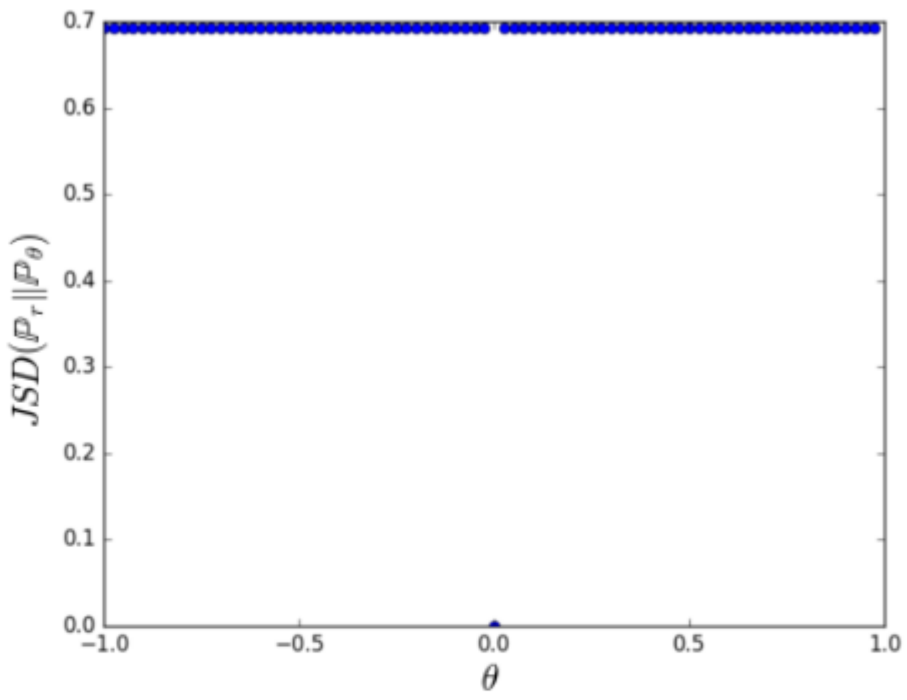
Wasserstein GAN

- a.k.a. WGAN
- Authors: M. Arjovsky (maths guy), S. Chintala and L. Bottou (Facebook AI guys)
- Arxiv: http://arxiv.org/abs/1701.07875
- Code: https://github.com/martinarjovsky/WassersteinGAN
- Reddit discussion, featuring comments from authors, Goodfellow, and other researchers: https://www.reddit.com/r/MachineLearning/comments/5qxoaz/r_170107875_wasserstein_gan/

A problem with existing GANs

- To my knowledge all existing GAN variants minimise the f-divergence between the real data distribution $P_r(x)$ and the generated data distribution $P_g(x)$
    - The usual GAN objective turns out to be very similar to Jenson-Shannon (JS) divergence, though the f-GAN paper explains how to use any f-divergence you like
- f-divergence is a function of the density ratio $\frac{P_r(x)}{P_g(x)}$
    - But what if the supports of the two distributions don't overlap significantly? The density ratio will be infinite or zero everywhere they don't overlap! 😲
    - As long as the supports are disjoint, the f-divergence will be constant since the density ratio is constant
- Simple example:
    - The real data consists of $(0, z)$ points where $z \sim U(0, 1)$
        - Samples are uniformly distributed along a vertical line at x = 0 from y = 0 to y = 1
    - The model has one parameter θ such that it produces samples (θ, z)
    - Either the distributions match perfectly or do not overlap at all

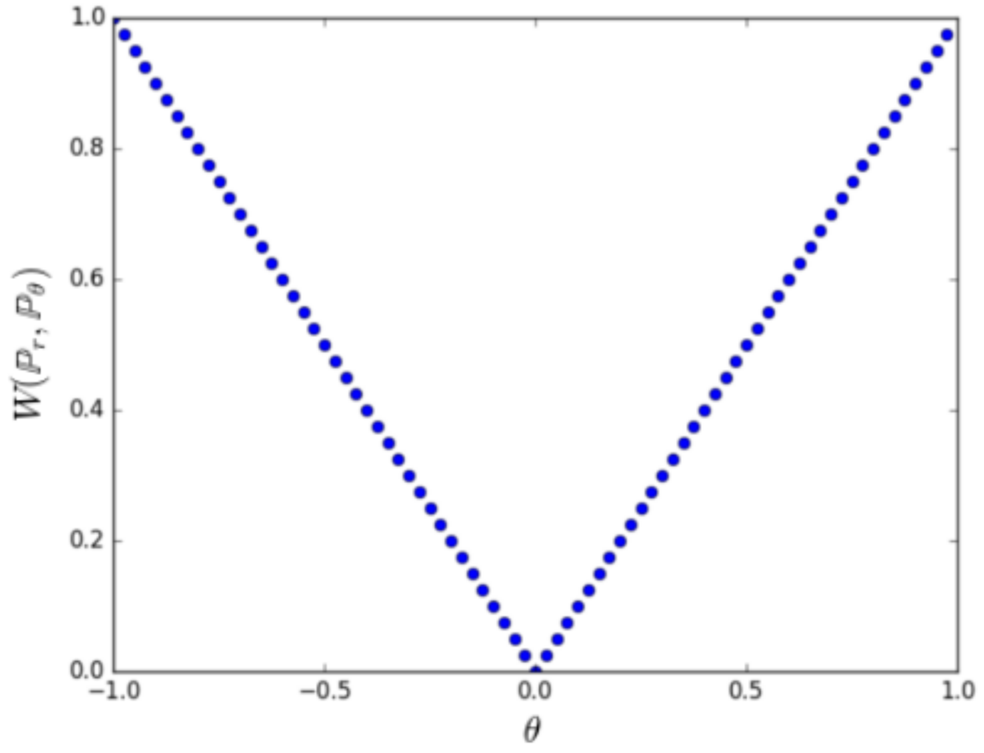


- ◦ The above graph shows the JS divergence for different values of θ
    - ▪ The graph is mostly flat
    - ▪ This means that the gradient of the divergence w.r.t. θ is zero almost everywhere
    - ▪ If the discriminator learns to output a highly accurate approximation of the JS divergence, the generator gets approximately zero gradient
        - This is an instance of the "vanishing gradient" problem found in GANs
- The problem of non-overlapping supports has been identified before
    - ◦ For example, Ferenc has a blog post about it (http://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/), which is based on the Instance Noise paper
    - ◦ Instance Noise isn't a very satisfying solution (it just adds noise to the inputs and says that the supports now overlap)

Earth-mover distance

- a.k.a. EM distance or Wasserstein-1 distance
- An alternative to f-divergence which is not a function of the density ratio
- If you think of the probability distributions as mounds of dirt, the EM distance describes how much effort it takes to transform one mound of dirt so it is the same as the other using an optimal transport plan

    - Accounts for both mass and distance
- If the supports of the distributions don't overlap, the EM distance will describe how far apart they are
- For the simple example described earlier:



    - Note that we now have gradients that always point towards the optimal θ!
- EM distance is defined as $W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} ||x - y||$
    - Notation: think of the infimum as a minimum
    - Considers all possible "configurations" of pairing up points from the two distributions
    - Calculates the mean distance of pairs in each configuration
    - Returns the smallest mean distance across all of the configurations
    - Intractable, can't compute directly 😞
- Fortunately there is an alternative definition
    - $K \cdot W(P_r, P_g) = \sup_{||f||_L \le K} (\mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_g}[f(x)])$
    - Result of the Kantorovich-Rubinstein duality
    - Notation: think of the supremum as a maximum
    - The supremum is over all K-Lipschitz functions (more on this later)
    - Intuitively we are finding some function with greatest margin between the mean value over real examples and the mean value over generated examples

What the Lipschitz?

- So, in order to use the EM distance we need to approximate the maximal $f(x)$
- $f(x)$ is K-Lipschitz, which means the magnitude of the gradient is upper bounded by K everywhere
- We can force a network to model only K-Lipschitz functions by clamping the weights to a fixed box
- Therefore instead of a discriminator we use a *critic* which models $f(x)$

Implementing WGAN

- Actually really simple in practice 🎉
- We refer to the discriminator as a critic now
- Repeatedly clamp weights in the critic during training, the range [-0.01, 0.01] for each individual weight works well

```
disc_params:clamp(-0.01, 0.01)
```

- No sigmoid on the output of the critic, we don’t want to constrain the output range
- Training the critic
    - For real examples, push the mean output for the batch upwards

```
local target = output + 1 -- We want the output to be bigger
abs_criterion:forward(output, target)
local dloss_dout = abs_criterion:backward(output, target)
```

    - For generated examples, push the mean output for the batch downwards

```
local target = output - 1 -- We want the output to be smaller
abs_criterion:forward(output, target)
local dloss_dout = abs_criterion:backward(output, target)
```

- The critic should undergo multiple training iterations for every generator update
    - Since we no longer suffer from the same type of vanishing gradient problem as other GANs, we want the critic to be as close to optimal as possible at all times to best approximate the EM distance
- Training the generator
    - Same as normal GANs - calculate gradients at critic input as if the samples were real, update generator weights

Claimed benefits of WGAN

- Stable training
  - Gradients of EM distance are much better behaved
- Works on a wide variety of architectures
  - They even trained an MLP GAN
- The plot of $\mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_g}[f(x)]$ converges during training
  - Finally we can figure out when to stop training GANs
  - Corresponds to sample quality
- Avoids the mode collapse problem
  - This seems to be more of an anecdotal observation

My experiences and thoughts

- Easy to implement (I've done so in Torch and tried it on CIFAR-10, SVHN, and CelebA)
- Slower to train than regular GANs
  - Epochs are longer due to training the critic more
  - Requires more epochs for samples to look nice
- The distance curve generally converges, but is not immune from increasing for a few epochs before dropping again
- Makes it harder to do tricksy things with the critic (especially semi-supervised learning)
  - Unclear how to output a class prediction since output is no longer real/fake
  - The critic network itself is a bit weird due to the clamping, so it is unclear how suitable it is for multi-task type learning or even feature extraction
    - My initial attempts at multi-task for semi-supervised learning have failed