Dynamic Heuristic Training of Generative Adversarial Networks

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of
Bachelor of Arts
in Computer Science
by
Kevin S. Kenneally

DARTMOUTH COLLEGE
Hanover, NH
May 31, 2018

Dartmouth Computer Science
Technical Report TR2018-XXX

Examining Committee:

_____

Richard H. Granger

Abstract

Generative adversarial networks (GANs) were proposed by Ian Goodfellow in 2014, and have quickly risen to the top of the field of machine learning for producing incredibly realistic and representative images, speech, and sentence generation samples. However, GANs have run into numerous issues with convergence including tail vibrations, gradient vaporization, and mode collapse. A number of alternative methods have been proposed to fix some of these issues with the most common proposals including updates to the distance metric and policy gradients used to update the parameters of the discriminator D and generator G models. While several of these proposed gradients have achieved greater success than Jensen-Shannon Divergence, including the Wasserstein-1 metric and gradient penalties, in preventing gradient vaporization and mode collapse issues with convergence remain an open topic. In this thesis, we propose a novel method for GAN training that takes advantage of both the advantages of the Wasserstein-1 metric in early training and the JS-Divergence metric in later convergence by changing the policy gradients when there is enough significant overlap between the model distribution P_g and the data distribution P_r.

Table of Contents

1 Introduction

In 1950, mathematician and computer scientist Alan M. Turing posed a question, "Can machines think?" While the question was highly speculative at the time, with the first electronic programmable general-purpose computer spinning to life only five years earlier at the United States Army's Ballistic Research Laboratory at the University of Pennsylvania, Turing believed this question would inevitably be necessary to ask as computing power grew. In order to provide a substantive basis for such a question, Turing proposed a test: a human interrogator would interact with both a human and a computer through a teleprinter. After asking a series of questions to discern the identities of "person A" and "person B," the interrogator would then be asked to identify which was the human and which the computer. A reliably deceptive computer program would suggest that it appears to think, and so we would answer the question "can machines think?" with the affirmative (Turing, 1950). While Turing had originally posed this question to further explore the mathematical properties that constituted information processing and computing, it inevitably inspired the field of "artificial intelligence," a term coined by computer scientist John McCarthy at Dartmouth College in 1955.

…

What does it mean to learn a probability distribution? The classical answer is to learn the probability density function. This can be done by defining a parametric set of probability densities $(P_\theta)_{\theta \in \mathbb{R}^d}$ and calculating the one that maximizes the likelihood on the data: given a real data set $\{x^{(i)}\}_{i=1}^m$, we would determine the maximum

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\Theta(x^{(i)})$$

Defining the real data distribution P_r and the parameterized density P_θ, this problem asymptotically amounts to minimizing the Kullback-Leibler (KL) divergence $KL(P_r||P_\theta)$. In order for this to be valid, the probability density P_θ must exist. However, this is not always the case. If the distributions are supported by low-dimensional manifolds in a high-dimensional representation space, it is unlikely that the distributions have a non-negligible intersection, which means the Kullback-Leibler divergence is not defined (or infinite) (Arjovsky et al., 2017). With an undefined KL divergence, we can not use it as a gradient to minimize the distance between P_r and P_θ, limiting its use as a metric.

There have been a number of remedies proposed in the machine learning literature to fix this problem. The typical solution is to add a noise term to the model distribution. For example, in the simplest case a Gaussian noise with a large variance is added to the model's output in order to encompass a wider range of the representation space. This reduces the possibility that the model distribution and the data distribution have only a negligible overlap; however, the noise degrades the quality of the model samples limiting the accuracy of our model's approximation to the real data (Goodfellow et al., 2014).

Therefore, rather than attempting to estimate the probability density of the model distribution P_θ, which may not exist, we can define a model that generates samples following a model distribution P_θ. Defining a random variable Z with a uniform distribution $p(z)$, and a parametric function $g_\theta : Z \rightarrow \mathcal{X}$ that maps inputs from the latent space to outputs in the representation space, we can use the parametric function to generate a given model distribution P_θ. By varying the parameter θ, we can change the parametric function's output, and change the model distribution P_θ to be as close as possible to the real data distribution P_r (Arjovsky et al., 2017). This is a powerful method as it allows us to represent distributions supported by a low-dimensional manifold in a high-dimensional representation space.

Some models that follow this approach are Variational Auto-Encoders (VAEs), Restricted Boltzmann machines (RMBs), and Markov Chain Monte Carlo (MCMC) methods.

However, the former two rely on maximum likelihood estimations and so suffer from the same issues of restricted overlap that the previous models did. Moreover, MCMC methods have difficulty with variation in high-dimensional representation spaces (Bengio et al., 2013). Generative Adversarial Networks (GANs) are well known to follow the model generation outlined above, so they are a strong candidate to fix the problem of low-dimensional manifolds.

GANs, which will be outlined in Section 2.1, provide a solution for creating robust generators that can effectively learn a data distribution and produce samples that appear to be from such a distribution. Numerous studies in the literature since Ian Goodfellow created the model in 2014 have shown the power of these models in producing images, speech, and image captions that appear realistic even to human's senses (Dziugate, 2015). Furthermore, GANs offer much more flexibility than the former models in the definition of the objective function, including KL divergence, Jensen-Shannon (JS) divergence, and all other *f*-divergences (Nowozin et al., 2016). These metrics and their rationales will be further explored in Section 2 and Section 3.

While the original GAN architecture Kullback-Leibler or Jensen-Shannon divergence, more recent research has demonstrated the promise of the Earth-Mover (EM) distance or Wasserstein-1 metric as a gradient for GAN convergence. In particular, the issues of a vanishing gradient, convergence fluctuations, and mode collapse have shown that the Wasserstein-1 metric provides a better policy gradient to update the loss function of the GAN. The Wasserstein Generative Adversarial Network (WGAN) architecture will be outlined in Section 4.

However, the WGAN still suffers from issues of convergence. As the authors outlined in their own paper absolute "[w]eight clipping is a clearly terrible way to enforce a Lipschitz constraint" on the weight values for the discriminator, and should be replaced by a more robust method (Arjovsky et al., 2017). WGANs still suffer from unstable training, slow convergence, and overly simplistic generator representations. Previous studies have

proposed the possibility of the addition of a gradient penalty to the loss function, but similar issues still arise (Gulrajani et al., 2017). Here we propose a novel method using Bhattacharrya distance to determine which metric to use at different points along the path to convergence. This algorithm will enable rapid convergence when there is little overlap between the model and real data distribution using the Wasserstein-1 metric, and then robust convergence to the real data distribution when there is significant overlap using Jensen-Shannon divergence. This new algorithm will be outline in Section 5.

2 Background

The promise of deep learning is to discover rich, hierarchical models that represent probability distributions over data sources encountered in artificial intelligence, such as images, speech audio files, and symbols encountered in natural language (Goodfellow et al., 2014). Until recently, much of the successes in deep learning had come from discriminative models mapping high-dimensional, rich sensory input to class outputs (Hinton et al., 2012). These successes have been primarily based on the backpropogation and dropout algorithms, using piecewise linear neural units which have a particularly well-defined gradient. Deep generative models have not had much success due to the difficulty of approximating many intractable probabilistic computations that arise in maximum likelihood estimation and the problems dealing with distributions supported by low dimensional manifolds in high-dimensional spaces. Generative Adversarial Networks (GANs) have been proposed to bridge this gap, and utilize the successes of discriminative models to create powerful generative models.

2.1 Generative Adversarial Networks

GANs consist of two models:

- A discriminator D learns to determine whether a sample is from the model distribution or the data distribution. D estimates the probability of a given sample coming from the real data set with a probability of 0 meaning the sample did not come from the real data set and with a probability of 1 meaning the sample did come from the real data set.
- A generator G outputs synthetic samples given a noise variable input z (z brings in potential output diversity). G attempts to create samples that appear to be from the data distribution so that it can fool D into classifying model samples as samples from the real data set.

These two models compete against each other during the training process in a zero-sum game: G is trying to trick the discriminator, while D is trying not to be fooled. This iterative game motivates both of the models to improve their accuracy, and eventually leads to the creation of powerful generators that create remarkably realistic looking samples to human tests. In essence, the GAN architecture uses Turing's original test as a training policy to create a computer model that can successfully produce outputs that appear to be genuine.

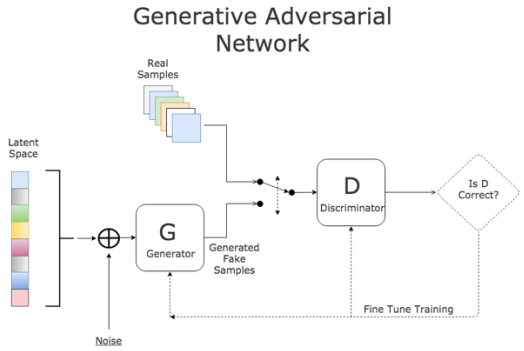

Figure 1: Architecture of a generative adversarial network.

(Image source: https://www.kdnuggets.com)

The GAN architecture is composed of D, G with the following probability distributions: given P_r the data distribution over the real sample x, and P_z the data distribution over the noise input z (usually just uniform), we obtain P_g the generator's distribution over the data x given by $G(z), z \sim P_z$. On the one hand, we want to make sure the discriminator D's decisions over the real data are accurate by maximizing $\mathbb{E}_{x \sim P_r}[\log D(x)]$. Also, given a fake sample $G(z), z \sim P_z$, the discriminator should output a probability, $D(G(z))$, close to zero by maximizing $\mathbb{E}_{z \sim P_z}[\log(1 - D(G(z))]$.

On the other hand, the generator is seeking to increase the chances of D producing a high probability for fax examples, thus to minimize $\mathbb{E}_{z \sim P_z}[\log(1 - D(G(z))]$. Combining these two different goals together, we see that D and G are playing a minimax game with the following loss function

$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{z\sim P_z}[\log(1 - D(G(z))]$$
$$= \mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1 - D(x)]$$

In the section 2.3 we will show that this loss function reaches its minimum value when $P_g = P_r$.

2.2 Kullbacker-Liebler and Jensen-Shannon Divergence

There are a number of objective functions that can be used to optimize GANs. Two of the most fundamental are the Kullbacker-Liebler (KL) and Jensen-Shannon (JS) divergence. Both of these metrics can be used as policy gradients to update the weights of the generator. The distributions mathematical definitions are given below:

(1) *Kullbacker-Liebler (KL) divergence measures how one probability distribution p diverges from a second expected probability distribution q.*

$$D_{KL}(p||q) = \int_x p(x) log \frac{p(x)}{q(x)} dx$$

D_{KL} achieves the minimum zero when p(x) == q(x) everywhere. Note that the formula for

KL divergence is asymmetric. In cases where p(x) is close to zero, but q(x) is non-zero,

q's effect is disregarded. Furthermore, when q(x) is zero, p(x) must also be zero.

(2) *Jensen-Shannon (JS) divergence measures the similarity between two probability distributions and is bounded by [0,1]. JS divergence is symmetric and generally more smooth providing a better policy gradient.*

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2}D_{KL}(p||\frac{p+q}{2})$$

2.3 Proving Bounds

The loss function of a GAN finds its global minimum at the value when $P_g = P_r$.

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

Proof. Given any G, the training criterion for the discriminator D is to maximize L(G,D).

$$L(G,D) = \int_x p_{data}(\boldsymbol{x}) \log\big(D(\boldsymbol{x})\big)\, dx + \int_z p_z(\boldsymbol{z}) \log\,(1 - D\big(G(\boldsymbol{z})\big))dz$$

$$\int_x p_{data}(\boldsymbol{x}) \log\big(D(\boldsymbol{x})\big) + p_g(\boldsymbol{x}) \log\,(1 - D(\boldsymbol{x}))dx$$

Given a function $y \to alog(y) + blog(1-y)$, the maximum in [0,1] is given by $\frac{a}{a+b}$.

□

Thus, when $P_g = P_r$, $D_G^*(\boldsymbol{x}) = \frac{1}{2}$ for all $\boldsymbol{x}$.

3 Metrics

There are a large variety of divergence measures and statistical distance metrics that can be introduced to GANs to inform a policy gradient. The various measures each constitute a convergence path; however, not each of the metrics will guarantee convergence under all conditions. In particular, a weaker induced topology will allow a larger array of model distributions to converge to a given data distribution. A weak topology's set of convergent distributions will be a superset to that of a strong topology (Arjovksy, 2017). Therefore, a metric that induces the weakest topology is the most desirable as it will induce the largest convergent set over the space of all possible distributions.

One of the simplest measures of distance is the metric of *Total Variation (TV)*. Total Variation provides a measure of the difference between the two probability measures P and Q by finding the largest possible difference between the possibilities that the two probability distributions can assign to the same event $d(P, Q) = \max_{A \in F} |P(A) - Q(A)|$. This measure can be useful to determining which part of the distribution requires the most revision; however, it is computationally intractable as it requires us to iterate over all possible values in the distribution X (Gulrajani et al., 2017).

Kullback-Leibler (KL) divergence is derived from statistical physics and constitutes and information theoretic transform from one distribution Q to another P. In essence, it provides a measure of the surprise of distribution P (typically the actual distribution), given distribution Q (the observed distribution) with a measure of 0 signifying that P was completely expected given Q, and thus $P == Q$, while with a measure of 1 signifying that P was completely unexpected given Q (Arjovsky, 2017). This measure, while it can be computed directly in a GAN and used a policy gradient, is often looked over in favor of Jensen-Shannon (JS) divergence due to its asymmetry.

Jensen-Shannon divergence was the original metric proposed by Ian Goodfellow in the breakthrough paper on GANs. The Jensen-Shannon Divergence is symmetrical and well-defined throughout the distribution:

$$JS(P_r, P_g) = D_{KL}(P_r||P_m) + D_{KL}(P_g||P_m)$$

However, Jensen-Shannon Divergence runs into many of the same issues as KL-Divergence, namely vanishing of the gradient when low-dimensionality manifolds are modeled in high-dimensional representation space, and mode collapse where a small subset of the total valid representation space is converged to and overproduced by the generator. Both of these problems pose an issue for generator convergence, especially early on the training. In order to ameliorate some of these concerns, the Wasserstein-1 metric was proposed as a better measure of distance between disjoint distributions.

4 Wasserstein GAN

The Earth-Mover (EM) or Wasserstein-1 distance metric is given by

$$W(P_r, P_g) = \min_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[\,||x - y||\,]$$

where $\Pi(P_r, P_g)$ denotes the set of all joint distributions γ whose marginal are respectively P_r and P_g. Intuitively γ indicates how much "mass" must be transported from *x* to *y* in order to transform the distributions P_r into the distribution P_g. The Wasserstein distance is the "cost" or "work" need for the optimal transport plan. In many cases this metric is better behaved than JS-Divergence, especially in low-manifold supports constituted in high-dimensionality representations because the gradient is Lipschitz and continuous throughout.

Lipschitz implies that the norm of the gradient is bounded by some real value K. By establishing an arbitrary compact set with window sizes, the Arjovsky et al., bound the norm of the gradient to be less than 1, giving a Lipschitz-1 criterion. This limitation enables faster compute speed-up time as well as a guaranteed well-behaved gradient policy function that can be iterated over at any point to find the minimum of the loss function. However, once P_g nears convergence with P_r, the model probability distribution often fails to converge to P_r because the Wasserstein metric provides an inaccurate step-size. Here, it would be better to use JS-Divergence when there is significant overlap and when the step size is appropriate for near convergent distributions.

5 Dynamic Heuristic Proposal

Here we propose a new algorithm for learning a probability distribution. Given some of the faults of previous algorithms, we believe a dynamic heuristic using both a Wasserstein metric during early convergent sequences and a JS-Divergence metric when there is significant overlap between P_r and P_g would enable a faster rate of convergence than either of the two metrics independently, and would constitute a more robust gradient policy for convergence.

The algorithm begins by using the Wasserstein metric to update the policy gradient:

$$W(P_r, P_g) = \min_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[\,||x - y||\,]$$

Once the model and real data distribution reach a significant amount of intersection, then the algorithm begins to use the JS-Divergence metric to update the policy gradient:

$$JS(P_r, P_g) = D_{KL}(P_r||P_m) + D_{KL}(P_g||P_m)$$

This change-over is determined by the Battacharyya Distance metric. The Battacharyya Distance is given by:

$$BC(P, Q) = \int \sqrt{p(x)q(x)}dx\,, \quad 0 \le BC \le 1$$

The Battacharyya Coefficent (BC) measures the similarity of two probability distributions by providing the proportion of each distribution that falls into the overlap. When the BC is high (more than 0.7), this signifies that there is a very high proportion of overlap (roughly approximates more than 70% overlap between both distributions), while when the BC is low (less than 0.3), this signifies that there is very little overlap. We propose that when the

BC reaches a threshold of 0.5, the algorithm switches from a Wasserstein metric to JS-Divergence.

Algorithm 1 Battacharyya-GAN, our proposed algorithm.
Require: α, the learning rate. *c,* the clipping parameter. *m,* the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: w_0, the initial critic parameter. θ_0, the initial generator's parameters.

while θ has not converged **do**
  for $t = 0, \ldots, n_{critic}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim P_r$ a batch from the real data
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim P_z$ a batch of prior samples
    $BC \leftarrow \sum_{i=1}^{m} \sqrt{D(x^{(i)}) * D(g_\theta(z^{(i)}))}$
    if $(BC \leq 0.5)$ **do**
      $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))]$
    else do
      $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^{m} logD(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(g_\theta(z^{(i)})\right)\right)]$
    end if
    $w \leftarrow w + \alpha * RMSProp(w, g_w)$
    $w \leftarrow clip(w, -c, c)$
  end for
  Sample $\{x^{(i)}\}_{i=1}^{m} \sim P_r$ a batch from the real data
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim P_z$ a batch of prior samples
  $BC \leftarrow \sum_{i=1}^{m} \sqrt{D(x^{(i)}) * D(g_\theta(z^{(i)}))}$
  if $(BC \leq 0.5)$ **do**
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  else do
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(g_\theta(z^{(i)})\right)\right)$
  $\theta \leftarrow \theta + \alpha * RMSProp(\theta, g_\theta)$
end while

6 Conclusion and Further Work

Using the BC coefficient in order to create an algorithm that uses the best of both the Wasserstein metric and the JS-Divergence metric enables us to theoretically have a faster rate of convergence with higher robustness. In particular, the Wasserstein metric is computationally tractable and is effective at guiding the model distribution P_g towards P_r early on in training as it requires no overlap in the two probability distributions (which is unlikely with low-manifold supports in a high-dimensional representation space). The JS-Divergence metric on the other hand is computationally efficient when there is significant overlap and provides sound and robust convergence to P_r once the two distributions are locally similar. Further work into this would include empirical measures to validate the claims made in this paper on the robustness and efficiency in convergence of the proposed algorithm.

Acknowledgements

I would first like to thank Professor Richard Granger for advising me. Ever since I requested to work as a research assistant in his lab freshman year, he has been more than willing to accommodate my varying interests. His door has at all times been open for me, whether for practical coursework considerations or philosophical discussions on the trajectory of history. I have always left his office having learned something new and interesting.

I would also like to thank the Computer Science faculty as a whole for four years of academic advising and guidance in and outside of the classroom. Finally, I would like to thank my parents, Scott and Kelly Kenneally, for supporting me in all of my endeavors.

References

Martin Arjovsky and L'eon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017. Under review.

Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *CoRR*, abs/1505.03906, 2015.

Aude Genevay, Marco Cuturi, Gabriel Peyr´e, and Francis Bach. Stochastic optimization for large-scale optimal transport. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3440–3448. Curran Associates, Inc., 2016.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Sch¨olkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.,* 13:723–773, 2012.

Ferenc Huszar. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *CoRR*, abs/1511.05101, 2015.

Shizuo Kakutani. Concrete representation of abstract (m)-spaces (a characterization of the space of continuous functions). *Annals of Mathematics,* 42(4):994–1024, 1941.

Hinton, G., Deng, L., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012a). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(6), 82–97.

Bengio, Y., Mesnil, G., Dauphin, Y., and Rifai, S. (2013a). Better mixing via deep representations. In *ICML'13.*

Gulrajani et al., (2017). Improved training of Wasserstein GAN. In *International Conference on Learning Represetnations*. Under Review.

Alan Turing. Computing machinery and intelligence. *Mind,* 49: 433-460, 1950.