人工智能大作业报告

田佳析

2021 年 1 月 30 日

目录

1 监督学习

1.1 问题描述

能够用拍照方式识别五子棋下棋过程中当前落子的位置，识别程序中应使用到监督学习算法。

1.2 方法描述

使用 yolov5 实现对五子棋的目标识别，使用编写的五子棋界面生成训练和测试数据。下载预训练模型后，进行迁移学习，训练得到识别黑白棋的模型。

1.3 原理描述

1.4 步骤描述

1. 利用五子棋界面生成训练数据

   对于五子棋界面的实现，考虑到界面不是本次作业实现的主要目标，所以使用 tkinter 库来实现五子棋简易界面。

   yolov5 所用的标签格式为 [物体类型 物体中心 x 值 物体中心 y 值 物体长度 物体宽度]。随机生成五子棋局面，绘制界面，生成标签，训练数据大小为 200 张图片

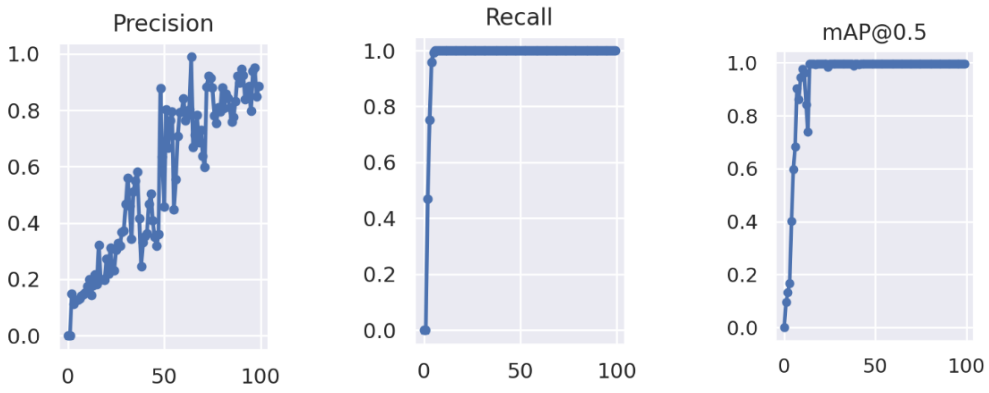
   生成训练数据的代码位于 Python/create_train_data.py 文件中。

2. 下载 yolov5 项目，下载预训练模型

   yolov5 所用的模型代码来自 github 上的一个 项目。考虑到黑白棋的特征明显，识别出来的难度不大，所以我使用的预训练模型为 YOLOv5s 模型。在进行 100 次的训练后能够较好地得到结果。

3. 对模型进行训练

   模型训练过程中准确率和召回率和 IoU 阈值为 0.5 时的 mAP 值变化如下：



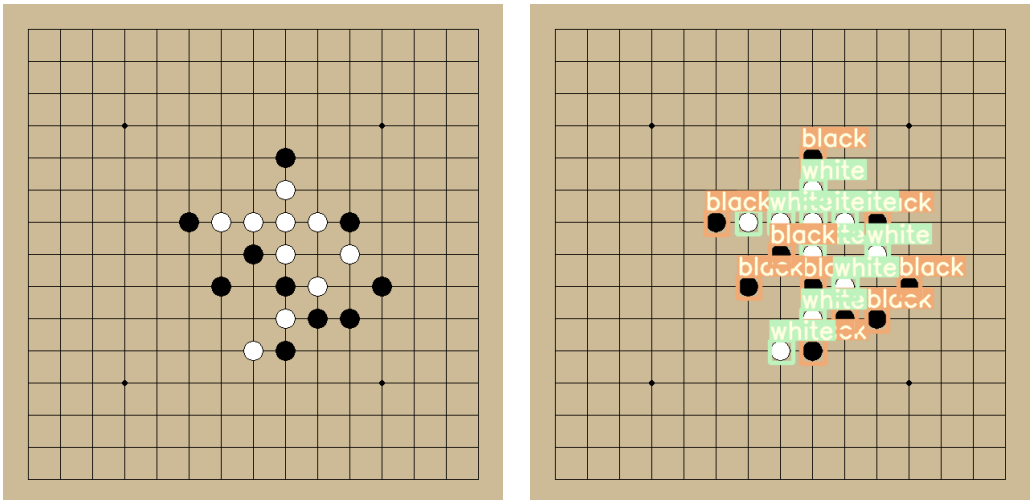
(a) 准确率　(b) 召回率　(c) mAP:0.5

图 1: 模型指标变化

可以看出模型效果随着训练逐渐变好。

1.5 结果分析

输入的测试图片和识别图片如下：



(a) 测试图片　　(b) 识别图片

图 2: 模型预测图片

可以看到白棋和黑棋被识别了出来。

2 博弈搜索

2.1 问题描述

采用一种博弈搜索算法，实现五子棋博弈程序，其中对棋局状态的判断采用人为设定函数方式。

2.2 方法描述

通过自定义的评估函数返回棋局评分，使用 alpha-beta 搜索，结合一定的优化方法，计算得到 ai 下一步所走的位置。

2.3 原理描述

AB 剪枝就是 minmax 搜索的优化。在 MAX 层，假设当前层已经搜索到一个最大值 X，如果发现下一个节点的下一层（也就是 MIN 层）会产生一个比 X 还小的值，那么就直接剪掉此节点。在 MIN 层，假设当前层已经搜索到一个最小值 Y，如果发现下一个节点的下一层（也就是 MAX 层）会产生一个比 Y 还大的值，那么就直接剪掉此节点。

当搜索到达了最大层数，对节点进行评分，根据不同节点评分进行剪枝。搜索完毕后，走分数最大的位置。

2.4 步骤描述

1. 编写自定义的评估函数

   评估函数采用了常用的根据形成的棋型打分的方式对于形成的不同棋型，如活四、活三、死四等，给予不同的分数。对于敌我双方分数的判定，没有进行特殊的分析方式，而是分别计算双方的总分，然后敌方分数降低一半进行相减，得到局面分数。

   评估函数的实现方式在 C++/code_file/Evaluation.cpp 文件中。

2. 编写获取搜索位置的函数

   根据五子棋“战场”的局部性，我认为下载偏远地方的棋子对于胜率影响不大，所以对于一个位置，只有它周围有棋子的时候，才进行搜索。

   可以根据启发式函数获取搜索位置，但我试着实现计算空位得分方式进行启发式搜索，发现结果反而更慢。。。于是就放弃了启发式搜索。

   搜索位置函数实现方式在 C++/code_file/Get_position.cpp 文件中。

3. 编写搜索函数

   首先调用获取搜索位置函数获得搜索位置，然后进行 alpha-beta 搜索，当搜索到达最大层数时，调用评估函数，返回评估值。

   在搜索的过程中，需要判断局面是否结束，如果敌方胜利，则上一步的落子是失败的，不能下在这。

   搜索函数实现方式 C++/code_file/AB_search.cpp 文件中。

2.5 结果分析

设置搜索层数为 4 时，ai 搜索一些步骤耗费时间较长，可能是评估函数定义得不够好，导致 alpha-beta 发生的剪枝次数较少，ai 搜索轻情况太多，搜索时间较长。

设置搜索层数为 2 时，ai 能够较快地得出结果，并且具有一定的棋力。

对于搜索效果的进一步优化，我认为从

3 进化计算

3.1 问题描述

将上述博弈搜索算法中判断棋局状态的函数改为一种人工神经网络模型，并采用进化计算方法对该人工神经网络模型来进行学习，使得五子棋博弈程序的下棋水平不断提高。

3.2 方法描述

在进行方法说明前，我得吐槽一句，从一开始我就对遗传算法优化神经网络的效果是不报任何希望的。不说 neat 算法同时进化网络结构和权值，所耗的时间无法估计。找了一个利用遗传算法优化网络

权值的论文，一看，方法是首先优化得到粗略的权值，最后还是用梯度下降训练权值，好家伙，再一看，才几百个参数。对于 225 的输入数量，上万的权值数，这点参数还不够塞牙缝的。

我将这个问题视为一个多分类问题，输入当前局面，输出为各点的落子概率。以网上找的棋局作为训练数据，进行监督学习。

首先确定网络结构，输入层节点数为 225，一个隐含层，节点数为 300，输出层，节点数为 225，最后输出值进行 softmax 归一化，预测下一步的落子。适应度函数为交叉熵函数，标签为棋手下一步的落子位置。根据各网络的适应度进行选择操作，之后进行交叉，变异操作，不断进行迭代。

3.3 原理描述

开始时本来不准备用监督学习的，打算以第二位的搜索 ai 作为环境，坚持的时间越久，适应度越高。但是每个网络都在 12 步之内就输掉，适应度也一直不会升高。想想还是不太靠谱，就换成监督学习了。

3.4 步骤描述

实现代码位于/Python/Evaluation/Net.py 文件中。

1. 适应度函数
2. 选择操作
3. 交叉操作
4. 编译操作

3.5 结果分析

下面是迭代了 100 代后最好适应度和平均适应度的变化：

可能是我没写对吧。。。种群的适应度没有什么太大的变化，可能是交叉操作过于简单，对于网络生成的权值影响较小。

4 强化学习

4.1 问题描述

采用强化学习算法对上述人工神经网络模型进行学习，使得五子棋博弈程序的下棋水平不断提高。

4.2 方法描述

4.3 原理描述

4.4 步骤描述

4.5 结果分析