High Frequency Trading Strategy Based on Prefix Trees

Yijia Zhou, 05592862, Financial Mathematics, Stanford University

December 11, 2010

1 Introduction

1.1 Goal

I am an M.S. Finanical Mathematics student pursuing a career in professionl trading; therefore I wish to leverage what I learned in the class to the development of a high frequency trading strategy based on pattern recognition. Specifically, I wish to achieve two goals:

1. The model should successfully predict future returns given historical data
2. The model should lead to a consistently profitable trading strategy with low risk

1.2 Challenge

High-frequency financial data (min-by-min) is known to be able to pass most martingale tests.

$$E\left[X_{t+s}|\mathcal{F}_t\right] = X_t$$

In plain English, it means given historical information, the best prediction of the future is the current state. That is, the data usually has no predictive value at all in conventional methods. A special technique must be developed to extract information from financial time series data.

Another special challenge in high-frequency trading is that the model must be able to make trading decisions and update itself quickly to keep pace with the high-frequency streaming data. Therefore the algorithm complexity is constrained.

2 Unsuccessful Conventional Regression/Classification Methods

Conventional regression/classification methods work poorly. This is not surprising because of the nature of financial data.

2.1 Regression: Time Series Model

Apply ARMA(p,q) to historical data. Use AIC to determine optimal p, q:

$$X_t = c + \epsilon_t + \sum_{i=1}^{p} \phi_p X_{t-p} + \sum_{i=1}^{q} \theta_q \epsilon_{t-q}$$

Failure: Model diagnostics are poor; typical significance level (out of different training set/testing set combinations)=0.6210

Reason: The high-frequency data is known to pass martingale tests. It is improbable that we can extract information in time-series methodology.

2.2 Classification: Multinomial Event Model

Discretize return into alphabet {-1,0,1}:

$$z_i = \begin{cases} -1 & r_i > a \\ 0 & a \geq r_i \geq -b \\ 1 & r_i < -b \end{cases}$$

(a, b are decision variables) Use a sliding window of m minutes,

$$X = \begin{bmatrix} 1 \\ 0 \\ 0 \\ ... \\ -1 \end{bmatrix}$$

The $(m+1)$th element is seen as the class label.

Failure: I tried 10/20/30-min as sliding window length, but the average probability of prediction being right is only 31%. There is no difference to rolling a die.

Reason: The conditional independence assumption

$$P(x_1, ..., x_n|y) = \prod_{i=1}^{n} P(x_i|y)$$

is not valid beucase in terms of financial data, because the precedence of events happening matters a lot in financial data (that is exactly the time series pattern I am looking for):

$$z_1 = [111001] \text{ and } z_2 = [101011]$$

though both having 4 1's and 2 0's, are very different from each other in time series.

3 Methodology: Theoretical Framework

3.1 Leaning: Tree Construction

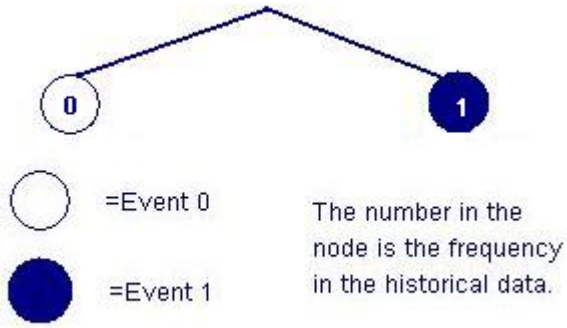
Discretize returns into {-1,0,1} first. For the purpose of demonstration, I use a binary tree here. Suppose I observe a sequence

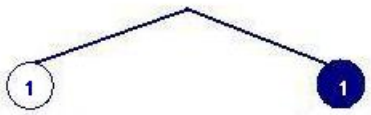
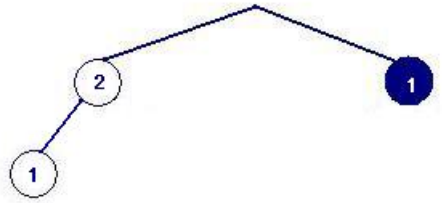
$$z = \{1000111100111011001011100101011101\}$$

It is parsed sequentially into a series of patterns that have not been observed before. For example, $z_1 = \{1\}$, $z_2 = \{0\}$, $z_3 = \{00\}$ but not $\{0\}$ because it has been observed at $z_2 = \{0\}$. Similarly $z_4 = \{11\}$ but not $\{1\}$. Thus z is parsed to

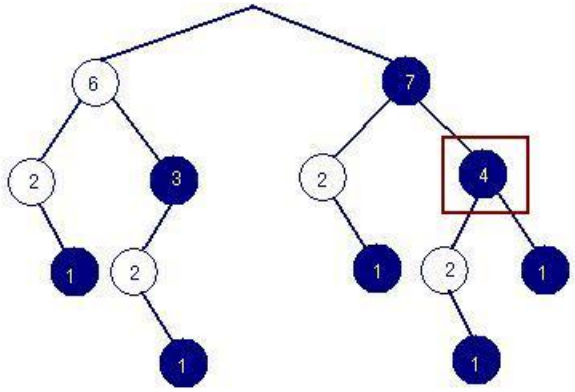
$$z = \{1\}; \{0\}; \{00\}; \{11\}; \{110\}; \{01\}; \{1101\}; \{10\}; \{010\}; \{111\}; \{001\}; \{0101\}; \{101\};$$

These are the basis patterns. Then I encode the data in a binary tree; the left child being event 0, the right child being event 1; the node value being the number of occurence. Each time we observe a new pattern, the values of all nodes on the path grow by 1. For example, after updating the tree by $z_1 = \{1\}$, I get



update after	$z_2 = \{0\}$	$z_3 = \{00\}$
tree		

Do this recursively and I get the fully-grown tree for the sequence z,



Note that the value of each nodes equals the sum of its children plus 1. In implementation parsing (which can take advantage of the tree to decide whether a pattern has appeared already) and updating are interweaved. The tree is developed and used in a Bayesian mindset, and that is why I call it "tree of conditional frequencies".

3.2 Prediction

After the model has finished learning from historical data, we then need to use it in prediction and trading. With no real-time streaming market data yet, our best guess of what happens next is from the perspective of the root looking at its children. Now if I receive the streaming data $z' = \{1\}$, the best estimation is then from the perspective of the right node. The square block in the tree graph above indicates the scenario of $z' = \{11\}$. By Laplace Smoothing the estimation of the return in the next period is

$$P(z'_3 = 0|\{11\}) = \frac{3}{5},\ P(z'_3 = 1|\{11\}) = \frac{2}{5}$$

3.3 Trading

Prediction alone does not suffice; trading style matters, too. I need to optimize the trading policy π that works best with the prediction model. Think this way: a more aggressive trader may trade as long as one outcome is more probable than the other, to fully take advantage of the Law of Large Numbers. However, this may lead to increased trading costs, and possible losses in a streak (thus larger risk).

A more defensive trader initiates trades only when she is confident enough of the probability of one outcome happening; therefore with each trade profitability is more probable. In this study I specify two trading styles following the rules below:

- Aggressive Trading
  - If I already have a long position, sell only if the most probable outcome is {-1}. Vice versa.
  - If I have no position at hand, initiate trade if the most probable outcome is in {1,-1}.
- Defensive Trading
  - Change the criterion for initiating/holding long positions to $P(1|\mathcal{F}_t) - P(0\, or\, -1|\mathcal{F}_t) > \epsilon$

I estimate ϵ and discretizatio parameters a, b using cross-validation. In a base case I assume $a = b$. My historical data are split evenly to 8 sections for cross-validation.

3.4 Model Evaluation

Two criteria are useful in evaluating how the model summarizes the data. Define

$$\text{Compression Ratio} = \frac{length(\text{source})}{length(\text{encoding})}$$

$$\text{Internal Node Ratio} = \frac{\#(\text{internal nodes})}{\#(\text{leaves})}$$

Compression ratio measures the randomness of the historical financial data. The larger compression ratio, the less random the data are. Internal node ratio measures the diversity of source patterns–The model is more predictive if there are fewer frequent patterns in the data.

4 Empirical Results

4.1 Data and Trading Assumptions

To make the study as representitative as possible, I retrived the 2-min/5-min/10-min market data of JPY/USD (foreign exchange, or FX), S&P 500 (equity index) and IBM (equity).

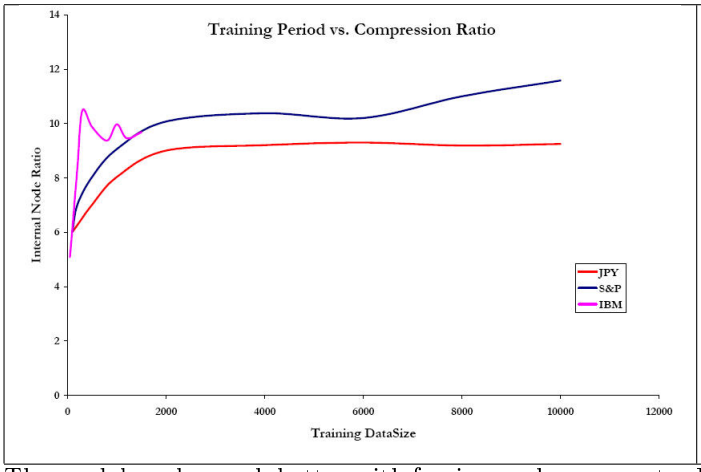
I assume a trading environment for institutional traders. That is, 400:1 leverge (which magnifies the profit and loss by 400 times) is available in FX trading and 20:1 leverage is available in equities. The in-and-out trading cost, including commissions and spreads, is 2 bps (1 bps=0.01%) for FX and 0.2 cents for equities.

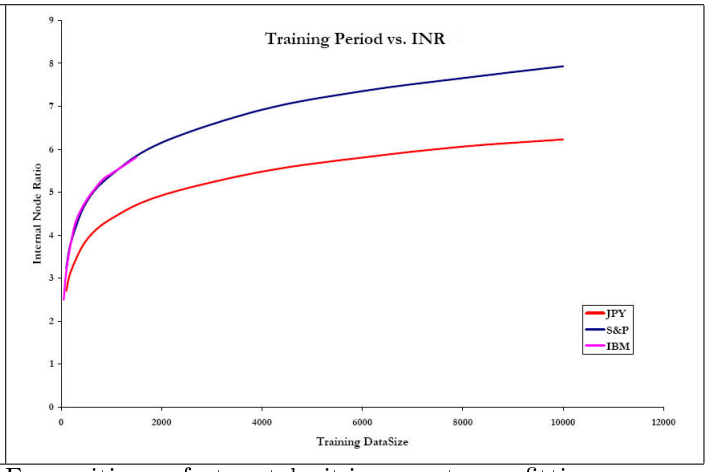
4.2 Results and Interpretation

Due to the limit of pages I show the graphs for aggressive trading style only. The defensive style results will be summarized in a table later.

First, the compression ratio is steady against the training data size. This suggests the entropy/predictibility of financial data does not vary.

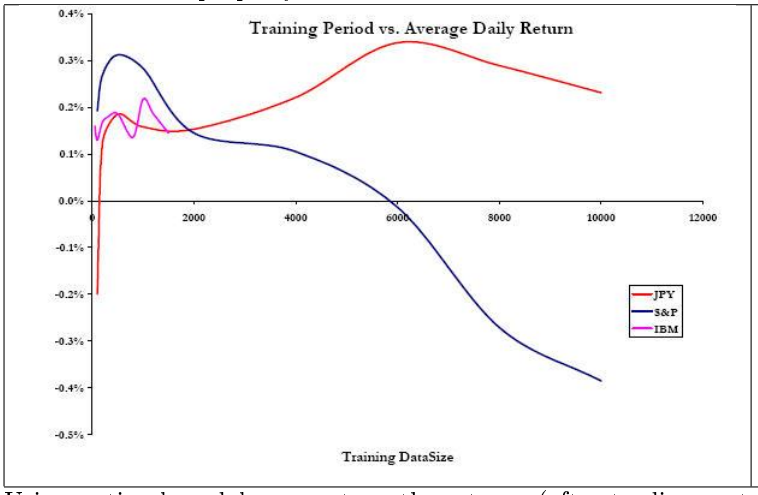
The internal node ratio grows steadily in a $\log n$ fashion, due to the nature of trees.

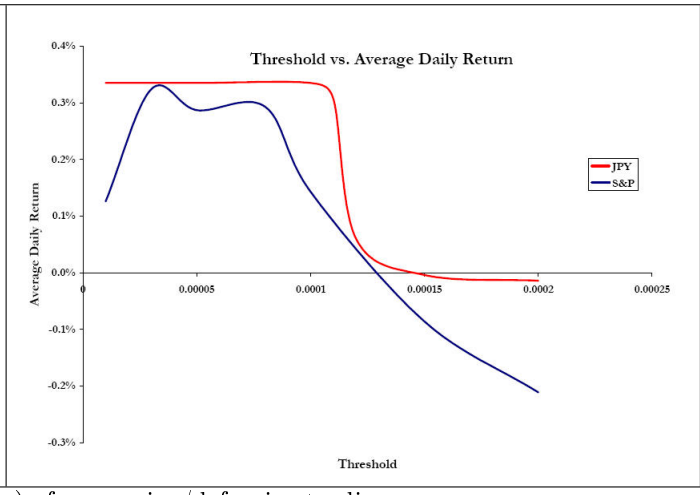




The model works much better with foreign exchange assets. For equities, unfortunately, it is prone to overfitting.

Before trading return dropping dramatically, the daily return is insensitive to discretization parameters a, b. This is an excellent property.





Using optimal model parameters, the returns (after trading costs) of aggressive/defensive trading are

Daily Return	Aggressive	Defensive
JPY/USD	0.21%	0.23%
S&P 500	0.01%	0.10%
IBM	-0.10%	0.01%

It is obvious that FX still works the best, while defensive trading can largely enhance the performance of equities trading.

5 Conclusions

- Kraft Inequality says the expected codeword length must be greater than or equal to the entropy of the encoded source. That is, the (lossless) compression ratio cannot be arbitrarily large. An asymptotically optimal encoding technique is Huffman Code, which movitates this method.
- Market price data are clearly compressible, which implies that there are some "preferred paths", that is, the ternary tree is not uniformly even – some parts are "bushier" than others. This suggests that the chain of event state is not completely "memoryless" – the development of the next event state is "path-dependent".
- Currency data has the nice "high frequency" feature.
- Properly chosen trading strategies might earn decent return.
- The training phase complexity is $O(n \log n)$. The trading phase complexity is $O(1)$ for each new streaming data point. This is excellent because the algorithm can run very fast so as not to delay making trading decisions in a high frequency setting.