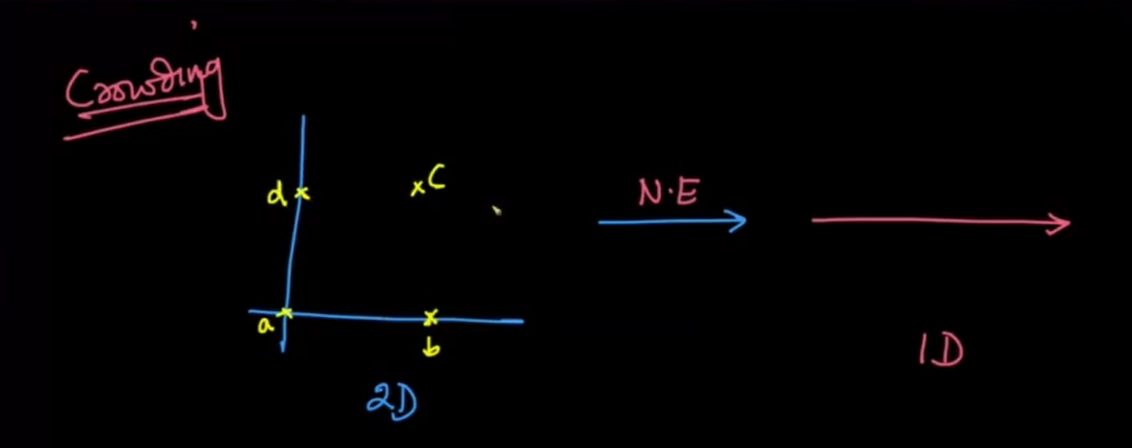
**High Dimension Visualization: t-SNE**

* t-SNE stands for **t-distributed Stochastic Neighborhood Embedding** which was presented by **Laurens van der Maaten** and **Geoffrey Hinton** in 2008.
* One of the limitations of PCA is that it does not preserve the neighborhood when points are projected from a higher dimension to a lower dimension.
* If one wants to project data from a higher dimension to a lower dimension, t-SNE will try to preserve the distances of the points that are close to each other.
* t-SNE tries to create an embedding that preserves the neighborhood using some probabilistic methods.
* Hence, the core idea behind t-SNE is;
  + When we go from 𝑑-dimensions to 𝑑′-dimensions where 𝑑<𝑑′, the core idea behind t-SNE is to preserve the pairwise distance in a neighborhood as best as possible.
* But, there is a problem that t-SNE faces while preserving neighborhood information. It is known as **The Crowding Problem**.

# **Crowding Problem**

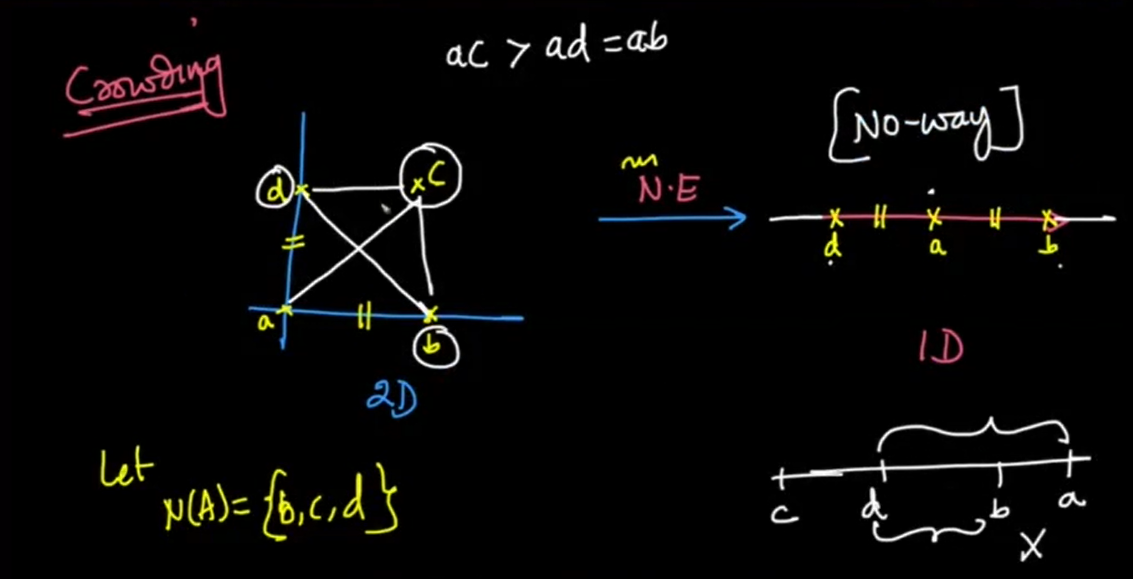
* Suppose we have 2D data and we want to project it in 1D data using any neighborhood embedding method.
* We have four data points in the shape of a square, where 𝑎 is at the origin, 𝑏 is on X-axis, and 𝑑 is on Y-axis as shown in the diagram given below.



* Now, consider a case, when we choose the neighborhood of the point 𝑎 that contains all the other points.

Let's try to project this data into 1D such that the pairwise distance is preserved

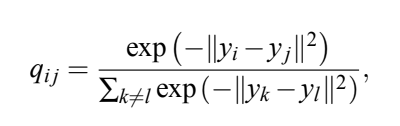
* We place point 𝑎 on a 1D axis, point 𝑏 on the right of point 𝑎, and point 𝑑 on the left of point 𝑎. Here, the distance of both the points 𝑑 and 𝑏 to point 𝑎 is the same.
* Now, if you try to project point 𝑐, it will be exactly projected at the coordinates of point 𝑎. Because, as 𝑎 is equidistant from point 𝑏 and 𝑑, so is the point 𝑐.



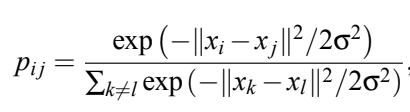
* This was just a simple case we saw for better understanding.
* In real-life data, there will be hundreds, probably thousands of points that will not be able to preserve pairwise distance when projected from a higher dimension to a lower dimension

# **Math for t-SNE**

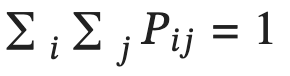
* Our objective is to project datapoints 𝑥𝑖 ∈ 𝑅𝑑 to 𝑦𝑖  using t-SNE, where 𝑦𝑖 ∈ 𝑅2
* In t-SNE, we compute the pairwise similarities as probabilities.
* We compute 𝑃𝑖𝑗 for 𝑑-dimensions and 𝑄𝑖𝑗 for 𝑑′-dimensions where 𝑑>𝑑′
* 𝑃𝑖𝑗 is the probability that the points 𝑥𝑖 and 𝑥𝑗 are neighbors in 𝑑-dimensional space.
* The pairwise similarities in the low-dimensional map 𝑄𝑖𝑗 are given by:



* The pairwise similarities in the high-dimensional space 𝑃𝑖𝑗 is:



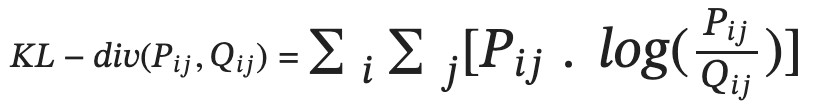
* As you can see in the equation above, the numerator term in 𝑃𝑖𝑗 is nothing but a sort of normal distribution with a variance of σ.
  + The term |𝑥𝑖−𝑥𝑗|2 computes the euclidean distance between 𝑥𝑖 and 𝑥𝑗.
  + As 𝑥𝑖 and 𝑥𝑗 move farther and farther away, we give the lower probability that 𝑥𝑖 and 𝑥𝑗 are neighbors
* Now, as we are computing probabilities, probabilities across all points should be equal to 1
* So, the denominator terms is just a normalization factor to make sure that sum of all the probabilities is equal to 1



* This technique is known as SNE.
* Now, in 𝑑′-dimensions space, every 𝑥𝑖 and 𝑥𝑗 would have corresponding 𝑦𝑖 and 𝑦𝑗.
* So, again we define 𝑄𝑖𝑗 with the same formulation as 𝑃𝑖𝑗
* Hence, if 𝑥𝑖 and 𝑥𝑗 are similar, then 𝑃𝑖𝑗 would be higher and we want our 𝑦𝑖 and 𝑦𝑗 such a representation such that 𝑄𝑖𝑗 is also high.
* Because of the crowding problem, we can never perfectly preserve the distance.
* So, in t-SNE, we try to preserve the probabilities when going from high dimension to low dimension space
* We compare probabilities 𝑃𝑖𝑗 and𝑄𝑖𝑗  with something known as KL-Divergence.

# **KL-Divergence**

* It measures the dissimilarity between the distributions.
* So, the KL-Divergence between two distributions 𝑃 and 𝑄 can be written as:



* KL-divergence is also known as **relative entropy.**

**Interpreting KL-Divergence**

* If 𝑃𝑖𝑗 and𝑄𝑖𝑗 are the same, then KL-divergence will be equal to 0.
* If 𝑃𝑖𝑗 is very small and, 𝑃𝑖𝑗 and𝑄𝑖𝑗 are the same, then KL-divergence will have a small value.
  + Think of 𝑃𝑖𝑗 working as a weightage, because if 𝑃𝑖𝑗 is small we don't really care as points 𝑥𝑖 and 𝑥𝑗 will be far away from each other in d-dimension space.
* So, now our optimization problem would be to find all the 𝑦𝑖s that minimize KL-divergence(P, Q)
* Lastly, since KL-divergence is a measure of dissimilarity, it is always greater than or equal to 0.

# **‘t’ in t-SNE**

* For computing 𝑃𝑖𝑗, we used gaussian like function.
* But, it was found that if we compute 𝑃𝑖𝑗 using t-distribution with 1 degree of freedom, the results were better.
* t-distributions with 𝑑𝑜𝑓=1 have longer tails than gaussian distributions
* Gaussian distribution falls exponentially while t-distributions sort of inversely
* So, for our 𝑄𝑖𝑗s, if we start using t-distribution, two points can go farther away and still get pairwise distance preserved of sort
* Meaning, in t-SNE, we use Gaussian distribution for 𝑃𝑖𝑗s and t-distribution for 𝑄𝑖𝑗s because of which if two points are far away in lower dimensional space, the probabilities will still remain the same
* Now, If we increase 𝑑𝑜𝑓 and keep increasing, it will behave like a gaussian distribution which will face the problem of crowding
* At, 𝑑𝑜𝑓=∞, it behaves very similar to a Gaussian distribution
* t-distribution with 𝑑𝑜𝑓=1 is also known as Cauchy Distribution

# **Perplexity**

* Perplexity is one of the most parameters that you might want to configure when using t-SNE.
* Perplexity can be interpreted as the effective number of neighbors whose distance we want to preserve
* Typically, we keep the value between [5,50]
* The optimization of t-SNE is very time taking as there are no single optima.
* Also, if you add a bunch of newer data points to the dataset, you won't get projections into lower dimensional space automatically.
* You would have to fit the t-SNE model again on the whole dataset again.

**Use this blog to play around with t-SNE on different data distributions:** <https://distill.pub/2016/misread-tsne/>