Time Series: Stochastic Models
Time Series Forecasting
Reinforcement Learning
Neural Networks
Projects
References
Thank You

BSDS-204 : Advanced Machine Learning

Course Instructor - Ms. Manpreet Kaur Bhatia

Gauri Sharan - BSc Data Science, Semester 4

June 11, 2024

Table of contents

1 Time Series: Stochastic Models

2 Time Series Forecasting

3 Reinforcement Learning

4 Neural Networks

5 Projects

6 Thank You

TIME SERIES: STOCHASTIC MODELS
INTRODUCTION TO TIME SERIES
COMPONENTS OF TIME SERIES
DATE-TIME INDEXING
CONCEPT OF HOLIDAYS
RESAMPLING
ROLLING STATISTICS
CONCEPT OF STATIONARITY
TEST FOR STATIONARITY
ACF AND PACF
DECOMPOSITION
DATA EXPLORATION AND CLEANING

WHAT IS A TIME SERIES?

A time series is a sequence of data points measured at regular time intervals.

COMPONENTS OF A TIME SERIES

- Trend
- Seasonality
- Cyclical
- Irregular

BSDS-204 : Advanced Machine Learning
Gauri Sharan
Time Series: Stochastic Models
Introduction to Time Series
Components of Time Series
Date-Time Indexing
Concept of Holidays
Resampling
Rolling Statistics
Concept of Stationarity
Test for Stationarity
ACF and PACF
Decomposition
Data Exploration and Cleaning

Trend
A trend is a long-term pattern or direction in the data.

Seasonality
Seasonality refers to the periodic fluctuations in the data which are seasonal in nature.

Cyclical
Cyclical fluctuations refer to the periodic fluctuations in the data which are cyclic in nature.

Irregular
Irregular fluctuations refer to the random or unpredictable fluctuations in the data.

Time Series: Stochastic Models
Introduction to Time Series
Components of Time Series
Date-Time Indexing
Concept of Holidays
Resampling
Rolling Statistics
Concept of Stationarity
Test for Stationarity
ACF and PACF
Decomposition
Data Exploration and Cleaning

Date-Time Indexing

What is Date-Time Indexing?

Date-time indexing is the process of assigning a unique identifier to each data point based on the date and time it was recorded.

Why is Date-Time Indexing Important?

Date-time indexing is important because it allows us to easily identify and manipulate specific data points based on their date and time.

Concept of Holidays

What are Holidays?
Holidays are special days that are not included in the data.

Why are Holidays Important?
Holidays are important because they can affect the data and need to be accounted for in the analysis.

BSDS-204 : Advanced Machine Learning

Gauri Sharan

Time Series: Stochastic Models

Introduction to Time Series
Components of Time Series
Date-Time Indexing
Concept of Holidays
Resampling
Rolling Statistics
Concept of Stationarity
Test for Stationarity
ACF and PACF
Decomposition
Data Exploration and Cleaning

What is Resampling?

Resampling is the process of reducing the frequency of the data.

Why is Resampling Important?

Resampling is important because it can help to reduce the noise in the data and make it easier to analyze.

Time Series: Stochastic Models
Introduction to Time Series
Components of Time Series
Date-Time Indexing
Concept of Holidays
Resampling
Rolling Statistics
Concept of Stationarity
Test for Stationarity
ACF and PACF
Decomposition
Data Exploration and Cleaning

What are Rolling Statistics?

Rolling statistics are statistics that are calculated over a moving window of data.

Why are Rolling Statistics Important?

Rolling statistics are important because they can help to identify trends and patterns in the data.

BSDS-204 : Advanced Machine Learning

Gauri Sharan

Time Series: Stochastic Models
Introduction to Time Series
Components of Time Series
Date-Time Indexing
Concept of Holidays
Resampling
Rolling Statistics
Concept of Stationarity
Test for Stationarity
ACF and PACF
Decomposition
Data Exploration and Cleaning

What is Stationarity?

Stationarity refers to the property of a time series that its statistical properties remain constant over time.

Why is Stationarity Important?

Stationarity is important because it is a necessary preprocessing condition for many time series analysis techniques.

Test for Stationarity

Time Series: Stochastic Models

Introduction to Time Series

Components of Time Series

Date-Time Indexing

Concept of Holidays

Resampling

Rolling Statistics

Concept of Stationarity

Test for Stationarity

ACF and PACF

Decomposition

Data Exploration and Cleaning

What is a Test for Stationarity?

A test for stationarity is a statistical test that is used to determine whether a time series is stationary or not.

Why is a Test for Stationarity Important?

A test for stationarity is important because it can help to identify whether a time series is suitable for analysis.

ACF and PACF

What are ACF and PACF?

ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) are statistical measures that are used to analyze the autocorrelation of a time series.

Why are ACF and PACF Important?

ACF and PACF are important because they can help to identify the underlying structure of a time series.

BSDS-204 : Advanced Machine Learning
Gauri Sharan
Time Series: Stochastic Models
Introduction to Time Series
Components of Time Series
Date-Time Indexing
Concept of Holidays
Resampling
Rolling Statistics
Concept of Stationarity
Test for Stationarity
ACF and PACF
Decomposition
Data Exploration and Cleaning

What is Decomposition?

Decomposition is the process of breaking down a time series into its component parts.

Why is Decomposition Important?

Decomposition is important because it can help to identify the underlying structure of a time series.

BSDS-204 : Advanced Machine Learning

Gauri Sharan

Time Series: Stochastic Models

Introduction to Time Series
Components of Time Series
Date-Time Indexing
Concept of Holidays
Resampling
Rolling Statistics
Concept of Stationarity
Test for Stationarity
ACF and PACF
Decomposition
Data Exploration and Cleaning

What is Data Exploration?

Data exploration is the process of analyzing and summarizing the data to gain insights.

Why is Data Exploration Important?

Data exploration is important because it can help to identify trends and patterns in the data.

Example code


```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv('data.csv')

# Explore the data
print(data.head())
print(data.info())
print(data.describe())

# Clean the data
data = data.dropna()
data = data.drop_duplicates()
```

EDA of Time Series

What is EDA of Time Series?

EDA (Exploratory Data Analysis) of time series is the process of analyzing and summarizing the data to gain insights.

Why is EDA of Time Series Important?

Through EDA, we can identify significant variables, detect outliers and anomalies, and test foundational assumptions. This approach helps in developing models and determining the best parameters for future predictions.

BSDS-204 : ADVANCED MACHINE LEARNING

GAURI SHARAN

TIME SERIES: STOCHASTIC MODELS

TIME SERIES FORECASTING

EDA OF TIME SERIES

GRAPHICAL REPRESENTATIONS

TRADITIONAL MODELS FOR TIME SERIES FORECASTING

SES, DES, AND TES

REINFORCEMENT LEARNING

NEURAL NETWORKS

WHAT ARE GRAPHICAL REPRESENTATIONS?

Graphical representations are visualizations of the data that can help to identify trends and patterns.

WHY ARE GRAPHICAL REPRESENTATIONS IMPORTANT?

Graphical representations are important because they can help to communicate insights to stakeholders.

BSDS-204 : Advanced Machine Learning

Gauri Sharan

Time Series: Stochastic Models

Time Series Forecasting

EDA of Time Series

Graphical Representations

Traditional Models for Time Series Forecasting

SES, DES, and TES

Reinforcement Learning

Neural Networks

What are Traditional Models?

Traditional models are statistical models that are used to forecast time series data.

Why are Traditional Models Important?

Traditional models are important because they can provide a baseline for comparison with more advanced models.

BSDS-204 : Advanced Machine Learning
Gauri Sharan
Time Series: Stochastic Models
Time Series Forecasting
EDA of Time Series
Graphical Representations
Traditional Models for Time Series Forecasting
SES, DES, and TES
Reinforcement Learning
Neural Networks

What are SES, DES, and TES?

SES (Simple Exponential Smoothing), DES (Double Exponential Smoothing), and TES (Triple Exponential Smoothing) are traditional models for time series forecasting.

Why are SES, DES, and TES Important?

SES, DES, and TES are important because they are widely used methods for forecasting time series data, particularly for modeling trends and seasonality, and are effective in capturing underlying patterns and making accurate predictions.

Example code

```
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv('data.csv')
print(data.head())
print(data.info())
print(data.describe())
data = data.dropna()
data = data.drop_duplicates()

model = SES(data)
forecast = model.forecast()

plt.plot(data)
plt.plot(forecast)
plt.show()
```

BSDS-204 : Advanced Machine Learning
Gauri Sharan
Time Series: Stochastic Models
Time Series Forecasting
Reinforcement Learning
Introduction to Reinforcement Learning
Selection of RL Algorithm
Approaches to Implement Reinforcement Learning
Elements of Reinforcement Learning
Markov Decision Process

What is Reinforcement Learning?

Reinforcement learning is a type of machine learning that involves training an agent to make decisions in an environment.

Why is Reinforcement Learning Important?

Reinforcement learning is important because it can be used to solve complex problems that involve making decisions in an uncertain environment.

WHAT IS THE SELECTION OF RL ALGORITHM?

The selection of RL algorithm is the process of choosing the appropriate algorithm for a given problem.

WHY IS THE SELECTION OF RL ALGORITHM IMPORTANT?

The selection of RL algorithm is important because it can affect the performance of the agent.

BSDS-204 : Advanced Machine Learning

Gauri Sharan

Time Series: Stochastic Models

Time Series Forecasting

Reinforcement Learning

Introduction to Reinforcement Learning

Selection of RL Algorithm

Approaches to Implement Reinforcement Learning

Elements of Reinforcement Learning

Markov Decision Process

What are Approaches to Implement Reinforcement Learning?

Approaches to implement reinforcement learning include model-based and model-free methods.

BSDS-204 : Advanced Machine Learning

Gauri Sharan

Time Series: Stochastic Models

Time Series Forecasting

Reinforcement Learning

Introduction to Reinforcement Learning

Selection of RL Algorithm

Approaches to Implement Reinforcement Learning

Elements of Reinforcement Learning

Markov Decision Process

What are Elements of Reinforcement Learning?

Elements of reinforcement learning include the agent, environment, and actions.

Markov Decision Process

What is Markov Decision Process?

Markov decision process is a mathematical framework for modeling decision-making problems.

Why is Markov Decision Process Important?

Markov decision process is important because it can be used to model complex decision-making problems.

Optimal Policy

What is Optimal Policy?

Optimal policy is a policy that maximizes the expected cumulative reward.

Why is Optimal Policy Important?

Optimal policy is important because it maximizes the expected discounted return, ensuring that the policy chosen is the best possible given the constraints and objectives of the system being managed.

What is Bellman Equation?

Bellman equation is a mathematical equation that is used to calculate the optimal policy.

Why is Bellman Equation Important?

it provides a necessary condition for optimality in dynamic programming, allowing for the calculation of the optimal value function and policy in Markov decision processes (MDPs).

BSDS-204 : Advanced Machine Learning

Gauri Sharan

Time Series: Stochastic Models

Time Series Forecasting

Reinforcement Learning

Introduction to Reinforcement Learning

Selection of RL Algorithm

Approaches to Implement Reinforcement Learning

Elements of Reinforcement Learning

Markov Decision Process

What is Temporal Differencing?

Temporal differencing is a technique that is used to estimate the value function.

Why is Temporal Differencing Important?

Temporal differencing is important because it can be used to estimate the value function.

Q-Learning

What is Q-Learning?

Q-learning is a type of reinforcement learning that is used to learn the optimal policy.

Why is Q-Learning Important?

Q-learning is important because it provides a model-free approach to reinforcement learning, allowing agents to learn optimal policies without prior knowledge of the environment. This flexibility and adaptability make Q-learning a valuable tool for optimizing decision-making processes in various fields

Exploration vs. Exploitation

Time Series: Stochastic Models
Time Series Forecasting
Reinforcement Learning
Introduction to Reinforcement Learning
Selection of RL Algorithm
Approaches to Implement Reinforcement Learning
Elements of Reinforcement Learning
Markov Decision Process

What is Exploration vs. Exploitation?

Exploration vs. exploitation is a trade-off that is used to balance the exploration of new actions and the exploitation of known actions.

SARSA

What is SARSA?

SARSA is a type of reinforcement learning that is used to learn the optimal policy.

Why is SARSA Important?

SARSA is important because it can handle stochastic and dynamic environments. SARSA is also simple to implement and can be used with different exploration strategies, making it a versatile tool for solving sequential decision-making problems.

Frozen Gym Lake Environment

Overview

The Frozen Gym Lake Environment is a classic problem in reinforcement learning.

Goal

The goal is to navigate from the starting point to the goal point.

Gym Library

Overview

The Gym library is a collection of environments for reinforcement learning.

Key Features

- Provides a variety of environments.
- Supports multiple reinforcement learning algorithms.

Approach

Use SARSA to learn the optimal policy.

```
import gym
env = gym.make('FrozenLake-v0')
# Initialize Q-table and policy
Q = np.zeros([env.observation_space.n, env.action_space.n])
policy = np.zeros([env.observation_space.n, env.action_space.n])
# Train using SARSA
for episode in range(1000):
    state = env.reset()
    done = False
    while not done:
        action = np.argmax(policy[state])
        next_state, reward, done, _ = env.step(action)
        Q[state, action] = Q[state, action] + 0.1 *
        (reward + 0.9 * np.max(Q[next_state]) - Q[state, action])
        policy[state] = np.argmax(Q[state])
```

BSDS-204 : Advanced Machine Learning
Gauri Sharan
Time Series: Stochastic Models
Time Series Forecasting
Reinforcement Learning
Neural Networks
Fundamentals of Artificial Neural Networks
Advanced Concepts in Artificial Neural Networks
Understanding Activation Functions in Neural Networks

Definition

Artificial Neural Networks (ANNs) are computational models inspired by the structure and function of the human brain.

Key Components

- Neurons (nodes)
- Connections (edges)
- Weights
- Bias

BSDS-204 : Advanced Machine Learning

Gauri Sharan

Time Series: Stochastic Models

Time Series Forecasting

Reinforcement Learning

Neural Networks

Fundamentals of Artificial Neural Networks

Advanced Concepts in Artificial Neural Networks

Understanding Activation Functions in Neural Networks

Regularization

Techniques to prevent overfitting.

Optimization Algorithms

- Stochastic Gradient Descent (SGD)
- Adam
- RMSProp

Understanding Activation Functions in Neural Networks

Definition

Activation functions introduce non-linearity into the neural network.

Common Activation Functions

- Sigmoid
- ReLU (Rectified Linear Unit)
- Tanh (Hyperbolic Tangent)

Loss Function

Definition

The loss function measures the difference between the predicted output and the actual output.

Common Loss Functions

- Mean Squared Error (MSE)
- Cross-Entropy

Perceptron:- Single Layered and Multiple Layered

Single Layer Perceptron

A single layer neural network with a linear activation function.

Multi-Layer Perceptron

A neural network with multiple layers, allowing for more complex representations.

Steps to Formulate ANN Algorithm

Step 1: Data Preparation
Preprocess the data.

Step 2: Model Definition
Define the neural network architecture.

Step 3: Training
Train the model using the training data.

Step 4: Evaluation
Evaluate the model using the testing data.

Building a Single Neuron Neural Network from Scratch in Python


```python
class Neuron:
    def __init__(self, inputs, bias):
        self.weights = [random.random() for _ in range(inputs)]
        self.bias = bias

    def forward(self, inputs):
        output = sum([x * y for x, y in zip(inputs, self.weights)]) + self.bias
        return self.sigmoid(output)

    def sigmoid(self, x):
        return 1 / (1 + math.exp(-x))

neuron = Neuron(2, 1)
inputs = [0, 0]
output = neuron.forward(inputs)
print(output)
```

BSDS-204 : Advanced Machine Learning
Gauri Sharan
Time Series: Stochastic Models
Time Series Forecasting
Reinforcement Learning
Neural Networks
Projects
Models of Artificial Neural Network
Practical Implementation of Neural Network Training in Python
Implementation of

Feedforward Neural Networks

Information flows only in one direction.

Recurrent Neural Networks (RNNs)

Information can flow in a loop.

Practical Implementation of Neural Network Training in Python

```
import numpy as np
# Define the neural network
class NeuralNetwork:
    def __init__(self, inputs, hidden, outputs):
        self.weights1 = np.random.rand(inputs, hidden)
        self.weights2 = np.random.rand(hidden, outputs)

    def forward(self, inputs):
        hidden_layer = np.maximum(np.dot(inputs, self.weights1), 0)
        output_layer = np.dot(hidden_layer, self.weights2)
        return output_layer
# Train the neural network
nn = NeuralNetwork(2, 2, 1)
inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
outputs = np.array([[0], [1], [1], [0]])
for _ in range(1000):
    output = nn.forward(inputs)
    error = outputs - output
    # Update weights
```

Implementation of Artificial Neural Network for AND Logic Gate


```
import numpy as np
# Define the neural network
class NeuralNetwork:
    def __init__(self, inputs, hidden, outputs):
        self.weights1 = np.random.rand(inputs, hidden)
        self.weights2 = np.random.rand(hidden, outputs)

    def forward(self, inputs):
        hidden_layer = np.maximum(np.dot(inputs, self.weights1), 0)
        output_layer = np.dot(hidden_layer, self.weights2)
        return output_layer
# Train the neural network
nn = NeuralNetwork(2, 2, 1)
inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
outputs = np.array([[0], [0], [0], [1]])
for _ in range(1000):
    output = nn.forward(inputs)
    error = outputs - output
    # Update weights
```

Implementation of Artificial Neural Network for OR Logic Gate


```
import numpy as np
# Define the neural network
class NeuralNetwork:
    def __init__(self, inputs, hidden, outputs):
        self.weights1 = np.random.rand(inputs, hidden)
        self.weights2 = np.random.rand(hidden, outputs)

    def forward(self, inputs):
        hidden_layer = np.maximum(np.dot(inputs, self.weights1), 0)
        output_layer = np.dot(hidden_layer, self.weights2)
        return output_layer
# Train the neural network
nn = NeuralNetwork(2, 2, 1)
inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
outputs = np.array([[0], [1], [1], [1]])
for _ in range(1000):
    output = nn.forward(inputs)
    error = outputs - output
    # Update weights
```

Time Series: Stochastic Models
Time Series Forecasting
Reinforcement Learning
Neural Networks
Projects
Models of Artificial Neural Network
Practical Implementation of Neural Network Training in Python
Implementation of

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Load data
data = pd.read_csv('data.csv')

# Prepare data
X = data.drop(['target'], axis=1)
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create LSTM model
model = Sequential()
model.add(LSTM(50, input_shape=(X.shape[1], 1)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam') #continued in the next slide
```

Time Series Project - Future Prediction

```
# Train model
model.fit(X_train, y_train, epochs=100, batch_size=32, verbose=2)

# Evaluate model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'MSE: {mse}')
```

References

- Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer, 2016.
- George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley, 2015.
- James D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- François Chollet. *Deep Learning with Python*. Manning Publications, 2017.
- Charu C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer, 2018.

Thank You

Hope you liked this presentation.

Gauri Sharan
Student, School of Data Science
AAFT Noida (Shobhit University)
BSc Data Science 2022-25
Semester 4, 2024

- LinkedIn: **linkedin.com/in/gauri-sharan**
- GitHub: **github.com/gaurisharan**
- Mail: **gaurisharan123@gmail.com**