

# 如何使用Python和Keras构建自己的AlphaZero AI

在这篇文章中，我将试图涵盖三个方面：

1. AlphaZero是人工智能的巨大进步的两个原因
2. 如何复制AlphaZero方法来玩游戏Connect4
3. 如何调整代码以插入其他游戏



首先，简要介绍一个新平台，网络，一个数据科学家可以找到与企业签订的有偿合同项目的地方！

单击“注册”开始创建您的个人资料。

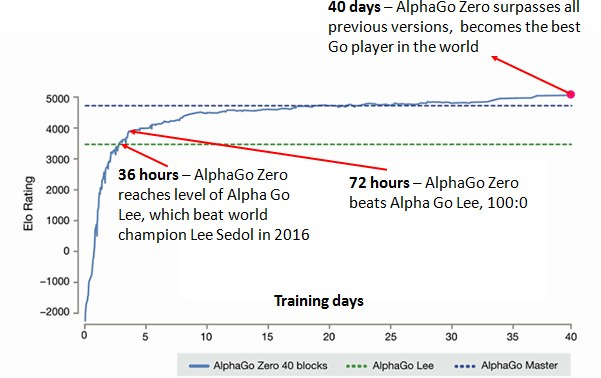
### AlphaGo→AlphaGo Zero→AlphaZero

2016年3月，Deepmind的AlphaGo在2亿多人观看的系列赛中，以4比1击败了18次世界围棋冠军Lee Sedol。一台机器已经学会了一种超人的围棋策略，这是一项以前认为不可能实现的壮举，或者至少离完成还有十年的时间。



这本身就是一个了不起的成就。然而，2017年10月18日，DeepMind又有了一个巨大的飞跃。

该报公布了一种新的算法变体AlphaGo Zero，它已经击败了AlphaGo 100-0。令人难以置信的是，它仅仅是通过自我游戏来学习的，开始“表格拉萨”（blank state），并逐渐找到能够击败自身先前化身的策略。不再是建立超级人工智能所需的人类专家游戏数据库。



仅仅48天后，2017年12月5日，DeepMind发布了另一篇论文，展示了AlphaGo Zero如何在国际象棋和shogi上击败世界冠军项目StockFish和Elmo。从第一次看游戏到成为世界上最好的电脑程序，整个学习过程只用了不到24小时。

有了这个，AlphaZero就诞生了 - 一种通用的算法，它可以在没有任何人类专家策略的先验知识的情况下，快速地精通某件事情。

这项成就有两个惊人之处：

1. AlphaZero不需要任何人类专业知识作为输入

这有多重要，怎么强调都不为过。这意味着AlphaGo Zero的基本方法可以应用于任何具有完美信息的游戏（游戏状态在任何时候都是双方都完全知道的），因为除了游戏规则之外，不需要任何先前的专业知识。

这就是为什么DeepMind在最初的AlphaGo Zero论文发表48天后才发表国际象棋和shogi论文的原因。从字面上讲，所有需要改变的是描述游戏机制的输入文件，以及调整与神经网络和蒙特卡罗树搜索相关的超参数。

1. 这个算法非常优雅

如果AlphaZero使用了世界上只有少数人能理解的超复杂算法，那将仍然是一个令人难以置信的成就。让它与众不同的是，论文中的许多观点实际上远没有以前的版本复杂。在它的核心，有以下美丽简单的学习咒语：

在精神上玩可能的未来场景，优先考虑有前途的道路，同时也考虑其他人如何最有可能对你的行动作出反应，并继续探索未知。

在达到一种不熟悉的状态后，评估你认为这个位置有多有利，并将分数通过导致这一点的心理路径中的先前位置重新串接回来。

在你思考完未来的可能性之后，采取你探索最多的行动。

在游戏结束时，回去评估一下你对未来位置价值判断错误的地方，并相应地更新你的理解。

听起来不是很像你学玩游戏吗？当你做一个糟糕的动作时，要么是因为你误判了结果位置的未来价值，要么是你误判了对手做某个动作的可能性，所以没有考虑去探索这种可能性。这正是AlphaZero学习游戏的两个方面。

### 如何建立自己的AlphaZero

首先，请查看以获得对AlphaGo Zero工作原理的高级理解。当我们遍历代码的每个部分时，有必要引用它。还有一篇很好的文章更详细地解释了AlphaZero是如何工作的。

#### 守则

克隆Git存储库，其中包含我将引用的代码。

要开始学习过程，请运行run.ipynbJupyter笔记本中的前两个面板。一旦它建立了足够的游戏位置来填满记忆，神经网络就会开始训练。通过额外的自我游戏和训练，它将逐渐在预测游戏价值和从任何位置的下一步行动方面变得更好，从而产生更好的决策和更聪明的整体游戏。

现在我们将更详细地查看代码，并显示一些结果，这些结果表明随着时间的推移，人工智能越来越强大。

N.B - 这是我自己根据上述文献中提供的信息对AlphaZero如何工作的理解。如果下面的任何一个是错误的，道歉，我会努力纠正它！

#### 连接4

我们的算法将学习的游戏是Connect4（或连续4个）。不太复杂…但仍然有4531985219092个游戏位置的总数。



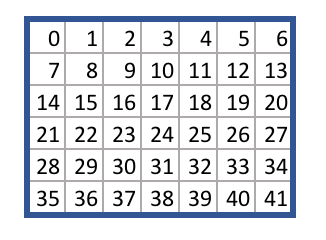
游戏规则很简单。玩家轮流在任何可用列的顶部输入他们的颜色。第一个连续获得四种颜色的玩家获胜。如果整个网格被填充而没有创建一行四个，则将绘制游戏。

下面是组成代码库的关键文件的摘要：

#### 游戏.py

此文件包含Connect4的游戏规则。

每个正方形都分配了一个从0到41的数字，如下所示：



py文件给出了在给定选定动作的情况下从一个游戏状态移动到另一个游戏状态的逻辑。例如，给定空棋盘和动作38，takeAction方法返回一个新的游戏状态，起始玩家的棋子位于中间列的底部。

您可以将game.py文件替换为任何符合相同API的游戏文件，算法将主要根据您给定的规则，通过自玩来学习策略。

#### 运行.ipynb

它包含启动学习过程的代码。它加载游戏规则，然后遍历算法的主循环，该循环由三个阶段组成：

1. 自我发挥
2. 神经网络再训练
3. 评价神经网络

在这个循环中有两个代理，最佳玩家和当前玩家。

best\_播放器包含性能最好的神经网络，用于生成自播放记忆。现在的球员然后在这些记忆上重新训练神经网络，然后与最好的球员对抗。如果它赢了，最好的玩家里面的神经网络会被切换到当前玩家里面的神经网络，循环会再次开始。

#### 代理.py

这包含代理类（游戏中的玩家）。每个玩家都用自己的神经网络和蒙特卡罗搜索树初始化。

模拟方法运行Monte Carlo树搜索过程。具体来说，代理移动到树的叶节点，用它的神经网络评估节点，然后通过树向上填充节点的值。

act方法多次重复模拟，以了解从当前位置移动哪个最有利。然后它将选择的动作返回游戏，以执行该动作。

重放方法利用以前游戏的记忆，重新训练神经网络。

#### 模型.py



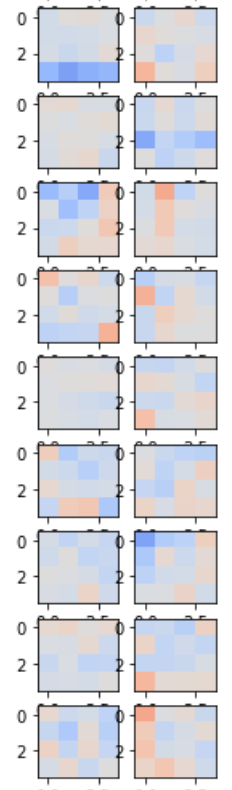
此文件包含remain\_CNN类，该类定义如何构建神经网络的实例。

它使用AlphaGoZero论文 - 即卷积层中的神经网络架构的浓缩版本，后面是许多剩余层，然后分成值和策略头。

卷积滤波器的深度和数量可以在配置文件中指定。

Keras库用于构建网络，后端是Tensorflow。

要查看神经网络中的单个卷积滤波器和密集连接层，请在run.ipynb笔记本中运行以下命令：



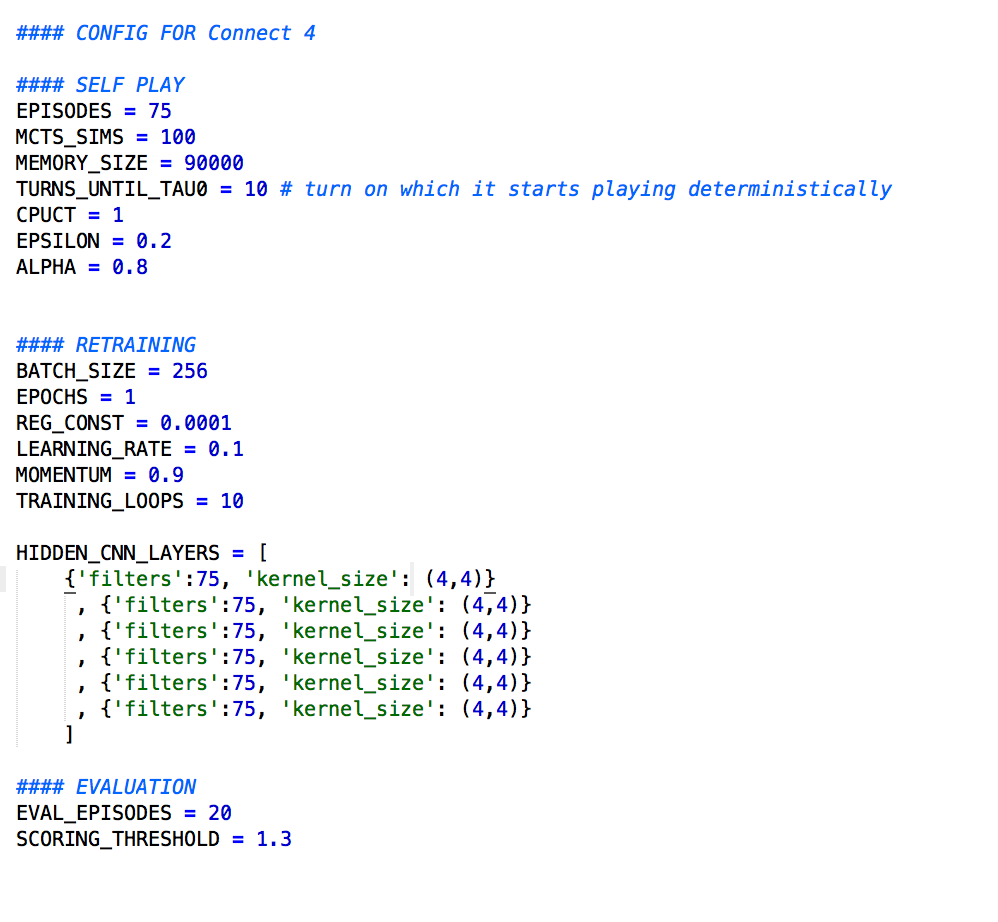
#### MCTS.py公司

它包含构成蒙特卡罗搜索树的节点、边和MCTS类。

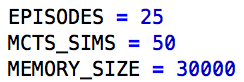
MCTS类包含前面提到的moveToLeaf和backillMethods，Edge类的实例存储关于每个潜在移动的统计信息。

#### 配置py

这是设置影响算法的关键参数的位置。



调整这些变量将影响算法的运行时间、神经网络的精度和整体的成功率。以上参数产生高质量的Connect4播放器，但需要很长时间才能完成。若要加快算法速度，请改为尝试以下参数。



#### 函数py

包含playMatchesandplayMatchesBetweenVersions函数，用于在两个代理之间播放匹配。

要播放您的创建，请运行以下代码（它也在run.ipynb笔记本中）

#### 初始化.py

运行算法时，所有模型和内存文件都保存在根目录的runfolder中。

若要稍后从此检查点重新启动算法，请将运行文件夹传输到运行存档文件夹，并将运行编号附加到文件夹名称。然后，在initialise.py文件中输入运行号、型号版本号和内存版本号，与运行存档文件夹中相关文件的位置相对应。然后像往常一样运行算法将从这个检查点开始。

#### 内存.py

内存类的一个实例存储以前游戏的内存，该算法用于重新训练当前玩家的神经网络。

#### 损失.py

此文件包含一个自定义损失函数，在传递给交叉熵损失函数之前，该函数屏蔽非法移动的预测。

#### 设置.py

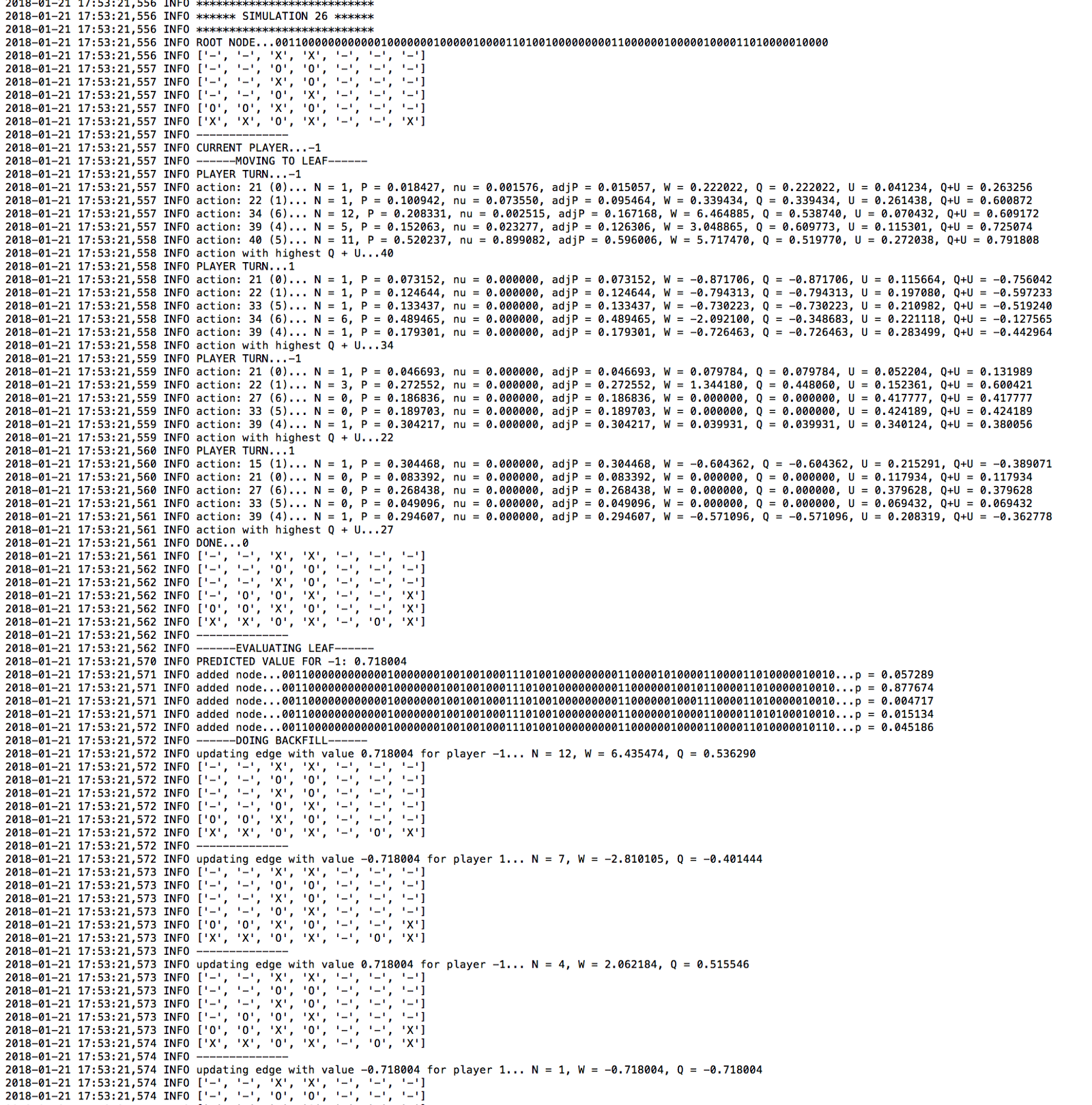
运行和运行存档文件夹的位置。

#### 伐木工人.py

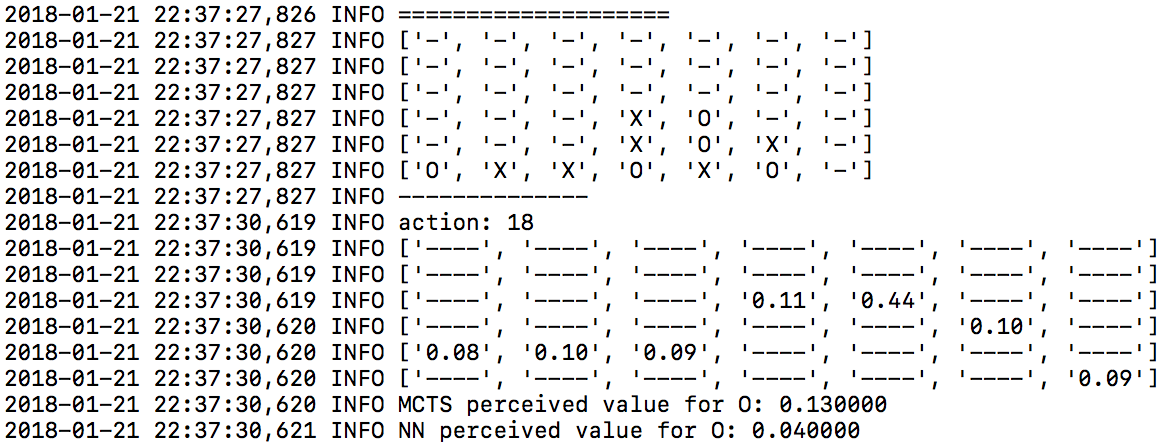
日志文件保存到运行文件夹中的日志文件夹中。

若要启用日志记录，请在此文件中将logger\_disabled变量的值设置为False。

查看日志文件将帮助您了解算法的工作原理并了解其“思想”。例如，下面是logger.mcts文件中的一个示例。

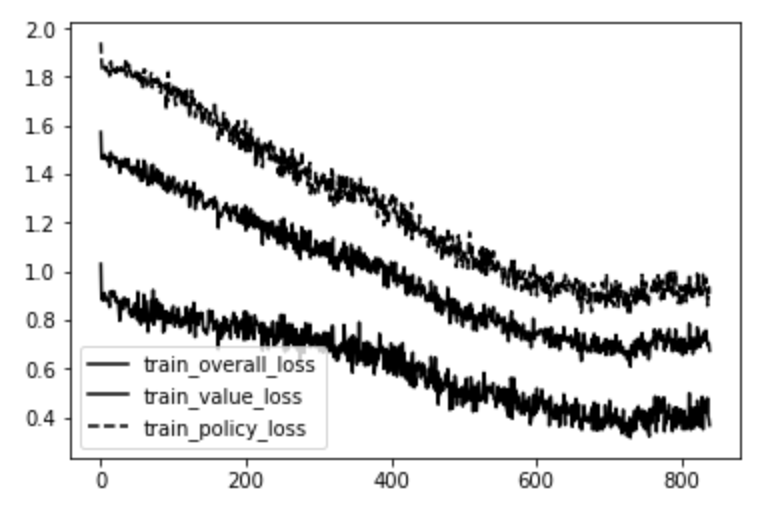


同样从logger.tourney文件中，您可以看到在评估阶段附加到每个移动的概率：



### 结果

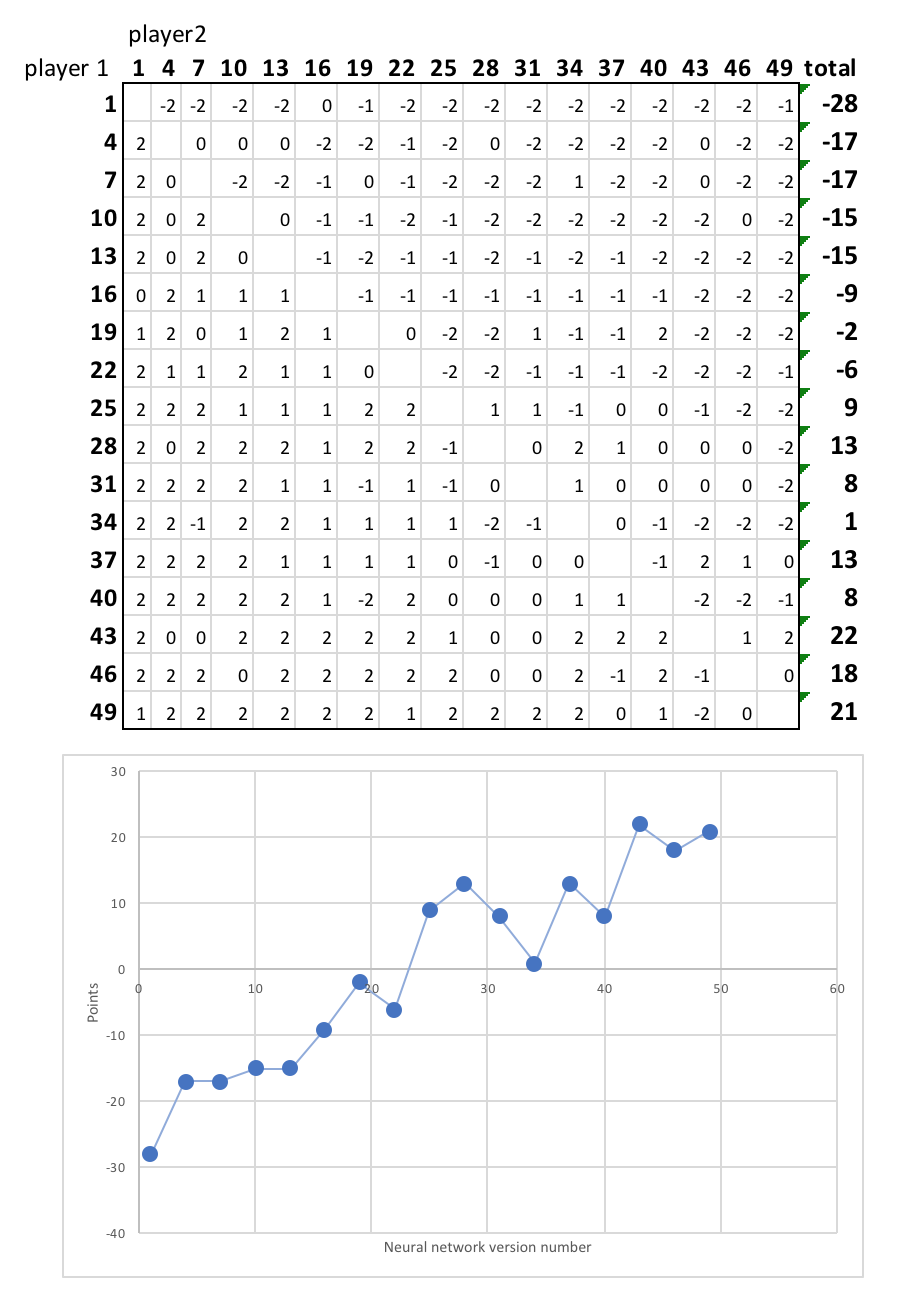
经过几天的培训，根据小批量迭代次数得出以下损失图表：



顶行是策略头中的错误（MCT的交叉熵移动概率，与神经网络的输出相对）。底线是价值头的误差（实际博弈值和神经网络预测值之间的均方误差）。中间线是两者的平均值。

显然，神经网络在预测每一个游戏状态的价值和可能的下一步行动方面正在变得更好。为了说明这是如何导致越来越强大的比赛，我在17名球员之间运行了一个联盟，从第一次迭代的神经网络，到第49次。每对选手都玩了两次，两人都有机会先玩。

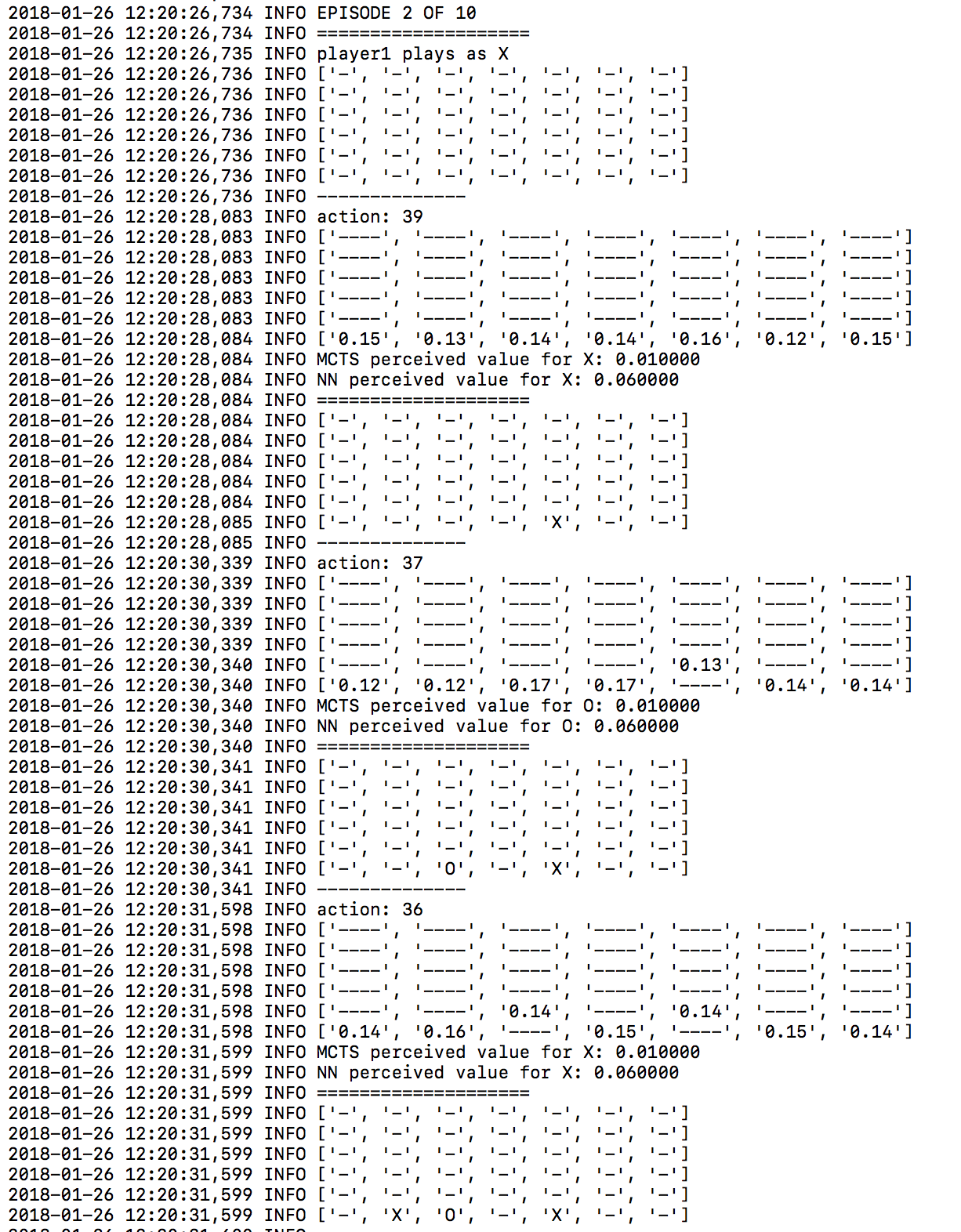
以下是最终的排名：



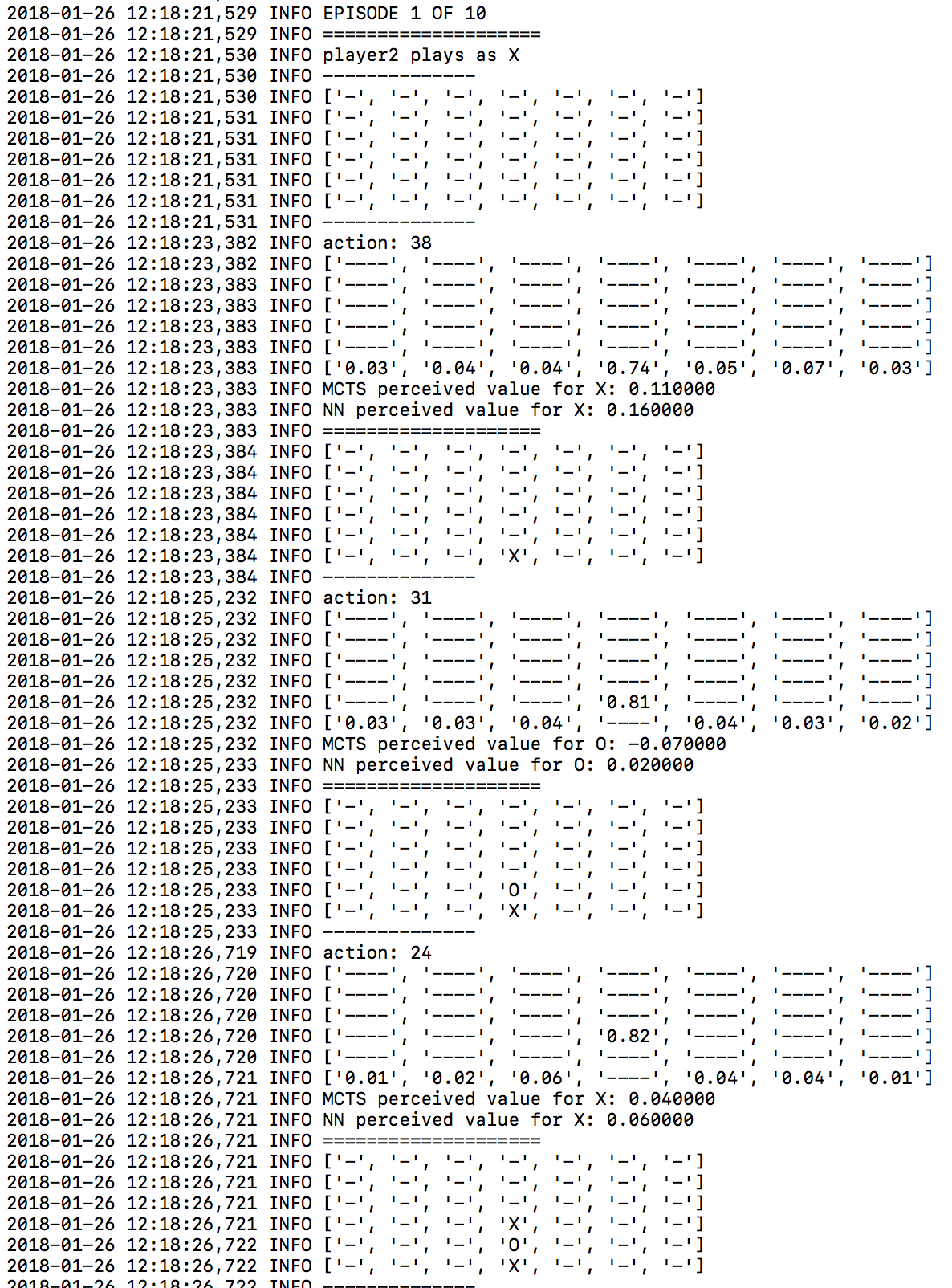
显然，神经网络的后一个版本比前一个版本优越，赢得了他们大部分的游戏。同时，随着训练时间的延长，运动员的学习还没有达到饱和状态，他们将继续变得更强，学习越来越多复杂的策略。

作为一个例子，随着时间的推移，神经网络所偏爱的一个明确的策略是尽早抓住中心柱。观察算法的第一个版本和第30个版本之间的差异：

第一个神经网络版本



第30版神经网络



这是一个很好的策略，因为许多线需要中柱 - 声称这是早期确保你的对手不能利用这一点。这是由神经网络学习到的，不需要任何人工输入。

### 学习不同的游戏

在games文件夹中有一个名为“Metasquares”的game.py文件。这包括在网格中放置X和O标记，以尝试形成不同大小的正方形。大方格比小方格得分多，格线满时得分最多的玩家获胜。

如果将Connect4 game.py文件切换为Metasquares game.py文件，相同的算法将学习如何播放Metasquares。

### 摘要

希望你觉得这篇文章有用 - 如果你在代码库或文章中发现任何错别字或有任何问题，请在下面的评论中告诉我，我会尽快回复你。



If you would like to learn more about how our company, Applied Data Science develops innovative data science solutions for businesses, feel free to get in touch through our website or directly through LinkedIn.

… and if you like this, feel free to leave a few hearty claps :)

Applied Data Science is a London based consultancy that implements end-to-end data science solutions for businesses, delivering measurable value. If you’re looking to do more with your data, let’s talk.