On Fenchel Mini-Max Learning

Chenyang Tao¹, Liqun Chen¹, Shuyang Dai¹, Junya Chen¹,², Ke Bai¹, Dong Wang¹, Jianfeng Feng³, Wenlian Lu², Georgiy Bobashev⁴, Lawrence Carin¹

¹ Electrical & Computer Engineering, Duke University, Durham, NC, USA
² School of Mathematical Sciences, Fudan University, Shanghai, China
³ ISTBI, Fudan University, Shanghai, China
⁴ RTI International, Research Triangle Park, NC, USA
{chenyang.tao, lcarin}@duke.edu

Abstract

Inference, estimation, sampling and likelihood evaluation are four primary goals of probabilistic modeling. Practical considerations often force modeling approaches to make compromises between these objectives. We present a novel probabilistic learning framework, called Fenchel Mini-Max Learning (FML), that accommodates all four desiderata in a flexible and scalable manner. Our derivation is rooted in classical maximum likelihood estimation, and it overcomes a longstanding challenge that prevents unbiased estimation of unnormalized statistical models. By reformulating MLE as a mini-max game, FML enjoys an unbiased training objective that (i) does not explicitly involve the intractable normalizing constant and (ii) is directly amendable to stochastic gradient descent optimization. To demonstrate the utility of the proposed approach, we consider learning unnormalized statistical models, nonparametric density estimation and training generative models, with encouraging empirical results presented.

1 Introduction

When learning a probabilistic model, we are typically interested in one or more of the following operations:

- **Inference**: Represent observation \( x \in \mathbb{R}^p \) with an informative feature vector \( z \in \mathbb{R}^d \), ideally with \( d \ll p \); \( z \) is often a latent variable in a model of \( x \).
- **Estimation**: Given a statistical model \( p_\theta(x) \) for data \( x \), learn model parameters \( \theta \) that best describe the observed (training) data.
- **Sampling**: Efficiently synthesize samples from \( p_\theta(x) \) given learned \( \theta \), with drawn \( x \sim p_\theta(x) \) faithful to the training data.
- **Likelihood evaluation**: With learned \( \theta \) for model \( p_\theta(x) \), calculate the likelihood of new \( x \).

One often makes trade-offs between these goals, as a result of practical considerations (e.g., computational efficiency); see Table S1 in the Supplementary Material (SUPP) for a brief summary. We are particularly interested in the case for which the model \( \tilde{p}_\theta(x) \) is unnormalized; i.e., \( \int \tilde{p}_\theta(x)dx = \frac{1}{Z(\theta)} \neq 1 \), with \( Z(\theta) \) difficult to compute \[15\].

Maximum likelihood estimation (MLE) is widely employed in the training of probabilistic models \[11\]⁵², in which the expected log-likelihood \( \log p_\theta(x) \) is optimized wrt \( \theta \), based on the training examples. For unnormalized model density function \( \tilde{p}_\theta(x) = \exp(-\psi_\theta(x)) \), where \( \psi_\theta(x) \) is the potential function and \( \theta \) are the model parameters, the likelihood is \( p_\theta(x) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x) \). The partition function \( Z(\theta) \) is typically not represented in closed-form when considering a flexible choice of \( \psi_\theta(x) \), such as a deep neural network. This makes the learning of unnormalized models particularly challenging, as the gradient computation requires an evaluation of the integral. In practice, this integral is approximated with averaging over a finite number of Monte Carlo samples. However, using the existing finite-sample Monte Carlo estimate of \( Z_\theta \) will lead to a biased approximation of

the log-likelihood objective (see Section 2.1). This issue is aggravated as the dimensionality of the problem grows.

Many studies have been devoted to addressing the challenge of estimation with unnormalized statistical models. Geyer [23, 24] proposed Markov chain Monte Carlo MLE (MCMC-MLE), which employs a likelihood-ratio trick. Contrastive divergence (CD) [33] directly estimates the gradient by taking MCMC samples. Hyvärinen [56] proposed score matching (SM) to estimate an unnormalized density, bypassing the need to take MCMC samples. Noise contrastive estimation (NCE) learns the parameters for unnormalized statistical models via discriminating empirical data against noise samples [28, 29]. This concept can be further generalized under the Bregman divergence [27]. More recently, dynamic dual embedding (DDE) explored a primal-dual view of MLE [15, 16], while Stein implicit learning (SIL) [46, 41] and kernel score estimation [60] match the landscape of the potential with that of kernel-smoothed empirical observations. However, these approaches are susceptible to poor scalability (SM, MCMC-MLE), biased estimation (CD), and computational (DDE, SIL) and statistical (NCE) efficiency issues.

Concerning design of models that yield realistic drawn samples, considerable recent focus has been placed on implicit generative models [48], which include the generative adversarial network (GAN) [25, 51, 4, 61], the generative moment matching network (GMMN) [24, 19], implicit MLE (IMLE) [59], among others. In this setting one typically doesn’t have an explicit \( p_{\theta}(x) \) or \( p_{\theta}(x) \), and the goal is to build a model of the data generation process directly. Consequently, such schemes typically have difficulty addressing the aforementioned likelihood goal. Additionally, such models often involve training strategies that are challenging due to instabilities or expressiveness, such as adversarial estimation (GAN) and kernelized formulation (GMMN).

For these reasons, likelihood-based models remain popular. Among them variational inference (VI) [6] and generative flows (FLOW) [56, 53] are two of the most promising directions, and have undergone rapid development recently [66]. Despite this progress, challenges remain. The variational bound employed by VI is often not sufficiently tight in practice (undermining the likelihood goal), and there exist model identifiability issues [62]. In FLOW a trade-off has to be made between the computational cost and model expressiveness.

This paper presents a novel strategy for MLE learning for unnormalized statistical models, that allows efficient parameter estimation and accurate likelihood approximation. Importantly, while competing solutions can only yield stochastic upper/lower bounds, our treatment allows unbiased estimation of log-likelihood and model parameters. Further, this setup can be used for effective sampling goals, and it has the ability to perform inference. This work makes the following contributions: (i) Derivation of a mini-max formulation of MLE, resulting in an unbiased log-likelihood estimator directly amenable to stochastic gradient descent (SGD) optimization, with convergence guarantees. (ii) Amortized likelihood estimation with deep neural networks, enabling direct likelihood prediction and feature extraction (inference). (iii) Development of a novel training scheme for latent-variable models, presenting a competitive alternative to VI. (iv) We show that our models compare favorably to existing alternatives in likelihood-based distribution learning, both in terms of model estimation and sample generation.

2 Fenchel Mini-Max Learning

2.1 Preliminaries

Maximum likelihood estimation Given a family of parameterized probability density functions \( \{p_{\theta}(x)\}_{\theta \in \Theta} \) and a set of empirical observations \( \{x_i\}_{i=1}^n \), MLE seeks to identify the most probable model \( \hat{\theta}_{\text{MLE}} \) via maximizing the expected model log-likelihood, i.e., \( \hat{L}(\theta) \triangleq \frac{1}{n} \sum_{i=1}^n \log p_{\theta}(x_i) \). For flexible choices of \( p_{\theta}(x) \), such as an unnormalized explicit-variable model \( p_{\theta}(x) \propto \exp(-v_{\theta}(x)) \) or latent variable model of the form \( p_{\theta}(x) = \int p_{\theta}(x|z)p(z)dz \), direct optimization wrt MLE loss is typically computationally infeasible. Instead, relatively inexpensive likelihood approximations are often used to derive surrogate objectives.

![Figure 1: Comparison of popular likelihood approximations: Monte-Carlo estimator (MC) (e.g., contrastive divergence (CD) [33]), Renyi, importance-weighted ELBO [10], and the proposed FML. Cheap approximations often lead to biased estimate of likelihood, a point FML seeks to fix.](image)
Variational inference Consider a latent variable model \( p_0(x, z) = p_0(x|z)p(z) \). To avoid direct numerical estimation of \( p_0(x) \), VI instead maximizes the variational lower bound to the marginal log-likelihood: \( \text{ELBO}(p_0(x|z), q_\beta(z|x)) = \mathbb{E}_{q_\beta(z|x)} \log \left[ \frac{p_0(x|z)}{q_\beta(z|x)} \right] \), where \( q_\beta(z|x) \) is an approximation to the true posterior \( p_0(z|x) \). This bound tightens as \( q_\beta(z|x) \) approaches the true posterior \( p_0(z|x) \). For estimation, we seek parameters \( \theta \) that maximize the ELBO, and the commensurately learned parameters \( \beta \) are used in a subsequent inference task with new data. However, with such learning, samples drawn \( x \sim p_\theta(x|z) \) with \( z \sim p(z) \) may not be as close to the training data as desired.

Adversarial distribution matching Adversarial learning \([25, 26]\) exploits the fact that many discrepancy measures have a dual formulation \( D(p_\theta, p_\mu) = \max_{D} \{ \mathbb{V}_{D}(p_\theta, p_\mu; D) \} \), where \( \mathbb{V}_{D}(p_\theta, p_\mu; D) \) is a variational objective that can be estimated with samples from the true distribution \( p_\mu(x) \) and the model distribution \( p_\theta(x) \), and \( D(x) \) is an auxiliary function commonly known as the critic (or discriminator).

Recalling the Fenchel conjugate of \( D(z) \) makes unbiased estimation of unnormalized statistical models a long-standing challenge posed to the statistical community, especially for high-dimensional problems \([27]\).

Fenchel conjugacy Let \( f(t) \) be a proper convex, lower-semicontinuous function; then its convex conjugate function \( f^*(v) \) is defined as \( f^*(v) = \sup_{t \in D(f)} \{ tv - f(t) \} \) where \( D(f) \) denotes the domain of function \( f \). \( f^* \) is also known as the Fenchel conjugate of \( f \), which is again convex and lower-semicontinuous. The Fenchel conjugate pair \((f, f^*)\) are dual to each other, in the sense that \( f^{**} = f \), i.e., \( f(t) = \sup_{v \in D(f^*)} \{ vt - f^*(v) \} \). As a concrete example, \(-\log(p(x)), \log(1 - \exp(-u))\) gives such a pair, as we exploit in the next section.

Algorithm 1 Fenchel Mini-Max

Biased finite sample Monte-Carlo for unnormalized statistical models For unnormalized statistical model \( p_\theta(x) = \exp(-\psi_\theta(x)) \), the naive Monte-Carlo estimator for the log-likelihood is given by \( \hat{\psi}_\theta(x) = -\psi_\theta(x) - \log \hat{Z}_\theta \), where \( \hat{Z}_\theta = \frac{1}{m} \sum_{j=1}^{m} \exp(-\psi_\theta(X'_j)) \) is the finite-sample estimator for the normalizing constant \( Z_\theta = \int e^{-\psi_\theta(x')} dx' \), with \( \{X'_j\} \) i.i.d. uniform samples on \( \Omega \). Via the Jensen’s inequality (i.e., \( \mathbb{E}_X [\log f(X)] \leq \log(\mathbb{E}_X [f(X)]) \)), it is readily seen that \( \mathbb{E}_{X'_j} [\log \hat{Z}_\theta] \leq \log(\mathbb{E}_{X} [\hat{Z}_\theta]) = \log Z_\theta \), which implies the naive MC estimator gives an upper bound of the log-likelihood, i.e., \( \mathbb{E}_{X'_j} [\hat{\psi}_\theta(x)] \geq \log p_\theta(x) \). The inability to take infinite samples makes unbiased estimation of unnormalized statistical models a long-standing challenge posed to the statistical community, especially for high-dimensional problems \([27]\).

2.2 Mini-Max formulation of MLE for unnormalized statistical models For unnormalized statistical model \( p_\theta(x) = \exp(-\psi_\theta(x)) \), we rewrite model log-likelihood as

\[
\log p_\theta(x) = \log \frac{e^{-\psi_\theta(x)}}{\int e^{-\psi_\theta(x')} dx'} = -\log \left( \int e^{\psi_\theta(x) - \psi_\theta(x')} dx' \right) \tag{1}
\]

Recalling the Fenchel conjugate of \(-\log(t)\), we have \(-\log(t) = \max_u \{-u - \exp(-u)t + 1\}\), and the optimal value of \( u \) is \( u^*_x = \log(t) \). Plugging this into (1) yields the following expression

\[
-\log p_\theta(x) = \min_{u_x} \left\{ u_x + \exp(-u_x) \int e^{\psi_\theta(x) - \psi_\theta(x')} dx' - 1 \right\} \tag{2}
\]

Since \( u^*_x = \log \left( \int e^{\psi_\theta(x) - \psi_\theta(x')} dx' \right) = -\log p_\theta(x) \), we have \( \exp(-u^*_x) = p_\theta(x) \). Consequently, the auxiliary dual variable \( u \) is an estimate of the negative log-likelihood. The key insight here is that we have turned the numerical integration problem into an optimization problem. This may seem like a step backward at first sight, as we are still summing over the support and we have a dual variable to optimize. The payoff is that we can now sidestep the log term and estimate the log-likelihood in
an unbiased manner using finite MC samples, a major step-up over existing estimators. As argued below and verified experimentally, this extra optimization can be executed efficiently and robustly. This implies we are able to more accurately estimate unnormalized statistical models at a comparable budget, without compromising training stability.

Denote \( I(x; \psi_g) = \int e^{\psi_g(x)} - \psi_g(x') \) \( dx' \). To estimate \( I(x; \psi_g) \) more efficiently, we may introduce a proposal distribution \( q(x) \) with tractable likelihood and leverage an importance weighted estimator:

\[
I(x; \psi_g) = \mathbb{E}_{q \sim q_{x}} [\exp(\psi_g(x) - \psi_g(x') - \log q(x'))].
\]

We discuss the practical choice of proposal distribution in more detail in Section 2.4. Putting everything together, we have the following mini-max formulation of MLE for unnormalized statistical models:

\[
\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} \left\{ -\min_{u} \left\{ \sum_{i} J_{\theta}(x_i; u_i, \psi) \right\} \right\},
\]

where \( J_{\theta}(x; u, \psi) \defeq u + \exp(-u) I(x; \psi_g) \).

In practice, we can model all \( \{u_i\} \) with only one additional free parameter as \( u_{\theta}(x) = \psi_g(x) + b_{\theta} \), where \( b_{\theta} \) models the log-partition function, i.e., \( b_{\theta} \defeq \log Z_{\theta} \); we make explicit here that \( u \) is a function of \( \theta \), i.e., \( u_{\theta}(x) \). Note that \( b_{\theta} \) is the log-partition parameter to be learned, that minimizes the objective if and only if it equals the true log-partition. Although model parameters \( \theta \) are shared between updating \( \theta \) and \( u \) in \( \psi \), the update of \( u \) corresponds to refining the update of the log-partition function \( b_{\theta} \) for fixed \( \theta \), followed by updating \( \theta \) with \( b \) fixed; we have isolated learning the partition function (the mini-\( u \)-step) and the model parameters (the mini-\( \theta \) step). We call this new formulation Fenchel Mini-Max Learning (FML), and summarize its pseudocode in Algorithm 1. For complex distributions, we also optimize the proposal \( q(x) \) to enable efficient & robust learning with the importance weighted estimator.

Considering the form of \( J(x; u, \psi_g) \), one may observe that the learning signal comes from contrasting data samples \( x_i \) with a random draw \( X' \) under the current model potential \( \psi_g(x) \) (e.g., the term \( \psi_g(x_i) - \psi_g(X') \)). Figure 1 compares our FML to other popular likelihood approximation schemes. Unlike existing solutions, FML targets the exact likelihood without explicitly using finite-sample estimator for the partition function. Instead, FML optimizes an objective where the untransformed integral directly appears, which leads to an unbiased estimator provided the minimization is solved accurately.

### 2.3 Gradient analysis of FML

To further understand the workings of FML, we inspect the gradient of model parameters. In classical MLE learning, we have \( \nabla \log p_{\theta}(x) = \frac{\nabla p_{\theta}(x)}{p_{\theta}(x)} \). That is to say, in MLE the gradient of the likelihood is normalized by the model evidence. A key observation is that, while \( \nabla p_{\theta}(x) \) is difficult to compute, because of the partition function, we can easily acquire an unbiased gradient estimate of the inverse likelihood \( \frac{1}{p_{\theta}(x)} \) using Monte-Carlo samples,

\[
\nabla \left\{ \frac{1}{p_{\theta}(x)} \right\} = \nabla \left\{ \int \exp(\psi_{\theta}(x) - \psi_{\theta}(x')) \right\} = \int \nabla \left\{ \exp(\psi_{\theta}(x) - \psi_{\theta}(x')) \right\} \right\} dx' \tag{4}
\]

which only differs from \( \nabla \log p_{\theta}(x) \) by a factor of negative inverse likelihood

\[
\nabla \left\{ \frac{1}{p_{\theta}(x)} \right\} = -\frac{\nabla p_{\theta}(x)}{(p_{\theta}(x))^2} = -\frac{\nabla \log p_{\theta}(x)}{p_{\theta}(x)}. \tag{5}
\]

Now considering the gradient of FML, we have

\[
\nabla J_{\theta}(x; \hat{u}_x, \psi) = -\nabla \left\{ \exp(-\hat{u}_x) \int e^{\psi_{\theta}(x) - \psi_{\theta}(x')} \right\} dx' \approx -\hat{p}_{\theta}(x) \nabla \left\{ \frac{1}{p_{\theta}(x)} \right\} = \frac{\hat{p}_{\theta}(x)}{p_{\theta}(x)} \nabla \log p_{\theta}(x) \approx \nabla \log p_{\theta}(x), \tag{6}
\]

where \( \hat{u}_x \) denotes an approximate solution to the Fenchel maximization game \( \hat{x} \) and \( \hat{p}_{\theta} \defeq \exp(-\hat{u}_x) \) is an approximation of the likelihood based on our previous analysis. We denote \( \xi \defeq \frac{\hat{p}_{\theta}(x)}{p_{\theta}(x)} \), and refer to \( \log \xi \) as the approximation error. If this approximation \( \hat{p}_{\theta} \) is sufficiently accurate then \( \xi \approx 1 \), which implies the FML gradient is a good approximation to the gradient of true likelihood.

\[\text{In practice, we find that instead of separated updates, simultaneous gradient descent of } \theta \text{ and } b \text{ also works well.}\]
We formalize this as the generalized SGD problem described below. When we model the auxiliary variable as \( u(x) = \psi_\theta(x) + b \), then the FML gradient \( \nabla J_\theta(x; u, \psi) \) differs from \( \nabla \log p_\theta(x) \) by a common multiplicative factor \( \xi = \exp(b - b_\theta) \) for all \( x \in \Omega \). Next we show SGD is insensitive to this approximation error; FML still converges to the same solution of MLE even if \( \xi \) deviates from 1 differently at each iteration.

2.4 Choice of proposal distribution

Like all importance-weighted estimators, the efficiency of FML critically depends on the choice of proposal \( q(x) \). A poor match between the proposal and integrand can lead to extremely high variance \([52]\), which compromises learning. In order to keep the variance in check, a general guiding principle for choosing a good \( q(x) \) is to make it close to the data distribution \( p_\mu \). Note this practice differs from the optimal minimal variance proposal, which is proportional to the integrand. However, it does not need to constantly update the proposal to adapt to the current parameter, which brings both robustness and computational savings. To obtain such a static proposal matched to the data distribution, we can pre-train a parameterized tractable sampler \( q_\phi(x) \) with empirical data samples by maximizing the empirical model log-likelihood \( \sum_i \log q_\phi(x_i) \), with \( \phi \) parameterizing the proposal. Note that we only require the proposal \( q(x) \) to be similar to the data distribution, using a rough approximation to facilitate the learning of an unnormalized model that more accurately characterize the data. The proposal does not necessarily need to capture every minute detail of the target distribution, as such simpler models are generally preferable for better computational efficiency, provided adequate approximation and coverage can be achieved. Popular choice of parameterized proposal include generative flows \([53]\) or mixture of Gaussians \([44]\). We leave a more detailed specification of our treatment to the Supplementary Material (SUPP).

2.5 Convergence results

In modern machine learning, first order stochastic gradient descent (SGD) is a popular choice, and in many cases the only feasible approach, for large-scale problems. In the case of MLE, let \( h(\theta; \omega) \) be an unbiased stochastic gradient estimator for \( \hat{L}(\theta) \), i.e., \( \mathbb{E}_{\omega \sim p(\omega)}[h(\theta; \omega)] = \nabla \hat{L}(\theta) \). Here we have used \( \omega \sim p(\omega) \) to denote the source of randomness for \( h(\theta; \omega) \). SGD finds a solution by using the following iterative procedure \( \theta_{t+1} = \theta_t + \eta_t h(\theta_t; \omega_t) \), where \( \{ \eta_t \} \) is a pre-determined sequence commonly known as the learning-rate schedule and \( \{ \omega_t \} \) are iid draws from \( p(\omega) \). Then under common technical assumptions on \( h(\theta; \omega) \) and \( \{ \eta_t \} \), if there exists only one unique minimizer \( \theta^* \) then the SGD solution \( \hat{\theta}_{\text{SGD}} \triangleq \lim_{t \to \infty} \theta_t \) will converge to it \([57]\).

Now consider FML’s naive stochastic gradient estimator \( \hat{h}(\theta; \omega) = e^{u(x)} \nabla \exp(\psi_\theta(x) - \psi_\theta(x')) \), where \( X \sim \hat{p}_\mu(X') \sim \mathcal{U}(\Omega) \); the contrast \( \psi_\theta(x) - \psi_\theta(x') \) between real and synthetic data is evident. Based on the analysis from the last section, we have the decomposition \( \hat{h}(\theta; \omega) = \xi h(\theta; \omega) \), where \( h(\theta; \omega) \) is the unbiased stochastic gradient term and \( \xi \) relates to the (unknown) approximation error. Using the same learning rate schedule, we are updating model parameter with effective random step-sizes \( \hat{\eta}_t \triangleq \xi_t \eta_t \) relative to SGD with MLE, where \( \xi_t \) depends on the current approximation error. We formalize this as the generalized SGD problem described below.

**Problem 2.1** (Generalized SGD). Let \( h(\theta; \omega) \sim p(\omega) \) be an unbiased stochastic gradient estimator for objective \( f(\theta) \), \( \{ \eta_t > 0 \} \) is the fixed learning rate schedule, \( \{ \xi_t > 0 \} \) is the random perturbations to the learning rate. We want to solve for \( \nabla f(\theta) = 0 \) with the iterative scheme \( \theta_{t+1} = \theta_t + \hat{\eta}_t h(\theta_t; \omega_t) \), where \( \{ \omega_t \} \) are iid draws and \( \hat{\eta}_t = \eta_t \xi_t \) is the randomized learning rate.

**Proposition 2.2** (Generalized stochastic approximation). Under the standard regularity conditions listed in Assumption D.1 in the SUPP, we further assume \( \sum_i \mathbb{E}[\hat{\eta}_t] = \infty \) and \( \sum_i \mathbb{E}[\hat{\eta}_t^2] < \infty \). Then \( \theta_n \to \theta^* \) with probability 1 from any initial point \( \theta_0 \).

**Remark.** This is a straightforward generalization of the Robbins-Monro theory. The original proof still applies by simply replacing expectation wrt the deterministic sequence \( \{ \eta_t \} \) with the randomized sequence \( \{ \hat{\eta}_t \} \). Assumptions \( \sum_i \mathbb{E}[\hat{\eta}_t] = \infty \) and \( \sum_i \mathbb{E}[\hat{\eta}_t^2] < \infty \) can be satisfied by saying \( \{ \log \xi_t \} \) is bounded. The \( \omega \)-updates used in FML force \( \{ \log \xi_t \} \) to stay close to zero, thereby enforcing the boundedness condition. Although such assumptions are too strong for deep neural nets, empirically FML converges to very reasonable solutions. We discuss more general theories in the SUPP.

**Corollary 2.3.** Under the assumptions of Prop. 2.2, FML converges to \( \hat{\theta}_{\text{MLE}} \) with SGD.

3 FML for Latent Variable Models and Sampling Distributions

3.1 Likelihood-free modeling & latent variable models

One can reformulate generative adversarial networks (GANs) \([25]\) into a latent-variable model, by introducing arbitrarily small Gaussian perturbations. Specifically, \( X' = G_\phi(Z) + \sigma \zeta \), where
\( \zeta \sim \mathcal{N}(0, 1) \) is standard Gaussian, and \( \sigma \) is the noise standard deviation. This gives the joint likelihood \( p_{\theta}^n(x, z) = \mathcal{N}(G_\theta(z), \sigma^2)p(z) \). It is well known the marginal likelihood \( p_{\theta}^n(x) \) converges to \( p_\theta(x) \) as \( \sigma \) goes to zero \( \mathbb{R} \). As such, we can always use a latent-variable model to approximate the likelihood of an implicitly defined distribution \( p_\theta(x) \), which is easy to sample from. It also allows us to associate generator parameters \( \theta \) to likelihood-based losses.

### 3.2 Fenchel reformulation of marginal likelihood

Replacing the log term with its Fenchel dual, we have the following alternative expression for the marginal likelihood: \( \log p_\theta(x) = \log \left( \int p_\theta(x, z) \, dz \right) = \min_{u_x} \left\{ u_x + \exp(-u_x) I(x; p_\theta) - 1 \right\} \), where \( I(x; p_\theta) \triangleq \int p_\theta(x, z) \, dz \). Note that, different from the last section, here estimate \( \hat{u}_x \) provides a direct approximation to the marginal likelihood \( \log p_\theta(x) \) rather than its negative. By analogy with variational inference (VI), an approximate posterior \( q_\beta(z|x) \) can also be introduced, assuming the role of proposal distribution for the integral term. Model parameter \( \theta \) can be learned via the following mini-max setup

\[
\max_{\theta} \left\{ \min_{u} \left\{ E_{X \sim p_\theta} \left[ u_X + \exp(-u_X) I(X; p_\theta, q_\beta) \right] \right\} \right\},
\]

where \( I(x; p_\theta, q_\beta) \triangleq \mathbb{E}_q \left[ \log \frac{p_\theta(x, z)}{q_\beta(x|z)} \right] \) is the importance weighted estimator with proposal \( q_\beta(z|x) \), and \( u \in \mathbb{R}^n \) is a vector modeling the marginal likelihood \( \log p_\theta(x_i) \) for each training example \( x_i \), with \( u_i \). A good proposal encodes the association between \( x \) and \( z \) (this is expanded upon in the SUPP); as such, we also refer to \( q_\beta \) as the inference distribution. We will return to the optimization of inference parameter \( \beta \) in Section 3.3. Our analysis from Sections 2.3 to 3.2 also applies in the latent variable case and is not repeated here. To further stabilize the training, annealed training can be considered, replacing integrand \( \mathbb{E}_q \left[ \log \frac{p_\theta(x, z)}{q_\beta(x|z)} \right] \) with \( \mathbb{E}_q \left[ \log \frac{p_\theta(x, z)}{q_\beta(x|z)} \right] \) as in Neal \( \mathbb{R} \). Here \( \{\tau_i\} \) is the annealing schedule, monotonically increasing wrt time \( t \) going from \( \tau_0 = 0 \) to \( \tau_\infty = 1 \).

### 3.3 Optimization of inference distribution

The choice of proposal distribution \( q_\beta(z|x) \) is important for the statistical efficiency of FML. To address this issue, we propose to encourage more informative proposal via regularizing the vanilla FML objective. In particular, we consider regularizing with the mutual information \( I_p \triangleq \mathbb{E}_p \left[ \log \frac{p(X,Z)}{p(X)p(Z)} \right] \). Let us denote our model distribution \( p_\theta(x, z) \) as \( p \) and the approximate joint \( q_\beta(x, z) \triangleq q_\beta(z|x)p_\theta(x) \) as \( q \), and the respective mutual information are denoted as \( I_p \) and \( I_q \). It is necessary to regularize both \( I_p \) and \( I_q \), since \( I_q \) directly encourage more informative proposal, while the “informativeness” is upper bounded by \( I_p \). In other words, this encourages the proposal to approach the posterior.

Direct estimation of \( I_p \) and \( I_q \) is infeasible, due to the absence of analytical expressions for the marginals \( p_\theta(x) \) and \( q_\beta(z) \). Instead, we use their respective lower bounds \( D_p(\theta, \beta) = \mathbb{E}_{(X,Z) \sim p_\theta} \left[ \log q_\beta(Z|X) \right] \) and \( D_q(\beta|\theta) = \mathbb{E}_{(X,Z) \sim q_\beta} \left[ \log p_\theta(X|Z) \right] \) as our regularizer (see the SUPP for details). Note these bounds are tight as the proposal \( q_\beta(z|x) \) approaches the true posterior \( p_\theta(z|x) \) (Lemma 5.1, Chen et al. \( \mathbb{R} \)). We then solve the following regularized mini-max game

\[
\max_{\theta, \beta} \left\{ \min_{u} \left\{ \mathcal{J}(u, \theta, \beta) \right\} - \lambda_p D_p(\theta, \beta) - \lambda_q D_q(\beta|\theta) \right\}.
\]

Here the nonnegative \( \lambda_p, \lambda_q \) are the regularization strengths, and we have used notation \( D_q(\beta|\theta) \) to highlight the fact this term does not contribute to the gradient of model parameter \( \theta \). Solving \( \mathbb{R} \) using standard simultaneous gradient descent/ascent as in standard GAN training is observed to be efficient and stable in practice.

### 3.4 Amortized inference of marginal likelihoods

Unlike the explicit likelihood case from Section 2, the marginal likelihoods \( \{\log p_\theta(x_i)\} \) are no longer directly related by an explicit potential function \( \psi_\phi(x) \). Individually update \( u_i \) for each sample \( x_i \) is computationally inefficient: (i) it does not scale to large datasets; (ii) parameters are not shared across samples; (iii) it does not permit efficient prediction of the likelihood at test time for a new observation \( x_{\text{new}} \). Motivated by its success in variational inference, we propose to employ the amortization technique to tackle the above issues \( \mathbb{R} \). When optimizing some objective function with distinct parameters \( \zeta_i \) associated with each training example \( x_i \), e.g., \( \mathcal{L}(\theta, \zeta_i) = \sum \ell_\theta(x_i, \zeta_i) \), amortized learning replaces these parameters with a parameterized function \( \zeta_\phi(x) \) with \( \phi \) as the amortization parameters. The optimization is then carried out wrt the amortized objective \( \mathcal{L}(\theta, \phi) = \sum \ell_\theta(x_i, \zeta_\phi(x_i)) \) instead. Contextualized under our FML, we amortize the marginal likelihood estimate \( \{u_i\} \) with
a parameterized function \( u_\phi(x) \), and optimize \( \max_\theta \{ -\min_b \{ \mathcal{F}(\psi, b; \theta) \} \} \) instead of \( \int \).

Since \( \mathbb{E}_{p_\theta} [\log p_\theta] = \min_\theta \{ \mathbb{E}_{p_q} [\mathcal{F}(u_\phi; p_\theta, q_\beta)] \} \leq \min_\theta \{ \mathbb{E}_{p_q} [\mathcal{F}(u_\phi; x; p_\theta, q_\beta)] \} \), amortized latent FML effectively optimizes an upper bound of the likelihood loss. This bound tightens as the function family \( u_\phi \) becomes more expressive, which makes expressive deep neural networks an appealing choice for \( u_\phi \) \[53\]. To further improve parameter efficiency, we note parameter \( \psi \) can be shared with the proposal parameter \( \beta \) used by \( q_\beta(z|x) \).

### 3.5 Sampling From Unnormalized Distribution

There are problems for which we are given an unnormalized distribution \( p_{\psi^*}(x) \propto \exp[-\psi^*(x)] \) and no data samples; we would like to model \( p_{\psi^*}(x) \) in the sense that we’d like to efficiently sample from it. This problem arises, for example, in reinforcement learning \[31\], among others. To address this problem under FML, we propose to parameterize a sampler \( X = G_\theta(Z), Z \sim p(z) \) and a nonparametric potential function \( \psi_\theta(x) \) \[4\]. FML is used to estimate the model likelihood via solving

\[
\max_\psi \{- \min_b \{ \mathcal{F}(\psi, b; \theta) \}, \mathcal{F}(\psi, b; \theta) = \mathbb{E}_Z p(z) \{ \mathcal{F}(G_\theta(Z), u_{\psi, b}, \psi) \}\}
\]

where \( u_{\psi, b}(x) = \psi_\theta(x) + b \) is our estimate for \(- \log p_\theta(x) \) implicitly defined by \( G_\theta(z) \).

To match model samples to the target distribution, \( G_\theta(z) \) is trained to minimize the KL-divergence

\[
\text{KL}(p_\theta \mid \mid p_{\psi^*}) = \mathbb{E}_{X \sim p_\theta} [\log p_\theta(X) - \log p_{\psi^*}(X)] = \mathbb{E}_{X \sim p_\theta} [\log p_\theta(X) + \psi^*(X)] + \log Z_{\psi^*}.
\]

Since the last term is independent of model parameter \( \theta \), we obtain the KL-based training objective

\[
\mathcal{J}_{\text{KL}}(\theta; \psi, b; \psi^*) = \mathbb{E}_{Z \sim p(z)} \{ \psi^*(G_\theta(Z)) - u_{\psi, b}(G_\theta(Z)) \} \text{ by replacing } \log p_\theta(x) \text{ with our FML estimate.}
\]

Due to the dependence of \( u_\psi(x) \) on \( \theta \), the final learning procedure is

\[
[\psi_t, b_t] = [\psi_{t-1}, b_{t-1}] - \eta_t \nabla_{[\psi, b]} \mathcal{F}(\psi_{t-1}, b_{t-1}; \theta_t), \theta_{t+1} \leftarrow \theta_t - \eta_t \nabla_\theta \mathcal{J}_{\text{KL}}(\theta; \psi_t, b_t, \psi^*).
\]

### 4 Related Work

**Fenchel duality** In addition to optimization schemes, the Fenchel duality also finds successful applications in probabilistic modeling. Prominent examples include divergence minimization \[4\] and likelihood-ratio estimation \[50\], and more recently adversarial learning \[51\]. In discrete learning, Fagan and Iyengar \[20\] employed it to speedup extreme classification. To the best of the authors’ knowledge, Fenchel duality has not been applied previously to likelihoods with latent variables.

**Nonparametric density estimation** To combat the biased estimation of the partition function, Burda et al. \[9\] proposed a conservative estimator, which partly alleviates this issue. Parallel to our work, Dai et al. \[16\] explored Fenchel duality in the setting of MLE for an unnormalized statistical model estimation, under the name dynamics dual embedding (DDE), which seeks optimal embedding in the space of probability measures. The authors used parameterized Hamiltonian flows for distribution embeddings, which limits its scalability and expressiveness. In particular, DDE fails if the search space does not contain the target distribution, while our formulation only requires the support of the proposal distribution to cover that of the target.

**Adversarial distribution learning** The proposed FML framework is complementary to the development of GANs. FML prioritizes the learning of a potential function, while GANs have focused on the training of a sampler. Both schemes are derived via learning by contrast. Notably \( f \)-GANs contrast the difference between likelihoods under respective models, while our FML contrasts data samples with proposal samples under the current model potential. Synergies can be explored between the two schemes.

**Approximate inference** Compared with VI, FML optimizes a direct estimate of the marginal likelihood instead of a variational bound. While tighter bounds can be achieved for VI via importance re-weighting \[10\], flexible posteriors \[47\] and alternative evidence scores \[42\], these strategies do not necessarily improve performance \[55\]. Another fundamental difference is that while VI discards all conditional likelihoods after the ELBO evaluation, FML consolidates them into an estimate of the marginal likelihood through SGD.

**Sampling unnormalized potentials** This is one of the fundamental topics in statistics and computer science \[45\]. Recent studies have explored the use of deep neural sampler for this purpose: Feng et al. \[21\] trains the sampler with kernel Stein variational gradients, and Li et al. \[38\] adversarially updates the sampler based on the adaptive contrast technique \[47\]. FML provides an expressive, scalable and

\footnote{With slight abuse of notation, we assume \( \psi_\theta(x) \) is parameterized by \( \psi \) to avoid notation collision with sampler \( G_\theta(z) \).}
Table 1: Quantitative evaluation on toy models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameter estimation error ↑</th>
<th>Likelihood consistency score ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC</td>
<td>3.46 (\pm 0.19)</td>
<td>0.961 (\pm 0.008)</td>
</tr>
<tr>
<td>SM (\text{[54]})</td>
<td>7.79 (\pm 0.25)</td>
<td>(\times) (\times) (\times) (\times) (\times)</td>
</tr>
<tr>
<td>NCE (\text{[28]})</td>
<td>3.88 (\pm 0.25)</td>
<td>(\times) 0.973 0.755 0.183 0.436 0.265</td>
</tr>
<tr>
<td>KEF (\text{[59]})</td>
<td>6.59 (\pm 0.31)</td>
<td>(\times) 0.944 0.830 0.426 0.520 0.186</td>
</tr>
<tr>
<td>DDE (\text{[16]})</td>
<td>6.59 (\pm 0.31)</td>
<td>(\times) 0.944 0.830 0.426 0.520 0.186</td>
</tr>
<tr>
<td>FML (ours)</td>
<td>3.05 1.9 2.59 1.13 1.27</td>
<td>0.974 0.901 0.562 0.731 0.782</td>
</tr>
</tbody>
</table>

Figure 2: FML predicted likelihood using nonparametric potentials.

5 Experiments

To validate the proposed FML framework and benchmark it against state-of-the-art methods, we consider a wide range of experiments, using synthetic and real-world datasets. All experiments are implemented with Tensorflow and executed on a single NVIDIA TITAN X GPU. Details of the experimental setup are provided in the SUPP, due to space limits, and our code is from [https://www.github.com/chenyang-tao/FML](https://www.github.com/chenyang-tao/FML). For the evaluation metrics reported, ↑ indicates a higher score is considered better, and vice versa with ↓. Our goal is to verify FML works favorably or similarly compared with competing solutions under the same setup, not to beat state-of-the-art results.

### 5.1 Estimating unnormalized statistical models

We compare FML with competing solutions on parameter estimation and likelihood prediction with unnormalized statistical models. We report × if a method is unable to compute or failed to reach a reasonable result. Grid search is used for KDE to optimize the kernel bandwidth.

#### Parameter estimation for unnormalized models

We first benchmark the performance on parameter estimation with a number of representative toy models, including both continuous distributions with varying dimensionality (see SUPP for details). The exact parametric form of the potential function is given, and the task is to estimate the parameter values that generate the samples. We use 1,000 and 5,000 samples, respectively, for training and evaluation. To assess performance, we repeat each experiment 10 times and report the mean absolute error \(\|\hat{\theta} - \theta^*\|_1\), where \(\hat{\theta}\) and \(\theta^*\) denote the parameter estimate and ground-truth, respectively. We benchmark FML against naive Monte-Carlo, score matching, noise contrastive estimation and dual dynamics embedding, with results reported in Table 1. FML provides comparable, if not better, performance on all the models considered.

#### Nonparametric likelihood prediction

In the absence of an explicit parametric model of the likelihood, a deep neural network is used as a nonparametric model of the potential. To evaluate model performance, we consider the likelihood consistency score, defined as the correlation between the learned nonparametric potential and the ground truth potential, \(\text{corr}(\log p_{\theta^*}(X), \log p_{\hat{\theta}}(X))\), where the expectation is taken wrt ground-truth samples. The results are summarized in Table 1. In Figure 2, we also visualize the nonparametric FML estimates of the likelihood compared with ground truth. Note SM proved computationally unstable in all cases, and DDE has to be trained with a smaller learning rate, due to stability issues.

In addition to the toy examples, we also evaluate the proposed FML on real datasets from the UCI data repository \([17]\). To evaluate model performance, we randomly split the data into ten folds, and use seven of them for training and three of them for evaluation. To cope with the high-dimensionality of the data, we use a GMM proposal for both NCE and FML. The averaged \(\log\)-likelihood on the test set is reported in Table 2 and the proposed FML shows an advantage over its counterparts.

### 5.2 Latent variable models and generative modeling

Our next experiment considers FML-based training for latent variable models and generative modeling tasks. In particular, we directly benchmark FML against the VAE \([37]\), for modeling complex distributions, such as images and natural language, for real-world applications. We focus on evaluating the model's ability to (efficiently) synthesize realistic samples. Additionally, we also demonstrate how FML can assist the training of generative adversarial nets by following the variational annealing setup.
We have developed a scalable and flexible learning scheme for probabilistic modeling. Rooted which is also verified based on our human evaluation (see SUPP).

The evolution of policy rewards wrt training iterations is provided in Figure 4(b-d), and overlays samples from the FML-trained policy network on continuous control tasks, namely swimmer, hopper and reacher, defined in OpenAI gym \cite{openai2018gym} and rllab \cite{rllab}, environments, with results summarized in Figure 4. Figure 4(a) overlays samples from the FML-trained policy network on the potential of the model estimated optimal policy, verifying FML’s capability to capture complex multi-modal distributions. The evolution of policy rewards wrt training iterations is provided in Figure 4(b-d), and FML-based policy updates improve on original SVGD updates.

**5.3 Sampling unnormalized distributions**

Our final experiment considers an application in reinforcement learning (RL) with FML-trained neural sampler. We benchmark the effectiveness of our FML-based sampling scheme described in Sec 3.5 by comparing it with the SVGD sampler used in state-of-the-art soft Q-learning implementation \cite{huangetal2017}. We examine the performance on three continuous control tasks, namely swimmer, hopper and reacher, defined in OpenAI gym \cite{openai2018gym} and rllab \cite{rllab} environments, with results summarized in Figure 4. Figure 4(a) overlays samples from the FML-trained policy network on the potential of the model estimated optimal policy, verifying FML’s capability to capture complex multi-modal distributions. The evolution of policy rewards wrt training iterations is provided in Figure 4(b-d), and FML-based policy updates improve on original SVGD updates.

**6 Conclusion**

We have developed a scalable and flexible learning scheme for probabilistic modeling. Rooted in classical MLE learning, our solution handles inference, estimation, sampling and likelihood evaluation in a unified framework, without major compromises. Empirical evidence verified the proposed method delivers competitive performance on a wide range of tasks.
Acknowledgements
The authors would like to thank the anonymous reviewers for their insightful comments. This research was supported in part by DARPA, DOE, NIH, ONR, NSF and RTI Internal research & development funds. J Chen was partially supported by China Scholarship Council (CSC). W Lu and J Feng were supported by the Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and ZJLab.

References


A. Notations and Assumptions

Capital letters (e.g., $X$) are used to denote random variables, and $\mathbb{E}_{X \sim p}[f(X)]$ denotes the expectation of function $f(x)$ wrt distribution $p(x)$. $\nabla f_\theta$ denotes the gradient of function $f_\theta(x)$ wrt parameters $\theta$, gradient wrt spatial parameters will be denoted as $\nabla_x f_\theta$. $\Delta_x$ is used to denote the Laplacian operator wrt spatial variable $x$. We say $p_\theta(x)$ is an explicit likelihood if its evaluation does not involve marginalization over latent variables. To simplify discussions, we always assume continuous variables, and probability measures of interest are defined on a compact domain $\Omega \subset \mathbb{R}^d$. Without loss of generality, $\Omega$ is assumed to have unit volume.

B. Comparison of Popular Probabilistic Modeling Procedures

We summarized popular probabilistic modeling schemes’ strength wrt the goals of inference, sampling, likelihood evaluation and scalability in Table S1. Note that this table is composed based on standard understand of these methods. For some specialized applications some of these methods can be extended beyond the limitations outlined in this table.

C. Biased Likelihood Estimation with Finite Sample Monte-Carlo Estimation

Consider the following naive Monte-Carlo estimator for the log-likelihood

$$\log \hat{p}_\theta(x) = -\psi_\theta(x) - \log \hat{Z}_\theta,$$

where

$$\log \hat{Z}_\theta = \log \left( \frac{1}{m} \sum_{j=1}^{m} \exp(-\psi_\theta(X'_j)) \right),$$


\begin{table}[h]
\centering
\caption{Comparison of popular probabilistic modeling procedures.}
\begin{tabular}{|l|c|c|c|c|}
\hline
Model & Inference & Sampling & Likelihood & Scalability \\
\hline
FML (ours) & Yes & Yes & Estimate & Good \\
CD & Yes & Yes & No & Good \\
SM & No & No & No & Poor \\
NCE & No & No & Estimate & Depends \\
KEF & No & No & No & Poor \\
DDE & No & Yes & Exact & Low \\
VI & Yes & Yes & Bound & Good \\
Flow & No & Yes & Exact & Tricky \\
Stein & No & Yes & No & Medium \\
GAN & No & Yes & No & Good \\
\hline
\end{tabular}
\end{table}

is the finite sample estimator for the log-partition, with $X'_j$ sampled uniformly from $\Omega$. Via the Jensen inequality (i.e., $\mathbb{E}_X [\log f(X)] \leq \log(\mathbb{E}_X [f(X)])$), it is easy to see

$$\mathbb{E}_{X_j} [\log \hat{Z}_\theta] \leq \log(\mathbb{E}_{X_j} [\hat{Z}_\theta]) = \log Z_\theta, (3)$$

which implies the naive MC estimator actually gives an upper bound of the log-likelihood

$$\mathbb{E}_{X_j} [\log \hat{p}_\theta(x)] \geq \log p_\theta(x). (4)$$

To partly alleviate this issue, \textsuperscript{?} considered an alternative estimator that lower bounds the log-likelihood.

D. Technical Assumptions for Robbins-Monro Stochastic Approximation

**Assumption D.1.** (Standard regularity conditions for Robbins-Monro stochastic approximation).

A1. $h(\theta) \triangleq \mathbb{E}_\omega[h(\theta; \omega)]$ is Lipschitz continuous;

A2. The ODE $\dot{\theta} = h(\theta)$ has a unique equilibrium point $\theta^*$, which is globally asymptotically stable;

A3. The sequence \{$\theta_t$\} is bounded with prob 1;

A4. The noise sequence \{$\omega_t$\} is a martingale difference sequence;

A5. For some finite constants $A$ and $B$ and some norm $\| \cdot \|_2$ on $\mathbb{R}^d$, $\mathbb{E}[\|\omega_t\|^2] \leq A + B \|\theta_t\|^2$ a.s. $\forall t \geq 1$. 

\textsuperscript{1}Equal contribution \textsuperscript{1}Electrical & Computer Engineering, Duke University, Durham, NC, USA \textsuperscript{2}School of Mathematical Science, Fudan University, Shanghai, China \textsuperscript{3}ISTBI, Fudan University, Shanghai, China \textsuperscript{4}RTI International, Research Triangle Park, NC, USA. Correspondence to: Chenyang Tao <chenyang.tao@duke.edu>, Lawrence Carin <lcarin@duke.edu>.
Remark. In the context of stochastic optimization, the globally asymptotic stability can be implied, for example, when \( f(\theta) \) is strict convex (recall \( h(\theta) = \nabla f(\theta) \)).

E. Proof of Proposition 3.2

Proof. We only need to verify that convergence still holds in probability when Robbins-Monro condition is satisfied in expectation. Without loss of generality suppose
\[
\langle \theta - \theta^*, h(\theta) \rangle \leq 0
\]
holds for all \( \theta \). Define a Markov chain \( \theta_t \) by taking \( \theta_1 \) to be an arbitrary constant and define
\[
\theta_{t+1} - \theta_t = -\xi_t \eta_t h(\theta_t).
\]
Let \( b_n = \mathbb{E} \left[ \| \theta_{t+1} - \theta^* \|_2^2 \right] \). We shall find conditions under which \( \lim_{n \to \infty} b_n = 0 \), no matter what the initial value \( \theta_1 \), which implies the convergence in probability of \( \theta_t \) to \( \theta^* \). From (6), we have:
\[
b_{n+1} = \mathbb{E} \left[ \| \theta_{t+1} - \theta^* \|_2^2 \right] = \mathbb{E} \left[ \| \theta_{t+1} - \theta_t \|_2^2 + \langle \theta_t - \theta^*, h(\theta_t) \rangle \right]
\]
\[
= b_n + 2\mathbb{E} [\langle \theta_{t+1} - \theta_t, \theta_t - \theta^* \rangle] + \mathbb{E} \left[ \| h(\theta_{t+1}) \|_2^2 \right]
\]
\[
= b_n - 2\mathbb{E} \left[ \xi_t \eta_t \mathbb{E} \left[ \langle \theta_t - \theta^*, h(\theta_t) \rangle \right] \right]
\]
\[
+ \mathbb{E} \left[ \xi_t^2 \eta_t^2 \mathbb{E} [\| h(\theta_t) \|_2^2] \right]
\]
Set \( d_n = \mathbb{E} [\langle \theta_t - \theta^*, h(\theta_t) \rangle] \), \( e_t = \mathbb{E} \left[ \| h(\theta_t) \|_2^2 \right] \), we can write
\[
b_{n+1} - b_n = -2\mathbb{E} \left[ \xi_t^2 \eta_t^2 \right] d_n + \mathbb{E} \left[ \xi_t^2 \eta_t \right] e_t.
\]
Note from (5), \( d_n \geq 0 \), while from the assumption on function \( h(\cdot) \), 0 \( \leq e_t < \infty \), together with \( \sum \mathbb{E} \left[ \xi_t^2 \eta_t^2 \right] < \infty \), we have \( \sum \mathbb{E} \left[ \xi_t^2 \eta_t^2 \right] e_t \) converges. Summing (7) gives
\[
b_{t+1} = b_1 + \sum_{j=1}^t \mathbb{E} \left[ \xi_j^2 \eta_j^2 \right] e_j - 2 \sum_{j=1}^t \mathbb{E} \left[ \xi_j \eta_j \right] d_j
\]
Since \( b_{n+1} \geq 0 \), we obtain
\[
\sum_{j=1}^t \mathbb{E} \left[ \xi_j \eta_j \right] d_j \leq \frac{1}{2} \left[ b_1 + \sum_{j=1}^t \mathbb{E} \left[ \xi_j^2 \eta_j^2 \right] e_j \right] < \infty
\]
It follows from (8) that \( \lim_{n \to \infty} b_n = b \) exists, \( b \) equals to 0 is proved in Robbins and Monro’s paper, more details can be found in (Robbins & Monro, 1951).

F. Proof of Corollary 3.3

Proof. Based on analysis from Section 3.2, FML executes SGD wrt MLE gradient with randomly perturbed step size \( \tilde{\eta}_t \). Then result directly follows from Proposition 3.1.

G. More general results on FML convergence

The following results relaxes the strong assumptions on the uniqueness of global minimizer, proving that under SGD FML training also reaches a stationary point of the ground-truth likelihood function as standard MLE training does. This results applies more generally to modern learning frameworks such as deep neural net. We note that due to the stochasticity and the nonlinearity involved, both FML and MLE may reach different solutions in separate runs for deep nets.

Assumption G.1. (Weaker regularity conditions for generalized Robbins-Monro stochastic approximation).

B1. The objective function \( f(\theta) \) is second-order differentiable.

B2. The objective function \( f(\theta) \) has a Lipschitz-continuous gradient, i.e., there exists a constant \( L \) satisfying
\[
-L \leq \nabla^2 f(\theta) \leq L
\]
B3. The noise has a bounded variance, i.e., there exists a constant \( \sigma > 0 \), satisfying
\[
\mathbb{E} \left[ \| h(\theta_t; \omega_t) - \nabla f(\theta_t) \|_2^2 \right] \leq \sigma^2.
\]

Theorem G.2. Under the technical conditions listed in Assumption G.1, the SGD solution \( \{ \theta_t \}_{t=0} \) updated with generalized Robbins-Monro sequence \( \tilde{\eta}_t; \sum_t \mathbb{E}[\tilde{\eta}_t^2] = \infty \) and \( \sum_t \mathbb{E}[\tilde{\eta}_t^2] < \infty \) converges to a stationary point of \( f(\theta) \) with probability 1 (equivalently, \( \mathbb{E} \left[ \| \nabla f(\theta_t) \|_2^2 \right] \to 0 \) as \( t \to \infty \)).

Proof. Define a Markov chain \( \theta_t \) by taking \( \theta_1 \) to be an arbitrary constant vector:
\[
\theta_{t+1} - \theta_t = -\xi_t \eta_t h(\theta_t; \omega_t) \equiv -\tilde{\eta}_t h_t(\theta_t)
\]
where \( \tilde{\eta}_t = \xi_t \eta_t \) and \( h_t(\theta_t) = h(\theta_t; \omega_t) \).

By Taylor’s theorem, the objective will be
\[
\bar{f}(\theta_{t+1}) = f(\theta_t - \tilde{\eta}_t h_t(\theta_t)) = f(\theta_t) - \tilde{h}_t(\theta_t)^T \nabla f(\theta_t) + \frac{\tilde{\eta}_t^2}{2} h_t(\theta_t)^T \nabla^2 f(\theta_t) h_t(\theta_t)
\]
Taking the expected value,
\[
\mathbb{E} \left[ \bar{f}(\theta_{t+1}) | \theta_t \right] \leq f(\theta_t) - \mathbb{E}[\tilde{\eta}_t] \mathbb{E} \left[ h_t(\theta_t)^T \nabla f(\theta_t) | \theta_t \right]
\]
\[
+ \frac{L}{2} \mathbb{E}[\tilde{\eta}_t^2] \mathbb{E} \left[ \| h_t(\theta_t) \|_2^2 | \theta_t \right]
\]
\[
\leq f(\theta_t) - \left( \mathbb{E}[\tilde{\eta}_t] - \frac{L}{2} \mathbb{E}[\tilde{\eta}_t^2] \right) \| \nabla f(\theta_t) \|_2^2
\]
\[
+ \frac{\sigma^2 L}{2} \mathbb{E}[\tilde{\eta}_t^2] = 0
\]
We further define the mutual information can be readily verified:

$$\mathbb{E}[\tilde{\eta}] = \frac{1}{2} \mathbb{E}[\tilde{\eta}],$$

for $t \geq T_0$ which is guaranteed by the convergence of $\sum_t \mathbb{E}[\tilde{\eta}^2]$, then

$$\mathbb{E}[f(\theta_{t+1})] \leq f(\theta_t) + \frac{1}{2} \mathbb{E}[\tilde{\eta}] \| \nabla f(\theta_t) \|^2 + \frac{\sigma^2 L}{2} \mathbb{E}[\tilde{\eta}^2].$$

Now taking the full expectation

$$\mathbb{E}[f(\theta_{t+1})] \leq \mathbb{E}[f(\theta_t)] - \frac{1}{2} \sum_{t=T_0}^{T-1} \mathbb{E}[\tilde{\eta}] \mathbb{E}[\| \nabla f(\theta_t) \|^2] + \frac{\sigma^2 L}{2} \sum_{t=T_0}^{T-1} \mathbb{E}[\tilde{\eta}^2],$$

and summing up from $T_0$ to $T$,

$$\mathbb{E}[f(\theta_T)] \leq \mathbb{E}[f(\theta_{T_0})] - \frac{1}{2} \sum_{t=T_0}^{T-1} \mathbb{E}[\tilde{\eta}] \mathbb{E}[\| \nabla f(\theta_t) \|^2] + \frac{\sigma^2 L}{2} \sum_{t=T_0}^{T-1} \mathbb{E}[\tilde{\eta}^2].$$

rearranging the terms,

$$\frac{1}{2} \sum_{t=T_0}^{T-1} \mathbb{E}[\tilde{\eta}] \mathbb{E}[\| \nabla f(\theta_t) \|^2] \leq \mathbb{E}[f(\theta_{T_0})] - \mathbb{E}[f(\theta_T)] + \frac{\sigma^2 L}{2} \sum_{t=T_0}^{T-1} \mathbb{E}[\tilde{\eta}^2].$$

Let $T \to \infty$, and notice that $\sum_t \mathbb{E}[\tilde{\eta}] = \infty$, $\sum_t \mathbb{E}[\tilde{\eta}^2] < \infty$, then

$$\sum_{t=T_0}^{\infty} \mathbb{E}[\tilde{\eta}] \mathbb{E}[\| \nabla f(\theta_t) \|^2] < \infty,$$

Hence $\mathbb{E}[\| \nabla f(\theta_t) \|^2] \to 0$ as $t \to \infty$. \hfill \Box

**H. Rate-Distortion Theory and Mutual Information Bounds**

We further define the $q$-rate score $R_q$ and $q$-distortion score $D_q$ as

$$R_q \triangleq \mathbb{E}_{(X,Z) \sim q}[\log g(Z|X) - \log \rho(Z)],$$

$$D_q \triangleq -\mathbb{E}_{(X,Z) \sim q}[\log \rho(X|Z)],$$

and similarly define $\rho$-rate score $R_p$ and $\rho$-distortion score $D_p$. Here $\{R_p, R_q, D_p, D_q\}$ are collectively referred to as the rate-distortion scores. We note the distortion score $D$ differs from the distortion regularizer $D_q$ defined in main text, as a compromise to avoid notational clutter. The following link between rate-distortion scores and mutual information can be readily verified:

**Algorithm 1 Amortized FML for Latent Variable model**

Learning rate schedule $\{\eta\}$, annealing schedule $\{\tau_t\}$, regularization strength $\lambda$

Initialize parameters $\theta, \beta, \phi$

% Optional pre-training with VAE

for $t = 1, 2, \ldots$ do

Sample $\{x_{t,j}\}_{j=1}^m \sim \hat{p}_0(x), \{z_{t,j}\}_{j=1}^m \sim q_\beta(z|x_{t,j}),$

$u_{t,j} = u_\phi(x_{t,j}),$

$I_t = \exp\{\tau_t \log p_\theta(x_{t,j}|z_{t,j}) + \log q_\beta(z_{t,j}) \} - \log q_\beta(z_{t,j}|x_{t,j})$

$J_t = \sum_j \{u_{t,j} + \exp(-u_{t,j}) I_t\}$

$D_{p,t} = \sum_j \log p_\theta(x_{t,j}|z_{t,j})$

$D_{q,t} = \sum_j \log q_\beta(z_{t,j}|x_{t,j})$

$D_t = D_{p,t} + D_{q,t}$

$\psi$-update: $\phi = \phi - \eta_t \nabla_{\phi} J_t$

end for

**Proposition H.1** (Rate-distortion inequalities (Berger, 1971; Alemi et al., 2018)).

$$H(p_d) - D_q \leq I_q \leq R_q, \quad H(p_\rho) - D_\rho \leq I_\rho \leq R_\rho. \quad (10)$$

These bounds are tight as the proposal $q_\beta(z|x)$ approaches the true posterior $p_\alpha(z|x)$ (Lemma 5.1, Chen et al. (2016)).

**I. Algorithm of Amortized FML for Latent Variable Models**

The pseudocode for latent variable FML is summarized in Algorithm 1.

**J. Connection to Langevin Gradient Flow**

We remark our procedure described in Section 5 actually simulates the discrete Langevin gradient flow (Chen et al., 2018)

$$x_{t+\Delta t} \leftarrow x_t - \Delta t \nabla_x \{\log p_\theta(x_t) - \log p_{\phi^*}(x_t)\} \quad (11)$$

to solve the Fokker-Plank system

$$\partial_t p_{\theta_t} + \nabla_x \cdot (p_{\theta_t} \nabla_x \log \frac{p_{\theta_t}}{p_{\phi^*}}) = 0. \quad (12)$$

It is well known that the solution $p_{\theta_t}(x)$ of (12) converges to $p_{\phi^*}(x)$ when $t \to \infty$ under mild technical assumptions (Jordan et al., 1998).

**K. Empirical Evaluation of FML’s Consistency**

We experimentally verify the proposed FML is a consistent estimator, that is to say FML estimate converges to ground truth.
L. Competing Solutions

For completeness, we briefly describe competing solutions used in this study.

L.1. Score matching (SM)

Hyvärinen (2005) proposed score matching (SM) to estimate an unnormalized density. In particular, score matching directly models the (data) score function \( \nabla_x \log p_d(x) \), and seek to minimize the score discrepancy metric

\[
\mathbb{F}(p_d, p_\theta) \triangleq \frac{1}{2} \mathbb{E}_{X \sim p_d(x)} [ \| \nabla_x \log p_d(X) - \nabla_x \log p_\theta(X) \|_2^2 ] = \mathbb{E}_{X \sim p_\theta(x)} [ \Delta_x \log p_\theta(X) + \frac{1}{2} \| \nabla_x \log p_\theta(X) \|_2^2 ] + C, \tag{13}
\]

where \( C \) is a constant wrt \( \theta \). Note (13) does not involve the partition function \( Z(\theta) \), and other than the constant term it only depends on \( p_\theta(x) \) through the expectation. As such, it can be easily estimated with a Monte Carlo average. A major drawback for score matching in a modern differentiable learning setting is that, the computation involves taking second-order derivatives (if the score function is directly modelled), which is costly in practice.

L.2. Noise contrastive estimation (NCE)

Noise contrastive estimation (NCE) is a technique used to estimated the parameters for unnormalized statistical models (Gutmann & Hyvärinen, 2010; 2012), i.e. models with density function known up to a normalization constant. Let \( p_\theta(x) = \tilde{p}_\theta(x)/Z(\theta) \) the model density function, where \( \tilde{p}_\theta(x) \) is the unnormalized pdf parameterized by \( \alpha \) and \( Z(\theta) = \int \tilde{p}_\theta(x) \, dx \) is the partition function (normalizing constant). Without loss of generality, we assume only the knowledge of \( \tilde{p}_\theta(x) \) and \( Z(\theta) \) is intractable. To address the intractable normalizing constant, we introduce an additional parameter \( c \in \mathbb{R} \) for it, and define (unnormalized) distribution \( p_\theta(x) = \tilde{p}_\theta(x)/C \), where \( \theta^* = (\theta, c) \) and \( C = \exp(c) \). Note that \( p_\theta(x) \) does not necessarily integrate to one. Let \( p_d(x) \) be the unknown data distribution, and further introduce a contrastive distribution \( q(x) \), also known as the noise distribution, which is both tractable and easy to sample from. Let \( X_n = \{ x_i \}_{i=1}^n \) and \( Y_n = \{ y_i \}_{i=1}^n \) be the respective empirical samples from data and contrastive distribution, then the contrastive objective is given by

\[
J_{\text{NCE}}(\theta) = \frac{1}{2n} \sum_i \{ \log h(x_i; \theta) + \log(1 - h(y_i; \theta)) \}, \tag{14}
\]

where

\[
\begin{align*}
    h(u; \theta) &= \sigma(r(u; \theta)), \\
    r(u; \theta) &= \log p_\theta(u) - \log q(u) \\
    &= \log \tilde{p}_\theta(u) - \log q(u) - c,
\end{align*}
\]

and \( \sigma(t) = 1/(1 + \exp(-t)) \) is the sigmoid function. This objective function is essentially the likelihood function for the class label of the mixture distribution \( \frac{1}{2} [p_d + p_\theta] \), and the NCE estimate of \( \theta \) is given by \( \hat{\theta} = \arg \max J_{\text{NCE}}(\theta) \), and we denote the corresponding model density by \( \hat{p}_{\text{NCE}}(x) = p_\hat{\theta}(x) \). NCE follows the idea of “learning by comparison”, it learns the properties of the data in terms of a statistical model by discriminating the samples between data and noise. It is known that when the data distribution \( p_d(x) \) is contained in the family of model distributions \( Q = \{ p_\theta(x) \}_{\theta \in \Theta} \), then \( \hat{p}_{\text{NCE}}(x) \) is a consistent estimator for \( p_d(x) \).

L.3. Dynamics dual embedding (DDE)

Dynamics dual embedding (DDE) considers the primal-dual view of MLE (Dai et al., 2018). In particular, DDE exploited the following fact:

\[ \text{Theorem L.1 (Theorem 1, (Dai et al., 2018)).} \]

Let \( H(q) \triangleq - \int q(x) \log q(x) \, dx \), we have

\[
Z(\theta) = \max_{q \in \mathcal{P}} \{ \langle q, \tilde{p}_\theta \rangle + H(q) \} \tag{17}
\]

\[
p_\theta = \arg \max_{q \in \mathcal{P}} \{ \langle q, \tilde{p}_\theta \rangle + H(q) \} \tag{18}
\]

where \( \mathcal{P} \) denotes the space of distributions, \( \langle f, g \rangle \triangleq \int \Omega f(x)g(x) \, dx \) is the regular \( L^2 \) inner product.

Plugging the Fenchel dual formulation of the partition \( Z(\theta) \) into the likelihood estimator renders MLE into a saddle-point optimization problem:

\[
\max_{\theta \in \Theta} \mathcal{L}(\theta) \leftrightarrow \max_{f \in \mathcal{F}} \min_{q \in \mathcal{P}} J(\theta, q) \tag{19}
\]

where

\[
J_{\text{DDE}}(\theta, q) \triangleq \mathbb{E}_{X \sim p_d}[\tilde{f}_\theta(X)] - \mathbb{E}_{X' \sim q}[\tilde{f}_{\theta}(X')] - H(q) \tag{20}
\]

is the DDE objective. In the original paper, Hamiltonian flow had been used to parameterize the dual embedding distribution \( q \).

L.4. Kernel exponential family estimation (KEF)

Kernel exponential family estimation (KEF) considers the problem of nonparametric density estimation in infinite dimensional space (Sriperumbudur et al., 2017). More specifically, KEF seeks a solution of the following form

\[
p_\psi \propto \exp(-\psi_d(x))p_d(x), \tag{21}
\]
where $p_0(x)$ is considered as prior regularization and $\psi(x)$ is constrained to an RKHS $\mathcal{H}_\psi$. To match the empirical distribution $p_d$, KEF optimizes the following regularized score discrepancy:

$$J_{\text{KEF}}(\theta) \triangleq F(p_d, p_\theta) + \lambda \|\psi\|^2_{\mathcal{H}}, \quad (22)$$

where $\lambda > 0$ is the regularization strength and $\| \cdot \|_{\mathcal{H}}$ is the RKHS norm. Analytical solution can be derived with provable convergence rates.

### L.5. Stein variational gradient descent (SVGD)

Stein variational gradient descent (SVGD) (Liu & Wang, 2016) considers the problem of steepest descent in the space of probability distributions wrt KL-divergence, with descent directions constrained in certain reproducing kernel Hilbert space (RKHS). Formally, define the Stein operator $A_p$ for $d$-dim vector function $\phi(x) \in \{C^1(\Omega)\}^d$ wrt distribution $p(x)$ as

$$A_p(\phi) \triangleq \phi(x) \nabla_x \log p(x)^T + \nabla_x \phi(x), \quad (23)$$

and the Stein discrepancy $S(q, p)$ between distribution $q$ and $p$ as

$$S(q, p) = \max_{\phi \in \mathcal{F}} \{ E_{X \sim q} [\text{tr} (A_p \phi(X))^2] \}, \quad (24)$$

where $\mathcal{F}$ denotes some function space. Let $\kappa(x, x')$ be a semi-positive definite function known as the kernel, which defines RKHS $\mathcal{H} \triangleq \text{Span}\{\kappa(\cdot, x) : x \in \Omega\}$. Let $q_{\phi}(x)$ be the distribution defined by the following transport operator to the mass of distribution $q(x)$:

$$T_{c,\phi}(x) = x + c \phi(x). \quad (25)$$

Then it can be shown that the steepest descend wrt KL($q_{\phi} \parallel p$) from the unit ball in $\mathcal{H}$ is given by

$$\phi^*_{q, p}(x) = \mathbb{E}_{X \sim q} [A_p \kappa(X, \cdot)] \quad (26)$$

with $\nabla_{KL}(q_{\phi} \parallel p)|_{c=0} = -S(q, p)$. In amortized SVGD (Wang & Liu, 2016), one optimizes the following objective to match model distribution $p_\theta$ (implicitly defined by the generator $G_\theta(Z)$, $Z \sim p(z)$) to the unnormalized target distribution $p_{\hat{\psi}}$:

$$\min_{\theta} J_{\text{SVGD}}(\theta) \triangleq \mathbb{E}_{Z \sim p(z)} [\{G_\theta(Z) - \text{StopGrad}(G_\theta(Z) + \eta_t \phi^*_{p_\theta, \hat{\psi}}(G_\theta(Z)))\}^2], \quad (27)$$

where $\eta_t$ denotes the learning rate.

### L.6. Generative flow (FLOW)

Generative flows (FLOW) consider modeling distribution $p_\theta$ with a generator $G_\theta(z)$ with non-degenerate tractable Jacobian (Tabak et al., 2010). More specifically, if $G_\theta(z)$ is invertible wrt $z$, then

$$p_\theta(z) = p(z) |\operatorname{det}(\nabla_z G_\theta^{-1}(x))|, \quad (28)$$

where $x \triangleq G_\theta(z)$. While the constraint imposed is very limiting, model flexibility can be significantly improved by stacking such simpler transformations $G_{\theta,l}(z_{l-1})$ (Rezende & Mohamed, 2015), e.g.,

$$\log p_\theta(x_z) = \log p(z) - \sum_{m=1}^{M} \log |\operatorname{det}(\nabla z_{l-1} G_{\theta,l})| \cdot (29)$$

Different flow implementations differs in their specific choices for $G_{\theta,l}(z)$.

### L.7. Kernel density estimation (KDE)

**Kernel density estimation** (KDE) is a classical solution to the problem of nonparametric estimation of likelihood, which exploits the idea of smoothing the data with a kernel. Formally, let $\kappa(x)$ be a smoothing kernel satisfying $\int \kappa(x) dx = 1$, then the simplest KDE likelihood estimate is given by

$$p_{h}^{\text{KDE}} = \frac{1}{n h^d} \sum_{i=1}^{n} \kappa \left( \frac{x - x_i}{h} \right), \quad (30)$$

where $h > 0$ is commonly known as the bandwidth parameter. Like almost all kernel-based solution, the choice of bandwidth parameter $h$ and smooth kernel $\kappa$ are critical. Isotropic Gaussian rbf is the most popular choice for kernel, and standard practices for bandwidth selection include cross-validation based estimate and rule-of-thumb estimator $h_j = (\frac{4}{\pi^2})^{1/(d+4)} \sigma_j$, where $h_j$ denotes the dimension specific bandwidth and $\sigma_j$ is the standard deviation for the $j$-th dimension.

### L.8. Naive Monte-Carlo (MC)

See our discussion in Section C.

### M. Validation of the normalizing constant

In order to verify the correctness of estimated normalizing constant from FML (and NCE), we use the following estimators: (i) Hamiltonian annealed importance sampling (HAIS) (Sohl-Dickstein & Culpepper, 2012); and (ii) standard importance sampling with GMM proposal. Our implementation of HAIS is modified from the tensorflow implementation found in https://github.com/JohnReid/HAIS. Note sometimes the HAIS estimator will encounter numerical issues when the nonparametric estimate is not sufficiently smooth, in which cases we switch to (ii). We confirmed FML training yields accurate estimate of the normalizing constant. For HAIS estimator (i), we use $3k$ chains with $5k$ steps. For standard IW estimator (ii), we use 50 component Gaussian and draw $50k$ samples.
We summarized basic info for all UCI datasets considered in this study in Table S2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Dimension</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>6</td>
<td>1358</td>
</tr>
<tr>
<td>Wine-red</td>
<td>8</td>
<td>1458</td>
</tr>
<tr>
<td>Wine-white</td>
<td>8</td>
<td>4502</td>
</tr>
<tr>
<td>HTRU2</td>
<td>8</td>
<td>17898</td>
</tr>
</tbody>
</table>

N. Toy Model Experiments

We used the KEF implementation from https://github.com/karlnapf/kernel_exp_family. We used our own implementation of SM, NCE and DDE. For DDE, we replace the Hamiltonian flow used in the original paper with a more expressive MAF flow. To estimate the partition function, we use a 50k sample MC estimator. For the parameter estimation task, we rescale the results by a factor of 100 to facilitate reading.

The exact mathematical form of the toy models we considered and the parameter specifications used are summarized below.

- **banana**: \( \frac{1}{2}((x_1 - (x_2/\kappa))^2/\sigma_1^2 + ((x_2 - \mu_2))^2/\sigma_2^2) \);
- **kidney**: \( \frac{1}{2}((||x| - \mu_1||/\sigma_1)^2 - \log(\exp(-.5 * ((x_1 - \mu_2)/\sigma_2)^2)) + \exp(-.5 * ((x_1 + \mu_2)/\sigma_2)^2)); \)
- **rings**: Cat([25 \times 4]), Cat-1N(0, \sigma_0^2), Cat - i : N(||x||; \mu_i, \sigma_i^2);
- **river**: \( -\log(a_1(x) + a_2(x)) \), where \( a_1(x) = \exp(-.5 * ((x_2 - w_1(x, \sigma_{w,1}))/\sigma_4)^2), \)
\( a_2(x) = \exp(-.5 * ((x_2 - w_1(x, \sigma_{w,1})) + w_3(x, \sigma_{w,3, \mu_3})/\sigma_3^2) \);
- **wave**: \( -\log(a_3(x) + a_2(x)) \), where \( a_3(x) = \exp(-.5 * ((x_2 - w_1(x, \sigma_{w,1}))/\sigma_3^2), \)
\( a_2(x) = \exp(-.5 * ((x_2 - w_1(x, \sigma_{w,1})) + w_2(x, \sigma_{w,3, \mu_3})/\sigma_3^2) \);

where \( w_3(x; \mu, \sigma) = 3 \ast \text{sigmoid}((x_1 - \mu)/\sigma), w_1(x; \sigma) = \sin(2\pi^2 \mu_2). \) For **banana**, we use \( \mu_2 = 0, \sigma_1^2 = 1, \sigma_2^2 = 2, \sigma_3 = 0.35, \sigma_4 = 0.4 \) and \( \kappa = 2. \) For **kidney**, **river** and **wave** we set the parameters according to Rezende & Mohamed (2015), for **rings** we set \( r_i = i \) and \( \sigma_i^2 = 0.2. \)

O. UCI Data Experiments

We summarized basic info for all UCI datasets considered in this study in Table S2.

**Preprocessing.** We removed all categorical variables and normalized each dimension to zero mean and unit variance. Entries with extreme values or missing values were removed from our analysis.

For KDE, we use the default implementation from the scipy package (scipy.stats.gaussian_kde). Since these datasets are all high dimensional, naive uniform proposal distribution is bound to fail. In this study we first used isotropic Gaussian mixture model to fit data, then the learned Gaussian mixture model (GMM) is used as the proposal distribution for NCE and FML. We use the GMM implementation from scikit-learn package (sklearn.mixture.GaussianMixture). We specify the GMM with 50 components and full covariance, unless this choice yields severe overfit or underfit, which is then handled on a case-by-case basis. For the flow model, we used a 4-bijsector MAF model with shift_and_scale transformations, each block has 2 hidden layers with size 256. As standard practice, permutation layers are inserted to avoid degeneracy. Our flow model is implemented with tensorflow probability library package (tensorflow_probability).

For FML and NCE, we used 3 layer feed-forward neural net to model the nonparametric potential. Each layer has 64 hidden units.

P. VAE Image Data Experiments

We summarized the image datasets in Table S5 and the network architectures used for respective datasets in Tables S3 and S4. As in standard VAE implementation, we used the *logit* model instead of Gaussian model at pixel level. We fixed our annealing factor to \( \tau = 0.1 \) in our experiments, which keeps all diagnostic statistics we used in a reasonable range (which indicates our FML is working properly, we omit details here). The results reported are from our unregularized FML implementation, regularized FML implementation show a similar trend, with improved sampling efficiency (results not shown). We use 10 latent dims for MNIST and 64 latent dims for CelebA.

Q. GAN Image Data Experiments

To investigate how FML learning can assist the training of likelihood-free models such as GAN, we adopted the variational annealing framework (Tao et al., 2019) to regularize GAN training with FML-learned likelihood estimate. Specifically, we first encode image data using an autoencoder and then use FML to estimate its likelihood, e.g., training GAN with \( L_{VA} = L_{GAN} + \lambda log \hat{p}_\theta \), where \( L_{GAN} \) is the standard GAN loss and \( \lambda \) is the regularization parameter. We compareFML-based likelihood regularization with vanilla GAN and denoising feature matching (DFM) GAN, which leverages a denoising auto-encoder as score estimator (Alain & Bengio, 2014) to attain the likelihood signal. We
### Table S3. MNIST experiment network architecture.

<table>
<thead>
<tr>
<th>Network</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder</td>
<td>conv2d(unit=32, kernel=5, stride=2) + BN + ReLU</td>
</tr>
<tr>
<td></td>
<td>conv2d(unit=64, kernel=5, stride=2) + BN + ReLU</td>
</tr>
<tr>
<td></td>
<td>fc(unit=1024) + BN + ReLU</td>
</tr>
<tr>
<td>Decoder</td>
<td>fc(unit=1024) + BN + ReLU</td>
</tr>
<tr>
<td></td>
<td>fc(unit=64<em>7</em>7) + BN + ReLU</td>
</tr>
<tr>
<td></td>
<td>reshape to 7 × 7 × 64</td>
</tr>
<tr>
<td></td>
<td>deconv(unit=64, kernel=5, stride=2) + BN + ReLU</td>
</tr>
<tr>
<td></td>
<td>deconv(unit=64, kernel=1, stride=2) + BN + Sigmoid</td>
</tr>
<tr>
<td>u-net</td>
<td>Same as Encoder net.</td>
</tr>
</tbody>
</table>

### Table S4. CelebA experiment network architecture.

<table>
<thead>
<tr>
<th>Network</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder</td>
<td>conv2d(unit=32, kernel=5, stride=2) + BN + ReLU</td>
</tr>
<tr>
<td></td>
<td>conv2d(unit=64, kernel=5, stride=2) + BN + ReLU</td>
</tr>
<tr>
<td></td>
<td>fc(unit=1024) + BN + ReLU</td>
</tr>
<tr>
<td>Decoder</td>
<td>fc(unit=1024) + BN + ReLU</td>
</tr>
<tr>
<td></td>
<td>fc(unit=64<em>7</em>7) + BN + ReLU</td>
</tr>
<tr>
<td></td>
<td>reshape to 7 × 7 × 64</td>
</tr>
<tr>
<td></td>
<td>deconv(unit=64, kernel=5, stride=2) + BN + ReLU</td>
</tr>
<tr>
<td></td>
<td>deconv(unit=64, kernel=1, stride=2) + BN + Sigmoid</td>
</tr>
<tr>
<td>u-net</td>
<td>Same as Encoder net.</td>
</tr>
</tbody>
</table>

### Table S5. Summary of image datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Dim</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>28 × 28</td>
<td>55k</td>
<td>10k</td>
</tr>
<tr>
<td>CelebA</td>
<td>64 × 64 × 3</td>
<td>180k</td>
<td>20k</td>
</tr>
<tr>
<td>Cifar10</td>
<td>32 × 32</td>
<td>60k</td>
<td>-</td>
</tr>
</tbody>
</table>

We evaluated model performance on Cifar10 with IS and FID with fixed positive annealing ($\lambda = 1$, results reported in main text), and studied the effect of regularization strength $\lambda$ on MNIST using IS (see Figure S1). We used the codebase from the DFM paper for the Cifar10 experiment and implemented our own MNIST experiment.

### R. Language Data Experiments

We summarized the language datasets in Table S7 and the network architectures used for respective datasets in Table S6.

For text generation task, we use EMNLP2017 WMT News dataset and MS COCO dataset. EMNLP News dataset consist of 278686 training sentences, 10000 testing sentences, with vocabulary size 5728. MSCOCO contains 120000 and 10000 sentences for training and testing respectively, vocabulary size = 27842. Our model consists of a 3-layer CNN encoder and a LSTM decoder for both datasets.

### S. Reinforcement Learning Experiments

#### S.1. Soft Q-learning

Reinforcement learning seeks to maximize some reward function $r(s, a)$ wrt actions $a$ drawn from the policy distribution $\pi(a|s)$, where $s$ denotes the state. Maximal entropy
Algorithm 2 FML Soft Q-learning

Require: Create replay memory $D = \emptyset$; Initialize policy network parameters $\theta$, FML network parameters $\psi, \mu, \mathcal{Q}$ network parameters $\phi$; Assign target parameters: $\overline{\theta} \leftarrow \theta, \overline{\psi} \leftarrow \phi$. The number of samples for each distribution $M$.

1: for each epoch do
   2:      for each $\tau$ do
      3:         % Collect experience
      4:            Sample an action for $s_t$ using $g^\theta$: $a_t \leftarrow g^\theta(\xi; s_t)$, where $\xi \sim \mathcal{N}(0, I)$.
      5:            Sample next state and reward from the environment: $s_{t+1} \sim P_s$ and $r_t \sim P_r$.
      6:            Save the new experience in the replay memory: $D \leftarrow D \cup \{s_t, a_t, r_t, s_{t+1}\}$
      7:      % Sample a minibatch from the replay memory
      8:         $\{(s^{(i)}_t, a^{(i)}_t, r^{(i)}_t, s^{(i)}_{t+1})\}_{i=0}^{n} \sim D$.
      9:      % Use FML to update $\psi$ and $\mu$ network
      10:     Sample actions for each $s^{(i)}_t$ from the stochastic policy via $a^{(i,j)}_t = f^\phi(\xi^{(i,j)}, s^{(i)}_t; \theta)$, where $\{\xi^{(i,j)}\}_{j=0}^{M} \sim \mathcal{N}(0, I)$
      11:     Follow the Algorithm.1 to estimate $\log P_{\psi,\mu}(a^{(i,j)}_t|s^{(i)}_t)$ and update $\psi, \mu$
      12:     % Update Q value network
      13:     Sample $\{a^{(i,j)}_t\}_{j=0}^{M} \sim q_\alpha$ for each $s^{(i)}_{t+1}$.
      14:     Compute the soft Q-values $Q^\phi(a_{t+1}, s_{t+1})$ and calculate the loss $L_\phi = \|Q^\phi(a_{t+1}, s_{t+1}) + r_t - Q^\phi(a_t, s_t)\|_2$
      15:     Compute gradient of Q-network and update $\phi$
      16:     % Update policy network via KL Divergence
      17:     Calculate the KL Divergence, $L_\theta = KL(P_{\psi,\mu}(a^{(i,j)}_t|s^{(i)}_t)||Q(a^{(i,j)}_t|s^{(i)}_t))$
      18:     Compute gradient of policy network and update $\theta$
   2:     end for
   3: end for

Table S6. Summary of language datasets

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>Encoder</th>
<th>Decoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder 3-layer CNN</td>
<td>LSTM</td>
<td>LSTM</td>
</tr>
</tbody>
</table>

Table S7. Summary of language datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Vocab</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMT News</td>
<td>5,728</td>
<td>278k</td>
<td>10k</td>
</tr>
<tr>
<td>MS COCO</td>
<td>27,842</td>
<td>120k</td>
<td>10k</td>
</tr>
</tbody>
</table>

The policy search tries to balance the exploration and exploitation wrt the policy through the following objective:

$$\mathbb{E}_{s,a \sim \pi, \mathcal{E}} \left[ \sum_{t=0}^{\infty} \gamma^t (r(s,a) + \alpha H(\pi(\cdot|s))) \right],$$  \hspace{1cm} (31)$$

$\alpha$ is a hyper parameter balancing the trade-off between exploitation (reward) and exploration (entropy), and $0 < \gamma < 1$ is the discounting factor. It’s well known the optimal policy then would follow $\pi^*(a|s) \propto \exp(Q(a, s)/\alpha)$, where $Q(a, s)$ is known as the Q-function (Sutton & Barto, 2018).

Soft-Q learning (Haarnoja et al., 2017) leverages the current policy $\pi_t$ to interact with the environment $\mathcal{E}$ to update Q-function estimate $Q_t(a, s)$, and then train the policy towards the optimal distribution defined by $Q_t(a, s)$.

**SVGD-SQL** In Soft Q-Learning (Haarnoja et al., 2017), the policy network is trained amortised in two steps: (1), draw action from the policy network and use these actions as the initial points for SVGD update. (2), use the $\ell_2$ distance between the origin samples and the updated ones to calculate the gradient of the policy network. The first step suffers from a risk that the updated samples are out of the action space. Constraints should be added to prevent from this, leading to the unexpected errors.

S.2. Experimental setup

Detailed architectures and parameter setting used in our experiments are summarized in Tables S9 to S9. We used the notation “X–H–H–Z” to denote a network with X as the in-
put size, \(H\) as the number of hidden units and \(Z\) is the output size, and the notation \(+\) denotes concatenation. Rectified linear units (ReLU) are used as the activation function for the hidden layers in all our experiments. Hyper-tangent (tanh) activation is applied to the policy network’s output. \(N\) denotes standard Gaussian noise with the same dimension as the action space.

Table S8. Environment hyper-parameters for SQL experiments.

<table>
<thead>
<tr>
<th>Environment</th>
<th>Action Space Dim</th>
<th>Reward Scale</th>
<th>Replay Pool Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swimmer (rllab)</td>
<td>2</td>
<td>100</td>
<td>(10^6)</td>
</tr>
<tr>
<td>Hopper-v1</td>
<td>3</td>
<td>1</td>
<td>(10^6)</td>
</tr>
<tr>
<td>Walker2d-v1</td>
<td>6</td>
<td>3</td>
<td>(10^6)</td>
</tr>
<tr>
<td>Reacher-v1</td>
<td>2</td>
<td>10</td>
<td>(10^6)</td>
</tr>
</tbody>
</table>

Table S9. Neural architectures used for SQL experiments.

<table>
<thead>
<tr>
<th>Network</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy-Network</td>
<td>(</td>
</tr>
<tr>
<td>Q-Network</td>
<td>(</td>
</tr>
<tr>
<td>(\Phi)-Network</td>
<td>(</td>
</tr>
<tr>
<td>(b)-Network</td>
<td>(</td>
</tr>
</tbody>
</table>

Table S10. Training hyper-parameters for SQL experiments.

<table>
<thead>
<tr>
<th>Hyper-parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate for policy-net</td>
<td>(3 \times 10^{-4})</td>
</tr>
<tr>
<td>Learning rate for Q-net</td>
<td>(3 \times 10^{-4})</td>
</tr>
<tr>
<td>Batch-size</td>
<td>128</td>
</tr>
<tr>
<td>SVGD particle size</td>
<td>32</td>
</tr>
</tbody>
</table>

References


