自动产生策略系统使用说明

周袤 2018.9.26

该系统是针对有一定编程基础的策略研发人员，该平台主要实现通过数据自动产出策略，能够批量产生中高频策略，在一定程度上减轻策略开发人员的压力。

1. **系统构成**

开发示例：通过简洁明了的代码展示如何使用API

交易平台：整合了VNPY，能够直接对接实盘

1. **快速开始 — 使用步骤**
2. 安装Anaconda3，推荐5.0版本
3. 安装需要的一些Python包
   1. pip install TA\_Lib-0.4.17-cp35-cp35m-win\_amd64.whl
   2. pip install tushare
   3. pip install pymysql
   4. pip install passlib
   5. statsmodels版本必须为0.8.0
4. 若要连接CTP接口进行程序化交易，请安装VNPY（传送门：https://github.com/vnpy/vnpy），注意，安装Anaconda2时候，路径为C:\Anaconda3\envs\Anaconda2。如果talib包无法安装，请cd到安装包下，并pip install TA\_Lib-0.4.17-cp27-cp27m-win32.whl。
5. 将AutoStrategy文件夹放入Anaconda3\Lib\site-packages\ 目录下
6. 在estrategyhouse.com上注册用户名和密码
7. 打开Anaconda
8. 使用import AutoStrategy导入自动产生策略系统，期间会要求用户输入estrategyhouse.com上注册的用户名和密码
9. 使用create, deploy以及run命令去进行产生策略，部署策略，实盘或模拟盘运行策略
10. 实盘时候首先要启动Anaconda2，输入activate anaconda2, 然后cd到\vnpy-master\examples\VnTrader\ 下， python run.py
11. **API**
    1. **API命令**
12. Machine\_Learning\_Create — 产生基于机器学习的策略

实现根据K线自动产生交易策略，使用时需至少输入产生策略基于的K线，策略存储的文件夹，基于的标的代码

|  |  |  |
| --- | --- | --- |
| 参数 | Python类型 | 说明 |
| TimestampPriceX | (pandas.DataFrame) | 原始输入的K线，具体要求参见API输入章节 |
| strategyfolder | (string) | 用于存放策略的文件夹 |
| code | (string) | 交易标的代码 |
| strategy | (string) | 策略名 |
| numfeature | (int) | 单次抽取的特征数目 |
| numstrategy | (int) | 需要产生策略的数目 |
| numtry | (int) | 尝试的次数 |
| opencost | (float) | 开仓成本 |
| closecost | (float) | 平仓成本 |
| intradayclosecost | (float) | 日内平仓成本（若允许日内平仓） |

实例 strategyName=AutoStrategy.Machine\_Learning\_Create(TimestampPriceX=TimestampPriceX, strategyfolder='C:\\Users\\maozh\\Desktop\\Work\\StartegyFolder', code='SH000905',numfeature=5,numtry=500)

1. Rule\_Based\_Create — 产生基于规则的策略

实现根据K线自动产生交易策略，使用时需至少输入产生策略基于的K线，策略存储的文件夹，基于的标的代码

|  |  |  |
| --- | --- | --- |
| 参数 | Python类型 | 说明 |
| TimestampPriceX | (pandas.DataFrame) | 原始输入的K线，具体要求参见API输入章节 |
| strategyfolder | (string) | 用于存放策略的文件夹 |
| code | (string) | 交易标的代码 |
| strategy | (string) | 策略名 |
| numfeature | (int) | 单次抽取的特征数目 |
| numstrategy | (int) | 需要产生策略的数目 |
| numtry | (int) | 尝试的次数 |
| target | (string) | 学习演进目标, 目前只支持夏普 'Sharpe\_Ratio' |
| maxdepth | (int) | 最大if else判断次数 |
| fpr | (float) | 该节点是判断节点的概率 |
| ppr | (float) | 该节点是变量节点的概率 |
| popsize | (int) | 起始策略数目，种群数目 |
| maxgen | (int) | 最大演化代数 |
| mutationrate | (float) | 发生变异（Mutation）的概率 |
| breedingrate | (float) | 发生交叉(crossover)的概率 |
| pexp | (float) | 产生随机数的参数，通常选择0.9 |
| pnew | (float) | 在演化是种群中随机产生一个新个体的概率 |
| opencost | (float) | 开仓成本 |
| closecost | (float) | 平仓成本 |
| intradayclosecost | (float) | 日内平仓成本（若允许日内平仓） |

实例

strategyName=AutoStrategy.Rule\_Based\_Create(TimestampPriceX=TimestampPriceX, strategyfolder='C:\\Users\\maozh\\Desktop\\Work\\StartegyFolder', code='SH000905',numfeature=5,numtry=500)

1. Deploy — 部署策略（若需要通过CTP接口连接实盘）

用于自动连接VNPY和CTP，自动部署策略

**若没有安装VNPY/或是由于股票没有办法程序化，则在signalpath下产生交易信号数据库，可以基于交易信号数据库手动下单**

|  |  |  |
| --- | --- | --- |
| 参数 | Python类型 | 说明 |
| strategy | (string) | 策略名 |
| strategyfolder | (string) | 用于存放策略的文件夹 |
| vtSymbol | (string) | vnpy的实盘交易代码，若没有请填写模拟盘交易代码 |
| vnpypath | (string) | VNPY的所在路径，不能和vnpysettingpath同时为None |
| vnpystrategyfolder | (string) | strategy 文件夹通常是在 vnpy\\trader\\app\\ctaStrategy' 下 |
| vnpysettingpath | (string) | 用来添加策略至策略字典的路径，通常在VnTrader下 |
| signalpath | (string) | 交易信号写入的数据库，若None则默认在strategyfolder向上一级文件夹下 |

实例

AutoStrategy.Deploy(strategy=‘test’,strategyfolder='C:\\Users\\pc\\Desktop\\AutoCTA\\AutoCTATest',vtSymbol='IC1809',vnpypath='C:\\Users\\pc\\Desktop\\vnpy-master')

1. Machine\_Learning\_Run — 实盘或模拟盘运行机器学习策略

如果有VNPY了，并且经过了Deploy，在运行Machine\_Learning\_Run就是实盘；

如果没有，就是模拟盘

|  |  |  |
| --- | --- | --- |
| 参数 | Python类型 | 说明 |
| strategy | (string) | 策略名 |
| strategyfolder | (string) | 用于存放策略的文件夹 |
| vtSymbol | (string) | vnpy的实盘交易代码，若没有请填写模拟盘交易代码 |
| Newdata | (pandas.DataFrame) | 实盘输入的K线，字段与TimestampPriceX一致 |
| signalpath | (string) | 交易信号写入的数据库，若None则默认在strategyfolder向上一级文件夹下 |

实例 AutoStrategy.Machine\_Learning\_Run(strategy=strategyName,strategyfolder='C:\\Users\\maozh\\Desktop\\Work\\StartegyFolder',vtSymbol='IC1810',Newdata=Newdata)

1. Rule\_Based\_Run — 实盘或模拟盘运行基于规则的策略

如果有VNPY了，并且经过了Deploy，在运行Rule\_Based\_Run就是实盘；

如果没有，就是模拟盘

|  |  |  |
| --- | --- | --- |
| 参数 | Python类型 | 说明 |
| strategy | (string) | 策略名 |
| strategyfolder | (string) | 用于存放策略的文件夹 |
| vtSymbol | (string) | vnpy的实盘交易代码，若没有请填写模拟盘交易代码 |
| Newdata | (pandas.DataFrame) | 实盘输入的K线，字段与TimestampPriceX一致 |
| signalpath | (string) | 交易信号写入的数据库，若None则默认在strategyfolder向上一级文件夹下 |

* 1. **API输入**

1. TimestampPriceX — 原始输入数据的形式

pandas.DataFrame，产生策略所需的输入的K线

|  |  |  |
| --- | --- | --- |
| 参数 | Python类型 | 说明 |
| Code | (string) | 交易标的代码 |
| DATETIME | (datetime.datetime) | 时间戳 |
| Date | (datetime.date) | 时间戳的日期 |
| Time | (datetime.time) | 时间戳的时间 |
| open | (float) | 开盘价 |
| high | (float) | 最高价 |
| low | (float) | 最低价 |
| close | (float) | 收盘价 |
| amount | (float) | 成交额 |
| volume | (float) | 成交量 |

实例



1. StrategyCriteria.json — 接受策略的标准

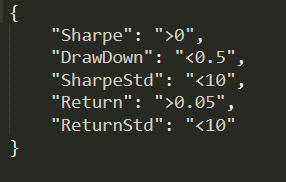
Json文件，作为接受或者拒绝策略的标准。一般位于AutoStrategy文件夹下。请用户根据自身需求调整标准

|  |  |
| --- | --- |
| 参数 | 说明 |
| DrawDown | 最大回撤 |
| Return | 平均年化收益 |
| ReturnStd | 年度之间收益的标准差 |
| Sharpe | 平均夏普 |
| SharpeStd | 年度之间夏普的标准差 |

实例

下图所代表的意思是，接受随机产生的策略标准

1. 为夏普大于0
2. 最大回撤小于50%
3. 年度之间的夏普标准差在10以内
4. 平均年化收益大于5%
5. 年度之间的收益标准差在10以内



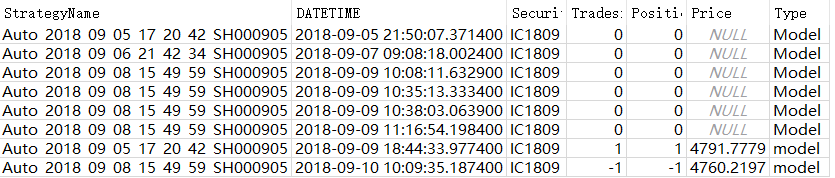
* 1. **API输出**

1. TradingSignal.db — 记录策略交易信号

Sqlite本地数据库，用于记录策略交易信号的各种参数

|  |  |  |
| --- | --- | --- |
| 参数 | Python类型 | 说明 |
| StrategyName | (string) | 策略名称 |
| DATETIME | (datetime.datetime) | 交易信号触发时间 |
| SecurityName | (string) | 交易标的 |
| Tradeside | (int) | 交易方向，1是做多，-1是做空 |
| Position | (int) | 本次交易信号的仓位（手数） |
| Price | (float) | 理论交易价格 |
| Type | (string) | 交易类型，model策略触发，stoploss止损触发 |

实例



1. Auto\_XXX.pkl — 策略文件

Pickle文件，用于记录策略的各种参数和模型

|  |  |  |
| --- | --- | --- |
| 参数 | Python类型 | 说明 |
| ParamXCohort | (dictionary) | 因子组合每个因子的输入 |
| FeaturesXCohort | (dictionary) | 因子组合 |
| RealaliasXCohort | (list) | 对于未指明输入列的因子，随机选取的数据列作为该因子计算时的输入 |
| ParamY | (dictionary) | 预测目标的输入 |
| FeatureY | (dictionary) | 预测目标 |
| RealaliasY | (dictionary) | 对于未指明输入列的预测目标，随机选取的数据列作为该预测目标计算时的输入 |
| Traindata | (pandas.DataFrame) | 训练集 |
| Method | (string) | 预测方法 |
| Methodcoef | -- | 模型超参数 |
| BacktestResult | (pandas.DataFrame) | 原始日净值 |
| EvalPerformence | Eval\_Performance类 | 交易流水分析工具 |
| TimestampSignal | (pandas.DataFrame) | 历史样本外预测值 |
| TimestampPriceX | (pandas.DataFrame) | 原始输入数据，参见API输入 |
| FinalDecision | (boolean) | 最终是否接受策略 |
| Code | (string) | 策略基于的交易标的代码 |

实例

Auto\_2018\_09\_08\_15\_49\_59\_SH000905