EE228 课程大作业

完全离线训练的 2048 游戏 CNN 代理

周睿文 5180219111150

2020 年 6 月 21 日

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

项目1完成情况

- 10 轮平均分数：1945.6
- 方法简述：完全离线训练 CNN 进行 2048 游戏
- 主要使用的代码框架：PyTorch
- 模型大小 (MB)：107.0
- 研究成果亮点：
  - 完全离线训练，方法简单有效
  - 拼接两个不同的 CNN，有效提升模型性能
  - 单局均分大于 1024 的比例在 70% 以上，模型具备高水准
- 代码地址：https://github.com/SkyRiver-2000/2048-api

问题描述及解决思路

- 训练基于机器学习方法的 2048 游戏代理
- 在完全 Offline 且没有分层处理的条件下，研究的主要难点：
  - 模型的训练“前后抵消”，即 TA 提到过的“遗忘”问题
  - 直接学习 Expectimax 产生的理想棋盘，会产生累积错误的问题
- 解决上述难点的思路:
  - 梯度下降中学习率过高导致的问题与遗忘现象相似 → 减小学习率
  - SGD 中，最终的结果将在最优解附近振荡而非收敛 → 增大 Batch
  - TA 提供的网络在Offline条件下，高分段“后劲不足” → 拼接不同网络
  - 累积错误问题的来源：用于训练的“经验”不够丰富 → 增加训练数据

解决方案综述

- 模型结构设计：
  - 在 GitHub 上调研时发现了另一个有效的 CNN 结构
  - 两个模型单独训练的跑分均停留在 800 附近，难以继续上升
  - What if they are combined?
- 模型训练设计：
  - Simple is beauty，采用完全离线训练
  - 如何防止遗忘和累积错误的问题？关键：调整学习率和批规模

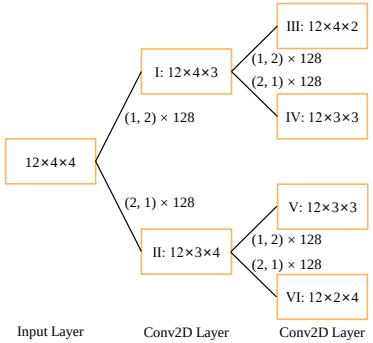
模型结构设计——网络拼接

- 卷积原型 1：

* 该原型引自：

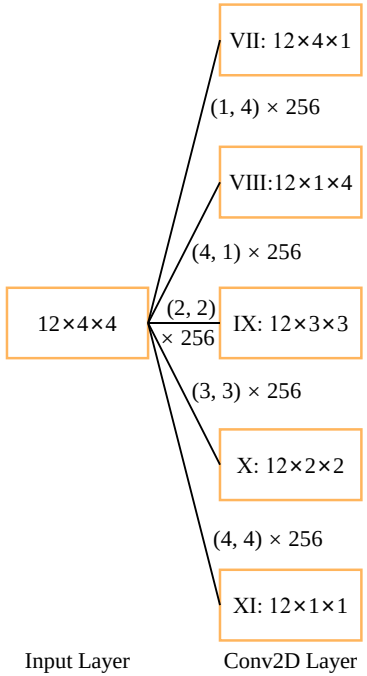
navjindervirdee

@ GitHub

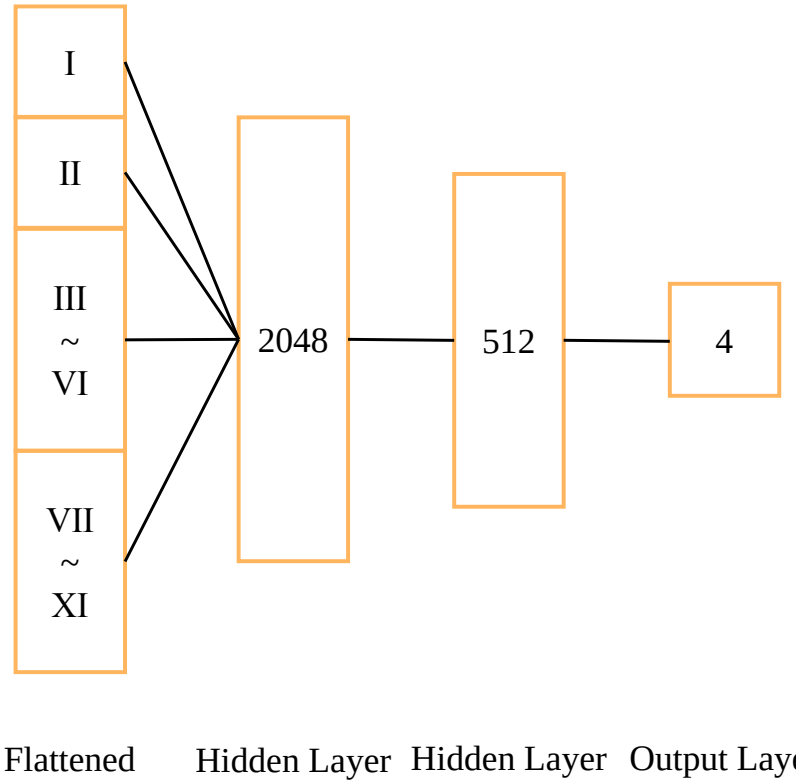


- 卷积原型 2：

* 该原型由 TA 提供



展开后最终的全连接结构：



结构改进的有效性分析

- 同样进行 50 轮训练，结束时前两个网络 Loss 和准确率改善已经很小
- 在不同数据规模下，训练后分别进行多次 10 轮测试
- 大致得到如下跑分结果：

训练数据量 / 采用的模型	8,000 局	15,000 局
TA 提供的网络	600 ~ 750	700 ~ 800
GitHub上发现的网络	500 ~ 700	550 ~ 750
拼接改进的网络	700 ~ 900	800 ~ 1100

- 观察到拼接得到的网络在相同的训练数据规模下，性能明显更出色

模型训练设计

- 优化器：Adam，初始学习率：1×10^{-5}
- 设置 30 轮学习率指数衰减，系数为 0.96，Final LR $\approx 3 \times 10^{-6}$
- Batch size = 1024，总训练轮数：100
- 训练总时长：约 35 小时（平均 20 min / epoch + 每轮训练后测试）
- 使用的数据：
  - 利用预提供的基于 Planning 的 Expectimax 生成理想棋盘
  - 共计进行 30,000 局游戏，约 21,000,000 条记录
  - 数据量大，即“经验足够丰富”，增强泛化能力，缓解累积错误的问题

模型性能分析

- Agent 的成绩分布与稳定性：经过 150,000 轮的测试，
  - 单局均分: 1116.08，10 轮均分 > 1024 的比例为 70.23%
  - 单局表现统计:

Max Tile	16 and 16-	32	64	128	256	512	1024	2048	Total
Frequence	27	69	427	2083	8918	31371	68926	38179	150000
Frequency	0.018%	0.043%	0.285%	1.389%	5.945%	20.914%	45.951%	25.453%	100.000%

- Agent 的时间和空间性能：
  - 单局游戏时长约为 1.1 s，单步预测时间约为 1.5 ms
  - 总体来说，107 MB 的模型规模较大，但拼接网络的优势清晰可见

研究心得与代码技巧

- 除了 LR 以外，Batch size 对解决遗忘问题同样不可忽视
- 网络描述能力不足：增加模型复杂度，如改进网络结构等
- 可能的算法改进 (不局限于 Offline)：
  - 就直观感受而言，内核选取恰当的 DQN 有可能稳定达到 2048
  - 在大规模离线训练结束后，进一步执行增量学习
  - 数据分层、使用多个网络分别进行学习
- 在代码方面的一些小技巧：
  - 充分利用 torchvision、tqdm、pandas 等工具
  - numpy、DataFrame 和 tensor 的矩阵操作
  - 由于规则允许，可以编写刷高分的脚本（详见附页）

Thank You

如何高效、自动化刷分?

Shell 刷分脚本

```
# Clear exist log
if [ ! -d "./log_storage/temp1.log" ]; then
    rm log_storage/temp*.log
fi

# Run evaluate.py continuously
# Store each log file produced
for i in $( seq 1 2000 )
do
    python evaluate.py >> log_storage/temp${i}.log

    # Display running status
    echo -n "Round: ${i}, "
    tail -1 log_storage/temp${i}.log
done

# C++ code written for data aggregation
g++ -o main main.cpp
./main
```

脚本输出的信息

```
Round: 1980, Average scores: @10 times 1305.6
Round: 1981, Average scores: @10 times 883.2
Round: 1982, Average scores: @10 times 1228.8
Round: 1983, Average scores: @10 times 793.6
Round: 1984, Average scores: @10 times 1280.0
Round: 1985, Average scores: @10 times 1024.0
Round: 1986, Average scores: @10 times 1843.2
Round: 1987, Average scores: @10 times 998.4
Round: 1988, Average scores: @10 times 1459.2
Round: 1989, Average scores: @10 times 1228.8
Round: 1990, Average scores: @10 times 908.8
Round: 1991, Average scores: @10 times 1049.6
Round: 1992, Average scores: @10 times 947.2
Round: 1993, Average scores: @10 times 1024.0
Round: 1994, Average scores: @10 times 1126.4
Round: 1995, Average scores: @10 times 1433.6
Round: 1996, Average scores: @10 times 1049.6
Round: 1997, Average scores: @10 times 1024.0
Round: 1998, Average scores: @10 times 1126.4
Round: 1999, Average scores: @10 times 1203.2
Round: 2000, Average scores: @10 times 1075.2
Average score per game: 1115.69
Max score @10 times: 1843.2
It appears at: temp1986.log
The proportion of 1024 and higher average score is: 71.35%
```