OPTIMAL SETS OF PROJECTIONS OF HIGH-DIMENSIONAL DATA

DIRK J. LEHMANN, HOLGER THEISEL. TVCG 22(1) 2016
PRESENTED BY JASON HARTFORD

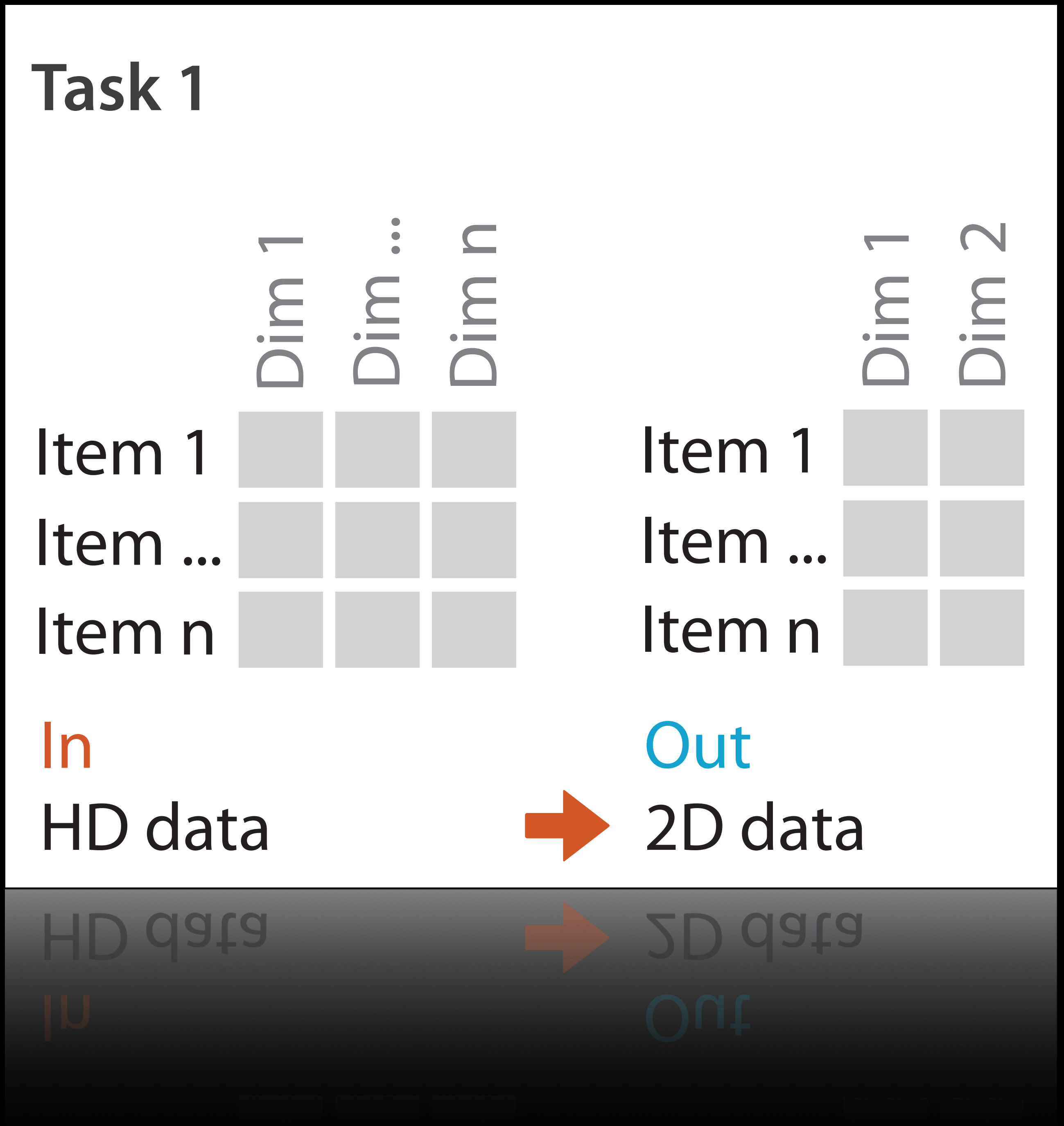
WHAT?

Recall **REDUCE** task:

- **In:** HD Data
- **Out:** 2D projection

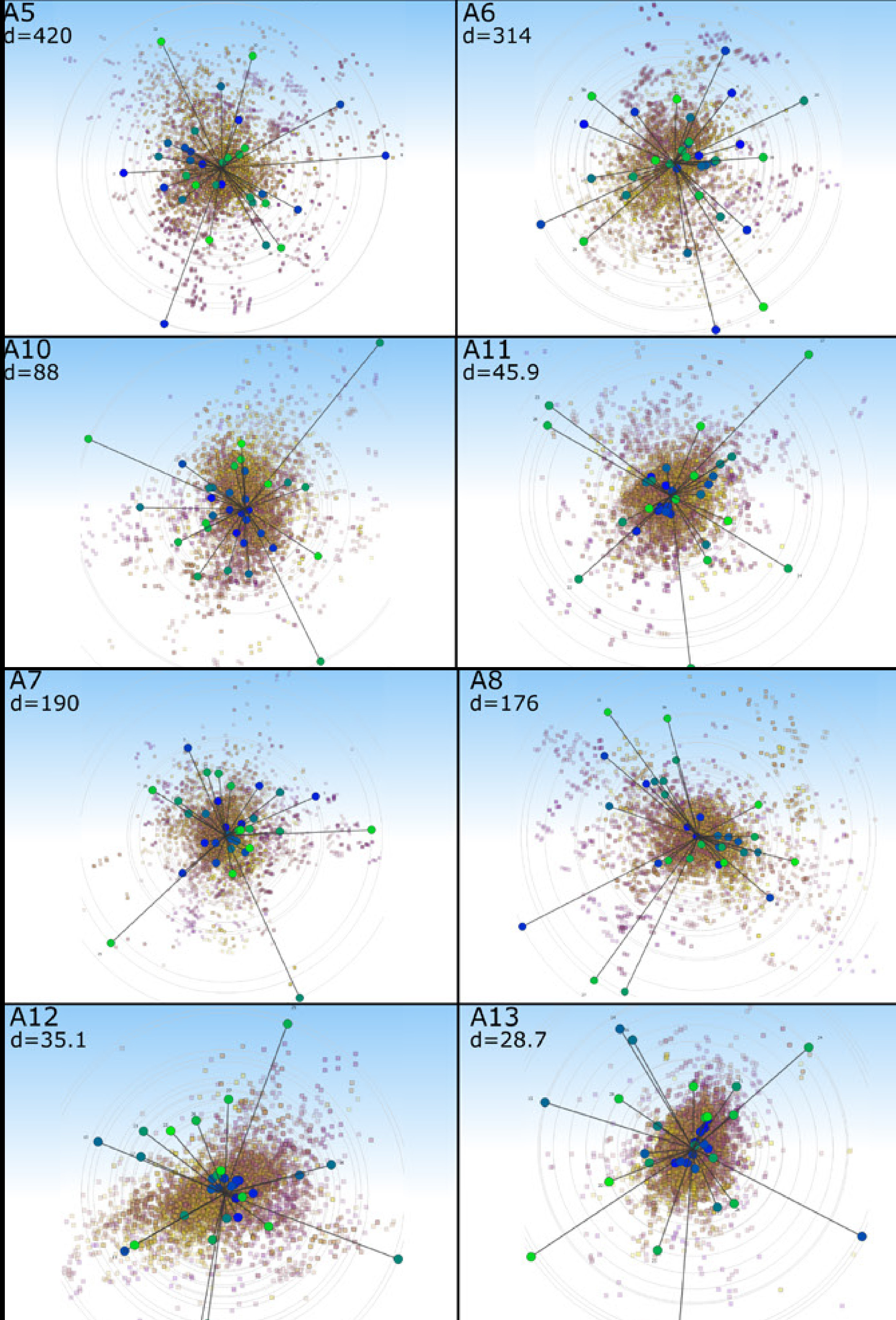
Today's paper:

- **In:** HD Data
- **Out:** "optimal" set of **2D projections**



WHY?

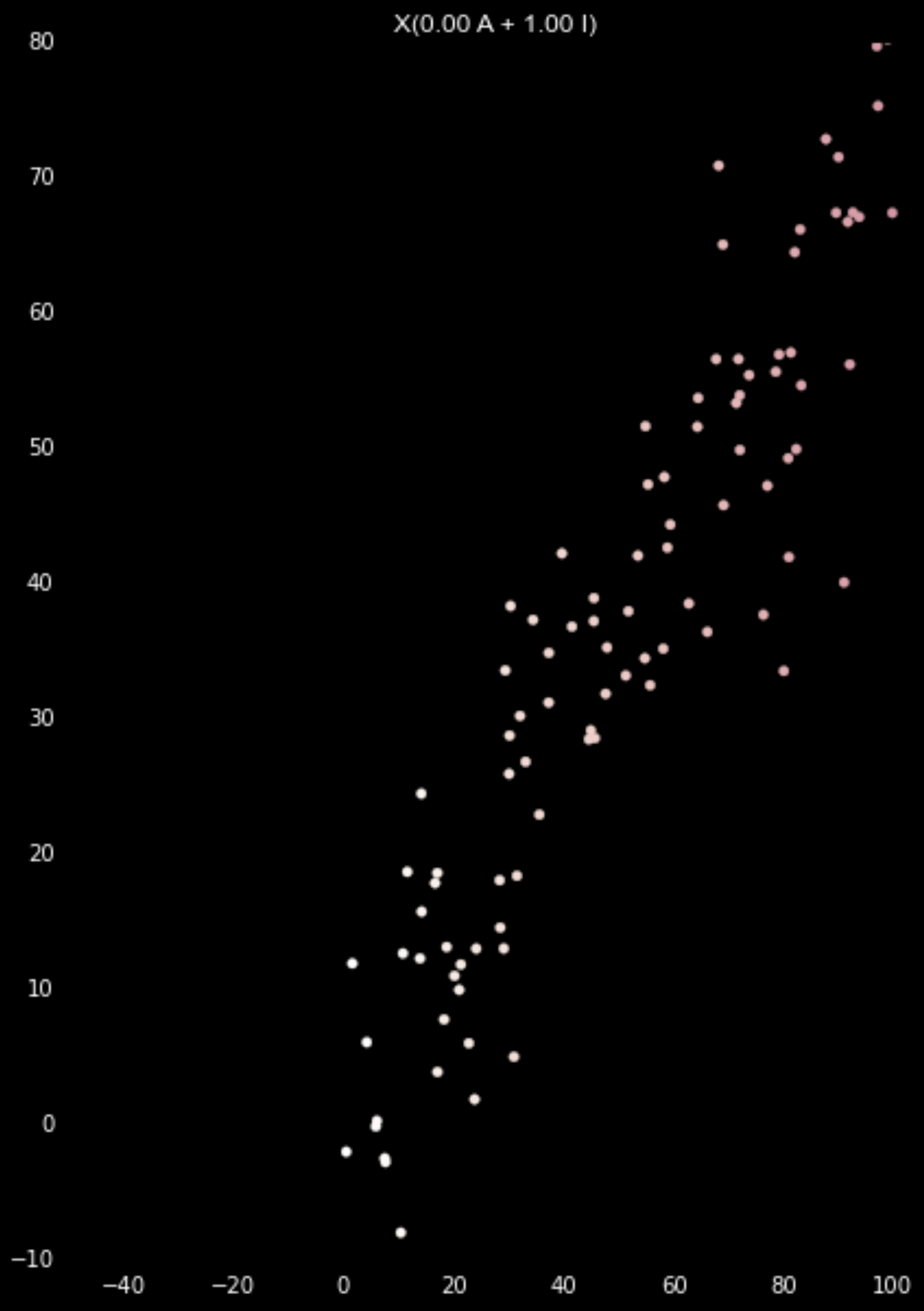
- Large space of potential projections
- Would like to find a minimal set of "interesting" projections to describe our dataset



HOW?

Core assumption:

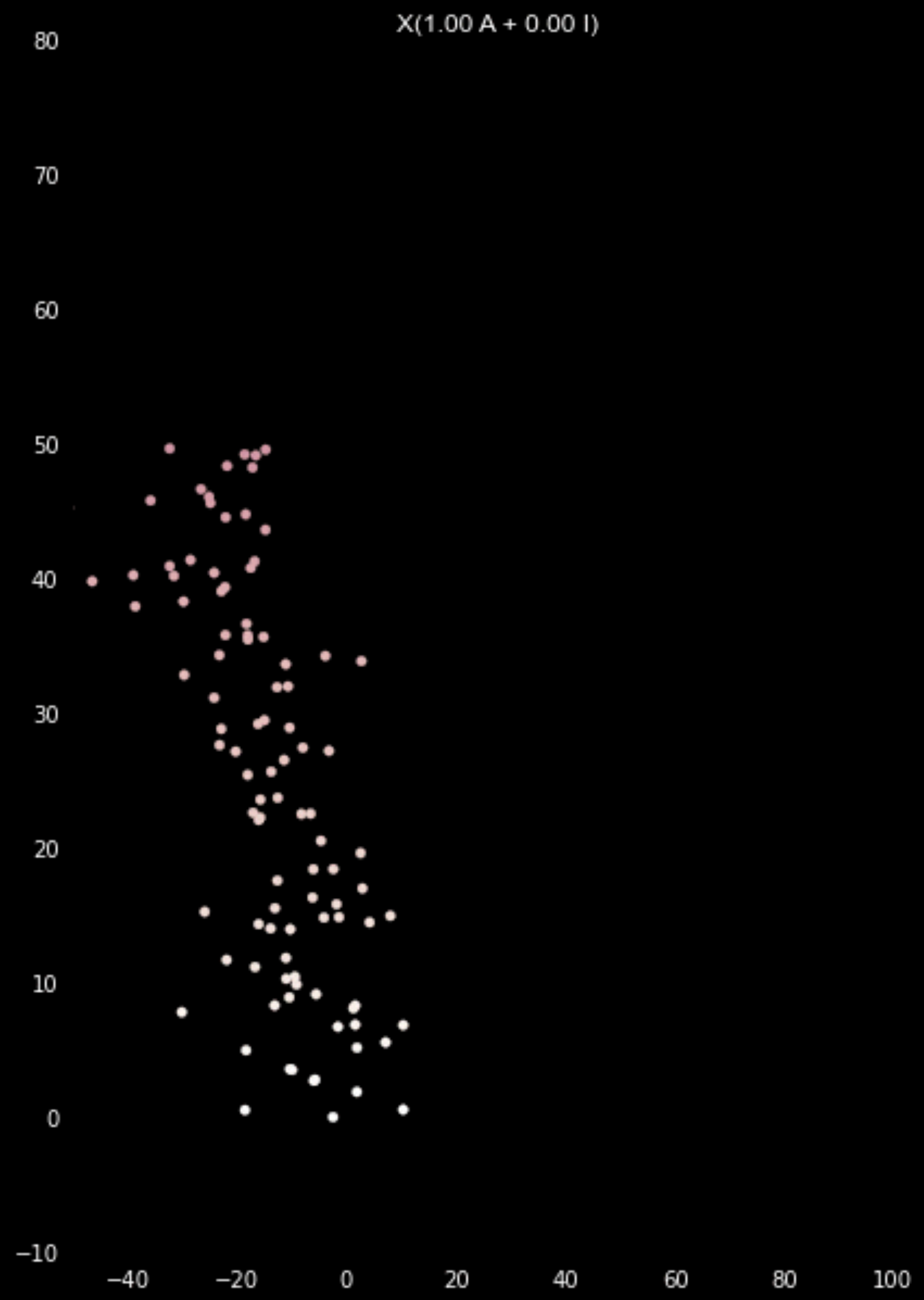
- Assume projections only provide **insight** if they're not equivalent up to an **affine** map.



HOW?

Core assumption:

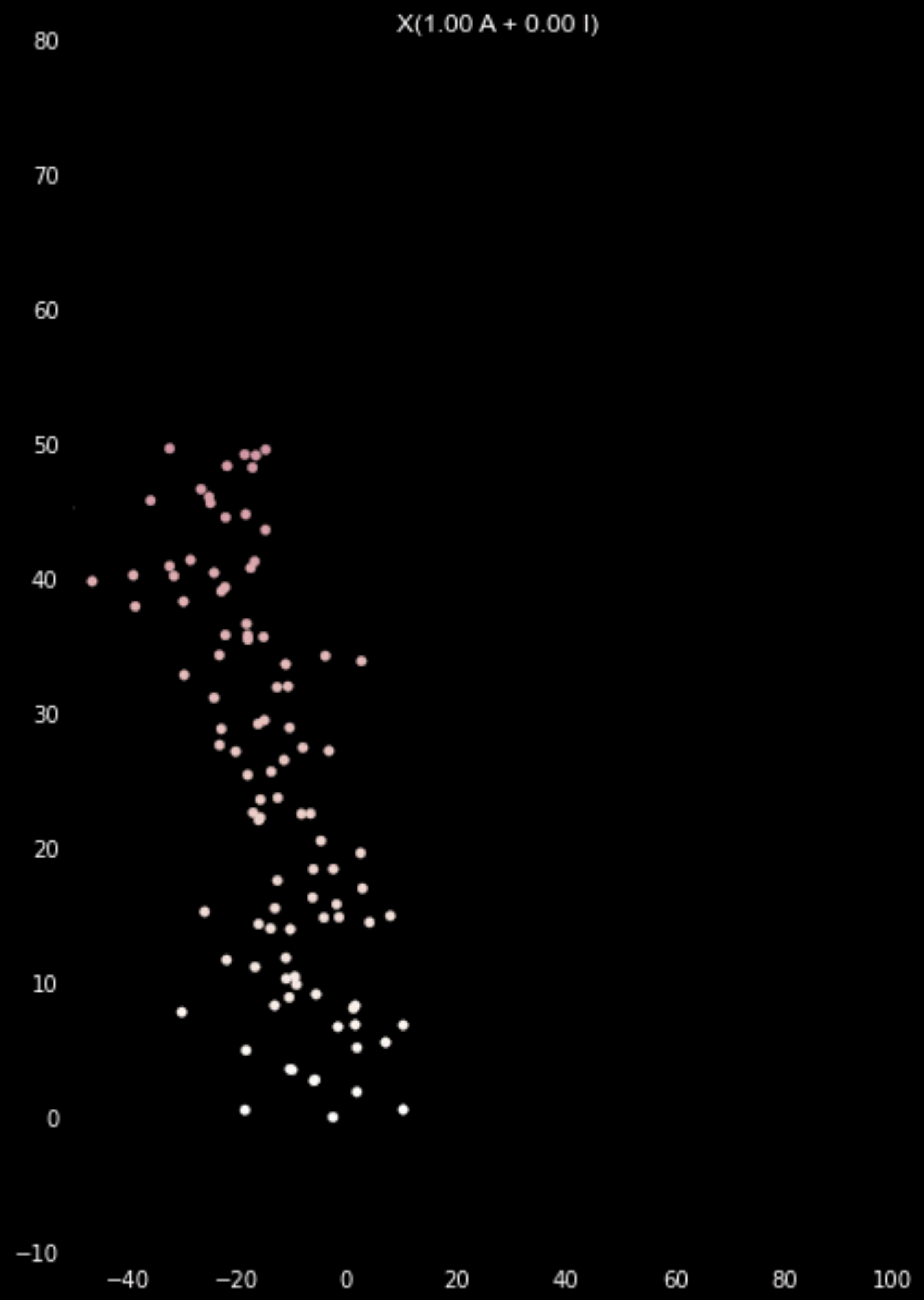
- Assume projections only provide **insight** if they're not equivalent up to an **affine** map.



HOW?

Core assumption:

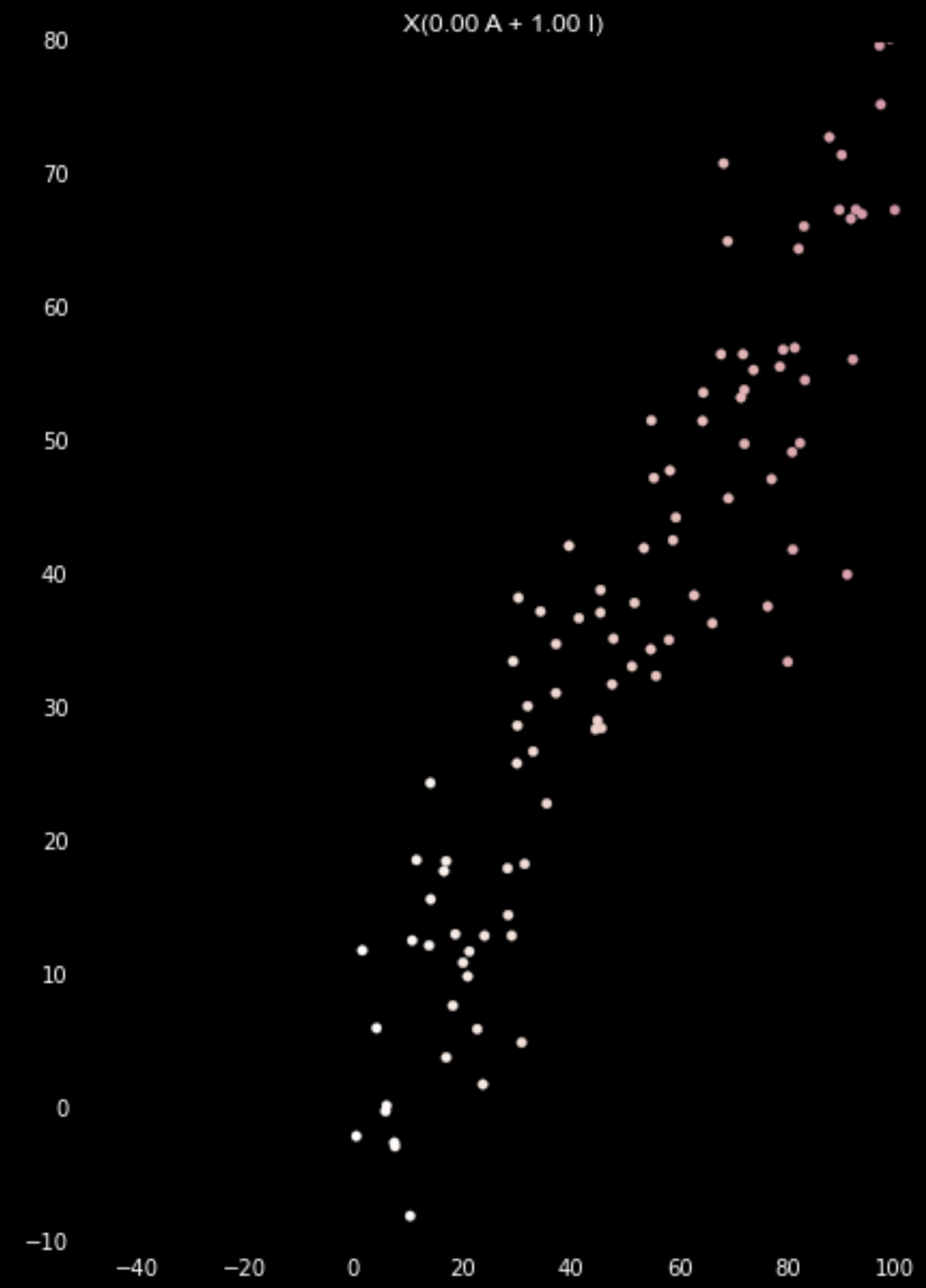
- Assume projections only provide **insight** if they're not equivalent up to an **affine** map.



HOW?

Core assumption:

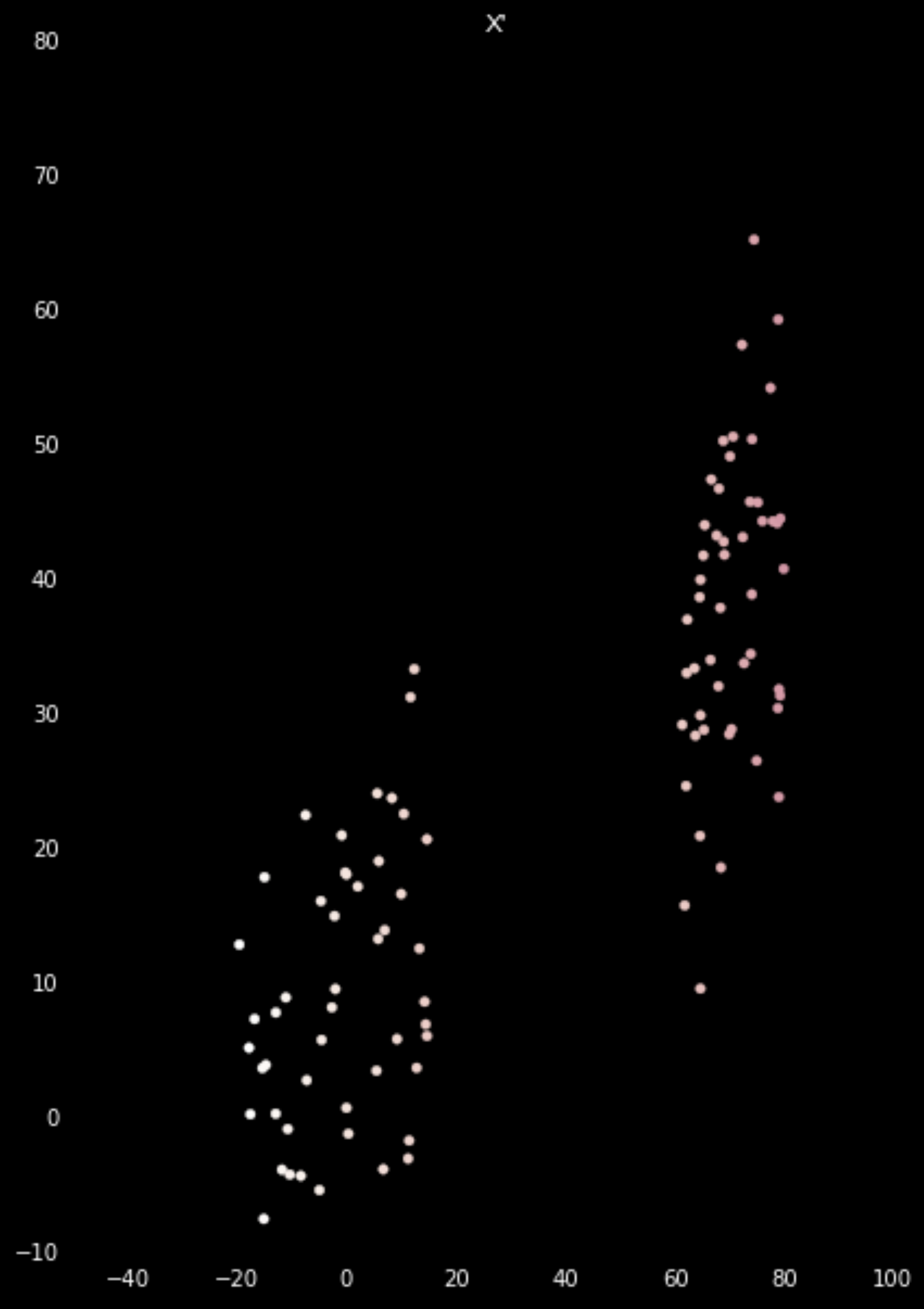
- Assume projections only provide **insight** if they're not equivalent up to an **affine** map.

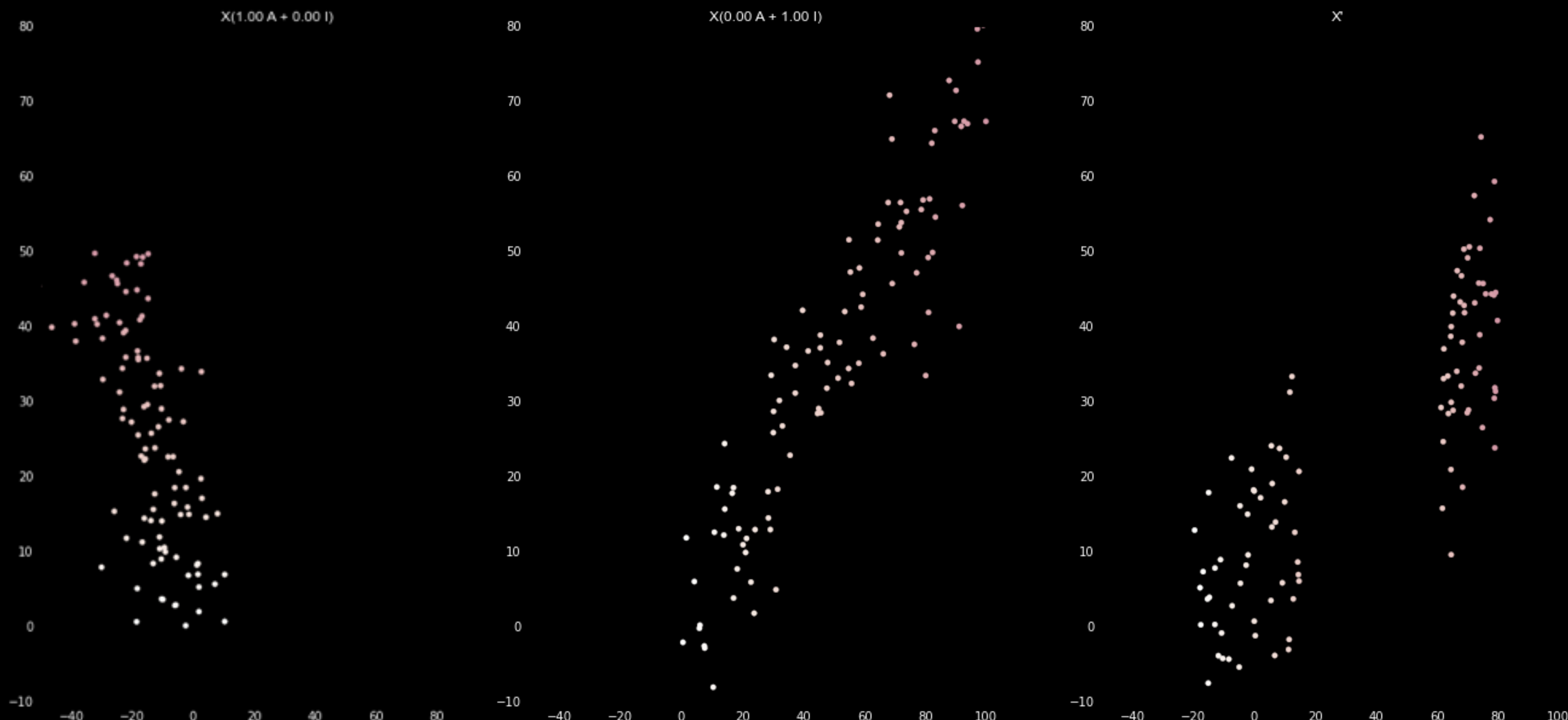


HOW?

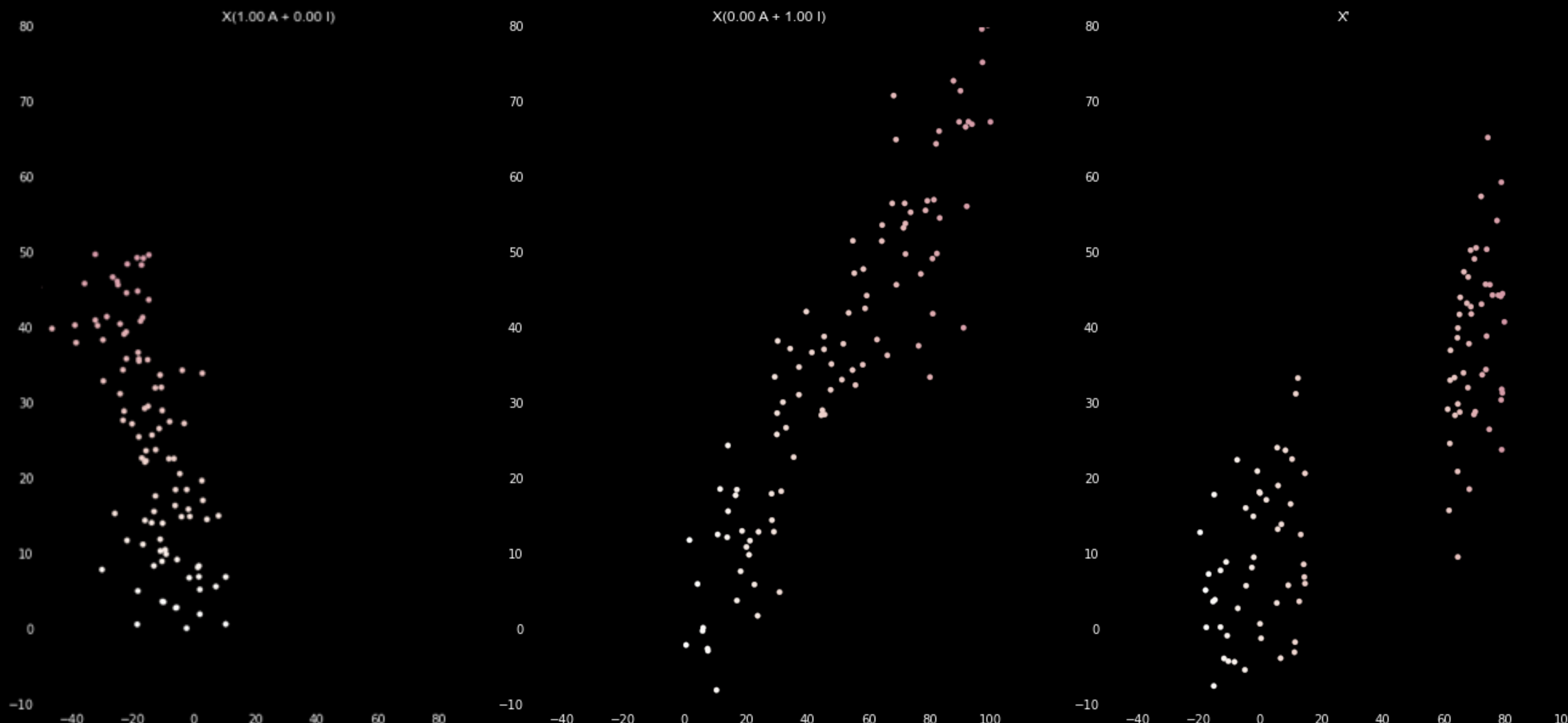
Core assumption:

- Assume projections only provide **insight** if they're not equivalent up to an **affine** map.

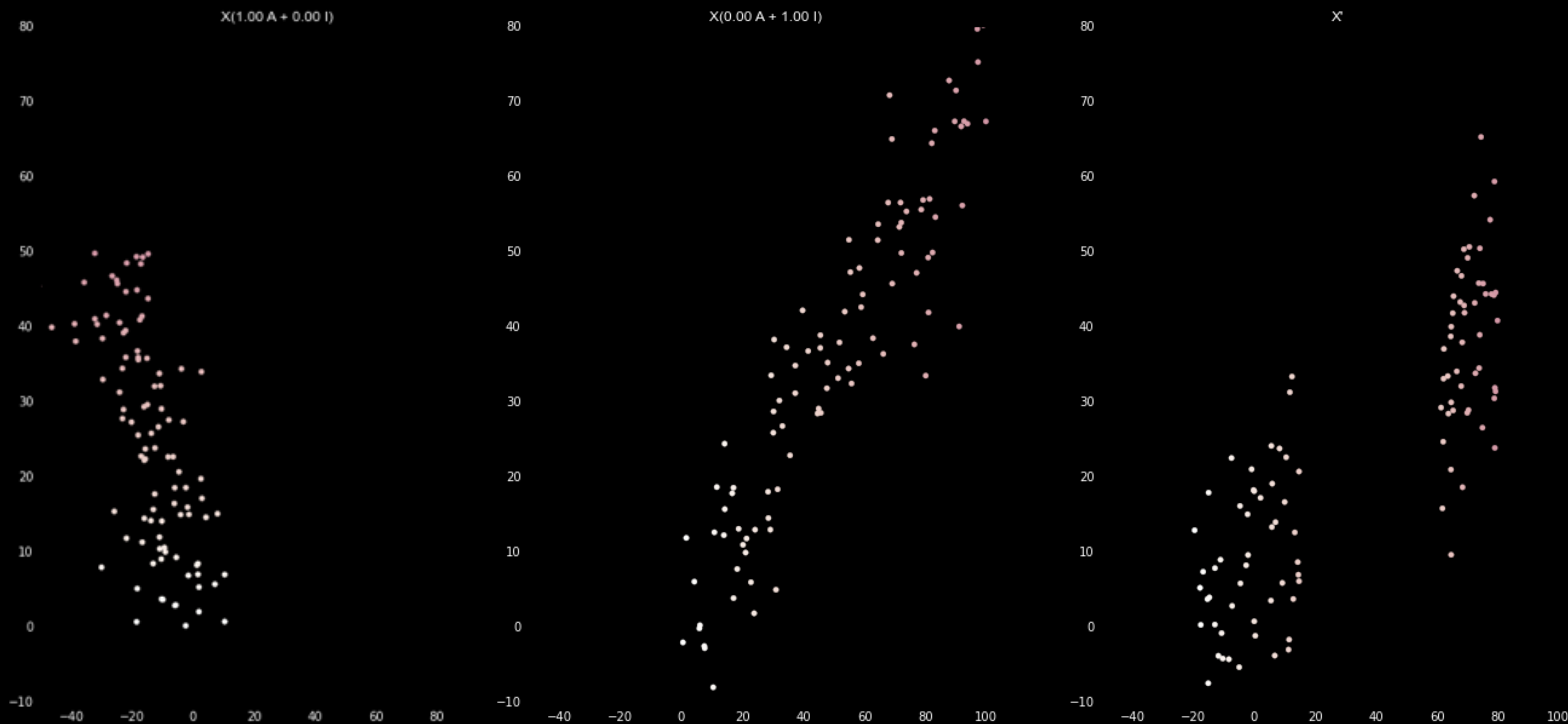



X(1.00 A + 0.00 I)
X(0.00 A + 1.00 I)
X'
80
70
60
50
40
30
20
10
0
−10
−40
−20
0
20
40
60
80
100

$A_1 \approx A_2 \neq A_3$



$$A_1 \approx A_2 \neq A_3$$



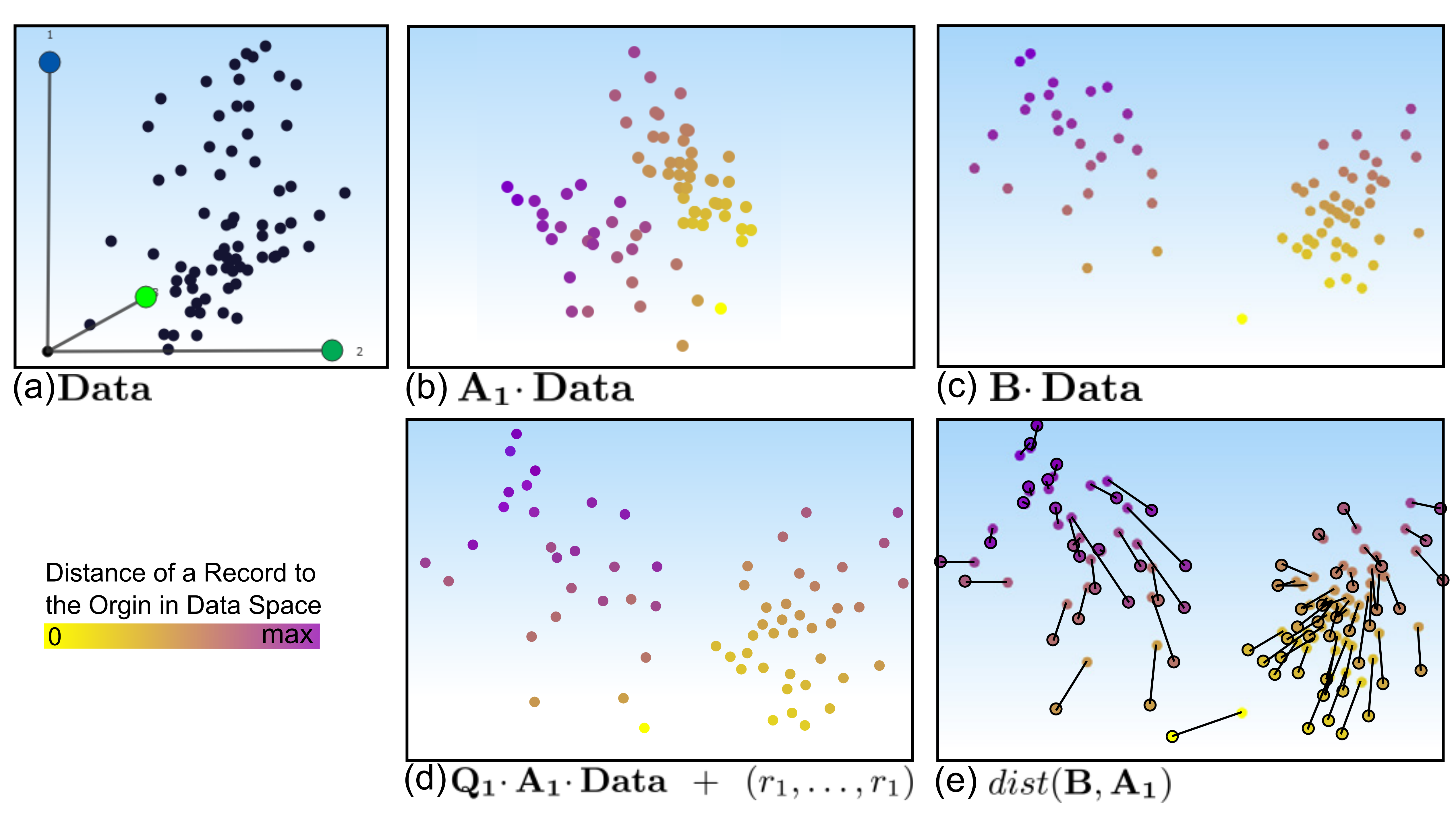
$$d(A_1, A_2) = 0 \qquad d(A_1, A_3) > 0$$

ALGORITHM - HIGH LEVEL

- At iteration i, given set of projections **A** = $\{A_0, \ldots, A_{i-1}\}$
- Greedily find linear projection B that is most dissimilar from the projections in **A**
- Add A_i = B to our set of projections
- Repeat until the best new projection gives no new insight (equivalent up to an affine transformation)

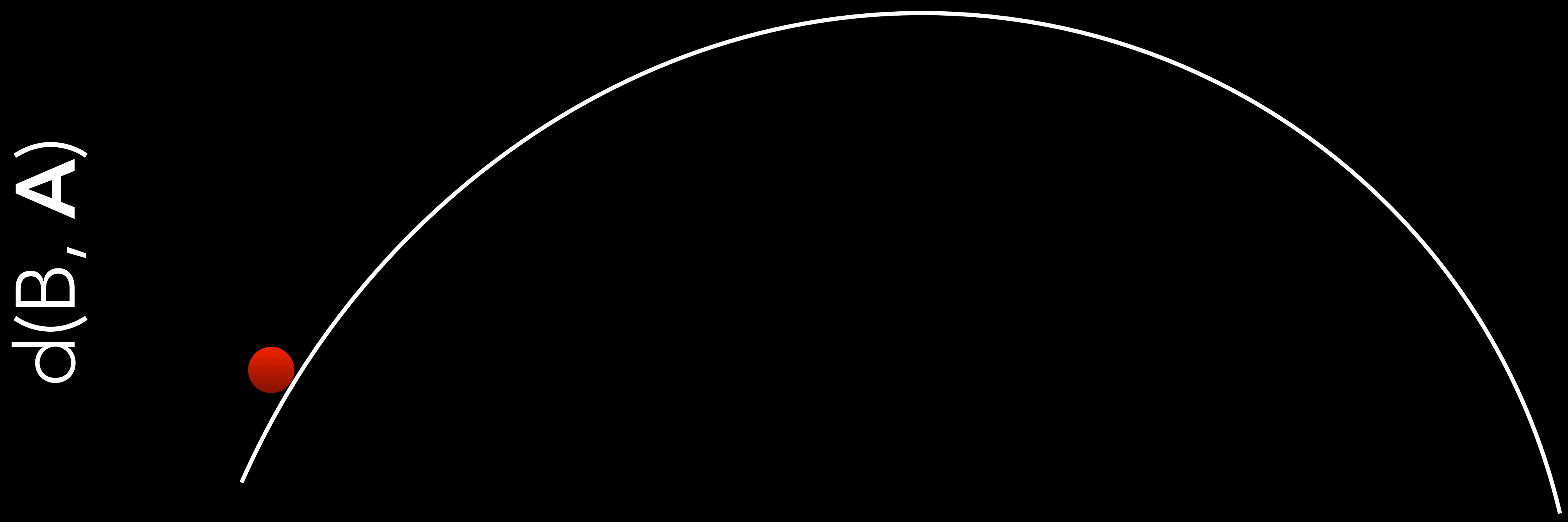
The devil's in the details….

MEASURING DISSIMILARITY



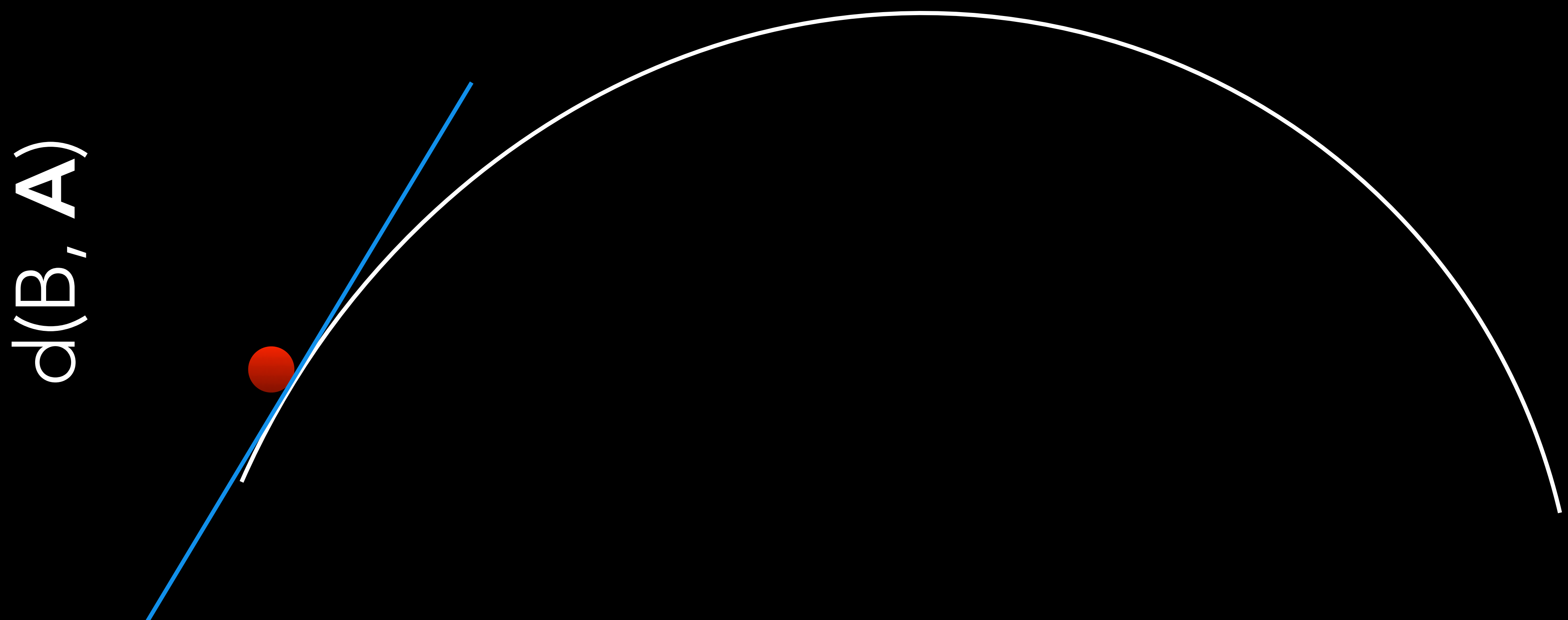
FINDING THE "MOST DISSIMILAR" PROJECTION

- Given **A** = $\{A_0, \ldots, A_{i-1}\}$ start by setting B = A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



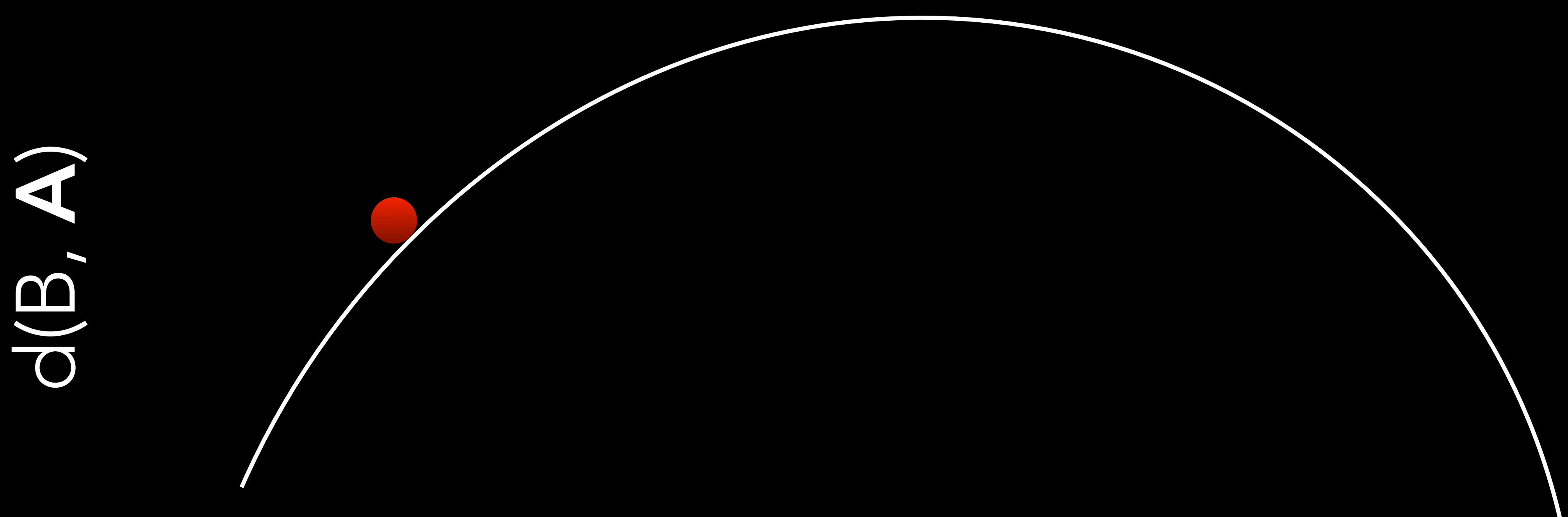
FINDING THE "MOST DISSIMILAR" PROJECTION

- Given **A** = {A_0, …, A_{i-1}} start by setting B = A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



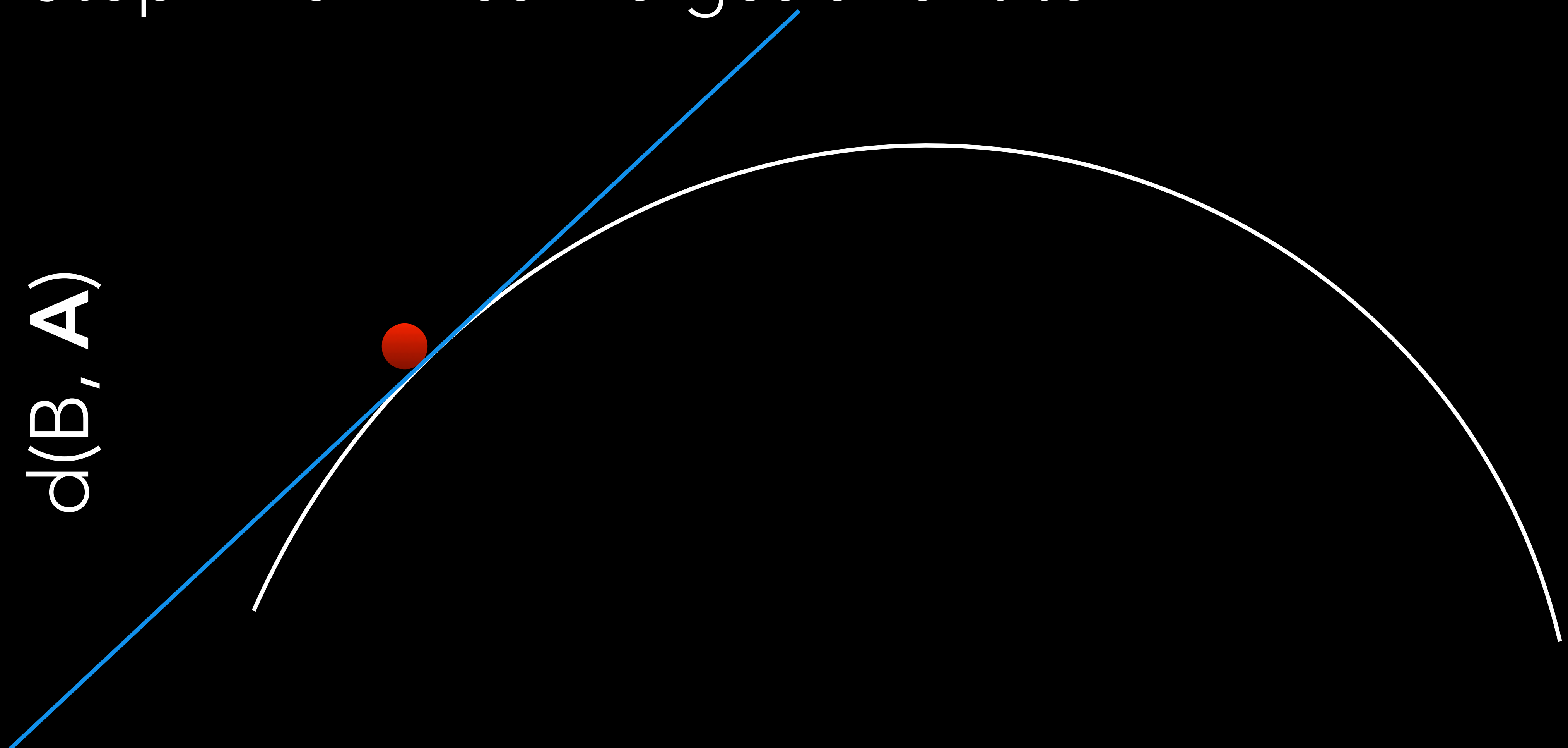
FINDING THE "MOST DISSIMILAR" PROJECTION

- Given **A** = $\{A_0, \ldots, A_{i-1}\}$ start by setting B = A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



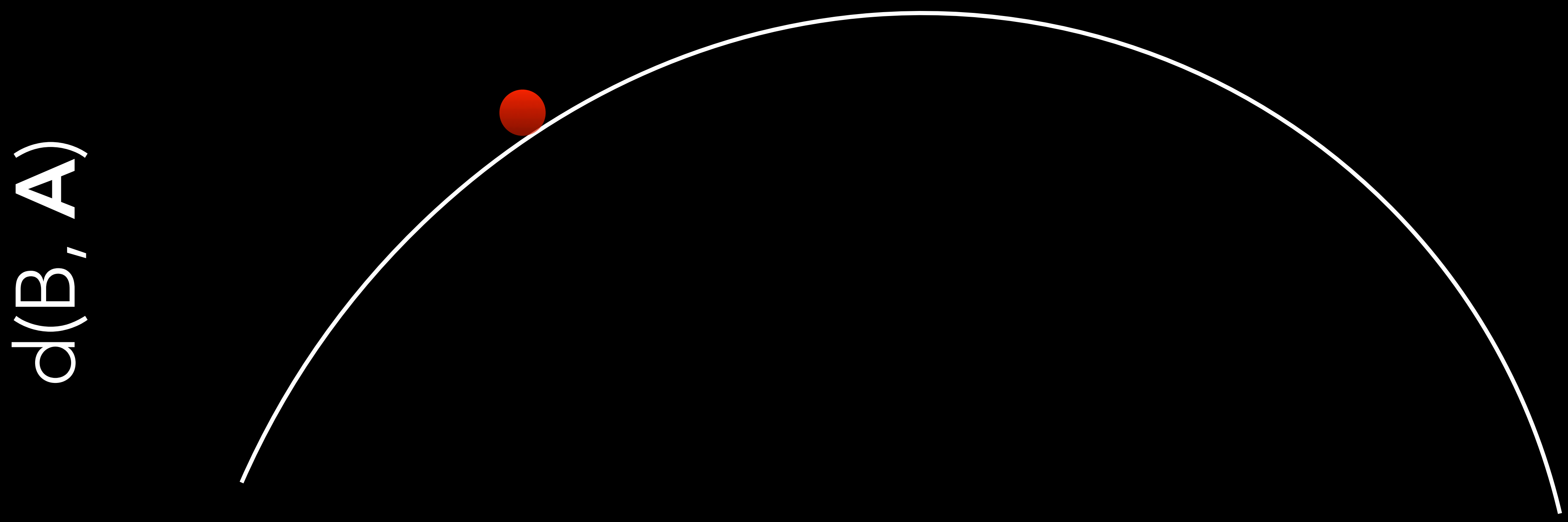
FINDING THE "MOST DISSIMILAR" PROJECTION

- Given **A** = $\{A_0, \ldots, A_{i-1}\}$ start by setting B = A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



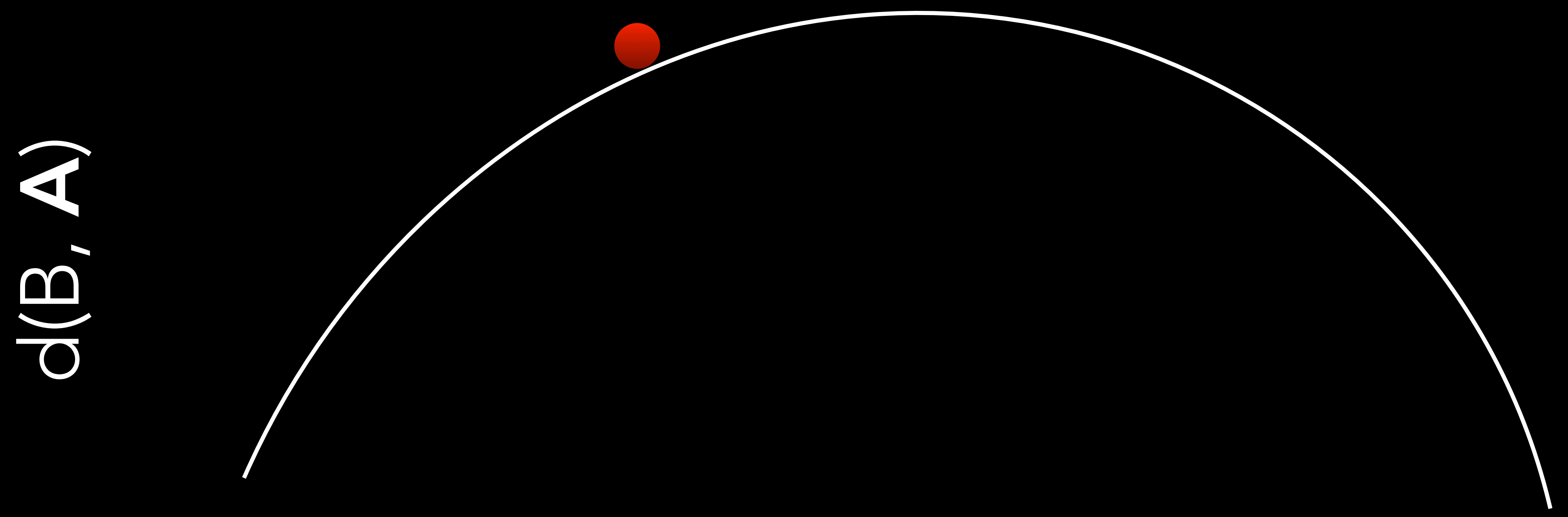
FINDING THE "MOST DISSIMILAR" PROJECTION

- Given **A** = {A_0, …, A_{i-1}} start by setting B = A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



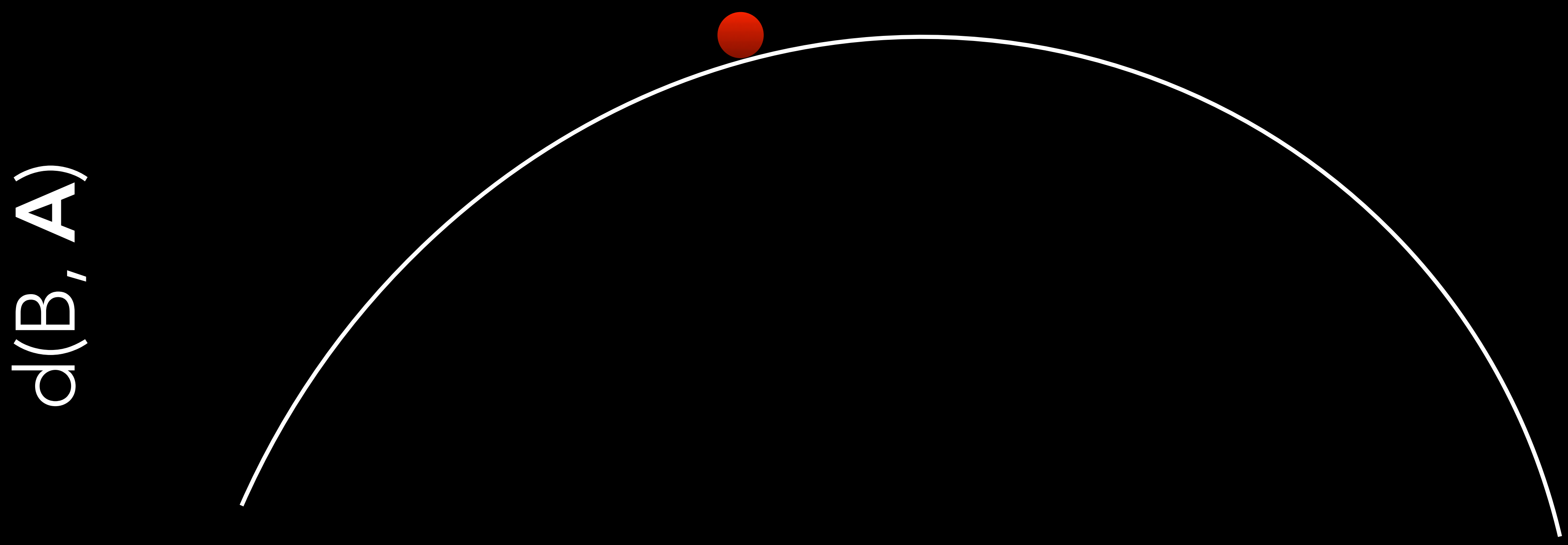
FINDING THE "MOST DISSIMILAR" PROJECTION

- Given **A** = {A_0, …, A_{i-1}} start by setting B = A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



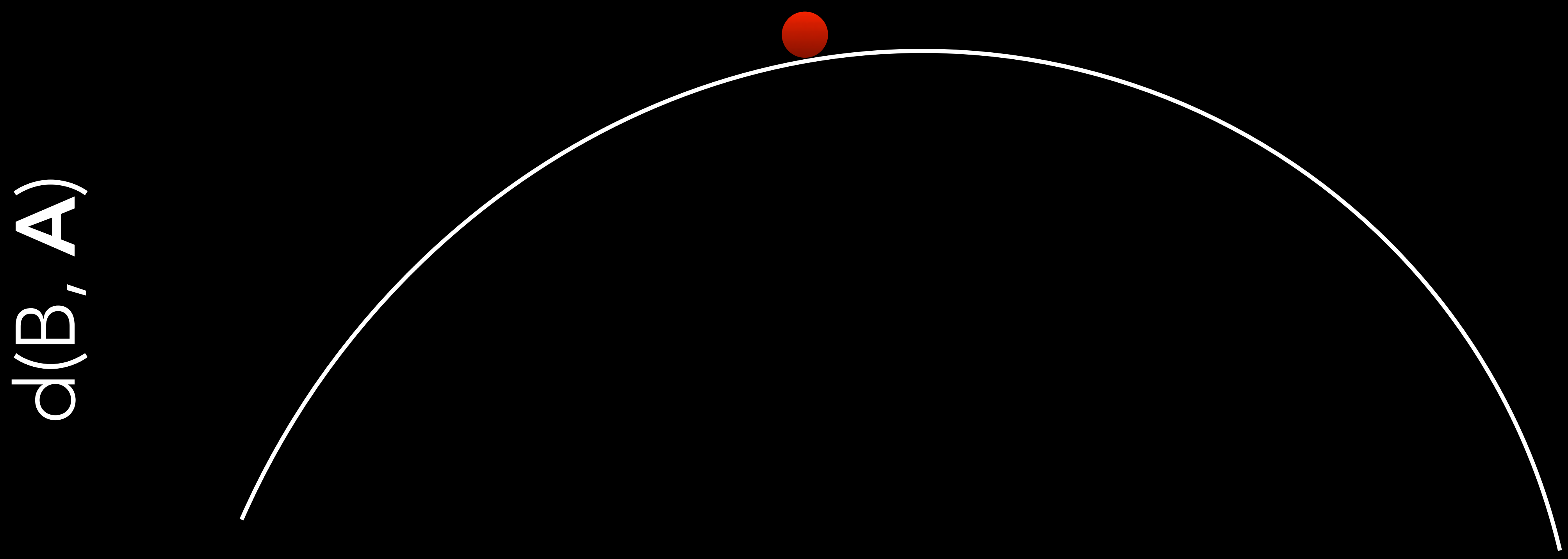
FINDING THE "MOST DISSIMILAR" PROJECTION

- Given **A** = {A_0, …, A_{i-1}} start by setting B = A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



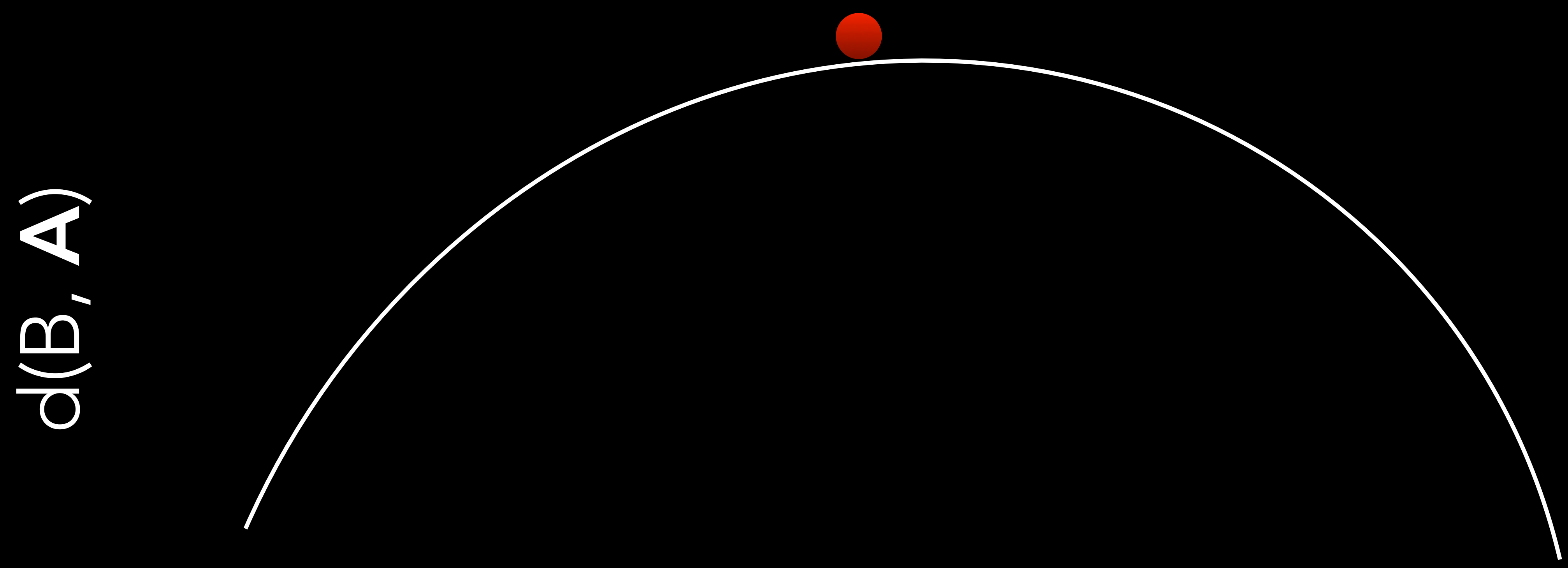
FINDING THE "MOST DISSIMILAR" PROJECTION

- Given **A** = $\{A_0, \ldots, A_{i-1}\}$ start by setting B = A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



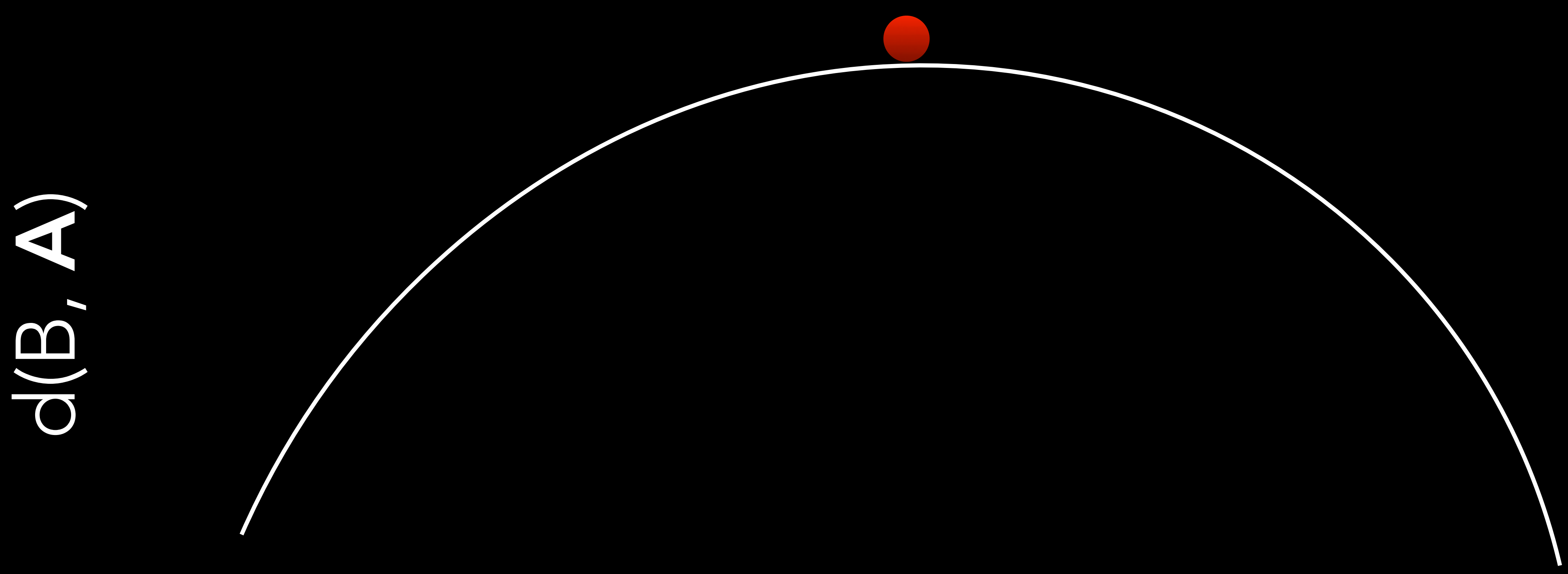
FINDING THE "MOST DISSIMILAR" PROJECTION

- Given **A** = $\{A_0, \ldots, A_{i-1}\}$ start by setting B = A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



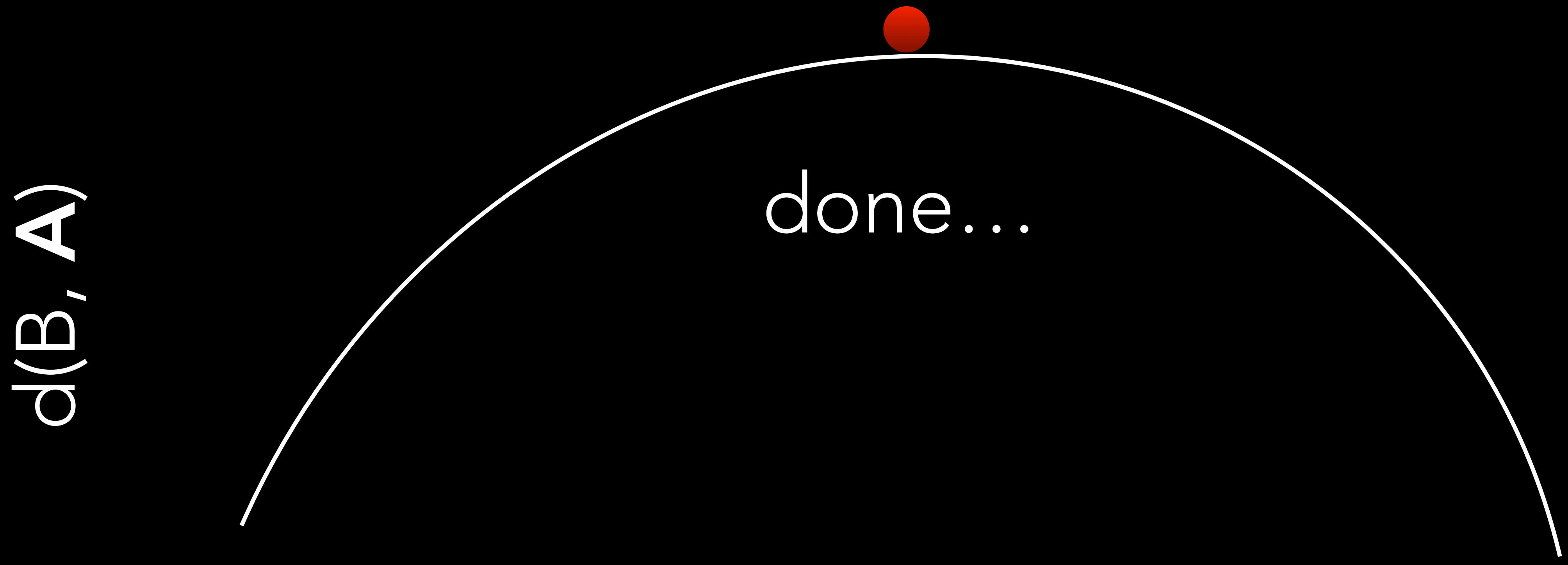
FINDING THE "MOST DISSIMILAR" PROJECTION

- Given **A** = {A_0, …, A_{i-1}} start by setting B $=$ A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



FINDING THE "MOST DISSIMILAR" PROJECTION

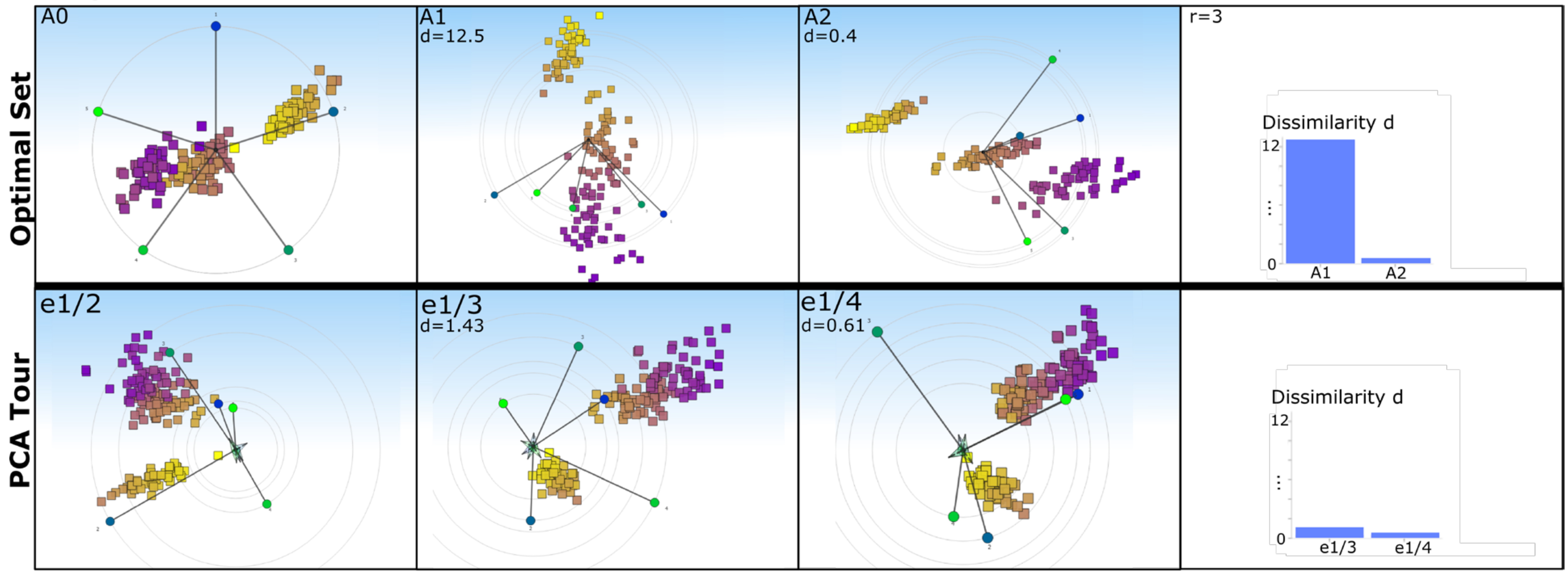
- Given **A** = {A_0, …, A_{i-1}} start by setting B = A_{i-1}.
- Apply gradient ascent to increase the dissimilarity
- Stop when B converges and it to **A**



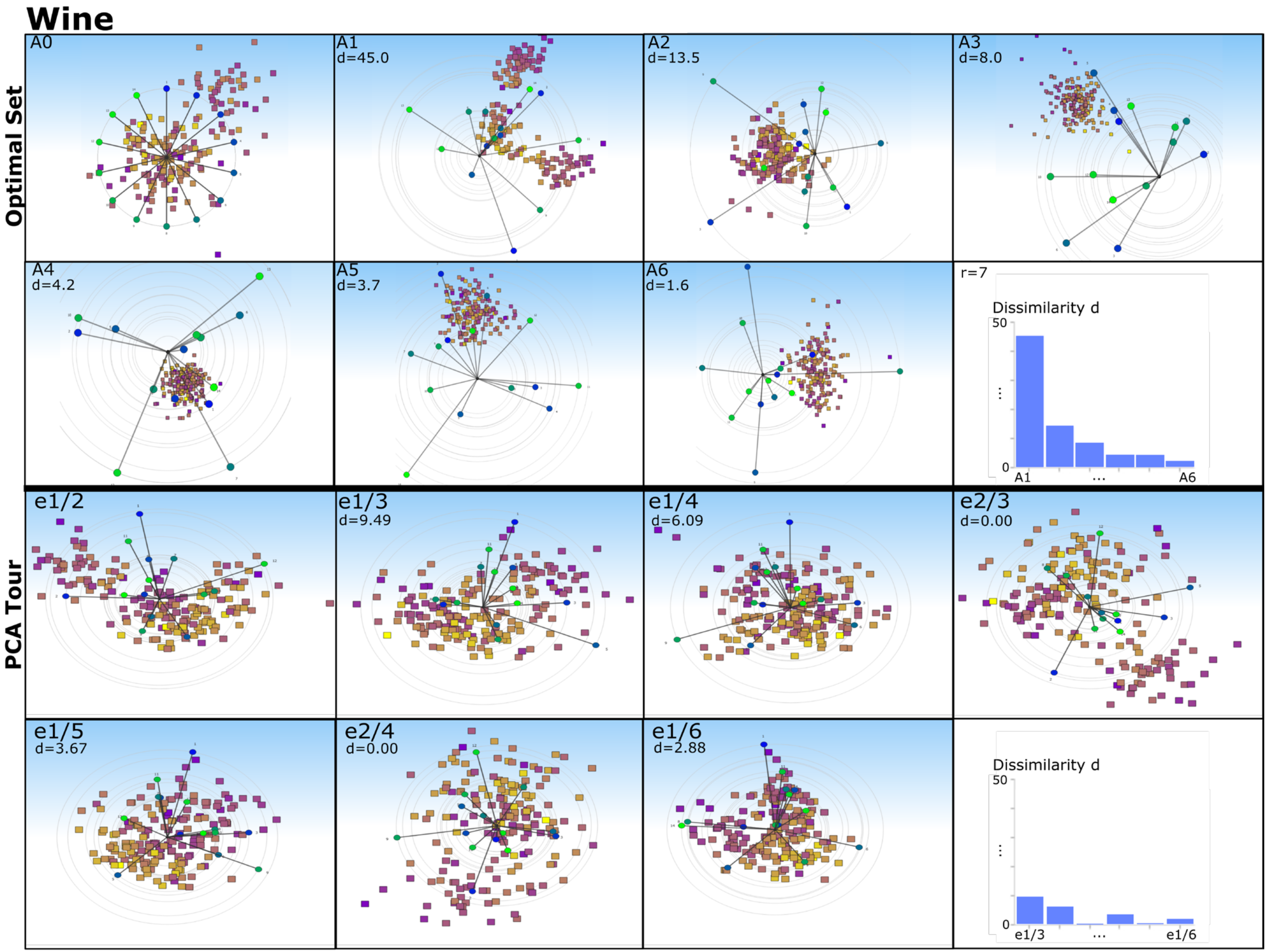
TERMINATING THE ALGORITHM

- Terminate when $d(B, A_0, \ldots, A_{i-1}) = 0$.
- i.e. We have a complete set of linear projects up to affine transforms.
- This occurs after at most n/2 projections.

Iris

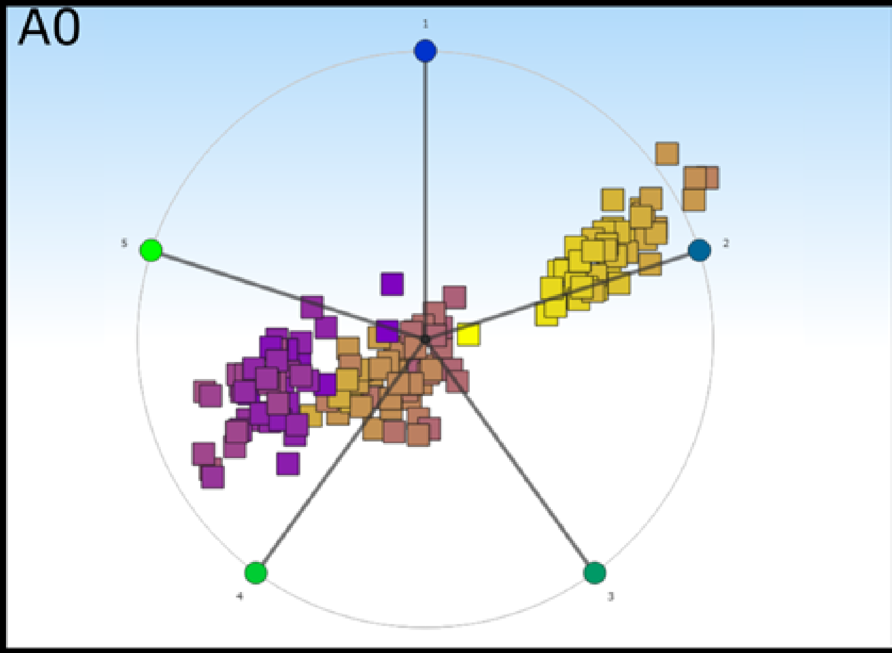


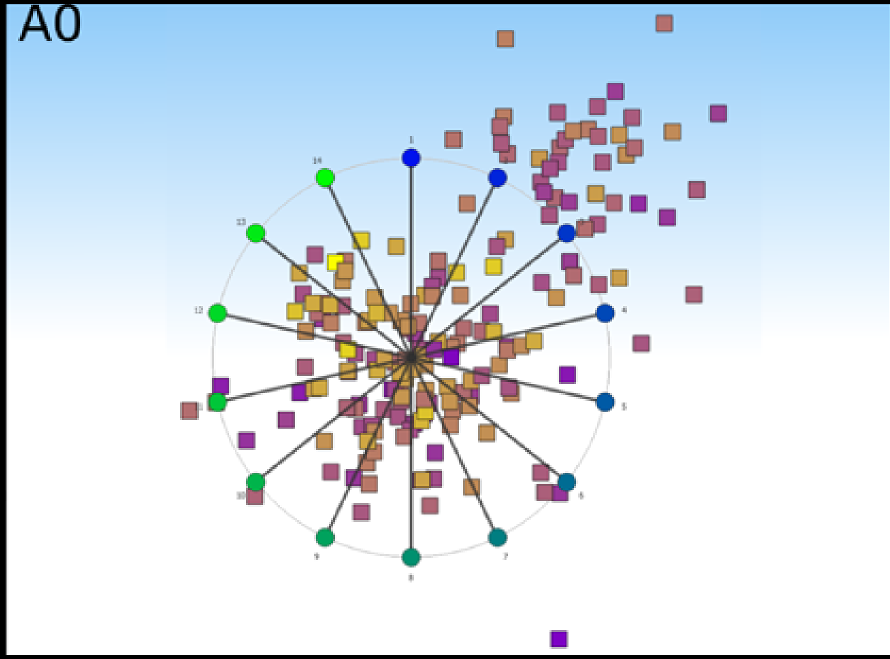
Optimal Set
A0
A1
d=12.5
A2
d=0.4
r=3
Dissimilarity d
12
0
A1
A2
PCA Tour
e1/2
e1/3
d=1.43
e1/4
d=0.61
Dissimilarity d
12
0
e1/3
e1/4

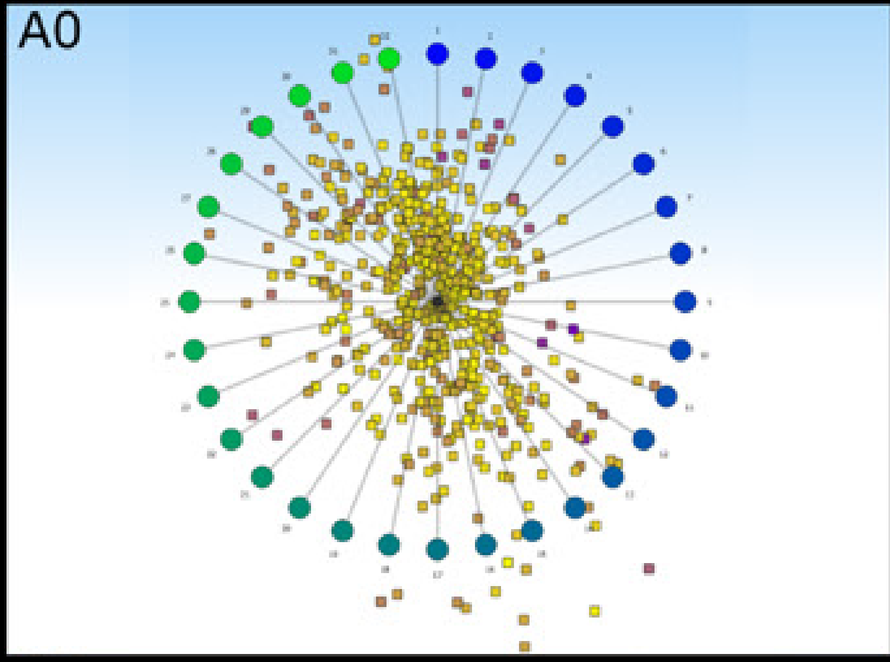

Wine
Optimal Set
PCA Tour
A0
A1
d=45.0
A2
d=13.5
A3
d=8.0
A4
d=4.2
A5
d=3.7
A6
d=1.6
r=7
Dissimilarity d
50
0
A1
...
A6
e1/2
e1/3
d=9.49
e1/4
d=6.09
e2/3
d=0.00
e1/5
d=3.67
e2/4
d=0.00
e1/6
d=2.88
Dissimilarity d
50
0
e1/3
...
e1/6

HOW DO WE CHOOSE $\{A_0\}$?

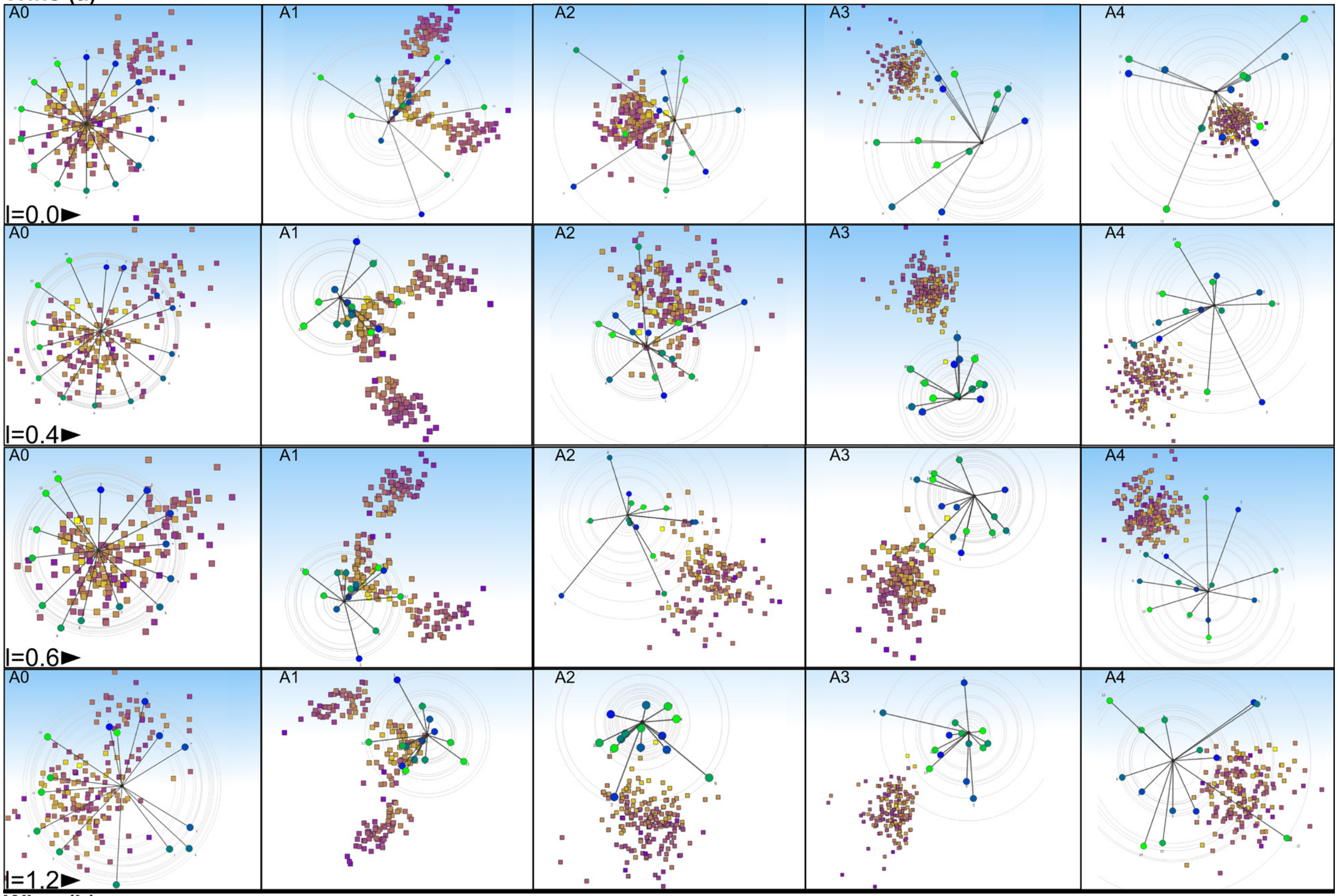
- Default choice: radial layout.
- Stable to alternative choices - the data patterns remain visible even if the projections change.







Wine (a)



A0
A1
A2
A3
A4
l=0.0
A0
A1
A2
A3
A4
l=0.4
A0
A1
A2
A3
A4
l=0.6
A0
A1
A2
A3
A4
l=1.2

SUMMARY

- The algorithm produces the optimal set of *linear* projections up to affine transforms.
- Produces < n/2 independent projections.
- Relatively robust to initialisation and convergence parameters.
- Scalability could be an issue? Distance is expensive.
- Needs testing to see if the affine assumption reasonable