CatBoost介绍

Categorical+Boosting

- 如何处理类别特征
- Prediction shift 问题
- 代码及主要参数
- 与XgBoost LightGBM对比

传统方法：

- 序列编码（ordinal encoding）

一般处理类别间具有大小关系的数据，例如期末成绩的 [A, B, C, D] 四挡可以直接转化为 [0, 1, 2, 3]。在转化后，依然保持类别之间的顺序关系。

- 独热编码（one-hot encoding）

一般处理非类别有序的类别型变量，如血型这样的类别特征时，如果将 [A, B, AB, O] 直接编码成 [1, 2, 3, 4]，显然A与B和B与AB之间的距离，并不具有相同的含义，甚至是完全抽象的无法理解的意义，此时，序列编码就不适用了。

CatBoost:基于TS的方法

- Target statistics

As discussed in Section 3.1, an effective and efficient way to deal with a categorical feature i is to substitute the category x_k^i of k-th training example with *one* numeric feature equal to some *target statistic* (TS) $\hat{x}_k^i$. Commonly, it estimates the expected target y conditioned by the category:

$$\hat{x}_k^i \approx \mathbb{E}(y \mid x^i = x_k^i)$$

- 存在问题

（1）噪声和低频率数据对于数据分布的影响较大。

（2）当特征对应的样本数量较少时，这种估计是不准确的。

改进：

- Greedy TS

Greedy TS A straightforward approach is to estimate $\mathbb{E}(y \mid x^i = x^i_k)$ as the average value of y over the training examples with the same category x^i_k [25]. This estimate is noisy for low-frequency categories, and one usually smoothes it by some prior p:

$$\hat{x}^i_k = \frac{\sum_{j=1}^{n} \mathbb{1}_{\{x^i_j = x^i_k\}} \cdot y_j + a\,p}{\sum_{j=1}^{n} \mathbb{1}_{\{x^i_j = x^i_k\}} + a}$$

- 存在问题

Conditional Shift:当特征对应的样本数量较少时，这种估计是不准确的。

Conditional Shift:

- 假设某一个特征向量的第i个特征是一个类别特征，每一个样本的这个类别特征都是不同的， 记每个样本为A，考虑一个二分类问题，其中：

$$\mathrm{P}(y=1 \mid x^i=A)=0.5$$

- 在训练集中：

$$\hat{x}_k^i = \frac{y_k+ap}{1+a}$$

- 对该特征的划分：

$$t = \frac{0.5+ap}{1+a}$$

改进：

- 添加条件：

P1 $\mathbb{E}(\hat{x}^i \mid y = v) = \mathbb{E}(\hat{x}^i_k \mid y_k = v)$, *where* $(\mathbf{x}_k, y_k)$ *is the* k*-th training example.*

- 如何满足P1：

There are several ways to avoid this conditional shift. Their general idea is to compute the TS for $\mathbf{x}_k$ on a subset of examples $\mathcal{D}_k \subset \mathcal{D} \setminus \{\mathbf{x}_k\}$ excluding $\mathbf{x}_k$:

$$\hat{x}^i_k = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x^i_j = x^i_k\}} \cdot y_j + a\,p}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x^i_j = x^i_k\}} + a}$$

改进：

- Holdout TS ：

Holdout TS One way is to partition the training dataset into two parts $\mathcal{D} = \hat{\mathcal{D}}_0 \sqcup \hat{\mathcal{D}}_1$ and use $\mathcal{D}_k = \hat{\mathcal{D}}_0$ for calculating the TS according to (5) and $\hat{\mathcal{D}}_1$ for training (e.g., applied in [8] for Criteo dataset). Though such *holdout* TS satisfies P1, this approach significantly reduces the amount of data used both for training the model and calculating the TS. So, it violates the following desired property:

P2 *Effective usage of all training data for calculating TS features and for learning a model.*

- 存在问题：

这样处理能够满足同分布的问题，但是却大大减少了训练样本的数量。

改进：

- Leave-one-out TS：

Leave-one-out TS At first glance, a *leave-one-out* technique might work well: take $\mathcal{D}_k = \mathcal{D} \setminus \mathbf{x}_k$ for training examples $\mathbf{x}_k$ and $\mathcal{D}_k = \mathcal{D}$ for test ones [31]. Surprisingly, it does not prevent target leakage. Indeed, consider a constant categorical feature: $x_k^i = A$ for all examples. Let n^+ be the number of examples with $y = 1$, then $\hat{x}_k^i = \frac{n^+ - y_k + a\,p}{n-1+a}$ and one can perfectly classify the training dataset by making a split with threshold $t = \frac{n^+ - 0.5 + a\,p}{n-1+a}$.

- 存在问题：
  - $\hat{x}_k^i = \frac{n^+ - y_k + a\,p}{n-1+a}$
  - 对于测试样本：$\hat{x}^i = \frac{n^+ + a\,p}{n+a}$

此时，同样可以用阈值$\hat{x}_k^i = \frac{n^+ - 0.5 + a\,p}{n-1+a}$将训练集完美的分类

Ordered TS ：

- 产生一个随机排列顺序σ并对数据集进行编号
- 对于训练样本：$D_k = \{X_j: \sigma(j) < \sigma(k)\}$
- 对于测试样本： $D_k = \mathrm{D}$
- 根据带先验概率的Greedy TS计算：

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a\, p}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

Ordered TS ： logloss / zero-one loss

	Greedy	Holdout	Leave-one-out
Adult	+1.1% / +0.8%	+2.1% / +2.0%	+5.5% / +3.7%
Amazon	+40% / +32%	+8.3% / +8.3%	+4.5% / +5.6%
Click	+13% / +6.7%	+1.5% / +0.5%	+2.7% / +0.9%
Appetency	+24% / +0.7%	+1.6% / -0.5%	+8.5% / +0.7%
Churn	+12% / +2.1%	+0.9% / +1.3%	+1.6% / +1.8%
Internet	+33% / +22%	+2.6% / +1.8%	+27% / +19%
Upselling	+57% / +50%	+1.6% / +0.9%	+3.9% / +2.9%
Kick	+22% / +28%	+1.3% / +0.32%	+3.7% / +3.3%

Prediction shift：

- 对于梯度提升：$F^t = F^{t-1} + \alpha^t h^t,\ h^t = \underset{h \in H}{\operatorname{argmin}} \mathcal{L}(F^{t-1} + h)$
- $g^t(X, y) := \frac{\partial L(y,s)}{\partial s}\Big|_{s=F^{t-1}(x)}$
- $\hat{h}^t = \underset{h \in H}{\operatorname{argmin}} \mathbb{E}\left(-g^t(X, y) - h(X)\right)^2$
- $h^t = \underset{h \in H}{\operatorname{argmin}} \frac{1}{n} \sum_{k=1}^{n} \left(-g^t(X_k, y_k) - h(X_k)\right)^2$

Prediction shift：

假设以下边界条件:

- 损失函数：$L(y, \hat{y}) = (y - \hat{y})^2$
- 两个相互独立的特征x^1, x^2，随机变量，符合伯努利分布，先验概率$p = 1/2$
- 目标函数：$y = f^*(x) = c_1\, x^1 + c_2\, x^2$
- 梯度提升迭代次数为2
- 树深度为1
- 学习率：$\alpha = 1$

最后得到的模型为：$F = F^2 = h^1 + h^2$，其中h^1, h^2分别基于x^1和 x^2。

Prediction shift：

区分数据集是否独立，我们有以下两个推论:

- 如果使用了规模为n的两个独立数据集$\mathcal{D}_1$和 $\mathcal{D}_2$来分别估算h^1 和h^2，则对于任意$x \in \{0,1\}^2$，有：$\mathbb{E}_{\mathcal{D}_1,\mathcal{D}_2} F^2(X) = f^*(X) + O(\frac{1}{2^n})$
- 如 果 使 用 了 相 同 的 数 据 集 $\mathcal{D}$ 来 估 算 h^1 和 h^2 ， 则 有 ：$\mathbb{E}_{\mathcal{D}_1,\mathcal{D}_2} F^2(X) = f^*(X) + O(\frac{1}{2^n}) - \frac{1}{n-1} c_2(x^2 - \frac{1}{2})$

偏差部分与数据集的规模成反比，与映射关系$f^*(x)$也有关系。

ordered boosting：

假设用I棵树来学习一个模型，为了使$\mathrm{r}^{I-1}(X_k,Y_k)$ 无偏。需要确保模型F^{I-1}的训练没有用到样本X^k

对于每一个样本维持一个模型$\mathrm{M}_{r,j}$，其中$\mathrm{M}_{r,j}(i)$表示基于在序列σ_r当中的前j个样本学习得到的模型对于第i个样本的预测，在算法的每一次迭代t，我们从$\{\sigma_1,\sigma_2,\ldots,\sigma_s\}$当中抽样一个随机序列$\sigma_\mathrm{r}$，并基于此构建第$t$步的学习树。最后基于$\mathrm{M}_{r,j}(i)$计算相应的梯度。
注意：每此迭代时抽取的随机序列σ_r 与用于Greedy TS的随机序列保持一致。

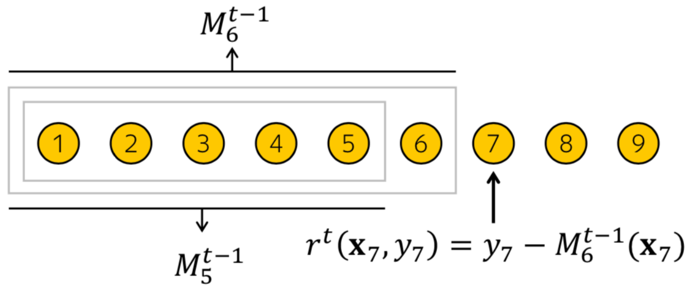

Figure 1: Ordered boosting principle, examples are ordered according to σ.

ordered boosting：

- 样本足

--boostin

Table 3: Plain boosting mode: logloss, zero-one loss and their change relative to Ordered boosting mode.

	Logloss	Zero-one loss
Adult	0.272 (+1.1%)	0.127 (-0.1%)
Amazon	0.139 (-0.6%)	0.044 (-1.5%)
Click	0.392 (-0.05%)	0.156 (+0.19%)
Epsilon	0.266 (+0.6%)	0.110 (+0.9%)
Appetency	0.072 (+0.5%)	0.018 (+1.5%)
Churn	0.232 (-0.06%)	0.072 (-0.17%)
Internet	0.217 (+3.9%)	0.099 (+5.4%)
Upselling	0.166 (+0.1%)	0.049 (+0.4%)
Kick	0.285 (-0.2%)	0.095 (-0.1%)

, but it may be slower

与其他Boosting算法对比：

	CatBoost	LightGBM	XGBoost
Adult	**0.270 / 0.127**	+2.4% / +1.9%	+2.2% / +1.0%
Amazon	**0.139 / 0.044**	+17% / +21%	+17% / +21%
Click	**0.392 / 0.156**	+1.2% / +1.2%	+1.2% / +1.2%
Epsilon	**0.265 / 0.109**	+1.5% / +4.1%	+11% / +12%
Appetency	**0.072 / 0.018**	+0.4% / +0.2%	+0.4% / +0.7%
Churn	**0.232 / 0.072**	+0.1% / +0.6%	+0.5% / +1.6%
Internet	**0.209 / 0.094**	+6.8% / +8.6%	+7.9% / +8.0%
Upselling	**0.166 / 0.049**	+0.3% / +0.1%	+0.04% / +0.3%
Kick	**0.286 / 0.095**	+3.5% / +4.4%	+3.2% / +4.1%

与其他Boosting算法对比：