Analysis of the 2023-algothons

Comments from the SIG staff (this was at the end of the video but ngl this is 10x more useful)
People often assume that a time series is stationary even when it usually is not.

“Typically you want to look at returns rather than prices”

“Exploring dependencies between different instruments”

Team: Algorithmically Based FP:2nd

Strategy seems to revolve entirely around using the MACD indicator

presenter indicates that their algorithm is leverages the concepts used in MACD.

Team might of included additional helper indicators / strategies like calculating expected values etc.

Code Summary

You’re building a basic forecasting band for each instrument:
- Trend: Based on linear regression of progressive mean.
- Prediction: Next value estimated via linear extrapolation.
- Bounds: Loosely defined confidence interval based on slope magnitude.

```
# For each instrument, calculate the mean of all past values up to day i.
# Result: prog_mean[i][j] is the mean of instrument_i up to day j.
prog_mean = []
for instrument in prcSoFar:
    instrument_means = []
    for i, value in enumerate(instrument):
        if i < 1:
            instrument_means.append(value)
            continue
        instrument_means.append(np.mean(instrument[:i]))
    prog_mean.append(instrument_means)

# For each prog_mean, fit a linear trend line using np.polyfit.
# Each linear_fit = (slope, intercept) for that instrument.
linear_fits = []
if current_day <= starting_day:
    linear_fits = initial_fits
else:
    for i, indicator in enumerate(prog_mean):
        x = np.array(list(range(0, len(indicator))))
        y = np.array(indicator)
        slope, intercept = np.polyfit(x, y, 1)
        linear_fits.append((slope, intercept))

# Compute the expected value at the next time point using the linear model:
# EV = m · x + c
evs = []
for i in linear_fits:
```

```
    x = len(prcSoFar[0])
    m = i[0]
    c = i[1]
    expected_value = (m * x) + c
    evs.append(expected_value)

# Computes dynamic bounds around the expected value.
# Width of the bounds is proportional to the slope (linear_fits[i][0]) — effectively
allowing more "freedom" when the trend is steep.
uppers = []
lowers = []
for i, indicator_history in enumerate(prcSoFar):
    freedom_factor = 1
    freedom = abs(linear_fits[i][0] * freedom_factor)
    upper = evs[i] + freedom
    lower = evs[i] - freedom
    uppers.append(upper)
    lowers.append(lower)
```

Team: Bears, Bulls and Battlestar Galactica

Strategies tried out

- Fibonacci retractment (did not use)
- Exponential moving average (worked great on backtest, not so great)

Actual strategy

$$x = \frac{\text{price} - \mu_{\text{price}}}{\mu_{\text{price}}}$$

$$f(x) = \begin{cases} \text{buy if } x \text{ in top 2 percentile} \\ \text{short if } x \text{ in bottom 2 percentile} \\ \text{hold else all other cases} \end{cases}$$

Identify statstically unlikely prices, 2 percent is decided based on experimentation

my comment: I feel like this was pure luck

Incredible things they have done that we should do

- Have a better result analyzer. They have a PnL graph for each instrument.
- Get more data through data generators. Apparently they have more test data.

Team: Big Knees

SLSQP is some sorta optimization algorithm https://mdolab-pyoptsparse.readthedocs-hosted.com/en/latest/index.html

Model

1. Position initialization without commission (SLSQP)

(optimze score without considering comissions)

↓

2. Predict using ARIMA

(auto.arima, implements some algorithm to find optimial paramters)

↓

3. Refine prediction with comissions (SLSQP)

(optimze score considering comissions)

Team: CookieAlgorists FP:1st

Methods tried out and their results

1. **Paris trading**
2. **Moving average / Mean reversion**
3. **Simple linear regression (actually used)**

Key differnce, used a threshold for gradient in order to trigger a trade. It is not a predictive model of next price.

4. **State machines (actually used)**

Used in complenment with previous method to handle drawdown periods

5. **Multi linear regression (actually used)**

Linear regression prediction where past data from all 49 other instruments is used to predict the current instrument

Team: Deeptrade FP:3rd

That Haskell white paper.

Team: Los Algos Hermanos

The memers.

Fast, Medium Slow windowed EMA

Team: SVY

Correlation grouping and trading based on latest trend

```
look_back = 153
num_instruments = prices.shape[1]
currentPos = [0] * num_instruments
```

```
    for i in range(num_instruments):
        this_inst = prices[-look_back:, i]

        candidate_indices = []
        for j in range(num_instruments):
            if j == i:
                continue
            other_inst = prices[-look_back:, j]
            corr = np.corrcoef(this_inst, other_inst)[0, 1]
            if corr >= 0.95:
                candidate_indices.append(j)

        if not candidate_indices:
            continue

        for j in candidate_indices:
            price_today = prices[-1, j]
            price_yesterday = prices[-2, j]

            allocation = positionLimit // len(candidate_indices)

            if price_today > price_yesterday:
                currentPos[i] += allocation
            elif price_today < price_yesterday:
                currentPos[i] -= allocation

    return currentPos
```

Team: Team Q

Fourier transformed the data and used an trend following strategy.