Bachelor of Technology In Artificial Intelligence

21MIS301 – Mathematics for Intelligent Systems

Observation Report

Topic:

Stock Price Prediction using Hidden Markov Model

Done by:

J Viswaksena – AIE21035

T Uday Krishna – AIE21064

Professor/Guide:

Dr. Don S, Dr. Sunder Ram K

1 Introduction:

Hidden Markov Models (HMMs) present an innovative solution to the intricate challenges posed by the dynamic and unpredictable nature of stock market behavior. This study delves into applying HMMs to predict stock prices for the upcoming trading day based on historical data. HMMs model underlying market states, representing factors such as economic conditions, company policies, and investor sentiment. The training of HMMs involves Maximum a Posteriori (MAP) estimation, determining the most probable state sequence and model parameters given observed stock prices. Predictions rely on state transitions, considering the probabilities of moving between states and expected price behavior within each state. HMMs seamlessly integrate the time series aspect of stock data and account for market volatility by representing distinct states like bullish or bearish markets. These hidden states, not directly observable, guide predictions by modeling transitions, offering insights into potential price movements. While not a crystal ball, HMMs provide a data-driven method for structured analysis of market dynamics, aiding informed investment decisions. It's essential to note that stock price prediction is inherently challenging, and while HMMs don't guarantee perfect accuracy, they offer valuable insights into the market's potential behavior.

2 Mathematical Implementation:

HMMs offer a valuable approach to stock price prediction by capturing the hidden dynamics of the market and providing insights into potential future trends. They can be used alone or in combination with other techniques to inform investment strategies and improve risk management.

1. We assume that the distribution of each features is across an evenly spaced interval instead of being fully continuous

2. We assume possible values for the start and end of the intervals

2.1 Gaussian Probability Distribution:

Notation:

- X: The random variable following a Gaussian distribution.
- μ: The mean (average) of the distribution.
- σ^2: The variance (spread) of the distribution.
- σ: The standard deviation, which is the square root of the variance.

Probability Density Function (PDF):

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Explanation:

- f(x): Represents the probability density of X taking a value x.
- The exponential term, giving the characteristic bell-shaped curve.

$$\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- $(1 / \sqrt{(2\pi\sigma^2)})$: Normalization constant, ensuring the total probability under the curve equals 1.

Key Features of the Gaussian Distribution:

- Symmetric: The shape is perfectly symmetrical around the mean μ.

- Unimodal: Has a single peak at the mean.
- Defined by μ and σ^2: These two parameters fully specify the distribution.

Standard Normal Distribution:

- A special case with $\mu = 0$ and $\sigma^2 = 1$.
- Often denoted by $\varphi(x)$ or $N(0,1)$.

Cumulative Distribution Function (CDF):

- Gives the probability that X is less than or equal to a certain value x.
- Denoted by $\Phi(x)$ for the standard normal distribution.
- Tables or software are usually used to calculate CDF values.

Using the Gaussian Distribution:

- Modeling real-world phenomena that exhibit a bell-shaped distribution (e.g., heights, IQ scores, measurement errors).
- Making statistical inferences (e.g., hypothesis testing, confidence intervals).
- Gaussian HMMs assume observations within each hidden state follow a Gaussian distribution.

2.2 Expectation-Maximization (EM) Algorithm:

Notation:

- N: Number of states in the HMM
- M: Number of distinct observation symbols
- T: Length of the observed sequence
- $O = (o_1, o_2, ..., o_T)$: Observed sequence
- $A = (a_{ij})$: Transition probability matrix (a_{ij} = probability of transitioning from state i to state j)
- $B = (b_j(k))$: Emission probability matrix ($b_j(k)$ = probability of observing symbol k in state j)
- $\pi = (\pi_i)$: Initial state probability vector (π_i = probability of starting in state i)
- $X = (x_1, x_2, ..., x_T)$: Hidden state sequence

E-Step (Expectation):

1. Calculate forward probabilities: $\alpha(i, t) = P(o_1, o_2, ..., o_t, x_t = i \mid \lambda)$
2. Calculate backward probabilities: $\beta(i, t) = P(o_{t+1}, o_{t+2}, ..., oT \mid x_t = i, \lambda)$
3. Calculate expected state occupancy: $\gamma(i, t) = P(x_t = i \mid O, \lambda) = \alpha(i, t)\, \beta(i, t) / P(O \mid \lambda)$

4. Calculate expected state pair occupancy: $\xi(i, j, t) = P(x_t = i, x_{t+1} = j \mid O, \lambda) = \alpha(i, t)\, a_{ij}\, b_j(o_{t+1})\, \beta(j, t+1) / P(O \mid \lambda)$

M-Step (Maximization):

1. Update transition probabilities:

$$\hat{a}_{ij} = \frac{\sum_t \xi(i, j, t)}{\sum_t \gamma(i, t)}$$

2. Update emission probabilities:

$$\hat{b}_j(k) = \frac{\sum_t \gamma(j, t)\delta(o_t, k)}{\sum_t \gamma(j, t)}$$

($\delta(o_t, k)$ = 1 if $o_t = k$, 0 otherwise)

3. Update initial state probabilities: $\hat{\pi}_i = \gamma(i, 1)$

Iteration:

- Repeat E-step and M-step until convergence (parameters stabilize).

2.3 Forward Algorithm:

Notation:

- $\alpha(i, t)$: Forward probability (probability of being in state i at time t and generating the first t observations)

Formulas:

1. Initialization:

$$\alpha(i, 1) = \pi_i \cdot b_i(o_1)$$

for i = 1, 2, ..., N

2. Induction:

$$\alpha(j, t+1) = \sum_{i=1}^{N} \alpha(i, t) \cdot a_{ij} \cdot b_j(o_{t+1})$$

for j = 1, 2, ..., N and t = 1, 2, ..., T-1

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha(i, T)$$

(total probability of the observed sequence)

Interpretation:

- α(i, t) represents the probability of being in state i at time t and having generated the observed sequence $o_1, o_2, ..., o_t$.
- The algorithm efficiently computes these probabilities using a dynamic programming approach.

Uses:

- Computing the likelihood of an observed sequence given an HMM: P(O | λ)
- Decoding the most likely state sequence: Viterbi algorithm uses forward probabilities as a basis.
- Parameter estimation: Expectation-Maximization (EM) algorithm uses forward probabilities in the E-step.
- Efficiently calculates the probability of an observed sequence given model parameters.
- Uses dynamic programming to avoid redundant computations.

2.4 Backward Algorithm:

Notation:

- β(i, t): Backward probability (probability of generating the remaining observations given that you are in state i at time t)

Formulas:

1. Initialization: β(i, T) = 1 for i = 1, 2, ..., N
2. Induction:

$$\beta(i,t) = \sum_{j=1}^{N} a_{ij} \cdot b_j(o_{t+1}) \cdot \beta(j,t+1)$$

for i = 1, 2, ..., N and t = T-1, T-2, ..., 1

Interpretation:

- β(i, t) represents the probability of generating the remaining observations $o_{t+1}, o_{t+2}, ..., o_T$ given that you are in state i at time t.
- The algorithm efficiently computes these probabilities using a dynamic programming approach, working backward through the sequence.

Uses:

- Computing the likelihood of an observed sequence given an HMM:

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i \cdot b_i(o_1) \cdot \beta(i, 1)$$

- Decoding the most likely state sequence: Viterbi algorithm utilizes backward probabilities to reconstruct the optimal path.
- Parameter estimation: Expectation-Maximization (EM) algorithm uses backward probabilities in the E-step.

2.5 Viterbi Algorithm:

Notation:

- δ(i, t): Highest probability of any path ending in state i at time t
- ψ(i, t): Most likely state at time t-1 in the most likely path ending in state i at time t (backpointer)

Formulas:

1. Initialization:

$$\delta(i, 1) = \pi_i \cdot b_i(o_1)$$ for i = 1, 2, ..., N ψ(i, 1) = 0 (no backpointer at the first time step)

2. Recursion:

$$\delta(j, t+1) = \max_{i=1}^{N}[\delta(i, t) \cdot a_{ij}] \cdot b_j(o_{t+1})$$

for j = 1, 2, ..., N and t = 1, 2, ..., T-1

ψ(j, t+1) = $\text{argmax}_{i=1}^{N}$ [δ(i, t) * a_{i}□] (store backpointer)

3. Termination: P* = $\max_{i=1}^{N}$ δ(i, T) (probability of the most likely path)
4. Path Reconstruction:

$$x_t^* = \arg\max_{i=1}^{N} \delta(i, T)$$

(most likely final state) x□$_{-1}$* = ψ(x□*, T) (backtrack to previous state) ... x_1* = ψ(x_2*, 2) (backtrack until the first state)

Interpretation:

- Viterbi algorithm finds the most likely sequence of hidden states (x_1*, x_2*, ..., x□*) that could have generated the observed sequence O, given an HMM λ.

- It uses dynamic programming to efficiently explore all possible state sequences and select the best one.

Key Points:

- Induction involves iteratively updating δ(j, t+1) and ψ(j, t+1) based on previous values and model parameters.
- Backpointers enable efficient reconstruction of the most likely path.
- It's a fundamental algorithm for HMM decoding and often used in speech recognition, natural language processing, and other sequence analysis tasks.

2.6 Baum-Welch Algorithm:

Notation:

- ξ(i, j, t): Expected number of transitions from state i to state j at time t, given the model and the observed sequence
- γ(i, t): Expected number of times in state i at time t, given the model and the observed sequence

Formulas:

E-Step (Expectation):

1. Calculate forward probabilities α(i, t) using the Forward Algorithm.
2. Calculate backward probabilities β(i, t) using the Backward Algorithm.
3. Calculate

$$\xi(i, j, t) = \frac{\alpha(i, t) \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot \beta(j, t+1)}{P(O|\lambda)}$$

4. Calculate

$$\gamma(i, t) = \frac{\alpha(i, t) \cdot \beta(i, t)}{P(O|\lambda)}$$

M-Step (Maximization):

1. Update transition probabilities:

$$\hat{a}_{ij} = \frac{\sum_t \xi(i, j, t)}{\sum_t \gamma(i, t)}$$

2. Update emission probabilities:

$$\hat{b}_j(k) = \frac{\sum_t \gamma(j,t) \cdot \delta(o_t, k)}{\sum_t \gamma(j,t)}$$

3. Update initial state probabilities: $\hat{\pi_i} = \gamma(i, 1)$

Iteration:

- Repeat E-step and M-step until convergence (parameters stabilize).

Key Points:

- Baum-Welch algorithm is a type of Expectation-Maximization (EM) algorithm specifically tailored for HMMs.
- It aims to find maximum likelihood estimates of the model parameters (A, B, π) when only the observed sequence is available.
- Works iteratively, alternating between estimating expected values of hidden variables (E-step) and maximizing likelihood based on those estimates (M-step).
- Convergence is not guaranteed, but often works well in practice.
- It's essential for training HMMs in various applications, including speech recognition, natural language processing, and bioinformatics.

2.7 Build Markov Grid

- Let P be the set of unique patterns.
- Let N be the number of unique patterns.

For each i and j in P, calculate the transition counts C_{ij}.

$$T_{ij} = \frac{C_{ij}}{\sum_{k=1}^{N} C_{ik}}$$

2.8 Log Odds Calculation

- Given a sequence $S=(s1, s2, \ldots, sm)$ where si represents a pattern.
- For each $i=1,2,\ldots,m-1$, calculate log odds logOdds_i as follows:

$$\text{logOdds}_i = \log\left(\frac{T_{s_i, s_{i+1}}}{\epsilon + T_{\text{nonexistent}, s_{i+1}}}\right)$$

3 Drawbacks or Novelty:

3.1 Drawbacks for existing model:

1. Assumption of Independence in Traditional Models:

Traditional time series models like ARIMA and linear regression often assume independence between data points, neglecting potential dependencies that HMMs can capture through hidden states.

2. Handling Non-Linearity in Machine Learning Models:

Machine learning models, while powerful, may struggle to capture non-linear relationships effectively. HMMs, with their inherent state transitions, can better represent complex dependencies in sequential data.

3. Incorporating Time-Varying Regimes:

HMMs are particularly useful for modeling time-varying regimes, where market conditions may switch between different states. Other models may struggle to adapt to such changes, especially if they assume a constant relationship over time.

4. Interpretable State Transitions:

HMMs provide a clear framework for interpreting hidden states and state transitions, which may be lacking in some machine learning models. Understanding the underlying states can be valuable for decision-making in financial markets.

5. Sequential Dependency in Neural Networks:

While neural networks, especially recurrent and LSTM networks, can capture sequential dependencies, they might suffer from vanishing or exploding gradient problems during training. HMMs, on the other hand, are designed to handle sequential data more explicitly.

6. Data Efficiency and Overfitting:

In the context of limited data, complex machine learning models may be prone to overfitting, especially if the number of parameters is high. HMMs, with a simpler structure, may be more data-efficient in certain situations.

7. Handling Missing Data:

HMMs can handle missing data more naturally than some other models. This can be important in financial markets where data may not be available for every time point.

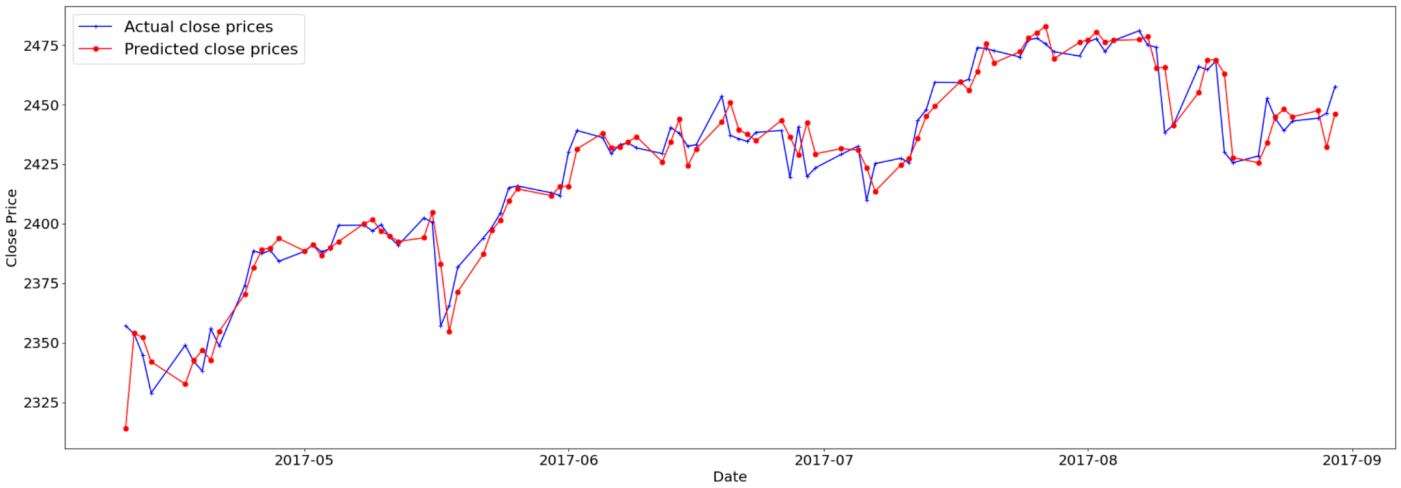
8. Market Microstructure Considerations:

HMMs can be adapted to model market microstructure, capturing latent states that represent different market conditions. This is a nuanced aspect that might be challenging for some other models to capture effectively.

4 Pseudo Code:

```
1. Data Preparation:

Import libraries: numpy, pandas, yfinance
Download S&P 500 data: Use yfinance to download data for "^GSPC" from 2000-01-01 to 2021-08-01
Split data: Divide data into 80% training and 20% testing sets
Augment features: Create new features based on open, high, low, and close prices:
Fractional change between open and close
Fractional change between high and open
Fractional change between open and low
Extract features: Combine these features into a 2D array

2. Model Training:

Create HMM model: Initialize a Gaussian HMM with 10 hidden states
Fit model: Train the model on the augmented features from the training set

3. Prediction Setup:

Define possible outcomes: Create a 3D array of possible fractional changes for each feature
Set prediction parameters: Set number of latent days (50) and days to predict (300)

4. Prediction Loop:

Iterate through prediction days:
Get previous data: Extract features from the recent past (up to 50 days)
Calculate outcome scores: For each possible outcome:
Append the outcome to the previous data
Calculate the model's score for the combined sequence
Find most probable outcome: Select the outcome with the highest score
Predict close price: Multiply the current day's open price by (1 + most probable fractional change)

5. Evaluation:

Plot actual and predicted prices: Visualize the comparison
Calculate and plot error: Analyze mean absolute error (MAE)

6. Parameter Tuning:

Experiment with different hyperparameters:
Number of latent days
Number of hidden states in the HMM
Number of intervals for discretizing features
Evaluate MAE for each combination: Find the best settings
```

5 Sample Implementation:



The plotted figure provides a visual comparison between the actual and predicted close prices of a financial instrument, likely a stock market index. The x-axis represents time, specifically the datetime values corresponding to the index of the 'test_data' DataFrame. The y-axis denotes the close prices.

In future endeavors, refining the predictive capabilities of the model could be achieved through systematic hyperparameter tuning. Focusing on key Hidden Markov Model (HMM) parameters, such as the number of hidden states, and exploring diverse feature sets, including external factors and alternative representations, may offer avenues for improvement. Additionally, investigating optimal training data window sizes, fine-tuning observation space sampling strategies, and employing ensemble methods for combining model predictions could enhance robustness. Rigorous cross-validation techniques and consideration of advanced modeling variants may contribute to mitigating overfitting and improving generalization. Integration of external factors, such as macroeconomic indicators, might further enhance the model's contextual awareness. This comprehensive approach aims to elevate the accuracy and adaptability of the HMM for more effective stock price prediction in dynamic financial markets.