Mathematics for AI Assignment

Laksh Jain

23110185

Problem 1: Comparing Diffusion Maps to other dimensional reduction techniques for clustering the UCI-HAR Dataset

Diffusion Maps is a nonlinear dimensionality reduction technique which captures the geometric structure of high dimensional data by modelling the data as a Markov chain.

It computes a family of embeddings of a data set into Euclidean space (low-dimensional) whose coordinates can be computed from the eigenvectors and eigenvalues of a diffusion operator on the data.

Step 1:

First step is to window the data to reduce the size of the time-series vectors. The dataset providers have already provided the windowed data with window length of 128. Corresponding to every window, we have an activity label.

There are 9 features given: body acceleration, gyro and total acceleration, each in x, y and z directions.

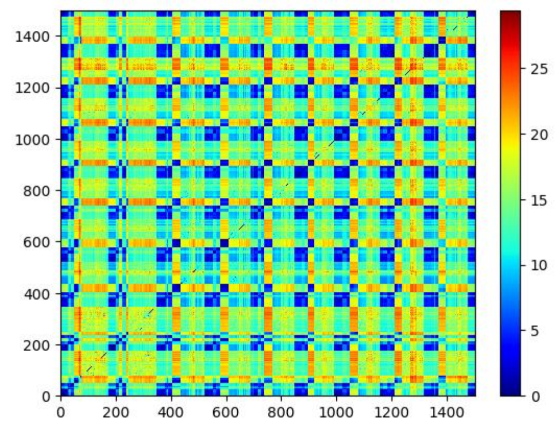
We stack the data to form a matrix of shape (number of examples x window length x number of features)

Due to computation constraints, we only consider first 1500 examples. Hence our signals matrix is of the shape (1500,128,9).

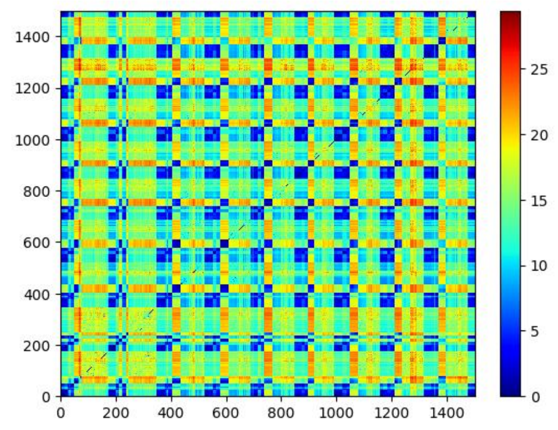
Step 2:

We compute the pairwise Euclidean distance and the n dimensional Dynamic time Warping Distances and construct the similarity matricies.

Euclidean Matrix:

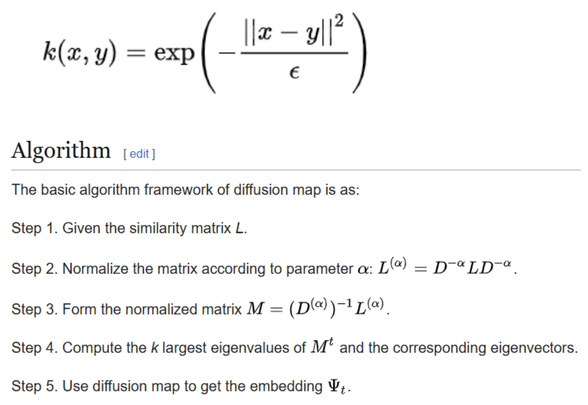


DTW Matrix:



Step 3:

We apply diffusion maps for dimensionality reduction. We follow the algorithm given on Wikipedia:



$$k(x,y) = \exp\left(-\frac{||x-y||^2}{\epsilon}\right)$$

Algorithm [edit]

The basic algorithm framework of diffusion map is as:

Step 1. Given the similarity matrix L.

Step 2. Normalize the matrix according to parameter α: $L^{(\alpha)} = D^{-\alpha} L D^{-\alpha}$.

Step 3. Form the normalized matrix $M = (D^{(\alpha)})^{-1} L^{(\alpha)}$.

Step 4. Compute the k largest eigenvalues of M^t and the corresponding eigenvectors.

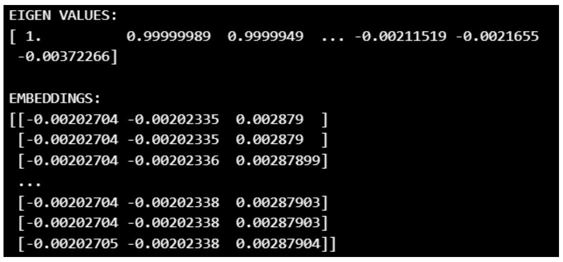
Step 5. Use diffusion map to get the embedding Ψ_t.

Here, L is our diffusion kernel initially and we compute M, known as the diffusion operator. **We compute the following for alpha equal to 1 and different values of epsilon.**

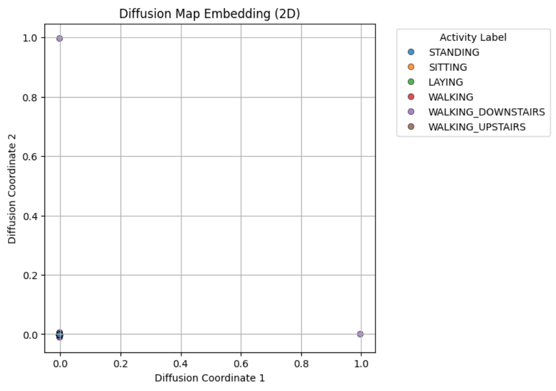
Note that the sum of the rows of M is 1, and it is also known as the Markov Matrix.

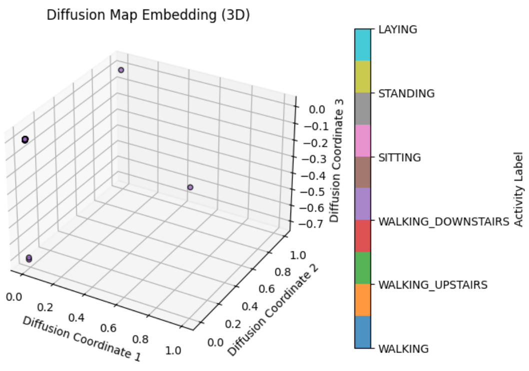
We plot the eigen values * eigen vectors as embeddings. We ignore the first eigen vector(constant vector with eigen value = 1)

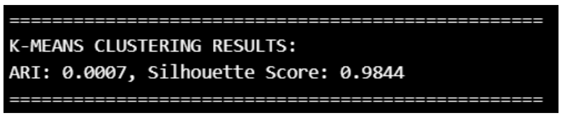
1. Diffusion and clustering for epsilon = median of similarity matrix ^ 1



```
EIGEN VALUES:
[ 1.          0.99999989  0.9999949   ... -0.00211519 -0.0021655
 -0.00372266]

EMBEDDINGS:
[[-0.00202704 -0.00202335  0.002879  ]
 [-0.00202704 -0.00202335  0.002879  ]
 [-0.00202704 -0.00202336  0.00287899]
 ...
 [-0.00202704 -0.00202338  0.00287903]
 [-0.00202704 -0.00202338  0.00287903]
 [-0.00202705 -0.00202338  0.00287904]]
```

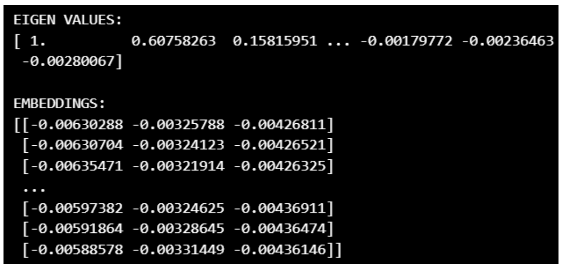




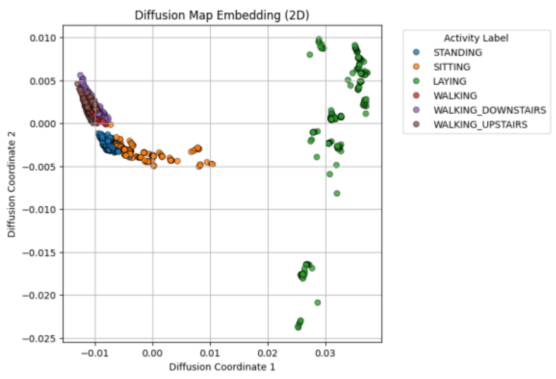


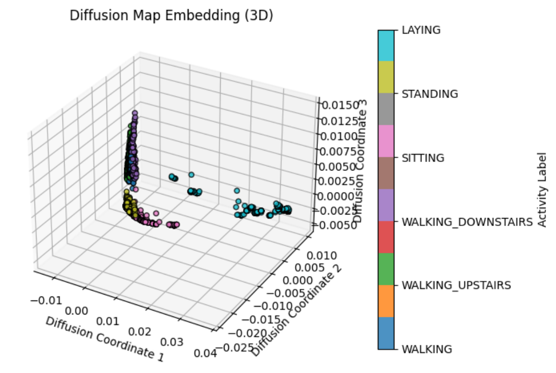
```
================================================
K-MEANS CLUSTERING RESULTS:
ARI: 0.0007, Silhouette Score: 0.9844
================================================
```

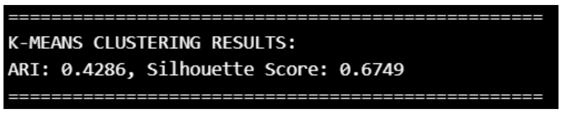
2. Diffusion and clustering for epsilon = median of similarity matrix ^ 2



```
EIGEN VALUES:
[ 1.          0.60758263  0.15815951 ... -0.00179772 -0.00236463
 -0.00280067]

EMBEDDINGS:
[[-0.00630288 -0.00325788 -0.00426811]
 [-0.00630704 -0.00324123 -0.00426521]
 [-0.00635471 -0.00321914 -0.00426325]
 ...
 [-0.00597382 -0.00324625 -0.00436911]
 [-0.00591864 -0.00328645 -0.00436474]
 [-0.00588578 -0.00331449 -0.00436146]]
```

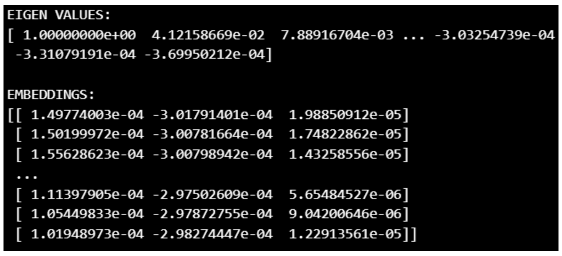




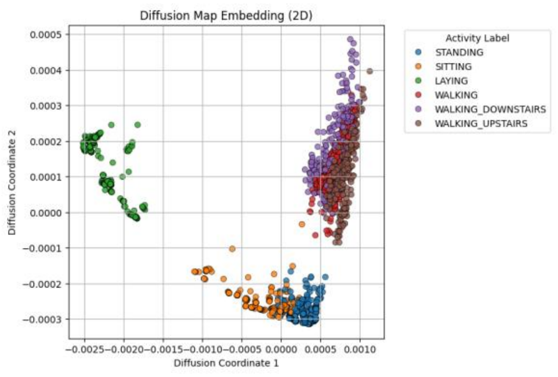


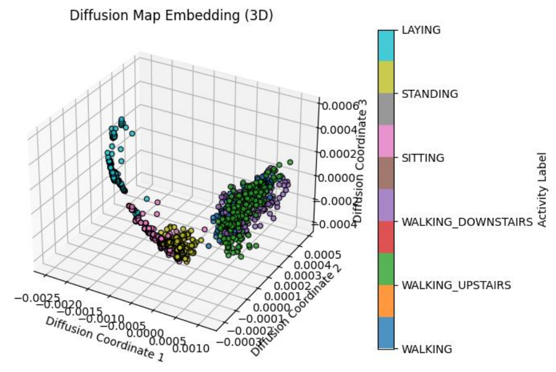
```
=================================================
K-MEANS CLUSTERING RESULTS:
ARI: 0.4286, Silhouette Score: 0.6749
=================================================
```

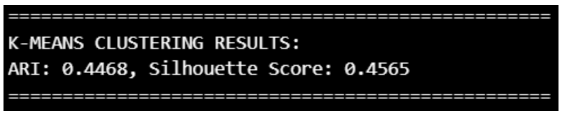
3. Diffusion and clustering for epsilon = median of similarity matrix ^ 3



```
EIGEN VALUES:
[ 1.00000000e+00  4.12158669e-02  7.88916704e-03 ... -3.03254739e-04
 -3.31079191e-04 -3.69950212e-04]

EMBEDDINGS:
[[ 1.49774003e-04 -3.01791401e-04  1.98850912e-05]
 [ 1.50199972e-04 -3.00781664e-04  1.74822862e-05]
 [ 1.55628623e-04 -3.00798942e-04  1.43258556e-05]
 ...
 [ 1.11397905e-04 -2.97502609e-04  5.65484527e-06]
 [ 1.05449833e-04 -2.97872755e-04  9.04200646e-06]
 [ 1.01948973e-04 -2.98274447e-04  1.22913561e-05]]
```







```
=================================================
K-MEANS CLUSTERING RESULTS:
ARI: 0.4468, Silhouette Score: 0.4565
=================================================
```

Clustering Results for different values of epsilon

RESULTS

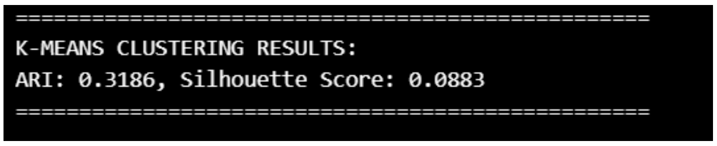
ϵ = (median of pairwise distance matrix)^n

- For n = 1: We get ARI = 0.0007 and Silhouette Score = 0.9844
- For n = 2: We get ARI = 0.4286 and Silhouette Score = 0.6749
- For n = 3: We get ARI = 0.4468 and Silhouette Score = 0.4565

$\therefore$ We get best results when ϵ = (median of pairwise distance matrix)^2

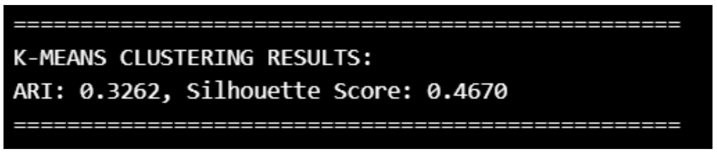
Comparison with other dimensional reduction techniques

1. Raw Feature Space

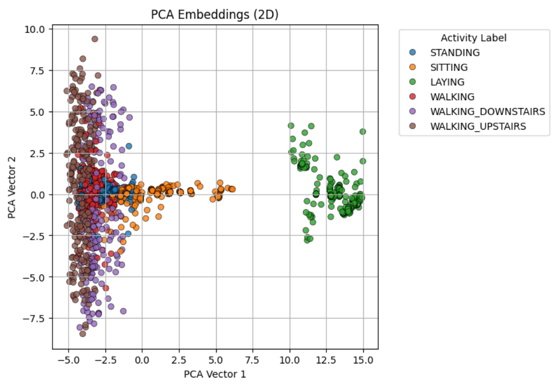


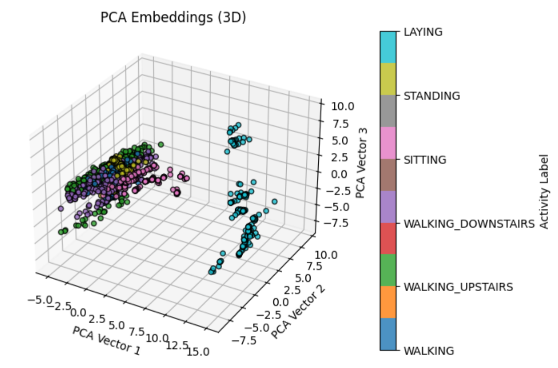
```
=================================================
K-MEANS CLUSTERING RESULTS:
ARI: 0.3186, Silhouette Score: 0.0883
=================================================
```

2. PCA

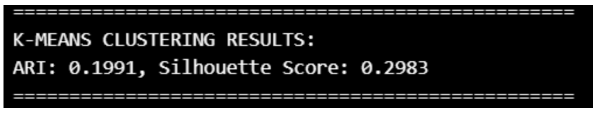


```
=================================================
K-MEANS CLUSTERING RESULTS:
ARI: 0.3262, Silhouette Score: 0.4670
=================================================
```

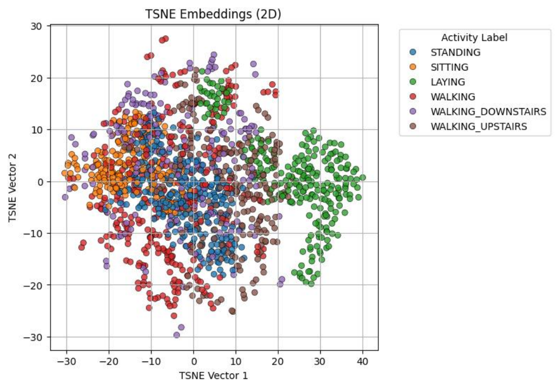


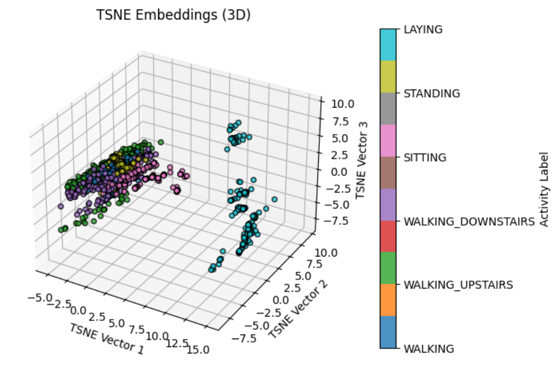


3. t-SNE



```
================================================
K-MEANS CLUSTERING RESULTS:
ARI: 0.1991, Silhouette Score: 0.2983
================================================
```





Why does diffusion Maps work well for time-series clustering?

We observe that we get better clustering results using diffusion maps, with dynamic time warping as distance metric when compared to PCA or TSNE. Diffusion maps work better because time-series data lie on low-dimensional manifolds embedded in high-dimensional spaces which are preserved. Also, it uses local similarity (dynamic time warping) to uncover complex nonlinear data manifold.

PCA projects the points on to a line. This results in loss of non-linearity.

t-SNE is good for preserving the local non-linear structure, however it cannot maintain the global non-linear structure.

Raw features, even though contain all the information, do not form good clusters because of lot of redundant information and noise.

Problem 2: Using Derivative-Free Optimization Methods for optimizing the SVM parameters to fit on MNIST Dataset

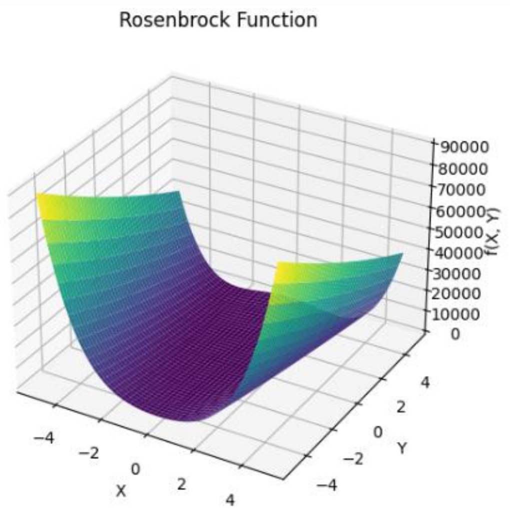
The 3 derivative free optimization techniques mentioned are:

- Nelder-Mead (Simplex Method)
- Simulated Annealing
- CMA-ES (Covariance Matrix Adaptation Evolution Strategy)

Task 1: Benchmarking on Test Functions

1. Rosenbrock function

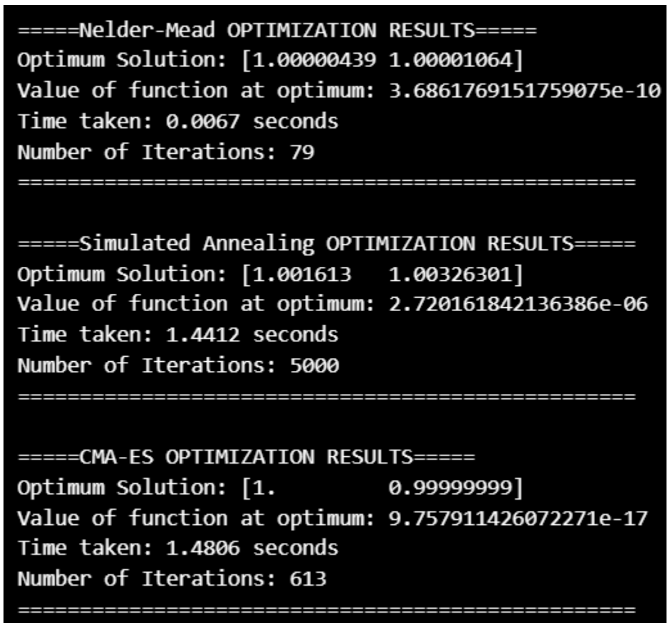
$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$



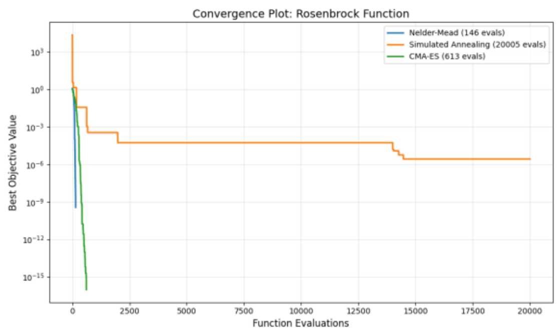
Global Minima: (1,1)

Initial Guess: (0,0)

Bounds: ([-5, 5], [-5, 5])



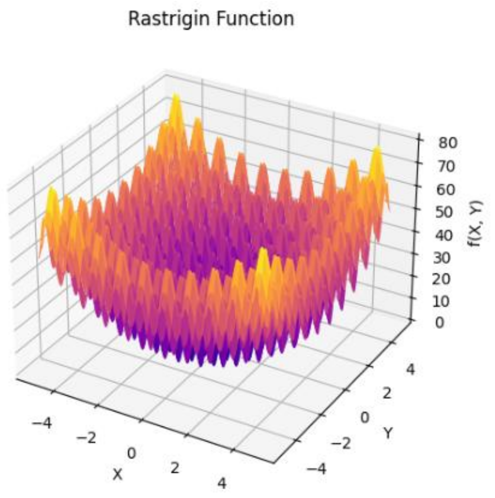
```
=====Nelder-Mead OPTIMIZATION RESULTS=====
Optimum Solution: [1.00000439 1.00001064]
Value of function at optimum: 3.6861769151759075e-10
Time taken: 0.0067 seconds
Number of Iterations: 79
==================================================

=====Simulated Annealing OPTIMIZATION RESULTS=====
Optimum Solution: [1.001613   1.00326301]
Value of function at optimum: 2.720161842136386e-06
Time taken: 1.4412 seconds
Number of Iterations: 5000
==================================================

=====CMA-ES OPTIMIZATION RESULTS=====
Optimum Solution: [1.         0.99999999]
Value of function at optimum: 9.757911426072271e-17
Time taken: 1.4806 seconds
Number of Iterations: 613
==================================================
```



We can observe that all the methods are able to converge to global minima.

2. Rastrigin function

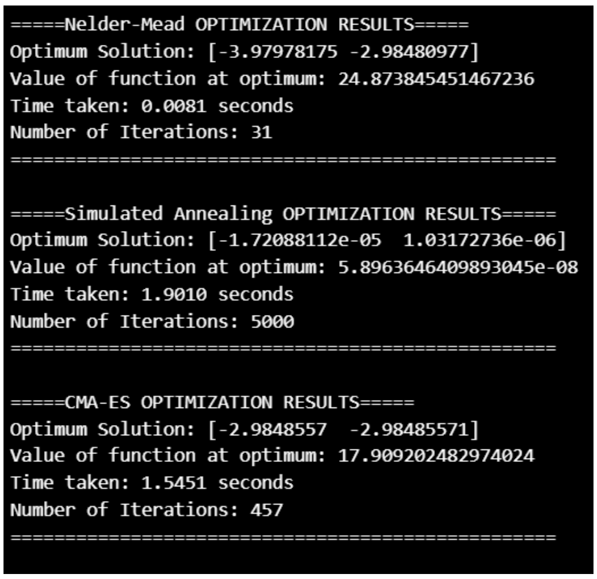
$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right]$$



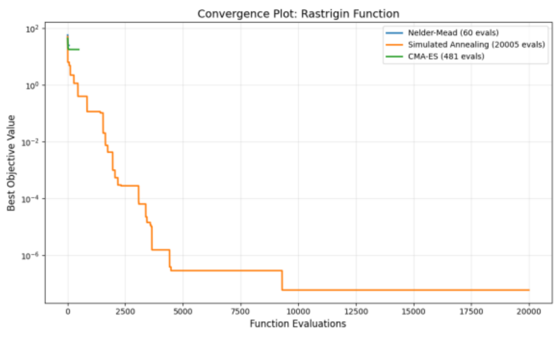
Global Minima: (0,0)

Initial Guess: Sample from uniform distribution with a=-5 and b=5

Bounds: ([-5, 5], [-5, 5])



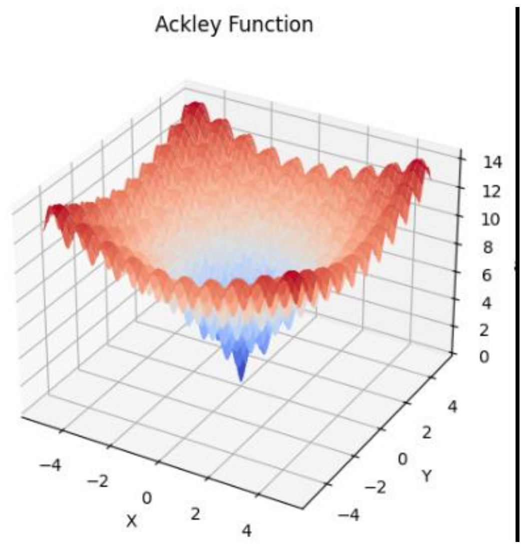
```
=====Nelder-Mead OPTIMIZATION RESULTS=====
Optimum Solution: [-3.97978175 -2.98480977]
Value of function at optimum: 24.873845451467236
Time taken: 0.0081 seconds
Number of Iterations: 31
==================================================

=====Simulated Annealing OPTIMIZATION RESULTS=====
Optimum Solution: [-1.72088112e-05  1.03172736e-06]
Value of function at optimum: 5.8963646409893045e-08
Time taken: 1.9010 seconds
Number of Iterations: 5000
==================================================

=====CMA-ES OPTIMIZATION RESULTS=====
Optimum Solution: [-2.9848557  -2.98485571]
Value of function at optimum: 17.909202482974024
Time taken: 1.5451 seconds
Number of Iterations: 457
==================================================
```



We can observe that only Simulated Annealing reaches the optimum solution, however it takes lot of iterations and time.

3. Ackley function

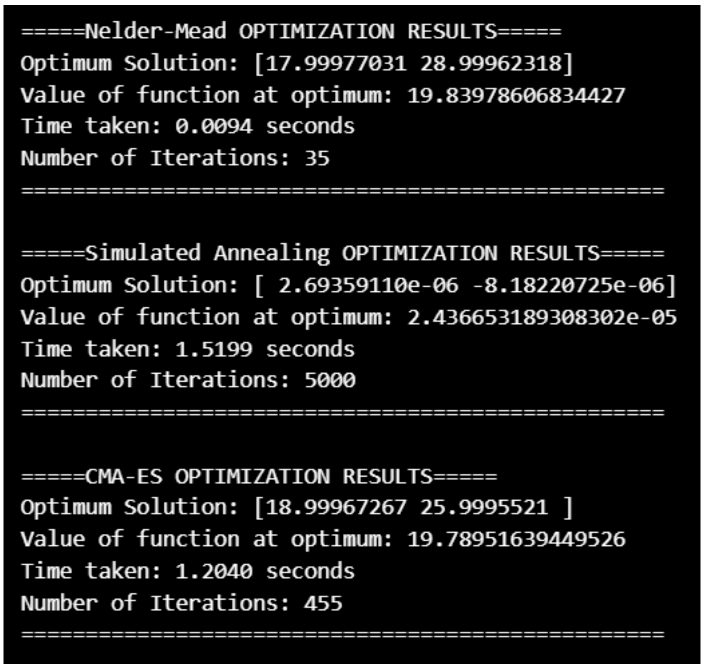
$$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$$



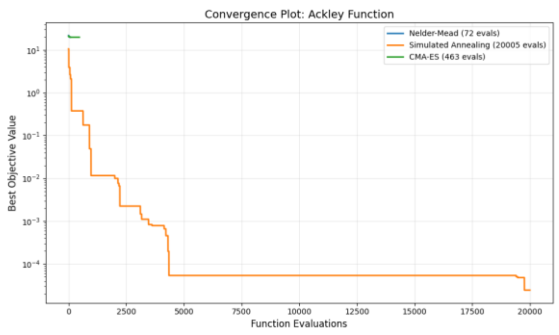
Global Minima: (0,0)

Initial Guess: Sample from uniform distribution with a=-5 and b=5

Bounds: ([-5, 5], [-5, 5])



```
=====Nelder-Mead OPTIMIZATION RESULTS=====
Optimum Solution: [17.99977031 28.99962318]
Value of function at optimum: 19.83978606834427
Time taken: 0.0094 seconds
Number of Iterations: 35
==================================================

=====Simulated Annealing OPTIMIZATION RESULTS=====
Optimum Solution: [ 2.69359110e-06 -8.18220725e-06]
Value of function at optimum: 2.436653189308302e-05
Time taken: 1.5199 seconds
Number of Iterations: 5000
==================================================

=====CMA-ES OPTIMIZATION RESULTS=====
Optimum Solution: [18.99967267 25.9995521 ]
Value of function at optimum: 19.78951639449526
Time taken: 1.2040 seconds
Number of Iterations: 455
==================================================
```



Here, again we observe that only Simulated Annealing reaches the optimum.

Task 2: Hyperparameter tuning for SVM

1. NELDER MEADS RESULTS

Best Kernel: poly

Best C: 1.0000

Best Gamma: 1.0000

Validation Error: 0.1800

Test Accuracy: 0.8660

Time: 1.73s

Function Evaluations: 49

2. SIMULATED ANNEALING RESULTS

Best Kernel: rbf

Best C: 2.3698

Best Gamma: 0.0103

Validation Error: 0.1000

Test Accuracy: 0.9020

Time: 1492.98s

Function Evaluations: 30005

3. CMA-ES RESULTS

Best Kernel: linear

Best C: 0.0277

Best Gamma: 3.0666

Validation Error: 0.1067

Test Accuracy: 0.8880

Time: 17.03s

Function Evaluations: 260

TEST ACCURACY COMPARISON

Nelder-Mead: 0.8660

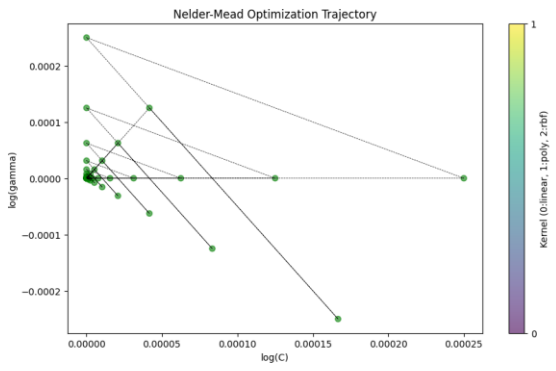
Simulated Annealing: 0.9020

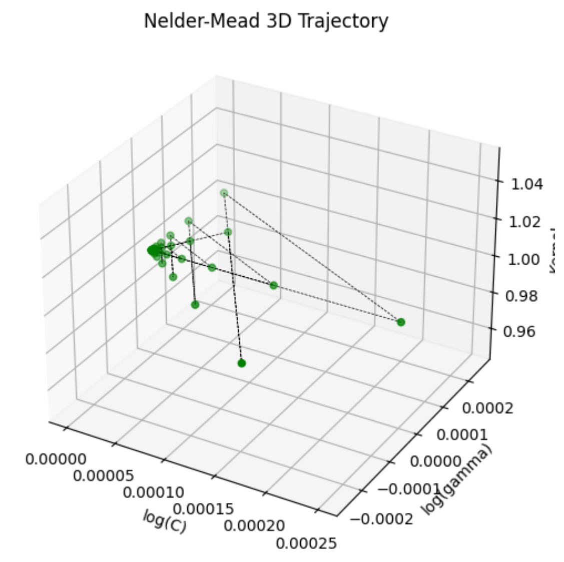
CMA-ES: 0.8880

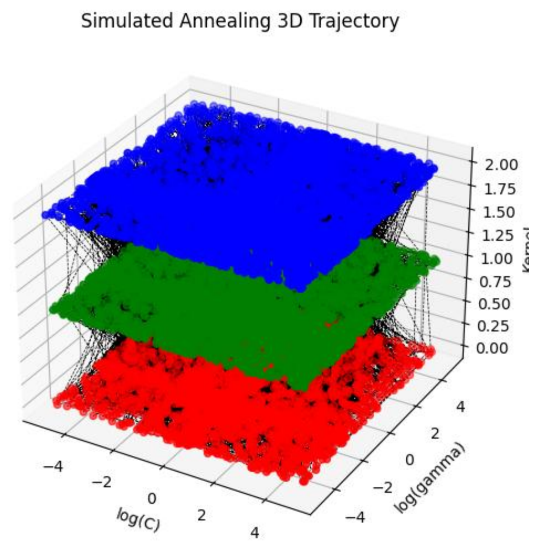
NOTE: Simulated Annealing and CMA are more stable to initial conditions where as Nelder meads method is very sensitive to initial conditions and gets stuck in local minima's.

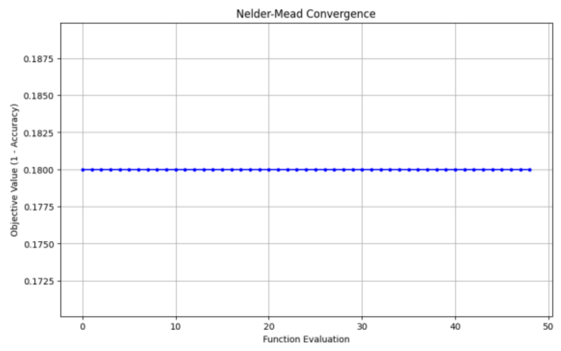
Task 3: Visualizations

1. Optimization Trajectory and Convergence for Nelder's Method

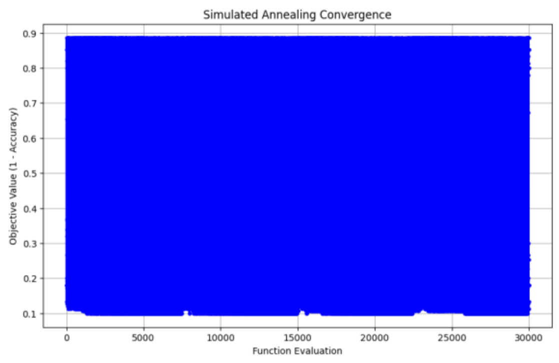




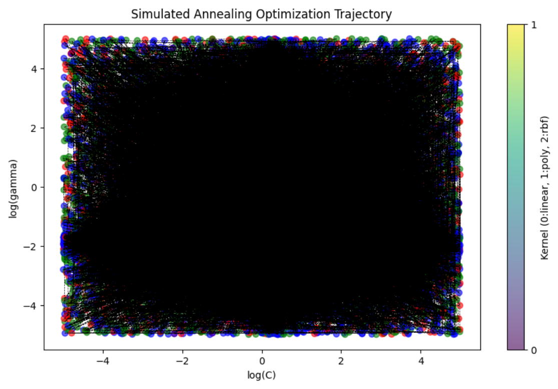




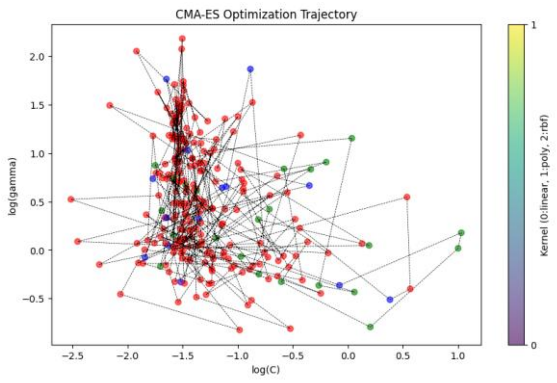
Here, blue is RBF kernel, green is polynomial kernel and red is linear kernel.

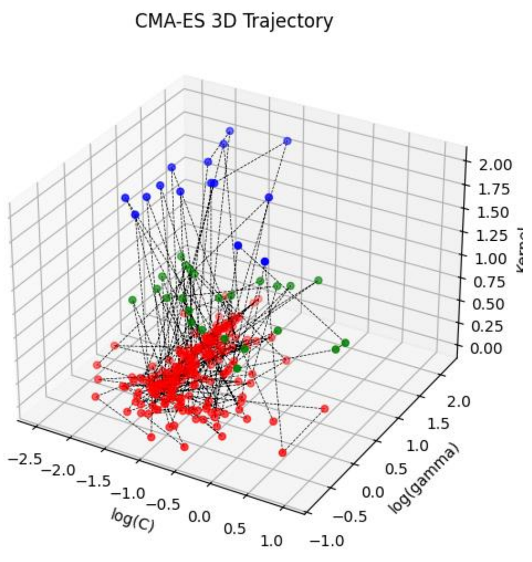


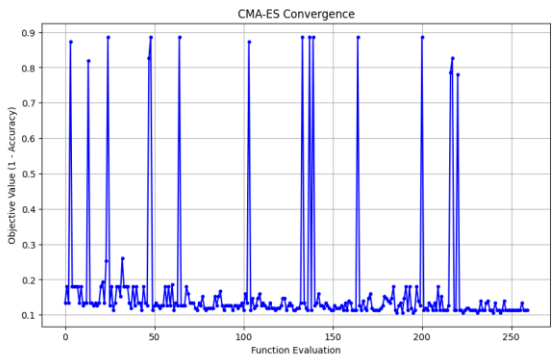
2. Optimization Trajectory and Convergence for Simulated Annealing



3. Optimization Trajectory and Convergence for CMA-ES

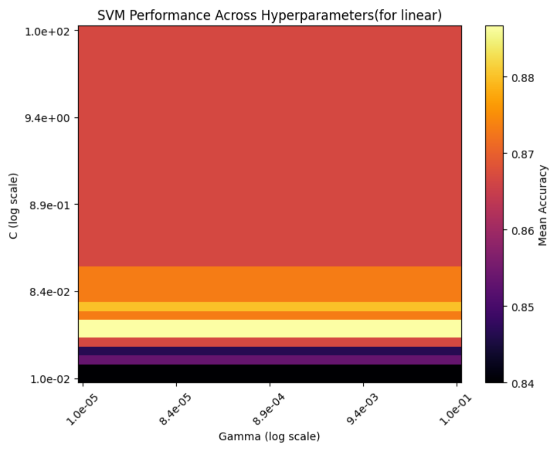




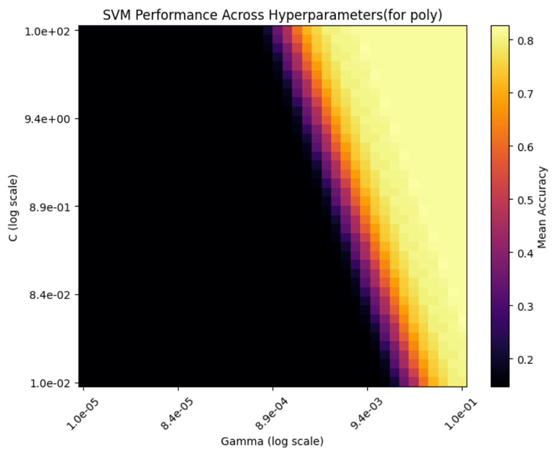


Visualizing the hyperparameter planes

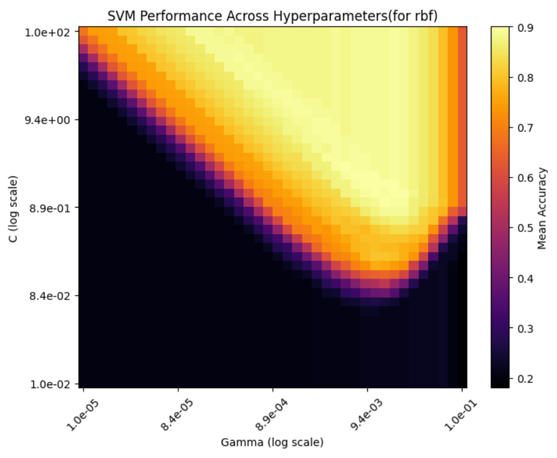
1. Linear Kernel



2. Polynomial Kernel



3. RBF Kernel



Final Observations and Conclusion

1. Nelder-Mead:

Pros: No gradient required, works well for low-dimensional problems.

Cons: Slow convergence in high dimensions, sensitive to initial simplex.

2. Simulated Annealing:

Pros: Escapes local minima, handles non-convex functions.

Cons: Requires careful tuning of temperature schedule, computationally intensive.

3. CMA-ES:

Pros: Efficient for global optimization, self-adapting covariance matrix.

Cons: Higher computational cost per iteration, complex parameter settings.

Performance Analysis

On benchmark functions, CMA-ES often converges faster to the global minimum, especially for multimodal functions like Rastrigin. Nelder-Mead performs well on smooth functions like Rosenbrock but gets trapped in local minima for non-convex landscapes. Simulated Annealing balances exploration and exploitation but requires more evaluations.

Hyperparameter Tuning:

CMA-ES achieved the high-test accuracy on MNIST with **fewer** evaluations, demonstrating efficiency in navigating the hyperparameter space. Simulated Annealing provided robust results across runs, while Nelder-Mead was less consistent due to sensitivity to initial guesses.

Conclusion:

CMA-ES is recommended for complex optimization tasks, whereas Nelder-Mead is suitable for simpler, low-dimensional problems. Simulated Annealing offers a middle ground with better global search capabilities at the cost of increased computational resources.