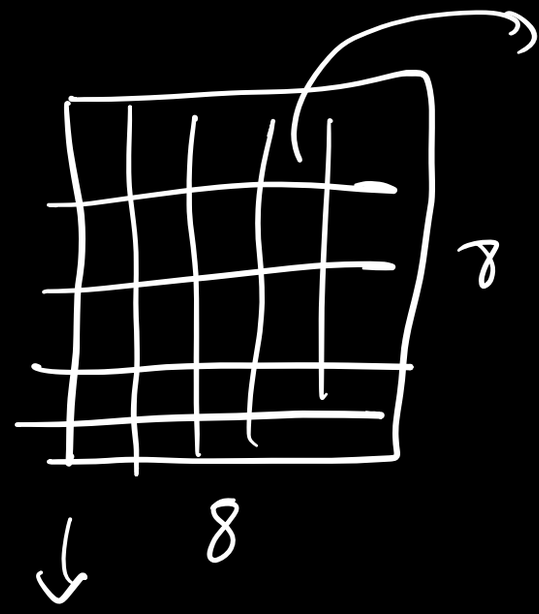
T-SNE

[Hov-3]

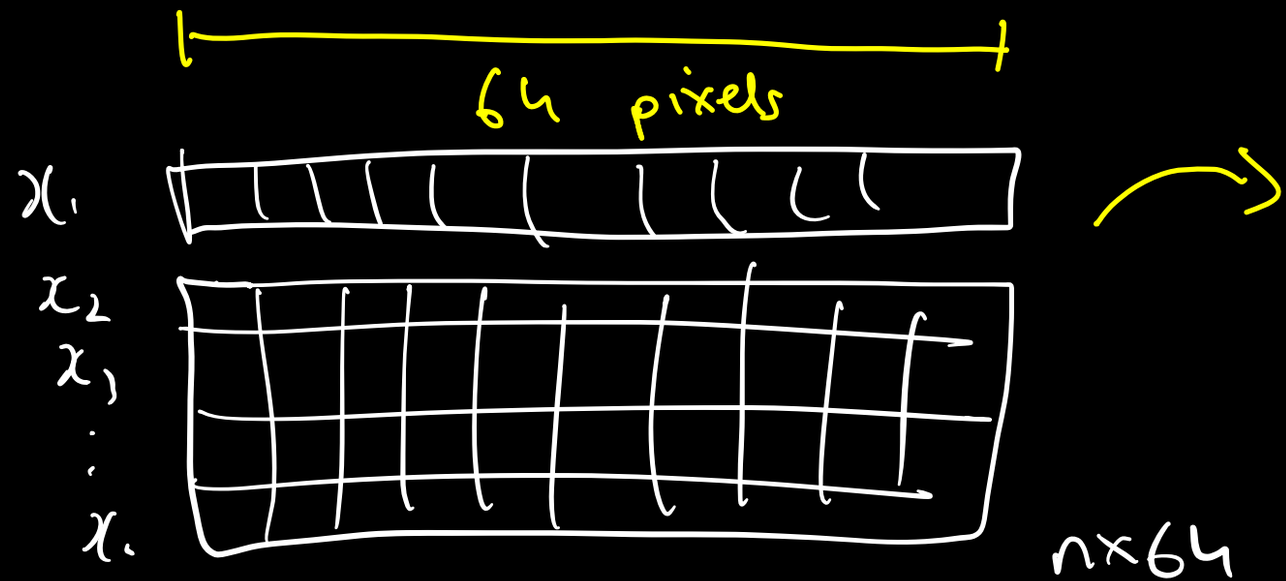
→ TSNE - in depth

Recap

image →



Pixels ↳ 0-1



PCA

↳ 64 ⟶ 2 20% Variance preserved

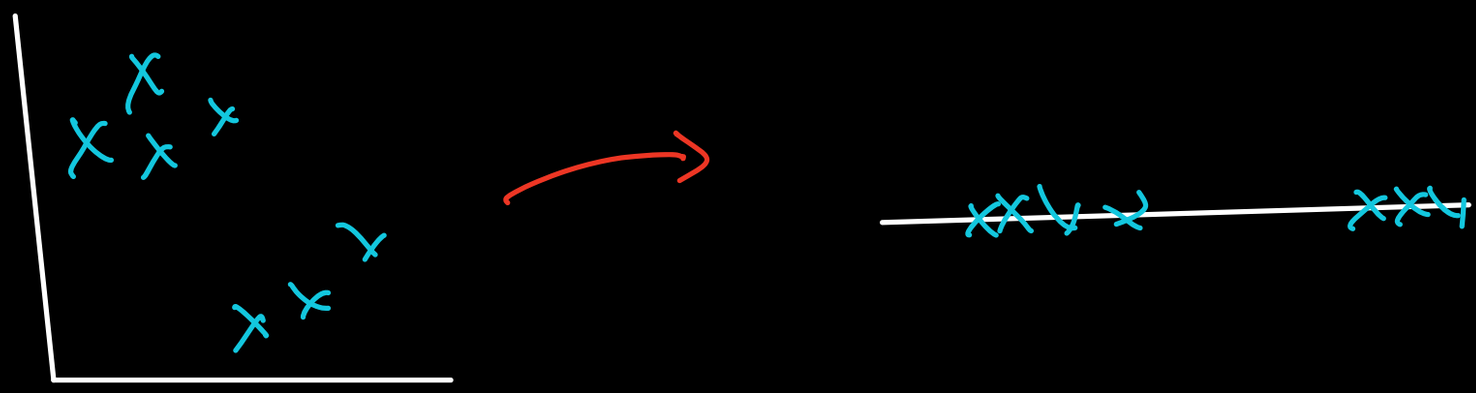
64 ⟶ 32 90%

2/64 = 3% ⇒ 97% compression → 80% loss

32/64 = 50% → 50% " → 10% loss

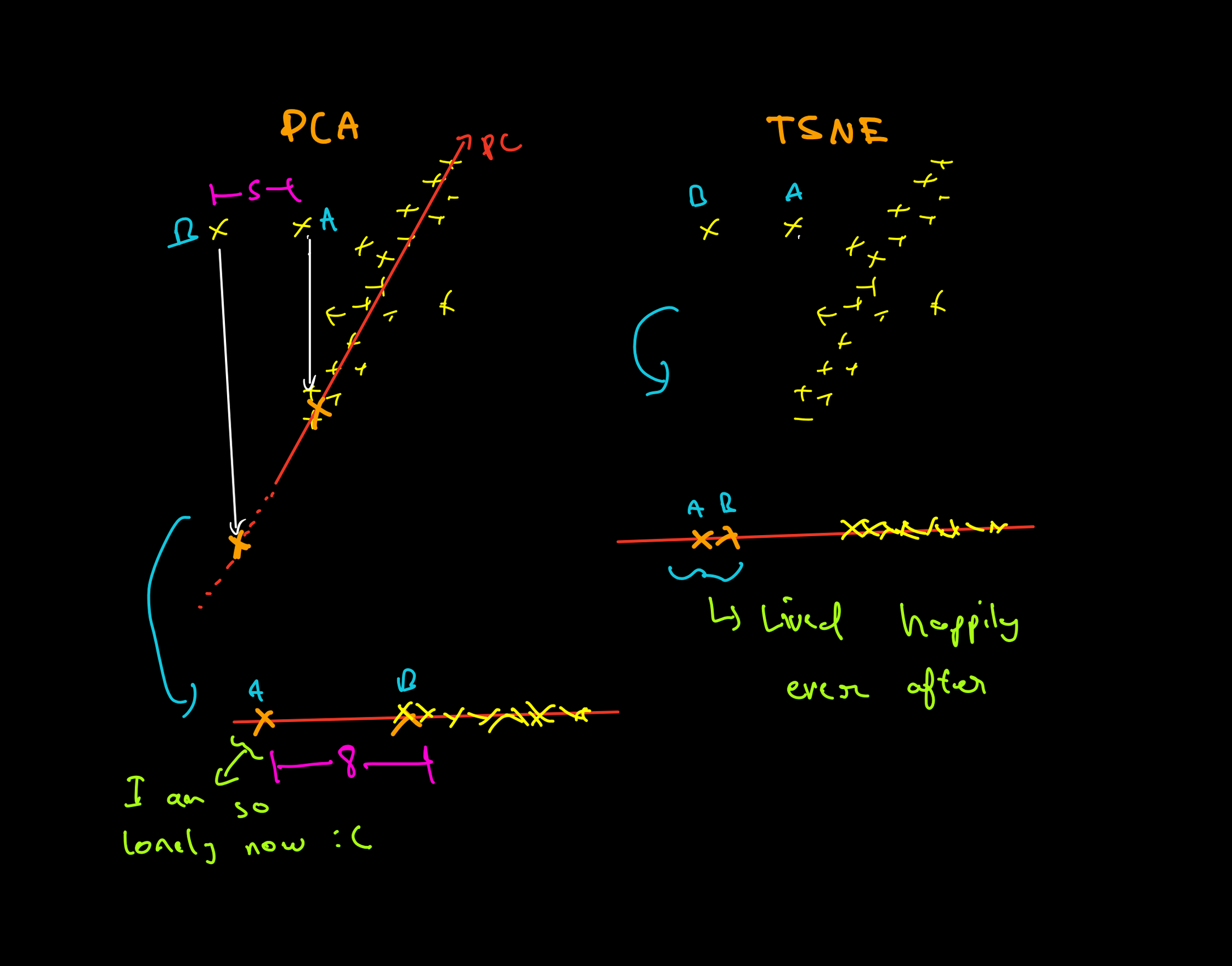
Tsne

T – stochastic Neighbourhood Embedding



Preserve relative distances!

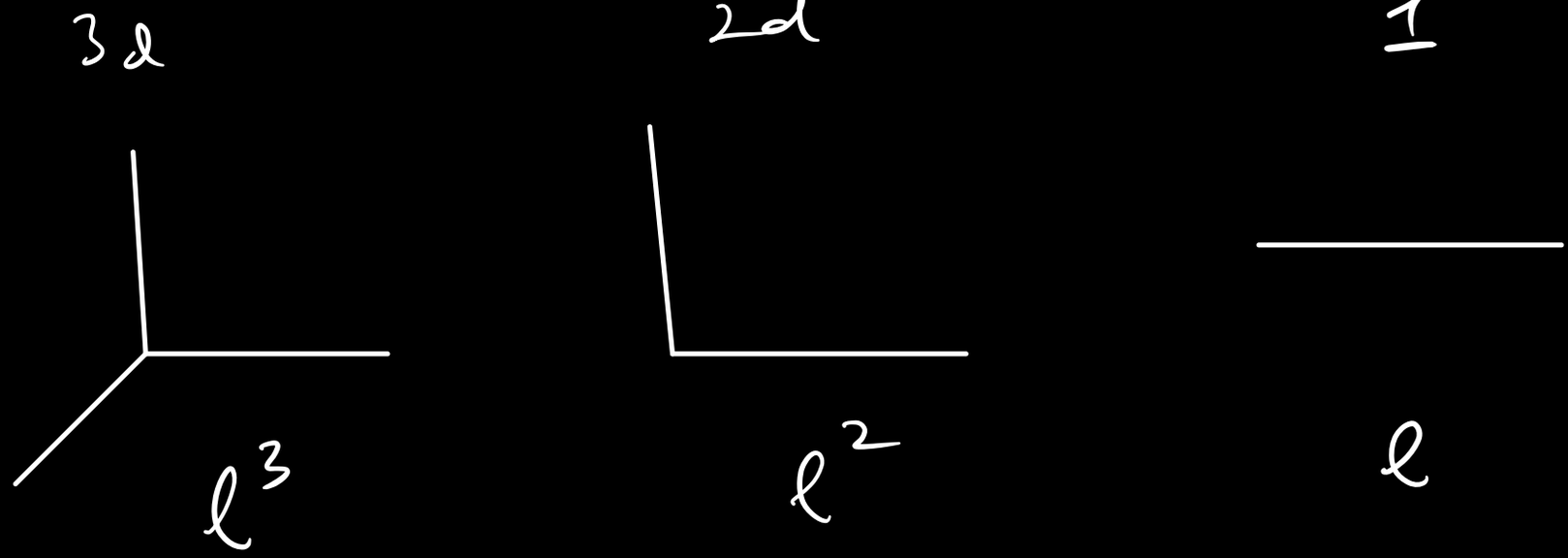
if you are my 3rd closest neighbour in d-dim, try to be my 3rd closest n. in d'-dim


PCA
PC
S
B
A
A
B
I am so lonely now :(
TSNE
B
A
A
B
They lived happily ever after

Quiz: Can we compare distance blw 2 points in 5d to distance in 3d?

a) Yes b) No ✓

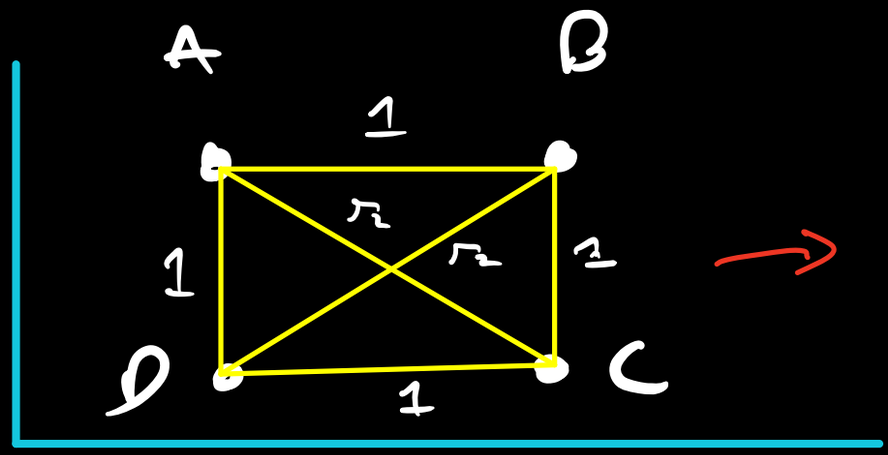
Note:



→ There is more space in high dim

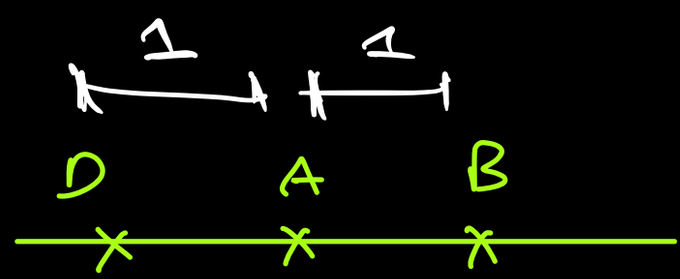
↳ hence longer distances.

⇒ Possible ?



d - dim = 2

C ?



d' - dim = 1

Wherever you place 'C', it will violate atleast one relationship !

So now, how do we approach this idea?

→ Probability

↳ How can you define probability that x_i is neighbour of x_j

→ $\propto$ 1/distance?

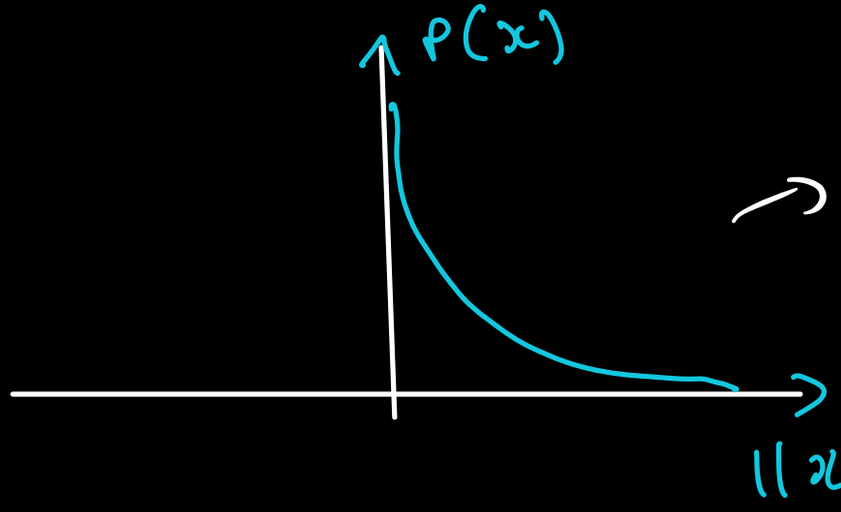
$$P_{ij} \propto \frac{1}{dist(x_i, x_j)} = \frac{1}{\| x_i - x_j \|}$$

(norm of $x_i - x_j$)

$$= \| x_i - x_j \|^{-1} \Rightarrow dist^{-1}$$

↓

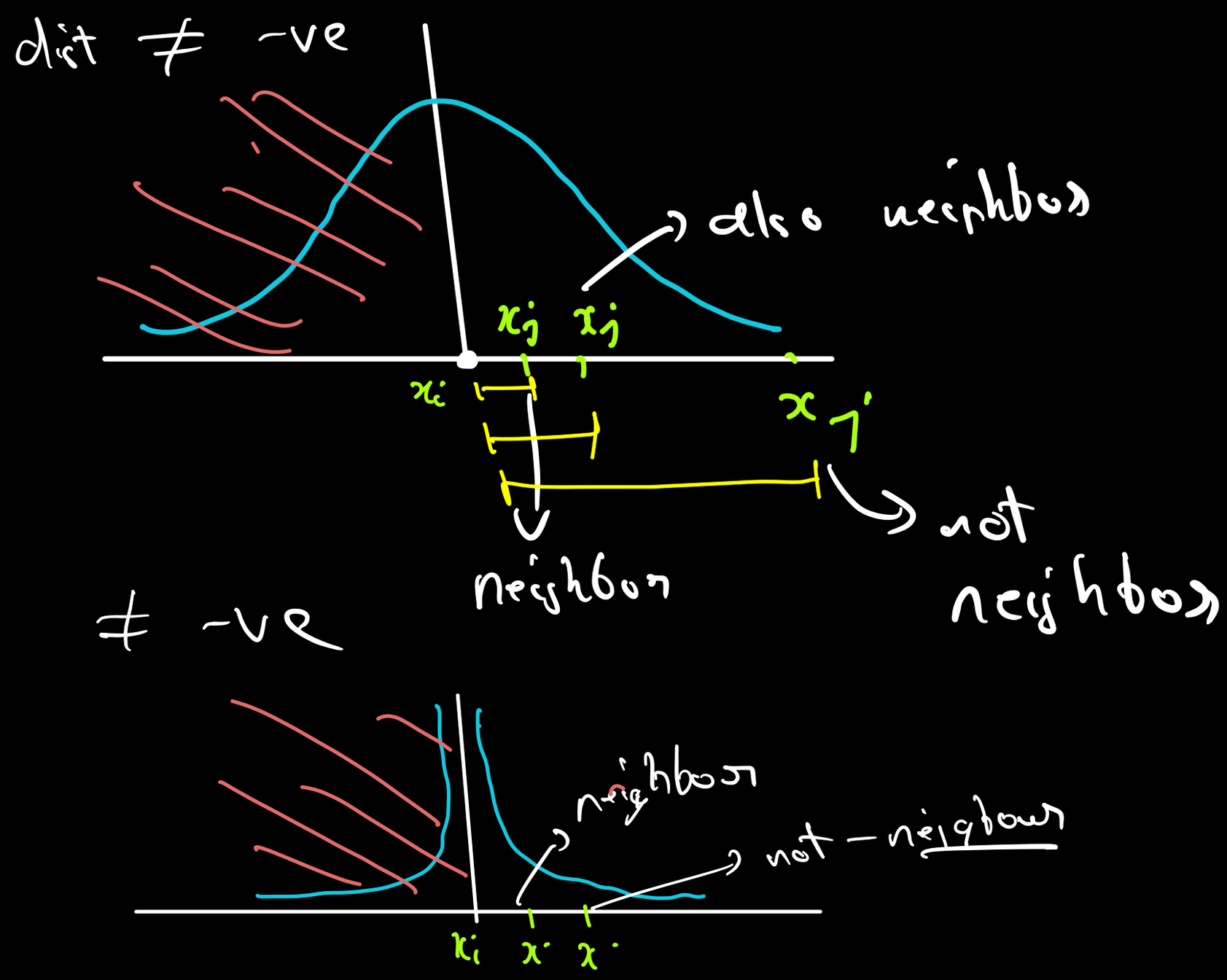
how will the plot look?

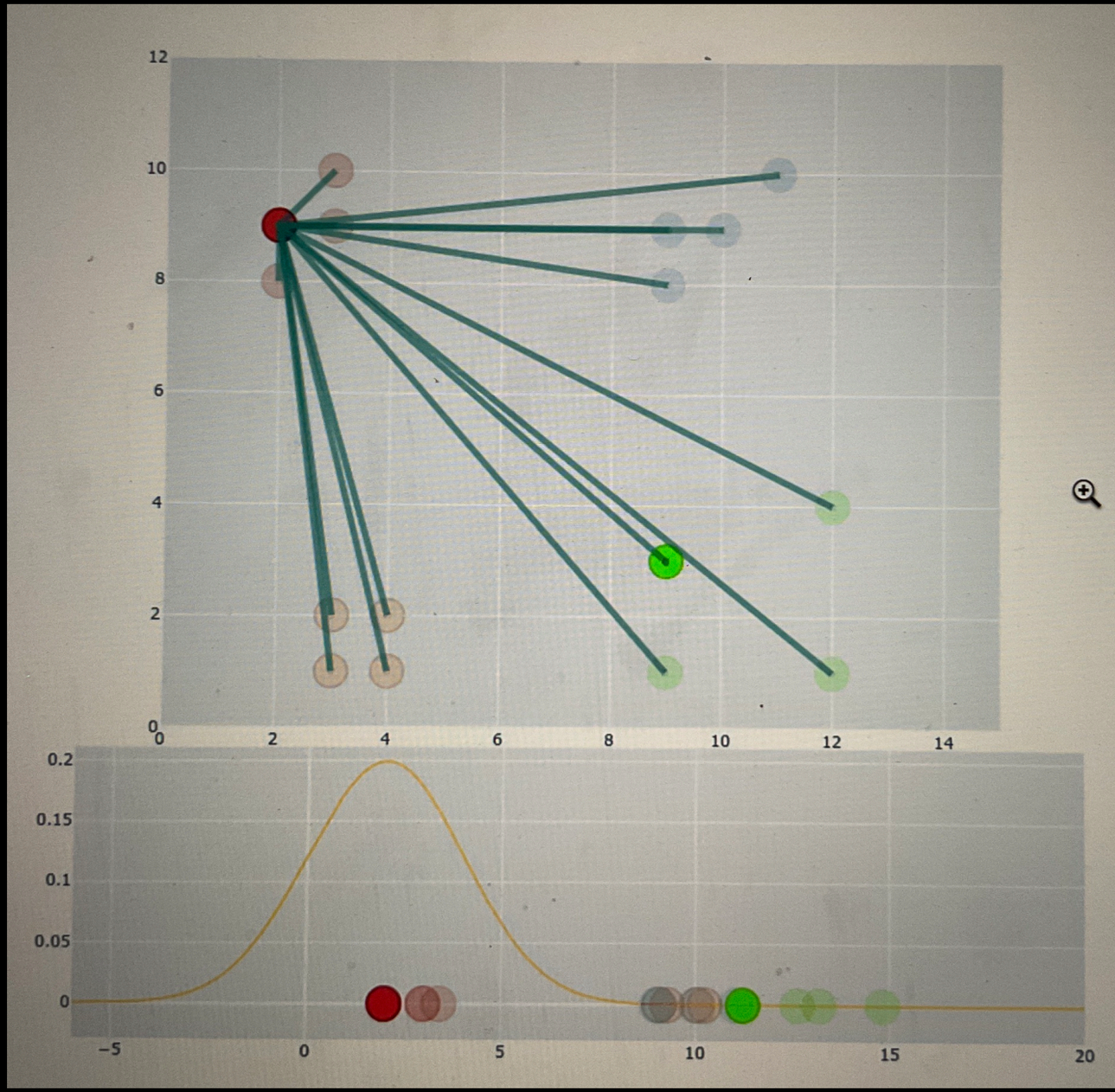


- infinite when dist = 0
- Prob reduces too fast !!
- depends on absolute value of dist

Let's take inspiration from gaussian distributions for a better pdf!

$$N(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$


dist ≠ -ve
also neighbor
x_j x_j
x_i
$x_{j'}$
neighbor
not neighbor
≠ -ve
neighbor
not-neighbour
x_i x' x'


12
10
8
6
4
2
0
0
2
4
6
8
10
12
14
0.2
0.15
0.1
0.05
0
−5
0
5
10
15
20

So let's define probability as:

$$P_{ij} = \frac{e^{-\|x_i - x_j\|^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|^2 / 2\sigma_i^2}}$$

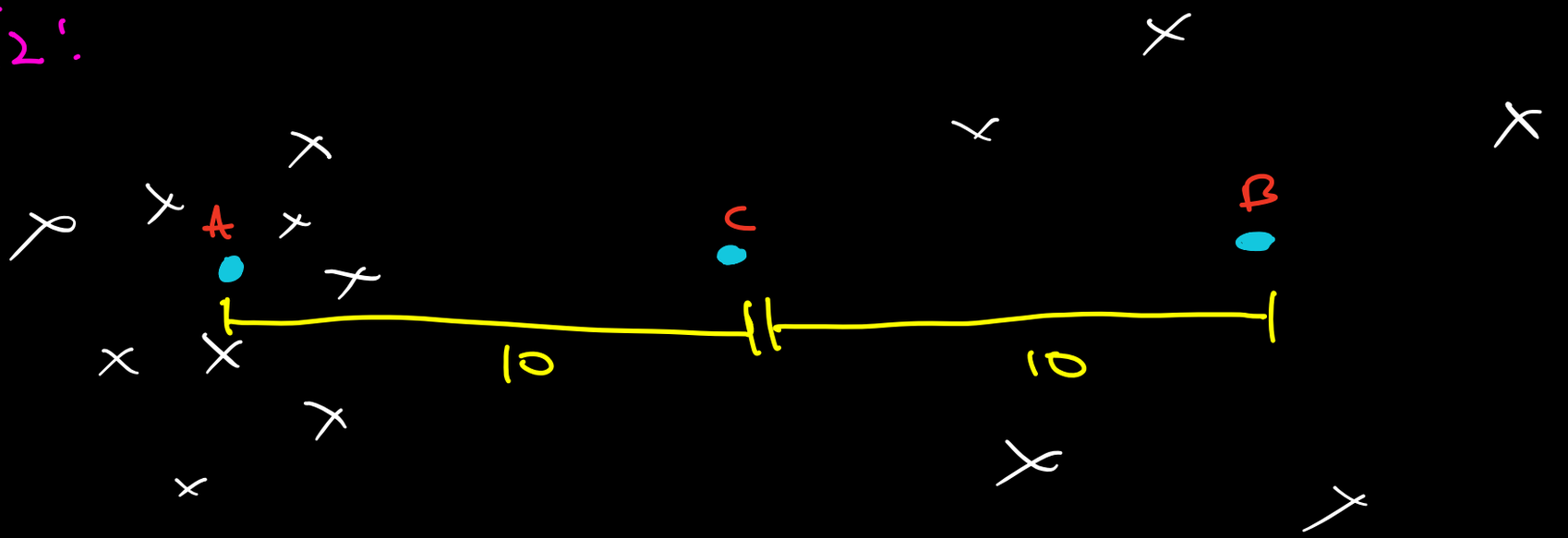
→ sort of gaussian

↓ denominator to make $\sum_i \sum_j P_{ij} =$

→ Paper

We may also write this as:

Quiz:

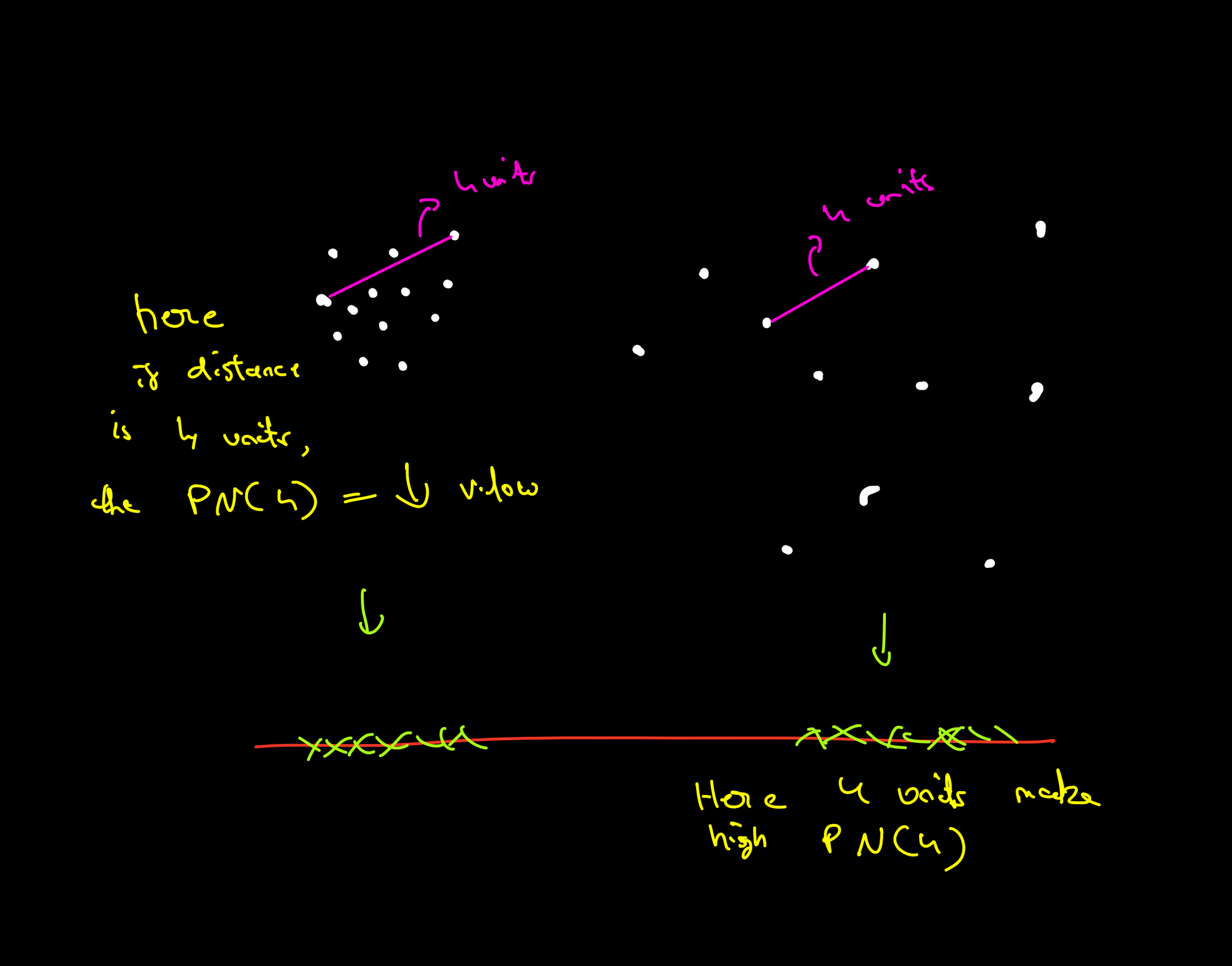


a) $P(A,B) = P(B,C)$

b) $P(A,B) > P(B,C)$

c) $P(A,B) < P(B,C)$ ✓

Since 'A' has more local density, C get's lower prob of being its neighbour.


h unit
h unit
here
if distance
is h unit,
the PN(h) = ↓ value
↓
↓
Here h units make
high PN(h)

dist of x_i to x_j

$$P_{ij} = \frac{C^{-d_{ij}^2 / 2\sigma_i^2}}{\sum \text{ sum of all } P_{ij}}$$

σ_i^2 → estimator of local density

- The std w.r.t each x_i creates a scaled distance effect in the neighbourhood of x_i [not absolute]
- denominator ensures $\sum P_{ij} = 1$

Pg: 2581

2. Stochastic Neighbor Embedding

Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities.[1] The similarity of datapoint x_j to datapoint x_i is the conditional probability, $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i. For nearby datapoints, $p_{j|i}$ is relatively high, whereas for widely separated datapoints, $p_{j|i}$ will be almost infinitesimal (for reasonable values of the variance of the Gaussian, σ_i). Mathematically, the conditional probability $p_{j|i}$ is given by

Pg: 2583

The remaining parameter to be selected is the variance σ_i of the Gaussian that is centered over each high-dimensional datapoint, x_i. It is not likely that there is a single value of σ_i that is optimal for all datapoints in the data set because the density of the data is likely to vary. In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions. Any particular value of σ_i induces a probability distribution, P_i, over all of the other datapoints. This distribution has an entropy which increases as σ_i increases. SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity that is specified by the user.[3] The perplexity is defined as

Perplexity

↳ how many neighbouring points to use to define neighbourhood (i.e compute σ_i)

→ V. imp will cover in detail.

produces a P_i with a fixed perplexity that is specified by the user.[3] The perplexity is defined as

$$Perp(P_i) = 2^{H(P_i)},$$

↳ Hyper param

where $H(P_i)$ is the Shannon entropy of P_i measured in bits

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

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

The minimization of the cost function in Equation 2 is performed using a gradient descent method. The gradient has a surprisingly simple form

Perplexity ∝ std(σ_i)

Similarly, in the d' dim,

$$Q_{ij} = \frac{e^{-\|y_i - y_j\|^2}}{\sum}$$

(Here there is no σ, i.e $2\sigma^2 = 1$)

$$\sigma = \frac{1}{\sqrt{2}}$$

where

$x_i \longrightarrow y_i$

d-dim $\quad$ d'-dim

Pg: 2581
Footnote - 2

Now, to get a useful d' transformation we need

$$P_{ij} \approx Q_{ij}$$

Probability of x_i and x_j being neighbours must be as close as possible to prob of y_i and y_j being neighbours

Q: What do we usually do next?

→ Optimisation

Q: What do we need?

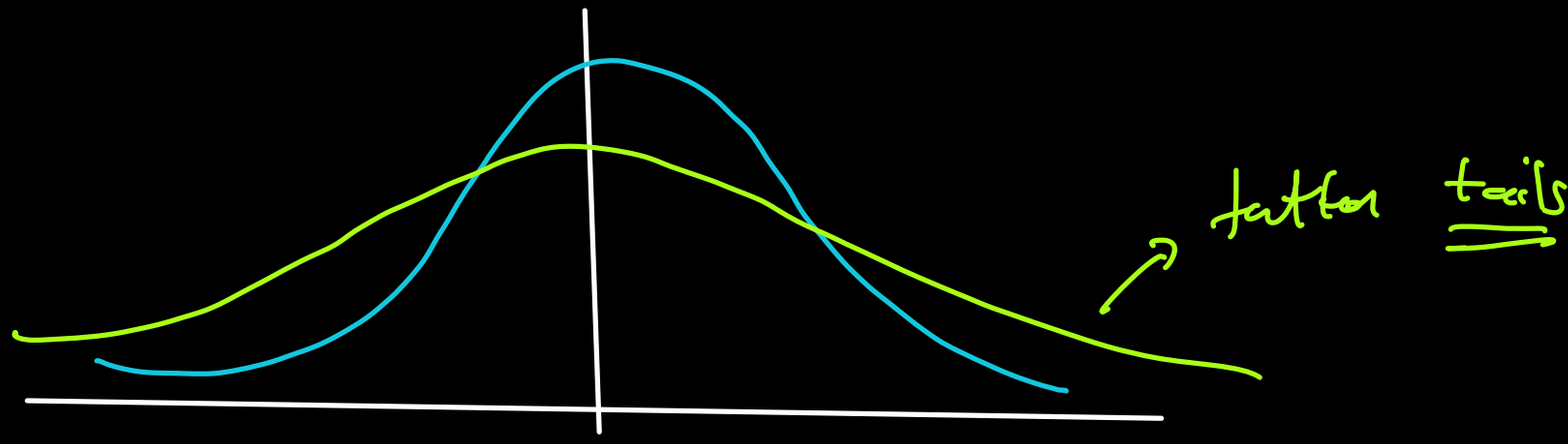
→ Loss func^

Q: Can you think of some metric that compares distributions?

→ KS test statistic

T-distribution

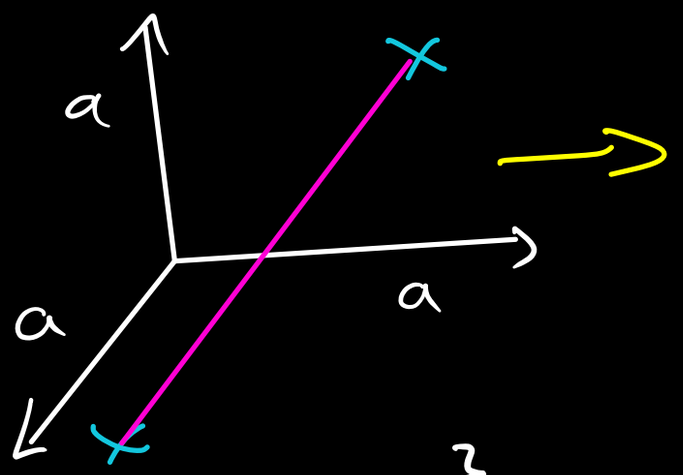
↳ Researchers proposed the use of T-dist instead of 'gaussian', with dof = 1 for d'-dim



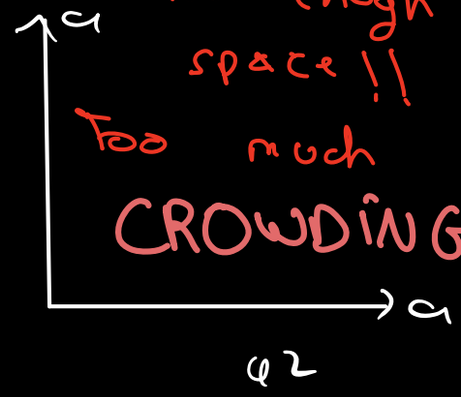
This is able to capture distant neighbours better. → Allow a bit larger distances in d' dimensions

	Volume
1D	l
2D	$l \times b$
3D	$l \times b \times h$
4D	$l \times b \times h \times \gamma \ldots$

→ longer distance exist!

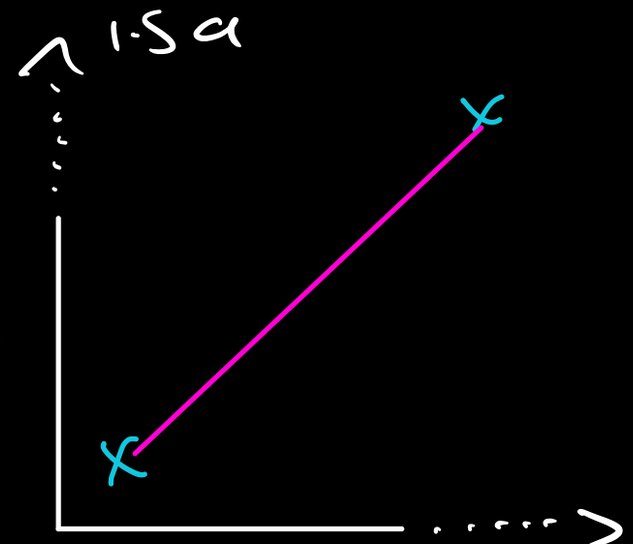


volume = a^3



Not enough space !!

Too much CROWDING



Relative distances need to be slightly longer

3.3 Mismatched Tails can Compensate for Mismatched Dimensionalities

Since symmetric SNE is actually matching the joint probabilities of pairs of datapoints in the high-dimensional and the low-dimensional spaces rather than their distances, we have a natural way of alleviating the crowding problem that works as follows. In the high-dimensional space, we convert distances into probabilities using a Gaussian distribution. In the low-dimensional map, we can use a probability distribution that has much heavier tails than a Gaussian to convert distances into probabilities. This allows a moderate distance in the high-dimensional space to be faithfully modeled by a much larger distance in the map and, as a result, it eliminates the unwanted attractive forces between map points that represent moderately dissimilar datapoints.

In t-SNE, we employ a Student t-distribution with one degree of freedom (which is the same as a Cauchy distribution) as the heavy-tailed distribution in the low-dimensional map. Using this distribution, the joint probabilities q_{ij} are defined as

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}. \quad (4)$$

tances from i to the other datapoints in the two-dimensional map, we get the following "crowding problem": the area of the two-dimensional map that is available to accommodate moderately distant datapoints will not be nearly large enough compared with the area available to accommodate nearby datapoints. Hence, if we want to model the small distances accurately in the map, most of the points

that are at a moderate distance from datapoint i will have to be placed much too far away in the two-dimensional map. In SNE, the spring connecting datapoint i to each of these too-distant map

Hence,

$$Q_{ij} = \frac{(1 + (\|y_i - y_j\|^2)^{-1}}{\sum}$$

KL divergence

Here we use a new loss function,

$$\text{KL Div}(P, Q) = \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

↓ ≠ -ve, proof in post recall.

p_{ij} ↓ if x_i and x_j are far away ≈ 0 we don't

$\frac{p_{ij}}{q_{ij}}$ ↑ large if they are not the

=> some core!

-> its hard for p_{ij} to be smaller for nearby points because q_{ij} comes from T-dist and p_{ij} comes from gaussian! Hence, this wont be -ve easily (very small -ve values)

KL Div is used to compare distributions

Finally, to get new coordinates, we do:

$$\min_{Y} \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

where, $p_{ij} = \dfrac{\exp(-\|x_i - x_j\|^2)/2\sigma_i^2}{\Sigma}$

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\Sigma}$$

Optimisation

Q: Any ideas to proceed further?

$$\min_{Y} \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

→ Gradient Descent

Q: # of learnable params?

→ num pts × num of dim (of y_i)