Лекция 25
Нейросетевые методы для табличных данных

Е. А. Соколов
ФКН ВШЭ

1 Введение

В настоящее время нейронные сети показывают state-of-the-art результаты в задачах компьютерного зрения, обработки текстов и многих других. Возникает вопрос, можно ли как-нибудь применить эти методы для уже хорошо нам знакомых табличных данных, или уже на них лучше всего работают классические методы (градиентный бустинг, например)? Если нам удастся придумать нейросетевую архитектуру для табличных данных, работающую хотя бы так же хорошо, как бустинг, то это уже отлично — мы дополнительно получим преимущества в виде масштабируемости обучения и возможности применения специфичных для нейронных сетей методов (например, состязательных атак).

Inductive bias. Известно, что любая непрерывная функция может быть с любой точностью аппроксимирована достаточно большой полносвязной нейронной сетью. Однако, на практике это не работает, и приходится вносить в архитектуру т.н. *inductive bias* — некоторые «смещения», учитывающие особенности наших данных. Например, для изображений это могут быть свертки, а для текстов — рекуррентные сети или трансформеры. А какой inductive bias подойдет для таблиц? Чтобы это понять, рассмотрим преимущества градиентного бустинга:

- Учитывает гетерогенность (столбцы в данных имеют различные распределения);
- Моделирование зависимости ответа от комбинаций признаков в небольшом количестве;
- Контролируемость переобучения.

По всей видимости, в табличных данных чаще всего наблюдается зависимость ответа лишь от небольших комбинаций признаков, в то время как полносвязные архитектуры моделируют зависимость от всевозможных их комбинаций.

Рассмотрим некоторые из существующих нейросетевых методов для табличных данных. В каких-то из них предпринимаются попытки перенести описанные выше преимущества бустинга на нейронные сети и получить лучшее качество, а какие-то просто сами по себе интересны.

Сразу отметим, что данная область сейчас активно развивается, предлагается много новых методов, однако полностью победить градиентный бустинг такими методами пока не удалось.

2 Deep FM

Для начала рассмотрим модификацию изученной нами ранее модели факторизационной машины, позволяющую находить более тонкие зависимости, по сравнению со стандартной моделью [1]. Последняя имеет следующий вид:

$$a_{FM}(x) = w_0 + \sum_{j=1}^{d} w_j x_j + \sum_{j=1}^{d} \sum_{k=j+1}^{d} \langle v_j, v_k \rangle x_j x_k,$$

где $w_0, w_1, \ldots, w_d \in \mathbb{R}$, $v_1, \ldots, v_d \in \mathbb{R}^r$ — параметры модели, $x = (x_1, \ldots, x_d)$ — признаковое описание объекта.

Здесь предлагается добавить «глубокую» компоненту, которая устроена следующим образом. Пусть у нас имеется m категориальных признаков. Для каждого из их возможных значений будем обучать векторное представление $s_{ij} \in \mathbb{R}^k$. Для заданного объекта рассмотрим его представления для всех категориальных признаков, сконкатенируем их и пропустим через несколько полносвязных слоев. Назовем эту часть модели $a_{DNN}(x)$. Поскольку полносвязных слоев может быть сколь угодно много, то теоретически можно моделировать зависимости любой сложности. Предсказание (для задачи бинарной классификации) рассчитывается по следующей формуле:

$$y \approx \sigma(a_{FM}(x) + a_{DNN}(x)).$$

Хоть эта идея сама по себе может представлять интерес, градиентный бустинг в плане качества показывает себя лучше.

3 AutoInt

Следующая идея заключается в применении механизма *self-attention*. Чтобы это реализовать, нужно как-то научиться строить векторные представления для признаков. В соответствующей статье [2] предлагается делать это следующим образом:

- Если x_j — категориальный, то обучаем свой вектор для каждой категории: $x_j \to v_{x_j}$;
- Если x_j — числовой, то обучим для него вектор $v_j \in \mathbb{R}^k$ и зададим векторное представление как $x_j \to x_j v_j$.

Теперь, имея представления для всех признаков, можно пропустить их через несколько слоев self-attention и в конце через полносвязный слой, получая предсказание.

Минус этого подхода в том, что если у нас, допустим, порядка 10000 признаков, то self-attention будет работать очень долго.

4 NODE

Заметим, что предыдущие методы пытались в каком-то смысле «изобрести велосипед». Действительно, ведь есть отлично работающий с табличными данными

бустинг, и можно попробовать сделать его дифференцируемым, что и предлагается в методе *Neural Oblivious Decision Ensembles (NODE)* [3].

Для начала, напомним, что *oblivious-деревом* называется решающее дерево, у которого все предикаты на одном уровне одинаковые. Запишем формулу для такого дерева (считаем, что все признаки бинаризованны, т.е. $x_1, \ldots, x_d \in \{0, 1\}$):

$$b(x) = \sum_{j_1=0}^{1} \sum_{j_2=0}^{1} \ldots \sum_{j_n=0}^{1} c_{j_1,\ldots,j_n} [x_{i_1} = j_1][x_{i_2} = j_2] \ldots [x_{i_n} = j_n], \tag{4.1}$$

где n — глубина дерева, $c_{j_1,\ldots,j_n}$ — прогноз и $i_1, \ldots, i_n$ — номера признаков, используемых в предикатах дерева. Поскольку в пределах одного уровня все предикаты совпадают, то всего различных признаков в oblivious-дереве может быть ровно n. В формуле (4.1) мы смотрим, какие значения приняли эти признаки и возвращаем соответствующий прогноз.

Предлагается сгладить эту формулу следующим образом. Введем функции

$$f_i(x) = \sum_{j=1}^{d} x_j \left(\text{entmax}(F_{i1}, \ldots, F_{id})\right)_j,$$

где $F_{ij} \in \mathbb{R}$ — обучаемые параметры и entmax — прореживающий аналог функции softmax, т.е. по вектору он возвращает вектор такой же размерности, все компоненты которого неотрицательны, суммируются в 1 и среди которых много нулей [4]. Введем еще один набор функций:

$$p_i(x) = \sigma_\alpha \left(\frac{f_i(x) - b_i}{\tau_i} \right),$$

где $b_i, \tau_i \in \mathbb{R}$ — обучаемые параметры и $\sigma_\alpha(z) = \text{entmax}([z, 0])$.

Сглаженная модель будет иметь вид

$$\tilde{b}(x) = \sum_{j_1=0}^{1} \ldots \sum_{j_n=0}^{1} c_{j_1,\ldots,j_n} (p_1(x))^{j_1} (1 - p_1(x))^{1-j_1} \cdot \ldots \cdot (p_n(x))^{j_n} (1 - p_n(x))^{1-j_n},$$

где $c_{j_1,\ldots,j_n} \in \mathbb{R}$ — обучаемые параметры. Здесь мы на каждом уровне дерева заменяем индикаторы на взвешенную сумму всех признаков, причем разреженную. Полученная функция не только является гладкой, но также моделирует зависимость ответа от небольшого числа признаков, чего мы и хотели в начале.

Отметим также, что модель получилась довольно сложной, из-за чего обучение может проходить долго. Вместе с тем, если правильно построить композицию таких дифференцируемых деревьев, то в некоторых случаях это может выдавать качество лучше градиентного бустинга.

5 TabNet

Также стоит упомянуть еще одну интересную идею: использовать метод self-supervision для табличных данных. Если у нас есть некоторый набор неразмеченных

данных, то можно «скрывать» от модели часть информации в них и обучать ее восстанавливать эти пропуски. В случае с изображениями, известно, что такой подход и дальнейшее дообучение на сравнительно небольшой выборке может давать весьма впечатляющие результаты.

Пусть имеется большой набор неразмеченных табличных данных и некоторая нейросетевая архитектура $a(x)$. Предлагается достроить ее, чтобы она умела восстанавливать пропуски, и далее обучать модель на нашей таблице, в которой случайным образом удалены какие-то ячейки. В статье [5] утверждается, что такое предобучение очень помогает повысить качество модели.

Список литературы

[1] *Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, Xiuqiang He* (2017). DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. // arXiv:1703.04247 [cs.IR]

[2] *Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, Jian Tang* (2018). AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. // arXiv:1810.11921v2 [cs.IR]

[3] *Sergei Popov, Stanislav Morozov, Artem Babenko* (2019). Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data. // arXiv:1909.06312v2 [cs.LG]

[4] *Ben Peters, Vlad Niculae, André F.T. Martins* (2019). Sparse Sequence-to-Sequence Models. // In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 1504–1519. https://aclanthology.org/P19-1146/

[5] *Sercan O. Arik, Tomas Pfister* (2019). TabNet: Attentive Interpretable Tabular Learning. // arXiv:1908.07442 [cs.LG]