SAPIENZA
UNIVERSITÀ DI ROMA

Artificial Intelligence and Robotics

Vision and Perception

2020 July

Nijat Mursali | 1919669

HOMEWORK 3

Rome, Italy

Exercises

Exercise 2.2

1.Why do we take the expectation of $log(x)$?

As we know from the book, one of the ways to interpret maximum likelihood estimation is to view it as minimizing the dissimilarity between the empirical distribution $\hat{p}_{data}$ defined by the training set and the model distribution, with the degree of dissimilarity between the two measured by the KL divergence. The KL divergence is given by

$$D_{KL}(\hat{p}_{data}||\, p_{model}) = E_{x\sim\hat{p}_{data}}[log\hat{p}_{data}(x) - logp_{model}(x)]$$

The term we have on the left is a function only of the data generating process, not the model. This means when we train the model to minimize the KL divergence, we need only minimize

$$-E_{x\sim\hat{p}_{data}}[logp_{model}(x)]$$

In other ways, we take the expectation of $log(x)$ because VAE maximizes marginal log likelihood by simply reducing the GAP between a lower bound.

2.Explain why the reparameterization trick is needed.

In the following equation *z* is the latent variable that can be calculated as

$$z = z_{mean} + sigma * epsilon$$

Where the *epsilon* simply reparametrizes variational autoencoder networks which is helpful for *mean* and variance to remain as the learnable parameters of the network while still maintaining the stochasticity of the entire system via epsilon.

3.Explain the difference between maximizing the lower bound and the log likelihood. What exactly is the role of KL in the implementation?

In alternative estimation method the log likelihood is approximated by a computationally tractable lower bound. The fundamental idea of the EM algorithm is to find the parameters θ such that it maximize the log likelihood defined as

$$L(\theta) = ln\, p(X;\ \theta) = ln \sum_{Z} p(X, Z;\ \theta)$$

In order to find the parameters, the EM algorithm introduces the distribution $q(Z)$ defined over the latent variables and maximizes a lower bound of log likelihood instead of the log likelihood itself. The fundamental difference between the log likelihood function and its lower bound equals the Kullback-Leibler divergence of the posterior distribution from its variational approximation where we estimate the parameters using the MM algorithm. As it is discussed in lecture video, we can find the similarity between q and p by using Kullback-Leibler (KL) divergence with following formula:

$$KL(q \parallel p) = -\sum q(z) log \frac{p(z)}{q(z)}$$

As mentioned, KL is just a divergence, but not a distance or symmetric and it doesn't satisfy triangle inequality.

4.Explain the difference between cross entropy loss and maximum likelihood.

Despite they are kind of same functions, there are also differences such as maximum likelihood normally represents a structured and principled approach to modeling and training, but cross entropy represents the special cases of that applied to problems that people care about. Thus, the process shows that it is sensible to train model by minimizing the cross-entropy loss since it can lead us to the maximum likelihood of the parameter θ_{ML} that yields the best model according to training examples. In other words, for the probability model that we need to train we use the maximum likelihood, but when we want to minimize the log likelihood we minimize the cross entropy.

5.In the loss function implemented in TF what is it effectively done?

As described in the video, the loss function is computed to find how the real image is distanced from what expected. It's also computed to find how the image generated by the image generator is distanced from what is expected.

Exercise 3.

1.Solve the two following games

$$A_1 = \begin{pmatrix} -2 & 3 \\ 3 & -4 \end{pmatrix} \qquad A_2 = \begin{pmatrix} 0 & -10 \\ 1 & 2 \end{pmatrix}$$

As described in the video, we will be doing the same strategy procedure such as

$$-2p + 3(1-p) = 3p - 4(1-p) = -2p + 3 - 3p = 3p - 4 + 4p \Rightarrow p = 7/12$$

Then if the second player calls 1 average gain which is

$$-2 * \tfrac{7}{12} + 3 * \tfrac{5}{12} = \tfrac{1}{12}$$

and if second player calls 2 first player average gain will be

$$3 * \tfrac{7}{12} - 4 * \tfrac{5}{12} = \tfrac{1}{12}$$

That means the first player has a procedure that guarantees his at least *1/12* on the average, and the second player has a procedure that keeps his average loss to at most *1/12*.

For the second matrix we can do the same step, which will be something like

$$0 * p - 1 * (1-p) = -10 * p + 2(1-p) = 1 - p = -10 * p + 2 - 2 * p \Rightarrow p = \tfrac{1}{11}$$

Then if the second player calles 1 average gain which is

$$0 * \tfrac{1}{11} + 1 * \tfrac{10}{11} = \tfrac{10}{11}$$

and if the second player calls 2 first player average gain will be

$$-10 * \tfrac{1}{11} + 2 * \tfrac{10}{11} = \tfrac{-10+20}{11} = \tfrac{9}{11}$$

Thus, in this case the second player will be having much better opportunity to win the game.

2.Show that if $x \sim Uniform(0, 1)$ is a data sample of size 10×10, further vectorised, then:

$$100 \leq \int_Z p_g(z) \sum_{i=1}^{100} (2x_i^2 + 1) dz \leq 200$$

We need to look at this problem as the special case of the Fubini/Tonelli theorem where it says if $f_n(x) \geq 0$ for all n, x, then

$$\sum \int f_n(x)dx = \int \sum f_n(x)dx$$

without any other further conditions needed. Thus, we can get the summanization sign out of integral and get the integral of $p_g(z)$ which gives us the following result

$$\sum_{i=1}^{100} (2x_i^2 + 1) \int_Z p_g(z)dz$$

Then, we get the 100 different random numbers using uniform distribution between 0 and 1 which will be helping us to compute the summation. We have taken the integral value equal to *1* in this case. In the next step, using the formula $2x_i^2 + 1$ we can calculate the summation that will give us the different types of numbers that we can use to calculate the final result by just summing the values we got before. I have done this assignment using *Python* as I will be including in the document and I have used the *Numpy* library in this case to get the uniform distribution and sum the elements of the array that I got from uniform distribution and summation. As we see from the final result, our result will always be between 100 and 200.

3.Show that if $z \sim N(0, 1)$ and z is a sample of size 10×10, further vectorised, then:

$$0 \leq \int_X p_{data}(x) \sum_{i=1}^{100} z_i^2 dx \leq 200$$

In this exercise, we need to implement almost the same steps we have got in the previous exercise. As we already know, this is special case of the Fubini/Tonelli theorem where it says if $f_n(x) \geq 0$ for all n, x, then

$$\sum \int f_n(x)dx = \int \sum f_n(x)dx$$

without any other further conditions needed. Thus, we can again separate the integral with summation which will give us the following result:

$$0 \leq \sum_{i=1}^{100} z_i^2 \int_X p_{data}(x)dx \leq 200$$

As we implemented in our previous exercise, we again will be using *NumPy* library in *Python* in order to get the normal distribution of random numbers between 0 and 1. Then, we calculated the summation of z_i^2 and store the values in the array in order to calculate the final value. We also took the value of integral as 1, we just compute the value of summation. The final result is simply the sum of all elements inside the array. We also included the script for this exercise as a separate file, as you see the value is always between 0 and 200.

4.Show why if $p_g = p_{data}$ then $JSD = 0$.

As we know, for the optimal discriminator D_G^* ,we have

$$V(G,\ D_G^*(x)) = \ldots = 2D_{JSD}[p_{data}, p_g] - log4$$

and from the symmetric KL divergence we know that

$$D_{JSD}[p,q] = \tfrac{1}{2}(D_{KL}[p,\ \tfrac{p+q}{2}] + D_{KL}[q, \tfrac{p+q}{2}])$$

So, as in the properties it says that $D_{JSD}[p,q] = 0$ *iff* $p = q$.

5.Explain in your words why *GANs* have convergence problems.

The fundamental reason for *GANs not to converge* is because of the problems or imbalances between the discriminator and the generator. For this case, the solution would be to balance their training to avoid overfitting.

6.List 7 major weaknesses of the *GAN*.

As, we already mentioned one of the problems/weaknesses of *GAN* which is **not converging**. However, it's not only a problem that *GAN* could have. There are other weaknesses that *GAN could have such as* **mode collapse** which happens when a generator collapses because of lack of samples, **overfitting** that can cause unbalance between the generator and discriminator, **diminished gradient**, **highly sensitive** to parameter selections, **difficult to train**, problems in **loss function** and etc. can be counted as weaknesses of *GAN*.

7.Tell which application of *GAN* is your favorite one.

There are of course amazing applications of *GAN* that were made till today, but from my point of view my favorite one is semantic image-to-photo translation that simply demonstrates the use of conditional GANs to generate photorealistic images given a semantic image or sketch as input.