State-Space Models: Unravelling Hidden Dynamics in Data

P. R. Vekariya, Prity Kumari, Y. A. Garde

State-space models are a strong framework in statistics for unraveling complicated and hidden processes in various datasets. These models offer a versatile way to identify temporal or sequential data, making them useful tools in disciplines ranging from economics to engineering to biology and beyond. Let us investigate the complexities of state-space models, learning how they reveal expose hidden variables, improve forecasting, and revolutionize our understanding of dynamic systems.

❖ Unveiling the State-Space Concept

A state-space model's fundamental function is to depict how a system changes over time. The two essential elements that define its evolution are the "state" and the "observation." While the observation relates to the quantifiable results we can immediately perceive, the state represents the underlying, unobservable variables that determine the system's behavior. These models resemble a secret story that is being played out in the background, a story that state-space models aim to reveal.

A state space model (SSM) is a time series model in which the time series Y_t is interpreted as the result of a noisy observation of a stochastic process X_t. The values of the variables X_t and Y_t can be continuous (scalar or vector) or discrete. SSMs belong to the realm of Bayesian inference, and they have been successfully applied in many fields to solve a broad range of problems. It is usually assumed that the state process X_t is Markovian, *i.e.*, X_t depends on the history only through X_{t-1}, and Y_t depends only on X_t:

$$X_t \sim \mathrm{p}(X_t|X_{t-1})$$

$$Y_t \sim \mathrm{p}(Y_t|Y_t)$$

❖ Components & Structures

1. **State Variables:**

   State variables are an array of internal variables that describe a dynamic system's existing state. They represent the fundamental elements of the system, which are critical for predicting how it will behave over time but cannot be observed directly. A vector, often written as X(t), is used to represent state variables, where "t" stands for the current time step. The scale of the system being modeled determines how many state variables are needed.

2. **Observation / Measurement Variables:**

   The measurable quantities or system outputs are known as observation variables. Unlike state variables, observation variables can be seen or measured immediately. They are typically denoted by a vector, typically written as Y(t), where "t" stands for the current time step. Depending on the information provided by the system in terms of measurements, the number of observation variables may change.

3. **Control Variables:**

   Control variables are applied to a system as external inputs or control signals to modify its behavior. These inputs can be regulated or optimized by an outside agent to maintain system performance. Control variables are commonly expressed as a vector, u(t), where "t" stands for the current time step.

4. **System Dynamics Equations:**

   The state variable evolution over time is described by the system dynamics equations. They stand in for the fundamental rules or precepts that direct how the system

behaves. According to the needs of the modeling, these equations are often differential equations, either in continuous-time or discrete-time form.

5. **Observation Equations:**

The state variables and the observation variables are related by the observation equations. They specify the mapping between the system's quantifiable outputs and the state variables. The measurement noise and uncertainties are also taken into consideration using observation equations.

❖ **Types of State Space Models**

There are two types of state space models (SSMs), depending on the linearity of their state and observation equations:

1. **Linear State Space Models (LSSMs)**

When the state and observation equations are written as linear functions of the state variables and observations, the resulting model is known as a linear state space model (LSSM).

a. State Equation (State Transition Model):

$$X_t = A_t * X_{t-1} + B_t * u_t + w_t$$

Where,

X_t : At time t, the state vector represents the system's hidden or unobservable variables

A_t : A state transition matrix is a matrix that connects the state at time t to the state at time t-1. It captures the dynamics of the system

X_{t-1} : State vector at time t-1

B_t : At time t, the control input matrix accounts for any external control or effect on the system

u_t : Control input vector at time t

w_t : Process noise represents the uncertainty or random fluctuations in the state transition process

b. Observation Equation:

$$Y_t = C_t * X_t + v_t$$

Where,

Y_t : At time t, the observation vector represents the system's measured or observed variables

C_t : The observation matrix maps the state vector to the observation space. It expresses how the states are related to the observable quantities

v_t : Observation noise, which accounts for measurement errors and uncertainty in the observed data

Key characteristics:

- The linearity of the state and observation equations leads to closed-form solutions and effective computing.
- LSSMs often assume Gaussian processes and observation noise, simplifying estimation and enabling the application of Kalman filters and smoothers.
- The most well-studied SSM is the Kalman filter, for which the processes above are linear and the sources of randomness are Gaussian. Namely, a linear state space model has the form:

$$X_{t+1} = GX_t + \varepsilon_{t+1}$$
$$Y_t = HX_t + \eta_t$$

- Here, the state vector $X_t R^r$ is possibly unobservable and it can be observed only through the observation vector $Y_t \in R^n$.

- The matrices $G \in MAT_r(R)$ and $H \in MAT_{n,r}(R)$ are assumed to be known. For example, their values may be given by (economic) theory, or they may have been obtained through MLE estimation.
- In fact, the matrices G and H may depend deterministically on time, *i.e.*, G and H may be replaced by known matrices G_t and H_t, respectively.
- We also assume that the distribution of the initial value X1 is known and Gaussian. The vectors of residuals $\varepsilon_t \in R^r$ and $\eta_t \in R^n$ satisfy

$$\mathrm{E}(\varepsilon_\iota \varepsilon_S^T) = \delta_{ts} Q,$$
$$\mathrm{E}(\eta_\iota \eta_S^T) = \delta_{ts} R,$$

- Where δ_{ts} denotes Kronecker's delta, and where Q and R are known positive definite (covariance) matrices.
- We also assume that the components of ε_ι and η_s are independent of each other for all t and s. The matrices Q and R may depend deterministically on time.
- The first of the equations is called the state equation, while the second one is referred to as the observation equation. Let T denote the time horizon.
- Our broad goal is to make inferences about the statescaps X_t based on a set of observations $Y_1, \ldots, Y_t$.

2. Nonlinear State Space Models

Nonlinear State Space Models (NSSMs) express the state equation, observation equation, or both as nonlinear functions of state variables and observations.

a. Nonlinear State Equation (State Transition Model):

$$X_t = f(X_{t-1}, u_t) + w_t$$

Where,

X_t : At time t, the state vector represents the system's hidden or unobservable variables

f : The nonlinear state transition function describes how the state at time t is affected by the state at time t-1 and any control inputs u_t

X_{t-1} : State vector at time t-1

u_t : Control input vector at time t

w_t : Process noise represents the uncertainty or random fluctuations in the state transition process

b. Nonlinear Observation Equation:

$$Y_t = h(X_t) + v_t$$

Where,

Y_t : At time t, the observation vector represents the system's measured or observed variables

H : A nonlinear observation function maps the state vector to the observation space. It describes how the states are related to the observable quantities

v_t : Observation noise, which accounts for measurement errors and uncertainty in the observed data

Key characteristics:

- The model is more powerful and able to handle complex system interactions since at least one of the state or observation equations incorporates nonlinear functions.
- When estimating the states and parameters of NSSMs, numerical techniques like the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), or Particle Filter (PF) are frequently used.

- To estimate the posterior distribution of states in NSSMs, particle filters are frequently used. This enables more accurate inference in non-Gaussian and highly nonlinear environments.
- NSSMs can model more real-world systems than LSSMs since linearity assumptions do not restrict them.

```
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.kalman_filter import KalmanFilter
from statsmodels.tsa.statespace import tools

# Generate synthetic data
np.random.seed(0)
n_samples = 100
true_values = np.sin(np.linspace(0, 4 * np.pi, n_samples))
noisy_values = true_values + np.random.normal(0, 0.5, n_samples)

# Define the state space model
class SinStateSpace(KalmanFilter):
    def __init__(self, endog):
        super().__init__(k_states=2, k_obs=1)

        self['design', 0, 0] = 1.0
        self['transition', 0, 0] = 1.0
        self['transition', 0, 1] = 1.0
        self['selection', 0, 0] = 1.0
        self['obs_intercept', 0, 0] = 0.0
        self.initialize_known([0.0, 0.0], [[1.0, 0.0], [0.0, 1.0]])
        self.loglikelihood_burn = 1

# Create the state space model
model = SinStateSpace(noisy_values)
# Fit the model
results = model.smooth(noisy_values)

# Plot the true values, noisy measurements, and smoothed estimates
plt.plot(true_values, label='True Values')
plt.plot(noisy_values, label='Noisy Measurements')
plt.plot(results.filtered_state[0], label='Smoothed Estimates')
plt.xlabel('Time Step')
plt.ylabel('Value')
plt.title('State Space Model Example')
plt.legend()
plt.show()
```

Kalman Filter:

- By predicting a joint probability distribution over the variables for each period of time, Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm used in statistics and control theory that uses a number of evaluations observed over time, such as statistical noise and other inaccuracies, to produce estimates of unknown variables that are typically more precise than those based on a single measurement alone.

- The filter is named after Rudolf E. Kálmán, who was one of the primary developers of its theory.
- A system's state can be estimated more accurately using Kalman filtering than it would by using just one measurement by using the dynamic model of the system (such as physical laws of motion), known control inputs to the system, and multiple sequential measurements from sensors. It is a typical sensor fusion and data fusion algorithm as a result.
- The accuracy of determining the system's state is constrained by a variety of reasons, including noisy sensor data, approximations in the equations that describe the system's evolution, and unaccounted-for external factors.
- The uncertainty resulting from noisy sensor data and, to some extent, from unpredictable external influences is successfully handled by the Kalman filter. By averaging the system's anticipated state and the current measurement using a weighted average, the Kalman filter generates an estimate of the system's state.
- Values with greater (i.e., smaller) estimated uncertainty are "trusted" more for the purpose of the weights. The weights are determined using the covariance, which is a metric for the predicted level of uncertainty in state prediction for the system.
- A new state estimate that fits between the expected and measured states and has a better-estimated uncertainty than each alone is the outcome of the weighted average.
- Every time step, this process is repeated, with the updated estimate and its covariance influencing the prediction utilised in the subsequent iteration.
- As a result, the Kalman filter operates recursively and calculates a new state using only the most recent "best guess" rather than the whole history of the system's state.
- Important factors to take into account include the measures' current-state estimation and certainty grading. The gain of the Kalman filter is frequently used to describe the filter's response.
- The Kalman gain, which can be "tuned" to obtain a certain performance, is the weight given to the measurements and current-state estimation.
- With a high gain, the filter gives the most recent findings more weight and responds to them more quickly. The filter more closely resembles the model predictions when the gain is low.
- At the extremes, a high gain near one will provide an estimated trajectory that jumps about more, whereas a low gain near zero will level out noise but reduce responsiveness.
- The state estimate and covariances are coded into matrices due to the various dimensions required in a single set of computations when carrying out the actual calculations for the filter (as detailed below).
- In any of the transition models or covariances, this enables the modeling of linear relationships between several state variables (such as location, velocity, and acceleration). Here's a basic example of how to implement a Kalman filter using Python and the filterpy library:

```
import numpy as np
from filterpy.kalman import KalmanFilter
import matplotlib.pyplot as plt

# Generate some synthetic data
np.random.seed(0)
n_samples = 100
true_values = np.linspace(0.1, 10.0, n_samples)
noisy_values = true_values + np.random.normal(0, 1, n_samples)

# Initialize the Kalman filter
kf = KalmanFilter(dim_x=2, dim_z=1)
kf.x = np.array([0.0, 1.0])  # Initial state [position, velocity]
kf.F = np.array([[1.0, 1.0], [0.0, 1.0]])  # State transition matrix
kf.H = np.array([[1.0, 0.0]])  # Measurement matrix
kf.P *= 1000.0  # Initial state covariance
kf.R = 1.0  # Measurement noise covariance
kf.Q = np.array([[0.001, 0.001], [0.001, 0.001]])  # Process noise covariance

# Store the filtered results
filtered_states = []
# Kalman filtering
for z in noisy_values:
    kf.predict()
    kf.update(z)
    filtered_states.append(kf.x[0])  # Estimated position

# Plot the true values, noisy measurements, and filtered estimates
plt.plot(true_values, label='True Values')
plt.plot(noisy_values, label='Noisy Measurements')
plt.plot(filtered_states, label='Filtered Estimates')
plt.xlabel('Time Step')
plt.ylabel('Value')
plt.title('Kalman Filter Example')
plt.legend()
plt.show()
```

- Before running the code, make sure you have the filterpy library installed in your Python environment:
  pip install filterpy

Applications of State Space Model

i. **Control Engineering:**

SSMs are used by dynamic systems to simulate and predict behavior, allowing for the efficient implementation of control and feedback mechanisms. In fields like process control, robotics, and aerospace, they are essential.

ii. **Finance and Economics:**

Models using state spaces are crucial for simulating financial time series, asset pricing, and economic variables. Their use is necessary for risk management and

portfolio optimization. The ability to predict stock prices, interest rates, and economic indicators is another benefit.

iii. Time Series Analysis:

State space models are useful for studying time-varying data, such as variations in temperature, traffic patterns, and economic trends. They include prediction, computation of missing variables, and the discovery of underlying patterns.

iv. Signal Processing

SSMs are essential for generating trustworthy and beneficial information from unstable signals and observations in applications involving signal processing. Systems for communication, speech recognition, and image processing all make use of them.

v. Ecology and Environmental Studies:

State Space Models (SSMs) are used by scientists to study ecological systems, animal populations, and environmental variables. They aid in the analysis of species interaction dynamics, ecosystem modeling, and climate change.

vi. Health and Medicine:

Researchers utilize state space models to forecast disease transmission, optimize medication doses, and examine patient health trajectories in the fields of epidemiology, pharmacokinetics, and disease modeling.

vii. Robotics and Autonomous Systems:

SSMs are used by researchers in robotics for behavior strategy, mapping, and localization. They enable robots to locate themselves and navigate hazy and dynamic situations.

viii. Economics and Finance:

SSMs forecast economic indicators, model financial time series, and forecast asset values for use in economic analysis and decision-making.