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 “Построение эмбеддингов методом обучения без учителя”.

Выполнил студент группы 183, 3 курса,
Косакин Даниил Юрьевич

Руководитель КР:
Приглашенный Преподаватель Габов Антон Валерьевич

Москва 2021

Оглавление

1. Введение

В банковской сфере довольно важной задачей является анализ транзакционных данных. На их основе можно получить дополнительную информацию о клиенте - например, насколько вероятно, что клиент с определенной транзакционной историей не сможет вернуть выданный ему кредит. Однако, получение такой информацией является крайне нетривиальной задачей. С развитием технологий машинного обучения, эту задачу в рамках класса задач предсказания на временных рядах начали решать с помощью базовых алгоритмов на основе линейных моделей, случайных лесов и бустингов. Однако, проблемой подобного подхода в данном классе задач было то, что на каждом этапе предсказание осуществлялось для отдельной транзакции, без использования информации о предыдущих транзакциях. Для решения этой проблемы использовалась генерация признаков - например, лагов различных порядков и статистик по этим лагам [3]. Однако, выбор порядков лагов, равно как и статистик, был эмпирической задачей, и количество получаемых признаков делало набор данных очень широким, затрудняя его обучение.

С развитием популярностей рекуррентных нейронных сетей и моделей на ее основе (GRU, LSTM), они стали активнее использоваться в задачах предсказания на временных рядах [4], поскольку преимуществом этих моделей было как раз сохранение информации об объектах, которые были до анализируемого. Однако, оставалась проблема того, что все эти алгоритмы были алгоритмами обучения с учителем, т.е. требовали изначально размеченных данных, чтобы эффективно обучаться. Очень часто для обучения эффективной и точной модели требуются недели, если не месяцы - и в случае с обучением с учителем для каждого нового задания и соответствующей новой целевой переменной необходимо повторять долгий процесс подбора оптимальных параметров и обучения заново.

В данном проекте мы рассматриваем новый подход к анализу транзакционных данных, применяющий обучение без учителя на основе Metric Learning [5]. Суть данного метода обучения заключается в создании эмбеддингов с низкой размерностью таким образом, чтобы расстояние между эмбеддиннгами одного класса была минимальна, а между эмбеддингами разных классов - максимальна. Под “классами” здесь подразумевается не класс целевой переменной,

а определенный класс, в рамках которого можно сделать предположение о потенциальной цикличности и повторяемости данных. Так, в задаче с анализом транзакций, в качестве этого класса был взят идентификатор клиента, поскольку можно сделать предположение, что определенная для каждого клиента модель поведения будет по-своему влиять на его транзакционную историю. Преимуществом данного подхода является то, что можно сначала обучить модель, которая генерирует такие эмбеддинги без использования размеченных данных, а в дальнейшем - создавать уже намного менее сложные модели для анализа размеченных данных с использованием сгенерированных эмбеддингов. К тому же, в банковской сфере у этой модели есть и еще одно преимущество - генерация таких эмбеддингов работает лишь в одну сторону, а значит, хранение информации о транзакционной истории клиента в форме эмбеддингов обеспечивает анонимность и безопасность личных данных клиентов.

В рамках нашего проекта была построена модель, которая применяет данный подход для предсказания возрастных групп клиентов на основе публично доступного набора данных с историей транзакций, и было проведено сравнение с моделью обучения с учителем. Также в качестве эксперимента была оценена точность модели, которая полностью строилась без использования размеченных данных, применяя алгоритм кластеризации к полученным эмбеддингам.

2. Работа с данными

2.1. Набор данных

В качестве используемого набора данных был выбран открытый датасет[1], содержащий информацию о 264850578 транзакциях, совершённых тридцатью тысячами клиентами. О каждой транзакции известно лишь то, кто её совершил, в какой день (с момента отсчета, который неизвестен), на какую сумму и в какой из 203 групп (группы отображаются как целые числа, в отдельном файле предоставлена расшифровка каждой из групп). Также для данных была доступна целевая переменная - одна из четырех возрастных групп, к которой принадлежит клиент и которую надо было предсказать.

[1] https://onti.ai-academy.ru/competition

2.2. Разведочный анализ данных

Был проведен разведочный анализ данных, на основе которого в дальнейшем создавались новые признаки. В первую очередь были проверены относительное распределение признаков и линейная корреляция признаков друг от друга, в результате чего было установлено, что признаки крайне слабо зависят друг от друга.

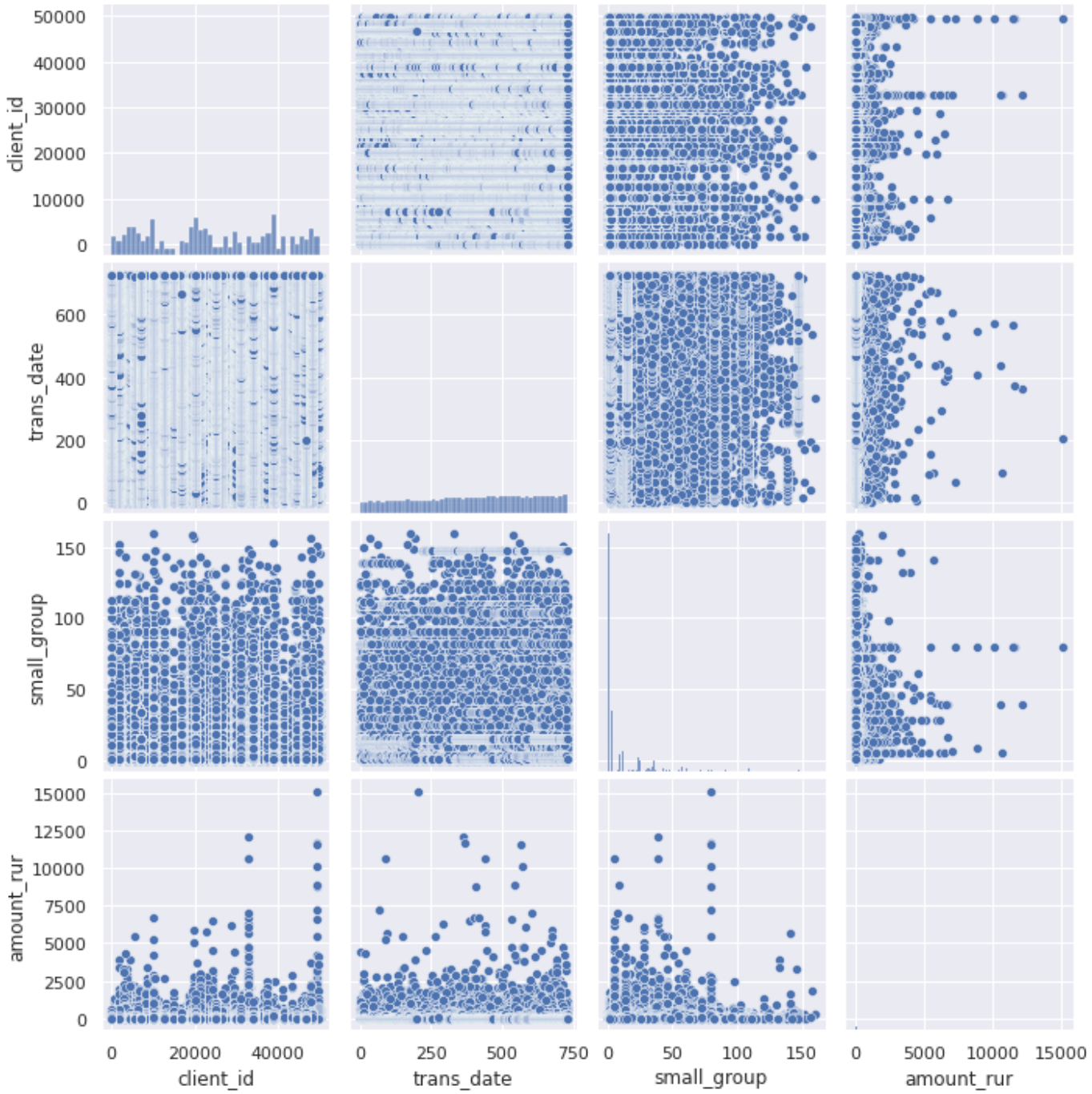

Рис. 1. Попарное распределение признаков

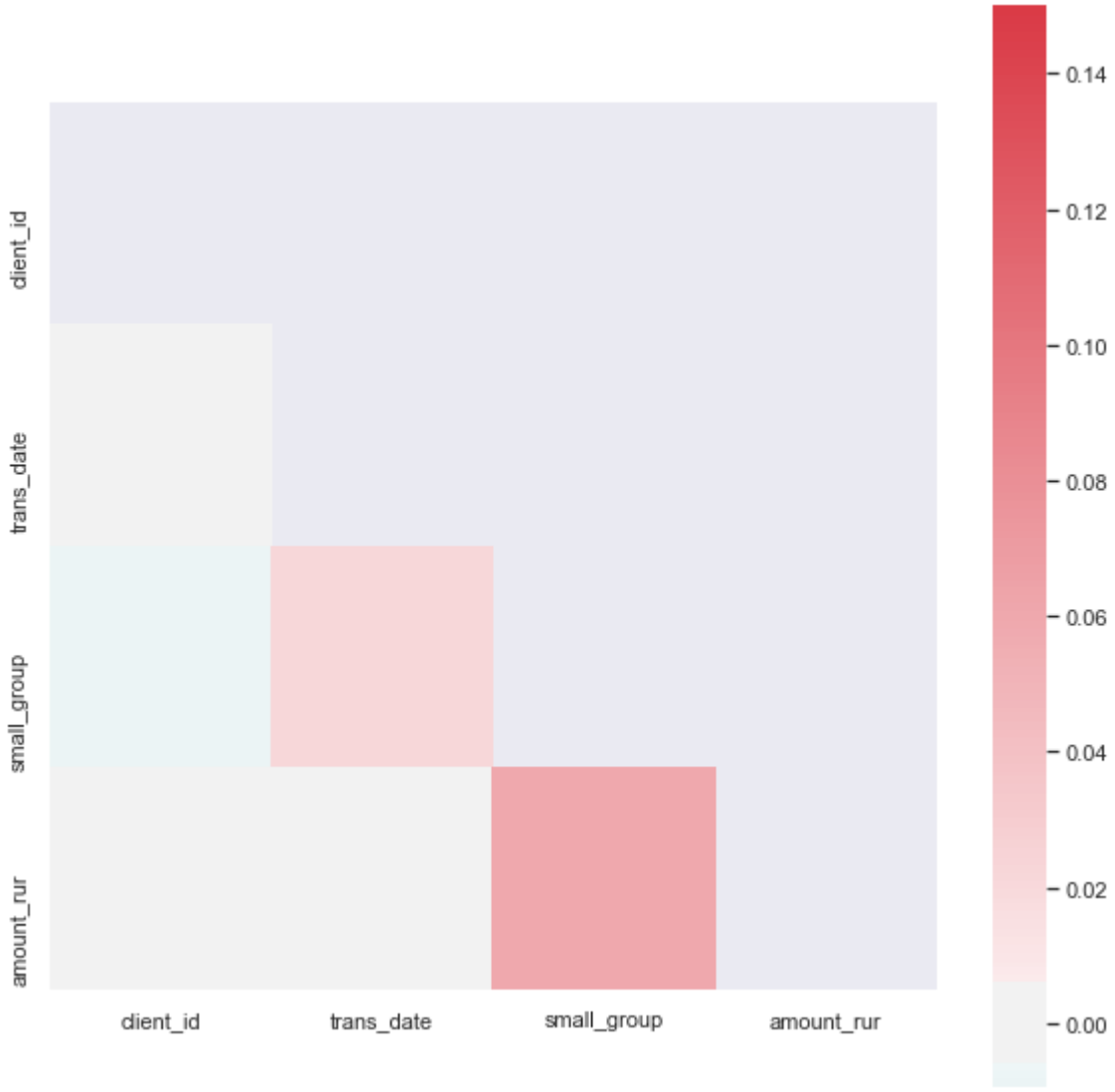

Рис. 2. Попарная корреляция Пирсона признаков.

2.3. Разработка новых признаков

Далее была проведена разработка новых признаков. В первую очередь были созданы признаки на основе информации о дате транзакции. Затруднял задачу тот факт, что дата представлялась лишь целым числом без известного момента начала отсчета, из-за чего приходилось делать предположение, что нулевым днём является день начала эпохи. Были сгенерированы следующие признаки:

- День недели, в который совершена транзакция. С точностью до сдвига корректен при любом начале отсчета, так как в неделе всегда семь дней.
- Месяц, в который совершена транзакция. Является точным лишь если нулевой день отсчета - это начало какого-то года. При этом появляется небольшая погрешность, если положение истинного года начала отсчета в

цикле високосных лет отличается от положения в нём года начала эпохи.

Остальные признаки по датам (такие как, например, день месяца, или год) было сложно надежно сгенерировать в условиях отсутствия начала отсчета.

Было опробовано значительное количество новых признаков, в частности лагов различных значений и статистик по ним, однако, в итоге в рамках текущей задачи выбор новых признаков был ограничен лишь признаками на основе дат, ввиду небольшого прироста к качеству от новых признаков относительно занимаемой ими памяти (что значительно усложняло работу с данными в условиях ограниченной ОЗУ). Все модели сравнивались на основе следующих признаков:

- client_id - уникальный номер клиента, определявший положительный / отрицательный класс в моделях без учителя. Ввиду этого данный признак не использовался для создания новых признаков.
- small_group - идентификатор группы продавца, в которой была совершена транзакция. Всего было 203 уникальные группы, каждая из которых была представлена в форме целого числа.
- day_of_week - день недели, в который была выполнена транзакция
- month - месяц, в который была выполнена транзакция
- amount_rur - сумма транзакции, число с плавающей точкой.

При обучении всех моделей признаки small_group, day_of_week и month были взяты как категориальные, amount_rur - как числовой, а признак client_id - не использовался для обучения.

2.4. Генерирование последовательностей

Все модели, которые использовались в рамках этой работы, используют рекуррентные нейронные сети в качестве своей основы, а значит, требуют последовательностей данных для обучения. Число транзакций для каждого из клиентов разнилось от нескольких десятков до десятков тысяч записей. Кроме того, для обучения моделей, применяющих Metric Learning, было необходимо использовать хотя бы две последовательности от каждого из клиентов.

Для этого из последовательности всех транзакций каждого клиента бралось определенное количество (SEQUENCES_PER_CLIENT) случайных

подпоследовательностей фиксированной длины (SEQUENCE_LEN). Генерировались такие последовательности путём выбора случайной транзакции, после чего брались SEQUENCE_LEN - 1 транзакций после неё. В случае, если транзакций не хватало, к подпоследовательности добавлялись нулевые векторы, которые не влияли на работу рекуррентной нейронной сети.

3. Используемые модели

3.1. Обучение с учителем

В первую очередь было необходимо установить ориентир для будущих моделей. Для сравнения была построена модель, которая полностью обучалась на размеченных данные. Архитектура была основана на предложенной в [1], и состояла из двух частей: кодирование событий (транзакций) и кодирование последовательностей. Первая часть состоит из кодирования категориальных признаков эмбеддингами с низкой размерностью (которые обучаются совместно со всей моделью) и применении нормализации по блоку к числовым признакам. Затем полученные векторные представления категориальных признаков и нормализованные числовые признаки объединяются в один вектор, который проходит через один слой полносвязной сети. Результатом работы этой сети является вектор фиксированной размерности, который тоже проходит нормализацию по блоку и подаётся на вход следующей части модели в качестве вектора, представляющего данную транзакцию. Следующая часть модели использует рекуррентную нейронную сеть для анализа последовательности векторов, представляющих транзакции. Выход последнего слоя рекуррентной нейронной сети подключается к полносвязному слою с количеством нейронов, равным количеству классов. К этому слою применяется софтмакс, и он подаётся на вход функции потерь, в качестве которой выступает функция перекрёстной энтропии.

3.2. Генерация эмбеддингов без учителя

Архитектура модели, которая обучалась создавать эмбеддинги, была очень схожа с архитектурой модели, которая обучалась классификации с учителем. Отличием являлось то, что в качестве результата модели брался последний слой рекуррентной нейронной сети, который подавался на вход функции потерь, стремящейся сделать так, чтобы векторы, полученные от одного клиента, были максимально близки, а векторы,

полученные от разных клиентов - максимально далеки. В нашем случае использовалась функция Margin Triplet Loss [6], которая анализировала тройки предметов - якорь, предмет положительного класса и предмет отрицательного класса. Предметами положительного класса считали те предметы, которые принадлежат той же группе (происходят от того же клиента), что и якорь. Предметы отрицательного класса - отличающиеся по группе (клиенту) от якоря. Расстояние между якорем и предметом положительного класса штрафуется (если оно больше заданного margin), а расстояние между якорем и предметом отрицательного класса поощряется. Для определения расстояния использовалась Евклидова метрика (L2). Для генерации троек использовался поиск полужёстких троек, то есть таких троек, в которых предмет положительного класса был ближе к якорю, чем предмет отрицательного класса, но при этом не ближе, чем на заданное минимальное расстояние (margin).

3.3. Классификация с использованием эмбеддингов

Для классификации на основе полученных эмбеддингов использовалась простая полносвязная неглубокая (два скрытых слоя) нейронная сеть с функцией перекрёстной энтропии в качестве функции потерь. При обучении использовался оптимизатор AdamW [7], реализующий градиентный спуск с адаптивной скоростью обучения для каждого параметра и угасанием весов.

3.4. Кластеризация с использованием эмбеддингов

В дополнение к реализации алгоритма классификации были проведены эксперименты с кластеризацией, чтобы проверить потенциал использования данного подхода при наличии полностью неразмеченных данных[2].

4. Реализация

4.1. Модель обучения с учителем

Поскольку модель в [1] не описывается подробнее основных концепций, при реализации собственных моделей во многом пришлось экспериментировать. Были

[2] В некоторых случаях при кластеризации небольшая разметка данных все же может потребоваться для того, чтобы правильно присвоить каждому номеру кластера свой класс; однако в таких случаях по крайней мере небольшая часть разметочных данных должна быть доступна изначально - иначе была бы неизвестна природа предсказываемой целевой переменной.

исследованы следующие степени свободы:

- Количество генерируемых подпоследовательностей для каждого клиента - SEQUENCES_PER_CLIENT. В статье [2] генерировалось по 5 подпоследовательностей, однако, эксперименты показали незначительное улучшение качества при увеличении этого значения до 10. При дальнейшем увеличении параметра улучшение точности обнаружено не было.
- Размер генерируемых подпоследовательностей. В [2] используется другой метод генерации последовательностей со случайным индексированием, что не предусматривало фиксированный размер подпоследовательностей, однако в среднем он составлял 90. В ходе экспериментов было обнаружено, что модель лучше учится на строго последовательных данных, поэтому был использован алгоритм, описанный в разделе 2.4, который работал на фиксированных размерах подпоследовательностей. Изначально использовались подпоследовательности с 90 транзакциями, однако позднее были проведены эксперименты, в результате которых размер последовательности в 120 показывал результаты чуть лучше.
- Размерность вектора, кодирующего каждое событие. Описываемая в [1] модель использует простую конкатенацию числовых признаков и векторных представлений категориальных признаков, однако в ходе экспериментов быстро выявилось преимущество модели, которая использует полносвязный слой фиксированного размера, через который проходит конкатенация, перед тем как подаваться на вход рекуррентной нейронной сети. Ввиду склонности модели к переобучению, размерности выше 32 слишком усложняли модель, поэтому в финальной версии использовалась размерность 32.
- Размерность векторов, кодирующих категориальные признаки. Поскольку требовалась низкая размерность данных векторов, было использовано значение 8, эксперименты со значениями 4, 16 и 32 показали результаты хуже.
- Структура рекуррентной нейронной сети. Были протестированы архитектуры RNN, LSTM и GRU с различными направлениями и количеством слоев, лучше всего себя показала однонаправленная GRU с одним слоем. Далее были протестированы различные размерности вектора памяти, среди которых была выбрана размерность 32.
- Количество эпох - после 80-85 эпох обучения значения функции потерь на валидационной выборке переставали падать, поэтому было выбрано значение 100.

- Скорость обучения - 0.0001. При более низких скоростях обучения функция потерь переставала падать на схожих значениях, но достигала их дольше, поэтому было выбрано это значение.
- В стремлении побороть переобучение были предприняты попытки использовать Dropout [8] на различных этапах (при векторизации событий, внутри GRU, на последнем полносвязном слое), однако они приводили лишь к ухудшению качества модели как на обучающей, так и на валидационной выборках.

Итоговая модель показывала качество в районе 55% на валидационной выборке для задачи классификации с четырьмя классами, сильно страдая от переобучения. Для борьбы с переобучением предпринимались попытки упростить модель и/или применить Dropout, но они лишь ухудшали качество модели, не уменьшая степень переобучения. Отчасти помогло увеличение размеров блоков до 256, но даже с этим изменением качество на валидационной выборке остается нестабильным и отстает от качества на тестовой выборке.

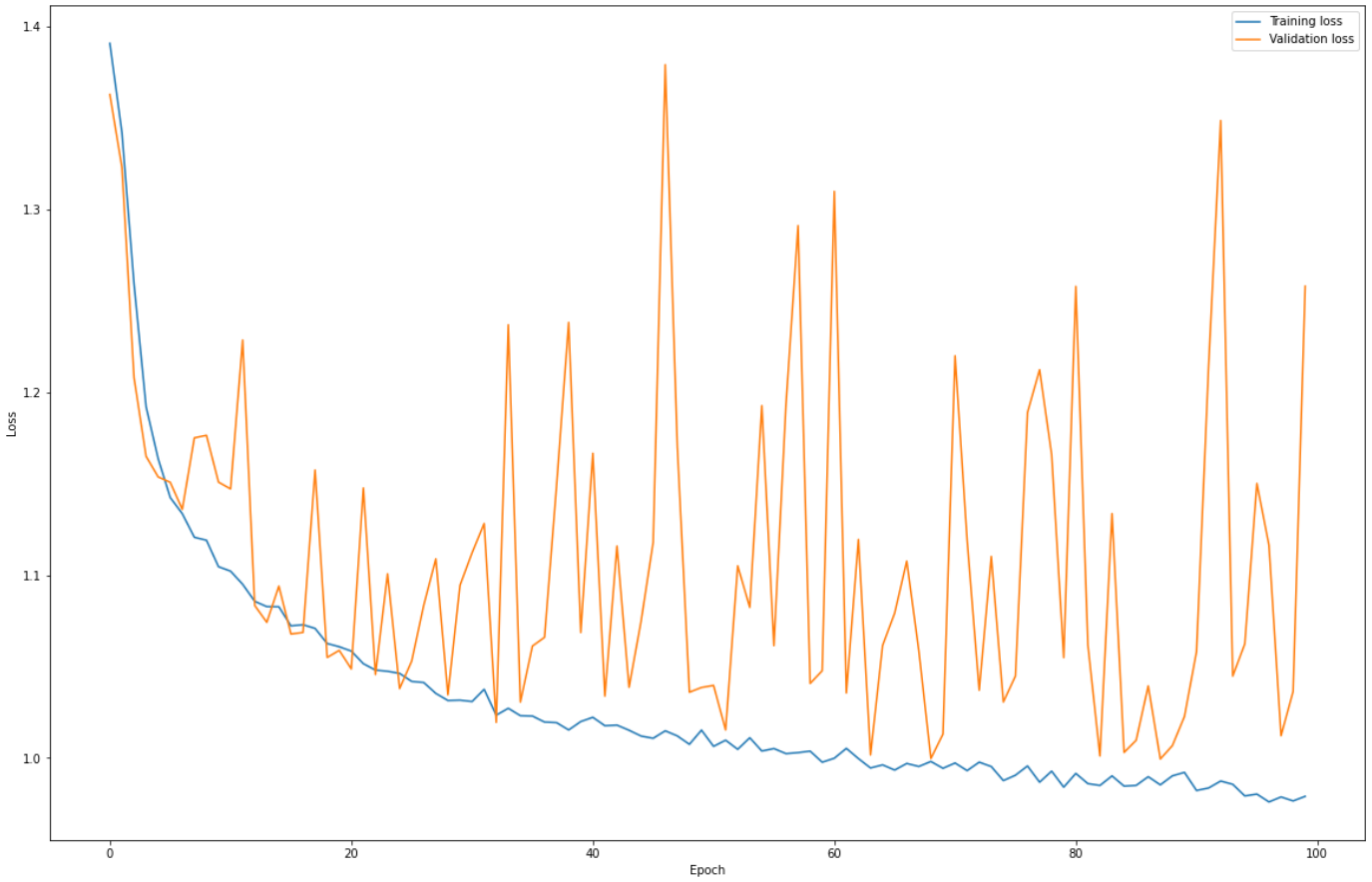

Рис. 3. Процесс обучения эмбеддингов с учителем.

4.2. Генерация эмбеддингов

Многие степени свободы совпадали с моделью из 4.1, однако открывались и некоторые новые. Опишем лишь те, которые отличались, и те, которые появились в этой модели:

- Архитектура рекуррентной нейронной сети почти не изменилась, изменилась лишь размерность ячейки GRU, которая и определяла размерность итогового эмбеддинга - она возросла до 256.
- В качестве функции потерь изначально использовалась предложенная в [2] Margin Loss по парам, однако переход на Margin Loss по тройкам дал значительный прирост к качеству классификаторов, используемых далее.
- Скорость обучения была снижена до 0.0005
- Количество эпох возросло до 40, т.к. функция потерь на валидационной выборке переставала падать примерно только после 35 эпохи.

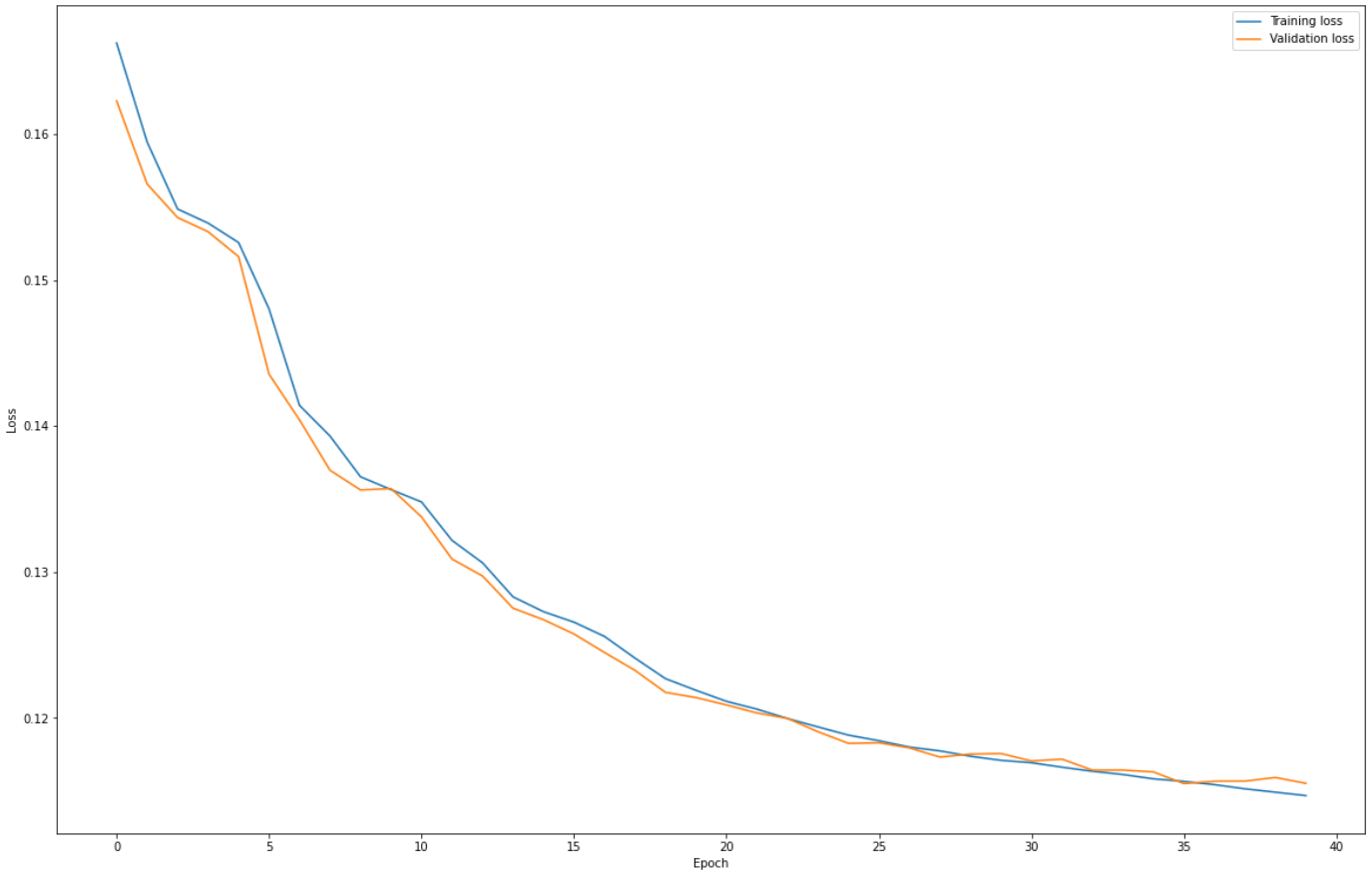

Рис. 4. Процесс обучения эмбеддингов без учителя.

4.3. Классификатор

На данном этапе пространство для экспериментирования было крайне широким. В [2] в качестве классификатора использовался бустинг LightGBM [9], который стал отправной точкой и для нас. Было протестировано множество параметров классификатора LightGBM, однако переход с бустингов к неглубоким Feed-Forward сетями дал итоговый прирост к качеству около 2-2.5%. В итоге в качестве классификатора была выбрана Feed-Forward сеть с двумя скрытыми слоями, по 256 нейронов в каждом. Обучался классификатор в течение 80 эпох со скоростью обучения 0.001, показывая в итоге точность на валидационной выборке около 59%.

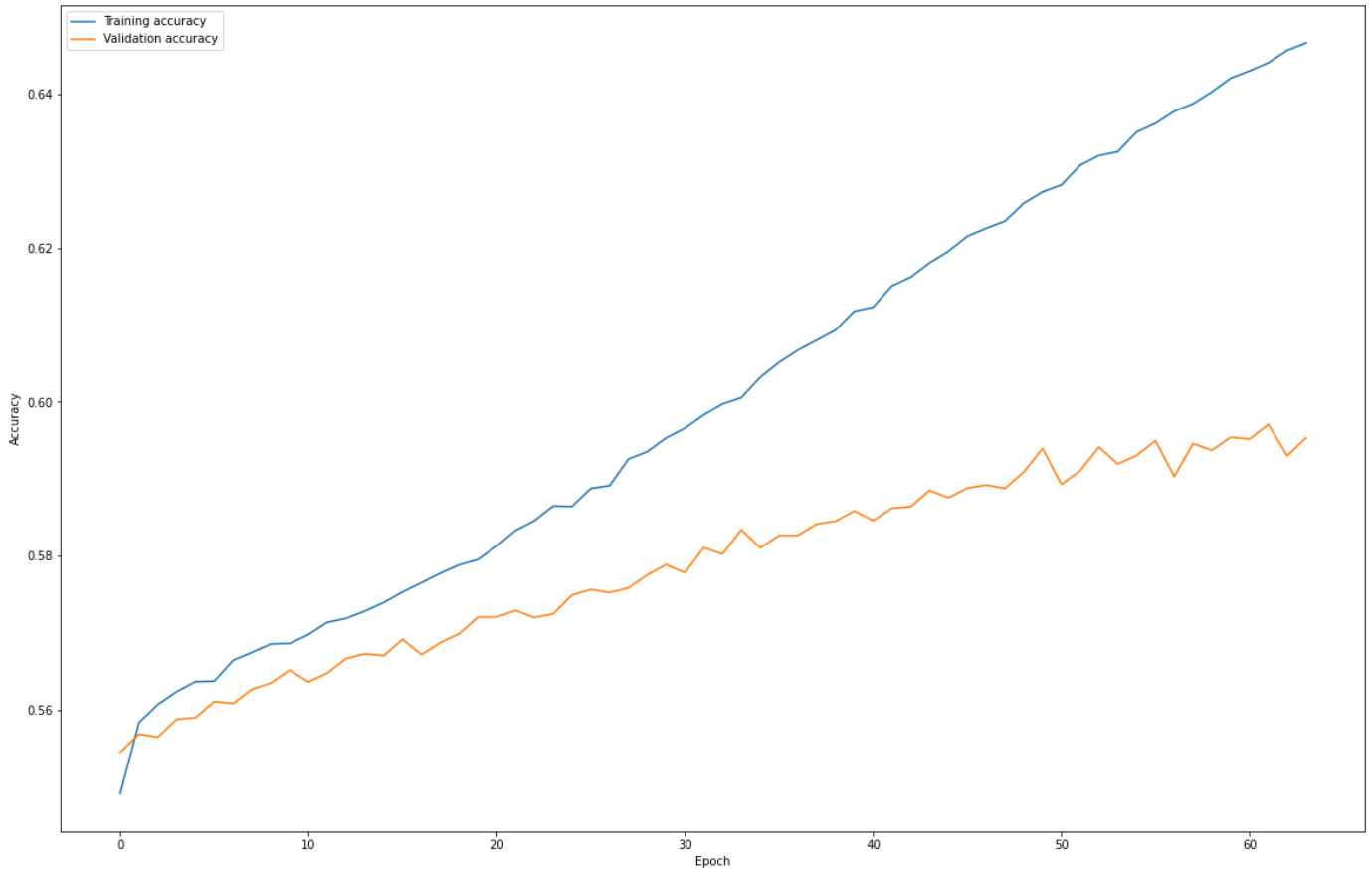

Рис. 5. Процесс обучения классификатора на полученных эмбеддингах.

4.4. Кластеризация

Для кластеризации был использован алгоритм k-средних ввиду большого объёма данных, из-за которого применение других алгоритмов с имеющимися вычислительными мощностями (в частности, ОЗУ) было затруднительно. Несмотря на свою простоту, даже данный алгоритм показал точность в 45,31% при обучении без разметочных данных. Для

этого была использована реализация из библиотеки scikit-learn с параметрами n_clusters=4 - количество классов целевой переменной, algorithm='full', tol=1e-3.

Далее были проверены все возможные биекции, отображающие номера кластеров в номера классов, и из них была выбрана та, которая давала максимальную точность - таким образом, мы поняли, какому классу должен соответствовать каждый из кластеров.

4.5. Уменьшение размерности

Наконец, для визуализации результатов кластеризации были протестированы два алгоритма уменьшения размерности - метод главных компонент и UMAP [10]. В отличие от метода главных компонент, UMAP выполняет нелинейное разложение, а значит, лучше замечает и выделяет нелинейные зависимости в данных. Результаты работы двух алгоритмов с предсказаниями алгоритма k-средних представлены на графиках ниже:

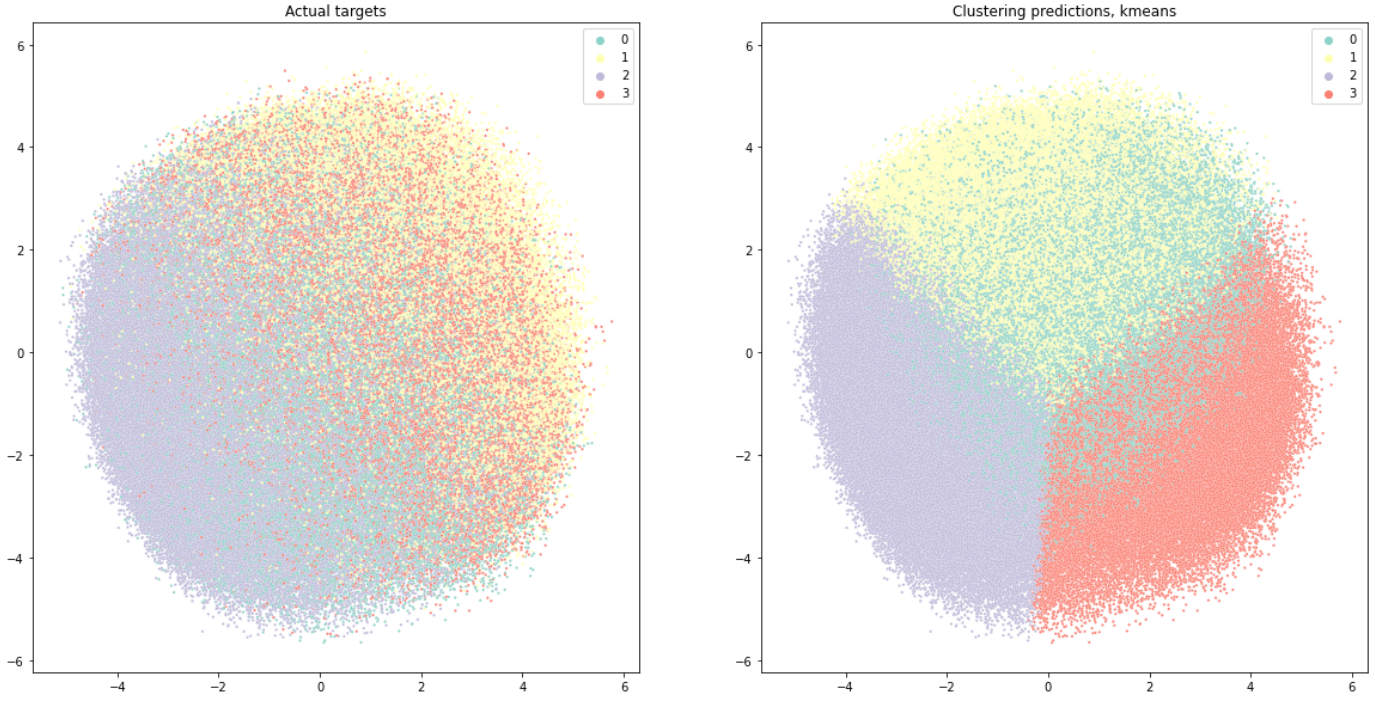

Рис. 6, результат работы k-средних на разложении с использованием МГК.

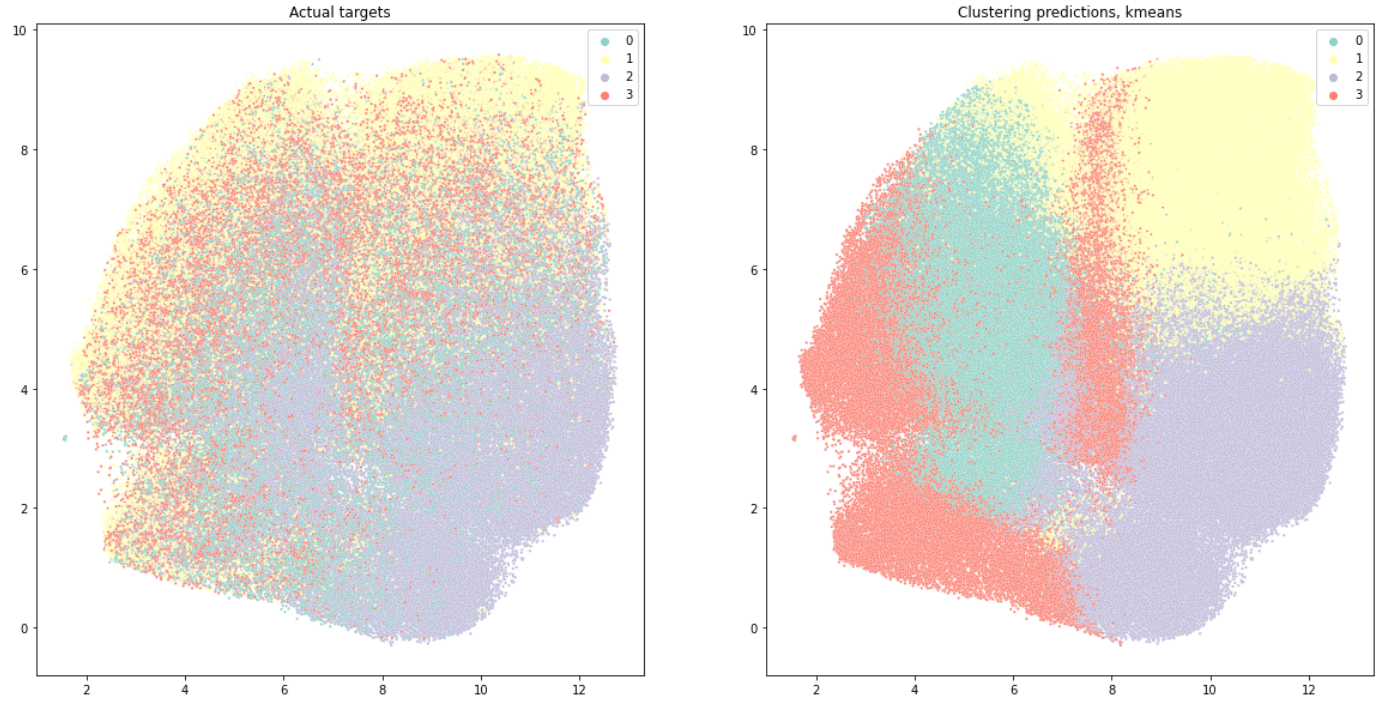

Рис. 7, результат работы k-средних на разложении с использованием UMAP.

Можем заметить, что довольно явно выделяется класс 2, который распознает и алгоритм k-средних. Поскольку мы не знаем, какой класс обозначает какую из возрастных групп, мы не можем точно узнать, какая именно из них так сильно выделяется. Остальные же классы при разложении до двух измерений больше напоминают шум, и по ним сложно распознать какое-то конкретное распределение (разве что можно заметить, что класс 1 дальше остальных от класса 2 на UMAP-разложении).

5. Вывод

В данной работе был проанализирован метод получения векторных представлений потоковых данных на примере задачи классификации данных о транзакциях, совершенных клиентами банка за определенный срок. Для сравнения была создана модель обучения с учителем с модификацией архитектуры, описанной в [1]. Модель, которая использовала нейронную сеть для классификации полученных векторных представлений последовательностей транзакций, показала точность не хуже, чем модель с учителем. При этом преимуществом этой модели было то, что разметочные данные были использованы лишь при классификации на полученных эмбеддингах. Сами же эмбеддинги генерировались без использования целевой переменной - вернее, с использованием другой целевой переменной, обозначавшей класс, в котором можно сделать предположение о цикличности и повторяемости данных - в нашем случае в качестве такого класса был использован идентификатор клиента. Обучение эмбеддингов было самой вычислительно интенсивной частью модели, обучение классификатора в данной модели же проходило намного быстрее, чем обучение классификатора с учителем. Исходя из этого факта можно сделать вывод, что подобный подход может быть полезен в тех случаях, когда в наличии имеются большие объёмы данных, среди которых либо присутствует значительная часть неразмеченных, либо данные имеют различные целевые переменные для разных задач. В таком случае обучить генератор эмбеддингов можно на всех данных и использовать простые классификаторы уже на полученных эмбеддингах, вместо того, чтобы обучать сложные классификаторы для каждой из задач по отдельности.

Эксперименты с кластеризацией показали, что модель, генерирующая эмбеддинги, может выдавать относительно высокую точность даже в отсутствие размеченных данных. Это означает, что векторы, представляющие временные ряды, которые выдает эта модель, в значительной степени отражают структуру и содержание большого объёма данных, на основе которых строятся эти векторы.

6. Цитируемая литература

1. Dmitrii Babaev, Maxim Savchenko, Alexander Tuzhilin, and Dmitrii Umerenkov. E.T.-RNN: Applying Deep Learning to Credit Loan Applications, 2019.
2. Dmitrii Babaev, Ivan Kireev, Nikita Ovsov, Mariya Ivanova, Gleb Gusev, and Alexander Tuzhilin. Event sequence metric learning, 2020.
3. Framing Time Series as a Supervised Learning Problem
4. Hansika Hewamalage, Christoph Bergmeir, Kasun Bandara. Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions, 2020
5. Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information, 2003.
6. Triplet Loss and Online Triplet Mining in TensorFlow
7. Ilya Loshchilov, Frank Hutter. Decoupled Weight Decay Regularization, 2019.
8. Nitish Srivastava, Geoffrey, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov.Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014.
9. Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree, 2017.
10. Leland McInnes, John Healy, James Melville, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, 2018

7. Ресурсы.

1. Github проекта.
2. Папка с данными проекта, среди которых исходный набор данных и различные контрольные точки работы моделей.