Statistica Sinica Preprint No: SS-2022-0404

Title	A New Paradigm for Generative Adversarial Networks Based on Randomized Decision Rules
Manuscript ID	SS-2022-0404
URL	http://www.stat.sinica.edu.tw/statistica/
DOI	10.5705/ss.202022.0404
Complete List of Authors	Sehwan Kim, Qifan Song and Faming Liang
Corresponding Authors	Faming Liang
E-mails	fmliang@purdue.edu
Notice: Accepted version subject to English editing.	

A New Paradigm for Generative Adversarial Networks Based on Randomized Decision Rules

Sehwan Kim, Qifan Song and Faming Liang

Purdue University

Abstract:

The Generative Adversarial Network (GAN) was recently introduced in the literature as a novel machine learning method for training generative models. It has many applications in statistics such as nonparametric clustering and nonparametric conditional independence tests. However, training the GAN is notoriously difficult due to the issue of mode collapse, which refers to the lack of diversity among generated data. In this paper, we identify the reasons why the GAN suffers from this issue, and to address it, we propose a new formulation for the GAN based on randomized decision rules. In the new formulation, the discriminator converges to a fixed point while the generator converges to a distribution at the Nash equilibrium. We propose to train the GAN by an empirical Bayes-like method by treating the discriminator as a hyper-parameter of the posterior distribution of the generator. Specifically, we simulate generators from its posterior distribution conditioned on the discriminator using a stochastic gradient Markov chain Monte Carlo (MCMC) algorithm, and update the discriminator using stochastic gradient descent along with simulations of the generators. We establish convergence of the proposed method to the Nash equilibrium. Apart from image generation, we apply the proposed method to nonparametric clustering and nonparametric conditional independence tests. A portion of the numerical results is presented in the supplementary material.

Key words and phrases: Generative Model, Minimax Game, Stochastic Approximation, Stochastic Gradient Markov Chain Monte Carlo.

1. Introduction

The Generative Adversarial Network (GAN) (Goodfellow et al., 2014) provides a novel way for training generative models which seek to generate new data with the same statistics as the training data. Other than image generation, the GAN has been used in many nonparametric statistical tasks, such as clustering (Mukherjee et al., 2019), conditional independent test (Bellot and van der Schaar, 2019), and density estimation (Singh et al., 2018; Liu et al., 2021). In this paper, we call the training data real samples, and those generated by the GAN fake samples.

In its original design, the GAN is trained by competing two neural networks, namely generator and discriminator, in a game. However, due to the instability issues such as *mode collapse* (i.e., lack of diversity among fake samples), non-convergence, and vanishing or exploding gradients, the GAN is notoriously hard to train (Wiatrak and Albrecht, 2019). In this paper, we identify the reasons why the GAN suffers from the mode collapse issue: *(i) The GAN evaluates fake samples at an individual level, lacking a mechanism for enhancing the diversity of fake samples; and (ii) the GAN tends to get trapped into a sub-optimal solution, lacking a mechanism for escaping from local traps* (see Remark 1 for more explanations). To address this issue, we propose a new formulation for the GAN based on randomized decision rules. In this formulation, the similarity between the fake and real samples can be evaluated at the population level; and the generator is simulated from its posterior distribution conditioned on the discriminator using a stochastic gradient MCMC algorithm, thereby mitigating the difficulty of getting trapped in local optima.

Our contribution. The main contribution of this paper is three-fold: (i) we have provided a new formulation for the GAN based on statistical randomized decision theory, which allows the mode collapse issue to be fully addressed; (ii) we have proposed a training algorithm associated with the new formulation, and shown that its convergence to the Nash equilibrium is asymptotically guaranteed, or said differently, the proposed algorithm is immune to mode collapse as the number of iterations becomes large; (iii) we have developed a Kullback-Leibler divergence-based prior for the generator, which enhances the diversity of fake samples and further strengthens the effectiveness of the proposed method in overcoming the issue of mode collapse. The proposed method is tested on image generation, nonparametric clustering, and nonparametric conditional independence tests (in the supplementary material). Our numerical results suggest that the proposed method significantly outperforms the existing ones in overcoming the mode collapse issue.

Related Works. To tackle the mode collapse issue, a variety of methods have been proposed in the literature, see Wiatrak and Albrecht (2019) for a recent survey. These methods can be roughly grouped to two categories, namely, metric-based methods and mixture generator methods.

The methods in the first category strive to find a more stable and informative metric to guide the training process of the GAN. For example, Nowozin et al. (2016) suggested f-divergence, Mao et al. (2017) suggested χ^2-divergence, Arjovsky et al. (2017) suggested Wasserstein distance, Binkowski et al. (2018) suggested maximum mean discrepancy, and Che et al. (2017) and Zhou et al. (2019) suggested some regularized objective functions. As mentioned previously, the GAN evaluates fake samples at the individual level and tends to get trapped to a sub-optimal solution. Therefore, the mode collapse issue is hard to resolve by employing a different metric unless (i) the objective function is modified such that the similarity between the fake and real samples can be enhanced at the population level, and (ii) a local-trap free optimization algorithm is employed for training. Recently, there has been a growing trend in the literature to incorporate gradient flow into the training of generative models, as explored by Gao et al. (2019). However, achieving this objective is generally considered a challenging task.

The methods in the second category are to learn a mixture of generators under a probabilistic framework with a similar motivation to this work. A non-exhaustive list of such types of methods include ensemble GAN (Wang et al., 2016), Mix+GAN (Arora et al., 2017), AdaGAN (Tolstikhin et al., 2017), MAD-GAN (Ghosh et al., 2018), MGAN (Hoang et al., 2018), Bayesian GAN (Saatci and Wilson, 2017), and ProbGAN (He et al., 2019). However, many of the methods are not defined in a proper probabilistic framework and, in consequence, the mode collapse issue cannot be overcome with a theoretical guarantee. In ensemble GAN, AdaGAN, MAD-GAN, Mix+GAN, and MGAN, only a finite mixture of generators is learned and thus the mode collapse issue cannot be overcome in theory. Bayesian GAN aims to overcome this obstacle by simulating the discriminator and generator from their respective conditional posterior distributions; however, the two conditional posterior distributions are incompatible and can lead unpredictable behavior (Arnold and Press, 1989). ProbGAN imposes an adaptive prior on the generator and updates the prior by successively multiplying the likelihood function at each iteration; consequently, the generator converges to a fixed point instead of a distribution.

The remaining part of this paper is organized as follows. Section 2 describes the new formulation for the GAN based on randomized decision rules. Section 3 proposes a training method and proves its convergence to the Nash equilibrium. Section 4 illustrates the performance of the proposed method using synthetic and real data examples. Section 5 concludes the paper with a brief discussion.

2. A New Formulation for GAN based on Randomized Decision Rules

2.1 Pure Strategy Minimax Game

In the original work Goodfellow et al. (2014), the GAN is trained by competing the discriminator and generator neural networks in a game. Let θ_d denote the parameters of the discriminator neural network, and let $D_{\theta_d}(x)$ denote its output function which gives a score for discriminating whether or not the input sample x is generated from the data distribution p_{data}. Let $G_{\theta_g}(z)$ denote the generator neural network with parameter θ_g, whose input

z follows a given distribution $q(z)$, e.g., uniform or Gaussian, and whose output distribution is denoted by p_{θ_g}. Define

$$\begin{aligned}\mathcal{J}_d(\theta_d;\theta_g) &= \mathbb{E}_{x\sim p_{data}}\phi_1(D_{\theta_d}(x)) + \mathbb{E}_{x\sim p_{\theta_g}}\phi_2(D_{\theta_d}(x)),\\ \mathcal{J}_g(\theta_g;\theta_d) &= -\mathbb{E}_{x\sim p_{data}}\phi_1(D_{\theta_d}(x)) + \mathbb{E}_{x\sim p_{\theta_g}}\phi_3(D_{\theta_d}(x)),\end{aligned} \tag{2.1}$$

where $\phi_1(D) = \log(D)$, $\phi_2(D) = \log(1-D)$, and $\phi_3(D) = -\log(1-D)$ or $\log(D)$ are as defined in Goodfellow et al. (2014). The general form of the game introduced by Goodfellow et al. (2014) is given as follows:

$$(i) \quad \max_{\theta_d} \mathcal{J}_d(\theta_d;\theta_g), \quad (ii) \quad \max_{\theta_g} \mathcal{J}_g(\theta_g;\theta_d). \tag{2.2}$$

If $\phi_3 = -\phi_2$, the objective of (2.2) represents a pure strategy minimax game, i.e.,

$$\min_{\theta_g}\max_{\theta_d} \mathcal{J}_d(\theta_g,\theta_d), \tag{2.3}$$

which is called minimax GAN. If $\phi_3(D) = \log(D)$, the objective is said to be non-saturating, which results in the same fixed point of the dynamics as the minimax GAN but addresses the issue of vanishing gradient suffered by the latter. Quite recently, Zhou et al. (2019) proposed to penalize $\mathcal{J}_d(\theta_d;\theta_g)$ by a quadratic function of the Lipschitz constant of θ_d, which addresses the gradient uninformativeness issue suffered by minimax GAN and improves its convergence.

2.2 Mixed Strategy Minimax Game

Let $\pi_g(\theta_g)$ denote a distribution of generators. Based on the randomized decision theory, we define a mixed strategy minimax game:

$$\min_{\pi_g} \max_{\theta_d} \mathbb{E}_{\pi_g} \mathcal{J}_d(\theta_d; \theta_g), \tag{2.4}$$

where $\mathcal{J}_d(\theta_d; \theta_g)$ is as defined in (2.1), and the expectation is taken with respect to $\pi_g(\theta_g)$. That is, the game is to iteratively search for an optimal discriminator θ_d by maximizing $\mathbb{E}_{\pi_g} \mathcal{J}_d(\theta_d; \theta_g)$ for a given generator distribution π_g and an optimal generator distribution π_g by minimizing $\max_{\theta_d} \mathbb{E}_{\pi_g} \mathcal{J}_d(\theta_d; \theta_g)$ for a given discriminator θ_d. In its Nash equilibrium, the discriminator is fixed and the generator is randomly drawn from the optimal generator distribution π_g, so the equilibrium is a mixed strategy Nash equilibrium. This is different from the pure strategy Nash equilibrium achieved by the minimax GAN, where both the discriminator and generator are fixed at equilibrium.

From the viewpoint of statistical decision theory, (2.4) is a minimax randomized decision problem, where π_g can be viewed as a randomized decision rule and $\mathbb{E}_{\pi_g} \mathcal{J}_d(\theta_d; \theta_g)$ can be viewed as a risk function. Compared to the deterministic decision formulation (2.3), such a randomized decision formulation naturally accounts for the uncertainty of the generator and thus helps to address the mode collapse issue. Note that a deterministic decision rule is a special case of a randomized decision rule where one decision or action has probability 1. Further, Young and Smith (2005) (p.11) pointed out that a minimax randomized decision rule might perform better than all other deterministic decision rules under certain situations.

Let p_{π_g} denote the distribution of the fake samples produced by the generators drawn from π_g, i.e., $p_{\pi_g}(x) = \int p_{\theta_g}(x)\pi_g(\theta_g)d\theta_g$. Lemma 1 studies the basic property of the mixed strategy minimax game (2.4). The proof of this lemma, along with the proofs of other theoretical results in this paper, is provided in the supplement.

Lemma 1. *Suppose the discriminator and generator have enough capacity, $\phi_1(D) = \log(D)$, and $\phi_2(D) = \log(1-D)$. For the game (2.4), $\min_{\pi_g} \max_{\theta_d} \mathbb{E}_{\pi_g} \mathcal{J}_d(\theta_d; \theta_g) = -\log(4)$. Further, if $\tilde{\theta}_d = \arg\max_{\theta_d} \mathbb{E}_{\tilde{\pi}_g} \mathcal{J}_d(\theta_d; \theta_g)$ for some $\tilde{\pi}_g$, then $(\tilde{\theta}_d, \tilde{\pi}_g)$ is a Nash equilibrium point if and only if $\mathbb{E}_{\tilde{\pi}_g} \mathcal{J}_d(\tilde{\theta}_d; \theta_g)$ $= -\log(4)$; at any Nash equilibrium point $(\tilde{\theta}_d, \tilde{\pi}_g)$, $p_{\tilde{\pi}_g} = p_{data}$ holds and $D_{\tilde{\theta}_d}(x) = 1/2$ for any $x \sim p_{data}$, where $p_{\tilde{\pi}_g} = \int p_{\theta_g} \tilde{\pi}_g(\theta_g) d\theta_g$ and $x \sim p_{data}$ means x is distributed according to p_{data}.*

Lemma 1 can be generalized to other choices of ϕ_1 and ϕ_2. In general, if ϕ_1 and ϕ_2 satisfy that (i) $\phi_1' > 0$, $\phi_2' < 0$, $\phi_1'' \leq 0$, $\phi_2'' \leq 0$, where ϕ_i' and ϕ_i'' denote the first and second derivatives of ϕ_i $(i = 1, 2)$, respectively; and (ii) there exists some value a such that $\phi_1'(a) + \phi_2'(a) = 0$, then the conclusion of the lemma still holds except that $D_{\tilde{\theta}_d} \equiv a$ in this case.

2.3 Mixed Strategy Nash Equilibrium

Let $q_g(\theta_g)$ denote the prior distribution of θ_g, and let N denote the training sample size. Define

$$\pi(\theta_g|\theta_d, \mathcal{D}) \propto \exp\{\mathbb{J}_g(\theta_g; \theta_d)\} q_g(\theta_g), \tag{2.5}$$

where

$$\mathbb{J}_g(\theta_g; \theta_d) = N\mathcal{J}_g(\theta_g; \theta_d) = N(-\mathbb{E}_{x \sim p_{data}} \phi_1(D_{\theta_d}(x)) + \mathbb{E}_{x \sim p_{\theta_g}} \phi_3(D_{\theta_d}(x)),$$

and ϕ_3 is an appropriately defined function, e.g., $\phi_3(D) = -\log(1-D)$ or $\log(D)$ as in Goodfellow et al. (2014). For the game (2.4), we propose to solve for θ_d by setting

$$\tilde{\theta}_d = \arg\max_{\theta_d} \int \mathcal{J}_d(\theta_d; \theta_g) \pi(\theta_g|\theta_d, \mathcal{D}) d\theta_g, \tag{2.6}$$

where $\mathcal{J}_d(\theta_d; \theta_g)$ is as defined in (2.1) and then, with a slight abuse of notation, setting

$$\tilde{\pi}_g = \pi(\theta_g|\tilde{\theta}_d, \mathcal{D}). \tag{2.7}$$

Theorem 1 shows that $(\tilde{\theta}_d, \tilde{\pi}_g)$ defined in (2.6)-(2.7) is a Nash equilibrium point for the game (2.4) as $N \to \infty$.

Theorem 1. *Suppose that the discriminator and generator have enough capacity,* $\phi_1(D) = \log(D)$*,* $\phi_2(D) = \log(1-D)$*,* $\phi_3 = -\log(1-D)$*, and the following conditions hold: (i)* $\dim(\theta_g)$*, the dimension of the generator, grows with* N *at a rate of* $O(N^\zeta)$ *for some* $0 \leq \zeta < 1$*; and (ii) the prior density function* $q_g(\theta_g)$ *is upper bounded on the parameter space* Θ_g *of the generator. Then* $(\tilde{\theta}_d, \tilde{\pi}_g)$ *defined in (2.6)-(2.7) is a Nash equilibrium point for the game (2.4) as* $N \to \infty$*.*

Condition (ii) can be satisfied by many prior distributions, e.g., the uninformative prior $q_g(\theta_g) \propto 1$ and the Gaussian prior. In addition, we consider an extra type of prior, namely, KL-prior, in this paper. The KL-prior is given by

$$q_g(\theta_g) \propto \exp\{-\lambda D_{KL}(p_{data}|p_{\theta_g})\}, \tag{2.8}$$

where λ is a pre-specified constant, and the KL-divergence $D_{KL}(p_{data}|p_{\theta_g})$ can be estimated by a k-nearest-neighbor density estimation method (Pérez-Cruz, 2008; Wang et al., 2009) based on the real and fake samples. The motivation of this prior is to introduce to the proposed method a mechanism for enhancing the similarity between p_{θ_g} and p_{data} at the density level. For the Gaussian prior, we generally suggest to set $\theta_g \sim \mathcal{N}(0, \sigma_N^2 I_{\dim(\theta_g)})$, where $\sigma_N^2 \geq 1/(2\pi)$ and increases with the training sample size N in such a way that the prior approaches uniformity asymptotically as $N \to \infty$.

There are ways other than (2.5)-(2.7) to define $(\tilde{\theta}_d, \tilde{\pi}_g)$ and still have Theorem 1 be valid. For example, one can define $\pi(\theta_g|\theta_d, \mathcal{D}) \propto \exp\{\mathbb{J}_g(\theta_g; \theta_d)/\tau\}$ $q_g(\theta_g)$ or $\pi(\theta_g|\theta_d, \mathcal{D}) \propto \exp\{\mathbb{J}_g(\theta_g; \theta_d)/\tau\}(q_g(\theta_g))^{1/\tau}$ for some temperature $\tau > 0$. That is, instead of the exact conditional posterior $\pi(\theta_g|\tilde{\theta}_d, \mathcal{D})$, one can sample from its tempered version. In the extreme case, one may employ the proposed method to find the Nash equilibrium point for the minimax GAN in a manner of simulated annealing (Kirkpatrick et al., 1983).

Corollary 1. *The conclusion of Theorem 1 still holds if the function* $\phi_3(D) = -\log(1-D)$ *is replaced with* $\phi_3(D) = \log(D)$.

To make a more general formulation for the game (2.4), we can include a penalty term in $\mathcal{J}_d(\theta_d; \theta_g)$ such that

$$\mathcal{J}_d(\theta_d; \theta_g) = \mathbb{E}_{x\sim p_{data}}\phi_1(D_{\theta_d}(x)) + \mathbb{E}_{x\sim p_{\theta_g}}\phi_2(D_{\theta_d}(x)) - \lambda l(D_{\theta_d}), \tag{2.9}$$

where $l(D_{\theta_d}) \geq 0$ denotes an appropriate penalty function on the discriminator. For example, one can set $l(D_{\theta_d}) = \|D_{\theta_d}\|^{\alpha}_{Lip}$ for some $\alpha > 1$, where $\|D_{\theta_d}\|_{Lip}$ denotes the Lipschitz constant of the discriminator. As explained in Zhou et al. (2019), including this penalty term enables the minimax GAN to overcome the gradient uninformativeness issue and improve its convergence. As implied by the proof of Lemma 1, where the mixture generator proposed in the paper can be represented as a single super generator, the arguments in Zhou et al. (2019) still apply and thus $\|D_{\tilde{\theta}_d}\|_{Lip} = 0$ holds at the optimal discriminator $\tilde{\theta}_d = \arg\max_{\theta_d} \mathbb{E}_{\theta_g \sim \pi_g} \mathcal{J}_d(\theta_d; \theta_g)$. This further implies that the extra penalty term $-\lambda \|D_{\theta_d}\|_{Lip}^{\alpha}$ does not affect the definition of $\pi(\theta_g | \tilde{\theta}_d, \mathcal{D})$ and, therefore, Theorem 1 still holds with (2.9).

3. Training Algorithm and Its Convergence

This section proposes an algorithm for solving the integral optimization problem (2.6) and studies its convergence to the Nash equilibrium.

3.1 The Training Algorithm

A straightforward calculation shows that

$$\begin{aligned}\nabla_{\theta_d} \int \mathcal{J}_d(\theta_d; \theta_g) \pi(\theta_g | \theta_d, \mathcal{D}) d\theta_g &= \mathbb{E}_{\pi_{g|d}}(\nabla_{\theta_d} \mathcal{J}_d(\theta_d; \theta_g)) \\ &\quad + \mathrm{Cov}_{\pi_{g|d}}(\mathcal{J}_d(\theta_d; \theta_g), \nabla_{\theta_d} \mathbb{J}_g(\theta_g; \theta_d)),\end{aligned}$$

where $\mathbb{E}_{\pi_{g|d}}(\cdot)$ and $\mathrm{Cov}_{\pi_{g|d}}(\cdot)$ denote the mean and covariance operators with respect to $\pi(\theta_g | \theta_d, \mathcal{D})$, respectively. By Lemma 1, at any Nash equilibrium point we have $p_{\tilde{\pi}_g} = p_{data}$ and $D_{\tilde{\theta}_d} = 1/2$. Then, following the arguments given in the proof of Theorem 1, it is easy to show by Laplace approximation that at the Nash equilibrium point $\mathrm{Cov}_{\pi_{g|d}}(\mathcal{J}_d(\tilde{\theta}_d; \theta_g), \nabla_{\theta_d} \mathbb{J}_g(\theta_g; \tilde{\theta}_d)) \to 0$ as $N \to \infty$. Therefore, when N is sufficiently large, the target equation $\nabla_{\theta_d} \int \mathcal{J}_d(\theta_d; \theta_g) \pi(\theta_g | \theta_d, \mathcal{D}) d\theta_g = 0$ can be solved by solving the mean field equation

$$h(\theta_d) = \int H(\theta_d, \theta_g)\pi(\theta_g|\theta_d, \mathcal{D}) = 0, \tag{3.10}$$

using a stochastic approximation algorithm, where $H(\theta_d, \theta_g)$ denotes an unbiased estimator of $\nabla_{\theta_d}\mathcal{J}_d(\theta_d; \theta_g)$. The convergence of the solution to the Nash equilibrium can be assessed by examining the plots described in Section 4. By the standard theory of stochastic approximation MCMC, see e.g., Benveniste et al. (1990); Andrieu et al. (2005); Deng et al. (2019); Dong et al. (2023), equation (3.10) can be solved by iterating between the following two steps, where $\theta_d^{(t)}$ denotes the estimate of the discriminator obtained at iteration t, and $\theta_g^{(t)}$ denotes a generic sample of the generator simulated at iteration t:

(i) Simulate $\theta_g^{(t)}$ by a Markov transition kernel which leaves the conditional posterior $\pi(\theta_g|\theta_d^{(t-1)}, \mathcal{D}) \propto \exp\{\mathbb{J}_g(\theta_g; \theta_d^{(t-1)})\}q_g(\theta_g)$ invariant.

(ii) Update the estimate of θ_d by setting $\theta_d^{(t)} = \theta_d^{(t-1)} + w_t H(\theta_d^{(t-1)}, \theta_g^{(t)})$, where w_t denotes the step size used at iteration t.

Stochastic gradient MCMC algorithms, such as stochastic gradient Langevin dynamics (SGLD) (Welling and Teh, 2011), stochastic gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014) and momentum stochastic gradient Langevin dynamics (MSGLD) (Kim et al., 2022), can be used in step (i). Under appropriate conditions, we will show in Section 3.2 that $|\theta_d^{(t)} - \tilde{\theta}_d| \xrightarrow{p} 0$ and $\theta_g^{(t)} \xrightarrow{d} \pi(\theta_g|\tilde{\theta}_d, \mathcal{D})$ as $t \to \infty$, where $\xrightarrow{p}$ and $\xrightarrow{d}$ denote convergences in probability and distribution, respectively. That is, the proposed algorithm converges to the Nash equilibrium of the mixed strategy minimax game (2.4).

The proposed algorithm can also be viewed as an empirical Bayes-like method (Morris, 1983). For the case $\phi_3 = -\phi_2$, the posterior $\pi(\theta_g|\theta_d, \mathcal{D})$ can be expressed as

$$\pi(\theta_g|\theta_d, \mathcal{D}) \propto \exp\Big\{-\sum_{i=1}^{N} \phi_1(D_{\theta_d}(x_i)) - N * \mathbb{E}_{z\sim q}\phi_2(D_{\theta_d}(G_{\theta_g}(z)))\Big\} q_g(\theta_g), \quad (3.11)$$

where θ_d can be viewed as a hyperparameter of the posterior; and the proposed algorithm is to determine θ_d by solving the equation

$$\frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_d}\left[\phi_1(D_{\theta_d}(x_i)) + \mathbb{E}_{\pi_g}\mathbb{E}_{z\sim q}\phi_2(D_{\theta_d}(G_{\theta_g}(z_i)))\right] = 0. \quad (3.12)$$

In terms of the computational procedure, solving (3.12) is equivalent to maximizing the expected log-marginal posterior of θ_d, which can be derived from (3.11) by imposing on θ_d an improper prior $\pi(\theta_d) \propto 1$. To distinguish the proposed computational procedure from Bayesian GAN (Saatci and Wilson, 2017), we call it an empirical Bayes-like GAN (or EBGAN in short).

Algorithm 1 summarizes the proposed algorithm as a solver for (2.6), where k_g generators are simulated using MSGLD (Kim et al., 2022) at each iteration, and the gradients are estimated with a mini-batch data of size n at each iteration. More precisely, we have

$$\begin{aligned} \nabla_{\theta_g} \tilde{L}(\theta_g, \theta_d) &= \frac{N}{n} \sum_{i=1}^{n} \nabla_{\theta_g} \phi_3(D_{\theta_d}(G_{\theta_g}(z_i))) + \nabla_{\theta_g} \log q_g(\theta_g), \\ H(\theta_d, \theta_g^{(t)}) &= \frac{1}{nk_g} \sum_{j=1}^{k_g} \sum_{i=1}^{n} \nabla_{\theta_d} \left[\phi_1(D_{\theta_d}(x_i^*)) + \phi_2(D_{\theta_d}(G_{\theta_g^{j,(t)}}(z_i))) \right], \end{aligned} \tag{3.13}$$

where $\{x_i^*\}_{i=1}^n$ denotes a set of mini-batch data and $\{z_i\}_{i=1}^n$ denotes independent inputs for the generator. As illustrated by Kim et al. (2022), MSGLD tends to converge faster than SGLD, where the momentum bias term can help the sampler to escape from saddle points and accelerate its convergence in simulations on the energy landscape with pathological curvatures.

```
**Algorithm 1** Empirical Bayesian GAN
  **Input:** Full data set $\mathcal{D} = \{x_i\}_{i=1}^N$, number of generators $k_g$, mini-batch size $n$, momentum smoothing factor $\alpha$, momentum biasing factor sequence $\{\rho_t\}_{t=1}^\infty$, learning rate sequence $\{\epsilon_t\}_{t=1}^\infty$, and step size sequence $\{w_t\}_{t=1}^\infty$.
  **Initialization:** $\theta_0$ from an appropriate distribution, set $m_0 = 0$;
  **for** $t = 1, 2, \ldots,$ **do**
    (i) Sampling step:
    **for** $j = 1, 2, \ldots, k_g$ **do**
      Draw a mini-batch data $\{x_i^*\}_{i=1}^n$, and set
      $\theta_g^{j,(t)} = \theta_g^{j,(t-1)} + \epsilon_t \left\{ \nabla_{\theta_g} \tilde{L}(\theta_g^{j,(t-1)}, \theta_d^{(t-1)}) + \rho_{t-1} m^{j,(t-1)} \right\} + \mathcal{N}(0, 2\tau\epsilon_t),$
      $m^{j,(t)} = \alpha m^{j,(t-1)} + (1-\alpha) \nabla_{\theta_g} \tilde{L}(\theta_g^{j,(t-1)}, \theta_d^{(t-1)}).$
    **end for**
    (ii) Parameter estimating step: $\theta_d^{(t)} = \theta_d^{(t-1)} + w_t H(\theta_d^{(t-1)}, \theta_g^{(t)})$, where $\theta_g^{(t)} = (\theta_g^{1,(t)}, \ldots, \theta_g^{k_g,(t)})$.
  **end for**
```

Regarding hyperparameter settings, we have the following suggestions.

In general, we set $w_t = c_1(t + c_2)^{-\zeta_1}$ for some constants $c_1 > 0$, $c_2 \geq 0$ and $\zeta_1 \in (0, 1]$, which satisfies Assumption S1. In this paper, we set $\zeta_1 = 0.75$ in all computations. Both the learning rate sequence and the momentum biasing factor sequence are required to converge to 0 as $t \to \infty$, i.e., $\lim_{t\to\infty} \epsilon_t = 0$ and $\lim_{t\to\infty} \rho_t = 0$. For example, one might set $\epsilon_t = O(1/t^{\zeta_2})$ and $\rho_t = O(1/t^{\zeta_3})$ for some $\zeta_2, \zeta_3 \in (0, 1)$. In the extreme case, one might set them to small constants for certain problems, however, under this setting, the convergence of $\theta_g^{(t)}$ to the target posterior distribution will hold approximately even when $t \to \infty$. In this paper, we set $k_g = 10$ and the momentum smoothing factor $\alpha = 0.9$ as the default.

3.2 Convergence Analysis

Lemma 2 establishes the convergence of the discriminator estimator, and Lemma 3 shows how to construct the mixture generator desired for generating fake samples mimicking the real ones. For simplicity, we present the lemmas under the setting $k_g = 1$.

Lemma 2 (Convergence of discriminator)**.** *Suppose Assumptions S1-S6 (given in the supplement) hold. If the learning rate ϵ_t is sufficiently small, then there exist a constant γ, an iteration number t_0 and an optimum $\tilde{\theta}_d =$*

$\arg\max_{\theta_d} \int \mathcal{J}_d(\theta_d; \theta_g)\pi(\theta_g|\theta_d, \mathcal{D})d\theta_g$ *such that for any* $t \geq t_0$,

$$\mathbb{E}\|\theta_d^{(t)} - \tilde{\theta}_d\|^2 \leq \gamma w_t,$$

where t *indexes iterations,* w_t *is the step size satisfying Assumption S1, and an explicit formula of* γ *is given in (S1.20).*

As shown in (S1.20), the expression of γ consists of two terms. The first term γ_0 depends only on the sequence $\{\omega_t\}$ and the stability constant of the mean field function $h(\theta_d)$, while the second term characterizes the effects of the learning rate sequence $\{\epsilon_t\}$ and other constants (given in the assumptions) on the convergence of $\{\theta_d^{(t)}\}$. In particular, $\{\epsilon_t\}$ affects the convergence of $\{\theta_d^{(t)}\}$ via the upper bound of $\mathbb{E}\|\theta_g^{(t)}\|^2$. See Lemma S1 for the definition of the upper bound.

Lemma 3 (Ergodicity of generator)**.** *Suppose Assumptions S1-S7 (given in the supplement) hold. For a smooth test function* $\psi(\theta_g)$ *with* $\|\psi(\theta_g)\| \leq C(1 + \|\theta_g\|)$ *for some constant* C*, define*

$$\hat{\psi}_T = \frac{\sum_{t=1}^T \epsilon_t \psi(\theta_g^{(t)})}{\sum_{t=1}^T \epsilon_t}, \tag{3.14}$$

where T *is the total number of iterations. Let* $\bar{\psi} = \int \psi(\theta_g)\pi(\theta_g|\tilde{\theta}_d, \mathcal{D})d\theta_g$, $S_T = \sum_{t=1}^T \epsilon_t$*, and* $\Delta V_t = \nabla_{\theta_g}\tilde{L}(\theta_g^{(t)}, \theta_d^{(t)}) - \nabla_{\theta_g}L(\theta_g^{(t)}, \theta_d^{(t)})$.

(i) Suppose the following conditions are satisfied: the momentum biasing factor sequence $\{\rho_t : t = 1, 2, \ldots\}$ *decays to 0, the learning*

rate sequence $\{\epsilon_t : t = 1, 2, \ldots\}$ *decays to 0,* $\sum_{t=1}^{\infty} \epsilon_t = \infty$*, and* $\lim_{T\to\infty} \frac{\sum_{t=1}^{T} \epsilon_t^2}{\sum_{t=1}^{T} \epsilon_t} = 0$*. Then there exists a constant* C *such that*

$$\mathbb{E}\|\hat{\psi}_T - \bar{\psi}_T\|^2 \leq C \left(\sum_{t=1}^{T} \frac{\epsilon_t^2}{S_T^2} \mathbb{E}\|\Delta V_t\|^2 + \frac{1}{S_T} + \frac{(\sum_{t=1}^{T} \epsilon_t^2)^2}{S_T^2} \right).$$

(ii) Suppose a constant learning rate of ϵ *and a constant momentum biasing factor of* ρ *are used. Then there exists a constant* C *such that*

$$\mathbb{E}\|\hat{\psi}_T - \bar{\psi}\|^2 \leq C \left(\frac{\sum_{t=1}^{T} \mathbb{E}\|\Delta V_t\|^2}{T^2} + \frac{1}{T\epsilon} + \epsilon^2 + \rho^2 \right).$$

The estimator (3.14) provides us a convenient way to construct $p_{\tilde{\pi}_g}$; that is, as $T \to \infty$, the corresponding mixture generator can contain all the generators simulated by Algorithm 1 in a run. We note that, by Theorem 1 of Song et al. (2020), the estimator (3.14) can be simplified to the simple path average $\hat{\psi}'_T = \frac{1}{T} \sum_{t=1}^{T} \psi(\theta_g^{(t)})$ provided that $\epsilon_t \prec \frac{1}{t}$ holds, where $a_t \prec b_t$ means $\frac{a_t}{b_t} \to 0$ as $t \to \infty$. In practice, we can use only the generators simulated after the algorithm has converged or those simulated at the last iteration. For the latter, we may require k_g to be reasonably large.

Remark 1. While the mixture generator produced by Algorithm 1 can overcome the mode collapse issue, a single generator might not, especially when an uninformative or Gaussian prior is used. Suppose that the uninformative prior $q_g(\theta_g) \propto 1$ is used, $\phi_3(D) = -\log(1 - D)$, and a discriminator

$\tilde{\theta}_d$ with $D_{\tilde{\theta}_d}(x) = 1/2$ for $x \in p_{data}$ has been obtained. With such a discriminator, there are many ϑ_g's maximizing $\mathbb{J}_g(\vartheta_g; \tilde{\theta}_d)$ as long as $p_{\vartheta_g} \subset p_{data}$, because the GAN evaluates the fake samples at the individual level. Here we use the notation $p_{\vartheta_g} \subset p_{data}$ to denote that the fake samples generated from p_{ϑ_g} resemble only a subset of the real samples. At such a point $(\tilde{\theta}_d, \vartheta_g)$, we have $\mathcal{J}_d(\tilde{\theta}_d; \vartheta_g) = -\log 4$ and $-\mathbb{J}_g(\vartheta_g; \tilde{\theta}_d) = -N \log 4$. The latter means that the generator has attained its minimum energy, although $p_{\vartheta_g} \subset p_{data}$ is still sub-optimal; in other words, such a generator is trapped to a sub-optimal solution. However, if Algorithm 1 is run for sufficiently long time and the generators from different iterations are used for estimation, we can still have $\frac{1}{Tk_g} \sum_{t=1}^{T} \sum_{i=1}^{k_g} \int p_{\vartheta_{g,i}^{(t)}} \pi(\vartheta_{g,i}^{(t)} | \tilde{\theta}_d^{(t)}, \mathcal{D}) d\vartheta_{g,i}^{(t)} \approx p_{data}$ by assembling many sub-optimal generators (provided the learning rate $\epsilon_t \prec 1/t$), where $\vartheta_{g,i}^{(t)}$ denotes the ith generator at iteration t and $\tilde{\theta}_d^{(t)}$ denotes the discriminator at iteration t. That is, using mixture generator is a valid way for overcoming the mode collapse issue. For the case of $\phi_3(D) = \log(D)$ and the case of the Gaussian prior, this is similar. The KL-prior provides a stronger force to drive $\frac{1}{k_g} \sum_{i=1}^{k_g} \int p_{\vartheta_{g,i}^{(t)}} \pi(\vartheta_{g,i}^{(t)} | \tilde{\theta}_d^{(t)}, \mathcal{D}) d\vartheta_{g,i}^{(t)}$ to p_{data} as $t \to \infty$, while the choice of k_g is not crucial.

4. Numerical Studies

We illustrate the performance of the EBGAN using various examples. Due to the space limit, some of the examples are presented in the supplement.

4.1 A Gaussian Example

Consider a 2-D Gaussian example, where the real samples are generated in the following procedure (Saatci and Wilson, 2017): (i) generate the cluster mean: $\mu \sim \mathcal{N}(0, I_2)$, where I_2 denotes a 2-dimensional identity matrix; (ii) generate a mapping matrix $M \in \mathbb{R}^{2\times 2}$ with each element independently drawn from $\mathcal{N}(0,1)$; (iii) generate 10,000 observations: $x_i \sim (\mathcal{N}(0, I_2) + \mu) \times M^T$, for $i = 1, 2, \ldots, 10,000$. The code used for data generation is available at `https://github.com/andrewgordonwilson/bayesgan/blob/master/bgan_util.py`. Both the discriminator and generators used for this example are fully connected neural networks with ReLU activation. The discriminator has a structure of $2 - 1000 - 1$, and the generator has a structure of $10 - 1000 - 2$.

The original GAN (Goodfellow et al., 2014) was first applied to this example with the parameter settings given in the supplement. Figure 1(a) shows the empirical means of $D_{\theta_d^{(t)}}(x)$ and $D_{\theta_d^{(t)}}(\tilde{x})$ along with iterations, where x represents a real sample and $\tilde{x}$ represents a fake sample simulated by the generator. For the given choices of ϕ_1 and ϕ_2, as implied by Lemma 1, we should have $\mathbb{E}(D_{\theta_d^{(t)}}(x)) = \mathbb{E}(D_{\theta_d^{(t)}}(\tilde{x})) = 0.5$ at the Nash equilibrium. As shown by Figure 1(a), the GAN did reach the 0.5-0.5 convergence. However, as shown by Figure 1(b), the generator still suffers from the mode collapse issue at this solution, where the fake samples resemble only a subset of the real samples. As mentioned previously, this is due to the reasons: *The GAN evaluates the fake samples at the individual level, lacking a mechanism for enhancing the diversity of fake samples, and tends to get trapped at a sub-optimal solution for which $p_{\theta_g} \subset p_{data}$ holds while the ideal objective value $-\log 4$ can still be attained.*

The mode collapse issue can be tackled by EBGAN, for which we consider both the KL-prior and Gaussian prior.

4.1.1 KL-prior

The KL-prior is given in (2.8), which enhances the similarity between p_{θ_g} and p_{data} at the density level. For this example, we set $\lambda = 100$, set $k = 1$ for k-nearest-neighbor density estimation (see Pérez-Cruz (2008) for the estimator), and used the auto-differentiation method to evaluate the gradient $\nabla_{\theta_g} \log q_g(\theta_g)$. Figure 1(c)&(d) summarize the results of EBGAN for this example with $\phi_3(D) = \log(D)$ and $k_g = 10$. The settings for the other parameters can be found in the supplement. For EBGAN, Figure 1(c) shows that it converges to the Nash equilibrium very fast, and Figure 1(d) shows that the fake samples simulated by a *single* generator match the real samples almost perfectly.

In summary, this example shows that EBGAN can overcome the mode collapse issue by employing a KL-prior that enhances the similarity between p_{θ_g} and p_{data} at the density level.

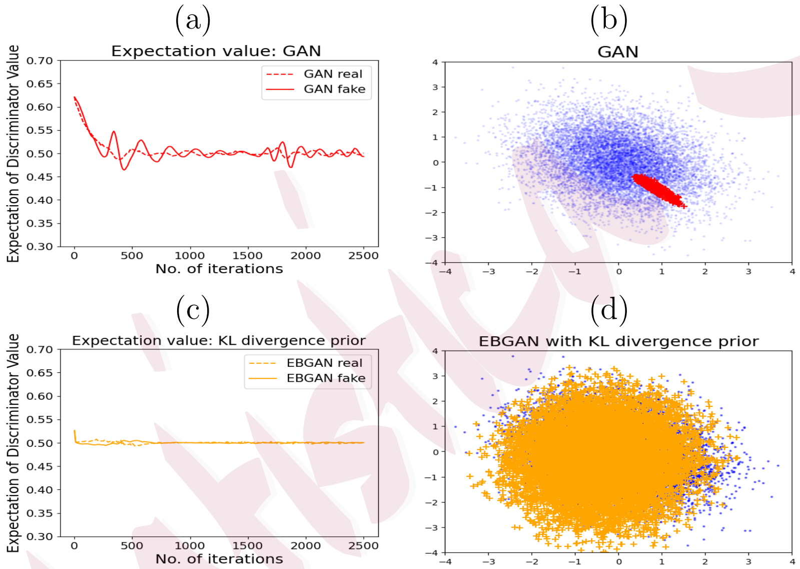
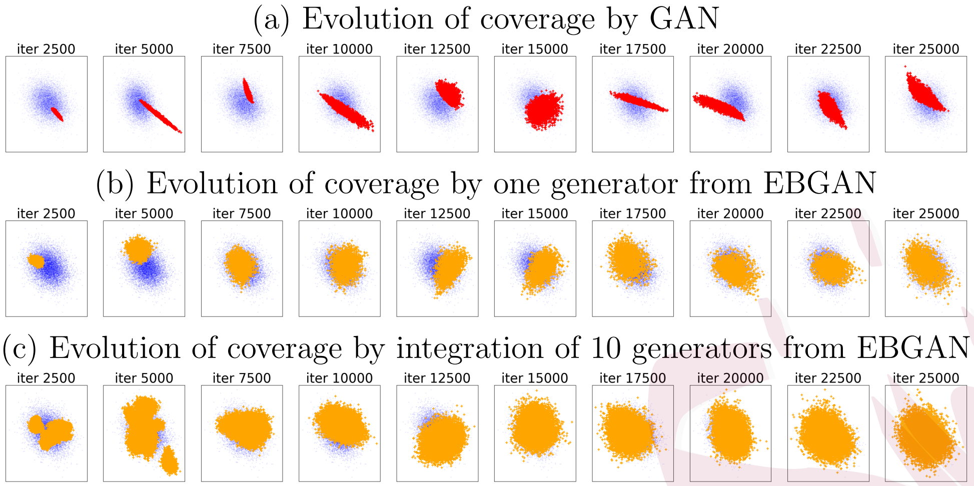


Figure 1: Illustration of the mode collapse issue: (a) empirical means of $D_{\theta_d^{(t)}}(x_i)$ and $D_{\theta_d^{(t)}}(\tilde{x}_i)$ produced by GAN; (b) coverage plot of the real (dots) and fake samples ('+') generated by GAN; (c) empirical means of $D_{\theta_d^{(t)}}(x_i)$ and $D_{\theta_d^{(t)}}(\tilde{x}_i)$ produced by EBGAN; (d) coverage plot of the real (dots) and fake samples ('+') generated by a *single* generator of EBGAN.

4.1.2 Gaussian prior

We have also tried the simple Gaussian prior $\theta_g \sim N(0, I_d)$ for this example. Compared to the KL-divergence prior, the Gaussian prior lacks the ability to enhance the similarity between p_{θ_g} and p_{data}, but it is much cheaper in computation. For this example, we have run EBGAN with $\phi_3(D) = \log(D)$ and $k_g = 10$. The settings for other parameters can be found in the supplement. To examine the performance of EBGAN with this cheap prior, we made a long run of 30,000 iterations. For comparison, the GAN was also applied to this example with $\phi_3(D) = \log(D)$. Figure S1 (in the supplement) shows the empirical means of $D_{\theta_d^{(t)}}(x)$ and $D_{\theta_d^{(t)}}(\tilde{x})$ produced by the two methods along with iterations, which indicates that both methods can reach the 0.5-0.5 convergence very fast. Figure 2 shows the evolution of the coverage plot of fake samples. Figure 2(a) indicates that the GAN has not reached the Nash equilibrium even with 25,000 iterations. In contrast, Figure 2(b) shows that even with the cheap Gaussian prior, the EBGAN can approximately reach the Nash equilibrium with 25,000 iterations, although it also suffers from the mode collapse issue in the early stage of the run. Figure 2(c) shows that for the EBGAN, the mode collapse issue can be easily overcome by integrating multiple generators.

Finally, we note that the convergence of the GAN and EBGAN should



(a) Evolution of coverage by GAN

(b) Evolution of coverage by one generator from EBGAN

(c) Evolution of coverage by integration of 10 generators from EBGAN

Figure 2: Coverage plots produced by (a) GAN, (b) a single generator of EBGAN, and (c) all 10 generators of EBGAN, where the generators for the plots from left to right are collected at iterations 2500, 5000, . . ., 25000, respectively. The dot points (in blue) represent real samples; and '+' points represent fake samples produced by GAN (in red) or EBGAN (in yellow).

be checked in two types of plots, namely, empirical convergence plot of $\mathbb{E}D_{\theta_d^{(t)}}(x)$ and $\mathbb{E}D_{\theta_d^{(t)}}(\tilde{x})$, and coverage plot of the fake and real samples. The former measures how well an individual fake sample fits into the population of real samples, while the latter measures the diversity of fake samples, i.e., whether a wide range of fake samples is generated.

4.2 A Mixture Gaussian Example

To further illustrate the performance of the EBGAN, we consider a more complex example which was taken from Saatci and Wilson (2017). The dataset was generated from a 10-component mixture Gaussian distribution in the following procedure: (i) generate 10 cluster means: $\mu^{(j)} \sim \mathcal{N}(0, 25I_2)$, $j = 1, 2, \ldots, 10$, where I_2 denotes a 2-dimensional identity matrix; (ii) generate 10 mapping matrices: $M^{(j)} \in \mathbb{R}^{100\times 2}$ for $j = 1, 2, \ldots, 10$, with each element of the matrices independently drawn from $\mathcal{N}(0, 25)$. (iii) generate 1000 observations of $x^{(j)}$ for each $(\mu^{(j)}, M^{(j)})$: $x_i^{(j)} \sim (\mathcal{N}(0, I_2) * 0.5 + \mu^{(j)}) \times (M^{(j)})^T$, for $j = 1, 2, \ldots, 10$, $i = 1, 2, \ldots, 1000$.

For this example, the EBGAN was run with the prior $q_g = \mathcal{N}(0, I)$, $k_g = 10$, and $\phi_3(D) = -\log(1-D)$ and $\log(D)$. The discriminator has a structure of $100-1000-1$ and the generator has a structure of $10-1000-100$, which are the same as those used in Saatci and Wilson (2017). The results with $\phi_3(D) = -\log(1-D)$ are presented below and those with $\phi_3(D) = \log(D)$ are presented in the supplement.

For comparison, the minimax GAN (Goodfellow et al., 2014), Bayesian GAN (Saatci and Wilson, 2017), ProbGAN (He et al., 2019), and Lipschitz GAN (Zhou et al., 2019) were applied to this example with the parameter settings given in the supplement. For all the four methods, we employed the same settings of ϕ_1, ϕ_2 and ϕ_3 and the same discriminator and generator as the EBGAN. For a thorough comparison, we have tried to train the EBGAN with a Lipschitz penalty.

Figure 3 examines the convergence of $\mathbb{E}(D_{\theta_d^{(t)}}(x))$ and $\mathbb{E}(D_{\theta_d^{(t)}}(\tilde{x}))$. It

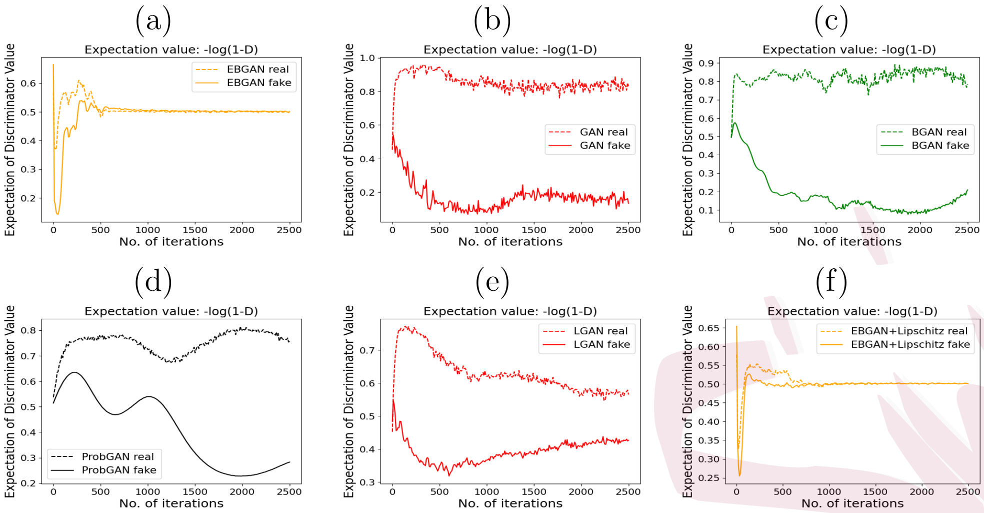


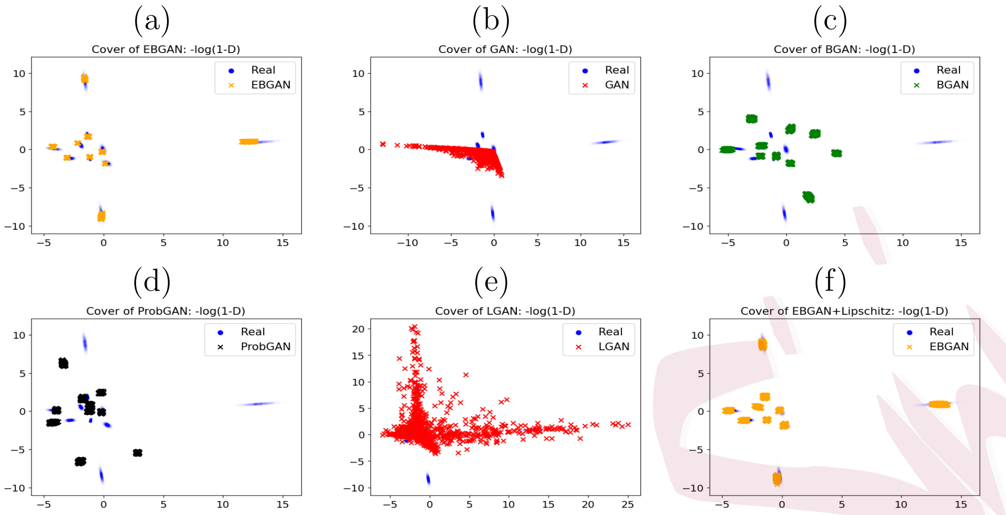
Figure 3: Nash equilibrium convergence plots with $\phi_3(D) = -\log(1-D)$, which compare the empirical means of $D_{\theta_d^{(t)}}(x_i)$ and $D_{\theta_d^{(t)}}(\tilde{x}_i)$ produced by different methods along with iterations: (a) EBGAN, (b) minimax GAN, (c) Bayesian GAN, (d) ProbGAN, (e) Lipschitz-GAN, (f) EBGAN with a Lipschitz penalty.

indicates that except for the EBGAN, none of the four methods, minimax GAN, Bayesian GAN, ProbGAN and Lipschitz-GAN, has reached the 0.5-0.5 convergence. Compared to Figure 3(a), Figure 3(f) shows that the Lipschitz penalty improves the convergence of the EBGAN slightly.

Other than the convergence plots of $\mathbb{E}(D_{\theta_d^{(t)}}(x))$ and $\mathbb{E}(D_{\theta_d^{(t)}}(\tilde{x}))$, we checked whether the fake samples recover all 10 components of the mixture distribution, where the principal component analysis (PCA) was used for high-dimensional data visualization. For EBGAN, we used only the generators obtained at the last iteration: we simulated 1000 fake samples from each of $k_g = 10$ generators. As shown in Figure 4, EBGAN recovered all 10 components in both cases with or without the penalty term, while the other four methods failed to do so. The minimax GAN and Lipschitz GAN, both of which work with a single generator, failed for this example. The BGAN and ProbGAN worked better than minimax GAN, but still missed a few components. In the supplement, we compared the performance of different methods with $\phi_3 = \log(D)$. The results are very similar to Figure 3 and Figure 4.

For the EBGAN, we have also tried to use generators simulated at multiple iterations, e.g., those in the last 2000 iterations. We found that the component recovery plot can be further improved. For simplicity, we used only the generators obtained at the last iteration. If the EBGAN is run with a larger value of k_g, the overlapping area can also be further improved.

In summary, EBGAN performs very well for this mixture example: the fake samples generated by it exhibit both good quality and diversity. The comparison with existing methods indicates that integrating multiple generators is essential for overcoming the mode collapse issue, particularly when the objective function lacks a mechanism to enhance the similarity between p_{θ_g} and p_{data} at the density level.



(a) (b) (c)

(d) (e) (f)

Figure 4: Component recovery plots with $\phi_3(D) = -\log(1-D)$: (a) EBGAN with $\lambda = 0$, (b) minimax GAN, (c) BGAN, (d) ProbGAN, (e) Lipschitz GAN, and (f) EBGAN with a Lipschitz penalty.

4.3 Image Generation

Fashion-MNIST is a dataset of 60,000 training images and 10,000 test images. Each image is of size 28×28 and has a label from 10 classes: T-shirt, Trouser, . . ., Ankle boot. The full description for the dataset can be found at `https://github.com/zalandoresearch/fashion-mnist`.

For this example, we compared EBGAN with GAN, BGAN and ProbGAN with parameter settings given in the supplement. The results are summarized in Figure 5 and Table 1. Figure 5 shows that for this example, the EBGAN can approximately achieve the 0.5-0.5 convergence for $\mathbb{E}(D_{\theta_d^{(t)}}(x_i))$ and $\mathbb{E}(D_{\theta_d^{(t)}}(\tilde{x}_i))$; that is, the EBGAN can produce high quality images which are almost indistinguishable from real ones. However, none of the existing three methods can achieve such good convergence.

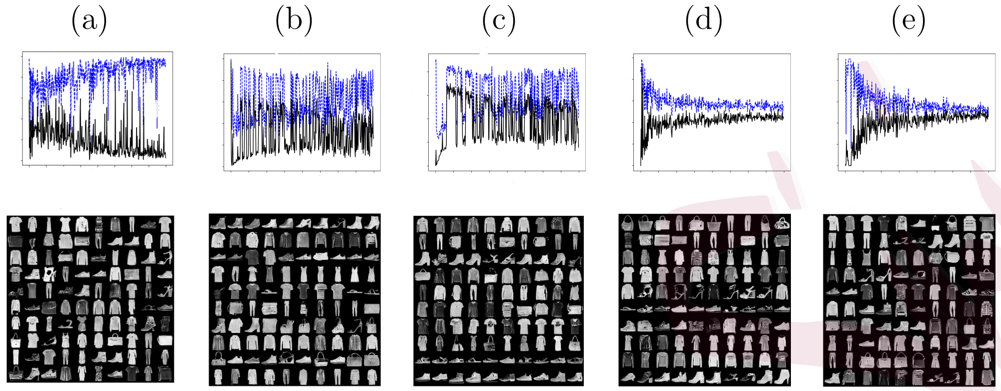


Figure 5: Convergence plots and images produced by (a) GAN, (b) Bayesian GAN, (c) ProbGAN, (d) EBGAN with a KL-divergence prior, and (e) EBGAN with a Gaussian prior. For each of the convergence plots, x-axis represents iterations, ranging from 0 to 40,000; and y-axis represents the empirical mean of discriminator values, ranging from 0 to 1.0, where dotted and solid lines are for real and fake samples, respectively.

Table 1: Average IS, 1-WD, and MMD values produced by different methods for Fashion MNIST, where the averages and standard deviations (given in the parentheses) were calculated based on 5 independent runs.

Method	GAN	Bayesian GAN	ProbGAN	EBGAN(KL)	EBGAN(Gaussian)
IS	7.525 (0.011)	7.450 (0.044)	7.384 (0.056)	7.606 (0.035)	7.712 (0.024)
1-WD	6.360 (0.012)	6.363 (0.015)	6.367 (0.024)	6.356 (0.015)	6.287 (0.025)
MMD	0.276 (0.002)	0.277 (0.002)	0.276 (0.002)	0.257 (0.001)	0.275 (0.003)

We further assess the quality of images generated by different methods using three metrics: inception scores (IS) (Salimans et al., 2016), first moment Wasserstein distance (1-WD), and maximum mean discrepancy (MMD). Refer to Section S2 of the supplement for their definitions and calculation procedures. The results are summarized in Table 1, which indicates the superiority of the EBGAN in image generation.

4.4 Nonparametric Clustering

Clustering has been extensively studied in unsupervised learning with classical methods such as expectation-maximization (EM) (Dempster et al., 1977) and K-means. Although its main focus is to group data into different classes, it would be even more beneficial if clustering was done along with dimension reduction, as it enhances the interpretability of clusters. This simultaneous goal of clustering and dimension reduction can be achieved through generative clustering methods such as Cluster GAN (Mukherjee et al., 2019) and GAN-EM (Zhao et al., 2019).

Since the cluster structure is generally not retained in the latent space of the GAN, Cluster GAN modifies the structure of the GAN to include an encoder network, which enforces precise recovery of the latent vector so that it can be used for clustering. Let the encoder be parameterized by θ_e. Cluster GAN works with the following two objective functions:

$$\begin{aligned} \mathcal{J}_d(\theta_d;\theta_g) &= \mathbb{E}_{x\sim p_{data}}\phi_1(D_{\theta_d}(x)) + \mathbb{E}_{x\sim p_{\theta_g}}\phi_2(D_{\theta_d}(x)), \\ \mathcal{J}_{g,e}(\theta_g,\theta_e;\theta_d) &= -\mathbb{E}_{x\sim p_{data}}\phi_1(D_{\theta_d}(x)) + \mathbb{E}_{x\sim p_{\theta_g}}\phi_3(D_{\theta_d}(x)) \\ &\quad - \beta_n\mathbb{E}_{x\sim p_{\theta_g}}\|z_n - \mathcal{E}^{(1)}_{\theta_e}(x)\|^2 - \beta_c\mathbb{E}_{x\sim p_{\theta_g}}\mathcal{H}(z_c,\mathcal{E}^{(2)}_{\theta_e}(x)), \end{aligned} \tag{4.15}$$

where $z=(z_n,z_c)$ is used as the latent vector to generate data, z_n denotes a random noise vector, and z_c denotes an one-hot vector representing the index of clusters; $\mathcal{H}(\cdot,\cdot)$ is the cross-entropy loss; and $\beta_n \geq 0$ and $\beta_c \geq 0$ are regularization parameters. The choice of β_n and β_c should balance the two ends: large values of β_n and β_c will delay the convergence of the generator, while small values of β_n and β_c will delay the convergence of the encoder. In general, we set $\beta_n = o(1)$ and $\beta_c = o(1)$. In this setup, the encoder is the inverse of the generator so that it recovers from data to the low-dimensional latent vector by $\mathcal{E}_{\theta_e}(x) = (\mathcal{E}_{\theta_e}^{(1)}(x), \mathcal{E}_{\theta_e}^{(2)}(x)) : \mathbb{R}^{d\times d} \to z = (\widehat{z}_n, \widehat{z}_c)$, where $\mathcal{E}_{\theta_e}^{(2)}(x)$ can be used for clustering.

Cluster EBGAN extends the structure of Cluster GAN by allowing multiple generators to be trained simultaneously. Similar to (2.6)-(2.7), cluster EBGAN works by solving the following integral optimization problem

$$\tilde{\theta}_d = \arg\max_{\theta_d} \int \mathcal{J}_d(\theta_d;\theta_g)\pi(\theta_g,\theta_e|\theta_d,\mathcal{D})d\theta_g d\theta_e, \tag{4.16}$$

and then simulate (θ_g, θ_e) from the distribution

$$\pi(\theta_g,\theta_e|\tilde{\theta}_d,\mathcal{D}) \propto \exp\{\mathbb{J}_{g,e}(\theta_g,\theta_e;\tilde{\theta}_d)\}p_{g,e}(\theta_g,\theta_e), \tag{4.17}$$

where $\mathbb{J}_{g,e}(\theta_g,\theta_e;\theta_d) = N\mathcal{J}_{g,e}(\theta_g,\theta_e;\theta_d)$ and $p_{g,e}(\theta_g,\theta_e)$ denotes the prior density function of (θ_g,θ_e). Let $\pi_c(\theta_g|\tilde{\theta}_d,\mathcal{D}) = \int \pi(\theta_g,\theta_e|\tilde{\theta}_d,\mathcal{D})d\theta_e$ be the marginal conditional density function of θ_g. Then, by Theorem 1 and

Corollary 1, $(\tilde{\theta}_d, \pi_c(\theta_g|\tilde{\theta}_d, \mathcal{D}))$ is an asymptotic solution to the game (2.4) as $N \to \infty$. Note that for $\pi_c(\theta_g|\tilde{\theta}_d, \mathcal{D})$, we can simply treat $q_g(\theta_g) \propto \int \exp\{-N\beta_n \mathbb{E}_{x\sim p_{\theta_g}} \|z_n - \mathcal{E}^{(1)}_{\theta_e}(x)\|^2 - N\beta_c \mathbb{E}_{x\sim p_{\theta_g}} \mathcal{H}(z_c, \mathcal{E}^{(2)}_{\theta_e}(x))\} p_{g,e}(\theta_g, \theta_e) d\theta_e$ as the prior of θ_g. Therefore, Theorem 1 and Corollary 1 still apply.

The equations (4.16)-(4.17) provide a general formulation for extended applications of the EBGAN. In particular, the embedded decoder (latent variable → fake data) and encoder (fake data → latent variable) enable the EBGAN to be used in many nonparametric unsupervised statistical tasks. Other than clustering, it can also be used for tasks such as dimension reduction and image compression.

Classical clustering methods can be roughly grouped into three categories, namely, partitional clustering, hierarchical clustering, and density-based clustering. The K-means clustering, agglomerative clustering, and density-based spatial clustering of applications with noise (DBSCAN) are well known representatives of the three categories, respectively. In what follows, Cluster-EBGAN is compared with the representative clustering methods as well as Cluster GAN on four different datasets. For MNIST, we used a deep convolutional GAN (DCGAN) with conv-deconv layers, batch normalization and leaky relu activations. For other datasets, simple feed forward neural networks were used. The results are summarized in Table

2, which indicates the superiority of the Cluster-EBGAN in nonparametric clustering. Refer to the supplement for the details of the experiments.

Table 2: Comparison of Cluster EBGAN and other methods on different datasets, where average purity, adjusted rand index (ARI) and their standard errors (in parentheses) were computed based on five independent runs.

Data	Metric	K-means	Agglomerative	DBSCAN	Cluster-GAN	Cluster-EBGAN
Iris	Purity	0.8933	0.8933	0.8867	0.8973(0.041)	**0.9333(0.023)**
	ARI	0.7302	0.7312	0.5206	0.5694(0.169)	**0.8294(0.050)**
Seeds	Purity	0.8905	0.8714	-	0.7686(0.049)	**0.9105(0.005)**
	ARI	0.7049	0.6752	-	0.4875(0.090)	**0.7550(0.011)**
MNIST	Purity	0.5776	0.7787	-	0.7217(0.02)	**0.8826(0.02)**
	ARI	0.3607	0.5965	-	0.5634(0.02)	**0.7780(0.03)**

5. Conclusion

This paper has identified the reasons why the GAN suffers from the mode collapse issue and proposes a new formulation to address this problem. Additionally, an empirical Bayes-like method is proposed for training the GAN under the new formulation. The proposed new formulation is general, allowing for easy reformulation and training of various GAN variants such as Lipschitz GAN and cluster GAN using the proposed empirical Bayes-like method.

The proposed empirical Bayes-like method can be extended in various ways. For example, the generator can be simulated using other stochastic gradient MCMC algorithms such as SGHMC (Chen et al., 2014) and preconditioned SGLD (Li et al., 2016); and the discriminator can be trained using an advanced SGD algorithm such as Adam (Kingma and Ba, 2014), AdaMax (Kingma and Ba, 2014) and Adadelta (Zeiler, 2012). Moreover, the proposed method can be easily extended to learn sparse generators by imposing an appropriate prior distribution on the generator. Refer to Sun et al. (2022a,b) for prior settings for consistent sparse deep learning.

In summary, this paper has presented a new formulation for the GANs as randomized decision problems, and proposed an effective method to solve them. From the perspective of statistical decision theory, further investigation into the application of the proposed method to other classes of risk functions would be of great interest. We anticipate that the proposed method will find wide applications in the field of statistical decision science.

Supplementary Materials

The supplementary material contains the proofs of the theoretical results and provides additional numerical examples

Acknowledgements

This research is supported in part by the NSF grants DMS-1811812 (Song) and DMS-2015498 (Liang) and the NIH grant R01-GM126089 (Liang). The authors thank the reviewers for their insightful and helpful comments.

References

Andrieu, C., E. Moulines, and P. Priouret (2005). Stability of stochastic approximation under verifiable conditions. *SIAM Journal on Control and Optimization 44*(1), 283–312.

Arjovsky, M., S. Chintala, and L. Bottou (2017, 06–11 Aug). Wasserstein generative adversarial networks. In D. Precup and Y. W. Teh (Eds.), *Proceedings of the 34 th International Conference on Machine Learning (ICML)*, Volume 70 of *PMLR*, International Convention Centre, Sydney, Australia, pp. 214–223.

Arnold, B. C. and S. J. Press (1989). Compatible conditional distributions. *Journal of the American Statistical Association 84*(405), 152–156.

Arora, S., R. Ge, Y. Liang, T. Ma, and Y. Zhang (2017). Generalization and equilibrium in generative adversarial nets (GANs). In *ICML*, pp. 224–232.

Bellot, A. and M. van der Schaar (2019). Conditional independence testing using generative adversarial networks. In *NeurIPS*, pp. 2202–2211.

Benveniste, A., M. Métivier, and P. Priouret (1990). *Adaptive Algorithms and Stochastic Approximations.* Springer.

Binkowski, M., D. J. Sutherland, M. Arbel, and A. Gretton (2018). Demystifying MMD GANs. In *ICLR*.

Che, T., Y. Li, A. P. Jacob, Y. Bengio, and W. Li (2017). Mode regularized generative adversarial networks. In *ICLR*.

Chen, T., E. B. Fox, and C. Guestrin (2014). Stochastic gradient hamiltonian monte carlo. In *ICML*.

Dempster, A., N. Laird, and D. Rubin (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B 39*, 1–38.

Deng, W., X. Zhang, F. Liang, and G. Lin (2019). An adaptive empirical bayesian method for sparse deep learning. *NeurIPS 2019*.

Dong, T., P. Zhang, and F. Liang (2023). A stochastic approximation-langevinized ensemble kalman filter for state space models with unknown parameters. *Journal of Computational and Graphical Statistics 32*(2), 448–469.

Gao, Y., Y. Jiao, Y. Wang, Y. Wang, C. Yang, and S. Zhang (2019). Deep generative learning via variational gradient flow. In *ICML*, pp. 2093–2101.

Ghosh, A., V. Kulharia, V. Namboodiri, P. Torr, and P. Dokania (2018, 06). Multi-agent diverse generative adversarial networks. In *CVPR*, pp. 8513–8521.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative adversarial nets. *NIPS*, 2672–2680.

He, H., H. Wang, G.-H. Lee, and Y. Tian (2019). Probgan: Towards probabilistic gan with theoretical guarantees. *ICLR*.

Hoang, Q., T. D. Nguyen, T. Le, and D. Phung (2018). MGAN: Training generative adversarial nets with multiple generators. In *ICLR*.

Kim, S., Q. Song, and F. Liang (2022). Stochastic gradient langevin dynamics with adaptive drifts. *Journal of Statistical Computation and Simulation 92*(2), 318–336.

Kingma, D. and J. Ba (2014). Adam: A method for stochastic optimization. *ICLR*, 1–13.

Kirkpatrick, S., C. Gelatt, and M. Vecchi (1983). Optimization by simulated annealing. *Science 220*, 671–680.

Li, C., C. Chen, D. Carlson, and L. Carin (2016). Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *AAAI*.

Liu, Q., J. Xu, R. Jiang, and W. H. Wong (2021). Density estimation using deep generative neural networks. *Proceedings of the National Academy of Sciences 118*(15), e2101344118.

Mao, X., Q. Li, H. Xie, R. Lau, W. Zhen, and S. Smolley (2017). Least squares generative adversarial networks. In *ICCV*, pp. 2813–2821.

Morris, C. N. (1983). Parametric empirical bayes inference: Theory and applications. *Journal of the American Statistical Association 78*(381), 47–55.

Mukherjee, S., H. Asnani, E. Lin, and S. Kannan (2019). Clustergan: Latent space clustering in generative adversarial networks. In *AAAI*, pp. 4610–4617. AAAI Press.

Nowozin, S., B. Cseke, and R. Tomioka (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *NIPS*.

Pérez-Cruz, F. (2008). Kullback-leibler divergence estimation of continuous distributions. *2008*

IEEE International Symposium on Information Theory, 1666–1670.

Saatci, Y. and A. G. Wilson (2017). Bayesian gan. In *NIPS*, pp. 3622–3631.

Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen (2016). Improved techniques for training gans. In *NIPS*, pp. 2234–2232.

Singh, S., A. Uppal, B. Li, C. Li, M. Zaheer, and B. Póczos (2018). Nonparametric density estimation under adversarial losses. In *NeurIPS*.

Song, Q., Y. Sun, M. Ye, and F. Liang (2020). Extended stochastic gradient mcmc for large-scale bayesian variable selection. *Biometrika 107*(4), 997–1004.

Sun, Y., Q. Song, and F. Liang (2022a). Consistent sparse deep learning: Theory and computation. *Journal of the American Statistical Association 117*(540), 1981–1995.

Sun, Y., Q. Song, and Y. Liang (2022b). Learning sparse deep neural networks with a spike-and-slab prior. *Statistics & Probability Letters 180*, 109246.

Tolstikhin, I., S. Gelly, O. Bousquet, C. J. Simon-Gabriel, and B. Schölkopf (2017). Adagan: Boosting generative models. In *NIPS*, pp. 5424–5433.

Wang, Q., S. R. Kulkarni, and S. Verdú (2009). Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Transactions on Information Theory 55*, 2392–2405.

Wang, Y., L. Zhang, and J. van de Weijer (2016). Ensembles of generative adversarial networks. *ArXiv abs/1612.00991*.

Welling, M. and Y. W. Teh (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*.

Wiatrak, M. and S. V. Albrecht (2019). Stabilizing generative adversarial network training: A survey. *ArXiv abs/1910.00927*.

Young, G. and R. Smith (2005). *Essentials of Statistical Inference*. London: Cambridge University Press.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR abs/1212.5701*.

Zhao, W., S. Wang, Z. Xie, J. Shi, and C. Xu (2019). GAN-EM: GAN based EM learning framework. In *IJCAI*, pp. 4404–4411.

Zhou, Z., J. Liang, Y. Song, L. Yu, H. Wang, W. Zhang, Y. Yu, and Z. Zhang (2019). Lipschitz generative adversarial nets. In *ICML*, pp. 7584–7593.

Sehwan Kim, Purdue University, West Lafayette, IN 4797

E-mail: kim3009@purdue.edu

Qifan Song, Purdue University, West Lafayette, IN 47907

E-mail: qfsong@purdue.edu

Faming Liang, Purdue University, West Lafayette, IN 47907

E-mail: fmliang@purdue.edu