## **vn.py社区精选9 - 解密并强化日内经典策略R-Breaker**

原创 Frankie、KeKe [VNPY](https://mp.weixin.qq.com/javascript:void(0);) 2019-10-21

收录于话题

#vn.py社区精选

25个

R-Breaker是一种中高频的日内交易策略，这个策略也长期被Future Truth杂志评为最赚钱的策略之一。

R-Breaker策略结合了**趋势**和**反转**两种交易方式，所以交易机会相对较多，比较适合日内1分钟K线或者5分钟K线级别的数据。

**策略逻辑**

R-Breaker策略的核心逻辑由以下4部分构成：

**1）计算6个目标价位**

根据昨日的开高低收价位计算出今日的6个目标价位，按照价格高低依次是：

突破买入价（Bbreak）

观察卖出价（Ssetup）

反转卖出价（Senter）

反转买入价（Benter）

观察买入价（Bsetup）

突破卖出价（Sbreak）

具体计算方法如下：（其中a、b、c、d为策略参数）

观察卖出价（Ssetup）= High + a \* (Close – Low)

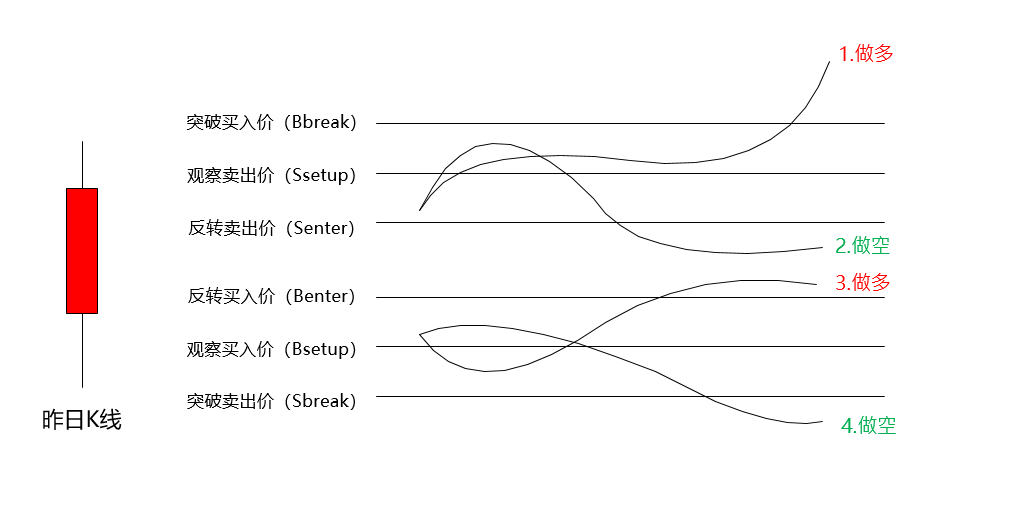
观察买入（Bsetup）= Low – a \* (High – Close)

反转卖出价（Senter）= b / 2 \* (High + Low) – c \* Low

反转买入价（Benter）= b / 2 \* (High + Low) – c \* High

突破卖出价（Sbreak）= Ssetup - d \* (Ssetup – Bsetup)

突破买入价（Bbreak）= Bsetup +  d \* (Ssetup – Bsetup)



**2）设计委托逻辑**

趋势策略交易：

若价格>突破买入价，开仓做多；

若价格<突破卖出价，开仓做空；

反转策略交易：

若日最高价>观察卖出价，然后下跌导致价格<反转卖出价，开仓做空或者反手（先平仓再反向开仓）做空；

若日最低价<观察买入价，然后上涨导致价格>反转买入价，开仓做多或者反手（先平仓再反向开仓）做多；

**3）设定相应的止盈止损**

**4）日内策略要求收盘前平仓**

以上是原版R-Breaker策略逻辑，但使用RQData从2010年至今（2019年10月）的1分钟沪深300股指期货主力连续合约（IF88）测试，效果并不理想。

**改进优化**

从逻辑上看R-Breaker策略可以拆分成趋势策略和反转策略，那么不妨试试将这两种逻辑分开，并逐个进行优化。

**1）趋势策略：**

若当前x分钟的最高价>观察卖出价，认为它具有上升趋势，在突破买入价挂上买入开仓的停止单；

若当前x分钟的最低价<观察买入价，认为它具有下跌趋势，在突破卖出价挂上买入开仓的停止单；

开仓后，使用固定百分比移动止损离场；

增加过滤条件：为防止横盘行情导致不断的开平仓，日内每次开仓买入开仓（卖出开仓）委托的价位都比上一次更高（更低）；

收盘前，必须平掉所持有的仓位。

**2）反转策略：**

若当前x分钟的最高价>观察卖出价，认为它已经到了当日阻力位，可能发生行情反转，在反转卖出价挂上卖出开仓的停止单；

若当前x分钟的最低价>观察买入价，认为它已经到了当日支撑位，可能发生行情反转，在反转买入价挂上买入开仓的停止单；

开仓后，使用固定百分比移动止损离场；

收盘前，必须平掉所持有的仓位。

其代码实现逻辑如下：

self.tend\_high, self.tend\_low = am.donchian(self.donchian\_window)  
if bar.datetime.time() < self.exit\_time:  
 if self.pos == 0: self.intra\_trade\_low = bar.low\_price self.intra\_trade\_high = bar.high\_price  
        if self.tend\_high > self.sell\_setup: long\_entry = max(self.buy\_break, self.day\_high) self.buy(long\_entry, self.fixed\_size, stop=True)  
 self.short(self.sell\_enter, self.multiplier \* self.fixed\_size, stop=True)  
        elif self.tend\_low < self.buy\_setup: short\_entry = min(self.sell\_break, self.day\_low) self.short(short\_entry, self.fixed\_size, stop=True)  
 self.buy(self.buy\_enter, self.multiplier \* self.fixed\_size, stop=True)  
 elif self.pos > 0: self.intra\_trade\_high = max(self.intra\_trade\_high, bar.high\_price) long\_stop = self.intra\_trade\_high \* (1 - self.trailing\_long / 100) self.sell(long\_stop, abs(self.pos), stop=True)  
 elif self.pos < 0: self.intra\_trade\_low = min(self.intra\_trade\_low, bar.low\_price) short\_stop = self.intra\_trade\_low \* (1 + self.trailing\_short / 100) self.cover(short\_stop, abs(self.pos), stop=True)  
*# Close existing position*else: if self.pos > 0: self.sell(bar.close\_price \* 0.99, abs(self.pos)) elif self.pos < 0: self.cover(bar.close\_price \* 1.01, abs(self.pos))

**回测效果**

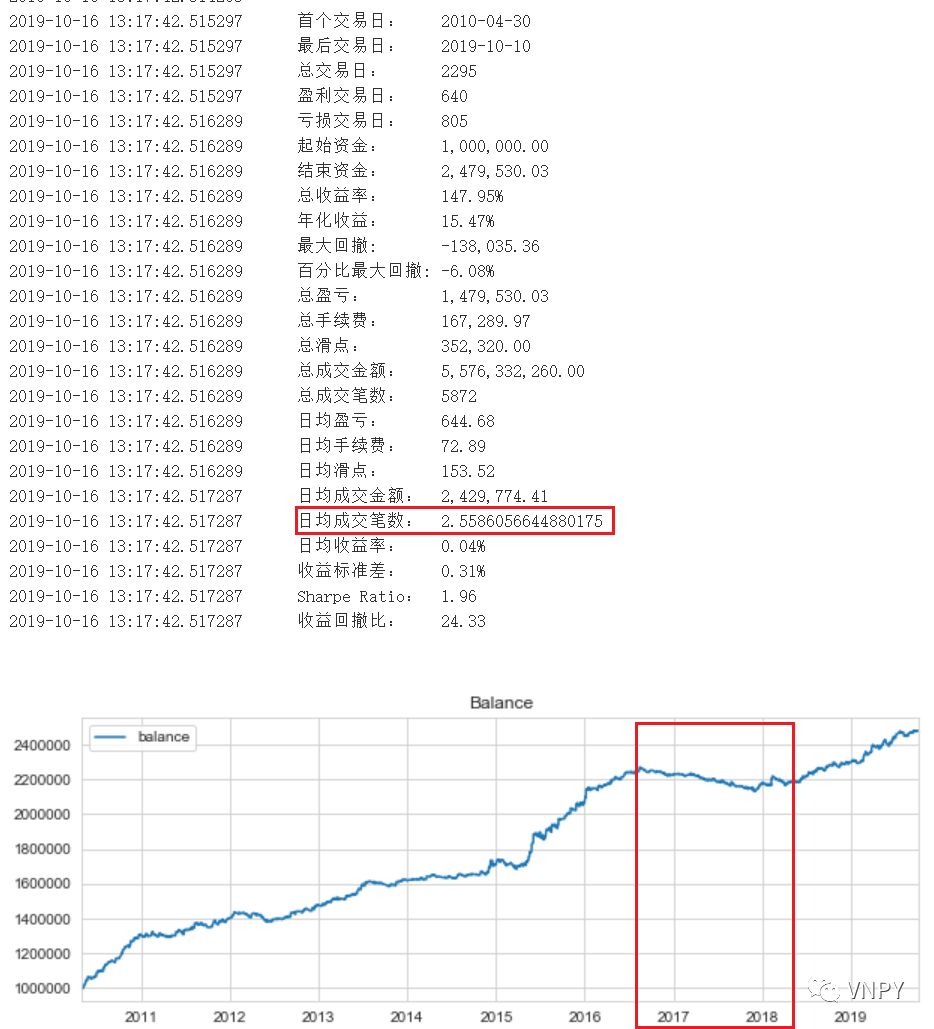
同样使用2010年至今的1分钟IF88数据进行回测。不过在展示强化版R-Breaker策略效果前，先分别展示一下拆分后的趋势策略和反转策略。

**1）趋势策略：**

趋势策略夏普比率1.96，日均成交2.6笔，资金曲线是整体上扬的；

但是在2017~2018年的盘整阶段，具有较大并且持续时间较长的回撤；

这凸显出趋势类策略自身无法规避的缺点：在趋势行情中盈利，意味着震荡行情必然亏损。



**2）反转策略**

反转策略夏普比率0.75，日均成交0.4笔，资金曲线缓慢上扬；

但是在2017~2018年的盘整阶段，资金曲线上扬最快，而且这个阶段是最平滑的；

这凸显出反转类策略优点：尽管在趋势行情亏损，在震荡行情必然能盈利。



综合对比2种策略的日均成交笔数和资金曲线，我们可以知道：

由于趋势策略日均交易笔数较多（2.08笔），它主要负责贡献R-Breaker策略的alpha；

趋势策略的亏损也是主要导致R-Breaker策略亏损的原因，但这时候的亏损可以由反转策略的盈利来填补。

由于趋势策略和反转策略是互斥的，在某些方面呈现出此消彼长的特点。那么，根据投资组合理论，可以把反转策略看作是看跌期权，买入一定规模的看跌期权来对消非系统性风险，那么组合的收益会更加稳健，即夏普比率更高。

由于趋势策略和反转策略日均成交手数比是2.6：0.4，若它们都只委托1手的话，反转策略的对冲效果微乎其微。

为了方便演示，我们设置趋势策略每次交易1手，反转策略则是3手，然后将两者合成为R-Breaker策略。

经过以上的仓位配置后，回测结果中的夏普比率提高到2，资金曲线整体上扬，而且没有较大且持续时间较长的回撤。



**最终结论**

R-Breaker策略的成功之处在于它并不是纯粹的趋势类策略，而是属于复合型策略，其alpha由两部分构成：趋势策略alpha和反转策略alpha。

这类复合型策略可以看作是轻量级的投资组合，因为它的交易标的只有一个：沪深300股指期货的主力合约。

更复杂的话，可以交易多个标的，如在商品期货做虚拟钢厂套利（同时交易螺纹钢、铁矿石、焦炭），在IF股指期货上做日内CTA策略。考虑到市场容量不同，价差套利能分配更多的资金，这样在价差套利提供稳定收益率基础上，CTA策略能在行情好的时候贡献更多alpha（高盈亏比特征导致的）。

从上面的例子可以看出，一个合理的投资组合，往往比单个策略具有更高的夏普比率。因为：夏普比率=超额收益/风险，所以夏普比率高意味着资金曲线非常平滑，也意味着我们可以有效的控制使用杠杆的风险。

当某个投资组合策略夏普足够高，而且策略资金容量允许，交易成本能有效控制等情况下，就可以通过杠杆来提升组合收益了。例如采用结构化产品的形式，经过银行等中介通过发行优先级份额，来为劣后级份额提供杠杆。这时劣后级投资者（有时100%是交易团队自身持有）同时也是债务人的角色，即在承担更大风险的同时，追求更高的最终收益，而优先级投资者则作为债权人享受利息收益。

##### ****策略源码****

最后，秉承vn.py社区的一贯精神：

**Talk is cheap, show me your pnl (or code) !**

自然必须附上策略的源代码：

from datetime import timefrom vnpy.app.cta\_strategy import ( CtaTemplate, StopOrder, TickData, BarData, TradeData, OrderData, BarGenerator, ArrayManager)  
  
class RBreakStrategy(CtaTemplate): """""" author = "KeKe"  
 setup\_coef = 0.25 break\_coef = 0.2 enter\_coef\_1 = 1.07 enter\_coef\_2 = 0.07 fixed\_size = 1 donchian\_window = 30  
 trailing\_long = 0.4 trailing\_short = 0.4 multiplier = 3  
 buy\_break = 0 *# 突破买入价* sell\_setup = 0 *# 观察卖出价* sell\_enter = 0 *# 反转卖出价* buy\_enter = 0 *# 反转买入价* buy\_setup = 0 *# 观察买入价* sell\_break = 0 *# 突破卖出价*  
 intra\_trade\_high = 0 intra\_trade\_low = 0  
 day\_high = 0 day\_open = 0 day\_close = 0 day\_low = 0 tend\_high = 0 tend\_low = 0  
 exit\_time = time(hour=14, minute=55)  
 parameters = ["setup\_coef", "break\_coef", "enter\_coef\_1", "enter\_coef\_2", "fixed\_size", "donchian\_window"] variables = ["buy\_break", "sell\_setup", "sell\_enter", "buy\_enter", "buy\_setup", "sell\_break"]  
 def \_\_init\_\_(self, cta\_engine, strategy\_name, vt\_symbol, setting): """"""        super().\_\_init\_\_(cta\_engine, strategy\_name, vt\_symbol, setting        )  
 self.bg = BarGenerator(self.on\_bar) self.am = ArrayManager() self.bars = []  
 def on\_init(self): """ Callback when strategy is inited. """ self.write\_log("策略初始化") self.load\_bar(10)  
 def on\_start(self): """ Callback when strategy is started. """ self.write\_log("策略启动")  
 def on\_stop(self): """ Callback when strategy is stopped. """ self.write\_log("策略停止")  
 def on\_tick(self, tick: TickData): """ Callback of new tick data update. """ self.bg.update\_tick(tick)  
 def on\_bar(self, bar: BarData): """ Callback of new bar data update. """ self.cancel\_all()  
 am = self.am am.update\_bar(bar) if not am.inited: return  
 self.bars.append(bar) if len(self.bars) <= 2: return else: self.bars.pop(0) last\_bar = self.bars[-2]  
 *# New Day* if last\_bar.datetime.date() != bar.datetime.date(): if self.day\_open:  
 self.buy\_setup = self.day\_low - self.setup\_coef \* (self.day\_high - self.day\_close) *# 观察买入价* self.sell\_setup = self.day\_high + self.setup\_coef \* (self.day\_close - self.day\_low) *# 观察卖出价*  
 self.buy\_enter = (self.enter\_coef\_1 / 2) \* (self.day\_high + self.day\_low) - self.enter\_coef\_2 \* self.day\_high *# 反转买入价* self.sell\_enter = (self.enter\_coef\_1 / 2) \* (self.day\_high + self.day\_low) - self.enter\_coef\_2 \* self.day\_low *# 反转卖出价*  
 self.buy\_break = self.buy\_setup + self.break\_coef \* (self.sell\_setup - self.buy\_setup) *# 突破买入价* self.sell\_break = self.sell\_setup - self.break\_coef \* (self.sell\_setup - self.buy\_setup) *# 突破卖出价*  
 self.day\_open = bar.open\_price self.day\_high = bar.high\_price self.day\_close = bar.close\_price self.day\_low = bar.low\_price  
 *# Today* else: self.day\_high = max(self.day\_high, bar.high\_price) self.day\_low = min(self.day\_low, bar.low\_price) self.day\_close = bar.close\_price             if not self.sell\_setup: return  
 self.tend\_high, self.tend\_low = am.donchian(self.donchian\_window)  
 if bar.datetime.time() < self.exit\_time:  
 if self.pos == 0: self.intra\_trade\_low = bar.low\_price self.intra\_trade\_high = bar.high\_price  
 if self.tend\_high > self.sell\_setup: long\_entry = max(self.buy\_break, self.day\_high) self.buy(long\_entry, self.fixed\_size, stop=True)  
 self.short(self.sell\_enter, self.multiplier \* self.fixed\_size, stop=True)  
                elif self.tend\_low < self.buy\_setup: short\_entry = min(self.sell\_break, self.day\_low) self.short(short\_entry, self.fixed\_size, stop=True)  
 self.buy(self.buy\_enter, self.multiplier \* self.fixed\_size, stop=True)  
 elif self.pos > 0: self.intra\_trade\_high = max(self.intra\_trade\_high, bar.high\_price) long\_stop = self.intra\_trade\_high \* (1 - self.trailing\_long / 100) self.sell(long\_stop, abs(self.pos), stop=True)  
 elif self.pos < 0: self.intra\_trade\_low = min(self.intra\_trade\_low, bar.low\_price) short\_stop = self.intra\_trade\_low \* (1 + self.trailing\_short / 100) self.cover(short\_stop, abs(self.pos), stop=True)  
 *# Close existing position* else: if self.pos > 0: self.sell(bar.close\_price \* 0.99, abs(self.pos)) elif self.pos < 0: self.cover(bar.close\_price \* 1.01, abs(self.pos))  
 self.put\_event()  
 def on\_order(self, order: OrderData): """ Callback of new order data update. """ pass  
 def on\_trade(self, trade: TradeData): """ Callback of new trade data update.        """ self.put\_event()  
 def on\_stop\_order(self, stop\_order: StopOrder): """ Callback of stop order update. """        pass