Advanced Machine Unearning: Programming Assignment 4

Tejas Sharma, Muskaan Jain, Ojas Maheshwari

Spring 2025

1 Task 0: Environment Setup and Result Reproduction

```
Using device: mps

--- Model Architecture ---
EnergyRegressor(
    (net): Sequential(
    (0): Linear(in_features=784, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=4096, out_features=2048, bias=True)
    (3): ReLU(inplace=True)
    (4): Linear(in_features=2048, out_features=1024, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=1024, out_features=512, bias=True)
    (7): ReLU(inplace=True)
    (8): Linear(in_features=512, out_features=256, bias=True)
    (9): ReLU(inplace=True)
    (10): Linear(in_features=256, out_features=128, bias=True)
    (11): ReLU(inplace=True)
    (12): Linear(in_features=128, out_features=64, bias=True)
    (13): ReLU(inplace=True)
    (14): Linear(in_features=64, out_features=32, bias=True)
    (15): ReLU(inplace=True)
    (16): Linear(in_features=32, out_features=16, bias=True)
    (17): ReLU(inplace=True)
    (18): Linear(in_features=16, out_features=8, bias=True)
    (19): ReLU(inplace=True)
    (20): Linear(in_features=8, out_features=4, bias=True)
    (21): ReLU(inplace=True)
    (22): Linear(in_features=4, out_features=2, bias=True)
    (23): ReLU(inplace=True)
    (24): Linear(in_features=2, out_features=1, bias=True)
    )
)
------------------------

Loading dataset from ./A4_test_data.pt...
Dataset loaded in 0.18s. Shape: x=torch.Size([100000, 784]), energy=torch.Size([100000, 1])
Using device: mps
Using device: mps

--- Test Results ---
Loss: 288.1554
--- Script Finished ---
```

We set `DATASET_PATH='./A4_test_data.pt'` and loaded the state dictionary of the `EnergyRegressor` model from `MODEL_WEIGHTS_PATH='./trained_model_weights.pth'`. Above is the output on running `get_results.py`.

The files `A4_test_data.pt` and `trained_model_weights.pth` are downloaded from the assignment drive link, and pasted onto the directory containing the python script `get_results.py`.

2 Task 1: MCMC Sampling Implementation

We imported from the task 0 file `get_results.py`, the `EnergyRegressor` class. We also used the `TSNE` class from `sklearn.manifold` to visualize the samples generated by the MCMC sampling algorithm that we implemented. Lastly, we imported the `clock_gettime` function from the `time` library to measure the time taken by our sampling algorithms. The file for task 1 is `sampling_algos.py`.

We assume that this file is in the same directory as the task 0 file, and also has the trained model weights file. On executing this file, we got the following output:

```
Using device: mps
--- Sampling Times ---
Algo-1 Burn-in Time: 26.6275 seconds
Algo-1 Completion Time: 48.6032 seconds
Algo-2 Burn-in Time: 10.4676 seconds
Algo-2 Completion Time: 21.3794 seconds
```

We found that the best results were obtained on setting the value of burn-in to 1000 (1000 samples pre-generated before actual sampling), and 1000 samples were generated after that, which were plotted. The value of τ or the learning rate is set to 0.01.

Above results were generated using these hyperparameters. It is evident that the second algorithm is more than twice as fast as the first, which is understandable considering that we do not evaluate any function (probabilistic) that decides whether we reject or accept the new sample.

2.1 2D and 3D visualization of generated samples

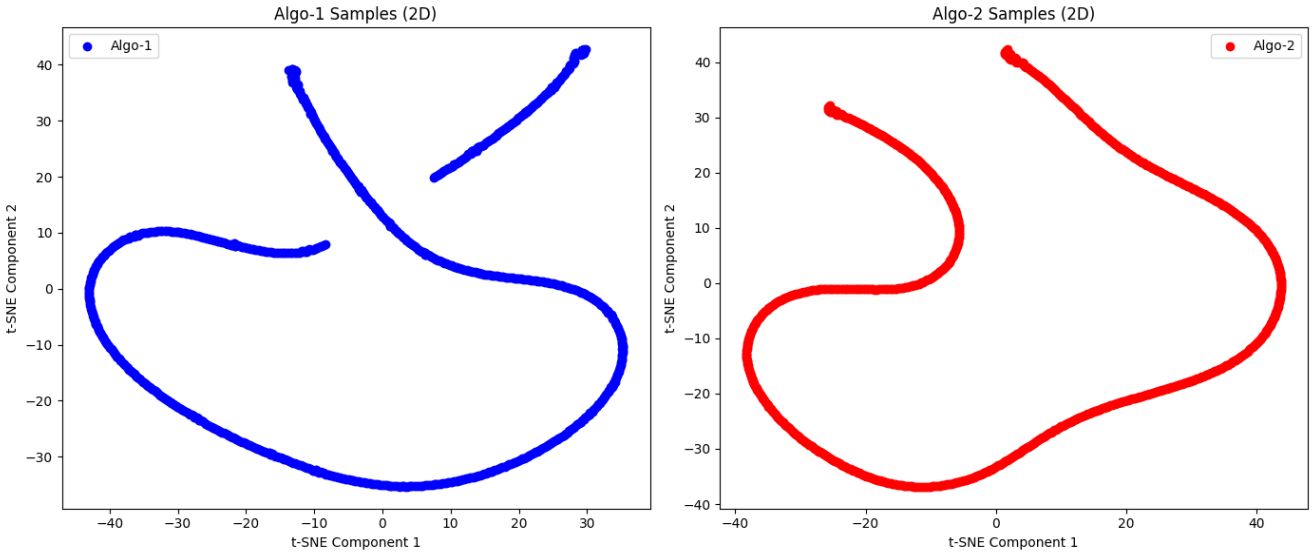

Figure 1: 2D visualization of generated samples

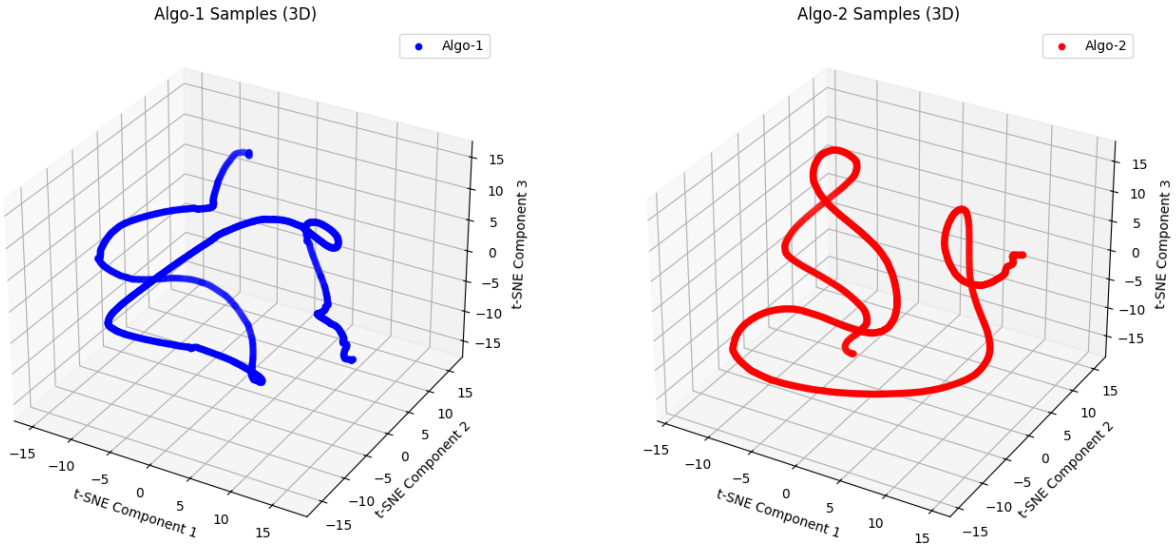

Figure 2: 3D visualization of generated samples