RTSNet - Data Driven Kalman Smoothing

Xiaoyong Ni, Guy Revach, Nir Shlezinger, Ruud J. G. van Sloun, and Yonina C. Eldar

ICASSP 2022
Motivation

Tracking of dynamic systems is encountered in many applications:

- Localization
- Navigation
- Task Planning

such settings can often be represented as smoothing tasks, which are typically tackled using either a Model-Based (MB) or a Data-Driven (DD) method.
Model-based Deep Learning

In this work we aim to design a hybrid MB DD smoother.

Key idea: replace part of the MB computation by NN, in order to incorporate the advantages of both domains.

---

Agenda

- Smoothing Problem Formulation
- RTSNet Architecture
- Experiments on Linear and Non-linear Cases
Smoothing Problem Formulation

Consider *fixed-interval* smoothing: the recovery of a state block $\{x_t\}_{t=1}^T$ given a block of noisy observations $\{y_t\}_{t=1}^T$ for a fixed length $T$. The state and the observations are related via a dynamical system represented by

$$x_t = f(x_{t-1}) + e_t, \quad e_t \sim \mathcal{N}(0, Q), \quad x_t \in \mathbb{R}^m,$$

$$y_t = h(x_t) + v_t, \quad v_t \sim \mathcal{N}(0, R), \quad y_t \in \mathbb{R}^n.$$
Traditional Model-Based Solution

**Linear case:**
Rauch-Tung-Striebel (RTS) Smoother achieves the optimal MMSE for linear State Space model.

**Non-linear case:**
- **Extended RTS smoother**
  - Subject to Linearization error
- **Particle smoother**
  - Performance is unstable and hard to quantify
  - Computation complexity increases dramatically with the number of particles

These drawbacks motivate deriving a **NN**-aided Kalman Smoother.
Smoothing Problem Formulation

RTS Smoother Review

The MB RTS Smoother recovers the latent state variables using the forward and backward passes.

The forward pass is a standard Kalman Filter (KF), Where $\mathcal{K}_t$ is the forward Kalman Gain (KG):

$$\mathcal{K}_t = \hat{\Sigma}_{t|t-1} \cdot \hat{\mathbf{H}}^\top \cdot \hat{\mathbf{S}}_{t}^{-1}.$$  \(2\)

On the other hand, the backward KG $\mathcal{G}_t$ is given by,

$$\mathcal{G}_t = \hat{\Sigma}_{t|t} \cdot \hat{\mathbf{F}}^\top \cdot \hat{\Sigma}_{t+1|t}^{-1}.$$  \(3\)

all domain knowledge encapsulated in KGs.
Choose RTS as Backbone: all domain knowledge encapsulated in KGs.

\[ \mathcal{K}_t = \hat{\Sigma}_{t|t-1} \cdot \hat{H}^T \cdot \hat{S}_t^{-1}. \]  

(4)

\[ \mathcal{G}_t = \hat{\Sigma}_{t|t} \cdot \hat{F}^T \cdot \hat{\Sigma}^{-1}_{t+1|t}. \]  

(5)

Replace forward KG (4) and backward KG (5) with \textbf{NNs}, where Low-complexity \textbf{NN} consists of an input FC, a two-layer GRU and an output FC layer.
- NN-aided KGs compensate for model mismatch
- Avoid linearization and is less sensitive to non-linearities
- Not require inverting matrices while inferring rapidly with low computation complexity due to efficient RNNs
- Utilize a single learned forward-backward pass, which can be extended to carry out multiple passes via deep unfolding
Experiments

Linear Model

- For Linear State-space Model with Gaussian noise, RTS smoother is optimal.
- Synthetic linear dataset: set F and H to take the controllable canonical and inverse canonical forms, respectively.

Our RTSNet converges to the optimal RTS smoother.
Linear - Model Mismatch

Rotate observation matrix $H$ by $10^\circ$.

Similar results can be achieved when rotate $F$. RTSNet is superior to RTS smoother for model mismatch.
Linear - Generalization

- Scale SS model F & H to 10x10
- Scale $T_{test}$ to 1000

Figure: Training trajectory length 20, testing trajectory length 1000

<table>
<thead>
<tr>
<th>MSE Loss [dB]</th>
<th>KF</th>
<th>RTS</th>
<th>RTSNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1.9271</td>
<td>-3.7917</td>
<td>-3.7658</td>
</tr>
</tbody>
</table>
Lorenz Attractor - Sampling and decimation

Evaluate RTSNet on long trajectories ($T = 3000$) with mismatches due to sampling a continuous-time process into discrete-time.

Compare with DD Benchmark: Similar MSE performance, much better training time and inference time.

Table: Sampling and decimation.

<table>
<thead>
<tr>
<th></th>
<th>MB KS</th>
<th>Benchmark$^2$</th>
<th>RTSNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean-squared error (MSE) [dB]</td>
<td>$-10.071$</td>
<td>$-15.346$</td>
<td>$-15.56$</td>
</tr>
<tr>
<td>Inference time [sec]</td>
<td>$9.93$</td>
<td>$30.5$</td>
<td>$5.007$</td>
</tr>
<tr>
<td>Training time [hours/epoch]</td>
<td>N/A</td>
<td>$0.4$</td>
<td>$0.16$</td>
</tr>
<tr>
<td>Number of trainable parameters</td>
<td>N/A</td>
<td>$41,236$</td>
<td>$33,270$</td>
</tr>
</tbody>
</table>

Lorenz Attractor - Sampling and decimation - Trajectories

Figure: Lorenz attractor with sampling mismatch, $T = 3000$. 
Lorenz Attractor - Model Mismatch
Future Work

1. Evaluate RTSNet on real-world data-set, e.g., NCLT.

2. Extend the network to handle jumps in the hidden state and to detect outlier observations, possibly using NUV priors.

3. Try fixed-lag smoothing with sliding window. (Although fix-lag can face computation inefficiency problem, it is sometimes of more practical use.)

4. Enable RTSNet to face asynchronous measurements update.
Check Us on Arxiv

Check us on GitHub