Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный университет»

Кафедра Системного Программирования

Белоус Михаил Андреевич

Использование регрессии в алгоритмах машинного обучения

Дипломная работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Терехов А. Н.

Научный руководитель:
д. ф.-м. н., профессор Терехов А. Н.

Рецензент:
Начальник отдела мобильного поиска в компании Яндекс, к. ф.-м. н. Куралёнок И. Е.

Санкт-Петербург
2015

SAINT-PETERSBURG STATE UNIVERSITY

Chair of Software Engineering

Mikhail Belous

Applying regression for machine learning algorithms

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Andrey Terehov

Scientific supervisor:
professor Andrey Terehov

Reviewer:
Head of mobile search department in Yandex company, Dr. Igor Kuralenok

Saint-Petersburg
2015

Оглавление

Введение

С увеличением доступности и развитием сети Интернет остро встал вопрос информационного поиска. Для поиска информации в сети интернет большинство людей пользуются различным поисковым порталам, отправляя запросы об интересующей их теме. В ответ на свой запрос пользователь ожидает получить список веб-страниц, на которых содержится информация, отвечающая на запрос. Для того, чтобы быть "в курсе" происходящего в сети Интернет, поисковый портал регулярно составляет индекс интернет-страниц и в ответ на запрос пользователя находит в этом индексе подходящие документы. Однако, рост объёмов информации в сети привёл к тому, что даже для самых специфичных запросов в индексе находятся тысячи документов, которые пользователь физически не сможет просмотреть. Поэтому одним из важнейших аспектов использования поисковой системы является порядок, в котором пользователь увидит документы. Для того, чтобы компьютер мог принять решение об упорядочивании выдачи, по паре документ-запрос вычисляется множество характиристик, которые потом передаются ранжирующему алгоритму, упорядочивающему выдачу. Примерами характеристик являются: язык запрос, язык документа, цитируемость сайта, посещаемость сайта, количество вхождений слов запроса в документ.

Важным и неоднозначным вопросом является оценка качества работы ранжирующего алгоритма. В данной работе для этого использовалась методология Кранфилд [1]:

1. Для оценки качества работы системы собирается набор запросов, которые должны быть характерны для данной системы.

2. Для каждой пары запрос-документ экспертом выставляется оценка в зависимости от того, насколько точно документ отвечает на запрос

3. Система тестируется на этих запросах и для каждого запроса вычисляется, насколько близко построенное ранжирование к экс-

пертному.

В данной работе подобным набором запросов и документов является собранный компанией Microsoft набор данных Letor [4]. В этом наборе пара документ-запрос представлена 46-ю характеристиками и 3-ех ранговой оценкой соответствия документа запросу.

Процедура использования ранжирующего алгоритма состоит в следующем:

- Создается набор запросов, на которых ранжирующий алгоритм будет обучаться
- Для некоторых документов, подходящих под запросы, специально обученные эксперты выставляют оценку соотвествия документа запросу
- По этим данным строится ранжирующий алгоритм
- Ранжирующий алгоритм тестируется на скрытых от него данных
- Если тестирование прошло успешно, то получившийся алгоритм какое-то время используется для ранжирования на практике

Следует отметить, что сеть Интернет динамически изменяется в каждую секунду: появляются новые сайты, изменяется содержимое вебстраниц, меняется поведение пользователей. Поэтому крайне важным свойством выдачи является стабильность её качества: малым изменениям входных данных алгоритма должны соответствовать малые изменения порядка выдачи.

Одной из моделей, используемых для ранжирования, является дерево решений. Дерево решений в каждой узловой вершине имеет условие на факторы документа, по которому выбирается следующий узел дерева, а в листьях – константное значение. По своей природе дерево решений является кусочно-постоянной функцией, а значит, малые изменения во входных данных могут привести к большим изменениям в выходном значении. Данная дипломная работа посвящена решению проблемы ненепрерывности деревьев решений.

1. Постановка задачи

Цель данной дипломной работы заключается в разработке новых алгоритмов обучения ранжированию, решающих проблему непрерывности деревьев решений. Для этого требуется выполнить следующие задачи:

- Создать модель близости точек к листьям деревьев решений, учитывающую природу факторов
- Разработать алгоритм построения деревьев решений, учитывающий близкие точки к листьям
- Создать модификацию деревьев решений, вычисляющую значение полиномиальной регрессией значимых факторов
- Провести тестирование на открытом наборе данных
- Сравнить результаты с существующими методами обучения ранжированию

2. Обзор существующих решений

2.1. Градиентный бустинг

Практика показывает, что взвешенная сумма (ансамбль) простых моделей дает более высокую точность, чем сложные модели. Для построения такой взвешенной суммы можно использовать метод градиентного спуска: действительно, в градиентном спуске вектор следующей итерации получается из вектора предущей итерации с помощью аддитивной поправки. Таким образом, имея базовое множество функций H и функцию потерь Ψ, мы можем использовать метод градиентного спуска для построения взвешенной суммы, минимизирующей потери: алгоритм начинает оптимизацию с h_0 из множества H, а потом итеративно добавляет новые функции $h_{k+1}(x) = h_k(x) + \alpha \cdot \partial h_k(x)$, где α – это небольшая константа, а $\partial h_k(x)$ – специально выбранная функция из H. Идея градиентного бустинга состоит в том, чтобы добавить такую функцию $\partial h_k(x)$, которая бы наиболее быстро уменьшала функцию потерь Ψ [2].

В случае обучения ранжированию данный алгоритм работает следующим образом: есть множество запросов $Q = \{q_1, \ldots, q_n\}$, для каждого запроса q есть множество документов $D_q = \{D_{q,1}, \ldots, D_{q,m_q}\}$, где каждый документ описывается численным вектором фиксированной размерности. Кроме того, для каждой пары запрос-документ есть оценка того, насколько этот документ подходит к этому запросу $L_q = \{L_{q_1}, \ldots, L_{q_m}\}$. Функция потерь Ψ вычисляется как сумма квадратов отклонений:

$$\Psi(h_k) = \sum_q \sum_{i=1}^{m_q} (h_k(D_{q_i}) - L_{q_i})^2 \tag{1}$$

$$\partial\Psi(h_k) = \sum_q \sum_{i=1}^{m_q} 2 \cdot (h_k(D_{q_i}) - L_{q_i}) \partial h_k(D_{q_i}) \tag{2}$$

Для того, чтобы наилучшим образом уменьшить функцию потерь, приращение ∂h_k должно в каждой точке хорошо приближать разницу между текущим значением взвешенной суммы h_k и истинным L. При-

ращение ищется в базовом множестве H как минимум квадратичного отклонения текущей функции от истинной:

$$\underset{\partial h_k \in H}{\operatorname{argmin}} \sum_{i=1}^{m_q} (h_k(D_{q_i}) + \alpha \partial h_k(D_{q_i}) - L_{q_i})^2 \tag{3}$$

Коэффициент α и количество слабых моделей N являются гиперпараметрами алгоритма и подбираются для каждой задачи вручную.

В одной из своих работ Джером Фридман показал, что если на каждом шаге градиентного бустинга «обучаться» на случайной части обучающего множества, то качество получаемой модели улучшится [3]. В данной работе использовалась модификация этой идеи: на каждом шаге обучающее множество набиралось из исходного выборкой с повторениями [5].

2.2. Деревья решений

Как правило, в качестве набора функций H в градиентном бустинге используются деревья решений, которые имеют одну и ту же глубину и на каждом уровне имеют одно и то же одинаковое условие — так называемые "забывчивые деревья решений" [8]. При таком построении для глубины $depth$, дерево будет содержать 2^{depth} листьев. Такое дерево можно рассматривать как разбиение пространства факторов на непересекающиеся регионы, соответствующие различным поддеревьям (например, листьям). Листу соответствует предсказываемое значение, которое будет возвращено моделью для нового наблюдения. В случае квадратичной функции потерь оптимальным значением будет среднее по ответам точек обучающей выборки, попавшим в рассматриваемый лист.

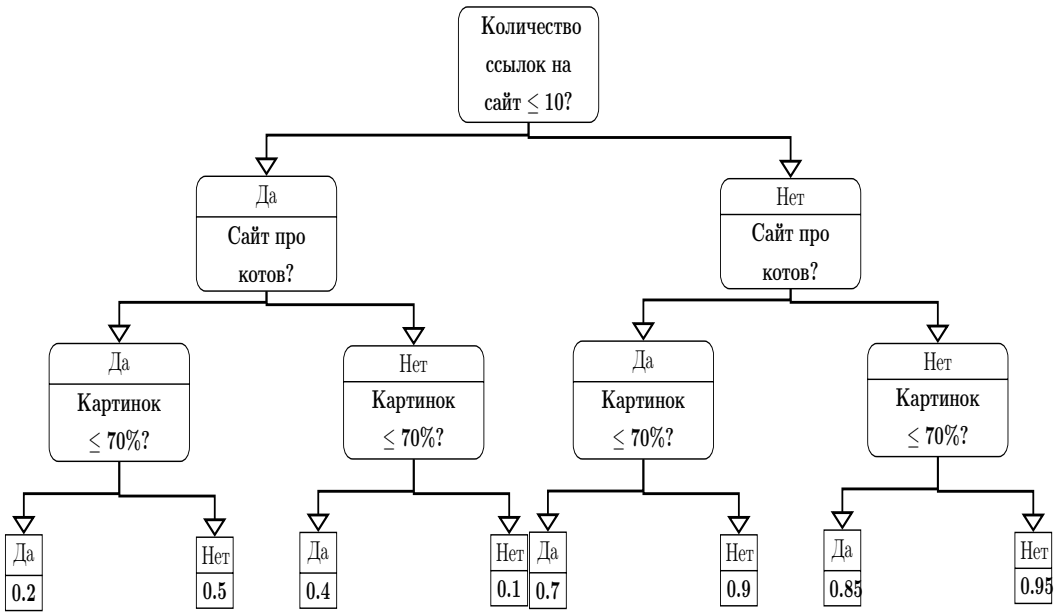


Рис. 1: Пример забывчивого дерева решений

Алгоритм построения забывчивого дерева решений состоит в последовательном добавлении в дерево новых разбиений. Разбиение, максимизирующее точность на обучающем множестве, – это разбиение, минимизирующее разброс внутри получившихся регионов. Поскольку в листьях для предсказания ответа используется выборочная статистика, следует следить за мощностью листа: если точек в листе мало, статистика посчитается неточно. Для борьбы "маленькими" листьями вводится аддитивный штраф за количество точек внутри листа. Среди разбиений пространства на регионы более точным будем считать то, у которого меньше суммарная ошибка внутри регионов и штраф за количество точек.

В качестве возможных разбиений используются не все возможные разбиения фактора — это сделало бы обучения слишком трудоёмким. Отсортируем наблюдаемые значения факторов. Разобьем полученный массив на N равных по объему частей. В качестве возможных разбиений фактора будем использовать максимальное и минимальное значения в этих частях [5].

3. Новые модели

3.1. Нечеткие границы

Как говорилось ранее, недостатком деревьев решений является их кусочно-постоянная природа. Рассмотрим пример из рисунка 1: первым разбиением является предикат "Количество ссылок на сайт ≤ 10", выбор этого разбиения подтверждается интуитивным соображениям, что на качественный сайт будут ссылаться другие сайты. Однако для сайтов, для которых количество ссылок близко к граничному, невозможно достоверно определить, к какому множеству принадлежит сайт: с течением времени количество ссылок может немного измениться и предсказанная релевантность существенно поменяется. Для решения этой проблемы предлагается концепция стохастических регионов.

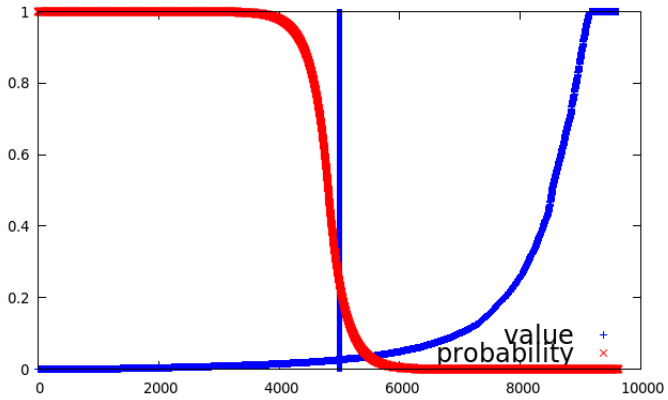
Разбиения в забывчивых деревьях решений имеют границы вида $x_i = c$, точки, у которых значение факторов меньше или равно c, попадают в одно множество, остальные — в другое. Для каждого из этих множества введем вероятность нахождение в нём зависящую от положение точки относительно множества.

Задача построения меры близости осложняется тем, что алгоритм не имеет информации о природе факторов и их распределении, поэтому в общем случае для оценки близости точки и границы нельзя полагаться на расстояние (абсолютную величину разницы значений).

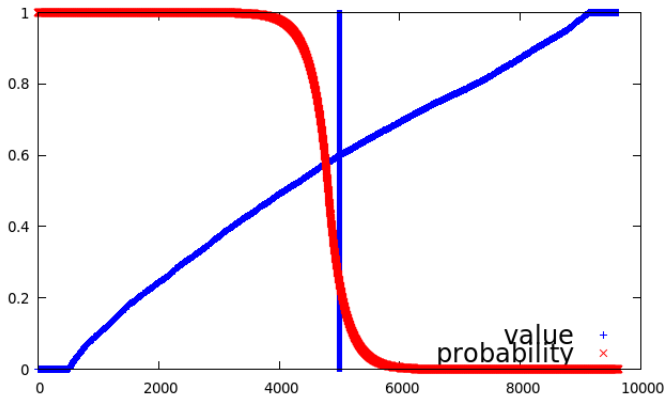
Для построения модели близости будем использовать наблюдаемое распределение фактора. Отсортируем наблюдения фактора $X_1 \leq X_2 \leq \cdots \leq X_N$. Обозначим за T_i истинное, скрытое значение фактора. Истинный порядок $i_1, \ldots, i_N$ такой что $T_{i_1} \leq T_{i_2} \leq T_{i_N}$, нам неизвестен поскольку неизвестны значения T_i. Введем следующую модель для перестановки, содержащей k инверсий в последовательности X: вероятность того, что она является истинной, пропорциональна $pSwap^k$. В этой модели вероятность того, что элемент, стоящий на позиции i в измеренном разбиении, в действительности должен стоять на позиции j, пропорциональна $pSwap^{|i-j|}$. Вычислим вероятность этого $T(x_i) \leq T(c)$,

не умаляя общности предположим, что $x_i > c$. Пусть L – индекс первого элемент в последовательности X, который равен x_i, B – индекс первого элемента, который больше c.

$$P(T(x_i) \leq T(c)) = \sum_{j=0}^{B-1} pSwap^{L-j} = \frac{pSwap^{L-B+1} - pSwap^{L+1}}{1 - pSwap} \tag{4}$$



(a) Длина ссылки



(b) Длина сайта

Рис. 2: Вероятность принадлежности к множеству слева от разделителя

Как видно на рисунке 2, для факторов, имеющих разное распределение, функция вероятности попадания в регион одинакова.

Используя эту модель для каждой точки, вычислим вероятность принадлежности точки x каждому из листьев l, обозначим ее $P(x \in l)$. Предсказываемое значение в листьях теперь будем вычислять с учетом

этого веса:

$$F(l) = \frac{\sum\limits_{(x,y)\in Learn} P(x \in l) \cdot y}{\sum\limits_{(x,y)\in Learn} P(x \in l)} \tag{5}$$

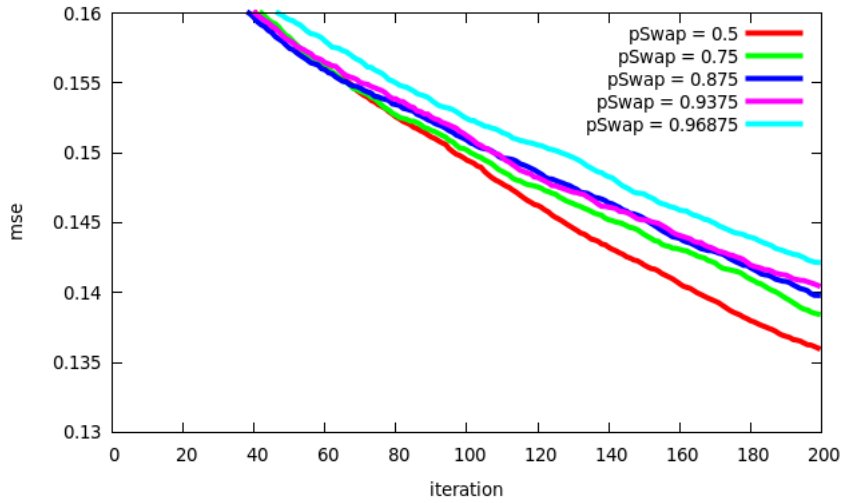


Рис. 3: Среднеквадратичная ошибка на обучающем множестве

Чем ближе $pSwap$ к единице, тем больше влияние точек, не попавших в регион. На рисунке 3 представлена зависимость функции потерь в зависимости от итерации. Видно, что, чем больше $pSwap$, тем медленнее обучается градиентный бустинг из таких моделей.

3.2. Полиномиальная регрессия листьев деревьев решений

В листьях дерева решений будем вычислять значение целевой функции полиномиальной регрессией факторов, по которым построено дерево. Пусть x_i – значение i-ого фактора, по которому происходило построение дерева, для точки x. Для удобства введем фиктивный фактор $x_0 = 1$. Значение модели для точки x, принадлежащей региону R^l, будет вычисляться по формуле:

$$F(x, y^l) = \sum_{I:i_1,i_2\ldots i_n} x_{i_1} \cdot x_{i_2} \cdots x_{i_n} \cdot y^l_I \tag{6}$$

Где y_I^l – коэффициенты регрессии в регионе R^l, соответствующем листу l, а $I = i_1, \ldots, i_n$ — мультииндекс, задающий используемые признаки. В качестве первого приближения для коэффициентов регрессии используем те, которые минимизируют сумму квадратов отклонений в точках обучающего множества, попавших в регион.

$$y_1^l = \underset{y}{\operatorname{argmin}} \sum_{x \in Learn, x \in R^l} (F(x, y) - target(x))^2 \tag{7}$$

Так как приведёная выше формула является квадратичной формой, необходимым и достаточным условием минимума будет выполнение для всех y_I:

$$\sum_{x \in Learn, x \in R^l} \sum_{J: j_1, j_2 \ldots j_n} x_{i_1} \cdot x_{i_2} \cdot \cdots \cdot x_{i_n} \cdot y_I \cdot (x_{j_1} \cdot x_{j_2} \cdot \cdots \cdot x_{j_n} \cdot y_J - 1) = 0 \tag{8}$$

Для каждого региона получаем независимую система линейных уравнений, которая задает коэффициенты регрессии.

Чтобы дерево было более непрерывным, введем штраф за разницу между коэффициентами соседних регионов.

$$y_2^l = \underset{y}{\operatorname{argmin}} \sum_{x \in Learn, x \in R^l} (F(x, y) - target(x))^2 + \lambda \cdot \sum_{n \in neighbors(l)} ||y^l - y_1^n||^2 \tag{9}$$

Так как y_1^n является константой, производная формулы на второй итерации также задает систему линейных уравнений, решение которой задают коэффициенты регрессии.

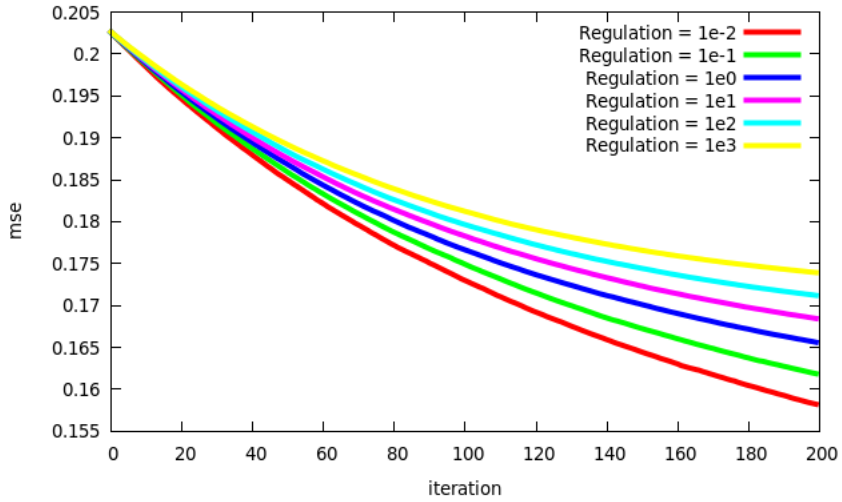


Рис. 4: Среднеквадратичная ошибка на обучающем множестве

Чем больше коэффициент λ, тем выше влияние соседних регионов, и, как видно из графиков на рис. 4, тем медленнее обучается модель.

4. Эксперименты

4.1. Нормированная взвешенная сумма выигрышей

Нормированая взвешенная сумма выигрышей (NDCG) – метрика качества ранжирования, вычисляемая по списку документов с оценкой релевантности. Сумма выигрышей вычисляет ценность документа в зависимости от его места в списке. Ценность документа вычисляется как 2 в степени оценки этого документа минус 1. Ценность позиции i в ранжировании принимается равной $\frac{1}{log_2(i+1)}$, для первого документа она равна 1, для второго – уже ≈ 0.63. Так как пользователь смотрит далеко не все документы, DCG считается по первым p документам[6].

$$DCG_p(q) = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{10}$$

$$NDCG_p(q) = \frac{DCG_p(q)}{IDCG_p(q)} \tag{11}$$

Где $IDCG_p(q) = DCG_p(q)$ при идеальном ранжировании документов.

4.2. Набор данных

Исследование проводилось на основе широко используемого набора данных Letor. Этот набор данных использовался более чем в ста работах об обучении ранжированию. Данные представленны в виде пар документ-запрос, которые оценены по трехранговой шкале (0 – документ не содержит искомой информации, 1 – документ содержит часть информации, 2 – документ идеально подходит под запрос). Информация о документе и запросе представлена в виде 46 факторов.

Номер фактора	Описание фактора
16	Длина основной части страницы
20	Длина всей страницы
41	Количество ссылок на страницу
42	Количество ссылок на странице
46	Количество подстраниц страницы

Таблица 1: Примеры факторов из Letor

4.3. Тестирование

Набор данных, на которых производится тестирование, разделен на 3 множества: обучающее, валидирующее и тестовое. Во время обучения программе доступно только обучающее множество. Валидирующее множество используется для того, чтобы измерять качество модели в процессе поиска параметров модели, шага бустинга и количества моделей в ансамбле. После этого модель проверяется на тестовом множестве, после этого параметры модели не изменяются.

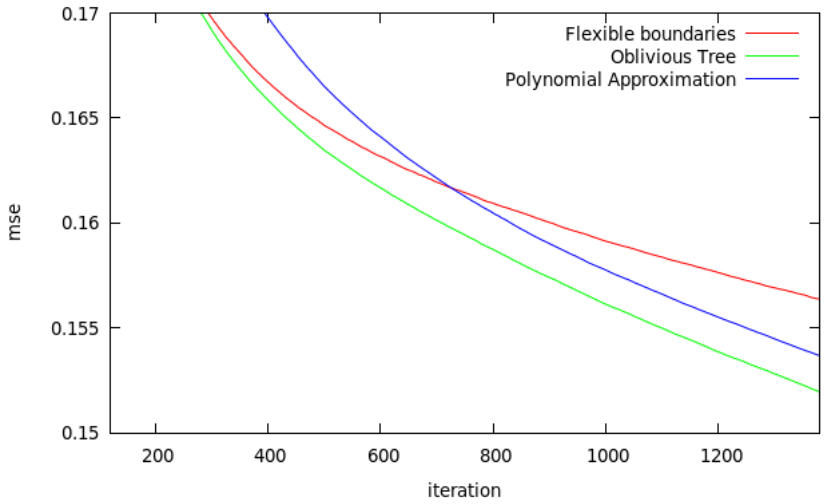


Рис. 5: Среднеквадратичное отклонение на обучающем множестве

По метрике среднеквадратичного отклонения MSE все методы стабильно обучались как на обучающем, так и на тестовом множестве. Примечателен тот факт, что хотя деревья с нечеткими границами хуже обучались на обучающем множестве, на тестовом множестве они

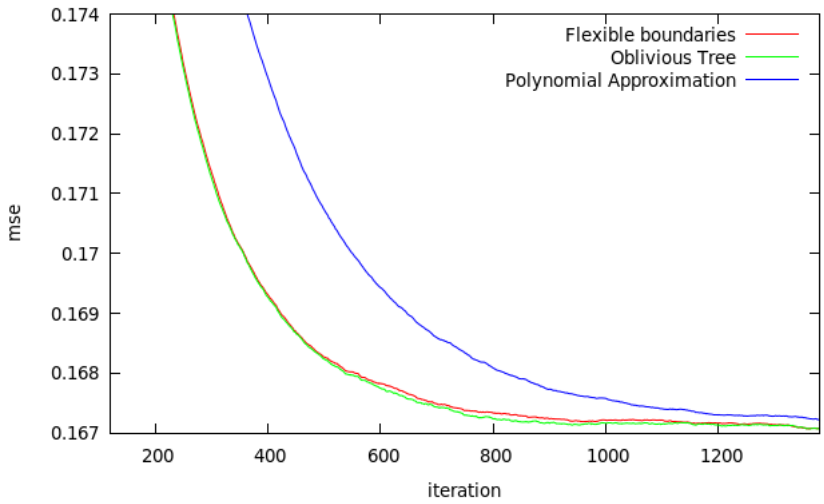


Рис. 6: Среднеквадратичное отклонение на тестовом множестве

обучались так же, как и деревья с четкими границами.

5. Сравнение с существующими методами

Алгоритм градиентного бустинга содержит в себе недетерминированность. Поэтому для измерения качества работы модели недостаточно его измерения на одном запуске. Для получения статичстически достоверных данных о том является ли один алгоритм лучше другого, нужно собрать статистику запусков. Путём запуска алгоритма 100 раз было получено 100 различных замеров качества алгоритмов на тестовом множестве. Для сравнения двух выборок использовался критерий Уилкоксона [9]. Этот критерий для двух выборок вычисляет вероятность того, что их средние значение одной выборки больше средние значение другой. В дальнейшем это вероятность будет обозначаться $pValue$. Чем ближе $pValue$ к нулю, тем более достоверно можно утверждать о превосходстве точности одного метода над другими.

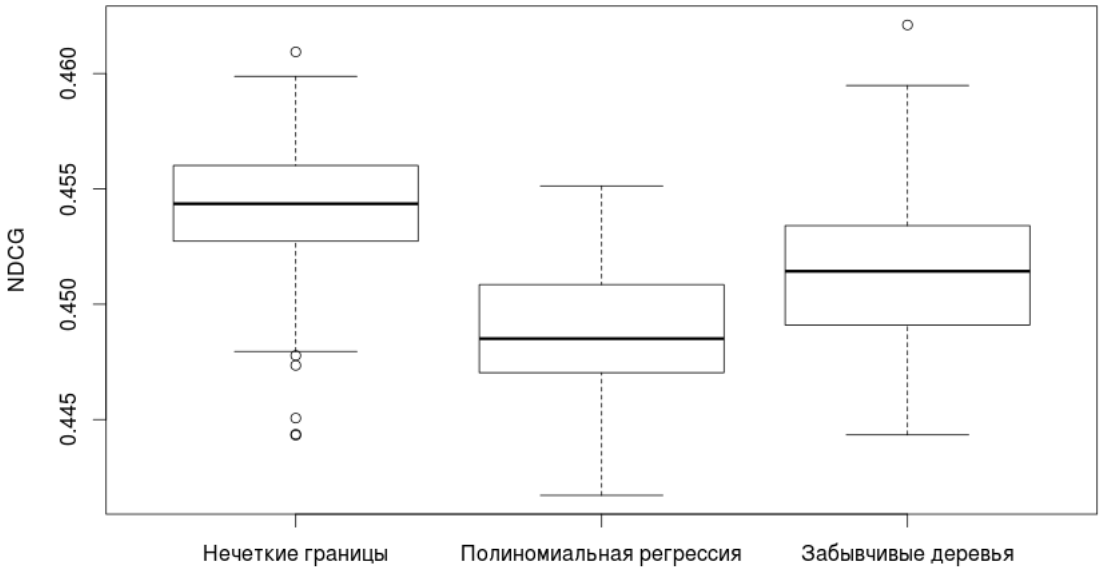


Рис. 7: Распределения значения $NDCG_5$

Тест Уилкоксона показал, что по метрике $NDCG_5$ все модели статистически значимо различаются. Наилучший результат показали деревья решений с нечеткими границами 0.4543 ± 0.0005, забывчивые деревья 0.4514 ± 0.0007, а деревья с полиномиальной регрессией $0.4485 \pm$

0.0006. Алгоритм обучения ранжированию ListNet[7] на этом наборе данных равен 0.4565, что значимо больше всех представленных в данной работе алгоритмов. По метрике MSE деревья с нечеткими границами значимо уступают остальным двум методам.

Сравниваемые методы	Значимость разницы
Полиномиальная регрессия $<$ Забывчивые деревья	3.449e-08
Полиномиальная регрессия $<$ Нечеткие Границы	2.2e-16
Забывчивые деревья $<$ Нечеткие границы	2.834e-09

Таблица 2: Сравнение методов по метрики $NDCG$

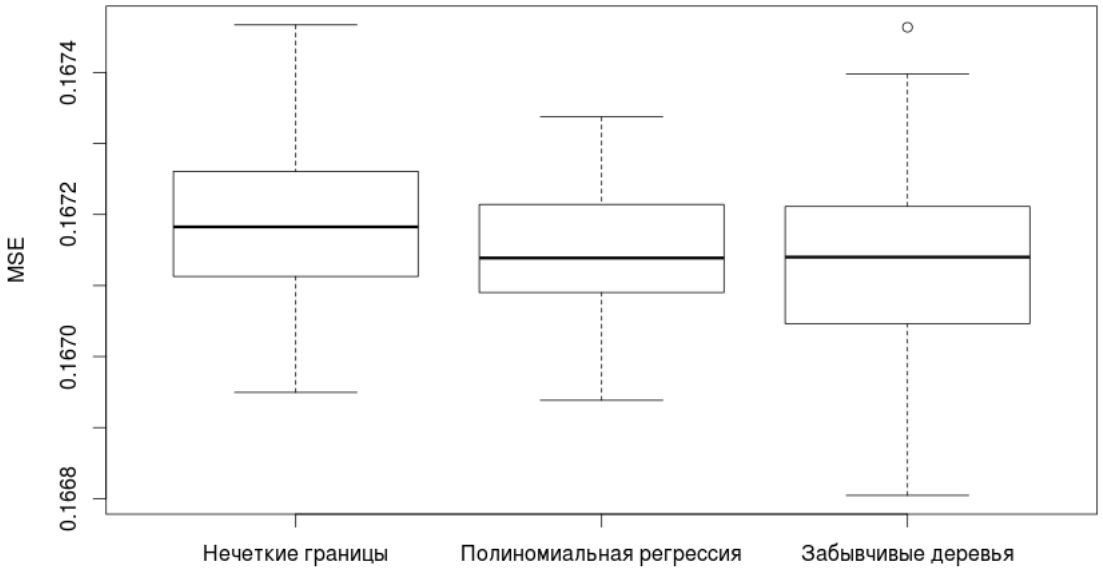


Рис. 8: Распределения значения MSE

Сравниваемые методы	Значимость разницы
Полиномиальная регрессия $<$ Забывчивые деревья	0.09536
Полиномиальная регрессия $<$ Нечеткие Границы	0.0091
Забывчивые деревья $<$ Нечеткие границы	0.001814

Таблица 3: Сравнение методов по метрики MSE

Заключение

Цель данной дипломной работы состояла в создании модификаций алгоритмов построения забывчивых деревьев решений, решающих проблему ненепрерывности. Для этого были выполнены следующие задачи:

- Реализован алгоритм построения деревьев решений с нечеткими границами регионов. В данной модели расстояние между точкой и границей вычислялось как количество точек более близких к границе региона. Такой подход позволил использовать одну метрику для факторов, имеющих разное распределение. В качестве вероятности принадлежности точки к региону используется экспоненциально убывающая функция от расстояния между точкой и границей региона. Значение в регионе вычисляется по всем точкам обучающего множества с учетом вероятности принадлежности точки региону. Значение регрессии для точки вычисляется по всем регионам дерева с учетом вероятности принадлежности точки региону.

- Разработан алгоритм построения полиномиальной регрессии в листьях деревьев решений. На первом шаге находятся коэффициенты регрессии, минимизирующие сумму квадратов отклонений регрессии на обучающем множестве. На втором шаге ведется поиск коэфициентов, минимизирующих сумму квадратов отклонений регрессии и сумму расстояний до коэффициентов регрессии соседних регионов, полученных на первом шаге.

- Разработанные алгоритмы были протестированы на открытом наборе поисковых данных Letor [4]. В этом наборе документы представлены набором из 46 факторов, описывающих веб-страницу, сайт, поисковый запрос и коэффициенты соответствия теста сайта тексту запроса.

- Проведено сравнение с существующими методами обучения ранжированию: градиентным бустингом забывчивых деревьев решений и ListNet'ом [7]

Метод	$NDCG_5$
Полиномиальная регрессия	0.4485 ± 0.0006
Забывчивые деревья	0.4514 ± 0.0007
Нечеткие границы	0.4543 ± 0.0005
ListNet	0.4565

Таблица 4: Метрика $NDCG$ на тестовом множестве

Все представленные в работе методы уступают по качеству ранжирования методу ListNet [7]. В ходе проведенного тестирования выявлено, что модели с нечеткими границами значимо превосходят по метрике $NDCG_5$ забывчивые деревья, которые, в свою очередь, превосходят забывчивые деревья с полиномиальной регрессией. По метрике MSE деревья с нечеткими границами значимо уступают остальным методам, между деревьями с полиномиальной регрессией и забычивыми деревьями значимого различия не обнаружено.

Список литературы

[1] Cyril Cleverdon. 1960. Report on the first stage of an investigation into the comparative efficiency of indexing systems. Technical report, ASLIB Cranfield Project.

[2] Greedy Function Approximation: A gradient boosting machine, Jerome H. Friedman, 1999.

[3] Stohastic Gradient Boosting, Jerome H. Friedman, 1999.

[4] T.-Y. Liu, T. Qin, J. Xu, W. Xiong, and H. Li. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In LR4IR 2007: Workshop on Learning to Rank for Information Retrieval, in conjunction with SIGIR 2007, 2007.

[5] Andrey Gulin, Igor Kuralenok, Dmitry Pavlov, Winning The Transfer Learning Track of Yahoo!'s Learning To Rank Challenge with YetiRank, 2011.

[6] Kalervo Jarvelin, Jaana Kekalainen: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems 20(4), 422–446 (2002)

[7] Cao, Z., Qin, T., Liu, T., Tsai, M., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. Proc. of the Int. Conf. on Mach. Learn. (pp. 129–136)

[8] Ron Kohavi and Chia-Hsin Li. Oblivious decision trees graphs and top down pruning. In Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2, pages 1071–1077, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc

[9] Frank Wilcoxon "Individual comparisons by ranking methods", 1945