Wasserstein GAN: Motivation

paper (https://arxiv.org/pdf/1701.07875.pdf)

To summarize what we have learned about standard GANs:

- Adversarial Training minimizes the Jensen Shannon Distance between p_{model} and p_{data}

But GANs trained using Adversarial Training have the reputation of being difficulty to train

- A Discriminator that is too good, too soon inhibits the ability of the Generator to learn to generate
- The Generator may "mode collapse" and not produce a variety of outputs

The *Wasserstein GAN (WGAN)* is a Generator/Discriminator pair

- that is trained to minimize
- an *approximation* of
$$\mathbb{W}(p_{\text{data}}, p_{\text{model}})$$

where $\mathbb{W}$ is the *Wasserstein Distance*, also know as the *Earth Move Distance (EMD)* measure.

Aside

Technically: the Discriminator is a "critic"

- rather than producing a probability of "Real"
- it produces a "score" measuring how real the input is
    - score is not limited to range of probability: $[0, 1]$
    - larger negative: more real
    - larger positive: less real

Earth Move Distance (EMD)

Aside

You need some knowledge of Measure Theory to understand the math.

In the absence, there are two good blogs I recommend in order to get a flavor

- Sorta Insightful (https://www.alexirpan.com/2017/02/22/wasserstein-gan.html)
- Wen (https://arxiv.org/pdf/1904.08994.pdf)

Like the KL and Jensen-Shannon Distances, the EMD is a measure of the difference between two distributions. p_{data} and p_{model}.

It has an intuitive explanation

> *The minimum amount of "work" involved in moving probability mass between the two distributions in order to make them identical*

"Work" means: the product of

- the quantity $\gamma(x, y)$ of the mass moved from x to y
- and the distance $\|x - y\|$ it is moved

We can easily illustrate with two discrete distributions (example from Wen (https://arxiv.org/pdf/1904.08994.pdf))

Let P, Q be the two distributions

- represented as frequency vectors
- position i in the vector is the *frequency* of element i occurring

(Since the number of elements of P and Q is the same, frequency is just probability scaled)

$$\begin{aligned} P &= [3, 2, 1, 4] \\ Q &= [1, 2, 4, 3] \end{aligned}$$

For illustration, we will move

- some frequency δ_i from P_i
- to Q_i
- such that the resulting $P_i' = P_i - \delta_i = Q_i$

Thus

$$\delta_i = P_i - Q_i$$

is moved from P_i to Q_i

This can be seen as moving

- "probability mass" (frequency)
- a distance of 1

But once we move $(P_i - Q_i)$ into Q_i

- the resulting $Q'_i = Q_i + (P_i - Q_i)$ is no longer equal to the new P_i (i.e, the old Q_i)
    - δ_i was chosen to make Q'_i equal to the *original* P_i
    - but the new P'_i is not $P'_i = P_i - \delta_i$

So we have to move some probability from Q_i to Q_{i+1}

We can define δ_i, the quantity of probability moved at each step, recursively:

$$\begin{aligned} \delta_0 &= 0 \\ \delta_{i+1} &= \delta_i + P_i - Q_i \end{aligned}$$

That is, the amount δ_{i+1} moved from P_i in order to make $P_i = Q_i$ is

- the difference $(P_i - Q_i)$ between original value of P_i and Q_i
- plus the additional quantity δ_i that was moved into P_i from the previous step

We illustrate on the example above.

Note

- the P_i, Q_i in the equations below refer to the *original* values

$$
\begin{aligned}
P &= [3,2,1,4] \\
Q &= [1,2,4,3]
\end{aligned}
$$

$$
\begin{aligned}
\delta_0 &= 0 \\
\delta_1 &= 0 + (3 - 1) = 2 \qquad \delta_1 = \delta_0 + (P_1 - Q_1) = 3 - 1 \\
\delta_2 &= 2 + (2 - 2) = 2 \\
\delta_3 &= 2 + (1 - 4) = -1 \\
\delta_4 &= -1 + (4 - 3) = 0
\end{aligned}
$$

Work is positive so taking absolute values

$$\mathbb{W}(P, Q) = \sum_{i=1}^{4} 1 * |\delta_i| = 5$$

where 1 is the distance

- one position in each turn of of our illustration

For continuous distributions

$$\mathbb{W}(p_{\text{data}}, p_{\text{model}}) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_{\text{model}})} \mathbb{E}_{(x,y)\sim\gamma} \|x - y\|$$

where

- $\Pi(p_{\text{data}}, p_{\text{model}})$ are the set of possible joint distributions with marginal p_{data} and p_{model}
- γ is a quantity to move from x to y (for all x, y)
    - distance between x and y is $\|x - y\|$
- $\inf$ is the infimum (Greatest Lower Bound)

Approximation of $\mathbb{W}(p_{\text{data}}, p_{\text{model}})$

Warning: the math is stated without much explanation

The infimum is intractable (or at least: not practical to compute).

Equation 2 in the paper states that for certain functions f, the distance is also equal to

$$\mathbb{W}(p_{\text{data}}, p_{\text{model}}) = \inf_{\|f\|_2 \leq 1} \mathbb{E}_{x \sim p_{\text{data}}} f(x) \; - \; \mathbb{E}_{x \sim p_{\text{model}}} f(x)$$

One can look at f as a "score" of x being "Real" (not fake) where

- a high negative score is a highly confident "Real"
- a high positive score is a highly confident "Fake"

The goal is

- for function f to create a *large spread* between scores of Real and Fake.
- for function f to be *approximated* by the Discriminator D_{Θ} with weights Θ_D

Under certain conditions on f, finding $\mathbb{W}$ is equivalent to solving

$$\max_{\Theta_D \in \mathcal{W}} \mathbb{E}_{x \sim p_{\text{data}}} D_{\Theta_D}(x) \; - \; \mathbb{E}_{x \sim p_{\text{model}}} D_{\Theta_D}(x)$$

where $\mathcal{W}$ is a "compact" space of possible weights

Certain conditions

f lies in the space of 1-Lipschitz functions

That is

- we solve for Discriminator (Critic) weights
- such that the scores it produces
- have a large spread between Real and Fake.

This is called a *Contrastive* objective.

But: what does this mean ?

For those (like me) struggling with the math, here are the implications from a practical perspective

- Scores for true Real is negative, for Fake is positive
- $\mathcal{L}_G$ will implement: minimize (make most negative) the score assigned to Fakes
- $\mathcal{L}_D$ will implement: "maximize **the spread** of scores between Real and Fake"
    - by minimizing the sum of
        - sum of scores for Real examples
        - minus sum of scores for Fake examples
            - i.e., Discriminator goal is for Fakes to have **positive** scores
            - in contrast to Generator goal to have Fakes have **negative** scores

- "Compact" Θ_D will be achieved by clipping
    - restricting elements of Θ_D to a small numerical range
    - by clipping the weights after a gradient update step for D

- $\mathcal{L}_D$ will dispense with the $\log$ since the Discriminator produces scores rather than probabilities
    - we see terms $D(\mathbf{x^{(i)}})$ and $1 - D(\mathbf{x^{(i)}})$
    - rather than $\log D(\mathbf{x^{(i)}})$ and $1 - \log D(\mathbf{x^{(i)}})$

When we visit the code, we will see these elements in practice.

Did I really need to change to EMD ?

The Wasserstein GAN avoids many of the problems associated with the plain GAN.

To some extent, this is due to replacing the Discriminator with a Critic

- unbounded scores in the WGAN versus bounded probabilities in the plain GAN

- There are mathematical problems with Expectation Maximization (KL distance) and Jensen-Shannon (JS)distance
    - the terms $\log(p_{\text{model}}(\mathbf{x}))$ and $\log(p_{\text{data}}(\mathbf{x}))$ appear
    - if p_{data} and p_{model} don't completely overlap (a possibility especially early in training)
        - we take logs of 0, which is infinite (negative)
    - No such problem with EMD

- No vanishing gradients with EMD
    - with KL and JS distance the true derivative goes to 0
    - no such problem with EMD
        - The Critic's scores are not bounded, so can't saturate
    - Thus: we can train the Discriminator to convergence immediately
        - No danger of being too good too soon
        - When we see code we will observe
            - The number of steps of Discriminator update is a multiple of the number of steps of Generator update
- No mode collapse with EMD
    - with a fixed Discriminator (classifier), the Generator in a plain GAN will seek out examples with highest probability of being mis-classified as Real

Training code for a simple GAN: Highlights

The Keras examples (https://keras.io/examples/generative/wgan_gp/#create-the-wgangp-model#wasserstein-gan-wgan-with-gradient-penalty-gp) include a method called *Gradient Penalty* to circumvent the requirement that the Discriminator lie within the space of 1-Lipschitz functions

```
WGAN requires that the discriminator (aka the critic) lie within the space of 1
-Lipschitz functions. The authors proposed the idea of weight clipping to achie
ve this constraint. Though weight clipping works, it can be a problematic way t
o enforce 1-Lipschitz constraint and can cause undesirable behavior, e.g. a ver
y deep WGAN discriminator (critic) often fails to converge.

The WGAN-GP method proposes an alternative to weight clipping to ensure smooth
training. Instead of clipping the weights, the authors proposed a "gradient pen
alty" by adding a loss term that keeps the L2 norm of the discriminator gradien
ts close to 1.
```

The *gradient penalty* is the most interesting part of the code

- Gradient used as a *term in the Loss*
- rather than as a means to update weights

See our module on Advanced Keras (Keras_Advanced.ipynb#Wasserstein-GAN-with-Gradient-Penalty)

In [ ]:
```
print("Done")
```