Markov Model
By Mehdi Daoudi

History

The mathematical concept behind the Markov Model was original developed by Andre Markov in the early 20th century. Andrey Markov was a prominent mathematician at the time but by the time he had begun development into probability of states, he was 50 and had already retired, though he still taught a few courses. So, his research pioneering the mathematical basis for such a prominent process in today's modern age was worked on through the later ends of his career. These models were where the probability of transitioning to a future state depends only on the current state, not on the sequence of events that preceded it.

The introduction to the hidden aspect of the hidden Markov model developed for this project was introduced to model situations where the system's state is not directly observable but can be inferred through otherwise observable events or outputs.

The key contributor to the Hidden Markov Model is Leonard E. Baum and colleagues with him developing algorithms such as Baum-Welch algorithm and the forward-backward algorithm for calculating the probabilities. These contributions were largely made in the 1960s and 1970s which is quite impressive considering it is a machine learning model and these are thought of as being recent innovations.

Some applications that Markov models have been featured in multiple machine learning techniques where it enhances the capability to handle complex and high dimensional data. Examples of this are weather prediction, transitioning between sunny days and rainy days based on weather patterns. There are also LLMs today that use Markov Models to generate text by predicting the likelihood of a word following a sequence of words. Search engine algorithms follow in a similar manner.

Today we are testing the hidden Markov model and the simple Markov model we are testing it within the stock market, more specifically it's application to market trends by analyzing the sequence of past market states.

Background

Within my application today, I sought out inspiration for the model from a paper titled, "Regime-Switching Factor Investing with Hidden Markov Models", by Matthew Wang from Northwestern University in the Electrical and Computer Engineering department. Within this paper there is a brief literature review on different models of investing and how a hidden Markov model applied to investment analysis. Within this, it speaks of this concept called "regime classification." It was articulated to have been used to determine the state of the

market as hidden states to optimize items such as a Sharpe ratio which could yield to superior portfolio performance across a variety of different asset classes.

Within this paper, it focuses on the concept of identifying different market regimes and uses the example of bull, bear, and neutral markets and how this impacts investment strategies. The implementation discussed classifies these regimes based on historical data from the S&P 500.

Within my own implementation, I sought out to achieve a different goal using a similar basis, that being rather than focusing on investing strategies, focusing on trading strategies. Similar to the other paper, it is only one piece of the puzzle and needs other parts to back-test using reliable data metrics like a Sharpe Ratio. Within the paper it shows that the regime switching model works in delivering a higher return and outperforms the same strategies less the HMM model implementation. For our implementation, it is in hopes of providing a model that will allow me to adjust the observable states based on more information, to identify trading focused strategies such as, “Mean Reversion”, “Trending”, and “Random Walk”.

“Mean Reversion”, is meant to show times that are trading within a range where typically it is trading around a mean during a period. “Trending”, is meant to show times that are trading in one direction up or down. “Random Walk”, is meant to describe market patterns that do not fit either of these market trading regimes. Due to my searching for this, the data I primarily focused on optimizing for was the 4-hour data, this as it is smaller than the daily time frame providing more periods for information and gives a piece of the intraday trading. This can be construed as a mid-frequency time frame.

Once a state is identified, it can be used to optimize a trading strategy specifically for those conditions.

Mathematical Model & Expression & Implementation

Within the mathematical model, there are a few key algorithms and concepts at play here, especially when discussing the hidden Markov model. Regarding the base Markov model, it is made using a simple transition matrix:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Figure 1. Transition Matrix

Within the transition matrix, there would lie the initial probabilities. Since this is a simple Markov model, it would require you to place them yourself based on your own knowledge or intuition. As you could probably tell, this is not very efficient. Here is a visualization of the Markov model process for the probabilities. This are called transition probabilities.

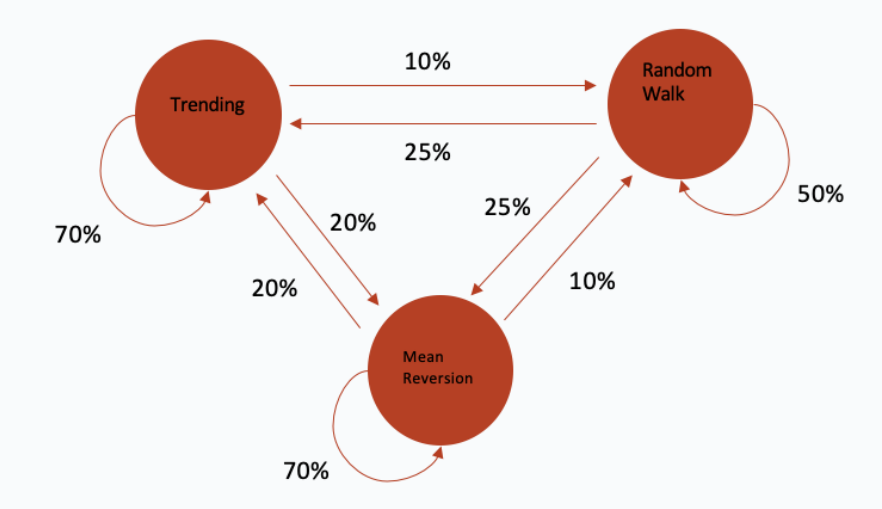

Figure 2. Simple Markov Model

Since you recognize that the issue is your inability to determine the probabilities of the states here, these will now be titled “Hidden States” and as we look to define these probabilities, we will add in some “Observable States”. The probability from these “Hidden States” to the “Observable States” are called Emission Probabilities and we will use a Gaussian Probability density function formula to calculate these values. This works especially well in this case as the Gaussian distribution with its spread determined by its variance is effective at modeling the variations in financial data which often is quite noisy. The emission probabilities being calculated are defined as the likelihood of observing a specific data feature given a hidden state. So, the Gaussian functions we have calculate the probability of observing a feature value given a state’s Gaussian distribution by using the Gaussian probability density function.

$$\mathrm{p}(\mathrm{x} \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Figure 3. Gaussian Probability Density Function

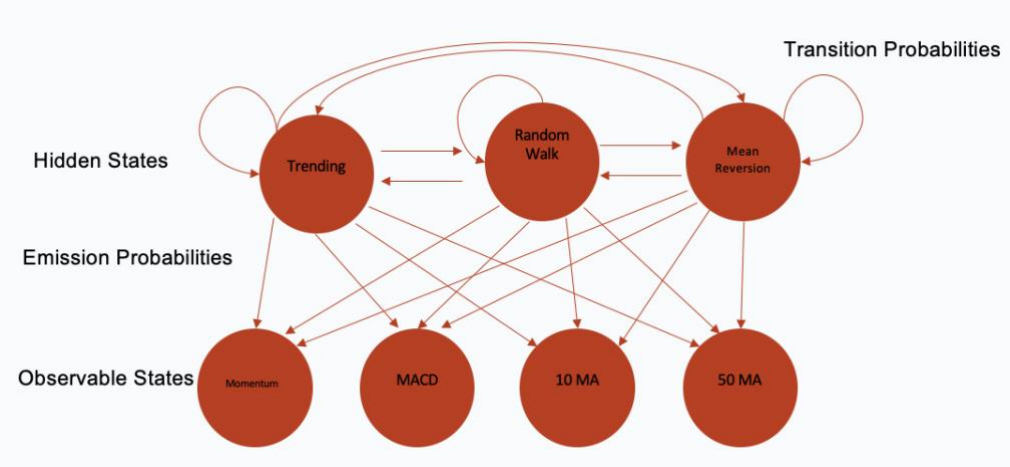

Figure 4. Hidden Markov Model Implementation Visualization

The "Observable States" defined in Figure 4 are "Momentum", "MACD", "10 MA", "50 MA", which are simple observations that are a little more complex than simply price. To explain each one by one, "Momentum" is defined by the percent change, a simple rate of price change. The moving averages are defined by a price period rolling average, the "10 MA" being 10 periods and the "50 MA" being 50 periods. The MACD is simply the "10 MA" subtracted by the "Long MA". These are all commonly used features in indicating what underlying price action is doing.

The other algorithm used within the scope of manipulating the data is the K-means clustering algorithm. This is used to identify natural groupings or clusters within the dataset. We apply the clustering to the financial indicators to find the clusters that potentially correspond to our different hidden states. The centroids resulting from the clustering gives us a practical and well-informed starting point for the state means which serves to improve efficiency and is supposed to help with the convergence used within the Baum-Welch algorithm later.

Onto the pieces within the Baum-Welch algorithm, we can discuss the Forward and Backward algorithms implemented here:

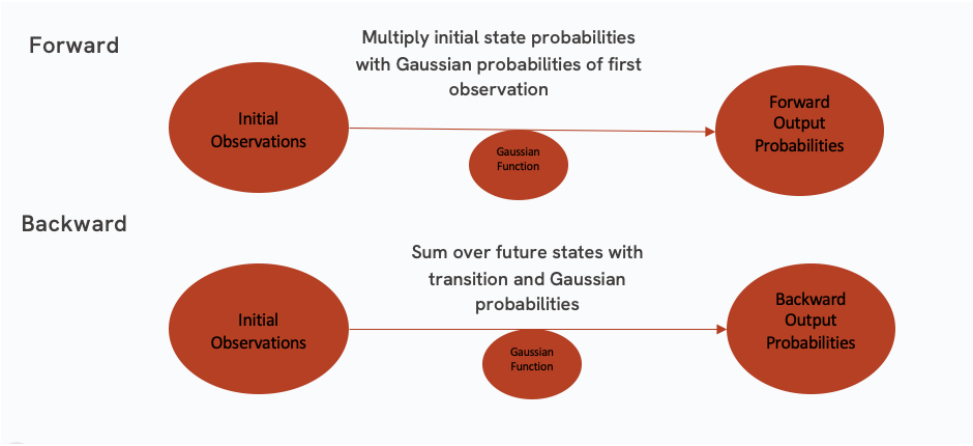

Figure 5. Visualization of Forward and Backward Algorithms

The forward and backward algorithms are meant to calculate the probabilities of state sequences given observed states from the indicators. The forward algorithm works by multiplying initial state probabilities by Gaussian emission probabilities for the first observation. It then iterates through each observation, calculates the state probabilities by summing over previous states and considering the transition probabilities and Gaussian emission probabilities. Through this, it gives the probability of being in each state given observations up to that point.

The backward algorithm works in a similar manner however outputs the probabilities of observing future sequences from each state. It does this by first setting probabilities from each state to the end of the observation sequence, iterating through each observation and calculating state probabilities by summing over future states and it also considers transition probabilities and Gaussian emission probabilities.

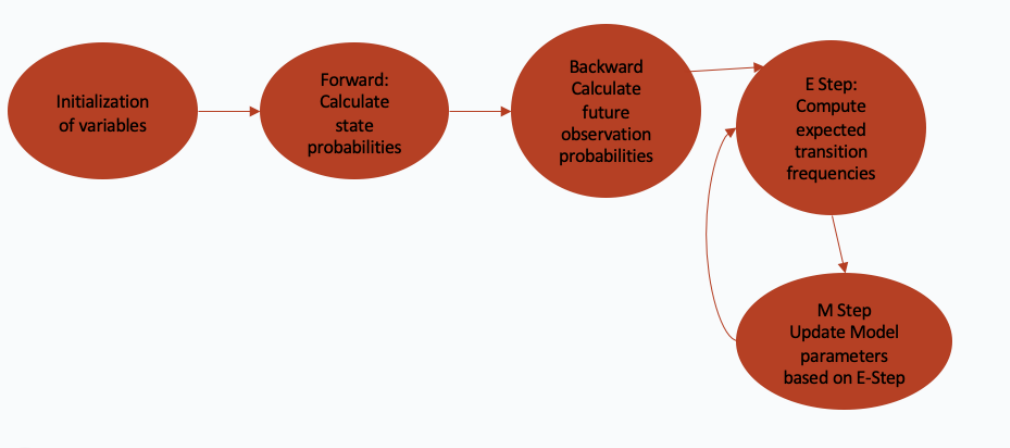

Figure 6. Visualization of Baum-Welch

The Baum Welch is a machine learning algorithm that works as described in the Figure 6. This algorithm aims to converge to a set of parameters that maximizes the likelihood of the observed data given the model. It iteratively updates and refines the state means variances for the gaussian emission probabilities for each state making the probabilities overtime more representative of the observed data. It is supposed to improve the probabilities of transitioning between states which better reflects the dynamics observed in the data.

Our implementation includes the initialization which defines the states and the variables. It then calls the forward and backward procedure to calculate the probabilities of being in each state up to each observation point, and to calculate probabilities of future observations from each current state moving backward in the observation sequence using Gaussian probabilities respectively. It then goes through an “Expectation Step” (E-Step) which computes the expected transition frequencies between states using the forward and backward probabilities. It then

goes through the “Maximization Step” (M-Step) which updates the model parameters, being the start probabilities, transition probabilities, state means and variances based on the E-step. After, it continuously iterates through the E-Step and M-Step refining the model and maximizing the likelihood of the observed data.

Lastly, with the returned probabilities, transition probabilities, state means and state variances, we call the Viterbi algorithm:

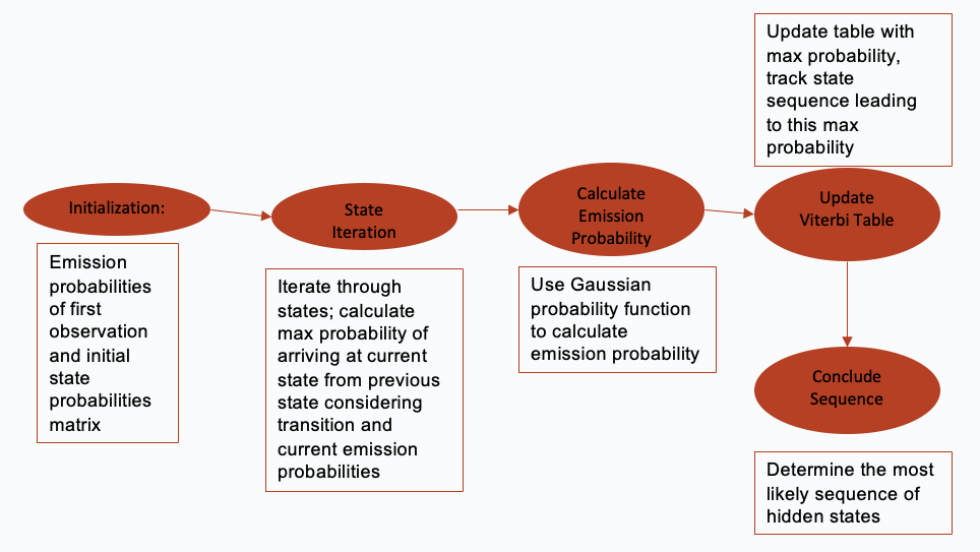

Figure 7. Visualization of the Viterbi Algorithm

The Viterbi algorithm is designed to determine the most likely sequence of hidden states. It first begins by initializing the first column of its table. This initialization involves the emission probabilities of the first observation for each hidden state and the initial state probabilities. After, the program iterates through the states and here it calculates the maximum probability of arriving at the current state from any of the previous states, considering the transition probabilities between states and the emission probability of the current observation given the current state. The next step uses the Gaussian probability function to calculate the emission probability. At each step the algorithm updates the “Viterbi table” with the maximum probability calculated and keeps track of the state sequence that led to this maximum probability. It then backtracks to find the optimal path from the last observation to the first following the path of maximum probabilities that were recorded and this is the most likely sequence of states.

Interpretation

These observable states are an area of potential improvement.

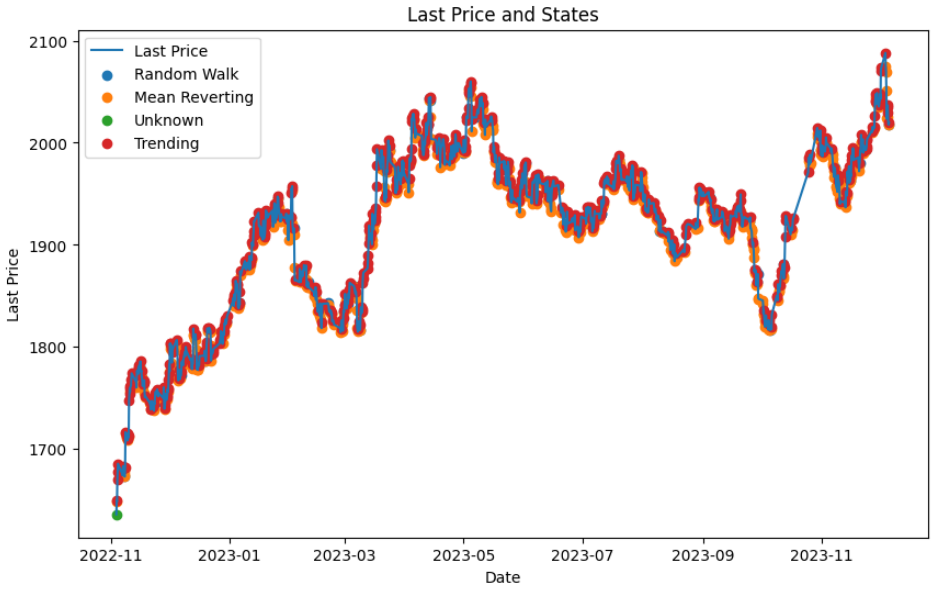

Figure 8. Simple Markov Model Tested on 4-Hour Gold Futures Data

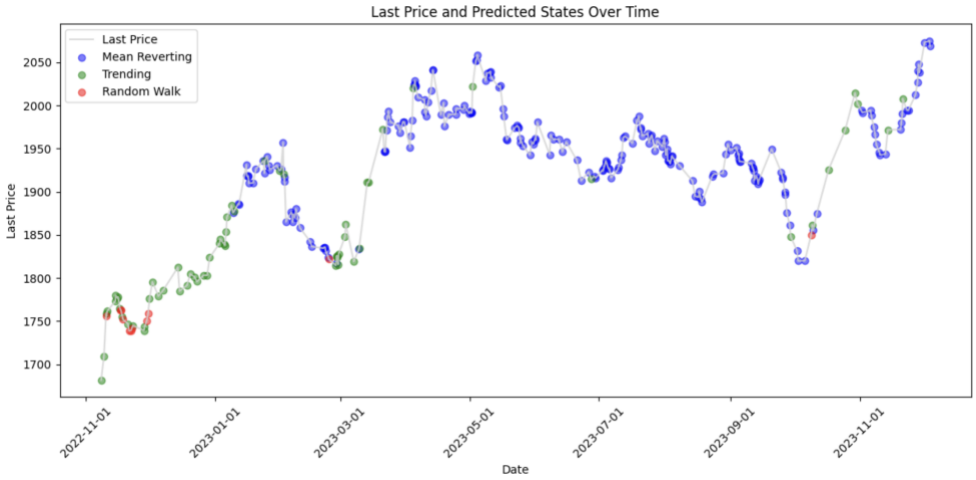

Figure 9. Hidden Markov Model Trained on 80% of the data Tested on 4-Hour Gold Futures Data

Within these two, we can see a stark contrast in the success of the models. In Figure 8, we can see that the states are seemingly random to which effectively the probabilities are as I just added them based on intuition. However, with the improvements made with the Hidden Markov Model with the probability calculation and maximizing of probability outcomes iteratively through the algorithms, we can see a drastic improvement of the model's ability to recognize a difference between the states despite the observations being rather simple.

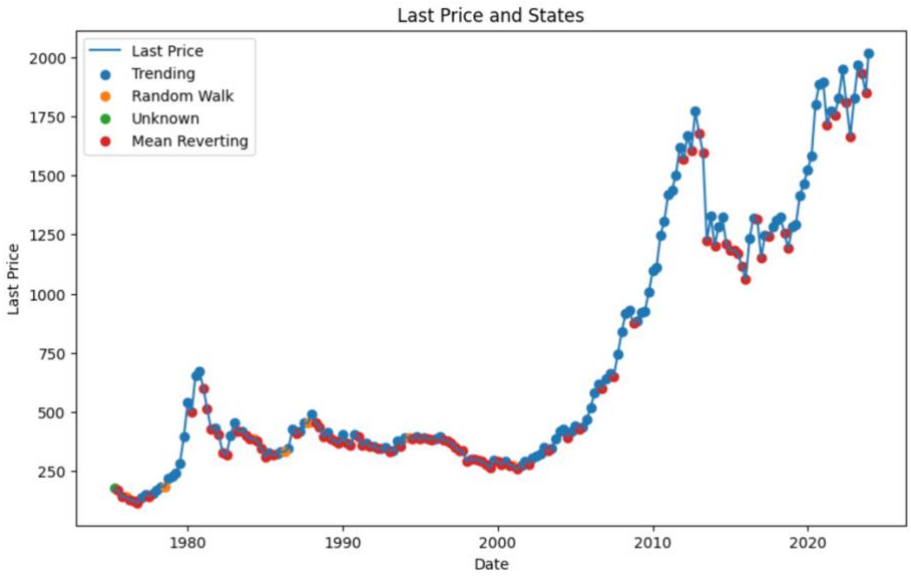

Figure 10. Simple Markov Model Tested on Quarterly Gold Futures Data

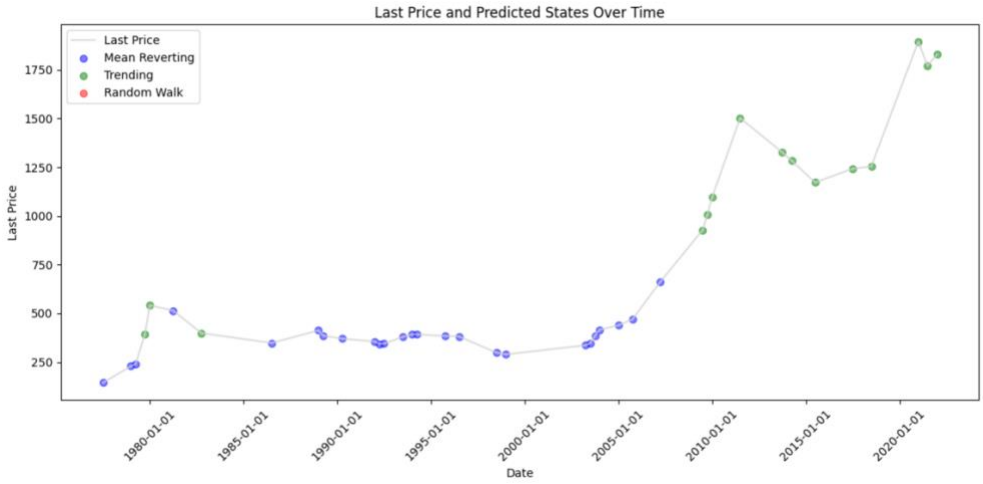

Figure 11. Hidden Markov Model Trained on 80% and Tested on Quarterly Gold Futures Data

Within the quarterly test, we can see exemplified how in the initial paper, using it for regime shifting for investing purposes does work as well as you can see, it is also just as good at recognizing these market states as it is for higher frequency data. You can also see that even the simpler model performed slightly better on the longer-term data citing that the longer your time horizon the less exact your implementation must be.

Conclusion

To conclude, we can see that while this is a great start, there are many further ways to improve upon this. One way is including more data within the training set to adequately test on more

data. We can also refine our observation states as well to be more precise and follow more nuanced or innovative techniques for determining these trends. We can also begin to work on specific strategies to implement during these time periods and utilize the Hidden Markov Model as a stop loss in effect to cut the algorithm or trade whenever a state shift is recognized.