

# 1 Introduction

Here I will discuss an attempt to use a Hidden Markov Model to predict and make simulated investments in the stock market. The problem of predicting the stock market is extremely complex - its rise and fall is dictated by too many factors to be able to predict accurately. However, my idea is we may not need to predict the stock market completely accurately in order to be able to profit from it - we just need to be able to predict it better than random. If we can consistently predict the market better than random, then repeated investments may allow an investor to beat the market.

The Markov Model approach to predicting the market is motivated by this philosophy. The MM is a relatively simple case of Artificial Intelligence wherein we try to predict the future using only the events of the recent past. Every event in a Markov chain has a set of possible states that have a certain probability of being the true state for the duration of that particular event. We can make an educated guess about the state of the current event by tabulating the sequential co-occurrence of states in the training data. For example, assume that we let the relative rise or fall of a stock price be the "state" we are measuring in the stock market. If we see a big jump in the stock market preceded by two moderate jumps at several occasions in the training data, our model will be likely to invest big after it sees two moderate jumps in the test data.

One could make the argument that this would be a weak predictor of the rise in the stock market; however, a strong predictor may not be needed. Over time, a repeated string of weak successes could compound on top of each other to create a large increase in income, and a large increase in performance of the stock market.

# 2 The Markov Model Framework

[MORE RESEARCH ON THE MATH OF HMMS NEEDED]

The Markov Model framework assumes a system with a finite number of possible states exists at every moment time, and it assumes that the state of the system at one moment depends entirely on the state of the system at the previous moment. In real life, a Hidden Markov Model might be well suited for predicting the next word in a sentence, because word order in sentences often follows a familiar pattern. This is in contrast to models that assume a completely independent relationship between neighboring states (such as predicting the outcome of a coin flip), or models in which subsequent states depend on lots of information about the previous state (like the position of all the pieces in a chess game).

In this specific case, we create a model which repeatedly makes a series of short-term investments in the stock market. Each short-term investment is

based on a very small number of previous states - this is done because it allows us to have much more accurate information on all the possible combinations of prior states we are examining. For example, assuming there are  $s$  states and our visibility (i.e. the number of prior states we use to make our prediction) is  $v$ , the total number of distinct prior combinations of states we would need to understand from the training data is  $v^s$  - so the amount of training data we need to make an accurate decision increases exponentially in the visibility and the number of states.

We could also differentiate between combinations of prior states that occur in a different order - for example, should we consider a sequence of states  $S_1 = \{st_1, st_2\}$  different from the sequence of states  $S_2 = \{st_2, st_1\}$ ?

In our model, we have initially set the visibility to 2 and the number of states to 7.

## 2.1 Meaning of "States"

Every state in our HMM model represents a different level of change in the stock we are trading on. More specifically, we split the distribution of the magnitude of daily change into  $s$  equal parts, and represent each sector with one of  $s$  states. Assume we are attempting to make investments in Apple's stock. We then select a set of historical daily stock closing prices for Apple ( $A = \{a_0, a_1, \dots, a_n\}$ ) and use this as our training data. To create states for our HMM, we want to analyze the differences in stock price from day to day - let's denote this as:  $A' = \{a'_0, a'_1, \dots, a'_{n-1}\}$ , where  $a'_k = a_{k+1} - a_k$ .

Then, each state we define will be of the form  $s_t = s_j$  iff  $[a'_{\frac{n*j}{s}}, a'_{\frac{n*(j+1)}{s}}]$ , where  $s_t$  is the state assigned to the model at time  $t$ . Thus, the first state will represent the 0th through  $\frac{100}{7} = 14$ th percentile of the daily stock market changes, the second will represent the 14th through the 28th, etc. Let  $\{S_j\}$  represent this set of 7 states, and  $\{S_T\}$  represent the series of states we encounter over time.

## 2.2 Predicting Future States

As mentioned previously, the model predicts the state  $s_{t+1}$  based on  $s_t$  and  $s_{t-1}$ . More specifically, it uses the training data to compute the probability distribution  $\{P(s_{t+1} = s_j | s_t, s_{t-1}) \forall s_j \in \{S_j\}\}$ . This prediction is based on the results that were seen in the training data. For every combination of states  $\{s_t, s_{t-1}\}$  that occur in the training data, the state  $s_{t+1}$  is recorded. After all the training data is recorded, the set of outcomes  $s_{t+1}$  that were seen in the data is converted into a probability distribution - note that this relies on the assumption that the test data will have the probability distribution of future states as the training data. See the pseudocode algorithm below for more details.

```
statemap = dictionary([s_a, s_b] ↦ list(len(S_j)))
for s_t in training_data:
...stateinput=sort(s_{t-1}, s_t)
```

```

...statelist ← statemap[stateinput]
...statelist[st+1] += 1
...statemap[stateinput] ← statelist
end for

stateprobs ← statemap / sum(statemap)

```

Note that we sort the inputs  $\{s_{t-1}, s_t\}$  to the dictionary because we want to compute these probabilities in a manner independent of the order in which the past states occur (otherwise there might be too many possibilities to predict accurately).

### 2.3 Computing the Utility of an Investment

We have now discussed the way in which our model tries to predict the stock market. One final component we need to discuss is the decision-making algorithm, where the model decides, based on the probability distribution of potential future states (which we will call  $P_{S_j}$ ), how much of its money to invest in the stock market.

The model is given the ability to choose the most efficient of nine potential amounts of money to invest:  $\{0, \frac{1}{8}M, \frac{2}{8}M, \dots, 1 * M\}$ , where  $M$  is our total wealth. For each amount, we compute the `moneychangeratio`, which is the fraction of our current wealth that will be present after we invest. In order to calculate this, we will need to know the expected gain ratio corresponding to each state in  $\{S_j\}$  - we compute this by finding the average gain of qualifying examples from the training data, and we denote it  $E_g[s_j]$ .

Below, we compute the expected monetary value of our investment ( $I_{t+1, s_j}$ ) after the next close of the stock market, assuming that the change is denoted by state  $s_j$ :

$$I_{t+1, s_j} = I_t * E_g[s_j]$$

Next, we compute the expected gain ratio for a given investment amount  $I_t$  and a given state  $s_j$  denoted  $E_g[I_t, s_j]$  - note that after the investment, the wealth *not* invested ( $M - I_t$ ) is unchanged and the wealth invested grows from  $I_t$  to  $I_{t+1, s_j}$ , so the future wealth is  $M - I_t + I_{t+1, s_j}$ :

$$E_g[I_t, s_j] = \frac{1}{M}[(M - I_t) + I_{t+1, s_j}]$$

Once we know the expected fraction of change in wealth  $E_g[I_t, s_j]$ , we apply a special utility function `util0` which maps the `moneychangeratio` to a user-specified utility value  $u$ . The function then adds up the value  $u$  computed for each potential state  $s_j$ , weighted by the probability of encountering the state  $s_j$ . The `util0` function allows the user to control the investment behavior of the model with a function such as  $util0 = 2^{E_g[I_t, s_j]} - 1$ , a potential monetary gain is given a utility that is very large in the positive direction, while a loss

( $E_g[I_t, s_j] < 1$ ) is given a utility that is very small in the negative direction. This creates a model that is that is very willing to take risks, in spite of the high costs it may encounter.

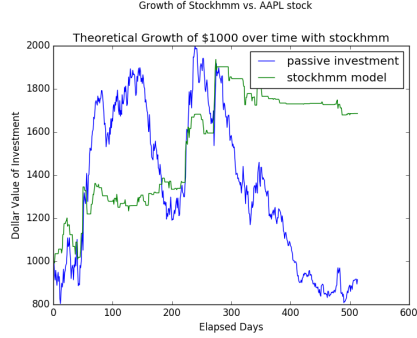
$$\text{Util}(I_t) = \sum_{s_j \in S_j} P(s_{t+1} = s_j | s_t, s_{t-1}) * \text{util0}(E_g[I_t, s_j])$$

Above, we computed the final utility for the investment  $I_t$ . We will compute such a utility for all possible investment amounts, and then choose to invest the amount with the highest utility.

Note that in our current model, we use the utility function:  $f(E_g) = \{E_g \text{ iff } E_g \geq 1, \ln(E_g) + 1 \text{ iff } E_g < 1\}$ . Thus, the model is very risk-averse.

### 3 Results

The results for the current model ( $f(E_g) = \{E_g \text{ iff } E_g \geq 1, \ln(E_g) + 1 \text{ iff } E_g < 1\}$ ) are shown below for the stock AAPL.



In this result, we can see that our model worked as we wanted it to - it was correctly able to predict times when the stock market would crash and pulled out at those times; and it was able to correctly predict times when the market would rise and invested a lot of money in those cases. As a result it was able to beat the market by a substantial amount. There are many specific cases in which our model outperformed the market. However, in general, its prediction accuracy was limited.

We also performed several other tests using 100 years of data from the Dow Jones industrial average. In order to test how successful the model was over short or long periods of time, we performed a procedure similar to k-fold cross validation, with increasing values of  $k$ , and computed the average amount of time that the HMM was valued higher than the stock market, as well as the average amount of difference between the HMM and the actual market. Here are the results:

num. folds	interval length (yrs)	avg. difference (\$)	ratio ahead of mkt.
5	20	-611.5	0.246
10	10	-179	0.468
100	1	-24.05	0.377

As mentioned before, these results were obtained using a risk-averse investment model. Several other models were used as well (i.e. linear-risk, risk-loving, etc.); however, their results were not significantly better than the above.

As we can see in this table, our model didn't manage to beat the market very often in the general case. As we can see by the average dollar differences, it did almost 60% worse than the market for very long term data (20 years) and about 20% worse in the medium-long term (10 years). Its best performance by far was for very short term investments, where it was only about 2% worse than the market. Interestingly, the ratio of time ahead of the market was higher in the medium term. It is important to note that the trends in this table can be prone to high levels of variation - for instance, it is possible that the model pushes far ahead of the market after just a few days, and then spends years making bad decisions but still remaining ahead of the market because of its entrenched advantage - in this case, the model will be ahead of the market 90% of the time, but this number does not represent the many bad decisions that it made after getting ahead of the market.

Finally, we test the precision and recall of the decision-making component of our HMM. Aggregating data over the entire 100-year Dow Jones data, we found that in instances when the model decided to invest any amount of money, the stock market went up 53% of the time. Therefore, we estimate the precision at 53%. In instances when the market actually increased, the model decided to invest 48% of the time. This was the result we found with the risk-averse model. Again, we tested the precision and recall with several other models, and found that the precision and recall were similarly low in all cases.

In light of this information, the long term trends we saw when using this model make sense. The model is only slightly more accurate than random guessing, and often will not invest any money; as a result, it is very easy for it to miss out on long-term growth. There are cases where the model does do well - usually for short term investments, and when the stock market tends to fall, it will do better than the market - in rare cases even seeing significant growth despite the poor performance of the market.