**SSA (Singular Spectrum Analysis)**

Singular Spectrum Analysis (SSA) is a powerful technique for time series analysis, especially useful for decomposing and forecasting complex, non-stationary signals. Here’s a simplified breakdown of how SSA works:

* **Embedding**: SSA starts by creating a matrix from the original time series. This involves selecting a window length, ***L***, and constructing a **"trajectory matrix"** by sliding the window across the series, which captures overlapping lagged segments of the data. Each row (or column) represents a shifted version of the series within the window.
* **Singular Value Decomposition (SVD)**: The trajectory matrix is then decomposed using SVD, which splits it into several **"components"** by identifying patterns in the data. This process breaks down the original signal into components that capture trends, oscillations, and noise.
* **Grouping**: After SVD, the resulting components are grouped. This step separates meaningful components (e.g., trends, seasonality) from random noise, which can then be reconstructed separately.
* **Reconstruction**: In this step, the grouped components are summed to reconstruct a version of the original series with enhanced features, like a denoised version or one highlighting a trend or seasonality.

SSA is widely applied for smoothing, trend analysis, and seasonal decomposition, making it a flexible preprocessing tool for time series forecasting models like LSTM. By isolating important patterns and removing noise, SSA can enhance forecasting accuracy, especially in complex, fluctuating data.

The math behind Singular Spectrum Analysis (SSA). The main steps are embedding, performing Singular Value Decomposition (SVD), grouping, and reconstructing.

**1. Embedding**

Given a time series , we start by creating a matrix, known as the **trajectory matrix**, by choosing a window length *L* (where ). The trajectory matrix, **X**, is constructed so that each column contains a segment of the time series:

Where . Each column in **X** is a "lagged" vector of the time series, with rows and *K* columns.

**2. Singular Value Decomposition (SVD)**

Next, we perform **SVD** on the trajectory matrix **X**:

Here:

* is the rank of **X**,
* are the eigenvalues (singular values squared) of **X**,
* are the left singular vectors (representing the “spatial” structure),
* are the right singular vectors (representing the “temporal” structure).

Each component ​ represents a distinct pattern or oscillation within the data, and the strength of each pattern is given by ​.

**3. Grouping**

In the grouping step, we select which components to keep by grouping eigenvalues or singular values that represent meaningful structures in the time series (like trends or periodic signals). The grouping can be manual or automatic based on criteria like eigenvalue magnitude or variance.

If we choose to retain, say, the first ***r*** components, we sum them as follows:

**4. Reconstruction**

Finally, we reconstruct the time series by summing up the selected components. Each column of the trajectory matrix can be averaged to obtain the reconstructed time series values, using diagonal averaging. For a matrix **Y**, the reconstructed series is given by averaging elements along each diagonal.

The reconstructed time series captures the key patterns or trends that were isolated in the grouping step, while filtering out noise or unwanted components. This makes it useful for smoothing or decomposing the time series for further analysis or forecasting.

**Summary**

In essence, SSA:

* Breaks down a time series into simpler components using SVD.
* Groups components that capture meaningful patterns.
* Reconstructs a cleaner time series with isolated trends and seasonality, filtering out noise.

SSA provides a way to analyze complex time series by separating signal from noise and decomposing trends, which is helpful for visualization, smoothing, and forecasting.

Python Implementation:

import numpy as np

import matplotlib.pyplot as plt

from scipy.linalg import svd

# Step 1: Generate synthetic time series data

np.random.seed(42)

n = 100  # Length of time series

trend = np.linspace(0, 1, n)

seasonality = 0.5 \* np.sin(2 \* np.pi \* np.arange(n) / 12)

noise = 0.1 \* np.random.randn(n)

time\_series = trend + seasonality + noise

# Plot the original time series

plt.plot(time\_series, label='Original Time Series')

plt.legend()

plt.show()

# Step 2: Embedding

window\_length = 20

K = len(time\_series) - window\_length + 1

X = np.array([time\_series[i:i + window\_length] for i in range(K)]).T  # Create trajectory matrix

# Step 3: Perform SVD on the trajectory matrix

U, s, VT = svd(X)

S = np.diag(s)

# Step 4: Reconstruct components by selecting SVD components

# Reconstruct trend and seasonality components

X\_trend = U[:, 0:1] @ S[0:1, 0:1] @ VT[0:1, :]

X\_seasonal = U[:, 1:3] @ S[1:3, 1:3] @ VT[1:3, :]

# Step 5: Diagonal averaging function to reconstruct the time series

def diagonal\_averaging(X):

    n, K = X.shape

    TS = np.zeros(n + K - 1)

    for k in range(n + K - 1):

        count = 0

        for m in range(max(0, k - K + 1), min(n, k + 1)):

            TS[k] += X[m, k - m]

            count += 1

        TS[k] /= count

    return TS

# Apply diagonal averaging to get trend and seasonal components

trend\_component = diagonal\_averaging(X\_trend)

seasonal\_component = diagonal\_averaging(X\_seasonal)

# Plot the decomposed components

plt.plot(time\_series, label='Original Time Series')

plt.plot(trend\_component, label='Trend Component')

plt.plot(seasonal\_component, label='Seasonal Component')

plt.legend()

plt.show()

