如何插入模型

19373750 杨宜凡

定义模型需要的dataset

这里的dataset需要首先加载原子文件格式的dataset，然后根据模型的需求构造图/邻接矩阵，也需要返回下游任务需要的label等。

已经在libcity.data.dataset.TrafficRepresentationDataset中实现了加载原子文件的操作，具体可以看方法注释。由于每种模型构造图，数据处理等过程都是不一样的，因此可以自定义dataset。需要放在libcity/data/dataset/dataset_subclass下，子类需要继承rafficRepresentationDataset。这里实现了一个例子libcity.data.dataset.dataset_subclass.Node2VecDataset。

模型需要的数据，label，邻接矩阵等都需要在这个类中构造好，通过get_data_feature方法返回一个字典。

```
yyf
def get_data_feature(self):
    """
    返回一个 dict，包含数据集的相关特征

    Returns:
        dict: 包含数据集的相关特征的字典
    """
    return {"adj_mx": self.adj_mx, "num_nodes": self.num_nodes,
            "geo_to_ind": self.geo_to_ind, "ind_to_geo": self.ind_to_geo,
            "label":{"od_matrix_predict":self.od_label,"function_cluster":np.array(self.function)}}
```

定义模型结构

模型的核心结构部分需要放在libcity/model/下，现在有一个例子是libcity/model/region_representation/Node2Vec.py

定义下游任务

下游模型在libcity/evaluator/downstream_models，这里是定义一些通用的下游模型，可以应用于若干具体的下游任务，比如回归模型，KMeans聚类模型等。

目前需要参数指定下游任务和对应的下游模型，使用两个列表传入，列表间需要一一对应。

```
self.evaluate_tasks=self.config.get('evaluate_tasks',["function_cluster"])
self.evaluate_model=self.config.get('evaluate_model',["KmeansModel"])
```

补充config

如果自己定义了dataset需要在libcity/config/data下添加对应的json文件，如果没有额外的配置需要，可以为空，例如libcity/config/data/Node2VecDataset.json。

evaluator目前应该统一使用我们写好的RepresentationEvaluator，因此不用管。

executor目前应该统一使用我们写好的TwoStepExecutor，因此不用管。

自定义model后需要在需要在libcity/config/model下添加对应的json文件,例如libcity/config/model/region_representation/Node2Vec.json

最后在libcity/config/task_config.json,指定目标任务，支持的模型，数据集，以及模型都采用什么dataset，executor，evaluator。

例如：

```
    "region_representation": {
        "allowed_model": [
            "Node2Vec"
        ],
        "allowed_dataset": [
            "bj_dataset"
        ],
         "Node2Vec": {
            "dataset_class": "Node2VecDataset",
            "executor": "TwoStepExecutor",
            "evaluator": "RepresentationEvaluator"
        }
    }
}
```

运行模型

```
python run_model.py --task region_representation --model Node2Vec --dataset
bj_dataset
```

```
> python run_model.py -h
usage: run_model.py [-h] [--task TASK] [--model MODEL] [--dataset DATASET]
                    [--config_file CONFIG_FILE] [--saved_model SAVED_MODEL]
                    [--train TRAIN] [--gpu GPU] [--batch_size BATCH_SIZE]
                    [--train_rate TRAIN_RATE] [--eval_rate EVAL_RATE]
                    [--learning_rate LEARNING_RATE] [--max_epoch MAX_EPOCH]
                    [--gpu_id GPU_ID]

optional arguments:
  -h, --help            show this help message and exit
  --task TASK           the name of task
  --model MODEL         the name of model
  --dataset DATASET     the name of dataset
  --config_file CONFIG_FILE
                        the file name of config file
  --saved_model SAVED_MODEL
```

```
                        whether save the trained model
  --train TRAIN         whether re-train model if the model is trained before
  --gpu GPU
  --batch_size BATCH_SIZE
  --train_rate TRAIN_RATE
  --eval_rate EVAL_RATE
  --learning_rate LEARNING_RATE
  --max_epoch MAX_EPOCH
  --gpu_id GPU_ID
```

更多参数介绍见libcity官方文档：https://bigscity-libcity-docs.readthedocs.io/zh_CN/latest/user_guide/config_settings.html