Random Processes with Python Simulations

Applied Stochastic Processes: HW 5

Due Date: November 26, 2024

Objectives

This assignment is designed to help students develop a deep understanding of random processes. Students will enhance their problem-solving skills and gain practical experience through Python-based simulations.

Policy

- You can discuss HW problems with other students, but the work you submit must be written in your own words and not copied from anywhere else. This includes codes.
- However, do write down (at the top of the first page of your HW solutions) the names of all the people with whom you discussed this HW assignment.
- You may decide to write out your solution with pen and paper and use a scanning app to turn in a PDF submission or may choose to type out your solution with LATEX. We strongly encourage the latter.

Expected Topics Covered

1. Kalman Filter and Hidden Markov Models
2. Random Processes and Markov Chains

Submission Guidelines

- Submit your code solutions as a Jupyter notebook file (.ipynb) or as Python scripts (.py). You can also convert them to pdf and attach them to your final submission document. Be sure to indicate which code belongs to what question.
- For those using **LaTeX**, you can paste your code by importing the python environment `pythonhiglights`.

```
\usepackage{pythonhiglights}

\begin{python}
# import necessary libraries
import math
import random

# example arithmetic evaluation
a, b = 2, 3
c = a + b
print(f"The sum of the numbers {a} and {b} is {c}")
\end{python}
```

- Ensure all code is well-documented and includes comments explaining each step.
- Provide a brief report summarizing your findings and the results of your simulations.

Question 1 (80 Points): Kalman Filters and Hidden Markov Models (HMMs) in Time Series Analysis

Hidden Markov Model (HMM)

An **HMM** is a statistical model that represents systems where hidden states influence observable data. The model consists of:

- **Transition Probability Matrix** A:

$$a_{ij} = P(S_{t+1} = j | S_t = i)$$

- **Emission Probability Matrix** B:

$$b_j(o_t) = P(O_t = o_t | S_t = j)$$

- **Initial State Distribution** π:

$$\pi_i = P(S_1 = i)$$

The **Viterbi algorithm** is used to find the most probable sequence of hidden states given the observations.

Viterbi Algorithm Equations

- **Initialization:**

$$\delta_1(i) = \pi_i \cdot b_i(o_1)$$
$$\psi_1(i) = 0$$

- **Recursion:**

$$\delta_t(j) = \max_i [\delta_{t-1}(i) \cdot a_{ij}] \cdot b_j(o_t)$$
$$\psi_t(j) = \arg\max_i [\delta_{t-1}(i) \cdot a_{ij}]$$

- **Termination:**

$$P^* = \max_i [\delta_T(i)]$$
$$q_T^* = \arg\max_i [\delta_T(i)]$$

- **Path Backtracking:**

$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$

Given a dataset containing daily maximum and minimum temperatures and a 'weather' column indicating weather conditions (e.g., rain, drizzle, sun), you are required to:

1. Kalman Filter Analysis (30 Points)

- Implement a Kalman Filter from scratch to smooth the daily average temperature data and make short-term predictions.
- Visualize the filtered estimates along with confidence intervals.
- Use the provided equations for implementation.

2. Hidden Markov Model (HMM) and Viterbi Algorithm (30 Points)

- Implement an HMM from scratch using the 'weather' column as hidden states and 'temp_avg' as observations.
- Implement the Baum-Welch algorithm to estimate the model parameters A, B, and π.
- Implement the Viterbi algorithm to find the most probable sequence of hidden weather states over a given period (e.g., 30 days).
- Visualize the predicted hidden state sequence.

3. Performance Assessment (20 points)

- Assess the performance of the Kalman Filter by comparing the predicted temperatures with actual data using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).
- Assess the performance of the HMM using the accuracy of the predicted hidden state sequence against actual observations or labels.
- Plot the predictions from both models and compare them to the actual temperature data.

Tasks

1. **Data Preparation**
   - Load and preprocess the weather dataset.
   - Encode the 'weather' column for HMM use.
   - Calculate 'temp_avg' as the average of 'temp_max' and 'temp_min'.
2. **Kalman Filter Implementation**
   - Implement the Kalman Filter for temperature estimation using the equations provided.
   - Visualize the results and confidence intervals.
3. **HMM and Viterbi Algorithm Implementation**
   - Implement an HMM with manually defined or estimated A, B, and π using the Baum-Welch algorithm.
   - Implement the Viterbi algorithm to find the most probable hidden state sequence.
   - Visualize the hidden state sequence.
4. **Performance Metrics**
   - Calculate MAE and RMSE for the Kalman Filter predictions.
   - Compare the predicted and actual hidden states for the HMM and measure accuracy.

Deliverables

- **Python Code**
  - Full implementation of the Kalman Filter, HMM, and Viterbi algorithm from scratch.
  - Code for data preprocessing, model training, and prediction.
- **Visualizations**
  - Original temperature data vs. Kalman Filter predictions with confidence intervals.
  - Predicted hidden state sequence from the HMM.
- **Performance Evaluation**
  - Tables showing MAE and RMSE for the Kalman Filter.
  - Comparison of actual and predicted hidden state sequences for the HMM.

Discussion Points

Kalman Filter

- How well did the Kalman Filter smooth the temperature data and predict future temperatures?
- Was the confidence interval effective in depicting the variability?

HMM and Viterbi Algorithm

- How accurately did the HMM predict the hidden weather states?
- What were the limitations of using an HMM for this type of data?

Comparison

- Which model provided better predictive performance for the weather data?
- Under what circumstances might each method be preferable?

Question 3: Markov Chain for Stock Price Prediction (20 points)

Reading: A **Markov chain** is a stochastic process that undergoes transitions from one state to another within a finite set of possible states. Markov chains are memoryless, meaning the probability of moving to the next state depends only on the current state and not on the sequence of events preceding it. This property is known as the ***Markov property***.
If X_t represents the state of the process at time t, then the process $X_1, X_2, \ldots, X_t$ satisfies the Markov property if:

$$P(X_{t+1} = j \mid X_t = i, X_{t-1} = k, \ldots, X_1 = l) = P(X_{t+1} = j \mid X_t = i)$$

Transition Matrix

The transition matrix A describes the probability of transitioning from one state to another:

$$A = \begin{bmatrix} P(S_1 \to S_1) & P(S_1 \to S_2) & \cdots & P(S_1 \to S_n) \\ P(S_2 \to S_1) & P(S_2 \to S_2) & \cdots & P(S_2 \to S_n) \\ \vdots & \vdots & \ddots & \vdots \\ P(S_n \to S_1) & P(S_n \to S_2) & \cdots & P(S_n \to S_n) \end{bmatrix}$$

where $P(S_i \to S_j)$ represents the probability of transitioning from state S_i to state S_j.

Stationary Distribution

A stationary distribution π satisfies:

$$\pi A = \pi$$

and

$$\sum_i \pi_i = 1.$$

This distribution shows the long-term behavior of the Markov chain where the probabilities remain constant over time.

You are tasked with predicting the future stock prices of Apple Inc. (AAPL) using a Markov chain model based on historical stock price data. The primary goal is to utilize daily returns and assign states based on quantiles, allowing a more granular classification of stock price behavior. You will follow these steps to implement this model:

Simulation Guide

1. **Data Acquisition**
   - Use the Yahoo Finance API (`yfinance` library) to download historical daily stock prices for Apple Inc. for the last two years.
2. **Preprocess Data and Assign States**
   - Calculate daily returns as the percentage change in closing prices.
   - Define five states based on the quantiles of daily returns:
     - State 0: "Very Bearish"
     - State 1: "Bearish"
     - State 2: "Neutral"
     - State 3: "Bullish"
     - State 4: "Very Bullish"
   - Assign each day to one of these states based on quantile ranges.
3. **Construct Transition Matrix**
   - Build the transition matrix by counting transitions between states over the historical period and normalizing these counts to get transition probabilities.
4. **Analyze Markov Chain Properties**
   - Visualize the transition matrix as a heatmap.
   - Calculate the stationary distribution of the Markov chain and interpret its meaning regarding long-term stock price behavior.
   - Display the mean return time for each state.
5. **Simulate Future Stock Prices**
   - Create a function to simulate the future stock price path over a 100-day forecast period based on the constructed transition matrix and historical return distributions for each state.
   - Run 1,000 simulations to generate a distribution of potential future price paths.
6. **Backtest Model Predictions**
   - Implement a rolling window approach to simulate backtesting for historical data and compare predicted prices with actual closing prices.
   - Calculate the mean and 95% confidence interval for backtested predictions.
7. **Visualize and Interpret Results**
   - Plot actual historical prices alongside model predictions and the 95% confidence interval for backtesting.
   - Plot simulated future stock prices for the 100-day forecast period and overlay the mean forecast and 95% confidence interval.
8. **Generate Summary Statistics**
   - Calculate relevant metrics including:
     - Current price.
     - Predicted price after 100 days.
     - Average predicted price over the simulation period.
     - Prediction range (minimum and maximum prices).
     - Historical and simulated volatility.
     - 90% and 95% confidence intervals for the final predicted price.
   - Display the summary statistics in a tabular format using the `tabulate` library.

9. **Evaluate Model Performance**

   - Compare the model's simulated average stock prices with actual prices for the last month to assess prediction accuracy.
   - Analyze the reliability and practical insights provided by the Markov chain model.