AlphaGo Zero——它的工作原理和原因

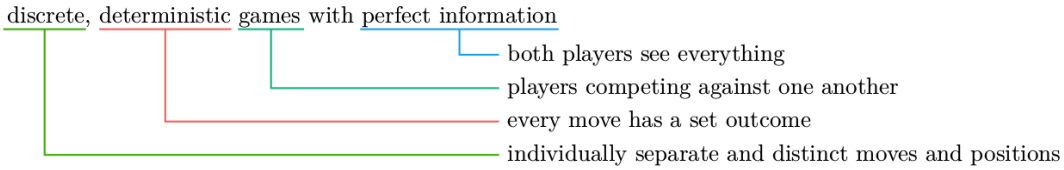
2017 年 11 月 2 日通过蒂姆

2016 年 3 月，DeepMind 的 AlphaGo 成为第一个击败人类顶级围棋棋手的人工智能，引起了轰动。这个版本的 AlphaGo - AlphaGo Lee - 在训练过程中使用了大量世界上最好的围棋棋手。 几天前发布了一篇新论文，详细介绍了一种新的神经网络——AlphaGo Zero——它不需要人类向它展示如何下围棋。它不仅胜过所有以前的围棋棋手，无论是人类还是机器，而且在仅仅三天的训练时间后就达到了这一点。本文将解释它的工作原理和原因。

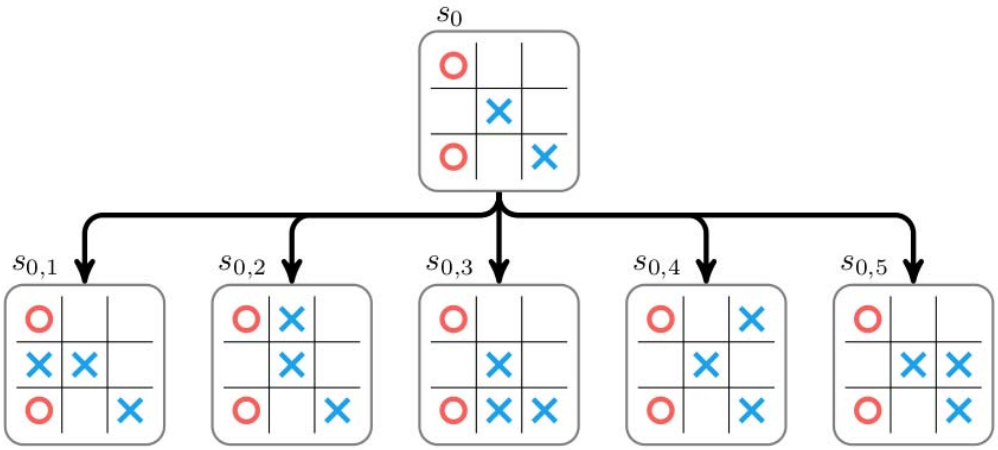
蒙特卡罗树搜索

编写机器人来玩具有完美信息的离散、确定性游戏的首选算法是蒙特卡罗树搜索 (MCTS)。玩围棋、国际象棋或跳棋等游戏的机器人可以通过尝试所有这些动作，然后检查对手的所有可能的反应，之后所有可能的动作等来确定它应该采取什么行动。对于像围棋这样的游戏，尝试的动作增长非常快。

蒙特卡洛树搜索将根据其认为的好坏有选择地尝试走法，从而将精力集中在最有可能发生的走法上。



从技术上讲，该算法的工作原理如下。进行中的游戏处于初始状态s_0，现在轮到机器人玩了。机器人可以从一组操作中进行选择$a \in A$。蒙特卡洛树搜索从一个由单个节点组成的树开始s_0. 通过尝试每个动作来*扩展*此节点$s_0 \in S$并为每个动作构造一个对应的子节点。下面我们展示了井字棋游戏的这个扩展：



然后必须确定每个新子节点的值。在子节点的游戏*推出*通过随机取从子状态移动，直到达到一个双赢，损失或领带。胜利的得分为

$$+1$$

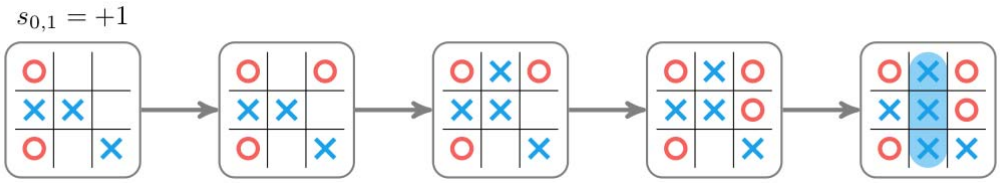
，损失在

$$-1$$

，并在

$$0$$

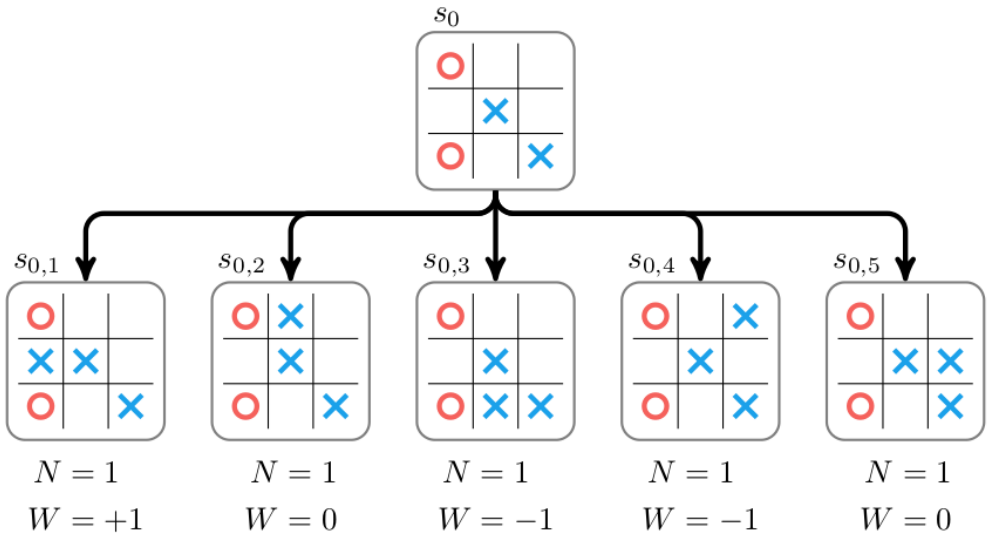
.



上面给出的第一个孩子的随机推出估计值

$$+1$$

. 此值可能并不代表最佳播放效果 - 它可能会根据推出的进度而有所不同。一个人可以不智能地运行卷展，绘制随机均匀移动。人们通常可以通过遵循一种更好的——尽管仍然是典型的随机策略，或者通过直接估计状态的价值来做得更好。稍后再谈。



上面我们展示了每个子节点的近似值的扩展树。注意我们存储了两个属性：累计值

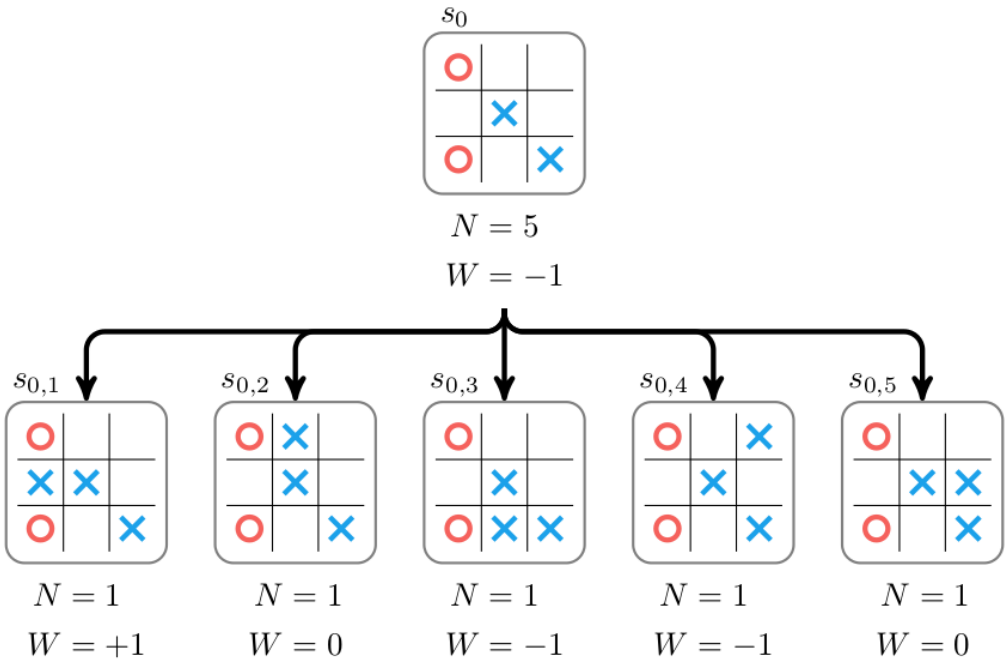
$$W$$

以及在该节点或该节点以下运行部署的次数，

$$N$$

. 我们只访问了每个节点一次。

然后通过增加父节点的值和访问计数，将来自子节点的信息沿树向上传播。然后将其累积值设置为其子项的总累积值：



蒙特卡罗树搜索继续进行多次迭代，包括选择一个节点、扩展它并传播新信息。已经涵盖了扩展和传播。

Monte Carlo 树搜索不会扩展所有叶节点，因为这会非常昂贵。相反，选择过程会选择在利润丰厚（具有高估计值）和相对未开发（具有低访问量）之间取得平衡的节点。

通过从根节点向下遍历树来选择叶节点，始终选择子节点

$$一世$$

具有最高上置信树 (UCT) 分数：

$$你_{一世} \frac{宽_{一世}}{N_{一世}} + c\sqrt{\frac{输入 N_p}{N_{一世}}}$$

在哪里

$$宽_{一世}$$

是累计值

$$一世$$

第一个孩子，

$$N_{一世}$$

是访问计数

$$一世$$

第一个孩子，和

$$N_p$$

是父节点的访问计数。参数

$$ç \geq 0$$

控制选择有利可图的节点之间的权衡（低

$$C$$

) 并探索访问次数较少的节点（高

$$C$$

）。它通常是根据经验设置的。

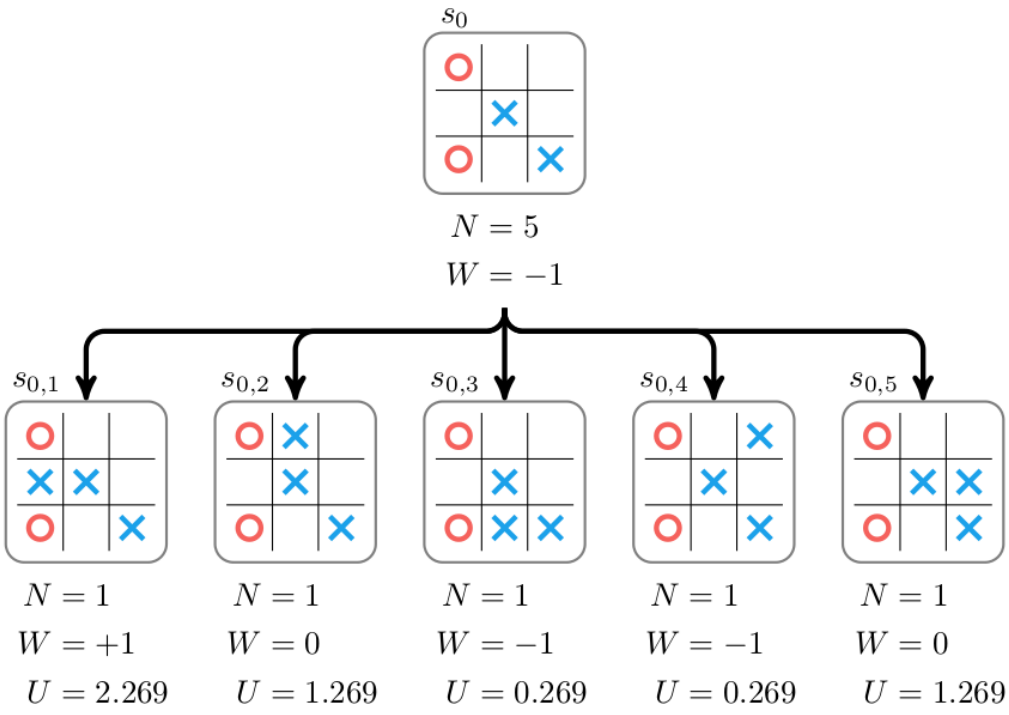
UCT 分数 (

你

's) 对于井字棋树

$$c = 1$$

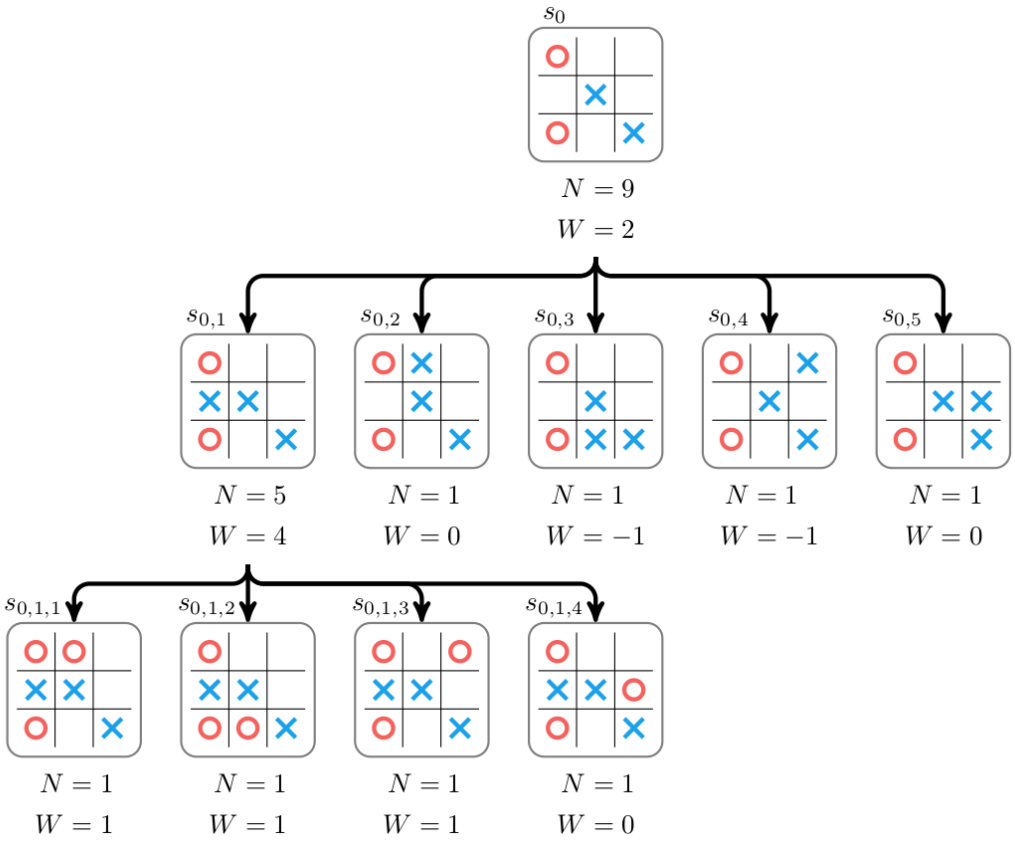
是：



在这种情况下，我们选择第一个节点，

松$_{,1}$

.（在平局的情况下，可以随机打破平局或只选择第一个下注节点。）该节点被扩展，并且值被传播回：



注意每个累计值

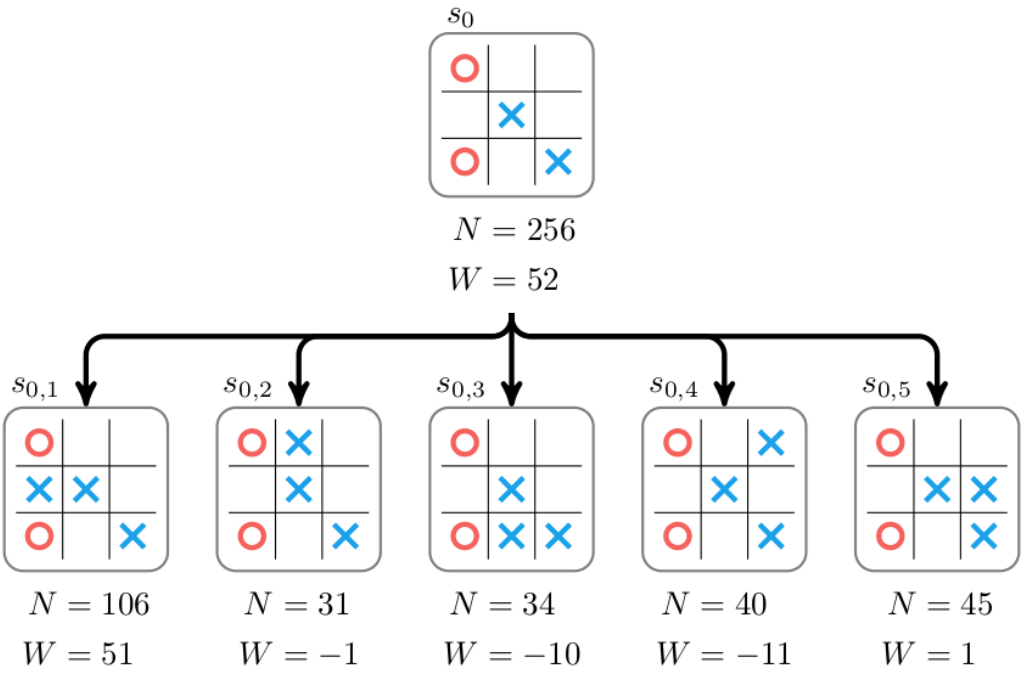
宽

反映 X 是赢还是输。在选择过程中，我们跟踪是轮到 X 还是 O 移动，并翻转

宽

每当轮到 O 时。

我们继续运行蒙特卡罗树搜索的迭代，直到时间用完。树逐渐扩大，我们（希望）探索可能的移动，确定最好的移动。然后，机器人实际上通过选择访问次数最多的第一个孩子在原始真实游戏中采取行动。例如，如果我们的树的顶部看起来像：



然后机器人会选择第一个动作并继续

$$松, 1$$

.

通过专家政策提高效率

象棋和围棋之类的游戏具有非常大的分支因子。在给定的游戏状态中，有许多可能的操作可以采取，这使得充分探索未来的游戏状态变得非常困难。因此，估计有

$$10^{46}$$

国际象棋棋盘状态，围棋按传统方式下棋

$$19 \times 19$$

板有周围

$$10^{170}$$

（井字游戏只有

$$5478$$

状态）。

使用普通蒙特卡罗树搜索进行移动评估不够有效。我们需要一种方法来进一步将注意力集中在有价值的行动上。

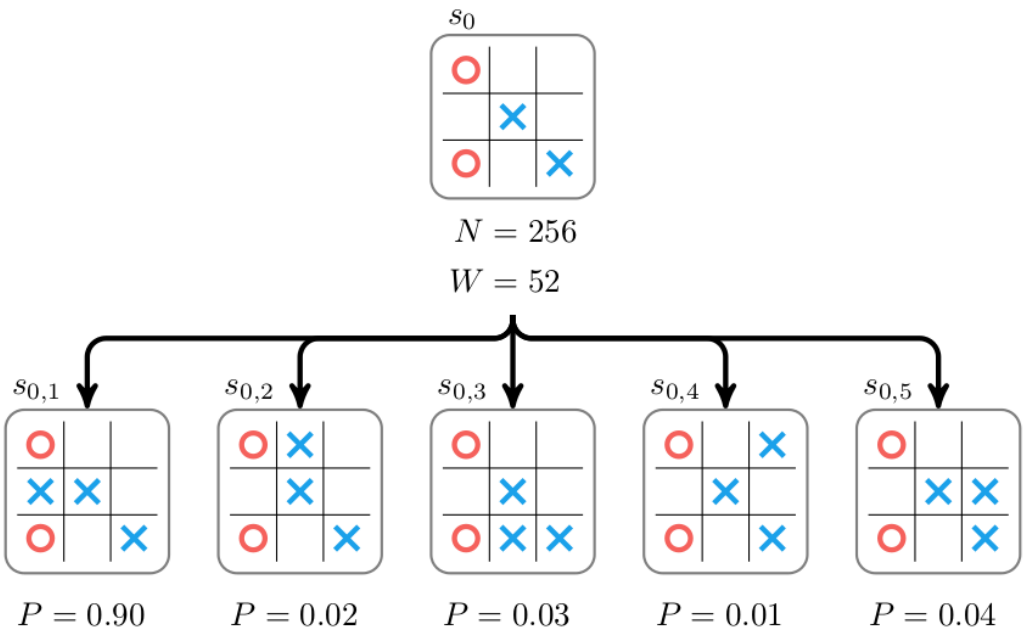
假设我们有一个*专家策略*

π

那，对于给定的状态

s

，告诉我们专家级玩家做出每个可能动作的可能性。对于井字游戏示例，这可能如下所示：



每个

a

是选择的概率

a

第一次行动

a

给定根状态

s

.

如果专家策略真的很好，那么我们可以通过根据产生的概率直接绘制我们的下一步动作来产生一个强大的机器人

π

，或者更好的是，以最高的概率采取行动。不幸的是，获得专家策略是困难的，并且验证一个人的策略是最优的也很困难。

幸运的是，可以通过使用一种改进形式的蒙特卡罗树搜索来改进策略。这个版本还会根据策略存储每个节点的概率，这个概率用于在选择时调整节点的分数。DeepMind 使用的概率上置信树分数是：

$$你_{一世} \frac{宽_{一世}}{N_{一世}} + c磷_{一世}\sqrt{\frac{输入 N_p}{1 + N_{一世}}}$$

和以前一样，分数在始终产生高分的节点和未探索的节点之间进行权衡。现在，节点探索是由专家策略引导的，偏向于移动专家策略认为可能的探索。如果专家策略真的很好，那么蒙特卡洛树搜索有效地专注于游戏状态的良好演变。如果专家策略很差，那么蒙特卡洛树搜索可能会关注游戏状态的不良演变。无论哪种方式，在样本数量变大的限制下，节点的价值由赢/输比决定

$$宽_{一世} / N_{一世}$$

，像以前一样。

通过价值近似提高效率

第二种形式的效率可以通过避免昂贵且可能不准确的随机部署来实现。一种选择是使用上一节中的专家策略来指导随机部署。如果政策是好的，那么推出应该反映更现实的、专家级的游戏进程，从而更可靠地估计一个状态的价值。

第二种选择是完全避免 rollouts，并使用值逼近函数直接逼近状态值

$$\hat{宽}(x)$$

. 这个函数接受一个状态并直接计算一个值

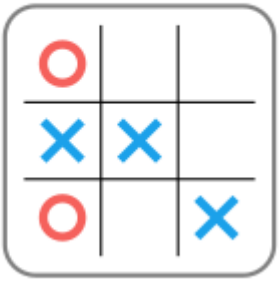
$$[-1, 1]$$

，无需进行发布。显然，如果

$$\hat{宽}$$

是真实值的一个很好的近似值，但可以比 rollout 执行得更快，然后可以在不牺牲性能的情况下节省执行时间。

$$\hat{W}(s_{0,1}) = 0.1$$



值近似可以与专家策略结合使用，以加速蒙特卡罗树搜索。一个严重的问题仍然存在——如何获得专家策略和价值函数？是否存在用于训练专家策略和价值函数的算法？

Alpha 零神经网络

Alpha Zero 算法通过加速蒙特卡罗树搜索与自己玩游戏，随着时间的推移产生越来越好的专家策略和价值函数。专家政策

$$\pi$$

和近似值函数

$$\hat{\text{宽}}$$

两者都由深度神经网络表示。事实上，为了提高效率，Alpha Zero 使用了一个神经网络

$$F$$

它接受游戏状态并产生下一步的概率和近似状态值。（从技术上讲，它需要前八个游戏状态和一个指示它轮到谁的指示器。）

$$F(\ s \ ) \ \rightarrow \ [\ \mathbf{p} \ , \ W]$$

搜索树中的叶子通过用神经网络评估它们来扩展。每个孩子都初始化为

$$N \ = \ 0$$

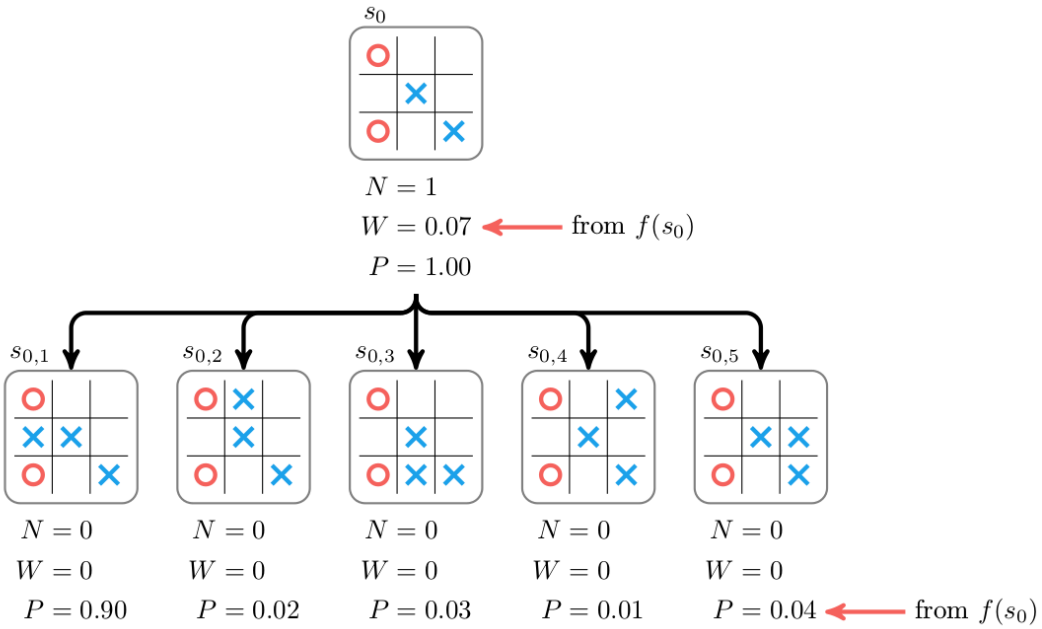
，

$$\text{宽} \ = \ 0$$

，与

$$\text{磷}$$

对应于网络的预测。扩展节点的值设置为预测值，然后将该值备份到树中。

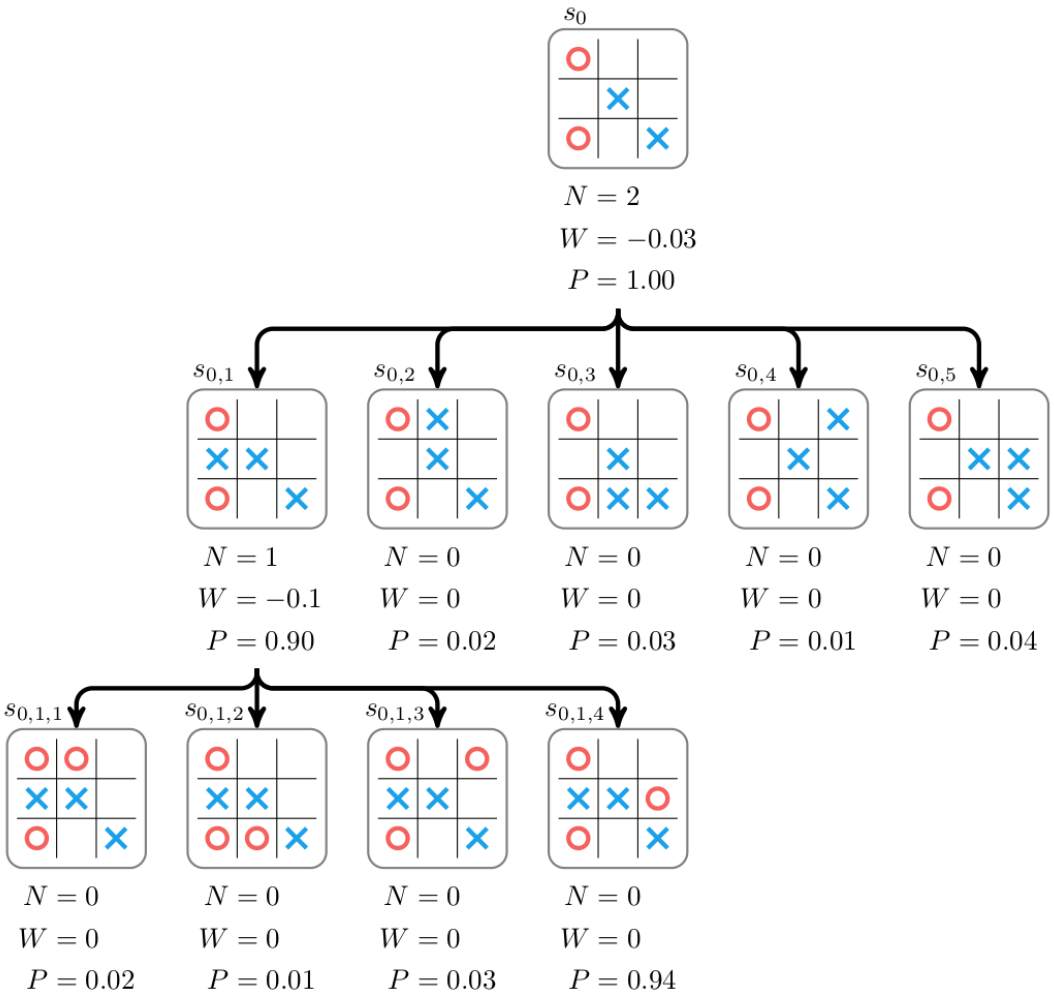


选择和备份不变。简单地说，在备份期间，父母的访问次数增加，其值增加

宽

.

另一个选择、扩展和备份步骤之后的搜索树是：



Alpha Zero算法的核心思想是可以提高神经网络的预测能力，利用蒙特卡罗树搜索产生的play来提供训练数据。通过训练预测概率来改进神经网络的策略部分

$$\mathbf{p}$$

为了

$$\mathbf{p}$$

匹配改进的概率

$$\boldsymbol{\pi}$$

从运行蒙特卡罗树获得

$$\mathbf{p}$$

. 运行蒙特卡罗树搜索后，改进后的策略预测为：

$$\pi_{\text{一世}} \ N_{\text{一世}}^{1 / \tau}$$

对于一个常数

$$\tau$$

.
的值

$$\tau$$

根据蒙特卡洛树搜索评估选择最佳移动的接近零产生策略。

通过训练预测值以匹配游戏的最终赢/输/平局结果来改进神经网络的值部分，

$$Z$$

. 他们的损失函数是：

$$(W- \ Z)^2 + \boldsymbol{\pi}^{\top} \text{输入} \, \mathbf{p} \ + \lambda \| \ \boldsymbol{\theta} \|_2^2$$

在哪里

$$(W- \ Z)^2$$

是价值损失，

$$\pi^{\top} \text{输入} \, \mathbf{p}$$

是保单损失，并且

$$\lambda \| \theta \|_2^2$$

是一个带有参数的额外正则化项

$$\lambda \geq 0$$

和

$$\boldsymbol{\theta}$$

表示神经网络中的参数。

训练完全是在自我游戏中完成的。从一组随机初始化的神经网络参数开始

$$\boldsymbol{\theta}$$

. 然后，这个神经网络被用于它自己玩的多个游戏中。在这些游戏的每一个中，对于每一步，蒙特卡罗树搜索用于计算

$$\pi$$

. 每场比赛的最终结果决定了这场比赛的价值

$$Z$$

. 参数

$$\boldsymbol{\theta}$$

然后通过使用梯度下降（或任何更复杂的加速下降方法 - Alpha Zero 使用带有动量和学习率退火的随机梯度下降。）在损失函数上改进随机选择的状态。

结束评论

就是这样。DeepMind 的人贡献了一种干净且稳定的学习算法，该算法仅使用来自自我游戏的数据有效地训练游戏代理。虽然当前的零算法仅适用于离散游戏，但未来是否会扩展到 MDP 或其部分可观察的对应物将会很有趣。

有趣的是，人工智能领域的发展速度如此之快。那些声称我们将能够及时看到机器人霸主到来的人应该注意——这些 AI 只会在短暂的瞬间达到人类水平，然后从我们身边飞过我们进入超人的领域，永远不会回头。

此条目发表在未分类。为永久链接添加书签。

← Apteryx 飞行数据与 C3.js 我们如何写教科书→

2 关于“ AlphaGo Zero – 如何以及为什么工作”的 想法

Pingback：如何使用 Python 和 Keras 构建您自己的 AlphaZero AI | 复制粘贴程序员

Pingback：AlphaZero 和人类知识的诅咒 | 复制粘贴程序员

评论被关闭。

元

- 登录