Selecting the best ML architecture among LSTM, GRU, and Reservoir for small-size data

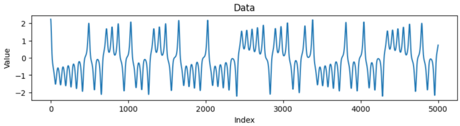
Libraries such as TensorFlow and PyTorch fundamentally rely on backpropagation (gradient descent), and modifying that core algorithm is VERY DIFFICULT at the user level!!

$\Rightarrow$ I take a math-first approach to model design.

Anticipating the conclusion:

- Bayes Opt with Delay coordinate will win against Back propagation
- Model selection phase will win against Grad descent
- RNN will win against Memory cell in LSTM

Experimental setup for model comparison (1D Lorenz data)



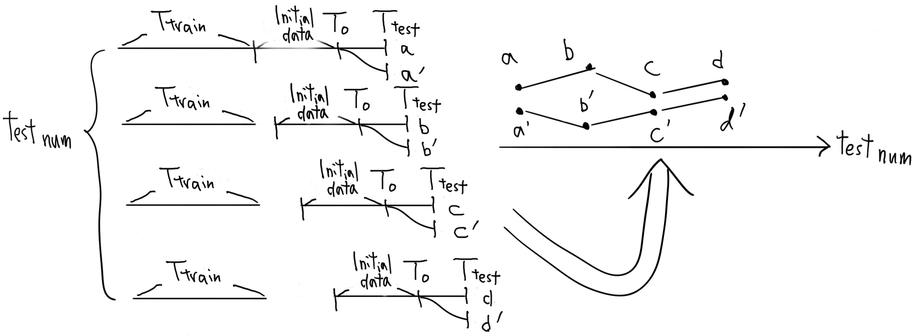
Parameters fixed for fair accuracy comparison:

- Number of trainable parameters: 20,000 (nodes: Reservoir ~ 150, LSTM ~ 70, GRU ~ 80)
- Training data size: 5,000
- Prediction horizon: 100 steps ahead
- L^2 regularization parameter: 0.0001

Candidate of ML architectures (3 models + later 2 models)

- Standard LSTM (TensorFlow): Adam optimizer for grad descent
- LSTM without back propagation (adopting Bayes Opt with Delay coordinate)
- GRU without back propagation (adopting Bayes Opt with Delay coordinate)

- Perform predictions 3000 times (test num), each shifted by one step, and compute the MAE (mean absolute error).
- Repeat the same process until the average MAE stabilizes (80 times).



BayesOpt with Delay coordinate VS back propagation

Delay Coordinates (see for e.g. Nakai-Saiki '21)

$$U_t = (data(t), data(t-\tau), \cdots, data(t-(M-1)\tau))$$

We train by using this U_t.

TensorFlow-LSTM: $\boxed{\text{Input}}$ $\tau = 1$, M is chosen prior to training

$\boxed{\text{Output}}$ $\tau = 1$, $M = 1$

Bayes Opt with Delay coordi: τ and M are also learned during training

Sigmoid function h, $\odot$: element-wise product

LSTM

$$
\begin{aligned}
\text{Input gate} \quad & i_t = h(W_{datain} U_t + W_{gatein} x_{t-1} + bias), \\
\text{Forget gate} \quad & f_t = h(W_{dataforget} U_t + W_{forget} x_{t-1} + bias), \\
\text{Output gate} \quad & o_t = h(W_{dataout} U_t + W_{gateout} x_{t-1} + bias), \\
\text{Memory cell} \quad & c_t = i_t \odot \tanh(W_{datacell} U_t + W_{cell} c_{t-1} + bias) + f_t \odot c_{t-1}, \\
\text{Output} \quad & x_t = o_t \odot \tanh(c_t), \quad \widehat{U}_{t+1} = W_{out} x_t
\end{aligned}
$$

Bayes Opt with Delay coordinate

Bayes Opt is the only way that we can adopt Delay coordinate!!

Random matrices

$W_{datain}, W_{dataforget}, W_{dataout}, W_{datacell}, W_{gatein}, W_{forget}, W_{gateout}, W_{cell}$

By ridge regression (convexity), we first determine the following W_{out}:

$$U_{t+1} \simeq W_{out} x_t \qquad \text{(the idea of the conventional Reservoir).}$$

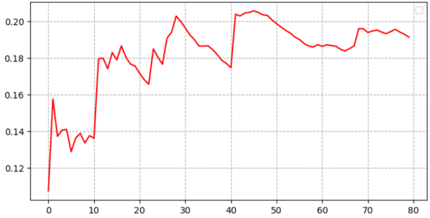
- By the convexity, W_{out} is explicitly obtained
- At this stage, not directly measuring $U_{t+1} - \widehat{U}_{t+1}$ so far

In the hyper-parameter space, we apply Bayesian Optimization (Optuna) to minimize the following objective function (RMSE):

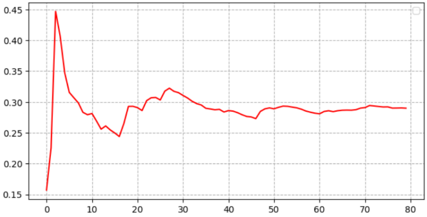
$$\left(\frac{1}{T} \sum_{t=0}^{T-1} |U_{t+1} - W_{out} x_t|^2 \right)^{1/2}.$$

Note: Random matrices must be continuously depending on seed value!

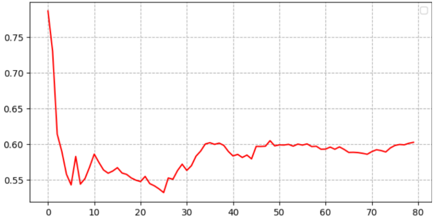
Results (GRU-Bayes, LSTM-Bayes, LSTM-TensorFlow)



GRU-BayesOpt-DelayCoordi: MAE 0.19



LSTM-BayesOpt-DelayCoordi: MAE 0.29



LSTM-TensorFlow: MAE 0.60

Bayes Opt with Delay coordinate
won against
back propagation!

Anticipating the conclusion:

- RNN will win against memory cell in LSTM
- Model selection phase will win against Grad descent

Only Memory Cell LSTM (RNN-ESP VS LSTM-ESP)

$$\begin{aligned} \text{input gate:} \quad & i_t = tanh(W_{datain} U_t) \\ \text{forget gate:} \quad & f_t = h(W_{dataforget} U_t + bias) \\ \text{memory cell:} \quad & c_t = i_t + f_t \odot c_{t-1} \\ \text{Ridge regression:} \quad & U_{t+1} \simeq W_{out} c_t \end{aligned}$$

$$\text{induction arg:} \quad c_t = \sum_{s=0}^{t} \left(i_s \prod_{r=s+1}^{t} (\odot f_r) \right), \quad \prod_{r=t+1}^{t} f_r := 1.$$

Now assume that the following probability:

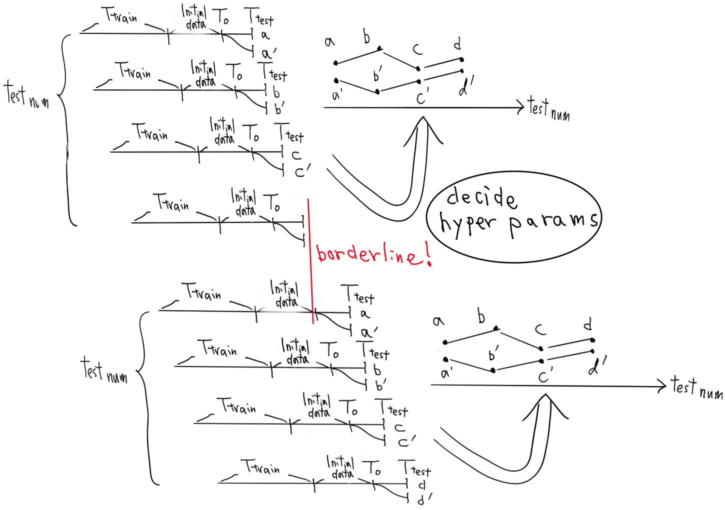
$$\gamma := P(f_r \cdot e_j \neq 0) \quad (0 < \gamma < 1)$$

is independent of j and r, where e_j is the j'th unit vector. Also assume f_r and $f_{r'}$ $(r \neq r')$ are probabilistically independent. Then we have ESP:

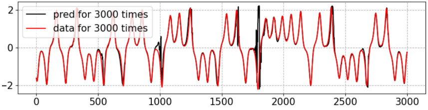
$$P\left(\prod_{r=s+1}^{t} f_r \cdot e_j \neq 0 \right) = \prod_{r=s+1}^{t} P(f_r \cdot e_j \neq 0) = \gamma^{t-s} \to 0 \quad (s \ll t \to \infty).$$

Reservoir (Online) with Model selection phase

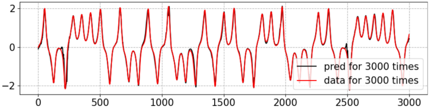
The first loop: Ridge regression, The second loop: Bayesian optimization using model selection data.



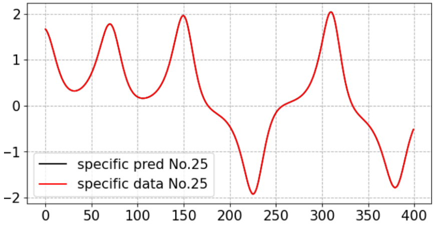
Both Only-Memory-Cell-LSTM and online Reservoir with model selection phase achieved good results



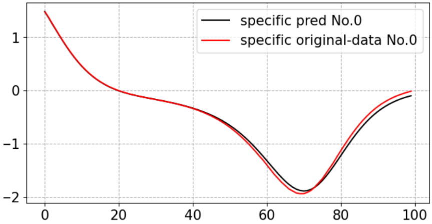
OnlyMemoryCellLSTM: MAE 0.11, prediction 400 steps ahead



Reservoir: MAE 0.067, 1500+150 training samples

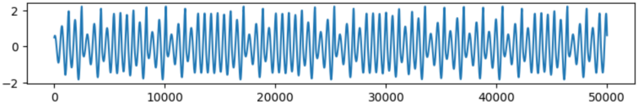


OnlyMemoryCellLSTM: MAE 0.11, prediction 400 steps ahead



Reservoir: MAE 0.067, 1500+150 training samples

Replication Test: Rössler Equation



- Number of trainable parameters: 20,000 (Reservoir ∼150 nodes, LSTM ∼70, GRU ∼80)
- Training data size: 5,000 (Reservoir uses 1,500+150)
- Prediction: 100 steps ahead (FastLightLSTM: 400 steps ahead)
- L^2 regularization parameter: 0.0001

Learning Results (MAE)

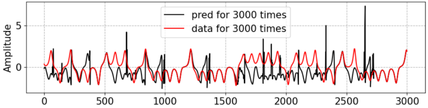
- OnlyMemoryCellLSTM: 0.005 (prediction 400 steps ahead)
- Res-ModelSelect: 0.030 (1,500+150 training samples)
- DelayCoodi-LSTM: 0.055
- DelayCoordi-GRU: 0.061
- TensorFlow-LSTM: 0.52

Final judge: OnlyMemoryCell-LSTM VS Reservoir

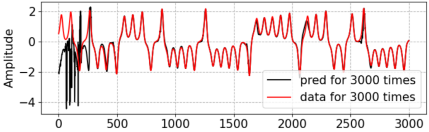
My research focuses on small-size real data.

$\Rightarrow$

For the final judge, we examine that the training data is more and more smaller-size case.



OnlyMemoryCellLSTM: MAE 0.854, 650 train data



Res: MAE 0.188, 500+150 train data

- With only 650 data points, a significant difference emerged between the Only-Memory-Cell-LSTM and reservoir computing. Future research should adopt $\boxed{\text{reservoir computing}}$ as the optimal approach.