Deep RNNs for Non-Linear State Estimation

Catherine Watkins
cwatki14@stanford.edu
Github repo: https://github.com/cwatki14/BenchmarkNonLinearEstimation, *Presentation:* https://youtu.be/X0VF0XUxnac

Predicting

Estimation of hidden system states based on a series of noisy measurement data is a classical problem in science and engineering addressed in many fields such as navigation, process control, or time series forecasting. Common model-based methods in Filtering in Dynamical Systems include the Kalman filter for linear systems, and the (Iterated) Extended Kalman filter, Unscented Kalman filter, or Particle filter for non-linear systems.

This project considers a challenging non-linear dynamical system in which traditional approximate techniques can perform poorly. Current results in literature conclude no single filtering technique dominates as a sure solution to all non-linear estimation problems. The solution proposed in this project, a deep recurrent neural network, aims to be a more widely applicable and consistent solution to challenging non-linear dynamical systems. The resulting RMSE of the neural network solution is superior than the current state of the art method, the Particle Filter.

Data & Features

The example considered was a non-linear time series, outlined in equations 1-3 below, which is widely used for bench-marking numerical filtering techniques (Reference 4.) Model parameters depicted in the equations below are chosen based on Reference 4 (optimal baseline RMSE = 4.68.) This dynamical system is a one-state example with a bimodal prior.

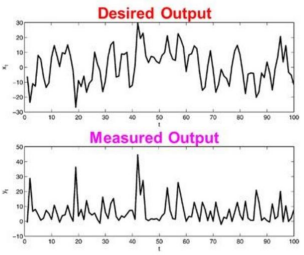
$$x_t = \frac{x_{t-1}}{2} + 25\frac{x_{t-1}}{{x_{t-1}}^2} + 0.8cos(1.2t) + w_t \quad (1)$$

$$y_t = \frac{{x_t}^2}{20} + v_t \quad (2)$$

$$x_o \sim N(0,25), w_o \sim N(0,10), v_t \sim N(0,1) \quad (3)$$

The data required to train and validate a neural filter solution can be simulated using the dynamical system above. For a single time step t, the measured outputs (y_t) represent our feature vector and the desired outputs (x_t) represent the label or ground truth at time t. This process of constructing measured outputs (y_t) and desired outputs (x_t) from sampling of the normal distribution is repeated for the series length (T), which represents the length of the time series which will be estimated by the sequence model. In this project we considered a series length of 100 time steps. The graph below represents the features and labels for a given observation of length T of the system.

The process of constructing data for training simply requires generating data by sampling from the normal distribution and propagating through time for each epoch. Between 5,000 and 25,000 epochs of data were required to train the network depending on the hyperparameters. The resulting models were validated on 10 epochs of validation data.



Models

This project explores applying a deep RNN consisting of stacked LSTM cells to solve the non-linear state estimation problem outlined in equations 1 and 2. The mathematical formulation of a single LSTM cell is outline in equations 4-9 below.

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \quad (4)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \quad (5)$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \quad (6)$$

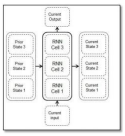
$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \quad (7)$$

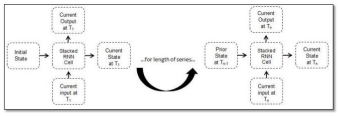
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \quad (8)$$

$$a^{<t>} = \Gamma_o * tanh(c^{<t>}) \quad (9)$$

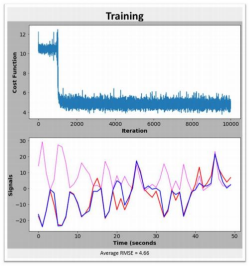
LSTMs, like Kalman Filters, allow information from prior states to information the prediction at a current time step.

The graphics below depict how multiple RNN cells, in this case, LSTMs can be stacked to create a deep 3-layer RNN architecture. This stacked cell unit is repeated for the length of the entire time series estimation problem, in this case T = 100 time steps. The loss function used was RMSE, and Adam was the selected optimizer.





Results



The graphs above summarize the training and validation of the tuned model. The RNN outputs (shown in blue) track closely with the Desired Outputs (shown in red.) The inputs to the model are shown in pink. This model produced an average RMSE of 4.67 in training average over the last 100 epoch, and an RMSE of 4.64 in 10 epochs of validation. This model was trained for a total of 10,000 epochs and validated on 10 epochs, which each consist of 10 batches of the time series of length 100.

Learning Rate	2 Layers	3 Layers	4 Layers
1e-2	12.89/12.97	5.51/5.57	11.55/11.72
1e-4	5.15/5.11	4.66/4.74	5.17/5.33
1e-6	10.44/10.60	10.50/10.46	10.45/10.35

The table above summarizes some preliminary hyper parameter tuning of the learning rate and number of stacked recurrent layers. The final model implemented a exponential learning rate decay schedule (3-layer RNN, batch size =10, with an exponential learning rate schedule with staircase step every 1000 epochs initialized at 1e3 with a base of 0.96.)

Discussion

The performance of the tuned RNN filter model is compared to other non-linear filtering techniques in the table below. The baseline optimal RMSE on this benchmark problem was produced by a tuned Particle Filter (RMSE 4.68.) The Neural Filter surpassed this error rate with a RMSE of 4.64. Further hyper parameter tuning would likely improve performance further.

Performance metrics for filters	
Filter	RSME
Extended Kalman Filter	21.53
Unscented Kalman Filter A	26.97
Particle Filter B	4.68
Neural Filter	4.64

These results are very interesting. Non-linear filtering problems are challenging because no single technique prevails over the others. A neural filter model can be seen as a possible "universal" estimator even for challenging non-linear systems. As long as accurate desired outputs can be generated by sufficient instrumentation, etc., a neural filter can be a robust alternative to other filtering approaches.

Future

Future work should consider a few things avenues to improve model performance. First a deeper exploration in tuning the learning rate schedule, such as various stepped decreases or a cosine annealing of the learning rate over the training process may be beneficial to model performance. Second, optimizing batch size would be a useful next step. Lastly, considering the use of Gradient Recurrent Units (GRUs) as an alternative to LSTMs. GRUs are less complex; they only contain a single gate, while the LSTM consists of three gates. Hence, a model consisting of GRUs will have fewer learn-able parameters and may be less intensive to train than one consisting of LSTMs.

References

1. James T. Lo (1994), "Synthetic Approach to Optimal Filtering", IEEE Transactions on Neural Networks, Vol. 5, No. 5.
2. Zachary C. Lipton, John Berkowitz, Charles Elkan (2015), "A Critical Review of Recurrent Neural Networks for Sequence Learning", arXiv.org
3. Cappe et. al. (2007), "An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo", Proceedings of the IEEE, Vol. 95, No. 5.
4. M. L. Psiaki (2013), "The blind tricyclist problem and a comparative study of nonlinear filters: A challenging benchmark for evaluating nonlinear estimation methods," IEEE Control Systems Magazine, Vol. 33, No. 3.
5. Ristic et. al. (2004), "Beyond the Kalman Filter. Particle Filters for Tracking Applications", Artech House.