Conditional Energy-Based Models for Implicit Policies: The Gap between Theory and Practice

Duy-Nguyen Ta, Eric Cousineau, Huihua Zhao, and Siyuan Feng
Toyota Research Institute
Cambridge, Massachusetts, USA
Email: eric.cousineau@tri.global

***Abstract*—We present our findings in the gap between theory and practice of using conditional energy-based models (EBM) as an implicit representation for behavior-cloned policies. We also clarify several subtle, and potentially confusing, details in previous work in an attempt to help future research in this area. We point out key differences between unconditional and conditional EBMs, and warn that blindly applying training methods for one to the other could lead to undesirable results that do not generalize well. Finally, we emphasize the importance of the Maximum Mutual Information principle as a necessary condition to achieve good generalization in conditional EBMs as implicit models for regression tasks.**

I. Introduction

With many intriguing properties recently shown in [5], conditional energy-based models (EBMs) have garnered significant interest in the robotics manipulation community as an implicit representation for behavior-cloned policies to deal with multimodal and inconsistent demonstrations. Unfortunately, training conditional EBMs for regression problems can be challenging in practice, despite lots of remarkable successes of its "cousin" approach, unconditional EBMs, which are typically used as generative models for data on very high dimensional and nonlinear manifolds, e.g. images [8, 3, 2].

This paper does not present any new methods, and provides only minor empirical tuning on top of previous work. Rather, we aim to clarify the gap between theory and practice in training conditional EBMs for regression problems, such as behavior cloning. Our path to improving conditional EBM training was filled with many dead-ends due to incorrect and confusing exposition in the previous text [5, 7] and our failure to realize the differences between conditional and unconditional EBMs, leading to unsuccessful attempts to apply methods for training unconditional EBMs to the other. We hope our lessons shared in this paper help clear the confusion, and prevents other people from making the same mistakes, and steer future research back to the right direction.

II. Background

Instead of representing a policy as an explicit function $y = F_\theta(x)$, directly mapping from an observation x to an action y, Implicit Behavior Cloning (IBC) [5] proposes to first learn an energy function $E_\theta(x, y)$ that maps an observation-action pair to an energy value in $\mathbb{R}$. For inference, this energy function is minimized to produce a (hopefully) optimal action: $\hat{y} = \arg\min_y E_\theta(x, y)$. The intriguing benefits of this policy representation scheme have been discussed in [5].

In IBC, training the policy entails learning the energy function $E_\theta(x, y)$ from a dataset of observation-action pairs collected from human demonstrations. As presented in [5], the loss function for training $E_\theta(x, y)$ is the negative log likelihood (NLL): $\mathcal{L}_{NLL} = \sum_{i=1}^{N} -\log p_\theta(y_i|x_i)$, in which the probability density function $\log p_\theta(y|x)$ relates to the EBM function $E_\theta(x, y)$ via the Gibbs distribution: $p_\theta(y|x) = \exp(-E_\theta(x, y))/Z_\theta(x)$, where $Z_\theta(x) = \int \exp(-E_\theta(x, y))dy$ is the normalization factor depending on the given x. In short,

$$\mathcal{L}_{NLL}(\theta) = \sum_{i=1}^{N} -\log \frac{\exp(-E_\theta(x_i, y_i))}{\int \exp(-E_\theta(x_i, y))dy}. \quad (1)$$

NLL loss is also very commonly used in most unconditional EBM works [10, 13], whose main concerns are different ways to deal with the intractable integral in the normalization factor. In [5], the nuisance integral is approximated as

$$\int e^{-E_\theta(x_i,y)}dy \approx e^{-E_\theta(x_i,y_i)} + \sum_{m=1}^{M} e^{-E_\theta(x_i,\tilde{y}_i^m)}, \quad (2)$$

where $\{\tilde{y}_i^m\}_{m=1}^M$ is a set of negative action samples for each observation x_i. This leads to the following loss function:

$$\mathcal{L}_{InfoNCE}(\theta) = \sum_{i=1}^{N} -\log \frac{e^{-E_\theta(x_i,y_i)}}{e^{-E_\theta(x_i,y_i)} + \sum_{m=1}^{M} e^{-E_\theta(x_i,\tilde{y}_i^m)}}. \quad (3)$$

Different sampling schemes could be used to obtain negative samples $\{\tilde{y}_i^m\}_{m=1}^M$ from the action domain. Beside the uniform sampling scheme in Derivative-Free Optimizer (DFO) method, [5] also proposes to use Langevin MCMC to sample directly from the currently learned distribution $p_\theta(y|x_i)$.

III. Common Misinterpretations

The InfoNCE loss (3) actually has a different meaning, which we will revisit in Section V. Here, we discuss the problems of interpreting it as an approximation of the NLL loss as currently presented. In fact, the approximation of the integral using a set of negative samples in (2) raises many questions. First, a set of samples is typically used to approximate the expectation of an arbitrary function: $\mathbb{E}_{p(y)}[f(y)] = \int f(y)p(y)dy \approx \frac{1}{M}\sum_m f(y^m)$, where $y^m \sim p(y)$. However, the normalization factor is **not** an expectation but the integral

of the function itself, hence the formula does not directly apply. A popular technique to better approximate the integral of a function $\int f(y)dy$ is to use importance sampling [7], which turns the integral into an expectation under a chosen proposal distribution $q(y)$: $\int f(y)dy = \int \frac{f(y)}{q(y)} q(y)dy \approx \frac{1}{M} \sum_m \frac{f(y^m)}{q(y^m)}$, where $y^m \sim q(y)$, and the weight $w_m = 1/q(y^m)$ of each sample have to be taken into account. Because of this, the approximation in (2) is only correct if the proposal $q(y)$ is a uniform distribution, as in DFO. Unfortunately, as noted in prior work, uniform sampling does not scale well for problems with high dimensional action spaces [5].[1]

We find that leveraging Langevin MCMC to obtain negative samples from the currently learned distribution $p_\theta(y|x_i)$ to approximate the integral in (2) is also confusing in several other ways, not just because the sample weights are not taken into account. The current set of samples approximating the current distribution becomes outdated when the distribution is updated in the next optimization step. To be correct, experiments (in addition to what is indicated in the IBC code) should be performed doing backpropagation through the entire chain of MCMC; however, this is a challenging task due to the stochasticity and discrete nature of the samples. A similar problem involving optimization of an integral of the distribution under optimization also arises in Variational Auto Encoder, where the reparameterization trick is used to obtain samples from a (different) *fixed* normal distribution [9]. Unfortunately, the technique is not applicable in our case, since the inverse distribution required in the reparameterization trick does not have a closed form formula in our general setting.

In EBM literature, samples of the currently learned distribution from Langevin MCMC are used to approximate the gradient of the NLL loss, not the integral in the loss itself. This effectively leads to another loss function whose gradient w.r.t. θ is the same as the gradient of (1) approximated using the set of MCMC samples [2]:

$$\mathcal{L}_{MCMC}(\theta) = \sum_{i=1}^{N} \left(E_\theta(x_i, y_i) - \frac{1}{M} \sum_{m=0}^{M} E_\theta(x_i, \tilde{y}_i^m) \right). \tag{4}$$

Unfortunately, we did not notice significant improvements in using this loss function in our experiments.

Another important mismatch in theory and practice lies in the Langevin MCMC formula used in IBC. The correct formulation [3, 6, 16] is

$$y^{k+1} = y^k - \frac{\lambda}{2} \nabla_y E_\theta(x, y^k) + \sqrt{\lambda}\, \omega^k, \quad \omega^k \sim \mathcal{N}(0, I) \tag{5}$$

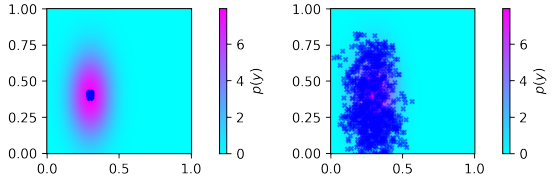
However, in IBC, the following formula is effectively used:

$$y^{k+1} = y^k - \frac{\lambda}{2} \nabla_y E_\theta(x, y^k) + \lambda\, \hat{\omega}^k, \quad \hat{\omega}^k \sim \mathcal{N}(0, \sigma) \tag{6}$$

where σ permits additional scaling of the noise term. Even assuming $\sigma = 1$, the subtle difference between the coefficients

[1]We tried using Gaussian mixture models as the proposal density, following [7], but did not achieve good results for problems in high dimensional spaces.

[2]Detailed derivation can be found in [10]



(a) IBC's Langevin samples (b) True Langevin samples

Fig. 1: IBC's vs True Langevin MCMC samples[3]

for gradient and noise terms can create a large difference in how the samples approximate the underlying distribution. Fig. 1 shows that the correct formulation approximates the mean and variance of the Gaussian accurately, whereas the IBC formulation does not. [3]

However, in training EBM, we found that the IBC's incorrect Langevin MCMC formulation seems to work *better* in practice than the correct formulation. For some of the tasks we tested against, we found it useful to tune the noise scaling σ of IBC's Langevin MCMC down to a small value, typically 0.01, so that the negative samples are very close to the positive samples for good training results. We also found that using one long chain MCMC, throwing away initial samples during the burn-in stage, and collecting M samples at the end of the chain, as typically done in MCMC literature, does not work as well as using multiple independent short chains and selecting the last sample from each chain, as is done in IBC.

These mismatches between theory and practice reveal the gap in our understanding of Langevin MCMC in conditional EBMs training, and we await a better theoretical explanation.

IV. Failed Attempts

To improve EBM's robustness and generalization, we first tried to avoid a well-known "deep valley" problem, where the low energy region around a positive sample is very small and closely surrounded by the high energy regions around it [10]. In fact, the ideal Gibbs distribution that minimizes the NLL loss in (1) is a mixture of Dirac delta distributions whose peaks are at the positive data points and volume under the curve on the entire domain (i.e. the integral in the denominator of (1)) is zero. Such a function, however, might suffer from robustness and generalization issues. Following MaxEnt RL [4], we added a maximum conditional entropy regularizer to the original loss function, detailed in the Appendix, which minimizes the variance of energies of negative samples, effectively forcing the resulting distribution to be close to a uniform distribution, which has the maximum entropy. We observe that the correct Langevin MCMC can produce reasonable results using this regularizer, however, it is not better than the original IBC's Langevin MCMC with a small noise σ.

We also tried to improve the quality of MCMC samples by using a replay buffer to simulate long-chain MCMCs [3]. This technique for training unconditional EBMs does not directly apply to conditional EBMs, since different conditional distributions cannot share the same replay buffer. Hence, we train

[3]Details on the correct and incorrect Langevin formulation, numerical experiments, and plots can be found in Appendix B.

a joint EBM, modeling the joint distribution $p_\theta(x, y)$, which is also reducible to the conditional for inferring the optimal action $p_\theta(x_i, y) \propto p_\theta(y|x_i)$. The results show that it can overfit to the training data very well only after a few epochs, but generalizes very poorly to unseen data in the validation set. This should not come as a surprise because all samples of unseen observations in the joint domain are treated as negative samples during training. The lessons we learned from this failure are two-fold. First, blindly applying techniques for unconditional EBMs to train conditional EBMs might not be a good idea. Second, there are many models with the same representational power that can fit the training data well, but not all of them can generalize well to unseen observations. Although the IBC's incorrect Langevin MCMC happens to generalize well, a principle to select implicit models with better generalization capability is currently lacking.

V. Maximum Mutual Information

We realize the following key difference between unconditional and conditional EBMs in their requirements for generalization: whereas unconditional EBMs for generative model are meant to sample new data from a learned distribution, conditional EBMs for regression requires the inference process to produce outputs that generalize well to unseen inputs.

A necessary condition for good generalization in regression is that the model should maximize the dependencies between the input and output. Hence, one of the goals of the loss function for training conditional EBMs should be to maximize the mutual information between them to improve generalization. In fact, the InfoNCE loss function in (3) was originally derived for that purpose [15]. It is **not** an approximation of the NLL loss, but the negative of a lower bound of the mutual information between observations and actions. Minimizing it effectively maximizes that lower bound, resulting in distributions with high mutual information. Due to that effect, it underlies many successful results in self-supervised contrastive learning, e.g. [1, 12].

However, the InfoNCE loss function has to be used with care. Essentially, it models the ratio between the conditional and the marginal densities in the mutual information $I(x, y) = \int p(x, y) \log \frac{p_\theta(y|x)}{p(y)} dxdy$. Part of its name "noise contrastive estimation" (NCE) stems from the goal to differentiate noisy samples of $p(y)$ from the true samples of $p_\theta(y|x)$. Following the rigorous proof in [11], [14] pointed out that in order for InfoNCE to be a proper bound of the mutual information, the negative samples $\{\tilde{y}_i^m\}_{m=1}^M$ have to be independently sampled from the marginal $p(y)$. This explains why our attempts to obtain the correct samples of the currently learned conditional distribution using a long-chain MCMC did not go well: they are not independent and do not come from a noise distribution. Using independent samples from multiple short chains as done in IBC has a better chance to succeed.

Moreover, the small noise scaling σ in IBC's incorrect Langevin MCMC actually helps in maximizing the mutual information. Due to the small σ, many generated negative samples are very close to the positive ones, leading to the aforementioned "deep valley" effect. However, despite being undesirable in unconditional EBMs, these deep valleys should be favorable in conditional EBMs since they produce conditional densities with very low entropy $\mathcal{H}(y|x)$, effectively increasing the mutual information, since $I(x, y) = \mathcal{H}(y) - \mathcal{H}(y|x)$. Explicit models are the extreme case where $p_\theta(y|x)$ are Dirac delta distributions with zero entropy. This explains why our earlier attempts to maximize the conditional entropy following MaxEnt RL literature headed to a wrong direction.

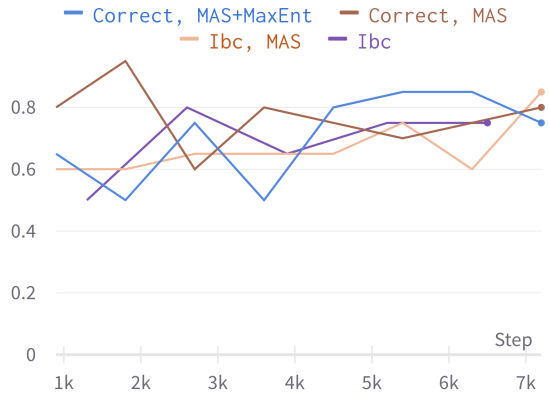


Fig. 2: Validation success rates of IBC's Langevin (cyan) and Marginal Action Sampler (red) [4]

Inspired by the mutual information lower-bound proof of InfoNCE, we also tried to replace the IBC's incorrect Langevin sampler with a marginal action sampler (MAS) to obtain independent negative samples from $p(y)$. We train the marginal EBM $p_\gamma(y)$ using the correct Langevin MCMC method with the maximum entropy regularization (see the Appendix). Maximizing the marginal entropy $\mathcal{H}(y)$ helps to further increase the mutual information (see the $I(x, y)$ formula above). We use MCMC samples from training the marginal action EBM $p_\gamma(y)$ as negative samples in the InfoNCE loss (3) to train the targeted conditional EBM concurrently. Our preliminary results on particle environments in [5] are encouraging. As shown in Fig. 2, EBMs trained by MAS appear to be on par with IBC's incorrect Langevin MCMC. [4]

VI. Conclusion and Future Work

We identified the gap between theory and practice in training conditional EBMs for implicit policy representation, and clarified the confusing interpretation in previous work. We also pointed out the key difference in generalization requirements between unconditional and conditional EBMs, which calls for more attention on conditional EBMs training methods. We arrived at the Maximum Mutual Information principle to improve the generalization of conditional EBMs for regression. However, that principle might not be sufficient. In order for the model to generalize well, it should also capture the "trend" of how the input-output data in their joint domain spread far beyond the training data regions. Leveraging the generalization power of explicit functions to improve that of the implicit ones, e.g. [17], is an interesting direction for future work.

[4] See Appendix D for more details about the training setup.

Acknowledgments

Pete Florence for valuable discussions, and all Google IBC authors for their excellent work.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[2] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

[3] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

[4] Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021.

[5] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[6] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[7] Fredrik K Gustafsson, Martin Danelljan, Radu Timofte, and Thomas B Schön. How to train your energy-based model for regression. In *Proceedings of the British Machine Vision Conference (BMVC)*, September 2020.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[9] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[10] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[11] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[13] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

[14] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*, 2019.

[15] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807, 2018.

[16] Wikipedia contributors. Metropolis-adjusted langevin algorithm. Wikipedia, the free encyclopedia, 2022. URL https://en.wikipedia.org/w/index.php?title=Metropolis-adjusted_Langevin_algorithm&oldid=1074347589. [Online; accessed 21-June-2022].

[17] Jianwen Xie, Zilong Zheng, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of fast thinking initializer and slow thinking solver for conditional learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

APPENDIX

A. Derivation of Maximum Entropy Regularization

Here we derive the maximum entropy ($\mathcal{H}$) regularizer in the negative log-likelihood loss function, used in our failed attempt in Section IV and in the Marginal Action Sampler in Section V:

$$\begin{aligned}
-\mathbb{E}_{p_{data}}\left[\log(p_\theta(y|x)) + \mathcal{H}(p_\theta(y|x))\right] &\approx -\frac{1}{N}\sum_{i=1}^{N}\left(\log(p_\theta(y_i|x_i)) - \int p_\theta(y|x_i)\log(p_\theta(y|x_i))dy\right) \\
&= -\frac{1}{N}\sum_{i=1}^{N}\left(\log\left(\frac{\exp(-E_\theta(x_i,y_i))}{Z_\theta(x_i)}\right) - \int p_\theta(y|x_i)\log\left(\frac{\exp(-E_\theta(x_i,y))}{Z_\theta(x_i)}\right)dy\right) \\
&= -\frac{1}{N}\sum_{i=1}^{N}\left(-E_\theta(x_i,y_i) - \log Z_\theta(x_i) - \int p_\theta(y|x_i)\left(-E_\theta(x_i,y) - \log Z_\theta(x_i)\right)dy\right) \\
&= -\frac{1}{N}\sum_{i=1}^{N}\left(-E_\theta(x_i,y_i) - \log Z_\theta(x_i) + \int p_\theta(y|x_i)E_\theta(x_i,y)dy + \int p_\theta(y|x_i)\log Z_\theta(x_i)dy\right) \\
&= -\frac{1}{N}\sum_{i=1}^{N}\left(-E_\theta(x_i,y_i) - \log Z_\theta(x_i) + \int p_\theta(y|x_i)E_\theta(x_i,y)dy + \log Z_\theta(x_i)\right) \\
&= \frac{1}{N}\sum_{i=1}^{N}\left(E_\theta(x_i,y_i) - \int p_\theta(y|x_i)E_\theta(x_i,y)dy\right),
\end{aligned}$$

Taking the gradient of the loss:

$$-\nabla_\theta\mathbb{E}_{p_{data}}\left[\log p_\theta(y|x) + \mathcal{H}(p_\theta(y|x))\right] \approx \frac{1}{N}\sum_{i=1}^{N}\left(\nabla_\theta E_\theta(x_i,y_i) - \nabla_\theta\int p_\theta(y|x_i)E_\theta(x_i,y)dy\right).$$

Expanding the second term:

$$\begin{aligned}
-\nabla_\theta\int p_\theta(y|x_i)E_\theta(x_i,y)dy &= -\int\left((\nabla_\theta p_\theta(y|x_i))E_\theta(x_i,y) + p_\theta(y|x_i)\nabla_\theta E_\theta(x_i,y)\right)dy \\
&= -\int(\nabla_\theta p_\theta(y|x_i))E_\theta(x_i,y)dy - \mathbb{E}_{p_\theta(y|x_i)}\left[\nabla_\theta E_\theta(x_i,y)\right] \\
&= -\int p_\theta(y|x_i)(\nabla_\theta\log p_\theta(y|x_i))E_\theta(x_i,y)dy - \mathbb{E}_{p_\theta(y|x_i)}\left[\nabla_\theta E_\theta(x_i,y)\right] \\
&= -\mathbb{E}_{p_\theta(y|x_i)}\left[(\nabla_\theta\log p_\theta(y|x_i))E_\theta(x_i,y)\right] - \mathbb{E}_{p_\theta(y|x_i)}\left[\nabla_\theta E_\theta(x_i,y)\right]
\end{aligned}$$

then further expanding the first term:

$$\begin{aligned}
-\mathbb{E}_{p_\theta(y|x_i)}\left[E_\theta(x_i,y)\nabla_\theta\log p_\theta(y|x_i)\right] &= -\mathbb{E}_{p_\theta(y|x_i)}\left[E_\theta(x_i,y)\nabla_\theta(-E_\theta(x_i,y) - \log Z_\theta(y_i))\right] \\
&= \mathbb{E}_{p_\theta(y|x_i)}\left[E_\theta(x_i,y)\nabla_\theta E_\theta(x_i,y) + E_\theta(x_i,y)\nabla_\theta\log Z_\theta(y_i)\right] \\
&= \mathbb{E}_{p_\theta(y|x_i)}\left[E_\theta(x_i,y)\nabla_\theta E_\theta(x_i,y)\right] + \mathbb{E}_{p_\theta(y|x_i)}\left[E_\theta(x_i,y)\right]\nabla_\theta\log Z_\theta(y_i) \\
&= \mathbb{E}_{p_\theta(y|x_i)}\left[E_\theta(x_i,y)\nabla_\theta E_\theta(x_i,y)\right] - \mathbb{E}_{p_\theta(y|x_i)}\left[E_\theta(x_i,y)\right]\mathbb{E}_{p_\theta(y|x_i)}\left[\nabla_\theta E_\theta(x_i,y)\right],
\end{aligned}$$

where we have used the well-known fact that

$$\begin{aligned}
\nabla_\theta\log\int\exp\left(-E_\theta(x_i,y)\right)dy &= \frac{1}{\int\exp\left(-E_\theta(x_i,y)\right)dy}\nabla_\theta\int\exp\left(-E_\theta(x_i,y)\right)dy \\
&= \frac{1}{Z_\theta(x_i)}\int\nabla_\theta\exp\left(-E_\theta(x_i,y)\right)dy \\
&= \int\frac{\exp\left(-E_\theta(x_i,y)\right)}{Z_\theta(x_i)}\nabla_\theta\left(-E_\theta(x_i,y)\right)dy \\
&= \int p_\theta(y|x_i)\nabla_\theta\left(-E_\theta(x_i,y)\right)dy \\
&= \mathbb{E}_{p_\theta(y|x_i)}\left[\nabla_\theta\left(-E_\theta(x_i,y)\right)\right] \qquad (7)
\end{aligned}$$

In summary,

$$-\nabla_\theta \mathbb{E}_{p_{data}}\left[\log p_\theta(y|x) + \mathcal{H}(p_\theta(y|x))\right] \approx \frac{1}{N}\sum_{i=1}^{N}\Big(\nabla_\theta E_\theta(x_i, y_i) - \mathbb{E}_{p_\theta(y|x_i)}\left[\nabla_\theta E_\theta(x_i, y)\right] + \mathbb{E}_{p_\theta(y|x_i)}\left[E_\theta(x_i, y)\nabla_\theta E_\theta(x_i, y)\right] - \mathbb{E}_{p_\theta(y|x_i)}\left[E_\theta(x_i, y)\right]\mathbb{E}_{p_\theta(y|x_i)}\left[\nabla_\theta E_\theta(x_i, y)\right]\Big)$$

Monte Carlo approximation of the gradient:

$$-\nabla_\theta \mathbb{E}_{p_{data}}\left[\log p_\theta(y|x) + \mathcal{H}(p_\theta(y|x))\right] \approx \frac{1}{N}\sum_{i=1}^{N}\left(\nabla_\theta E_\theta(x_i, y_i) - \frac{1}{M}\sum_{m=1}^{M}\nabla_\theta E_\theta(x_i, \tilde{y}_i^m) + \frac{1}{M}\sum_{m=1}^{M}E_\theta(x_i, \tilde{y}_i^m)\nabla_\theta E_\theta(x_i, \tilde{y}_i^m) - \frac{1}{M}\sum_{m=1}^{M}E_\theta(x_i, \tilde{y}_i^m)\,\frac{1}{M}\sum_{m=1}^{M}\nabla_\theta E_\theta(x_i, \tilde{y}_i^m)\right)$$

where $\tilde{y}_i^m \sim p_\theta(y|x_i)$.

The new loss function that has the same gradient is:

$$\mathcal{L}_{entropy}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(E_\theta(x_i, y_i) - \frac{1}{M}\sum_{m=1}^{M}E_\theta(x_i, \tilde{y}_i^m) + \frac{1}{2}\frac{1}{M}\sum_{m=1}^{M}\left(E_\theta(x_i, \tilde{y}_i^m)\right)^2 - \frac{1}{2}\left(\frac{1}{M}\sum_{m=1}^{M}E_\theta(x_i, \tilde{y}_i^m)\right)^2\right)$$

Intuitively, it is similar to the traditional Langevin MCMC loss function (4) with the added two terms that equal to the variance of the negative samples' energy values ($\text{Var}[E_\theta] = \mathbb{E}[E_\theta^2] - \mathbb{E}[E_\theta]^2$). Minimizing the energy function's variance effectively makes the resulting EBM close to the uniform distribution.

B. Langevin Correctness

There seemed to have been a small transcription error when the authors were implementing the Langevin step in both math and code. To illustrate, we compare three slightly different formulations existing, with minor adjustments to notation for consistency and equating $E_\theta(y^k) = E_\theta(x, y^k)$. The step sizes used in each are either ϵ, λ, or τ (where $\lambda = \epsilon^2 = 2\tau$), and are ultimately equivalent:

1) Equation 1 in [3] effectively states:

$$\begin{aligned} y^{k+1} &= y^k - \frac{\lambda}{2}\nabla_y E_\theta(y^k) + \psi^k, && \psi^k \sim \mathcal{N}(0, \lambda) \\ &= y^k - \frac{\lambda}{2}\nabla_y E_\theta(y^k) + \sqrt{\lambda}\,\omega^k, && \omega^k \sim \mathcal{N}(0, I) \end{aligned}$$

2) Equation 2 in [6] effectively states:

$$y^{k+1} = y^k - \frac{\epsilon^2}{2}\nabla_y E_\theta(y^k) + \epsilon\,\omega^k, \qquad \omega^k \sim \mathcal{N}(0, I)$$

3) The Euler-Maruyama update rule from [16] effectively states (with a sign-flip):

$$y^{k+1} = y^k - \tau\nabla_y E_\theta(y^k) + \sqrt{2\tau}\,\omega^k, \qquad \omega^k \sim \mathcal{N}(0, I) \tag{8}$$

However, while holding $\sigma = 1$, the equation in Sec. B.3 of [5] effectively states:

$$y^{k+1} = y^k - \frac{\lambda}{2}\nabla_y E_\theta(y^k) + \lambda\,\omega^k, \qquad \omega^k \sim \mathcal{N}(0, I)$$

We believe the transcription error happened where $\mathcal{N}(0, \lambda^2) = \lambda\,\mathcal{N}(0, I)$ was mistakenly equated with $\mathcal{N}(0, \lambda) = \sqrt{\lambda}\,\mathcal{N}(0, I)$.

Anecdotally, we believe that IBC's incorrect Langevin MCMC is closer to gradient descent than true Langevin MCMC, especially when the step size λ is "far" from 1. Because the step sizes tend to be smaller, due to the normalization IBC employs, then the gradient term dominates and the noise term is diminished.

The plots in Fig. 1 show the final 1000 iterations from a total of 4000 iterations for a single chain. The code (and parameters) to generate the results can be found in this Jupyter notebook.

C. Implementation Details

For inclusion in our codebase, we implemented Implicit Behavior Cloning in PyTorch, cross-referencing the published implementation in (which was using TensorFlow v2). We implemented both the DFO and Langevin based policies, and then built upon them for above experiments. We validated our implementation by comparing both overfit (1 episode) and "full" training (2000 episodes) for the 2d and 16d particle environment and ensured that the logged metrics (loss, energies, distances between positive and negative samples, success metrics, etc.) followed similar evolutions along training minibatches.

When we refer to experiments running correct Langevin, we actually use the formulation as stated in (8) rather than (5), meaning the step size indicated is actually $\tau = \frac{\lambda}{2}$. [5]

D. Training Setup

Our primary demonstration environment is the 2d particle environment for its simplicity and speed of training.
For training, we used a very similar setup as those in [5]:

- We use the same normalization for the observation and action space, such that the minimum and maximum in each space occur at $[-1, 1]$, respectively.
- Energy network is a simple MLP, 256 hidden units, 2 layers[6], 0% dropout, ReLU activation, using Keras "normal"-style initializer[7]. For our experiments, we did not need the spectral norm application.
- For negative samples, we use Langevin sampling, either "correct" (8) or "ibc" (6). For speed (both in training and inference), we use 10 iterations.
  - For training, we select $\sigma = 0.1$ with "ibc" to allow more for aggressive convergence with fewer iterations.
  - For inference, we select $\sigma = 0.01$ with "ibc" such that it behaves as gradient descent.
  - For both training and inference, we use both "global" clipping and per-iteration clipping at 25% of the original action space. When drawing initial samples from a uniform distribution, we apply the same 5% margin to both ends of the action space as is done in [5] (thus sampling from $[-1.1, 1.1]$).
  - For the step size (λ for "ibc", τ for "correct"), we use the same polynomial scheduler, starting at step size of 1, decreasing to 0.001, with a power of 2.
  - Due to a minor bug, we did not end up using gradient penalty as indicated in Sec. B.3.1 in [5].
- Training used PyTorch's Adam optimizer with learning rate of 0.001, batch size of 512, step decay learning rate scheduler with gamma of 0.99 decaying every 100 epochs.
- The following distinctions are made for the following trials, with 300 episodes for training:
  - **Ibc** - Running "ibc" Langevin formulation as mentioned above.
  - **Ibc, MAS** - Using Marginal Action Sampler (same MLP architecture, but using 64 hidden units).
  - **Correct, MAS** - Same as prior, but using "correct" Langevin formulation.
  - **Correct, MAS+MaxEnt** - Same as prior, but using maximum entropy as well as an L2 norm on positive energies (anchoring them towards zero).

[5] We used (5) in the paper to provide clearer distinction against (6).
[6] $(n_{in} \times n_{hidden}) \rightarrow (n_{hidden} \times n_{hidden}) \rightarrow (n_{hidden} \times n_{out})$
[7] Linear weights and biases initialized from $\mathcal{N}(0, \sigma)$ for $\sigma = 0.05$