summary of alphgo

2017 年 5 月 9 日

下文是对文章 Mastering the game of Go with deep neural networks and tree search 的 summary

2016 年 Alphgo 打败了围棋，意味着在完全博弈棋类游戏方面人类的胜利。围棋 19*19 的盘面共 361 个子，围棋的评价函数难以给定，搜索空间太大，导致在围棋上不能使用传统的 minmax 搜索和剪枝。搜索空间减小有两个常用的方法，第一是通过局面评估剪枝，降低搜索深度，第二是通过从一个分布 $p(a|s)$ 中采样 action 降低广度，$p(a|s)$ 指在局面 s 下可能的走法 a。之前强大的围棋 AI 用的是 MCTS 和走子网络。现在 Alphgo 在上面进行了一些改善。

深度学习在图片上的成功应用使作者考虑是否可以应用在围棋上，棋盘可以看成 19*19 的图片，其中每个位置可以用 (0,-1,1) 分别表示 (无子，黑子，白子)，Alphgo 主要包括四个部分，1 利用 cnn 用监督学习直接从人类专家棋盘走子中学到走子网络 P_σ，这一步利用即时的反馈和高质量的梯度下降可以快速有效更新，2 利用统计学习方法，建立快速走子 P_π 网络，可以在 roolouts 过程中快速选子，这部分相较于 cnn 准确率低，但是速度从 3 毫秒到 2 微妙，3 用增强学习建了 P_ρ，通过自我训练优化最终结果改善 P_σ，使学习的目的是赢而不是最大化预测走子的准确率，4 训练估值网络，通过增强学习的 P_ρ 自我博弈训练，预测游戏胜者。Alphgo 就是用 MCTS 有效组合上面的四部分再借 google 强大的计算能力打败围棋冠军。

下面分别介绍一下上面各部分

第一个走子网络 P_σ 是用 cnn 训练的，首先来了解下 CNN，全

连接网络有一些特点，每一个单元和下一层的所有单元都连接，且每个参数都不要求相同，这就使得需要训练的参数非常多，cnn 在这上面增加一些限制，首先一些单元只和下一层的部分单元连接，具体连接单元个数取决于卷积核 (有的地方称为 filter) 的大小，这一层中所有单元之间的连接共用一个卷积核，这被称为层级感受野和权值共享，这样需要训练和存储的参数大大减少，除卷积层外，还有池化层，常用 maxpooling, 另外还有最小值 pooling 和均值 pooling，maxpooling 是将指定范围内的数用它的最大值代替。在卷积和 pooling 的过程中，元素都会越来越少，可以通过边缘补齐来保证每一层单元数目不减少。作者从 KGS 上找到很多下棋序列，每一个 pair 都是 (s,a) 的形式，表示棋盘 s 下人类的下子 a，那么这个 pair 就可以转化成一个训练样本用 19*19*n 表示，其中 n 包括一些其他特征。不断训练 cnn，交替的用卷积层和非线性激活函数，最后用一个 softmax 输出所有合法走子上的概率，在随机选择的 pair(s, a) 上训练, 用随机梯度下降去极大似然 a，

$$\triangle\sigma \propto \frac{\partial log P_a(a|s)}{\partial\sigma}$$

利用 KGS 上找了 30million 的局面，13 层的神经网络，这种训练完成后就得到了 P_σ，这样的方法可以达到 57% 的准确率。

但是用 P_σ 与作者之前的一个围棋 AI 对弈，并没有达到很理想的效果。于是作者结合了当时最优的围棋 AI 的方法：MCTS。MCTS 的思想是我想知道这个局面上下哪个子最好，我就随机选一个点 a，用两个完全没有背景知识的人一直沿着状态随机下子，这一过程下子的宽度为 1，深度是一直下到出现胜负结果，被称为 roolout。当在这个选择的点 a 上下了足够多轮，认为可以用胜利的准确率表示这个下子位置的优劣。那么完全随机的 uniform 的选择点可能会浪费掉很多搜索的宽度，是否可以加点经验知识呢? 于是作者考虑不再随机选子，而是用 P_σ 对局面的输出结果作为概率来选子，这种选子策略不再那么简单，但是速度上慢了很多，原来 uniform 的方法每次选子只需要 $1\mu s$, 但是现在需要 $3ms$，而这个随机选子的过程只有轮数足够多胜利概率才有价值，因此作者

用简化版的走子网络快速走子 P_π 网络，在 roolouts 过程中快速选子，P_π 利用一层的特征加权，达到了 24.2% 的准确率，但是时间降低到 $2\mu s$，并且这种方法可以在等待对方下子的过程中，我仍然在模拟下子更新，等到对方下完子，局面更新，我从对方下子的局面开始模拟，而之前模拟的结果可以保存，因为对方下子后的局面很可能我已经模拟过。

现在得到的走子网络已经很强大了，但是我们在训练走子网络的过程中预测的目标是走子位置的正确性，而游戏的目的是使比赛胜利。因此利用增强学习完善 P_σ 得到走子网络 P_ρ。增强学习简单的讲，就是根据当前的状态，选择一个 action, 根据 action 会给一个 reward，智能体再根据给的 reward 调整自己的 action，大概率选择当前状态 s 下得分大的 action。不断这种调整，面对 s 越来越能找到 reward 大的 a。P_ρ 和 P_σ 结构完全相同，参数也是用 P_σ 初始化，我们让当前的 P_ρ 和之前迭代过程中随机选择一个 P 对弈，从之前迭代过程中随机选择对手可以防止过拟合当前的 P_ρ，用一个奖励函数 r(s)，在游戏没终止 $t < T$ 时 r(s) 都为 0，终止时，奖励 $z_t = \pm r(s_T)$, 在每个时间步上，权重都在用随机梯度下降更新，更新方向是最大化 Z_t

$$\triangle\rho \propto \frac{\partial log P_\rho(a_t|s_t)}{\partial\rho} Z_t$$

我们评估了 P_ρ 的表现，从它在合法走子上的输出概率上采样每一步，a_t $p_\rho(\bullet|s)$，当端对端对弈时，MCTS 结合 RL 下的 P_ρ 可以 80% 的胜过 SLP_σ，也胜过当时最好的 AI. 只用 MCTS 结合基于监督学习 cnn 的 P_σ 也胜过当时最好的 AI。

除了上面的走子网络，作者还想通过增强学习得到估值网络。功能是根据当前局面判断白子胜还是黑子胜。找到一个估值函数，对局面 s 用走子网络预测结果：

$$v^p(s) = E[z_t|s_t = s, a_{t...T}\ P]$$

理想情况下，我们希望找到最优的估值函数，在实验中，我们没有用 P_ρ 估计价值函数 V_ρ^p, 而是用权重为 θ 的价值网络 V_θ 近似。这

个估值网络和走子网络结构相似，但是不是输出一个概率分布，而是一个预测值。我们在训练对 (s, z) 上用回归训练，使用 sgd 最小化预测值和实际结果之间的均方差。

$$\triangle\theta \propto \frac{\partial v_\theta(s)}{\partial\theta}(z - v_\theta(s))$$

从完整游戏组成的数据集上用很朴素的方法预测游戏结果会导致过拟合，因为连续的状态强相关，但是预测的目标值在整个游戏上共享，当用 kgs 的数据集在这种方式上训练时，估值网络记忆了游戏结果，在训练集 0.19，测试集 0.37。为了减轻这个问题，我们用自对弈生成 30million 个不同的局面，自对弈过程中开局先用 p_σ 走 k 步，然后再随机走一步，来保证走子的多样性，接着用 p_ρ 来走，直到结束得到样本。每个样本来自不同的游戏，每个游戏都是用 P_ρ 自对弈直到游戏结束，结果是训练集和测试集上 0.226 和 0.234。这么做后得到的估值网络与利用蒙特卡洛和随机，P_ρ100 次得到的平均值对比，局面上已经下的子越多，评估越准确，估值网络可以得到和 P_ρ 一样的结果，但是少了 15000 倍的训练。

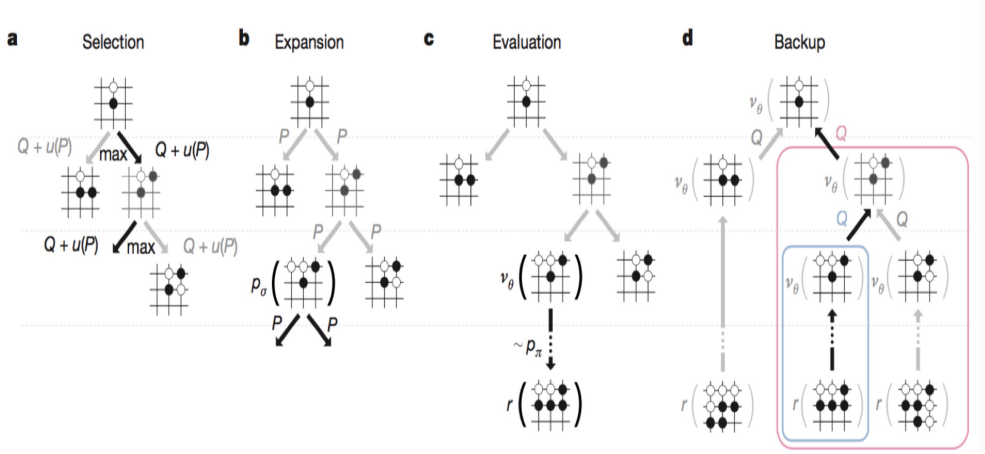

图 1: MCTS 过程

介绍完上面几部分，下面就是用 MCTS 将估值网络和走子网络结合起来，搜索树中每条边上存了走过的次数 N(s,a)，action value Q(s,a) 和 prior probabilit p(s,a)。MCTS 包括四个步骤，1

selection，先用 P_σ 开局，每步选择：

$$a_t = argmax_a(Q(s,a) + \mu(s,a))$$

最大的地方落子，其中 $Q(s,a)$ 是 P_σ 对局面评估分数，

$$\mu(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$$

, 这个值和 prior probability 成正比，但是与搜索过的次数成反比，这是为了给探索次数少的位置更大的机会。2 expansion 用上面的方法选择点先下 L 步，下到 L 步后，用 P_σ 计算下每个合法走法的概率作为 prior probability，用快速走子结束 P_π 下面的棋局 3 evauation 在一次模拟结束之后，用 r 函数评估叶子节点值 Z_L，并和估值网络 V_θ 结合作为一个合法走法的估计值 $V(s_L)$, 作者经过试验发现 $\lambda = 0.5$ 时效果最好。

$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda Z_L$$

4 backup，反向更新，更新所有走过的边，计算经过这条边的次数和在这条边上得到的 evalutions 的均值，其中 $1(s,a,i)$ 表示在第 i 次模拟中是否经过 (s,a)，

$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$$

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i)$$

一旦搜索结束，选择 Q(s,a) 最大的位置落子。