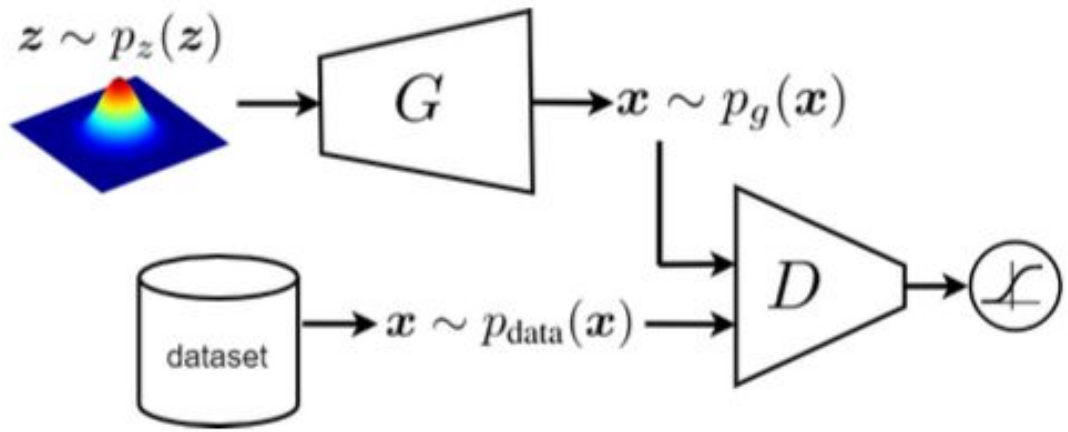
GAN

$$D(\boldsymbol{x}) := \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ is real,} \\ 0 & \text{if } \boldsymbol{x} \text{ is generated (fake).} \end{cases} \tag{2}$$



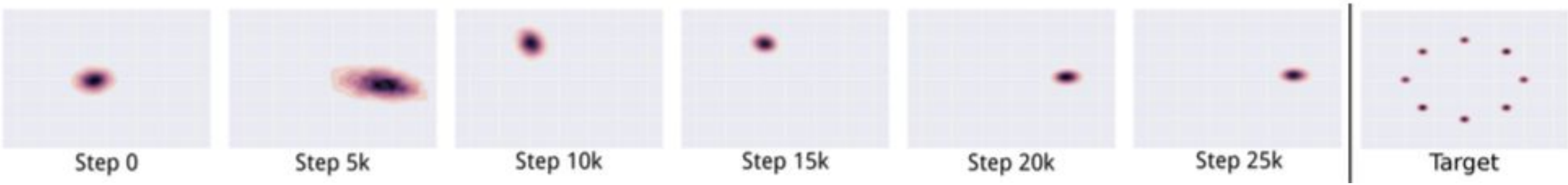
$$\min_G \max_D \; V(D,G) := \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\Big[\log\big(D(\boldsymbol{x})\big)\Big] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}\Big[\log\Big(1 - D\big(G(\boldsymbol{z})\big)\Big)\Big], \tag{3}$$

Explanation

- The second term in Eq. (3) is expectation over noise. It inputs the noise to the generator to have G(z). The output of generator, which is the generated point, is fed as input to the discriminator (see Fig. 1) to have D G(z).
- The discriminator wants to minimize D(G(z)) because the smaller label is assigned to the generated data, according to Eq. (2). In other words, the discriminator wants to maximize 1−D(G(z)). As logarithm is a monotonic function, we can say that the discriminator wants to maximize Ez~pz(z)[log(1−D(G(z)))] which is the second term in Eq. (3).
- As opposed to the discriminator, the generator minimizes Ez~pz(z)[log(1−D(G(z)))] which is the second term in Eq. (3). This is because the generator wants to fool the discriminator to label the generated data as real data.

Mode collapse

- We expect from a GAN to learn a meaningful latent space of z so that every specific value of z maps to a specific generated data point x. Also, nearby z values in the latent space should be mapped to similar but a little different generations.
- The mode collapse problem (Metz et al., 2017), also known as the Helvetica scenario (Goodfellow, 2016), is a common problem in GAN models. It refers to when the generator cannot learn a perfectly meaningful latent space as was explained. Rather, it learns to map several different z values to the same generated data point. Mode collapse usually happens in GAN when the distribution of training data, pdata(x), has multiple modes



An illustration of the mode collapse problem on a two-dimensional toy dataset. In the top row, we see the target distribution pdata that the model should learn. It is a mixture of Gaussians in a two-dimensional space. In the lower row, we see a series of different distributions learned over time as the GAN is trained. Rather than converging to a distribution containing all of the modes in the training set, the generator only ever produces a single mode at a time, cycling between different modes as the discriminator learns to reject each one. Images from Metz et al. (2016).

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

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

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:

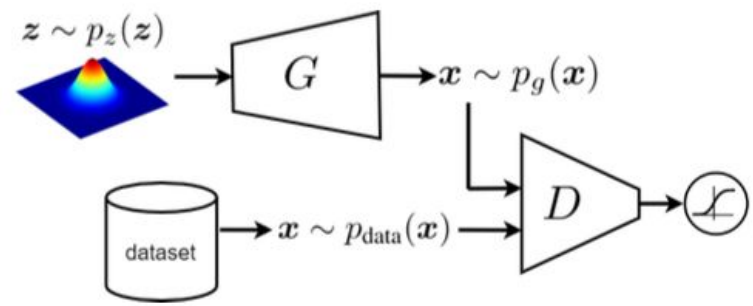
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  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

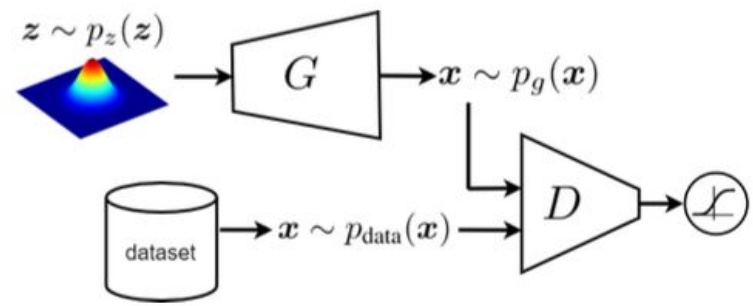


Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

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



the optimal discriminator D is

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

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

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

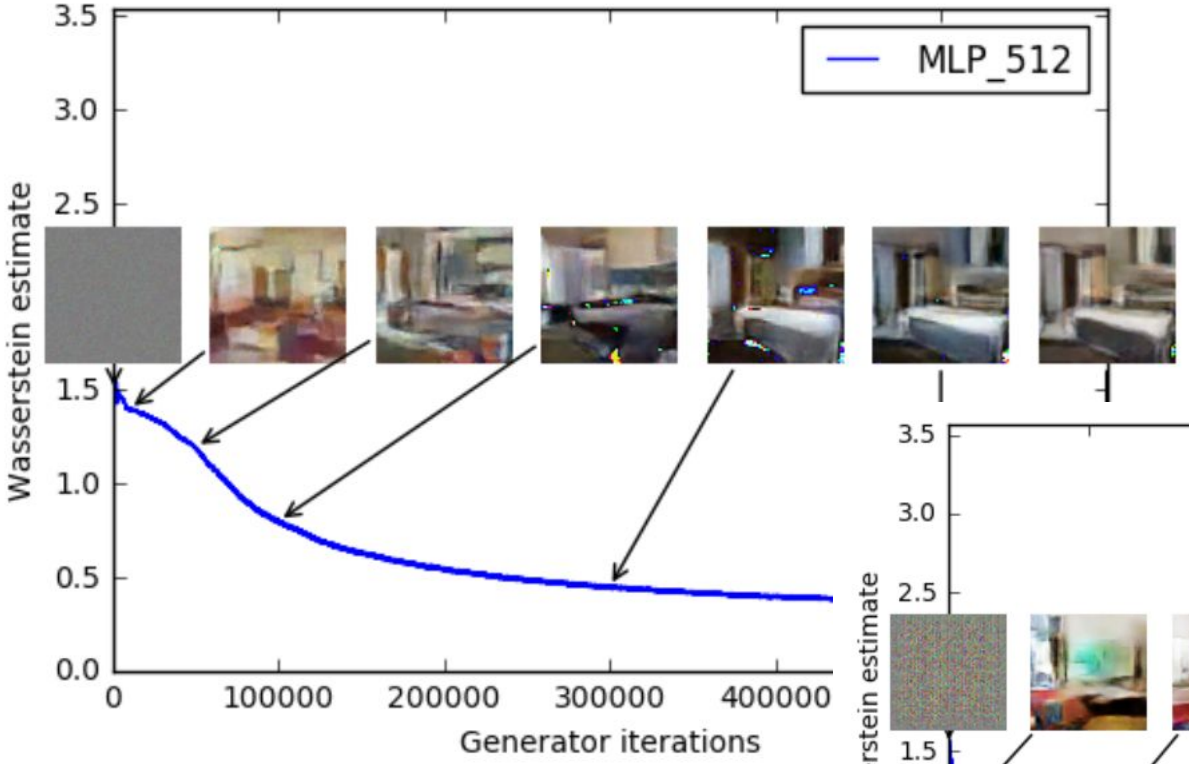
and that by subtracting this expression from $C(G) = V(D_G^*, G)$, we obtain:

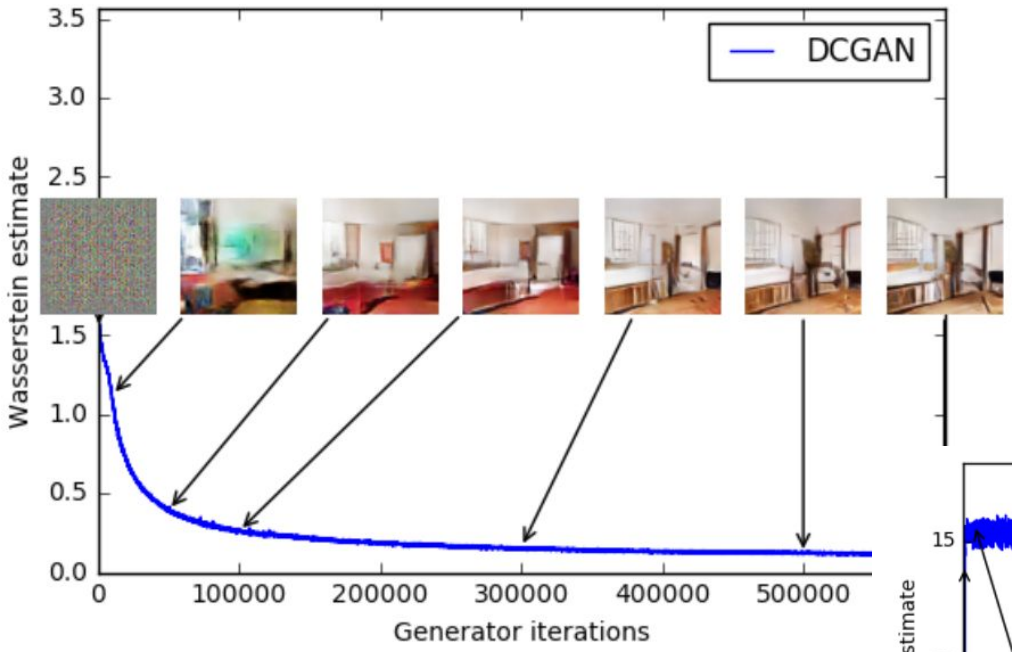
$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right) \tag{5}$$

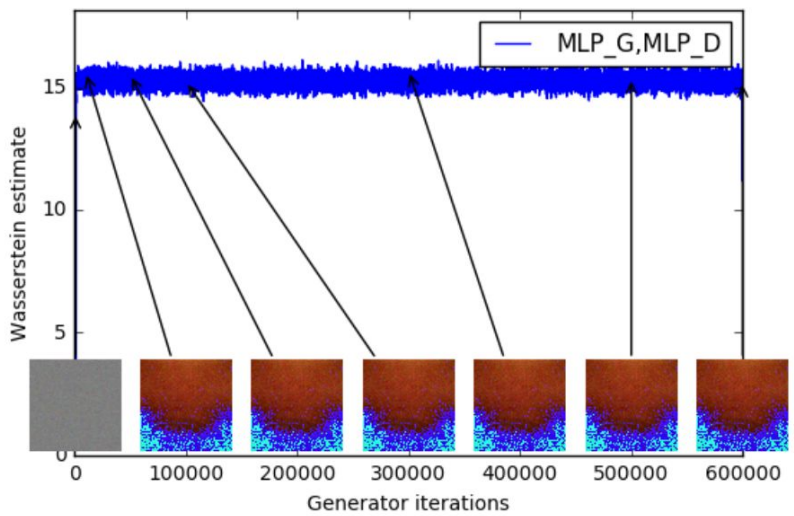
where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

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

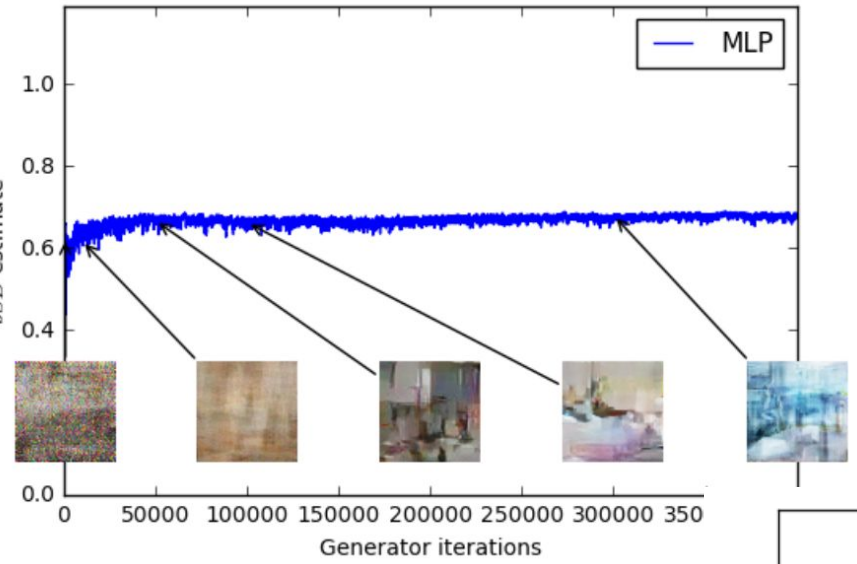
WGA
N

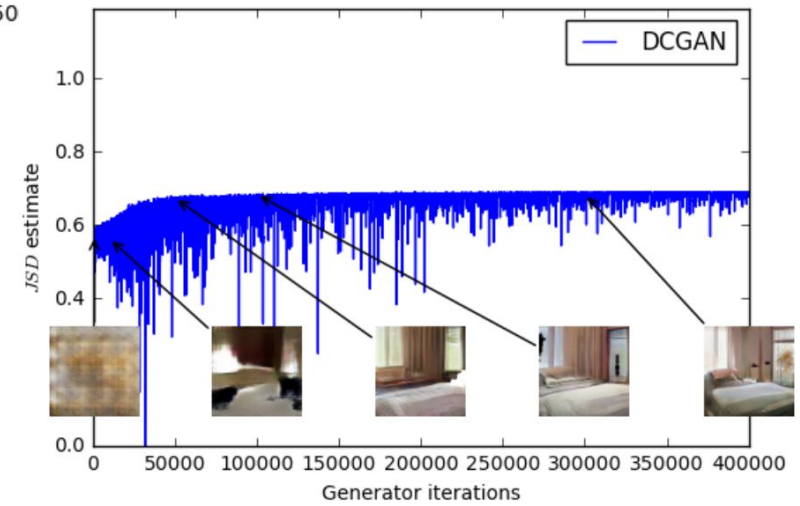

MLP_512
Wasserstein estimate
Generator iterations
3.5
3.0
2.5
1.5
1.0
0.5
0.0
0
100000
200000
300000
400000

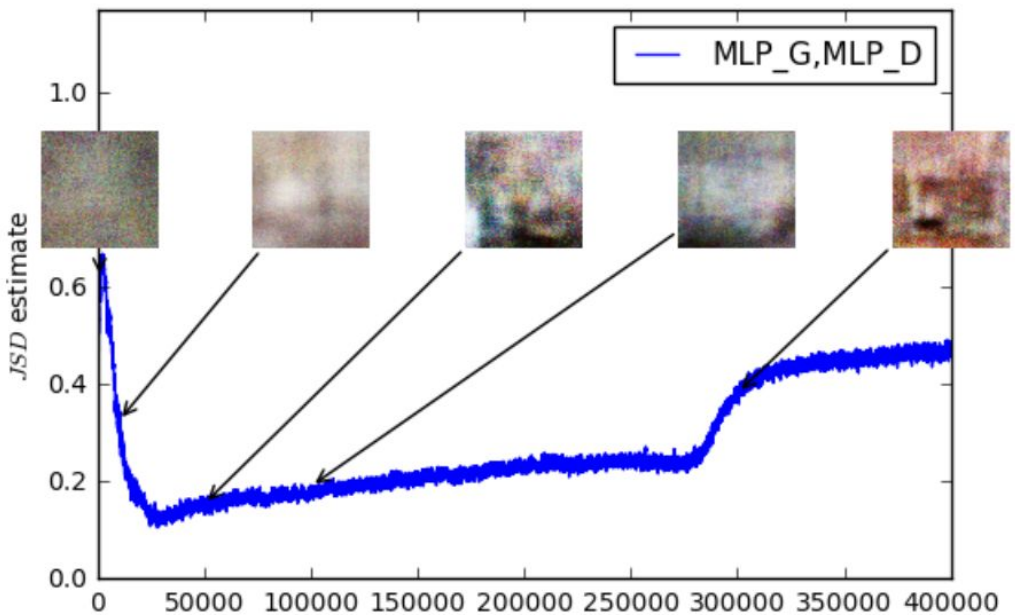

DCGAN
Wasserstein estimate
Generator iterations
3.5
3.0
2.5
1.5
1.0
0.5
0.0
0
100000
200000
300000
400000
500000


MLP_G,MLP_D
Wasserstein estimate
Generator iterations
15
10
5
0
0
100000
200000
300000
400000
500000
600000

Standard
GAN







Algorithm 1 OT Distance for batch of m samples with Sinkhorn-Knopp algorithm

Input: gen. dance sequences $\widetilde{\mathbf{X}} = \{\tilde{x}_i\}_{i=1}^{m}$
Input: data dance sequences $\mathbf{X} = \{x_j\}_{j=1}^{m}$
Hyperparameters: regularization ϵ, Sinkhorn iterations L
Dance Cost Matrix $C_{ij} = c_R(\tilde{x}_i, x_j)$ from Eqn 4,
$K = \exp(-C/\epsilon)$
$\mathbf{b}^{(0)} = \mathbb{1}_m$ where $\mathbb{1}_m = (1, \cdots, 1)^T \in \mathbb{R}^m$
for $\ell = 1 : L$ **do**
$\mathbf{a}^{(\ell)} = \mathbb{1}_m \oslash K\mathbf{b}^{(\ell-1)}$, $\mathbf{b}^{(\ell)} = \mathbb{1}_m \oslash K^T\mathbf{a}^{(\ell)}$
$\oslash$ denotes component-wise division
end for
Output: $OT_\epsilon(\widetilde{\mathbf{X}}, \mathbf{X}) = \sum_{i,j} C_{ij} a_i^{(L)} K_{ij} b_j^{(L)}$

Music-to-Dance Generation with Optimal Transport, IJCAI22

Consider the primal problem

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \le 0, \quad i = 1, \dots, m, \\ & h_j(x) = 0, \quad j = 1, \dots, p, \end{aligned}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$.

Lagrangian. Introduce multipliers $\lambda \in \mathbb{R}^m_+$ and $\nu \in \mathbb{R}^p$:

$$\mathcal{L}(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i \, g_i(x) + \sum_{j=1}^{p} \nu_j \, h_j(x).$$

Dual function (always concave).

$$g(\lambda, \nu) = \inf_{x \in \mathcal{X}} \; \mathcal{L}(x, \lambda, \nu).$$

Dual problem.

$$\max_{\lambda \ge 0, \; \nu \text{ free}} \; g(\lambda, \nu).$$

Weak duality

$$g(\lambda, \nu) \;\le\; f(x).$$

optimal value of the maximization problem agrees with the optimal value of the
minimization problem

Admits strong duality

Max and min can be switched.

Inception score
Improved Techniques for Training GANs
2016

- The images generated should contain clear objects (i.e. the images are sharp rather than blurry), or p(y|x) should be low entropy. In other words, the Inception Network should be highly confident there is a single object in the image.
- The generative algorithm should output a high diversity of images from all the different classes in ImageNet, or p(y) should be high entropy.

- We can construct an estimator of the Inception Score from samples x (i) by first constructing an empirical marginal class distribution,

$$\hat{p}(y) = \frac{1}{N}\sum_{i=1}^{N} p(y|\mathbf{x}^{(i)}),$$

$$\mathrm{IS}(G) \approx \exp(\frac{1}{N}\sum_{i=1}^{N} D_{KL}(p(y|\mathbf{x}^{(i)} \parallel \hat{p}(y)))$$

- P(y|x) = ?

A Note on the Inception Score, Shane Barratt Rishi Sharma, 2018

W-2 distance

- Frechet Inception distance (FID)

- Get mean and covariance from features of both generated and real images.

$$\mathrm{dist}_F^2(P,Q) = \|\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q\|_2^2 + \mathrm{Tr}(\boldsymbol{\Sigma}_P + \boldsymbol{\Sigma}_Q - 2(\boldsymbol{\Sigma}_P\boldsymbol{\Sigma}_Q)^{\frac{1}{2}}),$$

MMD

$k(x, y) = \exp(- \| x - y \|^2 / 2\sigma^2)$

Given two sets of vectors , $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ and $Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$, sampled from P and Q, respectively, an unbiased estimator for $d^2_{\mathrm{MMD}}(P, Q)$ is given by,

$$\begin{aligned}\hat{\mathrm{dist}}^2_{\mathrm{MMD}}(X, Y) = &\frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{m} k(\mathbf{x}_i, \mathbf{x}_j) \\ &+ \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} k(\mathbf{y}_i, \mathbf{y}_j) \\ &- \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} k(\mathbf{x}_i, \mathbf{y}_j). \qquad (4)\end{aligned}$$