Generative Adversarial Networks

Master in Data Science
Barcelona Graduate School of Economics

Daniel Velasquez Vergara

June 30, 2017

Abstract

Generative adversarial networks or GANs are implicit generative models that define a neural network which takes a random vector as input and returns a sample from a probability distribution. In this setting, training is performed through a discriminative neural network that estimates the probability that a sample is observed rather than generated. The goal of this master thesis is to understand the training process of GANs. To do so, we analyze different objective functions and illustrate their advantages and disadvantages by performing simple experiments. We also explore how adversarial networks can be used to approximate Bayesian inference in probabilistic models and illustrate a connection between GANs and Variational Autoencoders.

Contents

1 Introduction

Given a set of observations from a data generating process with unknown distribution p, a generative model learns another probability distribution q_θ that represents p. There are cases in which q_θ has an explicit parametric form and it is possible to specify a log-likelihood function $\log q_\theta(\mathbf{x})$ that depends on a set of parameters θ. In an *implicit generative model*, the parametric representation of $q_\theta(\mathbf{x})$ is unknown, however it is possible to generate samples from it.

Generative adversarial networks or GANs are a type of algorithms for unsupervised learning introduced by Goodfellow et al. (2014). GANs are implicit generative models that define a scheme to produce samples from a target distribution without a explicit likelihood function. To do so, GANs specify a feed-forward neural network in which the input is a vector of random numbers passed through the layers of the network. Its output is a sample from the desired distribution. Training is performed through an auxiliary discriminative neural network that estimates the probability that a sample comes from the same distribution as the training set and has not been observed.

GANs have gained a lot of attention due to their capacity to produce data from high dimensional complex probability distributions. Practical applications include images, video and speech generation. However, solving this problem is not necessarily an easy task. Despite the theory that justify the procedure, in practice, finding an optimal solution is not guaranteed. Additionally, the objective function to solve the problem can be posed in different ways, and this represents a challenge given the implications on the data generated by the algorithm. A specific objective function definition can lead to low quality samples.

Section 2 provides a general overview of generative models, and in particular, implicit generative models. Section 3 discusses GANs in detail, the training procedure and the advantages and disadvantages of different specifications of the objective function. Finally, Section 4 discusses how GANs can improve Variational Autoencoders (VAEs) and illustrates an application of GANs to estimate the posterior density of latent variables within a Bayesian framework.

2 Generative Models

A generative model allows to produce random data from a certain probability distribution given a set of parameters. The task is to device a mechanism to generate samples that resemble the training dataset. If the density function q_θ has an explicit parametric representation, we can estimate the parameters θ by performing maximum likelihood. However, tractability might come at the cost of the model not being able to capture the complexities of the data. One class of models with known explicit density function are neural autoregressive generative models where joint probabilities correspond neural networks as products of conditional distributions, e.g. Wavenet and PixelRNN, generative models for audio and images respectively (see Van den Oord et al. (2016a, 2016b)). In this case,

$$q_\theta(\mathbf{x}) = \prod_{i=1}^{n} q_\theta(x_i|x_1, ..., x_{i-1})$$

Some of the practical difficulties of this type of models is the computational cost of generating one data point and the impossibility to paralellize the process.

On the other hand, there exist a class of models for which the density function of the data generating process is intractable and needs to be approximated. One prominent example are Variational Autoencoders (Kingma and Welling (2014)), where a learning algorithm maximizes a tractable lower bound for the log-likelihood of the data generating process (see Section 4.1). Another example are stochastic recurrent neural network (Boltzmann machines) in which a learning algorithm generates samples from a Markov chain to train the model.

2.1 Implicit Generative Models

Suppose the goal is to train a generative model, i.e. we want to learn $p(\mathbf{x})$ with support in some space $\mathcal{X} \subseteq \mathbb{R}^m$. $\mathbf{x}$ can be a high dimensional random variable with complex dependencies across dimensions that are captured by a set of latent variables. Let $\mathbf{z}$ be a vector of latent variables in some space $\mathcal{Z} \subseteq \mathbb{R}^d$, with density function $q(\mathbf{z})$. On the other hand, there is a family of deterministic functions $G_\theta(\mathbf{z})$, parameterized by $\theta \in \Theta$, such that $G : \mathcal{X} \times \Theta \rightarrow \mathcal{X}$. Assuming we can sample from $q(\mathbf{z})$, we optimize the parameters θ such that $G_\theta(\mathbf{z})$ is able to generate data from another density q_θ that resembles the observed dataset. Therefore, as illustrated by Mohamed and Lakshminarayanan (2017), for $\mathbf{z}' \sim q(\mathbf{z})$, we have that

$$\mathbf{x} = G(\mathbf{z}')$$

and

$$q_\theta(\mathbf{x}) = \frac{\partial}{\partial x_1} ... \frac{\partial}{\partial x_d} \int_{\{G(\mathbf{z}) \leq \mathbf{x}\}} q(\mathbf{z}) d\mathbf{z}, \tag{1}$$

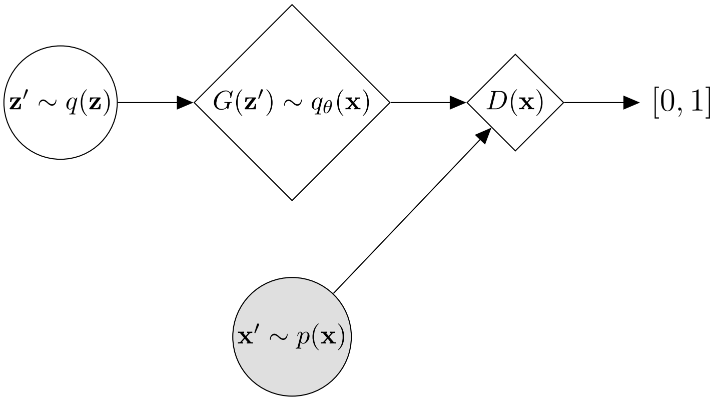
If G is invertible and $m = d$ we can recover the density of the generator. In general, G can be non-linear and $d > m$ when using deep networks. The integral in (1) is intractable, therefore the log-likelihood function of the model is not available. Learning in this context relies on the ability of generating samples from the model and discriminating it from real data.

Some generative models such as GANs and VAEs are supported on a neural networks that can be trained by stochastic gradient descent through back-propagation. However, when dealing with an implicit model, it is not entirely clear how to define an appropriate loss function to learn. Several problems arise in practice, e.g. lack of convergence or fitting a generator that only produces samples from a few modes of a target multi-modal distribution.

3 Generative Adversarial Networks (GANs)

Generative adversarial networks are a game between two players. The generator and the discriminator. The goal of the generator is to create samples that seem to come from the same distribution as the training data. The discriminator is a classifier that examines samples to determine whether they are real or fake. The discriminator is a typical supervised learning problem in which the samples have the label *real* or *fake*.

We start with a set of observations from some unknown density p. Let $\mathbf{z}$ be a latent variable with density q. The generator G is a differentiable function of $\mathbf{z}$ with parameters θ. Initially, $\mathbf{z}'$ is sampled from q, and $G(\mathbf{z}')$ generates $\mathbf{x}'$ from some density q_θ. The discriminator D is a function that takes a sample, either generated or observed, and assigns a probability that it comes from p.


$\mathbf{z}' \sim q(\mathbf{z})$
$G(\mathbf{z}') \sim q_\theta(\mathbf{x})$
$D(\mathbf{x})$
$[0, 1]$
$\mathbf{x}' \sim p(\mathbf{x})$

Assuming $\mathbf{z}$ is a normally distributed random vector in $\mathbb{R}^d$, it is possible to generate data from any distribution as long as G is sufficiently complex. In order for the support of q_θ to correspond to the full space of $\mathbf{x}$, the dimension of $\mathbf{z}$ needs to be greater or equal than the dimension of $\mathbf{x}$. In this context, we can draw samples from the model that are indistinguishable to the samples from the true data generating process.

GANs have attracted a lot of attention due to their capacity to produce data from complex high dimensional distributions. Additionally, samples can be generated in parallel, which can improve the speed of the algorithm. Unlike other methods such as VAEs or Boltzman machines they do not depend an a variational bound or a Markov chain process.

3.1 GANs Training

The objective is to train a generative model, i.e. we want to learn to sample from some density $q_\theta(\mathbf{x})$, such that the generated data resembles the actual observations that come from some unknown density $p(\mathbf{x})$ with support in $\mathcal{X}$. Let $\mathbf{z}$ be a vector of latent variables with density function $q(\mathbf{z})$ with support in some space $\mathcal{Z}$. The generator is a deterministic function $G(\mathbf{z})$, parameterized by $\theta \in \Theta$, such that $G : \mathcal{Z} \times \Theta \to \mathcal{X}$. More specifically, G is a (deep) neural network. On the other hand, the discriminator is a classifier $D(\mathbf{x})$ parameterized by

$\phi \in \Phi$, such that $D : \mathcal{X} \times \Phi \rightarrow [0, 1]$. The function D examines samples to determine whether they belong to the training dataset or have been generated by G. D is also a (deep) neural network.

Training is typically performed using Stochastic Gradient Descent. In particular, at each step we have two minibatches: one of $\mathbf{x}$ values from the training dataset, and a minibatch of $\mathbf{z}$ values drawn from the model's prior over latent variables. We update ϕ to reduce the loss function of the discriminator network $\mathcal{L}^D$, and then we update θ to reduce the cost of the generator network $\mathcal{L}^G$.

3.2 Minimax Approach

Goodfellow et al. (2014) propose a zero-sum game between the discriminator D and the generator G competing against each other. In order to derive the loss function, consider a binary random variable y. We have that $y = 1$ if $\mathbf{x}$ comes from p and $y = 0$ if $\mathbf{x}$ comes from q_θ. Let $p(\mathbf{x}) = p(\mathbf{x}|y = 1)$ and $q_\theta(\mathbf{x}) = p(\mathbf{x}|y = 0)$. Let $\pi = p(y = 1)$. The discriminator is a neural network with sigmoid output that estimates the probability of $\mathbf{x}$ being a real observation, i.e. $D(\mathbf{x}) = p(y = 1|\mathbf{x})$.Given a minibatch of $\mathbf{x}$ labeled 1 and another minibach labeled cero, the cross entropy loss is given by

$$\begin{aligned}\mathcal{L}(\phi, \theta) &= \mathbb{E}_{p(\mathbf{x}|y)p(y)}[-y \log D(\mathbf{x}) - (1 - y) \log (1 - D(\mathbf{x}))] \\ &= \pi \mathbb{E}_p[-\log D(\mathbf{x})] + (1 - \pi)\mathbb{E}_{q_\theta}[-\log (1 - D(\mathbf{x}))] \\ &= -\pi \mathbb{E}_p \log D(\mathbf{x}) - (1 - \pi)\mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z}))) \qquad (2)\end{aligned}$$

This loss function is optimized by two stages at each iteration. Assuming $\pi = 1/2$, first the discriminator is optimized by minimizing $\mathcal{L}(\phi, \theta)$ with respect to ϕ,

$$\mathcal{L}^D(\phi, \theta) = -\frac{1}{2}\mathbb{E}_p \log D(\mathbf{x}) - \frac{1}{2}\mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z}))) \qquad (3)$$

In another stage, the function is minimized with respect to θ. The terms that depend on θ correspond to the generator loss, i.e.

$$\mathcal{L}^G(\phi, \theta) = \frac{1}{2}\mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z}))) \qquad (4)$$

Consider the loss not as a function of ϕ and θ but as a functional of D and G. Goodfellow et al. (2014) show that for G fixed, the optimal discriminator D is given by

$$D^*(\mathbf{x}) = \frac{p(\mathbf{x})}{p(\mathbf{x}) + q_\theta(\mathbf{x})} \tag{5}$$

Replacing (5) in (2), define

$$\begin{aligned}
\mathcal{L}(G) &= \min_D \mathcal{L}(D, G) \\
&= -\frac{1}{2}\mathbb{E}_p \log D^*(\mathbf{x}) - \frac{1}{2}\mathbb{E}_{\mathbf{z}} \log\left(1 - D^*(G(\mathbf{z}))\right) \\
&= -\frac{1}{2}\mathbb{E}_p \left[\log \frac{p(\mathbf{x})}{p(\mathbf{x}) + q_\theta(\mathbf{x})}\right] - \frac{1}{2}\mathbb{E}_{q_\theta} \left[\log \frac{q_\theta(\mathbf{x})}{p(\mathbf{x}) + q_\theta(\mathbf{x})}\right]
\end{aligned} \tag{6}$$

Following Goodfellow et al. (2014), at this point $\mathcal{L}(G)$ is minimized when $p = q_\theta$, in which case $D^*(\mathbf{x}) = \frac{1}{2}$. The global optimum corresponds to $\frac{\log(4)}{2}$. Let the Kullback-Leibler divergence D_{KL} between two probability distributions p and q be given by

$$D_{KL}(p||q) = \mathbb{E}_p \left[\log \frac{p(\mathbf{x})}{q(\mathbf{x})}\right],$$

then, we can reformulate (6) as

$$\begin{aligned}
\mathcal{L}(G) &= \frac{\log(4)}{2} - \frac{1}{2}D_{KL}\left(p||\frac{p + q_\theta}{2}\right) - \frac{1}{2}D_{KL}\left(q_\theta||\frac{p + q_\theta}{2}\right) \\
&= \frac{\log(4)}{2} - D_{JSD}(p||q_\theta)
\end{aligned} \tag{7}$$

D_{JSD} is the Jensen-Shannon divergence and measures the similarity between two probability distributions. Unlike the D_{KL}, the D_{JSD} is symmetric and finite. In principle, as pointed by Nowosin et al. (2016), if there are sufficient training samples and q_θ is rich enough, p can be properly approximated. In practice, the best behavior is obtained by using simultaneous gradient descent, with one iteration for each player.

Algorithm 1 presents the basic structure of the GANs procedure. Given a loss function of the discriminator D and the generator G, the parameters ϕ and θ, of D and G respectively, are updated for a certain number of iterations using stochastic gradient descent. At each iteration, we need a sample $\mathbf{x}$ and $\mathbf{z}$ in order to estimate the losses $\mathcal{L}^D$ and $\mathcal{L}^G$.

```
Data: $\mathcal{L}^D$, $\mathcal{L}^G$, number of epochs $n\_epochs$, and minibatch size $M$.
Result: Trained parameters of $D$ and $G$.
for $i \in 1 : n\_epochs$ do
    Generate sample of size $M$, $\mathbf{x}' \sim p(\mathbf{x})$;
    Generate sample of size $M$, $\mathbf{z}' \sim q(\mathbf{z})$;
    $\min_\phi \mathcal{L}^D$;
    $\min_\theta \mathcal{L}^G$
end
```

Algorithm 1: GANs basic algorithm.

3.2.1 Bivariate Normal Distribution

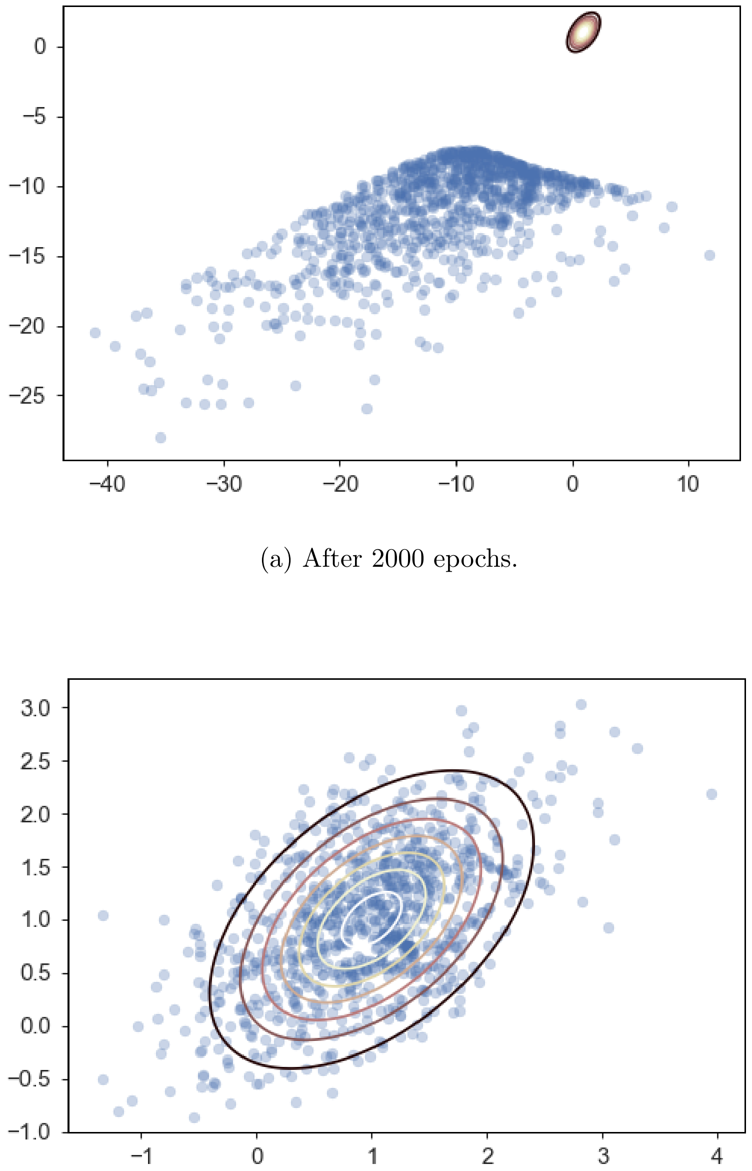
Suppose we want to train a neural network to sample from a bivariate normal distribution. In this example the target density p is $N(\mu, \Sigma)$, where

$$\mu = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

We assume the standard deviation of each marginal density is 0.7 and the correlation is 0.5, therefore

$$\Sigma = \begin{pmatrix} 0.49 & 0.245 \\ 0.245 & 0.49 \end{pmatrix}$$

The generator network has 1 hidden layer with 20 units and *softplus* activation function. The discriminator network has 2 hidden layers. Each layer has 40 units with *tanh* activation functions and *sigmoid* output. Figure 1 exhibits the contours plot and 1000 points produced by the generator after 2000 and 4000 epochs. We observe how the generated points are initially far away from the target, but eventually G learns to produce data consistent with p.



(a) After 2000 epochs.

(b) After 4000 epochs.

Figure 1: Contours plot and 1000 points produced by the generator. In figures (a) and (b) the generator parameters have been trained for 2000 and 4000 epochs respectively.

3.2.2 Practical considerations

The problem, as posed by Goodfellow et al. (2014) can be stated as minimizing the Jensen Shannon divergence between the data and the model distribution. However, the game converges to its equilibrium if the optimization is performed in a functional space. In practice, we do not know the distributions of the data generating process or the model. The discriminator and the generator are deep neural networks which are optimized with respect to their parameters. This is a non-convex optimization problem and convergence is not always possible. Additionally, the specification of the generator G imposes limitations on the random data that can be produced.

GANs do not naturally come with a likelihood function to be optimized and, because of the adversarial structure of the algorithm, it can be difficult to evaluate if the model is training in an appropriate way. Several problems arise in practice, e.g. the quality of the samples might be lower than expected. Suppose the real data comes from a multi-modal distribution. It is common for GANs to generate samples from very few modes, even if the generator is a deep neural network with significant capacity. Another problem that appears in practical implementations is related to the vanishing gradient of the objective function. Different authors have proposed alternative specifications of the generator's loss function in order the improve the odds of achieving convergence.

Salimans et al. (2016) posit a variety of techniques to train GANs and obtain better results. Some of their ideas are based on experiments and empirical results and do not have a theoretical support. They reformulate the generator loss function as

$$\mathcal{L}^G(\theta) = -\mathbb{E}_{\mathbf{z}} \log D(G(\mathbf{z})) \tag{8}$$

This last objective function allows the algorithm to converge faster when it does converge. Figure 2 presents each of the two specifications of the generator's loss, shown in equations (4) and (8), as a function of the discriminator's output. Equation (8) provides a stronger gradient at the start of the algorithm when the discriminator is expected to assign a low probability to a generated sample.

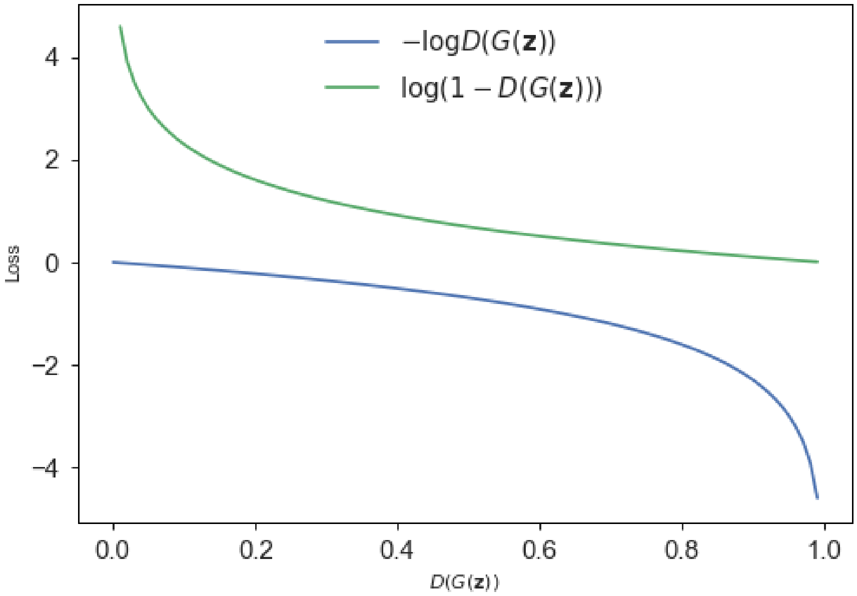


Figure 2: Generator's loss function.

However, the connection with the Jensen-Shanon divergence does not exist anymore. Arjovsky et al. (2017) attempt to tackle the vanishing-gradient problem by incorporating an approximation of the *earth mover* distance to measure the closeness between the densities q_θ and p. Nowozin et al. (2016) show that generative models can be trained using any f-divergence. Based on their approach, in the next section we evaluate particular cases such as Kullback-Leibler divergence.

3.3 Divergence minimization

Generative Adversarial Networks define a mechanism to generate samples from some density q_θ that is as similar as possible to the training dataset distribution. Therefore, it is natural to think about training the generator to minimize a divergence measure between q_θ and the true density p. As explained before, in the original formulation Goodfellow et al. (2014) attempt to minimize the Jensen-Shanon divergence between the q_θ and p. Another reasonable objective would be to minimize the Kullback-Leibler divergence between the unknown data generating process and the generator's distribution q_θ. Assuming both q_θ and p are continuous and have the same support, the (Reverse) Kullback-Leibler divergence is given by

$$D_{KL}(q_\theta || p) = \mathbb{E}_{q_\theta}\left[\log \frac{q_\theta(\mathbf{x})}{p(\mathbf{x})}\right] \tag{9}$$

The optimal discriminator is denoted D^*. From equation (5) we have

$$\begin{aligned} D^*(\mathbf{x}) &= \frac{p(\mathbf{x})}{p(\mathbf{x}) + q_\theta(\mathbf{x})} \\ \frac{1 - D^*(\mathbf{x})}{D^*(\mathbf{x})} &= \frac{q_\theta(\mathbf{x})}{p(\mathbf{x})} \end{aligned} \tag{10}$$

Assuming the discriminator is closer to optimality, i.e. $D \approx D^*$, and replacing (10) in (9) we obtain

$$\begin{aligned} D_{KL}(q_\theta || p) &= \mathbb{E}_{q_\theta}\left[\log \frac{1 - D^*(\mathbf{x})}{D^*(\mathbf{x})}\right] \\ &\approx \mathbb{E}_{q_\theta}\left[\log \frac{1 - D(\mathbf{x})}{D(\mathbf{x})}\right] \end{aligned}$$

The generator loss can be formulated as

$$\begin{aligned} \mathcal{L}^G &= \mathbb{E}_{q_\theta}\left[\log \frac{1 - D(\mathbf{x})}{D(\mathbf{x})}\right] \\ &= \mathbb{E}_z\left[\log \frac{1 - D(G(\mathbf{z}))}{D(G(\mathbf{z}))}\right] \end{aligned} \tag{11}$$

It is important to note that the KL divergence is not symmetric and we cannot expect the same result when optimizing $D_{KL}(q_\theta || p)$ rather than $D_{KL}(p || q_\theta)$. In Section 3.3.2 we explore in more detail the potential effects of incorporating in the algorithm different divergences.

3.3.1 f-Divergence

Nowozin et at. (2016) propose the class of f-divergences to train generative models. Let $f : \mathbb{R}_+ \to \mathbb{R}$ be a convex, semi-continuous function with $f(1) = 0$. The f-divergence between the generator density q_θ and the true density p is

$$D_f = \int q_\theta(\mathbf{x}) f\left(\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}\right) d\mathbf{x} \tag{12}$$

Let f^* be the convex conjugate of f,

$$f^*(t) = \sup_{u \in \mathrm{dom}_f} [ut - f(u)] \tag{13}$$

Therefore

$$f(u) = \sup_{t \in \mathrm{dom}_{f^*}} [tu - f^*(t)]. \tag{14}$$

Substituting (14) in (12), we obtain

$$D_f = \int q_\theta(\mathbf{x}) \sup_{t \in \mathrm{dom}_{f^*}} \left\{ t \frac{p(\mathbf{x})}{q_\theta(\mathbf{x})} - f^*(t) \right\} d\mathbf{x}$$

Nowozin et al. (2016) restrict the supremum over a class of functions $T \in \mathcal{T} : \mathcal{X} \to \mathbb{R}$. Then interchanging the integral and the supremum we have that

$$\begin{aligned} D_f &\geq \sup_{T \in \mathcal{T}} \left(\int_{\mathcal{X}} p(\mathbf{x}) T(\mathbf{x}) d\mathbf{x} - \int_{\mathcal{X}} q_\theta(\mathbf{x}) f^*(T(\mathbf{x})) d\mathbf{x} \right) \\ &= \sup_{T \in \mathcal{T}} \left(\mathbb{E}_p[T(\mathbf{x})] - \mathbb{E}_{q_\theta}[f^*(T(\mathbf{x}))] \right) \end{aligned} \tag{15}$$

By taking the derivative of (15) with respect to T, the bound is tight for

$$T^* = f' \left(\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})} \right) \tag{16}$$

therefore, the supremum in (15) becomes over the density ratio $\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}$. Finally, replacing (16) in (15) we obtain the objective function which corresponds to the lower bound of the divergence, i.e.

$$\mathcal{L} = \mathbb{E}_p \left[f' \left(\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})} \right) \right] - \mathbb{E}_{q_\theta} \left[f^* \left(f' \left(\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})} \right) \right) \right] \tag{17}$$

This loss function allows to train the model using the adversarial scheme with two minibatches: one coming from the dataset and another coming from the generator function.

3.3.1.1 Kullback-Leibler divergence, $D(p||q_\theta)$

Let $f(u) = u \log(u)$. Then, $f'(u) = 1 + \log(u)$. The convex conjugate corresponds to

$$f^*(t) = \sup_{u \in \mathrm{dom}_f} [ut - u\log(u)] \tag{18}$$

Deriving (18) with respect to u and equalizing to cero we obtain

$$\begin{aligned} t - 1 - \log(u) &= 0 \\ u &= e^{t-1} \end{aligned}$$

therefore,

$$\begin{aligned} f^*(t) &= te^{t-1} - (t-1)e^{1-t} \\ &= e^{t-1} \end{aligned}$$

with domain in $\mathbb{R}$. We have that

$$f'\left(\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}\right) = 1 + \log\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}$$

then, in this case, equation (17) corresponds to

$$\mathcal{L} = \mathbb{E}_p\left[1 + \log\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}\right] + \mathbb{E}_{q_\theta}\left[e^{\log\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}}\right] \tag{19}$$

Let $\sigma^{-1}(x)$ be the inverse of the sigmoid function, i.e.

$$\sigma^{-1}(x) = -\log\left(\frac{1}{x} - 1\right)$$

We have that

$$
\begin{aligned}
\sigma^{-1}(D(\mathbf{x})) &= -\log\left(\frac{1}{D(\mathbf{x})} - 1\right) \\
&\approx -\log\left(\frac{p(\mathbf{x}) + q_\theta(\mathbf{x})}{p(\mathbf{x})} - 1\right) \\
&= -\log\left(\frac{q_\theta(\mathbf{x})}{p(\mathbf{x})}\right) \\
&= \log\left(\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}\right),
\end{aligned}
$$

therefore, from equation (19), we conclude that the loss function associated with the generator is given by

$$
\begin{aligned}
\mathcal{L}^G &= -\mathbb{E}_{q_\theta}\left[\exp(\sigma^{-1}(D(\mathbf{x}))\right] \\
&= -\mathbb{E}_z\left[\exp(\sigma^{-1}(D(G(\mathbf{z})))\right]
\end{aligned}
$$

3.3.1.2 Reverse Kullback-Leibler divergence, $D(q_\theta||p)$

Let $f(u) = -log(u)$. Then, $f'(u) = -\frac{1}{u}$. The convex conjugate corresponds to

$$
f^*(t) = \sup_{u \in \mathrm{dom}_f} [ut + \log(u)] \tag{20}
$$

Deriving with respect to u and equalizing to cero we obtain

$$
\begin{aligned}
t + \frac{1}{u} &= 0 \\
u &= -\frac{1}{t}
\end{aligned}
$$

therefore,

$$
f^*(t) = -1 - \log(-t) \tag{21}
$$

with domain in $\mathbb{R}_-$. We have that

$$f'\left(\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}\right) = -\frac{q_\theta(\mathbf{x})}{p(\mathbf{x})} \tag{22}$$

then,

$$\mathcal{L} = \mathbb{E}_p\left[-\frac{q_\theta(\mathbf{x})}{p(\mathbf{x})}\right] + \mathbb{E}_{q_\theta}\left[-1 - \log\left(\frac{q_\theta(\mathbf{x})}{p(\mathbf{x})}\right)\right]$$

In this case, the loss function associated with the generator is the same as in equation (11).

3.3.2 Divergence effects on the generator network output

Consider the $D_{KL}(p||q_\theta)$. Suppose there is point x for which $p(\mathbf{x}) < q_\theta(\mathbf{x})$, i.e. it is more likely that $\mathbf{x}$ comes from the model rather than generated. As $q_\theta(\mathbf{x})$ goes to cero the divergence goes to infinity at a high speed. On the other hand, if $p(\mathbf{x}) < q_\theta(\mathbf{x})$ and $q_\theta(\mathbf{x}) > 0$, the cost as $p(\mathbf{x})$ goes to cero is very low. In this case, the generator will try to cover the whole domain of $\mathbf{x}$ and will not put enough mass to the modes of the true density p. Arjovsky and Bottou (2017) call this phenomenon *mode dropping.*

Now consider $D_{KL}(q_\theta||p)$. A point x for which $p(\mathbf{x}) < q_\theta(\mathbf{x})$, the divergence goes to infinity as $p(\mathbf{x})$ goes to cero. In this case the generator will try to produce samples for which $p(\mathbf{x})$ is significant leading to what is usually called as *mode collapse.*

3.3.2.1 Mixture of Gaussians

Suppose we want to approximate the density function of a mixture of two normal random variables with a univariate normal random variable. Assume the latent variable $z \sim N(0,1)$ and the generator is trained to sample from a normal distribution $N(\mu, \sigma^2)$. In this case, $G(z) = \mu + \sigma z$. Figure 3 shows the density function of the mixture of two Gaussians: $N(1, 0.3^2)$ and $N(3, 0.3^2)$. The weight of each component is 0.5. Additionally, Figure 3 exhibits the density function of the latent variable z, and the density function of a normal random variable with mean μ and variance σ^2, where μ and σ have been randomly initialized.

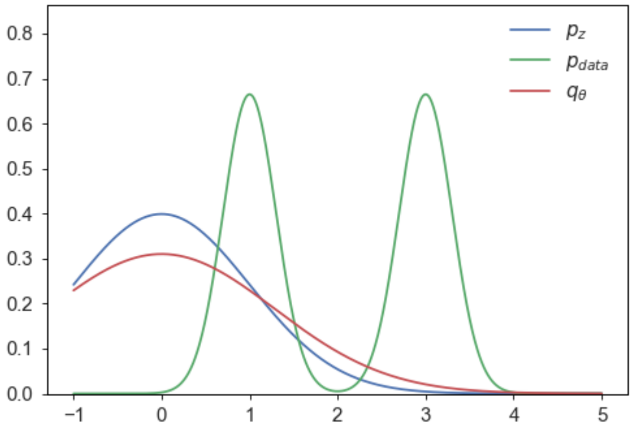


Figure 3: Plots of the density function of the mixture of Gaussians, the density function of the latent variable z, and the density function of a normal random variable with mean μ and variance σ^2, where μ and σ have been randomly initialized.

In this example the discriminator network has 2 hidden layers. Each layer has 40 units with *tanh* activation functions, and *sigmoid* output. First, we train G to learn μ and σ by minimizing $D_{KL}(p||q_\theta)$, i.e. using equation (20). As expected, in this case the resulting mean parameter approximates the average of the means of the two Gaussian densities as shown in Figure 4.

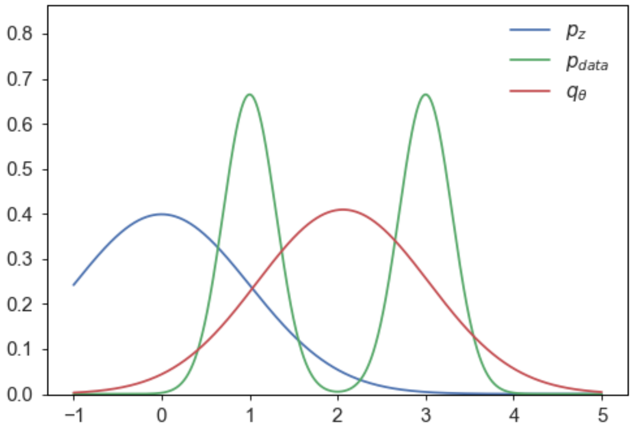


Figure 4: Plots Density function of the mixture of Gaussians, density function of the latent variable z, the density function of a normal random variable with mean μ and variance σ^2, after training the generator by minimizing $D_{KL}(p||q_\theta)$.

Then, we train G to learn μ and σ by minimizing $D_{KL}(q_\theta||p)$, i.e. using equation (11). Now the resulting mean parameter collapses to one of the modes of the target density as presented in Figure 5. Table 1 summarizes the results.

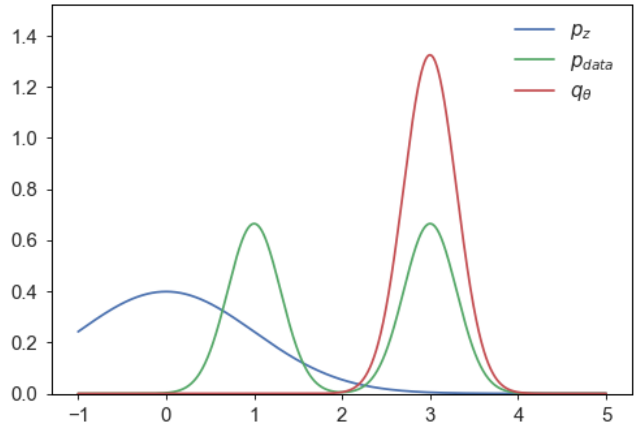


Figure 5: Plots Density function of the mixture of Gaussians, density function of the latent variable z, the density function of a normal random variable with mean μ and variance σ^2, after training the generator by minimizing $D_{KL}(q_\theta)||p$.

Divergence	Mean	Standard Deviation
$D_{KL}(p\|\|q_\theta)$	2.09541	0.95154
$D_{KL}(q_\theta\|\|p)$	2.99796	0.30128

Table 1: Mean and standard deviation obtained after training the generator by minimizing $D_{KL}(p||q_\theta)$ and $D_{KL}(q_\theta||p)$.

There exist different alternatives to pose the generator objective function for GANS training. Different specifications lead to varied results directly related to the characteristics of the samples produced by the algorithm.

4 GANs and Bayesian Inference

Within the GANs framework, we define $\mathbf{z}$ as the input variable variable and $\mathbf{x}$ as an observation. $\mathbf{z}$ is sampled from some distribution, and $G(\mathbf{z})$ generates a sample from an unknown density q_θ. From a probabilistic perspective, $\mathbf{z}$ is a latent variable with prior density $q(\mathbf{z})$. We can think of $\mathbf{x}$ as being sampled from the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$. The joint density is given by $p(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})q(\mathbf{z})$. The posterior density of $\mathbf{z}$ conditional on $\mathbf{x}$ is

$$p(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})q(\mathbf{z})}{q_\theta(\mathbf{x})}$$

Variational inference implies approximating the true posterior $p(\mathbf{z}|\mathbf{x})$ through a family of distributions $q_\gamma(\mathbf{z}|\mathbf{x})$ (denoted $q_\gamma(\mathbf{z})$) that depend on a certain set of parameters γ. The Kullback-Leibler divergence allows to measure the information loss when using $q_\gamma(\mathbf{z}|\mathbf{x})$ to approach the true posterior.

In general, likelihood-free inference methods estimate parameters for which the difference between simulated and observed data is small. However, it is not explicitly defined how the parameters affect the relation between the data generating process and the model to approximate it. First, we start with a review of Variational Autoencoders; a framework that comes not only with a generative model, but also with an inference model. Then, we illustrate how GANs can be used to approximate variational inference in the context of implicit models.

4.1 Variational Autoencoders

We have a set of observations from a unknown distribution p in some space $\mathcal{X}$. So far, we have assumed we can train a model that allows us to sample from some density q_θ. As mentioned before, assuming $\mathbf{x}$ is sample from $p_\theta(\mathbf{x}|\mathbf{z})$ we have that

$$q_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})q(\mathbf{z})d\mathbf{z} \tag{23}$$

In general, $q_\theta(\mathbf{x})$ does not have an analytic representation. It can be numerically approximated using Monte Carlo simulation, i.e.

$$q_\theta(\mathbf{x}) \approx \frac{1}{n}\sum_{i}^{n} p_\theta(\mathbf{x}|\mathbf{z}^{(i)}), \text{ where } \mathbf{z}^{(i)} \sim q(\mathbf{z}),$$

however, this expression might be computationally expensive, particularly in high dimensions.

Many values of $\mathbf{z}$ do not contribute to $q_\theta(\mathbf{x})$. Variational Autoencoders attempt to estimate a density function $q_\gamma(\mathbf{z}|\mathbf{x})$ that comprises the $\mathbf{z}$ values that are more likely to generate $\mathbf{x}$. Then

$$q_\theta(\mathbf{x}) \approx \int p_\theta(\mathbf{x}|\mathbf{z})q_\gamma(\mathbf{z})d\mathbf{z} \tag{24}$$

The main objective of the procedure is to maximize the log-likelihood $\mathbb{E}_p[\log q_\theta(\mathbf{x})]$. We have that

$$\begin{aligned}
\log q_\theta(\mathbf{x}) &= \log \int p(\mathbf{x},\mathbf{z})d\mathbf{z} \\
&= \log \int p(\mathbf{x},\mathbf{z})\frac{q_\gamma(\mathbf{z}|\mathbf{x})}{q_\gamma(\mathbf{z}|\mathbf{x})}d\mathbf{z} \\
&= \log\left(\mathbb{E}_{z\sim q_\gamma}\left[\frac{p(\mathbf{x},\mathbf{z})}{q_\gamma(\mathbf{z}|\mathbf{x})}\right]\right) \\
&\geq \mathbb{E}_{z\sim q_\gamma}[\log p(\mathbf{x},\mathbf{z}) - \log q_\gamma(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{z\sim q_\gamma}[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log q(\mathbf{z}) - \log q_\gamma(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{z\sim q_\gamma}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\gamma(\mathbf{z}|\mathbf{x})||q(\mathbf{z})) \qquad (25)
\end{aligned}$$

This last expression is known as the Evidence Lower Bound (ELB). Notice that the Kullback-Leibler divergence between the true posterior and the approximation is given by

$$\begin{aligned}
D_{KL}(q_\gamma(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{z\sim q_\gamma}\left[\log\frac{q_\gamma(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathbb{E}_{z\sim q_\gamma}[\log q_\gamma(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{z\sim q_\gamma}[\log q_\gamma(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x}|\mathbf{z}) - \log q(\mathbf{z})] + \log q_\theta(\mathbf{x}) \\
&= -(\mathbb{E}_{z\sim q_\gamma}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\gamma(\mathbf{z}|\mathbf{x})||q(\mathbf{z}))) + \log q_\theta(\mathbf{x}) \qquad (26)
\end{aligned}$$

It is not possible to minimize the KL divergence, however, this is equivalent to maximizing the ELB. From equation (26) we obtain

$$D_{KL}(q_\gamma(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) - \log q_\theta(\mathbf{x}) = -(\mathbb{E}_{z\sim q_\gamma}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\gamma(\mathbf{z}|\mathbf{x})||q(\mathbf{z}))) \tag{27}$$

The left-hand-side of equation (27) is the expression that we want to minimize. This is achieved by increasing the log-likelihood, and decreasing the divergence between the true posterior density of the latent variable and its approximate posterior $q_\gamma(\mathbf{z}|\mathbf{x})$. The divergence will go to cero as long as q_γ has enough capacity. The right-hand-side is the ELB which is something we can optimize using stochastic gradient descent. The divergence $D_{KL}(q_\gamma(\mathbf{z}|\mathbf{x})||q(\mathbf{z}))$ measures the information gain when $\mathbf{z}$ comes from $q_\gamma(\mathbf{z}|\mathbf{x})$ rather than $q(\mathbf{z})$. In this setting we perform maximum likelihood by

$$\min_{\theta} \min_{\gamma} \mathbb{E}_p \left[D_{KL}(q_\gamma(\mathbf{z}|\mathbf{x})||q(\mathbf{z})) - \mathbb{E}_{z\sim q_\gamma}[\log p_\theta(\mathbf{x}|\mathbf{z})] \right] \tag{28}$$

Variational Autoencoders typically assume $p_\theta(\mathbf{x}|\mathbf{z}) = N(\mathbf{x}|G(\mathbf{z}), \sigma^2\mathbf{I})$, where $\mathbf{I}$ is the identity matrix. In this setting, the main requirement is that $p_\theta(\mathbf{x}|\mathbf{z})$ is continuous in θ and tractable. The $\mathbf{z}$ values are typically drawn from a simple distribution, e.g. $N(0, \mathbf{I})$.

Kingma and Welling (2014) assume $q_\gamma(\mathbf{z}|\mathbf{x})$ is a Gaussian distribution with mean and variance represented by neural networks depending on $\mathbf{x}$, i.e. $q_\gamma(\mathbf{z}) \sim N(\mu_\gamma, \Sigma_\gamma)$. Then $D_{KL}(q_\gamma(\mathbf{z})||q(\mathbf{z}))$ corresponds to the Kullback-Leibler divergence between two multivariate normal distributions which has closed-form solution. This divergence expression takes a sample $\mathbf{x}$ as input and somehow works as an encoder. On the other hand, we approximate $\mathbb{E}_{z\sim q_\gamma}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ using one sample of $\mathbf{z}$.

In order to train the model we need to calculate the gradient of $D_{KL}(q_\gamma(\mathbf{z})||q(\mathbf{z})) - \log p_\theta(\mathbf{x}|\mathbf{z})$ and average over samples of $\mathbf{z}$ and $\mathbf{x}$. At some point the algorithm produces samples from $q_\gamma(\mathbf{z}) = N(\mu_\gamma, \Sigma_\gamma)$. In order to apply stochastic gradient descent, this is reparameterized such that $\mathbf{z} = \mu_\gamma + \Sigma_\gamma^{1/2}\epsilon$ where $\epsilon \sim N(0, \mathbf{I})$ and $\Sigma_\gamma^{1/2}$ is a matrix such that $\Sigma_\gamma = \Sigma_\gamma^{1/2}(\Sigma_\gamma^{1/2})^T$.

4.2 Variational Autoencoders Extension using GANs

The model described in the previous section has the advantage of being tractable. However, the assumption that $q_\gamma(\mathbf{z}|\mathbf{x})$ is a normal density imposes restrictions on $\mathbf{z}$ that potentially lead to less quality samples coming from the generator.

Theis et al. (2016) argue that VAEs distribute probability mass diffusely over the data space, therefore they fail to generate high quality samples. This has been particularly analyzed when used to generate images. Larsen et al. (2015) and Mescheder et al. (2017) assert that VAEs fail because the inference model is unable to capture the complexities of the true posterior distribution.

Mescheder et al. (2017) propose an extension of the VAE framework. Instead of assuming $q_\gamma(\mathbf{z}|\mathbf{x})$ has some parametric representation, they introduce an additional discriminator network to train Variational Autoencoders that sample from $q_\gamma(\mathbf{z}|\mathbf{x})$ by means of a generator neural network. From equation (28) we know that the objective is to minimize the negative ELB,

$$\mathbb{E}_p \left[\mathbb{E}_{z\sim q_\gamma} \left[\log \frac{q_\gamma(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} \right] - \mathbb{E}_{z\sim q_\gamma}[\log p_\theta(\mathbf{x}|\mathbf{z})] \right] \tag{29}$$

In this case, we train a discriminator D to receive a pair $(\mathbf{z}, \mathbf{x})$, where $\mathbf{x}$ is an observed data point and $\mathbf{z}$ comes either from its prior or the posterior density. There is a generator G that takes $\mathbf{x}$ and some noise ϵ and produces $\mathbf{z}$ from $q_\gamma(\mathbf{z}|\mathbf{x})$. The discriminator cost function is given by

$$\begin{aligned}\mathcal{L}^D &= -\mathbb{E}_p\left[\mathbb{E}_{z\sim q}[\log D(\mathbf{z},\mathbf{x})] + \mathbb{E}_{z\sim q_\gamma}[\log(1 - D(\mathbf{z},\mathbf{x}))]\right] \\ &= -\mathbb{E}_p\left[\mathbb{E}_{z\sim q}[\log D(\mathbf{z},\mathbf{x})] + \mathbb{E}_\epsilon[\log(1 - D(G(\epsilon,\mathbf{x}),\mathbf{x}))]\right]\end{aligned}$$

Following the same line of argument of Goodfellow et al. (2014) it is possible to show that the optimal discriminator is

$$D^*(\mathbf{z},\mathbf{x}) = \frac{q(\mathbf{z})}{q(\mathbf{z}) + q_\gamma(\mathbf{z}|\mathbf{x})}, \tag{30}$$

therefore,

$$\frac{D(\mathbf{z},\mathbf{x})}{1 - D(\mathbf{z},\mathbf{x})} = \frac{q_\gamma(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} \tag{31}$$

Assuming the discriminator is close to optimality, the generator loss function approximates the expression in equation (29), i.e

$$\begin{aligned}\mathcal{L}^G &= \mathbb{E}_p\left[\mathbb{E}_{z\sim q_\gamma}\left[\log\frac{D(\mathbf{z},\mathbf{x})}{1 - D(\mathbf{z},\mathbf{x})} - \log p_\theta(\mathbf{x}|\mathbf{z})\right]\right] \\ &= \mathbb{E}_p\left[\mathbb{E}_\epsilon\left[\log\frac{D(G(\epsilon,\mathbf{x}),\mathbf{x})}{1 - D(G(\epsilon,\mathbf{x}),\mathbf{x})} - \log p_\theta(\mathbf{x}|G(\epsilon,\mathbf{x}))\right]\right]\end{aligned}$$

Notice that $\mathcal{L}^G$ also depends on $p_\theta(\mathbf{x}|\mathbf{z})$, hence, at each iteration the loss function needs to be minimized with respect to the set of parameters θ. Dumoulin et al. (2016) propose an alternative approach that does not depend on the capacity to evaluate $p_\theta(\mathbf{x}|\mathbf{z})$. In this case, the objective function relies on the Kullback-Leibler divergence,

$$\begin{aligned}\mathbf{E}_p[D_{KL}(q_\gamma(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))] &= \mathbb{E}_p\left[\mathbb{E}_{z\sim q_\gamma}\left[\log\frac{q_\gamma(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})}\right]\right] \\ &= \mathbb{E}_p\left[\mathbb{E}_{z\sim q_\gamma}\left[\log\frac{q_\gamma(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p_\theta(\mathbf{x}|\mathbf{z})q(\mathbf{z})}\right]\right]\end{aligned} \tag{32}$$

The discriminator D needs to be trained to distinguish between a pair $(\mathbf{z}, \mathbf{x})$ where $\mathbf{x}$ is an observation and $\mathbf{z}$ comes from its posterior density, and a pair $(\mathbf{z}', \mathbf{x}')$ where $\mathbf{x}'$ comes from $p_\theta(\mathbf{x}|\mathbf{z})$ and $\mathbf{z}'$ comes from the prior density. The loss function is given by

$$\begin{aligned}
\mathcal{L}^D &= -\mathbb{E}_p \left[\mathbb{E}_{z\sim q_\gamma}[\log D(\mathbf{z}, \mathbf{x})]\right] + \mathbb{E}_{p_\theta} \left[\mathbb{E}_{z\sim q}[\log (1 - D(\mathbf{z}, \mathbf{x}))]\right] \\
&= -\mathbb{E}_p \left[\mathbb{E}_\epsilon \log[D(G(\epsilon, \mathbf{x}), \mathbf{x})]\right] + \mathbb{E}_{p_\theta} \left[\mathbb{E}_{z\sim q}[\log (1 - D(\mathbf{z}, \mathbf{x}))]\right]
\end{aligned}$$

Based in equation (32), the generator loss function is defined as

$$\begin{aligned}
\mathcal{L}^G &= \mathbb{E}_p \left[\mathbb{E}_{z\sim q_\gamma} \left[\log \frac{D(\mathbf{z}, \mathbf{x})}{1 - D(\mathbf{z}, \mathbf{x})}\right]\right] \\
&= \mathbb{E}_p \left[\mathbb{E}_\epsilon \left[\log \frac{D(G(\epsilon, \mathbf{x}), \mathbf{x})}{1 - D(G(\epsilon, \mathbf{x}), \mathbf{x})}\right]\right]
\end{aligned}$$

4.2.1 Example

Assume the $\mathbf{z} = [z_1, z_2]$ is a two-dimensional latent variable and its prior distribution q corresponds to $N(0, \sigma^2\mathbf{I})$, where $\sigma^2 = 2$. On the other hand, $\mathbf{x}|\mathbf{z}$ is an exponentially distributed random variable,

$$\mathbf{x}|\mathbf{z} \sim \text{Exp}\left(\frac{1}{\beta}\right), \text{ where } \beta = 1 + z_1^2 + z_2^2.$$

The conditional density of $\mathbf{x}$ given $\mathbf{z}$ is denoted $p_\beta(\mathbf{x}|\mathbf{z})$. The latent variable posterior log-density function

$$\log p(\mathbf{z}|\mathbf{x}) \propto \log p_\beta(\mathbf{x}|\mathbf{z}) + \log q(\mathbf{z}) \tag{33}$$

We have that

$$\log q(\mathbf{z}) = -\log 2\pi - \log \sigma^2 - \frac{1}{2\sigma^2}\mathbf{z}^T\mathbf{z}$$

and

$$\log p_\beta(\mathbf{x}|\mathbf{z}) = -\log \beta - \frac{\mathbf{x}}{\beta}.$$

Figure 6 shows the latent variable prior density contours plot. Figure 7 presents the contours of the unnormalized posterior density $p(\mathbf{z}|\mathbf{x})$ for $\mathbf{x} = 1$ and $\mathbf{x} = 30$. In the first case, mass is concentrated in a small area around cero. In the later case, the density function takes a ring shape around cero with no mass in the center.

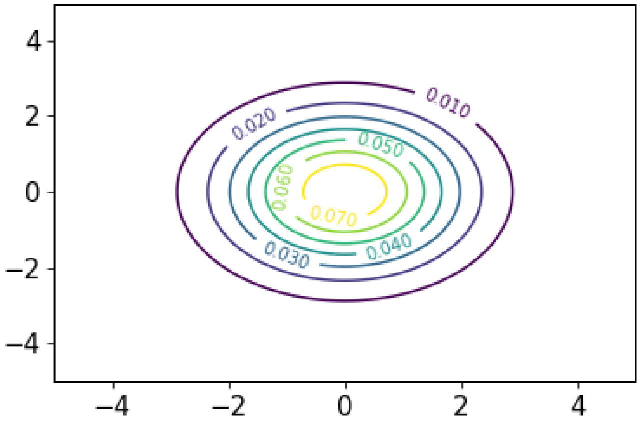
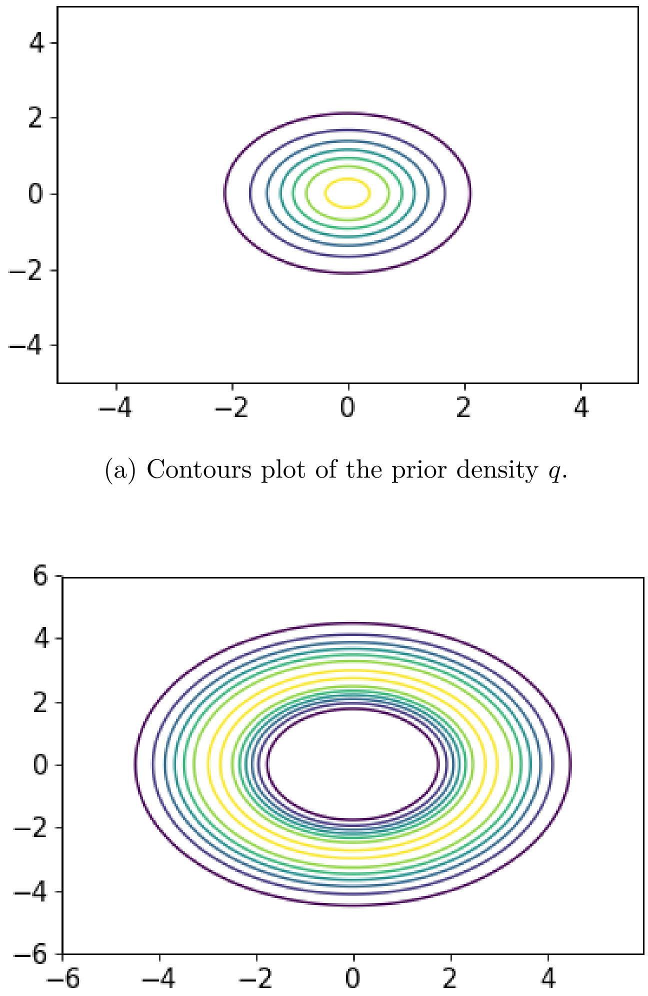


Figure 6: Point in euclidean space and $\epsilon/2$-balls around.

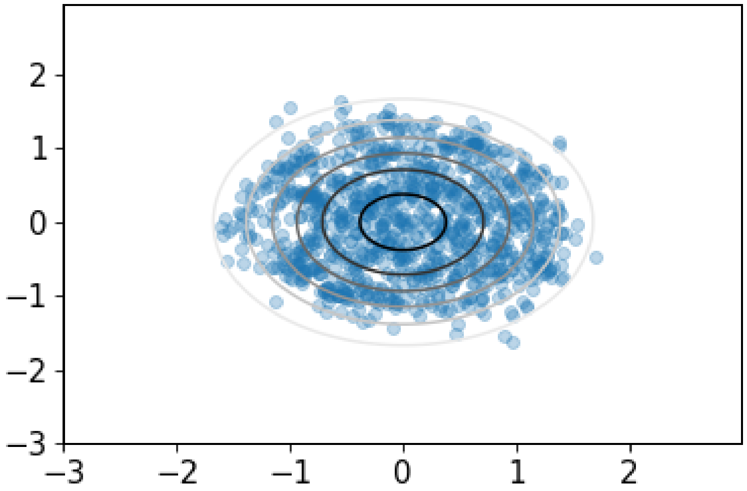


(a) Contours plot of the prior density q.

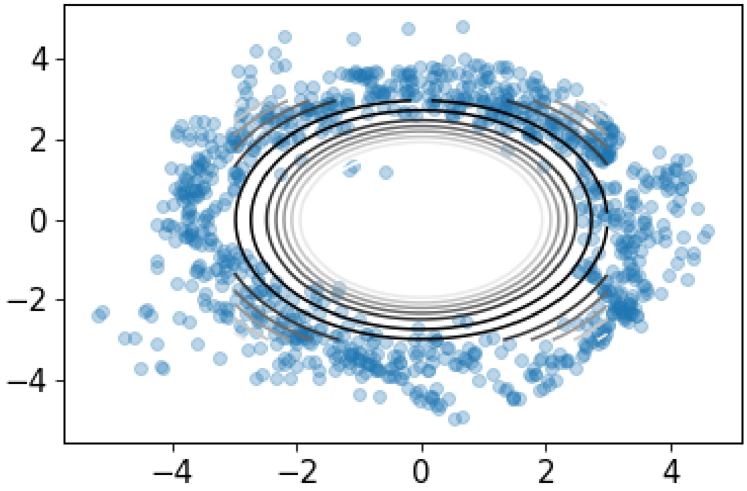
(b) Contours plot of the unnormalized posterior density $p(\mathbf{z}|\mathbf{x} = 30)$.

Figure 7: Plots of the contours of the unnormalized posterior density $p(\mathbf{z}|\mathbf{x})$ for $\mathbf{x} = 1$ and $\mathbf{x} = 30$.

The objective is to train a generator neural network, using the adversarial scheme, to sample from the latent variable posterior distribution for which we do not have a closed-form expression. In this example the generator network has 3 hidden layers. Each layer has 20 units with *tanh* activation functions. The network is fed with $\mathbf{x}$ values and noise coming from a 3-dimensional standard normal distribution. The discriminator network has also 3 hidden layers. Each layer has 20 units with *tanh* and *softplus* activation functions, and *sigmoid* output. The network is fed with sample $\mathbf{x}$ and $\mathbf{z}$.



(a) Contours plot of the unnormalized posterior density $p(\mathbf{z}|\mathbf{x} = 1)$ and 1000 data points generated from $q_\gamma(\mathbf{z}|\mathbf{x} = 1)$.



(b) Contours plot of the unnormalized posterior density $p(\mathbf{z}|\mathbf{x} = 30)$ and 1000 data points generated from $q_\gamma(\mathbf{z}|\mathbf{x} = 30)$.

Figure 8: Plots of the contours of the unnormalized posterior density $p(\mathbf{z}|\mathbf{x})$ for $\mathbf{x} = 1$ and $\mathbf{x} = 30$, together with 1000 data points produced by the generator's network.

Figure 8 portrays the contours of the unnormalized posterior density $q_\gamma(\mathbf{z}|\mathbf{x})$ for $\mathbf{x} = 1$ and $\mathbf{x} = 30$, and 1000 data points produced by the generator's network trained after 4000 epochs. The trained network achieves the goal of generating data consistent with the posterior density. For $\mathbf{x} = 1$ G yields samples concentrated in a small area around cero. For $\mathbf{x} = 1$ the samples form a ring as described by the contours.

5 Conclusions

GANs have become very popular during the last couple of years due to their capacity to produce data from high dimensional complex probability distributions. In practice, GANs are unstable and finding an optimal solution can be challenging. In order to increase the odds of reaching an optimum, different objective functions to train a generator network might be posed. However, when working with GANs is necessary to understand that a particular objective function has implications on the characteristics of the data generated by the algorithm and it might be important to do a careful analysis of these implications.

On the other hand, inference in GANs is not well understood. GANs is a black-box algorithm that returns appealing results. The lack of an inference model has led to search for connections with other models such as Variational Autoencoders, as the scheme presented in this document. In general, an analysis of this and other machine learning algorithm within a Bayesian framework can improve the the level of understanding these methods.

Acknowledgements

Thanks to Prof. Dr. Omiros Papaspiliopoulos, Prof. Dr. David Rossell and Dr. Alexandros Karatzoglou for introducing me to the interesting world of GANs, providing guide and helpful feedback.

References

- Arjovsky, Martin, Bottou, Leon. Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862, 2017.
- Arjovsky, Martin, Chintala, Soumith, Bottou, Leon. Wasserstein GAN. arXiv preprint arXiv:1701.07875, 2017.
- Dumoulin, Vincent, Belghazi, Ishmael, Poole, Ben, Lamb, Alex, Arjovsky, Martin, Mastropietro, Olivier, and Courville, Aaron. Adversarially learned inference. arXiv preprint arXiv:1606.00704, 2016.
- Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, Bengio, Yoshua. Generative Adversarial Nets. arXiv preprint arXiv:1406.2661, 2014.
- Hinton, Geoffrey E., Osindero, Simon, and Teh, Yee-Whye. A fast learning algorithm for deep belief nets. Neural Computation, 18, 1527–1554, 2006.
- Kingma, Diederik P., and Welling, Max. Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114, 2014.

- Kingma, Diederik P. Fast gradient-based inference with continuous latent variable models in auxiliary form. Technical report, arxiv:1306.0733, 2013.
- Larsen, Anders Boesen Lindbo, Sønderby, Søren Kaae, and Winther, Ole. Autoencoding beyond pixels using a learned similarity metric. arXiv preprint arXiv:1512.09300, 2015.
- Frey, Brendan, Goodfellow, Ian, Jaitly, Navdeep, Makhzani, Alireza, Shlens, Jonathon. Adversarial Autoencoders. arXiv preprint arXiv:1511.05644v2, 2016.
- Mescheder, Lars, Nowozin, Sebastian, Geiger, Andreas. Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks. arXiv preprint arXiv:1701.04722v2, 2017.
- Mohamed, Shakir, Lakshminarayanan, Balaji. Learning in Implicit Generative Models. arXiv preprint arXiv:1610.03483, 2017.
- Nowozin, Sebastian, Botond, Cseke. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. arXiv preprint arXiv:606.00709v1, 2016.
- Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training gans. In Advances in Neural Information Processing Systems, pages 2226–2234, 2016.
- Theis, Lucas, Van den Oord, Aaron, Bethge, Matthias. A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844, 2016.
- Van den Oord, Aaron, Kalchbrenner, Nal, and Kavukcuoglu, Koray. Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759, 2016a.
- Van den Oord, Aaron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew, and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499, 2016b.