Abstract
In spite of showing unreasonable effectiveness in modalities like Text and Image, Deep Learning has always lagged Gradient Boosting in tabular data - both in popularity and performance. But recently there have been newer models created specifically for tabular data, which is pushing the performance bar. But popularity is still a challenge because there is no easy, ready-to-use library like Sci-Kit Learn for deep learning.

PyTorch Tabular aims to change that by being an easy-to-use and flexible framework which makes using SOTA model architectures in tabular data as easy as Sci-Kit Learn.

Design Principles

- Low Resistance Useability
- Easy Customization
- Scalable and Easy to Deploy

Implemented Models
These are the currently implemented SOTA models in PyTorch Tabular

- Neural Oblivious Decision Trees (NODE) (https://openreview.net/forum?id=r1elu2VtwH)
- AutoInt (https://dl.acm.org/doi/10.1145/3357384.3357925)
- FTTransformer (https://openreview.net/forum?id=l_Q1yrOegLY)

Structure of the Library

- There are 5 config files which drives the whole process. They can be initialized programmatically and through YAML files.
- TabularDataModule encapsulates all the data handling, pre-processing and post-processing required.
- The whole process is orchestrated by TabularModel, which has a training engine and an inference engine.
- All implemented models are pure PyTorch models, or PyTorch Lightning models to be exact.
- Common training, validation, and inference steps required from a model, along with other tasks like loss and metric calculation and logging are implemented in a BaseModel.
- All the implemented models must inherit BaseModel and bare minimum implementation for any new model is the forward pass.
- ExperimentConfig enables easy tracking of experiments, logging of metrics, gradients, etc. to either Tensorboard or Weights & Biases.

Basic Usage

```
data.config: BatchConfig
  target: "target_"
  target should always be a list.
  categorical_config: [all names].

trainer_config: trainerConfig
  batch_size: 32
  num_workers: 0
  gpus: By default use CPU (None).

optimizer_config: optimizerConfig
  model: CategoryEmbeddingModelConfig
  task: classification.

Putting it all together
TabularModel = TabularModel(data_config, trainer_config, optimizer_config).
TabularModel.train().
```

Advanced Usage

- Handle Imbalanced Classification
- YAML Configs & Experiment Tracking
- Categorical Encoder & Feature Extractor
- Learning Rate Schedulers & Custom Optimizers, Losses

```
pip install pytorch_tabular[all]
```