Estimating Virtual Statistics in Simulations using kNN

221870001 Zihao Ma
221870264 Shaozhe Ke

April 21,2025

Outline

1. Introduction
2. kNN Method
3. Error Measurement
4. Algorithm: LORO-CV
5. Experimental Results
6. Conclusion

1.Introduction

- Motivation and Problem Statement
- What is Virtual Performance?
- Types of Virtual Statistics
- Why Choose Conditional Virtual Performance?
- Limitations of Traditional Methods
- New Paradigm: Simulation as Data
- Research Questions and Contributions

Motivation and Problem Statement

- Traditional simulation output analysis focuses on **long-run average performance**, assuming system stationarity.
- However, many real-world systems (e.g., nonstationary queues, maintenance, emergency response) exhibit dynamic behaviors.
- We are often interested in performance **conditioned on a specific event** at a particular time τ_0.
- **Goal:** Estimate virtual performance at τ_0 using saved simulation paths, without modifying the simulation model or rerunning it.

What is Virtual Performance?

- Consider a simulation with replications, each producing events at times $\{t_{ij}\}$ and associated outputs $Y(t_{ij})$.
- Define virtual performance $V(\tau_0)$ as the performance measure observed by a **natural arrival** at time τ_0.
- We focus on the **expected virtual performance**:

$$v(\tau_0) = \mathbb{E}[V(\tau_0)].$$

- **Challenge:** How to estimate $v(\tau_0)$ accurately from simulation outputs without bias or high variance?

Types of Virtual Statistics (Part 1)

In this section, we define and compare three types of virtual statistics: Injected, Phantom, and Conditional.

- **Injected Virtual Performance:**
  - This type of virtual statistic is based on the performance observed when an artificial customer or event is injected into the system.
  - For example, this could be done by artificially increasing the arrival rate at time τ_0.
  - This method alters the dynamics of the system, and hence the statistics derived from this approach do not reflect the actual performance of the system under natural conditions.
- **Phantom Virtual Performance:**
  - The phantom virtual performance measures the performance of the system by observing the system at τ_0, assuming that an event has occurred, but without actually introducing the event.
  - This is often used as an estimate of the system's performance at that specific time point, but can be biased in non-stationary systems where event arrivals are infrequent.

Types of Virtual Statistics (Part 2)

- **Conditional Virtual Performance:**
  - The conditional virtual performance measures the system's performance for natural arrivals that occur at τ_0.
  - Unlike injected or phantom performance, this method uses actual events that naturally occur within the system, making it the most accurate reflection of the real-world system's behavior.
  - This type is particularly useful because it does not introduce any artificial distortions or assumptions about the system's state.

Why Choose Conditional Virtual Performance?

- **Realism:** It reflects the performance for real customers or events naturally arriving at time τ_0, without artificially altering the system.
- **Unbiasedness:** Since this method uses actual data (natural arrivals), it avoids the biases introduced by artificial interventions in the system.
- **Applicability:** This type is especially useful for systems with rare events or nonstationary behavior, where artificial modifications or assumptions might distort the true performance.
- **Real-world relevance:** This approach is directly applicable to systems such as waiting times in queues, emergency response systems, and production systems where performance under natural conditions is of primary interest.

Limitations of Traditional Methods

- **PASTA (Poisson Arrivals See Time Averages):** Assumes stationarity; fails in nonstationary systems.
- **Time window averaging:** Use of $[\tau_0 - \delta, \tau_0 + \delta]$ around τ_0 leads to bias–variance tradeoff.
- **Online estimators:** Require tracking statistics during simulation runs; difficult for rare or dynamic events.

New Paradigm: Simulation as Data

- Modern simulation practices **retain full event-level sample paths**.
- This opens the door to **post-simulation analytics** using tools from machine learning and nonparametric statistics.
- The paper proposes using **k-nearest neighbors (kNN)** on stored simulation outputs to estimate conditional virtual performance.
- No need to rerun simulations—just analyze what you've already collected.

Research Questions and Contributions

1. How to formally define and estimate conditional virtual performance?
2. Can we ensure the estimator is **consistent and unbiased** as simulation replications grow?
3. How to choose the number of neighbors k optimally in a data-dependent way?
4. How does the method perform on real nonstationary simulation models?

Key Contributions:

- Theoretical guarantees for a kNN-based estimator.
- Error estimation via bootstrap and sample variance.
- A novel leave-one-replication-out cross-validation (LOCO-CV) scheme for tuning.

2.kNN Method

- Setup and Notation
- Key Variables And Assumptions
- Key Lemma 1: Shrinking Window Width
- Key Lemma 2: Dynamic Neighbor Selection
- Key Theorem 1: Asymptotic Independence
- Theorem 2: Asymptotic Consistency
- Theorem 3: Asymptotic Unbiasedness

Setup and Notation

- **Objective**: Estimate expected virtual performance $v(\tau_0) = \mathbb{E}[V(\tau_0)]$.
- **Estimator**:

$$\bar{V}(\tau_0) = \frac{1}{k} \sum_{\ell=1}^{k} Y(\tau_0^{(\ell,n)}),$$

  where $\tau_0^{(\ell,n)}$ are the k nearest neighbors to τ_0.
- **Superposed Process**:

$$\mathcal{T}_n = \{t_{ij} : i = 1, \ldots, M_j, j = 1, \ldots, n\}.$$

Key Variables And Assumptions

1. **Arrival Count** $N_w(t)$
   - **Definition**: Number of arrivals in the time interval $[0, t]$ from a single replication.
2. **Interval Width** $W_n^k(\tau_0)$
   - **Definition**: Width of the time interval containing the k nearest neighbor arrival times around τ_0.

A1 Arrival intensity $\lambda_t > 0$ and

$$\Pr\{N^w(t) \geq 1\} = \lambda_t w + o(w), \quad \Pr\{N^w(t) \geq 2\} = o(w).$$

A2 Finite second moment: $\mathbb{E}[Y^2(t)] < \infty$.

A3 Mean function $v(t)$ and variance $\sigma^2(t)$ are Lipschitz continuous.

A4 $k \to \infty, \quad k/n \to 0.$

Key Lemma 1: Shrinking Window Width

Lemma 1: If $k/n \to 0$, then the smallest interval containing k neighbors converges to zero:

$$W_n^k(\tau_0) \xrightarrow{L^2} 0, \quad W_n^k(\tau_0) \xrightarrow{a.s.} 0.$$

Interpretation: Nearest neighbors concentrate around τ_0, enabling accurate local estimation.

Key Lemma 2: Dynamic Neighbor Selection

Lemma 2: The probability that any fixed t_{ij} appears infinitely often in the kNN set is zero:

$$\Pr\{t_{ij} \text{ is selected infinitely often}\} = 0.$$

Meaning: As more data arrive, kNN selections adaptively update and diversify.

Key Theorem 1: Asymptotic Independence

Theorem 1: If $k/n \to 0$, then:

$$\Pr\{\text{All } k \text{ neighbors come from distinct replications}\} \to 1.$$

Consequence: kNN observations behave asymptotically like independent samples.

Theorem 2: Asymptotic Consistency

Under assumptions A1–A4:

$$\bar{V}(\tau_0) \xrightarrow{p} v(\tau_0).$$

Justification:

- Lemma 1 ensures neighbors are close.
- Theorem 1 ensures effective independence.
- Devroye (1981): consistency holds for kNN under $k/n \to 0$.

Theorem 3: Asymptotic Unbiasedness

Assuming $v(t)$ is Lipschitz with constant L, we have:

$$\left|\mathbb{E}[\bar{V}(\tau_0)] - v(\tau_0)\right| \leq L \cdot \mathbb{E}[W_n^k(\tau_0)] \to 0.$$

Conclusion: Estimator bias vanishes as neighborhood shrinks.

Summary of Asymptotic Properties

- **Consistency**: Estimator converges in probability to true value.
- **Unbiasedness**: Expectation approaches the truth as $n \to \infty$.
- **Variance**: Scales properly under mild smoothness assumptions.
- **Implication**: Justifies use of kNN for virtual statistics in large-sample regimes.

3.Error Measurement

- Sample Variance Method
- Bootstrap Variance Method
- Theoretical Comparison of Sample vs Bootstrap Methods
- Performance Comparison of Sample and Bootstrap Methods

Sample Variance Method

In this section, we explore two methods for measuring the error of the kNN estimator: the **sample variance method** and the **bootstrap variance method**. These methods are used to quantify the variability of the kNN estimator.

- **Sample Variance Method:**
  - This method estimates the variance of the kNN estimator by using the sample variance formula.
  - Specifically, for the estimator $\tilde{V}(\tau_0)$, the sample variance is given by:

$$\hat{\tau}^2_{n,k}(\tau_0) = \frac{1}{k(k-1)} \sum_{\ell=1}^{k} \left[Y(\tau_0^{(\ell,n)}) - \tilde{V}(\tau_0) \right]^2.$$

  - The sample variance is based on the assumption that the data points are independent. However, this assumption is often violated in simulation data where there is correlation between observations from the same replication.

Bootstrap Variance Method

- **Bootstrap Variance Method:**
  - The bootstrap method resamples the data and computes the variance of the kNN estimator using these resamples.
  - Specifically, the bootstrap variance is computed as:

$$\tau^{*2}_{n,k,B}(\tau_0) = \frac{1}{B-1}\sum_{b=1}^{B}\left[\tilde{V}^{*}_{n,k,b}(\tau_0) - \frac{1}{B}\sum_{l=1}^{B}\tilde{V}^{*}_{n,k,l}(\tau_0)\right]^2.$$

  - Here, B denotes the number of bootstrap samples, and $\tilde{V}^{*}_{n,k,b}(\tau_0)$ represents the kNN estimator for the b-th bootstrap sample.

Theoretical Comparison of Sample vs Bootstrap Methods

- **Theorem:** Asymptotic Bias and Variance of the Sample and Bootstrap Methods.
  - **Sample Method:** The sample variance method is asymptotically biased due to the dependency between samples in the same replication. As a result, it tends to **underestimate** the true variance in small sample settings.
  - **Bootstrap Method:** The bootstrap method is more robust in the presence of dependent data. It **corrects for the correlation** between observations from the same replication and thus provides a more accurate estimate of the variance, particularly in small samples.

Performance Comparison of Sample and Bootstrap Methods

- **Performance of Sample Method:**
  - The sample method is computationally simple but is prone to underestimating variance when there is correlation within data.
  - This method works well in situations where data points are nearly independent or the sample size is large enough to minimize the impact of dependencies.
- **Performance of Bootstrap Method:**
  - The bootstrap method provides a more accurate estimate of the variance, particularly when there is dependence within the data, which is typical in simulation outputs.
  - It is computationally more expensive due to the need to resample the data multiple times, but it leads to more reliable results, especially in small sample settings.

4.Algorithm: LORO-CV

- Cross-Validation Methods
- Why Choose LORO-CV?
- Algorithm implementation

Cross-Validation Methods(Part 1)

Cross-validation (CV) is an essential technique for selecting the optimal number of nearest neighbors k in k-NN estimators.

- **Traditional K-fold Cross-Validation:**
  - Data is split into K folds, and one fold is used for testing, while the remaining $K-1$ folds are used for training.
  - This method assumes that the data points are independent, but this assumption is often violated in simulations where data points are correlated within replications.
  - **Limitation:** The random split can lead to **biased results** when there is correlation between the observations within the same replication.

Cross-Validation Methods(Part 2)

- **Leave-One-Out Cross-Validation (LOO-CV):**
  - LOO-CV involves leaving one observation out of the training set and using it for testing. This is repeated for each observation.
  - **Limitation:** This method is computationally expensive, especially for large datasets, and does not address the correlation issue in simulation data.
- **Leave-One-Replication-Out Cross-Validation (LORO-CV):**
  - LORO-CV is designed specifically for simulation data, where observations from the same replication are correlated.
  - In LORO-CV, an entire replication is left out of the training set, and the model is tested on this left-out replication.
  - **Advantage:** This method ensures that training and testing sets are independent, preventing the bias caused by correlated data within the same replication.

Why Choose LORO-CV?

- Independence of Training and Testing Sets:
  - LORO-CV ensures that the training set and testing set are independent by leaving out entire replications for testing, which reflects the structure of simulation data.
- Prevents Bias:
  - Traditional CV methods, such as K-fold CV, assume independence between data points. This assumption is often violated in simulations, leading to biased estimates of k.
- Computationally Feasible:
  - While LORO-CV is computationally more intensive than traditional methods, it is still feasible for moderate numbers of replications (e.g., 10 to 100 replications).
- Optimal k:
  - By using LORO-CV, we can select the optimal k that minimizes the empirical mean squared error (EMSE), leading to more accurate estimates of virtual performance.

Algorithm implementation

Algorithm 1 kNN Estimation via LORO-CV

Input: Search range $k_L < k_U$, NN = "nearest neighbors."
for $j = 1, 2, \ldots, n$ **do** ▷ Loop through all replications
  $S_{\text{test}} \leftarrow \{Y(t_{ij}), t_{ij}; i = 1, 2, \ldots, M_j\}$ ▷ Set aside one replication as test set
  $S_{\text{train}} \leftarrow$ all data except S_{test} ▷ Use remaining data as training set
  Find k_U nearest neighbors in S_{train} for each $t_{ij} \in S_{\text{test}}$
  Store indices of k_U nearest neighbors into $\mathbf{M}_{\text{ind}} \in \mathbb{R}^{|S_{\text{test}}| \times k_U}$
end for
for $k = k_L$ to k_U **do** ▷ Search for optimal k
  Extract the first k columns from $\mathbf{M}_{\text{ind}}$
  Find k-nearest neighbors for each $t_{ij} \in S_{\text{test}}$ and compute $\tilde{V}(t_{ij}, k)$
end for
for $k = k_L$ to k_U **do** ▷ Compute the EMSE for each k
  Compute the empirical mean squared error (EMSE):

$$\text{EMSE}(k) = \frac{\sum_{j=1}^{n} \sum_{i=1}^{M_j} \left[Y_{ij} - \tilde{V}(t_{ij}, k) \right]^2}{\sum_{j=1}^{n} M_j}$$

end for
Output: k^* that minimizes EMSE(k)

5.Experimental Results

- Experimental Framework and Research Logic
- Queueing Models and Distribution Specifications
- Key Case Studies and KFEs Validation
- LORO-CV Tuning Performance
- Bootstrap vs Sample Variance Performance
- kNN vs k-of-1nn Performance Comparison

Experimental Framework and Research Logic

The authors systematically evaluated the kNN estimator's performance through carefully designed experiments focusing on three key aspects:

- **Nonstationarity Handling**: Testing under time-varying arrival processes ($H_2(t)$ and $E_2(t)$) to assess dynamic adaptation.
- **System Variability**: Comparing high-variance ($H_2(t)/M/s/c$) vs. low-variance ($E_2(t)/M/s/c$, $E_4(t)/E_4/s/c$) systems.
- **Data Dependency**: Evaluating the impact of within-replication correlation through bootstrap vs. sample variance methods.

Core Comparison Dimensions:

- Accuracy of kNN against ground truth (via KFEs)
- Performance of LORO-CV-tuned k^* vs. alternative methods
- Error estimation: Bootstrap vs. sample variance

Queueing Models and Distribution Specifications

The experiments employed three phase-type queueing systems with distinct characteristics:

1. $H_2(t)/M/s/c$ **Model**
   - **Arrival Process**: Time-varying hyperexponential (H_2) distribution

$$f(t) = p\lambda_1 e^{-\lambda_1 t} + (1-p)\lambda_2 e^{-\lambda_2 t}, \text{ cv} > 1$$

   - **Purpose**: Simulates bursty arrivals with over-dispersion
2. $E_2(t)/M/s/c$ **Model**
   - **Arrival Process**: Time-varying Erlang-2 (E_2) distribution

$$f(t) = \lambda^2 t e^{-\lambda t}, \text{ cv} = 1/\sqrt{2}$$

   - **Purpose**: Represents regular arrivals with under-dispersion
3. $E_4(t)/E_4/s/c$ **Model**
   - **Full PH Structure**: Both arrivals and services follow Erlang-4
   - **Purpose**: Tests low-variance systems with non-exponential dynamics

Key Case Studies and KFEs Validation

Core Experimental Cases (Selected from Table 1):

Case	Model	μ	s	n
2	$H_2(t)/M/1/50$	20	1	10-100
2	$E_2(t)/M/1/50$	20	1	10-100
7	$E_4(t)/E_4/1/50$	20	1	10

Validation via Kolmogorov Forward Equations (KFEs):

- Derived exact virtual waiting time $v(\tau_0)$ by solving:

$$\frac{dP(t)}{dt} = P(t)Q(t)$$

  where $Q(t)$ is generator matrix encoding transition rates
- Achieved numerical precision $< 10^{-6}$ via Python ODE solver
- Served as ground truth for kNN estimator evaluation

LORO-CV Tuning Performance

Optimal k^* Selection (Table 2):

n	$H_2(t)/M/1/50$	$E_2(t)/M/1/50$
10	197	136
100	509	418

Key Findings:

- Larger k^* for high-variance systems (H_2 vs E_2)
- $k^* \propto n$ but sublinearly ($k^*/n \downarrow$ as $n \uparrow$)
- Achieved MSE < 0.01 vs KFE ground truth when $n = 100$

Bootstrap vs Sample Variance Performance

Error Estimation Comparison:

- **Bootstrap Variance** ($B = 2000$ resamples):

$$\hat{\sigma}^2_{\text{boot}} = \frac{1}{B-1} \sum_{b=1}^{B} (\bar{V}_b - \bar{V})^2$$

- **Sample Variance**:

$$\hat{\sigma}^2_{\text{sample}} = \frac{1}{k(k-1)} \sum_{i=1}^{k} (Y_i - \bar{V})^2$$

Results for $H_2(t)/M/1/50$ ($n = 10$):

- Bootstrap coverage: 92% (95% CI expected)
- Sample variance coverage: 63% (severe undercoverage)
- RMS error ratio: $\hat{\sigma}_{\text{boot}}/\hat{\sigma}_{\text{sample}} = 2.1$

kNN vs k-of-1nn Performance Comparison

Methodology Contrast:

- **kNN**: Selects k nearest neighbors from all replications
- k-**of-1nn**: Takes 1 nearest neighbor per replication, then selects k

EMSE Comparison (Table 3 Excerpt):

τ_0	$H_2(t)/M/1/50$		$E_2(t)/M/1/50$	
	kNN	k-of-1nn	kNN	k-of-1nn
5	0.090	0.326	0.095	0.284
10	0.165	0.467	0.198	0.538
15	0.153	0.437	0.197	0.463

Superiority Factors:

- Density advantage: kNN uses $\sim n \times k$ candidates vs n for k-of-1nn
- Bias reduction: Neighbors closer to τ_0 in absolute time
- Variance control: Larger effective sample size

6.Conclusion

- Contributions
- Limitations
- Future Work

Conclusion

- **Contributions**:
  - kNN method: Flexible, data-driven estimation of virtual performance in nonstationary systems.
  - Bootstrap variance: Reliable uncertainty quantification, outperforms sample variance.
  - Practical validation: Robust across queueing models, moderate storage needs (e.g., 505 MB for 100 replications).
- **Limitations**:
  - Sparse data: Reduced accuracy at time boundaries or low arrival density.
- **Future Work**:
  - Extend to other metrics: Apply kNN to virtual queue length, loss probability.
  - Data compression: Optimize storage for large-scale simulations.