基于XGboost 的alpha mining

by 郭嘉

不少投资者对于技术面分析嗤之以鼻，但如果在找到合适的alpha后利用股指期货对冲掉系统性风险（neutralization部分人翻译为中性，其实其更常用的意思为消灭），收入应当还是比较乐观的。然而一些技术面分析方法被私募基金公司作为私有财产而未公开，下面仅通过目前比较流行的机器学习算法XGboost获得3个alpha作为简要示例，该算法被数据分析师在许多数据分析比赛中取得优异成绩，并且由其能够并行，在大规模数据上也能很快地进行训练，从精确度和速度两方面来说都是一个比较合适的选择。

导入必要的库

- baostock:导入股票数据（tushare现在需要收费所以使用这个）
- joblib:python并行计算库（后续需要大量处理数据，不并行会非常慢）
- xgboost算法实现模块和sklearn中的神经网络模块（后面发现神经网络找到的alpha效果一般，可以作为警示）

In [1]:
```
import baostock as bs
import pandas as pd
from joblib import Parallel,delayed
import numpy as np
import multiprocessing

from sklearn.neural_network import MLPRegressor
import xgboost as xgb
```

进行数据下载和一些必要的数据预处理

这里T日对应行内主要的特征包括T-5日到T-1日的每日开盘价open,收盘价close，最高价high，最低价low，平均交易价格vwap，高低价差hl，开低价差ol，收低价差cl，均低价差vl以及换手率turn

以上共$10*5=50$个特征

对vwap，hl，ol，cl，vl，turn作T-5日到T-1日的两两相关系数得到剩下的15个特征

In [2]:
```
sz_prefixs=['000','001','002','003']
sh_prefixs=['600','601','603','605','688']

sz_codes=["sz."+prefixs+str(suffixs).zfill(3) for prefixs in sz_prefixs for suffixs in range(1000)]
sh_codes=["sh."+prefixs+str(suffixs).zfill(3) for prefixs in sh_prefixs for suffixs in range(1000)]
stock_codes=sz_codes+sh_codes
stock_codes.append('sh.689009')

shift_keys=['open','close','high','low','vwap','hl','ol','cl','vl','turn']
keys_dict={'open': ['T-1_open', 'T-2_open', 'T-3_open', 'T-4_open', 'T-5_open'], 'close': ['T-1_close', 'T-2_close', 'T-3_close', 'T-4_close', 'T-5_close'], 'high': ['T-1_high
```

In [3]:
```
def get_stock_data(stock_code):
    lg = bs.login()
    rs_result = bs.query_history_k_data_plus(stock_code,
        "date,code,open,high,low,close,volume,amount,turn,isST",
        start_date='2020-12-25', end_date='2021-12-01',
        frequency="d", adjustflag="3")
    df_result = rs_result.get_data()

    if df_result.empty==False:
        df_result=df_result.apply(lambda x: pd.to_numeric(x) if x.name not in ['date','code'] else x)
        if 1 not in list(df_result['isST']):
            df_result['T+1_open'] = df_result['open'].shift(-1)
            df_result['return'] = df_result['T+1_open'] / df_result['open'] -1
            df_result['vwap'] = df_result['amount']/df_result['volume']

            df_result['hl']=df_result['high']-df_result['low']
            df_result['ol']=df_result['open']-df_result['low']
            df_result['cl']=df_result['close']-df_result['low']
            df_result['vl']=df_result['vwap']-df_result['low']

            for shift_key in shift_keys:
                keys=[]
                for shift_time in range(1,6):
                    df_result['T-'+str(shift_time)+'_'+shift_key]=df_result[shift_key].shift(shift_time)
            df_result.drop(['open','high','low','close','volume','amount','isST','T+1_open'],axis=1,inplace=True)
            df_result.dropna(how='any',inplace=True)
            return df_result[5:-1]
```

In [4]:
```
temp=Parallel(n_jobs=-1)(delayed(get_stock_data)(stock_code) for stock_code in stock_codes)
stock_list=[stock for stock in temp if stock is not None]
stock_data=pd.concat(stock_list,ignore_index=True)
```

In [5]:
```
def get_correlation(stock_data):
    for i,shift_key1 in enumerate(shift_keys):
            for shift_key2 in shift_keys[i+1:]:
                df1=stock_data[keys_dict[shift_key1]]
                df1.columns=[1,2,3,4,5]
                df2=stock_data[keys_dict[shift_key2]]
                df2.columns=[1,2,3,4,5]
                stock_data['cor_'+shift_key1+'_'+shift_key2]=df1.corrwith(df2,method='pearson',axis=1)
                cor_keys.append('cor_'+shift_key1+'_'+shift_key2)
```

In [6]:
```
cor_keys=[]
shift_keys=['vwap','hl','ol','cl','vl','turn']
get_correlation(stock_data)
stock_data.dropna(how='any',inplace=True)
```

试探性看一看后15个与相关系数有关的特征如果作为alpha的话RankIC表现

In [7]:
```
def get_rank(date,df,key):
    ranked_data=df[df['date']==date].sort_values(by=key,ascending=False)
    ranked_data['rank_'+key]=range(1,len(ranked_data)+1)
    return ranked_data
```

In [8]:
```
stock_data=pd.concat(Parallel(n_jobs=-1)(delayed(get_rank)(date,stock_data,'return') for date in stock_data['date'].unique()))
```

In [9]:
```
from tqdm import tqdm
for cor_key in tqdm(cor_keys):
    stock_data=pd.concat(Parallel(n_jobs=-1)(delayed(get_rank)(date,stock_data,cor_key) for date in stock_data['date'].unique()))
```

```
100%|████████████████████████████████████████████████████████████████| 15/15 [22:31<00:00, 90.10s/it]
```

In [10]:
```
alphas={possible_alpha:stock_data['rank_return'].corr(stock_data[possible_alpha]) for possible_alpha in cor_keys if abs(stock_data['rank_return'].corr(stock_data[possible_alph
```

In [11]:
```
cor_keys
```

Out[11]:
```
['cor_vwap_hl',
 'cor_vwap_ol',
 'cor_vwap_cl',
 'cor_vwap_vl',
 'cor_vwap_turn',
 'cor_hl_ol',
 'cor_hl_cl',
 'cor_hl_vl',
 'cor_hl_turn',
 'cor_ol_cl',
 'cor_ol_vl',
 'cor_ol_turn',
 'cor_cl_vl',
 'cor_cl_turn',
 'cor_vl_turn']
```

并不理想，最好的RankIC只有0.026

In [12]:
```
alphas
```

Out[12]:
```
{'cor_hl_cl': 0.025541762804313466,
 'cor_cl_vl': 0.021228305614368313,
 'cor_cl_turn': 0.020273131098434056}
```

试探性地使用单层神经网络处理前25个特征，以其输出作为alpha，效果也不尽理想

In [13]:
```
feat=['T-1_open', 'T-2_open', 'T-3_open', 'T-4_open', 'T-5_open']+ ['T-1_close', 'T-2_close', 'T-3_close', 'T-4_close', 'T-5_close']+ ['T-1_high', 'T-2_high', 'T-3_high', 'T-4
```

In [17]:
```
for neurons in tqdm(range(1,30)):
    regr = MLPRegressor(random_state=2, hidden_layer_sizes=(neurons),max_iter=200)
    regr.fit(stock_data[feat[:25]],stock_data['return'])
    stock_data['alpha1']=regr.predict(stock_data[feat[:25]])
    stock_data=pd.concat(Parallel(n_jobs=-1)(delayed(get_rank)(date,stock_data,'alpha1') for date in stock_data['date'].unique()))
    RankIC=stock_data['rank_alpha1'].corr(stock_data['rank_return'])
    if abs(RankIC)>0.03:
        print(neurons,':',RankIC)
```

```
100%|████████████████████████████████████████████████████████████████| 29/29 [55:20<00:00, 114.50s/it]
```

使用XGboost处理得到了三个令人满意的alpha

第一个alpha使用T-5日到T-1日的open，close，high，low，vwap作为输入，RankIC为0.08

$$alpha1=XGboost\ (open_{T-5:T-1},close_{T-5:T-1},high_{T-5:T-1},low_{T-5:T-1},vwap_{T-5:T-1},hyperparameters1)$$

In [24]:
```
xgb_model = xgb.XGBRegressor(n_jobs=multiprocessing.cpu_count())
xgb_model.fit(stock_data[feat[:25]],stock_data['return'])
stock_data['alpha1']=xgb_model.predict(stock_data[feat[:25]])
stock_data=pd.concat(Parallel(n_jobs=-1)(delayed(get_rank)(date,stock_data,'alpha1') for date in stock_data['date'].unique()))
RankIC=stock_data['rank_alpha1'].corr(stock_data['rank_return'])
RankIC
```

Out[24]:
```
0.0808810435150292
```

第二个alpha使用T-5日到T-1日的hl，ol，cl，vl，turn作为输入，RankIC为0.13

$$alpha2=XGboost\ (hl_{T-5:T-1},ol_{T-5:T-1},cl_{T-5:T-1},vl_{T-5:T-1},turn_{T-5:T-1},hyperparameters2)$$

In [33]:
```
xgb_model = xgb.XGBRegressor(n_jobs=multiprocessing.cpu_count())
xgb_model.fit(stock_data[feat[25:50]],stock_data['return'])
stock_data['alpha2']=xgb_model.predict(stock_data[feat[25:50]])
stock_data=pd.concat(Parallel(n_jobs=-1)(delayed(get_rank)(date,stock_data,'alpha2') for date in stock_data['date'].unique()))
RankIC=stock_data['rank_alpha2'].corr(stock_data['rank_return'])
RankIC
```

Out[33]:
```
0.12862307889420152
```

第三个alpha使用vwap，hl，ol，cl，vl，turn作T-5日到T-1日的两两相关系数作为输入，RankIC为0.09

$$alpha3=XGboost\ (Correlationset(vwap，hl，ol，cl，vl，turn)_{T-5:T-1},hyperparameters3)$$

In [34]:
```
xgb_model = xgb.XGBRegressor(n_jobs=multiprocessing.cpu_count())
xgb_model.fit(stock_data[feat[50:]],stock_data['return'])
stock_data['alpha3']=xgb_model.predict(stock_data[feat[50:]])
stock_data=pd.concat(Parallel(n_jobs=-1)(delayed(get_rank)(date,stock_data,'alpha3') for date in stock_data['date'].unique()))
RankIC=stock_data['rank_alpha3'].corr(stock_data['rank_return'])
RankIC
```

Out[34]:
```
0.09408876184880767
```

上述三个alpha的RankIC 在2021-01-04到 2021-11-30日的3234只沪深A股上测得

In [35]:
```
len(stock_data['code'].unique())
```

Out[35]:
```
3234
```