2) ISOMAP :-

→ Tenenbaum, De Silva & Langford (2000)

→ first algorithm that aims at doing "manifold algorithm"

→ Idea :- Perform MDS using Geodesic Distance as the distance metric btw data pts. estimated on a k-neighbour graph.

⇒ Geodesic is the line of the shortest length joining 2 points of a manifold.
Geodesic Dist. is the length of this line.

⇒ If our manifold is on a hyperplane,

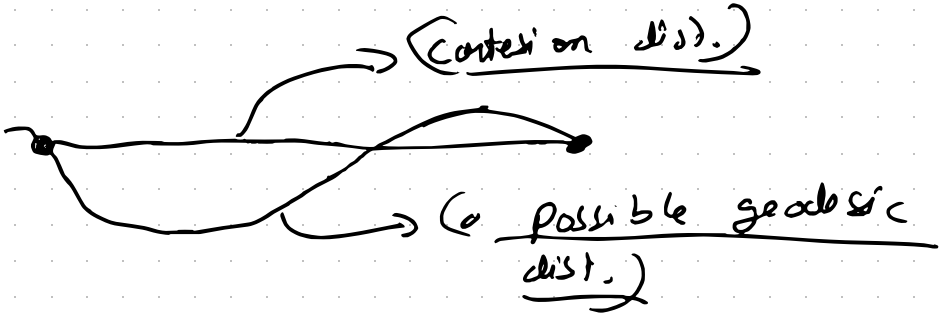
geodesic dist = cartesian dist.

⇒ If manifold is curved,

geodesic dist. ≥ cartesian dist.

↓

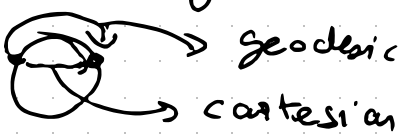
(= holds for manifold being on a hyperplane)



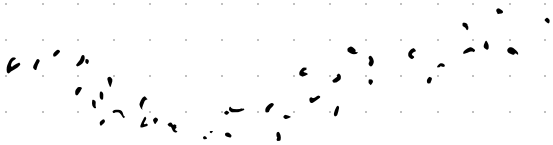
⇒ For eg, Trieste to NY, through the core ≡ cartesian dist.

→ along the surface of the Earth ≡ geodesic dist.

⇒ For a sphere,



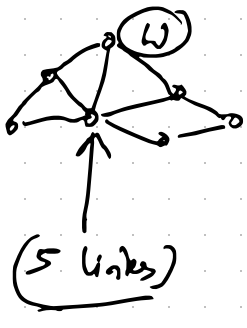
→ K - neighbour Graph :-



→ given a set of data, choose a k.

say, $k = 3$

→ join each data point with its k - nearest neighbours.



→ For $k = 3$ it means each data pt. will have AT LEAST 3 links.

→ it's okay for a point to have ($> k$) links.

→ k-nearest neighbour graph, doesn't imply that degree of node $= 3$ everywhere.

Degree of a network ≡ no. of nearest neighbours in a graph

→ This is a non-directed graph.

→ It is also a weighted graph, i.e. all links are NOT the same & have a weighting factor.

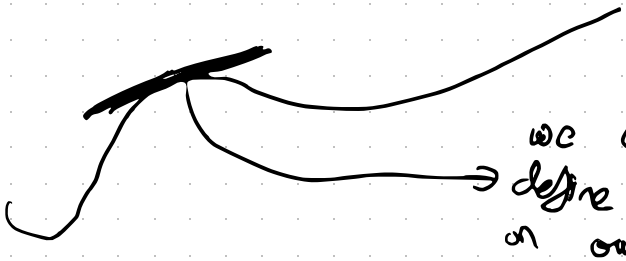
$$W^{ij} = \| X^i - X^j \|$$

↓

Weighting factor for ISOMAP (without the $(\,)^2$, due to convention)

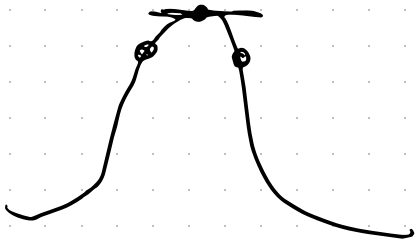
ASSUMPTION :- embedding manifold is locally flat

→



we can define a tangent hyperplane on our embedding manifold

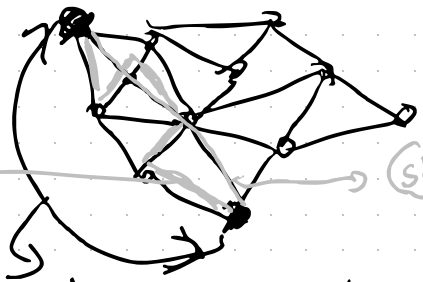
→ in this case $\mathcal{M}$ is NOT locally flat



→ if manifold is smooth, but data pts are so far, that already there is some curvature. on the scale of the first neighbours

⇒ locally flat ≡ on the scale of first k-neighbours the manifold is approximately flat.

⇒ Now we find the geodesic on the graph.



Imagine we have to find the geodesic dist. btw these 2 points

geodesic dist.

→ already we know ∧ can't contain any cycles, i.e., it can't go through the same node twice

⇒ weight of each path is NOT equal to the number of links.

Instead, weight of each path = sum of weights of each link comprising the path.

so even if we pass through more nodes, it doesn't matter as long as weights of each link are smaller when added up.

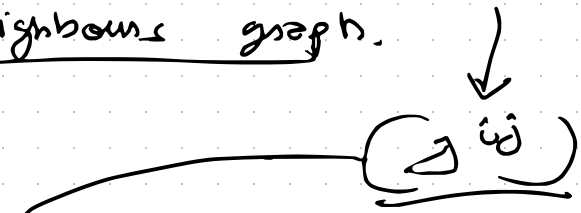
$$\therefore L(i_k) = \sum_{k=1} w_{i_k i_{k+1}}$$

$$w_{i_k} = \| x^{i_k} - x^{i_{k+1}} \|$$

→ find this for many diff. paths & then take the minimum

→ w_{i_k} ≡ weights of each link

⇒ Let's say $\forall i, j$ we can estimate the Geodesic Dist. on the k-neighbours graph.



↳ even though it's only labelled by (i, j), it depends (explicitly) on the pos^{n} of all the data points in the graph & it depends on the nature of the graph itself, & the weights of the links.

⇒ Perform MDS using Δ^{ij} as a distance.

That is, first we do double centering, then find the dominant eigenvectors in desc. order, until the eigenvalues become small enough.

⇒ The eigenvectors of $\{G\} \equiv$ Gram matrix, built in this manner, will solve

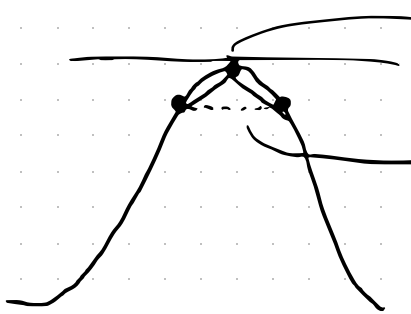
$$\| y^i - y^j \|^2 \approx \Delta^{ij}$$

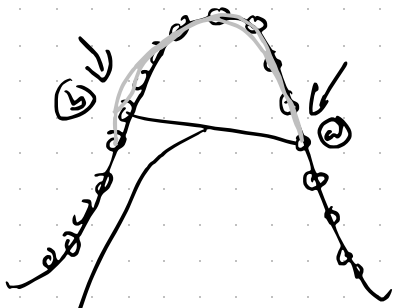
(i.e., it will unwrap the manifold)

→ Now (Δ^{ij}) implicitly depends on the whole graph, the dist. in Y-space is only dependent on $(i \& j)$

That is, we have managed to find a representation of our manifold, that flattens it, i.e., we have a Cartesian or Euclidean representation of our data manifold.

N.B.,





→ Here, if $k = 2$ / $k > 3$ geodesic dist. should be fine.

↳ But, let's say, we set $k = 10$

↳ then for some diffn the 10 nearest neighbours there will be pts. like (a) & (b). Now, the geodesic dist. btw (a) & (b) when computed will give us shortest lines (like the one marked), ∵ it is shorter than the dist. along the manifold.

→ ∴ We find sol^{n} that are wrong ∵ geodesic dist. btw (a) & (b) will be smaller than the correct geodesic dist.

→ This is why, the locally flat approx^{n} is important, ∵ otherwise we add shortcuts

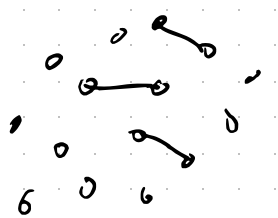
→ So, 'k' is chosen s.t. on the scale of euclidean dist. of k^{th} nearest neighbours, the manifold is approximately "locally" flat.

→ how do we choose 'k'?

→ we want 'k' to be as small as possible
∵ if 'k' is too large we run the risk of cutting through corners.

→ "k" also depends on no. of points.

→ if k is too small → graph can become disconnected,



→ if k = 1

i.e. there's no way of going from one point to another.

→ simplest rule of thumb is to choose the smallest value of 'k' that makes the graph "globally connected"

→ easy way to check this in Floyd algorithm.

For the same manifold n = 10^4, k = 10 may be okay
n = 10^3, k = 10 may be wrong
∵ density of pts. violates locally flat approx.

→ Floyd Algorithm :-
(ALL-TO-ALL)

→ To find the shortest path b/w any pair of nodes in a graph, such as, as kNN graph.

→ Output :- Δ^{ij} (Geodesic dist. on kNN graph b/w i & j)

→ Input :- Initial guess for the distances

$$\Delta^{ij} = \begin{cases} |x^i - x^j|, & \text{if } (i) \& (j) \text{ are linked} \\ \infty \text{ (large no.)}, & \text{otherwise} \end{cases}$$

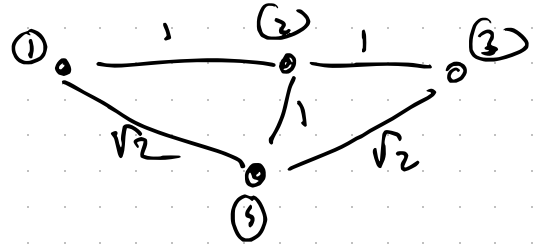
→ you can have a link even if you're not among the first (k) NN, ∵ it can be you are among (k) NN of (j) & are also linked.

N = no. of nodes → 3 nested loops that run over all the nodes

FOR K = 1, N
  FOR i = 1, N
    FOR j = 1, N

IF ($D^{ij} > D^{ik} + D^{kj}$)
$D^{ij} = D^{ik} + D^{kj}$

Eg. of working (how the algo. works in case of a missing link)



→ per initialization all dist. are known, except D_{13}

$D_{12} = D_{23} = D_{24} = 1$

$D_{14} = D_{34} = \sqrt{2}$

$D_{13} = \infty$ (as per initial value of Floyd alg.)

→ in one loop we will compare

$\because$ Δ_{13} (∞) $>$ $\Delta_{12} + \Delta_{23}$ (1) + (1)

$\Delta_{12} + \Delta_{23}$ ← Dist$(k=2)$

$\therefore \Delta_{13} = \Delta_{12} + \Delta_{23} = 2$

Δ_{13} (2) $<$ Δ_{14} ($\sqrt{2}$) $+$ Δ_{43} ($\sqrt{2}$)

$\Delta_{14} + \Delta_{43}$ ← $(k=4)$

(2) $< 2\sqrt{2}$

$\therefore (\Delta_{13} =$)

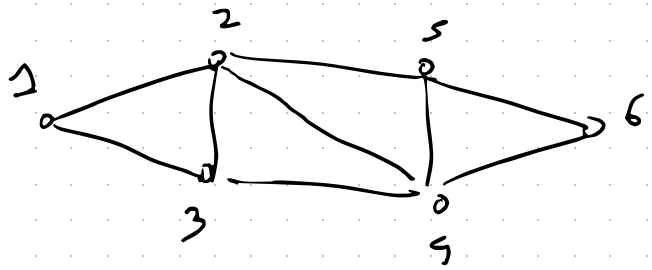
This shows us how using Floyd's alg. we systematically build the linked distances $\forall i,j$

⇒ Qualitatively, what is happening here?

(i) The previous page explains how the algo. works in the case of 1 missing link.

(ii) If we have more than 1 link to fill, at every iteration, we loop over all possible iterations of (i) & (j), we extend our link by 1.

e.g. :-



→ how to get dist btw (1) & (6)?

(i) not possible in a single cond^n.

(ii) first we get dist btw (1) & (4)/(5)

(iii) next loop, we get (1) & (6)

→ So, each iteration, we extend the connected part of the graph by 1 link.

∴ Eventually it loops over all nodes & connects all the graph.

→ Scaling of Floyd's Alg :- $O(N^3)$

↓ poor scaling

↳ but it is exact ↳ for any graph it will give correct results with exact values.

→ if the graph is sparse (# of links $\ll N$) & we accept the geodesic dist. within a certain tolerance we can reduce scaling to $O(N^2 \log N)$

↓

Dijkstra's Algorithm

↓

~ scaling is $O(N^2 \log N)$ in general

~ accuracy is only good for sparse graphs.

$N \approx 10^6$ pts

$O(N^2)$

$= 10^{12}$ operations

This also shows us if our graph is connected.

→ ∵ we loop over all distances in the algo & if some dist. haven't changed w.r.t. initial set (& value = ∞) then the graph is disconnected ∵ the algo can't find a connecting path.

→ If our data manifold is a circle clearly it is an ill-posed problem,

∴ we can't use

rigorous Euclidean distances on a circle without breaking it.

→ Floyd's algorithm is an efficient way of finding if our graph is connected or not.

→ This algo tries to approximate the geodesic dist on the graph with the euclidean dist in the space of $\{Y^i\}$.

→ (that is where this problem comes up.)

⇒ One of the key cond^{n} a manifold learning algorithm should satisfy is : 2 pts that are close in original dataset, CANNOT be very far away in reduced dimensional representation.

→ ID of circular manifold = 1

To have a 1D-representation, the requirement of having a euclidean metric on our data is at conflict with topological constraints.

→ It is essential do not lean close data pts. apart, i.e. instead of taking reduced dim. representation = ID, we should take (reduced dim = ID + 1) at least.