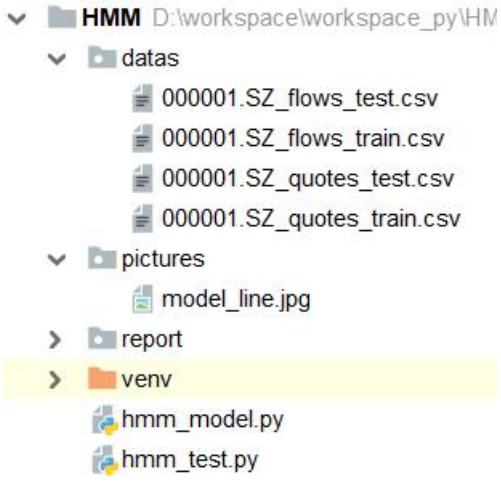
HMM 项目介绍

一、简介

HMM 项目是基于隐马尔可夫模型的股票预测及回测的系统。

二、项目结构

HMM 项目的目录结构如下图



datas 文件夹保存股票数据的 csv 文件，csv 文件名的末尾 train 和 test 分别表示是训练集和测试集的数据。

pictures 文件夹保存模型回测结果的曲线图，曲线图中绿线表示股票实际收盘价走势，红线表示模型回测的走势（不含手续费），黄线表示模型回测的走势（含手续费）。

report 文件是本报告所在文件夹。

venv 文件夹忽略。

hmm_model.py 文件是对隐马尔可夫模型的抽象类，包含模型的训练和回测。

hmm_test.py 文件获取股票的训练集和测试集数据，并调用模型的训练和回测。

三、代码流程说明

（1）hmm_test.py

1.1 获取训练集的数据（9-38 行）

```
start_date_train = '20150101'
end_date_train = '20181231'

# 导入金融数据
ts.set_token('94fb9d6bf6205a73f0337eb7d397be9911e06eaf902cb2074bc36e9b')
pro = ts.pro_api()
quotes = ts.pro_bar(ts_code=stock_code, start_date=start_date_train, end_date=end_date_train, adj='qfq')  # 前复权
#直接保存
quotes.to_csv('datas/'+stock_code+'_quotes_train.csv')
# 读csv
df1 = pd.read_csv('datas/'+stock_code+'_quotes_train.csv')
X1 = np.array([q for q in reversed(df1['pct_chg'])])  # 涨跌幅

flows = pro.moneyflow(ts_code=stock_code, start_date=start_date_train, end_date=end_date_train)
#直接保存
flows.to_csv('datas/'+stock_code+'_flows_train.csv')
# 读csv
df2 = pd.read_csv('datas/'+stock_code+'_flows_train.csv')
buy_lg_amount = np.array([q for q in reversed(df2['buy_lg_amount'])])      # 大单买入金额（万元）
sell_lg_amount = np.array([q for q in reversed(df2['sell_lg_amount'])])    # 大单卖出金额（万元）
buy_elg_amount = np.array([q for q in reversed(df2['buy_elg_amount'])])    # 特大单买入金额（万元）
sell_elg_amount = np.array([q for q in reversed(df2['sell_elg_amount'])])  # 特大单卖出金额（万元）
net_mf_amount = np.array([q for q in reversed(df2['net_mf_amount'])])      # 净流入额（万元）
total_amount = buy_lg_amount + buy_elg_amount + sell_lg_amount + sell_elg_amount  # 日总流动资金，计算方式为（大单买入金额 + 大单
X2 = net_mf_amount / total_amount   # 资金日净流入占当日所有流动资金的比例，这里计算方式为  净流入额 / 日总流动资金

X3 = []
for i in range(len(total_amount) - 1):
    X3.append((total_amount[i + 1] - total_amount[i]) / total_amount[i])
X3 = np.array(X3)     # 日总流动资金环比，这里计算方式为 （第二天日总流动资金 - 第一天日总流动资金）/ 第一天日总流动资金
```

1.2 构建并训练模型（40-43 行）

```
# 构建模型
model = HmmModel(state_num=10, sample_len=5)
# 训练模型，用X2和X3训练
model.train_model(X1[1:], X2[1:], X3)
```

1.3 获取测试集的数据（48-81 行）

```
#直接保存
quotes_test.to_csv('datas/'+stock_code+'_quotes_test.csv')
# 读csv
df1_test = pd.read_csv('datas/'+stock_code+'_quotes_test.csv')
close_test = np.array([q for q in reversed(df1_test['close'])])  # 收盘价
dates_test = np.array([i for i in range(len(df1_test['close']))])  # 第多少天

high_test = np.array([q for q in reversed(df1_test['high'])])  # 最高价
low_test = np.array([i for i in reversed(df1_test['low'])])  # 最低价
mean_test = (high_test + low_test) / 2.0  # 均价，在使用  均价 = 成交额 / 成交量  计算时，数据和前一日的收盘价差别过大，似乎数据有问题，

X1_test = np.array([q for q in reversed(df1_test['pct_chg'])])  # 涨跌幅

flows = pro.moneyflow(ts_code=stock_code, start_date=start_date_test, end_date=end_date_test)
#直接保存
flows.to_csv('datas/'+stock_code+'_flows_test.csv')
# 读csv
df2_test = pd.read_csv('datas/'+stock_code+'_flows_test.csv')
buy_lg_amount = np.array([q for q in reversed(df2_test['buy_lg_amount'])])
sell_lg_amount = np.array([q for q in reversed(df2_test['sell_lg_amount'])])
buy_elg_amount = np.array([q for q in reversed(df2_test['buy_elg_amount'])])
sell_elg_amount = np.array([q for q in reversed(df2_test['sell_elg_amount'])])
net_mf_amount = np.array([q for q in reversed(df2_test['net_mf_amount'])])
total_amount = buy_lg_amount + buy_elg_amount + sell_lg_amount + sell_elg_amount
X2_test = net_mf_amount / total_amount  # 资金日净流入占当日所有流动资金的比例，这里计算方式为  净流入额 / 日总流动资金

X3_test = []
for i in range(len(total_amount) - 1):
    X3_test.append((total_amount[i + 1] - total_amount[i]) / total_amount[i])
X3_test = np.array(X3_test)  # 日总流动资金环比，这里计算方式为 （第二天日总流动资金 - 第一天日总流动资金）/ 第一天日总流动资金
```

1.4 对测试集用训练好的模型进行回测（84 行）

```
83    # 回测模型，用X2和X3回测
84    model.back_test(X1_test[1:], close_test[1:], mean_test[1:], X2_test[1:], X3_test)
```

（2）hmm_model.py　定义模型的类名为 HmmModel

2.1 类的属性（9-18 行）

```
9     state_num = 0  # 隐状态个数
10    sample_len = 0  # 每次训练的样本长度
11
12    data_num = 0  # 训练传入的数据的组数
13
14    model = None  # 高斯隐马尔可夫模型
15
16    up_state = []  # 上涨的隐状态
17    down_state = []  # 下跌的隐状态
18    next_state = None  # 每个状态所对应的下一个状态
```

2.2 类的构造方法（20-22 行）

```
20    def __init__(self, state_num=10, sample_len=5):
21        self.state_num = state_num
22        self.sample_len = sample_len
```

2.3 训练模型的方法（24-85 行）

36-44 行：对传入的训练的数据做归一化处理并整合成一个二维矩阵

```
36        x_list = []
37        for x_temp in xs:
38            if len(x_temp) != x_len:
39                print('error: XS参数长度不相等！！！')
40                return
41            x_list.append(scale(x_temp))
42            self.data_num += 1
43
44        X = np.column_stack(x_list)  #训练集
```

53-55 行：用上面整合的二维矩阵训练模型

```
53        model = GaussianHMM(n_components=self.state_num, n_iter=500, tol=0.001)
54        model.fit(X, lengths)  # 训练模型————学习问题
55        self.model = model
```

74-77 行：判断每个隐状态表示涨还是跌，如下两种情况之外的求和结果表示震荡的隐状态

```
74            if judge_result[i] > 0:  # 隐状态的涨跌幅求和结果如果大于这个数，判断这个隐状态表示涨
75                self.up_state.append(i)
76            elif judge_result[i] < max_down:  # 隐状态的涨跌幅求和结果如果小于这个数，判断这个隐状态表示跌
77                self.down_state.append(i)
```

82-85 行：获得每个隐状态最可能出现的下一个隐状态

```
82        self.next_state = np.zeros((self.state_num))  # 每个状态所对应的下一个状态的列表，
83        for i in range(self.state_num):
84            self.next_state[i] = np.argmax(model.transmat_[i])
85        print('每个状态所对应的下一个状态：', self.next_state)
```

2.4 回测方法（87-200 行）

98-116 行：校验传入数据并做归一化处理，整合成二维矩阵

```
98            if len(xs) < 1:
99                print('error: 没有传入参数XS！！！')
100               return
101           x_len = len(pct_chg)
102           if len(close) != x_len:
103               print('pct_chg和close参数长度不相等！！！')
104               return
105           if len(mean) != x_len:
106               print('pct_chg和mean参数长度不相等！！！')
107               return
108           x_list = []
109           for x_temp in xs:
110               if len(x_temp) != x_len:
111                   print('error: XS参数长度不相等！！！')
112                   return
113               x_list.append(scale(x_temp))
114               self.data_num += 1
115
116           X_test = np.column_stack(x_list)  # 测试集
```

119-136 行：初始化统计的数据

```
119         correct_num = 0  # 预测正确的数量
120         base_money = close[self.sample_len - 1]  # 每天的金额，以第一天的前一天的收盘价为基础金额
121         base_money_fee = base_money  # 每天的金额，含手续费计算
122         base = 1  # 用来计算收益率
123         base_fee = base  # 用来计算含手续费的收益率
124         model_line = [base_money]  # 记录每天的金额的列表
125         model_line_fee = [base_money]  # 记录含手续费的每天的金额的列表
126         print('初始金额：', base_money, '，实际最终金额：', close[days_test-1])
127
128         buyed = 1  # 股票是否已经购买的状态
129         buy_num = 0  # 购买股票的天数
130         hold_num = 0  # 持有股票的天数
131         sell_num = 0  # 抛出股票的天数
132         empty_num = 0  # 空仓的天数
133
134         up_num = 0      # 预测涨正确的天数
135         down_num = 0    # 预测跌正确的天数
136         medium_num = 0  # 预测平正确的天数
```

138-180 行：根据模型的样本长度 sample_len，对当天的前 sample_len 天作隐状态序列的预测，根据序列的最后一个隐状态（即当天的前一个交易日的模型预测的隐状态）以及隐状态转移矩阵，获得当天的隐状态。155-178 行是根据预测的当天的隐状态，进行股票的买入卖出并计算收益。（第 55 行需要说明的是，如果是下跌的趋势，代码可修改为预测状态为涨买入，如果是上涨的趋势，代码可修改为预测状态不为跌，即涨或平，买入）

```
155         if (predict_state in self.up_state) and (not buyed):  # 预测结果不为跌 且 没有持有股票， 买入，手续费0.00032
156             buyed = 1
157             buy_num += 1
158             rate_temp = (mean[i] - close[i - 1]) / close[i - 1]  # 基于第二天股票均价相对于第一天收盘价的涨跌幅
159             base = base * (1 + rate_temp)
160             base_money = base_money * (1 + rate_temp)
161             base_fee = base_fee * (1 + rate_temp) * (1 - 0.00032)
162             base_money_fee = base_money_fee * (1 + rate_temp) * (1 - 0.00032)
163         elif (predict_state in self.down_state) and buyed:  # 预测结果为跌 且 持有股票，抛出，手续费0.00132
164             buyed = 0
165             sell_num += 1
166             rate_temp = (mean[i] - close[i - 1]) / close[i - 1]  # 基于第二天股票均价相对于第一天收盘价的涨跌幅
167             base = base * (1 + rate_temp)
168             base_money = base_money * (1 + rate_temp)
169             base_fee = base_fee * (1 + rate_temp) * (1 - 0.00132)
170             base_money_fee = base_money_fee * (1 + rate_temp) * (1 - 0.00132)
171         elif (predict_state not in self.down_state) and buyed:  # 预测结果为震荡或上涨 且 持有股票，不进行操作
172             hold_num += 1
173             base = base * (1 + pct_chg[i] / 100)
174             base_money = base_money * (1 + pct_chg[i] / 100)
175             base_fee = base_fee * (1 + pct_chg[i] / 100)
176             base_money_fee = base_money_fee * (1 + pct_chg[i] / 100)
177         else:  # 预测结果为跌 且 没有持有股票，不进行操作
178             empty_num += 1
```

194-200 行，绘制图表

```
194         ## 绘制曲线图
195         fig = plt.figure()
196         plt.plot(dates[self.sample_len - 1:], close[self.sample_len - 1:], color='green')
197         plt.plot(dates[self.sample_len - 1:], model_line, color='red')
198         plt.plot(dates[self.sample_len - 1:], model_line_fee, color='yellow')
199         plt.show()
200         fig.savefig("pictures\model_line.jpg")
```