参考代码

来源：CSDN

网址：<http://t.csdn.cn/3AiQl>

具体代码：

【《深入浅出python量化交易交易实战》Python复现（第二章：(三)海龟策略+小结）】

分类专栏： 量化交易 文章标签： python 开发语言 数据分析

订阅专栏

深入浅出Python量化交易实学习（四）

第二章 回测与经典策略

2.3 海龟策略

2.3.1 使用海龟策略生成交易信号

2.3.2 根据交易信号和仓位进行下单

2.3.3 对海龟策略进行回测

2.4 小结

第二章 回测与经典策略

2.3 海龟策略

海龟策略的完整描述还是推荐柯蒂斯·费思的《海龟交易法则》，这是一个趋势交易法则，本章只是介绍了其入市和退出指标，其还有包括ATR，仓位控制，止损的一整套完整的交易逻辑值得去学习

海龟策略的其核心要点是：在股价超过过去N个交易日的股价最高点时买入，在股价低于过去N个交易日的股价最低点时卖出。上述的若干个最高点和最低点会组成一个通道，称为“唐奇安通道”

2.3.1 使用海龟策略生成交易信号

海龟策略的一个重点是，使用过去N天的股价最高点和过去N天的股价最低点生成唐奇安通道。一般来说，N会设置为20。不过因为我们下载的股票数据时间范围跨度比较小，所以选择了使用过去5日的股价最高点和最低点来进行演示

输入代码如下：

#codeing=utf-8

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import mplfinance as mpf

from utils import wz\_data

def gen\_turtle\_strategy(data):

# 创建一个turtle的数据表，使用原始数据的日期序列

turtle = pd.DataFrame(index=data.index)

print("----------------turtle------------------")

# 为什么要用shift(1)要向后移1?

# 因为要让第六天才能用前5天最高点和最低点

turtle['price'] = data['Close']

# 唐其安通道上界=过去5日内的最高价

turtle['high'] = data['Close'].shift(1).rolling(5).max()

# 唐其安通道下界=过去5日内的最低价

turtle['low'] = data['Close'].shift(1).rolling(5).min()

# 中轨道=0.5\*（通道上界+通道下界）

turtle['mid'] = (turtle['high'] + turtle['low']) / 2

# 当股价突破下沿时卖出，发出卖出信号，反之买入

turtle['buy'] = turtle['price'] > turtle['high']

turtle['sell'] = turtle['price'] < turtle['low']

return turtle

if \_\_name\_\_ == '\_\_main\_\_':

#单独实现了去wz网获取数据的接口

wz = wz\_data()

#股票代码，起始日期，结束日期，这里走的是前复权

data = wz.get\_stock\_data\_online('601318', '2020-01-01','2020-03-20')

#返回的直接是一个DataFrame对象

data.rename(columns={

'交易时间': 'Date',

'开盘价': 'Open',

'最高价': 'High',

'最低价': 'Low',

'收盘价': 'Close',

'成交量': 'Volume',

'成交额': 'Money',

'换手率': 'Turnover'

}, inplace = True)

data.index = pd.DatetimeIndex(data['Date'])

print(data.head())

#实现海龟策略

turtle\_strategy = gen\_turtle\_strategy(data)

print(turtle\_strategy.head(10))

输出的数据如下：

可以看到，high中存储的是唐奇安通道的上沿数据；low中存储的是唐奇安通道的下沿；buy如果为True，则为买入信号；sell如果为True，则为卖出信号；而当buy和sell都是False时，则不进行下单。唐奇安通道中，还有一条中线，中线的值mid是上沿和下沿的均值

同样，可以看到，2020-01-09第六天才有high、low和mid数据，这个就是shift(1)的作用，因为当天的收盘价不能既用于计算最大最小值，又用于比较什么能够下单，当天的价格只能和之前的价格算出的指标进行比较

2.3.2 根据交易信号和仓位进行下单

下面我们就根据生成的交易信号来下单。需要说明的是，当程序给出交易信号时，还要结合仓位来判断：当交易信号为“买入”且空仓时，我们才会下买入订单；而交易信号为“卖出”且有持仓股票时，我们才会下卖出订单

输入代码如下：

def gen\_turtle\_strategy(data):

# 创建一个turtle的数据表，使用原始数据的日期序列

turtle = pd.DataFrame(index=data.index)

print("----------------turtle------------------")

# 为什么要用shift(1)要向后移1?

# 因为要让第六天才能用前5天最高点和最低点

turtle['price'] = data['Close']

# 唐其安通道上界=过去5日内的最高价

turtle['high'] = data['Close'].shift(1).rolling(5).max()

# 唐其安通道下界=过去5日内的最低价

turtle['low'] = data['Close'].shift(1).rolling(5).min()

# 中轨道=0.5\*（通道上界+通道下界）

turtle['mid'] = (turtle['high'] + turtle['low']) / 2

# 当股价突破下沿时卖出，发出卖出信号，反之买入

turtle['buy'] = data['Close'] > turtle['high']

turtle['sell'] = data['Close'] < turtle['low']

# 订单初始状态为0

turtle['order'] = 0

# 前一天的仓位

turtle['preposition'] = 0

# 当前仓位

turtle['position'] = 0

# 初始仓位为0

position = 0

# 遍历turtle表

for k in range(len(turtle)):

# 当买入信号为true时，且仓位为0时，下单买入一手

if turtle.buy[k] and position == 0:

turtle.preposition.values[k] = position # 保存前一天仓位

turtle.order.values[k] = 1

position = 1

turtle.position.values[k] = position # 修改当前仓位

# 卖出信号且持有的时候卖出

elif turtle.sell[k] and position > 0:

turtle.preposition.values[k] = position # 保存前一天仓位

turtle.order.values[k] = -1

position = 0

turtle.position.values[k] = position # 修改当前仓位

# 不买不卖时持有仓位

else:

turtle.preposition.values[k] = position

turtle.position.values[k] = position

return turtle

输出的数据如下：

可以看到，在1月13日这一天，程序下了买入单；而在1月22日，程序下了卖出单

可视化代码如下：

def draw\_turtle\_strategy(strategy):

# 绘制

plt.figure(figsize=(10, 5))

# 绘制上沿，下沿，中线

plt.plot(strategy['price'], lw=2, label="股价")

plt.plot(strategy['high'], ls='--', c='r', lw=2, label="上沿")

plt.plot(strategy['low'], ls='-.', c='g', lw=2, label="下沿")

plt.plot(strategy['mid'], ls='dotted', c='b', lw=2, label="中线")

# 买入卖出信号标记

plt.scatter(

strategy.loc[strategy.order == 1].index,

strategy['price'][strategy.order == 1],

marker='^', s=80, color='r', label='买入信号'

)

plt.scatter(

strategy.loc[strategy.order == -1].index,

strategy['price'][strategy.order == -1],

marker='v', s=80, color='g', label='卖出信号'

)

#图注，网格，显示

plt.legend()

plt.grid()

plt.show()

draw\_turtle\_strategy(turtle\_strategy)

绘制的图形如下：

实线部分体现的是该股每日的价格，三条虚线分别对应唐奇安通道的上沿、下沿和中线。我们仔细观察会发现，当股价第一次突破唐奇安通道上沿时，程序进行了买入，但随后的几天中，股价再次突破了上沿，但由于此时已经有1手持仓，故没有再次买入

之后股价急转直下，突破了通道下沿，程序下单卖出。依次类推，在选定的时间范围内，程序进行了6笔交易

2.3.3 对海龟策略进行回测

回测输入代码如下：

def back\_track(strategy, init\_cash):

print('============Test Turtle===============')

# 新建一个positions表，序号和strategy数据表保持一致

positions = pd.DataFrame(index=strategy.index).fillna(0.0)

# 因为A股都是最低100股

# 因此设置stock字段为交易手的100倍

positions['stock'] = 100 \* strategy['order'].cumsum()

#资产表

portfolio = pd.DataFrame(index=positions.index)

#持仓价值

portfolio['stock\_value'] = positions['stock'].multiply(strategy['price'], axis=0)

# 仓位变化

pos\_diff = positions.diff()

print('==============交易量===============')

portfolio['stock'] = positions['stock']

# 初始资金减去下单金额的总和就是剩余的资金

portfolio['cash'] = init\_cash - pos\_diff.multiply(strategy['price'], axis=0).cumsum()

portfolio['total'] = portfolio['cash'] + portfolio['stock\_value']

print('==============持有资产===============')

print(portfolio.tail(13))

return portfolio

portfolio = back\_track(turtle\_strategy, 20000)

输出的数据如下：

可以看到，小瓦最后的总资产约为19466元，相比初始资金20000元，减少了544元；而使用双移动平均策略进行交易，小瓦的总资产缩水694元。在本例中，海龟策略稍胜一筹（少赔的就是赚的）

可视化代码如下：

def draw\_turtle\_portfolio(portfolio):

plt.figure(figsize=(10, 5))

plt.plot(portfolio['stock\_value'], lw=2, ls='--', label='stock\_value')

plt.plot(portfolio['total'], lw=2, ls='-', label='total')

plt.legend()

plt.grid()

plt.show()

draw\_turtle\_portfolio(portfolio)

绘制的图形如下：

从图中可以清晰地看到使用海龟策略进行交易后总资产和持仓市值的变化情况：实线部分体现的是总资产变化，虚线部分代表持仓股票的市值。与使用双移动平均策略相似，在整个股价变动明显的情况下，总资产略有减少

PS：本章使用的股票数据恰好在选定的时间范围内，呈现出下跌的趋势。在这种趋势下，尽量降低资产损失也是很有必要的。可以测试一下在其它时间范围起始点买入该股票，并在最后时间点卖出，对比总资产的损失情况

2.4 小结

为了评估“低买高卖”策略，我们简单设计了一个回测程序，通过总资产的变化来评价策略的业绩表现；随后又介绍了两个经典的交易策略——双移动平均策略和海龟策略。当然，这两种策略都是基于市场动量的变化而设计的，核心思想如下：如果股价上涨并超过某个点位，说明其上升的动量变强，这时应该买入；反之，则下行的动量变强，此时应该卖出。类似的策略也可以看作基于直觉的交易策略

————————————————

版权声明：本文为CSDN博主「xulu010」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/xulu010/article/details/129720932