Step-by-step guide to train AlphaRenju Zero

谢铮 付星宇

符号说明

1. s_t：第 t 次落子前的局面。抽象为一个 15×15 的矩阵，元素取值为 1，-1，0。每一盘棋可以用有限序列 $(s_1, s_2, \ldots)$ 来表示。

2. a_t：第 t 次落子的动作。记录落子的位置。每一个局面 s_t 都对应有一个可行动作向量 $\vec{a}(s_t)$。

3. (s_t, a_t)：博弈树上的一条从结点 s_t 经由动作 a_t 到结点 s_{t+1} 的边。
   每一条边都有三个重要属性：
   3.1. $N(s_t, a_t)$：在模拟的过程中，这条边经过的总次数(由 MCTS 产生)
   3.2. $P(s_t, a_t)$：给定局面 s_t，动作 a_t 被选择的先验概率(由神经网络预测)
   3.3. $Q(s_t, a_t)$：给定局面 s_t，这个动作 a_t 的价值(由 MCTS 和神经网络共同决定)
   这三个属性的初始化和更新方法详见流程。

4. f_θ：以 θ 为参数的神经网络(ResNet)。神经网络的输入为局面 s，输出为当前局面下，下一落子 a 的先验概率分布 p 和**即将走子的一方**的胜率 v。可记为：$(p, v) = f_\theta(s)$。其中实施某一个动作 a 的概率即 $p_a = P(a \mid s)$。
   注：p 的维数是 225。神经网络的输入 s 是一个(15×15)×2 的张量，其中第一层是当前的棋局矩阵，第二层是即将要落子的颜色，例如，若即将落子的是黑子，则第二层是一个 15×15 的全 1 矩阵。

5. α_θ：基于神经网络 f_θ 的蒙特卡洛树搜索(MCTS)。MCTS 的输入为局面 s，如果 s 是最终局面，则 MCTS 的输出为根据规则计算出的胜方 z (取值为 1，0，0.5 分别对应黑胜、白胜、平局)，如果 s 不是最终局面，则 MCTS 的输出为下一动作的概率分布 π。可记为：$\pi = \alpha_\theta(s)$。我们把 π 看作是一个比神经网络的输出 p 更准确的概率分布，因此在每轮训练中，都基于 π 来设计自我对弈的策略，并以 π 和 z 为目标来训练神经网络。

训练流程

每轮训练可以大致分为三步：模拟、自我对弈、训练网络。

1. 模拟

模拟指的是基于由神经网络指导的 MCTS，对博弈树进行高效探索的过程。
它可以进一步分为两个部分：扩展和更新。

1.1. 扩展

扩展指的是探索博弈树新结点和新分支的过程。每次扩展总是从博弈树的根节点（也就是空的棋局）开始。

假设当前我们已探索到 s_t 结点，此时又有两种情形：

A) 该结点是第一次被探索到

对于新探索到的结点，首先要做的是搜索所有**可行(legal)**的分支（例如，落子到已经有棋子的位置是不允许的），如此我们就在博弈树上生成了一系列以 s_t 为端点的边 $(s_t,a_{t,1}),(s_t,a_{t,2}),...,(s_t,a_{t,M_t})$ 以及新的结点 $s_{t+1,1},s_{t+1,2},...,s_{t+1,M_t}$，其中 M_t 表示从结点 s_t 出发的所有可行分支（动作）数，并且每个新结点和结点 s_t 之间都满足关系：

$$s_{t+1,i} = s_t + a_{t,i}$$

注意，有一种特殊的情形：当前局面 s_t 已经是某盘棋的最终局面，这时不再有任何一种符合规则的落子动作。换言之，结点 s_t 是博弈树上一个不可继续扩展的结点，此时直接结束本次扩展。

搜索完结点 s_t 的可行分支后就可以初始化博弈树上新生成的边和结点的属性。对新生成的每条边 $(s_t,a_{t,i})$，令：

$$N(s_t,a_{t,i}) = 0；$$

$$Q(s_t,a_{t,i}) = 0；$$

$$P_0(s_t,a_{t,i}) = f_\theta(s_t').p(a_{t,i}')；$$

其中 $P_0(s_t,a_{t,i})$ 是神经网络的原始输出。此时并不直接输入局面 s_t，而是随机选择一个和 s_t 对称的局面 s_t'（对称包括旋转对称和镜面对称，D_8）。注意，由于进行了一

个随机的对称变换，因此相应的动作分支$a_{t,i}$也被变换为$a_{t,i}'$。我们还要给它添加噪声以保证所有走法都有可能被探索到：

$$P(s_t, a_{t,i}) = \varepsilon\eta_i + (1-\varepsilon)P_0(s_t, a_{t,i})\text{；}$$

其中$\eta \sim Dir(0.1)$，一般取$\varepsilon = 0.25$。

初始化完成之后，就结束本次扩展。

B）该结点之前已经被探索过至少一次

由于我们之前探索过结点s_t，故此时已经生成了从结点s_t出发的所有可行分支$(s_t, a_{t,1}), (s_t, a_{t,2}), ..., (s_t, a_{t,M_t})$。接下来要做的就是依照某种策略选择一个可行分支到达下一个结点s_{t+1}。一个很自然的想法是，根据神经网络输出的子结点胜率$f_\theta(s_{t+1,i}).v$，选择胜率最大的子节点作为下一个探索的结点：

$$s_{t+1} = argmax(f_\theta(s_{t+1,i}).v)$$

但这种策略有一个明显缺点，它的探索能力很差，往往囿于历史经验，无法跳出局部最优。一个理想的选择策略应该具备充分的探索能力。

为此，引入一个新的变量$U(s_t, a_{t,i})$，它反映了动作$a_{t,i}$的潜在探索价值。将其定义为：

$$U(s_t, a_{t,i}) = \beta P(s_t, a_{t,i}) \frac{\sqrt{\sum_j N(s_t, a_{t,j})}}{1 + N(s_t, a_{t,i})}$$

其中β是一个控制该项权重的正常数。

除此之外，仅仅考虑当前的神经网络输出的结点胜率也是不够的。神经网络在不同参数下输出的胜率都有参考的价值，因此我们把一个动作(s_t, a_t)的价值$Q(s_t, a_t)$定义为在扩展过程中每次经过边(s_t, a_t)后计算得到的子节点（局面）s_{t+1}的某个**对称局面**s_{t+1}'的胜率$f_\theta(s_{t+1}').v$平均值：

$$Q(s_t, a_t) = \frac{1}{N(s_t, a_t)} \sum f_\theta(s_{t+1}').v$$

注意，若θ和随机变换得到的局面s_{t+1}'确定，$f_\theta(s_{t+1}').v$的取值也是确定的。但上式中出现的神经网络参数θ以及每次随机变换得到的局面s_{t+1}'可能不同。

结合上面的讨论，动作选择策略可定义为：

$$a_t = argmax(Q(s_t, a_{t,i}) + U(s_t, a_{t,i}))$$

显然，β 越大，选择策略就越倾向于探索新的分支。

如果结点 s_t 对应的局面不是最终局面，即它至少有一个可行分支，则基于这一动作选择策略，我们可以找到一个最优的动作 a_t 和相应的下一结点 s_{t+1}，跳转到结点 s_{t+1} 后重复以上步骤即可。若 s_t 对应的局面是最终局面，直接结束扩展过程。

1.2. 更新

一次扩展过程确定了一条从博弈树根结点到某一个叶结点（仅对本次扩展过程而言）的有限长路径，路径可用一系列边 $(s_1, a_1), (s_2, a_2), \ldots, (s_N, a_N)$ 来表示。

更新指的是路径的每条边属性的更新，具体如下：（按顺序）

每条边的被经过次数加 1： $N(s_t, a_t) = N(s_t, a_t) + 1$；

每条边的价值更新：$Q(s_t, a_t) = \dfrac{(N(s_t, a_t) - 1)Q(s_t, a_t) + f_\theta(s_t + a_t).v}{N(s_t, a_t)}$

2. 自我对弈：

经过 1600 次模拟之后就可以基于扩展后的博弈树进行对弈，由于在同一次对弈的过程中，黑白双方将共用同一个落子策略，所以可以将其理解为 AI 的自我对弈。自我对弈的目的是为神经网络的训练提供样本。

自我对弈采用的落子策略是基于 MCTS 输出的概率分布 π 来设计的，π 定义如下：

对每一个结点 s_t，若它不是叶节点，则有

$$\pi_t = \left(\frac{N(s_t, a_{t,1})^{1/\tau}}{\sum N(s_t, a_{t,i})^{1/\tau}}, \frac{N(s_t, a_{t,2})^{1/\tau}}{\sum N(s_t, a_{t,i})^{1/\tau}}, \ldots, \frac{N(s_t, a_{t,M_t})^{1/\tau}}{\sum N(s_t, a_{t,i})^{1/\tau}} \right)$$

其中，τ 是温度参数，τ 的取值越小，分布 π_t 越集中，落子策略越谨慎。在整个训练的过程中，τ 的取值一开始定为 1 以保证落子策略有足够的探索能力，在对弈了一定次数之后，τ 的取值要逐渐减小直到趋于 0，使得落子策略趋于谨慎、保守。

上式可简记为：

$$\pi_t = \alpha_\theta(s_t)$$

若它是叶节点，则有

$$z = \alpha_\theta(s_t)$$

特别地，当 s_t 对应的局面是一盘棋的最终局面时，z 的取值为 1 或 0 或 0.5，表示这盘棋的胜者是黑方或白方或平局。否则 z 不能被赋值。

自我对弈的落子策略定义为：对每一个结点 s_t，真正的落子动作 a_t 的选择服从分布 π_t。

正如上面所讨论的，由于博弈树的扩展不完整，因此每一次对弈不一定能决出胜负（有可能到达的叶节点对应的局面不是最终局面，z 不能被赋值）。但训练网络必须要有每一盘棋的胜负情况，所以我们只记录能决出胜负的棋局。

按照上面定义的落子策略进行一局有胜负结果的对弈后，我们下一步要做的就是构建可供神经网络训练的样本。前文提到，神经网络的输入输出可表为 $(p,v)=f_\theta(s)$，故将样本的基本单元设为元组 (s,p,v,π,w)，其中 $w=|z-c|$，c 为在局面 s 下即将走子的一方，z 为这次对弈的胜方，那么每一次对弈的完整过程都可化为一系列样本元组：

$$(s_1,p_1,v_1,\pi_1,w_1),(s_2,p_2,v_2,\pi_2,w_2),...(s_N,p_N,v_N,\pi_N,w_N)$$

每一个元组都将存入记忆集 M 中。当自我对弈的次数充分多之后，就可以用 M 中的样本训练神经网络了。

注：M 是一个有容量上限的队列，容量设为 100,000。

3. 训练网络：

给定样本 (s,p,v,π,w)，训练的目标是使：

1）神经网络预测的落子概率分布趋于 MCTS 输出的落子概率分布；

2）在局面 s 下，神经网络预测的即将落子一方胜率尽可能符合本次对弈的胜负结果。

因此可构造损失函数如下：

$$L=(w-v)^2-\pi\ln(p)+c\|\theta\|^2$$

其中交叉熵 $-\pi\ln(p)$ 能够刻画分布 π 和分布 p 的相似性，$c\|\theta\|^2$ 是正则项，防止过拟合。

常数 $c>0$ 一般取 $c=10^{-4}$。

每次训练，都从记忆集 M 中随机抽取 2048 个样本元组，并采用随机梯度下降更新参数 θ。其中，动量参数 $\gamma=0.9$，学习率 α 的取值如下表：

Thousands of steps	Learning Rate
0-400	10^-2
400-600	10^-3
>600	10^-4

测试

不同于自对弈，在测试时 AI 的落子策略不是随机的，它总会选择目前的最优走法，*i.e.*

$$a_t = argmax(N(s_t, a_{t,i})), \forall t$$