了解TDK量化交易策略平台

TDK平台简介

TDK平台提供基于浏览器的量化投资策略开发工具，我们的平台非常提供了多样全面的金融数据和完善的后台运算，使您可以更专注于投资想法到量化策略的转化开发工作，无需为复杂的计算机技术劳心费神。

我们的平台要求使用者运用python语言完成在线策略的编写，并利用配置文件yaml的编写向系统提供全面的回测要求。

本次大赛，我们只支持日交易历史数据回测，并不考虑交易成本，即当您的策略开始回测的时候，我们会运行回测期内的每日历史数据，并以VWAP价格作为当日成交价格——这种设定可以更为有效地保证回测时的成交效果。本次大赛不支持日内交易策略回测。

数据

目前TDK平台使用的是中国A股市场从1990年12月到目前最新的每日行情数据和基本面数据。其中，我们为您直接提供的是截至2014年以前的数据作为研究源并用于样本内回测，其余的部分我们将作为样本外数据用于提交策略的进一步回测使用，以尽可能避免策略开发过程中可能出现的过度拟合显现。

所有数据以numpy格式储存。

股票日交易历史数据

我们的原始数据来源于Wind。目前我们支持多于2800支中国A股市场股票，包括已经退市的股票，可以有效避免幸存者偏差。另外，我们对股票的派股派息等也做了相应的预处理，并保留了复权因子方面您研究使用。

股票日交易数据记录了股票一天的交易信息，保存在数据库的eod模块下。

调用日交易数据需要首先在yaml文件中Data字段下给出eod模块调用路径和所需数据数据：

```
Data:
    eod:
        ModuleId: Eod
        inpath: /mnt/ssd/eod
```

在**python**文件Signal中，使用**self.eod.XXX** 来调用相关数据，具体如下：

数据代码	数据格式	数据名称
eod.OpenPrice	ndarray （天数，股数）	开盘价
eod.ClosePrice	ndarray （天数，股数）	收盘价
eod.HighestPrice	ndarray （天数，股数）	最高价
eod.LowestPrice	ndarray （天数，股数）	最低价
eod.Volume	ndarray （天数，股数）	成交量
eod.PreClosePrice	ndarray （天数，股数）	前收盘价
eod.VWAP	ndarray （天数，股数）	成交量加权平均价格
eod.AdjFactor	ndarray （天数，股数）	复权因子
eod.PRC	ndarray （天数，股数）	不复权收盘价
eod.Turnover	ndarray （天数，股数）	成交额
eod.TradeStatus	ndarray （天数，股数）	交易状态（停牌：0，交易：1）
eod.UpDownLimitStatus	ndarray （天数，股数）	涨跌停状态（涨停：1，跌停：-1，其他：0）
eod.dates	ndarray （天数，）	日期
eod.ticker_names	ndarray （股数，）	股票代码

指数日交易历史数据

除了个股的日交易数据，我们的数据库还预备了Wind认可的大量指数日交易历史数据。几乎所有市面上公开发布的A股指数，包括多级行业、规模、概念等，都可以在我们的数据库中获取每日交易历史信息。

其调用方法和个股交易数据几乎完全一致，如果已经在 yaml 文件中声明了 eod 模块，可以直接

在python 文件Signal中调用指数数据，代码是 self.index.XXX，具体如下

数据代码	数据格式	数据名称
index.OpenPrice	ndarray （天数，指数个数）	开盘价
index.ClosePrice	ndarray （天数，指数个数）	收盘价
index.HighestPrice	ndarray （天数，指数个数）	最高价
index.LowestPrice	ndarray （天数，指数个数）	最低价
index.Volume	ndarray （天数，指数个数）	成交量
index.PreClosePrice	ndarray （天数，指数个数）	前收盘价
index.Turnover	ndarray （天数，指数个数）	成交额
index.dates	ndarray （天数，）	日期
index.ticker_names	ndarray （指数个数，）	指数代码

考虑到指数运用往往和股票的批量处理不同，使用中如果希望获得指定某个指数的交易数据，可以利用 ticker_name 定位；或将 ndarray 的数据格式处理成 dataframe 格式。

基本面数据

我们为您提供了丰富的上市公司财务数据，采自上市公司发布的资产负债表、利润表和现金流量表。上市公司更正非当期季报的情况已经处理，同样的，借壳上市的财务处理也已经完成，避免了研究偏差。考虑到我国上市公司会计准则于2007年有过重大变更，我们建议您使用2007年及之后的财务数据研究开发策略。

每个财务数据都包含若干个lagged versions, 代表所在时点下倒退N个季度(暂定保留四个季度)的数据数值，图表中#的取值为(0,1,2,3,4)。**以息税折旧摊销前利润**为例：

ISQ0_EBITDA （最近季报）

ISQ1_EBITDA（比最近往前一次季报）

ISQ2_EBITDA（比最近往前二次季报）

ISQ3_EBITDA（比最近往前三次季报）

ISQ4_EBITDA（比最近往前四次季报）

每个基本面数据都对应了一个per share的版本,文件BSQP#_TOT_ASSETS对应了文件BSQ#_TOT_ASSETS，计算如下（除EPS之外）,:

ISQP#_tot_profit = ISQ#_tot_profit / Q#_tot_shr * AdjFactor[-Q#]

EPS的计算比较例外，因为他们本身就是per share的数值，ISQP#_S_FA_EPS_BASIC ISQP#_S_FA_EPS_DILUTED 代表了EPS/EPSD的per share版本

财务数据保存在fund模块下，调用首先需要在yaml文件中的Data字段下给出fund路径和需要的财务数据名称:

```
Data:
    fund:
        ModuleId: Eod
        inpath: /mnt/ssd/fundmental
        names: [CAPQ0_MKTCAP, ISQP0_TOT_OPER_REV]
```

在**python**文件**Signal**中，使用**self.fund.XXX** 来调用相关数据（**XXX** 必须是**yaml**文件中**names**字段中的明确指标），具体如下:

数据代码	名称	来源	单位
BSQ#_TOT_ASSETS	资产总计	Balance	元
BSQ#_TOT_LIAB	负债总计	Balance	元
BSQ#_MINORITY_INT	少数股东权益	Balance	元
BSQ#_TOT_SHRHLDR_EQY_EXCL_MIN_INT	股东权益合计（不含少数股东权益）	Balance	元
BSQ#_TOT_SHRHLDR_EQY_INCL_MIN_INT	股东权益合计（含少数股东权益）	Balance	元
BSQ#_TOT_LIAB_SHRHLDR_EQY	负债及股东权益合计	Balance	元
BSQ#_MONETARY_CAP	货币资金	Balance	元
EVQ#	企业价值	Balance	元
ISQ#_TOT_OPER_REV	营业总收入	Income	元
ISQ#_TOT_OPER_COST	营业总成本	Income	元
ISQ#_OPER_PROFIT	营业利润	Income	元

ISQ#_TOT_PROFIT	利润总额	Income	元
ISQ#_NET_PROFIT_INCL_MIN_INT_INC	净利润（含少数股东权益）	Income	元
ISQ#_NET_PROFIT_EXCL_MIN_INT_INC	净利润（不含少数股东权益）	Income	元
ISQ#_S_SF_EPS_BASIC	基本每股收益	Income	元
ISQ#-S_FA_EPS_DILUTED	稀释每股权益	Income	元
ISQ#_EBIT	息税前利润	Income	元
ISQ#EBITDA	息税折旧摊销前利润	Income	元
CFSQ#_STOT_CASH_INFLOWS_OPER_ACT	经营活动现金流入小计	CashFlow	元
CFSQ#_STOT_CASH_OUTFLOWS_OPER_ACT	经营活动现金流出小计	CashFlow	元
CFSQ#_NET_CASH_CLOWS_OPER_ACT	经营活动产生的现金流量净额	CashFlow	元
CFSQ#_STOT_CASH_INFLOWS_INV_ACT	投资活动现金流入小计	CashFlow	元
CFSQ#_STOT_CASH_OUTFLOWS_INC_ACT	投资活动现金流出小计	CashFlow	元
CFSQ#_NET_CASH_FLOWS_INV_ACT	投资或定产生的现金流量金额	CashFlow	元
CFSQ#_STOT_CASH_INFLOWS_FNC_ACT	筹资活动现金流入小计	CashFlow	元
CFSQ#_STOT_CASH_OUTFLOWS_FNC_ACT	筹资活动现金流出小计	CashFlow	元
CFSQ#_NET_CASH_FLOWS_FNC_ACT	筹资活动产生的现金流量净额	CashFlow	元
CFSQ#_NET_INCR_CASH_CASH_EQU	现金及现金等价物净增加额	CashFlow	元
CFSQ#_CASH_CASH_EQU_BEG_PERIOD	期初现金及现金等价物余额	CashFlow	元
CFSQ#_CASH_CASH_EQU_END_PERIOD	期末现金及现金等	CashFlow	元

	价物余额		
DIVQ0_DIV	每股派息（税前）	Dividend	元
CAPQ#_TOT_SHR	总股本	Capitalization	股
CAPQ#_FLOAT_SHR	流通股	Capitalization	股
CAPQ#_FLOAT_A_SHR	流通A股	Capitalization	股
CAPQ#_MKTCAP	市值	Capitalization	元
CAPQ#_FLOAT	流通市值	Capitalization	元
CAPQ#_FLOAT_A	流通A股市值	Capitalization	元

基于基本的财务数据，研究过程中往往需要计算衍生因子和指标。您可以在策略开发中自主计算您需要的任何衍生因子。同时，为了方便您的使用，我们已经为您提供了一些常用的财务基本面因子

常用因子保存在 fund/factors 模块下，调用方法与基本财务数据相似，yaml 文件中需要定义字段:

```
Data:
    factors:
        ModuleId: Eod
        inpath: /mnt/ssd/fundmental/factors
        names: [TRL_YLD, EPSY_T12MB, BP, ROE, EPSY_T12MD, EPSY_T12MO]
```

在python文件Signal中，使用 self.factors.XXX 来调用相关数据（XXX 必须是yaml文件中names字段中的明确指标），具体如下:

数据代码	名称	类型	范畴
TRL_YLD	Dividend yield, trailing 12M	Value	Dividends
~~EPSY_T12MB~~	Earnings yield, trailing 12M, Basic	Value	Earnings
~~EPSY_T12MD~~	Earnings yield, trailing 12M, Fully Diluted	Value	Earnings
EPSY_T12MO	Operating earning yield, trailing 12M, Basic	Value	Earnings
~~CFY_CF~~	Operating cash flow yield	Value	Cash Flow

FCF_YLD	Free cash flow yield	Value	Cash Flow
CHGCE_YLD	change in cash equivalent yield	Value	Cash Flow
SALY	Sales-to-price, trailing 12M	Value	Sales
BP	Book-to-price	Value	Book
~~EBIT_EV~~	EBIT/EV	Value	EBIT
~~EBITDA_EV~~	EBIDA/EV	Value	EBIT
YOY_DPS_G	Year-over-year DPS growth	Growth	Short tern
YOY_EPS_G	Year-over-year quarterly EPS growth	Growth	Short tern
YOY_CPS_G	Year-over-year quarterly CFPS growth	Growth	Short tern
YOY_SAL_G	Year-over-year quarterly SAL growth	Growth	Short tern
RTN21D	Total return, 21D(1M)	Price momentum and reversal	Short-term reversal
RTN63D	Total return, 63D(3M)	Price momentum and reversal	Mid-term reversal
RTN126D	Total return, 126D(6M)	Price momentum and reversal	Mid-term reversal
RTN252D	Total return, 252D(12M)	Price momentum and reversal	Mid-term reversal
RTN12_1M	12M - 1M total return	Price momentum and reversal	Mid-term reversal
~~ROE~~	ROE, trailing 12M	Quality	Profitability
~~ROE_CHG~~	Year-over-year change in trailing ROE	Quality	Profitability
ROA	ROA, trailing 12M	Quality	Profitability
ROA_CHG	Year-over-year change in trailing ROA	Quality	Profitability
CFROE_CF	Cash flow return in equity	Quality	Profitability
CFROA_CF	Cash flow return in asset	Quality	Profitability
SAL_TA	Sales to total assets (asset turnover)	Quality	Profitability
CHG_SAL_TA	Year-over-year change in asset trunover	Quality	Profitability
NET_MATGIN	Net profit margin	Quality	Profitability
GROSS_MARGIN	Operating profit margin	Quality	Profitability
CHG_G_MARGIN	Change in gross margin	Quality	Profitability

EBITDA_M	EBITDA margin	Quality	Profitability
TOT_DEBT_TE	Total debt / total equity	Quality	Leverage
TOT_DEBT_TA	Total debt / total assets	Quality	Leverage
TOT_D_MKTEQUITY	Total debt / market value of equity	Quality	Leverage
TOT_D_MKTASSETS	Total debt / market value of assets	Quality	Leverage
CF_CF_DEBT	Cash flow / debt	Quality	Leverage
REAL_VOL_1YD	Realized volatility, 1Y, daily	Technical	Risk

信号文件编写

在TDK平台，我们将发掘市场规律，寻求超额收益的方法称之为信号，信号开发层面**不考虑交易成本和交易实现方式，**您的主要任务也就是开发量化投资信号。

信号文件包括配置文件 .yaml 和 策略主体python文件 .py，我们可以分别称其为YAML 和 Signal。

YAML

YAML 的主要作用是定义信号，包括定义信号名称、使用的数据范围、调用的后期处理方法等，利用字段map的方法为信号研究属性定义。

下面是TDK默认的入门信号Signal_101的YAML，详细的注释可以让您理解各个字段的含义。

- 加亮字段决定了信号的身份标示，故每当您希望生成新的信号而非利用修改覆盖原信号，请务必全部按需更改。

```
# 信号Signal_101 配置文件
# Signal_101.yaml
Name: Signal_101                                    #信号名称，这是信号的身份标示
Data:                                               #Data 是必须字段
   eod:                                             #eod模块的id和路径都是必须的
      ModuleId: Eod
      inpath: /mnt/ssd/eod
Macro:                                              #Macro是必须字段
   capital: 5e7                                     #初始资本金
```

```
  end: 20131231                                  #回测结束日期
  maxlookback: 100                              #计算需要回测开始前获取的数据长度
  start: 20010101                                #回测开始日期

Modules:                                         # Modules是必须字段，
                                                  # 给出应用operator的调用路径
  DECAY:
    Module: signalop/linear_decay.py
  POWER:
    Module: signalop/power.py
  RANK:
    Module: signalop/rank.py
                                                  # 给出数据路径
  IndexMembers:
    Module: umgr/index_members.py
  Eod:
    Module: dmgr/eod_datamanager.py

# 给出portfolio字段中signalid 对应的signal 信息
  Signal_101:
    Module: Signal_101.py                       #输入路径直接将Name加.py 后缀
  TopLiquid:
    Module: umgr/topliquid_manager.py
Type: 1                                          # 交易形式 1：只做多；
                                                  # 0：既做多也做空；-1：只做空
Turnover: 1                                       # 调仓计算周期（天）
StockLock: 0                                     # 持仓锁定状态 0：不锁定（默认）；
                                                  # 1：锁定，即持仓不随其他股票被动变动

# Portfolio字段是必须的
Portfolio:
# 应至少包含一个Signal
  Signal:
    signalid: Signal_101                         # 给出信号id，一般情况下名称应同Name

# SignalOp字段是可选字段，用来使用不同方法调整signal输出权重
# SignalOp是一个列表，不同的operator顺序会产生不同的结果。
# 本例中给出部分可用operator，实际中请酌情选用，不要使用无用的operater
  SignalOp:
    - ModuleId: DECAY                            #线性衰减，通常用来降低信号的换手率。
      days: 3                                    #衰减强度为3天

    - ModuleId: TOP                              #保留输入值的TOP部分
      percent: 0.2                                #保留信号强度最大的20%

    - ModuleId: RANK                             #将输入值转化成rank index
                                                  #rank之后的NaN值依然为NaN
```

```
    - ModuleId: POWER                          #将输入值转换为其power运算的结果
      base: 2                                  #power base 为2

    - ModuleId: INDNEU                         #去除行业间的bias

# Universe，必需字段，可投资标的资产范围，请按需选择
Universe:
  IndexMembers:
    index_code: '000905'                      # 投资范围为中证500权重股
                                               # ‘000300’,‘000016’ 也是常用选股范围
# Benchmark，非必需字段，默认为沪深300指数，请酌情更改
# 请尽量保持benchmark 与 universe 的一致性
Benchmark: '000905'                                  # 中证500指数
```

Signal

Signal 是策略的实现部分，是信号开发的核心。

Signal 本身是一个模块式的开发内容，利用继承后台已有的更为强大和复杂的运算内容，使您更为便利地将开发精力集中在想法的实现上而非计算机技术上。

所以，首要地，Signal文件必需引入以下内容：

```
import numpy as np                             # numpy 是TDK默认的数据处理方案，必需引入

from simlib.signal.base import Signal          # 继承 Signal 基类，使开发接入后台
from simlib.signal.lib import *                # 亦是开发需要的重要内容
```

定义信号子类并继承基类**Signal**：

```
class SampleSignal(Signal):

                    .
                    .
                    .

    def generate(self,di):

                    .
                    .
                    .
```

```
signal = SampleSignal
```

- 实现信号主要是实现generate()函数， generate函数只有一个参数di, di的是一个下标，代表第di天。generate返回一个向量，包含每只股票的权重。
- 数据的使用参考：数据。
- 一些模式触发的信号，例如在模式产生之前就维持之前的信号值不变，可以通过self.weights[-1]来获取 系统会在roll(di)的时候更新这个变量，它始终是一个shape = (股票个数，di)的ndarray。

下面是TDK默认的入门信号Signal_101的Signal，详细的注释可以有效帮您快速上手。此外，TDK另有提供教学信号 Signal_102_GroupNIndex 和 Signal_102_Fund 分别帮助您使用行业分组、指数数据和基本面数据。

```
# coding=utf-8
# Signal_101.py

import numpy as np

from simlib.signal.base import Signal
from simlib.signal.lib import *


class Signal_101(Signal):                                   #继承Signal是必须的

    required_data = ["ClosePrice"]                          # required_data用来准备数据，不是必须的。
# 忽略required_data 会准备所有DataManager的数据

    def prebatch(self):                                             # prebatch 函数预处理数据，并非必需
                                                                    # 你当然也可以编写其他合适的函数


        data = self.eod.ClosePrice                          # 读取量价数据 ClosePrice。

        five_ma = self.function_wrapper("MA", data, timeperiod=5)    # data值的五日移动平均值
        ten_ma = self.function_wrapper("MA", data, timeperiod=10)    # data值的十日移动平均值
        self.dif = five_ma - ten_ma                         #赋值类内全局变量dif待用。

# generate 函数是信号计算和确认的核心部分，其输入变量是时间节点di
# 函数任务是返回 di 时点的所有头寸相对权重比例weights

    def generate(self, di):

        r = []
```

```
        data = self.dif                                                 #获得dif值。
# 这里的数据包含universe所有标的完整的时间序列，不能直接用于计算。
# 注意：任何包含di时点或之后未来数据的计算行为系统都将拒绝并报错

        observed = data[di-2:di].T                                  # di时点下“昨天”和“前天”的数据
        last_wgt = self.weights[-1]                                  # 定义lat_wgt表示前一时点weights

        for ix, ticker in enumerate(observed):                          #针对每一个标的资产
            w = last_wgt[ix]                                          #第ix资产配置权重w初始化
# 信号判断主体：如果五日移动平均线上穿十日移动平均线，则持有该资产多头头寸；
# 反之，不持该资产或持有其空头头寸（允许做空情况下）

            if ticker[1] < 0 and ticker[0] > 0:
                w = ticker[1] / ticker[0]

            if ticker[1] > 0 and ticker[0] < 0:
                w = abs(ticker[1] / ticker[0])

# 只做多情况下，负数权重将被认定为“不持有”

            r.append(w)

        res = np.array(r)

#返回universe涵盖标的权重数组，完成di时点计算和信号输出。
        return res

signal = Signal_101
```

目前支持的Python模块

目前TDK平台支持多种常用Python模块，您可以按需要手动引入。您可以引入我们目前支持的Python模块来实现丰富多彩的数据处理。当然，如果您有其他自己擅长和青睐的Python模块，请不要犹豫，通知我们，我们会及时满足您的需求。

下面是目前我们已经提供的Python模块和版本号。

模块名	版本号	模块名	版本号
Cython	0.22	python-dateutil	2.4.2

h5py	2.5.0	pytz	2015.4
iniparse	0.4	pyxattr	0.5.1
MySQL-python	1.2.3	PyYAML	3.11
numpy	1.9.2	scikit-learn	0.16.1
pandas	0.16.1	scipy	0.15.1
pip	6.1.1	setuptools	16.0
pkgconfig	1.1.0	six	1.9.0
pygpgme	0.3	TA-Lib	0.4.8
pyliblzma	0.5.3	urlgrabber	3.10
pymongo	3.0.2	yum-metadata-parser	1.1.4

其中，以下模块可能会是您经常用到的工具

numpy

numpy 是TDK默认的数据处理方案，所以是必需引入的模块。其功能齐全强大，可以满足大部分基本的处理需求。

文档链接： http://www.numpy.org/

pandas

pandas 是基于numpy 的数据处理和分析模块，在时间序列分析上有比较好的便利性。

文档链接： http://pandas.pydata.org/

scipy

scipy 在numpy 的基础上增加了众多的数学、科学以及工程计算中的常用模块，例如线性代数、常微分方程数值求解、信号处理等，其强大的统计模块会使策略开发过程更加简便。

文档链接： http://www.scipy.org/

talib

talib 是被交易软件开发者广泛使用的金融市场数据技术分析工具。

文档链接： http://mrjbq7.github.io/ta-lib/funcs.html

需要注意的是，talib 只能针对单一股票（或其他数据）时间序列分析，为了方便您的数据处理，我们提供了封装函数 **function_wrapper** 实现 talib 函数能够接收metrics，其应用表达式为：

real = function_wrapper("XX", a, b, …)

其中，XX 为 talib 中的对应函数名，a, b, … 则为 XX 应用中需要调用的变量。

scikit-learn

sklearn 是python下的机器学习模块，包含有丰富的机器学习方法。

文档链接： http://scikit-learn.org/stable/