DeepGBM KDD-2019

A Deep Learning Framework Distilled by GBDT for Online Prediction Tasks

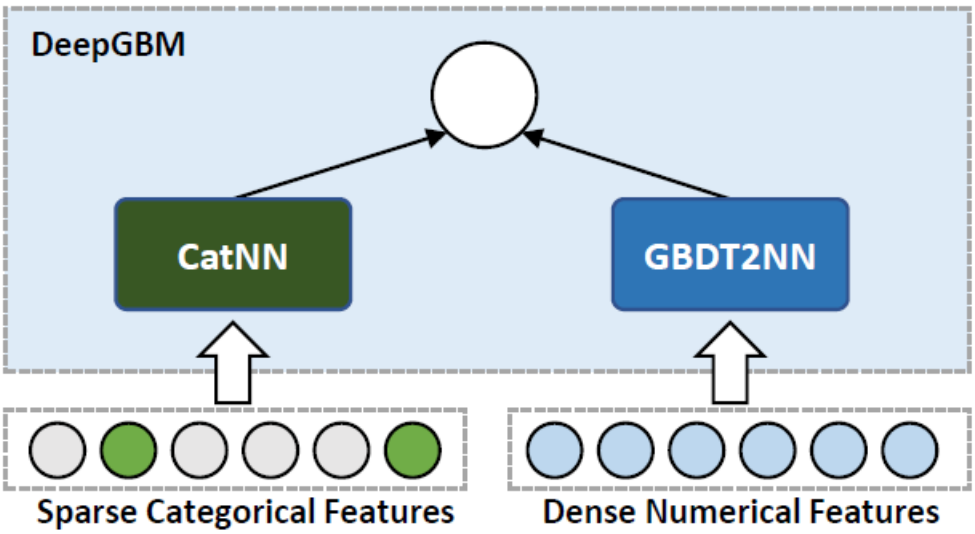
论文链接

背景

GBDT:无法处理动态数据生成，即无法在线实时更新。无法处理稀疏数据。

NN: 无法处理稠密数据特征

提出的新框架——DeepGBM



CatNN:处理稀疏特征

GBDT2NN：处理稠密的数值特征

DeepGBM不仅可以同时处理稀疏和稠密的数字特征，而且可以在线更新。

1.简介

在线预测可以用在很多工业应用上，点击率预测，web中的内容排序，推荐系统中的内容优化，交通规划中的行程时间估计。

一个典型的在线预测包括两种：表格数据空间tabular input space和在线数据online data generation两个特征。表格数据空间意味着输入特征包含离散和连续特征。

对于一个在线预测模型，有两个挑战：（1）如何学习具有表格输入空间的有效模型。（2）可以接受在线生成的数据。GBDT和NN可以分别解决上边两个问题，但是不能同时解决，他们有各自的优缺点。

GBDT的主要**优势**可以处理稠密数值特征，可以迭代选取最大信息增益最大的特征来构建树，GBDT可以自动选择有用的数值特征去训练目标，广泛应用在预测，搜索排序以及其他的预测任务。**弱点**（1）在在线模式下很难更新模型，只能从头开始训练，效率非常低。（2）稀疏分类上效率低下。比如在one-hot编码上。虽然有些编码可以让数据变得稠密，但是这些会损害raw信息，以至于难以用于分类问题上。

神经网络**优势**：可以用于在线大规模大规模的学习，以及通过embedding可以对稀疏特征也有学习能力，主要在于对于稠密数值特征的不足，比如全连接神经网络，用于稠密数值特征时，会构成非常复杂的优化超平面，很难达到局部最优，导致结果不理想。在稠密数值特征的分类中，不如GBDT。

Table 1: Comparison over different models.

	NN	GBDT	GBDT+NN	**DeepGBM**
Sparse Categorical Feature	✓	✗	✓	✓
Dense Numerical Feature	✗	✓	✓	✓
Online update & Large-scale data	✓	✗	✗	✓

如果将NN和GBDT结合，是非常有益的。也能解决两大挑战。

文章提出的DeepGBM,CatNN是一个输入分类特征的NN，GBDT2NN是输入为数值特征的NN，为了充分利用GBDT在处理数值方面的优势，GBDT2NN试图将GBDT学到的知识提炼成NN建模过程。具体来说，为了提高知识提取的有效性，GBDT2NN不仅传递了预先训练的GBDT的输出知识，而且还融合了所得到的树结构所隐含的特征重要性和数据划分知识(特征选择和特征生成)。这样，在达到与GBDT相当的性能的同时，采用神经网络结构的GBDT2NN在面对在线数据生成时，可以很容易地通过不断涌现的数据进行更新。

2.DeepGBM

介绍如何整合NN和GBDT

2.1 CatNN 稀疏特征

FM+Deep

2.2 GBDT2NN

作用，将GBDT树提取为神经网络模型，从单树推导到多树。

1. 由于决策树不会利用所有的特征进行判断，所以NN也只使用有用的特征作为输入
2. 样本经过决策树判断，最终会有一个输出叶子索引,NN最终会拟合这样的映射机制

2.2.1单树知识提取

树模型很多可以转换为神经网络，比如树的特征选择和特征提取能力，树结构所隐含的数据划分能力。

树的特征选择能力（Tree-Selected Features）

树并不是使用所有输入特征，会根据统计信息自动选择最有用的输入特征去训练模型，所以，根据树选择出来的特征作为NN的输入，以提高NN的效率，定义$\mathbb{I}^t$为树t 中使用的特征的索引。那么用$x[\mathbb{I}^t]$作为神经网络的输入。

树型结构知识(Tree Structure)

从本质上讲，决策树的树结构知识是指如何将数据划分成多个不重叠的区域（叶），即将数据聚类成不同的类，同一叶中的数据属于同一类。这种树结构很难直接转化为神经网络，因为它们从结构上有着明显的区别。所幸的是，神经网络已经被证明足以逼近任何函数，所以可以使用神经网络模型来逼近树结构的函数输出，并实现结构知识的蒸馏。因此，如图2所示，可以使用神经网络来拟合树生成的聚类结果，从而使神经网络逼近决策树的结构函数。

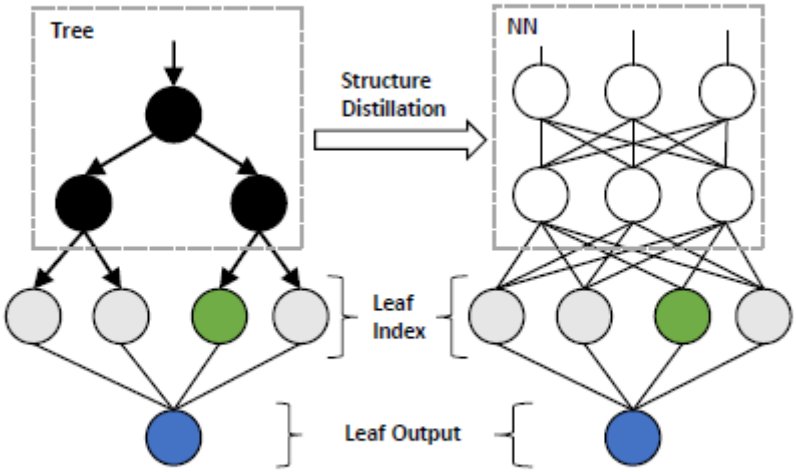


Figure 2: Tree structure distillation by leaf index. NN will approximate the tree structure by fitting its leaf index.

形式上，把树t表示为$C^t(x)$的结构函数，它返回样本的输出叶子索引，即树生成的聚类结果。然后，可以使用神经网络模型来逼近结构函数$C^t(\cdot)$，学习过程可以表示为：

$$min_\theta \frac{1}{n}\sum_{i=1}^{n}\mathfrak{L}'(N(x^i[\mathbb{I}^t];\theta), L^{t,i})$$

其中n是训练样本的数目，x^i是第i个训练样本，$L^{t,i}$是样本x^i的树$C^t(x^i)$叶子输出的独热(one-hot)表示，$\mathbb{I}^t$是树t中使用的特征的索引，θ是神经网络模型N的模型参数，可以通过反向传播更新，$\mathfrak{L}'$是交叉熵之类的多分类问题的损失函数。因此，在学习之后，就可以得到一个神经网络模型$N(.;\theta)$。由于神经网络具有很强的表达能力，经过学习的神经网络模型应该能完美地逼近决策树的结构函数。

树叶子值的输出(Tree Outputs).由于在前面的步骤中学习了从树输入到树结构的映射，所以要提取树的输出，只需要知道从树结构到树输出的映射。在决策树中叶子索引有相应的叶子值，因此实际上不需要学习此映射。将树t的叶子值表示为q^t，*那么* q_i^t表示第i个叶子的叶子值。要得到树模型的输出，只需要用$p^t = L^t \times q^t$将L^t*映射到树的值输出。
结合上述的单树蒸馏方法，从t 树蒸馏得到的神经网络的输出可以表示为

$$y^t(x) = N(x[\mathbb{I}^t];\theta) \times q^t$$

2.2.2 多棵树知识提取(Multiple Tree Distillation)

目的：降低叶子个数和神经网络个数，通过一组树经过leaf embedding，再连接NN

叶子嵌入蒸馏(Leaf Embedding Distillation)

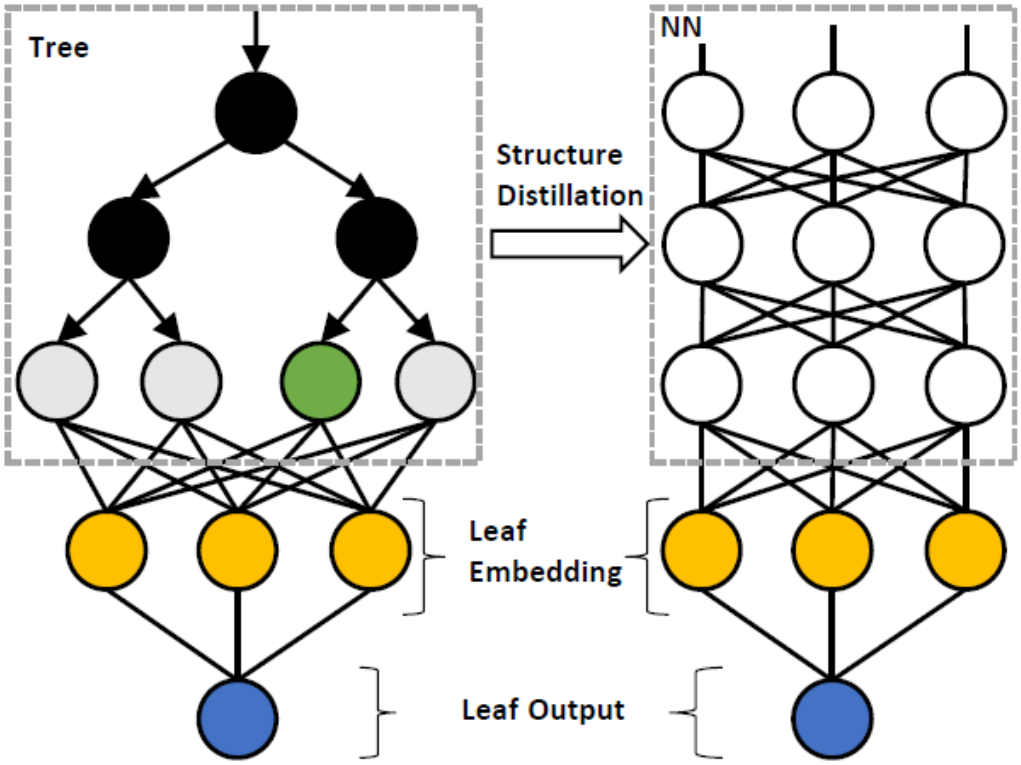


Figure 3: Tree structure distillation by leaf embedding. The leaf index is first transformed to leaf embedding. Then NN will approximate tree structure by fitting the leaf embedding. Since the dimension of leaf embedding can be significantly smaller than the leaf index, this distillation method will be much more efficient.

$$min_{w,w_0,\omega^t} \frac{1}{n} \sum_{i=1}^{n} \mathfrak{L}''(w^T H(L^{t,i};\omega^t) + w_0, p^{t,i}) \tag{7}$$

其中$H^{t,i} = H(L^{t,i};\omega^t)$是以$\omega^t$*为参数的一层全连接网络，主要能把one_hot的输入*$L^{t,i}$, $p^{t,i}$为样本在树中的叶子节点的预测值，$\mathfrak{L}''$是树学习过程中的损失函数，w 和w_0是用于将嵌入映射到叶子节点值的参数。完了之后，可以改用密集嵌入$H^{t,i}$作为目标来逼近树结构的函数，而不是稀疏高维的独热表示L 。这个新的学习过程可以表示为

$$min_\theta \frac{1}{n} \sum_{i=1}^{n} \mathfrak{L}(N(x^i[\mathbb{I}^t];\theta), H^{t,i}) \tag{8}$$

其中$\mathfrak{L}$是拟合密集嵌入的回归损失，如L2损失。由于$H^{t,i}$的维数要比one-hot的L小得多，因此叶节点嵌入蒸馏在多树蒸馏中更为有效。因为它将使用更少的神经网络参数，因此会效率会更高。

树分组法（Tree Grouping）

对树进行分组，比如随机分组、等顺序分组、基于重要性或相似性的分组等，论文采用了随机分组。假设有m棵树，想把它们分成k 组，每组中有s = [m / k] 树,第j 组中的树是$\mathbb{T}_j$，它包含来自**gbdt**的随机**s**棵树。其次，为了从多棵树中提取，可以扩展到多棵树的叶子索引嵌入蒸馏技术。给定一组树$\mathbb{T}$，扩展等式(7)从多个树学习叶子节点的嵌入表示。

$$min_{w,w_0,\omega^{\mathbb{T}}} \frac{1}{n} \sum_{i=1}^{n} \mathfrak{L}''(w^T H(||_{t\in\mathbb{T}}(L^{t,i});\omega^{\mathbb{T}}) + w_0, \sum_{t\in\mathbb{T}} p^{t,i}) \tag{9}$$

其中||(.)是拼接操作(concatenate operation)，$G^{\mathbb{T},i} = H(||_{t\in\mathbb{T}}(L^{t,i});\omega^{\mathbb{T}})$是一个一层全连通网络，它将多个单树叶子索引向量的拼联，转化为$\mathbb{T}$树中的密集嵌入$G^{\mathbb{T},i}$，然后用新的嵌入作为神经网络模型的蒸馏目标，其学习过程可以表示为

$$\mathfrak{L}^{\mathbb{T}} = min_{\theta^{\mathbb{T}}} \frac{1}{n} \sum_{i=1}^{n} \mathfrak{L}(N(x^i[\mathbb{I}^{\mathbb{T}}]), G^{\mathbb{T},i}) \tag{10}$$

其中$\mathbb{I}^{\mathbb{T}}$是树分组$\mathbb{T}$中用到的特征。综上所述，结合上述方法，从树组$\mathbb{T}$中提取神经网络模型的最终输出是

$$y_{\mathbb{T}}(x) = w^T \times N(x[\mathbb{I}^{\mathbb{T}}];\theta^{\mathbb{T}}) + w_0 \tag{11}$$

包含k个树组的**gbdt**模型的输出是

$$y_{GBDT2NN}(x) = \sum_{j=1}^{k} y_{\mathbb{T}_j}(x) \tag{12}$$

综上所述，由于叶子嵌入蒸馏和树分组，**gbdt2nn**可以有效地将**gbdt**中的许多树提取为一个紧凑的神经网络模型。而且除了树的模型值输出，树的特征选择和结构知识也被有效地提取到神经网络模型中。

2.3 DeepGBM训练

2.3.1端到端训练

为了训练deepgbm，首先需要使用离线数据训练**gbdt**模型，然后使用等式（9）得到**gbdt**中树的叶子节点嵌入表示。然后就可以端到端地训练**deepgbm**。将**deepgbm**的输出表示为

$$\hat{y}(x) = \sigma'(w_1 \times y_{GBDT2NN}(x) + w_2 \times y_{Cat}(x)) \tag{13}$$

其中w_1和w_2是用于组合**gbdt2nn**和**catnn**的可训练参数，σ'是输出变换函数，例如用于二进制分类的**sigmoid**。然后，可以使用下面的损失函数进行端到端的训练

$$\mathfrak{L}_{offline} = \alpha\mathfrak{L}''(\hat{y}(x), y) + \beta \sum_{j=1}^{k} \mathfrak{L}^{\mathbb{T}_j} \tag{14}$$

其中y 是样本x的训练目标，$\mathfrak{L}''$是分类任务的交叉熵等相应任务的损失函数，$\mathfrak{L}^{\mathbb{T}}$是树组$\mathbb{T}$的嵌入损失，并在等式(10)中定义，k是树组个数，α和β是预先给定的用于控制端到端损失强度和嵌入损耗的超参数。

2.3.2 在线更新

由于**gbdt**模型是离线训练的，在在线更新中嵌入学习(embedding learning)会影响在线时效性。因此，在在线更新模型的时候不再包含$\mathfrak{L}^{\mathbb{T}}$，在线更新模型的时候，损失函数表示成：

$$\mathfrak{L}_{online} = \mathfrak{L}''(\hat{y}(x), y) \tag{15}$$

它只使用端到端的损失。因此，当使用**deepgbm**在线时，我们只需要新的数据来通过$\mathfrak{L}_{\text{online}}$更新模型，而不需要涉及**gbdt**和从头开始的再训练模型。简而言之，**deepgbm**将非常有效地执行在线任务。此外，它还可以很好地处理稠密的数值特征和稀疏的分类特征。

3.实验结果

Table 3: Offline performance comparison. AUC (higher is better) is used for binary classification tasks, and MSE (lower is better) is used for regression tasks. All experiments are run 5 times with different random seeds, and the *mean* ± *std* results are shown in this table. The top-2 results are marked bold.

Model	Binary Classification						Regression
	Flight	Criteo	Malware	AutoML-1	AutoML-2	AutoML-3	Zillow
LR	0.7234 ±5e-4	0.7839 ±7e-5	0.7048 ±1e-4	0.7278 ±2e-3	0.6524 ±2e-3	0.7366 ±2e-3	0.02268 ±1e-4
FM	0.7381 ±3e-4	0.7875 ±1e-4	0.7147 ±3e-4	0.7310 ±1e-3	0.6546 ±2e-3	0.7425 ±1e-3	0.02315±2e-4
Wide&Deep	0.7353 ±3e-3	0.7962 ±3e-4	0.7339 ±7e-4	0.7409 ±1e-3	0.6615 ±1e-3	0.7503 ±2e-3	0.02304 ±3e-4
DeepFM	0.7469 ±2e-3	0.7932 ±1e-4	0.7307 ±4e-4	0.7400 ±1e-3	0.6577 ±2e-3	0.7482 ±2e-3	0.02346 ±2e-4
PNN	0.7356 ±2e-3	0.7946 ±8e-4	0.7232 ±6e-4	0.7350 ±1e-3	0.6604 ±2e-3	0.7418 ±1e-3	0.02207 ±2e-5
GBDT	0.7605 ±1e-3	0.7982 ±5e-5	0.7374 ±2e-4	0.7525 ±2e-4	0.6844 ±1e-3	0.7644 ±9e-4	0.02193 ±2e-5
DeepGBM (D1)	0.7668 ±5e-4	**0.8038 ±3e-4**	0.7390 ±9e-5	0.7538 ±2e-4	0.6865 ±4e-4	**0.7663 ±3e-4**	0.02204 ±5e-5
DeepGBM (D2)	**0.7816 ±5e-4**	0.8006 ±3e-4	**0.7426 ±5e-5**	**0.7557 ±2e-4**	**0.6873 ±3e-4**	0.7655 ±2e-4	**0.02190 ±2e-5**
DeepGBM	**0.7943 ±2e-3**	**0.8039 ±3e-4**	**0.7434 ±2e-4**	**0.7564 ±1e-4**	**0.6877 ±8e-4**	**0.7664 ±5e-4**	**0.02183 ±3e-5**

参考：

1. https://zhuanlan.zhihu.com/p/99381151
2. 官方代码
3. 论文笔记
4. 读DeepGBM代码