EECS 182 Deep Neural Networks
Fall 2022 Anant Sahai

WGAN Homework

Deliverables: Please submit the code/notebooks/saved model as a zip file to the code Gradescope assignment. Submit your written answers in the written Gradescope assignment, and attach a pdf printout of the notebooks.

1. Vanilla GAN Intuition and Pitfalls

Although we have seen in discussions, homeworks, and lecture that Generative Adversarial Networks have great success in producing realistic images, the training process is not a simple task. In fact, there are many pitfalls in the training of GANs that make them slow and unstable. In this problem, we will investigate some of motivations of the Vanilla GAN alongside the problems that arise while using them. Namely, we will investigate problems with convergence and vanishing gradients.

(a) Let us first consider an MLE based method of calculating a generator. In order to estimate the underlying input distribution P, we try to learn a probability distribution P_θ, parameterized by parameters θ. If we attempt this using MLE, we end up having to minimize the loss function $-\frac{1}{n}\sum_{i=1}^{n}\log P_\theta(x^{(i)})$ over θ, where $x^{(i)}$ represents the ith training example. Show that as the number of points in the input data tends to infinity, minimizing the loss function is the same as minimizing $D_{\text{KL}}(P||P_\theta)$, which is given by $D_{\text{KL}}(P||Q) = \mathbb{E}_{x\sim P}[\log \frac{P(x)}{Q(x)}]$. Conceptually, why might this pose a problem? Try considering points where $P(x)$ is nonzero but $P_\theta(x)$ is zero.
(Hint: Consider that when the number of input data points goes to infinity, the inputs are distributed according to the true probability distribution.)

Solution: Applying the hint, we have that

$$\min_\theta \lim_{n\to\infty} -\frac{1}{n}\sum_{i=1}^{n}\log P_\theta(x^{(i)}) = \min_\theta -\int P(x)\log P_\theta(x)dx$$

Now, we note that this is the same as

$$\min_\theta \int P(x)\log P(x)dx - \int P(x)\log P_\theta(x)dx$$

because $\int P(x)\log P(x)dx$ is a constant with respect to θ. Finally, we have that

$$\min_\theta \int P(x)\log P(x)dx - \int P(x)\log P_\theta(x)dx = \min_\theta \mathbb{E}_{x\sim p}[\log \frac{P(x)}{P_\theta(x)}] = D_{\text{KL}}(P||P_\theta)$$

Consider if we have a point where $P(x) > 0$ but $P_\theta(x) = 0$. Then, the KL-divergence is infinite, which is undesirable. This issue becomes a problem when the real distribution is on located on some small, low-dimensional space.

(b) The GAN approach, on the other hand, involves training a generator G_θ and discriminator D_ϕ. The generator takes in some random noise, $\epsilon \sim \mathcal{N}(0,1)$, and outputs a sample input. The discriminator takes in an input and assigns it a probability that it came from the data distribution versus the generator. We have seen in lecture that the GAN training procedure is analogous to a min-max two player noncooperative game. That is, the generator and the discriminator are two different models that are both trained simultaneously with gradient descent with the objective of finding the Nash Equilibrium (i.e. when the generator/discriminator does not change its action regardless of its opponent's action). More formally, it is that the pair (G^*, D^*) such that

$$\min_G \max_D \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1 - D(G(x)))] \tag{1}$$

is achieved where p_r is the distribution of the real data and p_g is the distribution of the generated data. Suppose one player takes control of x to minimize $f(x, y) = xy$ while at the same time the other player constantly updates y to minimize $g(x, y) = -xy$. Show that it is impossible for gradient descent to converge assuming simultaneous updates.

Solution: Since $\frac{\partial f}{\partial x} = y$ and $\frac{\partial g}{\partial y} = -x$, we have the following gradient descent updates for x and y:

$$x_t = x_{t-1} - \eta y_{t-1} \tag{2}$$

$$y_t = y_{t-1} + \eta x_{t-1} \tag{3}$$

If for some T, x_T and y_T have different signs, every following gradient descent update causes oscillation and unstable behavior. Thus, we will not have gradient descent converge. We can also see this illustrated in the figure below.

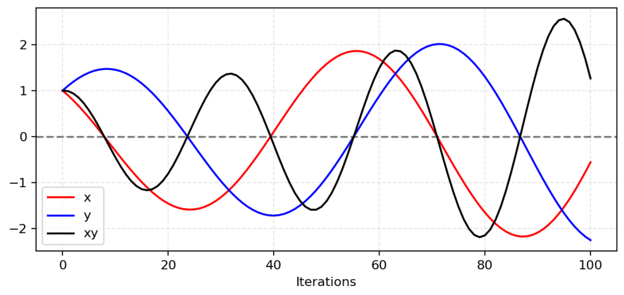

Figure 1: Simulated Example of Updates

This is an example where we cannot achieve a Nash Equilibrium between two different players, and this is why sometimes GANs cannot achieve an optimal pair (G^*, D^*).

(c) Vanilla GANs also face the problem of vanishing gradients. Let us assume the case where in the training of our GAN, we happen to have the discriminator converge to being perfect. That is, $\forall x \in p_r, D(x) = 1$ and $\forall x \in p_g, D(x) = 0$. Furthermore, if we optimize the minimax equation seen above for the discriminator by holding the generator fixed, we see that $D^* = \frac{p_r}{p_g + p_r}$. Show that this causes the loss function defined in the minimax equation to go to zero.

Solution: If we have a perfect discriminator, then there are two cases. Either $p_g = 0, p_r \neq 0$ or $p_g \neq 0, p_r = 0$. In either case, notice that either one of the terms will become zero since the discriminator is perfect. The other term will cancel because either p_g is zero or p_r is zero. This means we have vanishing gradients in the case of the discriminator being perfect. Many would think that one of the reasons that GAN training is very hard is due to the idea that these probability distributions lie in very high dimensional manifolds. However, practically speaking, it has been shown that these distributions seem to concentrate in lower-dimensional manifolds. Since both P_r and P_g rest in low dimensional manifolds, the probability that these are disjoint is almost surely 1. When they have disjoint supports, we are always capable of finding a perfect discriminator that separates real and fake samples. Thus, we can see why this is a common issue with Vanilla GAN training.

2. Wasserstein Distance

As we have seen, there are tons of problems encountered with Vanilla GAN training. As seen in lecture and discussion, Vanilla GAN training is based on a probability distribution measure of distance called Jenson-Shannon Divergence. In fact, this measure of distance is the root of all the problems we have seen above. In this problem, we will look at another measure of distance between probability distributions called Wasserstein Distance, also known as Earth-Mover (EM) Distance. This metric is the basis for why WGAN has better performance compared to Vanilla GAN.

(a) The Wasserstein Distance is sometimes called the Earth-Mover Distance because informally, it can be thought as the minimum energy cost of moving and transforming a pile of dirt in the shape of one probability distribution to the shape of another probability distribution. Let us consider a simple case where the probability distributions are discrete. Suppose P and Q are two probability distributions, each having four piles of dirt and both having ten shovelfuls of dirt in total. The number of shovelfuls of dirt in each dirt pile are as follows:

$$P_1 = 3, P_2 = 2, P_3 = 1, P_4 = 4Q_1 = 1, Q_2 = 2, Q_3 = 4, Q_4 = 3 \tag{4}$$

Suppose we label the cost of moving P_i to Q_i as δ_i. Then, we have the following relation: $\delta_{i+1} = \delta_i + P_i - Q_i$. Then, calculate the Earth-Mover Distance $W = \sum_i |\delta_i|$. Assume $\delta_0 = 0$

Solution: If we graph out the probability distributions and try to count how much dirt we need to move over and cut to transform P to Q, we have effectively calculated the EM Distance.

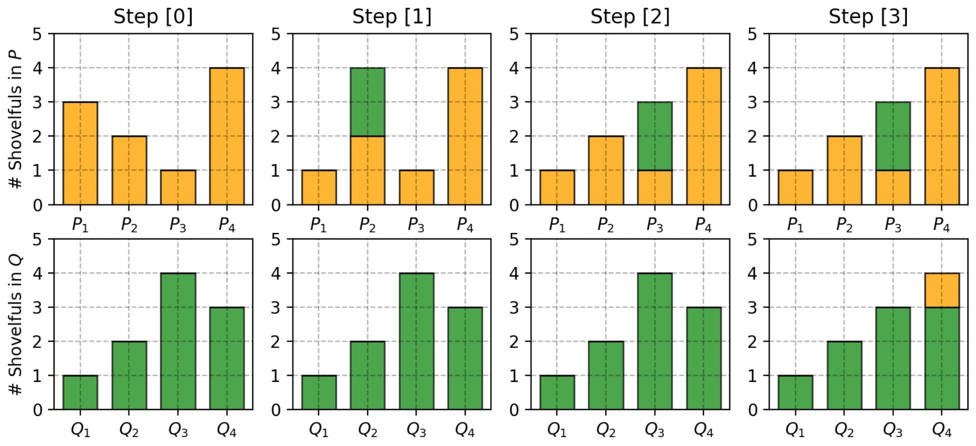

Figure 2: Step-By-Step Illustration of EM Distance Between P and Q

In order to change P into Q, we must do the following:

i. Move 2 shovefuls of dirt from P_1 to P_2
ii. Move 2 shovefuls of dirt from P_2 to P_3
iii. Move 1 shovefuls of dirt from Q_3 to Q_4

Notice that this corresponds with the result we obtain using the relation given:

i. $\delta_0 = 0$
ii. $\delta_1 = \delta_0 + P_1 - Q_1 = 0 + 3 - 1 = 2$
iii. $\delta_2 = \delta_1 + P_2 - Q_2 = 2 + 2 - 2 = 2$
iv. $\delta_3 = \delta_2 + P_3 - Q_3 = 2 + 1 - 4 = -1$

v. $\delta_4 = \delta_3 + P_4 - Q_4 = -1 + 4 - 3 = 0$

Therefore, $W = \sum_i |\delta_i| = 2 + 2 + 1 = 5$.

(b) One of the main reasons that the Wasserstein Distance metric is utilized in the WGAN is because it provides a smoother measure of distance between probability distributions compared to KL and JS Divergence. As we saw in Question 1, when there is no overlap between probability distributions P_r and P_g, we are always capable of finding a perfect discriminator that separates real and fake samples if we are using JS Divergence as our distance metric. This results in vanishing gradient problems. However, we will see in this example that this is not the case if we are using Wasserstein distance. Suppose we have two probability distributions P and Q such that the following hold:

$$\forall (x, y) \in P, x = 0, y \sim U(0, 1) \tag{5}$$
$$\forall (x, y) \in Q, x = \theta, 0 \le \theta \le 1, y \sim U(0, 1) \tag{6}$$

Calculate the following measures of distance for both $\theta \neq 0$ and $\theta = 0$

i. KL Divergence

ii. JS Divergence

iii. Wasserstein Distance

Note: A divergence measures the distance between two distributions p and q. In particular, for distributions p and q with common support $\mathcal{X}$, typically used divergence metrics include

$$D_{KL}(P||Q) = \mathbb{E}_{\mathbf{x}\sim P}[\log \frac{P(x)}{Q(x)}] \qquad \textit{(Kullback-Leibler Divergence)}$$
$$D_{JS}(P, Q) = \frac{1}{2} D_{KL}(P||\frac{P+Q}{2}) + \frac{1}{2} D_{KL}(Q||\frac{P+Q}{2}) \qquad \textit{(Jenson-Shannon Divergence)}$$

Solution: When $\theta \neq 0$:

$$D_{KL}(P||Q) = \sum_{x=0, y\sim U(0,1)} P(x) \log\left(\frac{P(x)}{Q(x)}\right) = \sum_{x=0, y\sim U(0,1)} 1 \log\left(\frac{1}{0}\right) = +\infty \tag{7}$$
$$D_{KL}(Q||P) = \sum_{x=\theta, y\sim U(0,1)} Q(x) \log\left(\frac{Q(x)}{P(x)}\right) = \sum_{x=\theta, y\sim U(0,1)} 1 \log\left(\frac{1}{0}\right) = +\infty \tag{8}$$
$$D_{JS}(P, Q) = \frac{1}{2} D_{KL}(P||\frac{P+Q}{2}) + \frac{1}{2} D_{KL}(Q||\frac{P+Q}{2}) = \sum_{x=0, y\sim U(0,1)} 1 \log\left(\frac{1}{1/2}\right) = \log 2 \tag{9}$$
$$W(P, Q) = |\theta| \tag{10}$$

When $\theta = 0$:

$$D_{KL}(P||Q) = D_{KL}(Q||P) = D_{JS}(P, Q) = 0 \tag{11}$$
$$W(P, Q) = |\theta| \tag{12}$$

(c) Using your calculations above, explain why Wasserstein Distance is a better measure of distance? Why does it provide a smoother measure of distance, and how can this help improve the learning process using gradient descent?

Solution: D_{KL} gives us an infinite measure of distance when the distributions are disjoint. The value of D_{JS} jumps very suddenly when the two distributions are fully overlapped, but this does not occur with Wasserstein distance. This example shows that there exist sequences of distributions that don't converge under the JS or KL, but which do converge under the Wasserstein distance. Furthermore, notice that when the two distributions are not overlapping, we can converge until we find the perfect discriminator. This results in $D_{JS} = \log 2$. When looking at the loss incurred by the generator, $L_G = 2D_{JS}(P, Q) - \log 4 = 0$. Thus, we encounter the vanishing gradient problem. This is something we do not have to worry about with Wasserstein Distance.

(d) When we extend the Wasserstein Distance to continuous probability distributions, we have the following: $W(p_r, p_g) = \inf_{\gamma \sim \prod(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma} \|x - y\|$ where $\prod(p_r, p_g)$ is the set of all joint probability distributions over p_r and p_g and $\gamma(x, y)$ is the percentage of dirt one should transport from x to y to make x the same probability distribution as y. Conceptually explain why it is computationally intractable to solve this optimization problem.

Solution: The reason that we cannot simply solve this optimization problem is because it is intractable to exhaust all the possible $\prod(p_r, p_g)$ in order to calculate $\inf_{\gamma \sim \prod(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma} \|x - y\|$. Luckily, we can take advantage of Kantorovich-Rubenstein Duality by taking the sup over 1-Lipschitz continuous functions.

Instead of optimizing the Wasserstein Distance directly, we make use of simple trick that you may have encountered in EECS 127/227 called duality. In particular, we make use of Kantorovich-Rubenstein Duality. This allows us to transform the Wasserstein Distance optimization problem into $\sup_{\|f\|_L \le 1} \mathbb{E}_{x\sim p_r} D(x) - \mathbb{E}_{x\sim p_g} D(x)$[1] where $\|f\|_L \le 1$ is notation to indicate f must be a 1-Lipschitz continuous function (i.e. $\forall x_1, x_2 \in \mathbb{R}, \big|f(x_1) - f(x_2)\big| \le |x_1 - x_2|$). Now, our optimization problem is computationally tractable and can be easily trained via gradient descent. Note that with this transformation, the discriminator is not a direct critic of telling the fake samples apart from the real ones anymore. Instead, it is trained to learn a 1-Lipschitz continuous function to help compute Wasserstein distance. As the loss function decreases in the training, the Wasserstein distance gets smaller and the generator model's output grows closer to the real data distribution. One problem with this formulation is that it is difficult to maintain 1-Lipschitz continuity. However, this problem can be resolved using weight clipping. After every gradient update, clamp the weights w to a small window, such as $[-0.01, 0.01]$. This results in a compact parameter space and allows us to preserve the 1-Lipschitz continuity.

[1]For those interested in the proof: Kantorovich-Rubenstein Duality Blog Post and Proof

3. Improving The WGAN Via The Gradient Penalty

Now that we have built some more intuition on why the WGAN is an improvement on the Vanilla GAN, we will lastly introduce a method to improve WGAN. In Question 2, we saw how we could use duality and Lipschitz continuity to create an easier optimization problem for gradient descent to solve. However, we needed to use a technique called weight clamping to ensure that we preserve Lipschitz continuity after each iteration of gradient descent. It turns out that weight clipping is a terrible way to enforce a Lipschitz constraint because it can cause WGAN to suffer slow convergence after weight clipping (when clipping window is too large) and vanishing gradients (when clipping window is too small). Luckily, a paper published in 2017 called *Improving Training of Wasserstein GANs*[2] proposed an alternative to weight clipping by using the gradient penalty. We will investigate the motivation for this penalty.

(a) A real-valued function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is called a K-Lipschitz continuous function if $\forall x_1, x_2 \in \mathbb{R}^n, \|f(x_1) - f(x_2)\| \leq K\|x_1 - x_2\|$. You may assume that f is differentiable on $\mathbb{R}^n$.

i. Let f is K-Lipschitz, its gradient norm must be atmost K.
(Hint: Consider $x_1 = x + h$ and $x_2 = x$ for $x, h \in \mathbb{R}^n$)

ii. Proof if f has a gradient norm at most K, then f is K-Lipschitz.
(Hint: Use Mean Value Theorem and consider the line $(1 - x)t + ty$)

Note: The Mean Value Theorem says that if f is a continuous function on the closed interval $[a, b]$ and differentiable on the interval (a, b), then $\exists c \in (a, b)$ such that $f(b) - f(a) = f'(c)(b - a)$

Solution:

i. We will first prove the forward direction. If f is K-Lipschitz, then $\forall x_1, x_2 \in \mathbb{R}^n, \|f(x_1) - f(x_2)\| \leq K\|x_1 - x_2\|$. Using $x_1 = x + h$ and $x_2 = x$ for $x, h \in \mathbb{R}^n$, we get $\|f(x+h) - f(x)\| \leq K\|x + h - x\|$. Thus, $\|\frac{f(x+h)-f(x)}{h}\| \leq K$. Taking the limit results in $\lim_{h \longrightarrow \infty} \|\frac{f(x+h)-f(x)}{h}\| = \|\nabla f\| \leq K$.

ii. We will prove the reverse direction. Let $\gamma(t) = (1 - x)t + ty$. Then, by the Mean Value Theorem, $f(y) - f(x) = f(\gamma(1)) - f(\gamma(0)) = (f(\gamma(r)))'$ where $r \in [x, y]$. Then, $(f(\gamma(r)))' = \nabla f(\gamma(r)) \cdot \gamma'(r) = \nabla f(\gamma(r))(y - x)$. Taking the norm on both sides results in $|f(x) - f(y)\| = \|\nabla f(\gamma(r))(x-y)\| \leq \|\nabla f(\gamma(r))\|\|x-y\|$. Since we know that the gradient norm of f is bounded by K, we get $|f(x) - f(y)\| \leq K\|x - y\|$. Thus, f is K-Lipschitz.

(b) The authors of the paper proposed an alternative objective for WGAN: $\mathbb{E}_{x\sim p_r} D(x) - \mathbb{E}_{x\sim p_g} D(x) + \lambda \mathbb{E}_{x\sim p_{\hat{x}}}[\|\nabla D(\hat{x})\| - 1)^2]$ where $\hat{x}$ is a randomly weighted average between a real and generated sample such that $\hat{x} = \epsilon x + (1 - \epsilon)x$ where $\epsilon \sim Unif([0, 1])$. Using what you learned from part a, explain why this alternative objective makes sense mathematically. How does the gradient penalty work? *(Hint: Recall that using Kantorovich-Rubenstein Duality, we required f to be a 1-Lipschitz continuous function)*

Solution: Using part a, we know that f is a 1-Lipschitz continuous function iff its gradient norm will be atmost 1. Therefore, by having this penalty of the form $\lambda \mathbb{E}_{x\sim p_{\hat{x}}}[(\|\nabla D(\hat{x})\| - 1)^2]$, we can enforce D be 1-Lipschitz continuous. This allows us to make full use of Kantorovich-Rubenstein Duality while avoiding a lot of the problems that came with weight clipping. This penalty term is intuitive. The discriminator is trained to output values as small as possible for real samples, and as large as possible for fake samples, while the gradient penalty term preserves D being 1-Lipschitz continuous. A loss that has no strict lower bound might seem strange, but in practice the competition between the generator and the discriminator keeps the terms roughly equal.

[2]For those interested in the paper: Improving Training of Wasserstein GANs

4. GAN and WGAN Compare and Contrast

In this question, you'll implement the architecture for a Vanilla GAN and WGAN with gradient penalty. Please click the link to access the question: Code For Question 4. **We highly encourage students to use Google Colab for this project as we have built in all the functionality to work smoothly on Google Colab. We also encourage students to start this question early since running implementations for GAN and WGAN can take a long time!** For this question, please submit a .zip file your completed work to the Gradescope assignment titled "WGAN HW (Code)".
Note: Due to the convergence issues we discussed in Question 1, you may run into problems when training your GAN. Around Epoch 30 to 40, if you start noticing your synthetic faces start looking worse / less human-like, you may have run into a convergence issue so please rerun.

(a) Implement the architecture for Vanilla GAN

Solution: Look at 1_GAN_ANSWERS.ipynb for solutions

(b) Implement the architecture for WGAN with Gradient Penalty

Solution: Look at 2_WGAN_ANSWERS.ipynb for solutions

(c) You may have had to rerun the GAN implementation a few times since there might be errors with convergence due to the issues we discussed in Question 1. Did you ever need to rerun the WGAN with Gradient Penalty? How does this reaffirm the benefits of WGAN over GAN? Which model generated the best synthetic images?

Solution: While there is a chance that students may have run into convergence issues with WGAN, this happens very rarely. A correct answer includes key ideas discussed in Question 1 and Question 2. Furthermore, most students should have WGAN performing better in generating synthetic images.

5. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework? Write it down here where you'll need to remember it for the self-grade form.**

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672–2680).

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17). JMLR.org, 214–223.

[3] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of Wasserstein GANs. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 5769–5779.

[4] Questions were adapted from the following blog: From GAN to WGAN

Contributors:

- Darsh Balani
- Sharan Sahu
- Raghav Ramanujam