Plan

- An Introduction To The Problem
- Vanilla GAN
- Wasserstein GAN
- Discrete Wasserstein GAN
- Experiments With Movielens-1M Sparse Data And Theoretical Analysis Of The results

Formulation Of The Problem

what does it mean to learn a probability distribution? The classical answer to this is to learn a probability density. This is often done by defining a parametric family

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

If the real data distribution $\mathbb{P}_r$ admits a density and $\mathbb{P}_\theta$ is the distribution of the parametrized density P_θ, then, asymptotically, this amounts to minimizing the Kullback-Leibler divergence $KL(\mathbb{P}_r \| \mathbb{P}_\theta)$.

About Kullback-Leibler Divergence

Entropy:
Shannon defined the entropy H of a discrete random variable X as:

$$\mathrm{H}(X) = \mathrm{E}[\mathrm{I}(X)] = \mathrm{E}[-\log(\mathrm{P}(X))].$$

$$H(X) = -\sum_{i} P_X(x_i) \log_b P_X(x_i)$$

With K-L divergence we can measure how a PDF is close to another PDF. Essentially, what we're looking at with the K-L divergence is the expectation of the log difference between the probability of data in the original distribution with the approximating distribution:

$$D_{KL}(p||q) = E[\log p(x) - \log q(x)]$$

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \cdot (\log p(x_i) - \log q(x_i))$$

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \cdot log\frac{p(x_i)}{q(x_i)}$$

Maximizing log likelihood is equivalent to minimizing KL divergence

Recall that for continuous distributions P and Q, the KL divergence is

$$KL(P||Q) = \int_x P(x) \log \frac{P(x)}{Q(x)} \, dx$$

In the limit (as $m \to \infty$), samples will appear based on the data distribution P_r, so

$$\begin{aligned}
\lim_{m \to \infty} \max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)}) &= \max_{\theta \in \mathbb{R}^d} \int_x P_r(x) \log P_\theta(x) \, dx \\
&= \min_{\theta \in \mathbb{R}^d} - \int_x P_r(x) \log P_\theta(x) \, dx \\
&= \min_{\theta \in \mathbb{R}^d} \int_x P_r(x) \log P_r(x) \, dx - \int_x P_r(x) \log P_\theta(x) \, dx \\
&= \min_{\theta \in \mathbb{R}^d} KL(P_r \| P_\theta)
\end{aligned}$$

GAN introduction

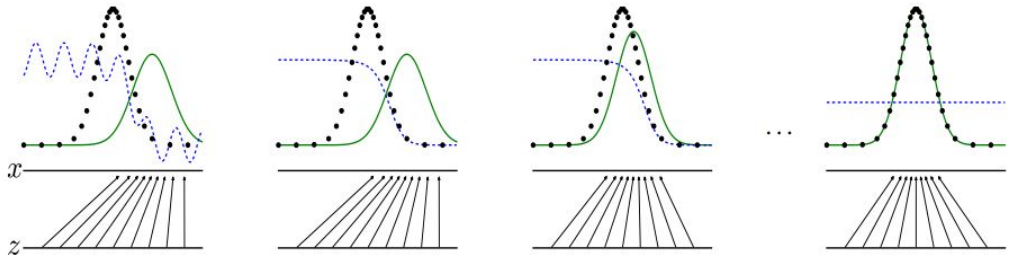
We simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to 1/2 everywhere

$$\max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]$$

$$\max_D \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\min_G \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$



Training of generative adversarial nets

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{3}$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \to a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, concluding the proof. □

Note that the training objective for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y = y|\boldsymbol{x})$, where Y indicates whether $\boldsymbol{x}$ comes from p_{data} (with $y = 1$) or from p_g (with $y = 0$). The minimax game in Eq. 1 can now be reformulated as:

$$
\begin{aligned}
C(G) =& \max_D V(G, D) \\
=& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} [\log(1 - D_G^*(G(\boldsymbol{z})))] \\
=& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [\log(1 - D_G^*(\boldsymbol{x}))] \\
=& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \right] + \mathbb{E}_{\boldsymbol{x} \sim p_g} \left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \right]
\end{aligned}
$$

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{\text{data}}$, $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

$$\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [-\log 2] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [-\log 2] = -\log 4$$

and that by subtracting this expression from $C(G) = V(D_G^*, G)$, we obtain:

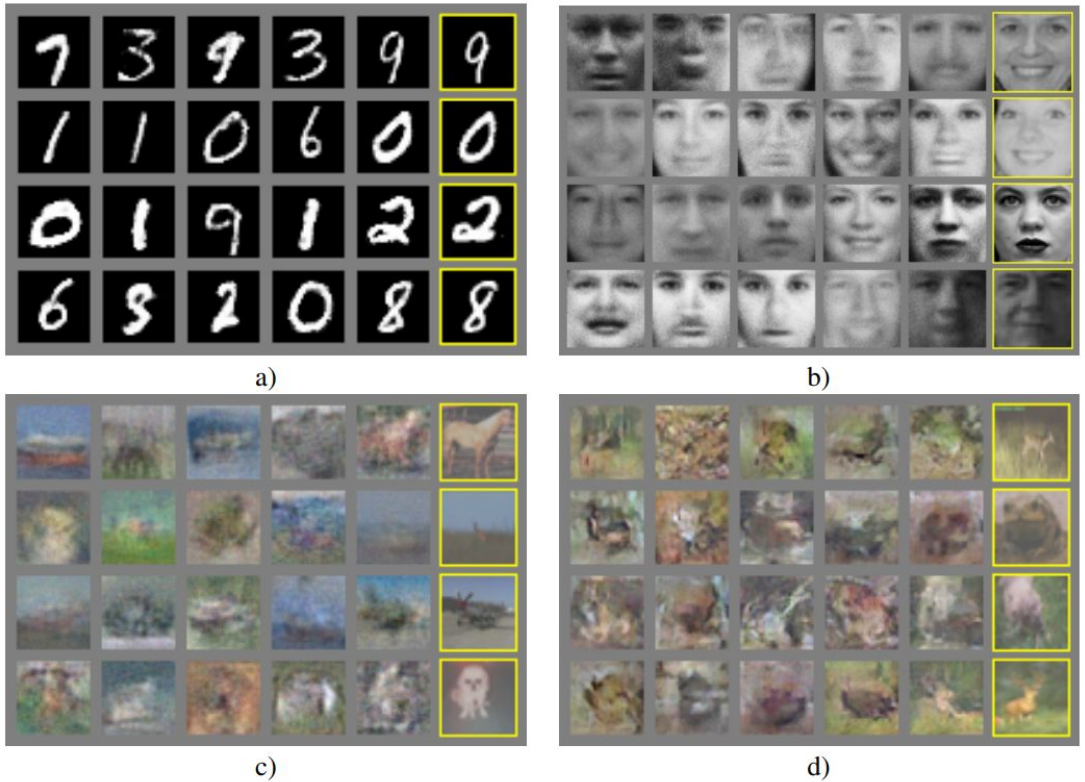
$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right) \tag{5}$$

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right) \tag{6}$$

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicating the data generating process.

Experiments



Different Distances

- The *Total Variation* (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \ .$$

- The *Kullback-Leibler* (KL) divergence

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log \left(\frac{P_r(x)}{P_g(x)} \right) P_r(x) d\mu(x) \ ,$$

where both $\mathbb{P}_r$ and $\mathbb{P}_g$ are assumed to be absolutely continuous, and therefore admit densities, with respect to a same measure μ defined on $\mathcal{X}$.[2] The KL divergence is famously assymetric and possibly infinite when there are points such that $P_g(x) = 0$ and $P_r(x) > 0$.

- The *Jensen-Shannon* (JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m) \ ,$$

where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_g)/2$. This divergence is symmetrical and always defined because we can choose $\mu = \mathbb{P}_m$.

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} \big[\, \|x - y\| \, \big] \ , \tag{1}$$

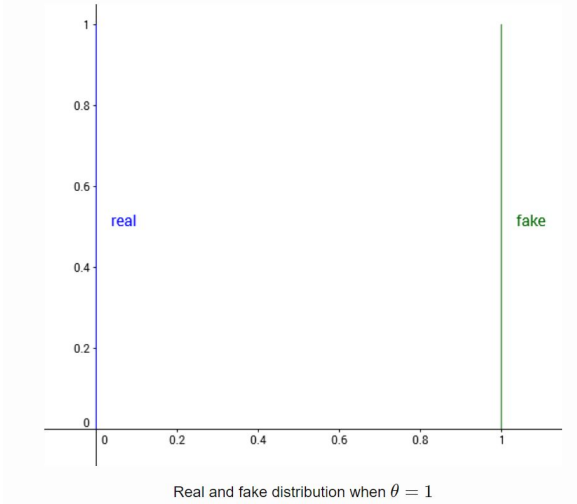
where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

Learning Parallel lines

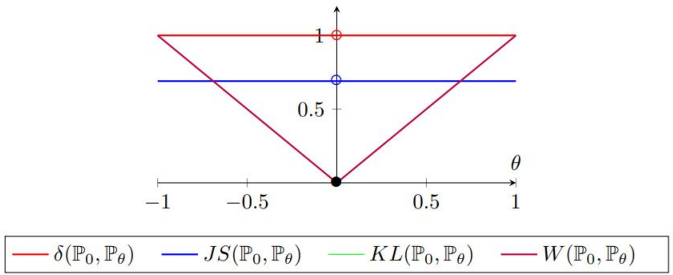
Example 1 (Learning parallel lines)**.** Let $Z \sim U[0,1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$ (a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with θ a single real parameter. It is easy to see that in this case,

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$,
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ . \end{cases}$

When $\theta_t \to 0$, the sequence $(\mathbb{P}_{\theta_t})_{t \in \mathbb{N}}$ converges to $\mathbb{P}_0$ under the EM distance, but does not converge at all under either the JS, KL, reverse KL, or TV divergences. Figure 1 illustrates this for the case of the EM and JS distances.



Real and fake distribution when $\theta = 1$



Behavior of the Wasserstein distance for the parallel lines problem.

Optimal Transport And Wasserstein Distance

Example 8. *Suppose that the production (P) of avocados in Canada is split* 70% *in Montréal, and* 30% *in Québec City. However,* 60% *of the avocados are consumed (C) by the people in Québec City, while the remaining* 40% *are eaten by the Montrealers. Suppose that the cost of transporting one avocado is* $0.001\$$ *per mile, and the distance between Montréal and Québec City is 150 miles. What is the best transportation plan to fulfill the demand in both cities at the minimum cost?*

Tables 3 and 4 show the independent coupling (always possible to construct) and the optimal coupling (check this!). The transportation cost under the independent coupling is $0.081 = 0.54 * 0.001 * 150\$$ *per avocado, while the optimal transportation cost is* $0.045 = 0.3 * 0.001 * 150\$$. *Note the role the distance between the cities plays here!*

One can interpret the entries of π *as a transportation plan as follows:* $\frac{\pi(p,c)}{P(p)}$ *represents the proportion of avocados produced in location* p *that will be shipped to location* c. *For example, in the optimal coupling, approximately* $57\% \approx \frac{0.4}{0.7}$ *of the avocados produced in Montréal remain there.*

P \ C	M	Q
M	0.28	0.42
Q	0.12	0.09

Table 3: Independent coupling

P \ C	M	Q
M	0.4	0.3
Q	0	0.3

Table 4: Optimal coupling

Definition 9 (Wasserstein-1 distance)**.** *Let* $\mathbb{P}$ *and* $\mathbb{Q}$ *be two distributions over a metric space* $(\mathcal{X}, d)$.

$$W(\mathbb{P}, \mathbb{Q}) = \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \mathbb{E}_{(x,y)\sim\pi}[d(x, y)] = \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \langle \pi, d \rangle,$$

Optimal Transport And Wasserstein Distance

$$W(P_r, P_s) = \inf_{T \in \mathbb{R}_+^{|\mathcal{X}| \times |\mathcal{X}|}} \langle T, D \rangle,$$

Example 1. *The following example provides a closer look at the dual optimization problem. Consider a finite set* $\mathcal{X} = \{1, 2, 3\}$. *Let* $P_s(x)$ *be given by the discrete distribution* $P_s(1) = 0.2$, $P_s(2) = 0.7$ *and* $P_s(3) = 0.1$. *Similarly, let* $P_r(x)$ *be given by the discrete distribution* $P_r(1) = 0.4$, $P_r(2) = 0.4$, $P_r(3) = 0.2$. *Define the elementary sample distance* $d_{\mathcal{X}}(x_1, x_2) = 1$ *if* $x_1 \neq x_2$ *and* $d_{\mathcal{X}}(x_1, x_2) = 0$ *if* $x_1 = x_2$. *Therefore, the sample distance matrix* D *for this discrete example is the following:*

$$D = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

The optimal value of the matrix T *provides the optimal transport of mass from* P_s *to* P_r,

$$T = \begin{bmatrix} 0.2 & 0.2 & 0.0 \\ 0.0 & 0.4 & 0.0 \\ 0.0 & 0.1 & 0.1 \end{bmatrix}.$$

Optimal transport 2 formulations

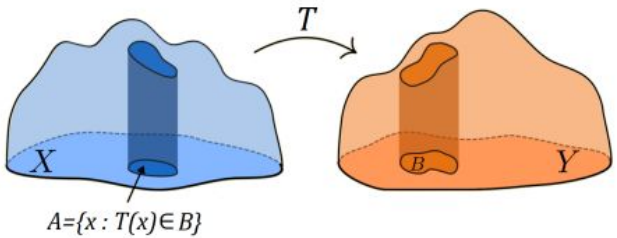


Figure 1.1: Monge's transport map, figure modified from Figure 1 in [9].

With this notation we can define Monge's optimal transport problem as follows.

Definition 1.3. ***Monge's Optimal Transport Problem:*** *given* $\mu \in \mathcal{P}(X)$ *and* $\nu \in \mathcal{P}(Y)$,

$$\textit{minimise } \mathbb{M}(T) = \int_X c(x, T(x))\, \mathrm{d}\mu(x)$$

Definition 1.4. ***Kantorovich's Optimal Transport Problem:*** *given* $\mu \in \mathcal{P}(X)$ *and* $\nu \in \mathcal{P}(Y)$,

$$\textit{minimise } \mathbb{K}(\pi) = \int_{X \times Y} c(x, y)\, \mathrm{d}\pi(x, y)$$

over $\pi \in \Pi(\mu, \nu)$.

Kantorovich-Rubinstein duality

Definition 1.4. ***Kantorovich's Optimal Transport Problem:*** *given* $\mu \in \mathcal{P}(X)$ *and* $\nu \in \mathcal{P}(Y)$,

$$\textit{minimise } \mathbb{K}(\pi) = \int_{X \times Y} c(x, y)\, \mathrm{d}\pi(x, y)$$

over $\pi \in \Pi(\mu, \nu)$.

Kantorovich-Rubinstein duality says that they're equivalent problems

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)] \tag{2}$$

where the supremum is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$. Note that if we replace $\|f\|_L \leq 1$ for $\|f\|_L \leq K$ (consider K-Lipschitz for some constant K), then we end up with $K \cdot W(\mathbb{P}_r, \mathbb{P}_g)$.

Final loss function for neural network setup

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z)]$$

And again min-max game to train the full system

$$\min_{\theta \in \Theta} \max_{\in} \mathbb{E}_{x \sim \mathbb{P}}[f\ (x)] - \mathbb{E}_{z \sim \mathbb{P}_z}[f\ (g_\theta(z))]$$

Now comes the question of finding the function f that solves the maximization problem in equation (2). To roughly approximate this, something that we can do is train a neural network parameterized with weights w lying in a compact space $\mathcal{W}$ and then backprop through $\mathbb{E}_{z \sim p(z)}[\nabla_\theta f_w(g_\theta(z))]$, as we would do with a typical GAN. Note that the fact that $\mathcal{W}$ is compact implies that all the functions f_w will be K-Lipschitz for some K that only depends on $\mathcal{W}$ and not the individual weights, therefore approximating (2) up to an irrelevant scaling factor and the capacity of the 'critic' f_w. In order to have parameters w lie in a compact space, something simple we can do is clamp the weights to a fixed box (say $\mathcal{W} = [-0.01, 0.01]^l$) after each gradient update.

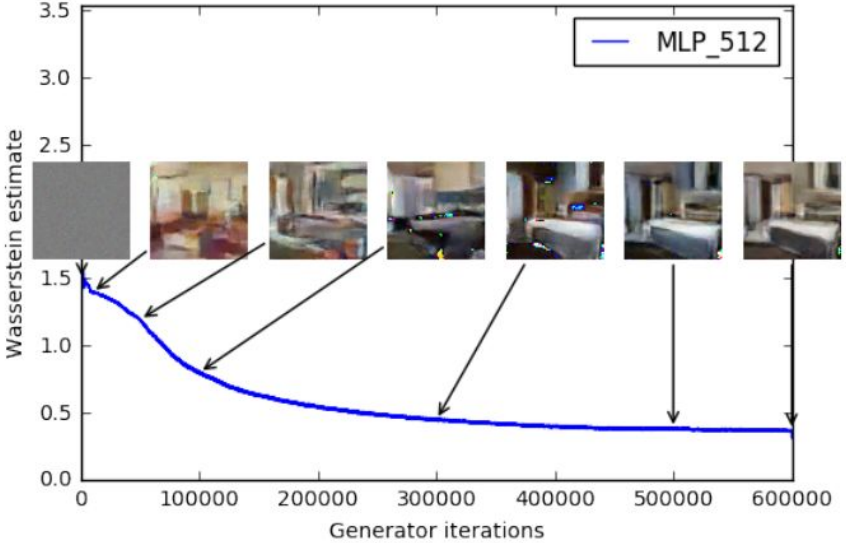
Training of WGAN

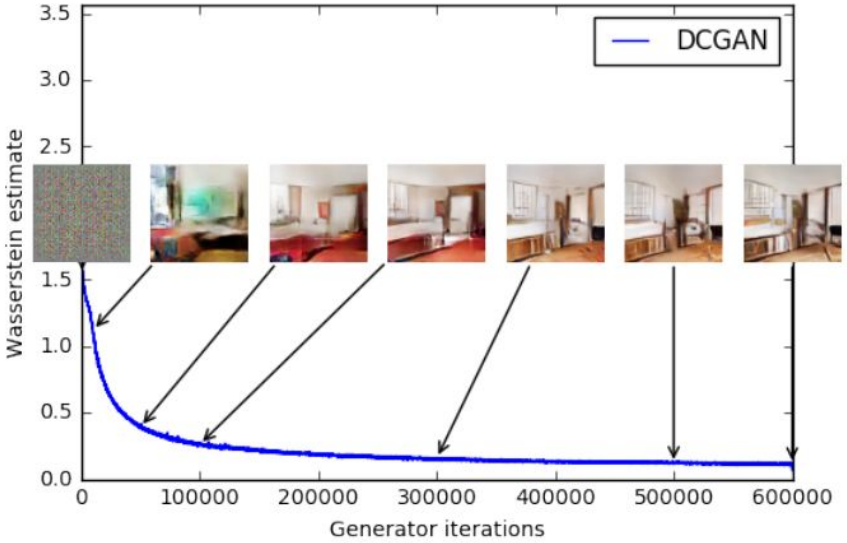
Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Experiments





Introduction

Table 1: Example of a mismatch between continuous distance and discrete distance.

TRAINING SAMPLE	GENERATOR's SOFTMAX OUTPUT	GENERATOR'S SAMPLE (ARGMAX, ONE-HOT)	DISCRETE DISTANCE	CONTINUOUS DISTANCE
$\begin{bmatrix}0&0&1\\0&1&0\end{bmatrix}$	$\begin{bmatrix}0.3&0.3&0.4\\0.3&0.4&0.3\end{bmatrix}$	$\begin{bmatrix}0&0&1\\0&1&0\end{bmatrix}$	0	1.04
$\begin{bmatrix}0&0&1\\0&1&0\end{bmatrix}$	$\begin{bmatrix}0.1&0.1&0.8\\0.5&0.3&0.2\end{bmatrix}$	$\begin{bmatrix}0&0&1\\1&0&0\end{bmatrix}$	1	0.92

Table 2: Example of a large gap between discrete and continuous distances for a discrete sample with 9 classes.

SAMPLES	DISCRETE DISTANCE	CONTINUOUS DISTANCE
Training sample: $\begin{bmatrix}0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0\end{bmatrix}$		
Generator's softmax output: $\begin{bmatrix}0.1&0.1&0.2&0.1&0.1&0.1&0.1&0.1&0.1\\0.1&0.1&0.1&0.1&0.1&0.1&0.2&0.1&0.1\end{bmatrix}$	0	1.2
Generator's sample (argmax, onehot): $\begin{bmatrix}0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0\end{bmatrix}$		

The usage of continuous sample distance in the standard Wasserstein GAN for discrete problems as described above creates some discrepancies in the model. These discrepancies are illustrated in Table 1 and Table 2. In Table 1, we have two different outputs from the generator's softmax with the same real sample reference. Although the first softmax output produces the same value as the real sample when it is rounded using argmax (hence has discrete distance 0 to the real sample), it has a larger continuous distance compared to the second softmax output which produces one mistake when rounded (has discrete distance 1 to the real sample). In the discrete case, with a large number of classes, as shown in Table 2, even though the generator output produces a discrete sample with the same value as the real sample when rounded, there still exists a very large continuous distance. This difference between continuous and discrete distance becomes greater for a larger number of discrete classes.

Use of Wasserstein Distance for Discrete Problems

Although the formulation of the Wasserstein GAN approximates the Wasserstein distance between two continuous distributions, using a continuous sample distance $\|x - y\|$, existing research efforts (Gulrajani et al., 2017; Subramanian et al., 2017; Press et al., 2017) have directly used it to model discrete probability distributions by adding the following modifications. Each component of the input vectors of training data is encoded in a one-hot representation. A softmax nonlinearity is applied in the last layer of the output of the generator to produce a probability that corresponds with the one-hot representation of the training data. During training, the output of the softmax layers becomes the input to the critic network without any rounding step. To generate a new sample, an argmax operation over each generator's softmax output vectors is applied to produce a valid discrete sample.

A vector of size **m** is mapped to a matrix of size **(mxk)** For example: (0 4 5 0) becomes a matrix (4x5):

0 0 0 0 0
0 0 0 1 0
0 0 0 0 1
0 0 0 0 0

Wasserstein Distance For Discrete Distribution

Let a vector $\mathbf{x} = (\mathbf{x}(1), \mathbf{x}(2), \ldots)$ be a discrete multivariate random variable where each component $\mathbf{x}(i)$ can take discrete values from $\{1, 2, 3, \ldots, k\}$. Let P_r and P_s be two probability distributions over the set of values for $\mathbf{x}$. The Wasserstein distance between two probability distributions P_r and P_s is defined as:

$$\mathbf{W}(P_r, P_s) = \min_{\gamma \in \Pi(P_r, P_s)} \mathbb{E}_{(\mathbf{x}, \mathbf{x}') \sim \gamma} \left[d(\mathbf{x}, \mathbf{x}') \right] = \min_{\gamma \in \Pi(P_r, P_s)} \sum_i \sum_j \gamma(\mathbf{x}_i, \mathbf{x}_j) d(\mathbf{x}_i, \mathbf{x}_j). \quad (2)$$

The notation $\Pi(P_r, P_s)$ denotes the set of all joint probability distributions $\gamma(\mathbf{x}, \mathbf{x}')$ whose marginals are P_r and P_s respectively, and $d(\mathbf{x}_i, \mathbf{x}_j)$ denotes the elementary distance between two samples $\mathbf{x}_i$ and $\mathbf{x}_j$. We are particularly interested with the sample distance that is defined as the hamming distance (the sum of zero-one distance of each component), i.e:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_k \mathbb{I}(\mathbf{x}_i(k) \neq \mathbf{x}_j(k)). \quad (3)$$

Table 3 shows an example of the sample distance metric.

Visible in the formulation above, computing the Wasserstein distance between two discrete probability distributions is a Linear Program (LP) problem for which the runtime is polynomial with respect to the size of problem. However, for generating real-world discrete distributions, the size of problem grows exponentially. For example, if the number of variables in vector $\mathbf{x}$ is 100, and each variable can take values in the set $\{1, 2, \ldots, 10\}$ so that $k = 10$, the size of the LP problem is $O(10^{100})$ reflecting the number of configurations for $\mathbf{x}$. The resulting LP is intractable to solve.

DWGAN Architecture

With Kantorovich duality we can instead solve the following optimization problem:

$$\max_f \mathbb{E}_{\mathbf{x}\sim P_r}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim P_s}[f(\mathbf{x})] \tag{4}$$

$$\text{subject to: } f(\mathbf{x}_i) - f(\mathbf{x}_j) \leq d(\mathbf{x}_i.\mathbf{x}_j), \tag{5}$$

The function f maps a sample to a real value. Note that unlike for the continuous Wasserstein distance, in which the maximization is over all 1-Lipschitz functions without additional constraints, the maximization above is over all functions that satisfy the inequality constraints in Eq. 5.
We note that the maximization in the dual formulation is equivalent to the following optimization:

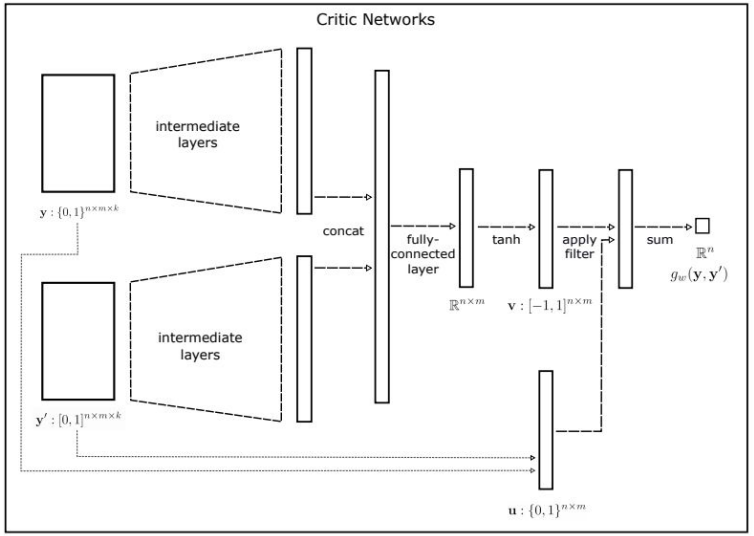
$$\max_h \mathbb{E}_{(\mathbf{x},\mathbf{x}')\sim(P_r,P_s)}[h(\mathbf{x},\mathbf{x}')] \tag{6}$$

$$\text{subject to: } h(\mathbf{x}_i,\mathbf{x}_j) \leq d(\mathbf{x}_i,\mathbf{x}_j), \tag{7}$$

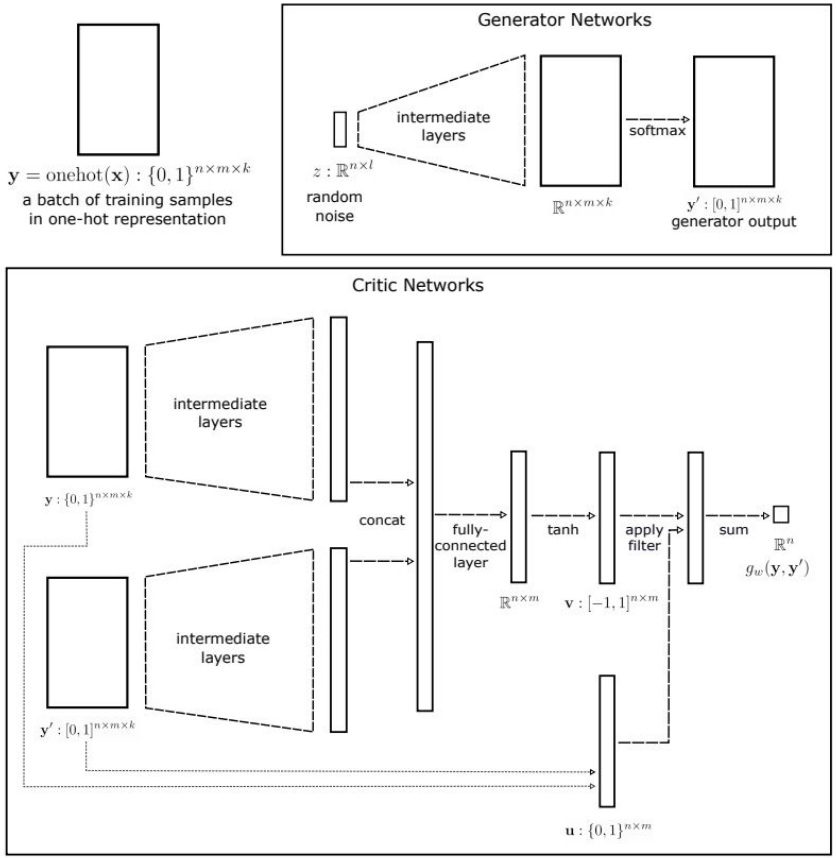
where $h(\mathbf{x},\mathbf{x}') = f(\mathbf{x}) - f(\mathbf{x}')$. Instead of approximating $f(\mathbf{x})$, we aim to design a neural network architecture that approximates $h(\mathbf{x},\mathbf{x}')$ and satisfies the inequality constraints in Eq. 5. The key idea is that this new optimization is equivalent to the original dual formulation of the Wasserstein distance (explained in the sequel), even though the optimal form for h is not explicitly specified.

DWGAN Architecture

Let $\mathbf{y} \in [0,1]^{m \times k}$ be the one-hot representation of $\mathbf{x}$ where m is the number of variables and k is the number of classes for each variable. The critic network takes two inputs: $\mathbf{y}$ from the real training data, and $\mathbf{y}'$ from the output of the generator network. Let us define ρ_w as a parameterized function that takes input $(\mathbf{y}, \mathbf{y}') \in [0,1]^{2 \times m \times k}$ and produces an output vector $\mathbf{v} \in [-1,1]^m$. From the generator output $\mathbf{y}'$, we compute the rounded sample $\tilde{\mathbf{x}}'$. Let $\mathbf{u} \in \{0,1\}^m$ be a vector that contains the element-wise zero one distance between a real training sample $\mathbf{x}$ and rounded sample $\tilde{\mathbf{x}}'$ from the generator, i.e. $\mathbf{u}(i) = \mathbb{I}(\mathbf{x}(i) \neq \tilde{\mathbf{x}}'(i))$. We define our approximation to the function h as a parameterized function h_w that is defined as $h_w = \mathbf{u}^T \mathbf{v} = \mathbf{u}^T \rho_w(\mathbf{y}, \mathbf{y}')$. The "filter" vector $\mathbf{u}$ ensures that the output of h_w always satisfies the inequality constraints $h_w(\texttt{onehot}(\mathbf{x}_i), \texttt{onehot}(\mathbf{x}_j)) \leq d(\mathbf{x}_i, \mathbf{x}_j)$



DWGAN Architecture



DWGAN Learning Algorithm

Algorithm 1 Discrete Wasserstein GAN

Input: learning rate α, batch size n, the number of critic iteration per generator iteration n_{critic}
repeat
  for $t = 1, \ldots, n_{\text{critic}}$ **do**
    Sample a batch from real data $\{\mathbf{x}_i\}_{i=1}^{n} \sim P_r$
    Sample a batch of random noise $\{z_i\}_{i=1}^{n} \sim p(z)$
    $w \leftarrow w + \alpha \cdot \nabla_w [\frac{1}{n} \sum_{i=1}^{n} h_w(\mathbf{x}_i, g_\theta(z_i))]$
  end for
  Sample a batch from real data $\{\mathbf{x}_i\}_{i=1}^{n} \sim P_r$
  Sample a batch of random noise $\{z_i\}_{i=1}^{n} \sim p(z)$
  $\theta \leftarrow \theta - \alpha \cdot \nabla_\theta [\frac{1}{n} \sum_{i=1}^{n} h_w(\mathbf{x}_i, g_\theta(z_i))]$
until converge

Experiments

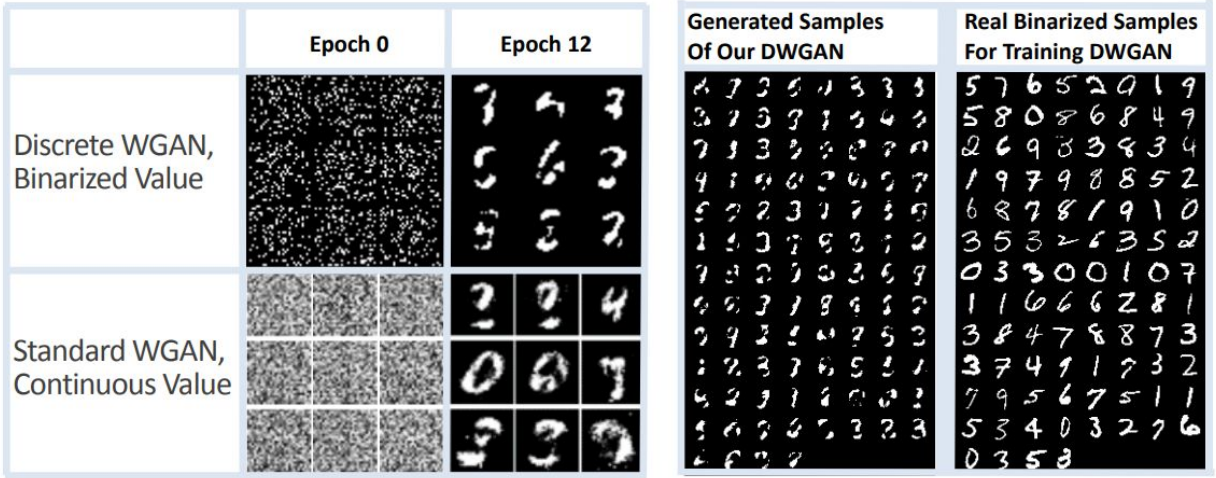


Figure 6: An example of Discrete WGAN with binary values vs. Standard WGAN with continuous values applied to generate MNIST handwritten digits. Both models feature 1 hidden layer for both the generator and critic within a fully-connected network. Modeling complex discrete distributions with GANs still requires future refinements in optimization, training, and stability.