Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук
Основная образовательная программа
Прикладная математика и информатика

КУРСОВАЯ РАБОТА

Исследовательский проект на тему

"Индуктивное обучение графовых моделей для задачи рекомендательных систем"

Выполнил студент группы 192, 3 курса,
Акулов Дмитрий Александрович

Руководитель КР:
приглашенный преподаватель Ананьева Марина Евгеньевна

Москва 2022

Содержание

1 Аннотация

Рекомендательные системы используются во множестве задач, но все еще имеется множество ограничений по их специфике. В данной работе исследуются подходы по обобщению области применения популярных алгоритмов на графах с целью улучшения их качества предсказаний и производительности.
Ключевые слова: Рекомендательные системы, Глубинное обучение, Теория графов

Recommender systems are used in a variety of tasks, but there are still many limitations on their specifics. This paper explores approaches to generalize the field of application of popular algorithms on graphs in order to improve their prediction quality and performance.
Keywords: Recommender Systems, Deep Learning, Graph Theory

2 Введение

2.1 Рекомендательные системы

Рекомендательные системы - одна из самых популярных областей машинного обучения, применимая как для стандартных задач рекомендаций объектов по истории взаимодействий пользователей с ними и друг с другом, так и, например, для медицинских задач, для сужения области обследования.
Задачи рекомендаций можно решать методами из линейной алгебры, используя матричные разложения и с помощью глубинного обучения, который и будет использован в данной работе.

2.2 Холодный старт

Проблема холодного старта - одна из основных в задачах рекомендаций. Мы мало что можем сказать о объектах, которые появились недавно, поэтому

для них мы будем иметь плохое качество. Хочется использовать альтернативные источники данных, для улучшения работы с такими объектами. Одно из желаний - уметь применять модели, обученные на одних данных к другим. Это часто нельзя сделать напрямую, потому что алгоритмы опираются на специфику объектов. Одна из задач - попытаться обобщить область определения модели.
Особенно эта проблема релевантна для графов, так как множество вершин на этапах обучения и применения могут значительно отличаться.

2.3 Операция свертки

Свертка - операция, агрегирующая информацию в объекте по некоторым другим, которые считаются связанными с ним. Например, соседние пиксели в изображении или близлежащие вершины в графе.
На графах свертка применяется в несколько слоев, чтобы обновлять скрытое векторное представление для объектов нашего графа.

2.4 Индуктивное обучение

Индуктивное обучение - способ бороться с проблемой холодного старта. В задачах на графах мы часто получаем векторное представление для вершин выполняя тяжеловесные сверточные алгоритмы, которые занимают много времени хотя бы потому, что они задействуют весь граф, размеры которых бывают очень большими. Когда у нас появляются новые вершины и ребра, мы не можем себе снова позволить запустить такой алгоритм, чтобы пересчитать значения векторных представлений.
Идея индуктивного обучения заключается в том, чтобы выучивать не конкретные векторные представления для вершин/ребер графа, с которым идет работа, а относительно легковесную функцию их получения, которая могла бы быть потом применена к новым объектам, которые появляются в нашем графе.

3 Обзор литературы

3.1 Архитектура Graph Convolutional Networks

GCN(Graph Convolutional Networks) - сверточная сеть на графах, обучаемая для задачи классификации, введенная в статье Kipf et al. (2016). Представление вершин в скрытых слоях сети содержит как информацию, находящуюся в самой вершине(например, некоторый текст), так и структуру окрестности в графе.

3.2 Архитектура GraphSage

В статье Hamilton et al. (2017) улучшают идею из статьи Kipf et al. (2016) про GCN, применяя индуктивное обучение, позволяя генерировать векторное представление даже для ранее невиданных данных. Ключевая идея работы - обучать функцию агрегации информации из вершин в окрестности нужной, вместо того, чтобы использовать заранее определенные функции и обучать классификацию на полученных векторных представлениях.

3.3 Архитектура PinSage

В статье Ying et al. (2018) описывается архитектура PinSage, применяемая к двудольному графу предметы-коллекции с $3 \cdot 10^9$ вершин и $18 \cdot 10^9$ ребер. Прошлые архитектуры были достаточно тяжеловесными и не справились бы с большими объемами данных, что и пытались решить в этой работе.
Нововведение заключается в определении окрестности вершины, по которой будет вестись агрегация информации. Вместо честного использования части вершин, находящихся не далее некоторого фиксированного расстояния, авторы используют технику случайных блужданий, что позволяет им выделить ключевые вершины, которые и будут определены как ближайшая окрестность.

3.4 Архитектура LightGCN

В статье He et al. (2020) описывается архитектура LightGCN. Авторы обратили внимание, что оптимальный подход в задаче коллаборативной фильтрации, использующий GCN, зазря использует нелинейности, так как это лишено смысла. Они упростили процесс свертки, оставив от GCN только агрегацию, после чего задача практически свелась к матричному перемножению. Тем самым они сделали модель более легкой и повысили качество.
В качестве потенциальных дальнейших доработок авторы оставили вопросы:

- С какими весами усреднять скрытые представления для получения итогового векторного представления.
- Применение негативного и состязательного отбора для процесса оптимизации.

Отличие этой архитектуры от рассмотренных ранее в том, что она никак не использует контентные признаки, а лишь агрегирует обученные на изначальном слое векторные представления, используя граф взаимодействий.
Эту архитектуру можно использовать как некоторый критерий при проверке гипотез, которые будут выдвинуты для проверки работы PinSage без контентных признаков.

3.5 Обзор области и направления развития

В свежей статье Wang et al. (2021) делается обзор на всю область задач рекомендательных систем на графах, и обозначаются тонкие места текущих походов, в направлении которых возможны улучшения.
Один из вариантов улучшения качества - применения внешних знаний. В открытом доступе имеется большое количество данных об отношениях разных типов объектов. Например, продукты выстроены в иерархическую систему, что можно использовать как признаки для наших объектов.

4 План работы

В работе стоит задача изучить архитектуру PinSage, попытаться обобщить ее на больший объем данных(например, содержащие только историю взаимодействия объектов, без описания и прочей дополнительной информации) и попытаться улучшить качество ее работы.

4.1 Вопросы для исследования

- Описанные выше архитектуры обычно применяются на графах, где для вершин есть некоторый ассоциированный с ней текст(описание товара, конкретный комментарий в социальной сети и т.п.), что явно помогает качеству модели. На сколько хорошо они будут работать без подобной информации, и как улучшить качество на подобных данных?

- В статьях Hamilton et al. (2017) и Ying et al. (2018) сравнивается качество работы модели для разных функций агрегации. Можно ли применить какие-то новые операции(например механизм внимания)?

- Граф в статье Ying et al. (2018) является двудольным, но это никак не используется. Можно ли как-то использовать структуру графа для улучшения работы моделей?

5 Эксперименты

Оригинальный код архитектуры PinSage, как и набор данных, на котором она была применена, не были выложены в открытый доступ, поэтому работа велась со сторонней реализацией, которая написана на основе библиотеки DGL: Wang et al. (2019). Библиотека DGL создана для задач глубинного обучения на графовых данных, что и является объектом изучения в этой работе.

Для первоначального изучения был выбран набор данных MovieLens-1m. Набор данных является достаточно большим, чтобы для него релевантным

было использование тяжеловесных моделей, и в то же время приемлемым, чтобы его можно было исследовать в не промышленных масштабах.
В используемой архитектуре уже содержался обработчик набора данных MovieLens-1m к требуемому в архитектуре формату представления данных, однако его пришлось доработать, например для того, чтобы корректно использовать некоторый подграф исходного, потому что полный граф был избыточен для изучения на первых этапах работы.
Сам набор данных представляет из себя информацию о том, какие пользователи посмотрели какие фильмы. То есть эта информация может быть натуральным образом представлена в виде двудольного графа. Помимо информации о взаимодействиях, которые будут ребрами между долями графа, имеется дополнительная информация, такая как: жанр фильма, название фильма, возраст пользователя, пол пользователя, оценка, поставленная пользователем фильму.

Модель PinSage генерирует векотрное представление для вершин на основе изначальной имеющейся информации, после чего эти представления обновляются с ходом обучения модели.
Несмотря на то, что PinSage использует индуктивный подход, в указанной реализации используется трасндуктивный вариант. Это было сделано по причине того, что предполагалось его использование на графах, которые по размеру сильно уступают целевому графу, под который разрабатывалась архитектура. Этот факт для нас является приемлемым, поскольку исследование таких данных и предполагается в этой работе.

5.1 Установка экспериментов

Качество моделей будет измеряться метрикой попадания в истинные взаимодействия. В связи со спецификой разбиения данных на обучающую, валидационную и тестовую выборки, эта метрика эквивалентна полноте.
Важным для качества и скорости работы алгоритма оказался выбор разме-

ра пакета при пакетной обработке данных в эпохах. Поэтому каждый запуск производился с размерами пакетов: 32, 128, 1024, 32768. Количество обрабатываемых за эпоху пакетов было выбрано таким, чтобы каждая эпоха занимала одинаковое количество времени: 1.5 минуты.
Требовалось делать выводы о качестве работы за более короткий отрезок времени, поскольку рассматривается большое количество различных предположений и параметров. Поэтому для сравнения было решено отводить одинаковое количество времени на каждый алгоритм.

5.2 Важность имеющихся признаков для модели

Один из вопросов - насколько важны контентные признаки для модели PinSage. Возможно, вычисление оптимальных представлений для вершин может быть произведено и без дополнительной изначальной информации.
Единственный признак, который присутствовал во всех экспериментах - это отметка времени взаимодействия. Его нельзя было удалить, поскольку эта информация важна для разделения данных на обучающую, валидационную и тестовую выборки, а также для алгоритма рекомендации.
Также был отдельно выделен признак оценки фильма пользователем. Это единственный признак, который соответствовал именно ребру, а не вершине. То есть оценка является единственным явным показателем отношения между пользователями и объектами.
Модель была запущена на данных

- со всеми признаками
- без признака "оценка"
- только с признаком "оценка"
- без признаков

Рис. 5.1: Данные со всеми признаками

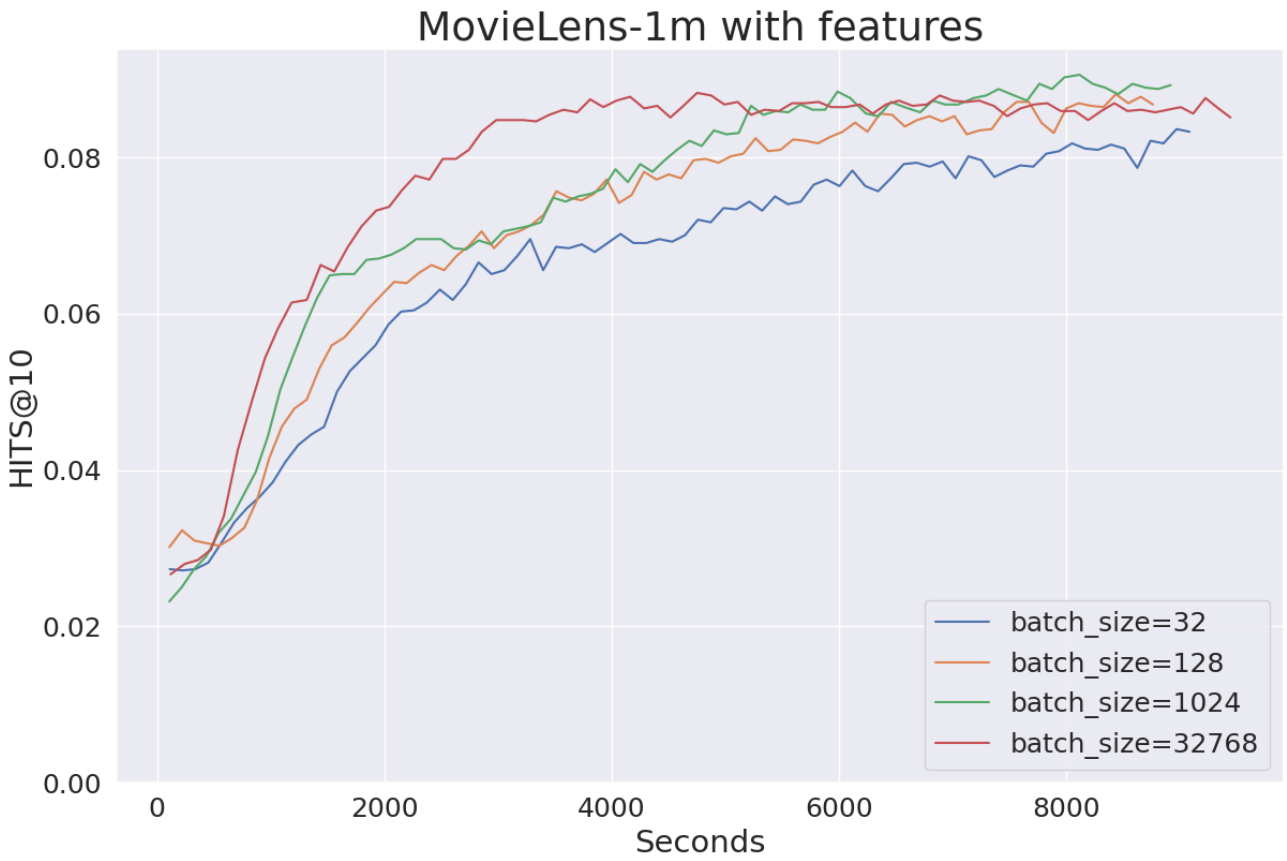
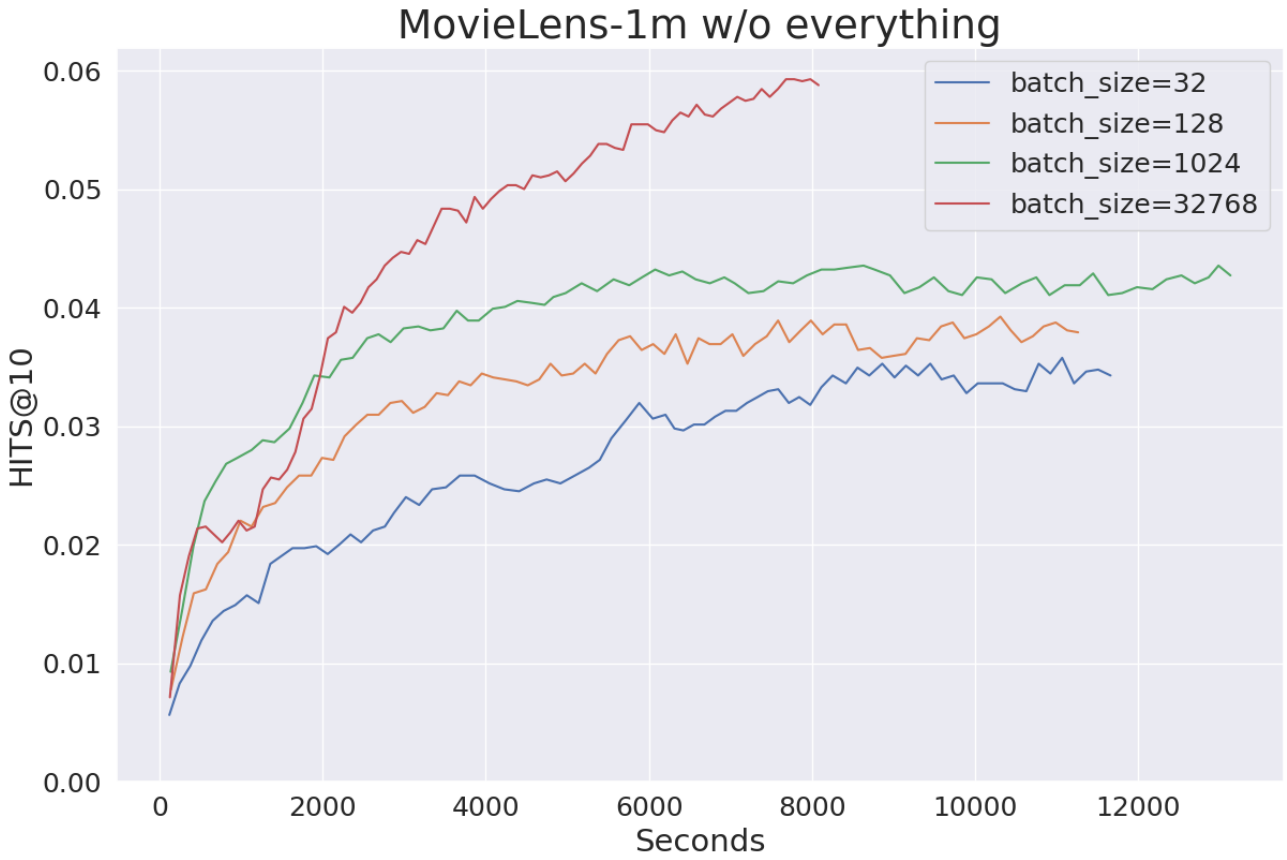


Рис. 5.2: Данные без признаков



На графиках 5.1 и 5.2 показано, как проходило обучение с набором данных со всеми признаками и без всех признаков. Как можно видеть, наличие признаков существенно влияет на качество работы модели.
Здесь алгоритмы не доведены до сходимости. Но уже можно сделать вывод, что признаки важны по крайней мере для качества работы на более коротких отрезках времени.

Обратим внимание, что для обоих наборов данных более удачными оказались большие размеры пакетов. Лучшее качество показано на пакете размера 32768. Поскольку в графе всего 10000 вершин, в этом случае стохастическая оптимизация, по сути, практически превращается градиентный спуск. Это в итоге приведет к тому, что алгоритмы будут сильно переобучаться, но при этом качество оказывается сильно выше.

Форсированное добавление/удаление признака "оценка" не повлияло на процесс обучения. Графики остались практически идентичны, их можно найти в приложении(7.1, 7.2).

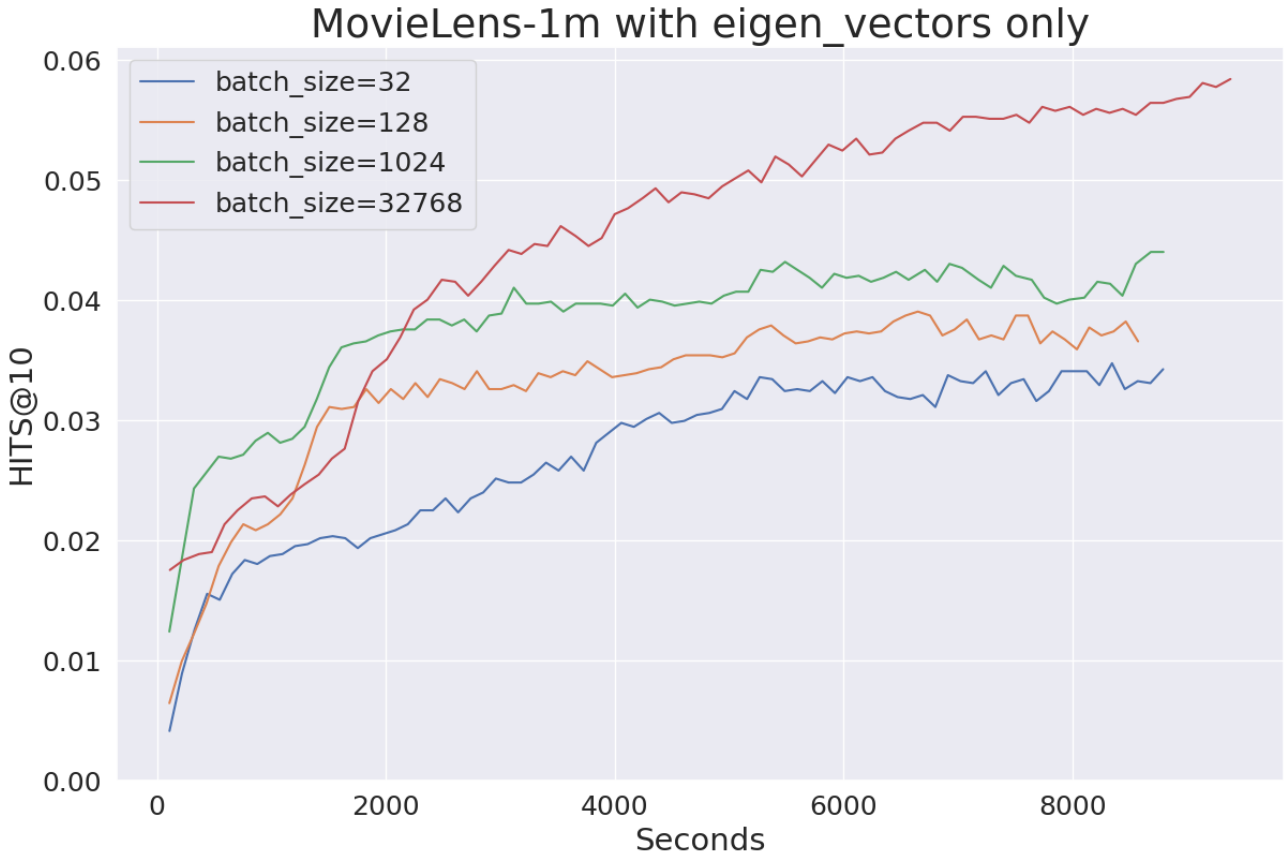
5.3 Вектор взаимодействия в качестве признаков

Однако, даже если у нас нет какой-либо информации о вершинах и ребрах графа, мы можем сами ее придумать, достав какую-либо информацию из структуры графа.
Для каждой вершины мы можем сопоставить ей некоторое векторное представление. Первым очевидным решением является вектор из лапласиана графа.
Однако сам лапласиан хранится в разреженном формате, поскольку он не может быть обработан в обычном виде по причине больших затрат по памяти. Поэтому явный вектор взаимодействия мы не можем сопоставить. Но мы можем попробовать получить информацию в сжатом виде.
Например, решением может быть взятие признаков из сингулярного разложения матрицы, которое часто используется с целью понижения размерно-

сти. Однако на текущий момент в стандартных библиотеках нет алгоритмов, которые бы эффективно вычисляли сингулярное разложение именно для разреженных матриц. Поэтому придется использовать другие подходы.

Ниже, в разделе следующего эксперимента, для каждой вершины вычисляются векторные представления на основе спектрального разложения. То есть мы преобразуем каждый вектор взаимодействия в некоторое сжатое представление. Как можно видеть на графике 5.3, качество такое же как и для запуска без этих данных (график 5.2).

Рис. 5.3: Данные с собственными векторами



5.4 Графовые метрики в качестве признаков

Графы - достаточно хорошо изученная структура. Теория графов может помочь извлечь какую-то информацию для вершин, исходя просто из структуры самого графа.

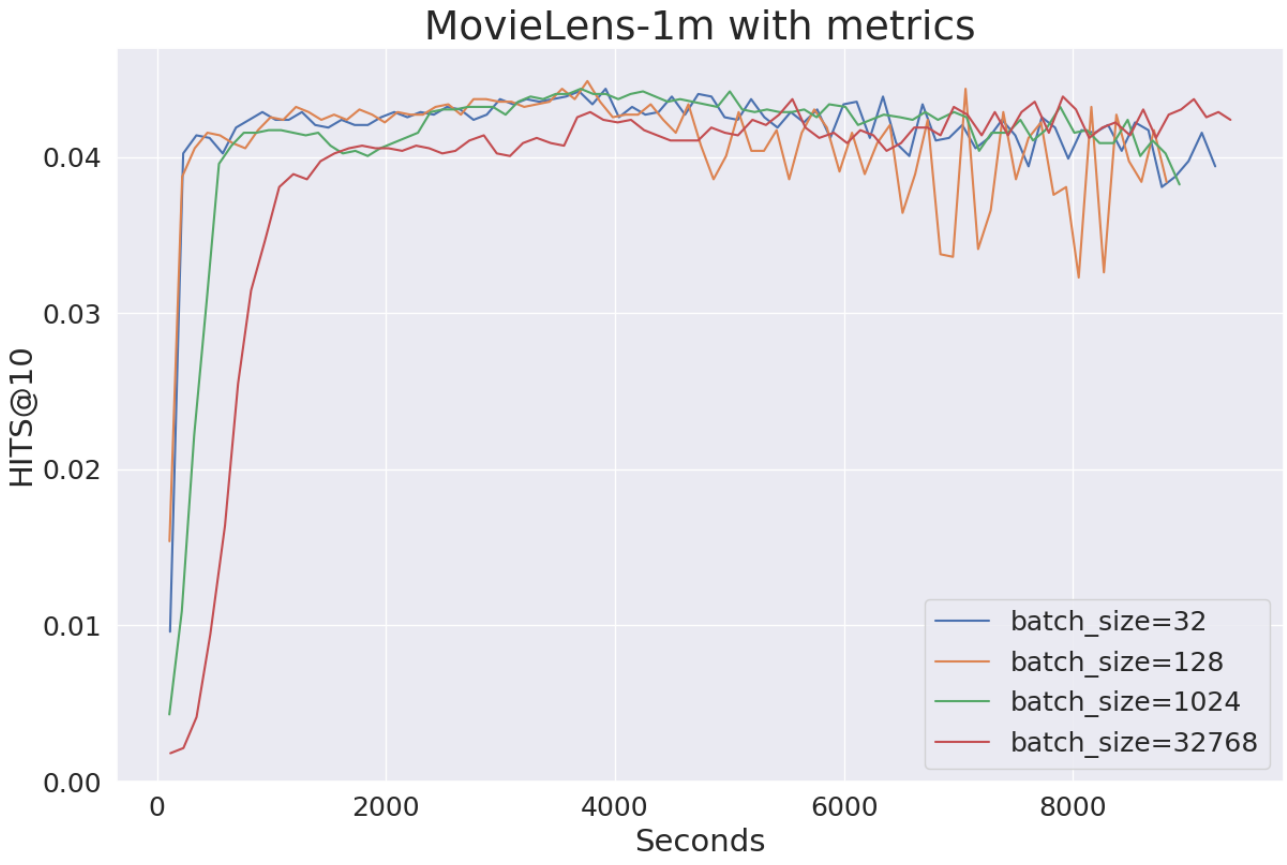
Для каждой вершины были вычислены следующие метрики:

- BFS-метрики: Количество вершин на каждом из нескольких первых слоев поиска в ширину.
- Близость: сумма обратных расстояний до других вершин.
- Промежуточность: количество путей в графе, для которых вершина является промежуточной.
- Крайние собственные вектора из спектрального разложения лапласиана.

Данные метрики были созданы для описания свойств самой вершины или ее окрестности, то есть они могут быть использованы для кластеризации или ранжирования вершин. Поэтому можно предположить, что они будут полезны для процесса обучения. Но также можно выдвинуть гипотезу, что архитектура сама сможет выучить подобную информацию о структуре.

Как видно из графика 5.4, на старте качество сильно лучше, чем обучение

Рис. 5.4: Данные с метриками



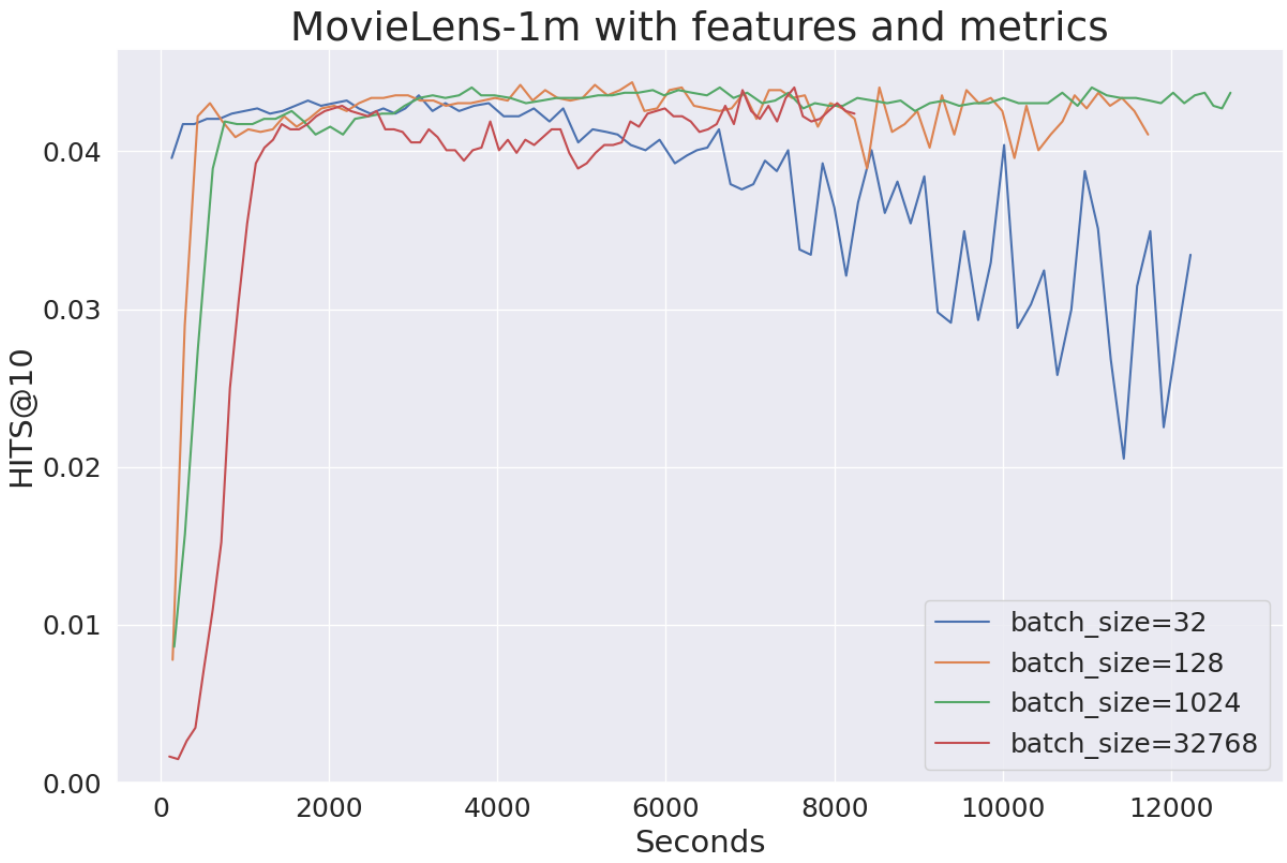
совсем без признаков(график 5.2). Но, при увеличении количества эпох, оно не растет, то есть заменить признаки на метрики не получилось.

Как можно видеть, при маленьких размерах пакета идеальное качество достигается сразу, то есть с очень коротким обучением. Можно предположить, что оптимизатор справляется для каждого объекта найти некоторое хорошее представление на основе простого преобразования метрик.

Была выдвинута гипотеза, что тем самым получается выучить какое-то простое правило, например рекомендацию самого популярного объекта. Из экспериментов ниже можно будет увидеть, что рекомендация по популярности как раз таки дает такое качество.

Интересно отметить, что здесь себя как раз таки хорошо показали маленькие пакеты на контрасте с результатом с полноценными данными.

Рис. 5.5: Данные с признаками и метриками



Главный вывод получается из графика 5.5. Добавление метрик вдобавок к изначальным признакам мешает модели обучаться. Качество застревает на уровне, которое достигается исключительно метриками(график 5.4).

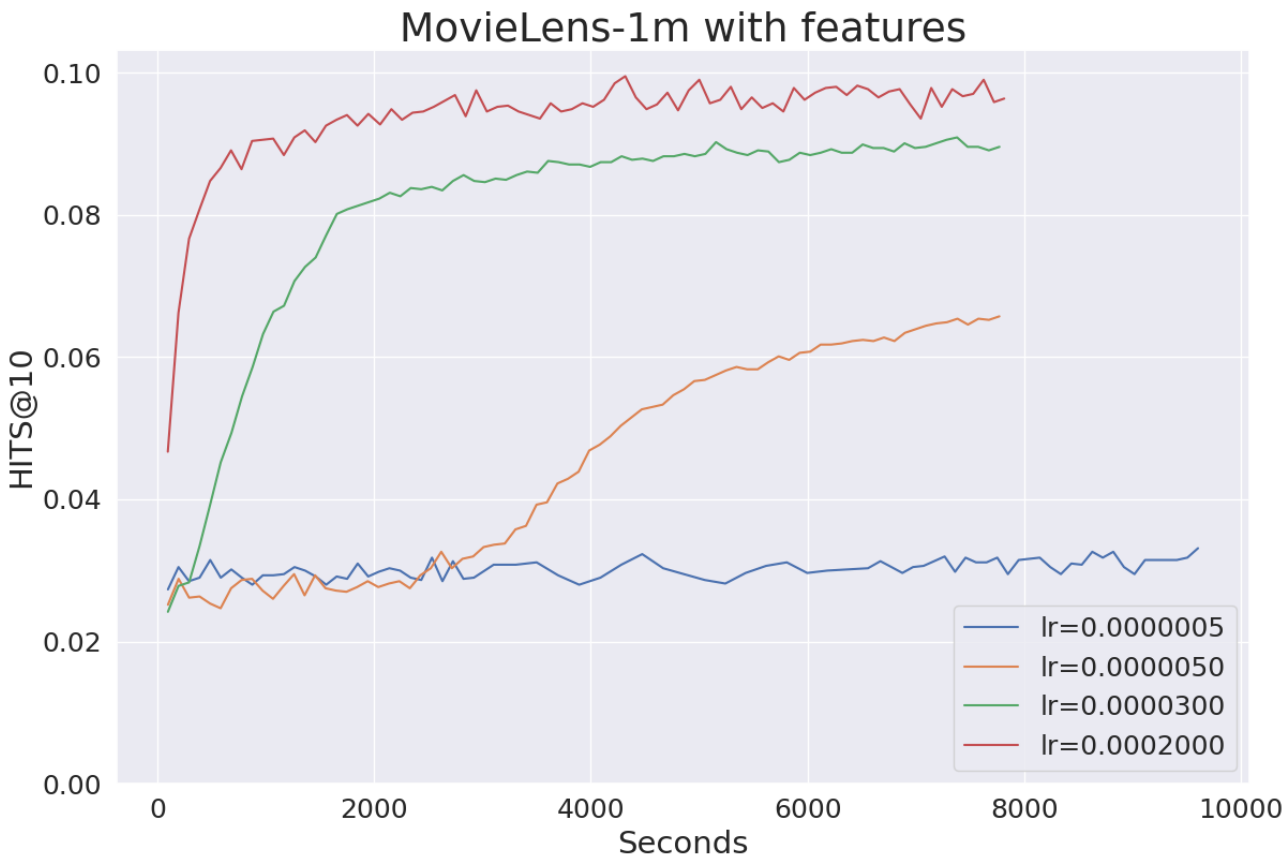
В приложении можно найти графики с более подробным разбиением на используемые метрики.

Быстрая сходимость обеспечивается именно BFS-метриками(график 7.3), остальные оказываются бессмысленными для модели, они не влияют на процесс обучения.

Странным образом себя ведет только метрика близости(график 7.4), с ней модель очень долго показывает нулевое качество, но потом быстро дообучается до таких же значений, как и у остальных наборов.

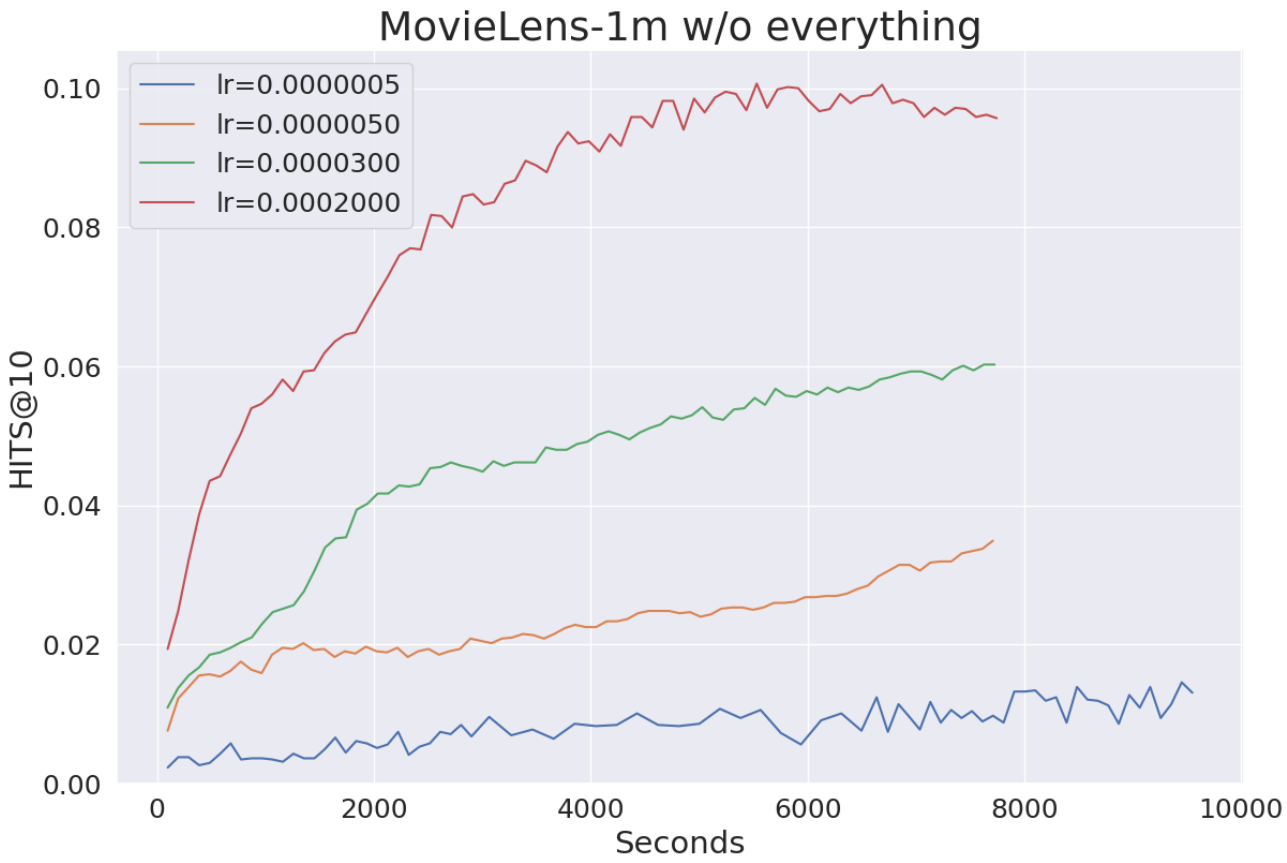
5.5 Подбор скорости сходимости

Рис. 5.6: Скорость обучения для данных с признаками



Из проведенных выше экспериментов было сделано предположение, что в данном случае уменьшение размера пакета эквивалентно уменьшению скорости обучения. Поэтому был проведен эксперимент, где был зафиксирован наиболее успешный размер пакета 32768, и перебиралась скорость обучения.

Рис. 5.7: Скорость обучения для данных без признаков



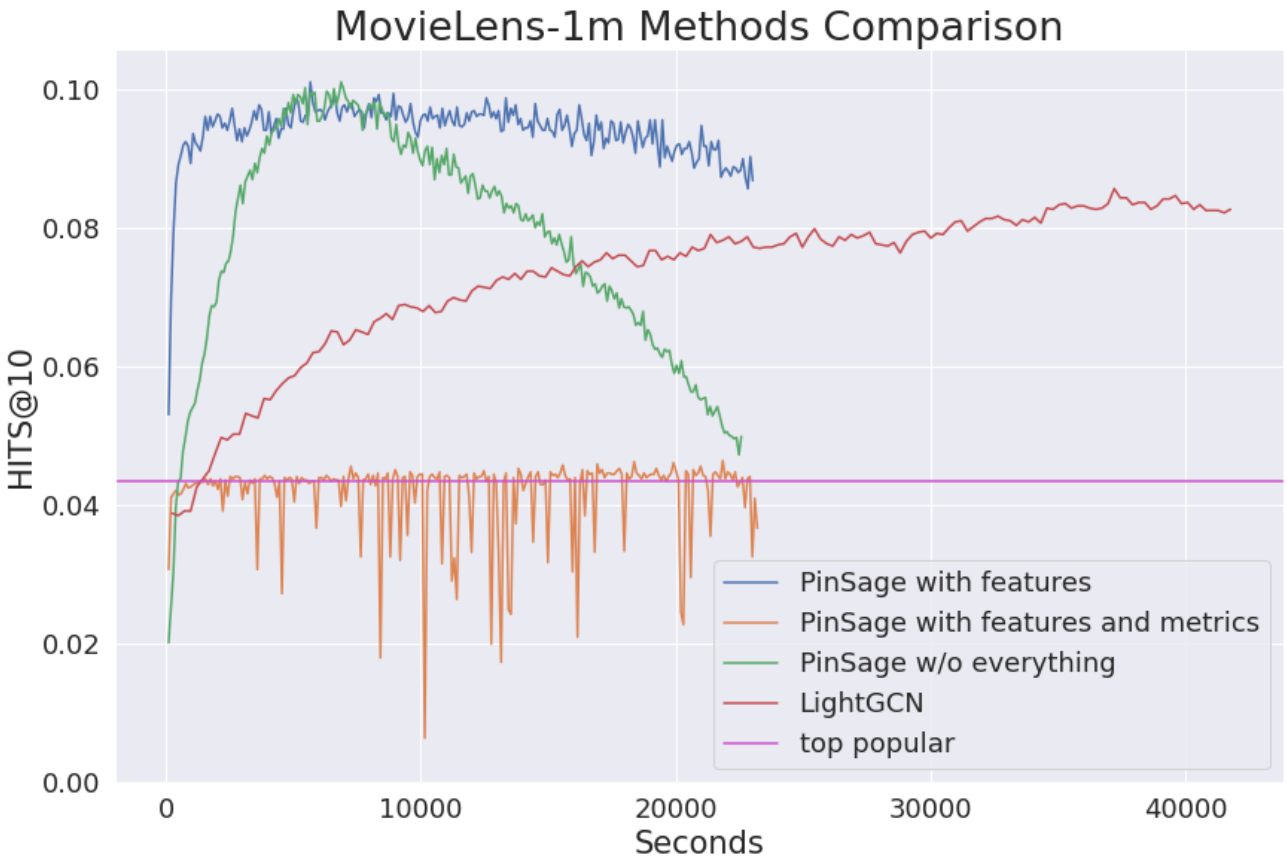
Графики 5.6 и 5.7 показывают, что при изменении скорости обучения поведение отличается от того, что мы могли видеть раньше.

5.6 Сравнение с LightGCN

Поскольку мы позволили себе избавиться от всех признаков, кроме самой структуры графа, то уместно было бы сравнить работу с какой-нибудь архитектурой, которая по сути своей работает только с самим представлением графа. Такой архитектурой как раз является LightGCN.
Также в качестве базовой отметки возьмем качество модели, выдающей константные рекомендации по популярности объектов в обучающей выборке.
Сравним лучшие методы, полученные выше, с этими.

На графике 5.8 можем видеть, что модель PinSage, к которой добавлены графовые метрики в качестве признаков, выдает качество в точности эквивалентное рекомендациям по популярности. Модель быстро сходится к такому

Рис. 5.8: Сравнение с другими методами



качеству и не справляется побить эту отметку.

Здесь мы можем видеть, что модель без признаков показыавет такое же качество, как модель, обученная на всех изначальных признаках, но у нее хуже качество на ранних эпохах, и сильнее переобучение на поздних.

LightGCN обучается медленнее и показывает более плохое качество. Он проигрывает в том числе запуску PinSage без признаков. Он обучался в 4 раза дольше указанного на графике отрезка, но качества лучше добиться не смог.

5.7 Представление для вершин пользователей

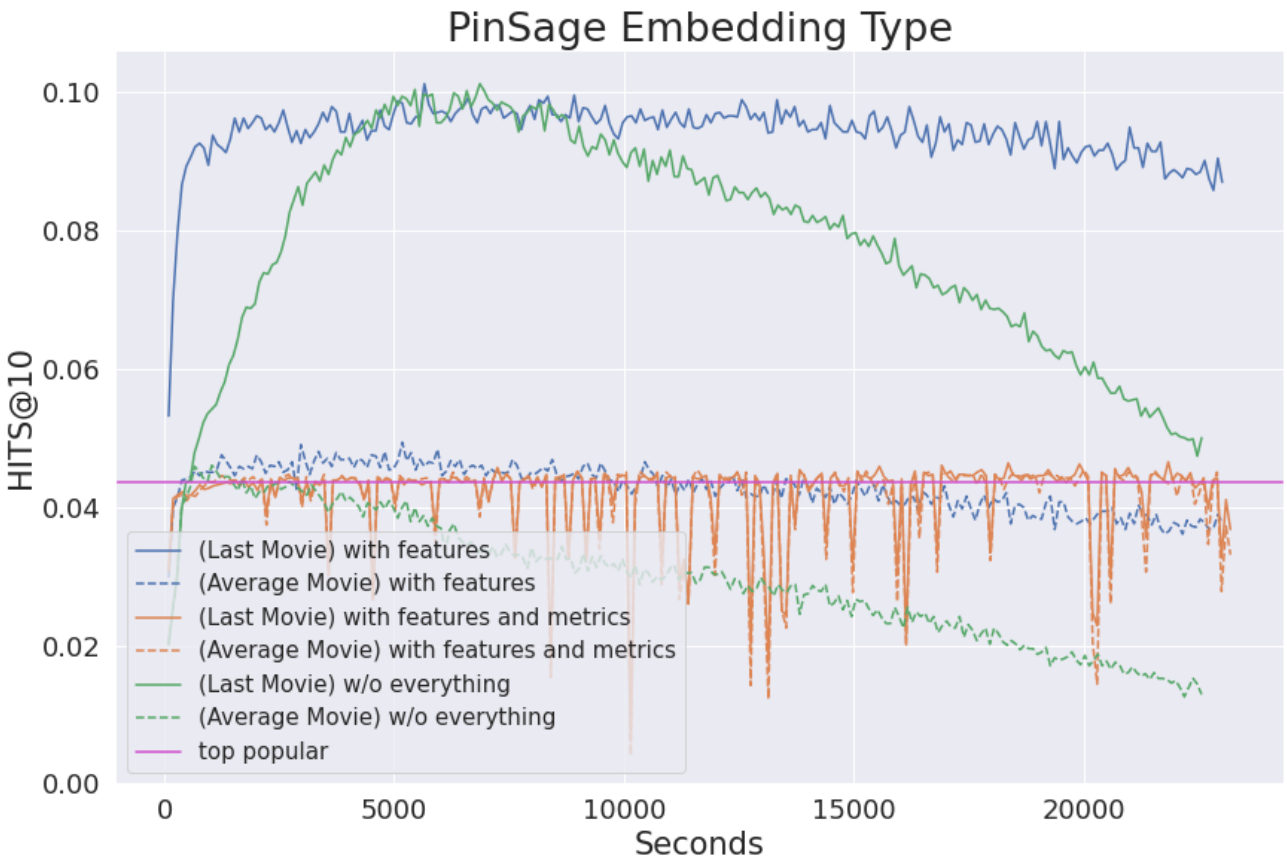
Архитектура работает с графами, в которых явно должны быть выделены две доли: пользователи и объекты. Ее особенностью является то, что она строит и обучает представление только для вершин объектов.

Когда требуется порекомендовать пользователю объект, то его представле-

нию назначается представление последнего объекта из обучающей выборки, с которым он взаимодействовал. Это кажется странным по двум причинам:

- Мы требуем от данных иметь временную отметку для взаимодействий. То есть мы не можем запустить архитектуру на графе без такой информации
- По сути, поскольку последний объект, с которым взаимодействовал пользователь, фиксирован, то у нас задача рекомендации пользователю объекта сводится к рекомендации объекту объекта. Это, например, означает, что многие пользователи с точки зрения рекомендаций становятся полностью идентичны, если их последнее взаимодействие совпадало. Пользователи, тем самым, используются только в процессе случайных блужданий

Рис. 5.9: Сравнение векторных представлений



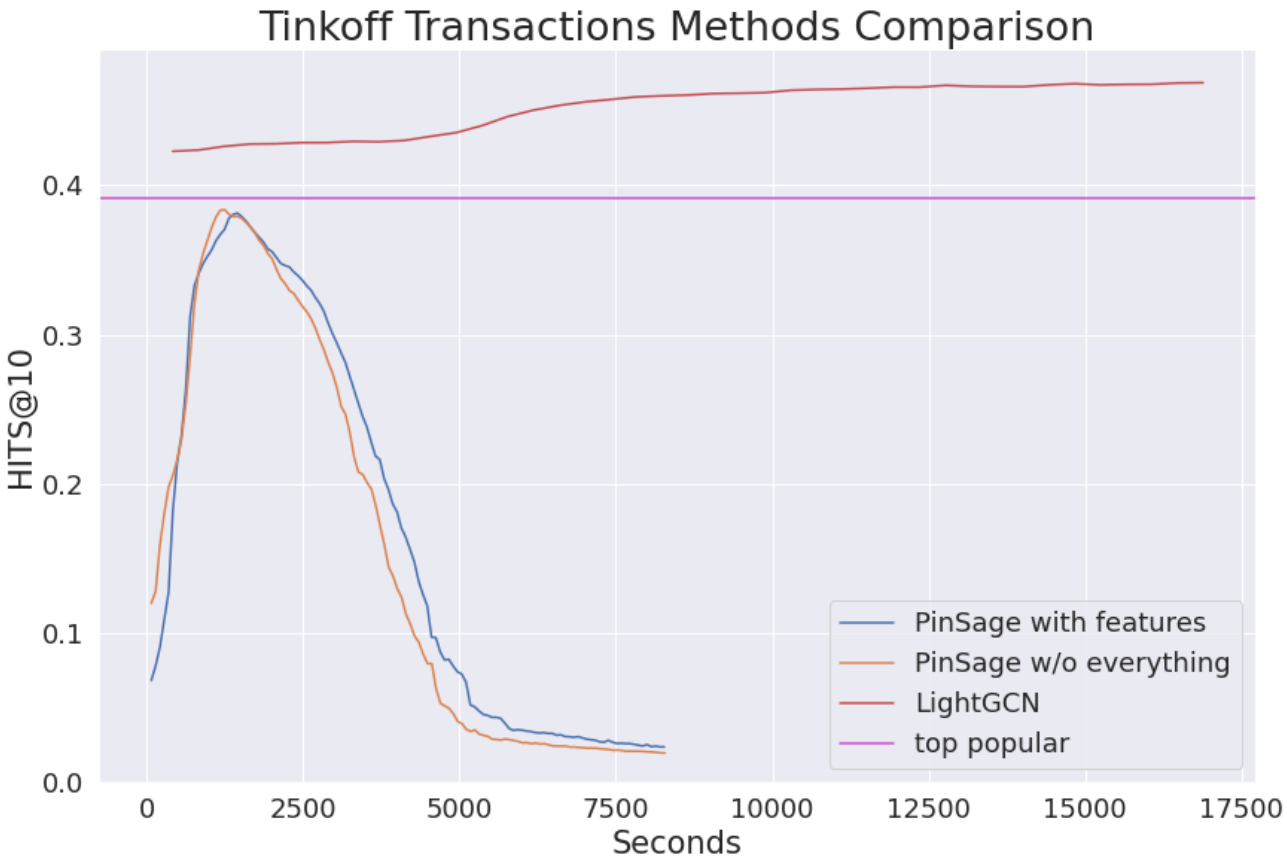
В качестве простого решения данной проблемы было выбрано усреднение представлений по всем объектам, с которыми было взаимодействие.

На графике 5.9 можно на примере трех запусков увидеть сравнение использования векторного представления по последнему фильму(сплошная линия) и усредненное представление(пунктирная линия).
Во всех запусках процесс обучения оставался неизмененным, с усредненным представление запускался только процесс оценки.
Справедливо, что качество оказалось лучше для представления по последнему фильму.
Для двух запусков, показавших лучшие результаты, усредненные представления достигают только отметки тривиального решения, при этом повторяя тренд с переобучением.
В то же время, усредненное представление для запуска с признаками и метриками, где достигается только тривиальное качество, оба представления ведут себя идентичным образом. То есть модель перестает привязываться к последнему фильму, даже несмотря на то, что это явно происходит при вычислении функции потерь.

5.8 Набор данных Tinkoff Transactions

Также стоит проверить все выводы и предположения, что были сделаны выше, на каких-нибудь других данных.
Был выбран набор данных Tinkoff Transactions, Kolesnikov et al. (2021). Набор содержит в 10 раз больше ребер, чем прошлый, но после потребовавшейся обработки их количество уменьшилось до старого значения. Обработка потребовалась, чтобы привести данные к похожему на старый формат. Рассматриваемые подходы не предполагали наличие дублирующихся ребер, почему было решено оставить только последнее взаимодействие.
Также не были использованы текстовые данные.
Модель была запущена с параметрами, которые оказались оптимальными ранее.

Рис. 5.10: Набор данных TTRS



Как можно видеть на графике 5.10, здесь не получилось сильно преодолеть тривиальный порог.

PinSage добрался до близких значений, но потом быстро снова начал переобучаться.

LightGCN на первой же эпохе преодолел порог тривиального решения, после чего смог получать некоторый медленный прирост.

6 Выводы

Эксперименты показали, что добавление признаков может плохо сказаться на качестве модели, если эти признаки обладают некоторыми свойствами. Это может привести к сходимости к некоторому тривиальному решению. В будущем можно попробовать изучить эти свойства и разобраться, что мешает модели выбраться из найденного локального минимума. Предположительно, решением проблемы может быть использование циклической скорости обу-

чения.

Также из экспериментов можно видеть, что дополнительные признаки для вершин и ребер могут быть не критически важны для итогового качества, но скорость сходимости точно увеличивают, если являеются осмысленными.

Модель PinSage обучается быстрее чем LightGCN. Обучение у LightGCN более размеренное, но это позволило ей показать лучшее качество на наборе данных Tinkoff Transactions.

Список литературы (или источников)

1. Thomas N. Kipf, Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. Published as a conference paper at ICLR 2017.

2. William L. Hamilton, Rex Ying, Jure Leskovec. Inductive Representation Learning on Large Graphs. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

3. Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, Jure Leskovec. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. KDD 2018.

4. Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, Meng Wang. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. Accepted by SIGIR.

5. Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z. Sheng, Mehmet A. Orgun, Longbing Cao, Francesco Ricci, Philip S. Yu. Graph Learning based Recommender Systems: A Review. Accepted by IJCAI 2021 Survey Track, copyright is owned to IJCAI.

6. Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, Zheng Zhang. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks.

7. Sergey Kolesnikov, Oleg Lashinin, Michail Pechatov, Alexander Kosov. TTRS: Tinkoff Transactions Recommender System benchmark.

7 Приложение

Рис. 7.1: Данные с признаком "оценка"

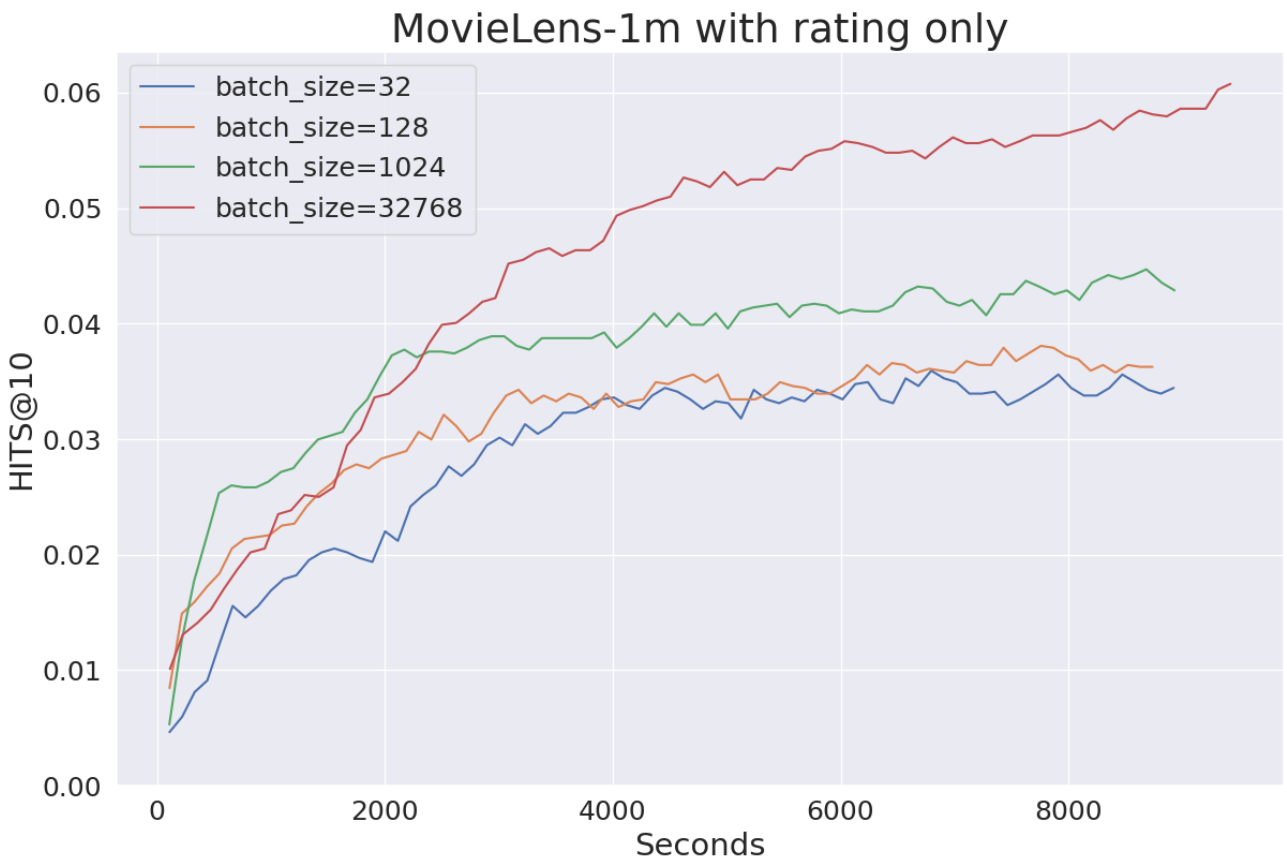


Рис. 7.2: Данные со всеми признаками кроме "оценки"

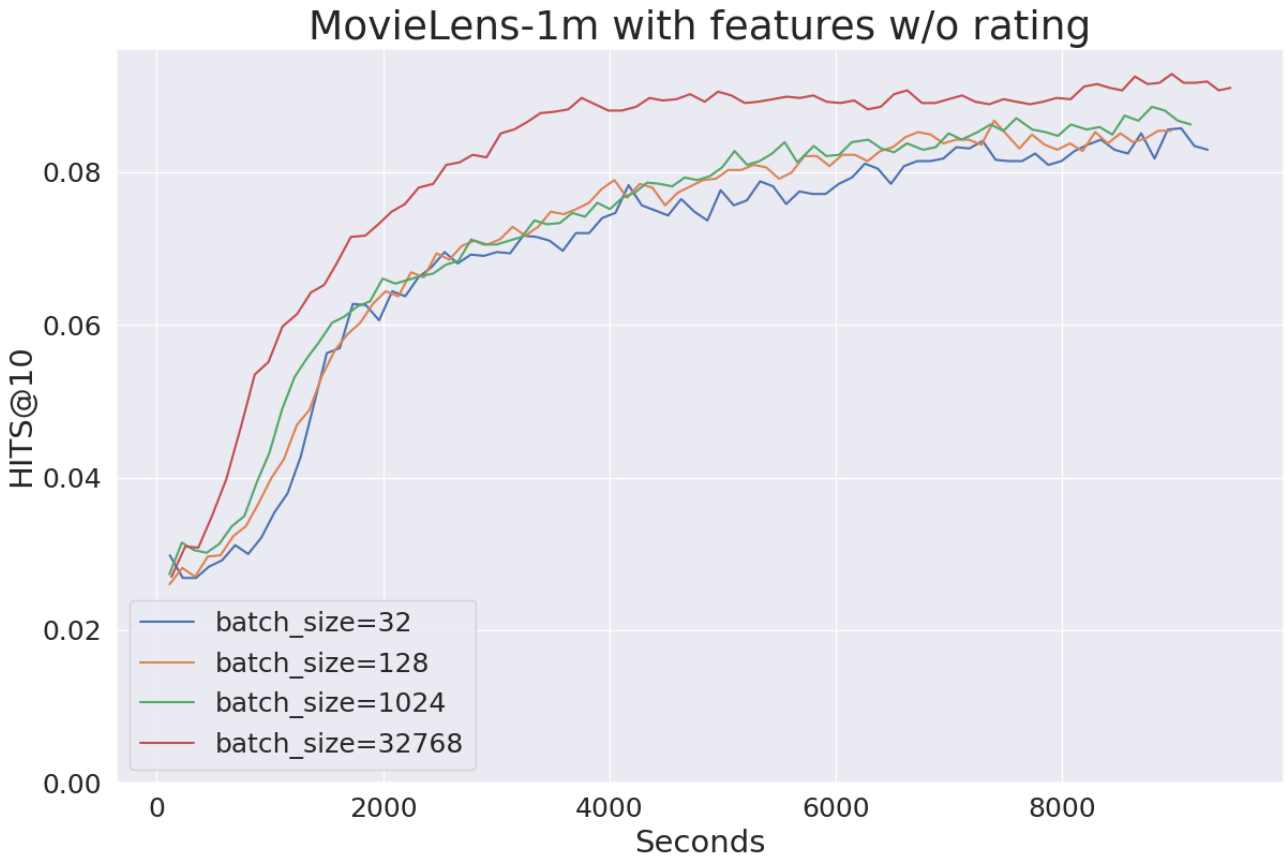


Рис. 7.3: Данные с BFS-метриками

Рис. 7.4: Данные с метрикой близости

Рис. 7.5: Данные с метрикой промежуточности