Supplementary for Model-Free Robust Probabilistic Prediction for Stochastic Dynamical Systems via Recurrent Neural Network

1. Proof Of Lemmas

Lemma 1 *Considering the one step SDS prediction problem on trajectory $y_{1:k-1}$, the optimal $\hat{\mu}_k$ parameter of the Gaussian probabilistic prediction in log-likelihood functional metric is the same as the optimal point prediction $\hat{y}_k$ in the MSE metric.*

Proof The expected log-likelihood functional loss of probabilistic Gaussian prediction on step k can be written as:

$$\mathbb{L}_k := \mathbb{E}_{y_k \sim p_{\mathbf{y}_k|y_{1:k-1}}} \mathscr{L}(\mathcal{F}, y_k) = \int_{y_k \in \mathbb{R}^d} \mathrm{d}y p_{\mathbf{y}_k|y_{1:k-1}}(y) \bigg(\log |\hat{\Sigma}_k| + (y - \hat{\mu}_k)^\top \Sigma^{-1} (y - \hat{\mu}_k) \bigg).$$

We analysis the optimizing goal of $\hat{\mu}_k$ by calculating its gradient and its Hessian matrix:

$$\frac{\partial \mathbb{L}_k(\mathcal{F})}{\partial \hat{\mu}_k} = -(\mathbb{E}_{y_k \sim p_{\mathbf{y}_k:y_{1:k-1}}}(y_k) - \hat{\mu}_k)^\top \Sigma_k^{-1}, \frac{\partial^2 \mathbb{L}_k(\mathcal{F})}{\partial \hat{\mu}_k^2} = \Sigma_k^{-1}.$$

The covariance matrix Σ_k is positive definite, so the optimal μ_k prediction of log-likelihood metric is the expectation of $\mathbf{y}_k$. On the other hand, the expected MSE loss function on step k can be written as:

$$\mathbb{L}_k = \int_{y_k \in \mathbb{R}^d} \mathrm{d}y p_{\mathbf{y}_k|y_{1:k-1}}(y)(y - \hat{\mu}_k)^\top (y - \hat{\mu}_k).$$

Similarly, calculate its gradient and Hessian matrix:

$$\frac{\partial \mathbb{L}_k(\mathcal{F})}{\partial \hat{\mu}_k} = -(\mathbb{E}_{y_k \sim p_{\mathbf{y}_k:y_{1:k-1}}}(y_k) - \hat{\mu}_k)^\top, \frac{\partial^2 \mathbb{L}_k(\mathcal{F})}{\partial \hat{\mu}_k^2} = I.$$

Also, the optimal $\hat{\mu}_k$ prediction is the expectation of $\mathbf{y}_k$. Therefore, the result of optimal $\hat{\mu}_k$ prediction of log-likelihood metric and MSE metric is consistent. ■

Lemma 2 *Considering the one step SDS prediction problem on trajectory $y_{1:k-1}$, the optimal $\hat{\mu}_k$ parameter of the Laplacian probabilistic prediction in log-likelihood functional metric is the same as the optimal point prediction $\hat{y}_k$ in the MAE metric.*

Proof The expected log-likelihood functional-based loss of probabilistic prediction on step k can be written as:

$$\mathbb{L}_{y_k}(\mathcal{F}) := \mathbb{E}_{y_k \sim p_{\mathbf{y}_k|y_{1:k-1}}} \mathscr{L}(\mathcal{F}, y_k) = \int_{y_k \in \mathbb{R}^d} \mathrm{d}y_k \log \hat{p}(y_k) p_{\mathbf{y}_k|\mathbf{y}_{1:k-1}}(y_k|y_{1:k-1}).$$

In the case of Laplacian prediction, the expression has an equivalent form:

$$\mathbb{L}_{y_k}(\mathcal{F}) = -\sum_{i=1}^{d} \log \hat{b}_k^{(i)} - \sum_{i=1}^{d} \frac{1}{\hat{b}_k^{(i)}} \int_{y_k^{(i)} \in \mathbb{R}} |y_k^{(i)} - \hat{\mu}_k^{(i)}| p_{\mathbf{y}_k|\mathbf{y}_{1:k-1}}(y_k^{(i)}|y_{1:k-1}) \mathrm{d}y_k^{(i)}.$$

On the other hand, the expected MAE loss of point prediction on step k can be written as:

$$\mathbb{E}_{y_k \sim p_{\mathbf{y}_k|y_{1:k-1}}} \mathrm{MAE}(\mathcal{F}, y_k) = \sum_{i=1}^{d} \int_{y_k^{(i)} \in \mathbb{R}} |y_k^{(i)} - \hat{\mu}_k^{(i)}| p_{\mathbf{y}_k|\mathbf{y}_{1:k-1}}(y_k^{(i)}|y_{1:k-1}) \mathrm{d}y_k^{(i)}.$$

Consider the dimensions separately, on each dimension, the MAE and the Laplacian-based predictor has the same optimization goal:

$$\int_{y_k^{(i)} \in \mathbb{R}} |y_k^{(i)} - \hat{\mu}_k^{(i)}| p_{\mathbf{y}_k|\mathbf{y}_{1:k-1}}(y_k^{(i)}|y_{1:k-1}) \mathrm{d}y_k^{(i)}.$$

Therefore, the optimization results in μ_k of MAE and Laplacian log-likelihood are consistent. ■

2. Experimental Settings

2.1. SDS Parameters

Throughout the experiments, the SDS noises are set as independently identically distributed at each time step. Specifically, the observation noise v_k is set to follow a normalized Gaussian distribution $\mathcal{N}(\cdot; 0, I)$, and the distribution of the plant noise w_k varies in different experimental settings. For experiments in $\mathcal{I}_2$ and $\mathcal{I}_1$ conditions, the plant noise is set to follow normalized Gaussian distribution and the Student's t(1.5) distribution respectively.

We tested the PP's performance on three different types of SDS, the linear system, the simple non-linear observation system, and the Lorenz system. For the linear system, we choose a typical marginally stable SDS, the dynamics are shown as follows:

$$x_{k+1} = Ax_k + w_k,\ y_k = Cx_k + v_k,$$

$$\text{where } A = \begin{bmatrix} 1 & 0.1 & 0 \\ 0 & 1 & 0.5 \\ 0 & 0 & 0.98 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.7 & 0.5 \end{bmatrix}.$$

The simple non-linear observation system has the same state dynamics as the linear system model, but its observation is changed to $y_k = ||x_k||_2 + v_k$.

For the case of the Lorenz system, we used the discretized form of its state dynamics:

$$\begin{cases} x_{k+1} = \begin{pmatrix} x_k^{(1)} + \mathrm{d}t\sigma(x_k^{(2)} - x_k^{(1)})) \\ x_k^{(2)} + \mathrm{d}t(\rho x_k^{(1)} - x_k^{(2)} - x_k^{(1)} x_k^{(3)}) \\ x_k^{(3)} + \mathrm{d}t(x_k^{(1)} * x_k^{(2)} - \beta x_k^{(3)}) \end{pmatrix} + w_k \\ \end{cases}.$$

In which $\mathrm{d}t$ is the discrete time interval, σ, ρ, and β are constants of the Lorenz system. By convention, we set $\mathrm{d}t = 0.01, \sigma = 10, \beta = \frac{8}{3}, \rho = 28$. The observation is set as a simple linear process that observes the first dimension of the state vector $y_k = x_k^{(1)} + v_k$, and a simple non-linear observation that observes the norm of the state vector $y_k = ||x_k||_2 + v_k$.

2.2. Network Parameters & Training Settings

The detailed parameters of the RNN are set as follows. We use RNNs with 2 hidden layers, the width of the hidden layer is set as 30 times the observation vector's dimension. The activation function of μ prediction RNNs is ReLU, and for Σ, b prediction RNN, we used tanh for activation. The output layer has the same size as the observation dimension, and particularly for Σ prediction, we specify that the output is regarded as the diagonal elements of the covariance matrix.

For simulation on each SDS, we numerically generate 200000 observation trajectories of the same length 100 with different random seeds and randomly initialized x_0 by a Gaussian distribution. Half of the trajectories are used for training. Specifically, for the predictors with $\mathcal{I}_2$ information set, the training loss for μ prediction is changed from MSE loss to the Huber loss with $\delta = 10$ as an approximation. Though harming the optimality slightly, the Huber loss has the advantage of a strong tolerance to outliers. Which prominently increases the stability and efficiency of the training process.

In the training process, we use 90% of the training data as the training set, and the remainder is used for validation.