SeedCup2017 初赛 - 你觉得我说的对不队

使用语言以及运行环境

使用语言

- Python 3
    - TensorFlow
    - Pandas
    - NumPy

运行环境

```
➜ ~ uname -a
Linux [xxxx] 4.10.0-35-generic #39~16.04.1-Ubuntu SMP Wed Sep 13
09:02:42 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
➜ ~ python3
Python 3.5.2 (default, Aug 18 2017, 17:48:00)
[GCC 5.4.020160609] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

程序函数以及接口

./my_utils/read_team_data.py

read_team_data(...)

- Param:
    - path：数据文件路径，i.e. teamData.csv
- Return:
    - team_features：以队名为 keys，统计后特征值为 value 的字典

modify_team_feature(...)

- Param:
    - team_feature：某个队所有队员的原始数据
- Return:
    - modified_team_feature：按以下公式计算的球队特征值（已基本归一）

$$\sum_{\text{所有队员}} \text{上场时间} \times \text{其余各项数据值}$$

./my_utils/read_match_data.py

read_match_data(...)

- Param:
  - path：数据文件路径，i.e. matchDataTrain.csv
    - **注：**已移除原数据最后 300 条，另存为 matchDataTest_Self.csv 作为测试数据集，防止过拟合情况发生
- Return:
  - match_results：以（<主场队名>，<客场队名>，<one-hot 格式比赛结果>）元组为元素的列表

./linear_model.py

注：本模组为学习过程中的副产物模型，是一个简单的一次线性模型，在这里只引入在 DNN 模型中用到的功能！

my_input_x(...)

- Param:
  - match_result：（<主场队名>，<客场队名>，<one-hot 格式比赛结果，赢得比赛的队伍的对应位标 1>）形式的元祖
- Return:
  - feature_diff：主场队伍特征向量与客场队伍特征向量的差，该列表将作为模型 feat_hold 输入

my_input_y(match_result)

- Param:
  - match_result：同上
- Return:
  - 直接返回 match_result[2]

./fully_connected.py

inference(...)

- Param:
  - feature：队伍特征值差
  - hidden1_unit：一层神经元个数
  - hidden2_unit：二层神经元个数
- Return:
  - logits：二元向量，分别代表主场、客场队伍赢得比赛可能性的代表值（**注：不是概率值！**因此随着模型不断训练，且后期并没有对 logits 值进行 sigmoid 处理，其输出值的绝对值可能无上确界）

loss(...)

- Param:
  - logits：同上
  - truth：one-hot 格式真值
- Return:
  - loss：预测值与真值之间的跨熵，为需要通过训练降低的目标

`training(loss, learning_rate)`

- Param:
  - `loss`：同上
  - `learning_rate`：学习率
- Return:
  - `train_op`：训练方式（减小上述的 `loss` ）

`evaluation(...)`

- Param:
  - `logits`：同上（预测值）
  - `truth`：同上（真值）
- Return:
  - 模型评判方式：`logits` 中更高的值是否对应获胜的球队

`placeholder_inputs(...)`

- Param:
  - `batch_size`：每步训练数据堆大小
- Return:
  - `(feat_hold, pred_hold)`：送给训练模型的数据在计算图中的占位符

`fill_feed_dict(...)`

- Param:
  - `feat_hold, pred_hold`：同上（占位符）
  - `size`：预留 `batch_size`
  - `eval_flag`：生成测试集重用该函数
- Return:
  - `(np.array(特征), np.array(真值))`

`do_eval(...)`

给模型喂数据，检查预测准确度

`run_training()`

主函数（训练后生成预测结果 csv）

数据提取与模型选取

这里直接用每个队员的上场时间乘以其各项数值，最后加起来，汇总成队伍的特征值。我们认为，上场时间长的队员以及队伍，更有“说话的权利”，因此以这种方式赋予更高的权重。由于我们使用深度网络（两层全链接）直接进行拟合，所以我们把通过以上方式得到的队伍特征值全部送入模型。

运行方法

```
(tf) ➜  SeedCup2017 git:(master) ✗ ./fully_connected.py
importing my_utils
loss = 0.790 at step 0
loss = 0.668 at step 100
loss = 0.643 at step 200
loss = 0.672 at step 300
...
loss = 0.395 at step 149600
loss = 0.511 at step 149700
loss = 0.308 at step 149800
loss = 0.482 at step 149900
Training data eval:
got 8193 out of 9900, accuracy 0.828
Test data eval:
got 207 out of 300, accuracy 0.690
(tf) ➜  SeedCup2017 git:(master) ✗
```

注：继续提高训练步数 `fully_connected.max_steps` 很大几率上能够提高准确度（我们最高的一次结果是 78% ）。当然，这一切都取决于你的运气如何了 :-)

一点声明

由于本方案并不采用概率值作为队伍是否能够获胜的评判标准，所以 predictPro.csv 中的数值对谁都没有什么实际意义，除了模型他自己。