y}\|^2,$$

где $\widetilde{y} = (y - Za)/\alpha_p$. Полученная задача оптимизации формально эквивалентна задаче минимизации «простейшего» функционала (2.1) и может быть решена теми же самыми численными методами.

Пусть теперь λ — действительное число из отрезка $[0,1]$. Определим функционал Q_λ как взвешенную сумму функционалов (2.1) и (2.2):

$$Q_\lambda(B_p) = (1-\lambda)\,\|z-y\|^2 + \lambda\alpha_p^2\,\|z-\widetilde{y}\|^2.$$

Приводя подобные, получаем, что задача оптимизации данного функционала эквивалентна задаче минимизации функционала

$$Q'_\lambda(B_p) = \|z-y'(\lambda)\|^2, \quad \text{где} \quad y'(\lambda) = \frac{(1-\lambda)y + \lambda\alpha_p^2\widetilde{y}}{1-\lambda+\lambda\alpha_p^2}. \tag{2.3}$$

При $\lambda = 0$ оператор B_p настраивается на исходные прецеденты, при $\lambda = 1$ — на компенсацию неточности остальных операторов. При промежуточных значениях λ получаются комбинированные стратегии настройки.

Итак, задачи минимизации трёх функционалов (2.1), (2.2) и (2.3) формально эквивалентны и отличаются только целевым вектором: y, $\widetilde{y}$, $y'(\lambda)$. Для их решения можно применять один и тот же численный метод.

2.1.2 Задача классификации

Рассмотрим применение семейства линейных корректирующих операций для решения задачи классификации.

Задача классификации (распознавания образов) с l классами является частным случаем задачи обучения по прецедентам при $\mathfrak{I}_f = \{0,1,\ldots,l-1\}$. Для наглядности рассмотрим самый простой случай — классификацию с двумя непересекающимися классами, $l = 2$.

Положим $\mathfrak{I}_e = \mathbb{R}$. Возьмём в качестве $\mathfrak{M}^1$ однопараметрическое семейство пороговых решающих правил

$$\mathfrak{M}^1 = \{C(B) = \theta(B-c) \mid c \in \mathbb{R}\}.$$

где θ — функция Хевисайда.

Положим функционал качества Q равным числу ошибочных классификаций объектов обучения при условии оптимального выбора решающего правила:

$$Q(B) = \min_{c\in\mathbb{R}} \sum_{k=1}^{q} |\theta(B(x_k)-c) - y_k|.$$

Перенумеруем исходные прецеденты $\{x_k\}_{k=1}^q$ таким образом, чтобы первые q_0 элементов принадлежали первому классу, а остальные $(q-q_0)$ элементов — второму классу. Таким образом, $y_1 = \ldots = y_{q_0} = 0$ и $y_{q_0+1} = \ldots = y_q = 1$. Легко видеть, что значение функционала равно наименьшему числу невыполненных неравенств в системе

$$\begin{cases} B(x_k) \leqslant c, & k = 1,\ldots,q_0; \\ B(x_k) > c, & k = q_0+1,\ldots,q; \end{cases}$$

Допустим без ограничения общности, что в наборе алгоритмических операторов $B_1, \ldots, B_p$ все операторы кроме последнего фиксированы и решается задача оптимизации оператора B_p. Введём матричные обозначения:

$$\begin{array}{ll} Z_0 = [B_i(x_k)]_{k=1,q_0}^{i=1,p-1}, & Z_1 = [B_i(x_k)]_{k=q_0+1,q}^{i=1,p-1}, \\ z_0 = [B_p(x_k)]_{k=1,q_0}, & z_1 = [B_p(x_k)]_{k=q_0+1,q}, \\ a = [\alpha_i]_{i=1,p-1}. & \end{array}$$

Тогда значение функционала $Q(B_p)$ равно наименьшему числу невыполненных неравенств в системе

$$\begin{cases} z_0 \leqslant c; \\ z_1 > c; \end{cases} \tag{2.4}$$

а значение функционала $Q(F(B_1, \ldots, B_p))$ — наименьшему числу невыполненных неравенств в системе

$$\begin{cases} Z_0 a + z_0 \alpha_p \leqslant c; \\ Z_1 a + z_1 \alpha_p > c. \end{cases} \tag{2.5}$$

Задача минимизации функционала $Q(F(B_1, \ldots, B_p))$ по корректирующей операции F при фиксированных алгоритмических операторах сводится к поиску максимальной совместной подсистемы системы линейных неравенств (2.5) с неизвестными a, α_p и c. Для решения этой задачи можно использовать стандартные численные методы [24, 25, 30, 47].

Задача минимизации этого же функционала по оператору B_p при фиксированных $B_1, \ldots, B_{p-1}$ и F сводится к поиску максимальной совместной подсистемы системы неравенств

$$\begin{cases} z_0 \leqslant (c - Z_0 a)/\alpha_p; \\ z_1 > (c - Z_1 a)/\alpha_p; \end{cases} \quad \text{либо} \quad \begin{cases} z_0 \geqslant (c - Z_0 a)/\alpha_p; \\ z_1 < (c - Z_1 a)/\alpha_p; \end{cases}$$

при $\alpha_p > 0$ и $\alpha_p < 0$ соответственно. Данная задача формально эквивалентна задаче (2.4).

Построим теперь комбинированный критерий настройки при произвольном λ из отрезка $[0, 1]$. Выберем $\lceil \lambda q \rceil$ номеров $k_1, \ldots, k_{\lceil \lambda q \rceil}$ из множества $\{1, \ldots, q\}$ и потребуем, чтобы для объектов обучающей выборки $x_{k_1}, \ldots, x_{k_{\lceil \lambda q \rceil}}$ выполнялись соответствующие им неравенства системы (2.5), а для остальных элементов — неравенства системы (2.4),

При $\lambda = 0$ получаем систему (2.4), приводящую к настройке оператора B_p на исходные прецеденты. При $\lambda = 1$ получаем систему (2.5), приводящую к настройке на компенсацию неточности остальных операторов.

При промежуточных значениях λ имеем всевозможные комбинированные стратегии настройки. Использование такой стратегии сводится к выбору подмножества точек обучающей выборки, на которых желательно достичь лучшего качества распознавания. Методом решения по-прежнему остаётся поиск максимальной совместной подсистемы.

2.2 Полиномиальные корректирующие операции

Возьмём в качестве пространства оценок множество действительных чисел, $\mathfrak{I}_e = \mathbb{R}$, и определим семейство полиномиальных корректирующих операций:

$$\mathfrak{F}_P = \left\{ F(B_1, \ldots, B_p) = \sum_{j=1}^{s} \alpha_j B_{p_{j-1}+1} \ldots B_{p_j} \;\middle|\; \alpha_j \in \mathbb{R},\ j = 1, \ldots, s \right\}.$$

Каждая полиномиальная корректирующая операция однозначно определяется числовой последовательностью $0 = p_0 < p_1 < \ldots < p_{s-1} < p_s = p$, которая задаёт структуру полинома, и s параметрами $\alpha_1, \ldots, \alpha_s$. Иногда на параметры накладывают дополнительные ограничения неотрицательности и/или нормировки.

Фактически рассмотрение полиномиальных корректирующих операций проводится в спрямляющем пространстве, поэтому результаты в большой степени аналогичны полученным в предыдущем параграфе.

Полиномиальные корректирующие операции над множествами некорректных алгоритмов были впервые введены Ю.И. Журавлёвым [14, 15] при рассмотрении задачи классификации.

2.2.1 Задача восстановления регрессии

Так же, как и в §2.1.1, положим, что $\mathfrak{I}_f = \mathfrak{I}_e = \mathbb{R}$, семейство решающих правил $\mathfrak{M}^1$ состоит из единственного тождественного отображения, функционал Q равен среднеквадратичной невязке.

Допустим без ограничения общности, что настраивается оператор B_p, в то время как все остальные операторы фиксированы. Введём матричные обозначения:

$$Z = [B_{p_{j-1}+1}(x_k) \ldots B_{p_j}(x_k)]_{k=1,q}^{j=1,s-1},$$
$$w = [B_{p_{s-1}+1}(x_k) \ldots B_{p-1}(x_k)]_{k=1,q},$$
$$z = [B_p(x_k)]_{k=1,q}, \qquad a = [\alpha_j]_{j=1,s-1}, \qquad y = [y_k]_{k=1,q}.$$

Тогда

$$Q(B_p) = \|z - y\|^2; \tag{2.6}$$
$$Q(F(B_1, \ldots, B_p)) = \|Za + wz - y\|^2; \tag{2.7}$$

где под умножением векторов понимается покомпонентное (адамарово) умножение.

Задача минимизации функционала $Q(F(B_1, \ldots, B_p))$ по корректирующей операции F при фиксированных алгоритмических операторах легко решается методом наименьших квадратов.

Рассмотрим задачу минимизации этого функционала по оператору B_p при фиксированных $B_1, \ldots, B_{p-1}$ и F. Путём элементарных преобразований он приводится к виду

$$Q(F(B_1, \ldots, B_p)) = \sum_{k=1}^{q} w_k^2 \, (z_k - \widetilde{y}_k)^2,$$

где $\widetilde{y} = (y - Za)/w$, деление векторов покомпонентное (проблема с делением на 0 не возникает, так как в случае $w_k = 0$ соответствующее k-ое слагаемое можно просто убрать из функционала). Полученный функционал отличается от (2.6) только появлением весовых коэффициентов $w_1, \ldots, w_q$, которые фактически являются весами объектов обучения. Численные методы оптимизации этого функционала могут быть получены на основе методов, используемых для минимизации (2.6) с учётом поправки на весовые коэффициенты.

Комбинированный критерий настройки для нахождения оператора B_p при заданном действительном λ из отрезка $[0, 1]$ имеет вид

$$Q_\lambda(B_p) = (1 - \lambda) \sum_{k=1}^{q} (z_k - y_k)^2 + \lambda \sum_{k=1}^{q} w_k^2 \, (z_k - \widetilde{y}_k)^2.$$

Приводя подобные, получаем, что задача оптимизации данного функционала эквивалентна задаче минимизации функционала

$$Q'_\lambda(B_p) = \sum_{k=1}^{q} (1 - \lambda + \lambda w_k^2)(z_k - y'_k(\lambda))^2, \quad \text{где} \quad y'_k(\lambda) = \frac{(1 - \lambda) y_k + \lambda w_k^2 \widetilde{y}_k}{1 - \lambda + \lambda w_k^2}. \tag{2.8}$$

При $\lambda = 0$ оператор B_p настраивается на исходные прецеденты, при $\lambda = 1$ — на компенсацию неточности остальных операторов.

Итак, задачи минимизации трёх функционалов (2.6), (2.7) и (2.8) формально эквивалентны и отличаются только конкретными значениями целевого вектора и вектора весов объектов обучения. Для их решения можно применять один и тот же численный метод.

2.2.2 Задача классификации

Рассмотрим семейство полиномиальных корректирующих операций применительно к задачам классификации с двумя непересекающимися классами. Положим, как и в §2.1.2, что $\mathfrak{I}_f = \{0, 1\}$, $\mathfrak{I}_e = \mathbb{R}$; $\mathfrak{M}^1$ — семейство пороговых решающих правил; функционал Q равен числу ошибок классификации выборки I_q при условии оптимального выбора решающего правила; объекты обучения перенумерованы таким образом, что $y_1 = \ldots = y_{q_0} = 0$ и $y_{q_0+1} = \ldots = y_q = 1$.

Пусть $\mathfrak{F} = \mathfrak{F}_P$ — семейство полиномиальных корректирующих операций, введённое в §2.2.1. Перейдём к матричным обозначениям:

$$\begin{aligned}
&Z_0 = [B_{p_{j-1}+1}(x_k) \ldots B_{p_j}(x_k)]_{k=1,q_0}^{j=1,s-1},\\
&Z_1 = [B_{p_{j-1}+1}(x_k) \ldots B_{p_j}(x_k)]_{k=q_0+1,q}^{j=1,s-1},\\
&w_0 = [B_{p_{s-1}+1}(x_k) \ldots B_{p-1}(x_k)]_{k=1,q_0},\\
&w_1 = [B_{p_{s-1}+1}(x_k) \ldots B_{p-1}(x_k)]_{k=q_0+1,q},\\
&z_0 = [B_p(x_k)]_{k=1,q_0}, \qquad z_1 = [B_p(x_k)]_{k=q_0+1,q}, \qquad a = [\alpha_j]_{j=1,s-1}.
\end{aligned}$$

Задачи минимизации функционалов $Q(B_p)$ и $Q(F(B_1, \ldots, B_p))$ по оператору $B_p \in \in \mathfrak{M}^0$ сводятся к поиску максимальных совместных подсистем в системах неравенств, соответственно,

$$\begin{cases} z_0 \leqslant c; \\ z_1 > c; \end{cases} \qquad \text{и} \qquad \begin{cases} w_0 z_0 \leqslant (c - Z_0 a); \\ w_1 z_1 > (c - Z_1 a); \end{cases} \tag{2.9}$$

где умножение векторов покомпонентное. Обе задачи формально эквивалентны.

Комбинированная критерий настройки строится так же, как и в §2.1.2, и задача снова сводится к выбору подмножества объектов обучения, на которых желательно достичь лучшего качества обучения, и последующему поиску максимальной совместной подсистемы.

2.3 Монотонные корректирующие операции

Целесообразность использования монотонных корректирующих операций вытекает из следующего соображения. Допустим, что все алгоритмические операторы $B_1, \ldots, B_p$ настроены на аппроксимацию одной и той же зависимости. Тогда разумно потребовать, чтобы одновременное увеличение (уменьшение) их выходных значений не приводило к уменьшению (соответственно увеличению) значения на выходе оператора $F(B_1, \ldots, B_p)$. Но это и означает монотонность F как p-арного отображения. Идея применения монотонных корректирующих операций принадлежит К.В. Рудакову [17, 41].

Пусть $\mathfrak{I}_f$ и $\mathfrak{I}_e$ — произвольные частично упорядоченные множества. Введём на $\mathfrak{I}_e^p$ отношение порядка, положив $(u_1, \ldots, u_p) \leqslant (v_1, \ldots, v_p)$ если $u_i \leqslant v_i$ для всех $i = 1, \ldots, p$. Запись $u \parallel v$ обозначает несравнимость векторов u и v из $\mathfrak{I}_e^p$. Соотношение $u < v$ равносильно тому, что $u \leqslant v$ и $u \neq v$. Отображение $g\colon U \to V$, где U и V — произвольные упорядоченные множества, называется монотонным, если для любых $u_1, u_2 \in U$ из $u_1 \leqslant u_2$ следует $g(u_1) \leqslant g(u_2)$. Приведённые определения ничем не отличаются от классических [31].

В качестве $\mathfrak{F}$ возьмём семейство всех монотонных отображений из $\mathfrak{I}_e^p$ в $\mathfrak{I}_f$ при произвольном натуральном p:

$$\mathfrak{F}_M = \bigcup_{p=0}^{\infty} \Big\{ F : \mathfrak{I}_e^p \to \mathfrak{I}_f \;\Big|\; (\forall u, v \in \mathfrak{I}_e)\;\; u \leqslant v \to F(u) \leqslant F(v) \Big\}.$$

Обратим внимание, что корректирующие операции действуют непосредственно в $\mathfrak{I}_f$, а не в $\mathfrak{I}_e$. Это позволит нам избежать отдельного рассмотрения решающих правил.

2.3.1 Оптимизация базиса при построении корректного алгоритма

Обозначим множество $\{1, \ldots, q\}$ через $\mathbb{Q}$. Введём последовательность $\{a_k\}_{k=1}^q$, состоящую из q векторов $a_k = [B_i(x_k)]_{i=1,p}$, $k \in \mathbb{Q}$. Тогда условие корректности алгоритма $F(B_1, \ldots, B_p)$ примет вид

$$F(a_k) = y_k, \quad \text{для всех } k \in \mathbb{Q}. \tag{2.10}$$

Набор векторов $\{a_k\}_{k=1}^q$ назовём *допустимым*, если для любой пары $(j, k) \in \in \mathbb{Q}^2$ из $y_j \neq y_k$ следует $a_j \neq a_k$. Допустимость является достаточным условием существования некоторого (не обязательно монотонного) отображения F, для которого выполнено (2.10). Далее, не оговаривая особо, будем предполагать, что условие допустимости выполнено.

Определение 3. Пусть $\{f_k\}_{k=1}^q$ — произвольная последовательность элементов множества $\mathfrak{I}_f$. Последовательность пар $\{a_k, f_k\}_{k=1}^q$ будем называть *монотонной*, если для всех $(j,k) \in \mathbb{Q}^2$ из $a_j < a_k$ следует $f_j \leqslant f_k$.

Лемма 1. *Монотонное отображение F, удовлетворяющее (2.10), существует тогда и только тогда, когда $\{a_k, y_k\}_{k=1}^q$ — монотонная последовательность.*

Доказательство. Необходимость очевидна. Докажем достаточность.

Пусть последовательность $\{a_k, y_k\}_{k=1}^q$ монотонна. Определим монотонное отображение F следующим образом:

$$F(a) = \begin{cases} \max\{y_k \mid k \in \mathbb{L}(a)\}, & \text{если } \mathbb{L}(a) \neq \varnothing, \\ \min\{y_1, \ldots, y_q\}, & \text{если } \mathbb{L}(a) = \varnothing, \end{cases}$$

где $\mathbb{L}(a) = \{k \in \mathbb{Q} \mid a_k \leqslant a\}$ для всех $a \in \mathfrak{I}_e^p$. Построенное отображение монотонно и удовлетворяет условию (2.10).

Лемма доказана.

Определение 4. Пара индексов $(j,k) \in \mathbb{Q}^2$ называется *дефектной парой* алгоритмического оператора B, если $y_j < y_k$ и $B(x_j) \geqslant B(x_k)$. Множество всех дефектных пар оператора B обозначим через $\mathbb{D}(B)$.

Непосредственно из доказанной леммы при $p = 1$ вытекает, что $\mathbb{D}(B) = \varnothing$ тогда и только тогда, когда существует монотонное отображение F такое, что $F(B(x_k)) = y_k$ для всех k из $\mathbb{Q}$. Этот факт позволяет определить функционал качества через число дефектных пар алгоритмического оператора: $Q(B) = |\mathbb{D}(B)|$.

Рассмотрим задачу (1.1) минимизации функционала $Q(F(B_1, \ldots, B_p))$ по одному из операторов. Предположим без ограничения общности, что строится оператор B_p при фиксированных $B_1, \ldots, B_{p-1}$.

Нашей ближайшей целью будет исследование множества $\mathbb{D}(F(B_1, \ldots, B_p))$ и его соотношения с множеством $\mathbb{D}(F_{p-1}(B_1, \ldots, B_{p-1}))$, где F_{p-1} — корректирующая операция, полученная на предыдущем шаге итерационного процесса (1.1)–(1.4). В дальнейшем это позволит переформулировать задачу оптимизации (1.1) в виде системы явных ограничений на оператор B_p.

Множество $\mathbb{D}(B_1, \ldots, B_p) = \mathbb{D}(B_1) \cap \ldots \cap \mathbb{D}(B_p)$ назовём *дефектом* набора операторов $B_1, \ldots, B_p$. Введение этого термина оправдывается следующей леммой.

Лемма 2. *Для любой p-арной монотонной корректирующей операции F*

$$\mathbb{D}(F(B_1, \ldots, B_p)) \supseteq \mathbb{D}(B_1, \ldots, B_p). \tag{2.11}$$

Доказательство. Пусть $(j,k) \in \mathbb{D}(B_i)$ для всех $i = 1, \ldots, p$. Тогда $y_j < y_k$ и $a_j \geqslant a_k$. Следовательно для любой корректирующей операции F из $\mathfrak{F}_M$ выполняется $F(a_j) \geqslant F(a_k)$, а значит пара (j,k) дефектная для алгоритма $F(B_1, \ldots, B_p)$.

Лемма доказана.

Дефект $\mathbb{D}(B_1, \ldots, B_p)$ имеет гораздо более прозрачное строение, чем множество $\mathbb{D}(F(B_1, \ldots, B_p))$. А именно, он состоит из тех пар объектов обучения, на которых все p операторов дали неверный порядок, то есть $a_j \geqslant a_k$, в то время как $y_j < y_k$. Отсюда немедленно вытекает стратегия сокращения множества $\mathbb{D}(B_1, \ldots, B_p)$. Оператор B_p необходимо строить с тем расчётом, чтобы он да-

вал правильный порядок на дефектных парах всех остальных алгоритмов. Тем самым векторы a_j и a_k переходят в разряд несравнимых, и дефектная пара исчезает.

К сожалению, в общем случае соотношение (2.11) является именно включением, а не равенством. Поэтому для обоснования и уточнения описанной стратегии необходимо понять, при каких условиях включение обращается в равенство, и из каких пар состоит разность множеств $\mathbb{D}(F(B_1,\ldots,B_p))\setminus\mathbb{D}(B_1,\ldots,B_p)$, когда равенства нет.

В следующей теореме выделен простейший, но важный частный случай, когда соотношение (2.11) обращается в равенство.

Теорема 1. *Дефект $\mathbb{D}(B_1,\ldots,B_p)$ пуст тогда и только тогда, когда в $\mathfrak{F}_M$ найдётся корректирующая операция F, для которой $\mathbb{D}(F(B_1,\ldots,B_p))=\varnothing$.*

Доказательство. Из условия $\mathbb{D}(B_1,\ldots,B_p)=\varnothing$ вытекает, что для любой пары $(j,k)\in\mathbb{Q}^2$ из $a_k\leqslant a_j$ следует $y_k\leqslant y_j$. Значит последовательность $\{a_k,y_k\}_{k=1}^q$ является монотонной, и по лемме 1 существует отображение F, удовлетворяющее (2.10). Очевидно, дефект оператора $F(B_1,\ldots,B_p)$ пуст.

Обратное утверждение докажем от противного. Пусть для некоторой $F\in\mathfrak{F}_M$ выполнено $\mathbb{D}(F(B_1,\ldots,B_p))=\varnothing$, но при этом дефект $\mathbb{D}(B_1,\ldots,B_p)$ не пуст. Тогда найдётся пара индексов $(j,k)\in\mathbb{Q}^2$ такая, что $y_j<y_k$, $a_j\geqslant a_k$ и $F(a_j)<F(a_k)$. Но это противоречит монотонности отображения F.

Теорема доказана.

Следствие. Чтобы получить корректный алгоритм, достаточно построить набор алгоритмических операторов, не имеющий дефекта.

Опираясь на теорему 1, докажем, что при некоторых не слишком сильных ограничениях на модель $\mathfrak{M}^0$ процесс построения корректного алгоритма сходится за конечное число шагов.

Теорема 2 (О сходимости). *Пусть модель $\mathfrak{M}^0$ такова, что для любой пары $(j,k)\in\mathbb{Q}^2$, $j\neq k$, найдётся алгоритмический оператор $B\in\mathfrak{M}^0$, для которого $B(x_j)<B(x_k)$. Тогда при произвольном $B_1\in\mathfrak{M}^0$ потребуется не более $p_*=Q(B_1)+1$ операторов, чтобы $Q(F(B_1,B_2,\ldots,B_{p_*}))=0$ при некоторой $F\in\in\mathfrak{F}_M$.*

Доказательство. Предположим, что оператор B_1 уже выбран. Будем последовательно строить операторы $B_2,B_3,\ldots,B_p,\ldots$ таким образом, чтобы $B_p(x_j)<B_p(x_k)$ для некоторой пары (j,k) из множества $\mathbb{D}(B_1,\ldots,B_{p-1})$. На каждом шаге дефект сокращается, как минимум, на одну пару. Следовательно понадобится не более $|\mathbb{D}(B_1)|$ операторов, чтобы полностью устранить дефект оператора B_1. По теореме 1 существует монотонная корректирующая операция F, для которой $Q(F(B_1,\ldots,B_{p_*}))=0$.

Теорема доказана.

Более сильная оценка числа операторов получается в случае, когда модель $\mathfrak{M}^0$ допускает построение алгоритмического оператора, воспроизводящего правильный порядок на произвольных m парах одновременно. Соответствующая теорема доказывается аналогично.

Теорема 3 (О сходимости). *Пусть модель $\mathfrak{M}^0$ такова, что для любого множества m пар, $m \geqslant 1$,*

$$\{(j_i, k_i) \in \mathbb{Q}^2 \mid j_i \neq k_i,\ i = 1, \ldots, m\}$$

найдётся алгоритмический оператор $B \in \mathfrak{M}^0$, для которого ни одна из этих пар не является дефектной. Тогда при произвольном $B_1 \in \mathfrak{M}^0$ потребуется не более $p_* = \left\lceil \frac{1}{m} Q(B_1) \right\rceil + 1$ алгоритмических операторов, чтобы $Q(F(B_1, \ldots, B_{p_*})) = 0$ при некоторой $F \in \mathfrak{F}_M$.

Таким образом, очередной базисный оператор B_p следует выбирать так, чтобы удовлетворялось как можно больше неравенств

$$B_p(x_j) < B_p(x_k) \quad \text{для всех } (j, k) \in \mathbb{D}(B_1, \ldots, B_{p-1}). \tag{2.12}$$

Данная система в общем случае несовместна, и задача сводится к поиску её максимальной совместной подсистемы.

2.3.2 Оптимизация базиса при фиксированном числе операторов

На практике далеко не всегда целесообразно наращивать число операторов до полного исчерпывания дефекта. Существует, как минимум, две причины, по которым построение базиса приходится останавливать до достижения корректности.

Во-первых, фиксация числа базисных операторов повышает надёжность конструируемого алгоритма. Это следует из результатов [32, 33], опирающихся на статистическую теорию надёжности восстановления зависимостей по эмпирическим данным [1, 9].

Во-вторых, построение корректного алгоритма, точно удовлетворяющего системе равенств (2.10), оказывается необязательным, если финальные информации изначально заданы с некоторой погрешностью. В этом случае целесообразно потребовать приближённого выполнения условий корректности, а процесс построения базиса прервать при достижении заданного достаточно малого значения функционала качества.

При этом возникает вопрос: в каком порядке следует устранять дефектные пары, чтобы значение функционала $Q(B_1, \ldots, B_p)$ убывало как можно быстрее с ростом p. Однако теперь дефект не является пустым множеством, и теорема 1 не даёт никаких подходов к минимизации функционала качества. Для этого случая необходимо более детальное описание множества $\mathbb{D}(F(B_1, \ldots, B_p))$, чем соотношение (2.11).

В случае $\mathbb{D}(B_1, \ldots, B_p) \neq \varnothing$ соотношение (2.11) может быть как равенством, так и строгим включением. Далее будет показано, что, не прибегая к построению корректирующей операции F, возможно указать, из каких элементов состоит разность $\mathbb{D}(F(B_1, \ldots, B_p)) \setminus \mathbb{D}(B_1, \ldots, B_p)$.

Введём на множестве индексов $\mathbb{Q} = \{1, \ldots, q\}$ бинарное отношение $\prec$, положив $j \prec k$ в том и только том случае, когда либо $a_j \leqslant a_k$, либо $a_j \parallel a_k$ и $y_j \leqslant y_k$.

Допустим, что отношение $\prec$ является отношением порядка. Ниже будет показано, что при выполнении этого условия возможно построить монотонную корректирующую операцию F, при которой дефектными парами оператора $F(B_1, \ldots, B_p)$ окажутся только элементы множества $\mathbb{D}(B_1, \ldots, B_p)$. Таким образом, задача сводится к тому, чтобы выявить препятствия, мешающие отношению $\prec$ быть отношением порядка, то есть рефлексивным, антисимметричным и транзитивным.

Заметим во-первых, что отношение $\prec$ не антисимметрично, то есть из $j \prec k$ и $k \prec j$ не следует $j = k$. Это, однако, не мешает расположить элементы множества $1, \ldots, q$ в таком порядке $\{i_1, i_2, \ldots, i_q\}$, чтобы из $s \leqslant t$ следовало $i_s \prec i_t$. Требование антисимметричности в данном случае не играет роли; достаточно, чтобы отношение $\prec$ было предпорядком, то есть только рефлексивным и транзитивным.

Во-вторых, оно не является транзитивным, так как существуют тройки индексов (j, s, k), на которых образуется цикл: $j \prec s \prec k \prec j$.

Определение 5. Тройка индексов $(j, s, k) \in \mathbb{Q}^3$ называется *дефектной тройкой* набора алгоритмических операторов $B_1, \ldots, B_p$, если:

(а) пара (j, k) дефектна для всех B_i, $i = 1, \ldots, p$;

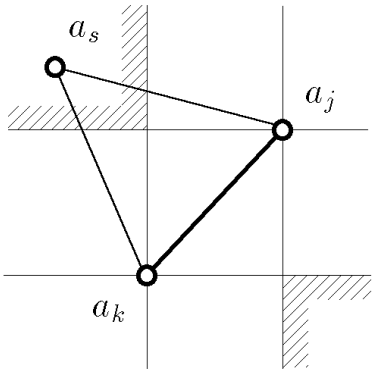
(б) вектор a_s несравним с a_j и a_k;

(в) выполнена цепочка неравенств $y_j \leqslant y_s \leqslant y_k$.

Дефектная тройка (j, s, k) называется *строго дефектной*, если $y_j < y_s < y_k$. Пара (j, k) называется *основанием* дефектной тройки (j, s, k), пары (j, s) и (s, k) — её *рёбрами*, а индекс s — её *вершиной*. Очевидно, основание любой дефектной тройки принадлежит множеству $\mathbb{D}(B_1, \ldots, B_p)$.

Пример 1. Рассмотрим три двумерных вектора $a_j = (3, 2)$, $a_s = (1, 3)$, $a_k = (2, 1)$ и соответствующие им финальные информации: $y_j = 1$, $y_s = 2$, $y_k = 3$. Тройка (j, s, k) строго дефектная. При замене y_s на 1 или 3 она становится не строго дефектной.

На рисунке изображены точки a_j, a_s, a_k, образующие дефектную тройку в пространстве размерности $p = 2$. Штриховкой обозначены области, в которых может находиться вершина дефектной тройки.



Отношение $\prec$ образует цикл на произвольной дефектной тройке (j, s, k), поскольку $j \prec s \prec k$ в силу условий (б) и (в), в то же время $k \prec j$ в силу условия (а). Следующая лемма утверждает, что только дефектные тройки препятствуют отношению $\prec$ быть предпорядком.

Лемма 3. *В $\mathbb{Q}^3$ нет дефектных троек тогда и только тогда, когда отношение $\prec$ является предпорядком на $\mathbb{Q}$.*

Доказательство. Если отношение $\prec$ является предпорядком на $\mathbb{Q}$, то дефектных троек быть не может, так как на них нарушается транзитивность данного отношения.

Для доказательства обратного утверждения покажем, что при отсутствии дефектных троек отношение $\prec$ транзитивно, то есть для любых j, k, s из $\mathbb{Q}$ из $j \prec \prec k$ и $k \prec s$ следует $j \prec s$. Каждое из отношений $j \prec k$ и $k \prec s$ имеет место в одном из двух случаев, поэтому всего возможны четыре варианта.

1. $a_j \leqslant a_k$ и $a_k \leqslant a_s$.

Тогда $a_j \leqslant a_s$ и получаем требуемое $j \prec s$.

2. $a_j \leqslant a_k$ и $a_k \parallel a_s$, $y_k \leqslant y_s$.

Рассмотрим возможные отношения между a_j и a_s. Если $a_j \leqslant a_s$, то получаем требуемое $j \prec s$. Случай $a_s \leqslant a_j$ невозможен, так как иначе было бы $a_s \leqslant a_k$, что противоречит их несравнимости. Пусть теперь $a_j \parallel a_s$. Рассмотрим возможные отношения между y_j и y_k. Если $y_j \leqslant y_k$, то $y_j \leqslant y_s$, и получаем требуемое $j \prec s$. Если $y_j > y_k$, то предположение $y_s \leqslant y_j$ приводит к наличию дефектной тройки (k, s, j), поэтому $y_j < y_s$, следовательно $j \prec s$.

3. $a_j \parallel a_k$, $y_j < y_k$ и $a_k \leqslant a_s$.

Рассмотрим возможные отношения между a_j и a_s. Если $a_j \leqslant a_s$, то получаем требуемое $j \prec s$. Случай $a_s \leqslant a_j$ невозможен, так как иначе было бы $a_k \leqslant a_j$, что противоречит их несравнимости. Пусть теперь $a_j \parallel a_s$. Рассмотрим возможные отношения между y_k и y_s. Если $y_k \leqslant y_s$, то $y_j \leqslant y_s$, и получаем требуемое $j \prec s$. Если $y_k > y_s$, то предположение $y_s \leqslant y_j$ приводит к наличию дефектной тройки (s, j, k), поэтому $y_j < y_s$, следовательно $j \prec s$.

4. $a_j \parallel a_k$, $y_j \leqslant y_k$ и $a_k \parallel a_s$, $y_k \leqslant y_s$.

Рассмотрим возможные отношения между a_j и a_s. Если $a_j \leqslant a_s$, то получаем требуемое $j \prec s$. Случай $a_s \leqslant a_j$ невозможен, так как иначе тройка (j, k, s) была бы дефектной. Если $a_j \parallel a_s$, то в силу $y_j \leqslant y_s$ имеем требуемое $j \prec s$.

Лемма доказана.

Введём на $\mathbb{Q}$ ещё одно бинарное отношение θ, положив $j\theta k$ в том и только том случае, когда $j \prec k$ и $k \prec j$. Легко проверить, что если $\prec$ — предпорядок, то θ — отношение эквивалентности на $\mathbb{Q}$. Два произвольных индекса j и k принадлежат одному классу эквивалентности тогда и только тогда, когда $a_j \parallel a_k$ и $y_j = y_k$.

Следующая теорема утверждает, что включение (2.11) возможно обратить в равенство в том и только в том случае, когда отсутствуют дефектные тройки.

Теорема 4. *Если в $\mathbb{Q}^3$ имеется хотя бы одна строго дефектная тройка, то для любой монотонной корректирующей операции F справедливо строгое включение*

$$\mathbb{D}(F(B_1, \ldots, B_p)) \supset \mathbb{D}(B_1, \ldots, B_p); \tag{2.13}$$

Если в $\mathbb{Q}^3$ нет дефектных троек, то существует такая монотонная корректирующая операция F, что

$$\mathbb{D}(F(B_1, \ldots, B_p)) = \mathbb{D}(B_1, \ldots, B_p). \tag{2.14}$$

Доказательство. Рассмотрим произвольную p-арную монотонную корректирующую операцию F и произвольную строго дефектную тройку (j, s, k). Пусть

f_j, f_s и f_k — значения отображения F в точках a_j, a_s и a_k соответственно. Поскольку $a_k \leqslant a_j$, из монотонности следует $f_k \leqslant f_j$. Если предположить, что ни одна из пар (j, s) и (s, k) не является дефектной, то получится $f_j < f_k$, что противоречит условию монотонности. Значит, хотя бы одна из этих двух пар дефектная. Поскольку $a_j \parallel a_s$ и $a_s \parallel a_k$, то ни одна из них не может содержаться в пересечении $\mathbb{D}(B_1) \cap \ldots \cap \mathbb{D}(B_p)$, и имеет место строгое включение (2.13).

Докажем теперь второе утверждение теоремы. Предполагая, что в $\mathbb{Q}^3$ нет дефектных троек, построим монотонную корректирующую операцию F, для которой верно равенство (2.14).

Упорядочим множество $\mathbb{Q}$ по отношению предпорядка $\prec$. Пусть σ — перестановка элементов множества $\mathbb{Q}$ такая, что из $s < t$ следует $\sigma(s) \prec \sigma(t)$. Образуем последовательность $\{\widetilde{y}_k\}_{k=1}^q$, положив

$$\widetilde{y}_{\sigma(t)} = \max(y_{\sigma(1)}, \ldots, y_{\sigma(t)}), \quad t \in \mathbb{Q}. \tag{2.15}$$

Покажем, что последовательность пар $\{a_k, \widetilde{y}_k\}_{k=1}^q$ монотонна. Возьмём произвольную пару $(j, k) \in \mathbb{Q}^2$, $j \neq k$. Очевидно, $(j, k) = (\sigma(s), \sigma(t))$ для некоторых s и t. По определению отношения $\prec$ из $a_j < a_k$ следует $\sigma(s) \prec \sigma(t)$.

Возможны два случая. Либо $s < t$, тогда из (2.15) сразу получаем требуемое $\widetilde{y}_j \leqslant \widetilde{y}_k$. Либо $t < s$, следовательно $\sigma(t) \prec \sigma(s)$, и все $(s - t + 1)$ индексов, начиная с $\sigma(t)$ и до $\sigma(s)$ включительно лежат в одном классе эквивалентности по отношению θ. Значит $y_{\sigma(t)} = y_{\sigma(t+1)} = \ldots = y_{\sigma(s)}$, откуда с учётом (2.15) вытекает $\widetilde{y}_j = \widetilde{y}_k$.

Монотонность последовательности пар $\{a_k, \widetilde{y}_k\}_{k=1}^q$ доказана. Согласно лемме 1 существует монотонное отображение F такое, что $F(a_k) = \widetilde{y}_k$ для всех $k \in \mathbb{Q}$.

Возьмём произвольную дефектную пару $(j, k) = (\sigma(s), \sigma(t))$ из множества $\mathbb{D}(F(B_1, \ldots, B_p))$ и покажем, что она принадлежит дефекту $\mathbb{D}(B_1, \ldots, B_p)$. По определению 4 выполнено $y_j < y_k$ и $F(a_j) \geqslant F(a_k)$. Из первого неравенства следует, что j и k не могут лежать в одном классе эквивалентности по θ. Второе неравенство приводит к $\widetilde{y}_j \geqslant \widetilde{y}_k$.

Допустим, что условие $a_k \leqslant a_j$ не выполнено. Тогда либо $a_j \leqslant a_k$, либо $a_j \parallel \parallel a_k$. С учётом $y_j < y_k$ заключаем, что в обоих случаях $j \prec k$. Противоположное соотношение $k \prec j$ не может иметь места, так как j и k не эквивалентны. Отсюда вытекает, что $s < t$ и $\widetilde{y}_j \leqslant \widetilde{y}_k$, а значит $\widetilde{y}_{\sigma(s)} = \widetilde{y}_{\sigma(t)}$. Из формулы (2.15) делаем вывод, что $y_{\sigma(s)} \geqslant y_{\sigma(t)}$, но это противоречит условию $y_j < y_k$.

Итак, $a_k \leqslant a_j$, следовательно пара (j, k) является дефектной для всех операторов $B_1, \ldots, B_p$, то есть принадлежит $\mathbb{D}(B_1, \ldots, B_p)$.

Теорема доказана.

Таким образом, множество $\mathbb{D}(F(B_1, \ldots, B_p))$ состоит из дефектных пар трёх типов: (1) элементов дефекта, (2) рёбер дефектных троек, (3) всех остальных пар. Из доказанной теоремы следует, что корректирующую операцию можно выбрать так, чтобы отсутствие пар второго типа автоматически приводило к отсутствию пар третьего типа, а отсутствие пар первого типа — к отсутствию любых дефектных пар. На этом основании предлагается следующий эвристический принцип ми-

нимизации функционала качества $Q(F(B_1,\ldots,B_p))$: последовательно устранять дефектные пары первого типа, стремясь вместе с ними устранить наибольшее число пар второго типа, и вообще не принимать во внимание пары третьего типа.

Согласно этому принципу будем строить оператор B_p с тем расчётом, чтобы как можно больше пар, принадлежащих $\mathbb{D}(B_1,\ldots,B_{p-1})$, исключить из множества $\mathbb{D}(B_1,\ldots,B_p)$. В первую очередь будем исключать пары, лежащие в основании наибольшего количества дефектных троек. Напомним, что для исключения пары (j,k) достаточно потребовать, чтобы B_p удовлетворял условию $B_p(x_j) < B_p(x_k)$.

Обозначим через t_{jk} число дефектных троек с основанием (j,k). Некоторые из них могут оказаться строго дефектными; обозначим их число через t^0_{jk}.

Теорема 5. *Пусть множество $\mathbb{D}(F(B_1,\ldots,B_p))$ состоит только из дефектных пар первых двух типов. Тогда для числа d_{jk} дефектных пар, устраняемых при выполнении условия $B_p(x_j) < B_p(x_k)$, справедлива оценка*

$$t^0_{jk} + 1 \leqslant d_{jk} \leqslant t^0_{jk} + t_{jk} + 1.$$

Доказательство. Возьмём произвольную дефектную тройку (j,s,k) набора операторов $B_1,\ldots,B_p$ и произвольную монотонную корректирующую операцию F.

Покажем, что если тройка строго дефектная, то оператор $F(B_1,\ldots,B_p)$ образует на элементах (j,s,k) либо две, либо три дефектные пары. Одна в любом случае образуется на основании (j,k). При доказательстве теоремы 4 было показано, что хотя бы одно из рёбер (j,s) или (s,k) является дефектной парой. Легко указать случай, когда дефект образуется на обоих рёбрах:

$$\begin{array}{ll} y_j = 1, & F(a_j) = 3; \\ y_s = 2, & F(a_s) = 2; \\ y_k = 3, & F(a_k) = 1. \end{array}$$

Покажем, что в случае не строго дефектной тройки образуется одна или две дефектные пары. Допустим без ограничения общности, что $y_j = y_s < y_k$. Тогда в силу равенства $y_j = y_s$ ребро (j,s) не может быть дефектной парой. Дефект образуется на основании (j,k) и, возможно, на ребре (s,k). Приведём пример, когда ребро (s,k) образует дефектную пару:

$$\begin{array}{ll} y_j = 1, & F(a_j) = 3; \\ y_s = 1, & F(a_s) = 1; \\ y_k = 3, & F(a_k) = 1. \end{array}$$

Легко также построить пример, когда ребро (s,k) не образует дефектной пары:

$$\begin{array}{ll} y_j = 1, & F(a_j) = 4; \\ y_s = 1, & F(a_s) = 1; \\ y_k = 3, & F(a_k) = 3. \end{array}$$

Таким образом, минимальное число дефектных пар, устраняемых при исключении произвольной пары (j,k) из множества $\mathbb{D}(B_1,\ldots,B_p)$, равно $t^0_{jk} + 1$.

Максимальное число устраняемых дефектных пар равно $2t^0_{jk} + (t_{jk} - t^0_{jk}) + 1 = = t^0_{jk} + t_{jk} + 1$.

Теорема доказана.

Сопоставим каждой паре (j,k) из дефекта операторов $B_1, \ldots, B_p$ оценку w_{jk} числа дефектных пар, исключаемых вместе с (j,k). Например, можно положить

$$w_{jk} = t^0_{jk} + \frac{1}{2} t_{jk} + 1.$$

Весовой коэффициент w_{jk} показывает, насколько предпочтительнее устранить дефект именно на данной паре.

Теперь сформулируем задачу оптимизации функционала $Q(F(B_1, \ldots, B_p))$ при фиксированных $B_1, \ldots, B_{p-1}$ в виде системы ограничений на оператор B_p. Рассмотрим систему взвешенных неравенств

$$B_p(x_j) < B_p(x_k) \quad \text{с весом } w_{jk} \quad \text{для всех } (j,k) \in \mathbb{D}(B_1, \ldots, B_{p-1}). \tag{2.16}$$

В общем случае эта система несовместна. Оптимизация функционала качества сводится к поиску её совместной подсистемы с максимальным весом. В качестве веса подсистемы возьмём сумму весов всех входящих в неё неравенств. Отметим, что суммарный вес не является в точности числом дефектных пар, устраняемых при выполнении всех неравенств данной подсистемы, так как множества дефектных пар, устраняемых при выполнении каждого из неравенств, могут иметь непустые пересечения. Веса подсистем, также как и веса отдельных неравенств, являются лишь приблизительными оценками числа устраняемых дефектных пар.

Наряду с (2.16) сформулируем в виде системы неравенств задачу оптимизации функционала $Q(B_p)$ по оператору B_p:

$$B_p(x_j) < B_p(x_k) \quad \text{для всех } (j,k)\colon y_j < y_k \tag{2.17}$$

Оптимизация функционала сводится к поиску максимальной по числу членов совместной подсистемы.

Объединим постановки задач (2.16) и (2.17) для получения комбинированной стратегии настройки. Зададим число λ из отрезка $[0,1]$. Определим веса неравенств как функцию параметра λ:

$$w_{jk}(\lambda) = \begin{cases} 1 + \lambda(t^0_{jk} + \frac{1}{2} t_{jk}), & (j,k) \in \mathbb{D}(B_1, \ldots, B_{p-1}), \\ 1 - \lambda, & (j,k) \notin \mathbb{D}(B_1, \ldots, B_{p-1}) \text{ и } y_j < y_k. \end{cases}$$

Комбинированная стратегия настройки оператора B_p сводится к поиску совместной подсистемы максимального веса для системы взвешенных неравенств

$$B_p(x_j) < B_p(x_k) \quad \text{с весом } w_{jk}(\lambda) \quad \text{для всех } (j,k)\colon y_j < y_k. \tag{2.18}$$

По форме постановки задача (2.18) ничем не отличается от (2.16). Все три задачи (2.16)–(2.18) могут быть решены одним и тем же методом — поиском совместной подсистемы максимального веса. В работе [7] мы ограничились получением постановок данных задач, здесь рассмотрим также и методы их решения.

Основной недостаток полученных постановок в том, что они не являются стандартными для задач обучения по прецедентам. Это препятствует простому переносу известных методов минимизации функционала $Q(B)$ на минимизацию функционала $Q(B_1, \ldots, B_p)$, как это было сделано для линейных и полиномиальных корректирующих операций. В стандартном случае каждое ограничение на оператор B_p относится только к одному объекту обучения, а число ограничений имеет порядок q. В случае монотонных корректирующих операций каждое ограничение относится к паре объектов. Число ограничений получается порядка q^2, причём ограничения очевидным образом зависимы: выполнение условия (2.18) для пар (j,k) и (k,s) по транзитивности влечёт их выполнение для пары (j,s). При построении эффективного численного метода этот факт либо должен учитываться явно, либо система (2.18) должна быть переформулирована в терминах ограничений, относящихся к отдельным объектам, а не парам объектов. Именно второго подхода будем придерживаться в дальнейшем.

Рассмотрим сначала задачу классификации, для которой такое переформулирование оказывается наиболее естественным.

2.3.3 Задача классификации

Положим, как прежде в §2.1.2 и §2.2.2, что решается задача классификации с двумя непересекающимися классами, $\mathfrak{I}_f = \{0,1\}$, $\mathfrak{I}_e = \mathbb{R}$.

Характерный для задач классификации функционал качества, равный числу ошибок классификации выборки I_q, и функционал качества $Q(B) = |\mathbb{D}(B)|$, введённый на стр. 24, очевидным образом взаимосвязаны. Необходимым условием дефектности пары (j,k) является, согласно определению, $y_j < y_k$, откуда следует $y_j = 0$, $y_k = 1$. Таким образом, пара дефектна тогда и только тогда, когда хотя бы на одном из объектов допущена ошибка классификации. В результате каждое из требований монотонности

$$B(x_j) < B(x_k) \quad \text{при} \quad y_j < y_k$$

распадается на два отдельных неравенства

$$B(x_j) \leqslant c, \quad B(x_k) > c,$$

где c — заданная пороговая константа.

Введём следующие обозначения:

$$\begin{aligned}
I_0 &= \{k \in \mathbb{Q} \mid y_k = 0\}, \\
I_1 &= \{k \in \mathbb{Q} \mid y_k = 1\}, \\
D_0^p &= \{k \in \mathbb{Q} \mid \exists j\ (k,j) \in \mathbb{D}(B_1, \ldots, B_p)\}, \\
D_1^p &= \{k \in \mathbb{Q} \mid \exists j\ (j,k) \in \mathbb{D}(B_1, \ldots, B_p)\}.
\end{aligned}$$

Множество индексов D_0^p перечисляет все объекты обучения из первого класса, образующие дефект. Множество индексов D_1^p перечисляет все объекты обучения из второго класса, образующие дефект. Очевидно, что $D_0^{p-1} \subseteq I_0$ и $D_1^{p-1} \subseteq I_1$.

Задача минимизации $Q(B_p)$ сводится к поиску максимальной совместной подсистемы системы неравенств

$$\begin{cases} B_p(x_k) \leqslant c, & k \in I_0; \\ B_p(x_k) > c, & k \in I_1. \end{cases} \tag{2.19}$$

Задача минимизации функционала $Q(B_1, \ldots, B_p)$ по $B_p \in \mathfrak{M}^0$ при фиксированных операторах $B_1, \ldots, B_{p-1}$ сводится к поиску совместной подсистемы с максимальным весом системы неравенств

$$\begin{cases} B_p(x_k) \leqslant c, & \text{с весом } w_k, \quad k \in D_0^{p-1}; \\ B_p(x_k) > c, & \text{с весом } w_k, \quad k \in D_1^{p-1}; \end{cases} \tag{2.20}$$

где веса w_k являются оценкой числа дефектных пар, устраняемых при выполнении k-го неравенства. Например, можно положить

$$w_k = \frac{1}{2} \sum_{j=1}^{q} (w_{jk} + w_{kj}), \tag{2.21}$$

причём считается, что $w_{jk} = 0$ для всех (j, k), не принадлежащих $\mathbb{D}(B_1, \ldots, B_{p-1})$.

Комбинированная стратегия настройки при заданном $\lambda \in [0, 1]$ сводится к поиску совместной подсистемы с максимальным весом в системе неравенств

$$\begin{cases} B_p(x_k) \leqslant c, & \text{с весом } 1 + \lambda w_k, \quad k \in D_0^{p-1}; \\ B_p(x_k) \leqslant c, & \text{с весом } 1 - \lambda, \quad k \in I_0 \setminus D_0^{p-1}; \\ B_p(x_k) > c, & \text{с весом } 1 + \lambda w_k, \quad k \in D_1^{p-1}; \\ B_p(x_k) > c, & \text{с весом } 1 - \lambda, \quad k \in I_1 \setminus D_1^{p-1}; \end{cases} \tag{2.22}$$

При $\lambda = 0$ система (2.22) преобразуется в (2.19) и оператор B_p настраивается на исходные прецеденты. При $\lambda = 1$ она преобразуется в (2.20) и настройка идёт на компенсацию неточности остальных операторов. При промежуточных значениях λ получаются комбинированные стратегии настройки.

Описанная комбинированная стратегия не является единственной возможной. Используем приём из §2.1.2 для построения ещё одной альтернативной стратегии.

Выберем произвольные подмножества $J_0(\lambda)$ и $J_1(\lambda)$ таким образом, чтобы

$$D_0^{p-1} \subseteq J_0(\lambda) \subseteq I_0;$$
$$D_1^{p-1} \subseteq J_1(\lambda) \subseteq I_1;$$

причём мощность этих множеств свяжем с параметром λ соотношением

$$|J_0(\lambda)| + |J_1(\lambda)| = (1 - \lambda)\left(|I_0| + |I_1|\right) + \lambda\left(|D_0^{p-1}| + |D_1^{p-1}|\right).$$

Комбинированная стратегия оптимизации оператора B_p сводится к поиску максимальной совместной подсистемы системы неравенств

$$\begin{cases} B_p(x_k) \leqslant c, & k \in J_0(\lambda); \\ B_p(x_k) > c, & k \in J_1(\lambda); \end{cases}$$

При $\lambda = 0$ множества $J_0(\lambda)$ и $J_1(\lambda)$ расширяются до I_0 и I_1 соответственно, система совпадает с (2.19) и оператор B_p настраивается на исходные прецеденты. При $\lambda = 1$ эти множества сужаются до D_0^{p-1} и D_1^{p-1} соответственно, система совпадает с (2.20) и настройка производится на компенсацию ошибок операторов $B_1, \ldots, B_{p-1}$.

В отличие от предыдущего случая (2.22) здесь комбинированная стратегия строится не путём приписывания уравнениям весов, зависящих от λ, а с помощью явного указания подмножества обучающей выборки, на котором желательно достичь лучшего качества распознавания. По форме эта постановка задачи ничем не отличается от использованной в §2.1.2 для линейных и в §2.2.2 для полиномиальных корректирующих операций.

Отличия между двумя рассмотренными стратегиями, по сути дела, незначительны. В обоих случаях требуется более точно настроить оператор B_p на объектах, образующих дефект остальных операторов. Выбор стратегии на практике должен производиться в зависимости от того,

(а) какую задачу проще решать для данной модели $\mathfrak{M}^0$: находить максимальную совместную подсистему или совместную подсистему максимального веса;

(б) возможно ли обоснованно выбрать подмножества объектов обучения, потребовав для них более точной настройки и отбросив при этом все остальные объекты.

2.3.4 Задача восстановления регрессии

Положим, как в §2.1.1 и §2.2.1, что $\mathfrak{I}_f = \mathfrak{I}_e = \mathbb{R}$. Обычно в задачах восстановления регрессии используют среднеквадратичный функционал качества:

$$Q_0(B) = \sum_{k=1}^{q} (B(x_k) - y_k)^2 \to \min_{B \in \mathfrak{M}^0}. \tag{2.23}$$

В то же время минимизация числа дефектных пар оператора $F(B_1, \ldots, B_p)$ при фиксированных $B_1, \ldots, B_{p-1}$ сводится к поиску совместной подсистемы максимального веса для системы неравенств (см. стр. 31)

$$B_p(x_j) < B_p(x_k) \quad \text{с весом } w_{jk} \quad \text{для всех } (j,k) \in \mathbb{D}(B_1, \ldots, B_{p-1}). \tag{2.24}$$

Различие в постановках задач (2.23) и (2.24) делает невозможным как их решение одним и тем же методом, так и построение комбинированной стратегии настройки оператора B_p. Попытка их прямого объединения приводит к задачам минимизации квадратичного функционала $Q_0(B_p)$ при ограничениях-неравенствах (2.24), что требует разработки новых специальных методов оптимизации для каждой модели $\mathfrak{M}^0$.

По этим причинам имеет смысл свести задачу (2.24) к минимизации функционала, аналогичного (2.23). Разумеется, точное сведение невозможно. Заметим

однако, что выбор функционала качества изначально основан на эвристиках, поэтому любое удобное изменение функционала, в общем сохраняющее цель настройки, является допустимым.

Воспользуемся тем, что близость значений $B_p(x_k)$ к значениям y_k является достаточным условием для выполнения соотношений (2.24). Потребуем, чтобы расстояния $|B_p(x_k) - y_k|$ были не слишком велики для всех k, образующих дефектные пары из множества $\mathbb{D}(B_1, \ldots, B_p)$. Это позволит упростить систему неравенств за счёт некоторого усиления ограничений на оператор B_p.

Введём множества индексов

$$\begin{aligned} D^+_{p-1,k} &= \{j \in \mathbb{Q} \mid (k,j) \in \mathbb{D}(B_1, \ldots, B_{p-1})\}, \quad k = 1, \ldots, q; \\ D^-_{p-1,k} &= \{j \in \mathbb{Q} \mid (j,k) \in \mathbb{D}(B_1, \ldots, B_{p-1})\}, \quad k = 1, \ldots, q; \\ D_{p-1} &= \{k \in \mathbb{Q} \mid D^+_{p-1,k} \cup D^-_{p-1,k} \neq \varnothing\}; \end{aligned}$$

и рассмотрим систему неравенств

$$y_k - \frac{1}{2}\Delta y_k^- < B_p(x_k) < y_k + \frac{1}{2}\Delta y_k^+, \qquad k \in D_{p-1}, \tag{2.25}$$

где

$$\begin{aligned} \Delta y_k^+ &= \min_{j \in D^+_{p-1,k}} (y_j - y_k); \\ \Delta y_k^- &= \min_{j \in D^-_{p-1,k}} (y_k - y_j); \end{aligned}$$

и предполагается, что если множество $D^+_{p-1,k}$ (или $D^-_{p-1,k}$) пусто, то значение Δy_k^+ (или Δy_k^-) не определено и соответствующее неравенство не включается в систему.

Тем самым мы свели систему ограничений, относящихся к парам объектов обучающей выборки, к системе ограничений, относящихся к самим объектам. Система (2.25) в общем случае несовместна. Так же как и в предыдущем параграфе, припишем каждому неравенству весовой коэффициент

$$w_k = \frac{1}{2}\sum_{j=1}^{q}(w_{jk} + w_{kj}),$$

который является оценкой числа дефектных пар, устраняемых при выполнении k-го неравенства. (как и прежде, предполагается, что $w_{jk} = 0$ для всех $(j,k) \notin \notin \mathbb{D}(B_1, \ldots, B_{p-1})$).

Задача свелась к поиску совместной подсистемы максимального веса в системе неравенств (2.25). Для её решения применим приближённый метод, не гарантирующий максимальности найденной подсистемы. Потребуем, чтобы все значения

$$\frac{|B_p(x_k) - y_k|}{\frac{1}{2}\min(\Delta y_k^+, \Delta y_k^-)}, \qquad k \in D_{p-1}$$

были одновременно малы. С учётом весовых коэффициентов w_k запишем это требование в виде

$$Q_1(B_p) = \sum_{k \in D_{p-1}} \gamma_k (B_p(x_k) - y_k)^2 \to \min_{B_p \in \mathfrak{M}^0}, \tag{2.26}$$

где

$$\gamma_k = \frac{\sum\limits_{j=1}^{q}(w_{jk} + w_{kj})}{\min^2(\Delta y_k^+, \Delta y_k^-)}, \quad k \in D_{p-1}.$$

В итоге путём двух последовательных модификаций мы свели задачу поиска совместной подсистемы максимального веса (2.24) к задаче минимизации функционала среднеквадратичной ошибки (2.26). Первая модификация состояла в усилении ограничений на оператор B_p, что позволило уменьшить число ограничений. Вторая модификация, напротив, привела к ослаблению требований к оператору B_p, но позволила видоизменить функционал качества.

Полученный функционал имеет совершенно прозрачную интерпретацию: настройка производится на объектах, образующих дефект операторов $B_1, \ldots, B_{p-1}$, причём приоритет отдаётся (посредством весовых коэффициентов γ_k) тем из них, которые лежат в основании наибольшего числа дефектных троек.

Сходство функционалов (2.23) и (2.26) позволяет легко построить комбинированную стратегию настройки при заданном $\lambda \in [0, 1]$:

$$\begin{aligned} Q_\lambda(B_p) &= (1-\lambda)\, Q_0(B_p) + \lambda\, Q_1(B_p) = \\ &= (1-\lambda) \sum_{k=1}^{q} (B_p(x_k) - y_k)^2 + \lambda \sum_{k \in D_{p-1}} \gamma_k (B_p(x_k) - y_k)^2, \end{aligned}$$

или, если условиться, что $\gamma_k = 0$ при $k \notin D_{p-1}$,

$$Q_\lambda(B_p) = \sum_{k=1}^{q} (1 - \lambda + \gamma_k \lambda)(B_p(x_k) - y_k)^2. \tag{2.27}$$

При $\lambda = 0$ этот функционал переходит в $Q_0(B_p)$ и оператор B_p настраивается на исходные прецеденты. При $\lambda = 1$ он преобразуется в $Q_1(B_p)$, и настройка идёт на компенсацию неточности остальных операторов. При промежуточных значениях λ получаются комбинированные стратегии настройки.

Для минимизации всех трёх функционалов (2.23), (2.26) и (2.27) можно применять одни и те же методы, например, метод наименьших квадратов.

2.3.5 О методах построения монотонных корректирующих операций

Методы, описанные в предыдущих параграфах, позволяют построить локальный базис, оптимизированный под монотонную корректирующую операцию. Теперь рассмотрим задачу построения собственно монотонного отображения F, доставляющего минимум функционалу качества

$$Q(F) = |\mathbb{D}(F(B_1, \ldots, B_p))|$$

при фиксированных алгоритмических операторах $B_1, \ldots, B_p$.

Вообще говоря, на набор операторов $B_1, \ldots, B_p$ не накладывается никаких ограничений кроме требования допустимости для векторов $\{a_k\}_{k=1}^q$, поэтому

он может иметь дефект. В этом случае последовательность пар $\{a_k, y_k\}_{k=1}^q$ не является монотонной в смысле определения 3, нулевое значение функционала $Q(F)$ недостижимо, и провести монотонную функцию F, точно проходящую через q точек (a_k, y_k), невозможно.

При построении монотонной корректирующей операции некоторые из векторов a_k, образующих дефект, придётся исключить из рассмотрения. Это можно сделать таким образом, чтобы число исключаемых векторов было минимально, а оставшееся множество образовывало монотонную последовательность. Алгоритм последовательного исключения дефектообразующих векторов будет рассмотрен в §2.3.8.

Разумеется, отбрасывание некоторых векторов приведёт к возникновению ошибок на соответствующих им объектах обучения. Величину этих ошибок можно минимизировать, если вернуть исключённые векторы в последовательность $\{a_k, y_k\}_{k=1}^q$, приписав им новые значения финальных информаций. Можно сделать это таким образом, чтобы последовательность осталась монотонной, и в то же время число возникших дефектных пар было минимальным. Соответствующий алгоритм обратного добавления дефектообразующих векторов также будет описан в §2.3.8.

Итак, приступая к построению монотонной корректирующей операции, мы имеем монотонную последовательность пар $\{a_k, y_k\}_{k=1}^q$. Задача состоит в том, чтобы построить монотонную функцию F, проходящую через заданные q точек, то есть удовлетворяющую равенствам $F(a_k) = y_k$ для всех $k \in \mathbb{Q}$.

С каждым из векторов a_k свяжем два множества, которые назовём соответственно верхней и нижней *областью монотонности* вектора a_k:

$$\begin{aligned} M_k^1 &= \{a \in \mathbb{R}^p \mid a_k \leqslant a\}; \\ M_k^0 &= \{a \in \mathbb{R}^p \mid a \leqslant a_k\}. \end{aligned}$$

Возьмём произвольную p-арную функцию $\mu(\rho_1, \ldots, \rho_p)$, удовлетворяющую двум требованиям:

1^0 Функция μ определена на $\mathbb{R}_+^p$ и не убывает на всей области определения.

2^0 $\mu(\rho_1, \ldots, \rho_p) = 0$ тогда и только тогда, когда $\rho_1 = \ldots = \rho_p = 0$.

На практике в качестве функции μ можно брать максимум, сумму или корень из суммы квадратов:

$$\begin{aligned} \mu(\rho_1, \ldots, \rho_p) &= \max(\rho_1, \ldots, \rho_p); \\ \mu(\rho_1, \ldots, \rho_p) &= \sum\nolimits_{i=1}^p \rho_i; \\ \mu(\rho_1, \ldots, \rho_p) &= \sqrt{\sum\nolimits_{i=1}^p \rho_i^2}. \end{aligned} \tag{2.28}$$

Заметим, что функция минимума не подходит, так как не удовлетворяет требованию 2^0.

С помощью функции μ введём функции расстояния от произвольного вектора $a = (a^1, \ldots, a^p)$ из $\mathbb{R}^p$ до верхних и нижних областей монотонности векторов a_k. Заметим, что вводимые функции не являются метриками, хотя и называются расстояниями.

Для произвольных векторов a из $\mathbb{R}^p$ и a_k из $\{a_k\}_{k=1}^q$ определим расстояние от a до верхней области монотонности вектора a_k:

$$r^1(a, a_k) = \mu\left((a_k^1 - a^1)_+, \ldots, (a_k^p - a^p)_+\right),$$

и расстояние от a до нижней области монотонности вектора a_k:

$$r^0(a, a_k) = \mu\left((a^1 - a_k^1)_+, \ldots, (a^p - a_k^p)_+\right),$$

где индекс «+» обозначает операцию срезки: $z_+ = z$ при $z \geqslant 0$, и $z_+ = 0$ при $z \leqslant 0$.

Непосредственно из определений вытекают следующие свойства введённых расстояний:

(а) функция $r^1(a, a_k)$ невозрастающая по первому аргументу;

(б) функция $r^0(a, a_k)$ неубывающая по первому аргументу;

(в) вектор $a \in \mathfrak{I}_e^p$ принадлежит области монотонности тогда и только тогда, когда расстояние до неё равно нулю: $a \in M_k^\beta \Leftrightarrow r^\beta(a, a_k) = 0$, где $\beta = 0, 1$;

(г) если функция μ непрерывна, то функции r^1 и r^0 также непрерывны.

Как и прежде, рассмотрим отдельно два случая: задачу классификации с $\mathfrak{I}_f = \{0, 1\}$ и задачу восстановления регрессии, в которой $\mathfrak{I}_f = \mathbb{R}$. Любопытно, что идея вычисления расстояний до областей монотонности, почти очевидная в первом случае, легко переносится на второй случай, позволяя строить достаточно гладкие монотонные поверхности.

2.3.6 Монотонные корректирующие операции в задаче классификации

Как обычно для случая классификации с двумя непересекающимися классами, положим $\mathfrak{I}_e = \mathbb{R}$, $\mathfrak{I}_f = \{0, 1\}$, и рассмотрим задачу построения монотонной корректирующей операции.

Имеется последовательность пар $\{a_k, y_k\}_{k=1}^q$, $a_k \in \mathbb{R}^p$, $y_k \in \{0, 1\}$, монотонная в смысле определения 3. Требуется построить монотонную функцию $F(a)$, проходящую через заданные q точек: $F(a_k) = y_k$, для всех $k \in \mathbb{Q}$.

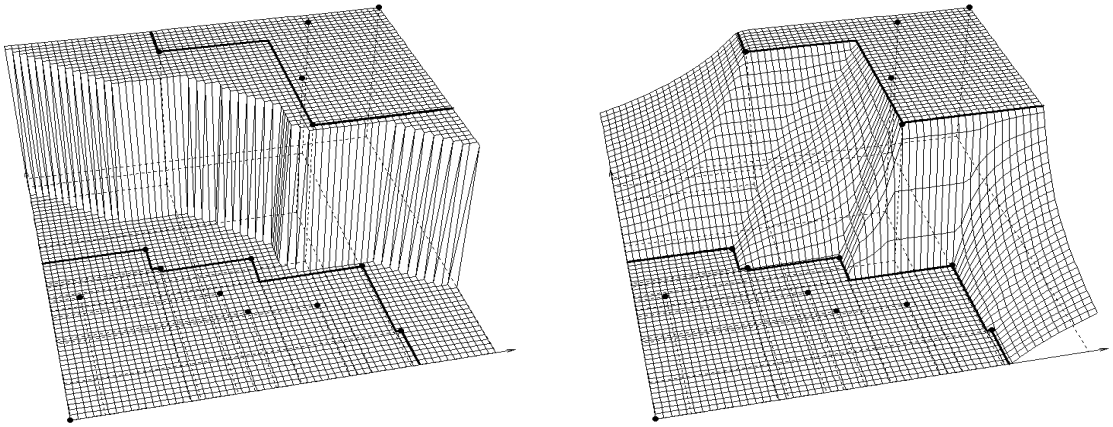
Определим для произвольного a значение $F(a)$ следующим образом. Вычислим расстояния от a до верхних областей монотонности всех векторов a_k, у которых $y_k = 1$, и до нижних областей монотонности всех векторов, у которых $y_k = 0$. Найдём k, для которого соответствующее расстояние минимально, и положим $F(a) = y_k$. В случае неопределённости, когда расстояния до двух областей монотонности, k-ой и j-ой, совпадают, минимальны и $y_j \neq y_k$, положим $F(a) = 0$. Запишем это формально:

$$h^1(a) \;=\; \min_{\{k: y_k = 1\}} r^1(a, a_k);$$

$$h^0(a) = \min_{\{k:y_k=0\}} r^0(a,a_k);$$
$$F(a) = \theta(h^0(a) - h^1(a));$$

где θ — функция Хевисайда. Определение функций h^0 и h^1 корректно, если в обучающей выборке найдутся хотя бы два объекта из разных классов.

Данное определение имеет простой геометрический смысл. Функция $F(a)$ представляет собой монотонную «ступеньку»: она равна 1 на объединении верхних областей монотонности множества векторов $\{a_k \mid y_k = 1\}$, и равна 0 на объединении нижних областей монотонности множества векторов $\{a_k \mid y_k = 0\}$. Пространство между этими областями разбивается такой ступенькой по средней линии, если среднее понимать в смысле выбранной функции расстояния. На рис. 3(а) показана ступенчатая функция для случая, когда $\mu = \max$.



(а) дискретная ступенчатая функция $F(a)$ для задачи классификации.

(б) непрерывная ступенчатая функция $F_Y(a)$ для задачи восстановления регрессии.

Рис. 3: Дискретная и непрерывная элементарные ступенчатые функции, построенные по одной и той же случайной монотонной выборке. Линией отмечены границы объединённых областей монотонности.

Теорема 6. *Если в обучающей выборке имеются хотя бы два объекта, принадлежащих разным классам, то функция $F(a)$ монотонно неубывающая и удовлетворяет условию $F(a_k) = y_k$ для всех $k \in \mathbb{Q}$.*

Доказательство. Покажем, что функция $h^1(a)$ невозрастающая. Возьмём произвольные векторы a и b из $\mathbb{R}^p$, $a \leqslant b$. По определению функции h^1 найдутся такие k и j из $\mathbb{Q}$, что

$$h^1(a) = r^1(a,a_k);$$
$$h^1(b) = r^1(b,a_j).$$

В силу минимальности расстояния $r^1(b,a_j)$ справедливо неравенство $r^1(b,a_j) \leqslant \leqslant r^1(b,a_k)$. Поскольку функция r^1 невозрастающая по первому аргументу, справедливо другое неравенство: $r^1(b,a_k) \leqslant r^1(a,a_k)$. Объединяя оба неравенства, по-

лучаем $r^1(b, a_j) \leqslant r^1(a, a_k)$, или, что то же самое, $h^1(b) \leqslant h^1(a)$. Но это в силу произвольности векторов a и b означает, что функция h^1 невозрастающая.

Аналогично доказывается, что функция $h^0(a)$ неубывающая.

Функция $F(a)$ неубывающая, так как она является суперпозицией неубывающей функции Хевисайда и суммы двух неубывающих функций h^0 и $-h^1$.

Докажем, что функции h^0 и h^1 не могут одновременно обратиться в нуль. Если бы это было так, то для некоторых k и j из $\mathbb{Q}$ имело бы место равенство $r^1(a, a_k) = r^0(a, a_j) = 0$. Отсюда следует, что $0 = y_j < y_k = 1$. С другой стороны, a принадлежит верхней области монотонности вектора a_k и нижней — вектора a_j, поэтому $a_k \leqslant a \leqslant a_j$. Полученные соотношения противоречат монотонности последовательности $\{a_k, y_k\}_{k=1}^q$ в смысле определения 3, значит h^0 и h^1 одновременно не нули.

Рассмотрим произвольный вектор a_k из последовательности $\{a_k\}_{k=1}^q$. Если $y_k = 1$, то $h^1(a_k) = 0$. Поскольку h^0 не может обратиться в нуль одновременно с h^1, аргумент функции Хевисайда положителен, и $F(a_k) = 1$. Если же $y_k = 0$, то $h^0(a_k) = 0$, значение $h^1(a_k)$ положительно, и $F(a_k) = 0$.

Теорема доказана.

2.3.7 Монотонные корректирующие операции в задаче восстановления регрессии

Положим, как обычно для случая восстановления регрессии, что $\mathfrak{I}_e = \mathfrak{I}_f = \mathbb{R}$ и решающие правила не используются.

Рассмотрим задачу построения монотонной корректирующей операции. Имеется последовательность пар $\{a_k, y_k\}_{k=1}^q$, $a_k \in \mathbb{R}^p$, $y_k \in \mathbb{R}$, монотонная в смысле определения 3. Требуется построить монотонную функцию $F(a)$, проходящую через заданные q точек:

$$F(a_k) = y_k, \quad k \in \mathbb{Q}. \tag{2.29}$$

В случае восстановления регрессии на функцию F целесообразно наложить дополнительные ограничения непрерывности и гладкости. Связано это с тем, что указанные ограничения часто используются в качестве универсальных в задачах восстановления регрессии. Под гладкостью обычно понимают дифференцируемость и требуют, чтобы искомая функция доставляла минимум функционалу

$$G(F) = \int_\Omega (DF(a))^2 \, da, \tag{2.30}$$

где Ω — область определения функции $F(a)$, D — некоторый дифференциальный оператор, называемый энергетическим. Во многих случаях в качестве D берут оператор Лапласа.

Вариационные задачи минимизации функционала (2.30) при ограничениях вида (2.29) рассматриваются в теории сплайнов [4, 23]. Для её применения необходимо, чтобы множество функций, из которого выбирается F, было гильбертовым

пространством. К сожалению, семейство монотонных функций таковым не является. Вопрос о построении монотонной наиболее гладкой функции, проходящей через заданные точки, является открытым.

Нашей задачей будет построение непрерывной *достаточно* гладкой функции. Причём гладкость в данном случае удобнее понимать не в смысле дифференцируемости, а в смысле отсутствия у функции резких скачков в промежутках между точками a_k. Для оценки гладкости в таком, менее строгом, понимании достаточно использовать конечно-разностные аналоги функционала (2.30).

Ниже будут описаны три способа построения монотонных корректирующих операций. Гладкость получаемых функций будет оцениваться по результатам численных экспериментов на случайных монотонных выборках.

Напомним, что с каждым из q векторов a_k связаны верхняя область монотонности M_k^1, нижняя область монотонности M_k^0 и функции расстояния до этих областей r^1, r^0. Потребуем дополнительно, чтобы p-арная функция μ была непрерывной. Заметим, что все три функции (2.28) удовлетворяют этому требованию.

Прежде расстояния $r^1(a, a_k)$ и $r^0(a, a_k)$ позволили построить дискретную монотонную функцию, принимающую только два значения. Теперь применим их для построения непрерывной монотонной функции.

Перенумеруем объекты обучающей выборки по возрастанию значений y_k, так чтобы выполнялась цепочка неравенств

$$y_1 \leqslant y_2 \leqslant \ldots \leqslant y_{q-1} \leqslant y_q.$$

Для произвольного $Y \in [y_1, y_q)$ определим функции

$$\begin{aligned} h_Y^1(a) &= \min_{\{k:y_k>Y\}} r^1(a, a_k); \\ h_Y^0(a) &= \min_{\{k:y_k\leqslant Y\}} r^0(a, a_k); \\ f_Y(a) &= \frac{h_Y^0(a)}{h_Y^0(a) + h_Y^1(a)}. \end{aligned}$$

Определения функций h_Y^1 и h_Y^0 корректны, так как множества индексов, по которым берутся минимумы, непустые в силу условия $y_1 \leqslant Y < y_q$.

Функция $f_Y(a)$ является непрерывным аналогом монотонной ступеньки, построенной в предыдущем параграфе, см. рис. 3(б). Она равна единице на объединении верхних областей монотонности множества векторов $\{a_k \mid y_k > Y\}$ и нулю — на объединении нижних областей монотонности множества векторов $\{a_k \mid y_k \leqslant Y\}$. В пространстве между этими областями она принимает всевозможные промежуточные значения.

Лемма 4. *Если $y_1 \leqslant Y < y_q$, то функция $f_Y(a)$ непрерывная, неубывающая, принимает значения из отрезка $[0, 1]$, причём на векторах a_k — только 0 или 1:*

$$f_Y(a_k) = \theta(y_k - Y), \quad k \in \mathbb{Q}.$$

Доказательство. Легко показать, что функция $h_Y^1(a)$ является невозрастающей, а функция $h_Y^0(a)$ — неубывающей. Рассуждения аналогичны первой части

доказательства теоремы 6. Отсюда вытекает, что функция $f_Y(a)$ является неубывающей.

Докажем, что функции h_Y^0 и h_Y^1 не могут одновременно обратиться в нуль. Допустим, что это не так. Тогда в силу условия $y_1 \leqslant Y < y_q$ существуют k и j из $\mathbb{Q}$, для которых имеет место равенство $r^1(a, a_k) = r^0(a, a_j) = 0$. Отсюда следует, что $y_j \leqslant Y < y_k$. С другой стороны, поскольку расстояния до областей монотонности равны нулю, $a_k \leqslant a \leqslant a_j$. Полученные соотношения противоречат монотонности последовательности $\{a_k, y_k\}_{k=1}^q$ в смысле определения 3. Таким образом, знаменатель в определении функции $f_Y(a)$ никогда не обращается в нуль.

Все функции $r^1(a, a_k)$, $r^0(a, a_k)$, $k = 1, \ldots, q$, непрерывны по первому аргументу. Функции $h_Y^1(a)$ и $h_Y^0(a)$ непрерывны, так как минимум непрерывных функций есть непрерывная функция. Следовательно функция $f_Y(a)$ непрерывна как суперпозиция непрерывных функций.

Для произвольного вектора $a \in \mathbb{R}^p$ в силу положительности $h_Y^1(a)$ и $h_Y^0(a)$ справедлива цепочка неравенств $0 \leqslant h_Y^0(a) \leqslant h_Y^0(a) + h_Y^1(a)$, откуда следует $0 \leqslant$ $\leqslant f_Y(a) \leqslant 1$.

Рассмотрим произвольный вектор a_k из последовательности $\{a_k\}_{k=1}^q$. Если $y_k > Y$, то $h_Y^1(a_k) = \min\limits_{k:y_k>Y} r^1(a_k, a_k) = 0$, следовательно $f_Y(a_k) = 1$. Если же $y_k \leqslant$ $\leqslant Y$, то $h_Y^0(a_k) = 0$, значит $f_Y(a_k) = 0$. Объединяя оба случая, заключаем, что в точках $\{a_k\}_{k=1}^q$ функция $f_Y(a)$ вычисляется по формуле $f_Y(a_k) = \theta(y_k - Y)$.

Лемма доказана.

Определим монотонную корректирующую операцию F как сумму непрерывных монотонных ступенек:

$$F(a) = y_1 + \sum_{k=1}^{q-1} (y_{k+1} - y_k)\, f_{y_k}(a). \tag{2.31}$$

Теорема 7. *Функция $F(a)$ определена на всём множестве $\mathbb{R}^p$, непрерывна, монотонно не убывает и удовлетворяет условию $F(a_k) = y_k$ для всех $k \in \mathbb{Q}$.*

Доказательство. Функция $F(a)$ непрерывная и неубывающая как сумма q непрерывных неубывающих функций. Согласно лемме справедливо тождество $f_{y_k}(a_j) = \theta(y_j - y_k)$ для всех j и k из $\mathbb{Q}$. Поэтому для любого $j \in \mathbb{Q}$

$$F(a_j) = y_1 + \sum_{k=1}^{q-1} (y_{k+1} - y_k)\, \theta(y_j - y_k) = y_1 + \sum_{k=1}^{j-1} (y_{k+1} - y_k) = y_j.$$

Теорема доказана.

Замечательная особенность описанной корректирующей операции состоит в том, что она является существенно более гладкой, чем другие конструкции, применявшиеся до сих пор в целях монотонной коррекции. Опишем вкратце две из них и приведём результаты их сравнительного анализа.

Первая конструкция представляет собой суперпозицию двух функций — суммы элементарных ступенчатых функций F_0 и одномерного монотонного сплай-

на M:

$$F_0(a) = \sum_{s=1}^{q'} \sigma(a, a_{k_s}), \qquad F_1(a) = M(F_0(a)).$$

Число ступенчатых функций q' не превышает q. Каждая ступенька $\sigma(a, a_{k_s})$ строится как непрерывный аналог разрывной ступеньки, равной единице, если $a \geqslant a_{k_s}$ и нулю в остальных точках. Непрерывность ступеньки обеспечивается путём её расширения по каждой i-ой координате, $i = 1, \ldots, p$, на величину минимума $(a_k^i - a_j^i)$ по всем j, для которых $a_k^i > a_j^i$. Элементарные ступеньки привязываются не ко всем векторам $\{a_k\}_{k=1}^q$, а только к некоторым из них, образующих специально выбранную подпоследовательность $\{a_{k_s}\}_{s=1}^{q'}$. Выбор подпоследовательности, а также высоты каждой элементарной ступеньки, производится таким образом, чтобы функция F_0 воспроизводила тот же порядок на векторах $\{a_k\}_{k=1}^q$, что и значения $\{y_k\}_{k=1}^q$. Затем строится одномерный монотонный сплайн M, обеспечивающий выполнение соотношений $M(F_0(a_k)) = y_k$. Построенная таким образом функция F_1 является монотонной и непрерывной.

Вторая конструкция основана на построении p неравномерных сеток с узлами $a_1^i, \ldots, a_q^i$ по каждой из координат, $i = 1, \ldots, p$. Эти сетки индуцируют p-мерную прямоугольную сетку на пространстве $\mathbb{R}^p$, состоящую из q^p узлов и разбивающую всё пространство на $(q+1)^p$ ячеек. Положение каждого узла этой сетки однозначно определяется набором p индексов $\vec{\imath} = (i_1, \ldots, i_p)$, так что $\vec{\imath}$-ый узел есть вектор $a_{\vec{\imath}} = (a_{i_1}^1, \ldots, a_{i_p}^p) \in \mathbb{R}^p$. Очевидно, векторы $\{a_k\}_{k=1}^q$ совпадают с некоторыми из узлов сетки. Во всех узлах задаются значения $F(a_{\vec{\imath}})$ исходя из двух требований: во-первых, чтобы соблюдались соотношения $F(a_k) = y_k$ для всех $k = 1, \ldots, q$, во-вторых, чтобы функция F была монотонной на узлах сетки. Для вычисления $F(a)$ при произвольном a сначала находят ячейку p-мерной сетки, в которую попадает вектор a. Затем вычисляют $F(a)$ как линейную комбинацию значений $F(a_{\vec{\imath}})$, взятых из 2^p узлов, являющихся вершинами данной ячейки. При этом коэффициенты линейной комбинации задаются таким образом, чтобы в узлах сетки функция F принимала заданные значения $F(a_{\vec{\imath}})$, а её «куски», соответствующие соседним ячейкам, «склеивались» на границе ячеек, образуя непрерывную функцию.

Для сравнения трёх методов построения монотонной корректирующей операции был проведён вычислительный эксперимент. В пространстве размерности $p = 2$ строились случайные монотонные выборки различной длины, и корректирующие операции сравнивались по критерию гладкости. Кроме трёх монотонных функций строился немонотонный интерполяционный сплайн, оптимальный в смысле функционала $G(F)$, который также оценивался на гладкость. Для его построения использовалось аналитическое представление интерполяционного сплайна в области с хаотически расположенными узлами [4].

Гладкость оценивалась по конечно-разностному аналогу функционала $G(F)$:

$$G_\Delta^2(F) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(\frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\Delta_1^2} \right)^2 + \left(\frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{\Delta_2^2} \right)^2,$$

где $f_{i,j}$ — значения функции $F(a)$ в узлах равномерной прямоугольной сетки размером $n \times m$ точек, шагом Δ_1 по первой координате и Δ_2 по второй. В данном эксперименте $n = m = 20$.

Результаты численного эксперимента (см. рис. 4) свидетельствуют о том, что предложенный метод позволяет строить монотонные корректирующие операции достаточной степени гладкости. По значению функционала гладкости они оказываются значительно ближе к классическим сплайнам, чем к монотонным корректирующим операциям, использовавшимся до сих пор.

Рассмотрим теперь асимптотические свойства монотонной корректирующей операции (2.31).

Теорема 8. *Функция $F(a)$ ограничена, $y_1 \leqslant F(a) \leqslant y_q$ для любого $a \in \mathbb{R}^p$, и постоянна на множествах M_1^0 и M_q^1:*

$$F(a) = \begin{cases} y_1, & a \in M_1^0; \\ y_q, & a \in M_q^1. \end{cases} \tag{2.32}$$

Доказательство. Ограниченность функции $F(a)$ вытекает из ограниченности функций $f_Y(a)$, см. лемму 4, и формулы (2.31).

Для доказательства (2.32) возьмём произвольное Y, $Y \geqslant y_1$, и произвольный вектор $a \in M_1^0$. В силу условия $Y \geqslant y_1$ индекс 1 принадлежит множеству $\{k \mid y_k \leqslant \leqslant Y\}$, а из условия $a \in M_1^0$ вытекает $a \leqslant a_1$. Следовательно

$$\begin{aligned} h_Y^0(a) = \min_{\{k:y_k \leqslant Y\}} \mu\Big((a^1 - a_k^1)_+, \ldots, (a^p - a_k^p)_+\Big) = \\ = \mu\Big((a^1 - a_1^1)_+, \ldots, (a^p - a_1^p)_+\Big) = 0. \end{aligned}$$

Таким образом, $h_{y_1}^0(a) = \ldots = h_{y_q}^0(a) = 0$, а значит $f_{y_1}(a) = \ldots = f_{y_q}(a) = 0$. По формуле (2.31) получаем, что $F(a) = y_1$ для всех $a \in M_1^0$.

Аналогично доказывается, что если $a \in M_q^1$, то $f_{y_1}(a) = \ldots = f_{y_q}(a) = 1$. По формуле (2.31) получаем, что $F(a) = y_q$ для всех $a \in M_q^1$.

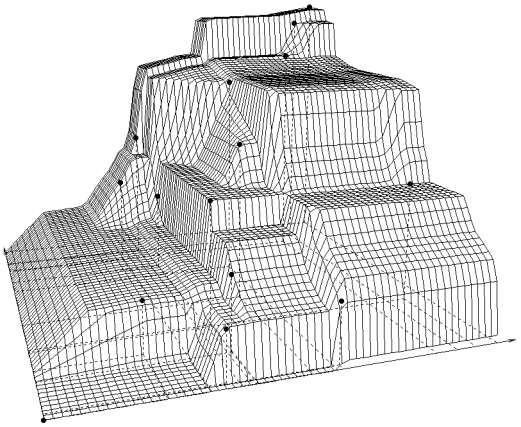
Теорема доказана.

В силу непрерывности, монотонности и ограниченности функция $F(a)$ имеет конечный предел по любой из координат:

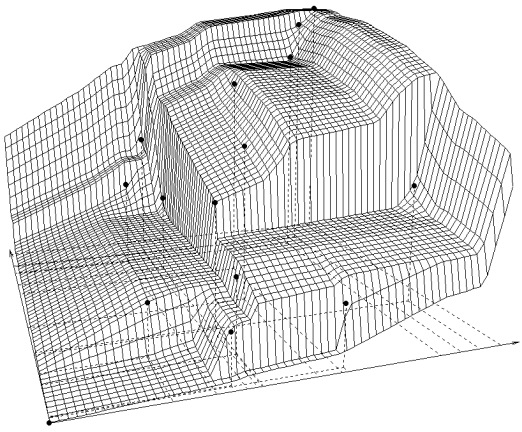
$$\lim_{a^i \to \infty} F(a^1, \ldots, a^p) \in [y_1, y_q], \qquad i = 1, \ldots, p.$$

Такое асимптотическое поведение не всегда приемлемо для монотонных корректирующих операций. Дело в том, что при применении итерационного процесса (1.1)–(1.4) только оператор B_1 настраивается на исходные прецеденты. Остальные операторы $B_2, \ldots, B_p$ используются для компенсации ошибок, допущенных B_1. Поэтому от корректирующей операции F разумно потребовать асимптотического поведения $F(a^1, \ldots, a^p) \sim a^1$ при неограниченном удалении вектора $a = (a^1, \ldots, a^p)$ от векторов $\{a_k\}_{k=1}^q$.

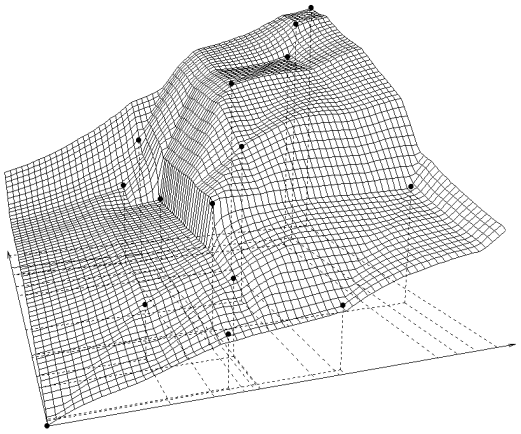
Рассмотрим более общий случай, когда задана непрерывная монотонная на $\mathbb{R}^p$ функция $\varphi(a)$ и требуется построить корректирующую операцию $\widetilde{F}(a)$



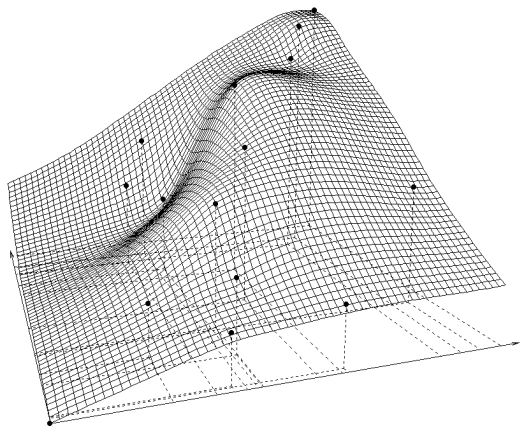
(а) интерполяция суммой элементарных ступенчатых функций, $G_\Delta(F) = 64.93$.



(б) интерполяция по ячейкам прямоугольной сетки, $G_\Delta(F) = 69.42$.



(в) интерполяция по расстояниям до областей монотонности, $G_\Delta(F) = 23.03$.



(г) интерполяция гладким немонотонным сплайном, $G_\Delta(F) = 9.95$.

Рис. 4: Монотонные корректирующие операции, построенные различными методами на одной и той же выборке длины $q = 15$ в пространстве размерности 2. Приводятся значения функционала гладкости $G_\Delta(F)$.

из $\mathfrak{F}_M$, удовлетворяющую асимптотическому требованию $\widetilde{F}(a) \sim \varphi(a)$ при неограниченном удалении a от $\{a_k\}_{k=1}^q$. За основу возьмём уже имеющуюся функцию $F(a)$.

Зададим произвольное положительное число C и положим

$$H(a) = \Big(y_1 - \varphi(a)\Big)_+ \min_{k\in\mathbb{Q}} r^1(a, a_k) + \Big(\varphi(a) - y_q\Big)_+ \min_{k\in\mathbb{Q}} r^0(a, a_k);$$

$$\widetilde{F}(a) = \frac{CF(a) + H(a)\varphi(a)}{C + H(a)}. \tag{2.33}$$

Теорема 9. *Функция $\widetilde{F}(a)$ непрерывная неубывающая на $\mathbb{R}^p$ и удовлетворяет условию $\widetilde{F}(a_k) = y_k$ для всех $k \in \mathbb{Q}$. Если функция μ неограничена на $\mathbb{R}^p_+$ и существует предел $\lim\limits_{a^i\to\infty} \varphi(a)$, $1 \leqslant i \leqslant p$, то*

$$\lim_{a^i\to\infty} \Big(\widetilde{F}(a) - \varphi(a)\Big) = 0 \tag{2.34}$$

Доказательство.

1. Введём следующие обозначения:

$$\begin{aligned}
R^1(a) &= \min_{k\in\mathbb{Q}} r^1(a, a_k);\\
R^0(a) &= \min_{k\in\mathbb{Q}} r^0(a, a_k);\\
H^1(a) &= (y_1 - \varphi(a))_+ R^1(a); &(2.35)\\
H^0(a) &= (\varphi(a) - y_q)_+ R^0(a). &(2.36)
\end{aligned}$$

В этих обозначениях $H(a) = H^1(a) + H^0(a)$. Согласно формулам (2.35)–(2.36) на любом векторе a либо $H^1(a) = 0$, либо $H^0(a) = 0$. Поэтому пространство $\mathbb{R}^p$ разбивается на три непересекающихся множества:

$$\begin{aligned}
\Omega^1 &= \{a \mid H^1(a) > 0\};\\
\Omega^0 &= \{a \mid H^0(a) > 0\};\\
\Omega &= \{a \mid H^1(a) = H^0(a) = 0\}.
\end{aligned}$$

Поскольку функция $\widetilde{F}(a)$ в каждой точке определена как взвешенная сумма функций $F(a)$ и $\varphi(a)$ с весовыми коэффициентами из отрезка $[0, 1]$, то

$$\varphi(a) < \widetilde{F}(a) < F(a), \qquad a \in \Omega^1; \tag{2.37}$$

$$F(a) < \widetilde{F}(a) < \varphi(a), \qquad a \in \Omega^0; \tag{2.38}$$

$$\widetilde{F}(a) = F(a), \qquad a \in \Omega. \tag{2.39}$$

2. Функция $\widetilde{F}(a)$ непрерывна на $\mathbb{R}^p$ как суперпозиция непрерывных функций.

3. Для любого вектора a_k, $k \in \mathbb{Q}$ имеем $R^1(a_k) = R^0(a_k) = 0$, следовательно $H(a_k) = 0$ и $\widetilde{F}(a_k) = F(a_k) = y_k$ для всех $k \in \mathbb{Q}$.

4. Покажем, что функция $\widetilde{F}(a)$ монотонная на $\mathbb{R}^p$. Возьмём произвольные векторы a и b из $\mathbb{R}^p$, $a \leqslant b$. Наибольший интерес представляют случаи, когда оба

вектора лежат либо в Ω^1, либо в Ω^0. Во всех остальных случаях их взаимного расположения из соотношений (2.37)–(2.39) и монотонности $F(a)$ легко получаем $\widetilde{F}(a) \leqslant \widetilde{F}(b)$.

Пусть $a, b \in \Omega^1$. Тогда $H(a) \equiv H^1(a)$ — невозрастающая функция. Используем этот факт, а также соотношение (2.37) при выводе цепочки неравенств:

$$\begin{aligned}
\widetilde{F}(b) - \widetilde{F}(a) &= \varphi(b) - \varphi(a) + C\frac{F(b) - \varphi(b)}{C + H(b)} - C\frac{F(a) - \varphi(a)}{C + H(a)} \geqslant \\
&\geqslant \varphi(b) - \varphi(a) + \frac{C}{C + H(a)}\Big(F(b) - F(a) - \varphi(b) + \varphi(a)\Big) \geqslant \\
&\geqslant \Big(\varphi(b) - \varphi(a)\Big)\left(1 - \frac{C}{C + H(a)}\right) \geqslant 0.
\end{aligned}$$

Пусть теперь $a, b \in \Omega^0$. Тогда $H(a) \equiv H^0(a)$ — неубывающая функция. Используем этот факт, а также соотношение (2.38) при выводе цепочки неравенств:

$$\begin{aligned}
\widetilde{F}(b) - \widetilde{F}(a) &= F(b) - F(a) + H(b)\frac{\varphi(b) - F(b)}{C + H(b)} - H(a)\frac{\varphi(a) - F(a)}{C + H(a)} \geqslant \\
&\geqslant F(b) - F(a) + \frac{H(a)}{C + H(a)}\Big(\varphi(b) - \varphi(a) - F(b) + F(a)\Big) \geqslant \\
&\geqslant \Big(F(b) - F(a)\Big)\left(1 - \frac{H(a)}{C + H(a)}\right) \geqslant 0.
\end{aligned}$$

Итак, для любых a и b из $a \leqslant b$ следует $\widetilde{F}(a) \leqslant \widetilde{F}(b)$. Монотонность функции $\widetilde{F}(a)$ доказана.

5. Теперь докажем, что функция $\widetilde{F}(a)$ удовлетворяет асимптотическому условию (2.34). Прежде всего заметим, что в силу неограниченности μ функции $R^1(a)$ и $R^0(a)$ также неограничены:

$$\lim_{a^i \to -\infty} R^1(a) = \infty, \qquad \lim_{a^i \to +\infty} R^0(a) = \infty,$$

откуда следует $\lim\limits_{a^i \to \infty} H(a) = \infty$. Используем этот факт, а также ограниченность функции $F(a)$:

$$\begin{aligned}
L &\equiv \lim_{a^i \to \infty} (\widetilde{F}(a) - \varphi(a)) = \lim_{a^i \to \infty} \frac{CF(a) - C\varphi(a)}{C + H(a)} = \\
&= \frac{C \lim\limits_{a^i \to \infty} \frac{F(a)}{H(a)} - C \lim\limits_{a^i \to \infty} \frac{\varphi(a)}{H(a)}}{C \lim\limits_{a^i \to \infty} \frac{1}{H(a)} + 1} = -C \lim_{a^i \to \infty} \frac{\varphi(a)}{H(a)}.
\end{aligned}$$

Если предел $\lim\limits_{a^i \to \infty} \varphi(a)$ конечен, то $L = 0$, что и требовалось доказать. Если он бесконечен, то найдётся такое число A, что для всех a^i, $|a^i| > A$ либо $\varphi(a) < y_1$, либо $\varphi(a) > y_q$.

В первом случае $H(a) = H^1(a)$,

$$L = -C \lim_{a^i \to \infty} \frac{\varphi(a)}{y_1 - \varphi(a)} \frac{1}{R^1(a)} = C \lim_{a^i \to \infty} \frac{1}{R^1(a)} = 0.$$

Во втором случае $H(a) = H^0(a)$,

$$L = -C \lim_{a^i \to \infty} \frac{\varphi(a)}{\varphi(a) - y_q} \frac{1}{R^0(a)} = -C \lim_{a^i \to \infty} \frac{1}{R^0(a)} = 0.$$

Теорема доказана.

Итак, формула (2.33) позволяет построить монотонную корректирующую операцию $\widetilde{F}(a)$ с заданным асимптотическим поведением на основе имеющейся монотонной корректирующей операции $F(a)$, ограниченной и принимающей значения из отрезка $[y_1, y_q]$. При этом обе функции удовлетворяют исходному требованию $F(a_k) = \widetilde{F}(a_k) = y_k$ для всех $k \in \mathbb{Q}$.

Константа C играет роль параметра сглаживания: чем больше C, тем медленней функция $\widetilde{F}(a)$ приближается к $\varphi(a)$ при стремлении a в бесконечность. На практике параметр C предполагается подбирать экспериментально. В качестве разумного начального приближения можно взять, скажем,

$$C = (y_q - y_1) \max_{k,j \in \mathbb{Q}} \|a_k - a_j\|,$$

где $\|\cdot\|$ — какая-либо норма в $\mathbb{R}^p$. Эта формула основана на эвристическом предположении, что параметр C должен быть того же порядка, что и величина $H(a)$ на некотором расстоянии от выпуклой оболочки векторов $\{a_k\}_{k=1}^q$, сравнимом с диаметром этой оболочки.

2.3.8 Некоторые алгоритмы монотонизации выборок

В данном параграфе приводятся два вспомогательных алгоритма, предназначенных для преобразования немонотонных последовательностей в монотонные (в смысле определения 3). Эти алгоритмы следует применять в тех случаях, когда набор операторов $B_1, \ldots, B_p$ имеет дефект, вследствие чего индуцируемая им последовательность $\{a_k, y_k\}_{k=1}^q$ не является монотонной.

Первый алгоритм предназначен для исключения дефектообразующих векторов из последовательности $\{a_k, y_k\}_{k=1}^q$. Он основан на поочерёдном исключении «наиболее дефектных» векторов и гарантирует, что оставшиеся векторы образуют монотонную последовательность.

Второй алгоритм позволяет вернуть исключённые векторы обратно, скорректировав соответствующие значения y_k таким образом, чтобы число образовавшихся от этого дефектных пар было минимальным. Он также гарантирует монотонность полученной в результате последовательности.

Алгоритм последовательного исключения дефектообразующих векторов. Имеется последовательность (a_k, y_k), $k \in \mathbb{Q}$. Требуется исключить из неё по возможности наименьшее число элементов так, чтобы не осталось дефектных пар. Напомним, что пара индексов $(j, k) \in \mathbb{Q}^2$ называется дефектной, если $y_j < y_k$ и $a_k \leqslant a_j$.

Будем по очереди исключать из множества $\mathbb{Q}$ индексы, входящие в максимальное количество дефектных пар. Пусть r пробегает значения $q, (q-1), (q-2)$, ... Положим $\mathbb{Q}_q = \mathbb{Q}$. Для произвольного $k \in \mathbb{Q}$ множество образуемых им дефектных пар есть

$$\mathbb{D}_r(k) = \{(j,k),(k,j) \mid j \in \mathbb{Q}_r\} \cap \mathbb{D}(B_1,\ldots,B_p).$$

Выберем индекс k_r, для которого число дефектных пар максимально:

$$k_r = \arg\max_{k\in\mathbb{Q}_r} |\mathbb{D}_r(k)|,$$

и исключим его из множества $\mathbb{Q}_r$:

$$\mathbb{Q}_{r-1} = \mathbb{Q}_r \setminus \{k_r\}.$$

Если при некотором $\widetilde{q}$ окажется $|\mathbb{D}_{\tilde{q}}(k)| = \varnothing$ для всех $k \in \mathbb{Q}_r$, это будет означать, что на множестве $\mathbb{Q}_{\tilde{q}}$ не осталось дефектных пар. Значит последовательность $\{(a_k, y_k)\}_{k\in\mathbb{Q}_{\tilde{q}}}$ монотонна.

Алгоритм обратного добавления дефектообразующих векторов. Пусть в результате применения предыдущего алгоритма получено подмножество индексов $\mathbb{Q}_{\tilde{q}}$, состоящее из $\widetilde{q}$ элементов. Поскольку на этом подмножестве нет дефектных пар и троек, отношение $\prec$ является предпорядком на $\mathbb{Q}_{\tilde{q}}$. Упорядочив множество $\mathbb{Q}_{\tilde{q}}$ по этому отношению, получим последовательность индексов $i_1 \prec i_2 \prec \prec \ldots \prec i_{\tilde{q}}$.

Обратное добавление дефектообразующих векторов состоит из двух шагов. Сначала все ранее исключённые индексы (элементы множества $\mathbb{Q}\setminus\mathbb{Q}_{\tilde{q}}$) по очереди вставляются в эту последовательность, причём оптимальное место вставки каждого индекса определяется из условия минимума числа дефектных пар. Затем значения в последовательности $\{y_{i_k}\}_{k=1}^{q}$, соответствующие добавленным индексам, корректируется так, чтобы она оказалась неубывающей.

Определение 6. Пусть $\mathbb{I}_r = \{i_1, i_2, \ldots, i_r\}$ — последовательность попарно различных элементов множества $\mathbb{Q}$. Пару $(i_s, i_t) \in \mathbb{I}_r^2$, $t < s$, условимся называть *нарушением порядка*, если $y_{i_s} < y_{i_t}$; и *сильным нарушением порядка*, если $a_{i_s} \leqslant a_{i_t}$. Множество нарушений порядка в последовательности $\mathbb{I}_r$ обозначим через $\mathbb{D}(\mathbb{I}_r)$.

Нарушения порядка в последовательности $\mathbb{I}_r$ приводят к невозможности удовлетворить интерполяционному условию $F(a_k) = y_k$ для некоторых $k \in \mathbb{Q}$, что в данном случае является допустимым. В то же время наличие сильных нарушений порядка снова приводит к немонотонности последовательности пар $\{a_k, y_k\}_{k=1}^{q}$ и, как следствие, невозможности построения монотонной корректирующей операции. Поэтому задача заключается в том, чтобы добавить индексы $k_{\tilde{q}+1}, \ldots, k_q$ в последовательность $\mathbb{I}_{\tilde{q}}$ так, чтобы число нарушений порядка было минимально, а сильных нарушений порядка не было вообще.

Будем добавлять эти индексы поочерёдно. Пусть r пробегает значения от $\widetilde{q}+$ $+1$ до q, и при каждом r индекс k_r вставляется в последовательность $\mathbb{I}_{r-1}$ перед

элементом с порядковым номером t_r, $1 \leqslant t_r \leqslant r$. Если $t_r = r$, то элемент k_r добавляется в конец последовательности. В результате получается последовательность $\mathbb{I}_r$.

Определим штрафные функции $\xi_r(t)$, $\eta_r(t)$ и $\varphi_r(t)$, положив:

$$\begin{aligned}
\xi_r(t) &= \sum_{s=1}^{t-1}(\chi(y_{k_r} < y_{i_s}) + M\chi(a_{k_r} \leqslant a_{i_s})),\\
\eta_r(t) &= \sum_{s=t}^{r-1}(\chi(y_{i_s} < y_{k_r}) + M\chi(a_{i_s} \leqslant a_{k_r})),\\
\varphi_r(t) &= \xi_r(t) + \eta_r(t),
\end{aligned}$$

где $t = 1, \ldots, r$; сумма нулевого числа слагаемых считается равной нулю; M — заданное неотрицательное число; $\chi(P)$ — характеристическая функция предиката P, равная 1, если предикат P истинен, и 0 если ложен. Положим t_r равным тому числу, которое доставляет минимум функции $\varphi_r(t)$.

Введённые функции имеют следующий смысл. Допустим, элемент k_r вставляется в последовательность $\mathbb{I}_{r-1}$ перед i_t. Тогда функции $\xi_r(t)$ и $\eta_r(t)$ определяют штраф за все нарушения порядка, образуемые добавляемым элементом в паре с элементами $i_1, \ldots, i_{t-1}$ и $i_t, \ldots, i_{r-1}$ соответственно. Величина штрафа за каждое нарушение порядка равна 1, за каждое сильное нарушение — M. Функция $\varphi_r(t)$ определяет суммарный штраф за нарушения порядка в последовательности $\mathbb{I}_r$, возникающие в результате добавления элемента k_r.

Напомним, что для набора векторов $\{a_k\}_{k=1}^q$ предполагается выполненным условие допустимости (см. стр. 23).

Теорема 10. *Пусть $M > q$. Тогда число нарушений порядка в последовательности $\mathbb{I}_q$ вычисляется по формуле*

$$|\mathbb{D}(\mathbb{I}_q)| = \sum_{r=\tilde{q}+1}^{q} \varphi_r(t_r),$$

причём сильных нарушений порядка в этой последовательности нет.

Доказательство проведём индукцией по r.

Последовательность $\mathbb{I}_{\tilde{q}}$ не содержит сильных нарушений порядка. В противном случае для некоторой пары $(i_s, i_t) \in \mathbb{I}_{\tilde{q}}^2$ выполнялось бы $i_t \prec i_s$ и $a_{i_s} \leqslant a_{i_t}$, следовательно $a_{i_s} = a_{i_t}$, что противоречит допустимости набора векторов $\{a_k\}_{k=1}^q$.

Последовательность $\mathbb{I}_{\tilde{q}}$ не содержит нарушений порядка. В противном случае для некоторой пары $(i_s, i_t) \in \mathbb{I}_{\tilde{q}}^2$ выполнялось бы $i_t \prec i_s$ и $y_{i_s} < y_{i_t}$, следовательно $a_{i_t} \leqslant a_{i_s}$, что невозможно в силу монотонности последовательности $\{(a_k, y_k)\}_{k \in \mathbb{Q}_{\tilde{q}}}$. Таким образом, $\mathbb{D}(\mathbb{I}_{\tilde{q}}) = \varnothing$.

Пусть для последовательности $\mathbb{I}_{r-1}$ утверждение теоремы верно.

Найдём в $\mathbb{I}_{r-1}$ элемент i_u с наибольшим порядковым номером u, для которого $a_{i_u} \leqslant a_{k_r}$. Положим $u = 0$, если таких элементов вообще нет. Аналогично, найдём элемент i_v с наименьшим порядковым номером v, для которого $a_{k_r} \leqslant a_{i_v}$, либо положим $v = r$, если таких элементов нет. По предположению индукции в $\mathbb{I}_{r-1}$ нет

сильных нарушений порядка, поэтому из $a_{i_u} \leqslant a_{k_r} \leqslant a_{i_v}$ вытекает $u \leqslant v$. Если допустить, что $u = v$, то получим равенство $a_{i_u} = a_{k_r}$, которое (с учётом $k_r \neq i_u$) приводит к противоречию с допустимостью набора векторов $\{a_k\}_{k=1}^q$. Следовательно $u < v$.

Оценим функцию $\varphi_r(t)$ в следующих трёх случаях:

а) при $t \leqslant u$ справедливы оценки $\varphi_r(t) \geqslant \eta_r(t) \geqslant \eta_r(u) \geqslant M > q$;

б) при $t > v$ справедливы оценки $\varphi_r(t) \geqslant \xi_r(t) \geqslant \xi_r(v) \geqslant M > q$;

в) при $u < t \leqslant v$ найдём оценку сверху. Поскольку $u < t$, отношение $a_{i_s} \leqslant a_{k_r}$ не выполняется для всех s: $t \leqslant s < r$. Следовательно $\eta_r(t) \leqslant r - t$. Аналогично, в силу $t \leqslant v$ отношение $a_{k_r} \leqslant a_{i_s}$ не выполняется для всех s таких, что $1 \leqslant s < t$. Следовательно $\xi_r(t) < t$. Суммируя, получаем, что $\varphi_r(t) < r \leqslant q$.

Таким образом, число t_r, доставляющее минимум функции $\varphi_r(t)$, удовлетворяет условию $u < t_r \leqslant v$. Отсюда вытекают два вывода. Во-первых, индекс k_r не участвует ни в одном сильном нарушении порядка в последовательности $\mathbb{I}_r$. С учётом предположения индукции получаем, что в $\mathbb{I}_r$ нет сильных нарушений порядка. Во-вторых, значение $\varphi_r(t_r)$ в точности равно числу нарушений порядка, в которых участвует элемент k_r. Следовательно $|\mathbb{D}(\mathbb{I}_r)| = |\mathbb{D}(\mathbb{I}_{r-1})| + \varphi_r(t_r)$.

Теорема доказана.

С помощью полученной последовательности $\mathbb{I}_q = \{i_k\}_{k=1}^q$ скорректируем значения y_k. Положим $y'_k = y_k$ для всех $k \in \mathbb{Q}_{\tilde{q}}$, а для остальных индексов $k \in \mathbb{Q} \setminus \mathbb{Q}_{\tilde{q}}$ зададим такие значения y'_k, чтобы выполнялась цепочка равенств

$$y'_{i_1} \leqslant y'_{i_2} \leqslant \ldots \leqslant y'_{i_q}. \tag{2.40}$$

Последовательность пар $\{a_k, y'_k\}_{k=1}^q$ монотонна, так как по доказанной теореме в $\mathbb{I}_q$ нет сильных нарушений порядка. Следовательно существует монотонная корректирующая операция F, удовлетворяющая системе равенств

$$F(a_k) = y'_k, \quad k \in \mathbb{Q}. \tag{2.41}$$

Для её построения можно воспользоваться одним из методов, описанных в предыдущих параграфах.

Теорема 11. *Значение функционала качества для построенной таким способом монотонной корректирующей операции F вычисляется по формуле*

$$Q(F(B_1, \ldots, B_p)) = \sum_{r=\tilde{q}+1}^{q} \varphi_r(t_r).$$

Доказательство. Легко убедиться, что множество дефектных пар алгоритмического оператора $F(B_1, \ldots, B_p)$ совпадает с множеством нарушений порядка в последовательности $\mathbb{I}_q$.

Возьмём произвольный элемент (i_s, i_t) множества $\mathbb{D}(\mathbb{I}_q)$. По определению имеем $t < s$ и $y_{i_s} < y_{i_t}$. Из первого неравенства вытекает $y'_{i_t} < y'_{i_s}$, что в силу (2.41) означает $F(a_{i_t}) < F(a_{i_s})$, следовательно $(i_s, i_t) \in \mathbb{D}(F(B_1, \ldots, B_p))$.

Пусть теперь (i_s, i_t) — произвольный элемент множества $\mathbb{D}(F(B_1, \ldots, B_p))$. Тогда $y_{i_s} < y_{i_t}$ и $y'_{i_t} < y'_{i_s}$. Из первого неравенства и (2.40) следует $t < s$. Сопоставляя это со вторым неравенством, заключаем, что $(i_s, i_t) \in \mathbb{D}(\mathbb{I}_q)$.

Таким образом, имеет место равенство $\mathbb{D}(F(B_1, \ldots, B_p)) = \mathbb{D}(\mathbb{I}_q)$, откуда, с учётом теоремы 10 получаем требуемую формулу.

Теорема доказана.

3 Проблемно-зависимые подзадачи и язык ASDIEL

Рассматривая в §1.4 подзадачи, возникающие при построении локальных базисов, мы условно разделили их на проблемно-независимые и проблемно-зависимые. К числу первых относятся задачи оптимизации базисных алгоритмических операторов и корректирующих операций, рассмотренные в предыдущей главе. В отличие от них проблемно-зависимые подзадачи не решаются едиными формальными методами. Способ их решения, и даже сама постановка, зависят от предметной области и специфических особенностей исходной информации. К их числу относятся в основном подзадачи, связанные с выбором структуры алгоритмической суперпозиции.

Обычно имеющаяся неформализованная или слабо формализованная информация о решаемой задаче привлекается именно на этапе построения суперпозиции. Таким образом структура суперпозиции, то есть последовательность применения алгоритмов, а также критерии и методы их настройки, — оказываются проблемно-зависимыми. Проиллюстрируем это следующим рядом соображений.

1. Суперпозиция может включать в себя узкоспециальные алгоритмы, в которых реализованы математические модели предметной области. Обычно использование готовых построенных и апробированных моделей повышает качество решения. При этом они могут применяться наряду с более общими методами обучения по прецедентам. Таким образом, структура суперпозиции может зависеть от степени изученности предметной области.

2. В случаях неполных, неточных и/или разнородных данных, а также при наличии особых точек или выбросов, возникает необходимость применения специальных алгоритмов предварительной обработки данных. Поэтому структура суперпозиции может зависеть также и от свойств входной информации.

3. В роли объектов могут выступать элементы различных по своей природе множеств, а также их декартовых произведений. Например, в динамических задачах объектом исследования может быть как фиксированный объект в фиксированный момент времени, так и объект со всей его предысторией, или совокупность объектов в некоторый момент времени. Размерность входной информации и степень детальности требуемого исследования тоже влияют на структуру суперпозиции.

Приведённые соображения свидетельствуют о том, что на практике структура суперпозиции сложным образом зависит от конкретных входных данных и совокупности содержательных представлений о решаемой задаче. Эта зависимость в общем случае не поддаётся формализации. Поэтому для решения проблемно-зависимых подзадач предлагается применять не формальные методы, а разработанное автором инструментальное средство ASDIEL (Algorithmic Superpositions Description and Investigation: Environment and Language), включающее язык описания алгоритмических суперпозиций. Основная идея использования специализированного языка заключается в том, чтобы свести в одном компактном описании все структурные особенности конструируемого алгоритма, предоставив исследо-

вателю возможность варьировать их, постепенно подстраиваясь под решаемую задачу. В настоящий момент существует и используется реализация этого языка в виде интерпретатора, снабжённого пошаговым отладчиком.

3.1 Введение в язык алгоритмических суперпозиций

Область применения языка ASDIEL не ограничивается построением локальных базисов в задачах распознавания. Вообще говоря, он предназначен для построения и настройки сложных процедур обработки данных и подходит для решения широкого класса задач в самых разных прикладных областях. Его целесообразно применять, когда искомая процедура допускает представление в виде суперпозиции стандартных алгоритмов. В случае задач распознавания, классификации и прогнозирования к числу стандартных относятся алгоритмы аппроксимации, кластеризации, шкалирования, оценивания близости, проецирования и другие. В частности, модели алгоритмических операторов и корректирующих операций, рассмотренные в предыдущей главе, также реализуются в виде стандартных алгоритмов. *Язык* ASDIEL *предназначен для описания суперпозиций, составленных из стандартных алгоритмов анализа и преобразования данных.*

На практике состав и структура искомой суперпозиции, как правило, заранее неизвестны — их приходится подбирать в ходе вычислительных экспериментов на реальных данных. При этом описание суперпозиции следует рассматривать как сценарий очередного эксперимента, и только по завершении исследований — как запись решения поставленной задачи. Таким образом ASDIEL *является одновременно языком описания вычислительных экспериментов для задач обработки данных.*

Почему приходится строить суперпозиции алгоритмов?

Язык ASDIEL задумывался как инструментальное средство, пригодное для решения широкого класса задач. Поэтому в его основу была положена библиотека методов, предназначенных для решения типовых подзадач с максимально широким спектром приложений. При таком подходе решение прикладной задачи сводится к описанию суперпозиции, составленной из нескольких стандартных методов. Необходимость представлять искомый алгоритм в виде суперпозиции продиктована не только использованием алгебраического подхода, но и рядом других обстоятельств, перечисляемых ниже.

- Исходные данные могут оказаться разнотипными, неполными, неточными или косвенными. Это потребует предварительной обработки данных с применением одного или нескольких стандартных алгоритмов, например:
  - в случае разнотипных признаков — приведение их к одной шкале;
  - в случае слишком обширного множества объектов — решение задачи кластеризации;

  - при наличии нетипичных объектов — оценивание нетипичности и устранение выбросов;
  - в случае неполных или разреженных данных — выделение наиболее информативной части данных и/или заполнение пропусков.

  Включение алгоритмов предобработки естественным образом приводит к возникновению суперпозиции.
- Некоторые алгоритмы, такие как АВО [15] или МГУА [22], сами имеют структуру суперпозиции. Причём далеко не всегда её выбор может быть полностью возложен на машину. Исследователь должен иметь возможность задавать отдельные её части непосредственно или получать с помощью других алгоритмов. Поэтому сложные многоступенчатые алгоритмы целесообразно разделять на более простые, предоставив исследователю возможность вмешаться в процесс построения такого алгоритма.
- В конкретной задаче может понадобиться то или иное специальное преобразование данных (например математическая модель некоторой части предметной области), которое можно реализовать как отдельный алгоритм и применять его наряду со стандартными.
- Наконец, при передаче данных от одного алгоритма другому часто приходится выполнять различные вспомогательные преобразования, которые также являются отдельными элементами суперпозиции.

Почему для описания суперпозиции нужен специальный язык?

Построение суперпозиции не ограничивается выбором нескольких стандартных алгоритмов и последовательности их выполнения. В нетривиальных прикладных задачах возникают различные дополнительные вопросы:

- как выделить из имеющегося множества объектов обучающую и контрольную выборки?
- как отсеять заведомо непоказательные или исключительные объекты?
- какую функцию близости объектов предпочесть?
- как оптимизировать состав признаков, подаваемых на вход того или иного алгоритма?
- какие функциональные преобразования повысят информативность признаков?
- какие значения придать тем параметрам алгоритмов, которые не определяются автоматически в результате обучения?
- какой способ хранения промежуточных данных выбрать, чтобы прийти к компромиссу между скоростью вычислений и размером требуемой памяти?

Ответы на эти вопросы, и даже сам круг вопросов, уникальны в каждом конкретном случае. В основном они касаются структуры суперпозиции и её компонент. Выбор структуры, как правило, наименее формализован и не может быть

осуществлён путём автоматической настройки. Чтобы описать суперпозицию, исследователь должен задать:

- состав суперпозиции и последовательность выполнения алгоритмов;
- структуру входных и выходных матриц каждого алгоритма;
- функциональные выражения, определяющие некоторые признаки как функции от других признаков.

Специализированный язык является подходящим средством для записи структурной информации такого вида. Он позволяет свести её в одном компактном описании, тем самым упростив проведение вычислительных экспериментов. Чтобы провести очередной эксперимент, достаточно слегка изменить описание и перезапустить интерпретатор языка.

В то же время язык скрывает от исследователя рутинные операции по формированию данных и организации вычислений. Ему не придётся вникать в тонкости хранения данных, реализации алгоритмов и их взаимодействия внутри суперпозиции. Все эти проблемы решены на уровне ядра ASDIEL.

Таким образом, специализированный язык позволяет сосредоточиться на содержательных сторонах решаемой задачи.

Каковы ключевые особенности языка ASDIEL?

- Благодаря общности модели данных и универсальности библиотечных методов его можно использовать в самых разных прикладных областях.
- Средства языка охватывают все этапы обработки данных в задачах восстановления зависимостей: импорт, предварительную обработку, настройку, расчёт, визуализацию и экспорт данных.
- Все библиотечные алгоритмы могут свободно сочетаться друг с другом и с обычными арифметическими выражениями, образуя суперпозиции.
- Входные и выходные матрицы алгоритмов описываются декларативно, как декартовы произведения упорядоченных множеств.
- Данные размещаются в оперативной памяти, в массивах произвольной размерности. Эффективное распределение памяти достигается за счёт отказа от хранения неиспользуемых или легко вычислимых ячеек массивов.
- Суперпозицию можно запускать в различных режимах, в том числе для настройки (`tune`) и вычисления (`calc`). Результат настройки можно сохранить как рабочую программу с фиксированными параметрами.
- Текст программы можно править во время отладочного запуска. При этом нет необходимости каждый раз перезапускать программу с самого начала — выполнение автоматически возобновляется с места правки.

Как решать прикладные задачи с помощью языка ASDIEL?

Язык ASDIEL предназначен в первую очередь для решения нетривиальных задач обработки данных, то есть таких задач, в которых ход решения изначально не ясен

и требуется проведение исследований. Опишем основные этапы решения подобных задач и то, каким образом язык ASDIEL используется на каждом из этапов. Отметим, что текст ASDIEL-программы многократно модифицируется в процессе поиска решения.

1. Выяснение структуры входных и выходных данных. Приведение всех данных к матричной форме представления, основанной на *наборах* и *массивах*. Описание наборов и массивов командой языка `USE`.
2. Загрузка входных данных из файлов (командой `IMPORT`) или непосредственно из прикладной программы.
3. Поиск и исследование различных зависимостей между отдельными признаками или группами признаков. Попытки применения стандартных методов из библиотеки ASDIEL для решения поставленной задачи; выяснение причин, по которым отдельные методы не дают качественных результатов.
4. Предварительная обработка данных в случаях неполных, разнородных, сложно структурированных данных. Для этого используются методы предобработки из библиотеки ASDIEL. Выделение выбросов и описание «особенных» областей путём формирования *поднаборов*.
5. При необходимости — разработка специфических для данной задачи *методов* и *алгоритмов* и их «пристыковка» к ASDIEL в виде динамической библиотеки. Отметим, что ASDIEL поддерживает только алгоритмическую форму представления знаний о предметной области.
6. Построение алгоритмической суперпозиции, при котором учитывается вся информация, полученная на предыдущих этапах исследования. Формализация и проверка всевозможных гипотез о наличии тех или иных закономерностей в данных, о целесообразности применения тех или иных методов, и т.д. Перебор гипотез осуществляется путём модификации текущего текста ASDIEL-программы, а их проверка — с помощью встроенных средств графического отображения информации (графиков, таблиц, сообщений). В процессе построения суперпозиции возможно итеративное возвращение к предыдущим этапам.
7. Настройка (обучение) и отладка суперпозиции. Оценивание её качества на контрольных выборках. Подбор оптимальных форм представления выходной информации. Сохранение настроенной суперпозиции в виде отдельной программы с жёстко фиксированными параметрами каждого метода. Такая программа уже не требует проведения настройки и представляет собой готовый алгоритм.
8. Использование настроенного алгоритма. Технически это реализуется как обращение к ядру ASDIEL из прикладной программы.

3.2 Модель данных ASDIEL

Приступая к решению задачи, необходимо выделить несколько упорядоченных множеств, которые в дальнейшем будут играть роль размерностей массивов данных. Например, это могут быть множества:

- исследуемых объектов;
- признаков (функций над объектами);
- моментов времени (в случае динамических задач);
- пар объектов;
- функций над парами объектов (расстояния, отношения, и т.д);
- свойств признаков (информативности, средние значения, и т.д);
- функций над парами признаков (близости, корреляции, и т.д).

Затем некоторым декартовым произведениям этих множеств сопоставляются массивы, предназначенные для хранения и представления данных. Все множества и массивы вводятся командой языка `USE`.

С помощью теоретико-множественных операций можно описывать произвольные подмножества и подмассивы, в частности — входы и выходы алгоритмов. Работа алгоритма состоит в том, чтобы считать данные из входных подмассивов, произвести вычисления и занести результат в выходные подмассивы. Однако само преобразование скрыто от пользователя, он обращается с алгоритмом как с «чёрным ящиком». Суперпозиция образуется в том случае, когда на вход одних алгоритмов подаются данные, генерируемые другими алгоритмами.

3.2.1 Терминология

Вводимая в данном параграфе терминология относится только к реализации языка ASDIEL и иногда не совпадает с общепринятой или использовавшейся ранее. Далее понятия «объект», «признак», «массив», «алгоритм», «метод» будут употребляться только в смысле языка ASDIEL, что не будет оговариваться особо.

Простой атом — элемент, имеющий имя.

Признак — простой атом, имеющий генератор (см. ниже). Генератором признака может быть алгоритм, выражение или внешняя таблица данных.

Объект — простой атом, не имеющий генератора.

Простой набор — линейно упорядоченное множество простых атомов, либо только объектов, либо только признаков.

Составной набор размерности k — декартово произведение вида $S = S_1 \times \ldots \times S_k$, где S_i — простые наборы, $k \geqslant 2$. Элементы составного набора упорядочены лексикографически.

Составной атом размерности k — элемент составного набора размерности k.

Атом — простой или составной атом.

Набор — простой или составной набор.

Поднабор — последовательность элементов набора (называемого *базовым* для данного поднабора). Один и тот же элемент может входить в поднабор несколько раз. Элементы поднабора не обязаны располагаться в том же порядке, что и в базовом наборе.

Массив $[\![F \times S]\!]$ *размерности* k — отображение, которое каждому элементу (f, s) декартова произведения $F \times S$ ставит в соответствие значение $\hat{f}(s)$, где:
F — набор признаков;
S — составной набор размерности $k-1$, либо простой набор при $k=2$;
$\hat{f}$ — отображение вида $S \to T_f$, называемое *генератором* признака f;
T_f — множество допустимых значений признака f, определяемое его типом (булевский, числовой, строковой, поднабор или подмассив).
Каждый признак входит в один и только один набор признаков. Каждый набор признаков определяет один и только один массив.

Подмассив — отображение, определённое на декартовом произведении $F' \times S'_1 \times \times \ldots \times S'_{k-1}$ и совпадающее на нём с массивом $[\![F \times S_1 \times \ldots \times S_{k-1}]\!]$, где $F' \subseteq F$ — поднабор признаков, $S'_i \subseteq S_i$ — поднаборы атомов.

Понижение размерности — группирование нескольких простых наборов массива в один составной. Например, если имеется массив $[\![F \times S_1 \times \ldots \times S_{k-1}]\!]$ и составной набор $S = S_1 \times \ldots \times S_j$, $j < k$, то массив можно представить в виде $[\![F \times S \times S_{j+1} \times \ldots \times S_{k-1}]\!]$. При этом его размерность понижается с k до $k - j + 1$.

Алгоритм — вычислимое отображение вида $(X_1, \ldots, X_m) \mapsto (Y_1, \ldots, Y_n)$, где
$X_1, \ldots, X_m$ — *входные* подмассивы;
$Y_1, \ldots, Y_n$ — *выходные* подмассивы.
Входные и выходные подмассивы алгоритма называются его *аргументами*.

Метод — совокупность конечного множества алгоритмов и конечного множества параметров. Любой алгоритм метода может свободно считывать и/или изменять значение любого параметра метода.

Действие — метод с единственным алгоритмом, не имеющим выходов.

Метод распознавания — метод, имеющий алгоритмы вычисления и настройки. *Алгоритм настройки*, называемый также алгоритмом обучения, устанавливает значения параметров метода. *Алгоритм вычисления* использует значения параметров, установленные алгоритмом настройки, при вычислении своих выходных подмассивов.

Внутренние параметры — параметры метода распознавания, вычисляемые алгоритмом настройки.

Внешние параметры — параметры метода распознавания, не вычисляемые алгоритмом настройки. Их значения явным образом задаются в программе и могут влиять на процедуру настройки.

3.2.2 Массивы

Массивы используются для хранения данных и результатов вычислений. Приведём некоторые наиболее типичные примеры массивов.

$[\![F \times S]\!]$ — массив «признаки–объекты», обычно используемый для представления исходных данных в задачах распознавания и статистическом анализе.

$[\![F \times S \times T]\!]$ — массив «признаки–объекты–моменты времени», являющийся обобщением предыдущего на случай динамических задач.

$[\![M \times S \times S]\!]$ — массив для представления функций близости объектов, бинарных отношений на множестве S и других функций над парами объектов.

$[\![M \times S \times S \times T]\!]$ — обобщение предыдущего на случай динамических задач.

$[\![P \times F]\!]$ — массив для хранения различных свойств признаков: диапазонов допустимых значений, весов или информативностей, характеристических векторов подмножеств признаков.

$[\![D \times F \times F]\!]$ — массив для представления функций на парах признаков, например корреляций или близостей признаков.

В каждом массиве имеется одна выделенная размерность — набор признаков, отвечающий за запись и считывание значений из массива. То, каким образом значения вычисляются и хранятся, зависит от способа генерации и способа хранения каждого конкретного признака.

Возможны три способа генерации значений признака.

- Значения признака считываются из текстового файла или таблицы базы данных. Данный способ генерации характерен для признаков, используемых в качестве исходных данных.
- Значения признака вычисляются некоторым алгоритмом — для всех признаков, фигурирующих в выходных подмассивах алгоритмов.
- Значения признака вычисляются по заданному выражению — для всех признаков, правило вычисления которых явным образом задано в программе в виде выражения.

Возможны три способа хранения значений признака.

- Значения признака хранятся в оперативной памяти с прямым доступом — для одномерных или бинарных упакованных признаков.
- Значения признака хранятся в оперативной памяти с индексированным доступом. В текущей реализации этот способ предпочтителен для многомерных разреженных массивов с числом пропусков не менее 80%.
- Значения признака вычисляются заново при каждом обращении, для их хранения память не отводится.

Способ хранения указывается явным образом при создании признака. Пользователь имеет возможность существенно влиять на эффективность вычислений, выбирая способ хранения признаков, используемых для представления промежуточных результатов.

3.2.3 Методы и алгоритмы

В простейшем случае алгоритм вычисления преобразует входную информацию о заданном подмножестве объектов в выходную информацию о тех же объектах. Это можно представить как вычисление признака $f \in F$ по признакам поднабора F_0:

$$C : [\![F_0 \times S_0]\!] \mapsto [\![\{f\} \times S_0]\!], \quad \text{где} \quad S_0 \subseteq S.$$

Построение суперпозиций возможно благодаря тому, что результаты алгоритмов отождествляются с новыми признаками, которые затем указываются на входе других алгоритмов.

Алгоритм настройки в типичном случае требует указать на входе два подмассива: значения того же поднабора признаков F_0 и значения целевого признака $g \in F$ на фиксированной обучающей выборке S_1. Как правило, алгоритм настройки не имеет выходов, и вся его работа состоит исключительно в настройке внутренних параметров метода:

$$T : [\![F_0 \times S_1]\!],\ [\![\{g\} \times S_1]\!] \mapsto \varnothing, \quad \text{где} \quad S_1 \subseteq S.$$

Некоторые методы имеют алгоритм *дополняющей настройки*, действие которого состоит в расширении обучающей выборки S_1 ещё некоторым количеством объектов. Для многих методов данная операция выполняется гораздо эффективнее, чем полная перенастройка на расширенной выборке.

Кроме упомянутых типов методы могут располагать алгоритмами инициализации параметров, записи и чтения параметров через массивы.

3.3 Некоторые элементы языка ASDIEL

Подробное описание синтаксиса ASDIEL выходит за рамки данной работы. Ниже мы перечислим лишь некоторые ключевые конструкции, необходимые для понимания основных принципов языка.

Команда `USE` объявляет все наборы и массивы, которые будут использоваться в программе. Она состоит из ключевого слова `USE` и списка описателей, например:

```
USE F[S], M[S|S], P[F];
```

Описатель `F[S]` объявляет простой набор объектов `S`, набор признаков `F` и двумерный массив $[\![F \times S]\!]$. Описатель `M[S|S]` объявляет составной набор $S \times S$ и трёхмерный массив $[\![M \times S \times S]\!]$. Последний описатель объявляет набор признаков `P` и двумерный массив $[\![P \times F]\!]$. Символ вертикальной черты обозначает декартово произведение.

Команда `METHOD` создаёт метод и делает его активным. После ключевого слова `METHOD` указывается тип метода, например `LeastSquares` — метод наименьших

квадратов, `GenMetrics` — метод генерации метрик, и т. д. Все команды описания алгоритмов и все обращения к параметрам будут относиться именно к этому методу вплоть до следующей команды `METHOD`.

Методу можно присвоить отдельное имя с помощью расширенной формы команды `METHOD`:

`METHOD` ТипМетода `NAMED` ИмяМетода`;`

и затем повторно активизировать данный метод:

`METHOD NAMED` ИмяМетода`;`

Повторные обращения к одному и тому же методу используются при описании итерационных процедур настройки, в частности, при построении локальных базисов по схеме, описанной в §1.3. Типовые фрагменты ASDIEL-программ, использующие итеративные схемы настройки, будут приведены в §3.5.

Команда описания алгоритма начинается именем алгоритма, за которым следует список входных подмассивов, символ `->`, и список выходных подмассивов. Подмассивы задаются декартовыми произведениями поднаборов. Запись команды напоминает формальное определение алгоритма (см. стр. 59), например:

```
calc S|F{x,y,z}, S|F{weight} -> S|F{result};
```

Параметры алгоритма используются точно так же, как и обычные переменные, за тем исключением, что их имена начинаются символом `@`. Для каждого метода все его параметры организованы в иерархическую структуру наподобие файловой системы, причём разделитель `@` играет ту же роль, что и символ `\` в полном имени файла:

`@n` — параметр с именем `n`;
`@coeff@9` — 9-ый элемент векторного параметра `coeff`;
`@dist@5@7` — $(5, 7)$-ый элемент двумерного массива параметров `dist`;

Выражения используются в трёх основных контекстах. Во-первых, также как и в обычных языках программирования, значения выражений можно присваивать переменным, выводить на печать, передавать в качестве аргументов функциям и командам, и т. д. Во-вторых, выражения, содержащие операцию декартова произведения, служат для формирования входных и выходных подмассивов алгоритмов. Наконец, с помощью выражений можно определять одни признаки как функции других признаков. Выражения могут иметь различные типы: булевский, числовой, строковой, поднабор, подмассив, и т.д.

Ключевую роль в ASDIEL играют выражения-поднаборы. Исходным материалом для их построения являются перечисления — поднаборы, задаваемые списком элементов. В перечисление могут входить не только отдельные атомы, но ещё диапазоны (атомы, расположенные подряд), и серии (атомы, имена которых вычисляются по общему правилу):

`S{obj3, A, B, C, #56..#100}` — поднабор набора `S`, состоящий из объектов с именами `obj3`, `A`, `B`, `C` и диапазона объектов с 56-го по 100-й включительно.

`new S{obj::500}` — поднабор, представляющий собой серию из пятисот объектов `obj1`, `obj2`,..., `obj500`. Ключевое слово `new` говорит, что если таких объектов ещё нет в наборе `S`, они будут созданы.

`S{*}` — поднабор, совпадающий со всем набором.

`F{x, y, x+y, z=log(x^2+y^2)}` — поднабор признаков, состоящий из двух ранее имевшихся признаков `x`, `y` и вычислимых признаков `x+y`, `z`.

`new M{dist::F{x,y,z}}` — поднабор, составленный из серии атомов, имена которых формируются из имён атомов другого поднабора. В данном случае в наборе признаков `M` создаются атомы `distx`, `disty`, `distz`.

`new Taxon{class::S{*}|F{result}}` — конструкция, применяемая для преобразования множества значений в новые элементы некоторого набора. В данном случае, если в подмассиве `S{*}|F{result}` встречаются значения, скажем, 0, 1, 15, 99, то в базовом наборе `Taxon` будут созданы атомы с именами `class0`, `class1`, `class15`, `class99`.

Язык располагает полным набором операций для работы с упорядоченными множествами (поднаборами), позволяя образовывать их объединения, пересечения, конкатенации и декартовы произведения. Имеются операции понижения размерности, выборки по условию и сортировки по одному или нескольким признакам:

`S{obj1..obj100}|F{x,y,z}` — декартово произведение двух поднаборов, представляющее подмассив массива $[\![F \times S]\!]$ размером 100×3.

`S{obj1..obj100}|S{obj1..obj100}|M{d}` — трёхмерный подмассив массива $[\![M \times \times S \times S]\!]$ размером $100 \times 100 \times 1$.

`S{obj1..obj100}|(S{obj1..obj100}|M{d})` — двумерный подмассив в массиве $[\![M \times S \times S]\!]$ размером 100×100. Скобки выполняют понижение размерности.

`S{s::500}:F{x<5}` — Поднабор объектов из `S`, выбранных из `s1`,..., `s500`, для которых значение признака `x` меньше 5.

`S{s::500}:F{x<5}$F{y}` — Тот же поднабор, дополнительно отсортированный по возрастанию значений признака `y`.

`S{s::500}:F{x<5} & S{A,B,C}` — Тот же поднабор, к концу которого присоединены объекты `A`, `B`, `C`.

3.4 Обоснование достаточности модели данных для решения прикладных задач

Одно из основных преимуществ языка ASDIEL заключается в том, что он подходит для решения широкого класса прикладных задач. Гибкость и универсальность языка обеспечиваются двумя важными свойствами модели данных:

- класс реализуемых в её рамках алгоритмов достаточно широк;
- возможно построение произвольных суперпозиций этих алгоритмов.

Второе свойство обеспечивается механизмом построения суперпозиций, основанным на образовании пересечений между входными и выходными подмассивами различных алгоритмов. Очевидно, этот механизм является достаточно общим и позволяет строить произвольные алгоритмические суперпозиции.

Первое свойство обеспечивается универсальностью введённых определений *метода* и *алгоритма*. Они охватывают широкий класс моделей алгоритмов, реально используемых на практике. По сути дела этот класс фиксируется двумя условиями: (а) модель является параметрической и (б) входные и выходные данные алгоритмов представляются в матричном виде. Как правило методы распознавания и восстановления зависимостей, работающие с признаковыми описаниями объектов, удовлетворяют этим двум условиям.

Для обоснования данного факта был проведён анализ большого числа различных методов, активно применяемых при решении прикладных задач [5, 12, 15, 22, 20, 24, 34, 46]. Данный анализ состоял в приведении методов к единому способу описания, диктуемому моделью данных ASDIEL. По сути дела, описанные в таком виде, они образуют библиотеку методов с единым интерфейсом и правилами применения. Нашей ближайшей целью является изложение результатов проведённого анализа.

Разумеется, полное описание самих методов выходит за рамки настоящей работы. Многие из них являются стандартными и подробно рассмотрены в литературе. Здесь лишь вкратце объясняется назначение каждого метода.

Типовую структуру алгоритмов методов будем описывать с помощью шаблонов. *Шаблон метода* — это формальное правило, которому необходимо следовать, чтобы корректно описать данный метод в ASDIEL-программе. Шаблон указывает, сколько алгоритмов имеет метод и каковы их имена, сколько входных и выходных аргументов у каждого алгоритма, какова размерность каждого аргумента, и какие имеются соответствия между размерностями различных аргументов.

При записи шаблонов соблюдаются следующие правила.

1. Предполагается, что базовые наборы и массивы описаны командой `USE F[S], M[S|S], P[F], D[F|F]`.
2. Поднаборы, которые могут состоять из произвольного числа элементов, обозначаются соответствующими прописными буквами, совпадающими с именем базового набора, и снабжаются индексами: S_t, S_c, F_0.
3. Поднаборы, состоящие из небольшого фиксированного числа элементов, записываются в виде перечислений: $F\{g\}$, $M\{\rho\}$, $P\{\alpha, \beta, \gamma\}$. Необязательные элементы выделяются квадратными скобками: $F\{g, [r], [w]\}$.
4. Необязательные аргументы алгоритмов заключаются в скобки: $\Big[P_N \mid Q\{\alpha\}\Big]$.
5. Операция понижения размерности, так же как и в ASDIEL, записывается с помощью обычных скобок, например: $S_1 \mid S_2 \mid M\{r\}$ — трёхмерный подмассив, $S_1 \mid (S_2 \mid M\{r\})$ — двумерный подмассив.

Замечание 1. Запись аргумента фиксирует размерность и ориентацию подмассива, но не базовый массив. Если в шаблоне записано $S_t \mid F_0$, то соответству-

ющий аргумент должен быть двумерной матрицей, но не обязан быть частью массива `F[S]`. Реально на его месте может оказаться подмассив вида $(S_1 \mid S_2) \mid M_0$ из массива `M[S|S]`, или $F_0 \mid P_0$ из массива `P[F]`, и т.д.

Замечание 2. Если один и тот же поднабор фигурирует в нескольких аргументах (см. S_t и F_0 в шаблоне метода `LeastSquares`), то совершенно необязательно, чтобы на его место реально подставлялся один и тот же поднабор. Единственное требование заключается в том, чтобы соответствующие размерности матриц совпадали.

3.4.1 Метод наименьших квадратов

Методы `LeastSquares` и `LeastSquaresMonot` применяются для образования нового признака y как линейной комбинации признаков поднабора F_0, аппроксимирующей заданный целевой признак g. Оба метода можно использовать при построении локальных базисов для задач восстановления регрессии в случаях линейных и полиномиальных корректирующих операций (см. §2.1.1, §2.2.1). Различные варианты метода наименьших квадратов подробно описаны в [28].

`METHOD LeastSquares;`
`tune` $S_t \mid F_0$, $S_t \mid F\{g\}$, $\Big[S_t \mid F\{w\}\Big]$;
`calc` $S_c \mid F_0$ `->` $S_c \mid F\{y\}$;
`save ->` $F_0 \mid P\{a\}$;
`load` $F_0 \mid P\{a\}$;

Алгоритм вычисления `calc` для всех объектов $s \in S_c$ рассчитывает значение нового признака $y \in F$ как линейную комбинацию всех признаков поднабора $F_0 \subset F$:

$$y(s) = \sum_{f \in F_0} a_f f(s), \tag{3.1}$$

где $\{a_f\}_{f \in F_0}$ — параметры метода.

Алгоритм настройки `tune` определяет коэффициенты линейной зависимости a_f из условия аппроксимации целевого признака $g \in F$ на заданной обучающей выборке объектов $S_t \subseteq S$. Для этого минимизируется функционал среднеквадратичной невязки

$$Q = \sum_{s \in S_t} w(s)\Big(y(s) - g(s)\Big)^2,$$

Если третий аргумент алгоритма `tune` опущен, то веса $w(s)$ объектов обучающей выборки полагаются равными 1. Отметим, что наличие весовых коэффициентов позволяет использовать этот метод при настройке локальных базисов под полиномиальную корректирующую операцию (см. §2.2.1).

Алгоритм записи параметров `save` записывает значения параметров a_f в массив $[\![P \times F]\!]$ как свойство $a \in P$ признаков поднабора F_0.

Алгоритм загрузки параметров `load` выполняет действие, обратное алгоритму `save` — загружает значения параметров из подмассива $F_0 \mid P\{a\}$.

Метод `LeastSquaresMonot` имеет ту же структуру алгоритмов и ту же формулу вычисления (3.1), что и метод `LeastSquares`. Единственное его отличие заключается в том, что при настройке учитывается дополнительное ограничение на коэффициенты линейной комбинации: $a_f \geqslant 0$ для всех $f \in F_0$. Корректирующая операция, построенная данным методом, обладает свойством монотонности.

Метод `LeastSquaresMonot` можно, в частности, применять при настройке метрик, если вместо подмассивов вида $S \mid F$ использовать подмассивы $(S \mid S) \mid F$.

3.4.2 Метод построения линейной разделяющей поверхности

Метод `LinearDiscr` применяется для решения задачи классификации с двумя непересекающимися классами. При настройке метода решается задача поиска максимальной совместной подсистемы в системе линейных неравенств.

`METHOD LinearDiscr;`
`calc` $S_c \mid F_0$, $\Big[S_c \mid F\{r\}\Big]$ `->` $S_c \mid F\{B, [C]\}$;
`tune` $S_t \mid F_0$, $S_t \mid F\{g, [r]\}$;

Алгоритм вычисления `calc` для всех $s \in S_c$ вычисляет линейную комбинацию $B(s) = \sum\limits_{f \in F_0} a_f f(s)$, где $\{a_f\}_{f \in F_0}$ — параметры метода, а также результат применения к ней порогового решающего правила $C(s) = \theta(B(s) - r(s))$, где θ — функция Хевисайда, $r(s)$ — функция порога, заданная признаком $r \in F$.

Алгоритм настройки `tune` служит для определения параметров метода a_f из условий $C(s) = g(s)$ для всех $s \in S_t$. Данная задача решается путём поиска максимальной совместной подсистемы в системе линейных неравенств

$$\sum_{f \in F_0} a_f f(s) \lesseqgtr r(s), \qquad s \in S_t,$$

где в каждом из неравенств предполагается знак $\leqslant$, если $g(s) = 0$ и знак $>$, если $g(s) = 1$. Признак r должен быть одним и тем же в алгоритмах `calc` и `tune`. Если он опущен, полагается $r(x) \equiv 0$.

3.4.3 Вычисление дефекта набора алгоритмических операторов

Метод `MonotDefect` предназначен для вычисления дефекта набора алгоритмических операторов (см. стр. 24). Метод не имеет настраиваемых параметров и алгоритма настройки.

`METHOD MonotDefect;`
`calc` $S \mid F\{B_1, \ldots, B_p\}$, $S \mid \{y\}$
`->` $\Big[S \mid S \mid M\{w_{jk}, t^0_{jk}, t_{jk}\}\Big]$, $\Big[S \mid F\{w_k\}\Big]$;

Алгоритм вычисления `calc` для каждой пары объектов (s_j, s_k) из S находит число строго дефектных троек t^0_{jk} и дефектных троек t_{jk}, в основании которых

лежит пара (j,k) (см. стр. 27). Вычисляется также оценка $w_{jk} = t^0_{jk} + \frac{1}{2}t_{jk} + 1$ числа дефектных пар, устраняемых вместе с дефектной парой (j,k). Если пара (j,k) не принадлежит дефекту набора операторов $B_1, \ldots, B_p$, то $w_{jk} = t^0_{jk} = t_{jk} = 0$. Кроме того, для каждого объекта s из S по формуле (2.21) вычисляется вес $w_k(s)$, который является оценкой числа дефектных пар, устраняемых при точной настройке на k-ом объекте выборки S. Один из выходных аргументов может быть опущен, но не оба сразу.

3.4.4 Метод монотонной интерполяции

Метод `MonotInterp` предназначен для построения монотонных функций, проходящих через заданные q точек в пространстве размерности p. Для построения функции используется алгоритм настройки, описанный в §2.3.7. Метод может использоваться в качестве монотонной корректирующей операции.

`METHOD MonotInterp;`
`tune` $S_t \mid F_0$, $S_t \mid F\{y\}$;
`calc` $S_c \mid F_0$ `->` $S_c \mid F\{f\}$;

Алгоритм настройки `tune` строит монотонную функцию $f(a)$, удовлетворяющую q равенствам $f(a_k) = y_k$, $k = 1, \ldots, q$. Векторы a_k соответствуют строкам $q \times p$-матрицы $S_t \mid F_0$, а значения y_k — строкам $q \times 1$-матрицы $S_t \mid F\{y\}$.

Если исходные данные таковы, что построить монотонную функцию невозможно (последовательность пар $\{a_k, y_k\}_{k=1}^q$ не является монотонной), алгоритм `tune` отбрасывает некоторое количество векторов a_k, при этом выполнение интерполяционного условия на отброшенных векторах не гарантируется. Для минимизации ошибок, связанных с отбрасыванием дефектообразующих векторов, рекомендуется применять метод `Monotonize` перед методом `MonotInterp`.

Алгоритм вычисления `calc` производит расчёт значений построенной монотонной функции f для всех объектов поднабора S_c.

3.4.5 Метод монотонизации выборки

Метод `Monotonize` предназначен для исключения дефектообразующих элементов из заданной последовательности пар $\{a_k, y_k\}_{k=1}^q$. Для решения данной задачи используются два алгоритма монотонизации, рассмотренные в §2.3.8. Метод может применяться перед методом `MonotInterp` для повышения качества настройки путём корректировки целевого вектора. Метод не имеет настраиваемых параметров и алгоритма настройки.

`METHOD Monotonize;`
`calc` $S \mid F_0$, $S \mid F\{y\}$ `->` $S \mid F\{y'\}$;

Алгоритм вычисления `calc` корректирует целевой признак y таким образом, чтобы последовательность пар $\{a_k, y'_k\}_{k=1}^q$ оказалась монотонной (векторы a_k соответствуют строкам $q \times p$-матрицы $S_t \mid F_0$, а значения y_k — строкам $q \times$ $\times$ 1-матрицы $S_t \mid F\{y\}$). Для этого последовательно применяются оба алгоритма,

описанные в §2.3.8: алгоритм исключения дефектообразующих векторов и алгоритм обратного добавления дефектообразующих векторов. Скорректированные значения признака y записываются в выходной признак y'.

Если исходная последовательность $\{a_k, y_k\}_{k=1}^q$ не содержала дефектообразующих элементов, то гарантируется, что признак y' будет точной копией признака y на объектах выборки S.

3.4.6 Метод нормировки признаков

Применяется для образования новых признаков $F_0^* = \{f_1^*, \ldots, f_n^*\}$ путём нормировки заданных признаков $F_0 = \{f_1, \ldots, f_n\}$. Обычно этот метод применяется для того, чтобы уменьшить вычислительные погрешности в тех случаях, когда значения различных признаков отличаются на несколько порядков.

```
METHOD Normalize;
tune S_t | F_0;
calc S_c | F_0 -> S_c | F_0^*;
save -> F_0 | P{f_min, f_max};
load F_0 | P{f_min, f_max};
```

Алгоритм настройки `tune` вычисляет максимальные и минимальные значения признаков $f_1, \ldots, f_n$ на объектах поднабора S_t. Эти значения запоминаются как параметры метода.

Алгоритм вычисления `calc` для всех объектов $s \in S_c$ и всех $i = 1, \ldots, n$ вычисляет значения $f_i^*(s)$, нормированные относительно найденных при настройке максимальных и минимальных значений.

Алгоритм записи параметров `save` заносит минимальные и максимальные значения признаков из F_0 в подмассив $F_0 \mid P\{f_{\min}, f_{\max}\}$.

Алгоритм загрузки параметров `load` выполняет действие, обратное алгоритму `save` — загружает параметры из подмассива $F_0 \mid P\{f_{\min}, f_{\max}\}$.

3.4.7 Метод генерации признаков по функции расстояния

Метод `MetricProj` по заданной функции расстояния $\rho(s_1, s_2)$ между объектами вычисляет координаты этих объектов в евклидовом пространстве заданной размерности n. Координаты подбираются таким образом, чтобы евклидовы расстояния между объектами как можно точнее приближали исходное расстояние ρ. Фактически метрическое пространство отображается на евклидово [3]. При $n = 2$ метод можно использовать для визуализации взаимного расположения объектов на плоскости.

Функция расстояния задаётся как признак в массиве `M[S|S]`, а пространство евклидова представления — как набор признаков $F_0 = \{f_1, \ldots, f_n\}$ в массиве `F[S]`.

```
METHOD MetricProj;
tune S_t | (S_t | M{ρ}) -> S_t | F_0;
calc S_c | (S_t | M{ρ}) -> S_c | F_0;
```

Алгоритм настройки `tune` вычисляет матрицу евклидовых представлений $S_t \mid F_0$ объектов обучающей выборки S_t по матрице попарных расстояний между ними $S_t \mid (S_t \mid M\{\rho\})$.

Текущая реализация метода такова, что результат настройки существенным образом зависит от того, в каком порядке расположены объекты выборки S_t. В процессе вычислительных экспериментов выяснилось, что наиболее устойчивое взаимное расположение точек получается в случае, когда объекты упорядочены по убыванию расстояния до ближайшего из предыдущих объектов, то есть в порядке образования всё более мелкой сетки. Для такого упорядочивания объектов можно применить метод `DistOrder`, описанный ниже.

Алгоритм вычисления `calc` вычисляет проекции объектов поднабора S_c, минимизируя невязку между ρ и евклидовым расстоянием по всем парам объектов из множества $S_c \times S_t$.

3.4.8 Метод упорядочивания объектов по убыванию расстояний

Применяется для упорядочивания поднабора объектов по мере убывания расстояния до ближайшего объекта. Фактически объекты располагаются в порядке образования всё более мелкой сетки. Используется при формировании обучающих выборок для настройки методов, основанных на использовании функции расстояния между объектами, в частности метода `MetricProj`.

```
METHOD DistOrder;
tune S_t | S_t | M{ρ} -> S_t | F{ord};
```

Алгоритм настройки `tune` для каждого объекта $s \in S_t$ вычисляет порядковый номер объекта $\texttt{ord}(s)$. Сначала выбираются два объекта s_1 и s_2 из S_t, для которых расстояние $\rho(s_1, s_2)$ максимально, и полагается $\texttt{ord}(s_1) = 1$, $\texttt{ord}(s_2) = 2$. На каждом из последующих шагов выбирается объект s_i из S_t, $i = 3, 4, \ldots, |S_t|$, для которого расстояние до ближайшего из уже выбранных объектов $s_1, s_2, \ldots, s_{i-1}$ максимально, и полагается $\texttt{ord}(s_i) = i$.

Для расположения объектов выборки S_t в порядке возрастания значений признака `ord` достаточно применить стандартную операцию упорядочивания, записав S_t `$ F{ord}`.

3.4.9 Метод генерации метрик по признакам

Применяется для построения функций близости в пространстве `M[S|S]` по числовым признакам в пространстве `F[S]`.

```
METHOD GenMetrics;
calc S_c | F_0 -> S_c | S_c | M_0;
```

Алгоритм вычисления `calc` для каждого признака $f(s)$ из F_0 определяет соответствующую ему функцию близости m_f из M_0 по формуле $m_f(s_1, s_2) = |f(s_1) - f(s_2)|$ для всех s_1, s_2 из S_c.

3.4.10 Метод ближайших соседей

Применяется для образования нового признака y, который строится на основе расстояния ρ между объектами и аппроксимирует заданный целевой признак g. Различные модификации данного метода описаны в [46].

METHOD NearestNeighbours;
tune S_t | S_t | $\{\rho\}$, S_t | $\{g\}$;
calc S_c | S_t | $\{\rho\}$ -> S_c | $\{y\}$;

Алгоритм вычисления `calc` для всех объектов $s \in S_c$ определяет значение нового признака $y \in F$ как взвешенную сумму с нормированными весами

$$y(s) = \frac{1}{W_0} \sum_{s' \in S_t} g(s') w(\rho(s, s')), \quad \text{где} \quad W_0 = \sum_{s' \in S_t} w(\rho(s, s')).$$

Алгоритм настройки `tune` настраивает весовую функцию $w(\rho)$, минимизируя среднюю ошибку аппроксимации на обучающей выборке S_t.

3.4.11 Метод таксономии

Применяется для решения задачи таксономии (классификации без учителя) на основе заданной функции расстояния между объектами ρ. Различные методы таксономии подробно рассмотрены в [5, 20]

METHOD Taxonomy;
tune S_t | S_t | $\{\rho\}$;
calc S_c | S_t | $\{\rho\}$ -> $\Big[S_c \mid \{y\}\Big]$, $\Big[S_c \mid F_0\Big]$;

Алгоритм настройки `tune` находит таксономию выборки S_t. Число таксонов n либо задаётся заранее с помощью параметра `@n`, либо выбирается алгоритмом автоматически, если `@n` $= 0$.

Алгоритм вычисления `calc` для каждого объекта s из S_c вычисляет оценки принадлежности объекта s всем n таксонам. и номер таксона $y(s)$, к которому следует отнести объект s. Поднабор F_0 должен состоять из n элементов; например он может быть задан как `F{taxon::@n}`. Один из выходных аргументов может быть опущен, но не оба сразу.

3.4.12 Метод вычисления расстояния между признаками

Метод `FeaturesDist` применяется для вычисления попарных расстояний между признаками, измеренными в шкале порядка [20]. Вычисленная мера может применяться для таксономии признаков методом `Taxonomy`, §3.4.11; отбрасывания «похожих» признаков методом `DistOrder`, §3.4.8; визуализации взаимного расположения признаков с использованием метода `MetricProj`, §3.4.7; и т.д.

METHOD FeaturesDist;
calc S_c | F_0 -> F_0 | F_0 | $D\{d\}$;

Алгоритм вычисления `calc` для каждой пары признаков f_1, f_2 из F_0 находит расстояние между ними как среднюю меру «несогласованности» на всех парах объектов (мера Кенделла-Кемени [26, 27]):

$$d(f_1, f_2) = \frac{1}{C^2_{|S_c|}} \sum_{s_1, s_2 \in S_c} \delta_{f_1, f_2}(s_1, s_2),$$

где $\delta_{f_1,f_2}(s_1, s_2) = 0$, если порядковые отношения по признакам f_1 и f_2 одинаковы для объектов s_1 и s_2 (либо $f_1(s_1) < f_1(s_2)$ и $f_2(s_1) < f_2(s_2)$, либо $f_1(s_1) > f_1(s_2)$ и $f_2(s_1) > f_2(s_2)$), и $\delta_{f_1,f_2}(s_1, s_2) = 1$ в противном случае.

3.5 Примеры описания алгоритмических суперпозиций

В этом параграфе будет продемонстрировано использование языка ASDIEL для решения некоторых типовых задач классификации и восстановления регрессии. Под типовой задачей в данном случае понимается задача, в которой не фиксированы ни предметная область, ни исходные данные, ни применяемые модели алгоритмических операторов. В то же время при описании суперпозиции фиксируются семейства корректирующих операций, структура суперпозиции и стратегия её настройки. Это позволяет легко переносить приводимые ниже схемы настройки на решение различных прикладных задач.

Основная цель приводимых ниже примеров заключается в том, чтобы продемонстрировать реализацию итерационного процесса (1.1)–(1.4) на языке ASDIEL. При этом в явном виде показывается, что для настройки алгоритмических операторов по критериям (1.3) и (1.1), а также по комбинированному критерию (1.4), используется один и тот же алгоритм настройки.

Как и прежде, предполагается, что базовые наборы и массивы введены командой

```
USE F[S], M[S|S], P[F], D[F|F];
```

3.5.1 Абстрактные методы с алгоритмами стандартной структуры

В приводимых ниже примерах в качестве алгоритмических операторов будут использоваться абстрактные методы, алгоритмы которых имеют стандартную структуру, наиболее подходящую для решения того или иного класса задач. Рассмотрим два таких метода, дадим краткое описание их алгоритмов и содержательную интерпретацию входных и выходных аргументов.

1. Абстрактный метод восстановления регрессии:

`CALC` $S_c \mid F_0$ `->` $S_c \mid F\{y\}$;
`TUNE` $S_t \mid F_0$, $S_t \mid F\{g\}$, $\Big[S_t \mid F\{w\}\Big]$;

Алгоритм `CALC` вычисляет по матрице $S_c \mid F_0$ признаковых описаний объектов выборки S_c значения выходного признака $y(x)$ для всех $x \in S_c$.

Алгоритм настройки `TUNE` требует на входе матрицу $S_t \mid F_0$ признаковых описаний объектов обучающей выборки $S_t \subseteq S$, целевой вектор $S_t \mid F\{g\}$ и необязательный вектор весов объектов обучения $S_t \mid F\{w\}$. Настройка производится таким образом, чтобы y аппроксимировал g на выборке S_t.

Некоторые из выше описанных методов являются типичными представителями данного класса методов, в частности `LeastSquares`, `LeastSquaresMonot` и `MonotInterp`.

2. Абстрактный метод классификации с двумя непересекающимися классами:

`CALC` $S_c \mid F_0$, $\left[S_c \mid F\{r\}\right]$ `->` $S_c \mid F\{B, [C]\}$;
`TUNE` $S_t \mid F_0$, $S_t \mid F\{g, [r], [w]\}$;

Алгоритм `CALC` для всех $x \in S_c$ вычисляет оценку $B(x)$ и классификацию $C(x) = \theta(B(x) - r(x))$, где θ — функция Хевисайда, $r(x)$ — функция порога, заданная признаком $r \in F$.

Алгоритм настройки `TUNE` требует на входе матрицу $S_t \mid F_0$ признаковых описаний объектов обучающей выборки $S_t \subseteq S$. Второй аргумент — матрица $S_t \mid F\{g, r, w\}$ содержит для каждого обучающего объекта $x \in S_t$ его правильную классификацию $g(x) \in \{0, 1\}$, значение порога $r(x)$ и вес объекта $w(x)$. Настройка метода состоит в нахождении совместной подсистемы максимального веса для системы неравенств

$$\begin{cases} B(x) > r(x) & \text{при } g(x) = 1; \\ B(x) \leqslant r(x) & \text{при } g(x) = 0; \end{cases} \qquad \text{с весом } w(x) \text{ для всех } x \in S_t.$$

Вес подсистемы равен сумме весов входящих в неё неравенств. Если признак w опущен, предполагается $w(x) \equiv 1$ и задача сводится к поиску максимальной совместной подсистемы. Признак r обязательно должен быть одним и тем же в алгоритмах `CALC` и `TUNE`. Если он опущен, предполагается $r(x) \equiv 0$.

Типичным представителем данного класса методов является `LinearDiscr`.

3.5.2 Линейная коррекция в задаче восстановления регрессии

В этом примере строится простейшая суперпозиция, состоящая из двух алгоритмических операторов и одной корректирующей операции. Настройка суперпозиции производится итерационным процессом (1.1), (1.2). Для настройки алгоритмических операторов применяется комбинированный критерий (1.4) с фиксированным параметром $\lambda = 1/2$. Процесс останавливается после проведения заданного числа итераций.

Постановка задачи в терминах ASDIEL. Рассмотрим следующую задачу восстановления регрессии. Имеется набор объектов `S` и набор признаков `F`. Заранее сформированы поднабор признаков `base` $\subseteq$ `F` и поднабор объектов обучения `train` $\subseteq$ `S`. Подмассив `train|base` содержит исходные признаковые описания объектов обучения. Подмассив `train|F{goal}` содержит обучающую информацию. Для объектов вне поднабора `train` значения признака `goal`, вообще говоря, не

определены. Требуется построить в наборе F признак regress, аппроксимирующий goal на обучающей выборке train и вычислимый на произвольном объекте из S.

Решение задачи. Для простоты ограничимся использованием массива F[S] и методов, работающих только с признаковыми описаниями объектов.

Первый шаг состоит в попытке применения некоторого стандартного метода Method1, имеющего структуру абстрактного метода восстановления регрессии (см. выше):

```
METHOD Method1 NAMED AlgOp1;
TUNE train|base, train|F{goal};
CALC S|base -> S|F{m1};
```

В наборе F образуется новый признак m1, аппроксимирующий goal. Отметим, что в алгоритме CALC вместо конкретной выборки объектов указан весь набор S. Тем самым гарантируется, что признак m1 будет корректно вычислен на любом объекте из S, даже если этот объект будет создан и добавлен в S после выполнения команды CALC.

Качество аппроксимации легко проверить с помощью отладочной печати:

```
print « S|F{goal, m1, abs(goal-m1)};
```

либо построив график

```
Chart S|F{x, goal, x, m1}, @Format="XY C=1/ XY C=0 L=2";
```

где x — произвольный признак из поднабора base, значения которого используются здесь в качестве абсцисс точек. На графике точками цвета 1 будут выделены значения признака goal, а ломаной цвета 2 соединены точки, соответствующие значениям признака m1.

Если в результате проверки выяснится, что качество аппроксимации приемлемо, то построение алгоритма на этом завершается.

Если же это не так, то для решения задачи применим алгебраический подход. Возьмём ещё один метод:

```
METHOD Method2 NAMED AlgOp2;
TUNE train|base, train|F{goal};
CALC S|base -> S|F{m2};
```

Здесь существенно, чтобы методы Method1 и Method2 были различны, иначе мы получим одинаковые признаки m1 и m2.

Построение суперпозиции заключается в том, чтобы применить корректирующую операцию к выходным признакам m1 и m2 алгоритмов вычисления методов AlgOp1 и AlgOp2 соответственно. Используем для этой цели метод наименьших квадратов:

```
METHOD LeastSquares NAMED Corrector;
TUNE train|F{m1,m2}, train|F{goal};
CALC S|F{m1,m2} -> S|F{regress};
```

В результате признак `regress` будет вычисляться как линейная комбинация признаков `m1` и `m2`:

$$\texttt{regress} = \alpha_1 \cdot \texttt{m1} + \alpha_2 \cdot \texttt{m2},$$

где коэффициенты линейной зависимости α_1 и α_2 являются параметрами метода `LeastSquares` с именами `@a@0` и `@a@1`.

Таким образом, построенная суперпозиция имеет вид $F(B_1, B_2)$, где B_1, B_2, F — алгоритмы `CALC` методов `AlgOp1`, `AlgOp2` и `Corrector` соответственно. Первые два играют роль алгоритмических операторов, третий — корректирующей операции.

На этом этапе можно снова проверить качество аппроксимации, теперь уже для признака `regress`. Для повышения качества применим итерационный процесс (1.1)–(1.4), в котором будем поочерёдно подстраивать методы `AlgOp1`, `AlgOp2` и корректирующую операцию `Corrector`. Процесс завершается после проведения десяти итераций.

```
iter = 10;
while iter<10;
  L1 = 1/2; L2 = 1/2;
  a1 = @a@0; a2 = @a@1;
  = F {
    g1 = ((1-L1)*goal + L1*a1*(goal-m2*a2)) / (1-L1+L1*a1*a1),
    g2 = ((1-L2)*goal + L2*a2*(goal-m1*a1)) / (1-L2+L2*a2*a2),
  };
  METHOD NAMED AlgOp1;
  TUNE train|base, train|F{g1};
  METHOD NAMED AlgOp2;
  TUNE train|base, train|F{g2};
  METHOD LeastSquares NAMED Corrector;
  TUNE train|F{m1,m2}, train|F{goal};
  iter = iter + 1;
end while;
```

Каждая итерация состоит в последовательной перенастройке методов `AlgOp1`, `AlgOp2` и `Corrector`. При этом целевые признаки `g1` и `g2` вычисляются на каждой итерации по формуле (2.3).

Как уже говорилось в §1.3, алгоритмы настройки, изначально предназначенные для решения задач вида (1.3), без каких-либо изменений можно использовать при решении задач построения локальных базисов (1.1) и (1.4). Приведённая программа демонстрирует применение этого принципа на практике. Нам не пришлось реализовать новый алгоритм настройки методов `Method1` и `Method2`, специально предназначенный для оптимизации алгоритмических операторов с учётом их места в суперпозиции. Вместо этого мы используем прежний алгоритм настройки, корректируя на каждом шаге целевой вектор.

Замечание 1. Параметры `@a@0` и `@a@1` принадлежат методу `Corrector`, так как именно его команда `METHOD` выполняется перед началом каждой итерации.

Замечание 2. Внутри итерационного цикла нет необходимости определять заново алгоритмы вычисления `CALC`, так как перенастройка меняет внутренние параметры методов, но не затрагивает структуры алгоритмов.

Замечание 3. Для настройки алгоритмических операторов был использован комбинированный критерий настройки с параметром $\lambda = 0.5$. Соответствующие переменные `L1` и `L2` в данном примере фиксированы, однако в общем случае их можно менять от итерации к итерации.

В результате настройки получается алгоритмическая суперпозиция, выходом которой является признак `regress`. Пусть задана рабочая выборка `work` — некоторый поднабор объектов из `S`. Для вычисления значений признака `regress` на этой выборке достаточно обратиться к подмассиву `work|F{regress}`, скажем, выдав его на печать:

```
print « work|F{m1, m2, regress};
```

После того, как суперпозиция настроена, интерпретатор ASDIEL «знает», какие алгоритмы, в какой последовательности и с какими параметрами следует вызывать, чтобы рассчитать финальную информацию.

3.5.3 Монотонная коррекция в задаче восстановления регрессии

В этом примере, как и в предыдущем, демонстрируется использование итерационного процесса (1.1), (1.2) при решении задачи восстановления регрессии. Однако теперь схема настройки несколько сложнее. Во-первых, в качестве корректирующей операции используется метод монотонной интерполяции, описанный в §2.3.7. Это требует дополнительного применения двух методов: оценивания дефектности пар объектов обучения (см. §2.3.2, §3.4.3) и монотонизации выборки (см. §2.3.8, §3.4.5). Во-вторых, число алгоритмических операторов, в отличие от предыдущего примера, не фиксируется. Дополнительные операторы добавляются до тех пор, пока не будет достигнуто заданное значение функционала качества на контрольной выборке.

Постановка задачи остаётся прежней, за тем исключением, что дополнительно задана контрольная выборка `control` $\subseteq$ `S`, на которой оценивается качество обучения.

Решение задачи. Первый шаг состоит в применении некоторого стандартного метода восстановления регрессии:

```
METHOD MainMethod NAMED AlgOp1;
TUNE train|base, train|F{goal};
CALC S|base -> S|F{y1};
Q = avr (control|F{abs(y1/goal-1)});
```

В наборе `F` образуется новый признак `y1`, аппроксимирующий `goal`. Качество аппроксимации `Q` вычисляется как средний модуль относительного отклонения `y1` от `goal` по выборке `control`.

Если качество аппроксимации неприемлемо, то для построения алгоритма применяется алгебраический подход. Алгоритмические операторы добавляются до тех пор, пока значение функционала качества не станет меньше заданного.

Процедура добавления очередного оператора состоит из пяти шагов:

1. Вычисление дефекта ранее построенных операторов и весов объектов обучения методом `MonotDefect`.
2. Добавление алгоритмического оператора и его настройка с учётом найденных весов. Все операторы берутся из одного и того же семейства $\mathfrak{M}^0$, определяемого абстрактным методом `MainMethod`.
3. Монотонизация выборки `train` методом `Monotonize`, в результате которой формируется модифицированный целевой признак `goal_m`, см. §3.4.5.
4. Монотонная коррекция совокупности построенных операторов методом монотонной интерполяции `MonotInterp`, см. §3.4.4.
5. Вычисление функционала качества `Q` для построенной суперпозиции. Дальнейшее построение прекращается, как только окажется `Q<0.1`.

Для предотвращения зацикливания число операторов ограничивается фиксированным значением, в данном случае 20.

```
oper = 2;
while Q>0.1 and oper<20;
  ! шаг 1:
  METHOD MonotDefect;
  BasePrev = F{y::(oper-1)};
  CALC train|BasePrev, train|F{goal} -> , train|F{w};
  ! шаг 2:
  METHOD MainMethod NAMED 'AlgOp[oper]';
  Lambda = 1/2;
  TUNE train|base, train|F{goal}, train|F{1+Lambda*(w-1)};
  CALC S|base -> S|F{'y[oper]'};
  ! шаг 3:
  BaseCurr = F{y::oper};
  METHOD Monotonize;
  CALC train|BaseCurr, train|F{goal} -> train|F{goal_m};
  ! шаг 4:
  if oper == 2;
    METHOD MonotInterp NAMED Corrector;
  else;
    METHOD NAMED Corrector;
  end if;
  TUNE train|BaseCurr, train|F{goal_m};
  CALC S|BaseCurr -> S|F{regress};
  ! шаг 5:
```

```
  Q = avr (control|F{abs(regress/goal-1)});
  oper = oper + 1;
end while;
p = oper -1;
```

В результате будет построена суперпозиция $F(B_1, \ldots, B_p)$, где $B_1, \ldots, B_p, F$ — алгоритмы `CALC` методов `AlgOp1`, ..., `'AlgOp[p]'` и `Corrector` соответственно. Первые p играют роль алгоритмических операторов, последний — корректирующей операции.

Замечание 1. Языковая конструкция `'AlgOp[oper]'` является идентификатором, в котором вместо `[oper]` подставлено текущее значение переменной `oper`. Таким образом, при первом проходе цикла создаётся метод `AlgOp2` и признак `y2`, при втором проходе — метод `AlgOp3` и признак `y3`, и так далее.

Замечание 2. В отличие от алгоритмических операторов корректирующая операция создаётся только один раз при первом проходе тела цикла. В дальнейшем мы только обращаемся к уже имеющемуся методу по имени `Corrector`, чтобы произвести его перенастройку. Это обеспечивается командами `if...else...end` в начале шага 4.

Если значение функционала качества `Q` по выходе из цикла `while` достаточно мало, то построение суперпозиции на этом можно завершить. Однако более целесообразным представляется проведение ещё нескольких итераций для повторной поочерёдной настройки всех алгоритмических операторов. Теперь каждая итерация состоит из трёх шагов:

1. перенастройка всех p операторов $F(B_1, \ldots, B_p)$ по очереди;
2. перенастройка корректирующей операции;
3. вычисление функционала качества.

Итерации прекращаются при достижении заданного качества (`Q=0.01`), либо по завершении фиксированного числа итераций (`iter=10`).

```
iter = 0;
while Q>0.01 and iter<10;
  ! шаг 1:
  i=1; while i<=p;
    base_i = F{y::1..i-1, y::i+1..p};
    METHOD MonotDefect;
    CALC train|base_i, train|F{goal} -> , train|F{w};
    METHOD NAMED 'AlgOp[i]';
    Lambda = 1/2;
    TUNE train|base, train|F{goal}, train|F{1+Lambda*(w-1)};
    i = i + 1;
  end while;
  ! шаг 2:
  METHOD Monotonize;
  CALC train|F{y::p}, train|F{goal} -> train|F{goal_m};
```

```
    METHOD NAMED Corrector;
    TUNE train|F{y::p}, train|F{goal_m};
    ! шаг 3:
    Q = avr (control|F{abs(regress/goal-1)});
    iter = iter + 1;
  end while;
```

В результате настройки получается алгоритмическая суперпозиция, выходом которой является признак `regress`. Для вычисления признака `regress` на рабочей выборке `work` достаточно обратиться к подмассиву `work|F{regress}`:

```
print « work|F{y::p, regress};
```

3.5.4 Полиномиальная коррекция в задаче классификации

Полиномиальные корректирующие операции над множествами некорректных алгоритмов были впервые введены Ю.И. Журавлёвым [14, 15] при рассмотрении задачи классификации. Им было установлено, что в классе полиномиальных расширений модели алгоритмов вычисления оценок (АВО) существуют корректные алгоритмы и предложен конструктивный способ построения такого алгоритма.

В следующем примере строится полином, отличающийся от рассмотренных в классических работах Ю.И. Журавлёва следующим:

1. Для простоты рассматривается задача классификации с двумя непересекающимися классами и берётся полином частного вида, в котором каждый одночлен состоит только из двух сомножителей:

$$A(x) = \theta\Big(\sum_{i=1}^{p} \alpha_i L_i(x) R_i(x)\Big), \qquad x \in \mathfrak{I}_i. \tag{3.2}$$

   Заметим, что обобщение этого полинома на более общий случай, рассмотренный в §2.2.2, легко получается добавлением в приводимой ниже программе второго вложенного цикла `while` для перебора сомножителей в каждом одночлене.
2. Каждый одночлен составляется из двух алгоритмических операторов, выбираемых из различных семейств: $L_i \in \mathfrak{M}^0_{\text{Left}}$, $R_i \in \mathfrak{M}^0_{\text{Right}}$. Данные семейства реализуется алгоритмами `calc` абстрактных методов классификации `ClassLeft` и `ClassRight` соответственно. Разумеется, на практике абстрактные методы должны быть заменены какими-либо реальными методами, наиболее подходящими для конкретной решаемой задачи.
3. Главное отличие состоит в том, что для получения корректного алгоритма строится не глобальный, а локальный базис. С целью оптимизации алгоритмической суперпозиции используется итерационный процесс (1.1), (1.2).

Для простоты комбинированный критерий (1.4) не используется, и настройка алгоритмических операторов L_i, R_i, начиная с $i = 2$, производится исключительно на компенсацию ошибок предыдущих операторов. Для настройки полиномиальной корректирующей операции используется метод `LinearDiscr` поиска максимальной совместной подсистемы в системе линейных неравенств, см. §3.4.2. Процесс останавливается при достижении безошибочного распознавания обучающей выборки.

Постановка задачи в терминах ASDIEL. Имеется набор объектов `S` и набор признаков `F`. Заданы поднабор признаков `base` $\subseteq$ `F` и обучающая выборка `train` $\subseteq$ $\subseteq$ `S`. Подмассив `train|base` содержит исходные признаковые описания объектов обучения. Подмассив `train|F{goal}` содержит обучающие классификации всех объектов из `train`: `goal`$(x) = 0$ если x принадлежит первому классу и `goal`$(x) =$ $= 1$ — если второму. Для объектов вне обучающей выборки значения признака `goal`, вообще говоря, не определены. Требуется построить в наборе `F` признак `class`, совпадающий с `goal` на обучающей выборке и вычислимый на произвольном объекте из `S`.

Решение задачи. Первый шаг состоит в построении одночлена $L_1(x)R_1(x)$ и оценивании функционала качества. Сначала описывается и настраивается оператор L_1:

```
METHOD ClassLeft;
TUNE train|base, train|F{goal};
CALC S|base -> S|F{L1};
```

При настройке оператора R_1 необходимо учитывать, что решается уже не первая система неравенств (2.9), как в случае L_1, а вторая. В соответствии с данной формулой знак неравенства, и соответственно, значение целевого признака `goal`(x), изменяется на противоположный для всех объектов обучения x, у которых `L1`$(x) < 0$. На языке ASDIEL это реализуется с помощью функции `if`. Стандартная функция `if` проверяет значение первого аргумента, и если оно истинно, то выдаёт значение второго аргумента, в противном случае — значение третьего аргумента.

```
METHOD ClassRight;
TUNE train|base, train|F{if(L1>0, goal, not goal)};
CALC S|base -> S|F{R1};
```

Затем вычисляется первый одночлен (признак `B1`) и соответствующая ему классификация (признак `class1`). Качество полученного таким образом решающего правила оценивается путём подсчёта числа ошибок на обучающей выборке:

```
= F{B1=L1*R1, class1=(B1>0), B=B1};
Q = sum (train|F{abs(goal-class1)};
```

Следующие слагаемые $L_2(x)R_2(x)$, $L_3(x)R_3(x)$, и т.д. добавляются в суперпозицию до тех пор, пока значение функционала Q не станет равным нулю. После

добавления очередного одночлена производится перенастройка корректирующей операции. Каждая итерация состоит из трёх шагов:

1. определение и настройка оператора L_i с учётом уже построенной суперпозиции $B = \alpha_1 L_1 R_1 + \ldots + \alpha_{i-1} L_{i-1} R_{i-1}$;
2. определение и настройка оператора R_i с учётом того, что к данному моменту суперпозиция имеет вид $B + L_i$;
3. перенастройка корректирующей операции, в результате которой получается суперпозиция

$$B = \alpha_1 L_1 R_1 + \ldots + \alpha_i L_i R_i; \tag{3.3}$$

и вычисление функционала качества.

Корректирующая операция `Corrector` создаётся на первом проходе цикла (при `i=2`). На следующих проходах происходит обращение к тому же методу и его перенастройка.

```
i = 1;  while Q>0;  i = i + 1;
  METHOD ClassLeft;
  TUNE train|base, train|F{goal, -'B[i-1]'};
  CALC S|base, S|F{-'B[i-1]'} -> S|F{'L[i]'};
  METHOD ClassRight;
  TUNE train|base,
    train|F{if('L[i]'>0, goal, not goal), -'B[i-1]'/'L[i]'};
  CALC S|base, S|F{-'B[i-1]'/'L[i]'} -> S|F{'R[i]'};
  = F{'B[i]'='L[i]'*'R[i]', 'class[i]'=('B[i]'>0)};
  Q = sum (train|F{abs(goal-class1)};
  if (i=2);
    METHOD LinearDiscr NAMED Corrector;
  else;
    METHOD NAMED Corrector;
  end if;
  TUNE train|F{B::1..i}, train|F{goal};
  CALC S|F{B::1..i} -> S|F{B, class};
  Q = sum (train|F{abs(goal-class)};
end while;
```

На каждом шаге итерационного процесса создаются 4 признака: `'L[i]'` и `'R[i]'`, соответствующие паре операторов L_i и R_i; признак `B`, соответствующий текущей суперпозиции (3.3); и признак `class`, определяемый через пороговое решающее правило: $\texttt{class}(s) = \theta(B(s))$ для всех $s \in S$.

В завершение можно вывести таблицу с результатами настройки промежуточных суперпозиций, чтобы показать, как увеличивалось число правильных классификаций по мере наращивания суперпозиции:

```
print « train | F{class::1..i,goal};
```

4 Заключение

Алгебраический подход к проблеме распознавания, развиваемый школой академика РАН Ю.И.Журавлёва, основан на идее построения корректных алгоритмов с помощью корректирующих операций над несколькими некорректными (эвристическими) алгоритмами. При практической реализации этой идеи возникает проблема уменьшения сложности и повышения экстраполирующей способности получаемой алгоритмической суперпозиции. В данной работе рассмотрены два пути её решения.

Во-первых, сложность суперпозиции уменьшается при использовании методов оптимизации для настройки (выбора параметров) алгоритмических операторов и корректирующей операции. При настройке предложено учитывать не только исходную обучающую информацию, но и структуру суперпозиции. Данный принцип использован при рассмотрении семейств линейных, полиномиальных и монотонных корректирующих операций для задач классификации и восстановления регрессии. Во всех случаях продемонстрировано применение общего подхода, при котором оптимизационная задача настройки алгоритмического оператора с учётом его положения в суперпозиции сводится к стандартной задаче его отдельной настройки без учёта суперпозиции. Описанные методы непосредственно применимы на практике.

Дальнейшие работы в этом направлении связаны с рассмотрением других семейств корректирующих операций и/или других классов задач. Успешное применение упомянутого общего подхода даёт основание полагать, что и в новых случаях задача настройки алгоритмического оператора может быть сведена к стандартным.

Во-вторых, сложность суперпозиции уменьшается за счёт более тщательного подбора её структуры и привлечения всей имеющейся априорной информации о предметной области и решаемой задаче. Ввиду огромного разнообразия всевозможных прикладных задач не существует ни общего критерия оптимальности структуры суперпозиции, ни эффективных способов формализации произвольной априорной информации. Поэтому построение суперпозиции предлагается проводить в режиме вычислительных экспериментов, используя для этого специальное инструментальное средство — язык описания алгоритмических суперпозиций.

В настоящий момент ядро языка реализовано полностью. Дальнейшее его развитие пойдёт, по-видимому, по пути наращивания библиотеки методов.

Таким образом, в диссертации разработана технология решения прикладных задач распознавания, классификации и прогнозирования, основанная на использовании алгебраического подхода с привлечением методов оптимизации и средств описания настраиваемых алгоритмических суперпозиций.

Список иллюстраций

Список литературы

[1] Вапник В. Н. Восстановление зависимостей по эмпирическим данным. М. Наука. 1979.

[2] Василега М.Ю. Платоненко И.М. Рудаков К.В. О введении метрик на объектных пространствах для решения задач распознавания // Математические методы распознавания образов–VI: Тез. докл. М. 1993.

[3] Василега М.Ю. О евклидовых представлениях конечных точечных метрических конфигураций // Математические методы распознавания образов–VII: Тез. докл. М. 1995.

[4] Василенко В.А. Сплайн-функции: теория, алгоритмы, программы. Новосибирск. Наука. 1983.

[5] Васильев В.И. Распознающие системы. Справочник. Киев. Наукова думка. 1983.

[6] Рудаков К.В., Воронцов К.В. О методах оптимизации и монотонной коррекции в алгебраическом подходе к проблеме распознавания // ДАН. (статья находится в печати).

[7] Воронцов К.В. О проблемно-ориентированной оптимизации базисов задач распознавания // ЖВМ и МФ. 1998. Т. 38, № 5. С. 870–880.

[8] Воронцов К.В. Оптимизационные методы линейной и монотонной коррекции в алгебраическом подходе к проблеме распознавания // ЖВМ и МФ. (статья находится в печати).

[9] Воронцов К.В. Качество восстановления зависимостей по эмпирическим данным // Математические методы распознавания образов–VII: Тез. докл. М. 1995.

[10] Воронцов К.В. О синтезе проблемно-ориентированных базисов в задачах распознавания // Математические методы распознавания образов–VIII: Тез. докл. М. 1997.

[11] Дюкова Е.В. О сложности реализации некоторых процедур распознавания // ЖВМ и МФ. 1987. Т. 27, № 1. С. 114–127.

[12] Дюкова Е.В., Рязанов В.В. О решении прикладных задач алгоритмами распознавания, основанными на принципе голосования. М. ВЦ АН СССР. 1986. 26 с.

[13] Журавлев Ю.И. Экстремальные алгоритмы в математических моделях для задач распознавания и классификации // ДАН СССР. 1976. Т. 231, № 3. С. 532–535.

[14] Журавлёв Ю. И. Корректные алгебры над множеством некорректных (эвристических) алгоритмов. I–III. // Кибернетика. 1977. № 4. С. 14–21. 1977. № 6. С. 21–27. 1978. № 2. С. 35–43.

[15] Журавлёв Ю. И. Об алгебраическом подходе к решению задач распознавания или классификации. // Проблемы кибернетики. 1979. Вып. 33. С. 5–68.

[16] Журавлев Ю.И., Зенкин А.А., Зенкин А.И., Исаев И.В., Кольцов П.П., Ко-

четков Д.В., Рязанов В.В. Задачи распознавания или классификации со стандартной обучающей информацией // ЖВМ и МФ. 1980. Т. 20, № 5. С. 1294–1309.

[17] Журавлев Ю.И., Рудаков К.В. Об алгебраической коррекции процедур обработки (преобразования) информации // Проблемы прикладной математики и информатики. М. Наука. 1987. С. 187-198.

[18] Журавлев Ю.И., Сергиенко И.В., Артеменко В.И., Чернякова А.М. Вопросы применения результатов теории распознавания при автоматизированном выборе алгоритмов решения задач в пакетах программ // Кибернетика. 1986. № 3. С. 11–17.

[19] Журавлёв Ю.И., Гуревич И.Б. Распознавание образов и распознавание изображений. // Распознавание, классификация, прогноз. Выпуск 2. М. Наука. 1989. С. 5–72.

[20] Загоруйко Н.Г., Ёлкина В.Н., Лбов Г.С. Алгоритмы обнаружения эмпирических закономерностей. Новосибирск. Наука. 1985.

[21] Зуев Ю. А. Метод повышения надежности классификации при наличии нескольких классификаторов, основанный на принципе монотонности // ЖВМ и МФ. 1981. Т. 21, № 1. С. 157–167.

[22] Ивахненко А.Г., Юрачковский Ю.П. Моделирование сложных систем по экспериментальным данным. Москва. Радио и связь. 1987.

[23] Игнатов М.И., Певный А.В. Натуральные сплайны многих переменных. Ленинград. Наука. 1991.

[24] Казанцев В. С. Задачи классификации и их программное обеспечение. М. Наука. 1990.

[25] Катериночкина Н.Н. Поиск максимального верхнего нуля монотонной функции алгебры логики // ДАН СССР. 1975. Т. 224, № 3. С. 557-560.

[26] Кенделл М. Ранговые корреляции. М. Статистика. 1975.

[27] Кемени Дж., Снелл Дж. Кибернетическое моделирование. М. Сов. радио. 1972.

[28] Лоусон Ч., Хенсон Р. Численное решение задач метода наименьших квадратов. М. Наука. 1986.

[29] Мазуров В.Д., Казанцев В.С., Белецкий Н.Г., Кривоногов А.И., Смирнов А.И. Вопросы обоснования и применения комитетных алгоритмов распознавания // Распознавание, классификация, прогноз. Выпуск 1. М. Наука. 1989. С. 114–148.

[30] Мазуров В.Д. Метод комитетов в задачах оптимизации и классификации. М. Наука. 1990.

[31] Мальцев А.И. Алгебраические системы М. Наука. 1970.

[32] Матросов В. Л. Ёмкость алгебраических расширений модели алгоритмов вычисления оценок. // ЖВМиМФ. 1984.

[33] Матросов В.Л. Корректные алгебры ограниченной ёмкости над множествами

некорректных алгоритмов // ДАН СССР. 1980. Т. 253, № 1. С. 25-30.

[34] Растригин Л.А., Эренштейн Р.Х. Коллективные правила распознавания. М. Энергия. 1981. 244 с.

[35] Рудаков К.В. О некоторых классах алгоритмов распознавания (общие результаты). М. ВЦ АН СССР. 1980. 66 с.

[36] Рудаков К.В. О некоторых классах алгоритмов распознавания (параметрические модели). М. ВЦ АН СССР. 1981. 48 с.

[37] Рудаков К.В. Универсальные и локальные ограничения в проблеме коррекции эвристических алгоритмов // Кибернетика. 1987. № 2. С. 30–35.

[38] Рудаков К.В. Полнота и универсальные ограничения в проблеме коррекции эвристических алгоритмов классификации // Кибернетика. 1987. № 3. С. 106–109.

[39] Рудаков К.В. О симметрических и функциональных ограничениях для алгоритмов классификации // ДАН СССР. 1987.Т. 297, № 1. С. 43–46.

[40] Рудаков К.В. Об алгебраической теории универсальных и локальных ограничений для задач классификации // Распознавание, классификация, прогноз. Выпуск 1. М. Наука. 1989. С. 176–201.

[41] Рудаков К.В. Монотонные и унимодальные корректирующие операции для алгоритмов распознавания // Математические методы распознавания образов–VII: Тез. докл. М. 1995.

[42] Рязанов В.В. Комитетный синтез алгоритмов распознавания и классификации // ЖВМ и МФ. 1981. Т. 21, № 6. С. 1533–1543.

[43] Рязанов В.В. О построении оптимальных алгоритмов распознавания и таксономии (классификации) при решении прикладных задач // Распознавание, классификация, прогноз. Выпуск 1. М. Наука. 1989. С. 229–279.

[44] Рязанов В.В. Сенько О.В. О некоторых моделях голосования и методах их оптимизации // Распознавание, классификация, прогноз. Выпуск 3. М. Наука. 1992. С. 106–145.

[45] Тер-Крикоров А.М. Шабунин М.И. Курс математического анализа. М. Наука. 1988.

[46] Хардле В. Прикладная непараметрическая регрессия. М. Мир. 1993.

[47] Черников С.Н. Линейные неравенства. М. Наука. 1968.

Список основных обозначений

$\mathfrak{I}_i$ — множество начальных информаций, стр. 7
$\mathfrak{I}_f$ — множество финальных информаций, стр. 7
$\mathfrak{I}_e$ — пространство оценок, стр. 8
$I_q = \{x_k\}_{k=1}^q$ — обучающая последовательность начальных информаций, стр. 7
$\widetilde{I}_q = \{y_k\}_{k=1}^q$ — обучающая последовательность финальных информаций, стр. 7
q — длина обучающей последовательности, стр. 7
$\mathbb{Q}$ — множество индексов $\{1, \ldots, q\}$, стр. 23
Q — функционал качества алгоритмических операторов, стр. 10
$\mathfrak{M}^0$ — модель алгоритмических операторов, стр. 8
$\mathfrak{M}^1$ — семейство решающих правил, стр. 8
$\mathfrak{M} = \mathfrak{M}^1 \circ \mathfrak{M}^0$ — эвристическая информационная модель алгоритмов, стр. 8
$\mathfrak{F}$ — семейство корректирующих операций, стр. 8
$\mathfrak{F}_L$ — семейство линейных корректирующих операций, стр. 17
$\mathfrak{F}_P$ — семейство полиномиальных корректирующих операций, стр. 21
$\mathfrak{F}_M$ — семейство монотонных корректирующих операций, стр. 23
$\mathfrak{F}(\mathfrak{M})$ — $\mathfrak{F}$-расширение эвристической информационной модели $\mathfrak{M}$, стр. 9
$B_1, \ldots, B_p$ — базисный набор алгоритмических операторов, стр. 9
λ — параметр, регулирующий степень компромисса между настройкой на исходные прецеденты и компенсацией неточностей других операторов, стр. 13
a_k — вектор оценок операторов $B_1, \ldots, B_p$ на k-ом объекте обучающей выборки, стр. 23
$\mathbb{D}(B)$ — дефектная пара алгоритмического оператора B, стр. 24
$\mathbb{D}(B_1, \ldots, B_p)$ — дефект набора операторов $B_1, \ldots, B_p$, стр. 24
$\|$ — бинарное отношение несравнимости, стр. 23
$\prec$ — бинарное отношение на множестве объектов обучения, обобщающее естественное отношение порядка на векторах размерности p, стр. 26
t_{jk} — число дефектных троек с основанием (j, k), стр. 30
t^0_{jk} — число строго дефектных троек с основанием (j, k), стр. 30
M^1_k, M^0_k — верхняя и нижняя области монотонности вектора a_k, стр. 37
$r^1(a, a_k)$, $r^0(a, a_k)$ — расстояния от вектора a до верхней и нижней областей монотонности вектора a_k, стр. 37
$h^1(a)$, $h^0(a)$ — расстояния от вектора a до ближайшей верхней и ближайшей нижней областей монотонности, стр. 38
$[\![F \times S_1 \times \ldots \times S_{k-1}]\!]$ — массив размерности k, см. модель данных ASDIEL, стр. 59
$\theta(x)$ — функция Хевисайда, $\theta(x) = \begin{cases} 0, & x \leqslant 0; \\ 1, & x > 0. \end{cases}$
x_+ — операция срезки, $x_+ = \begin{cases} 0, & x \leqslant 0; \\ x, & x > 0. \end{cases}$
$\lceil x \rceil$ — функция «потолок» — минимальное целое, не меньшее x
$\lfloor x \rfloor$ — функция «пол» — максимальное целое, не большее x