Stochastic Neighbourhood Components Analysis

Graham Laidler[1], Lucy E. Morgan[1], Barry L. Nelson[2], and Nicos G. Pavlidis[1]

[1]STOR-i Centre for Doctoral Training, Lancaster University, Lancaster, LA1 4YW, UK
[2]Department of Industrial Engineering & Management Sciences, Northwestern University, Evanston, IL 60208-3119

Abstract

Distance metric learning is a fundamental task in data mining, and is known to enhance the performance of various distance-based algorithms. In this paper, we consider stochastic training data in which repeated feature vectors can belong to different classes, a scenario in which existing methods of metric learning are known to struggle. This type of data is common in stochastic simulations, where multi-dimensional, recurrent system states are subject to inherent randomness. Classification models on such high-resolution simulation-generated data play a critical role in real-time decision-making across diverse applications. This paper presents and implements a stochastic version of the popular Neighbourhood Components Analysis. We demonstrate its behaviour on stochastic data using simulation models, and reveal its advantages when used for nearest neighbour classification. Meanwhile, the assumptions of stochastic labelling and repeated feature vectors extend to data from various domains, suggesting that the method can attain broad impact. For example, beyond its applications to system control and decision-making with digital twin simulation, it may enhance the analysis of data from sensor networks, recommender systems, and crowdsourced platforms, where stochasticity and recurring feature patterns are typical.

Keywords: distance metric learning, stochastic data, discrete-event simulation, simulation analytics, nearest neighbours

1 Introduction

Stochastic, discrete-event simulation is a valuable tool for the design of systems that evolve through time and must perform well in the face of uncertainty. By "design", we refer to one-time decisions, such as the layout of work centres in a factory, the number of beds allocated to post-surgery patients in a hospital, the securities to include in a portfolio, or the capacities of warehouses in a supply chain. Simulation is less well-suited for real-time control decisions due to long execution times, but is being pushed in that direction by the need for high-resolution, but timely, decisions in applications such as digital twins (dos Santos et al. 2022). A digital twin is a simulation that shadows a real-world system to support control or recovery decision as needed. This tension between simulation execution time and real-world decision time is the genesis of *offline simulation for online application*: extensive simulations are executed in advance and the results processed to be exploited for real-time decisions (Hong and Jiang 2019). This involves the careful mining of simulation state information; we refer to the time-dynamic trajectory of simulation state variables as the simulation *sample path*. Recent applications of sample path mining include financial risk assessment (Jiang et al. 2020), personalized medicine (Shen et al. 2021) and assortment optimization (Keslin et al. 2024). This work aims to more fully exploit the properties of simulation sample paths to enable higher-resolution predictions than previous methods.

We consider the task of classifying a future system state. Section 4.1 presents an analysis of a queueing system looking at the probability of the system becoming blocked, a commonly encountered issue across many industries. Section 4.2 describes a manufacturing context in which product completion times are classified as early or late at the start of a product cycle, based on simulation-generated completion times from the nearest neighbours of the current system state. At its highest level of detail, the system state is completely descriptive of the system at any moment in time, and so identifying nearest neighbours on system state provides a logical basis for predictions (Laidler et al. 2020). For this reason, we propose in this paper a method of distance metric learning (DML) tailored to the characteristics of sample path data.

Extensive research efforts in the field of DML have resulted in a wealth of notable formulations (Kulis 2013). However, existing algorithms are typically built with an implicit assumption of having unique feature vectors for training, an assumption which conflicts with the nature of sample path data. The typically discrete representation and recurrent nature of the system state in a discrete-event simulation model provides repeated feature vector observations, and calculating objective functions as a sum over individual points becomes inefficient. More significantly, inherent simulation randomness results in stochastic classifications. Repeated feature vectors belonging to different classes will often introduce conflicting constraints when existing DML methods are applied, and can lead to k-nearest neighbours (kNN) committing "*substantial errors*" (Suárez et al. 2021). In the current work, we address this shortcoming by proposing a method which we refer to as Stochastic Neighbourhood Components Analysis (SNCA). This is based on the Neighbourhood Components Analysis (NCA) of Goldberger et al. (2005), which builds a probabilistic model of class assignment. We modify NCA to the context of repeated data points and stochastic classifications.

Although providing a large family of problems, the simulation domain is not the beginning and the end of our application. Simulation sample paths identify a characteristic type of data which arises in numerous contexts. Crowdsourced datasets, for example, often result in multiple annotators providing various responses to the same labelling task (Vaughan 2017). Tasks such as sentiment analysis or media ratings are subjective in nature and certainly result in stochastic labelling. Beyond this, however, many real-life situations experience naturally stochastic behaviour. Medical prognoses (Suo et al. 2018), financial forecasts (Cao and Tay 2001), and sporting events (Horvat and Job 2020), for example, are often modelled with discrete features and encounter natural variability in their observed outcomes. In summary, repeated feature vectors and stochastic labelling characterise a large body of classification problems, and so we see a DML formulation tailored to this context as a useful contribution.

The remainder of the paper is organised as follows. Section 2 establishes a relevant background in simulation analytics and DML. In Section 3, we give a characterisation of the data and the proposed methodology for SNCA, including attention to the theoretical convergence of the optimal solution. We present experimental results to assess the performance of SNCA on sample path data in Section 4, before the paper concludes with a brief summary in Section 5.

2 Related Work

Numerous texts provide a comprehensive study on the subject of stochastic simulation (Nelson and Pei 2021, Law and Kelton 2007). In this section, we focus attention on the emerging topic of simulation analytics (Nelson 2016), which provides a motivation for the current work. We also introduce the task of DML, and present a focused review of the existing literature.

2.1 Simulation Analytics

Discrete-event simulation (DES) represents the operation of a real system as a sequence of events occurring at discrete points in time. Each event, such as a customer arrival or a machine failure, triggers a change in a set of variables describing the system state, including for instance a queue size or a machine status. The trajectory of state variables, or sample path, from a DES model hides a wealth of insight into the system behaviour. Nelson (2016) recognised modern capacity for the storage of sample path data as paving the way for machine learning solutions, and coined the term *simulation analytics*. Essentially, analysis of simulation has traditionally centred around static summaries of long-run performance, with average performance *over* time being prioritised above dynamic performance *through* time. However, simulation analytics aims to prioritise the latter, recognising that a far more complete picture of system behaviour can be unlocked by directing machine learning efforts to the simulation sample path. The current work adds to this endeavour.

We summarise some existing publications on the theme of simulation analytics. A number of papers have emerged in which sample path data are used to build metamodels for 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) also use a logistic regression model to dynamically predict the risk of financial portfolios. Morgan and Barton (2022), meanwhile, show that Fourier analysis can successfully detect changes in the trajectories of individual state variables to discriminate between congested and uncongested systems. Moving to distance-based methods, Lin et al. (2019) suggest a kNN approach to provide performance predictions based on simulation time. A first investigation of the potential for DML in simulation analytics was provided in Laidler et al. (2020), with the independent variable being extended to a multidimensional description of the system state. However, a tailored DML approach was not offered, and this is instead the main contribution of the current work.

As an accessible source of measurement-error-free data, sample paths provide a rich environment for machine learning. The attempts described above have applied off-the-shelf algorithms. However, to achieve the full potential of learning in this context, it is necessary to account for the distinctive properties of simulation sample path data. This presents an opportunity for machine learning research, and this work takes a step in that direction.

It is encouraging to note that beyond the sphere of research, applications and opportunities for machine learning in simulation are also filtering through to practitioners. Commercial providers of simulation software, including for example Simul8 (https://www.simul8.com), Simio (https://www.simio.com), and AnyLogic (https://www.anylogic.com), have recognised its benefits and provide easy integration with machine learning models, including with runtime execution. Crucially, the support is already in place for research advances in simulation analytics to quickly deliver impact to simulation users.

2.2 Distance Metric Learning

Distance calculations among data points comprise a fundamental aspect of many machine learning tasks. DML has been shown to improve the performance of nearest neighbour predictions (Weinberger and Saul 2009), clustering (Xing et al. 2002), and information retrieval (McFee and Lanckriet 2010), and it is therefore treated as an important objective in its own right (Kulis 2013). In a fully supervised setting, real-valued, multivariate feature vectors, $\boldsymbol{x}_i \in \mathbb{R}^d$, are supplied with a class label, y_i, and the goal is to adapt a pairwise distance function over the feature vectors, such that nearby points are more likely to belong to the same class. A common approach to DML is to learn a generalised Mahalanobis distance. Our work in this paper conforms to this framework, and this becomes the focus of our review.

Defining the squared distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as $d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top M(\boldsymbol{x}_i - \boldsymbol{x}_j)$, the Mahalanobis distance (Mahalanobis 1936) is defined by setting $M = \Sigma^{-1}$, where Σ is the covariance matrix of the data. In the metric learning literature, a broader family of Mahalanobis metrics is parameterised by restricting M to the set of $d \times d$ positive semidefinite matrices, which we will denote by $\mathbb{S}_+^d$. The task of learning a Mahalanobis metric is then generally expressed as an optimisation problem of the following form:

$$\min_{M \in \mathbb{S}_+^d} \ell(M, \mathcal{D}_n) + \lambda r(M).$$

Here, ℓ is a loss function relating the suitability of M to the supervision brought by the data, $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, and $r(M)$ describes some regularisation on the values of M, with the regularisation parameter, $\lambda \geq 0$. Many formulations have been proposed, with differences arising in their choice of loss functions and regularisation. For a comprehensive review of established methods, refer to surveys provided by Kulis (2013), Bellet et al. (2013), and, more recently, Li and Tian (2018). Here, we limit our discussion to the main directions of the research, and establish a relevant background for the current work.

A common approach converts supervision into the form of similarity judgments among tuples of training points. For instance, sets $\mathscr{S}$ and $\mathscr{D}$ are often used to contain pairs of training points deemed to be similar and dissimilar, respectively. Under this framework, the loss function, $\ell(M, \mathcal{D}_n)$, is used to encode violations of the desired relationships under the metric induced by M. Construction of a linear loss function leads to convex formulations and allows the problem to be treated as a semidefinite program. The earliest example of this is Xing et al. (2002), with the following intuitive formulation:

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

A succession of notable semidefinite programming formulations have since emerged, including the widely used Large Margin Nearest Neighbors (LMNN, Weinberger and Saul (2009)). Motivated by kNN, LMNN seeks local neighbourhoods of each instance to be populated with other instances of the same class, while those of different class are repelled by a '*large*' margin. LMNN has been the subject of a number of extensions. Torresani and Lee (2007) combine LMNN with dimensionality reduction of the feature space, and explore non-linear kernel methods, while Kedem et al. (2012) suggest alternative extensions to a non-linear metric. Further notable works, also making use of supervision sets, $\mathscr{S}$ and $\mathscr{D}$, and offering an information-theoretic perspective, include Bar-Hillel et al. (2003) and Davis et al. (2007).

The approach of populating supervision sets with pairwise relationships is less natural in our context of repeated feature vectors and stochastic classifications. In particular, we might expect the same pair to appear a number of times in *both* $\mathscr{S}$ and $\mathscr{D}$, leading to either inefficient or infeasible formulations. Instead, the stochastic nature of the class labels leads us to prefer a probabilistic formulation, and we hence turn our attention to existing probabilistic methods.

An influential formulation known as Neighbourhood Components Analysis (NCA, Goldberger et al. (2005)) proceeds to model the nearest neighbour probabilities with a softmax normalisation over distances. A point, $\boldsymbol{x}_i$, identifies its nearest neighbour as $\boldsymbol{x}_j$ with probability p_{ij}:

$$p_{ij} = \frac{\exp\{-\|A\boldsymbol{x}_i - A\boldsymbol{x}_j\|_2^2\}}{\sum_{k\neq i}\exp\{-\|A\boldsymbol{x}_i - A\boldsymbol{x}_k\|_2^2\}}, \quad p_{ii} = 0. \tag{1}$$

This imposed distribution provides a continuous relaxation to the deterministic point mass distribution on the nearest neighbour. With $\boldsymbol{x}_i$ inheriting its classification from a neighbour selected from this distribution, the quantity, $p_i = \sum_{j:\, y_j = y_i} p_{ij}$, represents the probability of $\boldsymbol{x}_i$ having the class assignment of y_i. The suggested objectives are to maximise the expected leave-one-out accuracy of the nearest neighbour classifier under this distribution, or the log-likelihood function:

$$\max_A \sum_{i=1}^n p_i, \quad \text{or} \quad \max_A \sum_{i=1}^n \log(p_i).$$

The optimisation of NCA is performed over an unconstrained matrix, A, which reveals a Mahalanobis matrix, M, via the relation $A^\top A = M$. Any $M \in \mathbb{S}_+^d$ permits this decomposition, such that a Mahalanobis metric can equivalently be viewed as a projection of Euclidean space under a linear transformation, $\boldsymbol{x} \to A\boldsymbol{x}$. With this perspective, optimising the transformation matrix, A, provides an alternative perspective to the task of Mahalanobis metric learning. Dimensionality reduction can be directly sought by restricting the number of rows in A, whilst costly projections onto $\mathbb{S}_+^d$ are avoided. However, an optimisation of A often comes at the expense of convexity: a loss function which is convex in M is typically non-convex in A. This is no concern in the case of NCA, since the objective function is non-convex in either parameterisation.

Underneath NCA lies a predictive model of class distributions: p_i can be seen as a kernel density estimator for the conditional probability of y_i given $\boldsymbol{x}_i$ (Devroye and Wagner 1980). The form of the softmax transformation (1) disguises a Gaussian kernel function, with the common bandwidth parameter being absorbed into the scale of A. Weinberger and Tesauro (2007) more explicitly present a metric learning method in this context, albeit for a continuous target variable. In particular, kernel regression estimates for the target variable are constructed in the same form as above, and a metric is sought to minimise the mean squared error.

A similar probabilistic model was proposed by Peltonen and Kaski (2005), in which an expression of the data log-likelihood is maximised by evaluating conditional class probability estimators of a similar form as those of NCA. Indeed, the estimates introduced by NCA establish a popular framework which has been adopted by a number of other authors, and also provides inspiration for our own approach. Globerson and Roweis (2005) obtain a convex objective function, while Tarlow et al. (2013) explicitly extend the NCA objective to reflect the expected kNN accuracy for any choice of k. Further related extensions of NCA include Wang and Tan (2014), who consider the context of noisy labels arising from measurement error. Our own context involves true noise arising from a probabilistic labelling mechanism, and in this direction, Yang (2020) adapts NCA to the context of probabilistic labels, in which a full probabilistic class distribution is assumed to be known for each instance. Our own approach assumes no such knowledge.

The NCA framework provides a useful foundation for probabilistic metric learning, although it is not designed to apply when input observations are characterised by *repeating* features and *stochastic* outcomes. In this context, the objective function is heavily influenced by points' identical *self-neighbours* which impede learning by remaining unchanged under any metric, while the pointwise formulation amounts to inefficient optimisation. We proceed in the next section to introduce a multinomial framework for our data setting, and propose an NCA-inspired metric learning formulation to serve it.

3 Methodology

We begin by describing a data framework in which repeated feature vectors and stochastic labelling characterise a set of classification problems. To establish a mechanism generating observation pairs, we assume a distribution, $q_{\boldsymbol{X},Y}$, over a pair of discrete random variables, $\boldsymbol{X}$ and Y. The input variable, $\boldsymbol{X}$, takes values from a set, $\mathcal{X} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_m\}$, containing a finite number of vectors in $\mathbb{R}^d$, which we refer to as states, whilst Y represents a class label drawn from a set, $\mathcal{Y}$, containing at least two unordered values. Our task is to find a pairwise distance function acting on the states in $\mathcal{X}$, which in some way reflects similarity of the conditional class distributions of $Y \mid \boldsymbol{X}$. We receive supervision from a finite training sample, $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, drawn independently from $q_{\boldsymbol{X},Y}$.

In many applications, the training sample will not represent the entirety of $\mathcal{X}$. Whilst we expect repeats of *some* states, many others may be observed only once or not at all. Therefore, in classifying a new input, $\boldsymbol{x}^\star$, we are not expecting to rely solely on observed training repetitions of $\boldsymbol{x}^\star$ and an empirical class distribution. Instead, we require a distance metric to point us to neighbouring states which, through a similarity of class distributions, will make good predictors. As the basis for this, we seek a model which can learn from the more frequently attended states to extend a conditional structure for $Y \mid \boldsymbol{X}$ across the whole set of $\mathcal{X}$.

The discrete nature of $\mathcal{X} \times \mathcal{Y}$ allows us to view $\mathcal{D}_n$ as a multinomial sample. We introduce a random variable, $C_l^y(n) = |\{i \in \{1, 2, \dots, n\} \colon \boldsymbol{X}_i = \boldsymbol{b}_l, Y_i = y\}|$, to represent the observed frequency of the pair $(\boldsymbol{b}_l, y)$ in a data sample of size n. The random vector, $\{C_l^y(n)\}_{l=1,\dots,m}^{y\in\mathcal{Y}}$, follows a multinomial distribution with parameters n and $\{q(\boldsymbol{b}_l, y)\}_{l=1,\dots,m}^{y\in\mathcal{Y}}$, and we can represent $\mathcal{D}_n = \{c_l^y\}_{l=1,\dots,m}^{y\in\mathcal{Y}}$ as a realisation of this random vector. Conditional on $\mathcal{D}_n$, the maximum likelihood estimates of the multinomial parameters are provided by $\hat{q}(\boldsymbol{b}_l, y) = c_l^y/n$. Using c_l to represent $\sum_{y\in\mathcal{Y}} c_l^y$, we also introduce marginal and conditional data distributions with $\hat{q}(\boldsymbol{b}_l) = c_l/n$ and $\hat{q}(y \mid \boldsymbol{b}_l) = c_l^y/c_l$ provided $c_l > 0$, respectively. Although these distributions depend on $\mathcal{D}_n$, we omit this to simplify the notation.

The remainder of this section is organised as follows. Section 3.1 introduces a probabilistic framework and objective function for SNCA. This establishes a model for the conditional distribution of $Y \mid \boldsymbol{X}$. The relationship of SNCA to NCA is described in Section 3.2, before Section 3.3 presents a convergence result relating to the asymptotic behaviour of the optimal SNCA solution. Section 3.4 contains details of the optimisation algorithm applied in this paper.

3.1 A Probabilistic Framework

In the style of NCA, we impose a probability distribution for neighbour assignment based on a softmax function over distances. In contrast to NCA, however, this is applied over multinomial cells as opposed to over individual $\boldsymbol{x}_i$. Under a transformation matrix, A, a cell with $\boldsymbol{X} = \boldsymbol{b}_l$ identifies the cell $(\boldsymbol{b}_h, y)$ as its nearest neighbour with probability $p^A((\boldsymbol{b}_h, y) \mid \boldsymbol{b}_l)$:

$$p^A((\boldsymbol{b}_h, y) \mid \boldsymbol{b}_l) = \frac{c_h^y \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}{\sum_{k\neq l} c_k \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_k\|_2^2\}}\ , \quad p^A((\boldsymbol{b}_l, y) \mid \boldsymbol{b}_l) = 0.$$

Again, we omit the conditioning on $\mathcal{D}_n$ from the notation. Assuming classifications to be inherited from a cell selected by this distribution, the probability of classifying $\boldsymbol{X} = \boldsymbol{b}_l$ as class y is given by

$$p^A(y \mid \boldsymbol{b}_l) = \sum_{h=1}^m p^A((\boldsymbol{b}_h, y) \mid \boldsymbol{b}_l).$$

Under any matrix, A, this specifies a complete conditional distribution, $p^A_{Y|\boldsymbol{X}}$, which is constructed as a weighted average over observed class distributions, $\hat{q}_{Y|\boldsymbol{X}}$. That is, we can write

$$p^A(y \mid \boldsymbol{b}_l) = \sum_{h=1}^m p_{lh}\hat{q}(y \mid \boldsymbol{b}_h),$$

where the weight, $p_{lh} = \sum_{y\in\mathcal{Y}} p^A((\boldsymbol{b}_h, y) \mid \boldsymbol{b}_l)$, given to the observed class distribution of the neighbour $\boldsymbol{b}_h$ decreases exponentially with the distance under A of $\boldsymbol{b}_h$ from $\boldsymbol{b}_l$. With this perspective, we treat our task as an optimisation problem to find a transformation, A, to align the model distribution, $p^A_{Y|\boldsymbol{X}}$, with the data distribution, $\hat{q}_{Y|\boldsymbol{X}}$.

Obtaining the modelled joint distribution as $p^A(\boldsymbol{b}_l, y) = \hat{q}(\boldsymbol{b}_l)p^A(y \mid \boldsymbol{b}_l)$, the multinomial likelihood, $L(A; \mathcal{D}_n)$, and log-likelihood, $\log L(A; \mathcal{D}_n)$, of $\mathcal{D}_n$ can be expressed as the following functions of A:

$$L(A; \mathcal{D}_n) = \frac{n!}{\prod_l \prod_y c_l^y!} \prod_{l=1}^m \prod_{y\in\mathcal{Y}} p^A(\boldsymbol{b}_l, y)^{c_l^y},$$

$$\log L(A; \mathcal{D}_n) = \log(n!) - \sum_{l=1}^m \sum_{y\in\mathcal{Y}} \log(c_l^y!) + \sum_{l=1}^m \sum_{y\in\mathcal{Y}} c_l^y \log(p^A(\boldsymbol{b}_l, y)).$$

Recalling that $\hat{q}(\boldsymbol{b}_l, y) = c_l^y/n$, we see that a maximisation of the log likelihood corresponds to the maximisation of a function, $g_n(A)$, and observe a connection to a Kullback-Leibler (KL) divergence:

$$\begin{aligned}
\max_A \log L(A; \mathcal{D}_n) &= \max_A g_n(A), \\
\text{where } g_n(A) &= \sum_{l=1}^{m} \sum_{y \in \mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \log(p^A(\boldsymbol{b}_l, y)) \\
&= \sum_{l=1}^{m} \sum_{y \in \mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \log\left(\frac{p^A(\boldsymbol{b}_l, y)}{\hat{q}(\boldsymbol{b}_l, y)}\right) + \sum_{l=1}^{m} \sum_{y \in \mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \log(\hat{q}(\boldsymbol{b}_l, y)) \\
&= -D_{\mathrm{KL}}(\hat{q}_{\boldsymbol{X},Y} \| p^A_{\boldsymbol{X},Y}) - H(\hat{q}_{\boldsymbol{X},Y}).
\end{aligned}$$

Here, H represents the Shannon entropy (Shannon 1948). Since $H(\hat{q}_{\boldsymbol{X},Y})$ is independent of A, the maximisation of $g_n(A)$ corresponds to a minimisation of the KL divergence of $p^A_{\boldsymbol{X},Y}$ from $\hat{q}_{\boldsymbol{X},Y}$. We receive a similar interpretation in relation to the conditional distributions:

$$\begin{aligned}
g_n(A) &= \sum_{l=1}^{m} \sum_{y \in \mathcal{Y}} \hat{q}(\boldsymbol{b}_l)\hat{q}(y \mid \boldsymbol{b}_l) \log(\hat{q}(\boldsymbol{b}_l) p^A(y \mid \boldsymbol{b}_l)) \\
&= \sum_{l=1}^{m} \hat{q}(\boldsymbol{b}_l) \sum_{y \in \mathcal{Y}} \hat{q}(y \mid \boldsymbol{b}_l) \log p^A(y \mid \boldsymbol{b}_l) + \sum_{l=1}^{m} \hat{q}(\boldsymbol{b}_l) \log \hat{q}(\boldsymbol{b}_l) \\
&= -\mathbb{E}_{\hat{q}_{\boldsymbol{X}}}[D_{\mathrm{KL}}(\hat{q}_{Y|\boldsymbol{X}} \| p^A_{Y|\boldsymbol{X}})] - \mathbb{E}_{\hat{q}_{\boldsymbol{X}}}[H(\hat{q}_{Y|\boldsymbol{X}})] - H(\hat{q}_{\boldsymbol{X}}).
\end{aligned}$$

Therefore, maximisation of $g_n(A)$ minimises the expected KL divergence of the class conditional distributions of the model from the data, under $\hat{q}_{\boldsymbol{X}}$. This perspective perhaps most convincingly aligns with our intuition for a desirable metric, whilst making use of a common probability-based divergence measure.

3.2 Relationship to NCA

The formulations of NCA and SNCA exactly coincide when applied to training data without repeated observations, i.e., training data for which $\sum_{y \in \mathcal{Y}} c_l^y \leq 1$ holds for all $l = 1, 2, \ldots, m$. When this condition does not hold, a difference arises. Expressing both formulations as

$$\max_A \sum_l \sum_y \hat{q}(\boldsymbol{b}_l, y) \log p^A(y \mid \boldsymbol{b}_l),$$

the difference occurs in the model estimates of the conditional probabilities, $p^A(y \mid \boldsymbol{b}_l)$. Whenever $\hat{q}(\boldsymbol{b}_l, y) > 0$, these estimates are as follows:

$$\text{NCA: } p^A_{(N)}(y \mid \boldsymbol{b}_l) = \frac{c_l^y - 1 + \sum_{h \neq l} c_h^y \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}{c_l - 1 + \sum_{h \neq l} c_h \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}, \tag{2}$$

$$\text{SNCA: } p^A_{(S)}(y \mid \boldsymbol{b}_l) = \frac{\sum_{h \neq l} c_h^y \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}{\sum_{h \neq l} c_h \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}. \tag{3}$$

The difference between (2) and (3) stems from the inclusion and exclusion, respectively, of a point's repeats in the softmax construction of its neighbour distribution. Since identical points will always have a distance of zero, the terms $c_l^y - 1$ and $c_l - 1$ appear regardless of A and effectively act as bias to $p^A_{(N)}(y \mid \boldsymbol{b}_l)$. As a result, the influence of A on the objective function of NCA is reduced. The adjustment made by SNCA is to exclude identical neighbours from consideration in the softmax functions, which allows the SNCA objective to solely reflect the quality of distance relationships among non-identical points.

We also recognise inconsistencies arising in the NCA estimates. Specifically, we may not have $\sum_{y \in \mathcal{Y}} p^A_{(N)}(y \mid \boldsymbol{b}_l) = 1$. As n increases, the discrepancy reduces, and we can in fact identify an asymptotically optimal solution. In particular, a solution, A, with infinite column norms, such that non-identical points are infinitely far apart, will yield $p^A_{(N)}(y \mid \boldsymbol{b}_l) = (c_l^y - 1)/(c_l - 1) \to \hat{q}(y \mid \boldsymbol{b}_l)$ as $n \to \infty$. This identifies a non-informative solution to which NCA becomes susceptible, under its log-likelihood objective function, as the data size increases. This essentially represents the situation of fully overfitting to the data, a hazard which SNCA avoids by introducing implicit leave-one-out validation during learning.

3.3 Convergence of Optimal Solutions

In Section 3.1, we understood $g_n(A)$ in relation to the observed distribution, $\hat{q}_{\boldsymbol{X},Y}$, of a finite sample. In this section, we seek to establish convergence guarantees with respect to the underlying true distribution, $q_{\boldsymbol{X},Y}$.

The strong law of large numbers (Kolmogorov and Širjaev 1992) ensures consistency of $\hat{q}_{\boldsymbol{X},Y} \overset{a.s.}{\to} q_{\boldsymbol{X},Y}$ as $n \to \infty$, and by the continuous mapping theorem we obtain the pointwise limits of $g_n(A)$ over fixed $A \in \mathbb{R}^{d\times d}$ as $n \to \infty$:

$$g_n(A) \overset{a.s.}{\to} \sum_{l=1}^{m} \sum_{y\in\mathcal{Y}} q(\boldsymbol{b}_l, y) \log \left(q(\boldsymbol{b}_l) \frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y) \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k) \exp\{-\|A\boldsymbol{b}_l - A\boldsymbol{b}_k\|_2^2\}} \right).$$

We use $g(A)$ to denote the limit of $g_n(A)$ given above, and consider the (set of) maximisers of $g(A)$, denoted $A_g^\star$, to be the true system-optimal solutions. To address the asymptotic behaviour of a finite sample solution, $\hat{A}_g$, we refer to a result relating to the consistency of Sample Average Approximation (SAA) estimators (Theorem 5.3, Shapiro et al. (2021)). Adopting the notation introduced here, the result is as follows:

Theorem 1 *Suppose there exists a compact subset $C \subset \mathbb{R}^{d\times d}$ such that:*

(i) $A_g^\star$ is nonempty and contained in C,

(ii) for sufficiently large n, with probability 1, $\hat{A}_g$ is nonempty and contained in C,

(iii) $g(A)$ is finite-valued and continuous on C, and

(iv) $g_n(A) \overset{a.s.}{\to} g(A)$ as $n \to \infty$, uniformly on C.

Then $g_n(\hat{A}_g) \overset{a.s.}{\to} g(A_g^\star)$ and $\mathbb{D}(\hat{A}_g, A_g^\star) \overset{a.s.}{\to} 0$ as $n \to \infty$.

Here, $\mathbb{D}(B, C)$ describes the deviation of the set B from the set C, defined as $\mathbb{D}(B, C) = \sup_{x\in B}\{\inf_{x'\in C}\|x - x'\|\}$. Central to this theorem is the assumption of a compact subset C in which $A_g^\star$ is contained. This excludes the possibility of infinite-valued solutions, which, although theoretically possible, we do not encounter in practice. For simplicity, we solve an optimisation problem in which the search space is not bounded, and assume that *(i)* and *(ii)* can be satisfied.

To address condition *(iii)*, we consider the argument of the logarithms to ensure that $g(A)$ is finite-valued. These arguments represent the limits of $p^A(\boldsymbol{b}_l, y)$ as $n \to \infty$ for all $\boldsymbol{b}_l$ and y. Whilst these naturally reside in $[0, 1]$, we require them in the slightly stricter domain of $[a, 1]$, where $a > 0$. That is, we seek to exclude the possibility of $p^A(\boldsymbol{b}_l, y)$ converging to zero when $q(\boldsymbol{b}_l, y) > 0$. This can be ensured with the following mild condition on $q_{\boldsymbol{X},Y}$:

$$\sum_{h\neq l} q(\boldsymbol{b}_h, y) > 0 \quad \forall \boldsymbol{b}_l \in \mathcal{X}, y \in \mathcal{Y} \text{ with } q(\boldsymbol{b}_l, y) > 0. \tag{4}$$

In other words, each class, $y \in \mathcal{Y}$, must be observable in conjunction with at least two states in $\mathcal{X}$. Alongside the assumption that $A \subset C$, condition (4) ensures that $g(A)$ is finite-valued on C, and since also continuous, condition *(iii)* is satisfied. A justification of the uniform convergence required by condition *(iv)* can be found in Appendix A.1.

Whilst we understand $\hat{A}_g$ as minimising an expected KL divergence of the class distributions of the model from the data, we consider $A_g^\star$ as relating to this expectation under the true data-generating distribution. Theorem 1 provides a helpful assurance that as the training sample increases and the empirical distribution, $\hat{q}_{\boldsymbol{X},Y}$, approaches its truth, our metric solutions should also approach their truth.

3.4 Optimisation

To implement SNCA, the differentiability of $g_n(A)$ allows us to apply a gradient-based optimiser. Recalling that $p_{lh} = \sum_{y\in\mathcal{Y}} p^A((\boldsymbol{b}_h, y) \mid \boldsymbol{b}_l)$, and denoting $\boldsymbol{b}_{lh} = \boldsymbol{b}_l - \boldsymbol{b}_h$, we have the following gradient:

$$\frac{\partial g_n(A)}{\partial A} = 2A \sum_{l=1}^{m} \sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y) \sum_{h\neq l} \boldsymbol{b}_{lh} \boldsymbol{b}_{lh}^\top p_{lh} \left(1 - \frac{\hat{q}(y \mid \boldsymbol{b}_h)}{p^A(y \mid \boldsymbol{b}_l)} \right).$$

Details for deriving this expression can be found in Appendix A.2.

In the metric learning literature, imposing some regularisation on the parameters of the metric is a common and accepted practice (Kulis 2013). In this work, we choose to impose orthogonality among the rows of A. Encouraging orthogonality has recently been recognised as an effective regularisation for metric learning (Dutta et al. 2020), and besides leading to a smaller set of projection vectors which bear less redundancy, benefits in mitigating the effects of class imbalance and avoiding overfitting have also been recognised (Xie et al. 2018). We choose not to require unit length vectors, since a flexible scaling allows the predictive relevance of the various orthogonal directions to be reflected. Our optimisation problem is thus expressed as follows:

$$\begin{aligned} &\max_{A \in \mathbb{R}^{r \times d}} g_n(A) \\ &\text{s.t. } AA^\top \text{ diagonal} \\ &r \leq d \end{aligned}$$

Optimisation over the full space of feasible matrices described in this formulation can be computationally challenging, and scale poorly with the problem dimension, d. Therefore, we propose a tailored optimisation scheme in two stages. In the first stage, an orthogonal row-wise optimisation approach is used to identify a good solution with rank $r \leq d$, which is used in the second stage as the initial solution to a constrained optimisation problem over the full matrix. This establishes a computationally efficient procedure leading to a good locally optimal solution. This approach is summarised in Algorithm 1.

In the first stage of the optimisation procedure (Algorithm 1, lines 1–18), the row-wise construction of an initial solution is made by optimising each row in the null space of the previously determined rows. Performing this null space projection is a computationally inexpensive step, and leads to orthogonal rows without the need for optimisation constraints. This row-by-row approach

Algorithm 1 SNCA

```
    Input: $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
    Stage 1 – Row-wise construction of initial solution
 1: $j \leftarrow 1$
 2: $A_0 \leftarrow []$                                   ▷ initialise empty solution matrix
 3: $P \leftarrow I_d$                                   ▷ initialise null space projection matrix
 4: old $\leftarrow -\infty$                              ▷ initialise old objective value
 5: new $\leftarrow g_n(P)$                               ▷ initialise new objective value
 6: obj_values $\leftarrow ()$                            ▷ initialise empty objective value vector
 7: while new > old & $j \leq d$ do
 8:     $\boldsymbol{a}_j^\star \leftarrow \arg\max_{\boldsymbol{a} \in \mathbb{R}^{1 \times d}} g_n\left(\begin{bmatrix} A_0 \\ \boldsymbol{a}P \end{bmatrix}\right)$     ▷ optimise row $j$ in null space of previous rows
 9:     old $\leftarrow$ new                              ▷ update old objective value
10:     new $\leftarrow g_n\left(\begin{bmatrix} A_0 \\ \boldsymbol{a}_j^\star \end{bmatrix}\right)$     ▷ update new objective value
11:     obj_values $\leftarrow$ (obj_values, new)         ▷ track objective value improvements
12:     if new > old then                                 ▷ check if row $j$ improves objective value
13:         $A_0 \leftarrow \begin{bmatrix} A_0 \\ \boldsymbol{a}_j^\star \end{bmatrix} \in \mathbb{R}^{j \times d}$     ▷ add row $j$ to solution matrix
14:         $\hat{\boldsymbol{a}}_j^\star \leftarrow \boldsymbol{a}_j^\star / \|\boldsymbol{a}_j^\star\|_2$
15:         $P \leftarrow (I_d - \hat{\boldsymbol{a}}_j^{\star\top} \hat{\boldsymbol{a}}_j^\star)P$     ▷ update projection matrix with null space of row $j$
16:         $j \leftarrow j + 1$                          ▷ move to next row
17:     end if
18: end while                                            ▷ stop when row $j$ does not improve objective value
    Stage 2 – Orthogonal optimisation
19: $r \leftarrow j - 1$     ▷ default rank is the number of rows in solution $A_0$. Alternatively, select
    $r < j - 1$ based on the objective value improvements in obj_values. Then $A_0 \leftarrow \begin{bmatrix} \boldsymbol{a}_1^\star \\ \vdots \\ \boldsymbol{a}_r^\star \end{bmatrix} \in \mathbb{R}^{r \times d}$.
20: $A^\star \leftarrow \arg\max_{A \in \mathbb{R}^{r \times d}} g_n(A)$ s.t. $AA^\top$ diagonal     ▷ optimisation with initial solution $A_0$
    Return: $A^\star$
```

ensures that the most informative directions are learned first, and we thus observe diminishing row norms as incrementally less improvement is found with each added dimension. Correspondingly, the diminishing improvements to the objective value allow a natural stopping rule when further improvement cannot be made, giving an appropriate $r \leq d$ and bringing explicit dimensionality reduction to the solution. In high dimensional applications, the diminishing marginal gains to the objective value may not justify the computational cost of further row-wise optimisation, and a stricter stopping rule, requiring objective improvements to exceed a percentage threshold, may be applied at the user's discretion.

The optimised orthogonal directions from the first stage generate a good solution, albeit one which may not be locally optimal. Therefore, this solution is used to initialise a constrained optimisation task in the second stage (Algorithm 1, lines 19–20), leading to a final solution with local optimality guarantee. Despite its good initial solution, however, the complexity of this second-stage optimisation can be significant, depending on parameters such as the problem dimension, d, and the matrix rank, r. For applications with high dimension, obtaining a lower-rank solution may be more advantageous than the diminishing improvements from the first stage, given the added complexity which an increased rank passes to the second stage, as well as to subsequent tasks such as nearest neighbour searches. Thus, choosing a lower rank, r, can be beneficial, and the objective values from the first stage optimisation may be used to produce a diminishing returns curve to guide this decision. In the experiments presented in Section 4, the problem dimension is low enough that the time complexity of the second-stage optimisation is not prohibitive, and manual rank-reduction is not required. In general, the orthogonal row-wise construction in the first stage naturally yields a low-rank solution, making further reduction of the rank unnecessary unless time constraints are a concern.

While the second-stage optimisation offers a theoretical guarantee, we have observed in practice that it tends to provide only minor refinement to the first-stage solution (for example, see Figure 2 and Figure 8 (top right)). Although this may not always be the case, its theoretical benefits might therefore be weighed against time constraints in practical application.

Since we are optimising a non-convex objective function, falling into 'bad' local optima remains a concern in the first stage of row-wise optimisation. Therefore, in the experiments presented in Section 4, we make multiple restarts for each row's optimisation. We choose a Latin Hypercube space-filling design using software provided by Urquhart et al. (2020) to generate a set of initialisations, and use the L-BFGS algorithm for the optimisation with a convergence limit of $\epsilon = 10^{-6}$ on the gradient. The orthogonality constraint in the second-stage optimisation is imposed using Riemannian optimisation. All implementation code is made available via the accompanying CodeOcean capsule.

4 Experiments on Simulated Data

In this section, we explore the performance of SNCA using data generated by stochastic simulation. We introduce a simple queueing model with a small and comprehensible state space, before presenting a more realistic application in the simulation of a semiconductor fabrication plant.

4.1 A Tandem Queueing System

Consider the simple queueing system represented in Figure 1. Two stations are arranged in tandem, with Station $I \in \{A, B\}$ containing s_I identical servers and c_I total capacity. Customers of type $k \in \{1, 2\}$ arrive to Station A according to a Poisson process with rate λ_k, and experience service times at Station I which are exponentially distributed with rate μ_{Ik}. The parameter q is used to denote the probability of rework.

This two-station, two-type set-up suggests a four-dimensional system state, $\boldsymbol{X} = (X_{A1}, X_{A2}, X_{B1}, X_{B2})$, where X_{Ik} represents the number of type k customers at Station I. The memoryless property of the exponential distribution makes elapsed service times irrelevant, while setting $s_I = c_I$ for $I \in \{A, B\}$

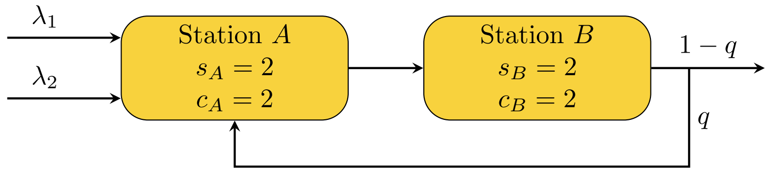

Figure 1. A tandem queueing system with feedback.

removes queueing space from both stations and ensures that $\boldsymbol{X}$ fully characterises the observable system state. We set $s_I = c_I = 2$ for $I \in \{A, B\}$, which allows $\boldsymbol{X}$ to take 36 distinct states.

We refer to Station $I \in \{A, B\}$ as being blocked whenever $X_{I1} + X_{I2} = c_I$, since no customers can enter the station in this state. In the logic of the system, customers completing their service in Station A will occupy space there while Station B is blocked. More significantly, customers arriving to a blocked Station A (including those returning from Station B) are immediately lost from the system. As such, blocking in Station A represents a critical state for the system, and our interest in this example is in predicting the future probability of this event based on the currently observed state. In particular, we observe the system state, $\boldsymbol{X}(t)$, at time t, and predict whether Station A will be blocked at time $t + T$. Let

$$Y(t) = \begin{cases} 1 & \text{if } X_{A1}(t+T) + X_{A2}(t+T) = c_A, \\ 0 & \text{otherwise.} \end{cases}$$

Simulating the system provides the data, $\mathcal{D}_n = \{(\boldsymbol{X}(t_i), Y(t_i))\}_{i=1}^n$, and we choose the recording times such that $t_{i+1} = t_i + T$ in order to obtain non-overlapping observations. To provide a roughly even distribution for Y, we simulated the system with the parameters $\lambda_1 = \lambda_2 = 2, \mu_{A1} = 2, \mu_{A2} = 3, \mu_{B1} = 1, \mu_{B2} = 2$, and $q = 0.2$, and we set the prediction horizon to $T = 0.25$. Choosing a smaller value of T would move towards a more deterministic problem, while a longer horizon would reduce the influence of $\boldsymbol{X}(t)$ and converge each state's conditional probability of $Y = 1$ towards the system's long-run blocking probability for Station A. With the specified system parameters, setting $T = 0.25$ provides a suitable middle ground in which the set of system states exhibit a wide range of class probabilities.

4.1.1 Metric Embeddings.

On a dataset of size $n = 100{,}000$, we obtained the following SNCA solution:

$$A = \begin{bmatrix} 7.53 & 6.92 & 3.17 & 1.85 \\ 0.31 & -0.71 & 0.25 & 1.01 \\ -0.15 & 0.03 & 0.31 & -0.01 \end{bmatrix}.$$

The transformation matrix consists of orthogonal rows representing data directions which are informative with respect to the class distributions. Following the optimisation scheme detailed in Section 3.4, rows are appended to the (first-stage) solution sequentially, and as such their relevance decreases and their magnitudes shrink, pointing us to an appropriate lower-dimensional solution. The trajectory of the objective function value throughout the optimisation process is shown in Figure 2. In this example, a dominant first dimension provides most of the projection variation, with a second and third dimension making smaller contributions. A fourth dimension was not found to provide any further improvements to the objective value.

We consider the first row of the solution as the most informative, and we can derive interpretation from the magnitudes in this row acting on the original data dimensions. X_{A1} is deemed slightly more relevant than X_{A2}, which we can understand since the system was simulated with $\mu_{A1} < \mu_{A2}$. This means that customers represented by X_{A1} tend to have longer service times and so are more likely to still occupy Station A when Y is observed. Customers in Station B can indirectly increase the risk of Station A becoming blocked, and in the same way, $\mu_{B1} < \mu_{B2}$ leads to a first dimension to which X_{B1} makes a greater contribution than X_{B2}.

To visualise the effect of SNCA, we plot the 36 unique states projected into the solution space, showing the first two dimensions which capture the majority of the variation (Figure 3, top). Recall

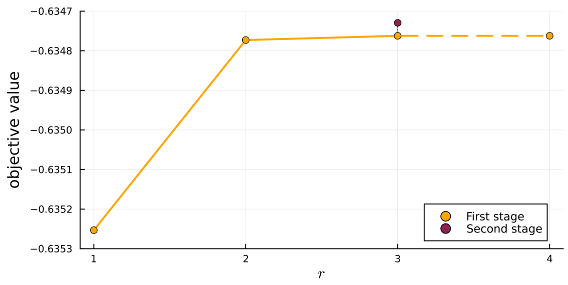

Figure 2. The objective value improvement throughout the SNCA algorithm.

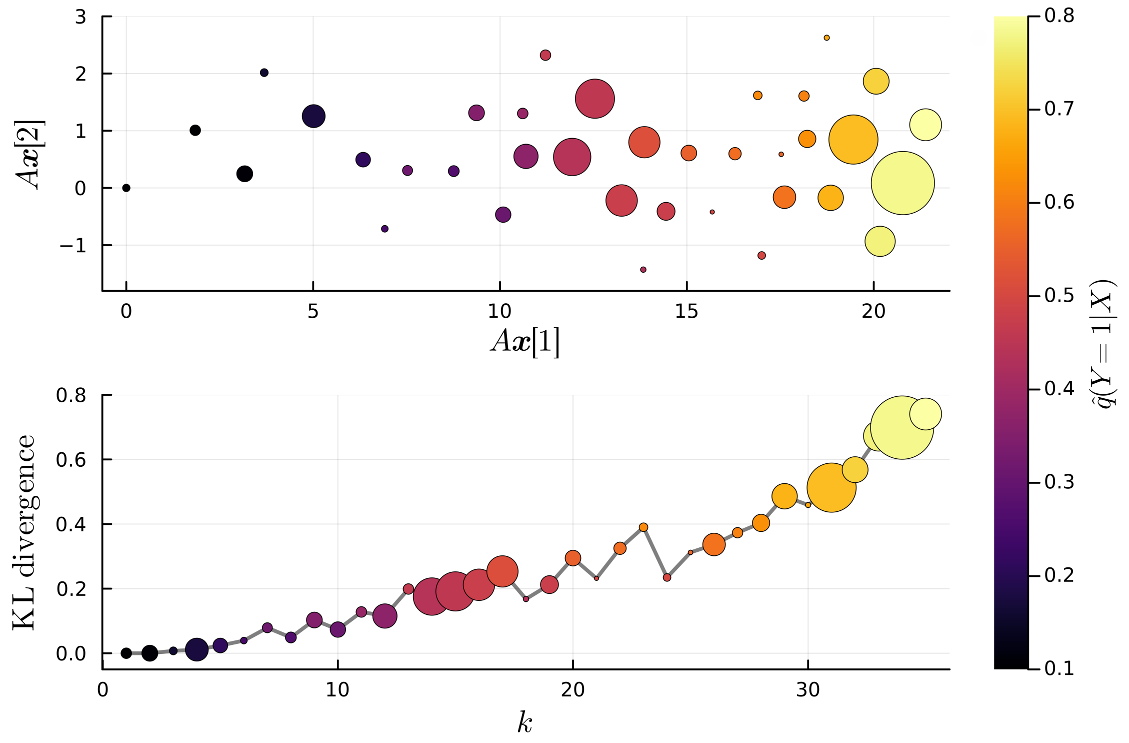

Figure 3. *Top*: the 36 unique states projected in the first two dimensions of the SNCA solution matrix. The colour of the points relates to their empirical class distribution, and the size is proportional to their observed frequency. *Bottom*: the KL divergence from the class distribution of the state $(0, 0, 0, 0)$ to that of its k^{th} nearest neighbour.

that our distance metric is represented by Euclidean distance in this transformed space. The projection plot shows the SNCA metric arranging the states with regards to the similarity of their stochastic class distributions, which are depicted by the colour scale. The state $\boldsymbol{X} = (0, 0, 0, 0)$ represents the empty system and is leftmost in the projection plot. The lower plot explicitly lays out the states in the order of their distance from $(0, 0, 0, 0)$, and shows that the KL divergence of a neighbouring state's class distribution from that of $(0, 0, 0, 0)$ tends to increase as the neighbour moves further away.

To allow an appropriate comparison for plotting, we obtain the following two-dimensional NCA solution on the same dataset:

$$A = \begin{bmatrix} 341.23 & 234.95 & 317.98 & -78.4 \\ -15.27 & -10.11 & -14.54 & 3.8 \end{bmatrix}.$$

The projection and KL divergence plots corresponding to Figure 3 for this NCA solution are shown in Figure 4. The comparison between Figure 3 and Figure 4 highlights the deficiency of NCA when applied to discrete stochastic data, and the suitable corrections made by SNCA. Notably, the main direction of the NCA solution is less effective at discriminating the class distributions than the main direction of SNCA (see the top panels of Figures 2 and 3, x direction). This highlights the effectiveness of the SNCA adjustment to exclude a point's repeats from its probabilistic neighbour distribution. Additionally, the usefulness of the orthogonality constraints introduced in the SNCA implementation is apparent, as the two rows of the NCA solution are seen to each represent a very similar direction. In the lower plots of Figures 3 and 4, the 35 remaining states are displayed along the x-axis in the order of their distance from the state $(0, 0, 0, 0)$, with the KL divergence of their class distribution from that of $(0, 0, 0, 0)$ shown on the y-axis. Thus, we are looking for a positive correlation between the two axes; SNCA noticeably achieves this pattern more successfully than NCA. These figures demonstrate that the adjustment made by SNCA to exclude self-neighbours is effective; it forces the algorithm to find good quality, non-identical neighbours. NCA does not have this incentive, since identical $\boldsymbol{X}$ observations dominate its neighbour distributions and the influence of non-identical neighbours on its objective function is diminished.

The reference state of $(0, 0, 0, 0)$ used for the lower plots of Figure 3 and Figure 4 was not specially selected. Figure 5 shows this relationship between the distance ordering and the KL divergence averaged with respect to $\hat{q}_{\boldsymbol{X}}$ over all 36 states. Across the whole input space, SNCA has consistently

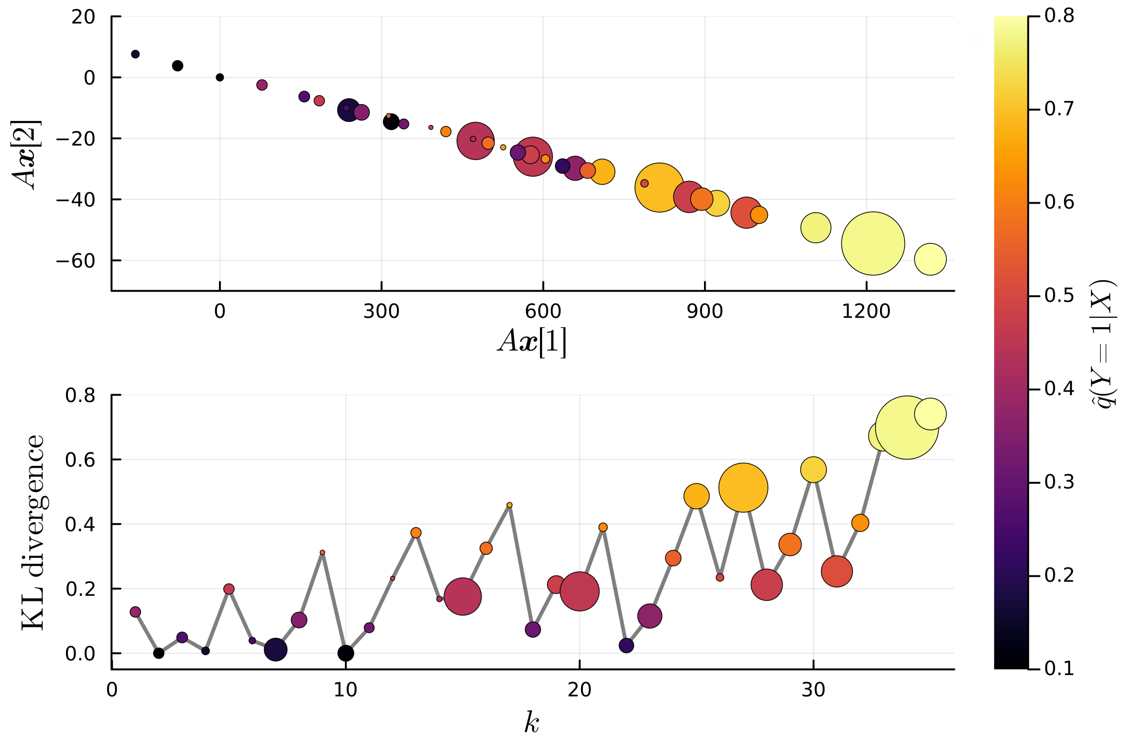

Figure 4. The equivalent plots from Figure 3 for a two-dimensional NCA solution. *Top*: the 36 unique states projected by the solution matrix. *Bottom*: the KL divergence from the class distribution of the state $(0, 0, 0, 0)$ to that of its k^{th} nearest neighbour.

arranged states' neighbours with broadly increasing divergence of their class distributions, whilst NCA is less reliable in this regard. For the purpose of nearest neighbour predictions, the closer neighbourhoods are the most relevant, and we see SNCA bringing a desirable structure in terms of the similarity of class distributions within the closer neighbourhoods.

4.1.2 kNN Classification.

As a further verification of metric performance, we display results of kNN classification under the learned distance metrics. To have a clearer interpretation of k in our data context, we consider k as

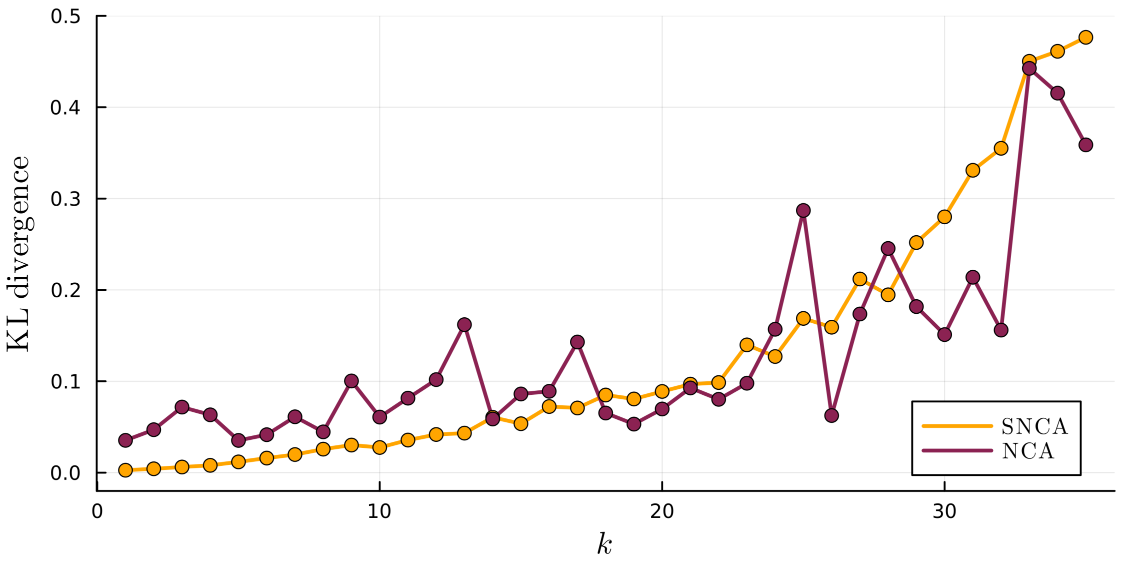

Figure 5. The average KL divergence from a point's class distribution to the class distribution of its k^{th} nearest neighbour, under the SNCA and NCA embeddings. This is equivalent to the lower plots of Figures 3 and 4 averaged with respect to $\hat{q}_{\boldsymbol{X}}$ over all states.

the number of distinct neighbours, and include all of their repeats. In other words, we classify an individual point, $\boldsymbol{x}_i$, by the majority class of its k nearest neighbour states, and additionally include any states which are tied in distance with the k^{th}. Further, we exclude the other observations of $\boldsymbol{x}_i$'s own state from the neighbour search. This would be the nearest neighbour state under any metric, so its inclusion does not help a comparison across metrics. Further details of the kNN procedures used in this section, and the generation of the figures, are found in Appendix A.3.

Since this example carries only a small number of distinct $\boldsymbol{X}$ values, we only show results for $k = 1$. Figure 6 concerns metrics trained on datasets of size n, and shows on the left the leave-one-out correct classification rates on a large (n = 100,000) test dataset. This is to approximate the expected classification accuracy on data drawn randomly from $q_{\boldsymbol{X},Y}$. As benchmarks for comparison, we include the 1NN classification rate achieved using a Euclidean distance metric, and a 1NN classifier which omnisciently selects the optimal neighbour, defined as the state whose class distribution has minimum KL divergence from that of the query point. We are encouraged to see SNCA still performing well with small training datasets, and achieving classification rates close to optimal given the level of stochasticity in the system.

In practical problems with large input spaces, we do not expect to exhaust $\mathcal{X}$ in our training sample. For this reason, we are particularly interested in classification performance on states which are unrepresented in the metric training. Figure 6 (right) uses 6-fold cross-validation (CV) on the 36 unique states, through which the metrics were trained on data covering 30 states, and the classification accuracy recorded on the remaining 6. We are encouraged to see strong 1NN classification performance by SNCA on unseen states. For each n marked in the plots in Figure 6, the metric training was repeated with 10 independent datasets of size n, with the ribbons showing pointwise ± 1.96 standard errors around the average classification rate. We see much greater variability in the NCA solutions as compared with SNCA.

The kNN performance of SNCA does not noticeably benefit by increasing the training set beyond a size of $n = 3000$. We can be confident that this is enough training data and the metric solution is not overfitted. In practice, we advise increasing the training set size for SNCA until the classification accuracy on test data, and the interpretations of the learned metric, appear consistent.

4.1.3 Execution Time.

To demonstrate the computational benefit of SNCA with discrete stochastic data, Figure 7 shows the mean execution time using a 1.6 GHz Intel Core i5 processor, for the metric training of SNCA and NCA across the 10 training sets of size n used in Figure 6 (left), ± 1.96 standard errors. The reported times include 10 initialisations for each optimisation. With the small state space in this example, all states are sampled in the smallest dataset size of $n = 1000$; after this, increasing the value of n only refines the empirical class distributions and does not increase the burden on SNCA. In contrast, a pointwise metric learning method such as NCA, which involves summation over n individual points, understandably suffers as n increases. It should be noted that when training NCA on the dataset of size $n = 100,000$ to illustrate the metric embedding in Section 4.1.1, the pointwise formulation was not feasible. Instead, we employed a modified algorithm in which identical points were grouped together in the calculations of the objective function and the gradient.

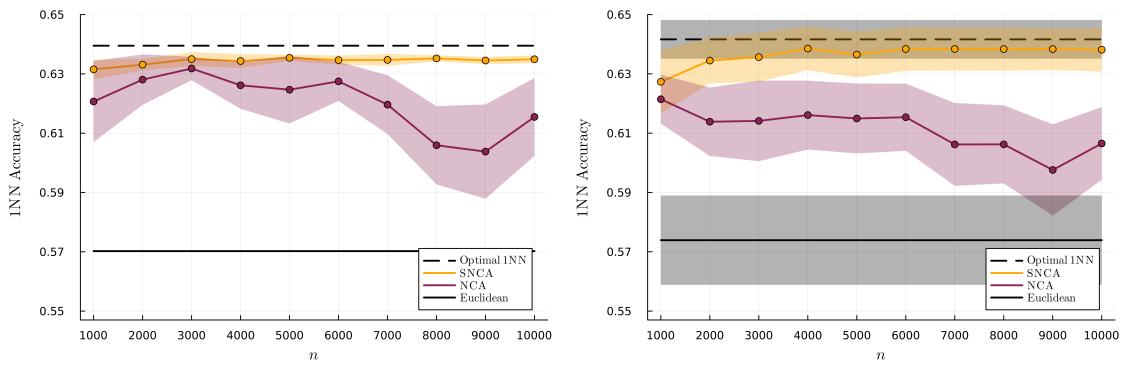

Figure 6. *Left*: the leave-one-out 1NN correct classification rate on a dataset of size 100,000, under metrics learnt from a training set of size n. *Right*: the 1NN classification performance on test states which were withheld from the metric training following 6-fold cross-validation. The shaded regions indicate ± 1.96 standard errors around the average classification accuracy.

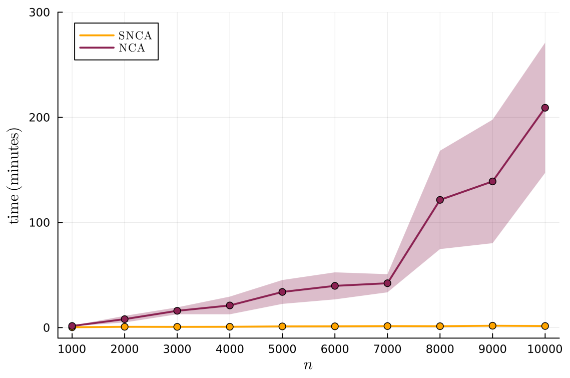

Figure 7. Metric learning execution time on a training set of size n. The shaded regions indicate ±1.96 standard errors following 10 independent trials for each n.

4.2 A Wafer Fab Model

As a more realistic example for discrete stochastic data, we explore the simulation model of a wafer fabrication facility (fab) described by Kayton et al. (1996). Wafers are released into this system at fixed intervals and follow a sequence involving several processing steps at a number of stations. Aspects of re-entrant flow, and machines offering different processing capacities and unpredictable breakdown patterns complicate the management of product flow through wafer fabs, and a simulation model becomes an indispensable tool. For this example, we have used Simul8 software to imitate the model described by Kayton et al. (1996).

To create a classification problem, we consider the practical task of predicting an early or late completion based on the system state on a wafer's release. To restrict the feasible state space and encourage repeated observations, we take an incomplete view of the system state, with $\boldsymbol{X}$ containing simply the queue sizes at each of the facility's 11 stations. The size of $\mathcal{X}$ is further limited by a total capacity constraint on the system, whereby new wafers cannot be released if the total work-in-process exceeds a threshold. The classification, Y, is observed as the wafer leaves the facility either before or after its due date, where the due date is pre-assigned by a constant inflation factor on its pure processing time. In this problem, we use a classification of $Y = 0$ to indicate an early or on-time wafer completion, while $Y = 1$ implies a late completion. Running the simulation provides an observation, $(\boldsymbol{X}, Y)$, for every wafer which passes through the system. A classification model from this simulated data can be applied on the actual wafer fab to alert planners to possible late deliveries.

The system design yields around 21,000 feasible combinations of the 11 queue sizes, although the distribution over this state space is highly imbalanced. Figure 8 (left) shows a dataset projected in the first two dimensions of an SNCA solution, where size is proportional to the observed frequency of the states. There is a small number of commonly visited states, while many more are observed very few times, and overall only a small percentage of the feasible space has been represented in the sample. To aid understanding of this plot, consider the point lying on the origin, labelled as state 1. In the original context, this represents the system state in which all queues are empty. This is a commonly observed state for the system. It is also a difficult state to classify; its colour indicates the $Y = 1$ class to have an observed proportion of around 0.55. In other words, a wafer released while the system has empty queues is marginally more likely to be completed late than on time. Although counter-intuitive, this is because the first station is a batch station which requires at least two wafers to be processed simultaneously. Therefore, a wafer released when the first queue is empty must invariably wait at least until the following wafer is released. Other states shown by the colour scale to have lower proportions of late completions represent states in which the first queue is non-empty.

The labelled state 2 in Figure 8 (left) represents the state in which six wafers occupy the queue at station 3 and all other queues are empty. In the system logic, the machine at station 3 is prone to breakdowns, which leads to the occasional build up of this station queue. This represents a rare yet critical system condition and accounts for the branch of lesser-observed states extending away from the main cluster. The class distributions of these states are heavily skewed in favour of $Y = 1$, as wafers entering a blocked system are highly likely to finish late. SNCA rightly identifies this third queue size as an effective indicator of the system performance and lets this variable dominate the first projection dimension.

Of primary interest is to understand metric performance with regards to states which are

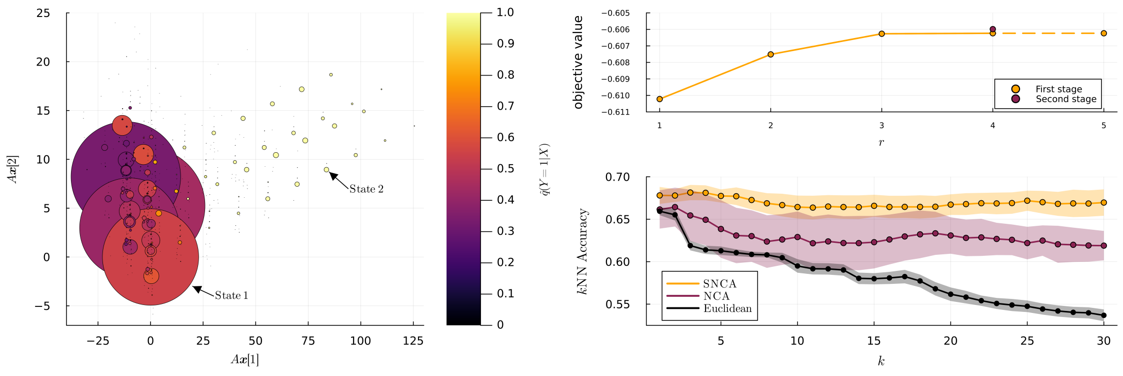

Figure 8. *Left*: using a dataset of size $n = 18{,}402$ from the wafer fab model, the states are projected by the first two rows of an SNCA solution. The colour of the points relates to their observed class proportions, and the size is proportional to their observed frequency. *Top right*: the objective value improvement throughout the SNCA algorithm. *Bottom right*: the kNN correct classification rate on rarely seen states, with the shaded regions indicating ± 1.96 standard errors.

underrepresented in the training sample, since these are the states for which we are more reliant on neighbours to make a prediction. Therefore, as a more targeted approach, we show the kNN accuracy on the rarely seen states. In particular, we consider states observed in the dataset with fewer than 25 repetitions. Figure 8 (bottom right) shows the average kNN accuracy on these states ± 1.96 standard errors, after repeating the experiment with 10 simulated datasets. Additional details on the generation of this plot are provided in Appendix A.3.

To provide context to the kNN accuracy results, the system was simulated to give roughly equal proportions of early and late wafers. Employing kNN on the simplified state description used in this example provides additional predictive power, and the benefit of a trained distance metric is evident from the comparison with Euclidean distance. Reaching around 68% accuracy following metric learning represents a marked and useful achievement considering the stochasticity of the system. This example is broadly representative of the type of data we aim to cater to with SNCA, and the further performance improvement brought over NCA is encouraging.

4.3 Discussion

The proposed SNCA constitutes a metric learning method designed under the assumption that data contain repeating input features with stochastic classifications, a scenario which is characteristic of discrete-event simulations, our intended application. The probabilistic formulation uses a multinomial framework to accommodate observations from a discrete and finite data space. We recognise that SNCA represents a *linear* and *global* metric learning method, in that it learns a single linear projection to apply to all data points. It is not designed to accommodate non-linear or value-dependant relationships among the predictors and the response; the suitability of this limitation should be considered with each application.

Nonetheless, the assumptions of SNCA are broadly characteristic of classification problems within stochastic simulation, and this section presented examples from this domain to explore its performance on such data. Initially, a simple simulation model was chosen to provide an example with a small state space and intuitive system behaviour against which to interpret the SNCA metric. Figures showed the desirable property of the SNCA solution in arranging states in the projected space according to the similarity of their various class distributions.

We note that applications in the simulation domain often have very low sampling cost, and additional trials on this initial example indicate the computational advantage of the SNCA formulation with increasing dataset size. However, the advantages seen in typical applications with larger state spaces will be less pronounced, since the small state space in this example was comprehensively sampled such that incremental sampling provided no additional contributions to the objective function summation. Further, the scaling of SNCA with dimensionality has not been explicitly explored, although we recognise that the suggested optimisation approach (Section 3.4) scales well owing to the row-wise construction of a good initial solution in the first stage, and the scalability of the second-stage matrix optimisation relies upon the gradient descent procedure. While we thus find practical advantages through the suggested optimisation scheme, our main interest in this paper is towards the behaviour of the SNCA projection when applied to simulation-generated data.

A second simulation example was presented to provide a more realistic dataset with greater dimension and state space. This example also showed good projection behaviour from SNCA and improved performance compared to NCA when using the distance metric for kNN classification. Compared with NCA across all examples we have looked at offering discrete stochastic data, SNCA was found to give equivalent or improved kNN performance.

5 Conclusion

In this work, we proposed a method of distance metric learning which we refer to as Stochastic Neighbourhood Components Analysis (SNCA). The aim of SNCA is to extend the reach of metric learning to scenarios involving repeated feature vectors with stochastic labelling. This is a type of data which is characteristic of classification problems arising from the sample paths of stochastic simulation. Interest in probing the dynamic behaviour of simulation is growing, and to this purpose we demonstrated SNCA for nearest neighbour classification on sample path data. This is a useful application for SNCA; the ability to anticipate stochastic behaviour provides valuable assistance to real-time planning and control in a live system and is well supported by the emergence of digital twin simulation, whilst an SNCA solution matrix can reveal the influential drivers of a system's stochastic behaviour. Experimental results show SNCA to be effective, and to bring further improvement to kNN performance over NCA when applied to this type of data.

We recognise numerous contexts to which the model of discrete stochastic data extends itself. For example, crowdsourcing represents an increasingly attractive solution to data generation, and will often result in an instance being assigned to various classes. Although we have focused our motivation on simulation analytics, the scope of SNCA stretches further and we see possibility for a wider impact.

Acknowledgments

The authors thank the editors and anonymous reviewers for their insightful comments and constructive feedback, which have improved the quality and clarity of this manuscript.

Funding

This work was supported by the EPSRC funded STOR-i Centre for Doctoral Training at Lancaster University under Grant number EP/L015692/1. In addition, Barry L. Nelson's work was supported by the National Science Foundation of the United States under Grant number DMS-1854562.

Data Ethics & Reproducibility Note

The code capsule is available on Code Ocean at https://codeocean.com/capsule/3348428/tree/v5. No data ethics considerations are foreseen related to this paper.

References

Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D. (2003). Learning distance functions using equivalence relations. In *Proceedings of the 20th International Conference on Machine Learning*, pages 11–18.

Bellet, A., Habrard, A., and Sebban, M. (2013). A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709.*

Cao, L. and Tay, F. E. (2001). Financial forecasting using support vector machines. *Neural Computing & Applications*, 10(2):184–192.

Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007). Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 209–216.

Devroye, L. P. and Wagner, T. (1980). Distribution-free consistency results in nonparametric discrimination and regression function estimation. *The Annals of Statistics*, pages 231–239.

dos Santos, C. H., Montevechi, J. A. B., de Queiroz, J. A., de Carvalho Miranda, R., and Leal, F. (2022). Decision support in productive processes through des and abs in the digital twin era: a systematic literature review. *International Journal of Production Research*, 60(8):2662–2681.

Dutta, U. K., Harandi, M., and Sekhar, C. C. (2020). Unsupervised deep metric learning via orthogonality based probabilistic loss. *IEEE Transactions on Artificial Intelligence*, 1(1):74–84.

Globerson, A. and Roweis, S. (2005). Metric learning by collapsing classes. *Advances in Neural Information Processing Systems*, 18:451–458.

Goldberger, J., Roweis, S. T., Hinton, G. E., and Salakhutdinov, R. R. (2005). Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, pages 513–520.

Hong, L. J. and Jiang, G. (2019). Offline simulation online application: A new framework of simulation-based decision making. *Asia-Pacific Journal of Operational Research*, 36(06):1–22.

Horvat, T. and Job, J. (2020). The use of machine learning in sport outcome prediction: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(5):e1380.

Jiang, G., Hong, L. J., and Nelson, B. L. (2020). Online risk monitoring using offline simulation. *INFORMS Journal on Computing*, 32(2):356–375.

Kayton, D., Teyner, T., Schwartz, C., and Uzsoy, R. (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*, pages 49–56. IEEE.

Kedem, D., Tyree, S., Sha, F., Lanckriet, G. R., and Weinberger, K. Q. (2012). Non-linear metric learning. In *Advances in Neural Information Processing Systems*, pages 2582–2590.

Keslin, G., Nelson, B. L., Pagnoncelli, B. K., Plumlee, M., and Rahimian, H. (2024). Ranking and contextual selection. *Operations Research*, 0(0).

Kolmogorov, A. N. and Širjaev, A. N. (1992). *Selected works of AN Kolmogorov. Vol. 2, Probability theory and mathematical statistics.* Kluwer.

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

Laidler, G., Morgan, L. E., Nelson, B. L., and Pavlidis, N. G. (2020). Metric learning for simulation analytics. In *Proceedings of the 2020 Winter Simulation Conference*, pages 349–360. IEEE.

Law, A. M. and Kelton, W. D. (2007). *Simulation Modeling and Analysis, 3rd edition.* Mcgraw-Hill New York.

Li, D. and Tian, Y. (2018). Survey and experimental study on metric learning methods. *Neural Networks*, 105:447–462.

Lin, Y., Nelson, B. L., and Pei, L. (2019). Virtual statistics in simulation via k nearest neighbors. *INFORMS Journal on Computing*, 31(3):576–592.

Mahalanobis, P. (1936). On the generalised distance in statistics. In *Proceedings of the National Institute of Science of India*, volume 2, pages 49–55.

McFee, B. and Lanckriet, G. R. (2010). Metric learning to rank. In *International Conference on Machine Learning*.

Morgan, L. E. and Barton, R. R. (2022). Fourier trajectory analysis for system discrimination. *European Journal of Operational Research*, 296(1):203–217.

Nelson, B. L. (2016). 'Some tactical problems in digital simulation' for the next 10 years. *Journal of Simulation*, 10(1):2–11.

Nelson, B. L. and Pei, L. (2021). *Foundations and Methods of Stochastic Simulation: A First Course, 2nd edition.* Springer Nature.

Ouyang, H. and Nelson, B. L. (2017). Simulation-based predictive analytics for dynamic queueing systems. In *Proceedings of the 2017 Winter Simulation Conference*, pages 1716–1727. IEEE.

Peltonen, J. and Kaski, S. (2005). Discriminative components of data. *IEEE Transactions on Neural Networks*, 16(1):68–83.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2021). *Lectures on Stochastic Programming: Modeling and Theory.* SIAM.

Shen, H., Hong, L. J., and Zhang, X. (2021). Ranking and selection with covariates for personalized decision making. *INFORMS Journal on Computing*, 33(4):1500–1519.

Suárez, J. L., García, S., and Herrera, F. (2021). A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing*, 425:300–322.

Suo, Q., Ma, F., Yuan, Y., Huai, M., Zhong, W., Gao, J., and Zhang, A. (2018). Deep patient similarity learning for personalized healthcare. *IEEE Transactions on NanoBioscience*, 17(3):219–227.

Tarlow, D., Swersky, K., Charlin, L., Sutskever, I., and Zemel, R. (2013). Stochastic k-neighborhood selection for supervised and unsupervised learning. In *International Conference on Machine Learning*, pages 199–207. PMLR.

Torresani, L. and Lee, K. (2007). Large margin component analysis. *Advances in Neural Information Processing Systems*, 19:1385.

Urquhart, M., Ljungskog, E., and Sebben, S. (2020). Surrogate-based optimisation using adaptively scaled radial basis functions. *Applied Soft Computing*, 88:106050.

Vaughan, J. W. (2017). Making better use of the crowd: How crowdsourcing can advance machine learning research. *Journal of Machine Learning Research*, 18(1):7026–7071.

Wang, D. and Tan, X. (2014). Robust distance metric learning in the presence of label noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(2).

Weinberger, K. Q. and Tesauro, G. (2007). Metric learning for kernel regression. In *Artificial Intelligence and Statistics*, pages 612–619. PMLR.

Xie, P., Wu, W., Zhu, Y., and Xing, E. (2018). Orthogonality-promoting distance metric learning: Convex relaxation and theoretical analysis. In *International Conference on Machine Learning*, pages 5403–5412. PMLR.

Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. (2002). Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems*, 15:521–528.

Yang, X. (2020). *Essays on Distance Metric Learning.* PhD thesis, UCL (University College London).

Appendix A.1. Proof of Uniform Convergence in Theorem 1

Theorem 1 *Suppose there exists a compact subset $C \subset \mathbb{R}^{d\times d}$ such that:*

(i) $A_g^\star$ is nonempty and contained in C,

(ii) for sufficiently large n, with probability 1, $\hat{A}_g$ is nonempty and contained in C,

(iii) $g(A)$ is finite-valued and continuous on C, and

(iv) $g_n(A) \overset{a.s.}{\to} g(A)$ as $n \to \infty$, uniformly on C.

Then $g_n(\hat{A}_g) \overset{a.s.}{\to} g(A_g^\star)$ and $\mathbb{D}(\hat{A}_g, A_g^\star) \overset{a.s.}{\to} 0$ as $n \to \infty$.

Letting $\boldsymbol{b}_{lh}$ denote $\boldsymbol{b}_l - \boldsymbol{b}_h$, we establish the uniform convergence of

$$g_n(A) \overset{a.s.}{\to} \sum_{l=1}^{m}\sum_{y\in\mathcal{Y}} q(\boldsymbol{b}_l, y)\log\left(q(\boldsymbol{b}_l)\frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}\right), \tag{5}$$

as required by condition *(iv)* of Theorem 1. Recall that

$$\begin{aligned} g_n(A) &= \sum_{l=1}^{m}\sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y)\log p^A(\boldsymbol{b}_l, y) \\ &= \sum_{l=1}^{m}\sum_{y\in\mathcal{Y}} \hat{q}(\boldsymbol{b}_l, y)\log\left(\hat{q}(\boldsymbol{b}_l)\frac{\sum_{h\neq l} \hat{q}(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} \hat{q}(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}\right). \end{aligned}$$

We begin by establishing the convergence of

$$p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l) = \frac{\hat{q}(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} \hat{q}(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}} \to \frac{q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}. \tag{6}$$

We assume that the following condition holds for some $\delta > 0$:

$$q(\boldsymbol{b}_l) > \delta \quad \forall l = 1, 2, \ldots, m. \tag{7}$$

We have that $\hat{q}(\boldsymbol{b}_h, y) \overset{a.s.}{\to} q(\boldsymbol{b}_h, y)$ and $\hat{q}(\boldsymbol{b}_h) \overset{a.s.}{\to} q(\boldsymbol{b}_h)$ $\forall h, y$ as $n \to \infty$. This can be expressed in the following way:

Let $\epsilon > 0$. Then $\exists$ N s.t. $\forall h \in \{1, 2, \ldots, m\}$ and $\forall y \in \{0, 1\}$, with probability 1 $\forall n > N$,

$$\begin{aligned} |\hat{q}(\boldsymbol{b}_h, y) - q(\boldsymbol{b}_h, y)| &< \frac{\epsilon\delta^2}{m+\epsilon\delta}, \quad \text{(note that } \frac{\epsilon\delta^2}{m+\epsilon\delta} < \frac{\epsilon\delta^2}{m} < \epsilon) \\ |\hat{q}(\boldsymbol{b}_h) - q(\boldsymbol{b}_h)| &< \frac{\epsilon\delta^2}{m+\epsilon\delta}. \end{aligned}$$

To target the convergence of (6), consider the absolute difference when $n > N$:

$$
\begin{aligned}
&\left|\frac{\hat{q}(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l}\hat{q}(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}} - \frac{q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}\right| \\
=&\left|\frac{\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}\left[\sum_{k\neq l}\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}(\hat{q}(\boldsymbol{b}_h, y)q(\boldsymbol{b}_k) - q(\boldsymbol{b}_h, y)\hat{q}(\boldsymbol{b}_k))\right]}{\left(\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}\right)\left(\sum_{k\neq l}\hat{q}(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}\right)}\right| \\
=&\left|\frac{\begin{array}{c}\exp\left\{-\|A\boldsymbol{b}_{lh}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\cdot \\ \left[\sum_{k\neq l}\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}(\hat{q}(\boldsymbol{b}_h, y)q(\boldsymbol{b}_k) - q(\boldsymbol{b}_h, y)\hat{q}(\boldsymbol{b}_k))\right]\end{array}}{\left(\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\right)\left(\sum_{k\neq l}\hat{q}(\boldsymbol{b}_k)\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\right)}\right| \\
&\text{where } r_l = \arg\min_{k\neq l}\|A\boldsymbol{b}_{lk}\|_2^2 \\
=&\left|\frac{\begin{array}{c}\exp\left\{-\|A\boldsymbol{b}_{lh}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\cdot \\ \left[\sum_{k\neq l}\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}(\hat{q}(\boldsymbol{b}_h, y)q(\boldsymbol{b}_k) - q(\boldsymbol{b}_h, y)\hat{q}(\boldsymbol{b}_k))\right]\end{array}}{\begin{array}{c}\left(q(\boldsymbol{b}_{r_l}) + \sum_{k\neq l, r_l} q(\boldsymbol{b}_k)\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\right)\cdot \\ \left(\hat{q}(\boldsymbol{b}_{r_l}) + \sum_{k\neq l, r_l}\hat{q}(\boldsymbol{b}_k)\exp\left\{-\|A\boldsymbol{b}_{lk}\|_2^2 + \|A\boldsymbol{b}_{lr_l}\|_2^2\right\}\right)\end{array}}\right| \\
&\text{Every exponential term is } \leq 0. \text{ The summations in the denominator are } > 0. \\
&\text{Therefore,} \\
\leq&\left|\frac{1\cdot\sum_{k\neq l} 1\cdot(\hat{q}(\boldsymbol{b}_h, y)q(\boldsymbol{b}_k) - q(\boldsymbol{b}_h, y)\hat{q}(\boldsymbol{b}_k))}{q(\boldsymbol{b}_{r_l})\hat{q}(\boldsymbol{b}_{r_l})}\right| \\
\leq&\frac{\sum_{k\neq l}|\hat{q}(\boldsymbol{b}_h, y)q(\boldsymbol{b}_k) - q(\boldsymbol{b}_h, y)\hat{q}(\boldsymbol{b}_k)|}{q(\boldsymbol{b}_{r_l})\hat{q}(\boldsymbol{b}_{r_l})} \\
=&\frac{\sum_{k\neq l}|(\hat{q}(\boldsymbol{b}_h, y) - q(\boldsymbol{b}_h, y))q(\boldsymbol{b}_k) + q(\boldsymbol{b}_h, y)(q(\boldsymbol{b}_k) - \hat{q}(\boldsymbol{b}_k))|}{q(\boldsymbol{b}_{r_l})\hat{q}(\boldsymbol{b}_{r_l})} \\
\leq&\frac{\sum_{k\neq l} q(\boldsymbol{b}_k)|\hat{q}(\boldsymbol{b}_h, y) - q(\boldsymbol{b}_h, y)| + q(\boldsymbol{b}_h, y)|q(\boldsymbol{b}_k) - \hat{q}(\boldsymbol{b}_k)|}{q(\boldsymbol{b}_{r_l})\hat{q}(\boldsymbol{b}_{r_l})} \\
\leq&\frac{\left(\frac{\epsilon\delta^2}{m+\epsilon\delta}\right)\sum_{k\neq l}(q(\boldsymbol{b}_k) + q(\boldsymbol{b}_h, y))}{q(\boldsymbol{b}_{r_l})\hat{q}(\boldsymbol{b}_{r_l})} \\
\leq&\frac{\left(\frac{\epsilon\delta^2}{m+\epsilon\delta}\right)m}{\delta\left(\delta - \frac{\epsilon\delta^2}{m+\epsilon\delta}\right)} \quad \left(\text{note that } \frac{\epsilon\delta^2}{m+\epsilon\delta} < \delta\right) \qquad (8) \\
=&\frac{\epsilon\delta^2 m}{\delta^2(m+\epsilon\delta) - \epsilon\delta^3} \\
=&\epsilon
\end{aligned}
$$

This shows that the convergence (6) holds uniformly over A. (Note that r_l depends on A, but the condition (7) ensures that the bound (8) holds for all r_l, i.e. for all A.)

To establish the uniform convergence of (5), we need to give attention to the logarithm function. Following the uniform convergence of (6), simple results regarding the sums and products of convergent sequences ensure that the argument of the logarithms in $g_n(A)$ converge uniformly. We also note that the logarithm is a uniformly continuous function on $[a, \infty), a > 0$ (a bounded derivative implies uniform continuity). Combining these two facts provides uniform convergence of the logarithms:

Let $\epsilon > 0$. Then $\exists\ \xi > 0$ s.t. for every $x, y \in [a, \infty)$ with $|x - y| < \xi$, we have that $|\log(x) - \log(y)| < \epsilon$ (uniform continuity of the logarithm on $[a, \infty), a > 0$). Using this $\xi, \exists\ N$ s.t. $\left|p^A(\boldsymbol{b}_l, y) - q(\boldsymbol{b}_l)\frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}\right| < \xi\ \forall n > N, \forall A$ (uniform convergence of the argument of the logarithms in $g_n(A)$). Provided that $p^A(\boldsymbol{b}_l, y) \in [a, \infty)$ and $q(\boldsymbol{b}_l)\frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}} \in [a, \infty)$, this gives

$$\left|\log(p^A(\boldsymbol{b}_l, y)) - \log\left(q(\boldsymbol{b}_l)\frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}\right)\right| < \epsilon \quad \forall n > N, \forall A$$

$$\implies \log(p^A(\boldsymbol{b}_l, y)) \to \log\left(q(\boldsymbol{b}_l)\frac{\sum_{h\neq l} q(\boldsymbol{b}_h, y)\exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} q(\boldsymbol{b}_k)\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}\right) \text{ uniformly over } A.$$

Therefore, the logarithms in $g_n(A)$ will preserve uniform convergence as long as their arguments $\in [a, \infty)$. The condition given by (4) in Section 3.3 ensures that this holds. With this, we establish the uniform convergence of (5).

Appendix A.2. Gradient Derivation

We derive the gradient expression for $\partial g_n(A)/\partial A$ given in Section 3.4. We let $\boldsymbol{b}_{lh}$ denote $\boldsymbol{b}_l - \boldsymbol{b}_h$. Recalling

$$p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l) = \frac{c_h^y \exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}}{\sum_{k\neq l} c_k \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}}, \quad p^A((\boldsymbol{b}_l, y)|\boldsymbol{b}_l) = 0,$$
$$p_{lh} = \sum_{y\in\mathcal{Y}} p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l),$$

and using that $d\|A\boldsymbol{x}\|_2^2/dA = 2A\boldsymbol{x}\boldsymbol{x}^\top$, we have

$$\begin{aligned}
\frac{\partial p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)}{\partial A} &= \frac{\begin{matrix}-2A\boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top c_h^y \exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}\sum_{k\neq l} c_k\exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\} \\ - c_h^y \exp\{-\|A\boldsymbol{b}_{lh}\|_2^2\}\sum_{k\neq l} -2A\boldsymbol{b}_{lk}\boldsymbol{b}_{lk}^\top c_k \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}\end{matrix}}{\left(\sum_{k\neq l} c_k \exp\{-\|A\boldsymbol{b}_{lk}\|_2^2\}\right)^2} \\
&= 2A\left\{-\boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l) + p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)\sum_{k\neq l}\boldsymbol{b}_{lk}\boldsymbol{b}_{lk}^\top p_{lk}\right\} \\
&= 2A\left\{p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)\left(\left(\sum_{k\neq l}\boldsymbol{b}_{lk}\boldsymbol{b}_{lk}^\top p_{lk}\right) - \boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top\right)\right\}.
\end{aligned}$$

Therefore, making use of the identity, $p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l) = p_{lh}\hat{q}(y|\boldsymbol{b}_h)$,

$$\begin{aligned}
\frac{\partial \sum_h p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)}{\partial A} &= 2A\sum_{h\neq l}\left\{p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)\left(\left(\sum_{k\neq l}\boldsymbol{b}_{lk}\boldsymbol{b}_{lk}^\top p_{lk}\right) - \boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top\right)\right\} \\
&= 2A\left\{p^A(y|\boldsymbol{b}_l)\sum_{k\neq l}\boldsymbol{b}_{lk}\boldsymbol{b}_{lk}^\top p_{lk} - \sum_{h\neq l}\boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh}\hat{q}(y|\boldsymbol{b}_h)\right\} \\
&= 2A\sum_{h\neq l}\boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh}(p^A(y|\boldsymbol{b}_l) - \hat{q}(y|\boldsymbol{b}_h)).
\end{aligned}$$

Now the gradient expression can be reached as follows:

$$\begin{aligned}
g_n(A) &= \sum_{l=1}^m\sum_{y\in\mathcal{Y}}\hat{q}(\boldsymbol{b}_l, y)\log\left(\hat{q}(\boldsymbol{b}_l)\sum_h p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)\right), \\
\frac{\partial g_n(A)}{\partial A} &= 2A\sum_{l=1}^m\sum_{y\in\mathcal{Y}}\hat{q}(\boldsymbol{b}_l, y)\frac{\hat{q}(\boldsymbol{b}_l)\sum_{h\neq l}\boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh}(p^A(y|\boldsymbol{b}_l) - \hat{q}(y|\boldsymbol{b}_h))}{\hat{q}(\boldsymbol{b}_l)\sum_h p^A((\boldsymbol{b}_h, y)|\boldsymbol{b}_l)} \\
&= 2A\sum_{l=1}^m\sum_{y\in\mathcal{Y}}\hat{q}(\boldsymbol{b}_l, y)\frac{\sum_{h\neq l}\boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh}(p^A(y|\boldsymbol{b}_l) - \hat{q}(y|\boldsymbol{b}_h))}{p^A(y|\boldsymbol{b}_l)} \\
&= 2A\sum_{l=1}^m\sum_{y\in\mathcal{Y}}\hat{q}(\boldsymbol{b}_l, y)\sum_{h\neq l}\boldsymbol{b}_{lh}\boldsymbol{b}_{lh}^\top p_{lh}\left(1 - \frac{\hat{q}(y|\boldsymbol{b}_h)}{p^A(y|\boldsymbol{b}_l)}\right).
\end{aligned}$$

Appendix A.3. kNN Procedures

To assess the practical performance of the distance metrics considered in the paper, we performed kNN classification in which k refers to the number of unique, non-identical neighbours and includes all of their repeats. Algorithm 2 outlines the procedure that was followed. We use $\mathcal{D}^{\text{test}}$ to denote the set of query points which cover the subset of states, $\mathcal{X}_{\text{test}}$. For the state $\boldsymbol{b}_l \in \mathcal{X}_{\text{test}}$, we use $c_l^y(\text{test})$ to denote the frequency of the pair $(\boldsymbol{b}_l, y)$ in $\mathcal{D}^{\text{test}}$. Similarly, $\mathcal{D}^{\text{train}}$, $\mathcal{X}_{\text{train}}$, and $c_l^y(\text{train})$ relate to the set of training points. The following figures in the paper are specifically constructed from Algorithm 2 as follows:

1. **Figure 6 (left):** The metrics are trained on a dataset, $\mathcal{D}_n$, where n is shown on the x-axis. For each metric, the leave-one-out 1NN accuracy is calculated on a fixed dataset, $\mathcal{D}_{100,000}$. That is, Algorithm 2 is applied with $\mathcal{X}_{\text{test}} = \mathcal{X}_{\text{train}} = \mathcal{X}$ and $\mathcal{D}^{\text{test}} = \mathcal{D}^{\text{train}} = \mathcal{D}_{100,000}$. For each n marked on the x-axis, we repeat the process for 10 independent datasets of $\mathcal{D}_n$, with the average classification accuracy and ± 1.96 standard errors plotted.

2. **Figure 6 (right):** We begin with a dataset, $\mathcal{D}_n$, where n is shown on the x-axis, and perform 6-fold CV on the 36 states in $\mathcal{X}$. For each fold, $|\mathcal{X}_{\text{test}}| = 6$ and $|\mathcal{X}_{\text{train}}| = 30$, and the metrics are trained on the dataset constructed as $\{c_l^y \in \mathcal{D}_n \mid \boldsymbol{b}_l \in \mathcal{X}_{\text{train}}\}$. The 1NN accuracy results are again obtained from a fixed dataset $\mathcal{D}_{100,000}$. We construct $\mathcal{D}^{\text{test}} = \{c_l^y \in \mathcal{D}_{100,000} \mid \boldsymbol{b}_l \in \mathcal{X}_{\text{test}}\}$, and $\mathcal{D}^{\text{train}} = \{c_l^y \in \mathcal{D}_{100,000} \mid \boldsymbol{b}_l \in \mathcal{X}_{\text{train}}\}$. We plot the average 1NN accuracy and ± 1.96 standard errors over the 6 folds and after repeating the metric training over 10 independent datasets of $\mathcal{D}_n$. The specific partitioning of the CV folds remained consistent throughout to create a fair comparison over n.

3. **Figure 8 (bottom right):** 10 datasets of $\mathcal{D}_{18,402} = \{c_l^y\}_{l:\,\boldsymbol{b}_l \in \mathcal{X}}^{y \in \mathcal{Y}}$ were used. For each dataset, the metric and kNN training states were taken to be those with at least 25 observations, and those with fewer than 25 observations made up the test states. Specifically, $\mathcal{X}_{\text{train}} = \{\boldsymbol{b}_l \in \mathcal{X} \mid c_l \geq 25\}$, $\mathcal{X}_{\text{test}} = \{\boldsymbol{b}_l \in \mathcal{X} \mid c_l < 25\}$, $\mathcal{D}^{\text{train}} = \{c_l^y \in \mathcal{D}_{18,402} \mid c_l \geq 25\}$, and $\mathcal{D}^{\text{test}} = \{c_l^y \in \mathcal{D}_{18,402} \mid c_l < 25\}$. From the 10 repetitions, the plot shows the average kNN accuracy and ± 1.96 standard errors over a range of k.

Algorithm 2 kNN

```
Input: $\mathcal{X}_{\text{test}}, \mathcal{X}_{\text{train}} \subseteq \mathcal{X}$,
  $\mathcal{D}^{\text{test}} = \{c_l^y(\text{test})\}_{l:\,\boldsymbol{b}_l \in \mathcal{X}_{\text{test}}}^{y \in \mathcal{Y}}$,
  $\mathcal{D}^{\text{train}} = \{c_l^y(\text{train})\}_{l:\,\boldsymbol{b}_l \in \mathcal{X}_{\text{train}}}^{y \in \mathcal{Y}}$,
  $A$,
  $k < |\mathcal{X}_{\text{train}}|$
count $\leftarrow 0$
for $\boldsymbol{b}_l \in \mathcal{X}_{\text{test}}$ do
    $\mathcal{N}_l^k \leftarrow \arg\min_{\boldsymbol{b}_h \in \mathcal{X}_{\text{train}} \setminus \{\boldsymbol{b}_l\}} \|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2$
    while $|\mathcal{N}_l^k| < k$ do
        $\mathcal{N}_l^k \leftarrow \mathcal{N}_l^k \cup \{\arg\min_{\boldsymbol{b}_h \in \mathcal{X}_{\text{train}} \setminus (\{\boldsymbol{b}_l\} \cup \mathcal{N}_l^k)} \|A\boldsymbol{b}_l - A\boldsymbol{b}_h\|_2\}$
    end while
    $\hat{y} \leftarrow \arg\max_{y \in \mathcal{Y}} \{\sum_{\boldsymbol{b}_h \in \mathcal{N}_l^k} c_h^y(\text{train})\}$
    count $\leftarrow$ count $+ c_l^{\hat{y}}(\text{test})$
end for
Return: Accuracy $\leftarrow$ count$/ \sum_{\boldsymbol{b}_l \in \mathcal{X}_{\text{test}}} \sum_{y \in \mathcal{Y}} c_l^y(\text{test})$
```