Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных сетей
Кафедра системного программирования

Киргизов Григорий Валерьевич

Использование алгоритмов машинного обучения для определения неэффективно кэшируемой нагрузки

Курсовая работа

Научный руководитель:
Руководитель исследовательской лаборатории ООО "Рэйдикс", к. т. н. Лазарева С. В.

Санкт-Петербург
2017

Оглавление

Введение

Сложно переоценить значение систем хранения данных (СХД). Они являются основой центров обработки данных и позволяют претворять в реальность решения сложнейших задач, требующих хранения и обработки огромных массивов данных. Каждый процент их производительности может вылиться в массу сэкономленных временных и денежных ресурсов.

Насколько СХД необходимы для функционирования современного мира, настолько жизненно важным для самих СХД является вопрос эффективного кэширования. Особенно это верно для СХД с твердотельными накопителями (SSD) в качестве кэша. Но традиционные алгоритмы кэширования не предназначены для работы с SSD, что приводит к их ускоренному износу. Необходимы специальные алгоритмы кэширования и новые подходы к решению этой проблемы.

Тема производительности СХД всегда лежала в рамках системного программирования. Несмотря на то, что это естественное положение вещей, оно может ограничивать создателей алгоритмов в арсенале своих методов, тем самым не позволяя достичь наилучших результатов.

Эта работа является попыткой преодолеть этот барьер и исследовать возможности анализа данных для увеличения эффективности кэширования в гибридных СХД. Данные — это истории запросов к системе. Подробное описание особенностей кэширования в таких СХД представлено в предыдущей работе [12].

Актуальность глубокого обучения

Методы глубокого обучения позволили достичь огромного прогресса в таких областях, как распознавание изображений и обработка текста. В контексте данной работы особый интерес представляют задачи последней области, так как с ними можно провести существенную параллель: и тексты, и последовательности запросов к СХД представляют собой данные со сложными временными зависимостями.

По этой причине именно на методы глубокого обучения, разработан-

ные для обработки естественных языков, обращено основное внимание в этой работе. В первую очередь, это рекуррентные нейронные сети, которые благодаря своей архитектуре (топологии связей между параметрами вероятностной модели) позволяют достичь хороших результатов на задачах классификации данных, имеющих последовательную природу. Полноценное введение в глубокое обучение можно найти в [8], и, в частности, в рекуррентные нейронные сети в [9].

1. Постановка задачи

Цель данной работы — исследовать применимость методов машинного обучения для увеличения эффективности кэширования в гибридных системах хранения данных.

1. Провести обзор методов машинного и, в частности, глубокого обучения, подходящих для решения данной задачи;
2. Составить план экспериментов с учетом обзора;
3. Подготовить данные для их проведения;
4. Провести эксперименты;
5. Выполнить анализ результатов.

2. Обзор существующих решений

Обзор алгоритмов кэширования для СХД с твердотельными накопителями представлен в [12]. Далее будут рассмотрены методы машинного обучения, которые применяются для решения задач с данными схожей структуры

Стоит отметить, что в перечисленных работах эксперименты зачастую проводятся на небольших и хорошо изученных проблемах. Соответственно, сложно судить об их применимости к реальным задачам и реальным данным. Более того, задача, рассматриваемая в данной работе, не имеет прямых аналогов среди рассматриваемых исследователями. Их работы в основном рассматривают задачи обработки натуральных языков и изображений. Ответ на то, окажутся ли эти методы эффективными, могут дать только эксперименты.

2.1. Ансамбли деревьев решений

Примером традиционных методов машинного обучения являются ансамбли деревьев решений. Одна из его эффективных реализаций — библиотека XGBoost [2], которая показала хорошие результаты во многих соревнованиях по анализу данных.

2.2. Глубокое обучение и рекуррентные нейронные сети

Классические методы машинного обучения требуют хорошего представления данных для получения приемлемых результатов. Создание новых признаков и проверка того, ведут ли они к улучшению обученной с ними модели, могут занимать большую часть работы с традиционными методами машинного обучения. Методы глубокого обучения являются альтернативой этому благодаря тому, что модели в ходе обучения способны находить подходящее преобразование данных [8, с. 155-165].

Более того, традиционные методы машинного обучения не справляются с моделированием протяженных во времени зависимостей в дан-

ных. В то же время, существует группа методов глубокого обучения, предназначенная для моделирования таких зависимостей (например, для работы с временными рядами) — рекуррентные нейронные сети.

2.2.1. Long Short Term Memory (LSTM) Recurrent Neural Networks (RNN)

Эти нейронные сети [6] показывают значительно лучшие результаты на данных с протяженными во времени зависимостями, чем базовые рекуррентные нейронные сети. Это достигается за счет более сложного "строительного блока" (LSTM unit) нейронной сети. Фактически, эта модель стала стандартом для работы с последовательными данными [8, с. 410-411].

2.2.2. Архитектурные модификации рекуррентных нейронных сетей

Авторы этой работы [1] исследуют, какие модификации рекуррентных нейронных сетей приводят к улучшению динамики и результатов их обучения. В частности, они показывают, что увеличение глубины (количества слоев) рекуррентных сетей не всегда приводит к лучшим результатам, в то время как добавление дополнительных связей внутри слоя (т.н. skip-connections) может значительно улучшить результаты обучения на данных с протяженными во времени зависимостями.

2.2.3. Частичное обучение с учителем

В данной работе [3] рассматривается использование неразмеченных данных в задачах классификации последовательностей с использованием рекуррентных нейронных сетей. Авторы предлагают два подхода с использованием предобучения моделей:

- Предобучение на задаче прогнозирования следующего элемента в последовательности;

- Предобучение автоэнкодера [8, с. 502-525] на следующей задаче: найти более компактное представление последовательности (меньшей размерности), поданной на вход, и затем восстановить исходную последовательность из этого представления. Автоэнкодер может найти более информативное представление данных, и затем использоваться для преобразования данных в основной модели.

Авторы статьи утверждают, что благодаря предобучению этим задачам обучение основной модели на задаче классификации становится более стабильным и достигает меньшего уровня ошибки.

2.2.4. Иерархические рекуррентные нейронные сети

Под этим заголовком можно объединить несколько работ (например, [4, 10]), которые исследуют иерархический подход к построению нейронных сетей. Таким образом модели могут лучше обучаться временным зависимостям на разных масштабах. Например, в случае обработки языков нейронная сеть может представлять собой иерархию из 3 уровней, в которой 1 уровень моделирует символы, 2 уровень — слова, а третий, последний, уровень — предложения. При этом каждый уровень — это отдельная нейросеть, и каждый следующий уровень принимает на вход множество выходных данных от предыдущего уровня.

3. Описание работы

3.1. Инструментарий

- Язык программирования: Python, так как он является стандартным инструментом для работы с глубоким обучением благодаря возможности быстрого прототипирования и обилию библиотек для машинного обучения и анализа данных;
- Библиотеки для работы с данными: pandas, dask;
- Библиотеки для машинного и глубокого обучения: XGBoost, Keras.

3.2. Обработка данных

Данные — это истории запросов к системам хранения данных. О каждом запросе известна следующая информация:

- Время поступления запроса;
- Логический адрес первого блока (logical block address — lba);
- Число запрошенных блоков (len);
- Тип запроса — запись или чтение;
- Тип запроса — последовательный или случайный.

Истории запросов требуется разбить на блоки последовательных запросов, т.к. отдельный запрос не несет никакой существенной информации. При таком разбиении важным является размер блока — параметр blocksize.

3.2.1. Метрика

В качестве метрики эффективности кэширования, как указано в [12], используется Write Efficiency:

$$WE = \frac{\text{число кэш-попаданий}}{\text{число записей в кэш}}$$

Затем по этой метрике множество данных разбивается на классы для обучения моделей.

Исходные данные не содержат этой информации, поэтому сначала требуется разметить данные. Для одного запроса она вычисляется следующим образом: в истории просматриваются ближайшие windowsize запросов и подсчитывается количество запросов к тем же данным, что и в исходном. Для блока из blocksize запросов подсчитывается среднее значение метрики по блоку.

Вычисление метрики для больших значений windowsize может занять значительное количество времени. Накладываются вычислительные ограничения на то, насколько далеко можно подсчитывать переиспользование блока. Возникают два вопроса:

1. Каким выбрать параметр windowsize?

2. Является ли переиспользование блоков в пределах windowsize репрезентативным значением для суждения о переиспользовании этих блоков вообще?

Для ответа на них был проведен следующий эксперимент:

1. Взята случайная выборка запросов;

2. Подсчитано переиспользование этих блоков в пределах значительно большего windowsizemax = 1000000;

3. Подсчитан коэффициент корреляции Пирсона между переиспользованием блоков в пределах windowsizemax и windowsize, где windowsize – различные тестируемые значения.

На рис. 1 показано изменение коэффициента корреляции на выборке во времени. В таблице 1 представлено значение коэффициента корреляции для всей выборки. Результат значимый (P-значение близко к нулю). Его можно интерпретировать следующим образом: блоки, которые востребованы в ближайшие windowsize запросов, также, вероятно, востребованы в ближайшие windowsizemax запросов. Таким образом, для вычисления метрики берется значение windowsize, равное 100000.

Рис. 1: Динамика коэффициента корреляции между windowsize и windowsizemax. Сверху вниз, windowsize: 100000, 10000, 30000, 3000. Размер выборки: 1000000 запросов. Ось x: номер бегущего окна размера 1000 запросов; ось y: значение коэффициента корреляции для бегущего окна.

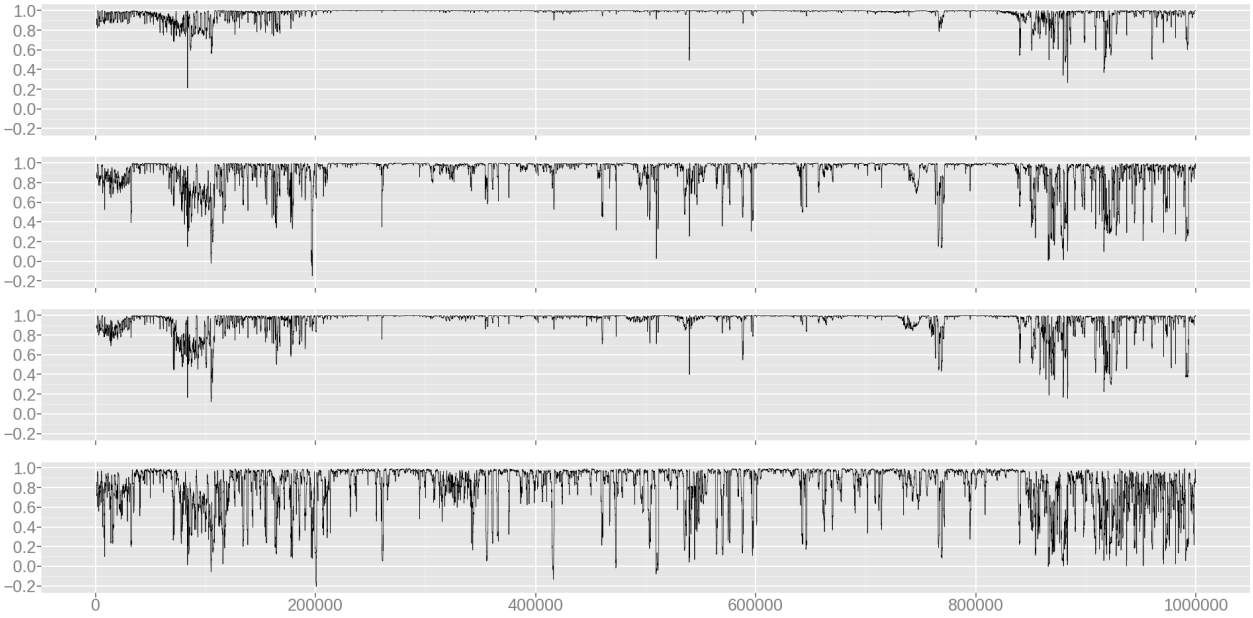


Таблица 1: Коэффициент корреляции Пирсона для различных windowsize.

windowsize	3000	10000	30000	100000
1000000	0.39	0.61	0.82	0.87

3.2.2. Разбиение наборов данных

Для построения вероятностных моделей набор данных стандартно [8, с. 120] делится на три части:

- Тренировочная выборка: 80%;
- Тестовая выборка: 10% (для итогового тестирования модели);
- Валидационная выборка: 10% (для подбора параметров и отслеживания прогресса модели во время обучения независимо от тренировочной выборки).

Стоит отметить, что в случае моделей глубокого обучения выборки

нельзя делать случайными, так как в таком случае будут разрушены временные связи между запросами.

3.3. Представление данных (признаки)

3.3.1. Частотные характеристики и производные от них

- *count_ lba* — количество использований lba в пределах блока;
- *count_ diff_ lba* — аналогично, частоты встречаемости разностей между lba текущего запроса и предыдущего;
- *count_ len* — аналогично, частоты встречаемости запросов к блокам данной длины len;
- *count_ diff_ len* — аналогично, частоты встречаемости разностей между len текущего и предыдущего запросов.

Далее для каждого признака подсчитываются статистики — моменты: среднее, стандартное отклонение, коэффициент ассиметрии и коэффициент эксцесса. Это и будут признаки.

3.3.2. Разностные характеристики и производные от них

Разность между lba текущего и i-предыдущего по счету блока (назовем ее diff_lba_i) может послужить в качестве основы нового признака. Каждая такая разность diff_lba_i распределена симметрично и имеет среднее, равное нулю. Поэтому различия между распределениями diff_lba_i для различных i можно описывать с помощью единственного числа — стандартного отклонения этой величины. Таким образом, для каждого блока получается набор из n значений std_diff_lba_i, где i = 1..10 (назовем этот набор *lba_ interdiffs*). Гипотеза состоит в том, что для хороших блоков запросов эти значения будут отличаться меньше, иначе говоря, будут более концентрированы, чем для плохих блоков. Концентрированность этих значений можно охарактеризовать с помощью их среднего и стандартного отклонения. Различия между ними

можно увидеть на рис. 4, который показывает, что гипотеза подтверждается.

Аналогично подсчитываются признаки для длин запрошенных данных (параметра len).

3.3.3. Анализ признаков

Для анализа различия между классами относительно признаков были построены диаграммы размаха (boxplots). Например, на рис. 3. отображены различия 1 и 2 моментов признака count_lba, и на рис. 4 представлены различия 1 и 2 моментов признака lba_interdiffs между классами.

Диаграммы размаха показывают, что ни один признак не проявляет достаточно значимых различий между классами для их определения только на его основе. В то же время, совокупность слабых некоррелирующих признаков может позволить различать классы блоков.

Важно отметить, что при вычислении признаков требуется взять большое значение парамера blocksize, так как на небольших блоках запросов многие признаки обращаются в ноль.

3.4. Проведенные эксперименты

Проведенные эксперименты можно разделить на две группы.

Первая — это эксперименты с непреобразованными блоками запросов и моделями глубокого обучения. Так проверяется предположение о том, что эти модели способны сами находить подходящее представление данных.

Вторая — это эксперименты с преобразованными блоками запросов (соглано разделу представления данных). Так как в таком случае количество данных для обучения существенно меньше, чем в первом случае (из-за большего значения параметра blocksize), то здесь подходят только стандартные методы машинного обучения. В данном случае — ансамбли деревьев решений.

3.5. Модели глубокого обучения

Построение модели глубокого обучения требует принятия множества решений. Привести подробное разъяснение каждого из них в данной работе не представляется возможным. Часть из них выбрана согласно рекомендациям в литературе [8] и оригинальных статьях. В них можно найти подробные мотивацию и пояснения.

Общие параметры:

- Целевая функция — перекрестная энтропия (стандартный выбор для задачи классификации) [8, с. 75];
- Оптимизационный алгоритм — Adam [11];
- Метод инициализации параметров — Glorot [7];
- Метод регуляризации — Dropout [5] с параметром 0.5.

Опробованные параметры:

- Количество рекуррентных слоев нейронной сети: 2, 3, 4;
- Количество выходных нейронов в каждом lstm unit: 256, 512;
- Количество временных шагов рекуррентной нейронной сети: 50, 100;
- Размер блока (количество запросов в блоке): 100, 300, 1000.

3.5.1. Базовый метод глубокого обучения — LSTM RNN

Было опробованы модели различной сложности. Для всех результат отрицательный. Динамика обучения представлена на рис. 5.

3.5.2. Предобучение и LSTM RNN

Было опробовано предобучение на задаче моделирования потока запросов (прогнозирования следующего запроса). В качестве целевой

функции использовалась среднеквадратичная ошибка между спрогнозированным значением lba и настоящим. Результат отрицательный: процесс обучения на побочной задаче не сходится. Динамика обучения представлена на рис. 6.

3.5.3. HRNN

Были опробованы модели для различных наборов параметров, с двумя уровнями: первый для каждого блока, второй для множества блоков. Результат эксперимента отрицательный. Динамика обучения представлена на рис. 7.

3.6. Ансамбли деревьев решений (XGBoost)

Динамику процесса обучения можно наблюдать на рис. 8. На тестовой выборке достигается ошибка классификации около 3.2%. Это позволяет судить об успешном обучении модели на тренировочной выборке. На рис. 9 можно увидеть прогноз модели на тестовой выборке.

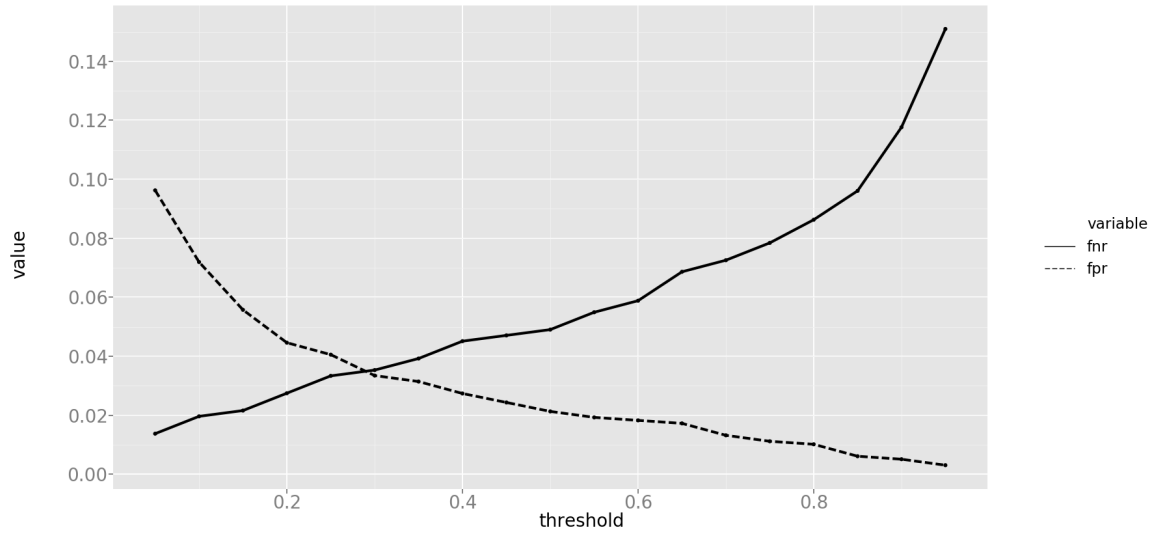
3.6.1. Ошибки 1 рода

В задаче кэширования важно не допускать ошибок 1 рода, то есть ложных положительных прогнозов. Другими словами, лучше пропустить блок, подходящий для кэширования, чем записать в кэш-память невостребованные данные. Для балансирования между вероятностью выдачи ложного положительного и ложного отрицательного прогнозов можно изменять барьер вероятности, после которого прогноз будет считаться положительным. Зависимость этих величин показана на рис. 2.

3.7. Итоги экспериментов

Для решения данной задачи существенным оказалось информативное представление данных. При этом существующие методы глубокого обучения не смогли найти подобное представление данных.

Рис. 2: Зависимость вероятности выдачи ложного положительного и ложного отрицательного прогнозов в зависимости от барьера вероятности, после которого прогноз считается положительным. Ось x: барьер вероятности, ось y: вероятность совершения ошибки. Пунктирная линия — вероятность совершения ложной положительной ошибки, сплошная линия — вероятность совершения ложной отрицательной ошибки.



Заключение

1. Выполнены следующие задачи:
2. Проведен обзор методов машинного и глубокого обучения;
3. Составлен план экспериментов;
4. Выполнена обработка данных;
5. Проведены эксперименты;
    - Получен положительный результат.

Приложения

Рис. 3: Различия для классов между средними и стандартными отклонениями признака count_lba.

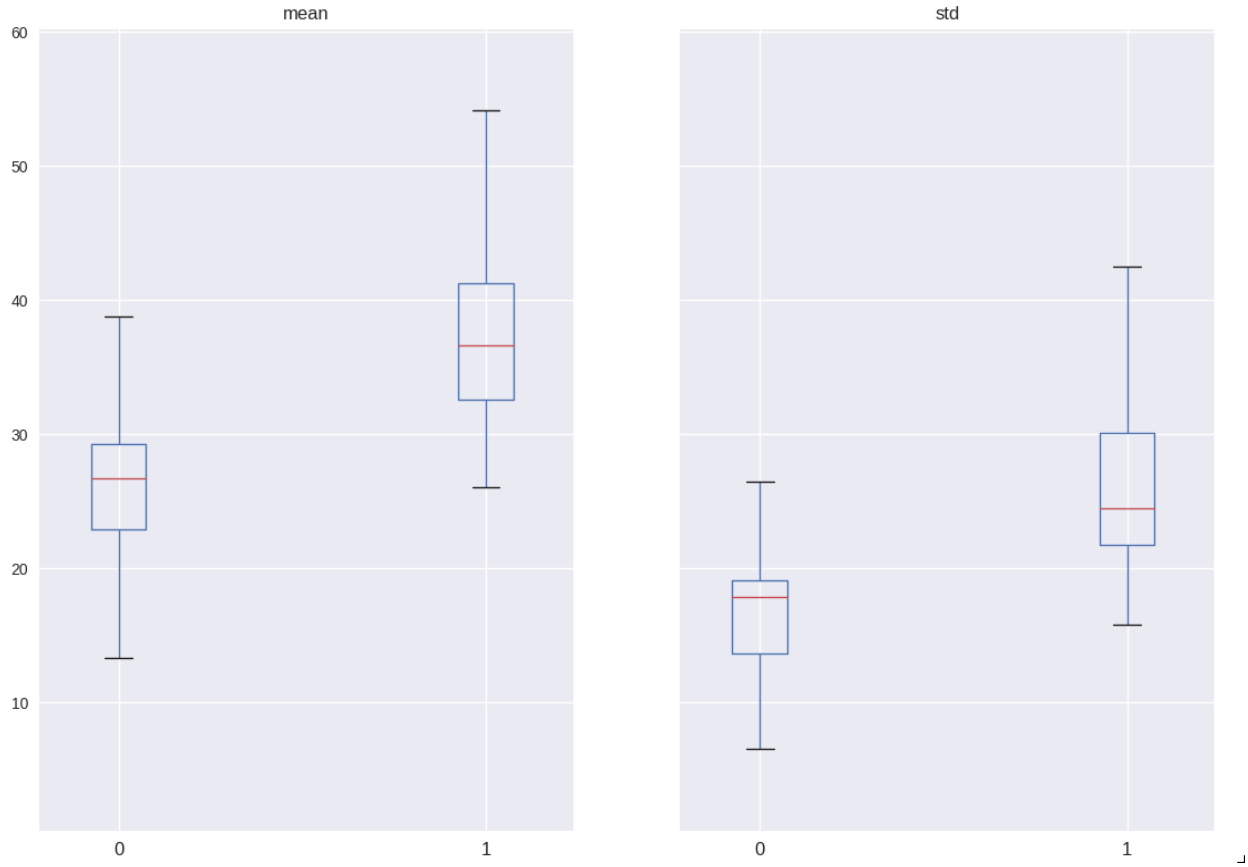


Рис. 4: Различия для классов между средними и стандартными отклонениями признака lba_interdiffs.

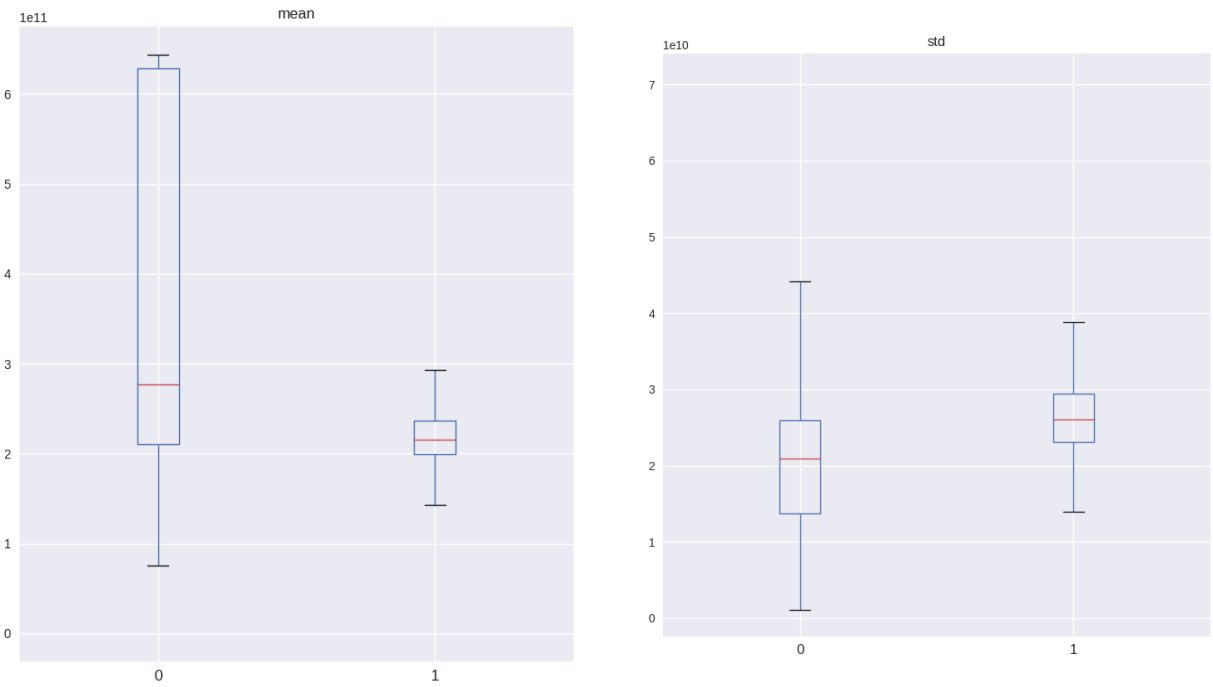


Рис. 5: LSTM RNN. Изменение значения функции потерь на протяжении обучения. Красный — тренировочная выборка, зеленый — тестовая. График для модели средней сложности с тремя рекуррентными слоями.

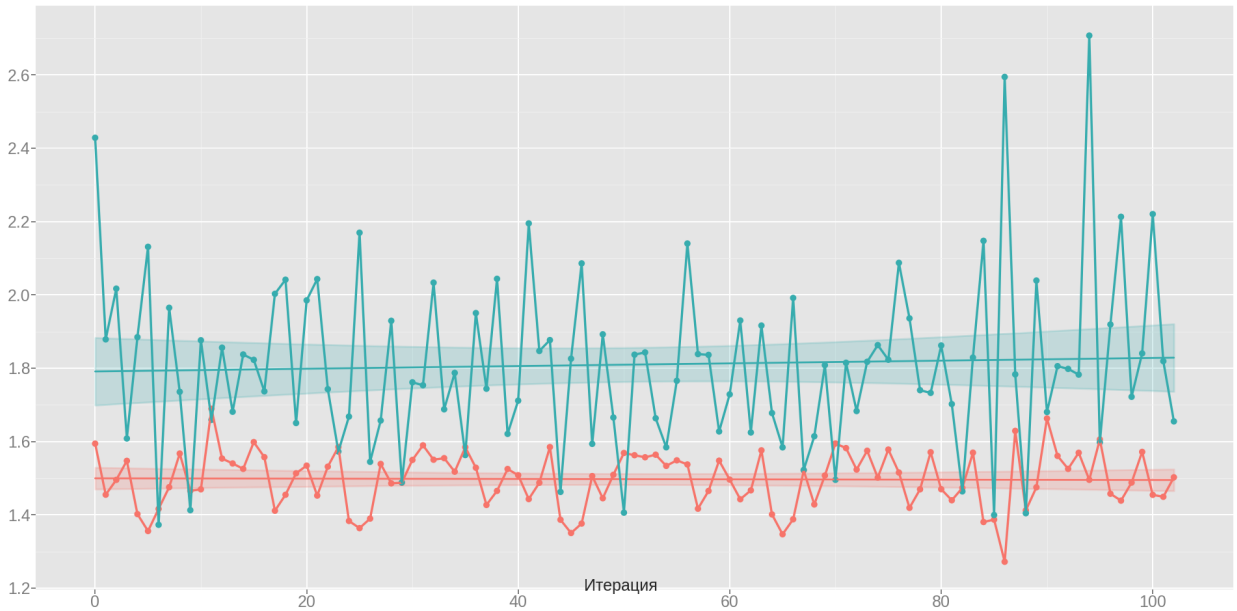


Рис. 6: Предобучение на задаче предсказания lba следующего запроса. Изменение среднеквадратичной ошибки на протяжении обучения. Красный — тренировочная выборка, зеленый — тестовая. График для модели с двумя рекуррентными слоями.

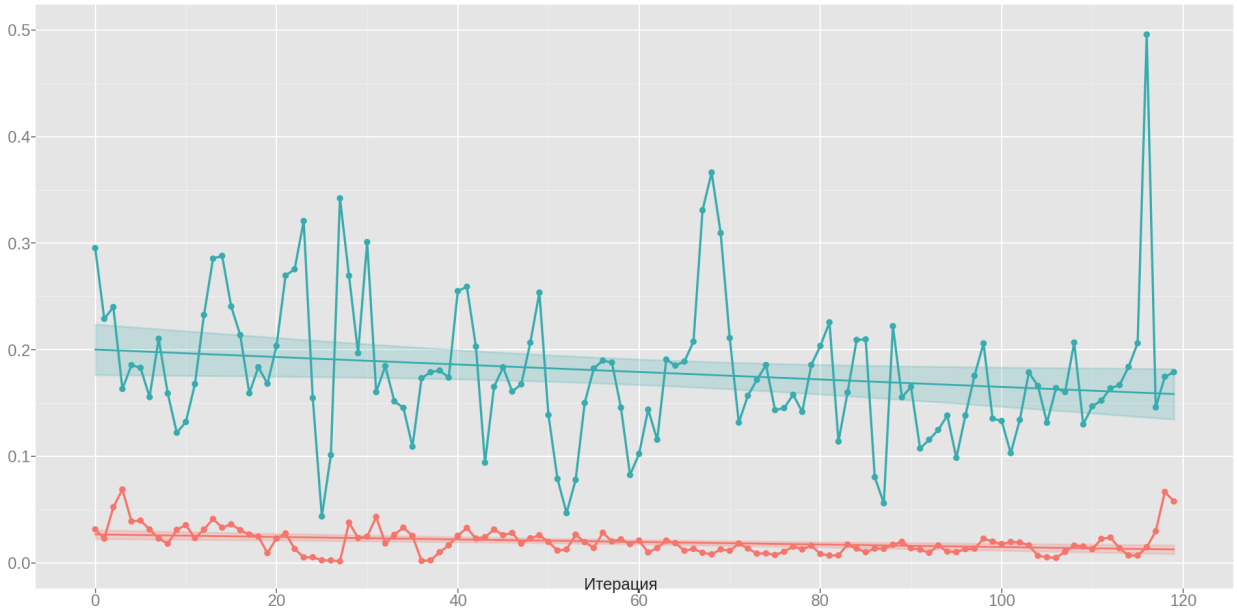


Рис. 7: HRNN. Изменение значения функции потерь на протяжении обучения. Красный — тренировочная выборка, зеленый — тестовая. График для модели средней сложности с двумя рекуррентными слоями на втором уровне.

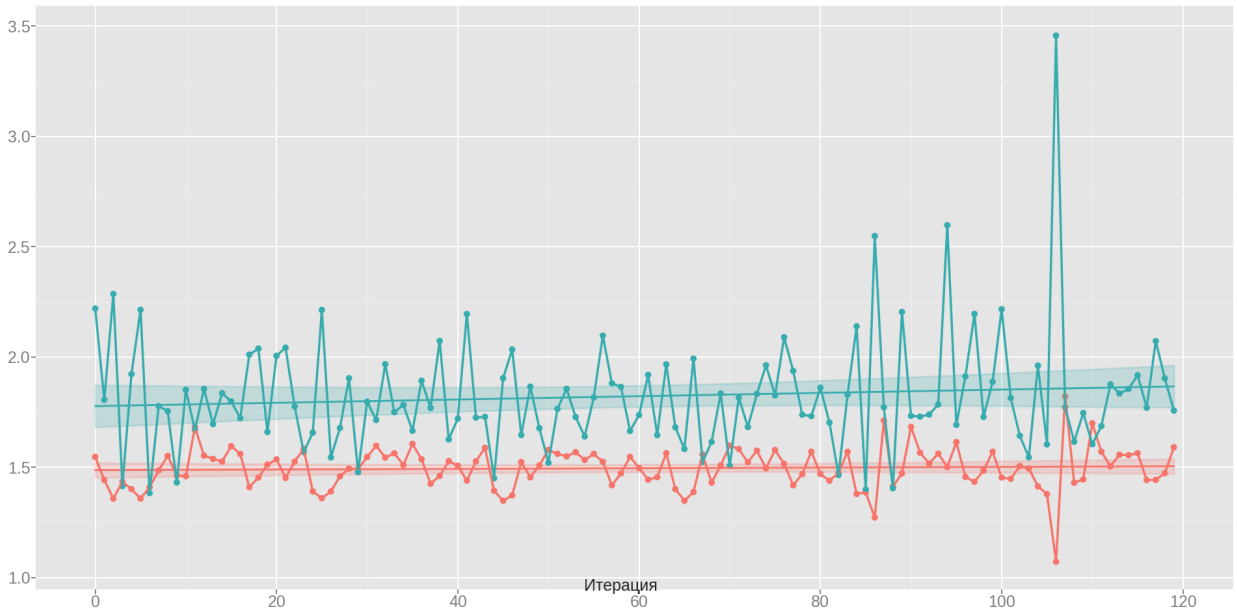


Рис. 8: XGBoost. Изменение точности классификации на протяжении обучения. Красный — тренировочная выборка, зеленый — тестовая.

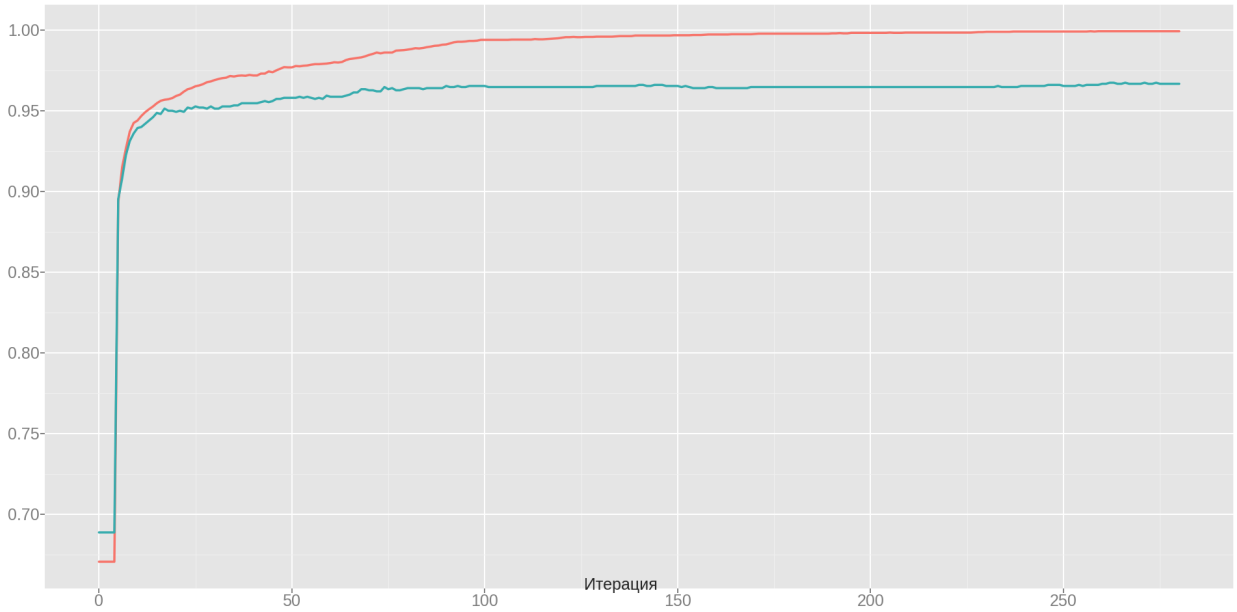
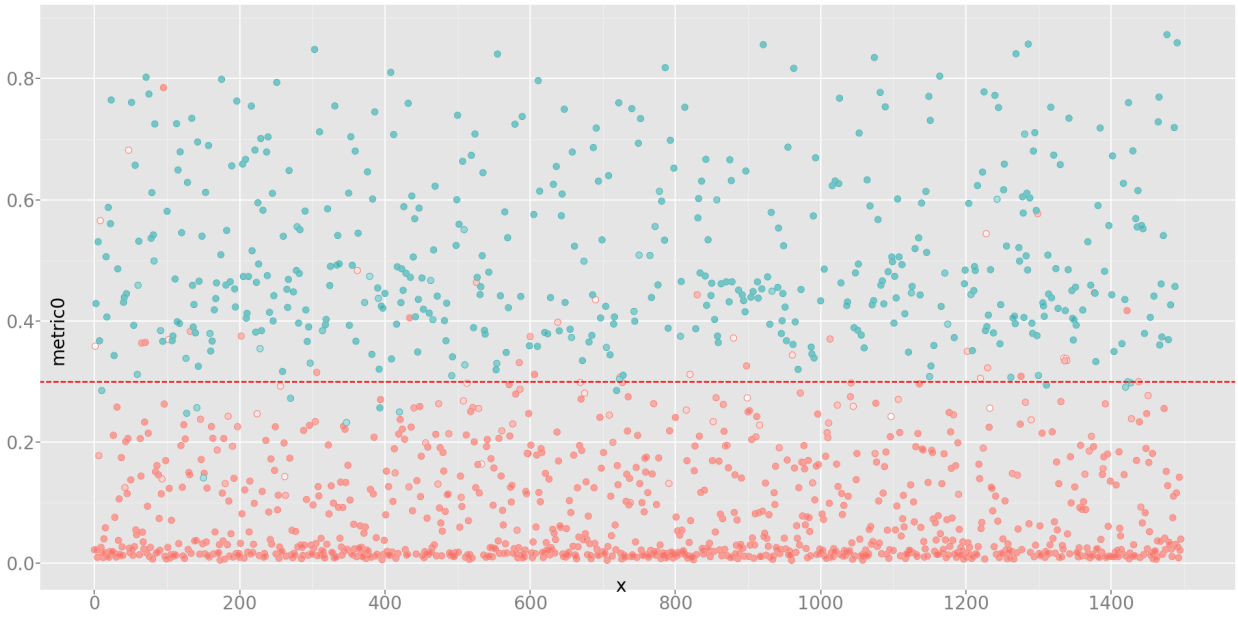


Рис. 9: XGBoost. Прогноз модели на случайной тестовой выборке. Цветом обозначен прогноз модели: красный — плохо кэшируемый блок, зеленый — хорошо кэшируемый. Степень насыщенности цвета — это вероятность, выведенная моделью. Ось x: номер блока в тестовой выборке, ось y: исходное значение метрики. Пунктирная линия - барьер метрики, по которому блоки были разделены на два класса для обучения модели. Ошибка классификации 3.2%.



Список литературы

[1] Architectural Complexity Measures of Recurrent Neural Networks / Saizheng Zhang, Yuhuai Wu, Tong Che et al. // Advances in Neural Information Processing Systems. — 2016. — P. 1822–1830.

[2] Chen Tianqi, Guestrin Carlos. XGBoost: A Scalable Tree Boosting System // CoRR. — 2016. — Vol. abs/1603.02754. — URL: http://arxiv.org/abs/1603.02754.

[3] Dai Andrew M, Le Quoc V. Semi-supervised Sequence Learning // Advances in Neural Information Processing Systems 28 / Ed. by C. Cortes, N. D. Lawrence, D. D. Lee et al. — Curran Associates, Inc., 2015. — P. 3079–3087. — URL: http://papers.nips.cc/paper/5949-semi-supervised-sequence-learning.pdf.

[4] Du Yong, Wang Wei, Wang Liang. Hierarchical recurrent neural network for skeleton based action recognition // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2015. — P. 1110–1118.

[5] Gal Yarin, Ghahramani Zoubin. A theoretically grounded application of dropout in recurrent neural networks // Advances in Neural Information Processing Systems. — 2016. — P. 1019–1027.

[6] Gers Felix A, Schmidhuber Jürgen, Cummins Fred. Learning to forget: Continual prediction with LSTM // Neural computation. — 2000. — Vol. 12, no. 10. — P. 2451–2471.

[7] Glorot Xavier, Bengio Yoshua. Understanding the difficulty of training deep feedforward neural networks. // Aistats. — Vol. 9. — 2010. — P. 249–256.

[8] Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep Learning. — 2016. — Book in preparation for MIT Press. URL: http://www.deeplearningbook.org.

[9] Graves Alex. Supervised sequence labelling // Supervised Sequence Labelling with Recurrent Neural Networks. — Springer, 2012. — P. 5–13.

[10] Hwang Kyuyeon, Sung Wonyong. Character-Level Language Modeling with Hierarchical Recurrent Neural Networks // arXiv preprint arXiv:1609.03777. — 2016.

[11] Kingma Diederik, Ba Jimmy. Adam: A method for stochastic optimization // arXiv preprint arXiv:1412.6980. — 2014.

[12] Григорий Киргизов. Использование алгоритмов машинного обучения для определения неэффективно кэшируемой нагрузки. — 2016. — СПбГУ, курсовая работа (3 курс, 5 семестр).