Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

METRIC LEARNING FOR SIMULATION ANALYTICS

Graham Laidler
Lucy E. Morgan
Barry L. Nelson
Nicos G. Pavlidis

Statistics and Operational Research Centre
for Doctoral Training in Partnership with Industry
Lancaster University
Lancaster, LA1 4YR, UK

ABSTRACT

The sample path generated by a stochastic simulation often exhibits significant variability within each replication, revealing periods of good and poor performance alike. As such, traditional summaries of aggregate performance measures overlook the more fine-grained insights into the operational system behavior. In this paper, we take a simulation analytics view of output analysis, turning to machine learning methods to uncover key insights from the dynamic sample path. We present a k nearest neighbors model on system state information to facilitate real-time predictions of a stochastic performance measure. This model is built on the premise of a system-specific measure of similarity between observations of the state, which we inform via metric learning. An evaluation of our approach is provided on a stochastic activity network and a wafer fabrication facility, both of which give us confidence in the ability of metric learning to provide interpretation and improved predictive performance.

1 INTRODUCTION

Analysis of stochastic simulation has long been centered around the evaluation of aggregate performance measures. This paper is motivated by the belief that static summaries such as these can represent a limited view of a highly dynamic and possibly non-stationary process. Dynamic performance measures such as waiting times or congestion levels are seen to fluctuate throughout simulated replications. Revealing the factors that drive these fluctuations is key to a deeper understanding of the represented system. With this aim, we propose a dual-purpose methodology for output analysis designed to support real-time predictions, whilst simultaneously revealing insight into the key drivers of dynamic performance.

Our work falls into the newly developing area of simulation analytics, first suggested by Nelson (2016). In this area, simulation is regarded as a generator of dynamic sample paths, from which data analytics and machine learning tools can glean insights into the conditional relationships and dependencies that characterize the system behavior. We are aided in this approach by recent advances in data storage to enable cheap and effectively unlimited retention of sample path data. Indeed, many commercial simulation products currently retain a record of the events and state transitions that occur throughout a replication, albeit primarily for the purpose of debugging. Crucially, the capability is there, and whilst we are not dealing with the technicalities of how to post-process and store system traces, we are considering ways to exploit the opportunity for deeper analysis that such a detailed transaction log presents. To add further motivation and plausibility, we note that the size of datasets generated by sample paths of discrete-event simulation will typically not approach the volumes associated with "big data" in modern-day analytics.

To motivate the scope of our work, consider for example a doctors' surgery seeking to evaluate the performance of different staffing schedules. For this system, performance indicators such as patient waiting times or staff utilization can be used to rank alternatives (Brailsford 2007). Whilst the daily averages of these indicators can allow an initial screening of solutions, they provide an incomplete picture of a system that is likely to exhibit significant variability throughout the day. A more probing analysis may relate to the systems' robustness to certain conditions such as a higher than usual demand for prescription medicines. Moreover, when our simulated systems exhibit periods of poor performance, we want to understand the cause. If we can reveal, for example, that the number of patients awaiting a blood test constitutes a driving influence behind variable waiting times, this is a useful insight which might suggest a more efficient allocation of resources. To enable such analyses, we build a predictive model for a dynamic system response, the structure of which is designed to expose the key factors which drive this response.

Specifically, our predictive model takes the form of a non-parametric k-nearest-neighbor (kNN) classifier on the system state. Our choice here is motivated by the understanding that components of the state in a simulation model interact jointly in a way that is difficult to capture with parametric functions. Deferring a definition of system state to the later sections and appealing to the example above, we might include variables representing the number of patients undergoing or awaiting different treatments, or the number of nurses on duty. We let $\boldsymbol{x} \in \mathscr{X} \subset \mathbb{R}^d$ denote the system state at a given time, and construct a measure of similarity over the space of $\mathscr{X}$. Each instance $\boldsymbol{x} \in \mathscr{X}$ carries with it an observed system response, y, which we measure as a categorical variable. For example, a patient entering the surgery in state $\boldsymbol{x}_i$ experiences the waiting time y_i, classed as above or below average. Our stored sample paths provide many observations of $(\boldsymbol{x}_i, y_i)$, which create our training data. Taking a kNN approach, we can classify the waiting time of a patient arriving to the system in state $\boldsymbol{x}^*$ according to the observed waiting time categories of the k nearest instances to $\boldsymbol{x}^*$ among the training data.

A key aspect of our methodology is in defining an appropriate measure of distance, or similarity, between observations in $\mathscr{X}$. For this, we delve into a rich literature on metric learning. The process of tuning a system-specific distance function is helpful in revealing interactions among the state variables and their relative contributions towards the system response. Thus, combining metric learning with kNN classification allows us to join interpretability with predictive performance. This dual benefit lends optimism to the scope of our work, which we believe extends to researchers and practitioners alike. In particular, whilst the application of metric learning serves to highlight the performance-dictating components and interactions within a system, the ability to make real-time predictions represents a fundamental step towards using simulation for system control.

The structure of the paper is as follows. In Section 2 we discuss related work and provide a background for kNN classification and metric learning. Section 3 presents results to motivate and demonstrate our methodology in the context of simulation, before we conclude with a brief summary in Section 4.

2 BACKGROUND

In this section, we establish a background for our work, drawing on related work in the area of simulation analytics and describing our proposed application of kNN. We also introduce the field of metric learning, providing a focused review of the relevant literature.

2.1 Related Work in Simulation Analytics

The possibility of using simulation to inform real-time decision problems has recently begun to draw attention. A number of papers have emerged in which simulation sample paths are stored and used to build metamodels for making dynamic predictions. In the context of a queueing network, Ouyang and Nelson (2017) proposed a two-stage logistic regression modelling approach in which the state and time aspects of the sample path are treated separately, while Jiang et al. (2020) use a logistic regression model to dynamically predict the risk of financial portfolios. Wu and Barton (2016), meanwhile, show that Fourier analysis can successfully detect changes in the dynamic trajectories of system state variables to discriminate between congested and uncongested systems.

The approaches outlined above represent attempts to model sample path behavior within a parametric framework. In reality, we understand simulation to be a complex stochastic process in which the dynamic and possibly non-stationary behaviour is difficult to capture in a parametric model. Accordingly, Lin et al. (2019) suggest a kNN approach to provide predictions for time-dependent mean performance measures. They describe this as “virtual performance”, and further consider the behavior of its higher order moments (Lin and Nelson 2018). An example of virtual performance is given by the waiting time of a customer in a service system conditional on arriving to the system at time t. In our own take on virtual performance, the conditioning event relates to the state of the system; modifying this example, we consider the waiting time of a customer conditional on arriving to the system in state $\boldsymbol{x}$. In the case of Lin et al. (2019), the kNN estimator of virtual performance is one-dimensional in the sense that neighbors are determined by a single variable: simulation time. Our own kNN model, meanwhile, takes a more comprehensive view of ‘neighbors’, accommodating multiple predictors in an attempt to fully characterize the state of a system. Importantly, we note that a multi-dimensional state description is likely to contain variables which are not immediately comparable in terms of scale or interpretation. On these grounds, we recognise that identifying neighbors based on simple Euclidean distance, whilst effective in the one-dimensional setting of Lin et al. (2019), will not be appropriate for us.

2.2 *k*NN Classification

A kNN classification model provides a simple rule whereby instances are classified according to the labels of their k nearest neighbors (Hastie et al. 2009). In this paper, we consider a binary classification task. Our training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ are obtainable directly from the stored sample paths. These observations are functions of the simulation time in a given replication; $\boldsymbol{x}_i \in \mathbb{R}^d$ represents the system state at time t_i, and $y_i \in \{0,1\}$ denotes an associated system performance measure, which in general may be observable at a later time. We use the term “system state” at time t to refer to some subset of the information generated by the simulation up to time t.

Using kNN classification to classify an instance $\boldsymbol{x}^*$, we identify its k nearest training instances, denoted by $\boldsymbol{x}^{*(1)}, \ldots, \boldsymbol{x}^{*(k)}$, and their corresponding labels $y^{*(1)}, \ldots, y^{*(k)}$. Our classification rule is defined as a function of $c \in [0,\infty)$:

$$\hat{y}^* = \begin{cases} 1 \text{ if } \frac{1}{k}\sum_{i=1}^k y^{*(i)} \geq c, \\ 0 \text{ if } \frac{1}{k}\sum_{i=1}^k y^{*(i)} < c. \end{cases} \tag{1}$$

The classification threshold $c = 1/2$ corresponds to a typical majority rule, although in general we can choose c to minimize some error criterion such as mean squared error (MSE) on a test set, or to represent a desired trade-off between the two types of misclassification.

A nearest neighbor classifier relies upon the assumption that instances which are similar to one another in the input space will yield a similar classification in the output space. Intuitively, the truth of this assumption requires that the distance measure used to identify neighbors reflects some domain-specific notions of similarity between input instances. Selecting a relevant distance measure to use with kNN classification therefore requires careful consideration of the problem domain. Whilst Euclidean distance is often viewed as a default, significant advantage can be gained by using a more tailored metric (Kulis 2013). This leads us to the topic of metric learning, which provides a data-driven way to automate the process of defining a suitable distance metric.

2.3 Mahalanobis Metric Learning

We recall our data of the form $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i$ is a d-dimensional vector of predictor variables and y_i its associated class label. The aim of metric learning is to adapt a distance function over the space of predictor vectors. As common in the metric learning literature, we consider the family of Mahalanobis distance functions. These take the form

$$d_M(\boldsymbol{x}_i,\boldsymbol{x}_j) = [(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top M(\boldsymbol{x}_i - \boldsymbol{x}_j)]^{1/2}, \tag{2}$$

parametrized by $M \in \mathbb{S}^d_+$, where $\mathbb{S}^d_+$ denotes the set of d-dimensional symmetric positive semi-definite matrices. This condition ensures that d_M satisfies the properties of a pseudometric (Bellet et al. 2014). We note here that the identity matrix $M = I_d$ recovers the standard Euclidean distance.

A positive semi-definite matrix, M, will always permit the decomposition $M = L^\top L$. This allows us to write the Mahalanobis distance (2) as $d_M(\boldsymbol{x}_i,\boldsymbol{x}_j) = [(L\boldsymbol{x}_i - L\boldsymbol{x}_j)^\top(L\boldsymbol{x}_i - L\boldsymbol{x}_j)]^{1/2}$. Hence, we can understand the metric d_M to be equivalent to the Euclidean metric after a linear transformation of the data defined by L. This is a useful relationship which suggests two different parametrizations for the metric learning problem. The optimization task can be performed with respect to the matrix $M \in \mathbb{S}^d_+$ or the linear transformation L. Metric learning methods have been proposed from both perspectives, with each offering their own advantages. Optimization over M is generally favoured as it leads to convex formulations which can be solved more efficiently. However, learning the transformation matrix L allows for rank constraints to be directly imposed. In general, $L \in \mathbb{R}^{r\times d}$, where $r \leq d$ is the rank of L and M. Learning a low rank matrix brings the data into a transformed space of fewer dimensions, which offers advantages when the original dimension, d, is large.

The metric learning task is typically supervised by a collection of constraints summarizing our prior intuition about the relative distances that we wish to emerge between training instances. Often, this supervision comes in the form of the following sets:

$$\mathscr{S} = \{(\boldsymbol{x}_i,\boldsymbol{x}_j) \mid \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ should be close}\} \text{ (similarity constraints)},$$
$$\mathscr{D} = \{(\boldsymbol{x}_i,\boldsymbol{x}_j) \mid \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ should not be close}\} \text{ (dissimilarity constraints)},$$
$$\mathscr{R} = \{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_l) \mid \boldsymbol{x}_i \text{ should be closer to } \boldsymbol{x}_j \text{ than it is to } \boldsymbol{x}_l\} \text{ (relative similarity constraints)}.$$

In the absence of particular intuition, these sets can be derived from the class labels of the training instances. For example, pairs of similarly labelled instances should populate $\mathscr{S}$, while pairs of differently labelled instances can populate $\mathscr{D}$. Intuitively, triplets $(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_l)$ with $y_i = y_j \neq y_l$ might populate the set $\mathscr{R}$. Whilst these sets are assumed to be given, their specific construction should be relevant to the data and the application at hand. The task of selecting appropriate constraint sets is treated as a learning task itself by Wang et al. (2012), although in general these sets are assumed to be given, and remain fixed throughout the metric learning procedure.

In general, the metric learning problem is an optimization of the form $\min_{M\in\mathbb{S}^d_+} \ell(M,\mathscr{S},\mathscr{D},\mathscr{R}) + \lambda r(M)$. Here, ℓ is a loss function to penalize violations of the training constraints under the metric d_M, and $r(M)$ describes some regularization on the values of M, with $\lambda \geq 0$ the regularization parameter. The main distinctions among different metric learning methods arise from their choices of loss function and regularization.

The earliest method for Mahalanobis metric learning is attributed to Xing et al. (2002). The intuitive formulation seeks to minimise the sum of squared distances in $\mathscr{S}$ whilst keeping the sum of distances in $\mathscr{D}$ above a threshold:

$$\min_{M\in\mathbb{S}^d_+} \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathscr{S}} d^2_M(\boldsymbol{x}_i,\boldsymbol{x}_j) \quad \text{s.t.} \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathscr{D}} d_M(\boldsymbol{x}_i,\boldsymbol{x}_j) \geq \gamma.$$

To solve this optimization, the authors proposed a gradient based algorithm with iterative projections onto $\mathbb{S}^d_+$ maintaining the positive semi-definite constraint. Whilst the primary motivation for this work was an application to clustering, the method has proved useful to many machine learning algorithms, paving the way for metric learning to be viewed as a convex optimization.

Following this formulation, several more tailored methods have emerged. With an objective function inspired by the task of nearest neighbor classification, Goldberger et al. (2004) proposed a method referred to as Neighbourhood Components Analysis (NCA). They compute a softmax version of the probability that $\boldsymbol{x}_i$'s nearest neighbor is from the same class, $p_i = \sum_{j:y_j=y_i} \exp(-d_M^2(\boldsymbol{x}_i,\boldsymbol{x}_j))/\sum_{l\neq i}\exp(-d_M^2(\boldsymbol{x}_i,\boldsymbol{x}_l))$. Their objective function, which is to maximise the sum of these probabilities over all training points, equates to learning the distance metric which minimises the expected leave-one-out error rate of the nearest neighbor classifier. To aid the gradient calculation, the optimization was presented in terms of the transformation matrix L. The objective function of NCA is therefore non-convex, making the method susceptible to finding only local maxima. However, a convex extension to NCA was proposed by Globerson and Roweis (2005).

A second approach tailored to the task of nearest neighbor classification was provided by the Large Margin Nearest Neighbor (LMNN) algorithm, developed by Weinberger and Saul (2009). The construction of the constraint sets for LMNN is motivated by the fact that success of kNN only relies on local clusterings of similarly labelled points, rather than a global clustering. Specifically, the concept of target neighbors is introduced, which, in the absence of prior knowledge, are defined for each training point as the k nearest points in Euclidean distance which share the same class label. Taking $\eta_{ij} \in \{0,1\}$ to indicate whether $\boldsymbol{x}_j$ is a target neighbor of $\boldsymbol{x}_i$, the sets $\mathscr{S} = \{(\boldsymbol{x}_i,\boldsymbol{x}_j) \mid \eta_{ij} = 1\}$, and $\mathscr{R} = \{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_l) \mid \eta_{ij} = 1 \text{ and } y_l \neq y_i\}$ are defined. The objective function of LMNN then takes the following form:

$$\min_{M\in\mathbb{S}_+^d} \ (1-\mu) \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathscr{S}} d_M^2(\boldsymbol{x}_i,\boldsymbol{x}_j) \ + \ \mu \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_l)\in\mathscr{R}} \max\{0, 1 + d_M^2(\boldsymbol{x}_i,\boldsymbol{x}_j) - d_M^2(\boldsymbol{x}_i,\boldsymbol{x}_l)\}.$$

This objective function seeks a local neighborhood of each instance that is populated with other instances of the same class, while those of a different class are repelled by a 'large' margin. The parameter $\mu \in [0,1]$ controls the trade-off between attracting target neighbors and repelling oppositely labelled instances. A number of extensions to LMNN have been proposed. Torresani and Lee (2006) explore kernel methods to combine LMNN with dimensionality reduction of the feature space, while Kedem et al. (2012) suggest extensions to a non-linear metric.

Whilst NCA and LMNN remain perhaps the most notable contributions in the direction of metric learning for nearest neighbor classification, many metric learning methods exist in the wider literature, offering different perspectives on the fundamental task. We refer the interested reader to the work of Kulis (2013) and Bellet et al. (2014) for a thorough review of established methods.

In this paper, we limit ourselves to considering a classification task. For this reason, the metric learning formulation and methods discussed above apply specifically to data in which the response variable is categorical. However, it should be recognised that kNN represents a versatile rule that readily extends to regression problems as well (see for instance Weinberger and Tesauro (2007)).

3 METRIC LEARNING FOR SIMULATION

In this section, we motivate our proposed methodology for metric learning and kNN classification on simulation models. Whilst many methods are available to perform metric learning, in the results presented here we used CVXR (Fu et al. 2018), an open-source convex optimization solver, to employ the original formulation of Xing et al. (2002). To specify the constraint sets, $\mathscr{S}$ and $\mathscr{D}$, we take a local neighborhood approach inspired in part by LMNN. We denote by $\mathscr{N}^{(q)}(\boldsymbol{x}_i)$ the set containing the q nearest points to $\boldsymbol{x}_i$ in Euclidean distance. Then, for all $\boldsymbol{x}_i$, and for all $\boldsymbol{x}_j \in \mathscr{N}^{(q)}(\boldsymbol{x}_i)$, we make the following assignments:

$$(\boldsymbol{x}_i,\boldsymbol{x}_j) \in \begin{cases} \mathscr{S} & \text{if } y_j = y_i, \\ \mathscr{D} & \text{if } y_j \neq y_i. \end{cases}$$

As noted by Weinberger and Saul (2009), the success of kNN requires only that the local neighborhood of each instance be populated by others of the same classification. Particularly in our simulation context, which allows a high-dimensional representation of system state, this will be relevant; a global clustering of each class may be inappropriate. Euclidean distance provides an initial view of the local neighborhoods of our training points. It stands to reason that pairs of nearby points in Euclidean distance with equivalent class labels should be encouraged to remain as neighbors. These points naturally fall close in the predictor space and they share the same classification; we can confidently describe them as 'similar'. On the other hand, nearby pairs with different classifications need to be forced apart, since they will impede the performance of our classifier. Moreover, owing to their different classifications, we assume that there is something fundamentally 'dissimilar' about these pairs of instances which is not detected by Euclidean distance. By placing these pairs in $\mathscr{D}$, we aim to learn a distance metric that is sensitive to these more subtle dissimilarities.

In the results presented in this section, we use the original metric learning formulation of Xing et al. (2002):

$$\min_{M \in \mathbb{S}_+^d} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{S}} d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \text{s.t.} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathscr{D}} d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \gamma.$$

We set the constraint constant to $\gamma = |\mathscr{D}|$. In other words, we aim to minimise the sum of squared distances in $\mathscr{S}$ whilst keeping the average distance in $\mathscr{D}$ from falling below 1. In reality, the choice of $\gamma > 0$ is unimportant, and, provided numerical stability of the optimization algorithm is maintained, a different choice results only in a scaling of the solution matrix by a constant factor. To generate the sets $\mathscr{S}$ and $\mathscr{D}$, we take a local neighborhood size of $q = 20$, and we evaluate the performance of kNN classification using $k = 50$ nearest neighbors. Whilst these values prove sufficient here to provide results supportive of our methodology, the optimal setting of the parameters q and k will in practice vary across systems, and can be selected via cross validation (Hastie et al. 2009).

We first consider a simple motivating example to illustrate the advantages of metric learning and kNN classification in a simulation context. We then evaluate a realistic application of our methodology on a more complex simulation of a wafer fabrication facility.

3.1 A Stochastic Activity Network

We consider the simple stochastic activity network represented in Figure 1. We denote the five activity times by $X_1, X_2, \ldots, X_5$, modeling each as an i.i.d. random variable, $X_i \sim \text{Exp}(1)$ for $i = 1, 2, \ldots, 5$. There are three possible paths through the network, such that the total time taken for completion is given by $T = \max\{X_1 + X_4, X_1 + X_3 + X_5, X_2 + X_5\}$. To bring ourselves into a classification setting, we consider the binary response $Y = 1$ if $T > 5$, and 0 otherwise.

Given the structure of this network, we understand that the path $X_1 \to X_3 \to X_5$, requiring three activities, will often represent the longest path through the network and hence define the value of Y. Of these three activity times, X_1 and X_5 appear in a second path also, giving them an edge over X_3 in terms of contribution towards the response. These two alternative paths suggest that some interaction effects will also exist between X_1 and X_4 and between X_2 and X_5; in each case, high values of both variables will encourage the response $Y = 1$.

We ran $n = 10,000$ replications of the network, recording the five activity times, $X_1, X_2, \ldots, X_5$, and the classification response, Y, for which the proportion of class 1 was around 16%. Knowing the exact mechanism by which the activity times generate the response gives us an understanding against which we can evaluate the output matrix of the metric learning. This output, M, provides a visual guide to the comparative relevance of the five activity times towards the response, and is shown in Figure 2. When

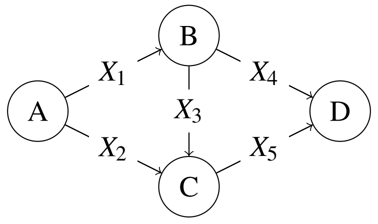


Figure 1: A small stochastic activity network.

used as the Mahalanobis distance matrix in (2), the diagonal elements in M indicate the weights given to differences in individual variables in defining the overall distance between two instances, whilst the off-diagonal elements additionally reflect relationships among the variables. We see immediately from Figure 2 that the largest diagonal elements correspond to the variables X_1, X_3, and X_5. Thus, under the metric d_M, instances with similar values of these three variables will be deemed more similar overall, with less importance given to their proximity in terms of X_2 and X_4. This reflects our understanding that $X_1 \to X_3 \to X_5$ is the most relevant path through the network. We can further note that the diagonal terms for X_1 and X_5 are slightly higher than for X_3, reflecting the additional contributions that these two variables make towards Y.

We turn our attention to the off-diagonal terms of M, and note that the positive semi-definite constraint imposes no restrictions on their sign. For the purpose of interpretation, we define $\boldsymbol{z}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$, and note from (2) that the off-diagonal element $M(k,l)$ appears in the contribution of the term $2\boldsymbol{z}_{ij}(k)\boldsymbol{z}_{ij}(l)M(k,l)$ to the squared Mahalanobis distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The magnitude of $M(k,l)$ indicates the impact of the product term $\boldsymbol{z}_{ij}(k)\boldsymbol{z}_{ij}(l)$ on $d_M(\boldsymbol{x}_i,\boldsymbol{x}_j)$, whilst its sign indicates the way in which these variables interact. Specifically, a positive value of $M(k,l)$ indicates that $d_M(\boldsymbol{x}_i,\boldsymbol{x}_j)$ will increase when $\boldsymbol{z}_{ij}(k)$ and $\boldsymbol{z}_{ij}(l)$ have the same sign, and decrease otherwise, whilst the opposite is true for negative $M(k,l)$. Applying this interpretation to Figure 2, we can understand the positive off-diagonal terms in connection with the additive relationship through which the variables generate the response, whilst the greatest strengths are understandably attributed to the relationships among the dominant variables X_1, X_3, and X_5. As such, we are encouraged to see that the matrix obtained by metric learning aligns with our intuitive understanding of this system. In realistic simulation models for which such intuition is less accessible, we suggest that metric learning can be effective in revealing relationships among system components, and their relative contributions towards driving the system performance.

The effectiveness of kNN classification using a learned distance metric is illustrated in Figure 3. The receiver operating characteristic (ROC) curves (Hastie et al. 2009) show the performance of the kNN classifier following 2-5-fold cross validation (CV). J-K-fold CV, consisting of J independent K-fold cross validations, is understood to represent a more robust procedure than traditional K-fold CV (Moss et al. 2018). Specifically in our experiments, each CV iteration takes four fifths of the data to comprise the training set on which the distance metric is learned, and the points in the remaining test set are classified by finding their k nearest neighbors from the training set, with respect to the learned distance metric. The ROC curves display the trade-off between the true positive rate and the false positive rate as the classification

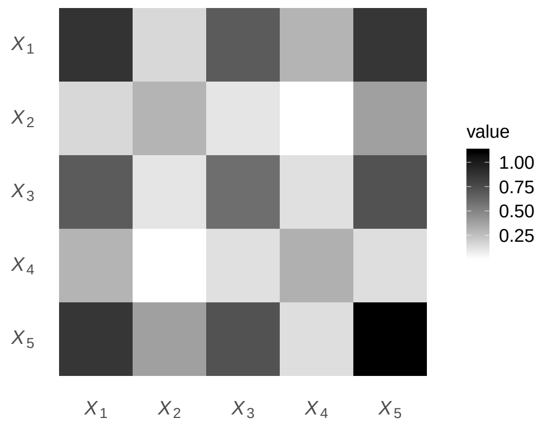


Figure 2: A visualization of the learned matrix, M, for the stochastic activity network.

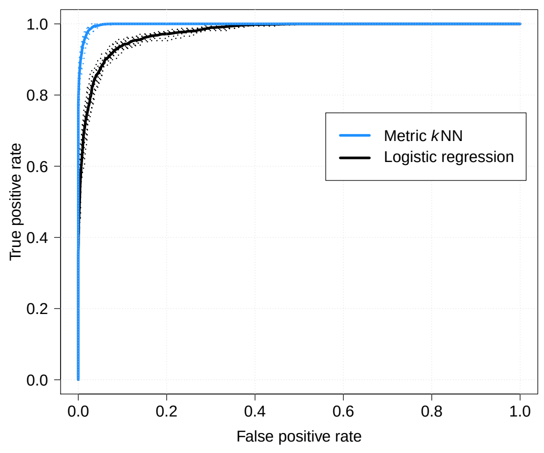


Figure 3: ROC curves for classification on the stochastic activity network.

threshold varies. Here, the true positive rate (sensitivity) refers to the proportion of class 1 points correctly classified, whilst the false positive rate (1− specificity) refers to the proportion of class 0 points incorrectly classified as class 1. Thus, in practice, the ROC curves can be used to select the classification threshold, c, in (1), based on a desired sensitivity-specificity trade-off. In Figure 3, the ten dashed CV curves are averaged at each threshold to produce the solid curves. Using the same CV partitions, we also show the ROC curves from logistic regression, a standard classification technique.

The comparison in Figure 3 reveals the value of the kNN approach. Logistic regression represents a parametric attempt to model the response as a function of the predictor variables. However, owing to the nature of the network, measuring similarity with regards to the key activity times proves a more successful foundation for prediction. The connected dependence structure among the predictors in this example is not untypical of the relationships among state variables in many simulation models.

Making a further appeal to the nature of simulation, we recognise that a multi-dimensional characterization of the system state is likely to include a number of variables that provide little or no contribution to the system response. To demonstrate the capability of metric learning in handling data of this nature, we augmented the five activity times with a further fifteen i.i.d. random variables, $X_i \sim \text{Exp}(1)$ for $i = 6, 7, \dots, 20$. The response, Y, depends only on $X_1, X_2, \dots, X_5$ as before, with these additional variables representing noise dimensions. Metric learning on this augmented data yields the matrix shown in Figure 4. We see that metric learning is able to successfully filter out the noise dimensions and recover the important structure among the five activity times. Figure 5 shows the ROC curves for this example, and also shows the performance of the kNN classifier with Euclidean distance, which is comparable to that of logistic regression.

We are encouraged to see in Figures 4 and 5 the capability of metric learning in the presence of noise variables, both in terms of retaining interpretability and bringing improvement to Euclidean kNN classification. This lends optimism to our proposed application, since we recognise that irrelevant variables will be a common feature of the large state descriptions of realistic simulation models. We progress now to one such model, in which we demonstrate further the capability of metric learning and its application to kNN classification.

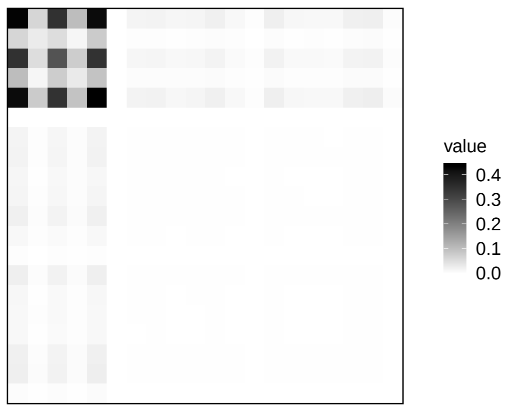


Figure 4: The learned matrix, M, after the data from the stochastic activity network is augmented with fifteen noise variables.

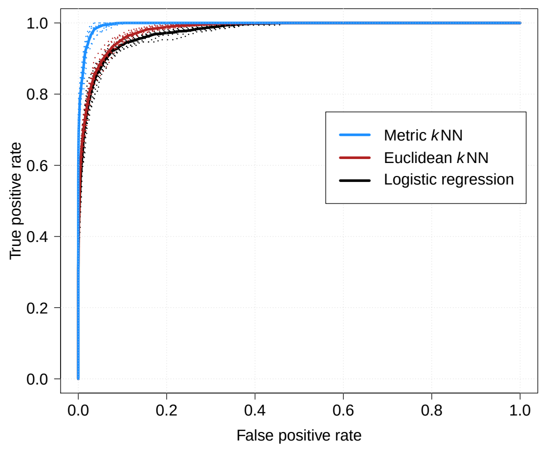


Figure 5: ROC curves for classification on the noise-augmented stochastic activity network.

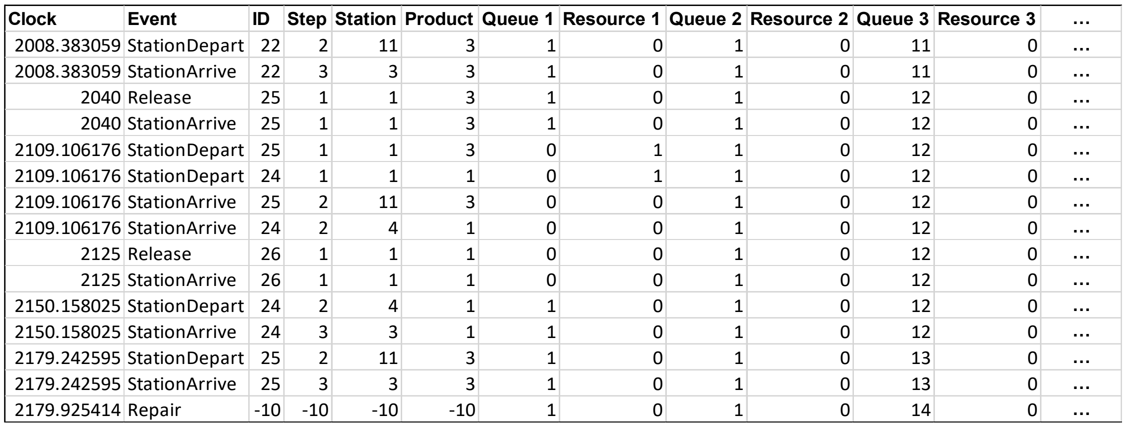
3.2 A Wafer Fab Model

To evaluate our approach with a more realistic simulation, we employ the model of a wafer fabrication facility (fab) described by Kayton et al. (1996). The manufacturing process of semiconductor wafers involves several processing steps at a number of stations. Machines with different processing capacities and unpredictable breakdown patterns present a challenge to the management of product flow through these facilities. Moreover, the layered nature of their circuitry design requires wafers to make multiple visits to particular stations, introducing an aspect of re-entrant flow that further complicates our view of the system.

Briefly, the simulation model is comprised of 11 stations with lognormal processing times. Three product types are produced by the facility, each requiring a specific routing sequence through the 11 stations. Notable stations include station 3, characterized by an unreliable machine, and station 4, which represents the bottleneck station to which products make repeated visits. Together with the varying processing behavior of the different stations, including batch processing at stations 1 and 2, these features establish a system with a level of complexity approaching that of typical simulation models.

On release into the system, each wafer is assigned a due date based on its expected processing time through an empty system. Therefore, we can consider observed completion times relative to due dates as a dynamic indicator of system performance. To describe the system state, we take a Markovian view in assuming that all information relevant to the future evolution of the system can be captured in the currently observable system conditions. Namely, we focus on the current values of the 22 integer variables describing the queue size and the number of resources in use at each station. We recognise that this state description is incomplete, given the different product types and processing stages of individual wafers. However, having tested numerous additional state descriptors, we find that a basic physical view of the system state is sufficient here to provide compelling support for our methodology. As such, our data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ are as follows. We record the 22-dimensional system state, $\boldsymbol{x}_i$, at the moment a wafer is released into the system, and observe its associated binary response, y_i, indicating whether this wafer was completed early or late with respect to its due date. To aid an understanding of our approach, Figure 6 shows an excerpt of the system trace information recorded by this simulation, from which we extract our system state description. In general, rather than trying to select a parsimonious state description, we suggest including all the observable state information that may prove relevant, and then allowing metric learning to discover what is actually relevant.

To avoid confusing the behaviors of the different product types, our data contains only observations from a single product type. In the results that follow, we combine observations from multiple replications of the system, discarding a warm-up period from each to leave us with insight into the operational steady-state behavior. The proportion of 'Late' responses in our data was around 72%. Additionally, the simulation



Clock	Event	ID	Step	Station	Product	Queue 1	Resource 1	Queue 2	Resource 2	Queue 3	Resource 3	...
2008.383059	StationDepart	22	2	11	3	1	0	1	0	11	0	...
2008.383059	StationArrive	22	3	3	3	1	0	1	0	11	0	...
2040	Release	25	1	1	3	1	0	1	0	12	0	...
2040	StationArrive	25	1	1	3	1	0	1	0	12	0	...
2109.106176	StationDepart	25	1	1	3	0	1	1	0	12	0	...
2109.106176	StationDepart	24	1	1	1	0	1	1	0	12	0	...
2109.106176	StationArrive	25	2	11	3	0	0	1	0	12	0	...
2109.106176	StationArrive	24	2	4	1	0	0	1	0	12	0	...
2125	Release	26	1	1	1	0	0	1	0	12	0	...
2125	StationArrive	26	1	1	1	0	0	1	0	12	0	...
2150.158025	StationDepart	24	2	4	1	1	0	1	0	12	0	...
2150.158025	StationArrive	24	3	3	1	1	0	1	0	12	0	...
2179.242595	StationDepart	25	2	11	3	1	0	1	0	13	0	...
2179.242595	StationArrive	25	3	3	3	1	0	1	0	13	0	...
2179.925414	Repair	-10	-10	-10	-10	1	0	1	0	14	0	...

Figure 6: An example of the trace from the wafer fab simulation, which we use to build a state description.

can be performed with a choice of dispatching rules, relating to the order in which queueing wafers are processed (Kayton et al. 1996). The results displayed in Figures 7 and 8 used a 'least work remaining' rule, in which priority is given to wafers which are nearer completion.

The metric learning procedure resulted in the matrix visualized in Figure 7. The stand-out elements in this matrix correspond to the queue sizes at stations 3 and 4, which we recall to coincide with the unreliable machine and the system bottleneck, respectively. The metric learning result highlights the significance of these two queue sizes in defining the overall system performance.

Performing *k*NN classification on this data, with 2-5-fold CV as in the previous example, yields the ROC curves shown in Figure 8. Compared to a Euclidean *k*NN classifier, the curves reveal the benefit to classification performance that metric learning brings. Essentially, as the size of the state space increases, we expect the benefit of metric learning over Euclidean distance to become even more pronounced.

As a statistical learning technique, *k*NN is best suited to low-dimensional data (Beyer et al. 1999). To visualize the effect of metric learning with respect to dimensionality reduction, it is convenient to consider the transformation matrix L given by the decomposition $M = L^\top L$. We recall that the metric d_M can be viewed as the Euclidean metric in the space transformed by L. Taking the eigen-decomposition of M, the ith row of L is given by $\lambda_i^{1/2}\boldsymbol{v}_i^\top$, where λ_i and $\boldsymbol{v}_i$ denote, respectively, the ith largest eigenvalue of M and its corresponding eigenvector. Thus, the eigenvalues of M directly impact the spread of our data in its transformed dimensions. Figure 9 shows the eigenvalues of the original data covariance matrix, in order of decreasing size, whilst Figure 10 shows the eigenvalues of M. We see that metric learning in this example results in much of the variability of the data being compressed into the first dimension. Hence, we can acknowledge the effective dimensionality reduction brought upon our data by metric learning. Projecting the transformed data against its first and second dimensions results in the plot shown in Figure 11. We can see that this dominant first dimension is effective in distributing our two response classes.

In formulations such as NCA, which directly seek a transformation of the feature space, dimensionality reduction can be straightforwardly enforced. However, even when not directly sought, the by-product of dimensionality reduction is almost inherent in the nature of the metric learning task. Since a nearest neighbor rule is understood to suffer the curse of dimensionality, and we aim to accommodate simulations with a high-dimensional state space, the dimensionality reduction encouraged by metric learning represents an important aspect of our methodology; it allows us to apply *k*NN without the need for user-intervention in trimming the state space.

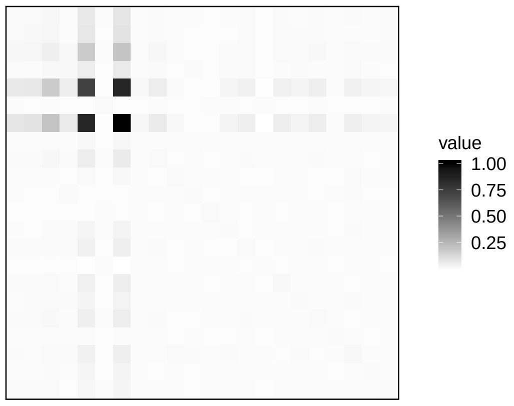


Figure 7: A visualization of the learned matrix, M, for the wafer fab problem.

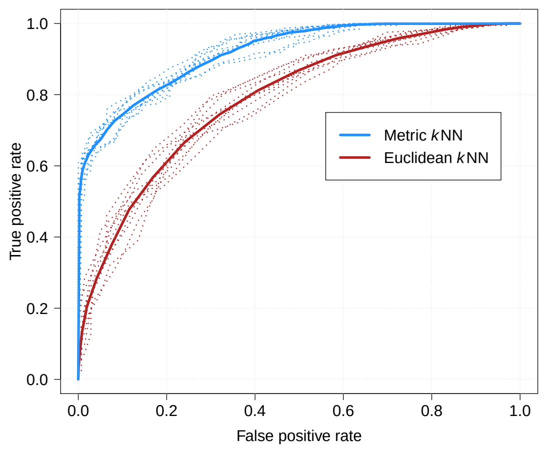


Figure 8: ROC curves for *k*NN classification on the wafer fab problem, comparing Euclidean distance with the learned metric.

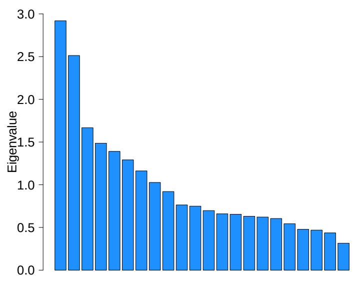


Figure 9: The eigenvalues of the original data covariance matrix.

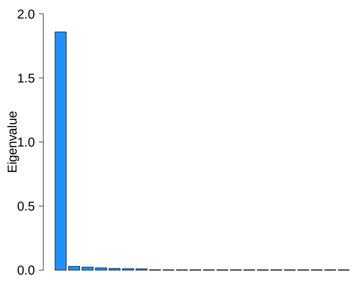


Figure 10: The eigenvalues of the learned matrix shown in Figure 7.

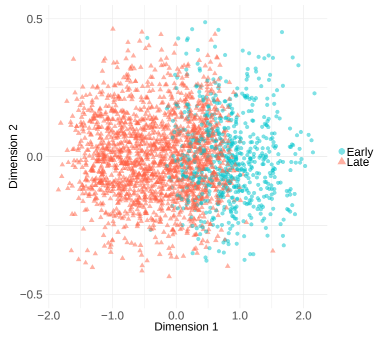


Figure 11: The data in the first two dimensions of the transformed feature space.

4 CONCLUSION

In this paper we have presented a novel methodology in the field of simulation analytics. The basis of our methodology, *k*NN classification, provides a non-parametric framework in which to model the behavior of a system, whilst the addition of metric learning is shown to bring both interpretability and improved prediction performance. We have demonstrated the merits of our approach for its intended application to simulation, showing that the typical features of sample path data, such as interacting and irrelevant variables, are well-handled by metric learning. Although we only present results from a single metric learning method in this paper, the extensive research and accomplishment in this field gives optimism to the scope of metric learning for simulation. In short, we propose that a *k*NN approach combined with metric learning can have wide-reaching benefits, as simulation users begin to look beyond aggregate performance measures and seek a more fine-grained analysis from their simulation models.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the EPSRC funded EP/L015692/1 STOR-i Centre for Doctoral Training and the EPSRC NPIF fund EP/S515127/1. In addition, Nelson's work is partially supported by the NSF under the Grant Number DMS-1854562.

REFERENCES

Bellet, A., A. Habrard, and M. Sebban. 2014. "A Survey on Metric Learning for Feature Vectors and Structured Data". https://arxiv.org/pdf/1306.6709.pdf, accessed 1st July 2020.

Beyer, K., J. Goldstein, R. Ramakrishnan, and U. Shaft. 1999. "When Is 'Nearest Neighbor' Meaningful?". In *International Conference on Database Theory*, edited by C. Beeri and P. Buneman, 217–235. Berlin, Heidelberg: Springer.

Brailsford, S. C. 2007. "Tutorial: Advances and Challenges in Healthcare Simulation Modeling". In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 1436–1448. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Fu, A., B. Narasimhan, and S. Boyd. 2018. "CVXR: An R Package for Disciplined Convex Optimization". https://arxiv.org/pdf/1711.07582.pdf, accessed 1st July 2020.

Globerson, A., and S. T. Roweis. 2005. "Metric Learning by Collapsing Classes". In *NIPS'05: Proceedings of the 18th International Conference on Neural Information Processing Systems*, edited by B. Schölkopf, J. C. Platt, and Y. Weiss, 451–458. Cambridge, Massachusetts: MIT Press.

Goldberger, J., G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. 2004. "Neighbourhood Components Analysis". In *NIPS'04: Proceedings of the 17th International Conference on Neural Information Processing Systems*, edited by L. K. Saul, Y. Weiss, and L. Bottou, 513–520. Cambridge, Massachusetts: MIT Press.

Hastie, T., R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer.

Jiang, G., L. J. Hong, and B. L. Nelson. 2020. “Online Risk Monitoring Using Offline Simulation”. *INFORMS Journal on Computing* 32(2):356–375.

Kayton, D., T. Teyner, C. Schwartz, and R. Uzsoy. 1996. “Effects of Dispatching and Down Time on the Performance of Wafer Fabs Operating Under Theory of Constraints”. In *Nineteenth IEEE/CPMT International Electronics Manufacturing Technology Symposium*, 49–56. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kedem, D., S. Tyree, F. Sha, G. R. Lanckriet, and K. Q. Weinberger. 2012. “Non-linear Metric Learning”. In *NIPS’12: Proceedings of the 25th International Conference on Neural Information Processing Systems*, edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, 2573–2581. Red Hook, New York: Curran Associates, Inc.

Kulis, B. 2013. “Metric Learning: A Survey”. *Foundations and Trends in Machine Learning* 5(4):287–364.

Lin, Y., and B. L. Nelson. 2018. “Variance and Derivative Estimation of Virtual Performance”. *ACM Transactions on Modeling and Computer Simulation* 28(3):1–20.

Lin, Y., B. L. Nelson, and L. Pei. 2019. “Virtual Statistics in Simulation via k Nearest Neighbors”. *INFORMS Journal on Computing* 31(3):576–592.

Moss, H. B., D. S. Leslie, and P. Rayson. 2018. “Using J-K-Fold Cross Validation to Reduce Variance When Tuning NLP Models”. In *Proceedings of the 27th International Conference on Computational Linguistics*, edited by E. Bender, L. Derczynski, and P. Isabelle, 2978–2989. Stroudsburg, Pennsylvania: Association for Computational Linguistics.

Nelson, B. L. 2016. “‘Some Tactical Problems in Digital Simulation’ For the Next 10 Years”. *Journal of Simulation* 10(1):2–11.

Ouyang, H., and B. L. Nelson. 2017. “Simulation-Based Predictive Analytics for Dynamic Queueing Systems”. In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D’Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 1716–1727. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Torresani, L., and K.-c. Lee. 2006. “Large Margin Component Analysis”. In *NIPS’06: Proceedings of the 19th International Conference on Neural Information Processing Systems*, edited by B. Schölkopf, J. C. Platt, and T. Hoffman, 1385–1392. Cambridge, Massachusetts: MIT Press.

Wang, J., A. Woznica, and A. Kalousis. 2012. “Learning Neighborhoods for Metric Learning”. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, edited by P. A. Flach, T. D. Bie, and N. Cristianini, 223–236. Berlin, Heidelberg: Springer.

Weinberger, K. Q., and L. K. Saul. 2009. “Distance Metric Learning for Large Margin Nearest Neighbor Classification”. *Journal of Machine Learning Research* 10(9):207–244.

Weinberger, K. Q., and G. Tesauro. 2007. “Metric Learning for Kernel Regression”. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, edited by M. Meila and X. Shen, 612–619. San Juan, Puerto Rico: PMLR.

Wu, X., and R. R. Barton. 2016. “Fourier Trajectory Analysis for Identifying System Congestion”. In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, 401–412. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Xing, E. P., M. I. Jordan, S. J. Russell, and A. Y. Ng. 2002. “Distance Metric Learning with Application to Clustering with Side-Information”. In *NIPS’02: Proceedings of the 15th International Conference on Neural Information Processing Systems*, edited by S. Becker, S. Thrun, and K. Obermayer, 521–528. Cambridge, Massachusetts: MIT Press.

AUTHOR BIOGRAPHIES

GRAHAM LAIDLER is a Ph.D. student at the EPSRC Centre for Doctoral Training in Statistics and Operational Research in Partnership with Industry (STOR-i) based at Lancaster University. His research interest is in the application of machine learning to data generated by stochastic simulation. His email address is g.laidler1@lancaster.ac.uk.

LUCY E. MORGAN is a Lecturer in Simulation and Stochastic Modelling in the Department of Management Science at Lancaster University. Her research interests are input uncertainty in simulation models and arrival process modelling. Her e-mail address is l.e.morgan@lancaster.ac.uk.

BARRY L. NELSON is the Walter P. Murphy Professor in the Department of Industrial Engineering and Management Sciences at Northwestern University and a Distinguished Visiting Scholar in the Lancaster University Management School. He is a Fellow of INFORMS and IISE. His research centers on the design and analysis of computer simulation experiments on models of stochastic systems. His email address is nelsonb@northwestern.edu.

NICOS G. PAVLIDIS is a Senior Lecturer in the Department of Management Science at Lancaster University. His research is on developing novel clustering and classification methods that can handle high dimensional data and can adapt in response to time-varying population distributions. His email address is n.pavlidis@lancaster.ac.uk.