Geometric Enclosing Networks

Trung Le[1], **Hung Vu**[2], **Tu Dinh Nguyen**[1] and **Dinh Phung**[1]
[1] Faculty of Information Technology, Monash University
[2] Center for Pattern Recognition and Data Analytics, Deakin University, Australia
{trunglm, tu.dinh.nguyen, dinh.phung}@monash.edu, hungv@deakin.edu.au

Abstract

Training model to generate data has increasingly attracted research attention and become important in modern world applications. We propose in this paper a new geometry-based optimization approach to address this problem. Orthogonal to current state-of-the-art density-based approaches, most notably VAE and GAN, we present a fresh new idea that borrows the principle of minimal enclosing ball to train a generator $G(\boldsymbol{z})$ in such a way that both training and generated data, after being mapped to the feature space, are enclosed in the same sphere. We develop theory to guarantee that the mapping is bijective so that its inverse from feature space to data space results in expressive nonlinear contours to describe the data manifold, hence ensuring data generated are also lying on the data manifold learned from training data. Our model enjoys a nice geometric interpretation, hence termed *Geometric Enclosing Networks* (GEN), and possesses some key advantages over its rivals, namely simple and easy-to-control optimization formulation, avoidance of mode collapsing and efficiently learn data manifold representation in a completely unsupervised manner. We conducted extensive experiments on synthesis and real-world datasets to illustrate the behaviors, strength and weakness of our proposed GEN, in particular its ability to handle multi-modal data and quality of generated data.

1 Introduction

Density estimation has been being a dominant approach in statistical machine learning since its inception due to its sound theoretical underpinnings and the ability to *explain* the data. This translates into the estimation of a distribution $\tilde{P}(\boldsymbol{x})$ as 'close' as possible to the true, but *unknown*, data distribution $P_{\text{data}}(\boldsymbol{x})$. Among other advantages, an important consequence of this approach is the ability to *generate* data by sampling from $\tilde{P}(\boldsymbol{x})$. In recent years, this data generation need has been growing rapidly with the scale and magnitude of modern applications. However, to this end, the computational aspect of several existing density estimation approaches becomes problematic (i.e., they cannot scale satisfactorily to ImageNet, a large-scale visual dataset) [Goodfellow, 2017].

What becomes a 'daring' idea is the recent success of an approach that aims to generate data directly *without* estimating $\tilde{P}(\boldsymbol{x})$ in analytical forms, pioneered most notably by Variational Autoencoder (VAE) [Kingma and Welling, 2014] and Generative Adversarial Nets (GANs) [Goodfellow *et al.*, 2014]. There are some key differences between VAE and GAN, but in the end, both can be used to generate data by first sampling $\boldsymbol{z}$ i.i.d from a noise space, then feeding $\boldsymbol{z}$ through a *generator* $G(\boldsymbol{z})$[1] parameterized by a neural net (NN). Let P_g be the distribution over the values of $G(\boldsymbol{z})$; then although P_g is not directly modeled, VAE and GAN's objective functions minimize a suitable distance between P_g and the true P_{data}. Both have enjoyed enormous recent popularity due to its scalability and most importantly, it is extremely efficient to generate data using the generator G in a single shot. Nonetheless, training VAE and GAN is still a challenging problem and taming the model to generate meaningful outputs is still like a black art. In particular, GAN suffers from two issues: convergence and mode collapsing[2] [Goodfellow, 2017]. There has been several recent attempts to address these problems. For example, modifying the criteria to measure the distance between P_g and P_{data} yields different variations of GANs and there has been a surging interest along this line of research (e.g., fGAN [Nowozin *et al.*, 2016], InfoGAN [Chen *et al.*, 2016], Wasserstein GAN [Arjovsky *et al.*, 2017], D2GAN [Nguyen *et al.*, 2017], MGAN [Hoang *et al.*, 2018]); others have tried to address the convergence and mode collapsing via modifying the optimization process (e.g., UnrolledGAN [Metz *et al.*, 2016]).

Kernel method with its mature principle [Vapnik, 1995; Cortes and Vapnik, 1995] has been broadly applied to a wide range of applications in machine learning and data analytics. The key principle of kernel method is that a simple geometric shape (e.g., hyperplane or hypersphere) in feature space when being mapped back to input space forms a set of nonlinear contours characterizing data. This principle was further exploited in support vector clustering [Ben-Hur *et al.*, 2001], wherein learning a minimal enclosing ball in the feature space

[1]The decoder $p(\boldsymbol{x} \mid \boldsymbol{z})$ in case of VAE.

[2]i.e., the generator might generate all data to a single mode of P_{data} while still guarantees that they have high likelihood.

can induce a set of nonlinear contours that capture data manifolds and clusters. There have been some recent attempts [Li *et al.*, 2015; Dziugaite *et al.*, 2015] to leverage kernel method with generative model. In nature, these works [Li *et al.*, 2015; Dziugaite *et al.*, 2015] base on kernel method to define maximum mean discrepancy (MMD), which is a frequentist estimator to measure the mean square difference of the statistics of the two sets of samples, and then minimizing this MMD to diminish the divergence between P_g and P_{data}.

This paper takes a radical departure from the density estimation view to data generation problem. The goal remains the same: we aim to train a generator $G(\boldsymbol{z})$ to be used to generate data efficiently through i.i.d $\boldsymbol{z}$ lying in any arbitrary uniform or noise space. However, unlike existing work, our approach does *not* model the relation between P_g and P_{data}, instead we work directly with *geometric* structure of the data. In particular, we depart from the original idea of support vectors in [Ben-Hur *et al.*, 2001] and formulate an optimization framework to ensure that the generated value $G(\boldsymbol{z})$ will be contained within the data manifold learned from training data. As in VAE or GAN, our approach is completely unsupervised, hence to richly characterize this (nonlinear) data manifold, we borrow the idea of constructing a *minimal enclosing ball* $\mathcal{B}$ in the feature space whose theory guarantees the inverse mapping to the data space produces nonlinear contours rich enough to describe the data manifold [Ben-Hur *et al.*, 2001]. Our high-level intuition is then to learn a generator $G(\boldsymbol{z})$ in such a way that its values, when being mapped to the same feature space, are also enclosed in the same ball $\mathcal{B}$, consequently $G(\boldsymbol{z})$ is guaranteed to lie in the data manifolds.

Directly using the primal form to construct $\mathcal{B}$ as in [Ben-Hur *et al.*, 2001] is, however, not tractable for our purpose since we need the mapping function $\Phi(\boldsymbol{x})$ from the input space to feature space to be explicit. Furthermore, this function must facilitate efficient parameter estimation procedure such as through backpropagation. We overcome this obstacle by using recent Fourier random feature representation proposed in [Rahimi and Recht, 2007] to approximate $\Phi(\boldsymbol{x})$ with an explicit $\tilde{\Phi}(\boldsymbol{x})$ in a finite-dimensional space. To enable efficient learning via gradient descent and backpropagation, we use the idea of reparameterization in [Kingma and Welling, 2014] to reformulate $\tilde{\Phi}(\boldsymbol{x})$. Altogether, we arrive at an optimization framework which can be efficiently solved via two stages: learn the enclosing ball $\mathcal{B}$ from data via random feature reparameterization, followed by backpropagation to train the generator G. We term our approach *Geometric Enclosing Networks* (GEN) to reflect the fact that the spirit of our approach is indeed corresponding to geometric intuition.

In addition, using the approximate mapping $\tilde{\Phi}(\boldsymbol{x})$ could however potentially result in a technical problem since the theory of [Ben-Hur *et al.*, 2001] requires $\Phi(\boldsymbol{x})$ to be bijective. To this end, we provide theoretical analysis to guarantee that the approximate mapping $\tilde{\Phi}(\boldsymbol{x})$ is a bijection. In addition to the construction of GEN, both the random feature reparameterization and the bijection result for the construction of minimal enclosing ball are also novel contributions, to our knowledge. We conducted experiments to demonstrate the behaviors of our proposed approach using synthetic and real-world datasets. We demonstrate how our GEN can avoid mode collapsing problem and result in better data generation quality via comparison with the true data (when using synthesis data) and visual inspection on MNIST, Frey Face, CIFAR-10, and CelebA datasets.

Compared with implicit density estimation approaches, epitomized by GAN, our proposed GEN possesses some key advantages. First it presents an orthogonal, fresh new idea to the problem of data generation via geometric intuition as opposed to density estimation view. Second, our optimization is simpler, much easier to control and it enjoys the maturity of the optimization field at its hand. Third, it can easily avoid the mode collapsing problem and efficiently learn manifold in the data in a completely unsupervised manner. Lastly, it opens up various promising future work on geometry-based approach to data generation.

2 Related Background

Related closely to the technical development of our model is the theory of support vectors and minimal enclosing ball [Vapnik, 1995; Cortes and Vapnik, 1995; Ben-Hur *et al.*, 2001], which we shall briefly describe in Section 2.1. The original primal form of [Vapnik, 1995; Cortes and Vapnik, 1995; Ben-Hur *et al.*, 2001], however, uses *implicit* infinitely-dimensional mapping $\Phi(\cdot)$, hence impeding the use of backpropagation and gradient descent training. To overcome this obstacle, we briefly revise the Fourier random feature representation of [Rahimi and Recht, 2007] in Section 2.2.

2.1 Minimal Enclosing Ball Optimization

Given an unlabeled training set $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, [Ben-Hur *et al.*, 2001] formulated an optimization to learn the data description via a minimal enclosing ball $\mathcal{B}$ of the feature vectors $\{\Phi(\boldsymbol{x}_1), \Phi(\boldsymbol{x}_2), \ldots, \Phi(\boldsymbol{x}_N)\}$ where Φ is a feature map from the input space to the feature space:

$$\begin{aligned}&\min_{R,\boldsymbol{c},\boldsymbol{\xi}}\left(\lambda R^2 + \frac{1}{N}\sum_{i=1}^{N}\xi_i\right)\\ &\text{s.t.}: \left\|\Phi(\boldsymbol{x}_i) - \boldsymbol{c}\right\|^2 \le R^2 + \xi_i,\ i = 1, ..., N;\\ &\quad \xi_i \ge 0,\ i = 1, ..., N\end{aligned}$$

where R, $\boldsymbol{c}$ are the radius and center of the minimal enclosing ball respectively, $\boldsymbol{\xi} = [\xi_i]_{i=1}^{N}$ is the vector of slack variables, and $\lambda > 0$ is the trade-off parameter. Our work utilizes its primal form, which can be stated as:

$$\min_{R,\boldsymbol{c}}\left(\lambda R^2 + \frac{1}{N}\sum_{i=1}^{N}\max\{0, \left\|\Phi(\boldsymbol{x}_i) - \boldsymbol{c}\right\|^2 - R^2\}\right) \quad (1)$$

For our interest, an important result in [Ben-Hur *et al.*, 2001] is that minimal enclosing ball in the feature space, when being mapped back the input space, generates nonlinear contours that can be interpreted as the clusters or data manifold of the training set. This principle has been proven to be able to learn nested, or complicated clusters and data manifolds in high dimensional space [Ben-Hur *et al.*, 2001].

2.2 Fourier Random Feature Representation

The mapping $\Phi(\boldsymbol{x})$ above is implicitly defined and the inner product $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle$ is evaluated through a kernel $K(\boldsymbol{x}, \boldsymbol{x}')$. To construct an explicit representation of $\Phi(\boldsymbol{x})$, the key idea is to approximate the symmetric and positive semi-definite (p.s.d) kernel $K(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x} - \boldsymbol{x}')$ with $K(\mathbf{0}, \mathbf{0}) = k(\mathbf{0}) = 1$ using a kernel induced by a random finite-dimensional feature map [Rahimi and Recht, 2007]. The mathematical tool behind this approximation is the Bochner's theorem [Bochner, 1959], which states that every shift-invariant, p.s.d kernel $K(\boldsymbol{x}, \boldsymbol{x}')$ can be represented as an inverse Fourier transform of a proper distribution $p(\boldsymbol{\omega})$ as below:

$$K(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{u}) = \int p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^\top \boldsymbol{u}} d\boldsymbol{\omega} \tag{2}$$

where $\boldsymbol{u} = \boldsymbol{x} - \boldsymbol{x}'$ and i represents the imaginary unit (i.e., $i^2 = -1$). In addition, the corresponding proper distribution $p(\boldsymbol{\omega})$ can be recovered through Fourier transform of kernel function as:

$$p(\boldsymbol{\omega}) = \left(\frac{1}{2\pi}\right)^d \int k(\boldsymbol{u}) e^{-i\boldsymbol{u}^\top \boldsymbol{\omega}} d\boldsymbol{u} \tag{3}$$

Popular shift-invariant kernels include Gaussian, Laplacian and Cauchy. For our work, we employ Gaussian kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{u}) = \exp\left[-\frac{1}{2}\boldsymbol{u}^\top \Sigma \boldsymbol{u}\right]$ parameterized by the covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$. With this choice, substituting into Eq. (3) yields a closed-form for the probability distribution $p(\boldsymbol{\omega})$ which is $\mathcal{N}(\mathbf{0}, \Sigma)$.

This suggests a Monte-Carlo approximation to the kernel in Eq. (2):

$$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{x}') &= \mathbb{E}_{\boldsymbol{\omega}\sim p(\boldsymbol{\omega})}\left[\cos\left(\boldsymbol{\omega}^\top (\boldsymbol{x} - \boldsymbol{x}')\right)\right] \\ &\approx \frac{1}{D}\sum_{i=1}^{D}\left[\cos\left(\boldsymbol{\omega}_i^\top (\boldsymbol{x} - \boldsymbol{x}')\right)\right] \end{aligned} \tag{4}$$

where we have sampled $\boldsymbol{\omega}_i \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\omega} \mid \mathbf{0}, \Sigma)$ for $i \in \{1, 2, ..., D\}$. Eq. (4) sheds light on the construction of a $2D$-dimensional random map $\tilde{\Phi} : \mathcal{X} \to \mathbb{R}^{2D}$:

$$\tilde{\Phi}(\boldsymbol{x}) = \left[\frac{1}{\sqrt{D}}\cos\left(\boldsymbol{\omega}_i^\top \boldsymbol{x}\right), \frac{1}{\sqrt{D}}\sin\left(\boldsymbol{\omega}_i^\top \boldsymbol{x}\right)\right]_{i=1}^{D} \tag{5}$$

resulting in the approximate kernel $\tilde{K}(\boldsymbol{x}, \boldsymbol{x}') = \tilde{\Phi}(\boldsymbol{x})^\top \tilde{\Phi}(\boldsymbol{x}')$ that can accurately and efficiently approximate the original kernel: $\tilde{K}(\boldsymbol{x}, \boldsymbol{x}') \approx K(\boldsymbol{x}, \boldsymbol{x}')$ [Rahimi and Recht, 2007]. While we now have an explicit mapping $\tilde{\Phi}(\boldsymbol{x})$, it is not yet an explicit function of Σ, hence does not facilitate learning this kernel parameter via gradient descent. In Section 3.2, we further develop reparameterized version of this random feature representation such that $\tilde{\Phi}(\boldsymbol{x})$ becomes an explicit function of the kernel parameter Σ, hence can be learned from data.

3 Geometric Enclosing Networks

We present our proposed GEN in this section, starting with a high-level geometric intuition. This is followed by detailed description of the framework, its algorithm and implementation details. Finally, theoretical analysis is presented.

3.1 High-level Geometric Intuition

To strengthen the intuition described earlier, Figure 1 summarizes the key high-level intuition of our approach. Given training data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$ and assume there is an explicit feature map $\tilde{\Phi}(\boldsymbol{x})$ from the input space to feature space, our network first learns a *minimal enclosing ball* $\mathcal{B}$ of the training data samples in the random feature space, i.e., the ball with minimal radius that encloses $\tilde{\Phi}(\boldsymbol{x}_1), \ldots \tilde{\Phi}(\boldsymbol{x}_n)$. Next, we train a generator $G_\psi(\boldsymbol{z})$ such that its generated samples $G_\psi(\boldsymbol{z}_i)$, once being mapped via $\tilde{\Phi}$ to produce $\tilde{\Phi}(G_\psi(\boldsymbol{z}_i))$ also fall into $\mathcal{B}$ where $\boldsymbol{z}$ are drawn i.i.d from any arbitrary noise space. So long as $\tilde{\Phi}$ is a bijective mapping (as shown in Theorem 3), the generated samples $G(\boldsymbol{z}) = \tilde{\Phi}^{-1}(\tilde{\Phi}(G(\boldsymbol{z})))$ must locate in the contours characterizing the true data samples in the input space.

Geometrically, all points in the feature space including those mapped from training data (green) and generated by the generator $G(\boldsymbol{z})$ (yellow) are enclosed in the ball $\mathcal{B}$. Because $\tilde{\Phi}(\boldsymbol{x})$ lies in the unit hypersphere in the random feature space for all $\boldsymbol{x} \in \mathcal{X}$, the mapping of training data points and generated data points also must lie on the surface of this unit hypersphere inside $\mathcal{B}$. Those points at the section between $\mathcal{B}$ and the unit hypersphere are called *support vectors.*

Our optimization framework then consists of two sub-optimization problems: learn the explicit mapping $\tilde{\Phi}$ and the minimal enclosing ball $\mathcal{B}$ in the first step and then train a generator $G(\boldsymbol{z})$ in the second step which we detail in Sections 3.2 and 3.3 respectively.

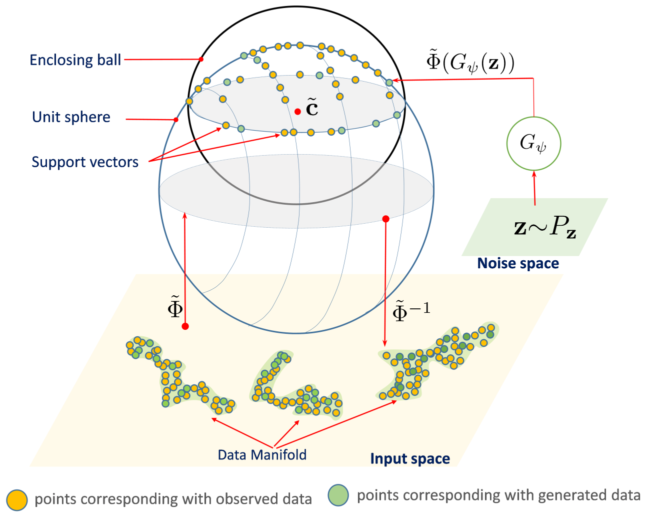

Enclosing ball
Unit sphere
Support vectors
$\tilde{\mathbf{c}}$
$\tilde{\Phi}(G_\psi(\mathbf{z}))$
G_ψ
$\mathbf{z}\sim P_\mathbf{z}$
Noise space
$\tilde{\Phi}$
$\tilde{\Phi}^{-1}$
Data Manifold
Input space
points corresponding with observed data
points corresponding with generated data

Figure 1: Geometric structure of our proposed network (GEN). The *explicit* mapping $\tilde{\Phi}$ (and its inverse) is formulated and learned via reparameterized random feature representation together with the minimal enclosing ball $\mathcal{B}$. The generator $G(\boldsymbol{z})$ is then trained via back propagation so that its image is contained within $\mathcal{B}$. This ensures generated sample $G(\boldsymbol{z})$ belongs to the contours characterizing the data manifold in the input space. Points at the intersection of $\mathcal{B}$ and the unit sphere are called *support vectors.* (best viewed in color).

3.2 Learning $\tilde{\Phi}$ and $\mathcal{B}$ via Random Feature Reparameterization

Recall that our first goal is to find a minimal enclosing ball $\mathcal{B}$ by solving the optimization problem in Eq. (1). However, in its primal form, the feature map Φ is unknown and our approach seeks for an approximate *explicit* mapping via random feature representation as in Eq. (5). Although $\tilde{\Phi}(\boldsymbol{x})$ has an explicit form, it is represented indirectly through samples drawn from $p(\boldsymbol{\omega} \mid \Sigma)$, hence it is still not possible to learn Σ through gradient descent yet. To do this, we need $\tilde{\Phi}(\boldsymbol{x})$ to be a direct function of the kernel parameter Σ. We apply the simple reparameterization trick [Kingma and Welling, 2014] to shift the source of randomness $\boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{\omega} \mid \boldsymbol{0}, \Sigma)$ to $\boldsymbol{\omega} = \boldsymbol{0} + L\boldsymbol{e}$ where $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$ and $L = \Sigma^{1/2}$ so that the kernel in Eq. (2) can now be re-written as:

$$K(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{u}) = \int \mathcal{N}(\boldsymbol{e} \mid \boldsymbol{0}, \boldsymbol{I}_d)\, e^{i(L\boldsymbol{e})^\top \boldsymbol{u}} d\boldsymbol{e}$$
$$= \int \mathcal{N}(\boldsymbol{e} \mid \boldsymbol{0}, \boldsymbol{I}_d) \left[\cos\left(\boldsymbol{e}^\intercal L\boldsymbol{u}\right) + i \sin\left(\boldsymbol{e}^\intercal L\boldsymbol{u}\right)\right] d\boldsymbol{e}$$
$$= \int \mathcal{N}(\boldsymbol{e} \mid \boldsymbol{0}, \boldsymbol{I}_d) \cos\left[\boldsymbol{e}^\intercal L(\boldsymbol{x} - \boldsymbol{x}')\right] d\boldsymbol{e} \tag{6}$$

which can be again approximated as

$$K(\boldsymbol{x}, \boldsymbol{x}') \approx \tilde{K}(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \tilde{\Phi}(\boldsymbol{x}), \tilde{\Phi}(\boldsymbol{x}') \right\rangle$$
$$= \frac{1}{D} \sum_{i=1}^{D} \left[\cos(a_i \boldsymbol{x}) \cos(a_i \boldsymbol{x}') + \sin(a_i \boldsymbol{x}) \sin(a_i \boldsymbol{x}')\right]$$

where $\boldsymbol{e}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$, $a_i = \boldsymbol{e}_i^\intercal L, \forall i = 1, \dots, D$, and the random feature map now has an explicit representation as well as being a direct function of the kernel parameter Σ:

$$\tilde{\Phi}(\boldsymbol{x}) = D^{-1/2} \left[\cos\left(\boldsymbol{e}_i^\intercal L\boldsymbol{x}\right), \sin\left(\boldsymbol{e}_i^\intercal L\boldsymbol{x}\right)\right]_{i=1}^{D} \tag{7}$$

Using this new reparameterized feature map for $\tilde{\Phi}$ in Eq. (7), the optimization problem for $\mathcal{B}$ in Eq. (1) is now reformulated as follows:

$$\min_{\tilde{R}, \tilde{\boldsymbol{c}}, \Sigma} \mathcal{J}_d\left(\tilde{R}, \tilde{\boldsymbol{c}}, \Sigma\right) \tag{8}$$

where we have defined the new objective function:

$$\mathcal{J}_d\left(\tilde{R}, \tilde{\boldsymbol{c}}, \Sigma\right) = \lambda \tilde{R}^2 + \frac{1}{N} \sum_{i=1}^{N} \max\left(0, \|\tilde{\Phi}(\boldsymbol{x}_i) - \tilde{\boldsymbol{c}}\|^2 - \tilde{R}^2\right)$$

with $\tilde{\boldsymbol{c}} \in \mathbb{R}^{2D}$ is the center of $\mathcal{B}$ in the random feature space, and $\tilde{R}$ is its radius. We then apply SGD to solve the optimization problem in Eq. (8) as summarized in Algorithm 1.. It is worth noting that we only learn a diagonal matrix Σ for efficient computation.

3.3 Training Generator $G(\cdot)$ via Backprop

We recruit a neural network G parameterized by ψ (i.e., G_ψ) to model the generator. The noise sample $\boldsymbol{z}$ is i.i.d sampled from the noise distribution $P_{\boldsymbol{z}}$, which could be any arbitrary noise distribution (e.g., uniform or Gaussian noise). Given any sample $\boldsymbol{z} \sim P_{\boldsymbol{z}}$, we train the generator such that $\tilde{\Phi}(G_\psi(\boldsymbol{z}))$ falls into the ball $\mathcal{B}$ of $\left\{\tilde{\Phi}(\boldsymbol{x}_1), ..., \tilde{\Phi}(\boldsymbol{x}_N)\right\}$ in the random feature space obtained in the previous step. This reduces to the following optimization:

$$\min_{\psi} \left(\mathcal{J}_g(\psi) = \mathbb{E}_{P_{\boldsymbol{z}}}\left[\max(0, \left\|\tilde{\Phi}(G_\psi(\boldsymbol{z})) - \tilde{\boldsymbol{c}}\right\|^2 - \tilde{R}^2)\right]\right) \tag{9}$$

We train the generator $G(\cdot)$ in such a way that its generated samples $G(\boldsymbol{z})$(s) spread over the surface of the unit sphere lying in the ball $\mathcal{B}$. This ensures the inverse mapping $G(\boldsymbol{z}) = \tilde{\Phi}^{-1}(\tilde{\Phi}(G(\boldsymbol{z})))$ to stretch out the data manifolds. However, minimizing the objective function in Eq. (9) could lead to the fact that the generator simply maps into a small region inside the ball $\mathcal{B}$, which can induce a negligible cost. To overcome this issue, we augment the objective function using the quantity $\| \mathbb{E}_{P_{\text{data}}}[\tilde{\Phi}(\boldsymbol{x})] - \mathbb{E}_{P_{\boldsymbol{z}}}[\tilde{\Phi}(G(\boldsymbol{z}))] \|^2$ [Salimans *et al.*, 2016], which encourages the generator to produce samples that spread over the ball $\mathcal{B}$. We now can employ the backpropagation to learn ψ via minimizing the augmented objective function.

3.4 Algorithm and Implementation

The key steps of GEN are presented in Algorithm 1. In the first phase (i.e., the first L epochs), we learn the ball $\mathcal{B}$ using training data mapped onto the random feature space. It is worth noting that we update the variables $\tilde{R}, \tilde{\boldsymbol{c}}, \Sigma$ and ψ using the mini-batches. However, for the sake of simplicity, we present their single-point updates (cf. Lines 6, 7, 8, and 14 in Algorithm 1). In the second phase (i.e., the last $T - L$ epochs), we keep fixed the ball $\mathcal{B}$ and the kernel (i.e., keeping fixed $\tilde{R}, \tilde{\boldsymbol{c}}$ and Σ) and train the generator such that for every $\boldsymbol{z} \sim P_{\boldsymbol{z}}$, the random feature image $\tilde{\Phi}(G_\psi(\boldsymbol{z}))$ falls into the ball $\mathcal{B}$.

3.5 Theoretical Analysis

In what follows, we present the theoretical analysis regarding to the tightness of kernel approximation using Fourier random feature with the reparameterization trick and the conditions under which the original and random feature maps (i.e., Φ and $\tilde{\Phi}$) are bijections.

Theorem 1. *With a probability at least* $1 - 2^7\left(\frac{diam(\mathcal{X})\|\Sigma^{1/2}\|_F}{\theta}\right)^2 \exp\left(\frac{-D\theta^2}{4(d+2)}\right)$ *where we assume that* $0 < \theta \leq diam(\mathcal{X}) \left\|\Sigma^{1/2}\right\|_F$*, diam* $(\mathcal{X})$ *is the diameter of the compact set* $\mathcal{X}$*, and* $\left\|\Sigma^{1/2}\right\|_F$ *specifies the Frobenius norm of the matrix* $\Sigma^{1/2}$*, we have the following inequality:*

$$\sup_{\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}} \left|\tilde{K}(\boldsymbol{x}, \boldsymbol{x}') - K(\boldsymbol{x}, \boldsymbol{x}')\right| < \theta$$

From the representation of the random feature map $\tilde{\Phi}(\boldsymbol{x})$, it is obvious to verify that $\| \tilde{\Phi}(\boldsymbol{x}) \| = \tilde{K}(\boldsymbol{x}, \boldsymbol{x})^{1/2} = 1$. This is expressed through Lemma 2 which helps us construct the geometric view of our GEN.

Lemma 2. *For every* $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$*, the random feature map of* $\boldsymbol{x}$ *(i.e.,* $\tilde{\Phi}(\boldsymbol{x})$*) lies in the unit hypersphere with the center origin and the radius* 1.

We are further able to prove that $\tilde{\Phi} : \mathcal{X} \to \tilde{\Phi}(\mathcal{X})$ is a bijective feature map if $\text{rank}\{\boldsymbol{e}_1, ..., \boldsymbol{e}_D\} = d$ and $\left\|\Sigma^{1/2}\right\|_F \text{diam}(\mathcal{X}) \max_{1 \leq i \leq D} \|\boldsymbol{e}_i\| < 2\pi$ where $\text{diam}(\mathcal{X})$ denotes the diameter of the set $\mathcal{X}$ and $\left\|\Sigma^{1/2}\right\|_F$ denotes the

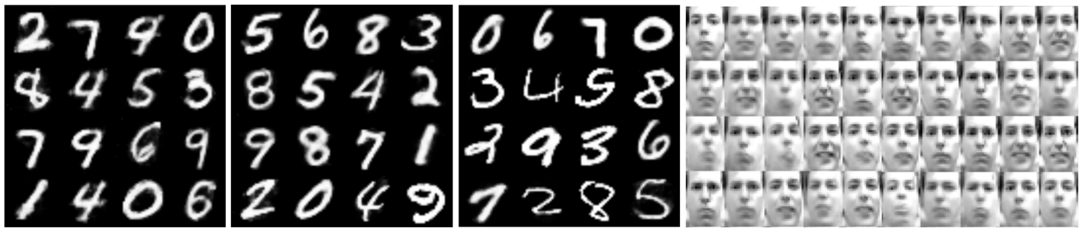


Figure 2: Data generated from our model when trained on MNIST dataset with $1,000$, $5,000$, and $60,000$ training examples (left) and Frey Face dataset (right).

Frobenius norm of the matrix $\Sigma^{1/2}$. This is stated in the following theorem.

Algorithm 1 Algorithm for GEN.

```
Input: $\lambda$, training data $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, random feature dimension $D$
Output: $\tilde{R}$, $\tilde{\boldsymbol{c}}$, $\Sigma$, $G_\psi$
 1: for $t = 1$ to $T$ do
 2:     if $t \leq L$ then
 3:         // Learn the ball $\mathcal{B}$
 4:         for $n = 1$ to $N$ do
 5:             Sample $\{\boldsymbol{x}^1, .., \boldsymbol{x}^b\}$ from training data $\mathcal{D}$
 6:             $\tilde{R} = \tilde{R} - \eta \nabla_{\tilde{R}} \mathcal{J}_d(\cdot)$
 7:             $\tilde{\boldsymbol{c}} = \tilde{\boldsymbol{c}} - \eta \nabla_{\tilde{\boldsymbol{c}}} \mathcal{J}_d(\cdot)$
 8:             $\Sigma = \Sigma - \eta \nabla_{\Sigma} \mathcal{J}_d(\cdot)$
 9:         end for
10:     else
11:         // Fix $\tilde{R}, \tilde{\boldsymbol{c}}, \Sigma$
12:         for $n = 1$ to $N$ do
13:             Sample $\{\boldsymbol{z}^1, .., \boldsymbol{z}^b\}$ from the noise distribution $P_{\boldsymbol{z}}$
14:             $\psi = \psi - \eta \nabla_{\psi} \mathcal{J}_g(\psi)$
15:         end for
16:     end if
17: end for
```

Theorem 3. *The following statements are guaranteed*

i) The original feature map $\Phi : \mathcal{X} \to \Phi(\mathcal{X})$ is a bijective mapping.

ii) If Σ is a non-singular matrix (i.e. positive definite matrix), $\left\|\Sigma^{1/2}\right\|_F$ diam $(\mathcal{X}) \max_{1 \leq i \leq D} \|\boldsymbol{e}_i\| < 2\pi$, and rank $\{\boldsymbol{e}_1, ..., \boldsymbol{e}_D\} = d$, then the random map as in Eq. (7) $\tilde{\Phi} : \mathcal{X} \to \tilde{\Phi}(\mathcal{X})$ is a bijection feature map.

Theorem 3 reveals that in order for the random feature map $\tilde{\Phi}$ to be a bijection we can either set the random feature dimension D to a sufficiently large value such that the random feature kernel $\tilde{K}(.,.)$ is sufficiently approximate the original kernel $K(.,.)$ or scales the data samples such that the inequality $\left\|\Sigma^{1/2}\right\|_F \operatorname{diam}(\mathcal{X}) \max_{1 \leq i \leq D} \|\boldsymbol{e}_i\| < 2\pi$ is satisfied. The supplementary material further presents the proofs for these claims.

4 Experimental Results

We conduct extensive experiments using both synthetic and real-world datasets to demonstrate the properties of our proposed network; in particular its ability to deal with multimodal data distribution (hence, avoiding mode collapsing problem), to model data manifold and the quality of generated samples. Unless otherwise specified, in all our experiments, when stochastic gradient descent was used, the ADAM optimizer [Kingma and Ba, 2014] with learning rate empirically turned by around 1e-3 and 1e-4 will be employed.

4.1 Synthetic Data

First, to see how well our GEN can deal with multiple modes in the data, we generate 10,000 samples drawn from a mixture of univariate Gaussians $0.45 \times \mathcal{N}(-0.6, 0.03) + 0.25 \times \mathcal{N}(0.7, 0.02) + 0.3 \times \mathcal{N}(0, 0.01)$ visualized as the blue curve on the left of Figure 4. Our baseline is GAN. The neural network specification for our generator $G(\boldsymbol{z})$ includes 2 hidden layers, each with 30 softplus units (and $D = 100$ for the number of random features, cf. Eq. (7)) and $\boldsymbol{z} \sim \text{Uni}(-1, 1)$. For GAN, we used a common setting [Goodfellow *et al.*, 2014] with one layer of 20 softplus hidden units for the generator, and 3 layers, each with 40 tanh hidden units for the discriminator.

Figure 4 (left) shows pdfs (estimated from histograms of $10,000$ data samples) generated by our GEN (red), GAN (cyan) and the true distribution (blue). As it can clearly be seen, data generated from our GEN distribute around *all* three mixture components, demonstrating its ability to deal with multi-modal data in this case; whereas as expected, data generated from GAN concentrate at a *single* mode, reflecting the known mode collapsing problem in GAN. In addition, we empirically observe that the pdf generated by GAN is not stable as it tends to jump and fluctuate around the x-axis during training, whereas ours is much more stable.

Next, we further compare the quality of our generated data with that of GAN quantitatively. Since we know the true distribution P_{data} in this case, we employ two measures, namely symmetric KL and Wasserstein distances. These measures compute the distance between the normalized histograms generated from our GEN and GAN to the true P_{data}. Figure 4 (middle) again clearly demonstrates the superiority of our approach over GAN w.r.t. both distances; with Wasserstein metric, the distance from ours to the true distribution almost reduces to zero. This figure also demonstrates the stability of our GEN (red curves) during training as it is much less fluctuated compared with GAN (blue curves). Finally, Figure 4 (right) displays the mapping from the noise space to the input data space learned by GEN (red) and GAN (cyan). This again confirms that GAN is vulnerable from the mode

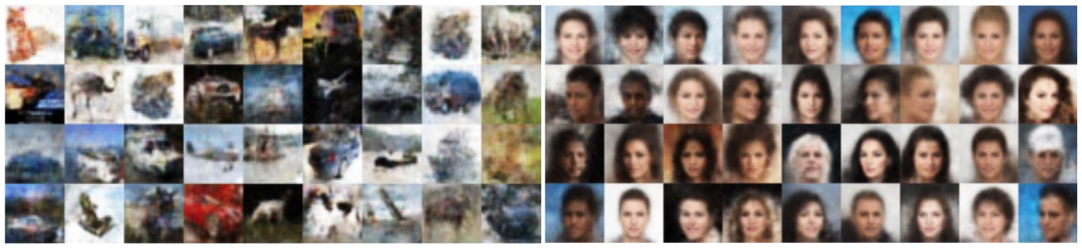


Figure 3: Data generated from our model when trained on CIFAR-10 dataset and CelebA dataset.

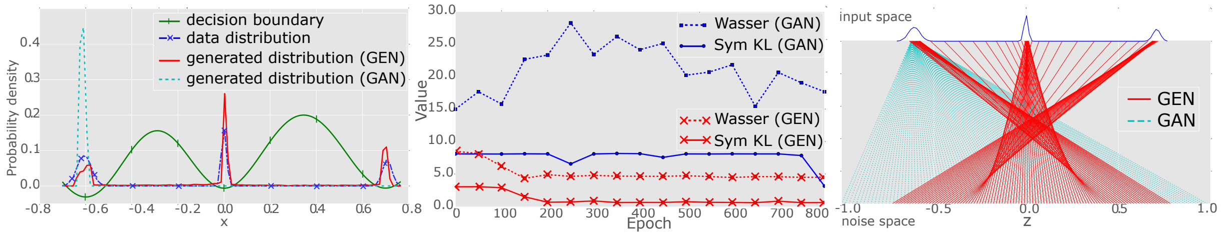


Figure 4: The comparison of GEN and GAN on the 1D synthetic dataset.

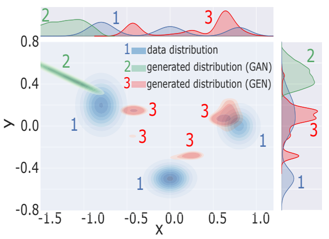
collapse whilst this is not the case for our method.

To further test our model beyond univariate case, we repeat a similar setting on 2D case with a mixture of Gaussian whose means centered at $[-0.8, 0.2]$, $[0.8, 0.0]$ and $[0.0, -0.5]$ respectively. Once again Figure 5a confirms that our model can generate data at multiple locations (pink) whereas GAN is stuck at one mode (green). To illustrate the idea of modeling data manifold, we synthesize a data manifold having the S-shape as shown in Figure 5b. The blue points represent the true data samples. Using these samples, we train GEN and then generate data shown as red points. It can be seen that the generated samples spread out across the manifold as expected. In addition, the GEN discriminator can recognize the region of the S-shape (i.e., the region covered by the green decision boundary).

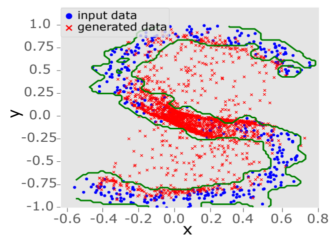
4.2 Experiment with Real Datasets

In this last experiment, we use four real-world image datasets: handwritten digits (MNIST), object images (CIFAR-10) and two human face datasets (Frey Face and CelebA). The popular MNIST dataset [Lecun *et al.*, 1998] contains $60,000$ images of digits from 0 to 9. We trained our GEN with three subsets including $1,000$, $5,000$, and $60,000$ images. The noise space for $\boldsymbol{z}$ has 10 dimensions; our generator $G(\boldsymbol{z})$ has the architecture of $1,000 \rightarrow 1,000 \rightarrow 1,000 \rightarrow 1,000$ (softplus units) and 784 sigmoid output units; and $D = 5,000$ random features was used to construct $\tilde{\Phi}$. As can be visually inspected from Figure 2 (left), the digits are generated with good quality and improved with more training data. We note that even with just $1,000$ training images, the quality is already sufficiently good. This observation concurs with the synthetic experiments demonstrated in Figure 5b where the geometric nature of our GEN could usually generalize data manifold well with moderate training data size.

The Frey Face dataset [Roweis and Saul, 2000] contains approximately 2000 images of Brendan's face, taken from sequential frames of a small video. We used a smaller network for our generator where $\boldsymbol{z}$ has 5 dimensions with a single layer of 200 softplus hidden units and 560 sigmoid outputs. Figure 2 (right) shows faces generated from our model where we can observe a good visual quality.



(a) Comparison of generated data from our GEN and GAN in 2D case. True contours are blue. Ours can again capture multiple modes (pink), whereas GAN failed to do so (green).



(b) Data manifold captured by our model. Blue points are training data, green curve is the boundary learned by our model via the discriminator and red points are data generated by our model.

Figure 5: Results on 2D synthetic data.

Our experiments were further extended to generating color images of real-life objects (CIFAR-10 [Krizhevsky, 2009]) and human faces (CelebA [Liu *et al.*, 2015]). After resizing original images into the size of $32 \times 32 \times 3$, we used a convolutional generator network with $512 \rightarrow 256 \rightarrow 128 \rightarrow 1,020$ (rectified linear units) and sigmoid output units and trained a leaky rectified linear discriminator network with 3 layers $32 \rightarrow 64 \rightarrow 128$. The random feature number and latent dimension are set to $5,000$ and 10 respectively. We show the images generated from our GEN network in Figure 3. Although the generated images are blurry and have low contrast, most of them contain meaningful objects and faces in various shapes, colors and poses. These results confirm the potential power of our proposed method as a generative network.

5 Conclusion

This paper has presented a new approach, called Geometric Enclosing Networks (GEN), to construct data generator via geometric view. Instead of examining the effectiveness of generator through the density of generated data with the true (unknown) distribution, our approach directly captures the nonlinear data manifolds in the input space via the principle of minimal enclosing ball. As the result, our model enjoys a nice geometric interpretation and possesses some key advantages, namely simple and easy-to-control optimization formulation, avoidance of mode collapse and efficiently learning data manifold representation. We have established experiments to demonstrate the behaviors of our proposed approach using synthetic and real-world datasets. The experimental results show that our GEN can avoid mode collapsing problem and result in better data generation quality via comparison with the true data (when using synthesis data) and visual inspection on MNIST, Frey Face, CIFAR-10, and CelebA datasets.

Acknowledgments

This work was partially supported by the Australian Research Council (ARC) DP160109394.

References

[Arjovsky *et al.*, 2017] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[Ben-Hur *et al.*, 2001] Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of Machine Learning Research (JMLR)*, 2:125–137, 2001.

[Bochner, 1959] Salomon Bochner. *Lectures on Fourier Integrals*, volume 42. Princeton University Press, 1959.

[Chen *et al.*, 2016] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2172–2180. Curran Associates, Inc., 2016.

[Cortes and Vapnik, 1995] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[Dziugaite *et al.*, 2015] Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680. 2014.

[Goodfellow, 2017] Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, 2017.

[Hoang *et al.*, 2018] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. MGAN: Training generative adversarial nets with multiple generators. In *International Conference on Learning Representations*, 2018.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kingma and Welling, 2014] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)*, 2014.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[Lecun *et al.*, 1998] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition, 1998.

[Li *et al.*, 2015] Yujia Li, Kevin Swersky, and Richard Zemel. Generative moment matching networks. In *International Conference on Machine Learning (ICML)*, pages 1718–1727, 2015.

[Liu *et al.*, 2015] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[Metz *et al.*, 2016] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[Nguyen *et al.*, 2017] Tu Dinh Nguyen, Trung Le, Hung Vu, and Dinh Phung. Dual discriminator generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2667–2677, 2017.

[Nowozin *et al.*, 2016] Sebastian Nowozin, Boston Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 271–279, 2016.

[Rahimi and Recht, 2007] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1184, 2007.

[Roweis and Saul, 2000] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[Salimans *et al.*, 2016] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2226–2234, 2016.

[Vapnik, 1995] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., 1995.