Algorithmic Trading
Predicting the S&P 500 with Hidden Markov Models

Siddharth Agarwal

Abstract

This paper aims to predict the S&P 500 on regular trading days using trends obtained by training a Hidden Markov Model (HMM) using commonly available market data. Daily market information is fed into this model with a window that shifts forward each day. Output probabilities are calculated by the HMM for the chance that the market is in a given state for the next day. This probability is passed on to an algorithm that interprets it to create a signal for the next day and make trades in the open market.

Introduction

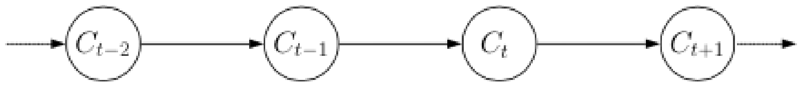
Financial market structure today has come a long way from the days of yelling bid/ask prices in a pit. With a huge movement towards electronic trading, trades are able to be submitted and matched in microseconds, creating a whole new sub-industry for high-frequency trading. A type of technology that goes hand in hand with this is algorithmic trading. What previously used to be an algorithm in a trader's mind is now programmed into a computer to make trades autonomously and generate profits for the firm running the program. These systems are now widely available for the day trader through broker APIs. A resourceful programmer can download the API and quickly code his/her program in a language of their choice; usually Java, C, or Python. However, barriers to entry in this field are usually the data sources, which can be expensive, and the access to low latency systems that submit the trades to the exchange used. Without these systems, it is sometimes difficult to produce profitable algorithms using only publicly available data and retail brokers systems. This project aims to use commonly available market data from Yahoo Finance and use it to try and predict the behavior of the S&P 500 on the next market day using R, a free language/software. The S&P 500 is the most common market index used in United States, and provides a low volatility model as well as a broad picture of the entire market. The asset being traded is the E-Mini S&P 500 future, a popular asset which tracks the behavior of the S&P 500 index.

Discussion

Along with algorithmic trading, machine learning is becoming more and more common to train a given system and have it predict future market movements from past behavior. One machine learning system is a Hidden Markov Model. Other options include random forests and other binary classification models. One of the motivations for choosing a Hidden Markov Model (HMM) to model the market is that it fits very neatly into the traditional 'bull/bear' market classification paradigm that is so popular. A bull market is a period of time when the market trends upwards, while a bear market is a time period when it is trending downwards. While their definitions are not very quantitatively oriented, they are popular terms in the general media. With a Hidden Markov Model, similar to a Markov Chain, it consists of 'states' or 'regimes' that change with every shift in time period t. Traditionally used for pattern recognition in the sciences, they have very recently become more popular outside of speech and handwriting recognition, being applied to financial time series data in other recent papers.

In this paper, two sorts of trades are used: long and short. Being 'long' an asset means that the trader owns the asset, and gains and losses in the assets price translate into the same amount of gains and losses of the capital c used, respectively. If the price p_1 is the price at which the asset is bought, and p_2 is the price at which it is sold, then the profit on this position is p_2-p_1. This is important because it is possible to have 'leverage' on the position, which means that either a financial derivative is being used, or trades are being made on margin, or borrowed money. This would lead to profits being multiples of the capital invested in the position. On the other hand, being 'short' means that a trader has borrowed the asset from another owner through their broker, and has sold it, with the intention of buying it back later. This somewhat riskier than being long an asset, because the trader could be forced to buy it back at a time when they would be losing money on the asset. This means that the trader's profit on a position depends on how much the asset's value has dropped, if the price sold at is p_1, and the price bought back at is p_2, then the profit is p_1-p_2.

Hidden Markov Models have been historically used to model patterns in speech recognition, and more recently, financial time series data. At its essence, a Hidden Markov Model is a Markov chain. A Markov chain is a "random process that undergoes transitions from one state to another on a state space." With this, a set of states $\{C_n : n \in N\}$ exist, with a transition probability matrix $\theta_{i=1..N,j=1..N}$ describing the probability of change from one state to another. In a discrete time Markov Chain, this definition is extended to create a time series that looks like this:



$$\begin{tikzcd} {} \arrow[r, dashed] & C_{t-2} \arrow[r] & C_{t-1} \arrow[r] & C_t \arrow[r] & C_{t+1} \arrow[r, dashed] & {} \end{tikzcd}$$

An important property of a Markov chain is the Markov property, which is that $P(C_{t+1}|C_t,...,C_1) = P(C_{t+1}|C_t)$. In words, this means that state at the time $t+1$ is only dependent on the state at time t. Another property of a Markov Chain is the calculation of state probabilities using the transition matrix at time t. With the current state at time t represented by the row matrix A of dimensions (1 x n), and the transition matrix θ of dimensions (n x n) we can model the row matrix of state probabilities at time n using this operation: $A^{(t+n)}=A \cdot \theta^n$. This gives us a row matrix of dimensions (1 x n).

Extending this to the case of an HMM, the states are 'hidden' to the user, and so only estimations can be made as to the values for their transition probabilities. This is done through observing output values from the states themselves, and using those to calculate what the transition probabilities for the states.

2.2.1 Definition and notation

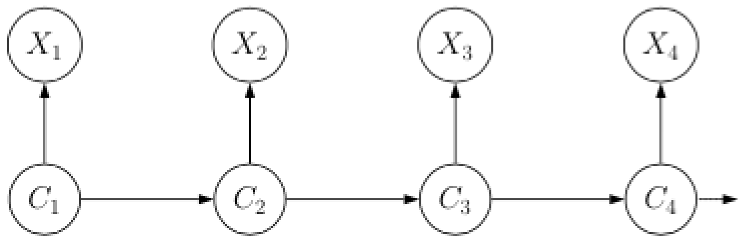


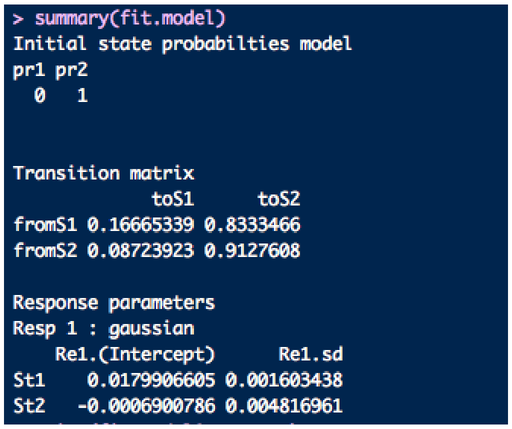
Figure 2.2 *Directed graph of basic HMM.*

In the figure above, $\{X_t : t \in N\}$ represents the observation outputted by each state $\{C_t : t \in N\}$. This observation is dependent on the probability distribution of $\{C_t : t \in N\}$. For this project, a two state HMM with a Gaussian distribution for each state was used. The distributions are independent.

Variables:
n: number of states
t: number of observations
$\theta_{i=1..n,j=1..n}$ = transition probability matrix to transition from state i to state j
$x_{t=1..n}$ = states at time t
$y_{t=1..n}$ = observations at time t
$N(\mu_i, \sigma_i^2)$ = Gaussian distribution parameters associated with each state i

In this case, I used S&P 500 daily closing values as my observations $\{X_t : t \in N\}$. The log differences from one day to another were used to normalize the values. The time window t = 60 days, and the number of states n = 2.

Given a series of observations, the objective of an HMM is to figure out the values of the transition matrix $\theta_{i=1..n,j=1..n}$ and the parameters $N(\mu_i, \sigma_i^2)$ for each state. To do this, an implementation of the expectation maximization (EM) algorithm is available in R. This algorithm works in two steps: an E step, which computes the conditional expectations of the data given the observations using θ, and the M step, which maximizes the conditional expectations with respect to θ using the new expectations from the previous step. Before the first iteration, an estimation of θ is used. This process is repeated until the changes in θ converge to a given point.



```
> summary(fit.model)
Initial state probabilties model
pr1 pr2
  0   1


Transition matrix
             toS1      toS2
fromS1 0.16665339 0.8333466
fromS2 0.08723923 0.9127608

Response parameters
Resp 1 : gaussian
    Re1.(Intercept)      Re1.sd
St1    0.0179906605 0.001603438
St2   -0.0006900786 0.004816961
```

An R screenshot showing a sample HMM's values.

In the above screenshot, the transition matrix and Gaussian parameters for a HMM on 10/10/2012 is displayed. S1 is the bull state while S2 is the bear state. As expected, the mean μ for the bull is positive and the bear is the opposite. With this, the program can calculate the probability row matrix of being in a certain state at time $t+60$. This is what is used to calculate what position needs to be taken. The algorithm functions in the following way. At time t,

1. Shift window from $X_{(t-61)}$ to X_{t-1} to $X_{(t-60)}$ to X_t and recalculate HMM
    a. The program first downloads the price history for the S&P 500 and stores it in memory. A 60 day closing price time series is then extrapolated from this time series to train the HMM with. These prices are normalized by taking the log returns on a daily scale. With $y_{t=1..60}$, we are able to create a 2 state Gaussian HMM with probability matrix $\theta_{i=1,2,j=1,2}$ using the Viterbi algorithm.
2. Use HMM to find probability p of current state
3. Recalculate 3 day exponential moving average using p
4. If probability $p > 0.7$, use respective bull/bear function
    a. Bull: long position with (p * capital)
    b. Bear: short position with (p * capital)

An exponential moving average is used for practical purposes as it reduces variance the output probability p. It is not very practical to go long one day, and have to go short the next day with the same capital. This leads to unsustainable transaction costs, which is a point that will be explained in more detail later. The figure '0.7' is an arbitrary figure that is used for both bull and bear probabilities. Since this HMM has two states, the two probabilities must add up to 1. This means that the bull market average is the inverse of the bear market average, so only one average needs to be computed for both positions. It also means that two positions cannot be entered in simultaneously.

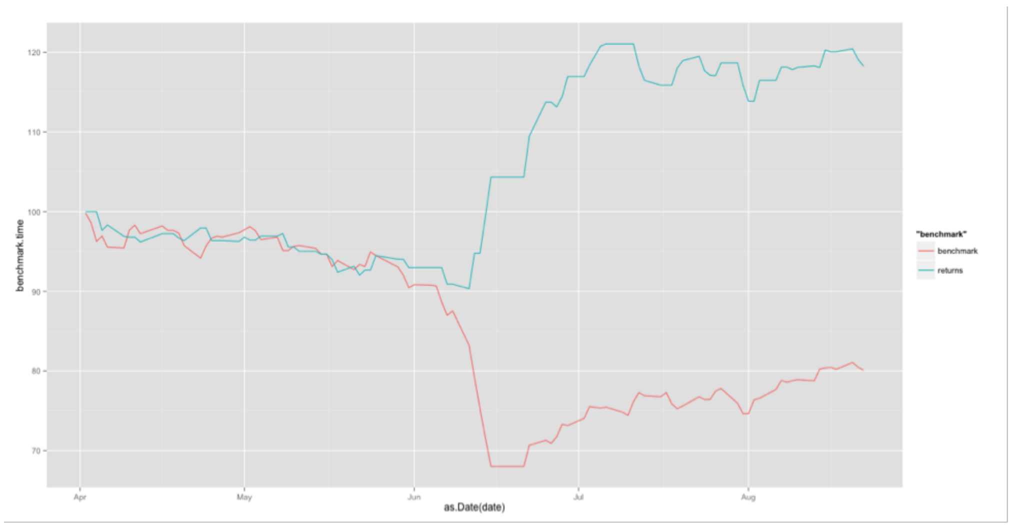
To evaluate my algorithm's return r_a, four comparison metrics were used. The first is the benchmark S&P 500. With the same starting capital c, this is invested in the same index for the whole time period t that the algorithm is run for to produce a return r_b. Comparing this to r_1 gives us a benchmark r_a-r_b. Another metric is the risk-free rate, usually defined as the rate r_t for a

1-year Treasury bond in that same year. Since Treasury bonds are universally considered to be the standard for a risk free investment, we can measure if r_a is better for the risk taken by investing in the stock market. Going along with this, the Sharpe ratio is also calculated. To calculate this, we a matrix of daily returns $r_{1..t}$ along with r_t. The Sharpe ratio function in R takes this time series and calculates $S = \frac{E[r_a - r_t]}{\sqrt{var[r_a - r_t]}}$. This is essentially the excess return over the risk free rate r_t for the standard deviation of the excess return. This gives us a better view of how risky our investments are for the amount of return that we are making. The final metric is expectancy, which is the average amount won/lost per trade $E = (avg\ winning\ trade) * (win\ \%) + (avg\ losing\ trade) * (lose\ \%)$.

Results

Year	**Return**	**Benchmark**	**Sharpe Ratio**	**Expectancy**	**Risk free rate**
2000	19.92%	-6.41%	0.728	0.001	6.12%
2001	4.6%	-24.7%	-0.197	0.0002	4.81%
2002	30.6%	-16.8%	0.962	0.001	2.16%
2003	-6.3%	13.0%	0.666	-0.0004	1.36%
2004	0.79%	9.72%	-0.125	0.0005	1.24%
2005	11.1%	5.11%	0.888	0.0008	2.86%
2006	-6.84%	11.2%	-1.212	-0.0005	4.45%
2007	-9.93%	3.28%	-1.039	-0.0007	5.06%

Table of metrics for selected years, with the best performing year highlighted



Graph of algorithm return (blue) vs S&P 500 (red) from 01/04/2001 to 08/22/2001

Conclusion

For this project, R was used for coding and implementation. R makes it easy to quickly prototype and implement an idea, and many financially oriented libraries exist, reducing the amount of work needed to test out ideas. Two of the most important libraries used were 'quantmod', which contained the function used to download time series data from Yahoo Finance, and 'TTR' which contained statistical methods to calculate metrics such as the Sharpe ratio.

To conclude, we can see from the results that the Sharpe ratio was not high enough even in the best year to conclude that the risk taken was worth the return over the risk free rate, even though in 2001 the benchmark and risk free rate were beaten by a considerable margin. In its worst year, the money lost would have eroded all the profit made in previous years, leading to the conclusion that a HMM, in the form that I used it, would not be successful in live trading. This is even with factors that were not considered that apply when live trading. Examples of these include slippage, low liquidity, transaction costs for entering and exiting the market, assuming that trades are possible at the close, and margin calls when shorting a stock.

With more refinement of the model, possibly including more states, different trading logic, and a longer/shorter window of time for HMM training, it may be possible to create a profitable algorithm. Based on our results, for a risk-taking individual, this algorithm would already have been profitable in some years, but

Code available at (https://github.com/siddharth-agarwal/hmm-algorithm) for reference.

References

Davey, Kevin. *Building Winning Algorithmic Trading Systems : A Trader's Journey from Data Mining to Monte Carlo Simulation to Live Trading*. Wiley & Sons Canada, Limited, John, 2014. Print.

"Function to Calculate NYSE Trading Day of Month." *Stack Overflow*. Web. 7 Mar. 2015. <http://stackoverflow.com/questions/13215246/function-to-calculate-nyse-trading-day-of-month>.

"Hidden Markov Models - Trend Following." *Gekko Quant – Quantitative Trading*. Web. 10 Feb. 2015. <http://www.gekkoquant.com>.

Hassan, Md Rafiul, and Baikunth Nath. "Stock market forecasting using hidden Markov model: a new approach." *Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on*. IEEE, 2005.

Idvall, Patrik, and Conny Jonsson. "Algorithmic trading: Hidden markov models on foreign exchange data." (2008).

The Comprehensive R Archive Network. Web. 21 Mar. 2015. <http://cran.r-project.org>.

Zucchini, Walter, and Iain L. MacDonald. *Hidden Markov Models for Time Series: An Introduction Using R*. Boca Raton: CRC, 2009. Print.

Acknowledgments

Dr Tao Pang of the NC State Math Department for providing his academic expertise.
My parents, for supporting me in whatever I do.