【国君金工】决胜样本外：因子挖掘算法革新

原创 陈奥林、杨能 Allin君行 2020年11月03日 15:10

点击上方“**辰奥临君**”，关注我们

陈奥林 从业证书编号 S0880516100001
杨　能 从业证书编号 S0880519080008

摘要

摘要

遗传规划是一项模拟生物进化的方法，并非暴力搜索的工具，我们期望算法能够真正发挥出进化的优势，通过多代进化更高效地找寻统计套利类ALPHA因子，而非把算法作为一种随机生成因子的方式。

本篇报告提出了**低population(初始化种群) + 高generation（进化代数）**的算法设计理念。我们认为，遗传规划应用于因子挖掘时，低population高generation的组合相较于传统的高population低generation组合更加合理，算力消耗更小，因子样本外表现更佳。

低population高generation提升了**生成因子的质量**（局部范围内寻找最优解），降低了**生成因子间的相关性**（初始population重复概率大幅降低），同时也带来了新的挑战：膨胀。

膨胀体现为因子形式不必要的复杂臃肿,大大增加了算法过拟合的风险，是算法必须加以控制的问题。通过**Size-Fair(结点数合理交叉算法)**能够有效控制模型膨胀现象，同时算法运行时间大幅缩短。

在低population高generation的算法设计理念下，**多目标适应度函数意义凸显**，通过设置合理的适应度函数，可以在不增加算力负担的情况下，大幅缓解**因子非线性、过拟合和高相关**的问题。

适应度函数中包括日度多空收益的统计量能有效降低因子非线性问题，测试表现优于传统的IC/IR指标。

过拟合问题的产生与训练数据不足密不可分，在不增加历史样本的前提下，基于不同持仓日的日度收益模拟可增加统计显著性，减少过拟合风险。

在适应度函数中加入相关性约束能够保证算法在搜寻过程中会优先生成与现有因子负相关的新因子，生成因子间相关性可控制在20%以内，同时保证与风格因子的低相关性。

我们分别在上证50成分股和沪深300成分股中测试了算法因子挖掘的效果。**为了避免引入未来函数，我们设置算法生成因子的时间必须早于组合构建，仅根据样本内表现选择因子，重点考察样本外表现。**

基于上述思想构造的统计套利ALPHA生成器大大提升了遗传规划生成因子样本外的表现情况。2020年上证50 周调仓策略样本外IC 6%，t统计量2.52；沪深300内生成的10个因子样本外首月因子存活率70%，7个因子中1个因子在2个月后失效,2个因子在3个月后失效，其余因子样本外表现尚佳，沪深300内统计套利类因子生命周期平均在两个月以上。

01 1. 前言——遗传规划挖掘因子设计思考

当前量化投资的人工智能时代已成大势所趋,重复性高的ALPHA挖掘工作正在逐步被机器所取代，其中遗传规划是最为常见的ALPHA生成工具，其基础框架可参考图1：生成初代公式群（**Population**）后通过多轮优胜劣汰（**Generation**）的方式获得最终的ALPHA因子公式，但通过该算法挖掘ALPHA因子在实践中，特别是挖掘中低频因子时，依然面临以下几个挑战，其主要难点包括：

一、 生成因子的过拟合（样本外存活率过低）问题
二、 生成因子的高线性相关问题
三、 因子非线性问题
四、 因子生命周期不确定问题

为了解决上述问题，本篇报告提出了**低population(初始化种群）+ 高generation（进化代数）的算法设计理念。我们认为，遗传规划应用于因子挖掘时，低population高generation的组合相较于传统的高population低generation组合更加合理，算力消耗更小，因子样本外表现更佳。**因为遗传规划是一项模拟生物进化的方法，并非暴力搜索的工具，我们期望算法能够真正发挥出进化理念的优势，通过多代进化更高效地找寻统计套利类ALPHA因子，而非把算法仅仅作为一种随机生成因子的方式。

低population高generation的优势在于在不增加算力负担的前提下，提升了生成因子的质量（局部范围内寻找最优解），降低了生成因子间的相关性（初始population重复概率大幅降低）。一方面，因子质量的优劣是多元化的，除了传统指标IC、IR等，与现有因子的相关性，因子分层效果等多方面都是因子质量的体现，因而，在generation足够多的情况下，我们可以设置多元的适应度评价函数，使得因子在多次进化迭代的过程中，能够依次考虑因子评价的各个方面，提升因子质量，而在generation次数较低的算法里，多个适应度函数意义不大。另一方面，在实操中，我们需要生成大量符合标准的因子。在初始population较大的情况下多次运行算法，其结果很容易收敛到一些类似的因子上，这些因子形式略有不同，但相关性很高，对算力和时间造成了浪费，且无法通过剔除已检验因子的方式进行规避，而低population 有效避免了上述问题。

低population，高generation同时也带来了新的挑战：膨胀。膨胀指在适应度没有提升的情况下，公式长度快速增加。幸运的是，我们找到了一些方法能够解决膨胀问题——Size-Fair算法等。低population，高generation的另一个问题是导致一定概率无法找到符合标准的因子，但该问题并不严重，多次运行算法即可。

基于上述思想构造的ALPHA生成器大大提升了遗传规划生成因子样本外的表现情况。

本文结构如下：第一部分介绍了遗传规划中的膨胀现象及解决方案，第二部分介绍了如何进一步避免算法的过拟合问题；第三部分，我们给出了ALPHA生成器生成因子效果的相关检验。

02 遗传规划的膨胀现象及解决方案

2.1. 遗传规划的膨胀现象（bloat）

在早期的研究中，我们通常不会设置高generation的算法。原因有两点，一是过于消耗算力和时间，二是在实际应用中会发现，如果控制了树的深度，则后期的generation并没有带来适应度的提升；如果不控制深度，则最后生成的因子异常复杂，过拟合风险极高，即膨胀现象。

膨胀问题大大增加了算法过拟合的风险，是算法必须加以控制的问题。以下图试验为例，红色框中平均的适应度从-0.009上升到了0.114，蓝色框中平均的size从3.795上升到了15.055。其中可以注意到从第16代开始，在平均适应度没有巨大提升的情况下，平均的size还是在持续的增加，可以认为出现了膨胀现象。

2.2. 膨胀问题的解决方法

控制膨胀的方法包括：

1. Size and Depth Limits(结点数与深度限制)：在每次交叉操作之后，对于新产生的子代树验证是否同时满足结点数和深度的限制，若满足，则子代树进入下一代，反之，则将父代树中适应度较高的树作为子代数进入下一代。
2. Tarpeian technique：在计算每个公式树的适应度之前，先将大于平均size的树拿出来，并且将其中百分之P的树适应度直接设置为0，然后剩余所有的树按正常计算适应度。根据得到的适应度进行tournament selection。
3. Static parsimony pressure: 这个算法是对每个公式树的适应度添加一个与他们size正相关的惩罚项，其中我们需要对惩罚系数进行调整。G(x)=f(x)-c*l(x)，其中x代表公式树，G(x)表示惩罚之后的适应度，f(x)表示原始适应度，c是惩罚系数，l(x)表示公式数的size。然后根据惩罚之后的适应度进行selection操作。
4. Size-Fair(结点数合理)交叉算法的核心在于添加一个交叉操作子树结点数的限制条件，使得最后产生的子代树结点数在平均值上保持不变，从而使得模型膨胀现象得到控制。

上述方法均能一定程度上控制膨胀，我们更推荐Size-Fair法作为膨胀控制的方法。传统交叉算法(过程如图3所示)对所产生的子代树添加限制，但是并没有真正的解决膨胀出现的根源，即遗传算子本身并没有受到限制。Size-Fair交叉算法就以遗传算子为目标，对交叉操作中的子树添加限制，这样可以从根源上控制膨胀现象，而且这样的限制条件并不会对模型探索最优解公式产生任何负面影响。其次，Size-Fair交叉算法相对于传统交叉算法在寻找最优解公式上更加有效，因为在传统交叉算法中，当遇到子代树不满足限制条件时会选择直接复制其中一个父代树，这会减缓模型的探索速度。

2.3. Size-Fair的具体实现流程

1.1. Size-Fair的具体实现流程

size-fair(结点数合理)交叉算法的核心原理在于对交叉操作的子树添加一个限制。我们可以把Size-Fair交叉算法分解为五步：

第一步，获得用于交叉进化的父代树

首先，在随机生成的初始公式种群中进行两次独立的竞争式选择法(tournament selection)，以此来获得两棵父代树，然后在第一棵父代树上随机选一个结点作为交叉结点(其中第一棵父代树上结点的选择要遵从90%来自于内部节点，10%来自于叶节点和内部节点)。然后保留第二棵父代树暂不选择交叉结点。

第二步，计算被删子树的结点数

计算从第一棵父代树上被删子树的结点数(size)，记为L。其中结点数的计算公式为：L=内部结点个数+叶结点个数。L将指导后面几步中第二棵父代树交叉结点的随机选择。

第三步，计算第二棵父代树上所有子树的结点数

假设第二棵父代公式树的所有子树有n棵，我们将这n棵子树组成的集合称为交叉备选子树集，然后分别计算出这n棵子树的结点数并分别记为l1,l2,l3,l4,.....ln。如图4所示，该示例中的所有子树个数为6棵，并且子树结点数分别为1，1，1，1，3，5。

第四步，移除较大的备选子树

为了控制膨胀的出现，我们将交叉备选子树集中结点数大于1+2L的子树移除，也就是说，我们在这里添加了一个结点数限制，为了让交叉生成的子代树的结点数不能比父代树多1+L。

第五步，选取子树

对于移除之后的备选子树集，我们记录比L结点数小的备选子树个数为n-，比L结点数大的备选子树个数为n+，以及结点数与L相等的备选子树个数为n0，并且也计算较大备选子树与较小备选子树之间的平均子树结点数差值。之后第二棵父代树上交叉结点的选择将被分成三种情况：

1）若在备选子树中不存在结点数与L相等的子树，那么我们将从第一棵父代树中重新选择一个交叉点，并且重复之前的步骤；
2）如果不存在备选子树大于或者小于L，那么我们将在备选子树中选择结点数与L相等的子树插入到第一棵父代树的交叉点，然后作为子代树进入下一代。这样的情况主要是为了说明如果在第一棵父代树上选择了叶结点作为交叉点，那么也只能选择叶结点来作为备选子树

3）若以上三种尺寸的备选子树都存在，那么我们首先使用有偏的轮盘选择法(biased roulette wheel selection)来选择出进行交叉操作的备选子树结点数。然后在知道结点数之后，若其中相同结点数存在多个备选子树，那么它们之间使用均匀概率来进行选择。最后将选择出的备选子树进行交叉操作得到子代树。

2.4. Size-Fair控制膨胀效果检验

本小节测试了Size-Fair算法控制膨胀的效果以及相对默认算法的耗时情况。测试具体设置如下：

测试股票池：沪深300成分股
Population：50
Generation：15
树形最大深度：8

由下表试验结果对比可知，Size-Fair算法能够有效控制膨胀。仅经过15轮进化，原始树形平均深度已经达到6.24，接近上限8，而Size-Fair算法下因子平均深度仅有2.46，算法控制膨胀效果显著。

另一方面，算法耗时也从2318秒减少至510秒，速度提升4倍。

应用上述方法，我们在ALPHA较少的上证50域内进行因子挖掘，得到了大量上证50内有效的量价因子。本报告以下因子为例，进行效果展示：

GP001（生成于2020年7月）：
Inv(Rolling_Std_20(Rank(Rolling_Max_10(hp))))

由右上图可知，以IR为适应度函数构建的算法，尽管IC显著，多空收益也比较稳定，但分层不一定线性。事实上，我们挖掘的大量因子均存在BOTTOM（TOP）组不一定是收益最低（高）的分组的现象。

GP002（生成于2020年7月）：
Rolling_Sum_5(neg(Rolling_Std_10(Rolling_Std_60(Rank(vwap)))))

GP002为另一个典型的例子。这也启示我们以IC/IR为适应度挖掘因子存在比较大的改进空间。

03 1. 遗传规划适应度设计

由上一小节可知，以IC/IR为适应度的算法存在因子非线性的问题。与此同时，过拟合问题、生成因子容易高度线性相关的问题依然存在。

Size-Fair虽然可以解决膨胀问题，但是因子过拟合问题依然非常严重。不少因子在样本外检验中都会出现与样本内效果差别过大的现象。过拟合因子占比过大容易降低模型的应用价值。

常规想法是把历史样本区分为训练区和测试区，但从实际应用效果来看，此方法效果大幅增加了算法运行时间，而效果没有显著改善。这是因为如果训练区的因子大部分没有通过测试区，那么最后通过测试区"幸存"的因子大概率依然是过拟合的。

在低population高generation的算法设计理念下，多目标适应度函数作用凸显，通过设置合理的适应度函数，可以在不增加算力负担的情况下，大幅缓解因子非线性、过拟合和高相关的问题。

3.1. 关于过拟合问题的思考

过拟合问题的产生与训练数据不足密不可分，在不增加历史样本的前提下，基于不同持仓日的日度收益模拟可增加统计显著性。由于统计套利类因子更容易受到市场环境变化的影响，因而回溯过长的历史数据意义不大。如果生成周频换仓的因子，样本内只考虑过去2年的情况，则只有100组样本数据，如果考虑月频换仓，样本数据更少。如何在有限的历史数据内，提高样本显著性呢？我们认为考察因子日频多空收益情况，特别是最大回撤情况，有助于增加样本内表现提升难度，减少过拟合风险，提升样本外表现。

具体而言，我们设置二元适应度函数：

ti（i=0,1,2,3,4,5）表示因子调仓的不同起始日期。

最大化多空收益虽然无法完全解决因子线性问题，但是基本能保证生成因子第一组（第五组）收益是最高（最低）的。从我们有限的试验经验来看，当设置适应度为最大化多空收益时，所得因子的IC、IR表现通常不错，但反之，多空收益不一定显著。

最小化最大回撤能够尽可能减轻样本外失效时带来的亏损，同时增加样本内过拟合的难度。

以我们在沪深300域内进行因子挖掘的因子GP003为例，因子非线性问题得到了较大改善。

GP003(生成于2020年9月)
Neut(EWA_3(op),SignedSqrt(Rolling_Min_10(Rolling_Ratio_20(EWA_3(op)))))

3.2. 关于因子高度线性相关问题的思考

在operator(算子)和输入特征足够多的情况下，低population高generation已经能够较好避免生成因子间相关性过高的问题。但是，在输入特征较少时，算法可能依然会出现收敛至与现有因子高度相关的因子。

如果采用正交化的方法，对算力的要求过高，也不利于原有因子的更新换代。因此，我们建议采用在适应度函数中加入相关性的限制：

加入该相关性约束后，算法在搜寻过程中会优先生成与现有因子负相关的新因子，相关性问题大幅改善。

以2020年2月生成的10个因子为例，从因子间相关性测试结果来看，生成因子间相关性控制了20%以内。

相关性适应度也可以用于避免生成因子与需要风险中性的风格因子具有过高的相关性。

04 样本外测试绩效分析

4.1. 算法基础设置

为了节省时间，我们首先设置初始化种群在20，在无法取得因子后逐步宽到50，迭代次数同理。

初始化种群：20-50
迭代次数：30-40
树最大深度：8
惩罚系数：1.1
变量个数：11
算子个数：40

因子选择最低标准：样本内最大回撤5%以内，平均日收益率大于万2
回溯周期：过去400个交易日
测试范围：上证50/沪深300成分股

4.2. 算法样本外表现检验

为了避免引入未来函数，我们设置算法生成因子的时间早于组合构建。首先，我们在上证50成分股内进行了滚动测试，此处适应度函数仅为最大回撤与相关性，为了避免人为因素的影响，我们没有逐一检验因子样本内的表现情况（下一部分再进行逐一检验），采用算法生成的所有因子组合平均的方式来考察生成因子整体的质量。

我们在每月月初运行算法生成因子（4-6个因子），每周换仓时，采用最新生成因子均值生成多空组合，计算多空收益，由于时间关系，我们仅测试了2020年1月至8月的情况。

为了进一步考察生成因子样本外存活率和生命周期，我们假设站在2020年4月初，用沪深300成分股两年的数据根据我们的算法生成10个GP因子，考察每一个因子的表现情况。沪深300成分股运行单次耗时约在1000秒-8000秒的区间内。

以下两张图展示了10个因子多空收益样本内和样本外多空收益的表现情况。观察可知，No.2、No.8、No.9在样本外直接失效，首月因子存活率70%，No.10在两个月后失效，No.3，No.6在3个月后失效，其余因子样本外表现尚佳。

除此之外，我们发现因子收益存在边际递减的情况，前5个因子收益整体高于后5个因子。

05 总结与展望

本篇报告提出了一个因子挖掘新思路，希望能对量化从业者们带来一些启发。因子挖掘自动生成算法是一个浩大的工程，算法本身改进空间非常大，至少算子的增加、基本面数据的增加等都有可能给模型带来提升。基于上证50和沪深300的研究也表明，即使在大市值域内依然存在个股间的统计套利机会。

详细报告请查看20201028发布的国泰君安金融工程专题报告《决胜样本外：因子挖掘算法革新》

特别声明：

本订阅号发布内容仅代表作者个人看法，并不代表作者所属机构观点。涉及证券投资相关内容应以所属机构正规发布的研究报告内容为准。

市场有风险，投资需谨慎。在任何情况下，本订阅号中的信息或所表述的意见均不构成对任何人的投资建议。在决定投资前，如有需要，投资者务必向专业人士咨询并谨慎决策。

本订阅号内容均为原创，未经书面授权，任何媒体、机构和个人不得以任何形式转载、发表或引用。