Instructor: Mikhail Smirnov
Class: Discrete Time Models in Finance, Spring 2024

AI TRADING:

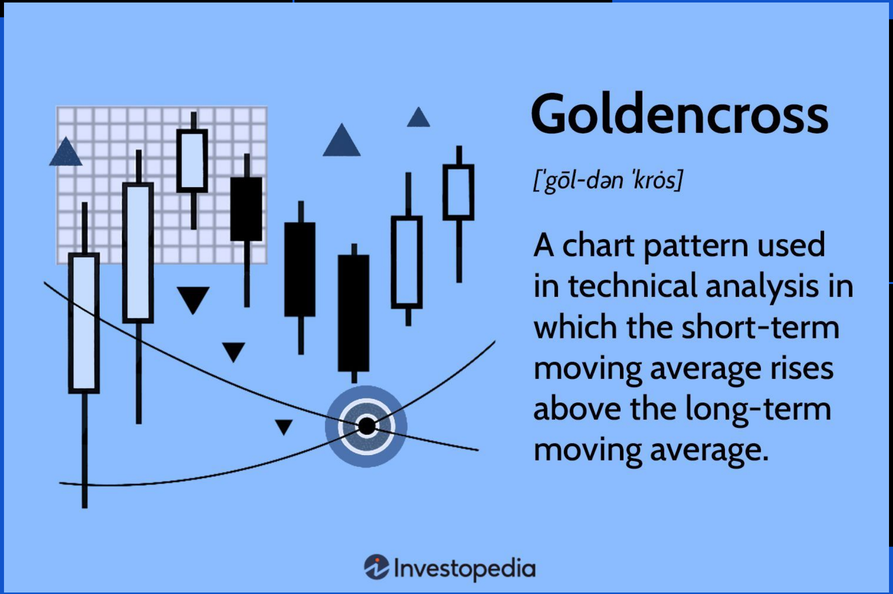
Optimizing moving average strategies

How can we use AI agents to make better trading decisions?

Brendan Fay, Icaro Bacelar, Yvon Lu

TABLE OF CONTENTS

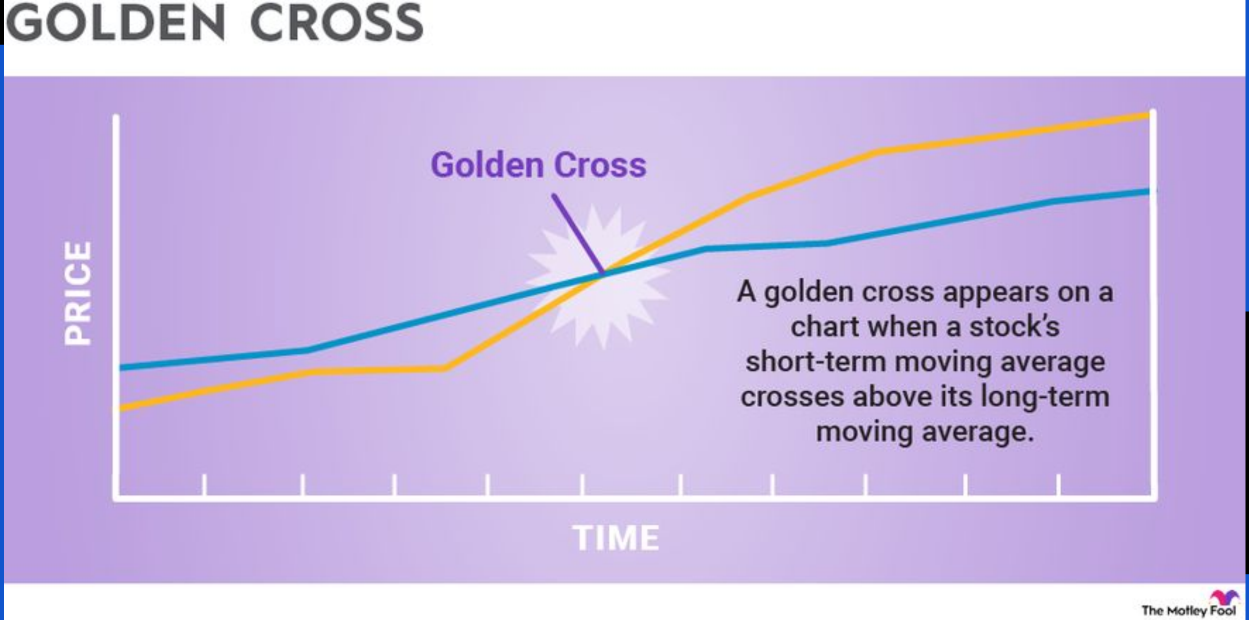
01 - THE PROBLEM



Moving average strategies - particularly a golden cross - are extremely popular trading strategies.

We want to explore how a trading agent can optimize a moving average strategy.

01 - THE PROBLEM

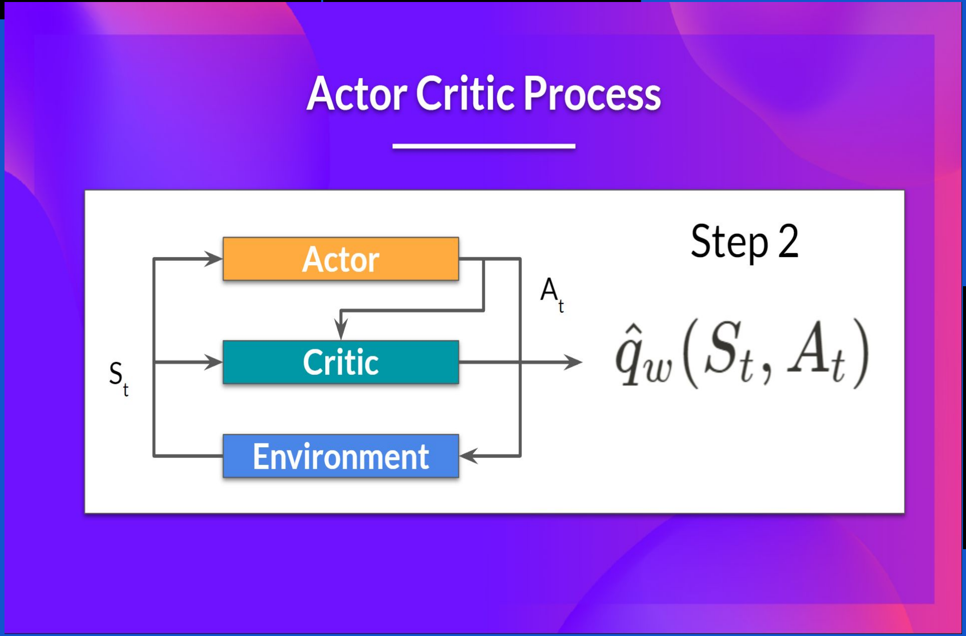


Short window - moving average over a short period of time

Long window - moving average over a long period of time

When the short-term average crosses the long-term average, that indicates a signal

01 - THE PROBLEM



Adversarial game between two AI agents with critical feedback after short periods

Pros: tight feedback loop, small number of parameters to optimize

Cons: highly stock specific, prone to premature convergence

01 - THE PROBLEM

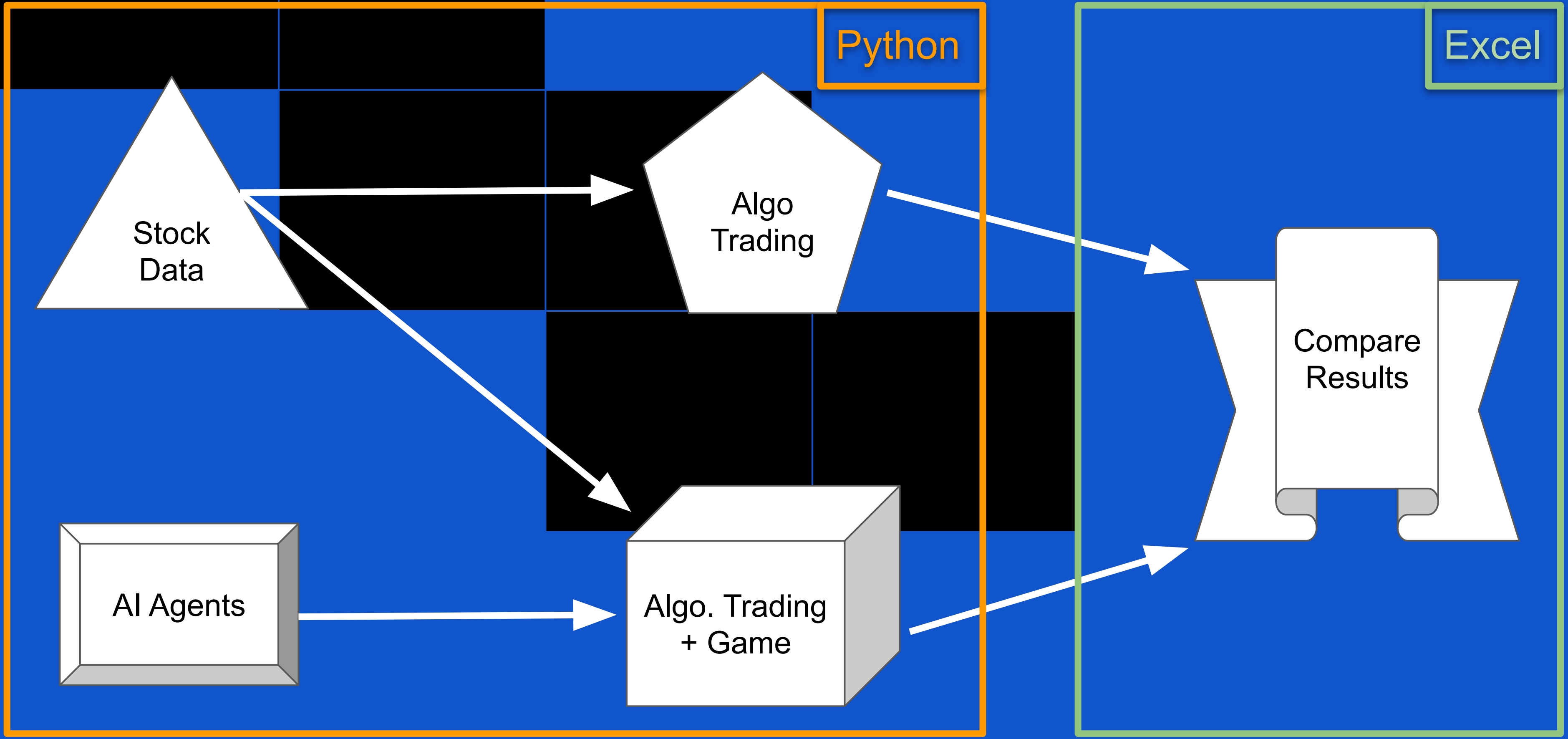
Highly stock specific → compare approaches of multiple stocks

Premature convergence → introduce randomness to preempt convergence

02

Methodology: Testing our hypothesis

02 - THE PROCESS



02 - GETTING DATA

- Using Yahoo Finance Python package
- Evaluating on ‘Close’ prices
- Appx 2600 days

```
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
import numpy as np
import random


# Choose the tickerand time window
ticker = "AAPL"
startD = "2015-01-01"

# Get historical market data for that ticker (Yahoo)
stock_data = yf.download(ticker, start=startD)
```

02 - TRADING WITH GOLDEN CROSS

- Use moving averages to establish Hold and exit signals

- Compute compound return based on periods wil non 0 signal

```
def calculate_strategies(start, end, results):
    for agent in agents:
        short_w = agents[agent]['short_window']
        long_w = agents[agent]['long_window']

        # Calculate Moving Averages using the entire dataset
        sma_short = stock_data['Close'].rolling(window=short_w, min_periods=1).mean()
        sma_long = stock_data['Close'].rolling(window=long_w, min_periods=1).mean()

        # Stabilish Hold or exit signals
        signals = np.where(sma_short.iloc[start:end+1] > sma_long.iloc[start:end+1], 1.0, 0.0)

        # Handle returns calculation to include the previous closing for accurate return calculation
        if start == 0:
            returns = stock_data['Close'].iloc[start:end+1].pct_change()
            strategy_returns = signals[1:] * returns.values[1:]  # Skip the first NaN
        else:
            returns = stock_data['Close'].iloc[start-1:end+1].pct_change()
            strategy_returns = signals[0:] * returns.values[1:]

        if start == 0:
            cumulative_returns = np.insert((1 + strategy_returns).cumprod(), 0, 1)
        else:
            previous_cumulative = results[agent][-1]
            cumulative_returns = (1 + strategy_returns).cumprod() * previous_cumulative

        results[agent].extend(cumulative_returns.tolist())
```

02 - THE AGENTS

```python
# Initialize agents with different windows
agents = {
    'A': {'short_window': 60, 'long_window': 240},
    'B': {'short_window': 40, 'long_window': 160}
}
```

- Initial windows chosen to be close to Golden Cross “ideal” Windows - 50 and 200
- Rounds of 10 days
    - Total of ~260 rounds

```python
# Store results in a dictionary
results = {'A': [], 'B': []}
original_results = {'A': [], 'B': []}

segment_length = 10  # Duration of each round

delta = len(stock_data)-round(len(stock_data)/segment_length)*segment_length

# Calculate original strategies without adjustment
calculate_strategies(0, len(stock_data) - 1-delta, original_results)


start = 0

while start + segment_length <= len(stock_data):
    end = start + segment_length - 1
    calculate_strategies(start, end, results)

    if not np.isnan(results['A'][-1]) and not np.isnan(results['B'][-1]):
        cumulative_A = results['A'][-1]
        cumulative_B = results['B'][-1]
        if start > 0:
            cumulative_A = results['A'][-1]/results['A'][-segment_length]
            cumulative_B = results['B'][-1]/results['B'][-segment_length]

        if cumulative_A > cumulative_B:
            adjust_windows('A', 'B')
        elif cumulative_A < cumulative_B:
            adjust_windows('B', 'A')



    start = end + 1
```

02 - AI STRATEGIES

- Heuristics:
  - Loser closes gap
  - Avoid converging
    - Random change
  - Keep short:long ratio 1:4
  - Short window > 1 month

- Ideally needs tuning for each stock

```python
def adjust_windows(winner, loser):
  for key in ['short_window', 'long_window']:
      if agents[winner][key] > agents[loser][key]:
          agents[loser][key] += round(agents[loser][key]/5)
      elif agents[winner][key] < agents[loser][key]:
          agents[loser][key] -= round(agents[loser][key]/5)
      else:
          # If the window sizes are the same, randomly increase or decrease by 20%
          if random.choice([True, False]):  # Randomly choose True or False
              agents[loser][key] += max(round(agents[loser][key] * 0.03),1)  #3% change
          else:
              agents[loser][key] -= max(round(agents[loser][key] * 0.03),1)

      if key ==  'short_window' and agents[loser][key] < 35:
          agents[loser][key] = 35
      if key ==  'short_window' and agents[loser][key] > 65:
          agents[loser][key] = 65

      if key == 'long_window' and agents[loser][key] < 140:
          agents[loser][key] = 140
      if key == 'long_window' and agents[loser][key] > 260:
          agents[loser][key] = 260
```

03

Results: How did our Agents perform?

03 - HOW WE OBTAINED RESULTS

- Simulated 50 runs for 3 tickers
  - Due to stochastic strategies, each simulation has a different result
  - High variance
- Appx. 260 rounds per simulation

03 - CUMULATIVE RETURNS: TSLA

Agent A	Agent A	Agent A
6.937622347	7.692675646	9.795010738
12.62967582	6.010273225	8.344744659
5.892683984	4.887781963	9.08119518
7.499429383	7.576435802	4.473409591
9.861518089	11.62238788	8.589368281
6.59624199	10.09743677	10.31736863
6.969330576	6.864884042	12.28357149
8.976139332	6.351333194	6.930128101
6.778179777	9.570014768	7.499429383
9.560055537	7.734019934	16.96452901
8.603296037	5.544759829	5.303145477
4.785507325	9.349205333	7.659451738
10.26815927	3.771557618	5.473219169
4.404318013	6.955239273	4.600883483
10.19352776	5.416984046	4.488772739
10.02694239	6.655550508	5.953793729
6.098504211	10.21383278	

Agent B	Agent B	Agent B
3.512398801	2.862561455	6.268808746
7.001095755	6.875418747	5.584129045
4.081752486	9.37216568	9.371384255
4.282521829	3.397996316	4.595632451
4.246432779	4.348806045	4.675736627
5.861816387	5.692944689	3.01541522
4.313594215	3.883971999	7.192369286
5.890959442	4.985571261	9.219653871
3.642429195	5.865347948	9.809285912
6.330022723	5.35273249	10.07888127
5.019470817	6.028750654	4.282521829
5.264133842	5.084743532	8.488936435
7.301424089	9.137171011	5.500238956
3.657581811	4.082030275	4.953224257
4.781677467	7.920524042	4.283415899
10.0760516	5.701823649	4.559337214
4.736721117	4.030958072	

Original A	4.3780325
Original B	5.145517854
Ideal: 50 - 200	5.438144341

03 - CUMULATIVE RETURNS: AMZN

Agent A	Agent A	Agent A
8.498578838	6.800617686	6.270316298
5.846542324	6.892664429	6.382140693
9.261489106	10.30294799	8.540704841
9.20090982	7.726911223	5.681466069
7.35216597	7.358186496	7.473906067
7.225242643	8.376101189	7.216196072
10.13411302	6.978566523	5.193225501
5.91829252	7.244164088	5.738219275
6.635082006	5.443021138	8.875168741
9.463095937	8.956881397	5.765832957
6.856573402	8.329641982	8.613040499
4.345249389	6.42185538	9.20090982
7.261400871	5.023726395	7.304117263
8.299255357	9.362330705	6.742078023
9.38144584	7.930201106	6.806733072
5.176837564	5.933426918	7.673384146
8.142502086	9.054304934	

Agent B	Agent B	Agent B
7.998587889	5.896128673	6.346837588
4.559402435	6.583736137	7.292376463
6.576428081	8.54627338	8.815814233
8.816716933	7.757170756	5.403035295
9.100949756	6.969174979	6.473747035
7.349138513	7.631196171	7.78376258
8.240349125	7.088036612	4.842890294
6.160061053	6.73035677	5.321570164
6.768835733	6.007176704	8.3847655
8.863500005	8.111058835	6.877823333
7.126141741	8.807651883	7.595856114
5.068920248	5.877834851	8.816716933
5.735725493	5.392081492	6.16779085
8.977982327	8.983414595	6.854808601
8.595574286	5.841398118	6.497947196
5.698521787	6.795606961	6.959392683
8.281182254	7.911681266	

Original A	7.375799124
Original B	7.278906273
Ideal: 50 - 200	7.726508486

03 - CUMULATIVE RETURNS: AAPL

Agent A	Agent A	Agent A	Agent B	Agent B	Agent B
2.542591912	2.855293118	2.66457935	2.651894468	2.769212885	3.962569014
3.884614802	3.522382707	3.615951046	3.575154869	3.872337589	4.459747734
2.99011752	3.03472913	3.573668924	3.212843796	2.730058317	2.686678249
2.480014966	3.365329493	2.678782926	3.275974248	4.398203513	3.27850378
2.708581219	3.962538212	2.8543236	3.459826735	2.720194262	2.647542831
2.466732356	2.243171095	3.631012253	3.78945745	2.5697293	2.910401208
2.42367906	2.109874597	2.365751245	4.298665979	3.298462311	2.563100041
3.733449009	2.721386474	2.729148804	3.043498052	3.810891063	3.230890969
2.809999641	3.18419252	2.903047249	3.318912208	3.38647157	2.601361607
2.374743833	2.603581865	2.386404143	2.694291157	3.056396926	3.608751002
2.656083335	3.601063043	2.912406832	1.999875141	3.62422513	2.61488377
2.534474268	2.63376728	2.723079824	2.390155246	3.348739182	3.845548644
2.431833237	2.132979046	2.326538221	3.003903621	4.505444808	3.763745639
2.999244062	2.763240363	2.674774757	3.295299496	2.287802238	3.514041617
3.05671832	2.700273525	2.748017391	3.514041617	2.725050037	2.551996855
4.047398505	2.86164723	3.296304699	3.304513495	2.511616151	2.906319569
3.567534764	3.775535687		2.24613162	2.731514176	

Original A	3.044689304
Original B	3.000271409
Ideal: 50 - 200	4.182353518

04 **Analysis:** Did our Agents outperform the original strategy?

04 - HYPOTHESIS

$$H_0: \frac{Return\ w/\ Agents}{Return\ w/out\ Agents} - 1 > 0$$

- Will test against:
    - Agents' original windows
    - Golden Cross Ideal windows: 50 and 200

04 - ANALYSIS OF RESULTS: TSLA

	Adapt:Original
Ratio of Retuns	1.421016479
Std. Dev	0.415260984
Z-Score	**7.169072426**

	Adapt:Ideal
Ratio of Retuns	1.244277564
Std. Dev	4.750390346
Z-Score	**4.750390346**

- Statistically significant improvement

04 - ANALYSIS OF RESULTS: AMZN

	Adapt:Original
Ratio of Retuns	1.147656954
Std. Dev	0.1269334749
Z-Score	**8.225508155**

	Adapt:Ideal
Ratio of Retuns	1.088368348
Std. Dev	0.12037602
Z-Score	**5.190889196**

- Statistically significant improvement

04 - ANALYSIS OF RESULTS: AAPL

	Adapt:Original
Ratio of Retuns	1.007217327
Std. Dev	0.1293699343
Z-Score	**0.3944827792**

- Improvement is NOT statistically significant
    - Heuristics need to be adjusted for each “game”

APPLICATIONS/FUTURE:

- More optimized trading algorithms
  - High frequency / Self correcting
- In future:
  - Can adapt for different stocks
  - Complete backtesting
  - More than 2 AI agents
  - Multiple parts of the same algorithm

CONCLUSION:

- ➔ Successfully developed a trading strategy with 2 competing AI agents implementing Golden crossover
  - ◆ Tested our method with 3 Stocks
- ➔ Found statistically significant improvement in using our strategy

Thank you! Any questions?