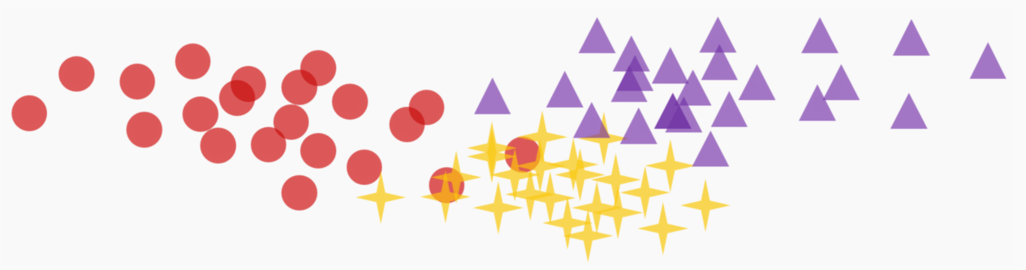
Metric Learning with Stochastic Data

Graham Laidler, Lucy Morgan, Barry Nelson, Nicos Pavlidis

Metric Learning...



- Distance calculations among data points are fundamental to many machine learning techniques.
  - E.g. **nearest neighbour** (NN) predictions, **clustering**, **information retrieval**...
- Metric learning tunes a **problem-specific** distance metric from **supervised data**.

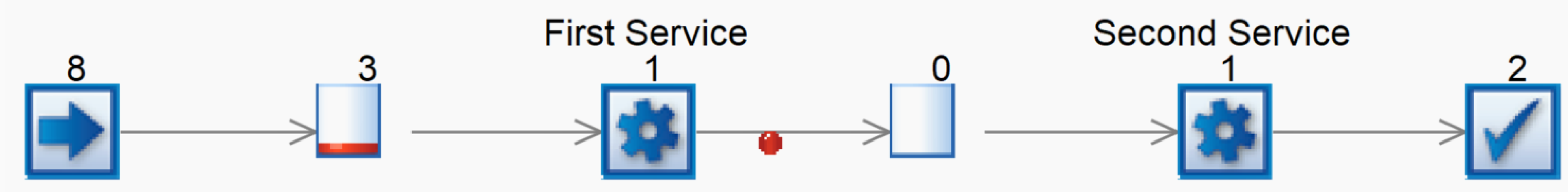
...for Discrete-Event Simulation

Given the **current state** of a simulation model, we want to **predict** something about its **future performance**:

- How long should a customer expect to wait given the system state on their arrival?
- Do we expect some condition (e.g. queues at full capacity) to be reached in the next T time units?

We can use metric learning to **improve** the performance of NN predictions.

For these problems, the **input** (the system state) will typically be **multivariate** and **discrete**, and the **output** (the future performance) will be **stochastic**.



[1]

The Data

multivariate system state → $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \vdots \end{bmatrix}$ ← e.g. queue size; e.g. number of busy servers

$\boldsymbol{X} \in \mathcal{X} = \{\boldsymbol{b}_1,\ \boldsymbol{b}_2,\ \dots,\ \boldsymbol{b}_m\}$ ← finite state space

stochastic outcome → $Y \in \mathcal{Y} = \{0, 1, \dots\}$

We observe pairs $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$ as the simulation runs.

the data ... → $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n} \overset{\text{i.i.d.}}{\sim} q(\boldsymbol{x}, y)$ ← unknown distribution

...as counts →
$$C_l^y = |\{i\colon \boldsymbol{x}_i = \boldsymbol{b}_l, y_i = y\}|$$
$$C_l = |\{i\colon \boldsymbol{x}_i = \boldsymbol{b}_l\}|$$

MLEs → $\hat{q}(y|\boldsymbol{b}_l) = \dfrac{C_l^y}{C_l}$

We want to find a distance metric $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which reflects **similarity** of the observed **class distributions** $\hat{q}(y|\boldsymbol{b}_l)$.

The Method

$$d_A(\boldsymbol{b}_l, \boldsymbol{b}_h) = \|A(\boldsymbol{b}_l - \boldsymbol{b}_h)\|_2$$
← Euclidean distance after linear transformation

Continuously model the NN distribution under d_A:

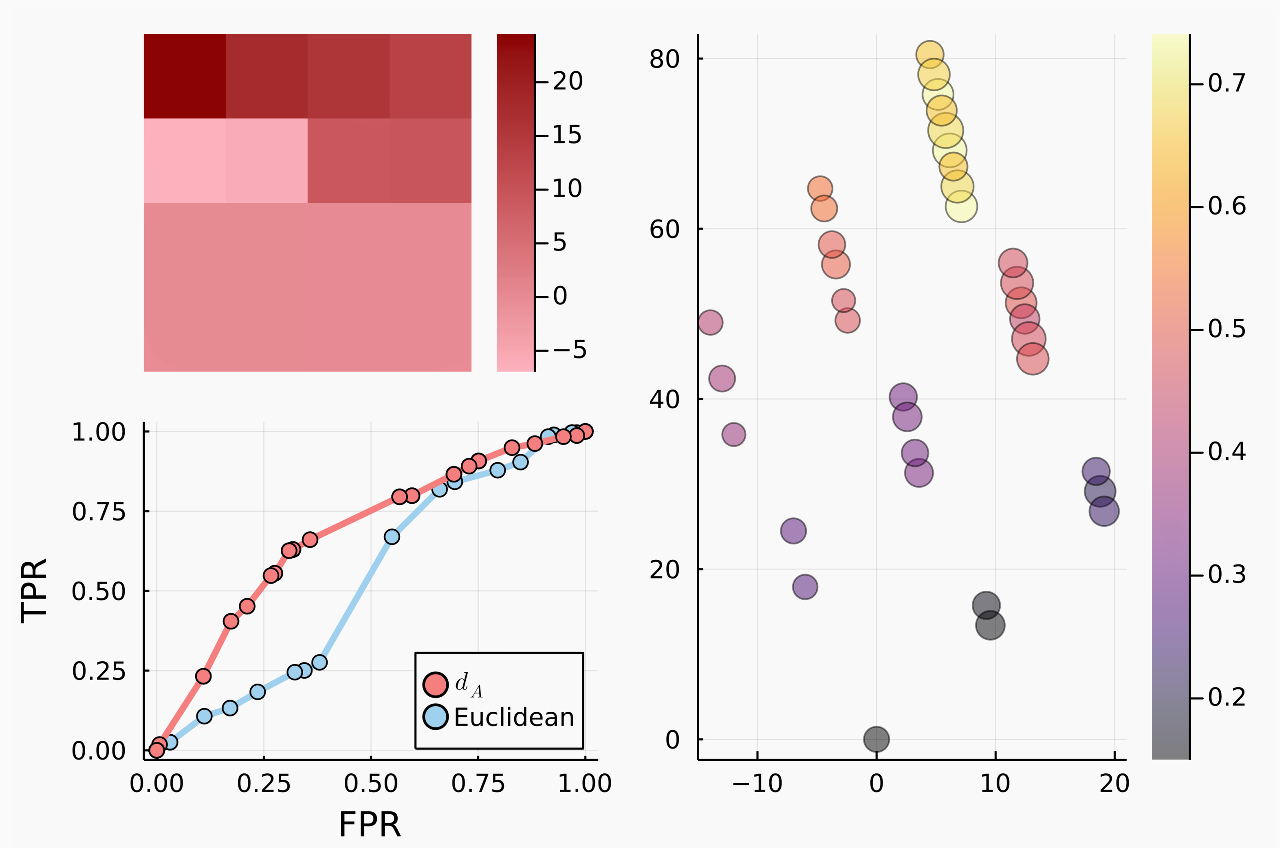
probability $\boldsymbol{b}_h$ is NN to $\boldsymbol{b}_l$, define $p_{ll} = 0$ →
$$p_{lh} = \underbrace{\frac{C_h \exp\{-d_A(\boldsymbol{b}_l, \boldsymbol{b}_h)^2\}}{\sum_{k \neq l} C_k \exp\{-d_A(\boldsymbol{b}_l, \boldsymbol{b}_k)^2\}}}_{\text{softmax over distances}}$$

..this formulation is based on [2]

model estimates →
$$p(y|\boldsymbol{b}_l) = \sum_h p_{lh}\ \hat{q}(y|\boldsymbol{b}_h) \quad \forall\ y, \boldsymbol{b}_l$$

optimisation →
$$\max_A \sum_l \underbrace{\hat{q}(\boldsymbol{b}_l)}_{C_l/n} \sum_y \hat{q}(y|\boldsymbol{b}_l) \log p(y|\boldsymbol{b}_l)$$

This minimises the expected, under $\hat{q}_{\boldsymbol{X}}$, KL divergence from $p_{Y|\boldsymbol{X}}$ to $\hat{q}_{Y|\boldsymbol{X}}$. We can view $p(y|\boldsymbol{b}_l)$ as a kernel estimate for $\hat{q}(y|\boldsymbol{b}_l)$, with d_A in a Gaussian kernel.



- **Top-left:** Example of a solution matrix A as a heatmap. There is effective dimensionality reduction from 4d to 2d.
- **Bottom-left:** ROC curves show the improvement of d_A over Euclidean distance for a 1NN binary classifier.
- **Right:** Projected points in the 2d solution space. Colour denotes $\hat{q}(Y{=}1|\boldsymbol{x})$, and size denotes $\hat{q}(\boldsymbol{x})$.

References

[1] *Simul8 software*. url: https://www.simul8.com.

[2] Jacob Goldberger et al. "Neighbourhood Components Analysis". In: *Advances in Neural Information Processing Systems* 17 (2004).

g.laidler1@lancaster.ac.uk