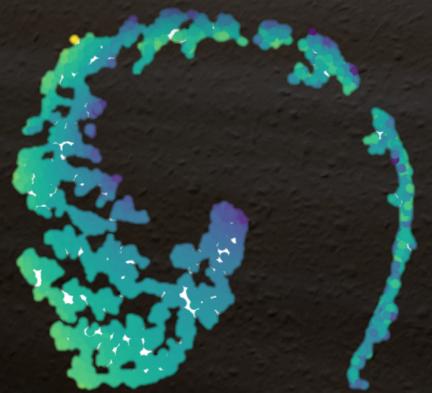
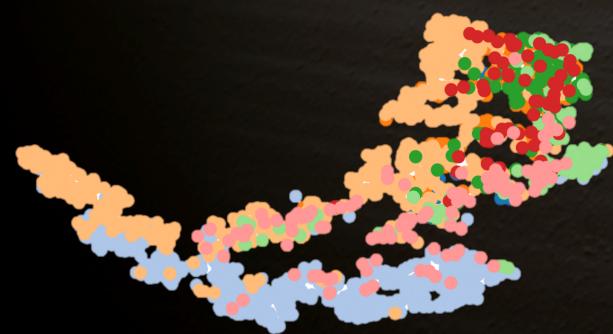
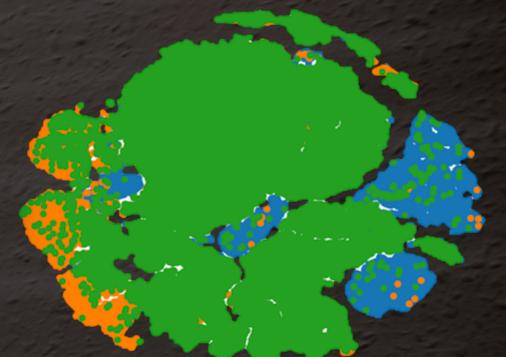
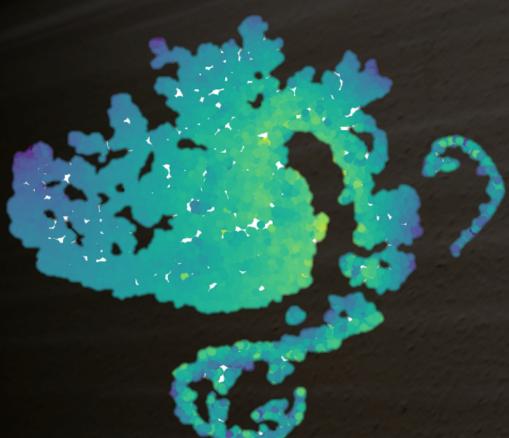


# t-distributed Stochastic Neighbour Embedding

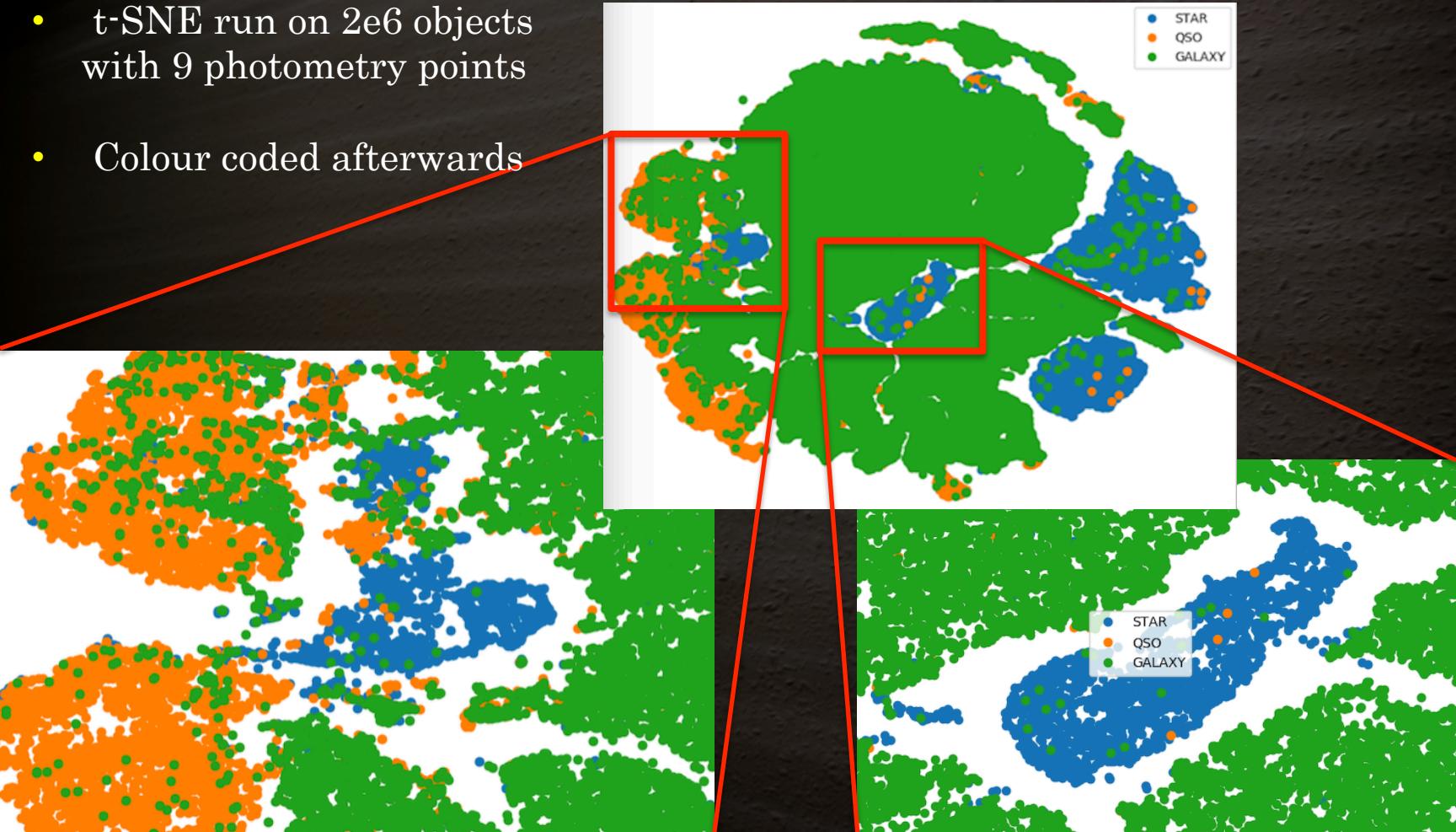


# t-distributed Stochastic Neighbour Embedding

- t-SNE is great for embedding high-dimensional data in 2/3D for visualisation, and discovering clustering in data sets
- It's non linear
- Can be slow on large data sets because it's an iterative algorithm
- Has a cost function that is not convex - different initializations give different results.
- Requires a level of interaction to interpret results

# t-distributed Stochastic Neighbour Embedding

- t-SNE run on 2e6 objects with 9 photometry points
- Colour coded afterwards



# t-distributed Stochastic Neighbour Embedding

Wikipedia says...

## Stage 1:

Construct a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked, whilst dissimilar points have an extremely small probability of being picked.

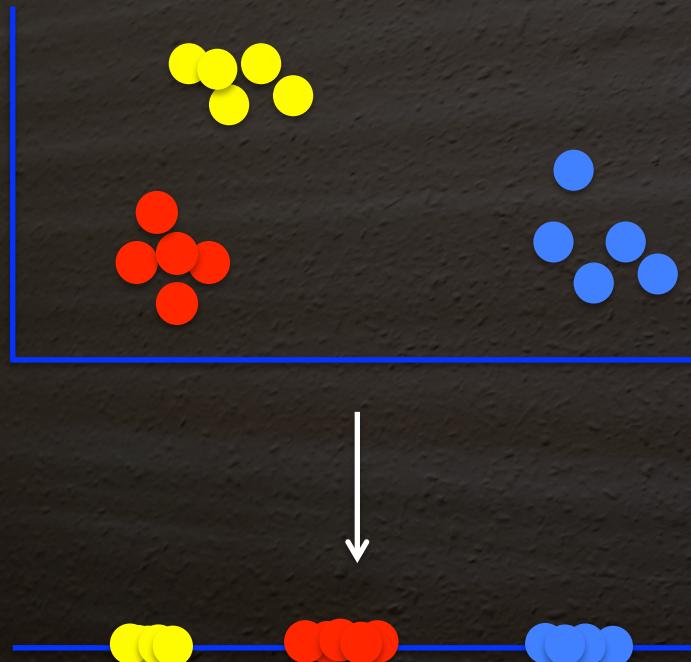
## Stage 2:

Define a similar probability distribution over the points in the 2D map, and minimizes the Kullback–Leibler divergence between the two distributions with respect to the locations of the points in the map.

Breaking this down...

# t-distributed Stochastic Neighbour Embedding

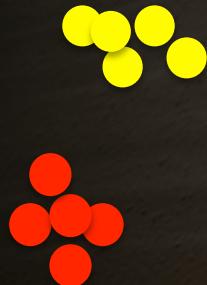
Simple 2D to 1D example



Note that this can't be achieved by projecting onto one axis

# t-distributed Stochastic Neighbour Embedding

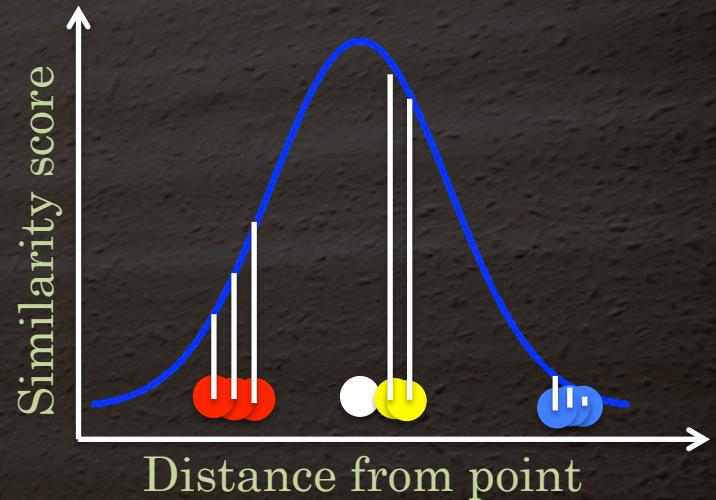
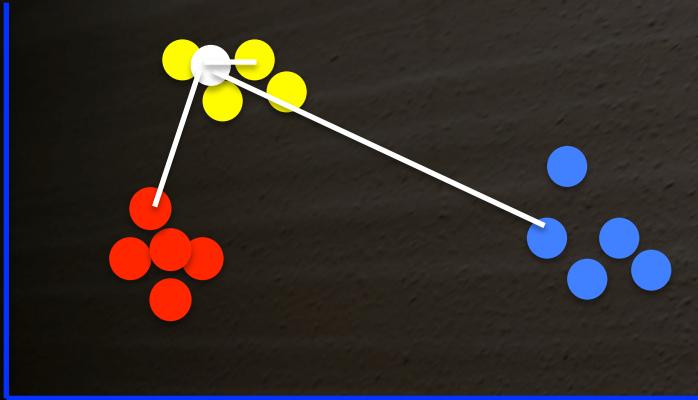
Simple 2D to 1D example



First find out how similar  
the points are in 2D

# t-distributed Stochastic Neighbour Embedding

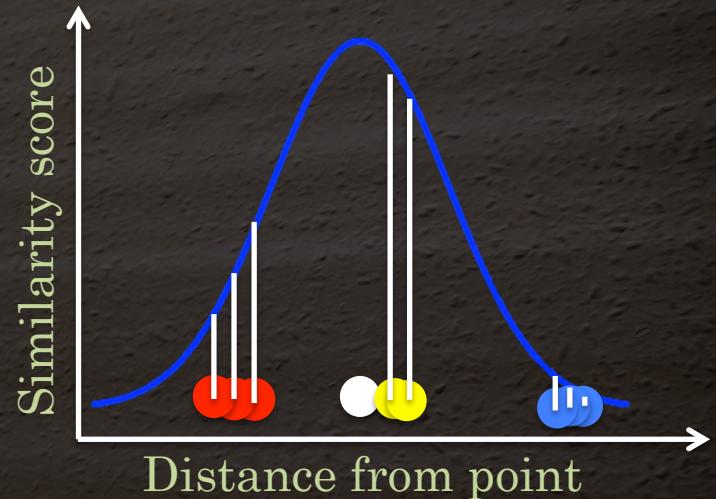
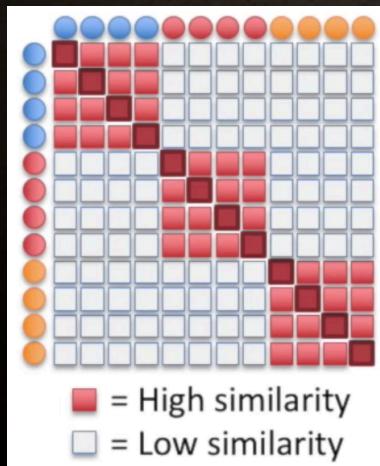
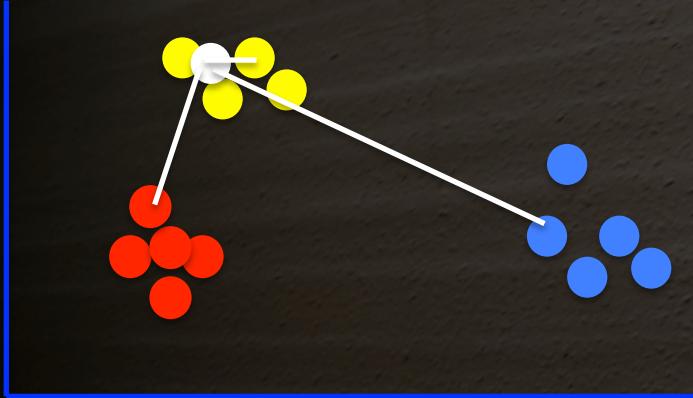
Simple 2D to 1D example



- Centre a Gaussian distribution on each point to obtain similarity scores for each pair of points

# t-distributed Stochastic Neighbour Embedding

Simple 2D to 1D example



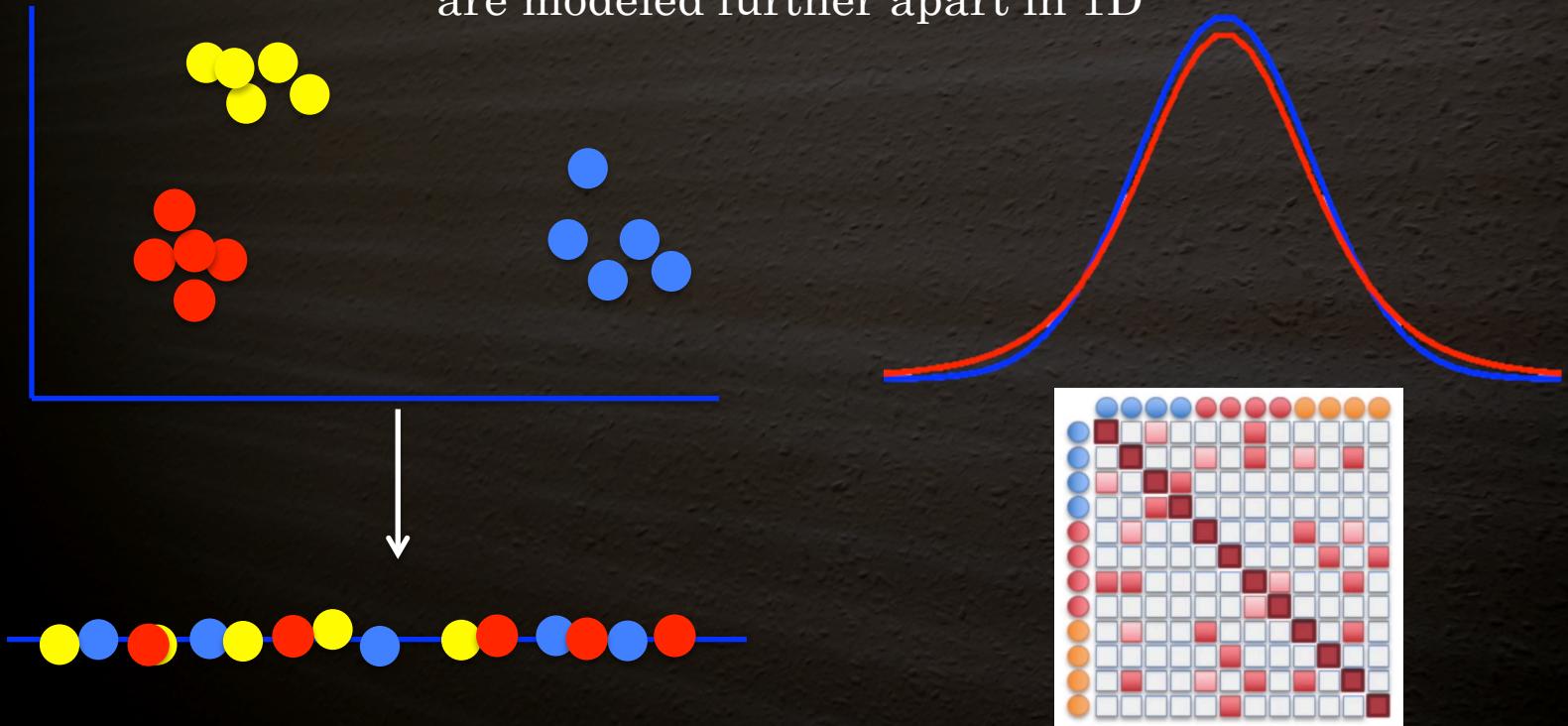
- Centre a Gaussian distribution on each point to obtain similarity scores for each pair of points
  - Obtain a matrix of scores for all points
- Note that a hyper-parameter called perplexity can tune the scaling of dense or sparse clusters

# t-distributed Stochastic Neighbour Embedding

Simple 2D to 1D example

Randomly project data from 2D into 1D, and calculate similarity scores in 1D

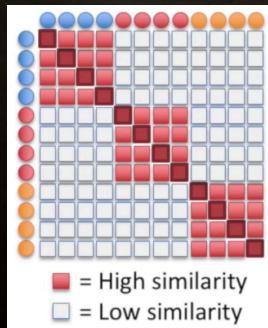
But this time use a **t-distribution**, not a **Gaussian**, so that dissimilar objects are modeled further apart in 1D



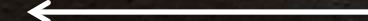
# t-distributed Stochastic Neighbour Embedding

Simple 2D to 1D example

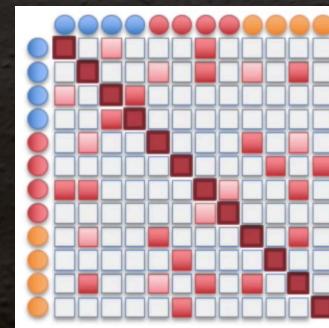
Points are nudged around slowly in 1D space – attracted to similar points, repelled by dissimilar points



Compare to original distribution in 2D



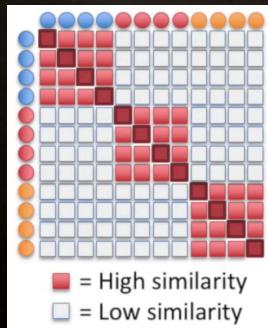
Compare all points, calculate similarity score from t-distribution



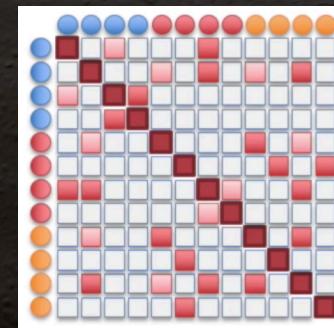
# t-distributed Stochastic Neighbour Embedding

Simple 2D to 1D example

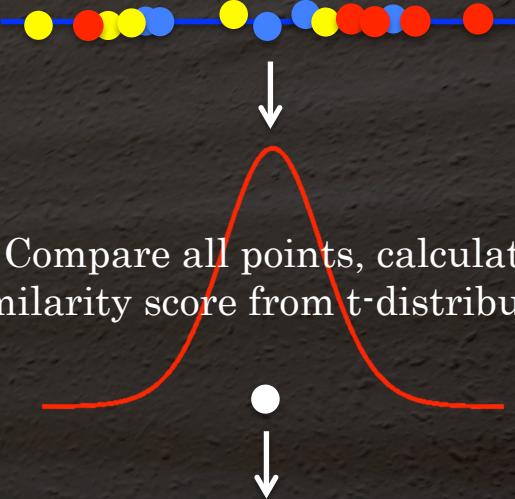
Points are nudged around slowly in 1D space – attracted to similar points, repelled by dissimilar points



Compare to original distribution in 2D



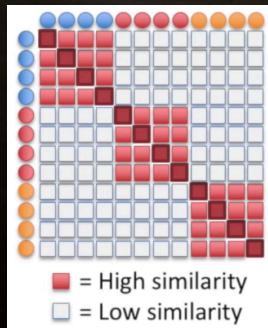
Compare all points, calculate similarity score from t-distribution



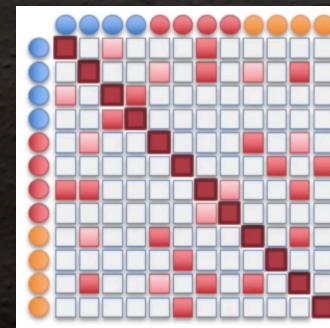
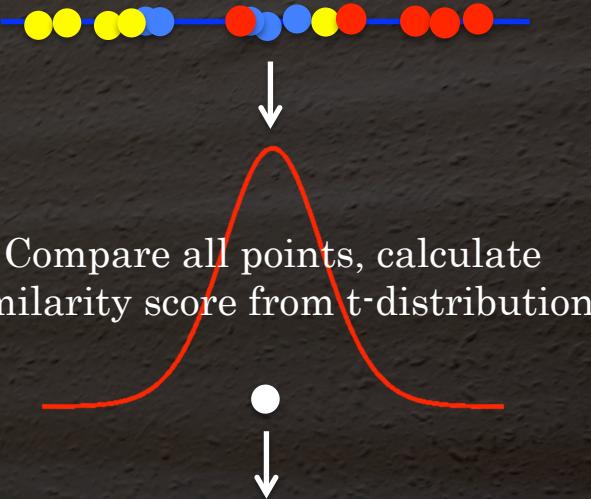
# t-distributed Stochastic Neighbour Embedding

Simple 2D to 1D example

Points are nudged around slowly in 1D space – attracted to similar points, repelled by dissimilar points



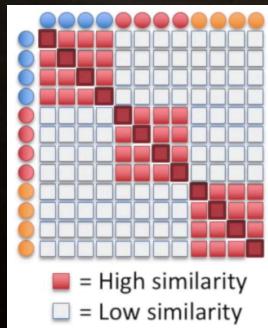
Compare to original distribution in 2D



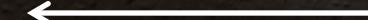
# t-distributed Stochastic Neighbour Embedding

Simple 2D to 1D example

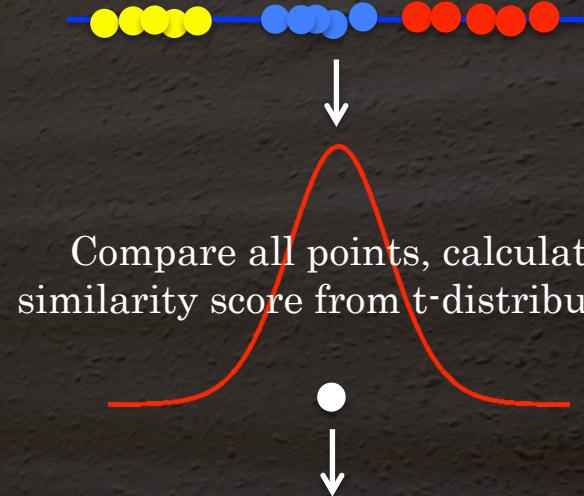
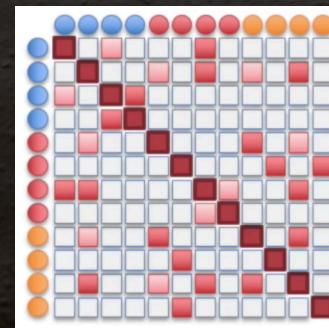
Points are nudged around slowly in 1D space – attracted to similar points, repelled by dissimilar points



Compare to original distribution in 2D



Compare all points, calculate similarity score from t-distribution



# t-distributed Stochastic Neighbour Embedding

Simple 2D to 1D example

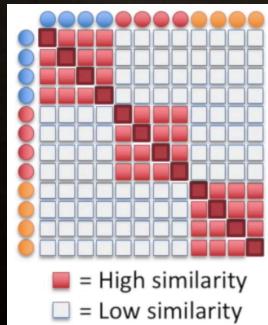
position      gradient decent      ‘momentum term’

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

Points are nudged around slowly in 1D space – attracted to similar points, repelled by dissimilar points

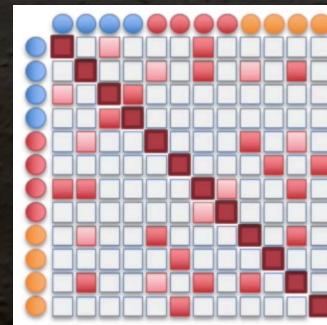


Compare all points, calculate similarity score from t-distribution



Compare to original distribution in 2D

←  
Create some cost function, C



(this plot should have changed every iteration to look similar to the original)

# t-distributed Stochastic Neighbour Embedding Code

```
>>> import numpy as np  
>>> from sklearn.manifold import TSNE  
>>> X = np.array([[0, 0, 0], [0, 1, 1], [1, 0, 1], [1, 1, 1]]) #some 4x3 array  
>>> T = TSNE(n_components=2, perplexity=30, learning_rate=200, n_iter=1000)  
>>> X_embedded=T.fit_transform(X)  
>>> X_embedded.shape  
(4, 2)
```

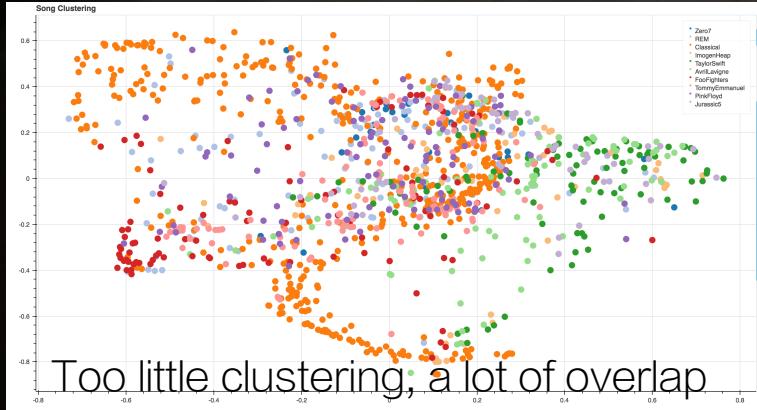
## Hyper-parameters

Iterations: as many as required, larger data sets need more

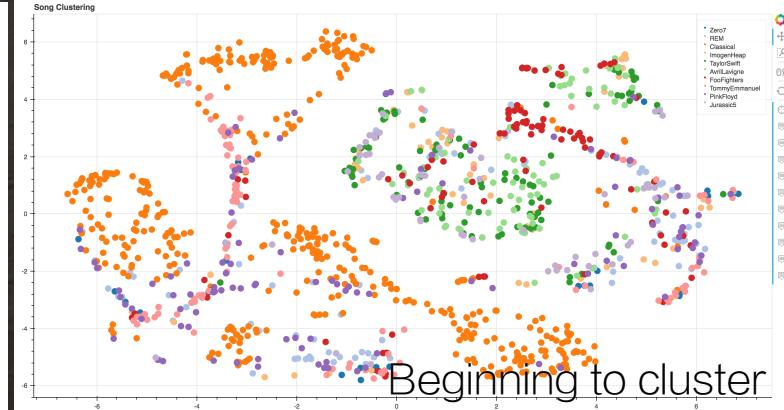
Perplexity: can be interpreted as a smooth measure of the effective number of neighbors per cluster. The performance is fairly robust to changes in the perplexity, and typical values are between 5 and 50

Learning rate: controls how quickly gradient decent takes place. Start low, but can save on iterations if larger. Too large can end up with all data points equally far away from each other ☹

# t-distributed Stochastic Neighbour Embedding

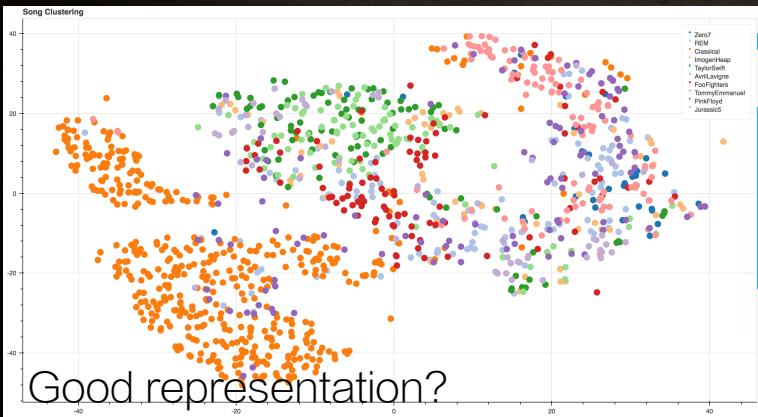


Too little clustering; a lot of overlap

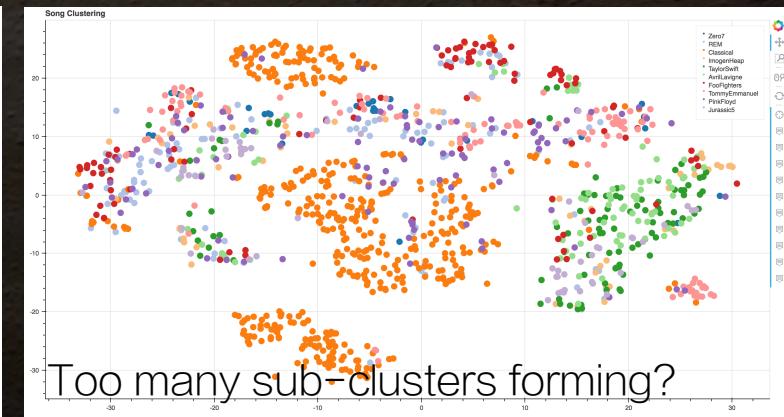


Beginning to cluster

Varying the learning rate

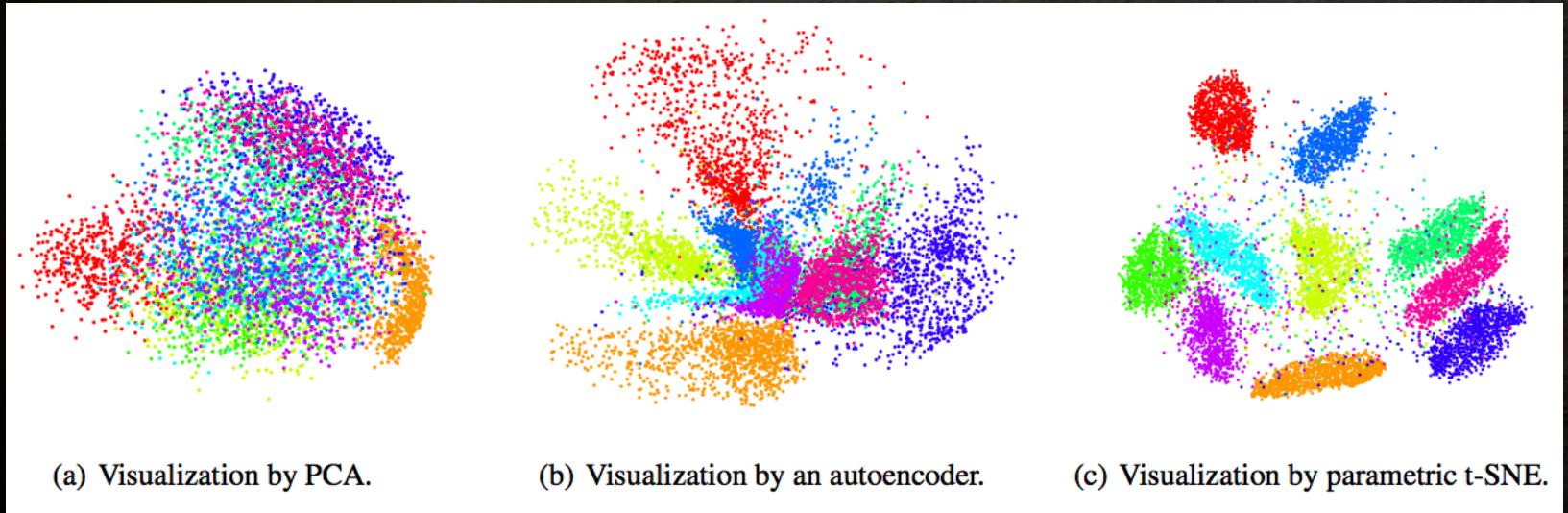


Good representation?



Too many sub-clusters forming?

# t-distributed Stochastic Neighbour Embedding



(a) Visualization by PCA.

(b) Visualization by an autoencoder.

(c) Visualization by parametric t-SNE.

# t-distributed Stochastic Neighbour Embedding

Why use it?

- Extremely effective (non-linear) method in dimensionality reduction and finding clustering patterns
- Visualising the data and probing potential patterns helps give you insight and guide you with further analysis

# t-distributed Stochastic Neighbour Embedding

## References

t-SNE overview paper:

<http://www.cs.toronto.edu/~hinton/absps/tsne.pdf>

A nice explanatory video:

<https://www.youtube.com/watch?v=NEaUSP4YerM>

Parametric embedding (obtaining a model from t-SNE):

[https://lvdmaaten.github.io/publications/papers/AISTATS\\_2009.pdf](https://lvdmaaten.github.io/publications/papers/AISTATS_2009.pdf)

Multicore t-SNE:

<https://github.com/DmitryUlyanov/Multicore-TSNE>