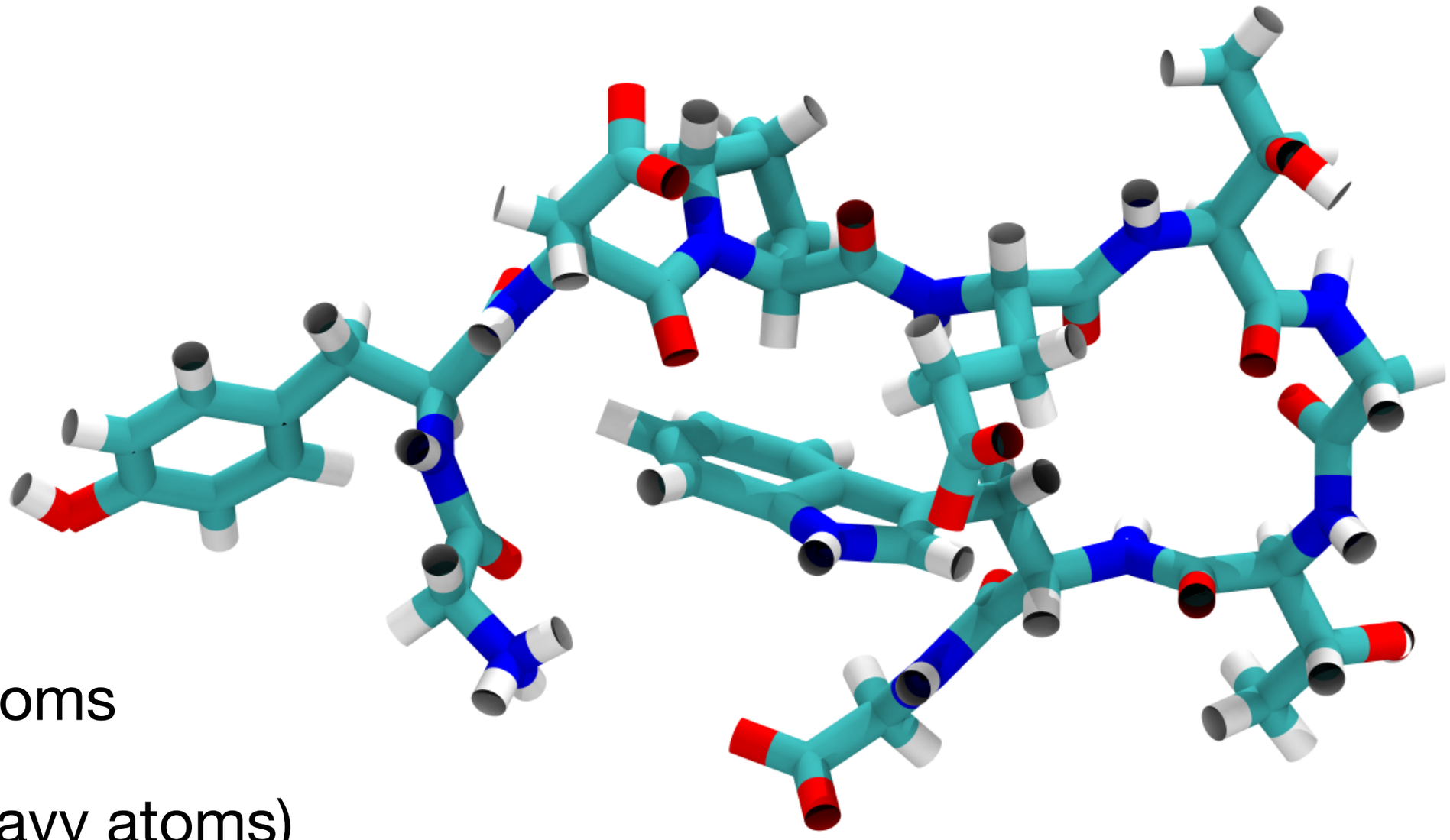
Clustering and Dimensionality reduction

an informal introduction to unsupervised ML

HLML

Eugene Klyshko
Sep 21, 2018

clustering of protein structures



<u>Protein X:</u>

- 138 atoms
- (83 heavy atoms)
- 10 residues
- 785,612 structures (frames)

Machine learning

<u>Supervised</u>

- Classification
- Regression

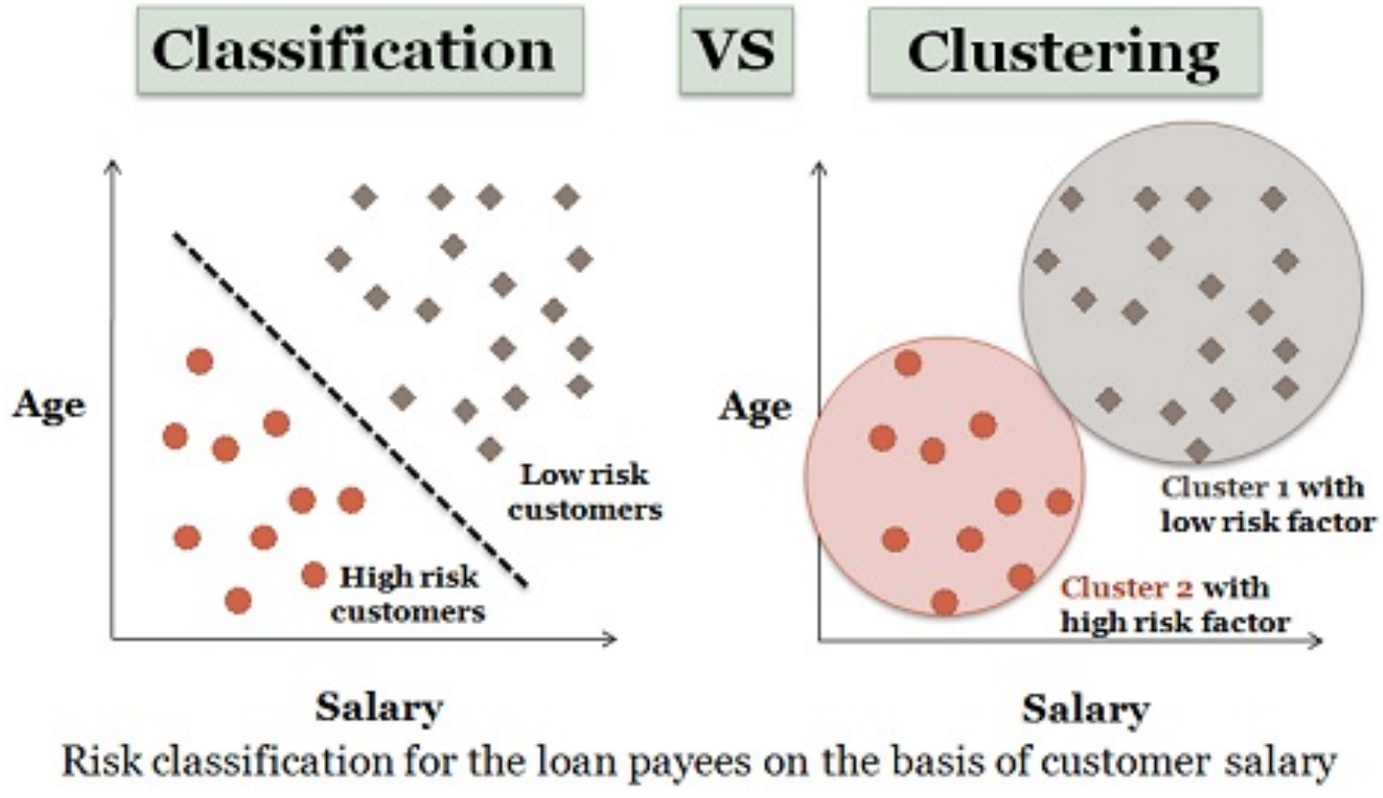
<u>Unsupervised</u>

- Clustering

X - vector of features – **Y** corresponding label

1. given data points: **X** -> **Y**
2. fit and train model
3. determine **Y** for new **X**

1. given data: **X** and <u>no</u> **Y**
2. determine **Y**



Risk classification for the loan payees on the basis of customer salary

<u>Clustering</u> is a technique that performs grouping of data points, so that data is similar within the same cluster, and dissimilar from distinct clusters

Outline

1.Clustering Algorithms:

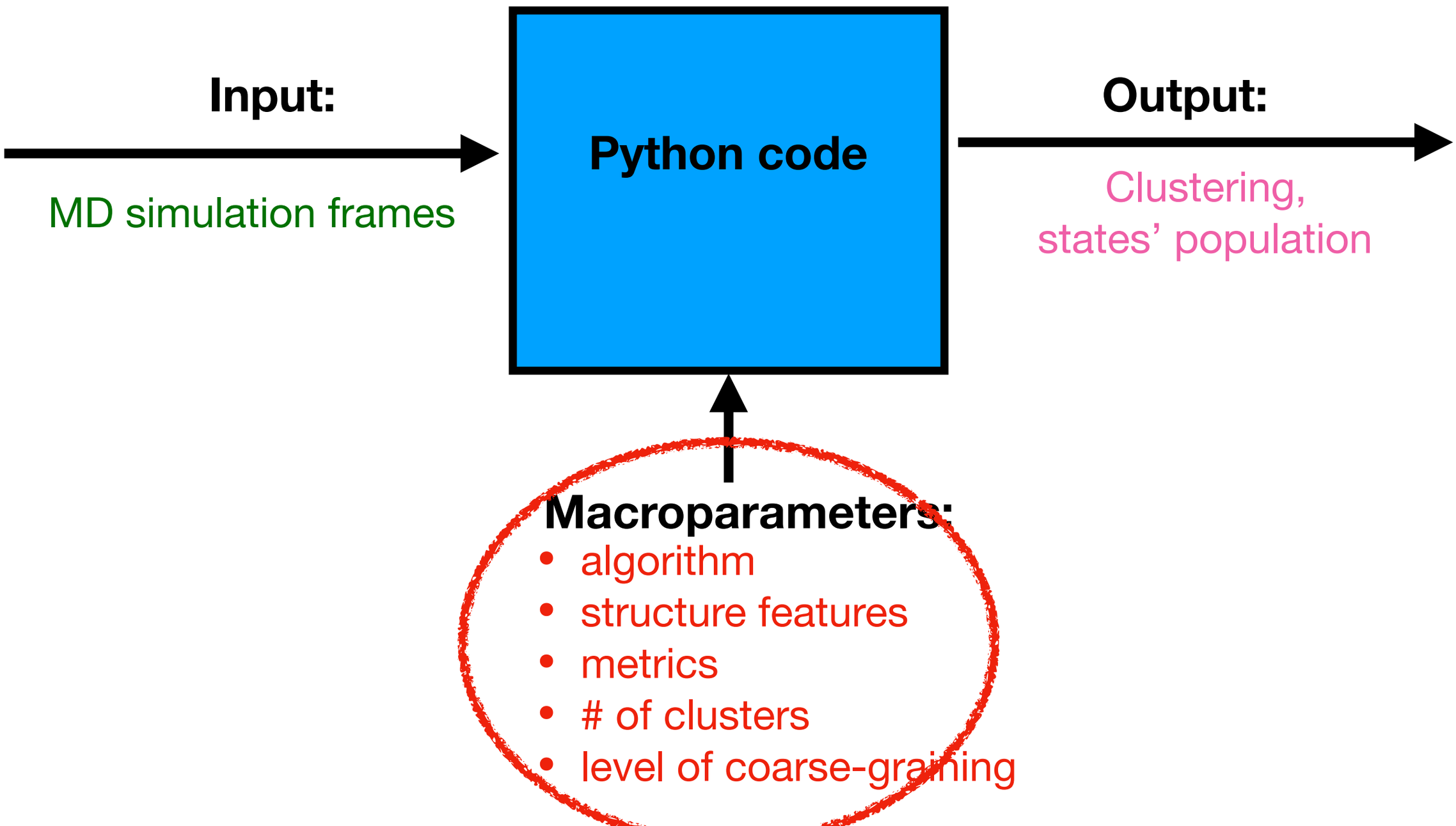
- K-Means
- Hierarchical Clustering

2. Dimensionality reduction tools:

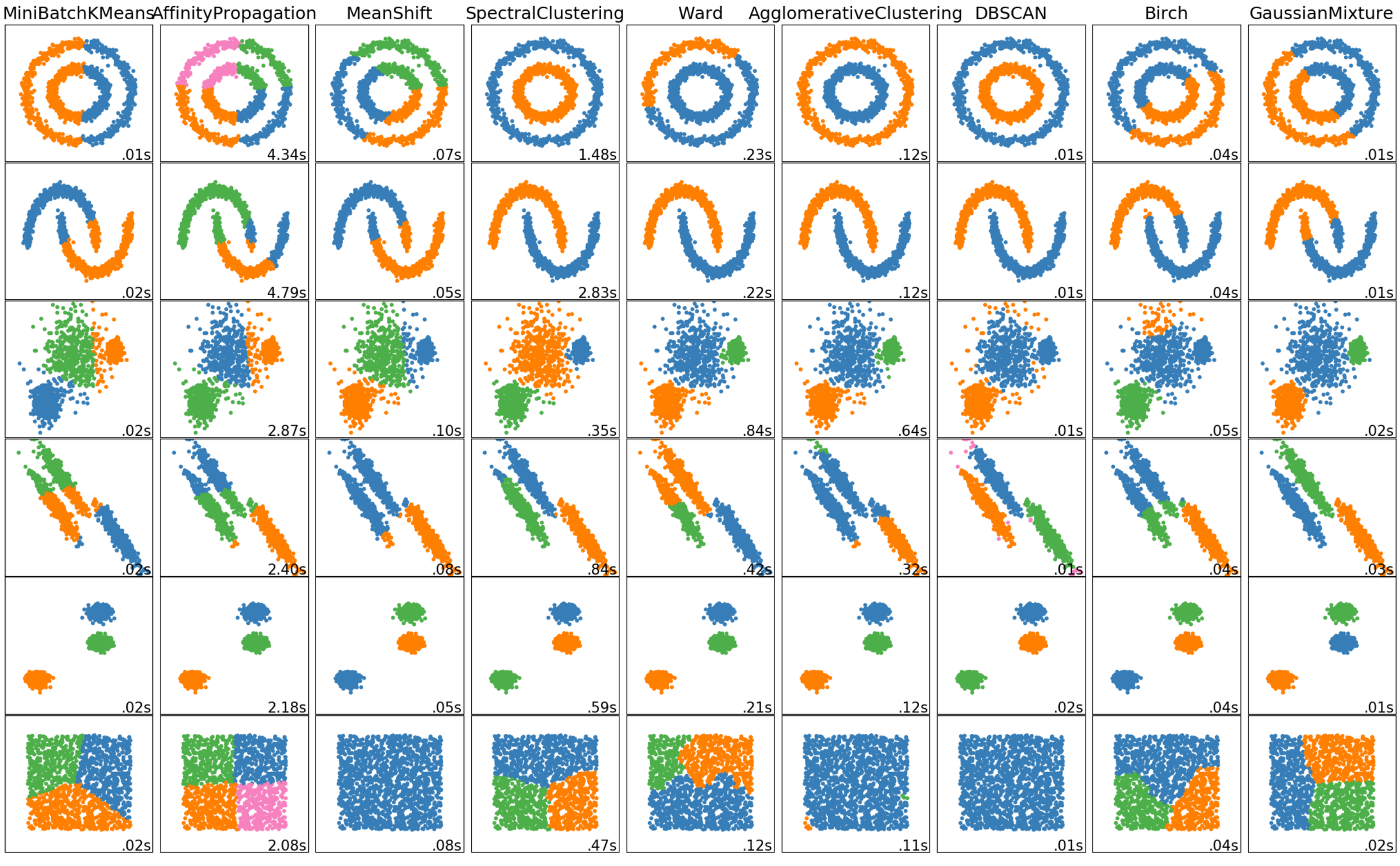
- PCA
- MDS
- tSNE
- UMAP
- Autoencoder NN

Clustering of structures of unknown protein X

- The main objective is to provide a robust platform for clustering of protein structures from MD trajectories using no prior knowledge



Clustering algorithms

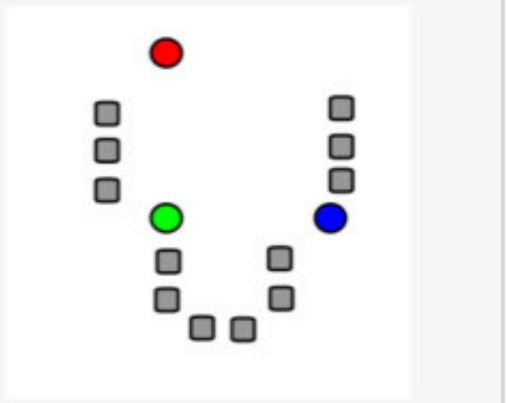


Macroparameters: algorithm

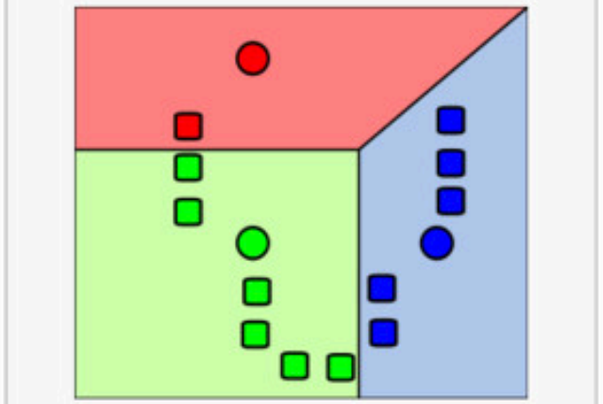
Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

K-Means

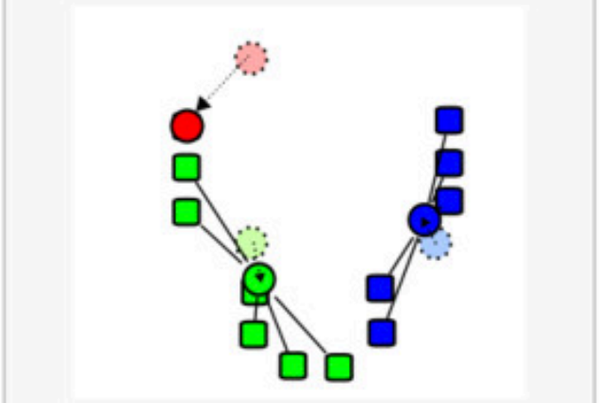
Demonstration of the standard algorithm



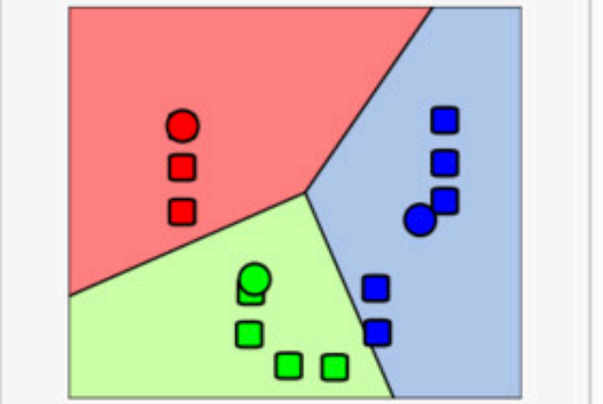
1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).



2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.



3. The centroid of each of the *k* clusters becomes the new mean.

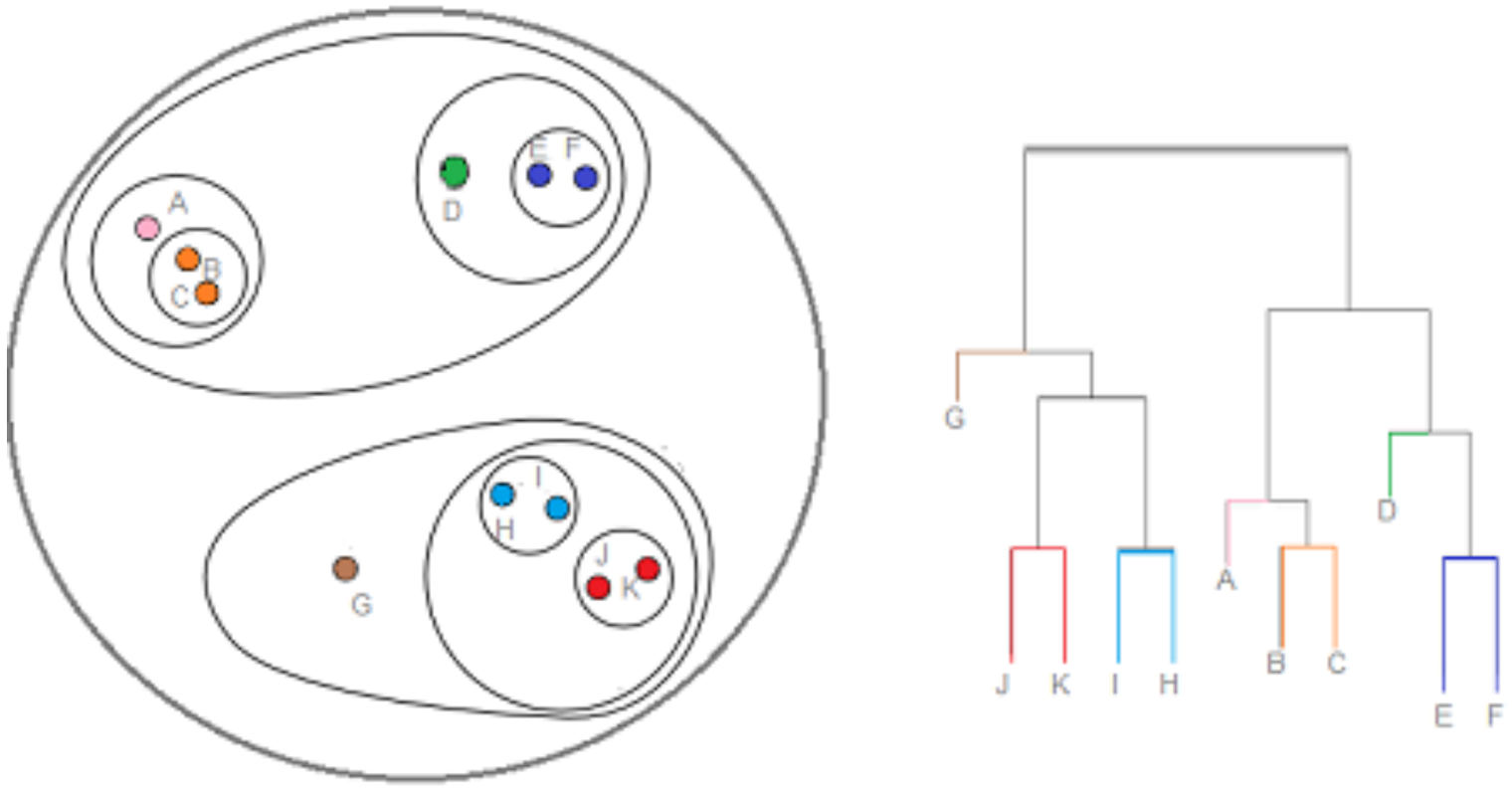


4. Steps 2 and 3 are repeated until convergence has been reached.

Hierarchical (Agglomerative) Clustering

- We do not know number of clusters, that's why hierarchical clustering algorithm can be a good choice

- Bottom-up approach: each observation starts in its own cluster, and then pairs are merged until we get 1 cluster containing all points



- The standard algorithm for hierarchical agglomerative clustering has a time complexity of $O(n^3)$ and requires $O(n^2)$ memory, which makes it too slow for even medium data sets.

Hierarchical (Agglomerative) Clustering

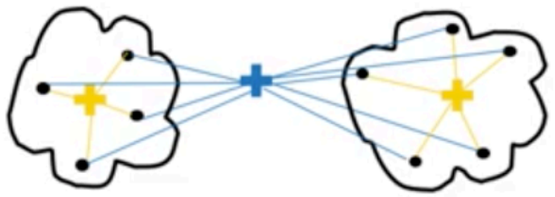
- Metric is how we calculate distance between feature vectors in high-dimensional space

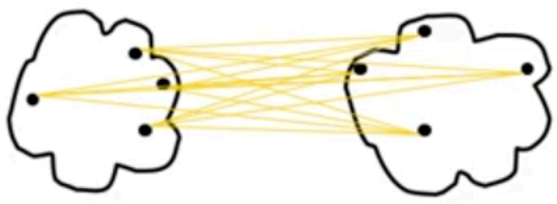
Names	Formula
Euclidean distance	$\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
Squared Euclidean distance	$\|a-b\|_2^2 = \sum_i (a_i - b_i)^2$
Manhattan distance	$\|a-b\|_1 = \sum_i \|a_i - b_i\|$
Maximum distance	$\|a-b\|_\infty = \max_i \|a_i - b_i\|$
Mahalanobis distance	$\sqrt{(a-b)^\top S^{-1} (a-b)}$ where S is the Covariance matrix

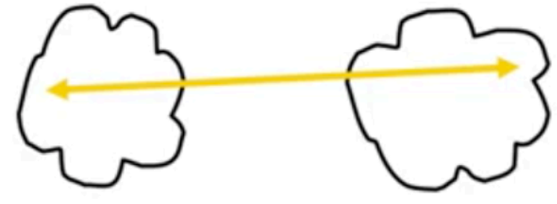
Hierarchical (Agglomerative) Clustering

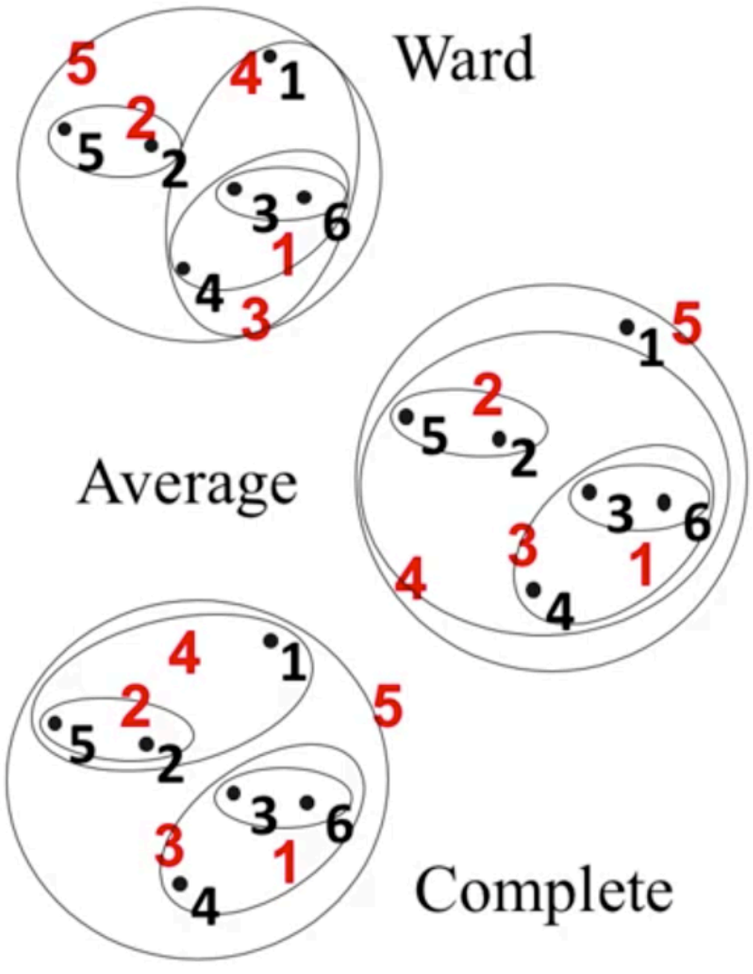
- Linkage is the method used to measure distance between newly created clusters

- **Ward's method**
  - *Least increase in total variance (around cluster centroids)*
- **Average linkage**
  - *Average distance between clusters*
- **Complete linkage**
  - *Max distance between clusters*



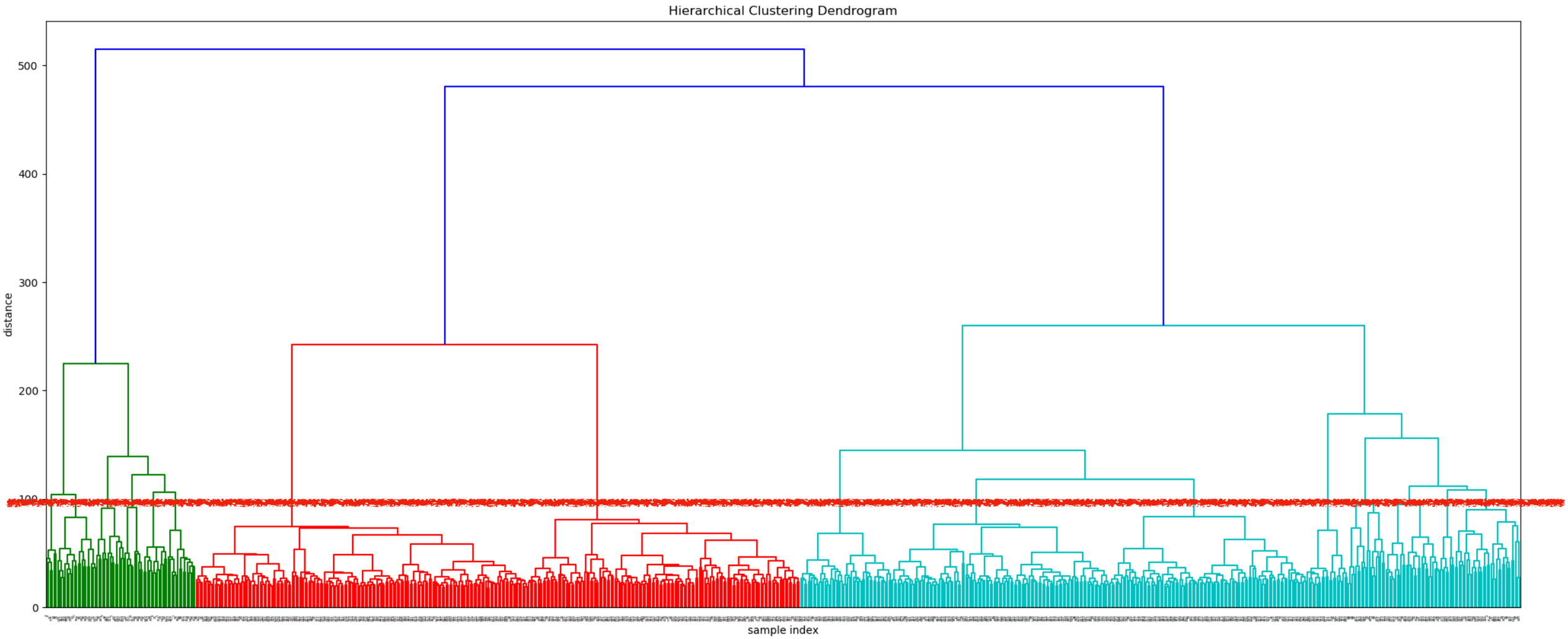


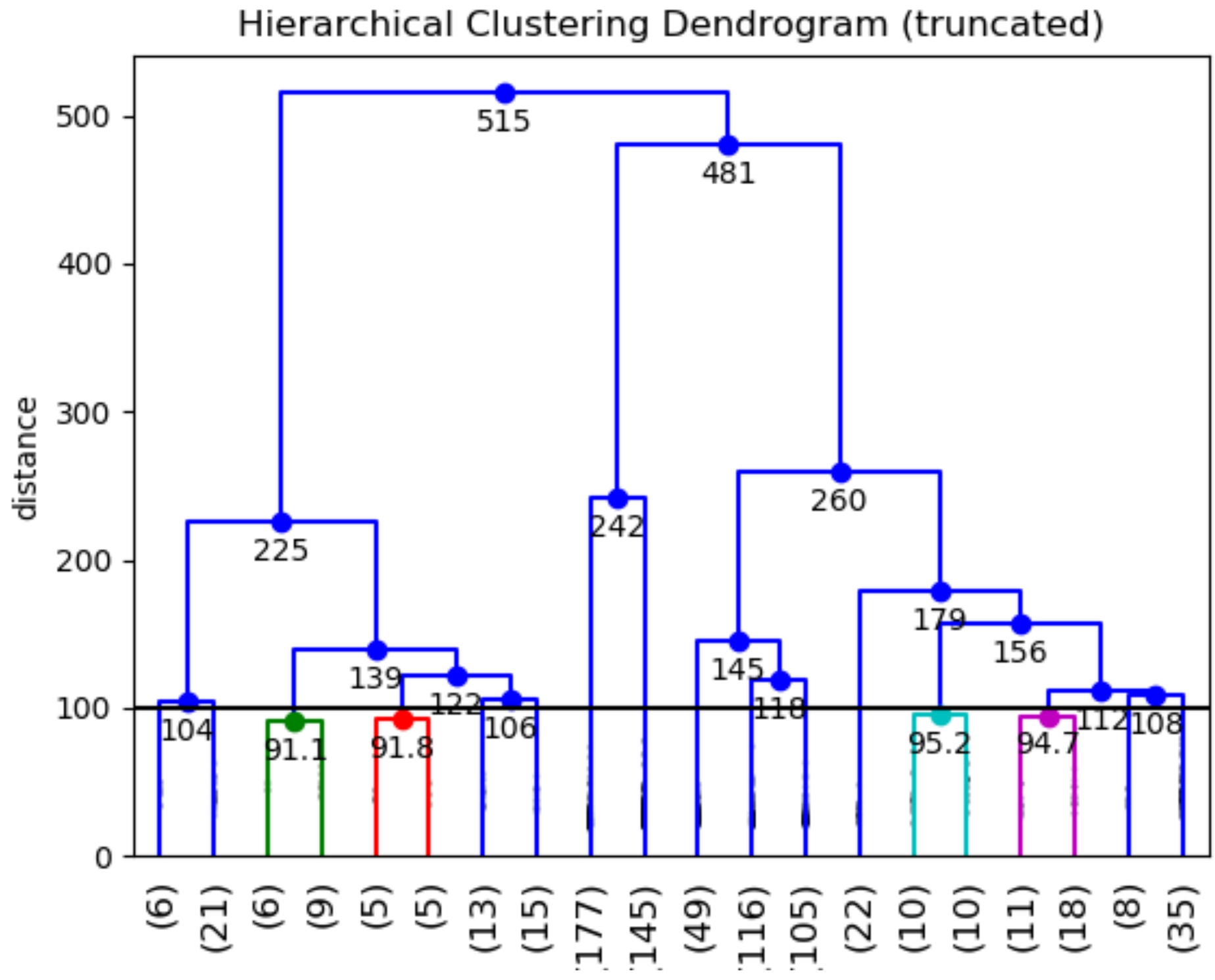




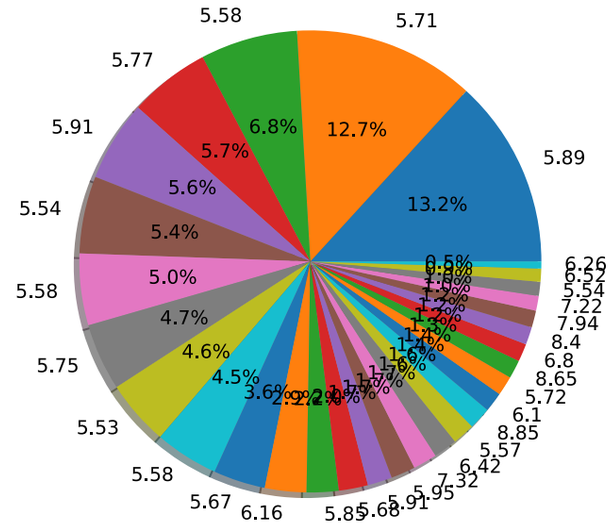
***cophenetic index to compare efficiency of linkage criteria**

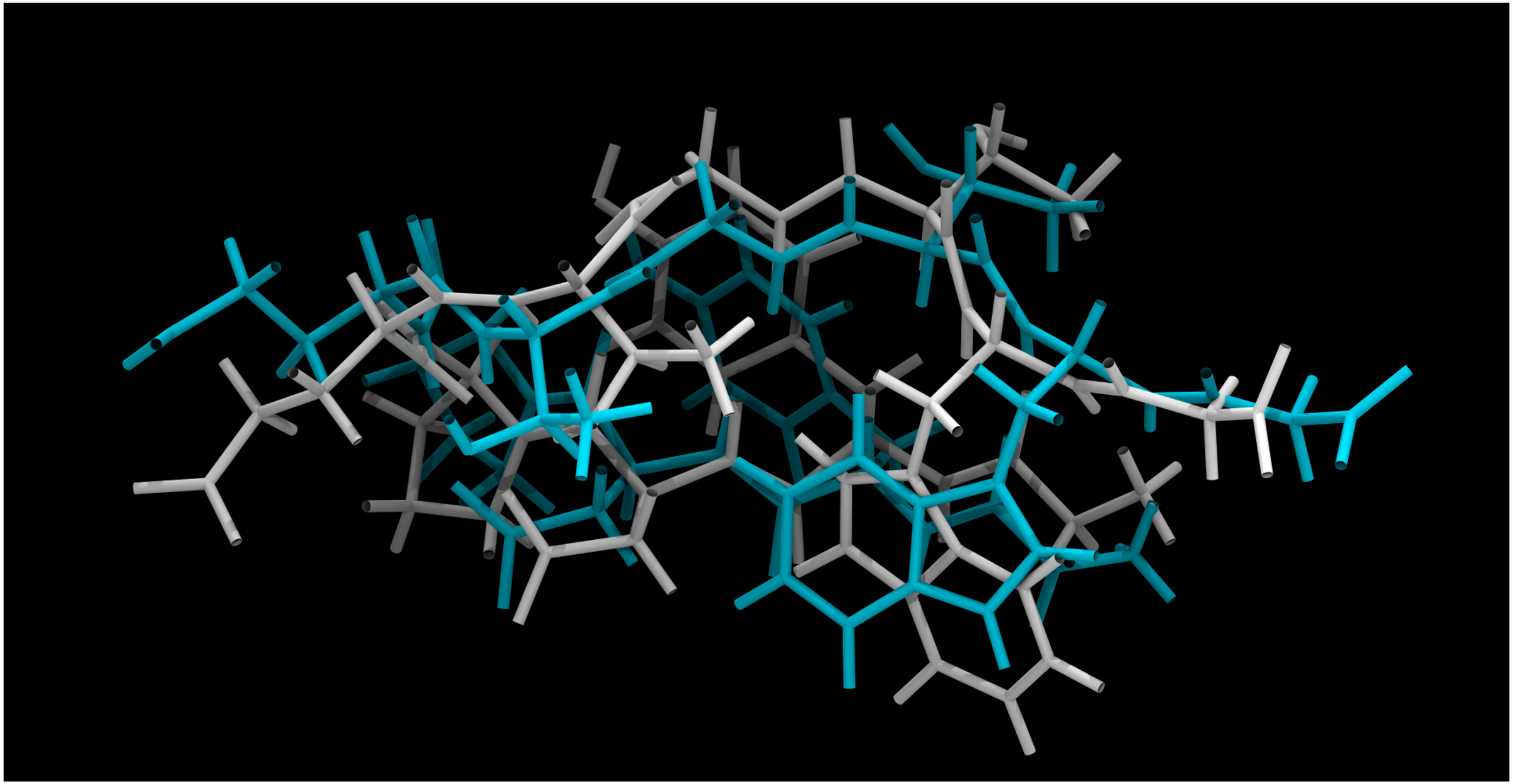
Full dendrogram



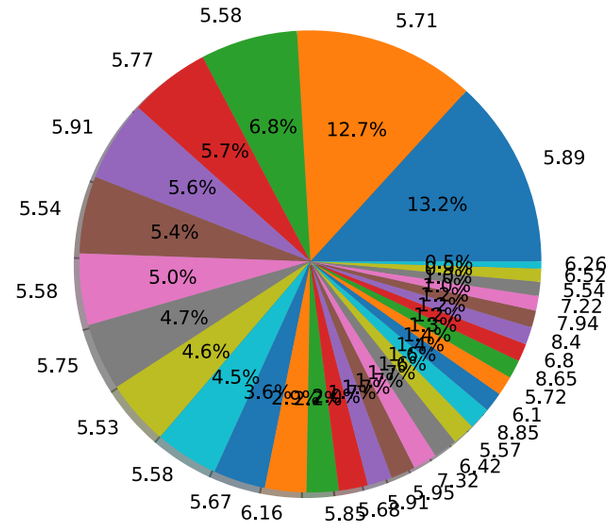

Hierarchical Clustering Dendrogram (truncated)
distance
500
400
300
200
100
0
515
481
225
242
260
139
122
145
118
179
156
104
91.1
91.8
106
95.2
94.7
112
108
(6)
(21)
(6)
(9)
(5)
(5)
(13)
(15)
(177)
(145)
(49)
(116)
(105)
(22)
(10)
(10)
(11)
(18)
(8)
(35)

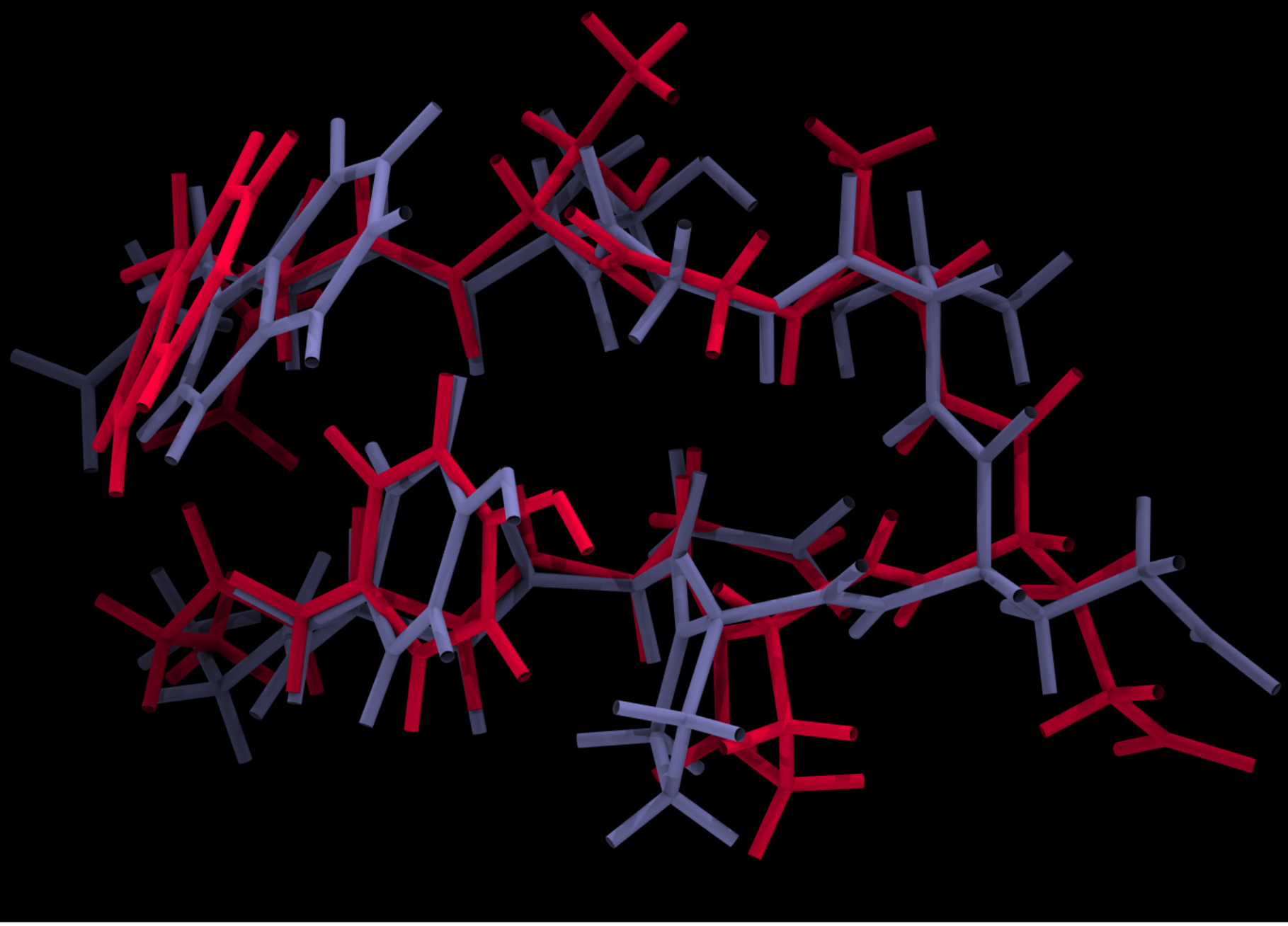
Visualizing (1-2)



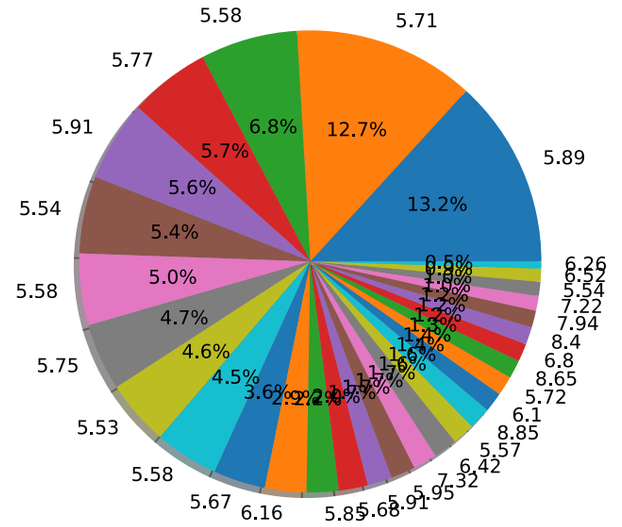


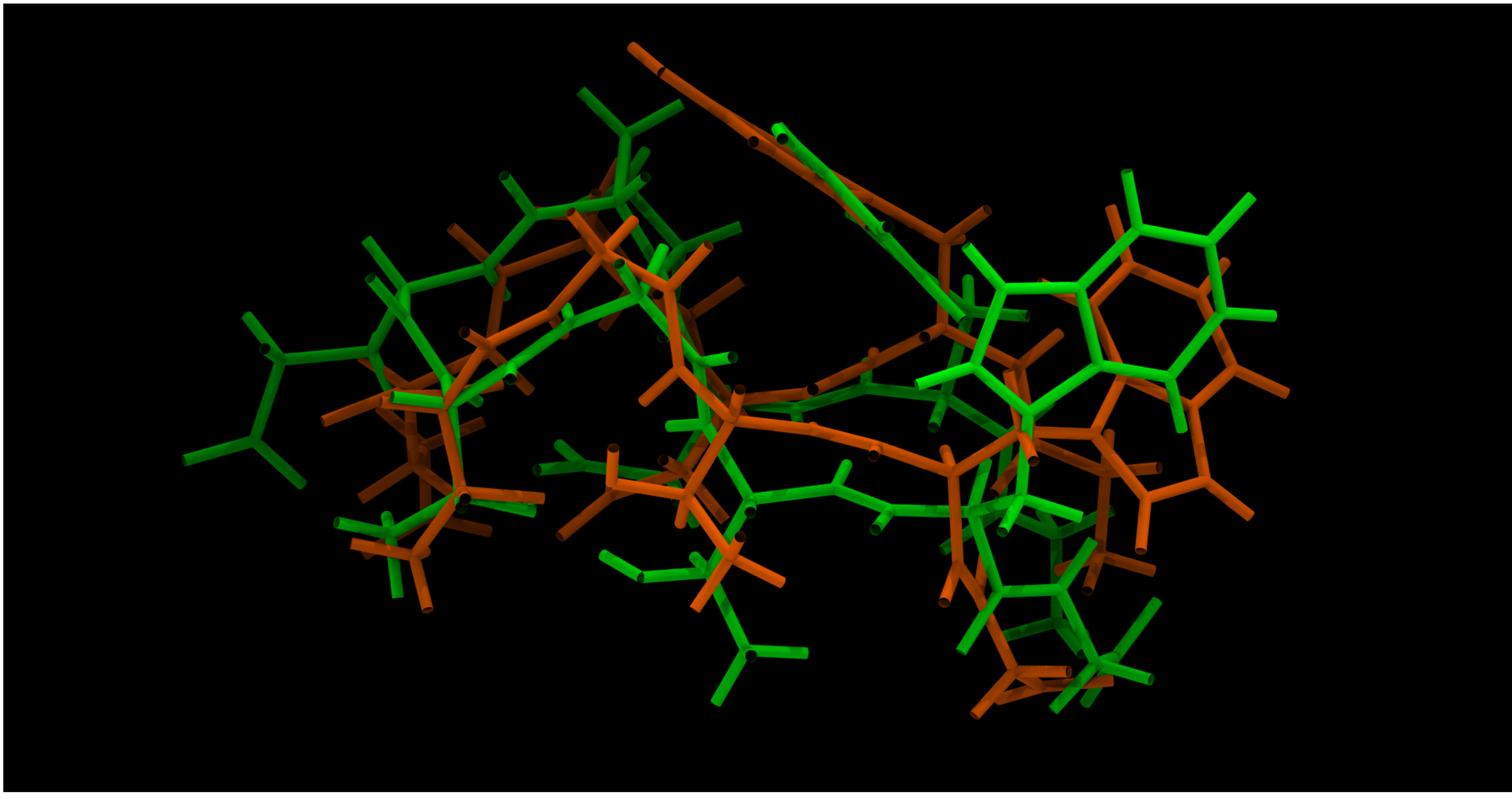
Visualizing (3–4)





Visualizing (5–6)



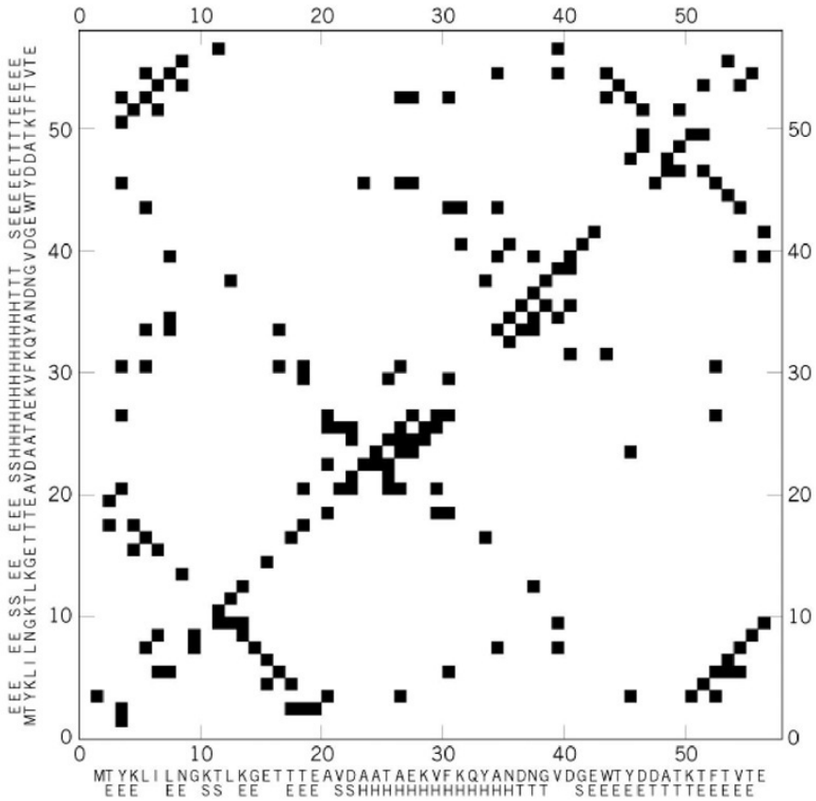


Feature vector

- Last time we agreed that contact map vector should be changed. Instead of determining the contact for every heavy atom, now we consider contact for each amino acid only (except itself and two neighbouring)

- If any two heavy atoms in the amino acid are closer than cutoff 5.5 Å, then **1**, else **0.** Total size 36 for n = 10 amino acids.

- (Let's also add radius of gyration as the 37th feature)



Feature vector - contact map

$$\mathbf{X} = \begin{vmatrix} 1 \\ 0 \\ \dots \\ 0 \\ 1 \end{vmatrix}$$

Total dimension = 36

~~12,000~~

# of data points = 780,000

# of unique data points = 16,000

- I used ‘cosine’ metrics - I’ll explain why
- the result doesn’t change much in terms of linkage ‘method’
- The results will be shown for ‘ward’ linkage

Metrics: cosine

- Euclidean distance has the property that true values and false values for features carry the same weight, and this does not work well for contact maps, which are generally sparse.

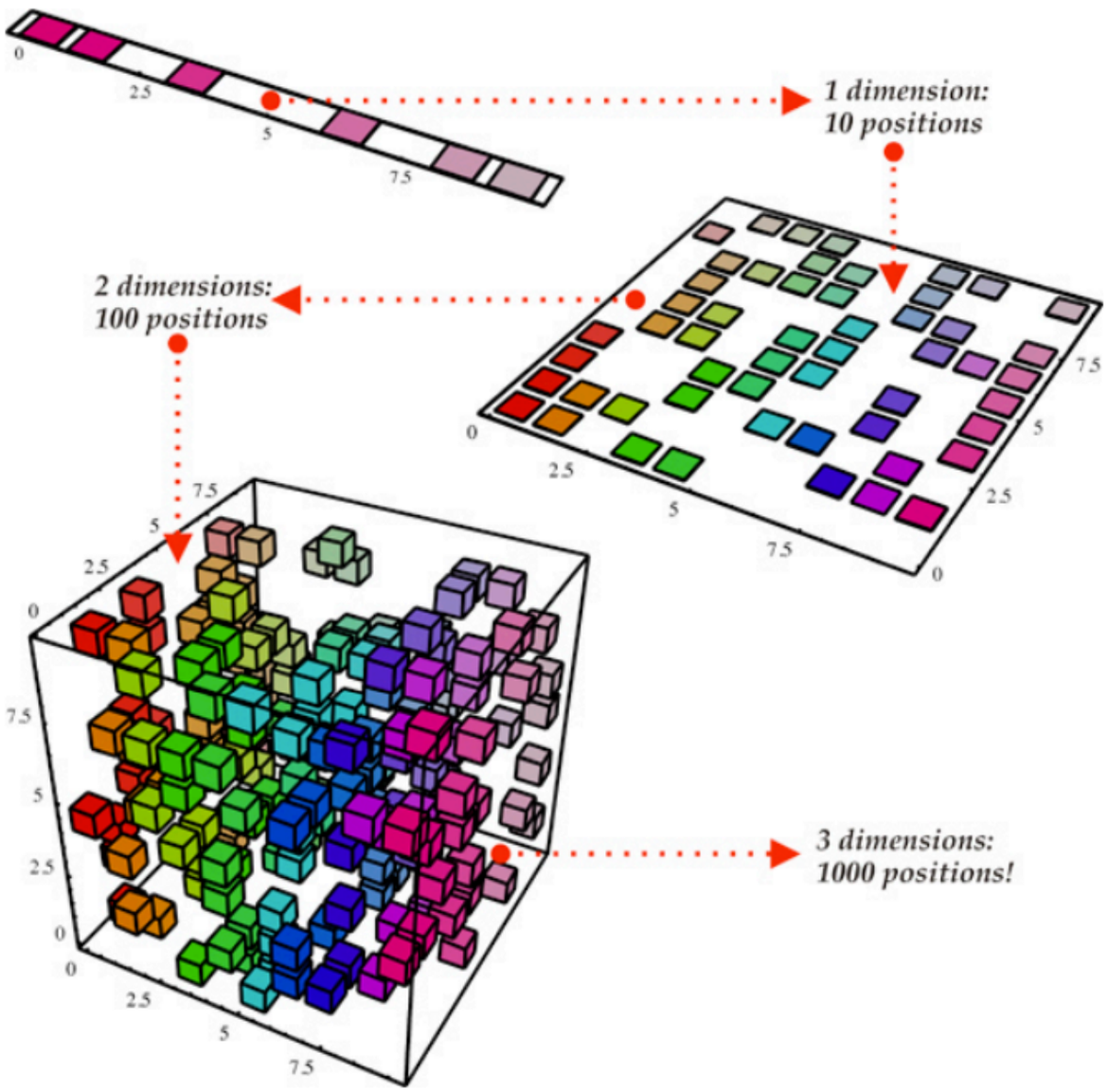
- When dealing with sparse data sets in high dimensions, it is often the case that the presence of an attribute is more significant than its absence.

the **cosine** distance between vectors u and v: $d(u, v) = 1 - \frac{u \cdot v}{||u||_2 ||v||_2}$

It is thus a judgment of orientation and not magnitude

Dimensionality reduction

- We often need this to represent multidimensional data in lower dimensional space:

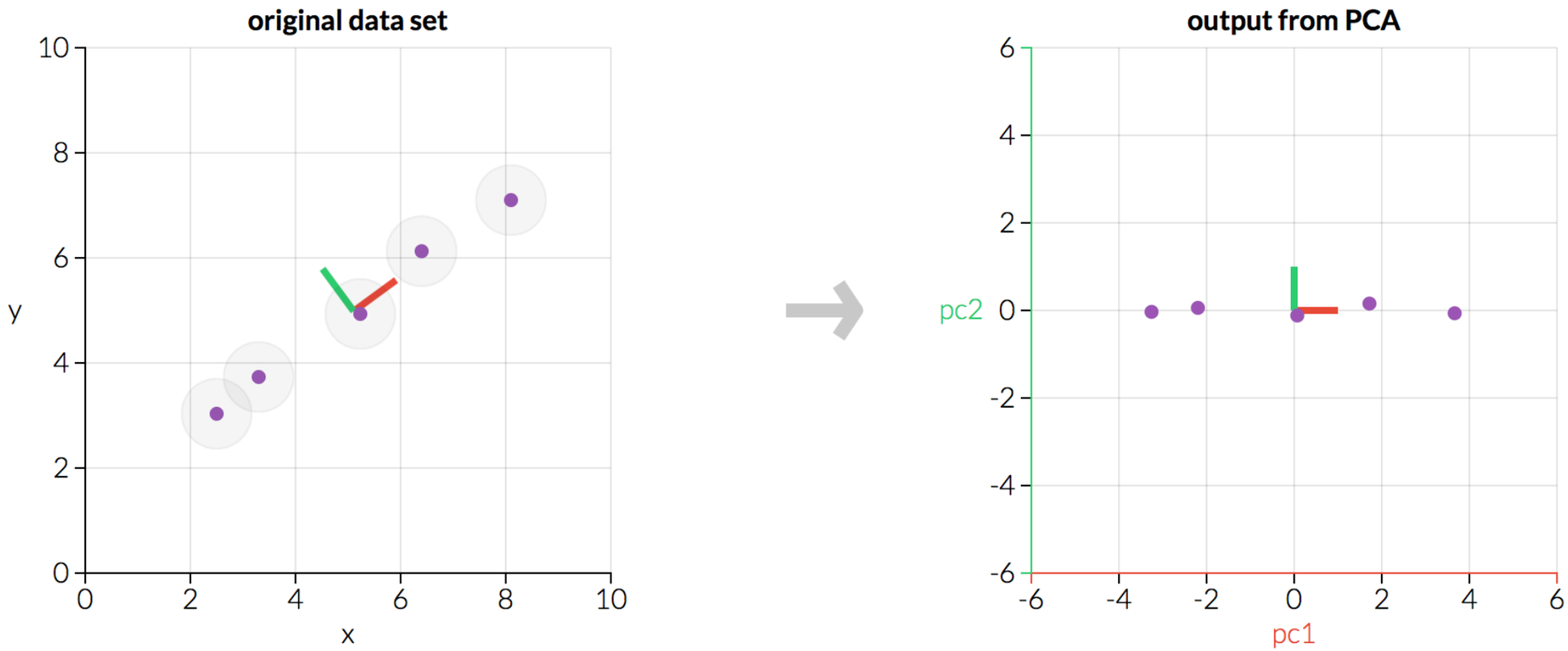


36D ⟶ **2D**

Why do we need it?

- visualization
- speed up computations
- representation
- simplification
- get insights about the data

Dimensionality reduction: PCA - Principal Component analysis



- Finding the directions of the greatest variance of data: i.e. find eigenvalues of covariance matrix, sort them in descending order

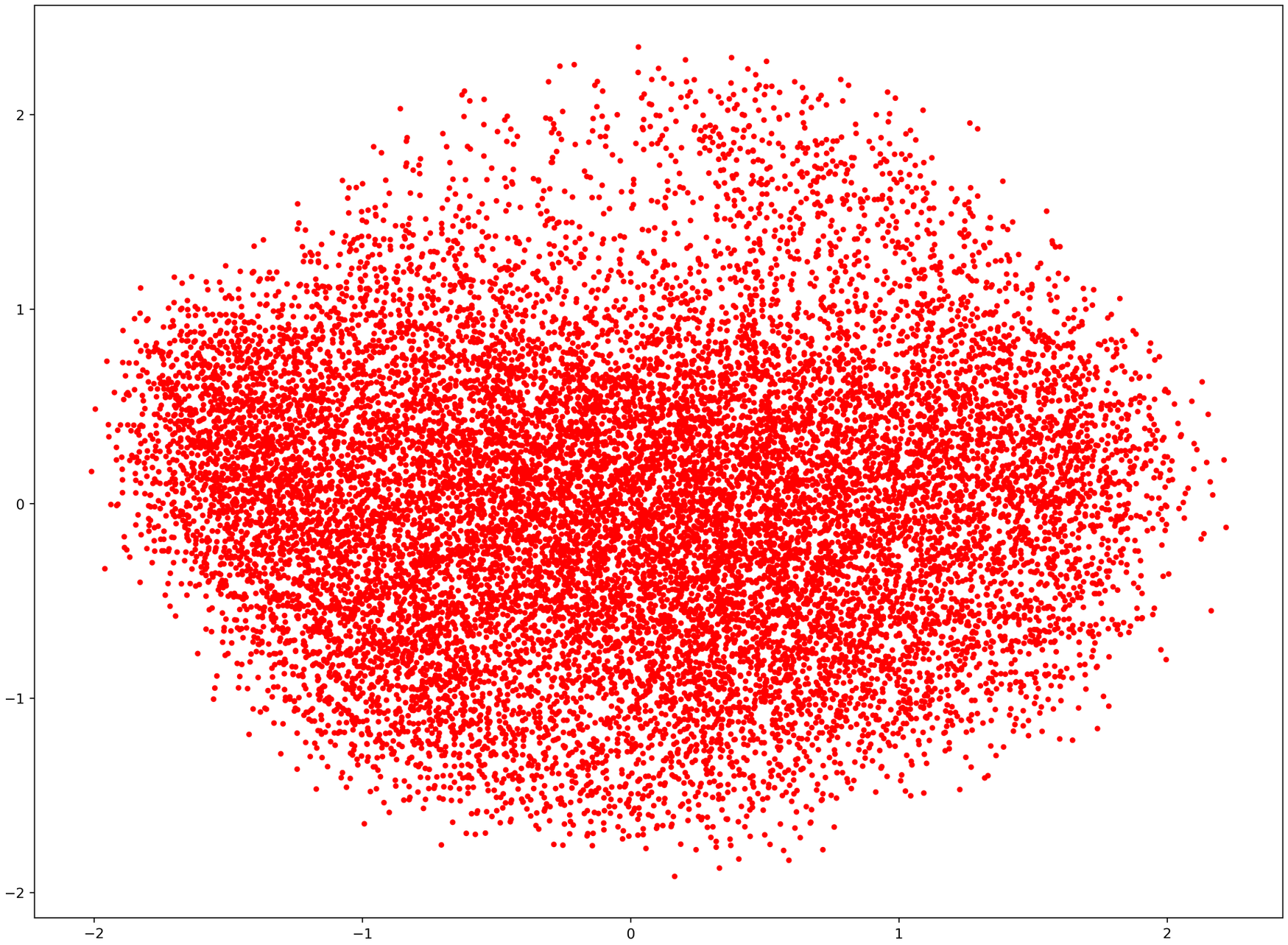
- Find projections of the data on the eigenvectors, corresponding to the largest k eigenvalues

- $\boldsymbol{W_L}$ is a matrix made up of the eigenvectors of correlation matrix

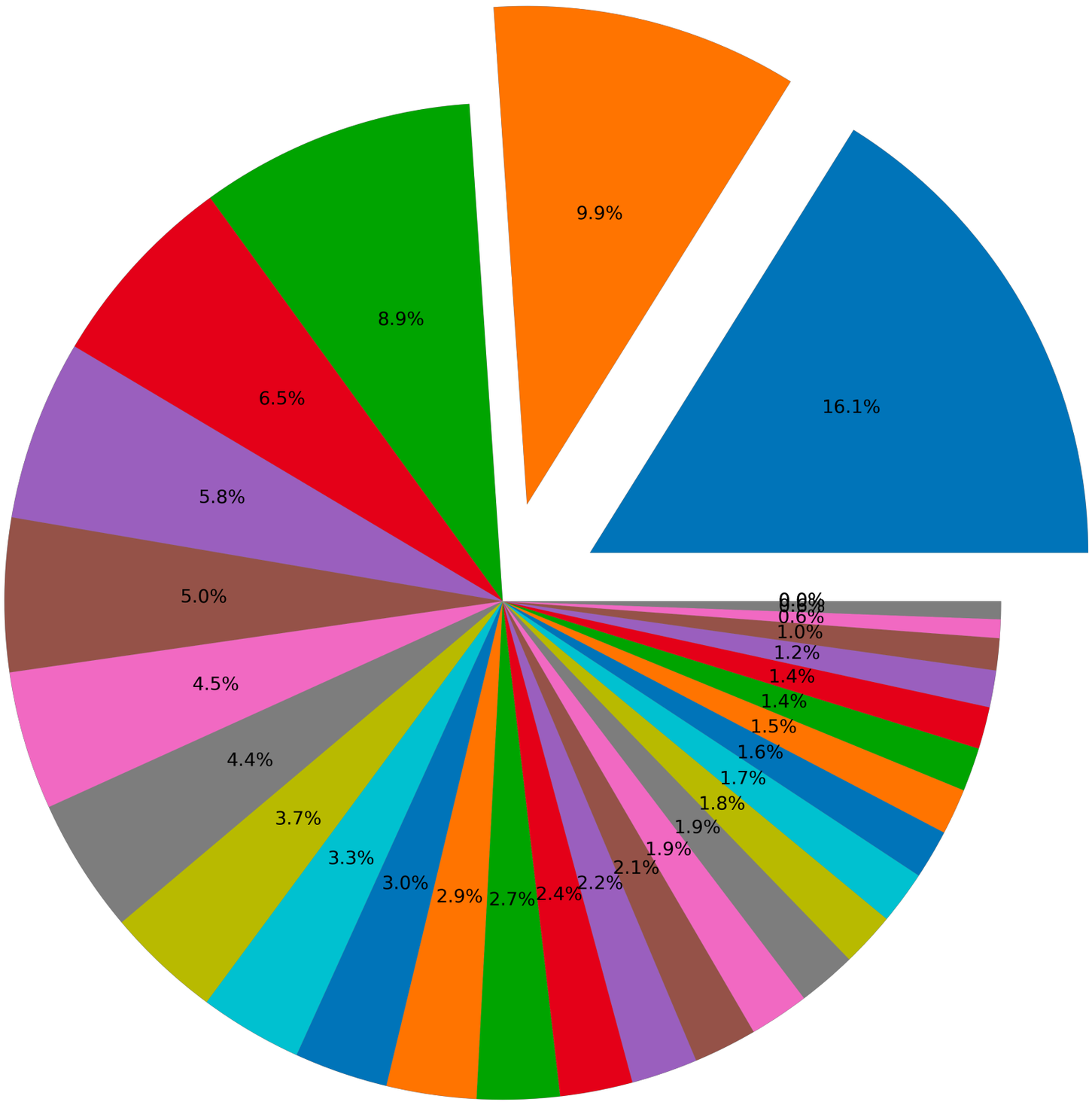
$$\mathbf{T}_L = \mathbf{X}\mathbf{W}_L$$

PCA - continue

- Our original data (16,000 points) represented by first 2 PCAs:

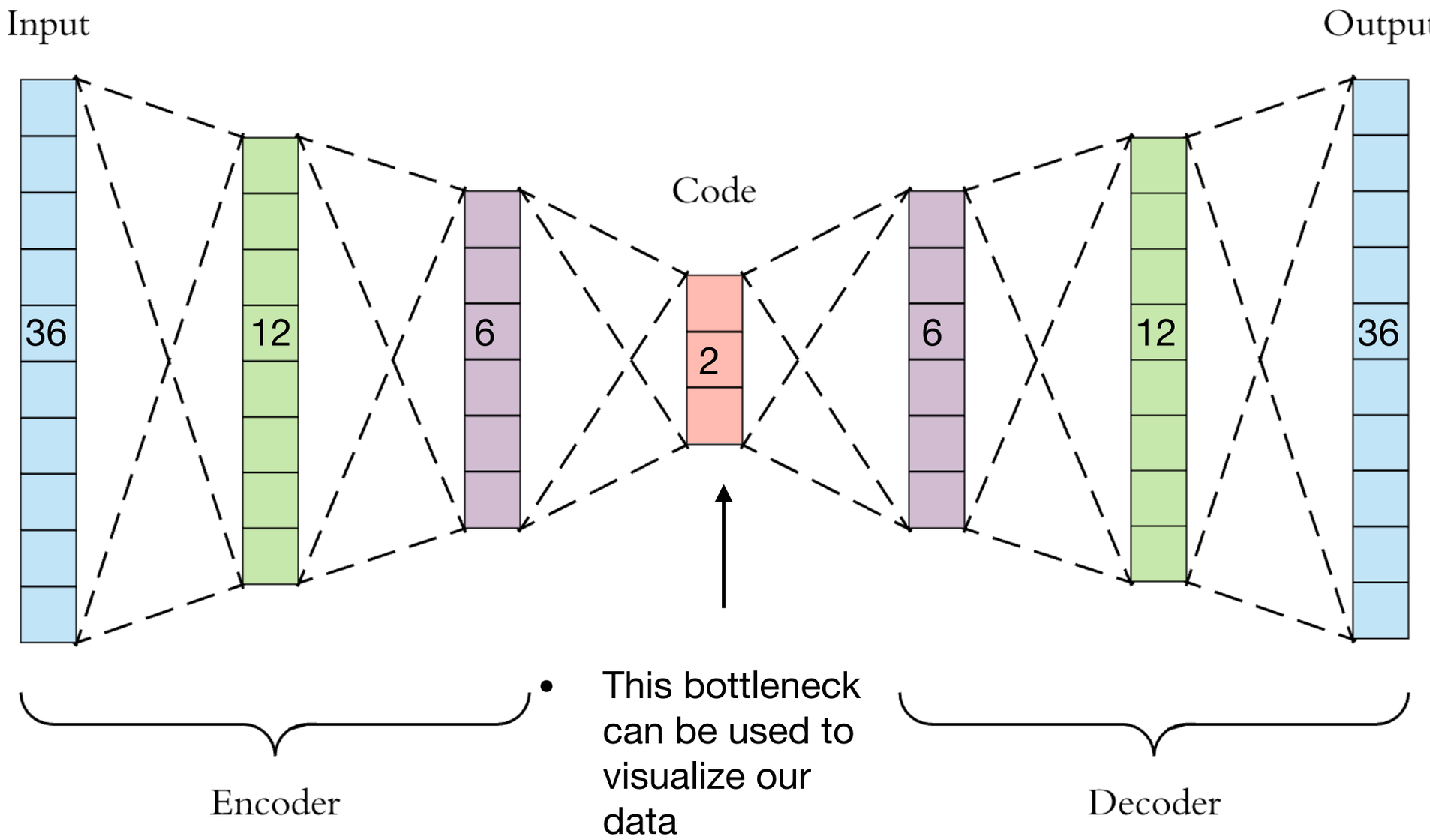


PCA - continue even more



- As you see, 2 first principal components make up only 26% of total information of 36 dimensions
- Hence, we need some other approach
- NN!

Dimensionality reduction - autoencoder Neural Network

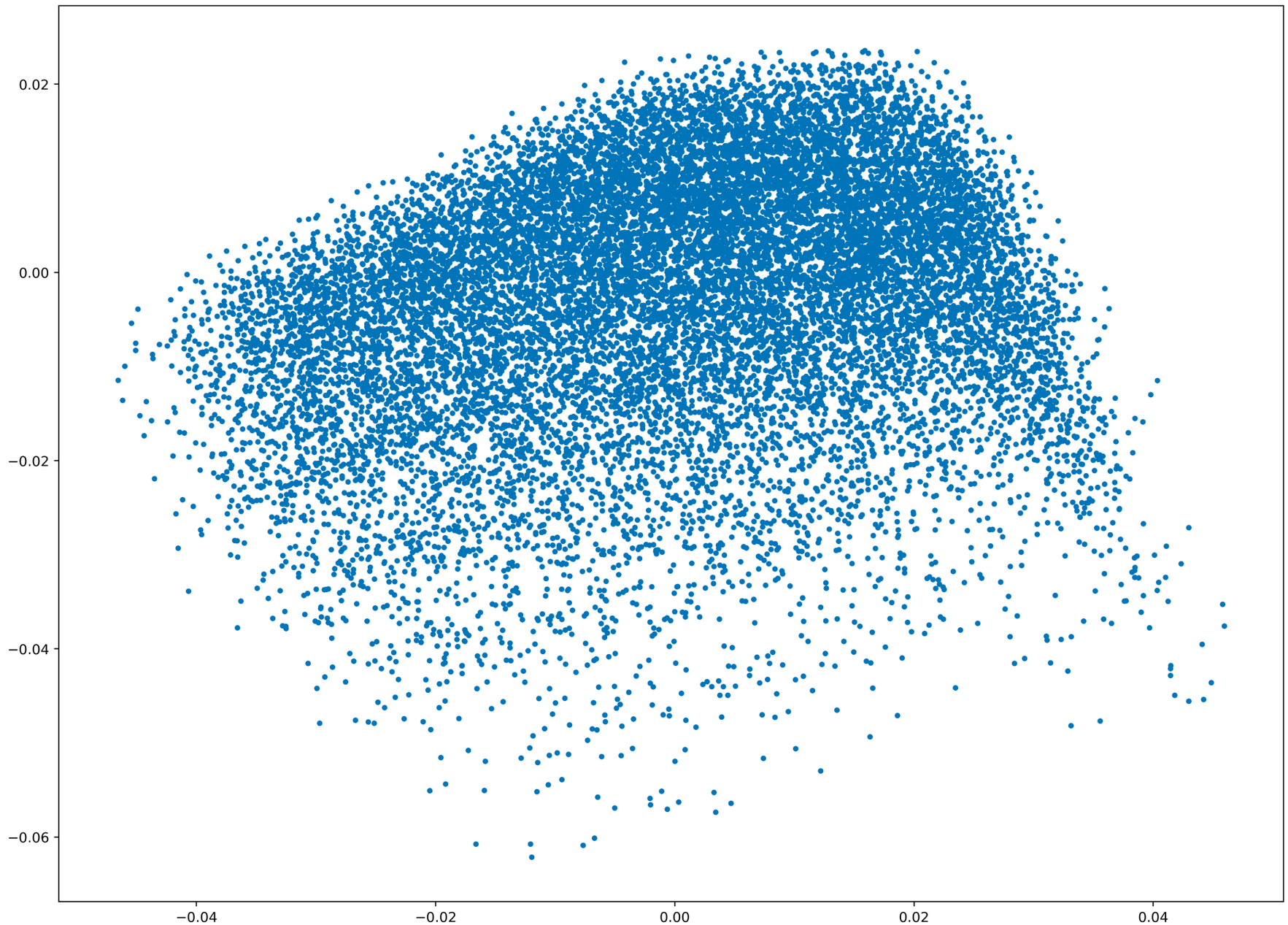


- This bottleneck can be used to visualize our data

- Used Keras with TensorFlow to build NN: tanh activation functions and regularization are used

- Trained on our 16,000 data points

Autoencoder Neural Network - continue

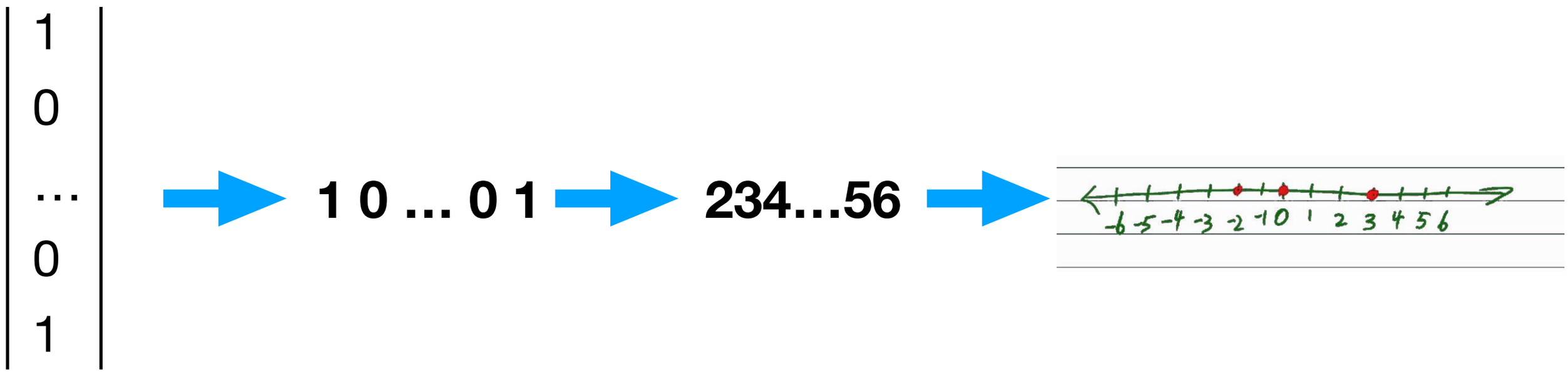


- 85% accuracy (in other words, it’s 85% of information)
- Unfortunately, it doesn’t give us any ideas about the number of clusters

My (~~maybe~~) stupid idea

- Our 36D vector is not quite in 36 dimensions of real numbers, right?

- Features can be only 0 or 1
- So let's represent every data point as a binary 36-bit number.
- Computers can deal with 64-bits
- Convert to 64-bit decimal integer
- Plot all integers on the coordinate axis. Voila! 1D is enough



The problem is that we need not only visualization, but the information about the pairwise distances. It should conserve

Dimensionality reduction - SNE

Stochastic Neighbour Embedding (SNE)

pj|i is a conditional probability for point xi to pick xj as its neighbour [original multidimensional space]

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

qj|i is a conditional probability for yi to pick yj as the neighbour [lower dimensional space]

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)}.$$

Cost function [KL-divergence]

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}},$$

Gradient of the cost function for minimization

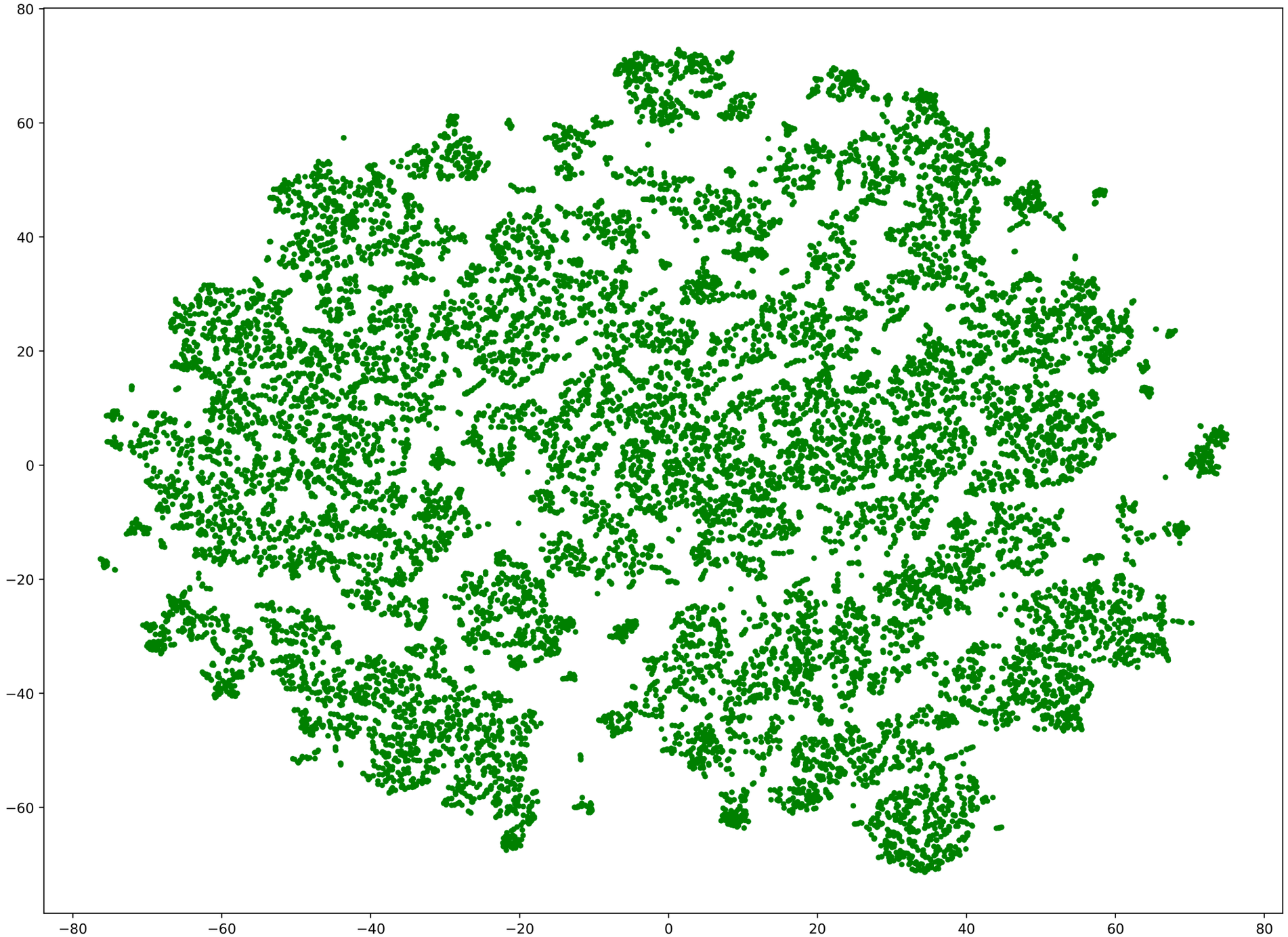
$$\frac{\delta C}{\delta y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j).$$

Gradient decsend method

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$$

t-SNE - modified SNE

tSNE - continue



Great tool for visualization and determining number of clusters