Task 1: Sampling

As mentioned in the introduction, in this part of the assignment, we will focus on learning Langevin dynamic sampling. However, before proceeding, it is essential to familiarize ourselves with a concept known as the score function.

Score Function

For the probability distribution with probability density function $P(x)$, score function is:

$$\nabla_x \log p(x)$$

A real-world use case arises when we have a distribution for which the normalization constant ($Z(\theta)$) is unknown. For instance, consider a distribution given by:

$$p(x) = \frac{e^{f_\theta(x)}}{Z(\theta)}$$

given a set of samples to estimate the density, $p(x)$ should be marginalized over x so that they sum up to one and $Z(\theta)$ is calculated. The point is, in many scenarios, x is a high-dimensional data which makes this estimation impossible. Thus we need to be able to get samples from a distribution whose normalization constant is unknown. In the above density function, score function is:

$$\nabla_x \log \frac{e^{f_\theta(x)}}{Z(\theta)} = \nabla_x[f_\theta(x) - \log Z(\theta)] = \nabla_x f_\theta(x)$$

As you can see, we omitted $Z(\theta)$. Langevin Dynamics is the algorithm which takes samples from a distribution, when provided with its score function.

Langevin Dynamics

To take sample using langevin dynamics algorithm, take the following steps:

1. Initialize a point randomly, given an arbitrary distribution $\pi : X^0 \sim \pi(x)$
2. Repeat until convergence(a fixed number of steps):

a. $X^{t+1} \leftarrow X^{t} + \epsilon S_{\theta}(X^{t}) + \sqrt{2\epsilon}Z^{t}$; where ϵ is a hyperparameter specifying step size, S_{θ} is score function and Z^{t} is a gaussian noise, sampled from standard normal distribution

What to do

First, make a 2D gaussian distribution with $mean = [-5, 5]$ and $cov = 5I$ where I is the identity matrix; then plot it. The output should look like the figure 1:

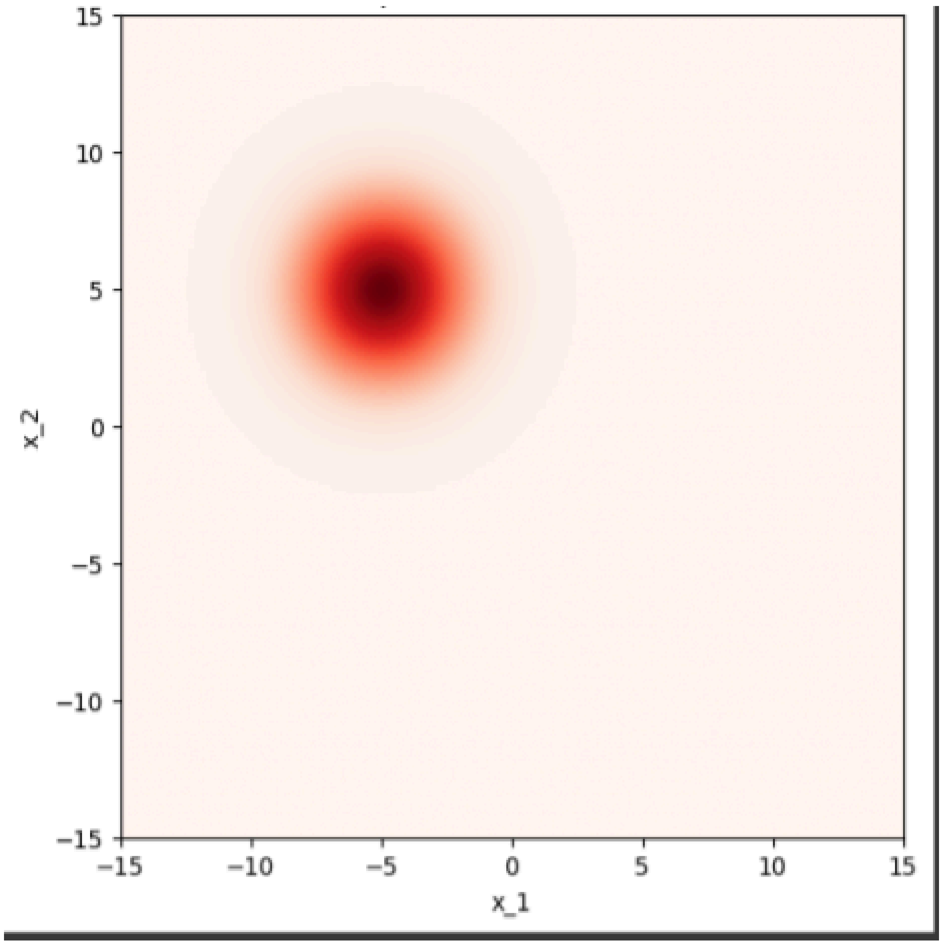

figure1:Heatmap of 2D gaussian

Then you need to implement the score function theoretically with the provided formula and pdf of the distribution. To make sure that you have calculated the score function correctly, you can plot a quiver from points of the grid to their corresponding score. The final result should look something like this:

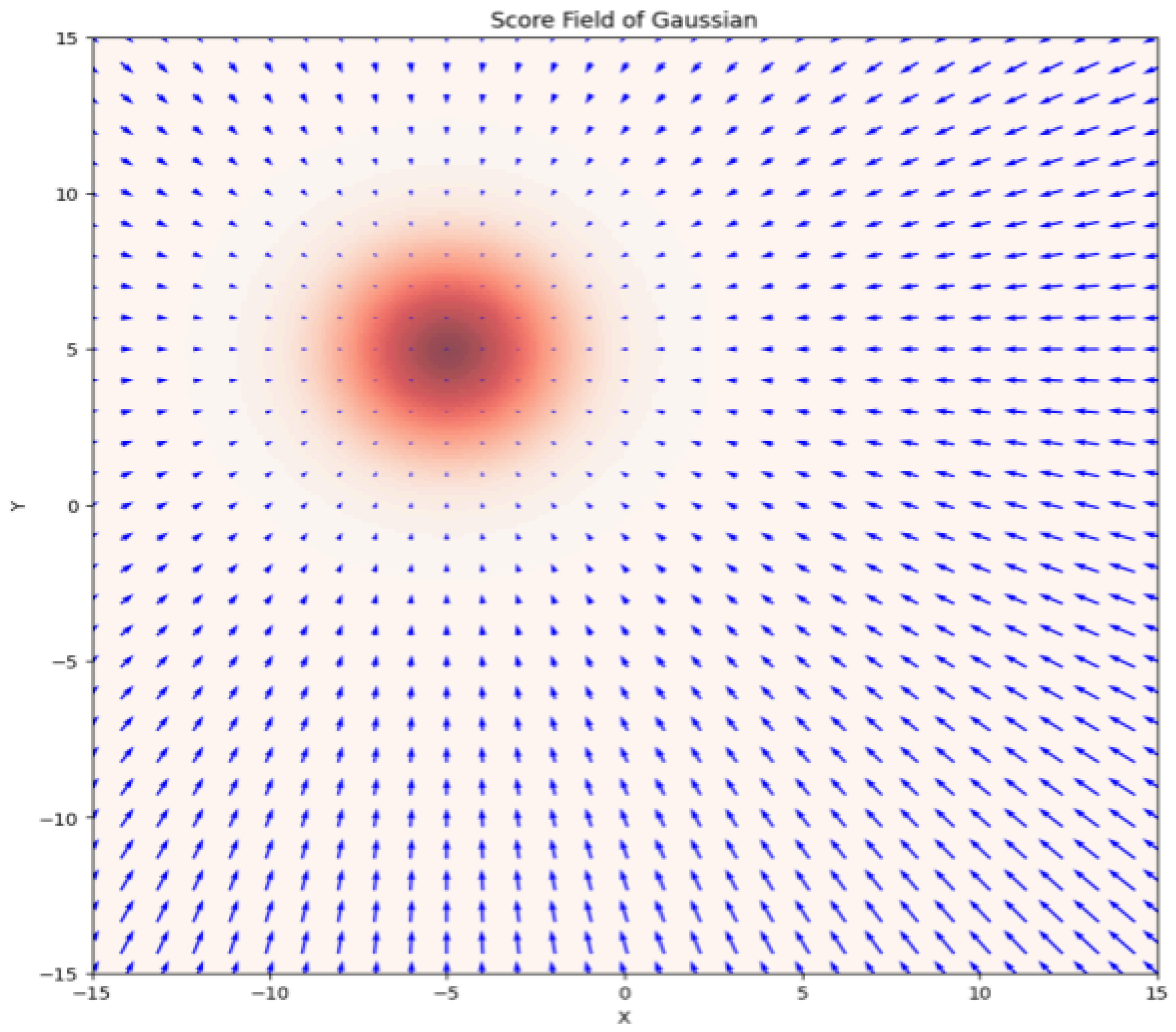

Figure 2: score field of distribution

Now you have all the required utility for langevin sampling. Implement a function which takes an array of initial points and runs langevin dynamics on them and returns the converged samples. To see the progress of the algorithm, you should keep a trajectory of points and plot them:

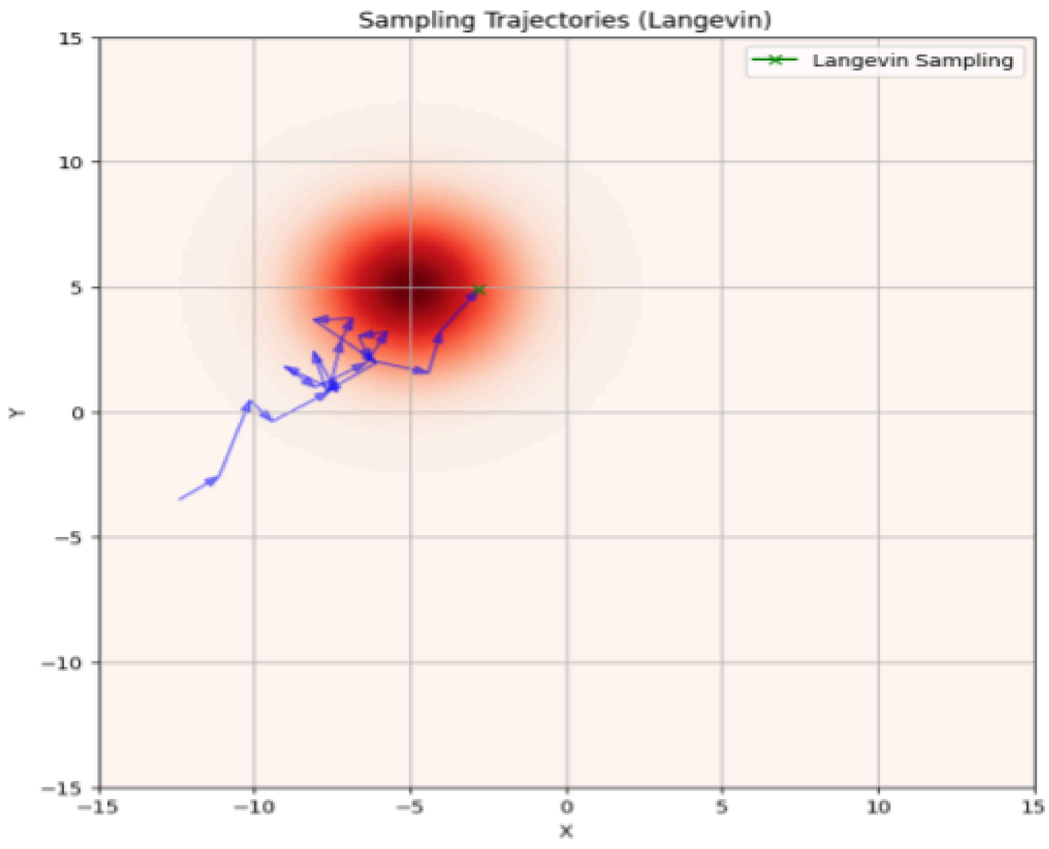

Figure 3:sampling trajectories

Now we want you to take 1000 samples using langevin dynamics and 1000 samples using numpy.random.multivariate_normal method. Compare the visualizations. How else can you compare these two methods?

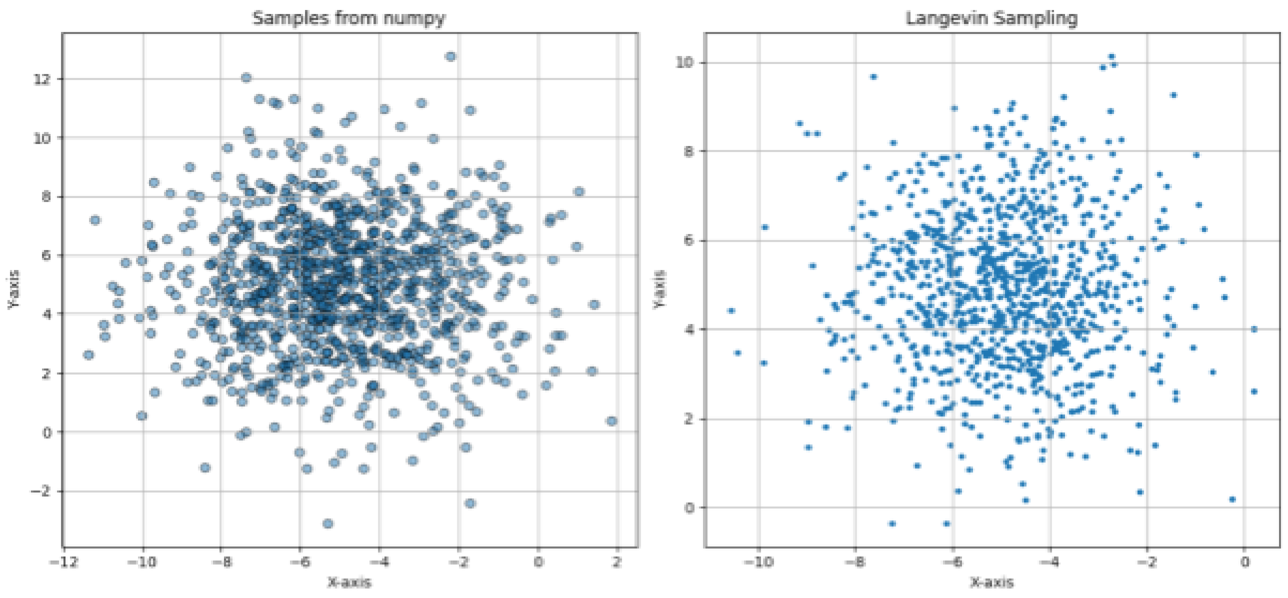

Figure 4: comparing samples

Questions(5% Bonus)

1- In the sampling part, suppose that instead of a gaussian distribution, we have a mixture of gaussians whose density function is:

$$p(x) = \alpha N(x;\ \mu_1,\ \Sigma_1) + (1 - \alpha)N(x;\ \mu_2,\ \Sigma_2)$$

for some $\alpha;\ 0 < \alpha < 1$ and normal distributions $N(x;\ \mu_1,\ \Sigma_1)$ and $N(x;\ \mu_2,\ \Sigma_2)$, will we be able to take proper samples using langevin dynamics algorithm? Justify your answer.