Algorithmic-Trading Summary — Fall 2018
Alex Thomas
Colgate University

Algorithmic-Trading - Simple Moving Average - Application: Dual Moving Average

Introduction

Simple Moving Average (SMA) is an elementary AT strategy. Because of this, it is also the most commonly available tutorial online. The most common application of SMA is found in dual moving average, which utilizes two different averages that indicate momentum swings in the market.

Motivations and Measures

Simple moving average looks at a rolling average of a specified window. Mathematically this can be defined as: SMA(t)= $1/n \sum_{i=t-n}^{t} x(i)$ [] In words, this gets the average price over a specified window of time for a specified function. In application to the stock market and closing prices, this creates an average closing price for a specified amount of time, which gives an indicator of price swings in that period of time.

Key Techniques

The dual moving average technique is the most widely used techniques. It works by taking two different SMA's - a short window and a long window. In the implementation, a 40 day window and 200 day window were chosen. When the short window intercepts the 200 day window this gives an action signal. The short window crossing below the long window gives a buy signal reinforced by high trading volumes while the opposite is considered bearish and gives a sell signal.

Analysis

Effectiveness

Due to the nature of SMA, which generally looks at long periods of time to highlight trends, this can induce a lagged effect to the buy and sell orders. For example, in my implementation the short window looks over a period of 40 days while the long window is 200 days. For a meaningful change to register it requires multiple days of continued poor or good performance for a sizable change to register. Therefore, this can cause a lagged effect, which is what we see in our implementation, making this algorithm not very effective.

Runtime

The components to consider for this strategy include pulling stock data into a pandas data frame and calculating rolling means from the data-frame and marking their differences. For each calculation, it has to scrape through the entire data-frame at a rolling window, giving a linear runtime.

Quality Metric

TODO - backtesting - how to get stock splits?

Space / Memory implications

The only space for SMA required is the data-frame, which is very reasonable.

Conclusion

SMA is a simple AT strategy that looks at rolling window averages of closing prices for a particular stock. SMA is often applied in this dual moving average implementation where both short and long windows crossing generates buy and sell orders.

Implementation

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from algorithmictrading.stockData.getStock import stockDataRetriever

# LOOK AT EMA - SHOULD HELP WITH LAGGED PROBLEM
def execute(stock, start_date, end_date):
    stock = stockDataRetriever(stock, start_date, end_date).getStock()

    # Initialize the short and long windows and buy sell in df
    short_window = 40
    long_window = 100
    df = pd.DataFrame(index=stock.index)
    df['signal'] = 0.0

    # Create short and long simple moving average
    df['short_mavg'] = stock['Close'].rolling(window=short_window, min_periods=1, center=False)
    df['long_mavg'] = stock['Close'].rolling(window=long_window, min_periods=1, center=False).m

    # Create df
    df['signal'][short_window:] = np.where(df['short_mavg'][short_window:] > df['long_mavg'][sh

    # when signal changes from 1 to 0 or 0 to -1 - is a buy or sell
    df['positions'] = df['signal'].diff()

    ax1 = plt.figure().add_subplot(111,  ylabel='Price in $')

    # Plot the closing price, long and short moving average
    stock['Close'].plot(ax=ax1, color='r', lw=2.)
    df[['short_mavg', 'long_mavg']].plot(ax=ax1, lw=2.)

    # Plot the buy and sell df
    ax1.plot(df.loc[df.positions == 1.0].index,
             df.short_mavg[df.positions == 1.0],
```

```
            '^', markersize=10, color='m')
    ax1.plot(df.loc[df.positions == -1.0].index,
             df.short_mavg[df.positions == -1.0],
             'v', markersize=10, color='k')

    return plt
```

References