Machine Learning Notes - 2

Lipschitz Continuity

https://math.stackexchange.com/questions/2374289/understanding-lipschitz-continuity/2374305#2374305

Think about the mean value theorem and Lipschitz continuity.
Mean value theorem says if f is continuous at [a,b] and differentiable at (a,b), then

$$\exists c \in (a, b) \text{ such that } \frac{f(b) - f(a)}{b - a} = f'(c).$$

Lipschitz says that

$$\exists K > 0, \forall a, b \in D_f, \text{ such that } \frac{|f(b) - f(a)|}{|b - a|} \leq K.$$

Then if the derivative of f as a function is bounded, then f will be **Lipschitz**.

Specifically, if **K == 1**, then the function is said to be **1-Lipschitz**.

Consider the case

$$f(x) = \sqrt{x} \text{ for } x \in [0, 1]$$

One interesting fact to notice from these examples is that if a 1D function is differentiable, then its Lipschitz constant is just the maximum value of its derivative. This gives a slightly more rigorous explanation of why √x is not Lipschitz continuous: its derivative (1 / 2) • (1 / √x) is unbounded as x goes to 0.

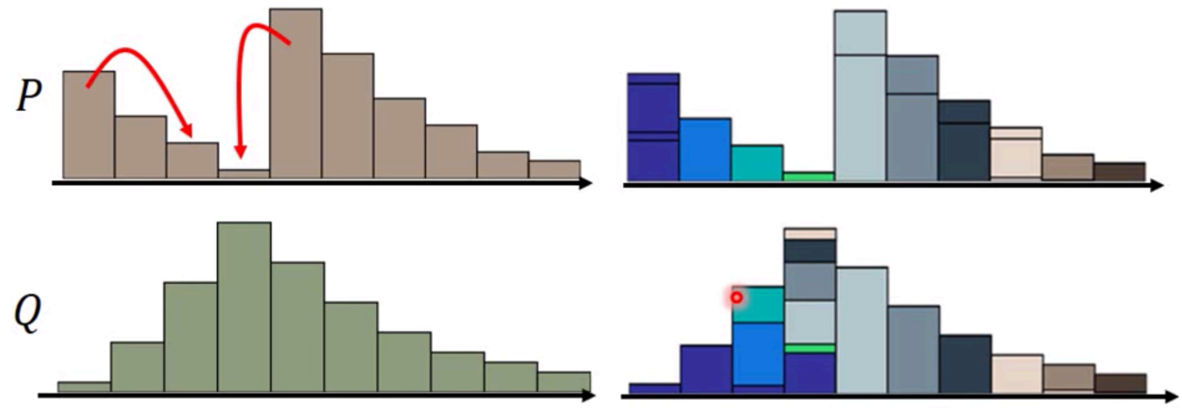
$$\sup_{x\in[0,1]} f'(x) = \lim_{x\to 0} f'(x) = +\infty.$$

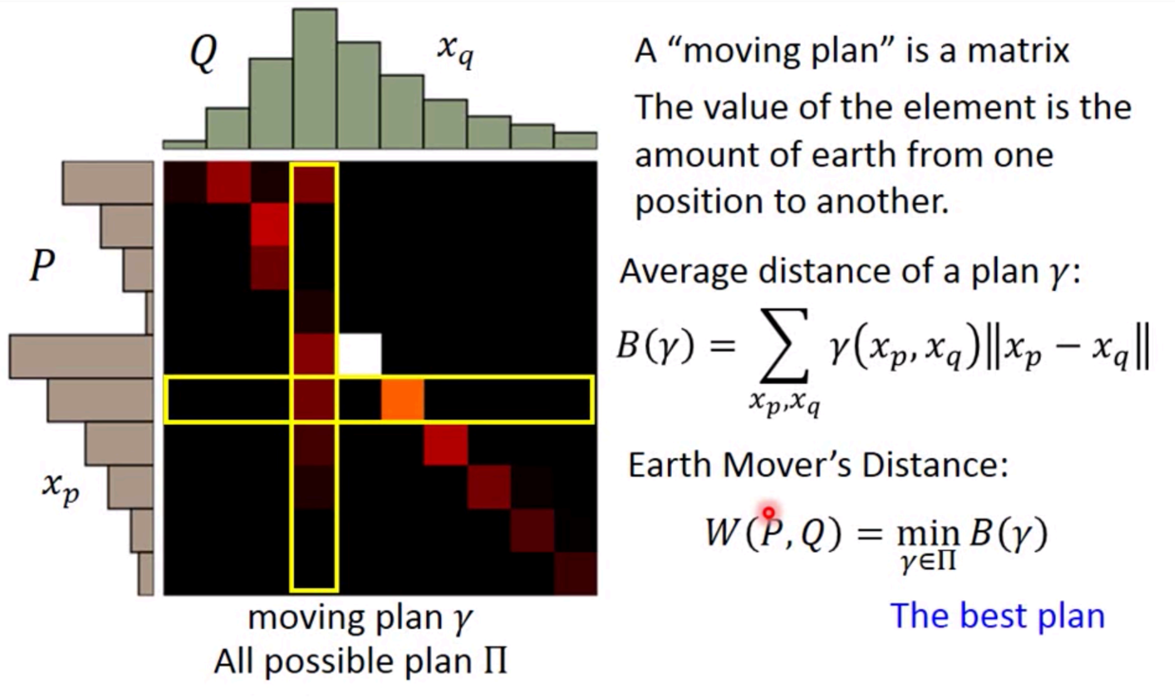
Spectral Normalization

https://christiancosgrove.com/blog/2018/01/04/spectral-normalization-explained.html

Earth Mover's Distance

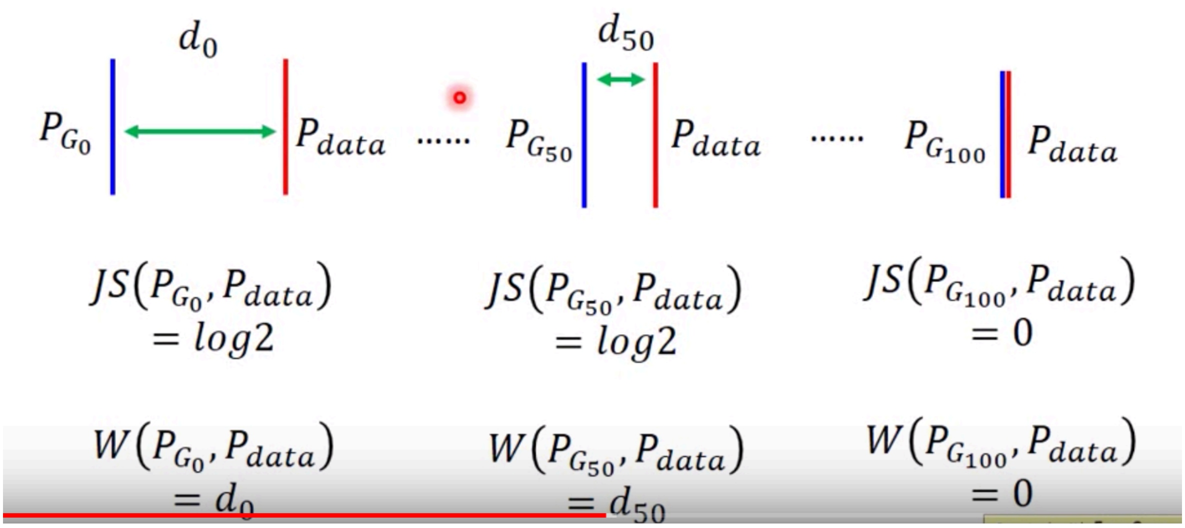
https://vincentherrmann.github.io/blog/wasserstein/
https://www.youtube.com/watch?v=KSN4QYgAtao&feature=youtu.be
Consider one distribution **P as a pile of earth**, and another distribution **Q as the target**,
Earth Mover's Distance is the average distance the earth mover has to move the earth.





Why Earth Mover's Distance ?

the comparation between **JS-Divergence** and **Earth Mover's Distance**
$\mathbf{P_G}$ denotes the distribution of Generator and $\mathbf{P_{DATA}}$ denotes the distribution of real data.
As shown below, **JSD** is less effectiveness to represent the "distance" between $\mathbf{P_G}$ and $\mathbf{P_{DATA}}$ while **Earth Mover's Distance** does.

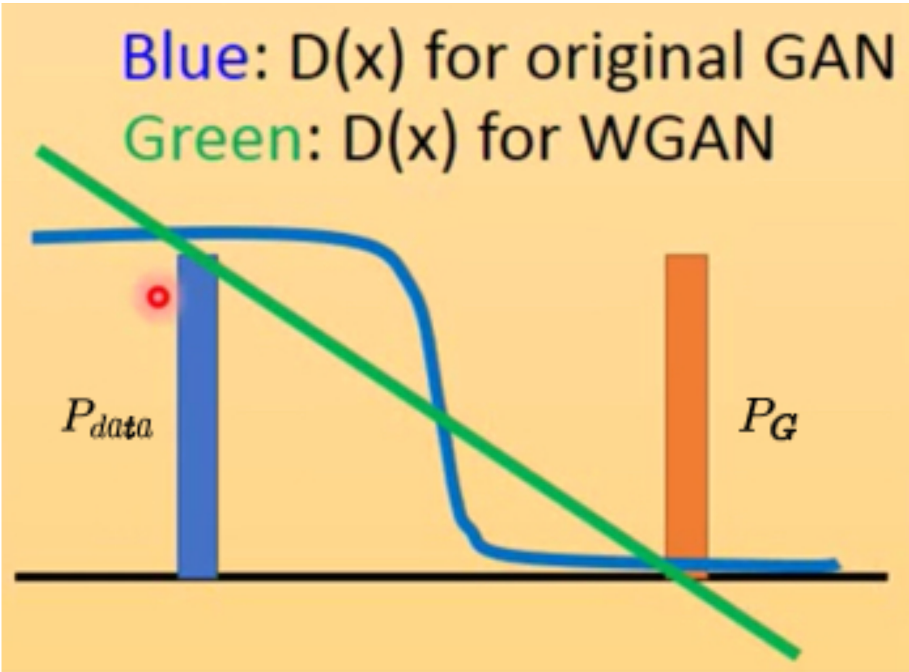


Back to the GAN framework

$$D_f(P_{data}||P_G) \Rightarrow W(P_{data}, P_G)$$

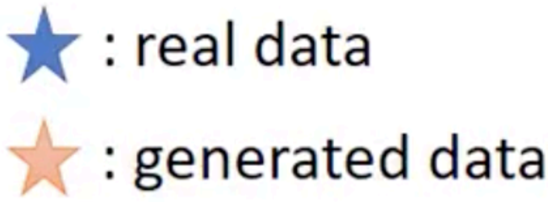
$$= \max_D \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[f^*(D(x))]\}$$

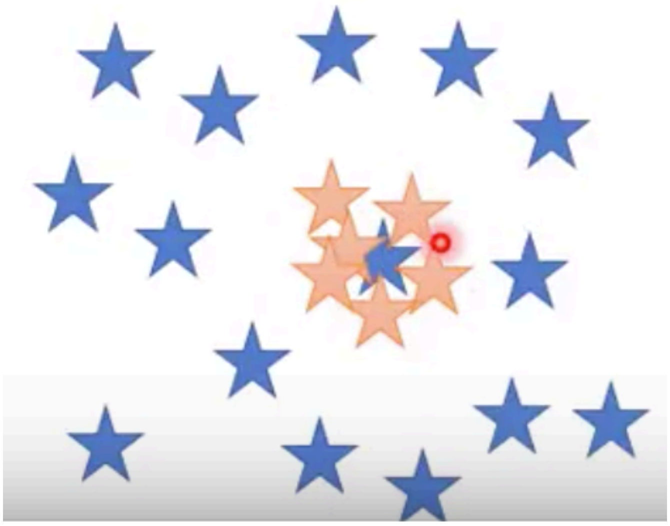
$$W(P_{data}, P_G) = \max_{D\in 1-Lipschitz} \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)]\}$$



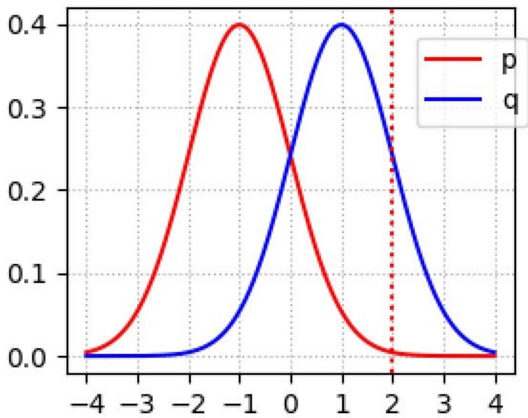
Mode Collapse

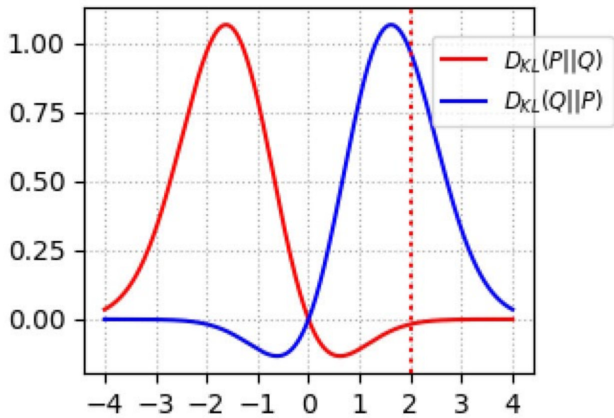
https://www.youtube.com/watch?v=av1bqilLsyQ





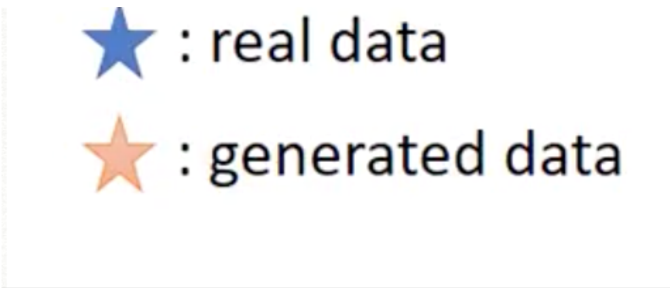
e.g., Reversed KL-Divergence
https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b
The reverse KL-divergence ***LKD(q, p)*** penalizes the generator if the images does not look real: high penalty if *p(x) → 0* but *q(x) > 0*. But it explores less variety: low penalty if *q(x) → 0* but *p(x) > 0*. **(Better quality but less diverse samples)**





Mode Dropping

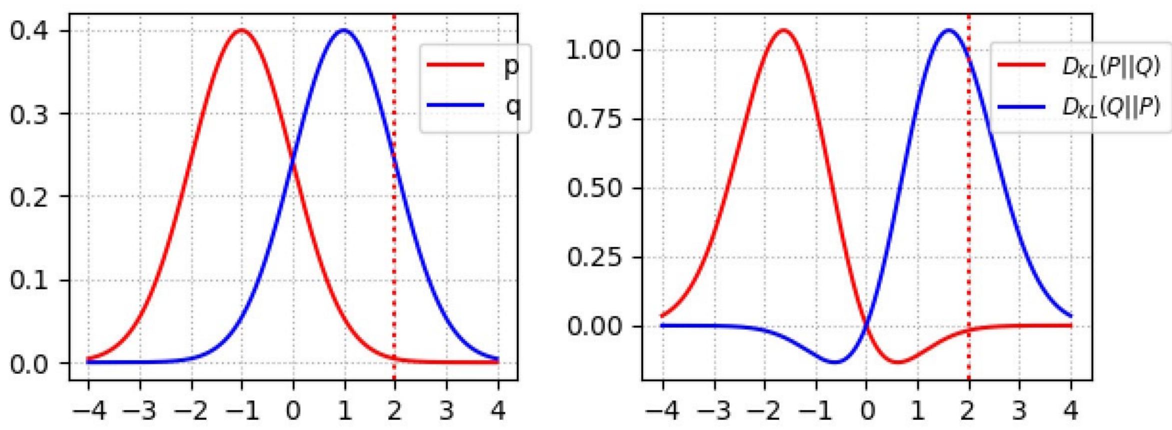
https://www.youtube.com/watch?v=av1bqilLsyQ



e.g., KL-Divergence

https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b

The KL-divergence ***KLD(p, q)*** penalizes the generator if it misses some modes of images: the penalty is high where *p(x) > 0* but *q(x)* → *0*. Nevertheless, it is acceptable that some images do not look real. The penalty is low when *p(x)* → *0* but *q(x)>0*. **(Poorer quality but more diverse samples)**



Some generative models (other than GANs) use MLE (a.k.a KL-divergence) to create models. It was originally believed that KL-divergence causes poorer quality of images (blurry images). But be warned that some empirical experiments may have disputed this claim.

Attention Model

https://www.youtube.com/watch?v=W2rWgXJBZhU

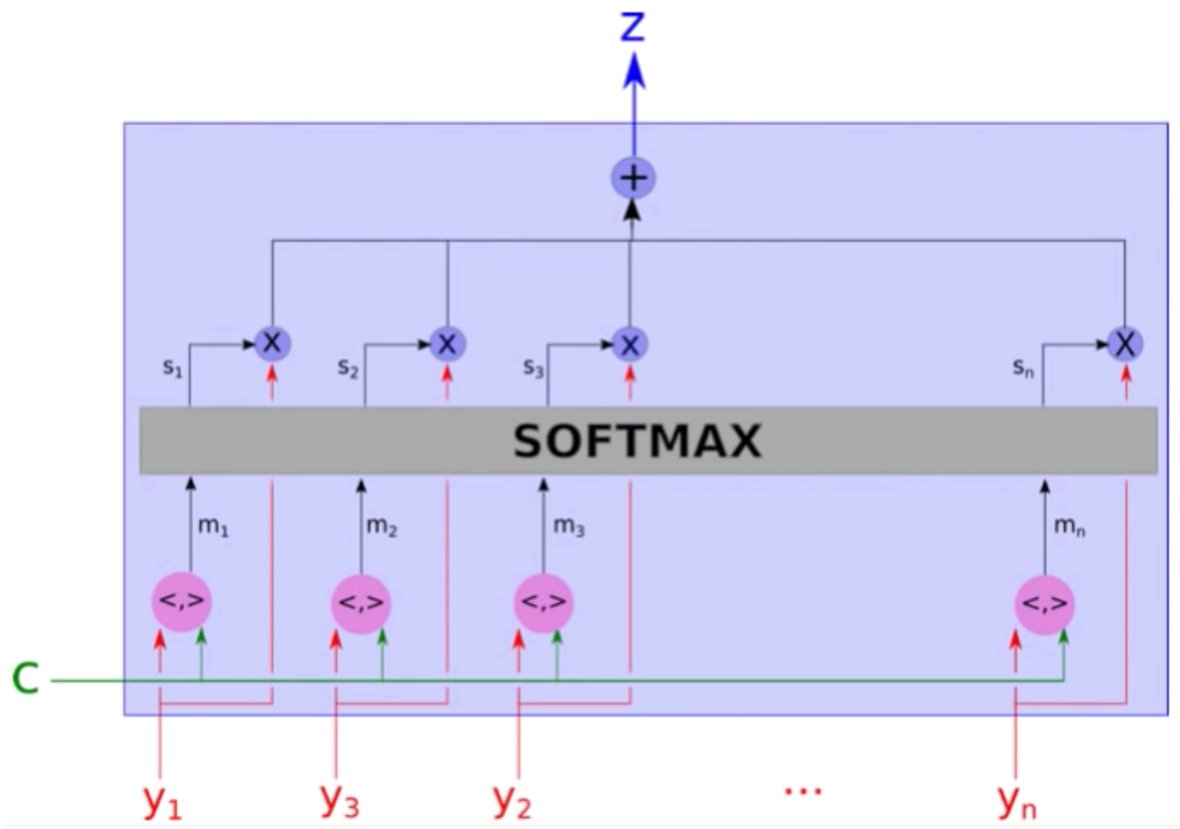
Say you have a bunch of features **m1 ... mn**, and you want to output a single scaler **z** that weighted sum all those features, how do you calculate the weights?
Simple. Transform features **m1 ... mn** into probabilities **s1 ... sn** by taking **m1 ... mn** as the logits of the softmax function.

$$s_i = \frac{e^{m_i}}{\sum_n e^{m_n}}$$

Then weighted sum them together

$$z = \sum_n s_n m_n$$

And this is the basic idea of **Attention Model**.



Note that it doesn't matter how to calculate **m1 ... mn**. The picture depicted above shows that **mi** is just the similarity (dot product) of features **ci** and **yi**

$$m_i = cy_i$$

And **z** is calculated by

$$z = \sum_n s_n y_n$$

Arithmetic, Geometric and Harmonic Means in Data Analysis

https://towardsdatascience.com/on-average-youre-using-the-wrong-average-geometric-harmonic-means-in-data-analysis-2a703e21ea0

arithmetic mean

$$(\Sigma_i x_i) \bullet (1 / m),$$

is dominated by numbers on the larger scale.

geometric mean

$$(\Pi_i x_i) \wedge (1 / m),$$

can handle varying proportions with ease, due to it’s multiplicative nature, but unitless.
s

harmonic mean

1. take the reciprocal of all numbers in the dataset
2. find the arithmetic mean of those reciprocals
3. take the reciprocal of that number

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} = \left(\frac{\sum_{i=1}^{n} x_i^{-1}}{n} \right)^{-1}$$

can find multiplicative / divisory relationships between fractions without worrying over common denominators.

relationships

the **geometric mean** of a dataset is equivalent to the **arithmetic mean** of the ***logarithms*** of each number in that dataset. So just as the **harmonic mean** is simply the **arithmetic mean** with a few reciprocal transformations, the **geometric mean** is just the **arithmetic mean** with a log transformation