Final Project: Extended Abstract

The Mathematical Engineering of Deep Learning[1]

Rohan Hitchcock, The University of Melbourne

In this project we review the paper "Wasserstein GAN" by M. Arjovsky, S. Chintala, and L. Bottou [1] ('the paper'). This paper outlines an approach to training generative adversarial networks (GANs) which uses the Wasserstein distance between probability distributions as a loss function. The paper begins by providing a theoretical analysis of the Wasserstein distance, and explains why one might expect this would lead to a better GAN training algorithm. These ideas are then implemented in the WGAN algorithm, which is compared empirically with other approaches to GAN training.

In GAN training, the problem of training the generator can be understood as approximating a random variable $X \sim \mathbf{P}_r$, $X \in \mathcal{X}$, where $\mathbf{P}_r$ is the distribution of real data. Typically this is done as follows. We fix some noise distribution $Z \sim \mathbf{P}_{\text{noise}}$, $Z \in \mathcal{Z}$ and consider some family of functions $\{g_\theta : \mathcal{Z} \to \mathcal{X}\}_{\theta \in \Theta}$ (for example, each g_θ is a neural network with weights $\theta \in \Theta$). We then attempt to find a $\theta^* \in \Theta$ such that $g_{\theta *}(Z) \sim X$. Let $\mathbf{P}_\theta$ denote the distribution of $g_\theta(Z)$. Given some metric D on probability distributions on $\mathcal{X}$ we can formulate this as finding $\theta^* \in \Theta$ which minimises $D(\mathbf{P}_r, \mathbf{P}_\theta)$. Note that the metric D can be implicit in the learning algorithm. For example, one can show than minimising the loss function of the standard GAN approach[2] is equivalent to minimising the Jensen-Shannon distance (Section 7.5.1 in [2]).

It is desirable that $D(\mathbf{P}_r, \mathbf{P}_\theta)$ be a continuous function of θ, so that if we have a sequence $\theta_n \to \theta^*$ (for example, produced by an optimisation algorithm) then this implies $D(\mathbf{P}_r, \mathbf{P}_\theta) \to \min D(\mathbf{P}_r, \mathbf{P}_\theta)$. Furthermore, if $D(\mathbf{P}_r, \mathbf{P}_\theta)$ is a differentiable function of θ then this distance may be used directly as the loss function in training since we can evaluate $\nabla_\theta D(\mathbf{P}_r, \mathbf{P}_\theta)$. Neither of these properties necessarily hold of Jensen-Shannon distance (see Example 1 of the paper), so nice behaviour as $\theta_n \to \theta^*$ is not guaranteed in the standard GAN training approach.

The Wasserstein distance between distributions $\mathbf{P}_a$ and $\mathbf{P}_b$ is defined as

$$W(\mathbf{P}_a, \mathbf{P}_b) = \inf_{\gamma \in \Pi(\mathbf{P}_a, \mathbf{P}_b)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\|\right]$$

where $\Pi(\mathbf{P}_a, \mathbf{P}_b)$ denotes the set of joint probability distributions with marginals $\mathbf{P}_a$ and $\mathbf{P}_b$. The key theoretical results of the paper are summarised below. As in the situation above, let $X \sim \mathbf{P}_r$ be the real data distribution on space $\mathcal{X}$, $Z \in \mathcal{Z}$ be a fixed random variable, and $\{g_\theta : \mathcal{Z} \to \mathcal{X}\}_{\theta \in \Theta}$ a family of functions with $g_\theta(Z) \sim \mathbf{P}_\theta$. Then:

(1) If g_θ is continuous in θ then so is $W(\mathbf{P}_r, \mathbf{P}_\theta)$ (Theorem 1 of the paper).

(2) In addition to (1), if the g_θ satisfies some further mild properties[3], then $W(\mathbf{P}_r, \mathbf{P}_\theta)$ is differentiable almost everywhere as a function of θ (Theorem 1 of the paper).

(3) If g_θ is a neural network[4] with weights θ then the assumptions of (1) and (2) hold. Therefore in this case $W(\mathbf{P}_r, \mathbf{P}_\theta)$ is a continuous, differentiable a.e. function of θ (Corollary 1 of the paper).

(4) Let $\mathbf{P}$ be a distribution and $(\mathbf{P}_n)_{n\in\mathbb{N}}$ a sequence of distributions, all on the same space. Then $\mathbf{P}_n$ converges in distribution to $\mathbf{P}$ if and only if $W(\mathbf{P}_n, \mathbf{P}) \to 0$ (Theorem 2 of the paper).

Together, (1), (2) and (3) tell us that the Wasserstein distance $W(\mathbf{P}_r, \mathbf{P}_\theta)$ is a suitable loss function for training a neural network generator g_θ since it can provide gradients to use in an optimisation algorithm. We also have an indication from (4) that the Wasserstein distance is the 'right' loss function for our purpose: if we are successful in minimising $W(\mathbf{P}_r, \mathbf{P}_\theta)$ then $g_\theta(Z)$ will converge in distribution to X.

The problem of how to actually compute the Wasserstein distance remains. The Kantorovich-Rubinstein duality (see Theorem 5.10 of [3]) says

$$\kappa \cdot W(\mathbf{P}_a, \mathbf{P}_b) = \sup_{\|f\|_L \leq \kappa} \left(\mathbb{E}_{x\sim\mathbf{P}_a}\left[f(x)\right] - \mathbb{E}_{x\sim\mathbf{P}_b}\left[f(x)\right]\right)$$

where the supremum is taken over all κ-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$. Consider a family of neural networks $\{f_w\}_{w\in\mathcal{W}}$ parameterised by their weights. If $\mathcal{W}$ is compact then all the f_w will be κ-Lipschitz for some $\kappa > 0$. We can then estimate $\kappa \cdot W(\mathbf{P}_a, \mathbf{P}_b)$ by finding $\max_{w\in\mathcal{W}} \left(\mathbb{E}_{x\sim\mathbf{P}_a}\left[f_w(x)\right] - \mathbb{E}_{x\sim\mathbf{P}_b}\left[f_w(x)\right]\right)$ via a gradient ascent based optimisation procedure. To enforce the condition that $\mathcal{W}$ be compact, we can set $\mathcal{W} = [-c, c]^p$ and after

[1] Course website: deeplearningmath.org

[2] For a discriminator d and generator g, the loss is $\mathbb{E}_{x\sim\mathbf{P}_r}\left[\log d(x)\right] - \mathbb{E}_{x\sim\mathbf{P}_{\text{noise}}}\left[\log(1 - d(g(x)))\right]$

[3] g_θ is locally Lipschitz, and Assumption 1 of the paper holds of g_θ.

[4] In the paper this is proved in the case that the activation functions of g_θ are smooth and Lipschitz. The authors also claim this holds when the activation functions of g_θ are ReLU.

each iteration adjust any weights which lie outside $\mathcal{W}$ to the closest value in $[-c, c]$. We call c the clipping parameter.

Using this idea, the paper proposes the WGAN algorithm for training GANs. Consider two families of neural networks $\{f_w\}_{\mathcal{W}}$ (the discriminator) and $\{g_\theta\}_{\theta\in\Theta}$ (the generator) parameterised by their weights, where $\mathcal{W} = [-c, c]^p$ for some $p \in \mathbb{N}$. Let $\mathbf{P}_r$ denote the distribution of the real data and $\mathbf{P}_\theta$ the distribution of $g_\theta(Z)$ for some fixed random variable $Z \sim \mathbf{P}_{\text{noise}}$. We first train f_w to estimate $\kappa \cdot W(\mathbf{P}_\theta, \mathbf{P}_r)$ by maximising $\mathbb{E}_{x\sim\mathbf{P}_r}[f_w(x)] - \mathbb{E}_{z\sim\mathbf{P}_{\text{noise}}}[f_w(g_\theta(z))]$ with respect to $w \in \mathcal{W}$ in the manner previously described. In each iteration we take samples $\{x_i\}_{i=1}^m \sim \mathbf{P}_r$ and $\{z_i\}_{i=1}^m \sim \mathbf{P}_{\text{noise}}$ and compute $\frac{1}{m}\sum^m f_1(x_i) \approx \mathbb{E}_{x\sim\mathbf{P}_r}[f_w(x)]$ and $\frac{1}{m}\sum^m f_w(g_\theta(z_i)) \approx \mathbb{E}_{z\sim\mathbf{P}_{\text{noise}}}[f_w(g_\theta(x))]$ in order to estimate the expectations. Once a sufficient estimate for $\kappa \cdot W(\mathbf{P}_\theta, \mathbf{P}_r)$ has been obtained we update the weights of g_θ once by computing $\kappa \cdot \nabla_\theta W(\mathbf{P}_\theta, \mathbf{P}_r)$, aiming to minimise $\kappa \cdot W(\mathbf{P}_\theta, \mathbf{P}_r)$. This entire process is repeated until g_θ has sufficiently converged. It is interesting to note that a discriminator network arises naturally as a way to estimate the Wasserstein distance, even though the only goal was to produce a generator.

The authors found that the WGAN algorithm was more robust to variations in network architecture than the standard approach, did not require batch normalisation to produce acceptable samples, and did not exhibit mode collapse. This is consistent with the the results stated in (1)–(4), noting that none of those statements hold of the Jensen-Shannon distance. The authors also note that $\kappa \cdot W(\mathbf{P}_r, \mathbf{P}_\theta)$ has the advantage of being a meaningful loss metric (see results (1) and (3)), and so can be used to quantitatively judge convergence in GAN training. As the authors state, however, the fact the constant κ is unknown and depends on the discriminator's architecture makes it unsuitable for direct comparison of different GAN architectures.

Our computational demonstration consists of two parts. In the first part we adapt the central idea of the WGAN algorithm into an algorithm which estimates the Wasserstein distance between two probability distributions. While other methods for this exist, they tend to impose restrictions on the distributions (e.g. non-singular, have the same support, or one-dimensional). While this approach only estimates the Wasserstein up to a positive scalar constant, it has the advantage of not making any assumptions about the two distributions. When applied to Example 1 of the paper, in which the Wasserstein distance $W(\mathbf{P}_{\theta^*}, \mathbf{P}_\theta)$ is computed analytically as a function of θ (for some specified distribution $\mathbf{P}_\theta$)), we found this method produced a good approximation of $\kappa \cdot W(\mathbf{P}_{\theta^*}, \mathbf{P}_\theta)$ for some $\kappa > 0$. It is worth noting that since we know $W(\mathbf{P}_{\theta^*}, \mathbf{P}_\theta)$ in this case, this gives us a method to compute κ for this architecture and clipping parameter. Since κ depends only on the space $\mathcal{W} = [-c, c]^p$, κ should be fixed for a given discriminator architecture and clipping parameter. It is, however, unclear how well this procedure estimates κ, or how this estimate would be verified. Therefore it is unclear whether this would be useful in using the Wasserstein distance to directly compare different GAN architectures.

In the second part of our computational demonstration we implement the WGAN algorithm in full and compare it to the standard GAN formulation. This is done with very simple network architectures. In both cases the generator and discriminator are networks with two fully connected layers with 100 neurons each and ReLU activation functions. The output neurons of the generator are linear combinations of neurons in the previous layer. In the WGAN algorithm the output neuron of the discriminator is a linear combination of neurons in the previous layer, while in the standard GAN formulation a sigmoid activation function is also applied. This is because the discriminator in the standard GAN formulation is assumed to be estimating the probability that a particular input was drawn from the real distribution, while in the WGAN case we are approximating a supremum taken all κ-Lipschitz functions, not all of which will be bounded by $[0, 1]$.

When trained on subsets of the MNIST dataset (only using instances belonging to a subset of the classes) we found the WGAN algorithm was able to train a generator to generate good samples when trained on a single MNIST class, and acceptable samples when trained on two MNIST classes. The standard GAN algorithm was not able to generate samples, even when trained on just one MNIST class.

The goal of this demonstration is to demonstrate the robustness of the WGAN algorithm compared to the standard approach. Indeed, there is no reason to expect that these network architectures are well-suited to this task of working with images. Despite this, the WGAN approach is able to produce decent results on a simple generation problem with a limited computational budget, while the standard training GAN approach is not. This is not to say the standard approach cannot produce good generative models, however this generally requires significant tuning of the networks' architecture (see [4]). In contrast the WGAN approach is able to train networks even without any adjustment of network architecture to suit the task at hand.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017. arXiv: 1701.07875 [stat.ML].

[2] B. Liquet, S. Moka, and Y. Nazarathy. (Feb. 2021). Generative Adversarial Networks, The Mathematical Engineering of Deep Learning, [Online]. Available: https://deeplearningmath.org/generative-adversarial-networks.html (visited on 02/11/2021).

[3] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.

[4] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016. arXiv: 1511.06434 [cs.LG].