Федеральный исследовательский центр "Информатика и управление"
Российской академии наук

На правах рукописи

Девяткин Дмитрий Алексеевич

Построение ансамблей деревьев решений с использованием линейных и нелинейных разделителей

Специальность 2.3.5 (05.13.11) —
«Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
к.ф-м.н
Соченков Илья Владимирович

Москва — 2022

Оглавление

Стр.

Стр.

Стр.

Введение

Методы на основе дифференцируемых функций, такие как нейронные сети, широко применяются для решения задач анализа структурированных данных, текстов и изображений. В некоторых случаях применимость этих методов ограничена: например, в исследовании A. Ek эмпирически показано, что незначительные изменения в тексте могут приводить к существенным изменениям его тональности и смысла, которые не всегда выявляются многослойными нейронными сетями, в том числе с архитектурой «Трансформер» [1]. В работах M. Tanchik и др., Z. Khan и др. [2] на примерах восстановления изображений МРТ и оценки вегетационных индексов по данным аэрофотосьемки показано, что многослойные нейронные сети не позволяют решать задачи регрессионного анализа с приемлемой точностью, если моделируемая зависимость имеет негладкий характер. В подобных задачах существенное значение имеет дискретная природа составляющих частей анализируемых объектов в контексте применения методов машинного обучения. Между тем, решение этих задач имеет большую практическую значимость в различных отраслях экономики, позволяя снизить стоимость, либо автоматизировать отдельные этапы технологических процессов. Большей точности решения в таких случаях можно было бы достигнуть при применении моделей машинного обучения с дискретными зависимыми переменными. В основе таких моделей могут лежать деревья решений и их композиции (ансамбли). Так, в работах Z. Zhou, J. Ren, Y. Chen, Л. Уткина [3], предложены каскадные композиции случайных лесов деревьев решений, на некоторых задачах превосходящие по качеству анализа методы на основе нейронных сетей. Однако деревья решений имеют ограниченную выразительную способность при фиксированной высоте (количество учитываемых признаков ограничивается числом узлов дерева). Они также характеризуются низкой вычислительной эффективностью при анализе данных большой размерности. Одним из подходов к решению этой проблемы является обучение деревьев решений с многомерными разделителями в узлах, например, с наклонными гиперплоскостями (oblique trees). Исследования S. Murphy, B. Menze [4] и других показали, что подобные деревья решений имеют низкую обобщающую способность (то есть способность формировать корректные результаты для объектов, не использовавшихся при обучении), поэтому они применимы

только в составе рандомизированных композиций, построенных методами бэггинга, со-ббэггинга или случайного леса. Кроме того, большинство подходов к построению таких деревьев имеют низкую вычислительную эффективность и большое количество гиперпараметров. При использовании таких композиций также могут наблюдаться эффекты, связанные с переобучением, поэтому необходимо использовать методы регуляризации, позволяющие найти баланс между сложностью получаемых алгоритмов и их обобщающей способностью. Время обучения композиций деревьев решений с линейными или нелинейными разделителями на данных большого объема существенно превосходит время построения ансамблей деревьев решений с одномерными разделителями. Кроме того, для построения различных видов разделителей требуются разные типы вычислительных ресурсов. Поэтому актуальной становится задача разработки специализированных распределенных архитектур систем построения композиций деревьев решений с многомерными разделителями.

Целью работы является повышение качества (полноты и точности) решения задачи классификации на основе лесов деревьев решений путем создания вычислительно-эффективного, метода построения случайных ансамблей деревьев решений с применением линейных и нелинейных разделителей.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. Разработать и реализовать метод построения деревьев решений применением линейных и нелинейных разделителей.
2. Развить теоретические основы для регуляризации случайных ансамблей деревьев решений: разработать методы оценки обобщающей способности случайных ансамблей деревьев решений.
3. Разработать методы классификации объектов сложной структуры (изображения, тексты) на основе исследованных деревьев решений с линейными и нелинейными разделителями.
4. Провести экспериментальное исследование разработанного метода, сравнить его с другими методами, в том числе на задачах обработки изображений.
5. Разработать модель (архитектуру) для организации глобально распределенного обучения случайных лесов деревьев решений с линейными и нелинейными разделителями.

6. Разработать комплекс программ для обучения случайных лесов деревьев решений с линейными и нелинейными разделителями.

Основные положения, выносимые на защиту:

1. Вычислительно-эффективный (линейная сложность) метод построения узлов деревьев решений с применением линейных и нелинейных разделителей, при обучении которых совместно оптимизируется отступ между разделяемыми объектами и произвольный критерий однородности данных.
2. Оценка обобщающей способности случайных ансамблей деревьев решений, учитывающая основные гиперпараметры алгоритмов их построения.
3. Метод классификации объектов, характеризующихся наличием связей между признаками.
4. Архитектура программного обеспечения глобально распределенного обучения случайных лесов деревьев решений с линейными и нелинейными разделителями.
5. Комплекс программ для обучения случайных лесов деревьев решений с линейными и нелинейными разделителями.

Перечисленные положения относятся к направлениям исследований 4, 7, 8, и 9 паспорта специальности 2.3.5.

Научная новизна: Разработан оригинальный вычислительно-эффективный метод построения узлов деревьев решений с применением линейных и нелинейных разделителей, при обучении которых совместно оптимизируется отступ между разделяемыми объектами и произвольный критерий однородности. Этот метод применен для обучения деревьев решений в составе случайных лесов. Выполнено развитие теоретических подходов к подбору методов регуляризации случайных ансамблей деревьев решений:

1. Теоретически обоснована связь между равномерной стабильностью1 алгоритмов обучения и формируемой структурой деревьев решений.
2. Предложена оценка обобщающей способности случайных ансамблей деревьев решений, учитывающая основные гиперпараметры алгоритмов их построения.
3. Предложена архитектура программного обеспечения глобально распределенного обучения случайных лесов деревьев решений с линейными и нелинейными разделителями.

Теоретическая и практическая значимость Теоретическая значимость состоит в развитии формальных подходов к исследованию обобщающей способности случайных ансамблей деревьев решений. Эти подходы могут быть применены для разработки или подбора методов регуляризации случайных ансамблей деревьев решений.

Практическая значимость состоит в повышении точности и полноты решения задач анализа данных с применением ансамблей деревьев решений (показано повышение точности на 8% на наборе данных Cifar-10, и на 2% на наборах Letter и USPS). Результаты работы использованы при реализации проектов № 075-15-2020-799, № 075-15-2020-805 поддержанных Министерством образования и науки Российской Федерации, проекта 19-29-07163мк, поддержанного Российским фондом фундаментальных исследований. Предложенные методы использовались для решения задач анализа психолингвистических характеристик сообщений в социальных сетях, выявления когнитивных операций в научных текстах, анализа нормативно-правовых документов, а также задач оценки вегетационных индексов по цветным изображениям.

Апробация работы. Основные результаты работы были представлены на следующих конференциях и семинарах:

- XI Международная научно-практическая конференция «Интегрированные модели и мягкие вычисления в искусственном интеллекте» (ИММВ-2022, Коломна, 16-19 мая 2022 г.
- Научный семинар "Математические модели информационных технологий"департамента анализа данных и искусственного интеллекта "Интеллектуальные системы и структурный анализ"НИУ ВШЭ 25.03.2021, Москва.
- Научный семинар по системному программированию под руководством академика РАН А.И. Аветисяна по теме «Построение рандомизированных ансамблей деревьев решений с использованием ядерных разделителей» , 16.06.2022, Москва.
- Научный семинар кафедры информационных технологий РУДН «Построение рандомизированных ансамблей деревьев решений с использованием ядерных разделителей», 17.06.2022, Москва.

Личный вклад. Все выносимые на защиту результаты получены лично автором.

Публикации. Основные результаты по теме диссертации изложены в шести работах, опубликованных в изданиях, рекомендованных ВАК или приравненных к ним (Scopus/ Web Of Science), а также представлены в форме зарегистрированной программы для ЭВМ. В статьях [5—7] вместе с соавтором была поставлена задача и проводилась редакторская правка, остальная часть выполнена соискателем. Работы [8—11] полностью выполнены автором.

Объем и структура работы. Диссертация состоит из введения, трёх глав и заключения. Полный объем диссертации составляет 115 страниц с 26 рисунками и 8 таблицами. Список литературы содержит 114 наименований.

Глава 1. Задачи и методы классификации

В настоящей работе будут использоваться следующие определения и понятия. Под алгоритмом классификации понимается алгоритм, возвращающий для заданного произвольного объекта соответствующую ему метку. Алгоритмом обучения называется процедура построения алгоритма классификации на основе заданной обучающей выборки.

В работе рассматриваются алгоритмы классификации – бинарные деревья решений, в каждом узле которых с применением разделителя происходит распределение анализируемых объектов по двум подмножествам, соответствующим левому и правому поддеревьям. Разделитель может представлять собой простое решающее правило, в котором значение определенного признака сравнивается с заданным порогом, а может являться более сложным линейным, либо нелинейным алгоритмом классификации объектов по поддеревьям. При обучении деревьев решений, как правило, оптимизируется некоторый критерий неоднородности данных, такой как неоднородность Джини (Gini impurity), информационная энтропия и другие. Под ансамблем понимается линейная композиция деревьев решений, под случайным ансамблем понимается композиция, отдельные алгоритмы которой обучены на случайных подмножествах обучающих данных (бэггинг) или случайных подмножествах данных и признаков (случайный лес).

1.1 Классификация. Оценки обобщающей способности алгоритмов классификации

1.1.1 Задача классификации в общем виде

Одной из задач, решаемых с помощью методов машинного обучения с учителем, является классификация [12]. Пусть задано множество признаковых описаний объектов $X \subseteq \mathbb{R}^n, n \in \mathbb{N}$, конечное множество меток классов объектов Y, а также существует некоторая неизвестная целевая зависимость $c : X \to Y$,

с помощью которой построен обучающий набор данных $D_t \sim c$, состоящий из t элементов. Необходимо на основе множества D_t построить алгоритм h из некоторого семейства H, $h : X \to Y$, который для любого заданного объекта из X возвращал бы метку класса из Y в соответствии с c. Пусть $L : Y^2 \to 0 \cup \mathbb{R}^+$ - функция потерь, такая что,

$$L(y,y') = \begin{cases} r \in \mathbb{R}^+, y \neq y' \\ 0, y = y' \end{cases} \quad ,\forall y,y' \in Y. \tag{1.1}$$

Необходимо выбрать такой алгоритм $h \in H$, который для заданного вектора признаков x возвращал бы соответствующую ему метку класса y и минимизировал при этом функционал риска

$$R(h) = \int_X L(c(x), h(x))dx. \tag{1.2}$$

Так как c - неизвестно, на практике оценивается эмпирический риск:

$$\hat{R}(h) = \frac{1}{T}\sum_{i=1}^{t} L(y_i, h(x_i)) \tag{1.3}$$

на обучающей выборке D_t. Оценка разности между риском и его эмпирической оценкой 1.4 в зависимости от сложности алгоритмов из H и мощности обучающей выборки D_t является предметом исследования работ, посвященных статистической теории обучения [12, 13]:

$$\Delta(h) = sup_{h \in H}(R(h) - \hat{R}(h)) \tag{1.4}$$

Вариантами задачи классификации являются:

1. Задача многоклассовой классификации - это задача обучения по прецедентам, в которой каждый анализируемый объект сопоставлен с одной меткой класса из конечного множества, причем мощность этого множества классов больше двух.
2. Задача многозначной классификации - это задача обучения по прецедентам, в которой каждый анализируемый объект сопоставлен сразу с несколькими метками классов из конечного множества.

Заметим, что целевая зависимость c каждому элементу множества X однозначно ставит в соответствие элемент множества Y, причем никакие другие элементы Y или X не влияют на результат. Однако, существуют задачи классификации, которые не могут быть корректно поставлены подобным образом.

Они зачастую могут быть представлены в виде задач классификации структур (classification for structured inputs and outputs), в которых элементы множества Y представляют из себя не дискретные метки, а структуры (например, графы или деревья), и элементы X содержат их описания. Зависимости между элементами множеств X или Y, таким образом, переносятся на уровень признаковых описаний их отдельных элементов. Если же элементы множеств X и Y можно представить в виде последовательностей, такие задачи относят к разметке последовательностей (sequence labeling).

1.1.2 Задача классификации объектов сложной структуры

Под классификацией объектов сложной структуры понимаются все задачи, в которых некоторому множеству связанных объектов необходимо сопоставить множество меток классов. Примерами таких задач являются, например, распознавание речи, рукописного текста, расшифровка вторичной структуры белка, синтаксический и семантический анализ текста. Как уже было показано выше, основное отличие этих задач от классификации в общем виде состоит в том, что в них нельзя однозначно сопоставить элементам из X элементы из Y, так как, имеются зависимости как между признаками объектов, так и между метками их классов. Например, при распознавания рукописного текста признаки отдельного символа определяются особенностями почерка и контекстом этого символа в слове, а последовательности меток классов (отдельные слова) ограничиваются правилами орфографии.

Формальное определение задачи классификации объектов сложной структуры было дано ранее в [14, 15]. Пусть задано множество признаковых описаний объектов $\mathbb{X} \subseteq X^n$, и конечное множество меток классов $\mathbb{Y} \subseteq Y^k$. Существует неизвестная целевая зависимость $c : \mathbb{X} \to \mathbb{Y}$, с помощью которой построена обучающая выборка D_t, состоящая из t элементов. Пусть H - множество алгоритмов классификации объектов сложной структуры $h : \mathbb{X} \to \mathbb{Y}$ и $L : \mathbb{Y}^2 \to 0 \cup \mathbb{R}^+$ - функция потерь, такая что,

$$L(Y,Y') = \begin{cases} r \in \mathbb{R}^+, Y \neq Y' \\ 0, Y = Y' \end{cases}, \forall Y,Y' \in \mathbb{Y}. \tag{1.5}$$

Необходимо найти такой алгоритм $Y = h(X)$, который для заданного множества векторов признаков X возвращал бы соответствующее ему множество меток классов Y и минимизировал при этом функционал риска:

$$R(h) = \int_{\mathbb{X}} L(c(X),h(X))dX. \tag{1.6}$$

Примеры прикладных задач, в которых используется классификации объектов сложной структуры:

1. Определение частей речи для словоупотреблений (part-of-speech tagging) при выполнении морфологического анализа [16].
2. Синтаксический анализ [17].
3. Извлечение информации из текстов (information extraction) [7], например, извлечение метаданных [18, 19].
4. Распознавание рукописных текстов [20].
5. Распознавание речи [21].
6. Расшифровка вторичной структуры белка [22].

Частными случаями классификации объектов сложной структуры являются разметка последовательностей (sequence labeling), классификация временных рядов, классификация последовательностей, классификация деревьев, сетей и графов. Одними из наиболее близких друг к другу являются задачи классификации временных рядов и разметки последовательностей. Основное их отличие состоит в том, что в случае разметки последовательностей все анализируемые данные известны до начала классификации, тогда как при предсказании временных рядов известна только часть последовательности от начала до момента проведения анализа. Вторым важным отличием является отсутствие меток объектов при разметке последовательностей, тогда как при анализще временных рядов известны метки объектов от начала последовательности до момента анализа.

Еще одним близким классом задач является классификация последовательностей (sequence classification). В этих задачах каждой последовательности необходимо поставить в соответствие одну метку класса из конечного множества. Примером такой задачи является выявление отдельных слов при распознавании рукописного текста.

1.1.3 Оценка обобщающей способности классификаторов

Под оценкой обобщающей способности понимается нахождение верхней границы 1.4 для заданного семейства классификаторов. Для решения этой задачи широко используются т.н. концентрационные неравенства, которые позволяют определить границы того, насколько случайная величина может отклоняться от некоторого значения (как правило, ее математического ожидания).

Оценки с использованием VC-размерности Оценка обобщающей способности алгоритмов классификации с использованием VC-размерности исторически является одной из первых и наиболее известных попыток создания теории надежности обучения по прецедентам [23]. Как показали последующие эмпирические исследования [12], эта оценка оказывается очень сильно завышенной, кроме того, она применима только для задач бинарной классификации с $Y = \{-1, +1\}$. Тем не менее, она позволила формализовать важную закономерность: чем сложнее алгоритм классификации, тем больше данных нужно для его надежного обучения.

Базовым элементом этой теории является функция роста. Для семейства алгоритмов классификации H функция роста $\Pi_H : \mathbb{N} \to \mathbb{N}$ определяется следующим образом:

$$\forall m, \Pi_H(m) = \max_{\{x_1,\ldots,x_m\}\subseteq X} |\{h(x_1),\ldots,h(x_m) : h \in H\}|. \tag{1.7}$$

То есть функция роста для произвольного множества объектов размерности m возвращает количество всех различных способов, которыми это множество может быть классифицировано с использованием алгоритмов из H.

Тогда VC-размерность - это наибольшая размерность m множества объектов, которое может быть классифицировано всеми возможными способами с использованием семейства алгоритмов H:

$$d(H) = \max\{m : \Pi_H(m) = 2^m\} \tag{1.8}$$

Для алгоритма классификации с VC-размерностью $d = d(H)$, обученного на наборе данных размерности m, с вероятностью не менее $1 - \delta$ будет соблюдаться следующее неравенство:

$$R(h) \leqslant \hat{R}(h) + \sqrt{\frac{2d \log \frac{em}{d}}{m}} + \sqrt{log\frac{1}{\delta}} \tag{1.9}$$

Оценки с использованием Радемахеровской сложности Пусть заданы семейство функций H, с образом $Z = X \times Y$ и прообразом $[a,b]$, множество $S \subset Z$ размерности m, тогда эмпирическая оценка радемахеровской сложности на этом множестве равна:

$$\hat{\mathfrak{R}}_S(H) = M_\sigma\left(\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i h(z_i) \right), \tag{1.10}$$

где $\sigma_i \in \sigma$ - элементы множества, состоящего из равномерно и независимо распределенных дискретных случайных величин $\{-1, +1\}$. В отличие от VC-размерности, радемахеровская сложность применима не только к алгоритмам классификации, но и регрессии. Существует следующее соотношение, связывающее радемахеровскую сложность и ее эмпирическую оценку, вычисленную на множестве S размерности m:

$$\mathfrak{R}_m(H) \leqslant \hat{\mathfrak{R}}_S(H) + 3\sqrt{\frac{log\frac{2}{\delta}}{2m}} \tag{1.11}$$

Рассмотрим случай многоклассовой классификации, когда используется l семейств алгоритмов, по одному на каждый класс: $H = \{H_1, \ldots, H_l\}$ и $H_i : X \times Y \to \mathbb{R}, \forall H_i \in H$. Будем называть отступом величину:

$$\rho(x,y) = h(x,y) - max_{y' \neq y} h(x,y'), \forall x,y : x \in X, y \in Y, y = c(x). \tag{1.12}$$

В ходе обучения классификатора, как правио, выбирается семейство и алгоритм, максимизирующий этот отступ 1.12.

Построим множество отображений, оптимизирующих ρ на S:

$$\Pi_1(H) = \{x \to h(x,y) : y \in Y, h \in H\} \tag{1.13}$$

В работах [24, 25] доказано, что для фиксированного минимального отступа $\rho > 0$ риск с вероятностью не менее $1 - \delta$ составит:

$$R(h) \leqslant \hat{R}(h) + \frac{2l^2}{\rho}\mathfrak{R}(\Pi_1(H)) + \sqrt{\frac{log\frac{1}{\delta}}{2m}} \tag{1.14}$$

Равномерная оценка стабильности (устойчивости) алгоритмов обучения классификаторов и ее связь с обощающей способностью Приведенные выше оценки зависят только от сложности *алгоритмов классификации* и размера обучающей выборки: при их вычислении не учитываются особенности *алгоритмов обучения*. Однако, существует ряд алгоритмов обучения (см., например раздел 1.2.3), которые существенным образом влияют на обобщающую способность получаемых с их помощью классификаторов. Для оценки обобщающей способности в таком случае может быть использован подход, основанный на понятии равномерной устойчивости алгоритмов обучения. Обозначим функцию потерь алгоритма $h : \mathbb{X} \to \mathbb{R}$ в точке x через $l(h(x))$. Пусть S и S' - это любые два обучающих набора длиной m, которые отличаются на один элемент. Тогда алгоритм обучения A равномерно стабилен с оценкой стабильности $\boldsymbol{\gamma}$, если алгоритм, который он строит при обучении на **всех** возможных выборках S и S' длины m, удовлетворяет:

$$\forall S \in Z^m, \parallel l(h_S(x)) - l(h_{S'}(x)) \parallel_\infty \leqslant \boldsymbol{\gamma}_m, \tag{1.15}$$

где $Z = \mathbb{X} \times \mathbb{R}$, а $\|\|_\infty$ - равномерная норма (норма Чебышева). Здесь и далее предполагается, что алгоритмы h_s возвращают вещественные числа, что применимо для случая регрессии, однако, как было показано в статье [26], такой подход может быть легко модифицирован для функций вида $h_s : \mathbb{X} \to \{0,1\}$ (случай бинарной классификации). Поэтому, далее, для упрощения рассуждений мы будем исходить из определения (1.15), полагая что все полученные далее результаты могут быть легко модифицированы для иных случаев.

Оценка равномерной стабильности может быть использована для определения обобщающей способности алгоритмов классификации (регрессии), построенных с помощью равномерно стабильных алгоритмов обучения [26].

Теорема 1. *(Evgeniou, 2004). Пусть алгоритм h_D построен с применением $\boldsymbol{\gamma}_m$-стабильного обучающего алгоритма и функции потерь l, которая ограничена $0 \leqslant l \leqslant M$. Тогда, с вероятностью не менее $1 - \delta$ для любого множества D, состоящего из m одинаково и независимо распределенных элементов $D \in Z^m$ имеет место следующее:*

$$R(h_D) \leqslant \hat{R}(h_D) + 2\boldsymbol{\gamma}_m + (4m\boldsymbol{\gamma}_m + M)\sqrt{\frac{log\frac{1}{\delta}}{2m}} \tag{1.16}$$

1.2 Методы классификации

1.2.1 Деревья решений

Бинарные деревья решений (ДР) широко используются для решения задач классификации. ДР представляют собой иерархические структуры (бинарные деревья), в узлах которых находятся разделители. В классических ДР с одномерными разделителями (далее - одномерные ДР) в узлах дерева происходит сопоставление отдельных орт векторов признаков (т.е. отдельных признаков) x_{ij} классифицируемых объектов с заранее определенными пороговыми значениями. В случае, если величина признака превосходит пороговое значение, происходит переход к правому поддереву-потомку, иначе - к левому (или наоборот). Листья дерева (терминальные узлы) помечены метками классов, которые присваиваются анализируемым объектам (См. рисунок 1.1).

Деревья решений обычно строятся с помощью жадных рекурсивных алгоритмов. На первом шаге в соответствии с заданным критерием из множества всех разделителей S выбирается правило для корневой вершины:

$$s : X \to \{left,right\}, \forall s \in S, \tag{1.17}$$

такое, что:

$$s(x) = \begin{cases} x_s > h_s, right \\ x_s \leqslant h_s, left \end{cases} , \forall x \in X, \forall s \in S, \tag{1.18}$$

где x_s - признак из признакового описания x, выбранный в s, а h_s - порог, использующийся в s. Затем обучающий набор данных разделяется согласно этому правилу, и алгоритм рекурсивно применяется к левому и правому поднабору. Алгоритм обучения заканчивает свою работу при достижении деревом заданной высоты, либо если все объекты в анализируемом поднаборе оказываются принадлежащими одному классу. По окончании обучения каждый лист помечается меткой того класса, к которому относится большинство объектов обучающей выборки, соответствующих этому листу.

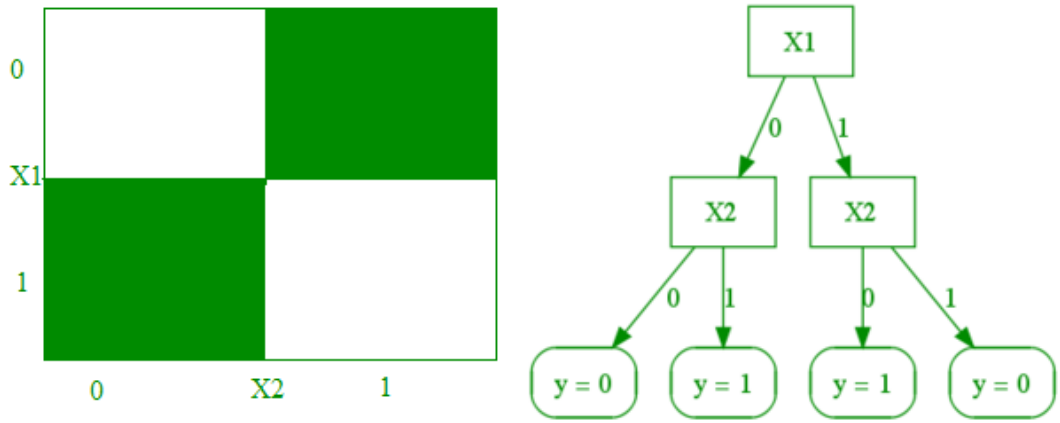

Рисунок 1.1 — Задача "XOR"и и дерево решений для нее

В предыдущих работах было предложено большое количество различных методов построения деревьев решений, которые отличаются друг от друга, главным образом, критериями, которые используется при построении разделителей: CART, C4.5 и ID3 [27, 28]. Все эти критерии так или иначе оценивают снижение равномерности распределения классов объектов в подмножествах, полученных с помощью разделяющего правила s (см. рисунок 1.2):

$$\theta(s) = g(p) - P_L g(p_L) - P_R g(p_R) \tag{1.19}$$

где P_L и P_R - эмпирические оценки вероятности отнесения объекта, соответственно, к левому и правому поддереву, вычисленные на обучающих данных. Векторы p, p_L и p_R содержат эмпирические оценки вероятности появления объектов определенных классов, соответственно, в корне, левом и правом поддеревьях. В качестве $g(p)$ наиболее часто используется критерий Джини:

$$g(p) = \sum_{i=1}^{N}(1 - p_i)p_i \tag{1.20}$$

или информационная энтропия:

$$g(p) = -\sum_{i=1}^{N} p_i log(p_i) \tag{1.21}$$

Здесь N - число различных классов, которым принадлежат объекты из обучающей выборки, попавшие в поддерево, а $p_i \in p$ - эмпирическая оценка вероятности появления объектов класса i в поддереве.

Основными достоинствами деревьев решений является легкость интерпретации результатов их обучения человеком, а также возможность распараллеливания процесса обучения. Недостатоком деревьев является их склонность к

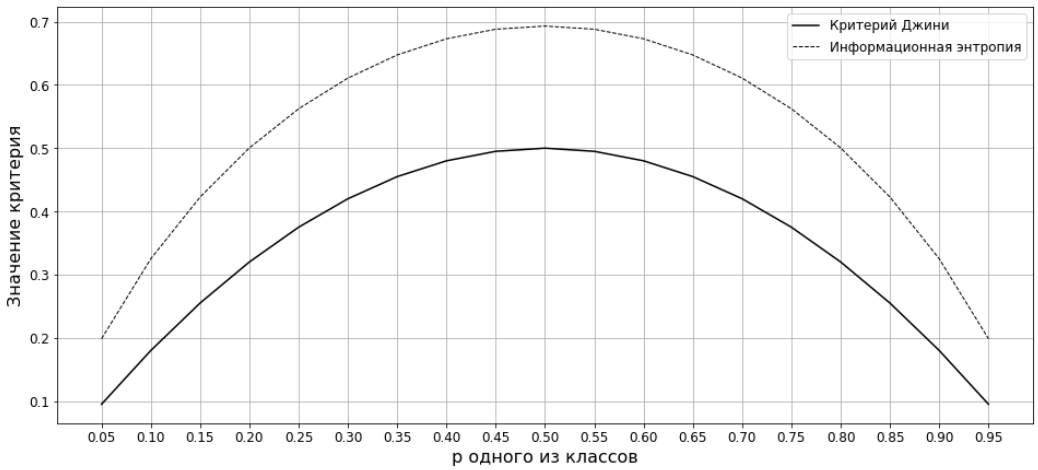

Рисунок 1.2 — Графики критериев построения разделителей для случая анализа выборки с двумя классами

переобучению (т.е. к большой разнице между риском и его эмпирической оценкой 1.4) и неустойчивость к незначительным изменениям в обучающем наборе данных, то есть небольшие изменения в обучающих данных могут влиять на результат обучения (см. выражение **??**).

Оценка обобщающей способности ДР для случая бинарной классификации была дана в [29]. Для начала, введем способ оценки эффективного количества листьев дерева N_{eff}.

$$N_{eff} = N(1 - ||P' - U||), \tag{1.22}$$

где N - количество листьев, P' - вектор, содержащий эмпирические вероятности принадлежности элементов обучающей выборки листьям, а $U = (\frac{1}{N}, \ldots, \frac{1}{N})$ - вектор аналогичной размерности, содержащий значения вероятностей в случае равномерного распределения элементов обучающей выборки по листьям. Тогда для обучающего набора данных размерности m, дерева высоты n с эффективным количеством листьев N_{eff} и VC-размерностью разделителя d будет соблюдаться следующее:

$$R(h) \leqslant \hat{R}(h) + O\left(\frac{N_{eff} d \log^2 m \log n}{m}\right)^{\frac{1}{3}} \tag{1.23}$$

Таким образом, чем ближе эмпирическое распределение обучающих данных по листьям к равномерному, тем ближе это число к фактическому. Показано также, что структура дерева решений зависит от выбора критерия неоднородности данных, оптимизируемого при обучении [27] и влияющего на обобщающую способность.

Большое влияние на обобщающую способность деревьев решений играет также взаимное расположение обучающих примеров и разделяющих поверхностей в признаковом пространстве: в статьях [30, 31] представлена оценка обобщающей способности деревьев решений, согласно которой увеличение отступа в узлах дерева позволяет снизить переобучение.

Для повышения устойчивости и снижения эффектов, связанных с переобучением, была предложена процедура редукции. Она состоит в переборе всех поддеревьев в дереве "снизу-вверх начиная с минимальных, состоящих только из одного разделителя и двух листьев. Каждое поддерево заменяется листом, помеченным мажоритарным классом в обучающем наборе данных, соответствующем поддереву, если это не ухудшает качество классификации. К сожалению, эта процедура не является панацеей во многих практических задачах. Дальнейшее повышение устойчивости алгоритмов, основанных на деревьях решений было связано с работами в области композиций классификаторов, а именно бэггинга и бустинга (см. раздел 1.2.3).

1.2.2 Линейные классификаторы

Линейными называются классификаторы, в которых для разделения признакового пространства, представимого в виде $F \subseteq \mathbb{R}^n$, на области, соответствующие отдельным классам, используются гиперплоскости:

$$h(x) = sign(w^T x - b), \tag{1.24}$$

где вектор действительных чисел $w = (w_1,w_2,..,w_n)$, $w_1,w_2,..,w_n \in \mathbb{R}$, $n \in \mathbb{N}$, и число $b \in \mathbb{R}$ задают положение этой гиперплоскости. Хотя эти классификаторы, сами по себе, не являются предметом исследования данной работы, они имеют важное значение как составной элемент деревьев решений с линейными разделителями, о которых пойдет речь в разделе 1.2.4 и главе 2.

Одним из наиболее широко используемых методов обучения линейных классификаторов является логистическая регрессия.В этом методе поиск разделяющей гиперплоскости осуществляется путем оптимизации логистической

функции потерь:

$$w^*,b^* = argmin_{w,b} \sum_{i=1}^{n} log_2(1 + e^{(b-w^T x_i)y_i}) \tag{1.25}$$

Основным достоинством логистической регрессии являктся возможность оценивать апостериорные вероятности принадлежности объекта классу, при условии, что функции правдоподобия классов $Li(Y|X)$ и признаки классифицируемых объектов удовлетворяют достаточно жестким ограничениям (подробнее - см. [32]). При решении практических задач необходимо проводить нормализацию признаков анализируемых объектов и отсев выбросов. Логистическая регрессия не содержит встроенных механизмов регуляризации получаемых моделей, оставляя этот вопрос открытым.

Метод опорных векторов (Support Vector Machine, SVM) [33], основан на работах В.Н. Вапника. Пусть решается задача бинарной классификации с $Y = \{-1,+1\}$. Будем именовать отступом для положительного класса величину $m^+ = \sum_{\forall x,y:y=+1} (w^T x - b)$ и для отрицательного - $m^- = \sum_{\forall x,y:y=-1} (b - w^T x)$. Обучение состоит в максимизации суммарного отступа $m = m^+ + m^-$ для заданного обучающего набора данных. Этот отступ не зависит от величины смещения гиперплоскости b (при условии, что она все еще разделяет подмножества объектов с положительными и отрицательными метками), поэтому изменим b, так чтобы m^+ и m^- стали равны (см. рисунок 1.3. Объекты, расположенные ближе всего к гиперплоскости, называются опорными векторами. Таким образом, отступ определяется как $s/||w||$, где s - расстояние между разделяющей гиперплоскостью и ближайшими к ней объектами из обучающего множества. Обычно b, w и m, подбираются таким образом, чтобы $s = 1$. Тогда максимизация отступа соответствует минимизации $||w||$ или $\frac{1}{2}||w||^2$, при условии, что расстояние от любого обучающего примера до разделяющей гиперплоскости будет не меньше, чем m. В случае, если обучающий набор данных линейно не разделим, это условие невозможно выполнить для всех объектов. Поэтому в оптимизационную задачу вводятся переменные невязки $\mathcal{E}_1$,..,$\mathcal{E}_n$ по одной для каждого условия. Благодаря этим переменным некоторым объектам разрешается находиться на расстоянии до разделяющей гиперплоскости меньшем, чем m, более того, они могут располагаться с противоположной стороны от разделяющей гиперплоскости:

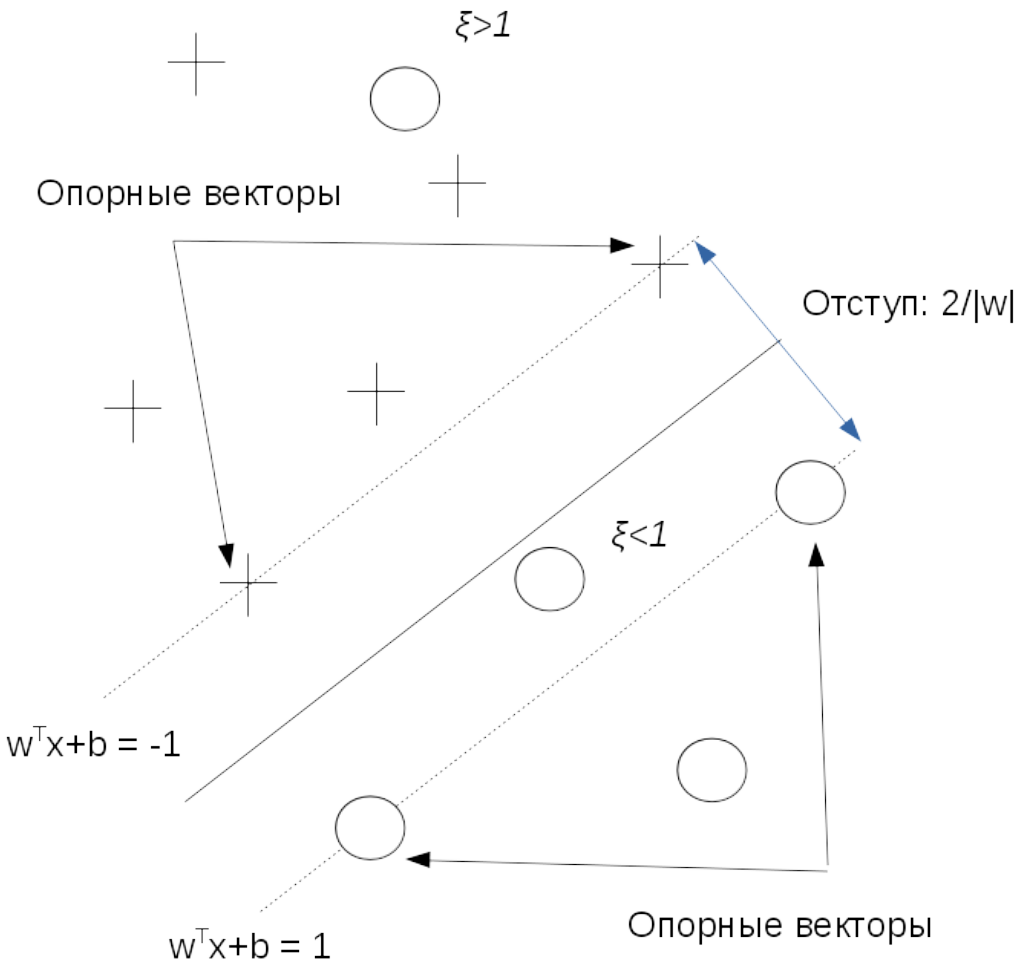

Рисунок 1.3 — Метод опорных векторов

$$w^*,b^*,\mathcal{E}^* = \arg\min_{w,b,\mathcal{E}} \left(\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\varepsilon_i\right), \tag{1.26}$$

при условии, что:

$$\forall(x_i,y_i) \in D, \varepsilon_i : \begin{cases} y_i(w^T x_i - b) \geqslant 1 - \varepsilon_i, \\ \varepsilon_i \geqslant 0. \end{cases} \tag{1.27}$$

Для постановки двойственной задачи оптимизации необходимо составить функцию Лагранжа:

$$\Lambda(w,b,\mathcal{E},A,\beta) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\mathcal{E}_i - \sum_{i=1}^{n}a_i(y_i(w^T x_i - b) - (1-\mathcal{E}_i)) - \sum_{i}^{n}\beta_i\mathcal{E}_i \tag{1.28}$$

После взятия частной производной от функции Лагранжа по b и приравнивания ее к 0 видно, что оптимальное значение Λ может быть получено только при $b = \sum_{i=1}^{n}$. Аналогично, взяв частную производную по w, можно отметить, что

множители Лагранжа определяют вектор весов в виде линейной комбинации обучающих примеров:

$$\frac{\partial}{\partial w}\Lambda(w,b,\mathcal{E},A,\beta) = w - \sum_{i=1}^{n} a_i y_i x_i \tag{1.29}$$

Теперь исходную задачу можно заменить задачей оптимизации функции 1.28:

$$A* = \arg\max_{A} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j x_i x_j + \sum_{i=1}^{n} a_i \tag{1.30}$$

при условии, что:

$$\forall a_i \in A, y_i \in D : \begin{cases} 0 \leqslant a_i \leqslant C \\ \sum_{i=1}^{n} a_i y_i = 0 \end{cases} \tag{1.31}$$

Необходимо отметить, что в задаче 1.30 признаки объектов в явном виде не учитываются, а используются лишь скалярные произведения векторов признаков $\langle x_i x_j \rangle$. Эта особенность позволяет в случае линейно-неразделимых данных использовать так называемый "трюк с ядром"(kernel trick). Он состоит в том, чтобы ввести некоторую нелинейную функцию ядра, отображающую скалярные произведения $\langle x_i x_j \rangle$ в исходном признаковом пространстве на скалярные произведения в некотором другом пространстве, модифицированном так, чтобы обучающие данные стали разделимыми.

Обобщающая способность классификаторов, обучаемых с помощью метода опорных векторов существенным образом зависит от математического ожидания отступа на обучающем наборе данных и нормы вектора w (которая, в свою очередь, зависит от величины параметра C). Пусть имеется следующее множество алгоритмов классификации $H = \{x \to w^T x : ||w|| < \Lambda\}$ и множество признаков классифицируемых объектов имеет вид $X = \{x : ||x|| \leqslant r\}$. Тогда для любого отступа ρ с вероятностью не менее $\delta - 1$ будет соблюдаться следующее:

$$R(h) \leqslant \hat{R}(h) + 2\sqrt{\frac{r^2\Lambda^2/\rho^2}{m}} + \sqrt{\frac{log(1/\delta)}{2m}} \tag{1.32}$$

Метод опорных векторов обладает несколькими преисмуществами над другими методами линейной классификации.

1. Результат обучения имеет свойство разреженности: положение разделяющей гиперплоскости зависит лишь от небольшой доли объектов из

обучающей выборки - опорных векторов, остальные объекты фактически не задействуются.

2. Обучение метода опорных векторов является задачей оптимизации выпуклой функции с ограничениями-неравенствами. Эта задача имеет единственное решение, и существуют эффективные методы его поиска на больших наборах данных [34].
3. С помощью "трюка с ядром"можно вместо гиперплоскостей использовать нелинейные разделяющие поверхности.

1.2.3 Композиции классификаторов

Композиция классификаторов включает в себя множество базовых алгоритмов, построенных на одном наборе данных, и правила формирования согласованного результата на основании оценок этих алгоритмов. Основная идея, лежащая в основе большинства подходов к построению композиций классификаторов состоит в том, чтобы взаимно компенсировать ошибки базовых алгоритмов, получая, в итоге, более низкое значение риска R, чем при использовании каждого из классификаторов по-отдельности. В работах Ю.И. Журавлева [35] для формального определения композиций классификаторов наряду с множеством объектов и множеством их меток вводится вспомогательное множество, называемое пространством оценок. Необходимо построить алгоритм, в котором функция, называющаяся алгоритмическим оператором, устанавливает соответствие между множеством объектов и пространством оценок, а функция, называющаяся решающим правилом, устанавливает соответствие между пространством оценок и множеством меток. Согласно этому определению, композиция классификаторов является суперпозицией алгоритмического оператора и решающего правила.

Одним из первых методов построения композиций классификаторов стал стекинг [36]. В этом методе итеративно строится иерархическая композиция классификаторов, в которой результаты работы классификаторов верхнего уровня используются как обучающие данные для классификаторов следующего уровня, и.т.д. На первом шаге обучающая выборка D разделяется на r частей. На каждой из этих частей обучается классификатор h_{1i}, где $i = 1..r$. Результа-

том классификации D с помощью h_{1i} будет вектор меток размерности r. Этот вектор далее используется в качестве набора данных для обучения классификаторов 2-го уровня. Процесс продолжается до тех пор, пока не будет достигнута заданная глубина иерархии, являющаяся гиперпараметром алгоритма.

Метод построения композиций классификаторов AdaBoost (адаптивный бустинг) был предложен в [37]. Пусть дана выборка $S \subset D$, будем обучать на ней композицию бинарных классификаторов h_1,h_2,..,h_n вида:

$$F_M(x) = \sum_{t=1}^{N} b_t h_t(x), \tag{1.33}$$

где $b_t \in \mathbb{R}$.

Для построения композиции используется жадный алгоритм, который последовательно генерирует b_t и h_t, минимизируя эмпирический риск. Зададим множество весов элементов из S: $W = \{w_1,w_2,..,w_n : n = |D|, w_i \in [0..1]\}$. На первом шаге построения композиции это множество будет содержать элементы $w_i = 1/|S|$. Пусть в композиции уже построено t базовых классификаторов. Классификатор $h_t \in H$ обучен на выборке S_t, построенной из S в соответствии с весами элементов W_t. Этой композиции соответствует эмпирический риск $\varepsilon_t = \hat{R}_{W_t}(h_t)$. Зададим вес классификатора h_t в композиции обратно пропорционально величине эмпирического риска:

$$b_t = \frac{1}{2} log \frac{1 - \varepsilon_t}{\varepsilon_t} \tag{1.34}$$

Теперь скорректируем веса примеров в обучающей выборке так, чтобы повысить количество примеров, некорректно классифицированных на предыдущем шаге:

$$W_{t+1} = \frac{W_t}{Z_t} \times \{\delta_i : \delta_i = -b_t y_i h_t(x_i)\}, \tag{1.35}$$

где Z_t - нормирующий коэффициент, и сгенерируем новую выборку S_{t+1}, на которой будем обучать классификатор $h_{t+1} \in H$. Количество таких шагов является гиперпараметром метода.

В [37] была доказана теорема, которая утверждает, что если эмпирический риск базовых алгоритмов на обучающей выборке не превосходит 0.5, то эмпирический риск композиции уменьшается экспоненциально с добавлением каждого следующего классификатора. Пусть $\varepsilon_t \leqslant 0.5, \forall t \in [1,\ldots,N]$ и

$\gamma_t = 0.5 - \varepsilon_t$, тогда:

$$\hat{R}(F_M) \leqslant \prod_{t=1}^{N} \sqrt{1 - 4\gamma_t^2} \leqslant e^{-2\sum_{t=1}^{N} \gamma_t^2} \tag{1.36}$$

Многочисленные исследования показали [38], что снижение эмпирической оценки риска при построении композиции с помощью AdaBoost сопровождается повышением математического ожидания отступа 1.12 на обучающей выборке.

В [38] была выполена оценка риска для построенной композиции в случае решения задачи бинарной классификации, где d - VC-размерность для семейства базовых классификаторов H, m - размер обучающей выборки, θ - математическое ожидание отсупа 1.12 на обучающей выборке. С вероятностью не менее $1 - \delta$ соблюдается следующее неравенство:

$$R(F_M) \leqslant P_S\big(\hat{R}(F_M) \leqslant \theta\big) + O\left(\frac{1}{\sqrt{m}}\Big(\frac{d \log^2 \frac{m}{d}}{\theta^2} + log\frac{1}{\delta}\Big)^{\frac{1}{2}}\right) \tag{1.37}$$

Отсюда следует, что обобщающая способность бустинга улучшается с увеличением отступа и не ухудшается с ростом размера композиции N. То есть, важную роль в этом случае играет не столько сама композиция классификаторов, но алгоритм ее построения. Отсюда же следует неэффективность этого подхода для построения композиций алгоритмов, при обучении которых максимизировался отступ (например, SVM). Однако, AdaBoost имеет и недостатки:

1. Чрезмерная чувствительность к выбросам, зашумленным данным из-за использования показательной функции потерь 1.34.
2. AdaBoost строит чрезмерно большие, плохо интерпретируемые композиции.
3. Не удаётся строить короткие композиции из «сильных» алгоритмов.
4. Для надежного обучения необходимы достаточно большие обучающие выборки.
5. Плохо поддается распараллеливанию.

В [39] был предложен другой способ построения композиций классификаторов, получивший название "градиентный бустинг":

$$F_M(x) = \sum_{m=1}^{M} b_{mh}(x; a_m), b_m \in \mathbb{R}, a_m \in A. \tag{1.38}$$

В нем, как и в адаптивном бустинге, построение композиции производится с помощью жадного алгоритма:

$$F_m(x) = F_{m-1}(x) + b_{mh}(x; a_m), b_m \in \mathbb{R}, a_m \in A. \tag{1.39}$$

Градиент эмпирического риска по функционалу построенной части композиции можно тогда записать так:

$$\nabla \hat{R} = \Big(\frac{\partial \hat{R}}{\partial F_{m-1}}(x_i)\Big)_{i=1}^{N} = \Big(\frac{\partial L(y_i, F_{m-1})}{\partial F_{m-1}}(x_i)\Big)_{i=1}^{N} \tag{1.40}$$

И композиция на шаге m может быть построена на основе композиции на шаге $m-1$ по формуле:

$$F_m = F_{m-1} - b_m \nabla \hat{R}, b_m \in \mathbb{R}. \tag{1.41}$$

Коэффициент b_m можно найти с помощью оптимизации линейной функции:

$$b_m = \arg\min_{b \in \mathbb{R}} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) - b \nabla \hat{R}_i) \tag{1.42}$$

Основные преимущества перед AdaBoost:

1. Более "общая"постановка задачи (AdaBoost можно представить как частный случай градиентного бустинга).
2. Проще реализовать аддитивную регуляризацию.
3. Можно подобрать функцию потерь, при которой композиция будет меньше переобучаться на небольших выборках.

Бэггинг (Bootstrap aggregating, bagging) и собэггинг (so-bagging) были предложены в [40] для повышения устойчивости классификаторов на основе деревьев решений к незначительным изменениям в обучающем наборе данных. В бэггинге выборки для обучения классификаторов формируется путем случайного размещения с повторениями объектов из исходной обучающей выборки. В собэггинге генерация обучающих выборок производится с помощью случайного размещения без повторений. Как и в предыдущих случаях, значительное влияние на обобщающую способность бэггинга и собэггинга оказывает алгоритм построения композиции. Наиболее достоверных результатов [41] здесь удалось добиться с помощью оценок с использованием равномерной стабильности (**??**). Для бэггинга получена следующая равномерная оценка стабильности. Пусть описанным выше способом построены обучающие наборы данных размерности m, на которых обучается композиция базовых алгоритмов, чья функция потерь является B-Липшицевой. Пусть равномерная оценка стабильности этих

алгоритмов составляет β_k. Тогда равномерная оценка стабильности композиции ограничена сверху:

$$\beta_m \leqslant B \sum_{k=1}^{m} \frac{k\beta_k}{m} P_r(d(r) = k), \tag{1.43}$$

где $d(r)$ - количество различных примеров в сгенерированной обучающей выборке, и $r \in \{1, \ldots, m\}^m$ - реализация случайной величины, управляющей выборкой примеров из общего обучающего набора D.

Для собэггинга равномерная оценка стабильности ограничена также сверху:

$$\beta_m \leqslant B\beta_p \frac{p}{m}, \tag{1.44}$$

где p - размерность генерируемых обучающих наборов, m - размерность исходного обучающего набора. В обоих случаях стабильность всей композиции не хуже (а на практике - лучше), чем стабильность базового классификатора. Далее β_m может использоваться для оценки обобщающей способности композиции с помощью формулы **??**.

Важно отметить, что методы оценки обобщающей способности случайных композиций деревьев решений на основе равномерной стабильности алгоритмов обучения отсутствуют. Создание таких методов позволило бы сформировать дополнительные требования к алгоритмам обучения и регуляризации композиций деревьев решений, подбору гиперпараметров, повысить их обобщающую способность.

Метод случайных подпространств (random subspace method, RSM) состоит в том, что базовые алгоритмы обучаются на различных подмножествах признакового описания, которые выделяются случайным образом [42], аналогично порождению обучающих примеров в бэггинге. В задачах с большой размерностью признаковых описаний и малым числом обучающих объектов, а также при наличии избыточных признаков, композиции, в которых базовые алгоритмы построены на случайных подмножествах признаков, могут обладать лучшей обобщающей способностью по сравнению с классификаторами, построенными с использованием всех признаков.

Случайный лес решающих деревьев предложен Луи Брейманом в работе [43], в которой обобщаются результаты его исследований в области построения композиций деревьев решений. Случайный лес является модификацией бэггинга, в котором для каждого классификатора помимо генерации обучающего

подмножества объектов формируется также случайное подмножество признаков, на котором производится обучение.

1. Качество классификации сопоставимо с Adaboost на небольших наборах данных. На больших Adaboost выигрывает.
2. Полученная композиция относительно устойчива к выбросам и шуму в обучающей выборке.
3. Меньшее время обучения по сравнению с бэггингом или бустингом.
4. Построение леса можно производить параллельно на распределенной вычислительной системе.

В этой работе был также предложен способ теоретической оценки верхней границы риска для случайных лесов. Пусть θ - случайный вектор, задающий объекты из D, использующиеся при обучении дерева $h \in H$ из леса H,

$$rmg(\theta,X,Y) = I(h(X,\theta) = Y) - I(h(X,\theta) = \hat{j}(X,Y)) \tag{1.45}$$

- отступ этого дерева решений на X,Y, где

$$\hat{j}(X,Y) = \arg\max_{j \neq Y} P_\theta(h(X,\theta) = j). \tag{1.46}$$

Заметим, что отступ тем больше, чем больше объектов из X правильно классифицируется деревом. Зададим функцию отступа

$$mr(X,Y) = P_\theta(h(X,\theta) = Y) - \max_{j \neq Y} P_\theta(h(X,\theta) = j) : X \times Y \to R \tag{1.47}$$

и будем называть "силой"дерева решений математическое ожидание этой функции на $X \times Y$:

$$s = M_{X,Y} mr(X,Y). \tag{1.48}$$

Пусть $\rho(\theta,\theta')$ - корреляция между отступами деревьев, обученных с различными θ: $rmg(\theta,X,Y)$ и $rmg(\theta',X,Y)$, а

$$\overline{\rho}(\Theta) = \frac{M_{\theta,\theta'}(\rho(\theta,\theta')\sigma(\theta)\sigma(\theta'))}{M_{\theta,\theta'}(\sigma(\theta)\sigma(\theta'))} \tag{1.49}$$

- математическое ожидание этой корреляции на Θ. Тогда верхняя граница риска:

$$R(H,\Theta) \leqslant \overline{\rho}(1 - s^2)/s^2 \tag{1.50}$$

Хотя эта оценка не может применяться на практике из-за невозможности вычисления s и $\overline{\rho}$, она показывает, что на эмпирический риск случайного леса

деревьев решений влияют две оставляющих: математическое ожидание отступа композиции H и корреляция отступов базовых классификаторов. Поэтому, можно было бы понизить риск для случайного леса, если удалось бы использовать в качестве классификаторов в композиции алгоритмы, обеспечивающие более высокий показатель s, при условии невозрастания показателя $\overline{\rho}$. Это предположение является основной мотивацией при создании модификаций этого метода, в том числе, предложенных в настоящем исследовании.

Для случайных композиций алгоритмов, в том числе для случайных лесов, ранее были предложены методы оценки обобщающей способности на основе равномерной стабильности алгоритмов обучения композиций [41].

Однако, такой подход не учитывает особенности случайных алгоритмов построения композиций. Была предложена равномерная оценка стабильности случайных алгоритмов, которая вычисляется следующим образом [41].

$$\sup_{D,z} |M_{\mathbf{r}}(l(f_{D(\mathbf{r})},z) - M_{\mathbf{r}}(l(f_{D^{\setminus i}(\mathbf{r})},z)))| \leqslant \beta_m, \tag{1.51}$$

где $D(\mathbf{r})$ - набор данных, сгенерированный в соответствии со случайным параметром $\mathbf{r}$, $D^{\setminus i} = D \setminus D_i$.

Заметим, что согласно этой работе, если функция потерь - B-Липшицева, то равномерная оценка стабильности случайных алгоритмов ограничена следующим образом:

$$\begin{aligned} &\sup_{D,z} \frac{B}{T} M_{\mathbf{r_1},\ldots,\mathbf{r_T}} |\sum_{t=1}^{T} \left(f_{D(\mathbf{r_t})}(x) - f_{D^{\setminus i}}(x)\right)| \\ &\qquad = \sup_{D,z} \frac{B}{T} M_{\mathbf{r_1},\ldots,\mathbf{r_T}} |\sum_{t=1}^{T} \left(\gamma_{d(\mathbf{r})}\right) \geqslant \beta_m, \end{aligned} \tag{1.52}$$

где $d(\mathbf{r})$ - количество различных элементов в обучающем наборе $D(\mathbf{r_t})$.

Для процедуры бэггинга была предложена равномерная оценка стабильности случайных алгоритмов. Перед ее описанием необходимо задать несколько обозначений и предварительных условий ее использования:

1. Пусть $\mathcal{R} = \{0,1\}^N$. Генерация обучающих наборов данных для алгоритмов в композиции производится в соответствии с одинаково и независимо распределенными случайными параметрами $\mathbf{r} = \{\mathbf{r_1},\mathbf{r_2},\ldots,\mathbf{r_T}\}, \mathbf{r_t} \in \mathcal{R}$. Каждому алгоритму в композиции соответствует случайный вектор длины N, каждый элемент которого принимает

значение 1, если соответствующий пример из набора D используется при обучении этого алгоритма и 0 в противном случае. Будем обозначать $D(\mathbf{r_t})$ - набор данных, сгенерированный для элемента композиции номер t в соответствии с параметром $\mathbf{r_t}$. Очевидно, что $\mathbf{r}$ зависит от алгоритма построения композиции и не зависит от обучающего набора D.

2. Один и тот же параметр $\mathbf{r_t} \in \mathbf{r}$ используется для обучения классификаторов f_D и $f_{D\setminus i}$, где $D \setminus i$ представляет из себя набор D с изъятым примером i.
3. Наличие нескольких копий одного примера в обучающем наборе данных алгоритма не влияет на результаты обучения.

Теорема 2. *(Eliseeff, 2005). Пусть выполнены все условия 1-3, $F_{D,\mathbf{r}}$ - композиция из T алгоритмов, построенная методом бэггинга, а обучение этих алгоритмов производилось на множестве одинаково и независимо распределенных объектов $D \in Z^m$ с использованием B-Липшецевой функции потерь. Пусть алгоритмы построения всех моделей в композиции имеют равномерную оценку стабильности γ_m. Тогда равномерная оценка стабильности алгоритма бэггинга будет ограничена сверху следующим образом.*

$$\beta_m \leqslant \frac{B}{T}\sum_{t=1}^{T} M_{\mathbf{r_t}}(\gamma_{\mathbf{r_t}}|1_{i\in\mathbf{r_t}}) = B\sum_{k=1}^{m}\frac{k\gamma_k}{m}P_{\mathbf{r}}(d(\mathbf{r})=k) \tag{1.53}$$

Несложно заметить, что в случае построения композиции алгоритмов, обучаемых с помощью равномерно стабильных алгоритмов, то есть таких, для которых $\gamma_m = O(\frac{1}{m})$, бэггинг не будет стабильнее этих обучающих алгоритмов.

С использованием равномерной оценки стабильности бэггинга была выведена следующая оценка обобщающей способности композиций алгоритомов.

Теорема 3. *(Eliseeff, 2005). Пусть выполнены все условия 1-3, $F_{D,\mathbf{r}}$ - композиция из T классификаторов, построенная методом бэггинга, а равномерная случайная оценка стабильности алгоритма построения этой композиции равна β_m. Пусть обучение этих классификаторов производилось на множестве одинаково и независимо распределенных объектов $D \in Z^m$ с использованием B-Липшецевой функции потерь l, которая ограничена $0 \leqslant l \leqslant M$. Тогда с вероятностью не менее $1-\delta$:*

$$R(F_{D,\mathbf{r}}) \leqslant \hat{R}(F_{D,\mathbf{r}}) + 2\beta_m + \left(\frac{M+4m\beta_m}{\sqrt{2m}} + \frac{\sqrt{2}BM}{\sqrt{T}}\right)\sqrt{log(2/\delta)} \tag{1.54}$$

1.2.4 Деревья решений с многомерными разделителями. Задача эффективного построения разделителей

Переобучение является одной из основных проблем, связанных с усложнением разделителей в узлах деревьев решений. В статье [29] предложена оценка обобщающей способности деревьев на задаче бинарной классификации. В основе этой оценки лежит понятие «эффективного числа листьев» дерева. Чем ближе эмпирическое распределение обучающих данных по листьям к равномерному, тем ближе это число к фактическому. Показано, что верхняя граница ошибки классификации положительно зависит от эффективного числа листьев и VC-размерности (размерности Вапника-Червоненкиса) [13] алгоритма, применяемого в узлах деревьев. Необходимо отметить, что структура дерева решений, в том числе распределение данных по листьям, зависит от выбора критерия неоднородности данных, оптимизируемого при обучении [27] влияющего на обобщающую способность. В статьях [30, 31] представлена оценка обобщающей способности деревьев решений, согласно которой увеличение отступа в узлах дерева позволяет снизить переобучение. В работе [43] анализируются случайные леса, показано, что усложнение базовых алгоритмов не приводит к переобучению при условии сохранения низкой корреляции ошибок, которые допускают отдельные деревья. Внесение дополнительной вариативности в структуру деревьев снижает корреляцию и, как следствие, улучшает обобщающую способность композиции. Одним из способов внесения такой вариативности является применение жадных алгоритмов для обучения отдельных деревьев композиции. Метод опорных векторов (Support Vector Machines, SVM) широко применяется для обучения регуляризованных линейных и нелинейных классификаторов; он также используется для построения разделителей в деревьях решений. Одна из первых попыток использования SVM представлена в статье [44]. В ней предлагается метод построения разделителей в деревьях с фиксированной структурой. Однако его использование при обучении случайных лесов нежелательно, так как фиксированная структура может приводить к высокой корреляции между ошибками отдельных деревьев композиции. Кроме того, предложенная функция потерь не является выпуклой, что ограничивает применимость существующих вычислительно-эффективных методов оптимизации для обучения. В статье [4] предлагается вариант случайного леса деревьев

решений с линейными разделителями. Для поиска разделяющих гиперплоскостей используется гребневая регрессия (ridge regression). Метод позволяет обучать деревья только для задач бинарной классификации. В работе [45] отмечается, что критерии неоднородности не отражают пространственного распределения данных. Между тем, учет подобного распределения позволил бы добиться упрощения структуры деревьев без увеличения ошибки на обучающем наборе данных, что привело бы к повышению обобщающей способности. В [46] предложен метод, в котором для объектов каждого класса строится кластеризующая гиперплоскость. Биссектрисы углов, которые образуют эти гиперплоскости, используют в качестве разделителей. Результаты экспериментальных исследований показывают, что метод позволяет получать компактные деревья решений, имеющие относительно высокую обобщающую способность по сравнению с деревьями, построенными с помощью других подходов. В статьях [30, 31] представлены теоретические оценки обобщающей способности деревьев решений с нелинейными разделителями, показывающие, что переобучение может быть снижено путем максимизации отступа в узлах и минимизации следа матрицы Грама [47], применяемой для задания скалярного произведения в нелинейном пространстве. Представлен метод обучения деревьев решений с нелинейными разделителями для задач классификации разреженных данных большой размерности. Перечисленные работы сфокусированы на оптимизации отступа в узлах деревьев или критерия неоднородности, в то время как оба эти фактора влияют на переобучение [29, 45]. В статье [48] предложены «CO2-деревья» с линейными разделителями. При построении этих деревьев обучение формулируется в виде задачи построения классификатора структур со скрытыми переменными [49]. Осуществляется оптимизация выпукло-вогнутой функции потерь, которая может быть достаточно эффективно осуществлена с помощью градиентного метода, предложенного в [50]. Функция является верхней гранью эмпирического риска, причем ее поведение зависит от гиперпараметров и масштаба признаков, что ограничивает применимость метода. В статье [51] представлены составные деревья решений, внутренние узлы которых являются одномерными, а разделители в терминальных узлах выбираются из множества алгоритмов классификации различной сложности. В работе получена оценка обобщающей способности таких деревьев, которая используется для выбора алгоритмов классификации для каждого терминального узла. Предложены случайные составные деревья решений, которые могут применяться в

качестве базовых алгоритмов классификации в случайных лесах деревьев решений (Random Composite Forest, RCF).

Еще одним перспективным направлением является сквозное (end-to-end) обучение деревьев решений с линейными разделителями и их лесов. В статье [52] представлен подход к обучению вероятностных деревьев решений с помощью EM-алгоритма (EM, expectation- maximization). Экспериментальные исследования с использованием размеченных наборов изображений показали, что можно формировать более сложные разбиения, чем детерминированные деревья с линейными разделителями, но EM алгоритм достаточно медленно сходится на больших наборах данных. В исследовании [53] предлагается использовать метод обратного распространения ошибки, который обычно используется для обучения многослойных нейронных сетей [54], для обучения деревьев решений с линейными разделителями и фиксированной структурой. Вместе с тем использование фиксированной структуры деревьев может привести к получению избыточно сложных алгоритмов с большим количеством параметров и, как следствие, к ухудшению обобщающей способности. В статье [55] отмечается, что алгоритмы для построения деревьев решений с линейными разделителями имеют более высокую вычислительную сложность, чем алгоритмы обучения деревьев с одномерными разделителями. Кроме того, результаты обучения часто зависят от инициализации параметров разделяющих гиперплоскостей. В статье предложен метод построения деревьев решений с линейными разделителями (WODT, взвешенное дерево решений с линейными разделителями), в котором применяется гладкая функция потерь. Метод обучения состоит в назначении весов для объектов обучающей выборки и обучении узла дерева решения, путем оптимизации взвешенной информационной энтропии распределения объектов по дочерним узлам. Стоит отметить, что функция потерь WODT является невыпуклой; следовательно, глобальный оптимум не может быть гарантированно найден градиентными методами. В то же время результаты экспериментальных исследований метода показывают конкурентоспособные оценки точности классификации для многих наборов данных.

В статье [56] предлагается алгоритм, который улучшает заданное дерево решений и создает новое дерево с такой же или с сокращенной структурой, но с новыми значениями параметров разделителей, что позволяет повысить точность и полноту анализа данных. Гибридное дерево решений предлагается в [57]. Задача этого исследования – сокращение количества примеров, для

разделения которых необходимо применение узлов на основе метода опорных векторов. Для решения выполняется аппроксимация границы между классами, сформированной методом опорных векторов с помощью деревьев решений. В итоге строится гибридное дерево решений, которое имеет как одномерные узлы, так и узлы с линейными разделителями. В настоящей работе предложен метод обучения деревьев решений с нелинейными разделителями, то есть деревьев, в которых в качестве многомерных разделителей в узлах используются нелинейные двоичные классификаторы. Аналогично методу CO2 Forest [48], обучение разделителей в узлах деревьев осуществляется с помощью подходов, применяемых для решения задач классификации структур, однако, в предлагаемом методе эта задача сформулирована в явном виде. Аналогично WODT [55] задаются веса объектов обучающей выборки и минимизируется критерий неоднородности данных, однако в предложенном методе этот критерий оптимизируется совместно с отступом. Построение узлов деревьев решений сводится к решению задачи обучения SVM с масштабированием переменных невязки, то есть выпуклой оптимизации с ограничениями-неравенствами. Для обеспечения высокой скорости оптимизации предложено отказаться от поиска глобального оптимума критерия неоднородности данных в каждом узле. Вместо этого реализован алгоритм поиска квазиоптимальных решений.

1.3 Методы классификации объектов сложной структуры

1.3.1 Методы классификации со скользящим окном

Первым и наиболее очевидным подходом для решения задачи классификации объектов сложной структуры стало ее преобразование к задаче машинного обучения в обычой постановке. Этот подход сводится к обучению классификатора следующего вида:

$$y_i = h(x_{i-t},x_{i-t+1},\ldots,x_i,\ldots,x_{i+t-1},x_{i+t}), \tag{1.55}$$

где $x_{i-t},\ldots,x_{i+t}$ называется скользящим окном. Такой подход имеет следующие недостатки:

1. Ограниченный размер окна позволяет учитывать только признаки ближайших соседей.
2. Не учитываются возможные связи между метками классов y_i.

Пример первого недостатка приведен на рисуноке 1.4. Пусть необходимо обучить классификатор с окном шириной 2 отличать последовательность символов *"riib"* от последовательности *"riob"*. Предположим, что анализируется последовательность *"riib"*. Заметим, что первый символ последовательности r будет соответствовать сразу двум классам - "1"и "5". То есть, верхний путь и нижний путь на рисунке будут примерно одинаково вероятны, независимо от наблюдаемой последовательности символов. Если одна из двух последовательностей чаще встречается в обучающей выборке, скорее всего, классификатор выберет её.

Эта проблема возникает из-за того, что в процессе классификации объекта на результат влияют только его признаки и признаки нескольких соседних объектов, а не признаки всех объектов анализируемого множества.

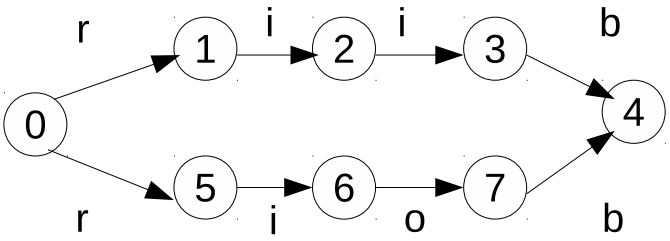

Рисунок 1.4 — Задача классификации последовательности, некорректно решаемая методами со скользящим окном

1.3.2 Скрытая марковская модель

Перечисленных недостатков лишена скрытая марковская модель [58]. Это генеративная графовая вероятностная модель, оценивающая совместное распределение $P(\mathbb{X},\mathbb{Y})$ наблюдаемой последовательности дискретных признаков объектов и последвательности меток их классов (рисунок 1.5). В этой модели

совместная вероятность оценивается следующим образом:

$$p(X,Y) = \prod_i p(y_i|y_{i-1}) \prod_j p(x_{i,j}|y_i) \tag{1.56}$$

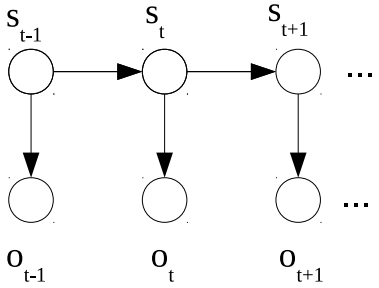

Рисунок 1.5 — Скрытая марковская модель

Для классификации с помощью этой модели используется алгоритм Витерби [58], позволяющий выявить последовательность меток, с наибольшей вероятностью соответствующих всей наблюдаемой последовательности объектов.

Слабой стороной скрытой марковской модели является лежащее в ее основе и заложенное в формуле (1.56) предположение о независимости признаков $x_{i,j}$ объектов друг от друга, что значительно ограничивает ее применимость для решения прикладных задач.

Другой проблемой является то, что для оценки совместной вероятности последовательностей объектов и последовательностей меток, должны быть явно заданы матрицы переходов соответствия признаков наблюдаемым объектам. При большом количестве признаков, классов и небольшом количестве обучающих примеров эти матрицы оказываются настолько разреженными, что вывод с использованием этой модели становится невозможен.

Еще один недостаток этого подхода получил название "проблема разделения меток это ситуация, когда преимущество получают состояния с меньшим количеством переходов. Она была впервые отмечена в работе [59]. Подобная проблема также встречается во всех методах, в которых при классификации не учитываются признаки всей последовательности, например в Марковских моделях максимальной энтропии (MEMM, Maximum Entropy Markov Model) [60].

1.3.3 Условное вероятностное поле

Другим подходом к решению задачи классификации объектов сложной структуры стало использование моделей, оценивающих распределение $P(\mathbb{Y}|\mathbb{X})$ вместо $P(\mathbb{X},\mathbb{Y})$. Одной из наиболее широко используемых моделей стало условное вероятностное поле (conditional random field, CRF) [61]. В этой модели условная вероятность $p(Y|X)$ оценивается следующим образом:

$$p(Y|X) = \frac{1}{Z(X)} e^{\sum_i \Psi(y_i, y_{i-1}, w_i(X))}, \tag{1.57}$$

где $w_i(X)$ - скользящее окно, $\Psi(y_i,y_{i-1},w_i(X))$ - функция, возвращающая оценку "совместимости"для y_i,y_{i-1} и $w_i(X)$, а $Z(X) = \sum_{\forall Y' \in Y} e^{\sum_i \Psi(y'_i,y'_{i-1},w_i(X))}$ - нормировочный множитель (рисунок 1.6).

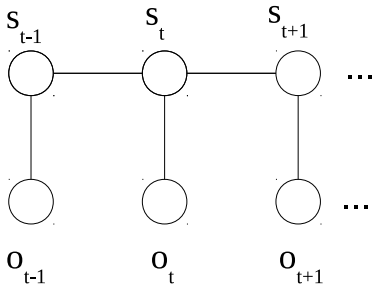

Рисунок 1.6 — Условное случайное поле

Благодаря использованию дискриминативной функции Ψ для оценки $P(\mathbb{Y}|\mathbb{X})$ в этой модели удалось преодолеть ограничение на зависимость признаков классифицируемых объектов. Для вывода, аналогично скрытой марковской модели, используется алгоритм Витерби, однако так как в условных случайных полях оценивается единое распределение вероятностей и нормализация производится для всей модели, а не в рамках каждого отдельного состояния, проблема разделения меток в этом методе отсутствует.

Такой подход имеет следующие недостатки:

1. Высокая сложность вычисления нормировочного множителя $Z(X)$.
2. Доказана только ассимптотическая сходимость обучения модели.
3. Процесс обучения модели плохо поддается распараллеливанию.
4. Отсутствует теория надежности обучения модели, подобная статистической теории переобучения [13] или комбинаторной теории належности обучения по прецедентам [62], что делает невозможной

теоретическую оценку верхней границы количества примеров, необходимых для обучения модели с заданным уровнем надежности.

1.3.4 Модификации метода опорных векторов для классификации структурированных объектов

Напомним, что под классификацией структурированных объектов понимаются задачи, в которых элементы множества Y представляют из себя не дискретные метки, а структуры, например, графы или деревья (1.7). Функции потерь, в которых оценивается подобие таких структур часто являются негладкими, что делает задачу минимизации эмпирического риска нерешаемой с помощью градиентных методов. Однако, все-таки существует несколько подходов, позволяющих трансформировать и решить эту задачу с помощью метода опорных векторов, то есть свести ее к выпуклой оптимизации с ограничениями-неравенствами. Один из этих подходов будет использоватья далее в главе 2 при построении линейных разделителей для деревьев решений, поэтому стоит подробно рассмотреть их.

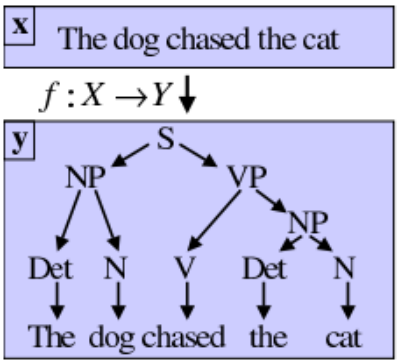

Рисунок 1.7 — Пример задачи классификации структурированных объектов

Интуитивно нарушение границы классов, связанное с высокими потерями L, должно наказываться более строго, чем нарушение, связанное с меньшими потерями. Это может быть достигнуто путем масштабирования отступов пропорционально потере или, что то же самое, за счет масштабирования параметров ограничений (slack variables).

Масштабирование отступов (margin re-scaling) было предложено в работе [63]. Пусть задано некоторое отображение признаковых описаний классифици-

руемых объектов и меток их классов на векторное пространство $\Phi : X \times Y \to \mathbb{R}^n$. Тогда прямую задачу оптимизации для метода опорных векторов можно сформулировать следующим образом:

$$w^*,\varepsilon^* = \frac{1}{2}||w||^2 + \frac{C}{m}\sum_{i=1}^{m}\varepsilon_i, \tag{1.58}$$

в соответствии с ограничениями:

$$\forall i,y \in Y : w^T\delta\Phi(x,y) \geqslant L(y,y_i) - \varepsilon_i, \tag{1.59}$$

где L - произвольная ограниченная сверху функция потерь вида (1.1), $\delta\Phi(x,y) = \Phi(x,y) - \Phi(x,y_i)$, C - параметр регуляризации, m - количество обучающих примеров.

Двойственная задача для такого подхода формулируется следующим образом:

$$A^* = \arg\max_a -\frac{1}{2}\sum_{i,y\neq y_i}\sum_{j,\tilde{y}\neq y_j} a_{iy}a_{j\tilde{y}}K(x_i,x_j) + \sum_{i,y\neq y_i} a_{iy}L(y_i,y) \tag{1.60}$$

, в соответствии с

$$\sum_{y\neq y_i} a_{iy} \leqslant \frac{C}{m} \tag{1.61}$$

Основной недостаток этого подхода состоит в том, что он не является инвариантным при масштабировании функции потерь. Например, если необходимо изменить масштаб отображения Φ на соответствующий масштабный коэффициент, то нужно еще изменить масштабирование функции потерь.

Еще одним недостатком масштабирования отступов является то, что он потенциально может давать большой вес классам $y \in Y$, которые даже совсем никак не пересекаются с целевыми y_i, потому что каждое увеличение функции потерь L риводит к увеличению отступа.

В работе [49] приведен другой подход, названный "масштабирование ограничений"(slack re-scaling). Этот подход лишен перечисленных выше недостатков. Прямая задача оптимизации формулируется в нем следующим образом:

$$w^*,\varepsilon^* = \frac{1}{2}||w||^2 + \frac{C}{m}\sum_{i=1}^{m}\varepsilon_i, \tag{1.62}$$

в соответствии с ограничениями:

$$\forall i, \forall y \in Y y_i : w^T \delta\Phi(x,y) \geqslant 1 - \frac{\varepsilon_i}{L(y,y_i)} \tag{1.63}$$

Двойственная задача для такого подхода формулируется следующим образом:

$$A^* = \arg\max_a -\frac{1}{2} \sum_{i,y\neq y_i} \sum_{j,\tilde{y}\neq y_j} a_{iy} a_{j\tilde{y}} K(x_i,x_j) + \sum_{i,y\neq y_i} a_{iy} \tag{1.64}$$

, в соответствии с

$$\sum_{y\neq y_i} \frac{a_{iy}}{L(y_i,y)} \leqslant \frac{C}{m} \tag{1.65}$$

где a_{ij} - вес примера.

Напомним, что $\hat{R}(w) = \frac{1}{n}\sum_1^n L(y_i,f(x_i,w))$ - оценка эмпирического риска на обучающем наборе данных. В работах [49, 63] приведены формальные доказательства того, что эти подходы действительно оптимизируют $\hat{R}(w)$, а именно:

Теорема 4. *Если $\varepsilon^*(w)$ - значения ограничений ε, оптимальные в смысле решения задачи (1.62) для заданного w^*, то оценка эмпирического риска будет ограничена сверху: $\frac{1}{n}\sum_{i=1}^n \varepsilon_i^* \leqslant \hat{R}(w)$.*

В прикладных задачах, как правило, размерность Y очень велика, поэтому решения задачи (1.62) в явном виде вычислительно неэффективны. В [49] и других предложены алгоритмы, позволяющие смягчить эту проблему, однако их подробное рассмотрение выходит за рамки настоящей работы.

1.3.5 Сверточные и рекуррентные нейронные сети

Сверточные сети (рис. 1.8) состоят из нескольких чередующихся слоев: слоя свертки и слоя субдискретизации. Слой свертки имеет относительно небольшое количество параметров и используется для преобразования признаков входных объектов путем "сканирования"полных признаковых описаний этих объектов (1.66).

$$x_{ij}^l = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} W_{ab} x_{(i+a)(j+b)}^{l-1}, \tag{1.66}$$

где W_{ab} - матрица весов сверточного слоя, x_{ij}^{l-1} - исходные признаки анализируемых объектов, x_{ij}^{l} - выходные признаки анализируемых объектов, m - размер сверточного слоя.

Слой дискретизации реализует нелинейное "уплотнение"преобразованных признаков объектов.

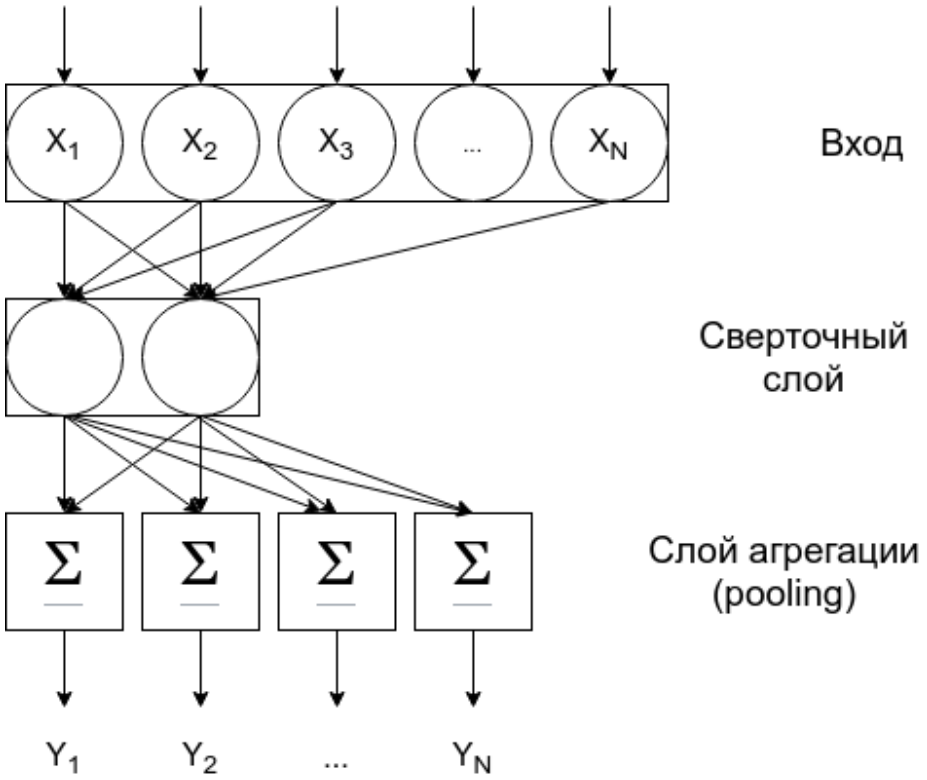

Рисунок 1.8 — Сверточная нейронная сеть

Сверточные сети наиболее активно применяются в задачах анализа изображений, известны различные модификации, различающиеся количеством и размером серточных слоев: AlexNet [64], U-net [65] и другие. Сверточные сети устойчивы к различным аффинным преобразованиям признаков анализируемых объектов, однако содержат большое количество гиперпараметров, подбор которых во время обучения может быть произведен только эмпирически.

Как и в случае со сверточными нейронными сетями, было предложено много разновидностей рекуррентных сетей, таких как сети Элмана [66], сети Джордана [67], нейронные сети с временной задержкой [68] и сети с эхо-состояниями [69]. Простая рекуррентная сеть содержит один скрытый слой (рис. 1.9). Хотя различие между многослойными нейронными сетями и рекуррентными нейронными сетями может показаться незначительным, оно имеет большое значение при решении задач классификации объектов сложной структуры.

Обычные многослойные нейронные сети могут только отображать векторы признаков объектов в векторы выходных значений, тогда как рекуррентные сети в принципе могут отображать всю историю предыдущих входных признаков на каждый выходной вектор (или вообще все признаки последовательности, в случае двунаправленных сетей). Ранее показано, что рекуррентная нейронная сеть с достаточным количеством скрытых слоев может аппроксимировать любое измеримое отображение последовательности в последовательность с произвольной точностью [70]. Ключевой особенностью является то, что соединение слоев, соответствующих различным моментам времени, позволяет «памяти» предыдущих входов сохраняться во внутреннем состоянии сети, что затем может использоваться для влияния на выход сети.

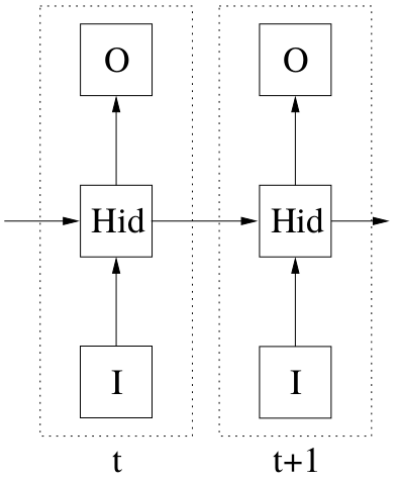

Рисунок 1.9 — Рекуррентная нейронная сеть

Прямой проход рекуррентной нейронной сети такой же, как и в многослойной сети с одним скрытым слоем, за исключением того, что в качестве данных в скрытый слой передается как вход сети на текущий момент времени, так и значение активационной функции скрытого слоя на предыдущем шаге. Рассмотрим входную последовательность x длины T, анализируемую сетью с I входными ней ронами, H скрытыми нейронами и K выходными нейронами. Пусть $x_t i$ будет значением входа i в момент времени t, и пусть $a_t j$ и $b_t j$ будут соответственно входом сети в нейрон j в момент времени t и активацией нейрона j в момент времени t. Для нейронов скрытого слоя получим:

$$a_{t,h} = \sum_{i=1} I w_{ih} x_{t,i} + \sum_{h'=1} H w_{h'h} b_{t-1,h'} \tag{1.67}$$

Функции активации затем применяются к выходам слоя точно так же, как и в случае многослойных нейронных сетей.

$$b_{t,h} = \theta_h(a_{t,h}) \tag{1.68}$$

Затем, последовательно рассчитываются значения активационных функций начиная с $t = 1$ и рекурсивно далее, применяя формулы (1.67) и (1.68), увеличивая t на каждом шаге. Заметим, что для этого требуется, чтобы начальные значения весов b_{0i} были выбраны для нейронов скрытого слоя, соответственно состоянию сети, прежде чем она получит какую-либо последовательность данных. Обычно b_{0i} устанавливается в ноль. Тем не менее, ряд исследователей обнаружили, что в некоторых случаях стабильность обучения и устойчивость к шуму можно улучшить, используя ненулевые начальные значения [71].

Выходы нейронов выходного слоя могут быть рассчитаны на то же время как на основе функций активации скрытого слоя:

$$a_{t,k} = \sum_{h=1} H w_{hk} b_{t,h} \tag{1.69}$$

Для обучения таких сетей используется обратное распространение ошибки во времени. [72]. Как и стандартном методе обратного распространения ошибки, распространение ошибки во времени состоит из многократного применения цепочечного правила. Особенность заключается в том, что для рекуррентных сетей при вычислении градиента ошибки учитывается, что скрытый слой влияет не только на выходной слой, но и на скрытый слой на следующем шаге.

К сожалению, в стандартных архитектурах рекуррентных нейронных сетей размер контекста (количество объектов в последовательности), к признакам которых можно получить доступ при обучении или классификации, ограничен. Проблема заключается в том, что влияние данного входа на скрытый слой, а следовательно, и на выход сети, либо затухает, либо возрастает экспоненциально, поскольку оно циклически повторяется вокруг повторяющихся соединений сети. На практике этот недостаток (называемый в литературе проблемой размытия градиента (gradient vanishing); [73]; [74] затрудняет использование рекуррентных нейронных сетей в задачах, в которых предполагаются задержки более чем примерно на 10 временных шагов между соответствующими входными данными и целевыми метками [73]. В 1990-х годах было предпринято

множество попыток решить проблему размывания градиента для рекуррентных нейронных сетей. Они включали не градиентные обучающие алгоритмы, такие как моделируемый отжиг и распространение дискретных ошибок [74], явно введенные временные задержки [68] или постоянные времени, и иерархические сжатие последовательности [75]. Однако до сих пор наиболее эффективным решением является архитектура долговременной памяти (Long-short Term Memory, LSTM) [76]. Её основным недостатком является большая сложность, то есть для обучения подобной сети необходимо большое количество размеченных данных. Еще одна модель с меньшим числом параметров, в которой эта проблема также была решена - GRU (Gated Recurrent Unit) [77]. Из-за того, что рекуррентные нейронные сети не оптимизируют вероятность $P(\mathbb{Y}|\mathbb{X})$, они не решают проблему разделения меток, поэтому часто используется комбинированный подход, в котором комбинируются рекуррентные нейронные сети и условные случайные поля.

Хотя многослойные нейронные сети, к которым относятся и рекуррентные сети, являются мощной парадигмой, у них есть недостатки.

Во-первых, рекуррентные нейронные сети часто имеют невыпуклую функцию потерь и большое количество гиперпараметров, причем эффективность обучения зависит от их тщательной настройки. Этот факт делает процесс обучения чрезвычайно сложным.

Во-вторых, обучение рекуррентной сети требует большого количество обучающих данных, и, следовательно, её вряд ли можно применять к задачам, где имеются только небольшие обучающие наборы данных. Во многих реальных прикладных задачах все еще не хватает больших размеченных обучающих наборов данных. Попытки решения этих задач с использованием нейронных сетей приводят к их переобучению. Кроме того, нейронные сети являются моделями черного ящика, процессы принятия решений в которых трудно понять, их поведение очень сложно для теоретического анализа. Кроме того, архитектура, а, соответственно, и сложность нейронной сети должна быть определена заранее, до обучения.

1.3.6 Многослойные нейронные сети с архитектурой Трансформер

Многослойные нейронные сети с архитектурой "Трансформер"[78] относятся к сетям типа "Автокодировщик". Они состоят из двух блоков: кодировщика, формирующего векторные представления входных данных и декодировщика, восстанавливающего входные данные на основе сформированных векторных представлений (рис. 1.10). Для идентификации взаимного положения анализируемых объектов к их признакам на входе конкатенируются дополнительные векторные представления, формируемые с помощью периодических функций. В сетях с архитектурой "Трансформер"кодировщик и декодировщик состоят из нескольких последовательных типовых блоков. Каждый блок состоит из "многоголового"механизма внимания (multihead attention), механизмов сложения и нормализации и многослойной полносвязной сети.

На вход механизма внимания подается матрица запросов Q, матрица ключей K, и матрица значений V. Все эти матрицы состоят из векторов размерности d_k. На практике, как правило, все три матрицы идентичны: $Q = K = V$. Механизм внимания с h головами представляет собой следующую функцию:

$$Multihead(Q,K,V) = W_0 Concat(head_1,head_2,\ldots,head_h), \tag{1.70}$$

где

$$head_i = softmax(\frac{W_i^Q Q W_i^k K^T}{\sqrt{d_k}})VW_i^V \tag{1.71}$$

и W_i^Q, W_i^k, W_i^V - матрицы весов, подбираемых в ходе обучения сети.

Значение этой функции конкатенируется со входным вектором, нормируется и передается на вход полносвязной сети.

Обучение сети проводится в два этапа. На первом этапе проводится обучение без учителя, часть входных данных маскируется и "Трансформер"обучается восстанавливать эти данные. На втором этапе в задачах классификации используется только кодировщик сети. Он до-обучается (fine-tuning) с учителем на размеченной выборке для решения заданной прикладной задачи.

Изначально эта архитектура была предложена для решения задач анализа текстов, в последующих работах были предложены модификации для анализа временных рядов [79], сетевых структур и изображений [80]. Отказ от использования обратного распространения ошибки во времени и использование

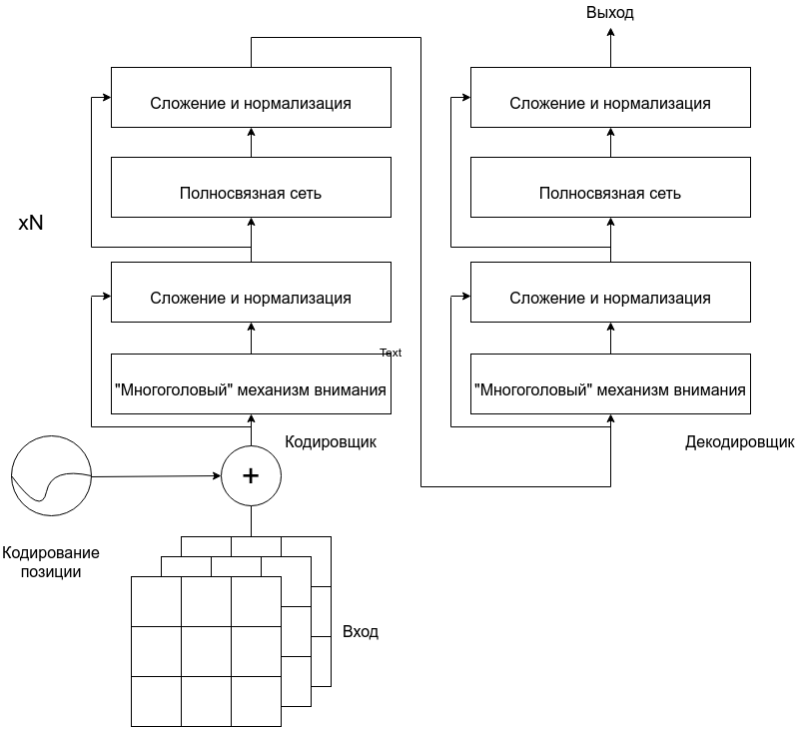

Рисунок 1.10 — Многослойная нейронная сеть с аритектурой Трансформер

механизма внимания позволили при обучении учитывать сложные взаимосвязи анализируемых объектов, в том числе достаточно удаленных друг от друга.

Среди основных недостатков сетей "Трансформер"необходимо отметить чрезвычайно избыточное количество параметров, усложняющее процесс обучения, а также ограниченный размер анализируемых множеств объектов. Предполагается, что многослойные сети обычно являются более сложными, чем необходимо. Это подтверждается тем, что появилось много исследований, в которых улучшение качества работы нейронных сетей достигается путем их упрощения: добавления сверточных слоев [81], дропаута [82], бинаризации [83]. Эти операции упрощают архитектуру сетей и фактически уменьшают сложность моделей. Также следует отметить, что по-прежнему существует много задач, в которых нейронные сети не превосходят другие подходы, например, Random Forest [43] или XGBoost [84] по-прежнему являются победителями на многих соревнованиях по решению прикладных задач методами машинного обучения.

1.3.7 Композиции (stack) на случайных лесах решающих деревьев

Нейронные сети являются композицией дифференцируемых нелинейных функций, однако не все свойства классифицируемых объектов при решении практических задач лучше всего моделируются с помощью таких функций. Был проведен ряд исследований, в которых выяснялось, можно ли реализовать глубокое обучение с типами алгоритмов, отличными от нейронных сетей.

Решением проблем методов на основе многослойных нейронных сетей, мог бы быть подход, в котором сложность алгоритмов классификации можно было бы регулировать автоматически в зависимости от данных. Для решения сложных задач необходимо использовать иерархические композиции из нескольких алгоритмов классификации. Однако подобные алгоритмы, как правило, основаны на нейронных сетях. Как было показано выше, есть некоторые причины для использования других подходов.

В работе [85] предложен метод gcForest (Deep Forest), который позволяет строить иерархические композиции классификаторов на основе деревьев решений. В ней показано, что подобные классификаторы можно строить не только с помощью многослойных нейронных сетей, обучаемых методом обратного распространения ошибки. Предложенный в этой работе метод позволяет строить композицию деревьев решений с иерархической, структурой, где каждый слой содержит несколько случайных лесов.

Иерархическая структура потенциально позволяет gcForest учитывать контекст объектов при классификации. Количество иерархических уровней может автоматически определяться так, чтобы сложность модели регулировалась в зависимости от количества обучающих данных, а не заранее, до обучения; что позволяет gcForest работать даже на небольших данных и регулировать использование вычислительных средств. Более того, у gcForest гораздо меньше гиперпараметров чем у мгогослойных нейронных сетей, и он менее требователен к их настройке.

Алгоритм построения gcForest состоит из двух шагов (рис. 1.11). На первом шаге используются скользящие окна для анализа исходных признаков классифицируемых объектов. Результат первого шага представляет собой набор векторов признаков, полученных с помощью окон нразличных размеров. На втором шаге выполняется непосредственно построение иерархического леса

- структуры, где каждый уровень получает информацию о признаках, сформированных на предыдущем уровне, и передает результат их обработки на следующий уровень.

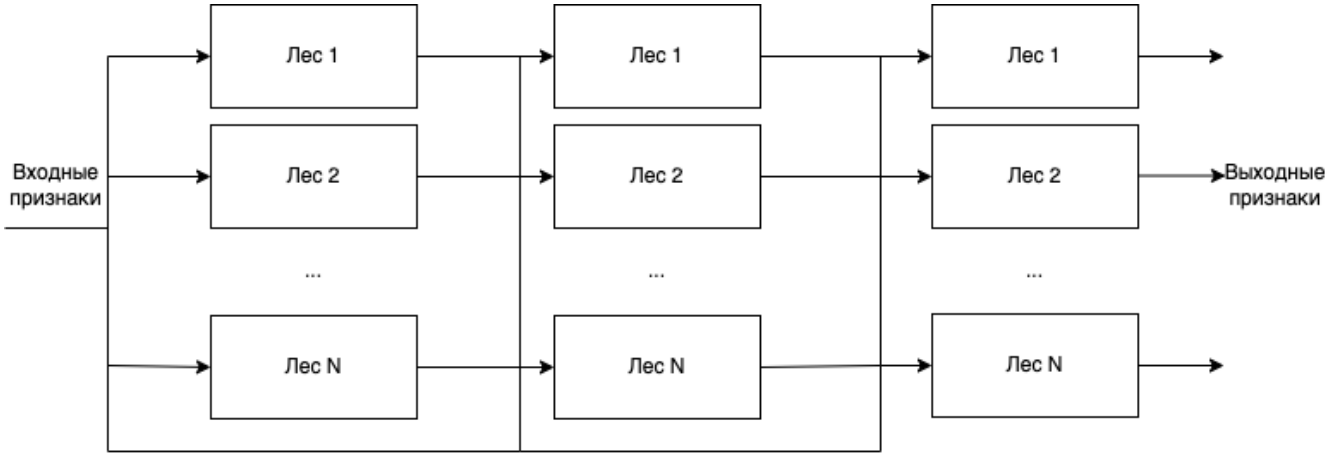

Рисунок 1.11 — Архитектура иерархической композиции классификаторов на основе деревьев решений, предложенной Zhou и др.

Использование вектора признаков, формируемого в процессе классификации, на более верхних уровнях композиции очень похоже на идею, лежащую в основе метода стекинга классификаторов [36]. Алгоритм начинается с обучения классификаторов первого уровня, используя исходный набор обучающих данных. Затем генерируется новый набор данных для обучения классификатора второго уровня (мета-классификатора), так что результаты классификаторов первого уровня рассматриваются как входные признаки для классификаторов второго уровня, в то время как метки классов все еще рассматриваются как классы для новых обучающих данных. В отличие от алгоритма стекинга, gcForest одновременно использует и исходный вектор признаков объектов и векторы, сгенерированные классификаторами более верхнего уровня. На последнем уровне формируется такое представление признаков входных объектов, которое можно классифицировать, чтобы получить окончательный результат.

В работе [86] была предложена модификация этого метода (Transfer Learning Deep Forest,TLDF), позволяющая при обучении каскадных классификаторов на основе случайных лесов деревьев решений применять трансдуктивное обучение.

Вместе с тем, для Deep Forest характерны те же недостатки, что и для деревьев решений с одномерными разделителями и их лесов: учет ограниченного количества признаков, невысокая вычислительная эффективность при работе

с данными, имеющими большую размерность признакового пространства, ограниченные возможности по учету взаимного влияния признаков [87].

Для снятия этих недостатков в настоящей диссертации предложен метод классификации объектов сложной структуры, в котором интегрируются случайные леса деревьев решений, имеющих ядерные разделители, и подход к построению их иерархических композиций (stack), предложенный в [85]. Этот метод позволяет создавать многослойные классификаторы произвольной сложности и обобщающей способности.

1.4 Системы распределенного обучения деревьев решений и их композиций

В области разработки методов и систем распределенного построения деревьев решений и их ансамблей выделяется четыре основных направления работ: создание универсальных распределенных систем машинного и глубокого обучения, создание методов и систем обучения распределенных деревьев решений, разработка систем для параллельного обучения ансамблей деревьев решений методом градиентного бустинга [39], создание распределенных случайных лесов деревьев решений. В основе универсальных распределенных систем машинного и глубокого обучения, как правило, лежат готовые платформы и библиотеки для организации распределенных вычислений: Apache Spark, Apache Kafka, Kubernetes, Hadoop. Так разработчики системы MLBase [88] реализовали механизм распределенного выполнения на основе библиотек Apache Spark [89] и Mesos [90]. MLbase предназначен для распределенного обучения различных моделей с учетом ограничений их применимости. В системе Pregel [91] для организации параллельной распределенной обработки данных используется массовая синхронная параллельная модель вычислений (Bulk Synchronous Parallel, BSP). В работе показано, что эта масштабируемая и отказоустойчивая система может быть запущена на кластерах из нескольких тысяч вычислительных узлов. Система Distributed GraphLab [92] — это распределённая реализация GraphLab [93] (в которой параллельная обработка данных была реализована только для машин с общей памятью). В Distributed GraphLab отказоустойчивость обеспечивается за счет использования двух ал-

горитмов получения глобального состояния системы: синхронного алгоритма и асинхронного алгоритма Чанди-Лампорта. Несколько исследований посвящено применению метода MapReduce для распределенной реализации алгоритмов машинного обучения. Так в работе [94] показано, что MapReduce позволяет организовать распределенное и масштабируемое машинное обучение при условии применения ограниченного набора методов и алгоритмов.

Ряд работ посвящен распределенным методам построения деревьев решений. Например, в библиотеке PLANET [95] реализовано распределенное построение деревьев решений с применением подхода MapReduce. В состав библиотеки включен модуль планирования, который преобразует отдельные шаги построения дерева решений в задания MapReduce, которые могут выполняться параллельно. Каждая задача по построению разделителя дерева решений отображается в одно задание MapReduce. Затем эти задания выполняются одновременно. Для снижения затрат, связанных с запуском заданий MapReduce используется механизм планирования, выполняющий подготовку заданий в фоновом режиме. В работе [96] представлена распределенная платформа машинного обучения для обработки больших массивов данных (Scalable Advanced Massive Online Analysis, SAMOA). Платформа позволяет обучать распределенные деревья решений, при этом обеспечивается производительность, сравнимая с централизованными системами машинного обучения. В SAMOA для каждого узла распределенной системы строится статистика по данным, сохраненным на этом узле. Далее эта статистика обобщается центральным узлом и используется для обновления параметров разделителей обучаемого дерева решений. В ходе экспериментальных исследований разработанной системы выявлено, что она позволяет эффективно обрабатывать массивы данных с высокой размерностью признакового пространства, однако необходима доработка, направленная на снижение задержек при взаимодействии компонентов системы.

Большую практическую значимость имеет также направление исследований, связанное с созданием систем и методов обучения ансамблей лесов деревьев решений с помощью алгоритма градиентного бустинга. Особенностью бустинга является итеративный характер формирования ансамбля, что ограничивает его применимость для обучения на больших размеченных выборках. В статье [97] предложен метод распределенного построения ансамблей деревьев решений методом градиентного бустинга. В этом методе данные распределяются по узлам системы на основе значения их атрибутов. В хо-

де экспериментальных исследований предложенного метода показано, что он позволяет обрабатыватьс большие наборы данных быстрее, чем последовательная версия алгоритма бустинга. В статье [98] предложен метод обучения ансамблей деревьев решений методом стохастического градиентного бустинга (EFLSGB). Экспериментальные исследования предложенного метода показали, что EFLSGB обеспечивает достаточную точность классификации по сравнению с последовательной версией алгоритма. В EFLSGB параллельно выполняется несколько последовательных алгоритмов стохастического градиентного бустинга. Оптимальное количество деревьев в каждом алгоритме определяется с помощью процедуры перекрёстного скользящего контроля. В статье [99] представлена масштабируемая библиотека для построения ансамблей методом бустинга.

Масштабируемость XGBoost обусловлена несколькими архитектурными решениями и алгоритмическими оптимизациями. Эти решения включают алгоритм обработки разреженных данных при обучении деревьев и процедуру приближенного обучения деревьев. Самый трудоемкий этап обучения деревьев решений – сортировка обучающих данных по значению определенных признаков. Чтобы снизить стоимость сортировки, предлагается хранить данные в блоках. Каждый блок соответствует подмножеству объектов в обучающем наборе данных. Для хранения данных в блоках используются разреженные матрицы (Compressed Sparse Column, CSC), при этом каждый столбец сортируется по значению соответствующего признака. Это представление входных данных формируется перед обучением, и далее повторно используется в последующих итерациях. Применение подобной структуры данных позволяет свести поиск разделителей к перебору отсортированных столбцов матрицы.

В работе [100] представлен алгоритм распределенного построения ансамбля деревьев решений методом градиентнного бустинга в системе CatBoost. В этой библиотеке используется распределение данных по узлам вычислительной системы на основе значений признаков. В работе [101] предлагается схема параллельной реализации бустинга для классификации объектов с большим числом категорий. Обучаемый ансамбль состоит из нескольких последовательностей деревьев решений заданной длины. В каждой из таких последовательностей оценивается вероятность принадлежности объекта обучающей выборки к некоторому классу, при этом построение каждой последовательности может быть выполнено параллельно. После построения очередного дерева решений работа

всех процессов синхронизируется, и каждый процесс рассылает всем остальным данные о текущих предсказаниях соответствующего ему ансамбля на объектах обучающей выборки, которые необходимы для выполнения следующей итерации алгоритма. По окончанию построения деревьев производится пересылка всех полученных последовательностей на главный процесс, где происходит построение модели.

В работе [102] предложен распределенный алгоритм построения леса деревьев решений. В этом алгоритме для снижения объема передачи информации признаковое пространство разбивается на подмножества признаков и, аналогично [100], обучающие данные распределяются между узлами вычислительной системы на основе значений признаков из этих подмножеств. Затем генерируются статистические показатели для каждого подмножества и, наконец, вычисляются значения порогов разделителей. Согласно приведенному обзору, проводимые исследования сфокусированы на создании распределенных алгоритмов и систем обучения одномерных деревьев решений. В этом случае построение разделителей деревьев решений может быть выполнено централизованно на основе статистической информации о распределении значений признаков обучающих данных на отдельных узлах системы. Вместе с тем, при построении лесов деревьев решений с линейными или нелинейными разделителями необходимо учитывать исходные значения признаков обучающей выборки. Поэтому, актуальной является разработка методов и систем распределенного обучения деревьев решений, в которых построение разделителей происходило бы на основе исходных обучающих данных.

1.5 Выводы к главе

1. Деревья решений с линейными (и нелинейными) разделителями позволяют моделировать сложные зависимости, но поиск вычислительно-эффективного способа их обучения остается открытой проблемой из-за негладкой функции потерь, возникающей при попытке описать связь между положением разделяющей гиперплоскости и каким-либо из критериев построения разделителей (критерий Джини, информационная энтропия и др.).

2. Деревья решений с линейными разделителями нестабильны, поэтому они перспективны лишь в составе композиций типа "бэггинг "собэггинг"или "случайный лес которые имеют лучшую стабильность, чем входящие в них базовые классификаторы.
3. Алгоритмы формирования структуры деревьев должны быть квазиоптимальными, чтобы повысить случайность формируемых деревьев.
4. При построении узлов деревьев решений должен оптимизироваться отступ между данными в поддеревьях и заданные критерии неоднородности данных в поддеревьях.
5. Обучение узлов деревьев решений должно сводиться к решению задачи оптимизации гладкой выпуклой функции потерь с ограничениями-неравенствами.
6. Узлы деревьев решений должны выполнять классификацию объектов в спрямляющих признаковых пространствах.
7. Для нахождения путей создания и регуляризации случайных композиций нужны более детализированные подходы к оценке обобщающей способности, чем те, что предложены в работах Бреймана (математическое ожидание и корреляция отступов) или для бэггина/собэггинга (стабильность базовых классификаторов).
8. В случае создания эффективного способа обучения композиций деревьев решений с линейными разделителями, естественным подходом к его применению для решения задачи классификации объектов сложной структуры стало бы построение иерархических композиций, подобных DeepForest.
9. Актуальной становится разработка архитектур систем распределенного обучения деревьев решений, в которых построение разделителей происходило бы на основе исходных обучающих данных, а не собранной статистики на уровне отдельных вычислительных узлов.

Глава 2. Метод построения деревьев решений с линейными и нелинейными разделителями и их композиций

2.1 Деревья решений с линейными и нелинейными разделителями

2.1.1 Основные определения

Зададим $H = \{h_1,h_2,\ldots,h_\Gamma\}$– множество длины Γ листовых функций $s = h_i(x,y)$ которые отображают объекты и их метки на закрытый вещественный промежуток $\mathbb{R}^f \times U \to [0..1]$, где Γ – количество листьев в дереве. Пусть все функции из множества H таковы, что:

$$h_{i_{D_{m_i}}}(x,y) = \frac{1}{|D_{m_i}|}|\{(\tilde{x},\tilde{y})|(\tilde{x},\tilde{y}) \in D_{m_i} \vee \tilde{y} = y\}| \in H, \tag{2.1}$$

то есть возвращают эмпирическую оценку вероятности того, что объект x с меткой y принадлежит листу с индексом i, и D_{m_i} – подмножество обучающего набора D_m, относящееся к листу с индексом i. Для упрощения нотации будем далее обозначать $h_{i_{D_{m_i}}}(x,y)$ как $h_{D_i}(x,y)$. Зададим $Q = \{q_1,q_2,\ldots,q_n\}$ – множество алгоритмов выбора узла $s = q_{j_{D_{m_j}}}(x)$, обученных на подмножествах $D_{m_j} \subset D_m$. Для упрощения нотации будем далее обозначать $q_{j_{D_{m_j}}}(x)$ как $q_{D_j}(x)$. Алгоритм $q_{D_j}(x)$ возвращает 1 если x принадлежит к узлу j, и 0 в противном случае.

Цепочкой решений для листа с индексом i будем считать произведение результатов всех алгоритмов выбора узла, соответствующих пути от корня до этого листа i, и значения функции выбора листа i. (см. рис. 2.1).

Потребуем, чтобы цепочки удовлетворяли следующим свойствам:

1. Существует строгий порядок обучения алгоритмов выбора узла в цепочке: $(i < j) \vee (q_{D_i}(x), q_{D_j}(x) \in Q) \to D_j \subset D_i$
2. Каждый алгоритм выбора узла в цепочке определяет какое подмножество обучающих данных будет использовано при обучении следующего узла в цепочке: $i < j \vee ((x_1,y_1),(x_2,y_2) \in D_j) \vee (q_{D_i}(x), q_{D_j}(x) \in Q) \to q_{D_i}(x_1)q_{D_i}(x_2) > 0$
3. Каждый алгоритм выбора узла в цепочке определяет какое подмножество обучающих данных не будет учитываться при обучении узла в

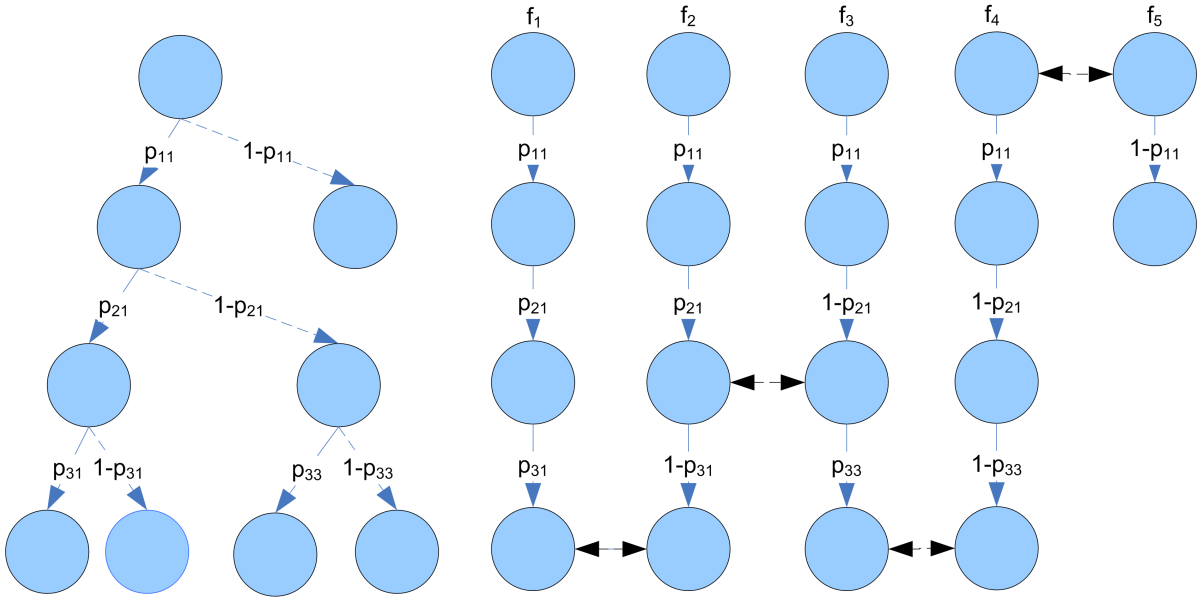

Рисунок 2.1 — Дерево решений высотой 3 и его представление в виде цепочек решений

цепочке: $i < j \vee ((x_1,y_1) \in D_i, (x_2,y_2) \in D_i \setminus D_j) \vee (q_{D_i}(x), (q_{D_j}(x) \in Q) \rightarrow q_{D_i}(x_1)q_{D_j}(x_2) = 0$

Пусть $\Omega = \{Q_1,Q_2,..,Q_\Gamma\}$ – набор множеств алгоритомов выбора узла. Бинарное дерево – это кортеж $< H,\Omega,f >$, где f имеет вид:

$$f_{D_m}(x,y) = \sum_{i=1}^{\Gamma} h_{D_i}(x,y) \prod_{j=1}^{|Q_i \in \Omega|} q_{D_{ij}}(x), \tag{2.2}$$

где $q_{D_{ij}}$ обозначает алгоритм выбора узла q_{D_j} из множества Q_i

Положим существует T множеств листовых функций $\mathbf{H} = \{H_1,..,H_T\}$ и T наборов множеств алгоритомов выбора узла $\mathbf{\Omega} = \{\Omega_1,\Omega_2,..,\Omega_T\}$. Бэггингом называется кортеж $< \mathbf{H},\mathbf{\Omega},F >$, где F имеет вид:

$$F_{D_{m(\mathbf{r})}}(x,y) = \frac{1}{T} \sum_{i=1}^{T} \sum_{j=1}^{\Gamma_i} h_{D_{ij(\mathbf{r_{ij}})}}(x,y) \prod_{k=1}^{|Q_j \in \Omega_i|} q_{D_{ijk(\mathbf{r_{ij}})}}(x), \tag{2.3}$$

где $h_{D_{ij}}$ обозначает листовую функцию h_{D_j} из множества $H_i, q_{D_{ijk}}$ обозначает алгоритм выбора узла q_{D_k} из набора Q_j, входящего в множество $\Omega_i, \mathbf{r} = \mathbf{r_{11}},\mathbf{r_{12}},..,\mathbf{r_{\Gamma_T T}}$ – случайные параметры выбора обучающих данных для цепочек решений, $D_m(\mathbf{r})$ – набор обучающих данных, выбранных с использованием случайного параметра $\mathbf{r}$, T количество деревьев в ансамбле, Γ_i количество листьев в дереве с индексом i.

2.1.2 Метод обучения деревьев решений с линейными и нелинейными разделителями

Для построения деревьев решений используется стандартный рекурсивный алгоритм (Алгоритм 1) [5, 6]. На каждом шаге алгоритма строится узел дерева, который разделяет обучающие данные, затем эта процедура рекурсивно повторяется для «левого» и «правого» подмножеств обучающих данных, пока не будет достигнута заданная глубина дерева.

Алгоритм 1 Обучение дерева с линейными и нелинейными разделителями.

Вход: Набор данных D_m, параметр регуляризации C, J порог выбора способа распределения классов по поддеревьям (точный/жадный), K – количество запусков жадной процедуры распределения классов по поддеревьям,N - размер списка лучших разбиений для случайного выбора распределения классов по поддеревьям.

1: **call** BuildTree(D_m):
2: BuildTree(D):
3: **if** D содержит один класс **then return**
4: **else**
5: **if** |U| < J **then**
6: $s := all\_distributions(U, H = \{+1, -1\})$
7: $H_{best} := sort(Impurity(s))[rnd(1..N)]$ ▷ $U \to H$
8: **else**
9: $H_{best} := greedy\_find\_best(D)$ ▷ $U \to H$
10: **end if**
11: $L_1^*, \ldots, L_m^* := L(h_i, -h_i), h_i \in H_{best}$.
12: $w^*, \varepsilon_1^*, \varepsilon_m^* := optimize\_node(C, X, L_1^*, \ldots, L_m^*)$ ▷ Решить задачу оптимизации SVM с масштабированием переменных невязки: $\varepsilon_1/L_1^*, \ldots, \varepsilon_m/L_m^*$.
13: $D_l := D[classify(D, w^*) \geqslant 0]$
14: $D_r := D[classify(D, w^*) < 0]$
15: BuildTree(D_l)
16: BuildTree(D_r)
17: **end if**

Пусть P_X - распределение объектов, P_Y - распределение меток этих объектов, P_{XY} - совместное распределение анализируемых данных. Рассмотрим построение разделителя как задачу обучения двоичного классификатора на наборе из m объектов $D_m = \{< x_i,y_i > | i = 1,\ldots,m; x_i \in X_m, y_i \in Y_m\}, X_m \sim P_X, Y_m \in P_Y, D_m \in P_{XY}$ с метками классов y_i, которые выбираются из множества U. Пусть этот двоичный классификатор распределяет объекты по поддеревьям (левому и правому), и $H = \{-1, +1\}$ - метки этих поддеревьев.

На первых шагах алгоритма (шаги 6-10) устанавливается целевое распределение классов в поддеревьях. Некоторые критерии могут разделять объекты одного класса по разным поддеревьям, но в предложенном методе эта особенность игнорируется в угоду скорости обучения. Если количество классов $|U|$ превышает пороговое значение J, являющееся гиперпараметром алгоритма, то перебираются все возможные распределения классов по поддеревьям и вычисляются значения соответствующего критерия неоднородности (шаги 7–8). Затем полученный список вариантов распределений сортируется по критерию неоднородности, и целевое распределение c_s выбирается случайным образом среди первых N элементов отсортированного списка. Если количество классов $|U|$ больше, чем J, то для выявления целевого распределения классов по поддеревьям применяется жадная процедура (шаг 10). Процедура начинается с генерации случайного распределения классов по поддеревьям. Затем сгенерированное распределение итеративно изменяется и сохраняются модификации, улучшающие критерий неоднородности данных. Эта процедура повторяется K раз, затем выбирается наилучшее отображение $c_s : U \to H$. В результате в процесс обучения узла дерева решений добавляется рандомизация. Затем алгоритм использует полученное распределение c_s для определения поддерева для каждого объекта из обучающего набора данных $D_m : H_{best} = c_s(Y_m)$.

Следующие шаги алгоритма (шаги 11-12) состоят в обучении разделителя в узле дерева решений. В процессе обучения одновременно оптимизируется отступ в узле и критерий неоднородности. Для вычислительно-эффективного построения разделителей, аналогично методам обучения SVM классификации структур, задается непрерывная гладкая функция потерь, которая отражает зависимость между параметрами разделителя и значением критерия неоднородности данных. Обучение разделителя на наборе данных из m обучающих примеров производится путем решения следующей задачи оптимизации:

$$w^*,\varepsilon^* = \arg\min_{w,\varepsilon} \frac{1}{2}||w||^2 + \frac{C}{m}\sum_{i=1}^{m}\varepsilon_i, \tag{2.4}$$

при условии:

$$\forall i, w^T x h_i \geqslant 1 - \frac{\varepsilon_i}{L(h_i, -h_i)}, \tag{2.5}$$

где w – параметры разделяющей гиперплоскости; $\varepsilon_1, \ldots, \varepsilon_m$ – переменные невязки; w^* - оптимальные значения параметров разделяющей гиперплоскости; ε^* - оптимальные значения переменных невязки; – параметр регуляризации, а $L(h_i, -h_i)$ отражает прирост критерия неоднородности данных в случае отнесения объекта i к некорректному поддереву $-h_i$ вместо целевого поддерева h_i (см. Рис. 1).

Если применить к выражению 2.4 условия Каруша-Куна-Таккера, то получится двойственная задача оптимизации [103]:

$$\tilde{a} = \arg\max_{a} -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j K(x_i,x_j) + \sum_{i=1}^{m} a_i, \tag{2.6}$$

при условии:

$$\sum_{i=1}^{m} \frac{a_i}{L(h_i, -h_i)} \leqslant \frac{C}{m}, \tag{2.7}$$

где a_i – вес примера i из обучающей выборки (отличный от нуля для опорных векторов), а $K(x_i,x_j)$ – положительно определенное ядро [47]. В отличие от классификации структур, эта задача эффективно решается в явном виде, поскольку классов всего два (два поддерева). Гиперпараметр регуляризации C должен быть подобран эмпирически для каждого набора данных. Шаги 13–16 алгоритма направлены на классификацию всех обучающих данных по поддеревьям с помощью построенного разделителя и рекурсивного обучения новых узлов дерева разделению обучающих данных поддеревьев: D_l, D_r.

Наконец, на основе деревьев, обученных с помощью представленного алгоритма, с использованием метода [43] формируется случайный лес.

Покажем, как решение задач 2.4 или 2.6 позволяет оптимизировать критерий неоднородности данных. Пусть целевое значение критерия неоднородности данных достигается, если эмпирические оценки вероятностей отнесения объекта обучающей выборки к левому и правому поддеревьям равны соответственно

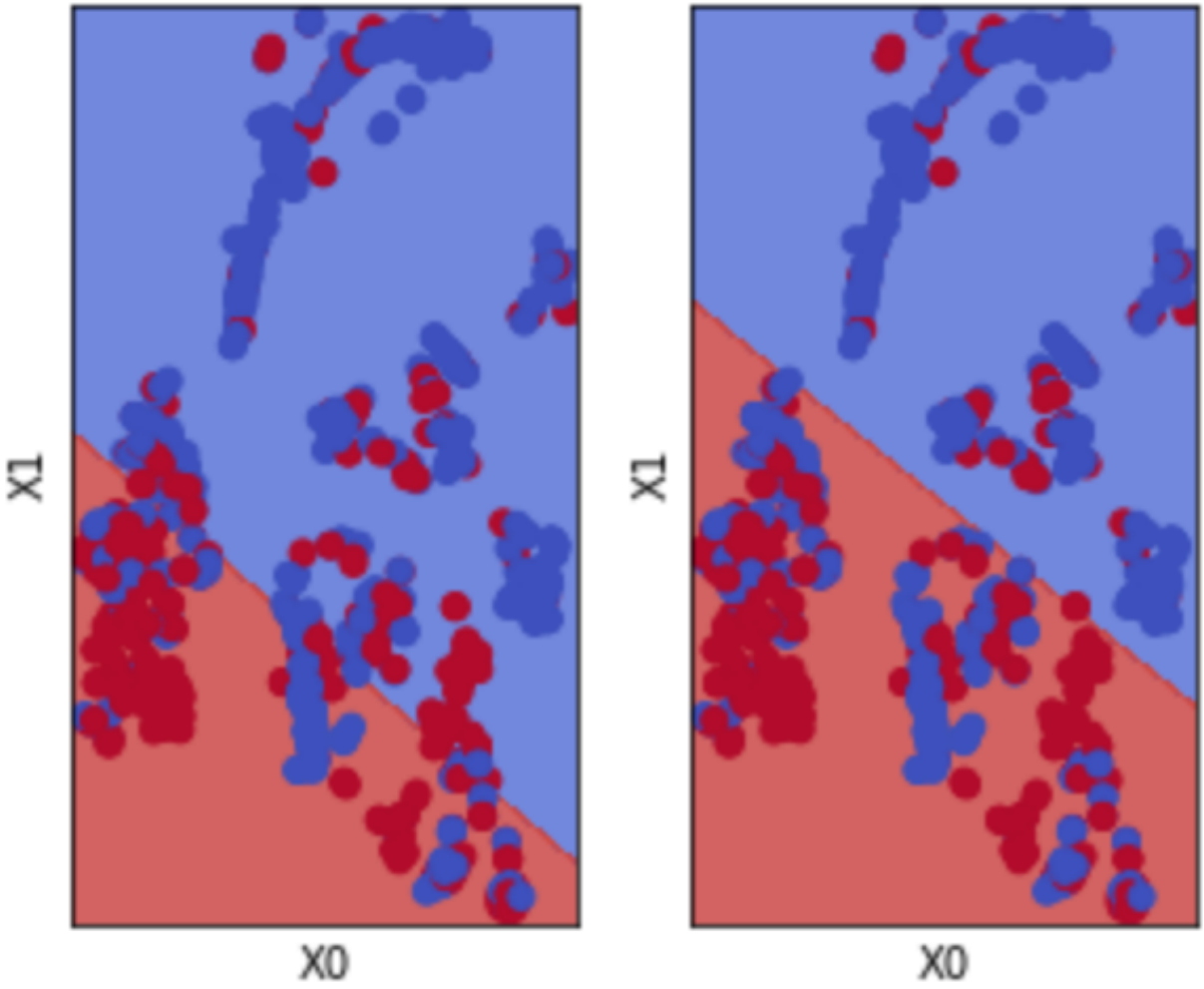

Рисунок 2.2 — Влияние масштабирования переменных невязки $\varepsilon_1, \ldots, \varepsilon_m$ на результаты построения разделяющей гиперплоскости. Без масштабирования (слева), с масштабированием – справа. Набор данных Titanic, критерий – неоднородность Джини.

P_L и P_R, а эмпирические оценки вероятностей классов в левом и правом поддеревьях равны $\mathbf{p_L} = \{p_{L_1}, \ldots, p_{L_{|U|}}\}$ и $\mathbf{p_R} = \{p_{R_1}, \ldots, p_{R_{|U|}}\}$. Зададим

$$s_{best} = \arg\min \left(P_L g(\mathbf{p_L}) + P_R g(\mathbf{p_R}) \right) \tag{2.8}$$

где s_{best} – целевое значение критерия неоднородности данных (в случае, если данные могут быть разделены без ошибок согласно заданному отображению c_s), $g(\mathbf{p})$ - неоднородность Джини, информационная энтропия, или другой критерий. Пусть w_{best} – параметры (возможно, недостижимые) разделяющей гиперплоскости, которая соответствует значению критерия неоднородности s_{best}.

Предположим, что полученный разделитель относит некоторый элемент к поддереву h^* вместо h. Измененные оценки вероятностей (по сравнению с оценками для лучшего разбиения $P_L, P_R, \mathbf{p_L}, \mathbf{p_r}$) можно обозначить $P_L^*, P_R^*, \mathbf{p_L^*}, \mathbf{p_r^*}$ и прирост критерия неоднородности данных равен:

$$L(h^*,h) = P_L^* g(\mathbf{p_L^*}) + P_R^* g(\mathbf{p_R^*}) - P_L g(\mathbf{p_L}) - P_R g(\mathbf{p_R}) \tag{2.9}$$

Зададим $\hat{R}(w) = \frac{1}{m}\sum_1^m L(h_i,x_i^T w)$ – эмпирический риск на обучающей выборке. В работе [104] приводятся доказательства того, что решение задач 2.4 или 2.6 минимизирует эмпирический риск $\hat{R}(w)$. Стоит отметить, что эмпирический риск $\hat{R}(w)$ не является тождественным критерию неоднородности данных. Эмпирический риск может быть переформулирован как $\hat{R}(p_{err}) = \sum_{u\in U} p_{err}(u) L(c_s(u), -c_s(u))$, где $p_{err}(u)$ возвращает долю некорректно распределенных объектов класса u (тех, что по итогам классификации распределяются по поддеревьям не в соответствии с $_s$). Все рассматриваемые критерии неоднородности данных (Джини, прирост информации, информационная энтропия и др.) являются выпуклыми функциями от вероятностей классов в поддеревьях. Можно преобразовать $L(c_s(u), -c_s(u))$ в функцию потерь от изменений вероятностей классов по сравнению с лучшим разбиением $_s$ (см. выражение 2.9). Функция $L(h_i,sign(x_i^T w))$ возвращает значение частной производной критерия неоднородности данных в точке, соответствующей значению критерия s_{best}. График функции $\hat{R}$ является касательной к графику критерия неоднородности в точке s_{best}, которой соответствуют параметры разделяющей гиперплоскости w_{best}. При минимизации 2.4 производится спуск по этой касательной настолько близко к точке, соответствующей s_{best} и w_{best}, насколько позволяют распределение классифицируемых объектов в признаковом пространстве и выбранная функция ядра (см. рис. 2.3).

Теоремой 1 (глава 1) гарантируется, что для заданного отображения классов на поддеревья разделяющая гиперплоскость будет построена так, чтобы минимизировать $\hat{R}(w)$ (см. рис. 2.2). Теоретически для минимизации критерия неоднородности необходимо рассмотреть все отображения $U \to H$. Однако, было решено отказаться от достижения глобального оптимума, ввиду того что сам алгоритм построения деревьев является жадным. Такой подход позволяет дополнительно рандомизировать структуру деревьев, что положительно сказывается на обобщающей способности случайных лесов, формируемых из них.

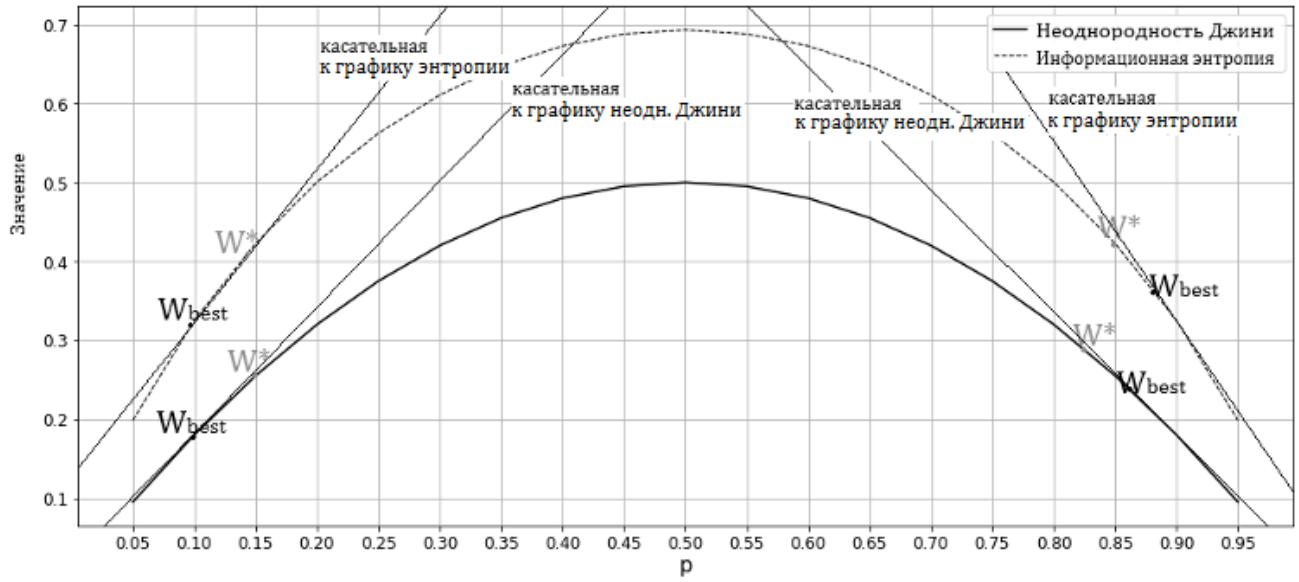

Рисунок 2.3 — Критерии построения разделителей и эмпирический риск $\hat{R}(w)$. Точке w^* соответствуют параметры разбиения, достижимые на практике на анализируемых данных.

2.1.3 Вычислительная сложность алгоритма обучения дереьев с многомерными разделителями

Для деревьев решений с одномерными разделителями сложность обучения можно оценить следующим образом. Сбалансированные деревья решений содержат не более $log(m)$ узлов, сложность обучения каждого узла определяется необходимостью перебора f признаков и m обучаюших примеров, чтобы подобрать признак и порог, оптимизирующий снижение критерия неоднородности данных. Таким образом, вычислительная сложность обучения сбалансированного дерева решений с одномерными разделителями составляет $O(mflog(m))$.

При обучении деревьев решений с линейными и нелинейными разделителями на верхнем уровне применяется тот же рекурсивный алгоритм, что и для стандартных деревьев. Поэтому сложность обучения узла в этом случае зависит от типа разделителя (линейный или нелинейный) и от метода оптимизации, используемого для обучения разделителя. В случае линейного разделителя можно использовать метод покоординатного спуска [34]. Этот метод имеет линейную сложность относительно количества обучающих примеров.

Следовательно, оценка сложности такая же, как и для деревьев, с одномерными разделителями составит $O(mflog(m))$.

В случае нелинейных ядер одним из применимых подходов к обучению является метод структурной минимальной оптимизации (SMO). Вычислительная сложность этого метода составляет $O(m^2f)$ [105]. Поэтому итоговая оценка сложности является полиномиальной, что может являться проблемой при обучении на больших наборах данных $O(m^2flog(m))$.

2.2 Теоретическая оценка обобщающей способности композиций деревьев решений с линейными и нелинейными разделителями

Для оценки обобщающей способности композиций алгоритмов, формируемых с помощью предложенных алгоритмов обучения, был использован подход, основанный на равномерной стабильности (устойчивости) [5]. Вывод оценки обобщающей способности случайных ансамблей деревьев решений состоит из следующих шагов:

1. Вывод оценки равномерной стабильности алгоритма обучения цепочек решений.
2. Вывод оценки равномерной стабильности алгоритма обучения случайного ансамбля деревьев решений (как линейной композиции зависимых цепочек решений).
3. Вывод оценки обобщающей способности случайного ансамбля деревьев решений, зависящей от равномерной стабильности алгоритма обучения этого ансамбля.

2.2.1 Равномерная оценка стабильности деревьев решений и их композиций типа бэггинг

Для оценки равномерной стабильности алгоритма обучения цепочек решений получим сначала выражение для более общего случая – произведения

результатов конечного числа произвольных алгоритмов классификации, обученных с использованием равномерно стабильных алгоритмов.

Лемма 1. *(Девяткин). Пусть* $Pr(x) = \prod_{q \in G} q(x)$ *- произведение результатов всех алгоритмов классификации* q *с областью значений* $[0..1]$ *из множества* G *мощности* $n \in \mathbb{N}^+$*. Эти алгоритмы классификации обучены с помощью алгоритмов, оптимизирующих некоторые B-Липшицевы функции потерь. Положим, что эти алгоритмы* γ_q*-равномерно стабильны. Тогда равномерная стабильность алгоритма обучения этого произведения* $Pr(x)$ *ограничена сверху следующим образом:*

$$\gamma_{Pr} \leqslant \frac{1}{n}\left(\sum_{i=1}^{n-2}\left(\frac{B}{2}\right)^{i} + 2\left(\frac{B}{2}\right)^{n-1}\right)\sum_{q \in G}\gamma_q = \varphi(B,n)\sum_{q \in G}\gamma_q \tag{2.10}$$

Доказательство. Рассмотрим сначала композицию, представляющую собой произведение двух алгоритмов $Pr_(x) = q_{1_D}(x)q_{2_D}(x)$:

$$\begin{gathered}\gamma_{Pr} = \sup_D |l(q_{1_D}(x)q_{2_D}(x),y) - l(q_{1_{D^{\setminus i}}}(x)q_{2_{D^{\setminus i}}}(x),y)|, \\ \forall i \in 1,\ldots,|D|, \forall x \in X.\end{gathered} \tag{2.11}$$

Так как функция потерь - *B-Липшицева*, это выражение можно записать следующим образом:

$$\gamma_{Pr} \leqslant B \sup_D |q_{1_D}(x)q_{2_D}(x) - q_{1_{D^{\setminus i}}}(x)q_{2_{D^{\setminus i}}}(x)|, \tag{2.12}$$

Так как $q_1(x)$ и $q_2(x)$ принимают значения в диапазоне $[0..1]$ при любых x, то $q_1(x)q_2(x) \leqslant q_1(x) + q_2(x)$.

$$\begin{gathered}\gamma_{Pr} \leqslant \frac{B}{2}\sup_D\left(|(q_{1_D}(x) + q_{2_D}(x)) - (q_{1_{D^{\setminus i}}}(x) + q_{2_{D^{\setminus i}}}(x))|\right) \\ \leqslant \frac{B}{2}(\gamma_1 + \gamma_2),\end{gathered} \tag{2.13}$$

что завершает доказательство.

Рассмотрим теперь произведение n алгоритмов. Его можно представить как $q = q_{i_1}\prod_{j \neq i_1} q_j$ и, в соответствии с доказанным выше выражением,

$$\gamma_n \leqslant \frac{B}{2}(\gamma_{i_1} + \gamma_{j \neq i_1}). \tag{2.14}$$

Из-за коммутативности произведения i_1 может быть любым натуральным числом из интервала $[1,\ldots,n]$, все варианты выбора равнозначны. Произведение

$\prod_{j\neq i_1} q_j$ также может быть декомпозировано: $\prod_{j\neq i_1} q_j = q_{i_2} \prod_{j\neq i_1,i_2} q_j$. Процесс декомпозиции повторяется до тех пор, пока в произведении остается более одного множителя. В итоге равномерная оценка стабильности алгоритма построения композиции будет ограничена следующим образом:

$$\beta_n \leqslant \frac{B}{2}\gamma_{i_1} + (\frac{B}{2})^2\gamma_{i_2} + \cdots + (\frac{B}{2})^{n-1}\gamma_{i_{n-1}} + (\frac{B}{2})^{n-1}\gamma_{i_n} \tag{2.15}$$

Можно задать вероятностное пространство (Ω,$\mathcal{A}$,P) и на нем дискретную случайную величину $\Psi : (\Omega,\mathcal{A}) \to (\mathbb{R},\mathcal{B})$, такую, что $\Psi(\omega) = \omega, \forall\omega \in \Omega$, где $\Omega = \{1,2,\ldots,n\}$, $\mathcal{A}$ - алгебра на Ω, $\mathcal{B}$ - борелевская σ-алгебра. Эта случайная величина задает для каждого β_i позицию в правой части выражения 2.15. Благодаря коммутативности, для любого значения $i \in \mathcal{B}$ вероятность составит:

$$P(i \in \Psi) = \frac{1}{n} \tag{2.16}$$

Очевидно, что P удовлетворяет свойствам вероятности. Теперь можно оценить математическое ожидание правой части выражения (2.15):

$$\begin{aligned} \gamma_n &\leqslant \Big(\sum_{i=1}^{n-2}(\frac{B}{2})^i P(i) + (\frac{B}{2})^{n-1}P(n-1) + (\frac{B}{2})^{n-1}P(n)\Big)\sum_{i=1}^{n}\gamma_i \\ &\leqslant \frac{1}{n}\Big(\sum_{i=1}^{n-2}(\frac{B}{2})^i + 2(\frac{B}{2})^{n-1}\Big)\sum_{j=1}^{n}\gamma_j, \end{aligned} \tag{2.17}$$

что соответствует (2.10). □

Следствие 1. *(Девяткин). Для указанного выше множества G равномерная стабильность алгоритма обучения произведения $Pr(x)$ асимптотически ограничена сверху равномерной стабильностью алгоритмов обучения элементов G: $\gamma_{Pr} = O(\gamma_q)$.*

Доказательство. Правая часть выражения 2.17 при фиксированном n является произведением оценки равномерной стабильности алгоритов обучения, входящих в G, на константу $C \sim \varphi(B,n)$: $\gamma_{Pr} = C\gamma_q$. Однако, по определению $O()$, $|\gamma_{Pr}| \leqslant |C\gamma_q|$, что доказывает следствие. □

Следствие 2. *(Девяткин). Пусть соблюдаются свойства 1-3 цепочек решений. Алгоритмы обучения узлов в цепочке решений равномерно стабильны: $\gamma = O(\frac{1}{m})$. $P_{D(j)} : \mathbb{N} \to [0..1]$ - вероятность того, что элементы из набора*

$Dj-1$ представлены также в наборе Dj, тогда равномерная стабильность алгоритма обучения цепочки решений длины $n \in \mathbb{N}^+$ ограничена сверху:

$$\gamma_{Pr} \leqslant \varphi(B,n) \sum_{q \in G} \frac{\gamma_q}{\prod_{j=1}^{i} P_D(j)} \tag{2.18}$$

Доказательство. Согласно свойствам цепочек узловых классификаторов, в ходе построения дерева решений каждый последующий узел j цепочки функций выбора листа обучается на подмножестве данных, на которых обучался предыдущий узел $j-1$, то есть $D_j \subset D_{j-1}$.

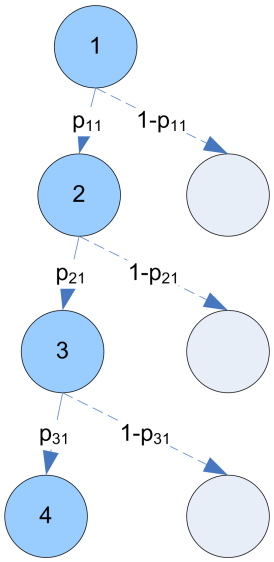

Рисунок 2.4 — Цепочка функций выбора листа высотой 3

Пусть вероятность того, что элементы из множества D_{j-1} попадут в множество D_j задается функцией $P_D(j) : \mathbb{N} \to [0..1]$, а равномерная оценка стабильности корневого узла цепочки не превышает $\gamma = O\left(\frac{1}{m}\right)$. Тогда равномерная оценка стабильности для такой цепочки составит.

$$\begin{aligned} \gamma_{Pr} &\leqslant \frac{1}{n}\left(\sum_{i=1}^{n-2}\left(\frac{B}{2}\right)^i + 2\left(\frac{B}{2}\right)^{n-1}\right) \sum_{i=1}^{n} \frac{\gamma}{\prod_{j=1}^{i} P_D(j)} \\ &= \varphi(B,n) \sum_{i=1}^{n} \frac{\gamma}{\prod_{j=1}^{i} P_D(j)} \end{aligned} \tag{2.19}$$

□

Для метода опорных векторов равномерная оценка стаблильности зависит от параметра регуляризации: $\beta_{SVM} = \frac{\theta^2 C}{2m}$ (где C - параметр регуляризации и

$\Phi(x,x) \leqslant \theta^2$ - ограничение области значения функции ядра), поэтому верхняя граница оценки стабильности для цепочки узловых классификаторов, обученных с помощью SVM будет равна:

$$\beta_{SVM} = \varphi(B,n) \sum_{i=1}^{n} \frac{\theta^2 C}{2m \prod_{j=1}^{i} P_D(j)} \tag{2.20}$$

Следствие 3. *(Девяткин). Пусть соблюдаются свойства 1-3 цепочек решений (см. раздел 2.1), а алгоритмы обучения узлов в цепочке решений равномерно стабильны, тогда алгоритм обучения цепочки решений является равномерно стабильным.*

Доказательство. По определению равномерно стабильного алгоритма, $\gamma_q = O\left(\frac{1}{m}\right)$. По определению цепочки решений, $\forall i, P_D(i) > 0$ (в противном случае цепочка укорачивается). Согласно следствию 1, $\gamma_q = O\left(\frac{1}{m}\right)$, тогда $\gamma_{Pr} = O\left(O(\frac{1}{m})\right) = O(\frac{1}{m})$. □

К сожалению, при оценке случайной равномерной стабильности композиции деревьев решений нельзя рассматривать как композиции независимых цепочек узловых классификаторов,так как параметры $\mathbf{r_t}$ цепочек, входящих в каждое дерево решений, зависимы друг от друга.

Зададим смягченный набор условий:

1. Пусть $\mathcal{R} = \{0,1\}^m$. Алгоритм использует случайные векторы $\mathbf{r} = \{\mathbf{r_1},\mathbf{r_2},\ldots,\mathbf{r_T} | r_t \in \mathcal{R}\}$ для выбора объектов из D_m, для обучения алгоритма f_t: 1 если соотв. объект из D_m используется, 0 - иначе. $D(\mathbf{r_t})$ - набор данных, сгенерированный для алгоритма f_t.
2. Несколько копий одного и того же объекта не влияют на результат обучения.

Введем случайную равномерную оценку стабильности композиции деревьев решений с многомерными разделителями, как композиции зависимых цепочек функций выбора листа. Согласно [41] выражение для оценки равномерной стабильности рандомизированного алгоритма моожно записать следующим образом.

$$\beta_m \leqslant \frac{B}{T} \sum_{t=1}^{T} M_{\mathbf{r_t}}(\gamma_{\mathbf{r_t}} | 1_{i \in \mathbf{r_t}}) \tag{2.21}$$

Важно отметить, что это выражение не накладывает ограничений на зависимость параметров $\mathbf{r_t}$.

Для простоты будем оценивать стабильность построения композиции из T деревьев, каждое из которых состоит из Γ цепочек высотой n. Каждая цепочка содержит узловые классификаторы, построенные алгоритмом с равномерной оценкой стабильности γ.

$$\beta_m \leqslant \frac{B}{T}\sum_{t=1}^{T}\sum_{j=1}^{\Gamma} M_{\mathbf{r_{tj}}}(\gamma_{\mathbf{r_{tj}}}|1_{i\in\mathbf{r_{tj}}}) = \\ = \frac{B}{T}\sum_{k=1}^{m}\sum_{t=1}^{T}\sum_{j=1}^{\Gamma} P_{\mathbf{r_{tj}}}(d(\mathbf{r_{tj}})=k)\gamma_k M_{\mathbf{r_{tj}}}(1_{i\in\mathbf{r_{tj}}}; d(\mathbf{r_{tj}})=k) \tag{2.22}$$

Вычислим математическое ожидание $M_{\mathbf{r_{tj}}}(1_{i\in\mathbf{r_{tj}}}; d(\mathbf{r_{tj}})=k)$ - вероятность события, состоящего в присутствии изъятого элемента i в обучающих данных в некоторой цепочке узловых классификаторов, при условии, что алгоритмом для обучения этой цепочки было отобрано k различных объектов из набора данных размерности m.

Введем событие A, состоящее в присутствии элемента i в обучающих данных в некоторой цепочке узловых классификаторов, событие A_1, состоящее в присутствии этого элемента при обучении корневого узла, A_2 - присутствии этого элемента при обучении следующего узла, A_n - наличии этого элемента при обучении узла n. Напомним, что согласно свойствам цепочек, при обучении узлов в цепочке каждый следующий узел обучается на подмножестве данных, на которых обучался предыдущий узел: $D_n \subset \cdots \subset D_2 \subset D_1$, поэтому $A_n \subset \cdots \subset A_2 \subset A_1$. Отсюда следует, что $P(\bigcup_{i=1}^{n} A_i) = P(A_1)$ - искомая вероятность и $P(A_1) = \frac{k}{m}$.

Теперь подствляя выражение для равномерной стабильности алгоритма построения цепочки 2.19 получим.

Теорема 5. *(Девяткин). Пусть ансамбль из T деревьев решений обучается методом бэггинга на наборе данных D_m размера m, каждое дерево из ансамбля i содержит Γ_i цепочек решений длины n_j. Пусть для обучения узлов деревьев используются алгоритмы с равномерной стабильностью γ, которые оптимизируют B-Липшицеву функцию потерь. Пусть выполняются смягченные условия 1 и 2. Тогда равномерная стабильность алгоритма бэггинга β_m будет ограничена сверху:*

$$\beta_m \leqslant \frac{B}{T}\sum_{k=1}^{m}\sum_{i=1}^{T}\sum_{j=1}^{\Gamma_i}\gamma_{Pr}(i,j,k)\frac{k}{m}P_{\mathbf{r_{ij}}}(d(\mathbf{r_{ij}})=k) \tag{2.23}$$

где $\gamma_{Pr}(i,j,k) = \varphi(B,n_j)\sum_{l=1}^{n_j}\frac{\gamma_k}{\prod_{t=1}^{l}P_{D(\mathbf{r_{ij}})(t)}}$, $d(\mathbf{r_{ij}})$ *- кол-во различных элементов в* $D(r_{ij})$, $P_{\mathbf{r_{ij}}}(d(\mathbf{r_{ij}})=k)$ *- вероятность того, что в наборе данных* $D(r_{ij})$ *ровно* k ***различных*** *элементов.*

Если $\gamma_k = O\left(\frac{1}{k}\right)$, то есть базовый алгоритм обучения элементов композиции равномерно стабилен, сумма для β_m минимальна в случае, если распределение, задаваемое для разных цепочек и деревьев выражением $\prod_{t=1}^{l}P_{D(\mathbf{r_{ij}})}(t)$ равномерно.

Как отмечалось в работе [41], с ростом количества деревьев $T\to\infty$ параметр k стремится к своему математическому ожиданию $k=(1-\frac{1}{e})m\approx 0.632m$:

$$\beta_m \leqslant 0.632B\varphi(B,n)\sum_{i=1}^{\Gamma}M_{\mathbf{r_i}}\left(\sum_{l=1}^{n}(\gamma_{0.632m\prod_{t=1}^{l}P_{D(\mathbf{r_i})}(t)})\right) \tag{2.24}$$

То есть для каждой ветки деревьев в композиции средняя вероятность ее выбора будет стремиться к некоторому математическому ожиданию, вследствие чего сглаживаются "выбросы"в отдельных деревьях композиции, содержащих ветки с очень низкой или высокой вероятностью выбора (См. рис 2.5).

Для определения скорости сходимости этой средней вероятности (а следовательно - равномерной стабильности алгоритма построения композиции) к математическому ожиданию необходимо использовать следующее неравенство Мак-Диармида [106].

Теорема 6. *(McDiarmid, 1998). Пусть* $X=(X_1,\ldots,X_n)$ *- семейство случайных независимых величин, принимающих значения из множества* A_k *и вещественная функция* f*, определенная на множестве* $\prod A_k$ *удовлетворяет ограничению* $|f(x)-f(x')|\leqslant c_k$*, где вектора* x *и* x' *отличаются только одним элементом номер* k*. Пусть* μ *- математическое ожидание случайной величины* $f(X)$*. Тогда для любого* $t>0$*:*

$$P(f(X)-\mu\geqslant t)\leqslant \exp^{-2t^2/\sum c_k^2} \tag{2.25}$$

$$P(f(X)-\mu\leqslant -t)\leqslant \exp^{-2t^2/\sum c_k^2} \tag{2.26}$$

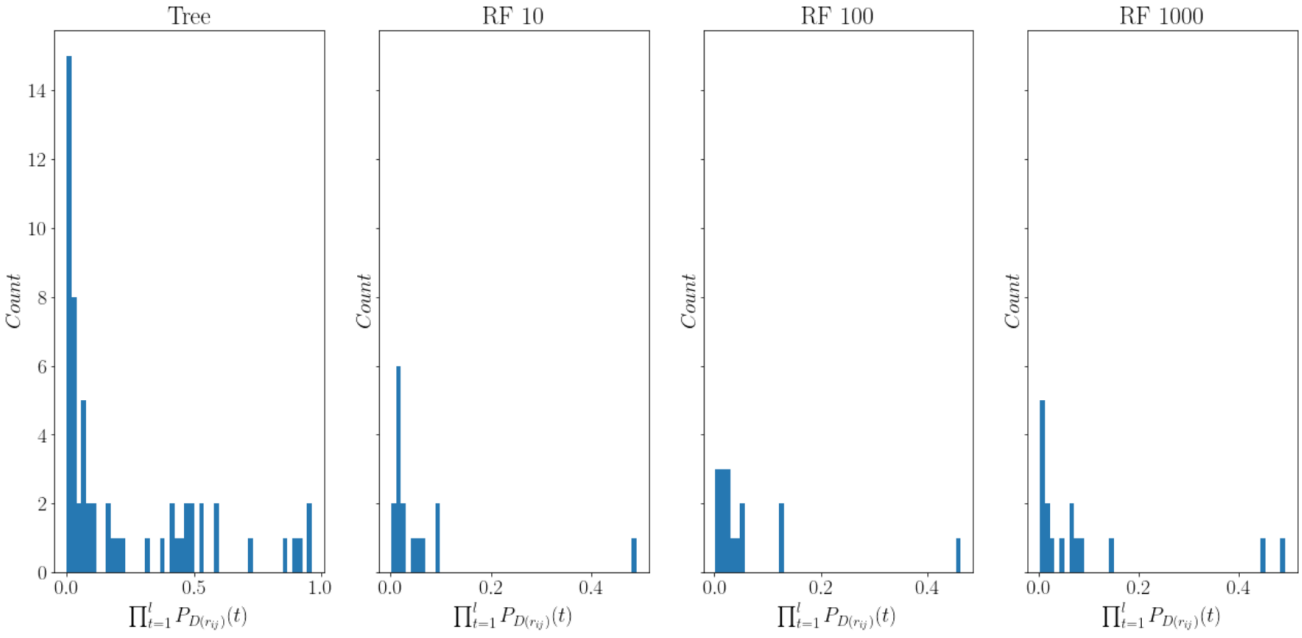

Рисунок 2.5 — Гистограмма вероятностей $\prod_{t=1}^{l} P_{D(r_{ij})}(t)$ для одиночного дерева решений и композиции типа "бэггинг"с различным количеством деревьев решений

$$P(|f(X) - \mu| \leqslant t) \leqslant 2\exp^{-2t^2/\sum c_k^2} \tag{2.27}$$

Следствие 4. *(Девяткин) Равномерная стабильность алгоритма бэггинга сходится с увеличением количества деревьев к своему математическому ожиданию* μ *с экспоненциальной скоростью. То есть:*

$$P(|\beta_m - \mu| \leqslant t) \leqslant 2\exp^{-\frac{T\Gamma t^2}{\left(\gamma_{k/m} - \gamma_{k(m-1)/m}\right)^2}} = O\left(\frac{1}{\exp^T}\right) \tag{2.28}$$

Этот результат согласуется с многократно подтвержденным эмпирически тезисом о бессмысленности построения бэггинга на 1000 и более деревьев.

Доказательство. В теореме 5 величина $\prod_{t=1}^{l} P_{D(\mathbf{r_{ij}})}(t)$ может находиться на отрезке $[\frac{1}{m}..\frac{m-1}{m}]$. Рассматривать случай, когда вероятность перехода по цепочке равна 0 не имеет смысла, так как такая цепочка может быть сведена к более короткой. В ином же случае после перехода по цепочке остается хотя бы один элемент данных, что дает вероятность $\frac{1}{m}$. Следуя аналогичным рассуждениям эта величина не превысит $\frac{m-1}{m}$.

Тогда изменение равномерной стабильности алгоритма построения цепочки конечно и не превысит величины $|\gamma_{k/m} - \gamma_{k(m-1)/m}| \leqslant c_k$. Значит,

согласно Теореме 6, равномерная оценка стабильности алгоритма построения композиции деревьев решений сходится к своему математическому ожиданию с экспоненциальной скоростью. □

2.2.2 Оценка обобщающей способности композиций деревьев решений типа бэггинг

Теорема 3 может быть применена только для оценки обобщающей способности композиций, построенных случайными алгоритмами, в которых параметры построения независимы для каждого элемента в композиции. В случае использования формализма 2.3, то есть представления композиции деревьев решений в виде набора цепочек классификаторов, это условие нарушается. Введем модифицированную версию этой оценки, в которой независимость параметров построения элементов композиции является необязательной. Для вывода оценки будет использовано следующее неравенство МакДиармида для зависимых случайных величин [106]. Пусть $X = (X_1, \ldots, X_n)$ - семейство случайных величин, принимающих значения из множества A_k и функция f - ограниченная вещественная функция, определенная на множестве $\prod A_k$. Для всех $x \in A_k$ зададим функцию

$$g(x) = M(f(X)|B, X_k = x) - M(f(X)|B), \tag{2.29}$$

где B - событие, заключающееся в том, что $X_i = x_i$ для каждого $i = 1, \ldots, k-1$.

Обозначим:

$$var(x_1, \ldots, x_{k-1}) = var_{x,y \in A_k} |g(x) - g(y)| \tag{2.30}$$

$$\hat{v} = \sup_{\forall x \in \prod A_k} \sum_{k=1}^{n} (var(x_1, \ldots, x_{k-1})) \tag{2.31}$$

Теорема 7. *(McDiarmid, 1998). Пусть $X = (X_1, \ldots, X_n)$ - семейство случайных величин, принимающих значения из множества A_k и функция f - ограниченная вещественная функция, определенная на множестве $\prod A_k$. Пусть μ - математическое ожидание $f(X)$, $b = sup\{g(x)|x \in \prod A_k\}$, а $\hat{v}$ определена в соответствии с выражением 2.31. Тогда для любого $t \geqslant 0$:*

$$P(f(X) - \mu \geqslant t) \leqslant \exp^{-\frac{t^2}{2\hat{v}(1+(bt/3\hat{v}))}} \tag{2.32}$$

Теорема 8. *(Девяткин). Пусть выполняются условия 1 и 2 из заданного выше набора условий,* $F_{D(\mathbf{r})}(x) = \frac{1}{T}\sum_{t=1}^{T\Gamma} f_{D(\mathbf{r_t})}(x)$ *- ансамбль деревьев решений, построенный алгоритмом бэггинга на н.о.р. наборе данных* D_m *размером* m*. Каждое дерево ансамбля имеет* Γ *цепочек решений и равномерная стабильность алгоритма бэггинга ограничена сверху величиной* β_m*.*

Тогда с вероятностью не менее $1-\delta$ *верно неравенство.*

$$R(F_{D(\mathbf{r})}) \leqslant \hat{R}(F_{D(\mathbf{r})}) + 2\beta_m + \frac{1}{2}B\left(ln(\frac{1}{\delta})\frac{2M}{3} + \sqrt{\left(ln(\frac{1}{\delta})\frac{2M}{3}\right)^2 + 8ln\frac{1}{\delta}\left(B^2m\left(\beta_m\right)^2 + \frac{(2M)^2\Gamma^3}{T}\right)}\right) \tag{2.33}$$

Доказательство. Для упрощения записи в доказательстве будем использовать "плоскую"индексацию цепочек решений в композиции. Пусть в композици содержится $T\Gamma$ цепочек решений.

Пусть разность между ошибкой и эмпирическим риском композиции задается следующей случайной функцией:

$$K(D,\mathbf{r}) = R(F_{D(\mathbf{r})}) - \hat{R}(F_{D(\mathbf{r})}), \tag{2.34}$$

где $D = \{D_1,\ldots,D_m\}$ обучающая выборка размера m и $\mathbf{r} = \{\mathbf{r_1},\ldots,\mathbf{r_{T\Gamma}}\}$ множество случайных двоичных векторов,использующихся как параметры выбора данных при обучении разделителей. Важно отметить, что некоторые случайные векторы в $\mathbf{r}$ зависят друг от друга. Чтобы получить верхнюю грань ошибки обучения необходимо оценить верхнюю грань случайной функции $K(D,\mathbf{r})$. Для оценки отклонения значений K от её математического ожидания на $\mathbf{r}$ and D применяется неравенство МакДиармида (Теорема 7). Раскроем K в сотвествии с определениями $\hat{R}$ и R.

$$\begin{gathered} K(D,\mathbf{r}) = \\ M_{XY}(l(F_{D(\mathbf{r})}(X),Y)) - \frac{1}{m}\sum_{i=1}^{m} l(F_{D(\mathbf{r})}(x_i),y_i) \leqslant \\ \left|M_{XY}(l(F_{D(\mathbf{r})}(X),Y))\right| + \left|\frac{1}{m}\sum_{i=1}^{m} l(F_{D(\mathbf{r})}(x_i),y_i)\right|, \end{gathered} \tag{2.35}$$

где $x_i, y_i \in D$.

Так как $l(F_{D(\mathbf{r})}(x),y)$ является B-Липшицевой, можно записать:

$$K(D,\mathbf{r}) \leqslant 2B \sup_X \left| F_{D(\mathbf{r})}(X) \right| \tag{2.36}$$

По определению цепочки решений $F_{D(\mathbf{r})}(x) > 0$, тогда:

$$K(D,\mathbf{r}) \leqslant 2B \sup_X F_{D(\mathbf{r})}(X) \tag{2.37}$$

Пусть $\mathfrak{R}_{k-1}$ обозначает событие, когда $\mathbf{r_1} = r_1,\ldots,\mathbf{r_{k-1}} = r_{k-1}$, и $\mathfrak{D}_{k-1}$ - это событие, когда $D_1 = d_1,\ldots,D_{k-1} = d_{k-1}$, $\mathfrak{R}$ обозначает $\mathfrak{R}_{T\Gamma}$ и $\mathfrak{D}$ обозначает $\mathfrak{D}_m$. Пусть $\mathbf{D} = d_1,\ldots,d_{k-1},D_k,\ldots,D_m$ и $\mathbf{D}^{\backslash\mathbf{k}\cup\mathbf{d}} = d_1,\ldots,d_{k-1},d,D_{k+1},\ldots,D_m$.

Теперь рассмотрим следующие функции:

$$g_1(d,k) = M_{D,\mathbf{r}}(K(D,\mathbf{r})|\mathfrak{D}_{k-1},D_k = d) - M\mathfrak{D}_{k-1}(K(D,\mathbf{r})|\mathfrak{D}_{k-1}) \tag{2.38}$$

и

$$g_2(r,k) = M_{\mathbf{r}}(K(D,\mathbf{r})|\mathfrak{D},\mathfrak{R}_{k-1}\mathbf{r_k} = r) - M_{\mathbf{r}}(K(D,\mathbf{r})|\mathfrak{D},\mathfrak{R}_{k-1}) \tag{2.39}$$

Значение

$$\begin{aligned}\hat{v} = \sup_{\substack{r_1,\ldots,r_k-1,\\ d_1,\ldots,d_{k-1},\\ x}} \Bigg(\sum_{k=1}^{m} var_(g_1(d,k)) + \\ \sum_{k=1}^{T\Gamma} var(g_2(r,k)) \Bigg) = \\ \sup_{\substack{d_1,\ldots,d_{k-1},\\ x}} \Bigg(\sum_{k=1}^{m} var_(g_1(d,k)) \Bigg) + \\ \sup_{\substack{r_1,\ldots,r_k-1,\\ d_1,\ldots,d_{k-1},\\ x}} \Bigg(\sum_{k=1}^{T\Gamma} var(g_2(r,k)) \Bigg)\end{aligned} \tag{2.40}$$

является верхней гранью суммы дисперсий этих двух функций g_1 и g_2. Точнее говоря, эти выражения соответствуют дисперсии ошибки обучения, обусловленной обучающими данными (для g_1) и случайными параметрами выборки этих данных (для g_2).

Дисперсия величины g_1 ограничена следующим образом.

$$\sum_{k=1}^{m} \Bigg(var_(g_1(d,k)) = \sum_{k=1}^{m} var(M_{D,\mathbf{r}}(K(D,\mathbf{r})|\mathfrak{D}_{k-1},\mathbf{d_k} = d) - M_{D,\mathbf{r}}(K(D,\mathbf{r})|\mathfrak{D}_{k-1})) \Bigg) \tag{2.41}$$

Математическое ожидание $M_{D,\mathbf{r}}(K(D,\mathbf{R})|\mathfrak{D}_{k-1})$ для фиксированного значения $\mathfrak{D}_{k-1}$ не влияет на дисперсию. Известно также, что $D_1,\ldots,D_m$ независимы, кроме того $\mathbf{r}$ не зависит от D и наоборот.

Тогда, для фиксированного $\mathfrak{D}_{k-1}$ будет соблюдаться:

$$M_{D,\mathbf{r}}(K(D,\mathbf{r})|\mathfrak{D}_{k-1},\mathbf{d_k}=d)=M_{D,\mathbf{r}}(K(\mathbf{D}^{\setminus\mathbf{k}\cup\mathbf{d}},\mathbf{r})) \tag{2.42}$$

В соответствии с Теоремой 6 верно следующее неравенство.

$$\begin{gathered}\sum_{k=1}^{m}\sup_{\substack{d_1,\ldots,d_{k-1},\\ x}} var_(g_1(d,k))\leqslant \\ 4B^2\sum_{k=1}^{m}\sup_{\substack{d_1,\ldots,d_{k-1},\\ x}}\Big(M_d\big(M_d(M_{D,\mathbf{r}}(F_{\mathbf{D}^{\setminus\mathbf{k}\cup\mathbf{d}}(\mathbf{r})}(x,y)))- \\ M_{D,\mathbf{r}}(F_{\mathbf{D}^{\setminus\mathbf{k}\cup\mathbf{d}}(\mathbf{r})}(x,y))\big)\Big)\leqslant B^4 m\left(\beta_m\right)^2\end{gathered} \tag{2.43}$$

Теперь рассмотрим $g_2(r,k)$. Объединяя выражения (2.37) и $F_{D(\mathbf{r})}(x,y)=\sum_{i=1}^{T\Gamma} f_{D(\mathbf{r})}(x,y)$ получаем:

$$\begin{gathered}\sum_{k=1}^{T\Gamma} var(g_2(r,k))\leqslant \\ \left(\frac{2B}{T}\right)^2\sum_{k=1}^{T\Gamma}\Big(var(\sum_{i=1}^{T\Gamma}M_{\mathbf{r_i}}(f_{D(\mathbf{r_i})}(x,y)|\mathfrak{D},\mathfrak{R}_{k-1},\mathbf{r_k}=r)- \\ M_{\mathbf{r_i}}(f_{D(\mathbf{r_i})}(x,y)|\mathfrak{D},\mathfrak{R}_{k-1}))\Big)\end{gathered} \tag{2.44}$$

Отметим, что разность между математическими ожиданиями ненулевая только для цепочек решений, которые зависят от $\mathbf{r_k}$. Обозначим множество номеров таких цепочек как ρ_k. Заметим, что для фиксированного $\mathfrak{R}_{k-1}$ и $\mathfrak{D}$ математическое ожидание $M_{r_i}(f_{D(\mathbf{r_i})}(x,y)|\mathfrak{D},\mathfrak{R}_{k-1}))$ - константа, не влияющая на дисперсию. Тогда получаем:

$$\begin{gathered}\sum_{k=1}^{T\Gamma} var(g_2(r,k))\leqslant \\ \left(\frac{2B}{T}\right)^2\sum_{k=1}^{T\Gamma}\Big(var(\sum_{i\in\rho_k}M_{\mathbf{r_i}}(f_{D(\mathbf{r_i})}(x,y)|\mathfrak{D},\mathfrak{R}_{k-1},\mathbf{r_k}=r))\Big)\end{gathered} \tag{2.45}$$

Размер множества ρ_k не может превзойти количество листьев соответствующего дерева (так как зависимыми могут быть только цепочки, входящие в одно и то же дерево). Обозначим максимальное количество листьев у деревьев ансамбля как $\Gamma = \sup_k |\rho_k|$. Для упрощения рассуждений, будем полагать, что все деревья в ансамбле имеют одинаковое количество листьев. Значение $M_{r_i}(f_{\mathbf{r_i},D}(x,y)|\mathfrak{D},\mathfrak{R}_{k-1},\mathbf{r_k} = r)$ ограничено максимальным значением функции f, которое не превышает M; таким образом, можно записать:

$$\sum_{k=1}^{T\Gamma} var(g_2(r,k)) \leqslant \left(\frac{2B}{T}\right)^2 \sum_{k=1}^{T\Gamma} \left(M\Gamma\right)^2 = \frac{(2BM)^2\Gamma^3}{T} \tag{2.46}$$

Теперь применим Теорему 7:

$$P_{D,\mathbf{r}}\left(K(D,\mathbf{r}) - M_{D,\mathbf{r}}(K(D,\mathbf{r})) \geqslant t)\right) \leqslant exp(-\frac{t^2}{2\hat{v}(1+(bt/3\hat{v}))}), \tag{2.47}$$

где $b = BM$. Обозначим правую часть неравенства как δ выразим из нее t:

$$\delta = exp(-\frac{t^2}{2\hat{v}(1+(bt/3\hat{v}))}) \tag{2.48}$$

Так как $t > 0$:

$$t = \frac{1}{2}\left(ln(\frac{1}{\delta})\frac{2BM}{3} + \sqrt{\left(ln(\frac{1}{\delta})\frac{2BM}{3}\right)^2 + 8ln\frac{1}{\delta}\hat{v}}\right) \tag{2.49}$$

С вероятностью $1-\delta$ относительно D и $\mathbf{r}$:

$$K(D,\mathbf{r}) - M_{D,\mathbf{r}}(K(D,\mathbf{r})) \leqslant \frac{1}{2}\left(ln(\frac{1}{\delta})\frac{2BM}{3} + \sqrt{\left(ln(\frac{1}{\delta})\frac{2BM}{3}\right)^2 + 8ln\frac{1}{\delta}\hat{v}}\right) \tag{2.50}$$

Следующие шаги полностью аналогичны доказательству Теоремы 3.4 из работы [41]. Оценим математическое ожидание $K(D,\mathbf{r})$ на D и r.

$$M_{D,\mathbf{r}}(K(D,\mathbf{r})) = 2\beta_m \tag{2.51}$$

Затем подставим его в выражение (2.50):

$$R(F_{D(\mathbf{r})} - \hat{R}(F_{D(\mathbf{r})} - 2\beta_m \leqslant \frac{1}{2}\left(ln(\frac{1}{\delta})\frac{2BM}{3} + \sqrt{\left(ln(\frac{1}{\delta})\frac{2BM}{3}\right)^2 + 8ln\frac{1}{\delta}\hat{v}}\right) \tag{2.52}$$

В итоге, с вероятностью не менее $1-\delta$ будет соблюдаться следующее:

$$R \leqslant \hat{R} + 2\beta_m + \frac{1}{2}B\left(ln(\frac{1}{\delta})\frac{2M}{3} + \sqrt{\left(ln(\frac{1}{\delta})\frac{2M}{3}\right)^2 + 8ln\frac{1}{\delta}\left(B^2m\left(\beta_m\right)^2 + \frac{(2M)^2\Gamma^3}{T}\right)}\right) \tag{2.53}$$

что и требовалось доказать. □

В полученном выражении величина $B^4m\left(\beta_m\right)^2$ является верхней гранью дисперсии ошибки, связанной с данными (см. доказательство). Эта величина зависит от равномерной стабильности алгоритма обучения случайного ансамбля, то есть «сглаживание» распределения вероятностей переходов от корня к узлам деревьев решений леса приводит к снижению дисперсии ошибки. Величина $\frac{(2BM)^2\Gamma^3}{T}$ является верхней гранью дисперсии ошибки, связанной с параметрами выборки данных $\mathbf{r}$, эта величина снижается с ростом размера ансамбля T и увеличивается с ростом количества листьев у деревьев решений Γ.

2.2.3 Регуляризация лесов решающих деревьев с ядерными разделителями

На основе полученных оценок (Теоремы 5 и 8) можно сформулировать следующие неформальные критерии подбора метода регуляризации случайного ансамбля деревьев решений:

– Сокращение общего числа цепочек (листьев) в деревьях.

– Сокращение длины цепочек решений.

– Сокращение цепочек решений с низкими вероятностями перехода.

Большая часть подходов к регуляризации случайных ансамблей состоит в искусственном внесении случайных различий между деревьями композиции. Вместе с тем, подобные подходы могут отрицательно повлиять на смещение ошибки. S. Ren и др. [107] предложили отбросить эмпирические оценки вероятностей классов, хранящиеся в листьях деревьев предварительно обученного случайного леса, и переопределить их путем оптимизации некоторой общей функции потерь композиции. Однако такой подход делает деревья в композиции зависимыми. В настоящем исследовании предлагается модификация метода усиления, позволяющая найти баланс между глобальной оптимизацией векторов листьев и независимостью деревьев внутри ансамбля.

Пусть $\Phi : \mathbb{R}^{\mathrm{f}} \to \{0,1\}^{T\Gamma}$ - функция, которая для каждого обучающего примера x возвращает двоичный вектор, чьи элементы равны 1 если x попадает в соответствующий лист и 0 в обратном случае.

$$\Phi_r(x) = (\varphi_1(x)\mathbf{r_1},\varphi_2(x)\mathbf{r_2},\ldots,\varphi_{T\Gamma}(x)\mathbf{r_{T\Gamma}}), \tag{2.54}$$

где $\mathbf{r_1}, \mathbf{r_2}, \ldots, \mathbf{r_{T\Gamma}}$ - н.о.р. случайные величины, принимающие значения из множества $\{0, 1\}$. Вероятность того, что эти величины принимают нулевое значение является гиперпараметром в предложенной модификации и позволяет имитировать отсутствие отдельных обучающих примеров при глобальной оптимизации функции потерь композиции.

Матрица W содержит переопределенные веса классов в листьях деревьев решений.

$$W = (w_1,w_2,\ldots,w_{T\Gamma}) \tag{2.55}$$

Для получения новых значений оптимизируется глобальная функция потерь, сходная с применяемой в методе опорных векторов.

$$y = W^*\Phi(x) \tag{2.56}$$

где W^* может быть найдена путем оптимизации следующего выражения:

$$\begin{gathered} W^* = \arg\min_W \frac{1}{2}||W||_F^2 + \frac{C}{m}\sum_{i=1}^{m} l(y_i,\hat{y}_i), \\ y_i = W\Phi(x), \forall i \in [1,m] \end{gathered} \tag{2.57}$$

Оптимизация глобальной функции потерь может привести к переобучению ансамбля. Для преотвращения этого эффекта используется прунинг,

который состоит в соединении соседних листьев деревьев, в случае, когда норма разности векторов этих листьев (векторов переопределенных эмпирических вероятностей классов) близка к нулю. Эта операция приводит к уменьшению количества цепочек решений, уменьшению их длины, повышению вероятности переходов от корня к узлам деревьев, то есть выполняются все предложенные критерии подбора метода регуляризации, при этом за счет глобальной оптимизации функции потерь не происходит увеличения эмпирического риска.

2.3 Метод классификации структур с помощью композиции деревьев решений с ядерными разделителями

Метод представляет собой модификацию метода DeepForest [85], в которой обработка данных происходит последовательно на нескольких слоях.

Основными отличиями модифицированного метода от оригинального DeepForest являются [5, 9]:

- Использование случайных лесов деревьев решений с нелинейными разделителями в качестве базового алгоритма классификации, что позволяет учитывать связи между признаками анализируемых объектов, уменьшить количество слоев обработки данных и, следовательно, повысить производительность.
- Отказ от применения экстремально случайных лесов деревьев решений (Extremely Random Forests), вместо них для повышения обобщающей способности используются усиление и прунинг из раздела 2.4, что позволяет повысить стабильность работы метода.

Входной слой в модифицированном методе имеет следующую структуру (рис. 2.6). На основе каждого объекта из обучающего набора методом «скользящего окна» генерируется набор примеров, помеченных классом исходного примера [11]. Эти примеры используются для обучения случайного леса деревьев решений с линейными и нелинейными разделителями, далее выполняется усиление и прунинг обученного леса. После процедуры прунига лес используется для формирования векторных представлений (эмбеддингов) обрабатываемых объектов для следующего слоя. Эти представления включают

исходные признаки объектов, а также векторы эмпирических вероятностей, возвращаемые деревьями леса.

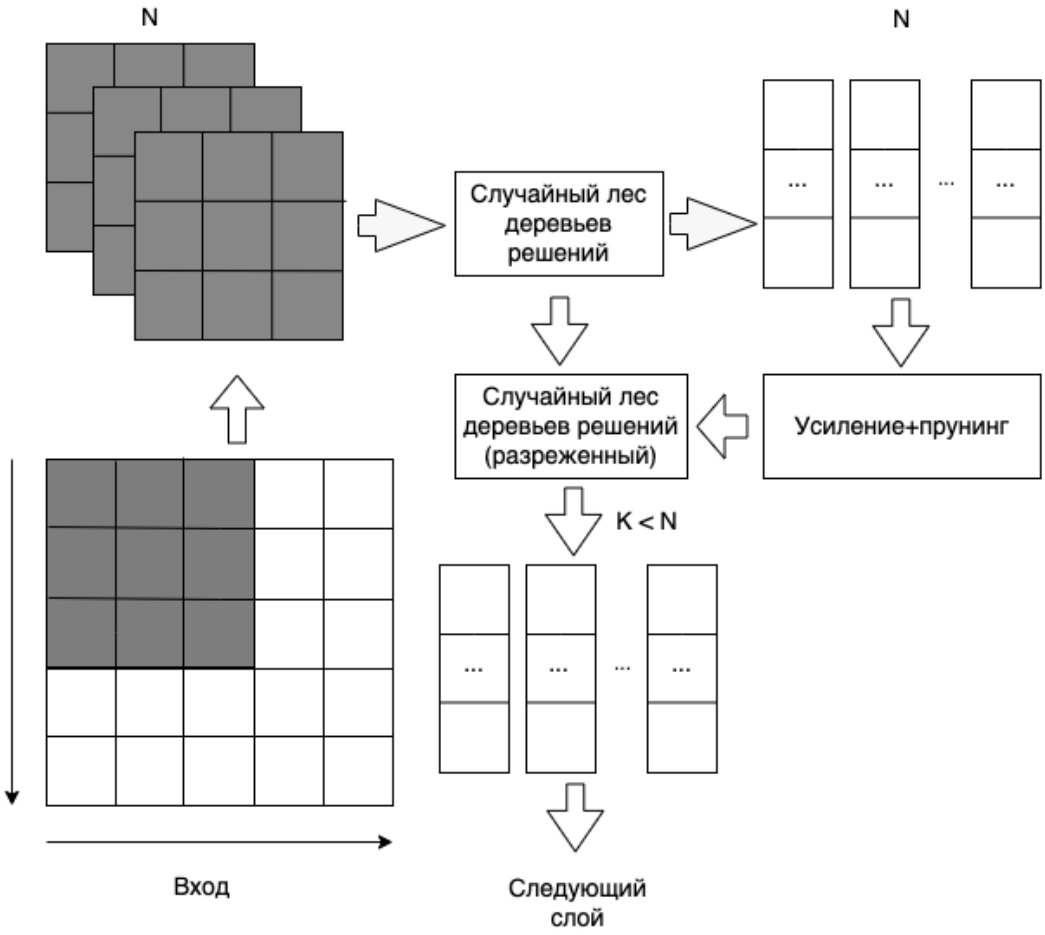

Рисунок 2.6 — Схема метода классификации объектов, характеризующихся наличием связей между признаками (один слой)

На практике, как и в исходном исследовании [85], при формировании векторных предствлений используется процедура перекрестного скользящего контроля, которая снижает смещение полученных значений. Следующие слои композиции имеют аналогичную структуру, за исключением того, что в них не проводится перегенерация обучающих примеров с помощью скользящего окна.

2.4 Выводы к главе

1. Предложен метод обучения деревьев решений с линейными и нелинейными разделителями и их композиций. Согласно работам [27, 29—31]

совместная оптимизация отступа и критерия неоднородности данных, реализованная в этом методе позволит улучшить обобщающую способность деревьев решений с линейными и нелинейными разделителями.

2. Представлены оценки обобщающей способности ансамблей деревьев решений с линейными и нелинейными разделителями. На основе полученных оценок (Теоремы 5 и 8) можно сформулировать критерии подбора методов регуляризации:
   - сокращение общего числа цепочек (листьев) в деревьях,
   - сокращение длины цепочек решений,
   - сокращение цепочек решений с низкими вероятностями перехода.
3. Предложен модифицированный метод регуляризации случайных ансамблей деревьев решений, удовлетворяющий всем предложенным критериям подбора. В этом методе за счет глобальной оптимизации функции потерь не происходит увеличения эмпирического риска.

Глава 3. Экспериментальные исследования предложенных методов

3.1 Архитектура программных средств построения композиций деревьев решений с линейными или нелинейными разделителями

Для проведения экспериментальных исследований предложенных методов была разработана архитектура для организации глобально распределенного обучения случайных лесов деревьев решений с линейными и нелинейными разделителями и на ее основе разработан комплекс программ [8]. Время обучения случайного леса деревьев решений с многомерными (линейными и нелинейными) разделителями на данных большого объема может существенно превосходить время обучения лесов с одномерными разделителями [5]. При разработке систем обучения случайных лесов таких деревьев необходимо учитывать, что их построение имеет следующие особенности. 1. Время обучения отдельных деревьев ансамбля может существенно отличаться ввиду отличий в структуре и высокой вычислительной сложности построения многомерных разделителей (алгоритм построения нелинейных разделителей имеет полиномиальную вычислительную сложность), поэтому распараллеливание обучения на уровне отдельных деревьев может привести к простою оборудования. 2. Существует необходимость подбора аппаратных средств и библиотек для оптимизации в зависимости от типа разделителя: если линейный разделитель может быть построен за приемлемое время с использованием ресурсов центральных процессоров, то для обучения нелинейных разделителей необходимо задействование графических или тензорных процессоров (Graphic processing unit, GPU; Tensor processing unit, TPU). Для сокращения времени обучения лесов деревьев решений может применяться облачная вычислительная инфраструктура, однако она также накладывает определенные ограничения:

– практически неограниченное количество доступных виртуальных вычислительных узлов при ограниченном времени доступа к ним,
– относительно высокие задержки при передаче данных между узлами.

При использовании облачной инфраструктуры важную роль играет экономический фактор – стоимость обучения ансамбля, пропорциональная затраченному процессорному времени. Для того чтобы соответствовать перечисленным огра-

ничениям система обучения случайных лесов деревьев решений должна соответствовать следующим требованиям:

- Параллелизм на уровне отдельных разделителей, что позволит снизить простой аппаратных ресурсов и назначать задания на обучение различным видам вычислительных узлов в зависимости от типа разделителя.
- Возможность динамического масштабирования вычислительных ресурсов.
- Минимизация передачи данных между узлами вычислительной системы.

Комплекс программ для обучения случайных лесов деревьев решений с линейными и нелинейными разделителями состоит из следующих основных компонентов (рис. 3.1):

- Сервис обучения узлов деревьев решений с применением графических ускорителей (GPU node trainer).
- Сервис обучения узлов деревьев решений с использованием ресурсов центрального процессора (CPU node trainer).
- Сервис «Очередь задач на обучение узлов деревьев решений (Очередь)» реализует конкурентный доступ к очереди для чтения и записи задач на обучение узлов деревьев решений.
- Монитор выполняет контроль загруженности имеющихся аппаратных средств и состояние очереди. В случае роста размера очереди и наличия свободных аппаратных средств Монитор инициирует запуск дополнительных сервисов обучения узлов. В случае выявления простаивающих сервисов обучения узлов инициируется их остановка и освобождение аппаратных ресурсов.

Время обучения разделителя в системе труднопрогнозируемо, так как оно зависит от количества обучающих примеров, их конфигурации в признаковом пространстве, типа разделителя, производительности рабочего узла, что усложняет централизованную балансировку нагрузки и распределение задач. Поэтому в предложенной архитектуре централизованный механизм распределения заданий на обучение разделителей отсутствует: каждый компонент обучения самостоятельно выполняет мониторинг Очереди на предмет наличия подходящих заданий, что позволяет упростить балансировку нагрузки и снизить количество пересылаемых сообщений между рабочими узлами. Запуск и управление системой производится с помощью платформы Kubernetes (рис.

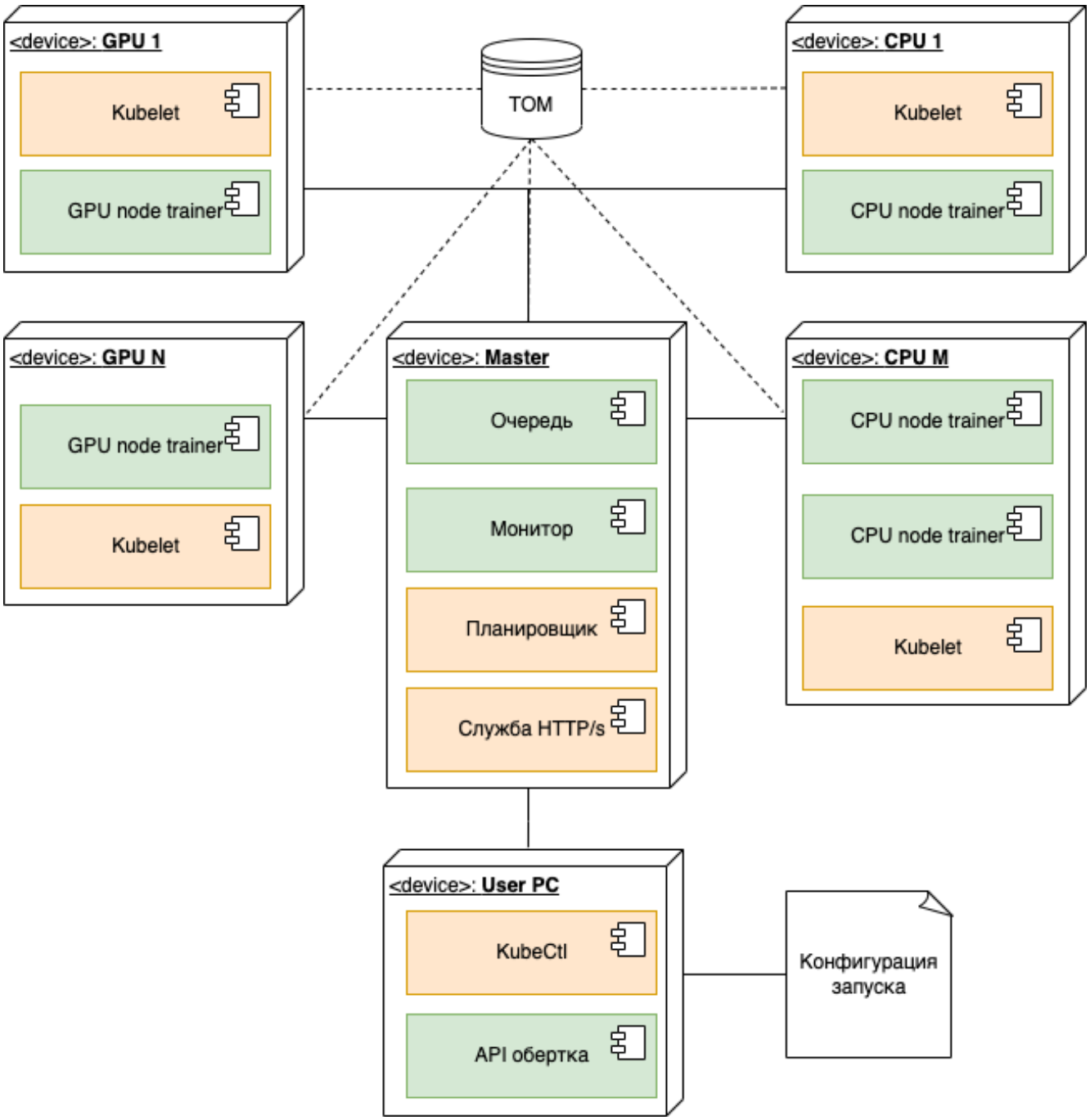

Рисунок 3.1 — Архитектура комплекса программ для обучения случайных лесов деревьев решений с линейными и нелинейными разделителями

3.2). Компонент ReplicationController (контроллер репликации) этой платформы инициирует запуск компонентов Очередь, Монитор и заданного количества сервисов NodeTrainer (Сервис обучения узлов деревьев решений). Управление размещением этих компонентов по рабочим узлам системы осуществляется с помощью компонента «Планировщик». Каждый компонент системы запускается в отдельном изолированном контейнере с помощью инструмента Docker. В процессе работы системы компонент «Монитор» периодически запрашивает

количество ожидающих и выполняющихся задач в Очереди и изменяет ограничение на количество запущенных компонентов NodeTrainer.

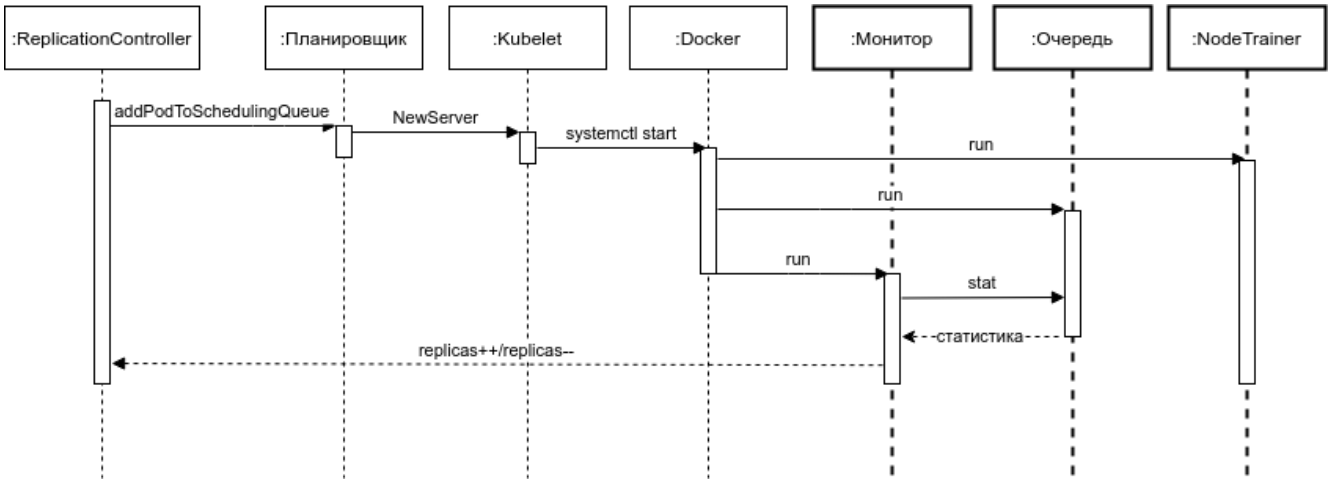

Рисунок 3.2 — Диаграмма запуска распределенной системы обучения случайных лесов деревьев решений с линейными и нелинейными разделителями. Полужирным выделены компоненты, входящие непосредственно в состав системы

На рис. 3.3 показана последовательность действий, выполняемых в системе при обучении разделителя дерева решений. Обучение разделителя выполняется компонентом NodeTrainer. В системе может параллельно функционировать множество подобных компонентов. Перед запуском обучения леса NodeTrainer с помощью компонента загрузки данных DataLoader получает и сохраняет локальную копию обучающих данных. Далее NodeTrainer запрашивает у Очереди задач битовую маску, определяющую примеры из обучающие выборки, которые будут использоваться для обучения разделителя. Очередь помечает загруженную задачу флагом «на исполнении». В случае, если время обучения превысит заранее заданный порог, Очередь сбросит этот флаг, что приведет к перезапуску задачи на другом NodeTrainer. По итогу обучения полученная модель разделителя загружается в Очередь, которая инициирует сохранение результатов обучения на диск. Приведенная схема запуска заданий позволяет минимизировать передачу обучающих данных между компонентами системы в процессе обучения, так как для указания обучающих примеров при запуске заданий необходима лишь битовая маска. Обучение разделителей дерева в компоненте NodeTrainer проводится с помощью модифицированных реализаций метода опорных векторов (SVM) из библиотек Scikit-Learn (для узлов с CPU)

и ThunderSVM (для узлов с GPU). Суть модификации состоит в отказе от хранения в явном виде полученных опорных векторов. Вместо них сохраняются номера векторов в исходной обучающей выборке, что позволяет также сократить как объем данных, передаваемых между компонентами системы в ходе обучения, так и размер обученного ансамбля.

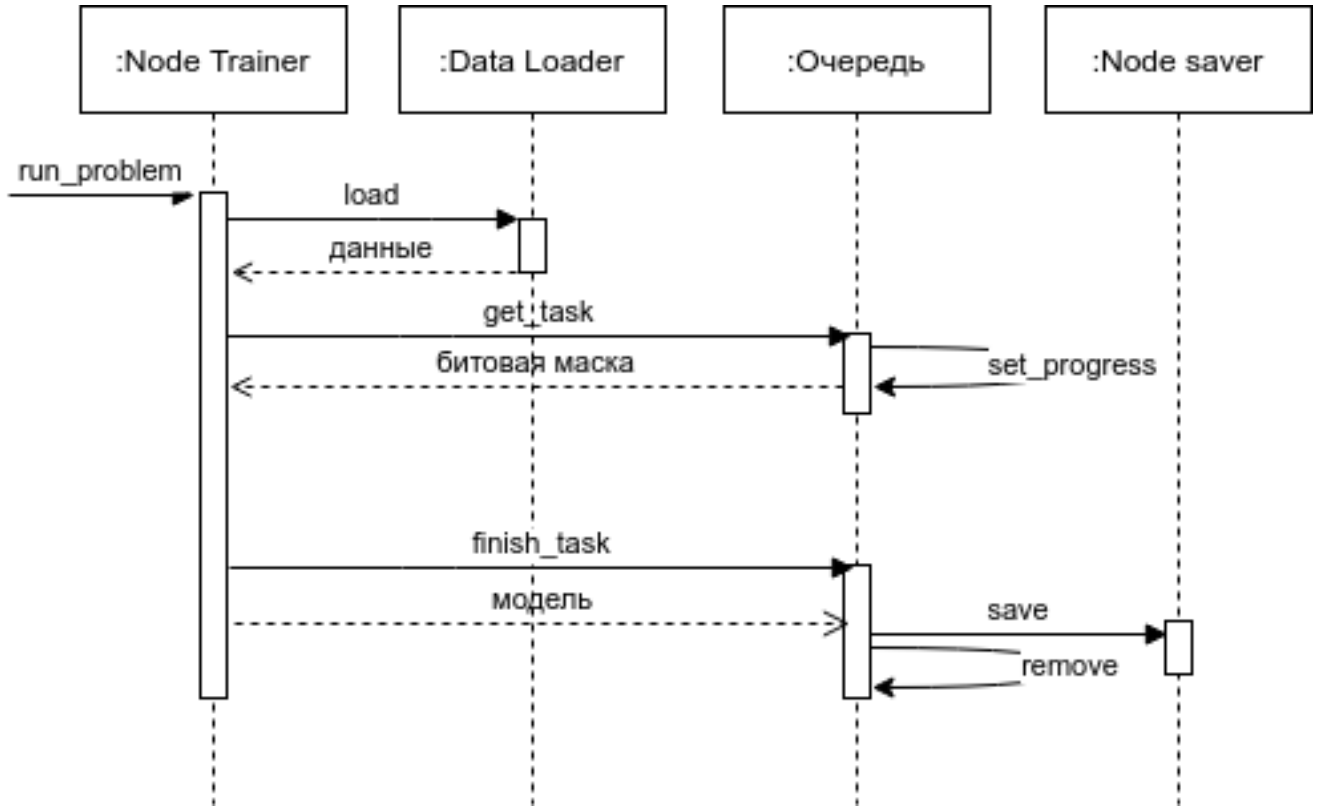

Рисунок 3.3 — Архитектура комплекса программ для обучения случайных лесов деревьев решений с линейными и нелинейными разделителями

Для разработки комплекса программ использовались языки программирования C++/Cython/Python, программные библиотеки LibLinear, LibSVM, ThunderSVM, Numpy, Scipy, утиллиты Kubernetes и Docker.

3.2 Исследование методов и программных средств построения деревьев решений с линейными или нелинейными разделителями и их композиций

3.2.1 Исследование метода построения деревьев решений с линейными или нелинейными разделителями и их композиций

Экспериментальные исследования проводились на размеченных наборах данных Titanic, CIFAR-10 [108] и «Youtube Channels» [109], а также на четырех выборках из коллекции UCI: SatImage, USPS, Letter, MNIST [110]. Наборы данных из UCI применяются для обучения и оценки методов распознавания изображений: MNIST и USPS содержат черно-белые изображения рукописных цифр низкого разрешения. Обучающим примерам в этих выборках соответствуют векторы признаков, каждый элемент которых кодирует яркость (вещественное число в интервале от 0 до 1) соответствующей точки на изображении. SatImage содержит черно-белые фотографии различных видов земной поверхности, а Letter – латинские буквы. Принцип формирования признаков аналогичен применяемому в MNIST и USPS.

Набор CIFAR-10 содержит цветные (RGB) изображения размером 32x32, относящиеся к одному из 10 классов (самолет, лошадь, птица и др.). Обучающим примерам в этой выборке соответствуют векторы признаков, каждый элемент которых кодирует яркость (вещественное число в интервале от 0 до 1) одного из каналов цветности соответствующей точки на изображении.

«Youtube Channels» – набор данных, содержащий психолингвистические маркеры текстов дискуссий трех категорий российских Youtube-каналов: политических проправительственных, политических оппозиционных, аполитичных. В качестве интервала времени для сбора дискуссий был выбран период с 30 апреля 2020 г. по 30 апреля 2021 г. Набор содержит маркеры комментариев к 4807 видео: 2629 видео с политическими дискуссиями (5 млн. сообщений) и 2178 аполитичных видео (1,2 млн сообщений).

Для каждого видео, все комментарии и ответы были объединены в единый текст. Извлечение признаков из текстов дискуссий осуществлялось с помощью библиотеки Titanis [111], разработанной в ФИЦ ИУ РАН. Отбор значимых для

классификации признаков производился с применением статистического подхода, предложенного ранее в работах с с участием автора диссертационного исследования: [112, 113]. В результате, для классификации были отобраны следующие категории признаков:

1. Психолингвистические маркеры. Показатели, вычисляемые на основе морфологических признаков словоупотреблений текстов:
   - отношение количества глаголов к количеству прилагательных (Коэффициент Трейгера);
   - отношение количества инфинитивов к общему количество глаголов;
   - частота употребления местоимений первого лица множественного числа;
   - частота глаголов первого лица в прошедшем времени.
2. Тематическая лексика: частоты встречаемости лексики из следующих словарей: тематических (медицинские термины,экологические термины, политические термины), тональности/настроения (мотивационная лексика, тревожная лексика, враждебная лексика).
3. Лексика основных эмоций. Набор словарей, включающих лексику, связанную с выражением основных эмоций в тексте: отвращение, стыд, гнев, страх, печаль, счастье и удивление.
4. Тональность: численная характеристика текста, рассчитанная с помощью словаря оценочной лексики Лиииса-Крауда [114].

Состав перечисленных словарей рассмотрен в работе [111].

Titanic - демонстрационный размеченный набор данных, предназначенный для обучения методов прогнозирования выживаемости при кораблекрушении (на примере катастрофы Титаника). После отбора значимых для классификации признаков с применением подхода [112] в наборе были оставлены категориальные (класс билета, пол, порт посадки) и числовые поля(возраст, количество братьев и сестер, количество родителей, плата за проезд). Для преобразования категориальных полей в численные использовалось унитарное кодирование (one-hot encoding). Каждому обучающему примеру в наборе соответствует метка ’1’ - выживший или ’0’ - не выживший.

Основные характеристики перечисленных наборов данных приведены в таблице 1.

Таблица 1 — Характеристики наборов данных, используемых в ходе исследования

Набор данных	Размер обучающей выборки	Размер тестовой выборки	Признаков	Классов
MNIST	60,000	10,000	784	10
USPS	7,291	2,007	256	10
Letter	15,000	5,000	16	26
SatImage	4,435	2,000	36	6
Titanic	712	179	25	2
Cifar-10	50,000	10,000	8192	10
Youtube channels	3364	1442	82	3

Исследовались случайные леса, деревья которых построены с помощью методов C4.5 (Random Forest), CO2 (CO2 Forest), WODT, а также подхода, предложенного в настоящей работе (Kernel Forest). Для подбора гиперпараметров использовалась статистическая процедура перекрестного скользящего контроля. Оценка точности (accuracy, precision) и полноты (recall) классификации проводилась на отложенной выборке. Для каждого класса примеров в анализируемых наборах данных перечисленные оценки вычислялись с использованием следующих выражений:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{3.1}$$

$$precision = \frac{tp}{tp + fp} \tag{3.2}$$

$$recall = \frac{tp}{tp + fn}, \tag{3.3}$$

где tp - количество примеров, корректно размеченных, как относящиеся к определенному классу; tn - количество примеров, корректно размеченных как не относящиеся к этому классу; fp - количество примеров, некорректно отмеченных как принадлежащие этому классу, fn - количество примеров, некорректно отмеченных как не принадлежащие этому классу. Для задач бинарной классификации (Titanic) в качестве результата выбирались оценки целевого класса (выжившие). Для задач многоклассовой классификации (MNIST,USPS,

SatImage, Youtube channels, Letter, Cifar-10) использовалось макроусреднение (то есть усреднение оценок для всех классов).

Для предложенного подхода к обучению случайных лесов деревьев решений с линейными и нелинейными разделителями подбирались следующие гиперпараметры: количество деревьев T=30,100,300, параметр регуляризации при построении разделителей C=100, 1000, 3000, 5000, максимальная глубина дерева n=3,4,5,6,7, доля признаков, которые необходимо учитывать в каждом узле f=0.08,0.1,0.2,0.3,0.4,0.5, а также параметры ядра gamma=10,100 для Гауссовского ядра и degree=3 для полиномиального ядра.

В таблице 2 представлены результаты экспериментального исследования. Наилучшие результаты для большинства наборов данных для были достигнуты при обучении случайного леса деревьев с нелинейными разделителями (с Гауссовским ядром), тогда как деревья с полиномиальным ядром превзошли другие в наборе данных «Каналы Youtube». Для набора данных Titanic наилучшие результаты были получены с помощью леса деревьев с линейными разделителями, а нелинейные ядра приводили к переобучению. Согласно приведенным результатам, предложенный метод превосходит по точности другие подходы на большей части наборов данных. Статистический анализ полученных результатов с помощью критерия Уилкоксона показывает, что предложенный в исследовании метод (Kernel Forest) превосходит все другие подходы, кроме CO2 Forest (значения p составляют около 0,004). Метод CO2 Forest с линейными разделителями позволяет одновременно оптимизировать кросс-энтропию и отступ; поэтому он показывает уровень точности, сопостовимый с предложенным методом, на относительно небольших наборах данных (например, «Letter»), когда использование нелинейных ядер приводит к переобучению.

В таблицах 3 и 4 представлены показатели точности и полноты. Они также показывают, что предлагаемый метод (Kernel Forest) превосходит большинство других рассмотренных методов, кроме CO2 Forest.Согласно полученным результатам, наибольший прирост точности и полноты классификации по сравнению с аналогами получен на наборах изображений с относительно большим количеством обучающих примеров (от 50 тыс. и более).

В таблице 5 показаны результаты экспериментальных исследований усиления и регуляризации случайных лесов деревьев решений с нелинейными разделителями. Следует отметить, что усиление из раздела 2.4 позволяет значительно улучшить результаты классификации на этих наборах данных (до

Таблица 2 — Результаты экспериментального исследования точности (accuracy) методов классификации на основе деревьев решений (без регуляризации)

Набор данных	Random Forest	CO2 Forest	OC1	WODT	Kernel Forest
MNIST	96.7	97.5	79.0	93.8	98.5
USPS	94.9	95.7	94.7	92.0	95.8
Letter	93.3	95.4	93.5	85.1	97.4
SatImage	84.2	91.1	90.5	87.6	91.0
Titanic	79.3	78.1	68.0	73.1	80.2
Cifar-10	50.1	-	-	-	58.0
Youtube channels	84.5	57.0	84.7	84.0	94.1

Таблица 3 — Результаты экспериментального исследования точности (precision) методов классификации на основе деревьев решений (без регуляризации)

Набор данных	Random Forest	CO2 Forest	OC1	WODT	Kernel Forest
MNIST	96.6	97.6	79.0	93.7	98.6
USPS	95.5	95.9	95.3	92.5	95.9
Letter	94.1	98.2	96.2	87.9	97.5
SatImage	89.3	89.2	88.9	85.9	90.6
Titanic	82.0	82.6	74.3	78.0	83.1
Cifar-10	49.6	-	-	-	57.5
Youtube channels	92.9	55.7	81.3	89.7	94.7

0,6%), повышая применимость деревьев со сложными ядрами. В то же время применение регуляризации в дополнение к усилению практически не привело к улучшению результатов классификации на тестовых выборках. Полученные оценки качества предложенных методов обучения и регуляризации случайных лесов деревьев решений с линейными и нелинейными разделителями показывают, что они могут применяться в прикладных системах распознавания рукописных символов (точность более 99%) и анализа дискуссий в сети интернет (точность более 95%).

Таблица 4 — Результаты экспериментального исследования полноты (recall) методов классификации на основе деревьев решений (без регуляризации)

Набор данных	Random Forest	CO2 Forest	OC1	WODT	Kernel Forest
MNIST	96.7	97.5	79.0	93.8	98.5
USPS	94.9	95.7	94.7	92.0	95.8
Letter	93.3	95.4	93.5	85.1	97.4
SatImage	84.2	91.1	90.5	87.6	91.0
Titanic	79.3	78.1	68.0	73.1	80.2
Cifar-10	50.1	-	-	-	58.0
Youtube channels	84.5	57.0	84.7	84.0	94.1

Таблица 5 — Результаты исследования усиления и регуляризации леса деревьев решений с линейными и нелинейными разделителями (точность, accuracy)

Набор данных	Усиление	Усиление + шум	Усиление + базовая регуляризация	Усиление + l^2-регуляризация	Усиление + l^2-регуляризация + шум
MNIST	99.2	99.0	99.2	99.3	99.3
Cifar-10	59.0	59.0	59.1	59.4	59.7
Youtube channels	95.5	95.5	95.5	95.6	95.7

3.2.2 Исследование метода классификации объектов со сложной структурой с помощью композиции деревьев решений с линейными и нелинейными разделителями

В исследовании использовались композиции Deep Forests с тремя слоями. Для подбора гиперпараметров использовался перебор по сетке, для получения оценок точности классификации использовалась статистическая процедура перекрестного скользящего контроля. Далее выполнялась оценка точности (precision и accuracy) и полноты (recall) с макроусреднением на отложенной выборке. Параметры регуляризации разделителей выбирались из множества

$\{100, 1000, 3000, 5000\}$, максимальная глубина дерева {4,5,6,7,8}, доля признаков, которые учитываются в ходе построения разделителя {0,08,0,1,0,2}, параметры Гауссовского ядра ($gamma = \{10100\}$, размер скользящего окна $[8 - 128]$, размер шага скользящего окна $[2 - 32]$ в зависимости от набора данных. Результаты представлены в таблице 6. В таблице Deep Forest - стандартная композиция на основе лесов деревьев решений с одномерными разделителями, Deep Kernel Forest - композиция на основе лесов деревьев решений с нелинейными разделителями и Гауссовским ядром.

Таблица 6 — Результаты экспериментальных исследований (точность, accuracy) методов анализа объектов со сложной структурой

Набор данных	Deep Forest	Kernel Forest	Deep Kernel Forest	Deep Kernel Forest + регуляризация	Скользящее окно + Deep Kernel Forest + регуляризация
MNIST	99.2	99.1	98.0	99.2	99.4
USPS	95.9	95.8	93.5	97.8	97.9
Letter	96.3	97.4	97.2	98.5	98.5
Cifar-10	61.8	58.0	60.3	62.9	63.2

Точность предложенной модификации для анализа объектов со сложной структурой сопоставима с Deep Forest. Таблица 6 показывает, что дальнейшее усложнение разделителей в Deep Forest без добавления регуляризации не приводит к улучшению точности классификации. Это с тем, что сложность композиции Deep Forest достаточно высока, и дальнейшее её увеличение приводит к переобучению. С другой стороны, добавление усиления и регуляризации [107] позволяет добиться существенного прироста качества классификации (до 2% по точности) по сравнению с немодифицированным методом DeepForest.

В таблицах 7, 8 представлены результаты оценки точности (precision) и полноты (recall) с макроусреднением анализируемых методов. Согласно полученным результатам, предложенный метод превосходит аналоги по уровню точности и полноты.

Таблица 7 — Результаты экспериментальных исследований (точность, precision) методов анализа объектов со сложной структурой

Набор данных	Deep Forest	Kernel Forest	Deep Kernel Forest	Deep Kernel Forest + регуляризация	Скользящее окно + Deep Kernel Forest + регуляризация
MNIST	99.3	99.2	99.5	99.3	99.5
USPS	96.9	97.2	95.3	97.5	97.6
Letter	95.8	96.0	96.2	96.0	96.1
Cifar-10	62.4	58.5	60.9	62.9	63.3

Таблица 8 — Результаты экспериментальных исследований (полнота, recall) методов анализа объектов со сложной структурой

Набор данных	Deep Forest	Kernel Forest	Deep Kernel Forest	Deep Kernel Forest + регуляризация	Скользящее окно + Deep Kernel Forest + регуляризация
MNIST	99.3	99.2	99.0	99.3	99.4
USPS	95.3	95.0	94.1	95.2	95.3
Letter	89.7	91.1	90.0	96.9	96.9
Cifar-10	61.8	58.0	60.3	62.8	63.1

3.2.3 Исследование времени обучения композиций лесов деревьев решений с линейными и нелинейными разделителями

В ходе экспериментальных исследований оценивалось время, необходимое для обучения случайных лесов деревьев решений с линейными и нелинейными разделителями с помощью различных методов. Исследования проводились с применением следующих аппаратных средств: 12-ядерный процессор AMD Ryzen Threadripper 1920, 128 ГБ ОЗУ, Nvidia GeForce RTX 2080. В ходе исследования анализировалось время обучения на нескольких различных наборах данных, оценивалось время построения ансамблей размером 10 деревьев и глубиной 5 (рис. 3.4). Выбор этих гиперпараметров обусловлен возможностями используемых аппаратных средств. При исследовании методов построения лесов деревьев решений с линейными разделителями использовался метод двойного координатного спуска, реализованный в Scikit-Learn. В результате было выявлено, что большинство рассмотренных подходов, в том числе и предложен-

ный метод Kernel Forest, показывают линейную зависимость от размера набора данных при фиксированной глубине дерева. Однако метод обучения CO2 Forest, в котором производится поиск глобального оптимума выпукло-вогнутой функции потерь, показывает полиномиальную зависимость времени обучения от размера обучающей выбора.

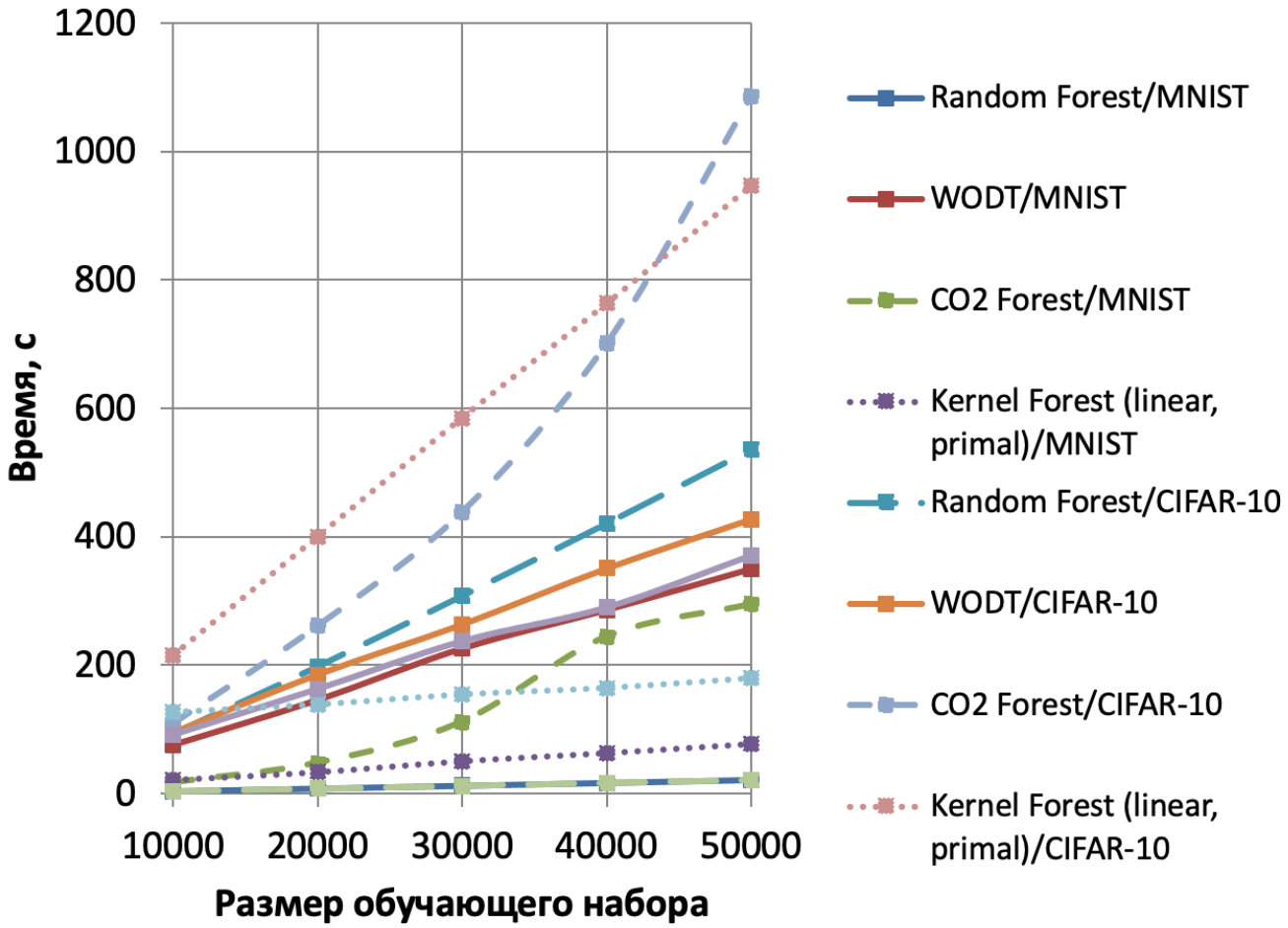

Рисунок 3.4 — Время обучения леса деревьев решений с линейными разделителями на наборе данных MNIST (1000 деревьев высотой 4)

Полученные оценки согласуются с приведенными в главе 2 теоретическими результатами анализа сложности алгоритмов обучения, таким образом, предложенный подход может быть использован для обучения случайных лесов деревьев решений с линейными разделителями на больших наборах данных.

Во втором эксперименте была исследована производительность предложенного метода для построения лесов деревьев решений с Гауссовским ядром на тех же наборах данных, что и в первом эксперименте (рис. 3.5). При обучении лесов деревьев решений с нелинейными разделителями предложенным методом

используются библиотеки ThunderSVM и LibSVM. В ThunderSVM при оптимизации используются ресурсы графических процессоров, поэтому он позволяет обучать деревья с нелинейными разделителями в несколько раз быстрее, чем LibSVM; однако при использовании обеих библиотек выявлена полиномиальная зависимость между временем обучения и размером обучающей выборки.

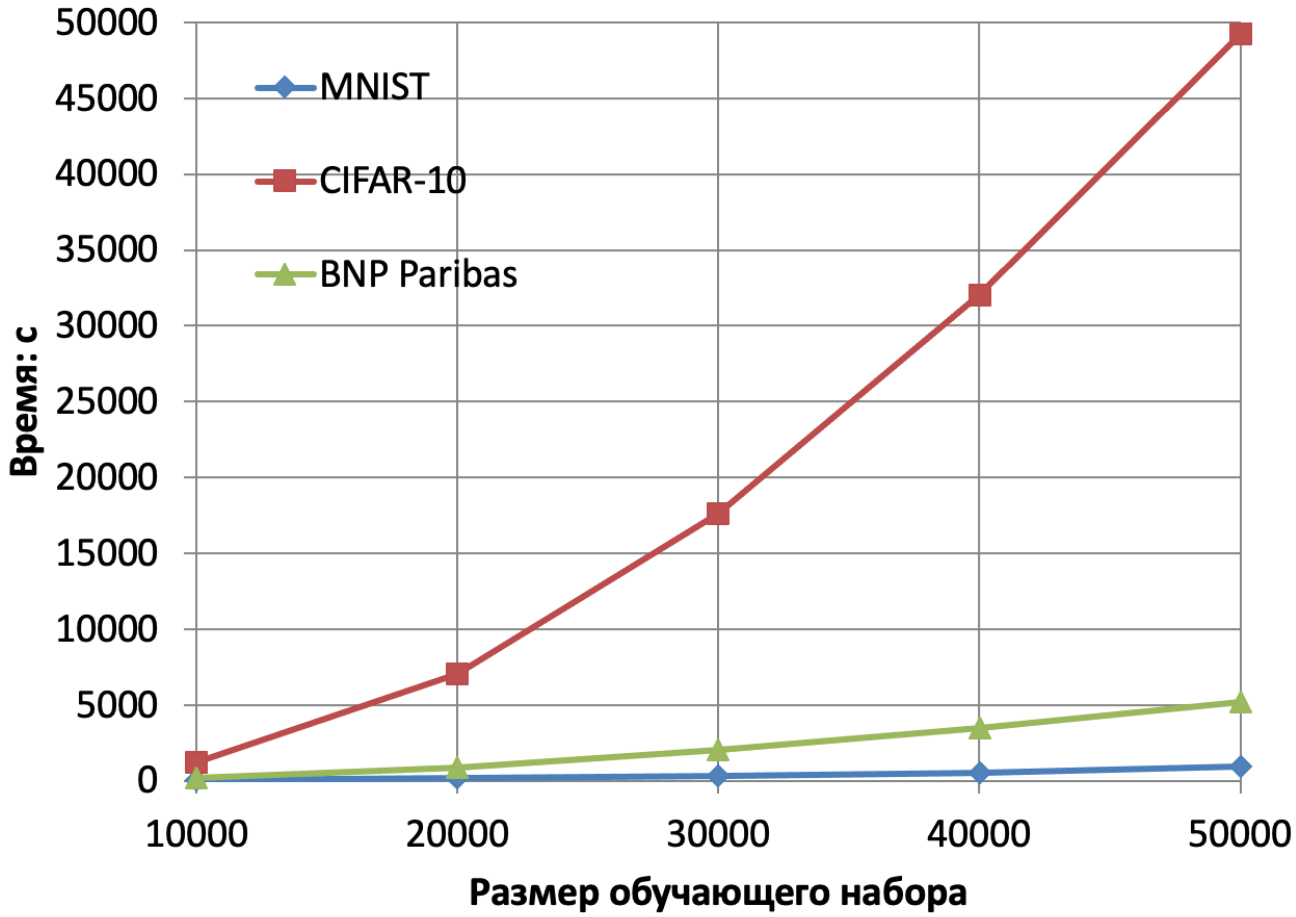

Рисунок 3.5 — Время обучения леса деревьев решений с линейными разделителями на наборе данных MNIST (1000 деревьев высотой 4)

3.2.4 Исследование архитектуры и комплекса программ для обучения случайных лесов деревьев решений с нелинейными разделителями

Экспериментальные исследования разработанной системы распределенного построения случайных лесов деревьев решений с линейными и нелиней-

ными разделителями проводились на размеченных наборах данных MNIST и CIFAR-10, которые применяются для обучения и оценки методов распознавания изображений:

В ходе исследования оценивалась производительность двух типов систем:

– Система с параллелизмом на уровне отдельных деревьев, предназначенная для работы на машинах с общей разделяемой памятью. Такой подход используется в большинстве систем и библиотек построения случайных лесов деревьев решений, например, в Scikit-learn.
– Система с параллелизмом на уровне отдельных разделителей, предложенная в настоящей статье. Предназначена для работы на машинах с распределенной памятью.

Эксперименты выполнялись на высокопроизводительном вычислительном комплексе с архитектурой IBM Power 9 (160 виртуальных процессоров, 1024 Гб ОЗУ). В ходе экспериментов на случайным образом выбранных подмножествах наборов данных MNIST и CIFAR-10 проводилось обучение лесов деревьев решений с линейными и нелинейными (с RBF-ядром) разделителями высотой 4 и 10. Для устранения смещения полученных оценок, связанных с рандомизацией выборки данных и процесса построения лесов, обучение повторялось 3 раза для каждой подвыборки данных и фиксировалось среднее время обучения. Предварительные исследования показали, что при обучении лесов деревьев решений высотой 4 на приведенных наборах данных практически все деревья ансамбля оказываются полными, то есть время обучения таких деревьев должно быть приблизительно одинаковым, что снижает простой оборудования при использовании систем с общей памятью.

На рис. 3.6 представлена зависимость времени обучения леса деревьев решений из 1000 деревьев высотой не более 4 от размера обучающей выборки и системы, применяемой для обучения (для набора данных MNIST). На рисунке использованы следующие обозначения:

– CENTR_LINEAR – система построения деревьев решений с параллелизмом на уровне отдельных деревьев, линейный разделитель, 20 потоков.
– CENTR_RBF LINEAR – система построения деревьев решений с параллелизмом на уровне отдельных деревьев, Гауссовский (RBF) разделитель, 20 потоков.

– DIST_LINEAR – система построения деревьев решений с параллелизмом на уровне отдельных разделителей, линейный разделитель, 80 компонентов обучения NodeTrainer.
– DIST_RBF –система построения деревьев решений с параллелизмом на уровне отдельных разделителей, Гауссовский (RBF) разделитель, 80 компонентов обучения NodeTrainer.

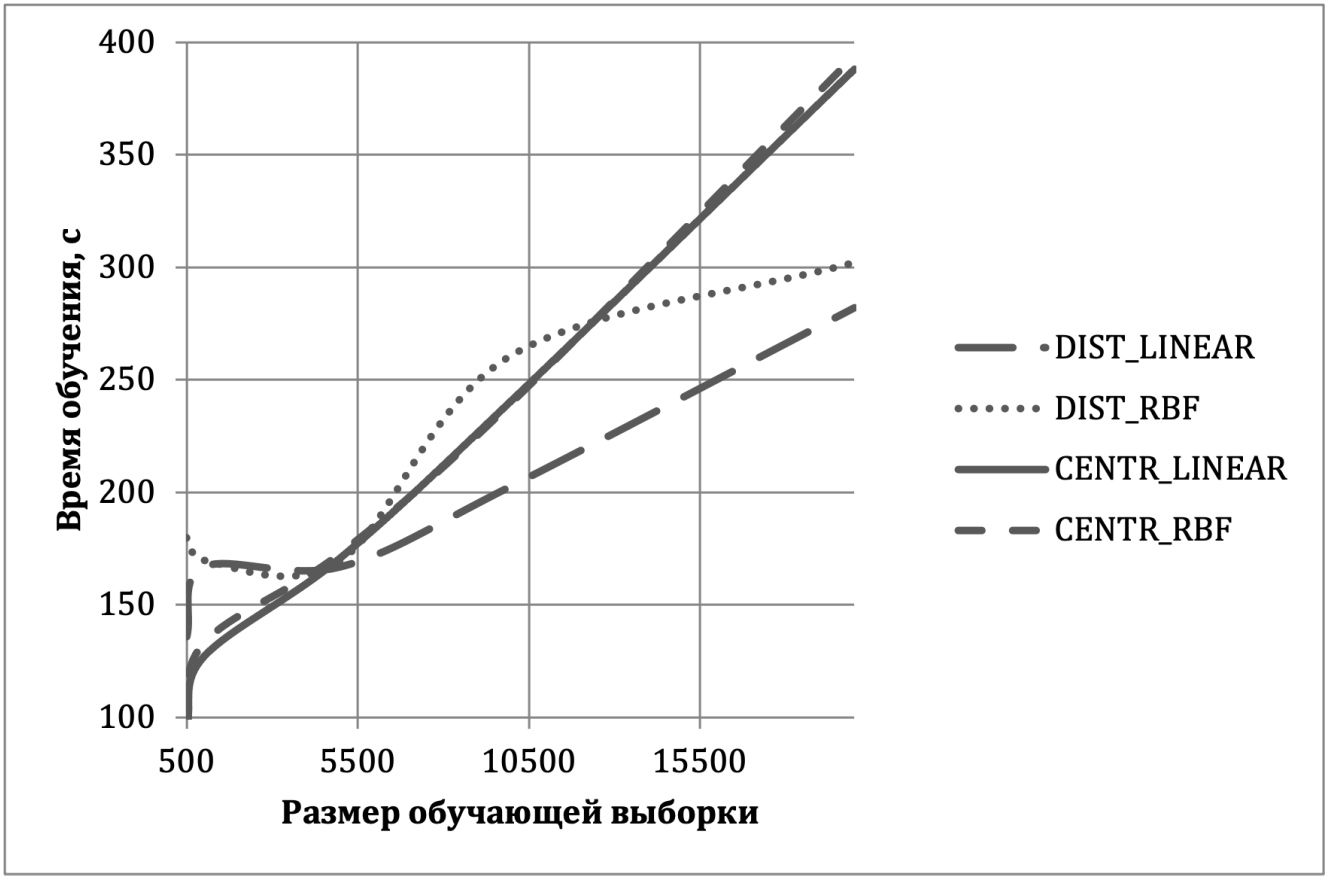

Рисунок 3.6 — Время обучения леса деревьев решений с линейными разделителями на наборе данных MNIST (1000 деревьев высотой 4)

Согласно полученным результатам, скорость обучения на небольшом наборе данных (не более 12 тыс. обучающих примеров с размерностью признакового пространства менее 500 признаков) у системы с разделяемой памятью выше, чем у предложенной системы с распределенной памятью, что связано с затратами на передачу данных между вычислительными узлами в распределенной системе. Вместе с тем графики зависимости времени обучения от размера выборки в предложенной системе оказалось более пологими, чем в системе с разделяемой памятью. При обучении на выборках с более чем 12 тыс. примеров скорость построения леса в предложенной системе оказывается выше, чем

в системе с разделяемой памятью благодаря более полному задействованию (отсутствию простоев) аппаратных ресурсов.

На рис. 3.7 представлена зависимость времени обучения леса деревьев решений из 1000 деревьев высотой 4 от размера обучающей выборки и алгоритма (для набора данных CIFAR-10). Как и в предыдущем случае, графики зависимости времени обучения от размера выборки в предложенной системе являются более пологими, чем в системе с разделяемой памятью, однако применение распределенной системы оправдано только при обучении на больших наборах данных.

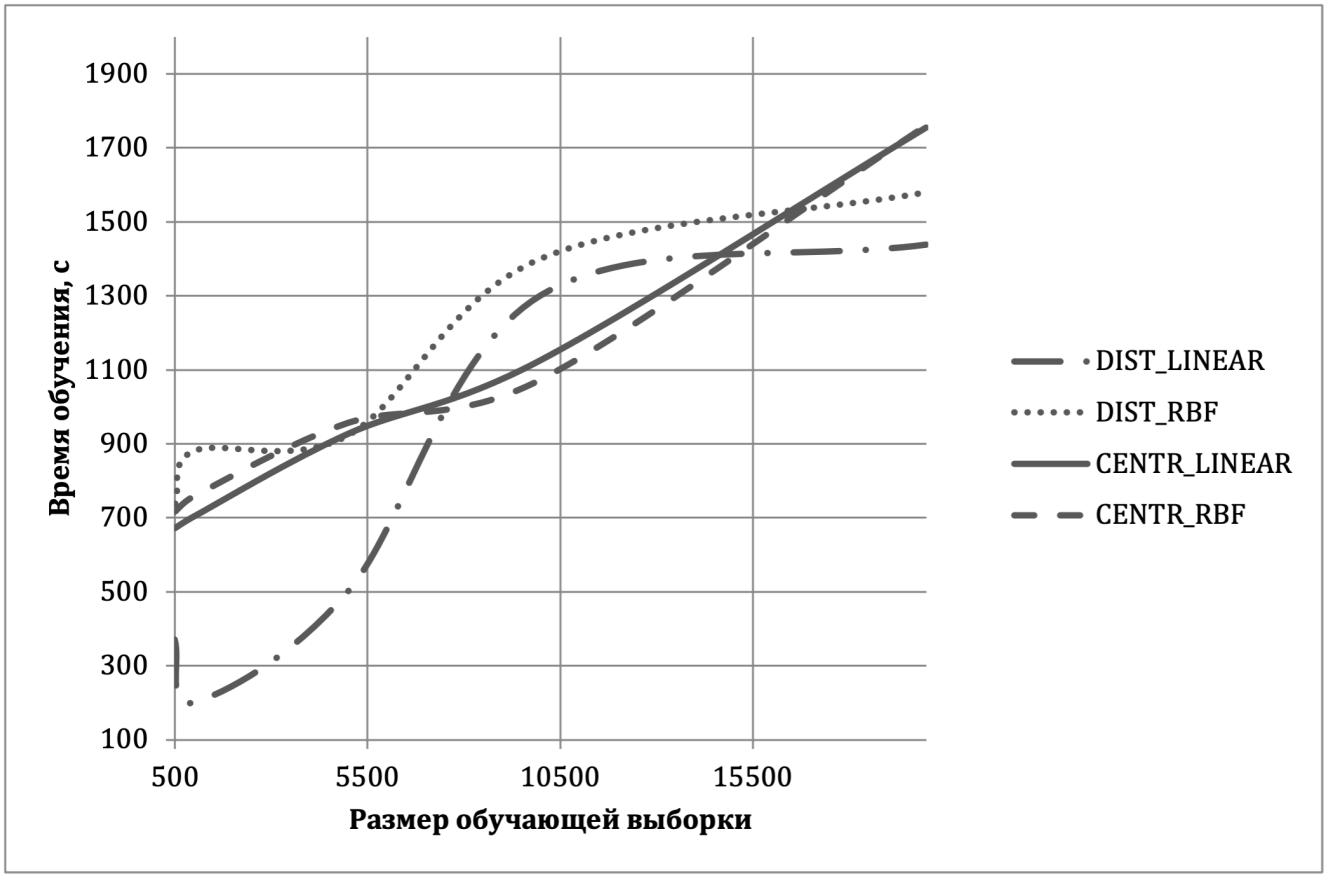

Рисунок 3.7 — Время обучения леса деревьев решений с RBF разделителями на наборе данных CIFAR-10 (1000 деревьев высотой 4)

Предварительные исследования показали, что при обучении лесов деревьев решений высотой 10 на больших наборах данных (более 15 тыс. примеров), структура отдельных деревьев в ансамбле существенно отличается, поэтому и время обучения таких деревьев должно различаться, что увеличивает простой оборудования при использовании систем с общей памятью. На рис. 3.8 представлена зависимость времени обучения леса деревьев решений из 100

деревьев высотой не более 10 от размера обучающей выборки и алгоритма (для набора данных MNIST). Выявлено, что система с разделяемой памятью превосходит предложенную систему по скорости обучения. При построении лесов деревьев решений на небольших подмножествах выборки MNIST высота деревьев оказалась примерно одинаковой и существенно меньшей порогового значения, поэтому использование распределенной системы не привело к ускорению обучения.

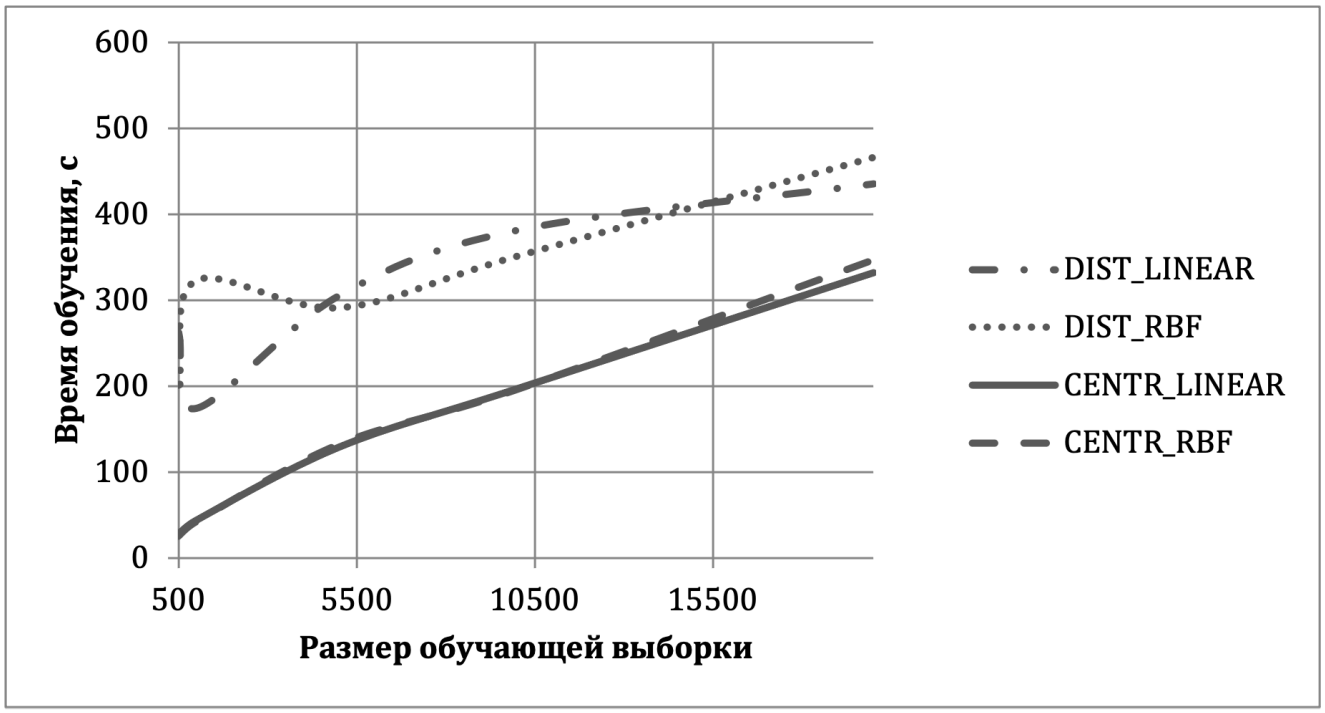

Рисунок 3.8 — Время обучения леса деревьев решений с линейными разделителями на наборе данных MNIST (100 деревьев высотой 10)

На рис. 3.9 представлена зависимость времени обучения леса деревьев решений из 100 деревьев высотой 10 от размера обучающей выборки и алгоритма (для набора данных CIFAR-10). Согласно полученным результатам, предложенная система обеспечивает значительный прирост скорости обучения при построении лесов деревьев решений большой высоты (10 и более) на данных большой размерности (более 3 тыс. признаков), таким образом, применение предложенной системы для обработки подобных данных приведет к снижению общего машинного времени, необходимого для обучения и, как следствие, к снижению затрат.

Исследования системы обучения деревьев решений, основанной на предложенной архитектуре показали, что она позволяет значительно снизить общее

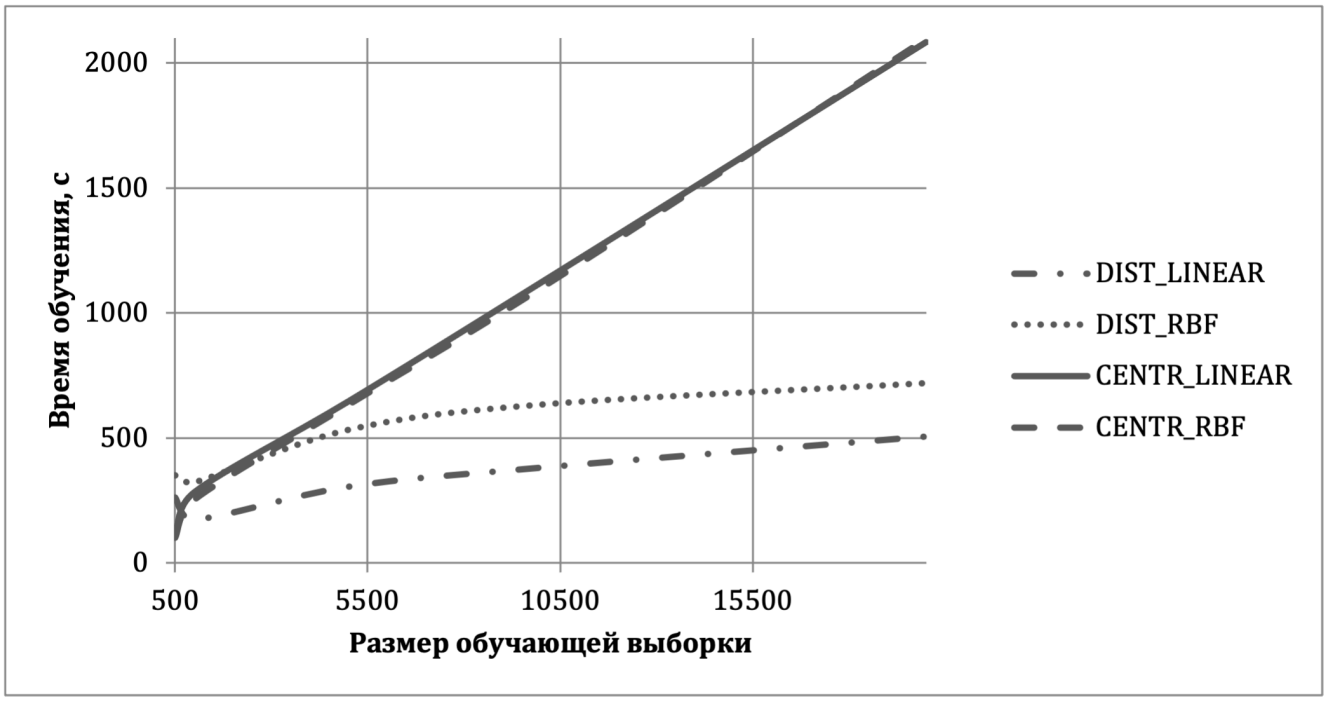

Рисунок 3.9 — Время обучения леса деревьев решений с линейными разделителями на наборе данных CIFAR-10 (100 деревьев высотой 10)

машинное время, необходимое для обучения случайных лесов деревьев решений на массивах данных большой размерности. Дальнейшие исследования будут направлены на создание модификаций предложенной архитектуры, поддерживающих выполнение онлайн-обучения лесов деревьев решений.

3.3 Выводы к главе

1. Представлена архитектура распределенной системы обучения случайных лесов деревьев решений с линейными и нелинейными разделителями.
2. Исследования системы обучения деревьев решений, основанной на предложенной архитектуре, показали, что она позволяет значительно снизить общее машинное время, необходимое для обучения случайных лесов деревьев решений на массивах данных большой размерности.
3. Согласно результатам исследований, предложенные методы превосходит по точности и полноте другие подходы на основе деревьев решений на большей части рассмотренных наборов данных.

Заключение

Основные результаты работы заключаются в следующем:

1. Разработан и реализован метод построения деревьев решений с применением линейных и нелинейных разделителей, при построении которых оптимизируется отступ между разделяемыми объектами и произвольный критерий однородности.
2. Проведены экспериментальное исследование разработанного метода, выполнено его сравнение с другими методами.
3. Разработаны методы оценки обобщающей способности случайных ансамблей деревьев решений. Оценки применены для подбора подхода к регуляризации случайных лесов деревьев решений с линейными и нелинейными разделителями. Доказаны соответствующие теоремы и следствия.
4. Разработан метод классификации объектов, характеризующихся наличием связей между признаками.
5. Разработана архитектура программных средств глобально распределенного построения случайных лесов деревьев решений с линейными и нелинейными разделителями.
6. Реализован комплекс программ для обучения лесов деревьев решений с линейными и нелинейными разделителями.

Полученные результаты относятся к направлениям исследований 4, 7, 8, и 9 паспорта специальности 2.3.5.

Список литературы

1. *Ek, A.* How does punctuation affect neural models in natural language inference / A. Ek, J.-P. Bernardy, S. Chatzikyriakidis // Proceedings of the Probability and Meaning Conference (PaM 2020). — 2020. — С. 109—116.
2. Estimation of vegetation indices for high-throughput phenotyping of wheat using aerial imaging / Z. Khan [и др.] // Plant methods. — 2018. — Т. 14, № 1. — С. 1—11.
3. A deep forest improvement by using weighted schemes / L. Utkin [и др.] // 2019 24th Conference of Open Innovations Association (FRUCT). — IEEE. 2019. — С. 451—456.
4. On oblique random forests / B. Menze [и др.] // Machine Learning and Knowledge Discovery in Databases. — 2011. — С. 453—469.
5. *Devyatkin, D.* Random Kernel Forests / D. Devyatkin, O. Grigoriev // IEEE Access. — 2022. — Т. 10. — С. 77962—77979.
6. *Девяткин, Д.* Метод обучения деревьев решений с нелинейными разделителями / Д. Девяткин, О. Григорьев // Искуственный интеллект и принятие решений. — 2022. — Т. 3. — С. 95—104.
7. Методы кросс-языкового поиска тематически похожих нормативно-правовых документов на основе машинного обучения / В. Жебель [и др.] // Искуственный интеллект и принятие решений. — 2022. — Т. 2. — С. 27—35.
8. *Девяткин, Д.* Система распределенного построения случайных лесов деревьев решений с линейными и нелинейными разделителями / Д. Девяткин // Системы высокой доступности. — 2022. — Т. 3. — С. 59—67.
9. *Devyatkin, D.* Estimation of vegetation indices with Random Kernel Forests / D. Devyatkin // Applied Sciences. — 2022.
10. *Девяткин, Д.* Программный комплекс для обучения случайных лесов деревьев решений с нелинейными разделителями / Д. Девяткин // Свидетельство о регистрации программы для ЭВМ. — 2022.

11. *Devyatkin, D.* Extraction of Cognitive Operations from Scientific Texts / D. Devyatkin // Russian Conference on Artificial Intelligence. — Springer. 2019. — С. 189—200.

12. *Воронцов, К. В.* Комбинаторная теория надёжности обучения по прецедентам / К. В. Воронцов // Дис. на соиск. уч. степени д. ф.-м. н. М.: ВЦ РАН. — 2010.

13. *Vapnik, V. N.* An overview of statistical learning theory / V. N. Vapnik // IEEE transactions on neural networks. — 1999. — Т. 10, № 5. — С. 988—999.

14. *Hao, G.* Efficient training and feature induction in sequential supervised learning / G. Hao. — 2009.

15. *Dietterich, T. G.* Machine learning for sequential data: A review / T. G. Dietterich // Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). — Springer. 2002. — С. 15—30.

16. *Huang, Z.* Bidirectional LSTM-CRF models for sequence tagging / Z. Huang, W. Xu, K. Yu // arXiv preprint arXiv:1508.01991. — 2015.

17. *Sha, F.* Shallow parsing with conditional random fields / F. Sha, F. Pereira // Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. — Association for Computational Linguistics. 2003. — С. 134—141.

18. *Peng, F.* Information extraction from research papers using conditional random fields / F. Peng, A. McCallum // Information processing & management. — 2006. — Т. 42, № 4. — С. 963—979.

19. *Chiu, J. P.* Named entity recognition with bidirectional LSTM-CNNs / J. P. Chiu, E. Nichols // Transactions of the Association for Computational Linguistics. — 2016. — Т. 4. — С. 357—370.

20. *Graves, A.* Offline handwriting recognition with multidimensional recurrent neural networks / A. Graves, J. Schmidhuber // Advances in neural information processing systems. — 2009. — С. 545—552.

21. End-to-end attention-based large vocabulary speech recognition / D. Bahdanau [и др.] // Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on. — IEEE. 2016. — С. 4945—4949.

22. *Sønderby, S. K.* Protein secondary structure prediction with long short term memory networks / S. K. Sønderby, O. Winther // arXiv preprint arXiv:1412.7828. — 2014.

23. *Vapnik, V.* The nature of statistical learning theory / V. Vapnik. — Springer science & business media, 2013.

24. Empirical margin distributions and bounding the generalization error of combined classifiers / V. Koltchinskii, D. Panchenko [и др.] // The Annals of Statistics. — 2002. — Т. 30, № 1. — С. 1—50.

25. *Cortes, C.* Multi-class classification with maximum margin multiple kernel / C. Cortes, M. Mohri, A. Rostamizadeh // International Conference on Machine Learning. — 2013. — С. 46—54.

26. *Evgeniou, T.* Leave one out error, stability, and generalization of voting combinations of classifiers / T. Evgeniou, M. Pontil, A. Elisseeff // Machine learning. — 2004. — Т. 55, № 1. — С. 71—97.

27. *Breiman, L.* Technical note: Some properties of splitting criteria / L. Breiman // Machine Learning. — 1996. — Т. 24, № 1. — С. 41—47.

28. *Quinlan, J. R.* Induction of decision trees / J. R. Quinlan // Machine learning. — 1986. — Т. 1, № 1. — С. 81—106.

29. Generalization in decision trees and DNF: Does size matter? / M. Golea [и др.] // Advances in Neural Information Processing Systems. — 1998. — С. 259—265.

30. *Liu, W.* Sparse perceptron decision tree for millions of dimensions / W. Liu, I. W. Tsang // Thirtieth AAAI Conference on Artificial Intelligence. — 2016.

31. *Liu, W.* Making decision trees feasible in ultrahigh feature and label dimensions / W. Liu, I. W. Tsang // Journal of Machine Learning Research. — 2017.

32. *Воронцов, К.* Математические методы обучения по прецедентам (теория обучения машин) / К. Воронцов // Москва. — 2011. — С. 119—121.

33. *Cortes, C.* Support vector machine / C. Cortes, V. Vapnik // Machine learning. — 1995. — Т. 20, № 3. — С. 273—297.

34. A dual coordinate descent method for large-scale linear SVM / C.-J. Hsieh [и др.] // Proceedings of the 25th international conference on Machine learning. — ACM. 2008. — С. 408—415.

35. *Журавлев, Ю.* Об алгебраическом подходе к решению задач распознавания или классификации / Ю. Журавлев // Проблемы кибернетики. — 1978. — Т. 33. — С. 5—68.

36. *Wolpert, D. H.* Stacked generalization / D. H. Wolpert // Neural networks. — 1992. — Т. 5, № 2. — С. 241—259.

37. *Freund, Y.* A decision-theoretic generalization of on-line learning and an application to boosting / Y. Freund, R. E. Schapire // Journal of computer and system sciences. — 1997. — Т. 55, № 1. — С. 119—139.

38. Boosting the margin: A new explanation for the effectiveness of voting methods / R. E. Schapire [и др.] // The annals of statistics. — 1998. — Т. 26, № 5. — С. 1651—1686.

39. *Friedman, J. H.* Greedy function approximation: a gradient boosting machine / J. H. Friedman // Annals of statistics. — 2001. — С. 1189—1232.

40. *Breiman, L.* Bagging predictors / L. Breiman // Machine learning. — 1996. — Т. 24, № 2. — С. 123—140.

41. *Elisseeff, A.* Stability of randomized learning algorithms / A. Elisseeff, T. Evgeniou, M. Pontil // Journal of Machine Learning Research. — 2005. — Т. 6, Jan. — С. 55—79.

42. *Barandiaran, I.* The random subspace method for constructing decision forests / I. Barandiaran // IEEE Trans. Pattern Anal. Mach. Intell. — 1998. — Т. 20, № 8.

43. *Breiman, L.* Random forests / L. Breiman // Machine learning. — 2001. — Т. 45, № 1. — С. 5—32.

44. *Bennett, K. P.* A support vector machine approach to decision trees / K. P. Bennett, J. Blue // 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227). Т. 3. — IEEE. 1998. — С. 2396—2401.

45. *Tibshirani, R.* Margin Trees for High-dimensional Classification. / R. Tibshirani, T. Hastie // Journal of Machine Learning Research. — 2007. — Т. 8, № 3.

46. *Manwani, N.* Geometric decision tree / N. Manwani, P. Sastry // IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). — 2011. — Т. 42, № 1. — С. 181—192.

47. *Hofmann, T.* Kernel methods in machine learning / T. Hofmann, B. Schölkopf, A. J. Smola // The annals of statistics. — 2008. — С. 1171—1220.

48. Co2 forest: Improved random forest by continuous optimization of oblique splits // arXiv preprint arXiv:1506.06155. — 2015.

49. Large margin methods for structured and interdependent output variables / I. Tsochantaridis [и др.] // Journal of machine learning research. — 2005. — Т. 6, Sep. — С. 1453—1484.

50. *Yuille, A.* The convex concave procedure (CCCP) / A. Yuille, A. Rangarajan // Advances in neural information processing system. — 2002. — Т. 14. — С. 219—236.

51. *DeSalvo, G.* Random Composite Forests. / G. DeSalvo, M. Mohri // AAAI. — 2016. — С. 1540—1546.

52. *Hehn, T. M.* End-to-end learning of decision trees and forests / T. M. Hehn, J. F. Kooij, F. A. Hamprecht // International Journal of Computer Vision. — 2020. — Т. 128, № 4. — С. 997—1011.

53. *Irsoy, O.* Autoencoder trees / O. Irsoy, E. Alpaydin // Asian Conference on Machine Learning. — 2016. — С. 378—390.

54. Theory of the backpropagation neural network. / R. Hecht-Nielsen [и др.] // Neural Networks. — 1988. — Т. 1, Supplement—1. — С. 445—448.

55. *Yang, B.-B.* Weighted oblique decision trees / B.-B. Yang, S.-Q. Shen, W. Gao // Proceedings of the AAAI Conference on Artificial Intelligence. Т. 33. — 2019. — С. 5621—5627.

56. *Carreira-Perpinán, M. A.* Alternating optimization of decision trees, with application to learning sparse oblique trees / M. A. Carreira-Perpinán, P. Tavallali // Advances in neural information processing systems. — 2018. — Т. 31.

57. *Kumar, M. A.* A hybrid SVM based decision tree / M. A. Kumar, M. Gopal // Pattern Recognition. — 2010. — Т. 43, № 12. — С. 3977—3987.

58. *Rabiner, L.* An introduction to hidden Markov models / L. Rabiner, B. Juang // ieee assp magazine. — 1986. — Т. 3, № 1. — С. 4—16.

59. *BOTTOU, L.* UNE APPROCHE THEORIQUE DE L'APPRENTISSAGE CONNEXIONNISTE ET APPLICATIONS A LA RECONNAISSANCE DE LA PAROLE : дис. ... канд. / BOTTOU LEON.. — 1991.

60. *McCallum, A.* Maximum Entropy Markov Models for Information Extraction and Segmentation. / A. McCallum, D. Freitag, F. C. Pereira // Icml. Т. 17. — 2000. — С. 591—598.

61. *Lafferty, J.* Conditional random fields: Probabilistic models for segmenting and labeling sequence data / J. Lafferty, A. McCallum, F. Pereira // Proceedings of the eighteenth international conference on machine learning, ICML. Т. 1. — 2001. — С. 282—289.

62. *Воронцов, К. В.* Комбинаторная теория надёжности обучения по прецедентам / К. В. Воронцов // Дис. на соиск. уч. степени д. ф.-м. н. М.: ВЦ РАН. — 2010.

63. *Taskar, B.* Max-margin Markov networks / B. Taskar, C. Guestrin, D. Koller // Advances in neural information processing systems. — 2004. — С. 25—32.

64. *Krizhevsky, A.* 2012 AlexNet / A. Krizhevsky, I. Sutskever, G. E. Hinton // Adv. Neural Inf. Process. Syst. — 2012. — С. 1—9.

65. *Ronneberger, O.* U-net: Convolutional networks for biomedical image segmentation / O. Ronneberger, P. Fischer, T. Brox // International Conference on Medical image computing and computer-assisted intervention. — Springer. 2015. — С. 234—241.

66. *Elman, J. L.* Finding structure in time / J. L. Elman // Cognitive science. — 1990. — Т. 14, № 2. — С. 179—211.

67. *Jordan, M.* Attractor dynamics and parallelism in a connectionist sequential machine / M. Jordan // Proc. of the Eighth Annual Conference of the Cognitive Science Society (Erlbaum, Hillsdale, NJ), 1986. — 1986.

68. *Lang, K. J.* A time-delay neural network architecture for isolated word recognition / K. J. Lang, A. H. Waibel, G. E. Hinton // Neural networks. — 1990. — Т. 3, № 1. — С. 23—43.

69. *Jaeger, H.* The "echo state" approach to analysing and training recurrent neural networks-with an erratum note / H. Jaeger // Bonn, Germany: German National Research Center for Information Technology GMD Technical Report. — 2001. — Т. 148, № 34. — С. 13.

70. *Hammer, B.* On the approximation capability of recurrent neural networks / B. Hammer // Neurocomputing. — 2000. — Т. 31, № 1—4. — С. 107—123.

71. Identification and forecasting of large dynamical systems by dynamical consistent neural networks / H. Zimmermann [и др.] // New Directions in Statistical Signal Processing: From Systems to Brain. — 2006. — С. 203—242.

72. *Werbos, P. J.* Backpropagation through time: what it does and how to do it / P. J. Werbos // Proceedings of the IEEE. — 1990. — Т. 78, № 10. — С. 1550—1560.

73. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies / S. Hochreiter [и др.]. — 2001.

74. *Bengio, Y.* Learning long-term dependencies with gradient descent is difficult / Y. Bengio, P. Simard, P. Frasconi // IEEE transactions on neural networks. — 1994. — Т. 5, № 2. — С. 157—166.

75. *Schmidhuber, J.* Learning complex, extended sequences using the principle of history compression / J. Schmidhuber // Neural Computation. — 1992. — Т. 4, № 2. — С. 234—242.

76. *Hochreiter, S.* Long short-term memory / S. Hochreiter, J. Schmidhuber // Neural computation. — 1997. — Т. 9, № 8. — С. 1735—1780.

77. Empirical evaluation of gated recurrent neural networks on sequence modeling / J. Chung [и др.] // arXiv preprint arXiv:1412.3555. — 2014.

78. Attention is all you need / A. Vaswani [и др.] // Advances in neural information processing systems. — 2017. — Т. 30.

79. *Tang, B.* Probabilistic transformer for time series analysis / B. Tang, D. S. Matteson // Advances in Neural Information Processing Systems. — 2021. — Т. 34. — С. 23592—23608.

80. *Esser, P.* Taming transformers for high-resolution image synthesis / P. Esser, R. Rombach, B. Ommer // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2021. — С. 12873—12883.

81. *Srivastava, R. K.* Training very deep networks / R. K. Srivastava, K. Greff, J. Schmidhuber // Advances in neural information processing systems. — 2015. — С. 2377—2385.

82. Dropout: a simple way to prevent neural networks from overfitting / N. Srivastava [и др.] // The Journal of Machine Learning Research. — 2014. — Т. 15, № 1. — С. 1929—1958.

83. *Courbariaux, M.* Training deep neural networks with binary weights during propagations. arXiv preprint / M. Courbariaux, Y. Bengio, J.-P. B. David // arXiv preprint arXiv:1511.00363. — 2015.

84. *Chen, T.* Xgboost: A scalable tree boosting system / T. Chen, C. Guestrin // Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. — ACM. 2016. — С. 785—794.

85. *Zhou, Z.-H.* Deep forest: Towards an alternative to deep neural networks / Z.-H. Zhou, J. Feng // arXiv preprint arXiv:1702.08835. — 2017.

86. *Utkin, L. V.* A Deep Forest for Transductive Transfer Learning by Using a Consensus Measure / L. V. Utkin, M. A. Ryabinin // Conference on Artificial Intelligence and Natural Language. — Springer. 2017. — С. 194—208.

87. *Chen, Y.-H.* Improving deep forest by exploiting high-order interactions / Y.-H. Chen, S.-H. Lyu, Y. Jiang // 2021 IEEE International Conference on Data Mining (ICDM). — IEEE. 2021. — С. 1030—1035.

88. MLbase: A Distributed Machine-learning System. / T. Kraska [и др.] // Cidr. Т. 1. — 2013. — С. 2—1.

89. Resilient distributed datasets: A {Fault-Tolerant} abstraction for {In-Memory} cluster computing / M. Zaharia [и др.] // 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12). — 2012. — С. 15—28.

90. Mesos: A Platform for {Fine-Grained} Resource Sharing in the Data Center / B. Hindman [и др.] // 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11). — 2011.

91. Pregel: a system for large-scale graph processing / G. Malewicz [и др.] // Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. — 2010. — С. 135—146.

92. Distributed graphlab: A framework for machine learning in the cloud / Y. Low [и др.] // arXiv preprint arXiv:1204.6078. — 2012.

93. Graphlab: A new framework for parallel machine learning / Y. Low [и др.] // arXiv preprint arXiv:1408.2041. — 2014.

94. *Gillick, D.* Mapreduce: Distributed computing for machine learning / D. Gillick, A. Faria, J. DeNero // Berkley, Dec. — 2006. — Т. 18.

95. Planet: massively parallel learning of tree ensembles with mapreduce / B. Panda [и др.]. — 2009.

96. Distributed decision tree learning for mining big data streams / A. Murdopo [и др.] // Master of Science Thesis, European Master in Distributed Computing. — 2013.

97. Stochastic gradient boosted distributed decision trees / J. Ye [и др.] // Proceedings of the 18th ACM conference on Information and knowledge management. — 2009. — С. 2061—2064.

98. *Li, B.* Ensemble of fast learning stochastic gradient boosting / B. Li, Q. Yu, L. Peng // Communications in Statistics-Simulation and Computation. — 2022. — Т. 51, № 1. — С. 40—52.

99. Xgboost: extreme gradient boosting / T. Chen [и др.] // R package version 0.4-2. — 2015. — Т. 1, № 4. — С. 1—4.

100. *Dorogush, A. V.* CatBoost: gradient boosting with categorical features support / A. V. Dorogush, V. Ershov, A. Gulin // arXiv preprint arXiv:1810.11363. — 2018.

101. *Дружков, П.* Реализация параллельного алгоритма обучения в методе градиентного бустинга деревьев решений для систем с распределенной памятью / П. Дружков, А. Половинкин // Параллельные вычислительные технологии 2012 (ПАВТ'2012). Новосибирск, 26-30 марта 2012 г.–Новосибирск. — 2012. — С. 459—465.

102. Real-time distributed-random-forest-based network intrusion detection system using Apache spark / H. Zhang [и др.] // 2018 IEEE 37th international performance computing and communications conference (IPCCC). — IEEE. 2018. — С. 1—7.

103. *Девяткин, Д.* Построение случайных лесов деревьев решений с применением ядерных разделителей / Д. Девяткин // XI Международная научно-практическая конференция (ИММВ-2022). Т. 1. — РАИИ. 2022. — С. 213—223.

104. Hidden markov support vector machines / Y. Altun, I. Tsochantaridis, T. Hofmann [и др.] // ICML. Т. 3. — 2003. — С. 3—10.

105. *Abdiansah, A.* Time complexity analysis of support vector machines (SVM) in LibSVM / A. Abdiansah, R. Wardoyo // International journal computer and application. — 2015. — Т. 128, № 3. — С. 28—34.

106. *McDiarmid, C.* Concentration / C. McDiarmid // Probabilistic methods for algorithmic discrete mathematics. — Springer, 1998. — С. 195—248.

107. Global refinement of random forest / S. Ren [и др.] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2015. — С. 723—730.

108. Learning multiple layers of features from tiny images / A. Krizhevsky, G. Hinton [и др.]. — 2009.

109. *Smirnov, I.* YouTube Channels Dataset / I. Smirnov // http://keen.isa.ru/youtub. — 2022.

110. *Murphy, P. M.* UCI repository of machine learning databases / P. M. Murphy // ftp:/pub/machine-learning-databaseonics. uci. edu. — 1994.

111. Titanis: A tool for intelligent text analysis in social media / I. Smirnov [и др.] // Russian Conference on Artificial Intelligence. — Springer. 2021. — С. 232—247.

112. Исследование характеристик текстов противоправного содержания / М. И. Ананьева [и др.] // Труды Института системного анализа Российской академии наук. — 2017. — Т. 67, № 3. — С. 86—97.

113. Feature Engineering for Depression Detection in Social Media. / M. Stankevich [и др.] // ICPRAM. — 2018. — С. 426—431.

114. *Koltsova, O.* An opinion word lexicon and a training dataset for russian sentiment analysis of social media / O. Koltsova, S. Alexeeva, K. S. // Computational Linguistics and Intellectual Technologies: Materials of DIALOGUE. — 2016. — С. 277—287.

Список рисунков

Список таблиц