Manifold Learning (ML) for large data sets

Group D

Xinyu Liang

Mark Sandal

Contents

- Introduction
- AE Implementation
- AE performance test on noisy swiss-roll
- AE performance test on word2vec
- Comparison with sklearn and drtoolbox

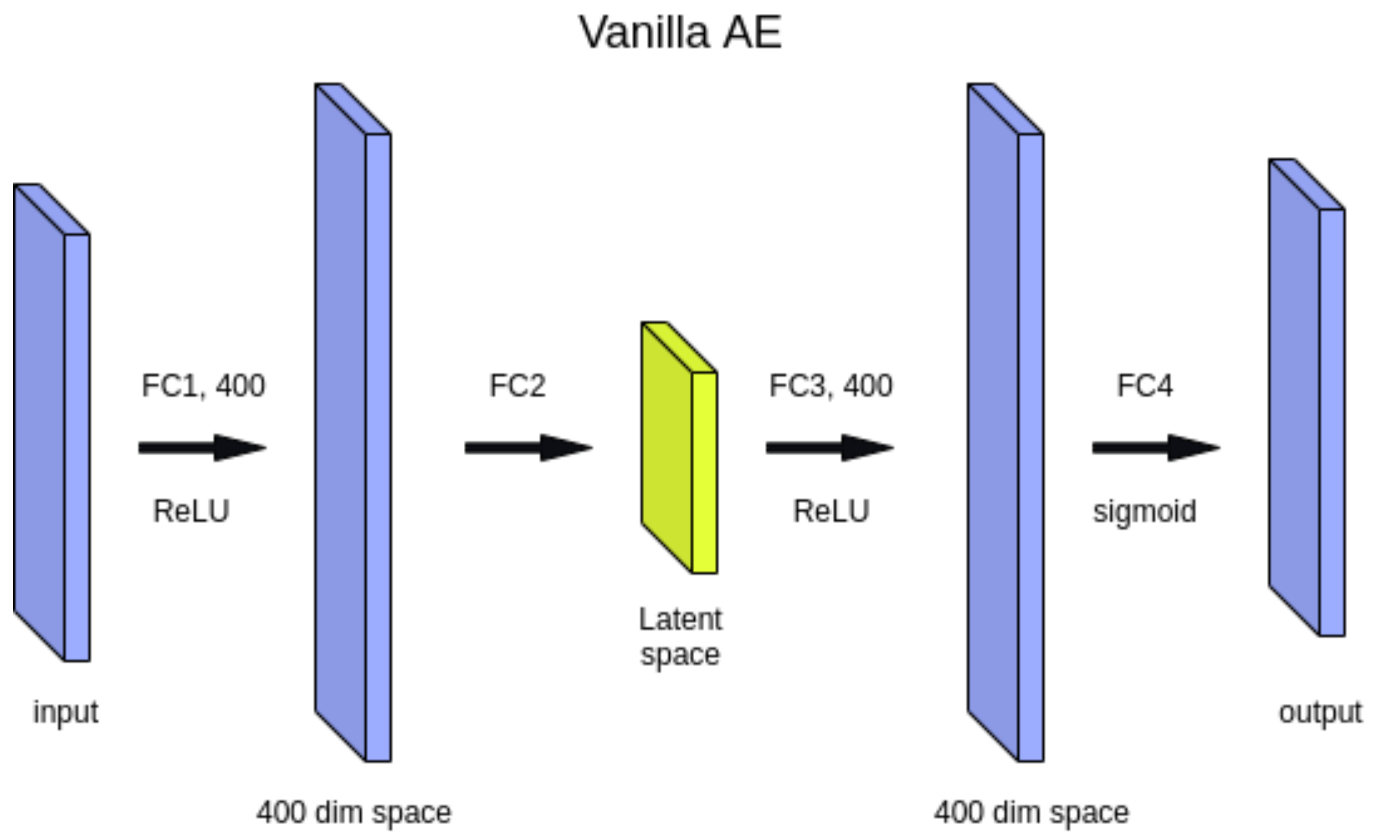
Introduction

- Large dimensionality = problems
- Different methods of dimensionality reduction
- Information loss

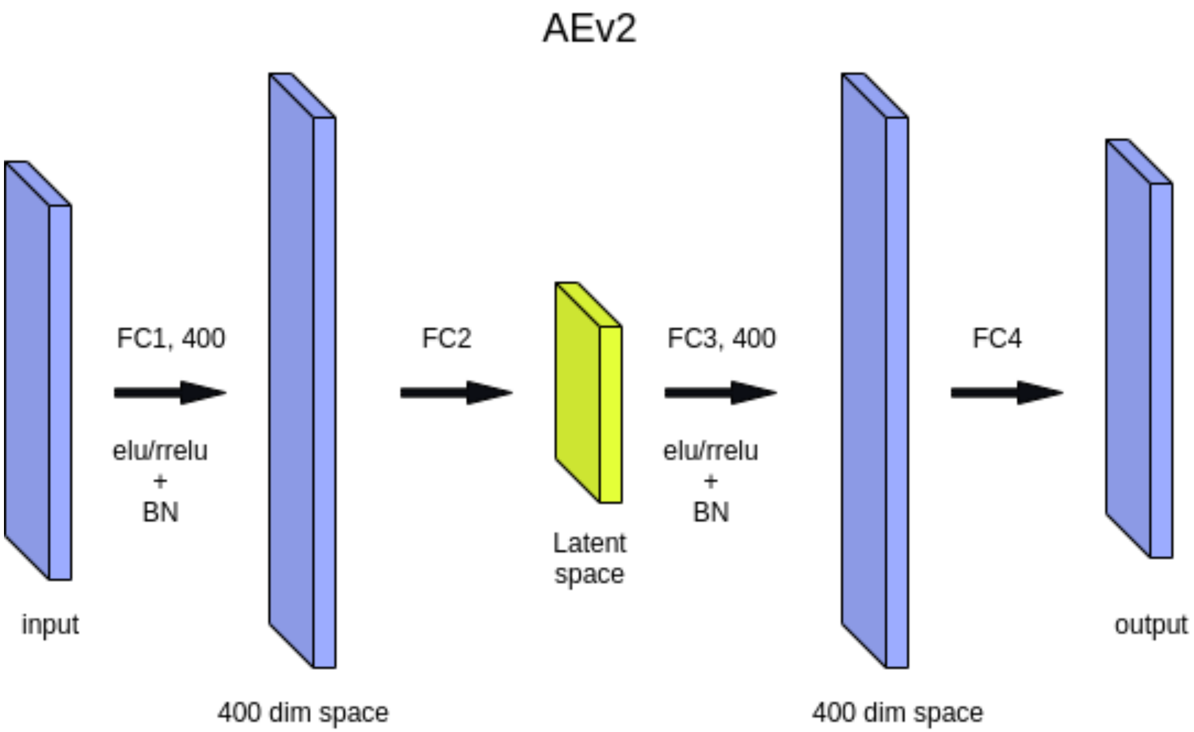
AutoEncoder (AE)

- AE is a type of NN that has the goal to learn a representation of the given data (which would, typically, be in lower dimension than the original) in an unsupervised fashion. AE can be split into three parts - encoder, latent space and decoder.

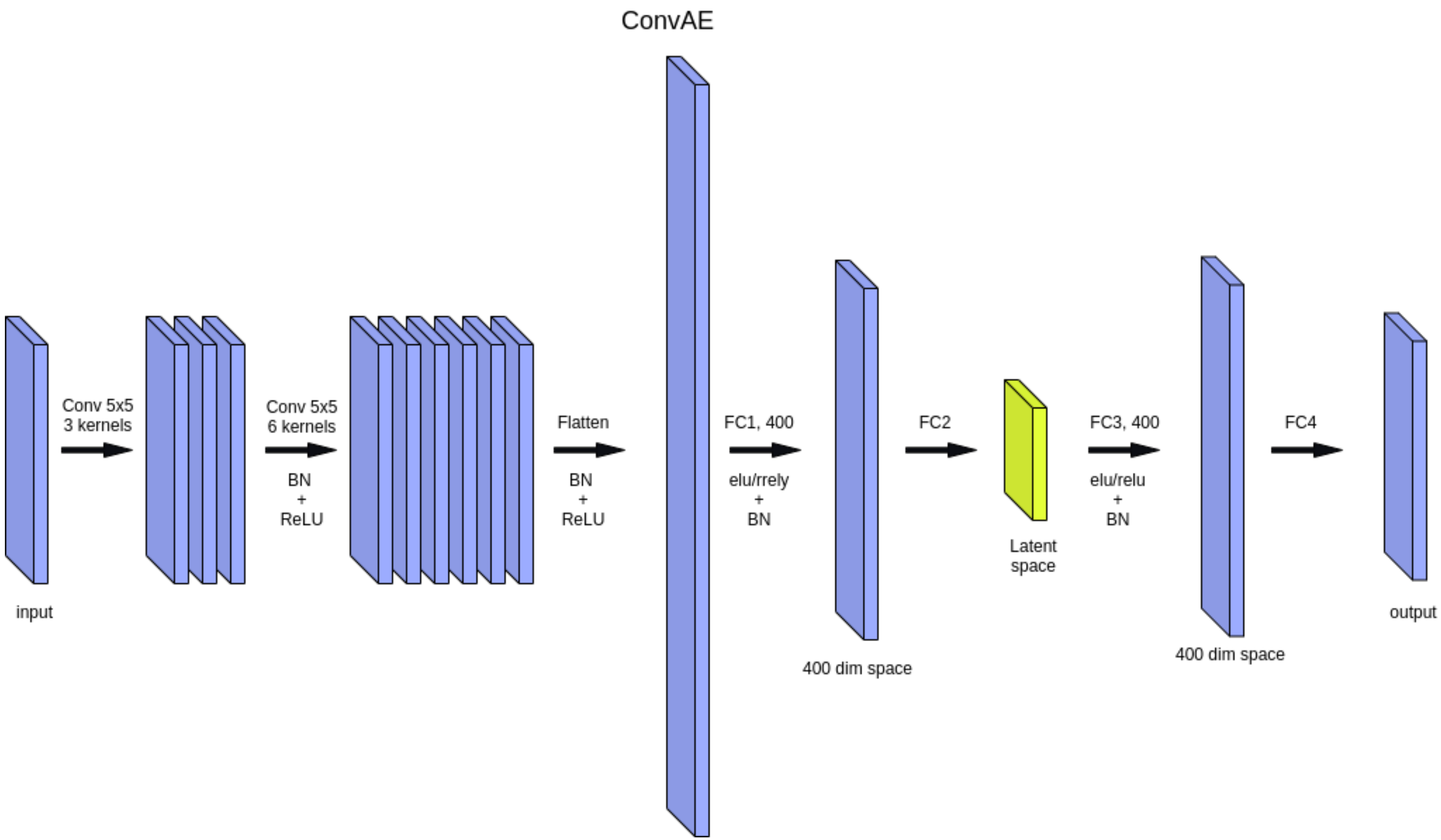
Implementation 1



Implementation 2



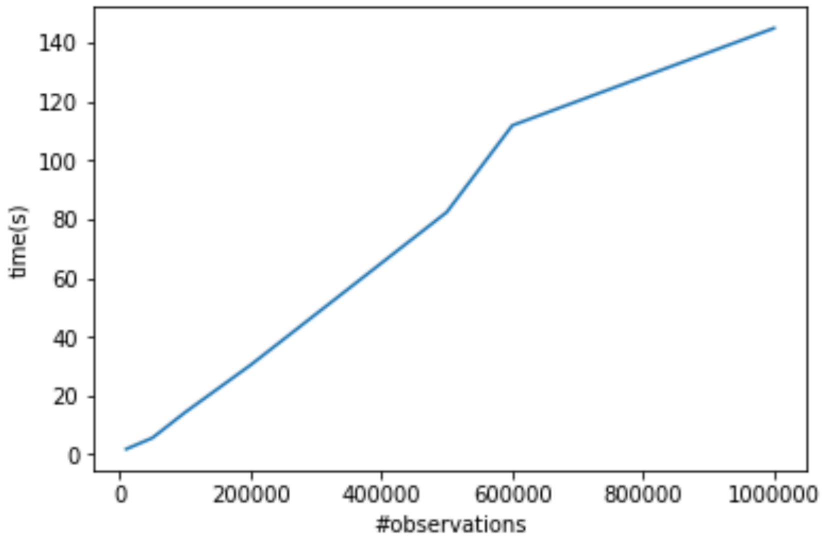
Implementation 3

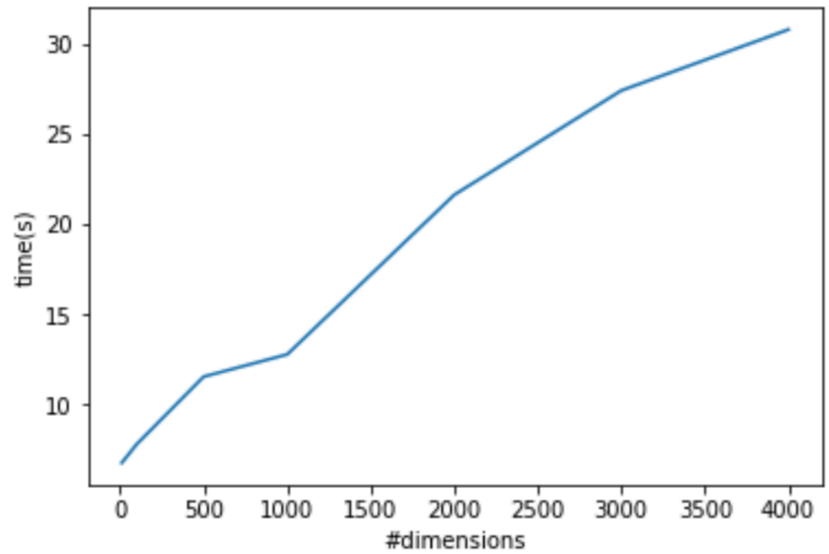


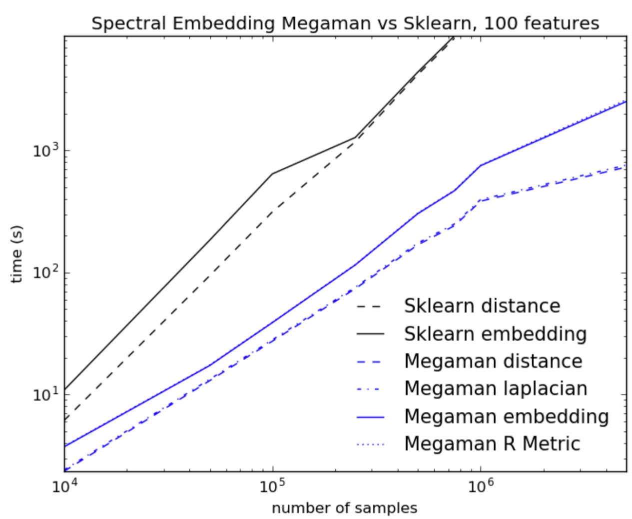
Two datasets to test

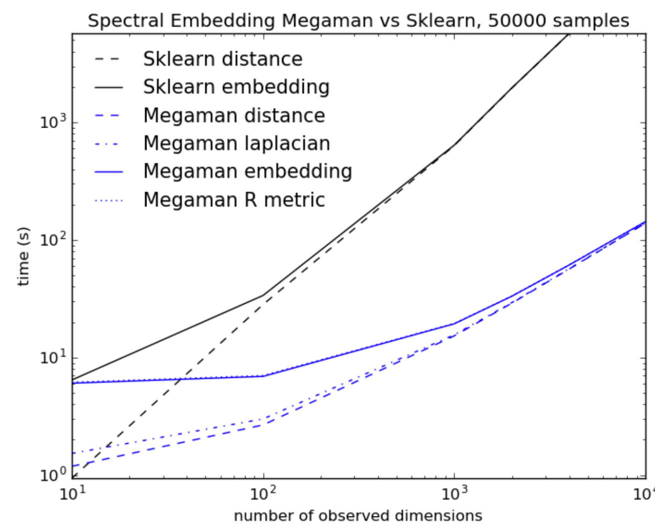
- Noisy Swiss-Roll
- Word2vec

Performance

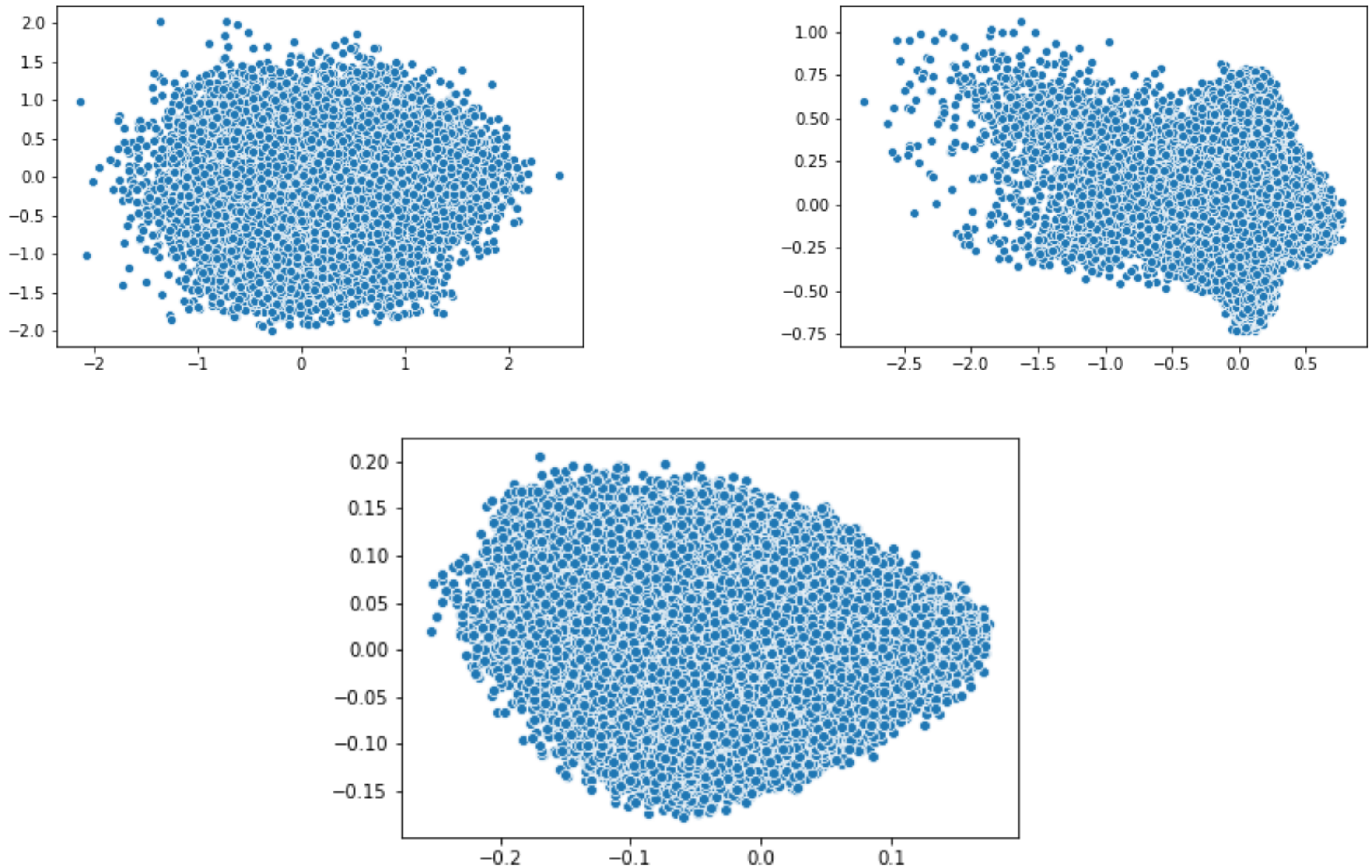








Word2vec performance



Comparisons

- Isometric feature mapping
- Locally Linear Embedding (LLE)
- Sklearn package performance
- Matlab drtoolbox performance

Isomap

- Nearest Neighbor Search

  connect all points within a fixed radius (choose yourself) or like KNN

- Shortest-path Graph Search

  Estimates the geodesic distance between all pairs of points

- Partial Eigenvalue Decomposition

  The embedding is encoded in the eigenvectors corresponding to the largest d eigenvalues of the N × N isomap kernel

$$O[D\log(k)N\log(N)] + O[N^2(k+\log(N))] + O[dN^2]$$

LLE

- Nearest Neighbor Search

    Same as Isomap

- Weight Matrix Construction

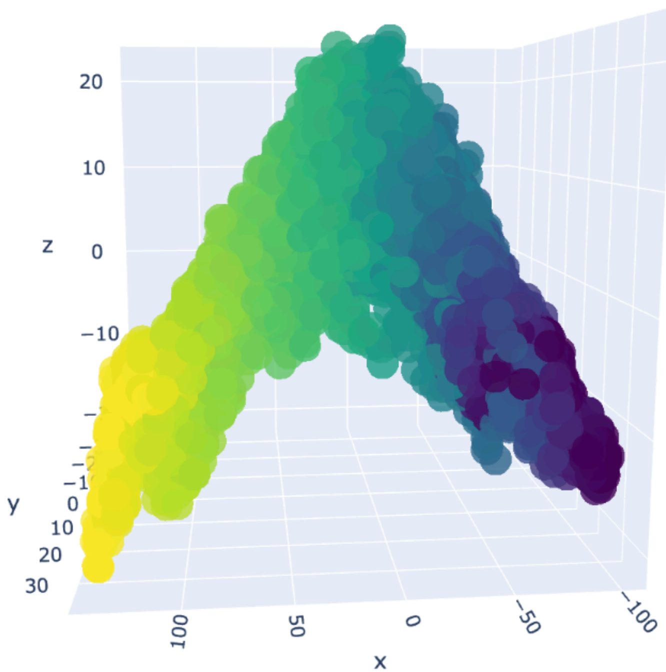
    Compute the weights W_ij best reconstruct each data point from its neighbors, minimizing the cost $E(W) = \sum_i |\vec{X}_i - \sum_j W_{ij}\vec{X}_j|^2$

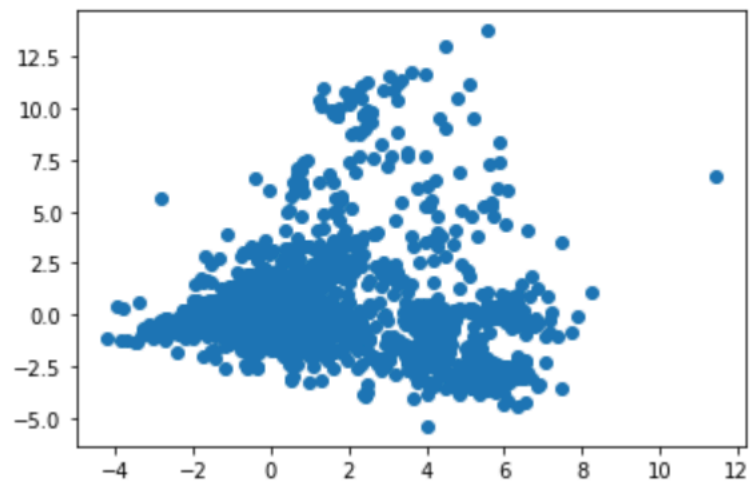
- Partial Eigenvalue Decomposition

    Reconstruct Y_i best reconstructed by the weights W_ij, minimizing $\Phi(Y) = \sum_i |\vec{Y}_i - \sum_j W_{ij}\vec{Y}_j|^2$ by its bottom nonzero eigenvectors

$$O[D\log(k)N\log(N)] + O[\boldsymbol{D}Nk^3] + O[dN^2]$$

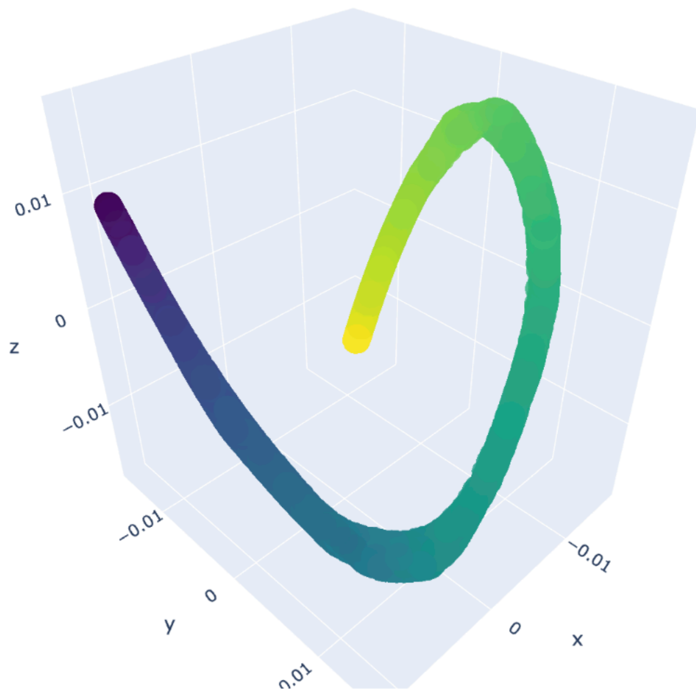
Sklearn isomap

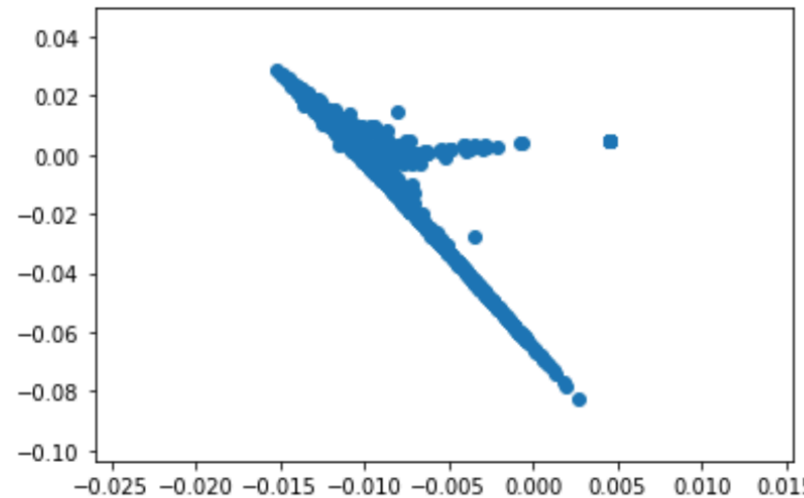




N\d	5	10	50	100	500	1000	5000
1000				0.4645			
10000				59.9786			
50000	too long	too long	too long	too long	too long	too long	too long
100000				too long			
500000				too long			
100000				too long			

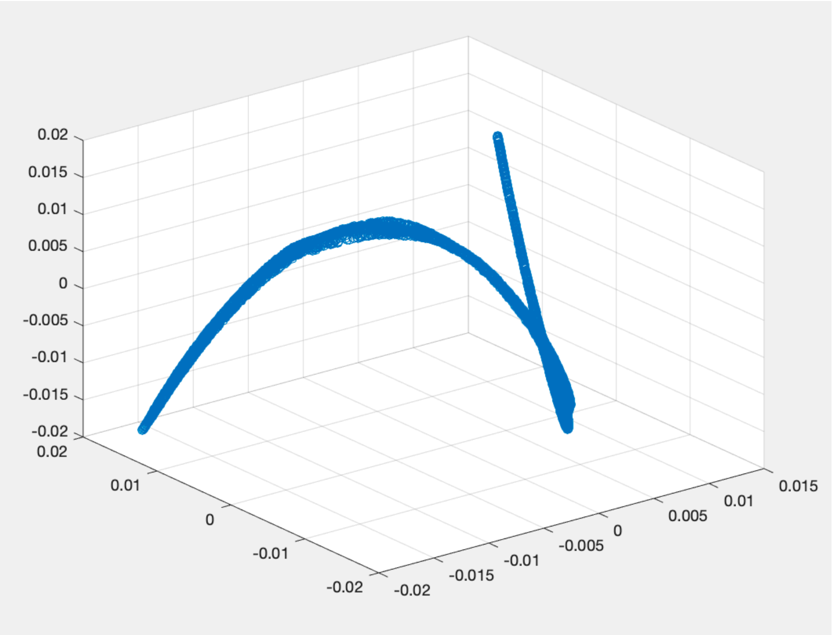
Sklearn LLE

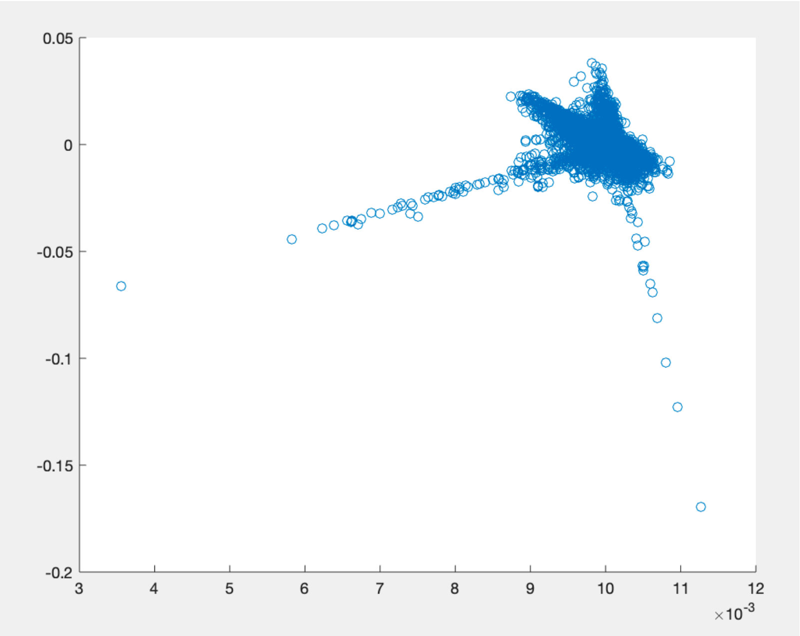




N\d	5	10	50	100	500	1000	5000
1000				0.1605			
10000				4.1318			
50000	14.2359	23.1115	45.2676	74.7430	550.8176	1872.5697	too long
100000				502.2671			
500000				too long			
100000				too long			

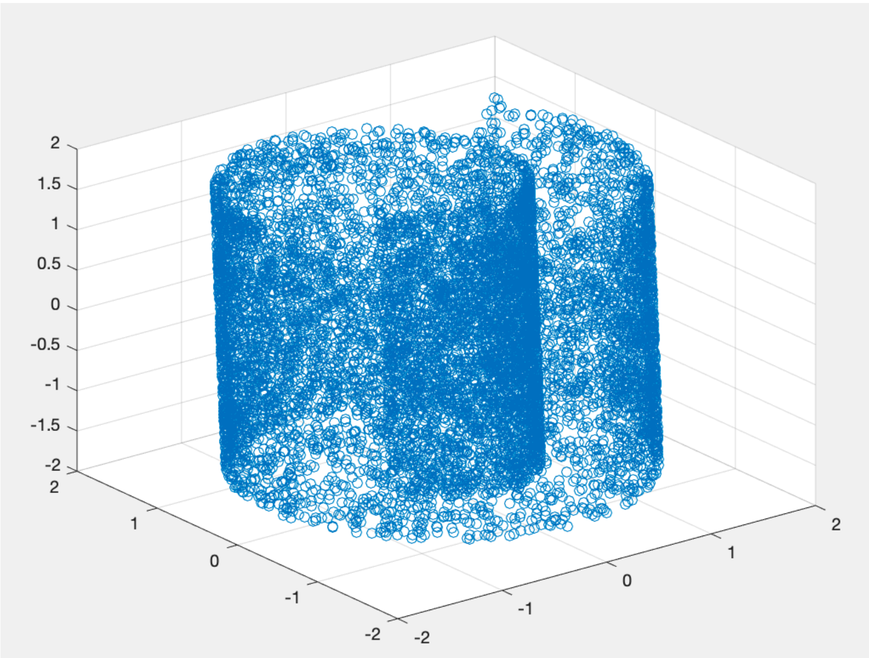
Matlab drtoolbox LLE

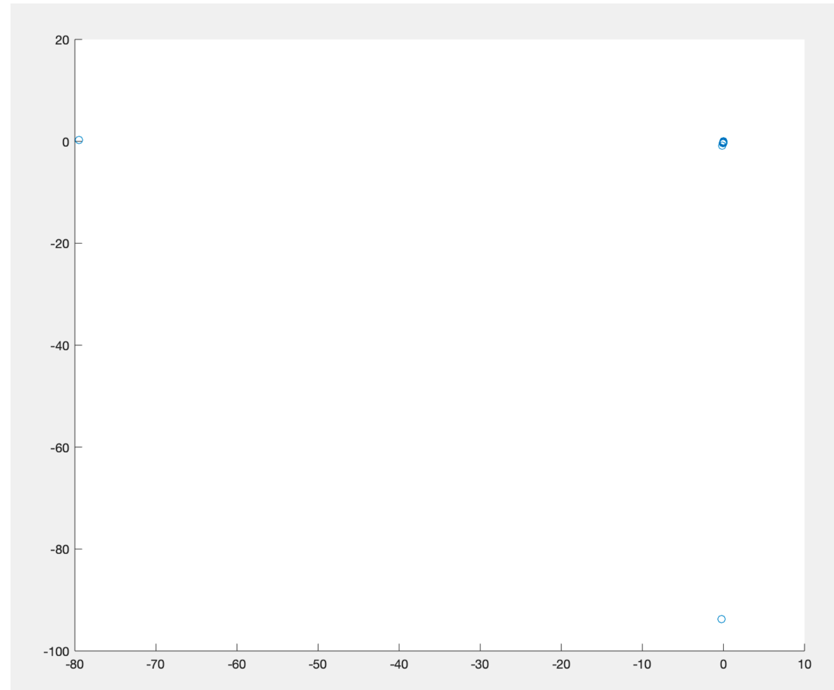




N\d	10	50	100	500	1000	5000
10000			42.0565			
50000	931.5965	too long	too long	too long	too long	too long
100000			too long			
500000			too long			
100000			too long			

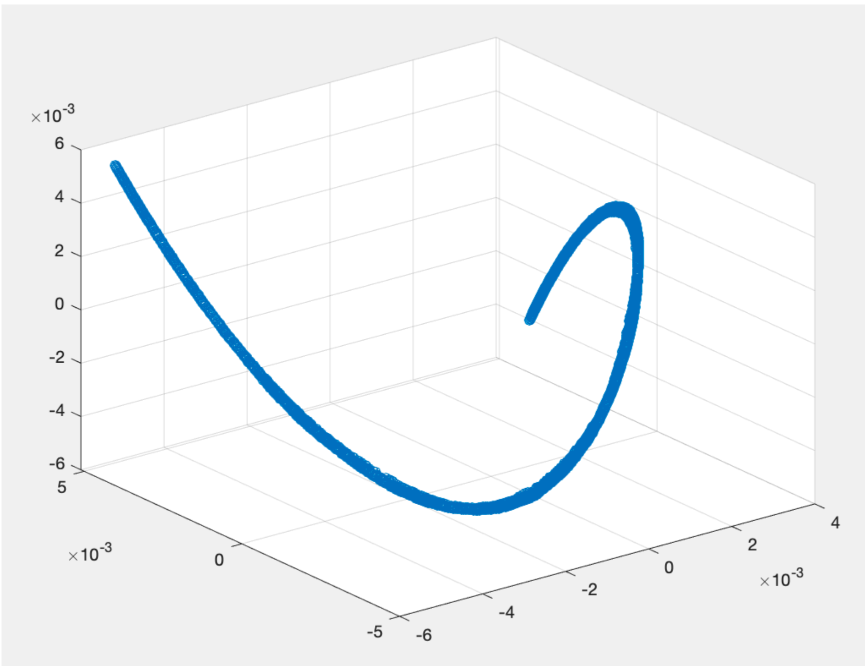
Matlab drtoolbox Diffusion Maps

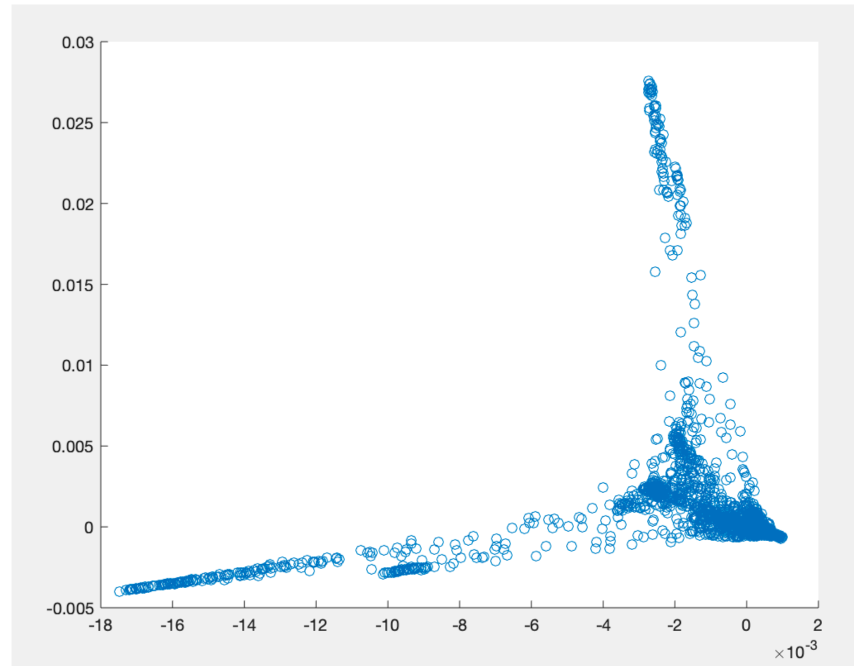




N\d	10	50	100	500	1000	5000
10000			292.3643			
50000	memory exceed	memory exceed	memory exceed	memory exceed	memory exceed	memory exceed
100000			memory exceed			
500000			memory exceed			
100000			memory exceed			

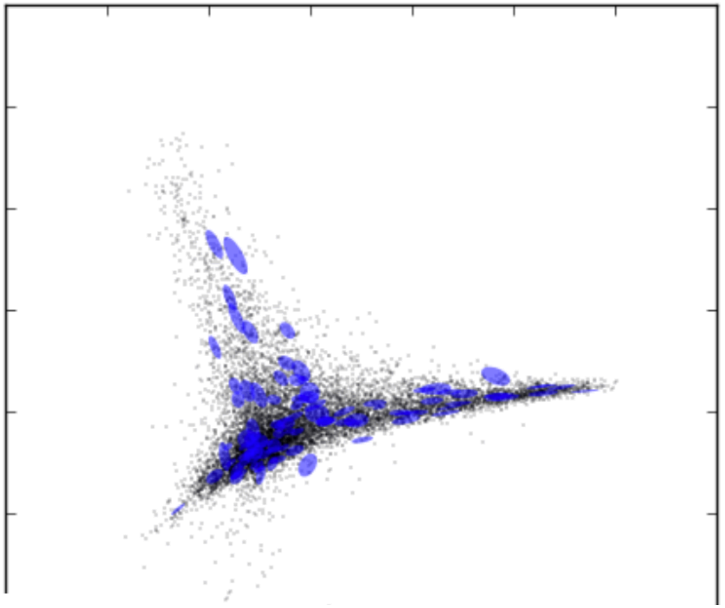
Matlab drtoolbox Laplacian Eigenmaps

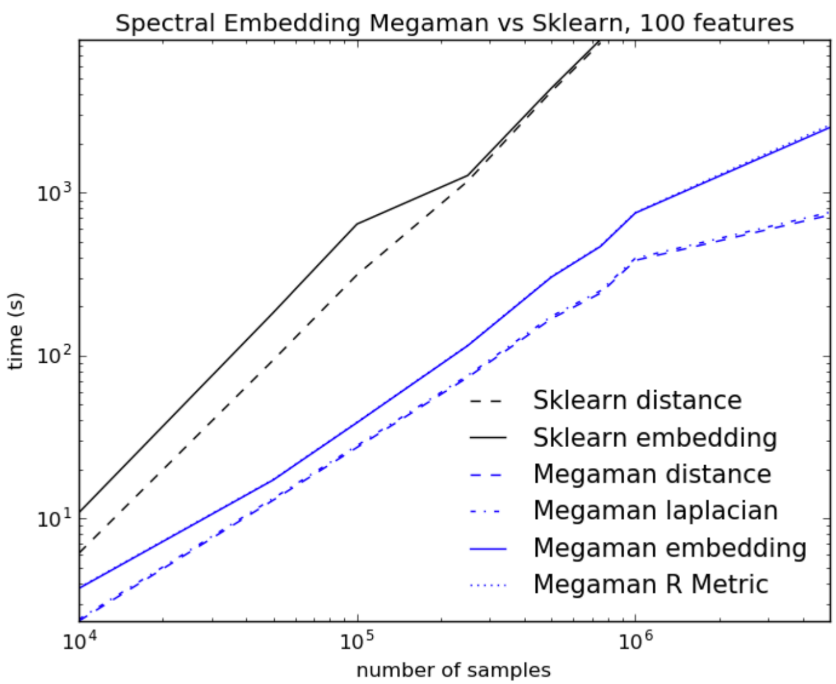




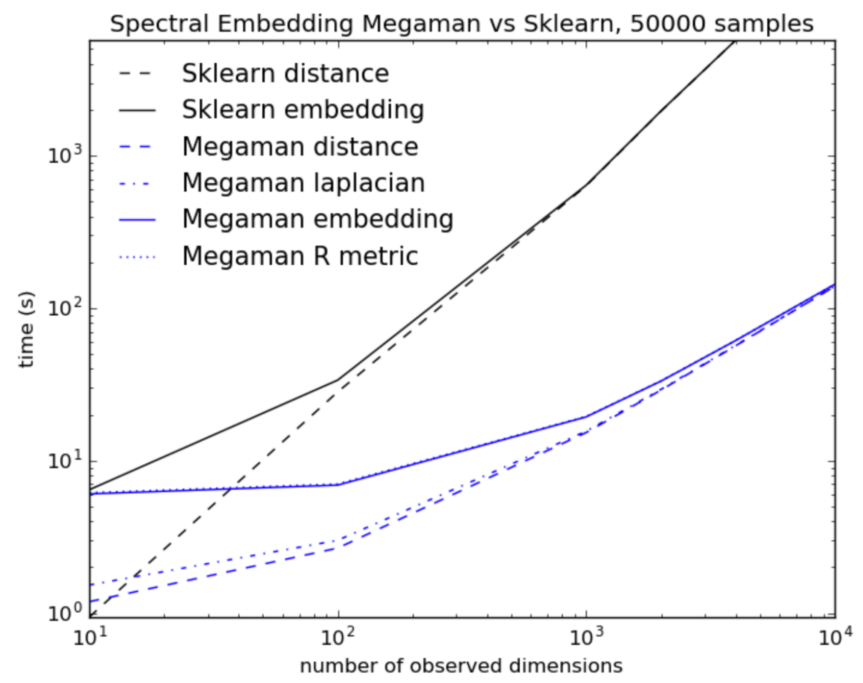
N\d	10	50	100	500	1000	5000
10000			10.8780			
50000	389.19060	417.0323	432.8003	not tested	not tested	not tested
100000			too long			
500000			too long			
100000			too long			

Megaman





Spectral Embedding Megaman vs Sklearn, 100 features
Sklearn distance
Sklearn embedding
Megaman distance
Megaman laplacian
Megaman embedding
Megaman R Metric
time (s)
number of samples



Spectral Embedding Megaman vs Sklearn, 50000 samples
Sklearn distance
Sklearn embedding
Megaman distance
Megaman laplacian
Megaman embedding
Megaman R metric
time (s)
number of observed dimensions

Conclusion

- AEs work well when the compression is reasonable (most of information is preserved)
- AEs scale better - they are fairly quick to train and due to the nature of NNs are able to use of transfer learning
- When we want to compress data extremely tightly - from 300 to 2, for example - neighborhood graph methods are to be preferred.
- Complex networks require more data to train - due to higher number of weights.
- No free lunch - there is no one good NN that will work for every dataset. Improvise. Adapt. Overcome.
- Plotly is awesome, use it instead of matplotlib when you can.

Reference

- megaman: Manifold Learning with Millions of points, James McQueen et al. Mar 2016
- drtoolbox, https://lvdmaaten.github.io/drtoolbox/
- Manifold-based tools: ISOMAP algorithm, Matteo Alberti, Nov 2017 https://www.deeplearningitalia.com/manifold-based-tools-isomap-algorithm/#pll_switcher
- Manifold learning, scikit-learn https://scikit-learn.org/stable/modules/manifold.html#isomap
- An Introduction to Locally Linear Embedding, Lawrence K. Saul et al, https://cs.nyu.edu/~roweis/lle/papers/lleintroa4.pdf

Thank You!

Q&A