149

[R] [1701.07875] Wasserstein GAN

(arxiv.org)
submitted 1 month ago by ajmooch
157 comments share save hide give gold report

all 157 comments

sorted by: **best**

save content policy

[–] **danielvarga** 24 points 1 month ago

- For mathematicians: it uses Wasserstein distance instead of Jensen-Shannon divergence to compare distributions.
- For engineers: it gets rid of a few unnecessary logarithms, and clips weights.
- For others: it employs an art critic instead of a forgery expert.

permalink embed save report give gold reply

[–] **cbeak** 1 point 1 month ago

Why is it called critic rather than discriminator?

permalink embed save parent report give gold reply

[–] **ogrisel** 10 points 1 month ago

A forgery expert / discriminator would tell "I am 99.99999%" confident this is a fake Picasso. The "gradient" of that judgement would be very flat and therefore useless to help the generator improve.

An art critic would instead tell "I think this looks 2x better than the previous fake Picasso you showed to me (even though it still looks 5x worse than a real Picasso)". With non-zero gradient, the critic is better able to teach the generator in which direction it's likely to improve. The critic does not output probabilities of forgery, it outputs some unnormalized and therefore unbounded score.

permalink **embed** **save** **parent** **report** **give gold** **reply**

[–] **BananaCode** 2 points 1 month ago

It's not called a discriminator because its purpose is not to discriminate. It's an approximation to the Wasserstein distance. Why they called it critic I do not know.

permalink **embed** **save** **parent** **report** **give gold** **reply**

[–] **martinarjovsky** 7 points 29 days ago

Indeed it's not called a discriminator because its purpose is not to discriminate :)

We decided to call it a critic with actor critic methods in RL in mind. There, the actor (in our case the generator) is directly trained with the output of the critic as a reward, instead of passing it through another loss term. The name change is not to be taken too seriously though (we even still call it netD in the code), we just thought 'critic' was a broader term than discriminator for our case, and that writing it like that in the paper made the difference in the training procedure clearer.

permalink **embed** **save** **parent** **report** **give gold** **reply**

[–] **yield22** 1 point 10 days ago

For all: does this mean we should use WGAN instead (as default) from now on?

permalink **embed** **save** **parent** **report** **give gold** **reply**

[–] **ian_goodfellow** 24 points 1 month ago

"mode collapse comes from the fact that the optimal generator for a fixed discriminator is a sum of deltas on the points the discriminator assigns the highest values, as brilliantly observed by [11]"

This was actually known since the first GAN paper. I don't think [11] claim identifying this as a contribution. Their solution to that problem is very nice though.

You can see this claim in my slides, for example these: http://www.iangoodfellow.com/slides/2016-08-31-Berkeley.pdf "Fully optimizing the generator with the discriminator held constant results in mapping all points to the argmax of the discriminator"

It's worth mentioning that, depending on the structure of the discriminator, the set of points defining the argmax might not be isolated deltas, so the description in this paper isn't quite correct.

permalink **embed** **save** **report** **give gold** **reply**

[–] **martinarjovsky** 13 points 1 month ago

Thanks for the comments!

Ah, our bad on the credit assignment. We found out of this fact by the unrolled GAN paper, we will do a bit more literature review and update the text accordingly.

It's interesting what you say about the structure of the discriminator. The good thing is that we can check this in practice fairly easily! I'll just run a DCGAN on something like faces, fix the discriminator after 1, 5 or 20 epochs and train the generator for a while, see what happens.

permalink embed save parent report give gold reply

[–] **gheinrich** 2 points 1 month ago

Tried this some time ago. I guess results would depend on which particular state you leave the discriminator in but I noticed that the generator degenerated into a state where it would generate faces with extremely red lips, fierce eyes and fuzzy background. This actually looked pretty artistic, though not realistic.

permalink embed save parent report give gold reply

[–] **NotAlphaGo** 1 point 1 month ago

Would you mind explaining what you mean by checking this in practise? What's the observed behavior?

permalink embed save parent report give gold reply

[–] **JaejunYoo** 1 point 9 days ago

> It's worth mentioning that, depending on the structure of the discriminator, the set of points defining the argmax might not be isolated deltas, so the description in this paper isn't quite correct.

Could you elaborate on this? What kind of structures can the generator and discriminator be?

permalink embed save parent report give gold reply

[–] **ian_goodfellow** 3 points 9 days ago

Consider a function like $-x^{2\cdot}$ The argmax of this function is a single point, at x=0. Consider a function like $x^{2\cdot}$ The argmax of this function doesn't exist because the function increases without bound. Consider a function like $-(x-y)^{2\cdot}$ The argmax of this function is a line, not a countable set of isolated points. Depending on the parameters of the discriminator, it can have point a point argmax, an undefined argmax, or an argmax that contains uncountably many points.

permalink embed save parent report give gold reply

[–] **JaejunYoo** 1 point 8 days ago

Ah! Thank you very much for your quick reply!

permalink embed save parent report give gold reply

[–] **NotAlphaGo** 20 points 1 month ago

I guess here's the Repo:

https://github.com/martinarjovsky/WassersteinGAN

permalink embed save report give gold reply

[–] **rumblestiltsken** 38 points 1 month ago

Why is everyone talking about the maths? This has some pretty incredible contents:

- GAN loss that corresponds with image quality
- GAN loss that converges (decreasing loss actually means something), so you can actually tune your hyperparameters with something other than voodoo
- Stable gan training, where generator nets without batch norm, silly layer architectures and even straight up MLPs can generate decent images
- Way less mode collapse
- Theory about why it works and why the old methods had the problems we experienced. JS looks like a terrible choice in hindsight!

Can't wait to try this. Results are stunning

permalink embed save report give gold reply

[–] **ajmooch** [S] 11 points 1 month ago*

I've got an (I think) fairly faithful replication that's handling the UnrolledGAN toy MoG experiment with ease. Trying it out in my hybrid VAE/GAN framework on CelebA, we'll see how that goes.

permalink embed save parent report give gold reply

[–] **gwern** 5 points 1 month ago

I'm currently trying it on some anime images. The pre-repo version didn't get anywhere in 2 hours using 128px settings, but at least it didn't explode! I'm rerunning it with HEAD right now.

permalink embed save parent report give gold reply

[–] **NotAlphaGo** 6 points 1 month ago*

I'm trying it on grayscale images at 64px it gave me excellent results. Had to change the code a bit to allow single channel images but running smooth. Training 128px right now. Edit: I did ramp up my learning rate by factor 10.

permalink embed save parent report give gold reply

[–] **gwern** 5 points 1 month ago*

Interesting. I'll try changing the lrs too. EDIT: that definitely helps a ton so far: https://imgur.com/a/po73N http://imgur.com/a/SiSZ8 https://imgur.com/a/A5pdQ https://imgur.com/a/GZksh https://imgur.com/a/ARKxS

permalink embed save parent report give gold reply

[–] **NotAlphaGo** 2 points 1 month ago

I've definitely managed to get 128px to converge as well. Although my image training set is not your typical "lsun"

permalink embed save parent report give gold reply

[–] **gwern** 3 points 1 month ago

(Speaking of Danbooru, now I kinda want a 'lsundere' dataset...)

permalink embed save parent report give gold reply

[–] **cbeak** 1 point 1 month ago

Are you using https://github.com/255BITS/HyperGAN?

permalink embed save parent report give gold reply

[–] **gwern** 1 point 15 hours ago

No, WGAN. HyperGAN does look neat and supports a lot of stuff, though.

permalink embed save parent report give gold reply

[–] **gwern** 3 points 1 month ago*

BTW, one suggestion I've seen is that the whole 'watercolor' effect might be coming from RGB conflating brightness/color. That is, instead of working on RGB JPG images, the GANs might work better on a different colorspace like HSV, which avoid the GAN having to de/reconstruct it. I don't know much about how image encodings work; is WGAN agnostic about colorspace or would it have to be edited like you did? EDIT: I learned about a trick to avoid the library re-encoding into RGB - convert the RGB into HSB PNGs but leave it marked RGB, then train the GAN, and the samples can be converted back; this is very lossy as JPGs but seems to work ok if you convert to PNG instead. I'm training it now. It's working but it's hard to tell if it's

working *better*. EDITEDIT: after experimenting with the `--n_extra_layers` options, I now think that the watercolor/smearing is probably more due to convolution layers run amok without any fully-connected layers or bottlenecks like IllustrationGAN/StackGAN.

permalink embed save parent report give gold reply

[–] **MrEldritch** 3 points 1 month ago

If you're going to do that, why not go all the way and convert into one of the CIE color spaces? Those specifically attempt to approximate human color perception, so the GAN should have to reconstruct even less.

For instance, the CIE spaces attempt to be approximately perceptually uniform, so distance in the color space should more directly correspond to perceived difference* .This seems like an especially nice property to have in this sort of application.

* (It does better but not perfect, so there are also more complex norms than L2 distance that try and incorporate various factors, if you're into that.)

permalink embed save parent report give gold reply

[–] **gwern** 1 point 1 month ago*

That makes sense. I'd give CIE Lab a try since ImageMagick apparently supports it, but I'm kinda committed to the current run as WGAN doesn't (yet) support loading from checkpoints. EDIT: oh, actually it does. I just assumed it didn't because the original DCGAN code never did. That makes things much easier.

permalink embed save parent report give gold reply

[–] **r-sync** 1 point 1 month ago

would be interested to run this as well. where do you have the anime images from? is there a dataset i can download?

permalink embed save parent report give gold reply

[–] **gwern** 5 points 1 month ago

It's not a standard dataset*. I used a program called DanbooruDownloader to dump tags from Danbooru. To make it a bit easier on the GANs I've tried this out on, I start with just ~4k images of Asuka Soryu Langley from *Evangelion* (if the GAN doesn't pick up on her red hair/plugsuit within a few hours of training, that's a big... red flag) and then I switch to a bigger more generic dataset of ~70k downloaded from the tags `1girl` / `2girls` (again, limited for consistency - most `1girl` tags are portrait oriented, where a random selection of anime images would be far more diverse). The Asuka one, downscaled to 256x256px is ~200MB, and the full one unscaled is ~51GB.

So far I've never experienced good enough results on the simple Asuka dump to justify trying moving to the larger one :(Maybe this WGAN will finally do the trick - I'll have a better idea when I see where it gets overnight, which is around when most GAN implementations diverge or get stuck at 'fuzzy red blobs'.

* although I *have* looked extensively into the idea of turning Danbooru into a corpus for deep learning. It would be amazing for tag/multi-label classification, as none of the existing public datasets come anywhere close to it in terms of richness or thoroughness of annotation.

permalink embed save parent report give gold reply

[–] **trevinstein** 2 points 1 month ago

Your comment on anime reminded me of this repo: https://github.com/tdrussell/IllustrationGAN

permalink embed save parent report give gold reply

[–] **Sztefanol** 1 point 1 month ago

There was also Illustration2Vec which used images scraped from danbooru and pixiv: http://illustration2vec.net/

permalink embed save parent report give gold reply

[–] **Skylion007** 1 point 1 month ago

Awesome, I'm also working on a similar dataset, would love to chat with you about results.

permalink embed save parent report give gold reply

[–] **martinarjovsky** 1 point 1 month ago

This would be an awesome dataset!

permalink embed save parent report give gold reply

[–] **rumblestiltsken** 1 point 1 month ago

Well, that seems promising. Are you working in lasagne?

permalink embed save parent report give gold reply

[–] **ajmooch** [S] 7 points 1 month ago

Yep, I managed to get it running for DCGAN (it's literally just two lines of code changed), but because of the way my hybrid works (using the discriminator as encoder) there's still a bit of hyperparameter searching to be done before the Wasserstein version runs properly. I'll share my implementation in a GIST if people are curious.

permalink embed save parent report give gold reply

[–] **NotAlphaGo** 2 points 1 month ago

Yes please do!

permalink embed save parent report give gold reply

[–] **rumblestiltsken** 1 point 1 month ago

More code online never hurts :)

permalink embed save parent report give gold reply

[–] **tmiano** 4 points 29 days ago

I had to come back to this thread because I'm amazed people aren't talking about this result more. Maybe we're trying to not be too optimistic and be disappointed later. I have to say though, my results with this so far have been really impressive. It's not just way less mode collapse, it's no mode collapse at all. And even when your hyperparameters are poorly tuned, the worst thing that seems to happen is your loss oscillates wildly, but actually the samples continue to get better despite this.

Are there reasons not to be excited about this? Besides a few twitter discussions I'm not seeing a lot of people talk about it much yet.

permalink embed save parent report give gold reply

[–] **rumblestiltsken** 2 points 27 days ago

Yeah, I totally agree. How is this not the biggest thing? I haven't seen any reason for disinterest.

permalink embed save parent report give gold reply

[–] **tmiano** 4 points 17 days ago

Now that I've had the chance to play around with them a bit more, I've seen a couple of things that could temper the excitement about them: 1) Long training times, require the learning rate to be small and many critic updates per generator update. 2) Samples aren't quite as crisp and realistic as they tend to be in the original GAN formulation. 3) Still suffer from instability when learning rate, clipping parameter, and critic updates are not fine tuned.

Still, it seems to show that the problem of mode collapse in GANs might not be as difficult to solve as previously thought.

permalink embed save parent report give gold reply

[–] **feedthecreed** 3 points 1 month ago

> Theory about why it works and why the old methods had the problems we experienced. JS looks like a terrible choice in hindsight!

Was JS ever being optimized in standard GANs? As far as I remember, JS was just used for the proof of the global minima being correct.

permalink embed save parent report give gold reply

[–] **AlexCoventry** 1 point 1 month ago

How do you separate the maths from the "Theory about why it works"?

permalink embed save parent report give gold reply

[–] **ogrisel** 3 points 1 month ago*

The paper is actually very good at stating the intuition behind the main results without having to understand the technical details of the proofs.

permalink embed save parent report give gold reply

[–] **rumblestiltsken** 1 point 1 month ago

Nothing wrong with talking about the maths ... but no-one was talking about the implications. Seemed a strange response to the results in the paper.

permalink embed save parent report give gold reply

[–] **ian_goodfellow** 11 points 1 month ago

When this paper refers to "regular GAN" does it mean the minimax formulation of GANs? That's the way I'm reading it, but not actually sure.

To use the terminology of my tutorial (https://arxiv.org/pdf/1701.00160v3.pdf), does this paper use "regular GAN" to refer to GANs trained with Eq 10 or with Eq 13?

I think it is using Eq 10, but I would usually consider Eq 13 to be the "regular GAN." Eq 10 is nice for theoretical analysis but Eq 13 is what I recommend that people actually implement.

permalink embed save report give gold reply

[–] **martinarjovsky** 4 points 1 month ago

Hi! We'll try to make the text clearer :)

All the normal GANs in the paper were trained using Eq 13 of your tutorial, as is (as far as we are aware) typically done.

I think the connection you mention is super interesting. If the cost of the generator is linear on the logits that would be an equivalent cost to the generator, which is really cool. That being said I think the way we train the discrimator is different, and the clipping of the weights makes a very big difference as well (otherwise there is no notion of optimality for example).

In any case it's very interesting! We'll explore a bit more this connection for the ICML version. Thanks for the valuable insights.

permalink embed save parent report give gold reply

[–] **ian_goodfellow** 3 points 1 month ago

Based on Fig 2 it looks like "regular GAN" definitely means Eq 10. If Eq 13 were used, the cost for G would be linearly increasing in the right half of the plot, the same as it is for WGAN.

permalink embed save parent report give gold reply

> [–] **ian_goodfellow** 9 points 1 month ago
>
> I think in terms of the cost applied to the generator, the approach in this paper is equivalent to taking the average of Eq 10 and Eq 13. That's something we tried while writing the original GAN paper and you can see it in our public code: https://github.com/goodfeli/adversarial/blob/27eac0351588f486c11fbe7fe88a17e4a1aa4888/__init__.py#L362 The equivalence follows because softplus(x) - softplus(-x) = x, so blending Eq 10 and Eq 13 with equal coefficients gives a cost that is linear in the logits of the discriminator.
>
> What's new in this paper is to make the cost purely linear for the discriminator too.
>
> **permalink embed save parent report give gold reply**

[–] **ian_goodfellow** 2 points 1 month ago

OK, I see that "regular GAN" refers to Eq 10 and "the -log D trick" is Eq 13.

permalink embed save parent report give gold reply

[–] **gpeyre** 9 points 1 month ago

This is a great paper with a great idea. Using a modified W1 loss is very nice, and plugging a deep network to search for the dual potential will probably become a landmark idea.

I believe this previous work of Marco Cuturi is very relevant: https://papers.nips.cc/paper/6248-wasserstein-training-of-restricted-boltzmann-machines It introduces the idea of using W-distances in place of MLE/KL in order to cope with singular distributions, and discusses its use in the context of training deep-architecture as well.

This even more recent paper https://arxiv.org/abs/1701.05146 discusses some theoretical aspect (in particular consistency of the estimator) and uses the same generative framework (a parametric mapping applied to some fixed distribution).

Congratulation for this nice contribution.

permalink embed save report give gold reply

[–] **galapag0** 6 points 1 month ago

Why you cannot remove batch normalization from the critic even using Wasserstein GAN?

permalink embed save report give gold reply

> [–] **martinarjovsky** 5 points 1 month ago
>
> We're working on it! It seems that since the loss in the critic or discriminator is very nonstationary (note that as you change the generator, the loss for the disc changes), something that reduces covariate shift such as batchnorm is necessary to use nontrivial learning rates.
>
> We are however exploring different alternatives, such as weightnorm and such (which for WGANs make perfect sense, since that would naturally allow us to have weights lie in a compact space, without even need for clipping). We hope to have more on this for the ICML version.
>
> **permalink embed save parent report give gold reply**

> [–] **LucasUzal** 5 points 1 month ago

You used BN on the critic in all the experiments although it do not satisfy conditions for f in equation 2. It seems to me that BN dramatically changes WGAN formulation (please, let me know if I am wrong). I would like to see a more detailed discussion about BN in the next manuscript version and some comparative results with and without BN.

permalink embed save parent report give gold reply

[–] **martinarjovsky** 4 points 1 month ago

The role that batchnorm takes is fairly complicated, and we are still not sure what to make of it. Remember however that practical implementations have (x - mu(x)) / (std(x) + epsilon). The epsilon in there allows it to still have the theoretical formulation trivially. There is some evidence that batchnorm actually approximates a 1-Lipschitz constraint by itself, but we are not sure how true this is. In any case we are working on taking out batchnorm and the clipping and putting something like weightnorm (with the g terms fixed) that would take care of this problem. This hopefully should be ready by the ICML version.

permalink embed save parent report give gold reply

[–] **dylanbyte** 1 point 1 month ago

A finite sample of size n with zero mean with bounded empirical variance is itself bounded no? So batchnorm bounds the layer with fixed minibatch size.

permalink embed save parent report give gold reply

[–] **martinarjovsky** 4 points 1 month ago

After some tiny math I think we know why batchnorm can't cause a break in the lipschitzness (which we already saw in practice). I promise to add this later properly in the paper.

If $V(x)^{1/2} > c$ (the variance is bounded below), then this c becomes a Lipschitz constant that's independent of the model parameters, so we are good to go. For V(x) to not be bounded below as training progresses, it has to go arbitrarily close to 0. In this case, X has to converge to it's mean, so the term X - E[X] in the numerator of batchnorm will go to 0, and therefore (X - E[X]) / $(V[X]^{1/2}$ + epsilon) comes 0 (which is obviously 1-Lipschitz). This will also further render the activation X inactive, which the network has no incentive to do, and explains why we didn't see this kind of behaviour.

permalink embed save parent report give gold reply

[–] **LucasUzal** 1 point 1 month ago

That is a interesting analysis about the lipschitzness of f. But I still worry that f is not a function X->R, since it is not applyed to a single sample but to the whole minibach. What consequences does this have on the estimation of the expected values in the equation 2?

permalink embed save parent report give gold reply

[–] **matcha-ai** 1 point 1 month ago

Hi, great work! Naive question: How would you explain this whole lipschitz thing and its application here, to a software engineer?

permalink embed save parent report give gold reply

[–] **ppwwyyxx** 1 point 1 month ago

One important reason why it doesn't work without BN seems to be that you clip the w & b of the affine transformation inside BN as well! So the network actually scales

> down the whole activation to about 1%. Otherwise I found the loss growing several magnitudes higher.
>
> It's not clear to me why w & b in BN also needs to be clipped to such a small value. Usually w is around 1 and b is around 0 which makes it nearly an identical mapping, and won't change the Lipschitz continuity.
>
> **permalink embed save parent report give gold reply**

[–] **galapag0** 3 points 1 month ago

Uhm, interesting! Btw, in the official code, you seem to disable batch normalization for the generation and the critic. Can you clarify if this is the parameters used in the paper or we should enable batch normalization in the critic? (thanks a lot for sharing code!)

permalink embed save parent report give gold reply

> [–] **martinarjovsky** 5 points 1 month ago
>
> Oops, nice spotting. The nobn in the paper is only on the generator, we mistakenly took it out on both in this version of the code. Edit: Code is now fixed.
>
> **permalink embed save parent report give gold reply**

[–] **ogrisel** 2 points 1 month ago*

It would be nice if you could also try the normalizers investigated there:

Normalizing the Normalizers: Comparing and Extending Network Normalization Schemes Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H. Sinz, Richard S. Zemel

https://arxiv.org/abs/1611.04520

permalink embed save parent report give gold reply

[–] **danielvarga** 7 points 1 month ago

More than one year ago I've created a model I've called the Earth Moving Generative Net. It optimized an empirical approximation to the Wasserstein distance:

https://github.com/danielvarga/earth-moving-generative-net

It worked fine on MNIST, but it did not scale much further. Poor thing had no chance, it couldn't exploit Kantorovich-Rubinstein duality.

permalink embed save report give gold reply

[–] **Imnimo** 11 points 1 month ago

Section 2 really kicked my ass, so forgive me if this is a stupid question. When I look at the algorithm in this paper, and compare it to the algorithm in the original GAN paper, it seems there are a few differences:

1) The loss function is of the same form, but no longer includes a log on the outputs of the discriminator/critic.

2) The discriminator/critic is trained several steps to approximate-optimality between each generator update.

3) The weights of the discriminator/critic are clamped to a small neighborhood around 0.

4) RMSProp is used instead of other gradient descent schemes.

Is that really all there is to it? That seems pretty straightforward and understandable, and so I'm worried I've missed something critical.

permalink embed save report give gold reply

> [–] **AlexCoventry** 11 points 1 month ago*
>
> > I'm worried I've missed something critical.

I think you're roughly right about the changes to the algorithm. (1) The loss function is no longer a probability, so it's not just that you're no longer taking the log (which in expectation optimized for the total probability of the discriminator's training examples.) In addition, the complement is no longer taken for the generator examples. The key idea is that the difference of expectations is an estimate of the Wasserstein distance between the generator distribution and the "real" one. (3) I believe the weights are clamped to impose a Lipschitz constant on the function the discriminator is now approximating, because the Wasserstein distance is a max over Lipschitz functions.

(2) They're able to train the discriminator closer to optimality because the Wasserstein distance imposes a weaker topology on the space of probability measures (see figure 1, and compare figures 3 vs 4). With the standard total probability loss function, if the discriminator gets too good it just rejects everything the generator tries, and there's no gradient for the generator to learn from.

permalink embed save parent report give gold reply

[–] **r-sync** 14 points 1 month ago*

1. The last layer of the critic is to take the mean over the mini-batch to give an output of size 1. Then you backward with all ones (or all -ones).
2. There is no sigmoid / log at the end of the critic
3. the weights of the critic are clamped within a bound around 0.
4. Using RMSProp is a detail that's not super important, it speeds up training but even SGD will converge (switching on momentum schemes will make it slightly unstable due to the nature of GANs).

Here's quick code that implements Wasserstein GANs in PyTorch, we'll release a proper repo later

Edit: proper repo: https://github.com/martinarjovsky/WassersteinGAN

permalink embed save parent report give gold reply

[–] **Imnimo** 5 points 1 month ago

Is there a difference between considering the mean over the mini-batch as a layer of the critic vs. considering it as part of the loss function?

I guess the other part I don't quite grasp is which part constitutes the Wassertein distance. I see all this stuff about infimums and supremums in the theoretical section (and I haven't really wrapped my head around it yet) and I was expecting to come across a complicated loss function, but then it turns out to be in a sense even simpler than the original GAN loss.

permalink embed save parent report give gold reply

[–] **martinarjovsky** 10 points 1 month ago

It's the same, this way it was just easier to implement :)

There are two elements that make the transition to the Wasserstein distance.

- Taking out the sigmoid in the discriminator, and the difference between the means (equation 2). While it's super simple in the end, and looks quite similar to a normal GAN, there are some fundamental differences. The outputs are no longer probabilities, and the loss now has nothing to do with classification. The critic now is just a function that tries to have (in expectation) low values in the fake data, and high values in the real data. If it can't, it's because the two distributions are indeed similar.
- The weight clipping. This constraints how fast the critic can grow. If two samples are close, therefore the critic will have no option than to have values that are

close for them. In a normal GAN, if you train the discriminator well, it will learn to put a 0 on fake and a 1 on real, regardless of how close they are, as long as their not the same point.

In the end, the flavor in here is much more 'geometric' than 'probabilistic', as before. It's not of differentiating real from fake. It's about having high values on real, and low values on fake. Since how much you can grow is constrained by the clipping, as samples get closer this difference will shrink. This is much more related to the concept of 'distance' between samples, than to the probability of being from one distribution or the other.

permalink embed save parent report give gold reply

[–] **dunomaybe** 2 points 1 month ago

Have you tried experiments testing the empirical performance with only subsets of 1-3?

permalink embed save parent report give gold reply

[–] **davikrehalt** 2 points 1 month ago*

Have you tried adding regularization terms to the network instead of clipping weights, it seems to me what you want is bounded gradients, so couldn't you do that in a more natural way?

I'm assuming you are one of the authors, so congratulations on the results.

permalink embed save parent report give gold reply

[–] **AlexCoventry** 2 points 1 month ago

> The last layer of the critic is to take the mean over the mini-batch to give an output of size 1. Then you backward with all ones (or all -ones).

What part of the paper does this correspond to?

permalink embed save parent report give gold reply

[–] **martinarjovsky** 2 points 1 month ago

This corresponds to the fact that now the loss of the generator is just - the output of the critic when it has the mean as the last layer (hence backproping with -ones).

permalink embed save parent report give gold reply

[–] **AlexCoventry** 1 point 1 month ago

Thanks. What's the purpose of taking the mean, though?

Also, why ones? Why not drive the generator->discriminator outputs as low as possible, and the real->discriminator outputs as high as possible?

permalink embed save parent report give gold reply

[–] **ajmooch** [S] 2 points 1 month ago

> Also, why ones? Why not drive the generator->discriminator outputs as low as possible, and the real->discriminator outputs as high as possible?

As far as I can tell, you're backpropping the ones as the gradient (the equivalent of theano's known_grads), which is just the equivalent of saying "regardless of what your output value is, increase it," basically meaning that the value of the loss function doesn't really affect its gradient. You could presumably backpropagate higher values (twos, or even the recently proposed theoretical number THREE) but that feels like we're getting into a hyperparameter choice--if you double the gradient at the output, how different is that from increasing the learning rate? Might be something to explore, but it doesn't really feel like it to me.

permalink embed save parent report give gold reply

[–] **AlexCoventry** 1 point 1 month ago

Ah, I was reading the "ones" as a target, not as a gradient. Thanks.

permalink embed save parent report give gold reply

[–] **JaejunYoo** 1 point 15 days ago

Still having hard time to understand this issue. If I am getting right, it seems like that you are saying that the **line10 of the algorithm1 (in the original paper) is actually just a "ones" or "tf.neg(ones)" vector.** Am I right? How this can be understood based on the pseudo code they provided? Any help please? I tried to see the github codes of WGAN implementation still having hard time to figure out this issue. I guess the link which /u/dismal_denizen kindly provided us has some related parts in implementation section: - 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) though I cannot get the intuition from those....

And..this is a minor question: why is the figure 2(for comparing the gradients of GAN and WGAN) in the original paper shows the WGAN critic values oppositely? I thought that the value should be high in the real density and small in the other. What am I getting wrong here?

permalink embed save parent report give gold reply

[–] **dismal_denizen** 2 points 13 days ago

My implementation trick with AbsCriterion is just meant as a way of implementing constant gradients in Torch. By setting the target to be output +- 1, we guarantee that the gradient will be +- 1.

My initial reason for using +- 1 gradients was pretty much that the comments here say to. However, I was able to muster some intuition for it. If you look at Eq. 3 in the paper, you will notice that the maximum is reached when f_w(x) yields high values for real examples and low values for generated examples. This is exactly what we are working towards by pushing the critic up and down with +-1 gradients.

permalink embed save parent report give gold reply

[–] **JaejunYoo** 1 point 9 days ago

THANK YOU VERY MUCH. I got your point. So.. basically it is due to the details required for Torch implementation. I was confused because I was not familiar with Torch. Thank you for your quick reply again.

permalink embed save parent report give gold reply

[–] **NotAlphaGo** 1 point 1 month ago

Thank you for the repo!

Has anyone put this through an MNIST test?

permalink embed save parent report give gold reply

[–] **fogandafterimages** 1 point 1 month ago

It seems that with respect to your point 3), it's the symmetric weight clamping that's important, and the magnitude of the range is completely arbitrary. The range used in the paper looks like it was chosen for numerical stability rather than theoretically motivated.

permalink embed save parent report give gold reply

[–] **martinarjovsky** 4 points 1 month ago

Yep, larger clipping values simply took longer to train the critic.

That being said it might be that higher clipping values increase the capacity in nontrivial nonlinear ways, which might be helpful, but we don't yet have full empirical conclusions on this.

permalink embed save parent report give gold reply

[–] **davikrehalt** 5 points 1 month ago

Here's what I think the main insight of this paper is, we should train Lipschitz functions of fixed constant to maximize the expected difference on predicted values and true values. I haven't gone through the maths behind the niceness of the new metric, but in the mean time I think this insight is pretty significant. Limiting space of possible discriminators can automatically improve training.

permalink embed save report give gold reply

[–] **davikrehalt** 6 points 1 month ago

Maybe Lipschitz here is the wrong word, more precisely differentiable functions with bounded derivative automatically fixes gradient problems. I'm curious though, the authors limit their weights to small values, but I suspect a strong regularization term can do this better, regularize the Maximal gradient back propagation

permalink embed save parent report give gold reply

[–] **martinarjovsky** 11 points 1 month ago*

Hi! Very insightful comment, a few follow ups

a) A differentiable function f is K-Lipschitz if and only if it is differentiable and has derivatives with norm bounded by K everywhere, so the two views are equivalent :)

b) We thought about regularizing instead of constraining. However, we really didn't want to penalize weights (or gradients) that are close to the constraint.

The reason for this is that the 'perfect critic' f that maximizes equation 2 in the paper has actually gradients with norm 1 almost everywhere (this is a side effect of the Kantorovich-Rubinstein duality proof and can be seen for example in Villani's book, Theorem 5.10 (ii) (c) and (iii)). Therefore, in order to get a good approximation of f we shouldn't really penalize having larger gradients, just constrain them :)

That being said it still remains to be seen which is better in practice, whether to regularize or constrain.

permalink embed save parent report give gold reply

[–] **davikrehalt** 5 points 1 month ago

Thanks for the reply, you have good point that your implementation would be a better approximation of the metric you named, however I still think it should be tried, my interpretation is that bounded gradients increases the "width" of the decision boundary, so maybe just putting a cost on it is enough

permalink embed save parent report give gold reply

[–] **martinarjovsky** 3 points 1 month ago

I agree :)

permalink embed save parent report give gold reply

[–] **AnvaMiba** 1 point 1 month ago

Is it necessary to use a box constraint on the weight, or would norm constraints (global, per-layer or per-row) also work?

permalink embed save parent report give gold reply

[–] **Thomjazz** 1 point 1 month ago*

Hi, thank you for this fascinating paper! Super insightful. I love how the Wasserstein distance fits so nicely in the GAN framework so that the WGAN critic provide a natural lower bound on the EMD.

Reading this discussion on regularization vs. constrain, I was wondering whether a more natural way to force the function parametrized by the critic to be K-Lipschitz would not be to directly add a cost penalty over the function derivatives in the loss function.

For instance in the form of a term sum(alpha*(abs(f'(x)) - 1)) over the data points.

permalink embed save parent report give gold reply

[–] **davikrehalt** 1 point 1 month ago

I think this result is quite fundamental, constraining the neutral network to have bounded gradients appear in other applications right? Didn't deepmind use something similar before? I don't think the derivation from the EM metric is the only justification, or even the most intuitive one.

permalink embed save parent report give gold reply

[–] **to4life2** 3 points 1 month ago*

So...

Drop the logs?

Edit - and drop those sigmoids too.

I was being kind of sarcastic, obviously this is a great work and has some real theoretical "meat" behind it, which in turn led to fantastic empirical results.

A few thoughts --

1) This work reminds me of when DNN practitioners switched from sigmoid activations to ReLUs. Training just got a lot better (faster). Maybe we don't need such fancy "activations" because multiplying a bunch of matrices in sequence already gives a lot of room to learn exotic functions and mappings.

2) The BCE (binary crossentropy) loss metric may not be great for certain problems where you want output decisions to have some *space* between right and wrong (real and fake). Reading Goodfellow's tutorial and looking at figure 16 pg. 26 (https://arxiv.org/pdf/1701.00160v3.pdf) I couldn't help but think to myself: that looks hard to learn! Those are some unforgiving gradients as your generator is trying to "cross the chasm" between fake and real. So while BCE was probably a good theoretical choice to describe the game, we see that it's tough to train in practice.

permalink embed save report give gold reply

[–] **dismal_denizen** 3 points 1 month ago

Here are my notes on the paper, *caveat emptor*: https://paper.dropbox.com/doc/Wasserstein-GAN-GvU0p2V9ThzdwY3BbhoP7

permalink embed save report give gold reply

[–] **ajmooch** [S] 3 points 1 month ago

Your thoughts mirror mine surprisingly well--I'm going to run some SVHN experiments later this week to see how it stacks up on the semi-supervised case, but thus far I've had trouble using D as the encoder for my VAE hybrid. One of the things I've noticed is that because you have improved mode-spreading behavior, sample quality from a DCGAN architecture is actually *worse* because you've lost the "weaponized overfitting" that you normally see. Beefing up the model solves this (and is also a lot, LOT easier to do than normal.

permalink embed save parent report give gold reply

[–] **spongebob_jy** 4 points 1 month ago

I am supposed to know Wasserstein distance a bit more than averaged audiences. I have two questions:

1. The authors use a parametric family of functions searching for the dual potential of optimal transport using a batch based estimator. There is of course something nice about it: The loss becomes more computational tractable. But there are very little justifications to approximate the Wasserstein distance in this way. The approximation quality then heavily depends on the choice of the structure of discriminator (or dual potential approximator). I personally wouldn't consider it proper to use the name "Wasserstein". The RKHS distance or more broadly MMD is also in the same form.
2. What worries me most is the proposed GAN architecture may never be successfully trained with a *true* Wasserstein loss under the batch mode. The central idea of Wasserstein loss is the matching between two sets of full samples. The batch-wise estimator is so rough and biased that it has to work with some regularizations. Constraining the dual potentials to a particular neural network somehow acts as the effect of regularization. Given these, it probably unexpectedly happens to make GAN trained properly.

Anyway, it could be a good work for GAN community. I am happy to see it attracts more people to look at the Wasserstein distance, an emerging area for machine learning.

permalink embed save report give gold reply

[–] **mcuturi** 6 points 1 month ago

Well, as the saying goes, it does not matter if the cat is white or black, as long as it catches the mouse! The authors of this paper have proposed a new and interesting way to approximate the W metric that's interesting in its own right.

I agree with a previous comment by Gabriel that the idea of using Wasserstein for parameter estimation, within a GAN framework or not, is not new.

However, when comparing distributions (from a generative model to real data) under the Wasserstein metric, the key to differentiate (to make it smaller) that Wasserstein loss is to use the dual W problem, and more precisely the potential function f the paper discusses. In the case of the W_1 that dual problem is even simpler, and there is indeed an "integral probability metric" flavor that W_1 shares with other metrics, notably the MMD,

(connexions have been nicely summarized here: www.gatsby.ucl.ac.uk/~gretton/papers/SriFukGreSchetal12.pdf)

but in my opinion the similarities with MMD stop there. I have always found the IPM interpretation of the MMD overblown: because MMD boils down, in an IPM framework, to a quadratic problem that has a closed form solution, we're never really "optimizing" over anything with MMD. It's essentially a cosmetic addition, much like saying that the mean of a few vectors corresponds to the argmin_x sum $|x_i-x|^2$ .... It's interesting to write it in a variational approach precisely because you are looking for problems that cannot be written with a closed form solution.

In the W case things are indeed much tougher computationally.

We've tried other approaches to solve exactly the same problem of W estimation in this NIPS paper, and reached the same problems of estimating accurately the dual potential:

https://papers.nips.cc/paper/6248-wasserstein-training-of-restricted-boltzmann-machines

in that paper we tried to approximate dual potentials with an entropic regularization, which has also been further studied in a stochastic setting in another NIPS paper

https://papers.nips.cc/paper/6566-stochastic-optimization-for-large-scale-optimal-transport

The main innovation of this "Wasserstein GAN" paper lies in a very clever way of proposing that the dual potential, constrained to be 1-Lipschitz, can be approximated with a neural net.

permalink embed save parent report give gold reply

[–] **spongebob_jy** 1 point 29 days ago

I agree with almost all what you said, specially "we're never really "optimizing" over anything with MMD" and "The main innovation of this "Wasserstein GAN" paper lies in a very clever way of proposing that the dual potential can be approximated with a neural net.". Thanks for your clarifications.

I also noticed that the Wasserstein RBM paper is optimizing things in full batch mode, while the Wasserstein GAN paper's training strategy seems more scalable due to the use of small batches.

permalink embed save parent report give gold reply

[–] **mcuturi** 1 point 29 days ago

indeed... use of mini batches to optimize a dual potential (in the more general case than that addressed in the paper, and with a regularization if needed) is discussed in the "discrete / discrete" section of the stochastic approach we proposed to approximate W

https://papers.nips.cc/paper/6566-stochastic-optimization-for-large-scale-optimal-transport

permalink embed save parent report give gold reply

[–] **spongebob_jy** 2 points 28 days ago*

That's right. But I feel the approach in Wasserstein GAN may still be more scalable, because it samples mini-batch from both distributions (real one and generative one). In comparison, the "discrete / discrete" setting of your paper can only sample mini-batch at one side, and keep the other side as full batch.

Of course, I think the entropy regularization has an edge in terms of accuracy and generality. The approach in the Wasserstein GAN paper does not have any backups for how accurate it could be and only deals with W_1. I do believe using a neural net to approximate the dual potential is promising, and more research should be done to analyze this approach not from the GAN context but from the general optimal transport context.

permalink embed save parent report give gold reply

[–] **mcuturi** 2 points 26 days ago

I agree with your second comment, using NN to approximate dual functions is promising. In our "stochastic optimization for large scale OT" paper, we do address in the last section a purely sample based approach in which we approximate the potential functions of both measures. Maybe that would answer your first point above. We use RKHS though, not NN in that part. It might be a

good idea to use NN to approximate both dual potentials when the cost is not simply a distance. I also believe the entropy regularizer is more consistent with what you might want to do. As in the Wasserstein GAN paper shows (and as you pointed out!) it's not easy to constrain a function to be 1-Lipschitz!! The hacks presented in the Wasserstein GAN paper only "kind of" do that.

permalink embed save parent report give gold reply

[–] **Thomjazz** 1 point 26 days ago

Very interesting, thanks for your comment ! (and thanks for your papers and work on Wasserstein distances that I discover now)

permalink embed save parent report give gold reply

[–] **arXiv_abstract_bot** 7 points 1 month ago

Title: Wasserstein GAN

Authors: Martin Arjovsky, Soumith Chintala, Léon Bottou

> Abstract: We introduce a new algorithm named WGAN, an alternative to traditional GAN training. In this new model, we show that we can improve the stability of learning, get rid of problems like mode collapse, and provide meaningful learning curves useful for debugging and hyperparameter searches. Furthermore, we show that the corresponding optimization problem is sound, and provide extensive theoretical work highlighting the deep connections to other distances between distributions.

PDF link Landing page

permalink embed save report give gold reply

[–] **darkconfidantislife** 9 points 1 month ago

My head spun several times with the Borel sets, so excuse my idiot questions...

But the main point is using EM (or Wasserstein-1) distance in lieu of KL-divergence as the measure of distance between probability distributions to minimize, correct?

permalink embed save report give gold reply

[–] **AlexCoventry** 6 points 1 month ago

Yes.

permalink embed save parent report give gold reply

[–] **darkconfidantislife** 3 points 1 month ago

Thanks :)

permalink embed save parent report give gold reply

[–] **ogrisel** 2 points 1 month ago*

Thanks /u/martinarjovsky for this excellent paper. I found it very educational and enlightening.

Is there any theoretical guidance or practical trick to detect when the critic capacity is too low to get an optimal approximation? Can the critic ever be too strong (leading to some sort of overfitting of the critic itself)? Or is just a matter of computational constraints?

Looking forward to reading your results about the study of the unsuitability of momentum based optimizers.

In Appendix A, when you introduce \delta the Total Variance distances, I think you miss TV as a subscript of the norm (as at this point you are still referring to the TV norm and not yet to the dual norm):

```
\delta(\mathbb{P}_r, \mathbb{P}_\theta) := ||\mathbb{P}_r - \mathbb{P}_\theta||_{T
```

permalink embed save report give gold reply

[–] **ogrisel** 2 points 1 month ago

Also other question: how much is weight clipping important in practice and in particular the what is the impact of changing the magnitude of the clipping parameter. That is, how much is it a problem to allow for a larger Lipschitz constant? Have you made any experiment to investigate this?

Would "soft-clipping" via an L2 regularizer on the weights work too?

permalink embed save parent report give gold reply

[–] **martinarjovsky** 3 points 29 days ago

Hi! Thanks, I'm glad you liked the paper :).

Theoretical guidance for when the critic capacity is too low for now is complicated to say, since the capacity of neural nets is hard to quantify, it has a lot to do with the specific problem and the inductive bias by the architecture. That being said I found that the capacity of the disc is usually too low when I see sign changes in the estimate of equation 2 (this means the critic's error is close to 0, so it's either not well trained till optimality or it's capacity is no longer good enough). I think net2net ideas might come pretty useful here eventually.

Woops, thanks for the typo! I'll add the TV :)

The weight clipping parameter is not massively important in practice, but more investigation is required. Here are the effects of having larger clipping parameter c:

- The discriminator takes longer to train, since it has to saturate some weights at a larger value. This means that you can be a risk of having an insufficiently trained critic, which can provide bad estimates and gradients. Sometimes sign changes are required in the critic, and going from c to -c on some weights will take longer. If the generator is updated in the middle of this process the gradient can be pretty bad.
- The capacity is increased, which helps the optimaly trained disc provide better gradients.

In general it seems that lower clipping is more stable, but higher clipping gives a better model if the critic is well trained.

permalink embed save parent report give gold reply

[–] **ogrisel** 1 point 29 days ago

Thanks for your reply.

Do you think it would be a good idea to combine the feedback of an ensemble of critics with different capacities? Or could the lowest capacity critic be detrimental? I like the idea of net2net. But then the critic capacity scheduling might be tricky to get right.

Also in the paper I don't understand why the mode collapse issue should disappear with WGAN. This is not really guaranteed, right?

Have you tried to run experiments in a semi-supervised setting? I guess this would have been too much for a first paper on Wasserstein GANs but I would be interested in the results.

permalink embed save parent report give gold reply

[–] **alexmlamb** 2 points 15 days ago

W-GANs are just a special case of LambGANs which implements GAN(x)*alpha* + *WGAN(x)*(1-alpha). W-GANs are just the trivial alpha=0 case whereas LambGANs work for not just an infinite set of alpha values but an uncountably infinite set of alphas. That means that the degree to which

LambGAN is more general than W-GAN is infinitely greater than the number of atoms in the universe.

permalink embed save report give gold reply

[–] **feedthecreed** 2 points 1 month ago

I found the introduction of this paper difficult to understand. What is the noise term they're referring to that plagues models where the likelihood is computed?

Also, what terms are they referring to in this part:

> Because VAEs focus on the approximate likelihood of the examples, they share the limitation of the standard models and need to fiddle with additional noise terms.

permalink embed save report give gold reply

[–] **AlexCoventry** 3 points 1 month ago*

> What is the noise term they're referring to that plagues models where the likelihood is computed?

The support for the "real" distribution P_r lies on a submanifold, and KL(P_r||P_θ) will be ~~zero~~ infinite unless your learning algorithm nails that submanifold, plus such measures are a pain to parameterize. So instead they model a "blurred" version of P_r. Generatively speaking, first they draw a sample $z \sim P_r$, then they apply some Gaussian noise, $z+\epsilon$ for $\epsilon \sim N(0,\sigma)$. The distribution of this blurred version has support on all of $\mathbb{R}^n$, so KL is a sensible comparison metric.

permalink embed save parent report give gold reply

[–] **feedthecreed** 1 point 1 month ago

> KL(P_r||P_θ) will be zero

Wouldn't KL(P_r||P_θ) be infinite if P_θ doesn't nail the submanifold. Why would it be zero?

permalink embed save parent report give gold reply

[–] **AlexCoventry** 2 points 1 month ago

Thanks, I meant infinite.

permalink embed save parent report give gold reply

[–] **feedthecreed** 1 point 1 month ago

Thanks for explaining, does that mean maximum likelihood isn't a meaningful metric if your model support doesn't match the support of the "real" distribution?

permalink embed save parent report give gold reply

[–] **AlexCoventry** 2 points 1 month ago

If your model, at almost all points in its parameter space, expresses probability measures in which the real data has zero probability, then you don't get gradients you can learn from.

permalink embed save parent report give gold reply

[–] [deleted] 1 month ago

[deleted]

[–] **AlexCoventry** 1 point 1 month ago

Suppose your model is the family of distributions (θ, Z), like example 1 in the paper, and the target distribution is (0, Z). So your training data is going to be {(0, y_1), …, (0, y□)}, and for any non-zero θ, all your training data is going to have probability 0, and the total probability is going to be locally constant and 0. Since the gradient of the total

probability is 0, you can't use standard learning methods to move towards (0, Z) from any other (θ, Z).

permalink **embed** **save** **report** **give gold** **reply**

[–] [deleted] 1 month ago

[deleted]

[–] **AlexCoventry** 1 point 1 month ago

> Doesn't this assume that the discriminator is perfect?

Can you expand on that? Discriminators (in the GAN sense, at least) haven't really entered into the discussion in this subthread.

permalink **embed** **save** **report** **give gold** **reply**

continue this thread

[–] **PURELY_TO_VOTE** 1 point 1 month ago

Wait, what manifold is the real distribution a submanifold of? Do you mean that the real distribution's support is a manifold embedded in the much higher dimensional space of the input?

Also, won't KL(P_r||P_θ) be 0? Or is the fear that P_θ is exactly 0 some place that P_r isn't?

permalink **embed** **save** **parent** **report** **give gold** **reply**

[–] **AlexCoventry** 1 point 1 month ago

See my other subthread with feedthecreed

permalink **embed** **save** **parent** **report** **give gold** **reply**

[–] **jrkirby** 3 points 1 month ago

I found the mathematics in this paper opaque. Perhaps it's getting at a profound point, but I wasn't able to glean knowledge. After reading the wiki for borel sets and understanding very little, I realized it would probably take me several hours to understand the crux of the paper, where I gave up.

permalink **embed** **save** **report** **give gold** **reply**

[–] **dylanbyte** 12 points 1 month ago

I think any set a sane person would think of is Borel, so I would just substitute Borel set for set.

permalink **embed** **save** **parent** **report** **give gold** **reply**

[–] **serge_cell** 5 points 1 month ago*

This is not entirely correct (c). It depends on what you count as open sets (that is topology of the space). If open sets category is restricted enough some simple sets will not be borel sets. For example define open sets as set which include all rectangles (m, m+1) X (n, n+1), empty set and R^2 and all their unions. In this case simple circle of radius 1 is non-borel set. This is not an abstract nonsense, but our knowledge about of probabilities of events. Sigma-algebras explore this subjects in depth.

permalink **embed** **save** **parent** **report** **give gold** **reply**

[–] **martinarjovsky** 6 points 1 month ago

Hi! While this is technically true, all spaces we consider are metric spaces (and not arbitrary topological spaces) which typically are very reasonable. While the work is fairly general, our paper really is focused towards $X = R^d$ or X = M a compact manifold with the standard Euclidean topology. Thus, really, non Borel sets are fairly pathological and I have yet to encounter a set in my research that's not Borel.

permalink embed save parent report give gold reply

[–] **martinarjovsky** 3 points 1 month ago

I'd like to second this thought. To this day I have to google when I'm trying to find an example of a non Borel set.

permalink embed save parent report give gold reply

[–] **serge_cell** 4 points 1 month ago

To put it simple borel set (borel measurable set) is set for which integral of index function is easily defined (index function is 1 on the set and 0 elsewhere, and it's integral is in fact the measure of the set).This is relevant because probability is a measure. In other words borel set is a set for which probability is defined (can be calculated) More general is Lebesgue measurable set.

permalink embed save parent report give gold reply

[–] **Imnimo** 5 points 1 month ago

Story of my life.

permalink embed save parent report give gold reply

[–] **feedthecreed** 1 point 1 month ago

> Let P0 be the distribution of (0, Z) ∈ R 2 (a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin.

Can anyone show me what a plot of P0 is supposed to look like in Section 2? The written description is very confusing to me.

permalink embed save report give gold reply

[–] **AlexCoventry** 2 points 1 month ago*

Probably easier to think about it generatively. You sample y~U[0,1], and then the actual sample is (0, y). So the plot is just a vertical line from (0,0) to (0,1). The entire mass is concentrated uniformly along that line.

permalink embed save parent report give gold reply

[–] **feedthecreed** 1 point 1 month ago

So is γ(x, y) just γ((0, z), (θ, z))?

If I understand this EM distance correctly, its purpose here is to give gradients to the GAN when the support of Pr doesn't match Pθ?

permalink embed save parent report give gold reply

[–] **AlexCoventry** 1 point 1 month ago

> So is γ(x, y) just γ((0, z), (θ, z))?

The mass of the optimal transport plan from $\mathbb{P}_0$ to $\mathbb{P}_\theta$ is uniformly concentrated on the diagonal {((0,z), (θ, z)): z∈[0,1]}, yes.

> its purpose here is to give gradients to the GAN when the support of Pr doesn't match Pθ?

Yes.

permalink embed save parent report give gold reply

[–] **think_yet_again** 1 point 1 month ago

Could anyone please guide me through the math in example 1?

If I understood correctly, the \gamma(x,y) is now \gamma((0, z), (\theta, z)). Then the norm under the expectation operator, is just the norm of the theta, right? But I don't understand how we go from the infimum and expectation to the final answer. When calculating KL, I am plugging

into the definition (x, y) instead of just x, and measure of x and y. Then, as the distribution P0 is only valid at points (0, z), I replace x with 0 and keep only the second integral. And then I am stuck... I know that the authors are writing 'it is easy to see', but for me it is not that easy.

permalink embed save report give gold reply

[–] **Zardinality** 1 point 1 month ago*

No you misunderstood at the very beginning, \gamma(x,y) may or may not be \gamma((0, z), (\theta, z)) (I think by \gamma((0, z), (\theta, z)) you mean a joint distribution that x and y are of the same z, that is, a uniform distribution in the joint space). But luckily it is exactly the dist that take the inf value. Think about it, whatever the outer inf or expectation are, the inner part is always greater than \theta, and there does exist a joint distribution \gamma allowing the inner part be \theta every where in the joint space, so the inf must be obtained in such dist, thus the result is \theta. As for KL, when \theta is not 0, since the inner value of expectation is \inf, so it is \inf.

permalink embed save parent report give gold reply

[–] **think_yet_again** 1 point 1 month ago

How would someone implement the last layer (loss) in TensorFlow? I tried to do it like '-tf.reduce_mean(tf.div(real_scores, tf.stop_gradient(tf.identity(real_scores))))', but it gave horrendous results.

permalink embed save report give gold reply

[–] **ajmooch** [S] 1 point 1 month ago*

So I don't know the exact tensorflow syntax for this, but my understanding is that you need to provide the output of the mean of the discriminator with 1 or -1 as the *gradient* of the loss, rather than as a normal loss function. Normally you'd have:

*output = D(G(Z)) or D(X)

*Loss = f(output)

*gradient to last layer of D = dL/d_output

*backpropagate dL/df through the net to get weights

Instead, you need to modify the second and third steps:

*gradient to last layer of D, dL/d_output is 1 if evaluating X or -1 if evaluating G(Z)

Basically, the value of the gradient at the output doesn't depend on the output value as in a normal loss function. Does that make sense?

permalink embed save parent report give gold reply

[–] **f0k-** 3 points 1 month ago

> my understanding is that you need to provide the output of the mean of the discriminator with 1 or -1 as the gradient of the loss

But that's exactly what you get when you define the mean critic output to be the (negative) loss, as done in Algorithm 1 of the paper. If L = output (or L = -output), then dL/doutput = 1 (or -1). Should be as easy in TF as in Theano.

Quickly done Lasagne WGAN example on MNIST: https://gist.github.com/f0k/f3190ebba6c53887d598d03119ca2066 Eventually generates something digit-like. IIRC the DCGAN example it's based on worked better, but it was tuned to work well, and I mostly kept the architecture/hyperparameters (feel free to tinker with it).

permalink embed save parent report give gold reply

[–] **ajmooch** [S] 1 point 1 month ago

Makes sense, my Lasagne implementation just mirrors the provided pytorch repo and uses known_grads.

permalink embed save parent report give gold reply

[–] **hma02** 1 point 20 days ago*

Thanks for the quick example. I tried removing the batchnorm from both critic and generator, and the model was still able to generate digit-like samples after 200 epochs. Here's the code: https://github.com/uoguelph-mlrg/Theano-MPI/blob/master/theanompi/models/lasagne_model_zoo/wgan.py. The changes I made include using a faster rmsprop and clipping weights instead of grad updates.

permalink embed save parent report give gold reply

[–] **think_yet_again** 1 point 1 month ago

Yes, that makes sense, thank you!

permalink embed save parent report give gold reply

[–] **omair_kg** 1 point 1 month ago

How would i implement the last layer (loss) in torch. I added nn.Sum as the last layer (since it's a mean over the batch my arguments were (1,-1,true)). However this throws an error 'attempt to index local 'gradOutput' (a number value)' during the backward pass.

I think i'm making a really basic mistake here somewhere. Any help would be much appreciated.

permalink embed save report give gold reply

[–] **ajmooch** [S] 1 point 1 month ago

Check the pytorch implementation, they just backpropagate 1/-1 as the gradient using .backward(). As Jan pointed out, you can also just set loss = -output/output since if L(x) = sign*x, dL(x)/dX = sign.

permalink embed save parent report give gold reply

[–] **ieee8023** 1 point 29 days ago

Here is the shortscience.org summary page: http://www.shortscience.org/paper?bibtexKey=journals/corr/1701.07875

permalink embed save report give gold reply

[–] **Zardinality** 1 point 27 days ago

For me the duality form of Wasserstein distance resembles MMD in a interesting way. You just need to take the \mathcal F in MMD as Lipschiz-1. So I wonder if there was any try on minimizing MMD without resort to kernel method?

permalink embed save report give gold reply

[–] **dougalsutherland** 2 points 25 days ago

It's true that MMD has the same basic kind of form, often called an integral probability metric (as discussed e.g. by Marco in this thread above). But RKHS norms are hard to compute, and the whole motivation for MMD in the first place is that its solution to this problem is easy to compute, so there's little reason to wanting to do MMD that way.

permalink embed save parent report give gold reply

[–] **Thomjazz** 1 point 26 days ago

Beautiful paper ! Thanks a lot /u/martinarjovsky (and co.) !

One small question: would it make any sense to regularize directly on the function derivative to ensure Lipschitzness, something like a regularization term max(abs(f'(x)-1)) over the x ?

We are not talking about computing the full Hessian but only a single column here so I guess we could stay O(n).

permalink embed save report give gold reply

[–] **JaejunYoo** 1 point 8 days ago

From the WGAN implementation point of view, WGAN seems to have a slight difference in objective function and weight clipping. I was surprised that these small changes could make a big difference in the results. Is there anything I missed in this implementation? I am also curious about the objective function. While talking to my colleagues, he asked me, "Then what is different from just using Euclidean distance and l_1 distance? with weight clipping?". His point was if WGAN got really good outcome due to those very small changes in the implementation of missing sqrt or absolute bars. Any comments?

permalink embed save report give gold reply

[–] **dingling00** 1 point 6 days ago

I have a question about the tensorflow realization of WGAN, I found the code from https://github.com/shekkizh/WassersteinGAN.tensorflow the question is that in the code, the critic loss = logits_real - logits_fake, through the WGAN paper, I know that we need to maximize logits_real - logits_fake when we training the critic, but in tensorflow, if we define the loss, we will minimize the loss, so I'm confused about the tensorflow realization of WGAN, maybe I have misunderstand the paper's meaning, so please help me.

permalink embed save report give gold reply

> [–] **ajmooch** [S] 1 point 6 days ago
>
> I suspect this is just a case of sign flippage--if you just switch conventions so that your goal is to make the critic output highly negative values for real samples and highly positive values for fake samples (and change the generator's objective accordingly) then it still works the same; since there's no nonlinearity we can flip those signs with impunity.
>
> **permalink embed save parent report give gold reply**
>
> > [–] **dingling00** 1 point 6 days ago
> >
> > I think about it for a while, do you mean that we can minmax W(pr,pg) or minmax W(pg, pr), it's the same thing. when we minmax W(pr,pg), the critic will try to output highly positive values for the real samples and highly negative values for the fake samples, in order to max W(pr,pg).Thank you very much!
> >
> > **permalink embed save parent report give gold reply**

[+] ***hoiafshioioh*** *comment score below threshold (9 children)*

π