Московский Государственный Университет
им. М. В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических методов прогнозирования

Дипломная работа

Генетические алгоритмы синтеза локальных базисов в алгебраическом подходе к проблеме распознавания

студента 517 группы
Каневского Д. Ю.

Научные руководители:
чл.-корр. РАН, д.ф.-м.н. Рудаков К.В.
к.ф.-м.н. Воронцов К.В.

Москва, 2005г

Содержание

Введение

Методы классификации, прогнозирования, распознавания образов являются важной частью интеллектуально-информационного анализа в рамках широкого круга прикладных задач. Использование накопленной информации о различных объектах и явлениях предполагает определение зависимости между начальными и финальными информациями, что и формулируется в виде задач классификации и прогнозирования. Начальная информация может быть неточной и неполной, предметная область недостаточно формализованной, а вид зависимости слишком сложным и неявным для построения адекватной модели нужной зависимости. В таких случаях используется т. н. *обучение по прецедентам*. В рамках данного подхода анализируется множество пар вида (<начальная информация>, <финальная информация>) и строится алгоритм, корректно воспроизводящий заданное соответствие на имеющемся конечном множестве объектов. Множество прецедентов называют *обучающей выборкой*, а начальные информации, описывающие отдельные прецеденты, — *объектами обучения*.

Так как информация о предметной области отсутствует, выбор зависимости производится из некоторого эвристического семейства алгоритмов. Такое семейство является в некотором смысле универсальным, поскольку оно должно быть применимо для широкого класса задач. Это делает такие семейства сложными, из-за чего поиск наилучшего для данной задачи алгоритма может не увенчаться успехом по причине отсутствия эффективных численных методов. В то же время более «бедные» модели могут не содержать корректного алгоритма.

Стремление к устранению указанных недостатков эвристических моделей привело к возникновению алгебраического подхода [9]. Исходная модель алгоритмов расширяется путем введения корректирующих операций, позволяющих строить зависимость в виде суперпозиции набора вообще говоря некорректных алгоритмов и объединяющего их корректора. При этом к алгоритму предъявляются следующие требования: во-первых, он не должен допускать ошибок на прецедентах, и во-вторых — удовлетворять некоторым дополнительным ограничениям, отражающим свойства и особенности предметной области [13, 14].

В рамках алгебраического подхода было доказана возможность построения корректных алгоритмов для широкого класса *регулярных* задач [12, 17, 15]. Алгоритмические суперпозиции, использованные для теоретических исследований, выбирались из чрезвычайно обширного семейства, что позволяло выписать их в явном виде, не решая задачи оптимизации. Удобные для доказательств, они оказались слишком «громоздкими» и не применимыми на практике [2].

Для адаптации методов алгебраического подхода к практическим задачам было предложено строить алгоритмические композиции, названные *локальными базисами* [1, 2, 3], ориентированные на решение конкретной проблемы и получаемые в результате решения последовательности оптимизационных задач. Получаемые таким образом суперпозиции имеют существенно меньшую, по сравнению с «конструктивными», мощность и обладают значительно лучшей обобщающей способностью.

Среди других популярных методов построения алгоритмических композиций преобладает использование линейных корректирующих операций — взвешенное или простое усреднение ответов, полученных базовыми алгоритмами. Приведем несколько наиболее распространенных методов составления ансамблей классификаторов.

Метод баггинга (bagging) был предложен Л. Брейманом в 1996 году [28, 29, 30]. Основная его идея заключается в том, чтобы целенаправленно увеличивать различия между базовыми алгоритмами, делая их, по возможности, более независимыми. Один из способов достижения этой цели — обучать базовые алгоритмы не по всей выборке, а по различным её частям. Выделение подмножеств объектов производится, как правило, случайным образом. Построенные алгоритмы объединяются в композицию с помощью простого голосования.

В 1995 году американские ученые Р. Френд и И. Шапир предложили метод бустинга (boosting) [42, 44, 73], который быстро завоевал популярность благодаря своей простоте и высокой эффективности. В бустинге производится взвешенное голосование, базовые алгоритмы строятся последовательно, а процесс увеличения различий между ними управляется детерминированным образом. Перед настройкой каждого следующего алгоритма веса обучающих объектов пересчитываются — больший вес получают те объекты, на которых чаще ошибались все предыдущие алгоритмы. В ряде экспериментов бустинг демонстрировал практически неограниченное улучшение качества обучения при наращивании числа алгоритмов в композиции. Более того, качество на тестовой выборке часто продолжало улучшаться даже после достижения безошибочного распознавания обучающей выборки [45].

Также существует класс методов, добивающихся различий между алгоритмами путем выбора подмножеств признаков. Говорят, что задача решается в подпространствах пространства объектов обучения. Выбор подпространств чаще всего осуществляется стохастически [51], после чего, возможно, корректируется [80]. Также существуют методики объединения выбора подмножеств признаков с бустингом и баггингом [94, 95].

Все эти методы являются, по сути дела, надстройками (wrappers) над эвристическими моделями. Обучение каждого базового алгоритма производится стандартным для соответствующей модели методом. Перед обучением каждого алгоритма исходные данные несколько модифицируются, например, в бустинге меняются веса обучающих объектов. Общим достоинством этих методов является возможность задействовать богатый арсенал готовых процедур для обучения базовых алгоритмов.

Кроме общих достоинств имеются и общие недостатки. В частности, жадная стратегия последовательного наращивания набора базовых алгоритмов приводит к тому, что локально оптимальный выбор каждого базисного алгоритма не является глобально оптимальным. Роль алгоритма в композиции ограничивается компенсацией ошибок предыдущих (построенных раньше него) алгоритмов, при этом никак не учитываются следующие алгоритмы. В результате увеличивается сложность композиции и, возможно, ухудшается её обобщающая способность. В работах [48, 69, 68] показано, что бустинг начинает переобучаться, если в композицию включено слишком много базовых алгоритмов — порядка 10^3. Кроме того, бустинг и баггинг оказываются бесполезными для *стабильных* алгоритмов, мало подверженных изменению при малых изменениях обучающей выборки. К данному классу относятся такие семейства, как правило Байеса, методы k ближайших соседей, потенциальных функций, линейные разделители.

Данная работа предлагает новый подход к построению локальных базисов, основанный на специального вида эволюционном алгоритме. Следующий раздел обосновывает использование такого подхода и описывает основные результаты работы.

Актуальность и новизна

Итак, во всех вышеперечисленных методах построения алгоритмических композиций используется локальная оптимизация. Она производится потому, что задача одновременной оптимизации всех базовых алгоритмов и корректирующей операции чрезвычайно сложна. Она является многопараметрической, многоэкстремальной, и в общем случае не сводится к готовым процедурам обучения базовых алгоритмов.

При практическом решении сложных, плохо формализованных, многопараметрических задач неожиданно хорошие решения могут находить эволюционные алгоритмы. Одними из самых крупных и распространенных среди них являются генетические алгоритмы, предложенные Холландом в 1975 году [53]. Алгоритмы этого класса основаны на дарвиновских эволюционных принципах, а именно обеспечения генетического разнообразия и естественного отбора. Так, в рамках генетического алгоритма формируется популяция потенциальных решений задачи, называемые индивидами. В пространстве индивидов определяются бинарная операция рекомбинации и унарная — мутации (возможны и другие «генетические» операции). Далее запускается итерационный процесс, на каждом шаге которого текущая популяция подвергается рекомбинациям и мутациям, после чего производится селекция лучших вариантов решения для следующего поколения.

Генетические алгоритмы использовались для широкого круга задач, в том числе и в распознавании [41]. Так, в [91] генетический алгоритм используется для отбора наиболее информативных признаков, в [40] — для синтеза логических привил классификации. Известны также методы построения решающих деревьев, когда рекомбинация реализуется как обмен поддеревьями у пары решающих деревьев.

В данной работе используется особый вид генетического алгоритма, называемый *кооперативной коэволюцией* (Cooperative Coevolution) [64, 63]. Данная модель использует несколько популяций, каждая из которых соответствует решению части задачи или ее подзадаче. Популяции взаимодействуют друг с другом по принципу симбиоза, выбирая алгоритмы, наилучшим образом кооперирующих с остальными. Кооперативная коэволюция уже использовалась в распознавании. В [65] она применяется для построения нейронных сетей, в [26] — для формирования признакового описания объектов обучения, в [81] — для построения набора «антител» по принципу действия иммунитета человека для решения задачи рубрикации.

Во всех вышеописанных примерах эволюционный алгоритм обязан «знать» внутреннее устройство классификатора. Фактически, он используется как метод настройки конкретной модели алгоритмов. В данной работе предлагается новый подход, при котором генетический алгоритм выступает в роли надстройки над стандартными методами обучения базовых алгоритмов, не вмешиваясь в их внутреннюю структуру. В этом варианте генетический алгоритм более всего напоминает баггинг. Но в отличие от баггинга, он выполняет не последовательную, а одновременную оптимизацию базовых алгоритмов и корректирующей операции. Для этого реализуется модель кооперативной коэволюции, каждая популяция которой соответствует одному алгоритму композиции, а коррекция производится только при оценке приспособленности индивидов. В процессе эволюции каждый базовый алгоритм получает возможность специализироваться в определённой, только ему присущей роли.

В рамках данного подхода разработан и детально описан в настоящей работе метод построения локальных базисов на основе кооперативной коэволюции. Данный

метод, в отличие от проблемно-ориентированных методов алгебраического подхода, не является жадным и позволяет строить композиции минимальной мощности. В отличие от бустинга, баггинга и настроек по подмножествам признаков, метод позволяет строить композиции с произвольными корректирующими операциями и использовать различные критерии оптимизации, как, например, отступы объектов. Кооперативная коэволюция впервые используется для построения алгоритмических композиций с произвольными базовыми семействами алгоритмов и корректирующих операций.

Метод реализован для линейных корректирующих операций и протестирован с правилом Байеса в качестве базового семейства алгоритмов на ряде модельных и реальных задач. Результаты экспериментов доказывают перспективность и эффективность предлагаемого подхода.

Краткое содержание работы

Работа состоит из пяти разделов.

В первом разделе приводится постановка задачи обучения по прецедентам, кратко описываются оптимизационный и алгебраический подходы к ее решению, вводятся понятия глобального и локального базисов, обсуждаются их отличия.

Второй раздел содержит краткий обзор существующих методов построения алгоритмических композиций. Описаны проблемно-ориентированные методы алгебраического подхода, бустинг, баггинг и настройки по подмножествам признаков. Приводятся соответствующие алгоритмы, достоинства и недостатки перечисленных методов, обсуждаются причины эффективности различных подходов.

Третий раздел посвящен обзору эволюционных алгоритмов. В нем описаны общие принципы построения, достоинства и недостатки, сферы применения эволюционных алгоритмов. Приводится подробное изложение двух наиболее крупных подсемейств — генетических алгоритмов и эволюционных стратегий. Также рассмотрены модели алгоритмов, использующие несколько популяций. Особое внимание уделено кооперативной коэволюции.

Четвертый раздел представляет собой детальное описание предлагаемого коэволюционного метода синтеза локальных базисов. Рассмотрен общий принцип построения алгоритмических композиций на базе кооперативной коэволюции, описаны параметры соответствующей модели алгоритмов и варианты их задания, приведены достоинства и недостатки предложенного метода. Далее, приводится описание реализации данного метода для случая линейных корректирующих операций. Раздел завершается экспериментальным анализом реализованного метода с правилом Байеса в качестве базового семейства алгоритмов. Ряд экспериментов на модельных и реальных задачах (из репозитория UCI [27]) демонстрирует возможности предложенного метода.

Пятый раздел содержит совокупность выводов, сделанных на основе приведенного материала, и предлагает большой набор направлений дальнейших исследований.

1 Основные понятия

1.1 Задача обучения по прецедентам

В самой общей постановке задача синтеза алгоритмов преобразования информации состоит в следующем [9]. 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 $I_q = \{x_i\}_{i=1}^q$ элементов множества $\mathfrak{I}_i$ и последовательность $\widetilde{I}_q = \{y_i\}_{i=1}^q$ элементов множества $\mathfrak{I}_f$. Искомый алгоритм A должен точно или приближённо удовлетворять системе из q равенств

$$A(x_i) = y_i, \quad i = 1, \ldots, q,$$

или сокращенно — $A(I_q) = \widetilde{I}_q$. Ограничения такого типа называются *локальными*. 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 $\mathfrak{I}_f$. Такие ограничения названы *универсальными*. Алгоритм, удовлетворяющий локальным и универсальным ограничениям, называют *корректным*. Таким образом, рассматриваемая задача обучения по прецедентам определяется пятёркой $Z = \langle \mathfrak{I}_i, \mathfrak{I}_f, \mathfrak{M}^u, I_q, \widetilde{I}_q \rangle$.

Принципиальное различие между локальными и универсальными 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 и т. д. На практике универсальные ограничения учитываются на этапе построения параметрического семейства алгоритмов, а локальные — при последующей настройке параметров алгоритма на заданные прецеденты.

1.2 Оптимизационный и алгебраический подходы к проблеме распознавания

Оптимизационный подход к решению задачи обучения по прецедентам заключается в выборе эвристической информационной модели алгоритмов $\mathfrak{M} \subseteq \mathfrak{M}^u$, функционала качества $Q : \mathfrak{M}^u \to \mathbb{R}$ и поиску алгоритма A^*, на котором достигается минимум $Q(A)$. Точнее, в оптимизационном подходе к проблеме распознавания производятся следующие действия:

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).

Однако 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 [6, 7, 8]. Основная его идея состоит в том, чтобы расширить исходную модель алгоритмов с помощью корректирующих операций. В рамках алгебраического подхода производится построение некоторого количества алгоритмов и корректирующей операции над ними, причем подбор параметров осуществляется с целью гарантированного удовлетворения локальных ограничений. Универсальные ограничения выполняются по построению, так как $\mathfrak{M} \subseteq \mathfrak{M}^u$.

Для реализации этой идеи в алгебраическом подходе к проблеме распознавания наряду с множествами $\mathfrak{I}_i$ и $\mathfrak{I}_f$ вводится пространство оценок $\mathfrak{I}_e$. Затем выбираются три семейства отображений:

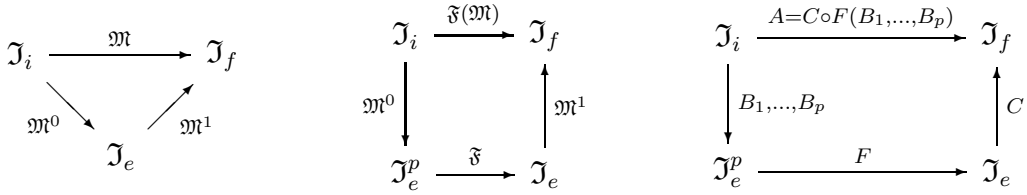
— модель алгоритмических операторов $\mathfrak{M}^0 \subseteq \{B : \mathfrak{I}_i \to \mathfrak{I}_e\}$,

— семейство решающих правил $\mathfrak{M}^1 \subseteq \bigcup_{p=0}^{\infty} \{C : \mathfrak{I}_e^p \to \mathfrak{I}_f\}$,

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

Искомый алгоритм A строится в виде суперпозиции $C(F(B_1, \ldots, B_p))$ алгоритмических операторов $B_1, \ldots, B_p$ из $\mathfrak{M}^0$, корректирующей операции F из $\mathfrak{F}$ и решающего правила C из $\mathfrak{M}^1$. Всевозможные алгоритмы такого вида образуют так называемое $\mathfrak{F}$-расширение модели $\mathfrak{M}$, обозначаемое $\mathfrak{F}(\mathfrak{M})$, в рамках которого и ведется 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 [10, 11, 13, 12, 14].

В работах по алгебраическому подходу была доказана возможность построения корректного алгоритма для широкого класса задач, называемых регулярными, при условии наличия «богатых» (обладающих свойством полноты) семейств $\mathfrak{M}^0$ и $\mathfrak{F}$. Доказательство соответствующих теорем являлось конструктивным, искомая суперпозиция строилась непосредственно без применения оптимизации. Такие конструкции были удобны для теоретических рассмотрений, но не предназначались для непосредственного применения на практике. Алгоритмы, реализованные по явным формулам,



(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.*

получались слишком громоздкими и не давали надежных 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 [2]. Стремление к получению суперпозиций высокого качества привело к появлению проблемно-ориентированных методов алгебраического подхода [16, 1, 3, 4], рассмотренных в разделе 2.1. Их отличительной особенностью является решение набора оптимизационных задач для построения набора алгоритмических операторов, обеспечивающих существование корректного алгоритма для конкретной поставленной задачи. Реализация данного подхода связана с понятием локального базиса.

1.3 Глобальные и локальные базисы

Переход от теоретических исследований к практике потребовал введения новых понятий, отвечающих новым задачам. Приведем два определения.

Определение 1. Конечное множество алгоритмических операторов $B_1, \ldots, B_p$ называется *глобальным базисом*, если для любой финальной информации $\{z_i\}_{i=1}^q$ найдется корректирующая операция $F \in \mathfrak{F}$ и решающее правило $C \in \mathfrak{M}^1$ такие, что алгоритм A является корректным на обучающей выборке $\{x_i, z_i\}_{i=1}^q$.

Определение 2. Конечное множество алгоритмических операторов $B_1, \ldots, B_p$ называется *локальным базисом*, если найдется корректирующая операция $F \in \mathfrak{F}$ и решающее правило $C \in \mathfrak{M}^1$ такие, что алгоритм A является корректным на обучающей выборке $\{x_i, y_i\}_{i=1}^q$.

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

Глобальные базисы соответствуют задачам теоретического характера, они использовались для конструктивного доказательства теорем существования. Требования к локальному базису существенно слабее, нежели к глобальному, что позволяет строить наборы операторов существенно меньшей мощности. При этом локальный базис ориентирован на решение конкретной задачи, поэтому качество классификации оказывается существенно выше.

На практике задача построения локального базиса сводится к задаче минимизации функционала качества $Q(A)$, то есть превращается в оптимизационную задачу. Корректность получаемого алгоритма не всегда достижима и не всегда является основной целью, поэтому от задачи построения локального базиса «в чистом виде»

перейдем к задаче оптимизации Q по набору алгоритмических операторов и корректирующей операции.

2 Методы построения композиций алгоритмов классификации

2.1 Проблемно-ориентированные методы алгебраического подхода

Пусть задан функционал качества алгоритмических операторов Q, принимающий нулевое значение $Q(B) = 0$ тогда и только тогда, когда существует корректный алгоритм вида $A = C \circ B$. Рассмотрим задачу минимизации функционала $Q(F(B_1, \ldots, B_p))$ по алгоритмическим операторам $B_1, \ldots, B_p$ из $\mathfrak{M}^0$ и корректирующей операции F из $\mathfrak{F}$.

Общий подход к решению данной задачи заключается в организации следующего итерационного процесса. На первом шаге выбирается оператор B_1 из модели $\mathfrak{M}^0$ путем минимизации функционала $Q(B)$. Следующие базисные операторы B_2, $B_3, \ldots$ строятся по очереди, причем после добавления очередного оператора проводится повторная оптимизация ранее построенных операторов и корректирующей операции. При этом на каждом шаге решается одна из двух задач:

$$B_r^* = \arg\min_{B_r \in \mathfrak{M}^0} Q(F(B_1, \ldots, B_r, \ldots, B_p)), \quad 1 \leqslant r \leqslant p, \tag{2.1}$$

$$F^* = \arg\min_{F \in \mathfrak{F}} Q(F(B_1, \ldots, B_p)). \tag{2.2}$$

Описанный итерационный 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. Для минимизации функционала (2.1) предлагается использовать методы, изначально предназначенные для решения более простой задачи

$$B_r^* = \arg\min_{B_r \in \mathfrak{M}^0} Q(B_r). \tag{2.3}$$

Переход от (2.3) к (2.1) осуществляется по-разному в зависимости от $\mathfrak{F}$, $\mathfrak{I}_e$ и $\mathfrak{I}_f$, общей идеей является возможность использования существующих численных методов оптимизации функционала Q для решения задачи 2.1. В [2] детально описан такой переход и указаны методы решения соответствующих оптимизационных задач для семейств линейных, полиномиальных, а также монотонных корректирующих операций.

2.2 Бустинг

Семейство алгоритмов бустинга [42, 45, 74, 46, 73] представляет собой набор методов построения композиции классификаторов, действующих по принципу взвешенного голосования. Для обеспечения разнообразия и специализации алгоритмов, входящих в линейную комбинацию, в бустинге используются веса объектов из обучающей выборки.Предложенный в [45] алгоритм AdaBoost итерационно добавляет в композицию по одному алгоритму, настроенному на «взвешенной» обучающей выборке. Веса объектов изменяются на каждом шаге образом, что больший вес получают объекты,

Алгоритм 2.1 Алгоритм бустинга AdaBoost

Вход:

$X^q = \{x_i, y_i\}$, $i = \overline{1 \dots q}$ — обучающая выборка, $y_i \in Y = \{1 \dots l\}$;

$\mu(X^q, \bar{w})$ — процедура обучения «слабого» классификатора на «взвешенной» выборке;

T — число итераций;

Выход:

$A(x)$ — «сильный» классификатор;

для всех $i = 1 \dots q$

$w_1(i) = 1/q$

для всех $t = 1 \dots T$

Настроить «слабый» классификатор $a_t(x) = \mu(X^q, \bar{w}_t)$

Рассчитать ошибку $a(x)$: $\varepsilon_t = \sum\limits_{i:\ a_t(x_i) \neq y_i} w_t(i)$

Вычислить $\beta_t = \varepsilon_t/(1 - \varepsilon_t)$

для всех $i = 1 \dots q$

$$w_{t+1}(i) = w_t(i) \times \begin{cases} \beta_t, a_t(x_i) = y_i; \\ 1, \text{иначе}; \end{cases}$$

Нормализовать $\bar{w}_{t+1}$: $\sum_{i=1}^{q} w_{t+1}(i) = 1$

Сильный классификатор $A(x) = \arg\max\limits_{y \in Y} \sum\limits_{t:\ a_t(x)=y} log\frac{1}{\beta_t}$

неверно классифицированные предыдущими построенными алгоритмами (2.1). Веса алгоритмов также зависят от количества допущенных ими ошибок.

Бустинг позволяет заметно улучшать качество классификации, повышая его даже после достижения отсутствия ошибок на обучающей выборке. Переобучение начинает сказываться только на композициях очень большой мощности (порядка 10^3 и выше). Также экспериментально показано, что бустинг направлен на снижение как *смещения* (bias), так и *вариации* (variance) [30, 25], вместе с теоретической минимальной (байесовской) ошибкой составляющими ошибку генерализации алгоритмов распознавания. Феномен бустинга объясняется максимизацией *отступов* объектов, являющихся (упрощенно) расстоянием от объекта до границы классов [75, 48, 74]. Однако бустинг не всегда повышает качество распознавания,а для некоторых алгоритмов и задач может даже ухудшить его [58, 25]. Считается, что бустинг хорошо повышает качество распознавания *нестабильных* алгоритмов. Свойство стабильности (stability) означает, что небольшие вариации обучающей выборки приводят к незначительным изменениям получаемого при настройке алгоритма [5]. Стабильный алгоритм обладает малой вариацией, что определяет неэффективность алгоритмов, направленных в первую очередь на ее уменьшение. Высокой стабильностью обладают, к примеру, линейные решающие правила, правило Байеса, методы ближайших соседей и потенциальных функций [5, 78, 55]. Бустинг также обладает свойством стабильности. Нестабильными являются решающие деревья и нейронные сети, к которым наиболее эффективно применяется бустинг [58, 34].

2.3 Баггинг

Баггинг (bagging, акроним от bootstrap aggregating) [28, 30] также является методом построения линейной комбинации классификаторов. Однако, в отличие от бустинга, каждый из алгоритмов композиции настраивается независимо от остальных, на «своем» подмножестве объектов из обучающей выборки. Для формирования таких подмножеств используется следующая процедура *бутстреппинга* (bootstrapping): из множества объектов X^q обучающей выборки формируются случайные подмножества X^b, которые и используется для настройки алгоритма. Выбор объектов в X^b обычно осуществляется случайным образом, как правило, с повторами. Далее настроенные на таких *бутстреп-выборках* алгоритмы объединяются (aggregate) на основе простого или взвешенного голосования. На (2.2) представлен предложенный в [28] алгоритм, использующий простое большинство для определения класса объекта.

Алгоритм 2.2 Алгоритм баггинга

Вход:

$X = (x_1, \ldots, x_q)$ — множество объектов в обучающей выборке;
$Y = 1, \ldots, l$ — номера классов (возможные значения классификаторов);
$\mu(X)$ — процедура обучения «слабого» классификатора на выборке X;
T — число итераций;

Выход:

$A(x)$ — «сильный» классификатор;
для всех $t = 1 \ldots T$
Получить бутстреп-выборку $X^b = (x_1^b, \ldots, x_q^b)$
Настроить слабый классификатор $a_t(x) = \mu(X^b)$

$A(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} [a_t(x) = y]$

Баггинг, в отличие от бустинга, направлен почти исключительно на снижение вариации [30, 25]. Это обуславливает его малую эффективность применительно к стабильным алгоритмам. Но, в отличие от бустинга, баггинг не ухудшает качество распознавания по сравнению с одним классификатором [25]. Также баггинг может быть эффективнее бустинга для задач с *лжесвидетелями* (outliers) — объектами, способными «дезориентировать» обучающийся алгоритм, приводя его к неверному восстановлению зависимости и неизбежно низкому качеству распознавания. При баггинге лжесвидетели не попадают в часть бутстреп-выборок, что способствует более верному обучению соответствующих алгоритмов. Бустинг же, напротив, увеличивает веса лжесвидетелей, поскольку чаще всего они являются трудными для классификации объектами. В [35] выдвигаются и эмпирически проверяются две гипотезы относительно причин успеха баггинга, приводимые в терминах байесовской теории. Первая гипотеза состоит в том, что он позволяет более верно оценить апостериорную байесовскую вероятность $P(i \mid x)$ класса i при условии предъявления объекта x, максимизация которой дает теоретически оптимальную оценку качества распознавания. Вторая гипотеза гласит, что баггинг способствует уточнению распределения обучающей выборки, то есть априорных вероятностей вида $P(x \mid i)$, и/или расширяет исходное семейство алгоритмов. Все проведенные автором эксперименты опровергали первую гипотезу и подтверждали вторую.

2.4 Настройки по подмножествам признаков

Еще одним способом получения набора разнообразных алгоритмов из одной обучающей выборки является выбор подмножеств признаков, описывающих объекты. Иначе говоря, задача восстановления зависимости решается в некотором подпространстве пространства объектов. Такой подход может быть эффективен в случае наличия малоинформативных признаков, или «признаков-лжесвидетелей». Также методы отбора признаков могут показывать лучшие результаты при малом объеме обучающей выборки (порядка размерности пространства объектов), поскольку размерность пространства уменьшается, соответственно увеличивая соотношение между объемом выборки и числом признаков [77]. Существуют различные методы отбора признаков с последующей агрегацией, среди которых наиболее часто используются стохастические.

Метод случайных подпространств (Random Subspace Method – RSM) [51]. Пусть каждый объект исходной обучающей выборки X^q представляет собой вектор длины s. Выбрав $r < s$ признаков, можно получить выборку X^r r-мерных векторов, то есть рассмотреть задачу классификации в подпространстве размерности r исходного s-мерного пространства. Метод случайных подпространств заключается в случайном выборе поднаборов набора признаков, настройке алгоритмов на обучающих выборках «в подпространствах» и объединении путем процедуры голосования. Схема метода случайных подпространств представлена в (2.3).

Алгоритм 2.3 Метод случайных подпространств

Вход:

$X = (x_1, \ldots, x_q)$ — множество объектов в обучающей выборке;

$Y = 1, \ldots, l$ — номера классов (возможные значения классификаторов);

$\mu(X^r)$ — процедура обучения «слабого» классификатора на выборке размерности r;

T — число итераций;

Выход:

$A(x)$ — «сильный» классификатор;

$b = 0$

для всех $b = 1 \ldots T$

Случайным образом выбрать r признаков и получить выборку X_b^r размерности r

Настроить слабый классификатор $A^b(x) = \mu(X_b^r)$

$A(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} [a_t(x) = y]$

Метод случайных подпространств способен «справляться» со стабильностью алгоритмов и показывать лучшие, нежели баггинг и бустинг, результаты на выборках малого объема [77].

Существуют также другие методы настройки по подмножествам признаков, так или иначе использующие стохастические приемы. Например, в [80] описан переборный метод подбора оптимального набора признаков для построения композиции байесовских алгоритмов классификации, в котором перебор начинается со случайного

выбора подмножества признаков для каждого алгоритма. В [93] производится попытка объединения бустинга и стохастического выбора признаков для построения композиций решающих деревьев, а в [94, 95] успешно объединяются стохастический выбор признаков, бустинг и баггинг.

3 Эволюционные алгоритмы

Основанные на идеях, почерпнутых из теории происхождения видов, алгоритмы этого семейства имеют в основе дарвиновские эволюционные принципы в наиболее общей форме [83], [79]. В рамках эволюционного алгоритма формируется *популяция индивидов*, каждый из которых представляет собой потенциальное решение поставленной задачи. Для таких индивидов определяются *генетические операции*, обычно это бинарная операция рекомбинации и/или унарная — мутации. Эволюция представляет собой итерационный процесс, каждая итерация которого называется *поколением*. В каждом поколении на основе текущей популяции «родителей» с помощью генетических операций формируется новая популяция «детей», из которой селекторный механизм выбирает наилучших индивидов для порождения следующего поколения. Селекция производится на основе *функции соответствия* (приспособленности, адаптивности) (fitness-function), определяемой исходя из поставленной задачи. Функция соответствия определена на множестве индивидов популяции и принимает действительные значения, трактуемые как оценка адаптивности (fitness) данного индивида. Адаптивность индивида в терминах решаемой задачи имеет смысл оценки качества решения задачи, определяемого данным индивидом. Таким образом генетические операции обеспечивают разнообразие индивидов, а селекция направляет эволюционный процесс в сторону наилучшего решения рассматриваемой задачи.

Эволюционные алгоритмы успешно применяются в решении сложных комбинаторных, плохо поставленных, нестандартных задач, таких, как мультимодальная и многокритериальная оптимизация, машинное обучение и распознавание. Они являются инструментом синтеза сложных структур, таких, как нейронные сети, программы на формальных языках и т.д.

К достоинствам эволюционного подхода следует отнести его применимость к чрезвычайно широкому классу задач, так как практически отсутствуют ограничения, накладываемые на входные данные, область поиска решения и критерий адаптивности, что делает его практически незаменимым в условиях, когда не применимы «стандартные» методы оптимизации. Также признано, что эволюционный алгоритм часто способен найти адекватное решение тогда, когда другие методы либо не дают решения надлежащего качества, либо требуют слишком больших затрат вычислительных ресурсов.

Наряду с этими достоинствами, существует один чрезвычайно важный недостаток: динамика эволюционных алгоритмов настолько сложна, что не существует адекватной математической модели, позволяющей формально описывать эволюционный процесс или доказать его сходимость. Подбор значений параметров самого алгоритма приходится производить эмпирическим путем, то есть методом проб и ошибок.

Существуют два основных подхода к формальному описанию эволюционной динамики: на макроуровне – в терминах теории игр [37], на микроуровне – с помощью цепей Маркова [52], [90]. Предприняты также попытки алгебраического рассмотрения задачи [67]. В рамках самого эволюционного подхода проблема частично решается с помощью т.н. *самоадаптирующихся* (self-adaptive) эволюционных алгоритмов [18], [20]. Их отличительной особенностью является внесение в код индивида параметров самого алгоритма. Набор параметров алгоритма «эволюционирует» наряду с решением задачи, что, согласно экспериментальным исследованиям, ускоряет ра-

боту алгоритма и позволяет быстрее находить лучшие решения. Однако адаптивное получение параметров усложняет алгоритм и также не дает гарантий сходимости.

Наиболее распространенными представителями семейства являются *генетические алгоритмы* (genetic algorithms) [33] и *эволюционные стратегии* (evolutionary strategies) [23]. Описание каждого из них представлено ниже в одноименных подразделах. Далее рассмотрены эволюционные алгоритмы, содержащие несколько взаимодействующих популяций - *коэволюционные алгоритмы*. Особое внимание уделено симбиотическому взаимодействию видов, описанному в рамках модели *кооперативной коэволюции* [63], послужившей основой для предлагаемого в работе алгоритма синтеза локальных базисов.

3.1 Генетические алгоритмы

Генетические алгоритмы являются наиболее распространенными и наиболее изученными эволюционными алгоритмами. Их отличительной чертой является представление индивидов в виде бинарных векторов, определенным образом кодирующих решение поставленной задачи. Генетический алгоритм обычно содержит две генетические операции - скрещивания (рекомбинации, кроссинговера) и мутации. Формально «обычный» генетический алгоритм может быть представлен следующим набором из 9 параметров [21]:

$$GA = (P^0, n, n_i, m, S, C, M, f, t)$$

где

$P^0 = (a_1, \ldots, a_n) \in I^n$, $I = \{0,1\}^m$ — начальная популяция;

n — размер популяции;

n_i — размер промежуточной популяции, из которой производится селекция наилучших потомков;

m — длина бинарного представления индивида;

$S: \quad I^{n_i} \to I^n$ — оператор селекции;

$C: \quad I \times I \to I$ — оператор скрещивания;

$M: \quad I \to I$ — оператор мутации;

$f: \quad I \to \mathbb{R}$ — функция соответствия;

$t: \quad I^n \to \{0,1\}$ — критерий останова;

Алгоритм выполняется следующим образом: на первом этапе создается начальная популяция P^0 двоичных векторов длины m, состоящая из n индивидов. Далее на каждой итерации алгоритма из текущей популяции P^k с помощью операторов скрещивания и мутации формируется промежуточная (intermediate) популяция P^{ik} размера n_i, из которой оператор селекции S отбирает n наиболее приспособленных для

«размножения» на следующей итерации. Приспособленность, или адаптивность, индивида, определяется функцией соответствия f, служащей связующим звеном между популяцией и предметной областью. Индикатором останова алгоритма служит критерий t.

В силу упомянутых выше причин существует множество способов задания каждого из перечисленных параметров, любой из которых не может быть, вообще говоря, признан хуже или лучше другого. Ниже описаны наиболее распространенные варианты задания операторов селекции, скрещивания, мутации, критерия останова.

Селекторный механизм. Приведем три варианта селекции [49], [22]:

Адаптивно-пропорциональная селекция.

Пусть $P^{ik} = (a_1^k, \ldots, a_{ni}^k)$ — промежуточная популяция на шаге k, $f(a_i^k)$ — приспособленность индивида a_i^k. Определяется вероятностное распределение $(p_1, \ldots, p_n)$, где

$$p_i = \frac{f(a_i^k)}{\sum_{j=1}^{n} f(a_j^k)}$$

Выбор индивидов из популяции осуществляется путем генерации случайной подвыборки с возвращениями в соответствии с данным распределением. Выбранный элемент помещается в популяцию следующего поколения, но не удаляется из текущей популяции. Процесс выбора повторяется n раз, то есть пока не будет набрана новая популяция из n индивидов. Более адаптивные индивиды при такой селекции получают шанс получить несколько своих копий в следующем поколении, менее адаптивные могут не быть представлены вовсе. Описанная процедура применяется в том случае, когда промежуточная популяция имеет то же количество индивидов, что и исходная($n = n_i$), то есть потомки замещают родителей. Та же схема может быть применена и в случае, когда выбор осуществляется из промежуточной популяции большего размера, в этом случае возможно удаление выбранного индивида из исходной популяции.

Процедура селекции Бейкера (линейное ранжирование).

Адаптивно-
пропорциональная селекция имеет один существенный недостаток: она может привести к ситуации с одним наиболее приспособленным индивидом, фактически определяющим сходимость алгоритма к локальному максимуму. Чтобы избежать этого, Бейкером [24] была предложена следующая процедура. Индивиды в текущей популяции P^k упорядочиваются по убыванию адаптивности, фиксируется *статическое* вероятностное распределение $(p_1, \ldots, p_n)$, где

$$p_i = \frac{1}{n}\left(\eta_{max} - (\eta_{max} - \eta_{min})\frac{i-1}{n-1}\right)$$

Здесь $\eta_{min} = 2 - \eta_{max}$ и $1 \leq \eta_{max} \leq 2$. Экспериментально показано, что замена динамического распределения статическим улучшает качество работы алгоритма в мультимодальных задачах [24, 82].

Турнирная селекция . При использовании данного вида селекции из промежуточной популяции случайным образом выбираются пары индивидов, из которых более приспособленные попадают в следующее поколение. Данный тип селекции представляет особый интерес при теоретических исследованиях, так как не содержит стохастических параметров.

Детерминированный отбор наиболее приспособленных.
В случае, если промежуточная популяция содержит большее, нежели исходная, число индивидов ($n_i > n$), можно применять линейное ранжирование с детерминированным отбором. Это означает, что после упорядочивания первые n индивидов образуют популяцию следующего поколения. Такая схема может замедлить поиск, но способствовать большему разнообразию индивидов.

Дополнительным параметром селекции является *элитарность* (elitism). Элитарная селекция освобождает от конкурентного отбора некоторое количество наиболее приспособленных индивидов, непосредственно перенося их в популяцию следующего поколения.

Оператор скрещивания [33], [83] призван нести из поколения в поколение удачные сочетания «генов» и обеспечивать разнообразие генетических комбинаций в популяции. При скрещивании отбирается пара «индивидов-родителей», на основе значений которых создается новый «индивид-потомок». Вводится также дополнительный параметр - p_c, вероятность скрещивания. В стандартном генетическом алгоритме популяция случайным образом разбивается на пары (либо нужное число раз случайно отбирается пара индивидов), для каждой из пар с вероятностью p_c выполняется скрещивание, в результате которого получается один или два потомка, добавляемые в промежуточную популяцию. Выбранная пара «родителей» может быть удалена из исходной популяции, а также наряду с потомками перенесена в промежуточную. Наиболее распространены следующие варианты операторов:

Одноточечный кроссинговер (1-point crossover).
Пусть рассматривается пара индивидов $p^1, p^2 \in I$. Случайным образом выбирается число $d \in \{1; m\}$, которое мы будем называть *точкой разрыва*. Создается пара потомков c^1, c^2, для которых

$$c_i^1 = \begin{cases} p_i^1, & i \le d; \\ p_i^2, & i > d; \end{cases} \qquad c_i^2 = \begin{cases} p_i^2, & i \le d; \\ p_i^1, & i > d; \end{cases}$$

Таким образом, первый потомок «наследует» гены первого родителя слева от точки разрыва, второго - справа, а второй потомок - наоборот. Если подразумевается получение одного потомка, фиксируется выбор одного из них. Преимущество одноточечного кроссинговера в том, что он хорошо сохраняет удачные последовательности битов, что определило его использование в ранних вариантах генетических алгоритмов. Однако, поскольку набор битов потомков полностью определяется родителями, множество возможных вариантов битовых строк, получаемых в результате скрещивания, жестко ограничено и определяется начальной популяцией и операцией скрещивания. Одноточечный кроссинговер предоставляет мало возможностей для повышения разнообразия особей, что, наряду с адаптивно-пропорциональной селекцией, приводит к нежелательной сходимости алгоритма к локальному максимуму.

Двухточечный кроссинговер (2-point crossover). Наряду с 1-точечным, определяется и двухточечный кроссинговер, при котором определяются две точки разрыва d_1, d_2. Потомок c^1 наследует от p^1 компоненты от 1 до d_1 и от d_2 до m, а от p^2 – компоненты между d_1 и d_2. Потомок c^2, соответственно, наоборот. Аналогично, определяется k-точечный кроссинговер с точками разрыва $d_1, \ldots, d_k$. Но на практике обычно используется не более, чем двухточечный вариант, что считается эмпирически разумным компромиссом между сохранением битовых последовательностей и обеспечением разнообразия популяции.

Равномерный кроссинговер (uniform crossover) . При использовании такой операции каждый бит потомка равновероятно выбирается у одного из родителей. Равномерный кроссинговер представляет собой предельный случай, обеспечивая максимальное разнообразие, но не сохраняя удачных битовых последовательностей.

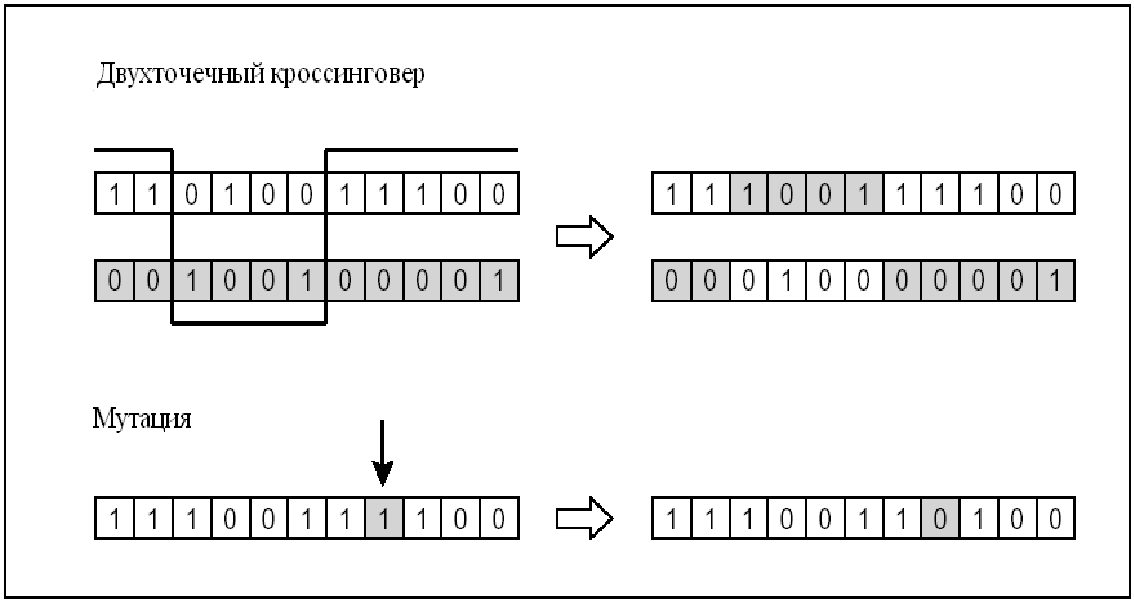


Рис. 2: Генетические операции: двухточечный кроссинговер и мутация инверсией битов.

Выбор варианта скрещивания определяется спецификой задачи. К примеру, в задачах оптимизации функционала $f(x_1, \ldots, x_l) : \mathbb{R}^l \to \mathbb{R}$ бинарное представление индивида задается склейкой l «генов» - бинарных представлений действительных переменных $x_1, \ldots, x_l$ с заданной точностью. В этом случае важны битовые последовательности, определяющие действительные числа. При унимодальной оптимизации наиболее эффективным считается одноточечный кроссинговер, для мультимодальной оптимизации более адекватны несколько точек разрыва. Часто накладываются естественные ограничения на множество допустимых точек разрыва. Если же каждый бит индивида является отдельным геном (например, если индивид является характеристическим вектором подмножества объектов) равномерный кроссинговер может быть заметно эффективнее.

Оператор мутации используется для обеспечения дополнительного разнообразия популяции. Как уже было сказано выше, оператор скрещивания производит новых индивидов на основе уже имеющихся, поэтому множество индивидов, которых можно получить путем рекомбинации, ограниченно и определяется только начальной популяцией. Чтобы расширить пространство поиска и тем самым способствовать глобальной оптимизации, используются мутации.*Мутацией* применительно к эволюционному алгоритму называется всякая модификация генов индивида, имеющая стохастическую природу. В генетическом алгоритме мутация производится инверсией некоторых битов индивида (bit-flip mutation).

Наиболее распространенной является следующая схема. Вводится параметр $p_d \in (0, 1)$ – вероятность мутации, то есть инверсии бита. Далее, для каждого бита каждого индивида в популяции с вероятностью p_d производится инверсия. Мутации обычно производятся после кроссинговера, в промежуточной популяции. В «традиционном» генетическом алгоритме роль мутации значительно меньше, нежели скрещивания. Считается, что рекомбинация является основным инструментом быстрого поиска оптимального решения, в то время как мутации должны давать гарантию, что ни одна точка пространства поиска не будет рассмотрена алгоритмом с нулевой вероятностью. Поэтому p_d берется равным не более, чем $\frac{1}{m}$, соответствующей в среднем одной мутации каждого индивида в поколении. Однако существуют исследования, показывающие, что мутации могут играть более важную роль при поиске оптимального решения [50]. Повышение p_d может позволить существенно расширить пространство поиска, а наиболее приспособленные индивиды могут быть защищены от нежелательных мутаций с помощью механизма элитарности (см. «оператор скрещивания»).

Критерий останова генетического алгоритма во многом определяет соотношение между качеством результата и затратами вычислительных ресурсов, прежде всего, времени. Поскольку глобальный максимум в задаче неизвестен или недостижим на практике, используются эвристические критерии, в основном одного из следующих трех типов:

- выполнение алгоритмом априорно заданного числа итераций или истечение отведенного машинного времени;
- выполнение алгоритмом априорно заданного числа итераций без (заметного) улучшения функции соответствия;
- достижение некоторого априорно заданного значения функции соответствия;

3.2 Эволюционные стратегии

Данный вид эволюционных алгоритмов отличается вещественнозначным представлением индивидов и ведущей ролью мутаций в процессе поиска оптимального решения. Параметры мутаций часто включаются в код индивида, то есть эволюционные стратегии часто являются самоадаптирующимися. Наиболее распространены так называемые эволюционные стратегии (μ, λ)-типа и $(\mu + \lambda)$-типа. Здесь μ соответствует числу индивидов в популяции родителей, λ – числу потомков в каждом поколении. В (μ, λ)-стратегиях в каждой итерации происходит генерация λ потомков, из которых

выбирается μ индивидов. В $(\mu + \lambda)$-стратегиях селекция производится из $(\mu + \lambda)$ индивидов - объединенной популяции родителей и потомков. Дадим более формальное описание одной из стратегий, вторая может быть описана аналогично [22]:

$$(\mu, \lambda) - ES = (P^0, m, \mu, \lambda, S, C, M, f, t)$$

где

$P^0 = (a_1, \ldots, a_\mu) \in I^\mu$, $I = \mathbb{R}^m$ — начальная популяция;

μ — количество родителей;

λ — количество потомков, $\lambda > \mu$;

m — длина вещественного представления индивида;

$S: \quad I^\lambda \to I^\mu$ — оператор селекции;

$C: \quad I \times I \to I$ — оператор скрещивания;

$M: \quad I \to I$ — оператор мутации;

$f: \quad I \to \mathbb{R}$ — функция соответствия;

$t: \quad I^n \to \{0, 1\}$ — критерий останова;

Подобно генетическим алгоритмам, операторы скрещивания и мутации C и M из популяции P^k размера μ создают промежуточную популяцию P^{ik} размера λ, после чего оператор селекции S выбирает μ индивидов на основе функции соответствия f. Алгоритм заканчивает работу, когда выполняется условие останова t. Соотношение между λ μ обычно берется в диапазоне $\mu/\lambda \approx 1/5 - 1/7$ [22].

Вариантов скрещивания векторов из R^n несколько больше, нежели в генетических алгоритмах. Все операторы, описанные применительно к генетическим алгоритмам, непосредственно переносятся и на эволюционные стратегии. Кроме того, существуют и специфичные операторы, как, например, линейные комбинации родителей [89], [23]. Точнее, если рассматривается пара индивидов p^1, p^2, то предлагаются потомки вида $\frac{1}{2}p^1 + \frac{1}{2}p^2, \frac{3}{2}p^2 - \frac{1}{2}p^1, \frac{3}{2}p^1 - \frac{1}{2}p^2$. Такого типа операторы скрещивания называют *линейным кроссинговером*. Показано, что использование линейного кроссинговера и линейного совместно с равновероятным кроссинговером (случайно выбирая один из вариантов) может дать лучшие результаты оптимизации [89].

Если скрещивание может быть перенесено непосредственно с бинарного на вещественное представление индивида, то мутации в эволюционных стратегиях имеют ярко выраженную специфику. Опишем два наиболее важных и используемых оператора:

Гауссова мутация [22].

Пусть мутируемый индивид представлен вектором $a = (a_1, \ldots, a_l)$. Определим мутировавший вектор $a' = M(a): \quad a'_i = a_i + N(0, \sigma_i)$, где $N(0, \sigma)$ – значение нормально распределенной случайной величины с нулевым математическим

ожиданием и среднеквадратическим отклонением σ; $\sigma = (\sigma_1, \ldots, \sigma_m)$ - вектор отклонений компонент вектора a. Вектор σ традиционно является в эволюционных стратегиях самоадаптирующимся параметром: каждый индивид в этом случае имеет длину $2m$, где первые m компонент кодируют решение задачи, а остальные определяют вектор σ среднеквадратических отклонений, использующихся при мутации данного индивида. Определяются также особые правила мутации этих дополнительных параметров, например: $\sigma' = \sigma \exp N(0, \Delta\sigma)$, где $\Delta\sigma$ – дополнительный контролирующий параметр, называемый *размером шага* (step-size). Главным отличием данной мутации от рассмотренной в предыдущем подразделе является ее применение не к отдельной компоненте(-ам) индивида, а ко всему индивиду в целом. Такой вид мутаций применяется в «чистых» эволюционных стратегиях, не использующих скрещивание. Существуют и другие способы задания мутации, также базирующиеся на нормальном распределении.

Равномерная мутация [89].

Данный вариант ближе к «генетической» мутации - инверсии битов. Подобно ей, фиксируется вероятность мутации отдельного компонента p_d, определяющая среднее число мутаций в поколении. Пусть выбран мутируемый компонент k вектора a со значением $a_k = x$. Пусть также фиксирован диапазон значений $[\,a, b\,]$ и максимальное изменение значения ΔM. Выбирается направление изменения значения. Если выбрано положительное направление, новое a_k определяется как значение случайной величины, распределенной равномерно в диапазоне $[\,x, min\{x + \Delta M, b\}\,]$. Если выбрано отрицательное направление, диапазон случайной величины определяется как $[\,max\{x - \Delta M, a\}, x\,]$. Такой вариант мутации используется в качестве аналога инверсии битов. Это означает, что ее применение характерно для алгоритмов, существенно использующих рекомбинацию (генетических алгоритмов с вещественнозначным представлением индивидов).

3.3 Коэволюционные алгоритмы. Кооперативная коэволюция.

Как уже говорилось выше, эволюционные алгоритмы успешно используются для решения сложны многопараметрических задач, к которым относится и задача распознавания. Тем не менее, существует множество задач подобного рода, неадекватных парадигме «классического» генетического алгоритма. Причины неадекватности разнообразны: сложность или невозможность построения фиксированной функции соответствия, слишком высокая комбинаторная сложность задачи, потребность в синтезе структур варьируемого размера и сложности. Естественным расширением парадигмы явились эволюционные системы из нескольких популяций, различно взаимодействующих друг с другом и предметной областью. Неизбежное усложнение и увеличение параметров системы в коэволюционных алгоритмах окупается дополнительными возможностями по увеличению разнообразия, разбиению задачи на более простые подзадачи, построению более адекватной функции соответствия и т.д. Рассмотрим несколько подходов к использованию нескольких популяций и обеспечению их эффективного взаимодействия.

Островная модель (Island Model). Идея заключается в создании нескольких независимо развивающихся, изолированных друг от друга популяций идентичной

структуры («популяции одного вида на разных островах»), занятых поиском решения одной и той же задачи [84], [83], [63]. Хотя популяции в целом изолированы, с «острова на остров» может происходить миграция отдельных индивидов, то есть между популяциями происходит обмен генетической информацией. Объем миграции контролируется параметром p_m – вероятностью миграции (migration rate). Известно, что качество работы островной модели напрямую зависит от p_m: большие значения миграции приводят к результатам, аналогичным алгоритму с одной популяцией, малые значения способствуют разнообразию популяций, но качество (значения функции соответствия) снижается по отношению к одной более крупной популяции. Островная модель является более эффективной в задачах мультимодальной оптимизации, так как независимые популяции способны найти различные локальные максимумы функции [63]. Таким образом, островная модель помогает справиться с преждевременной сходимостью генетического алгоритма к локальному максимуму без специализированных методик изменения функции соответствия и генетических операторов. Дополнительным достоинством островной модели является возможность эффективного применения параллельных вычислений, независимость популяций и редкая миграция позволяют использовать отдельный процессор для каждой популяции, что является чрезвычайно важным для медленных и ресурсоемких генетических алгоритмов. Островная модель использовалась, например, для построения решающего правила в форме ДНФ с помощью распределенного генетического алгоритма в рамках островной модели [47], а также для синтеза набора правил на основе нечеткой логики [62].

Соревновательная коэволюция (Competitive Coevolution) [19, 60, 70]. Идея взаимодействия, используемого в данной модели, происходит из взаимоотношений носителей (hosts) и паразитов (parasites). Для того, чтобы избавиться от вмешательства паразитов, вид-носитель вырабатывает специфические средства защиты, что, в свою очередь, заставляет паразитов изменяться, чтобы проникнуть в организм носителя и т.д. Таким образом, взаимоотношения носителей и паразитов повышают разнообразие и адаптивность обоих видов. Применительно к задачам, в которых качество решения определяется фиксированным критерием (как задачи обучения и распознавания), роль носителей могут играть потенциальные решения задачи (аналогично «обычному» эволюционному алгоритму). Роль паразитов же будут играть различные тесты, на которых проверяется качество решения задачи (например, различные разбиения множества обучающих объектов на тренировочную и контрольную выборку). Так, модель соревновательной коэволюции подразумевает наличие двух популяций: носителей и паразитов. Функция соответствия, а значит, и приспособленность индивидов обеих популяций строится на основе взаимодействия индивидов одной популяции с представителями другой. На каждой итерации алгоритма для каждого индивида-носителя выбираются один или несколько индивидов-паразитов, на которых он тестируется. Качество работы на предложенных тестах определяет приспособленность носителей. Приспособленность индивидов-паразитов тем выше, чем хуже качество работы носителей, которым был предложен для тестирования данный индивид. Такой вид взаимодействия поощряет появление все более сложных тестов, позволяющих в свою очередь получать все более качественные решения. Таким образом, соревновательная коэволюция позволяет заменить задаваемую вручную функцию соответствия эволюционирующей системой тестов.

Соревновательная коэволюция успешно применялась для задач удовлетворения ограничениям, настройки нейронных сетей [60], поиска стратегий в играх [19, 70] и т.д. Анализ данной модели, наряду с типичным для эволюционных теоретико-игровым подходом [38], имеет свою специфику. Так, в [31, 39, 32] коэволюция исследуется с помощью специального вида отношений порядка и оптимальности по Парето.

Кооперативная коэволюция

Кооперативная коэволюция является моделью взаимодействия нескольких видов, находящихся в симбиозе друг с другом. Каждый вид представлен популяцией своих индивидов, популяции изолированы (не смешиваются). Виды определяются таким образом, чтобы представитель одного вида кодировал часть решения задачи (или решение некоторой подзадачи). При этом совокупность индивидов, выбранных по одному из каждой популяции, соответствует решению задачи в целом. Такую совокупность индивидов будем называть *коллаборацией* (collaboration). Каждая из популяций развивается в соответствии с моделью некоторого эволюционного алгоритма, за исключением оценки адаптивности индивидов, зависящей от «окружающей среды», то есть от взаимодействия с представителями других популяций. Для оценки адаптивности индивида формируются одна или несколько коллабораций с его участием, качество которых как потенциальных решений и определяет приспособленность рассматриваемого индивида (рис. 3). Так, лучшим становится индивид, который лучше «сотрудничает» (кооперирует) с представителями других популяций. В зависимости от задачи и состояния системы, число популяций может увеличиваться или уменьшаться.

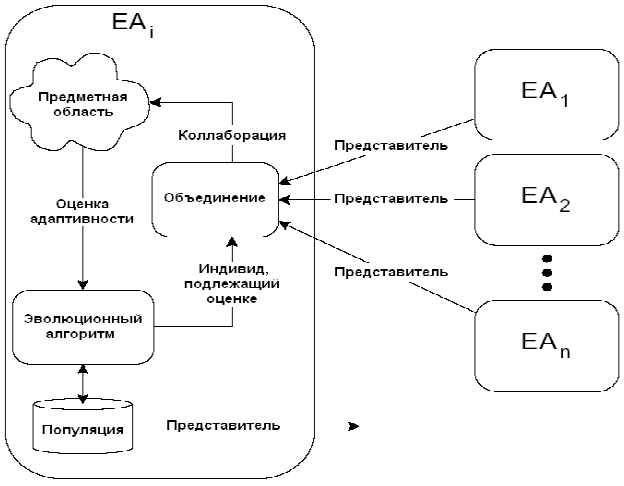


Рис. 3: Модель кооперативной коэволюции.

Вообще говоря, модель кооперативной коэволюции не накладывает никаких ограничений на количество и виды используемых для развития популяций эволюционных алгоритмов. Это значит, что одна популяция может контролироваться генетическим алгоритмом, другая — эволюционной стратегией и т.д. В данной же работе используется коэволюция популяций идентичной структуры и правил развития под

контролем генетического алгоритма (см. страницу 16). Поэтому более формальное описание и подробное рассмотрение будем производить для коэволюции набора популяций «одного вида», с бинарным представлением индивидов. После такого упрощения модель кооперативной коэволюции может быть записана в виде следующего набора:

$$CCGA = (GA, k^0, P^0, S^c, f, \sigma, a, r, t)$$

где

GA — Генетический алгоритм;

k^0 — Начальное число популяций системы;

$P^0 = (P_1^0, \ldots, P_{k^0}^0)$, $P_j^0 \in I^n$, $I = \{0,1\}^m$ — Начальный состав популяций;

$S^c \subset \mathfrak{S} = \{ S : I^n \to I \}$ — один или несколько операторов выбора индивида из популяции, определяющие правила составления коллабораций;

$f : I^k \to \mathbb{R}$ — функция соответствия, оценивающая коллаборацию длины k. Если число популяций может меняться, f должна быть определена для всех возможных k;

$\sigma : \mathbb{R}^d \to \mathbb{R}, \quad d = |\, S_c \,|$ — функция аггрегации оценок качества коллабораций для получения единой оценки адаптивности индивида;

$a : \mathfrak{K} \to \{0,1\}$ — критерий добавления популяции, где $\mathfrak{K}$ — весь коэволюционный процесс от начала работы алгоритма до текущего момента;

$r : \mathfrak{K} \to \{0,1\}$ — критерий удаления популяции;

$t : \mathfrak{K} \to \{0,1\}$ — критерий завершения работы;

Работу кооперативной коэволюции можно описать следующим образом (см. 3.1).

Алгоритм представляет собой итерационный процесс, итерация которого — последовательное изменение состава («смена поколений») каждой из популяций системы. Опишем набор действий, выполняемых на одном шаге процесса. В начале работы алгоритма создается k^0 популяций, инициализируемых случайными числами. Размер популяций и прочие их параметры определяются алгоритмом GA. Далее начинаются итерации алгоритма, производящие однотипные действия с популяциями системы. Опишем эти действия на примере одной популяции.

Рассмотрим популяцию P_i. Для нее выполняется селекция наиболее приспособленных индивидов, определяемая оператором селекции алгоритма GA. К выбранным индивидам применяются генетические операторы (скрещивание и мутация), также определяемые алгоритмом GA. Далее, производится оценка адаптивности полученных «потомков». Эта процедура является центральной во всем процессе, по-

Алгоритм 3.1 Алгоритм кооперативной коэволюции

$gen = 0$
$k = k^0$
для всех популяций $s = 1 \dots k$
  Инициализировать $P_s(gen)$ случайными числами
пока критерий останова = false
  для всех популяций s=1 ... k
    Селекция $P_s(gen)$ из $P_s(gen-1)$ на основе адаптивности
    Применение генетических операторов к $P_s(gen)$
    Оценка адаптивности индивидов $P_s(gen)$
  для всех популяций s=1 ... k
    если критерий удаления популяции $P_s(gen)$ = true **то**
      Удалить популяцию $P_s(gen)$
      k=k-1
  если критерий добавления популяции = true **то**
    добавить $P_{k+1}(gen)$, инициализировав случайными числами
    k=k+1
  gen=gen+1

этому остановимся на ней подробнее. Для этого введем дополнительные обозначения. Пусть $S \in S^c$ - некоторый оператор выбора индивида из популяции. Применением этого оператора выберем по одному индивиду из всех остальных популяций: $v_j = S(P_j)$, $j = 1..k$, $j \neq i$, обозначив этот набор $C(S)$. Пусть теперь $v_i \in P_i$ — индивид из рассматриваемой популяции, адаптивность которого подлежит оценке. Обозначим коллаборацию $(v_1, \dots, v_k)$ с участием рассматриваемого индивида v_i как $C(S)+v_i$. Адаптивность индивида v_i будем обозначать $A(v_i)$. Вычисление адаптивности для всех индивидов популяции P_i производится следующим образом (см. 3.2): для всех операторов из множества $S \in S^c$ формируются наборы представителей остальных популяций $C(S)$. Эти наборы последовательно дополняются каждым индивидом $v_i \in P_i$, формируя коллаборации вида $C(S) + v_i$. Коллаборации оцениваются с помощью функции соответствия f, результаты оценок объединяются в единую оценку с помощью функции аггрегации σ, значение которой и становится адаптивностью индивида v_i.

Алгоритм 3.2 Процедура оценки адаптивности индивида.

для всех $S_p \in S^c, p = 1, \dots, d$
  Сформировать набор представителей остальных популяций $C(S_p)$
для всех $v_i \in P_i$
  для всех $p = 1, \dots, d$
    $f_p = f(C(S_p) + v_i)$
  $A(v_i) = \sigma(f_1, \dots, f_d)$

Фиксация представителей остальных популяций позволяет оценивать конкурирующих индивидов одной популяции в идентичных условиях, способствуя отбору лучших в популяции. С другой стороны, операторы из S^c могут иметь стохастическую составляющую, и «неправильный» выбор представителей остальных популяций

может неверно «ориентировать» все поколение. Поэтому возможен вариант алгоритма, при котором для каждого индивида заново производится выбор представителей остальных популяций.

На каждом шаге алгоритма производится выполнение описанных действий для всех $P_i, i = 1 \dots k$. Далее, для каждой популяции проверяется критерий ее удаления r, и в случае положительного ответа соответствующая популяция удаляется из системы. После этого производится проверка выполнения критерия добавления вида a, и в случае его выполнения система дополняется еще одной популяцией, инициализированной случайным образом. Итерации продолжаются, пока не окажется выполнен критерий останова t.

Рассмотрим возможные варианты определения параметров алгоритма 3.3. Обсуждение параметров алгоритма GA приведено в подразделе 3.1, поэтому здесь остановимся на специфичных для кооперативной коэволюции элементах. К ним относится: множество операторов выбора индивида S^c, функция аггрегации σ и критерии добавления и удаления видов a и r. Ниже представлены некоторые варианты их определения.

Операторы выбора индивида (мультимножество S^c). Адаптивность индивида определяется коллаборациями, в которых он участвует, а значит, представителями других популяций. Поэтому коэволюция во многом определяется способом, которым выбираются индивиды для коллаборации с данным. Несмотря на аналитические и эмпирические исследования, проводимые в этой области ([87],[86],[85],[88]), вопрос об оптимальном количестве и виде операторов выбора индивида остается открытым. Приведем два варианта операторов, используемых в упомянутых исследованиях:

- Оператор «выбора лучшего» S_b: из популяции выбирается индивид с максимальным значением адаптивности;
- Оператор « случайного выбора» S_r: индивид из популяции выбирается случайным образом.

Эти операторы могут участвовать в создании одной или нескольких коллабораций: например, в [88] анализируются наборы из 3 и более коллабораций, первая из которых создается оператором S_b, а остальные S_r. В [87] производится тестирование системы с повторным использованием S_b: в первой коллаборации из популяции выбирается индивид с максимальным значением адаптивности, во второй – следующий по этому показателю и т.д. Таким образом, S^c представляет собой мультимножество, каждый элемент которого отвечает за формирование одной коллаборации. Результаты эмпирического анализа [87], проведенного для задачи оптимизации функций, показывают следующее: если разбиение решения задачи на части по популяциям хорошо соответствует разбиению задачи на подзадачи в предметной области, то наилучший результат показывает $S^c = \{S_b\}$; если же декомпозиция неадекватна, требуется большее число коллабораций с привлечением S_r.

Функция аггрегации σ. Когда коллаборации сформированы и качество их как потенциальных решений задачи оценено с помощью функции соответствия f, возникает задача объединения вычисленных оценок в единую оценку адаптивности индивида. В работах, посвященных анализу модели кооперативной коэволюции,

предложены три варианта аггрегации. Пусть $f_1, \ldots, f_d$ — оценки качества коллабораций, тогда предлагаются:

- «Оптимистичная» (optimistic) оценка: $\sigma_+ = \max_{j \in \overline{1,d}} f_j$;
- Усредненная (hedge) оценка: $\sigma_h = \frac{1}{d} \sum_{j=1}^{d} f_j$
- «Пессимистичная» (pessimistic) оценка: $\sigma_- = \min_{j \in \overline{1,d}} f_j$;

Эмпирический анализ [88, 87] позволяет выделить из этих вариантов оптимальный: практически во всех проводимых тестах лучше оказывалась «оптимистичная» оценка σ_+.

Критерий добавления популяции a**.** Изменение числа популяций в системе возможно только в тех задачах, где решение допускает динамическое наращивание структуры. При этом требуется именно возможность наращивания, а не динамического переразделения на подзадачи, потому что добавление популяции не должно повлечь за собой структурные изменения остальных популяций. Этому критерию не удовлетворяет, например, задача оптимизации функции n переменных, в этом случае $a \equiv 0$. Но рассматриваемая в данной работе задача построения локального базиса удовлетворяет этому ограничению, более того, динамическое наращивание числа алгоритмических операторов позволяет находить базисы малой мощности, поэтому подробное рассмотрение критериев a и r имеет смысл. Опишем критерий добавления популяции, предложенный в [63] и названный автором «условием стагнации». Под стагнацией понимается ситуация, когда в течение нескольких поколений не происходит заметного улучшения найденного алгоритмом решения. Условие стагнации задается двумя параметрами: эволюционным промежутком (evolutionary window) L_a и минимальным улучшением (minimal improvement) ε_a. Пусть $f(g)$ — лучший показатель адаптивности среди всех коллабораций, рассмотренных к шагу g. Тогда условие стагнации можно записать как

$$a_s = (f(g) - f(g - L_a) < \varepsilon_a)$$

Критерий удаления популяции r**.** Удаление популяции также возможно только в задачах динамического наращивания решения, но существует и дополнительное требование к предметной области алгоритма. Неформально, удаление популяции должно производиться в том случае, когда ее представители не способствуют улучшению решения. В «обычном» эволюционном алгоритме мерой качества каждого индивида являлась его приспособленность, но в кооперативной коэволюции адаптивность является мерой работы коллаборации в целом, и она не дает возможности оценить работу каждого из представителей различных популяций. Поэтому дополнительным требованием к задаче является возможность формализации понятия *вклада* (contribution), вносимого каждым из индивидов коллаборации «в общее дело». Итак, если оценить вклад отдельных индивидов не удается, $r \equiv 0$. Если же можно определить количественную оценку вклада, то можно оценивать вклад популяции на основе вкладов ее представителей, выбираемых операторами из S^c. Например, если $S^c = \{S_b\}$, то вклад $C(P_i)$ популяции P_i на данной итерации можно определить через вклад «лучшего» представителя $v^* = \arg\max_{v \in P_i} A(v_i)$. Обозначим через $C(P_i(g))$ вклад популяции P_i на шаге g, текущий шаг итерации обозначим G. В качестве условия удаления популяции P_i

можно рассмотреть малость суммарного вклада за последние L_r итераций:

$$r_+ = (\sum_{g=G-L_r}^{G} C(P_i(g)) < \varepsilon_r)\,,$$

или же малость вклада популяции в течение L_r итераций:

$$r_* = (\bigwedge_{g=G-L_r}^{G} (C(P_i(g)) < \delta_r))$$

Для некоторых задач, в которых выполняется условие динамического наращивания решения, но не удается построить количественную оценку вклада, можно использовать следующую процедуру *проверки изъятием*. Оцениваются две коллаборации — с участием и без участия индивида, вклад которого подлежит оценке. Вкладом индивида может считаться разность этих оценок или просто положительность этой разности, указывающая на «небесполезность» индивида.

4 Коэволюционный метод синтеза алгоритмических композиций

Данный раздел посвящен описанию алгоритма составления алгоритмических композиций, основанного на модели кооперативной коэволюции (см. 3.3) и позволяющего строить корректные алгоритмы, обладающие достаточно высокой обобщающей способностью. Подобно бустингу и баггингу, алгоритм использует метод обучения $\mu(X^q)$ семейства алгоритмических операторов $\mathfrak{M}^0$, который ставит в соответствие каждой обучающей выборке некоторый алгоритм из $\mathfrak{M}^0$. Описываемый метод обладает следующими свойствами:

1. Метод является универсальным, поскольку независим от структуры и метода обучения рассматриваемого семейства алгоритмов.

2. Метод не является жадным, что позволяет производить не локальную, а глобальную оптимизацию набора алгоритмических операторов.

3. Метод не накладывает ограничений на область значений генотипа, а значит, множество допустимых генотипов замкнуто относительно генетических операций. Это позволяет максимально использовать возможности генетического алгоритма по исследованию пространства поиска (search space) для нахождения наилучшего решения.

4. Модель кооперативной коэволюции способствует *динамической декомпозиции задачи* ([63, 66]), что позволяет строить оптимальные по мощности базисные наборы специализированных алгоритмов.

Постановка задачи. Пусть имеется обучающая выборки $X^q = \{x_1, \ldots, x_q\}$, $x_j \in \mathfrak{I}_i$, с заданными на ней номерами классов $Y^q = \{y_1, \ldots, y_q\}$, $y_j \in \mathfrak{I}_f = \{1, \ldots, l\}$. Обозначим через s число признаков, описывающих каждый объект в $\mathfrak{I}_i$. Также зафиксируем семейство алгоритмических операторов $\mathfrak{M}^0$, семейство корректоров $\mathfrak{F}$ и семейство решающих правил $\mathfrak{M}^1$. Как было отмечено выше, алгоритм классификации будет строиться в виде суперпозиции

$$A = C \circ F(B_1, \ldots, B_k), B_1, \ldots, B_k \in \mathfrak{M}^0, F \in \mathfrak{F}, C \in \mathfrak{M}^1 \tag{4.1}$$

Будем также считать, что определен неотрицательный функционал качества набора алгоритмических операторов $Q(B_1, \ldots, B_k)$, принимающий нулевое значение тогда и только тогда, когда $\exists F \in \mathfrak{F}, \exists C \in \mathfrak{M}^1$, такие, что алгоритм A вида (4.1) - корректный алгоритм. Без ограничения общности будем считать, что Q принимает значения от 0 до 1. Поставим задачу минимизации Q по набору алгоритмических операторов $B_1, \ldots, B_k$.

4.1 Описание метода

Предлагаемый метод использует модель кооперативной коэволюции вида (3.3), содержащую набор однотипных популяций под контролем генетического алгоритма.

Каждая популяция соответствует одному алгоритмическому оператору, число популяций в системе может динамически изменяться, соответственно меняя мощность образуемой композиции. Рассмотрим все аспекты построения такой системы, начав с задания семантики индивида и следуя набору параметров алгоритма (3.1).

Индивид: кодирование решения

Для определения системы следует в первую очередь описать, каким образом индивид кодирует алгоритмический оператор (в генетический терминах — «соответствие генотипа и фенотипа»). Индивиды системы являются двоичными векторами $v \in \{0,1\}^m$. Положим $m = q+s$, то есть длина индивида равна сумме числа объектов обучающей выборки и числа признаков, используемых для описания каждого объекта. Семантика индивида такова: он определяет подмножество объектов обучающей выборки и подмножество признаков для их описания. Точнее, компоненты $v_1, \ldots, v_q$ вектора v трактуются как характеристический вектор подмножества объектов, которое мы обозначим $X(v)$, а компоненты $v_{q+1}, \ldots, v_{q+s}$ — как характеристический вектор подмножества признаков, которое обозначим как $F(v)$. После такой трактовки каждому индивиду v можно сопоставить алгоритмический оператор $B(v) \in \mathfrak{M}^0$, полученный в результате применения метода обучения μ к подмножеству объектов $X(v)$, описываемых подмножеством признаков $F(v)$. Описанную процедуру обучения обозначим $\mu(v)$. Таким образом, $\mu(v)$ определяет принципиальное для каждого эволюционного алгоритма отображение «генотипа» в «фенотип». Имея такое отображение, становится очевидно, каким образом коллаборация индивидов кодирует совокупность алгоритмических операторов: каждому индивиду v_j коллаборации $(v_1, \ldots, v_k)$ сопоставляется алгоритмический оператор $B = \mu(v_j)$, а набор $B_1, \ldots, B_k$ далее может подвергаться коррекции и оценке функционалом качества Q, рассматриваемого в следующем разделе.

Функция соответствия

Постановка задачи подразумевает наличие некоторого функционала качества $Q(B_1, \ldots, B_k)$, по сути включающего в себя коррекцию и комбинирование с решающим правилом для построения суперпозиции вида (4.1) с последующей оценкой ее работы на обучающей выборке. Известно, что хорошая работа алгоритма на прецедентах не гарантирует хорошей обобщающей способности, поэтому существуют и другие критерии качества работы алгоритма, направленные на оценку его экстраполирующего поведения. Поэтому приведем несколько вариантов построения функции соответствия, различные по функционалу, подлежащему максимизации:

1. Уменьшение числа ошибок (accuracy).
   Уже упомянутый, наиболее простой и естественный вариант — заставить алгоритм меньше ошибаться на имеющихся объектах. Но экстраполирующая способность алгоритма не всегда соответствует качеству его работы на обучающей выборке, поэтому наряду с данным критерием вводятся и другие оценки, способствующие улучшению обобщающей способности алгоритма.
2. Увеличение разнообразия (diversity).
   Одним из таких критериев является мера разнообразия (diversity measure) совокупности классификаторов, формирующих композицию. Существует большое

количество способов определения меры разнообразия [56], использование конкретного варианта зависит в первую очередь от семейства корректирующих операций $\mathfrak{F}$ [36]. Однако мера разнообразия не используется в качестве единственного критерия качества, так как разнообразие набора классификаторов может быть достигнуто и при плохом качестве классификации каждого из них. Обычно применяется линейная комбинация меры разнообразия и количества верно классифицированных объектов ([71]). Например, в [80] максимизируется $Q = E + \alpha D$, где E — частота верной классификации на обучающей выборке, а D — мера разнообразия комбинации классификаторов. Проведенный анализ показал, что для различных реальных задач лучшие результаты получаются при различных α от 0.25 до 4.

3. Увеличение отступов (margins).
   Исследования причин феноменально высокой обобщающей способности бустинга [45, 75, 48, 74] привели к выводу о том, что бустинг даже после достижения нулевой ошибки на обучающей выборке продолжает увеличивать отступы объектов (см. раздел 2.2). Этот вывод способствовал появлению модифицированных вариантов алгоритма AdaBoost, обеспечивающих явную максимизацию отступов ([69, 72]). Эксперименты отвечали теории: модифицированные алгоритмы обладали лучшей обобщающей способностью и меньше были подвержены переобучению.

   Основываясь на этих выводах, предлагается использовать в качестве функции соответствия некую функцию отступов объектов обучающей выборки. В реализованном автором алгоритме производится максимизация усреднения отступов объектов, планируются эксперименты по максимизации минимального отступа.

Параметры коэволюции популяций

Определив, каким образом коллаборация кодирует вариант решения задачи и каким образом оценивается ее качество — в нашем случае это совокупность алгоритмических операторов и функционал качества Q — мы семантически определили систему в целом. Фактически, дальнейшее определение параметров играет роль технических подробностей, в предельном случае не имеющих значения. Под предельным случаем в данном контексте понимается работа алгоритма, позволяющая просмотреть все варианты решений, закодированных в виде коллабораций индивидов. Параметры кооперативной коэволюции, как и любого эволюционного алгоритма, определяют способ, каким пространство поиска будет исследоваться, но не семантику решения. Однако поскольку эволюционный алгоритм завершается после некоторого ограниченного числа итераций, его параметры в конечном счете определяют, каким будет лучший из просмотренных алгоритмом вариантов решения. Данное рассуждение приведено потому, что выбор параметров в отсутствие аналитического описания динамики алгоритма становится чисто эвристическим. Поэтому приводимые ниже варианты определения параметров алгоритма носят скорее рекомендательный характер. На базе общего подхода, описанного выше, можно создавать различные варианты алгоритма, подбирая числовые характеристики в зависимости от семейств $\mathfrak{M}^0$, $\mathfrak{F}$ и $\mathfrak{M}^1$. Приведем некоторые варианты задания параметров и укажем некоторые качественные предпосылки для выбора характеристик алгоритма, использованного автором при экспериментальном анализе.

Генетический алгоритм (ГА).

Генетический алгоритм в первую очередь определяет разнообразие рассматриваемых вариантов решения. Также характеристики ГА в первую очередь определяют скорость работы алгоритма (критическими параметрами являются размеры основной и промежуточной популяций n и n_i). Вариантов задания параметров существует множество (см. раздел 3.1). В данной работе использовался элитарный подход, при котором значительная часть индивидов $(\frac{1}{6} - \frac{1}{3})$ является «элитой» и непосредственно переносится из популяции текущего поколения в следующую. С этим сочетается высокая вероятность мутации p_m $(0.05 - 0.1)$, призванная обеспечить высокое разнообразие индивидов значительными изменениями в оставшейся «неэлитарной» части популяции. Некоторое эмпирическое обоснование такого подхода можно найти в [61]. Также для повышения разнообразия популяции использован равномерный кроссинговер. Так как в нашем случае каждый бит индивида значим сам по себе (является отдельным «геном»), равномерный кроссинговер обеспечивает большее по сравнению с двухточечным кроссинговером разнообразие индивидов, не портя их разрушением столь нужных для мультибитовых генов последовательностей. Размер промежуточной популяции брался в 2–3 раза большим, чем размер основной популяции, что характерно скорее для эволюционных стратегий, но опять же служит для быстрейшего исследования бóльших областей пространства поиска. Для селекции реализованный алгоритм использовал детерминированный отбор.

Контроль числа популяций (k^0, a, r).

Задача построения композиции классификаторов отвечает требованию динамического наращивания решения (см. 3.3), позволяющего динамически изменять количество популяций системы, что соответствует изменению мощности композиции. Поэтому имеет смысл рассматривать совместно начальное число популяций k^0 и критерии a и r добавления и удаления популяции. Эти три параметра определяют стратегию наращивания (фиксации, уменьшения) мощности композиции алгоритмических операторов. Наиболее естественной является стратегия постепенного наращивания мощности, начиная с малого числа алгоритмических операторов. Такая стратегия используется в алгоритме, использованном автором для экспериментов, причем $k^0 = 1$. Так поиск лучших сочетаний может быть ускорен, поскольку вновь добавленная популяция кооперирует с уже развитыми, нашедшими свою «нишу специализации», популяциями. С другой стороны, более «старые» популяции диктуют новой направление развития, которое может оказаться «политикой невмешательства», только чтобы не испортить существующий порядок вещей. Поэтому возможны стратегии с большим k^0 и отсутствием a, что может лучше способствовать глобальной оптимизации. Однако следует помнить о сложности работы алгоритма, растущей экспоненциально с увеличением числа популяций. К одному из направлений дальнейшей работы следует отнести также эксперименты со стратегией последовательного уменьшения мощности композиции по мере создания ниш специализации алгоритмов.

Рассмотрим также варианты определения критериев a и r. В отличие от большинства параметров эволюционных алгоритмов, критерий добавления популяции a не имеет разнообразия вариантов. Подобно другим коэволюционным системам ([63, 66, 87]), описываемый алгоритм использует в качестве a условие

стагнации (см. раздел 3.3). Выбор же критерия удаления популяции r не столь однозначен. Дело в том, что для произвольной корректирующей операции нельзя определить единое понятие «вклада» индивида в коллаборацию, являющегося центральным при построении r. Упомянутая процедура проверки изъятием, хотя и применима к задаче построения локального базиса, вдвое увеличивает число вычислений для оценки адаптивности индивида, являющейся наиболее ресурсоемкой в системе, поэтому может оказаться практически неэффективной. Однако для конкретного семейства $\mathfrak{F}$ корректирующих операций формализация понятия вклада может заметно упроститься. Так, в следующем разделе описан критерий удаления популяции r, использованный в задаче с линейной корректирующей операцией.

Формирование коллабораций (операторы S^c).

Формирование коллабораций является процедурой, связывающей популяции между собой в единую систему. Выбор количества и вида коллабораций, в которых должен участвовать данный индивид для получения оценки адаптивности, определяется соответствием между разделением задачи на подзадачи в предметной области и делением решения на части по популяциям внутри коэволюционной системы. В [87],[88] показано, что чем более точно «внешнее» деление соответствует «внутреннему», тем меньшее число коллабораций необходимо и тем предпочтительнее детерминированный оператор выбора (S_b) перед стохастическим (S_r). Так, на ряде экспериментов с оптимизацией функций многих переменных одна коллаборация, составленная из данного индивида и лучших по показателю адаптивности представителей других популяций (в наших обозначениях $S^c = \{S_b\}$) являлась наиболее эффективной (с точки зрения качества результирующего решения при одинаковом количестве вычислений), когда каждой переменной функции соответствовала отдельная популяция. В некотором смысле задача построения композиции классификаторов с помощью алгебраической коррекции аналогична задаче оптимизации функции, роль переменных здесь играют алгоритмические операторы. В силу приведенных рассуждений и в силу вычислительной простоты, немаловажной в коэволюционной системе, автором использовано $S^c = \{S_b\}$. Наиболее развернутый анализ в этой области представлен в [87].

Критерий завершения работы (t).

В данной работе, подобно большинству эволюционных алгоритмов, используется критерий останова, соответствующий *длительной стагнации*, то есть большое число итераций, проходящих без заметного улучшения адаптивности лучшего из найденных решений. Количественные характеристики данного критерия, наряду с параметрами контроля числа популяций (см. выше), отвечают за время работы алгоритма. Фактически они выбираются как эвристико-эмпирический компромисс между вычислительной сложностью и шансом улучшения решения.

Собрав воедино информацию об использованных вариантах определения параметров, работу алгоритма можно представить следующим образом (4.1):

В соответствии со схемой работы кооперативной коэволюции (см. 3.1), алгоритм является итерационным процессом, на каждой итерации которого последовательно

Алгоритм 4.1 Коэволюционный алгоритм построения обучаемых алгоритмических композиций

Основные обозначения:
k^0 — начальное количество популяций в системе;
k — текущее количество популяций в системе;
g — текущее поколение процесса;
$P_i(g)$ — i-ая популяция на шаге g
$A(v)$ — адаптивность индивида v
$f(g)$ — самый высокий достигнутый к шагу g показатель адаптивности
$\mu(v)$ — алгоритм, настроенный на подмножествах объектов и признаков, определенных вектором v
$I(x_i, a(x_i))$ — индикатор ошибки алгоритма a при классификации объекта x_i
$gMax$ — максимальное допустимое число поколений
$sgMax$ — максимальное допустимое число поколений без улучшения $f(g)$
L — эволюционный промежуток (evolutionary window), в течение которого ожидается увеличение $f(g)$ не менее, чем на ε, после чего считается выполненным условие стагнации
$g = 0$
$sg = 0$
$k = k^0$
для всех $i = 1 \dots k$
  $P_i(g)$ =популяция случайных индивидов
пока $g < gMax$ и $sg < sgMax$
  для всех $i = 1 \dots k$
    $P'_i(g) =$ Скрещивания$(P_i(g))$
    $P''_i(g) =$ Мутации$(P'_i(g))$
    для всех $j = 1 \dots p, j \neq i$
      $v_j^* = \arg\max_{v_j \in P_j(g)} A(v_j)$
      $B_j^* = \mu(v_j^*)$
    для всех $v_i \in P''_i(g)$
      $B_i = \mu(v_i)$
      $A(v_i) = 1 - Q(B_1^*, \dots, B_{i-1}^*, B_i, B_{i+1}^*, \dots, B_p^*)$
      если $A(v_i) > f(g)$ **то**
        $f(g) = A(v_i)$
    $P_i(g+1) =$ Селекция$(P''_i(g))$
  если $f(g) - f(g-L) < \varepsilon$ **то**
    $k = k + 1$
    $P_{k+1}(g+1) =$ популяция случайных индивидов
  если $f(g) = f(g-1)$ **то**
    $sg = sg + 1$
  иначе
    $sg = 0$
  $g = g + 1$

происходит «смена поколений» каждой из популяций. Из популяции P_i с помощью скрещивания и мутаций создается промежуточная популяция P''_i, каждый инди-

вид v_i которой подлежит оценке. Для этого из остальных популяций выбираются лучшие по адаптивности индивиды v_j^*, $j = 1 \ldots, k, j \neq i$, определяющие набор алгоритмических операторов $B_j^* = \mu(v_j^*)$, $j = 1 \ldots, k, j \neq i$. Этот набор последовательно дополняется операторами вида $B_i = \mu(v_i)$ и оценивается функционалом Q, на основе которого вычисляется приспособленность индивида v_i: $f(v_i) = 1 - Q$, так как в отличие от функционала Q приспособленность тем лучше, чем выше значение f. После получения оценок индивидов производится селекция из P''_i, формирующая популяцию следующего поколения. В случае выполнения условия стагнации происходит добавление новой популяции, инициализированной случайными числами. Если выполняется условие длительной стагнации, алгоритм заканчивает работу. Заметим, в этом общем алгоритме отсутствует критерий удаления популяции r. Это обусловлено тем, что для произвольных корректирующих операций не сформулировано понятие «вклада» индивида в коллаборацию, а процедура проверки изъятием представляется автору вычислительно неэффективной.

4.2 Достоинства и недостатки предложенного метода

Данный раздел посвящен обсуждению свойств и возможностей предложенного метода, как положительных, так и отрицательных. К основным его достоинствам можно отнести следующие.

1. Данный метод, подобно бустингу и баггингу, представляет собой надстройку (wrapper) над существующими алгоритмами классификации. Обучение каждого из алгоритмов композиции производится с использованием стандартного для соответствующей модели метода. Это позволяет задействовать всю совокупность наработанных и обоснованных методик обучения отдельных алгоритмов для построения их композиций.
2. В отличие от бустинга и баггинга, явно использующих в качестве корректирующей операции (взвешенное) голосование, предложенный метод позволяет использовать произвольные корректирующие операции, не изменяя структуры алгоритма.
3. Все перечисленные в разделе 2 методы построения композиций используют жадные стратегии последовательного наращивания числа алгоритмов в композиции. При этом в лучшем случае производится локальная оптимизация алгоритма, направленная на компенсацию ошибок предыдущих алгоритмов композиции. Предлагаемый коэволюционный метод осуществляет оптимизацию работы отдельных алгоритмов, основываясь на информации о качестве классификации композиции в целом. Эта его особенность обеспечивает оптимизацию, близкую к глобальной.
4. Метод предоставляет большую гибкость в выборе критерия оптимизации: в частности, возможно явным образом максимизировать отступы объектов обучающей выборки и разнообразие алгоритмов в композиции. Также изменение стратегии наращивания композиции, включение/исключение отбора признаков или объектов и варьирование других стратегических параметров позволяет получать модификации алгоритма, обладающие различными свойствами и возможностями.
5. Метод позволяет говорить о смысловой интерпретации получаемых композиций

в терминах специализации алгоритмов на различных частях обучающей выборки и подмножествах признаков. Возможна корректировка алгоритма для указания «областей компетентности» алгоритмов композиции, модифицируемых с помощью механизма обратной связи.

6. Анализ работы метода позволяет выделять наиболее информативные признаки и объекты. Появляется возможность (повторной) предварительной обработки обучающей выборки. Таким образом, метод может быть использован итерационно с дополнительной обработкой входной информации на каждом этапе.
7. Как эволюционный алгоритм, метод не накладывает ограничений на область значений генотипа, что способствует более эффективному «свободному» поиску в пространстве решений.
8. Подобно другим алгоритмам на основе кооперативной коэволюции, данный метод допускает эффективную организацию параллельных вычислений, способных значительно повысить скорость работы алгоритма. При параллельных вычислениях популяции могут развиваться одновременно, обмениваясь лишь индивидами для составления коллабораций [87].

Наряду с важными достоинствами, предложенный метод имеет и серьезные недостатки, унаследованные от эволюционных алгоритмов в целом и кооперативной коэволюции в частности.

1. Динамика процесса построения композиции настолько сложна, что не поддается теоретическому анализу. Следствием этого является отсутствие доказательства сходимости метода и оценок качества его работы. Подбор параметров осуществляется эмпирическим путем, что требует серьезной предварительной работы.
2. Метод требует большого объема ресурсов, в первую очередь времени работы. Это ограничение пока не позволяет использовать его для сложных семейств алгоритмов, таких, как деревья решений и нейронные сети, в то время как именно эти семейства алгоритмов находятся в фокусе исследований по построению эффективных композиций [45, 30, 58, 25, 34, 92, 95]. Возможно, в дальнейшем это ограничение будет снято путем распараллеливания работы алгоритма.
3. Модель кооперативной коэволюции, составляющая основу метода, накладывает свои ограничения на оптимизационные возможности метода. В частности, в [85, 87] показано, что кооперативная коэволюция в некоторых случаях не находит глобального максимума функции многих переменных. Авторы объясняют это несоответствием между разбиением исходной задачи на независимые подзадачи и разделением кодировки решения по популяциям. В случае, если в разных популяциях оказываются потенциальные решения зависимых подзадач, «покомпонентная» оптимизация, проводимая в рамках кооперативной коэволюции, может оказаться неэффективной. Вопрос о существенности этого ограничения для задачи построения локального базиса является одним из важнейших направлений дальнейшей работы над развитием данного метода.

Совокупность достоинств и недостатков предлагаемого коэволюционного подхода к построению алгоритмических композиций позволяет говорить о его перспективности, однако оставляет открытыми многие важные вопросы, такие, как сходимость, применимость и эффективность данного метода.

4.3 Алгоритм синтеза локальных базисов с линейной корректирующей операцией для задачи с двумя классами

Данный раздел описывает пример приведенного алгоритма с использованием семейства линейных корректирующих операций (или, в других терминах, взвешенного голосования) для задачи разделения множества объектов на два класса. Приведем постановку такой задачи. Имеется множество X^q объектов обучающей выборки $X = \{x_1, \ldots, x_q\}$, $x_j \in \mathfrak{I}_i$, которым соответствуют $Y = \{y_1, \ldots, y_q\}$, $y_j \in \mathfrak{I}_e = \mathfrak{I}_f = \{-1, +1\}$. Отметим, что условие $\mathfrak{I}_e = \mathfrak{I}_f$ подразумевает отсутствие решающих правил, заменяемых непосредственно значениями алгоритмических операторов. Также зафиксировано множество $\mathfrak{F}_\mathfrak{L}$ линейных корректирующих операций:

$$\mathfrak{F}_\mathfrak{L} = \left\{ F(B_1, \ldots, B_k) = \sum_{i=1}^{p} \alpha_i B_i \,|\, \alpha_i \in \mathbb{R}, i = \overline{1,k} \right\} \tag{4.2}$$

Для построения суперпозиции вида (4.1) при фиксированных $B_1, \ldots, B_k$ необходимо определить коэффициенты $\alpha_i \in \mathbb{R}, i = \overline{1,k}$ корректирующей операции $F \in \mathfrak{F}_\mathfrak{L}$. Предлагаемый способ определения весов алгоритмов близок к используемому в бустинге в том смысле, что вес алгоритма в композиции определяется его качеством классификации. Но алгоритмы бустинга тестируют алгоритмический оператор на всем множестве объектов X, а в предлагаемом алгоритме — только на тех объектах, которые не использовались для обучения. Формализуем этот подход. Пусть оператор B_i был настроен на подмножестве объектов $X(v_i)$, определенном индивидом v_i. Рассмотрим подмножество объектов $\overline{X} = X \setminus X(v_i)$ тех объектов, которые не предъявлялись алгоритму B_i во время обучения. Пусть q_i - число ошибок классификации алгоритма B_i на множестве $\overline{X}$. Определим коэффициент α_i при операторе B_i в композиции (4.2) следующим образом:

$$\alpha_i = \max\left\{ \ln \frac{q - q_i + 1}{q_i + 1}, 0 \right\} \tag{4.3}$$

Таким образом, чем меньше ошибок B_i делает на «новых для него» объектах, тем выше его вес. Если же $q_i > \frac{q}{2}$, то есть алгоритм работает хуже классификации наугад, его вес принимается равным 0.

Теперь по набору алгоритмических операторов однозначно определяется суперпозиция вида 4.1, которая может быть оценена с точки зрения качества классификации. Основываясь на такой оценке, определим функцию соответствия f. В подразделе 4.1 указаны три основных функционала, максимизация которых может способствовать лучшему решению задачи, в первую очередь, с точки зрения обобщающей способности строящейся композиции. Фиксация семейств $\mathfrak{F}$ и $\mathfrak{M}^1$ позволяет записать эти функционалы в виде формул. Итак, предлагаются следующие варианты определения функции соответствия:

1. **Доля верно классифицированных объектов из обучающей выборки.** Правильность классификации на множестве прецедентов X является наиболее распространенным критерием качества алгоритма классификации. Отвечающую этому подходу функцию соответствия можно записать в следующем виде:

$$f_a = \sum_{i=1}^{q} [F(B_1, \ldots, B_k)(x_i) = y_i]$$

2. **Мера разнообразия.**
   Как уже отмечалось выше, мера разнообразия D набора алгоритмов вводится для повышения обобщающей способности композиции. Она используется в сочетании с критерием f_a, чтобы направить эволюционный процесс в сторону более разнообразных композиций, сохраняя качество классификации. Применяемая для этого функция соответствия

$$f_{ad} = f_a + \alpha D$$

   порождает дополнительный параметр α, оптимальное значение которого также подбирается эмпирически и может быть различным для разных задач. Тем не менее, введение меры разнообразия оправдано результатами экспериментов [71, 80, 56]. Отдельным вопросом при использовании функции соответствия f_{ad} является выбор меры разнообразия D. В данной работе для экспериментального анализа использовалась мера разнообразия Кохави–Вулперта (Kohavi–Wolpert), определяемая следующим образом. Пусть q — число объектов обучающей выборки, k — число алгоритмов в композиции, $l(x_i)$ — число алгоритмов, верно классифицирующих объект x_i. Тогда мера разнообразия D_{kw} определяется как

$$D_{kw} = \frac{1}{Nk^2} \sum_{i=1}^{q} l(x_i)(k - l(x_i))$$

3. **Отступы объектов обучающей выборки.**
   Для линейной композиции алгоритмов классификации $F = \sum_{i=1}^{k} \alpha_i B_i$ отступ объекта $x_j \in X$ записывается как

$$M(x_j, y_j) = y_j \cdot F(x_j) = y_j \cdot \sum_{i=1}^{k} \alpha_i B_i(x_j)$$

   Предлагаемая функция соответствия f_m максимизирует «средний отступ»:

$$f_m = \frac{1}{q} \sum_{j=1}^{q} M(x_j, y_j)$$

   Данная оценка коррелирует с оценкой f_a в том смысле, что уменьшение числа ошибок на обучающей выборке приводит к увеличению f_m. Но, дополнительно, f_m поощряет композиции, в которых операторы «соглашаются друг с другом» на тех объектах, которые композиция классифицирует верно и «спорят» в тех случаях, когда результат неверный. Такие композиции обладают лучшей обобщающей способностью [69, 72, 59].

Следует отметить отдельно небольшую коррекцию критерия добавления популяции для линейных корректирующих операций. В случае, если выбирается стратегия наращивания числа алгоритмов, и наращивание начинается с одного алгоритма, критерий добавления популяции дополняется следующим условием. При первом наращивании числа алгоритмов добавляется сразу две новые популяции, так как композиция из двух алгоритмов классификации с голосованием в качестве корректирующей операции не имеет смысла. Более того, эксперименты показали, что коэволюция двух популяций в таком случае приводит к «политике невмешательства»

как нише специализации одного из алгоритмов, что заметно усложняет построение эффективных композиций на основе этой пары алгоритмов. Назовем данное дополнение «правилом 1-3».

Для линейной корректирующей операции оказывается возможным определить легко вычисляемый и адекватный предметной области критерий удаления популяции r. Рассмотрим индивид $v_i \in P_i$. Ему соответствует алгоритмический оператор $B_i = \mu(v_i)$, который входит в композицию из k операторов со своим весом w_i. Определим вклад индивида v_i как

$$C(v_i) = \frac{w_i}{\sum_{j=1}^{k} w_j}$$

Вклад популяции $P_i(g)$ в поколении g определим как вклад лучшего из ее индивидов:

$$C(P_i(g)) = C(\arg \max_{v_j \in P_j(g)} A(v_j))$$

Вклад популяции определяется индивидом с наибольшей адаптивностью, потому что именно этот индивид участвует в коллаборациях с индивидами других популяций при оценке их адаптивности (в силу $S^c = \{S_b\}$) и поэтому оказывает наибольшее влияние на эволюцию системы в целом. Имея количественную оценку вклада популяции, можно различными способами определять критерий удаления популяции (см. 3.3). В данной работе используется критерий вида r_*:

$$r_*(P_i(G)) = (\bigwedge_{g=G-L_r}^{G} (C(P_i(g)) < \delta_r))$$

Этот критерий отражает ситуацию, когда в течение L_r поколений нормированный вес алгоритмического оператора, соответствующего лучшему в популяции индивиду, не превышает малой величины δ_r, то есть фактически этот алгоритмический оператор не участвует в процессе принятия решения. То, что популяция «работает вхолостую» в течение нескольких поколений подряд, свидетельствует о том, что такое ее положение устойчиво, то есть нишей специализации данной популяции становится «невмешательство», что и является поводом ее удаления.

Уточнив определение функции соответствия и добавив критерий удаления популяции, мы фактически определили алгоритм синтеза локальных базисов с использованием линейных корректирующих операций для задачи с двумя классами. Перечислим набор параметров алгоритма, следуя введенной нотации и определению кооперативной коэволюции 3.3:

1. Длина индивида $m = q + s$, где q — число объектов обучающей выборки, s — число признаков, описывающих каждый объект. Индивид кодирует подмножество объектов и подмножество признаков, каждому индивиду v соответствует алгоритмический оператор $B = \mu(v)$.

2. Начальное число популяций $k^0 = 1$, используется стратегия постепенного наращивания мощности композиции с возможностью удаления «лишних» популяций.

3. Начальная популяция $P_1(0)$ инициализируется случайными числами.

4. Мультимножество правил выбора индивида из популяции $S^c = \{S_b\}$, формируется одна коллаборация, для которой выбираются лучшие по показателю адаптивности представители остальных популяций.

5. Функция адаптивности $f = f_m$, максимизируется среднее значение отступов объектов обучающей выборки.
6. Функция аггрегации σ не определяется, так как для оценки адаптивности индивида формируется только одна коллаборация.

7. Критерий добавления популяции $a = a_s$, условие стагнации. Также используется описанное выше «правило 1-3» перехода от одной популяции сразу к трем.

8. Критерий удаления популяции $r = r_*$ — популяция в течение нескольких итераций не участвует в процессе принятия решения.

9. Критерий останова t — условие длительной стагнации.

Конкретные числовые значения констант, используемых в параметрах генетического алгоритма, а также в критериях a, r, t не указаны потому, что до проведения экспериментов большего объема и выявления хотя бы общих эмпирических закономерностей нельзя говорить о каких-либо рекомендациях, кроме качественных, которые и приведены в двух последних разделах. Следует еще раз подчеркнуть, что описанный набор параметров является эвристическим и не претендует на оптимальность, но является базой экспериментального анализа, описанного в следующем разделе.

4.4 Экспериментальный анализ

Базовый алгоритм классификации: правило Байеса.

В качестве базового алгоритма классификации, на основе которого будет строиться композиция, в данной работе выбрано правило Байеса (наивный байесовский классификатор). Представим краткое описание этого метода. Пусть имеется обучающая выборка $X = \{x_i, y_i\}_{i=1}^q$, $x_j = (f_1, \ldots, f_s)$, $y_j \in \{1 \ldots l\}$. Для определения класса y_k объекта x_k, не входящего в обучающую выборку, предлагается оценить апостериорную вероятность $P(i \,|\, x_k)$ класса i при условии предъявления объекта x_k, для всех классов $i = 1 \ldots l$. Объект относится к тому классу, для которого значение такой вероятности наибольшее. Но поскольку значение $P(i \,|\, x_k)$ неизвестно, для его вычисления применяется формула (правило, теорема) Байеса:

$$P(i \,|\, x_k) = \frac{P(i)P(x_k \,|\, i)}{P(x_k)}$$

Для определения $P(x_k \,|\, i)$ предполагается, что признаки в описании объектов независимы, что позволяет представить эту вероятность в виде произведения $P(x_k \,|\, i) = \prod_{j=1}^s P(f_j \,|\, i)$. Учитывая, что априорная вероятность $P(x_k)$ постоянна для всех классов и не влияет на процесс классификации, определение класса объекта

производится так:

$$y_k = \arg\max_{i \in \{1 \ldots l\}} P(i) \prod_{j=1}^{s} P(f_j \,|\, i) \tag{4.4}$$

Значения вероятностей, участвующих в данной формуле, оцениваются на основе статистической обработки обучающей выборки X. Отметим, что для задачи с двумя классами решение принимается на основе сравнения двух оценок апостериорной вероятности.

Формула 4.4 была получена из предположения о независимости признаков, описывающих объекты классификации. Данное предположение на практике чаще всего неверно, поэтому данный алгоритм называют "наивным". Тем не менее, байесовский классификатор может показывать хорошие результаты даже в случае нарушения указанного предположения [80]. Данное обстоятельство в сочетании с низкой вычислительной сложностью и послужило основой для выбора байесовского правила в качестве базового для построения алгоритмической композиции. Данный алгоритм мало используется для построения композиций, поскольку он обладает свойством чрезвычайной стабильности. Современные методики построения алгоритмических композиций, такие, как бустинг и баггинг, базируются на варьировании обучающей выборки, поэтому они не дают заметного улучшения качества классификации для стабильных алгоритмов. Более того, они могут даже ухудшить его, поэтому применение бустинга и баггинга к байесовкому алгоритму классификации считается нецелесообразным. Данный постулат будет проверен далее при проведении экспериментов на модельных и реальных задачах.

Методология экспериментов

Для оценки работы алгоритмов как для модельных, так и для реальных задач производилась следующая процедура:

1. Обучающая выборка случайным образом *со стратификацией* разделялась на тренировочную и контрольную подвыборки в пропорции 7: 3. Стратификация гарантирует сохранение в подвыборках соотношения числа элементов из разных классов.
2. На тренировочной подвыборке ($\frac{7}{10}$ исходной) производилось построение композиции алгоритмов.
3. Полученная композиция использовалась для классификации объектов из контрольной подвыборки ($\frac{3}{10}$ исходной), подсчитывался процент неверно классифицированных объектов.

Для получения оценок качества классификации данная процедура повторялась 100 раз, процент ошибок усреднялся по всем запускам алгоритма. Различные алгоритмы тестировались на одних и тех же разбиениях.

Для тестирования кроме коэволюционного метода были выбраны алгоритм бустинга AdaBoost (см. 2.2), алгоритм баггинга, предложенный Брейманом (см. 2.3) и метод случайных подпространств (RSM, см. 2.4). Для этих методов размер строящейся композиции был принят равным 250 [30].

Эксперименты на модельных задачах.

Для проверки возможностей коэволюционного метода на хорошо изученном и наглядном материале были смоделированы 4 задачи классификации с двумя классами, объекты обучения которых располагаются на плоскости ($\mathfrak{I}_i = \mathbb{R}^2$). Обозначим классы через C_1 и C_2. Через $N(m_x, \sigma_x, m_y, \sigma_y)$ обозначим двумерное нормальное распределение с параметрами (m_x, σ_x) по оси абсцисс и (m_y, σ_y) по оси ординат. Перечислим рассматриваемые модельные задачи.

1. Задача `Gauss-1`. Класс C_1 представляет собой выборку из распределения $N(m_{x1}, \sigma_{x1}, m_{y1}, \sigma_{y1})$, где $m_{x1} = 1, \sigma_{x1} = 1, m_{y1} = 2, \sigma_{y1} = 1$. Класс C_2 является выборкой из распределения $N(m_{x2}, \sigma_{x2}, m_{y2}, \sigma_{y2})$, где $m_{x2} = 5, \sigma_{x2} = 1, m_{y2} = 2, \sigma_{y2} = 1$. Таким образом, второй класс «сдвигается» относительно первого по оси x на 4 единицы. Получаем два компактных класса со слабым перекрытием.
2. Задача `Gauss-2`. Класс C_1 остается тем же, что и в задаче 1. Класс C_2 также остается выборкой из двумерного нормального распределения, но параметры его теперь таковы: $m_{x2} = 4, \sigma_{x2} = 2, m_{y2} = 2, \sigma_{y2} = 2$. Второй класс теперь ближе к первому и более «растянут».
3. Задача `Gauss-3`. Класс C_1 остается тем же, что и в задаче 1. Класс C_2 несколько «отодвигается» от него: $m_{x2} = 6, \sigma_{x2} = 1, m_{y2} = 2, \sigma_{y2} = 1$. Далее, к первому классу добавляются объекты, полученные смещением имеющихся в классе точек по оси x на $m_{x2} - m_{x1} + 3(\sigma_{x2} + \sigma_{x1})$. Таким образом, получаем компактный набор точек, расположенный между двумя идентичными компактными множествами так, чтобы пересечение маловероятно. Задача сложна для правил Байеса, так как наилучшее решение представляет собой пару наклонных прямых.
4. Задача `Gauss-4`. Класс C_1 берется из задачи 3. Класс C_2 «сдвигается» в положительном направлении по обеим осям: $m_{x2} = 12, \sigma_{x2} = 1, m_{y2} = 8, \sigma_{y2} = 1$. Задача заметно упрощается, так как теперь лучшей является поверхность второго порядка.

Результаты расчетов приведены в нижеследующей таблице 1. В ней указаны ошибки классификации на тестовой части выборки в процентах с точностью до сотых долей. В графе, соответствующей коэволюционному методу, в скобках приведено среднее число алгоритмов в композиции. Напомним, что при коэволюции используется «правило 1-3», то есть нет композиций мощности 2. Иллюстрации, демонстрирующие типичные примеры работы различных методов для указанных 4 задач, приведены на рисунках 4 — 7.

Прокомментируем полученные результаты. Во-первых, во всех задачах баггинг работает практически идентично одному алгоритму, что соответствует предположениям (баггинг снижает вариацию, а правило Байеса — чрезвычайно стабильный метод). Метод случайных подпространств также работает практически идентично одному правилу Байеса, что объясняется всего двумя признаками в описании объекта и нормальным распределением выборок по каждой из координат. Бустинг не всегда работает стабильно, иногда приближаясь к оптимальному разбиению, иногда — выдавая наихудшие результаты. В задачах `Gauss-1` и `Gauss-4` приведены по две иллюстрации работы бустинга, так как получаемые бустингом результаты делятся на две принципиально различные группы. Для наиболее сложной задачи `Gauss-3` ни один из методов не нашел адекватного решения (что наглядно представлено на ил-

Задача	**Метод Байеса**	**Коэв. метод**	**Бустинг**	**Баггинг**	**RSM**
`Gauss-1`	16.60	3.47 (1.06)	22.55	16.47	16.40
`Gauss-2`	27.20	17.13 (1.69)	27.05	27.22	27.23
`Gauss-3`	29.16	16.12 (2.2)	30.31	29.18	29.86
`Gauss-4`	4.39	1.38 (1.99)	20.56	4.38	4.50

Таблица 1: Результаты расчетов, произведенных для различных методов построения алгоритмических композиций на модельных задачах. Приведены проценты ошибок на тестовой выборке. Для коэволюционного метода в скобках указано среднее число алгоритмов в композиции.

люстрации). Общим результатом является заметное превосходство коэволюционного метода, что объясняется плохой применимостью бустинга и баггинга к стабильным методам классификации.

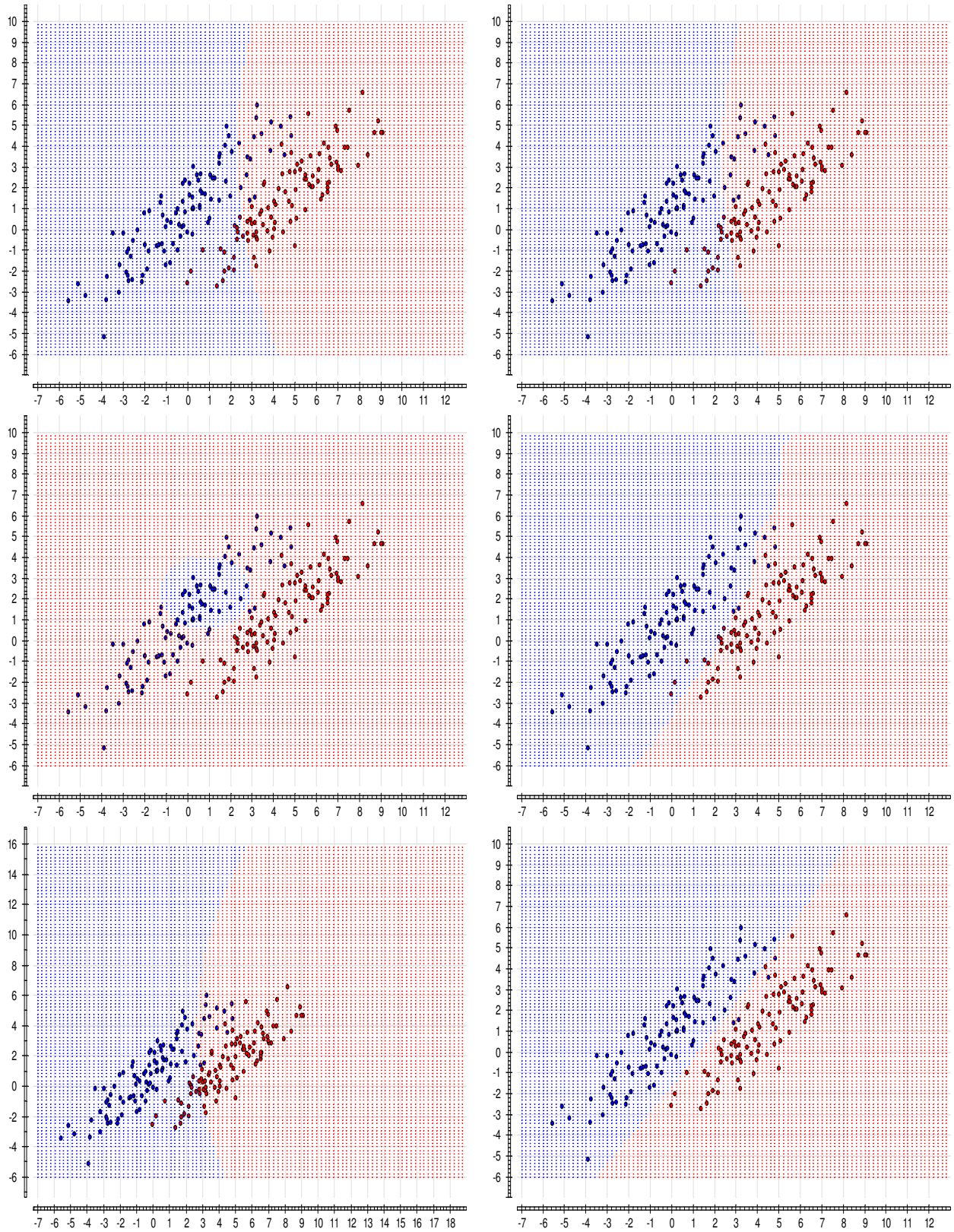


Рис. 4: Работа различных алгоритмов для задачи `Gauss-1`. Сверху: слева один базовый алгоритм, справа баггинг (250 алгоритмов). В середине: Две типичные картины работы бустинга (250 алгоритмов). Внизу: слева метод случайных подпространств (250 алгоритмов), справа коэволюционный метод (в среднем 1.06 алгоритмов).

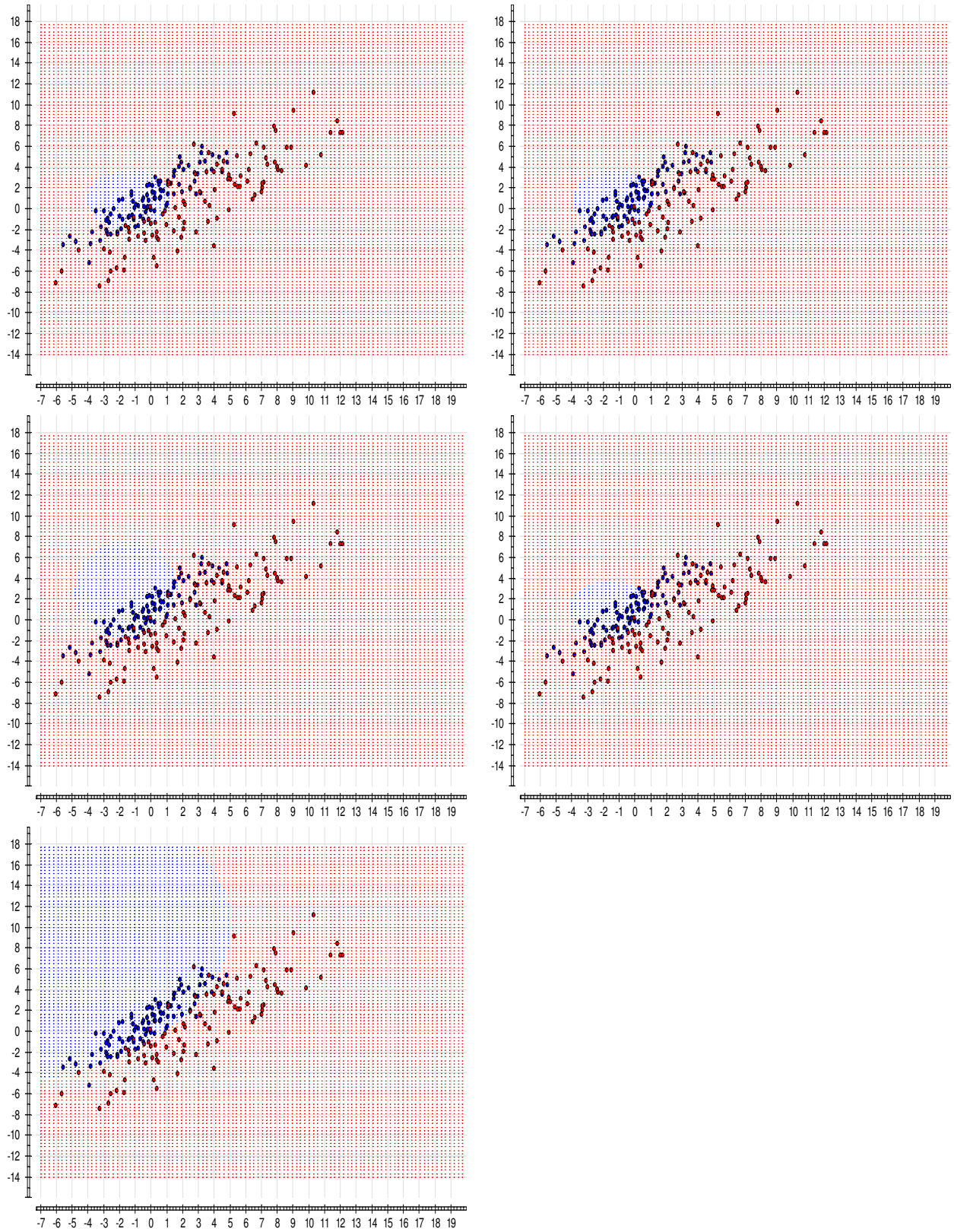


Рис. 5: Работа различных алгоритмов для задачи `Gauss-2`. Сверху: слева один базовый алгоритм, справа баггинг (250 алгоритмов). В середине: слева бустинг (250 алгоритмов), справа метод случайных подпространств (250 алгоритмов). Внизу: коэволюционный метод (в среднем 1.69 алгоритмов).

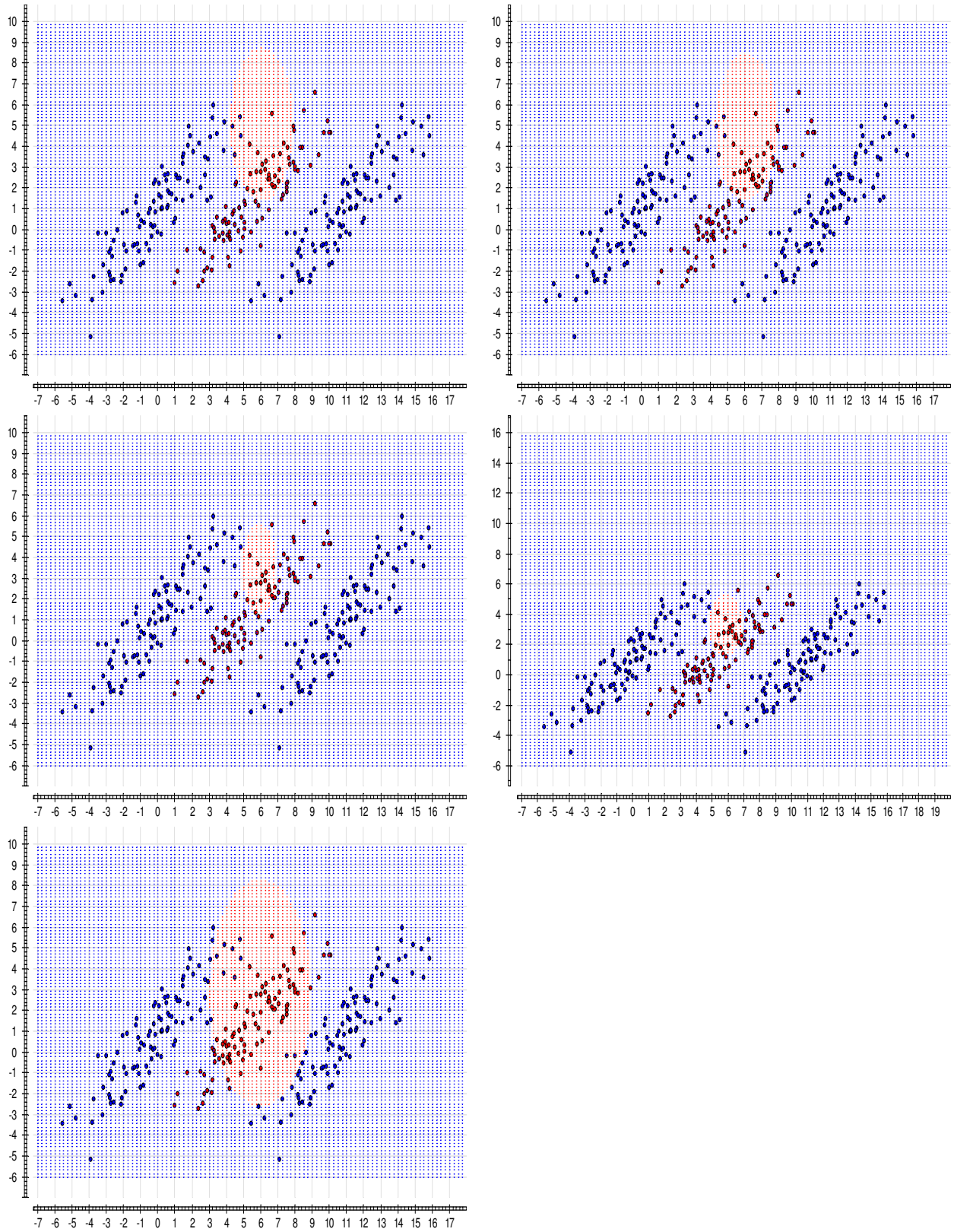


Рис. 6: Работа различных алгоритмов для задачи `Gauss-3`. Сверху: слева один базовый алгоритм, справа баггинг (250 алгоритмов). В середине: слева бустинг (250 алгоритмов), справа метод случайных подпространств (250 алгоритмов). Внизу: коэволюционный метод (в среднем 2.2 алгоритмов).

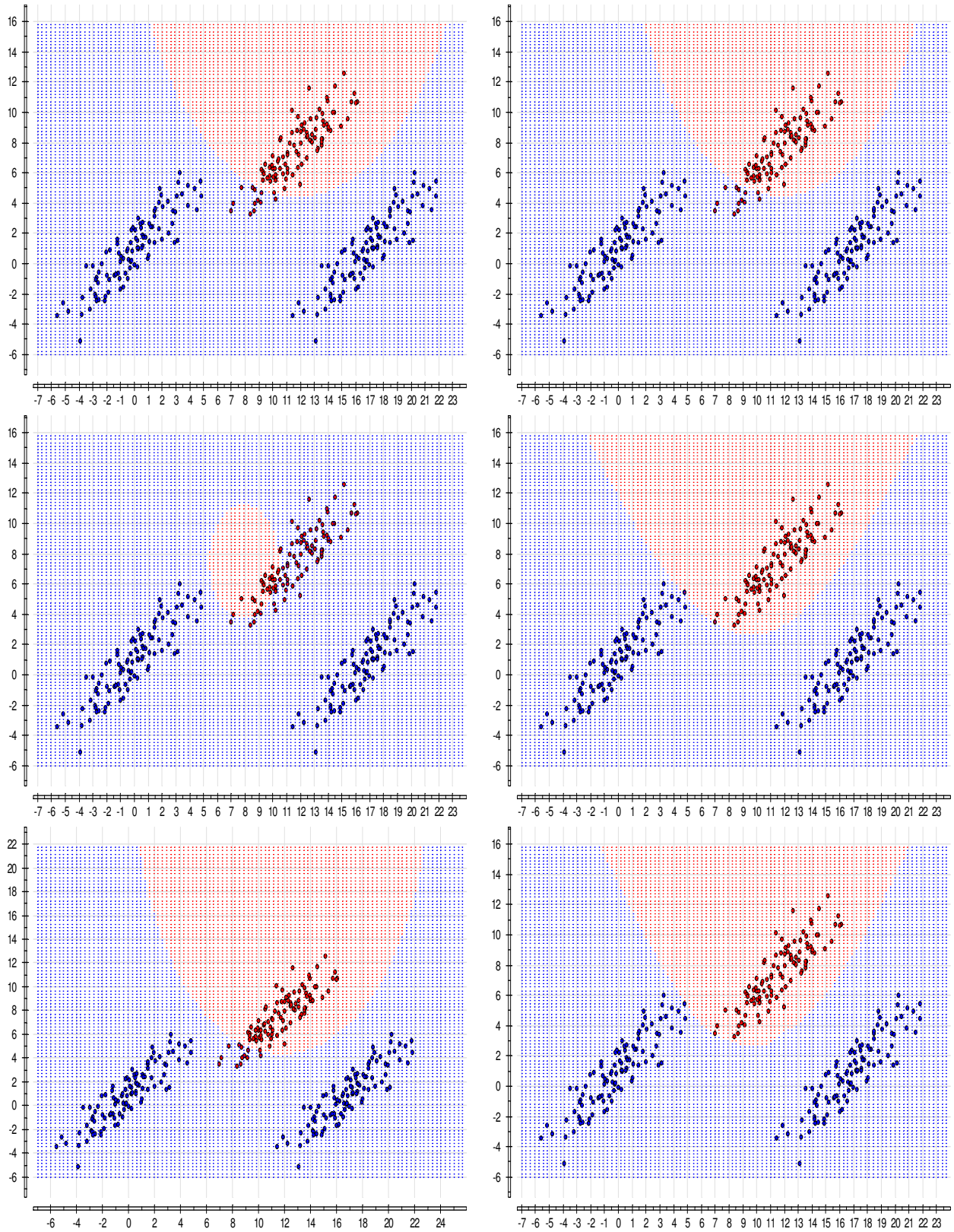


Рис. 7: Работа различных алгоритмов для задачи `Gauss-1`. Сверху: слева один базовый алгоритм, справа баггинг (250 алгоритмов). В середине: Две типичные картины работы бустинга (250 алгоритмов). Внизу: слева метод случайных подпространств (250 алгоритмов), справа коэволюционный метод (в среднем 1.99 алгоритмов).

Эксперименты на реальных данных

Для проведения экспериментов на реальных данных были выбраны 12 задач с двумя классами из репозитория UCI [27]. Их характеристики приведены в нижеследующей таблице 2.

Задача	**Объектов**	**Признаков**	**Непр.**	**Дискр.**
Breast Cancer	699	9	9	0
Credit(Australian) - 1	690	14	6	8
Credit (Australian) - 2	690	15	6	9
Credit (German)	1000	24	24	0
Diagnostic Breast Cancer (DBC)	569	30	30	0
Heart Disease	303	13	13	0
Hepatitis	155	19	6	13
Liver Disorders	345	6	6	0
Pima Indians Diabetes	768	8	8	0
Survival	306	3	3	0
Tic-tac-toe game	958	9	0	9
Voting records	435	16	0	16

Таблица 2: Характеристики реальных задач, для которых производилось тестирование и сравнительный анализ коэволюционного метода.

Подобно экспериментам на модельных задачах, коэволюционный метод сравнивается с одним правилом Байеса, бустингом, баггингом и методом случайных подпространств (RSM). Для этих методов построения композиции число алгоритмов принято равным 250. В таблице 3 собраны результаты расчетов для указанных задач. Таблица содержит проценты ошибок на тестовой выборке с точностью до сотых долей. Для коэволюционного алгоритма в скобках указано среднее число алгоритмов в композиции. Напомним, что в коэволюционном методе не использовались композиции мощности 2, то есть наращивание происходило с одного до трех алгоритмов («правило 1-3»).

Результаты, представленные в таблице 3, дают материал для следующих наблюдений.

- Для выбранного базового семейства алгоритмов коэволюционный метод существенно превосходит остальные приведенные методы. В 10 задачах из 12 он показывает наилучшие результаты. В задаче Liver Disorders коэволюция проигрывает только бустингу, а в задаче Hepatitis ни один из методов построения ансамблей классификаторов не улучшает результатов одного правила Байеса.
- Похожий по заложенным идеям на баггинг и метод случайных подпространств, коэволюционный метод работает принципиально иначе. Так, он позволяет существенно понизить ошибку байесовского классификатора, который является одним из самых стабильных алгоритмов, для набора различных задач. Этот факт позволяет предположить, что, в отличие от баггинга и RSM, предлагаемый метод способствует уменьшению не столько вариации, сколько смещения, что характерно для бустинга. При этом коэволюционный метод работает го-

Задача	**Баз. алгоритм**	**Коэв. метод**	**Бустинг**	**Баггинг**	**RSM**
Cancer	5.48	**4.0** (3.95)	4.44	5.63	5.05
Credit-a-1	15.30	**13.37** (4.52)	23.96	15.28	14.78
Credit-a-2	15.87	**14.79** (5.23)	18.88	15.86	15.95
Credit-g	32.11	**25.42** (2.25)	30.01	31.60	32.29
DBC	11.43	**5.61** (5.0)	23.61	11.35	11.44
Heart	19.52	**17.59** (3.65)	22.79	19.53	19.26
Hepatitis	**16.51**	17.64 (3.08)	20.17	16.60	16.57
Liver	37.06	35.59 (1.9)	**34.40**	37.03	36.84
Diabetes	31.80	**23.65** (3.38)	31.62	31.66	30.87
Survival	30.99	**26.45** (2.66)	27.11	29.52	28.02
Tic-tac-toe	28.92	**20.29** (1.95)	33.17	28.96	27.78
Voting	6.51	**5.17** (1.77)	7.1	6.55	6.33

Таблица 3: Результаты расчетов, произведенных для набора задач из таблицы 2. Приведены проценты ошибок на тестовой выборке. Для коэволюционного метода в скобках указано среднее число алгоритмов в композиции.

раздо стабильнее бустинга, в большинстве рассмотренных задач ухудшающего качество классификации.

- Результаты работы бустинга и баггинга соответствуют предположениям и коррелируют с известными об этих алгоритмах фактами (см. раздел 2). Так, бустинг и баггинг не улучшают качество работы правила Байеса, так как последнее относится к стабильным алгоритмам. Баггинг почти не ухудшает качество классификации, но и не дает значительных улучшений ни для одной из задач. Бустинг в большинстве задач выдает бóльшую ошибку классификации, но в тех задачах, где имеет место улучшение, оно является существенным.
- Метод случайных подпространств работает с правилами Байеса в целом аналогично баггингу, получая лучшие результаты для задач с малым числом признаков.

Итак, результатом проведенных экспериментов является существенное лучшее качество классификации алгоритмов, построенных с помощью коэволюционного метода в большинстве рассмотренных задач. Тем не менее, это не позволяет сделать вывод о превосходстве коэволюционного метода над остальными. Как уже говорилось выше, базовое семейство алгоритмов было выбрано таким образом, что баггинг и бустинг работают на нем заведомо плохо. Рассматриваемые задачи имели достаточно большой объем выборки по отношению к числу признаков, поэтому метод случайных подпространств также не имел заведомого преимущества. Эксперименты с правилами Байеса позволили проверить возможности предлагаемого коэволюционного подхода и сделать некоторые предположения о его воздействии, но они не являются основанием для оценки качества нового метода по отношению к существующим. Для получения более точного и развернутого представления о возможностях предлагаемого коэволюционного метода синтеза алгоритмических композиций требуется целый ряд как эмпирических, так и аналитико-теоретических исследований.

5 Заключение

Данная работа посвящена описанию и исследованию возможностей нового метода построения композиций алгоритмов, использующего модель кооперативной коэволюции для глобальной оптимизации каждого из алгоритмов композиции. Был описан подход к построению алгоритма из $\mathfrak{F}$-расширения семейства алгоритмических операторов $\mathfrak{M}^0$ для произвольных $\mathfrak{M}^0$ и $\mathfrak{F}$. Подробно метод изложен на примере семейства линейных корректирующих операций и задачи классификации с двумя классами. Проведен ряд экспериментов на модельных и реальных задачах, для которых в качестве семейства базовых алгоритмов $\mathfrak{M}^0$ было выбрано правило Байеса.

5.1 Выводы

Указанные в разделе 4.2 достоинства предложенного метода вкупе с результатами экспериментов позволяют говорить о применимости и эффективности коэволюционного подхода к построению композиций алгоритмов классификации. Тот факт, что в рамках выбранного семейства базовых алгоритмов предложенный метод для большей части тестовых задач показал наилучшие среди рассмотренных методов комбинирования результаты, позволяет также назвать перспективным дальнейшее исследование и развитие данного подхода. Существенное улучшение качества классификации по сравнению с одним алгоритмом Байеса в большинстве случаев позволяет предположить, что коэволюционный алгоритм способствует уменьшению смещения, что является определяющим при построении композиций стабильных алгоритмов.

Однако полученные в рамках экспериментов результаты не имеют непосредственной практической ценности, так как качество классификации композиций правил Байеса существенно ниже, нежели композиций на основе нейронных сетей и деревьев решений [45, 30, 25, 34, 95]. Также в последние несколько лет на основе существующих подходов были разработаны более совершенные методы построения композиций классификаторов, для которых не произведено сравнения с предлагаемым методом [43, 76, 68, 72, 54, 94, 95]. Поэтому, принимая во внимание перспективность коэволюционного подхода, говорить о его реальных возможностях и преимуществах пока рано. Эти и другие вопросы являются наиболее актуальными объектами будущих исследований.

5.2 Направления дальнейшей работы

Итак, набор параметров предлагаемого метода следует считать ориентировочным, а результаты экспериментов — предварительными. Среди основных направлений дальнейших исследований можно выделить следующие группы:

1. Применение метода для различных семейств алгоритмических операторов и корректирующих операций. Среди «простых» семейств $\mathfrak{M}^0$ алгоритмов, скорость настройки которых позволяет использовать предлагаемый метод без значительных модификаций, можно выделить методы оптимальной разделяющей гиперплоскости, k ближайших соседей, потенциальных функций. В качестве корректирующих операций, следуя проблемно-ориентированным методами алгебраического подхода (см. 2.1), можно рассмотреть полиномиальные и монотонные

семейства корректоров [1, 3, 4, 16].

2. Эмпирический анализ работы алгоритма с целью определения влияния параметров на качество результирующей композиции. Данное направление работы включает проведение экспериментов с варьированием различных параметров алгоритма. В качестве наиболее интересных экспериментов такого типа укажем сравнение различных критериев оптимизации (отступы, разнообразие, частота ошибок), а также различные правила составления коллабораций.

3. Эмпирико-аналитический анализ эволюционной динамики с целью выявления механизма и особенностей специализации алгоритмов композиции. Предполагается анализ состава популяций на различных итерациях, составление модельных задач для проверки выдвигаемых гипотез и т.д. Результатом такой работы может стать разработка дополнительного регуляторного механизма специализации, направляющего алгоритмы в различные «ниши», или области компетентности.

4. Адаптация метода для построения композиций более сложных и эффективных методов классификации и проведение экспериментов на реальных задачах. В первую очередь, речь идет о деревьях решений и нейронных сетях, являющихся эффективными нестабильными методами. Эта их особенность допускает существенное улучшение качества классификации с помощью таких методик, как бустинг. С другой стороны, время настройки алгоритмов данных семейств существенно превышает таковое для правил Байеса, что может послужить серьезным препятствием для применения к ним коэволюционного метода. Данное направление работы подразумевает проверку данного предположения и, в случае его подтверждения, оптимизацию работы коэволюции по скорости. Одним из возможных путей такой оптимизации является распараллеливание работы алгоритма.

5. Разработка механизмов «обратной связи», позволяющих использовать побочные результаты работы алгоритма для улучшения его работы. Как говорилось выше, анализ динамики и результатов работы предлагаемого метода может способствовать выделению наиболее информативных объектов и признаков, указывать наиболее эффективные стратегии развития популяций и т.д. Разработка и реализация способов накапливания, анализа и использования такой информации может способствовать как развитию данного подхода, так и решению задачи с помощью других методов.

6. Анализ возможностей модели кооперативной коэволюции применительно к задаче построения композиций алгоритмов классификации. Подобно работам [85, 87, 57], в которых анализируется применимость и эффективность кооперативной коэволюции для оптимизации функций многих переменных, данное направление исследований является поиском ответов на аналогичные вопросы для задачи построения алгоритмических композиций. Насколько эффективной является проводимая в рамках данной модели оптимизация? Какие ограничения накладывает данная модель на пространство поиска, в данном случае $\mathfrak{F}(\mathfrak{M}^0)$? Когда приме-

нение данного подхода оправданно? Эти и другие вопросы являются стимулом дальнейшего изучения динамики эволюционных и коэволюционных алгоритмов, в особенности применительно к задачам классификации.

Список литературы

[1] *Воронцов К. В.* О проблемно-ориентированной оптимизации базисов задач распознавания // *ЖВМ и МФ.* — 1998. — Т. 38, № 5. — С. 870–880.
http://www.ccas.ru/frc/papers/voron98jvm.pdf.

[2] *Воронцов К. В.* Локальные базисы в алгебраическом подходе к проблеме распознавания. — Диссертация на соискание учёной степени к.ф.-м.н., М.: ВЦ РАН. — 1999.
http://www.ccas.ru/frc/thesis/VoronCanDisser.pdf.

[3] *Воронцов К. В.* Оптимизационные методы линейной и монотонной коррекции в алгебраическом подходе к проблеме распознавания // *ЖВМ и МФ.* — 2000. — Т. 40, № 1. — С. 166–176.
http://www.ccas.ru/frc/papers/voron00jvm.pdf.

[4] *Воронцов К. В.* Проблемно-ориентированные методы алгебраического подхода. — 2002.
http://www.ccas.ru/frc/papers/voron02po4.pdf.

[5] *Воронцов К. В.* Обзор современных исследований по проблеме качества обучения алгоритмов // *Таврический вестник информатики и математики.* — 2004.
http://www.ccas.ru/frc/papers/voron04twim.pdf.

[6] *Журавлёв Ю. И.* Корректные алгебры над множествами некорректных (эвристических) алгоритмов. часть I // *Кибернетика.* — 1977. — № 4. — С. 5–17.

[7] *Журавлёв Ю. И.* Корректные алгебры над множествами некорректных (эвристических) алгоритмов. часть II // *Кибернетика.* — 1977. — № 6. — С. 21–27.

[8] *Журавлёв Ю. И.* Корректные алгебры над множествами некорректных (эвристических) алгоритмов. часть III // *Кибернетика.* — 1978. — № 2. — С. 35–43.

[9] *Журавлёв Ю. И.* Об алгебраическом подходе к решению задач распознавания или классификации // *Проблемы кибернетики.* — 1978. — Т. 33. — С. 5–68.

[10] *Рудаков К. В.* О некоторых универсальных ограничениях для алгоритмов классификации // *ЖВМиМФ.* — 1986. — Т. 26, № 11. — С. 1719–1730.
http://www.ccas.ru/frc/papers/rudakov86universal.pdf.

[11] *Рудаков К. В.* О симметрических и функциональных ограничениях для алгоритмов классификации // *ДАН СССР.* — 1987. — Т. 297, № 1. — С. 43–46.
http://www.ccas.ru/frc/papers/rudakov87dan.pdf.

[12] *Рудаков К. В.* Полнота и универсальные ограничения в проблеме коррекции эвристических алгоритмов классификации // *Кибернетика.* — 1987. — № 3. — С. 106–109.

[13] *Рудаков К. В.* Универсальные и локальные ограничения в проблеме коррекции эвристических алгоритмов // *Кибернетика.* — 1987. — № 2. — С. 30–35.
http://www.ccas.ru/frc/papers/rudakov87universal.pdf.

[14] *Рудаков К. В.* Об алгебраической теории универсальных и локальных ограничений для задач классификации // *Распознавание, Классификация, Прогноз.* — 1988. — Т. 1. — С. 176–200.
http://www.ccas.ru/frc/papers/rudakov88rkp.pdf.

[15] *Рудаков К. В.* Алгебраическая теория универсальных и локальных ограничений для алгоритмов распознавания. — Диссертация на соискание учёной степени д.ф.-м.н., М.: ВЦ РАН. — 1992.
http://www.ccas.ru/frc/thesis/RudakovDocDisser.pdf.

[16] *Рудаков К. В., Воронцов К. В.* О методах оптимизации и монотонной коррекции в алгебраическом подходе к проблеме распознавания // *Докл. РАН.* — 1999. — Т. 367, № 3. — С. 314–317.
http://www.ccas.ru/frc/papers/rudvoron99dan.pdf.

[17] *Рудаков К. В., Трофимов С. В.* Алгоритм синтеза корректных процедур распознавания для задач с непересекающимися классами // *ЖВМиМФ.* — 1988. — Т. 28, № 9. — С. 1431–1434.
http://www.ccas.ru/frc/papers/rudakov88trofim.pdf.

[18] *Angeline P. J.* Adaptive and self-adaptive evolutionary computations // Computational Intelligence: A Dynamic Systems Perspective / Ed. by M. Palaniswami, Y. Attikiouzel. — IEEE Press, 1995. — Pp. 152–163.
http://citeseer.ist.psu.edu/angeline95adaptive.html.

[19] *Angeline P. J., Pollack J. B.* Competitive environments evolve better solutions for complex tasks // Proceedings of the 5th International Conference on Genetic Algorithms (GA-93). — 1993. — Pp. 264–270.
http://citeseer.ist.psu.edu/angeline93competitive.html.

[20] *Back T.* Self-adaptation in genetic algorithms.
http://citeseer.ist.psu.edu/14572.html.

[21] *Back T.* Optimization by means of genetic algorithms // 36th International Scientific Colloquium / Ed. by E. Kohler. — Technical University of Ilmenau: 1991. — Pp. 163–169.
http://citeseer.ist.psu.edu/71967.html.

[22] *Back T., Hoffmeister F.* Global optimization by means of evolutionary algorithms.
http://citeseer.ist.psu.edu/219671.html.

[23] *Back T., Hoffmeister F., Schwefel H.* A survey of evolution strategies. — 1991.
http://citeseer.ist.psu.edu/back91survey.html.

[24] *Baker J. E.* Adaptive Selection Methods for Genetic Algorithms / Ed. by P. of an International Conference on Genetic Algorithms, e. their Applications (J. J. Grefenstette. — Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

[25] *Bauer E., Kohavi R.* An empirical comparison of voting classification algorithms: Bagging, boosting, and variants // *Machine Learning.* — 1999. — Vol. 36, no. 1-2. — Pp. 105–139.
http://citeseer.ist.psu.edu/bauer99empirical.html.

[26] *Bhanu B., Krawiec K.* Coevolutionary construction of features for transformation of representation in machine learning.
http://citeseer.ist.psu.edu/krawiec02coevolutionary.html.

[27] *Blake C., Merz C.* UCI repository of machine learning databases: Tech. rep.: Department of Information and Computer Science, University of California, Irvine,

CA, 1998.
http://www.ics.uci.edu/~mlearn/MLRepository.html.

[28] *Breiman L.* Bagging predictors // *Machine Learning.* — 1996. — Vol. 24, no. 2. — Pp. 123–140.
http://citeseer.ist.psu.edu/breiman96bagging.html.

[29] *Breiman L.* Bias, variance, and arcing classifiers: Tech. Rep. 460: Statistics Department, University of California, 1996.
http://citeseer.ist.psu.edu/breiman96bias.html.

[30] *Breiman L.* Arcing classifiers // *The Annals of Statistics.* — 1998. — Vol. 26, no. 3. — Pp. 801–849.
http://citeseer.ist.psu.edu/breiman98arcing.html.

[31] *Bucci A., Pollack J.* Order-theoretic analysis of coevolution problems: Coevolutionary statics. — 2002.
http://citeseer.ist.psu.edu/article/bucci02ordertheoretic.html.

[32] *Bucci A., Pollack J.* A mathematical framework for the study of coevolution. — 2003.
http://citeseer.ist.psu.edu/bucci03mathematical.html.

[33] *Busetti F.* Genetic algorithms overview.
http://citeseer.ist.psu.edu/busetti01genetic.html.

[34] *Dietterich T. G.* An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization // *Machine Learning.* — 2000. — Vol. 40, no. 2. — Pp. 139–157.
http://citeseer.ist.psu.edu/article/dietterich98experimental.html.

[35] *Domingos P.* Why does bagging work? a bayesian account and its implications // Knowledge Discovery and Data Mining. — 1997. — Pp. 155–158.
http://citeseer.ist.psu.edu/21089.html.

[36] *Dymitr Ruta B. G.* Analysis of the correlation between majority voting error and the diversity.
http://citeseer.ist.psu.edu/705285.html.

[37] *Ficici S. G., Melnik O., Pollack J. B.* A game-theoretic investigation of selection methods used in evolutionary algorithms // Proceedings of the 2000 Congress on Evolutionary Computation CEC00. — La Jolla Marriott Hotel La Jolla, California, USA: IEEE Press, 6-9 2000. — P. 880.
http://citeseer.ist.psu.edu/ficici00gametheoretic.html.

[38] *Ficici S. G., Pollack J. B.* A game-theoretic approach to the simple coevolutionary algorithm // Parallel Problem Solving from Nature — PPSN VI 6th International Conference / Ed. by H.-P. S. Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo. — Paris, France: Springer Verlag, 16-20 2000.
http://citeseer.ist.psu.edu/322969.html.

[39] *Ficici S. G., Pollack J. B.* Pareto optimality in coevolutionary learning // *Lecture Notes in Computer Science.* — 2001. — Vol. 2159. — Pp. 316+.
http://citeseer.ist.psu.edu/article/ficici01pareto.html.

[40] *Fidelis M. V., Lopes H. S., Freitas A. A.* Discovering comprehensible classification rules a genetic algorithm // Proceedings of the 2000 Congress on Evolutionary Computation CEC00. — La Jolla Marriott Hotel La Jolla, California, USA: IEEE Press, 6-9 2000. — Pp. 805–810.

http://citeseer.ist.psu.edu/fidelis00discovering.html.

[41] *Freitas A.* A survey of evolutionary algorithms for data mining and knowledge discovery. — 2001.
http://citeseer.ist.psu.edu/freitas01survey.html.

[42] *Freund Y.* Boosting a weak learning algorithm by majority // COLT: Proceedings of the Workshop on Computational Learning Theory. — Morgan Kaufmann Publishers, 1990.
http://citeseer.ist.psu.edu/freund95boosting.html.

[43] *Freund Y.* An adaptive version of the boost by majority algorithm // COLT: Proceedings of the Workshop on Computational Learning Theory. — Morgan Kaufmann Publishers, 1999.
http://citeseer.ist.psu.edu/article/freund00adaptive.html.

[44] *Freund Y., Schapire R. E.* A decision-theoretic generalization of on-line learning and an application to boosting // European Conference on Computational Learning Theory. — 1995. — Pp. 23–37.
http://citeseer.ist.psu.edu/article/freund95decisiontheoretic.html.

[45] *Freund Y., Schapire R. E.* Experiments with a new boosting algorithm // International Conference on Machine Learning. — 1996. — Pp. 148–156.
http://citeseer.ist.psu.edu/freund96experiments.html.

[46] *Freund Y., Schapire R. E.* A short introduction to boosting // *J. Japan. Soc. for Artif. Intel.* — 1999. — Vol. 14, no. 5. — Pp. 771–780.
http://citeseer.ist.psu.edu/freund99short.html.

[47] *Giordana A., Neri F.* Search-intensive concept induction // *Evolutionary Computation.* — 1995. — Vol. 3, no. 4. — Pp. 375–419.
http://citeseer.ist.psu.edu/article/giordana95searchintensive.html.

[48] *Grove A. J., Schuurmans D.* Boosting in the limit: Maximizing the margin of learned ensembles // AAAI/IAAI. — 1998. — Pp. 692–699.
http://citeseer.ist.psu.edu/grove98boosting.html.

[49] *Hancock P. J. B.* An empirical comparison of selection methods in evolutionary algorithms // Evolutionary Computing, AISB Workshop. — 1994. — Pp. 80–94.
http://citeseer.ist.psu.edu/hancock94empirical.html.

[50] *Hinterding R., Gielewski H., Peachey T. C.* The nature of mutation in genetic algorithms // Proceedings of the Sixth International Conference on Genetic Algorithms / Ed. by L. Eshelman. — San Francisco, CA: Morgan Kaufmann, 1995. — Pp. 65–72.
http://citeseer.ist.psu.edu/hinterding95nature.html.

[51] *Ho T. K.* The random subspace method for constructing decision forests // *IEEE Transactions on Pattern Analysis and Machine Intelligence.* — 1998. — Vol. 20, no. 8. — Pp. 832–844.
http://citeseer.ist.psu.edu/ho98random.html.

[52] *Horn J.* Finite Markov Chain Analysis of Genetic Algorithms with Niching // Proceedings of the Fifth International Conference on Genetic Algorithms / Ed. by S. Forrest. — San Mateo, California: Morgan Kaufmann Publishers, 1993. — Pp. 110–117.
http://citeseer.ist.psu.edu/article/horn93finite.html.

[53] *J.H. H.* Adaptation in Natural and Artificial Systems. — University of Michigan

Press, 1975.

[54] *Jin R., Liu Y., Si L., Carbonell J., Hauptmann A.* A new boosting algorithm using input-dependent regularizer // The 20-th International Conference on Machine Learning. — 2003.
http://citeseer.ist.psu.edu/jin03new.html.

[55] *Kuncheva L., Skurichina M., Duin R.* An experimental study on diversity for bagging and boosting with linear classifiers. — 2002.
http://citeseer.ist.psu.edu/kuncheva02experimental.html.

[56] *Kuncheva L., Whitaker C.* Measures of diversity in classifier ensembles. — 2000.
http://citeseer.ist.psu.edu/kuncheva00measures.html.

[57] *Luke S., Wiegand R. P.* When coevolutionary algorithms exhibit evolutionary dynamics // *In Workshop Proceedings of the 2003 Genetic and Evolutionary Computation Conference.* — 2002.

[58] *Maclin R., Opitz D.* An empirical evaluation of bagging and boosting // AAAI/IAAI. — 1997. — Pp. 546–551.
http://citeseer.ist.psu.edu/maclin97empirical.html.

[59] *Mason L.* Margins and Combined Classifiers: Ph.D. thesis / Australian National University. — 1999.

[60] *Paredis J.* Coevolutionary computation // *Artificial Life.* — 1995. — Vol. 2, no. 4. — Pp. 355–375.
http://citeseer.ist.psu.edu/158996.html.

[61] *Pena-Reyes C. A.* Coevolutionary fuzzy modeling.
http://citeseer.ist.psu.edu/569226.html.

[62] *Pena-Reyes C. A.* Island fuzzy coco: Island model-based coevolution of fuzzy systems.
http://citeseer.ist.psu.edu/568976.html.

[63] *Potter M. A.* The Design and Analysis of a Computational Model of Cooperative Coevolution: Ph.D. thesis. — 1997.
http://citeseer.ist.psu.edu/potter97design.html.

[64] *Potter M. A., De Jong K.* A cooperative coevolutionary approach to function optimization // Parallel Problem Solving from Nature – PPSN III / Ed. by Y. Davidor, H.-P. Schwefel, R. Männer. — Berlin: Springer, 1994. — Pp. 249–257.
http://citeseer.ist.psu.edu/potter94cooperative.html.

[65] *Potter M. A., De Jong K.* Evolving neural networks with collaborative species // Proc. of the 1995 Summer Computer Simulation Conf. — The Society of Computer Simulation, 1995. — Pp. 340–345.
http://citeseer.ist.psu.edu/potter95evolving.html.

[66] *Potter M. A., De Jong K. A.* Cooperative coevolution: An architecture for evolving coadapted subcomponents // *Evolutionary Computation.* — 2000. — Vol. 8, no. 1. — Pp. 1–29.
http://citeseer.ist.psu.edu/potter00cooperative.html.

[67] *Radcliffe N. J.* The algebra of genetic algorithms: Tech. Rep. TR92-11: 1992.
http://citeseer.ist.psu.edu/article/radcliffe94algebra.html.

[68] *Ratsch G., Onoda T., Muller K. R.* An improvement of adaboost to avoid overfitting.
http://citeseer.ist.psu.edu/6344.html.

[69] *Ratsch G., Onoda T., Muller K.-R.* Soft margins for AdaBoost // *Machine Learning.* — 2001. — Vol. 42, no. 3. — Pp. 287–320.

http://citeseer.ist.psu.edu/ratsch00soft.html.

[70] *Rosin C. D., Belew R. K.* New methods for competitive coevolution // *Evolutionary Computation.* — 1997. — Vol. 5, no. 1. — Pp. 1–29.
http://citeseer.ist.psu.edu/rosin96new.html.

[71] *Ruta D., Gabrys B.* Classifier selection for majority voting.
http://citeseer.ist.psu.edu/699534.html.

[72] *Saharon Rosset J. Z., Hastie T.* Margin maximizing loss functions.
http://www.research.ibm.com/dar/papers/pdf/margmax1.pdf.

[73] *Schapire R.* The boosting approach to machine learning: An overview // MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA. — 2001.
http://citeseer.ist.psu.edu/schapire02boosting.html.

[74] *Schapire R. E.* Theoretical views of boosting and applications // Algorithmic Learning Theory, 10th International Conference, ALT 99, Tokyo, Japan, December 1999, Proceedings. — Vol. 1720. — Springer, 1999. — Pp. 13–25.
http://citeseer.ist.psu.edu/article/schapire99theoretical.html.

[75] *Schapire R. E., Freund Y., Lee W. S., Bartlett P.* Boosting the margin:a new explanation for the effectiveness of voting methods // *Annals of Statistics.* — 1998. — Vol. 26, no. 5. — Pp. 1651–1686.
http://citeseer.ist.psu.edu/article/schapire98boosting.html.

[76] *Schapire R. E., Singer Y.* Improved boosting using confidence-rated predictions // *Machine Learning.* — 1999. — Vol. 37, no. 3. — Pp. 297–336.
http://citeseer.ist.psu.edu/article/singer99improved.html.

[77] *Skurichina M., Duin R. P. W.* Limited bagging, boosting and the random subspace method for linear classifiers.
http://citeseer.ist.psu.edu/skurichina02limited.html.

[78] *Skurichina M., Kuncheva L., Duin R.* Bagging and boosting for the nearest mean classifier: Effects of sample size on diversity and accuracy // Multiple Classifier Systems (Proc. Third International Workshop MCS, Cagliari, Italy) / Ed. by J. K. F. Roli. — Vol. 2364. — Springer, Berlin, 2002. — Pp. 62–71.
http://citeseer.ist.psu.edu/539135.html.

[79] *Sloman A.* The semantics of evolution: Trajectories and trade-offs in design space and niche space // *Lecture Notes in Computer Science.* — 1998. — Vol. 1484. — Pp. 27–??
http://citeseer.ist.psu.edu/75107.html.

[80] *Tsymbal A., Puuronen S.* Ensemble feature selection with the simple bayesian classification in medical diagnostics.
http://citeseer.ist.psu.edu/623033.html.

[81] *Twycross J., Cayzer S.* An immune-based approach to document classification.
http://citeseer.ist.psu.edu/twycross02immunebased.html.

[82] *Whitley D.* The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best // Proceedings of the Third International Conference on Genetic Algorithms / Ed. by J. D. Schaffer. — San Mateo, CA: Morgan Kaufman, 1989.
http://citeseer.ist.psu.edu/whitley89genitor.html.

[83] *Whitley D.* An overview of evolutionary algorithms: practical issues and common pitfalls // *Information and Software Technology.* — 2001. — Vol. 43, no. 14. — Pp. 817–831.

http://citeseer.ist.psu.edu/whitley01overview.html.

[84] *Whitley D., Rana S. B., Heckendorn R. B.* Island model genetic algorithms and linearly separable problems // Evolutionary Computing, AISB Workshop. — 1997. — Pp. 109–125.
http://citeseer.ist.psu.edu/whitley97island.html.

[85] *Wiegand R., Jansen T.* The cooperative coevolutionary (1+1) ea // *MIT Press.* — 2003.

[86] *Wiegand R., Liles W., De Jong K.* Analyzing cooperative coevolution with evolutionary game theory. — 2002.
http://citeseer.ist.psu.edu/wiegand02analyzing.html.

[87] *Wiegand R. P.* An Analysis of Cooperative Coevolutionary Algorithms: Ph.D. thesis / George Mason University, Fairfax, VA. — 2004.
http://http://www.tesseract.org/paul/papers/rpw-dissertation.pdf.

[88] *Wiegand R. P., Liles W. C., De Jong K. A.* An empirical analysis of collaboration methods in cooperative coevolutionary algorithms // *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO).* — 2001.

[89] *Wright A. H.* Genetic algorithms for real parameter optimization // Foundations of genetic algorithms / Ed. by G. J. Rawlins. — San Mateo, CA: Morgan Kaufmann, 1991. — Pp. 205–218.
http://citeseer.ist.psu.edu/article/wright91genetic.html.

[90] *Wright A. H., Zhao Y.* Markov chain models of genetic algorithms // Proceedings of the Genetic and Evolutionary Computation Conference / Ed. by W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, R. E. Smith. — Vol. 1. — Orlando, Florida, USA: Morgan Kaufmann, 13-17 1999. — Pp. 734–741.
http://citeseer.ist.psu.edu/wright99markov.html.

[91] *Yang J., Honavar V.* Feature subset selection using A genetic algorithm // Genetic Programming 1997: Proceedings of the Second Annual Conference / Ed. by J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, R. L. Riolo. — Stanford University, CA, USA: Morgan Kaufmann, 13-16 1997. — P. 380.
http://citeseer.ist.psu.edu/article/yang97feature.html.

[92] *Zheng Z., Webb G.* Multiple boosting: A combination of boosting and bagging.
http://citeseer.ist.psu.edu/zheng98multiple.html.

[93] *Zheng Z., Webb G., Ting K.* Integrating boosting and stochastic attribute selection committees for further improving the performance of decision tree learning.
http://http://citeseer.ist.psu.edu/article/zheng98integrating.html.

[94] *Zheng Z., Webb G. I.* Stochastic attribute selection committees with multiple boosting: Learning more accurate and more stable classifer committees // Pacific-Asia Conference on Knowledge Discovery and Data Mining. — 1999. — Pp. 123–132.
http://citeseer.ist.psu.edu/article/zheng98stochastic.html.

[95] *Zheng Z., Webb G. I.* Multi strategy ensemble learning: Reducing error by combining ensemble learning techniques. — 2003.
http://www.csse.monash.edu.au/~webb/Files/WebbZheng03.pdf.