Model identification and uncertainty prediction using deep learning

Prithvi Dake

Department of Chemical Engineering

ChE 230D Project
March 21, 2024

Outline

1. Incentive for deep learning (specifically hybrid modelling)
2. Case study for partial state measurement
3. Towards a structured greybox model
4. Quantile regression

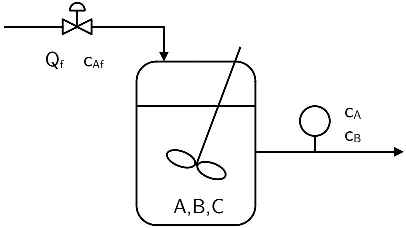
Introduction

Motivation

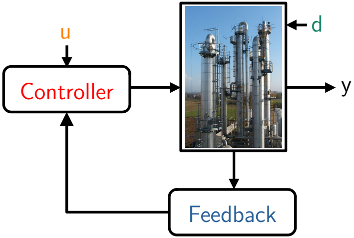
1. Interactions are difficult to model
2. We can't use the plant for ourselves!
3. These interactions are too expensive to investigate
4. Some crucial intermediates are never measured (partial state measurements)

Solution

1. Construct a complete black-box model (or a hard-coded model)
2. OR use neural networks to represent diffcult-to-model portions of a first-principles model



Don't expect data particularly useful for model building in closed-loop plants



Sample training data

Data generating model - noise added

$$\frac{dc_A}{dt} = \frac{Q_f(c_{Af} - c_A)}{V} - r_1$$
$$\frac{dc_B}{dt} = \frac{-Q_f c_B}{V} + r_1 - 3r_2$$
$$\frac{dc_c}{dt} = \frac{-Q_f c_C}{V} + r_2$$

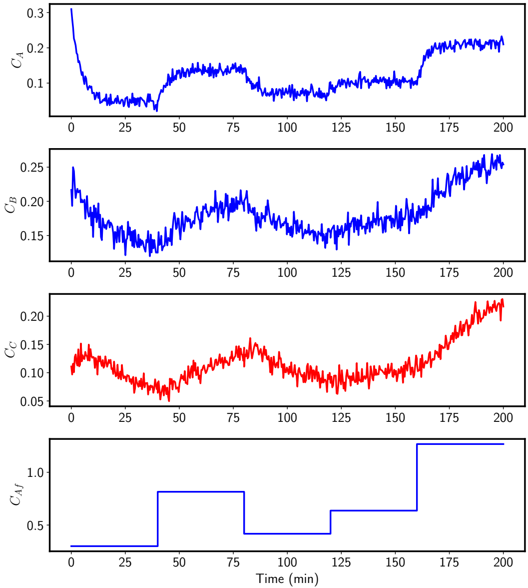
$$r_1 = k_1 c_A \qquad r_2 = k_2 c_B^3 - k_{-2} c_C$$

$$y = (c_A, c_B) \qquad u = c_{Af}$$

Simulated data for the process model using PRBS signal

Red: Not measured

Blue: Measured



Structured greybox model

Hybrid model

$$\frac{dc_A}{dt} = \frac{Q_f(c_{Af} - c_A)}{V} - \phi_1(\mathbf{x}, \mathbf{p}, u, \beta)$$
$$\frac{dc_B}{dt} = \frac{-Q_f c_B}{V} + \phi_1(\mathbf{x}, \mathbf{p}, u, \beta) - 3\phi_2(\mathbf{x}, \mathbf{p}, u, \beta)$$

$$y = (c_A, c_B) \quad u = c_{Af}$$
$$x = [c_A, c_B]^T \quad p = [x(t - N_p\Delta)^T, \ldots, x(t-\Delta)^T]^T$$

Primer on FNNs ($\mathbb{R}^3 \to \mathbb{R}^2$)

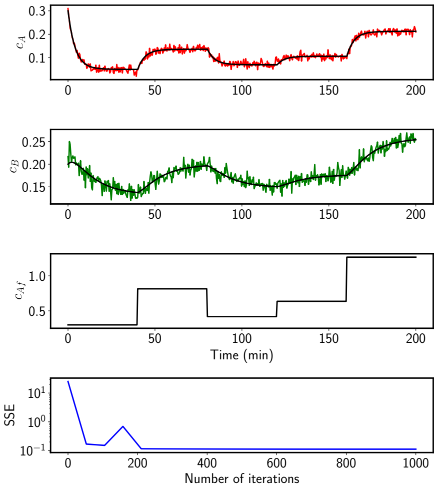
$$\phi_{2\times 1}(\cdot) := \sigma(\mathbf{W}_{2\times 3}(\cdot)_{3\times 1} + \mathbf{b}_{2\times 1})$$
$$\beta = (\mathbf{W_q}, \mathbf{b_q}, \ldots, \mathbf{W}_1, \mathbf{b}_1)$$
$$\phi(\mathbf{u}, \beta) = \mathbf{W_q}(\phi_{q-1} \circ \ldots \circ \phi_1(\mathbf{u})) + \mathbf{b_q}$$
$$f \circ g := f(g(\cdot))$$

Reconstructing unmeasured states with history
Hawkeye technology in cricket
Famous DRS by Tendulkar on LBW
India won btw (World Cup 2011)!

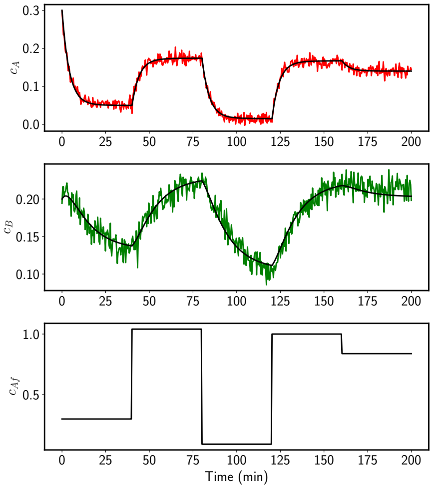
$$L(x, p, u, \beta) = \sum_{i=1}^{N_{tr}} \sum_{k=0}^{N_t} |y_i(k) - \hat{y}_i(k)|^2$$

1. Custom integrator like RK4
2. Take care to simultaneously update the past vector $\mathbf{p}$ (it is a moving window)
3. $\dfrac{\partial L}{\partial \beta}$ use autodifferentiation libraries
4. Eg. Pytorch, TensorFlow or JAX

Results of training and performance



Training of the hybrid model



Performance of trained model on test data set. We get considerably good-fit considering we have partial state measurements.

Uncertainty prediction (but no estimation)

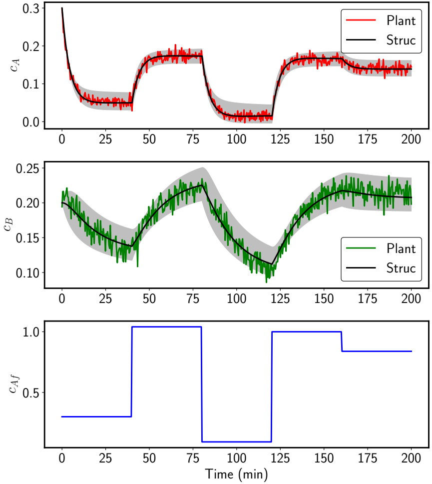
Quantile regression (Pinball loss)

$$L_\tau(y, \hat{y}) = max[\tau(y - \hat{y}), (1 - \tau)(\hat{y} - y)]$$

$$L_\tau = \sum_{i=1}^{N} L_\tau(y_i, \hat{y}_i)$$

Conclusion

1. Quantile regression is a computationally cheap uncertainty estimation method for neural network
2. It predicts uncertainty in data not model uncertainty
3. Ensembling (bootstrap) or Bayesian neural networks could be better approaches but are computationally expensive



The grey region indicates 2.5% and 97.5% quantile ranges for error in the hybrid model prediction.

Code and Resources

1. All the code can be pulled from:
   `https://github.com/dakeprithvi/ChE-230D.git`
2. If you are using linux OS, just run 'make' over the pulled repo
3. Advanced libraries like PyTorch or TensorFlow are deliberately avoided to keep the code simple and reproducible.
4. For this purpose, JAX (a new library developed by Google) has been used.