强化学习：作业一

丁云翔 191250026

October 21, 2021

1 作业内容

在“蒙特祖马的复仇”环境中实现Dagger算法。

2 实现过程

修改了code文件夹下的main.py和Dagger.py文件，修改之处如下。

在main.py中增加了Dagger算法相关代码，该算法主体都位于该文件的main函数内。

初始化数据集D为空集。

```
data_set = {'data': [], 'label': []}
with open('./imgs/label.txt', 'w') as f:
    f.truncate()
```

策略的初始化在Dagger.py中进行，此处无需修改。
接着在循环中进行训练，一共训练$num_updates$轮。

```
# start train your agent
for i in range(num_updates):
    # an example of interacting with the environment
    # we init the environment and receive the initial observation
    obs = envs.reset()
    # we get a trajectory with the length of args.num_steps
    for step in range(args.num_steps):
```

建立使用的策略，进行$args.num_steps$步采样，以$epsilon$的概率询问专家，以$1-epsilon$的概率选用当前策略$\hat{\pi}_i$，并记录下那些询问了专家的局面和对应的动作，将他们添加到数据集D中。

```
        # we get a trajectory with the length of args.num_steps
        for step in range(args.num_steps):
            # Sample actions
            epsilon = 0.05
            if np.random.rand() < epsilon:
                # ask the expert
                im = Image.fromarray(obs)
                im.save('imgs/' + str('screen') + '.jpeg')
                action = int(input('input expert action'))
                while action < 0 or action >= action_shape:
                    action = int(input('re-input action'))
                query_cnt += 1
                data_set['data'].append(obs)
                data_set['label'].append(action)
            else:
                # we choose a special action according to our model
                print("Choose action from model")
                action = agent.select_action(obs)

```

在新的数据集D上训练新的策略$\hat{\pi}_{i+1}$。 当$num_updates$轮结束后，得到最后的策略$\hat{\pi}_i$。

```
        # design how to train your model with labeled data
        agent.update(data_set['data'], data_set['label'])

```

在Dagger.py中添加了自己的Agent。此处使用了tensorflow自带的estimator.DNNClassifier作为模型来训练并根据局面导出动作。

init函数部分

```
# here is my own Agent
class MyDaggerAgent(DaggerAgent):
    def __init__(self, necessary_parameters=None):
        super(DaggerAgent, self).__init__()
        # init your model
        my_feature_columns = [tf.feature_column.numeric_column(key="obs")]
        self.model = tf.estimator.DNNClassifier(
            feature_columns=my_feature_columns,
            hidden_units=[10, 10],
            n_classes=necessary_parameters)

```

update函数部分（这里将obs的多维列表化为了单维，便于使用神经网络训练）

```
    # train your model with labeled data
    def update(self, data_batch, label_batch):
        data = []
        for i in data_batch:
            for j in i:
                for k in j:
                    data.append(k)
        data_dict = {"obs": data}
        self.model.train(input_fn=lambda: MyDaggerAgent.train_input_fn(
            self, data_dict, label_batch), steps=100)

```

select_action函数部分

```python
    # select actions by your model
    def select_action(self, data_batch):
        data = []
        for i in data_batch:
            for j in i:
                for k in j:
                    data.append(k)
        data_dict = {"obs": data}
        predictions = self.model.predict(input_fn=lambda: MyDaggerAgent.eval_input_fn(self, data_dict))
        id = 0
        predictions = list(predictions)
        for pred in predictions:
            id = pred['class_ids'][0]
        return id

```

辅助函数部分

```python
    def train_input_fn(self, features, labels):
        return tf.data.Dataset.from_tensor_slices((dict(features), labels)).batch(1)

    def eval_input_fn(self, features):
        return tf.data.Dataset.from_tensor_slices(dict(features)).batch(1)
```

3 复现方式

在主文件夹下运行 `python main.py`.运行过程中若出现“input expert action”，则输入最佳动作对应的数字，此即专家的选择；若出现“Choose action from model”，则表示该回合由策略导出动作，无需其他操作。

4 实验效果

使用复现方式所述的方法运行算法，经测试运行良好，给出的专家动作可以正常输入，策略也能给出动作给agent执行。但未跑出结果，原因在小结部分中。

Figure 1: 执行过程截图，输入专家的动作和由策略导出动作均正常

5 思考题

在玩游戏的过程中标注数据与Dagger算法中的标注数据方式有何不同？这个不同会带来哪些影响?

答：在玩游戏的过程中标注数据是人先玩完（标注完）以后，用这些已经被标注的数据来进行训练，在训练中一般不会再新增标注数据，即标注数据与训练是完全分开的；而Dagger算法是在训练的过程中标注数据，每轮采样都会有一定数量的询问专家的数据被标注，然后加入训练样本集，即训练数据是在探索和训练的过程中被产生和标注的，随着探索和训练的进行，训练数据集逐渐扩充。带来的影响是使得训练数据的生成变得序列化，因为训练数据是在探索过程中生成和标记的，前面的探索和学习会影响到后面的探索和新的训练数据的产生，使得训练数据的分布和权重发生了变化，不再像玩游戏的过程中标注的数据那样满足独立同分布和各个数据权重相等且不变。

6 小结

这次实验的代码大体上完成了，但是没有跑出最终结果。由于是第一次使用tensorflow这样的深度学习库来训练模型，还不太熟悉它的api，并且笔记本自带的显卡也不支持cuda，光靠cpu跑tensorflow实在是太慢了，除非将询问专家的占比提到很高，否则模型训练的时间开销几乎难以承受，所以只能降低训练轮数，并适当提高询问专家的占比，才可能得到performance。但由于每一轮采样步数依然很高，所以需要求助专家的次数也很多，需要人力去一次次决策和输入，几乎也不可能完成，所以最后算法虽然能够运行，却没能得到performance。