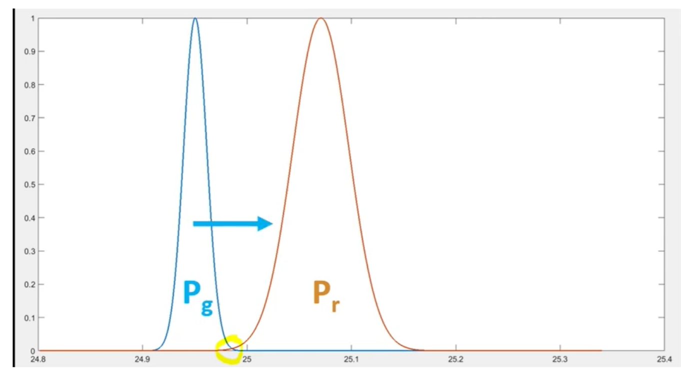
# Day 13: Wasserstein GANs

## Introduction

- It's a modification of the traditional GANs with few changes in the algorithm
- Pros of WGAN
  - Better stability
  - The loss actually means something, unlike the previous GANs
- Cons of WGAN
  - Longer to train
- More stability essentially means - helps prevent mode collapse
- In normal GANs
  - We have two probability distributions Pg & Pr
    - Pg - Distribution that comes from the generator
    - Pr - Distribution from the real images
  - We wanted Pg and Pr to be as close as possible to generate realistic looking images
  - 

  - So if we look at the probability distributions, the goal is to move the Pg towards Pr, or reduc their distance
  - So how do we define this distance
    - KL Divergence
    - JS Divergence
    - Wasserstein Divergence
      - WGAN bases its loss from Wasserstein distance to prevent the gradient issues and unstable training
- In the research paper they do a LOT of math
- So, what do we get in the end of it:

$$\max_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

Constraint on Discriminator

Real data

Generator

  - Basically, Discriminator wants to separate (maximize) this as much as possible
  - And, Generator wants to pull (minimize) this as much as possible
  - Discriminator is now called the critic, because we no longer use the sigmoid function for output
- If you're interested:

Wasserstein GAN

Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

[1]Courant Institute of Mathematical Sciences
[2]Facebook AI Research

1 Introduction

The problem this paper is concerned with is that of unsupervised learning. Mainly, 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 of densities $(P_\theta)_{\theta \in \mathbb{R}^d}$ and finding the one that maximized the likelihood on our data: if we have real data examples $\{x^{(i)}\}_{i=1}^m$, we would solve the problem

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

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

For this to make sense, we need the model density $P_\theta$ to exist. This is not the case in the rather common situation where we are dealing with distributions

- **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:** : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size. $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
**Require:** : $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.

**while** $\theta$ 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**

  - Implementation details: learning rate, batch size, n_critic.
  - When the critics loss is not zero
  - n_critic, it trains the critic more than the generator. Every five steps for critic, one step for generator
  - They use RMSProp instead of Adam
  - Clip weight parameters between [-c,c]
  - Train generator
- 4 Empirical Results

We run experiments on image generation using our Wasserstein-GAN algorithm and show that there are significant practical benefits to using it over the formulation used in standard GANs.

We claim two main benefits:

  - a meaningful loss metric that correlates with the generator's convergence and sample quality
  - improved stability of the optimization process

## Hands-on