知乎 首发于 Python 实战教程

Backtrader 教程 — 量化投资原来这么简单(1)

Python实...

都说Python可以用于量化投资，但是很多人都不知道该怎么做，甚至觉得是非常高深的知识，其实并非如此，任何人都可以在只有一点Python的基础上回测一个简单的策略。

Backtrader是一个基于Python的自动化回溯测试框架，作者是德国人 Daniel Rodriguez，是一个易懂、易上手的量化投资框架。今天我们就来试试用Backtrader进行简单的量化策略回溯。

当然，第一篇文章将会使用最简单的投资策略给大家起个头。通过学习这一篇文章，你将能学会以下这个简单的量化策略：

买入：五日价格移动平均线(MA5)和十日价格移动平均线(MA10)形成均线金叉（MA5上穿MA10）原理：最近处于涨势

卖出： 五日价格移动平均线(MA5)和十日价格移动平均线(MA10)形成均线死叉（MA10下穿MA5）原理：最近处于跌势

这个策略真的有用吗？普通人可能要炒一辈子股才能发现它的实际作用，而使用Python进行量化验证，则能迅速得到答案。

本系列教程的Github仓库：github.com/Ckend/python...

知乎 首发于 Python 实战教程

Windows环境下打开Cmd(开始—运行—CMD)，苹果系统环境下请打开Terminal(command+空格输入Terminal)，准备开始输入命令安装依赖。

当然，我更推荐大家用VSCode编辑器，把本文代码Copy下来，在编辑器下方的终端装依赖模块，多舒服的一件事啊：Python 编程的最好搭档—VSCode 详细指南。

输入以下命令安装本文所需要的依赖模块:

```
pip install backtrader
```

看到 Successfully installed xxx 则说明安装成功。

2.基础使用

在开始之前，你必须要知道backtrader的数据结构特点：

```
self.dataclose[0] # 当日的收盘价
self.dataclose[-1] # 昨天的收盘价
self.dataclose[-2] # 前天的收盘价
```

这一点我在一开始使用的时候也被作者的逻辑震惊了，原来还能这么玩，总而言之，请记住这个特点，否则你可能会完全看不懂策略。

2.1 资金与佣金

Backtrader 初始化模型后，即可通过broker(经纪人)来设定初始资金，如下所示：

```
# -*- coding:utf-8 -*-
# Python 实用宝典
# 量化投资原来这么简单(1)
# 2020/04/12

import backtrader as bt

if __name__ == '__main__':
```

```
cerebro.broker.setcash(100000.0)

# 策略执行前的资金
print('Starting Portfolio Value: %.2f' % cerebro.broker.getvalue())

cerebro.run()

# 策略执行后的资金
print('Final Portfolio Value: %.2f' % cerebro.broker.getvalue())
```

现实生活中的股票交易里，每次交易都需要支付一定的佣金，比如万五（交易额每满一万元收取5元佣金）万三等，在Backtrader里你只需要这么设定即可：

```
cerebro.broker.setcommission(0.005)
```

设定需要设定每次交易买入的股数，可以这样

```
cerebro.addsizer(bt.sizers.FixedSize, stake=100)
```

2.2 加载数据

Backtrader 将数据集称作为 Data Feeds，默认的数据集是yahoo的股票数据，通过以下方式可以加载：

```
data = bt.feeds.YahooFinanceCSVData(
    dataname='数据文件所在位置',
    fromdate=datetime.datetime(2000, 1, 1),
    todate=datetime.datetime(2000, 12, 31)
)
```

当然，载入自己的数据也是可以的，只不过你需要设定每个列的含义，比如开盘价在第4列，则open=3（从0开始算起），如下所示：

```
data = bt.feeds.GenericCSVData(
    dataname='数据文件所在位置',
    datetime=2,
```

```
        volume=10,
        dtformat=('%Y%m%d'),
        fromdate=datetime(2010, 1, 1),
        todate=datetime(2020, 4, 12)
    )
```

下面，咱会使用自己的数据进行回测，这样才够有代入感。

2.3 构建策略

使用backtrader构建策略是一件很简单的事情，你只需要继承backtrader的策略类，并重写部分方法，就能实现策略。比如说重写属于我们自己的log函数：

```
class TestStrategy(bt.Strategy):
    """
    继承并构建自己的bt策略
    """

    def log(self, txt, dt=None, doprint=False):
        ''' 日志函数，用于统一输出日志格式 '''
        if doprint:
            dt = dt or self.datas[0].datetime.date(0)
            print('%s, %s' % (dt.isoformat(), txt))
```

最重要的是，重写我们自己的交易策略，比如咱在开头提到的**均线金叉死叉策略**：

```
class TestStrategy(bt.Strategy):
    """
    继承并构建自己的bt策略
    """

    def next(self):
        # 记录收盘价
        self.log('Close, %.2f' % self.dataclose[0])

        # 是否正在下单，如果是的话不能提交第二次订单
        if self.order:
            return
```

```
            # 还没买，如果 MA5 > MA10 说明涨势，买入
            if self.sma5[0] > self.sma10[0]:
                self.log('BUY CREATE, %.2f' % self.dataclose[0])
                self.order = self.buy()

        else:
            # 已经买了，如果 MA5 < MA10 ，说明跌势，卖出
            if self.sma5[0] < self.sma10[0]:
                self.log('SELL CREATE, %.2f' % self.dataclose[0])
                self.order = self.sell()
```

有用吗？待会儿我们回测后就知道了。

2.4 添加指标

backtrader内置了许多指标的计算方法，比如移动平均线、MACD、RSI等等，我们这一篇文章仅需要移动平均线MA，设置方法如下：

```
self.sma5 = bt.indicators.SimpleMovingAverage(self.datas[0], period=5)
```

其中，datas[0]是第一个数据集，period是指多少天的移动平均线，比如5，则返回MA5的相关数据。

3.策略回测

为了验证我们开头提到的策略，咱使用了 贵州茅台600519.SH 在2020年1月1日至今(2020/04/12)的股票数据。

将数据命名为600519.csv，保存在当前文件夹下，主函数如下：

```
if __name__ == '__main__':

    # 初始化模型
    cerebro = bt.Cerebro()

    # 构建策略
    strats = cerebro.addstrategy(TestStrategy)
```

```
    data = bt.feeds.GenericCSVData(
        dataname='600519.csv',
        fromdate=datetime.datetime(2010, 1, 1),
        todate=datetime.datetime(2020, 4, 12),
        dtformat='%Y%m%d',
        datetime=2,
        open=3,
        high=4,
        low=5,
        close=6,
        volume=10
    )

    cerebro.adddata(data)

    # 设定初始资金和佣金
    cerebro.broker.setcash(1000000.0)
    cerebro.broker.setcommission(0.005)

    # 策略执行前的资金
    print('启动资金: %.2f' % cerebro.broker.getvalue())

    # 策略执行
    cerebro.run()
```

最后补全策略就完成了，我们的backtrader策略如下：

```
class TestStrategy(bt.Strategy):
    """
    继承并构建自己的bt策略
    """

    def log(self, txt, dt=None, doprint=False):
        ''' 日志函数，用于统一输出日志格式 '''
        if doprint:
            dt = dt or self.datas[0].datetime.date(0)
            print('%s, %s' % (dt.isoformat(), txt))

    def __init__(self):
```

```
        self.buycomm = None

        # 五日移动平均线
        self.sma5 = bt.indicators.SimpleMovingAverage(
            self.datas[0], period=5)
        # 十日移动平均线
        self.sma10 = bt.indicators.SimpleMovingAverage(
            self.datas[0], period=10)

    def notify_order(self, order):
        """
        订单状态处理

        Arguments:
            order {object} -- 订单状态
        """
        if order.status in [order.Submitted, order.Accepted]:
            # 如订单已被处理，则不用做任何事情
            return

        # 检查订单是否完成
        if order.status in [order.Completed]:
            if order.isbuy():
                self.buyprice = order.executed.price
                self.buycomm = order.executed.comm
            self.bar_executed = len(self)

        # 订单因为缺少资金之类的原因被拒绝执行
        elif order.status in [order.Canceled, order.Margin, order.Rejected]:
            self.log('Order Canceled/Margin/Rejected')

        # 订单状态处理完成，设为空
        self.order = None

    def notify_trade(self, trade):
        """
        交易成果

        Arguments:
            trade {object} -- 交易状态
        """
```

```
            self.log('OPERATION PROFIT, GROSS %.2f, NET %.2f' %
                     (trade.pnl, trade.pnlcomm), doprint=True)

    def next(self):
        ''' 下一次执行 '''

        # 记录收盘价
        self.log('Close, %.2f' % self.dataclose[0])

        # 是否正在下单，如果是的话不能提交第二次订单
        if self.order:
            return

        # 是否已经买入
        if not self.position:
            # 还没买，如果 MA5 > MA10 说明涨势，买入
            if self.sma5[0] > self.sma10[0]:
                self.order = self.buy()
        else:
            # 已经买了，如果 MA5 < MA10 ，说明跌势，卖出
            if self.sma5[0] < self.sma10[0]:
                self.order = self.sell()

    def stop(self):
        self.log(u'(金叉死叉有用吗) Ending Value %.2f' %
                 (self.broker.getvalue()), doprint=True)
```

这份代码看起来很长，但其实把注释去掉后，实现的是很简单的逻辑。效果如何？看下图就知道了：

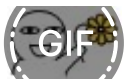
可以看到，我们初始资金是100万，每次交易100股，虽然偶尔有盈利，如果严格按照这个策略执

知乎 首发于 Python 实战教程

但是这种简单的实现方式，往往最能帮助你理性地分析该策略的合理性，如果说一个策略总是需要你主观地去加仓、减仓，那该策略势必存在问题。真正好的策略，从概率上来讲，简单回测的结果总会是盈利的。

所以这种单纯的、简单的均线金叉死叉策略有用吗？ 我认为效果有限。网上策略很多，大家也可以试试别的策略，如果有好用的，记得告诉我（滑稽）。

我们的文章到此就结束啦，如果你喜欢我们今天的Python 教程，请持续关注我们，如果对你有帮助，麻烦在下面点一个赞/在看哦



有任何问题都可以在下方留言区留言，我们都会耐心解答的！

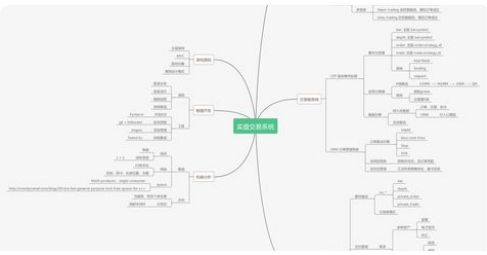
Python实用宝典 (pythondict.com)
不只是一个宝典
欢迎关注公众号：Python实用宝典

知乎 首发于 Python 实战教程

文章被以下专栏收录

Python 实战教程
来看一下这些Python教程，肯定会有收获的

推荐阅读



Quant工具箱：量化开发之事件驱动回测框架与实盘交易系统

Jesse

如何用python进行量化投资?

最近程序化交易很热， 量化金融也是我很感兴趣的一块。 国内 量化投资近几年开始盛行，如果你想对量化投资有进一步了解的话，里面有几篇量化投资策略类干货贴分享给大家，希望对各位有帮助...

金程教育

3 条评论

切换为时间排序

写下你的评论...

杨杨 2020-06-27

```
def log(self, txt, dt=None, doprint=False):
''' 日志函数，用于统一输出日志格式 '''
if doprint:
dt = dt or self.datas[0].datetime.date(0)
print('%s, %s' % (dt.isoformat(), txt))
```

感谢分享,十分实用的backtrader教程, 比市面上的强太多了

赞

伪学委 2020-10-04

只有datetime，a，close 三列数据。可以导入backtrader吗?

赞