Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Солодкая Анастасия Сергеевна

«Предсказание предпочтение пользователя с использованием нейросетевого подхода»

Выпускная работа бакалавра

Допущена к защите

Зав. кафедрой:

д.ф.-м.н., проф. А.Н. Терехов

Научный руководитель:

к.ф.-м.н. Д.Ю. Бугайченко

Рецензент:

к.ф.-м.н., доцент И.П. Соловьев

Санкт-Петербург

2012

Saint-Petersburg State University

Mathematics and Mechanics Faculty

Software Engineering Department

Anastasiya Solodkaya

User preferences prediction based on neural networks

Bachelor's thesis

"Approved by"

Head of Department

Professor A. N. Terekhov

Scientific advisor

Ph.D, Dmitry Bugaychenko

Reviewer

Ph.D, associate professor Igor Soloviev

Saint-Petersburg

2012

ОГЛАВЛЕНИЕ

Стр.

1. Введение

В течении последнего десятилетия развитие электронной коммерции, социальных сетей, а так же других сервисов, сделало крайне актуальным вопрос поиска и фильтрации информации. Пользователи имеют доступ к большим массивам информации, и не имеют ни возможности, ни времени, ни желания искать ту информацию, которая могла бы их заинтересовать. К примеру, на заре развития электронных видео-магазинов, пользователь имел доступ к нескольким сотням CD-дисков с фильмами. Среди них он мог выбрать те, которые его интересуют и заказать их. Сейчас в базах подобных магазинов имеются десятки тысяч фильмов в каждой из категорий. Немногие люди готовы потратить время на просмотр всех имеющихся в наличии фильмов. В этом случае выигрывают магазины, которые могут заранее спрогнозировать, какие фильмы могут заинтересовать клиента.

Подобные проблемы встают перед многими сервисами. Музыкальные и видео-базы, социальные сети, новостные ленты, блоги, электронные магазины нуждаются во вспомогательных системах, которые помогли бы на основе имеющихся данных о пользователе, предложить ему то, что могло бы его заинтересовать.

1.1 Рекомендательные системы

Программы, которые пытаются спрогнозировать на основе имеющейся информации о пользователе, какие объекты могли бы быть ему интересны, называются рекомендательными системами[1].

Уже сейчас в интернете можно найти множество сервисов, использую-

щих рекомендательные системы. Пожалуй, самые известные из этих сервисов - это Imhonet (рекомендации контента разного рода: музыка, фильмы, фотографии, литература), интернет-магазины Ozon.ru, Amazon.com, а так же узко-специализированные медиа-проекты Last.fm, Netflix.

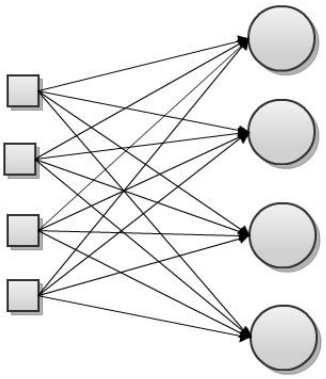
Для некоторых из этих сервисов рекомендательные системы являются основой бизнеса. Для них качество рекомендаций является особенно критичным. Примером такого сервиса является свервис продажи видео-дисков Netflix.

История рекоммендательных систем уходит в середину 1990х годов, когда была опубликована первая работа в этой области. С тех пор рекоммендательные системы набирали популярность среди исследователей в связи со стремительным развитием сети Интернет. Одним из крупных прорывов в разработке данной области было исследование, проведенное в рамках соревнования рекомендательных систем Netflix Prize Challenge[2], проходившее с 2006 по 2009 год. Организатором соревнования была компания Netflix. Ими был заявлен приз US$1,000,000 комманде, которая предоставит рекомендательную систему, обрабатывающую существующую базу данных в 100 миллионов фильмов и вернет рекоммендации на 10% точнее, чем существующая в компании система рекоммендаций. 21 сентября 2009 года приз был вручен команде AT&T “BellKor”. Эта комманда обнаружила, что рейтинги, которые пользователи выставляют фильмам, просмотренным давно, отличаются от тех, которые выставлены сразу после просмотра. Так же на поставленный рейтинг влияет настроение, и даже день недели: рейтинги, выставленные в пятницу, сильно отличаются от рейтингов, выставленных в понедельник.[3]

В данной работе рекомендательные системы рассматриваются на примере музыкального сервиса. Одной из самых известных музыкальных баз является база проекта last.fm. Этот проект имеет свою систему музыкальных рекомендаций. В последнее время разрабатываются музыкальные рекомендации в социальных сетях (например, в популярной российской социальной сети vk.com)

1.2 Нейронные сети

В этой работе задача рекоммендаций решается с помощью применения нейронных сетей. В общем случае нейронная сеть представляет собой машину, моделирующую способ обработки мозгом конкретной задачи[4]. Нейроны - элементарные ячейки вычислений.



Можно определить нейронную сеть как громадный распределенный параллельный процессор , состоящий из элементарных единиц обработки информации, накапливающих экспериментальные знания и предоставляющих их для последующей обработки. Нейронная сеть сходна с мозгом с двух точек зрения:

- Знания поступают в нейронную сеть из окружающей среды и используются в процессе обучения.
- Для накоплений знаний применяются связи между нейронами, называемые синаптическими весами.

Настрока нейронной сети традиционно происходит с помощью изменений синаптических весов.

Основные преимущества нейронных сетей заключаются в параллельности обработки информации и способностью к обобщениям (обучению). Благодаря этому нейронные сети позволяют решать трудноразрешимые масштабные задачи. Кроме того, использование такого подхода обеспечивает многие полезные свойства систем, среди которых адаптивность, контекстность существования нейронной сети, отказоустойчивость, масштабируемость, единообразие анализа и проектирования, а так же аналогия с нейробиологией.

Первые работы в области исследований нейросетей были предоставлены в 1943 году Мак-Каллоком и Питцом[5]. Область активно развивалась вплоть до 1969 года, когда Минский и Пейперт [6] выпустили книгу, в кото-

рой были обоснованы ограничения однослойного персептрона и утверждалось, что ограничения вряд ли удастся преодолеть в многослойных персептронах. Кроме психологической проблемы, возникшей из-за этой работы, была так же техническая: довольно трудно было провести эксперимент.

В начале 1980х годов интерес к нейросетям возобновился и к текущему времени нейросети имеют уже довольно широкую область применения. Это прогнозирование, кластеризация, распознавание образов, принятие решений, визуалиация и так далее. Одним из примеров может служить предсказание курсов валют, основанное на нейросетях. Подобную функциональность можно найти в продуктах компании TradingSolutions. Другой пример - фильтрация спама на основе нейросетей, а так же принятие решений о выдачи банком кредита клиенту (определение платежнеспособности клиента).

1.3 Уменьшение размерности данных

Одной из главных проблем при использовании нейронных сетей является то, что большие наборы данных нейронные сети могут обрабатывать медленно и неэффективно. С целью уменьшения количества обрабатываемых данных, используются алгоритмы, называемые алгоритмами для уменьшения размерности данных.

Уменьшение размерности данных - это процесс уменьшения количества рассматриваемых свободных переменных. Техника уменьшения размерности используется в разных сферах. Она является широко используемой в областях статистического обучения, визуализации данных, анализе данных и кластеризации. Одним из самых известных способов применения этой техники является проекция многомерных наборов данных на два измерения для последующей визуализации.

В общем случае необходимо уменьшать размерность данных ввиду больших затрат на обработку многомерных массивов данных.

1.4 Проблема рекоммендаций

В рамках этой работы проводится исследование одного из довольно плохо изученных подходов к построению рекоммендательных систем: построение рекоммендательной системы на основе нейронной сети. Актуальность проблемы рекоммендаций растет с каждым годом. Рекоммендательная система является одной из весьма важных частей любого крупного сервиса электронного бизнеса. Она помогает повысить качество обслуживания, удовлетворить потребность пользователей и увеличить продажи. Вкратце можно сформулировать функции, выполняемые рекоммендательной системой:

а) Увеличение продаж. Простейший пример: при покупке сотового телефона можно порекомендовать аксессуары к нему

б) Уменьшение количества информации, с которой пользователь должен ознакомиться, прежде, чем сделать выбор. К примеру, вместо просмотра огромного каталога с музыкой, пользователь может прослушать композиции, которые рекомендованы лично ему.

в) Увеличение удовлетворенности и лояльности пользователей.

г) Помощь в анализе пользовательских предпочтний, потребностей, а так же тенденций и трендов.

2. Постановка задачи

В рамках данной работы была поставлена задача предсказания предпочтений пользователей на основе нейросетевого подхода для систем музыкальных рекомендаций. Для реализации задачи были выделены следующие подзадачи:

- Реализация нейронной сети для разделения пользователей на группы по предпочтениям
- Реализация механизма предсказания рейтингов треков у каждого конкретного пользователя на основе группировки пользователей по предпочтениям
- Реализация подсчета и демонстрации метрик для анализа точности и эффективности работы системы предсказаний рейтингов
- Получение и подготовка данных для проведения эксперимента для проведения экспериментального анализа эффективности работы алгоритма

3. Обзор

3.1 Рекомендательные системы

Рекомендательные системы - это программы, которые на основе известной о пользователе информации пытаются предсказать, какие объекты (товары, новости, книги, музыка, фильмы, услуги) будут ему интересны[4].

Проблема рекомендаций обычно возникает в контектсе объектов определенного рода, оцененых пользователем по некоторой шкале. В большинстве случаев она сводится к проблеме предсказания рейтингов тех объектов, которые еще не были оценены пользователем. Это предсказание осуществляется на основе рейтингов уже оцененных объектов, рейтингов объектов у похожих пользователей, на основе связей между пользователями и информации из их профиля. После того, как предполагаемая оценка рассчитана, выбирается некоторое количество объектов с наивысшим рейтингом и рекомендуется пользователю.

Более формально эту проблему можно выразить следующим образом: допустим U - набор пользователей, I - набор существующих объектов. r - некоторая функция, которая представляет полезность объекта i для пользователя u:

$$r : U \times I \to V$$

V - вполне упорядоченное множество.
Таким образом для каждого пользователя $u \in U$ проблема рекоммендаций состоит в поиске i^*:

$$i^* = \arg\max_{i \in I} r(u, i)$$

В большинстве случаев r представляет собой меру “насколько пользователю нравится этот объект”

Основные подходы

Можно выделить три основных подхода для решения задачи предсказания рейтингов:

а) content-based подход

б) social-based подход

в) collaborative filtering

Остановимся подробнее на этих подходах.

Content-based подход

Контекстно-ориентированный подход предоставляет техники для рекомендации пользователю объектов, похожих на те, которые ему нравились ранее. “Похожесть” объектов рассчитывается на основе описаний этих объектов. Так же рекомендации производятся на основе информации, доступной в профиле пользователя. Один из самых крупных проектов по исследованию и развитию контекстно-ориентированного подхода - это проект FOAF[7] (Friend Of A Friend). По сути это онтология, описывающая пользователей, их связи с другими пользователями и объектами, а так же их активность в сети. Эта онтология является машиночитаемой. Для описания пользователей и их связей используется RDF - модель представления данных и метаданных, а так же OWL - язык описания онтологий в сети. Эта онтология может использоваться для решения задач рекомендации с помощью конntекстно-ориентированного подхода. К примеру, с помощью этой онтологии можно найти всех людей, проживающих в России.

Social-based подход

Следующий подход - социально ориентированный. В рамках социально-ориентированного подхода рекомендации происходят на основе социальных связей и социальной активности пользователей. Этот подход довольно нов,

но с развитием социальных сетей он привлекает все больше и больше исследователей. В современных сетях существует довольно много информации, которую можно было бы использовать в рамках разработки системы рекомендаций: друзья пользователей, люди, которые ходят на одни и те же концерты и прочие события, люди, чатсо общающиеся и состоящие в одних и тех же клубах и так далее.

Collaborative filtering

И, наконец, совместная фильтрация. В отличии от предыдущих подходов, здесь не принимается во внимание информация из профиля пользователя и информация из профиля объекта. Вместо того, чтобы рекомендовать пользователю объекты, похожие на те, которые понравились ему в прошлом, здесь рекомендуются объекты, которые нравятся пользователям, "похожим на него". В основе этого подхода лежит предположение, что пользователям, которые оценили одинаковые объекты похожим образом, другие объекты будут примено так же "нравиться". Таким образом, оценка $R(u.i)$ основывается на оценках $R(U_k, i)$, где $u_k \in U$ - пользователи, "похожие" на пользователя u и оценившие объект i.

В большинстве случаев рекомендательные системы построены не на основе одного из этих подходов, а скорее на основе их смешения. Это так называемые гибридные рекоммендательные системы. Система, разработанная в рамках данной работы, является системой, основанной на подходе совместной фильтрацией. Далее полученная система расширяется с помощью добавления к ней элементов социального подхода. Социальный подход реализован в рамках другой работы.

Итак, в свою очередь, техники, используемые в подходе совместной фильтрации можно разделить на две основные части:

а) Memory-based

б) Model-based

Memory-based

Данная техника (известная как “поиск соседних”) использует набор известных рейтингов R. Сначала применяются различные способы статистической обработки и приближений для определения $N(u)$ пользователей, близких (похожих) к u - “соседей”. Затем рейтинг объекта i у пользователя u определяется следующим образом:

$$R(u,i) = h(\{R(U_k,i) \mid u_k \in N(u)\})$$

h - некоторая агрегирующая функция. Один из главных минусов таких алгоритмов - это высокая чувствительность к разреженности данных. Это проблема, так как зачастую данные действительно весьма сильно разрежены: очень малая часть значений рейтингов известна.

Model-based

Эта группа техник была разработана для решения проблем, возникающих в техниках на основе памяти. В этом случае набор существующих рейтингов исследуется, выясняются общие закономерности и образцы и на основе них производится рекомендация. Недостаток данного подхода - длительное время изучения и анализа данных, длительное обновление.

В данной работе используется техника “поиска соседей”. Среди пользователей выделяются группы похожих и на основе этих групп прогнозируются неизвестные рейтинги.

Основные алгоритмы

Определение рейтингов

Не в каждой системе существует настоящая система рейтингов. Зачастую пользователи не оценивают объекты, но все же объекты упорядочены неко-

торым образом. То есть присутствует нейявное ранжирование. Для решения задачи рекоммендации необходимо определить рейтинг каждого объекта, то есть сделать ранжирование явным. Зачастую для этого используется некоторый набор значений: например от 0 до 1 или от 1 до 5. Таким образом перед нами встает проблема определения рейтинга.

Кроме того существует другая проблема: различные пользователи могут расценивать один и тот же рейтинг различным образом: для кого-то оценка “3 из 5” - это минимум, а для кого-то эта оценка довольно высока.

Так же проблемой является сильная разреженность данных: мы должны каким-то образом учитывать те объекты, для которых еще не выяснены рейтинги (или о которых пользователь еще не знает).

В нашей работе используются данные базы last.fm, то есть нам доступны количества прослушиваний каждого из объектов каждым пользователем (или информация об отсутствии прослушиваний). И для решения всех проблем (неоднородности данных, неявного рейтинга и прочего) используются следующие алгоритмы:

- Рейтинг “по умолчанию”.

  Это некоторый рейтинг, который принимается по умолчанию для тех объектов, для которых рейтинг еще не известен. В большинстве случаев это 0 или значение, чуть меньше 0. В данной работе используется два вида рейтингов по умолчанию:
  - Нулевой рейтинг
  - Средний рейтинг (количество прослушиваний) по всем оцененым пользователем артистам.
- Инверсия рейтингов [8].

  Техника, основанная на предположении, что больше информации можно получить при обработки объектов, которые не были оценены почти никем, чем при обработке объектов, которые были оценены практически всеми. Таким образом, эта техника уменьшает количества шума в данных.

  Для расчета инверсированного рейтинга используется следующая формула:

$$R_{inv}(u,i) = f(i)R(u,i)$$

$$f(i) = \log \frac{|U|}{|U_i|}$$

где $|U|$ - количество всех пользователей в системе, а $|U_i|$ - количество пользователей, оценивших данный объект i.

- Расчет явного рейтинга.

  Расчет явного рейтинга может производиться несколькими способами. Обозначим через $l(u,i)$ - количество прослушиваний артиста i пользователем u. L - множество значений $l(u,i)$ - известные данные о количестве прослушиваний для каждой пары артист-пользователь. В данной работе используются следующие два способа:

  – Масштабирование: $r(u,i) = \frac{l(u,i) - l_{u\_min}}{l_{u\_max} - l_{u\_min}}$, где l_{u_max} и l_{u_min} -это максимальное и минимальное значения соответственно среди количества прослушиваний пользователем u известных ему артистов:

$$l_{u\_max} = \max_{l(u,k) \in L} l(u,k)$$

$$l_{u\_min} = \min_{l(u,k) \in L} l(u,k)$$

  – Стандартный рейтинг (z-рейтинг): $r(u,i) = \frac{l(u,i) - \mu}{\sigma}$.
  Где $\mu = \frac{1}{N} \sum_{k=1}^{N} l(u,k)$, $\sigma = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (l(u,k) - \mu)^2}$

Расчет рейтинга на основе соседей

Допустим для пользователя u известны n его “соседей”. Необходимо определить рейтинг $R(u,i)$ на основе рейтингов $R(u_k,i)$, где $u_k \in N(u)$.

Существует несколько схем для определения рейтинга в этом случае:

- Простейший способ:

$$R(u,i) = \frac{1}{|N(u)|} \sum_{u_k \in N(u)} R(u_k,i)$$

.

- Более эффективный метод принимает во внимание меру “похожести”

пользователей, задаваемую с помощью весовой функции $w(u_i, u_k)$:

$$R(u,i) = \frac{\sum_{u_k \in N(u)} w(u,u_k) R(u_k,i)}{\sum_{u_k \in N(u)} |w(u,u_k)|}$$

.

- И наконец для того, чтобы решить проблему подразумевания пользователями разных шкал рейтингов, используется более точная формула:

$$R(u,i) = \overline{R(u)} + \frac{\sum_{u_k \in N(u)} w(u,u_k)(R(u_k,i) - \overline{R(u)})}{\sum_{u_k \in N(u)} |w(u,u_k)|}$$

.

В данной работе используются второй и третий подходы.

Анализ работы системы

Существует много техник для анализа качества работы системы. Мы рассмотрим только те два вида метрик, которые будут использоваться в данной работе[9].

- **Recall и Precision.** Наиболее популярные метрики для анализа рекомендательных систем. Для расчета этих метрик необходимо помечтить все предсказанные рейтинги в таблицу 2x2: разделить рейтинги на “выбранные” и “не выбранные”, а так же на “релевантные” и “не релевантные”.

	Не выбранные	Выбранные	Всего
Релевантные	N_{rn}	N_{rs}	N_r
Нерелевантные	N_{in}	N_{is}	N_i
Всего	N_n	N_s	N

Для определения релевантности рейтинга необходимо выбрать порог релевантности. Например для системы с рейтингами от 0 до 1 можно взять релевантными рейтинги от 0.5 до 1. В таком случае порог релевантности будет 0.5.

Кроме того расположение рейтингов в указанной таблице зависит от того, сколько “наивысших” рейтингов мы демонстрируем пользовате-

лю.

Precision (точность) - вероятность того, что выбранный (показанный пользователю объект) релевантен. Определяется по формуле

$$P = \frac{N_{rs}}{N_s}$$

Recall (охват) - вероятность того, что релевантный объект показан пользователю. Определяется по формуле

$$R = \frac{N_{rs}}{N_r}$$

Из формул видно, что при увеличении количества выбранных элементов (рекомендованных пользователю) точность убывает, а охват возрастает. Таким образом для представления полной картины необходимо определить значение точности и охвата для различных размеров списка рекомендаций.

- **Half-life utility.** Эта метрика пытается предсказать полезность полученного списка рекомендованных объектов пользователю. В данном случае полезность - это разность между рейтингом и “рейтингом по умолчанию”. Значение метрики определяется по формуле:

$$R = 100\frac{\sum_a a}{\sum_a R_a^{max}}$$

где R_a - это полезность списка для пользователя a, вычисляемая по формуле

$$R_a = \sum_j \frac{\max(r_{a,j} - d, 0)}{2^{(j-1)/(\alpha-1)}}$$

R_a^{max} - это максимальная достижимая полезность списка для опльзователя, если бы список был ранжирован в правильном порядке. Коэффициент α - это “период полураспада”, а точнее, номер в списке, объект, соответствующий которому пользователь просмотрит с вероятностью 50%.

Период полураспада определяется исходя из пользовательского интер-

фейса. Для n показываемых в списке рекомендаций артистов период полураспада необходимо взять n. В работе [10] было использовано значение $\alpha = 5$, но было отмечено, что значение $\alpha = 10$ дает более точный результат.

3.2 Нейронные сети

Для определения соседних пользователей в данной работе используется нейросетевой подход.

В этой работе задача рекоммендаций решается с помощью применения нейронных сетей. В общем случае нейронная сеть представляет собой машину, моделирующую способ обработки мозгом конкретной задачи[4]. Нейроны - элементарные ячейки вычислений.

Основы

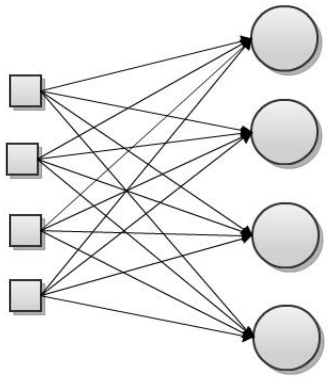


Рис. 3.1: Сеть прямого распространения с одним слоем нейронов (один входной слой и один выходной)

Нейронную сеть как громадный распределенный параллельный процессор , состоящий из элементарных единиц обработки информации, накапливающих экспериментальные знания и предоставляющих их для последую-

щей обработки.

Опишем функционирование нейрона k:

$$u_k = \sum_{j=1}^{m} (w_{kj}, x_j)$$

$$u_k = \phi(u_k + b_k)$$

$x_1, x_2, \ldots x_m$ - входные сигналы, $w_{k1}, w_{k2}, \ldots w_{km}$ - синаптические веса нейрона k, u_k - линейная комбинация входных воздействий, ϕ - функция активации, y_k - выходной сигнал нейрона.

Функции активации бывают различных видов, но можно выделить три основных вида: функцию единичного скачка, кусочно-линейную функцию и сигмоидальную функцию.

Главным свойством нейросети является способность к обучению. Таким образом сеть повышает свою производительность. Обучение - это процесс, в котором свободные параметры нейронной сети настраиваются посредством моделирования среды, в которую эта сеть попала. Тип обучения определяется способом подстройки этих параметров.

Существует несколько различных видов обучения. Но в общем процесс обучения состоит из нескольких этапов. На первом этапе в нейронную сеть поступают стимулы из внешней среды. На втором этапе свободные параметры нейросети изменяются. После изменения внутренней структуры нейронная сеть отвечает на запросы другим способом.

Карты самоорганизации

В данной работе задача нейронной сети - выделение групп “похожих” пользователей. Для этой цели выбрано использование карт самоорганизации.

Этот класс нейронных сетей основан на конкурентном обучении. В картах самоорганизации нейроны помещаются в узлах решетки обычно одно- или двухмерной.Нейроны в ходе конкурентного процесса (выигравший данное соревнование нейрон называется “победителем”) настраиваются на различные входные образы или классы входных образов. Положения таким об-

разом настроенных нейронов упорядочиваются по отношению друг к другу так, что образуется система координат.

Две основные модели отображения признаков - модель Уилшоу-ван дер Мальсбурга и модель Кохонена.

Основная идея модели Уилшоу-ван дер Мальсбурга состоит в том, что для представления информации огеометрической близости предсинаптических нейронов в форме их электрической активностии для использования этих корелляций в постсинаптической решетке соседние предсинаптические нейроны должны связываться с соседними постсинаптическими. Однако эта модель подходит лишь представления отображений, в которых размерности входного и выходного сигналов равны.

В отлиичии от нее модель Кохонена не концентрируется на биологических деталях. кроме того, эта модель подходит для сжатияя данных (сокращения размерности входного сигнала).

Для задачи группировки пользователей подходит второй вид (карта Кохонена). Постановка задачи для нейросети в данном случае будет выглядеть так: на вход нейросети подаются рейтинги исполнителей для определенного набора пользователей (координаты пользователей в системе артистов). Необходимо топологически сгруппировать пользователей. Выходные нейроны сетей при этом играют роль групп. Таким образом произойдет сжатие данных от рейтингов артистов к группам, котолрым принадлежат опльзователи.

Основные этапы обучения нейронной сети.

а) *Конкуренция.* Для каждого входного образа нейроны сети вычисляют относительные значения дискриминантной функции. Эта функция является основой для конкуренции среди нейронов.

б) *Кооперация.* Победивший нейрон определяет пространственное положение топологической окрестности нейронов (обеспечивая базис для кооперации нейронов).

в) *Синаптическая адаптация.* Позволяет возбужденным нейронам увеличивать свои собственные значения дискримнантных функций по отношению к входным образом посредством соответствующих корректировок синаптических весов.

Самоорганизующаяся карта Кохонена

Опишем вкратце алгоритм, по которому происходит работа карты Кохонена.

а) *Инициализация.* Для входных синаптических весов $w_j(0)$ выбираются случайные значения. Главное и единственное требование - раличные значения векторов для $j = 1, 2 \dots l$, где l - общее количество нейронов в решетке

б) *Подвыборка.* Выбирается вектор x из входного пространства с определенной вероятностью. Этот вектор представляет собой возбуждение, применяющееся к решетке нейронов.

в) *Поиск максимального подобия.* Находим наиболее подходящий (победивший) нейрон x на шаге n, используя критерий минимума Евклидова расстояния:

$$i(x) = \arg\min_j \|x - w_j\|,\ j = 0, 1, \dots l$$

г) *Коррекция.* Корректируем векторы синаптических весов всех нейронов, используя следующую формулу:

$$w_j(n+1) = w_j(n) + \nu(n) h_{j,i(x)}(n)(x - w_j(n))$$

где $\nu(n)$ - параметр скороти обучения, а $h_{j,i(x)}(n)$ - функция окресности с центром в победившем нейроне $i(x)$. Эти параметры динамически изменяются в ходе обучения. Опишем их значения.

Функция окрестности - $h_{j,i(x)}(n) = exp(-\frac{d_{j,i}^2}{2\sigma^2(n)})$, $n = 0, 1, \dots$, где $\sigma(n)$ - ширина функции топологической окрестности. Она убывает на каждой итерации. Популярный вариант этой функции: $\sigma(n) = \sigma_0 exp(-\frac{n}{\tau_1})$. σ_0 - начальное значение функции ширины. τ_1 - некоторая константа. $d_{j,i}$ - Евклидово расстояние между победившим (i) и возбуждаемым (j) нейроном в выходном пространстве.

Параметр скорости обучения так же убывает с увеличением количества обучений. Для некого можно выбрать так же экспоненциальную

функцию $\nu(n) = \nu_0 exp(-\frac{n}{\tau_2})$, где τ_2 - так же некоторая константа.

д) *Продолжение* (возвращение к шагу 2).

Обучение происходит до тех пор, пока в карте признаков не перестанут происходить заметные изменения.

Определим используемые в карте параметры.

Параметр скорости обучения лучше выбрать близким к значению 0.1.Со временем он должен убывать, но оставаться больше величины 0.01. Этого можно добиться, взяв $\nu_0 = 0.1$ и $\tau_2 = 1000$ в расчете на 1000 итераций алгоритма.

Функция окрестности должна изначально охватывать практически все нейроны сети и иметь центр в подившем нейроне i. Поэтому можно установить значение σ_0 равное "радиусу" решетки. Константу τ_1 определить по формуле $\tau_1 = \frac{1000}{\log \sigma_0}$ в расчете на 1000 итераций алгоритма.

3.3 Уменьшение размерности данных

Одним из главных недостатков нейронных сетей является проблема обработки входных сигналов очень больших размерностей. Для того, чтобы решить эту проблему, перед группировкой пользователей с помощью нейросети ,мы уменьшаем размерность входных данных.

К примеру у нас в системе имеется N артистов. Чтобы сгруппировать пользователей в пространстве всех артистов, необходимо построить самоорганизующуюся карту с N входных нейронов. Так как это может оказаться проблематичным для больших N, мы уменьшаем количество артистов, по которым происходит группировка опльзователей до некоторого значения K. Для этого мы используем алгоритм уменьшения размерности данных (группируем артистов в небольшие группы с похожими рейтингами в пространстве пользователей).

Существующие алгоритмы

Существует большое количество различных алгоритмов для уменьшения размерности данных. В общем случае они разделяются на два вида:

- Feature selection - основаны на выборке подмножества наиболее релевантных объектов и удалении наименее релевантных.
- Feature extraction - основаны на трансформации многомерного массива данных в массив данных с меньшим количеством измерений.

Перечислим нескольких алгоритмы для уменьшения размерности данных

- Multifactor dimensionality reduction (MDL)
- Multilinear subspace learning (MSL)
- Nonlinear dimensionality reduction (NLDR)
- Semantic mapping (SM)
- Locality-sensitive hashing (LSH)

Для выполнения поставленной задачи нам необходимо трансформировать существующий набор данных (пользователей в координатах артисов) в более малый набор данных. То есть по сути нам необходимо уменьшить количество артистов. Это позволит использовать нейронную сеть с гораздо меньшим числом входов, нежели исходное количество артистов.

Locality-sensitive hashing

Locality-sensitive hashing [11] является одним из популярных алгоритмов для уменьшения размерности данных, производящий кластеризацию данных. По сути является алгоритмом поиска ближайших соседей. Рассмотрим основную идею алгоритма:

Алгоритм дает огромный выигрыш в производительности. Так же существуют его расширения, позволяющие выбрать функции хеширования более аккуратно, таким образом сделав разбиение на кластеры так же лучше.

а) Определяем семейство F хеш-функций

б) Определяем новое семейство G, состоящее из хеш-функций g, определенных как конкатенация k случайно выбранных функций функций

$h_1, h_2, \ldots h_k$ из F.

в) Затем создается L хеш-таблиц, каждая из которой относится к своей функции g

г) На этапе предварительной обработки точки из исходного набора хешируются в каждую из L таблиц.

д) На этапе запроса для точки p проходим по каждой из L функций g и по хеш-таблице определяем точки, лежащие внутри того же множества, что и q.

LSH с k-квантователями

Алгоритм locality-sensitive hashing, основанный на неструктурированных k-means квантователях[12] является расширением описанного выше алгоритма. В отличии от предыдущего, он не использует напрямую хеш-функции.

Опишем алгоритм снижения размерности:

- Выбирается набор T тестовых векторов из исходных данных
- Из тестового набора набирается L наборов (словарей) из K векторов (центроидов) $c_{t,i} \in T$, $t = 1, 2 \ldots L$; $i = 1, 2 \ldots K$
- Для каждого вектора исходных данных для каждого из L наборов находится ближайший вектор:

$$g_t(x) = arg \min_{i=1..K} L_2(x, c_{t,i}),\ t = 1, 2 \ldots L$$

где $c_{t,i}$ - i-й вектор из t-ого словаря.

- Найденный вектор включается в группу
- Центроид сдвигается к центру образовавшейся группы (согласно новому добавленному вектору)

4. Инструменты

Для постановки и реализации эксперимента нам понадобились следующие инструменты:

Apache Mahout

Для целей разработки рекоммендательное системы мы использовали проект Apache Mahout[13] - библиотеку для масштабируемого машинного обучения. Она входит в Apache Hadoop[14] - проект с открытым исходным кодом, в рамках которого разрабатывается программное обеспечение для надежных, масштабируемых, распределенных вычислений. Эта библиотека содержит как инструменты, с помощью которых можно построить свою рекомендательную систему, так и несколько уже реализованных алгоритмов рекоммендаций. Последнии использовались нами для сравнения полученных результатов с уже существующими.

Last.fm

Last.fm[15] - это музыкальный сервис, позволяющий пользователям прослушивать различные композиции, структурировать композиции с помощью тегов, организовывать массовые походы на музыкальные концерты и так далее. Он имеет огромную базу пользователей и общедоступный API, включающий в себя методы для получения информации о пользователях, артистах, наиболее прослушиваемых пользователем исполнителей и прочем. Для проверки работы получившейся рекомендательной системы мы загрузили небольшую часть базы last.fm.

Для получения данных использовалась библиотека last.fm-bindings[16]

5. Реализация

Рассмотрим общую архитектуру реализации.

5.1 Описание программы

Входные данные

Для работы программа получает файл с количеством прослушиваний пользователями артистов в формать tsv.

Выходные данные

Выходными данными являются следующие файлы:

- файл с рекомендованными артистами (включает идентификатор пользователя, артиста и его положение в списке рекомендаций).
- файл с результатами анализа рекомендаций (значения precision и recall для заданных порогов релевантности)

Список рекомендаций формируется таким способом, что артисты с наибольшим спрогнозированным для пользователя рейтингом были в списке выше, чем артисты с меньшим спрогнозированным системой рейтингом для данного пользователя.

Так же в процессе работы программы создается ряд файлов, представляющих собой:

- Количество порослушиваний артистов пользователями во внутреннем представлении рекомендательной системы (идентификаторы пользователей и артистов отличаются от идентификаторов, определенных во

входном файле). Так же файлы соответствия входных идентификаторов внутренним идентификаторам.
- Рейтинги артистов у пользователей, рассчитанные системой.
- Маску разбиениения входных данных на данные для обучения и данные для прогнозирования и анализа.
- Цетроиды как результат алгоритма уменьшения размерности и рейтинги пользователей для них.
- Представление уже обученной нейронной сети (синаптические веса)
- Группы, полученные в результате обучения нейронной сети и пользователи, входящие в них.
- Рейтинги артистов у каждой из групп, а так же рейтинги артистов в результате пересчета на основе группировки нейронной сетью.
- Список рекомендаций (рейтингов артистов, стоящих на каждом месте списка рекомендаций).

Аргументы командной строки

Реализация программы позволяет запускать как полный цикл алгоритма рекомендаций, так и каждую фазу по отдельности. Последнее было сделано для простоты управления процессом работы.
Фаза алгоритма задается с помощью аргументов командной строки. Ключ для указания фазы - **-p**. Доступны следующие фазы:

а) **map** - перенумерация артистов и пользователей во внутреннее представление.
б) **split** - вычисление маски, разбивающей исходные данные на данные для обучения и данные для прогнозирования и анализа.
в) **make_ratings** - вычисление рейтингов на основе данных о прослушивании пользователем композиций.
г) **reduce** - уменьшение размерности данных для последующей обработке их нейросетью.
д) **neuro** - группировка пользователей с помощью нейросети.
е) **recalculate_ratings** - пересчет рейтингов артистов у пользователей на основе группировки.

ж) **recommend** - составление списков рекомендаций.

з) **analyze** - анализ полученных списков рекомендаций.

Отсутствие указанной фазы (или любая неверно указанная фаза) трактуется как запуск полного алгоритма.

Так же существует два параметра для указания директории с исходными данными (**-i**) и директории для сохранения полученных данных (**-o**). При отсутствии указанных путей используется текущая рабочая директория.

Для указания режима подробного отчета, в котором выводятся все сообщения о процессе обработке, необходимо указать ключ **-verbose**.

Параметры

Параметры работы программы настраиваются с помощью двух файлов:

а) filenames.properties

б) algorithm.properties

filenames.properties

Содержит в себе все названия файлов, используемые для хранения промежуточных и рассчитаных данных.

algorithm.properties

Содержит все параметры для настройки работы алгоритма. Доступные параметры:

- Пересчет количества прослушиваний в рейтинги
  - **initial_data**: *listen_count, inversed* - данные, используемые для пересчета в рейтинги: количество прослушиваний или инверсированное количество прослушиваний
  - **normalizaton**: *average, zscore* - тип обработки рейтингов: усреднение или стандартный рейтинг.
- **learning_set_percent** - размер части от общего количества данных, которую занимает набор данных для обучения. (например, 0.7)

- Уменьшение размерности данных
  - **test_set_size** - размер набора, из которого выбираются данные для словаря, использующиеся для уменьшения размерности.
  - **dictionary_size** - количество центроидов в словаре.
  - **dictionary_count** - количество словарей, использующихся для уменьшения размерности данных.
- Группировка с помощью нейронной сети
  - **max_learning_cicles** - максимальное количество циклов обучения.
  - **user_groups_count** - количество групп пользователей, на которые разбиваются пользователи.
  - **training_set_size** - количество пользователей, использующихся для обучения нейронной сети.
- Анализ рекомендаций
  - **recall_and_precision_threshold** - набор порогов релевантности для расчета метрик precision и recall.
  - **alpha** - период полураспада для метрики “half-life utility”.

5.2 Общая структура

На диаграмме 5.1 на странице 31 показана общая структура классов программы. Программа состоит из набора алгоритмов (таких, как перенумерация исходных данных, уменьшение размерности и прочее), которые расширяют класс ***Algorithm***. Необходимый (согласно аргументам командной строки) алгоритм создается на фабрике ***AlgorithmFactory***.

Соответствующие каждому алгоритму экземпляры классов, раширяющих ***AbstractInputStorage*** и ***AbstractOutputStorage*** создаются непосредственно внутри соответствующего алгоритма.

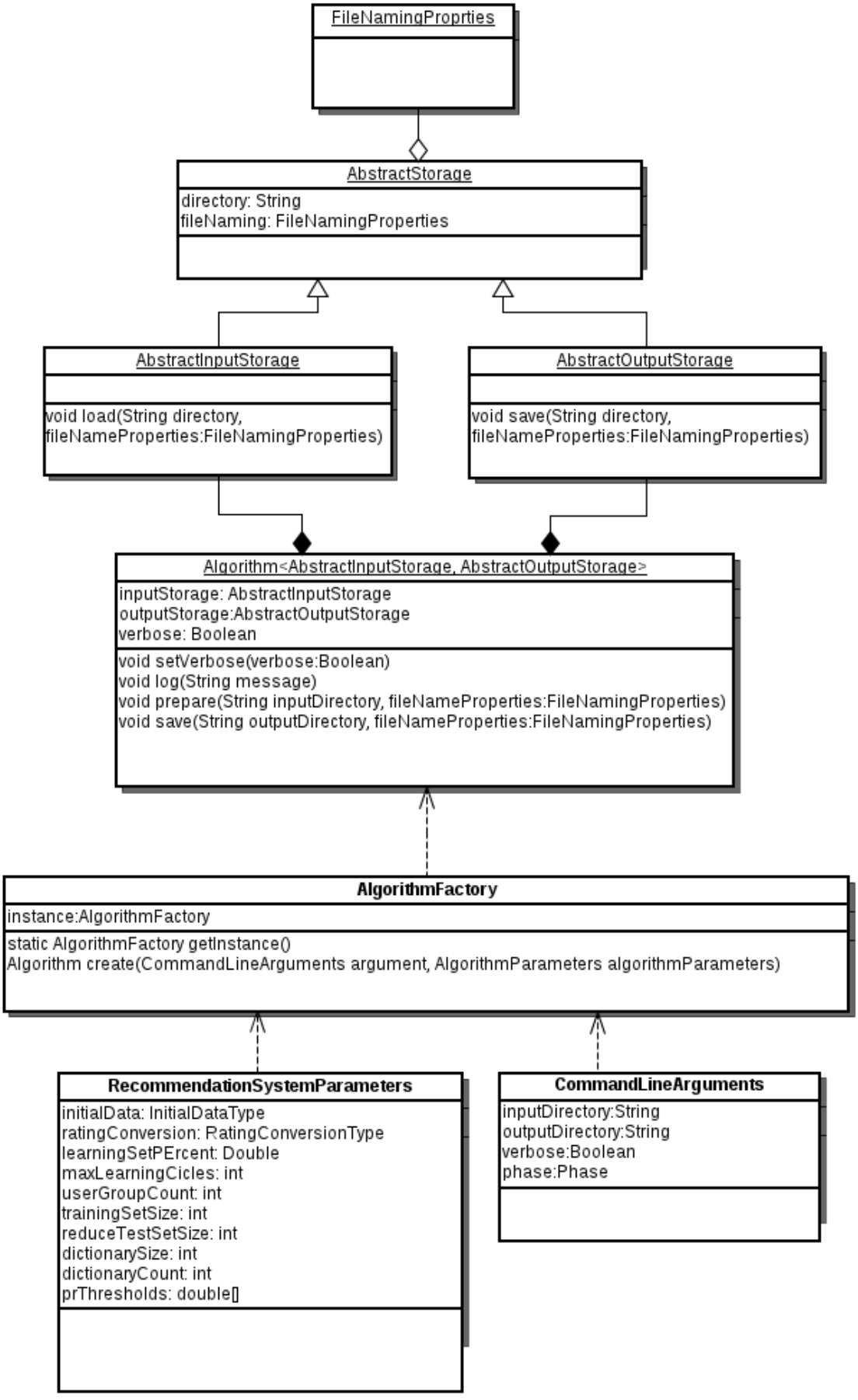


Рис. 5.1: Основная структура классов программы

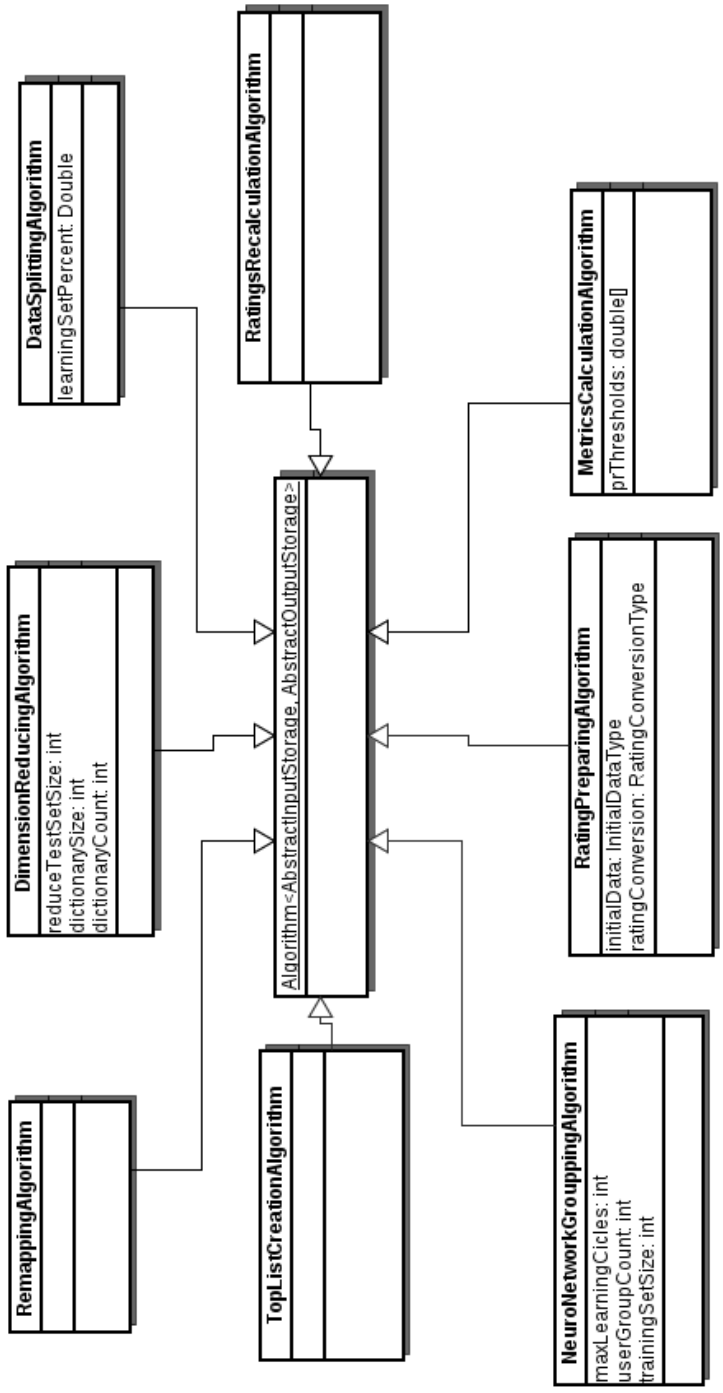


Рис. 5.2: Реализованные алгоритмы

На следующей диаграмме ?? на странице ?? отражены основные реализованные алгоритмы. Для каждого из алгоритмов реализованы хранилища-расширения классов ***AbstractInputStorage*** и ***AbstractOutputStorage***. Так же на диаграмме показаны параметры каждого из этих реализованных алгоритмов. Как указано выше, параметры рекомендательной системы указываются в файле *algorihm.properties*. Соответствующие параметры передаются в каждый из реализованных алгоритмов.

Каждый из реализованных алгоритмов (согласно фазе выполнения) использует данные, полученные в ходе предыдущего. Далее перечислю все алгоритмы и необходимые для их работы данные в порядке выполнения фаз:

а) ***RemappingAlgorithm*** - перенумерует исходные данные (пользователей и артистов) для внутреннего использования.
   - Входные данные:
     – Исходный файл с количеством прослушиваний артистов каждым из пользователей
   - Выходные данные:
     – Файл соответствий нумерации артистов в исходных файлах новым
     – Файл соответствий нумерации пользователей в исходных файлах новым
     – Файл с количеством прослушиваний артистов каждым из пользователей в новой нумерации

б) ***DataSplittingAlgorithm*** - создает маску, разделяющую исходный набор данных на набор данных для обучения и набор данных для тестирования. Маска создается таким образом, что в набор для обучения в любом случае для каждого артиста попадется хотя бы один пользователь, прослушавший артиста, а для каждого пользователя - хотя бы один артист, которого он прослушал.
   - Входные данные:
     – Исходный файл с количеством прослушиваний артистов каждым из пользователей
   - Выходные данные:

  - Файл маски: список пар артист-пользователей, входящих в набор для обучения

в) ***RatingPreparingAlgorithm*** - конвертирует количество прослушиваний артистов пользователем в рейтинги
   - Входные данные:
     - Исходный файл с количеством прослушиваний артистов каждым из пользователей
   - Выходные данные:
     - Файл с рейтингами артистов у пользователей

г) ***DimensionReducingAlgorithm*** - уменьшает размерность данных: подготавливает данные для обработки в нейросети, формирует список центроидов, которые будут взяты за векторы обучения для нейросети.
   - Входные данные:
     - Файл с рейтингами артистов у каждого из пользователей
     - Файл с маской разбиения данных на набор обучения и набор тестирования
   - Выходные данные:
     - Файл с рейтингами центроидов (“центральных артистов”) у пользователей

д) ***NeuroNetworkGrouppingAlgorithm*** - алгоритм группировки пользователей с помощью нейронной сети.
   - Входные данные:
     - Файл с рейтингами центроидов (“центральных артистов”) у пользователей
   - Выходные данные:
     - Файл с обученной нейронной сетью
     - Файл с полученными группами пользователей: принадлежность каждого пользователя группе

е) ***RatingsRecalculationAlgorithm*** - алгоритм пересчета рейтингов на основе группировки, полученной с помощью нейросети.
   - Входные данные:
     - Файл с полученными группами пользователей: принадлежность каждого пользователя группе

  - Файл с рейтингами артистов у каждого из пользователей
  - Файл с маской разбиения данных на набор обучения и набор тестирования
- Выходные данные:
  - Файл с рейтингами артистов у групп, полученных с помощью нейросети
  - Файл со спрогнозированными рейтингами артистов у пользователей

ж) ***TopListCreationAlgorithm*** - алгориттм, сортирующий полученные рейтинги.
- Входные данные:
  - Файл со спрогнозированными рейтингами артистов у пользователей
  - Файл с маской разбиения данных на набор обучения и набор тестирования
- Выходные данные:
  - Файл с артистами, расположенными в порядке убывания спрогнозированного рейтинга для каждого из пользователей
  - Файл со спрогнозированными рейтингами и их положением в списке (в порядке убывания) для каждого пользователя.

з) ***MetricsCalculationAlgorithm*** - алгоритм получения метрик для анализа.
- Входные данные:
  - Файл со спрогнозированными рейтингами и их положением в списке (в порядке убывания) для каждого пользователя.
- Выходные данные:
  - Файл с полученными метриками для каждого из указанных в параметре порогов релевантности

Все входные и выходные данные хранятся в формате csv, исключая результат работы алгоритма ***MetricsCalculationAlgorithm***. Результаты этого алгоритма хранятся в формате, пригодном для быстрого построения графиков в matlab.

5.3 Самоорганизующаяся карта Кохонена

В ходе разработки была реализована структура, представляющая собой самоорганизующуюся карты Кохонена. Описание этой структуры и метода находятся в классе ***SelfOrganizedMap***. Синаптические связи преставлены с помощью матрицы (класс ***DenseMatrix*** библиотеки Mahout).

Для констурирования экземпляра класса ***SelfOrganizedMap*** в конструктор передаются следующие параметры:

а) Количество входных нейронов

б) Количество выходных нейронов

Основной интерфейс представлен следующими методами:

- *startLearning(Vector[] trainingSet, int learningCount)* - обучает сеть. Принимает набор данных для обучения сети, а так же максимальное количество циклов обучения.
- *save(String fileName)* - сохраняет текущее состояние нейросети в файл (формат: csv)
- *print()* - выводит на экран текущее состояние нейросети (состояние синаптических связей)
- *int calculateGroup(Vector vector)* - определяет принадлежность пользователя (представляемого в координатах артистов-центроидов) к группе

Так же объект имеет следующие настраиваемые параметры:

- *initialLearningRate* - начальное значение функции обучения
- *learningRateConst* - константа убывания функции обучения
- *initialEffectiveWidth* - начальное значение функции ширины окрестности
- *effectiveWidthConst* - константа убывания функции ширины окрестности

6. Эксперимент

6.1 Описание

Для анализа точности работы был проведен эксперимент, в ходе которого система была применена к данным, полученным с last.fm (через last.fm API) о живых пользователях. Для выбранных пользователей были получены их самые часто прослушиваемые артисты (с количеством прослушиваний).

Размер полученных данных - 10.000 пользователей, 50.000 артистов, 500.000 рейтингов.

Затем эти данные были обработаны написанной системой рекомендаций и были сняты метрики, необходимые для анализа работы системы. Значения метрик и сопутствующие графики вы можете увидеть ниже.

6.2 Результаты эксперимента

Реализованные алгоритмы

Для реализованных в системе алгоритмов были получены результаты, изображенные на нижеследующих рисунках

На рисунках 6.1 и 6.2 изображены графики для сравнения результатов, полученных с помощью разработанной системой и алгоритмом рекомендаций Slope One. Эти результаты получены на основе данных, переработанных в рейтинги с помощью усреднения количества прослушиваний. Следующими на рисунках 6.3 и 6.8 изображены графики для тех же двух

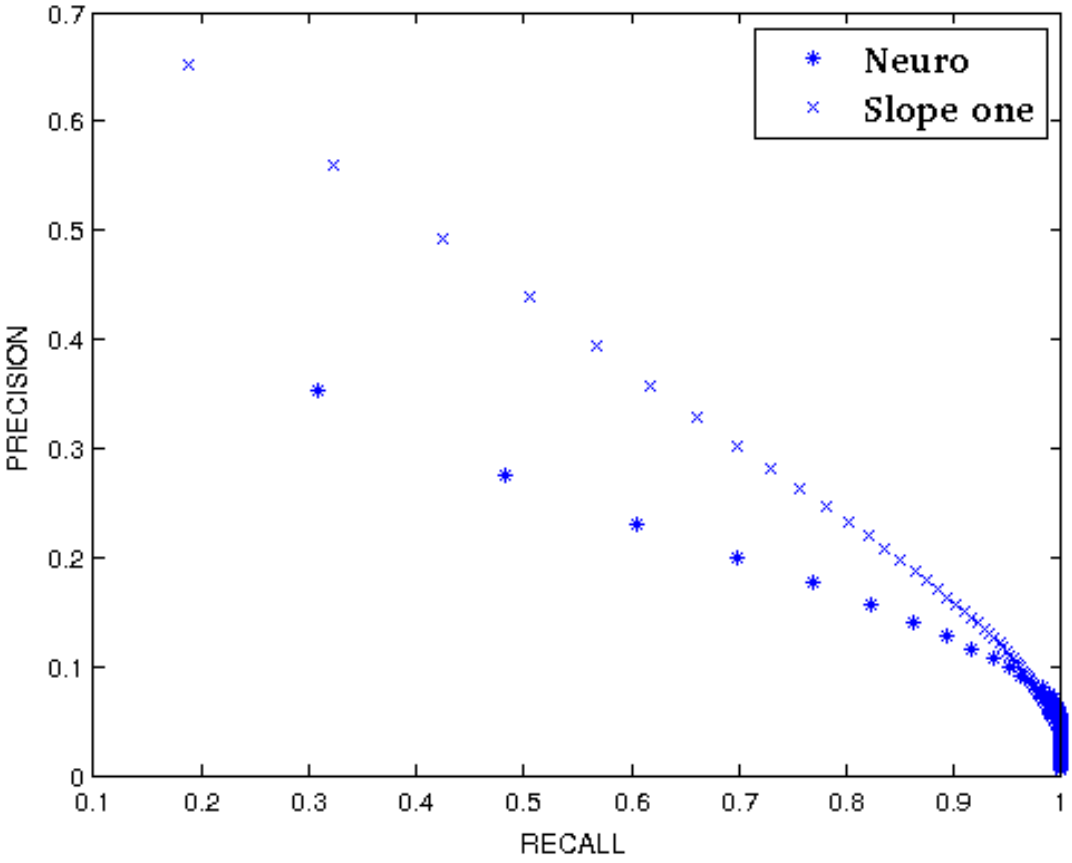


Рис. 6.1: График Recall-Precision для полученной системы (“Neuro”) и алгоритма Slope One. Релевантные рейтинги - начиная с 0.2. Расчет рейтингов - усреднение по количествам прослушиваний

алгоритмов, полученные на основе данных, для которых рейтинги были переработаны с помощью усреднения инверсированных количеств прослушиваний.

На рисунках 6.5 и 6.6 изображены графики для сравнения результатов, полученных с помощью разработанной системой и алгоритмом рекомендаций Slope One. Эти результаты относятся к системе, построенной с на стадартных рейтингах по количеству прослушиваний. Аналогичным образом следующие рисунки 6.7 и 6.8 относятся к системе, построенной с на стадартных рейтингах по инверсированному количеству прослушиваний

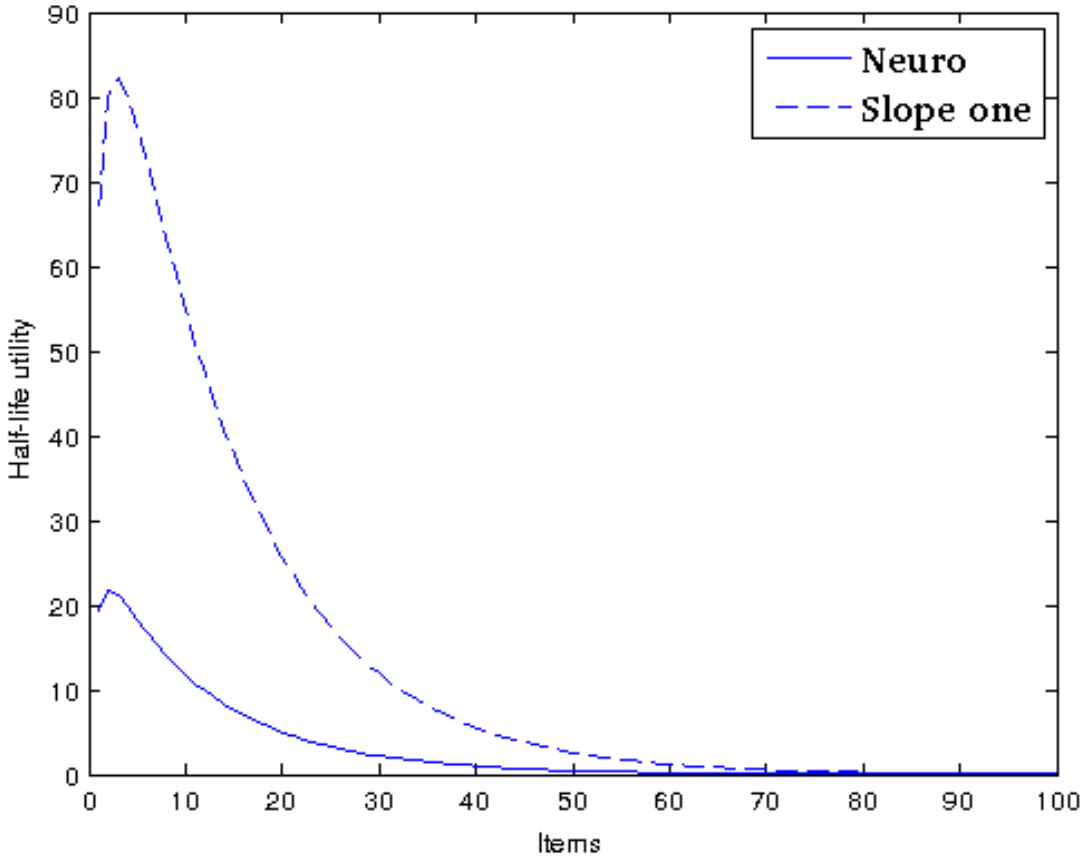


Рис. 6.2: График Half-life utility для полученной системы (“Neuro”) и алгоритма Slope One. Расчет рейтингов - усреднение по количествам прослушиваний

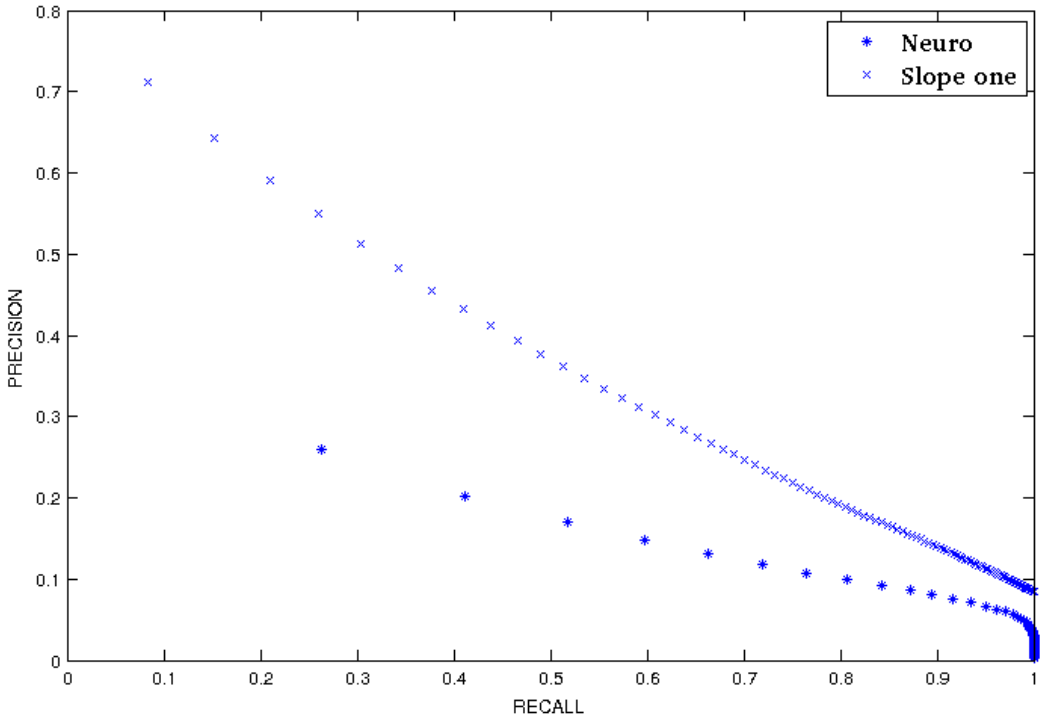


Рис. 6.3: График Recall-Precision для полученной системы (“Neuro”) и алгоритма Slope One. Релевантные рейтинги - начиная с 0.2. Расчет рейтингов - усреднение по инверсированным количествам прослушиваний

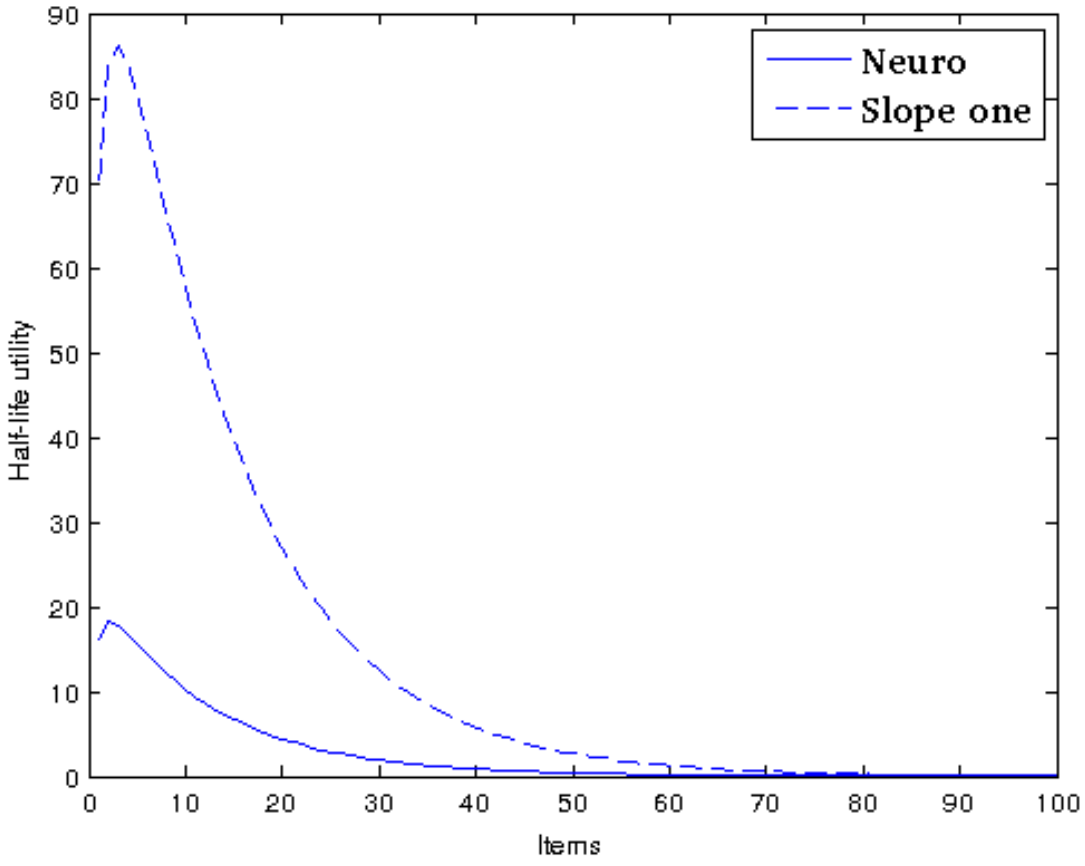


Рис. 6.4: График Half-life utility для полученной системы (“Neuro”) и алгоритма Slope One. Расчет рейтингов - усреднение по инверсированным количествам прослушиваний

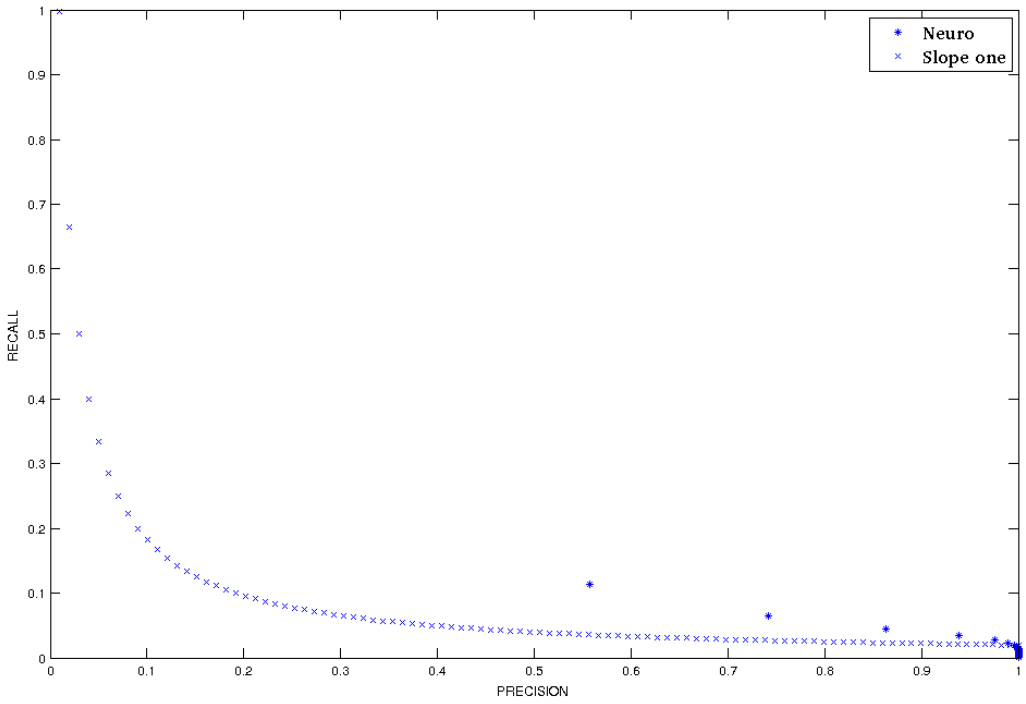


Рис. 6.5: График Recall-Precision для полученной системы (“Neuro”) и алгоритма Slope One. Релевантные рейтинги - начиная с 0.02. Расчет рейтингов - стандартный рейтинг по количествам прослушиваний

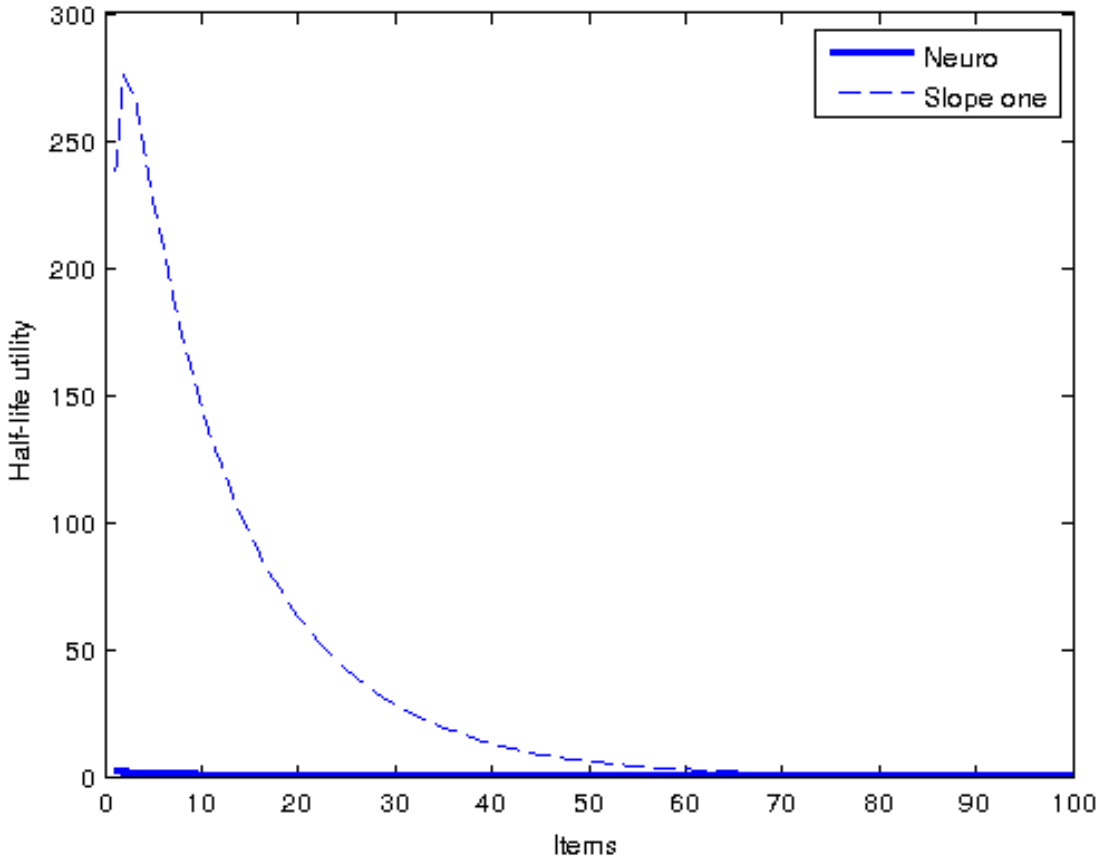


Рис. 6.6: График Half-life utility для полученной системы (“Neuro”) и алгоритма Slope One. Расчет рейтингов - стандартный рейтинг по количествам прослушиваний

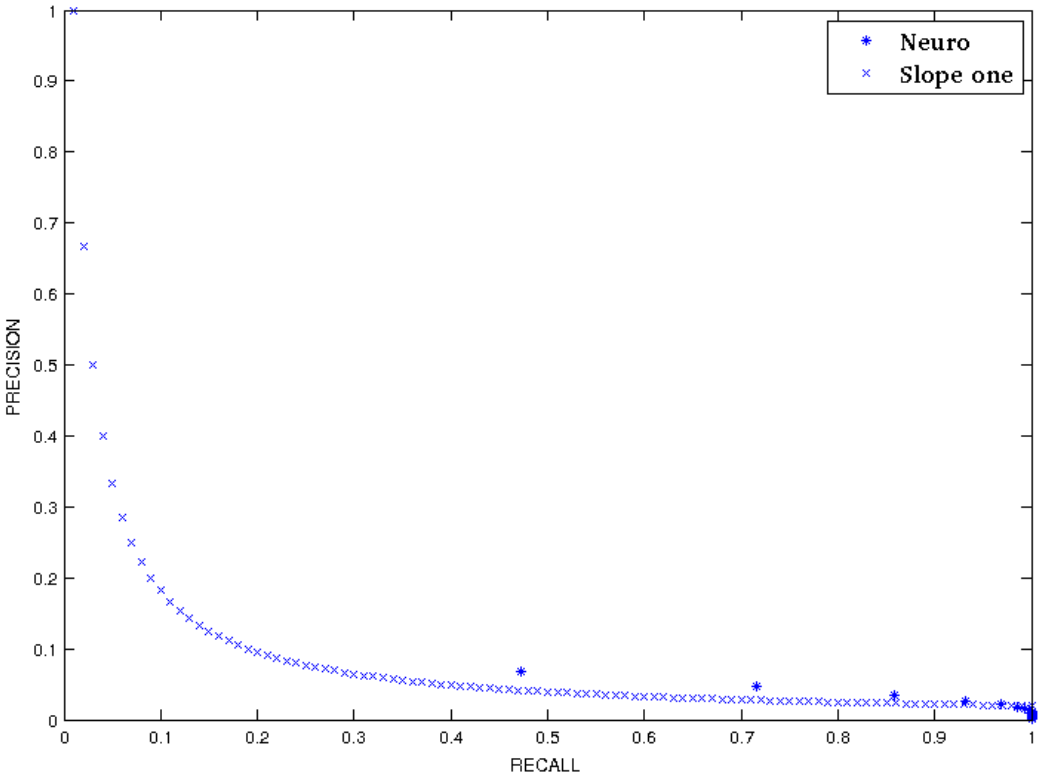


Рис. 6.7: График Recall-Precision для полученной системы (“Neuro”) и алгоритма Slope One. Релевантные рейтинги - начиная с 0.02. Расчет рейтингов - стандартный рейтинг по инверсированным количествам прослушиваний

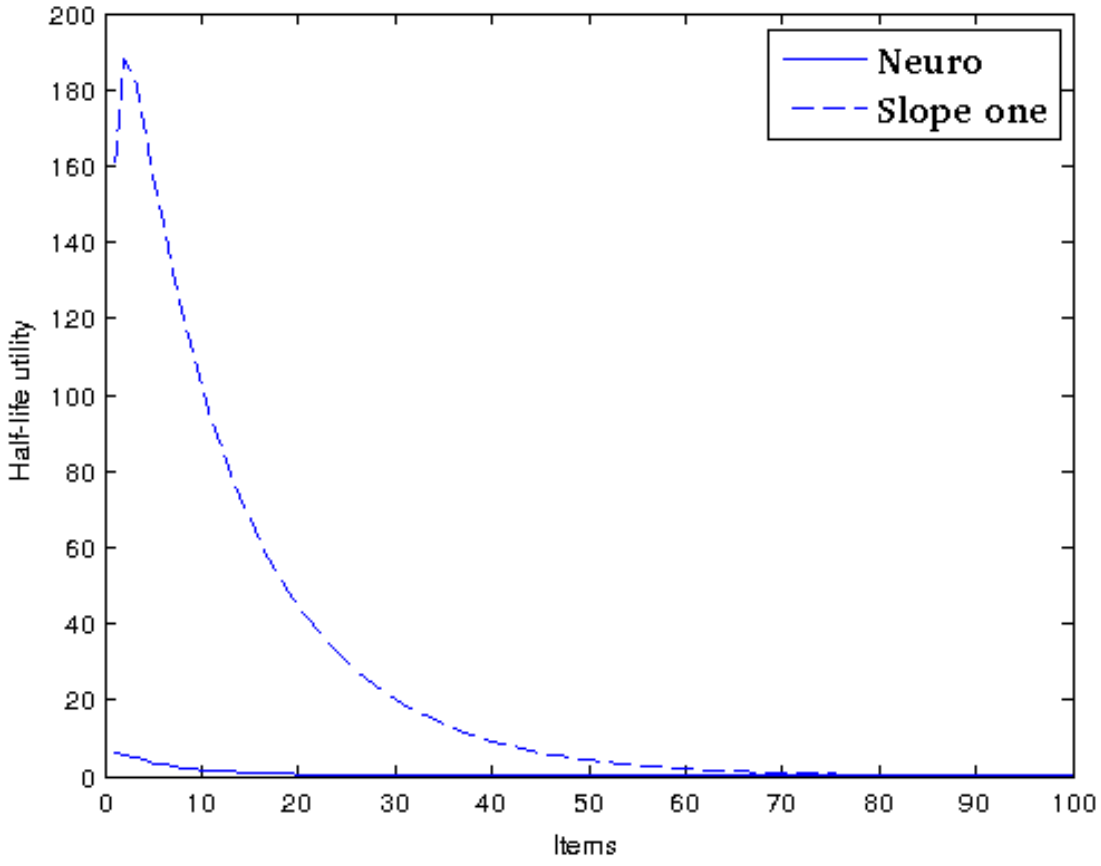


Рис. 6.8: График Half-life utility для полученной системы (“Neuro”) и алгоритма Slope One. Расчет рейтингов - стандартный рейтинг по инверсированным количествам прослушиваний

7. Выводы

Получившаяся рекомендательная система уступает существующим алгоритмам и нуждается в значительной доработке для того, чтобы использоваться в “реальных” рекомендательных системах. Для реализованной системы оптимальное значение размера списка рекомендаций - не более 10ти позиций, лучшие результаты - для 5ти позиций. При увеличении количества позиций наблюдается значительное уменьшение точности рекомендаций, а так же их “полезности” пользователю.

Наиболее хорошей с точки зрения охвата и точности получилась рекомендательная система на основе стандартных рейтингов, однако с точки зрения полезности пользователю выигрывает система на основе усредненных рейтингов.

8. Заключение

В рамках данной работы была реализована система предсказания предпочтений пользователей на основе нейросетевого подхода для систем музыкальных рекомендаций. Были реализованы следующие подзадачи:

- Реализован механизм получения данных для эксперимента
- Реализованы различные механизмы для обработки рейтингов треков
- Реализован алгоритм уменьшения размерности данных для уменьшения размера используемой нейронной сети
- Реализована нейронная сеть для разделения пользователей на группы по предпочтениям
- Реализован механизм предсказания рейтингов треков у каждого конкретного пользователя на основе группировки пользователей по предпочтениям
- Проведен эксперимент, в ходе которого были предсказаны рейтинги для набора пользователей, входящих в тестовый набор
- По результатам эксперимента были получены метрики для анализа эффективности работы и точности предсказаний системы
- роведен анализ результатов эксперимента

Литература

[1] Gilles Louppe. Collaborative filtering: Scalable approaches using restricted boltzmann machines. 2010.

[2] http://www.netflixprize.com/.

[3] The bellkor solution to the netflix prize.

[4] Саймон Хайкин. *Нейронные сети. Полный курс.* 2006.

[5] McCulloch W.S. and W. Pitts. A logical calculus of the ideas immanent in nervous activity.

[6] Theory of neural-analog reinforciment systems and its application to the brain-model problem. 1954.

[7] http://www.foaf-project.org/.

[8] Alexandros Nanopoulos Panagiotis Symeonidis and Yannis Manolopoulos. Feature-weighted user model for recommender systems. 2007.

[9] Johnatan Herloker. Evaluating collaborative filtering recommender systems. 2004.

[10] Empirical analysis of predictive algorithms for collaborative filtering. Brees, j.s., heckerman, d., kadie, c. 1998.

[11] Rajeev Motwani Aristides Gionis, Piotr Indyk. Similarity search in high dimensions via hashing. 1999.

[12] Loic Pauleve. Locality sensitive hashing: a comparsion of hash function types and querying mechanism. 2010.

[13] http://mahout.apache.org/.

[14] http://hadoop.apache.org/.

[15] http://last.fm.

[16] http://code.google.com/p/lastfm-java/.

[17] Joseph Konstan John Riedl Badrul Sarwar, George Karypis. Item-based collaborative filtering recommendation algorithms. 2001.