Name: Sayantan Sarkar
Roll No: 21ME10073
Indian Institute of Technology, Kharagpur

KDAG Neural Art Selections

Task: 2

Topic: Mathematics behind GANs

GAN is not a single model but is a combination of two separate models:

- The Generator(denoted by G), and
- The Discriminator model

These are two methods actually for predicting models in Machine Learning.
The D method is the most famous. Here the model learns the conditional probability of the target variable given the input variable.

P (Y | X = X)

In G model, the model learns the joint probability distribution of the input and the output variable. If it wants to predict something, it uses the Bayes' theorem and computes the conditional probability of the target variable given the input variable.

P (X , Y) = P (X | Y) * P (Y)
P (Y | X) = P (X , Y) / P(X)

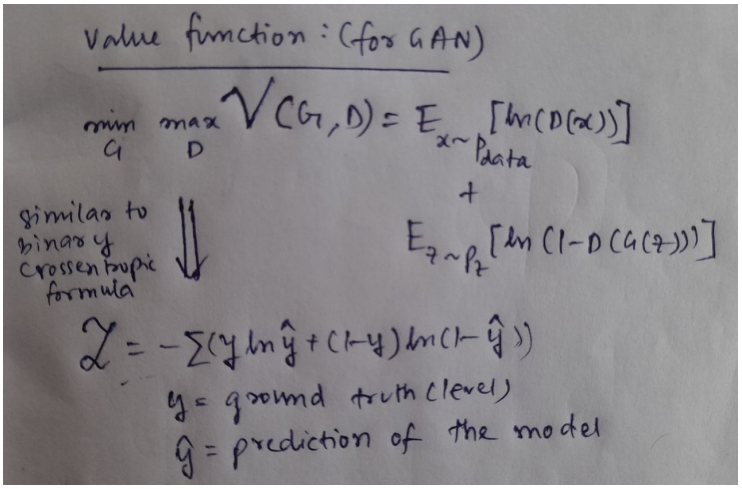
Eg. The naive Bayes' model

Advantage of G model over D model:

We can use the generative model to make new instances of data as here we are learning the distribution function of the data itself. In GANs, G Model produce new data points which are but fake and using the Discriminator model, it tells whether a given data point is an original one or produced by a G module. But this is not possible using a D model.

The two models, G and D, work in an adversarial setup, i.e, they compete with each other and in this process, eventually both of them gets better in their respective jobs.

The G and D in the above illustration are the multilayered neural networks. Here, Θ_G and Θ_D are the weights. We use neural networks as they can approximate any function(from universal approximation theorem).
Some random data is sampled from the noise distribution and fed into the generator. So, here the input is the noise(Z) with no information. The output obtained is G(Z)
p_{data} —> Probability distribution of the original data
p_z —> Probability distribution of the noise
p_g —> Probability distribution function for the output of the generator.
The original and reconstructed data is passed to D model. This will provide us a single no. which will tell the probability of the input belonging to the original data. So, the discriminator is a binary classifier and for the training purpose when we are putting up the original data to the discriminator, say y = 1 and when we pass the reconstructed data, say y = 0 to it. D will try to maximise the chances of predicting correct classes and G will try to pull D. So G and D basically plays a 2-player minimax game (like tic-tac-toe) where we can interpret the objective as one player is trying to maximize its probability of winning. The other player is minimising the likelihood of winning of the first player.
But what is maximised/minimised? Its a mathematical expression called the value function

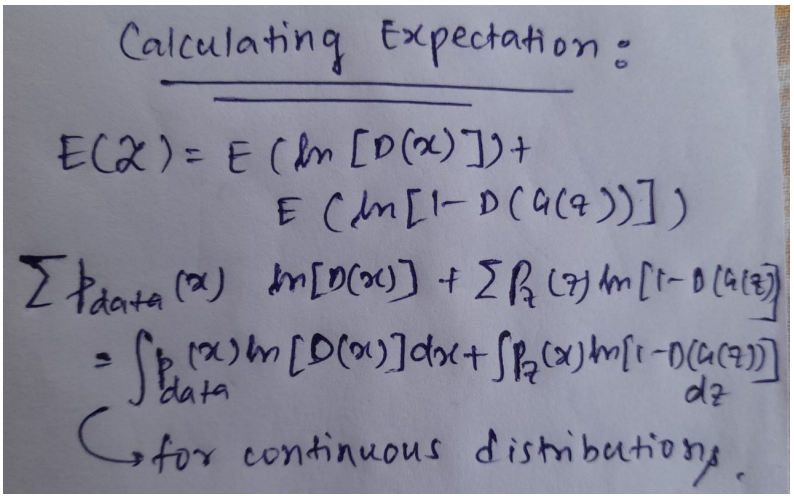


Carefully observing the above expression, the value function of GAN closely resembles to binary crossentropy function (for one input)

Now when y = 0, ŷ = D(G(z)). Thai is because firstly, we pass the noise to the G and it has produced some output. Then we are giving the produced fake data to the model D. Adding them, we get an expression looking similar to the value function for GAN. The capital E's in the value function are expectations.

Why Expectations? The mathematical expressions used are valid for only one datapoint but we have to do it or use it for the entire dataset. To represent that mathematically, we need something called expectations.

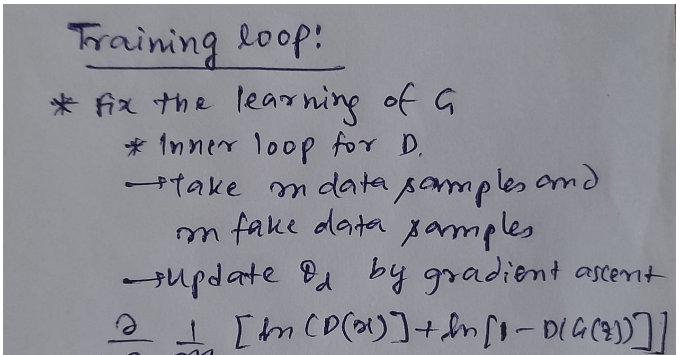
Expectation is the average value of some experiment, if the experiment is performed for a large no. of times. For e.g.,if a dice is rolled for infinite no. of times, then expected score = $\frac{1}{6}(1 + 2 + 3 + 4 + 5 + 6) = 3.50$ .That is probability of each occurrence multiplied by the sum of all possible occurrences. This resembles kindof weighted mean, in a way.



Now, this is only true for a discrete distribution. If we assumed that p_{data} and p_z are continuous distributions then the summation sign is replaced by the integral sign and dx and dz are added at the end of respective terms. This is, in short, written as letter E.

How to optimize this in practice?

Training loop:



Just like every other neural networks, we have

to optimize the loss function using some stochastic process. Here, stochastic gradient descent process is used.
The inner loop in the above process continues for K steps, updates each time by gradient ascent and moves forward .

Why?

This is because our discriminator is trying to maximize the value function. The ln [D(z)] term is not taken in the second half of the process(i.e., during the learning of G). This is because no G term is there in this term and partial derivative w.r.t Θ_G will equate to zero.

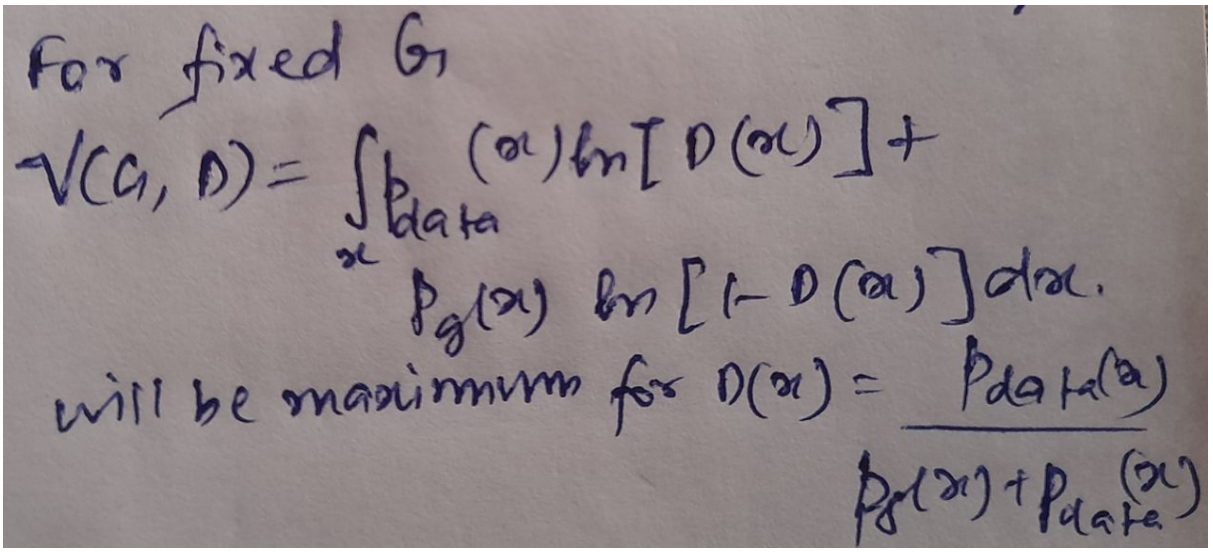
[**Note:** For every K updates of D, we are updating the G once]

What is the guarantee that G will always replicate the distribution?

So, for determining the truth of this, we have to prove that pg will converge to pdata when our G is able to find the global minima of the value function

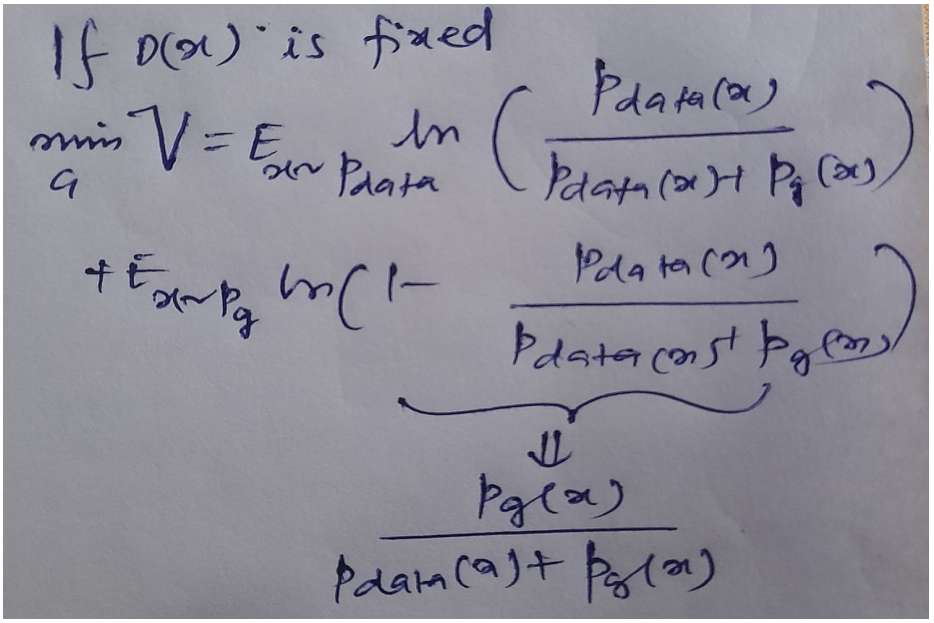
OR

We have to show the global optimality that P_g = P_data when Global minimum of the value function is reached.
Now, this is a 2-step process

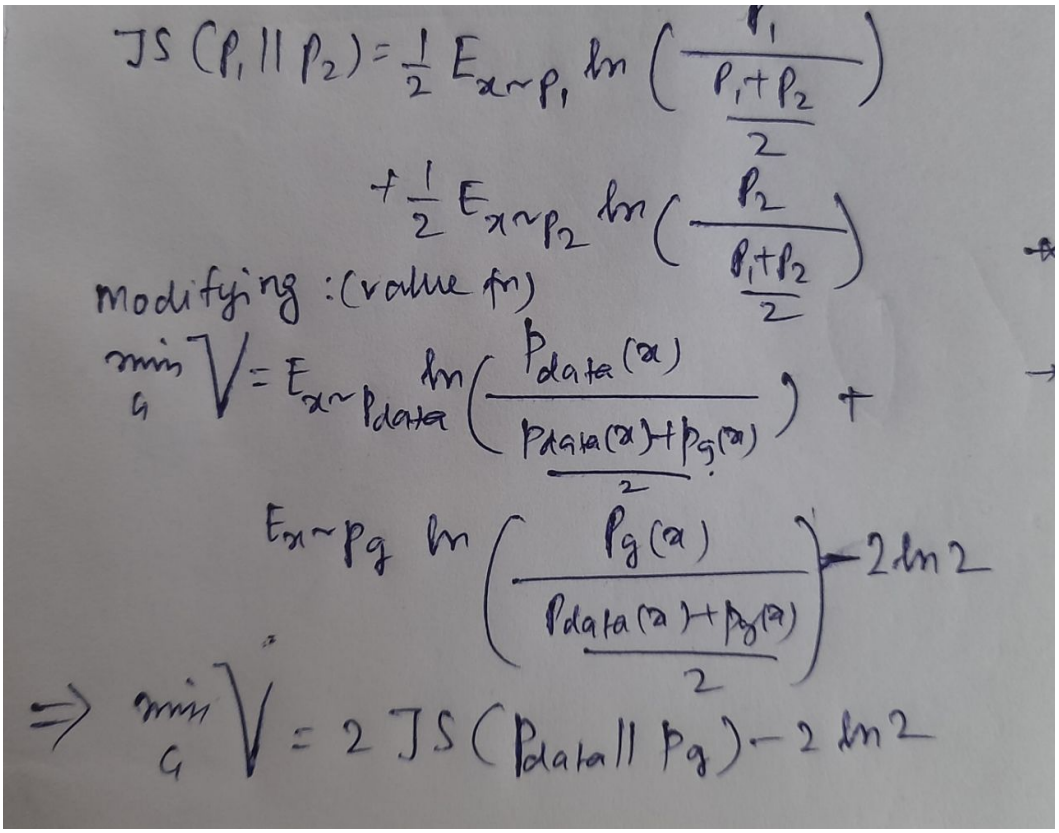


For finding G, we find for what value of D, the value function is maximum. G(z) is replaced by x as the domain of both of them are

same. On differentiating, we see that the maximum value of this expression will occur if $D(x) = \frac{Pdata(x)}{Pdata(x) + Pg(x)}$



We want to prove that the probability distribution of G will be exactly same as the probability distribution of the data. One useful method to measure the difference of distributions is JS Divergence(Jenson-Shannon Divergence):



Now, G wants to minimise this. JS divergence between any 2 distributions cannot be negative. The minimum it cxan achieve is, therefore 0. It will attain 0 only when $p_1 = p_2$.

So, $V_{min} = -2 \ln(2)$ .....possible when $p_{data} = p_g$

How does G attain this stage?

At beginning:

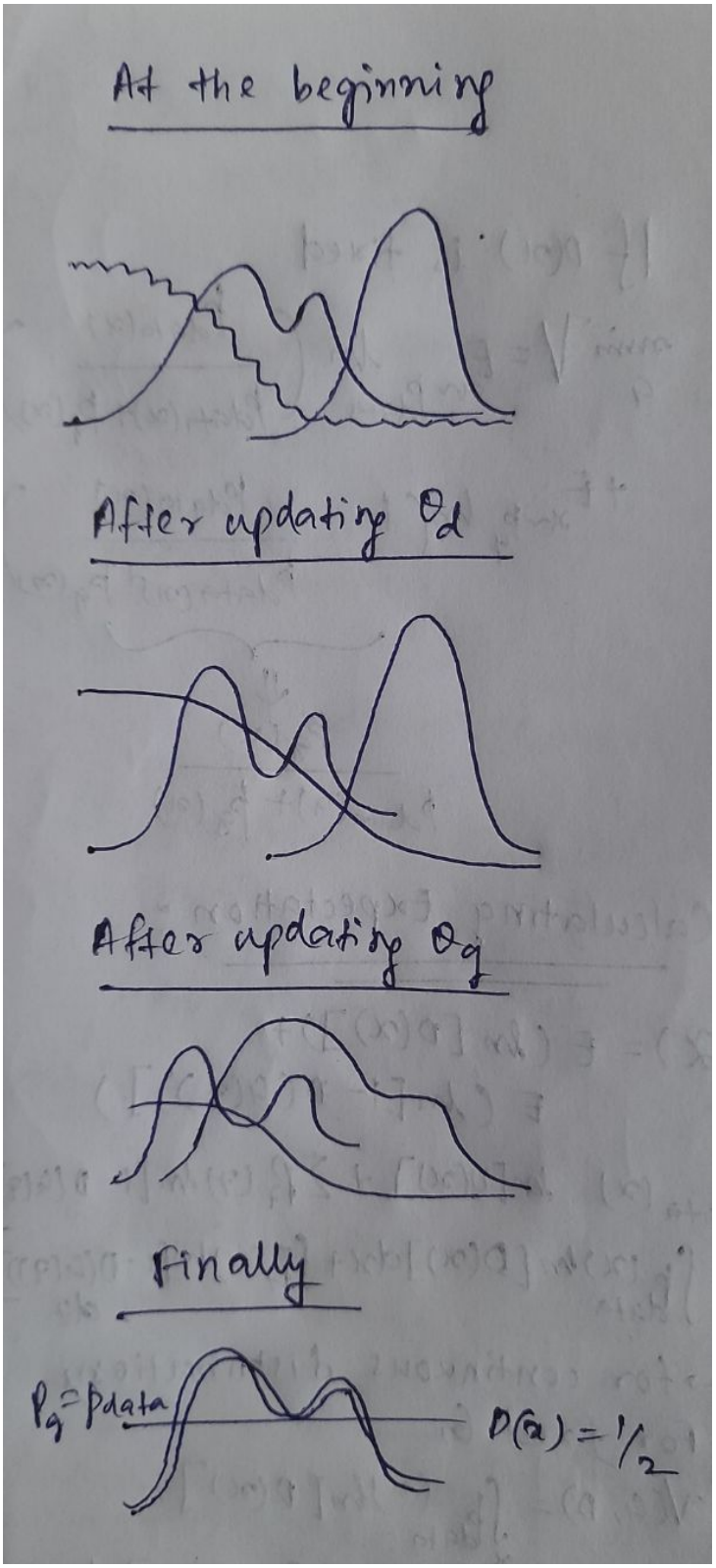
Neither G nor D know what it is doing. So as we can see, p_g is not replicating p_{data} and the classifier discriminator is not classifying as well.

After updating Θ_d:

D has now learned something, so the classifier would work better. Now, D can discriminate between real and fake data

After updating Θ_g:

Now, G has learnt something. p_g is much closer to p_{data} and the discriminator is trying to predict the true level of the data-points but it is not performing as well as it should.



Finally:

G has now attained the minimum of the value function and it has successfully replicated the distribution of the data-points. Its now impossible to tell for the discriminator D to tell which data-point is original and which is fake. Henceforth, D(x) = 0.5 for every input. Thus the nash equilibrium is achieved.