# Отчет НИР

#### Дискретизаторы

**Описание экспериментов**

Для проверки чистого влияния дискретизации числовых признаков на качество модели была проведена серия экспериментов, в которых числовые признаки преобразовывалиь в категориальные с помощью одного из дискретизаторов, после чего массив данных состоящий полностью из категориальных признаков передавался в TrxEncoder, где кодировался с помощью NoisyEmbedding - baseline эмбеддингами, использующимися в модели CoLES в данный момент.

После кодирования над эмбеддингами применялась одна из агрегирующих функций - CAT (конкатенация эмбеддингов), MEAN (среднее эмбеддинов) или SUM (сумма эмбеддинов) и результат передавался в SeqEncoder (исп. GRU, как и в предыдущих экспериментах). Веса NoisyEmbedding не замораживались на время обучения.

В экспериментах использовались следующие дискретизаторы:

1. Quant20, Quant100

Разбивает числовой признак поквантильно, на 20 и 100 бинов соответственно. Этот дискретайзер использовался в ранних экспериментах с glove-эмбеддингами в качестве базового.

1. Uni20, Uni100

Разбивает числовой признак равномерно, на 20 и 100 бинов соответственно.

1. KMeans20, Kmeans100

Проводит 1d-кластеризацию и разбивает числовой признак по центрам кластеров (каждое числовое значение кодируется id кластера, в который оно попало. Разбивает на 20 и 100 бинов соответственно.

1. SingleTree20, SingleTree100

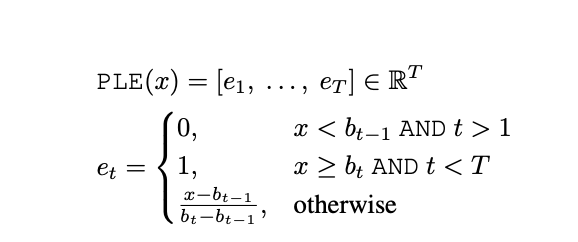
Архитектура этих дискретизаторов была взята из статьи <https://ai.stanford.edu/~ronnyk/disc2.pdf.>

Данный дискретизатор является target-aware: для каждого числового признака строится решающее дерево с помощью алгоритма CART (в статье использовался C4.5, но я решил заменить его на CART, т к для него есть хорошо оптимизированная реализация в sklearn). В качестве target берется target из исходного датасета транзакций. После построения дерева проводится прунинг, чтобы исключить наименее значимые узлы. Параметры для алгоритма прунинга подбираются таким образом, чтобы приблизить число бинов к указанному (т е в SingleTree20 ~20 бинов).

1. DeepTLF3, DeepTLF6, DeepTLF9

Дискретизатор, принцип работы которого основан на обучении ансамблей деревьев (3, 6, 9 - число деревьев в ансамбле) на регрессивную задачу предсказания самих значений исходных признаков. Подробнее писан в предыдущем отчете. Его использование дало хорошие результаты в связке с glove-эмбеддингами. Гипотетически должен превосходить по качеству другие энкодеры, поэтому далее будем именовать первые четыре дискретизатора как baseline discretizers.

Дополнительно, вместе с экспериментами, описанными выше был опробовал альтернативный подход к кодированию числовых признаков - вместо дискретизации + general embedding (NoisyEmbedding или Glove) использовалось PLE над числовыми признаками в эмбеддинг размерности эмбеддингов над категориальными.



Размер такого эмбддинга задавался через количество бинов. Этот альтернативный способ кодирования был опробован для всех дискретизаторов.

**Анализ дискретизаций**

В ноутбуке coles\_discretizer\_analysis.ipynb проведен анализ дискретизаций, полученных при использовании различных дискретизаторов над числовым признаком amount\_rur (объем транзакции) в age\_bins датасете и amount (объем транзакции со знаком) в gender датасете. Рассмотрено распределение границ бинов (в линейных и логарифмических координатах), а так же распределение по ним числовых значений.

В целом, можно отметить, что наиболее удачными оказались дискретизаторы с сбалансированным распределением внутри бинов: поэтому выпадают Kmens и Uniform, где, несмотря на большое количество бинов, большая часть значений сконцентрирована в нескольких, это отражено и в распределением разделителей — у Quant, SingleTree и DeepTLF основная масса разделителей смещена в зону с наибольшим количеством элементов выборки, а не растянута по всему числовому диапазону.

P. S.

В дальнейшем разделе с результатами экспериментов будем наблюдать, что качество моделей содержащих в своей архитектуре DeepTLF деградировало на gender датасете, относительно Quant — на графиках распределения значений по бинам видно, что на датасете gender, даже на DeepTLF9, получилось довольно малое количество бинов, содержащих значимое количество элементов выборки.

Это говорит о том, что DeepTLF подход не особо стабилен и на разных данных может потребоваться больше или меньше деревьев для получения достаточного количества значимых трешхолдов ( в основном это зависит от общего кол-ва признаков, полагаю, т к каждое дерево в ансамбле строится на основе всего массива признаков, значит чем больше признаков, тем меньше шанс что разбиение произойдет по дискретизируемому)

**Обучение и тестирование**

\* Все эксперименты проводились с эмбеддингами размера 16. (в предыдущих экспериментах я брал 24, тут решил взять поменьше чтобы ускорить процесс обучения, исследование касательно влияния размера эмбеддинга на качество модели будет описано во втором разделе).

\* Во избежании переобучения / недообучения использовался алгоритм early-stopping на валидационную метрику recall\_top\_k, количество эпох варьировалось от 5-6 до 22-24.

\* Для оценки качества модели использовались три метрики: recall\_top\_k (k = 5), accuracy (средн. на 5 обученных моделях) на задаче классификации для sklearn.RandomForestClassifier (acc\_random\_forest) и точность на задаче классификации для lightgbm.LGBMClassifier (acc\_boosting). Предыдущие модели оценивались только с помощью метрики acc\_random\_forest, поэтому я пересчитал новые метрики для ряда старых моделей с glove и baseline-подхода - подробнее в разделе ‘результаты’.

\* Полная серия экспериментов проводилась на датасете age\_bins, так же для проверки стабильности, лучшие модели из каждой подгруппы были проверены на датасете gender.

**Результаты**



Результаты для датасета age\_bins (age\_bins\_metrics.xlsx в репозитории)

Эксперименты разбиты по типу агрегации и подгруппам определенным спецификой используемого подхода:

1. Первая подгруппа: agg\_func(NoisyEmbedding(DeepTLF(numerical), categorical))
2. Вторая подгруппа: agg\_func(NoisyEmbedding(baseline\_disc20bins(numerical), categorical))
3. Третья подгруппа: agg\_func(NoisyEmbedding(baseline\_disc100bins(numerical), categorical))
4. Четвертая подгруппа: agg\_func(PLE(DeepTLF(numerical)), NoisyEmbedding(categorical))
5. Пятая подгруппа: agg\_func(PLE(baseline\_disc(numerical)), NoisyEmbedding(categorical))

По итогам, при CAT агрегации эмбеддингов лучше всего себя показал DeepTLF9 с кодированием всех признаков через NoisyEmbedding. SingleTree20 тоже показал довольно хорошие результаты, особенно учитывая мале количество бинов при дискретизации. Модели с PLE-based эмбеддингами над числовыми признаками напротив оказались не особо удачным: можно отметить только SingleTree с довольно неплохим качеством. Так или иначе в среднем обучаемые эмбеддинги оказались эффективнее PLE кодирования.

Для MEAN и SUM агрегаций DeepTLF уступает обычной Quantile100 дискретизации. Так же важно отметить, что MEAN и SUM агрегации практически не просаживают качество в такой реализации, что достаточно важно, так как такие агрегации сильно снижают размер итогового представления, которое поступает на вход SeqEncoder.



Результаты для датасета gender (gender\_metrics.xlsx в репозитории)

На датасете gender эксперименты проводились только для моделей показавших хорошее качество на age\_bins. Метрики accuracy оказались не особо показательными на этих данных - их значения варьируются в пределах +-0.015 п, поэтому разумно будет считать ключевой recall\_top\_k. По ней наилучший результат показала модель с quantile100 дискретизатором на всех типах агрегаций. На age\_bins DeepTLF показывал лучший результат, но там разница была менее 0.01 п + по другим метрикам был лучше Quantile100. Из этого можно сделать вывод, что Quantile100 лучше DeepTLF, в моделях с NoisyEmbedding + дискретизация.

Так же интересно было сравнить вышеописанные модели с моделями, использующими glove эмбеддинги для кодирования категориальных и дискретизированных числовых признаков, опираясь на новые метрики. В таблицах ниже результаты для ряда моделей с glove-based trx\_encoder на обоих датасетах:

Age\_bins



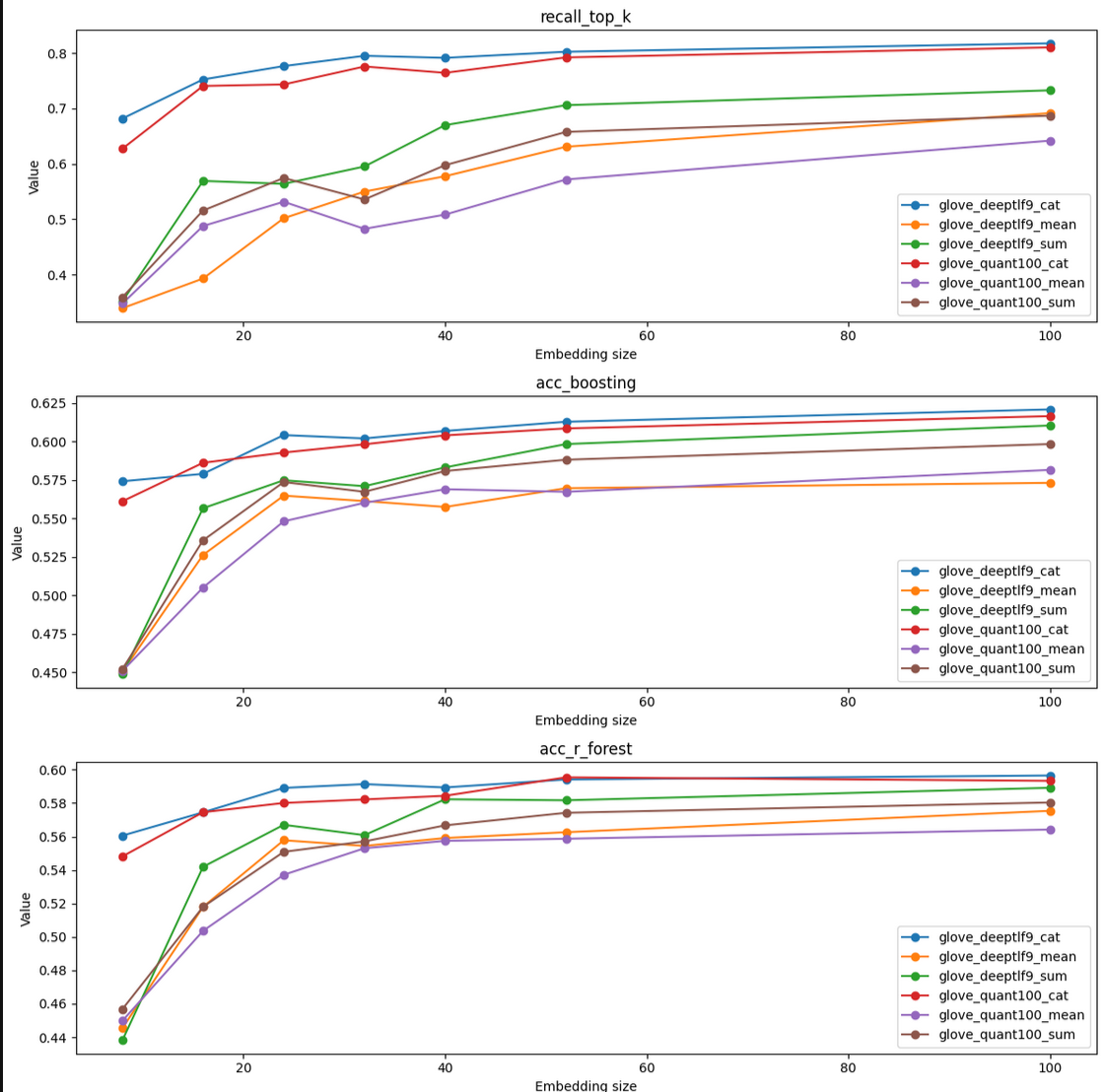
Gender



По результатам видно, что хотя в частных случаях (для агрегации cat) glove-based подход показывает лучшие результаты, качество моделей очень сильно падает для агрегаций mean и sum, ценность использования которых заключается в малом размере итоговых представлений, принимающих участие в обучении нейронной сети. Кроме того, glove подход требует предобучения эмбеддингов и использования замороженных вестов в процессе обучения, в то время как NoisyEmbedding встроен в общую архитектуру модели.

#### Эмбеддинги

**Анализ чувствительности модели к размеру эмбеддинга**



Был проведен ряд экспериментов включающих изменение размера эмбеддинга от 8 до 100 для двух лучших моделей glove и их вариаций с разными функциями агрегации эмбеддингов.

В целом, на датасете age\_bins DeepTLF-based модели показывают результат стабильно лучше, при этом Серьезное изменение качества происходит только для SUM и MEAN агрегаций. После увеличения размера эмбеддинга с 40-50 до 100 улучшение качества минимально для всех моделей.

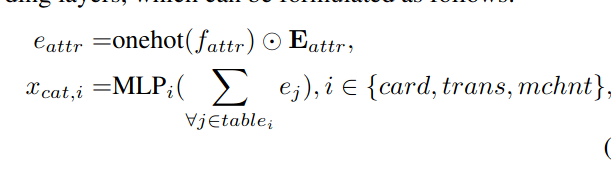
Так же можно отметить, что метрика recall\_top\_k, показывающая непосредственное качество эмбеддинга, как представления транзакции (т е отвечающая критерию близости эмбеддингов транзакций одного клиента друг к другу) – наиболее чувствительна к размеру эмбеддинга для всех типов агрегаций.

**Альтернативный эмбеддинг**

Проведены эксперименты с альтернативным подходом к построению эмбеддингов и кодированию признаков. Он оказался неудачным, но я все же решил включить его в отчет.

В качестве основы метода брался подход из статьи https://arxiv.org/pdf/2412.18287

В чистом виде категориальные признаки кодировались следующим образом:



Где размерность эмбеддинга = размерности всего массива признаков (например, у нас 2 категориальных, один числовой – значит размерность эмбеддинга – 3). Далее x\_cat\_i суммировались друг с другом и с базовым массивом признаков (включающим как числовые, так и категориальные), после чего сумма передавалась в seq\_encoder.

Результаты экспериментов на датасете age\_bins

Я попробовал ряд своих модификаций, совместив идеи из статьи с элементами предыдущих подходов:

1) original no-disc: подход из статьи в числом виде

2) original DeepTLF/Quant100: дискретизация числовых признаков + подход из статьи (т е nn.Embedding + MLP → сумма + сумма с исходным массивом)

в classic экспериментах я пробовал брать эмбеддинг побольше и просто использовать архитектуру embedding + MLP в рамках старых архитектур trx\_encoder

3) classic no disc cat: числовые признаки не дискретизируются, категориальные преобразуются через embedding + MLP блок, после чего конкатинируются друг с другом и с числовыми

4) classic DeepTLF/Quant100 cat и sum: числовые признаки дискретизируются и вместе с категориальными преобразуются через embedding + MLP блок, после чего конкатинируются или суммируются друг с другом.