AlphaGo为什么那么厉害？

52cs 09/18/2016 0

作者：田渊栋，facebook科学家

声明：本文转载自雷锋网，原文链接。

最近我仔细看了下AlphaGo在《自然》杂志上发表的文章，写一些分析给大家分享。

AlphaGo这个系统主要由几个部分组成：

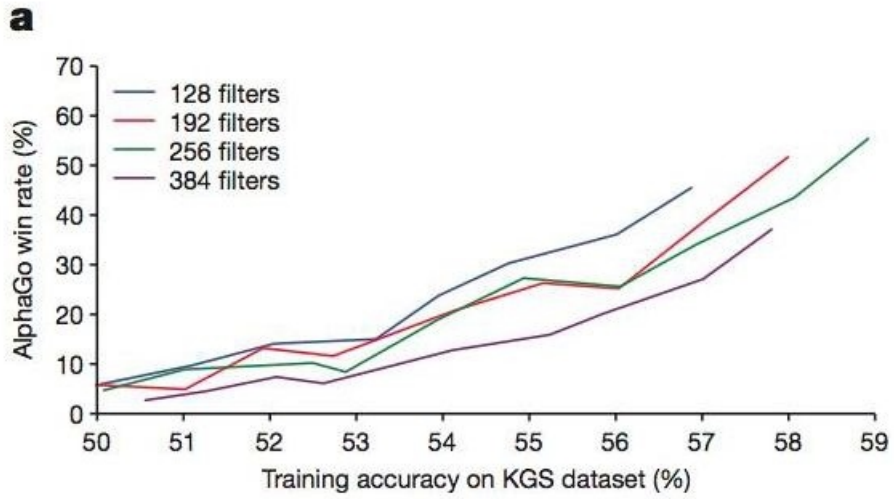
- 走棋网络（Policy Network），给定当前局面，预测/采样下一步的走棋。
- 快速走子（Fast rollout），目标和1一样，但在适当牺牲走棋质量的条件下，速度要比1快1000倍。
- 估值网络（Value Network），给定当前局面，估计是白胜还是黑胜。
- 蒙特卡罗树搜索（Monte Carlo Tree Search，MCTS），把以上这三个部分连起来，形成一个完整的系统。

我们的DarkForest和AlphaGo同样是用4搭建的系统。DarkForest较AlphaGo而言，在训练时加强了1，而少了2和3，然后以开源软件Pachi的缺省策略 (default policy)部分替代了2的功能。以下介绍下各部分。

1.走棋网络

走棋网络把当前局面作为输入，预测/采样下一步的走棋。它的预测不只给出最强的一手，而是对棋盘上所有可能的下一着给一个分数。棋盘上有361个点，它就给出361个数，好招的分数比坏招要高。DarkForest在这部分有创新，通过在训练时预测三步而非一步，提高了策略输出的质量，和他们在使用增强学习进行自我对局后得到的走棋网络（RL network）的效果相当。当然，他们并没有在最后的系统中使用增强学习后的网络，而是用了直接通过训练学习到的网络（SL network），

理由是RL network输出的走棋缺乏变化，对搜索不利。



有意思的是在AlphaGo为了速度上的考虑，只用了宽度为192的网络，而并没有使用最好的宽度为384的网络（见图2(a)），所以要是GPU更快一点（或者更多一点），AlphaGo肯定是会变得更强的。

所谓的0.1秒走一步，就是纯粹用这样的网络，下出有最高置信度的合法着法。这种做法一点也没有做搜索，但是大局观非常强，不会陷入局部战斗中，说它建模了“棋感”一点也没有错。我们把DarkForest的走棋网络直接放上KGS就有3d的水平，让所有人都惊叹了下。可以说，这一波围棋AI的突破，主要得益于走棋网络的突破。这个在以前是不可想像的，以前用的是基于规则，或者基于局部形状再加上简单线性分类器训练的走子生成法，需要慢慢调参数年，才有进步。

当然，只用走棋网络问题也很多，就我们在DarkForest上看到的来说，会不顾大小无谓争劫，会无谓脱先，不顾局部死活，对杀出错，等等。有点像高手不经认真思考的随手棋。因为走棋网络没有价值判断功能，只是凭“直觉”在下棋，只有在加了搜索之后，电脑才有价值判断的能力。

2. 快速走子

那有了走棋网络，为什么还要做快速走子呢？有两个原因，首先走棋网络的运行速度是比较慢的，AlphaGo说是3毫秒，我们这里也差不多，而快速走子能做到几微秒级别，差了1000倍。所以在走棋网络没有返回的时候让CPU不闲着先搜索起来是很重要的，等到网络返回更好的着法后，再更新对应的着法信息。

其次，快速走子可以用来评估盘面。由于天文数字般的可能局面数，围棋的搜索是毫无希望走到底的，搜索到一定程度就要对现有局面做个估分。在没有估值网络的时候，不像国象可以通过算棋子的分数来对盘面做比较精确的估值，围棋盘面的估计得要通过模拟走子来进行，从当前盘面一路走到底，不考虑岔路地算出胜负，然后把胜负值作为当前盘面价值的一个估计。这里有个需要权衡的地方：在同等时间下，模拟走子的质量高，单次估值精度高但走子速度慢；模拟走子速度快乃至使用随机走子，虽然单次估值精度低，但可以多模拟几次算平均值，效果未必不好。所以说，如果有一个质量高又速度快的走子策略，那对于棋力的提高是非常有帮助的。

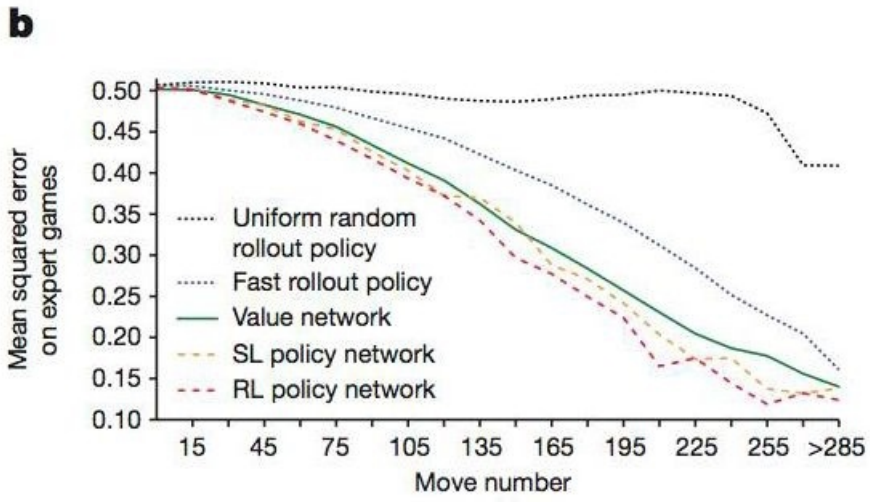
为了达到这个目标，神经网络的模型就显得太慢，还是要用传统的局部特征匹配（local pattern matching）加线性回归（logistic regression）的方法，这办法虽然不新但非常好使，几乎所有的广告推荐，竞价排名，新闻排序，都是用的它。与更为传统的基于规则的方案相比，它在吸纳了众多高手对局之后就具备了用梯度下降法自动调参的能力，所以性能提高起来会更快更省心。AlphaGo用这个办法达到了2微秒的走子速度和24.2%的走子准确率。24.2%的意思是说它的最好预测和围棋高手的下子有0.242的概率是重合的，相比之下，走棋网络在GPU上用2毫秒能达到57%的准确率。这里，我们就看到了走子速度和精度的权衡。

Extended Data Table 4 | Input features for rollout and tree policy

Feature	# of patterns	Description
Response	1	Whether move matches one or more response pattern features
Save atari	1	Move saves stone(s) from capture
Neighbour	8	Move is 8-connected to previous move
Nakade	8192	Move matches a *nakade* pattern at captured stone
Response pattern	32207	Move matches 12-point diamond pattern near previous move
Non-response pattern	69338	Move matches 3×3 pattern around move
Self-atari	1	Move allows stones to be captured
Last move distance	34	Manhattan distance to previous two moves
Non-response pattern	32207	Move matches 12-point diamond pattern centred around move

Features used by the rollout policy (first set) and tree policy (first and second set). Patterns are based on stone colour (black/white/empty) and liberties $(1, 2, \geq 3)$ at each intersection of the pattern.

和训练深度学习模型不同，快速走子用到了局部特征匹配，自然需要一些围棋的领域知识来选择局部特征。对此AlphaGo只提供了局部特征的数目（见Extended Table 4），而没有说明特征的具体细节。我最近也实验了他们的办法，达到了25.1%的准确率和4-5微秒的走子速度，然而全系统整合下来并没有复现他们的水平。我感觉上24.2%并不能完全概括他们快速走子的棋力，因为只要走错关键的一步，局面判断就完全错误了；而图2(b)更能体现他们快速走子对盘面形势估计的精确度，要能达到他们图2(b)这样的水准，比简单地匹配24.2%要做更多的工作，而他们并未在文章中强调这一点。



在AlphaGo有了快速走子之后，不需要走棋网络和估值网络，不借助任何深度学习和GPU的帮助，不使用增强学习，在单机上就已经达到了3d的水平（见Extended Table 7倒数第二行），这是相当厉害的了。任何使用传统方法在单机上达到这个水平的围棋程序，都需要花费数年的时间。在AlphaGo之前，Aja Huang曾经自己写过非常不错的围棋程序，在这方面相信是有很多的积累的。

3. 估值网络

Extended Data Table 7 | Results of a tournament between different variants of AlphaGo

Short name	Policy network	Value network	Rollouts	Mixing constant	Policy GPUs	Value GPUs	Elo rating
α_{rvp}	p_σ	v_θ	p_π	$\lambda = 0.5$	2	6	2890
α_{vp}	p_σ	v_θ	–	$\lambda = 0$	2	6	2177
α_{rp}	p_σ	–	p_π	$\lambda = 1$	8	0	2416
α_{rv}	$[p_\tau]$	v_θ	p_π	$\lambda = 0.5$	0	8	2077
α_v	$[p_\tau]$	v_θ	–	$\lambda = 0$	0	8	1655
α_r	$[p_\tau]$	–	p_π	$\lambda = 1$	0	0	1457
α_p	p_σ	–	–	–	0	0	1517

Evaluating positions using rollouts only (α_{rp}, α_r), value nets only (α_{vp}, α_v), or mixing both (α_{rvp}, α_{rv}); either using the policy network p_σ (α_{rvp}, α_{vp}, α_{rp}), or no policy network (α_{rvp}, α_{vp}, α_{rp}), that is, instead using the placeholder probabilities from the tree policy p_τ throughout. Each program used 5 s per move on a single machine with 48 CPUs and 8 GPUs. Elo ratings were computed by BayesElo.

AlphaGo的估值网络可以说是锦上添花的部分，从Fig 2(b)和Extended Table 7来看，没有它AlphaGo也不会变得太弱，至少还是会在7d-8d的水平。少了估值网络，等级分少了480分，但是少了走棋网络，等级分就会少掉800至1000分。特别有意思的是，如果只用估值网络来评估局面（2177），那其效果还不及只用快速走子（2416），只有将两个合起来才有更大的提高。我的猜测是，估值网络和快速走子对盘面估计是互补的，在棋局一开始时，大家下得比较和气，估值网络会比较重要；但在有复杂的死活或是对杀时，通过快速走子来估计盘面就变得更重要了。考虑到估值网络是整个系统中最难训练的

部分（需要三千万局自我对局），我猜测它是最晚做出来并且最有可能能进一步提高的。

关于估值网络训练数据的生成，值得注意的是文章中的附录小字部分。与走棋网络不同，每一盘棋只取一个样本来训练以避免过拟合，不然对同一对局而言输入稍有不同而输出都相同，对训练是非常不利的。这就是为什么需要三千万局，而非三千万个盘面的原因。对于每局自我对局，取样本是很有讲究的，先用SL network保证走棋的多样性，然后随机走子，取盘面，然后用更精确的RL network走到底以得到最正确的胜负估计。当然这样做的效果比用单一网络相比好多少，我不好说。

一个让我吃惊的地方是，他们完全没有做任何局部死活/对杀分析，纯粹是用暴力训练法训练出一个相当不错的估值网络。这在一定程度上说明深度卷积网络（DCNN）有自动将问题分解成子问题，并分别解决的能力。

另外，我猜测他们在取训练样本时，判定最终胜负用的是中国规则。所以说三月和李世石对局的时候也要求用中国规则，不然如果换成别的规则，就需要重新训练估值网络（虽然我估计结果差距不会太大）。至于为什么一开始就用的中国规则，我的猜测是编程非常方便（我在写DarkForest的时候也是这样觉得的）。

4. 蒙特卡罗树搜索

这部分基本用的是传统方法，没有太多可以评论的，他们用的是带先验的UCT，即先考虑DCNN认为比较好的着法，然后等到每个着法探索次数多了，选择更相信探索得来的胜率值。而DarkForest则直接选了DCNN推荐的前3或是前5的着法进行搜索。我初步试验下来效果差不多，当然他们的办法更灵活些，在允许使用大量搜索次数的情况下，他们的办法可以找到一些DCNN认为不好但却对局面至关重要的着法。

一个有趣的地方是在每次搜索到叶子节点时，没有立即展开叶子节点，而是等到访问次数到达一定数目(40)才展开，这样避免产生太多的分支，分散搜索的注意力，也能节省GPU的宝贵资源，同时在展开时，对叶节点的盘面估值会更准确些。除此之外，他们也用了一些技巧，以在搜索一开始时，避免多个线程同时搜索一路变化，这部分我们在DarkForest中也注意到了，并且做了改进。

5. 总结

总的来说，这整篇文章是一个系统性的工作，而不是一两个小点有了突破就能达到的胜利。在成功背后，是作者们，特别是两位第一作者David Silver和Aja Huang，在博士阶段及毕业以后五年以上的积累，非一朝一夕所能完成的。他们能做出AlphaGo并享有现在的荣誉，是实至名归的。

从以上分析也可以看出，与之前的围棋系统相比，AlphaGo较少依赖围棋的领域知识，但还远未达到通用系统的程度。职业棋手可以在看过了寥寥几局之后明白对手的风格并采取相应策略，一位资深游戏玩家也可以在玩一个新游戏几次后很快上手，但到目前为止，人工智能系统要达到人类水平，还是需要大量样本的训练的。可以说，没有千年来众多棋手在围棋上的积累，就没有围棋AI的今天。

在AlphaGo中，增强学习（Reinforcement Learning）所扮演的角色并没有想像中那么大。在理想情况下，我们希望人工智能系统能在对局中动态地适应环境和对手的招式并且找到办法反制之，但是在AlphaGo中增强学习更多地是用于提供更多质量更好的样本，给有监督学习（Supervised Learning）以训练出更好的模型。在这方面增强学习还有很长的路要走。

另外，据他们的文章所言，AlphaGo整个系统在单机上已具有了职业水平，若是谷歌愿意开几万台机器和李世石对决（这对它来说再容易不过了，改个参数就行），相信比赛会非常精彩。

下面是根据读者提问做的一些更新。

问题1：“Alphago的MCTS做rollout的时候，除了使用快速走子，还用了搜索树的已有部分，看起来像是AMAF/RAVE反过来：AMAF是把快速走子的信息传导到树的其它无关部分，Alphago是把树的其它无关部分拿来增强快速走子。我怀疑这是不是它棋力比其它DCNN+MCTS强的原因之一。"

这个办法在解死活题的文章中出现过，会在一定程度上提高搜索效率，但是提高多少还不知道。

问题2：“rollout的走法质量变好可能会导致棋力下降。”

这里要分两种情况，tree policy和default policy。在AlphaGo的文章里面已经说过了，tree policy的分布不能太尖，不然在搜索时太过重视一些看起来的好着，可能使得棋力下降。但是除了这种原因，一般来说tree policy变好棋力还是会变强的。

default policy这边，即（半）随机走子到最后然后判分，就很复杂了，质量变好未必对局面能估得更准。default policy需要保证的是每块棋的死活大体正确，不要把死的棋下成活的或者反之，而对大局观的要求反而没有那么高。双方完全可以配合着把每块棋下完，然后转战另一块，而不是说抢在对方前去别处占先手。

分类 人工智能

标签 alphago Google 围棋 阿尔法狗

上一篇文章
从特斯拉到计算机视觉之「图像语义分割」

下一篇文章
21个经典数据科学问题及答案（上）

或许您还喜欢这些文章

前辈之路(5) 刘鹏专访

神经网络简史

计算机首次通过图灵测试

初探计算机视觉的三个源头、兼谈人工智能 | 正本清源

迁移学习（ Transfer Learning ）

林达华：Computer Vision的尴尬

LEAVE A REPLY

Comment

Name *

Email *

Website

POST COMMENT

站内搜索

Search …

SEARCH

分类导航

- 业界新闻
- 云计算

- 人工智能
- 人物传记
- 前辈之路
- 国外留学
- 大数据
- 大牛点评
- 推荐系统
- 数学基础
- 数据库
- 数据科学
- 机器学习
- 深度学习
- 系统架构
- 经典论文
- 编程语言
- 脑科学
- 自然语言处理
- 计算视觉
- 默认分类

标签导航

Andrew ctr Google Hadoop mxnet NLP Spark SVM 云计算 人工智能 人机交互 分布式 判别式
前辈之路 图灵 图灵测试 图灵生平 大数据 实践经验 序列标注 推荐系统 数学基础 数据库 数据挖掘
数据科学 机器学习 深度学习 特征工程 生成式 神经网络 科学家 系统架构 编程语言 老师木 脑科学
自然语言 视觉研究 计算广告 计算机 计算机科学 计算机系统 计算机视觉 词向量 邓侃 静点评

重要通知

欢迎推荐好文章给 “我爱计算机”。本站长期招募志愿者参与小站的日常维护和建设。您可通过新浪微博 @52cs 联系我们。

不想错过本站信息，请邮件订阅小站：

请输入您的邮箱

点击订阅