

Deep learning

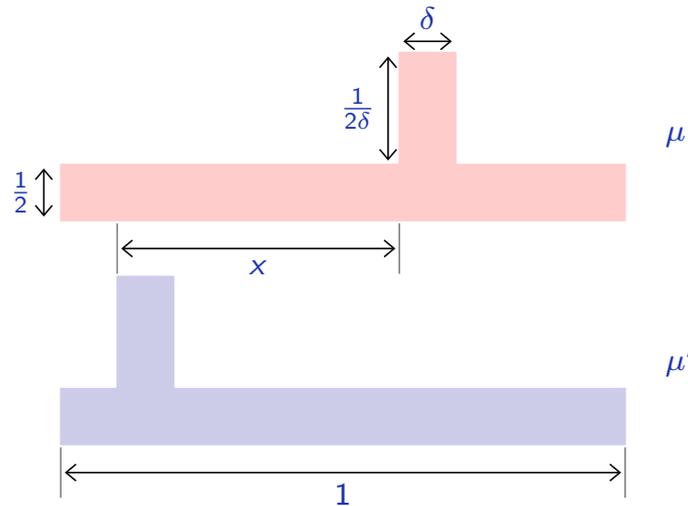
11.2. Wasserstein GAN

François Fleuret
<https://fleuret.org/dlc/>



Arjovsky et al. (2017) pointed out that \mathbb{D}_{JS} does not account [much] for the metric structure of the space.

E.g. in the following example:



We can show that $\mathbb{D}_{\text{JS}}(\mu, \mu') \propto \min(\delta, |x|)$, hence all $x \notin [-\delta, \delta]$ are “as good”.

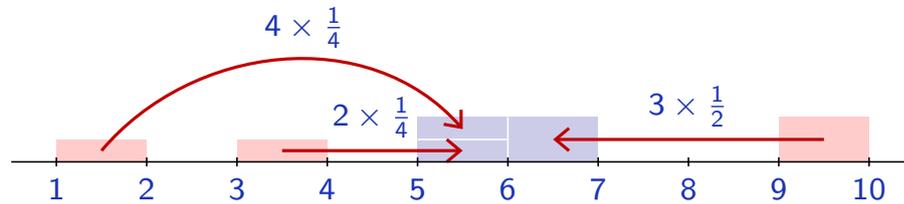
Notes

This example illustrates that the Jensen-Shannon divergence poorly accounts for the metric of the space.

We consider two distributions μ and μ' on $[0, 1]$ that have half of their mass uniformly spanned over the full interval, and half added uniformly over another interval of width δ . These two small intervals are located at two different locations, separated by x .

The Jensen-Shannon divergence is proportional to $\min(\delta, |x|)$, which means that as soon as the two “peaks” no longer overlap (i.e. $x > \delta$), the divergence does not depend on x anymore. Hence, for instance in an optimization context, this divergence would not inform that we are making the distributions similar when x is decreased.

An alternative choice is the “earth moving distance”, or **Wasserstein** distance, which intuitively is the minimum mass displacement to transform one distribution into the other.



$$\mu = \frac{1}{4} \mathbf{1}_{[1,2]} + \frac{1}{4} \mathbf{1}_{[3,4]} + \frac{1}{2} \mathbf{1}_{[9,10]} \quad \mu' = \frac{1}{2} \mathbf{1}_{[5,7]}$$

$$W(\mu, \mu') = 4 \times \frac{1}{4} + 2 \times \frac{1}{4} + 3 \times \frac{1}{2} = 3$$

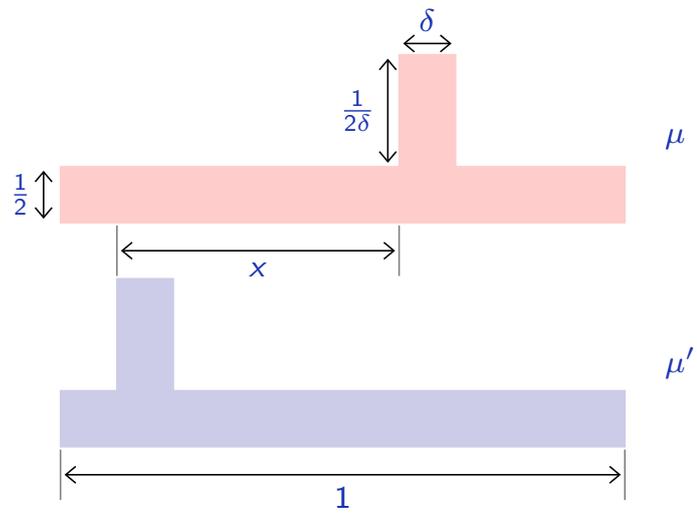
Notes

In the notation here,

$$\forall a < b, \forall x, \mathbf{1}_{[a,b]}(x) = \begin{cases} 1 & \text{if } a \leq x \leq b, \\ 0 & \text{otherwise.} \end{cases}$$

The distribution μ (red) consists of a mixture of three components uniform over segments, which distribution μ' (blue) is uniform over a segment. The optimal way of transforming μ into μ' is depicted with the red arrows. To compute the Wasserstein distance, each displacement distance is weighted by the probability of the “block” to be moved.

Intuitively, it increases monotonically with the distance between modes



$$\mathbb{W}(\mu, \mu') = \frac{1}{2}|x|$$

The Wasserstein distance can be defined as

$$\mathbb{W}(\mu, \mu') = \min_{q \in \Pi(\mu, \mu')} \mathbb{E}_{(X, X') \sim q} [\|X - X'\|],$$

where $\Pi(\mu, \mu')$ is the set of distributions over \mathcal{X}^2 whose marginals are μ and μ' .

So while it would make a lot of sense to look for a generator matching the density for this metric, that is

$$\mathbf{G}^* = \underset{\mathbf{G}}{\operatorname{argmin}} \mathbb{W}(\mu, \mu_{\mathbf{G}}).$$

the definition of \mathbb{W} is not an operational way of estimating it.

A duality theorem from Kantorovich and Rubinstein implies

$$W(\mu, \mu') = \max_{\|f\|_L \leq 1} \mathbb{E}_{X \sim \mu} [f(X)] - \mathbb{E}_{X \sim \mu'} [f(X)]$$

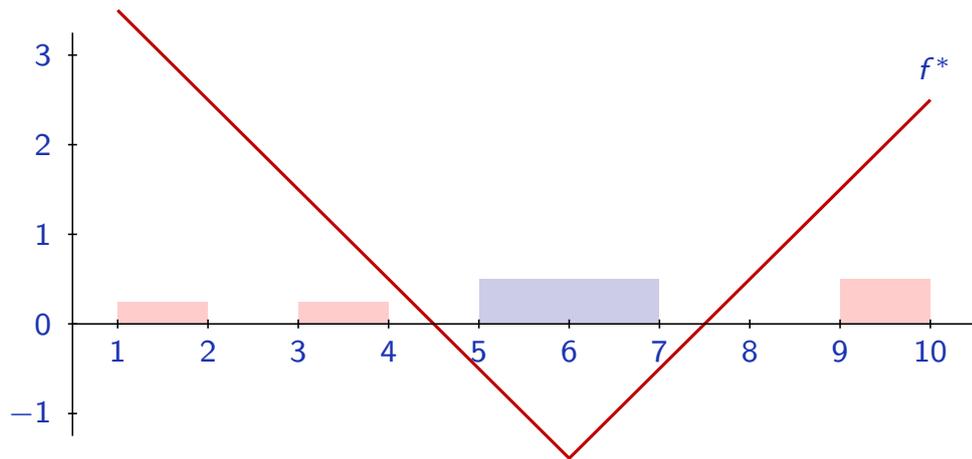
where

$$\|f\|_L = \max_{x, x'} \frac{\|f(x) - f(x')\|}{\|x - x'\|}$$

is the Lipschitz seminorm.

Notes

The Lipschitz seminorm upper-bounds how much a function can change between two points, given the distance between them.



$$\mu = \frac{1}{4} \mathbf{1}_{[1,2]} + \frac{1}{4} \mathbf{1}_{[3,4]} + \frac{1}{2} \mathbf{1}_{[9,10]} \quad \mu' = \frac{1}{2} \mathbf{1}_{[5,7]}$$

$$\mathbb{W}(\mu, \mu') = \underbrace{\left(3 \times \frac{1}{4} + 1 \times \frac{1}{4} + 2 \times \frac{1}{2} \right)}_{\mathbb{E}_{X \sim \mu}[f^*(X)]} - \underbrace{\left(-1 \times \frac{1}{2} - 1 \times \frac{1}{2} \right)}_{\mathbb{E}_{X \sim \mu'}[f^*(X)]} = 3$$

Notes

Here

$$f^* = \operatorname{argmax}_{\|f\|_L \leq 1} \mathbb{E}_{X \sim \mu}[f(X)] - \mathbb{E}_{X \sim \mu'}[f(X)].$$

Using this result, we are looking for a generator

$$\begin{aligned} \mathbf{G}^* &= \underset{\mathbf{G}}{\operatorname{argmin}} \mathbb{W}(\mu, \mu_{\mathbf{G}}) \\ &= \underset{\mathbf{G}}{\operatorname{argmin}} \max_{\|\mathbf{D}\|_L \leq 1} \left(\mathbb{E}_{X \sim \mu} [\mathbf{D}(X)] - \mathbb{E}_{X \sim \mu_{\mathbf{G}}} [\mathbf{D}(X)] \right), \end{aligned}$$

where the \max is now an optimized predictor.

This is very similar to the original GAN formulation, except that the value of \mathbf{D} is not interpreted through a \log -loss, and there is a strong regularization on \mathbf{D} .

The main issue in this formulation is to optimize the network \mathbf{D} under a constraint on its Lipschitz seminorm

$$\|\mathbf{D}\|_L \leq 1.$$

Arjovsky et al. achieve this by clipping \mathbf{D} 's weights.

The two main benefits observed by Arjovsky et al. are

- A greater stability of the learning process, both in principle and in their experiments: they do not witness “mode collapse”.
- A greater interpretability of the loss, which is a better indicator of the quality of the samples.

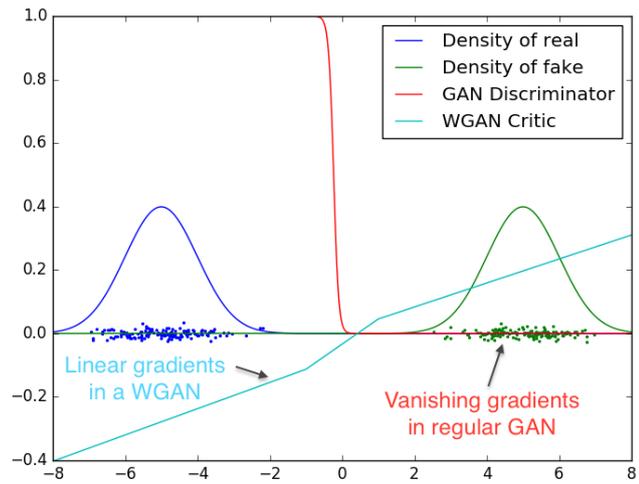


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the traditional GAN discriminator saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

(Arjovsky et al., 2017)

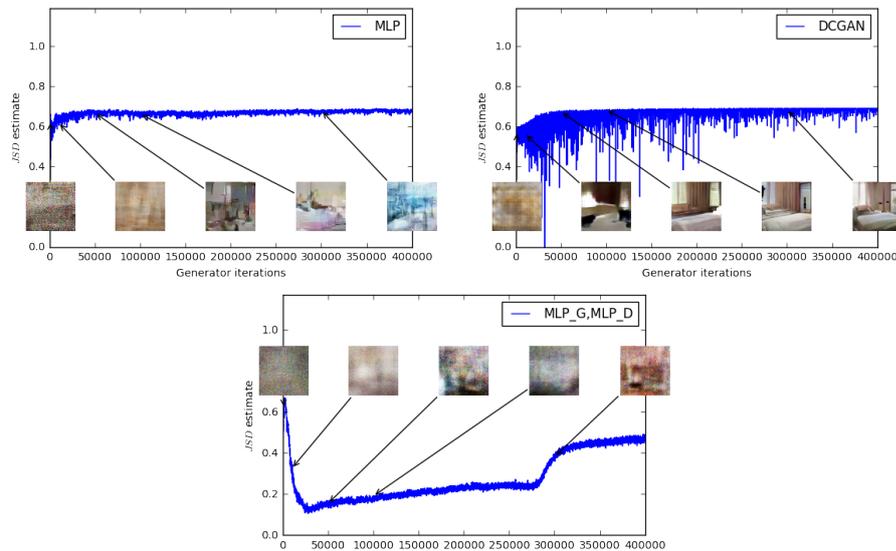


Figure 4: JS estimates for an MLP generator (upper left) and a DCGAN generator (upper right) trained with the standard GAN procedure. Both had a DCGAN discriminator. Both curves have increasing error. Samples get better for the DCGAN but the JS estimate increases or stays constant, pointing towards no significant correlation between sample quality and loss. Bottom: MLP with both generator and discriminator. The curve goes up and down regardless of sample quality. All training curves were passed through the same median filter as in Figure 3.

(Arjovsky et al., 2017)

Notes

With standard GANs, the loss is a poor indicator of the quality of the samples: in the first two cases (top), the loss starts to plateau, although the samples are improving a lot. In the second case, the loss first goes down, then increases, and has a strong increase followed by a plateau.

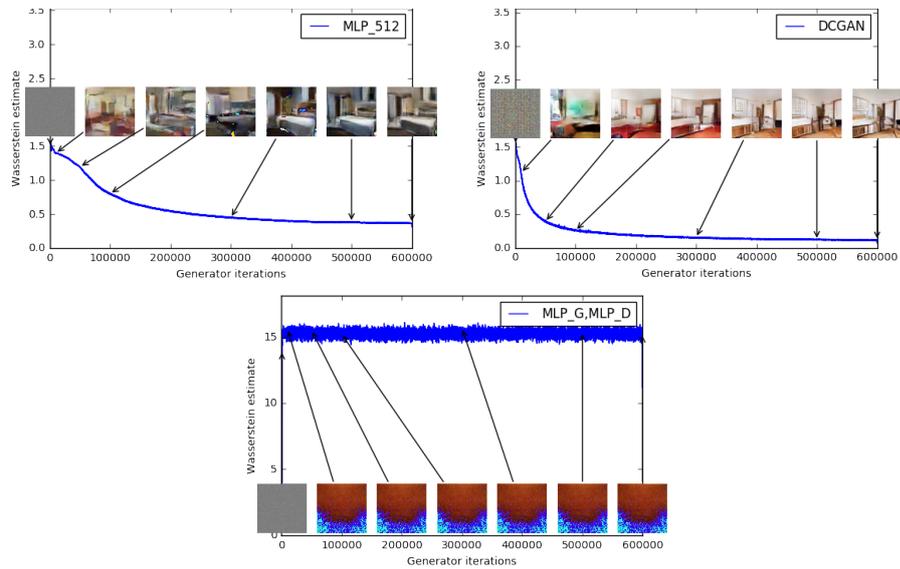


Figure 3: Training curves and samples at different stages of training. We can see a clear correlation between lower error and better sample quality. Upper left: the generator is an MLP with 4 hidden layers and 512 units at each layer. The loss decreases consistently as training progresses and sample quality increases. Upper right: the generator is a standard DCGAN. The loss decreases quickly and sample quality increases as well. In both upper plots the critic is a DCGAN without the sigmoid so losses can be subjected to comparison. Lower half: both the generator and the discriminator are MLPs with substantially high learning rates (so training failed). Loss is constant and samples are constant as well. The training curves were passed through a median filter for visualization purposes.

(Arjovsky et al., 2017)

Notes

With Wasserstein GANs, the loss is a better indicator of the quality of the samples: as the loss goes down, the samples are indeed improving.

However, as Arjovsky et al. wrote:

“Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. If the clipping parameter is large, then it can take a long time for any weights to reach their limit, thereby making it harder to train the critic till optimality. If the clipping is small, this can easily lead to vanishing gradients when the number of layers is big, or batch normalization is not used (such as in RNNs).”

(Arjovsky et al., 2017)

In some way, the resulting Wasserstein GAN (WGAN) trades the difficulty to train **G** for the difficulty to train **D**.

In practice, this weakness results in extremely long convergence times.

Notes

In the original GAN, no constraint is imposed on **D**, which can easily be optimized to discriminate real from fake images.

This makes **G** hard to train because the response of **D**, which is very confident, and the resulting gradient of the loss is therefore very small, and consequently the gradient w.r.t. **G**'s parameters is also very small.

With Wasserstein GAN, due to the constrain on the discriminator, it does not saturate and there is always a gradient flowing back to the generator. However now the discriminator is harder to train since the gradient w.r.t. its parameters has to be clipped or projected in some way and may be set to zero.

In some way the Wasserstein GAN trades the difficulty to optimize the generator for the difficulty to train the [regularized] discriminator.

Spectral Normalization

Miyato et al. (2018) proposed to control the Lipschitz seminorm of \mathbf{D} by rescaling its weights, so that all the linear layers have their singular values lesser than 1, and consequently Lipschitz seminorm, lesser than 1.

Since the Lipschitz seminorm of a composition is [upper-bounded by] the product of the seminorms, if the non-linear layers are also Lipschitz of constant lesser than 1 (e.g. ReLU), this is a sufficient condition.

Spectral Normalization is a layer normalization that estimates the largest singular value of a weight matrix, and rescale it accordingly.

While computing the SVD of a matrix is expensive, computing [a good approximation of] the largest SV can be done iteratively for a reasonable cost.

Notes

Spectral normalization addresses the control of the Lipschitz seminorm in a way which is less brutal than the weight clipping.

Regarding the Lipschitz seminorm of a composition: if we consider a function g with a Lipschitz constant of L_f , and f with a Lipschitz constant of L_g , then by definition of the Lipschitz constant we have for any x and y on the domain of f

$$\frac{f(x) - f(y)}{x - y} \leq L_f.$$

Then

$$\begin{aligned} & \frac{g \circ f(x) - g \circ f(y)}{x - y} \\ &= \frac{g \circ f(x) - g \circ f(y)}{f(x) - f(y)} \frac{f(x) - f(y)}{x - y} \\ &\leq L_g L_f \end{aligned}$$

So, in particular, if both f and g have a Lipschitz constant smaller than 1, then $g \circ f$ also have a Lipschitz constant smaller than 1.

The largest singular value of a matrix W is also its spectral norm

$$\sigma(W) = \max_{h: \|h\|_2 \leq 1} \|Wh\|_2.$$

To calculate it, the **power iteration** method starts with a random vector u_0 and iterates

$$v_{n+1} = \frac{W^T u_n}{\|W^T u_n\|_2}$$
$$u_{n+1} = \frac{W v_{n+1}}{\|W v_{n+1}\|_2}$$

that gives

$$\sigma(W) = \lim_{n \rightarrow \infty} u_n^T W v_n.$$

```
W = torch.randn(15, 15)
print(W.svd().S.max())

u = torch.randn(W.size(0))

for k in range(10):
    v = W.t() @ u
    v = v / v.norm()
    u = W @ v
    u = u / u.norm()

print(u.t() @ W @ v)
```

prints

```
tensor(7.9129)
tensor(7.9129)
```

Notes

This example illustrates that the power iteration methods can approximate the largest singular value of a matrix.

The largest singular value is computed with `W.svd().S.max()`.

Miyato et al. update u_n and v_n before every gradient step, and rescale the weight matrix accordingly.

The same can be done in PyTorch with `torch.nn.utils.spectral_norm`, that wraps any linear layer into a module that performs the normalization.

```
m = nn.Linear(5, 5)
print(m.weight.svd().S.max())

x = torch.rand(100, 5)
optimizer = torch.optim.SGD(m.parameters(), lr = 1e-1)

for k in range(100):
    loss = -m(x).norm()
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

print(m.weight.svd().S.max())
```

prints

```
tensor(0.9277, grad_fn=<MaxBackward1>)
tensor(114.1429, grad_fn=<MaxBackward1>)
```

Notes

Here, we consider a simple linear module which we optimize to maximize its norm, without normalization. As expected we get a large norm in the end. In the next slide, the module is wrapped into a [spectral_norm](#).

```
m = nn.Linear(5, 5)
print(m.weight.svd().S.max())

m = nn.utils.spectral_norm(m)

x = torch.rand(100, 5)
optimizer = torch.optim.SGD(m.parameters(), lr = 1e-1)

for k in range(100):
    loss = -m(x).norm()
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

print(m.weight.svd().S.max())
```

prints

```
tensor(0.8716, grad_fn=<MaxBackward1>)
tensor(1.0000, grad_fn=<MaxBackward1>)
```

References

- M. Arjovsky, S. Chintala, and L. Bottou. **Wasserstein GAN**. CoRR, abs/1701.07875, 2017.
- T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. **Spectral normalization for generative adversarial networks**. In International Conference on Learning Representations (ICLR), 2018.