Theory Exercise

This theoretical task is going to be about Generative Adversarial Networks, which are affectionately known as GAN. As you might know already that GANs have created a lot of buzz in the ML and DL community lately. But most of us don't know what is the real reason behind this hype. Are GANs really useful? Is it a novel idea? Does it even work? Well, these are questions that you will hopefully be able to answer by the end of this exercise.

Task 1

Read and understand the idea behind GANs and try to analyze the hype around it.

The following are some useful links (not all of them are mandatory but are highly recommended):

1. A well written blog describing GANs http://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them
2. Ian Goodfellow talking about GANs at NIPS 2016 https://www.youtube.com/watch?v=AJVyzd0rqdc
3. Recent papers/developments on GANs: http://gkalliatakis.com/blog/delving-deep-into-gans

The video should be enough for you to answer the questions.

Question 1

State the objective function of a GAN (the one proposed by Goodfellow et al.)

The objective function of a Generative Adversarial Network (GAN), as proposed by Ian Goodfellow and colleagues, is a min-max game played between the generator (G) and the discriminator (D). This objective function is typically expressed as follows:

min_G max_D V(D, G) = E_[xP_data(x)] [log D(x)] + E_[zP_z(z)] [log(1 - D(G(z)))]

In words, the discriminator aims to maximize its ability to correctly classify real data (first term) and correctly classify fake data (second term). Meanwhile, the generator aims to minimize this objective function, effectively trying to "fool" the discriminator into thinking that the generated data is real.

Question 2

Describe each term in the objective function of the GAN

The objective function can be divided into two parts, each part corresponding to the cost function for the Discriminator and the Generator respectively.

E_[x~P_data(x)] [log D(x)]: This term corresponds to the Discriminator's objective to correctly classify real data. 'x' is drawn from real data distribution (P_data). The discriminator, D(x), aims to output 1 (meaning 'real') for these inputs. log D(x) is maximized when D(x) is close to 1.

E_[z~P_z(z)] [log(1 - D(G(z)))]: This term corresponds to the Discriminator's objective to correctly classify fake data. 'z' is drawn from a latent space (P_z). G(z) generates fake data, and D(G(z)) is the Discriminator's classification of this fake data. D aims to output 0 (meaning 'fake') for these inputs, so log(1 - D(G(z))) is maximized when D(G(z)) is close to 0.

The Generator's goal is to minimize this whole function, i.e., it wants to generate data that the Discriminator will incorrectly classify as real.

Question 3 (optional)

Can you reformulate the objective function of GAN in terms of Categorical Cross-Entropy? Justify your answer.

Yes, the objective function of a GAN can be reformulated in terms of binary cross-entropy. The binary cross-entropy is used in the loss functions of both the Generator and Discriminator, which is a special case of the more general categorical cross-entropy. Binary cross-entropy is used because the Discriminator is a binary classifier, determining whether an image is real (from the dataset) or fake (from the generator).

Question 4

Describe the Generator and Discriminator. Be as formal as possible.

Generator (G): The Generator is a function, typically a neural network, that aims to map a latent space (random noise) to the data distribution. In formal terms, if z is a point in the latent space, G(z) is a point in data space. The goal of the Generator is to "fool" the Discriminator into believing that the generated data is real.

Discriminator (D): The Discriminator is another function, typically a neural network, that takes as input a point from the data space and outputs a scalar representing the probability that the input data is real (as opposed to being generated). In other words, D(x) is the probability that x came from the real data rather than the generator.

Question 5

What are the problems that GANs face while training. Describe them if any.

GANs can face several challenges during training, including:

Mode collapse: This occurs when the Generator starts to produce a limited diversity of samples, or even the same sample, regardless of the input noise.

Vanishing gradients: This occurs when the Discriminator becomes too good and the Generator struggles to learn from its feedback, slowing or halting learning.

Non-convergence: GANs involve a dynamic competition between two networks, and as such, they do not always converge to an optimal solution. This means the networks may continue to change without necessarily improving.

Lack of explicit control over the types of samples generated: Unless guided in some way, a GAN can generate any kind of sample that matches the training data, but it may not have any specific features that a user might want.

Question 6

How can you evaluate GANs? Please provide objective answers.

Evaluating GANs is a challenging task due to the absence of a definitive ground truth and the generative nature of the models. However, several methods have been proposed:

Inception Score (IS): This metric calculates how much the generated images contain meaningful objects and how diverse the images are. The limitation is that it's heavily dependent on the model used (Inception model in this case) and does not consider the quality of individual samples.

Frechet Inception Distance (FID): This metric calculates the distance between the feature representation of the real and generated images in the Inception model's space. Lower FID indicates better image quality and variety.

Precision, Recall, and F1 Score: These metrics calculate how much of the generated images are close to the real images (Precision), how much of the real images are covered by generated images (Recall), and the balance between them (F1 Score).

Manual inspection: In many cases, visual inspection of generated samples by humans is used as a subjective evaluation method.

Turing test: An extension of manual inspection, where a human evaluator would try to distinguish between real and generated samples.

Quantitative evaluation: If the generated data has a specific task, like a text-to-image GAN, we can evaluate the generated data based on how well it achieves that task.

Remember, each of these evaluation methods has its own strengths and weaknesses, and there is no single perfect way to evaluate GANs.

Practical Exercise

Task 1:

1.1. Implement a Vanilla GAN as described by Ian Goodfellow in his first paper : https://arxiv.org/abs/1406.2661 to generate adversarial MNIST images.

You can take help from here : https://wiseodd.github.io/techblog/2016/09/17/gan-tensorflow/

Run the code with the original loss as described by Ian Goodfellow and observe the results :

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

1.2. Run the same code with a different loss functions : **Logistic loss** as described in Brandon Amos blog (http://bamos.github.io/2016/08/09/deep-completion/) and compare the results with above (1.1.)

1.3. Run the code for **20k iterations** and then for **100k iterations** and observe the results in both cases. How/why is the output different for both the cases ? Try to find a suitable reason for both .

Also, try to find how the results can be improved.

Check out this video from Soumith Chintala to describing how to train and improve GANs : https://www.youtube.com/watch?v=myGAju4L7O8

1.4. **OPTIONAL**: Using the concept of conditional GANs, try to generate a particular output class image : (8 digit for example in case of MNIST).

HINT: Just change 2 lines of code in previous code

Task 2:

2.1. Train a CNN MNIST model and create adversarial images to classify 4s as 9s. (**VDL Experts**: Use the model already trained in Exercise 1)

HINT: Take help from Jasoni Carter Github : https://github.com/jasonicarter/MNIST-adversarial-images

Try to reason how these adversarial images are produced.

2.2. Now use a random noise image to classify it as 9 and observe the results. Compare the results with 2.1.

2.3. Use a zero image (matrix with all values 0) and classify it as 9. Observe the activations and the results and compare the result with above observations.

Optional Task:

3. **OPTIONAL FOR EXPERTS**: Try other GAN methods like DCGANs with other datasets like Celeb dataset and observe the results.

https://github.com/jasonicarter/MNIST-adversarial-images

Acknowledgement

We would like to thank **Saurabh Varshneya** and **Kumar Shridhar** for making this very interesting exercise on GANs. For any queries, contact: **Saurabh Varshneya** (varshney@rhrk.uni-kl.de)