NIM: Generative Neural Networks for Simulation Input Modeling

Wang Cen, Emily A. Herbert, Peter J. Haas
University of Massachusetts Amherst

Motivation

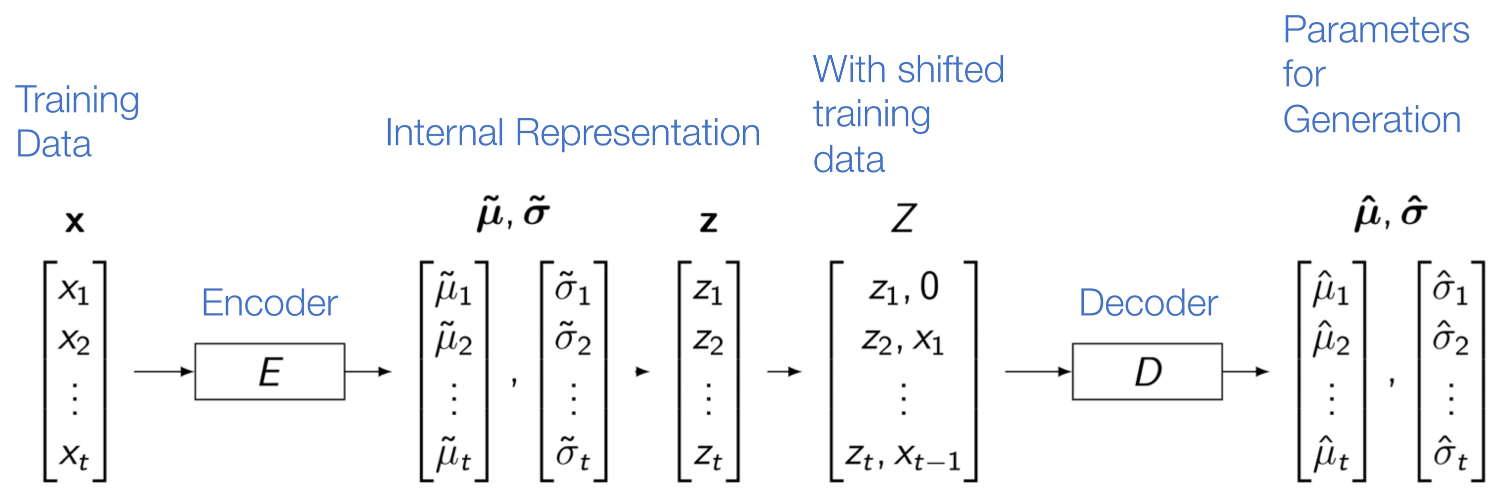
- Input modeling is key to a simulation study
- But modeling input processes is challenging because
  - Distribution-fitting software fails for complex i.i.d. distributions
  - Real world input processes are often complex time-dependent stochastic processes, so we must rely on a simulation expert
- Good news: data is becoming abundant thanks to
  - IoT sensors, logs, annotated machine vision, etc.

A Solution: Neural Input Modeling (NIM)

- NIM is a generative neural network
  - Automatically fits complex stochastic processes without a priori knowledge of even the type of the process
  - Automatically and efficiently generates sample paths during a simulation run
- Variational Autoencoder (VAE) + Long Short-Term Memory (LSTM)
  - VAE is an easy-to-use generative neural network
  - LSTM concisely captures temporal correlation
- NIM has good modeling accuracy and fast generation speed

NIM Training Architecture

Two neural networks (Encoder and Decoder) are trained jointly



Use backpropagation (gradient descent) to minimize loss function:

$$L(\mathbf{x}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}) = -\sum_{i=1}^{t}(\log \tilde{\sigma}_i^2 - \tilde{\mu}_i^2 - \tilde{\sigma}_i^2 + 1) + \sum_{i=1}^{t}(\log 2\pi + \log \hat{\sigma}_i^2 + \frac{(x_i - \hat{\mu}_i)^2}{\hat{\sigma}_i^2})$$

First term: KL divergence between $N(\tilde{\boldsymbol{\mu}}, \mathrm{diag}(\tilde{\boldsymbol{\sigma}}))$ and $N(\mathbf{0}, \mathbf{I})$

Goal: Make $z_1, \ldots, z_t$ look like i.i.d. $N(0,1)$ samples (as assumed during generation)

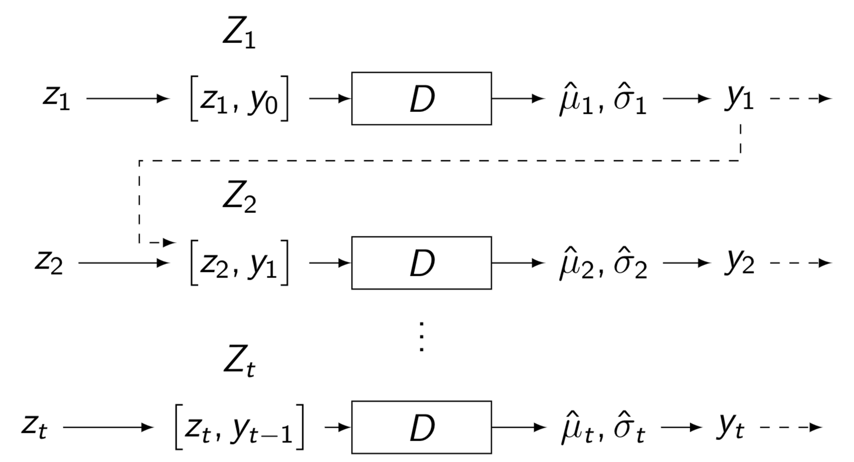
Second term: negative log-likelihood of training data under $N(\hat{\boldsymbol{\mu}}, \mathrm{diag}(\hat{\boldsymbol{\sigma}}))$

Goal: Make joint distribution of $y_1 \sim N(\hat{\mu}_1, \hat{\sigma}_1), \ldots, y_k \sim N(\hat{\mu}_k, \hat{\sigma}_k)$ close to distribution of training data

NIM Generation Architecture

Only the decoder is used

- Sample $z_i \sim N(0,1)$
- Pass $[z_i, y_{i-1}]$ to the decoder, where y_{i-1} is value generated in the previous step and $y_0 = 0$
- Sample $y_i \sim N(\hat{\mu}_i, \hat{\sigma}_i^2)$
- Repeat until a sample path $[y_1, y_2, \ldots, y_t]$ is generated



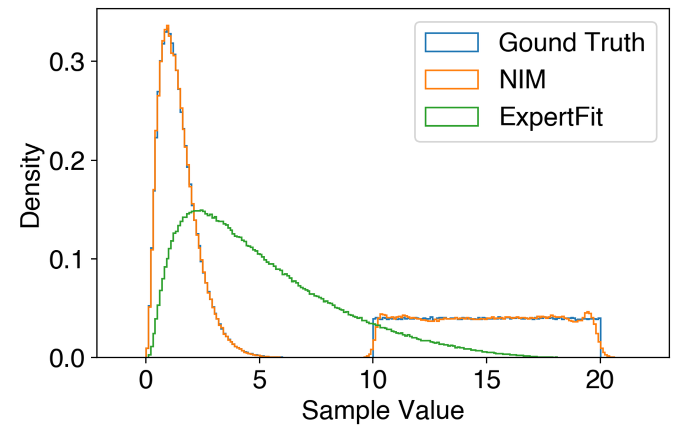
Exploiting Domain-Specific Knowledge

- NIM can exploit prior knowledge about the process to improve accuracy and generation speed
  - Dataset is non-negative: apply log transformation to the raw data
  - Dataset is inside a range: apply inverse sigmoid function to the raw data
  - Dataset is truly i.i.d.: a simplified version of NIM can be used, replacing LSTM units with multi-layer perceptrons (no explicit modeling of temporal correlation)
  - Dataset is multi-modal: for final generation step, VAE now learns parameters of a Gaussian mixture model

Results

- Modeling accuracy

Gamma-Uniform mixture

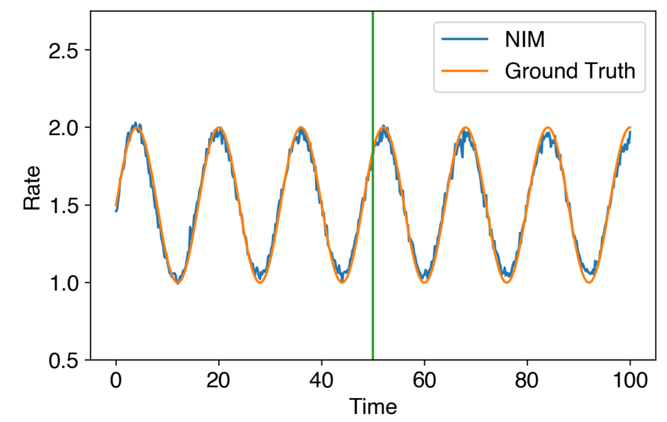


- Training distribution = 0.6 * Gamma(2, 2.875) + 0.4 * Uniform(10, 20)
- Distribution-fitting software fails to capture complex multi-modal structure
- NIM gives close approximation

Results

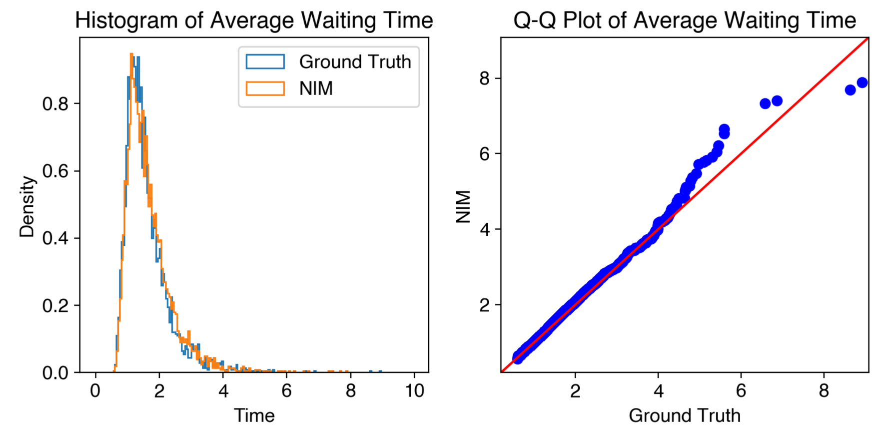
- Modeling accuracy

Nonhomogeneous Poisson process



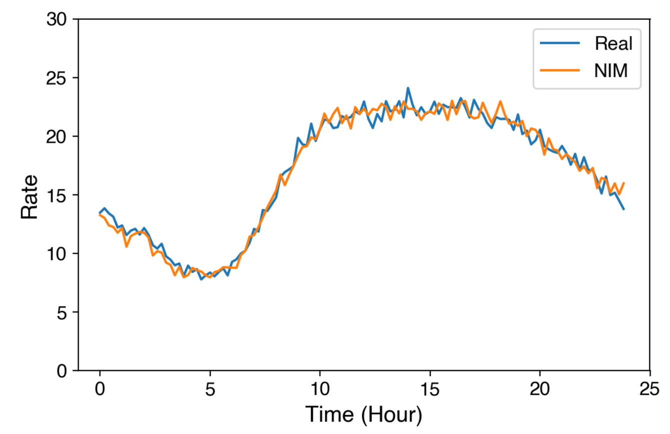
- Nonhomogeneous Poisson process with rate function $\lambda(t) = \frac{1}{2}\sin(\frac{\pi}{8}t) + \frac{3}{2}$
- The figure shows the empirical rate function
- NIM is only trained with data for t < 50 (left of the green line)
  - Can extrapolate semi-stationary process over time (right of the green line)
  - Future work: modeling high-order differences (like ARIMA) for nonstationary process extrapolation

End-to-end queue simulation (4000 i.i.d. reps)



- Ground Truth
  - Simulated a NHPP/Gamma/1 queue
  - NHPP and Gamma as previous
  - Computed average waiting time of the first 100 jobs
- NIM
  - Learned both arrival process distribution and service time distribution
  - Used these in the queueing simulation
  - Again computed the average waiting time of the first 100 jobs

Daily San Francisco Emergency Calls in 2018



- Real world data: emergency calls to SF Fire Department
- Computed empirical rate function of daily calls
- Closely approximated the empirical rate for actual data

- Generation speed
  - Roughly 8 million i.i.d. random variables per second
  - Roughly 1,200 length 1,000 sample paths per second
  - Can be further improved by using GPUs

Conclusion

- NIM uses generative neural networks to model and generate complex stochastic sequences, without a priori knowledge of the underlying process
- NIM can help lower one of the key barriers to simulation, making it more easily available to non-experts.