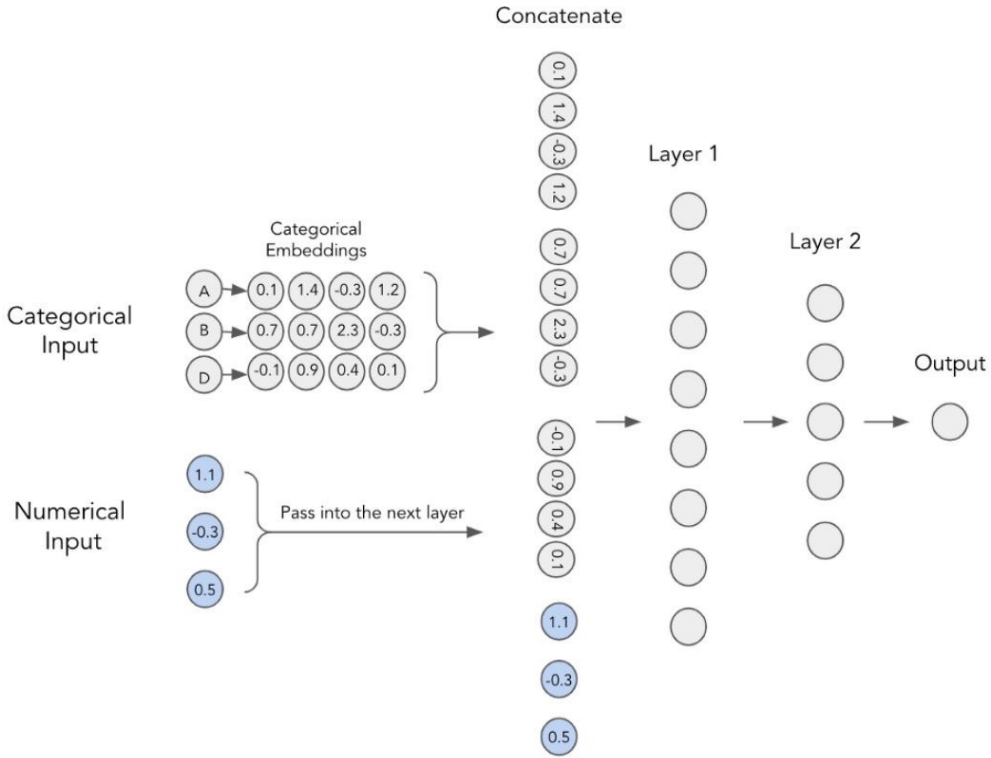
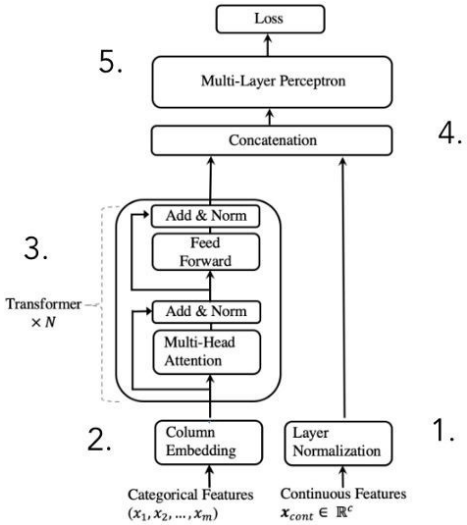
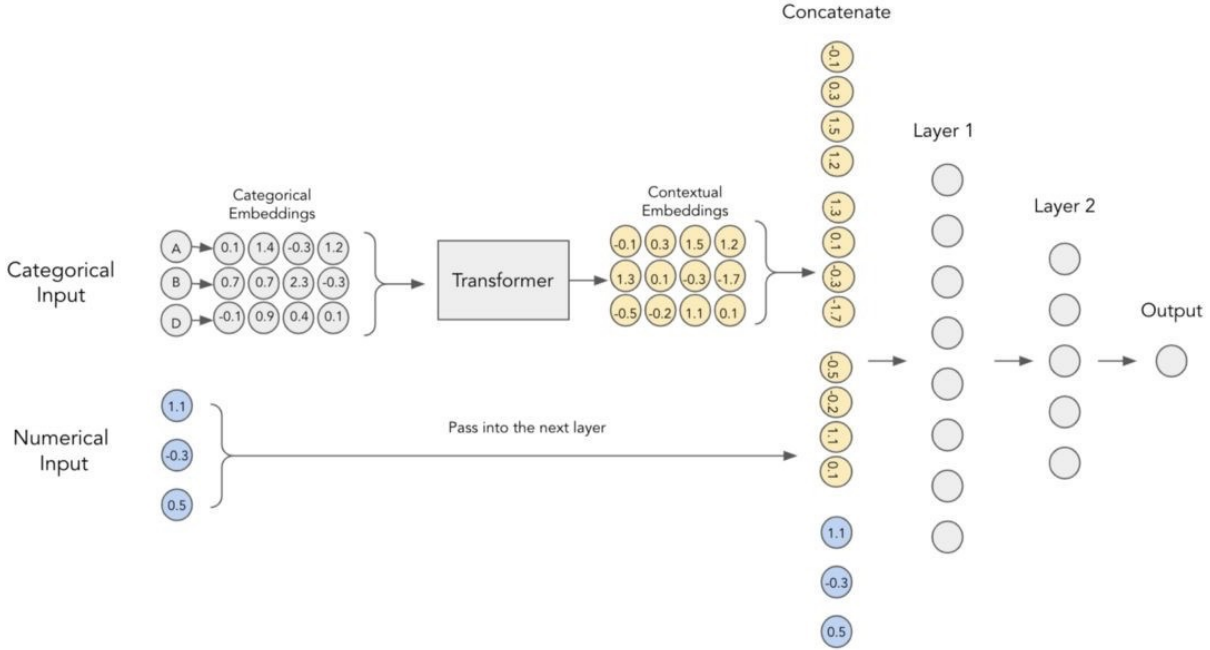
On Embeddings for Numerical Features in Tabular Deep Learning

Типичная MLP модель



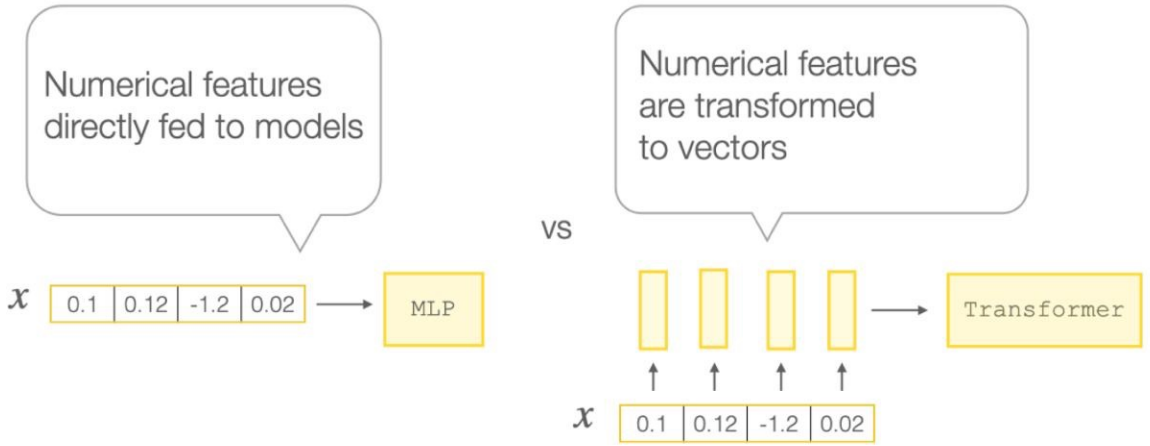
Типичная модель с трансформером





Более современная модель с трансформером

Преобразование числа в вектор обычно было очень простым (линейным или около того) и ему не уделялось внимания в статьях



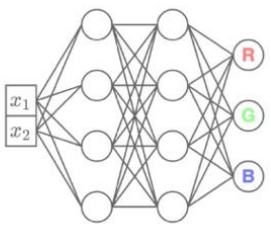
Что предлагается в статье

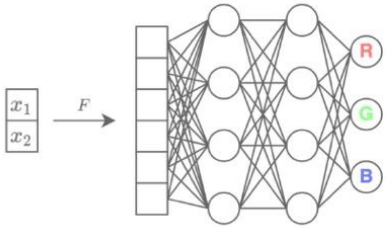
- Доказать, что способ преобразования числа в вектор имеет значение
- Доказать, что это может сильно влиять на качество итоговой модели
- Сравнить MLP и Transformer подходы с GBDT* подходами

* GBDT – Gradient Boosting Decision Tree

Почему способ представления имеет значение

Обучаем модель по координатам пикселя предсказывать его цвет



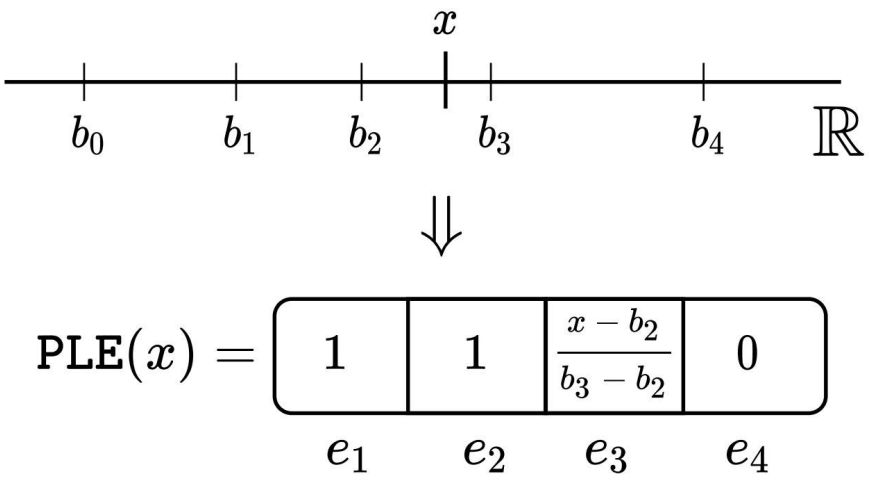


The original image

Inputs are raw scalar coordinates

Inputs are Fourier features

PLE for numerical features (MLP)



PLE (peicewise linear encoding)

PLE for numerical features (Transformer)

Добавляем некоторую позиционную компоненту, необходимую трансформеру - взвешиваем каждую позицию в векторе

$$f_i(x) = v_0 + \sum_{t=1}^{T} e_t \cdot v_t = \texttt{Linear}(\texttt{PLE}(x))$$

Как выбирать границы бинов

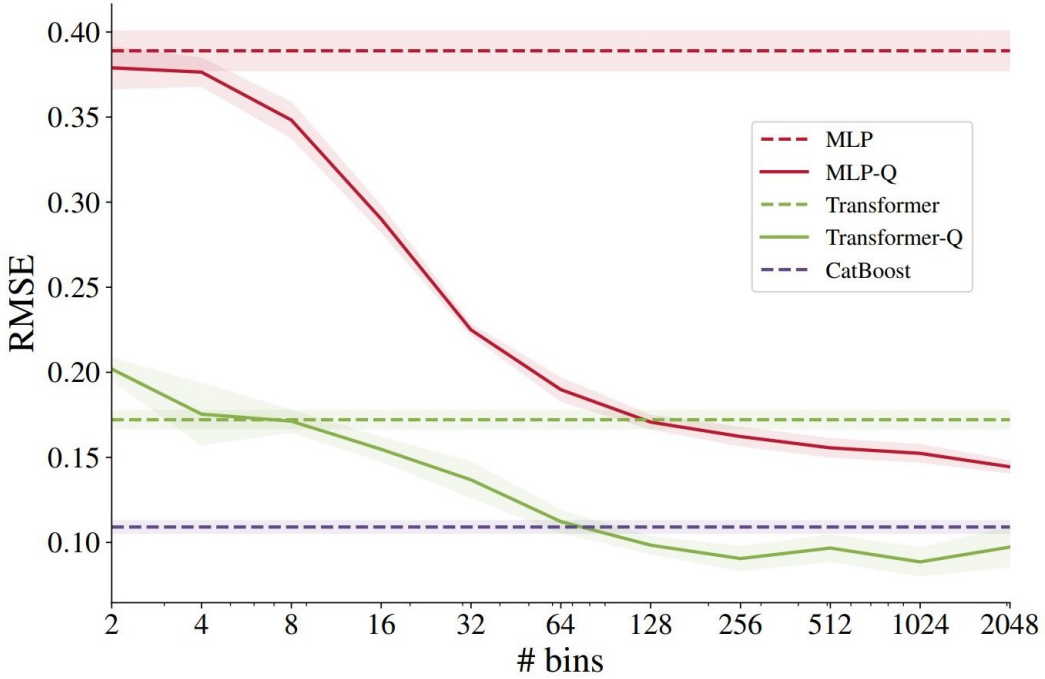
1. Как квантили из обучающего набора данных
2. Target-aware bins с помощью алгоритма, похожего на решающее дерево над 1 фактором

In a nutshell, for each feature, we recursively split its value range in a greedy manner using target as guidance, which is equivalent to building a decision tree (which uses for growing only this one feature and the target) and treating the regions corresponding to its leaves as the bins for PLE

Наглядные результаты на синтетическом датасете

Synthetic GBDT-friendly dataset – «This task turns out to be easy for GBDT, but hard for traditional DL models (Gorishniy et al., 2021).»

Этот слайд показывает, что профит от такого подхода может существовать для некоторых данных

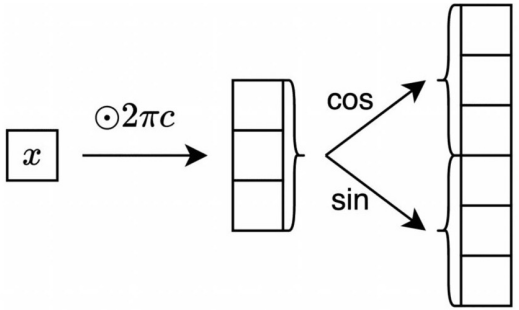


Периодические представления

$$f_i(x) = \texttt{Periodic}(x) = \texttt{concat}[\sin(v),\ \cos(v)],$$
$$v = [2\pi c_1 x,\ \ldots,\ 2\pi c_k x]$$

c_i взяты из N(0, σ) и являются обучаемыми, σ – гиперпараметр

Еще авторы предлагают ReLU пробовать сверху накинуть, потому что это «is a natural approach»



Почему и sin, и cos

Чтобы в рамках признака оставалось линейное преобразование
Подробнее об этом почитать можно тут в контексте позиционных ембедингов - https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

For every sine-cosine pair corresponding to frequency ω_k, there is a linear transformation $M \in \mathbb{R}^{2\times 2}$ (independent of t) where the following equation holds:

$$M.\begin{bmatrix}\sin(\omega_k.t)\\ \cos(\omega_k.t)\end{bmatrix} = \begin{bmatrix}\sin(\omega_k.(t+\phi))\\ \cos(\omega_k.(t+\phi))\end{bmatrix}$$

Пояснения для понимания бенчмарков

Name	Embedding function (f_i)
L	Linear
LR	ReLU ∘ Linear
LRLR	ReLU ∘ Linear ∘ ReLU ∘ Linear
Q	PLE_q
Q-L	Linear ∘ PLE_q
Q-LR	ReLU ∘ Linear ∘ PLE_q
Q-LRLR	ReLU ∘ Linear ∘ ReLU ∘ Linear ∘ PLE_q
T	PLE_t
T-L	Linear ∘ PLE_t
T-LR	ReLU ∘ Linear ∘ PLE_t
T-LRLR	ReLU ∘ Linear ∘ ReLU ∘ Linear ∘ PLE_t
P	Periodic
PL	Linear ∘ Periodic
PLR	ReLU ∘ Linear ∘ Periodic
PLRLR	ReLU ∘ Linear ∘ ReLU ∘ Linear ∘ Periodic
AutoDis	Linear ∘ SoftMax ∘ Linear_- ∘ LReLU ∘ Linear_-

PLE_q - PLE quantiles

PLE_t - PLE target aware bins

Бенчмарки PLE

	GE ↑	CH ↑	EY ↑	CA ↓	HO ↓	AD ↑	OT ↑	HI ↑	FB ↓	SA ↑	CO ↑	MI ↓
MLP	0.632	0.856	0.615	0.495	3.204	0.854	0.818	0.720	5.686	0.912	0.964	**0.747**
MLP-Q	**0.653**	0.854	0.604	0.464	**3.163**	0.859	0.816	0.721	5.766	0.922	0.968	0.750
MLP-T	**0.647**	**0.861**	0.682	0.447	**3.149**	0.864	**0.821**	0.720	5.577	0.923	0.967	0.749
MLP-Q-LR	**0.646**	0.857	**0.693**	0.455	**3.184**	0.863	0.811	0.720	**5.394**	0.923	**0.969**	**0.747**
MLP-T-LR	0.640	**0.861**	**0.685**	**0.439**	3.207	**0.868**	0.818	**0.724**	**5.508**	**0.924**	**0.968**	0.747
Transformer-L	0.632	0.860	0.731	0.465	3.239	0.858	**0.817**	0.725	**5.602**	0.924	0.971	**0.746**
Transformer-Q-L	**0.659**	0.856	0.753	0.451	3.319	0.867	0.812	**0.729**	5.741	**0.924**	**0.973**	0.747
Transformer-T-L	**0.663**	**0.861**	**0.775**	0.454	**3.197**	**0.871**	**0.817**	0.726	5.803	**0.924**	**0.974**	0.747
Transformer-Q-LR	**0.659**	0.857	**0.796**	0.448	3.270	0.867	0.812	0.723	5.683	0.923	0.972	0.748
Transformer-T-LR	**0.665**	0.860	**0.789**	**0.442**	**3.219**	0.870	**0.818**	**0.729**	5.699	**0.924**	0.973	0.747

Бенчмарки периодических представлений

	GE ↑	CH ↑	EY ↑	CA ↓	HO ↓	AD ↑	OT ↑	HI ↑	FB ↓	SA ↑	CO ↑	MI ↓
MLP	0.632	0.856	0.615	0.495	3.204	0.854	**0.818**	0.720	5.686	0.912	0.964	0.747
MLP-P	0.631	**0.860**	0.701	0.489	3.129	0.869	0.807	0.723	5.845	0.923	0.968	0.747
MLP-PL	0.641	**0.859**	0.866	**0.467**	3.113	0.868	**0.819**	0.727	**5.530**	0.924	**0.969**	0.746
MLP-PLR	**0.674**	**0.857**	**0.920**	**0.467**	**3.050**	**0.870**	**0.819**	**0.728**	**5.525**	**0.924**	**0.970**	**0.746**
Transformer-L	**0.632**	0.860	0.731	**0.465**	3.239	0.858	**0.817**	0.725	**5.602**	**0.924**	**0.971**	**0.746**
Transformer-PLR	**0.646**	**0.863**	**0.940**	**0.464**	**3.162**	**0.870**	0.814	**0.730**	5.760	**0.924**	**0.972**	**0.746**

В сравнении с GBDT

	GE ↑	CH ↑	EY ↑	CA ↓	HO ↓	AD ↑	OT ↑	HI ↑	FB ↓	SA ↑	CO ↑	MI ↓	Avg. Rank
CatBoost	0.692	0.861	0.757	0.430	3.093	0.873	0.825	0.727	5.226	0.924	0.967	**0.741**	6.8 ± 4.9
XGBoost	0.683	0.859	0.738	0.434	3.152	**0.875**	0.827	0.726	5.338	0.919	0.969	0.742	9.0 ± 5.7
MLP	0.665	0.856	0.637	0.486	3.109	0.856	0.822	0.727	5.616	0.913	0.968	0.746	15.6 ± 2.4
MLP-LR	0.679	0.861	0.694	0.463	3.012	0.859	0.826	0.731	5.477	0.924	0.972	0.744	10.2 ± 4.4
MLP-Q-LR	0.682	0.859	0.732	0.433	3.080	0.867	0.818	0.724	**5.144**	0.924	0.974	0.745	10.7 ± 4.6
MLP-T-LR	0.673	0.861	0.729	0.435	3.099	0.870	0.821	0.727	5.409	0.924	0.973	0.746	10.3 ± 3.8
MLP-PLR	**0.700**	0.858	0.968	0.453	**2.975**	0.874	**0.830**	**0.734**	5.388	**0.924**	0.975	0.743	4.9 ± 4.8
ResNet	0.690	0.861	0.667	0.483	3.081	0.856	0.821	0.734	5.482	0.918	0.968	0.745	12.1 ± 4.7
ResNet-LR	0.672	0.862	0.735	0.450	2.992	0.859	0.822	0.733	5.415	0.923	0.971	0.743	9.8 ± 4.3
ResNet-Q-LR	0.674	0.859	0.794	0.427	3.066	0.868	0.815	0.729	5.309	0.923	0.976	0.746	9.2 ± 4.8
ResNet-T-LR	0.683	0.862	0.817	**0.425**	3.030	0.872	0.822	0.731	5.471	0.923	0.975	0.744	7.8 ± 3.6
ResNet-PLR	0.691	0.861	0.925	0.443	3.040	**0.874**	0.825	0.734	5.400	0.924	0.975	0.743	5.2 ± 2.3
Transformer-L	0.668	0.861	0.769	0.455	3.188	0.860	0.824	0.727	5.434	0.924	0.973	0.743	10.6 ± 3.3
Transformer-LR	0.666	0.861	0.776	0.446	3.193	0.861	0.824	0.733	5.430	0.924	0.973	0.743	9.4 ± 4.1
Transformer-Q-LR	0.690	0.857	0.842	**0.425**	3.143	0.868	0.818	0.726	5.471	**0.924**	0.975	0.744	8.5 ± 5.5
Transformer-T-LR	0.686	0.862	0.833	**0.423**	3.149	0.871	0.823	0.733	5.515	0.924	**0.976**	0.744	7.2 ± 4.6
Transformer-PLR	0.686	**0.864**	**0.977**	0.449	3.091	0.873	0.823	0.734	5.581	**0.924**	0.975	0.743	6.0 ± 4.5

Выводы

- MLP и трансформеры могут работать лучше за счет ембедингов чисел
- За счет этого могут сравниться в результатах с GBDT подходами для задач, для которых это было недостижимо раньше