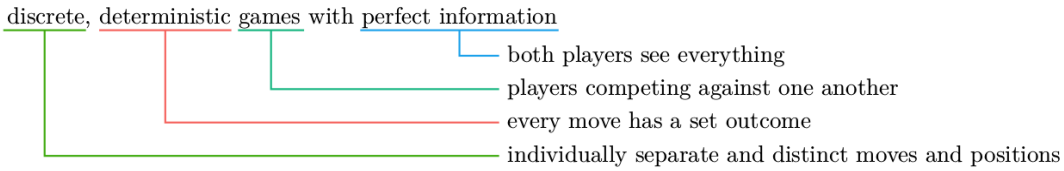
AlphaGo 零 - 如何以及为何工作

2017 年 11 月 2 日通过蒂姆

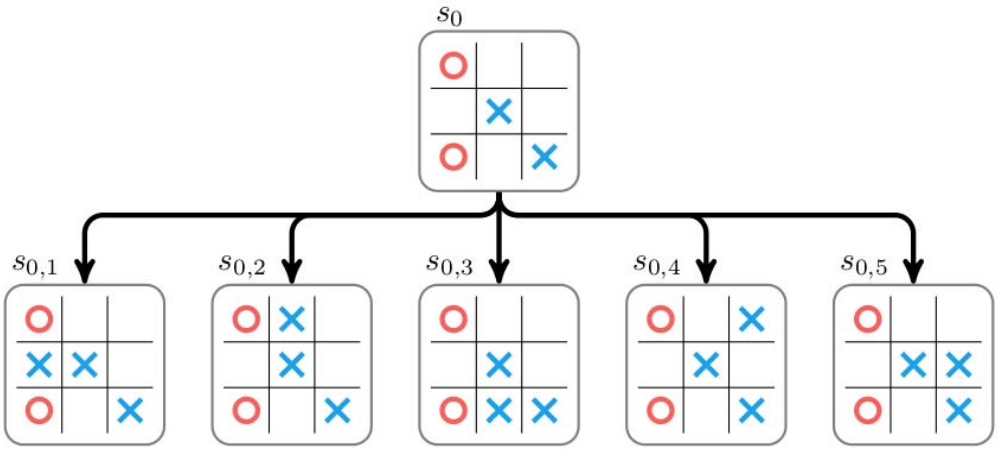
DeepMind 的 AlphaGo 在 2016 年 3 月成为第一个击败人类顶级围棋棋手的 AI 时引起了轰动。这个版本的 AlphaGo——AlphaGo Lee——在训练过程中使用了来自世界上最优秀棋手的大量围棋游戏。 几天前发布了一篇新论文，详细介绍了一种新的神经网络——AlphaGo Zero——它不需要人类向它展示如何下围棋。它不仅超越了所有以前的围棋选手，无论是人类还是机器，而且仅在三天的训练时间后就达到了这一点。本文将解释它的工作原理和原因。

蒙特卡洛树搜索

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洛树搜索 (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动作，然后检查对手所有可能的反应，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，尝试的动作增长得非常快。
蒙特卡洛树搜索将根据它认为的好坏有选择地尝试移动，从而将精力集中在最有可能发生的移动上。



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，轮到机器人上场了。机器人可以从一组动作中进行选择一个。蒙特卡洛树搜索从一棵由单个节点组成的树开始s_0。通过尝试每个操作来*扩展*此节点一个，并为每个动作构造一个对应的子节点。下面我们展示了这款井字游戏的扩展：



然后必须确定每个新子节点的值。子节点中的游戏通过从子状态随机采取移动来*展开*，直到达到赢、输或平局。胜利得分为

$$+1$$

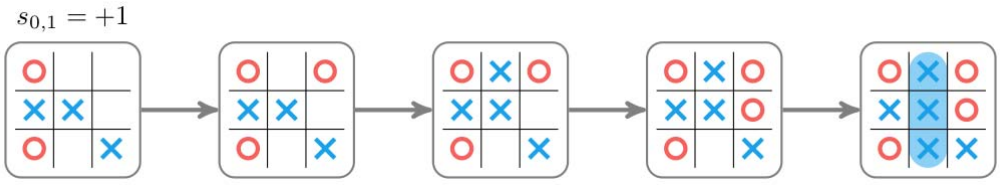
，损失在

$$-1$$

，并且关系在

$$0$$

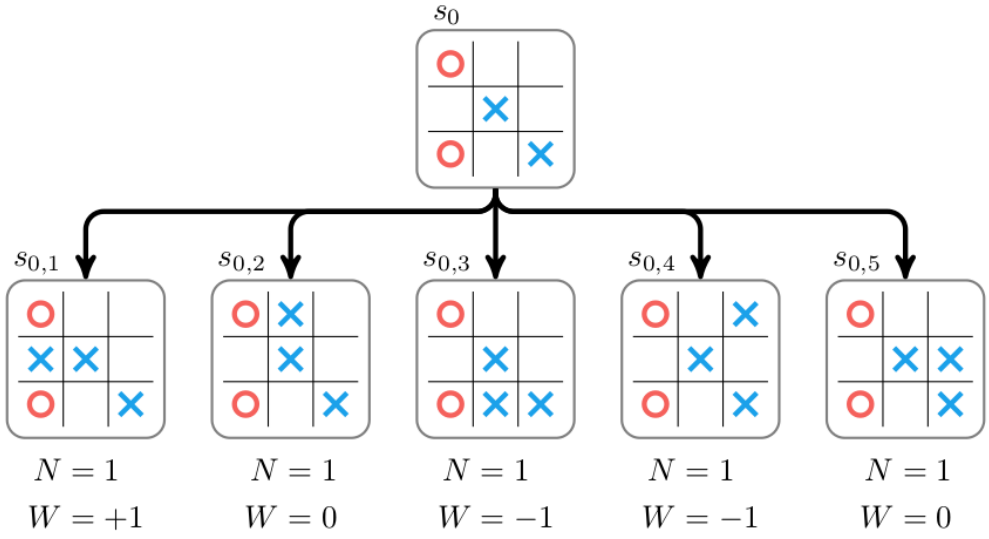
.



上面给出的第一个孩子的随机推出估计值为

$$+1$$

. 此值可能不代表最佳播放 - 它可能会根据部署的进展情况而有所不同。一个人可以不智能地进行部署，随机均匀地绘制移动。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。



上面我们显示了每个子节点具有近似值的扩展树。注意我们存储两个属性：累计值

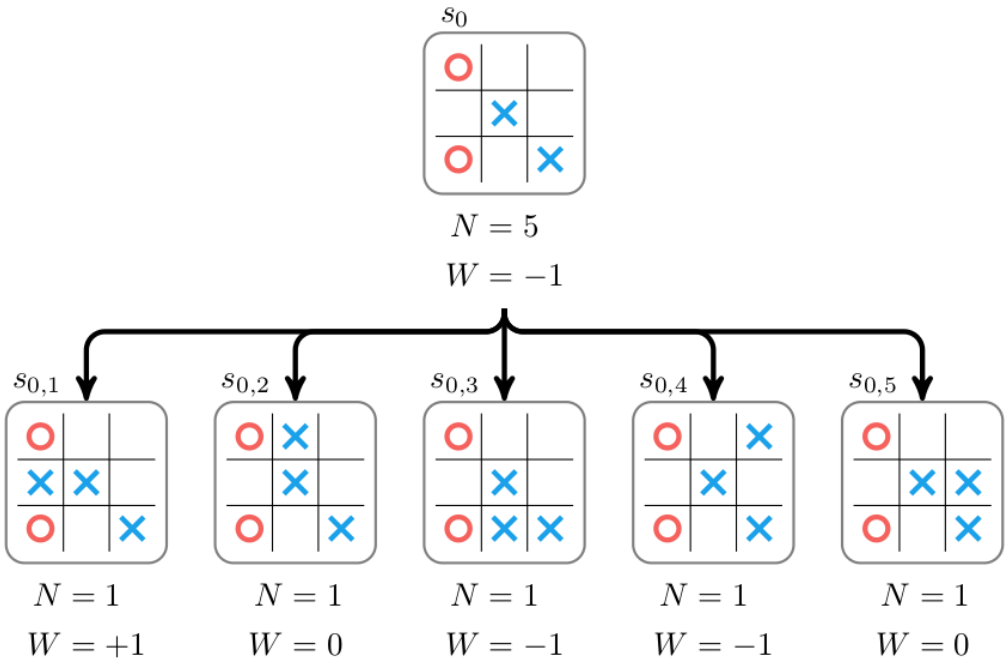
$$W$$

以及在该节点或该节点下运行推出的次数，

$$\tilde{n}$$

. 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次数，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：



蒙特卡洛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。扩展和传播已经涵盖。

蒙特卡洛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（具有低访问次数）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

$$\text{—世}$$

具有最高的上置信树 (UCT) 分数：

$$\ddot{\text{u}}_{\text{—世}} \frac{W_{\text{—世}}}{\tilde{\text{n}}_{\text{—世}}} + c\sqrt{\frac{\ln \tilde{\text{n}}_{p}}{\tilde{\text{n}}_{\text{—世}}}}$$

在哪里

$$W_{\text{—世}}$$

是的累计值

$$\text{—世}$$

第一个孩子，

$$\tilde{\text{n}}_{\text{—世}}$$

是访问次数

$$\text{—世}$$

第一个孩子，和

$$\tilde{\text{n}}_{p}$$

是父节点的访问次数。参数

$$c \geq 0 \quad \_ \quad \_$$

控制选择有利可图的节点之间的权衡（低

$$C$$

) 并探索访问次数少的节点（高

$$C$$

）。它通常是根据经验设置的。

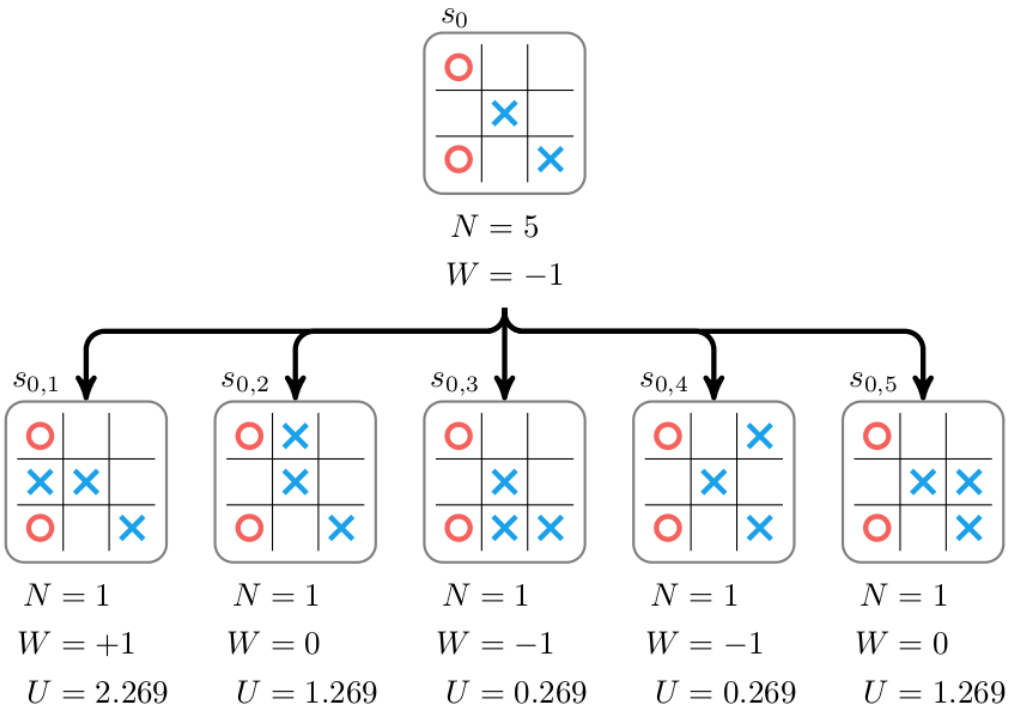
UCT 分数 (

ü

's) 用于井字游戏树

$$c = 1$$

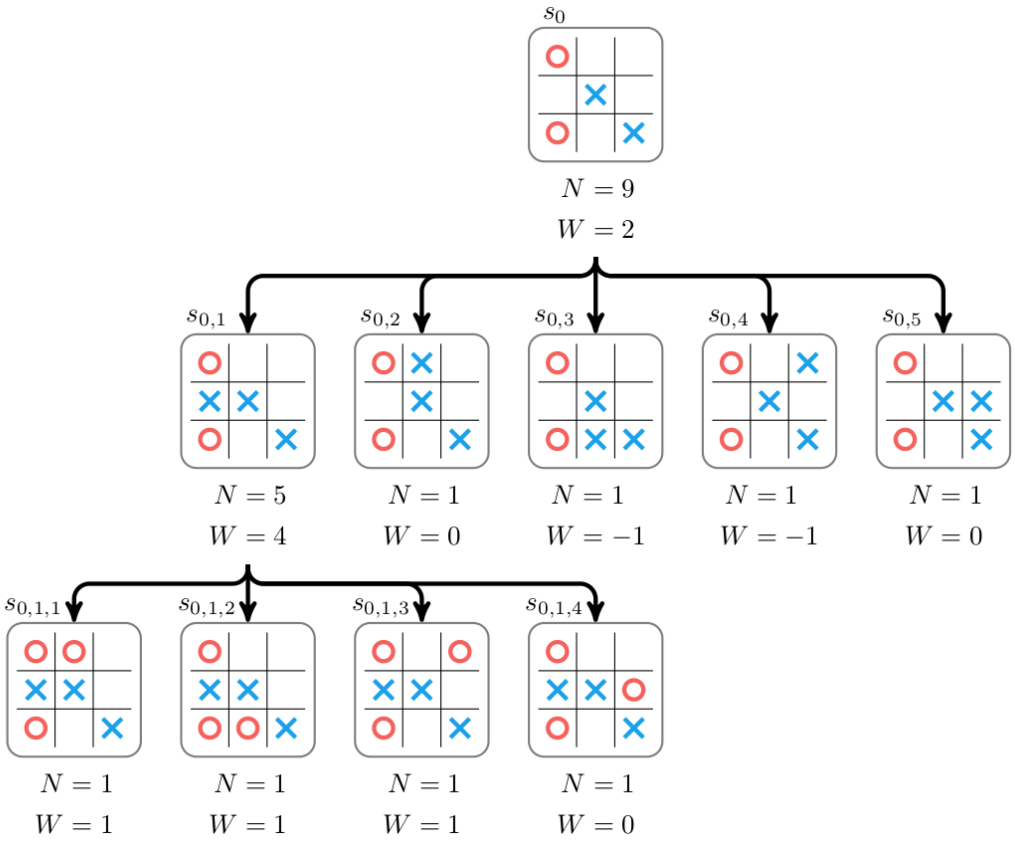
是：



在这种情况下，我们选择第一个节点，

$$s_{0\,,\,1}$$

.（如果出现平局，则可以随机打破平局，或者只选择第一个下注节点。）该节点被扩展并且值被传播回来：



请注意，每个累积值

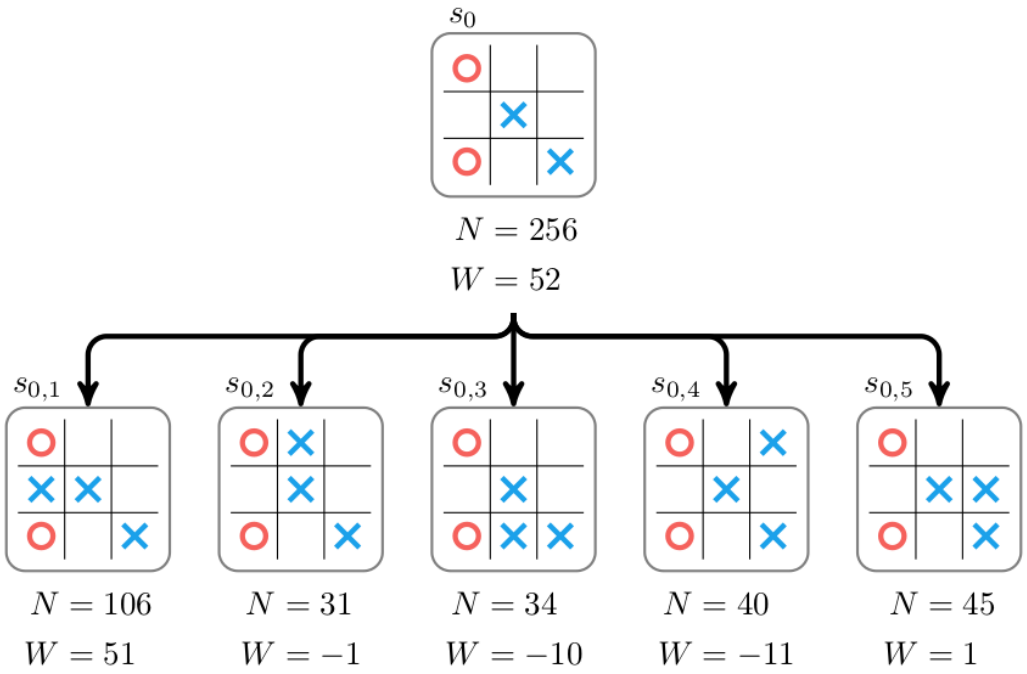
$$W$$

反映 X 的输赢。在选择过程中，我们跟踪是轮到 X 还是 O 移动，并翻转

$$W$$

每当轮到O时。

我们继续运行蒙特卡洛树搜索的迭代，直到时间用完。树逐渐扩展，我们（希望）探索可能的移动，确定最佳移动。然后，机器人实际上通过选择访问次数最多的第一个孩子来在原始的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

$$s_{0\,,\,1}$$

.

通过专家策略提高效率

国际象棋和围棋等游戏具有非常大的分支因子。在给定的游戏状态下，有许多可能的动作要采取，因此很难充分探索未来的游戏状态。结果，估计有

$$10^{46}$$

国际象棋中的棋盘状态和传统的围棋

$$19 \times 19$$

板子周围有

$$10^{170}$$

（井字游戏只有

$$5478$$

状态）。

使用 vanilla Monte Carlo 树搜索进行移动评估的效率不够高。我们需要一种方法来进一步将注意力集中在有价值的举措上。

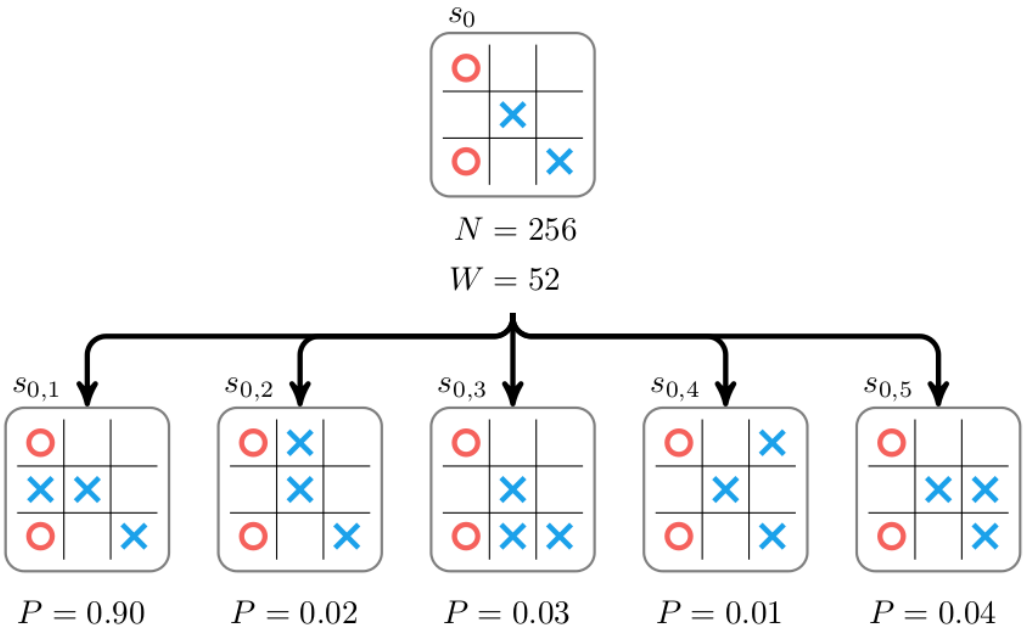
假设我们有一个*专家策略*

π

那，对于给定的状态

s

，告诉我们专家级玩家做出每一个可能的动作的可能性有多大。对于井字游戏示例，这可能如下所示：



其中每个

磷 $\pi(\text{一}|s_0)$

是选择的概率

一

动作

一

给定根状态

s_0

.

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个动作来产生一个强大的机器人

π

，或者更好的是，以最高概率采取行动。不幸的是，获得专家策略很困难，验证一个人的策略是否最优也很困难。

幸运的是，可以通过使用蒙特卡洛树搜索的修改形式来改进策略。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得分。DeepMind 使用的概率上置信树分数是：

$$\text{ü}_{\text{—世}} \frac{W_{\text{—世}}}{\tilde{\text{n}}_{\text{—世}}} + c\text{磷}\sqrt{\frac{\ln \tilde{\text{n}}_{p}}{1+\ \_\tilde{\text{n}}_{\text{—世}}}}$$

和以前一样，分数在持续产生高分的节点和未探索的节点之间进行权衡。现在，节点探索由专家策略引导，将探索偏向于专家策略认为可能的移动。如果专家策略真的很好，那么蒙特卡洛树搜索有效地关注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/输比支配

$$W_{\text{—世}}/\tilde{\text{n}}_{\text{—世}}$$

， 和以前一样。

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推出来实现。一种选择是使用上一节中的专家策略来指导随机推出。如果策略是好的，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推出，并使用值逼近函数直接逼近状态的值

$$\hat{W}(\ \times\ )$$

. 这个函数接受一个状态并直接计算一个值

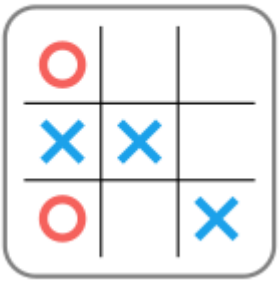
$$[\,-1\,,\ 1\,]$$

，不进行部署。显然，如果

$$\hat{W}$$

是真实值的一个很好的近似值，但可以比 rollout 更快地执行，因此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一起使用，以加速蒙特卡洛树搜索。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

随着时间的推移，Alpha Zero 算法通过加速蒙特卡洛树搜索与自己进行博弈，从而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

$$\hat{W}$$

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移动的概率和近似状态值。（从技术上讲，它需要前八种游戏状态和一个指示轮到它的指示器。）

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

通过使用神经网络评估它们来扩展搜索树中的叶子。每个孩子都初始化为

$$\tilde{n}\ =\ 0$$

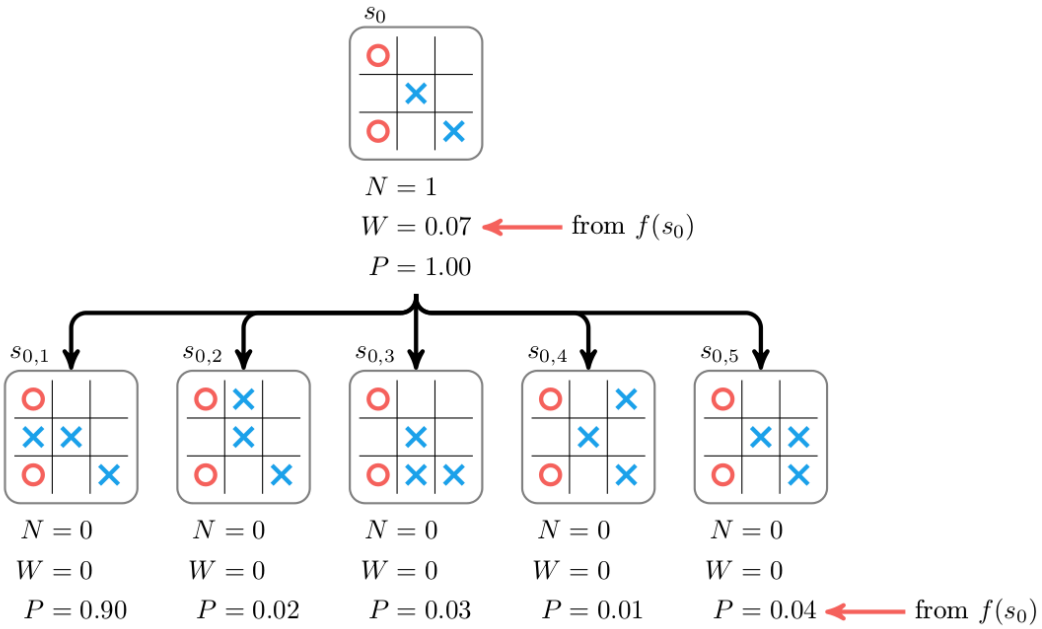
，

$$W\ =\ 0$$

，与

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。

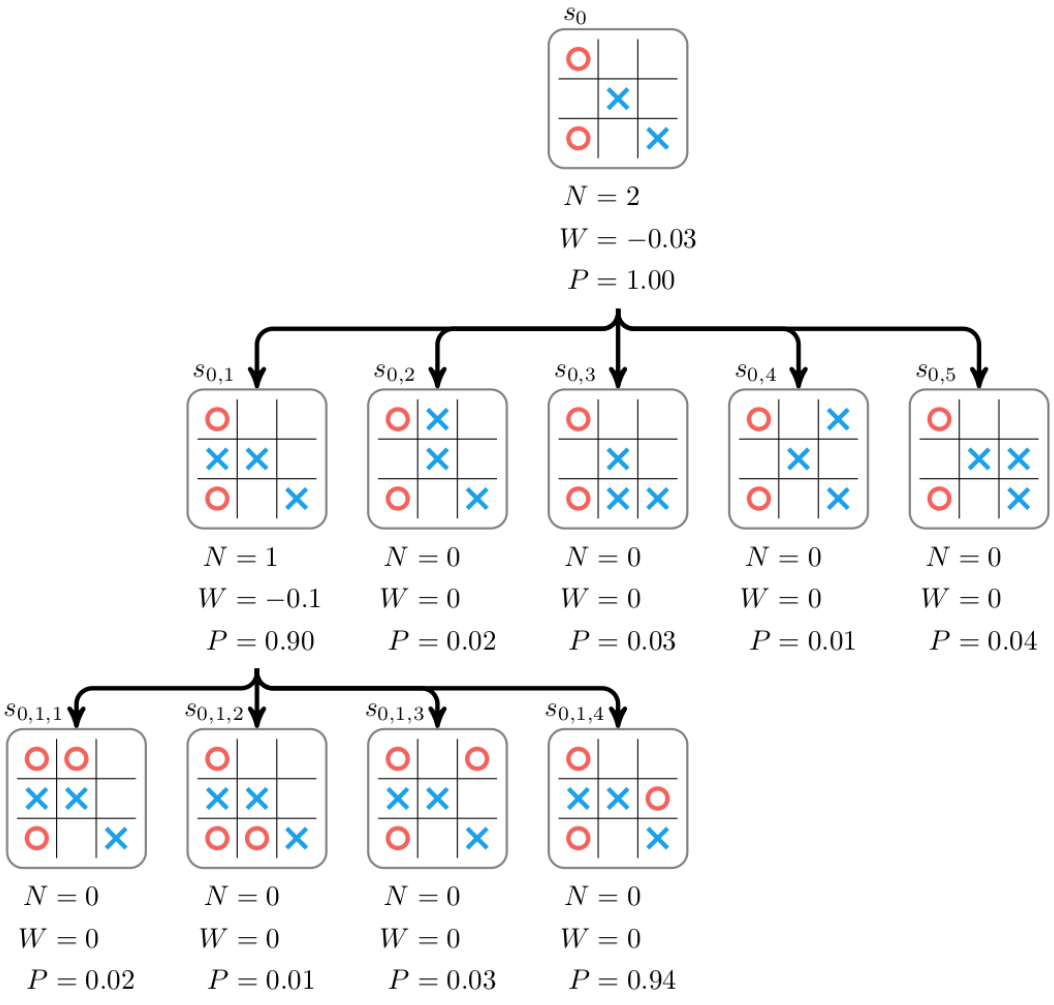


选择和备份不变。简而言之，在备份期间，父母的访问计数会增加，并且其值会根据

$$W$$

.

在另一个选择、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，利用蒙特卡洛树搜索产生的玩法来提供训练数据。通过训练预测概率来改进神经网络的策略部分

$$\mathbf{p}$$

为了

$$s_0$$

匹配改进的概率

$$\boldsymbol{\pi}$$

从运行蒙特卡洛树获得

$$s_0$$

. 运行蒙特卡洛树搜索后，改进后的策略预测为：

$$\pi_{\text{—世}} \tilde{\text{n}}^{1/\tau}_{\text{一世}}$$

对于一个常数

$$\tau$$

.
的价值观

$$\tau$$

接近零产生根据蒙特卡洛树搜索评估选择最佳移动的策略。

通过训练预测值以匹配游戏的最终赢/输/平结果，提高了神经网络的价值部分，

$$Z$$

. 他们的损失函数是：

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

在哪里

$$(W- Z)^2$$

是价值损失，

$$\pi^\top \ln \mathbf{p}$$

是策略损失，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。一个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。在这些游戏中，对于每一步，蒙特卡洛树搜索用于计算

$$\pi$$

. 每场比赛的最终结果决定了该场比赛的价值

$$Z$$

. 参数

$$\boldsymbol{\theta}$$

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

结束评论

就是这样。DeepMind 的人们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，但它是否会在未来扩展到 MDP 或它们的部分可观察对应物将会很有趣。

有趣的是，人工智能领域的发展速度有多快。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——这些人工智能只会在短暂的瞬间达到人类水平，然后冲过我们进入超人类领域，永不回头。

此条目发表在未分类。为永久链接添加书签。

←使用 C3.js 的 Apteryx 飞行数据 我们如何编写教科书→

关于“ AlphaGo 零 - 如何以及为什么工作” 的 2 个想法

Pingback：如何使用 Python 和 Keras 构建自己的 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