Sequential data-consistent model inversion

Timothy Rumbell
IBM Research
thrumbel@us.ibm.com

Catherine Wanjiru
IBM Research
catherine.wanjiru@ibm.com

Isaiah Onando Mulang'
IBM Research
mulang.onando@ibm.com

Stephen Obonyo
Strathmore University
sobonyo@strathmore.edu

James Kozloski
IBM Research
kozloski@usibm.com

Viatcheslav Gurev
IBM Research
vgurev@us.ibm.com

Abstract

Data-consistent model inversion problems aim to infer distributions of model parameters from distributions of experimental observations. Previous approaches to solving these problems include rejection algorithms, which are impractical for many real-world problems, and generative adversarial networks, which require a differentiable simulation. Here, we introduce a sequential sample refinement algorithm that overcomes these drawbacks. A set of parameters is iteratively refined using density ratio estimates in the model input and output domains, and parameters are resampled by training a generative implicit density estimator. We implement this novel approach using a combination of standard models from artificial intelligence and machine learning, including density estimators, binary classifiers, and diffusion models. To demonstrate the method, we show two examples from computational biology, with different levels of complexity.

1 Introduction

Data-consistent model inversion (DCMI) problems [1–3] aim to model ensembles of individuals that have variability in their observable features. In DCMI for deterministic models, a vector of latent variables $\boldsymbol{x} \in \mathbb{R}^n$ linked to every individual is related to a vector of observed features $\boldsymbol{y} \in \mathbb{R}^m$ through a known function $\boldsymbol{y} = M(\boldsymbol{x})$, and the goal is to find a distribution of latent variables $\mathcal{Q}_X$ that produces the observed target distribution $\mathcal{Q}_Y$ of $\boldsymbol{y}$ within the ensemble. In this case, the distribution of $\mathcal{Q}_Y$ is the push-forward of the distribution of latent variables $\mathcal{Q}_X$ using $\boldsymbol{y} = M(\boldsymbol{x})$. To solve a DCMI problem, a prior distribution $\mathcal{P}_X$ of $\boldsymbol{x}$ is also incorporated to address uncertainty due to non-invertibility of $\boldsymbol{y} = M(\boldsymbol{x})$. DCMI is similarly formulated for non-deterministic models, for which $\boldsymbol{y} = M(\boldsymbol{x}, \boldsymbol{\xi})$, as the problem of finding a latent variable distribution that generates a distribution of target observations in the presence of additional noise $\boldsymbol{\xi}$. It should be noted that DCMI is different from parameter inference problems that aim to find the most likely set of parameters from multiple observations of a single individual, termed 'parameter identification problems' or 'simulation-based inference', which are often solved using Bayesian inference methods, which have a different purpose and have been demonstrated empirically to solve a different inference problem than DCMI [1, 2].

Previous methods for constructing the parameter distribution 'consistent' with the experimental data include density estimation and rejection sampling algorithms [1–3], adversarial variational optimization (AVO) [4], and generative adversarial networks (GANs) [5, 6]. We based the current work on multiple previous methods that solve DCMI, with the goal of creating a powerful approach that combines the benefits of prior methods into a flexible and reliable framework that can be implemented using standard machine learning (ML) density estimators and classifiers, which we term sequential data-consistent model inversion (sDCMI).

NeurIPS 2023 Workshop on Deep Learning and Inverse Problems.

In our framework, DCMI is presented in the form of constrained optimization, as in [6]. In this form, the goal is to minimize divergence between the prior $\mathcal{P}_X$ and the distribution $\mathcal{Q}_{X_g}$ of model inputs sampled by a parametric generative model $G_\theta \in \{G_\theta(\cdot)|\theta \in \Theta\}$, given that the distribution $\mathcal{Q}_{Y_g}$ of the push-forward of $\mathcal{Q}_{X_g}$ through the model M matches the target distribution of observations $\mathcal{Q}_Y$:

$$\begin{array}{ll} \text{given} & \mathcal{P}_X,\ \mathcal{Q}_Y,\ \boldsymbol{y} = M(\boldsymbol{x}) \\ \text{minimize} & D_f(\mathcal{Q}_{X_g}||\mathcal{P}_X) \\ \text{subject to} & supp(X_g) \subseteq supp(X),\ D_f(\mathcal{Q}_{Y_g}||\mathcal{Q}_Y) = 0 \\ \text{where} & \boldsymbol{y}_g = M(\boldsymbol{x}_g) \sim \mathcal{Q}_{Y_g},\ \boldsymbol{x}_g \sim \mathcal{Q}_{X_g}. \end{array} \tag{1}$$

In (1), $D_f(\cdot||\cdot)$ is an f-divergence measure such as Jensen-Shannon (JS) divergence. In [5, 6], G_θ was defined as a generator network in a regularized GAN (termed r-GAN), with separate discriminator networks used to guide the optimization towards minimizing $D_f(\mathcal{Q}_{X_g}||\mathcal{P}_X)$ and $D_f(\mathcal{Q}_{Y_g}||\mathcal{Q}_Y)$. A weighted sum of losses from each discriminator was used to train G_θ, solving (1) with the penalty method. To encourage exploration and prevent early convergence to local minima, w_X can initially be large, and gradually reduced according to a schedule, similar to simulated annealing. However, a drawback of the r-GAN approach is that M must be differentiable to backpropagate loss from the discriminator in Y to G_θ, which is problematic as arbitrary simulators consisting of complex sets of differential equations might be used.

Another option to solve (1) is to minimize the divergence terms using a non-parametric model of $\mathcal{Q}_{X_g}$ and adapt points based on estimation of the density ratios $\frac{p_X(\boldsymbol{x})}{\hat{q}_X(\boldsymbol{x})}$ and $\frac{q_Y(\boldsymbol{y})}{\hat{q}_Y(\boldsymbol{y})}$. Here $p_X(\boldsymbol{x})$, $\hat{q}_X(\boldsymbol{x})$, $q_Y(\boldsymbol{y})$, $\hat{q}_Y(\boldsymbol{y})$ are densities of $\mathcal{P}_X$, $\mathcal{Q}_{X_g}$, $\mathcal{Q}_Y$, and $\mathcal{Q}_{Y_g}$, respectively. Previous rejection algorithm approches to DCMI have used this idea, minimizing divergence between $\mathcal{Q}_{Y_g}$ and $\mathcal{Q}_Y$ by rejecting points based on the $\frac{q_Y(\boldsymbol{y})}{\hat{q}_Y(\boldsymbol{y})}$ ratio [2]. We followed a similar approach in this work, and designed an optimization where the parameter distribution in a non-parametric model is optimized based on a series of iterations by sampling a density model of $\mathcal{Q}_{X_g}$, perturbing the samples, and re-sampling based on the density ratio between generated samples and the target densities $p_X(\boldsymbol{x})$, $q_Y(\boldsymbol{y})$ to optimize the divergence terms in (1).

2 Methods

Sequential sampling for data-consistent model inversion. The core method we introduce to solve (1) is a sequential refinement of a set of parameter points for non-parametric representation of $\mathcal{Q}_{X_g}$ (Figure 1). The initial point set is generated by sampling the prior $\mathcal{P}_X$ with density $p_X(\boldsymbol{x})$ to obtain initial samples of $\mathcal{Q}_{X_g}$. First, we train a density estimator on the points in $\mathcal{Q}_{X_g}$, and then explore the parameter space by generating 'perturbed' points in $\mathcal{Q}_{X_g}$ by sampling from the trained model. In the current implementation, a parametric and non-parametric density estimator are applied in series, generating new samples from a diffusion model, and perturbing those samples with a Gaussian kernel. This process is somewhat analogous to sample perturbation approaches from sequential Monte Carlo methods [7]. New points are added to the pool of existing points, and two rejection steps are performed on the pool, one based in X and one based in Y, minimizing the divergence terms. The number of rejected points are equal to the number of newly generated points, keeping the non-parametric model (set of samples) a constant size. To minimize the divergences, we used an adaptation of the rejection algorithm from [6], calculating the density ratio between generated samples $\mathcal{Q}_{X_g}$ ($\mathcal{Q}_{Y_g}$) and the target

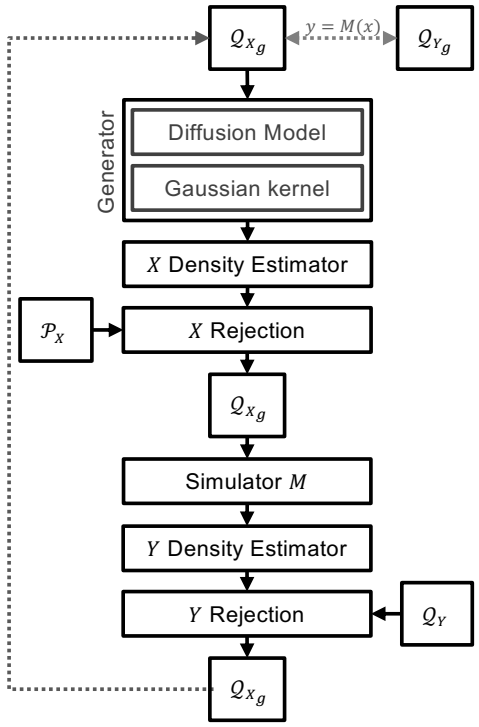

Figure 1: sDCMI sequence.

density $\mathcal{P}_X$ ($\mathcal{Q}_{Y_g}$). The flow of one sequence is shown in Figure 1, and we can iterate over the sequence until convergence.

In the non-constrained problem (1) we minimize a weighted sum of divergences, $w_X \times D_f(\mathcal{P}_X||\mathcal{Q}_{X_g}) + w_Y \times D_f(\mathcal{Q}_{Y_g}||\mathcal{Q}_Y)$. Typically, we gradually increase weight w_Y with respect to w_X, to first explore the prior, then converge to the data-consistent solution. To minimize this sum, we apply the rejection algorithm twice, first rejecting according to $\frac{p_X(\boldsymbol{x})}{\hat{q}_X(\boldsymbol{x})}$. When the prior $\mathcal{P}_X$ is assumed to be a uniform distribution over some parameter range, as is often used in the literature, this step encourages exploration in the parameter space to capture all modes of the latent distribution. Because the first rejection step only requires information in X, we can apply $\boldsymbol{y} = M(\boldsymbol{x})$ on new points after the first rejection to reduce computational costs associated with M. In the second rejection step, points are rejected based on $\frac{q_Y(\boldsymbol{y})}{\hat{q}_Y(\boldsymbol{y})}$, minimizing the divergence to render $\mathcal{Q}_{X_g}$ 'data consistent'. The rejection algorithm is presented in the Appendix, algorithm 3, and the constant B in the algorithm is calculated using a binary search (Appendix algorithm 4) to reject approximately a specified number of points N_{S_r} over the two steps. To simulate changes in ratio between weights w_X and w_Y, $N_{S_r} \times w_X$ are rejected using $\frac{p_X(\boldsymbol{x})}{\hat{q}_X(\boldsymbol{x})}$, and then the remaining $N_{S_r} \times (1 - w_X)$ samples are rejected using $\frac{q_Y(\boldsymbol{y})}{\hat{q}_Y(\boldsymbol{y})}$, with w_X gradually decreasing according to a schedule.

Boosted density estimation. The critical part of the DCMI rejection algorithm (Appendix algorithm 3) is the calculation of density ratios. For two distributions represented by their samples, direct calculation of densities is challenging, and the density ratio trick is used instead by training a binary classifier. There are multiple methods to calculate density ratios, including those that employ auxiliary distributions [8]. The context of this work is different, however, since $p_X(\boldsymbol{x})$ and $q_Y(\boldsymbol{y})$ are often given explicitly, and the challenge is then to estimate the density ratios based on direct estimation of $\hat{q}_X(\boldsymbol{x})$ and $\hat{q}_Y(\boldsymbol{x})$. Here, we use ideas from boosted density estimation [9] by training a density estimator that is then augmented with a binary classifier. First, we train a density estimator on data ($\hat{q}_X(\boldsymbol{x})$, $\hat{q}_Y(\boldsymbol{x})$). Then, we train a binary classifier between samples from the density estimator and samples from the data to calculate a density ratio between the estimator and data samples. This density ratio is then multiplied with the density from the estimator to provide a final density $\hat{q}_X(\boldsymbol{x})$, $\hat{q}_Y(\boldsymbol{x})$) to use in the calculation of $\frac{p_X(\boldsymbol{x})}{\hat{q}_X(\boldsymbol{x})}$ and $\frac{p_Y(\boldsymbol{y})}{\hat{q}_Y(\boldsymbol{y})}$ in the rejection step. Boosted density estimation requires a density estimator that allows both an explicit density calculation, and implicit samples, for which we use a normalizing flow network [10]. A simple feed forward neural network was used as a binary classifier.

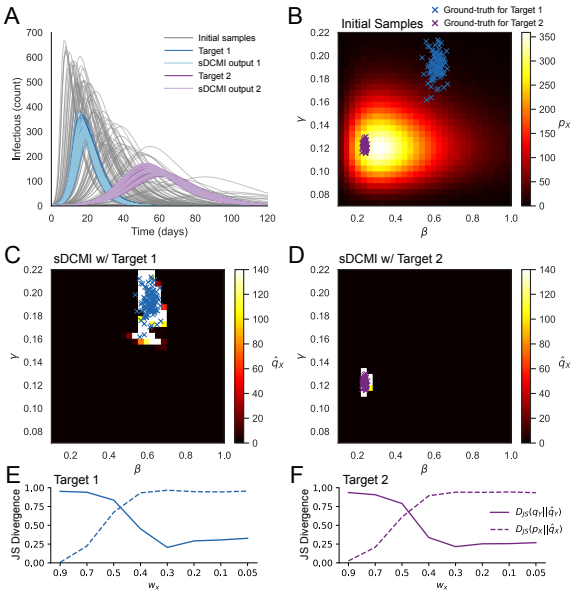

Figure 2: SIR model. (A) SIR simulation outputs (Infected count) for initial, target, and generated samples. (B) Initial sample density in parameter space, and location of ground-truth for 2 targets. (C-D) sDCMI generated parameter densities. (E-F) JS Divergence for final samples from $\hat{q}_X$ (dashed lines) and $\hat{q}_Y$ (solid lines) from sDCMI algorithm during a decreasing w_X schedule.

3 Experiments

We demonstrate sequential DCMI using 2 experiments. First, we show solutions to an ordinary differential equation (ODE) model of disease epidemiology that has been used for benchmarking simulation-based inference (SBI) [11]. Second, we use a complex ODE model of a cardiac cell contraction in a scenario where observations are recorded across multiple experimental protocols. Further details of experimental configurations are provided in the Appendix.

SIR model. This common epidemiological model simulates disease dynamics, representing the portion of a population in each of 3 states, susceptible S, infectious I, and recovered

or deceased R, according to

$$\frac{dS}{dt} = -\beta \frac{SI}{N} \quad \frac{dI}{dt} = \beta \frac{SI}{N} - \gamma I \quad \frac{dR}{dt} = \gamma I. \tag{2}$$

We used the same form of the model as in previous studies [11], inferring contact rate β and mean recovery rate γ given observations of I at 10 time points, with population size $N = 1000$. To test sDCMI with this model, 2 sets of target data were simulated (Figure 2A) by sampling β and γ according to 2 different "ground-truth" distributions (x's in Figure 2B). We ran sDCMI twice, once with each target, starting from the same distribution P_X of initial samples (density in Figure 2B). While simulations from initial samples showed a wide variety of time courses, simulations from the final Q_{X_g} closely matched the two respective targets (Figure 2A). Ground-truth parameter distributions were narrow relative to P_X (Figure 2B), but the sDCMI algorithm converged to produce final generated samples close to the ground-truth for both targets (Figure 2C-D). While reducing w_X across training, $D_{JS}(\mathcal{P}_X || \mathcal{Q}_{X_g})$ increased as w_X decreased, and $D_{JS}(\mathcal{Q}_{Y_g} || \mathcal{Q}_Y)$ decreased, converging by $w_X = 0.3$, which we therefore used for plotting the final results (Figure 2E-F).

Cardiac cell model. In some experiments, multiple experimental protocols are performed on individuals from the ensemble. In such a scenario, DCMI can include constraints from each protocol to find Q_{X_g} consistent with all available data. Sequential DCMI can leverage the iterative optimization approach to gradually incorporate new protocols and data, training Q_{X_g} first on a subset of Q_Y, then using that Q_{X_g} as the initial samples when introducing additional data. To test this scenario, we adapted a model of cardiac myocyte contraction [5], simulating an experiment measuring contraction force in response to periodic stimulation. The model produces a force transient when stimulated, and steady-state transients increase in amplitude and duration across stimulation frequencies of 1, 2, and 4Hz (Figure 3A). We sampled 102 parameter sets from a Gaussian mixture to represent a "ground-truth" distribution in parameter space (blue points in Figure 3B-C, left), and simulated the model at the 3 frequencies to represent a series of experiments conducted on each individual in the ensemble, producing a target distribution $\mathcal{Q}_Y$ (blue points in Figure 3B-C, right). The model input and output spaces in this example are 12- and 14-dimensional, respectively, when all 3 frequencies are used, so we visualize results using the first 2 principal components (PCs) from principal component analyses (PCA) on samples from $\mathcal{P}_X$ and $\mathcal{P}_Y$. The ground-truth samples are located in a relatively small region of the prior ranges and q_Y samples are in a low-density region of p_Y (Figure 3B). We used 10 iterations of the DCMI algorithm ($N_{I_g} = 10$ in Appendix algorithm (1)) at each of 5 w_X values, decreasing w_X over training, and using the final samples from sDCMI at one weight as input to sDCMI at the next weight. An outer loop over experimental protocols gradually increased the model output dimensionality, training initially with only 1Hz stimulation, then adding the other 2 frequencies one at a time, and using the final samples from one protocol as initial samples for the next protocol. Alternatively, we could have updated p_X for the next protocol with the final $\hat{q}_X$ from the previous protocol as a form of autoregressive update to the prior density. By gradually converging in this way, we were able to obtain an accurate solution to a complex DCMI problem, sampling model parameters from a broader space than the "ground-truth"

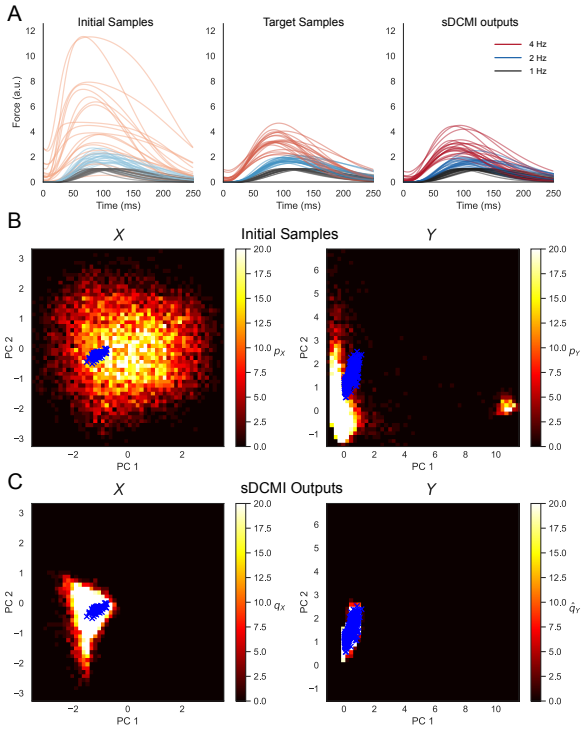

Figure 3: Cardiac cell model. (A) Force during stimulation at 1 Hz (black), 2 Hz (blue), and 4 Hz (red) frequencies. Samples from (left) initial parameters $\mathcal{P}_Y$, (middle) target outputs $\mathcal{Q}_Y$, (right) sDCMI result $\hat{\mathcal{Q}}_Y$. (B) Density of initial samples in 1st 2 PCs from PCA on initial samples in X and Y. (C) Density of sDCMI trained Generator output samples. Blue x's show "ground-truth" in X and target observations in Y.

(which would be unknown in a real-world example), while converging to an accurate fit to target data across multiple experimental protocols (Figure 3C).

4 Conclusions

We have introduced a new approach to DCMI problems using sequential refinement of a set of samples, approximated by a sequence of generative models. This approach alleviates several practical hurdles in employing DCMI for complex problems, and can be reliably and flexibly implemented using standard ML/AI methods. In the current implementation, the use of boosted density estimation enables models that are not necessarily optimal for each specific problem to be used effectively, and we used identical configuration of the individual components configurations in both examples shown here. A number of enhancements to this algorithm are possible in future work, including autoregressive prior distribution updates across multiple experiments performed on an ensemble and conditioning the generative model on covariate information. We have developed software tools for configuration and deployment of this algorithm.

References

[1] M. Pilosov, C. del Castillo-Negrete, T. Y. Yen, T. Butler, and C. Dawson, "Parameter estimation with maximal updated densities," *Computer Methods in Applied Mechanics and Engineering*, vol. 407, p. 115906, 2023.

[2] T. Butler, J. Jakeman, and T. Wildey, "Combining push-forward measures and bayes' rule to construct consistent solutions to stochastic inverse problems," *SIAM Journal on Scientific Computing*, vol. 40, no. 2, pp. A984–A1011, 2018.

[3] T. Butler, T. Wildey, and T. Y. Yen, "Data-consistent inversion for stochastic input-to-output maps," *Inverse Problems*, vol. 36, p. 085015, 2020.

[4] G. Louppe, J. Hermans, and K. Cranmer, "Adversarial variational optimization of non-differentiable simulators," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and M. Sugiyama, Eds., vol. 89. PMLR, 16–18 Apr 2019, pp. 1438–1447.

[5] J. Parikh, T. Rumbell, X. Butova, T. Myachina, J. C. Acero, S. Khamzin, O. Solovyova, J. Kozloski, A. Khokhlova, and V. Gurev, "Generative adversarial networks for construction of virtual populations of mechanistic models: simulations to study omecamtiv mecarbil action," *Journal of Pharmacokinetics and Pharmacodynamics*, vol. 49, pp. 51–64, 2022.

[6] T. Rumbell, J. Parikh, J. Kozloski, and V. Gurev, "Novel and flexible parameter estimation methods for data-consistent inversion in mechanistic modeling," *Royal Society Open Science*, vol. 10, p. 230668, 2023.

[7] A. Doucet, N. De Freitas, N. J. Gordon *et al.*, *Sequential Monte Carlo methods in practice*. Springer, 2001, vol. 1, no. 2.

[8] A. Srivastava, S. Han, K. Xu, B. Rhodes, and M. U. Gutmann, "Estimating the density ratio between distributions with high discrepancy using multinomial logistic regression," *arXiv preprint arXiv:2305.00869*, 2023.

[9] Z. Cranko and R. Nock, "Boosted density estimation remastered," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 1416–1425.

[10] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *International Conference on Learning Representations*, 2017.

[11] J.-M. Lueckmann, J. Boelts, D. Greenberg, P. Goncalves, and J. Macke, "Benchmarking simulation-based inference," in *International conference on artificial intelligence and statistics*. PMLR, 2021, pp. 343–351.

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *arXiv preprint arxiv:2006.11239*, 2020.

A Algorithms

Algorithm 1 Sequential data-consistent model inversion (sDCMI)

Require: Number of: samples to generate N_{S_g}, samples to retain N_{S_r}, generation iterations N_{I_g}, rejection iterations N_{I_r}; simulator/model M; target output density q_Y; initial parameter density p_X; parameter space rejection weight w_X

$F_{rej} \leftarrow \frac{N_{S_g}}{N_{S_g}+N_{S_r}}$ ▷ Fraction of samples to reject each iteration
$P :=$ GENERATESAMPLES($p_{\boldsymbol{X}}$, M, N_{S_r}) ▷ Initial samples
for j=1...N_{I_g} **do**
    Set $Q := \{\boldsymbol{x}_i : (\boldsymbol{x}_i, \boldsymbol{y}_i) \in P\}$
    Train Generator $\hat{q}_{\boldsymbol{X}}$ from Q ▷ Diffusion model plus Gaussian noise
    $\hat{P} :=$ GENERATESAMPLES($\hat{q}_{\boldsymbol{X}}$, M, N_{S_g})
    $P := P \cup \hat{P}$
    $F_0 \leftarrow 1.0$
    for k=1...N_{I_r} **do**
        $F \leftarrow \frac{F_0 \times F_{rej}}{N_{I_r}}$; $F_X \leftarrow F \times w_X$; $F_Y \leftarrow \frac{F \times (1-w_X)}{1-F_X}$; $F_0 \leftarrow \frac{F_0}{1-F}$ ▷ Rejection fractions
        $P \leftarrow$ REJECTION(P, $p_{\boldsymbol{X}}$, $q_{\boldsymbol{Y}}$, F_X, F_Y)
    end for
end for
return $\{\boldsymbol{x}_i : (\boldsymbol{x}_i, \boldsymbol{y}_i) \in P\}$

Algorithm 2 sDCMI subroutines - GenerateSamples

Require: Density to sample from $g_{\boldsymbol{X}}$, model M, number of samples to generate N

function GENERATESAMPLES($g_{\boldsymbol{X}}$, M, N)
    Sample $\{\boldsymbol{x}_i\}_{i=1}^{N} \sim g_{\boldsymbol{X}}$
    for i=1...N_{S_r} **do**
        $\boldsymbol{y}_i := M(\boldsymbol{x}_i)$
    end for
    $P := \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N_{S_r}}$
    return P
end function

Algorithm 3 sDCMI subroutines - Rejection

Require: Set of $\{\boldsymbol{x}, \boldsymbol{y}\}$ pairs involved in rejection P, X target density $p_{\boldsymbol{X}}$, Y target density $q_{\boldsymbol{Y}}$, target fraction to reject using X F_X, target fraction to reject using Y F_Y

function REJECTION(P, $p_{\boldsymbol{X}}$, $q_{\boldsymbol{Y}}$, F_X, F_Y)
  Set $Q := \left\{\boldsymbol{x}_i : (\boldsymbol{x}_i, \boldsymbol{y}_i) \in P\right\}$
  Train density estimator $\hat{q}_{\boldsymbol{X}}$ from Q
  $B \leftarrow$ COMPUTEB(Q, $p_{\boldsymbol{X}}$, $\hat{q}_{\boldsymbol{X}}$, F_X)
  for $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in P$ **do**
    $\lambda_i = \min\left(\frac{p_X(\boldsymbol{x}_i)}{B \times \hat{q}_X(\boldsymbol{x}_i)}, 1\right)$
    $\rho \sim \mathcal{U}([0, 1])$
    if $\rho > \lambda_i$ **then**
      Reject $P := P \setminus (\boldsymbol{x}_i, \boldsymbol{y}_i)$
    end if
  end for
  Set $Q := \left\{\boldsymbol{y}_i : (\boldsymbol{x}_i, \boldsymbol{y}_i) \in P\right\}$
  Train density estimator $\hat{q}_Y$ from Q
  $B \leftarrow$ COMPUTEB(Q, $q_{\boldsymbol{Y}}$, $\hat{q}_{\boldsymbol{Y}}$, F_Y)
  for $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in P$ **do**
    $\lambda_i = \min\left(\frac{q_Y(\boldsymbol{y}_i)}{B \times \hat{q}_Y(\boldsymbol{y}_i)}, 1\right)$
    $\rho \sim \mathcal{U}([0, 1])$
    if $\rho > \lambda_i$ **then**
      Reject $P := P \setminus (\boldsymbol{x}_i, \boldsymbol{y}_i)$
    end if
  end for
  return P
end function

Algorithm 4 sDCMI subroutines - ComputeB

Require: Set of points involved in rejection Q, target density p, generated sample density q, target fraction to reject F

function COMPUTEB(Q, p, q, F)
  $R \leftarrow []$
  for $(\boldsymbol{x}_i) \in P$ **do**
    $R := R \cup \frac{q(\boldsymbol{x}_i)}{p(\boldsymbol{x}_i)}$
  end for
  Sort R in ascending order
  $l \leftarrow 0, h \leftarrow |R| - 1$
  while $l < h$ **do** ▷ binary search for optimal B
    $m = (l + h)/2$ ▷ index of current B estimate
    $b \leftarrow R[m]$ ▷ assign current B estimate to b
    $F_b \leftarrow 0$
    for $(r_i) \in R$ **do**
      $F_b = F_b + \min\left(\frac{b}{r_i}, 1\right)$ ▷ expected contribution of r_i to number of remaining points
    end for
    $F_b = \frac{F_b}{|R|}$ ▷ convert expected number of points to fraction
    if $F_b < F$ **then** ▷ binary search
      $l = m + 1$
    else
      $h = m - 1$
    end if
  end while
  $B = R[l]\frac{F}{F_b}$ ▷ adjust B to account for difference between F and estimated F from points
  return B
end function

B AI/ML model implementations

The models used for each component of sDCMI are flexible. Here, we used neural networks for generative models, density estimators and binary classifiers to implement these components, but many alternate AI/ML approaches may work effectively. All neural networks used to produce the results in the example experiments had the same structure and properties, and were implemented in pytorch using custom code. Below we briefly detail the implementation for each component shown in Figure 1 in the main manuscript.

Diffusion model. For the generator, we trained a diffusion model [12] to match samples Q_{X_g} each sDCMI iteration. We used a sigmoid with 100 time steps to calculate the diffusion schedules, and an embedding layer for t. The neural network was a fully connected network with 8 hidden layers of 256 nodes per layer, and ReLU activation. The batch size was 512, learning rate 0.001, and we trained for 20 epochs. The input and output dimensions matched the dimension of X.

Gaussian kernel. After generating samples Q_{X_g} from the trained diffusion model, we added Gaussian noise according to $\mathcal{N}(0, 0.1)$ to each parameter. Samples were normalized based on the prior, so we applied the same variance to each dimension, to 'explore' an equivalent proportion of P_X.

Density Estimator. To train the density estimators in X and Y at lines 3 and 13 of algorithm 3, we used boosted density estimators, refining the density estimated by a neural network with the output of a binary classifier trained to discriminate the density estimator training set from the density estimator outputs.

For the neural network density estimator, we used normalizing flow networks, implemented using the 'normflows' python package (https://pypi.org/project/normflows/). Starting from a diagonal Gaussian base distribution, we used 32 layers, each comprising an 'AffineCouplingBlock', consisting of a fully connected network with 2 hidden layers of 200 nodes per layer, a dropout rate of 0.3, followed by a 'Permute' operation to swap the roles of the output 'shift' and 'scale' values for each dimension, to implement Real NVP [10]. We trained for 1 epoch with batch size 512, using all input samples as the training data.

For the binary classifier, we used a fully connected network with 8 hidden layers and 128 nodes per layer, ReLU activation, a dropout rate of 0.2, and trained using cross-entropy loss for 1 epoch with batch size 512. Batches of training data consisted of samples used to train the density estimator, labeled '1', and samples randomly generated by the trained density estimator, labeled '0'.

After training both the density estimator and the classifier, the final density for an input is computed by multiplying the output of both networks, as described in the 'boosted density estimation' section of the main manuscript.

sDCMI and Rejection parameters. The following hyperparameters were used in algorithm 1: $N_{S_g} = 65,536$; $N_{S_r} = 65,536$; $(N_{I_g} = 8; N_{I_r} = 4)$ for the SIR model example; $(N_{I_g} = 10; N_{I_r} = 5)$ for the Cardiac cell model example.

C sDCMI examples - further details

C.1 SIR model.

The SIR model experiments generally followed the example in [11]. The prior was specified as $\beta \sim$ LogNormal$(log(0.4), 0.5)$, $\gamma \sim$ LogNormal$(log(1/8), 0.2)$. Two target distributions were simulated from two "ground-truth" parameter distributions according to: ground-truth for target 1: $\beta \sim \mathcal{N}(0.615, 0.615/20)$, $\gamma \sim \mathcal{N}(0.192, 0.192/20)$; ground-truth for target 2: $\beta \sim \mathcal{N}(0.238, 0.238/40)$, $\gamma \sim \mathcal{N}(0.121, 0.121/40)$. The means of these ground-truth distributions were two of the ground-truth points used for the simulation-based inference benchmarks in [11], and we added variation to those means to represent multiple observations of disease dynamics from an ensemble, to make the problem relevant for DCMI. Initial conditions were $(S(0), I(0), R(0)) = (N - 1, 1, 0)$, where $N = 1000$ was the population size. The model was simulated for 160 time points (days), and 10 evenly spaced points from the resulting I were used as $\boldsymbol{y}$. We ran the sDCMI algorithm with 8 kernels and 4 rejection steps across iterations for 8 different w_X values, $[0.9, 0.7, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05]$. The results from $w_X = 0.3$ are shown in Figure 2, as at this point the simulation outputs visually matched the target distributions, and subsequent reductions in w_X did not reliably improve JS divergence between $\hat{q}(\boldsymbol{y})$ and $q(\boldsymbol{y})$.

C.2 Cardiac cell model.

The cardiac cell model used was simplified from [5] to reproduce only one aspect of that simulation, the typical cooperative relationship between force and calcium concentrations (i.e., significant increase in maximum force for small changes in calcium levels) upon stimulation at fixed sarcomere lengths and calcium concentration. Stimulation was effectively performed by modulating calcium concentration, inducing calcium transients at the stimulation frequency, which in turn cause a contraction force transient. The ordinary differential equations used for this simulation are provided below.

At each of 3 stimulation frequencies, 1, 2 and 4Hz, we simulated the equations for a duration of 5s, and extracted features of the final (steady-state) transients, with the unitless force values normalized to the peak value at 1Hz. The features extracted from the transients at each frequency were: duration at 5% of maximum amplitude; duration 50% of maximum amplitude; ratio of durations at 5 and 50%; quiescent period (i.e. duration between stimulation onset and 5% of maximum amplitude). Additional features calculated across multiple transients were: ratio of amplitudes at 1Hz and 2Hz; ratio of amplitudes at 1Hz and 4Hz. The full distribution of features used as optimization targets are shown in orange in Figure 5. Note that these feature targets occupy a relatively narrow range of the possible features generated by initial samples in the model, shown in main manuscript Figure 3B. The final distribution of simulated outputs generated by the final trained generator is shown in blue in Figure 5.

As described in the main manuscript, sDCMI was performed in a set of nested loops to iteratively converge to a solution. The main sDCMI algorithm (algorithm 1) performs a sequence of generative model training, sampling, and rejection steps for a given w_X value. Starting with only the 1Hz stimulation frequency, a second loop progressively reduced the w_X values across a series of sDCMI trainings through the sequence $[0.4, 0.3, 0.2, 0.15, 0.1, 0.05]$. The third loop added the 2Hz and 4Hz stimulation frequencies one at a time, and repeated the second loop of w_X reductions, but starting from a lower w_X, through the sequence $[0.3, 0.2, 0.15, 0.1, 0.05]$.

Cardiac cell model equations

Calcium equations

$$\beta = \left(\frac{\tau_1}{\tau_2}\right)^{-1/\left(\frac{\tau_1}{\tau_2}-1\right)} - \left(\frac{\tau_1}{\tau_2}\right)^{-1/\left(1-\frac{\tau_2}{\tau_1}\right)}$$

$$[Ca](t) = \left(\frac{Ca_{amp} - Ca_d}{\beta}\right) \times \left(e^{-\frac{t-t_{start}}{\tau_1}} - e^{-\frac{t-t_{stim}}{\tau_2}}\right) + Ca_d$$

Cooperativity equations

$$H(A) = \left(\frac{A^{n_A}}{A^{n_A} + {A_{50}}^{n_A}}\right)$$

$$g_{on}(A) = \bar{f}_G \times H(A)^{\zeta}$$

$$g_{off}(A) = \bar{g}_G \times min\left\{H(A)^{\zeta-1}, g_{max}\right\}$$

Binding of calcium to troponin

$$\frac{dA}{dt} = k_{\text{on}}[\text{Ca}](1 - A) - k_{\text{off}}A$$

Cross-bridge groups

$$\frac{d\text{G}_{XB}}{dt} = g_{on}(1 - \text{G}_{XB}) - g_{off}\text{G}_{XB}$$

Force calculation

$$F = \gamma \times \text{G}_{XB}$$

Cardiac cell model parameters

The default fixed parameters, and bounds of a uniform distribution used as the prior for variable parameters parameters are listed in Table 1. These were optimized in previous work [5] to match *in vitro* experiments performed in rat myocyte preparations. A total of 8 model parameters were free to vary during the sDCMI algorithm at 1Hz stimulation, and 2 additional parameters were added at each subsequent iteration, to simulate the effects of increased stimulation frequency on the magnitude of the calcium transient (driving the force transient). For each parameter set (individual), these additional parameters scaled the tau_2 and k_{on}T parameters relative to the values used in the 1Hz simulation for that parameter set. Between stimulation frequency iterations (e.g. from 1Hz, to (1Hz and 2Hz)), the final trained generator from the previous iteration was used for all parameters used in the previous iteration, and the additional parameters required for the new additional stimulation frequency simulation were sampled uniformly within the prior bounds shown in Table 1. At the final iteration when all 3 frequencies were simulated the parameter space was 12-dimensional. Figure 4 shows the full distribution of "ground-truth" samples used for those parameters in orange, which come from a relatively narrow region of the full prior, as demonstrated in main manuscript Figure 3B. Figure 4 also shows the final distribution of sDCMI output parameter samples in blue, which encompass the ground-truth region, but also spread according to the uniform prior, within the constraint that the simulated outputs should match.

Parameter	Value	Bounds	Units
Ca_{amp}	1.2	-	μM
τ_1	18.0	-	ms
Ca_d	8e-3	-	μM
ζ	-	[0.3, 0.7]	unitless
τ_2	-	[40, 120]	ms
k_{on}	-	[8e-4, 8e-2]	$\mu M^{-1}s^{-1}$
k_{off}	-	[7.5e-4, 7.5e-2]	s^{-1}
$\bar{f}_G$	-	[2e-3, 1.0]	ms^{-1}
$\bar{g}_G$	-	[1e-3, 3.5e-2]	ms^{-1}
A_{50}	-	[0.1, 0.9]	unitless
n_A	-	[7.0, 10.0]	unitless
tau_2scale, 2Hz	-	[0.5, 2.0]	unitless
k_{on}scale, 2Hz	-	[0.5, 2.0]	unitless
tau_2scale, 4Hz	-	[0.5, 2.0]	unitless
k_{on}scale, 4Hz	-	[0.5, 2.0]	unitless

Table 1: Cardiac cell model parameters.

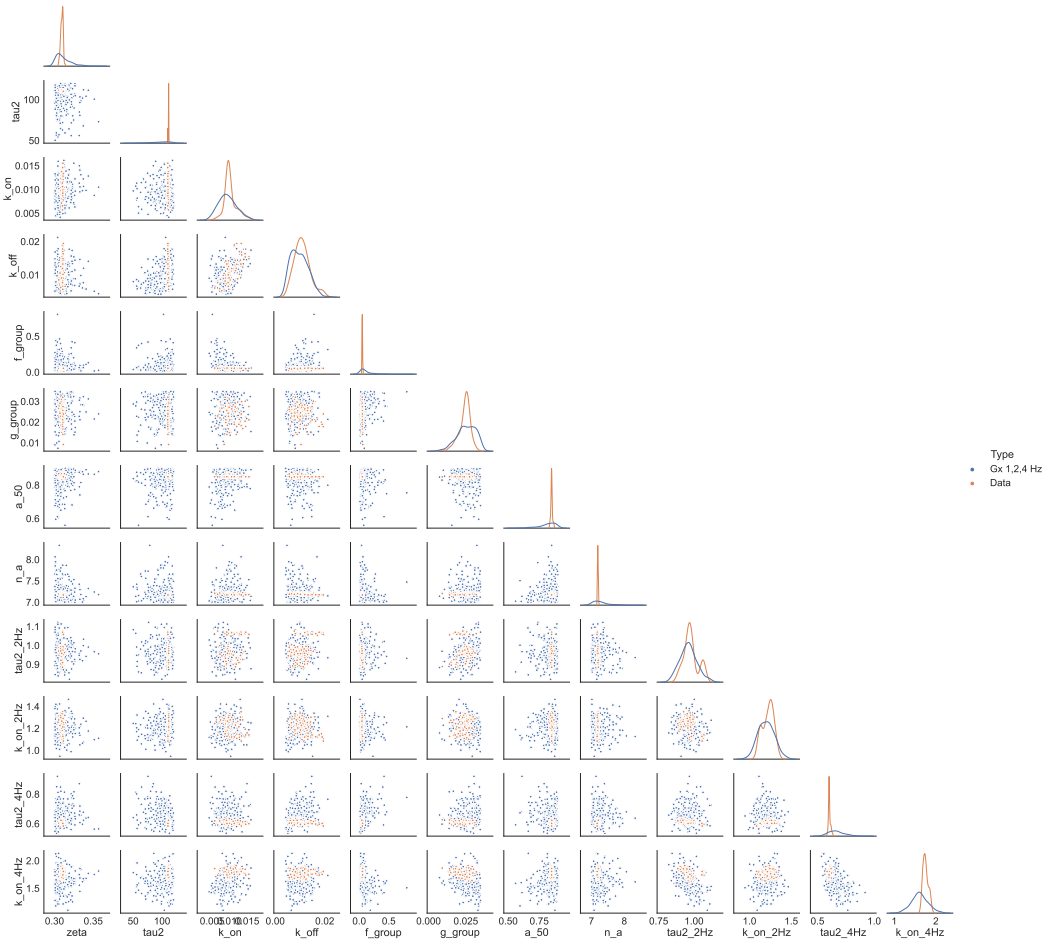

Figure 4: Cardiac model X (parameter) space samples. $1,000$ samples from the ground-truth distribution (orange) sampled from a mixture of 3 Gaussians, and from the trained generator G after sDCMI (blue).

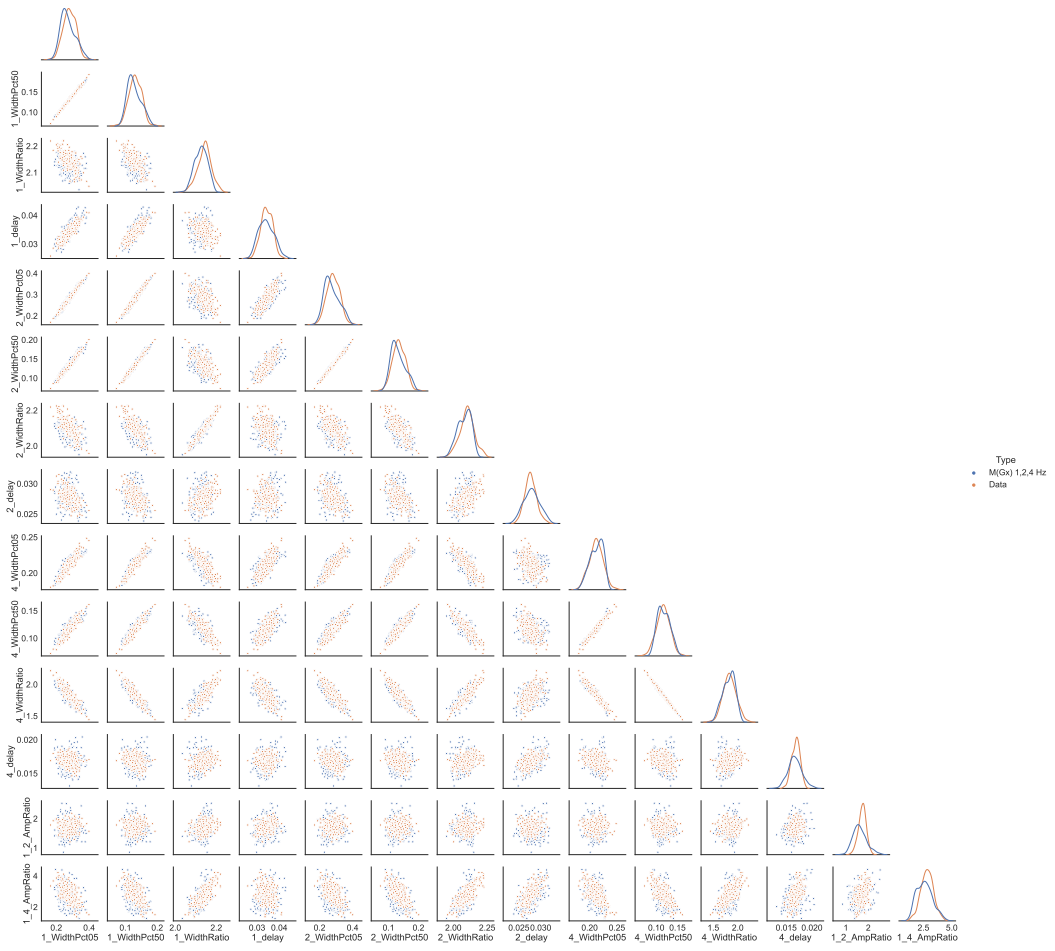

Figure 5: Cardiac model Y (feature) space samples. $1,000$ samples from the target distribution (orange), and from the sDCMI output (blue), both simulated from the orange and blue points in Figure 4.