Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева»

На правах рукописи

Становов Владимир Вадимович

САМОНАСТРАИВАЮЩИЕСЯ ЭВОЛЮЦИОННЫЕ АЛГОРИТМЫ ФОРМИРОВАНИЯ СИСТЕМ НА НЕЧЕТКОЙ ЛОГИКЕ

05.13.01 – Системный анализ, управление и обработка информации (космические и информационные технологии)

ДИССЕРТАЦИЯ
на соискание ученой степени кандидата технических наук

Научный руководитель
Семенкин Евгений Станиславович
доктор технических наук,
профессор

Красноярск – 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность. На сегодняшний день разработка методов интеллектуального анализа данных является стремительно развивающимся направлением. Цель интеллектуальных систем анализа и обработки информации заключается не только в минимизации затрат исследователей или же пользователя интеллектуальной системы при решении сложных задач, но и полностью автоматический поиск закономерностей в исследуемой предметной области.

Среди всех задач интеллектуального анализа данных стоит выделить задачи классификации, так как к ним сводится множество реальных задач, в том числе классификация изображений, распознавание фрагментов текста, устной речи, классификация поисковых запросов, а также задачи медицинской диагностики. На сегодняшний день разработано множество интеллектуальных систем анализа данных (ИСАД), которые в зарубежной литературе, как правило, называются алгоритмами Data mining. Среди современных отечественных научных школ, занимающихся данной проблематикой, следует выделить Ю.И. Журавлёва, К.В. Рудакова (ВЦ РАН), Н.Г. Загоруйко (ИМ СО РАН), А.А. Дорофеюка (ИПУ РАН).

Недостатком большинства подходов ИСАД является то, что зачастую они работают по принципу «черного ящика», что значительно затрудняет интерпретацию результатов классификации и построенных классификатором закономерностей. По этой причине ряд отечественных и зарубежных исследователей занимается проблемами формирования классификаторов, которые могут быть легко поняты и представлены в форме естественного языка. Наиболее популярным направлением здесь является формирование нечетких систем. Среди отечественных исследователей данной проблематикой занимаются, например, И.А. Ходашинский (ТУСУР), А.П. Рыжов (МГУ), а среди зарубежных следует выделить работы Х. Ишибучи (Hisao Ishibuchi, Osaka University, Japan) и Ф. Херреры (Francisco Herrera, Granada University, Spain).

Системы на нечеткой логике (НЛС) позволяют строить лингвистические правила и объединять их в базы правил, которые представляют собой модель «белого ящика». Нечеткая база правил представляет собой набор независимых

правил, каждое из которых выражает причинно-следственную связь между входными переменными и соответствующим классом. Нечеткие правила оперируют лингвистическими понятиями, вследствие чего могут быть непосредственно восприняты экспертом. Эта особенность позволяет использовать нечеткие базы правил не только как инструмент классификации, но и как метод интеллектуального анализа данных для извлечения новых знаний.

Формирование нечеткой системы классификации заключается в определении структуры базы правил – то есть в поиске значимых правил и выборе наилучшей комбинации этих правил. Данная задача может быть сформулирована как задача оптимизации. При этом целевая функция характеризуется значительной вычислительной сложностью, так как задана алгоритмически, имеет большую размерность и пространство поиска, характеризуется наличием дискретных переменных и т.д.

Эволюционные методы оптимизации хорошо зарекомендовали себя для решения сложных оптимизационных задач, вследствие чего их применение к формированию баз нечетких правил для задачи классификации является целесообразным.

Применение эволюционных методов для построения нечетких классификаторов может повлечь значительные временные затраты. С ростом объемов данных, которые необходимо подвергать интеллектуальному анализу вследствие развития интернет-технологий и отсутствия экспертов в некоторых областях, разработка быстрых и эффективных средств интеллектуального анализа становится всё более востребованной. Процедуры селекции обучающих примеров, подразумевающие выбор обучающих примеров в процессе работы алгоритма позволяют значительно снизить объем требуемых вычислительных ресурсов, и, помимо того, повысить качество и робастность получаемых интеллектуальных систем.

Таким образом, разработка и исследование методов автоматизированного формирования баз нечетких правил методами эволюционных алгоритмов с

активным выбором обучающих примеров для классификации с извлечением скрытых знаний является **актуальной научно-технической задачей.**

Целью диссертационной работы является повышение качества и интерпретируемости нечетких классификаторов, а также снижение требуемых вычислительных ресурсов при их формировании за счет применения самонастраивающихся эволюционных алгоритмов.

Достижение поставленной цели предполагает решение следующих задач:

1) Выполнить обзор существующих методик и алгоритмов формирования нечетких правил и баз правил с целью выявления наиболее эффективных подходов и направлений.

2) Исследовать методы самонастройки эволюционных алгоритмов оптимизации на репрезентативном множестве тестовых задач.

3) Разработать алгоритм формирования баз нечетких правил для решения задач классификации с несбалансированными данными.

4) Разработать метод селекции обучающих примеров для нечеткого классификатора, позволяющий снизить временные затраты и повысить эффективность алгоритма.

5) Реализовать разработанные подходы в виде программных систем и протестировать их эффективность на репрезентативном множестве тестовых и реальных задач.

Методы исследования. В процессе выполнения данной диссертационной работы использовались методы статистической обработки данных, теории вероятностей, эволюционных вычислений, оптимизации, нечеткой логики, системного анализа данных, моделирования динамических систем, выявления закономерностей в исходных данных.

Научная новизна работы включает следующие пункты:

1) Разработан новый самонастраивающийся эволюционный алгоритм формирования нечетких систем для решения задач классификации с представлением баз правил в форме матриц переменной размерности, отличающийся от известных использованием оценки достоверности правил при

назначении их весовых коэффициентов и за счет этого превосходящий по эффективности другие методы эволюционного построения нечетких систем.

2) Разработан новый метод гибридизации Питтсбургского и Мичиганского подходов в эволюционном алгоритме формирования баз нечетких правил, отличающийся от известных использованием при построении новых правил вероятностной процедуры выбора релевантных нечетких термов и позволяющий существенно повысить точность классификации на первых поколениях работы эволюционного алгоритма.

3) Разработан новый метод селекции примеров для обучения классификаторов, отличающийся от известных адаптивной вероятностной процедурой организации подвыборок и назначения весовых коэффициентов и позволяющий одновременно повысить точность классификации и снизить объем требуемых для этого вычислительных ресурсов.

4) Разработан новый метод самонастройки эволюционных алгоритмов, отличающийся от известных схемой оценки успешности операторов, применяемых несколько раз к каждому индивиду, и позволяющий настраивать вероятности применения эвристик в Мичиганской части алгоритма.

Теоретическая значимость результатов диссертационной работы состоит в разработке новых эволюционных алгоритмов формирования нечетких систем, позволяющих получать компактные и точные базы правил посредством использования кодирования в форме матриц переменной размерности, гибридизации Питтсбургского и Мичиганского подходов и применения алгоритма самонастройки, для решения задач классификации и разработке нового метода активной селекции обучающих примеров для классификаторов, что представляет собой существенный вклад в теорию и практику исследования методов формирования нечетких систем посредством эволюционных алгоритмов.

Практическая ценность. Разработанные методы реализованы в виде программной системы, для решения задач классификации. Программная система позволяет быстро формировать базы нечетких правил за счет использования самонастройки, а также снижения количества пересчетов степеней

принадлежности и весов правил. Программная система протестирована на задачах классификации из области техники, распознавания изображений, банковского скоринга и медицинской диагностики.

Реализация результатов работы. Разработанные алгоритмы использованы при выполнении исследований в рамках российско-германских проектов (совместно с университетом г. Ульм) «Распределенные интеллектуальные информационные системы обработки и анализа мультилингвистической информации в диалоговых информационно-коммуникационных системах» (ФЦП ИР, ГК №11.519.11.4002) и «Математическое и алгоритмическое обеспечение автоматизированного проектирования аппаратно-программных комплексов интеллектуальной обработки мультилингвистической информации в распределенных высокопроизводительных системах космического назначения» (ФЦП НПК, ГК № 16.740.11.0742), российско-словенского проекта (совместно с университетом г. Марибор) «Manpower control strategy determination with self-adapted evolutionary and biologically inspired algorithms» (ARRS Project BI-RU/14-15-047), а также в рамках проекта №8.5541.2011 «Развитие теоретических основ автоматизации математического моделирования физических систем на основе экспериментальных данных» и проекта № 140/14 «Разработка теоретических основ эволюционного проектирования интеллектуальных информационных технологий анализа данных» тематического плана ЕЗН СибГАУ. Диссертационная работа была поддержана Фондом содействия развитию малых форм предприятий в научно-технической сфере по программе «У.М.Н.И.К» («Участник молодежного научно-инновационного конкурса») в рамках НИОКР «Разработка программного обеспечения интеллектуального анализа данных "FuzzyMiner"» на 2014-2016 гг., а также Российским Фондом Фундаментальных Исследований в рамках проекта № 16-31-00349 «Разработка алгоритмов и подходов к повышению качества и скорости формирования технологий интеллектуального анализа данных посредством снижения размерности данных» на 2016-2017 гг.

Три программные системы, разработанные в ходе выполнения диссертации, зарегистрированы в Роспатенте. Данные программные системы используются в учебном процессе Института информатики и телекоммуникаций СибГАУ при выполнении лабораторных и курсовых работ и переданы в две инновационные IT-компании.

Основные защищаемые положения:

1) Разработанный метод формирования нечетких систем для решения задачи классификации самонастраивающимся эволюционным алгоритмом позволяет формировать компактные и легко интерпретируемые базы правил.

2) Предложенная схема кодирования базы правил в эволюционном алгоритме позволяет снизить вычислительную сложность алгоритма.

3) Гибридный алгоритм формирования нечетких баз правил для решения задач классификации не уступает по точности другим подходам.

4) Разработанный метод селекции обучающих примеров позволяет существенно снизить объем требуемых вычислительных ресурсов.

5) Применение метода селекции обучающих примеров к гибридному эволюционному алгоритму формирования нечетких баз правил позволяет формировать более эффективные классификаторы в смысле точности, полноты и -меры.

Публикации. По теме данной работы опубликовано более 35 печатных работ, в том числе восемь в журналах из Перечня ВАК, а также зарегистрировано в Роспатенте три программные системы.

Апробация работы. Результаты диссертационной работы были доложены на 12 всероссийских и международных научно-практических конференциях и конференциях с международным участием, в том числе на Пятой международной конференции «Системный анализ и информационные технологии» САИТ-2013 (Красноярск, 2013), Второй и Третьей международных конференциях по математическим моделям и их применениям (2nd and 3rd International Workshops on Mathematical Models and their Applications, Красноярск, 2013, 2014), III Всероссийской научной конференции с международным участием «Теория и

практика системного анализа» (ТПСА, Рыбинск, 2014), 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD, Xaimen, China, 2014), 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO, Vienna, Austria, 2014), International Congress on Evolutionary Computations (CEC, Sendai, Japan, 2015), International Conference on Swarm Intelligence (ICSI, Peking, China, 2015), IEEE Symposium Series on Computational Intelligence (SSCI 2015, South Africa), 4th International Congress on Advanced Applied Informatics (AAI 2015), July 12-16, Okayama Convention Center, Okayama, Japan, 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2016).

Структура работы. Диссертация содержит 129 страниц основного текста, состоит из введения, четырех глав, заключения, списка литературы на 127 источников, основной текст включает 39 рисунков и 48 таблиц.

Глава 1. Разработка и исследование эффективности методов самонастройки эволюционных алгоритмов

1.1 Общие сведения об эволюционных алгоритмах

В самом общем смысле, эволюционные алгоритмы есть алгоритмы, моделирующие принципы естественной, природной эволюции видов. Основной движущей силой развития и эволюции видов является естественный отбор, то есть принцип выживания сильнейшего. Как известно, основы эволюционной теории заложил Ч. Дарвин ещё в XIX веке, однако идея использования этих принципов в других областях знаний, не связанных с живой природой, возникла только с появлением ЭВМ и развитием генетики.

Широко известный принцип выживания наиболее пригодных индивидов в эволюционных алгоритмах и является, пожалуй, основным их отличием от других подходов. Однако стоит отметить и другие специфические понятия и термины, которые используются при описании эволюционных алгоритмов [60].

Стоит отметить, что эволюционные алгоритмы применяются [54] для решения всевозможных задач с целью получения какого-то хорошего решения. Множество потенциальных решений задачи будем называть популяцией. Каждое решение из популяции по-разному приспособлено к внешней среде, соответственно необходимо его каким-то образом оценить. Для оценки решения, которое также называется индивидом в популяции, используется некоторая функция, называемая функцией пригодности, или целевой функцией. В общем случае таких функций может быть несколько, если необходимо оценить несколько характеристик индивида-решения. В этом случае мы имеем дело с задачей многокритериальной оптимизации [127].

Работа эволюционного алгоритма заключается в проведении обычно большого числа итераций-поколений, с течением которых должны получаться все более приспособленные индивиды. Приспособление индивидов к условиям внешней среды происходит с использованием двух основных принципов, которые встречаются в природе – наследственности и изменчивости. Наследственность

предполагает, что индивиды, полученные на текущем поколении сохранили, унаследовали часть признаков и свойств своих родителей – индивидов предыдущего поколения. Изменчивость в свою очередь означает, что, несмотря на то, что индивиды-потомки формируются только благодаря своим родителям, в них могут и должны появиться какие-то признаки, которых не было на предыдущем поколении.

Как правило, за наследственность в эволюционных алгоритмах отвечают операторы селекции и скрещивания индивидов. За изменчивость отвечает оператор мутации. Все эти операторы специфичны и зависят от конкретного типа и конфигурации алгоритма. Тем не менее, можно выделить общие черты, характерные для них: селекция выбирает из текущей популяции наиболее пригодных индивидов в соответствии с их целевой функцией, затем скрещивание создает новых потомков, смешивая признаки родителей, и мутация добавляет случайные изменения в полученных потомков.

Ещё одним важным аспектом эволюционных алгоритмов является то, что в них зачастую решения кодируются, то есть представляют собой некоторые структуры, которые могут быть преобразованы в решение задачи. Способ и тип кодирования индивидов зависит от конкретной задачи и может существенно отличаться.

Для начала, рассмотрим классический [10] генетический эволюционный алгоритм (ГА). Название «генетический алгоритм» подразумевает, что в нем будут использоваться гены, то есть структуры, чем-то напоминающие молекулы Дезоксирибонуклеиновой кислоты (ДНК). ДНК состоит из четырех нуклеотидов, кодирующих генетическую информацию. При кодировании индивидов в генетическом алгоритме используется тот же принцип, то есть решение задается в виде хромосомы, генотипа, и затем при вычислении значений пригодности раскодируется и формирует фенотип. Однако, в отличие от ДНК, при реализации генетических алгоритмов на ЭВМ чаще используют бинарное кодирование, при котором хромосома является бинарной (из нулей и единиц) строкой.

Эволюционные алгоритмы интенсивно используют случайные числа в процессе своей работы, поэтому их относят к стохастическим алгоритмам. На самом деле при реализации на ЭВМ используются псевдослучайные числа, однако в дальнейшем будет использоваться термин «случайные» для простоты. Случайные числа используются для задания начальной популяции, отбора, селекции родителей, в процессе скрещивания и при проведении мутации. Поэтому качество работы эволюционных алгоритмов напрямую зависит от качества используемого генератора случайных чисел.

Классической областью применения генетических алгоритмов является решение задач оптимизации функций. При этом одной из основных особенностей ГА является то, что они не требуют никаких дополнительных знаний о природе целевой функции, поэтому применимы даже для оптимизации сложных, алгоритмических заданных функций. Более того, в силу своей стохастической многоагентной природы, ГА являются методами глобальной оптимизации и не столь сильно подвержены свойственной множеству методов оптимизации проблеме попадания в локальный оптимум.

Применение генетических алгоритмов как методов оптимизации может быть не всегда оправданно вследствие достаточно большого времени счета ГА, особенно если речь идет об очень простых функциях. Однако, на задачах больших размерностей, при значительной сложности оптимизируемой функции, генетические алгоритмы могут быть очень полезны.

Рассмотрим другой тип эволюционных алгоритмов – генетическое программирование (ГП). ГП является методом, расширяющим возможности генетического алгоритма за счет изменения способа кодирования решений. В генетическом программировании каждое решение представляется в виде графа, дерева, то есть разветвленной структуры. ГП было предложено впервые Джоном Козой [76], и использовалось для эволюции интегральных схем. Преимуществом ГП является практически неограниченное пространство поиска, в отличие от ГА.

Например, если длина бинарной строки в ГА равна 10, то всего возможно закодировать 1024 решения. В ГП же размер дерева не фиксируется заранее,

таким образом, сложность закодированного решения может быть больше или меньше.

Классическим примером использования ГП является решение задачи символьной регрессии, то есть восстановления функциональной зависимости между входами и выходами с помощью математических функций, таких как $+, -, *, /, \sin(x), \cos(x), x^2, e^x$ и так далее. Дерево состоит из функциональных вершин, а на концах ветвей стоят терминальные вершины –переменные-входы задачи или константы.

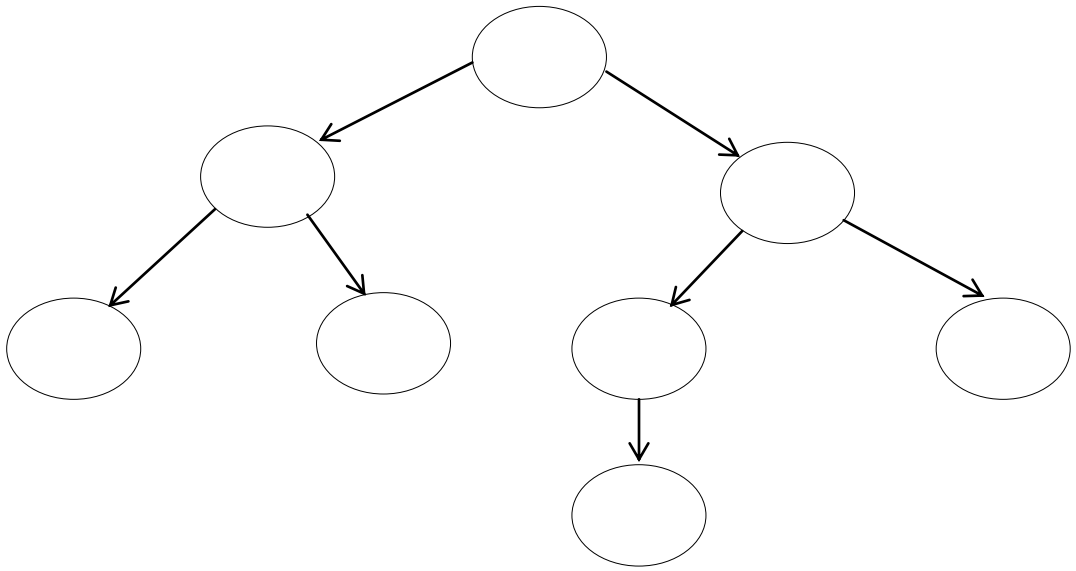


Рисунок 1.1. Пример дерева в генетическом программировании

В ходе эволюционного процесса получаются деревья, чей выход все более приближается к искомому выходу. При этом для формирования новых потомков используются те же операторы – селекция, скрещивание и мутация, но в отличном от ГА виде. ГП может также решать и другие задачи, такие как подбор структуры нейронной сети, формирование базы нечетких правил, формирование ансамблей и т.д.

Помимо ГА и ГП существуют и другие эволюционные алгоритмы, например эволюционные стратегии, эволюционное программирование, дифференциальная эволюция и т.д. Существует также множество специфических алгоритмов, использующих те же принципы, что были описаны выше, и предназначенные для решения узкого класса задач.

В заключение стоит отметить одну работу [84], посвященную пониманию того, что же является эволюционным алгоритмом. В частности, одна из идей, которая там высказывается, заключается в том, что все алгоритмы и методы оптимизации являются эволюционными, в том числе классические и хорошо известные, такие как метод Ньютона и т.д [3, 17]. Их отличие от классических эволюционных алгоритмов заключается, по сути, в использовании специфических эволюционных операторов, которые позволяют эффективно решать узкий круг задач. Эволюционные же алгоритмы, в данном контексте, являются обобщением всех известных когда-либо способов поиска и оптимизации.

В следующих разделах данной главы мы более подробно остановимся на описании генетических алгоритмов и алгоритмов генетического программирования.

1.2 Стандартный генетический алгоритм

Рассмотрим более подробно генетические алгоритмы [85]. Их отличительной чертой является кодирование индивида при помощи бинарной строки, как уже было упомянуто выше. Поиск новых решений в ГА происходит посредством проведения итераций-поколений. На каждом поколении применяются несколько генетических операторов для формирования потомков из родителей. Общий цикл генетического алгоритма можно представить следующим образом:

1) Создать начальную популяцию, задавая гены в хромосоме равными 0 или 1 с вероятностью 0.5.
2) Вычислить пригодность индивидов в популяции.
3) Если критерий остановки достигнут, перейти к шагу 8.
4) Произвести селекцию – отбор наиболее пригодных индивидов, сформировать популяцию родителей.
5) Произвести скрещивание родителей и получить потомков.
6) Произвести мутацию потомков.
7) Сформировать новое поколение и перейти к шагу 2.

8) Выдать наилучшее найденное решение в качестве ответа.

На этапе инициализации каждая хромосома заполняется нулями или единицами с равной вероятностью. Такой подход реализует принцип максимума энтропии, то есть ситуацию, при которой мы ничего не знаем о пространстве поиска и не строим никаких предположений. Если же допускать некоторые предположения и изменять вероятности нулей и единиц на определенных местах в хромосоме, то в общем случае это может привести к худшим результатам, так как некоторые области пространства поиска станут труднодостижимыми.

Селекция есть оператор выбора наиболее пригодных индивидов для последующего скрещивания. Селекция является одним из ключевых этапов ГА и от её эффективности существенно зависит качество работы алгоритма. Существует три основных типа селекции. Пропорциональная селекция предполагает, что вероятность выбора индивида пропорциональна его пригодности, то есть чем выше пригодность, тем больше вероятность быть выбранным. Ранговая селекция схожа с пропорциональной, в ней сначала индивиды сортируются по значениям пригодностей, далее каждому индивиду назначается ранг, и в соответствии с этими рангами рассчитываются вероятности как в пропорциональной селекции. При турнирной селекции создаются турниры, в которых соревнуются несколько индивидов, победителем считается индивид с наибольшей пригодностью. Размер турнира, то есть число индивидов, попадающих в него, может быть различным.

Скрещивание есть оператор, перемешивающий генетическую информацию отобранных родителей. Этот оператор реализует стратегию использования имеющейся генетической информации с целью поиска новых решений. Как правило, в скрещивании принимают участие два индивида. Одноточечное скрещивание предполагает задание некоторой точки, в которой происходит разрыв хромосомы. Первая часть хромосомы потомка берется у первого родителя, вторая – у второго, для второго потомка наоборот. Двухточечное скрещивание предполагает задание двух точек, в которых происходит разрыв хромосомы. Первый потомок получает первую часть от первого родителя, вторую – от

второго, а третью – от первого; второй – наоборот. Равномерное скрещивание означает, что каждый ген потомка может быть с равной вероятностью унаследован от обоих родителей, хотя существуют подходы, в которых эта вероятность может зависеть от пригодностей родителей [8].

Мутация применяется к одному из потомков, полученных после скрещивания, этот потомок выбирается случайно, второй отбрасывается. При мутации в ГА каждый ген может изменить свое значение на противоположное с некоторой малой вероятностью. Эта вероятность обычно зависит от длины хромосомы. Слабая мутация означает, что вероятность инвертирования гена меньше или равна $\frac{1}{3L}$, где L – длина хромосомы. При средней мутации вероятность инвертирования равна $\frac{1}{L}$, при высокой - больше $\frac{3}{L}$. Оператор мутации позволяет восстановить утерянную в ходе эволюционного процесса генетическую информацию или же добавить новую. От вероятности мутации зависит степень разброса точек в пространстве оптимизации.

Генетический алгоритм прекращает работу, если было найдено достаточно точное решение, или же достигнуто максимальное число поколений.

1.3 Алгоритм генетического программирования

Алгоритм генетического программирования отличается от ГА в первую очередь способом представления решений. В ГП каждый индивид представляется в виде дерева, хотя существуют особые подходы ГП, в которых используются другие структуры, например декартов ГП [86, 115]. Дерево состоит из множества узлов, которые могут быть из терминального (T) и функционального (F) множества. В терминальное множество включаются входы системы и константы, в функциональном множестве содержатся различные математические и логические операции, в том числе операции, заданные пользователем.

Типичной и наиболее простой задачей для ГП является символьная регрессия [35]. При решении данной задачи необходимо восстановить зависимость между входными и выходными переменными в форме функции,

используя арифметические и логические операции. Данный подход может быть использован для аппроксимации функций и значительным его преимуществом является возможность анализа полученных в ходе эволюционного процесса зависимостей. Хотя, в некоторых случаях, такой анализ может представлять трудности.

Шаги алгоритма ГП в целом аналогичны шагам ГА, описанным в предыдущем пункте, основное отличие состоит в используемых операторах. Рассмотрим каждый из них более подробно.

На этапе **инициализации** необходимо вырастить начальную популяцию деревьев. В отличие от ГА, где задание бинарных строк не представляет сложности и особого интереса, в ГП этот этап значительно сложнее и в целом очень сильно влияет на всю дальнейшую работу алгоритма. Выделяют два основных типа инициализации деревьев – полная инициализация и инициализация выращиванием. Суть полной инициализации заключается в том, что изначально задается некоторое ограничение на максимальную глубину дерева, а затем все деревья выращиваются до тех пор, пока не будет достигнута максимальная глубина. На максимальной глубине дерева ставится случайно выбранный элемент терминального множества, а на всех предыдущих вершинах – случайно выбранные элементы функционального множества.

Отличие мутации выращиванием заключается в том, процесс выращивания может оборваться на некотором этапе и в дальнейшем рост дерева прекратится. В точке, где рост прекращается, ставится случайно выбранный элемент терминального множества. При этом ключевым моментом в этом методе является задание вероятности окончания роста, так как от нее сильно зависит размер дерева. Также как и в полном методе, если достигается максимальная глубина дерева, рост прекращается.

Каждый из этих методов инициализации имеет свои достоинства и недостатки. При полном методе деревья получаются более сложными, и, возможно, более точными, в то время как при выращивании получаются более простые решения на первом поколении. Найти компромисс между сложностью и

точностью решений бывает проблематично, поэтому на практике зачастую применяют оба метода формирования начальной популяции с равной вероятностью.

Селекция в генетическом программировании по своей сути ничем не отличается от селекции в ГА, поэтому все типы селекции могут быть применены для ГП в полной мере и без изменений.

Скрещивание в ГП является, пожалуй, ключевым оператором, от которого зависит качество работы алгоритма. В отличие от ГА, где можно просто поменять части хромосом местами, в ГП необходимо скрещивать деревья, то есть разветвленные структуры. Самым простым и зачастую не самым эффективным является стандартное скрещивание. В этом методе в каждом из деревьев – родителей, полученных после проведения селекции, случайным образом выбирается вершина. Далее, деревья обмениваются поддеревьями в этой точке и формируют двух потомков.

Одноточечное скрещивание происходит следующим образом. Два дерева накладываются друг на друга так, что их корни совпадают, и далее выделяется общая область у деревьев. Из этой общей области выбирается одна вершина, которая и является точкой скрещивания. Далее деревья обмениваются поддеревьями, начиная с этой точки. Структура общей области зависит от арности операций, стоящих в функциональных вершинах дерева, а также от наличия этих вершин.

Мутация в генетическом программировании может производиться как точечно, так и заменой части дерева. Точечная мутация меняет случайно выбранную вершину в дереве на вершину того же типа. То есть, если вершина принадлежит функциональному множеству, то она не может быть заменена на элемент терминального множества, также учитывается арность операций. Второй тип мутации называется мутацией выращиванием. При этом способе сначала случайным образом выбирается одна из вершин дерева, а затем в этой точке выращивается поддерево, с использованием того же метода, который применялся

на этапе инициализации. Как правило, мутация в ГП применяется с достаточно низкой вероятностью.

Помимо решения задачи символьной регрессии алгоритм ГП может быть применен для множества других задач, см. например [90].

1.4 Методы самонастройки эволюционных алгоритмов

Несмотря на то, что эволюционные алгоритмы широко используются в различных сферах, их применение связано с некоторыми трудностями. Главной трудностью является выбор типов генетических операторов, так как от них зависят свойства алгоритма. То есть, на эффективность работы эволюционного алгоритма сильно влияет его конфигурация. Оптимальный набор операторов и параметров сильно зависит от решаемой задачи, причем определить этот набор на основании какой-то информации о задаче бывает сложно. Даже после проведения множества всеобъемлющих тестов на различных типах задач и некого теоретического анализа применимости различных конфигураций настроек, нет никакой гарантии, что выбранная конфигурация будет эффективной на реальной задаче. Вследствие этого, многие исследователи пытаются разработать механизмы адаптации алгоритма к решаемой задаче в ходе поиска её решения.

Среди методов адаптации эволюционных алгоритмов можно выделить методы самоконфигурации и методы самонастройки. К методам самоконфигурации обычно относят методы, которые выбирают типы операторов, исходя из некоторого набора, например один из трех типов селекции. Методы самонастройки подразумевают автоматическую настройку каких-либо вещественных параметров, таких как вероятность мутации.

В начале работы алгоритма самоконфигурации предполагается, что все операторы имеют равные шансы быть успешными, поэтому им назначаются одинаковые вероятности применения. В ходе работы самоконфигурируемого алгоритма вероятности применения операторов должны сдвигаться в сторону тех операторов, которые показывают себя как более успешные по сравнению с другими. В том, как определять успешность операторов в эволюционном

алгоритме, а также как изменять эти вероятности на каждом поколении и состоит отличие одних методов самоконфигурации от других.

Стоит также отметить, что при любом методе самоконфигурации, как правило, устанавливают некую нижнюю границу вероятности применения оператора. Эта граница необходима для того, чтобы ни одна вероятность применения оператора не стала равной нулю. В этом случае данный оператор перестает применяться вообще, и, следовательно, если на каком-то этапе эволюционного процесса он мог бы оказаться самым востребованным и эффективным, он не сможет повысить свою вероятность применения. Несмотря на то, что подобные ситуации крайне редки, и некоторые исследователи выступают против введения подобных порогов, в данной работе они будут введены.

Здесь будут рассмотрены два метода самоконфигурации, схожие друг с другом по основной идее, но отличающиеся по методу определения успешности оператора. Первый метод был взят из статьи [89] Вольфганга Банзафа, в которой он описал свой метод для настройки вероятности применения операторов мутации в генетическом программировании. Метод PDP (Population-level Dynamic Probabilities) определяет успешность оператора на основании сравнения пригодностей потомков и родителей, полученных при помощи этого оператора. Опишем весь метод более подробно.

Каждому генетическому оператору назначаются минимальные вероятности его применения $p_{all} = \frac{0.2}{n}$, где n – число различных операторов данного типа. Далее, вводятся значения $r_i = \frac{success_i{}^2}{used_i}$ для каждого оператора i, где $used_i$ – число применений данного оператора, $success_i$ – число успешных применений оператора, то есть случаев, когда после использования данного оператора пригодность потомка превысила пригодность соответствующего родителя. Для каждого оператора рассчитывается вероятность его применения p_i по следующей формуле:

$$p_i = p_{all} + \left[r_i \frac{1.0 - np_{all}}{scale} \right],$$

$$scale = \sum_{j=1}^{n} r_j.$$

Значения $success$ возводятся в квадрат из-за очень низкой степени успешности генетических операторов. Значения $scale$ позволяют нормировать значения вероятностей так, что их сумма всегда равна единице. Среди недостатков метода PDP стоит отметить, что процедура оценки успешности оператора имеет некоторые слабые места, в частности, при сравнении пригодности потомка с пригодностями родителей неясно, с которым из родителей сравнивать. В данной работе родитель выбирался случайным образом, и с ним производилось сравнение.

Второй метод самоконфигурации основывается на поощрении того оператора, который доставил наибольшую суммарную пригодность на данном поколении, и был взят из [104, 7]. Пусть, z – число различных операторов k-го типа. Начальные значения вероятностей устанавливаются как $p_i = \frac{1}{z}$. Оценка эффективности каждого оператора каждого типа производится на основании усредненных значений пригодности:

$$AvgFit_i = \frac{\sum_{j=1}^{n_i} f_{ij}}{\sum_{j=1}^{n_i} 1}, i = 1,2 \dots, z$$

где n_i – количество потомков, сформированных при участии -го оператора, f_{ij} – пригодность j-го потомка, построенного при участии i-го оператора, $AvgFit_i$ – средняя пригодность решений, построенных при помощи i-го оператора.

После этого вероятность применения оператора, чье значение $AvgFit_i$ является наибольшим среди всех операторов такого типа, увеличивается на $\frac{(z-1)K}{zN}$, а вероятности применения остальных операторов уменьшаются на $\frac{K}{zN}$, где N – число поколений генетического алгоритма, K – константа, её значение устанавливалось равным 0.5.

Стоит также отметить, что есть варианты алгоритмов самоконфигурации, в которых рассматриваются не вероятности применения отдельных типов операторов, а вероятности применения комбинаций операторов (например, турнирная селекция, равномерное скрещивание и сильная мутация в ГА). Однако в нескольких работах было показано, что такой подход имеет меньшую эффективность.

Описанные выше методы самонастройки могут быть применены для генетического алгоритма, для алгоритма генетического программирования, а также для других эволюционных алгоритмов, имеющих различные операторы.

1.5 Программная реализация и исследование эффективности самоконфигурируемых алгоритмов

Стандартный генетический алгоритм и алгоритм генетического программирования были реализованы в виде программных систем для проверки эффективности и сравнения методов самоконфигурации. Рассмотрим сначала генетический алгоритм.

Стандартный генетический алгоритм был реализован в среде CodeBlocks 12.11 на языке C++ без использования визуальной оболочки, то есть в консольном режиме, и без средств объектно-ориентированного программирования. Это было необходимо для ускорения работы алгоритма и повышения надежности работы программы. Результаты работы программной системы выводились в текстовые файлы для последующей обработки и анализа в других приложениях. Стоит отметить, что такая реализация генетического алгоритма позволяет относительно просто переносить и добавлять его в другие программные системы в качестве оптимизационной процедуры.

Первое исследование было проведено для сравнения двух методов самоконфигурации, описанных в предыдущем пункте. Для определения качества работы генетического алгоритма на этих задачах вычислялось два показателя – надежность и средний номер поколения. Надежность вычислялась как отношение числа запусков, когда алгоритм находил решение с заданной точностью, к

общему числу запусков алгоритма. Средний номер поколения вычислялся как среднее по номерам поколений, на которых было впервые найдено решение задачи.

Оценка надежности необходима вследствие того, что генетический алгоритм имеет стохастическую природу, и, следовательно, найденные решения меняются от запуска к запуску. Поэтому единичный запуск алгоритма не может характеризовать его качество. Оценивая надежность, можно получить представление о том, может ли алгоритм найти решение при заданном ограничении на число индивидов и поколений. Средний номер поколения также является немаловажным фактором, показывающим скорость работы алгоритма. Его следует применять для сравнения двух конфигураций алгоритмов, имеющих одинаковую надежность.

Для сравнения было взято несколько сложных оптимизационных задач из открытого соревнования оптимизаторов Black Box Optimization Benchmark – Comparing Continuous Optimizers [55], в частности, 15 функций с номерами 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15, 17, 18. Набор тестовых задач [51] в этом соревновании [56, 57] состоит из примеров вещественных функций, на которых стандартные оптимизационные процедуры показывают плохие результаты – это функции с множеством экстремумов, плохой обусловленностью, наличием плато, и т.д. Размерность решаемой задачи также варьировалась, тесты были проведены для размерностей 5 и 10.

Число запусков алгоритма равнялось 100 для каждой функции. Среднее число поколений – это усредненный по всем запускам номер поколения, на котором было найдено решение. Требуемая точность устанавливалась равной 0.01.

Для каждой функции устанавливался свой объем вычислительных ресурсов так, чтобы надежность не была равна ни единице, ни нулю, так как в этом случае невозможно будет определить разницу между алгоритмами. Заданное число поколений и индивидов для каждой функции представлено в таблице 1.1.

Таблица 1.1 Объем ресурсов ГА для различных задач

№ F	1	2	3	4	5	6	7	8	11	12	13	14	15	17	18
Инд.	30	50	100	100	30	100	100	100	150	100	100	100	150	75	100
Пок.	30	50	100	100	30	100	100	100	150	100	100	100	150	75	100

Для сравнения двух алгоритмов по надежности использовался непараметрический U-критерий Манна-Уитни. Он позволяет выявлять различия в значениях параметра между двумя малыми выборками. То есть, если значение критерия меньше ли равно табличному значению (с уровнем значимости 0.05 в данной работе), то признается существенная разница между выборками, а, следовательно, и алгоритмами. Значения надежности алгоритма замерялись 10 раз для каждой функции, таким образом, каждая из выборок имела 10 измерений. Также был протестирован стандартный генетический алгоритм на всех наборах настроек без использования самонастройки. Размер турнира устанавливался равным 3.

Для наглядного представления результатов тестирования в таблицах ниже было введено значение T, при этом 1 означает, что первый из алгоритмов лучше второго, 0 – алгоритмы неразличимы на данной функции, -1 – второй алгоритм лучше первого. В таблице 1.2 представлены результаты сравнения второго алгоритма самонастройки с методом PDP при размерности 5.

Таблица 1.2 Сравнение метода PDP со вторым алгоритмом, размерность 5

№F	1	2	3	4	5	6	7	8	11	12	13	14	15	17	18	Сумм.
T	0	1	-1	0	1	-1	0	-1	0	0	0	0	-1	0	-1	-5+2=-3

Как можно видеть из таблицы, только на второй и пятой функции метод PDP показал лучшие результаты. На семи функциях второй метод продемонстрировал лучшие значения пригодностей. Результаты при размерности 10 представлены в таблице 1.3.

Таблица 1.3 Сравнение метода PDP со вторым алгоритмом, размерность 10

№F	1	2	3	4	5	6	7	8	11	12	13	14	15	17	18	Сумм.
T	1	1	0	0	1	0	-1	0	0	-1	0	0	-1	-1	-1	-5+3=-2

Аналогичные тесты были произведены для сравнения второго метода самонастройки, который показал себя лучше, со стандартным генетическим алгоритмом на всех возможных 27 вариантах настроек (по три типа селекции, скрещивания и мутации). В данном случае для каждой конфигурации настроек (номер конфигурации обозначен №K в таблице) рассчитывалась сумма значений тестов по всем функциям. Сумма тестов $\sum T$ может изменяться в пределах от -15 до 15. Значение -15 означает, что данная стандартная конфигурация показала себя лучше на всех 15 функциях, 15 означает, что данная конфигурация всегда оказывалась хуже самонастраивающегося алгоритма. Тесты были проведены при размерности 5, результаты приведены в таблице 1.4.

Таблица 1.4 Результаты тестирования ГА

Селекция		Пропорциональная			Ранговая			Турнирная (3)		
Мутация	Скрещивание	одноточечное	двухточечное	равномерное	одноточечное	двухточечное	равномерное	одноточечное	двухточечное	равномерное
Слабая	$\sum T$	14	14	14	13	13	12	12	12	12
Средняя	$\sum T$	10	10	11	4	3	7	9	10	11
Сильная	$\sum T$	11	12	11	2	3	3	-1	1	2

Таким образом, в подавляющем большинстве случаев алгоритм с самонастройкой показал лучшие результаты. Стоит отметить, что стандартный алгоритм немного превзошел алгоритм с самонастройкой в комбинациях с турнирной селекцией, особенно в комбинации с сильной мутацией.

В качестве примера, демонстрирующего изменение вероятностей применения операторов в процессе работы алгоритма, приведем графики, построенные при оптимизации функции 11 (Discus Function). Эволюционный процесс в данном эксперименте не прекращался даже после нахождения решения с заданной точностью. Изменение вероятностей применения оператора селекции представлено на рисунке 1.2.

Из рисунка можно видеть, что турнирная селекция «забрала» большую часть вероятности и стала доминировать над другими типами, хотя вначале ранговая и турнирная селекция были примерно равны по эффективности. Аналогичный график для скрещивания представлен на рисунке 1.3. Как можно видеть, ни один из операторов скрещивания не доминирует над другими операторами, то есть на данной задаче операторы скрещивания показывают схожую эффективность. Графики для мутации представлены на рисунке 1.4.

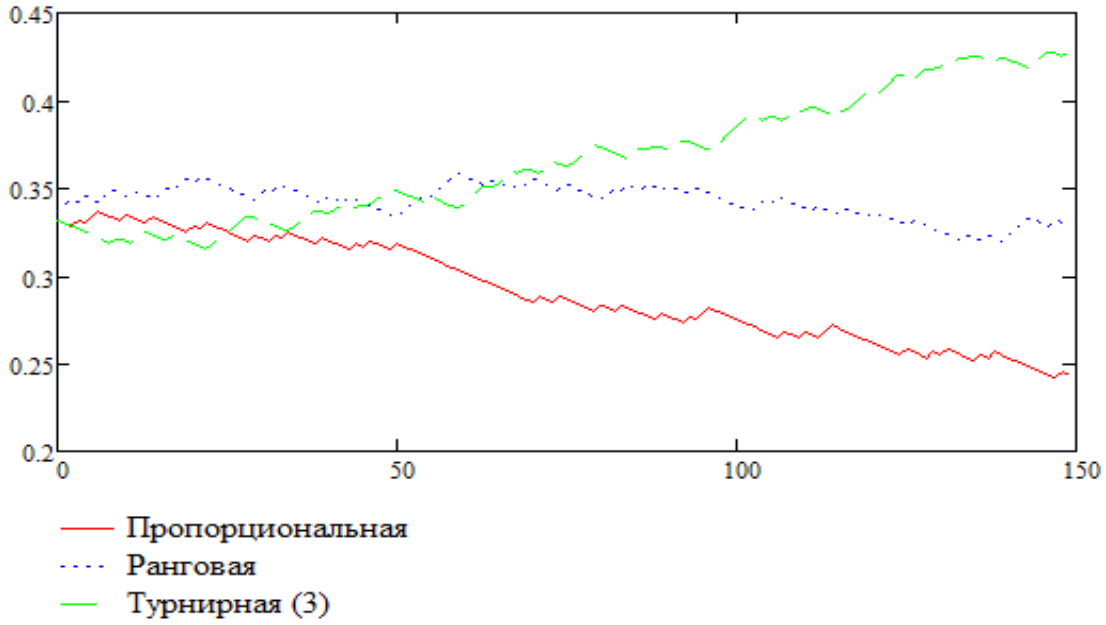


Рисунок 1.2. Изменение вероятностей применения операторов селекции

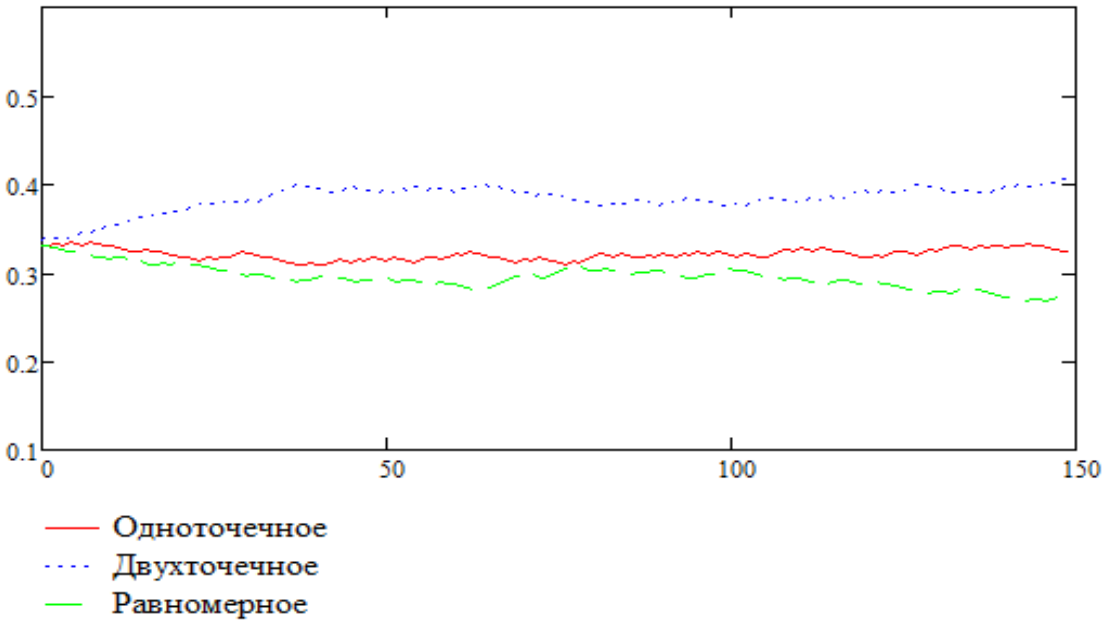


Рисунок 1.3. Изменение вероятностей применения операторов скрещивания

Различия в вероятности применения операторов мутации не столь сильны после 150 поколений, как для операторов селекции. Стоит отметить, что все же средняя мутация предпочтительнее.

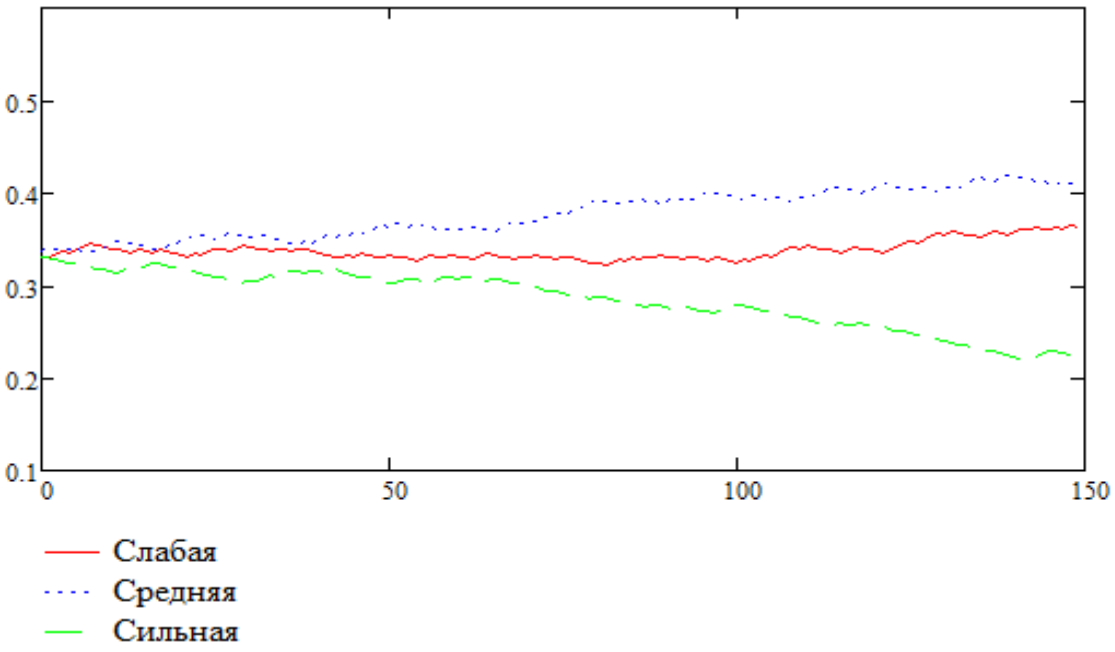


Рисунок 1.4. Изменение вероятностей применения операторов мутации

Пример поведения вероятностей операторов для метода PDP представлен на следующих трех рисунках.

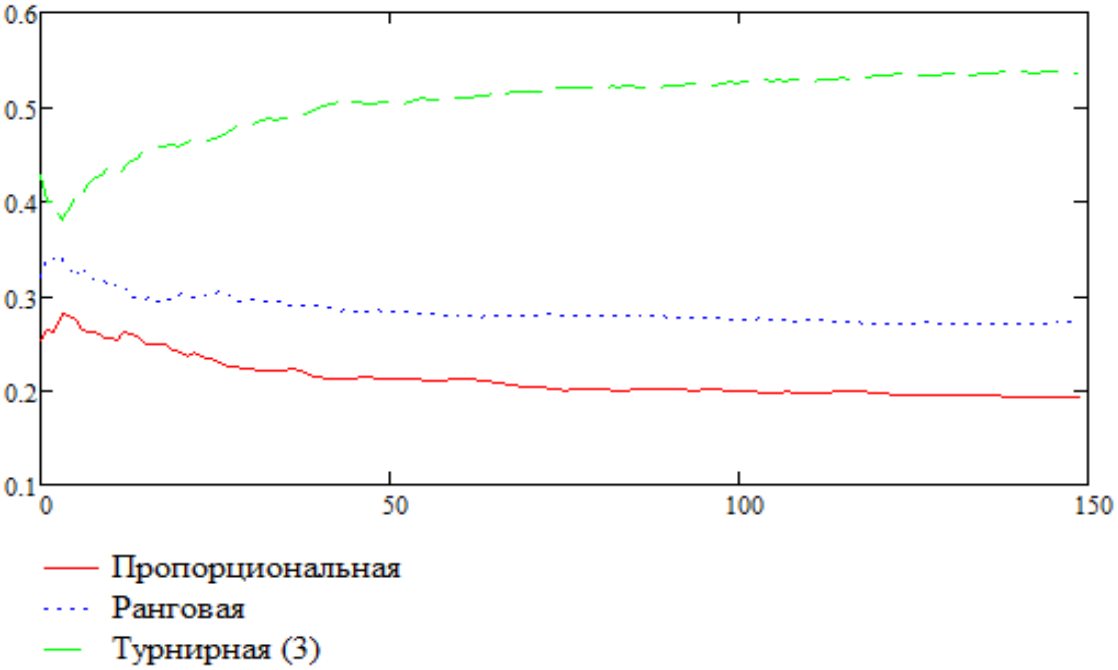


Рисунок 1.5. Изменение вероятностей применения операторов селекции, PDP

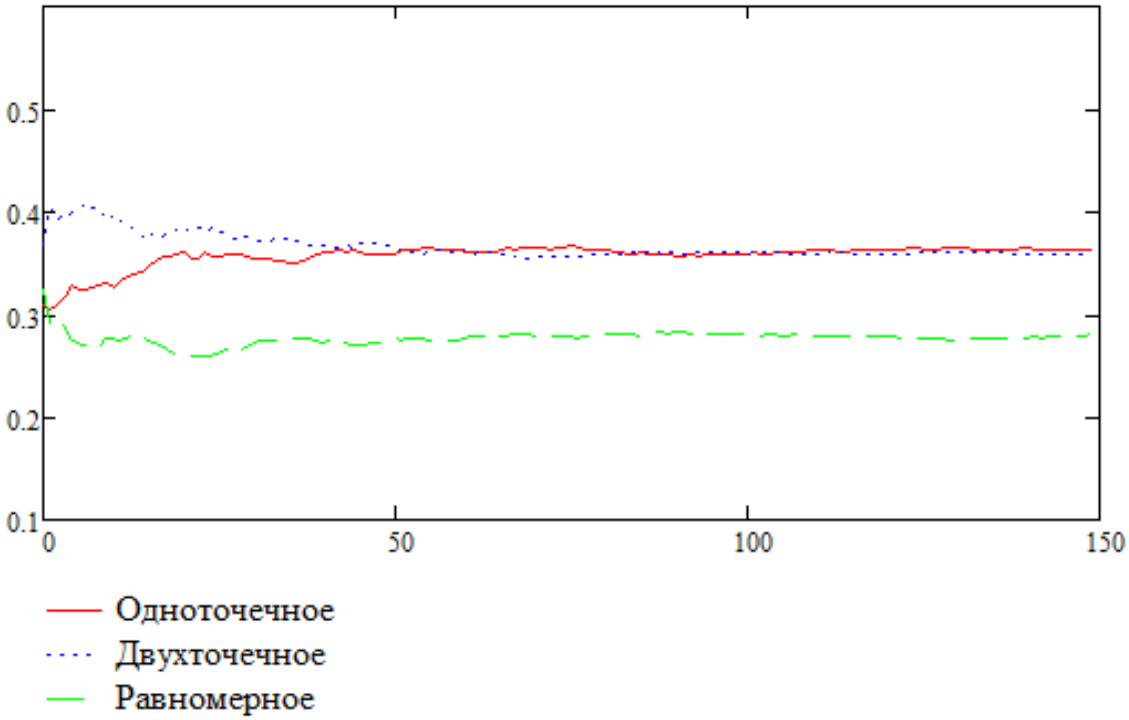


Рисунок 1.6. Изменение вероятностей применения операторов скрещивания, PDP

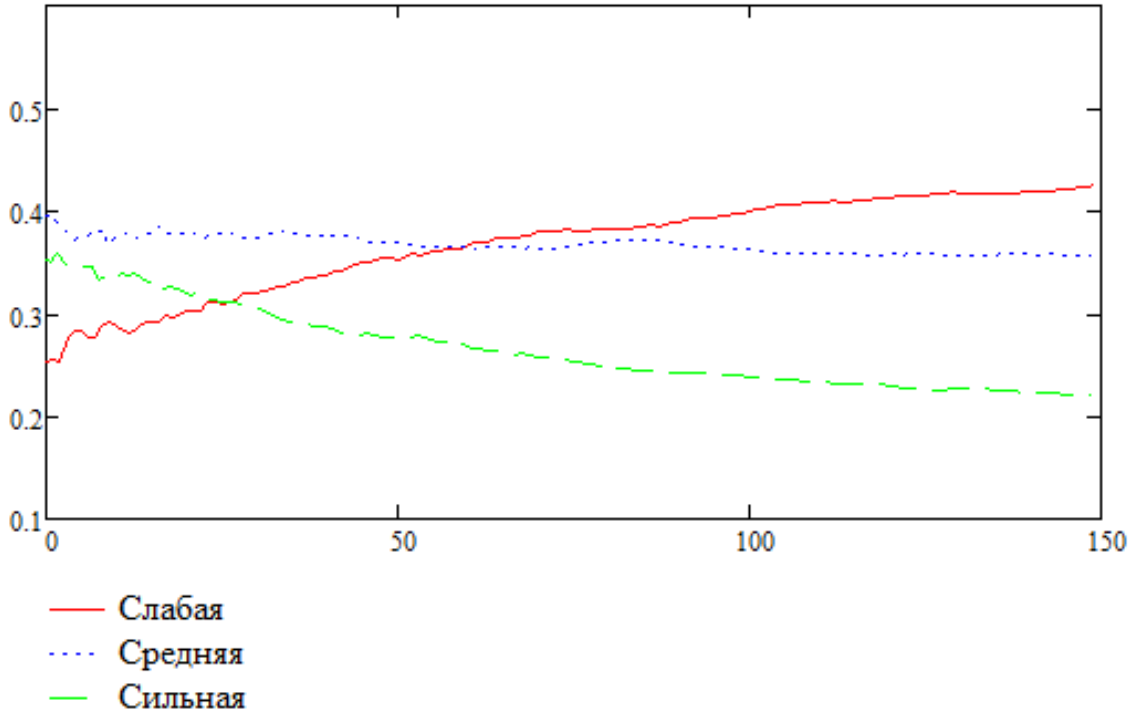


Рисунок 1.7. Изменение вероятностей применения операторов мутации, PDP

В целом, метод PDP демонстрирует те же тенденции для данной задачи, что и второй метод. То есть, среди типов селекции однозначно доминирует турнирная селекция, среди типов скрещивания нет однозначного лидера. Для мутации

ситуация чуть более интересная, на первых этапах работы алгоритма наибольшие вероятности имеют средняя и сильная мутация, в то время как по ходу эволюционного процесса все более востребованной и удачной оказывается слабая мутация. Этот пример наглядно демонстрирует, что в начале работы ГА эффективнее иметь больший разброс точек, чтобы охватить как можно большую область, так как таким образом, обеспечивается глобальность поиска. Однако на более поздних этапах, когда более востребована конфигурация алгоритма, подходящая для локального поиска, слабая мутация начинает доминировать, так как она снижает разброс точек в поисковом пространстве.

Также можно отметить, что метод PDP быстрее настраивается и меняет вероятности применения операторов. Второй метод делает изменения в вероятностях более медленно, однако эту скорость изменения можно настроить, меняя константу K в уравнении.

Аналогичное исследование было произведено для алгоритма генетического программирования для решения задачи восстановления символьной регрессии. В качестве тестовой задачи для сравнения методов самоконфигурации была взята задача классификации вин с репозитория UCI Machine Learning [27]. Данная задача содержит 13 переменных, 3 класса и 178 измерений и является не самой сложной для методов машинного обучения, но подходит для проведения сравнений.

Для сравнения со стандартным алгоритмом было проведено для каждой возможной комбинации операторов по 100 запусков алгоритма, ресурсы алгоритма составляли 100 индивидов и 500 поколений. Прочие настройки - тип инициализации, штраф на число отсутствующих переменных и длину дерева, максимальная начальная глубина, и другие, устанавливались вручную на основе нескольких предварительных запусков и были одинаковы в ходе всего сравнения.

Таблица 1.5 Расшифровка номеров конфигураций ГП

№ конфигурации	Селекция	Скрещивание	Мутация
1	Пропорциональная	Стандартное	Точечная
2	Пропорциональная	Стандартное	Выращиванием
3	Пропорциональная	Одноточечное	Точечная

Продолжение таблицы 1.5

4	Пропорциональная	Одноточечное	Выращиванием
5	Ранговая	Стандартное	Точечная
6	Ранговая	Стандартное	Выращиванием
7	Ранговая	Одноточечное	Точечная
8	Ранговая	Одноточечное	Выращиванием
9	Турнирная (3)	Стандартное	Точечная
10	Турнирная (3)	Стандартное	Выращиванием
11	Турнирная (3)	Одноточечное	Точечная
12	Турнирная (3)	Одноточечное	Выращиванием

По результатам сравнения можно построить следующий график, показывающий доли верно классифицированных объектов выборки в зависимости от номера конфигурации алгоритма, а также долю верно классифицированных объектов для методов самонастройки. Расшифровка номеров конфигурации представлена в таблице 1.5.

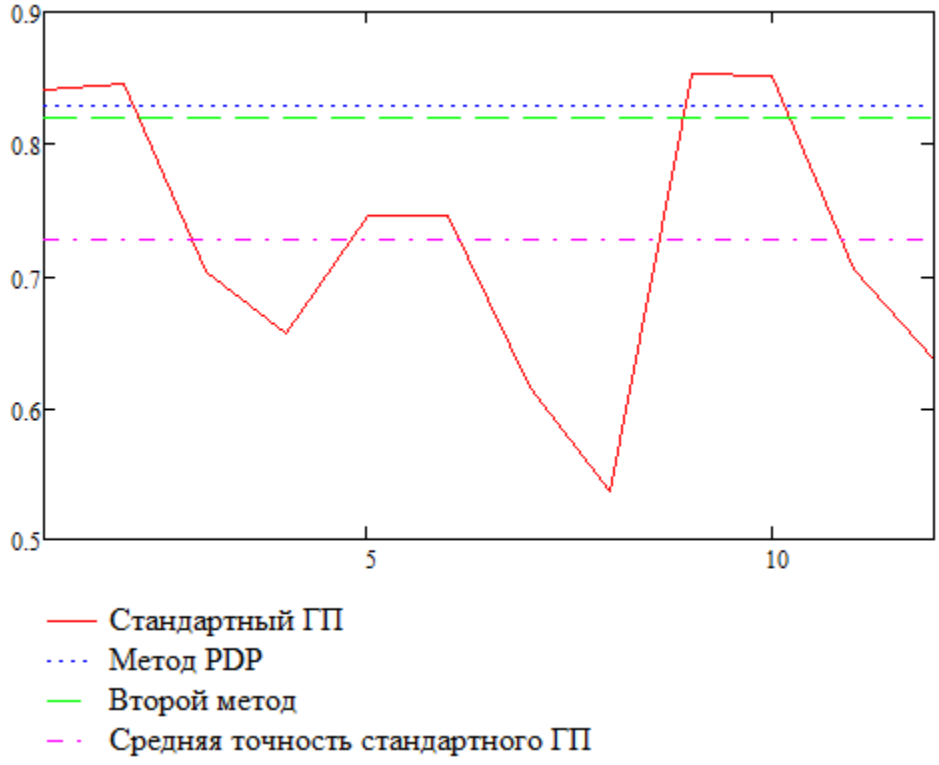


Рисунок 1.8. Сравнение самонастраивающегося и стандартного ГП

Из построенной зависимости можно заключить, что в среднем самоконфигурируемый алгоритм показывает худшие результаты, чем четыре лучших стандартных конфигурации, однако лучшие, чем стандартный алгоритм в среднем и со всеми остальными настройками. При этом стоит отметить, что

точность классификации выборки сильно зависит от выбранных операторов в данном случае. В сравнении с другими методами на данной задаче полученные значения точности могут показаться низкими, однако это объясняется малым объемом ресурсов алгоритма вследствие необходимости многократных запусков для усреднения точности и сравнения.

Метод PDP для ГП показал себя немного лучше, чем второй метод самонастройки в то время как для ГА ситуация была противоположной. Вследствие этого, нельзя сделать вывод о превосходстве одного метода самонастройки над другим. Вероятно, для некоторых эволюционных алгоритмов предпочтительнее первый метод, в то время как для других – второй.

1.6 NFL-теорема для задач оптимизации и самоконфигурация алгоритма

Проблема поиска оптимального набора параметров эволюционного алгоритма, который бы обеспечил высокую эффективность поиска активно исследовалась до появления работ Вольперта и Макреди, в которых формируется так называема NFL-теорема (No Free Lunch, бесплатных завтраков не бывает) в работах [120, 122], опубликованных в 1996 году. Данные публикации вызвали оживленную дискуссию в научном сообществе, так как одним из её следствий являлось то, что невозможно подобрать параметры генетического алгоритма так, чтобы он всегда давал лучшие результаты независимо от решаемой задачи. Эту теорему можно сформулировать следующим образом:

Пусть $P\left(d_m^y \middle| f, m, a\right)$ – условная вероятность получения частного решения d_m после m итераций алгоритма a при целевой функции f. Тогда для любой пары алгоритмов a_1 и a_2 имеет место равенство:

$$\sum_f P\left(d_m^y \middle| f, m, a_1\right) = \sum_f P\left(d_m^y \middle| f, m, a_2\right).$$

То есть, сумма условных вероятностей того, что алгоритм в пространстве решений посетит решение d_m одинакова для множества всех целевых функций, независимо от используемого алгоритма. Это означает, что не существует

универсального алгоритма, который был бы способен решать любые задачи, причем как эволюционного, так и любого другого (например, классического метода вроде метода деления отрезка пополам или метода Ньютона). Хотя, как уже упоминалось выше, в соответствии с утверждениями З. Михалевича, любой оптимизационный алгоритм можно считать эволюционным [84].

Данная теорема означает, что любые поиски лучших операторов для генетических алгоритмов не имеют смысла, так как их эффективность проверяется только на наборе тестовых задач. Даже если исследователи предпримут всевозможные попытки составить репрезентативный набор тестовых задач, в соответствии с NFL теоремой, это не означает, что на реальной задаче с неизвестной структурой пространства поиска алгоритм покажет лучшие результаты.

В качестве критики данной теоремы высказывается то, что на практике исследователи сталкиваются лишь с определенным классом задач, а не со всевозможными задачами. Однако даже сужение области возможных проблем не гарантирует получения преимущества данного метода по сравнению с другими.

По утверждению Вольперта и Макреди [121], «бесплатные обеды» возможны для коэволюционных алгоритмов, которые по своей сути являются мета-алгоритмами, использующими кооперацию либо конкуренцию между собственными элементами. Таким образом, ключом к решению рассмотренной проблемы может являться разработка гибридных алгоритмов, способных менять структуру поиска решений в зависимости от информации, которую они получают при решении проблемы. Иными словами, метод оптимизации, который мог бы на основании информации о предыдущих шагах поиска и собственной оценки структуры задачи, использовать соответствующий наиболее эффективный подход для дальнейшего решения, позволил бы решать эффективно более широкий круг задач, чем любой из его компонентов.

В соответствии с этим, видится разумным разработка адаптивных и самоконфигурирующихся оптимизационных процедур, способных менять свою структуру и принимать решение о применении того или иного метода

оптимизации на определенных шагах алгоритма. В соответствии с последними разработками в области методов оптимизации, можно заключить, что наилучшие результаты получаются при комбинировании нескольких методов оптимизации в коллектив, (как например метод COBRA [2, 20-23], комбинирующий 5 алгоритмов роя частиц с автоматическим перераспределением ресурсов), использовании коэволюции, либо же применении самоконфигурации, как в данной работе. Естественно, эффективность данных методов опять же проверялась на ограниченном наборе тестовых функций ввиду невозможности проверки на всех существующих функциях, однако полученные результаты дают основания утверждать о возможности разработки мета-алгоритмов, демонстрирующих хорошие результаты на подавляющем большинстве встречающихся оптимизационных проблем.

ВЫВОДЫ

В первой главе были рассмотрены два метода самоконфигурации эволюционных алгоритмов и применены для классических алгоритмов – генетического алгоритма и генетического программирования. Оба рассмотренных метода самоконфигурации хорошо показали себя при настройке вероятностей применения генетических операторов. В целом, подтвердилось известное правило, которое заключается в том, что методы самоконфигурации позволяют добиться лучших результатов, чем могут быть получены в среднем по всем стандартным конфигурациям алгоритма, однако лучшая комбинация параметров для данной задачи дает лучшую точность/надежность, чем алгоритм с самоконфигурацией. В следующей главе будет рассмотрена задача формирования баз нечетких правил при помощи самоконфигурирующихся эволюционных алгоритмов.

Глава 2. Классификация на нечеткой логике

2.1 Общие сведения о нечетких системах

Системы на нечеткой логике широко используются в задачах автоматического и автоматизированного управления. Понятие «нечеткий» имеет отношение к расширению принятых в математической логике понятий «ложь» и «истина» в ситуациях, когда необходимо обозначить частичную истинность. Несмотря на то, что другие виды интеллектуальных информационных технологий, такие как эволюционные алгоритмы и нейронные сети, позволяют решать те же задачи, что и нечеткая логика с примерно равной эффективностью, системы на нечеткой логике имеют одно важное преимущество – функционирование таких систем может быть объяснено на естественном, понятном человеку языке. Это также позволяет использовать знания человека – эксперта в предметной области, например, при проектировании нечеткого контроллера [117]. Таким образом, нечеткая логика позволяет легко автоматизировать решение тех задач, которые уже достаточно хорошо могут быть решены человеком. Более того, такие системы могут решать и задачи, которые значительно сложнее для решения человеком.

Первая работа и предложение идеи нечеткой логики было представлено Л. Заде в его работе в 1965 году [124]. В дальнейшем он же разработал понятия лингвистических переменных и весь основной математический аппарат нечеткой логики. Стоит отметить, что, несмотря на нечеткость, данный тип логики не использует аппарат теории вероятности.

Наибольшее распространение нечеткая логика получила в 70х и 80х годах в Японии, где была применена для решения множества задач управления. В их числе управление системами ускорения и торможения поездов, удержание перевернутого маятника на тележке, управление силой всасывания пылесоса, автофокусировка в фотоаппаратах, управление системами кондиционирования, распознавание символов, управление лифтами и многое другое.

В США и Европе нечеткие системы получили меньшее распространение, и, тем не менее, были использованы в следующих областях: система управления автоматической стыковкой в космосе, холодильники с низким потреблением энергии, автомобильные коробки переключения передач, посудомоечные машины и т.д.

Более того, нечеткие системы нашли применение в экспертных системах, системах поддержки принятия решений, а также были интегрированы с нейронными сетями [106, 72, 78, 73]. Отдельный класс алгоритмов формируют так называемые генетические нечеткие системы [59, 45], чья основная цель состоит в организации процесса самообучения системы.

Область применения нечеткой логики обширна, существует множество различных методов и подходов, основанных на этой идее. Однако в данной работе основной акцент будет сделан именно на генетических нечетких системах, то есть обучении нечеткой системы для решения практических задач. В частности, будут рассмотрены нечеткие системы классификации.

Нечеткой системой классификации называется классификатор, состоящий из нечетких правил. Нечеткое правило состоит из условия по типу «если… то…» с нечеткими условиями в части «если…» и соответствующим номером класса в части «то…». В качестве примера, можно привести следующую базу правил (классификация студентов):

1) Если знания физики хорошие и знания математики хорошие, то класс 1

2) Если знания физики удовлетворительные и знания математики хорошие, то класс 2

3) Если знания физики хорошие и знания математики удовлетворительные, то класс 2

4) Если знания физики удовлетворительные и знания математики удовлетворительные, то класс 3

Здесь класс 1 – хорошие студенты, класс 2 – студенты, подающие надежды, класс 3 – нехорошие студенты. Такую базу правил может составить практически любой, а аппарат нечеткой логики может помочь при формализации. К примеру,

знания по физике и математике можно задать в виде нечетких множеств имеющих по два терма – «хорошие» и «удовлетворительные», заданные на множестве оценок. По усредненным оценкам можно рассчитать степени принадлежности успеваемости конкретного студента по физике или математике к «хорошей» и «удовлетворительной». На основании этих степеней принадлежности студенту ставится в соответствие наиболее подходящий класс. Таким образом, вербально сформулированное знание может быть формализовано и использовано в автоматизированных системах классификации. Этап формирования нечетких множеств и функций принадлежности называется фаззификацией.

Рассмотрим вкратце концепцию лингвистической переменной [123, 83]. Лингвистической переменной называется переменная, которая может принимать значения фраз естественного или искусственного языка. Например, переменная «температура» может принимать значения «низкая», «очень низкая», «средняя», «высокая» и так далее. Каждое из этих значений описывается нечетким множеством или термом. Пример лингвистической переменной «рост человека» с тремя значениями представлен на рисунке 2.1.

Для решения задач классификации применяются базы правил – наборы нечетких правил, которые используются для определения номера класса при предъявлении ему конкретного объекта для классификации. Полные базы правил описывают все возможные комбинации значений входных лингвистических переменных. То есть, например, если классификацию необходимо провести по 7 лингвистическим переменным, каждая из которых содержит по 5 нечетких множеств, то полная база правил будет содержать 5^7 правил. Естественно, что столь большие базы правил сложны для восприятия, и их трудно построить. Поэтому чаще используют неполные базы, составленные из правил, которых достаточно для решения задачи. Размер неполной базы правил может варьироваться в зависимости от задачи, но в целях упрощения анализа число правил редко превышает несколько десятков.

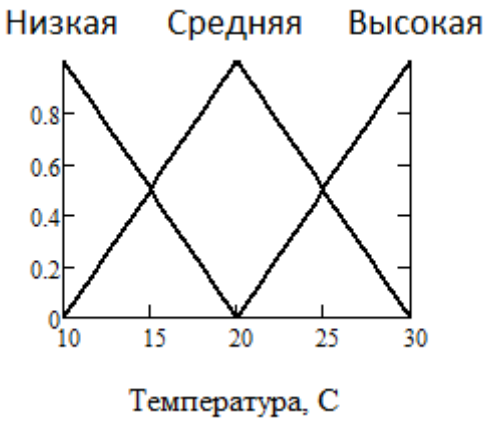


Рисунок 2.1. Лингвистическая переменная «Температура»

Из-за особенностей нечетких множеств, для определения номера класса по базе необходимо применять специализированные процедуры нечеткого вывода. Наиболее распространенной процедурой является вывод по Мамдани [83], который является наиболее прозрачным и простым. Хотя, например, в задачах управления чаще используется вывод по типу Тагаки-Сугено [114], который обладает большей точностью, но более сложен в интерпретации. В рамках данной работы будет использован вывод по Мамдани при решении задач классификации.

После этапа нечеткого вывода выполняется этап дефаззификации – перевода нечетких значений в четкие. В случае задачи классификации дефаззификация сводится к выбору класса, степень принадлежности к которому по данной базе правил максимальна. Более подробно каждый из этих этапов будет рассмотрен отдельно для каждого метода в последующих пунктах данной главы.

2.2 Построение базы нечетких правил классификации генетическими алгоритмами

В данном пункте будет рассмотрена постановка задачи классификации объектов при помощи нечетких баз правил. Также будут описаны два простейших подхода к формированию баз правил при помощи генетического алгоритма.

Классификация объектов состоит в приписывании объекту -мерного пространства R^F некоторого класса C_j из заранее заданного множества $C =$

$\{C_1, \dots, C_k\}$. Пусть X= $\{X_1, \dots, X_F\}$ –набор входных переменных задачи, а U_f, $f = 1, \dots, F$ – область определения -ой переменной. Пусть $P_f = \left\{A_{f,1}, \dots, A_{f,T_f}\right\}, f = 1, \dots, F$ – разбиение области определения U_f на T_f нечетких множеств.

Нечеткое правило R_m, $m = 1, \dots, M$, где M – число правил, обычно имеет вид:

$$R_m\text{: ЕСЛИ } X_1 \text{это } A_{1,j_{m,1}} \text{ и } \dots \text{и } X_F \text{это } A_{F,j_{m,F}} \text{ТО } Y \text{ это } C_{j_m},$$

где Y – выход, $C_{j_m} \in$ C – номер класса для m-го правила, а $j_{m,f} \in \left[1, T_f\right]$ - это номер нечеткого множества из разбиения P_f, выбранный для переменной X_f.

Таким образом, базу правил можно задать матрицей $J \in N^{M\times(F+1)}$

$$J = \begin{bmatrix} j_{1,1} & \cdots & j_{1,F} & C_{j_1} \\ \cdots & \cdots & \cdots & \cdots \\ j_{m,1} & \cdots & j_{m,F} & C_{j_m} \\ \cdots & \cdots & \cdots & \cdots \\ j_{M,1} & \cdots & j_{M,F} & C_{j_M} \end{bmatrix},$$

где элемент $j_{m,f}$ обозначает, что для правила R_m и переменной X_f было выбрано нечеткое множество $A_{f,j_{m,f}}$ и правилу поставлен в соответствие класс C_{j_m}.

Все числа в данной матрице целые, так что задача поиска оптимальной матрицы, описывающей нечеткую базу правил при заданном числе правил сводится к задаче целочисленной оптимизации. Данная задача может быть решена генетическим алгоритмом, наиболее важным моментом в этом случае является способ кодирования матрицы в хромосому. В дальнейшем в этой главе будет рассмотрено несколько методов кодирования, их преимущества и недостатки. Помимо кодирования в бинарную строку ГА, существуют подходы для представления базы правил в виде дерева в ГП, однако такие подходы сложнее в вычислительном смысле и потому реже используются.

Рассмотрим один из наиболее простых и логичных методов кодирования. В этом методе каждый из элементов матрицы J кодируется как отдельная переменная и далее все бинарные строки «склеиваются» и формируют хромосому. То есть, каждый элемент $j_{m,f} \in \left[1, T_f\right]$ кодируется в бинарную строку

длины l, такую, что $2^l \geq T_f + 1$. Добавление единицы к числу нечетких множеств необходимо для включения терма игнорирования значения переменной. То есть, если по переменной f для правила m имеется 7 нечетких множеств, то с учетом включения возможности игнорирования, бинарная строка для кодирования этого элемента матрицы должна кодировать 8 различных элементов, то есть её длина должна быть равна 3.

Терм игнорирования (чаще обозначается DC – «Don’t Care») означает, что для данной переменной в данном правиле нет разницы, какое значение приняла переменная. Включение этого терма позволяет существенно упростить сами правила, и, соответственно, базу правил в целом. То есть, к примеру, правило может содержать для всех переменных кроме одной игнорирование и соответствующий номер класса, и, тем не менее, быть достаточно точным. В случае, если не использовать игнорирование, для описания той же зависимости, необходимо было бы построить множество правил, описывающих все возможные комбинации остальных переменных и содержащие всё тот же номер класса для каждого из этих правил. Правила с игнорированием являются более простыми, общими и позволяют компактно описывать значительную часть искомой зависимости.

Графическое представление кодирования правила в хромосому показано на рисунке 2.2.

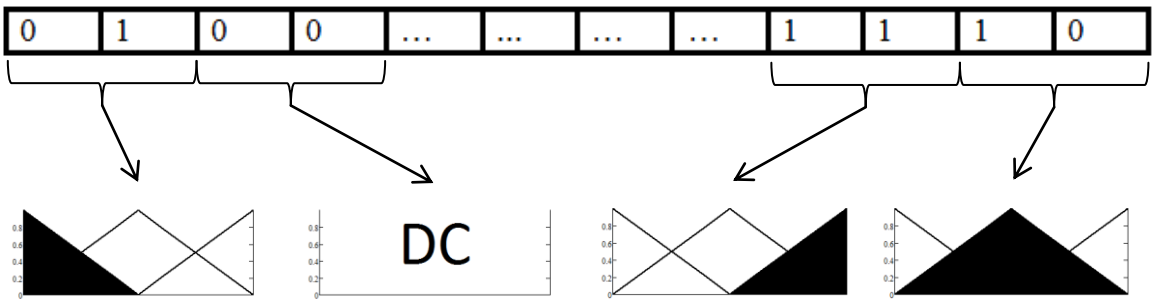


Рисунок 2.2. Кодирование правила в хромосому, первый метод

Каждому правилу ставится в соответствие номер класса, который кодируется аналогичным образом. Если для каждой из F переменных установить

по t нечетких термов, при том, что в задаче классификации только два класса, то длина хромосомы будет равна $(t+1)\cdot F+2$.

Рассмотрим альтернативный подход к кодированию и представлению базы правил в виде хромосомы. В данном методе каждое из правил в левой части для каждой переменной может содержать по несколько нечетких множеств одновременно, в отличие от предыдущего метода. То есть, правила строятся по принципу включения или невключения конкретного нечеткого терма в правило. Различные термы для одной переменной в правиле объединяются по типу операции «ИЛИ», таким образом каждое правило в данном методе равносильно нескольким правилам в предыдущем методе. Каждый бит в хромосоме кодирует наличие или отсутствие данного терма в правиле. Графическое представление правила, его расшифровка показаны на рисунке 2.3.

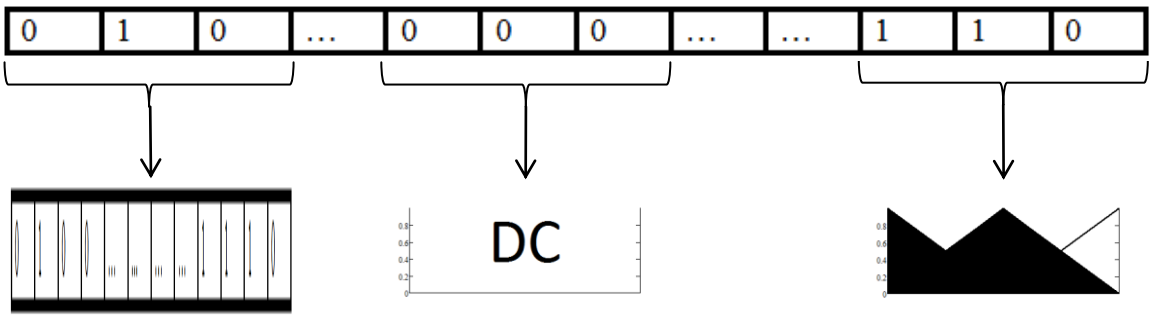


Рисунок 2.3. Кодирование правила в хромосому, второй метод

В данном методе длина хромосомы, необходимая для кодирования правила получается больше, чем в предыдущем методе, при том же числе переменных F и термов для каждой переменной T_f.

Процедура классификации объекта по базе правил в обоих методах производится по принципу определения правила – победителя. Для определения этого правила необходимо для каждого из правил R_m рассчитать минимум по степеням принадлежности по всем переменным $\min_f\left(\mu_{j_{m,f}}(x_f)\right)$. Правило с наибольшим значением функции принадлежности будет считаться победителем,

объекту приписывается номер класса, соответствующий этому правилу. Итоговое выражение можно записать следующим образом:

$$w = arg\max_{m}\left(\min_{f}\left(\mu_{j_{m,f}}(x_f)\right)\right),$$

где w – номер правила-победителя. Данная процедура схожа с процедурой вывода по Мамдани, которая используется при нечетком выводе в задачах управления, где выходы являются вещественными. По своей сути процедура определения правила-победителя отличается упрощенным способом дефаззификации вследствие дискретности выхода в задачах классификации (номер класса).

Выше были рассмотрены методы формирования базы правил посредством выбора лучшей комбинации номеров используемых в правилах термов. Это относительно простой, но не единственный подход. Одним из его недостатков является то, что функции принадлежностей в нем фиксированы и не изменяются в ходе работы алгоритма, то есть не настраиваются.

В реальных задачах, когда присутствуют вещественные переменные, определить заранее наилучшее число термов для них бывает проблематично. Более того, равномерное распределение термов по области значений переменной может быть не лучшим вариантом. То есть, к примеру, для повышения качества классификации может потребоваться смещение и изменение формы функций принадлежности. В описанных в предыдущем пункте алгоритмах такой возможности нет.

Среди путей решения данной проблемы можно выделить использование нескольких разбиений на нечеткие множества одновременно с тем, чтобы алгоритм мог выбирать не только номер нечеткого множества в разбиении, но и число нечетких множеств, на которое разбивается область значений переменной. Данный метод показал относительно высокую эффективность и будет подробно рассмотрен в следующей главе.

Другой способ повышения качества классификации заключается в подстройке положений функций принадлежности для имеющейся базы правил. То есть, к примеру, если есть некоторая база правил, относительно хорошо

решающая задачу, то можно попытаться задействовать некоторую оптимизационную процедуру, чьими параметрами будут положения функций принадлежности, для того чтобы сместить и изменить форму нечетких множеств и повысить качество классификации. Таким образом, можно получить двухступенчатую процедуру, в которой на первом этапе формируются базы правил, а на втором настраиваются формы и положения нечетких термов.

Ещё один способ состоит в формировании базы правил только настройкой положений термов, без их выбора. В этом случае вся задача формирования нечеткой базы правил сводится к задаче безусловной вещественной оптимизации. Рассмотрим пример такого метода подробно.

Зададим для каждой переменной в каждом правиле базы по две функции, описывающие степени принадлежности сигмоидального типа. Зададим сигмоидальную функцию таким образом, чтобы она имела 3 аргумента x, y и z. Здесь x – текущее значение переменной, y – положение на оси X, в котором функция принимает нулевое значение, z – положение на оси X, в котором функция принимает единичное значение. Концы функции округлены до 0 и 1 после y и z. Функции представляются формулой

$$s(x,y,z) = \begin{cases} 1\ if (x \geq y\ and\ y \geq z) or (x \leq y\ and\ z > y), \\ 0\ if (x \leq z\ and\ y \geq z) or (x \geq z\ and\ z > y), \\ \dfrac{1}{1 + e^{\frac{-c}{y-z}\left(x + \frac{z-y}{2} - \min(y,z)\right)}}\ if\ y > z, \\ \dfrac{1}{1 + e^{\frac{-c}{y-z}\left(x - \frac{z-y}{2} - \min(y,z)\right)}}\ if\ z > y, \end{cases}$$

$$if\ y = z\ then\ y \leftarrow y + \frac{k}{c};\ z \leftarrow z - \frac{k}{c}$$

Константы имеют значения $c = 15, k = 0.1$. Пример двух таких функций представлен на рисунке 2.4.

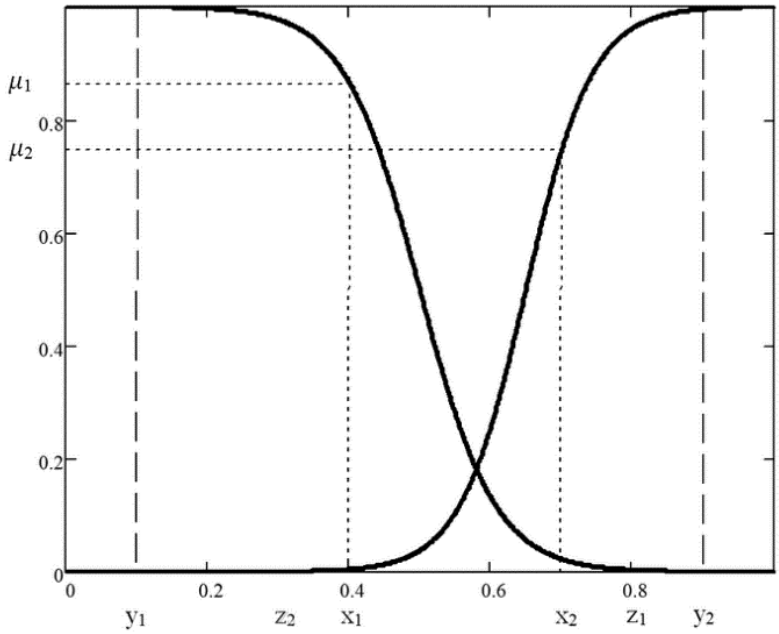


Рисунок 2.4. Сигмоидальные функции

Функция была модифицирована для того, чтобы избежать необходимости решения задачи условной оптимизации, так как значения y и z лежат в области определения переменной и могут быть использованы напрямую в алгоритме оптимизации.

Пара функций для каждой из переменных кодирует единственный нечеткий терм для конкретной переменной конкретного правила. Однако различные комбинации положений сигмоидальных функций позволяют получать всевозможные формы нечетких термов, обычно недостижимые при использовании других алгоритмов. При этом в процедуре нечеткого вывода используется минимум из значений двух функций принадлежности. Примеры возможных вариантов расположения двух сигмоидальных функций и соответствующие формы нечетких множеств (выделены жирным) представлены на рисунке 2.5.

Как можно видеть из рисунка, комбинация двух сигмоид позволяет получать классические формы нечетких термов, как например варианты 3, 4, 6, так и редко используемые формы, как 1 или 5.

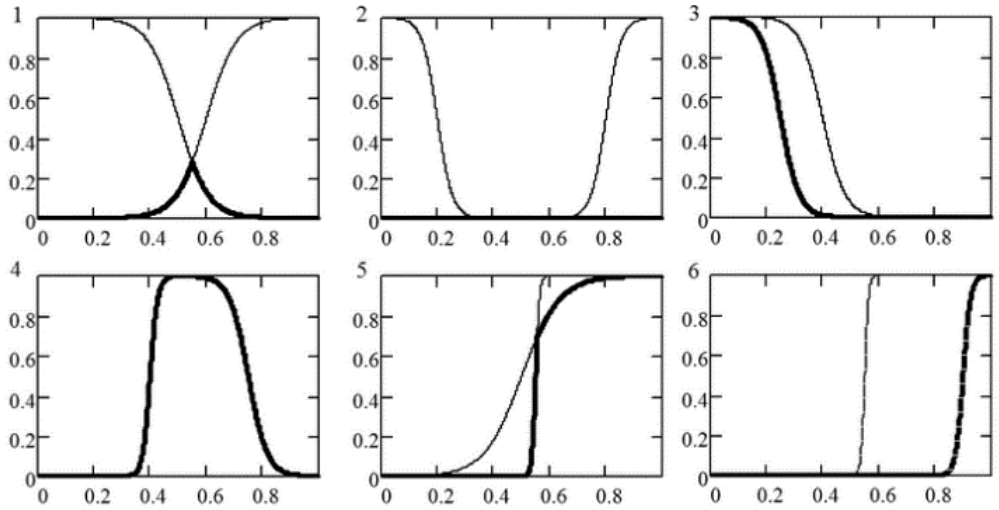


Рисунок 2.5. Примеры возможных форм нечетких термов, получаемых при комбинации сигмоидальных функций

Вариант 2 ($\mu = \min\big(s(x_1, y_1, z_1), s(x_1, y_2, z_2)\big) = 0$ для всех x) представляет особый интерес, так как в этом случае при любом значении переменной процедура нечеткого вывода будет получать нулевое значение функции принадлежности. Этот вариант в алгоритме рассматривается как игнорирование значений переменной, и в этом случае возвращаемое значение степени принадлежности равно единице.

В данном алгоритме также используется определение номера класса по принципу правила-победителя. В общем виде эту процедуру можно записать следующим образом:

$$R = \underset{i \in N}{\operatorname{argmax}}\left(\min_{j \in M}\left(\min\left(s\left(X_j, y_{i,j}^1, z_{i,j}^1\right), s\left(X_j, y_{i,j}^2, z_{i,j}^2\right)\right)\right)\right)$$

Где N – число правил, M – число переменных в задаче.

Генетический алгоритм кодирует значения y^1, z^1, y^2, z^2 для каждой переменой каждого терма, таким образом, общее число переменных равно $4 \cdot N \cdot M$. Стоит отметить, что в этом подходе не используется некоторого общего набора функций принадлежности, каждое правило получает свою комбинацию сигмоид. Номер класса кодируется отдельно для каждого правила.

В следующем пункте данной главы будут описаны особенности программной реализации изложенных алгоритмов, а также приведены результаты работы методов на нескольких задачах классификации и их сравнение.

2.3 Программная реализация методов построения баз нечетких правил и их сравнение на реальных задачах

Программная реализация разработанных алгоритмов была проведена в среде CodeBlocks 12.11 на языке C++ без использования средств объектно-ориентированного программирования и визуальных оболочек с целью ускорения работы программ.

В программных системах была предусмотрена возможность разбиения выборки на обучающую и тестовую в различных соотношениях. Разбиение производилось случайным образом каждый раз при каждом запуске алгоритма. Распределения классов в тестовой и обучающей выборке не учитывались. Для получения более достоверных результатов значения точности классификации на обучающей и тестовой выборке усреднялись по 25 запускам алгоритма.

Для упрощения получаемых алгоритмом баз правил использовалась модифицированная функция пригодности, включающая помимо числа неверно классифицированных объектов также коэффициент, определяющий число игнорированных термов:

$$Fitness = Error + \frac{T_{total} - T_{ignored}}{T_{total} \cdot 10}.$$

Где $Error$ – число неверно классифицированных объектов обучающей выборки (всегда целое), $T_{ignored}$ – число игнорированных термов, T_{total} – общее число термов, которые могут быть включены во всех правилах. Коэффициент 10 в делителе гарантирует, что отношение всегда будет меньше единицы, таким образом, всегда будут предпочитаться более точные решения, однако среди решений с одинаковой точностью преимущество будут иметь более простые.

Тестирование производилось на 14 задачах с репозитория UCI Machine Learning [27]. Среди них:

1) Banknote – задача об определении достоверности банкнот по изображениям.
2) Column 2c - задача классификации заболеваний позвоночника, 2 класса.
3) Column 3c – задача классификации заболеваний позвоночника, 3 класса.
4) Iris – задача классификации ирисов по форме листков.
5) Liver – задача о заболеваниях печени.
6) Seeds – задача классификации зёрен по их форме.
7) Austrlian credit – задача о подозрительных операциях с банковскими картами.
8) German credit – задача о классификации клиентов банка.
9) Breast cancer – задача о раке груди.
10) Heart – задача о заболеваниях сердца.
11) Glass – задача классификации типов стекла.
12) Ionosphere – классификация сигналов радаров отраженных от ионосферы.
13) Pima – заболевание диабетов у индейцев Пима.
14) Wine – классификация происхождения вина по данным химического анализа.

Описание характеристик выборок представлено в таблице 2.1.

Таблица 2.1. Задачи классификации, взятые с репозитория UCI

Выборка	Число измерений	Число переменных	Число классов
Australian credit	690	14	2
Banknote	1372	4	2
Breast cancer	683	9	2
Column 2c	310	6	2
Column 3c	310	6	3
German credit	1000	24	2
Glass	214	9	6
Heart	270	13	2
Ionosphere	351	34	2
Iris	150	4	3
Liver	345	6	2
Pima	768	8	2
Seeds	210	7	3
Wine	178	13	3

Всего тестировалось три алгоритма – метод с классической кодировкой (FUZZY_GA_OCD), с альтернативной кодировкой (FUZZY_GA_NCD) и с настройкой положений термов. Для всех трех алгоритмов использовался метод самонастройки PDP. Методы работали в практически одинаковых условиях – с одинаковым ресурсом и числом правил. Число индивидов устанавливалось равным 200, число поколений – 500, максимальное число правил – 20.

В таблице 2.2 приведены значения точности, то есть отношения доли верно классифицированных объектов к их общему числу как для обучающей, выборки для всех перечисленных задач для первых двух алгоритмов. Жирным выделен победивший для данной задачи алгоритм(ы).

По приведенным результатам можно судить о том, что первый метод, с классической схемой кодировки показывает значительно лучшие результаты на большинстве задач. Однако существуют примеры, для которых второй метод превосходит первый, хотя это превосходство незначительно. Аналогичные результаты для тестовой выборки представлены в таблице 2.3.

Таблица 2.2. Точность первых двух методов на обучающей выборке

Выборка	FUZZY_GA_OCD	FUZZY_GA_NCD
Australian credit	0.894	**0.900**
Banknote	**0.952**	0.899
Breast cancer	0.976	0.979
Column 2c	0.805	0.801
Column 3c	0.711	**0.730**
German credit	**0.788**	0.736
Glass	**0.619**	0.599
Heart	0.893	**0.907**
Ionosphere	**0.833**	0.702
Iris	**0.969**	0.823
Liver	0.665	0.669
Pima	**0.783**	0.771
Seeds	**0.946**	0.828
Wine	**0.977**	0.884

Таблица 2.3. Точность первых двух методов на тестовой выборке

Выборка	FUZZY_GA_OCD	FUZZY_GA_NCD
Australian credit	0.839	**0.844**
Banknote	**0.947**	0.893
Breast cancer	0.952	0.951
Column 2c	**0.773**	0.751
Column 3c	0.668	**0.673**
German credit	**0.707**	0.696
Glass	**0.532**	0.483
Heart	0.758	**0.788**
Ionosphere	**0.747**	0.631
Iris	**0.939**	0.777
Liver	0.567	**0.593**
Pima	**0.746**	0.701
Seeds	**0.874**	0.773
Wine	**0.880**	0.761

В целом тенденция на тестовой выборке повторяет таковую на обучающей выборке, хотя стоит отметить, что второй алгоритм показал лучшие результаты. Вероятно, это происходит потому, что второй метод может строить более полные правила, покрывающие большую часть признакового пространства.

Таблица 2.4. Точность всех трех методов на обучающей выборке

Выборка	FUZZY_GA_OCD	FUZZY_GA_NCD	FUZZY_GA_ADJ
Australian credit	0.903	**0.914**	0.909
Banknote	0.952	0.901	**0.999**
Breast cancer	0.979	0.982	**0.989**
Column 2c	0.801	0.802	**0.931**
Column 3c	0.719	0.743	**0.929**
German credit	**0.822**	0.794	0.806
Glass	0.677	0.654	**0.776**
Heart	0.915	**0.937**	0.929
Ionosphere	**0.923**	0.845	0.864
Iris	0.968	0.821	**0.998**
Liver	0.671	0.669	**0.854**
Pima	0.787	0.780	**0.853**
Seeds	0.941	0.838	**0.992**
Wine	0.986	0.971	**0.995**

Аналогичные результаты были получены при больших ресурсах алгоритмов для всех задач и сравнены с третьим алгоритмом (таблица 2.5), настраивающим функции принадлежности (который обозначен FUZZY_GA_ADJ). Число индивидов устанавливалось равным 700, число поколений – 700. Число индивидов и поколений для последнего метода также равнялось 700, а максимальное число правил устанавливалось равным 10.

Таблица 2.5. Точность всех трех методов на тестовой выборке

Выборка	FUZZY_GA_OCD	FUZZY_GA_NCD	FUZZY_GA_ADJ
Australian credit	0.840	0.836	**0.852**
Banknote	0.942	0.894	**0.995**
Breast cancer	0.949	**0.954**	**0.956**
Column 2c	0.777	0.771	**0.807**
Column 3c	0.642	0.706	**0.803**
German credit	**0.707**	0.707	**0.709**
Glass	0.536	0.491	**0.600**
Heart	**0.770**	0.764	0.759
Ionosphere	**0.798**	0.695	0.767
Iris	0.936	0.761	**0.943**
Liver	0.568	0.573	**0.629**
Pima	**0.732**	0.694	**0.732**
Seeds	0.883	0.777	**0.917**
Wine	**0.877**	0.836	0.851

Как можно видеть из таблиц, несмотря на то, что для последнего метода было выделено в 2 раза меньше правил, он показал лучшие результаты практически на всех задачах, причем как на обучающей, так и на тестовой выборке. Столь высокое качество работы последнего алгоритма объясняется возможностью настраивать расположение нечетких термов в соответствии с расположением точек в пространстве признаков. В этом смысле данный подход подобен нейронным сетям, в которых также используются сигмоидальные функции в качестве функций активации нейронов.

Стоить отметить, что, например, задачу banknote последний алгоритм решает практически с нулевой ошибкой – её величина как на обучающей, так и на тестовой выборке стремится к нулю. Однако, несмотря на то, что на задачах iris,

seeds и wine наблюдается похожая картина на обучающей выборке, точность на тестовой выборке оказывается значительно ниже.

Недостатком последнего метода является слабая интерпретабельность получаемых баз правил. В них отсутствуют привычные понятия по типу «низкий», «средний», и «высокий», функции принадлежности смешены и имеют различные формы, так что переводить в естественный язык такие базы правил – проблематично. Рассмотрим полученные базы правил на примере задачи Iris. База правил для первого метода, FUZZY_GA_OCD, приведена в таблице 2.6.

Таблица 2.6. Пример полученной базы правил для задачи Iris, первый метод

№	Вход	1	2	3	4		Класс *C*
1	Если	2	-	-	2	то	2
2	Если	-	0	-	1	то	0
3	Если	-	0	-	-	то	1
4	Если	-	-	-	-	то	1
5	Если	-	-	-	2	то	2
6	Если	-	2	-	-	то	2
7	Если	-	-	-	0	то	1
8	Если	-	-	0	-	то	0
9	Если	-	-	-	-	то	0
10	Если	-	-	-	-	то	0
11	Если	-	-	-	-	то	1
12	Если	-	-	-	-	то	0
13	Если	-	2	-	-	то	1
14	Если	-	-	2	-	то	2
15	Если	2	-	0	-	то	0
16	Если	0	-	1	-	то	1
17	Если	-	-	-	-	то	0
18	Если	-	-	0	-	то	0
19	Если	-	-	1	-	то	1
20	Если	-	-	0	-	то	1

Точность базы на обучающей выборке 0.978, на тестовой - 0.927.

База правил для второго метода, обозначенного FUZZY_GA_NCD, приведена в таблице 2.7.

Таблица 2.7. Пример полученной базы правил для задачи Iris, второй метод

№	Вход	1	2	3	4		Класс *C*
1	Если	-	1	-	2	то	2
2	Если	2	0	2	-	то	0
3	Если	-	-	-	-	то	2
4	Если	2	-	-	-	то	0
5	Если	-	-	-	-	то	2
6	Если	-	-	2	2	то	2
7	Если	-	1	-	0	то	1
8	Если	-	0	0	-	то	2
9	Если	1	-	2	-	то	0
10	Если	-	2	-	-	то	2
11	Если	-	2	1	-	то	2
12	Если	2	-	-	-	то	0
13	Если	-	1, 2	-	-	то	0
14	Если	-	0	2	0, 2	то	2
15	Если	-	-	-	-	то	2
16	Если	-	-	-	1	то	0
17	Если	-	1	-	-	то	1
18	Если	-	1	0, 1, 2	-	то	0
19	Если	-	1	-	-	то	1
20	Если	0, 2	2	-	0	то	0

Точность на обучающей выборке: 0.827, на тестовой выборке: 0.807.

Полученные базы правил характеризуются относительно небольшим числом правил и для первых двух подходов могут быть достаточно легко интерпретированы, однако высокая вычислительная сложность является основным препятствием к их применению. Их преимуществом является то, что вместо генетического алгоритма может быть использован любой другой алгоритм бинарной оптимизации (для первых двух методов) или же даже вещественной оптимизации.

ВЫВОДЫ

В данной главе были рассмотрены несколько простейших методов построения баз нечетких правил для задачи классификации. Данные методы легко реализуются и сводятся в общем случае к задачам бинарной либо вещественной

оптимизации. Таким образом, решение задачи построения базы нечетких правил в данных методах сводится к построению алгоритмически заданной функции пригодности для генетического алгоритма, а также заданию параметров алгоритма. В качестве алгоритмов оптимизации могут быть использованы и другие подходы, позволяющие эффективно решать задачи бинарной или вещественной оптимизации, такие как стайные алгоритмы, эволюционные стратегии или иные методы оптимизации, включая многокритериальные подходы, или же классические методы оптимизации.

Из-за того, что реализация данных методов является универсальной и по своей сути не зависит от метода оптимизации, их эффективность получается относительно невысокой в сравнении со специализированными эволюционными алгоритмами для построения систем на нечеткой логике. Один из таких подходов будет рассмотрен в следующей главе.

Глава 3. Специализированный гибридный эволюционный алгоритм построения баз нечетких правил

3.1. Определение весовых коэффициентов правил и наилучшего соответствующего номера класса

Несмотря на то, что нечеткие правила имеют высокую обобщающую способность, их возможности ограничены расположением функций принадлежности на областях определения переменных-признаков. Одна из первых работ в области генетических нечетких систем [117] использует те же правила, что были показаны в предыдущей главе. Они во многих случаях позволяют точно классифицировать входные данные при наличии удачного разбиения (грануляции) области значений переменных на нечеткие множества.

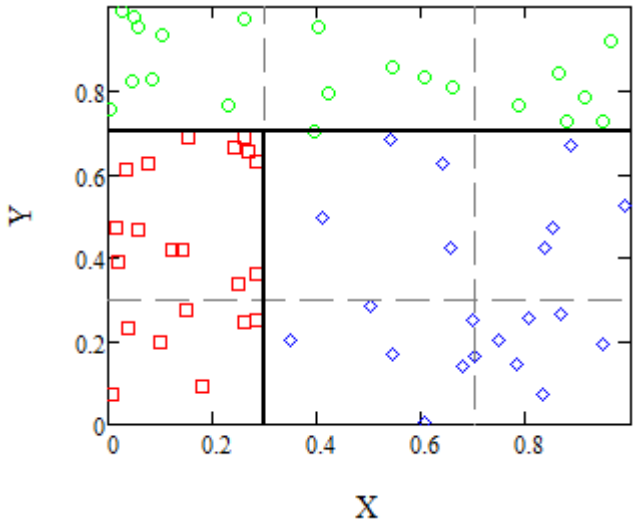


Рисунок 3.1. Пример классификации с удачным разбиением на нечеткие множества

Однако в некоторых случаях точность, получаемая классификаторами с правилами подобного вида, может быть относительно низкой. Это происходит ввиду такого распределения точек в пространстве признаков, при котором стандартное равномерное разбиение пространства поиска на нечеткие множества оказывается неудачным и не отражающим реальное распределение точек в соответствии с их классами. При этом даже изменение грануляции по переменным не всегда может решить эту проблему.

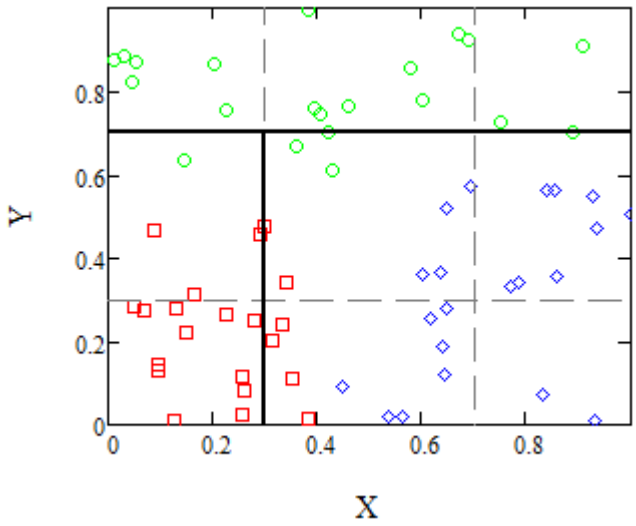


Рисунок 3.2. Пример классификации с неудачным разбиением на нечеткие множества

Разрешить данную проблему можно введением весовых коэффициентов для нечетких правил, которые влияют на процедуру нечеткого вывода и определения правила-победителя. То есть, чем выше вес правила (Certainty Factor, CF), тем больше у него возможностей стать победителем. Использование весов повышает гибкость классификатора, не меняя при этом грануляцию и расположение нечетких множеств, то есть, оставляя понятность базы правил на прежнем уровне.

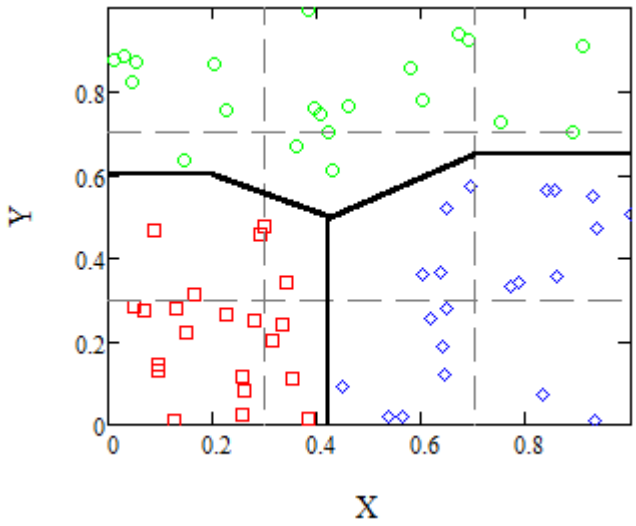


Рисунок 3.3. Пример классификации с использованием весов

Существуют и другие методы, помимо задания весов правил, которые изменяют форму и положение нечетких множеств. Среди них можно отметить минимаксные нейронные сети [106, 72, 78] а также методы, основанные на

определении положений нечетких множеств из выборки [18]. В этих методах каждое правило может быть снегерировано и настроено независимо от других, что дает высокую точность классификации. Однако зачастую в полученном классификаторе нечеткие множества значительно пересекаются и накладываются друг на друга. В такой ситуации их интерпретируемость значительно падает, так как возникает сложность в определении понятий «низкий», «средний», «высокий» и т.п. Тем же недостатком обладают и методы, в которых используется настройка функций принадлежности после формирования классификатора ради повышения качества классификации.

Несмотря на то, что было предложено несколько методов объединения сильно пересекающихся нечетких множеств после их настройки, они сохраняют тот же недостаток, что и остальные подобные методы, хотя и движутся в направлении некоторого компромисса между точностью и понятностью баз правил. В дальнейшем в данной работе будут рассматриваться лишь методы, не меняющие положения функций принадлежности, но использующие веса правил.

Назначение весов правил может быть произведено несколькими методами, среди которых можно выделить эвристические методы и методы оптимизации. Эвристические методы определяют вес правил на основании того, насколько хорошо правила покрывают выборку данных, используя при этом различные значения, вычисляемые по выборке. Оптимизация подразумевает настройку весов правил в классификаторе так, чтобы максимально повысить качество классификации.

Рассмотрим более подробно эвристические методы назначения весов правил. Пусть, как и раньше, классификация объектов состоит в приписывании объекту -мерного пространства R^F некоторого класса C_j из заранее заданного множества $C = \{C_1, \dots, C_k\}$. Пусть X= $\{X_1, \dots, X_F\}$ –набор входных переменных задачи, а U_f, $f = 1, \dots, F$ – область определения -ой переменной. Пусть $P_f = \left\{A_{f,1}, \dots, A_{f,T_f}\right\}, f = 1, \dots, F$ – разбиение области определения U_f на T_f нечетких множеств.

Нечеткое правило R_m, $m = 1, \dots, M$, где M – число правил, которые имеют вид:

$$R_m: \text{ЕСЛИ } X_1 \text{это } A_{1,j_{m,1}} \text{ и } \dots \text{и } X_F \text{это } A_{F,j_{m,F}} \text{ТО } Y \text{ это } C_{j_m} \text{ с весом } CF_{jm},$$

где Y – это выход, $C_{j_m} \in C$ – это номер класса для j_m-го правила, а $j_{m,f} \in [1, T_f]$ – это номер нечеткого множества из разбиения P_f, выбранный для переменной X_f.

Каждое нечеткое правило может быть рассмотрено как ассоциативное правило [19], в котором номер класса следует из значений, которые приняли переменные, то есть $A_q \to C_q$, где $A_q = (A_{q1}, \dots, A_{qn})$.

Таким образом, базу правил можно задать матрицей $J \in N^{M \times (F+2)}$

$$J = \begin{bmatrix} j_{1,1} & \dots & j_{1,F} & C_{j_1} & CF_{j_1} \\ \dots & \dots & \dots & \dots & \dots \\ j_{m,1} & \dots & j_{m,F} & C_{j_m} & CF_{j_m} \\ \dots & \dots & \dots & \dots & \dots \\ j_{M,1} & \dots & j_{M,F} & C_{j_M} & CF_{j_M} \end{bmatrix}.$$

Правило-победитель в этом случае определяется с учетом весов:

$$w = arg\max_m \left(CF_{j_m} \cdot \prod_f \left(\mu_{j_{m,f}}(x_f) \right) \right).$$

Вес правила может быть вычислен с использованием двух мер, часто применяемых при оценке ассоциативных правил [19] – *confidence* и *support*. *Confidence*, или достоверность нечеткого правила, может быть записана следующим образом [71]:

$$c(A_q \to C_q) = \frac{\sum_{x_p \in Class\ C_q} \mu_{A_q}(x_p)}{\sum_{p=1}^{m} \mu_{A_q}(x_p)}.$$

Достоверность может быть также рассмотрена как численное приближение к условной вероятности. Поддержка (*support*) правила $A_q \to C_q$ может быть записана следующим образом:

$$s(A_q \to C_q) = \frac{\sum_{x_p \in Class\ C_q} \mu_{A_q}(x_p)}{M}$$

Поддержку *support* можно рассматривать как меру покрытия обучающей выборки соответствующим правилом.

Номер класса, соответствующий нечеткому правилу в прошлой главе кодировался в хромосому ГА. Однако, в данном случае возможна ситуация, при которой правило, хорошо описывающее некоторый класс, может быть исключено из рассмотрения, так как в хромосоме закодирован неподходящий для него класс. Это является одной из основных причин низкой эффективности подходов, описанных в данной главе. Величины поддержки и достоверности правил могут быть использованы не только для расчета весов правил, но также и для определения номера класса, в наибольшей степени соответствующего конкретному правилу. Для этого необходимо найти такой номер класса, для которого значение *confidence* максимально для заданной левой части правила A_q:

$$c(A_q \to C_q) = \max\{c(A_q \to Class\ g) | g = 1,2 \dots k\}$$

Достоверность может быть напрямую использована как вес правила:

$$CF_q = c(A_q \to C_q).$$

Хотя есть и другие методы. Например, в [67] был использован следующий подход:

$$CF_q = c(A_q \to C_q) - c_{ave},$$

где c_{ave} это средняя поддержка среди всех правил с той же левой частью A_q, но различными номерами классов C_q:

$$c_{ave} = \frac{1}{k-1} \sum_{\substack{h=1 \\ h \neq C_q}}^{k} c(A_q \to Class\ g).$$

В работе [71] было предложено ещё два эвристических метода оценки весов правил. В первом из них из максимального значения достоверности вычитается второе по величине:

$$CF_q = c(A_q \to C_q) - c_{2nd},$$

где

$$c(A_q \to C_q) = \max\{c(A_q \to Class\ g) | g = 1,2 \dots k, g \neq C_q\}$$

Во втором предложенном методе значение CF может быть отрицательным:

$$CF_q = c(A_q \to C_q) - c_{sum},$$

где c_{sum} это сумма достоверностей по всем правилам с тем же A_q, но отличающимися номерами классов C_q:

$$c_{sum} = \sum_{\substack{h=1 \\ h \neq C_q}}^{k} c\left(A_q \rightarrow Class\ g\right)$$

Нечеткие правила с отрицательными весами неприменимы для классификации. Стоит отметить, что три последних метода задания весов совпадают в случае, если всего имеется 2 класса, то есть $k = 2$, в этом случае $c_{ave} = c_{2nd} = c_{sum}$. Последний метод назначения весов может быть представлен в другой форме ради упрощения расчетов:

$$CF_q = 2 \cdot c\left(A_q \rightarrow C_q\right) - 1.$$

В дальнейшем будет использоваться последний метод назначения весов, так как он является наиболее «сильным» и позволяет отсеивать значительное множество неперспективных правил, что было показано в [71].

3.2 Методы комбинирования Мичиганского и Питтсбургского подходов

При решении задач классификации с множеством переменных число возможных правил значительно возрастает, как и число возможных их комбинаций. Поэтому для формирования базы правил необходим некоторый алгоритм, который бы позволял одновременно и строить отдельные правила, и объединять их в базы, эффективно решающие поставленную задачу.

В предыдущей главе было рассмотрено несколько методов Питтсбургского типа, которые формируют базы нечетких правил целиком, не концентрируясь на конкретных правилах и не рассматривая их. Однако, эффективность такого подхода может быть достаточно ограничена, более того, он зачастую требует значительных вычислительных ресурсов.

Что касается методов Мичиганского типа, их отдельное рассмотрение несколько затруднено вследствие того, что эти методы не позволяют построить классификатор, но формируют лишь отдельные правила.

Рассмотрим для начала так называемую двухступенчатую схему, в которой методы Мичиганского и Питтсбургского типа работают поочередно. Эта схема подразумевает формирование некоторого набора из большого числа перспективных правил на первом этапе, после чего на втором этапе из этого набора необходимо выбрать несколько правил и объединить их в базу. Первый этап называется извлечением правил, второй – выбором правил.

Алгоритмы Мичиганского типа могут быть успешно применены в задачах онлайн-обучения, то есть когда новые измерения поступают непрерывно. В этом случае вся популяция в ГА, в котором каждый индивид представляет собой отдельное правило, формирует классификатор. При поступлении новых измерений и данных такая система может легко адаптироваться и перестроиться под новые закономерности. Более подробно такие подходы описываются в [34].

Для задач оффлайн-обучения, в которых все имеющиеся данные уже получены и требуют классификации, лучше показывают себя алгоритмы Питтсбургского типа. Однако вернемся к двухступенчатой схеме построения генетической нечеткой системы.

Один из примеров такой системы был представлен в работе [70]. В ней на первом этапе извлечение правил происходит без использования эволюционных алгоритмов. Для выбора наиболее перспективных правил используются меры *confidence* и *support*. Из-за того, что число всевозможных правил слишком велико, используются только относительно простые правила с максимальной длиной равной 3. В данной работе также было показаны влияние различных методов задания весовых коэффициентов нечетких правил на качество классификации.

В качестве тестовой задачи была выбрана задача классификации вин с 3-мя классами. После извлечения правил, они сортировались по весу, бралось по равному числу правил для каждого класса, чтобы построить классификатор. По результатам работы было показано, что для данной задачи использование в качестве веса произведения $support \cdot confidence$ давало лучший результат, в то время как классификатор без весов едва отличался по качеству классификации от случайного угадывания.

Второй этап построения базы правил использовал многокритериальный ГА, критериями которого выступали число неверно классифицированных объектов, число правил в базе и общая длина всех правил. Каждый бит в хромосоме ГА обозначал включение или исключение правила из базы. Многокритериальный подход позволил получить точные классификаторы с 5-7 правилами.

Схожий метод построения классификатора был использован в работе [9]. В ней Мичиганский этап представлял собой выбор правил с посредством специализированной процедуры инициализации с использованием данных из выборки. Полученные на первом этапе классификаторы, содержащие сотни правил, были способны решать задачу с определенной, хотя и относительно низкой точностью. Использование процедуры выбора правил, то есть второго этапа, позволило формировать достаточно точные и компактные классификаторы.

Таким образом, можно заключить, что подобный подход к формированию нечетких баз правил может быть достаточно эффективен. Однако стоит отметить несколько его недостатков. Во-первых, извлечение правил на первом этапе подразумевает их оценивание лишь по значениям *support* и *confidence*. Данные меры покрытия выборки действительно позволяют найти наилучшие правила, покрывающие наибольшую часть выборки и позволяющие верно классифицировать объекты. Однако итоговое качество классификации зависит от имеющихся правил в базе. Наиболее эффективная и точно решающая задачу база правил в свою очередь может содержать правила, покрывающие малую часть закономерности, без которых, однако, высокое качество классификации невозможно. Если подобные правила не попадут в итоговую популяцию Мичиганского этапа, то они не смогут быть сгенерированы на Питтсбургском этапе, так как на нем производится лишь выбор правил. В результате, для нахождения достаточно точного классификатора этим методом может потребоваться несколько запусков алгоритма. К преимуществам же такого подхода следует отнести относительную простоту и малое время работы при грамотной программной реализации.

Другой метод комбинирования Мичиганскго и Питтсбургского подхода подразумевает использование первого в качестве оператора мутации. То есть, в алгоритме Питтсбургского типа каждый индивид представляет собой базу правил, при этом оператор мутации изменяется. В операторе мутации теперь база правил рассматривается как популяция, где каждый индивид – это правило. Данная база правил модифицируется добавлением или удалением правил из нее. Один из вариантов подобной гибридизации был описан в [62]. Как было показано в данной работе, подобный подход показывает большую эффективность, чем Питтсбургский и Мичиганский подходы по отдельности. Мичиганский подход в данном алгоритме ищет хорошие правила, в то время как Питтсбургский подход ищет хорошие комбинации правил, таким образом, возможности обоих методов используются одновременно, в одном алгоритме. Такой подход лишен описанного выше недостатка, так как новые правила могут быть успешно сгенерированы в процессе работы алгоритма Мичиганским подходом.

3.3 Инициализация правил и грануляция нечетких множеств

Грануляцией называется разбиение пространства входных переменных на нечеткие множества. Как правило, для каждой из входных вещественных переменных задается по несколько нечетких термов, причем число термов фиксируется в начале работы алгоритма и не меняется при его работе.

Стоит пояснить, от чего зависит и на что влияет число нечетких термов. При отсутствии каких-либо априорных данных о характере имеющихся в данных закономерностей, о расположении кластеров точек в пространстве переменных нечеткие термы располагают равномерно на области значений переменной. Сама область значений определяется по имеющейся выборке или из иных экспертных данных при их наличии. При задании малого числа термов (2 или 3) для лингвистической переменной существует риск, что терм может неудачно покрывать часть данных и не отражать истинное распределение точек в пространстве. При задании большого числа термов (порядка 7 или более), система получает большую гибкость, так как теперь есть возможность оперировать

меньшими областями пространства в правилах. Однако сильная грануляция по переменным приводит к усложнению интерпретации нечеткой базы правил, хотя и позволяет достичь большей точности. Иногда можно выбрать достаточно удачное число термов для каждой из переменных для конкретной задачи, однако такая подстройка под задачу требует многократных запусков алгоритмов машинного обучения и больших временных затрат.

Частично проблему подстройки под данные решает введение весов для правил, как было описано в первом пункте данной главы. Изменение форм нечетких множеств [43] значительно снижает возможность интерпретации полученных баз правил, и потому не рассматривается. В качестве альтернативы настройке термов в некоторых работах, например [24], было предложено представление термов, кодирующее также смещение терма от его начального положения. Такой подход позволяет повышать точность нечетких баз правил, практически не влияя на их интерпретабельность.

Ещё один подход заключается в использовании нескольких грануляций пространства входных переменных одновременно. Такой подход многократно применялся в работах [70, 62, 66]. При использовании нескольких разбиений на нечеткие множества в правило может быть выбрано любое из имеющихся множеств, относящихся к разным грануляциям. То есть, в одном правиле для различных переменных могут присутствовать термы из различных разбиений. В работах Ишибучи используется 4 разбиения – на 2, 3, 4 и 5 нечетких множеств, плюс используется терм игнорирования, итого 15 возможных вариантов.

Правило в данном случае может быть представлено как строка целых чисел от 1 до 15. Такой подход вкупе с использованием эвристически определяемых весов правил дает высокую гибкость классификатору, при этом не меняя положения нечетких множеств и не требуя настройки. Следует, однако, отметить, что пространство поиска в этом случае возрастает.

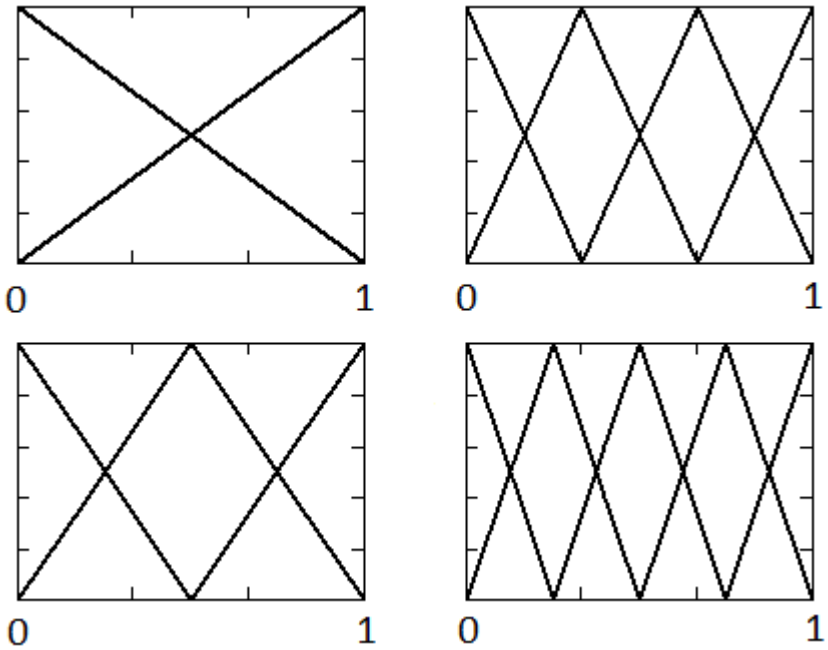


Рисунок 3.4. Разбиения на нечеткие множества

Схожий метод с использованием нескольких разбиений можно найти в работах [49, 50].

Как только заданы и определены нечеткие множества для всех переменных, следующим этапом во всех методах, использующих генетические алгоритмы, является инициализация популяции. Причем этот этап по своей сути мало отличается как для Мичиганского, так и для Питтсбургского подходов.

В случае если число переменных в решаемой задаче достаточно велико, случайная генерация правил, подобно случайной генерации начальных точек в ГА для вещественной оптимизации, становится неприемлемой. Особенно сильно это проявляется в случае, если число возможных нечетких множеств достаточно велико, например, равно 15, как описано выше. Число возможных различных правил при 15 нечетких множествах и F переменных равно 15^f. При 10 переменных это число равно порядка $5.7 \cdot 10^{11}$ возможных правил. Естественно, что генерация даже десятков тысяч случайных правил в таком случае не гарантирует, что хотя бы одно из них покроет хотя бы один объект обучающей выборки.

Решением этой проблемы является генерация правил на основании тех значений, которые имеются в выборке. Во многом эта процедура схожа с генерацией правил, предложенной в одной из первых работ ещё в начале 90-х годов [117]. Основная идея заключается в том, чтобы, взяв одно измерение случайным образом, построить нечеткое правило, в наилучшей степени отвечающее этому измерению. То есть, для каждой входной переменной берется значение из выборки, соответствующее заранее выбранному случайно измерению, вычисляется степень принадлежности ко всем нечетким множествам, и далее в правило ставится то нечеткое множество, которому соответствует максимальная степень принадлежности. Такой метод генерации правил гарантирует, что полученное правило будет покрывать, по крайней мере, одно измерение из выборки. В некоторых работах, например, [9], также часть нечетких множеств заменяется на терм игнорирования ради упрощения. В работе [66] используется слегка усложненная схема, в которой нечеткие множества выбираются случайно на основании полученных степеней принадлежности по процедуре, схожей с пропорциональной селекцией, где в качестве значений пригодности выступают степени принадлежности.

3.4 Самонастраивающийся эволюционный алгоритм построения нечетких баз правил

В данном пункте будет описан гибридный эволюционный алгоритм, разработанный для построения баз нечетких правил, с использованием описанных в данной главе методов и подходов. Итоговый алгоритм схож по общей схеме с генетическим алгоритмом и алгоритмом генетического программирования, однако отличается от него методом кодирования индивидов и потому в дальнейшем будет называться эволюционным алгоритмом.

Схема алгоритма и используемые операторы основаны на представленных в работе [66]. Опишем все используемые процедуры и параметры алгоритма подробно.

В правилах используется 4 разбиения на нечеткие множества (по 2, 3, 4 и 5 термов), плюс возможность игнорировать значение переменной. В общей сложности получается 15 различных нечетких множеств, при этом терм A_0 обозначает игнорирование значение переменной.

Таким образом, правило представляет собой строку целых чисел от 1 до 15. Включение терма игнорирования необходимо для уменьшения средней длины правил и повышения их обобщающей способности. Класс C_q и вес правила CF_q вычисляются по обучающей выборке. Степень принадлежности конкретного измерения x_p к правилу R_q вычисляется посредством оператора умножения следующим образом:

$$\mu_{Aq}(x_p) = \mu_{Aq1}(x_{p1}) \times \mu_{Aq2}(x_{p2}) \times \ldots \times \mu_{Aqn}(x_{pn}),$$

где $\mu_{Aqi}(x_{pi})$ – это значение функции принадлежности нечеткого множества A_{qi} при входе x_{pi}. Для того, чтобы определить соответствующий номер класса и вес правила, рассчитаем достоверность (*confidence*) правила:

$$Conf(A_q \to Class\ k) = \frac{\sum_{x_p \in Class\ k} \mu_{Aq}(x_p)}{\sum_{p=1}^{m} \mu_{Aq}(x_p)}.$$

Если значение $Conf > 0.5$, то генерируется нечеткое правило с вектором A_q и классом C_q. При этом C_q является классом с максимальным значением $Conf$. Вес правила CF_q определяется как в последнем случае в пункте 3.1, следующим образом:

$$CF_q = Conf(A_q \to Class\ k) - \sum_{k=1, k \neq C_q}^{M} Conf(A_q \to Class\ k).$$

Если значения *confidence* меньше или равны 0.5 для всех классов, то вес правила оказывается отрицательным, соответствующее нечеткое правило считается неперспективным и не формируется.

Классификация по базе правил RS производится с использованием одного правила-победителя. То есть, для каждого правила определяется произведение веса правила на степень принадлежности

$$\mu_{Aq}(x_p) \cdot CF_q$$

и далее выбирается правило с наибольшим произведением. В случае если имеется несколько правил с одинаковым значением произведения, но разными номерами классов, данный объект помечается как неверно классифицированный.

Общая схема алгоритма имеет вид:

1) Инициализация популяции с использованием выборки.
2) Оценивание индивидов-баз правил.
3) Генерация потомков с использованием селекции, скрещивания и мутации.
4) Применение Мичиганской части к каждому индивиду.
5) Формирование нового поколения.
6) Если условие остановки не удовлетворено, переход к шагу 2.

Мичиганская часть состоит из следующих этапов:

1) Пусть каждое правило в текущей базе – отдельный индивид.
2) Классифицировать обучающую выборку при помощи базы правил и назначить пригодность.
3) Сгенерировать или удалить несколько правил из популяции.
4) Вернуть полученную популяцию в Питтсбургскую часть.

В данном алгоритме число правил в базе заранее не определено, однако определена верхняя граница – максимальное число правил. Индивид представляет собой матрицу $n \cdot N$, где N – максимальное число правил, n – число переменных. Каждая строка в матрице – это отдельное правило, состоящее из целых чисел, кодирующих соответствующие номера нечетких множеств. В программе индивид кодируется именно в форме матрицы, а не в форме строки. Номер класса и вес правила хранятся отдельно. Если в процессе работы получается правило, чье значение *confidence* оказывается меньше 0.5, оно считается пустым – ему приписывается пустой класс и нулевой вес. Таким образом, в базе может быть множество пустых правил, и число правил определяется в ходе работы алгоритма. Также для каждого правила хранятся вычисленные значения $\mu_{Aq}(x_p)$, чтобы не

пересчитывать их заново при каждом новом подсчете пригодности, это позволяет ускорить работу алгоритма.

Инициализация начальной популяции происходит по нескольким объектам обучающей выборки. Для каждой базы правил генерируются $N/2$ нечетких правил из $N/2$ объектов выборки. При этом некоторые правила могут оказаться пустыми. Чтобы сформировать правило, необходимо определить A_q. Для этого рассчитывается степень принадлежности каждого значения по каждой переменной к каждому из 14 нечетких множеств. Далее, соответствующее значение номера нечеткого множества определяется из набора нечетких множеств, к которым степени принадлежности ненулевые. Этот механизм схож с процедурой пропорциональной селекции в классическом генетическом алгоритме. Вероятность того, что будет установлено B_j-е нечеткое множество:

$$P(B_j) = \frac{\mu_{B_j}(x_{pi})}{\sum_{k=1}^{14} \mu_{B_j}(x_{pi})}.$$

После этого каждая левая часть правила заменяется на терм игнорирования значения переменной с вероятностью 0.5. Эта процедура позволяет получить более простые базы правил, причем как минимум один объект из обучающей выборки будет покрыт правилом. В случае, если после генерации правил для одного индивида все они оказываются пустыми, база правил генерируется заново. Значение вероятности замены терма на терм игнорирования может варьироваться в относительно больших пределах (вплоть до 0.95), позволяя строить правила разной длины.

Оценка пригодности происходит как свертка трех основных критериев – ошибки на обучающей выборке $f_1(i)$, числа правил $f_2(i)$ и суммарной длинны всех правил $f_3(i)$. Ошибка на обучающей выборке берется в процентах и с весовым коэффициентом, равным $w_1 = 100$. Два других критерия берутся с весовыми коэффициентами, равными единице, т.е. $w_2 = 1, w_3 = 1$. Ошибка берется в процентах, чтобы исключить влияние объема выборки.

$$Fitness(i) = f_1(i) * w_1 + f_2(i) * w_2 + f_3(i) * w_3.$$

Стоит отметить, что в последнее время все чаще используются многокритериальные алгоритмы при формировании нечетких баз правил, такие как SPEA2 и NSGA-II. Обзор современных тенденций в области применения многокритериальных генетических нечетких систем можно найти в [47].

В алгоритме используется 3 типа селекции – пропорциональная, ранговая и турнирная с размером турнира 2. Все три типа селекции не отличаются по своей сути и реализации от селекции в стандартном ГА.

Скрещивание предполагает перемешивание генетической информации двух родителей – баз правил для формирования потомков. В данном случае скрещивание подразумевает под собой создание новой базы правил, включающей случайно выбранные правила из обоих родителей. При этом число правил определяется заранее, как случайное число в диапазоне $[1, |RS_1| + |RS_2|]$. При этом если суммарное число правил у родителей больше N, то случайное число генерируется в диапазоне $[1, N]$. Оператор скрещивания применяется с вероятностью 0.9. Правила для потомка выбираются из общего пула правил обоих родителей с равными вероятностями. При этом одно и то же правило может попасть в базу не более одного раза.

Мутация предполагает случайное изменение одного или нескольких генов в индивиде. В данном алгоритме используется три типа мутации, отличающиеся вероятностями – слабая, средняя и сильная. Мутация изменяет номер нечеткого множества в базе правил на другой, при этом могут изменяться любые части непустых правил, в том числе терм игнорирования значения переменной может быть заменен на какой-либо другой. Вероятность мутации, однако, в отличие от ГА, зависит от текущего числа правил в базе $|RS|$:

1) Слабая: $p_{mut} = 1/(3 \cdot n \cdot |RS|)$
2) Средняя: $p_{mut} = 1/(n \cdot |RS|)$
3) Сильная: $p_{mut} = 3/(n \cdot |RS|)$

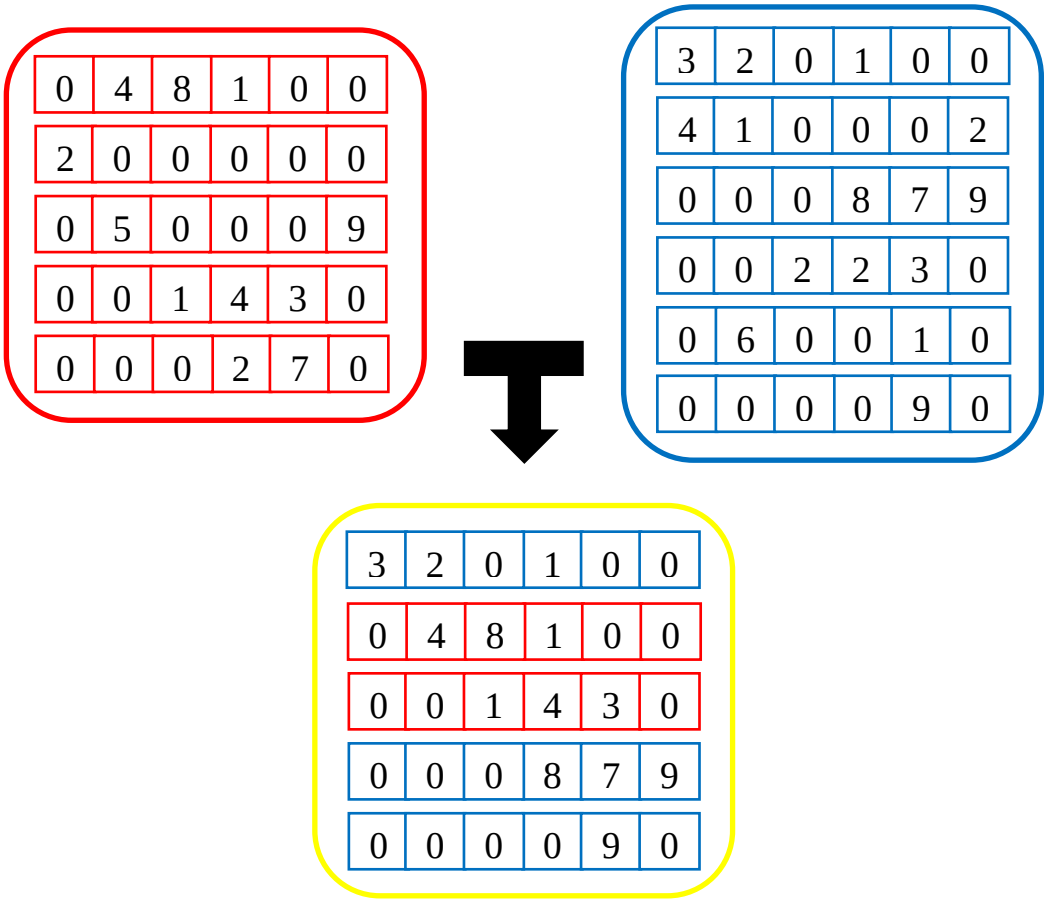


Рисунок 3.5. Процедура скрещивания

Мичиганская часть используется наряду с оператором мутации, и включает два этапа: удаление правил и добавление правил в базу. Добавление правил, так же как и удаление может быть применено с вероятностью 0.5. Таким образом, могут быть применены оба этапа, один из них или ни один из них.

Добавление и удаление правил основывается на пригодности правил, присутствующих в базе. Их пригодность оценивается на основании того, сколько объектов из обучающей выборки они классифицируют верно. То есть, если измерение x_p верно классифицируется правилом-победителем R_w, то пригодность правила увеличивается на единицу. При этом пересечений между областями, которые классифицируются различными правилами, нет, так как всегда есть только одно правило-победитель. Если же два одинаковых правила классифицируют один и тот же объект выборки с одинаковыми значениями

$\mu_{Aq}(x_p) \cdot CF_q$, то пригодность повышается только для первого из них. В этом случае правило-дубликат по окончании подсчета получает нулевую пригодность.

После назначения пригодностей может последовать добавление или удаление. В случае добавления есть две процедуры: добавление генетическим методом или эвристическим методом. В случае генетического метода применяются операторы селекции, скрещивания и мутации для формирования нового правила. Используется турнирная селекция с размером турнира 2, равномерное скрещивание с вероятностью 0.9 и средняя мутация – с вероятностью $1/n$. Данная комбинация настроек была выбрана, так как она достаточно хорошо показывает для классического генетического алгоритма.

В случае эвристического подхода после классификации и оценки пригодности каждого правила помечаются неверно классифицированные объекты. Данные объекты используются для формирования новых правил тем же методом, что использовался на этапе инициализации. То есть, для неверно классифицированного объекта формируется правило, покрывающее его и улучшающее качество классификации.

Число сгенерированных правил k определяется как $5(k-1) < |RS| \leq 5k$, эти правила добавляются в базу к имеющимся в ней. Если же число правил в базе достигает максимального, то правила не генерируются. Если же все объекты обучающей выборки классифицированны верно, то используется только генетический метод генерации новых правил. В обычном же случае вероятности применения генетического и эвристического подходов равны. Таким образом, совершается одно поколение в мичиганской части.

Удаление правил производится на основании той же оценки пригодности. Удаляется k правил, при $5(k-1) < |RS| \leq 5k$. Для удаления правила сортируются по убыванию пригодности и для последних k правил все нечеткие множества меняются на терм игнорирования, то есть правило исключается. Полученная в итоге база правил используется как индивид далее в Питтсбургской части.

Новое поколение формируется как лучшие индивиды из потомков и родителей, для этого все индивиды сортируются по пригодности, и выбирается N_p лучших.

В алгоритм была добавлена процедура самоконфигурации, основанная на оценивании успешности операторов по значениям пригодности, которая для операторов мутации и селекции аналогична использованной для ГА. Помимо селекции и мутации самоконфигурации была применена к Мичиганской части, где настраивался тип Мичиганского этапа. Всего использовалось 3 типа:

1) Только добавление правил в базу
2) Только удаление правил из базы
3) Добавление и удаление правил (замещение)

Как и раньше, при инициализации каждому из этих типов приписывались равные вероятности применения. Помимо этого, процедура самоконфигурации была добавлена и для выбора способа добавления правил в базу. То есть, имелось два способа добавления правил в базу: генетический и эвристический. Процедура оценивания успешности оператора в данном случае меняется, так как число применений эвристического и генетического подходов зависит от сгенерированных случайных чисел. Поэтому то, какой оператор считать примененным определялось по числу применений генетического и эвристического подходов. Если эвристический подход был применен больше раз чем генетический, то текущая пригодность потомка прибавлялась к сумме пригодностей для эвристического подхода. В противном случае – для генетического подхода. Если оба подхода были применены равное число раз, пригодность добавлялась к суммам обоих методов.

В следующем пункте данной главы будут рассмотрены методики тестирования представленного алгоритма в различных условиях, в том числе с самоконфигурацией и без, с различным объемом ресурсов, на различных задачах, а также сравнение с другими методами и подходами по эффективности классификации.

3.5 Программная реализация, тестирование алгоритма и результаты

Самонастраивающийся гибридный эволюционный алгоритм построения нечетких баз правил для задачи классификации комбинирующий Мичиганский и Питтсбургский подходы был реализован на языке C++ без использования средств объектно-ориентированного программирования. Программная система представляет собой однопоточное консольное приложение, без графического интерфейса.

В алгоритме была реализована процедура кросс-валидации [75] с произвольным числом частей. Обучающая выборка нормировалась на интервал [0,1], на котором задавались функции принадлежности термов. Таким образом, для каждого измерения обучающей выборки один раз рассчитывались степени принадлежности ко всем нечетким термам, и они использовались в ходе обучения. То есть, благодаря нормировке удалось избежать многократного пересчета степеней принадлежности. При этом нормировка производилась только по обучающей выборке, таким образом, в тестовой выборке значения после нормировки могли выходить за границы [0,1]. Это необходимо так как при решении реальных задач в некоторых ситуациях могут быть неизвестны минимальные и максимальные значения переменных, и в этой ситуации база правил должна быть способна корректно функционировать.

Также ради оптимизации времени работы для каждого правила в базе рассчитывались и сохранялись номер класса, вес и степени принадлежности по каждому из измерений. Эти значения сохранялись, в том числе и при проведении процедуры скрещивания, и при переходе в новое поколение, и пересчитывались только тогда, когда изменялось само правило.

Тестирование алгоритма производилось на наборе задач классификации, взятых с репозиториев UCI [27] и KEEL [26]. Качество классификации измерялось как доля верно классифицированных объектов на тестовой и обучающей выборке. Процедура 10-частной кросс-валидации повторялась 3 раза, значения точности усреднялись.

Первая серия экспериментов посвящена сравнению самоконфигурирующегося и стандартного алгоритма. Тестирование проводилось на шести задачах, среди них:

1) Задача об австралийских кредитных картах, 690 измерений, 14 переменных, 2 класса – Australian,

2) Задача классификации клиентов банка, 1000 измерений, 24 признака, 2 класса – German,

3) Задача классификации сегментов изображения, 2310 измерений, 19 признаков, 7 классов – Segment.

4) Задача классификации областей распознанного текста, 5472 измерения, 10 признаков, 5 классов – Pageblocks;

5) Задача классификации назальных и оральных звуков, 5404 измерения, 5 признаков, 2 класса, – Phoneme;

6) Задача классификации пикселей на снимках со спутника, 6435 измерений, 36 переменных, 6 классов, – Satimage;

Тестирование производилось на ноутбуке с процессором Intel Core i5-2410M (2 ядра, 2.3ГГц) и 4 Гб оперативной памяти. Для первой серии экспериментов число индивидов устанавливалось равным 100, число поколений – 500, максимальное число правил – 40. В таблице 3.1 представлены протестированные конфигурации алгоритма.

Таблица 3.1 Расшифровка номеров конфигураций алгоритма

№ конфигурации	Тип селекции	Тип мутации	Мичиганская часть	Добавление правил
1	Пропорц.	Слабая	Добавление с вероятностью 0.5; Удаление с вероятностью 0.5	Эвристический и генетический подходы равное число раз
2		Средняя		
3		Сильная		
4	Ранговая	Слабая		
5		Средняя		
6		Сильная		
7	Турнирная (2)	Слабая		
8		Средняя		
9		Сильная		

В таблице 3.2 представлено сравнение точности для стандартного и самоконфигурирующегося алгоритмов на обучающей выборке. Также приведены результаты для самоконфигурирующегося алгоритма без пропорциональной селекции.

Таблица 3.2 Доля верно классифицированных на обучающей выборке

Алгоритм	Australian	German	Segment	Phoneme	Pageblocks	Satimage
1	0.875	0.730	0.840	0.797	0.939	0.798
2	0.873	0.709	0.827	0.803	0.944	0.791
3	0.874	0.722	0.833	0.799	0.939	0.794
4	0.909	0.793	0.913	0.828	0.955	0.850
5	0.913	0.807	0.919	**0.833**	**0.958**	0.850
6	**0.917**	**0.821**	**0.927**	**0.832**	**0.959**	**0.852**
7	0.909	0.788	0.911	0.828	0.954	0.845
8	0.910	0.789	0.914	0.829	0.956	0.845
9	0.908	0.801	0.916	**0.832**	0.957	0.844
Средний стандартный	0.899	0.773	0.889	0.820	0.951	0.830
Средний без пропорцион. селкции	0.911	0.800	0.917	0.830	0.957	0.847
Самоконфи-гурируемый	0.911	0.802	0.913	0.825	0.957	0.847
Самоконфи-гурируемый без пропор-циональной	0.914	0.803	0.918	0.830	**0.959**	0.850

Результаты для тестовой выборки представлены в таблице 3.3. Как можно видеть из приведенных таблиц, в среднем алгоритм с самонастройкой оказывается лучше, чем стандартный алгоритм, но хуже чем лучшая стандартная конфигурация для данной задачи. Отдельно стоит отметить хороший результат, получаемый при комбинировании ранговой селекции с сильной мутацией. Данная конфигурация (№ 6) выделяется не только на обучающей, но и на тестовой выборке, причем на всех задачах, из чего можно сделать вывод, что её можно рекомендовать к применению в данном алгоритме. Стоит также отметить, что пропорциональная селекция показывает значительно худшие результаты, чем два других типа селекции. Средняя точность без учета пропорциональной селекции

равна или даже больше, чем с использованием самонастройки. Самонастраивающийся алгоритм, в котором не используется пропорциональная селекция, показывает результаты, сравнимые или превосходящие полученные стандартным алгоритмом.

Таблица 3.3 Доля верно классифицированных на тестовой выборке

Алгоритм	Australian	German	Segment	Phoneme	Pageblocks	Satimage
1	0.839	0.701	0.791	0.784	0.920	0.790
2	0.839	0.707	0.790	0.790	0.931	0.785
3	0.841	0.710	0.767	0.789	0.932	0.782
4	**0.872**	0.759	0.880	0.805	0.942	0.831
5	0.869	0.748	0.888	0.807	**0.950**	**0.840**
6	0.861	**0.768**	**0.893**	**0.810**	**0.947**	0.835
7	0.857	0.743	0.878	0.806	0.939	0.827
8	0.860	0.746	0.885	**0.810**	0.945	0.836
9	**0.874**	0.743	0.883	0.806	0.950	0.830
Средний Стандартный	0.857	0.736	0.851	0.801	0.934	0.817
Средний без пропорцион. селекции	0.865	0.751	0.884	0.807	0.947	0.833
Самоконфи-гурируемый	0.857	0.749	0.876	0.806	0.945	0.833
Самоконфи-гурируемый Без пропор-циональной	**0.871**	0.751	0.881	**0.812**	**0.950**	0.835

Для сравнения с другими методами были выбраны метод опорных векторов (SVM) [5, 6] и нейронные сети (NN), так как эти методы хорошо известны и часто применяются в задачах классификации. Были использованы реализации обоих методов в программной системе Statistica 10 [4]. Для корректности сравнения была также использована процедура 10-частной кросс-валидации. Для определения структуры нейронных сетей были использованы встроенные инструменты Statistica. В таблице 3.4 представлены результаты сравнения эффективности самоконфигурируемого гибридного эволюционного алгоритма (обозначен FHEA, взята версия без использования пропорцинальной селекции) с

двумя другими методами. Приведены результаты без пропорциональной селекции.

Таблица 3.4 Сравнение подхода с другими методами

Алгоритм	FHEA		SVM		Neural Network	
Задача	Обуч.	Тестов.	Обуч.	Тестов.	Обуч.	Тестов.
Australian	0.914	**0.871**	0.858	0.826	**0.923**	0.852
German	0.803	0.751	0.786	**0.770**	**0.863**	0.758
Segment	0.918	0.881	**0.932**	**0.926**	0.909	0.898
Pageblocks	0.959	**0.950**	0.944	0.923	**0.966**	**0.951**
Phoneme	**0.830**	**0.812**	0.768	0.761	0.815	0.794
Satimage	0.850	0.835	**0.876**	**0.870**	0.843	0.819

Как можно видеть из таблицы, предлагаемый алгоритм незначительно уступает по качеству классификации другим известным методам. К примеру, на задаче phoneme он показал себя лучше, чем нейронные сети и метод опорных векторов. При этом алгоритму достаточно малого объема ресурсов для получения хороших результатов. При увеличении объема ресурсов до 200 индивидов и 5000 поколений на первых двух задачах алгоритм показывает значительно лучшие результаты.

Таблица 3.5 Сравнение подхода с другими методами при увеличенном ресурсе

Алгоритм	FHEA		SVM		Neural Network	
Задача	Обуч.	Тестов.	Обуч.	Тестов.	Обуч.	Тестов.
Australian	**0.938**	**0.876**	0.858	0.826	0.923	0.852
German	**0.860**	**0.768**	0.786	**0.770**	**0.863**	0.758

При увеличении ресурса алгоритма в 20 раз повысилась точность как на тестовой, так и на обучающей выборке, и алгоритм смог сравниться по эффективности с другими методами или даже превзойти их.

Также во время работы алгоритма замерялось число правил в базе, средняя длина правила и время работы алгоритма. В таблице 3.6 приведены средние значения для всех задач. Число правил подбирается алгоритмом в ходе работы и отличается для каждой решаемой задачи. Приведены результаты с использованием пропорциональной селекции.

Таблица 3.6 Параметры баз правил и время счета

Задача	Обуч.	Тестов.	Число правил	Длина правила	Время работы
Australian	0.911	0.857	18.5	4.2	25.4c
German	0.802	0.749	20.4	5.8	48.9c
Segment	0.913	0.876	15.6	7.7	556c
Pageblocks	0.957	0.945	7.80	4.2	309c
Phoneme	0.825	0.806	11.5	2.7	313c
Satimage	0.850	0.835	15.6	13.7	822c

Стоит также отметить, что алгоритм обладает относительно высокой скоростью работы, хотя и не может сравниться по этому показателю со специализированными пакетами. Например, на задаче Australian время работы составляет порядка 25 секунд, хотя для задач с большим числом измерений и переменных время растет. Результаты опубликованы в [15, 14, 105, 108, 13].

Таблица 3.7 Сравнение с другими методами построения баз правил, обучающая выборка

Выборка	FUZZY_GA OCD	FUZZY_GA NCD	FUZZY_GA ADJ	FHEA
Australian	0.903	**0.914**	0.909	0.912
Banknote	0.952	0.901	**0.999**	**0.999**
Breast cancer	0.979	0.982	**0.989**	**0.987**
Column 2c	0.802	0.802	**0.931**	0.899
Column 3c	0.719	0.743	**0.929**	0.885
German credit	**0.822**	0.794	0.806	0.802
Glass	0.678	0.654	0.776	**0.841**
Heart	0.915	0.937	0.929	**0.948**
Ionosphere	0.923	0.845	0.864	**0.971**
Iris	0.968	0.821	**0.998**	0.989
Liver	0.671	0.669	0.854	0.803
Pima	0.787	0.780	**0.853**	0.826
Seeds	0.941	0.838	**0.992**	0.983
Wine	0.986	0.971	0.995	**1.000**

Также был проведен ряд тестов для сравнения с алгоритмами из главы 2. В качестве тестовых задач были взяты все задачи, представленные в таблице 2.1. В таблицах 3.7 и 3.8 представлены результаты сравнения на обучающей и тестовой

выборке соответственно. Приведены результаты с использованием пропорциональной селекции.

В большинстве случаев гибридный алгоритм значительно превосходит остальные методы, при этом объем требуемых ресурсов для него значительно ниже. Более того, его точность на тестовой выборке выше практически для всех задач. Схожую эффективность на обучающей выборке показывает только алгоритм, настраивающий расположение сигмоидальных функций принадлежности. Для задачи Wine на обучающей выборке гибридный алгоритм всегда находил решения, точно классифицирующие все объекты, однако на тестовой выборке точность составляла порядка 0.91.

Таблица 3.8 Сравнение с другими методами построения баз правил, тестовая выборка

Выборка	FUZZY_GA OCD	FUZZY_GA NCD	FUZZY_GA ADJ	FHEA
Australian	0.840	0.836	0.852	**0.857**
Banknote	0.942	0.893	**0.995**	0.967
Breast cancer	0.949	0.953	0.956	**0.965**
Column 2c	0.777	0.770	0.807	**0.815**
Column 3c	0.642	0.706	**0.803**	0.779
German credit	0.707	0.707	0.709	**0.749**
Glass	0.536	0.491	0.600	**0.729**
Heart	0.771	0.764	0.759	**0.854**
Ionosphere	0.799	0.695	0.767	**0.879**
Iris	0.936	0.761	**0.943**	0.902
Liver	0.568	0.573	0.629	**0.696**
Pima	0.732	0.694	0.732	**0.778**
Seeds	0.883	0.777	**0.917**	0.909
Wine	0.878	0.836	0.851	**0.913**

Первым трем алгоритмам было выделено $700 \cdot 700 = 490000$ вычислений функции пригодности, в то время как гибридному методу - $100 \cdot 500 = 50000$, то есть практически в 10 раз меньше. За счет оптимизации и сохранения степеней принадлежности время счета при этом было снижено более чем в 10 раз. Таким образом, можно заключить, что полученный самонастраивающийся алгоритм является эффективным как в смысле качества построенных классификаторов, так и по времени работы.

Для демонстрации работы самоконфигурации приведем несколько графиков изменения вероятностей применения операторов при работе алгоритма на задаче Australian.

Рисунок 3.6 Самоконфигурация оператора селекции

Рисунок 3.7 Самоконфигурация оператора мутации

Рисунок 3.8 Самоконфигурация Мичиганской части

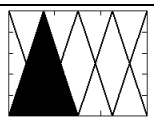
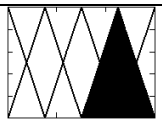
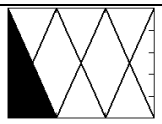
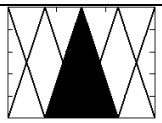
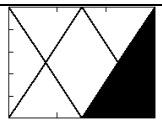
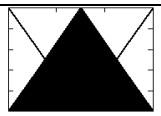
Рисунок 3.9 Самоконфигурация добавления правил

Тенденции по изменению вероятностей в значительной степени зависят от запуска алгоритма, и все же можно выделить несколько закономерностей, который наблюдались в процессе работы. В большинстве случаев слабая мутация получала всегда меньшую вероятность применения, чем два других метода,

которые были практически равноправны. Турнирная и ранговая селекция схожи по поведению, однако пропорциональная селекция зачастую получает значительно меньшие вероятности. Среди вероятностей в мичиганской части не удалось выделить сколько-нибудь заметных различий, все три метода получают примерно одинаковые вероятности, или же методы меняются местами от раза к разу. Среди способов добавления правил в базу чаще большую вероятность получает генетический метод, однако вероятность применения эвристики никогда не достигает минимума.

Приведем также одну из баз правил в качестве примера результата, выдаваемого алгоритмом для задачи Iris.

Таблица 3.9 Пример базы правил, полученной алгоритмом для задачи Iris

Вход	1	2	3	4		Класс *C*	Вес *CF*
Если	DC		DC		то	2	0.798
Если	DC	DC		DC	то	2	0.632
Если	DC	DC			то	1	0.801
Если	DC	DC	DC		то	0	0.888

Точность классификации по данной базе – 0.985 на обучающей и 1.000 на тестовой выборке при заданном разбиении, т.е. 2 неверно классифицированных объекта на обучающей выборке. Терм DC расшифровывается как “Don’t Care” и обозначает игнорирование. Средняя длина правила равна 1.5. Данную базу правил можно относительно легко интерпретировать, например, последнее правило говорит, что если ширина лепестка средняя, то это Iris Setosa (0 класс). Если ширина лепестка большая и длина средняя, то это Iris Versicolour. Если длина лепестка очень малая или ширина лепестка выше среднего и длина чашелистика

ниже среднего, то это Iris Virginica. Такая база правил дает практически стопроцентную точность и не только понятна, но и может быть легко запомнена человеком.

Естественно, для более сложных задач число правил увеличивается, и анализ несколько усложняется, тем не менее, его сложность несопоставима со сложностью анализа, например, нейросетевых моделей.

ВЫВОДЫ

В третьей главе был разработан современный и эффективный метод построения нечетких баз правил. Самоконфигурируемый гибридный эволюционный алгоритм, комбинирующий Мичиганский и Питтсбургский подходы, является специализированным методом для решения сложных задач классификации, позволяющим быстро получать надежные и легко воспринимаемые человеком базы правил. В нем используются современные методы самоконфигурации эволюционных алгоритмов, специальная процедура инициализации с использованием данных, имеющихся в выборке измерений, особая схема гибридизации Мичиганского и Питтсбургского подходов, применяемая наряду с классической мутацией в генетическом алгоритме, эвристические методы назначения весов правил и добавления правил в базу на Мичиганском этапе, а также различные методики оптимизации скорости работы, возможные благодаря жестко заданным термам.

Все эти особенности позволяют алгоритму успешно справляться с решением задач классификации, включающих десятки переменных, а также достаточно большой объем данных. Алгоритм является перспективным в том смысле, что не требует от пользователя специализированных знаний об эволюционных методах и влиянии размеров базы правил на качество классификации, так как вероятности применения операторов алгоритма самонастраиваются, а наилучшее число правил определяется автоматически в ходе работы алгоритма.

Среди путей улучшения данного подхода можно отметить дальнейшее исследование свойств алгоритма на различных задачах, применение иных эвристик, разработку новых операторов, таких как скрещивание, использование многокритериальных подходов вместо свертки критериев, а также модификацию алгоритма для возможности эффективного решения задач с несбалансированными данными [33].

В следующей главе будут рассмотрены модификации предложенного гибридного эволюционного алгоритма для решения задач с несбалансированными данными, а также предложен метод селекции обучающих примеров для повышения качества формируемых классификаторов и снижения требований к объемам вычислительных ресурсов.

Глава 4. Гибридный эволюционный алгоритм построения нечетких баз правил для задач с несбалансированными данными с отбором обучающих примеров

4.1 Несбалансированные данные в задачах классификации

Задачей машинного обучения является формирование системы, способной к предсказанию значения для ранее неизвестного примера с хорошей обобщающей способностью. В процессе обучения происходит извлечение знаний из выборки данных -мерного пространства R^F. При наличии номеров классов C_j в выборке, происходит обучение с учителем [1]. Таким образом, формируемая система представляет собой отображение $R^F \to C$.

При решении реальных задач классификации зачастую возникает ситуация, когда невозможно получить достаточное количество измерений по некоторым классам. Причиной может служить значительная стоимость получения новых измерений, или же невозможность получение измерений вследствие естественных причин. В этих условиях возникают задачи классификации с несбалансированными данными.

Набор данных называется несбалансированным в том случае, если число измерений, представляющих один класс меньше, чем число измерений другого класса. Более того, класс, имеющий меньшее число измерений, как правило, является классом, представляющим наибольший интерес с точки зрения обучения. Проблема несбалансированных данных чаще всего возникает в таких областях, как определение загрязнений [82], управление рисками [61], выявление случаев мошенничества [44], и в особенности в области медицинской диагностики [74, 40, 95]. Класс, имеющий наибольшее число измерений, называется мажоритарным классом, в противоположность ему, класс с наименьшим числом измерений называется миноритарным.

При наличии дисбаланса в распределении классов, стандартные методы обучения классификаторов смещаются в сторону более представленных классов, так как более полное описание данных классов сильнее вознаграждается в смысле

величины меры точности классификации. При этом менее представленные в выборке классы зачастую игнорируются алгоритмом обучения, так как рассматриваются им в качестве шума. Таким образом, измерения миноритарного класса чаще классифицируются неверно. Стоит отметить, что дисбаланс в распределении числа измерений не делает решение задачи классификации невозможным, однако из-за него зачастую возникает ряд сложностей.

Среди основных сложностей следует отметить следующие. Во-первых, в большинстве случаев несбалансированные наборы данных имеют малое число измерений миноритарного класса. В [52] авторы показали, что значение ошибки, полученное вследствие несбалансированности данных может быть уменьшено, если число измерений в миноритарном классе станет репрезентативным.

Ещё одной проблемой является значительное пересечение классов. В случае, измерений мажоритарного класса значительно больше, чем измерений миноритарного, последний может сильно пересекаться в мажоритарным классом. Это может происходить, к примеру, вследствие статистических ошибок при проведении измерений. В подобных ситуациях строятся более общие классификаторы, которые показывают высокую точность, при этом ошибаясь лишь на измерениях миноритарного класса. В случае отсутствия пересечения классов, для большинства алгоритмов обучения разделение классов не представляет значительной проблемы.

Также возможны ситуации, когда миноритарный класс формируется из ряда мелких подклассов, в основании возникновения которых лежат разные причины. При этом миноритарный класс представляет собой несколько групп точек, лежащих в разных областях признакового пространства. При наличии малого числа измерений, данные группы точек рассматриваются алгоритмом обучения в качестве шума, либо верно классифицируются наиболее представленные из них. Данная ситуация существенно усложняет задачу классификации.

Большинство исследователей, фокусируются на двух-классовых задачах с несбалансированными данными [107], мотивируя это тем, что любая задача, имеющая большее число классов может быть представлена как набор двух-

классовых задач. Вероятно, одной из причин также является то, что некоторые методы машинного обучения могут работать только с двумя классами. Однако, в случае работы с нечеткими системами классификации, которые способны описывать множество классов одной базой правил, нет необходимости в разделении изначальной задачи на подзадачи. Более того, один классификатор видится более предпочтительным, чем набор классификаторов, поэтому в данной работе далее будут рассматриваться задачи классификации с несбалансированными данными и числом классов больше двух.

Таблица 4.1. Матрица ошибок для задач с двумя классами

	Предсказан первый класс	Предсказан второй класс
Истинный первый класс	True Positive (TP)	False Negative (FN)
Истинный второй класс	False Positive (FP)	True Negative (TN)

Критерий оценки классификации является ключевым фактором в оценке качества классификации и направлении процесса обучения. Классическим методом оценки качества является мера точности классификации, которая может быть выражена как отношение числа неверно классифицированных объектов к их общему числу. В соответствии с матрицей ошибок для задач с двумя классами, представленной в таблице 4.1, точность может быть записана следующим образом:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Однако, в случае несбалансированных данных, точность классификации не может служить адекватной мерой качества классификации. К примеру, для задачи, в которой отношение числа измерений первого класса к числу измерений второго, обозначаемое мерой дисбаланса, равно 9, точность классификации, равная 90% может означать, что классификатор относит все измерения в первый класс. Таким образом, хотя точность классификации и остается высокой, на самом деле задача оказывается нерешенной.

Вследствие важности проблемы несбалансированных данных, был разработан ряд методов для её разрешения. Среди них можно выделить четыре группы:

1) Подходы на уровне алгоритма, или внутренние подходы. Целью данных методов является смещение процесса обучения в сторону миноритарного класса. Данные методы, как правило, требуют специальных знаний о природе классификатора и задаче [81, 79, 29].

2) Подходы на уровне данных, или внешние подходы. Данная группа методов призвана восстановить баланс между классами посредством изменения выборки данных [30, 112, 48, 88]. Таким образом, данные методы независимы от применяемого алгоритма классификации, что делает их более универсальными.

3) Подходы, чувствительные к стоимости располагаются между внешними и внутренними подходами. Они модифицируют выборку, добавляя веса измерениям, и модифицируют алгоритм, чтобы он корректно работал с весами [41, 80, 125]. При этом большие веса назначаются миноритарному классу, в предположении, что данная мера приведет к нахождению классификаторов, одинаково точных на обоих классах.

4) Подходы, использующие ансамбли. Данные методы могут использовать некоторые подходы из трех предыдущих, чтобы сформировать ансамбль классификаторов для несбалансированных наборов данных. Как правило, используются подходы на уровне данных, а также назначение весов измерениям.

Для оценки качества классификации используется ряд мер, отличных от точности и использующих значения TP, TN, FP, FN из таблицы 4.1. Для задач бинарной классификации эти меры представлены в таблице 4.2 [107].

Таблица 4.2. Меры качества классификации, 2 класса

Мера	Формула	Описание
$Accuracy$	$\frac{TP+TN}{TP+TN+FP+FN}$	Общая точность классификации
$Precision$	$\frac{TP}{TP+FP}$	Согласованность классификации первого класса с данными

Продолжение таблицы 4.2

$Recall\ (Sensitivity)$	$\frac{TP}{TP+FN}$	Эффективность классификатора по выделению первого класса
$Fscore$	$\frac{(\beta^2+1)\cdot TP}{(\beta^2+1)\cdot TP+\beta^2\cdot FN+FP}$	Отношение между объектами первого класса в данных и предсказанными классификатором
$Specificity$	$\frac{TN}{FP+TN}$	Эффективность классификатора по выделению второго класса
$AUC\ (Area\ Under\ Curve)$	$\frac{1}{2}(\frac{TP}{TP+FN}+\frac{TN}{TN+FP})$	Способность классификатора по избеганию неверной классификации

Для задач классификации с числом классов больше двух, представленные в таблице 4.2 меры модифицируются. В первом случае, происходит макро-усреднение при котором усредняются меры, рассчитанные по всем классам; при микро-усреднении рассчитываются суммарные значения TP, FN, TN, FP, и затем рассчитывается мера качества классификации. Макро-усреднение рассматривает все классы одинаково, в то время как микро-усреднение смещается в сторону мажоритарных классов. В таблице 4.3 представлены меры классификации в соответствии с [107] Здесь K – число классов, микро усреднение обозначено μ, макро-усреднение – M.

Таблица 4.3. Меры качества классификации, произвольное число классов

Мера	Формула	Описание
$Average\ Accuracy$	$\frac{\sum_{i=1}^{K}\frac{TP_i+TN_i}{TP_i+TN_i+FP_i+FN_i}}{K}$	Средняя эффективность классификации по всем классам
$Error\ Rate$	$\frac{\sum_{i=1}^{K}\frac{FP_i+FN_i}{TP_i+TN_i+FP_i+FN_i}}{K}$	Средняя ошибка классификации по всем классам
$Precision_{\mu}$	$\frac{\sum_{i=1}^{K}TP_i}{\sum_{i=1}^{K}(TP_i+FP_i)}$	Согласованность классификации по всем классам с данными
$Recall_{\mu}$	$\frac{\sum_{i=1}^{K}TP_i}{\sum_{i=1}^{K}(TP_i+FN_i)}$	Эффективность классификатора по выделению всех номеров классов
$Fscore_{\mu}$	$\frac{(\beta^2+1)Precision_{\mu}Recall_{\mu}}{\beta^2 Precision_{\mu}+Recall_{\mu}}$	Отношение между номерами классов в данных и полученными классификатором

Продолжение таблицы 4.3

$Precision_M$	$\frac{\sum_{i=1}^{K} \frac{TP_i}{TP_i + FP_i}}{K}$	Средняя согласованность между классификатором и данными
$Recall_M$	$\frac{\sum_{i=1}^{K} \frac{TP_i}{TP_i + FN_i}}{K}$	Средняя эффективность по выделению номеров классов
$Fscore_M$	$\frac{(\beta^2 + 1) Precision_M Recall_M}{\beta^2 Precision_M + Recall_M}$	Отношение между номерами классов в данных и полученными классификатором в результате усреднения

Мера AUC не приведена в таблице 4.3, так как до сих пор нет хорошо разработанного представления этой меры для многоклассовых наборов данных. Для двух классов мера AUC рассчитывается как площадь под кривой ROC-кривой [77].

Все рассмотренные меры рассчитываются по имеющейся матрице ошибок классификации.

4.2 Модификация гибридного эволюционного алгоритма нечеткой классификации для несбалансированных данных

В данном пункте будут рассмотрены модификации алгоритма нечеткой классификации, представленного в главе 3 для успешного решения задач с несбалансированными данными. Стоит отметить, что каждое нечеткое правило в базе в данном методе описывает только один класс, таким образом можно отследить какая часть базы правил описывает конкретный класс. К примеру, в случае, если база правил имеет существенное смещение в сторону мажоритарного класса, все правила в базе могут описывать только мажоритарный класс.

Как было описано в главе 3, в гибридном алгоритме нечеткой классификации используется ряд эвристик для назначения весов правил и определения наиболее подходящего номера класса. В частности, используется значение достоверности (confidence), рассчитываемое по выборке данных.

$$Conf(A_q \to Class\ k) = \frac{\sum_{x_p \in Class\ k} \mu_{Aq}(x_p)}{\sum_{p=1}^{m} \mu_{Aq}(x_p)}.$$

В случае работы с несбалансированными данными, значение confidence также смещается в сторону мажоритарного класса. Причиной этого смещения является то, что достоверность рассчитывается по числу измерений, принадлежащих тому или иному классу. В случае, если имеется существенное пересечение между мажоритарным и миноритарным классом в области, которую покрывает правило, мажоритарному классу отдастся предпочтение, вследствие того, что правило будет покрывать большее число измерений этого класса.

В то же время, в случае, если пересечение классов незначительное, и правило, для которого рассчитывается достоверность в значительной степени покрывает миноритарный класс, оно будет успешно сгенерировано и включено в базу правил.

Для того чтобы избежать отнесения правил, в большей степени отписывающих миноритарный класс, чем мажоритарный в случае наличия пересечений между классами, предлагается модифицировать процедуру определения наиболее подходящего номера класса в соответствии с числом измерений следующим образом:

$$Class\ q = \underset{k}{\operatorname{argmax}}\left(\frac{Conf(A_q \to Class\ k)}{n_k} \cdot n\right),$$

где n_k это число измерений в выборке, принадлежащих к -му классу, n – общее число измерений. Данная модификация смещает процедуру определения наилучшего номера класса в сторону миноритарного класса, так что правила, одинаково хорошо описывающие как миноритарный, так и мажоритарный класс, будут отнесены к миноритарному классу.

Для проверки эффективности предложенной модификации был произведен ряд вычислительных экспериментов на ряде задач с несбалансированными данными. При этом для оценки эффективности классификации использовались меры, $Accuracy$ и $Recall_M$. Мера $Recall_M$ была использована, так как показывает среднюю по всем классам точность классификации.

Тестирование было проведено на 5 задачах с репозиториев UCI Machine Learning [27] и KEEL [26]. Среди них:

1. Page-blocks, 5472 измерения, 10 переменных, 5 классов
2. German credit, 1000 измерений, 24 переменнык, 2 класса
3. Phoneme, 5404 измерения, 5 переменных, 2 класса
4. Segment, 2310 измерений, 19 переменных, 7 классов
5. Satimage, 6435 измерений, 36 переменных, 6 классов

Распределение числа измерений по классам представлено в таблице 4.4

Таблица 4.4. Распределение измерений по классам

Задача\Класс	1	2	3	4	5	6	7
Page-blocks	4913	329	28	87	115	-	-
German	700	300	-	-	-	-	-
Phoneme	3818	1586	-	-	-	-	-
Segment	330	330	330	330	330	330	330
Satimage	1533	703	1358	626	707	1508	-

Среди наборов данных, задача Segment имела сбалансированное число измерений по всем классам, и была включена в тестирования для демонстрации того, что модификация не влияет на качество классификации для сбалансированных задач.

Для получения адекватных результатов использовалась процедура 10-частной стратифицированной кросс-валидации, которая повторялась 3 раза. Таким образом, результаты усреднялись по 30 измерениям. Ресурсы алгоритма задавались следующим образом: число индивидов: 100, число поколений: 500, максимальное число правил: 40.

Тестировалось 4 варианта алгоритма: с мерой $Average\ Accuracy\ (Acc)$ в функции пригодности и с мерой $Recall_M$, с оригинальной процедурой определения наиболее подходящего номера класса и с модифицированной процедурой. Результаты тестирования на задаче Page-blocks представлены в таблице 4.5. Для каждой из конфигураций приведены величины мер *Acc* и *Recall* как на тестовой, так и на обучающей выборке.

Таблица 4.5. Результаты для задачи Page-blocks

Конфигурация	Обуч. Acc	Обуч. Recall	Тест. Acc	Тест. Recall
Acc+смещ.	**0.959**	0.600	**0.955**	0.584
Recall+смещ.	0.879	0.810	0.874	0.767
Acc+несмещ.	0.951	0.602	0.947	0.573
Recall+несмещ.	0.891	**0.855**	0.887	**0.822**

При использовании меры *Acc* как критерия качества классификации, точности как на обучающей, так и на тестовой выборках получаются высокими, в то время как значения *Recall* остаются низкими. Применение меры *Recall* значительно улучшает сбалансированность классификатора, но точность снижается. Добавление несмещенной процедуры назначения номеров классов не только увеличивает значения *Recall*, но также повышает точность.

Таблица 4.6. Результаты для задачи German

Конфигурация	Обуч. Acc	Обуч. Recall	Тест. Acc	Тест. Recall
Acc+смещ.	0.797	0.681	**0.726**	0.591
Recall+смещ.	0.794	0.759	0.715	0.633
Acc+несмещ.	**0.802**	0.694	0.721	0.596
Recall+несмещ.	**0.802**	**0.777**	0.719	**0.678**

Для задачи German были получены схожие результаты, однако в этом случае точности на обучающей выборке отличаются незначительно для всех конфигураций. Что касается значений *Recall*, то наилучшие результаты были получены для случая использования этой меры в функции пригодности и с модифицированной процедурой назначения номеров классов. Более того, данная конфигурация показала незначительное уменьшение точности классификации на тестовой выборке.

Таблица 4.7. Результаты для задачи Phoneme

Конфигурация	Обуч. Acc	Обуч. Recall	Тест. Acc	Тест. Recall
Acc+смещ.	**0.825**	0.781	**0.815**	0.768
Recall+смещ.	0.803	0.824	0.792	0.811
Acc+несмещ.	0.821	0.778	0.810	0.764
Recall+несмещ.	0.803	**0.826**	0.791	**0.812**

Для задачи Phoneme модификация процедуры назначения номеров классов не дает каких-либо преимуществ, что может говорить об отсутствии пересечения

классов. При этом использование меры *Recall* в функции пригодности заставляет классификатор отдавать предпочтение миноритарному классу. Это можно видеть по тому, как меняется разница между мерами *Acc* и *Recall*: для первой конфигурации точность значительно превосходит *Recall,* а для второй конфигурации ситуация меняется на противоположную. Это может говорить о том, что использовании средней эффективности по распознаванию номеров классов приводит к тому, миноритарный класс распознается лучше, чем мажоритарный, что отражается в меньшем значении *Acc*. Выводы справедливы как для обучающей, так и для тестовой выборки.

Таблица 4.8. Результаты для задачи Segment

Конфигурация	Обуч. Acc	Обуч. Recall	Тест. Acc	Тест. Recall
Acc+смещ.	0.912	0.900	0.912	0.900
Recall+смещ.	0.909	0.896	0.909	0.896
Acc+несмещ.	0.909	0.895	0.909	0.895
Recall+несмещ.	0.911	0.903	0.911	0.903

Несмотря на то, что задача Segment имеет 7 классов, все они сбалансированы. Результаты показывают, что в данном случае нет разницы в качестве классификации для всех модификаций, так как критерий качества не является чувствительным к смещениям классификации в данном случае.

Таблица 4.9. Результаты для задачи Satimage

Конфигурация	Обуч. Acc	Обуч. Recall	Тест. Acc	Тест. Recall
Acc+смещ.	0.837	0.756	0.829	0.748
Recall+смещ.	0.838	**0.767**	0.830	**0.757**
Acc+несмещ.	0.836	0.755	0.826	0.745
Recall+несмещ.	0.837	**0.763**	0.828	**0.755**

Задача Satimage оказалась малочувствительной к изменению критериев оценки качества классификации, однако незначительное улучшение в значениях меры *Recall* может быть отслежено, когда она используется в функции пригодности. Важным моментом для данной задачи является то, что алгоритм не смог построить ни одного хорошего правила для третьего класса, так что он практически всегда был неверно классифицирован во второй класс.

Чтобы показать действительные результаты классификации в таблицах 4.10-4.12 приведены усредненные матрицы ошибок для первой, второй и четвертой конфигурации для тестовой выборки, чтобы показать разницу для задачи Page-blocks. Данная задача была выбрана, так как она является самым ярким случаем, вследствие существенного дисбаланса в числе измерений различных классов. При этом при классификации возможны были случаи, в которых измерение не могло быть классифицировано ни в один из имеющихся классов. В данном случае, классификация не производилась, и измерение считалось классифицированным некорректно. Это было возможно, если ни одно правило не покрывало измерение, либо два правила покрывали измерения в одинаковой мере.

Таблица 4.10. Матрица ошибок для Page-blocks, Acc+смещ.

	Предск. 1	Предск. 2	Предск. 3	Предск. 4	Предск. 5	Неизв.
Истин. 1	**487,26**	2,80	0,06	0,53	0,60	0,03
Истин. 2	4,33	**28,03**	0,00	0,16	0,33	0,03
Истин. 3	**1,83**	0,00	0,93	0,00	0,00	0,03
Истин. 4	3,00	0,13	0,00	**5,36**	0,13	0,06
Истин. 5	**9,83**	0,00	0,13	0,00	1,50	0,03

Таблица 4.11.Матрица ошибок для Page-blocks, Recall+смещ.

	Предск. 1	Предск. 2	Предск. 3	Предск. 4	Предск. 5	Неизв.
Истин. 1	**432,06**	22,16	5,96	10,93	20,06	0,10
Истин. 2	1,36	**29,66**	0,30	0,43	1,13	0,00
Истин. 3	0,90	0,00	**1,53**	0,03	0,33	0,00
Истин. 4	0,53	0,53	0,13	**6,83**	0,63	0,03
Истин. 5	1,86	0,80	0,23	0,26	**8,30**	0,03

Таблица 4.12.Матрица ошибок для Page-blocks, Recall+несмещ.

	Предск. 1	Предск. 2	Предск. 3	Предск. 4	Предск. 5	Неизв.
Истин. 1	**437,83**	21,56	2,73	12,23	16,86	0,06
Истин. 2	1,43	**29,80**	0,03	0,50	1,06	0,06
Истин. 3	0,53	0,00	**2,00**	0,00	0,23	0,03
Истин. 4	0,30	0,13	0,16	**7,43**	0,63	0,03
Истин. 5	1,73	0,63	0,23	0,26	**8,63**	0,00

Из данных таблиц можно видеть, что использование меры *Recall* вместе со смещенной процедурой назначения номера класса позволяет классифицировать

большинство измерений верно, хотя точность на мажоритарном классе, также как и общая точность становится ниже. Из сравнения таблиц 4.11 и 4.12, можно видеть, что, несмотря на то, что несмещенная процедура не изменяет точность на мажоритарном классе, для классов 3 и 4 наблюдается значительное изменение в среднем числе верно классифицированных объектов. Результаты опубликованы в [111, 11, 13].

Таким образом, предложенная модификация процедуры назначения наилучшего номера класса позволяет строить более качественные классификаторы на нечеткой логике, что было показано на ряде задач.

4.3 Методы селекции обучающих примеров для задач классификации

В предыдущих пунктах данной главы было рассмотрено применение описанного в главе 3 самонастраивающегося гибридного эволюционного алгоритма построения баз нечетких правил для задач с несбалансированными данными, а также его модификация. В данном пункте будут рассмотрены подходы к уменьшению объемов данных, требующих обработки для снижения времени работы предложенного подхода, а также для повышения качества классификации.

Методы снижения объемов данных, называемые также Data Reduction (DR) ставят своей целью уменьшение объемов обучающей выборки. Существует несколько групп методов, в том числе селекция обучающего множества (Training Set Selection, TSS) [37], включающий активное обучение (Active Learning), селекцию обучающих примеров (Instance Selection) и отбор информативных признаков (Feature Selection) [101].

Методы активного обучения концентрируются на задачах обучения с учителем с возможностью запроса новых измерений, селекция обучающих примеров концентрируется на составлении сокращенной репрезентативной подвыборки данных. В данной работе будут исследованы методы селекции обучающих примеров, методы отбора информативных признаков рассматриваться не будут.

Селекция обучающих примеров [91, 39, 58], или примеров из обучающей выборки применяется для методов обучения с учителем, когда имеется выборка с известными номерами классов. Как правило, обучающая выборка T имеет нерелевантные измерения, многократное повторение схожих измерений и прочую бесполезную информацию. Целью метода селекции обучающих примеров является выделить подмножество S из множества T, которое не содержит лишних измерений, при том что точность на S примерно равна точности на T. Классические методы селекции обучающих примеров могут либо последовательно добавлять измерения в S, начиная с пустого множества, либо постепенно удалять их.

Подобно отбору информативных признаков, выделяют 2 группы методов селекции обучающих примеров: Wrapper и Filter. Wrapper использует информацию о точности классификации, в то время как Filter не зависит от классификации. Сначала рассмотрим Wrapper-подходы.

Первые разработки в области селекции обучающих примеров относятся к концу 60-х – началу 70-х годов, и в основном базируются на методе k ближайших соседей (kNN). Среди них такие методы, как Condensed Nearest Neighbor (CNN) [58], SNN (Selective Nearest Neighbour rule) [99], Generalized Condensed Nearest Neighbor rule (GCNN) [42]. Данные методы позволяют снизить объем выборки, однако имеют тенденцию к сохранению зашумленных данных. Для выбора измерений используются понятия ближайших соседей и ближайших врагов.

Добиться исключения зашумленных данных позволяет метод ENN (Edited Nearest Neighbor) [118], в котором измерение исключается, если его класс отличается от класса ближайших соседей.

Методы DROP1, DROP2… DROP5 (Decremental Reduction Optimization Procedure) [119] базируются на понятии связи. Связанными с измерением p называются измерения, для которых p является одним из k ближайших соседей. DROP1 удаляет p, если связанные с p также верно классифицируются и без p. DROP1 позволяет удалять зашумленные данные, однако не в полном объеме. Проблемы DROP1 решены в DROP2, в котором связанными с p считаются все

измерения. DROP3 и DROP4 сначала используют метод ENN, а затем применяют DROP2. DROP5 основан на DROP2, однако он начинает с удаления ближайших врагов (т.е. ближайших измерений других классов).

Ряд методов, такие как Iterative Case Filtering algorithm (ICF) [36] и C-Pruner используют множества соседства и связи соответственно для исключения зашумленных измерений. К примеру, если мощность множества соседства больше мощности множества связи, то измерение p считается зашумленным.

Эволюционные алгоритмы также были использованы для селекции обучающих примеров [38]. При этом включение или исключение конкретного измерения из выборки кодируется в хромосому. Далее новая выборка используется для классификации.

Помимо ГА, также разработаны методы, использующие искусственные иммунные системы для селекции обучающих примеров, в частности Clonal Selection Algorithm (CSA) [53]. CSA отбирает по одному антигену на класс, затем патоген (набор измерений) атакует иммунную систему и набор клонов – антигенов, наиболее схожих с патогеном, мутирует для того, чтобы выбрать наилучших для отражения атаки патогена.

Другой набор методов использует обратный последовательный поиск для селекции обучающих примеров. Данные методы исключают измерения по одному, при этом на каждом шаге исключается то измерение, которое вносит наименьший вклад в качество классификации. Процесс повторяется до тех пор, пока точность классификации не начнет падать. В алгоритме SFS (Sequential Floating Search) [96] помимо этого, удаленные ранее измерения могут быть возвращены в выборку.

Filter-подходы не используют точность классификации в этом смысле являются более универсальными, и часто более быстродействующими, в особенности для больших выборок.

Определенный набор методов использует понятия пограничных и внутренних измерений. Измерение является пограничным, если его ближайший

сосед относится к другому классу. Пограничные измерения предоставляют важную информацию о расположении классов.

К примеру, методы POP (Pattern by Ordered Projections) [98] и POC-NN(Pair Opposite Class-NearestNeighbor) [97] удаляют внутренние измерения классов и выбирают только граничные измерения. Данные методы отличаются способом определения пограничных измерений.

Другая группа методов использует процедуру кластеризации для селекции обучающих примеров, при этом центры кластеров считаются новыми измерениями, среди них методы GCM (Generalized-Modified Chang Algorithm) [87] и NSB (Nearest Sub-class Classifier) [116]. Метод OSC (Object Selection by Clustering) [92] также использует кластеризацию, однако в гомогенных кластерах ищутся внутренние измерения, в то время как в негомогенных кластерах – граничные.

Некоторые методы используют взвешивание измерений для последующего выбора некоторого набора измерений, чей вес выше некоторого порога. Метод WP (Weighting Prototypes) [94] использует градиентный спуск для расчета веса по ближайшим соседям и ближайшим врагам. Метод PSR (Prototype Selection by Relevance) [93] рассчитывает релевантность каждого измерения в выборке в смысле усредненной схожести с другими измерениями в классе. Помимо этого, также выбираются наиболее похожие измерения из другого класса, то есть пограничные измерения.

Для большинства методов селекции обучающих примеров характерно то, что время вычислений увеличивается. Однако, было предложено несколько алгоритмов, позволяющих снизить это время, например, разбив выборку на несколько непересекающихся наборов. После этого, для каждого набора применяется селекция обучающих примеров. Далее отобранные множества собираются в выборку, и процедура селекции обучающих примеров повторяется рекурсивно.

В следующем пункте данной главы будет предложен метод селекции обучающих примеров, реализующий Wrapper-подход с многократной адаптивной

вероятностной схемой построения подвыбороки, и применимый для итерационных методов формирования классификаторов.

4.4 Адаптивный алгоритм селекции обучающих примеров для задач классификации с несбалансированными данными

Снижение объемов обрабатываемых данных неминуемо приводит к уменьшению информации, доступной в процессе обучения. При этом, однако, такое снижение не обязательно отрицательно отражается на качестве классификации. В некоторых случаях исключение измерений из обучающей выборки может привести к повышению качества классификации, вследствие того, что отброшенные измерения представляли собой шум, многократно повторялись и т.д.

Таким образом, значительное число методов селекции обучающих примеров, описанных в предыдущем пункте данной главы, нацелены не только на снижение объемов данных, но также и на повышение качества классификации. Однако данные методы в основном предназначены для однократного выбора некоторого множества измерений, и ставят своей целью именно формирование данного множества, а не построение качественного классификатора.

Предлагаемый подход селекции обучающих примеров нацелен на классификаторы, использующие множество итераций в ходе обучения. Основной идеей при его разработке было создать метод, позволяющий не только существенно снизить объем обрабатываемой информации, и, следовательно, требуемых вычислительных ресурсов, но также повысить качество классификации в смысле приведенных ранее мер.

Данный метод не требует какой-либо предобработки данных, не основан на методе k ближайших соседей и не требует вычисления расстояний между объектами. Вместо этого, измерения выбираются на основании точности их классификации обучающим алгоритмом.

Опишем идею метода более детально. На первом этапе из обучающей выборки формируется подвыборка фиксированного (заданного пользователем)

объема. При этом измерения выбираются с равной вероятностью. Далее запускается процесс обучения классификатора в течение нескоторого числа итераций (поколений), называемого периодом адаптации. При этом обучение происходит только по сформированной подвыборке.

При этом каждому измерению из выборки назначается счетчик U_i, обозначающий число успешных использований данного измерения. Изначально счетчики устанавливаются $U_i = 1, i = 1 \dots n,$ и далее изменяются для каждого обучающего примера. По окончанию периода адаптации, лучшее текущее решение, то есть лучшее для подвыборки, используется чтобы обновить счетчики U_i. При этом обновляются только счетчики тех измерений, которые были в обучающей подвыборке. Если измерение j было классифицировано верно, то $U_i = U_i + 1$, иначе $U_i = 1$.

Таким образом, увеличиваются счетчики для измерений, которые были распознаны классификатором верно. Для некорректно классифицированных измерений счетчики сбрасываются. После обновления счетчиков, формируется новая подвыборка, с учетом новых значений счетчиков. При этом вероятность того, что измерение i будет выбрано, рассчитывается по формуле:

$$p_i = \frac{1/U_i}{\sum_{j=1,n} 1/U_j}$$

В соответствии с данной формулой, увеличение счетчика приводит к уменьшению вероятности включения измерения в обучающую подвыборку. Знаменатель при этом выполняет роль нормировки.

Таким образом, алгоритм селекции обучающих примеров назначает меньше вероятности примерам, которые могут быть легко классифицированы. В то же время, сложно классифицированные и ранее не использованные измерения получают большие вероятности быть выбранными. Данная процедура реализует два основных принципа: исследование ранее неизвестных областей пространства признаков и использование информации о качестве классификации для построения лучшего разделения между классами.

В процессе обучения, лучшее решение для каждой подвыборки меняется после каждого периода адаптации, так как зависит от выбранных измерений. Вследствие этого, вероятности выбора также меняются, так как лучшее для подвыборки решение может показывать различные результаты на всей выборке. Из-за этого подвыборка постоянно меняется, вследствие чего могут быть получены более разнообразные решения, что позволяет улучшить процесс поиска. В случае использования эволюционного алгоритма в качестве алгоритма обучения, на каждом новом периоде адаптации сохраняется популяция решений, которые были хороши в смысле принятой функции пригодности на предыдущем периоде. Данная популяция в определенной мере позволяет классифицировать новую подвыборку, но как правило с меньшей точностью.

Для демонстрации процедуры изменения вероятностей, рассмотрим с 8 измерениями, приведенный на рисунке 4.1.

Рис. 4.1. Изменение вероятностей и счетчиков

В начале все вероятности равны между собой, и все значения U_i равны единице. Выбранные измерения обозначены серым. После первого поколения, третье измерение классифицировано некорректно, так что U_3 становится равным 1, в то время как $U_6 = 2$. Другие счетчики при этом не изменяются. После

нескольких поколений, значения p_i изменяются до некоторых значений, которые показывают важность измерения для классификатора на данном этапе. Таким образом, измерения 1 и 7 получают большие вероятности быть выбранными.

На рисунке 4.2 приведен графический пример того, какие из измерений выбираются в процессе обучения, и каким образом селекции обучающих примеров позволяет повысить качество классификации. Пример представляет собой двухклассовую задачу с 31 измерениями, различные классы показаны квадратами и крестиками.

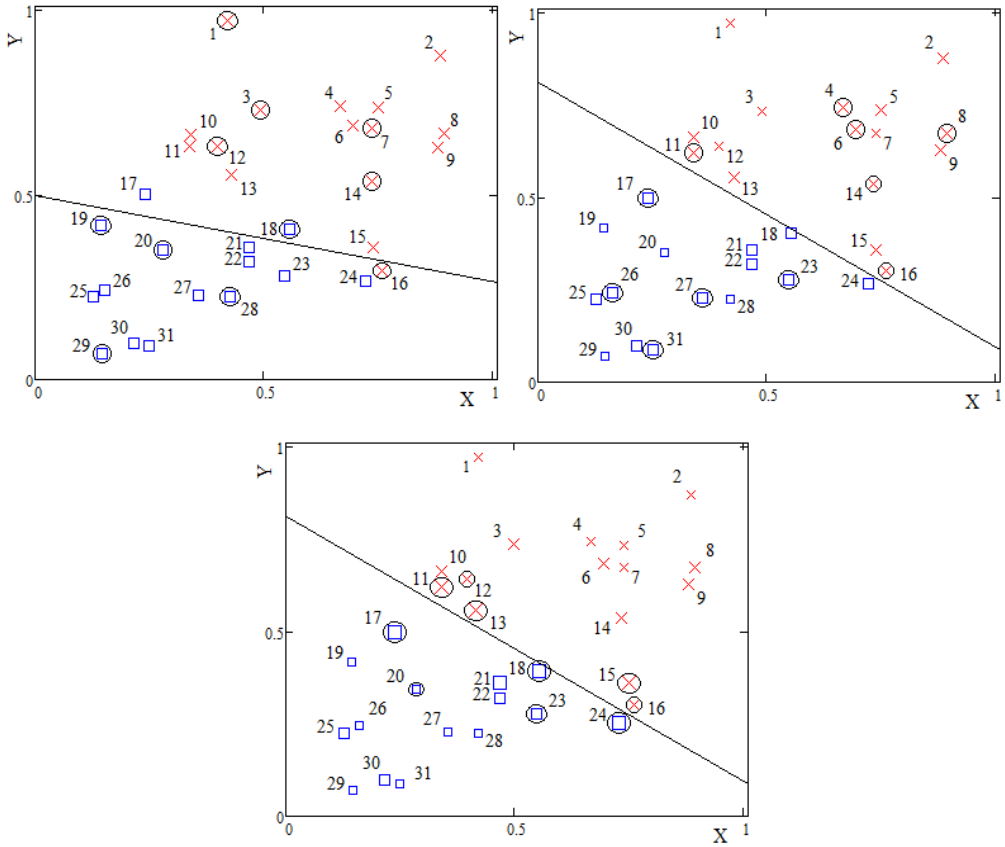


Рис. 4.2. Изменение выбираемых измерений

Слева показано начало работы алгоритма, при котором все измерения имеют равные вероятности быть выбранными. Вероятности показаны размерами значков. Измерения в кружках – это те, которые были выбраны на текущем периоде адаптации. Разделяющая поверхность, полученная к концу периода адаптации, не классифицирует все измерения верно, в частности, измерения 16 и 18 классифицированы некорректно. Справа показан следующий период адаптации. Измерения 16 и 18 не изменили своего размера (т.е. вероятности), в

сравнении с другими измерениями, в то время как все остальные (1, 3, 7, 12, 14, 19, 20, 28, 29), использованные и классифицированные верно, получили меньшие вероятности (меньший размер). Нижний график на рисунке 4.2 показывает ситуацию после нескольких периодов адаптации. Здесь те измерения, которые близки к разделяющей поверхности между классами, получили большие вероятности, быть выбранными, в то время как остальные измерения, которые лежат дальше, получили меньшие вероятности.

Также, при применении данного метода к эволюционным алгоритмам построения классификаторов существует несколько особенностей. В частности, для описанного в предыдущей главе алгоритма, на каждом поколении лучшее найденное решение для подвыборки проверялось на всей выборке. Этот шаг необходим для того, чтобы не потерять в ходе поиска лучшее найденное решение. Более того, лучшее решение для всей выборки всегда включалось в популяцию, наравне с лучшим решением для подвыборки. По окончании периода адаптации, все индивиды в текущей популяции проверялись на всей выборке. Данный шаг необходим, так как, несмотря на то, что лучшее решение для подвыборки оказывается также лучшим и для всей выборки, популяция может содержать другие решения, обладающие лучшей обобщающей способностью на всей обучающей выборке.

Стоит отметить, что выше также не был затронут вопрос о сохранении распределения числа элементов по классам в подвыборке. В случае использования описанной процедуры формирования подвыборки в чистом виде, из-за использования вероятностного подхода соотношение числа объектов в классах может отличаться от изначального. Поэтому, в данной процедуре формирования подвыборки необходимо учитывать распределение объектов по классам.

Одним из методов учета этого распределения может быть так называемый стратифицирующий подход, при котором сохраняется исходное распределение объектов. Данный подход используется, к примеру, при разбиении выборки на обучающую и тестовую, либо в процедуре кросс-валидации. Подход является

важным для сбалансированных задач, так как позволяет сохранить соотношение числа объектов разных классов.

Однако в случае решения задач с несбалансированными данными, стратифицирующий подход может быть не лучшим решением. Так как в предлагаемом методе селекции обучающих примеров могут быть выбраны различные группы объектов, то для несбалансированных задач возможно создавать подвыборки, которые более сбалансированы, чем изначальная выборка. Данная идея отражена в балансирующем подходе, который строит подвыборку так, чтобы распределение объектов по классом было как можно равномернее. Естественно, в этом случае слабо представленные классы могут включаться в подвыборку в полном объеме, однако, к примеру, в стратифицированном методе в подвыборку могут включаться лишь 1-2 измерения.

Следующий пункт данной главы будет посвящен практической реализации метода селекции обучающих примеров и сравнению предложенных подходов на ряде задач классификации с несбалансированными данными.

4.5 Программная реализация алгоритма, тестирование и результаты

Для выяснения эффективности описанного в предыдущем пункте метода селекции обучающих примеров для обучения классификаторов, была модифицирована программная система, реализующая самонастраивающийся гибридный эволюционный алгоритм формирования баз нечетких правил. Итоговая программная система получила название HEFCA_IS, модульную структуру и расширенные возможности по распараллеливанию и встраиванию в другие программные системы. Так как данный метод работает с популяцией решений, то для него применялось сохранение лучшего индивида для всей выборки на каждом поколении и его включение в популяцию. Результаты опубликованы в [12, 109, 110].

Для выяснения преимуществ адаптивного алгоритма селекции обучающих примеров в сравнении с классическим подходом была произведена серия вычислительных экспериментов на ряде задач классификации, которым

свойственен большой объем обучающей выборки. 9 задач были взяты с репозиториев KEEL [26] и UCI Machine Learning [27], список задач представлен в таблице 4.13.

Таблица 4.13. Использованные задачи классификации

Задача	Число измерений	Число признаков	Число классов
Magic	19020	10	2
Page-blocks	5472	10	5
Penbased	10992	16	10
Phoneme	5404	5	2
Ring	7400	20	2
Satimage	6435	36	6
Segment	2310	19	7
Texture	5500	40	11
Twonorm	7400	20	2

Некоторые из этих задач уже были использованы ранее во втором пункте, в частности задачи Page-blocks, Phoneme, Segment и Satmage. Следующая таблица 4.14 демонстрирует распределение объектов по классам в данных задачах.

Таблица 4.14. Распределение объектов по классам

	1	2	3	4	5	6	7	8	9	10	11
Magic	12332	6688	-	-	-	-	-	-	-	-	-
Page-blocks	4913	329	28	87	115	-	-	-	-	-	-
Penbased	1143	1143	1144	1055	1144	1055	1056	1142	1055	1055	
Phoneme	3818	1586	-	-	-	-	-	-	-	-	-
Ring	3664	3736	-	-	-	-	-	-	-	-	-
Satimage	1533	703	1358	626	707	1508	-	-	-	-	-
Segment	330	330	330	330	330	330	330	-	-	-	-
Texture	500	500	500	500	500	500	500	500	500	500	500
Twonorm	3703	3697	-	-	-	-	-	-	-	-	-

Наибольшими мерами дисбаланса отличаются задачи Page-blocks и Phoneme, Satimage и Magic, остальные задачи либо сбалансированы, либо почти сбалансированы, т.е. отличие в числе измерений незначительно.В ходе тестирования на каждом поколении алгоритма сохранялась лучшая найденная база правил, а также матрица ошибок, рассчитанная по этой базе правил как на обучающей, так и на тестовой выборке. Наличие матрицы ошибок позволяет рассчитывать все меры качества классификации, представленные в таблицах 4.2-4.3.

Для тестирования были выбраны следующие параметры: число индивидов – 100, число поколений – 10000, максимальное число правил – 40, размер подвыборки варьировался от 5% до 30% с шагом 5%, длина периода адаптации устанавливалась равной 50, 100, 200 и 400 поколений. Был протестирован стандартный подход без селекции обучающих примеров, а также модифицированный подход со стратифицированной и балансирующей стратегией. Для каждого набора настроек процедура стратифицированной 10-частной кросс-валидации повторялась 2 раза, значения мер качества классификации усреднялись по всем 20 запускам.

В таблице 4.15 представлены результаты тестирования для оригинального алгоритма без селекции обучающих примеров для всех задач. Приведены значения общей ошибки классификации на обучающей и тестовой выборке, среднее число правил, средняя длина правила, а также среднее время работы алгоритма в минутах.

Таблица 4.15. Результаты классификации стандартного алгоритма

Задача	Ошибка на обучающей	Ошибка на тестовой	Число правил	Длина правила	Время работы
Magic	15.06	15.73	12.6	3.82	370.62
Page-blocks	3.52	3.96	10.1	3.49	94.48
Penbased	7.06	7.46	30.5	6.23	385.2
Phoneme	15.03	16.48	18.3	3.01	84.95
Ring	4.64	5.82	26.6	3.83	226.70
Satimage	12.22	14.22	20.4	11.06	345.82
Segment	4.55	6.45	22.2	6.69	146.18
Texture	6.50	7.75	25.8	14.90	352.26
Twonorm	4.42	6.06	17.4	7.40	254.88

В следующей таблице приведены лучшие результаты, полученные с применением алгоритма селекции обучающих примеров для всех задач с использованием стратифицированной стратегии. Приведены значения общей ошибки.

Таблица 4.16. Лучшие результаты классификации со стратифицированной стратегией

Задача	Ошибка на обучающей	Ошибка на тестовой	Число правил	Длина правила	Время работы
Magic	14.98	15.23	10.7	4.77	121.19
Page-blocks	3.59	3.83	7.8	4.94	20.73
Penbased	3.18	3.87	31.3	6.42	65.36
Phoneme	16.29	16.77	12.5	3.15	15.45
Ring	3.38	4.86	30.0	4.12	54.08
Satimage	11.28	13.05	27.9	6.72	96.31
Segment	3.01	5.13	25.1	6.32	39.21
Texture	3.34	4.31	28.1	11.76	97.47
Twonorm	3.07	4.66	18.7	7.45	58.92

Лучшими конфигурациями для стратифицированной стратегии оказались для задачи Magic: 30%, 400; Page-blocks: 30%, 200; Penbased: 15%, 50; Phoneme: 25%, 200; Ring: 25%, 50; Satimage: 30%, 50; Segment: 30%, 50; Texture: 25%, 50; Twonorm: 30%, 100. Таким образом, наибольшая точность была получена при максимальном объеме подвыборки в большинстве случаев. Таблица 4.17 содержит сравнение таблиц 4.15 и 4.16. Относительное время работы приведено в процентах.

Таблица 4.17. Разница между таблицами 4.15 и 4.16

Задача	Ошибка на обучающей	Ошибка на тестовой	Число правил	Длина правила	Время работы
Magic	0.08	0.5	1.9	-0.95	32.70
Page-blocks	-0.07	0.13	2.3	-1.45	21.94
Penbased	3.88	3.59	-0.8	-0.19	16.97
Phoneme	-1.26	-0.29	5.8	-0.14	18.19
Ring	1.26	0.96	-3.4	-0.29	23.86
Satimage	0.94	1.17	-7.45	4.34	27.85
Segment	1.54	1.32	-2.9	0.37	26.82
Texture	3.16	3.44	-2.3	3.14	27.67
Twonorm	1.35	1.4	-1.3	-0.05	23.12

Как можно видеть, практически для всех задач применение селекции обучающих примеров привело к существенному увеличению точности классификации. Разница составила до 3.59% на тестовой выборке. Отрицательный эффект наблюдался только для задачи Phoneme. При этом время работы

алгоритма составляло от 18% до 32% от изначального. Число правил в базе при этом уменьшилось для 6 задач из 9, также как и длина правил. Таким образом, селекции обучающих примеров для стратифицированной стратегии позволил получить более компактные и точные базы правил за более короткое время. В следующих двух таблицах приведены результаты для балансирующей стратегии и их сравнение с оригинальным алгоритмом.

Таблица 4.18. Лучшие результаты классификации с балансирующей стратегией

Задача	Ошибка на обучающей	Ошибка на тестовой	Число правил	Длина правила	Время работы
Magic	14.62	15.08	17.3	3.63	129.65
Page-blocks	2.71	3.25	18.9	4.82	18.53
Penbased	3.27	3.81	30.8	6.11	91.42
Phoneme	15.63	16.88	24.0	2.84	19.62
Ring	3.23	5.08	30.2	3.85	68.23
Satimage	10.57	12.93	27.2	5.84	85.12
Segment	3.55	5.19	25.1	6.24	32.40
Texture	3.37	4.45	27.0	12.81	114.79
Twonorm	4.03	4.81	15.0	7.74	38.11

Таблица 4.19. Разница между таблицами 4.15 и 4.18

Задача	Ошибка на обучающей	Ошибка на тестовой	Число правил	Длина правила	Время работы
Magic	0.44	0.65	-4.7	0.19	34,98
Page-blocks	0.81	0.71	-8.8	-1.33	19,61
Penbased	3.79	3.65	-0.3	0.12	23,73
Phoneme	-0.6	-0.4	-5.75	0.17	23,10
Ring	1.41	0.74	-3.6	-0.02	30,10
Satimage	1.65	1.29	-6.8	5.22	24,61
Segment	1	1.26	-2.9	0.45	22,16
Texture	3.13	3.3	-1.2	2.09	32,59
Twonorm	0.39	1.25	2.4	-0.34	14,95

Применение балансирующей стратегии изменило поведение алгоритма на большинстве наборов данных. Для задачи Phoneme отставание существенно сократилось и составило всего 0.4% на тестовой выборке. Наибольшее улучшение было снова продемонстрировано для задачи Penbased, 3.65%. Однако при этом

среднее число правил возросло практически для всех задач, что говорит о том, что алгоритм сформировал большие по размеру базы. Лучшими конфигурациями оказались для задачи Magic: 30%, 200; Page-blocks: 20%, 50; Penbased: 25%, 100; Phoneme: 30%, 50; Ring: 30%, 50; Satimage: 30%, 50; Segment: 25%, 50; Texture: 30%, 50; Twonorm: 20%, 200.

Таблица 4.20. Сравнение мер $(1 - Recall_M) * 100$

Задача	Обуч. Оригинал.	Тест. Оригинал.	Обуч. Стратиф.	Тест. Стратиф.	Обуч. Баланс.	Тест. Баланс.
Magic	18.74	19.47	18.83	19.12	**17.45**	**17.99**
Page-blocks	34.85	36.80	36.43	38.52	**19.64**	**23.05**
Penbased	7.04	7.44	3.16	3.84	3.25	**3.80**
Phoneme	18.98	20.80	21.48	22.38	**16.60**	**18.10**
Ring	4.64	5.82	3.38	**4.87**	**3.23**	5.09
Satimage	15.78	17.98	15.85	17.73	**13.63**	**15.28**
Segment	4.55	6.45	**3.01**	**5.13**	3.55	5.19
Texture	6.50	7.75	**3.34**	**4.31**	3.37	4.45
Twonorm	4.42	6.06	**3.07**	**4.66**	4.03	4.81

Однако наибольший эффект был продемонстрирован не в смысле общей точности классификации, а в смысле усредненной точности по каждому классу, или же меры $Recall_M$. В таблице приведены величины $(1 - Recall_M) * 100$ для сравнения оригинального алгоритма со стратифицированной и балансирующей стратегиями.

По полученным результатам можно заключить, что использование балансирующей стратегии не только повышает общую точность классификации, но также позволяет распознавать различные классы более точно. При этом выбранные конфигурации выбирались по максимальной точности классификации. Однако, из-за того, что в подавляющем большинстве случаев максимальная точность была достигнута при 30% измерений в подвыборке, то применение балансирующей стратегии, например, к задаче Page-blocks, не позволило сформировать полностью сбалансированную подвыборку. Тем не менее, улучшение для этой задачи составляет порядка 13% по сравнению с оригинальным алгоритмом. Если принять во внимание тот факт, что

балансирующая стратегия также имеет тенденцию к увеличению числа правил, то можно предположить, что данная комбинация параметров формирует базы правил, которые в значительно лучшей степени решают задачу классификации.

Приведем для задачи Page-blocks значения меры $(1 - Recall_M) * 100$ для всех длин периода адаптации и для всех размеров обучающей подвыборки в таблице 4.21 в случае использования балансирующей стратегии для тестовой выборки.

Таблица 4.21. Величины мер $(1 - Recall_M) * 100$ для задачи Page-blocks

Задача	50	100	200	400
5%	10.18	12.92	15.18	12.06
10%	12.92	13.71	14.59	17.51
15%	20.51	19.63	18.35	19.26
20%	23.05	21.23	22.72	22.36
25%	28.35	26.14	23.05	22.24
30%	28.04	27.73	28.42	29.51

Как можно видеть из таблицы, наименьшие значения $(1 - Recall_M) * 100$ были получены для случая, когда размер подвыборки был минимален, то есть, когда выборка была максимально сбалансирована. Стоит отметить, что для данной задачи даже при 5% не удалось сформировать полностью сбалансированную выборку, из-за того, что третий класс содержит всего 28 измерений. Остальные классы в этом случае содержали по 54 измерений. Величина смещения в сторону мажоритарных классов в данном случае была практически нивелирована благодаря балансирующей стратегии. Для сравнения, также приведем значения точности для тех же параметров для данной задачи.

Таблица 4.22. Величины общей точности для задачи Page-blocks

Задача	50	100	200	400
5%	*7.95*	8.44	9.25	8.48
10%	*5.81*	6.03	6.67	7.27
15%	*4.00*	4.50	4.90	5.77
20%	***3.25***	**3.40**	3.93	4.40
25%	3.78	3.78	***3.69***	4.15
30%	*3.38*	3.69	3.78	**4.06**

Для 5% обучающей выборки значения точности снижаются практически в 2 раза в сравнении с наилучшими значениями, характерными для 30% в

подвыборке. Тем не менее, варианты классификаторов, полученные при 5% могут на практике оказаться более предпочтительными, так как такие классификаторы способны одинаково точно распознавать все классы. Чтобы продемонстрировать данный эффект, приведем также матрицы ошибок для задачи Page-blocks для оригинального алгоритма, и активного селекции обучающих примеров с балансирующей стратегией, усредненные по всем прогонам алгоритма.

Таблица 4.23. Матрица ошибок для задачи Page-blocks, оригинальный алгоритм

Задача	Предск. 1	Предск. 2	Предск. 3	Предск. 4	Предск. 5	Неизвест.
Истин. 1	**487.18**	2.90	0	0.36	0.90	0
Истин. 2	3.09	**29**	0.18	0.18	0.45	0
Истин. 3	**1.72**	0	1.09	0	0	0
Истин. 4	2.45	0.09	0	**6**	0.18	0
Истин. 5	**8.73**	0	0.27	0	2.54	0

Таблица 4.24. Матрица ошибок для задачи Page-blocks, селекции обучающих примеров с балансирующей стратегией, 5%, 50 поколений

Задача	Предск. 1	Предск. 2	Предск. 3	Предск. 4	Предск. 5	Неизвест.
Истин. 1	**452**	19.36	3.72	6.54	9.64	0.09
Истин. 2	1.72	**30.27**	0.09	0.36	0.36	0.09
Истин. 3	0.18	0	**2.54**	0.09	0	0
Истин. 4	0	0.27	0	**8.27**	0.18	0
Истин. 5	1.27	0.36	0.18	0.36	**9.36**	0

В каждой строке таблиц 4.23 и 4.24 выделены классы, к которым алгоритм отнес наибольшее число измерений. Как можно видеть, оригинальный алгоритм точно классифицирует мажоритарный первый класс, а также второй класс, однако малые по объему классы 3 и 5 относятся им к первому классу, и по сути не распознаются. С применением балансирующей стратегии ситуация меняется: большинство объектов каждого класса классифицированы верно, хотя точность по первому классу ниже. Последнее приводит к уменьшению общей точности классификации, однако такой вариант классификатора более предпочтителен.

Для остальных задач были получены схожие результаты. Основная тенденция заключается в том, что сбалансированные классификаторы можно

получить с меньшим объемом подвыборки, причем уменьшать объем подвыборки есть смысл только до тех пор, пока классы не станут полностью сбалансированными. Всех случаях в функции пригодности использовались лишь значения общей точности классификации по выборке. Чтобы показать зависимость точности классификации от длины периода адаптации и размера подвыборки, приведем графическое изображение зависимости в виде поверхности для тестовой выборки.

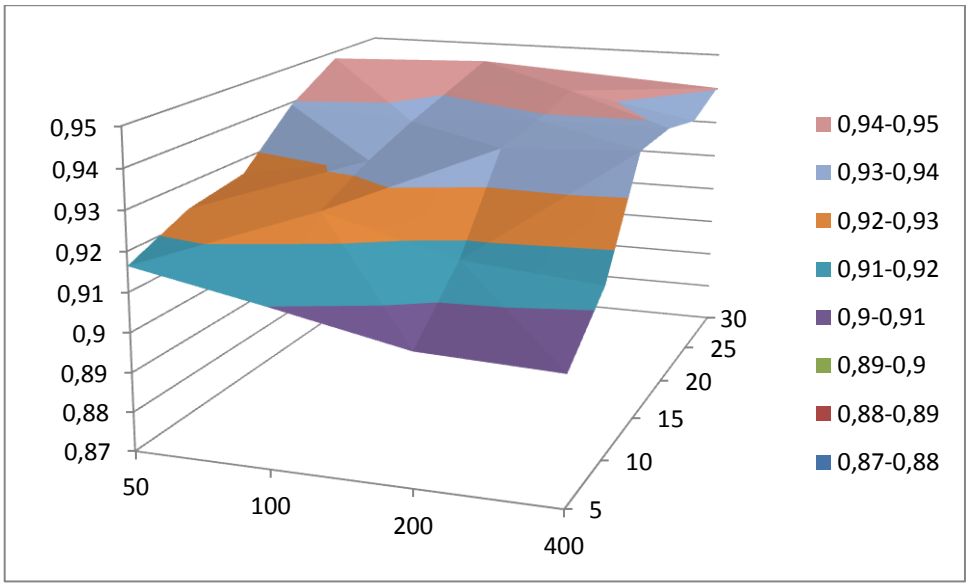


Рис. 4.3 Зависимость точности от параметров селекции обучающих примеров, Segment

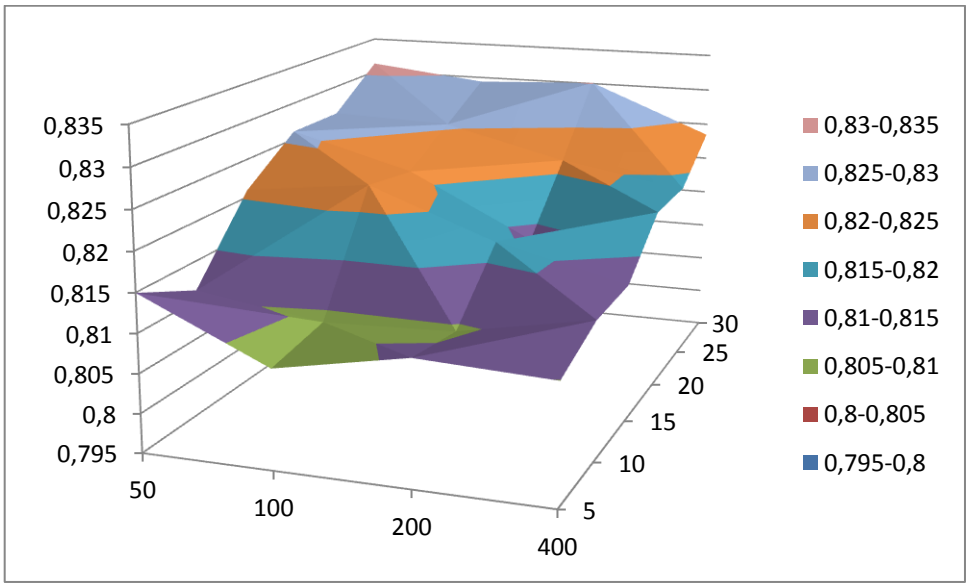


Рис. 4.4 Зависимость точности от параметров селекции обучающих примеров, Phoneme

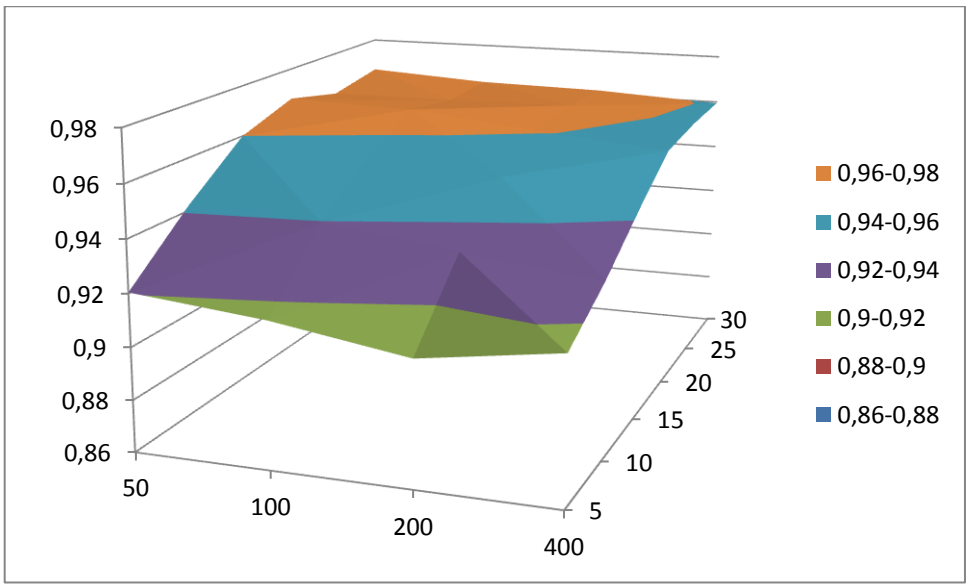


Рис. 4.5 Зависимость точности от параметров селекции обучающих примеров, Page-blocks

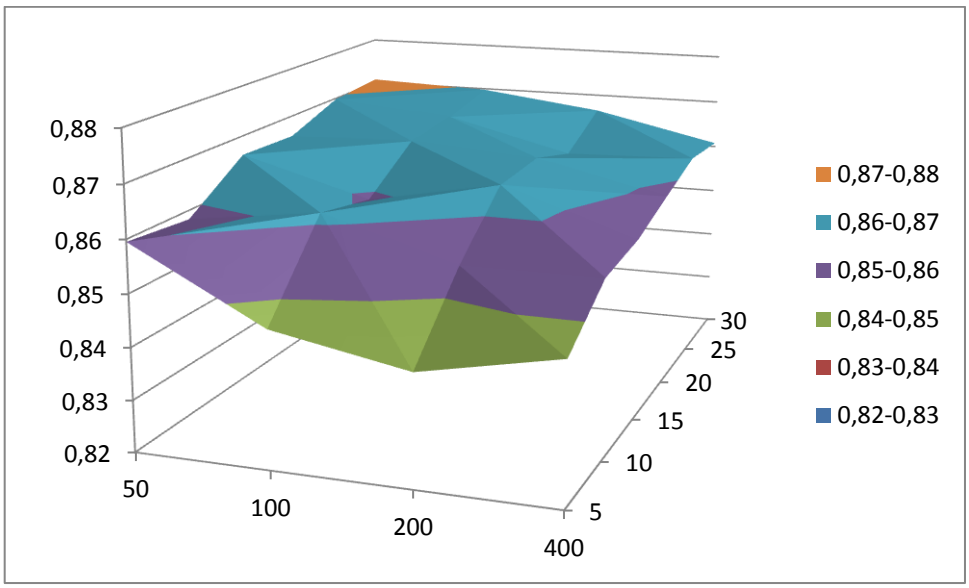


Рис. 4.6 Зависимость точности от параметров селекции обучающих примеров, Satimage

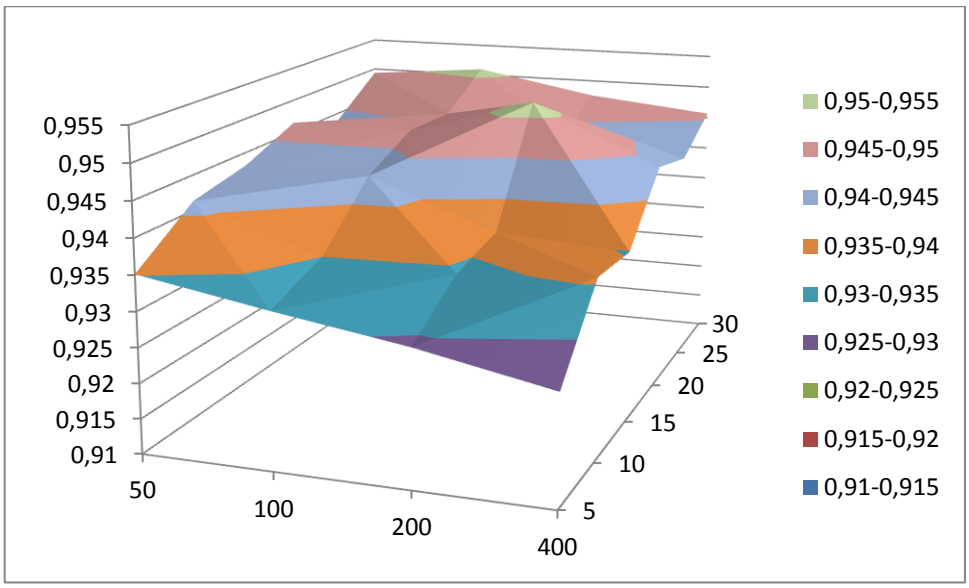


Рис. 4.7 Зависимость точности от параметров селекции обучающих примеров, Twonorm

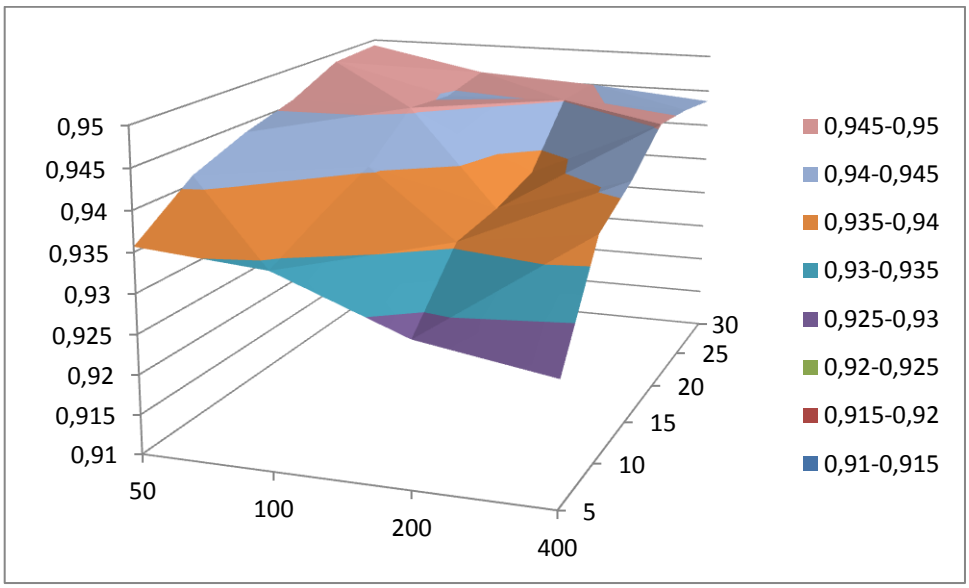


Рис. 4.8 Зависимость точности от параметров селекции обучающих примеров, Ring

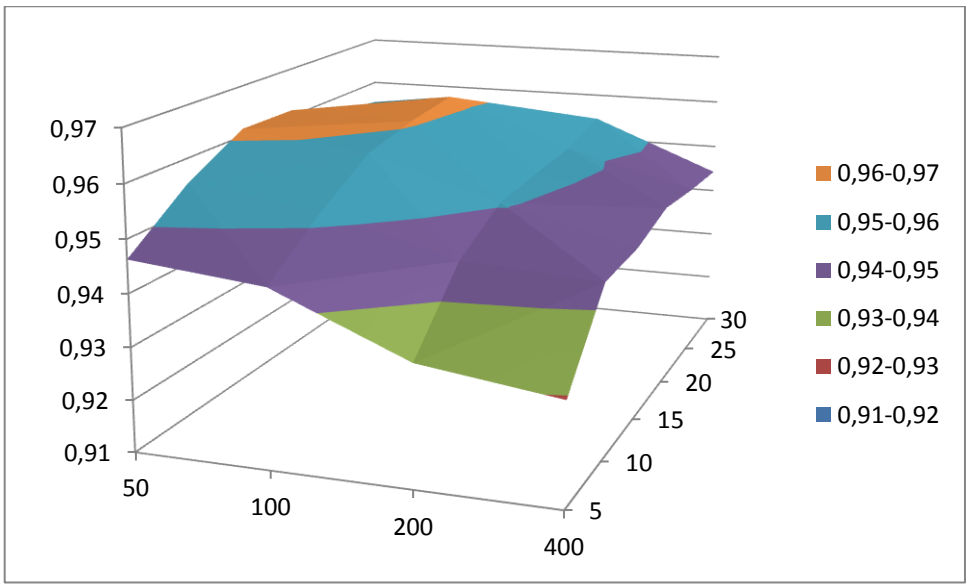


Рис. 4.9 Зависимость точности от параметров селекции обучающих примеров, Penbased

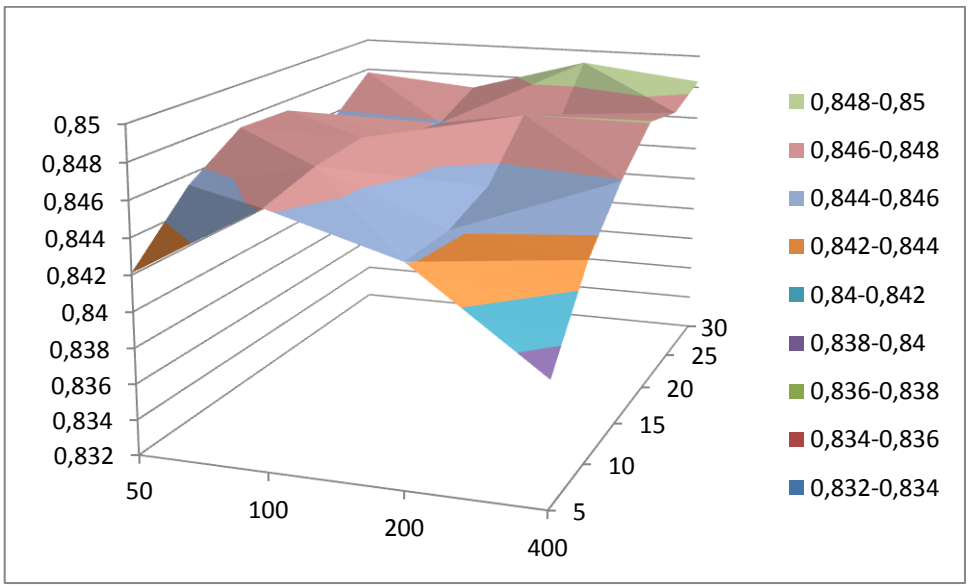


Рис. 4.10 Зависимость точности от параметров селекции обучающих примеров, Magic

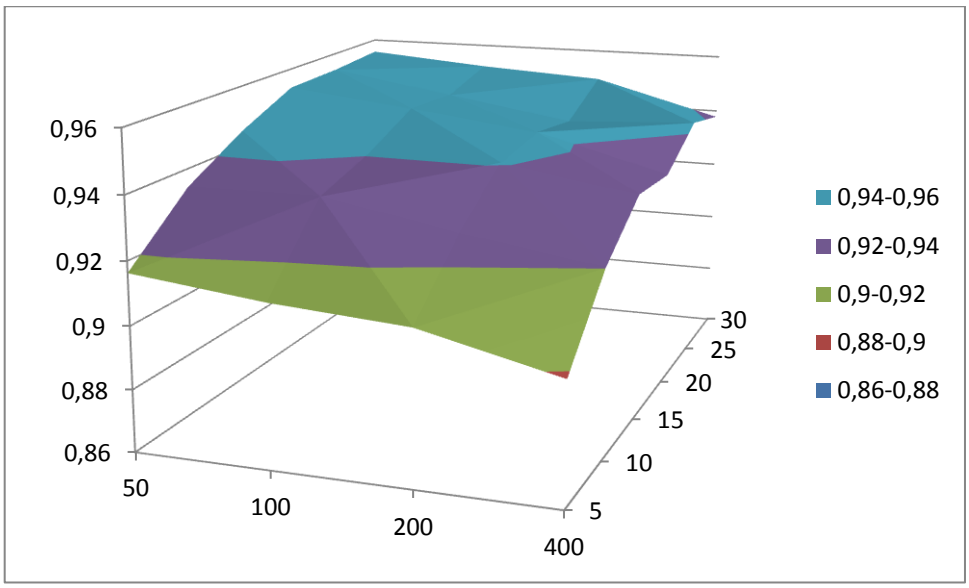


Рис. 4.11 Зависимость точности от параметров селекции обучающих примеров, Texture

Как можно заметить, в подавляющем большинстве случаев при увеличении размера подвыборки точность классификации увеличивается. Зависимость точности от длины периода адаптации не столь однозначна, однако для большинства задач более короткие периоды оказываются предпочтительнее. Это может говорить о том, что даже за 50 поколений алгоритм успевает перестроиться под новую подвыборку. Более того, короткий период адаптации при фиксированном максимальном числе поколений позволяет произвести большее число обновлений значений счетчиков U_i, и лучше выделить проблемные области пространства переменных.

Для сравнения времени работы алгоритма селекции обучающих примеров с оригинальным методом, результаты по времени работы для каждого размера обучающей подвыборки усреднялись по всем длинам периодов адаптации. Полученные значения сравнивались со значениями для оригинального алгоритма. Далее рассчитывалось ускорение, как $T_{\text{ориг}}/T$. Значения ускорения для всех задач приведены в таблице 4.24.

Максимальное достигнутое ускорение составило 21.5, минимальное – 3.09. Для большинства задач алгоритм с селекцией обучающих примеров показывал результаты не хуже стандартного алгоритма уже при 10-15% от выборки, таким

образом, при там качестве классификации, селекция обучающих примеров позволяет ускорить процесс обучения в 6-12 раз.

Таблица 4.24. Величины ускорения времени работы алгоритма

% выборки	5	10	15	20	25	30	Ориг.
Magic	13.51	9.11	6.15	4.83	3.47	3.09	1
Page-blocks	13.51	8.90	6.70	5.41	4.41	3.60	1
Penbased	14.04	9.47	6.39	5.02	3.61	3.20	1
Phoneme	21.50	11.39	8.72	6.05	4.96	4.16	1
Ring	16.13	10.47	6.83	5.57	4.22	3.46	1
Satimage	17.25	11.62	8.10	5.73	4.34	3.50	1
Segment	19.65	10.65	7.37	5.56	4.42	3.72	1
Texture	17.70	11.09	7.15	4.82	3.76	3.15	1
Twonorm	18.54	12.41	8.86	6.72	5.13	4.39	1

Для лучшего понимания процесса обучения, приведем примеры графиков изменения ошибки классификации лучшего индивида для обучающей подвыборки и для всей выборки. На рисунке 4.12 приведен график для задачи Penbased с периодом адаптации 50, величина подвыборки – 15%. На рисунке 4.13 представлен фрагмент процесса обучения для задачи Penbased с периодом адаптации 400 поколений, 15%.

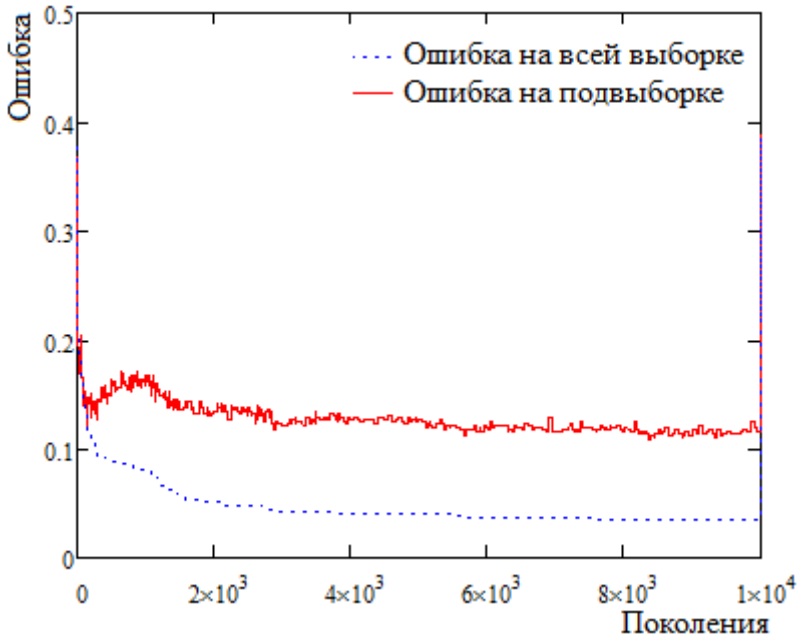


Рис. 4.12 Изменение точности в процессе обучения, задача Penbased, период адаптации 50 поколений

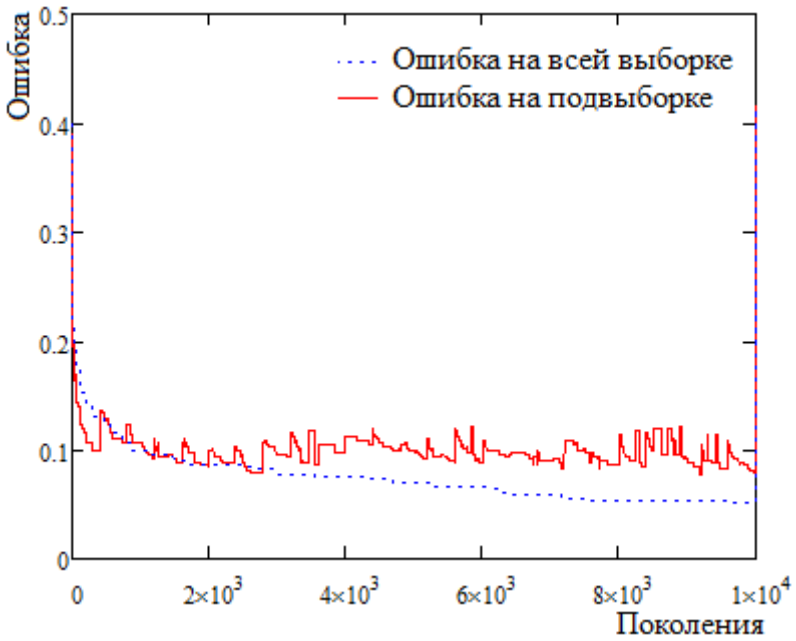


Рис. 4.13 Изменение точности в процессе обучения, задача Penbased, период адаптации 400 поколений

Характерной особенностью для метода активной селекции обучающих примеров является то, что величина ошибки на подвыборке не достигает величины ошибки на всей выборе по относительным величинам. Тем не менее, это не мешает успешному продолжению процесса обучения. Причиной такого поведения классификатора является то, что селекция обучающих примеров фокусируется на сложно классифицируемых объектах, которые, как правило, находятся на границе между классами. На начальных этапах обучения, то есть в первые несколько периодов адаптации, точность на подвыборке и на всей выборке идентична, что говорит о том, что измерения выбираются практически равновероятно. Однако далее ситуация меняется, легко классифицируемые измерения получают большие значения счетчиков и меньшие вероятности быть выбранными. Данные измерения реже попадают в подвыборку, однако остаются в изначальной выборке, а так как они классифицируются верно, получается, что ошибка на всей выборе в среднем ниже. При увеличении длины периода адаптации, процесс обучения происходит медленнее. На рисунке отчетливо видно, что на первом периоде адаптации ошибка на обучающей подвыборке становится ниже относительной ошибки на всей выборки. Это говорит о том, что происходит переобучение на подвыборку, то есть база правил начинает описывать

закономерности, присутствующие только в подвыборке. В течение нескольких следующих поколений ошибка на подвыборке и на всей выборке практически идентична, однако со временем вероятности изменяются таким образом, что алгоритм начинает фокусироваться на проблематичных областях пространства переменных. При этом разница между ошибкой на подвыборке и на всей выборки для более длинного периода адаптации меньше.

Рассмотрим подробнее процесс обучения в смысле различных мер качества классификации, описанных в пункте 4.1. Рассматриваться будет задача Page-blocks, так как она представляет наибольший интерес из-за большого дисбаланса по классам. Для этого для каждого поколения алгоритма рассчитывалась матрица ошибок на обучающей и тестовой выборке, далее рассчитывались меры Acc (общая точность классификации), $Precision_{\mu}$, $Recall_{\mu}$, $Fscore_{\mu}$, $Precision_{M}$, $Recall_{M}$, $Fscore_{M}$. Сначала приведем усредненные по всем прогонам графики всех мер для стандартного алгоритма на тестовой выборке на рисунке 4.14.

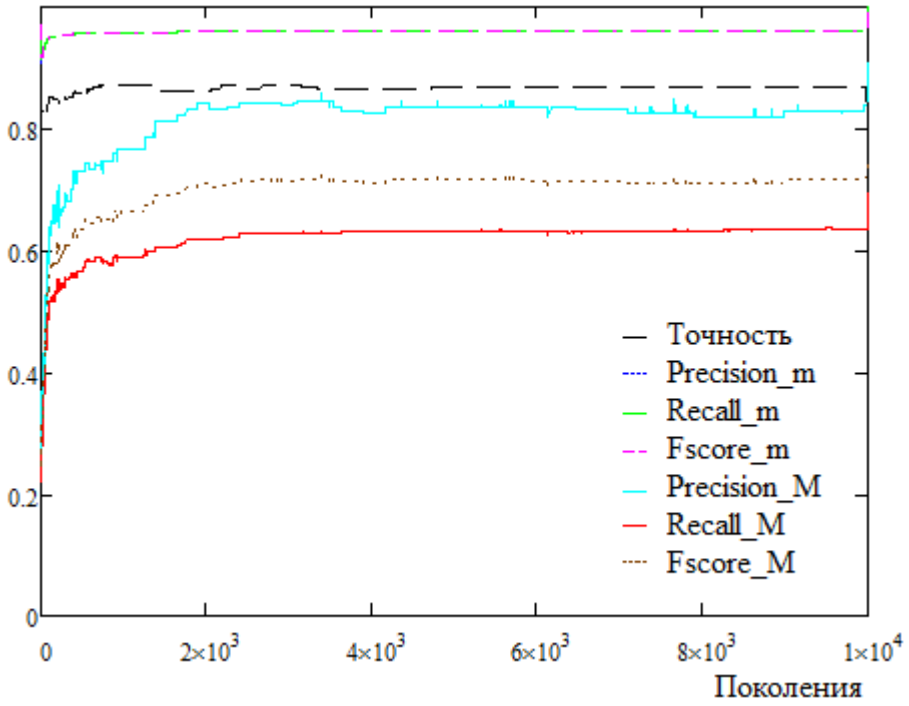


Рис. 4.14 Изменение мер качества классификации в процессе обучения, задача Page-blocks, оригинальный алгоритм

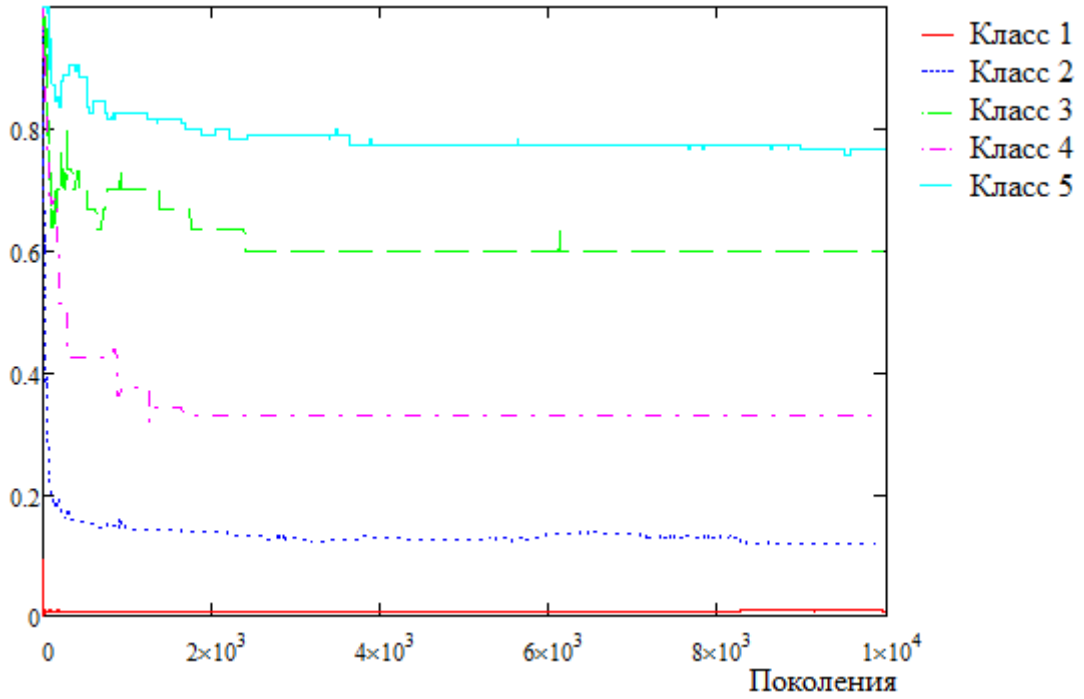


Рис. 4.15 Изменение ошибок классификации по разным классам, задача Page-blocks, оригинальный алгоритм

Для данной задачи меры $Precision_{\mu}$, $Recall_{\mu}$, $Fscore_{\mu}$ практически совпадают, однако меры $Precision_M$, $Recall_M$, $Fscore_M$ отличаются достаточно сильно. Это происходит из-за того что последние три сначала рассчитываются для каждого класса в отдельности, а затем усредняются по всем классам. В случае несбалансированной классификации, мера $Precision_M$ завышается и становится больше точности, в то время как $Recall_M$ становится ниже. На рисунке 4.15 приведены ошибки классификации по разным классам для оригинального алгоритма.

Как можно видеть, точно распознается только первый, самый многочисленный класс. Остальные же классы распознаются плохо (ошибка порядка 0.1-0.3), либо не распознаются вообще (ошибка 0.7-0.8). Следующие два рисунка показывают результаты работы для алгоритма с селекцией обучающих примеров и балансирующей стратегией, размер подвыборки – 5%, длина периода адаптации – 50 поколений.

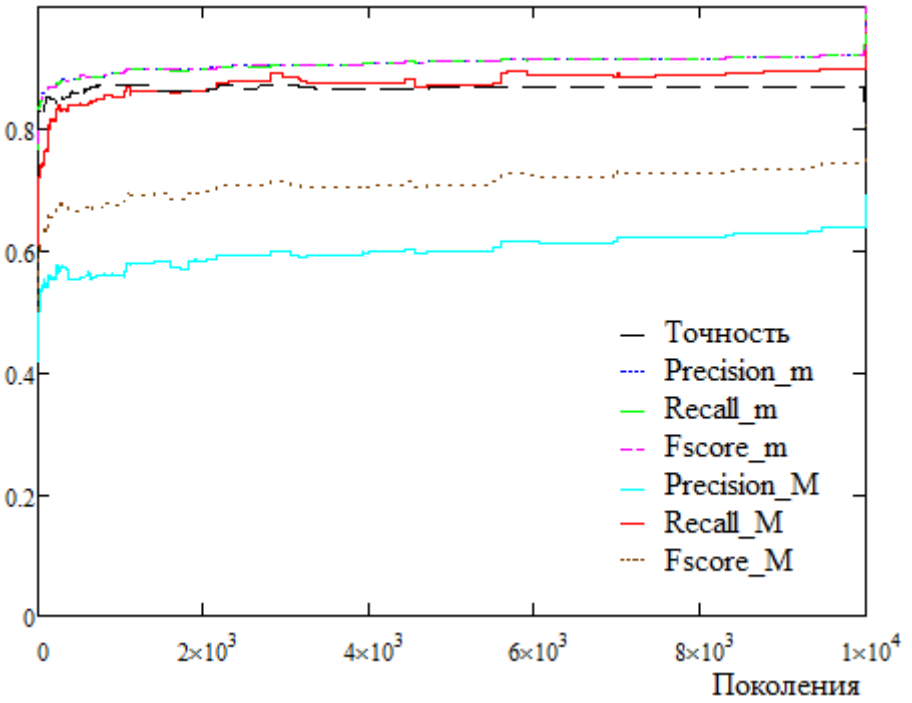


Рис. 4.16 Изменение мер качества классификации в процессе обучения, задача Page-blocks, оригинальный алгоритм

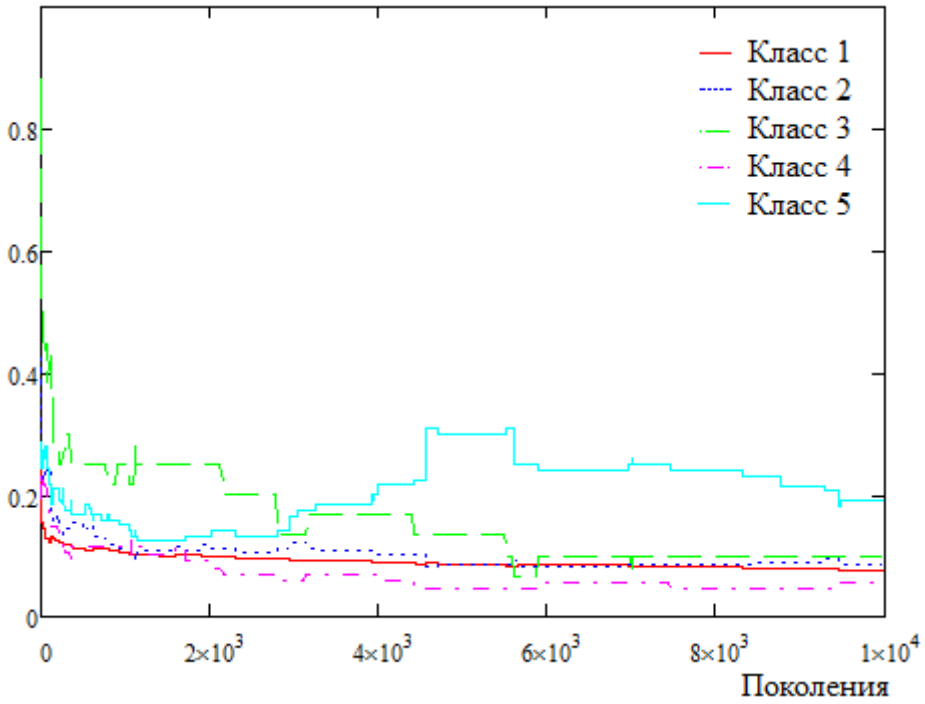


Рис. 4.17 Изменение ошибок классификации по разным классам, задача Page-blocks, оригинальный алгоритм

В сравнении с предыдущими двумя рисунками, можно заметить, что полученные решения стали более сбалансированными, особенно отчетливо это видно по рисунку 4.17. Точность на классах 1-4 стала практически идентичной,

хотя точность на пятом классе немного ниже. При этом значения $Fscore_M$ во втором случае выше, чем в первом. Приведенные результаты подтверждают повышение качества классификации при использовании процедуры селекции обучающих примеров с балансирующей стратегией для несбалансированных данных.

При использовании значений $Fscore_M$ для оценки качества классификации в ходе работы алгоритма в функции пригодности возможно дальнейшее повышение точности распознавания каждого класса. Для сравнения эффективности алгоритма при использовании точности классификации и при использовании значений $Fscore_M$ была проведена дополнительная серия экспериментов, с использованием селекции обучающих примеров. Число индивидов, как и раньше, устанавливалось равным 100, поколений – 10000, максимальное число правил 40. В таблице 4.25 приведено сравнение значений точности и $Fscore_M$. Жирным выделены лучшие значения каждой из мер в сравнении с алгоритмом с другой мерой в функции пригодности.

Таблица 4.25. Сравнение точности и меры $Fscore_M$.

Задача	В пригодности: Acc		В пригодности: $Fscore_M$	
	Acc	$Fscore_M$	Acc	$Fscore_M$
Magic	0.849	0.832	**0.854**	**0.837**
Page-blocks	**0.967**	**0.811**	0.962	0.807
Penbased	**0.961**	**0.963**	0.951	0.958
Phoneme	**0.831**	0.807	0.818	**0.812**
Ring	0.949	0.949	**0.955**	**0.956**
Satimage	**0.871**	0.846	0.867	**0.849**
Segment	**0.948**	**0.948**	0.942	0.944
Texture	**0.955**	**0.956**	0.946	0.948
Twonorm	0.952	0.952	**0.961**	**0.960**

Стоит отметить, что использование $Fscore_M$ снижает общую точность классификации, т.е. алгоритм с Ac-мерой в пригодности лучше в 6 случаях из 9, при этом алгоритм с $Fscore_M$ в функции пригодности лучше в смысле этой меры в 5 случаях из 9. Таким образом, можно заключить, что использование -меры действительно позволяет строить более сбалансированные классификаторы, кроме того, при необходимости оптимизации некоторой специфической меры,

будь то $Precision_M$, $Recall_M$, или что-либо другое, достаточно установить её в качестве основной меры оценки пригодности решений.

Для сравнения с другими методами на приведенном списке задач был также произведен дополнительный вычислительный эксперимент. Тестировался алгоритм без селекции обучающих примеров с увеличенным объемом ресурсов, а именно, число индивидов устанавливалось равным 210, число поколений – 50000, максимальное число правил не менялось и равнялось 40. Для сравнения были выбраны несколько алгоритмов классификации, использующих аппарат нечеткой логики. В их числе также и алгоритм, использованный в качестве прототипа в данной работе, т.е. две его реализации, однопоточный и параллельный гибридный нечеткий алгоритм эволюционный алгоритм машинного обучения.

Таблица 4.26. Сравнение с другими подходами

Задача	HEFCA _IS Баланс.	HEFCA оригин.	HEFCA увелич. ресурс	Fuzzy GBML [66]	Parallel Fuzzy GBML [66]	GP-Coach [31]	IVFS-Coop [103]	IVFS-Amp [102]	FARC-HD [25]	Bio HEL [28]
Magic	15.08	15.73	15.87	15.42	**14.89**	20.18	19.82	20.82	15.49	-
Page-blocks	**3.25**	3.96	3.78	3.81	3.62	8.77	6.57	5.84	4.99	-
Penbased	3.81	7.46	5.02	**3.07**	3.30	17.80	17.00	21.73	3.96	6.00
Phoneme	16.88	16.48	**15.36**	15.43	15.96	-	-	-	17.86	-
Ring	**5.08**	5.82	5.52	6.73	5.25	-	12.57	16.89	5.92	-
Satimage	12.93	14.22	12.86	15.54	12.96	27.50	-	-	12.68	**11.60**
Segment	5.19	6.45	5.62	5.99	5.90	24.04	-	-	-	**2.90**
Texture	**4.45**	7.75	4.90	4.64	4.77	-	-	-	7.11	-
Twonorm	4.81	6.06	5.57	7.36	**3.39**	15.17	-	-	4.72	-

Как можно видеть из таблицы, предложенный подход с балансирующей стратегией оказался лучшим на 3 из 9 задач (выделен жирным). При этом по точности модифицированный алгоритм с селекцией обучающих примеров превзошел не только оригинальный алгоритм с тем же числом индивидов и поколений, но и алгоритм с увеличенным в 10.5 раз ресурсом на 7 задачах из 9. Кроме того, удалось превзойти алгоритм прототип и его параллельную реализацию на 4 задачах из 9. На задаче Satimage результаты практически неотличимы. При этом параллельная реализация использовала 7 потоков при работе, и ей было выделено в 10.5 раз больше вычислительных ресурсов.

Что касается времени работы, то в таблице 4.27 представлено сравнение времени работы в минутах между предложенным подходом с балансирующей

стратегией и параллельным гибридным алгоритмом [66]. В среднем предложенный подход медленнее в 4.6 раза, однако параллельный гибридный алгоритм использовал 7 потоков в ходе работы, что позволяло ему работать до 7 раз быстрее. При параллельной реализации на 7 потоках самоконфигурирующийся гибридный алгоритм нечеткой классификации с селекцией обучающих примеров сможет превзойти алгоритм-прототип по времени работы. Стоит также отметить, что приведенное время работы характерно для лучших по общей точности классификации конфигураций, а так как для большинства задач максимальная точность была достигнута лишь при максимальном объеме подвыборки, можно заключить, что представленное время работы может быть снижено ещё больше без существенных потерь в качестве классификации.

Таблица 4.27. равнение времени работы, мин.

Задача	HEFCA_IS	Parallel Fuzzy GBML	Отношение
Magic	129.65 мин.	22.58 мин.	5.74
Page-blocks	18.53 мин.	4.74 мин.	3.91
Penbased	91.42 мин.	35.56 мин.	2.57
Phoneme	19.62 мин.	13.19 мин.	1.49
Ring	68.23 мин.	22.52 мин.	3.03
Satimage	85.12 мин.	15.38 мин.	5.53
Segment	32.40 мин.	4.69 мин.	6.91
Texture	114.79 мин.	15.72 мин.	7.30
Twonorm	38.11 мин.	7.84 мин.	4.86

При всех приведенных условиях и учитывая тот факт, что предложенный алгоритм использовал меньшее максимальное число правил (40 против 60 у прототипа), с учетом паритета данных методов по точности классификации, продемонстрированном в таблице 4.26, а также принимая во внимание способность качественно решать задачи с несбалансированными данными, следует сделать заключение о превосходстве разработанного самоконфигурирующегося гибридного эволюционного алгоритма формирования нечетких баз правил с селекцией обучающих примеров по балансирующей

стратегии над алгоритмом-прототипом и другими существующими методами нечеткой классификации.

ВЫВОДЫ

В четвертой главе были подробно рассмотрены задач классификации с несбалансированными данными, а также методы селекции обучающих примеров для снижения времени работы алгоритма и повышения точности построенных классификаторов.

Предложенная модификация процедуры определения наиболее подходящего номера класса позволила получать классификаторы, которые лучше распознают мало представленные в выборке классы. Данный вывод был подтвержден с применением различных мер классификации, а также после анализа матриц ошибок.

Разработанный в четвертой главе адаптивный алгоритм селекции обучающих примеров для задач с несбалансированными данными был реализован и включен в модифицированную программную систему. Данный метод формирует подвыборки ограниченного объема на основании изначальной обучающей выборки с учетом качества классификации каждого конкретного измерения. Целью данного метода является снижение времени работы, а также повышение качества классификации. По результатам масштабного тестирования данного метода можно заключить, что предложенный подход позволяет существенно повысить качество классификации, причем не только в смысле общей точности классификации, но и в смысле других мер, чувствительных к несбалансированности данных. Применение балансирующей стратегии при формировании подвыборок позволяет строить равномерно точные на всех классах нечеткие модели, что является важным результатом для множества прикладных задач. При этом время работы алгоритма снижается существенно, позволяя превзойти алгоритм-прототип и приблизиться по скорости к многопоточной параллельной реализации. Метод селекции обучающих примеров может быть

также применен к другим итерационным, в том числе не-эволюционным алгоритмам классификации.

Применение описанных модификаций позволяет разработанному алгоритму успешно решать задачи классификации с большим числом измерений, переменных и множеством классов за приемлемое время.

ЗАКЛЮЧЕНИЕ

В итоге данной работы можно заключить, что поставленные задачи были решены, и цель работы была достигнута, а именно:

1) Обоснована необходимость разработки новых методов формирования нечетких баз правил для задачи классификации с помощью эволюционных алгоритмов.

2) Разработано две новые схемы формирования нечетких баз правил с помощью самоконфигурируемого генетического алгоритма глобальной оптимизации, использующие Питтсбургский подход.

3) Разработан новый самоконфигурируемый гибридный эволюционный алгоритм формирования нечетких баз правил использующий комбинацию Питтсбургского и Мичиганского подходов, и отличающийся от известных схемой организации Мичиганской части, набором применяемых эволюционных операторов и процедурой самоконфигурирования.

4) Проведено комплексное исследование эффективности гибридного эволюционного алгоритма в сравнении с алгоритмами, использующими только Питтсбургский подход и показано его превосходство в быстродействии и точности на большинстве задач за счет использования ряда эвристик.

5) Обосновано использование процедуры самоконфигурирования для гибридного эволюционного алгоритма формирования нечетких баз правил. Самоконфигурируемый алгоритм в среднем показывает результаты не хуже, чем стандартный алгоритм в среднем.

6) Обоснована целесообразность использования метода селекции обучающих примеров для формирования нечетких баз правил для задач с большим числом измерений и несбалансированными данными.

7) Разработана новая схема адаптивного вероятностного метода селекции обучающих примеров для эволюционных алгоритмов для задачи классификации. Разработанная схема применена к самоконфигурируемому гибридному эволюционному алгоритму, статистически доказано не только

существенное увеличение быстродействия, но также и повышение точности классификации.

8) Разработана новая схема селекции обучающих примеров с балансированием обучающей подвыборки. Разработанная схема применена к самоконфигурируемому гибридному эволюционному алгоритму, статистически доказано, что данная схема позволяет формировать нечеткие базы правил, демонстрирующие схожую точность, как на мажоритарных, так и на миноритарных классах.

9) Разработанные в ходе исследования программные системы, реализующие предложенные подходы, успешно применены для решения реальных практических задач.

Таким образом, в диссертации разработаны, исследованы и апробированы новый самокофигурируемый гибридный эволюционный алгоритм формирования баз нечетких правил и новый адаптивный метод селекции обучающих примеров с возможностью формирования сбалансированных подвыборок ограниченного объема для обучения классификаторов, что является существенным вкладом в теорию и практику обработки информации и интеллектуального анализа данных.

Список использованной литературы

1. Айвазян, С.А., Енюков, И.С., Мешалкин, Л.Д. Прикладная статистика: основы моделирования и первичная обработка данных. – М.: Финансы и статистика, 1983. – 487 с.
2. Ахмедова, Ш.А. Генерирование нейросетевых классификаторов кооперативными бионическими алгоритмами оптимизации. – Материалы XVIII Международной научной конференции «Решетневские чтения», посвященной 90-летию со дня рождения генерального конструктора ракетно-космических систем академика М. Ф. Решетнев, Том 2, – 2014. С. 224-226.
3. Банди, Б. Методы оптимизации. Вводный курс. – М.: Радио и связь. – 1988. – 128 с.
4. Боровиков, В.П. Популярное введение в программу Statistica. – М.: Компьютер-Пресс, 1998. – 267 с.
5. Вапник, В.Н., Червоненкис, А.Я. Теория распознавания образов. – М.: Наука, 1974. – 416 с.
6. Воронцов, К. В. Лекции по методу опорных векторов. – 2007. – 21 декабря
7. Семенкина М.Е. Самоадаптивные эволюционные алгоритмы проектирования информационных технологий интеллектуального анализа данных, – Искусственный интеллект и принятие решений. №1. – 2013. – С. 13-23.
8. Семенкина М.Е. Самоконфигурируемый генетический алгоритм с модифицированными операторами равномерного скрещивания. – Кибернетика и высокие технологии XXI века. Сб. тр. 13-й международной научно-технической конференции – Т. 1. – Воронеж: ИСА РАН, 2012. – С. 32 – 41.
9. Сергиенко Р.Б., Автоматизированное формирование нечетких классификаторов самонастраивающимися коэволюционными алгоритмами, дис. канд. техн. наук / Р. Б. Сергиенко. – Красноярск, 2010.

10. Сергиенко, А.Б, Галушин, П.В, Бухтояров, В.В., Сергиенко, Р.Б., Сопов, Е.А., Сопов, С.А., Генетический алгоритм. Стандарт. [Электронный ресурс]. – 2010. Режим доступа: http://www.harrix.org/main/project_standart_ga.php
11. Становов В. В. Применение самоконфигурируемого эволюционного алгоритма построения нечетких баз правил для решения задач классификации с несбалансированными данными. – Материалы XVIII междун. науч. конф. Решетневские чтения, (Красноярск, 11-14 ноября 2014 г.) . – Т. 2, С. 127-129.
12. Становов В. В., Семенкина О.Э. Самоконфигурирующийся гибридный эволюционный алгоритм формирования нечетких классификаторов с активным обучением для несбалансированных данных. – Вестник СибГАУ 2014. № 5(57). С. 128–135.
13. Становов В.В. Самоконфигурируемый эволюционный алгоритм формирования нечетких классификаторов для задач с несбалансированными классами. – Сборник материалов Всероссийской научно-практической конференции «Информационно-телекоммуникационные системы и технологии (ИТСИТ-2014)» (Кемерово, 16-17 октября 2014 г.). С. 427-428.
14. Становов В.В., Семенкин Е.С. Self-adjusted evolutionary algorithms based approach for automated design of fuzzy logic systems. – Вестник СибГАУ. – 2013. № 5 (51), С. 148-152.
15. Становов В.В., Семенкин Е.С. Самонастраивающийся эволюционный алгоритм проектирования баз нечетких правил для задачи классификации. – Системы управления и информационные технологии. – 2014. Т. 57. № 3. С. 30-35.
16. Становов В.В., Семенкин Е.С., Бежитский С.С. Гибридный эволюционный алгоритм формирования нечетких баз правил для задачи классификации // Теория и практика системного анализа. Труды III Всероссийской научной конференции с международным участием (ТПСА-2014). – 2014. - Том 2. - С. 115-122.

17. Aarts, E.H.L. Simulated Annealing: Theory and Applications / E.H.L. Aarts, P.J.M. van Laarhoven. – London: Kluwer. – 1987.
18. Abe S., Lan M. S. A method for fuzzy rules extraction directly from numerical data and its application to pattern classification. – IEEE Trans. on Fuzzy Systems. – 1995.
19. Agrawal R., Srikant R. Fast algorithms for mining association rules. – Proceedings of the 20th VLDB Conference Santiago, Chile. – 1994.
20. Akhmedova Sh., Semenkin E., Gasanova T., Minker W. Co-Operation of Biology Related Algorithms for Support Vector Machine Automated Design. – In proceedings of the International Conference on Engineering and Applied Sciences Optimization (OPT-i'2014), pp. 1831-1837.
21. Akhmedova, Sh., Semenkin, E. Co-Operation of Biology Related Algorithms Meta-Heuristic in ANN-Based Classifiers Design. – In proceedings of the World Congress on Computational Intelligence (WCCI'14), pp. 867-873. – 2014.
22. Akhmedova, Sh., Semenkin, E. Co-Operation of Biology Related Algorithms. – In proceedings of 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 2207-2214. – 2013.
23. Akhmedova, Sh., Semenkin, E. Data Mining Tools Design with Co-Operation of Biology Related Algorithms. – Advances in Swarm Intelligence, Lecture Notes in Computer Science 8794, Y. Tan et al. (Eds.), Part 1, pp. 499-506. – 2014.
24. Alcala R., Alcala-Fernandez J., Herrera F., Otero J. Genetic learning of accurate and compact fuzzy rule based systems based on the 2-tuples linguistic representation. – International Journal of Approximate Reasoning 44.– p. 45–64. – 2007.
25. Alcala-Fdez J., Alcala R., Herrera F., A fuzzy association rulebased classification model for high-dimensional problems with genetic rule selection and lateral tuning. – IEEE Trans. Fuzzy Syst., vol. 19, no. 5, pp. 857–872. – Oct. 2011.
26. Alcalá-Fdez J., Sánchez L., Garcia S., Jesus M. J., Ventura S., Garrell J.M., Otero J., Romero C., Bacardit J., Rivas V.M., Fernández J. C., Herrera F. KEEL: A

software tool to assess evolutionary algorithms for data mining problems. – Soft Comput., vol. 13, no. 3. – p. 307–318. – Feb. 2009.

27. Asuncion A., Newman D. UCI machine learning repository. – University of California, Irvine, School of Information and Computer Sciences [Электронный ресурс]. Режим доступа: http://www.ics.uci.edu/~mlearn/MLRepository.html – 2007
28. Bacardit J., Burke E. K., Krasnogor N., Improving the scalability of rule-based evolutionary learning. – Memetic Comput. J., vol. 1, no. 1, pp. 55–67, – Mar. 2009.
29. Barandela R., S´anchez J. S., Garc´ıa V., Rangel E., Strategies for learning in class imbalance problems. – Pattern Recog., vol. 36, no. 3, pp. 849–851, – 2003.
30. Batista G. E. A. P. A., Prati R. C., Monard M. C., A study of the behavior of several methods for balancing machine learning training data. – SIGKDD Expl. Newslett., vol. 6, pp. 20–29, – 2004.
31. Berlanga F. J., Rivera A. J., del Jesus M. J., Herrera F., GP-COACH: Genetic programming-based learning of compact and accurate fuzzy rulebased classification systems for high-dimensional problems. – Inf. Sci., vol. 180, no. 8, pp. 1183–1200. – Apr. 2010.
32. Bezdek J.C. Pattern Recognition with Fuzzy Objective Function Algorithms. – Kluwer, 1981.
33. Bhowan U. Genetic Programming for Classification with Unbalanced Data. – Victoria University of Wellington. – 2012.
34. Bodenhofer U., Herrera F. Ten Lectures on Genetic Fuzzy Systems. – Preprints of the International Summer School: Advanced Control – Fuzzy, Neural, Genetic. – Slovak Technical University, Bratislava. p. 1–69. – 1997.
35. Brameier M., Banzhaf W. Genetic Programming. – Proceedings Third European Conference, EuroGP'2000, Edinburgh. – April 2000.
36. Brighton H, Mellish C. Advances in instance selection for instance-based learning algorithms. – Data Min Knowl Discov 6(2):153–172. – 2002.

37. Cano J. R., Herrera F., Lozano M., Evolutionary Stratified Training Set Selection for Extracting Classification Rules with trade off Precision-Interpretability. – Data & Knowledge Engineering archive, Volume 60 Issue 1, pp. 90-108. – January, 2007.
38. Cano J.R., Herrera F, Lozano M. Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study. – IEEE Trans Evol Comput 7(6):561–575. – 2003.
39. Cano J.R., Herrera F., Lozano M. Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study. – IEEE Transactions on Evolutionary Computation, Vol. 7, No. 6. – December 2003.
40. Celebi M. E., Kingravi H. A., Uddin B., Iyatomi H., Aslandogan Y. A., Stoecker W. V., Moss R. H. A methodological approach to the classification of dermoscopy images. – Comput.Med. Imag. Grap., vol. 31, no. 6, pp. 362–373. – 2007.
41. Chawla N., Cieslak D., Hall L., Joshi A., Automatically countering imbalance and its empirical relationship to cost. – Data Min. Knowl. Discov., vol. 17, pp. 225–252. – 2008.
42. Chien-Hsing C, Bo-Han K, Fu C. The generalized condensed nearest neighbor rule as a data reduction method. – In: Proceedings of the 18th international conference on pattern recognition. IEEE Computer Society, Hong-Kong, pp 556–559. – 2006.
43. Choi J., Oh S., Pedrycz W. Structural and parametric design of fuzzy inference systems using hierarchical fair competition-based parallel genetic algorithms and information granulation. – International Journal of Approximate Reasoning 49., p. 631–648. – 2008.
44. Cieslak D., Chawla N., Striegel A., Combating imbalance in network intrusion datasets. – in IEEE Int. Conf. Granular Comput., pp. 732– 737. – 2006.
45. Cordon O., Herrera F., Hoffmann F., Magdalena L. Genetic Fuzzy Systems. Evolutionary tuning and learning of fuzzy knowledge bases. – Advances in Fuzzy Systems: Applications and Theory. – World Scientific. – 2001.

46. Deb K., Pratap A., Agarwal S., Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. – IEEE Trans. Evol. Comput., vol. 6, no. 2., p. 182–197. – Apr. 2002.
47. Fazzolari M., Alcalá R., Nojima Y., Ishibuchi H., Herrera F. A Review of the Application of Multi-Objective Evolutionary Fuzzy Systems: Current Status and Further Directions. – IEEE Transactions on Fuzzy Systems 21:1. – p. 45-65. – 2013.
48. Fern´andez A., Garc´ıa S., del Jesus M. J., Herrera F., A study of the behaviour of linguistic fuzzy-rule-based classification systems in the framework of imbalanced data-sets. – Fuzzy Sets Syst., vol. 159, no. 18, pp. 2378–2398. – 2008.
49. Fernández A., García S., Jesus M., Herrera F. A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets. – Fuzzy Sets and Systems 159., pp. 2378 – 2398. – 2008.
50. Fernández A., Jesus M., Herrera F. Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets. – International Journal of Approximate Reasoning 50., p. 561–577. – 2009.
51. Finck S., Hansen N., Ros R., Auger A. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. – Technical Report 2009/20, Research Center PPE. – 2009.
52. Galar M., Fern´andez A., Barrenechea E., Bustince H., Herrera F.. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. – IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews.
53. Garain U. Prototype reduction using an artificial immune model. – Pattern Anal Appl 11. pp. 353–363.
54. Goldberg D. Genetic Algorithms in Search, Optimization and Machine Learning. – MA: Addison-Wesley Professional. – 1989.
55. Hansen N. et al Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009. – Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference 2010, ACM. – 2010.

56. Hansen N., Auger A., Finck S., Ros R. Real-parameter black-box optimization benchmarking 2009: Experimental setup. – Technical Report RR-6828, INRIA. – 2009.
57. Hansen N., Finck S., Ros R., Auger A. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. – Technical Report RR-6829, INRIA. – 2009.
58. Hart P.E. The condensed nearest neighbor rule. – IEEE Trans Inf Theory 14:515–516. – 1968.
59. Herrera F., Magdalena L. Genetic Fuzzy Systems: A Tutorial. – Tatra Mountains Mathematical Publications Vol. 13., p. 93-121. – 1997.
60. Holland, J.H. Adaptation in Natural and Artificial Systems. – The University of Michigan Press, Ann Arbor. – 1975.
61. Huang Y.-M., Hung C.-M., Jiau H. C., Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem. – Nonlinear Anal. R. World Appl., vol. 7, no. 4, pp. 720–747. – 2006.
62. Ishibuchi H. et al. Hybridization of fuzzy GBML approaches for pattern classification problems. – IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics. – 2005.
63. Ishibuchi H. et al. Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. – Fuzzy Sets and Systems. – 1997.
64. Ishibuchi H. et al. Three-objective genetics-based machine learning for linguistic rule extraction. – Information Sciences. – 2001.
65. Ishibuchi H. et al.: Selecting fuzzy if-then rules for classification problems using genetic algorithms. – IEEE Trans. on Fuzzy Systems. – 1995.
66. Ishibuchi H., Mihara S., Nojima Y. Parallel Distributed Hybrid Fuzzy GBML Models With Rule Set Migration and Training Data Rotation – IEEE Transactions on fuzzy systems, Vol. 21, No. 2. – April 2013.

67. Ishibuchi H., Nakashima T. Effect of Rule Weights in Fuzzy Rule-Based Classification Systems. – IEEE Transactions on fuzzy systems, vol. 9, no. 4. – August 2001.

68. Ishibuchi H., Nojima Y. Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. – International Journal of Approximate Reasoning. – 2007.

69. Ishibuchi H., Nozaki K., Tanaka H. Distributed representation of fuzzy rules and its application to pattern classification. – Fuzzy Sets and Systems. – 1992.

70. Ishibuchi H., Yamamoto T. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. – Fuzzy Sets and Systems 141. – 2004. – 59–88.

71. Ishibuchi H., Yamamoto T. Rule weight specification in fuzzy rule-based classification systems. – IEEE Trans. on Fuzzy Systems, vol. 13, no. 4. p. 428-435. – August 2005.

72. Jang J.S.R. ANFIS: Adaptive-network-based fuzzy inference system. – IEEE Trans. on Systems, Man and Cybernetics. – 1993.

73. Karr C.L., Gentry E.J. Fuzzy control of pH using genetic algorithms. – IEEE Trans. on Fuzzy Systems. – 1993.

74. Kilic K., Uncu O¨zge Tu¨rksen I. B., Comparison of different strategies of utilizing fuzzy clustering in structure identification. – Inf. Sci., vol. 177, no. 23, pp. 5153–5162. –2007.

75. Kohavi R.. A study of cross-validation and bootstrap for accuracy estimation and model selection. – International Joint Conference on Artificial Intelligence (IJCAI). – 1995.

76. Koza J. Genetic Programming. – The MIT Press Cambridge, Massachusetts, London, England. – 1998.

77. Lachiche, N., & Flach, P. A. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. – In Proceedings of ICML'2003 pp. 416–423. – 2003.

78. Lin C.T., Lee C.S.G., Neural-network-based fuzzy logic control and decision system. – IEEE Trans. on Computers. – 1991.
79. Lin Y., Lee Y., Wahba G., Support vector machines for classification in nonstandard situations. – Mach. Learn., vol. 46, pp. 191–202. – 2002.
80. Ling C., Sheng V., Yang Q., Test strategies for cost-sensitive decision trees. – IEEE Trans.Knowl. Data Eng., vol. 18, no. 8, pp. 1055–1067. – 2006.
81. Liu B., Ma Y., Wong C., Improving an association rule based classifier, – in Principles of Data Mining and Knowledge Discovery (Lecture Notes in Computer Science Series 1910), D. Zighed, J. Komorowski, and J. Zytkow, Eds., pp. 293–317. – 2000.
82. Lu W.-Z., Wang D., Ground-level ozone prediction by support vector machine approach with a cost-sensitive classification scheme. – Sci. Total. Enviro., vol. 395, no. 2-3, pp. 109–116. – 2008.
83. Mamdani E.H., Assilian S. An experiment in linguistic synthesis with a fuzzy logic controller. – International Journal of Man-Machine Studies. – 1975.
84. Michalewicz Z. Quo Vadis, Evolutionary Computation? On a growing gap between theory and practice. – Advances in Computational Intelligence, IEEE World Congress on Computational Intelligence, WCCI 2012, Brisbane, Australia, pp. 98-121. – June 10-15, 2012.
85. Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs (3rd Edn.). – New York: Springer-Verlag. – 1996.
86. Miller J.F. Cartesian Genetic Programming. – Natural Computing Series. – Springer-Verlag Berlin Heidelberg. – 2011.
87. Mollineda R.A., Ferri F.J., Vidal E. An efficient prototype merging strategy for the condensed 1-NN rule through class-conditional hierarchical clustering. – Pattern Recognit 35:2771–2782. – 2002.
88. Napierała K., Stefanowski J., Wilk S., Learning from Imbalanced data in presence of noisy and borderline examples, – in Rough Sets Curr. Trends Comput, pp. 158–167. – 2010.

89. Niehaus, J., Banzhaf, W. Adaption of Operator Probabilities in Genetic Programming. – EuroGP 2001, LNCS 2038. – p. 325-336. – 2001.

90. Nordin P., Banzhaf W., Brameier M. Evolution of a World Model for a Miniature Robot using Genetic Programming. – Robotics and Autonomous Systems, 25, pp. 105-116. – 1998.

91. Olvera-López J. A., Ariel Carrasco-Ochoa J., Francisco Martínez-Trinidad J., Kittler J. A review of instance selection methods. – Artif Intell Rev 34:133–143. –2010. DOI 10.1007/s10462-010-9165-y.

92. Olvera-López J.A., Carrasco-Ochoa J.A., Martínez-Trinidad J.F. Object selection based on clustering and border objects. – In: Kurzynski M et al (eds) Computer recognition systems 2, ASC 45. Wroclaw, Poland, pp. 27–34. – 2007.

93. Olvera-López J.A., Carrasco-Ochoa J.A., Martínez-Trinidad J.F. Prototype selection via prototype relevance. – In:Ruiz-Shulcloper J., Kropatsch W.G. CIARP 2008,LNCS5197. Habana, Cuba, pp. 153–160. – 2008.

94. Paredes R., Vidal E. Weighting prototypes. A new editing approach. – In: Proceedings of the international conference on pattern recognition ICPR, vol. 2. pp 25–28. – 2000.

95. Peng X. King I., Robust BMPM training based on second-order cone programming and its application in medical diagnosis. – Neural Netw., vol. 21, no. 2–3, pp. 450–457. – 2008.

96. Pudil P., Ferri F.J., Novovicová J., Kittler J. Floating search methods for feature selection with nonmonotonic criterion functions. – In: Proceedings of the 12th international conference on pattern recognition. IEEE Computer Society Press, pp 279–283. – 1994.

97. Raicharoen T., Lursinsap C. A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm. – Pattern Recognit Lett 26(10): pp. 1554–1567. – 2005.

98. Riquelme J.C, Aguilar-Ruíz J.S., Toro M. Finding representative patterns with ordered projections. – Pattern Recognit 36: pp. 1009–1018. – 2003.

99. Ritter G.L., Woodruff H.B., Lowry S.R., Isenhour T.L. Analgorithm for a selective nearest neighbor decision rule. – IEEE Trans Inf Theory 21(6):665–669. – 1975.

100. Ruspini E.H. Numerical methods for fuzzy clustering. – Information Sciences. – 1970.

101. Sánchez L., Rosario Suárez M., Villar J.R., Couso I.. Mutual information-based feature selection and partition design in fuzzy rule-based classifiers from vague data. – International Journal of Approximate Reasoning 49 pp. 607–622. – 2008.

102. Sanz J. A., Fernandez A., Bustince H., Herrera F., Improving the performance of fuzzy rule-based classification systems with interval-valued fuzzy sets and genetic amplitude tuning. – Inf. Sci., vol. 180, no. 19, pp. 3674–3685. – Oct. 2010.

103. Sanz J., Fernandez A., Bustince H., Herrera F., A genetic tuning to improve the performance of fuzzy rule-based classification systems with interval-valued fuzzy sets: Degree of ignorance and lateral position. – Int. J. Approx. Reason., vol. 52, no. 6, pp. 751–766. – Sep. 2011.

104. Semenkin E., Semenkina M. Self-configuring genetic algorithm with modified uniform crossover operator. – Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7331 LNCS (PART 1). – p. 414-421. – 2012.

105. Semenkin E., Stanovov V. Fuzzy rule bases automated design with self-configuring evolutionary algorithm – Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO-2014), pp. 318-323. – 2014.

106. Simpson P.K. Fuzzy min-max neural networks. – IEEE Trans. on Neural Networks. – 1992.

107. Sokolova M., Lapalme G. A systematic analysis of performance measures for classification tasks. – Information Processing and Management 45, pp. 427–437. – 2009.

108. Stanovov V., Semenkin E. Hybrid self-configuring evolutionary algorithm for automated design of fuzzy logic rule base. – 11th international conference on Fuzzy Systems and Knowledge Discovery (FSKD, Xaimen, China), pp. 317-321. – 19-21 August 2014.
109. Stanovov V., Semenkin E., Semenkina O. Instance Selection Approach for Self-configuring Hybrid Fuzzy Evolutionary Algorithm for Imbalanced Datasets // ICSI-CCI 2015, Part I, LNCS 9140, pp. 451–459. – 2015.
110. Stanovov V., Semenkin E., Semenkina O. Self-configuring hybrid evolutionary algorithm for fuzzy classification with active learning // IEEE Congress on evolutionary computation (CEC 2015, Japan). – 2015.
111. Stanovov V., Semenkina O. Self-configuring hybrid evolutionary algorithm for multi-class unbalanced datasets – Вестник СибГАУ. Т. 16, № 1. С. 131–136. – 2015.
112. Stefanowski J. Wilk S., Selective pre-processing of imbalanced data for improving classification performance. – in Data Warehousing and Knowledge Discovery (Lecture Notes in Computer Science Series 5182), I.-Y. Song, J. Eder, and T. Nguyen, Eds., pp. 283–292. – 2008.
113. Takagi T., Sugeno M. Fuzzy identification of systems and its applications to modeling and control. – IEEE Trans. on Systems, Man, and Cybernetics. – 1985.
114. Takagi T., Sugeno M., Fuzzy Identification of Systems and Its Application to Modeling and Control. – IEEE Transactions on Systems, Man and Cybernetics, Vol. 15, pp. 116-132. – 1985.
115. Turner A.J., Miller J.F. Cartesian Genetic Programming encoded Artificial Neural Networks: A Comparison using Three Benchmarks. – GECCO'13, Amsterdam, The Netherlands. – July 6–10, 2013.
116. Venmann C.J., Reinders M.J.T. The nearest sub-class classifier: a compromise between the nearest mean and nearest neighbor classifier. – IEEE Trans Pattern Anal Mach Intell 27(9): pp. 1417–1429. – 2005.

117. Wang L.X., Mendel J.M. Generating fuzzy rules by learning from examples. – IEEE Transactions on Systems, Man, and Cybernetics 22:6. – pp. 1414-1427. – 1992
118. Wilson D.L. Asymptotic properties of nearest neighbor rules using edited data. – IEEE Trans Syst Man Cybern 2:408–421. –1972.
119. Wilson D.R., Martínez T.R. Reduction techniques for instance-based learning algorithms. – Mach Learn 38:257–286. – 2000.
120. Wolpert, D. The Lack of A Priori Distinctions between Learning Algorithms. – Neural Computation, pp. 1341–1390. – 1996.
121. Wolpert, D.H., Macready, W.G. Coevolutionary free lunches. – IEEE Transactions on Evolutionary Computation, 9(6): pp. 721–735. – 2005.
122. Wolpert, D.H., Macready, W.G., No Free Lunch Theorems for Optimization, – IEEE Transactions on Evolutionary Computation 1, 67. – 1997.
123. Zadeh L.A. The concept of a linguistic variable and its application to approximate reasoning. – I. Information Sciences. –1975.
124. Zadeh, L.A. Fuzzy sets, Information and Control 8 (3)., p. 338–353. – 1965.
125. Zhang S., Liu L., Zhu X., Zhang C., A strategy for attributes selection in cost-sensitive decision trees induction, in Proc. IEEE 8th Int. Conf. Comput. Inf. Technol. Workshops, pp. 8–13. – 2008
126. Zitzler E., Laumanns M., Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm. – Computer Engineering and Networks Laboratory (TIK). – Swiss Federal Institute of Technology (ETH) Zurich. – 2001.
127. Zitzler E., Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. – IEEE Transactions on Evolutionary Computation 3(4). – p. 257–271. – 1999.

Список публикаций автора

Статьи в ведущих рецензируемых научных журналах и изданиях

1. Становов В.В., Бежитский С.С., Бежитская Е.А., Семенкин Е.С. Исследование эффективности многоагентного алгоритма решения задач глобальной поисковой оптимизации большой размерности // Системы управления и информационные технологии, № 4(62). – 2015.
2. Становов В.В., Бежитский С.С., Бежитская Е.А., Попов Е.А. Многоагентный алгоритм проектирования баз нечетких правил для задачи классификации // Вестник СибГАУ, Т. 16, №4, С. 842-848. – 2015.
3. Stanovov V., Semenkina O. Self-configuring hybrid evolutionary algorithm for multi-class unbalanced datasets // Вестник СибГАУ. 2015. Т. 16, № 1. С. 131–136.
4. Stanovov V., Skraba A., Kofiac D., Znidarsic A., Maletic M. Rozman C., Semenkin E., Semenkina M. Application of Self-Gonfiguring Genetic Algorithm for Human Resource Management // Journal of Siberian Federal University. Mathematics and Physics 2015, 8(1), 98-107.
5. Становов В. В., Семенкина О.Э. Самоконфигурирующийся гибридный эволюционный алгоритм формирования нечетких классификаторов с активным обучением для несбалансированных данных // Вестник СибГАУ 2014. № 5(57). С. 128–135.
6. Становов В.В., Семенкин Е.С. Самонастраивающийся эволюционный алгоритм проектирования баз нечетких правил для задачи классификации // Системы управления и информационные технологии. 2014. Т. 57. № 3. С. 30-35.
7. Становов В.В., Семенкин Е.С. Self-adjusted evolutionary algorithms based approach for automated design of fuzzy logic systems // Вестник СибГАУ. 2013. № 5 (51), С. 148-152.
8. Шкраба А., Становов В.В., Жнидаршич А., Розман Ч., Кофьяч Д. – Рассмотрение стратегии оптимального управления строго иерархической

системы управления человеческими ресурсами // Вестник СибГАУ, Т. 17, №1, С. 97-102. – 2016.

Публикации в изданиях, индексируемых в международных базах

9. Stanovov V., Semenkin E., Semenkina O. Instance selection approach for self-configuring evolutionary fuzzy rule based classification systems // 4th International Congress on Advanced Applied Informatics July 12-16, 2015, Okayama Convention Center, Okayama, Japan. (Scopus).
10. Škraba A., Semenkin E., Kofjac D., Semenkina M., Znidaršic A., Maletic M., Akhmedova Sh., Rozman C., Stanovov V. Modelling and Optimization of Strictly Hierarchical Manpower System. 12th International Conference on Informatics in Control, Automation and Robotics, ICINCO. – 2015. (Web of Science, Scopus).
11. Stanovov V., Semenkin E., Semenkina O. Instance Selection Approach for Self-configuring Hybrid Fuzzy Evolutionary Algorithm for Imbalanced Datasets // Advances in Swarm and Computational Intelligence, LNCS 9140, 2015, pp. 451–459. (Web of Science).
12. Stanovov V., Semenkin E., Semenkina O. Self-configuring hybrid evolutionary algorithm for fuzzy classification with active learning // 2015 IEEE Congress on Evolutionary Computation (CEC’2015, Japan). (Scopus).
13. Semenkin E., Stanovov V. Fuzzy rule bases automated design with self-configuring evolutionary algorithm // Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO’2014, Austria), pp. 318-323. (Scopus).
14. Stanovov V., Semenkin E. Hybrid self-configuring evolutionary algorithm for automated design of fuzzy logic rule base // 11th international conference on Fuzzy Systems and Knowledge Discovery (FSKD’2014, China), pp. 317-321. (Scopus, Web of Science).
15. Akhmedova Sh., Semenkin E., Stanovov V., Fuzzy Rule-based Classifier Design with Co-Operative Bionic Algorithm for Opinion Mining Problems // 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2016, Lisbon, Portugal). – 2016. (Scopus).

16. Akhmedova Sh., Stanovov V., Semenkin E., Fuzzy Rule-Based Classifier Design with Co-operation of Biology Related Algorithms // Advances in Swarm Intelligence, pp.198-205. – 2016. (Scopus).
17. Stanovov V., Sopov E., Semenkin E. Multi-Strategy Multimodal Genetic Algorithm for Designing Fuzzy Rule Based Classifiers // IEEE Symposium Series on Computational Intelligence (SSCI 2015, South Africa). – 8-10 December. – 2015. (Web of Science, Scopus).

Публикации в зарубежных журналах

18. Stanovov V., Semenkin E., Semenkina O., Self-Configuring Hybrid Evolutionary Algorithm for Fuzzy Imbalanced Classification with Adaptive Instance Selection., Journal of Artificial Intelligence and Soft Computing Research 6(3), June 2016, pp. 173-188. – 2016.

Публикации в сборниках трудов конференций

19. Становов В. В. Применение самоконфигурируемого эволюционного алгоритма построения нечетких баз правил для решения задач классификации с несбалансированными данными. – Материалы XVIII междун. науч. конф. Решетневские чтения, (Красноярск, 11-14 ноября 2014 г.). – Т. 2, С. 127-129. – 2014.
20. Становов В. В. Решение задачи определения уровня озона в атмосфере при помощи самоконфигурирующегося эволюцинного алгоритма построения нечетких баз правил. – Материалы XVIII междун. науч. конф. Решетневские чтения, (Красноярск, 11-14 ноября 2014 г.). – Т. 2, С. 355-357. – 2014.
21. Становов В.В., Семенкин Е.С., Бежитский С.С. Гибридный эволюционный алгоритм формирования нечетких баз правил для задачи классификации. – Теория и практика системного анализа. Труды III Всероссийской научной конференции с международным участием (ТПСА-2014). Том 2. - С. 115-122. – 2014.
22. Skraba A., Semenkin E., Semenkina M, Stanovov V.V., Kofiac D., Znidarsic A., Rozman C., Maletic M. Development of discrete manpower model and determination of optimal strategies. – Решетневские чтения – материалы XVIII

Междунар. науч. конф., (11-14 нояб. 2014, г. Красноярск): в 3 ч. / под общ. ред. Ю.Ю. Логинова; Сиб. гос. аэрокосмич. ун-т. – Красноярск, 2014. Ч. 2. – С. 421-423. – 2014.

23. Становов В.В., Бежитский С.С. Самоконфигурирующийся гибридный эволюционный алгоритм формирования баз нечетких правил для задач с несбалансированными данными в многоагентных системах. – Материалы Восьмого Всероссийского форума студентов, аспирантов и молодых ученых (Санкт-Петербург, 27-29 октября 2014 г.). С. 57-59. – 2014.
24. Становов В.В. Самоконфигурируемый эволюционный алгоритм формирования нечетких классификаторов для задач с несбалансированными классами. – Сборник материалов Всероссийской научно-практической конференции «Информационно-телекоммуникационные системы и технологии (ИТСИТ-2014)» (Кемерово, 16-17 октября 2014 г.). С. 427-428. – 2014.
25. Сравнение методов самонастройки генетического алгоритма. Молодежь и наука: сборник материалов X Юбилейной Всероссийской научно-технической конференции студентов, аспирантов и молодых ученых с международным участием, [Электронный ресурс]. – Красноярск: Сиб. федер. ун-т. – Секция «Математика, информатика: моделирование и оптимизация сложных систем». Режим доступа: http://conf.sfu-kras.ru/sites/mn2014/ sect?sec_id=1029. – 2014.
26. Stanovov V.V., Semenkin E.S. Self-adjusted evolutionary algorithms for design of fuzzy logic systems. – Proceedings of the 2nd International Workshop on Mathematical Models and its Applications (IWMMA). – 2013.
27. Становов В. В., Семенкин Е.С. Особенности генерации случайных чисел при распараллеливании эволюционных алгоритмов. – Системный анализ и интеллектуальные технологии. Труды V Международной конференции (САИТ-2013). - Том 2. - С. 411-418. – 2013.

28. Становов В.В. О распараллеливании генетического алгоритма. Интеллект и наука: труды XIII Междунар. науч. конференции. – Железногорск: Филиал СФУ. – С. 122-124. – 2013.
29. Становов В.В. Исследование эффективности распараллеливания алгоритма генетического программирования для задачи формирования систем на нечеткой логике. Актуальные проблемы авиации и космонавтики: материалы IX Всерос. науч.-практ. конф. творческой молодежи, посвященной Дню космонавтики. – Красноярск: СибГАУ. – 2013.
30. Становов В.В. О распараллеливании алгоритма генетического программирования для задачи символьной регрессии. – Молодежь и наука: сборник материалов IX Всероссийской научно-технической конференции студентов, аспирантов и молодых ученых с международным участием, посвященной 385-летию со дня основания г. Красноярска [Электронный ресурс]. – Красноярск: Сиб. федер. ун-т. – Секция «Математика, информатика: моделирование и оптимизация сложных систем» Режим доступа: http://conf.sfu-kras.ru/sites/mn2013/ section065.html. – 2013.
31. Становов В.В. Применение алгоритма генетического программирования для проектирования систем на нечеткой логике. – Сборник научных работ всероссийского конкурса научно-исследовательских работ студентов и аспирантов в области информатики и информационных технологий. – Белгород: БелГУ. – Том. 3. - С. 553-559. – 2012.
32. Исследование эффективности алгоритма генетического программирования для проектирования систем на нечеткой логике. – «Решетневские чтения». Материалы XVI Международной научной конференции, посвященной памяти генерального конструктора ракетно-космических систем академика М. Ф. Решетнева. - Красноярск: СибГАУ. - Том 2., С. 517-518. – 2012.
33. Становов В.В. Исследование эффективности алгоритма генетического программирования с настройкой коэффициентов в задаче символьной регрессии. – Молодежь и наука: сборник материалов VIII Всероссийской научно-технической конференции студентов, аспирантов и молодых

ученых, посвященной 155-летию со дня рождения К.Э. Циолковского. – 2012.

34. Становов В.В. Исследование эффективности различных методов самонастройки алгоритма генетического программирования для задачи символьной регрессии. – Молодежь и наука: сборник материалов VIII Всероссийской научно-технической конференции студентов, аспирантов и молодых ученых, посвященной 155-летию со дня рождения К.Э. Циолковского. – 2012.
35. Становов В.В. Исследование эффективности различных методов самонастройки генетического алгоритма. – «Актуальные проблемы авиации и космонавтики». Труды Всероссийской научно-практической конференции творческой молодежи. - Красноярск: СибГАУ, – 2012.

Зарегистрированные программные системы

36. Становов В. В., Панфилов И.А., Сопов Е.А. Распределенная самонастраивающаяся система формирования нечетких баз правил при помощи генетического алгоритма. Свидетельство №2014610096 о гос. регистрации в Реестре программ для ЭВМ от 03.03.2014.
37. Становов В. В., Семенкин Е. С. Самонастраивающаяся система формирования символьных выражений для решения задач классификации и регрессии методом генетического программирования. Свидетельство № 2013619070 о гос. регистрации в Реестре программ для ЭВМ от 25.09.2013.
38. Становов В. В., Сергиенко Р. Б. Распределенная программная система автоматического формирования нечетких систем методом генетического программирования. Свидетельство № 2013611035 о гос. регистрации в Реестре программ для ЭВМ от 9.01.2013.