What's so special about predictions of Stock time series?

- A hard problem! Is it even possible?
- Looks very much like random walk!
- The process is "regime shifting". The markets move in and out of periods of "turbulence", "hause" and "baise". Hard for traditional algorithms!
- The evaluation of predictability is extremely hard! When have we learned and when have we memorised?
- A successful prediction algorithm does not have to give predictions for all points in the time series. Can we predict predictability?

What does the data look like?

Clear Trending behaviour in two time series

Technical analysis: Triangles

Clear Trending behaviour in two time series

The efficient market hypothesis
The prices reflect ALL available information and new information is assimilated immediately.
Implies a random walk. "Impossible to predict!"

Traders viewpoints
"Just a question of hard work and good intuition!"
The market clearly goes through periods of positive and negative trends. It's just to identify the peaks and the troughs
Data in Technical analysis

- Close price
- Highest paid during day
- Lowest paid during day
- Volume (no. of traded stocks)

"tick" data sometimes available

Data in Fundamental analysis

1) The general economy
   - Inflation
   - Interest rates
   - Trade balance etc.

2) The condition of the industry
   - Other stock's prices, normally presented as indexes.
   - The prices of related commodities such as oil, metal prices and currencies
   - The value on competitors stocks

3) The condition of the company
   - P/E: Stock price divided by last 12 months earning per share
   - Book value per share: Net assets (assets minus liabilities) divided by total number of shares
   - Net profit margin: Net income divided by total sales
   - Debt ratio: Liabilities divided by total assets
   - Prognoses of future profits
   - Prognoses of future sales
**Derived entities**

- **k-day Returns:**
  \[ R_k(t) = \frac{y(t) - y(t-k)}{y(t-k)} = \log \left( \frac{y(t)}{y(t-k)} \right) \]

- **Moving average of order \( k \):**
  \[ \text{ma}_k(y) = (z(1), z(2), ..., z(N)) \]
  \[ z(t) = \frac{1}{K} \sum_{i=1}^{K} y(t-i) \]

---

**Inductive Learning**

**Given:** A set of \( N \) examples \( \{(x_i, z_i), i=1,N\} \) and an unknown function \( f \) such that \( f(x_i) = z_i \) \( \forall i \)

The task of **pure inductive inference** or **induction** is:

Learn a function \( g \) that minimises the norm of the error vector \( \text{E} = [e_1,...,e_n] \)

where \( e_i = e(g(x_i), z_i) \)

I.e.: \( g \) should "approximate" \( f \)

**Note:**

The error function \( e \) and the norm \( |E| \) are still not defined.

---

**1) Standard Time series approach**

Inputs: \( X(t) = (y(t), ..., y(t-k+1)) \)

Output: \( z(t+h) = y(t+h) \) where \( h \) is the prediction horizon

I.e Predict future prices with past prices

\[ e_i = g(X(t)) - z(t+h) \]

\[ E_i = \frac{1}{N} \sum_{i=1}^{N} e_i^2 \] (RMSE)

Typical choices of function \( g \):

- \( g(t) = \sum_{i=1}^{k} a_i y(t-i) \) **AR-model**
- \( g \) is a general non linear function **Neural network**

---

**Feed-forward neural network**

- **Input layer with 4 inputs**
- **Two Hidden layers with 3 and 5 nodes**
- **Output layer with 1 output node**

The weights \( w \) are selected to minimise

\[ E_i = \frac{1}{N} \sum_{i=1}^{N} (g_i(t) - y(t+1))^2 \]
1) Standard Time series approach

Drawbacks:
- A stationary model is not realistic
- Fixed horizon not realistic. A profit 2 days ahead is as good as 1 day ahead.
- The MSE measure treats all predictions $g$ as large as equal.

2) Pattern classification approach

Inputs: $X(t) = (R_1(t), R_2(t), R_{10}(t), R_{20}(t))$
Output: $z(t+5) = R_1(t+5)$

$e_t = \begin{cases} 
1 & g(X(t)) > \alpha \text{ AND } z(t) < 0 \\
1 & g(X(t)) < -\alpha \text{ AND } z(t) > 0 \\
0 & \text{otherwise}
\end{cases}$

$\alpha$ is a trading threshold $\geq 0$

$E = \frac{1}{N} \sum_{i=1}^{N} e_i^2$

$g$ can be used in a trading rule $T$:

$T(t) = \begin{cases} 
\text{buy} & \text{if } g(X(t)) > \alpha \\
\text{sell} & \text{if } g(X(t)) < -\alpha \\
\text{do nothing} & \text{otherwise}
\end{cases}$

Feed forward neural network

- Input layer with 4 inputs
- Two Hidden layers with 3 and 5 nodes
- Output layer with 1 output node

The weights $w$ are selected to minimise:

$|E| = \frac{1}{N-24} \sum_{i=24}^{N} e_i^2$

Recurrent neural network

- Feedback to input layer
- The hidden layer stores previous values and can reconstruct the dynamics

The weights $w$ are selected to minimise:

$E = \frac{1}{N} \sum_{i=1}^{N} (g_u(t) - y(t+f))^2$

Technical Indicators

- The tools for Technical trading
- Include principles such as:
  - The trending nature of prices
  - Volume mirroring changes in price
  - Support/Resistance
- Examples:
  - Moving averages
  - Formations such as triangles
  - RSI - the relation between the average upward price change and the average downward price change within a time window normally 14 days backwards

Technical Indicators

- Can often be described as a trading rule:

$T(t) = \begin{cases} 
\text{buy} & \text{if } g(\text{X}(t)) > \alpha \\
\text{sell} & \text{if } g(\text{X}(t)) < -\alpha \\
\text{do nothing} & \text{otherwise}
\end{cases}$

where $X(t) = (y(t),...y(t-k+1))$

- Example:

$mav_u(y) = (z(1), z(2),..., z(N))$

$z(t) = \frac{1}{k} \sum_{i=1}^{k} y(t-i)$

$g = \Delta(\text{sign(mav_u(y)) - mav_d(y))}$

$\Delta v(t) = v(t) - v(t-1)$
Benchmarks

- Naive prediction of stock prices:
  \[ y'(t) = y(t-1) \]

- Naive prediction of returns:
  \[ R'(t) = R(t-1) \]

The naive predictors are local minimum in many models e.g AR-models (but also Neural Networks):

\[ y'(t) = \sum_{i=1}^{N} a_i \cdot y(t-i) \]

- Buy and hold:
  Buy at day 1 and sell at day N

Performance measures

- Theil coefficient:
  Compares the RMSE (root mean square error) for our predictions with the naive price predictions

  \[ T = \frac{\sum_{t=1}^{N} (y(t) - y'(t))^2}{\sum_{t=1}^{N} (y(t) - y(t-1))^2} \]

  \( T < 1 \) for real predictive power

- Directional prediction “Hit rate”

  Predicting \( [R(t), t=1,N] \) with \( [R'(t), t=1,N] \)

  \[ H = \begin{cases} 1 & |R(t)| = 0, t = 1, N \\ \{ t | R(t)R'(t) > 0, t = 1, N \} & \{ t | R(t)R'(t) = 0, t = 1, N \} \end{cases} \]

  For the naive return predictor:

  \[ H_n = \begin{cases} 1 & |R(t)| = 0, t = 1, N \\ \{ t | R(t)R'(t) > 0, t = 1, N \} & \{ t | R(t)R'(t) = 0, t = 1, N \} \end{cases} \]

  Normalised hit rate:

  \[ H_s = \frac{H}{H_n} \quad H_s < 1 \text{ for real predictive power} \]

Relative stock performance

- Portfolio management
  - Minimise the variance in a portfolio by quadratic programming
  - Also possible with single stock methods by:
    \[ y^1(t) = y(t) \sum_{i=1}^{N} y_i(t) / K \]

- Ranking stock returns:
  The stock with highest \( R \) gets rank 1:

  \[ \text{Rank}_t(y) = 1 - \left[ \frac{R(t)}{R(t)|R(i)(t) > R_i(0), i = 1..N} \right] \]

Performance measures

- Mean profit per trade:
  - Trading rule approach:
    - “Run” the trading and compute the mean profit
  - Time series approach:

  \[ \text{Mean profit} = \sum_{t=1}^{N} \text{sign}( y'(t) \cdot y(t-1) ) \cdot (y(t) - y(t-1)) / N \]

  I.e:
  A trade is assumed at every time step, in the direction of the predicted change.
Evaluating performance

**What is a reasonable goal?**

- Efficient market hypothesis implies random walk which is impossible to predict.
- The ACF has very low values.
- Nearest neighbour analysis shows very low correlation.
- There are so few $100 notes laying around!
- Published research (with proper evaluation) often shows about 54% hit rate.
- Even 54% real hit rate is enough to make a fortune!
- Compare with a casino: They don’t know what number comes up next, they just improve the odds by adding the 0 and 00.

© Thomas Hellström 1997

Evaluating performance

**We are predicting a stock with equal numbers of moves up and down during one year of 250 trading days.**

**Apply a totally random prediction algorithm on each day.**

**What is the probability that the hit rate>54%?**

The distribution for number of hits is given by:

\[ P(H = x) = \begin{cases} \frac{250}{x} & \text{for } 0 \leq x \leq 135 \\ 0.5^x & \text{for } x > 135 \end{cases} \]

\[ P(H > x) = 1 - P(H \leq x) = 1 - \text{binomcdf}(x, 250, 0.5) \]

\[ x = 0.5^250 \times 135 \text{ gives } P(H > 135) = 0.092 \]

I.e. There is a 9% risk that a random algorithm gives 54% hit rate.

© Thomas Hellström 1997

Evaluating performance

**We want to compare 100 indicators that each produce Sell and Buy signals on average once a week. The test period is 10 years! We demand 55% hit rate!**

**Apply 100 totally random prediction algorithm on each week.**

**The probability that any one of them gets exactly \( x \) hits is:**

\[ P(H > 0.55 \times 500) = 0.0112 \]

The probability that ANY of the 100 indicators produce 55% hit rate is 1-minus the probability that all are less then 55%:

\[ 1 - (1 - 0.0112)^{100} = 0.68 \]

How do we know when we have learned?

Algorithm evaluation is a part of the learning process!

- It must be done “in sample” and not on the test set.
- Best: A final test on data that didn’t exist at the time of the development of the algorithm.
- It is sensitive to “over training”.
- Be aware of the data-snooping problem!

© Thomas Hellström 1997

Results so far

**Used methods**

- Artificial Neural Networks
- Fuzzy rule bases
- State space reconstruction and local models
- \( k \) nearest neighbour techniques
- Adaptive AR
- Hundreds of technical indicators

**Results:**

- No statistically significant predictions
- Significant seasonal patterns in data

Future work

- Finding regions with predictability
- How do we know that we have learned?
- Fundamental analysis much easier?
  - Problem: lack of huge amounts of data
- Other methods