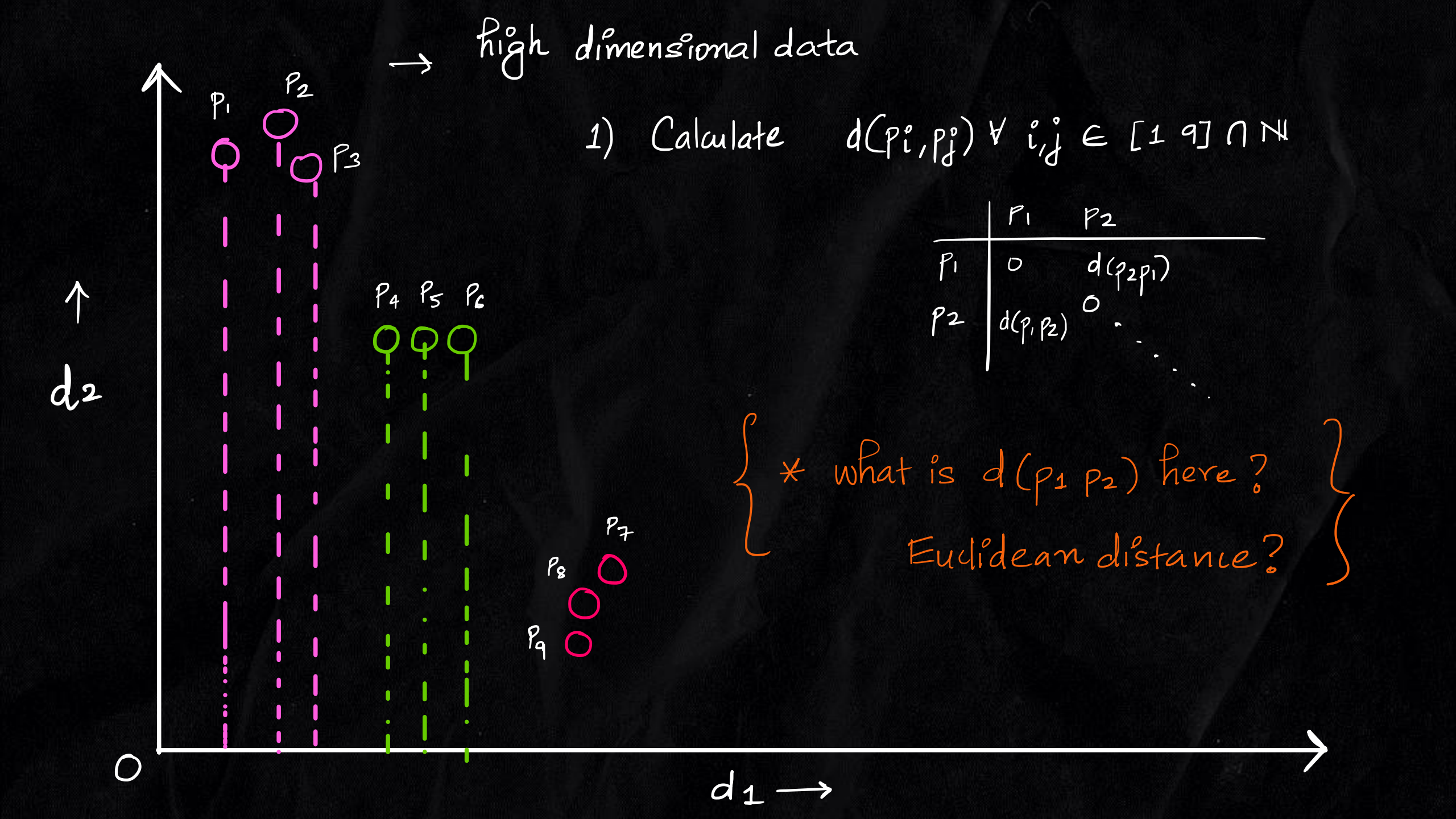
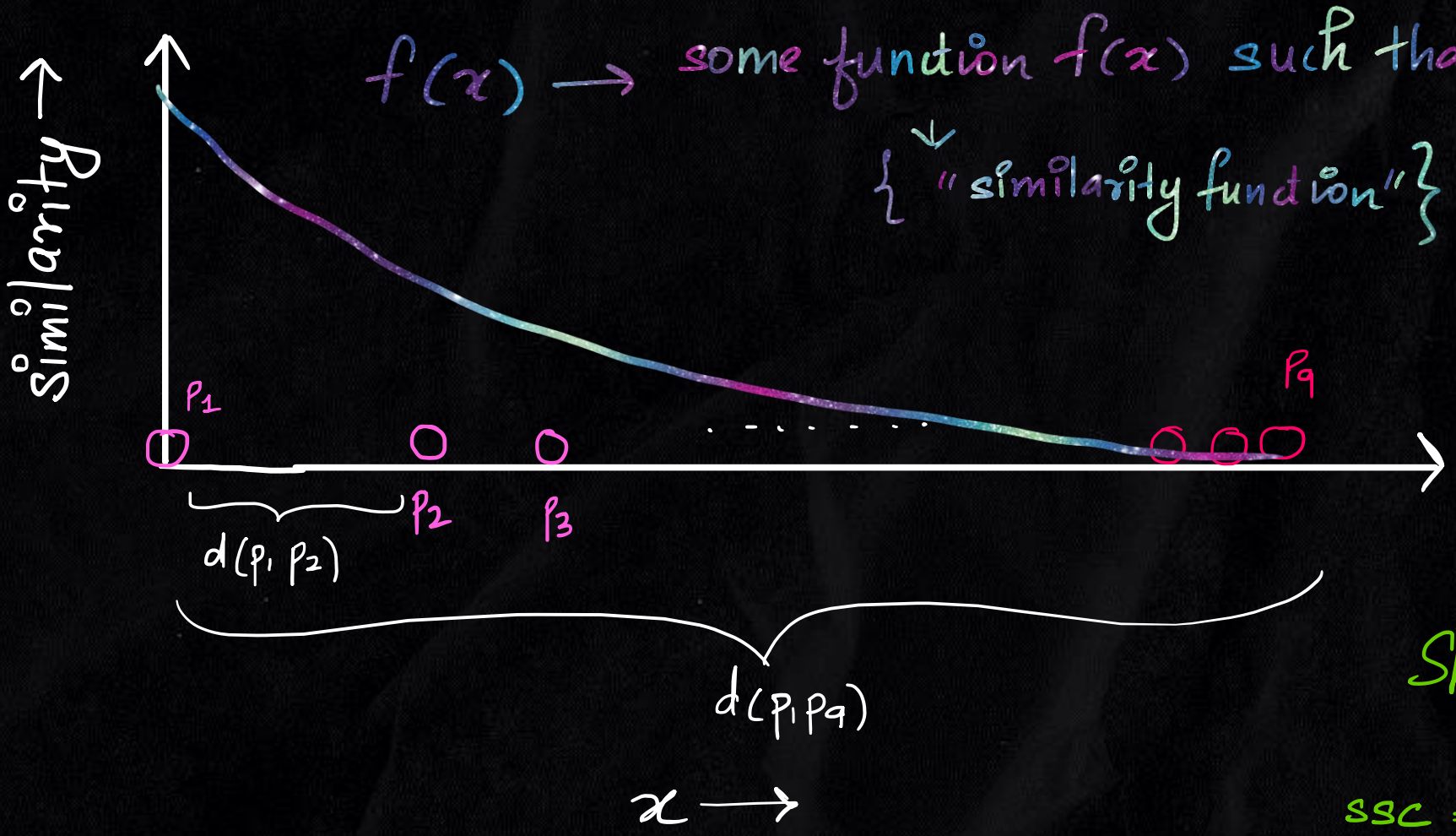

high dimensional data
1) Calculate $d(p_i, p_j)\ \forall\ i,j \in [1\ 9] \cap \mathbb{N}$
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9
d_2
d_1
O
	p_1	p_2
p_1	0	$d(p_2 p_1)$
p_2	$d(p_1 p_2)$	0
* what is $d(p_1\ p_2)$ here?
Euclidean distance?

2) Calculating high dimensional similarity scores for each p_i :→



High dimensional neighbours

parameter (HDN) = 3

Shape of the similarity curve

$$SSC = \log_2(HDN)$$

$$f(d(p_1 p_2)) + f(d(p_1 p_3)) + \cdots + f(d(p_1 p_{HDN-1})) = \log_2(HDN)$$

$$f(x) = e^{-(\underbrace{\text{raw dist}}_{x} - \text{dist to nearest neighbour})/\sigma}$$

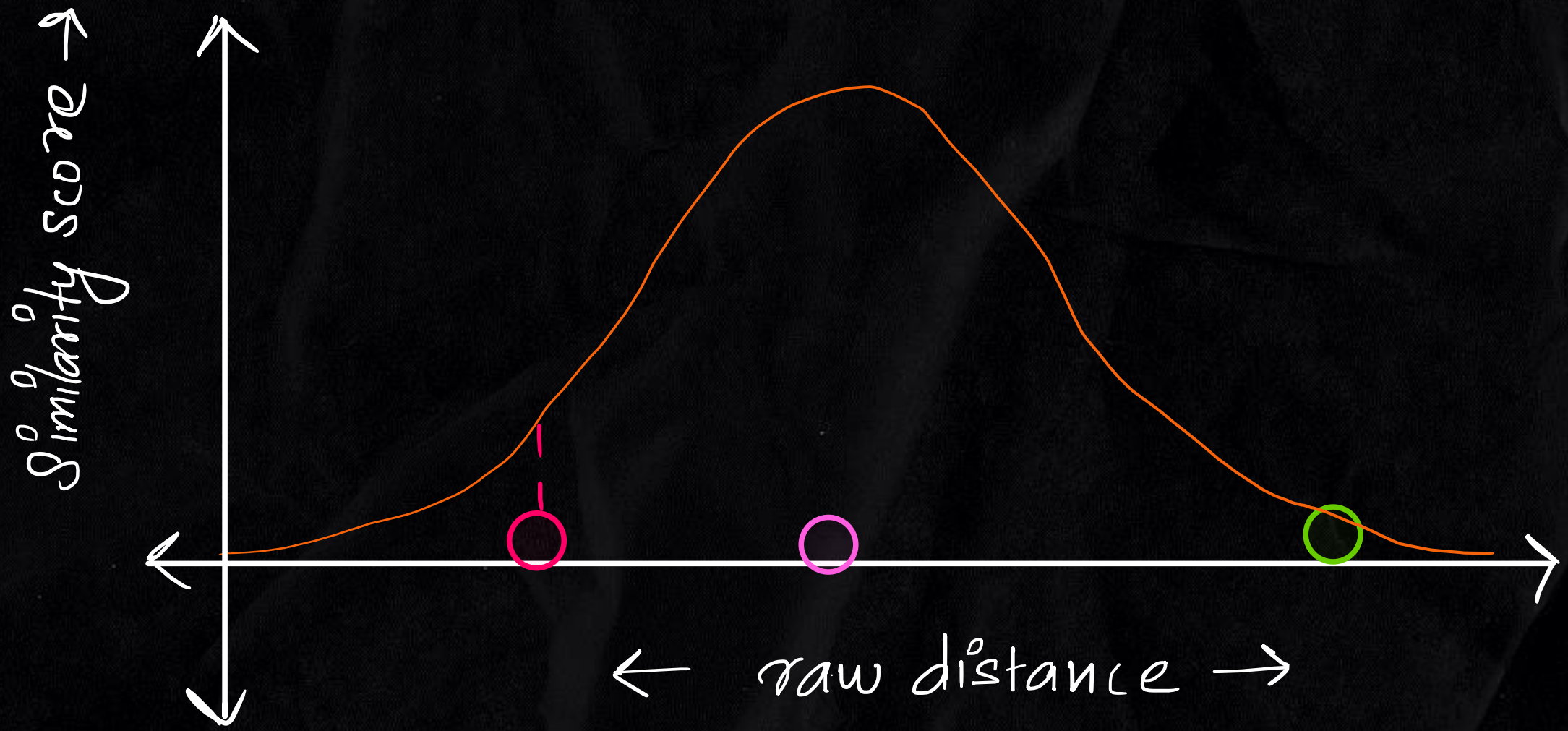
$$f(p_i) = e^{-\frac{[d(o, p_i) - d(o, \text{nearest neighbour})]}{\sigma}}$$

We adjust σ such that

$$\sum_{i=1}^{HDN-1} f(p_i) = \log_2(HDN)$$

→ Note :→ This is the reason why we get unique curves w.r.t each point.

* t - SNE :→ t - SNE differs from U-MAP at this stage in that t - SNE uses a Gaussian distribution to calculate the similarity scores. The shape of the gaussian is controlled by the perplexity parameter {very similar to HDN in UMAP}.


Similarity score →
← raw distance →

Shape of the curve determined by :→

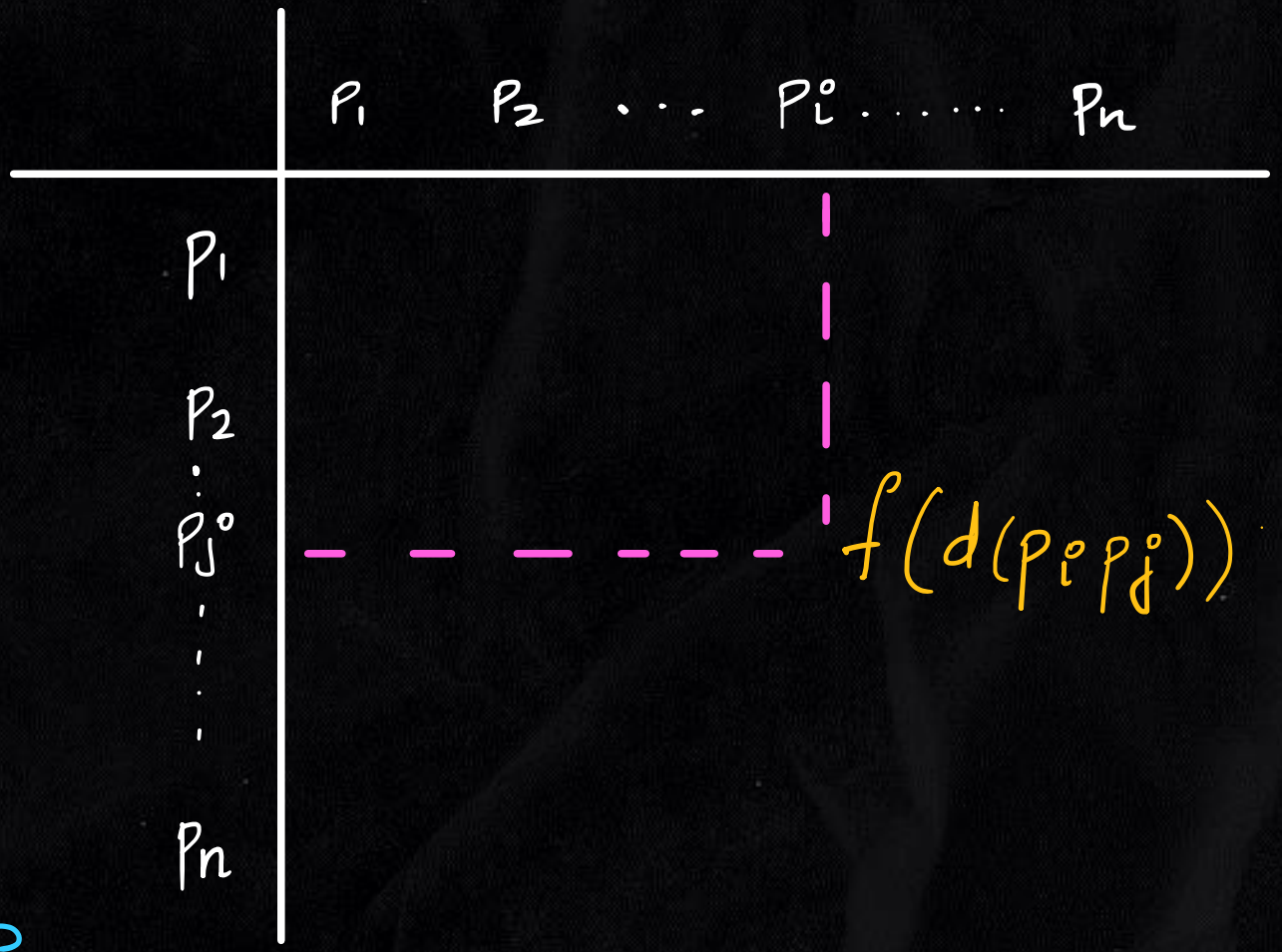
{HDN is the parameter that determines the shape of the curve}

$$\sum_{i=2}^{HDN-1} f(d(p_1 p_i)) = \log_2(HDN)$$

{* what is this function f or f ?}

$f(d(p_1 p_i))$ is the similarity score (HD) of point p_i w.r.t p_1.

Repeat step two for all the points



$f(d(p_i p_j)) \Rightarrow$ similarity score of p_j w.r.t p_i

Similarity scores are not symmetrical $\Rightarrow f(d(p_1 p_2)) \neq f(d(p_2 p_1))$

{you can also write this as $f^*(p_1 p_2) \neq f^*(p_2 p_1)$}

UMAP makes the similarity score ($_{HD}$) symmetrical by doing something like taking the average.

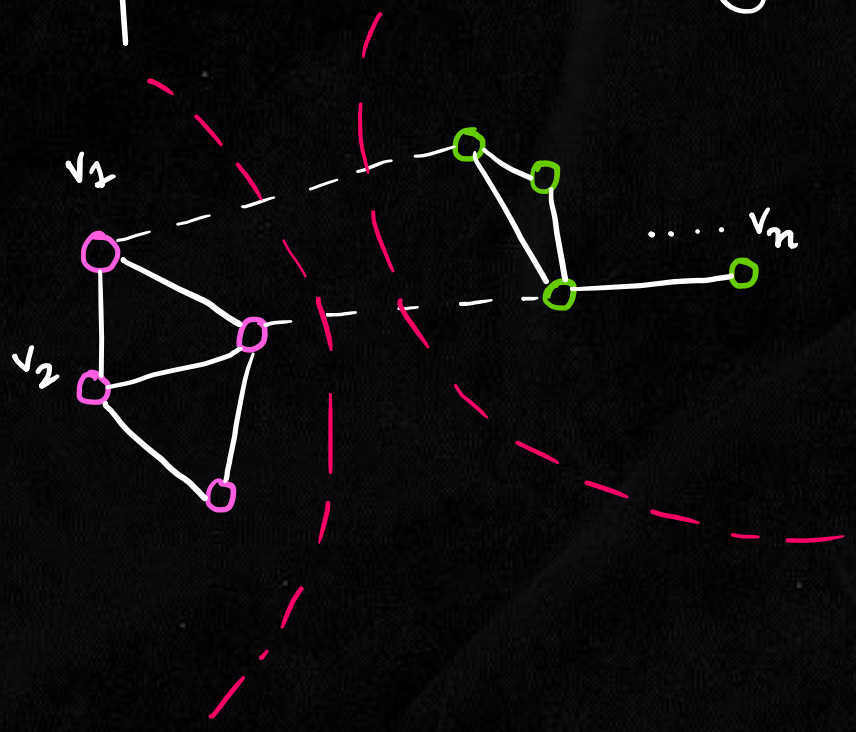
$$f(p_1\, p_2) = \Big[f(p_1\, p_2) + f(p_2\, p_1) \Big] - f(p_1\, p_2) \cdot f(p_2\, p_1)$$

"Fuzzy Union Operation"

3)

Initializing a low dimensional graph {spectral embedding} :→

Spectral Embedding :→



Matrix representation of an undirected graph

	V_1	V_2	⋯	V_n
V_1	0	1	⋯	
V_2	1	0	⋱	
⋮	⋮		⋱	
V_n				0

What is a good cut?

$$cut(A) = \sum_{i \in A, j \notin A} w_{ij}$$

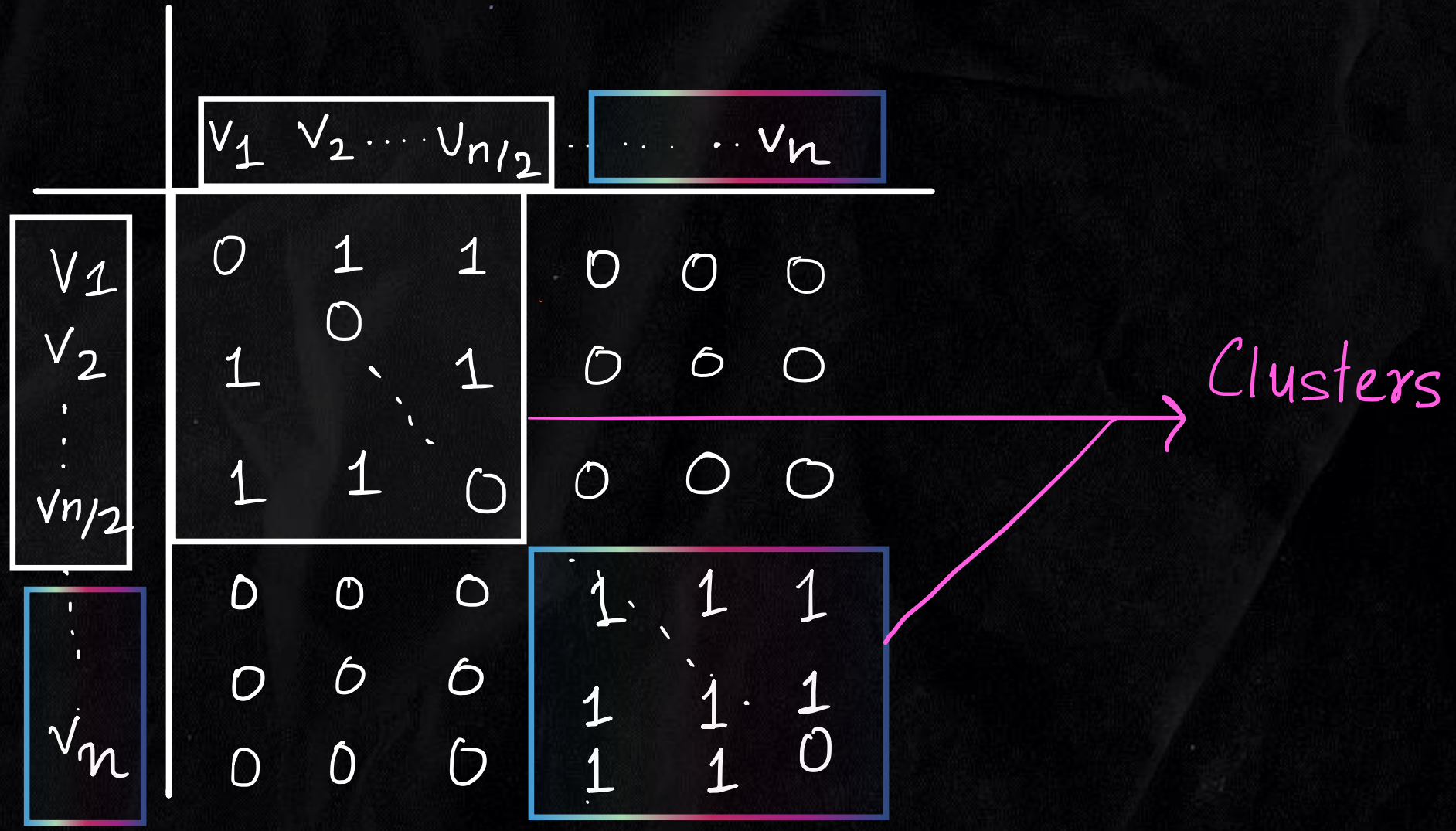
Criterion for good cut :→ Conductance / modularity

$$\phi(A) = \frac{|\{(i,j) \in E ; i \in A, j \notin A\}|}{\min(\text{vol}(A), 2m - \text{vol}(A))}$$

$m \rightarrow |E|$

↳ this prevents you from choosing an unreasonably large A

Adjacency matrix of a two component graph



$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$y_i = \sum_{j=1}^{n} A_{ij} x_j = \sum_{(i,j) \in E} x_j$$

→ for a vertex x_i this is the sum of the vertex values of all its neighbours x_j.

Spectral Graph Theory :→

→ set of eigenvalues in ascending order and their corresponding eigenvectors.

Analyse the spectrum of matrix representing G.

Spectrum :→

$$\Lambda = \{\lambda_1, \lambda_2, \ldots \lambda_n\} \quad \lambda_1 \leq \lambda_2 \cdots \leq \lambda_n$$

$$A \times x_i = \lambda_i \cdot x_i$$

Consider a d-regular graph G :→

$$A \times x = \lambda x$$

↳ solutions to this? λ? x?

For a d-regular graph, $x = [\underbrace{1, 1, \cdots 1}_{n}]$ $\{[1]^n\}$ & $\lambda = d$ is a solution

Now consider a graph with 2 d-regular components.

Suppose you assign node values s.t

$$\text{Node value } (x) = \begin{cases} 1 & \text{if } x \in 1^{st} \text{ component (I)} \\ 0 & \text{if } x \in \text{II } (2^{nd} \text{ component}) \end{cases}$$

Then, the vector looks something like this: →

$$x' = [1, 1, \cdots 0, 1, 0, 0 \cdots 1]$$

$$A \times x' = [d, d, \cdots 0, d, 0, 0 \cdots d]$$

$$\lambda_1 = \lambda_2 = d.$$

some intuition :→

I haven't got this intuition fully, yet

If the two components are poorly connected rather than disconnected, then,

$$\lambda_1 \approx \lambda_2$$

Eigenvectors :→ They are real & orthogonal

$$\{x_i \cdot x_j = 0\}$$

	v_1	v_2	...	v_n
v_1	d_{v_1}	0	0	0
v_2	0	d_{v_2}		⋮
⋮	0	⋱		⋮
v_n	0	...	⋱	dv_n

→ Degree matrix

Graph Laplacian matrix L :→

$$L = D - A$$

	V_1	V_2	⋯	⋯	V_n
V_1	dv_1	-1	0	-1 ⋯	0
V_2	-1	dv_2			⋮
⋮	⋮	0	⋱		
	⋮			⋱	
V_n	-1	⋯	⋯	0	dv_n

→ Sum of each row and column is zero

What are the eigenvectors and eigenvalues of the L matrix?

Trivial solution exists, $x = [1]^n$

$L \times x = [0]^n$, $\lambda = 0 = \lambda_1$ {smallest eigenvalue}

Important fact that we will use without proving :→

$$\lambda_2 = \min_{x} \frac{x^T \cdot M \cdot x}{x^T \cdot x}$$

M → matrix What is $x^T L x$?

$$x^T L x = \sum_{i,j=1}^{n} L_{ij} x_i x_j = \sum_{i,j=1}^{n} (D_{ij} - A_{ij}) x_i x_j$$

$$= \sum_i D_{ii} x_i^2 - \sum_{(i,j) \in E} 2 x_i x_j$$

$$= \sum_{(i,j) \in E} (x_i^2 + x_j^2 - 2 x_i x_j) = \sum_{(i,j) \in E} (x_i - x_j)^2$$

x is a unit vector since x is an eigenvector

x is orthogonal to the first eigenvector $[1]^n$

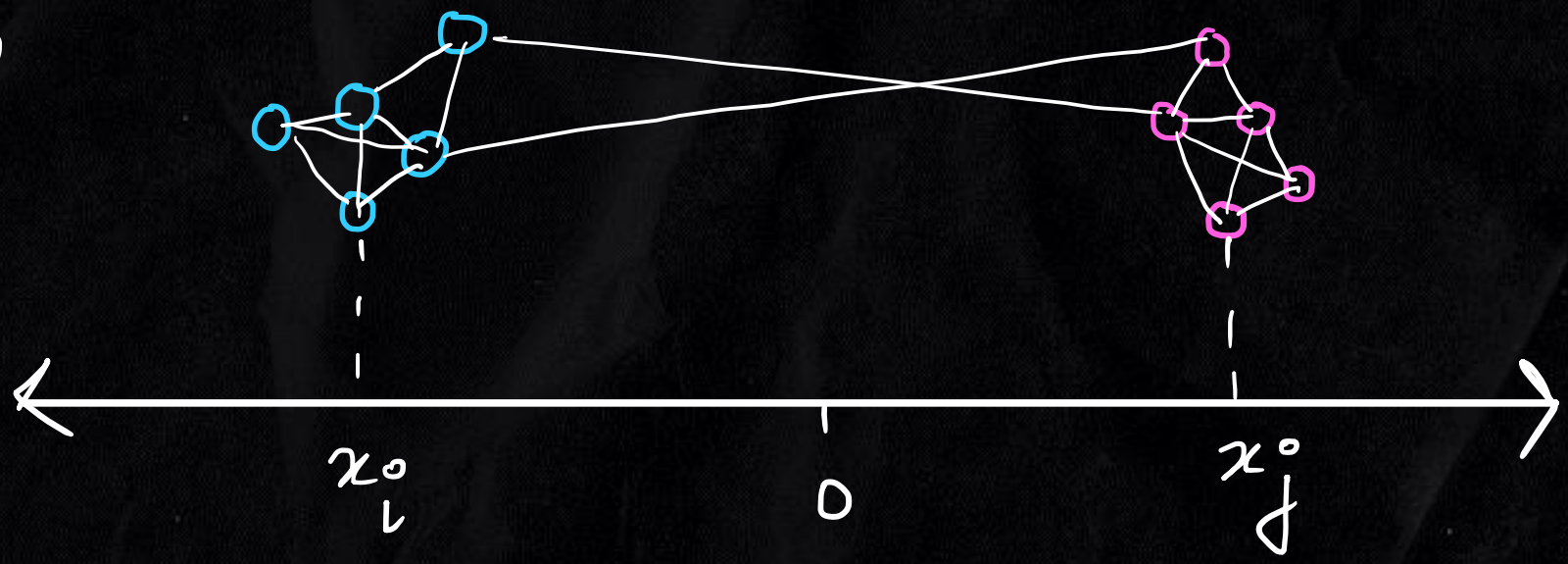
$$\Rightarrow \sum_i x_i = 0$$

$$\lambda_2 = \min_x \frac{x^T L x}{x^T x}$$

$$= \min_{\text{all labelling of nodes } i \text{ so that } \sum x_i = 0} \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum x_i^2 \rightsquigarrow 1}$$

{since x is a unit vector}

$$\lambda_2 = \min_{\substack{\text{all labelling} \\ \text{of node } i \\ \text{so that} \\ \sum x_i = 0}} \sum_{(i,j) \in E} (x_i - x_j)^2$$



So for each $e \in E$, we would like e_{ij}, $i,j \in$ same duster i.e same side of 0.

When we are punished the most? when we have an edge e_{ij} where

$$x_i < 0 \ , \ x_j > 0$$

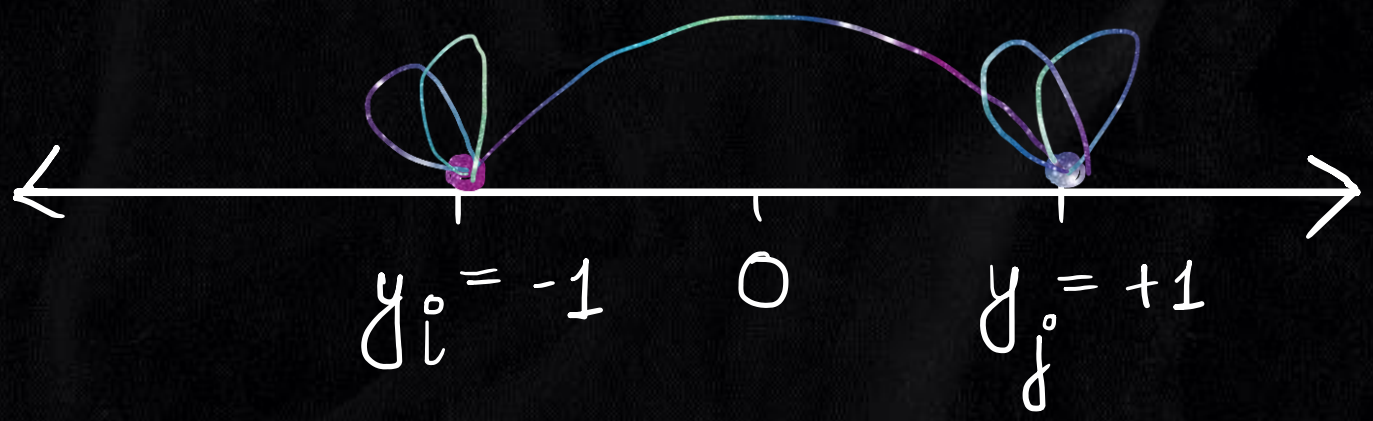
If given a graph & asked to find 2 clusters in it, then

express the two clusters as a vector

$$Y_i = \begin{cases} 1 & \text{if } i \in A \\ -1 & \text{if } i \in B \end{cases}$$

$$\text{Best clustering} = \min_{Y} \sum_{(i,j) \in E} (Y_i - Y_j)^2$$

vector Y



Relaxation :→

instead of $y_i \in \{-1, 1\}$

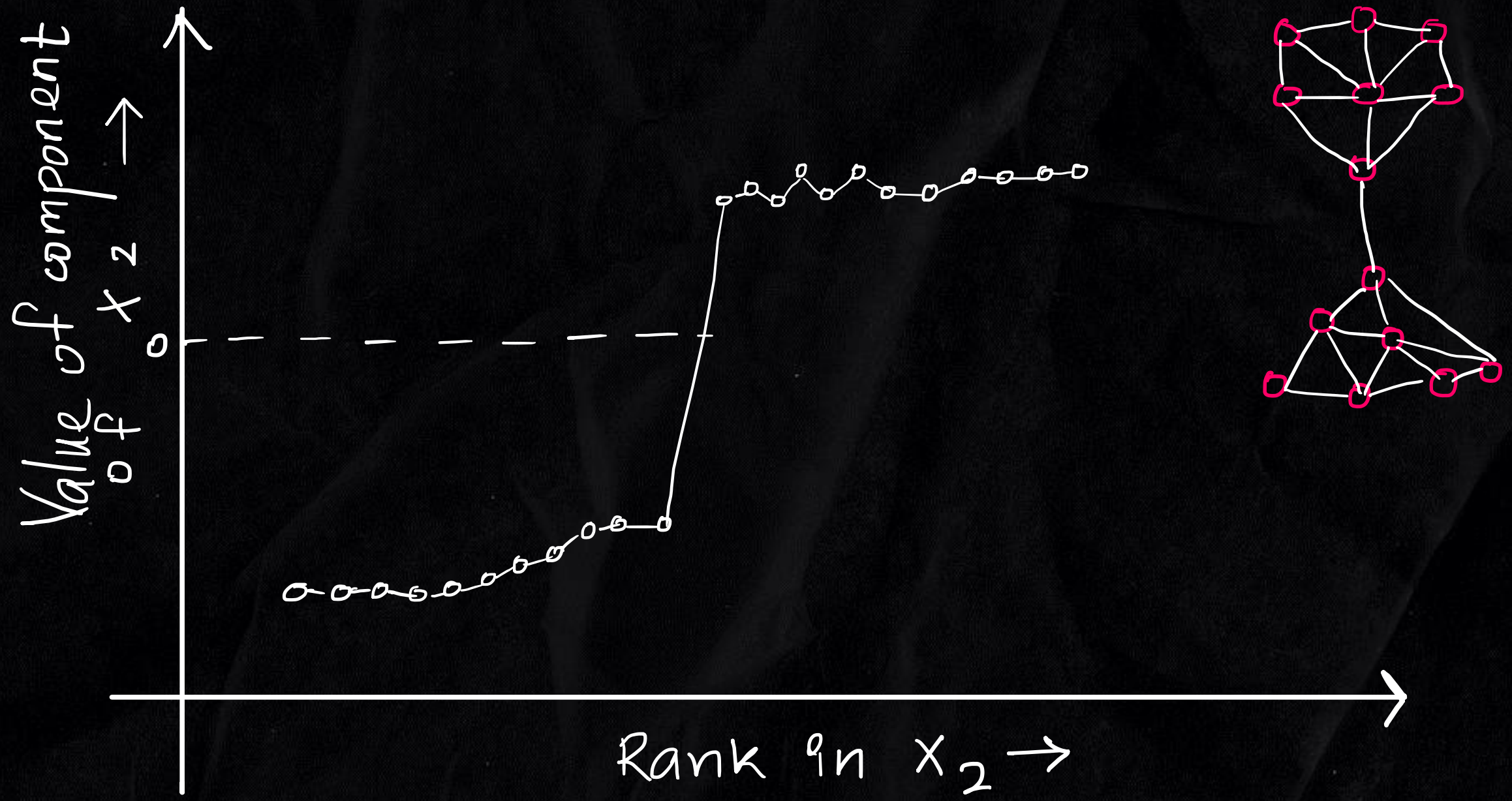
$y_i \in \mathbb{R}$

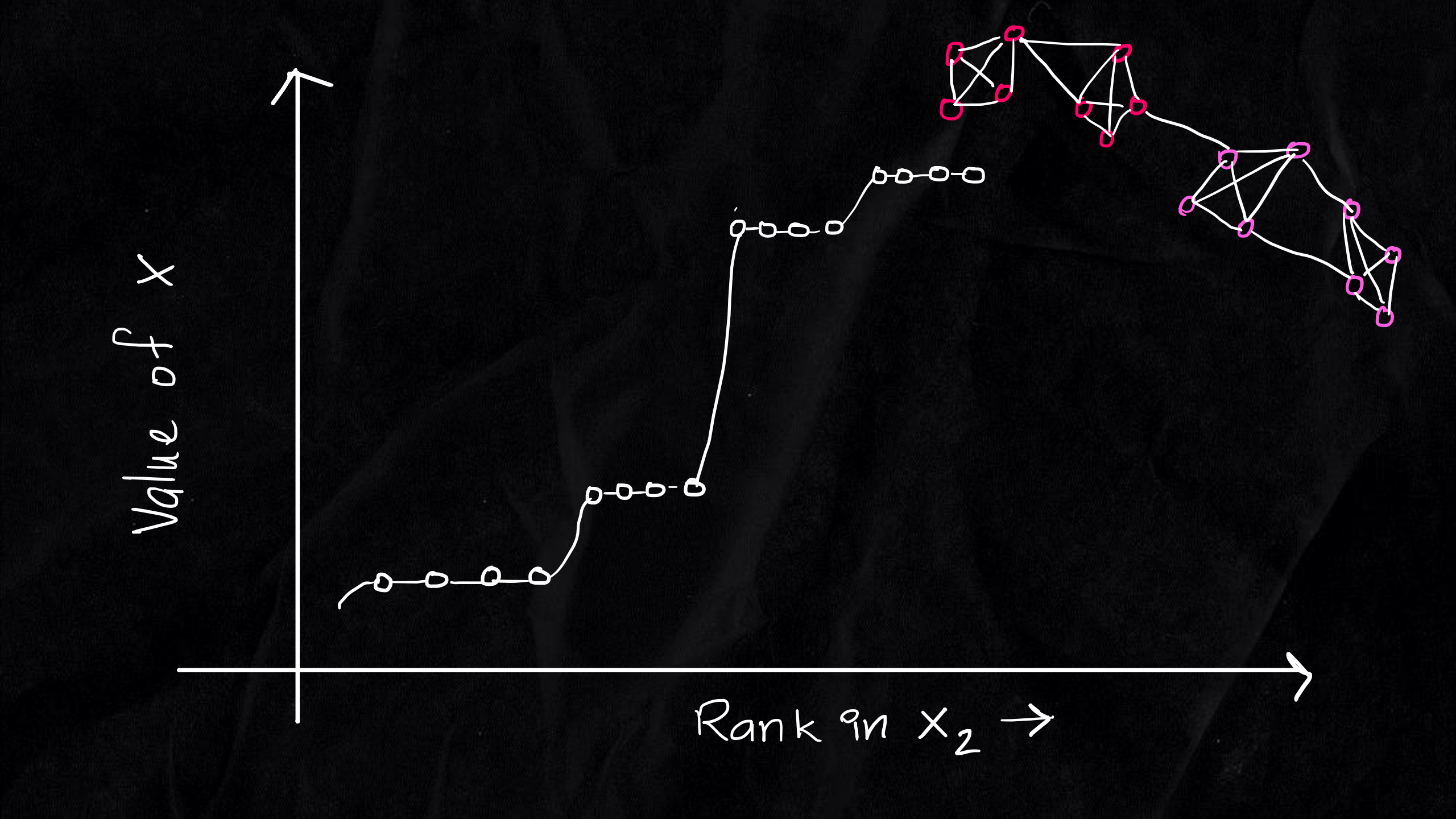
The vector Y that solves this minimzation is the second smallest eigenvector, also known as the Fiedler vector.

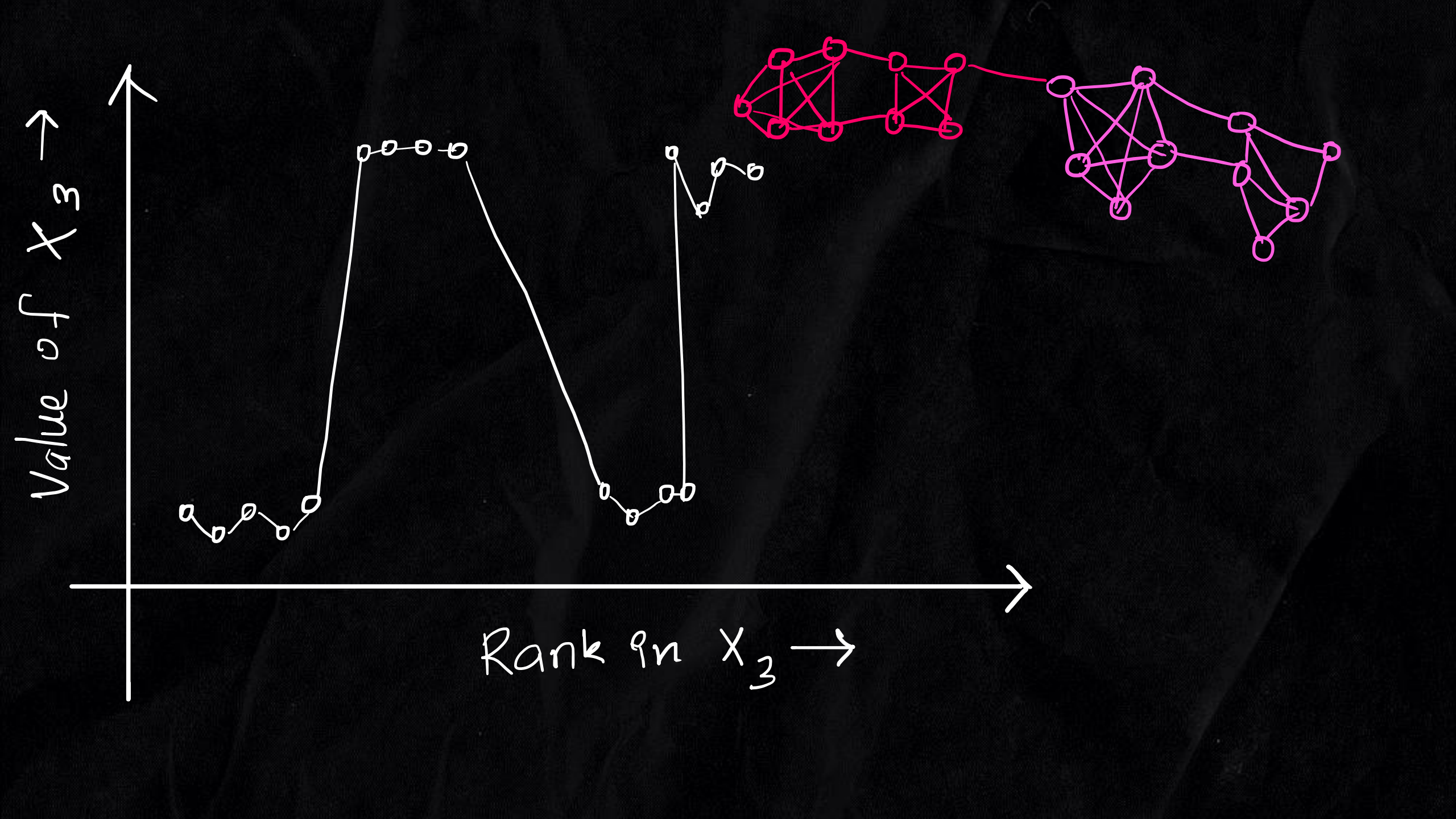
P.T.O

Steps of spectral embedding

- Preprocessing
  - Make matrix Laplacian
- Decomposition
  - eigenvalues & eigenvectors
  - especially λ_2, X_2
- Labelling the vertices / embedding them on the number line using the components of X_2


Value of component of x_2
0
Rank in x_2 →


Value of x
Rank in x_2 →


Value of X_3 →
Rank in X_3 →

Partitioning a graph into k- clusters :→

Express each node of the graph as a vector

for node i → $[x_{2i}, x_{3i}, \ldots x_{ni}]$

↓

x_{ji} ⟶ i^{th} component of the j^{th} smallest eigenvector

Now that each node is expressed by a vector we can use k-means to identify the clusters in the n-dimensional space.

5) Adjustments

→ Randomly choose 2 points within a cluster { P_i and P_i }

{ how does UMAP know whether they are in the same cluster or not? }

→ non-zero similarity score

→ Randomly choose to move one of the two points { P_j }

→ Choose a point from another cluster that the first point should move away from. { P_k }

↓ HD

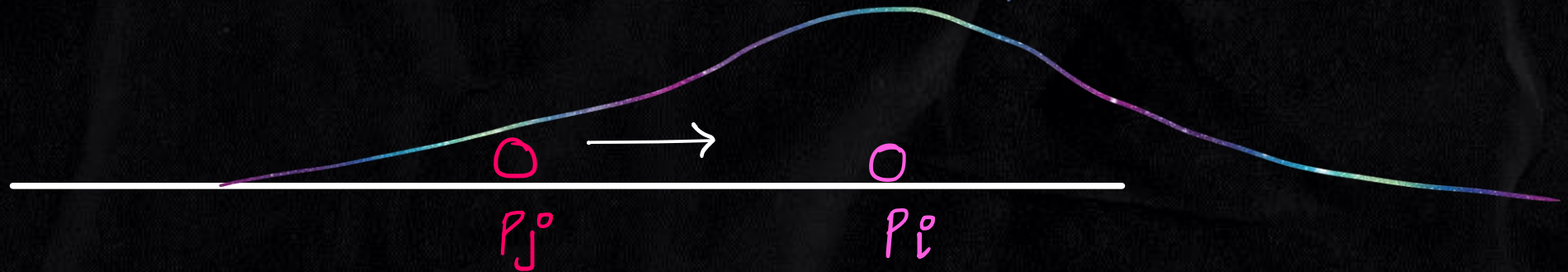
{ using similarity scores? }

→ Calculate low dimensional similarity scores for P_j, P_k & P_j, P_i

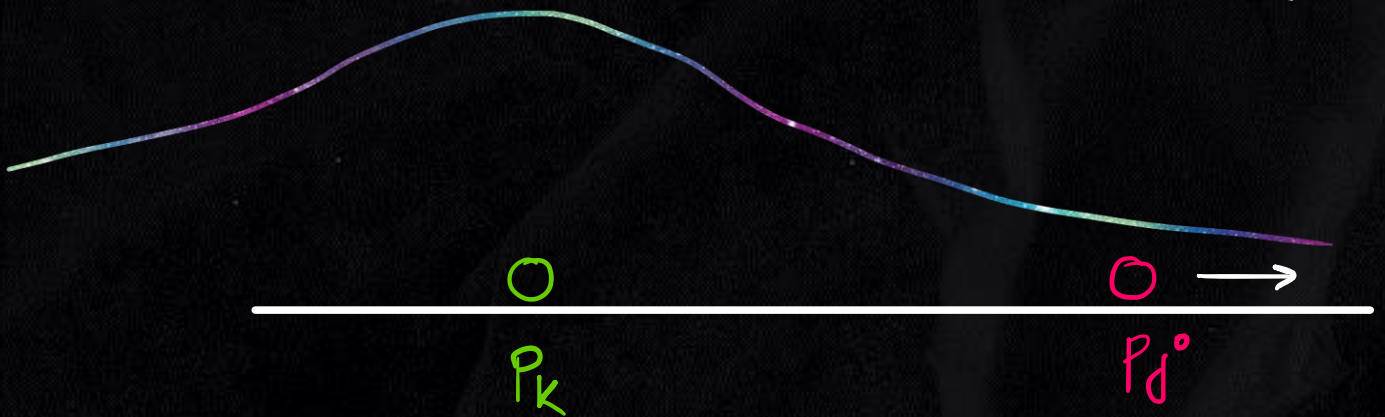
(LD)

5.1) Calculating LD similarity score :→

→ t-distribution

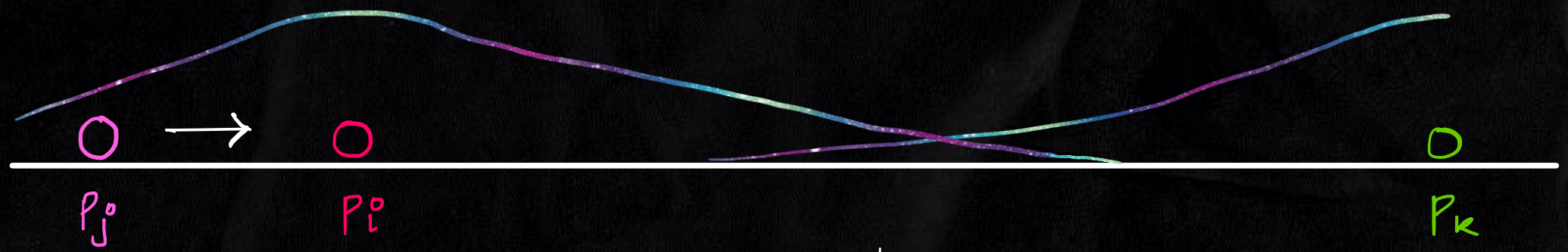


→ Increases the LD similarity score of Pj, Pi & decreases LDSS Pj, Pk.



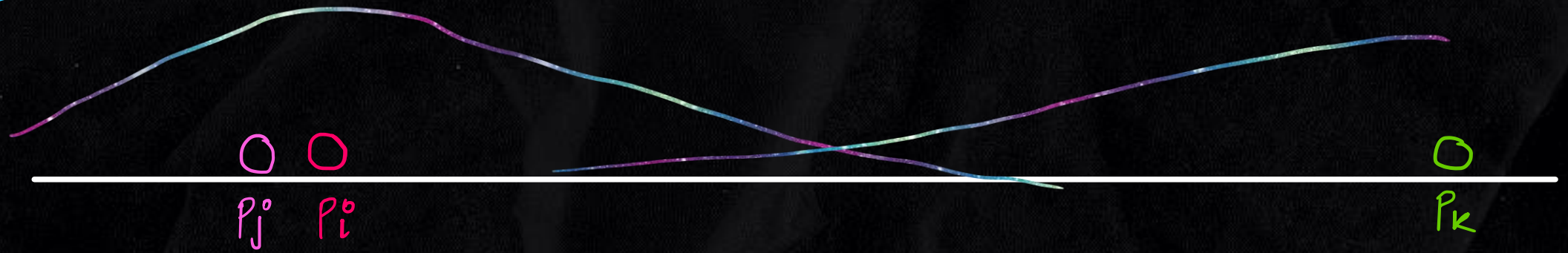
{ how much to move? A lil bit, how little? }

Alternate case →



$f_{LD}(p_j^o, p_i^o) = x$

$f_{LD}(p_j^o, p_k) \approx 0$



$f_{LD}(p_j^o, p_i) > x$

$f_{LD}(p_j^o, p_k) \approx 0$

LDSS between p_i & p_j

$\alpha, \beta \rightarrow$ user defined parameters that decide the minimum distance b/w points on the LD graph & their spread.

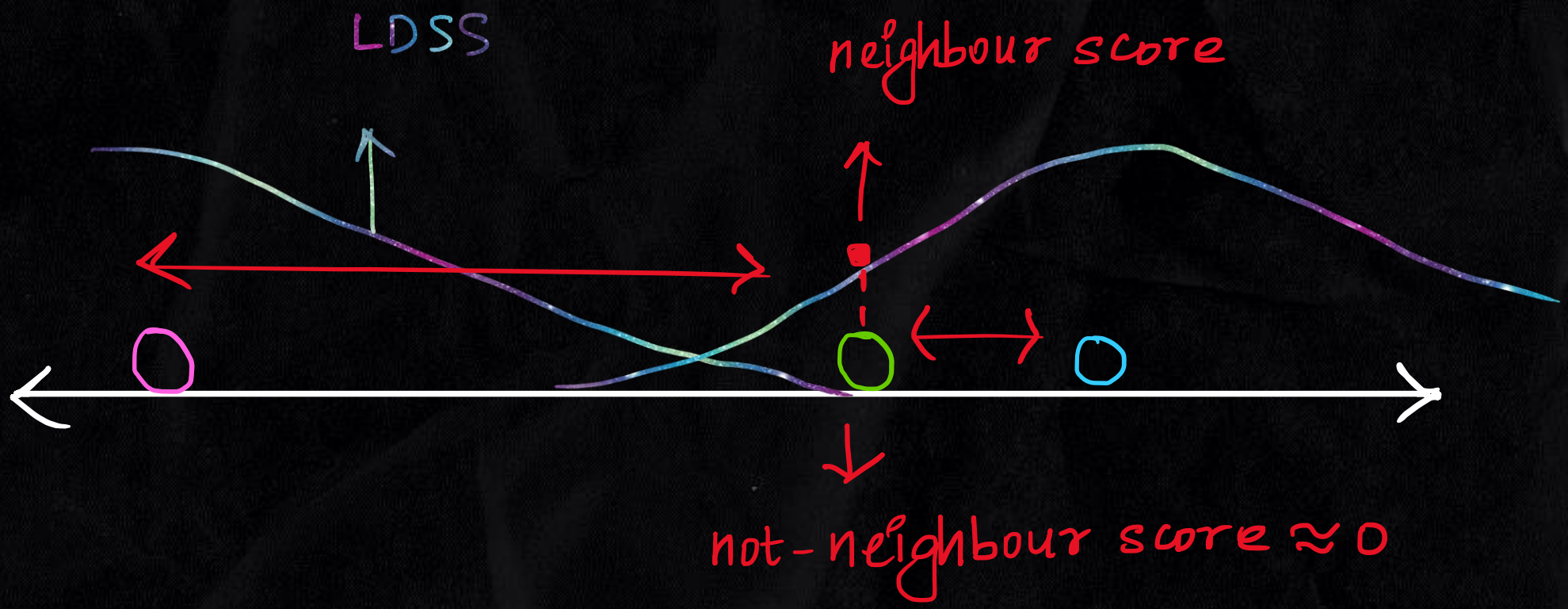
$$LDSS(p_i, p_j) = \frac{1}{1 + a\, d^*(p_i, p_j)^{2\beta}}$$

$d^*(p_i, p_j) \rightarrow$ LD distance b/w p_i & p_j

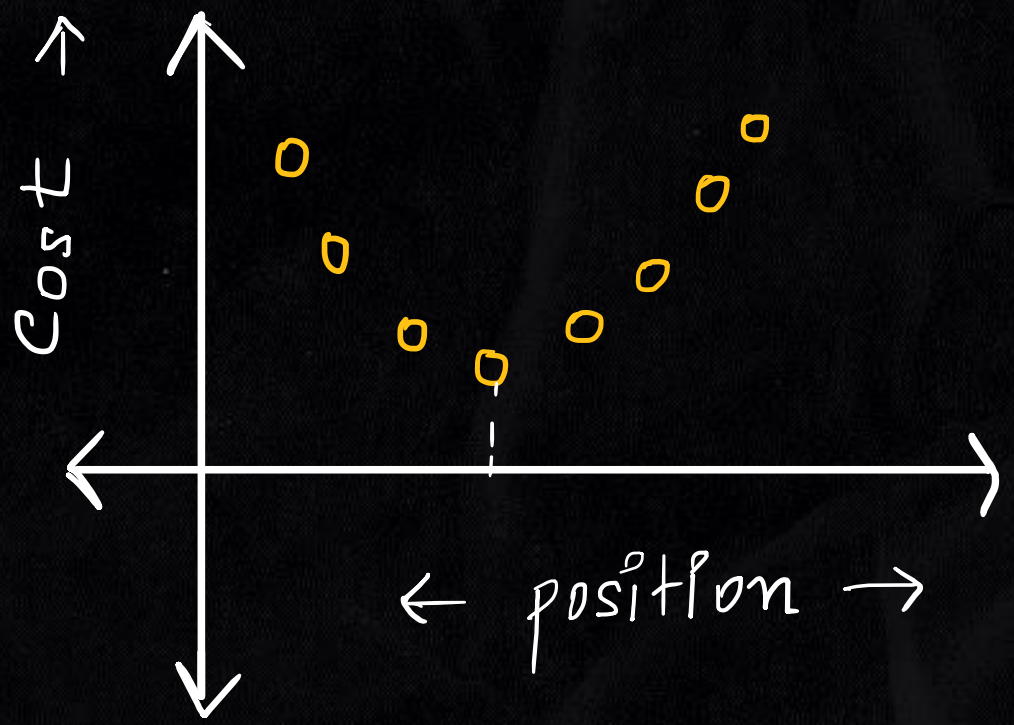
by default $\alpha = 1.577$

$\beta = 0.8951$

t-SNE uses $\alpha = \beta = 1$



$$\text{Cost} = \log\left(\frac{1}{\text{neighbour}}\right) + \log\left(\frac{1}{1-\text{not neighbour}}\right)$$



Cost vs Position for a particular point

UMAP uses stochastic gradient descent to optimise the cost.

→ Move the points based on calculated LDSS

↺ repeat till desired same number of clusters as that in the HD data is acheived.