Reinforcement learning (CS6700)
Paper critique report - Generative adversarial networks

Aravind S
EE14B013

27th Jan. 2017

Abstract

With the advent of improved computing resources, deep learning community started focussing on generative modelling. Discriminative models are those where the goal is to predict the output given some representation of the input i.e you model the distribution $p(y|x)$. Generative models, on the other hand model the joint distribution of the (input, output) from which the data is sampled i.e $p(x, y)$ is learnt. Recently, adversarial set ups became popular and are an active area of research. Generative adversarial networks is a framework where a generative model is learnt by interacting with a discriminator in the aim that after sufficient epochs of training, the generative model captures the distribution from which the training data is sampled.

1 Summary

In this framework, 2 models are used - a **Generator** G and a **Discriminator** D. Both of them are neural networks. Say, the data is obtained from a true distribution $p_{true}(X)$ and our goal is to make sure that the generator G learns this distribution as much as possible. Now the training data is samples from $p_{true}(X)$. The role of the generator (a deep neural net) is to generate x' from a random input (sampled from a prior distribution $p_z(Z)$) z. Generally, the prior distribution $p_z(Z)$ is chosen to be uniformly random. This is to ensure sufficient variety in the inputs passed to the generator. In other words, these inputs are "codes" for the input which the network sees. More variety in the code, easier to learn the true distribution's manifold. Also, more variety in the code also results in a rich variety in the samples generated from the network.

Consider a training data point x. The discriminator is another neural network which takes x and x' (true and fake points as inputs) and scores the data point based on whether it is sampled from the distribution $p_{true}(x)$ i.e ideally x should get a score of 1 and x' should get a score of 0. Based on these constraints the network is trained and weights of both G and D are learnt using gradient decent. During the course of training, the generator G learns the distribution $p_g(X)$ and tries to fool D by generating data points indistinguishable from real data points. Simultaneously, D learns to classify the given data point as real or fake, in turn resulting in training a better G. So, G and D play a **minimax game** with a value function $V(G, D)$.

$$\underset{G}{min}\ \underset{D}{max}\, V(G, D) = E_{x\sim p_{data}(X)}[logD(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))]$$

Ultimately, if G, D has sufficient representative power, G will learn the distribution $p_{data}(X)$ so well that, $D(G(z)) = D(x) = 0.5$. Once learnt, new samples can be generated by sampling z from $p_z(Z)$ and feed-forwarding it through the generator which generates a sample x. Altering the prior distribution $p_z(Z)$, leads to different solution and hence different types of samples are generated.

2 Criticism

- Convergence of GANs
  - One possible issue with GANs is even though theoritical convergence is proven, practically training them is difficult. The loss functions for the discriminator (L_D) and the generator (L_G) are chosen differently in such a way that eventually $p_g(X) = p_d(X) = p_{true}(X)$. Here $p_d(X)$ refers to the distribution assigned by the discriminator.
  - Such a fixed point exists only when both D and G have **enough capacity** to model the distribution. The theoritical proof assumes D and G having infinite capacity.
  - Even if it exists, converging to that point is tough because both D and G are parallely learnt. Weight updates for θ_g can reduce L_G but increase L_D.
  - The optimal solution is a case of **Nash equilibria** where the solution (θ_g, θ_d) exists such that L_G is minimum w.r.t θ_g and L_D is minimum w.r.t θ_d. Finding this point is **extremely tough** as the cost function is **non-convex** and the number of **dimensions** is **huge**.
- Lack of explicit representation for $p_g(X)$
  - Since, the generator learns the mapping from $p_z(z)$ to x and there is **no closed form** parametric representation for the learnt distribution, **interpretability** still poses a problem.
  - As a result, the generator proves to be a black box where you can sample data points but can't mathematically assess/utilise its understanding.
  - Having a closed form solution will be useful in many situations where better models can be obtained by combining the generator with some other framework, thus leading to better solutions.
- Training GANs
  - Having spoken about convergence, training GANs is yet another art in itself. Since, both the networks are trained simultaneously, there is no guarantee that generated samples become better over time.
  - This is because, discriminative networks recognizes classes **without understanding human perceptible attributes** of it. As a result, generated samples become better in fooling the discriminator and not in understanding the underlying distribution.
  - Optimal solutions can be obtained using different loss functions. Energy based Generative Adversarial Network (EBGAN) is a modification of GAN where the discriminator assigns low energy to regions near the data manifold and higher energy to generated samples. When the discriminator behaves as an energy function, it opens up multiple architectures to explore and other loss functions. It uses an Auto-encoder framework for the discriminator as it is known to learn energy manifolds quite well.
  - Not only that, uneven training of discriminator/generator still poses a problem. Sometimes, the generator overtrains the discriminator. This results in an extremely powerful discriminator and puts the generator into a fix where it can't learn much. Methods like **feature matching**, where expectations of hidden layer's activations of discriminator is compared seem to solve this issue.
- Comments about images generated by GANs
  - GAN has to produce a complex image to fool the discriminator. Based on what manifold the discriminator learns, the generator obtained will be of different types.
  - If the discriminator is a deep neural network and we have a mixed dataset (like say CIFAR-10, where individual pixels take a variety of values), then the generator learns to generate samples which look fuzzy/blurry. Proper finetuning is needed to avoid such generators.
  - This is because the discriminator gets easily fooled by blurry images. Changes to loss functions and parameter tuning can result in better solutions.

- On the other hand, considering datasets like MNIST (where pixels can take binary values), the generated samples are not blurry but there are lot of dots/holes in the output even though the training data has continuous strokes.
- Modifications were suggested in future papers like,
  * Laplace Pyramid Generative Adversarial Networks (LPGAN)
    · Uses a Convolutional neural network as a generator.
    · **Sequence of generators** are trained that improve resolution of the image, refining images from coarse to fine fashion, thus building the **Laplace pyramid**.
    · Each GAN is conditioned on the previous output (the lower resolution one) and takes a random input to generate a higher resolution image.
    · Generates **clearer** pictures (non-blurry).
  * Deep Convolutional Generative Adversarial Networks (DCGAN)
    · A deep convolutional neural network is used as the generator.
    · Generated data samples are more appealing and comparable to real data.
    · kNN classification of generated data points has better accuracy than that obtained from the naive-GAN, which indirectly shows that DCGAN **learns** $p_{true}(X)$ **better**.

3 Future work

- Generating discrete data
  - It is difficult to generate discrete data like text using a GAN. Discrete data generation using RNNs is possible using one-hot encodings and a hidden state with sufficient capacity. On the other hand, using GAN to model sequential data is not straightforward. This is because an RNN has tied parameters which compresses/represents an arbitrary length sequence as a fixed length vector. But a GAN cannot do that in a straight forward.
  - It is a topic of ongoing research and not many articles have been published till now.
  - Sequence of discrete elements can be modelled with GANs using **Gumbel-softmax** distribution. A Gumbel distribution has the PDF $f(x) = e^{-(x+e^{-x})}$. And the model uses $y = softmax(\frac{h+g}{\tau})$ where g follows a Gumbel distribution with zero mean and unit scale and h is the penultimate layer's activation. But the paper shows only a proof of concept by learning simple distributions and explains the fact that sequence learning is possible in an adversarial set up.
  - Recent advancements in GANs like **training using variational divergence maximisation** and **density ratio estimation** can be tweaked to fit sequential data and is a possible direction for future research.
- Efficiency improvements
  - As discussed in the previous section, even though GANs is a theoritically sound concept, practically there is still scope for efficiency improvement.
  - Obvious choices for efficiency improvements include - different **loss functions**, different **frameworks**, different choice of **priors** for z ($p(z)$). Some of these have already been explored like - EBGANs (different loss function), DCGANs (CNN for generator), but it proves to be a topic of further research.
- Avoiding poor solutions
  - One possible problem faced when training a GAN is **mode collapse**. This occurs when the generator learns to produce samples which belong to a very small portion of the manifold and as a result it generates samples which are very similar in nature.

  - **Unrolling** the optimisation of the **discriminator** seems to solve this problem but with increased computation cost. It increases linearly with the number of steps we unroll. Since, this number can be arbitrarily large, there is a need for better ways to avoid mode-collapsed solutions.

- Bridge with conditional models
  - Recently, methods using pixel-wise conditioning (like PixelCNN, PixelCNN++, PixelRNN) became popular in generative modelling. Here, pixels are **generated sequentially** and each pixel is conditioned on the **causal neighbourhood** of it. The joint distribution can be decomposed using chain rule as,

$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_{<i})$$

    where x is an $n \, x \, n$ image.
  - Each conditional distribution is modelled using a CNN/LSTM and techniques like **Markov assumption**, **parameter tying** are used to improve efficiency.
  - Using the idea of **conditional modelling** with GANs is also a topic for further research. An obvious technique is to condition G on **class labels** and hence explore how the model performs when two different class labels are combined etc.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. Generative Adversarial Networks, 2014; arXiv:1406.2661.

[2] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford and Xi Chen. Improved Techniques for Training GANs, 2016; arXiv:1606.03498.

[3] Alec Radford, Luke Metz and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015; arXiv:1511.06434.

[4] Emily Denton, Soumith Chintala, Arthur Szlam and Rob Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks, 2015; arXiv:1506.05751.

[5] Carl Doersch. Tutorial on Variational Autoencoders, 2016; arXiv:1606.05908.

[6] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle and Ole Winther. Autoencoding beyond pixels using a learned similarity metric, 2015; arXiv:1512.09300.

[7] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, 2016; arXiv:1606.03657.

[8] Junbo Zhao, Michael Mathieu and Yann LeCun. Energy-based Generative Adversarial Network, 2016; arXiv:1609.03126.

[9] Luke Metz, Ben Poole, David Pfau and Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks, 2016; arXiv:1611.02163.

[10] Aaron van den Oord, Nal Kalchbrenner and Koray Kavukcuoglu. Pixel Recurrent Neural Networks, 2016; arXiv:1601.0675

[11] Matt J. Kusner and José Miguel Hernández-Lobato. GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution, 2016; arXiv:1611.04051.

[12] Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka and Daan Wierstra. Towards Conceptual Compression, 2016; arXiv:1604.08772.