t-SNE

[T-distributed Stochastic Neighbor Embedding]

Ayush Thada
16BCE1333

- Most data-sets exhibit non-linear relationship among features and data points reside in high dimensional space.
- Therefore, we want a low-dimensional embedding of high-dimensional data that preserves the relationship among different points in the original space in order to visualize data and explore the inherent structure of data such as clusters.
- In general cases many linear dimensionality methods such as PCA and classical manifold embedding algorithms such as Isomap fail.
- t-SNE is a dimension reduction/data visualization method
- Proposed by Laurens van der Maaten & Geoffrey Hinton in 2008
- t-SNE tends to preserve local structure at the same time preserving the global structure as much as possible

SNE
Optimization

- t-SNE is a dimension reduction/data visualization method
- Proposed by Laurens van der Maaten & Geoffrey Hinton in 2008
- t-SNE tends to preserve local structure at the same time preserving the global structure as much as possible
- Aim is to match distributions of distances between points in high and low dimensional space via conditional probabilities
- Assume distances in both high and low dimensional space are Gaussian-distributed

- Let x_i be the i^{th} object in high dimensional space.
- Let y_i be the i^{th} object in low dimensional space.
- Construct the conditional distribution:

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

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

- Let x_i be the i^{th} object in high dimensional space.
- Let y_i be the i^{th} object in low dimensional space.
- Construct the conditional distribution:

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

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

- $p_{i|i} = q_{i|i} = 0$
- Match these functions by minimizing sum of Kullback-Leibler (KL) divergences:

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

- Since KL divergence is asymmetric,
  - large cost for representing nearby data points in high dimensional map by widely separated points in the low dimensional map.
  - smaller coast for representing widely separated data points in high dimensional map by nearby points in the low dimension.

- Due to asymmetric nature of KL Divergence local structure is highly preserved.
- σ_i is associated with a parameter called perplexity which can be loosely interpreted as the number of close neighbors each point has.
- Σ_i is found via binary search (More in the paper)
- Gradient of the cost function

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

- Given the cost function SNE uses gradient descent for optimization.

- Due to non convex nature of loss function there are several local minima. Hence in addition to the gradient of the cost function it also includes a momentum term to speed up the optimization and to avoid local optima.

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

where
α(t): Momentum at iteration t
y(t) : Solution at iteration t
η: Learning rate

Drawbacks of SNE & Introduction to Symmetric SNE

- SNE has two main drawbacks:
  - Cost function is difficult to optimize
  - Crowding problem
- Also, SNE is not robust to outliers. Hence symmetric SNE is introduced.
- The main feature in symmetric SNE is that $p_{ij} = p_{ji}$ and $p_{ii} = q_{ii} = 0$ for all i,j.

$$q_{ij} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq l} exp(-||y_k - y_l||^2)}$$

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_k - x_l||^2/2\sigma^2)}$$

- Gradient of the cost function:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ji} - q_{ji})(y_i - y_j)$$

t-SNE

- To address the crowding problem and make SNE more robust to outliers, t-SNE was introduced. Compared to SNE, t-SNE has two main changes:
  - A symmetrized version of the SNE cost function with simpler gradients
  - A Student-t distribution rather than a Gaussian to compute the similarity in the low-dimensional space to alleviate the crowding problem.
- In SNE, p_{ij} is not necessarily equal to p_{ji}, because τ_{ij} is not necessarily equal to τ_{ji}.
- This makes SNE prone to outliers, because an outlier xi would have very small pji for all other points, and so its embedded location becomes irrelevant. Thus, in t-SNE

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

- In t-SNE a student t distribution with one degree of freedom (Cauchy distribution) is used to represent the low dimensional map:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

- t-distribution is robust to outliers and unlike a Gaussian distribution it doesn't have exponent in it so faster to evaluate.
- Gradient of the cost function:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ji} - q_{ji})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}$$

$$\frac{dC}{dy_i} = 4 \sum_{j=1, j \neq i}^{n} (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-1}(y_i - y_j)$$

$$= 4 \sum_{j=1, j \neq i}^{n} (p_{ij} - q_{ij}) q_{ij} Z (y_i - y_j)$$

$$= 4 \Big(\sum_{j \neq i} p_{ij} q_{ij} Z (y_i - y_j) - \sum_{j \neq i} q_{ij}^2 Z (y_i - y_j) \Big)$$

$$= 4(F_{attraction} + F_{repulsion})$$

where Z = $\sum_{l,s=1, l \neq s}^{n} (1 + ||y_l - y_s||^2)^{-1}$

Algorithm 1 tSNE

Input: Dataset $X = \{x_1, ..., x_n\} \in \mathbb{R}^d$, perplexity k, exaggeration parameter α, step size $h > 0$, number of rounds $T \in \mathbb{N}$

Compute $\{p_i j : i, j \in [n], i \neq j\}$

Initialize $y_1^{(0)}, y_2^{(0)}, ..., y_n^{(0)}$ i.i.d. from the uniform distribution on $[-0.01, 0.01]^2$

for t=0 **to** T-1 **do**

$$Z^{(t)} \leftarrow \sum_{i,j \in [n], i \neq j} \left(1 + ||y_i^{(t)} - y_j^{(t)}||\right)^{-1}$$

$$q_{ij}^{(t)} \leftarrow \frac{\left(1+||y_i^{(t)} - y_j^{(t)}||\right)^{-1}}{Z^{(t)}}, \forall i, j \in [n], i \neq j$$

$$y_i^{t+1} \leftarrow y_i^{(t)} + h \sum_{j \in [n]/\{i\}} \left(\alpha p_{ij} - q_{ij}^t\right) q_{ij}^t Z^t \left(y_i^{(t)} - y_j^{(t)}\right), \forall i \in [n]$$

end for

Output: 2D embedding $Y^{(T)} = \left\{y_1^{(T)}, y_2^{(T)}, ..., y_n^{(T)}\right\} \in \mathbb{R}^2$

Notice that there is an exaggeration parameter $\alpha > 1$ in the t-SNE algorithm, which is used as a coefficient for p_{ij}. This encourages the algorithm to focus on modeling large p_{ij} by fairly large q_{ij}. A natural result is to form tightly separated clusters in the map and thus makes it easier for the clusters to move around relative to each other in order to find an optimal global organization.

The general idea of t-SNE algorithm:

- Data: data set X with data points = x1, x2, ..., xn each of these points have very high dimension.
- cost function parameter: Perplexity Perp, perplexity is associated with variance σ in the cost function
- Optimization parameters: number of iterations T, learning rate η, momentum α(t)

[Previous slides contain the snapshot of the algorithm in paper.]
[Next Slide contain the algorithm in without much mathematical notations.]

t-SNE algorithm:

Start

1. Compute pairwise affinities $p_{j|i}$ with perplexity *Perp*.
2. Set $p_{j|i} = \frac{p_{i|j} + p_{j|i}}{2n}$ [n: Number of data points]
3. Sample initial solution Y(0) = $y_1, y_2, ..., y_n$ from $N(0, 10^{-4})$
   for t=1 to T do
   a. Compute low-dimensional affinities p_{ij}
   b. Compute gradient : $\frac{\delta C}{\delta y}$
   c. Set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta y} + \alpha(t)(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)})$

   end

End

Drawbacks of t-SNE

1. The perplexity parameter needs to be chosen carefully and might need knowledge about some general knowledge about the data. Varying perplexity can give drastically different visualizations that show different

Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

2. The coordinates after embedding have no meaning. While tSNE can preserve the general structure of data in the original space such as clusters, it may distort those structure in the embedded 2D space. Therefore, the embedded tSNE components carry no inherent meaning and can merely be used for visualization.

Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

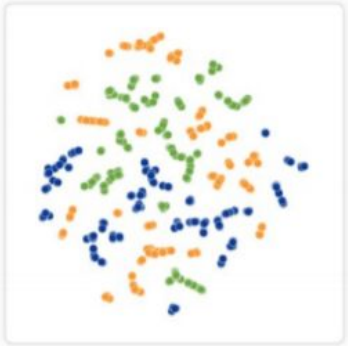
Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

3. Finally, since tSNE focuses on the local structure, the global structure is only sometimes preserved. Consequently, interpretation of the relationship between clusters cannot be obtained from t-SNE embedding alone. [tSNE fails to capture the fact that blue and orange clusters are closer to each other than to the green cluster.]

Original



Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

4. tSNE does not work well for general dimensionality problem where the embedded dimension is greater than 2D or 3D but the meaning of distances between points needs to be preserved as well as the global structure.

5. Curse of dimensionality (tSNE employs Euclidean distances between near neighbors so it implicitly depends on the local linearity on the manifold)

6. $O(N^2)$ computational complexity

7. Perplexity number, number of iterations, the magnitude of early exaggeration parameter have to be manually chosen.

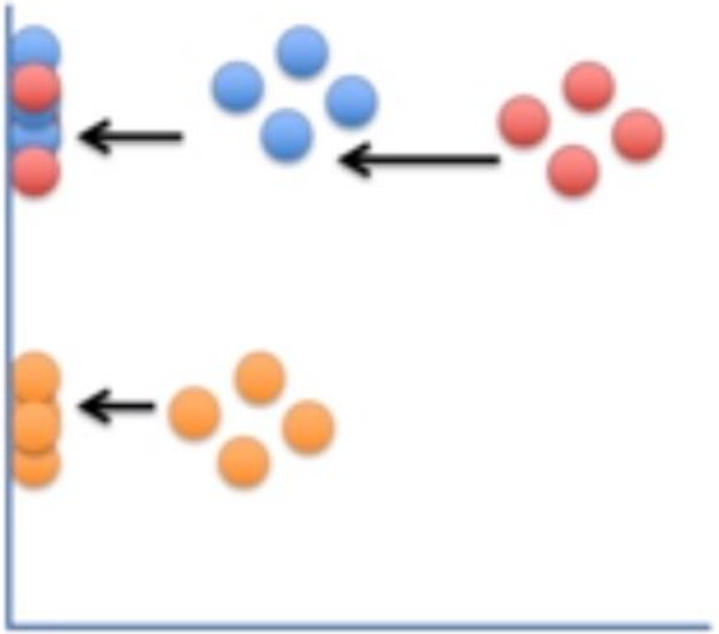
Solved

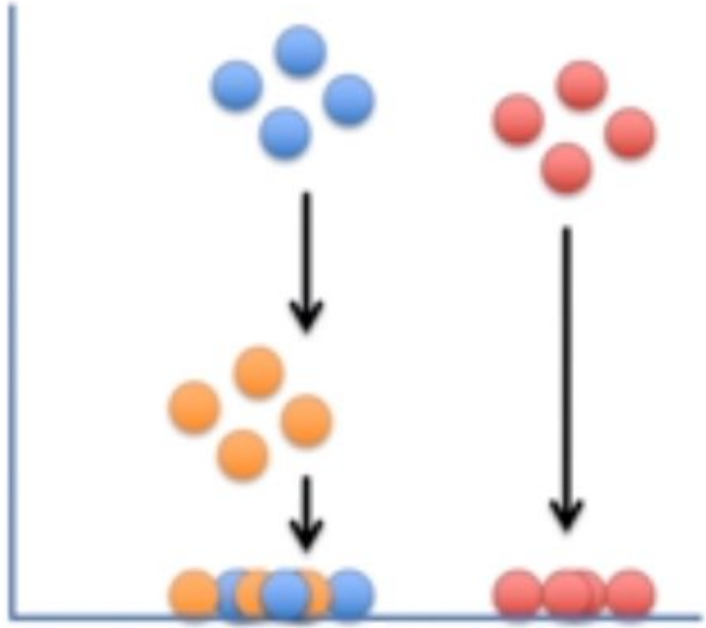
Example

Given: Data in High Dimension

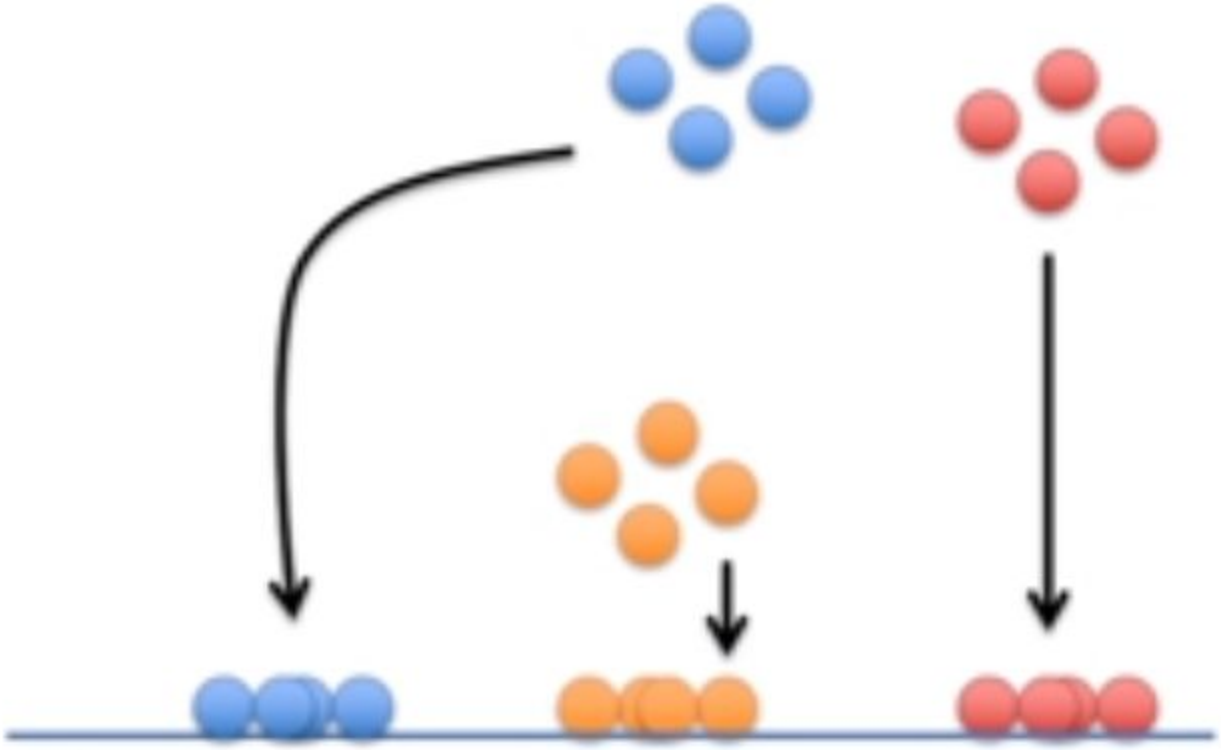
Expected Output: Data in Lower Dimension

Projection is not an option. It leads to cluttering in many cases.

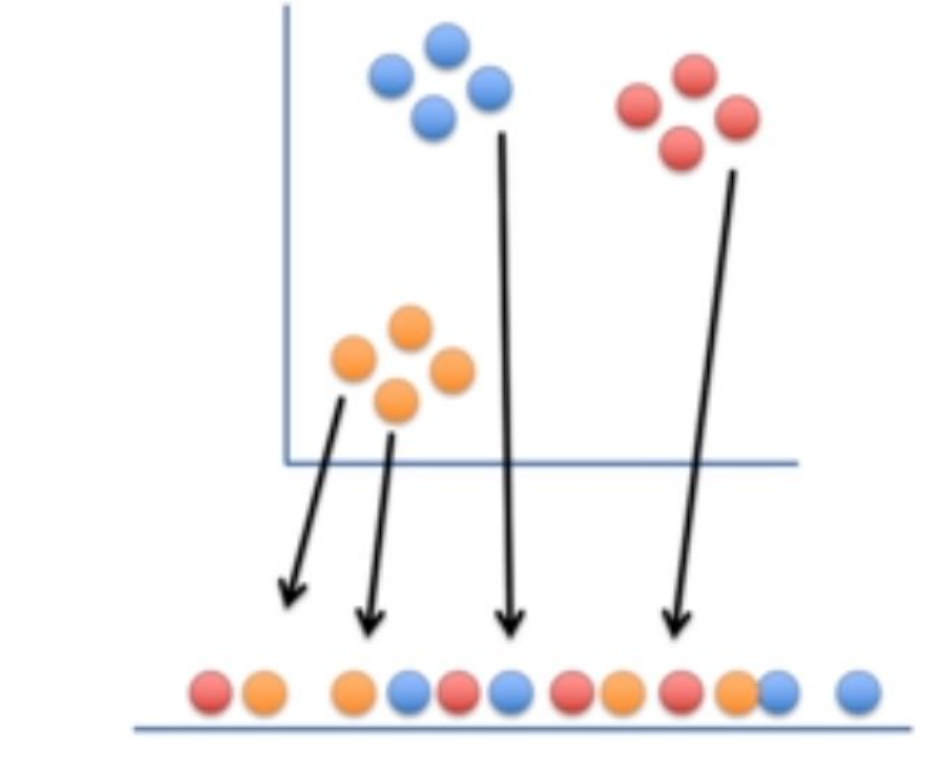




We’re expecting this kind of result.

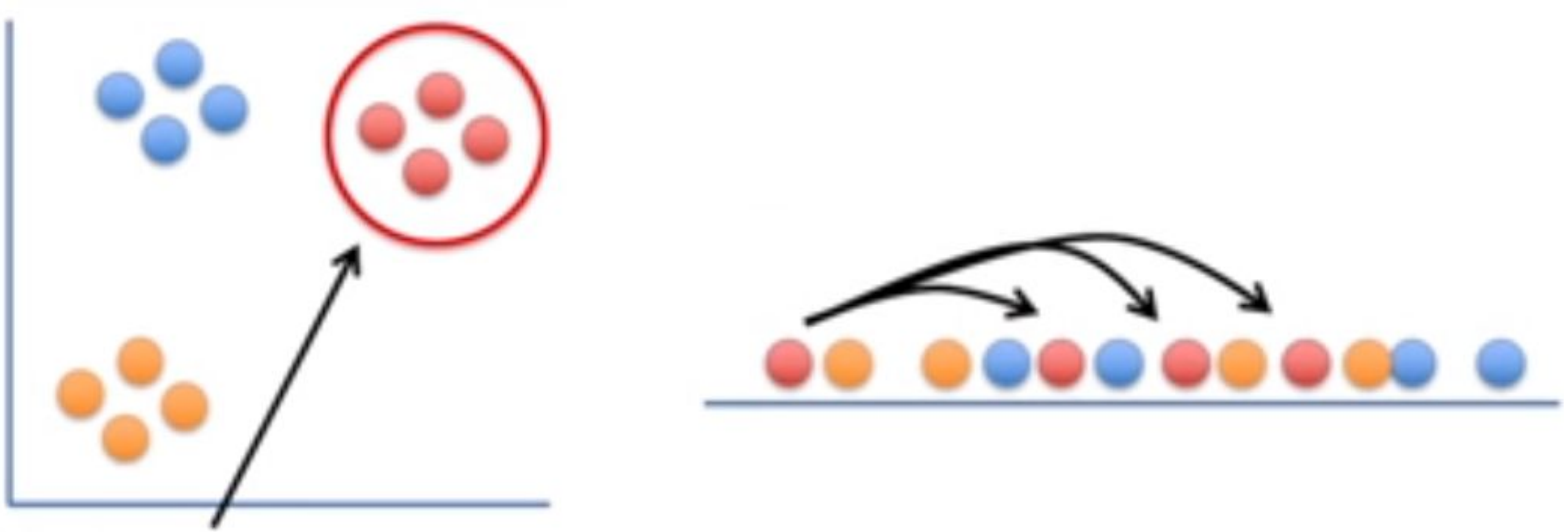


Step 1: Put the points on line in random order.

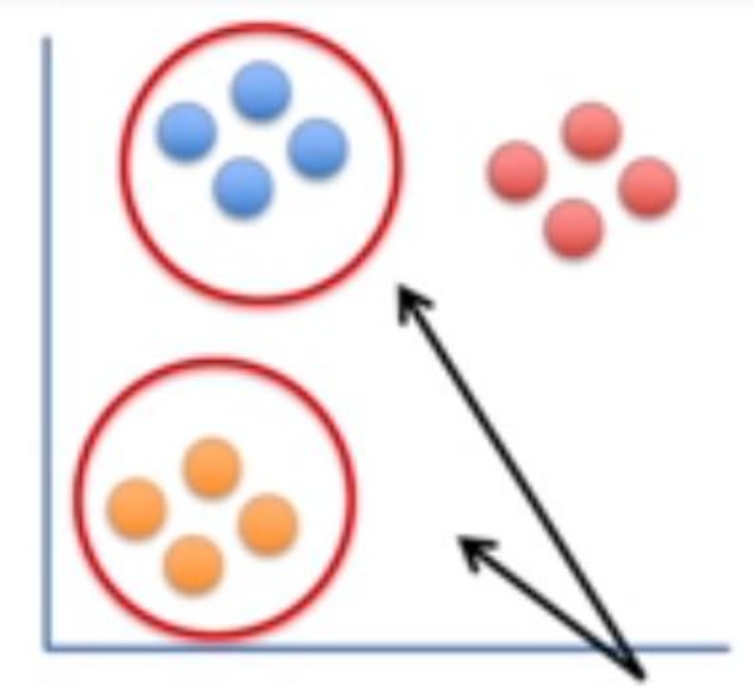


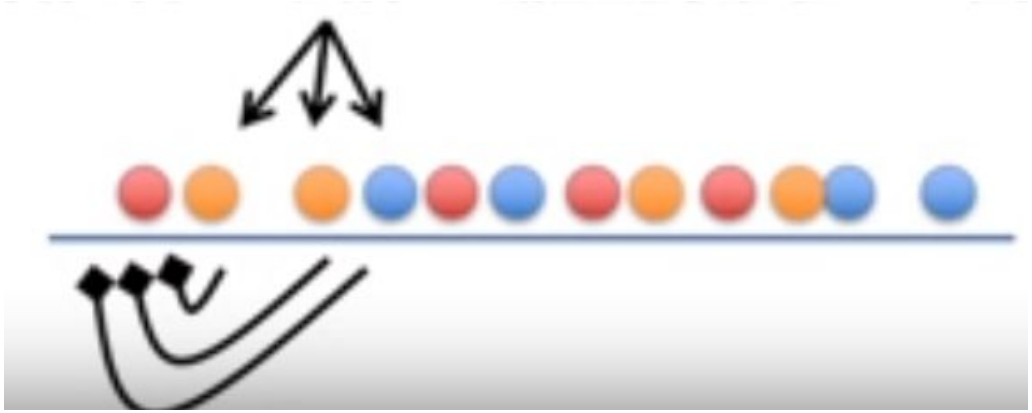
Step 2: t-SNE move these points with small displacement until they’re all clustered.

Step 3a: As red points are part of cluster in higher dimension it will be attracted to other points of cluster in lower dimension.

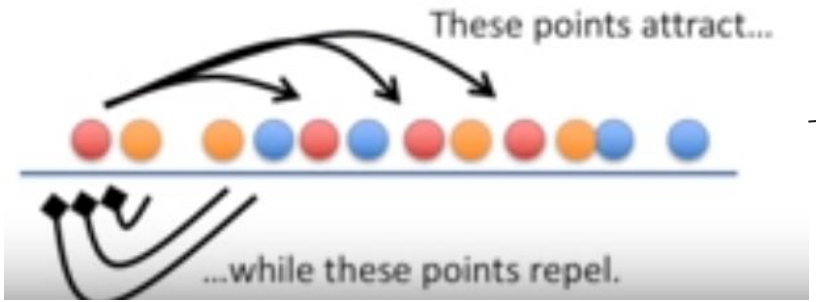


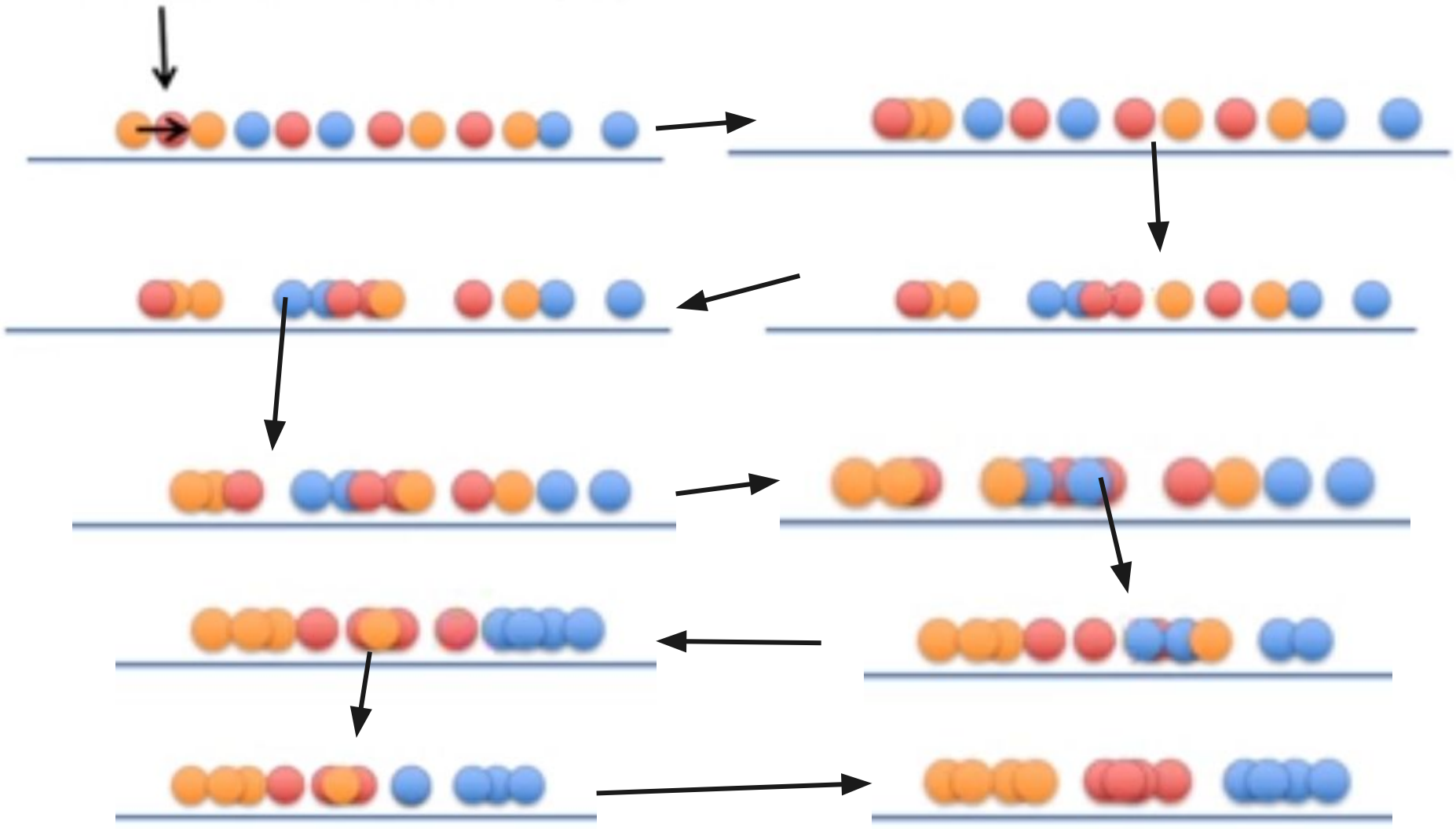
Step 3b: As orange and green points are scattered far away in higher dimension, hence these points will repel the red point in lower dimension.



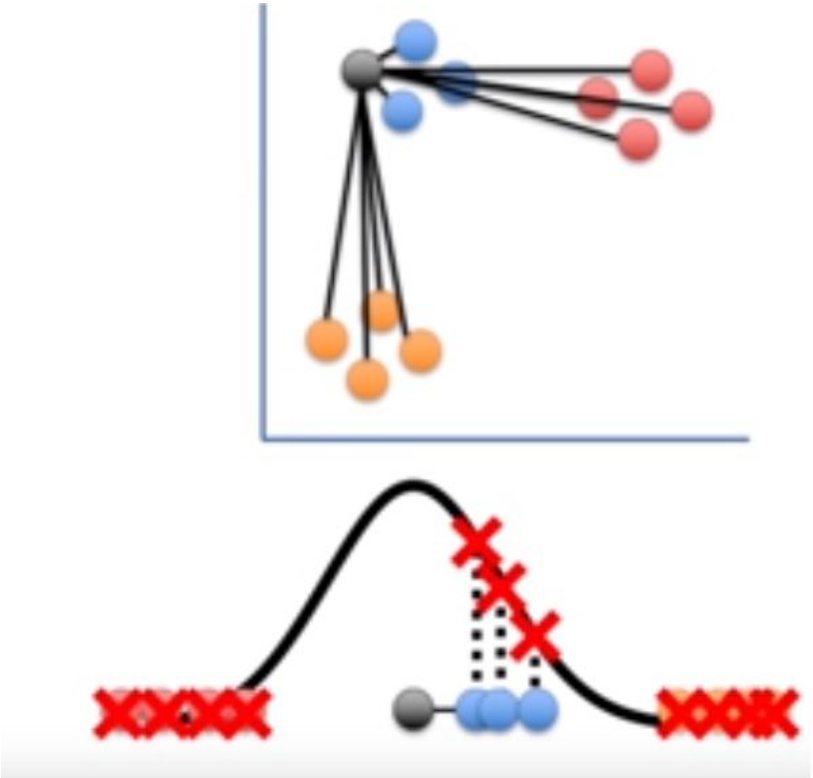


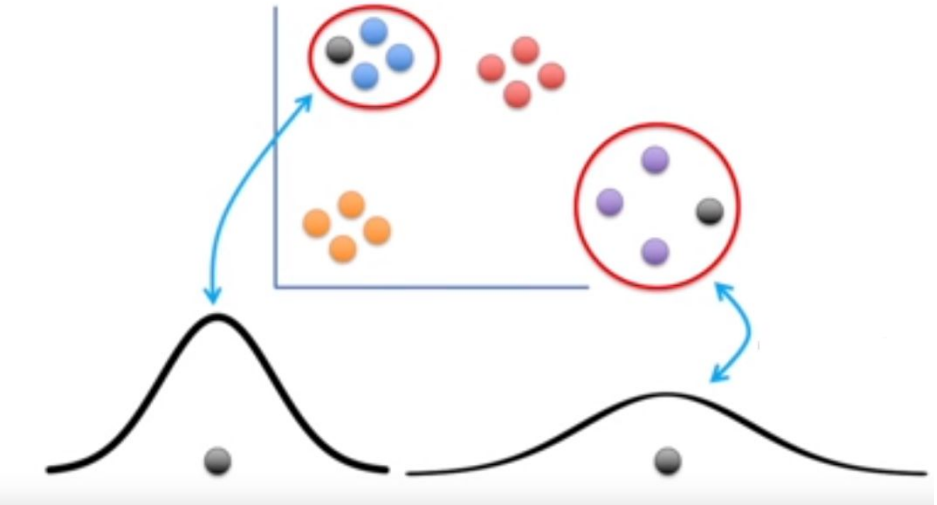
Step 3c: But as attraction is more dominant than repulsion power moves in the direction. This process repeats until convergence (theoretically) or some fixed number of threshold.



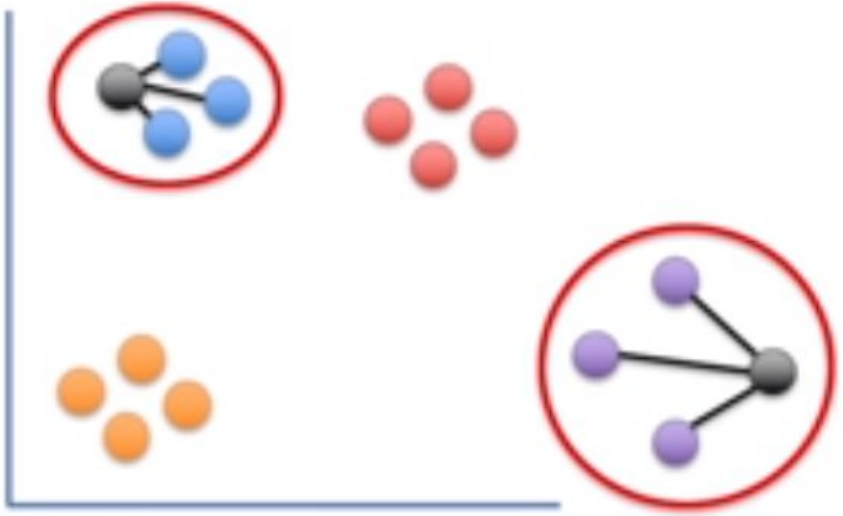


We measure the similarity scores with respect to point of interest using a Gaussian Curve. Similarity score with a point is the height of gaussian curve at the distance away from mean which is equal to distance distance between the point in higher dimension. This similarity score is unscaled. Width of Gaussian curve depends on density of point near the point of interest.

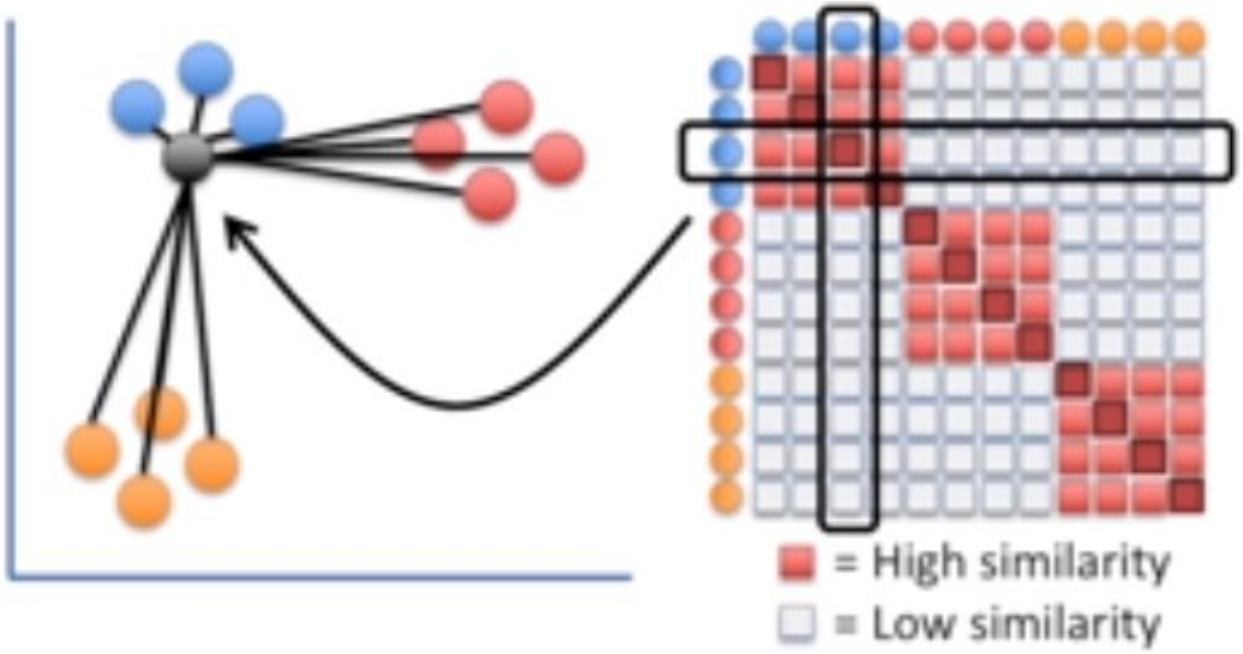




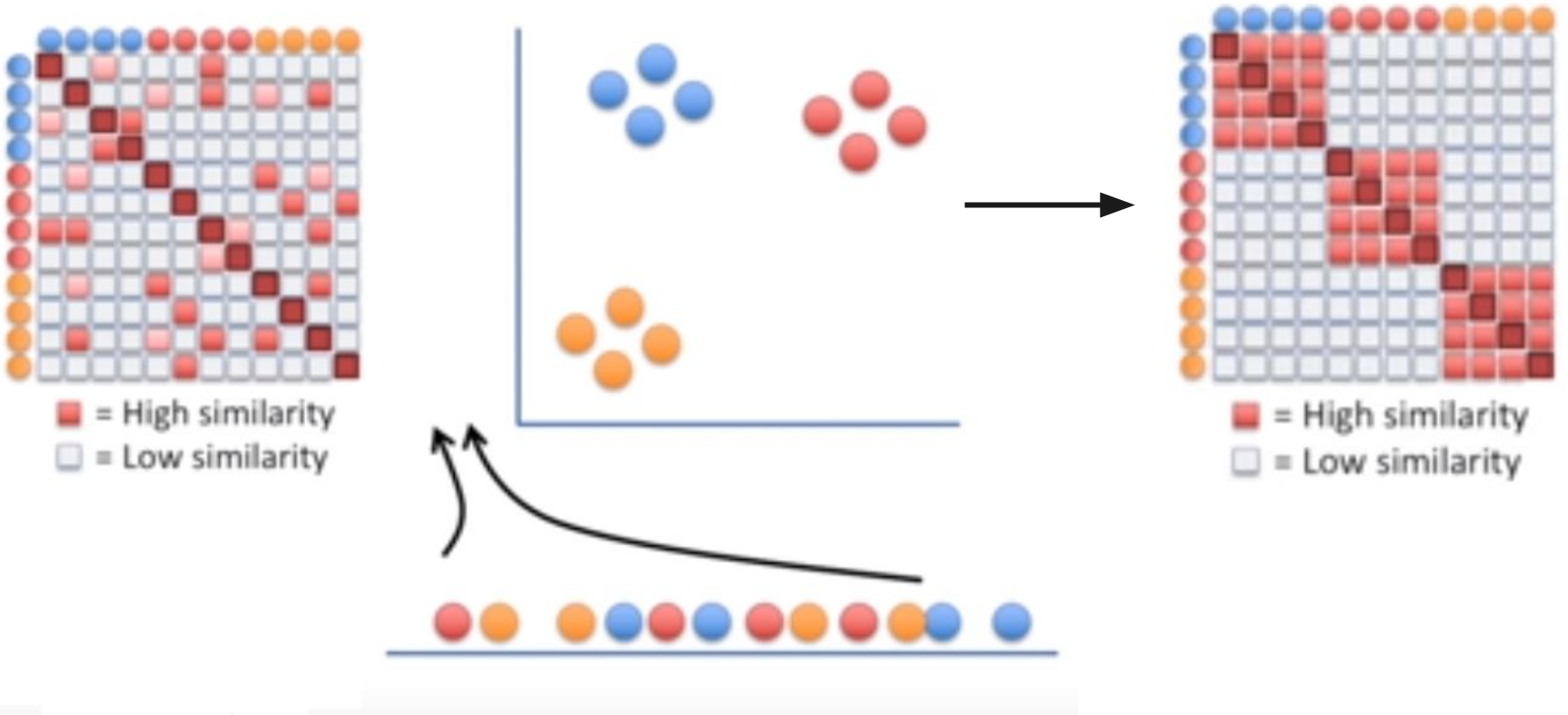
Normalise the similarity score, it will dissolve the effect of relative density of these clusters. [t-SNE has a parameter quals to the expected density and that comes into play].



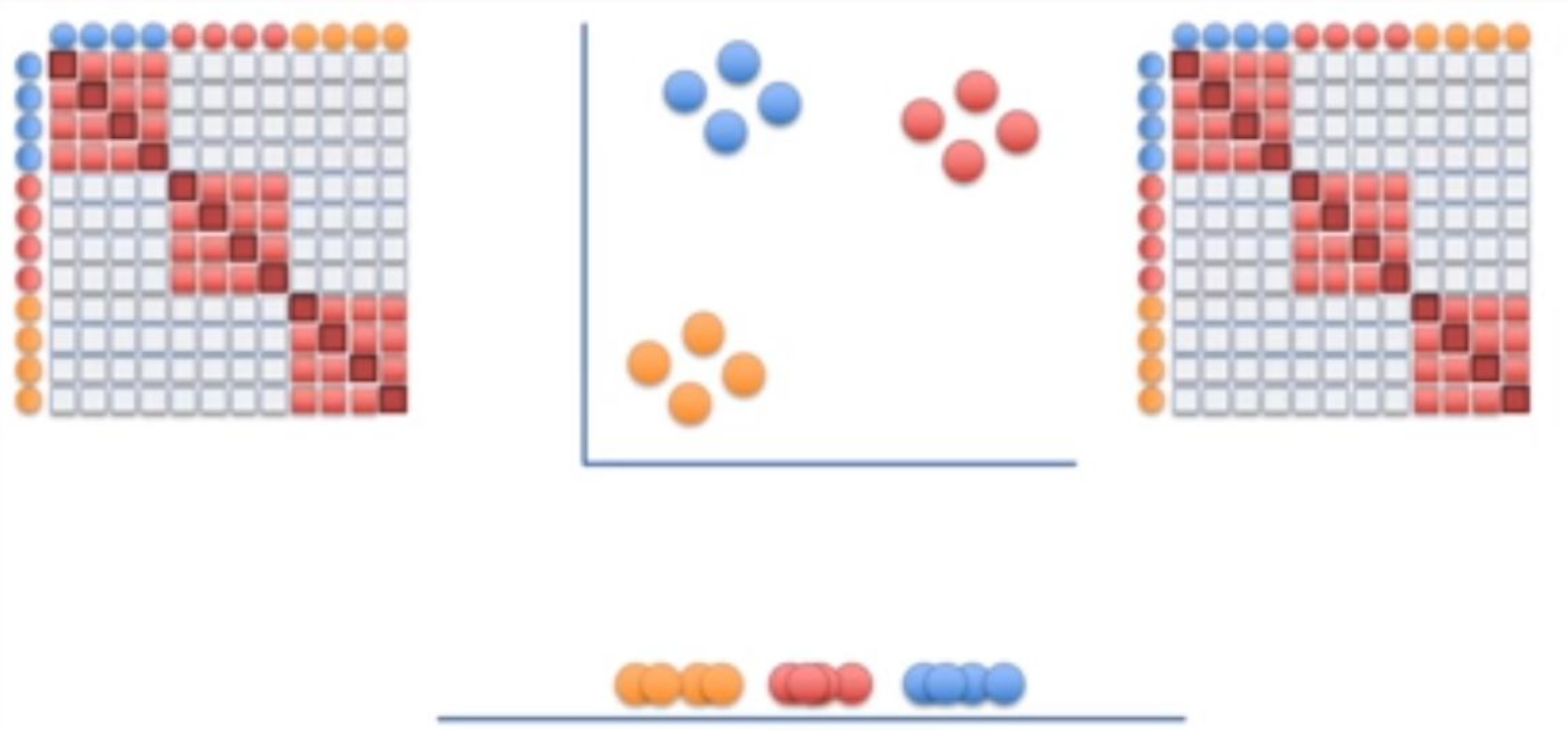
The matrix, ***p*** contains the similarity score between the points. t-SNE implementation assigns 0 similarity to a point to itself. Also <i,j> and <j,i> is average in the matrix.



Now we have to calculate the similarity score in lower dimension say ***q*** using the Student t distribution with one degree of freedom. Then we have to compare this matrix with original matrix, ***p***. To compare these two matrix we'll use KL Divergence.



We have to move until both $\boldsymbol{p}$ and $\boldsymbol{q}$ matrix become similar.



References & Further Reading

- How to Use t-SNE Effectively: https://distill.pub/2016/misread-tsne/
- http://www.cs.columbia.edu/~verma/classes/uml/lec/uml_lec8_tsne.pdf
- Implementations: https://lvdmaaten.github.io/tsne/
- Original Paper: http://www.cs.toronto.edu/~hinton/absps/tsne.pdf
- Pytorch: https://github.com/cemoody/topicsne
- MNIST Sample: https://nbviewer.jupyter.org/github/danielfrg/py_tsne/blob/master/examples/mnist.ipynb
- Why is t-SNE not used as a dimensionality reduction technique for clustering or classification? https://stats.stackexchange.com/questions/340175/why-is-t-sne-not-used-as-a-dimensionality-reduction-technique-for-clustering-or
- Clustering output of t-SNE: https://stats.stackexchange.com/questions/263539/clustering-on-the-output-of-t-sne?noredirect=1&lq=1
- Real time t-SNE with tensorflow.js: https://nicola17.github.io/tfjs-tsne-demo/
- StatQuest: t-SNE, Clearly Explained: https://www.youtube.com/watch?v=NEaUSP4YerM