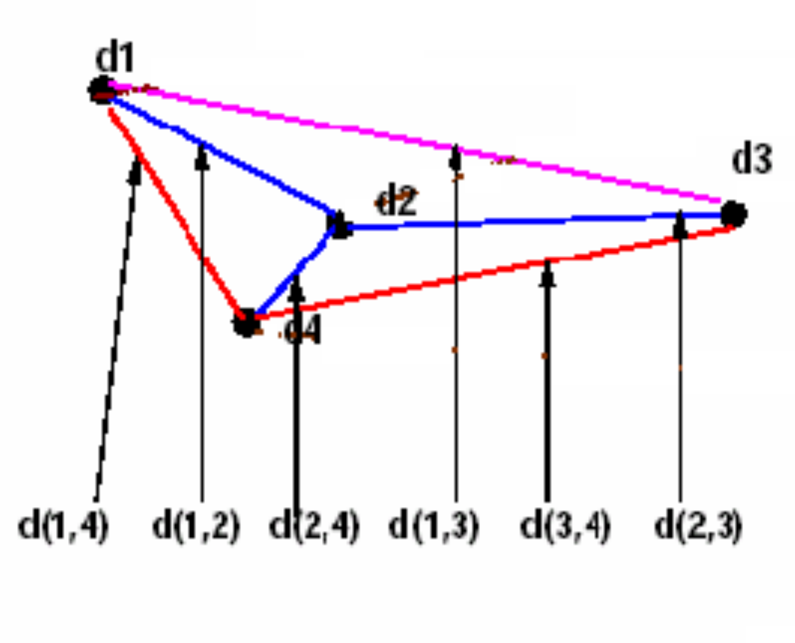
MULTIDIMENSIONAL SCALING AND FASTMAP

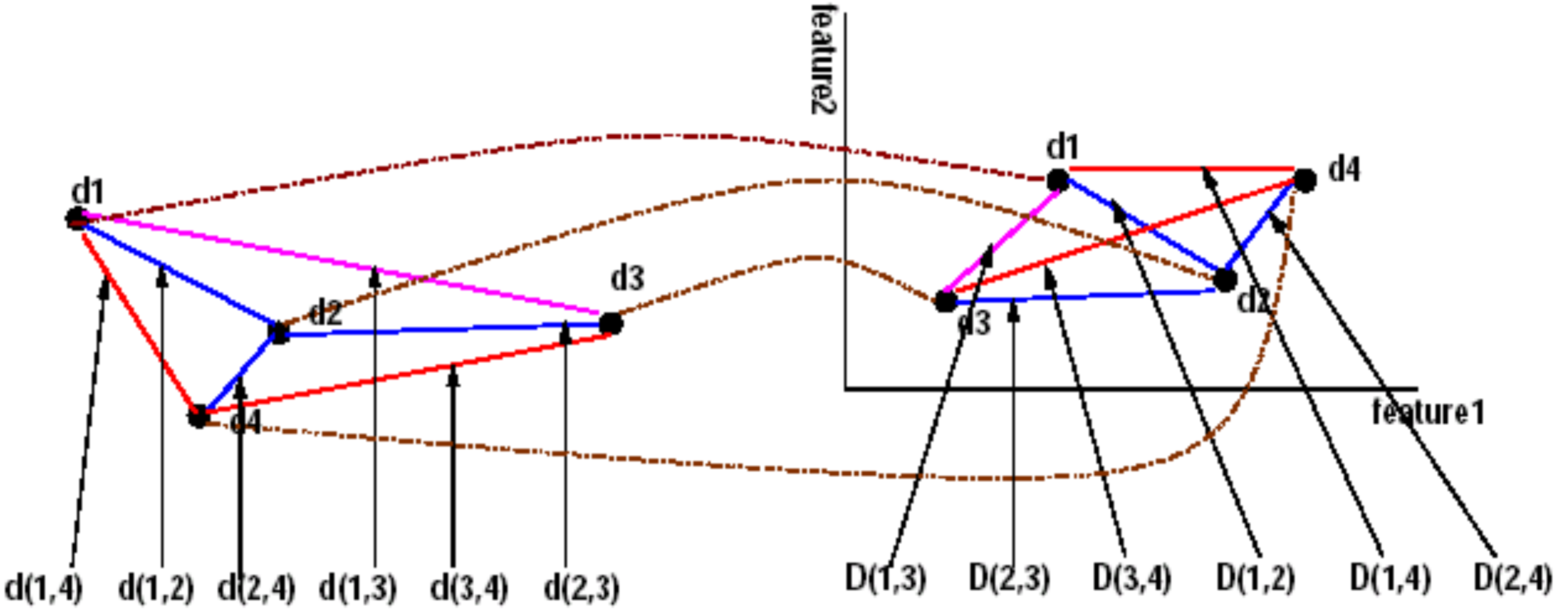
What if we do not have features??



We know the distances, but

- we do not have explicit features, or
- they are not metric….

MDS: How do we map the query?



(a)

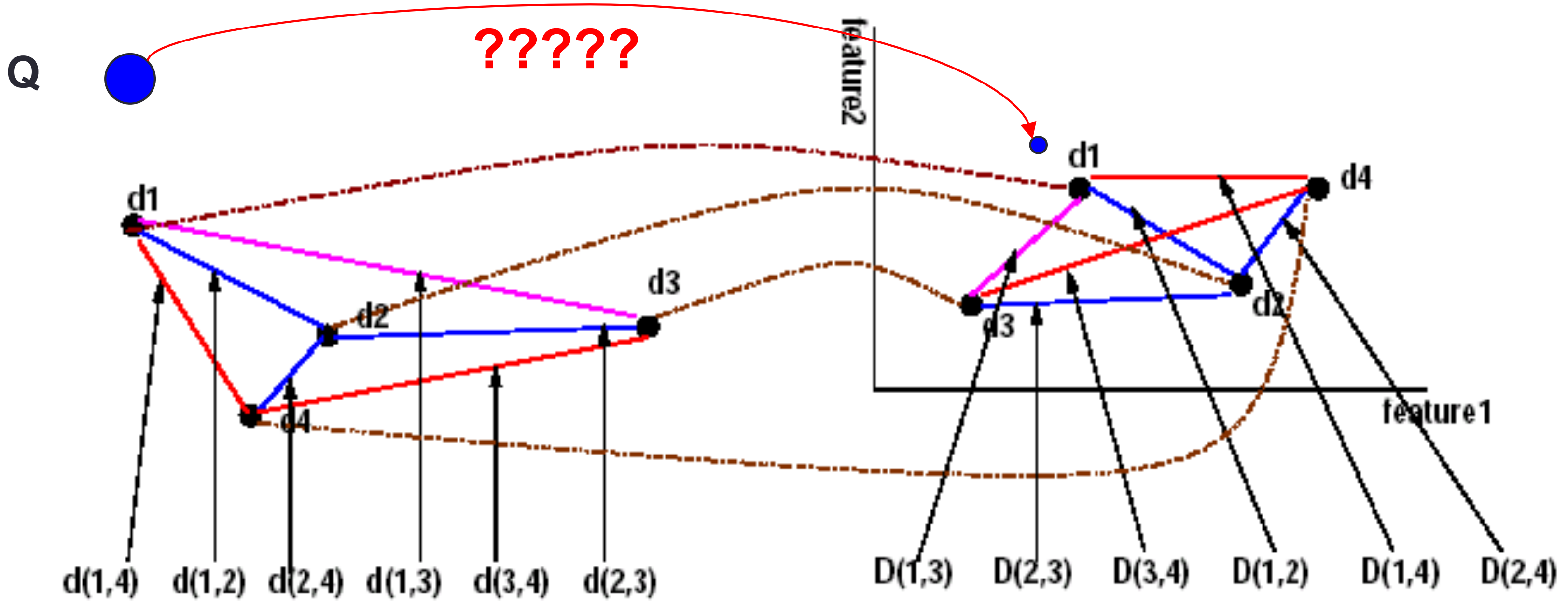
MDS

- The criterion for the mapping is to minimize *stress*

$$stress = \sqrt{\frac{\sum_{i,j}(d'_{ij} - d_{ij})^2}{\sum_{i,j} d_{ij}^2}}$$

- Start with a (random) configuration of points in small # dimension
- Apply some form of steepest descent iteratively to minimize the stress.
  - move objects
  - If moving objects does not help add dimensions
- Requires *$O(N^2)$* distance computations + the cost of iterations

Embedding into a “space”



(a)

$$
\begin{array}{lll}
d(1,2) \approx D(1,2) & d(1,3) \approx D(1,3) & d(1,4) \approx D(1,4) \\
d(2,3) \approx D(2,3) & d(2,4) \approx D(2,4) & d(3,4) \approx D(3,4)
\end{array}
$$

MDS: How do we map the query?

- Option 1: Find the distances between the query and all the objects to update the mapping
  - *O(N)* distance computations
  - ..why bother using index structures if cost is *O(N)*

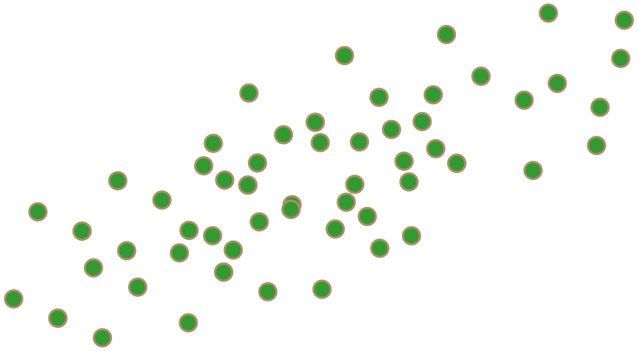
- Option 2: Find the distances between the query and select few objects
  - Can save a lot of time
  - ..the query mapping is not perfect…depends on the objects selected
  - How to select the objects to help mapping of the query?

FastMap

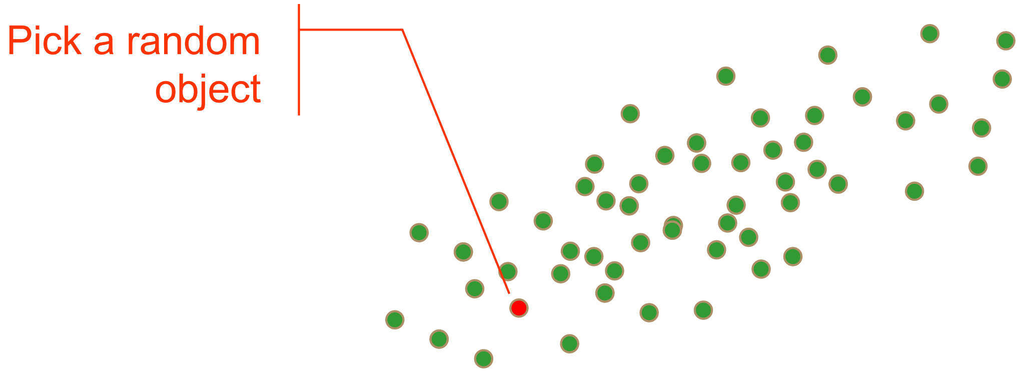
- MDS requires
  - *$O(N^2)$* distance computations (+ the cost of iterations) for the initial mapping
  - *$O(N)$* distance computations for mapping the query
- Can we do better?
- FastMap requires
  - *$O(kN)$* distance computations for the initial mapping
  - *$O(k)$* distance computations for mapping the query

where *$k \ll N$* is the resulting number of dimensions.

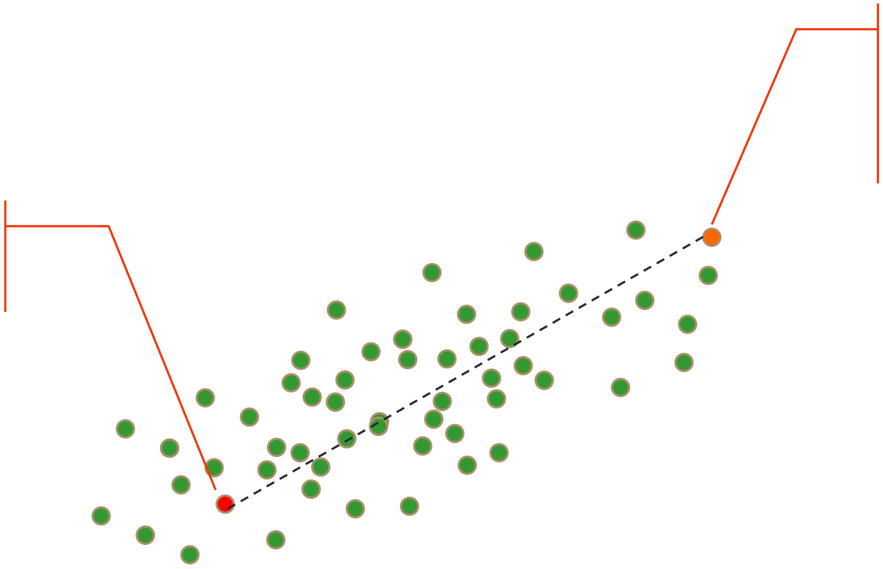
FastMap



FastMap



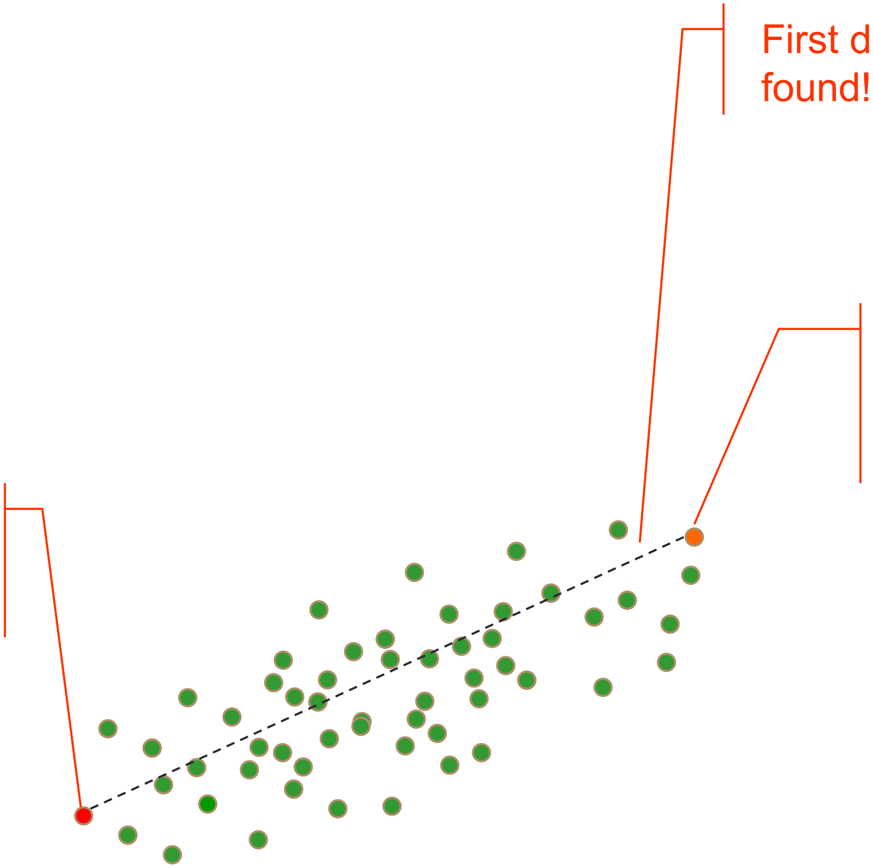
FastMap



Pick a random object

Pick the furthest object(pivot) O(N) time

FastMap

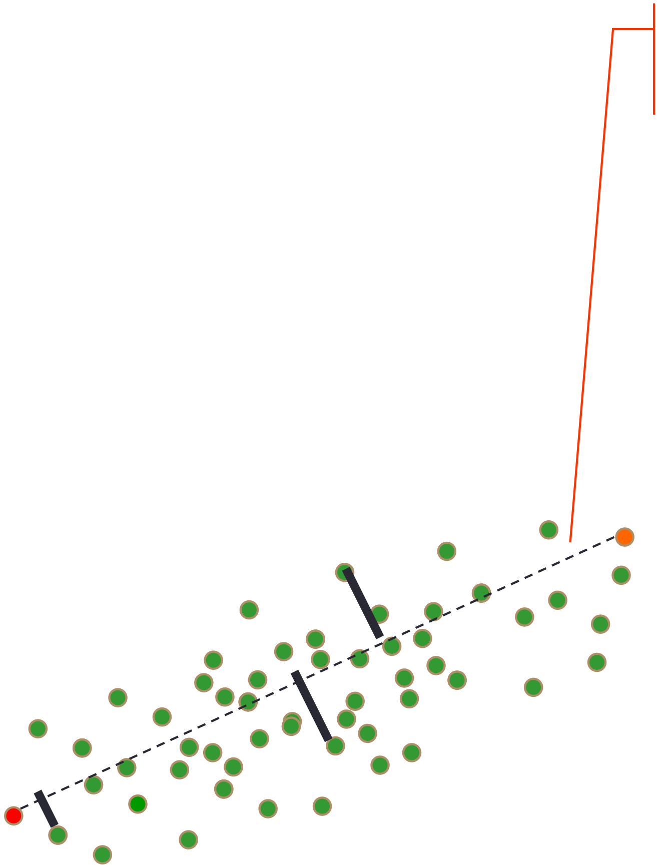


First dimension is found!!!

Pick the furthest object (pivot 1) O(N) time

Pick the furthest object (pivot 2) O(N) time

FastMap

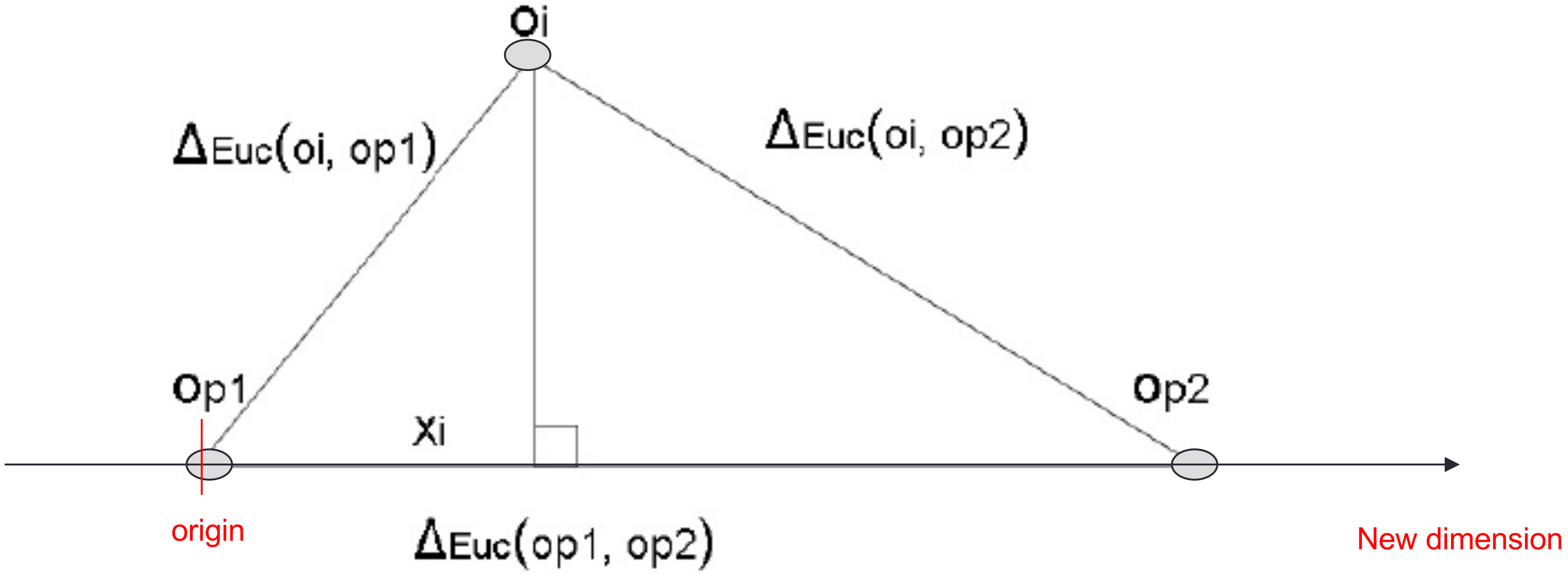


First dimension is found!!!

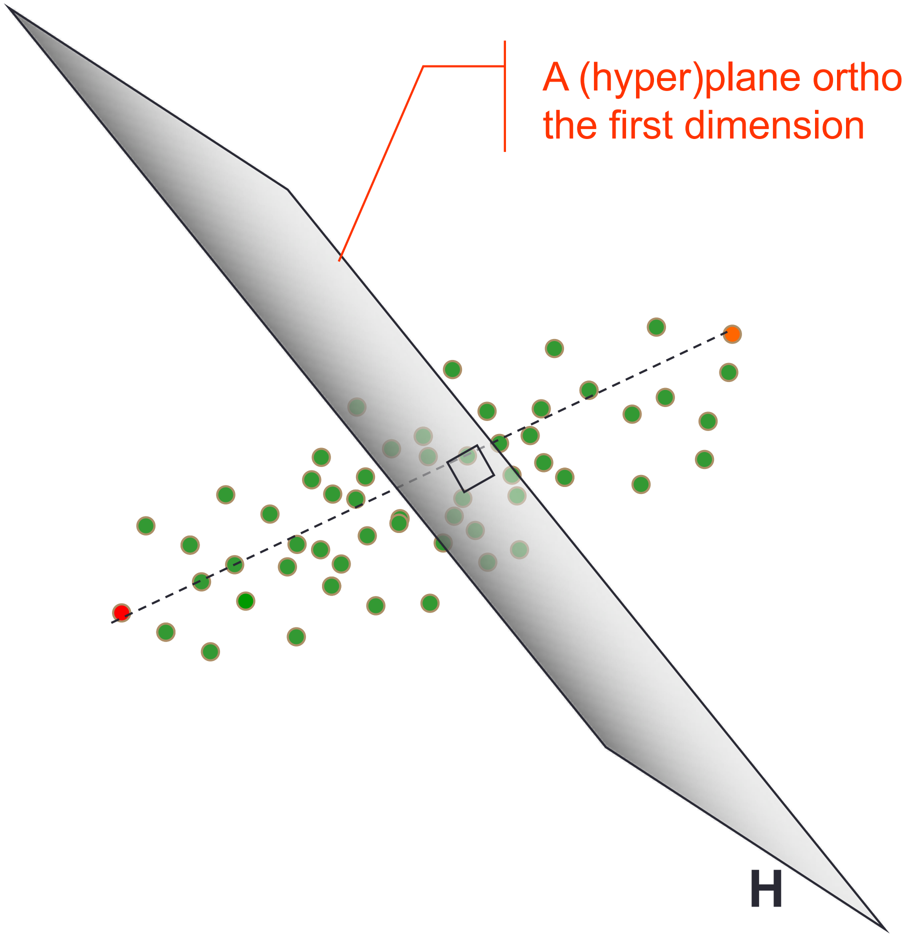
How do we map the objects in the database onto the line to find their position along this dimension?

$$x_i = \frac{(\Delta_{Euc}(o_i, o_{p1}))^2 - (\Delta_{Euc}(o_i, o_{p2}))^2 + (\Delta_{Euc}(o_{p1}, o_{p2}))^2}{2\Delta_{Euc}(o_{p1}, o_{p2})}$$

How do we find distances in the projected space?

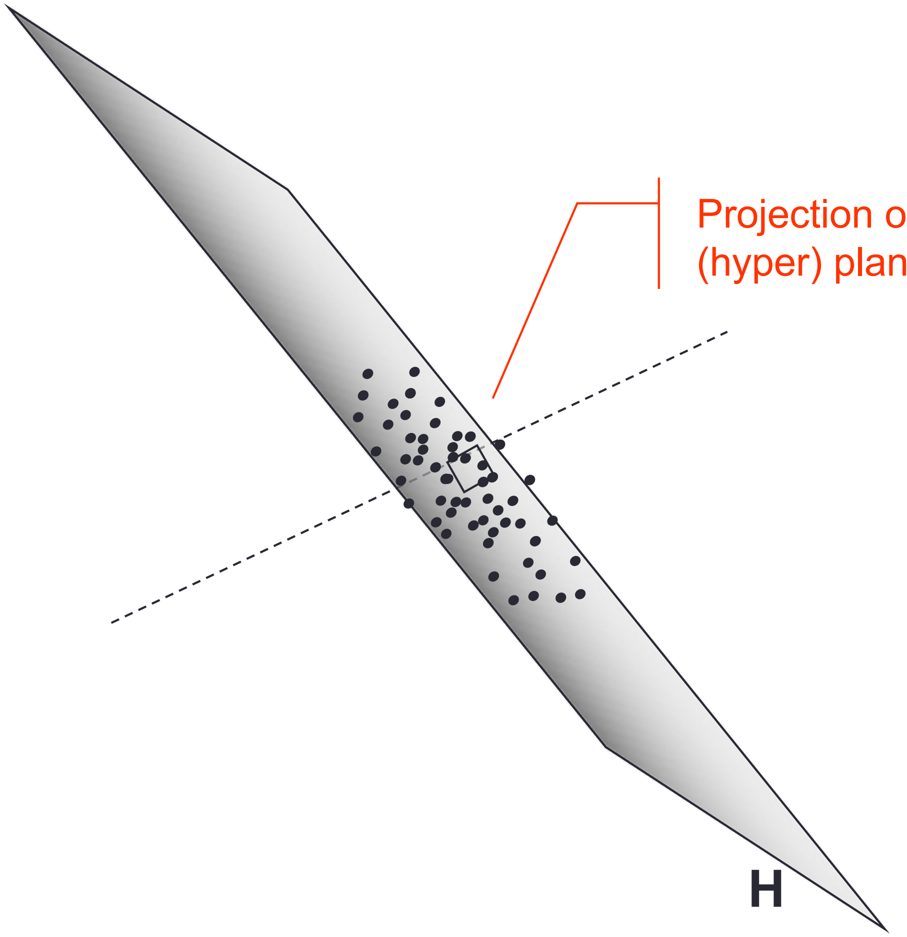


FastMap



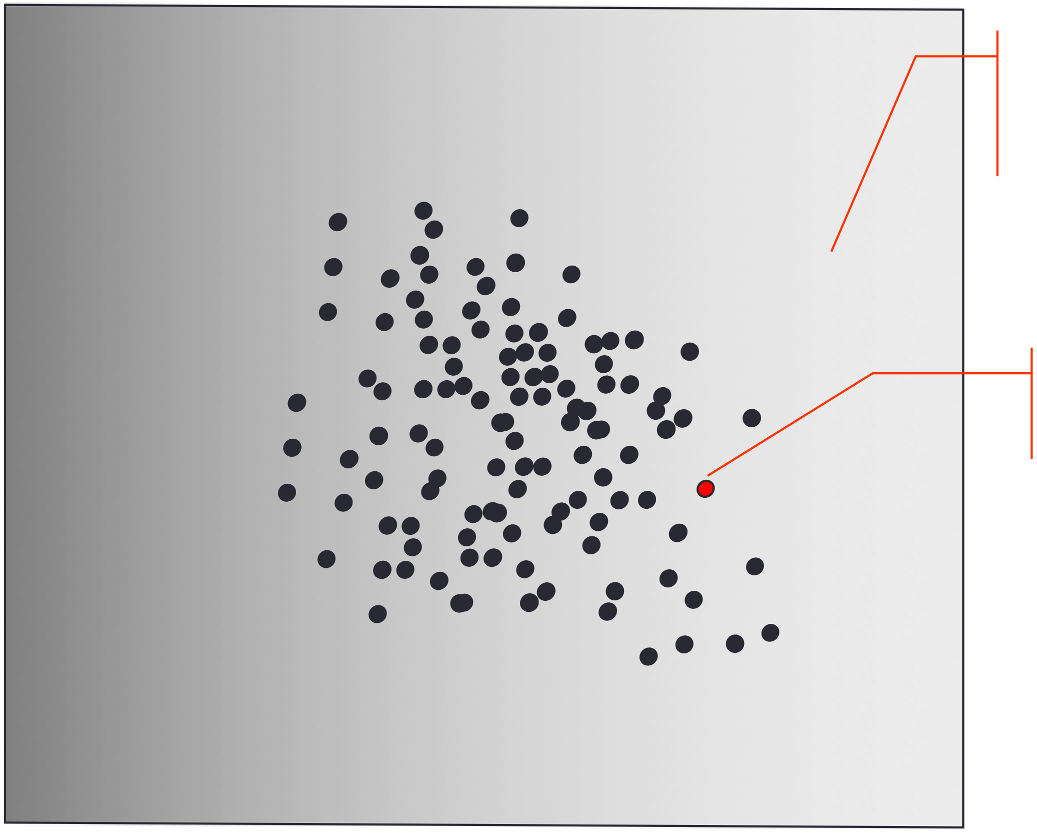
A (hyper)plane orthogonal to the first dimension

FastMap



Projection of the data to the (hyper) plane

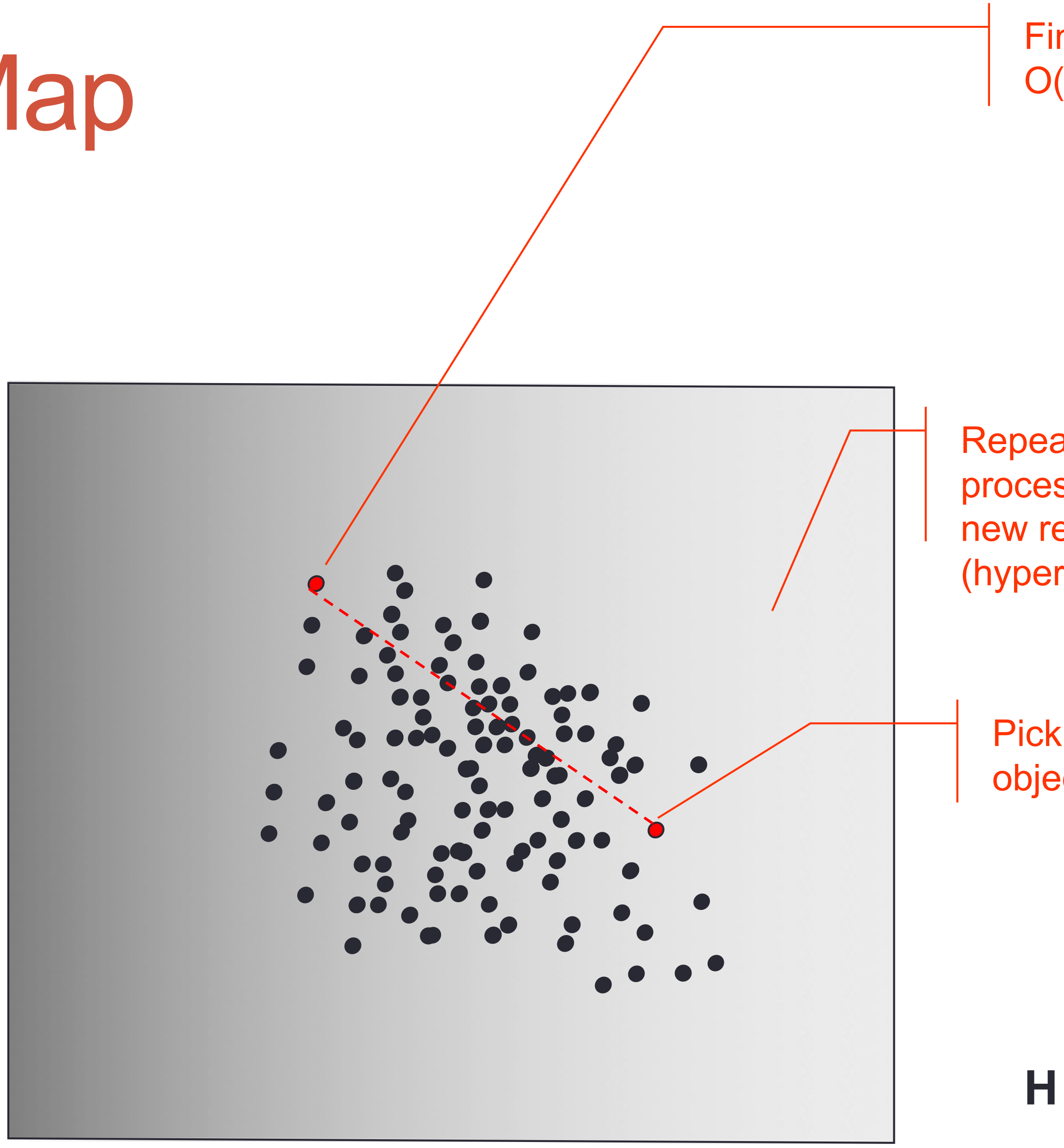
FastMap



Repeat the process in this new reduced (hyper-) space

Pick a random object

FastMap

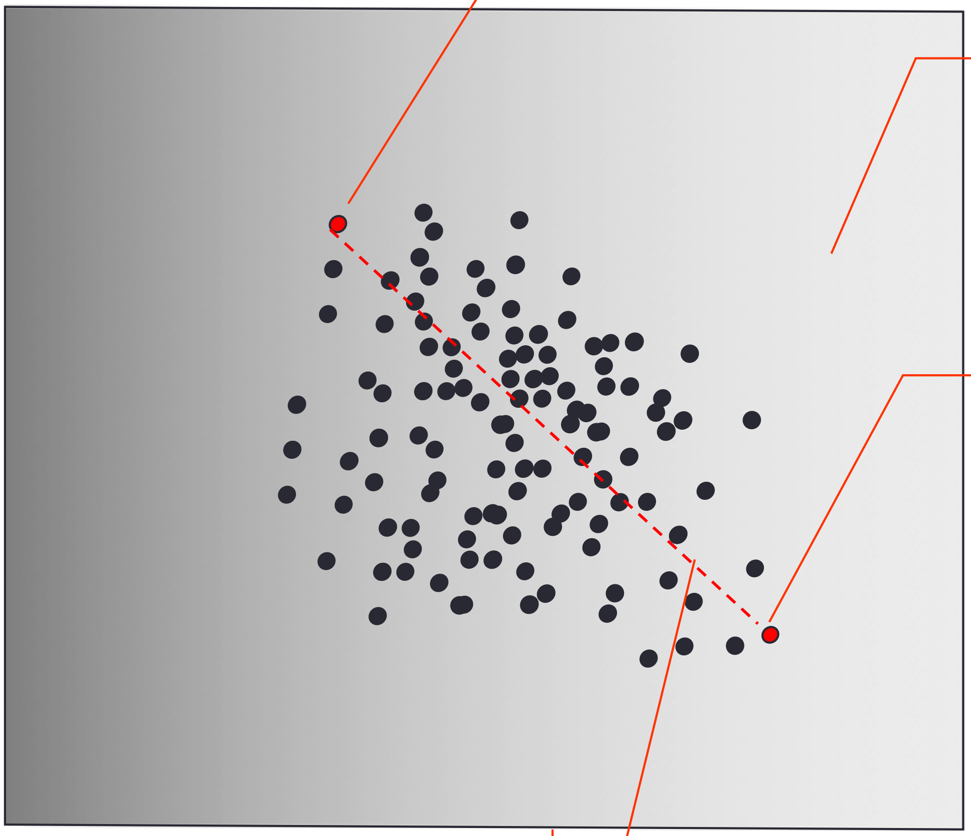


Find the furthest object O(N) time

Repeat the process in this new reduced (hyper-) space

Pick a random object

FastMap



Find the furthest object
O(N) time

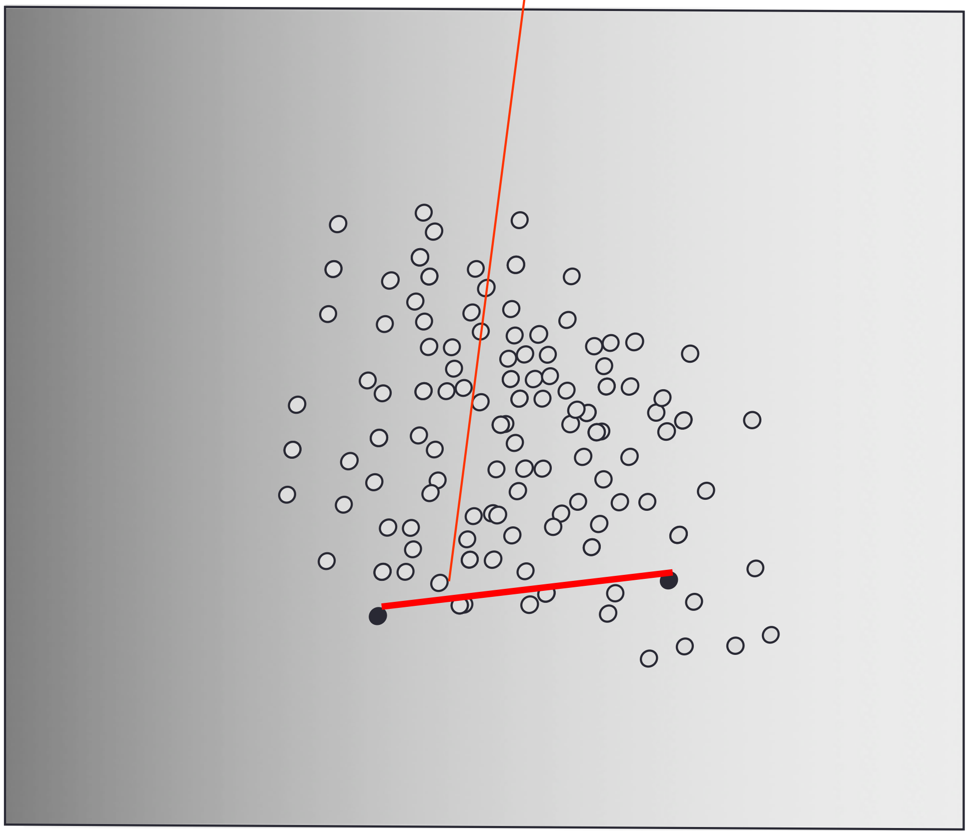
Repeat the
process in this
new reduced
(hyper-) space

Find the furthest object
O(N) time

Second dimension
(orthogonal to the first
one!)

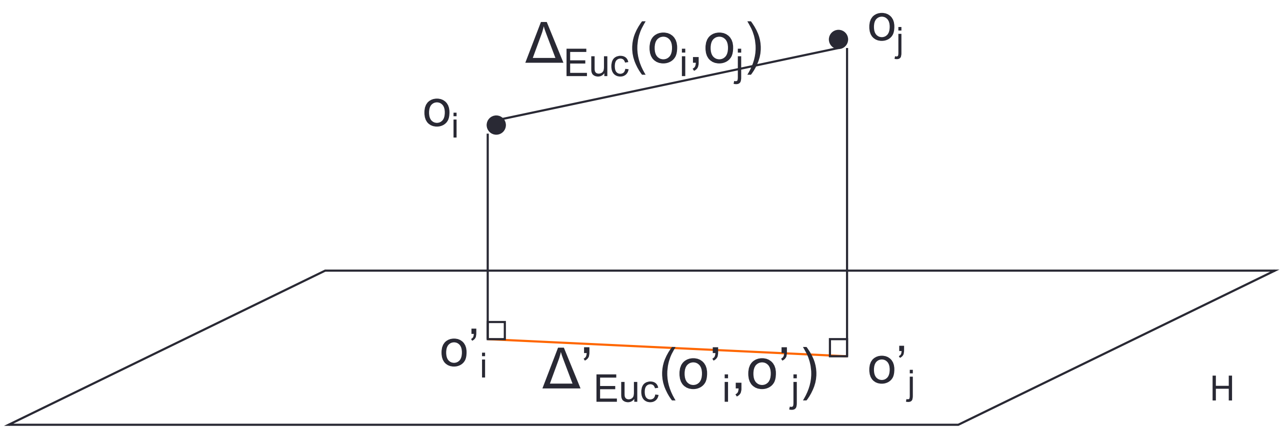
FastMap

How do we measure the distances in this new projected space



How do we find distances in the projected space?

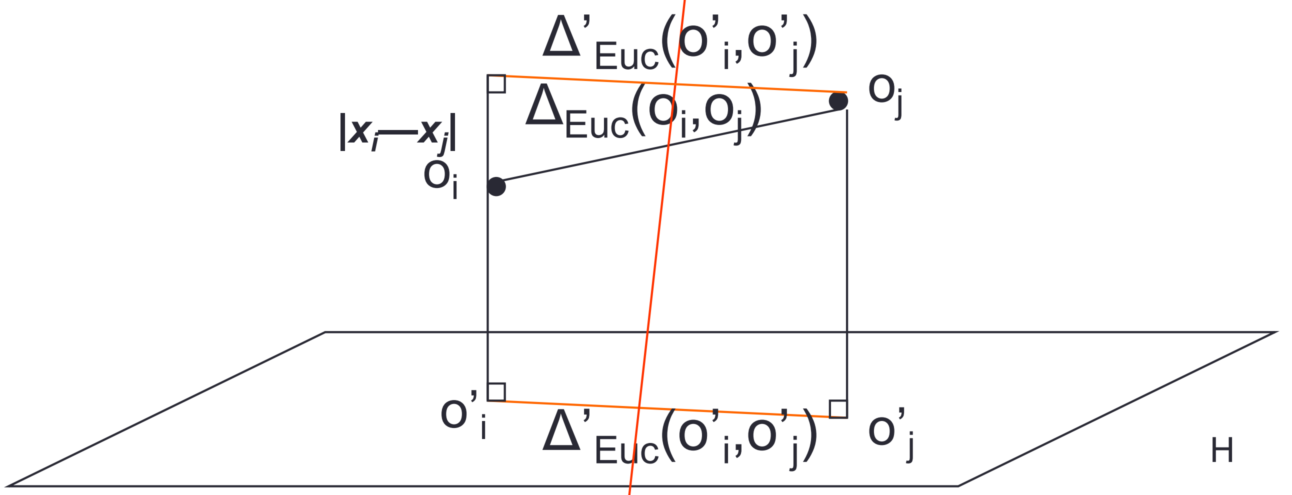
- Leverage metric space properties!



..already available..

How do we find distances in the projected space?

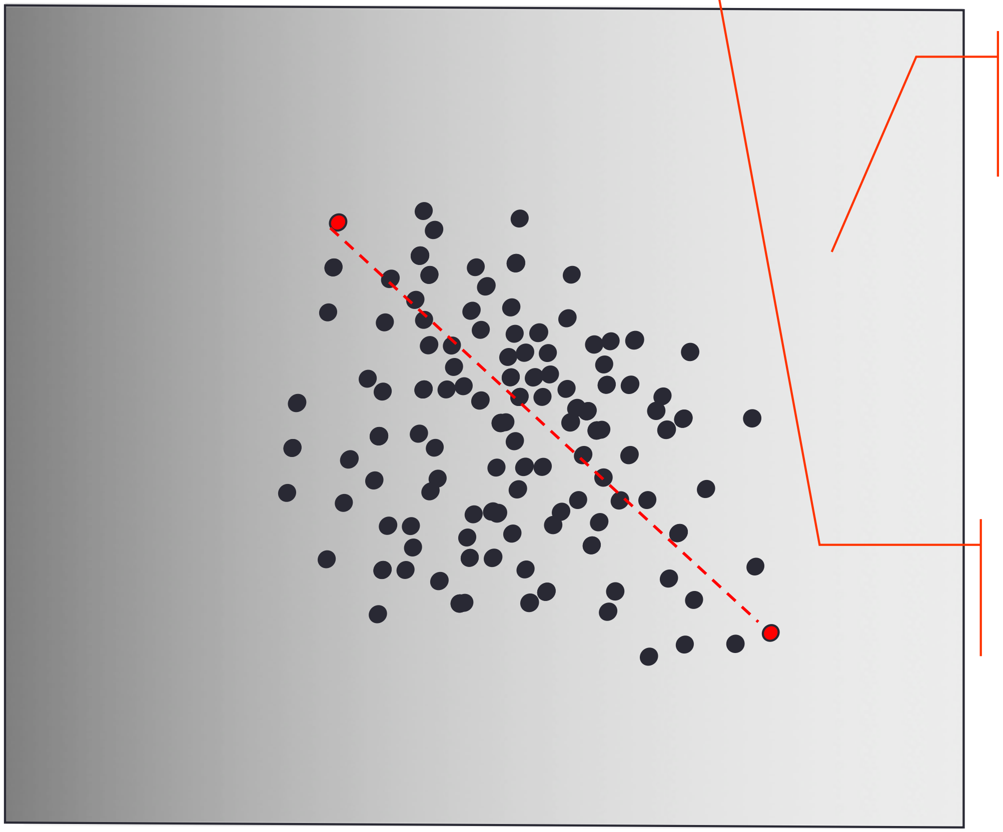
- Leverage metric space properties!



$$(\Delta'_{Euc}(o'_i, o'_j))^2 = (\Delta_{Euc}(o_i, o_j))^2 - (x_i - x_j)^2$$

FastMap

Each iteration takes O(N) times



Repeat in this new reduced (hyper-) space

Total execution time O(kN)

H