Dissecting Characteristics Nonparametrically

代码

cvxpy_lasso.py: 使用cvxpy进行lasso算法的实现

cvxpy_glasso.py: 使用cvxpy进行group lasso算法的实现

其中，有一个s1_data_trans_pred.py主要是验证在没有filter的效率低下。

跑代码的顺序：s1_with_filter.py -> s2_eval_result.py 两个代码是我们结果展示的代码。

注：s1_with_filter.py 13行将路径改成数据的路径。

算法描述

传统的针对超额收益的建模用回归的逻辑来表示：

$$R_{it} = \alpha + \sum_{s=1}^{S} \beta_s C_{s,it-1} + \varepsilon_{it} \tag{1}$$

其中s表示了拥有的特征量，如果我们将回归的线性部分看成一个函数，那么整体表达可以写成如下形式

$$R_{it} = \sum_{s=1}^{S} \tilde{m}_{ts}\left(\tilde{C}_{s,it-1}\right) + \varepsilon_{it} \tag{2}$$

此时的特征$\tilde{C}_{s,it-1}$是原始特征$C_{s,it-1}$的排序变换。公式(2)与公式(1)的目的都是为了估计参数，一个是为了估计出$\tilde{m_{ts}}$函数，一个是为了估计出β系数。

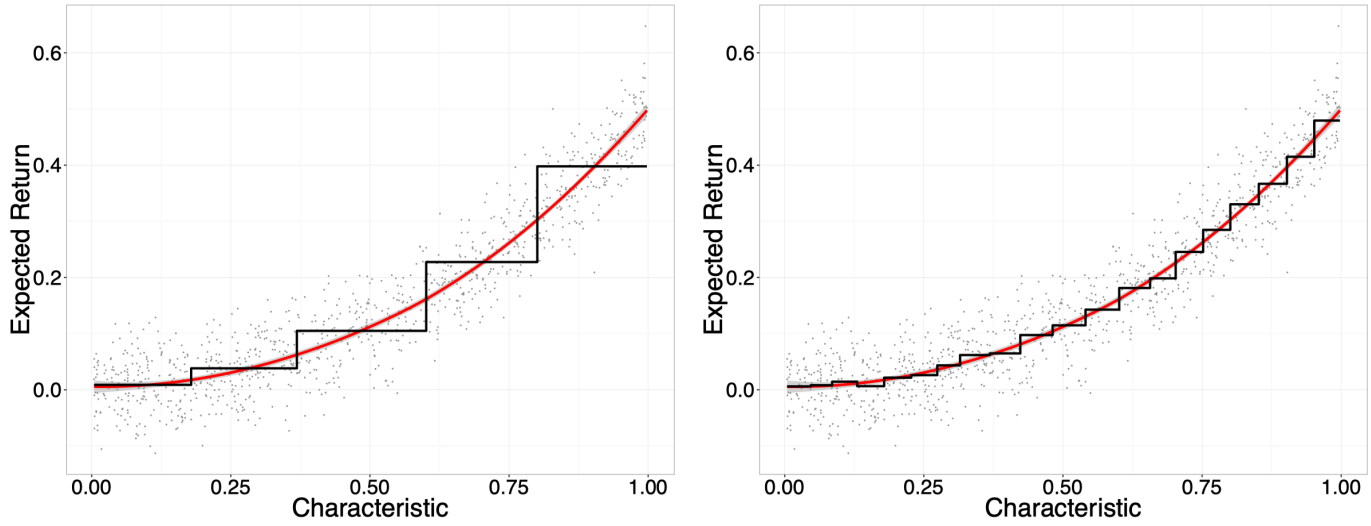
Lasso的机制：同时如果给定特征s不能帮助预测回报，则将给定特征函数$m_{ts}(\tilde{C}_{s,it-1})$设置为 0。

了解机制之后，重新梳理该文章的方法以及为什么要这么使用：

文章的目的是为了筛选出对预测有意义的变量，为了达成这个目的，最为直接简单的使用Lasso回归。但是，涉及到多种问题。

1. Lasso对异常值敏感，因此，文中会使用排序变换来对特征进行改造，从而使算法对异常值不那么敏感。
2. 用线性去估计并不是一个好的选择，因此，希望用更强的方法来对一个特征的收益进行估计。

文章想到了使用分段函数来对收益-特征函数将，即$\tilde{m}_{ts}\left(\tilde{C}_{s,it-1}\right)$函数进行估计。明显，这种估计是一种非参数估计的方法。假设，用分位数分段，用每一段的均值作为非参数的函数估计。



如果直接使用线性回归，那么对奇怪的分布很难拟合。可以看出来简单的二次分布都很难拟合。直接用分段的线性函数也会出现一个问题，就是分太多不合适，可能出现对异常值的敏感情况，分太少拟合不够。

所以，对这一部分可以尽量的改进，比如，用线性函数对每一段的数据进行拟合，文中使用了二次样条。

为什么用二次样条?

文中给出的解释是，可导可微的性质可以让样条函数在每一段的估计都连起来。

此时，问题开始变成，二次样条函数怎么来估计？二次样条函数是不是一个固定的函数，它与点的数量有关。更像是一种平滑。文中介绍了Chen（2007）的一篇文章，来说明二次样条的这个函数其实也可以有一个更有趣的表达。

基函数是一个二次函数，这点毫无疑问。然后，对每一个区间的特定值用一个二次函数去修正。如果有L个区间，则有（L-1）区间参数需要估计。加上基函数是二次函数，有三个参数-常数项、一次项、二次项需要估计。使用公式(3)对二次样条函数进行近似。

$$\tilde{m}(c) \approx \sum_{k=1}^{L+2} \gamma_k p_k(c) \tag{3}$$

其中γ_k是需要估计的参数，$p_k(c)$是已知的参数。

$$p_1(c) = 1, p_2(c) = c, p_3(c) = c^2, \text{and} \quad p_k(c) = \max\{c - t_{k-3}, 0\}^2$$

以上，我们介绍了如果对一个单独的特征进行参数的估计。每一个特定的特征都会有一个函数会被估计出来。对于参数的选择，文章侧重使用Lasso方法。与普通的Lasso对比，目标优化函数变成如下：

$$\tilde{\boldsymbol{\beta}}_t = \underset{b_{sk}:s=1,\ldots,S;k=1,\ldots,L+2}{\arg\min} \underbrace{\sum_{i=1}^{N} \left(R_{it} - \sum_{s=1}^{S} \sum_{k=1}^{L+2} b_{sk} p_k \left(\tilde{C}_{s,it-1} \right) \right)^2}_{\text{OLS Part}} + \underbrace{\lambda_1 \sum_{s=1}^{S} \left(\sum_{k=1}^{L+2} b_{sk}^2 \right)^{\frac{1}{2}}}_{\text{LASSO Penalty}} \tag{4}$$

其中，$\tilde{\boldsymbol{\beta}}_t$是一个$(L+2) \times S$的一个向量。此时，Group Lasso就与二次样条函数结合起来了。此时，我们需要探究的只剩文中提到的Adaptive是什么意思?

Adaptive体现在我们对不同部分的惩罚项应给予不同的权重。本文需要做两次Group Lasso。原因是第一次可能选出太多的特征了，Adaptive意在对选出来的多个特征再精简一次：

我们先做一次Group Lasso，得到参数的估计，进行如下操作：

$$w_{ts}=\begin{cases}\left(\sum_{k=1}^{L+2}\tilde{\beta}_{sk}^{2}\right)^{-\frac{1}{2}} & \text{if } \sum_{k=1}^{L+2}\tilde{\beta}_{sk}^{2}\neq 0\\ \infty & \text{if } \sum_{k=1}^{L+2}\tilde{\beta}_{sk}^{2}=0\end{cases} \tag{5}$$

再进行第二次的Group Lasso：

$$\breve{\boldsymbol{\beta}}_{t}=\underset{b_{sk}:s=1,\ldots,S;k=1,\ldots,L+2}{\arg\min}\sum_{i=1}^{N}\left(R_{it}-\sum_{s=1}^{S}\sum_{k=1}^{L+2}b_{sk}p_{k}\left(\tilde{C}_{s,it-1}\right)\right)^{2}+\lambda_{2}\sum_{s=1}^{S}\left(w_{ts}\sum_{k=1}^{L+2}b_{sk}^{2}\right)^{\frac{1}{2}} \tag{6}$$

注意公式(6)中，罚项的参数已经从之前的1变成了第一次计算得到的$w_{ts}=(\sum_{k=1}^{L+2}\tilde{\beta}_{sk}^{2})^{-\frac{1}{2}}$

至此，我们描述清楚了整个算法流程。

实验设置

实验过程出现的一些问题：直接选择244个特征和滚动12个月进行训练会发现一个问题，就是数据量太大优化时间过长的问题。通过实验，估计一次优化可能需要一个小时左右。因此，在进行算法的选择过程中，我们需要先进行一次删选。

逻辑如下：如果要判断一个特征是否是有效的，那么，只使用该特征进行建模，至少单独有效才能作为有效特征。判断是否有效即方程的F检验量是不是显著的。其次，拓展的二次样条相关的系数是不是有一些是有效的。于是，我们对公式(3)进行每个特征的单独建模，选择方程F检验量和任意变量的T检验量有效的特征进行建模，做第一步的特征删选。

我们选择滚动12个月做训练，来估计下一个月的收益率分位数。

我们对特征与收益率都要进行分位数处理，来满足文中所描述的排序问题的回归化等价化假设。

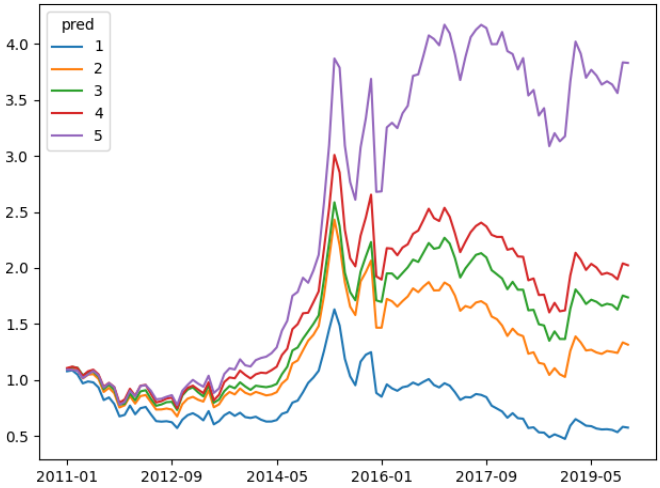
为了第一次选择更多的特征，我们第一次group lasso的惩罚参数选择较。由于Adaptive的存在，会进行进一步的筛选，所以第二次的罚项要做更小的处理。

所以，最终我们对原文的算法进行了一些改进进行了三步筛选的过程。

策略

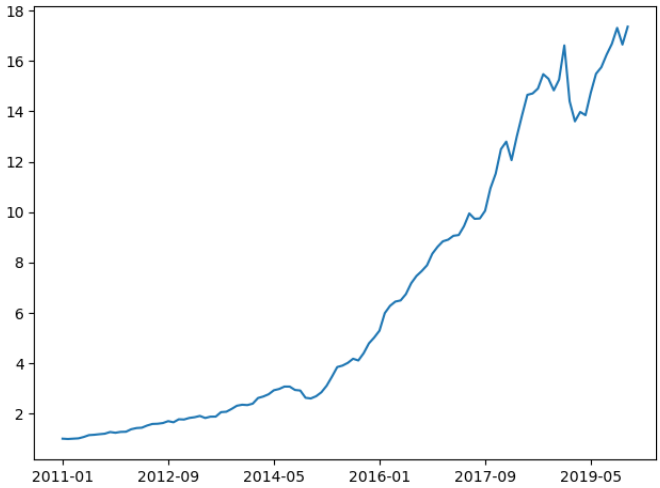
文章重要的点是在通过group lasso来进行特征的选择，但是，文中并没有给出策略，只是通过FF来验证选出来的因子是有效的。文章主要的逻辑是使用一个二次样条函数来模拟排序，就是使用一个更加复杂的函数来拟合收益，但是复杂的函数会造成过拟合问题，而group lasso可以用来抑制过拟合。所以，我们更加直观的策略还是把这个问题看成是一个预测问题，只是算法解决了一个非线性的鲁棒预测问题。

实验结果



我们对预测结果进行五组分层，可以看到分层明显。如果把预测得到的因子看成是非线性和因子，那个这个合成的因子是有效的。

我们分20组，并进行top-bottom的对冲，最后得到的对冲收益如下，效果显著：



结果的数据展示：我们把算法最终选择的特征放在结果的最后一列中。

	ticker	tradeDate	real_return	pred	chs_factor
1	600048	2017-09-29	0.04711538	0.5593056955638 12	ETOP
2	600077	2017-09-29	-0.05263158	0.4705600587623746	FixAssetRatio
3	600094	2017-09-29	-0.04066074	0.4551760396735217	OperatingProfitToTOR
4	600095	2017-09-29	-0.1407767	0.44617998672746767	TOBT
5	600383	2017-09-29	0.03138622	0.5878490458122634	GREV
6	600657	2017-09-29	-0.03833866	0.49844412278401656	TA2EV
7	600658	2017-09-29	-0.03532609	0.5083161819813468	KDJ_J
8	600724	2017-09-29	-0.05263158	0.49593754103808535	BBIC
9	600895	2017-09-29	0.03697078	0.45085093178535324	0
10	600208	2017-09-29	0.0023309999999999997	0.5599919252085716	0
11	600555	2017-09-29	-0.01036269	0.44112749674195273	0
12	600890	2017-09-29	0.0	0.45702801700537504	0
13	600240	2017-09-29	-0.07149853	0.5147062451888316	0
14	600246	2017-09-29	-0.07889126	0.4587460534681538	0
15	600322	2017-09-29	-0.05580694	0.4640014791070682	0
16	600325	2017-09-29	-0.03520209	0.5428262281539322	0