alphago 论文学习

Yuchen Zhong

2018 年 3 月 21 日

1 alphago

用传统的办法，主要是搜索，但是对于围棋这种 depth 和 breadth 都很大的棋类来说，是非常困难的。目前最强的传统方法只达到了业余水平。

本文训练了 4 个神经网络:

- p_σ: 用 Supervised learning 学习人类棋谱
- p_π: 同上，但是神经网络更浅，速度更快，称为快速下子 (rollouts)
- p_ρ: 强化学习，左右互搏来 optimise
- v_θ: 价值网络，根据局面算出胜率,predict winner

第一步是监督学习, Supervised Learning 主要是用了 CNN 训练一个策略网络, 输入是 19x19x48 的 tensor. 为什么维度这么高呢? 19x19 是棋盘大小，48 包括了各种信息 (黑子，白子，打吃，征吃等等). 网络结构是 13 层，最后一层是 softmax，输出是各个可下点的概率值 $p(a|s)$.

然后用人类下的位置作为 labels, 用的是随机梯度上升来训练. 目标是 maximize 人类大师下的位置的概率值. 权值 σ 用随机梯度上升的修改:

$$\Delta\sigma \propto \frac{\partial log p_{\sigma(a|s)}}{\partial\sigma} \tag{1}$$

学习了 3 千万步的围棋走法后，这个策略网络可以模拟人类落子 57% 正确率. 但是缺点是速度太慢. 所以还训练了一个更快的但是准确率所有降低的快速下子策略网络 (rollout policy network).

第二步是强化学习，是为了进一步提高 policy network 的能力, 就是从和上一步最终得到的神经网络开始训练，然后随机从之前迭代中训练的某个策略网络进行对弈 (随机是为了防止过拟合)

虽然网络结构和 SL 相同，但是目标不再相同.SL 的目标是模拟人类的下棋, 但是 RL 的目的是为了赢棋. 它的方法是定义了一个 reward function $r(s_t)$. T 代表终止的步骤, 当 $t < T$ 时, 那么 $r(s_t) = 0$ 每一步的收益定义为 $z_t = \pm r(s_T)$ 如果赢棋, 那么为 $+1$；输棋就为-1. 权值修改如下:

$$\Delta\rho \propto \frac{\partial log p_{\sigma(a_t|s_t)}}{\partial\rho} z_t \tag{2}$$

乘以 z_t 是为了让网络朝着赢棋的方向训练.

这样训练出来的 RL 策略网络，对战 SL 策略网络就有 80% 的胜率了. 而且对战传统搜索方法的围棋 AI 也有 85% 的胜率.

第三步是训练 value network, 首先当然是要定义什么是 value,

$$v^p(s) = E[z_t|s_t = s, a_{t...T} \in p] \tag{3}$$

意思就是给定策略方案 p, 从状态 s 出发一直到结束的期望收益

当然我们是不知道最优的方案 p 是什么，所以只能用当前最强的 RL 策略网络来计算一个棋面的价值 $v^{p_\rho}(s)$. 然后训练一个 value network 来拟合, 即 $v_\theta \approx v^{p_\rho}(s) \approx v^*(s)$

Value network 也是一个 CNN，但是输出是一个神经元，也就是一个标量，代表价值.cost function 就用 MSE, 然后用 SGD 来收敛.

$$\Delta\theta \propto= \frac{\partial v_\theta(s)}{\partial\theta}(z - v_\theta(s)) \tag{4}$$

第四步就是 MCTS(蒙特卡洛搜索) 了. MCTS 主要分为 4 步:

1. selection: 首先每个节点 (论文中是边，这个没有区别) 都记录三个值:
   - $Q(s, a)$action value
   - $N(s, a)$visit count
   - $P(s, a)$prior probability 计算公式后面会谈到.

   每次模拟从当前局面开始. 在第 t 步的时候，根据如下公式选择 a_t

   $$a_t = \arg\max_a(Q(s_t, a) + u(s_t, a)) \tag{5}$$

其中

$$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)} \tag{6}$$

$u(s,a)$ 用了一种反馈的思想，就是访问地越多，下轮中就越难访问，而先验概率越大，下轮中就越容易访问.（分母中的 1 是为了防止除零)

2. expansion: 根据如上方法终止于叶子节点，局面为 s_L. 这时候要对叶子节点进行 expansion. 怎么拓展呢? 使用 SL 策略网络算出 $p(a|s_L)$, 把这个顺便作为先验概率 $P(s_L,a)$

3. simulation: 一方面用快速下子策略网络 (rollout) 一直模拟到棋局结束. 得到一个反馈 (z_L), 另一方面用 value 网络来评估出一个 value, 然后总的 value 是二者的加权平均 (参数 λ),

$$V(s_L) = (1-\lambda)v_\theta(S_L) + \lambda z_L \tag{7}$$

后文提到 λ 取值 0.5 表现最好

4. backpropagation: 最后把模拟路径上所有的节点的访问次数增加 1，同时更新 Q 值，形式化描述如下:

$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i) \tag{8}$$

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i) \tag{9}$$

其中,s_L^i 是第 i 次模拟中的叶子节点.

模拟多次后，采用 root 下面访问最多的节点为下一步的走法.(理由很简单，访问次数和访问概率是成反比的，所以访问次数最多证明是 best move)

看到这里不免有个疑问，为什么不采用 RL 却用 SL 呢? 文中提到，是因为 SL performed better 可能是因为人类是选择了几步有希望的棋，但是 RL 只选择当前最优.

所以纵观 alphago，主要是以 MCTS 为主要框架，SL 来算先验概率,rollout 和 value network 来评估局面，RL 主要是用来产生训练 value network 的数据.

2 alphago zero

如果说 alphago 的算法比较累赘的话，那么 alphago zero 的算法极为简洁. 总的来讲就是 ResNet + MCTS

ResNet 的权重参数为 θ, 输入是 19x19x17(比 alphago 降低了很多) 的棋面, 输出有两种值, 一种为可能落子点的概率 $p(a|s)$，另一个就是胜率 $v(s)$. (看到这里，感觉就是把 alphago 的 SL 和 value network 合并了) 这个网络称为 f_θ

这里要注意的是，既然不用人类棋谱，那么棋谱从哪来呢? 答案就是机器自我对弈生成棋谱. 当然一开始生成的棋谱都是很弱智的, 不过在短短 40 天内，就能超越人类水平，进步还是非常神速.

然后利用 MCTS 进行下棋，过程和 alphago 类似, 不过在 simulation 那步，用 f_θ 代替 SL. 同样, 输出的概率作为先验概率, 但是评估就直接用 f_θ 算出来的 v 了. 后面过程和 alphago 一样.

这样用更新的 MCTS 在来下棋，下好的棋谱再来训练 ResNet，如此循环，棋力日强. 不仅发现了已有的人类定式，甚至发现了新的定式.

其实看到这里，就会发现其实 alphago zero 的算法其实没有什么突破，就是用 resnet 替代了之前的 SL,rollout 以及 value network 罢了.

但是, 最让人震惊的在于，即便用机器的对战棋谱，居然还能收敛!!!
如果没有强大 GPU 来验证，**恐怕真是贫穷限制了我的想象力.**