Optimization and Machine Learning, Fall 2023
Homework 5
(Due Thursday, Jan 11 at 11:59pm (CST))

1. [10 points] [Deep Learning Model]

   (a) Consider a sequential 2D convolution block consist of 10 layers. Suppose the input size is $4 \times 64 \times 64 \times$ (channel, width, height) and we use 3×3 (width, height) Conv2D with 4 channels input and 4 channels output to convolve with it. Set stride $= 1$ and pad $= 1$. What is the output size? Let the bias for each kernel be a scalar, how many parameters do we have in the ? [5 points]

   (b) The convolution layer is followed by a max pooling layer with 2×2 (width, height) filter and stride $= 2$. What is the output size of the pooling layer? How many parameters do we have in the pooling layer? [5 points]

   (a) Since the input image is $4 \times 64 \times 64$, so $W = 64, H = 64$.
   And since stride $S = 1$, pad $P = 1$, kernel size $F = 3$,
   so the output size is $W_{conv} = \frac{W + 2P - F}{S} + 1 = 64$, $H_{conv} = \frac{H + 2P - F}{S} + 1 = 64$.
   Since we take the **pytorch convention**, the output has 4 channels, so the output size is $4 \times 64 \times 64$.
   Which is exactly the same as the input size.
   And since we have 10 kernels, as 10 layers, and the output size is same as the input size, so the final output size is $4 \times 64 \times 64$.

   For each convolution layer, the kernels have total $4 \times 3 \times 3 = 36$ parameters.
   And each kernel has a bias, which is 1 parameter.
   So the total number of parameters is $10 \times (4 \times (3 \times 3 + 1)) = 400$.

   So above all, the output size is $4 \times 64 \times 64$, and the total number of parameters is 400.

   (b) Since the output size of the convolution layer is $4 \times 64 \times 64$.
   And for the pooling layer, the filter size is $F' = 2$, stride $S' = 2$,
   so the output size is $W_{pooling} = \frac{W_{conv} - F'}{S'} + 1 = 32$, $H_{pooling} = \frac{H_{conv} - F'}{S'} + 1 = 32$.
   So the output size is $4 \times 32 \times 32$.

   And since the pooling layer is a max pooling layer, so there is no parameter in this layer.

   So above all, the output size is $4 \times 32 \times 32$, and the total number of parameters is 0.

2. [10 points] Use the k-means++ algorithm and Euclidean distance to cluster the 8 data points into $K = 3$ clusters. The coordinates of the data points are:

$$x^{(1)} = (2, 8),\ x^{(2)} = (2, 5),\ x^{(3)} = (1, 2),\ x^{(4)} = (5, 8),$$
$$x^{(5)} = (7, 3),\ x^{(6)} = (6, 4),\ x^{(7)} = (8, 4),\ x^{(8)} = (4, 7).$$

Suppose that initially the first cluster centers is $x^{(1)}$.
To ensure consistent results, please use random numbers in the order shown in the table below. When selecting a center, arrange it in ascending order of sequence number. For example, when the normalized weights of 5 nodes are 0.2, 0.1, 0.3, 0.3, and 0.1, if the random number is 0.3, the selected node is the third one. Note that you don't necessarily need to use all of them.

0.6	0.2	0.5	0.9	0.3

(a) Perform the k-means++ algorithm to initialize other centers and report the coordinates of the resulting centroids. [3 points]

(b) Calculate the loss function

$$Q(r, c) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K} r_{ij}||x^{(i)} - c_j||^2, \tag{1}$$

where $r_{ij} = 1$ if $x^{(i)}$ belongs to the j-th cluster and 0 otherwise. [2 points]

(c) How many more iterations are needed to converge? [3 points] Calculate the loss after it converged. [2 points]

(a) We can calculate the other points' Euclidean distance to $x^{(1)}$ is $D(x^{(i)})$, and the probability of selecting $x^{(i)}$ as the next center is $p(x^{(i)})$, which is proportional to $D(x^{(i)})^2$.
So the $D^2(x^{(i)})$ and $p(x^{(i)})$ are shown in the table below.

point	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$	$x^{(7)}$	$x^{(8)}$
$D^2(x^{(i)})$	0	9	37	9	50	32	52	5
$p(x^{(i)})$	0	0.05	0.19	0.05	0.26	0.16	0.27	0.03

We randomly sample a point. The random number is 0.6, and since $\sum_{i=1}^{5} p(x^{(i)}) = 0.55 < 0.6$,

$\sum_{i=1}^{6} p(x^{(i)}) = 0.71 > 0.6$, so we choose $x^{(6)}$ as the second class center.

2. Then, we need to choose the third center.
Suppose that for the i-th point $x^{(i)}$, the Euclidean distance for it to $x^{(1)}$ is $D_1(x^{(i)})$, the Euclidean distance for it to $x^{(6)}$ is $D_2(x^{(i)})$.
So the Euclidean distance to the closest center $D(x^{(i)}) = \min(D_1(x^{(i)}), D_2(x^{(i)}))$. So the $D_1^2(x^{(i)}), D_2^2(x^{(i)}), D^2(x^{(i)})$ and $p(x^{(i)})$ are shown in the table below.

point	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$	$x^{(7)}$	$x^{(8)}$
$D_1^2(x^{(i)})$	0	9	37	9	50	32	52	5
$D_2^2(x^{(i)})$	32	17	29	17	2	0	4	13
$D^2(x^{(i)})$	0	9	29	9	2	0	4	5
$p(x^{(i)})$	0	0.16	0.50	0.16	0.03	0	0.07	0.09

We randomly sample a point. The random number is 0.2, and since $\sum_{i=1}^{2} p(x^{(i)}) = 0.16 < 0.2$,

$\sum_{i=1}^{3} p(x^{(i)}) = 0.76 > 0.2$, so we choose $x^{(3)}$ as the third class center.

So above all, the initialized centers are:
$c_1 = x^{(1)} = (2, 8)$, $c_2 = x^{(3)} = (1, 2)$, $c_3 = x^{(6)} = (6, 4)$.

(b) The center after initialization is:
$c_1 = x^{(1)} = (2, 8)$, $c_2 = x^{(3)} = (1, 2)$, $c_3 = x^{(6)} = (6, 4)$.
And $x^{(1)}, x^{(2)}, x^{(4)}, x^{(8)}$ belong to c_1, $x^{(3)}$ belong to c_2, $x^{(5)}, x^{(6)}, x^{(7)}$ belong to c_3.

So the loss is

$$Q(r, c) = \frac{1}{8}\sum_{i=1}^{8}\sum_{j=1}^{3} r_{ij}||x^{(i)} - c_j||^2 = \frac{(0 + 9 + 9 + 5) + (0) + (2 + 0 + 4)}{8} = \frac{29}{8}$$

So above all, the loss is $Q(r,c) = \frac{29}{8}$.

(c) For the 1-st iteration, we have:
$c_1 = \frac{1}{4}(x^{(1)} + x^{(2)} + x^{(4)} + x^{(8)}) = (\frac{13}{4}, 7)$.
$c_2 = x^{(3)} = (1, 2)$.
$c_3 = \frac{1}{3}(x^{(5)} + x^{(6)} + x^{(7)}) = (7, \frac{11}{3})$.

Then we calculate the Euclidean distance for each point to each center:
i.e. $D_1^2(x^{(i)}) = ||x^{(i)} - c_1||^2$, $D_2^2(x^{(i)}) = ||x^{(i)} - c_2||^2$, $D_3^2(x^{(i)}) = ||x^{(i)} - c_3||^2$.
The row c_j means that the of the corresponding point is to the center c_j.
i.e. for the point $x^{(i)}$, the distance to the center c_j has the smallest Euclidean distance among all centers.

point	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$	$x^{(7)}$	$x^{(8)}$
center c_j	c_1	c_1	c_2	c_1	c_3	c_3	c_3	c_1

And we can see that the center c_j for each point is the same as the previous iteration.
So it converged. i.e. it only needs 1 iteration to converge.

And the loss after it converged is:

$$Q(r,c) = \frac{1}{8}\sum_{i=1}^{8}\sum_{j=1}^{3} r_{ij}||x^{(i)} - c_j||^2 = \frac{(\frac{41}{16} + \frac{89}{16} + \frac{65}{16} + \frac{9}{16}) + 0 + (\frac{40}{9} + \frac{10}{9} + \frac{10}{9})}{8} = \frac{233}{96}$$

So above all, 1 iteration is needed to converge, and the loss after it converged is $Q(r,c) = \frac{233}{96}$.

3. [10 points] Name 2 deep generation networks. [2 points] Briefly describe the training procedure of a GAN model.(What's the objective function? How to update the parameters in each stage?) [8 points]
(a) 2 generation networks: VAE, GAN.

(b) The training procedure of a GAN model:
The GAN model has two parts: generator G and discriminator D.
The generator G is a neural network, which takes a random noise z as input, and outputs a fake image $G(z)$.
The discriminator D is also a neural network, which takes a image x as input, and outputs a probability $D(x)$, which means the probability that x is a real image.
Then we can train the GAN model by training the generator G and discriminator D alternately.
The generator wants to generate a fake image $G(z)$, which can fool the discriminator D.
The discriminator wants to discriminate the real image x and the fake image $G(z)$.

To balance the generator and discriminator, the GAN model has a minimax objective funcion:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

To update discriminator, it wants to discriminate to use gradient ascent to

$$\max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

To update generator, it wants to generator to use gradient descent to

$$\min_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$