International Journal of Computer Vision and Signal Processing, 15(1), 1-8(2025) ORIGINAL ARTICLE

On Variational Bayes Approach to State Space Modeling with Its Implementation for Simple Mathematical Models

Norikazu Ikoma

Dept. of Electrical and Information Engineering, Faculty of Fundamental Engineering, Nippon Institute of Technology, Japan

Abstract

Deep Markov Model, which would be called as "Deep Kalman filters", as well as "Structured Inference Networks", models structure behind time series data by employing nonlinear mapping of neural network for system model and observation model within state space modeling framework. So obtained hidden state estimate becomes distributed representation within neural network that leads to difficulty for interpretation of its meanings. This work begins with applying simple mathematical models to the framework of Deep Markov Model in order to address the above issues. Its implementation employs PyTorch based framework "Pyro" within programming language Python in demonstrative examples of numerical experiment. The most simple state space model is so-called trend model has been implement within the framework and parameters have been estimated via variational Bayes in the numerical experiment.
Contribution of the Paper: Bridge Deep Markov Model and mathematical state space models.

Keywords: state space model, state estimation, deep Markov model, variational Bayes, implementation

IJCVSP
International Journal of Computer Vision and Signal Processing

ISSN: 2186-1390 (Online)
http://cennser.org/IJCVSP

Article History:
Received: 1 Jan. 2025
Revised: 12 July 2025
Accepted: 20 July 2025
Published Online: 22 Nov. 2025

1. Introduction

State space modeling approaches [1] [2] to real world's phenomena by formulating them into dynamical systems, which have time-varying feature essentially [3] [4] [5] [6]. The formulation specifies 1) hidden state, 2) time evolution of the hidden state, and 3) observation process to get measurement derived from the hidden state. The hidden state governs the dynamical system with its time evolution, then, measurement is by-product of the hidden state. Because a measurement of single time step contains only partial information of the hidden state, we need to utilize time series models for estimating the hidden state.

The above three elements are represented in a state space model typically as 1) state vector, 2) system model / equation, and 3) observation model / equation according to traditional mathematical formulation. State estimation techniques provide solution to the state space model for given series of measurements. Among such techniques for state estimation, Kalman filter [7] [8] is exact solution for linear and Gaussian case, while particle filters [9] [10] are approximate solutions for general case of non-linear and/or non-Gaussian. Those techniques are derived under assumption of given state space model.

To match the assumption of given state space model, human design of the model in mathematical form is typical, where small number of unknown parameters are involved in the model. Then, the unknown parameters are tuned with observed measurement data under maximum likelihood criterion or Bayesian framework to get posterior of the parameters, as investigated in, say, [11] [12] [13]. Thus, human design is dominant in this approach of mathematical modeling.

Recent advances of deep neural network based approach allows more flexible state space modeling as following instance. Deep Markov Model, which would be called as Deep Kalman filters [14], as well as Structured Inference Networks [15], models structure behind time series data by employing nonlinear mapping of neural network for system model and observation model within state space modeling framework. More researches, such as in [16] [17] [18] are following along with this direction. Variational Bayes approach allows to tune the parameters of the neural networks for given measurement data without being suffered from difficulty to estimate of hidden state and unknown parameters simultaneously. Price to pay is to design ap-

proximate posterior distribution.

So obtained hidden state estimate becomes distributed representation within neural network that leads to difficulty for interpretation of its meanings. In order to circumvent this difficulty, simple mathematical models have been examined in the same framework of variational Bayes. So obtained results of the experimental analysis provide some insight how the framework performs. Note that some researches try to circumvent this difficulty, e.g., by introducing bridge to physical model [16].

Our work begins with applying simple mathematical models to the framework of Deep Markov Model in order to address the above issues. Its implementation employs PyTorch based framework "Pyro" [19] of programming language Python in demonstrative examples of numerical experiment.

2. Model

State space model, state estimation, and fixed parameter estimation are summarized according to literature, say [1] [2].

2.1. State Space Model

Hidden state at discrete time k and its series beginning from $k = 0$ and up to k are respectively denoted by

$$\mathbf{x}_k \in \mathbb{R}^n, \quad \mathbf{x}_{0:k} \equiv \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k\}. \tag{1}$$

Observation at time k and its series beginning from $k = 1$ and up to k are respectively denoted by

$$\mathbf{y}_k \in \mathbb{R}^m \quad \mathbf{y}_{1:k} \equiv \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}. \tag{2}$$

Initial distribution, i.e. probability distribution of the hidden state at $k = 0$, is denoted, with its parameter $_0$, by

$$p_0(\mathbf{x}_0; _0). \tag{3}$$

Time evolution of the hidden state is governed by system model having parameter $_f$ such that

$$f(\mathbf{x}_k|\mathbf{x}_{k-1}; _f) \tag{4}$$

where it implies that Markov property $p(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k-1}) = f(\mathbf{x}_k|\mathbf{x}_{k-1}; _f)$ holds.

Observation, i.e., measurement, is governed by observation model having parameter $_h$ such that

$$h(\mathbf{y}_k|\mathbf{x}_{k-1}; _h) \tag{5}$$

where it implies that for $K > k$ conditional independent property $p(\mathbf{y}_k|\mathbf{x}_{0:K}, \mathbf{y}_{1:k-1}, \mathbf{y}_{k+1:K}) = h(\mathbf{y}_k|\mathbf{x}_{k-1}; _h)$ holds .

All the parameters appearing in above three distributions are all together in

$$\equiv \{_0, _f, _h\}. \tag{6}$$

2.2. Formal Solution to State Estimation

State estimation is to get posterior distribution of hidden state for given series of observation. There are three major categories of the state estimation, filtering, prediction, and smoothing, depending on the relation between time step of hidden state $\mathbf{x}_k$ and observation series $\mathbf{y}_{1:k+L}$. That is, $L = 0$ is filtering, $L < 0$ is prediction, and $L > 0$ is smoothing,

One-step-ahead prediction is obtained by convolution

$$p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \int f(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})d\mathbf{x}_{k-1}. \tag{7}$$

where filtering at time $k-1$ is combined with system model.

Filtering is obtained by Bayes rule

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \frac{h(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})} \tag{8}$$

where one-step-ahead prediction is used as prior combined with likelihood as observation model.

Denominator of eq.(8) is utilized to evaluate likelihood as described later. It is obtained by integration of the numerator

$$p(\mathbf{y}_k|\mathbf{y}_{1:k-1}) = \int h(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1})d\mathbf{x}_k. \tag{9}$$

Smoothing for fixed interval of measurements from $k = 1$ to $k = K$ can be derived as

$$p(\mathbf{x}_k|\mathbf{y}_{1:K}) = p(\mathbf{x}_k|\mathbf{y}_{1:k}) \int p(\mathbf{x}_{k+1}|\mathbf{y}_{1:K}) \frac{f(\mathbf{x}_{k+1}|\mathbf{x}_k)}{p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k})} d\mathbf{x}_{k+1} \tag{10}$$

where the backward recursion consists of filtering $p(\mathbf{x}_k|\mathbf{y}_{1:k})$, smoothing $p(\mathbf{x}_{k+1}|\mathbf{y}_{1:K})$ and prediction $p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k})$ at time $k + 1$, combined with system model.

Joint posterior of the hidden state series can be obtained recursively by

$$p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) = p(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1}) \frac{h(\mathbf{y}_k|\mathbf{x}_k)f(\mathbf{x}_k|\mathbf{x}_{k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})}. \tag{11}$$

2.3. Fixed Parameter Estimation

Fixed parameters, collectively represented as shown in eq.(6), are necessary to be estimated beforehand conducting state estimation task according to the formal solutions in eqs (7), (8), (10) and (11) . There are two major approaches for the fixed parameter estimation, maximum likelihood estimation and Bayesian approach.

Maximum likelihood estimation $\hat{}_{\mathrm{ML}}$ is obtained by maximizing likelihood for fixed interval of measurements from $k = 1$ to $k = K$

$$p(\mathbf{y}_{1:K};) = \prod_{k=1}^{K} p(\mathbf{y}_k|\mathbf{y}_{1:k-1};), \tag{12}$$

i.e., $\hat{}_{\mathrm{ML}} = \arg\max p(\mathbf{y}_{1:K};)$. Note that right-hand-side of eq.(12) comes from eq.(9), and it can be evaluated through state estimation task for a given instance of .

Bayesian approach is based on joint posterior of the hidden state series $\mathbf{x}_{0:K}$ and the fixed parameters

$$p(\mathbf{x}_{0:K}, |\mathbf{y}_{1:K}) \propto p(\mathbf{y}_{1:K}|\mathbf{x}_{0:K},)p(\mathbf{x}_{0:K},). \tag{13}$$

By marginalizing eq.(13) with respect to the hidden state series

$$p(|\mathbf{y}_{1:K}) = \int p(\mathbf{x}_{0:K}, |\mathbf{y}_{1:K}) d\mathbf{x}_{0:K} \tag{14}$$

and maximizing eq.(14) with respect to the fixed parameters , we can obtain MAP (Maximum A Posteriori) estimation of the fixed parameters, i.e.,

$$\hat{}_{\text{MAP}} = \arg\max p(|\mathbf{y}_{1:K}). \tag{15}$$

2.4. Variational Bayes as Approximate State Estimation

Joint posterior distribution of the hidden state series, appeared in eq.(11), can be re-written as

$$\begin{aligned} p(\mathbf{x}_{0:K}|\mathbf{y}_{1:K};) &\propto p(\mathbf{x}_0) \prod_{k=1}^{K} f(\mathbf{x}_k|\mathbf{x}_{k-1}) h(\mathbf{y}_k|\mathbf{x}_k) \\ &= p(\mathbf{x}_0) \prod_{k=1}^{K} f(\mathbf{x}_k|\mathbf{x}_{k-1}) \times \prod_{k=1}^{K} h(\mathbf{y}_k|\mathbf{x}_k) \\ &\equiv \mathcal{F}(\mathbf{x}_{0:K}) \times \mathcal{H}(\mathbf{y}_{1:K}|\mathbf{x}_{1:K}) \\ &\equiv \mathcal{P}(\mathbf{y}, \mathbf{x}). \end{aligned}$$

In the above, we have denoted $\mathbf{y} \equiv \mathbf{y}_{1:K}$, $\mathbf{x} \equiv \mathbf{x}_{0:K}$, and $\mathcal{P}(\mathbf{x}|\mathbf{y}) \equiv p(\mathbf{x}_{0:K}|\mathbf{y}_{1:K};)$, where

$$\mathcal{P}(\mathbf{y}, \mathbf{x}) = \mathcal{P}(\mathbf{x}|\mathbf{y})\mathcal{P}(\mathbf{y})$$

and

$$= \int \mathcal{P}(\mathbf{y}, \mathbf{x}) d\mathbf{x}.$$

Also, we simply denote

$$\mathcal{F}_{_{0,f}}(\mathbf{x}) \equiv \mathcal{F}(\mathbf{x}_{0:K})$$

and

$$\mathcal{H}_{_h}(\mathbf{y}|\mathbf{x}^+) \equiv \mathcal{H}(\mathbf{y}_{1:K}|\mathbf{x}_{1:K})$$

with $\mathbf{x}^+ \equiv \mathbf{x}_{1:K}$ for later use.

Variational approximate posterior distribution $\mathcal{Q}(\mathbf{x}|\mathbf{y})$ has been introduced from now on. The degree of approximation can be evaluated by KL(Kullback-Leibler) divergence of $\mathcal{P}$ with respect to $\mathcal{Q}$, where the KL divergence is defined by

$$\begin{aligned} \text{KL}\left\langle \mathcal{Q}(\mathbf{x}) \| \mathcal{P}(\mathbf{x}) \right\rangle &\equiv \int \mathcal{Q}(\mathbf{x}) \log \frac{\mathcal{Q}(\mathbf{x})}{\mathcal{P}(\mathbf{x})} d\mathbf{x} \\ &= \mathbb{E}_{\mathcal{Q}} \left[\log \frac{\mathcal{Q}(\mathbf{x})}{\mathcal{P}(\mathbf{x})} \right]. \end{aligned} \tag{16}$$

We can derive a relation between log of marginal likelihood in eq.(12), which is also called log of evidence, and the KL divergence as

$$\log p(\mathbf{y};) = \mathcal{L}(\mathbf{y}; \mathcal{P}, \mathcal{Q}) + \text{KL}\left\langle \mathcal{Q} \| \mathcal{P} \right\rangle \tag{17}$$

where $\mathcal{L}(\cdot)$ is ELBO (Evidence Lower BOund) as defined

$$\begin{aligned} \mathcal{L}(\mathbf{y}; \mathcal{P}, \mathcal{Q}) &\equiv \mathbb{E}_{\mathcal{Q}} \left[\log \frac{\mathcal{P}(\mathbf{y}, \mathbf{x})}{\mathcal{Q}(\mathbf{x}|\mathbf{y})} \right] \\ &= \mathbb{E}_{\mathcal{Q}} \left[\log \frac{\mathcal{H}_{_h}(\mathbf{y}|\mathbf{x}^+) \mathcal{F}_{_{0,f}}(\mathbf{x})}{\mathcal{Q}(\mathbf{x}|\mathbf{y})} \right] \\ &= \mathbb{E}_{\mathcal{Q}} \left[\log \mathcal{H}_{_h}(\mathbf{y}|\mathbf{x}^+) \right] - \text{KL}\left\langle \mathcal{Q}(\mathbf{x}|\mathbf{y}) \| \mathcal{F}_{_{0,f}}(\mathbf{x}) \right\rangle. \end{aligned} \tag{18}$$

According to eq.(17) with fixed measurement $\mathbf{y} \equiv \mathbf{y}_{1:K}$, maximizing ELBO of eq.(18) leads to minimizing KL divergence of $\mathcal{P}$ with respect to $\mathcal{Q}$ since left-hand-side of eq.(17) is constant for fixed measurement $\mathbf{y}$.

Thus, parameter of variational approximate posterior distribution $\mathcal{Q}(\mathbf{x}|\mathbf{y})$ can be learned by maximizing the ELBO in eq.(18) with gradient ascend for small $\epsilon_\Psi > 0$

$$_{(t)} = _{(t-1)} + \epsilon_\Psi \nabla \mathcal{L}(\mathbf{y}; \mathcal{P}, \mathcal{Q}_{_{(t-1)}}). \tag{19}$$

3. Deep Markov Model

Deep Markov Model, which would be called as Deep Kalman filters [14], Structured Inference Networks [15] as well, utilizes neural networks for system model eq.(4) and observation model eq.(5) in order describe nonlinear feature of the target system, such as real world's phenomena.

Specifically, initial distribution of eq.(3) is in parametric form such as Gaussian distribution $\mathcal{N}(\mathbf{x}_0; \mathbf{m}_0, \mathbf{Q}_0)$ with parameters $_0 = \{\mathbf{m}_0, \mathbf{Q}_0\}$ where $\mathbf{m}_0$ is mean vector and $\mathbf{Q}_0$ is covariance matrix. Note that dimension d of the state $\mathbf{x}_k$ for $k \geq 0$ is one of the design parameters to be given by human decision.

System model of eq.(4) is Gaussian distribution

$$\mathcal{N}(\mathbf{x}_k; \mathbf{m}(\mathbf{x}_{k-1}; \mathbf{W}_\alpha), \mathbf{Q}(\mathbf{x}_{k-1}; \mathbf{W}_\beta))$$

with neural networks $\mathbf{m}$ and $\mathbf{Q}$ having weight parameters $\mathbf{W}_\alpha$ and $\mathbf{W}_\beta$, respectively. Thus, the parameter of system model is $_f = \{\mathbf{W}_\alpha, \mathbf{W}_\beta\}$.

Observation model of eq.(5) is parametric distribution $\mathcal{P}(\mathbf{y}_k; \mathbf{h}(\mathbf{x}_k; \mathbf{W}_\gamma))$, such as Gaussian distribution for continuous values, Bernoulli distribution for categorical values, etc. Here, the parameter $\mathbf{h}(\mathbf{x}_k; \mathbf{W}_\gamma)$ of those distributions is output of neural network having weight parameter $\mathbf{W}_\gamma$. Thus, the parameter of observation model is $_h = \mathbf{W}_\gamma$.

Whole the parameter, except the design parameter d, of the state space model in Deep Markov Model form is

$$= \{\mathbf{W}_\alpha, \mathbf{W}_\beta, \mathbf{W}_\gamma\}. \tag{20}$$

According to literature of Deep Markov Model [14] [15], the parameter of eq.(20) can be optimized by maximizing the ELBO in eq.(18) with gradient ascend for small $\epsilon_\Theta > 0$

$$_{(t)} = _{(t-1)} + \epsilon_\Theta \nabla \mathcal{L}(\mathbf{y}; \mathcal{P}_{_{(t-1)}}, \mathcal{Q}). \tag{21}$$

Design of variational approximate posterior distribution $\mathcal{Q}(\mathbf{x}|\mathbf{y})$ for Deep Markov Model is crucial to achieve sufficient performance to the objective of analysis. Recurrent type of neural networks, such as LSTM(Long Short

Term Memory) or GRU(Gated Recurrent Unit), bidirectional in terms of time index have been examined in the literature [14] [15]. Refer them for more details.

One important feature of Deep Markov Model is hidden state representation consisting of d-dimensional Gaussian distribution. This is a kind of distributed representation of information coded by the neural networks in use. One significant drawback is its difficulty to interpret the meaning, despite of its flexible nonlinearity and learning ability.

4. Experiment with Simple Mathematical Models

Variational Bayes approach to state space modeling and state estimation explained above has been examined here with simple mathematical model. The most simple state space model is so-called trend model

$$\left\{ \begin{array}{lcll} x_0 & \sim & p_0(x\,;_0) & \\ x_k & = & x_{k-1} + v_k & , \quad v_k \sim q(v\,;_f) \\ y_k & = & x_k + w_k & , \quad w_k \sim r(w\,;_h) \end{array} \right. \tag{22}$$

which is a linear model having scalar state x_k and observation y_k. When initial distribution p_0, system noise distribution q, and observation noise distribution r are all Gaussian, state estimation can be conducted by Kalman filter [7] [8]. Otherwise, it is necessary to use approximate filter for non-Gaussian posterior state such as particle filters [9] [10].

4.1. Implementation in "Pyro"

To implement the variational Bayes inference in programming language Python, we have employed PyTorch based framework "Pyro" [19]. It provides simple implementation method with two functions named "`model`" and "`guide`" corresponding to state space model and variational approximate distribution, respectively.

4.1.1. Implementation of function "`model`"

An implementation of the function "`model`" for the trend model in eq.(22) with Gaussian distributions for p_0, q, and r has been shown in Fig. 1. Where, initial distribution of Gaussian having parameter $_0 = \{\mu_0, \sigma_0\}$ with a constraint $\sigma_0 > 0$ is subject to infer in variational Bayes framework with initial values $\mu_0 = 0.0$ and $\sigma_0 = 0.1$ at the first five lines of the function in the code of Fig. 1. Here, hidden state at time $k = 0$ is described by `pyro.sample` at the fifth line.

System noise and observation noise of the trend model are both zero centered Gaussian with variance paramters $_f = \{\tau^2\}$ and $_h = \{\sigma^2\}$, respectively. Initial values of those parameters are $\tau = 0.5$ and $\sigma = 0.1$ written at the following four lines in the code of Fig. 1.

Following `for` loop in the function "`model`" evaluates the first term $\mathbb{E}_{\mathcal{Q}}\left[\log \mathcal{H}_h(\mathbf{y}|\mathbf{x}^+)\right]$ of ELBO in eq.(18), where observed measurements $\mathbf{y} \equiv y_{1:K}$ are given as the argument `obs_y` of the the function "`model`". The evaluation with Gaussian distribution has been conducted by `pyro.sample` code having `obs=` argument in the loop.

Hidden state series $\mathbf{x}^+ \equiv x_{1:K}$ are generated by following the system model, which is the second equation in eq.(22), with Gaussian distribution evaluated by `pyro.sample` code not having `obs=` argument in the loop.

All the hidden state in series $\mathbf{x} \equiv x_{0:K}$, which includes initial time step $k = 0$, have correspondence to function "`guide`". This correspondence is guaranteed by the first argument of `pyro.sample` function with the identical strings having time index k governed by `'x_%d'%(i+1)`.

```
import pyro.distributions as dist

def model(obs_y):
    m_0_p = pyro.param('m_0_p', torch.tensor(0.0) ) # m_0_p = 0.0
    s_0_p = pyro.param('s_0_p', torch.tensor(0.1),  # s_0_p = 0.1
                constraint=constraints.positive)    # s_0_p > 0
    x0_prior = dist.Normal(m_0_p, s_0_p)
    x_prev = pyro.sample('x_%d'%0, x0_prior)

    tau_p = pyro.param('tau_p',torch.tensor(0.5),   # tau_p = 0.5
                constraint=constraints.positive)    # tau_p > 0
    sig_p = pyro.param('sig_p',torch.tensor(0.1),   # sig_p = 0.1
                constraint=constraints.positive)    # sig_p > 0

    for i in range( len(obs_y) ):
        x = pyro.sample('x_%d'%(i+1), dist.Normal(x_prev, tau_p))
        pyro.sample("obs_y_%d"%i, dist.Normal(x, sig_p),
            obs=obs_y[i])
        x_prev = x
```

Figure 1: Function `model` for trend model.

4.1.2. Choices on variational approximate distribution

As a variational approximate distribution $\mathcal{Q}(\mathbf{x}|\mathbf{y})$, we have several choices to design it. First choice is crude proposal having time evolution of the hidden state by system model's equation in eq.(22) without referring observed measurements $\mathbf{y}$, thus, $\mathcal{Q}(\mathbf{x}|\mathbf{y}) = \mathcal{Q}(\mathbf{x})$ and

$$\mathcal{Q}(\mathbf{x}) = q_0(x_0\,;_0) \prod_{k=1}^{K} f(x_k|x_{k-1}\,;_f) \tag{23}$$

where initial distribution $q_0(x_0\,;_0)$ is Gaussian distribution $N(\mu_\Psi, \sigma_\Psi^2)$ having parameters $_0 = \{\mu_\Psi, \sigma_\Psi\}$ with constraint $\sigma_\Psi > 0$, and system noise is Gaussian of zero centered having variance parameter τ_Ψ^2 in $_f = \{\tau_\Psi\}$.

Second choice is improved proposal from the first choice having variational approximate distribution $\mathcal{Q}(\mathbf{x}|\mathbf{y})$ to incorporate observation measurement $\mathbf{y}$.

$$\mathcal{Q}(\mathbf{x}|\mathbf{y}) = q_0(x_0\,;_0) \prod_{k=1}^{K} q(x_k|x_{k-1}, y_k\,;_q) \tag{24}$$

where initial distribution $q_0(x_0\,;_0)$ is Gaussian with parameter $_0 = \{\mu_\Psi, \sigma_\Psi\}$ with constraint $\sigma_\Psi > 0$ as same

as the case in eq.(23). Rest part of eq.(24) consists of $q(x_k|x_{k-1}, y_k\ ;_q)$, which is

$$x_k = (1-\alpha)x_{k-1} + \alpha y_k + v_k,\ v_k \sim N(0, \tau_q^2) \qquad (25)$$

where it has parameters $_q = \{\alpha, \tau_q\}$ consisting of variance τ_q^2 of Gaussian distribution and weight-coefficient α.

Third choice is more improved proposal from the second choice, which is exact estimation of the hidden state according to Kalman filter (KF) algorithm.

4.1.3. Implementation of function "guide"

An implementation of the first choice of variational approximate distribution $\mathcal{Q}$ for the trend model in eq.(22) into a function "guide" has been shown in Fig.2. Where, parameters of the initial distribution, $_f = \{\tau_\Psi\}$, and system noise variance parameter, $_f = \{\tau_\Psi\}$, are subject to learn under variational Bayes approach, as they are decleared in `pyro.param` with given initial value at the beginning three lines and at the two lines before the `for` loop in the function.

The `for` loop in the function "guide" evaluates the second term $\mathrm{KL}\left\langle \mathcal{Q}(\mathbf{x}|\mathbf{y}) \| \mathcal{F}_{0,f}(\mathbf{x}) \right\rangle$ of ELBO in eq.(18), where observed measurements $\mathbf{y} \equiv y_{1:K}$ are given but not substantially used (just used for counting the length of series).

The correspondences of each hidden state variable is guaranteed by the first argument of each `pyro.sample` function call with the identical strings `'x_%d'%(i+1)`.

```
import torch.distributions.constraints as constraints

def guide(obs_y):
    m_0_q = pyro.param('m_0_q', torch.tensor(0.0) ) # m_0_q = 0.0
    s_0_q = pyro.param('s_0_q', torch.tensor(0.1),    # s_0_q = 0.1
                constraint=constraints.positive)     # s_0_q > 0
    x0_posterior = dist.Normal( m_0_q, s_0_q )
    x_prev = pyro.sample('x_%d'%0, x0_posterior)

    tau_q = pyro.param('tau_q', torch.tensor(0.5),# tau_q = 0.5
                constraint=constraints.positive) # tau_q > 0

    for i in range( len(obs_y) ):
        x = pyro.sample('x_%d'%(i+1), dist.Normal(x_prev, tau_q))
        x_prev = x
```

Figure 2: Function of crude `guide` for trend model.

Implementation of function "guide" for the second choice has been shown in Fig.3.

Implementation of function "guide" for the third choice has been shown in Fig.4.

4.1.4. Implementation of main procedure

Main procedure consists of a loop of learning and data generating function, which are given in Fig.5 and Fig.6, respectively. Where, in Fig.5, instance of `SVI` has been created as a name `svi` by giving the two functions "`model`" and "`guide`", as well as optimizer such as Adam and loss function as ELBO in eq.(eq.(18), before entering the learning loop of 2,000 iterations. As the conditions of synthetic

```
def guide(obs_y):
    m_0_q = pyro.param('m_0_q', torch.tensor(0.0) ) # m_0_q = 0.0
    s_0_q = pyro.param('s_0_q', torch.tensor(0.1),    # s_0_q = 0.1
                constraint=constraints.positive)     # s_0_q > 0
    x0_posterior = dist.Normal( m_0_q, s_0_q )
    x_prev = pyro.sample('x_%d'%0, x0_posterior)

    tau_q = pyro.param('tau_q', torch.tensor(0.5),# tau_q = 0.5
                constraint=constraints.positive) # tau_q > 0
    Kgain = pyro.param('Kgain', torch.tensor(0.75), #Kgain = 0.75
        constraint=constraints.interval(0.0,1.0) ) # 0 < Kgain < 1

    for i in range( len(obs_y) ):
        x_comb = (1-Kgain)*x_prev + Kgain*obs_y[i]
        x = pyro.sample('x_%d'%(i+1), dist.Normal(x_comb, tau_q))
        x_prev = x
```

Figure 3: Function of improved `guide` for trend model.

```
def guide(obs_y):
    m_0_q = pyro.param('m_0_q',torch.tensor(0.0) ) #  = 0.0
    s_0_q = pyro.param('s_0_q',torch.tensor(0.1),   #  = 0.1
                constraint=constraints.positive)    #  > 0
    tau_q = pyro.param('tau_q',torch.tensor(0.5),#  = 0.5
                constraint=constraints.positive) #  > 0
    sig_q = pyro.param('sig_q',torch.tensor(0.1),#  = 0.1
                constraint=constraints.positive) #  > 0
    m_prev = m_0_q
    s_prev = s_0_q
    m_list = [ m_prev ]
    s_list = [ s_prev ]
    for i in range( len(obs_y) ):
        V_p = s_prev**2 + tau_q**2
        Kgain = V_p / ( V_p + sig_q**2 )
        m_curr = m_prev + Kgain * ( obs_y[i] - m_prev )
        V_f = ( 1 - Kgain ) * V_p
        s_curr = torch.sqrt( V_f )
        m_list.append( m_curr )
        s_list.append( s_curr )
        m_prev, s_prev = m_curr, s_curr

    N = len(obs_y)
    k = N
    x = pyro.sample('xs_%d'%(k),dist.Normal( m_prev, s_prev ))

    for i in range( N ):
        k = N-1 - i
        m_f = m_list[k]
        V_f = s_list[k]**2
        m_p_1 = m_f
        V_p_1 = V_f + tau_q**2
        A = V_f / V_p_1
        m_s_1 = m_prev
        V_s_1 = s_prev**2
        m_s = m_f + A * ( m_s_1 - m_p_1 )
        V_s = V_f + A * ( V_s_1 - V_p_1 ) * A
        m_curr = m_s
        s_curr = torch.sqrt( V_s )
        x = pyro.sample('xs_%d'%(k),dist.Normal( m_curr, s_curr ))
        m_prev, s_prev = m_curr, s_curr
```

Figure 4: Function of more improved `guide` for trend model.

data generation, length of series is $K = 100$, initial hidden value is according to $N(1.5, 0.01^2)$, and variance parameters are $\tau^2 = 0.1^2$ and $\sigma^2 = 0.01^2$, as implemented in Fig.6.

4.2. Results of variational Bayes estimation of parameters

4.2.1. Result of first choice : crude proposal

Results of learning have been shown as follows. At first, loss function is shown in Fig. 7. We can see that the value of loss function fluctuates dramatically and generally tends to decrease. It is almost stable after 1,500 iterations.

Estimated results of parameters are shown in Fig. 10, in which both mathematical symbols and variable names in Python are written for each panel. Where, true value is shown in dashed thin line with blue color. Note on `guide`, the true value is displayed as a reference only.

```
from pyro.optim import SGD, Adam
from pyro.infer import SVI, Trace_ELBO

adam_params = {"lr":0.01 }
optimizer = Adam(adam_params)
svi = SVI(model, guide, optimizer, loss=Trace_ELBO())

history = { 'loss': [], 'm_0_q': [], 's_0_q': [], 'tau_q': [],
  # 'Kgain': [], # use this line if necessary
   'm_0_p': [], 's_0_p': [], 'tau_p': [], 'sig_p': [],  }

NUM_STEPS = 2000
for step in range(1, NUM_STEPS + 1):
    loss = svi.step(obs)
    for key in history.keys():
        if key=='loss':
            history['loss'].append(loss)
        else:
            history[key].append(pyro.param(key).item())
    x0 = s_0_true * torch.randn(1) + m_0_true
    obs = gen_data(N,x0, tau_true,sig_true)
```

Figure 5: Main loop of learning process.

```
import torch
pyro.set_rng_seed(0)

def gen_data(_N,_x_prev,_tau,_sig):
    _obs = torch.zeros(_N)
    for i in range(_N):
        x = _x_prev + _tau*torch.randn(1)
        y = x + _sig*torch.randn(1)
        _obs[i] = y
        _x_prev = x
    return _obs

N = 100
m_0_true = 1.5
s_0_true = 0.01
tau_true = 0.1
sig_true = 0.01

x0 = s_0_true * torch.randn(1) + m_0_true
obs = gen_data(N,x0, tau_true,sig_true)
```

Figure 6: Synthetic data generation for trend model.

We can see that means of initial distribution in (a),(b) and systen noise variances in (f),(g) are converging to the true values for both `model` and `guide`, while variances of initial ditribution in (c),(d) for both `model` and `guide` and observation noise variance in (e) for `model` is not converging to the true values.

4.2.2. Result of second choice : improved proposal

Results of learning have been shown as follows. At first, loss function is shown in Fig. 8. We can see that the value of loss is smaller and more stable than the first case in Fig.7. Also, it almost stable after 1,500 iterations.

Estimated results of parameters are shown in Fig. 11 in the same manner of Fig. 10. Where, new parameter, weight-coefficient α, has been added in (f) without showing its true value, which is unknown.

We can see that means of initial distribution in (a),(b) for both `model` and `guide` and systen noise variances in (g) for `model` are converging to the true values, while systen noise variance in (h) for `guide` is underestimated. Observation noise variance in (e) for `model` is not reaching to the true values within 2,000 iterations. Generally, more stable results than the simple "`guide`" function have been obtained, while variances of initial ditribution in (c),(d) vary more.

4.2.3. Result of third choice : exact proposal with KF

Results are shown as follows. At first, loss function is shown in Fig. 9. We can see that the value of loss is much smaller and more stable than the previous two cases in Fig.7 and Fig. 8. It has been stabilized quickly, before 500-th iteration.

Estimated results of parameters are shown in Fig. 12 in the same manner of Fig. 10 and Fig. 11. Where, new parameter, observation noise variance for `guide`, has been shown in (f) in place of α in Fig. 11.

We can see that mean of initial distribution in (a) for `model` and systen noise variances in (g),(h) for both `model` and `guide` are converging to the true values, while mean of initial distribution in (a) for `model` and observation noise variance in (e),(f) for both `model` and `guide` are not reaching to the true values within 2,000 iterations.

Much more stable results than the previous two cases in Fig. 10 and Fig. 11 have been obtained in general. However, we recognize that variances of initial ditribution in (c),(d) are converging to value for unknown reasons.

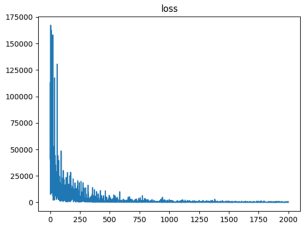


Figure 7: Loss of crude proposal for trend model.

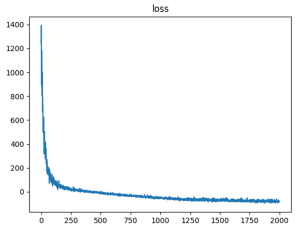


Figure 8: Loss of improved proposal for trend model.

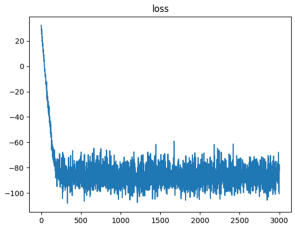


Figure 9: Loss of more improved proposal for trend model.

5. Concluding Remarks

After reviewing the state space model and state estimation formulation, methods for estimating fixed parameters in state space models are formulated as maximum likelihood estimation, Bayesian estimation, and variational inference. Then, Deep Markov Model, as one of the latest approaches for flexisible state space modeling, has been summarized together with the state space model formulation.

Implemenation of simple mathematical state space model has been examined in Python programming language under PyTorch based framework "Pyro". Numerical experiments for learning the model parameters based on variational inference have been conducted and loss function and parameters' estimation plots are shown as the experimental results.

Future works include experiments with more examples of simple state space models, such as, non-Gaussian extension for system noise and/or observation noise in the trend model, nonlinear model, target tracking model, decomposition model, and so on [20]. More complicated models, such as visual/radar tracking models in a manner of Track-Before-Detect, such as in [21] [22], are interesting to be examined. Also, connection and comparison with latest methods such as in [23] [24] [25] is important direction for the future works.

References

[1] Branko Ristic, Sanjeev Arulampalam, Neil Gordon, Beyond the Kalman Filter: Particle Filters for Tracking Applications, Artech House, 2004.

[2] Andreas Svensson, Thomas B. Schön, A flexible state–space model for learning nonlinear dynamical systems, Automatica, Vol.80, pp.189-199, 2017.

[3] Liangyi Lyu, Lei Lu, Hanjie Chen, David A. Clifton, Yuanting Zhang, Tapabrata Chakraborti, An improved deep regression model with state space reconstruction for continuous blood pressure estimation, Computers and Electrical Engineering, Volume 118, Part A, 109319, 2024.

[4] Yonggu Lee, Gyul Lee, Austin White, Yong-June Shin, Oscillation Parameter Estimation via State-Space Modeling of Synchrophasors, IEEE Trans. on Power Systems, Volume: 39, Issue: 3, pp.5219 - 5228, 2024.

[5] Junyi Shi, Tomasz Piotr Kucner, Learning State-Space Models for Mapping Spatial Motion Patterns, arXiv preprint, arXiv:2309.00333, 2023.

[6] Toru Yano, State Space Model of Realized Volatility under the Existence of Dependent Market Microstructure Noise, arXiv preprint, arXiv:2408.17187, 2024.

[7] R. E. Kalman, A New Approach to Linear Filtering and Prediction Problems, Journal of Basic Engineering, 82(1): 35-45, 1960.

[8] Miroslav Plašil, State Space Estimation: from Kalman Filter Back to Least Squares, Statistika, 103(2): 235-245, 2023.

[9] A.Doucet, N.de Freitas, and N.J.Gordon (eds), "Sequential Monte Carlo Methods in Practice", New York, Springer, 2001.

[10] M.S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking IEEE Trans. on Signal Processing, Vol.50, Issue 2, pp.174 - 188, 2002.

[11] Christophe Andrieu, Arnaud Doucet, Roman Holenstein, Particle Markov chain Monte Carlo methods, Journal of the Royal Statistical Society: Series B, Volume 72, Issue 3 pp.269-342, 2010.

[12] Jaafar AlMutawa, Parameter estimation of multisensor state space models with outlier contamination, 10th International Conference on Control, Decision and Information Technologies (CoDIT), pp.2146-2151, 2024.

[13] Yuxiong Gao, Wentao Li, Rong Chen, Parameter Estimation of State Space Models Using Particle Importance Sampling, arXiv preprint, arXiv:2502.00904, 2025.

[14] R.G.Krishnan, U.Shalit, D.Sontag, "Deep Kalman Filters", arXiv preprint, arXiv:1511.05121, 2015.

[15] R.G.Krishnan, U.Shalit, D.Sontag, "Structured Inference Networks for Nonlinear State Space Models", Proc. of 31st AAAI Conference on Artificial Intelligence (AAAI-17), pp.2101-2109, 2017.

[16] Wei Liu, Zhilu Lai, Kiran Bacsa, Eleni Chatzi, Physics-guided Deep Markov Models for learning nonlinear dynamical systems with uncertainty Mechanical Systems and Signal Processing Vol.178, 109276, 2022.

[17] Yuhao Liu, Marzieh Ajirak, Petar Djuric, Sequential Estimation of Gaussian Process-based Deep State-Space Models, arXiv preprint, arXiv:2301.12528, 2023.

[18] Raunaq Bhirangi, Chenyu Wang, Venkatesh Pattabiraman, Carmel Majidi, Abhinav Gupta, Tess Hellebrekers, Lerrel Pinto, Hierarchical State Space Models for Continuous Sequence-to-Sequence Modeling, arXiv preprint, arXiv:2402.10211, 2024.

[19] Pyro Deep Markov Model, `https://pyro.ai/examples/dmm.html`

[20] Genshiro Kitagawa, Introduction to Time Series Modeling with Applications in R, second edition, Chapman and Hall, 2020.

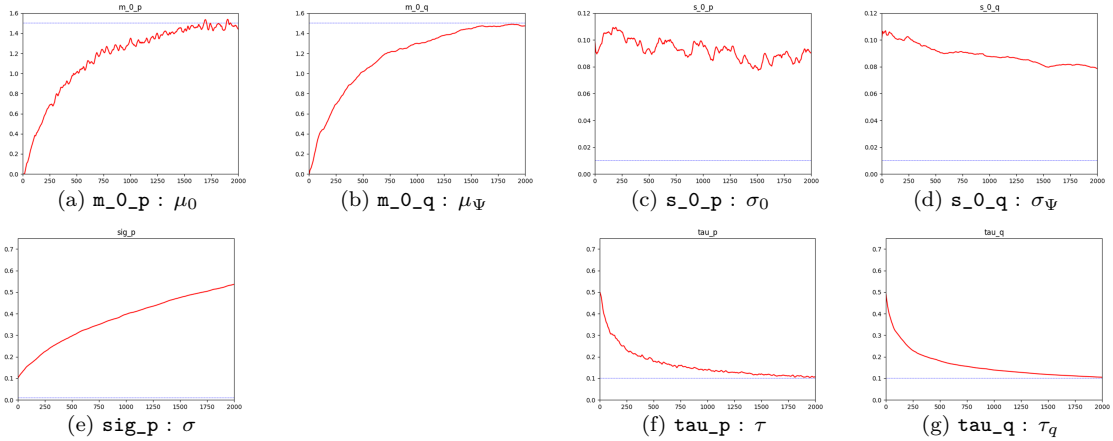
[21] Chaozhu Zhang, Lin Li, Yu Wang, A Particle Filter Track-Before-Detect Algorithm Based on Hybrid Differential Evolution, Algorithms, 8(4), 965-981, 2015.

[22] Audrey Cuillery, François Le Gland, Track-before-detect on radar image observation with an adaptive auxiliary particle filter 22th International Conference on Information Fusion (Fusion 2019), 8 pages, 2019

[23] Anubhab Ghosh, Mohamed Abdalmoaty, Saikat Chatterjee, Håkan Hjalmarsson, DeepBayes—An estimator for parameter estimation in stochastic nonlinear dynamical models, Automatica, Volume 159, 111327, 2024.

[24] José Francisco Lima, Fernanda Catarina Pereira, Arminda Manuela Gonçalves, Marco Costa, Bootstrapping State-Space Models: Distribution-Free Estimation in View of Prediction and Forecasting, 6(1), pp.36-54, 2024.

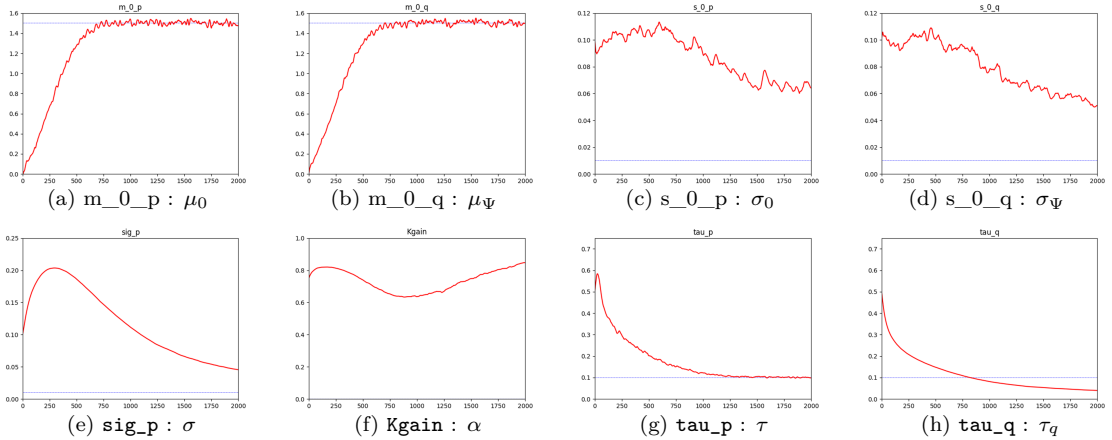
[25] Erik Chinellato, Fabio Marcuzzi State, parameters and hidden dynamics estimation with the Deep Kalman Filter: Regularization strategies, Journal of Computational Science, Volume 87, 2025.



(a) `m_0_p` : μ_0 (b) `m_0_q` : μ_Ψ (c) `s_0_p` : σ_0 (d) `s_0_q` : σ_Ψ

(e) `sig_p` : σ (f) `tau_p` : τ (g) `tau_q` : τ_q

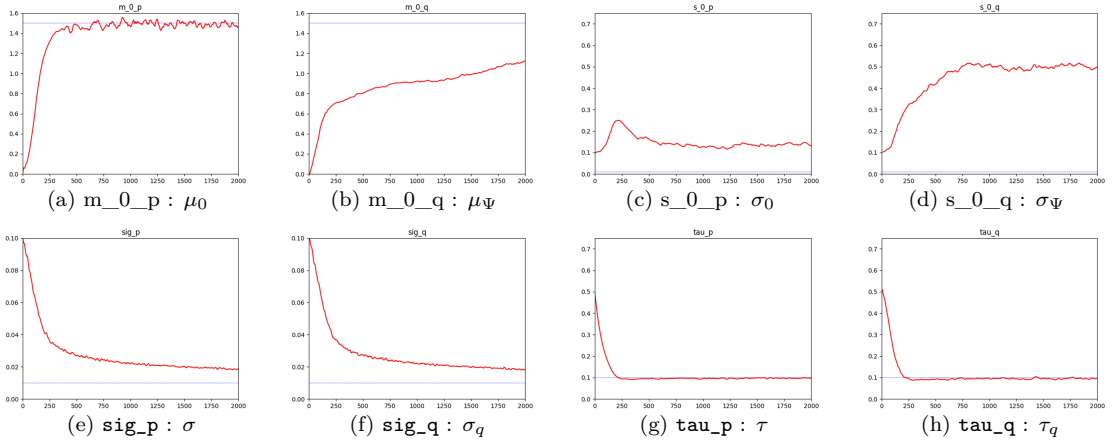
Figure 10: Estimation results of crude proposal for trend model.



(a) m_0_p : μ_0 (b) m_0_q : μ_Ψ (c) s_0_p : σ_0 (d) s_0_q : σ_Ψ

(e) `sig_p` : σ (f) `Kgain` : α (g) `tau_p` : τ (h) `tau_q` : τ_q

Figure 11: Estimation results of improved proposal for trend model.



(a) m_0_p : μ_0 (b) m_0_q : μ_Ψ (c) s_0_p : σ_0 (d) s_0_q : σ_Ψ

(e) `sig_p` : σ (f) `sig_q` : σ_q (g) `tau_p` : τ (h) `tau_q` : τ_q

Figure 12: Estimation results of more improved proposal for trend model.