On the connection between Noise-Contrastive Estimation and Contrastive Divergence

Citation for the original published paper (version of record):
Olmin, A., Lindqvist, J., Svensson, L. et al (2024). On the connection between Noise-Contrastive Estimation and Contrastive Divergence. Proceedings of Machine Learning Research, 238: 3016-3024

(article starts on next page)

On the connection between Noise-Contrastive Estimation and Contrastive Divergence

Amanda Olmin[⋆,1] **Jakob Lindqvist**[⋆,2] **Lennart Svensson**[2] **Fredrik Lindsten**[1]
Linköping University[1] Chalmers University of Technology[2] [⋆]Equal contribution

Abstract

Noise-contrastive estimation (NCE) is a popular method for estimating unnormalised probabilistic models, such as energy-based models, which are effective for modelling complex data distributions. Unlike classical maximum likelihood (ML) estimation that relies on importance sampling (resulting in ML-IS) or MCMC (resulting in contrastive divergence, CD), NCE uses a proxy criterion to avoid the need for evaluating an often intractable normalisation constant. Despite apparent conceptual differences, we show that two NCE criteria, ranking NCE (RNCE) and conditional NCE (CNCE), can be viewed as ML estimation methods. Specifically, RNCE is equivalent to ML estimation combined with conditional importance sampling, and both RNCE and CNCE are special cases of CD. These findings bridge the gap between the two method classes and allow us to apply techniques from the ML-IS and CD literature to NCE, offering several advantageous extensions.

1 INTRODUCTION

Unnormalised probabilistic models, such as energy-based models (LeCun et al., 2006; Gustafsson et al., 2020; Gao et al., 2020; Du et al., 2021; Florence et al., 2022), products of experts (Hinton, 2002) and Markov random fields (Köster et al., 2009), can be used for modelling complex data distributions by trading exact normalisation for flexibility. Estimating unnormalised models is however not straightforward since maximum likelihood (ML) estimation involves the typically intractable normalisation constant.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

One way to handle this challenge is to estimate the normalisation constant using importance sampling (IS), resulting in a learning algorithm denoted ML-IS. In gradient-based learning, an alternative to ML-IS is contrastive divergence (CD) (Hinton, 2002), where Markov chain Monte Carlo (MCMC) sampling is used to approximate the gradient of the log-normalisation constant.

A different solution to handling an intractable normalisation constant is to reformulate the model estimation as a binary classification problem, as done in noise-contrastive estimation (NCE) (Gutmann and Hyvärinen, 2012). In NCE, the model implicitly learns the data distribution by learning to distinguish between true samples and samples from a *noise* distribution.

Several extensions of NCE have been proposed: mainly ranking NCE (RNCE), which is a multi-class version of its predecessor (Jozefowicz et al., 2016), and conditional NCE (CNCE), where the noise distribution is conditioned on the data (Ceylan and Gutmann, 2018). RNCE in turn, has been extended into new estimation methods (Gustafsson et al., 2020; Ma and Collins, 2018). In Gao et al. (2020) a version of NCE is proposed, where the data and noise distributions are jointly learned.

Evidently, there is a plethora of methods for estimating unnormalised models, some of which seem conceptually different. We hence argue for a need to create a more coherent framework. To contribute to this objective, we provide a direct relationship between NCE and ML-IS as well as CD. We believe that this link makes it easier to understand and analyse the methods, and brings additional theoretical insights apart from what has previously been established (Ma and Collins, 2018; Ceylan and Gutmann, 2018). Specifically, we strengthen the connection between NCE, ML-IS and CD by:

- clarifying the connection between RNCE and standard IS by showing that RNCE can be derived through an extension of IS, referred to as conditional IS (CIS) (Andrieu et al., 2010; Naesseth et al., 2019);
- showing that both CNCE and RNCE are special

cases of CD, with kernels based on CIS and a variant of the Metropolis–Hastings algorithm, respectively.

Previous work has connected the original NCE criterion to general (statistical) frameworks based on Bregman divergences (Gutmann and Hirayama, 2011; Uehara et al., 2020), but to the best of our knowledge, RNCE and CNCE have not been connected to such frameworks. RNCE has previously been linked to IS (Jozefowicz et al., 2016), but in an informal way and without making the connection to CIS, which is necessary for the equivalence to hold. Moreover, another common method for training unnormalised models, namely, score matching, is a limiting case of CNCE (Ceylan and Gutmann, 2018). This is especially interesting as also CD has links to score matching (Hyvärinen, 2007).

Closest to our contribution is the work by Yair and Michaeli (2021). They focus on CNCE and show that CD is recovered by selecting the noise distribution in CNCE as an MCMC kernel. We go in the other direction and show that CNCE is in fact a special case of CD *for any noise distribution*. Importantly, this interpretation holds in the general case, without the need of re-weighting the CD gradient, as done in (Yair and Michaeli, 2021). Furthermore, for multi-step sampling we can rely on standard MCMC theory and do not need to introduce the "time-reversal adversarial game" as proposed by Yair and Michaeli (2021). Finally, we consider also RNCE, which they do not.

Based on the established connections, we use techniques from existing literature on ML-IS and CD to (i) theoretically justify why RNCE empirically performs better than ML-IS (Gustafsson et al., 2020), (ii) motivate why, for optimal learning, the noise distribution should resemble the model distribution, and not the data distribution as proposed previously (Gutmann and Hyvärinen, 2012) and (iii) identify several extensions to RNCE and CNCE and show empirically that these improve performance, with little or no increase in computational cost. We hope that these connections can lead to more valuable insights, apart from those identified in this paper.

2 BACKGROUND

Given i.i.d. training data $\{\mathbf{x}_0^i\}_{i=1}^N$ from some unknown data distribution $p_d(\cdot)$, we seek to approximate $p_d(\cdot)$ with a parametric model

$$p_\theta(\mathbf{x}) = \tilde{p}_\theta(\mathbf{x})/Z_\theta, \quad Z_\theta = \int \tilde{p}_\theta(\mathbf{x}')\mathrm{d}\mathbf{x}', \tag{1}$$

where $\tilde{p}_\theta$ is the unnormalised model and Z_θ is the normalisation constant. We assume that $p_\theta \gg p_d$, meaning that p_θ covers the support of p_d, such that $p_\theta(\mathbf{x}) > 0$ whenever $p_d(\mathbf{x}) > 0$. The model is estimated by minimising some criterion $\mathcal{L}(\theta)$ with respect to (w.r.t.) the parameter vector θ. For the ML estimator, the criterion is the negative log-likelihood (NLL) of the model

$$\mathcal{L}(\theta; \mathbf{x}_0) = -\log p_\theta(\mathbf{x}_0) = -\log \tilde{p}_\theta(\mathbf{x}_0) + \log Z_\theta. \tag{2}$$

We use $\mathbf{x}_0$ to denote a sample from p_d. In practice, $\mathcal{L}(\theta; \mathbf{x}_0)$ is computed as an average over N independent samples $\mathbf{x}_0^i \sim p_d(\cdot)$ but for brevity we perform all derivations for a single data point $\mathbf{x}_0$.

The first term in Eq. (2) is normally easy to evaluate but the second term involves the typically intractable integral in Eq. (1). Below, we introduce common methods for handling this. Derivations and proofs that are omitted from the main article can be found in the supplementary material.

2.1 Importance sampling

Using an importance sampling (IS) estimate of Z_θ in Eq. (2) results in an approximate ML criterion abbreviated ML-IS. Assume that we have defined a proposal distribution q, such that $q \gg p_\theta$. In IS, we draw J i.i.d. samples $\mathbf{x}_{1:J} = [\mathbf{x}_1, \dots, \mathbf{x}_J]$ from q; we use $q(\mathbf{x}_{1:J}) \coloneqq \prod_{j=1}^J q(\mathbf{x}_j)$ to denote their joint distribution. Given $\mathbf{x}_{1:J}$, we approximate the normalisation as

$$Z_\theta \approx \hat{Z}_\theta^{\mathrm{IS}} \coloneqq \frac{1}{J}\sum_{j=1}^J \tilde{w}_\theta(\mathbf{x}_j), \text{ with } \tilde{w}_\theta(\mathbf{x}_j) \coloneqq \frac{\tilde{p}_\theta(\mathbf{x}_j)}{q(\mathbf{x}_j)}. \tag{3}$$

The estimate $\hat{Z}_\theta^{\mathrm{IS}}$ is unbiased, see e.g. (Naesseth et al., 2019), but the gradient $\nabla_\theta \log \hat{Z}_\theta^{\mathrm{IS}}$ is not (Robert et al., 1999). Meanwhile, having an unbiased gradient estimate is undoubtedly advantageous. It is, for example, a standard condition for proving general convergence of stochastic gradient descent (SGD) (Bottou et al., 2018).

2.2 Contrastive divergence

Instead of approximating the NLL in Eq. (2), we can estimate its gradients using the identity (LeCun et al., 2006)

$$-\nabla_\theta \log p_\theta(\mathbf{x}_0) = \\ -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}')\right]. \tag{4}$$

Again, this gradient is intractable since it involves an expected value w.r.t. p_θ, but it can be approximated with MCMC methods. Contrastive divergence (CD) (Hinton, 2002), uses the approximation

$$-\nabla_\theta \log p_\theta(\mathbf{x}_0) \approx \\ -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \mathbb{E}_{K_\theta(\mathbf{x}'|\mathbf{x}_0)}\left[\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}')\right], \tag{5}$$

where $K_\theta(\mathbf{x}' \mid \mathbf{x}_0)$ is a p_θ-invariant MCMC-kernel initialised at a data sample $\mathbf{x}_0$.

For efficiency, CD runs the MCMC chain for only a few steps or even a single step (CD-1). This is typically not enough to obtain a sample from the target distribution p_θ. However, initialising the chain with a sample from the data distribution is a means of initialising it close to the modes of p_θ, especially as p_θ converges to p_d. Empirical evidence suggests that the bias of the CD parameter estimate, in relation to the ML estimate, is small (Carreira-Perpinan and Hinton, 2005).

CD was originally based on a different criterion than the log-likelihood (Hinton, 2002). However, it is common to derive CD as done here, see, e.g., (Welling et al., 2003; Asuncion et al., 2010), and the resulting gradient expression is the same.

2.3 Noise-contrastive estimation

Noise-contrastive estimation (NCE) avoids computing the NLL in Eq. (2) altogether, by transforming the model estimation into a task of discriminating between true samples from p_d and samples from a noise distribution (Gutmann and Hyvärinen, 2012; Pihlaja et al., 2010). For later comparison, we interpret the noise distribution as a proposal and denote it with q. We focus on two extensions of the original NCE method: RNCE (Jozefowicz et al., 2016) and CNCE (Ceylan and Gutmann, 2018). The first is consistent under weaker assumptions than standard NCE (Ma and Collins, 2018), while the other improves upon the original formulation by conditioning the noise distribution on the data. Furthermore, both eliminate the need to include the normalisation constant as an extra model parameter.

Ranking NCE (RNCE), extends the binary classification problem of original NCE to a multi-class classification problem (Jozefowicz et al., 2016). Consider a scenario with one data sample $\mathbf{x}_0 \sim p_d(\cdot)$ and J i.i.d. noisy samples $\mathbf{x}_{1:J} \sim q(\cdot)$. As in ordinary NCE, we train the model to classify $\mathbf{x}_0$ as the sample coming from p_d. Specifically, RNCE maximises the posterior distribution $p(z = 0 \mid \mathbf{x}_{0:J})$, where $z \in \{0, 1, \dots, J\}$ is a latent categorical variable corresponding to the index, or class, of the data point drawn from p_d. When calculating the posterior, the unknown data distribution p_d is replaced with p_θ, forcing p_θ to approach p_d. With a uniform prior on z, the empirical loss for one data point $\mathbf{x}_0 \sim p_d(\cdot)$ becomes

$$\mathcal{L}_{\mathrm{R}}(\theta; \mathbf{x}_{0:J}) = -\log \tilde{w}_\theta(\mathbf{x}_0) + \log\left(\sum_{j=0}^{J} \tilde{w}_\theta(\mathbf{x}_j)\right), \quad (6)$$

where we re-write the criterion with the weights from Eq. (3); see Appendix A.2 for the derivation.

Another extension is Conditional NCE (CNCE) (Ceylan and Gutmann, 2018) which allows the noise distribution to depend on the data sample, resulting in more difficult discrimination and thereby better model estimation. CNCE, like standard NCE, considers a binary classification problem, where a data point $\mathbf{x}_0 \sim p_d(\cdot)$ is discriminated from a sample $\mathbf{x}_1 \sim q(\cdot \mid \mathbf{x}_0)$.

Following Ceylan and Gutmann (2018), we average the posterior of the latent (Bernoulli) class variable $z \in \{0, 1\}$ over J i.i.d. samples $\mathbf{x}_j \sim q(\cdot \mid \mathbf{x}_0)$, to reduce the variance of the estimate, and minimise

$$\mathcal{L}_{\mathrm{C}}(\theta; \mathbf{x}_{0:J}) = \frac{1}{J}\sum_{j=1}^{J} \log\left(1 + \frac{\tilde{w}_\theta(\mathbf{x}_j \mid \mathbf{x}_0)}{\tilde{w}_\theta(\mathbf{x}_0 \mid \mathbf{x}_j)}\right). \quad (7)$$

See Appendix A.6 for the derivation. The weights from Eq. (3) are modified as

$$\tilde{w}_\theta(\mathbf{x}_\ell \mid \mathbf{x}_k) = \frac{\tilde{p}_\theta(\mathbf{x}_\ell)}{q(\mathbf{x}_\ell \mid \mathbf{x}_k)}, \quad \ell, k \in \{0, \dots, J\}. \quad (8)$$

For brevity, we omit the dependency on $\mathbf{x}_0, \mathbf{x}_{1:J}$ in the criteria from here on.

3 IMPORTANCE SAMPLING AND RNCE

In this section, we explore the connection between RNCE and ML-IS. Conceptually, ML-IS and RNCE are two distinct methods for estimating unnormalised models, but their gradient updates are very similar. Indeed, Jozefowicz et al. (2016) claim that with the RNCE criterion they "derive a similar surrogate classification task akin to NCE which arrives at IS". We show here that this statement is not entirely accurate as there is a slight, but important, difference.

The gradient of the RNCE criterion (Eq. (6)) is

$$\nabla_\theta \mathcal{L}_{\mathrm{R}}(\theta) = -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \sum_{j=0}^{J} w_j \nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j),$$
$$w_j = \frac{\tilde{w}_\theta(\mathbf{x}_j)}{\sum_{\ell=0}^{J} \tilde{w}_\theta(\mathbf{x}_\ell)}, \quad (9)$$

where w_j is the weight of the jth sample, normalised over all samples $\mathbf{x}_{0:J}$. There is a subtle difference between this gradient and the ML-IS gradient. For RNCE, the data sample $\mathbf{x}_0 \sim p_d(\cdot)$ is included in the sum in the second term. However, for ML-IS, the second term, corresponding to the estimate of $\nabla_\theta \log Z_\theta$, only includes samples from the proposal distribution q.

Instead, we show that RNCE corresponds to an ML criterion based on conditional importance sampling (CIS). CIS is a special case of Particle MCMC (Andrieu

et al., 2010; Naesseth et al., 2019). It is almost identical to standard IS, except that we condition on a sample $\mathbf{x}_0$. For our task, the CIS estimator is defined as

$$Z_\theta \approx \hat{Z}_\theta^{\text{CIS}} := \frac{1}{J+1} \sum_{j=0}^{J} \tilde{w}_\theta(\mathbf{x}_j). \tag{10}$$

where $\mathbf{x}_{1:J} \sim q(\cdot)$ and $\mathbf{x}_0$ is the conditioned sample.

Assuming that $\mathbf{x}_0 \sim p_d(\cdot)$, we can derive the RNCE criterion directly from CIS.

Proposition 3.1 (RNCE is ML-CIS)**.** *RNCE is equivalent to ML estimation using CIS, conditioning on $\mathbf{x}_0 \sim p_d(\cdot)$, for estimating the normalisation constant in Eq.* (2)*.*

See Appendix A.3 for the proof.

This link between RNCE and ML-CIS unifies these seemingly different methods of estimation, offering us a deeper understanding of RNCE and allowing us to reason about the (relative) empirical performance of this method. Interestingly, in the special case where $p_\theta = p_d$, such that $\mathbf{x}_0$ is a sample from the model distribution, the following holds:

Proposition 3.2 (Unbiased CIS estimate of $\nabla_\theta \log Z_\theta$)**.** *Assume $\mathbf{x}_0 \sim p_\theta(\cdot)$, $\mathbf{x}_{1:J} \sim q(\cdot)$ and that q is independent of θ. Then, the CIS estimator gives an unbiased estimate of the gradient of the log-normalisation constant*

$$\mathbb{E}_{p_\theta(\mathbf{x}_0), q(\mathbf{x}_{1:J})} \left[\nabla_\theta \log \hat{Z}_\theta^{CIS} \right] = \nabla_\theta \log Z_\theta. \tag{11}$$

See Appendix A.4 for the proof.

Note that, in practice, the conditions of Proposition 3.2 are not strictly fulfilled for RNCE, since it uses the "approximation" $\mathbf{x}_0 \sim p_d(\cdot)$. However, as p_θ aims to approximate p_d, we hope that $p_\theta \approx p_d$, at least during the later stages of training. If the data distribution is a good substitute for the model distribution, we then obtain "approximately unbiased" estimates of $\nabla_\theta \log Z_\theta$. By conditioning on the extra sample $\mathbf{x}_0$ we therefore get an advantage over IS, which does not give unbiased gradient estimates (Robert et al., 1999). We hypothesise that this property of CIS can help explain why RNCE appears to give better gradient estimates, compared to ML-IS. This is further motivated by the fact that the bias of RNCE will decrease as p_θ converges to p_d while the bias of IS will not.

4 CONNECTING NCE WITH CD

In this section, we connect RNCE and CNCE to the family of CD methods, and specifically CD-1, i.e., CD where the expectation in Eq. (5) is approximated with a single MCMC step. It has been shown that CD-1 is a special case of CNCE if q is an MCMC kernel fulfilling detailed balance (Yair and Michaeli, 2021). Here, we show the reverse: that not only CNCE but also RNCE are special cases of CD-1. The idea is to construct p_θ-invariant kernels, such that the gradient estimates of the resulting CD-1 variants are equivalent to those of RNCE and CNCE respectively. We note that the assumptions that we make are standard in the NCE literature, and therefore that the equivalences hold whenever the NCE methods are applicable.

4.1 RNCE criterion

To show that RNCE is a CD-1 method, we introduce an MCMC kernel $K_\theta^{\text{R}}(\mathbf{x}'|\mathbf{x}_0)$ based on CIS. Algorithm 1 shows how to generate a sample from this kernel conditioned on the state $\mathbf{x}_0$. We note that CIS was initially introduced as an MCMC procedure and that the kernel $K_\theta^{\text{R}}(\mathbf{x}'|\mathbf{x}_0)$ is known to be p_θ-invariant, see e.g. (Andrieu et al., 2010; Naesseth et al., 2019). It thus fits into the CD framework.

Using $K_\theta^{\text{R}}(\mathbf{x}'|\mathbf{x}_0)$ in CD-1 to estimate the expectation in Eq. (5), we can exactly recover the gradient of the RNCE criterion in Eq. (9). This connection is formalised in Proposition 4.1.

Proposition 4.1 (RNCE = CD-1)**.** *Model estimation with the RNCE criterion (Eq.* (6)*) is a special case of CD-1, using the MCMC kernel $K_\theta^{\text{R}}(\mathbf{x}'|\mathbf{x}_0)$ defined in Algorithm 1 if, when evaluating the expected value in Eq.* (5)*, the variable z used in Algorithm 1 is marginalised out.*

See Appendix A.5 for the proof.

4.2 CNCE criterion

Next, we establish a connection between CNCE and CD. First, the gradient of the CNCE criterion in Eq. (7) can be written as (see Appendix A.7)

$$\nabla_\theta \mathcal{L}_{\text{C}}(\theta) = -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \frac{1}{J} \sum_{j=1}^{J} \Big((1 - w_{j|0}) \cdot \nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + w_{j|0} \nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j) \Big), \tag{12}$$

$$w_{j|0} = \frac{\tilde{w}_\theta(\mathbf{x}_j \mid \mathbf{x}_0)}{\tilde{w}_\theta(\mathbf{x}_j \mid \mathbf{x}_0) + \tilde{w}_\theta(\mathbf{x}_0 \mid \mathbf{x}_j)}. \tag{13}$$

With the CD framework, we can derive the CNCE criterion by formulating a kernel $K_\theta^{\text{C}}(\mathbf{x}' \mid \mathbf{x}_0)$, conditioned on $\mathbf{x}_0$, according to Algorithm 2. The kernel is similar to the Metropolis–Hastings algorithm (Metropolis et al., 1953; Hastings, 1970), but uses another acceptance probability, which was also considered by Hastings (1970). For a symmetric proposal distribution,

Algorithm 1 CIS kernel

Input: $\mathbf{x}_0$

1. Sample $\mathbf{x}_{1:J} \sim q(\cdot)$
2. Calculate weights w_j, $j = 0, \dots, J$, using Eqs. (3) and (9)
3. Sample $z \sim \text{Categorical}\left([w_0, \dots, w_J]\right)$
4. Return $\mathbf{x}' = \mathbf{x}_z$

Algorithm 2 CNCE kernel

Input: $\mathbf{x}_0$

1. Sample $\mathbf{x}_1 \sim q(\cdot|\mathbf{x}_0)$
2. Calculate the weight $w_{1|0}$, using Eqs. (8) and (13)
3. Sample $z \sim \text{Bernoulli}\left(w_{1|0}\right)$
4. Return $\mathbf{x}' = \mathbf{x}_z$

i.e. $q(\mathbf{x}_1|\mathbf{x}_0) = q(\mathbf{x}_0|\mathbf{x}_1)$, it reduces to Barker's method (Barker, 1965). The kernel is p_θ-invariant as it fulfils the detailed balance condition, see (Hastings, 1970).

The main result concerning CNCE and its connection to CD, is given by Proposition 4.2.

Proposition 4.2 (CNCE = CD-1)**.** *Model estimation with the CNCE criterion (Eq.* (6)*) is a special case of CD-1, using the MCMC kernel defined in Algorithm 2 if, when estimating the expected value in Eq.* (5)*: (i) an average is taken over J independent samples $\mathbf{x}_j \sim K_\theta^{\text{C}}(\cdot|\mathbf{x}_0)$, and (ii) the variable z used in Algorithm 2 is marginalised out for each sample.*

See Appendix A.8 for the proof.

In contrast to Yair and Michaeli (2021), where CD-1 is derived from CNCE using the specific choice of q as the CD kernel, we derive CNCE from CD-1, for any choice of q. While Yair and Michaeli (2021) do consider general q, it is viewed as an extension of CD-1, where the gradient is by design re-weighted to match that of the CNCE criterion. In our derivation, this re-weighting instead follows naturally from a Rao-Blackwellisation of the MCMC kernel, i.e. a marginalisation of the latent variable z, and no additional weighting is required to recover the CNCE gradient.

In both Algorithms 1 and 2, a variable z is used to select the next sample in the Markov chain. However, if we only take a single step of the kernel, as in CD-1, then we can marginalise over z when computing the expected value in Eq. (5). Furthermore, for the CNCE connection, we assume that we average over J independent draws from the underlying MCMC kernel. These measures, which are necessary for an exact equivalence between the two NCE criteria and CD-1, are standard variance reduction techniques for MC estimators.

5 INSIGHTS FROM THE CD CONNECTION

With the connection between CD and NCE outlined in Propositions 4.1 and 4.2, we can apply extensions of CD to NCE to improve the performance of the latter. Apart from the examples described in this section, an obvious extension is that of taking multiple MCMC steps in the kernel, which we leave to Appendix B.

5.1 Choice of proposal distribution q

For NCE, the proposal distribution q is an important design choice. In the original interpretation as a proxy-classification problem, an intuitive and common choice is to construct a hard classification problem by choosing q as similar to the data distribution p_d as possible (Ceylan and Gutmann, 2018; Xu, 2022; Gao et al., 2020; Gustafsson et al., 2020). A choice that has also been theoretically motivated (Gutmann and Hyvärinen, 2012).

Our interpretation of both RNCE and CNCE as special cases of CD-1, instead suggests that we should choose q as close as possible to p_θ. Indeed, the proposal distribution is used to construct the kernel meant for estimating $\nabla_\theta \log Z_\theta$ in Eq. (4), and this kernel has p_θ as its stationary distribution. Setting q as an approximation to p_θ has been proposed before (Goodfellow, 2015; Xu, 2022) and recent work has shown empirically and in some limit cases that p_d is not the optimal proposal distribution (Chehab et al., 2022). These results however, apply only to standard NCE and the current literature remains inconclusive, where setting q close to p_d is still a common choice (Gutmann and Hyvärinen, 2012; Ceylan and Gutmann, 2018). With the connection between NCE and CD we provide another motivation in favour of the model distribution, by showing that setting $q = p_\theta$ gives unbiased estimates of the gradient in Eq. (4), up to a constant scaling, for both RNCE and CNCE:

Proposition 5.1 (Gradient estimate for RNCE with $q = p_\theta$)**.** *If $q = p_\theta$, then the expected gradient of the RNCE criterion $\nabla_\theta \mathcal{L}_R(\theta)$ in Eq.* (9) *is*

$$\mathbb{E}_{q(\mathbf{x}_{1:J})}\left[\nabla_\theta \mathcal{L}_R(\theta)\right] = \frac{J}{J+1} \nabla_\theta(-\log p_\theta(\mathbf{x}_0)). \quad (14)$$

Proposition 5.2 (Gradient estimate for CNCE with $q = p_\theta$)**.** *If $q(\cdot \mid \mathbf{x}_0) = p_\theta(\cdot)$, independent of $\mathbf{x}_0$, then the expected gradient of the CNCE criterion $\nabla_\theta \mathcal{L}_C(\theta)$*

in Eq. (12) *is:*

$$\mathbb{E}_{q(\mathbf{x}_{1:J}|\mathbf{x}_0)}\left[\nabla_\theta \mathcal{L}_C(\theta)\right] = \frac{1}{2}\nabla_\theta(-\log p_\theta(\mathbf{x}_0)). \quad (15)$$

See Appendix A.9 for the proofs.

For RNCE, while Proposition 3.2 holds for any q, Lemma 5.1 is stronger in the sense that it indicates that there is an idealised case, i.e., $q = p_\theta$, for which RNCE gives unbiased gradient estimates also when $\mathbf{x}_0 \sim p_d(\cdot)$. In contrast, $q = p_d$ is not guaranteed to cover the support of p_θ, in which case the requirements of Proposition 3.2 are not fulfilled. Of course, neither the data nor model distribution is available for us to evaluate in practice, but it nevertheless gives a guideline for selecting q.

Here we consider a method akin to Markovian Score Climbing (Naesseth et al., 2020) for learning a parameterised proposal q_φ jointly with p_θ. With the aim to make q_φ resemble p_θ, we propose to minimise the KL divergence between the two distributions, which is equivalent to minimising the cross-entropy:

$$\begin{aligned}\operatorname*{argmin}_\varphi \mathrm{KL}\left[p_\theta \| q_\varphi\right] &= \operatorname*{argmin}_\varphi \mathbb{E}_{p_\theta(\mathbf{x}')}\left[-\log q_\varphi(\mathbf{x}')\right] \\ &=: \operatorname*{argmin}_\varphi \mathcal{L}(\varphi). \end{aligned} \quad (16)$$

Note that we use the divergence from p_θ to q_φ, since we require q_φ to cover the support of p_θ. The expectation w.r.t. p_θ is intractable, but we already have a method to sample from this distribution: $K_\theta(\mathbf{x}'|\mathbf{x}_0)$. For example, we can estimate the gradient with the CIS kernel defined in Algorithm 1:

$$\begin{aligned}\nabla_\varphi \mathcal{L}(\varphi) &\approx \mathbb{E}_{K_\theta^{\mathrm{R}}(\mathbf{x}'|\mathbf{x}_0)}\left[-\nabla_\varphi \log q_\varphi(\mathbf{x}')\right] \\ &\approx -\sum_{j=0}^{J} w_j \nabla_\varphi \log q_\varphi(\mathbf{x}_j) =: \nabla_\varphi \hat{\mathcal{L}}(\varphi). \end{aligned} \quad (17)$$

Therefore, the model p_θ and the proposal q_φ can be estimated simultaneously, using samples from the same kernel K_θ^{R}.

As in Proposition 3.2, this estimate is unbiased under the idealised assumption that $\mathbf{x}_0 \sim p_\theta(\cdot)$.

Proposition 5.3 (Unbiased CIS estimate of $\nabla_\varphi \mathcal{L}(\varphi)$)**.** *If $\mathbf{x}_0 \sim p_\theta(\cdot)$, then the CIS estimator gives an unbiased estimate of the gradient $\mathbb{E}_{p_\theta(\mathbf{x}_0), q_\varphi(\mathbf{x}_{1:J})}\left[\nabla_\varphi \hat{\mathcal{L}}(\varphi)\right] = \nabla_\varphi \mathcal{L}(\varphi)$.*

See Appendix A.10 for the proof.

Adapting q_φ towards p_θ has been proposed before, especially in the field of Adaptive IS (Bugallo et al., 2017). It has also been used for NCE; Xu (2022) proposed it, but as a means of achieving $q_\varphi \approx p_d$, and Gustafsson et al. (2022) motivates $q_\varphi \approx p_\theta$ when estimating p_θ with ML-IS and then also use this proposal for RNCE. Our connection to CD provides a theoretical argument for why this is a good design choice.

5.2 Persistent NCE

Persistent Contrastive Divergence (PCD) is an extension of CD, with a modified kernel-based sampling method (Tieleman, 2008). Instead of re-initialising the MCMC chain based on a sample $\mathbf{x}_0 \sim p_d(\cdot)$ at every training iteration, PCD initialises the chain at iteration t using the sampled output at the previous iteration, $t-1$. Only at the start is the chain initialised with an actual data sample. The motivation is that this will improve convergence over standard CD, as the samples from the kernel will lie closer to the model distribution.

For persistent RNCE and CNCE, we update the Markov chain at iteration t by sampling an actual index z as in Algorithm 1 or 2. At iteration t, we estimate the gradient using $K_\theta(\mathbf{x}' \mid \mathbf{x}_0^{(t)})$ in place of $K_\theta(\mathbf{x}' \mid \mathbf{x}_0)$ in Eq. (5), where $\mathbf{x}_0^{(t)} := \mathbf{x}_z^{(t-1)}$ is a sample from the kernel in the previous iteration, $t-1$. Note that, while we sample z to update the Markov chain, we still marginalise over this latent variable when evaluating the expectation in Eq. (5). Similarly to Tieleman (2008), when training with SGD, we keep track of one continuing chain for each training data point in a batch. For CNCE, this translates to running J chains per data point in parallel.

5.3 MH variant of CNCE

The kernel used in the CD formulation of the CNCE criterion has similarities with the Metropolis–Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970), but with a non-standard acceptance probability. In the context of MCMC, and specifically the class of methods proposed by Hastings (1970), the MH acceptance probability is optimal in terms of Peskun ordering (Peskun, 1973).

With this in mind, we might expect that the MH acceptance probability will improve performance also in CNCE. Hence, we consider CD with the kernel given in Algorithm 2, but with the acceptance probability, $\alpha(\mathbf{x}_0, \mathbf{x}_1) = w_{1|0}$, replaced by the standard MH acceptance probability

$$\alpha(\mathbf{x}_0, \mathbf{x}_1) = \min\left\{1, \tilde{w}_\theta(\mathbf{x}_1 \mid \mathbf{x}_0)/\tilde{w}_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)\right\}. \quad (18)$$

We use a conditional proposal distribution and calculate the weights according to Eq. (8). The kernel will leave p_θ invariant as it fulfils the detailed balance condition (Hastings, 1970). Just as with the original CNCE criterion, we propose to marginalise over the latent

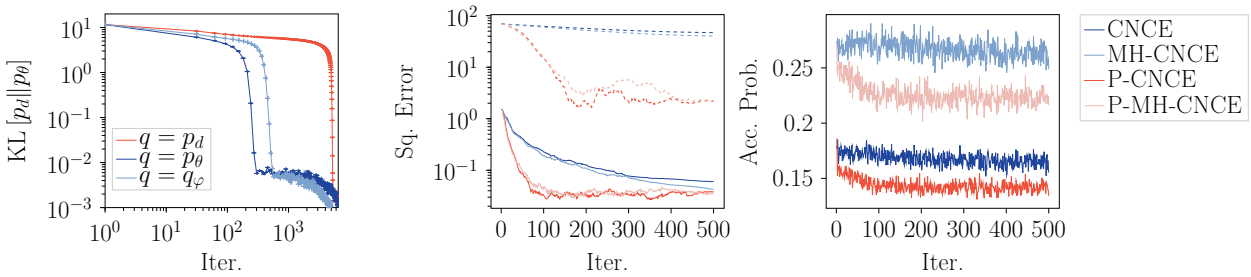


Figure 1: **Left:** Convergence of p_θ for different choices of proposal distribution q. Here, q_φ is initialised at p_d and we show the median divergence KL $[p_d \| p_\theta]$. The error bars mark the 25th and 75th percentile respectively, estimated from 20 repetitions. **Middle-Right:** Results for ring model experiments reported over training iterations and as median (solid lines) and worst-case (dashed lines) estimated from 100 experiments. Middle: Squared parameter error of CNCE, CNCE with Metropolis–Hastings acceptance probability (MH-CNCE), persistent CNCE (P-CNCE) and persistent MH-CNCE (P-MH-CNCE). Right: Acceptance probability of (P-)CNCE and (P-)MH-CNCE when training with (P-)CNCE.

variable z and to use an average over J noisy examples to reduce the variance of the Monte Carlo estimate. We refer to this method as MH-CNCE.

5.4 Sequential Monte Carlo RNCE

Sequential Monte Carlo (SMC) is a generalisation of IS which interleaves IS steps with resampling in a sequential manner; see, e.g., (Naesseth et al., 2019). SMC is particularly useful for sampling from time series or other sequential models, but can be used more generally (Naesseth et al., 2019). The interpretation of RNCE as ML-CIS suggests a generalisation of RNCE obtained by replacing CIS with Conditional SMC (CSMC; see (Andrieu et al., 2010)). Details on this algorithm are given in Appendix B. The resulting SMC-RNCE method has the potential to improve RNCE for sequential models or, wherever SMC is more efficient than IS.

6 EXPERIMENTS

We provide experiments to empirically test the theoretical results of the paper and to demonstrate the proposed extensions of the NCE criteria. For additional experiments and details, see Appendix D [1].

6.1 Adaptive proposal distribution

To support the claims of Section 5.1, we conduct a toy experiment where $p_d = \mathcal{N}(\mathbf{0}, \mathbb{I})$ and $p_\theta = \mathcal{N}(\boldsymbol{\mu}_\theta, \Sigma_\theta)$ are 5-dimensional multivariate Gaussians, allowing us to sample from and evaluate both distributions exactly. The model p_θ is parameterised by a mean vector $\boldsymbol{\mu}_\theta$ and a diagonal covariance matrix Σ_θ, which are estimated using RNCE.

We study the effect of adapting the proposal distribution to either the data or model distribution. For reference, we test the idealised cases where $q = p_d$ and $q = p_\theta$, the former is fixed whereas for the latter we set q to the current p_θ at every step. The adaptive proposal $q = q_\varphi$ is parameterised the same way as p_θ, but with independent parameters φ. It is jointly estimated with p_θ, using the approximation of the gradient in Eq. (17), which only requires the unnormalised model $\tilde{p}_\theta$. To make the problem more challenging and realistic, we initialise q_φ equal to p_d, since we in practice would only have access to samples from p_d for initialisation.

Fig. 1 shows the convergence of p_θ for the different choices of proposals. It is clear that $q = p_\theta$ is the best choice. Interestingly, the (only practically applicable) adaptive proposal q_φ performs much better than using the exact data distribution. Note that an adaptive proposal which targets the static p_d corresponds to matching the moments of p_d from data, and would therefore be a very close approximation of $q = p_d$.

6.2 MH variant and persistent CNCE

The MH acceptance probability in Eq. (18) is known to perform well in the MCMC setting and we investigate its impact for CNCE, by evaluating the proposed MH-CNCE extension. We also investigate the performance benefits of applying persistence to CNCE (P-CNCE) as well as to the MH extension (P-MH-CNCE). To this end, we conduct an experiment similar to the ring model experiment in (Ceylan and Gutmann, 2018). The unnormalised probability density function (pdf) is given by: $\log \tilde{p}_\theta(\mathbf{x}) = -0.5 \exp(\theta)(\|\mathbf{x}\|_2 - \mu)^2$ with $\mathbf{x} \in \mathbb{R}^5$ and $\|\cdot\|_2$ the Euclidean norm. We seek to estimate the log precision θ, while the mean μ is known.

We train models using SGD, with $N = 200$ data sam-

[1] Code available at `github.com/jackonelli/nce_cd_cis`

Table 1: Results for the autoregressive EBM, given as mean ± standard error over ten estimates. **Top:** Test log-likelihood estimated using SMC with $5 \cdot 10^6$ samples. **Bottom:** 2-Wasserstein distance between p_θ and p_d, estimated using $1 \cdot 10^4$ samples ($2 \cdot 10^3$ samples for Miniboone). Samples from p_θ are drawn using SMC.

Dataset (D):	Power (6)	Gas (8)	Hepmass (21)	Miniboone (43)	BSDS300 (63)
			Log-likelihood		
ML-IS	-3.93 ± 0.195	-2037.2 ± 0.25	-607.42 ± 48.02	-	-
RNCE	0.617 ± 0.010	2.60 ± 4.68	-14.95 ± 0.001	-249.72 ± 8.70	**155.37 ± 1.09**
SMC-RNCE	**0.695 ± 0.0002**	**13.01 ± 0.013**	**-13.47 ± 0.0004**	**-15.24 ± 0.65**	145.73 ± 1.22
			Wasserstein distance		
ML-IS	255.75 ± 51.54	5183.4 ± 4332.5	747.01 ± 60.25	-	-
RNCE	44.11 ± 1.54	203.69 ± 12.57	244.84 ± 0.614	1175.7 ± 201.52	85.47 ± 6.54
SMC-RNCE	**39.32 ± 1.25**	**64.02 ± 8.79**	**236.70 ± 0.300**	**259.94 ± 49.74**	**76.50 ± 9.34**

ples drawn from the true pdf and $J = 5$ samples from q. The proposal distribution, q, is a Gaussian, centred at the data sample, $\mathbf{x}_0$, and with a diagonal covariance matrix, $q(\mathbf{x}_1 \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_1; \mathbf{x}_0, \epsilon^2 \mathbb{I}\right)$. The parameter ϵ is a hyperparameter, estimated as the mean standard deviation of the training data.

During training, we measure the squared error of the estimated precision, $\exp(\theta)$. Fig. 1 shows the median and worst-case squared error obtained over 100 experiments. To assess the difference between the CNCE and MH-CNCE acceptance probabilities, we additionally track these quantities and report their median in Fig. 1. For the comparison to be reasonable, we need to evaluate the probabilities on the same set of samples and hence show the acceptance probabilities obtained when training with CNCE (or P-CNCE for PCD), but where we also calculate the MH acceptance probability. The trends are similar when training with (P-)MH-CNCE and evaluating both probabilities.

The MH acceptance probability is indeed larger than the one used by CNCE, as confirmed by Fig. 1. This also seems to lead to slightly faster converge, at least for standard CNCE. Even when the improvement in performance is small, changing the acceptance probability comes without any additional costs. Further improvements over both CNCE and MH-CNCE are seen by adding persistence, in terms of convergence rate as well as both median and worst-case performance.

6.3 Autoregressive EBM

Inspired by Nash and Durkan (2019); Strauss and Oliva (2021), we perform experiments with an autoregressive EBM (AR-EBM). Specifically, we factorise the model as $p_\theta(\mathbf{x}) = \frac{1}{Z_\theta} \prod_{d=1}^{D} \tilde{p}_\theta(\mathbf{x}_d \mid \mathbf{x}_{1:(d-1)})$, for a given ordering of the D features $\mathbf{x}_d$, $d = 1, \ldots, D$, and where $\mathbf{x}_{1:(d-1)} = [\mathbf{x}_1, \ldots, \mathbf{x}_{d-1}]^\top$. The AR-EBM predicts the energy $-\log \tilde{p}_\theta(\mathbf{x}_d \mid \mathbf{x}_{1:(d-1)})$ of feature d conditional on the preceding features $\mathbf{x}_{1:(d-1)}$.

We learn a parameterised proposal distribution q_φ, together with the AR-EBM. The proposal distribution also has an autoregressive factorisation and each factor is a Gaussian Mixture Model (GMM) with 10 components. In the experiments, both the AR-EBM and the proposal are parameterised by fully-connected neural networks with residual connections (He et al., 2016), see Appendix D for details.

The AR-EBM has a sequential structure that can be leveraged by an SMC algorithm. We therefore compare the following methods for training the model: ML-IS, RNCE, and the proposed SMC extension of RNCE (SMC-RNCE, Section 5.4). We perform experiments on four datasets (Power, Gas, Hepmass and Miniboone) from the UCI machine learning repository (Kelly et al.) as well as the BSDS300 dataset (Martin et al., 2001), pre-processed according to Nash and Durkan (2019).

All methods use a total of $J = 20$ samples from the proposal, either to estimate the log-normalisation constant in Eq. (2) (ML-IS) or as negative examples (RNCE/SMC-RNCE). In Table 1, we report test log-likelihoods, estimated using SMC, as well as estimated 2-Wasserstein distances (Villani, 2009) between p_θ and p_d for each AR-EBM.

Results for ML-IS are omitted for the two datasets of highest dimension, (Miniboone and BSDS300), as we found training to be highly unstable. As expected, we observe a performance advantage of the proposed SMC-RNCE criterion, and particularly for the Gas and Miniboone datasets. We also observe an advantage of RNCE over ML-IS, as suggested by the established equivalence between RNCE and ML-CIS.

7 CONCLUSION

In this paper, we contributed to building a more coherent framework for the estimation of unnormalised models, by linking the proxy-criterion noise-contrastive estimation (NCE) to approximate maximum likelihood (ML) methods. Firstly, we established that ranking NCE is equivalent to ML estimation with conditional importance sampling (CIS). This equivalence gives a possible explanation for why ranking NCE would perform better than ML estimation with standard importance sampling; the gradient of the ranking NCE criterion is an approximately unbiased gradient of the log-likelihood. Secondly, we derived ranking NCE and conditional NCE as special cases of contrastive divergence, using MCMC kernels based on CIS and a Metropolis–Hastings-like method, respectively. The established links provide theoretical support for why the optimal noise distribution in NCE is the model, and not the data, distribution and we propose a practical method for adapting the proposal to this distribution. Moreover, our integration of NCE into a standard MCMC setting enables the use of more robust MCMC approaches while at the same time preserving the efficiency and simplicity of NCE. We propose several extensions to the NCE methods and showcase their potential to improve model performance.

Acknowledgments

This research is financially supported by the Swedish Research Council via the project *Handling Uncertainty in Machine Learning Systems* (contract number: 2020-04122), the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and the Excellence Center at Linköping–Lund in Information Technology (ELLIIT).

References

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society Series B*, 72(3):269–342, 2010.

A. U. Asuncion, Q. Liu, A. T. Ihler, and P. Smyth. Particle filtered MCMC-MLE with connections to contrastive divergence. In *International Conference on Machine Learning*, 2010.

A. A. Barker. Monte Carlo calculations of the radial distribution functions for a proton-electron plasma. *Australian Journal of Physics*, 18(2):119–133, 1965.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

M. F. Bugallo, V. Elvira, L. Martino, D. Luengo, J. Miguez, and P. M. Djuric. Adaptive importance sampling: The past, the present, and the future. *IEEE Signal Processing Magazine*, 34(4):60–79, 2017.

M. A. Carreira-Perpinan and G. Hinton. On contrastive divergence learning. In *International Workshop on Artificial Intelligence and Statistics*, 2005.

C. Ceylan and M. U. Gutmann. Conditional noise-contrastive estimation of unnormalised models. In *International Conference on Machine Learning*, 2018.

O. Chehab, A. Gramfort, and A. Hyvarinen. The optimal noise in noise-contrastive learning is not what you think. In *Conference on Uncertainty in Artificial Intelligence*, 2022.

Y. Du, S. Li, J. Tenenbaum, and I. Mordatch. Improved Contrastive Divergence Training of Energy Based Models. In *International Conference on Machine Learning*, 2021.

P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, 2022.

R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu. Flow contrastive estimation of energy-based models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

I. Goodfellow. On distinguishability criteria for estimating generative models. In *Workshop Contribution in International Conference on Learning Representations*, 2015.

F. K. Gustafsson, M. Danelljan, R. Timofte, and T. B. Schön. How to train your energy-based model for regression. In *British Machine Vision Virtual Conference*, 2020.

F. K. Gustafsson, M. Danelljan, and T. B. Schön. Learning proposals for practical energy-based regression. In *International Conference on Artificial Intelligence and Statistics*, pages 4685–4704, 2022.

M. Gutmann and J. Hirayama. Bregman divergence as general framework to estimate unnormalized statistical models. In *Conference on Uncertainty in Artificial Intelligence*, 2011.

M. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361, 2012.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14 (8):1771–1800, Aug 2002.

A. Hyvärinen. Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *IEEE Transactions on Neural Networks*, 18(5):1529–1531, 2007.

R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

M. Kelly, R. Longjohn, and K. Nottingham. UCI machine learning repository. URL `http://archive.ics.uci.edu`.

D. P. Kingma and J. Ba Lei. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

U. Köster, J. T. Lindgren, and A. Hyvärinen. Estimating Markov random field potentials for natural images. In *International Conference on Independent Component Analysis and Signal Separation*. Springer-Verlag, 2009.

Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang, and et al. A tutorial on energy-based learning. *Predicting structured data*, 2006.

Z. Ma and M. Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *Conference on Empirical Methods in Natural Language Processing*, 2018.

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision*, 2001.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, and A. H. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

C. Naesseth, F. Lindsten, and D. Blei. Markovian score climbing: Variational inference with $\mathrm{KL}(p||q)$. In *Advances in Neural Information Processing Systems*, 2020.

C. A. Naesseth, F. Lindsten, and T. B. Schön. Elements of sequential Monte Carlo. *Foundations and Trends in Machine Learning*, 12(3):307–392, Nov 2019.

C. Nash and C. Durkan. Autoregressive energy machines. In *International Conference on Machine Learning*, 2019.

P. Peskun. Optimum Monte-Carlo sampling using Markov chains. *Biometrika*, 60(3):607–612, 1973.

M. Pihlaja, M. Gutmann, and A. Hyvärinen. A family of computationally efficient and simple estimators for unnormalized statistical models. In *Conference on Uncertainty in Artificial Intelligence*, page 442–449. AUAI Press, 2010.

C. P. Robert, G. Casella, and G. Casella. *Monte Carlo statistical methods*. Springer, 1999.

R. Strauss and J. B. Oliva. Arbitrary conditional distributions with energy. *Advances in Neural Information Processing Systems*, 34, 2021.

T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *International Conference on Machine Learning*, 2008.

M. Uehara, T. Kanamori, T. Takenouchi, and T. Matsuda. A unified statistically efficient estimation framework for unnormalized models. In *International Conference on Artificial Intelligence and Statistics*, 2020.

C. Villani. *The Wasserstein distances*, pages 93–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-71050-9. doi: 10.1007/978-3-540-71050-9_6. URL `https://doi.org/10.1007/978-3-540-71050-9_6`.

M. Welling, A. Mnih, and G. E. Hinton. Wormholes improve contrastive divergence. *Advances in Neural Information Processing Systems*, 2003.

N. Xu. Self-adapting noise-contrastive estimation for energy-based models. Master's thesis, Tsinghua University, 2022.

O. Yair and T. Michaeli. Contrastive divergence learning is a time reversal adversarial game. In *International Conference on Learning Representations*, 2021.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes, in the main paper with some additional details given in the supplementary material.**
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes.**
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **The full source code is published at** `github.com/jackonelli/nce_cd_cis`.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. **Yes.**
   (b) Complete proofs of all theoretical results. **Yes. All proofs are provided in the supplementary material.**
   (c) Clear explanations of any assumptions. **Yes.**

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Yes, we use only simulated or publicly available data and the experiments are well described. The source code for all experiments is published at** `https://github.com/jackonelli/nce_cd_cis`.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes, in the experiment section of the main paper, as well as the supplementary material.**
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes.**
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes, these details are written in the supplementary material.**

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. **Yes.**
   (b) The license information of the assets, if applicable. **Not applicable. For the non-simulated data we have cited the asset providers as per their instructions.**
   (c) New assets either in the supplemental material or as a URL, if applicable. **Not applicable.**
   (d) Information about consent from data providers/curators. **Not applicable.**
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not applicable.**

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. **Not applicable.**
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not applicable.**
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not applicable.**

A THEORETICAL DERIVATIONS

A.1 Gradient of the negative log-likelihood with the IS estimator

In maximum likelihood estimation with importance sampling (ML-IS), we approximate the normalisation constant Z_θ in Eq. (2) with the IS estimator, defined in Eq. (3), using J samples $\mathbf{x}_j \sim q(\cdot)$, $j = 1, \ldots, J$. The gradient of this approximation is

$$-\nabla_\theta \log p_\theta(\mathbf{x}_0) \approx -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \nabla_\theta \log \hat{Z}_\theta^{\mathrm{IS}}.$$

Using the definition of the weights $\tilde{w}_\theta(\mathbf{x}_j)$ in Eq. (3), we write the gradient of the estimated log-normalisation constant:

$$\begin{aligned}\nabla_\theta \log \hat{Z}_\theta^{\mathrm{IS}} &= \frac{1}{\hat{Z}_\theta^{\mathrm{IS}}} \nabla_\theta \hat{Z}_\theta^{\mathrm{IS}} = \frac{1}{J\hat{Z}_\theta^{\mathrm{IS}}} \sum_{j=1}^{J} \nabla_\theta \tilde{w}_\theta(\mathbf{x}_j) = \frac{1}{J\hat{Z}_\theta^{\mathrm{IS}}} \sum_{j=1}^{J} \tilde{w}_\theta(\mathbf{x}_j) \nabla_\theta \log \tilde{w}_\theta(\mathbf{x}_j) \\ &= \frac{1}{J\hat{Z}_\theta^{\mathrm{IS}}} \sum_{j=1}^{J} \tilde{w}_\theta(\mathbf{x}_j) \nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j) = \sum_{j=1}^{J} \frac{\tilde{w}_\theta(\mathbf{x}_j)}{\sum_{\ell=1}^{J} \tilde{w}_\theta(\mathbf{x}_\ell)} \nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j). \end{aligned} \tag{19}$$

Note that this gradient is a self-normalised estimate of the desired gradient, $\mathbb{E}_{p_\theta(\mathbf{x})}[\nabla_\theta \log \tilde{p}_\theta(\mathbf{x})]$, and will therefore typically be biased (Robert et al., 1999). Note also that the normalisation is only done over the samples from $q(\cdot)$ and differs from the normalised weight w_j as defined in Eq. (9), where the sum is over $j = 0, \ldots, J$.

A.2 RNCE criterion derivation

The RNCE criterion by Jozefowicz et al. (2016) is based on a multi-class classification problem with a single true data point and multiple noisy ones. Recall, we have $\mathbf{x}_0 \sim p_d(\cdot)$ and noisy samples $\mathbf{x}_j \sim q(\cdot), j = 1, \ldots, J$. We prepend $\mathbf{x}_0$ to the noisy samples and define $\mathbf{x}_{0:J} = [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_J]$.

Assume we forget the origin of $\mathbf{x}_0, \mathbf{x}_{1:J}$. Let the variable $z \in \{0, \ldots, J\}$ denote the index, or class, of the true data sample, and assume that all outcomes are equally probable a priori, i.e., $p(z = j) = 1/(J + 1)$ for $j = 0, 1, \ldots, J$. Conditioned on $\mathbf{x}_{0:J}$, we want the model to maximise the posterior probability of $z = 0$:

$$\begin{aligned} p(z = 0 \mid \mathbf{x}_{0:J}) &= \frac{p(\mathbf{x}_{1:J} \mid z = 0) p(\mathbf{x}_0 \mid z = 0) p(z = 0)}{p(\mathbf{x}_{0:J})} = \frac{p_\theta(\mathbf{x}_0) \prod_{j=1}^{J} q(\mathbf{x}_j) p(z = 0)}{\sum_{j=0}^{J} p_\theta(\mathbf{x}_j) \prod_{\ell \neq j} q(\mathbf{x}_\ell) p(z = j)} \\ &= \frac{p_\theta(\mathbf{x}_0) \prod_{j=1}^{J} q(\mathbf{x}_j)}{\sum_{j=0}^{J} p_\theta(\mathbf{x}_j) \prod_{\ell \neq j} q(\mathbf{x}_\ell)} = \left\{ \text{Divide num. and den. by } \frac{1}{Z_\theta} \prod_{j=0}^{J} q(\mathbf{x}_j) \right\} \\ &= \frac{\tilde{p}_\theta(\mathbf{x}_0)/q(\mathbf{x}_0)}{\sum_{j=0}^{J} \tilde{p}_\theta(\mathbf{x}_j)/q(\mathbf{x}_j)} = \frac{\tilde{w}_\theta(\mathbf{x}_0)}{\sum_{j=0}^{J} \tilde{w}_\theta(\mathbf{x}_j)}. \end{aligned} \tag{20}$$

The RNCE criterion in Eq. (6) follows from minimising the negative logarithm of this probability.

A.3 Proof of Proposition 3.1

First, we derive an expression for the gradient of the RNCE criterion in terms of the CIS estimator $\hat{Z}_\theta^{\mathrm{CIS}}$ in Eq. (10). We use the expression for the RNCE criterion in Eq. (6):

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\mathrm{R}}(\theta, \mathbf{x}_{0:J}) &= \nabla_\theta \left(-\log \tilde{w}_\theta(\mathbf{x}_0) + \log \sum_{j=0}^{J} \tilde{w}_\theta(\mathbf{x}_j) \right) \\ &= \nabla_\theta \left(-\log \tilde{w}_\theta(\mathbf{x}_0) + \log(J+1) \hat{Z}_\theta^{\mathrm{CIS}} \right) \\ &= -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \nabla_\theta \log q(\mathbf{x}_0) + \nabla_\theta \log \hat{Z}_\theta^{\mathrm{CIS}} + \nabla_\theta \log(J+1) \\ &= -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \nabla_\theta \log \hat{Z}_\theta^{\mathrm{CIS}}. \end{aligned} \tag{21}$$

This is equivalent to the gradient of the negative log-likelihood in Eq. (2), with the normalisation constant estimated with CIS:

$$\begin{aligned}\nabla_\theta(-\log p_\theta(\mathbf{x}_0)) &\approx \nabla_\theta(-\log \tilde{p}_\theta(\mathbf{x}_0) + \log \hat{Z}_\theta^{\text{CIS}}) \\ &= -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \nabla_\theta \log \hat{Z}_\theta^{\text{CIS}}. \end{aligned} \tag{22}$$

This concludes the proof. However, for completeness, we also demonstrate that the expression in Eq. (21) matches the expression in Eq. (9). The second term in the last equality of Eq. (21) can be expressed as

$$\begin{aligned}\nabla_\theta \log \hat{Z}_\theta^{\text{CIS}} &= \frac{1}{\hat{Z}_\theta^{\text{CIS}}} \nabla_\theta \hat{Z}_\theta^{\text{CIS}} = \frac{1}{(J+1)\hat{Z}_\theta^{\text{CIS}}} \sum_{j=0}^{J} \nabla_\theta \tilde{w}_\theta(\mathbf{x}_j) \\ &= \frac{1}{(J+1)\hat{Z}_\theta^{\text{CIS}}} \sum_{j=0}^{J} \tilde{w}_\theta(\mathbf{x}_j) \nabla_\theta \log \tilde{w}_\theta(\mathbf{x}_j) = \sum_{j=0}^{J} \frac{\tilde{w}_\theta(\mathbf{x}_j)}{\sum_{\ell=0}^{J} \tilde{w}_\theta(\mathbf{x}_\ell)} \nabla_\theta \log \tilde{w}_\theta(\mathbf{x}_j) \\ &= \sum_{j=0}^{J} w_j \nabla_\theta \log \tilde{w}_\theta(\mathbf{x}_j) = \sum_{j=0}^{J} w_j \nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j). \end{aligned} \tag{23}$$

For the last equality, we use $\nabla_\theta \log q(\mathbf{x}_j) = 0$, assuming that q is independent of θ. Clearly, plugging Eq. (23) into Eq. (21) yields the expression in Eq. (9).

A.4 Proof of Proposition 3.2

To compute the CIS estimator of the normalisation constant $\hat{Z}_\theta^{\text{CIS}}$ we first sample an index z uniformly, such that

$$p(z = i) = \begin{cases} \frac{1}{J+1} & \text{if } i \in \{0, 1, \dots, J\}, \\ 0 & \text{otherwise,} \end{cases} \tag{24}$$

and then sample

$$\begin{aligned} &\mathbf{x}_z \sim p_\theta(\mathbf{x}), \\ &\mathbf{x}_j \sim q(\mathbf{x}),\ j \neq z, j = 0, \dots, J. \end{aligned} \tag{25}$$

To simplify the notation, we introduce $\mathbf{x}_{-z} := [\mathbf{x}_0, \dots, \mathbf{x}_{z-1}, \mathbf{x}_{z+1}, \dots, \mathbf{x}_J]$. Note that we in the main paper, without loss of generalisation, fix $z = 0$ (we can just re-order the indices).

First, we prove a general property of the CIS estimator:

Lemma A.1 (Unbiased general CIS estimate)**.** *Assume $\mathbf{x}_z \sim p_\theta(\cdot)$ and $\mathbf{x}_{-z} \sim q(\cdot)$. Then, for any function f and deterministic index $i \in \{0, \dots, J\}$*

$$\mathbb{E}_{p(z, \mathbf{x}_{0:J})} \left[\frac{f(\mathbf{x}_i)}{\hat{Z}_\theta^{CIS}} \right] = \frac{1}{Z_\theta} \mathbb{E}_{q(\mathbf{x})} \left[f(\mathbf{x}) \right]. \tag{26}$$

Proof. We can write the joint distribution as

$$p(z, \mathbf{x}_{0:J}) = \overbrace{\frac{1}{J+1}}^{p(z)} \times \overbrace{p_\theta(\mathbf{x}_z)}^{p(\mathbf{x}_z|z)} \times \overbrace{\prod_{j \neq z} q(\mathbf{x}_j)}^{p(\mathbf{x}_{-z})} \tag{27}$$

Then, we find the marginal distribution of $\mathbf{x}_{0:J}$ as

$$\begin{aligned} p(\mathbf{x}_{0:J}) &= \sum_{z=0}^{J} \frac{1}{J+1} p_\theta(\mathbf{x}_z) \prod_{j \neq z} q(\mathbf{x}_j) = \sum_{z=0}^{J} \frac{1}{(J+1)Z_\theta} \tilde{w}_\theta(\mathbf{x}_z) \prod_{j=0}^{J} q(\mathbf{x}_j) \\ &= \frac{1}{(J+1)Z_\theta} \prod_{j=0}^{J} q(\mathbf{x}_j) \sum_{z=0}^{J} \tilde{w}_\theta(\mathbf{x}_z) = \frac{\hat{Z}_\theta^{\text{CIS}}}{Z_\theta} \prod_{j=0}^{J} q(\mathbf{x}_j) \end{aligned} \tag{28}$$

and from Eq. (27)

$$p(z \mid \mathbf{x}_{0:J}) = \frac{p(z, \mathbf{x}_{0:J})}{p(\mathbf{x}_{0:J})} = \frac{\tilde{w}_\theta(\mathbf{x}_z)}{\sum_{\ell=0}^{J} \tilde{w}_\theta(\mathbf{x}_\ell)}. \tag{29}$$

Then, for any function $f(\cdot)$ and deterministic index i we have:

$$\begin{aligned}
\mathbb{E}_{p(z,\mathbf{x}_{0:J})}\left[\frac{f(\mathbf{x}_i)}{\hat{Z}_\theta^{\text{CIS}}}\right] &= \mathbb{E}_{p(\mathbf{x}_{0:J})}\left[\frac{f(\mathbf{x}_i)}{\hat{Z}_\theta^{\text{CIS}}}\right] = \int \frac{f(\mathbf{x}_i)}{\hat{Z}_\theta^{\text{CIS}}} \frac{\hat{Z}_\theta^{\text{CIS}}}{Z_\theta} \prod_{j=0}^{J} q(\mathbf{x}_j) d\mathbf{x}_{0:J} \\
&= \frac{1}{Z_\theta} \int f(\mathbf{x}_i) q(\mathbf{x}_i) d\mathbf{x}_i \prod_{j \neq i} q(\mathbf{x}_j) d\mathbf{x}_{-i} = \frac{1}{Z_\theta} \mathbb{E}_q\left[f(\mathbf{x})\right] \int \prod_{j \neq i} q(\mathbf{x}_j) d\mathbf{x}_{-i} \\
&= \frac{1}{Z_\theta} \mathbb{E}_q\left[f(\mathbf{x})\right],
\end{aligned} \tag{30}$$

where the first equality follows from the fact that the integrand does not depend on z. □

To prove Proposition 3.2, we first note that

$$\begin{aligned}
\mathbb{E}_{p_\theta(\mathbf{x}_0) q(\mathbf{x}_{1:J})}\left[\nabla_\theta \log \hat{Z}_\theta^{\text{CIS}}\right] &= \mathbb{E}_{p(z,\mathbf{x}_{0:J})}\left[\nabla_\theta \log \hat{Z}_\theta^{\text{CIS}}\right] = \mathbb{E}_{p(z,\mathbf{x}_{0:J})}\left[\frac{1}{\hat{Z}_\theta^{\text{CIS}}} \nabla_\theta \hat{Z}_\theta^{\text{CIS}}\right] \\
&= \frac{1}{J+1} \sum_{j=0}^{J} \mathbb{E}_{p(z,\mathbf{x}_{0:J})}\left[\frac{\nabla_\theta \tilde{w}_\theta(\mathbf{x}_j)}{\hat{Z}_\theta^{\text{CIS}}}\right].
\end{aligned} \tag{31}$$

From Lemma A.1 we have

$$\begin{aligned}
\mathbb{E}_{p(z,\mathbf{x}_{0:J})}\left[\frac{\nabla_\theta \tilde{w}_\theta(\mathbf{x}_j)}{\hat{Z}_\theta^{\text{CIS}}}\right] &= \frac{1}{Z_\theta} \mathbb{E}_q\left[\nabla_\theta \tilde{w}_\theta(\mathbf{x})\right] \\
&= \frac{1}{Z_\theta} \int \tilde{w}_\theta(\mathbf{x}) \left[\nabla_\theta \log \tilde{w}_\theta(\mathbf{x})\right] q(\mathbf{x}) \mathrm{d}\mathbf{x} \\
&= \frac{1}{Z_\theta} \int \frac{Z_\theta p_\theta(\mathbf{x})}{q(\mathbf{x})} \left[\nabla_\theta \log \frac{Z_\theta p_\theta(\mathbf{x})}{q(\mathbf{x})}\right] q(\mathbf{x}) \mathrm{d}\mathbf{x} \\
&= \int p_\theta(\mathbf{x}) \left[\nabla_\theta \log Z_\theta + \nabla_\theta \log p_\theta(\mathbf{x}) - \nabla_\theta \log q(\mathbf{x})\right] \mathrm{d}\mathbf{x} \\
&= \nabla_\theta \log Z_\theta.
\end{aligned} \tag{32}$$

The middle term in the second to last row is zero since:

$$\int p_\theta(\mathbf{x}) \nabla_\theta \log p_\theta(\mathbf{x}) \mathrm{d}\mathbf{x} = \int \nabla_\theta p_\theta(\mathbf{x}) \mathrm{d}\mathbf{x} = \nabla_\theta \int p_\theta(\mathbf{x}) \mathrm{d}\mathbf{x} = \nabla_\theta 1 = 0, \tag{33}$$

and $\nabla_\theta \log q(\mathbf{x}) = 0$ if $q(\mathbf{x})$ is independent of θ.

Finally, we conclude the proof with

$$\mathbb{E}_{p_\theta(\mathbf{x}_0) q(\mathbf{x}_{1:J})}\left[\nabla_\theta \log \hat{Z}_\theta^{\text{CIS}}\right] = \frac{1}{J+1} \sum_{j=0}^{J} \nabla_\theta \log Z_\theta = \nabla_\theta \log Z_\theta. \tag{34}$$

That is, we get an unbiased estimate of the $\nabla_\theta \log Z_\theta$ if q is independent of θ and if the estimate is computed using a sample from the distribution $p(z, \mathbf{x}_{0:J})$ defined above, corresponding to the CIS procedure where we sample from $p_\theta(\mathbf{x}_0) q(\mathbf{x}_{1:J})$.

A.5 Proof of Proposition 4.1

The expectation w.r.t. K_θ^{R} in Eq. (5), starting at a data sample $\mathbf{x}_0 \sim p_d$, is given by

$$\mathbb{E}_{K_\theta^{\mathrm{R}}(\mathbf{x}'|\mathbf{x}_0)}\left[\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}')\right] = \mathbb{E}_{q(\mathbf{x}_{1:J})}\left[\mathbb{E}_{\mathrm{Categorical}(z;w_{0:J})}\left[\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_z)\right]\right] = \mathbb{E}_{q(\mathbf{x}_{1:J})}\left[\sum_{j=0}^{J} w_j \nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j)\right]. \quad (35)$$

Approximating the expected value with a single Monte Carlo sample $\mathbf{x}_{1:J} \sim q(\cdot)$ and plugging in the expression for the weights w_j when evaluating Eq. (5), we recover Eq. (9).

A.6 CNCE criterion derivation

The CNCE criterion is a proxy-criterion based on a binary classification problem with true sample $\mathbf{x}_0 \sim p_d(\cdot)$ and noisy sample $\mathbf{x}_1 \sim q(\cdot \mid \mathbf{x}_0)$. We forget the origin of $\mathbf{x}_0, \mathbf{x}_1$ and introduce the latent class variable $z \in \{0, 1\}$ according to

$$p_\theta(\mathbf{x}_0, \mathbf{x}_1 \mid z) = \begin{cases} p_\theta(\mathbf{x}_0) q(\mathbf{x}_1 \mid \mathbf{x}_0), & \text{if } z = 0, \\ p_\theta(\mathbf{x}_1) q(\mathbf{x}_0 \mid \mathbf{x}_1), & \text{if } z = 1. \end{cases} \quad (36)$$

Let $p(z = 0) = \eta$. The posterior of z follows as

$$\begin{aligned} p_\theta(z = 0 \mid \mathbf{x}_0, \mathbf{x}_1) &= \frac{p_\theta(\mathbf{x}_0, \mathbf{x}_1 \mid z = 0)\eta}{p_\theta(\mathbf{x}_0, \mathbf{x}_1 \mid z = 0))\eta + p_\theta(\mathbf{x}_0, \mathbf{x}_1 \mid z = 1))(1 - \eta)} \\ &= \frac{p_\theta(\mathbf{x}_0) q(\mathbf{x}_1 \mid \mathbf{x}_0)\eta}{p_\theta(\mathbf{x}_0) q(\mathbf{x}_1 \mid \mathbf{x}_0)\eta + p_\theta(\mathbf{x}_1) q(\mathbf{x}_0 \mid \mathbf{x}_1)(1 - \eta)} \\ &= \frac{\tilde{p}_\theta(\mathbf{x}_0) q(\mathbf{x}_1 \mid \mathbf{x}_0)\eta}{\tilde{p}_\theta(\mathbf{x}_0) q(\mathbf{x}_1 \mid \mathbf{x}_0)\eta + \tilde{p}_\theta(\mathbf{x}_1) q(\mathbf{x}_0 \mid \mathbf{x}_1)(1 - \eta)} \\ &= \frac{1}{1 + \dfrac{\tilde{w}_\theta(\mathbf{x}_1 \mid \mathbf{x}_0)(1 - \eta)}{\tilde{w}_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)\eta}}. \end{aligned} \quad (37)$$

Note that the normalisation constant of p_θ cancels, so that we can use the unnormalised model $\tilde{p}_\theta$ directly to calculate the posterior. In accordance with Ceylan and Gutmann (2018), we assume a uniform prior on z, i.e. $\eta = \frac{1}{2}$, as well as average over J noisy samples, which yields the CNCE criterion in Eq. (7).

A.7 Gradient of CNCE criterion

We derive the gradient of the CNCE criterion in Eq. (7):

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\mathrm{C}}(\theta, \mathbf{x}_{0:J}) &= \frac{1}{J} \sum_{j=1}^{J} \nabla_\theta \log \left(1 + \frac{\tilde{w}_\theta(\mathbf{x}_j \mid \mathbf{x}_0)}{\tilde{w}_\theta(\mathbf{x}_0 \mid \mathbf{x}_j)}\right) \\ &= -\frac{1}{J} \sum_{j=1}^{J} \nabla_\theta \log \left(\frac{\tilde{p}_\theta(\mathbf{x}_0) q(\mathbf{x}_j \mid \mathbf{x}_0)}{\tilde{p}_\theta(\mathbf{x}_0) q(\mathbf{x}_j \mid \mathbf{x}_0) + \tilde{p}_\theta(\mathbf{x}_j) q(\mathbf{x}_0 \mid \mathbf{x}_j)}\right) \\ &= \frac{1}{J} \sum_{j=1}^{J} \big(-\nabla_\theta \log \left(\tilde{p}_\theta(\mathbf{x}_0) q(\mathbf{x}_j \mid \mathbf{x}_0)\right) + \nabla_\theta \log \left(\tilde{p}_\theta(\mathbf{x}_0) q(\mathbf{x}_j \mid \mathbf{x}_0) + \tilde{p}_\theta(\mathbf{x}_j) q(\mathbf{x}_0 \mid \mathbf{x}_j)\right) \big) \\ &= -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \frac{1}{J} \sum_{j=1}^{J} \big(\nabla_\theta \log \left(\tilde{p}_\theta(\mathbf{x}_0) q(\mathbf{x}_j \mid \mathbf{x}_0) + \tilde{p}_\theta(\mathbf{x}_j) q(\mathbf{x}_0 \mid \mathbf{x}_j)\right) \big) \end{aligned} \quad (38)$$

Let $Z_\theta^z = \tilde{p}_\theta(\mathbf{x}_0)q(\mathbf{x}_j \mid \mathbf{x}_0) + \tilde{p}_\theta(\mathbf{x}_j)q(\mathbf{x}_0 \mid \mathbf{x}_j)$, then

$$\begin{aligned}\nabla_\theta \log Z_\theta^z = \frac{\nabla_\theta Z_\theta^z}{Z_\theta^z} &= \frac{q(\mathbf{x}_j \mid \mathbf{x}_0)\nabla_\theta \tilde{p}_\theta(\mathbf{x}_0) + q(\mathbf{x}_0 \mid \mathbf{x}_j)\nabla_\theta \tilde{p}_\theta(\mathbf{x}_j)}{\tilde{p}_\theta(\mathbf{x}_0)q(\mathbf{x}_j \mid \mathbf{x}_0) + \tilde{p}_\theta(\mathbf{x}_j)q(\mathbf{x}_0 \mid \mathbf{x}_j)} \\ &= \frac{q(\mathbf{x}_j \mid \mathbf{x}_0)\tilde{p}_\theta(\mathbf{x}_0)\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + q(\mathbf{x}_0 \mid \mathbf{x}_j)\tilde{p}_\theta(\mathbf{x}_j)\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j)}{\tilde{p}_\theta(\mathbf{x}_0)q(\mathbf{x}_j \mid \mathbf{x}_0) + \tilde{p}_\theta(\mathbf{x}_j)q(\mathbf{x}_0 \mid \mathbf{x}_j)} \\ &= w_{0|j}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + w_{j|0}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j) \\ &= (1 - w_{j|0})\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + w_{j|0}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j). \end{aligned} \tag{39}$$

A.8 Proof of Proposition 4.2

To show that CNCE is equivalent to using CD-1 together with the kernel given by Algorithm 2, we calculate the expectation with respect to K_θ^{C}, initialising at $\mathbf{x}_0$

$$\mathbb{E}_{K_\theta^{\mathrm{C}}(\mathbf{x}' \mid \mathbf{x}_0)}\left[\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}')\right] = \mathbb{E}_{q(\mathbf{x}_1 \mid \mathbf{x}_0)}\left[(1 - w_{1|0})\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + w_{1|0}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_1)\right]. \tag{40}$$

We recover Eq. (12) by approximating the expectation with an average over samples $\mathbf{x}_j \sim q(\cdot \mid \mathbf{x}_0)$, $j = 1, \ldots, J$ and by plugging the result into Eq. (5).

A.9 Proofs of Propositions 5.1 and 5.2

With $q = p_\theta$ we have uniform weights $w_j = \frac{1}{J+1}, j = 0, \ldots, J$ (from Eq. (9)). From Eq. (9), we then have

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\mathrm{R}}(\theta) &= -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \sum_{j=0}^{J} \frac{1}{J+1}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j) \\ &= -\frac{J}{J+1}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \frac{1}{J+1}\sum_{j=1}^{J}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j). \end{aligned} \tag{41}$$

Taking the expectation w.r.t. $q(\mathbf{x}_{1:J}) = p_\theta(\mathbf{x}_{1:J})$

$$\begin{aligned}\mathbb{E}_{q(\mathbf{x}_{1:J})}\left[\nabla_\theta \mathcal{L}_{\mathrm{R}}(\theta)\right] &= -\frac{J}{J+1}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \frac{1}{J+1}\sum_{j=1}^{J}\mathbb{E}_{p_\theta(\mathbf{x}_j)}\left[\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j)\right] \\ &= \frac{J}{J+1}\left(-\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \nabla_\theta \log Z_\theta\right) \\ &= \frac{J}{J+1}\left(-\nabla_\theta \log p_\theta(\mathbf{x}_0)\right), \end{aligned} \tag{42}$$

which proves Proposition 5.1.

Analogous to the above proof, with $q(\cdot \mid \mathbf{x}_0) = p_\theta(\cdot)$ we have uniform weights $w(\mathbf{x}_0 \mid \mathbf{x}_j) = w(\mathbf{x}_j \mid \mathbf{x}_0) = \frac{1}{2}$, $j = 1, \ldots, J$. From Eq. (12), we then have

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\mathrm{C}}(\theta) &= -\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \frac{1}{J}\sum_{j=1}^{J}\left(\frac{1}{2}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \frac{1}{2}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j)\right) \\ &= -\frac{1}{2}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \frac{1}{2J}\sum_{j=1}^{J}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j). \end{aligned} \tag{43}$$

Taking the expectation w.r.t. $q(\mathbf{x}_{1:J} \mid \mathbf{x}_0) = p_\theta(\mathbf{x}_{1:J})$

$$\begin{aligned}\mathbb{E}_{q(\mathbf{x}_{1:J} \mid \mathbf{x}_0)}\left[\nabla_\theta \mathcal{L}_{\mathrm{C}}(\theta)\right] &= -\frac{1}{2}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \frac{1}{2J}\sum_{j=1}^{J}\mathbb{E}_{p_\theta(\mathbf{x}_j)}\left[\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j)\right] \\ &= \frac{1}{2}\left(-\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_0) + \nabla_\theta \log Z_\theta\right) \\ &= \frac{1}{2}\left(-\nabla_\theta \log p_\theta(\mathbf{x}_0)\right), \end{aligned} \tag{44}$$

which proves Proposition 5.2.

A.10 Proof of Proposition 5.3

From Eq. (17) we have

$$\nabla \mathcal{L}(\varphi) = \mathbb{E}_{p_\theta(\mathbf{x})}\left[-\nabla \log q_\varphi(\mathbf{x})\right] \approx \nabla \hat{\mathcal{L}}(\varphi) = -\sum_{j=0}^{J} w_j \nabla_\varphi \log q_\varphi(\mathbf{x}_j)$$
$$= -\frac{1}{J+1}\sum_{j=0}^{J} \frac{\tilde{w}_\theta(\mathbf{x}_j)\nabla_\varphi \log q_\varphi(\mathbf{x}_j)}{\hat{Z}_\theta^{\mathrm{CIS}}}. \tag{45}$$

Then

$$\mathbb{E}_{p_\theta(\mathbf{x}_0)q_\varphi(\mathbf{x}_{1:J})}\left[\nabla \hat{\mathcal{L}}(\varphi)\right] = \mathbb{E}_{p(z,\mathbf{x}_{0:J})}\left[\nabla \hat{\mathcal{L}}(\varphi)\right] = -\frac{1}{J+1}\sum_{j=0}^{J}\mathbb{E}_{p(z,\mathbf{x}_{0:J})}\left[\frac{1}{\hat{Z}_\theta^{\mathrm{CIS}}}\tilde{w}_\theta(\mathbf{x}_j)\nabla_\varphi \log q_\varphi(\mathbf{x}_j)\right]. \tag{46}$$

From Lemma A.1 we have

$$\mathbb{E}_{p(z,\mathbf{x}_{0:J})}\left[\frac{\tilde{w}_\theta(\mathbf{x}_j)\nabla_\varphi \log q_\varphi(\mathbf{x}_j)}{\hat{Z}_\theta^{\mathrm{CIS}}}\right] = \frac{1}{Z_\theta}\mathbb{E}_{q_\varphi(\mathbf{x})}\left[\tilde{w}_\theta(\mathbf{x})\nabla_\varphi \log q_\varphi(\mathbf{x})\right] = \mathbb{E}_{q_\varphi(\mathbf{x})}\left[\frac{1}{Z_\theta}\tilde{w}_\theta(\mathbf{x})\nabla_\varphi \log q_\varphi(\mathbf{x})\right]$$
$$= \int q_\varphi(\mathbf{x})\frac{1}{Z_\theta}\tilde{w}_\theta(\mathbf{x})\nabla_\varphi \log q_\varphi(\mathbf{x})\mathrm{d}\mathbf{x}$$
$$= \int q_\varphi(\mathbf{x})\frac{1}{Z_\theta}\frac{\tilde{p}_\theta(\mathbf{x})}{q_\varphi(\mathbf{x})}\nabla_\varphi \log q_\varphi(\mathbf{x})\mathrm{d}\mathbf{x}$$
$$= \int p_\theta(\mathbf{x})\nabla_\varphi \log q_\varphi(\mathbf{x})\mathrm{d}\mathbf{x} = \mathbb{E}_{p_\theta(\mathbf{x})}\left[\nabla_\varphi \log q_\varphi(\mathbf{x})\right] = -\nabla_\varphi \mathcal{L}(\varphi). \tag{47}$$

Then

$$\mathbb{E}_{p_\theta(\mathbf{x}_0)q_\varphi(\mathbf{x}_{1:J})}\left[\nabla \hat{\mathcal{L}}(\varphi)\right] = \frac{1}{J+1}\sum_{j=0}^{J}\nabla_\varphi \mathcal{L}(\varphi) = \nabla_\varphi \mathcal{L}(\varphi), \tag{48}$$

which concludes the proof.

B EXTENSIONS OF NCE

In the main paper, we outlined a new adaptive proposal strategy, persistent NCE, an MH-variant of CNCE as well as an SMC variant of RNCE, as extensions of the NCE criteria. Here, we detail one additional extension based on CD, namely that of taking several MCMC steps in the CD kernel. We also give further details on the SMC variant of RNCE (SMC-RNCE).

B.1 NCE with multiple MCMC steps

In the light of interpreting RNCE and CNCE as special cases of contrastive divergence, a natural extension of these criteria is that of taking several MCMC steps in the kernel. This, with the hope that it will improve convergence of the algorithm. We outline the procedure for RNCE with k MCMC steps, using the kernel in Algorithm 1. At each MCMC step $\ell = 1, \dots, k$, we sample $\mathbf{x}_{1:J}^{(\ell)}$ and condition on $\mathbf{x}_0^{(\ell)} = \mathbf{x}_z^{(\ell-1)}$, with $\mathbf{x}_z^{(\ell-1)}$ being the sampled output from the previous step. Marginalising over the index variables, z, we estimate the second term in Eq. (5) as

$$\mathbb{E}_{K_\theta^{\mathrm{R}}(\mathbf{x}'|\mathbf{x}_0)}\left[\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}')\right] \approx \frac{1}{k}\sum_{\ell=1}^{k}\sum_{j=0}^{J} w_j^{(\ell)}\nabla_\theta \log \tilde{p}_\theta(\mathbf{x}_j^{(\ell)}). \tag{49}$$

Hence, we estimate the expected gradient as an average over the estimates obtained at each step of the kernel. Note that the weight normalisation, Eq. (9), is performed independently at each step ℓ, using only samples involved in that particular step. The procedure would be similar for CNCE, but instead employing the kernel outlined in Algorithm 2.

B.2 Sequential Monte Carlo RNCE

Interpreting RNCE as CD-1 with a kernel based on CIS, we propose an extension to RNCE, where the CIS kernel is replaced by a kernel based on conditional Sequential Monte Carlo (SMC), see e.g. (Naesseth et al., 2019), referred to by SMC-RNCE. Here, we give details on the CSMC algorithm. The CSMC kernel is outlined in Algorithm 3.

SMC, in general, tries to address the issue of weight degeneracy sometimes observed in IS, see e.g. (Naesseth et al., 2019), by solving the inference problem recursively. Assume that the model density factorises as

$$p_\theta(\mathbf{x}) = \frac{1}{Z_\theta} \prod_{d=1}^{D} \tilde{p}_\theta(\mathbf{x}_d \mid \mathbf{x}_{1:(d-1)}), \tag{50}$$

for a given ordering of the D features $\mathbf{x}_d$, $d = 1, \dots, D$, and where $\mathbf{x}_{1:(d-1)} = [\mathbf{x}_1, \dots, \mathbf{x}_{d-1}]$. In this case, SMC can make use of the autoregressive structure to recursively draw samples from the proposal distribution and by adapting said distribution based on the previously drawn samples.

In Conditional SMC (CSMC), similar to CIS, we condition on a sample $\mathbf{x}_0$ (a data sample in SMC-RNCE), which is set deterministically in the SMC algorithm. Following the notation in Eq. (50), CSMC iterates over all features starting at $d = 1$ and ending at $d = D$. At step d, samples $\mathbf{x}_{1:d}^{(j)}$, $j = 1, \dots, J$, are drawn from the proposal

$$q(\mathbf{x}_{1:d}) = q(\mathbf{x}_d \mid \mathbf{x}_{1:(d-1)}) \sum_{j=0}^{J} w_{j,d-1} \delta_{\mathbf{x}_{1:(d-1)}^{(j)}}(\mathbf{x}_{1:(d-1)}), \tag{51}$$

with $\delta_{\mathbf{x}_{1:(d-1)}^{(j)}}(\cdot)$ the Dirac delta distribution at the previously drawn sample $\mathbf{x}_{1:(d-1)}^{(j)}$ (or the conditioning sample for the case $j = 0$). The weights at step d are calculated as

$$w_{j,d} = \frac{w_\theta(\mathbf{x}_d^{(j)})}{\sum_{\ell=0}^{J} w_\theta(\mathbf{x}_d^{(\ell)})}, \tag{52}$$

with, for a model that factorises according to Eq. (50),

$$w_\theta(\mathbf{x}_d^{(j)}) = \frac{\tilde{p}_\theta(\mathbf{x}_d^{(j)} \mid \mathbf{x}_{1:(d-1)}^{(j)})}{q(\mathbf{x}_d^{(j)} \mid \mathbf{x}_{1:(d-1)}^{(j)})}. \tag{53}$$

The CSMC estimate of the normalisation constant, Z_θ, is

$$\hat{Z}_\theta^{\text{CSMC}} = \prod_{d=1}^{D} \frac{1}{J+1} \sum_{j=0}^{J} w_\theta(\mathbf{x}_d^{(j)}). \tag{54}$$

One issue that can arise in SMC is so-called path degeneracy, see e.g. (Naesseth et al., 2019). To alleviate this issue, we can use *adaptive resampling* and sample $\mathbf{x}_{1:(d-1)}^{(j)}$, $j = 1, \dots, J$, at step d only if the effective sample size (ESS_{d-1}) goes below $(J+1)/2$, and otherwise keep the corresponding samples from the last iteration. The effective sample size is calculated as

$$\text{ESS}_d = \frac{1}{\sum_{j=0}^{J} w_{j,d}^2}.$$

In case we do not resample $\mathbf{x}_{1:(d-1)}^{(j)}$, we account for this by calculating the weights according to

$$w_\theta(\mathbf{x}_d^{(j)}) = \frac{w_{j,d-1}}{1/(J+1)} \frac{\tilde{p}_\theta(\mathbf{x}_d^{(j)} \mid \mathbf{x}_{1:(d-1)}^{(j)})}{q(\mathbf{x}_d^{(j)} \mid \mathbf{x}_{1:(d-1)}^{(j)})}. \tag{55}$$

These adapted weights are then used in the CSMC estimate of the normalisation constant (Eq. (54)).

Algorithm 3 CSMC kernel

Input: $\mathbf{x}_0$
for $d = 1$ **to** D **do**
  for $j = 1$ **to** J **do**
    1. **if** $d = 1$
        Set $\mathbf{x}_{1:(d-1)}^{(j)} \coloneqq \emptyset$
      else
        Sample $z \sim \text{Categorical}\left([w_{0,d-1}, \dots, w_{J,d-1}]\right)$, set $\mathbf{x}_{1:(d-1)}^{(j)} \coloneqq \mathbf{x}_{1:(d-1)}^{(z)}$
    2. Sample $\mathbf{x}_d^{(j)} \sim q(\cdot \mid \mathbf{x}_{1:(d-1)}^{(j)})$
    3. Calculate weight $w_\theta(\mathbf{x}_d^{(j)})$, using Eq. (53)
  4. Calculate weights $w_\theta(\mathbf{x}_d^{(0)})$, $w_{j,d}$, $j = 0, \dots, J$, using Eqs. (52) and (53)
6. Sample $z \sim \text{Categorical}\left([w_{0,D}, \dots, w_{J,D}]\right)$
7. Return $\mathbf{x}' = \mathbf{x}_{1:D}^{(z)}$

C COMPARISON WITH YAIR AND MICHAELI (2021)

Most similar to our contribution is the work by Yair and Michaeli (2021). They show that CD-1 can be derived from CNCE, while we show that CNCE (as well as RNCE) can be derived as special cases of CD-1. Both results give valuable insight to two important families of estimation methods. However, our two different approaches lead to some crucial distinctions, both conceptual and theoretical, which we detail below.

Yair and Michaeli (2021) use the original derivation of the CD gradient, starting at another objective function than the log-likelihood, see (Hinton, 2002). From there, they argue that this derivation is flawed, since it assumes that an intractable term can be neglected. Deriving CD-1 from CNCE is therefore more principled, they claim. Here, we take the opposite view. We view the CD gradient in Eq. (5) as a straightforward MCMC approximation of the log-likelihood gradient in Eq. (4). This view is common and leads to exactly the same gradient expression, see e.g. (Welling et al., 2003; Asuncion et al., 2010). NCE on the other hand, is derived by introducing a proxy-criterion without any apparent connection to standard ML estimation. Therefore, we claim that it is more useful to formulate the NCE methods in terms of their connection to ML.

The theoretical differences stem from the way Yair and Michaeli (2021) derive their connection. Specifically, they rewrite the gradient of the CNCE criterion on the form

$$\nabla_\theta \mathcal{L}_{\text{C}}(\theta) = \mathbb{E}_{\mathbf{x}_0 \sim p_d, \mathbf{x}_1 \mid \mathbf{x}_0 \sim q}\left[\alpha_\theta(\mathbf{x}_0, \mathbf{x}_1)\left(-\nabla_\theta \log p_\theta(\mathbf{x}_0) + \nabla_\theta \log p_\theta(\mathbf{x}_1)\right)\right] \tag{56a}$$

$$\alpha_\theta(\mathbf{x}_0, \mathbf{x}_1) = \left(1 + \frac{p_\theta(\mathbf{x}_0) q(\mathbf{x}_1 \mid \mathbf{x}_0)}{p_\theta(\mathbf{x}_1) q(\mathbf{x}_0 \mid \mathbf{x}_1)}\right)^{-1}. \tag{56b}$$

If q is chosen to be the transition probability of a reversible Markov chain, then the detailed balance condition is fulfilled and $\alpha_\theta(\mathbf{x}_0, \mathbf{x}_1) = 1/2$, $\forall \mathbf{x}_0, \mathbf{x}_1$, which means that the gradient of the CNCE criterion is proportional to the CD-1 gradient. Note that the derivation holds only for proposal (or noise) distributions which satisfy this property.

Going in the opposite direction, we start from the CD gradient estimate in Eq. (5) and derive both CNCE and RNCE as special cases of CD-1. We establish these links without any restrictions on the proposal distribution q. By deriving CNCE from CD-1, we are also able to discover that CNCE corresponds to CD with a well-known MH kernel, albeit with a sub-optimal acceptance probability. This allows us to propose a theoretical improvement to CNCE (MH-CNCE), at virtually no cost.

Our interpretation of RNCE and CNCE as CD-1 also provides a strong argument for choosing q similar to p_θ, rather than p_d. Yair and Michaeli (2021) are more ambiguous on this point and simultaneously claim that q should not significantly deviate from p_d, while also requiring that q depend on p_θ.

We also arrive at different ways of generalising the methods to CD-k, that is with multi-step sampling in the MCMC kernel. For us, the multi-step versions of RNCE and CNCE follows naturally by taking multiple steps in the respective MCMC kernels, resulting in CD-k, see Appendix B.1. This avoids the need to introduce the "time-reversal classification task" used by Yair and Michaeli (2021). Importantly, we do pair-wise comparison of the conditional sample and the newly proposed one at each step of the MCMC kernel, while Yair and Michaeli (2021) compare the full MCMC chain to the reversed one for all k steps at once.

D ADDITIONAL EXPERIMENTS AND EXPERIMENT DETAILS

We provide additional details on the experiments performed in the main article, as well as additional experimental results.

D.1 Adaptive proposal distribution

The model p_θ is parameterised by a mean vector $\boldsymbol{\mu}_\theta$ and a vector $\boldsymbol{s}_\theta$, such that Σ_θ is a diagonal covariance matrix, where the diagonal is the elements of $\boldsymbol{s}_\theta$ squared. We initialise $\boldsymbol{\mu}_\theta = \mathbf{4}$ and $\Sigma_\theta = 2\mathbb{I}$.

The models are trained using $N = 100$ samples from p_d and $J = 10$ proposal samples from q for every data point drawn from p_d. The parameters are estimated using SGD with learning rate $\kappa = 0.01\sqrt{B}$, where $B = 32$ is the batch size. The parameter vectors θ and φ are updated once every batch, using the same learning rate.

D.2 MH variant and persistent CNCE

We provide additional results, investigating the effect of changing the CNCE acceptance probability to the one of the standard Metropolis–Hastings (MH) algorithm as well as that of using persistence in CNCE and MH-CNCE (P-CNCE and P-MH-CNCE, respectively). We also give additional details on the ring model experiments performed in the main paper.

We perform experiments using the ring model explained in Section 6.2 with $N = 200, 1000$ and $J = 5, 10$. The proposal q is the same as previously. Each experiment is repeated 100 times, each time with a new set of uniformly sampled model parameters $\mu \in \{5, 10\}$, $\sigma^2 \in \{0.3, 1.5\}$, with $\theta = \log(\sigma^{-2})$, and a new data set of N data points. Initial estimates of θ is sampled uniformly from the same interval as the true value. To the best of our ability, we follow the setup in (Ceylan and Gutmann, 2018), but select only two data set sizes, as well as run additional experiments with a smaller number of noise samples ($J = 5$). Moreover, we train our models using SGD, training each model for 50 epochs with a batch size of $B = 20$.

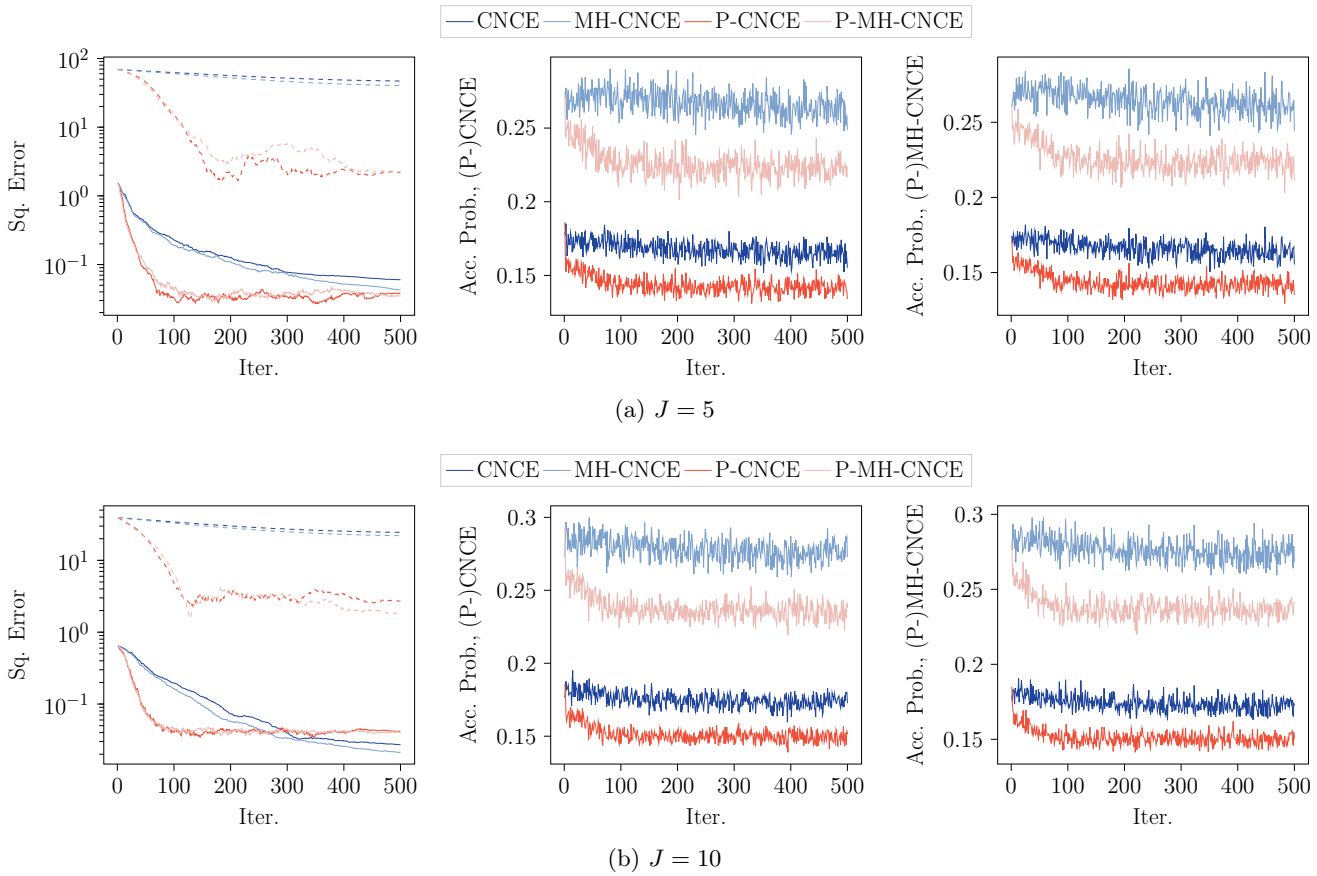
For improving stability of the persistent CNCE methods, we use a decaying learning rate. The learning rate is set as $\kappa = \kappa_{\text{base}} \cdot \sqrt{B}$ and is linearly decayed, starting at $\kappa_{\text{base}} = 0.01$ and ending at $\kappa_{\text{base}} = 0.001$. For P-CNCE as well as P-MH-CNCE, we run $B \cdot J$ MCMC chains in parallel, as each data point $\mathbf{x}_0^i$ in a batch forms a total of J pairs $(\mathbf{x}_0, \mathbf{x}_j)$, $j = 1, \ldots, J$.

Results are shown in Fig. 2 and 3. In addition to the median squared error of the estimated precision over iterations, we show the worst-case performance at each iteration. We also show the median acceptance probabilities obtained with all methods, calculated when training with CNCE and MH-CNCE for the standard algorithms, or P-CNCE and P-MH-CNCE for the persistent algorithms.

The advantages of a higher acceptance probability as well as persistence is most evident for the smaller sample size. However, also for $N = 1000$ we observe that both P-CNCE and P-MH-CNCE improves worst-case performance over standard CNCE and MH-CNCE. While the median performance of P-CNCE and P-MH-CNCE is worse, this still indicates that the methods have some robustness. The performance advantage of using the MH acceptance probability, compared to the one used in CNCE, is less evident when combined with persistence.

D.3 Autoregressive EBM

The experimental setup for the experiments performed with the autoregressive EBM (AR-EBM) are based on similar experiments performed by Nash and Durkan (2019); Strauss and Oliva (2021). Note that, while the experiments are based on previous work (Nash and Durkan, 2019; Strauss and Oliva, 2021), we formulate the model distribution such that the conditional distributions in Eq. (50) share a single normalisation constant, Z_θ.



(a) $J = 5$

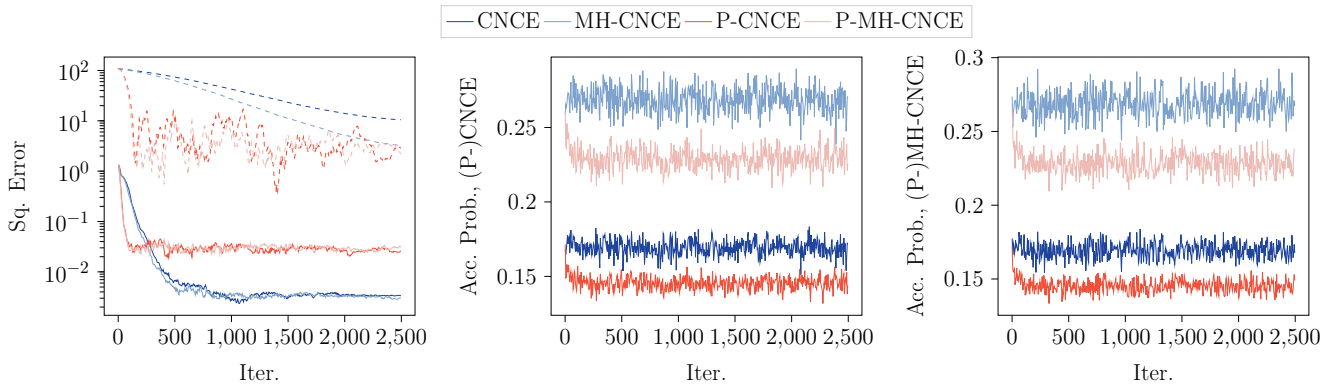
(b) $J = 10$

Figure 2: Results for ring model experiments with $N = 200$ training data points. Results are reported over training iterations and as median (solid lines) and worst-case (dashed lines) estimated from 100 experiments. Left: Squared parameter error of standard CNCE, CNCE with Metropolis–Hastings acceptance probability (MH-CNCE), persistent CNCE (P-CNCE) and persistent MH-CNCE (P-MH-CNCE). Middle: Acceptance probability of (P-)CNCE and (P-)MH-CNCE when training with (P-)CNCE. Right: Acceptance probability of CNCE and MH-CNCE when training with (P-)MH-CNCE.

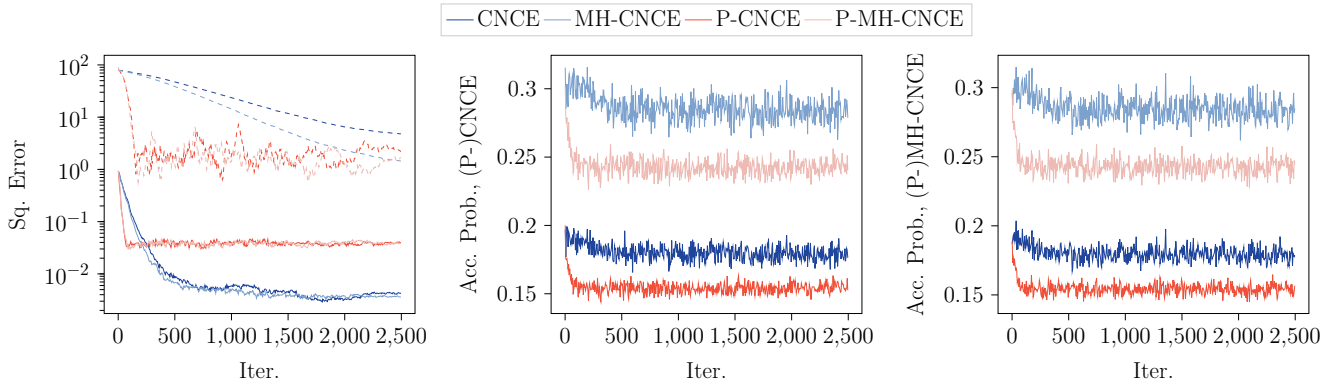
In contrast, the model distributions in (Nash and Durkan, 2019; Strauss and Oliva, 2021) are formulated such that each one-dimensional conditional distribution has its own normalisation constant, which is also estimated separately from the other's.

For the experiments, we select reasonable hyperparameters based on values used in (Nash and Durkan, 2019). In some cases, we adapt the hyperparameters based on initial test runs using ML-IS or to reduce computational cost (see below). We also use the test runs, guided by Nash and Durkan (2019), to decide on a sufficient number of training iterations for each dataset. The hyperparameters are summarised in Table 3. We found training with ML-IS on the Miniboone and BSDS300 datasets to be highly unstable. As we did not manage to find hyperparameters that stabilise training for this method combined with these datasets, we omit the corresponding results.

In all experiments, both the AR-EBM, p_θ, and the proposal network, q_φ, are fully-connected neural networks with residual connections, consisting of pre-activation residual blocks with two layers (He et al., 2016). For the AR-EBM we use four residual blocks with 128 hidden units for all datasets. For the proposal network, we use two residual blocks with 512 hidden units for the dataset of highest dimension (BSDS300) and two residual blocks with 256 hidden units for the remaining datasets. The size of the AR-EBM is directly taken from Nash and Durkan (2019), while we choose a smaller proposal network than used in (Nash and Durkan, 2019), to reduce computational cost. In both models, we use ReLU activations and apply dropout between the two layers of each residual block, using a dropout rate of 0.1, as a default. However, as we did not observe convergence of the loss on



(a) $J = 5$



(b) $J = 10$

Figure 3: Results for ring model experiments with $N = 1000$ training data points. Results are reported over training iterations and as median (solid lines) and worst-case (dashed lines) estimated from 100 experiments. Left: Squared parameter error of standard CNCE, CNCE with Metropolis–Hastings acceptance probability (MH-CNCE) and persistent CNCE (P-CNCE). Middle: Acceptance probability of CNCE and MH-CNCE when training with (P-)CNCE. Right: Acceptance probability of CNCE and MH-CNCE when training with (P-)MH-CNCE.

the Gas dataset using this setup, we exchange the activation function in the AR-EBM to Tanh and train without dropout for this dataset, following Nash and Durkan (2019). Moreover, for the smaller Miniboone dataset, we use a dropout rate of 0.5 as in (Nash and Durkan, 2019), to avoid overfitting.

The structure of the proposal network is taken from (Strauss and Oliva, 2021). The input to the network is the feature vector $\mathbf{x}_0$ with all but the observed features $x_{0,1:(d-1)}$ set to 0, and a one-hot mask indicating which features are observed (with zeros in the positions of unobserved features, and ones in the positions of observed features). The proposal network parameterises a Gaussian Mixture Model (GMM) with ten components. The GMM is used to evaluate the conditional density $q_\varphi(\mathbf{x}_d \mid x_{1:(d-1)})$ and to generate proposal/noise samples. To avoid numerical issues, we set the standard deviation of each Gaussian to a minimum of $1 \cdot 10^{-3}$.

Apart from the parameters of the GMM, we follow previous experimental setups (Nash and Durkan, 2019; Strauss and Oliva, 2021) and let the proposal network output a context vector of length 64. This context vector, which can be interpreted as a latent representation of $x_{1:(d-1)}$, is given as input to the AR-EBM as a means of sharing information between the two models. Given the context vector as well as the unobserved feature $x_{0,d}$ as input, the AR-EBM directly predicts the unnormalised log density (the negative energy) $\log \tilde{p}_\theta(x_d \mid x_{1:(d-1)})$. As in (Nash and Durkan, 2019), we apply a softplus non-linearity to the output of the AR-EBM, such that the predicted density is upper bounded by 1.

Prior to training, data is pre-processed according to Nash and Durkan (2019). For the Power, Gas and Miniboone datasets we hold out 10% of the data for testing, and split the remaining dataset into training and validation sets

Table 2: Dimension and total size of the datasets used in the AR-EBM experiments.

	Power	Gas	Hepmass	Miniboone	BSDS300
Dimension (D)	6	8	21	43	63
Size (N)	2,049,280	1,052,065	525,123	36,488	1,300,000

using a 90%-10% split. For Hepmass and BSDS300 we use pre-existing data splits. For Hepmass, this means that the test set makes up approximately one third (instead of 10%) of the full dataset. For BSDS300, the test set instead consists of approximately 20% of the full dataset, and the remaining data is split into training and validation sets using a (approximately) $95\% - 5\%$ split. Each dataset but BSDS300 is standardised and features that are discrete, highly correlated or with too many reoccurring values are removed. For the Power dataset, we additionally add uniform noise to the features, for the purpose of numerical stability. Moreover, instead of using the full images of the original BSDS300 dataset, we use the same data as in (Nash and Durkan, 2019), consisting of patches extracted from the original images. The final number of data dimensions, as well as the total size of each dataset are given in Table 2.

The AR-EBM, p_θ, and proposal network, q_φ, are trained in parallel. We investigate several methods for training the AR-EBM (ML-IS, RNCE and SMC-RNCE) and train q_φ using maximum likelihood as done in (Nash and Durkan, 2019; Strauss and Oliva, 2021). Although we argue that q_φ should resemble p_θ, we find that our proposed method for adapting q_φ to the model distribution is not suitable for the particular setup where information is shared (through the context vector) between the AR-EBM and the proposal network. Having both p_θ and q_φ aiming at the same target (the data distribution), is beneficial in this case, as this updates the context vector in the same direction, while having different targets (the data vs. the model distribution) instead risks stalling training, as the proposal might have a negative impact on the AR-EBM through the context vector.

Just as in (Nash and Durkan, 2019), the models are trained using Adam optimisation (Kingma and Ba Lei, 2015) and a learning rate following a cosine annealing schedule. The initial learning rate is set to $5 \cdot 10^{-4}$. We use a batch size of 512, with the exception for the Miniboone and BSDS300 datasets, where we, because of limitations in GPU memory, use a batch size of 128. The total number of training iterations used for each dataset is reported in Table 3. In all cases, we use the first 5000 iterations as a warm-up phase, where we train only the proposal network and keep p_θ fixed. For training the AR-EBM, all methods use $J = 20$ proposal/noise samples per observation. In SMC-RNCE, we use adaptive resampling, as explained in Appendix B. During training, we evaluate the model on a smaller part (10%) of the validation set every 1,000 training iterations, and keep the one with the highest log-likelihood out of the evaluated (together with the corresponding proposal network). An exception is made for the Miniboone dataset, where we evaluate the models on the full validation set every 1,000 training iterations, because of the smaller number of observations in the dataset.

We train the models on one GPU (NVIDIA GeForce RTX 3090, 24 GB). The total training time depends on the method used to train the AR-EBM as well as the dataset in question. For the Power, Gas and Hepmass datasets, training takes around 5-25 hours, while the corresponding numbers for the Miniboone and BSDS300 datasets are 15-40 and 40-120 hours, respectively.

We evaluate the log-likelihood over the test set, applying SMC to estimate the normalisation constant. We make a total of ten estimates, using $5 \cdot 10^6$ particles in the SMC algorithm, and report the mean as well as standard error of the log-likelihood. The 2-Wasserstein distance is estimated with sampling. We randomly draw (with replacement) $1 \cdot 10^4$ samples from the test set, to represent the data distribution, and use SMC to draw an equal amount of samples from the model distribution. An exception is again made for the Miniboone dataset, where we draw only $2 \cdot 10^3$ from each distribution, because of the smaller size of the test set. We make a total of ten estimates, and report the mean as well as the standard error of the estimated Wasserstein distance.

Table 3: Hyperparameters used in the AR-EBM experiments.

	Power	Gas	Hepmass	Miniboone	BSDS300
Number of blocks, p_θ	4	4	4	4	4
Hidden dimension, p_θ	128	128	128	128	128
Activation function, p_θ	ReLU	Tanh	ReLU	ReLU	ReLU
Number of blocks, q_φ	2	2	2	2	2
Hidden dimension, q_φ	256	256	256	256	512
Activation function, q_φ	ReLU	ReLU	ReLU	ReLU	ReLU
Context dimension	64	64	64	64	64
GMM components	10	10	10	10	10
Negative samples, J	20	20	20	20	20
Batch size	512	512	512	128	128
Dropout rate (p_θ and q_φ)	0.1	0.0	0.1	0.5	0.1
(Initial) learning rate	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
Training iterations	$1 \cdot 10^{6}$	$6 \cdot 10^{5}$	$2 \cdot 10^{5}$	$3 \cdot 10^{5}$	$6 \cdot 10^{5}$
Warm-up iterations	5000	5000	5000	5000	5000