Simulation Analytics - A kNN Model

Graham Laidler[1], Lucy Morgan[1], Barry Nelson[2], Nicos Pavlidis[1]

[1]Lancaster University, [2]Northwestern University

STOR-i
excellence with impact

EPSRC
Pioneering research and skills

The Motivation

Common Random Numbers (CRN) is a widely used simulation technique to improve the reliability of comparisons between systems. In each replication, common random number streams ensure that each system is viewed under the same conditions.

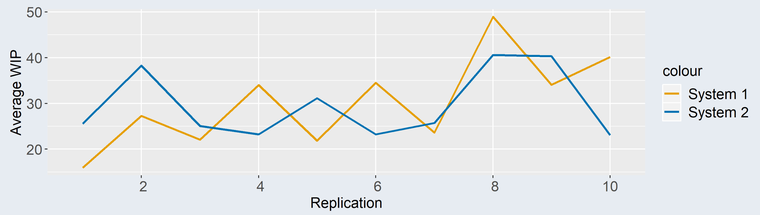

Figure 1: Which system performs best varies from replication to replication.

Figure 1 shows that the two systems must be responding differently to certain conditions that arise in the different replications. For example, the two systems may be affected differently by:

- A burst of arrivals
- An unusually long service time, or sequence of such
- ...More generally, particular features of the random input processes coinciding with the system state.

The Objective

We want to model how features of the system state and the input processes affect time-dynamic system performance. For the simulation model in Figure 2, we could have the following **predictors:**

- Queue i size
- Server i working (1 or 0)
- Server i blocked (1 or 0)

} System state

- Service time at station i
- Arrivals in next Δt minutes

} Random inputs

Each time we record these predictors, we can record a dynamic system **response**. For example:

- Change in WIP over the next Δt minutes.
- Average cycle time for items arriving in the next Δt minutes.
- Proportion of the next Δt minutes for which a server is blocked.

What is Simulation Analytics?

Traditionally, simulation output has been focused on **time-averaged** performance measures, which masks the **time-dynamic** behaviour of the actual system performance. The aim of simulation analytics is to unlock **deeper insights** into system behaviour by developing **machine learning** methods to apply to **sample path data**.

The Simulation

Figure 2: The three-server M/M/1 tandem queue

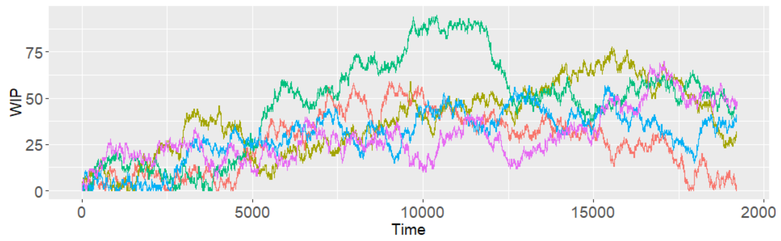

Figure 3: Sample path of WIP (work-in-process) for 5 replications of the system with buffer allocation $(\infty, 1, 0)$ and parameters $\lambda = 0.125,\ \mu_1 = \mu_2 = \mu_3 = 0.2$.

The Model - kNN

For a set of predictors, a kNN **regression** model **predicts** the response by averaging the response values of the k **most similar** sets of predictors in the **training** data.

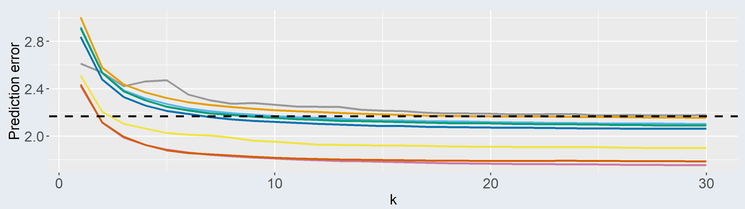

Figure 4: CV error of a kNN model, using different subsets of predictors.

The Challenge

The kNN method relies on a concept of **similarity**, or **distance**, between predictor vectors.

- Figure 4 suggests that some predictors have more prediction value than others, so we want a distance metric that reflects this.
- Also, our set of predictors may include variables of **different types**, so we require some pre-processing before we can apply a distance metric.

```
"character" "numeric" "integer" "logical"
```

The Method - Metric Learning

The task of **learning** a distance metric tuned to a **specific task** is known as metric learning. A **Mahalanobis** distance function has the form

$$d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T A (\boldsymbol{x}_i - \boldsymbol{x}_j).$$

To train a metric learning algorithm, we need to tell it pairs of points which are **similar** and **dissimilar**:

$$\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are similar}\},$$
$$\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are dissimilar}\}.$$

Our learned distance metric should respect these relationships. For example, as an optimization problem:

$$\min_{A \succeq 0} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{S}} d_A(\boldsymbol{x}_i, \boldsymbol{x}_j)$$
$$\text{s.t.} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{D}} d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 1.$$

If we restrict A to be a **diagonal matrix**:

$$A = \begin{pmatrix} A_{11} & & 0 \\ & \ddots & \\ 0 & & A_{pp} \end{pmatrix} \longleftarrow \begin{array}{l} \text{the diagonal element } A_{ii} \\ \text{represents the } \textbf{weight} \\ \text{(prediction value) for variable } i. \end{array}$$

- The weights A_{ii} will indicate which are the **driving factors** of system performance.
- A **challenge** is in defining the sets $\mathcal{S}$ and $\mathcal{D}$. We need to consider the response values whilst respecting the underlying topology of the predictor space.

www.lancaster.ac.uk/~laidler1

g.laidler1@lancaster.ac.uk