## **vn.py社区精选20 - 做不出好策略？试试这3个新的风险分析指标！**

原创 用Python的交易员 VNPY  2020-03-08

收录于话题

#vn.py社区精选

25个

**原文作者：KeKe**

**传统的回撤分析**

在对CTA策略的历史数据回测研究中，除了直接观察资金曲线的形状外，更多会基于各类围绕资金曲线的统计指标，来对策略的业绩表现进行评估，例如：

年化收益率

收益率标准差

夏普比率

历史最大回撤

最长回撤时间

等等

其中历史最大回撤和最长回撤时间两个统计指标，描述的是在最坏情况下（即连续亏损时）策略的损失程度。但在任何给定的样本数据上跑完回测，得到的回撤指标都分别只有一个点的估计值（最坏情况），同时其对策略参数变化的敏感性非常高。

因此在对策略的整体风险评估上，如果仅仅使用以上两个回撤指标，可能使得我们分关注极端情况。尤其是在参数优化时，导致我们选择一些仅仅**只是幸运**避开了个别几笔亏损交易，但整体预测效果未必最佳的参数组合。

好在，最近一篇新的论文提供了一个可靠的研究改进方向。

**一篇新的研究论文**

2019年德国科隆大学金融研究中心的Korn, Moller和Schwehm合作发布了一篇研究量化策略回撤分析的文章：

**《Drawdown Measures: Are They All the Same?》**

1） 在文章中，他们首先创建了6个不同的回撤指标：

Average Drawdown, 平均回撤（ADD）

Linearly Weighted Drawdown, 线性加权回撤（lwDD）

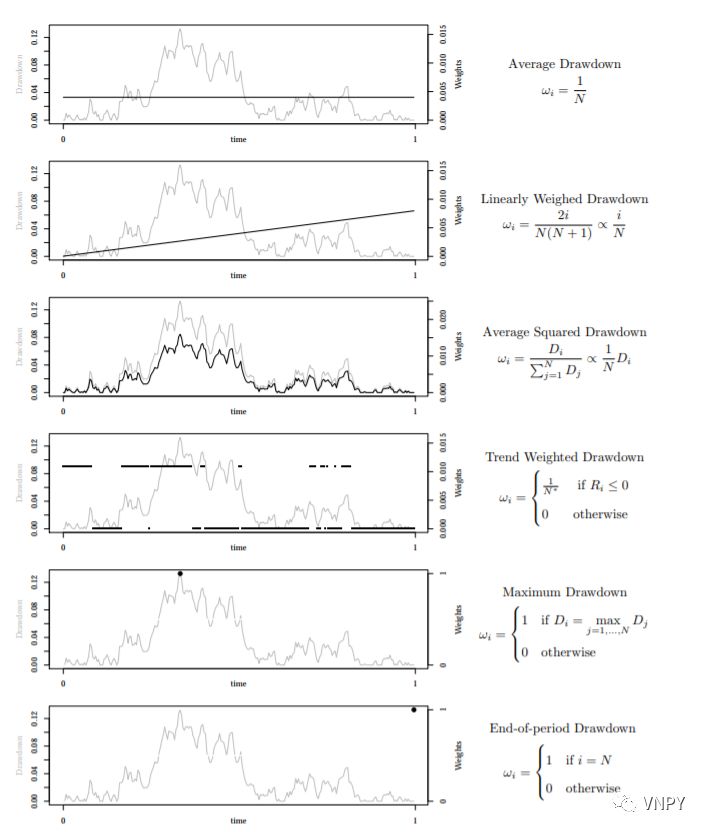
Average Squared Drawdown, 均方回撤（ADD^2)

Trend Weighted Drawdown, 趋势加权回撤（twDD)

Maximum Drawdown, 最大回撤（MDD）

End-of-period Drawdown, 期末回撤（eopDD）

指标的公式和图形如下：

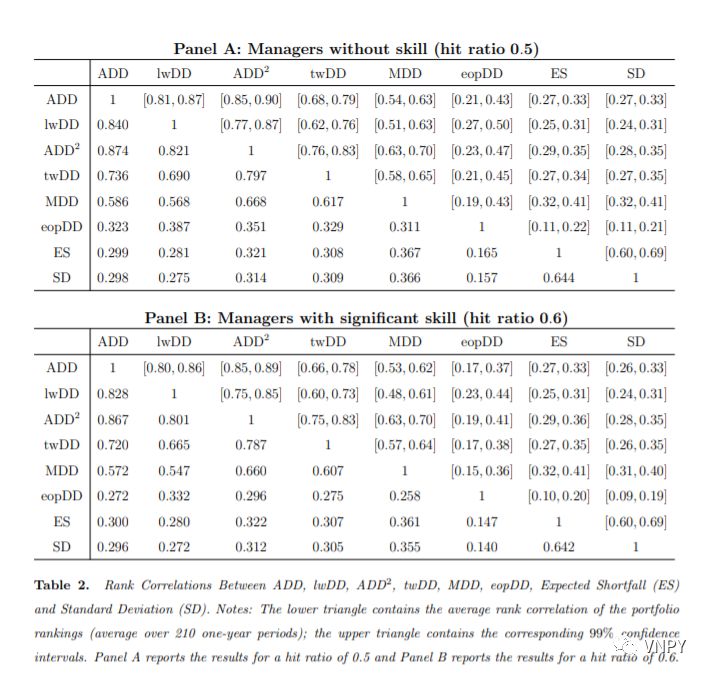


2）然后做了一个对比分析，用不同回撤指标来识别：

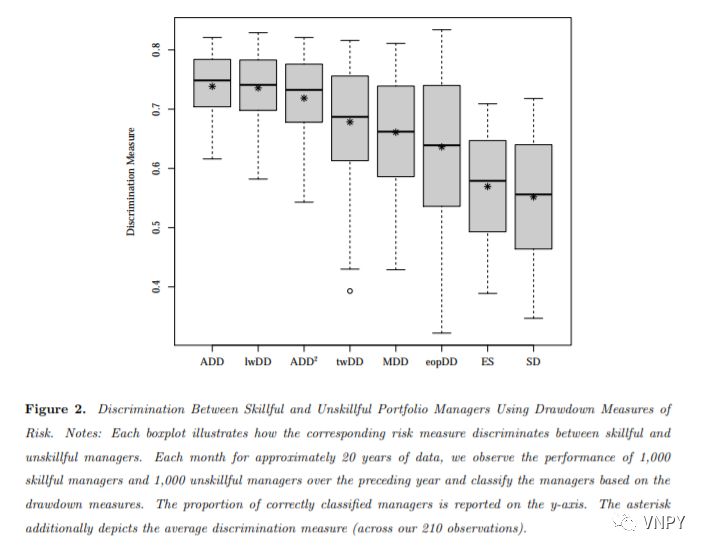
随机策略（hit ratio 0.5）

具有正数学期望的策略（hit ratio 0.6）

发现平均回撤（ADD）、线性加权回撤（lwDD）以及均方回撤（ADD^2）三者的效果最为接近：

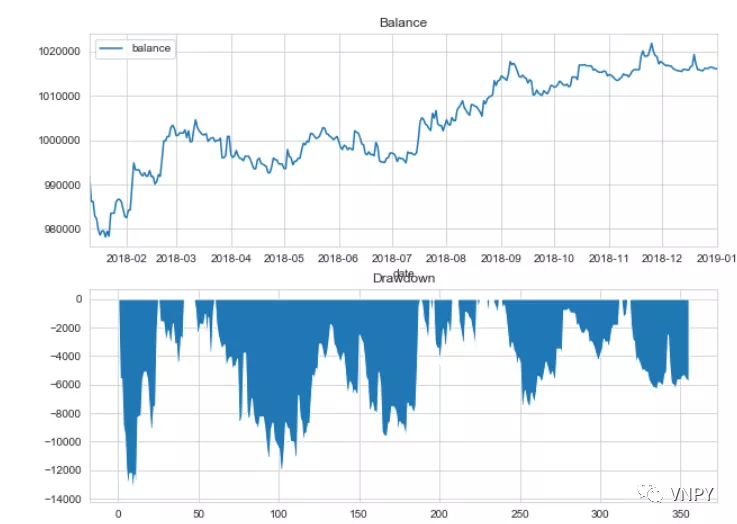


3）除了相似性以外，这3个指标的识别效果也更佳，即相比于最大回撤 MDD，这些回撤指标能够更有效的区分随机和正期望策略：

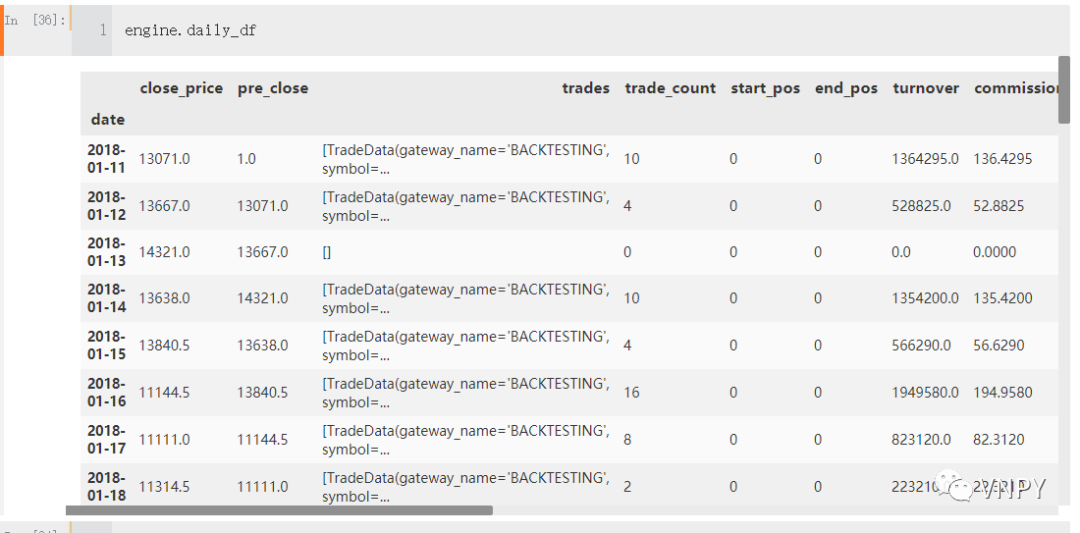


**基于vn.py的代码实现**

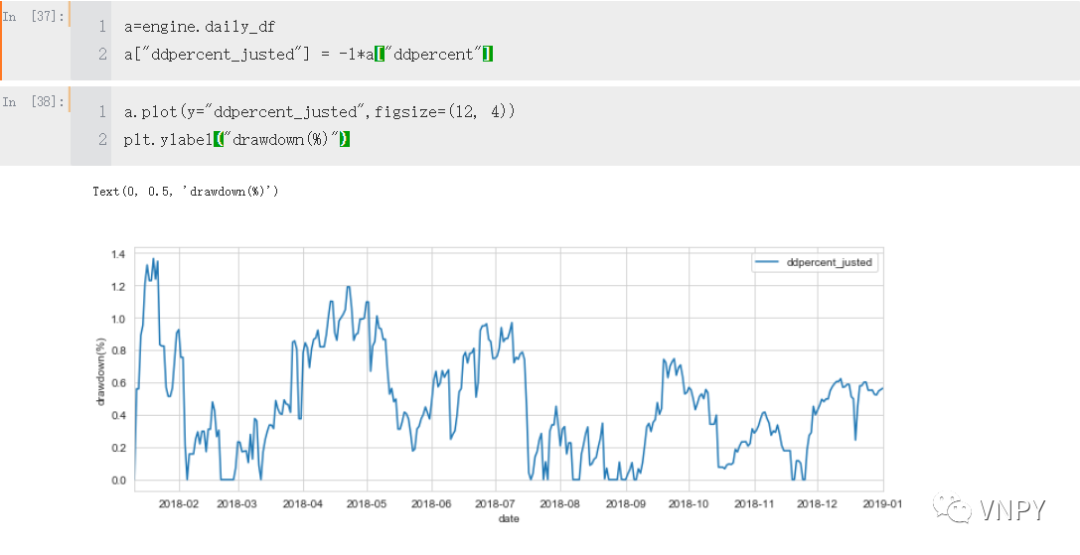
首先，我们需要在Jupyter Notebook中，以命令行CLI交互的模式调用CtaStrategy模块的回测功能（[不会的请戳这里](https://github.com/vnpy/vnpy/tree/master/examples/cta_backtesting" \t "/Users/huangning/Documents\\x/_blank)），对一个策略执行历史数据回测，得到资金曲线图和百分比回撤图：



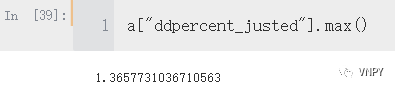
此时可以访问BacktestingEngine对象实例的daily\_df（DataFrame格式）数据，获取基于逐日盯市规则计算的每日策略统计数据（如当日盈亏、累计盈亏、成交笔数、百分比最大回撤等）：



由于百分比回撤是负数，为了后续研究的时候方便观察，先创建一个新的列【ddpercent\_justed】，这是调整为正数后的百分比最大回撤，然后画图显示出来：



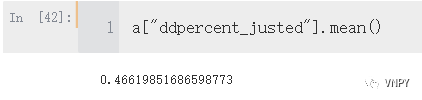
最大回撤可以通过对百分比回撤序列调用max函数方法，得到结果为1.4%：



有了参考对象，接下来我们开始计算以上论文中提到的3个更为有效回撤指标。

**平均回撤（ADD）**

平均回撤就是每日百分比回撤的算数平均，直接调用mean方法得到结果为0.41%，即采用简单的算术平均计算策略整体每日亏损的风险为0.41%：

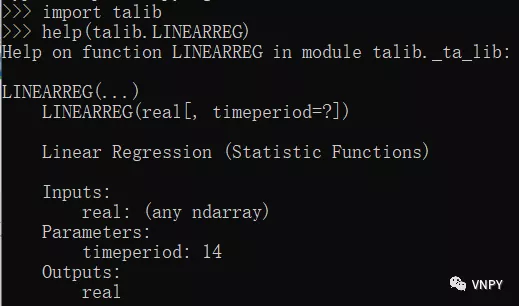


**线性加权回撤（lwDD）**

线性加权回撤就是使用最小二乘法（OLS）对回撤曲线进行线性回归，从而得到一条回撤的趋势线。最小化误差平方和的方法有利于避免极端情况的影响，让我们把关注点更多集中在策略的整体风险水平上。一个好的策略，其回撤的回归直线的斜率应该尽可能的小。

由于这里要求的只是回归线的终值，而不需要过程中的其他信息，所以无需动用那些巨型的数据分析库，如sklearn、statsmodels、scipy等等，这里我们选择更加轻量级的talib库（没错，算技术指标的那个）。

在交互式模式下执行命令：help(talib.LINEARREG)，我们可以知道线性回归函数的入参有2个：

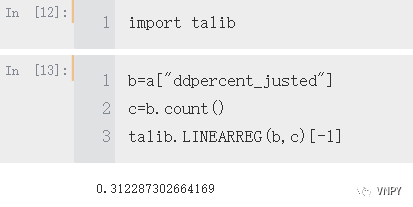


其中：

real：需要被回归的数据，即每天的百分比回撤数据；

tiemperiod：回望周期，我们需要最整体样本进行回归，故回望周期等于数据的样本数（在逐日盯市统计下，为策略回测的天数）。

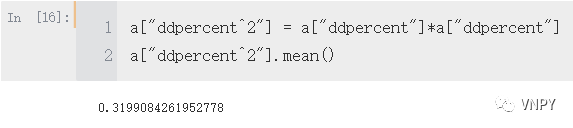
这样，我们得到线性加权回撤的结果为0.31%，即通过最小二乘法去噪声后，策略整体每日亏损的风险为0.31%：



**均方回撤（ADD^2)**

均方回撤就是每日百分比回撤的平方的期望值，采用这种计量方法的原因在于认为策略样本内回测属于小样本评估，属于有偏估算。

均方回撤的计算方法同样简单，只需要将百分比回撤这一列的数据平方后，再求平均数即可，最终得到均方回撤的结果为0.32%：



**回测引擎扩展**

最后我们可以将上述3个新的回撤统计指标，添加到回测引擎的统计指标计算函数中。打开位于C:\vnstudio\Lib\site-packages\vnpy\app\cta\_strategy目录中的backtesting.py文件，找到其中的calculate\_statistics函数，修改的内容分为三步：

初始化新的回撤指标数据；

计算3个新的回撤指标数值；

打印输出新的回撤指标；

新增的代码如下：

... average\_drawdown = 0 lw\_drawdown = 0 average\_square\_drawdown = 0... average\_drawdown = df["ddpercent"].mean() lw\_drawdown = talib.LINEARREG(df["ddpercent"], total\_days)[-1]  
 df["ddpercent^2"] = df["ddpercent"] \* df["ddpercent"] average\_square\_drawdown = - df["ddpercent^2"].mean()... self.output(f"百分比平均回撤: {average\_drawdown:,.2f}%") self.output(f"百分比线性加权回撤: {lw\_drawdown:,.2f}%") self.output(f"百分比均方回撤: {average\_square\_drawdown:,.2f}%")

完成修改后，再次回到Jupyter Notebook中执行回测，已经可以在打印输出的回测结果中找到三个新增的回撤指标：



同样在CLI命令行模式下，我们也可以很方便的这3个新的回撤指标（average\_drawdown、lw\_drawdown、average\_square\_drawdown），作为CTA策略参数优化的目标函数，筛选出稳健性更好的参数组合。