Московский государственный университет имени М.В. Ломоносова
Факультет Вычислительной математики и кибернетики
Кафедра Математических методов прогнозирования

РЫЖКОВ Александр Михайлович

Композиции алгоритмов, основанные на случайном лесе

ДИПЛОМНАЯ РАБОТА

Научный руководитель:
д.ф-м.н., профессор
А.Г.Дьяконов

Москва, 2015

Содержание

Аннотация

Данная работа посвящена исследованию композиций решающих деревьев и их модификаций в задачах машинного обучения. За основу взят алгоритм случайного леса, как наиболее универсальный инструмент для решения широкого спектра прикладных задач.

В рамках работы был проведен анализ деревьев решений, как структурных элементов леса, и методов их комбинирования, выявлены основные недостатки самого алгоритма и его реализаций в прикладных пакетах в текущих реалиях анализа больших данных и приведены методы их устранения. Также были описаны основные типы задач, в которых обосновано применение рассматриваемых модификаций, и проведены необходимые предварительные исследования на модельных данных.

Кроме того, эффективность и качество работы полученных композиций были протестированы на реальных данных международных соревнований. В работе представлено экспериментальное исследование и сравнительный анализ построенных моделей. Полученные в работе результаты можно использовать при решении прикладных задач, получая высокое качество работы даже при малой размерности исходного обучающего множества.

1 Введение

В настоящее время алгоритмы машинного обучения, использующие деревья принятия решений, очень распространены и универсальны для большинства прикладных задач. С их помощью можно решать как задачи разделения объектов на два и более классов (задача классификации), так и прогнозировать вещественный отклик для каждого объекта (задача регрессии). Составляя композицию из нескольких построенных деревьев, комбинируя ответы каждого из них, можно получать устойчивое и гораздо более качественное решение, чем у многих других алгоритмов.

В 2001 году Лео Брейман и Адель Катлер предложили мировому сообществу алгоритм *Random Forest* [7], заключающийся в использовании комитета (ансамбля) решающих деревьев. Этот алгоритм произвел настоящий фурор в области машинного обучения, поскольку он серьезно превосходил существующие аналоги, практически не переобучался с ростом числа деревьев в композиции, получал достаточно устойчивое решение.

Однако ни что не идеально и с течением времени стали появляться примеры задач, на которых Random Forest работает не так хорошо, как этого бы хотелось, поэтому в данной работе исследуются несколько вариантов улучшения построения композиции, состоящей из модифицированных решающих деревьев.

2 Теоретическая часть

2.1 Общая постановка задачи

В данной работе рассматриваются две постановки задачи — построение алгоритма для задачи классификации и для задачи регрессии. Но в силу их сходства здесь будет приведена только постановка для задачи классификации.

В качестве исходных данных для поставленной задачи используются размеченные пары объектов $\{(x_i, y_i)\}_{i=1}^{N}$, где x_i — признаковое описание объекта, а y_i — метка класса (значение из дискретного множества, мощность которого равна числу классов в рассматриваемой задаче). Требуется построить композицию решающих деревьев, обладающую следующими свойствами:

- итоговая композиция с высокой точностью классифицирует новые объекты z_i, метка класса которых не известна;
- при помощи композиции получено достаточно устойчивое решение поставленной задачи, т.е. при небольшом изменении параметров построенного решения не происходит серьезного снижения качества работы.

Более формально постановки задач с описанием выборок и рассматриваемых функционалов будут описаны в разделе вычислительных экспериментов.

2.2 Алгоритм случайного леса

Начать описание алгоритма случайного леса следует с дерева принятия решений, как с основного структурного элемента леса, ведь именно от того, каким образом построено каждое дерево, серьезно зависит качество работы и устойчивость всей финальной композиции.

Рассмотрим размеченную выборку объектов $\{(x_i, y_i)\}_{i=1}^{N}$, где $x_i \in \mathbb{R}^2$ — признаковое описание объекта в двумерном пространстве, а $y_i \in \{0, 1\}$ — метка класса:

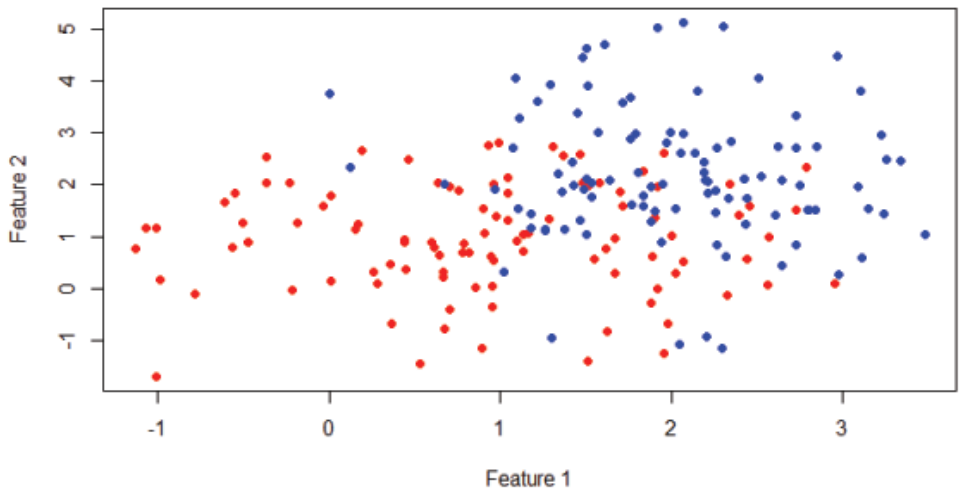


Рис. 1: Исходные данные задачи классификации

Несмотря на то, что в середине объекты разных классов сильно перемешаны, при помощи дерева решений с такой выборкой достаточно удобно работать: на каждом шаге необходимо выбирать признак и значения порога, по которому происходит оптимальное по заданному критерию разбиение. При решении прикладных задач часто используются следующие критерии:

- для задач классификации iGain:

$$\text{iGain}(S) = H(S) - \sum_{v \in \{L,R\}} \frac{|S_v|}{|S|} H(S_v),$$

$$H(S) = -\sum_{c \in C} p_c \log_2(p_c),$$

где C — множество классов рассматриваемой задачи, а p_c — вероятность класса c для множества объектов S;

- для задач регрессии аналогичный iGain критерий с использованием дисперсий:

$$\text{iGain}(S) = |S|\text{Var}(S) - \sum_{v \in \{L,R\}} |S_v|\text{Var}(S_v),$$

где $\text{Var}(S)$ — дисперсия откликов объектов из множества S.

При каждом делении все объекты делятся на две более мелкие группы, т.е. рассматриваемая в каждом из узлов задача разбивается на две более мелкие подзадачи. Заданием максимального числа объектов в вершине-листе дерева устанавливается один из возможных критериев останова для алгоритма.

Полученное в итоге работы алгоритма дерево можно изобразить несколькими способами, как показано на рисунке 2. Таким образом, можно достаточно качественно классифицировать рассматриваемую выборку объектов при помощи всего одного дерева решений, если в качестве ответа для тестового объекта, попавшего в ячейку A_i, выдавать номер наиболее часто встречающегося в этой ячейке класса.

Однако в реальных задачах часто встречаются погрешности в измерениях и объекты-выбросы, которые серьезно портят качество классификации одним конкретным деревом решений. Поэтому перед построением каждого нового дерева происходит сэмплирование с повторениями новой выборки $\{(x_i^k, y_i^k)\}_{i=1}^N$ из $\{(x_i, y_i)\}_{i=1}^N$, на которой происходит обучение дерева с номером k. После построения всех деревьев каждый тестовый объект z_i получает в качестве промежуточного ответа вектор меток, присвоенных ему каждым деревом, который преобразуется в финальную метку по методу простого голосования.

2.3 Недостатки случайного леса и методы их устранения

Помимо всех заявленных ранее преимуществ алгоритм построения случайного леса имеет и ряд недостатков:

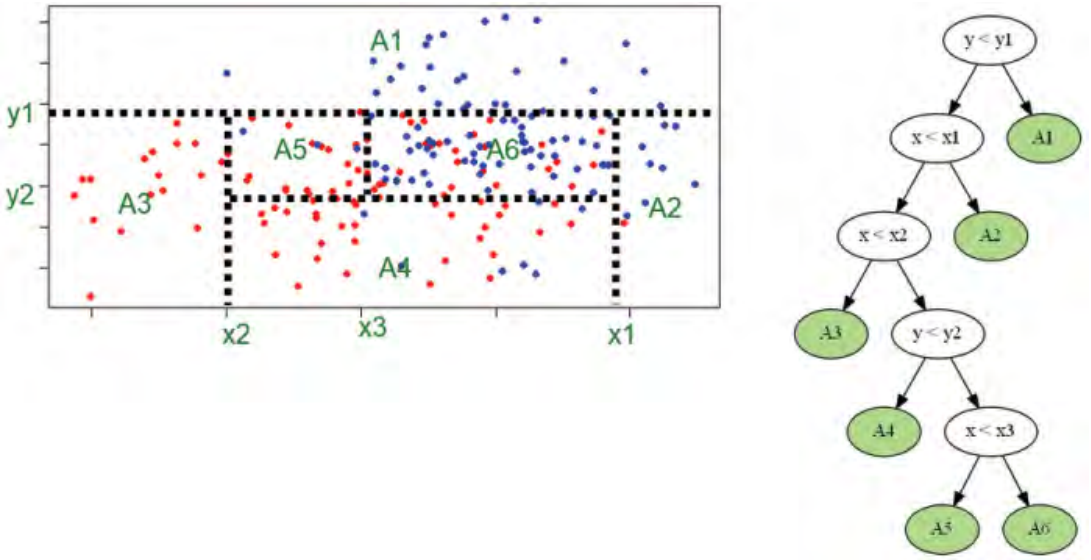


Рис. 2: Различные варианты изображения решающих деревьев

- качество полученной композиции сильно ухудшается в случае, когда на вход алгоритму подается мало размеченных данных, поскольку деревья не могут качественно выявить скрытые в данных закономерности;

- по мере приближения к листьям разбиения в узлах каждого дерева становятся все менее статистически обоснованными из-за малого числа рассматриваемых объектов;

- в процессе увеличения числа используемых деревьев уменьшается *отклонение (bias)*, но повышается *дисперсия (variance)* [12]. При этом также возрастает гибкость модели, что позволяет ей настроиться на выбросы в данных и способствует снижению обобщающей способности финальной композиции на стадии тестирования.

В качестве недостатков реализации Random Forest в различных пакетах алгоритмов машинного обучения хотелось бы отметить:

- практически не поддерживается работа с категориальными признаками (как представленными в виде строк, так и в виде целых чисел). В случае с представлением категориального признака в виде целых чисел, уровни 1 и 2 считаются гораздо более близкими, чем 1 и 10, что может противоречить логике — например, если значениями этого признака являются цвета кузова автомобиля;

- медленно обрабатываются пропущенные значения в тренировочной выборке и еще хуже в тестовой:
    - в тренировочной выборке пропуски в вещественных признаках заполняются медианой (более устойчивой оценкой по сравнению со средним значением), в категориальных признаках — модой;
    - в тестовой выборке проходят запуски алгоритма с различными вариантами заполнения и определяется лучший кандидат-замена для пропуска.

В качестве идей по улучшению композиции, основанной на случайном лесе, можно отметить:

- создание композиций случайных лесов (аналог смеси экспертов с использованием априорной оценки плотности классов по тренировочной выборке), что позволит улучшить работу финальной модели на всей генеральной совокупности данных;
- использование более сложных критериев разделения областей для повышения обобщающей способности алгоритма (модель гораздо меньше подстраивается под выбросы за счет более качественного дробления исходной задачи на верхних уровнях дерева);
- установка ограничений на деление объектов в узлах дерева при помощи порога на iGain — удаление плохо обоснованных делений в узлах, близких к листам дерева — и использование регрессии в терминальных узлах позволяет снизить возможность переобучения (явления, когда построенная модель хорошо объясняет примеры из обучающей выборки, но относительно плохо работает на тестовых объектах) финальной композиции;
- семплирование дополнительного набора размеченных объектов при помощи специализированных композиций решающих деревьев решает проблему плохо размеченных данных;
- деление объектов во внутренних узлах дерева более чем на две группы (например, как в алгоритме опорных векторов [8], — отступ больше 1, отступ меньше

-1, точки неуверенности), при помощи которого можно получить деревья меньшей высоты и, как следствие, повысить интерпретируемость модели;

- изменение структуры голосования деревьев и переход к анализу компетентности деревьев на глобальной совокупности объектов позволяют сконцентрировать внимание алгоритма на наиболее «сложных» для классификации объектах;
- использование матрицы ценности (*cost matrix*) в случае несбалансированности классов для повышения качества работы итоговой модели;
- нахождение наиболее частых пар признаков, появляющихся в деревьях на уровнях, близких корню, для наглядной визуализации структуры имеющихся данных;
- добавление весов объектов на основе априорных оценок плотности для выявления «трудных» объектов и выбросов;
- обработка выбросов — похожая идея используется для алгоритма построения композиций *AdaBoost* [11]: обучение запускается примерно на 10–15 итераций, после чего удаляются объекты с большими весами и обучение запускается повторно на обновленной выборке.

2.4 Алгоритм эффективного построения деревьев

При построении дерева решений на каждом шаге производится ряд похожих действий, которые зачастую повторяют друг друга, поэтому работа алгоритма построения дерева решений не всегда в достаточной степени эффективна. В данном разделе работы приводятся некоторые идеи по улучшению быстродействия стандартного алгоритма построения решающего дерева, в следствие чего увеличивается и производительность общего алгоритма построения композиции.

Для разбиения выборки, дошедшей до определенного листа решающего дерева, на две части необходимо случайным образом выбрать множество I, состоящее из одного (или более) признака, на основе которого будет выполняться деление, т.е.

нужно решить следующую задачу:

$$\begin{cases} S_L = \{x \in X_v | x_i \leqslant \tau, i \in I\} \\ S_R = \{x \in X_v | x_i > \tau, i \in I\} \\ \text{iGain}(S) \to \max\limits_{i \in I, \tau \in \mathbb{R}} \end{cases}$$

При решении поставленной задачи возникают однотипные задачи выбора порога τ для некоторого фиксированного признака с номером i, поэтому изначально необходимо подготовить таблицу возможных разбиений по каждому из признаков для экономии времени и загрузки нужных пороговых значений за константное время, что значительно улучшает эффективность алгоритма.

Более того, общий алгоритм построения дерева решений может быть организован на основе одной из следующих парадигм — построения дерева в ширину [15] или в глубину [10], каждая из которых обладает своими преимуществами и недостатками. Так, например, при построении дерева в глубину в каждый момент времени обрабатывается всего один узел и для него выбирается наилучшее разбиение на основе дошедших до этого узла данных, в то время как при построении в ширину одновременно строится целый уровень дерева решений. Более наглядно процесс построения дерева в каждой из парадигм изображен на рисунке 3.

Еще больше повысить эффективность построение отдельного дерева решений становится возможным за счет комбинирования двух парадигм построения — используя на первых уровнях алгоритм построения в ширину, строим дерево до того момента, пока в некотором узле не останется число элементов, меньшее заданного при запуске алгоритма порога. Найденный узел запоминается и алгоритм продолжает свою работу для остальных узлов, пока не останется ни одного текущего не сохраненного узла. После этого все сохраненные узлы достраиваются до листьев при помощи алгоритма построения дерева в глубину. Такой подход обладает следующим рядом преимуществ, по сравнению с каждой из парадигм в отдельности:

- при построении в ширину происходит не более, чем D (заданная высота дерева) проходов пути от корня до листьев данными исходной задачи;
- появляется возможность обойтись без использования рекурсии при построении верхних уровней дерева, тем самым экономя память на хранение необходимых данных и модели, а не стека рекурсии;

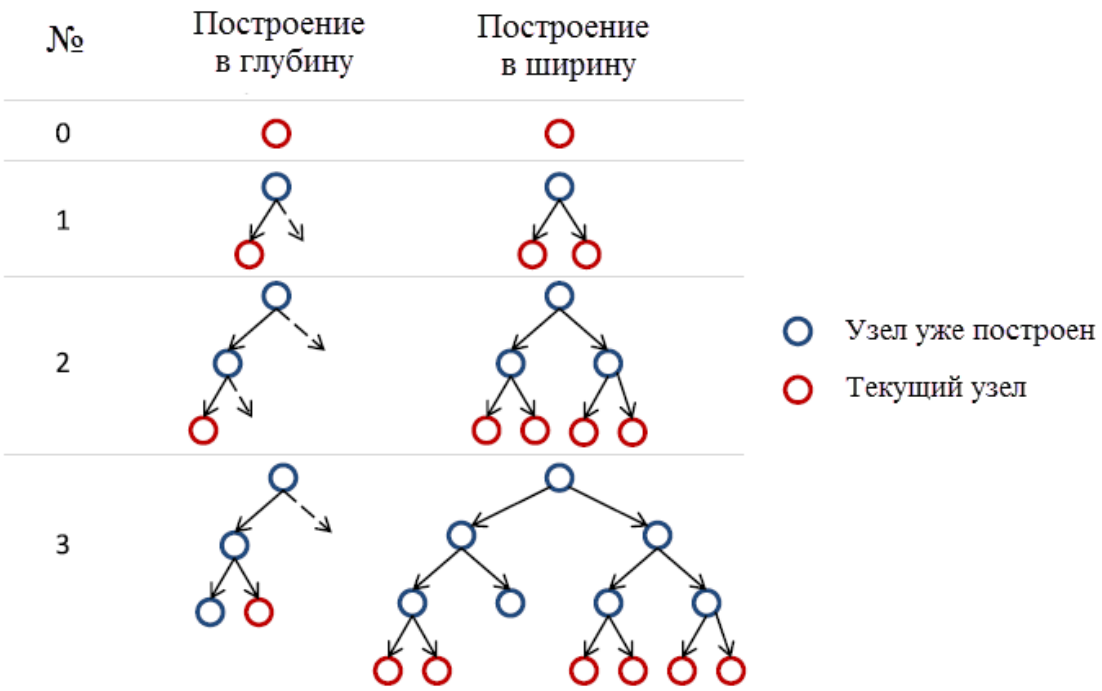


Рис. 3: Различные парадигмы построения деревьев решений

- алгоритм построения дерева в ширину достаточно просто реализовать в виде параллельной программы для нескольких узлов CPU (одной из таких реализаций является технология PLANET [17] от компании Google на основе Map-Reduce [9]);
- накладные расходы, необходимые для использования алгоритма построения дерева решений в ширину, значительно сокращаются, если для узлов с небольшими объемами данных достраивать поддеревья в глубину.

2.5 Ансамблирование алгоритмов

В алгоритме случайного леса в качестве композиции используется простое усреднение ответов всех построенных деревьев, однако это не единственный и не всегда самый качественный и устойчивый вариант создания ансамбля базовых алгоритмов $b_t(x)$.

Существует несколько наиболее известных методов объединения базовых алгоритмов в композиции:

- простое голосование (Simple Voting):

$$Y_{ans}(x) = \frac{1}{T}\sum_{t=1}^{T} b_t(x);$$

- взвешенное голосование (Weighted Voting):

$$Y_{ans}(x) = \sum_{t=1}^{T} w_t b_t(x),$$

$$\sum_{t=1}^{T} w_t = 1,\ w_t \geqslant 0;$$

- смесь экспертов (Mixture of Experts [14]):

$$Y_{ans}(x) = \sum_{t=1}^{T} w_t(x) b_t(x),$$

$$\sum_{t=1}^{T} w_t(x) = 1,\ \forall x \in X.$$

Очевидным является факт, что простое голосование — это лишь частный случай взвешенного голосования, а взвешенное голосование является частным случаем смеси экспертов. Также стоит отметить, что из-за наличия большого числа степеней свободы обучение смеси экспертов происходит значительно дольше других алгоритмов построения композиции, поэтому их практическая применимость обоснована только в случае априорной информации о функциях компетенции $g_t(x)$, которые чаще всего определяются:

- признаком $f(x)$:

$$w_t(x;\alpha,\beta) = \sigma(\alpha f(x) + \beta),\ \alpha,\beta \in \mathbb{R};$$

- направлением $\alpha \in \mathbb{R}^n$:

$$w_t(x;\alpha,\beta) = \sigma(x^T\alpha + \beta),\ \alpha \in \mathbb{R}^n, \beta \in \mathbb{R};$$

- расстоянием до $\alpha \in \mathbb{R}^n$

$$w_t(x;\alpha,\beta) = \exp(-\beta||x-\alpha||^2),\ \alpha \in \mathbb{R}^n, \beta \in \mathbb{R};$$

- более сложными способами (при помощи оценки плотности распределения данных и т.п.).

В приведенных выше формулах функций компетенции $\sigma(z) = \dfrac{1}{1+e^{-z}}$ — сигмоида.

В случае наличия выборки, размеры которой по числу объектов и/или признаков значительно превышают объем оперативной памяти ПК, используются алгоритмы бэггинга [6] и случайных подпространств [13], которые позволяют обучать базовые алгоритмы на некоторых случайно выбранных подмножествах объектов и признаков. Более подробно применение такого рода алгоритмов будет рассмотрено во второй части вычислительных экспериментов.

2.6 Зависимость качества от структурных параметров алгоритма

Качество работы любого алгоритма на тестовой выборке сильно зависит от используемых при его настройке гиперпараметров, поэтому важным аспектом этого процесса является проведение следующих экспериментов:

- проведение анализа различных разделяющих критериев, использующихся для дробления текущей задачи, рассматриваемой в узле дерева, на более мелкие подзадачи;

- исследование зависимости качества работы алгоритма от разного числа признаков, используемых при построении узла;

- анализ связи качества решения задачи с порогом на число объектов, попадающих в лист дерева (над этими наблюдениями может строиться регрессия, которая корректирует конечные ответы).

Введем некоторые дополнительные обозначения:

- $X_v \in \mathbb{R}^{n_v \times K}$ — совокупность объектов обучающей выборки, дошедших до некоторого рассматриваемого внутреннего узла v дерева;

- $x = (x_1, \ldots, x_K) \in X_v$ — некоторый объект из множества X_v, описываемый набором из K признаков; вектор-строка;

- $w \in \mathbb{R}^{K \times 1}$ — вектор весов признаков в линейной комбинации;
- $W \in \mathbb{R}^{K \times K}$ — матрица весов признаков для построения разделяющего критерия второго порядка;
- $\tau \in \mathbb{R}$ — порог, с которым сравнивается полученное значение выражения внутри разделяющего критерия;
- $$[A] = \begin{cases} 1, & \text{если выражение A истинно;} \\ 0, & \text{иначе.} \end{cases}$$

В каждом из узлов дерева решений присутствует разделяющий критерий одного из следующих видов:

- «решающий пень»:
$$C_1(x, \tau) = [x_i \leqslant \tau];$$
- линейный (ориентированный) критерий:
$$C_2(x, w, \tau) = [xw \leqslant \tau];$$
- нелинейный критерий второго порядка:
$$C_3(x, w, W, \tau) = [xWx^T + xw \leqslant \tau];$$
- более сложные разделяющие критерии.

Примеры разделяющих поверхностей, получаемых при помощи каждого из указанных выше критериев, изображены на рисунке 4.

В проведенном эксперименте на модельных данных на плоскости с различными разделяющими критериями была выявлена проблема «решающих пней», оси которых параллельны осям координат — при таком способе деления объектов в узлах в большинстве случаев деревья получаются значительно большей высоты и требуется гораздо большее их число для достижения сравнимого качества, чем в случае с произвольно ориентированными линейными (а тем более нелинейными) разделяющими

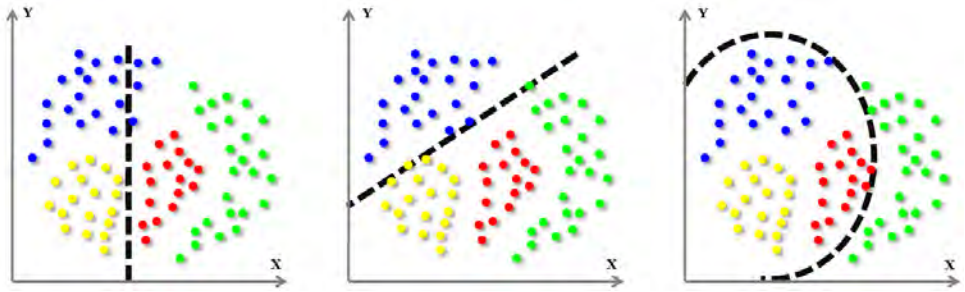


Рис. 4: Различные разделяющие критерии (слева направо: «решающий пень», линейный и нелинейный критерии)

поверхностями. Описанный результат можно визуально оценить на рисунке 5. Также, нетрудно видеть, что с увеличением сложности критерия повышается гладкость поверхности, отделяющей объекты разных классов, которая положительно влияет на обобщающую способность алгоритмов, построенных на реальных данных.

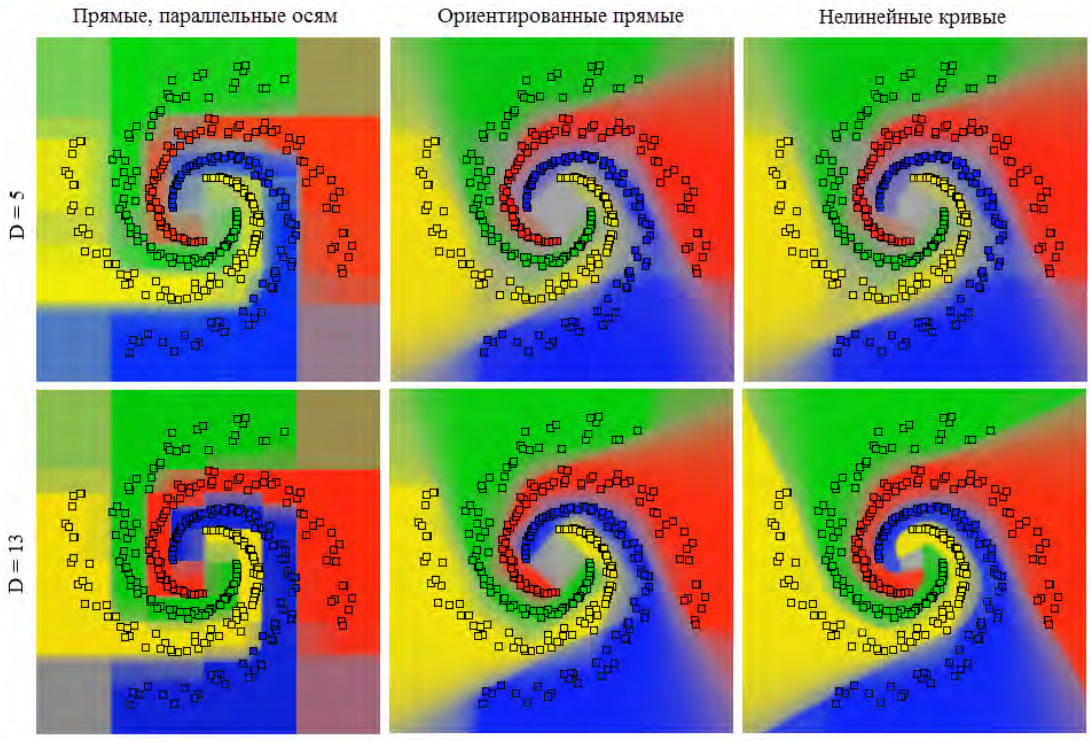


Рис. 5: Разбиения пространства при помощи различных разделяющих критериев в узлах деревьев случайного леса в зависимости от выбранной высоты дерева

При стандартном построении дерева решений, основанного на «решающих пнях», для каждого узла случайным образом выбирается признак, по которому будет происходить разбиение. Определим для такого случая переменную $\rho = 1$, которая будет

равна размерности подмножества признаков, выбранных для просмотра. Такой подход к построению узла можно обобщить следующим образом: для каждого узла дерева случайным образом выбирается $1 \leqslant \rho \leqslant K$ различных признаков, из порогов разбиения которых выбирается наилучшее разбиение объектов, дошедших от корня до рассматриваемого узла. При тестировании на модельных данных ярко проявляется следующая тенденция:

- при выборе низкого значения параметра ρ алгоритм показывает среднее качество, высокую скорость работы и минимально коррелированные деревья;
- при выборе среднего значения ρ алгоритм получает высокое качество, неплохую скорость работы и слабо коррелированные деревья — наилучший вариант, поскольку он сочетает в себе скорость и обобщающую способность, что позволяет говорить о его практической применимости для решения реальных задач;
- при выборе большого значения ρ алгоритм получает сравнимое с предыдущим вариантом качество, но очень медленную скорость работы и практически полностью идентичные деревья за счет многократного перебора большого количества одинаковых точек разбиения — такой вариант явно не стоит применять на практике.

Визуальные изменения деревьев с ростом значения параметра ρ можно отобразить на рисунке 6.

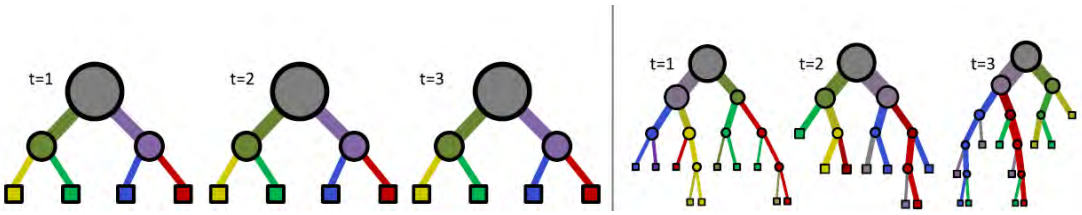


Рис. 6: Коррелированность деревьев при больших (слева) и малых (справа) значениях параметра ρ

Для более сложных критериев параметр ρ также будем использовать в качестве размерности подмножества признаков, выбранных для построения разделяющего критерия в фиксированном узле дерева, однако в данном случае он также будет равен эффективной размерности вектора весов w для линейного критерия и матрицы W в случае нелинейного критерия.

Использование порога на количество элементов в листах деревьев и запуск небольших задач регрессии на них приносит незначительный вклад в общий результат по сравнению с предыдущими экспериментальными изменениями. Более того, скорость работы существенно падает даже несмотря на то, что алгоритмы построения регрессионной зависимости работают на совсем небольших объемах данных. Переход от алгоритма построения решающего дерева «в глубину» к алгоритму «в ширину» или их комбинации (несколько верхних уровней строятся при помощи алгоритма «в ширину», а остальная часть достраивается при помощи алгоритма «в глубину») не дают новому варианту алгоритма приблизиться к предыдущей версии по скорости работы.

2.7 Анализ плотности распределения исходных данных

Стандартный алгоритм построения композиции деревьев решений испытывает сложности в задачах с малыми объемами размеченных исходных данных, принадлежащих к каждому классу, что негативно сказывается на качестве получаемого решения. Эту проблему можно преодолеть несколькими способами:

- обратиться к поставщику данных с просьбой дополнительного набора размеченных объектов для обучения моделей;
- оценить плотность распределения данных на генеральной совокупности и сэмплировать из него новый набор размеченных объектов необходимого размера.

Если первый вариант решения проблемы доступен далеко не всегда, то для использования второго достаточно применения композиции специализированных решающих деревьев, настроенной на всех имеющихся данных без учета меток классов. Для ее описания введем необходимые определения и докажем несколько теоретических фактов, используемых при построении каждого узла дерева.

Определение 1. *Плотность распределения случайной величины $x \in \mathbb{R}^n$, имеющей многомерное нормальное распределение, выглядит следующим образом:*

$$\phi(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)},$$

где $\mu \in \mathbb{R}^n$ — математическое ожидание, $\Sigma \in \mathbb{R}^{n \times n}$ — невырожденная неотрицательно определенная симметричная ковариационная матрица, $|\Sigma|$ — определитель ковариационной матрицы, Σ^{-1} — матрица, обратная к ковариационной

Определение 2. *Дифференциальная энтропия — средняя информация непрерывного источника:*

$$h(x) = -\int f(x) \log_2 f(x) dx,$$

где $f(x)$ — плотность распределения сигнала непрерывного источника как случайной величины

Основываясь на введенных определениях, докажем лемму:

Лемма 1. *Пусть $\bar{y} = (y_1, \ldots, y_n)^T \in \mathbb{R}^{n \times 1}$ — случайный вектор-столбец. Тогда если $\Sigma = \mathbb{E}[\bar{y}\bar{y}^T]$, то $\mathbb{E}[\bar{y}^T \Sigma^{-1} \bar{y}] = n$*

Доказательство. Определим

$$\Sigma = \mathbb{E}[\bar{y}\bar{y}^T] = \begin{pmatrix} | & | & \ldots & | \\ \bar{s}_1 & \bar{s}_2 & \ldots & \bar{s}_n \\ | & | & \ldots & | \end{pmatrix}$$

и

$$\Sigma^{-1} = \begin{pmatrix} — & \bar{a}_1^T & — \\ — & \bar{a}_2^T & — \\ — & \ldots & — \\ — & \bar{a}_n^T & — \end{pmatrix}$$

Тогда получаем, что $\bar{s}_i = \mathbb{E}[y_i \bar{y}]$ и $\bar{a}_i^T \bar{s}_j = \delta_{ij}$, где δ_{ij} — символ Кронекера:

$$\delta_{ij} = \begin{cases} 1, & \text{если } i = j; \\ 0, & \text{иначе} \end{cases}$$

Распишем при помощи введенных обозначений следующее выражение:

$$\bar{y}^T \Sigma^{-1} \bar{y} = \bar{y}^T \begin{pmatrix} — & \bar{a}_1^T & — \\ — & \bar{a}_2^T & — \\ — & \ldots & — \\ — & \bar{a}_n^T & — \end{pmatrix} \bar{y} = (y_1, \ldots, y_n) \begin{pmatrix} \bar{a}_1^T \bar{y} \\ \bar{a}_2^T \bar{y} \\ \vdots \\ \bar{a}_n^T \bar{y} \end{pmatrix} = y_1 \bar{a}_1^T \bar{y} + y_2 \bar{a}_2^T \bar{y} + \cdots + y_n \bar{a}_n^T \bar{y}$$

Вычислим математическое ожидание:

$$\mathbb{E}[\bar{y}^T\Sigma^{-1}\bar{y}] = \bar{a}_1^T\mathbb{E}[y_1\bar{y}] + \bar{a}_2^T\mathbb{E}[y_2\bar{y}] + \cdots + \bar{a}_n^T\mathbb{E}[y_n\bar{y}] = \bar{a}_1^T\bar{s}_1 + \bar{a}_2^T\bar{s}_2 + \cdots + \bar{a}_n^T\bar{s}_n = n,$$

поскольку $\forall i \in \{1, 2, \ldots, n\} : \ \bar{a}_i^T\bar{s}_i = \delta_{ii} \equiv 1$ ■

Используя лемму, докажем следующую теорему:

Теорема 1 (Дифференциальная энтропия многомерного нормального распределения)**.** *Пусть случайная величина $x \in \mathbb{R}^n$ имеет многомерное нормальное распределение с математическим ожиданием $\mu \in \mathbb{R}^n$ и ковариационной матрицей $\Sigma \in \mathbb{R}^{n\times n}$. Тогда:*

$$h(x) = \frac{1}{2}\log_2((2\pi e)^n|\Sigma|)$$

Доказательство. Запишем плотность многомерного нормального распределения:

$$\phi(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

По определению дифференциальной энтропии имеем:

$$h(x) = -\int \phi(x)\log_2\phi(x)dx = \int \phi(x)\left[\frac{1}{2}\log_2((2\pi)^n|\Sigma|) + \right.$$

$$\left. + \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\log_2 e\right]dx = \frac{1}{2}\log_2((2\pi)^n|\Sigma|) +$$

$$+\frac{1}{2}\log_2 e \underbrace{\mathbb{E}\Big[(x-\mu)^T\Sigma^{-1}(x-\mu)\Big]}_{=n \text{ по доказанной лемме}} = \frac{1}{2}\log_2((2\pi)^n|\Sigma|) + \frac{1}{2}n\log_2 e =$$

$$= \frac{1}{2}\log_2((2\pi e)^n|\Sigma|)$$

■

При построении дерева решений (как стандартного, так и модифицированного) наилучший критерий разбиения для внутреннего узла выбирался при помощи критерия iGain следующего вида:

$$\text{iGain}(S) = H(S) - \sum_{v\in\{L,R\}}\frac{|S_v|}{|S|}H(S_v),$$

где

$$H(S) = -\sum_{c\in C}p_c\log_2(p_c)$$

Если в качестве $H(S)$ подставить доказанную в теореме формулу для дифференциальной энтропии многомерного нормального распределения, то рассматриваемый критерий после упрощения примет вид:

$$\text{iGain}(S) = \log_2(|\Sigma(S)|) - \sum_{v \in \{L,R\}} \frac{|S_v|}{|S|} \log_2(|\Sigma(S_v)|),$$

где нотация $|\cdot|$ имеет смысл определителя для матриц и мощности для множеств.

При использовании описанного функционала выбора наилучшего разделяющего критерия в конкретном узле можно построить решающие деревья, в листьях которых будут подбираться многомерные нормальные распределения, описывающие распределение дошедших до них объектов. Из полученных распределений путем объединения описываемых листьями «ячеек» пространства можно построить плотность распределения объектов на всей генеральной совокупности (пример такого распределения для пространства $\mathbb{R}^2$ приведен на рисунке 7), а, усреднив плотности по всем деревьям композиции, — и итоговую оценку плотности распределения объектов на всем пространстве.

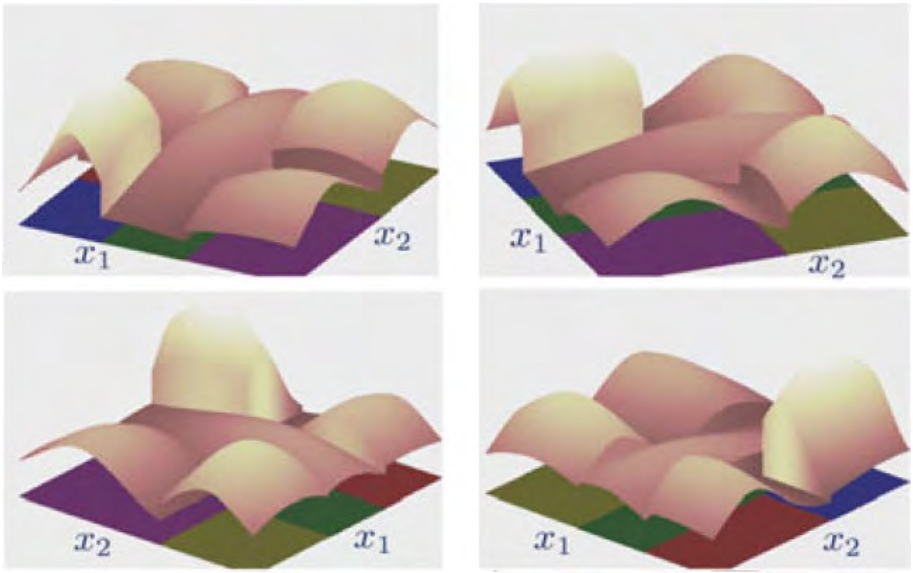


Рис. 7: Оценка распределения объектов деревом решений с различных ракурсов

Стоит отметить, что качество оценки плотности распределения при помощи такого метода напрямую зависит от выбранных структурных параметров алгоритма. Результаты работы на модельных данных при различных значениях параметров приведены на рисунках 8 и 9, где более светлыми тонами показаны области с более

высокими значениями плотности. Из полученных результатов можно сделать следующие выводы:

- увеличение числа деревьев в композиции позитивно влияет на качество получаемого ответа;
- используемую глубину деревьев необходимо подбирать для каждого отдельного случая:
  - в случае нескольких сгустков точек (как на рисунке 9) глубина дерева, равная 6 уровням, способствует проявлению эффекта переобучения при малом числе деревьев. Использование глубоких деревьев в этой ситуации оправдано только при большом числе деревьев, объединенных в единую композицию;
  - в случае наличия достаточно мелких деталей, которые важно учитывать (как на рисунке 8), использование более глубоких деревьев, чем для сгустков, полностью оправдано.

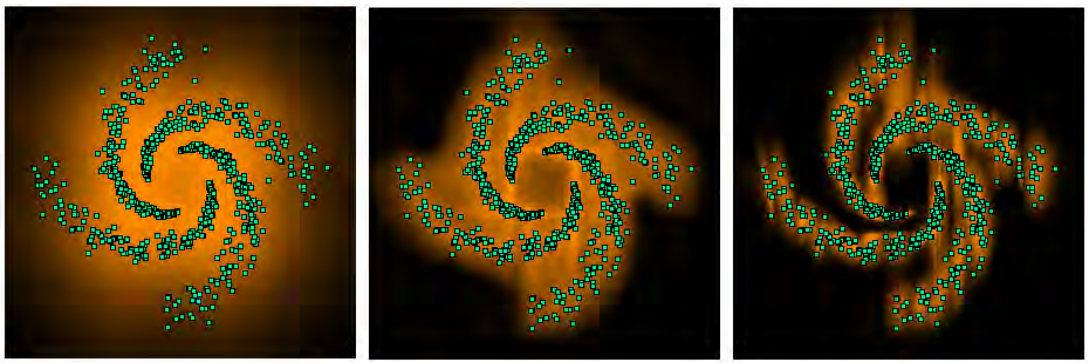


Рис. 8: Результаты работы алгоритма восстановления плотности при различной глубине деревьев (слева направо: глубина деревьев равна двум, четырем и шести уровням соответственно)

Результаты экспериментов по сравнению описанного алгоритма восстановления плотности с более известными аналогами, такими как метод парзеновского окна [18] и K ближайших соседей [4, 16], приведены на рисунке 10. Из них видно явное превосходство алгоритма, основанного на случайном лесе, над своими конкурентами — в случае наложения исходной выборки на каждую из восстановленных плотностей

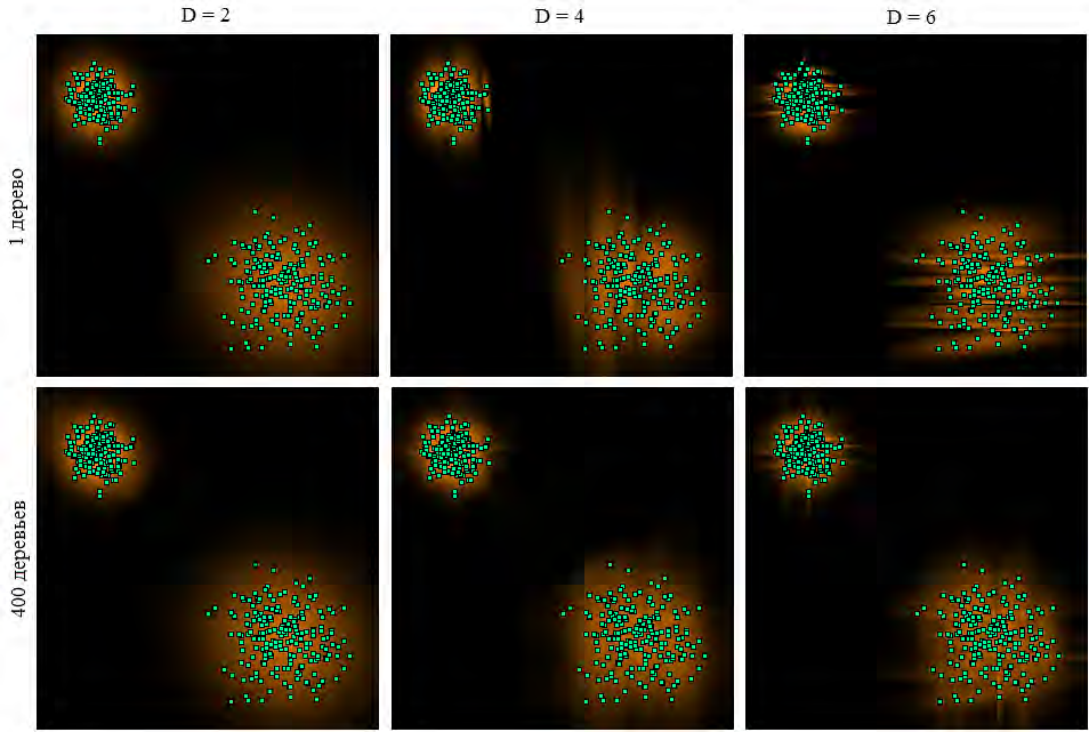


Рис. 9: Результаты работы алгоритма восстановления плотности при различной глубине деревьев и их количестве в композиции

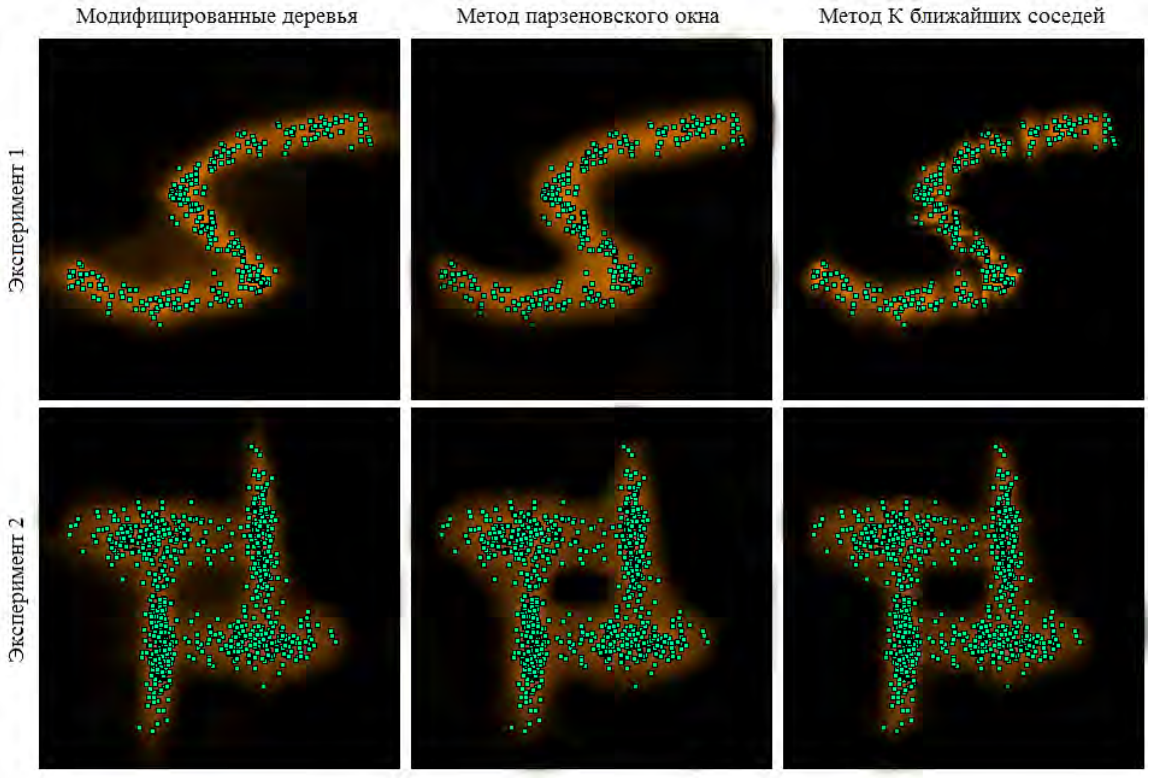


Рис. 10: Сравнение результатов работы описанного алгоритма с более известными методами восстановления плотности

видны яркие пятна на местах сгустков точек и чуть более темные и рассеянные логические переходы, скрытые в данных, которые почти не улавливают другие алгоритмы.

Таким образом, в случае необходимости генерации дополнительного набора размеченных данных следует применять композицию деревьев, оценивающих плотность распределения объектов на генеральной совокупности. Этот метод позволяет не только получить качественную оценку искомой плотности, но и обнаружить возможные взаимодействия между отдельными сгустками точек обучающей выборки.

3 Вычислительные эксперименты

3.1 Построение композиций модифицированных решающих деревьев

3.1.1 Постановка задачи

Данные для проведения экспериментов в этом разделе взяты из международного соревнования Higgs Boson Machine Learning Challenge [1], проходившего с 12 мая по 15 сентября 2014 года.

Ключевым свойством любой физической частицы является то, как часто она распадается на другие частицы. Эксперимент ATLAS физики элементарных частиц проходит в Большом Адронном Коллайдере в ЦЕРНе и нацелен на расщепление частиц различной природы с использованием лобовых столкновений протонов чрезвычайно высокой энергии. Именно при помощи этого эксперимента был получен бозон Хиггса [2], который распадается на две τ-частицы, но этот распад происходит слишком редко, в следствие чего возникает большое количество «шумовых» столкновений.

Задача состоит в том, чтобы, исследуя признаковые описания событий, обнаруженных в ходе эксперимента ATLAS, классифицировать их на «τ-τ распад бозона Хиггса» и «шум».

3.1.2 Описание данных и функционала качества

Тренировочная выборка состоит из 250000 объектов, которые описываются 30 вещественными признаками и весом наблюдения. Тестовая выборка содержит 550000 объектов, также описываемых 30 признаками, но веса и метки для них неизвестны.

Функционал качества в рассматриваемой задаче был нестандартный:

$$\text{AMS} = \sqrt{2\Big((\text{tp} + \text{fp} + \text{fp}_r)\ln\big(1 + \frac{\text{tp}}{\text{fp} + \text{fp}_r}\big)\Big)} \to \max,$$

где tp и fp — значения *true-positive rate* и *false-positive rate*, $\text{fp}_r = 10$ — константа регуляризации. Если посмотреть более детально, то, обозначив верный вектор ответов $(y_1, \ldots, y_n) \in \{b, s\}^n$, спрогнозированный $(\hat{y}_1, \ldots, \hat{y}_n) \in \{b, s\}^n$, а вектор весов $(w_1, \ldots, w_n) \in \mathbb{R}^{+n}$, получим следующие формулы:

$$\text{tp} = \sum_{i=1}^{n} w_i[y_i = s][\hat{y}_i = s]$$

$$\text{fp} = \sum_{i=1}^{n} w_i[y_i = b][\hat{y}_i = s]$$

3.1.3 Зависимость качества от структурных параметров алгоритма

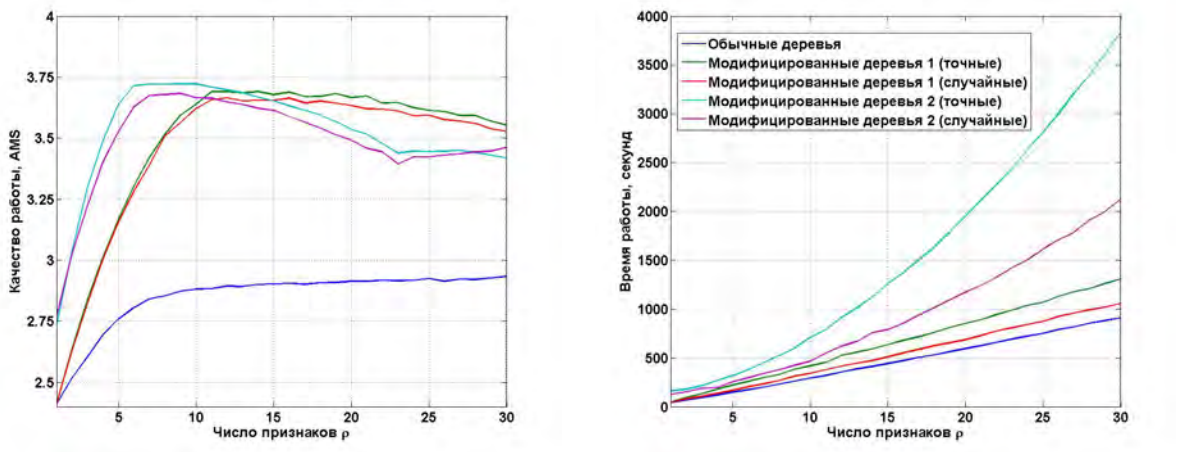
Проведем исследование качества и общего времени построения алгоритма, настроенного на тренировочной выборке, в зависимости от структурных параметров:

- числа признаков ρ, используемых при построении каждого из узлов дерева;
- вида разделяющего критерия в каждом из узлов дерева;
- порога на число объектов, попадающих в лист дерева, для построения регрессионной зависимости, которая корректирует конечные ответы.

Определение 3. *Модифицированное дерево решений с линейными (второго порядка) разделяющими критериями в узлах будем называть* ***точным****, если при построении каждого из внутренних узлов дерева решалась задача по подбору коэффициентов композиции первого (второго) порядка выбранных случайным образом ρ признаков*

Определение 4. *Если выбор разделяющего критерия производился из некоторого случайного множества вариантов, то такие деревья будем называть модифицированными* ***случайными*** *решающими деревьями*

Графики качества и общего времени построения композиции из 100 деревьев каждого типа в зависимости от параметра ρ приведены на рисунках 11a и 11b, а лучшие результаты для каждого из методов построения отражены в таблице 1, где решающие деревья с линейным разделяющим критерием обозначены как «Модифицированные деревья 1», а деревья с разделяющим критерием второго порядка — как «Модифицированные деревья 2».



(a) Качество AMS **(b)** Время настройки

Рис. 11: Результаты эксперимента для различных значений параметра ρ и вида разделяющего критерия

Из полученных в эксперименте результатов, можно сделать следующие выводы:

- при увеличении значения параметра ρ модифицированные деревья решений довольно быстро наращивают свое преимущество над стандартным алгоритмом построения, однако, в отличие от обычных деревьев, имеют склонность к потере качества при больших значениях ρ из-за мультиколлинеарности признаков исходной задачи;
- в случае использования линейных разделяющих критериев необходимо выбирать значение параметра $\rho = 12$ из множества возможных значений $\{1, \ldots, 30\}$,

Базовый алгоритм	**Качество работы (AMS)**	**Время настройки (секунд)**
Дерево решений ($\rho = 23$)	2.91	647.3
Точное модифицированное дерево 1($\rho = 12$)	3.69	457.1
Случайное модифицированное дерево 1 ($\rho = 12$)	3.66	395.2
Точное модифицированное дерево 2 ($\rho = 7$)	3.72	495.7
Случайное модифицированное дерево 2 ($\rho = 8$)	3.68	386.9

Таблица 1: Результаты эксперимента для различных значений параметра ρ и вида разделяющего критерия

поскольку именно при этом значении достигается максимальное качество работы, что соответствует наблюдениям на модельных данных;

- значение параметра $\rho = 7$ позволяет получить наилучшее качество работы при использовании точно настроенных разделяющих критериев второго порядка, однако алгоритмы такого рода гораздо больше склонны к переобучению из-за огромного числа параметров, что подтверждается быстрым падением качества при увеличении параметра ρ;

- случайные модифицированные деревья наиболее пригодны для практического использования, поскольку они получают приемлемое качество, сравнимое с точными деревьями, за значительно меньшее время.

При проведении эксперимента, связанного с изменением порога на количество объектов в листьях с последующим решением задачи регрессии для каждой листовой вершины, были получены результаты, показанные в таблице 2, которые полностью подтверждают наблюдения на модельных данных. Применение такого рода модификации алгоритма приводит к серьезному увеличению времени работы при незначительном улучшении качества в терминах оптимизируемого функционала.

Базовый алгоритм	**Качество работы (AMS)**	**Время настройки (секунд)**
Случайное модифицированное дерево 1 ($\rho = 12$)	3.66	395.2
Случайное модифицированное дерево 1 с порогом	3.67	612.3
Случайное модифицированное дерево 2 ($\rho = 8$)	3.68	386.9
Случайное модифицированное дерево 2 с порогом	3.69	593.1

Таблица 2: Результаты эксперимента с порогом на количество объектов в листьях дерева

Следующий эксперимент, основанный на данных рассматриваемой задачи, связан с построением композиции модифицированных деревьев решений для получения устойчивого итогового алгоритма. Помимо простого усреднения ответов, полученных каждым деревом, можно использовать следующие варианты комбинирования:

- вместо среднего арифметического ответов деревьев вычислять среднее геометрическое:

  $$\hat{Y}_i^{answer} = \sqrt[T]{\prod_{t=1}^{T} b_i^t(x)}$$

  При использовании такого варианта построения композиции может получиться итоговый ответ с $\sum\limits_{i=1}^{N_{class}} \hat{Y}_i^{answer} < 1$, поэтому необходимо производить финальную нормировку:

  $$Y_i^{answer} = \frac{\hat{Y}_i^{answer}}{\sum\limits_{s=1}^{N_{class}} \hat{Y}_s^{answer}};$$

- веса деревьям присваиваются согласно среднему качеству разбиений данных — чем ближе среднее значение отношения числа объектов, пошедших в левое и правое поддерево относительно некоторого внутреннего узла, к значению 0.5, тем лучше:

  $$\hat{w}_t = \frac{1}{|\text{MeanSplitQuality}(T_t) - 0.5| + \epsilon},$$

$$w_t = \frac{\hat{w}_t}{\sum_{s=1}^{T} \hat{w}_s},$$

где MeanSplitQuality(T_t) — среднее качество разбиения для дерева T_t, а $\epsilon = 10^{-6}$ для исключения возможности деления на ноль. Стоит отметить, что этот способ задания весов также учитывает высоту дерева, поскольку чем ближе MeanSplitQuality(T_t) к значению 0.5, тем более сбалансированным и низким получится итоговое дерево.

Результаты проведенного эксперимента с комбинированием случайных модифицированных деревьев решений различной глубины с разделяющими критериями второго порядка отображены на графике 12 и в таблице 3, исходя из которых можно сделать вывод о том, что среднее геометрическое ответов деревьев с последующей нормировкой позволяет получить наилучшее качество работы при незначительном дополнительно затраченном времени.

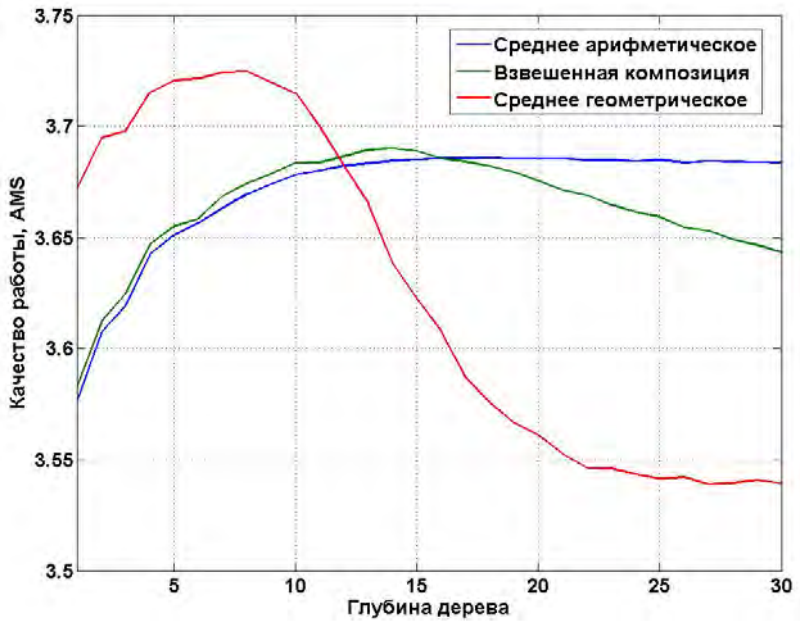


Рис. 12: Результаты эксперимента для различных алгоритмов комбинирования случайных модифицированных деревьев

Также на основе приведенных результатов можно сделать следующие выводы:

- при увеличении максимальной глубины дерева среднее геометрическое ответов деревьев все больше переобучается и одновременно растет время его настрой-

Базовый алгоритм	**Качество работы (AMS)**	**Время настройки (секунд)**
Среднее арифметическое	3.68	386.9
Взвешенная композиция	3.69	415.3
Среднее геометрическое	3.72	390.1

Таблица 3: Результаты эксперимента для различных алгоритмов комбинирования случайных модифицированных деревьев

ки, что позволяет ограничиться лишь небольшой высотой деревьев в итоговой композиции;

- при работе со слишком низкими деревьями наблюдается эффект недообучения (полученная модель не обеспечивает получение достаточно малой величины ошибки на тренировочной выборке), который негативно сказывается на качестве работы итоговой модели — низкие деревья обладают недостаточной обобщающей способностью и часто неверно классифицируют тестовый объект, что приводит к получению неправильного ответа всей композицией.

Поясним последнее утверждение на примере: рассмотрим задачу классификации на два класса при помощи среднего геометрического десяти случайных модифицированных деревьев. Пусть тестовый объект x_i принадлежит к классу $y_i = 1$ и все деревья $T_k (k \in \{1, \ldots, 10\})$, кроме T_j, выдали в качестве ответа верное значение вероятности $p_k = \mathbb{P}(y_i = 1 | x_i, T_k) = 1, k \neq j$. Тогда при $p_j = \mathbb{P}(y_i = 1 | x_i, T_j) = 0$ композиция выдаст значение $Y_{answer} = \sqrt[10]{1^9 \cdot 0} = 0 \neq 1 = y_i$. Таким образом, итоговый ответ композиции существенно зависит от всех ответов базовых алгоритмов, а значит необходимым условием качественной работы такого ансамбля является высокая обобщающая способность входящих в него алгоритмов.

В заключение этого раздела проанализируем качество работы различных композиций случайных деревьев при наличии пропущенных значений в исходных данных. Постановка этого эксперимента выглядит следующим образом:

- выберем наилучшие конфигурации стандартного и модифицированного случайных лесов для случая данных без пропусков;

- случайным образом заменим фиксированный процент значений на пропуски и запустим следующие вариации алгоритмов для настройки параметров:
    - стандартный случайный лес для выборки, в которой пропуски заполнялись медианным значением по столбцу;
    - модифицированный случайный лес с комбинированием деревьев на основе среднего арифметического без заполнения пропусков;
    - модифицированный случайный лес с комбинированием деревьев на основе среднего геометрического без заполнения пропусков.
- получим предсказания для тестовой выборки при помощи каждого из настроенных алгоритмов и вычислим качество их работы;
- повторим всю процедуру 10 раз, после чего усредним результаты для каждого алгоритма и рассматриваемого числа пропусков в данных для обучения

При построении модифицированных случайных лесов заполнение пропусков не использовалось, поскольку это могло внести в исходные данные информацию, которой в них не было изначально. Этот факт мог бы негативно сказаться на обобщающей способности алгоритма на стадии тестирования.

Введем дополнительные обозначения:

- NA — значение признака, соответствующее пропуску;
- $A \circ B$ — адамарово (поэлементное) произведение матриц;
- $[x = \text{NA}]$ — бинарный вектор с единицами на позициях пропусков в векторе x.

Следует отметить, что для работы с пропусками были проведены следующие модификации разделяющих критериев во внутренних узлах дерева:

- линейный (ориентированный) критерий:

$$C_2^{new}(x, w_1, w_2, \tau) = [((x \circ [x \neq \text{NA}])w_1 + [x = \text{NA}]w_2) \leqslant \tau];$$

- нелинейный критерий второго порядка:

$$C_3^{new}(x, w_1, w_2, W, \tau) = [((x \circ [x \neq \text{NA}])W(x \circ [x \neq \text{NA}])^T +$$

$$+(x \circ [x \neq \text{NA}])w_1 + [x = \text{NA}]w_2) \leqslant \tau].$$

Таким образом, для данных без пропусков разделяющие критерии C_2^{new} и C_3^{new} полностью аналогичны критериям C_2 и C_3, а в случае наличия пропусков происходит разделение каждого объекта на известную и неизвестную части, каждой из которых соответствует свой вектор (для C_3^{new} еще и матрица для известной части) весов. Результаты проведенного эксперимента изображены на рисунке 13 и в таблице 4.

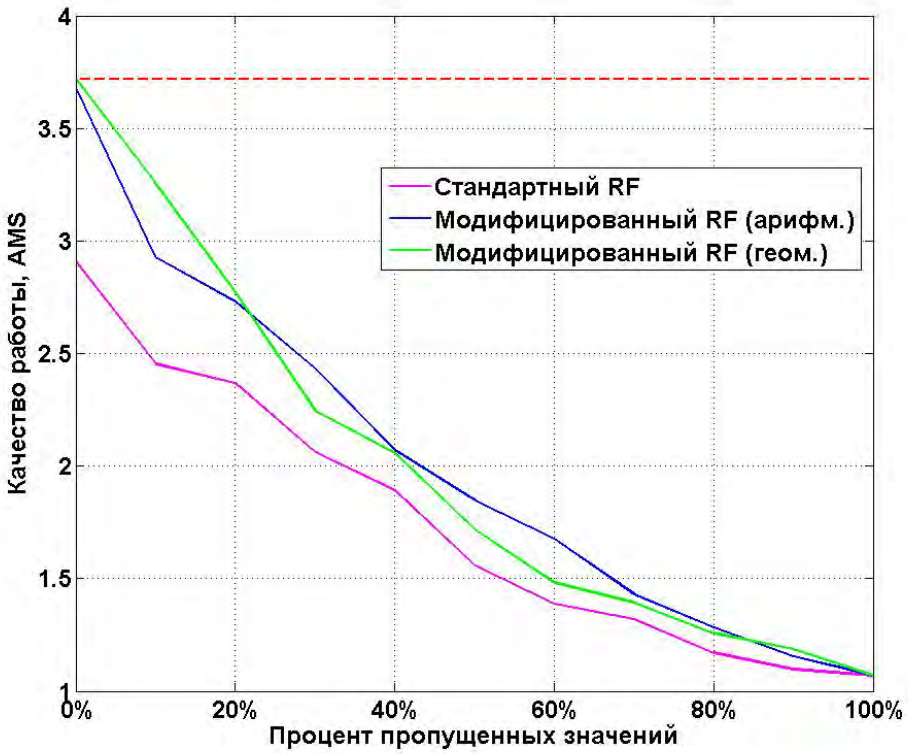


Рис. 13: Качество работы различных случайных лесов при наличии пропусков в данных

Из полученных в эксперименте результатов можно сделать следующие выводы:

- для построения конкурентоспособных композиций необходимо использовать все исходные данные, поскольку при добавлении даже 10% пропусков происходит серьезное ухудшение качества работы для всех видов композиций;
- оба варианта модифицированных случайных лесов показывают лучшее качество по сравнению со стандартным вариантом при любом проценте пропусков в исходных данных;

Алгоритм / Процент	Стандартный RF	Модифицированный RF (ср. арифм.)	Модифицированный RF (ср. геом.)
0%	2.9132	3.6815	3.7223
10%	2.4534	2.9298	3.2538
20%	2.3678	2.7290	2.7724
30%	2.0633	2.4337	2.2467
40%	1.8931	2.0716	2.0581
50%	1.5577	1.8481	1.7186
60%	1.3885	1.6758	1.4827
70%	1.3200	1.4314	1.3954
80%	1.1699	1.2817	1.2581
90%	1.0977	1.1555	1.1872
100%	1.0791	1.0791	1.0791

Таблица 4: Качество работы различных случайных лесов при наличии пропусков в данных

- построение композиции, основанной на среднем геометрическом ответов модифицированных деревьев решений, оправдано при наличии не более чем 25% пропусков. При увеличении доли пропусков следует использовать среднее арифметическое ответов деревьев, что позволяет избежать проблем с эффектом недообучения из-за нехватки данных.

Стоит также отметить, что категориальные признаки можно обрабатывать методом подобным тому, что использовался для пропусков. Пусть в исходных данных есть единственный категориальный признак с индексом i, принимающий значения из множества $\{a, b, c\}$, тогда для его учета в разделяющий критерий внутренней вершины дерева нужно добавить одно дополнительное произведение векторов:

- линейный (ориентированный) критерий:

$$C_2^{new}(x, w_1, w_2, w_3, \tau) = [((\hat{x} \circ [\hat{x} \neq \text{NA}])w_1 + [\hat{x} = \text{NA}]w_2 + \begin{pmatrix} [x_i = a] \\ [x_i = b] \\ [x_i = c] \end{pmatrix}^T w_3) \leqslant \tau];$$

- нелинейный критерий второго порядка:

$$C_3^{new}(x, w_1, w_2, w_3, W, \tau) = [((\hat{x} \circ [\hat{x} \neq \text{NA}])W(\hat{x} \circ [\hat{x} \neq \text{NA}])^T +$$

$$+(\hat{x} \circ [\hat{x} \neq \text{NA}])w_1 + [\hat{x} = \text{NA}]w_2 + \begin{pmatrix} [x_i = a] \\ [x_i = b] \\ [x_i = c] \end{pmatrix}^T w_3) \leqslant \tau].$$

В приведенных формулах $\hat{x} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_K)$, т.е. $\hat{x}$ является вещественным вектором на единицу меньшей длины по сравнению с вектором x. Очевидно, что в добавленном произведении векторов будет всего одно отличное от нуля слагаемое, поскольку x_i не может одновременно равняться сразу нескольким значениям из множества $\{a, b, c\}$. Таким образом, не возникает никаких зависимостей между уровнями категориального признака, которые появлялись при их кодировании целыми числами.

3.1.4 Анализ полученных ответов алгоритма

Большинство композиций алгоритмов машинного обучения являются для конечного пользователя своеобразными «черными ящиками», поскольку нельзя легко определить, каким образом алгоритм получил именно такой ответ для некоторого конкретного объекта. Однако все обстоит по-другому с деревьями решений — для каждого дерева можно определить лист, в который попал рассматриваемый объект и, рассматривая путь от корня до полученного листа, получить конъюнкцию логических условий, которую обращают в истину значения признаков этого объекта.

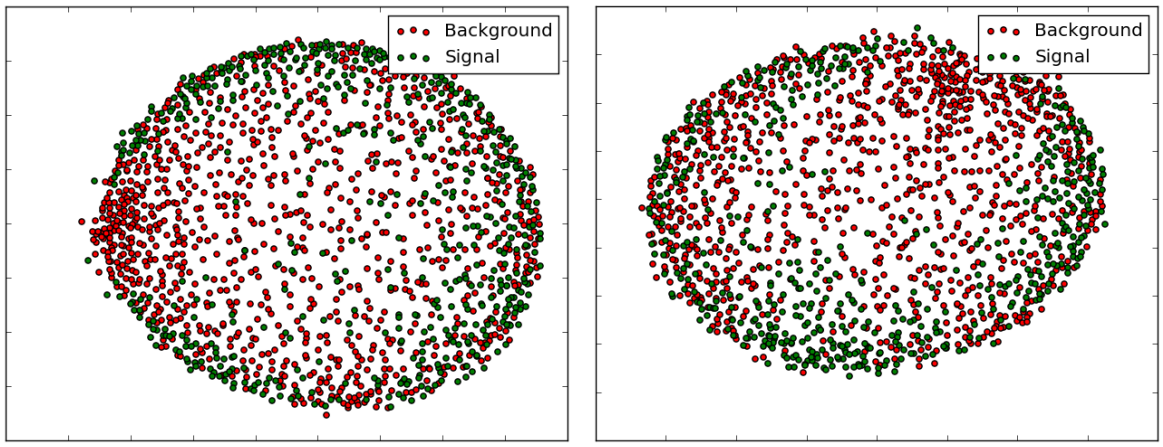
Более того, можно наглядно показать, что объекты, попадающие в один и тот же лист дерева, схожи друг с другом. Определим для оценки схожести объектов x_1 и x_2 следующую формулу:

$$Sim(x_1, x_2) = \frac{1}{M} \sum_{m=1}^{M} \sum_{l=1}^{L_m} \left([x_1 \in T_l^m] \cdot [x_2 \in T_l^m] \right)$$

где $T^1, \ldots, T^M$ — построенные в результате работы алгоритма M деревьев решений, T_l^m — лист с индексом l дерева T^m. Тогда для выборки $\{(x_i, y_i)\}_{i=1}^N$ можно определить матрицу похожести размера $N \times N$:

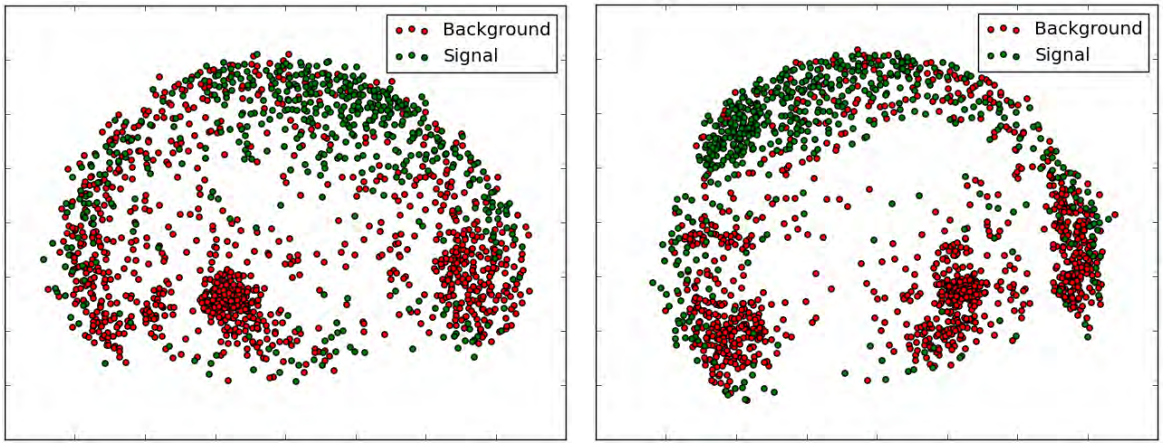
$$S_{ij} = Sim(x_i, x_j) = \frac{1}{M} \sum_{m=1}^{M} \sum_{l=1}^{L_m} \left([x_i \in T_l^m] \cdot [x_j \in T_l^m] \right),$$

где L_m — общее число листьев в дереве T^m. Заметим, что $S_{ij} \geqslant 0, \forall (i,j) : i,j \in \{1,\ldots,N\}$ и $S_{ij} = 1$ тогда и только тогда, когда объекты x_i и x_j неразличимы с точки зрения случайного леса. Используя такое определение матрицы похожести S, можно рассмотреть задачу многомерного шкалирования [5] с матрицей расстояний между объектами $D = 1 - S$, результаты решения которой для стандартного и модифицированного алгоритмов комбинирования случайных деревьев при различных значениях параметра ρ можно увидеть на рисунках 14–16.



(a) Стандартное дерево решений **(b)** Модифицированное дерево решений

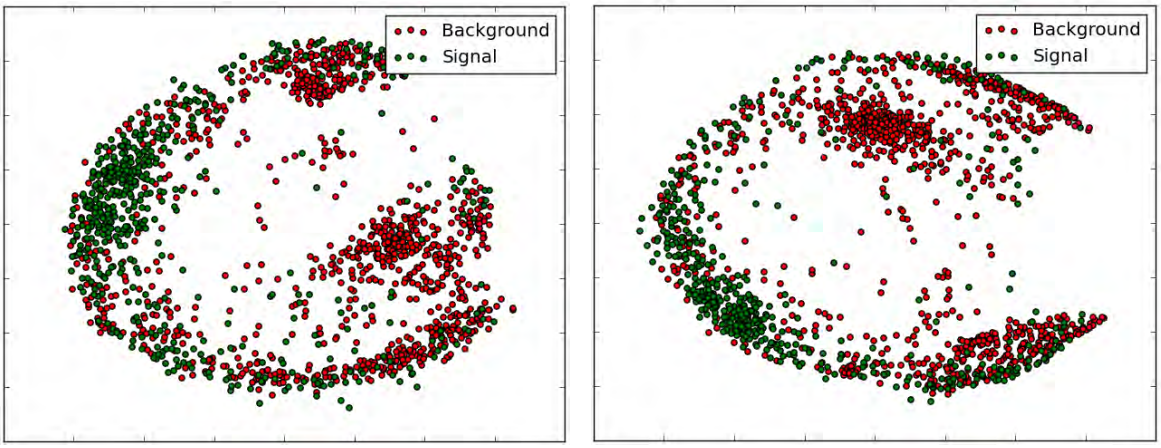
Рис. 14: Результаты эксперимента для низких значений параметра ρ



(a) Стандартное дерево решений **(b)** Модифицированное дерево решений

Рис. 15: Результаты эксперимента для средних значений параметра ρ

Из полученных результатов можно сделать следующие выводы:



(a) Стандартное дерево решений **(b)** Модифицированное дерево решений

Рис. 16: Результаты эксперимента для значений параметра ρ, близких к общему числу признаков

- для низких значений параметра ρ с точки зрения алгоритмов усреднения различных решающих деревьев (с использованием композиций признаков во внутренних узлах деревьев и без) объекты отложенной выборки, участвующие в эксперименте, имеют примерно одинаковую похожесть, так как на рисунках 14a и 14b значимых отличий не выявлено;

- для средних значений параметра ρ, результаты эксперимента для которых изображены на рисунках 15a и 15b, уже проявляется превосходство алгоритма, основанного на усовершенствованных деревьях решений, поскольку на рисунке 15b более четко видна кластерная структура пространства ответов алгоритма по сравнению с результатом на рисунке 15a, а значит и качество, и обоснованность ответов такого алгоритма будет выше;

- для значений параметра ρ, близких к общему числу признаков рассматриваемой задачи, тенденция улучшения кластерной структуры объектов отложенной выборки сохраняется, так как на рисунке 16b явно видны три кластера «шумовых» объектов и два кластера объектов типа «сигнал», в то время как на рисунке 16a эти кластеры серьезно размыты. Этот факт позволяет говорить о общем превосходстве усовершенствованных деревьев решений над стандартным вариантом.

3.1.5 Выводы

Подытожим полученные в ходе проведенных экспериментов результаты:

- при увеличении сложности разделяющих критериев, используемых во внутренних узлах деревьев решений, повышается качество и обобщающая способность полученного алгоритма, но и одновременно серьезно увеличивается общее время работы алгоритма, поэтому использование деревьев решений с разделяющими поверхностями сложнее, чем линейная комбинация некоторого набора признаков, оправдано только в случае однократного построения такого рода классификатора;

- количество признаков ρ, используемое для построения каждого разделяющего критерия, необходимо брать равным 20–40% от общего числа признаков, что позволит сохранить минимальную коррелированность полученных в результате построения деревьев решений, повысить качество и обобщающую способность за счет улучшения кластерной структуры пространства объектов с точки зрения итогового алгоритма, но в то же время незначительно ухудшить общее время работы по сравнению со случаем $\rho = 1$;

- использование порога на количество элементов в листьях деревьев, а также запуск регрессионных алгоритмов на них не дают значительного прироста качества, поэтому их использование для решения практических задач нецелесообразно;

- в качестве алгоритма построения композиции необходимо использовать среднее геометрическое ответов с последующей нормировкой полученных ответов, однако применять такой способ обоснованно только в случае деревьев средней высоты для избежания эффектов недо- и переобучения.

3.2 Построение комитетов модифицированных случайных лесов

3.2.1 Постановка задачи

Данные для проведения экспериментов в этом разделе взяты из международного соревнования Avazu Click-Through Rate (CTR) Prediction [3], проходившего с 18 ноября 2014 года по 9 февраля 2015 года.

В онлайн рекламе уровень кликабельности баннера — очень важный показатель его доходности, поэтому алгоритмы его прогнозирования стали неотъемлемой частью специализированных систем рекламных компаний. Чем ближе к действительности будут предсказанные значения, тем более выгодное расписание показа баннеров можно создать, а значит тем больше будет прибыль компании за некоторый фиксированный промежуток времени, поскольку именно интернет является крупнейшей международной рекламной площадкой.

Основной задачей соревнования являлось прогнозирование уровня кликабельности баннера (значения $CTR \in [0; 1]$ — будет ли осуществлен клик по баннеру или нет) на основе информации о самом баннере, сайте, на котором он размещен, времени показа и устройства, с которого его будет просматривать пользователь.

3.2.2 Описание данных и функционала качества

Обучающая выборка состояла из 40428968 объектов-баннеров, собранных всего за 10 дней показа (с 21.10.2014 по 30.10.2014). Каждый баннер описывался 23 признаками, 10 из которых являлись категориальными. Общий объем CSV-файла с тренировочной выборкой составлял почти 6Гб. В качестве тестовой выборки был взят следующий за тренировочной выборкой, 11-й день показа (31.10.2014), содержащий описания примерно 4.5 миллиона баннеров.

Оптимизируемый функционал в рассматриваемой задаче являлся стандартной мерой — логарифмической ошибкой (LogLoss):

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}) \to \min_{p_{ij}}$$

где N — общее число объектов в тестовой выборке, M — количество предсказываемых значений для каждого из объектов, y_{ij} — правильный ответ, а p_{ij} — предсказанное значение. Поскольку в задаче всего два класса («произошел клик» и «клика

не было»), то приведенную общую формулу можно упростить следующим образом:

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}\big(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\big) \to \min_{p_i},$$

где p_i — предсказанная вероятность клика по баннеру с индексом i.

3.2.3 Анализ зависимости качества алгоритма от гиперпараметров

Поскольку данные рассматриваемой задачи целиком не помещаются в оперативную память ПК, то использовать все данные одновременно для построения модели не представляется возможным. В таких случаях используется следующий прием, позволяющий уменьшить количество данных, обрабатываемых алгоритмом в единицу времени — производится сэмплирование случайных подвыборок объектов со случайным набором признаков. Введем необходимые в дальнейшем обозначения:

- N — общее число объектов в имеющихся данных;
- M — общее число признаков в имеющихся данных;
- D — матрица «объект–признак» размера $N \times M$;
- $\alpha \in [0; 1]$ — доля выбранных в подвыборку объектов от общего числа объектов;
- $\beta \in [0; 1]$ — доля выбранных для подвыборки признаков от общего числа признаков;
- Mem_size — размер оперативной памяти ПК, на котором решается задача;
- $V(\alpha, \beta, D)$ — множество всех случайных подвыборок размера αN, в которой каждый объект описан βM признаками;
- $R(\alpha, \beta, D) \in V(\alpha, \beta, D)$ — некоторая случайная подвыборка размера αN, в которой каждый объект описан βM признаками.

В приведенных выше обозначениях и везде в дальнейшем значения αN и βM округляются до ближайшего целого. Основываясь на введенных обозначениях, можно описать алгоритм построения модели в виде псевдокода 1.

Из введенных выше обозначений также следует, что размер подвыборки, семплированной из полной выборки, можно обозначить как $\alpha N \cdot \beta M$, и, в силу очевидных

Алгоритм 1: Построение модели в случае, когда объем данных превышает объем памяти ПК

Вход: α, β, D, T;

Выход: $Model$;

1: **для** $t = 1, \ldots, T$

2: Выбрать случайную $R(\alpha, \beta, D)$ из множества $V(\alpha, \beta, D)$;

3: Обучить модель m_i для $R(\alpha, \beta, D)$;

4: Агрегировать модели m_i в итоговую модель $Model$;

ограничений размера памяти, получаем, что $\alpha N \cdot \beta M \leqslant Mem\_size$. Таким образом, долю единовременно используемых данных можно ограничить при помощи следующего неравенства:

$$Batch\_size = \alpha\beta \leqslant \frac{Mem\_size}{NM}$$

В данном разделе будем исследовать зависимость качества работы и времени настройки модели в зависимости от размера подвыборок, сгенерированных для обучения, а также чувствительность получаемого решения задачи к изменениям параметров α и β.

На первый взгляд кажется, что для обучения каждой модели необходимо использовать максимальное число данных, которые помещаются в память, т.е. решать задачу

$$\begin{cases} \alpha\beta \to \max\limits_{\alpha\in[0;1],\beta\in[0;1]} \\ \alpha\beta \leqslant \dfrac{Mem\_size}{NM}, \end{cases}$$

однако такой метод не всегда оказывается успешным по нескольким причинам:

- для построения модели на основе такой подвыборки затрачивается максимально возможное время по сравнению с другими вариантами;
- среди признаков нередко встречаются группы сильно коррелированных признаков, что также затрудняет решение задачи и снижает качество полученного результата;
- в случае, когда значение $Batch\_size$ близко к 1, генерируются подвыборки с сильным перекрытием, что повышает близость получаемых базовых моделей и,

как следствие, уменьшает преимущество композиции перед отдельно взятым алгоритмом.

Поскольку в задаче уже имеется логическое деление всей выборки на 10 примерно равных частей, каждая из которых соответствует одному дню наблюдения, то эксперименты по анализу качества и времени настройки всей композиции можно разделить на два этапа по типу генерации подвыборок:

- генерация подвыборок из всей совокупности данных;
- генерация подвыборок из данных, собранных в течение каждого отдельного дня.

Сравнение композиций модифицированных решающих деревьев будем производить на основе каждого из двух критериев в отдельности:

- качеству работы итоговой композиции на тестовой выборке;
- объему данных, затрачиваемому для построения композиции.

В первом эксперименте проанализируем качество работы комитета композиций модифицированных решающих деревьев для случая генерации случайных подвыборок для каждой пары параметров (α, β). Постановка и процесс выполнения эксперимента выглядят следующим образом:

- $\alpha \in A = \{0, 0.125, \ldots, 0.875, 1\}$;
- $\beta \in B = \{0, 0.125, \ldots, 0.875, 1\}$;
- для каждой пары $(\alpha, \beta) \in A \times B$ сгенерируем по 5 выборок, состоящих из αN объектов, описываемых βM признаками. Здесь $N = 40422968$, т.е. равно числу объектов в исходной обучающей выборке;
- на каждой выборке настроим параметры композиции модифицированных случайных деревьев и получим предсказания для тестовой выборки, равной по объему данным, собранным за один день (около 4 миллионов объектов-баннеров);

- усредним предсказания, полученные моделями на тестовой выборке для каждой из пар (α, β).

Результаты, полученные в этом эксперименте, можно видеть на рисунке 17 и в таблице 5. Из них можно сделать вывод, что для обучения стоит использовать сравнительно небольшие объемы данных, поскольку именно в этом случае полученный $LogLoss$ существенно ниже по сравнению с граничными результатами. Этот факт подтверждает теоретические предположения о наличии мультиколлинеарности в данных и возникновении эффектов недо- и переобучения в случае использования недостаточного или избыточного количества признаков, которыми описываются объекты-баннеры в тренировочной выборке.

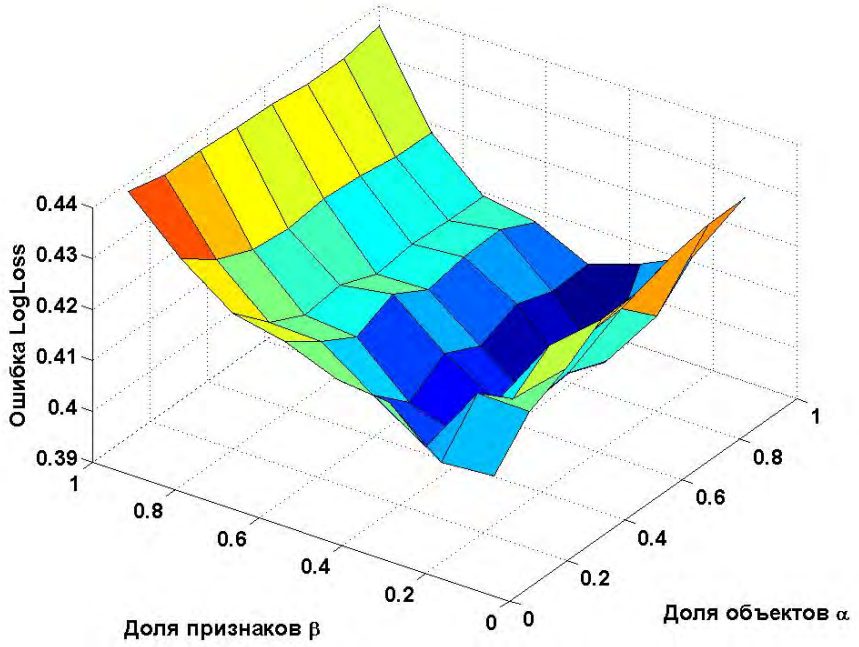


Рис. 17: Результаты эксперимента при генерации случайных подвыборок

Следующий эксперимент будет связан с выборками, сгенерированными из данных, собранных в течение фиксированных дней. Сопоставим исходные данные календарным дням недели:

- 21 октября — вторник;
- 22 октября — среда;

α \ β	**0.1250**	**0.2500**	**0.3750**	**0.5000**	**0.6250**	**0.7500**	**0.8750**	**1.0000**
0.1250	0.4061	0.4054	0.4138	0.4150	0.4189	0.4211	0.4284	0.4383
0.2500	0.4138	0.4140	0.4001	0.4070	0.4140	0.4144	0.4233	0.4365
0.3750	0.4174	0.4080	0.3972	0.4005	0.4100	0.4109	0.4202	0.4363
0.5000	0.4220	0.4136	0.3991	0.4052	0.4134	0.4117	0.4194	0.4358
0.6250	0.4225	0.4102	0.3940	0.4016	0.4093	0.4089	0.4203	0.4356
0.7500	0.4249	0.4109	0.3948	0.4049	0.4107	0.4099	0.4199	0.4344
0.8750	0.4258	0.4044	0.3919	0.4024	0.4111	0.4096	0.4185	0.4345
1.0000	0.4261	0.4111	0.4063	0.4027	0.4076	0.4094	0.4184	0.4360

Таблица 5: Результаты эксперимента при генерации случайных подвыборок

- . . . ;
- 29 октября — среда;
- 30 октября — четверг;
- 31 октября — пятница.

Таким образом, можно провести эксперимент со следующей постановкой:

- $\alpha \in A = \{0, 0.125, \ldots, 0.875, 1\}$;
- $\beta \in B = \{0, 0.125, \ldots, 0.875, 1\}$;
- для каждой пары $(\alpha, \beta) \in A \times B$ сгенерируем по 5 выборок, состоящих из αN_{21} объектов, описываемых βM признаками, причем все объекты взяты только из набора данных, соответствующих 21 октября (вторник). Стоит отметить, что $N_{21} = 4122995$ — количество описаний баннеров-объектов, показанных пользователям 21 октября, которое составляет около 10% от общего числа объектов в исходной обучающей выборке;
- на каждой выборке настроим параметры композиции модифицированных случайных деревьев и получим предсказания для тестовой выборки, данные для которой взяты из набора за 22 октября (среда);

- усредним предсказания, полученные моделями на тестовой выборке для каждой из пар (α, β).

Эта постановка интересна тем, что подобранные в ходе эксперимента параметры композиций можно использовать при работе с любой парой дней «вторник–среда». Для решения поставленной задачи необходимо проделать аналогичную процедуру с 23 и 24 октября (четверг и пятница соответственно), после чего, дообучив композицию на данных, собранных за 30 октября, получить итоговый прогноз на тестовый день 31 октября.

Качество работы построенных композиций в ходе этого эксперимента можно видеть на рисунке 18 и в таблице 6. Из полученных результатов можно сделать вывод о том, что, как и в предыдущем эксперименте, полученные композиции модифицированных случайных лесов показывают наилучшее качество при использовании около 40% признаков исходной задачи. Однако, по сравнению с предыдущим экспериментом, в обучении моделей здесь в качестве исходной выборки для генерации использовалось всего 10% от общего числа объектов, что дополнительно положительно сказывается на времени настройки, позволяя тем самым обучать больше базовых моделей за фиксированное время. Следует также отметить, что полученные в этом эксперименте значения ошибки $LogLoss$ для различных пар (α, β) имеют более обширное и низкое плато, чем при случайной генерации выборок, что говорит о важности учета логических связей, имеющихся в данных.

В заключительном эксперименте совместим различные варианты настройки модели (по всем данным и по предыдущему дню) в итоговый алгоритм при помощи взвешенной линейной комбинации следующего вида:

$$Y_{answer} = \delta * Y_{all} + (1 - \delta) * Y_{previous}$$

Наилучшие достигнутые значения функционала $LogLoss$ для различных пар (α, β) показаны на рисунке 19 и в таблице 7, из которых можно сделать вывод о том, что для оптимального решения поставленной задачи необходимо использовать всего примерно 40% признаков и 90% объектов исходной выборки, т.е. всего **36%** от общего объема данных. Также не стоит забывать о том, что эти 36% потребуются полностью только для алгоритма, использующего случайные подвыборки, в то время как модель, основанная на прогнозировании значений следующего дня,

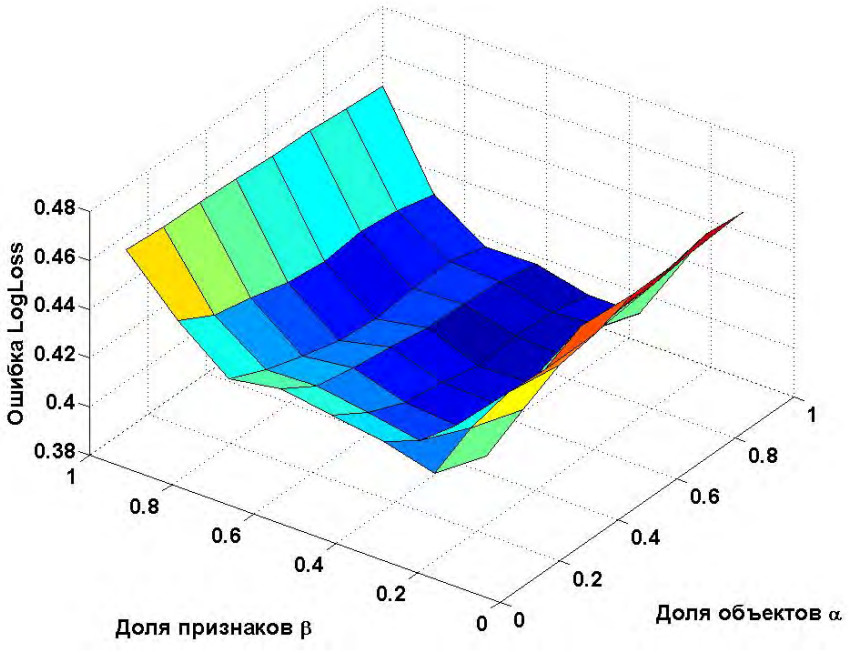


Рис. 18: Результаты эксперимента при генерации связанных во времени выборок

α \ β	**0.1250**	**0.2500**	**0.3750**	**0.5000**	**0.6250**	**0.7500**	**0.8750**	**1.0000**
0.1250	0.4217	0.4072	0.4128	0.4163	0.4193	0.4158	0.4323	0.4540
0.2500	0.4304	0.4155	0.4027	0.4067	0.4099	0.4092	0.4242	0.4527
0.3750	0.4383	0.4195	0.3983	0.4001	0.4071	0.4042	0.4206	0.4524
0.5000	0.4405	0.4182	0.3971	0.3979	0.4050	0.4001	0.4163	0.4515
0.6250	0.4462	0.4254	0.3950	0.3941	0.4024	0.3984	0.4145	0.4501
0.7500	0.4463	0.4214	0.3947	0.3958	0.4016	0.4002	0.4161	0.4500
0.8750	0.4495	0.4218	0.3970	0.3950	0.4002	0.4008	0.4143	0.4487
1.0000	0.4481	0.4241	0.3918	0.3917	0.3967	0.3964	0.4105	0.4474

Таблица 6: Результаты эксперимента при генерации связанных во времени выборок

будет работать лишь с 3.6% исходной выборки (так как за каждый день в выборке имеется примерно 10% объектов от общего числа объектов N).

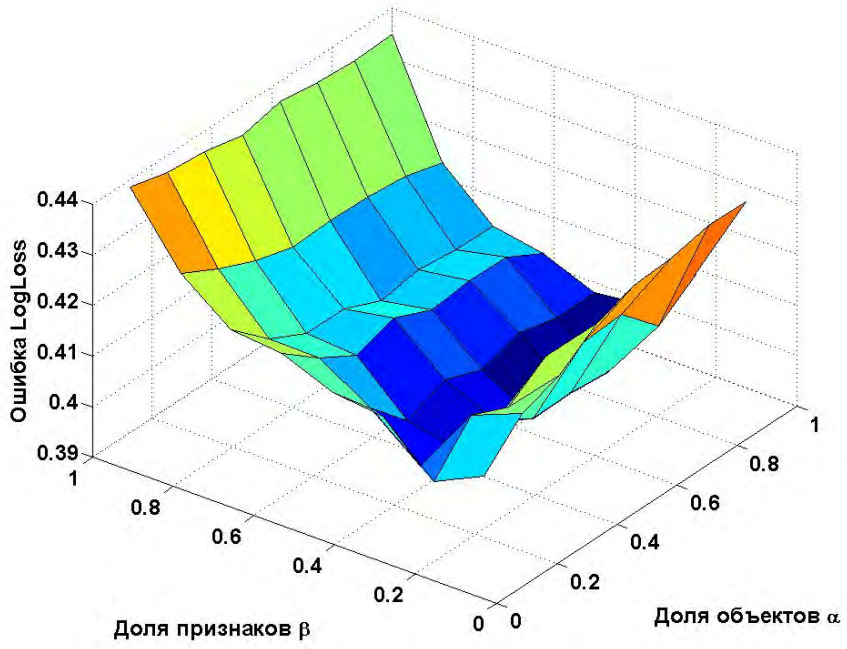


Рис. 19: Результаты эксперимента комбинирования алгоритмов для различных вариантов генерации обучающей выборки

3.2.4 Методы уменьшения объема используемой памяти

В следующем разделе рассмотрим пары значений $(\bar{\alpha}, \bar{\beta})$, при которых достигается минимальная ошибка *LogLoss* при различных вариантах генерации тренировочной выборки. Постановку этого эксперимента можно записать следующим образом:

- 9 раз произведем генерацию случайных выборок из исходной выборки и сделаем по 10 запусков построения композиции решающих деревьев для всевозможных пар значений (α, β);
- для каждой из девяти задач прогнозирования CTR на следующий день при известных значениях текущего дня сделаем по 10 запусков построения композиции решающих деревьев для всевозможных пар значений (α, β);
- полученные 18 множеств пар $\{(\bar{\alpha}_i, \bar{\beta}_i)\}_{i=1}^{10}$ (по 9 для случайных и связанных во времени выборок), доставляющие минимум функционалу *LogLoss*, изобразим

α \ β	**0.1250**	**0.2500**	**0.3750**	**0.5000**	**0.6250**	**0.7500**	**0.8750**	**1.0000**
0.1250	0.4066	0.4020	0.4112	0.4123	0.4164	0.4175	0.4250	0.4386
0.2500	0.4141	0.4102	0.3966	0.4045	0.4107	0.4110	0.4211	0.4366
0.3750	0.4168	0.4070	0.3950	0.3971	0.4064	0.4071	0.4178	0.4359
0.5000	0.4221	0.4121	0.3958	0.4004	0.4089	0.4071	0.4156	0.4342
0.6250	0.4248	0.4100	0.3914	0.3975	0.4057	0.4039	0.4161	0.4360
0.7500	0.4254	0.4106	0.3911	0.3999	0.4065	0.4050	0.4160	0.4349
0.8750	0.4270	0.4034	0.3908	0.3979	0.4064	0.4042	0.4151	0.4343
1.0000	0.4268	0.4088	0.3991	0.3980	0.4023	0.4041	0.4130	0.4346

Таблица 7: Результаты эксперимента комбинирования алгоритмов для различных вариантов генерации обучающей выборки

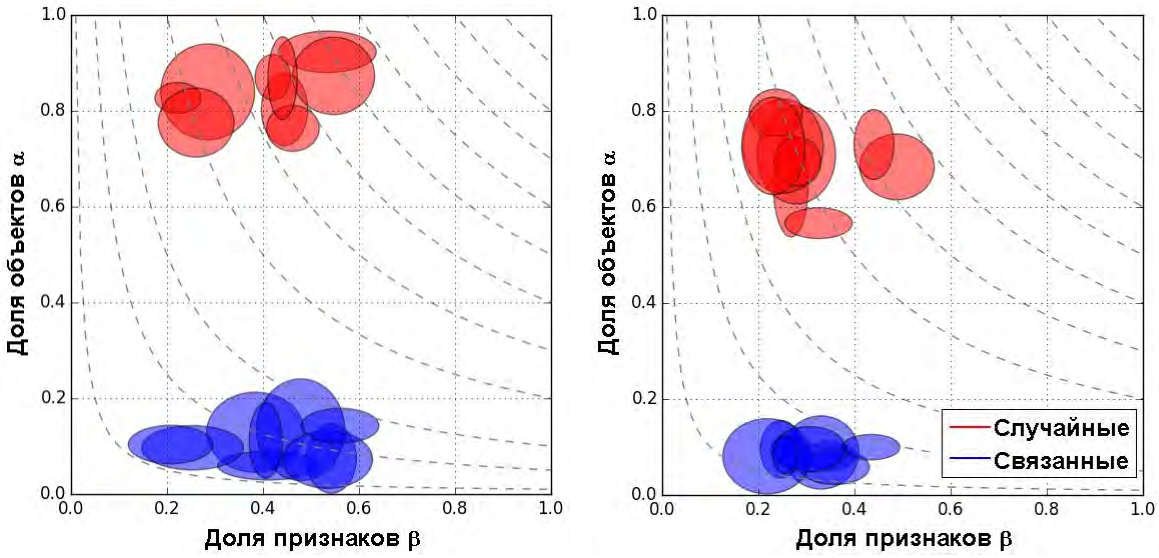
на плоскости $O\alpha\beta$ в виде эллипсов с центрами $(\hat{\alpha}, \hat{\beta})$ и параллельными осям $O\alpha$ и $O\beta$ полуосями, равными a и b соответственно, где:

$$\hat{\alpha} = \frac{\sum\limits_{i=1}^{10} \bar{\alpha}_i}{10}, \ \hat{\beta} = \frac{\sum\limits_{i=1}^{10} \bar{\beta}_i}{10}$$

$$a = \sqrt{\frac{\sum\limits_{i=1}^{10} (\bar{\alpha}_i - \hat{\alpha})^2}{10}}, \ b = \sqrt{\frac{\sum\limits_{i=1}^{10} (\bar{\beta}_i - \hat{\beta})^2}{10}}.$$

Полученную плоскость $O\alpha\beta$ с нанесенными на нее эллипсами можно видеть на рисунке 20a. Для наглядности на эту плоскость также нанесены линии уровня $\{0.01, 0.05, 0.1, 0.2, \ldots, 0.9\}$ функции $Mem\_size = \alpha \cdot \beta$, которые позволяют оценить количество памяти (а, соответственно, и время, требуемое на настройку итоговой композиции), которое приходится расходовать при решении каждой из задач для получения оптимального результата. По результатам проведенного эксперимента видно, что далеко не в каждой рассмотренной задаче лучшие результаты по функционалу качества $LogLoss$ получаются с использованием небольших объемов данных (в некоторых случаях при работе со случайными подвыборками требуется почти 60% исходных данных), поэтому в следующем эксперименте для каждой задачи найдем такую границу по объему памяти Mem_size', которая позволяет максимально сокра-

тить объем используемых данных без существенных потерь качества работы. Полученные результаты для каждой задачи приведены на рисунке 20b и в таблице 8. Используя их, можно сделать вывод, что в случае ослабления ограничения минимальности функционала $LogLoss$ на сравнительно небольшое $\epsilon = 3 \cdot 10^{-3}$ удается использовать примерно на 20% меньше данных для обучения модели со случайными подвыборками и более чем на 50% в случае связанных во времени обучающих выборок. Это позволяет построить гораздо больше моделей за фиксированное время, что может оказать довольно критичным параметром в задачах реального времени.



(a) До преобразования

(b) После преобразования

Рис. 20: Результаты эксперимента по уменьшению объема используемой памяти для настройки параметров модели

3.2.5 Выводы

Остановимся на наиболее важных результатах, полученных в ходе экспериментов этого раздела:

- в случае использования случайных подвыборок исходной обучающей выборки для построения композиции модифицированных деревьев решений необходимо производить генерацию обучающих множеств размером примерно 36% от общего объема данных — ошибка такой композиции по функционалу $LogLoss$ составит 0.3919;

Тип выборки / Критерий	До преобразования		После преобразования	
	Случайные	Связанные	Случайные	Связанные
Качество работы	0.3919	0.3917	0.3921	0.3917
Минимальный объем памяти	20.7%	2.1%	18.5%	2%
Средний объем памяти	39.4%	5.2%	21.4%	2.5%
Максимальный объем памяти	58.5%	11.2%	39.7%	5.2%
Среднее отклонение, α	0.0662	0.0531	0.0639	0.0467
Среднее отклонение, β	0.0689	0.0728	0.0644	0.0489

Таблица 8: Результаты эксперимента по уменьшению объема используемой памяти для настройки параметров модели

- в случае использования подвыборок, сгенерированных из данных за один день, для построения композиции модифицированных деревьев решений необходимо использовать выборки размером всего 5% от общего объема данных (или 50% от объема данных, собранных за этот день) — ошибка такой композиции по функционалу $LogLoss$ составит 0.3917;
- при использовании линейной комбинации двух композиций модифицированных решающих деревьев можно улучшить результат, получаемый при использовании каждой из моделей в отдельности, до 0.3908;
- при незначительном ослаблении ограничения минимальности ошибки $LogLoss$ на $\epsilon = 3 \cdot 10^{-3}$ можно использовать:
  - на 20% меньше данных в случае построения модели, основанной на случайных подвыборках;
  - на 50% меньше данных в случае построения модели прогнозирования значений на следующий день.

4 Заключение

В настоящей дипломной работе:

- предложены и исследованы модифицированные решающие деревья и методы их объединения в композиции, позволяющие получить устойчивое решение и существенно повысить итоговое качество работы;
- программно реализованы все необходимые алгоритмы и модуль для проведения экспериментов;
- проведены вычислительные эксперименты на двух реальных наборах данных: описаниях распада физических частиц в Большом Адронном Коллайдере (250000 объектов) и рекламных баннеров, размещенных в сети Интернет (более 40 млн. объектов).
- итоговый алгоритм показал приемлемый результат работы на обеих задачах по сравнению с существующими аналогами, чем доказал свою актуальность и практическую применимость.

Список литературы

[1] http://www.kaggle.com/c/higgs-boson.

[2] http://home.web.cern.ch/topics/higgs-boson.

[3] http://www.kaggle.com/c/avazu-ctr-prediction.

[4] *Biau G. et al.* A weighted k-nearest neighbor density estimate for geometric inference // *Electronic Journal of Statistics*. — 2011. — Vol. 5. — Pp. 204–237.

[5] *Borg I., Groenen P.* Modern Multidimensional Scaling. Theory and Applications (2 ed.). — Springer, 2005.

[6] *Breiman L.* Bagging predictors // *Machine Learning*. — 1996. — Vol. 24, no. 2. — Pp. 123–140.

[7] *Breiman L.* Random forests // *Machine Learning*. — 2001. — Vol. 45, no. 1. — Pp. 5–32.

[8] *Cortes C., Vapnik V.* Support-vector networks // *Machine learning*. — 1995. — Vol. 20, no. 3. — Pp. 273–297.

[9] *Dean J., Ghemawat S.* Mapreduce: simplified data processing on large clusters // *Communications of the ACM*. — 2008. — Vol. 51, no. 1. — Pp. 107–113.

[10] *Even S., Even G.* Graph Algorithms. — Cambridge University Press, 2011.

[11] *Freund Y., Schapire R. E.* A short introduction to boosting // In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. — Morgan Kaufmann, 1999. — Pp. 1401–1406.

[12] *Friedman J. H.* On bias, variance, 0/1—loss, and the curse-of-dimensionality // *Data mining and knowledge discovery*. — 1997. — Vol. 1, no. 1. — Pp. 55–77.

[13] *Ho T. K.* The random subspace method for constructing decision forests // *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. — 1998. — Vol. 20, no. 8. — Pp. 832–844.

[14] *Jordan M. I., Jacobs R. A.* Hierarchical mixtures of experts and the em algorithm // *Neural computation*. — 1994. — Vol. 6, no. 2. — Pp. 181–214.

[15] *Leiserson C. E., Schardl T. B.* A work-efficient parallel breadth-first search algorithm (or how to cope with the nondeterminism of reducers) // Proceedings of the Twenty-second Annual ACM Symposium on Parallelism in Algorithms and Architectures. — SPAA '10. — New York, NY, USA: ACM, 2010. — Pp. 303–314.

[16] *Orava J.* K-nearest neighbour kernel density estimation, the choice of optimal k // *Tatra Mountains Mathematical Publications*. — 2011. — Vol. 50, no. 1. — Pp. 39–50.

[17] *Panda B. et al.* Planet: Massively parallel learning of tree ensembles with mapreduce // *Proceedings of the VLDB Endowment*. — 2009. — Vol. 2, no. 2. — Pp. 1426–1437.

[18] *Parzen E.* On estimation of a probability density function and mode // *The annals of mathematical statistics*. — 1962. — Pp. 1065–1076.