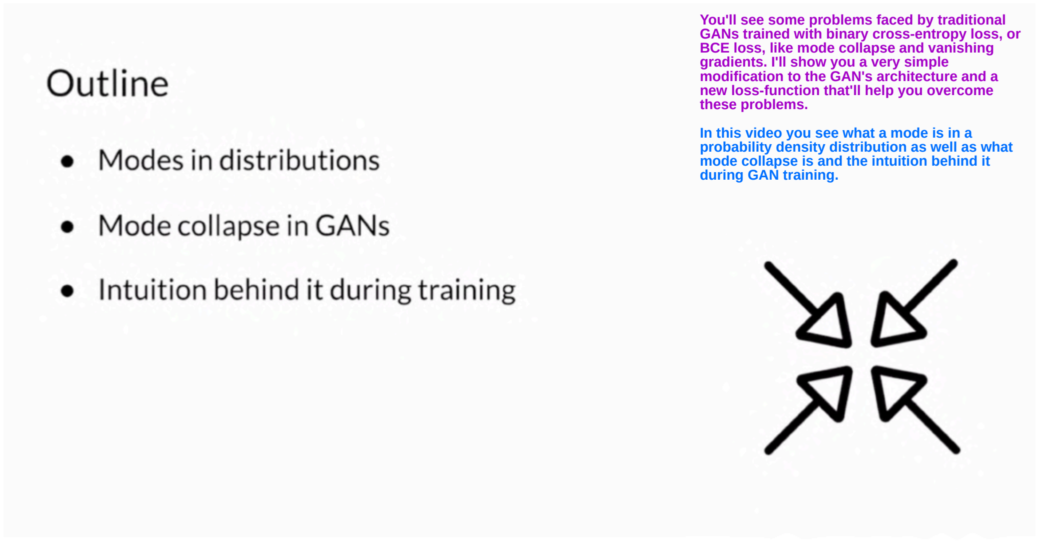
Mode Collapse

deeplearning.ai

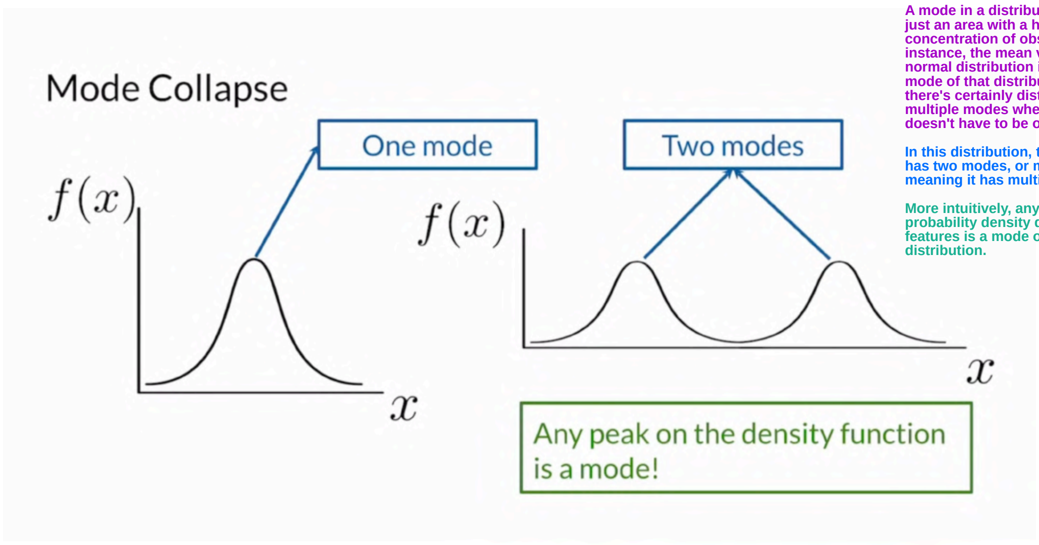


Outline

- Modes in distributions
- Mode collapse in GANs
- Intuition behind it during training

You'll see some problems faced by traditional GANs trained with binary cross-entropy loss, or BCE loss, like mode collapse and vanishing gradients. I'll show you a very simple modification to the GAN's architecture and a new loss-function that'll help you overcome these problems.

In this video you see what a mode is in a probability density distribution as well as what mode collapse is and the intuition behind it during GAN training.



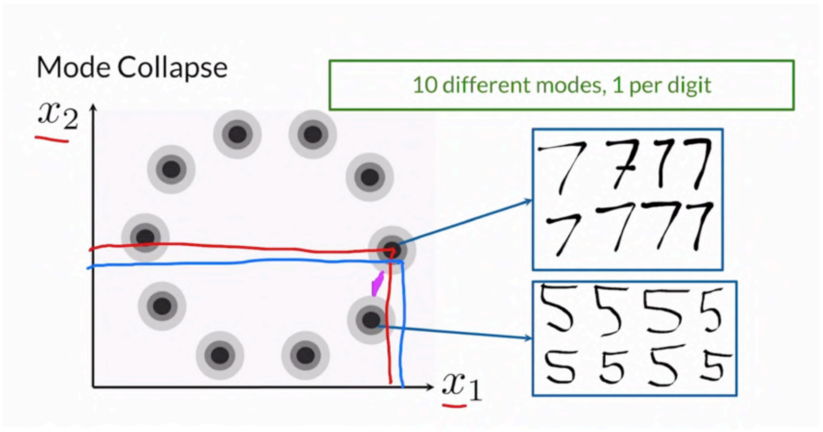
Mode Collapse

Any peak on the density function is a mode!

A mode in a distribution of data is just an area with a high concentration of observations. For instance, the mean value in a normal distribution is the single mode of that distribution, and there's certainly distributions with multiple modes where the mean doesn't have to be one of them.

In this distribution, that's bimodal, it has two modes, or multimodal, meaning it has multiple modes.

More intuitively, any peak on the probability density distribution over features is a mode of that distribution.

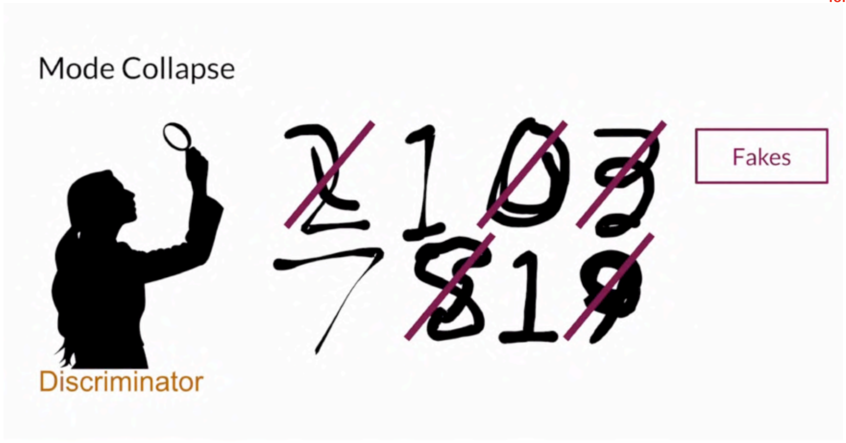


For example, take handwritten digits represented by features x_1 and x_2, meaning that these are just dimensions along which you can represent these numbers, like values you can use to represent different handwritten digits.

The probability density distribution in this case will be a surface with many peaks corresponding to each digit. This is definitely multimodal with 10 different modes and different observations of the number 7, for example, will be represented by similar x_1 and x_2 pairs. Those who values there would both be sevens and the one marked in red there in the mean would be an average looking seven.

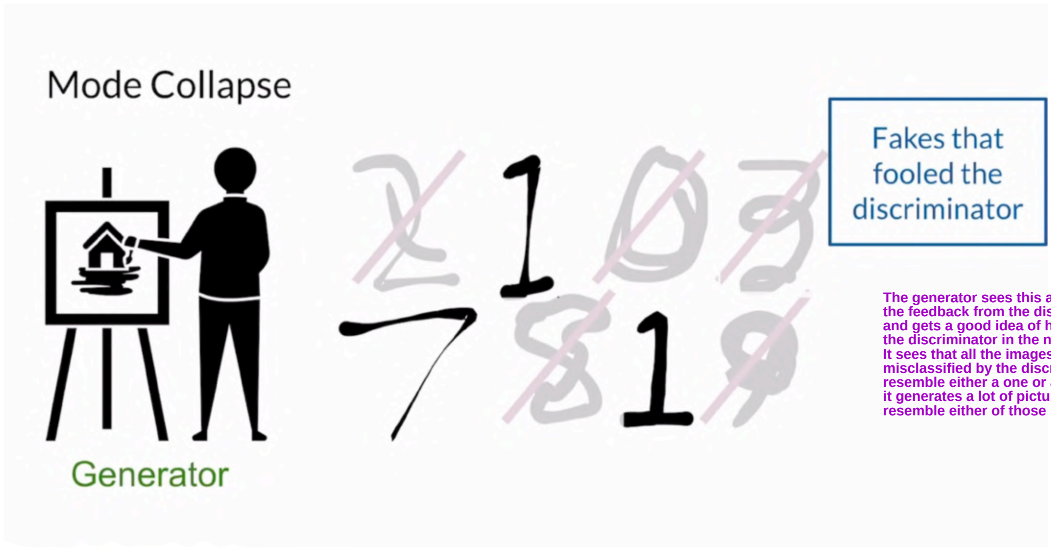
You can imagine each of these peaks coming out at you in a 3D representation where the darker circle represents higher altitudes. Different pairs of x_1 and x_2 features would create these handwritten fives here and a value in between the seven and five, where it's very low density, so very low probability of generating something that is an intermediate between seven and five in the real dataset, it would be probably a mixture of seven and five. But there's a very low probability that you would see that x_1, x_2 pair in this in-between space, that would produce an intermediate five seven looking number.

Different observations of the same digit will be grouped together in this feature space with high concentration in the area with the most common way to write that digit or where that average seven is there and, of course, an average five is in the center of the five mode peak. This probability density distribution over these features, x_1 and x_2, will have 10 modes, one for each of these digits.

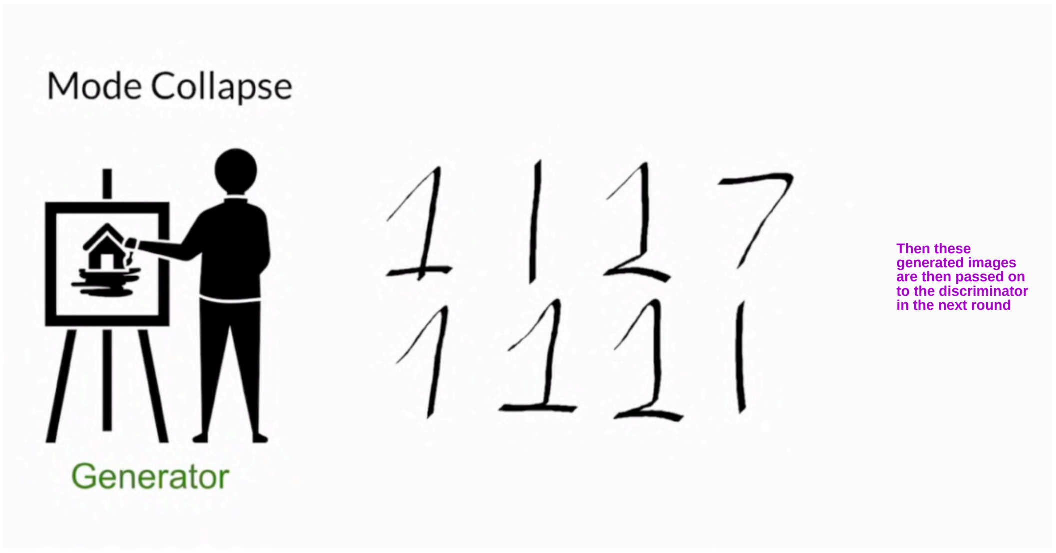


Many real life distributions used for training GANs are multimodal like this one, and I'll continue with a handwritten digit example to show you what mode collapse might mean, which intuitively, as you just get started, sounds like things collapsing to one mode or fewer modes and some of the modes disappearing.

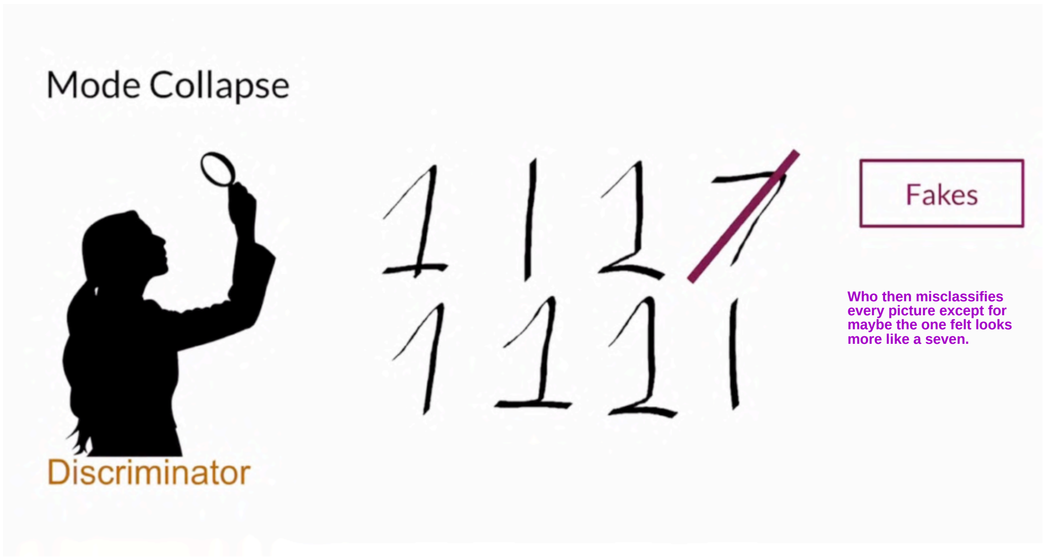
Take a discriminator that has learned to be good at identifying which handwritten digits are fakes, except for cases where the generated images look like ones and sevens. This could mean the discriminator is at of local minima of its cost function. The discriminator classifies most of the digits correctly, except for the ones that resembled those ones and sevens, then this information is passed on to the generator.



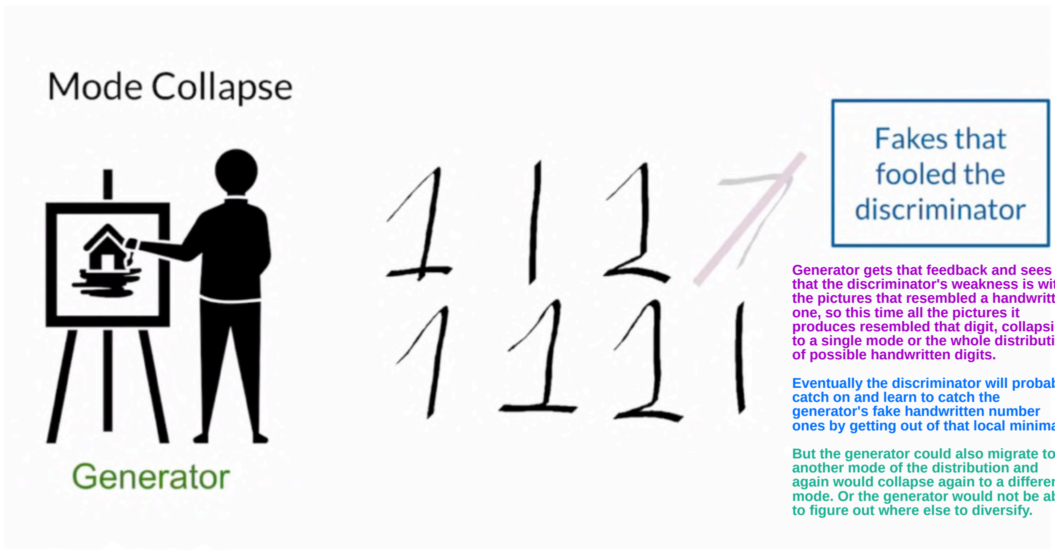
The generator sees this and looks at the feedback from the discriminator and gets a good idea of how to fool the discriminator in the next round. It sees that all the images were misclassified by the discriminator, resemble either a one or a seven, so it generates a lot of pictures that resemble either of those numbers.



Then these generated images are then passed on to the discriminator in the next round



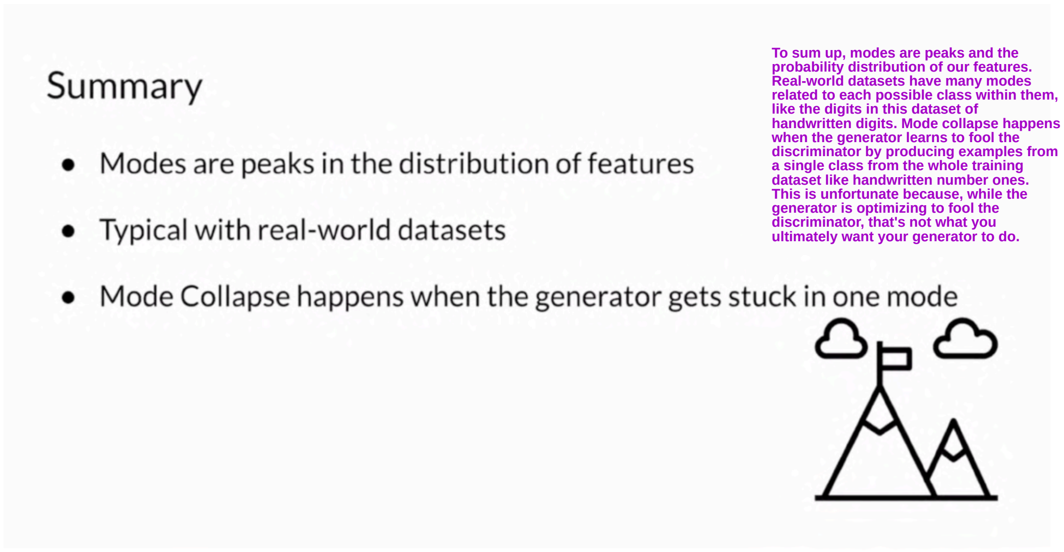
Who then misclassifies every picture except for maybe the one felt looks more like a seven.



Generator gets that feedback and sees that the discriminator's weakness is with the pictures that resembled a handwritten one, so this time all the pictures it produces resembled that digit, collapsing to a single mode or the whole distribution of possible handwritten digits.

Eventually the discriminator will probably catch on and learn to catch the generator's fake handwritten number ones by getting out of that local minima.

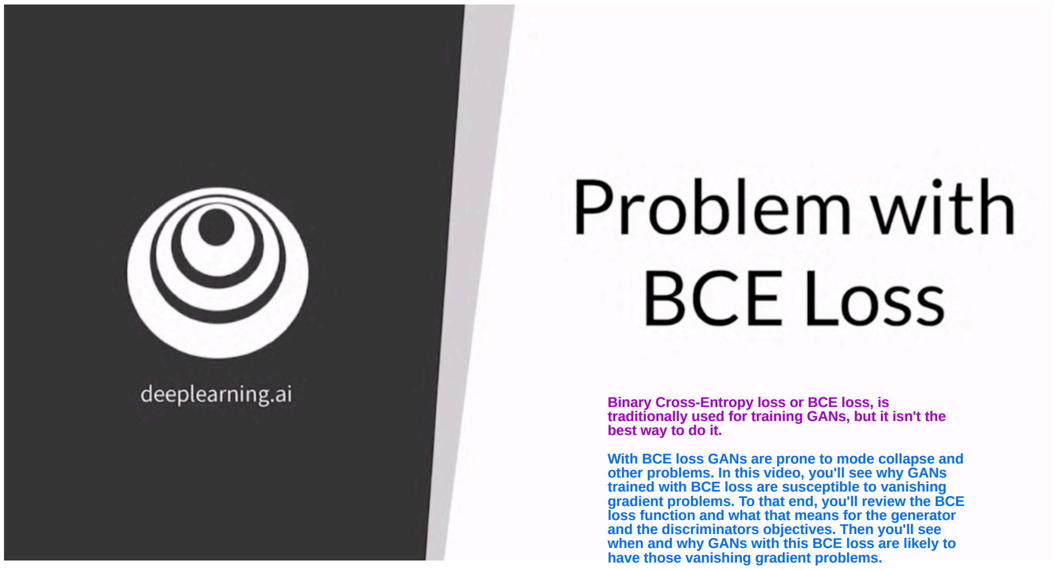
But the generator could also migrate to another mode of the distribution and again would collapse again to a different mode. Or the generator would not be able to figure out where else to diversify.



Summary

- Modes are peaks in the distribution of features
- Typical with real-world datasets
- Mode Collapse happens when the generator gets stuck in one mode

To sum up, modes are peaks and the probability distribution of our features. Real-world datasets have many modes related to each possible class within them, like the digits in this dataset of handwritten digits. Mode collapse happens when the generator learns to fool the discriminator by producing examples from a single class from the whole training dataset like handwritten number ones. This is unfortunate because, while the generator is optimizing to fool the discriminator, that's not what you ultimately want your generator to do.

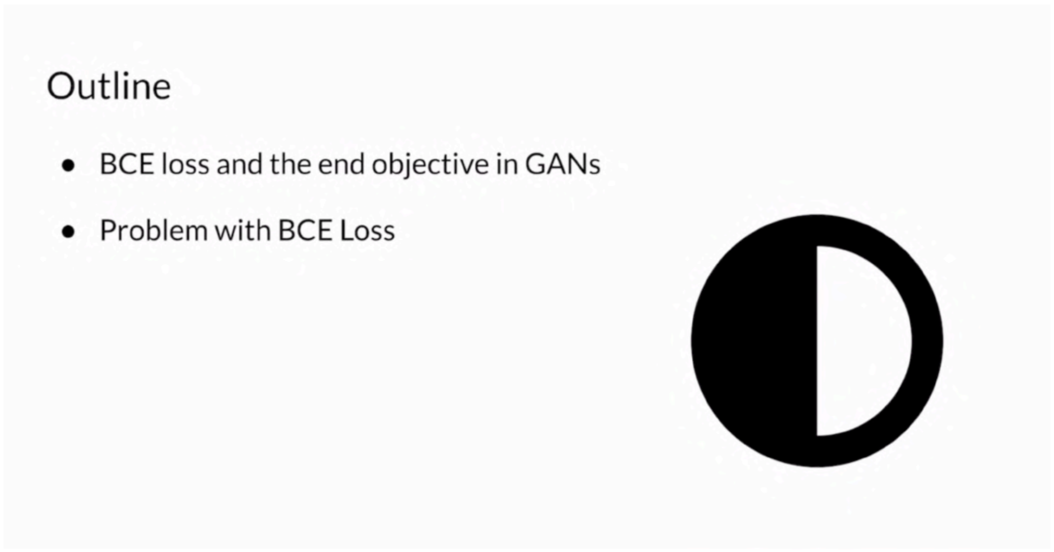


deeplearning.ai

Problem with BCE Loss

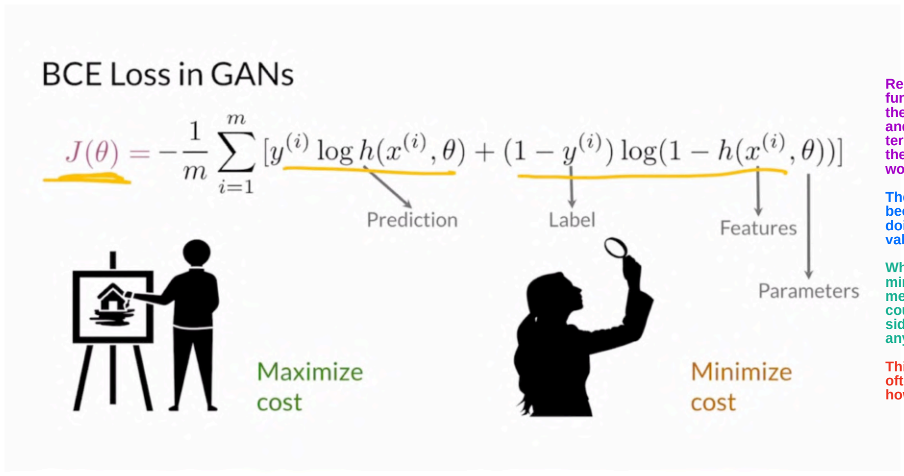
Binary Cross-Entropy loss or BCE loss, is traditionally used for training GANs, but it isn't the best way to do it.

With BCE loss GANs are prone to mode collapse and other problems. In this video, you'll see why GANs trained with BCE loss are susceptible to vanishing gradient problems. To that end, you'll review the BCE loss function and what that means for the generator and the discriminators objectives. Then you'll see when and why GANs with this BCE loss are likely to have those vanishing gradient problems.



Outline

- BCE loss and the end objective in GANs
- Problem with BCE Loss



BCE Loss in GANs

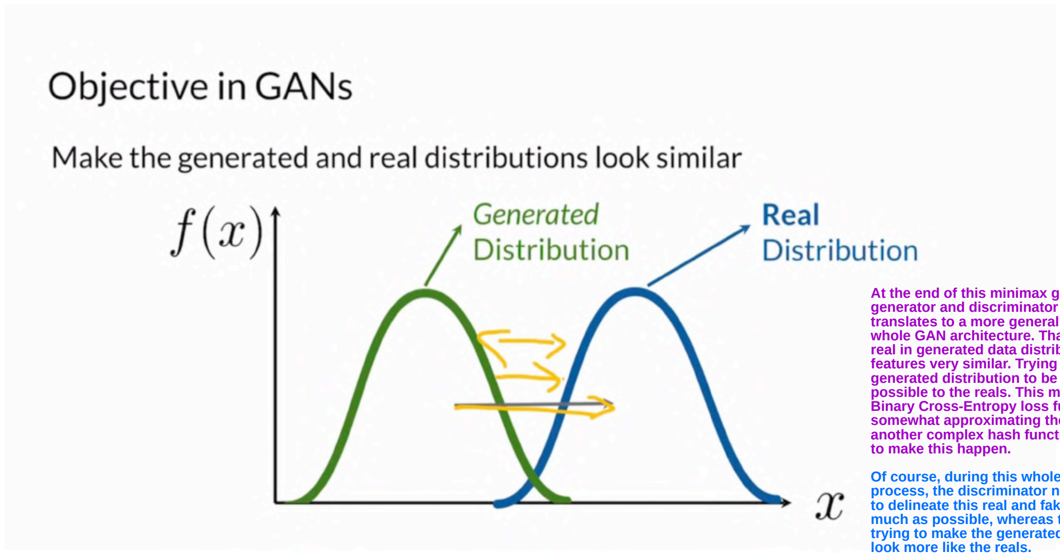
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))]$$

Remember the form of the BCE loss function, its just an average of the cost for the discriminator for misclassifying real and fake observations. Where the first term is for reals and the second term is for the fakes. The higher this cost value is, the worse the discriminator is doing at it.

The generator wants to maximize this cost because that means the discriminator is doing poorly and is classifying it's fake values into reals.

Whereas the discriminator wants to minimize this cost function because that means it's classifying things correctly. Of course the generator only sees the fake side of things, so it actually doesn't see anything about the reals.

This maximization and minimization is often called a minimax game, and that's how you might hear it being referred to as.

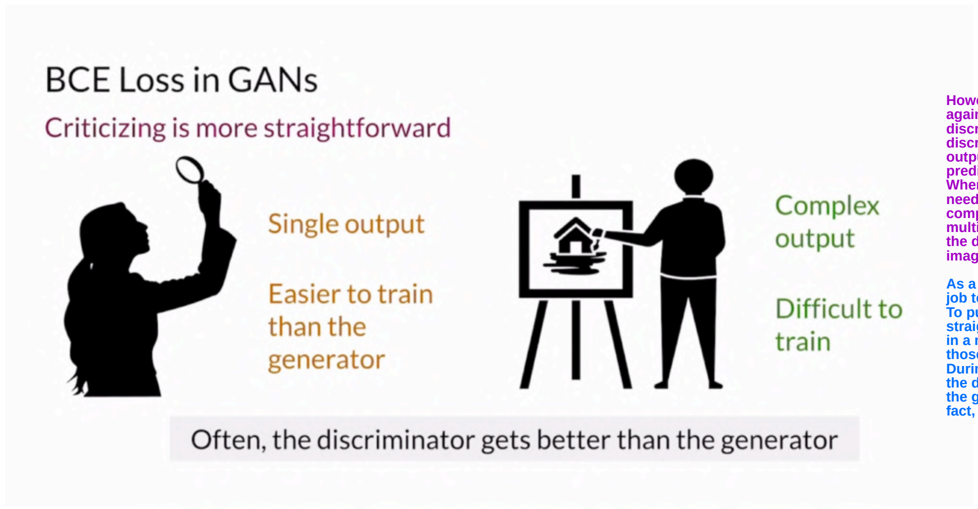


Objective in GANs

Make the generated and real distributions look similar

At the end of this minimax game, the generator and discriminator interaction translates to a more general objective for the whole GAN architecture. That is to make the real in generated data distributions of features very similar. Trying to get the generated distribution to be as close as possible to the reals. This minimax of the Binary Cross-Entropy loss function is somewhat approximating the minimization of another complex hash function that's trying to make this happen.

Of course, during this whole training process, the discriminator naturally is trying to delineate this real and fake distribution as much as possible, whereas the generator is trying to make the generated distribution look more like the reals.



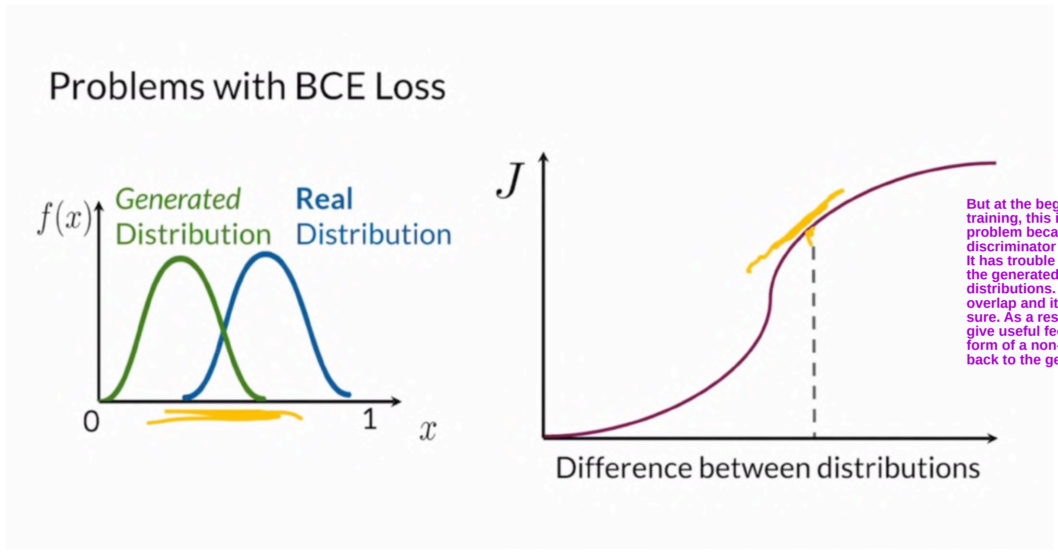
BCE Loss in GANs

Criticizing is more straightforward

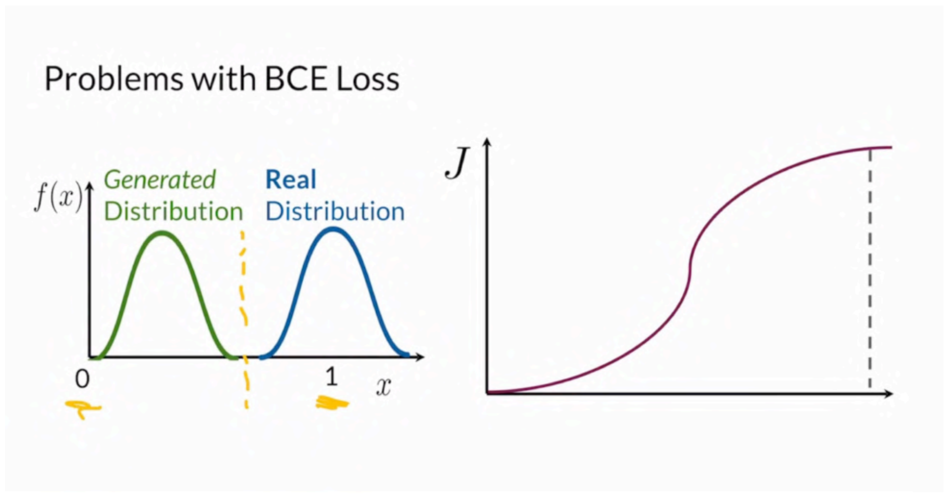
Often, the discriminator gets better than the generator

However, let's take a step back again to the generator and discriminators roles. The discriminator and again, needs to output just a single value prediction within zero and one. Whereas the generator actually needs to produce a pretty complex output composed of multiple features to try and fool the discriminator, for example, an image.

As a result that discriminators job tends to be a little bit easier. To put it in another way, it's more straightforward to look at images in a museum than it is to paint those masterpieces, right? During training it's possible for the discriminator to outperform the generator, very possible, in fact, quite common.

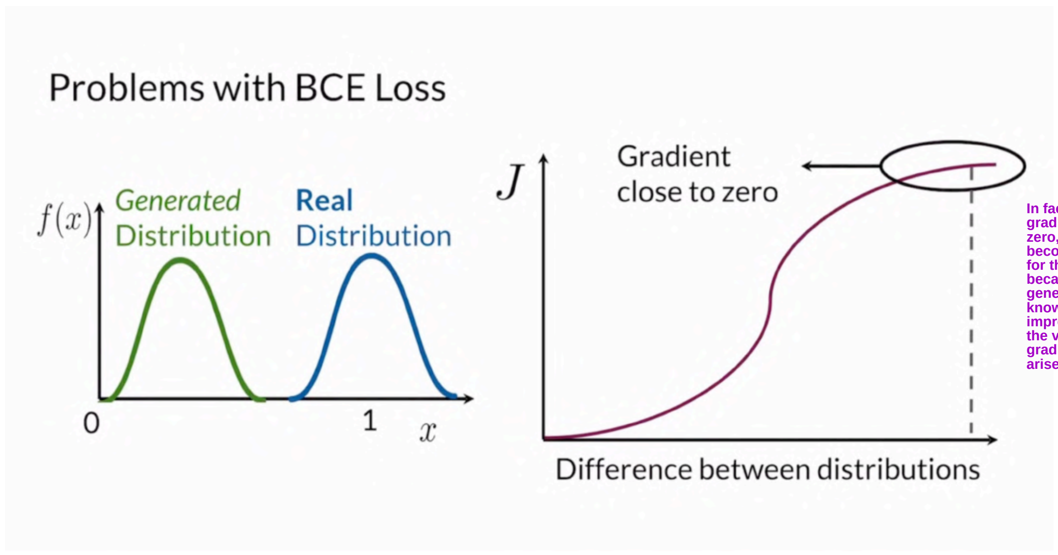


But at the beginning of training, this isn't such a big problem because the discriminator isn't that good. It has trouble distinguishing the generated and real distributions. There's some overlap and it's not quite sure. As a result, it's able to give useful feedback in the form of a non-zero gradient back to the generator.

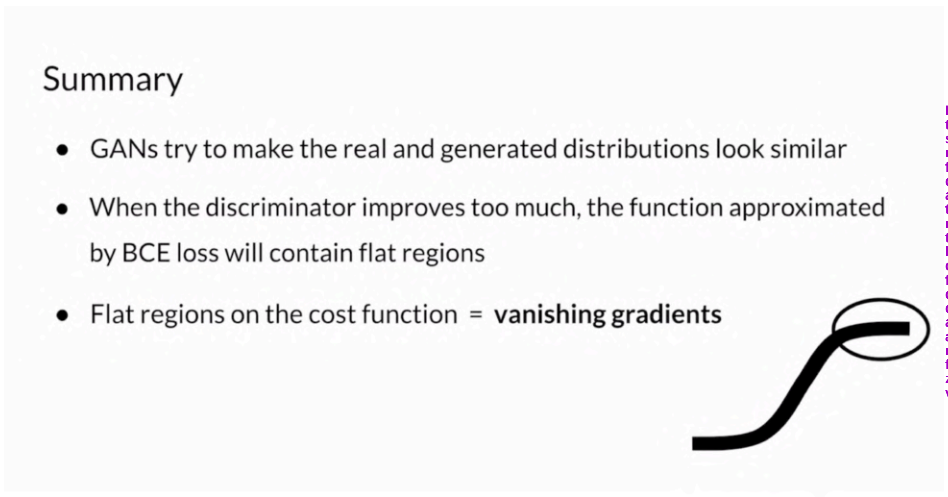


However, as it gets better at training, it starts to delineate the generated and real distributions a little bit more such that it can start distinguishing them much more. Where the real distribution will be centered around one and the generated distribution will start to approach zero.

As a result, when it's starting to get better, as this discriminator is getting better, it'll start giving less informative feedback.



In fact, it might give gradients closer to zero, and that becomes unhelpful for the generator because then the generator doesn't know how to improve. This is how the vanishing gradient problem will arise.



Summary

- GANs try to make the real and generated distributions look similar
- When the discriminator improves too much, the function approximated by BCE loss will contain flat regions
- Flat regions on the cost function = **vanishing gradients**

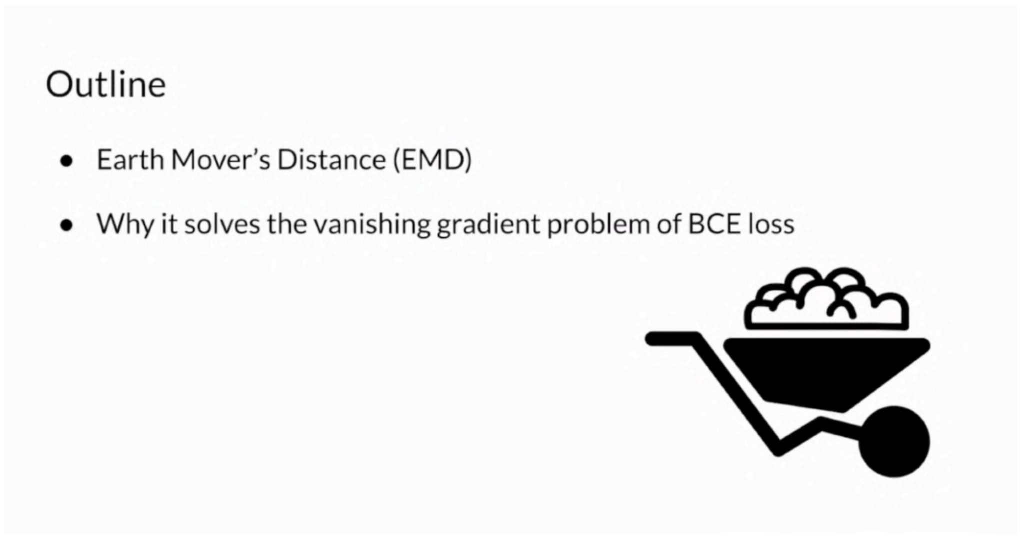
In summary, GANs try to make the generated distribution look similar to the real one by minimizing the underlying cost function that measures how different the distributions are. As a discriminator improves during training and sometimes improves more easily than the generator, that underlying cost function will have those flat regions when the distributions are very different from one another, where the discriminator is able to distinguish between the reals and the fakes much more easily, and be able to say, "Reals look really real, a label of one and fakes look really fake, a label of zero." All of this will cause vanishing gradient problems.

Earth Mover's Distance

deeplearning.ai

When using BCE loss to train a GAN, you often encounter mode collapse, and vanishing gradient problems due to the underlying cost function of the whole architecture. Even though there is an infinite number of decimal values between zero and one, the discriminator, as it improves, will be pushing towards those ends.

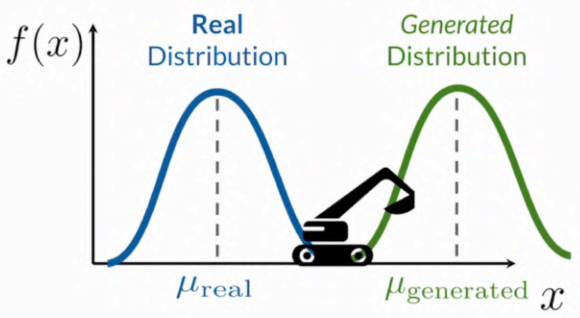
In this section, you'll see a different underlying cost function called Earth mover's distance, that measures the distance between two distributions and generally outperforms the one associated with BCE loss for training GANs. At the end, you'll see why this helps with the vanishing gradient problem.



Outline

- Earth Mover's Distance (EMD)
- Why it solves the vanishing gradient problem of BCE loss

Earth Mover's Distance



Effort to make the *generated* distribution equal to the **real** distribution

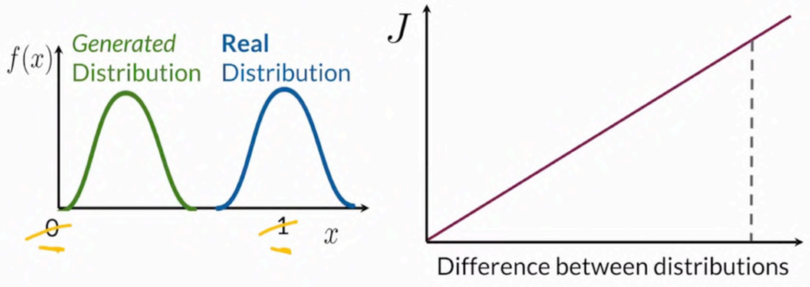
Depends on the distance and amount moved

So take this generated and real distributions with the same variance but different means, and assume they might be normal distributions. What the Earth Mover's distance does, is it measures how different these two distributions are, by estimating the amount of effort it takes to make the generated distribution equal to the real.

So intuitively, the generate distribution was a pile of dirt, how difficult would it be to move that pile of dirt and mold it into the shape and location of the real distribution? So that's what this Earth mover's distance means.

The function depends on both the distance and the amount that the generated distribution needs to be moved.

Earth Mover's Distance



So the problem with BCE loss is that as a discriminator improves, it would start giving more extreme values between zero and one, so values closer to one and closer to zero. As a result, this became less helpful feedback back to the generator. So the generator would stop learning due to vanishing gradient problems.

With Earth mover's distance, however, there's no such ceiling to the zero and one. So the cost function continues to grow regardless of how far apart these distributions are. The gradient of this measure won't approach zero and as a result, GANs are less prone to vanishing gradient problems and from vanishing gradient problems, mode collapse.

Summary

- Earth mover's distance (EMD) is a function of amount and distance
- Doesn't have flat regions when the distributions are very different
- Approximating EMD solves the problems associated with BCE

So wrapping up, Earth mover's distance is a function of the effort to make a distribution equal to another. So it depends on both distance and amount. Unlike BCE, it doesn't have flat regions when the distributions start to get very different, and the discriminator starts to improve a lot.

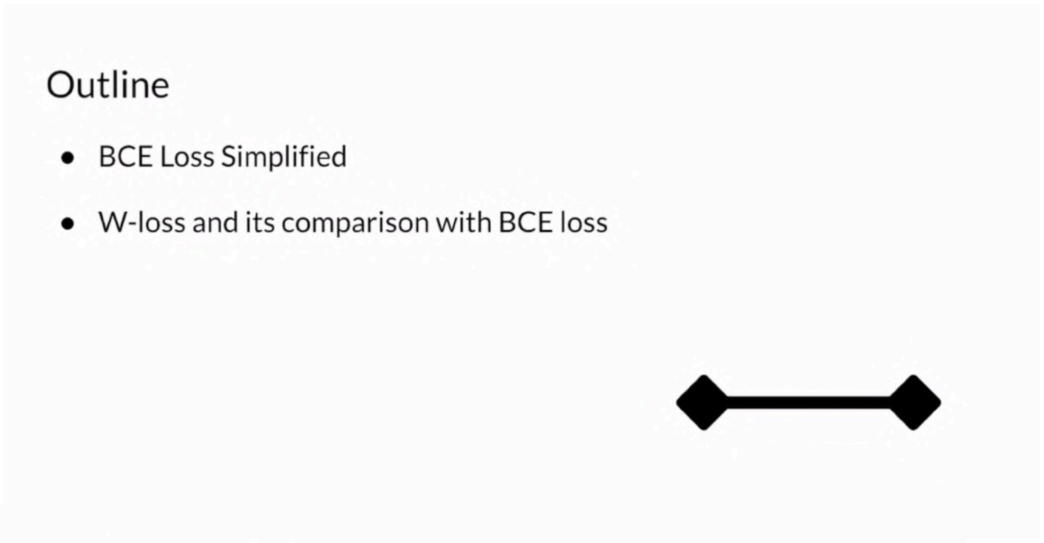
So approximating this measure eliminates the vanishing gradient problem, and reduces the likelihood of mode collapse in GANs. In the next few sections, you will see a loss function that uses Earth mover's distance for training GANs.

deeplearning.ai

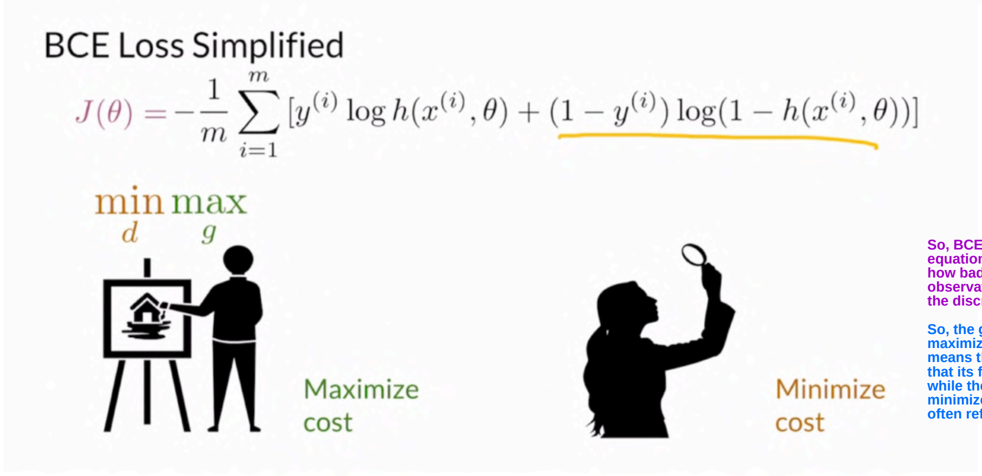
Wasserstein Loss

As you've seen previously, BCE Loss is used traditionally to train GANs. However, it has many problems due the form of the function it's approximated by.

So in this section, you will see an alternative loss function called Wasserstein Loss, or W-Loss for short, that approximates the Earth Mover's Distance that you saw in the previous video. So to that end, first you'll see an alternative way to look at the BCE Loss function that's more simple and compact, and I'll show you how W-Loss is calculated, and I'll compare this loss with BCE Loss.


Outline

- BCE Loss Simplified
- W-loss and its comparison with BCE loss


BCE Loss Simplified

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log h(x^{(i)},\theta) + \underline{(1-y^{(i)})\log(1-h(x^{(i)},\theta))}]$$

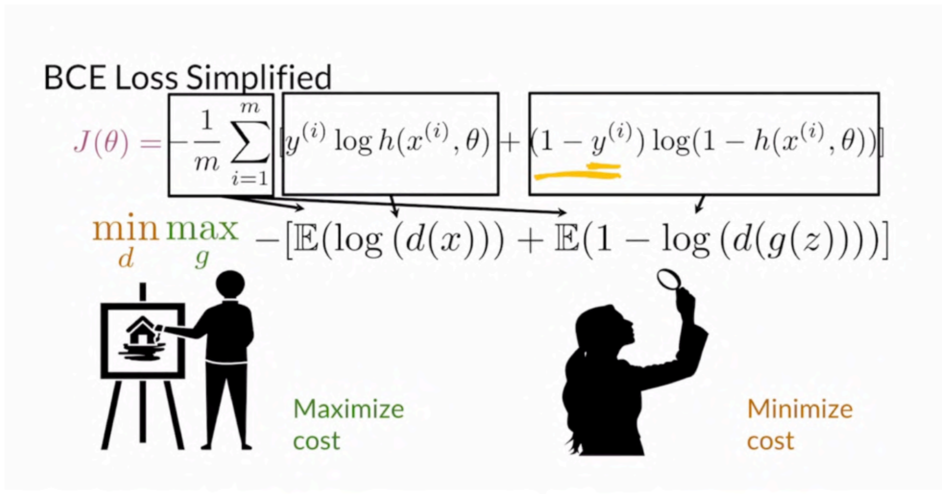
$$\min_{d}\max_{g}$$

Maximize cost

Minimize cost

So, BCE Loss is computed by a long equation that essentially measures how bad, on average, some observations are being classified by the discriminator, as fake and real.

So, the generator in GANs wants to maximize this cost, because that means the discriminator is saying that its fake values seem really real, while the discriminator wants to minimize that cost. And so, this is often referred to as a Minimax game.



BCE Loss Simplified

$$J(\theta) = \boxed{-\frac{1}{m}\sum_{i=1}^{m}}\left[\boxed{y^{(i)} \log h(x^{(i)},\theta)} + \boxed{\underline{(1-y^{(i)})} \log(1-h(x^{(i)},\theta))}\right]$$

$$\min_{d}\max_{g} -[\mathbb{E}(\log(d(x))) + \mathbb{E}(1-\log(d(g(z))))]$$

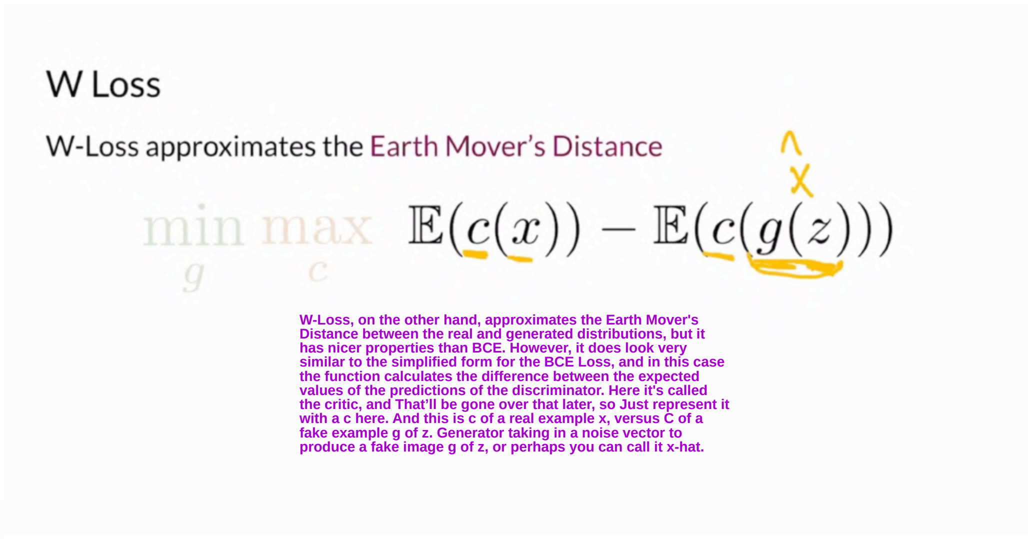
Maximize cost

Minimize cost

And this very long equation for BCE Loss can be simplified as follows.

The sum and division over examples M is nothing but a mean or expected value. In the first part, inside the sum, measures how bad the discriminator classifies real observations, where y equals 1, and 1 means real.

The second part measures how bad it classifies fake observations produced by the generator, where y of 1 means real, but 1 minus y, y of 0, means fake.



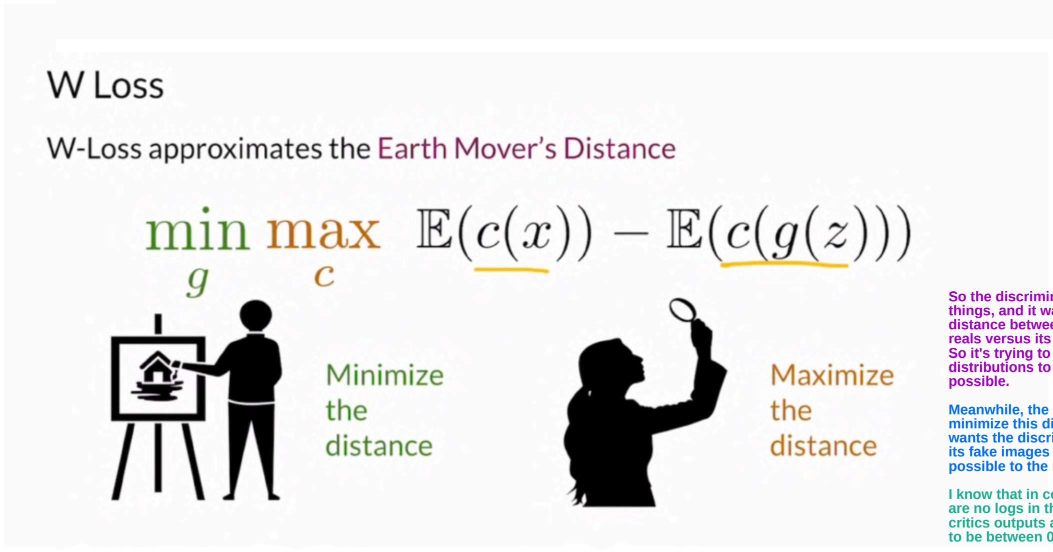
W Loss

W-Loss approximates the Earth Mover's Distance

$$\min_{g}\max_{c} \ \mathbb{E}(\underline{c}(\underline{x})) - \mathbb{E}(\underline{c}(\underbrace{g(z)}_{}))$$

$\hat{x}$

W-Loss, on the other hand, approximates the Earth Mover's Distance between the real and generated distributions, but it has nicer properties than BCE. However, it does look very similar to the simplified form for the BCE Loss, and in this case the function calculates the difference between the expected values of the predictions of the discriminator. Here it's called the critic, and That'll be gone over that later, so Just represent it with a c here. And this is c of a real example x, versus C of a fake example g of z. Generator taking in a noise vector to produce a fake image g of z, or perhaps you can call it x-hat.



W Loss

W-Loss approximates the Earth Mover's Distance

$$\min_{g}\max_{c} \ \mathbb{E}(\underline{c(x)}) - \mathbb{E}(\underline{c(g(z))})$$

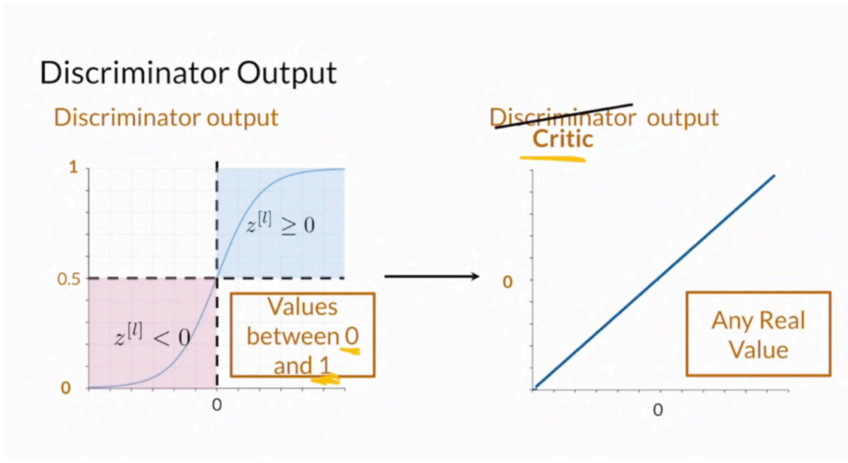
Minimize the distance

Maximize the distance

So the discriminator looks at these two things, and it wants to maximize the distance between its thoughts on the reals versus its thoughts on the fakes. So it's trying to push away these two distributions to be as far apart as possible.

Meanwhile, the generator wants to minimize this difference, because it wants the discriminator to think that its fake images are as close as possible to the reals.

I know that in contrast with BCE there are no logs in this function, since the critics outputs are no longer bounded to be between 0 and 1.



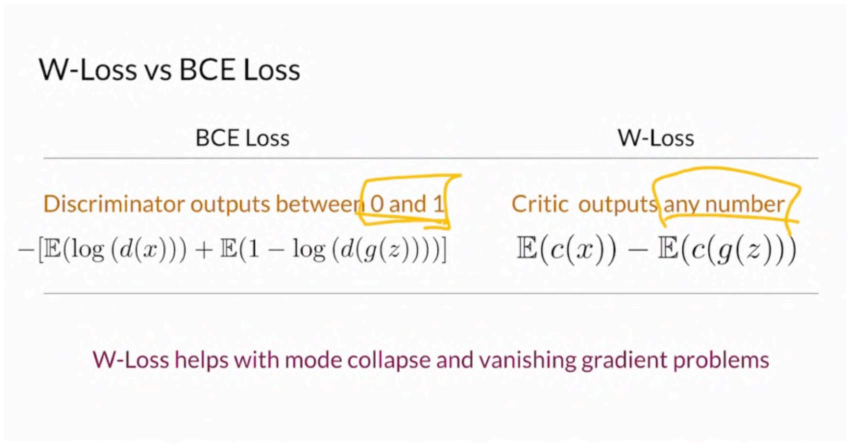
Discriminator Output

So, for the BCE Loss to make sense, the output of the discriminator needs to be a prediction between 0 and 1. And so the discriminator's neural network for GANs, trained with BCE Loss, have a sigmoid activation function in the output layer to then squash the values between 0 and 1.

W-Loss, however, doesn't have that requirement at all, so you can actually have a linear layer at the end of the discriminator's neural network, and that could produce any real value output. And you can interpret that output as, how real an image is considered by the critic, which, by the way, is now what we're calling the discriminator instead, because it's no longer bounded between 0 and 1, where 0 means fake, and 1 means real. It's no longer classifying into these two, or discriminating between these two classes.

And so, as a result, it wouldn't make that much sense to call that neural network a discriminator, because it doesn't discriminate between the classes.

And so, for W-Loss, the equivalent to a discriminator is called a critic, and what it tries to do is, maximize the distance between its evaluation on a fake, and its evaluation on a real.



W-Loss vs BCE Loss

BCE Loss	W-Loss
Discriminator outputs between 0 and 1	Critic outputs any number
$-[\mathbb{E}(\log(d(x))) + \mathbb{E}(1 - \log(d(g(z))))]$	$\mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$

W-Loss helps with mode collapse and vanishing gradient problems

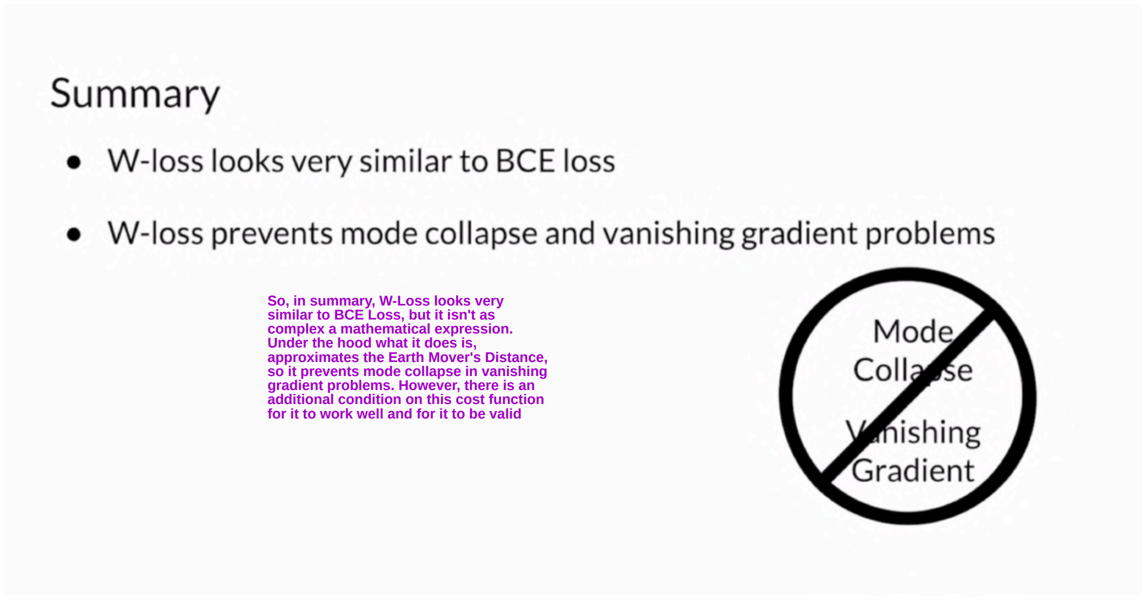
So, some of the main differences between W-Loss and BCE Loss is that, the discriminator under BCE Loss outputs a value between 0 and 1, while the critic in W-Loss will output any number.

Additionally, the forms of the cost functions is very similar, but W-Loss doesn't have any logarithms within it, and that's because it's a measure of how far the prediction of the critic for the real is from its prediction on the fake.

Meanwhile, BCE Loss does measure that distance between fake or a real, but to a ground truth of 1 or 0.

And so what's important to take away here is largely that, the discriminator is bounded between 0 and 1, whereas the critic is no longer bounded ,and just trying to separate the two distributions as much as possible.

And as a result, because it's not bounded, the critic is allowed to improve without degrading its feedback back to the generator. And this is because, it doesn't have a vanishing gradient problem, and this will mitigate against mode collapse, because the generator will always get useful feedback back.

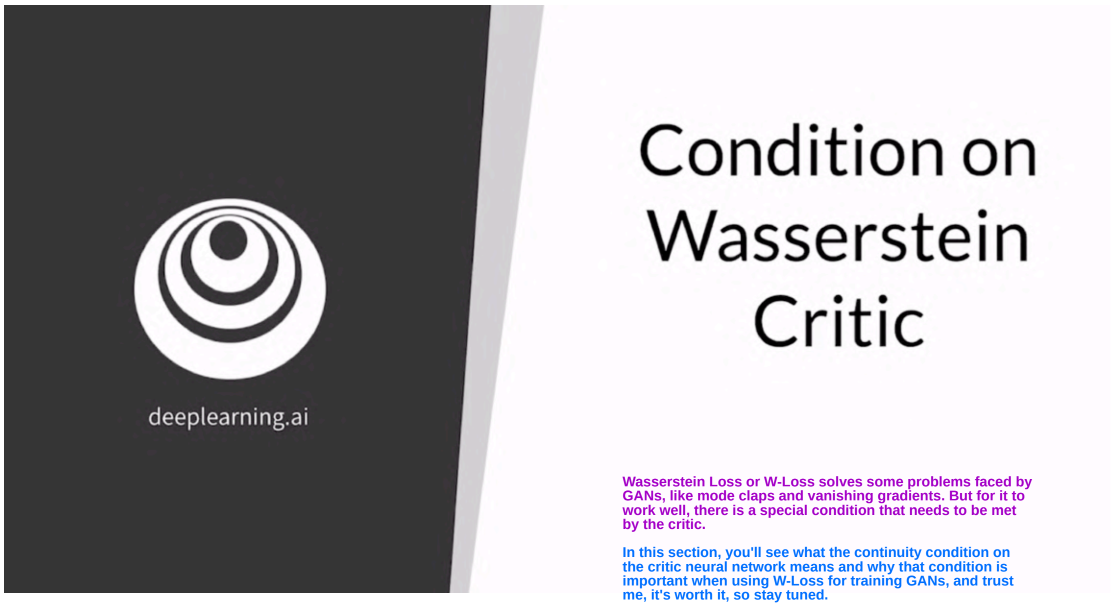


Summary

- W-loss looks very similar to BCE loss
- W-loss prevents mode collapse and vanishing gradient problems

Mode Collapse
Vanishing Gradient

So, in summary, W-Loss looks very similar to BCE Loss, but it isn't as complex a mathematical expression. Under the hood what it does is, approximates the Earth Mover's Distance, so it prevents mode collapse in vanishing gradient problems. However, there is an additional condition on this cost function for it to work well and for it to be valid

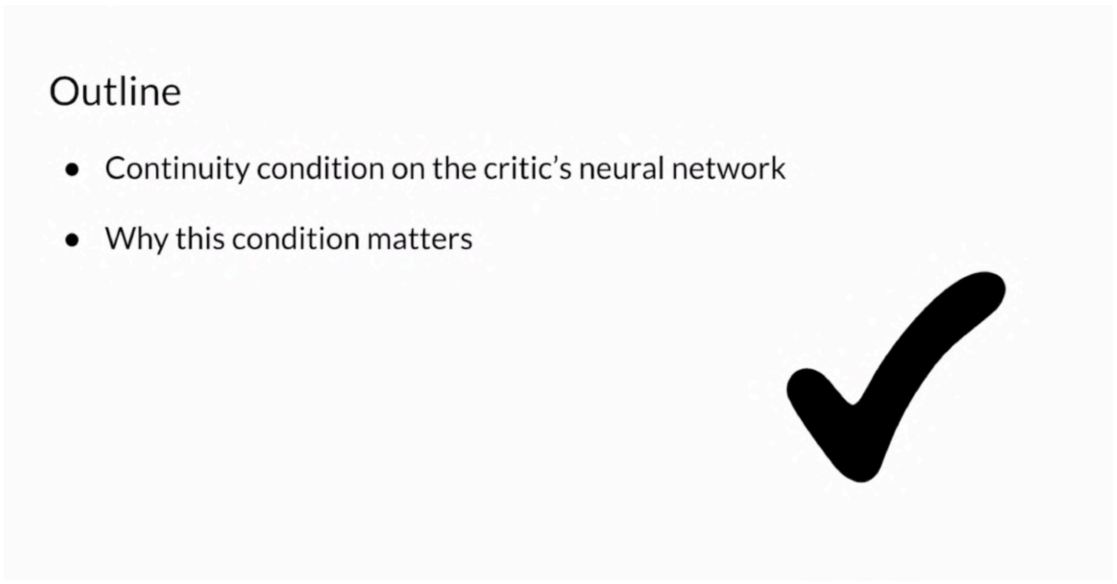


deeplearning.ai

Condition on Wasserstein Critic

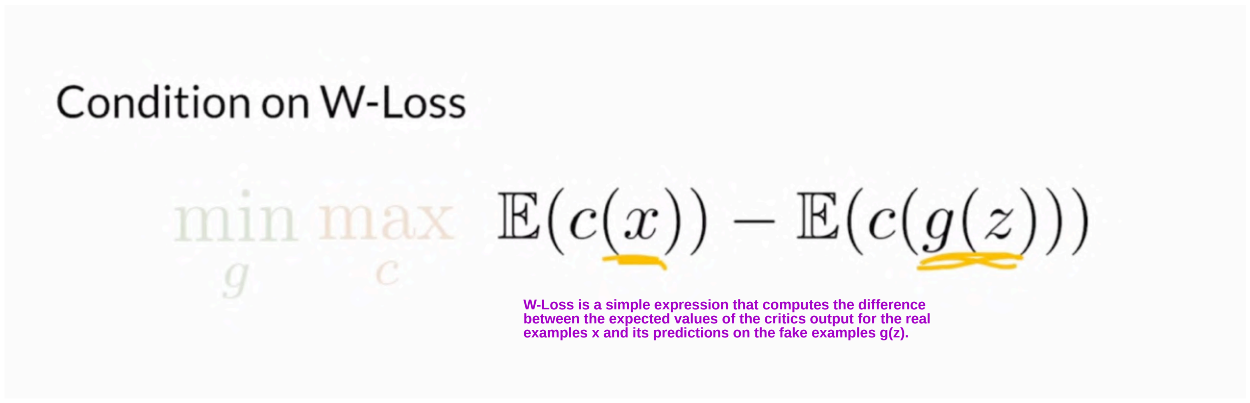
Wasserstein Loss or W-Loss solves some problems faced by GANs, like mode claps and vanishing gradients. But for it to work well, there is a special condition that needs to be met by the critic.

In this section, you'll see what the continuity condition on the critic neural network means and why that condition is important when using W-Loss for training GANs, and trust me, it's worth it, so stay tuned.



Outline

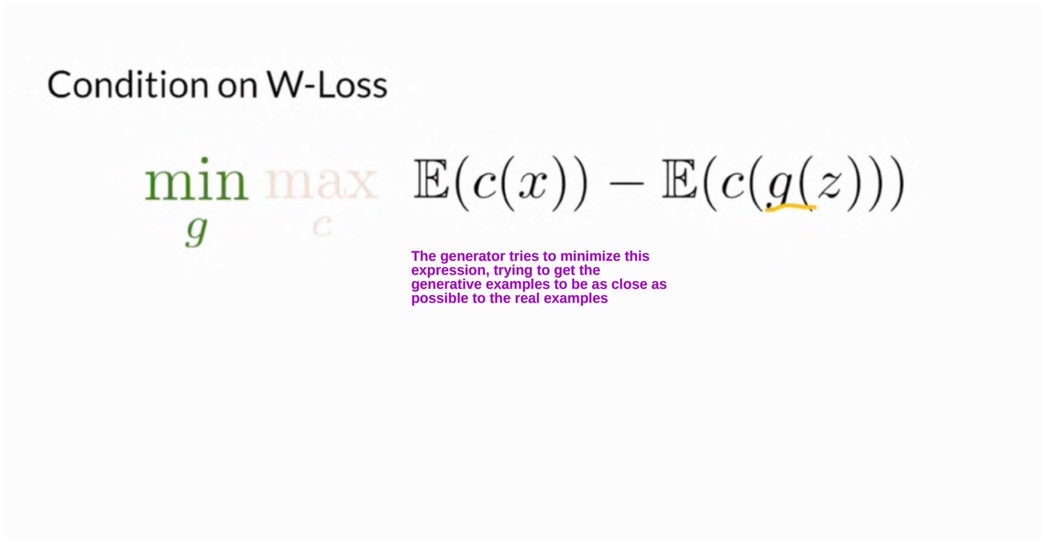
- Continuity condition on the critic's neural network
- Why this condition matters



Condition on W-Loss

$$\min_g \max_c \mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$$

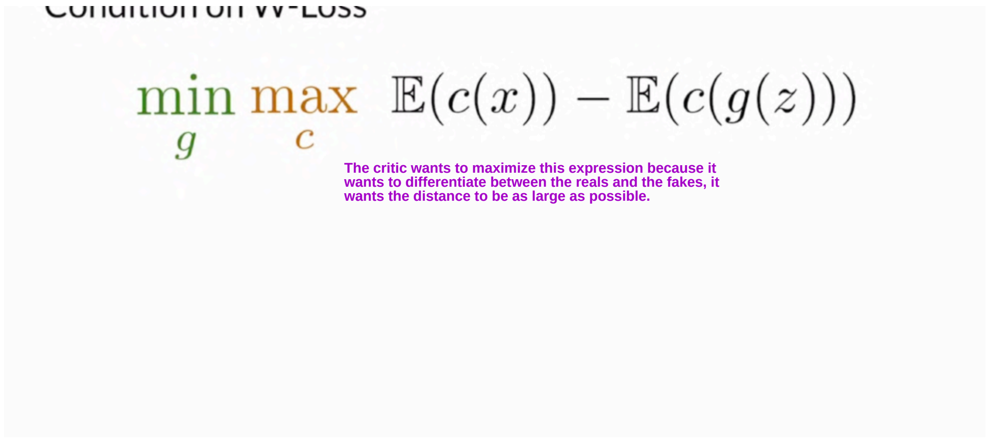
W-Loss is a simple expression that computes the difference between the expected values of the critics output for the real examples x and its predictions on the fake examples g(z).



Condition on W-Loss

$$\min_{g} \max_{c} \mathbb{E}(c(x)) - \mathbb{E}(c(\underline{g}(z)))$$

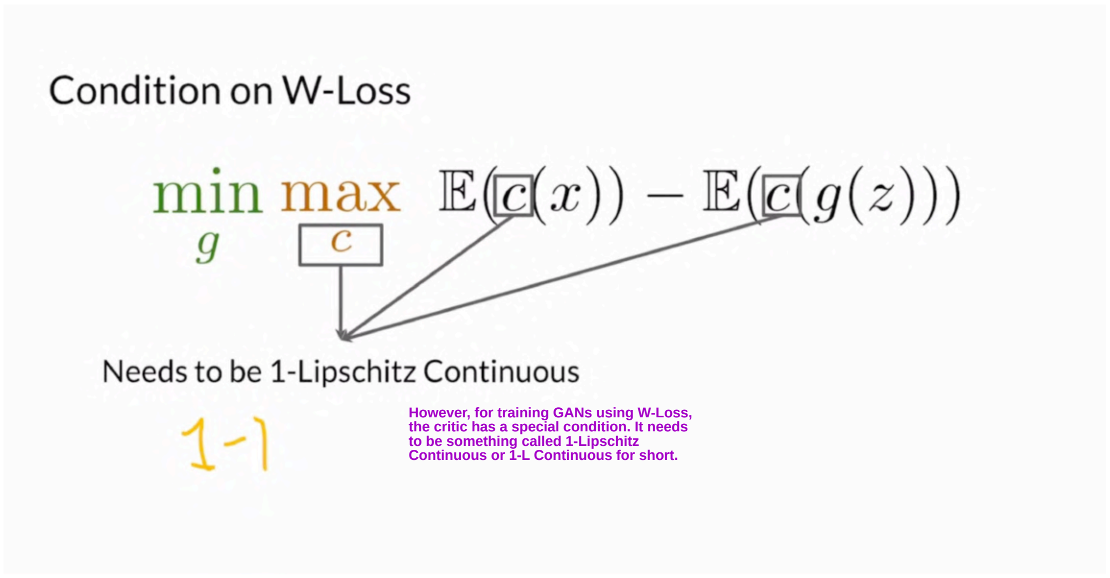
The generator tries to minimize this expression, trying to get the generative examples to be as close as possible to the real examples



Condition on W-Loss

$$\min_{g} \max_{c} \mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$$

The critic wants to maximize this expression because it wants to differentiate between the reals and the fakes, it wants the distance to be as large as possible.



Condition on W-Loss

$$\min_{g} \max_{\boxed{c}} \mathbb{E}(\boxed{c}(x)) - \mathbb{E}(\boxed{c}(g(z)))$$

Needs to be 1-Lipschitz Continuous

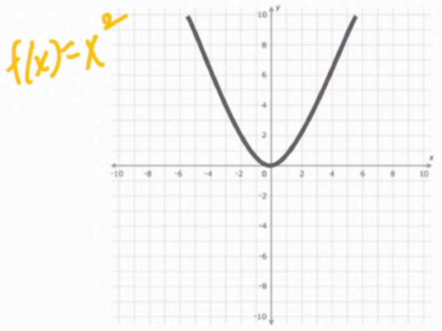
1-1

However, for training GANs using W-Loss, the critic has a special condition. It needs to be something called 1-Lipschitz Continuous or 1-L Continuous for short.

Condition on W-Loss

Critic needs to be **1**-L Continuous

The norm of the gradient should be at most **1** for *every point*

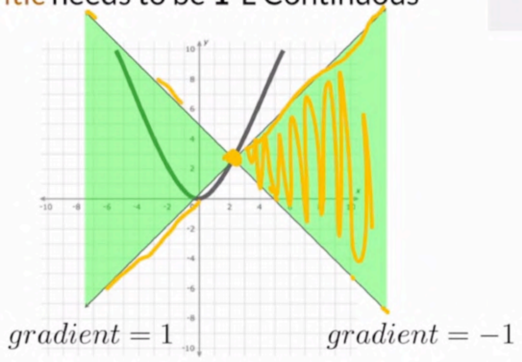


This condition sounds more sophisticated than it really is. For a function like the critics neural network to be at 1-Lipschitz Continuous, the norm of its gradient needs to be at most one. What that means is that, the slope can't be greater than one at any point, its gradient can't be greater than one. To check if a function here, for example, this function you see here, f(x) equals x squared, is 1-Lipschitz continuous...

Condition on W-Loss

Critic needs to be **1**-L Continuous

The norm of the gradient should be at most **1** for *every point*



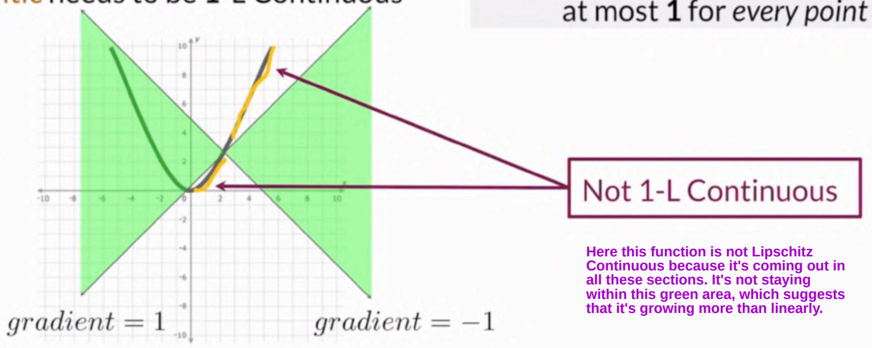
You want to go along every point in this function and make sure its slope is less than or equal to one, or its gradient is less than or equal to one, and what you can do is, you can actually draw two lines, one where the slope is exactly one at this certain point that you're evaluating function, and one where the slope is negative one where you're evaluating our function.

You want to make sure that the growth of this function never goes out of bounds from these lines because staying within these lines means that the function is growing linearly.

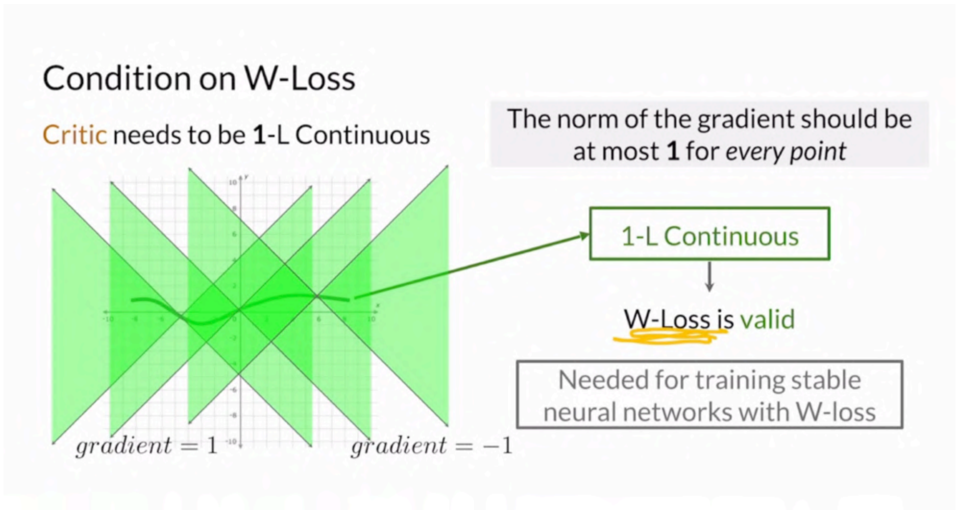
Condition on W-Loss

Critic needs to be **1**-L Continuous

The norm of the gradient should be at most **1** for *every point*



Here this function is not Lipschitz Continuous because it's coming out in all these sections. It's not staying within this green area, which suggests that it's growing more than linearly.



Condition on W-Loss

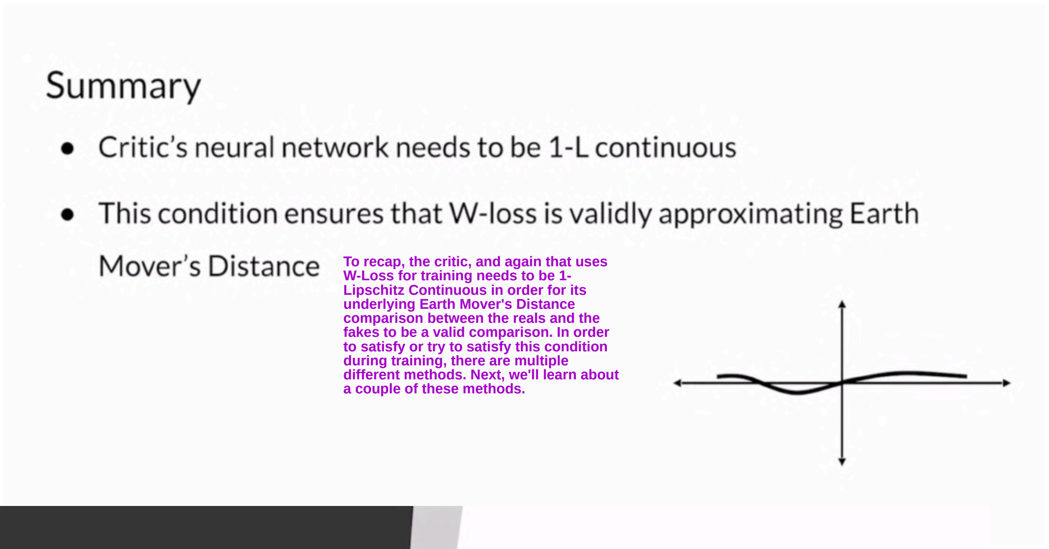
Critic needs to be **1**-L Continuous

The norm of the gradient should be at most **1** for *every point*

Look at another example here. This is a smooth curve functions. You want to again check every single point on this function before you can determine whether or not that this is 1-Lipschitz Continuous. Here it looks fine, function looks good. Here it also looks good, here looks good. Let's say you take every single value and the function never grows more than linearly.

This function is 1-Lipschitz Continuous. This condition on the critics neural network is important for W-Loss because it assures that the W-Loss function is not only continuous and differentiable, but also that it doesn't grow too much and maintain some stability during training. This is what makes the underlying Earth Movers Distance valid, which is what W-Loss is founded on.

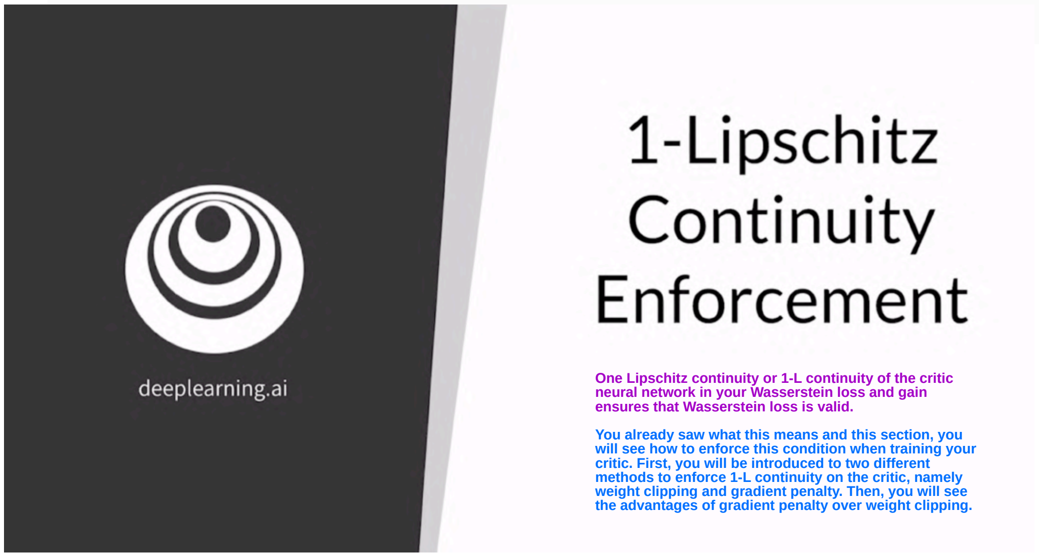
This is required for training both the critic and generators neural networks and it also increases stability because the variation as the GAN learns will be bounded.



Summary

- Critic's neural network needs to be 1-L continuous
- This condition ensures that W-loss is validly approximating Earth Mover's Distance

To recap, the critic, and again that uses W-Loss for training needs to be 1-Lipschitz Continuous in order for its underlying Earth Mover's Distance comparison between the reals and the fakes to be a valid comparison. In order to satisfy or try to satisfy this condition during training, there are multiple different methods. Next, we'll learn about a couple of these methods.

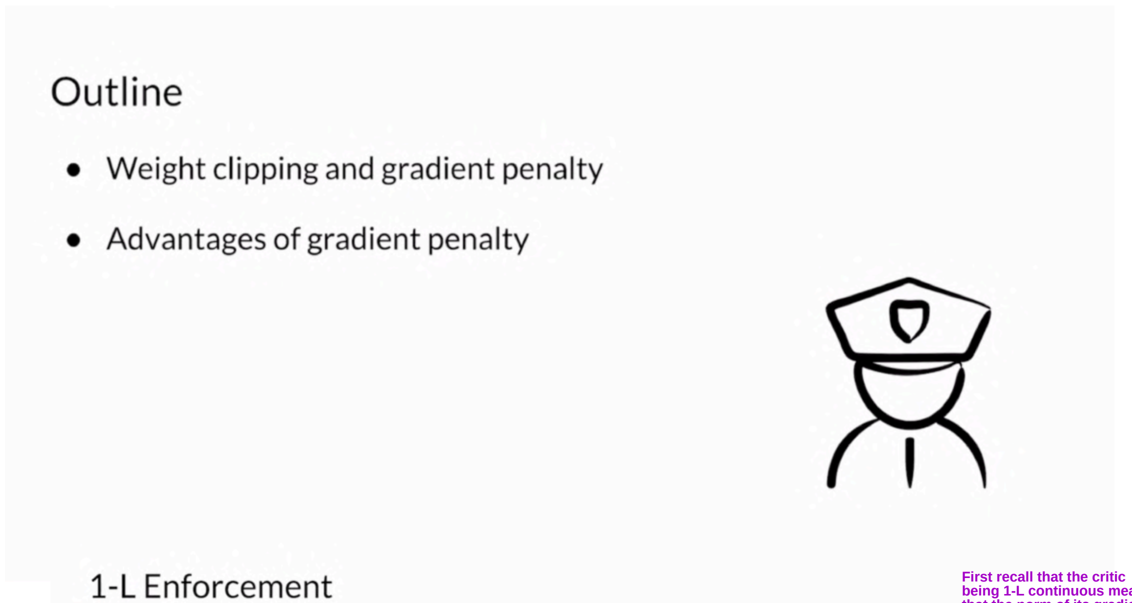


1-Lipschitz Continuity Enforcement

deeplearning.ai

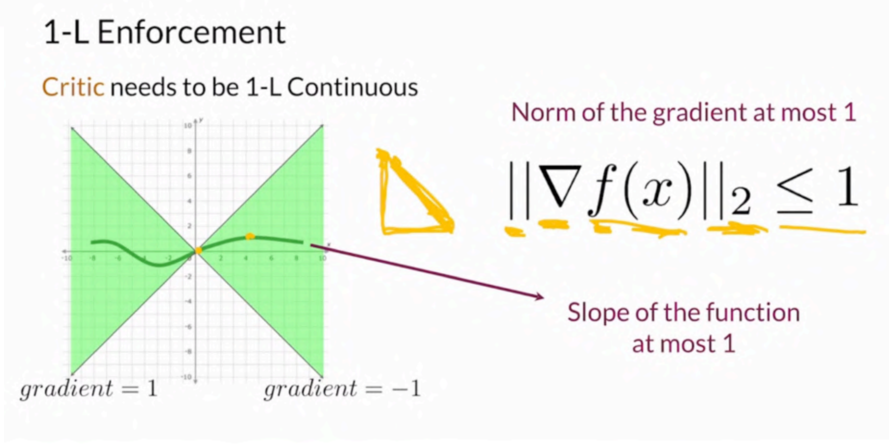
One Lipschitz continuity or 1-L continuity of the critic neural network in your Wasserstein loss and gain ensures that Wasserstein loss is valid.

You already saw what this means and this section, you will see how to enforce this condition when training your critic. First, you will be introduced to two different methods to enforce 1-L continuity on the critic, namely weight clipping and gradient penalty. Then, you will see the advantages of gradient penalty over weight clipping.



Outline

- Weight clipping and gradient penalty
- Advantages of gradient penalty



1-L Enforcement

Critic needs to be 1-L Continuous

Norm of the gradient at most 1

$$||\nabla f(x)||_2 \leq 1$$

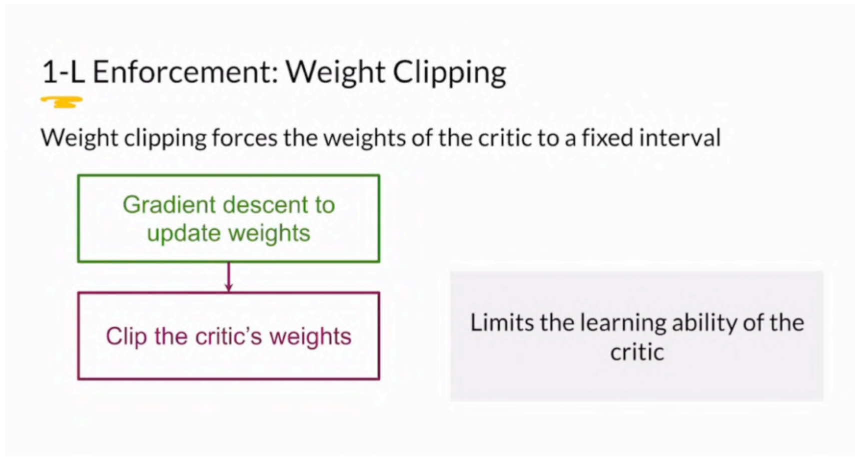
Slope of the function at most 1

$gradient = 1$ $gradient = -1$

First recall that the critic being 1-L continuous means that the norm of its gradient is at most one at every single point of this function. This upside down triangle is assigned for gradient, this is the function, perhaps F is the critic here and X is the image. This just represents the norm of that gradient being less than or equal to one.

Using the L2 norm is very common here, which just means its Euclidean distance or often thought of as your triangle distance of your hypotenuse. This is the distance between these two points not going this direction. It's this hypotenuse.

Intuitively in two-dimensions, it's that the slope is less than or equal to one. At every single point of this function, it'll remain within these green triangles.



1-L Enforcement: Weight Clipping

Weight clipping forces the weights of the critic to a fixed interval

Gradient descent to update weights

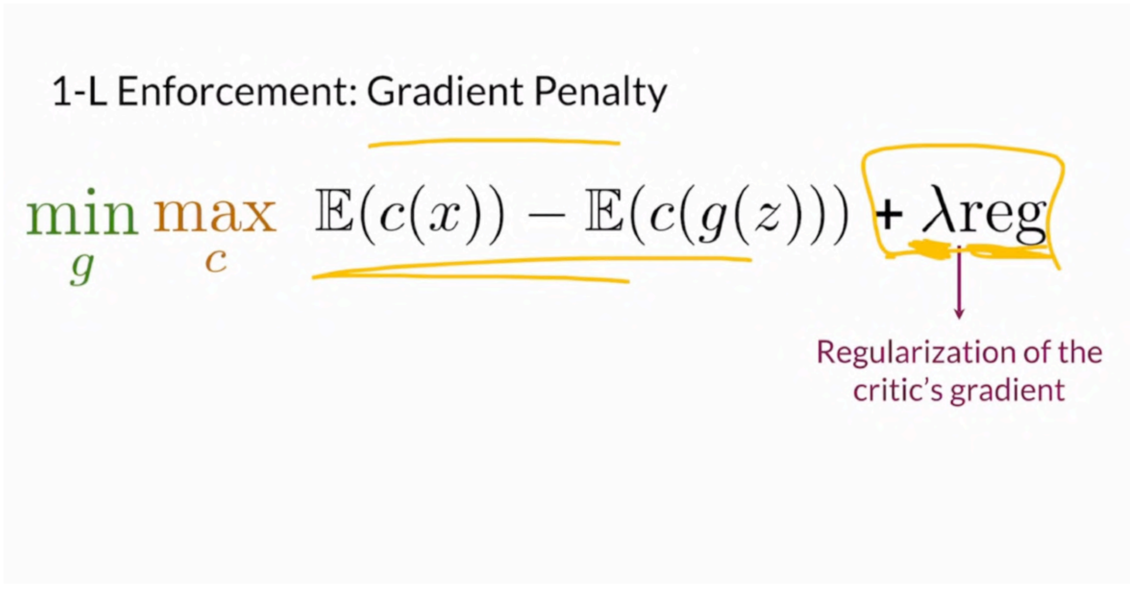
Clip the critic's weights

Limits the learning ability of the critic

Two common ways of ensuring this condition are weight clipping and gradient penalty.

With weight clipping, the weights of the critics neural network are forced to take values between a fixed interval. After you update the weights during gradient descent, you actually will clip any weights outside of the desired interval. Basically what that means is that weights over that interval, either too high or too low, will be set to the maximum or the minimum amount allowed. That's clipping the weights there.

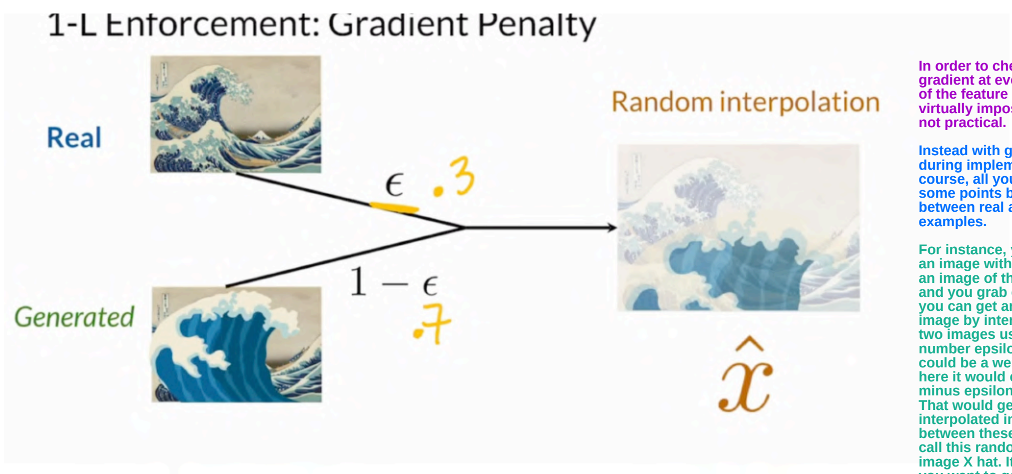
This is one way of enforcing the 1-L continuity, but it has a way to downside. Forcing the weights of the critic to a limited range of values could limit the critics ability to learn and ultimately for the gradient to perform because if the critic can't take on many different parameter values, it's weights can't take on many different values, it might not be able to improve easily or find good loop optimal for it to be in. Not only is this trying to do 1-L continuity enforcement, this might also limit the critic too much. Or on the other hand, it might actually limit the critic too little if you don't clip the weights enough. There's a lot of hyperparameter tuning involved.



1-L Enforcement: Gradient Penalty

$$\min_{g} \max_{c} \; \mathbb{E}(c(x)) - \mathbb{E}(c(g(z))) + \lambda \text{reg}$$

Regularization of the critic's gradient

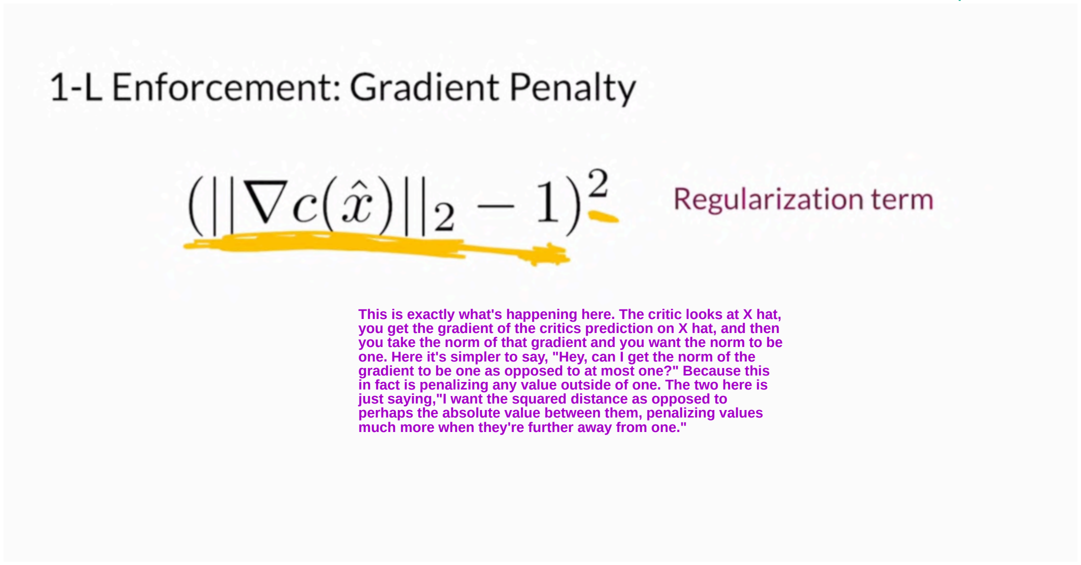


1-L Enforcement: Gradient Penalty

In order to check the critics gradient at every possible point of the feature space, that's virtually impossible or at least not practical.

Instead with gradient penalty during implementation, of course, all you do is sample some points by interpolating between real and fake examples.

For instance, you could sample an image with a set of reals and an image of the set of fakes, and you grab one of each and you can get an intermediate image by interpolating those two images using a random number epsilon. Epsilon here it could be a weight of 0.3, and here it would evaluate one minus epsilon would be 0.7. That would get you this random interpolated image that's in-between these two images. I'll call this random interpolated image X hat. It's on X hat that you want to get the critics gradient to be less than or equal to one.

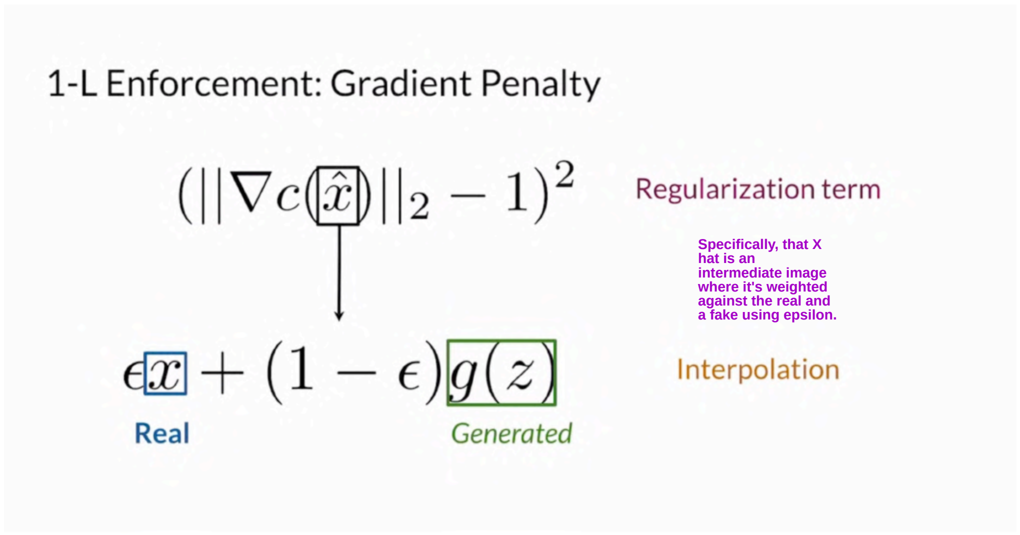


1-L Enforcement: Gradient Penalty

$$(\|\nabla c(\hat{x})\|_2 - 1)^2$$

Regularization term

This is exactly what's happening here. The critic looks at X hat, you get the gradient of the critics prediction on X hat, and then you take the norm of that gradient and you want the norm to be one. Here it's simpler to say, "Hey, can I get the norm of the gradient to be one as opposed to at most one?" Because this in fact is penalizing any value outside of one. The two here is just saying,"I want the squared distance as opposed to perhaps the absolute value between them, penalizing values much more when they're further away from one."



1-L Enforcement: Gradient Penalty

$$(||\nabla c(\boxed{\hat{x}})||_2 - 1)^2$$

Regularization term

$$\Big\downarrow$$

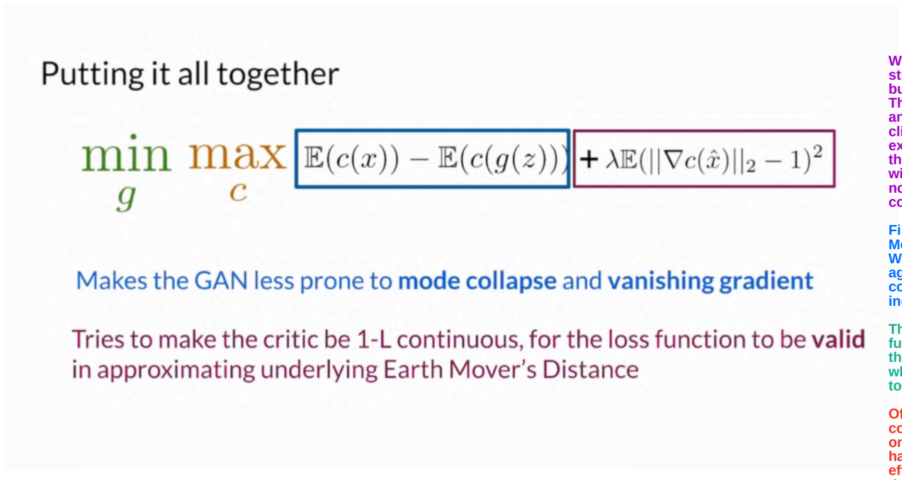
$$\epsilon \boxed{x} + (1-\epsilon)\boxed{g(z)}$$

Real

Generated

Interpolation

Specifically, that X hat is an intermediate image where it's weighted against the real and a fake using epsilon.



Putting it all together

$$\min_{g} \max_{c} \boxed{\mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))} \boxed{+ \lambda \mathbb{E}(||\nabla c(\hat{x})||_2 - 1)^2}$$

Makes the GAN less prone to **mode collapse** and **vanishing gradient**

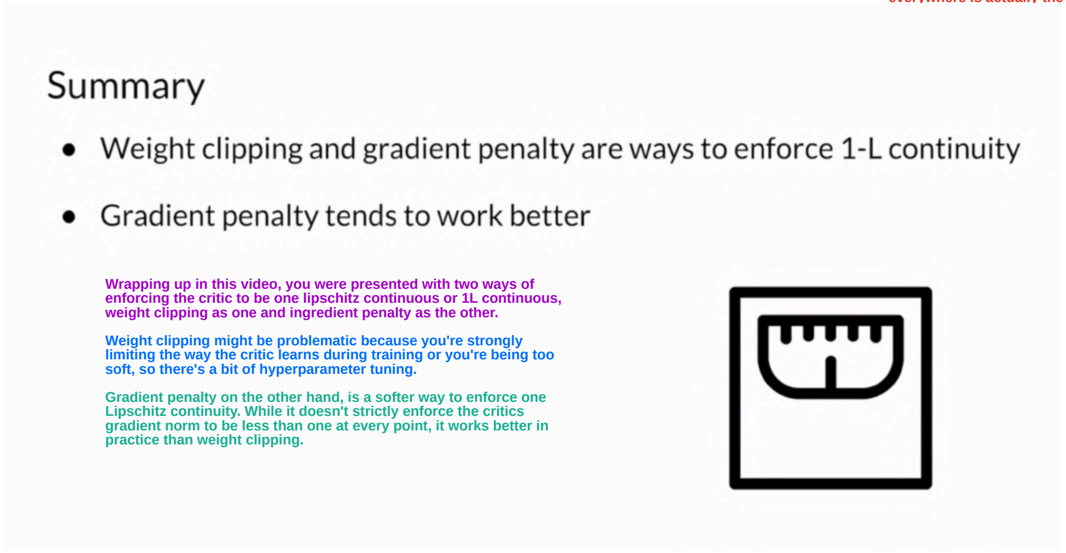
Tries to make the critic be 1-L continuous, for the loss function to be **valid** in approximating underlying Earth Mover's Distance

With this method, you're not strictly enforcing 1-L continuity, but you're just encouraging it. This has proven to work well and much better than weight clipping. The complete expression, the loss function that you use for training again with W loss ingredient penalty now has these two components.

First, you approximate Earth Mover's distance with this main W loss component. This makes again less parental mode collapse and managed ingredients.

The second part of this loss function is a regularization term that meets the condition for what the critic desires in order to make this main term valid.

Of course, this is a soft constraint on making the critic one lipschitz continuous, but it has been shown to be very effective. Keeping the norm of the critic close to one almost everywhere is actually the



Summary

- Weight clipping and gradient penalty are ways to enforce 1-L continuity
- Gradient penalty tends to work better

Wrapping up in this video, you were presented with two ways of enforcing the critic to be one lipschitz continuous or 1L continuous, weight clipping as one and ingredient penalty as the other.

Weight clipping might be problematic because you're strongly limiting the way the critic learns during training or you're being too soft, so there's a bit of hyperparameter tuning.

Gradient penalty on the other hand, is a softer way to enforce one Lipschitz continuity. While it doesn't strictly enforce the critics gradient norm to be less than one at every point, it works better in practice than weight clipping.

(Optional) The WGAN and WGAN-GP Papers

Interested in the papers behind the Wasserstein GAN with Gradient Penalty (WGAN-GP) you just implemented? Check them out! The first paper is the original WGAN paper and the second proposes GP (as well as weight clipping) to WGAN in order to enforce 1-Lipschitz continuity and improve stability.

Wasserstein GAN (Arjovsky, Chintala, and Bottou, 2017): https://arxiv.org/abs/1701.07875

Improved Training of Wasserstein GANs (Gulrajani et al., 2017): https://arxiv.org/abs/1704.00028

(Optional) WGAN Walkthrough

Want another explanation of WGAN? This article provides a great walkthrough of how WGAN addresses the difficulties of training a traditional GAN with a focus on the loss functions.

From GAN to WGAN (Weng, 2017): https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

Works Cited

All of the resources cited in Course 1 Week 3, in one place. You are encouraged to explore these papers/sites if they interest you—for this week, both papers have been included as an optional reading! They are listed in the order they appear in the lessons.

From the notebook:

- Wasserstein GAN (Arjovsky, Chintala, and Bottou, 2017): https://arxiv.org/abs/1701.07875
- Improved Training of Wasserstein GANs (Gulrajani et al., 2017): https://arxiv.org/abs/1704.00028
- MNIST Database: http://yann.lecun.com/exdb/mnist/