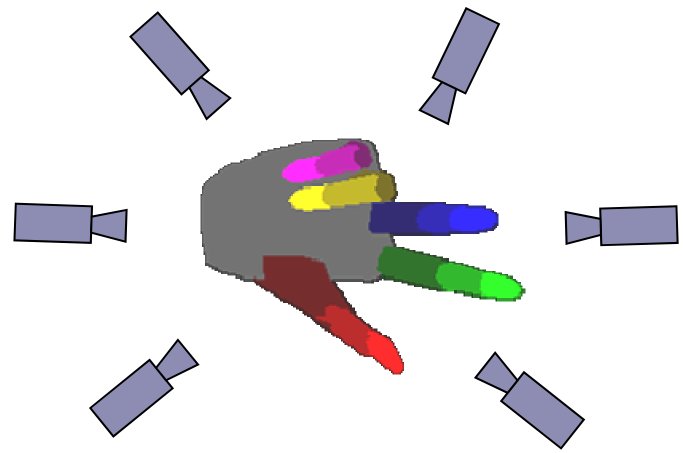
Fast Similarity Search in Image Databases

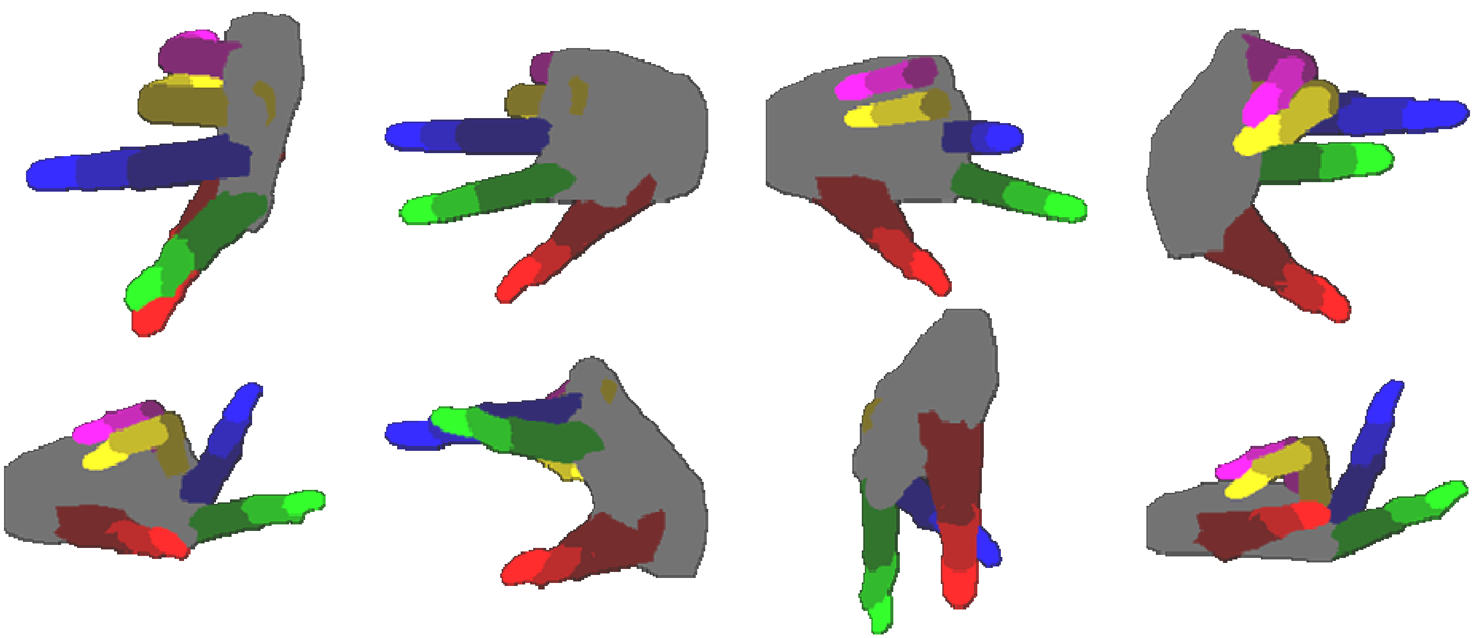
CSE 6367 – Computer Vision
Vassilis Athitsos
University of Texas at Arlington

A Database of Hand Images

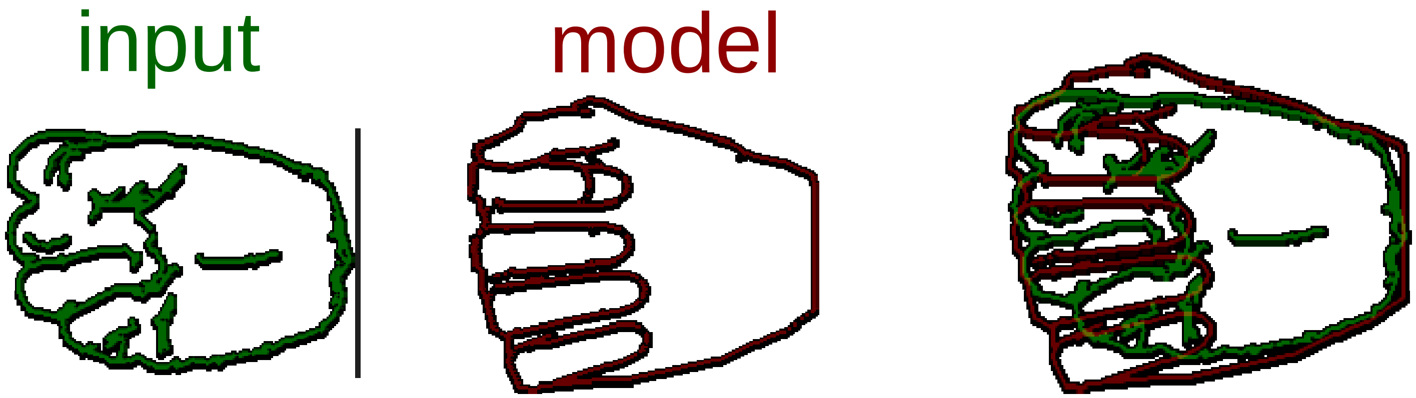
4128 images are generated for each hand shape.

Total: 107,328 images.





Efficiency of the Chamfer Distance

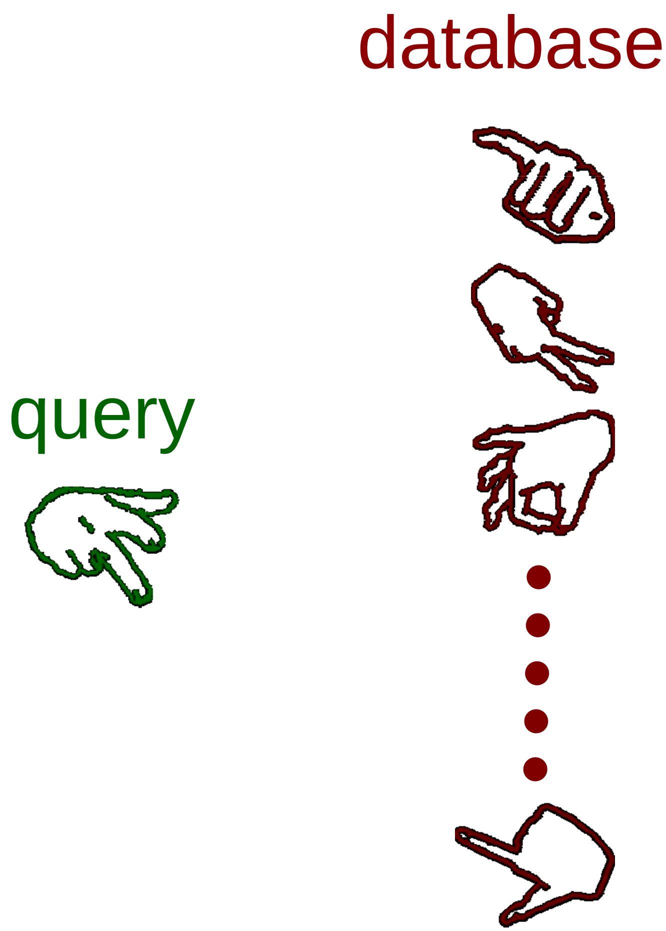


- Computing chamfer distances is slow.
  - For images with d edge pixels, O(d log d) time.
  - Comparing input to entire database takes over 4 minutes.
    - Must measure 107,328 distances.

The Nearest Neighbor Problem

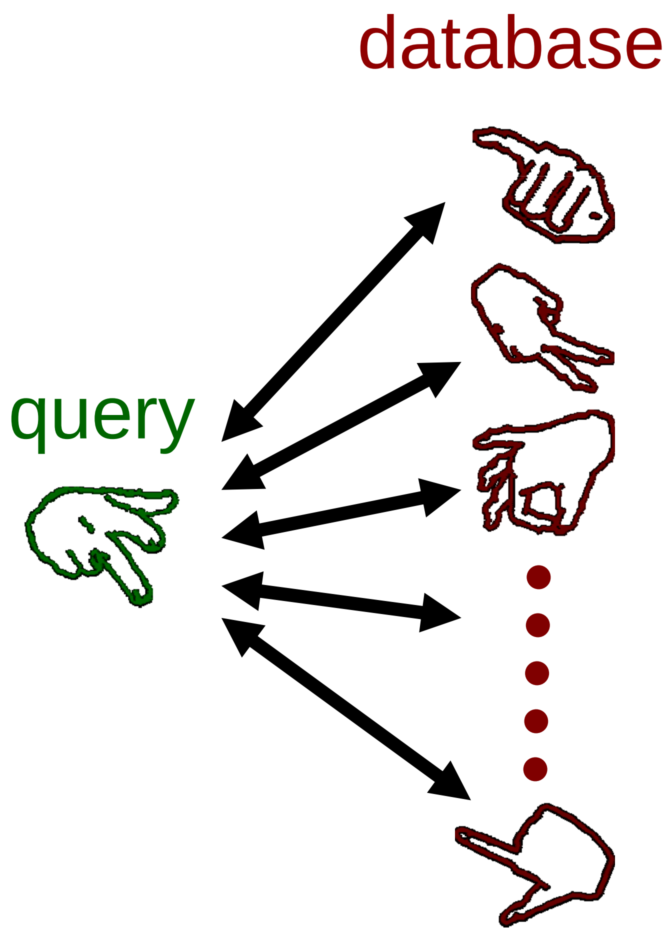
database

The Nearest Neighbor Problem



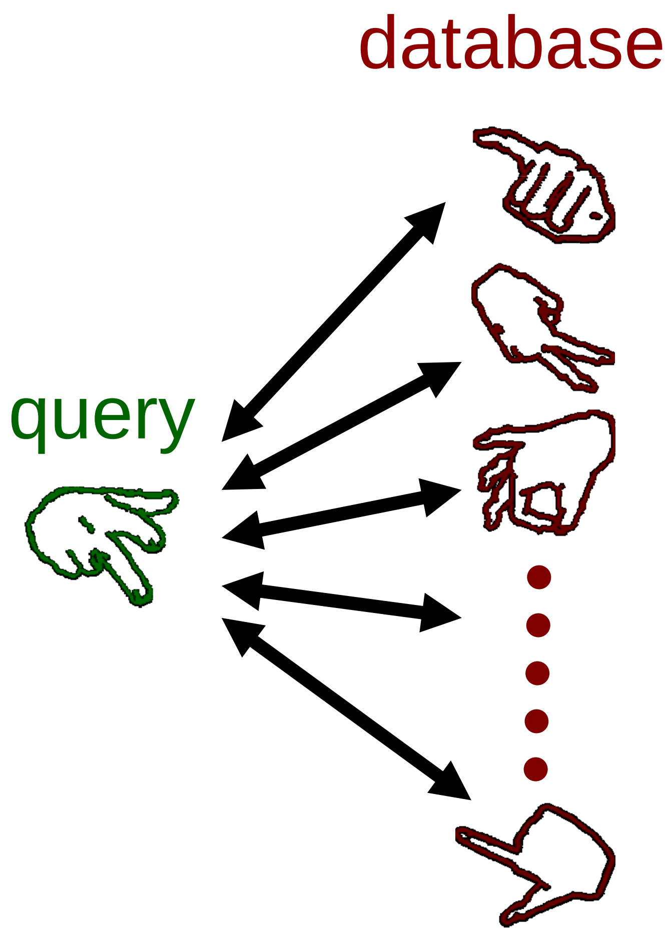
- Goal:
  - find the k nearest neighbors of query q.

The Nearest Neighbor Problem



- Goal:
  – find the k nearest neighbors of query q.
- Brute force time is linear to:
  – n (size of database).
  – time it takes to measure a *single distance*.

The Nearest Neighbor Problem



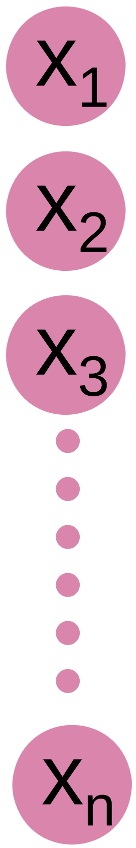
- Goal:
  - find the k nearest neighbors of query q.
- Brute force time is linear to:
  - n (size of database).
  - time it takes to measure a *single distance*.

Examples of Expensive Measures

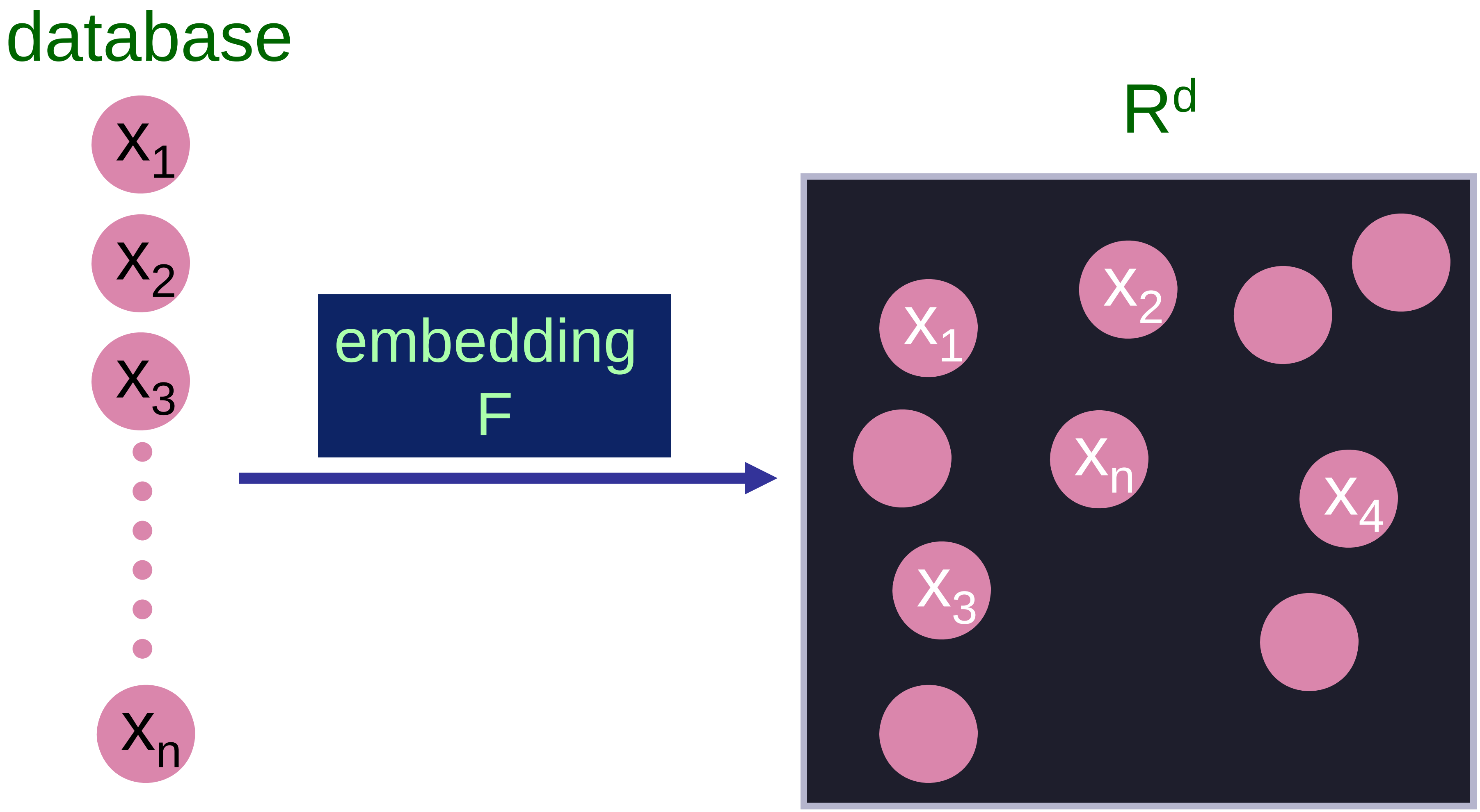
- DNA and protein sequences:
  - Smith-Waterman.
- Dynamic gestures and time series:
  - Dynamic Time Warping.
- Edge images:
  - Chamfer distance, shape context distance.
- These measures are non-Euclidean, sometimes non-metric.

Embeddings

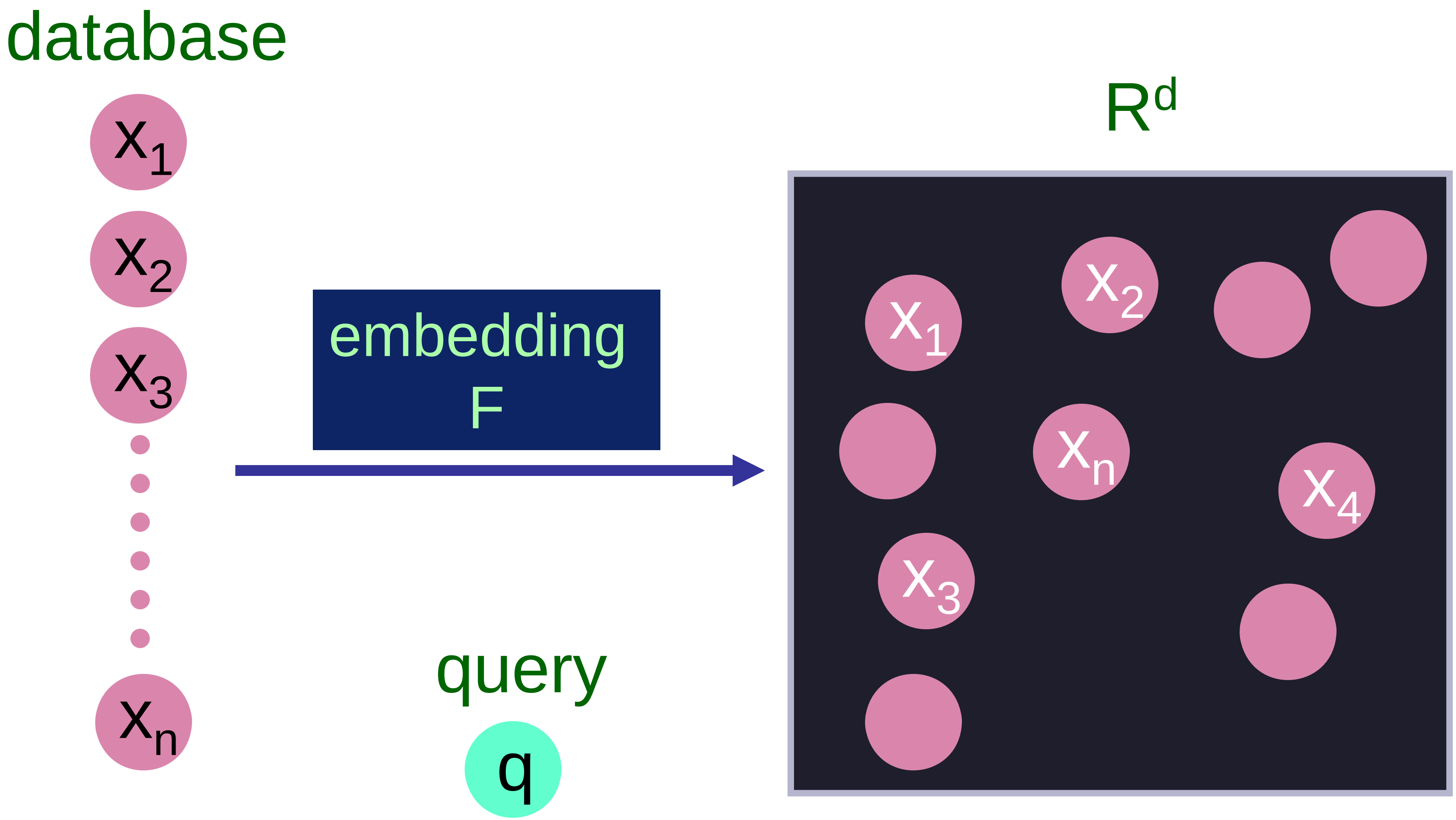
database



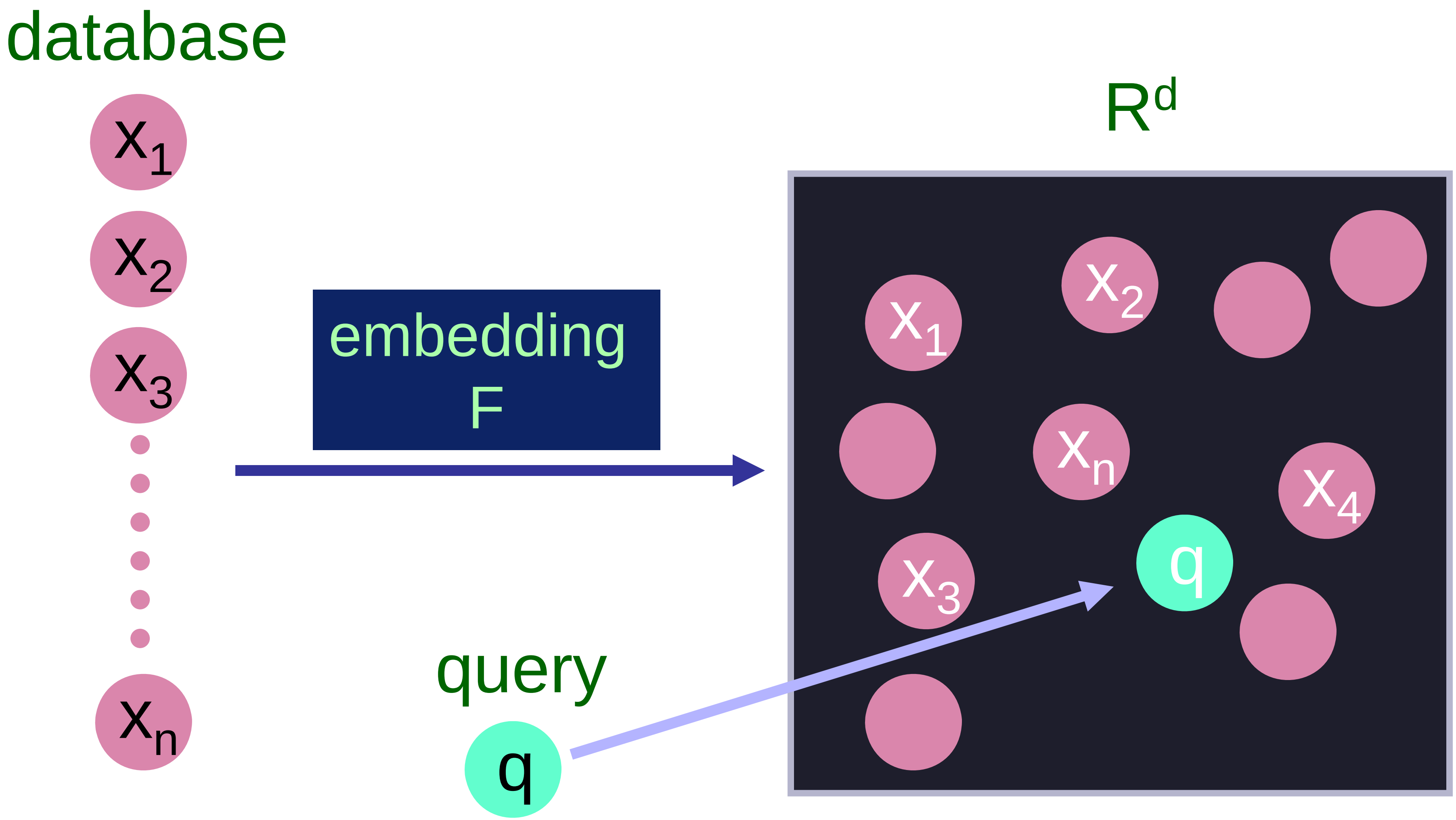
Embeddings



Embeddings

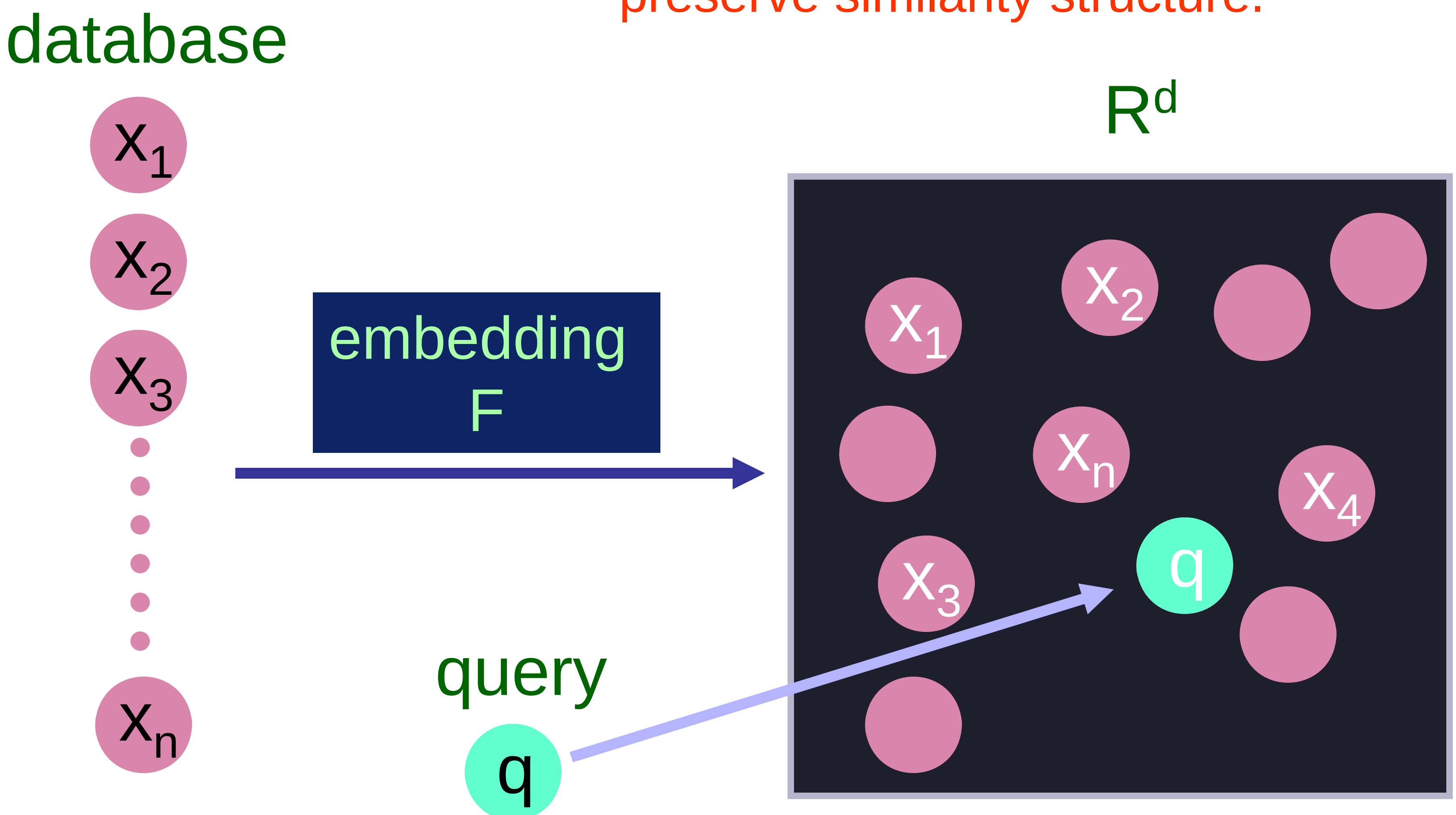


Embeddings

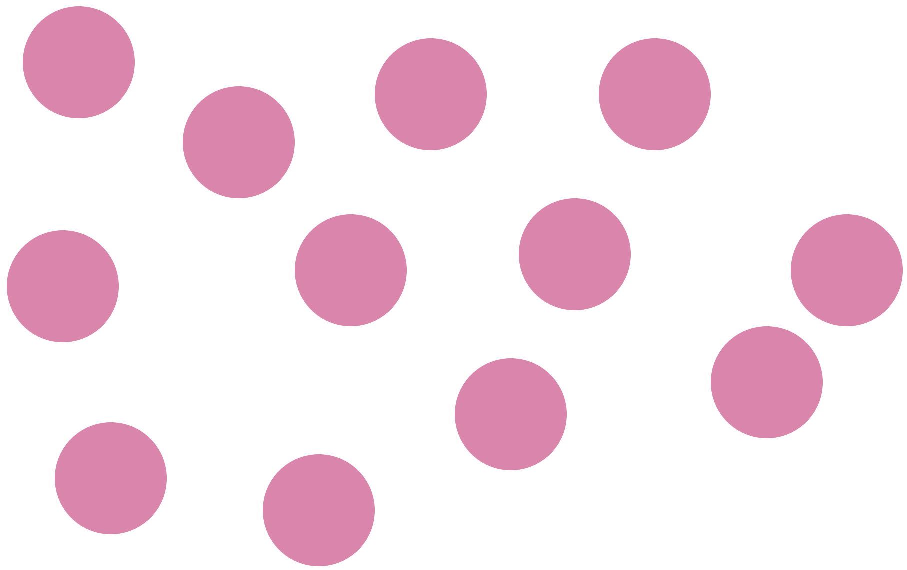


Embeddings

- Measure distances between vectors (typically much faster).
- Caveat: the embedding must preserve similarity structure.

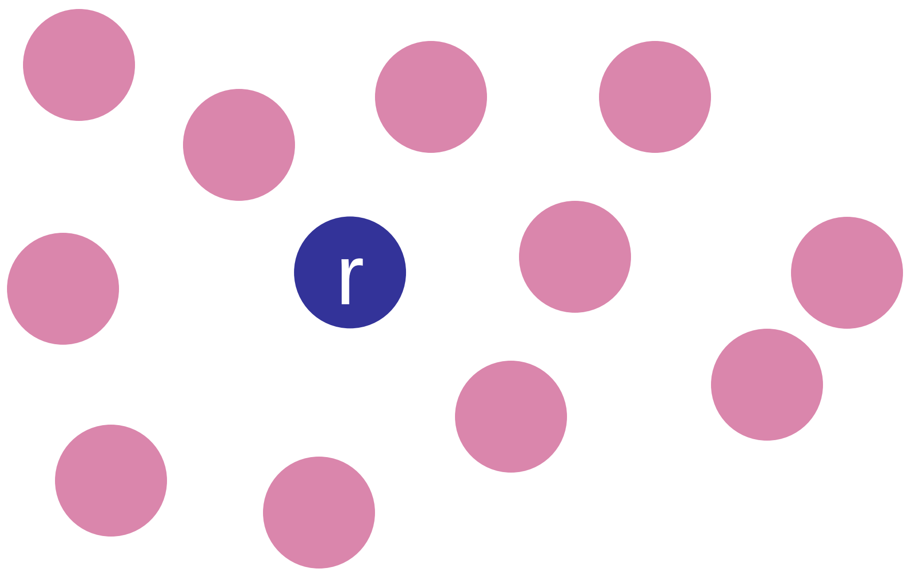


Reference Object Embeddings



original space X

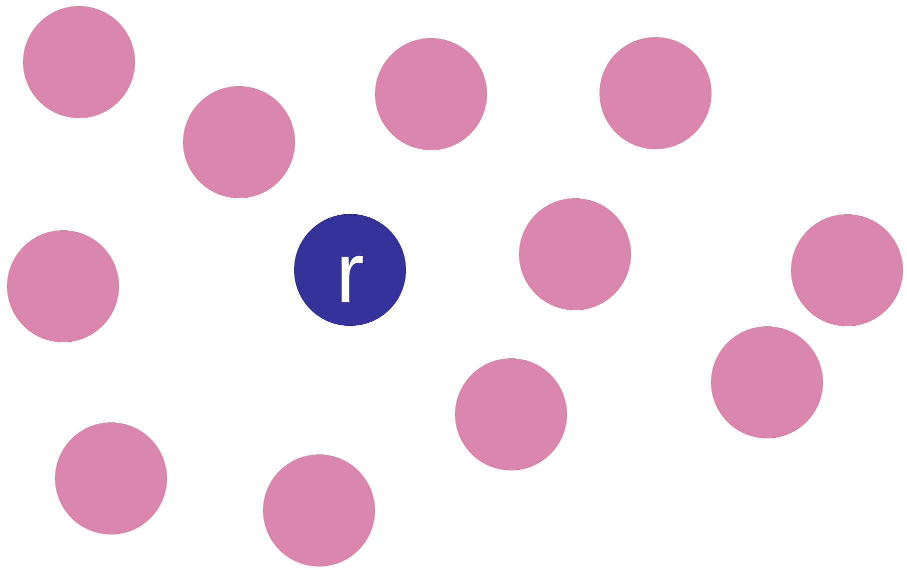
Reference Object Embeddings



original space X

r: reference object

Reference Object Embeddings



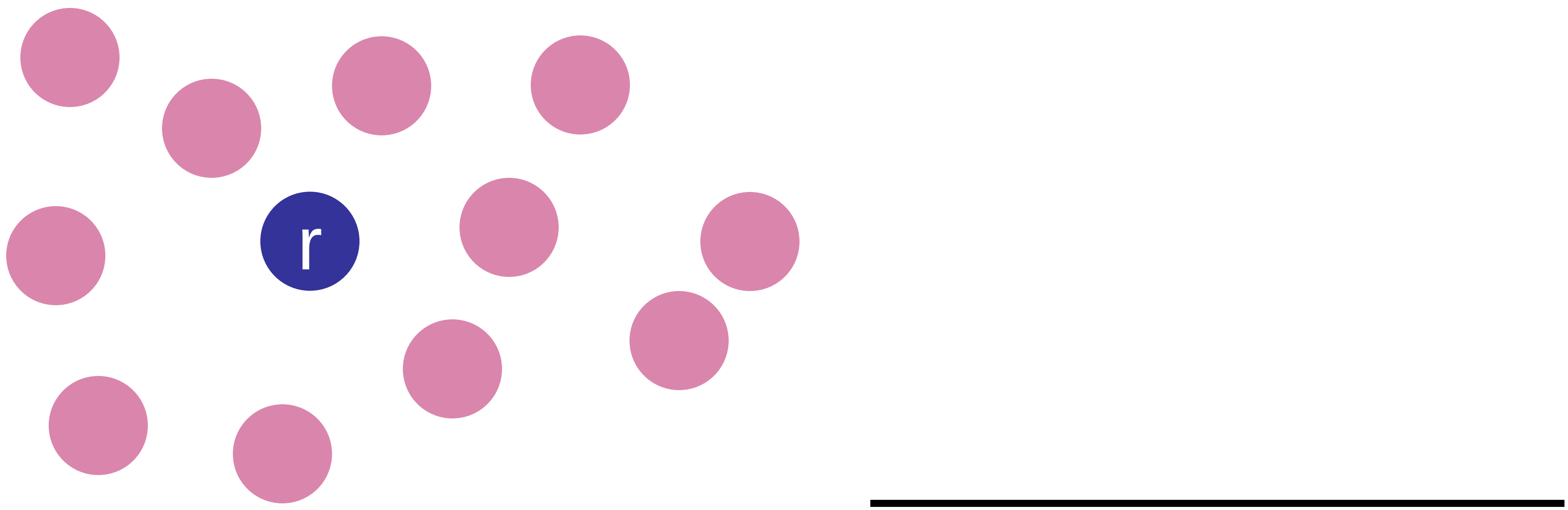
original space X

r: reference object

Embedding: F(x) = D(x,r)

D: distance measure in X.

Reference Object Embeddings



original space X $\xrightarrow{F}$ Real line

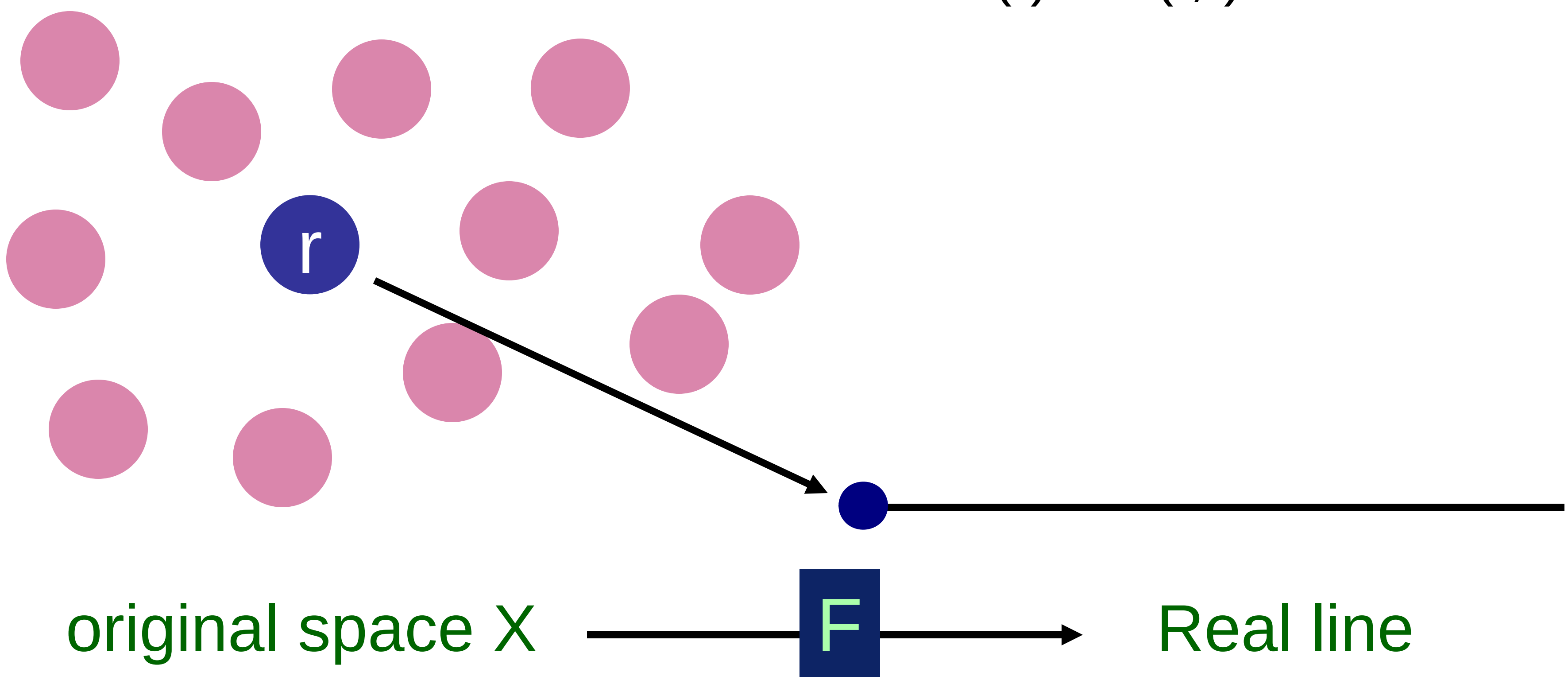
r: reference object

Embedding: F(x) = D(x,r)

D: distance measure in X.

Reference Object Embeddings

- F(r) = D(r,r) = 0



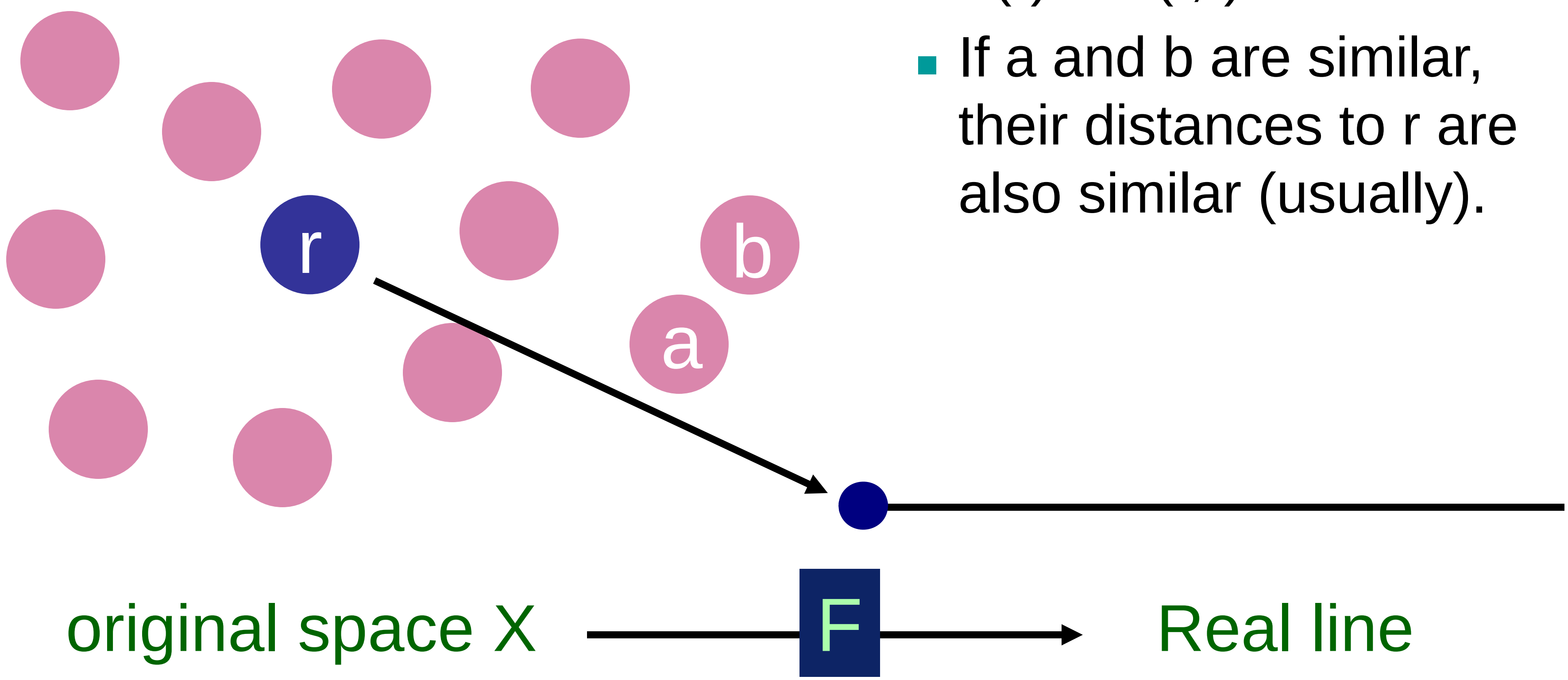
r: reference object

Embedding: F(x) = D(x,r)

D: distance measure in X.

Reference Object Embeddings

- F(r) = D(r,r) = 0
- If a and b are similar, their distances to r are also similar (usually).



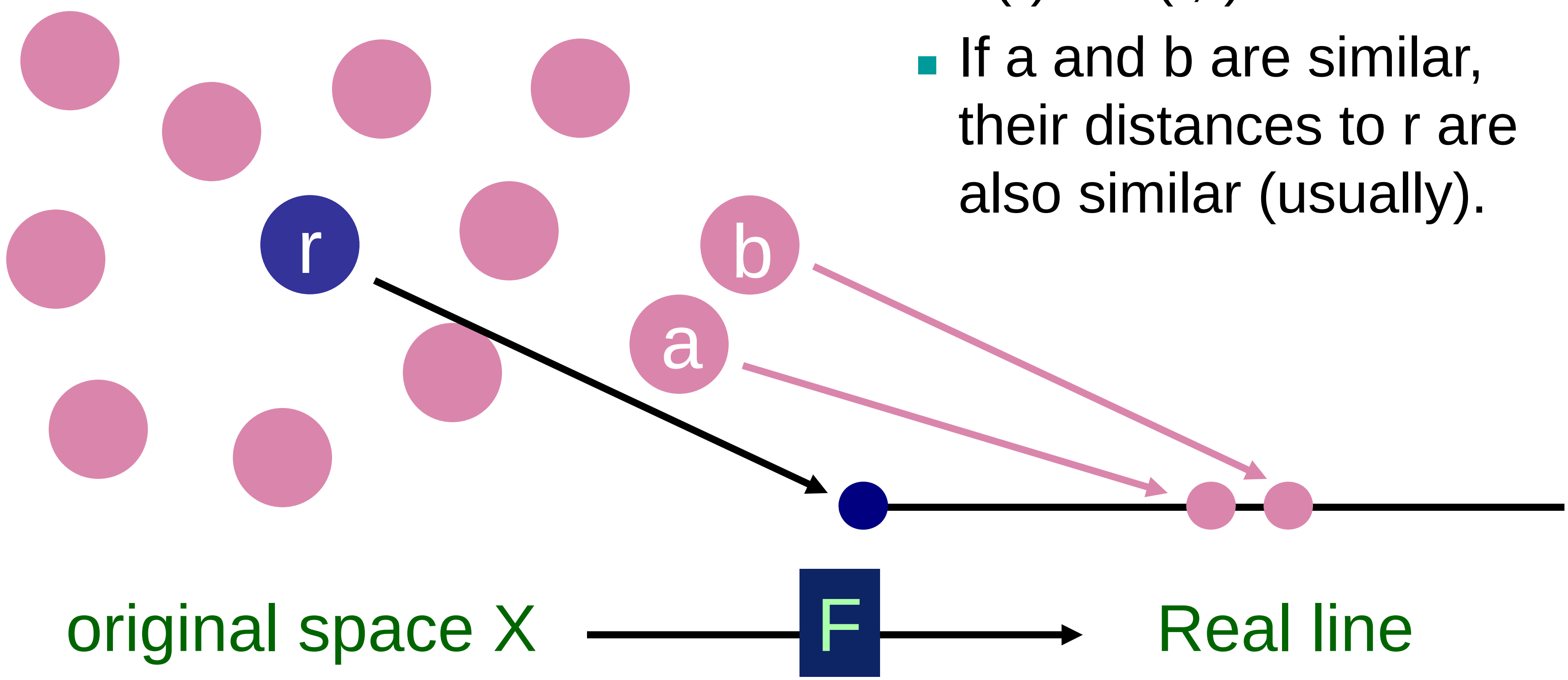
r: reference object

Embedding: F(x) = D(x,r)

D: distance measure in X.

Reference Object Embeddings

- $F(r) = D(r,r) = 0$
- If a and b are similar, their distances to r are also similar (usually).



r: reference object

Embedding: $F(x) = D(x,r)$

D: distance measure in X.

F(x) = D(x, Lincoln)

F(Sacramento)....= 1543

F(Las Vegas).....= 1232

F(Oklahoma City).= 437

F(Washington DC).= 1207

F(Jacksonville)..= 1344

F(x) = (D(x, LA), D(x, Lincoln), D(x, Orlando))


```
F(Sacramento)....= ( 386, 1543, 2920)
F(Las Vegas).....= ( 262, 1232, 2405)
F(Oklahoma City).= (1345,  437, 1291)
F(Washington DC).= (2657, 1207,  853)
F(Jacksonville)..= (2422, 1344,  141)
```

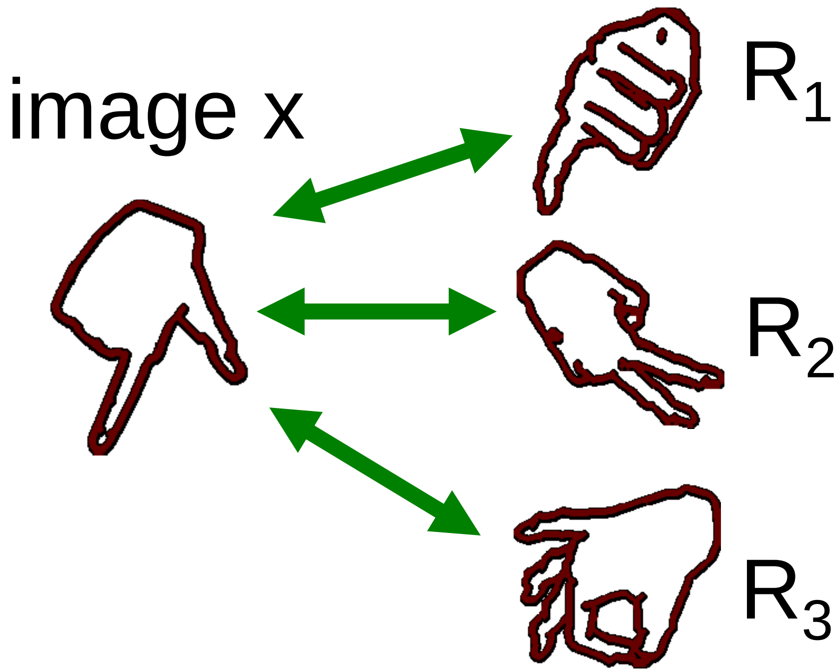
F(x) = (D(x, LA), D(x, Lincoln), D(x, Orlando))


```
F(Sacramento)....= ( 386, 1543, 2920)
F(Las Vegas).....= ( 262, 1232, 2405)
F(Oklahoma City).= (1345,  437, 1291)
F(Washington DC).= (2657, 1207,  853)
F(Jacksonville)..= (2422, 1344,  141)
```

Embedding Hand Images

$$\boxed{F(x) = (C(x, R_1), C(A, R_2), C(A, R_3))}$$

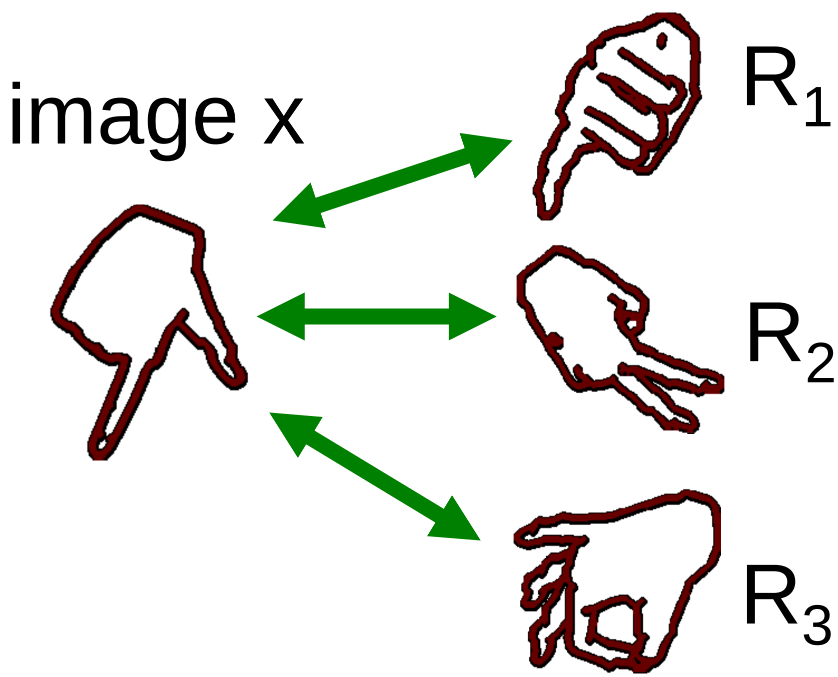
x: hand image. C: chamfer distance.



Basic Questions

$$F(x) = (C(x, R_1), C(A, R_2), C(A, R_3))$$

x: hand image. C: chamfer distance.

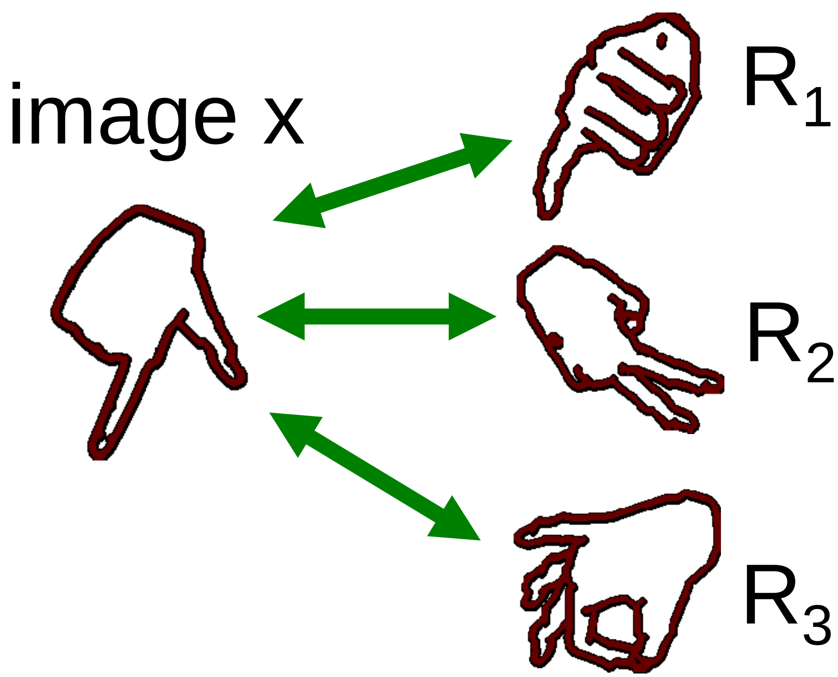


- How many prototypes?
- Which prototypes?
- What distance should we use to compare vectors?

Some Easy Answers.

$$F(x) = (C(x, R_1), C(A, R_2), C(A, R_3))$$

x: hand image. C: chamfer distance.



- How many prototypes?
  - Pick number manually.
- Which prototypes?
  - Randomly chosen.
- What distance should we use to compare vectors?
  - L_1, or Euclidean.

Filter-and-refine Retrieval

- Embedding step:
  - Compute distances from query to reference objects → F(q).
- Filter step:
  - Find top p matches of F(q) in vector space.
- Refine step:
  - Measure exact distance from q to top p matches.

Evaluating Embedding Quality

How often do we find the true nearest neighbor?

- Embedding step:
  - Compute distances from query to reference objects → F(q).
- Filter step:
  - Find top p matches of F(q) in vector space.
- Refine step:
  - Measure exact distance from q to top p matches.

Evaluating Embedding Quality

How often do we find the true nearest neighbor?

- Embedding step:
  - Compute distances from query to reference objects → F(q).
- Filter step:
  - Find top p matches of F(q) in vector space.
- Refine step:
  - Measure exact distance from q to top p matches.

Evaluating Embedding Quality

How often do we find the true nearest neighbor?

How many exact distance computations do we need?

- Embedding step:
  - Compute distances from query to reference objects → F(q).
- Filter step:
  - Find top p matches of F(q) in vector space.
- Refine step:
  - Measure exact distance from q to top p matches.

Evaluating Embedding Quality

How often do we find the true nearest neighbor?

How many exact distance computations do we need?

- Embedding step:
  - Compute distances from query to reference objects → F(q).
- Filter step:
  - Find top p matches of F(q) in vector space.
- Refine step:
  - Measure exact distance from q to top p matches.

Evaluating Embedding Quality

How often do we find the true nearest neighbor?

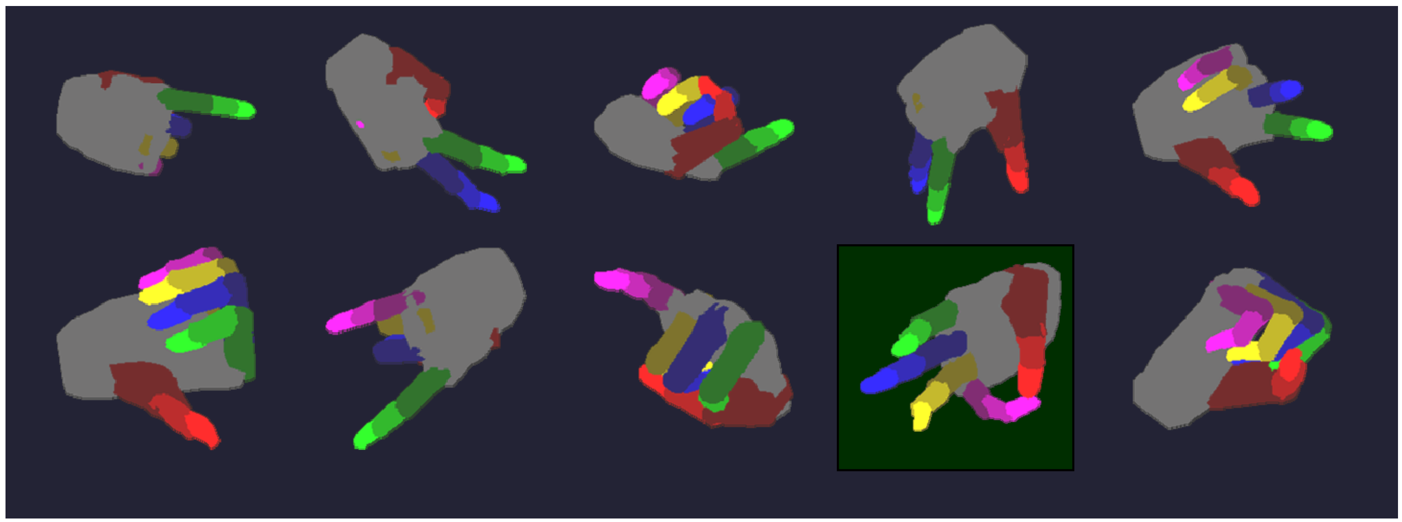
How many exact distance computations do we need?

- Embedding step:
  - Compute distances from query to reference objects → F(q).
- Filter step:
  - Find top p matches of F(q) in vector space.
- Refine step:
  - Measure exact distance from q to top p matches.

Results: Chamfer Distance on Hand Images

Database (107,328 images)

query



nearest neighbor

Brute force retrieval time: 260 seconds.

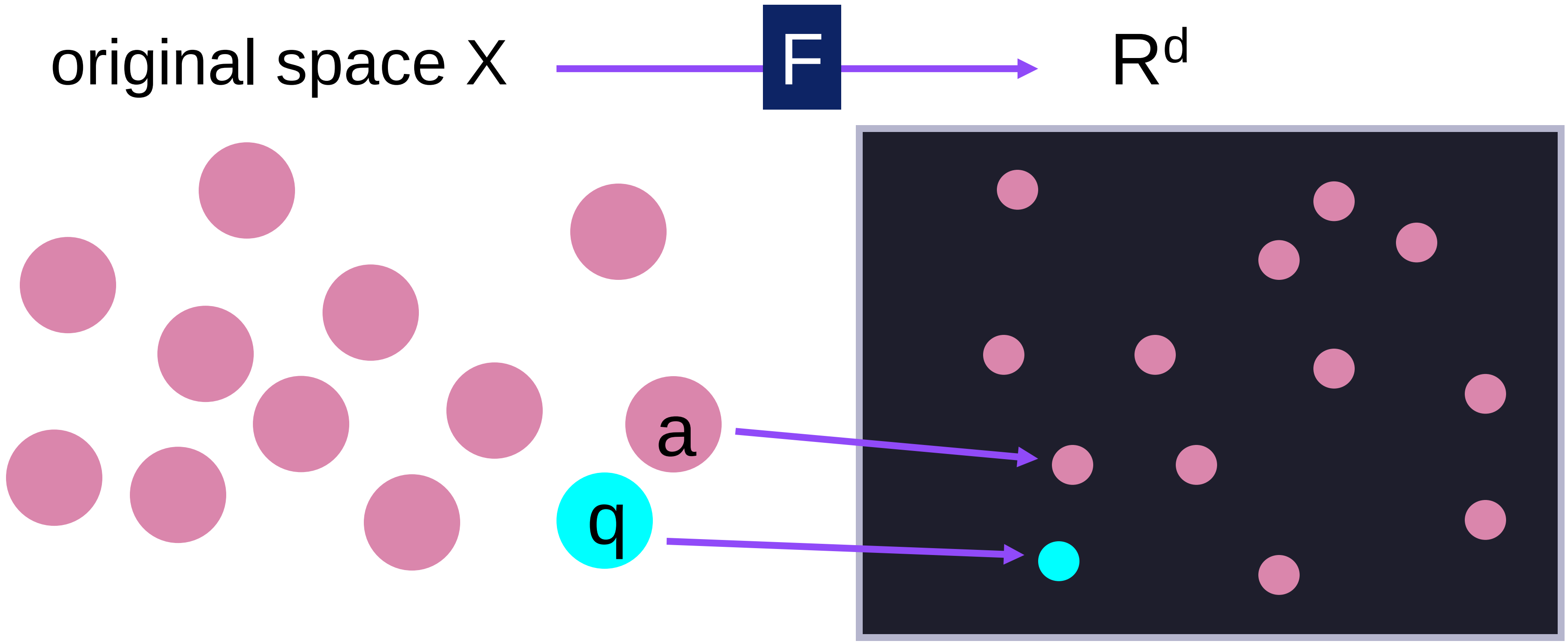
Results: Chamfer Distance on Hand Images

Database: 80,640 synthetic images of hands.

Query set: 710 real images of hands.

	Brute Force	Embeddings	Embeddings
Accuracy	**100%**	**95%**	**100%**
# of distances	**80640**	**1866**	**24650**
Sec. per query	**112**	**2.6**	**34**
Speed-up factor	**1**	**43**	**3.27**

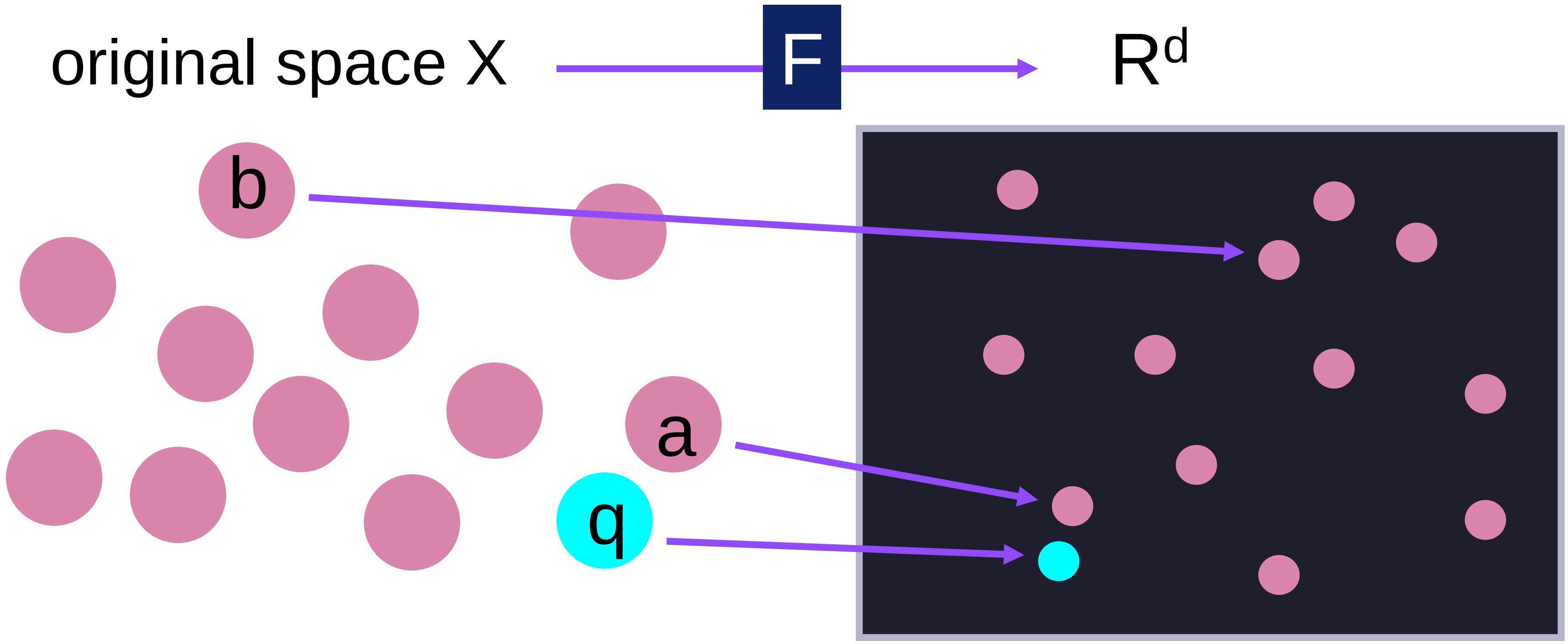
Ideal Embedding Behavior



Notation: NN(q) is the nearest neighbor of q.

For any q: if a = NN(q), we want F(a) = NN(F(q)).

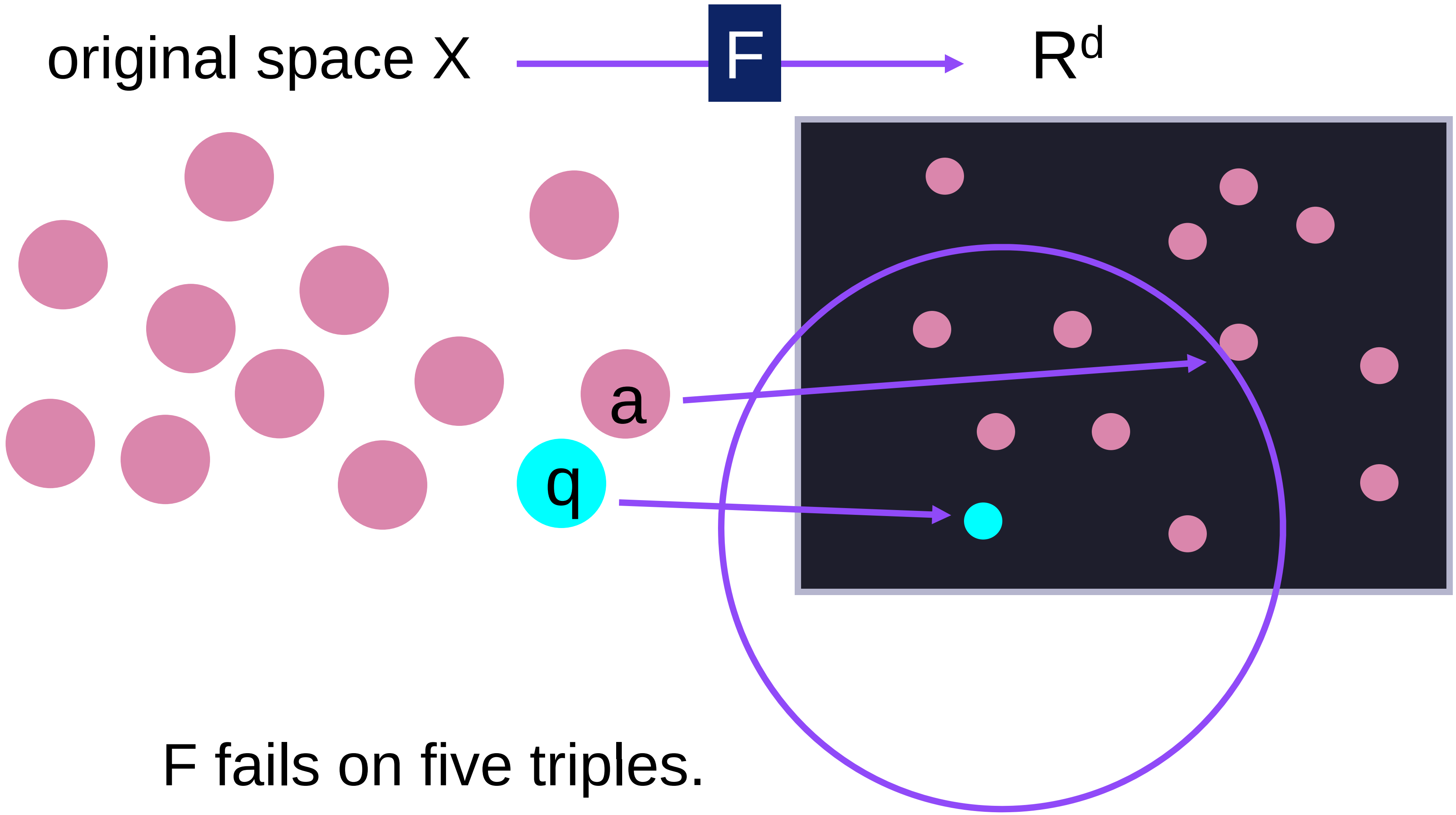
A Quantitative Measure



If b is not the nearest neighbor of q,
F(q) should be closer to F(NN(q)) than to F(b).

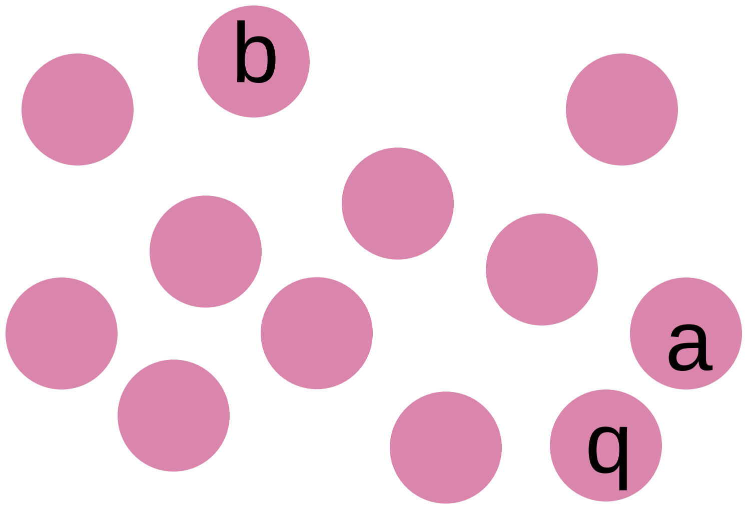
For how many triples (q, NN(q), b) does F fail?

A Quantitative Measure



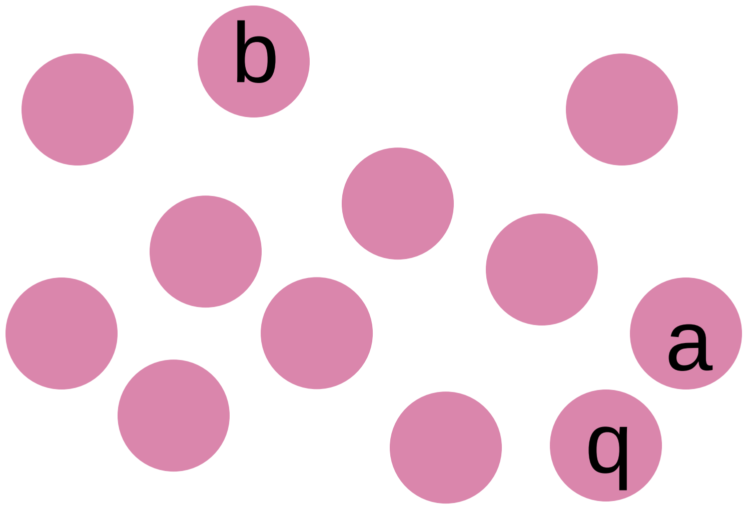
F fails on five triples.

Embeddings Seen As Classifiers



Classification task: is q closer to a or to b?

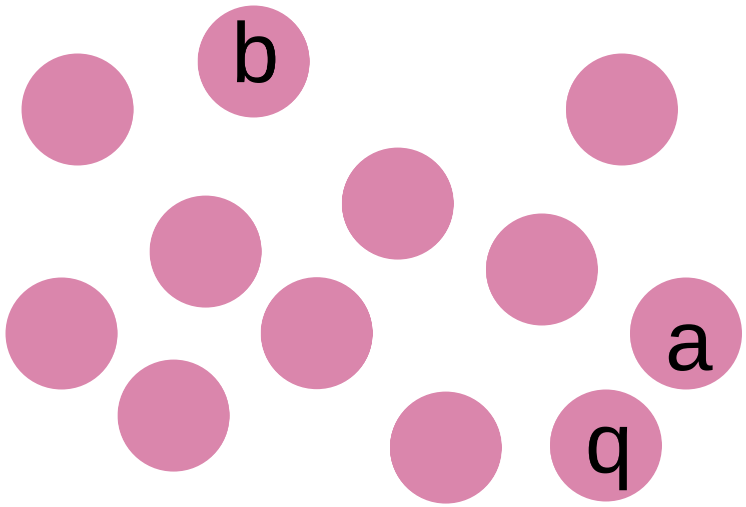
Embeddings Seen As Classifiers



Classification task: is q closer to a or to b?

- Any embedding F defines a classifier F'(q, a, b).
  - F' checks if F(q) is closer to F(a) or to F(b).

Classifier Definition



Classification task: is q closer to a or to b?

- Given embedding $F: X \rightarrow R^d$:
  - $F'(q, a, b) = ||F(q) – F(b)|| - ||F(q) – F(a)||$.
- $F'(q, a, b) > 0$ means “q is closer to a.”
- $F'(q, a, b) < 0$ means “q is closer to b.”

Classifier Definition

Goal: build an F such that F' has low error rate on triples of type (q, NN(q), b).

- Given embedding $F: X \rightarrow R^d$:
  - $F'(q, a, b) = ||F(q) - F(b)|| - ||F(q) - F(a)||$.
- F'(q, a, b) > 0 means "q is closer to a."
- F'(q, a, b) < 0 means "q is closer to b."

1D Embeddings as Weak Classifiers

- 1D embeddings define *weak classifiers.*
  - Better than a random classifier (50% error rate).

1D Embeddings as Weak Classifiers

- 1D embeddings define *weak classifiers.*
  - Better than a random classifier (50% error rate).
- We can define lots of different classifiers.
  - Every object in the database can be a reference object.

Question: how do we combine many such classifiers into a single *strong* classifier?

1D Embeddings as Weak Classifiers

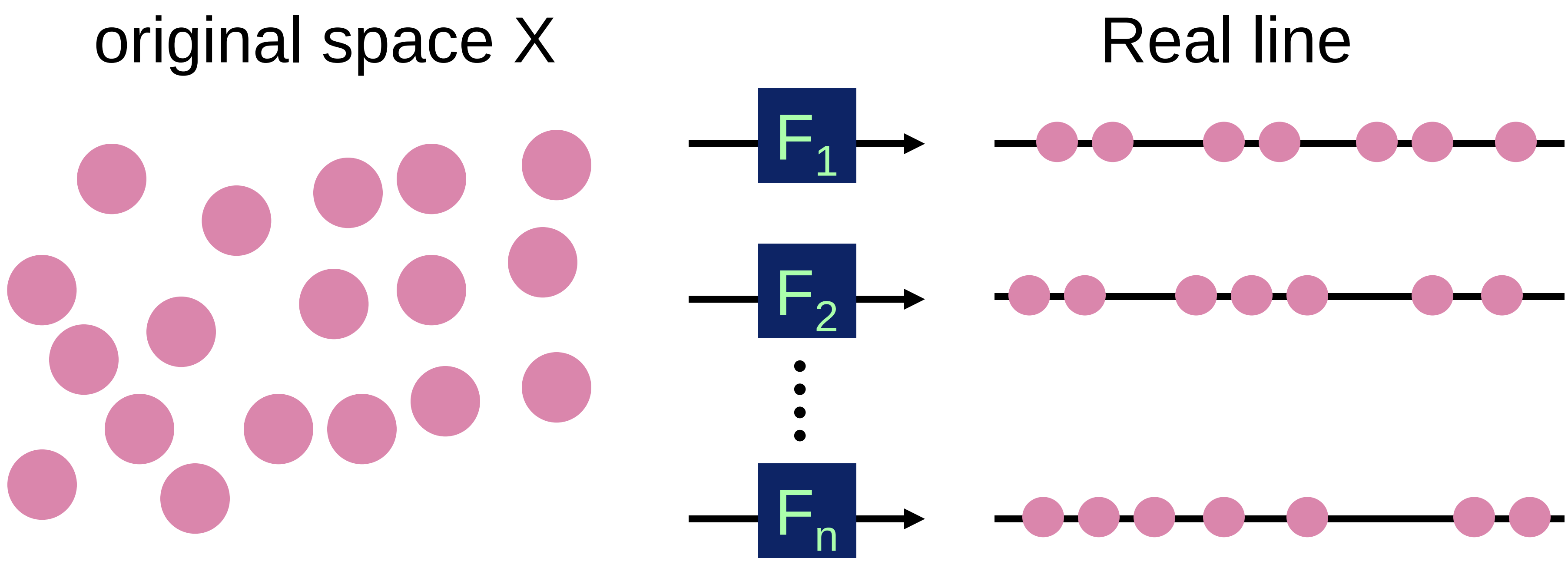
- 1D embeddings define *weak classifiers.*
  - Better than a random classifier (50% error rate).
- We can define lots of different classifiers.
  - Every object in the database can be a reference object.

Question: how do we combine many such classifiers into a single *strong* classifier?

Answer: use AdaBoost.

- AdaBoost is a machine learning method designed for exactly this problem.

Using AdaBoost



- Output: $H = w_1F'_1 + w_2F'_2 + \ldots + w_dF'_d$.
  - AdaBoost chooses 1D embeddings and weighs them.
  - Goal: achieve low classification error.
  - AdaBoost trains on triples chosen from the database.

From Classifier to Embedding

AdaBoost output $\boxed{H = w_1F'_1 + w_2F'_2 + \ldots + w_dF'_d}$

What embedding should we use?
What distance measure should we use?

From Classifier to Embedding

AdaBoost output $\boxed{H = w_1 F'_1 + w_2 F'_2 + \ldots + w_d F'_d}$

BoostMap embedding $\boxed{F(x) = (F_1(x), \ldots, F_d(x)).}$

From Classifier to Embedding

AdaBoost output

$$\boxed{H = w_1 F'_1 + w_2 F'_2 + \ldots + w_d F'_d}$$

BoostMap embedding

$$\boxed{F(x) = (F_1(x), \ldots, F_d(x)).}$$

Distance measure

$$\boxed{D((u_1, \ldots, u_d), (v_1, \ldots, v_d)) = \sum_{i=1}^{d} w_i |u_i - v_i|}$$

From Classifier to Embedding

AdaBoost output $\boxed{H = w_1F'_1 + w_2F'_2 + \ldots + w_dF'_d}$

BoostMap embedding $\boxed{F(x) = (F_1(x), \ldots, F_d(x)).}$

Distance measure $\boxed{D((u_1, \ldots, u_d), (v_1, \ldots, v_d)) = \sum_{i=1}^{d} w_i|u_i - v_i|}$

Claim:

Let q be closer to a than to b. H misclassifies triple (q, a, b) if and only if, under distance measure D, F maps q closer to b than to a.

Proof

$H(q, a, b) =$

$$= \sum_{i=1}^{d} w_i F'_i(q, a, b)$$

$$= \sum_{i=1}^{d} w_i(|F_i(q) - F_i(b)| - |F_i(q) - F_i(a)|)$$

$$= \sum_{i=1}^{d} (w_i|F_i(q) - F_i(b)| - w_i|F_i(q) - F_i(a)|)$$

$$= D(F(q), F(b)) – D(F(q), F(a)) = F'(q, a, b)$$

Proof

$H(q, a, b) =$

$= \sum_{i=1}^{d} w_i F'_i(q, a, b)$

$= \sum_{i=1}^{d} w_i(|F_i(q) - F_i(b)| - |F_i(q) - F_i(a)|)$

$= \sum_{i=1}^{d} (w_i|F_i(q) - F_i(b)| - w_i|F_i(q) - F_i(a)|)$

$= D(F(q), F(b)) – D(F(q), F(a)) = F'(q, a, b)$

Proof

$H(q, a, b) =$

$= \sum_{i=1}^{d} w_i F'_i(q, a, b)$

$= \sum_{i=1}^{d} w_i(|F_i(q) - F_i(b)| - |F_i(q) - F_i(a)|)$

$= \sum_{i=1}^{d} (w_i|F_i(q) - F_i(b)| - w_i|F_i(q) - F_i(a)|)$

$= D(F(q), F(b)) – D(F(q), F(a)) = F'(q, a, b)$

Proof

$$H(q, a, b) =$$

$$= \sum_{i=1}^{d} w_i F'_i(q, a, b)$$

$$= \sum_{i=1}^{d} w_i(|F_i(q) - F_i(b)| - |F_i(q) - F_i(a)|)$$

$$= \sum_{i=1}^{d} (w_i|F_i(q) - F_i(b)| - w_i|F_i(q) - F_i(a)|)$$

$$= D(F(q), F(b)) – D(F(q), F(a)) = F'(q, a, b)$$

Proof

$H(q, a, b) =$

$= \sum_{i=1}^{d} w_i F'_i(q, a, b)$

$= \sum_{i=1}^{d} w_i(|F_i(q) - F_i(b)| - |F_i(q) - F_i(a)|)$

$= \sum_{i=1}^{d} (w_i|F_i(q) - F_i(b)| - w_i|F_i(q) - F_i(a)|)$

$= D(F(q), F(b)) – D(F(q), F(a)) = F'(q, a, b)$

Proof

$H(q, a, b) =$

$= \sum_{i=1}^{d} w_i F'_i(q, a, b)$

$= \sum_{i=1}^{d} w_i(|F_i(q) - F_i(b)| - |F_i(q) - F_i(a)|)$

$= \sum_{i=1}^{d} (w_i|F_i(q) - F_i(b)| - w_i|F_i(q) - F_i(a)|)$

$= D(F(q), F(b)) – D(F(q), F(a)) = F'(q, a, b)$

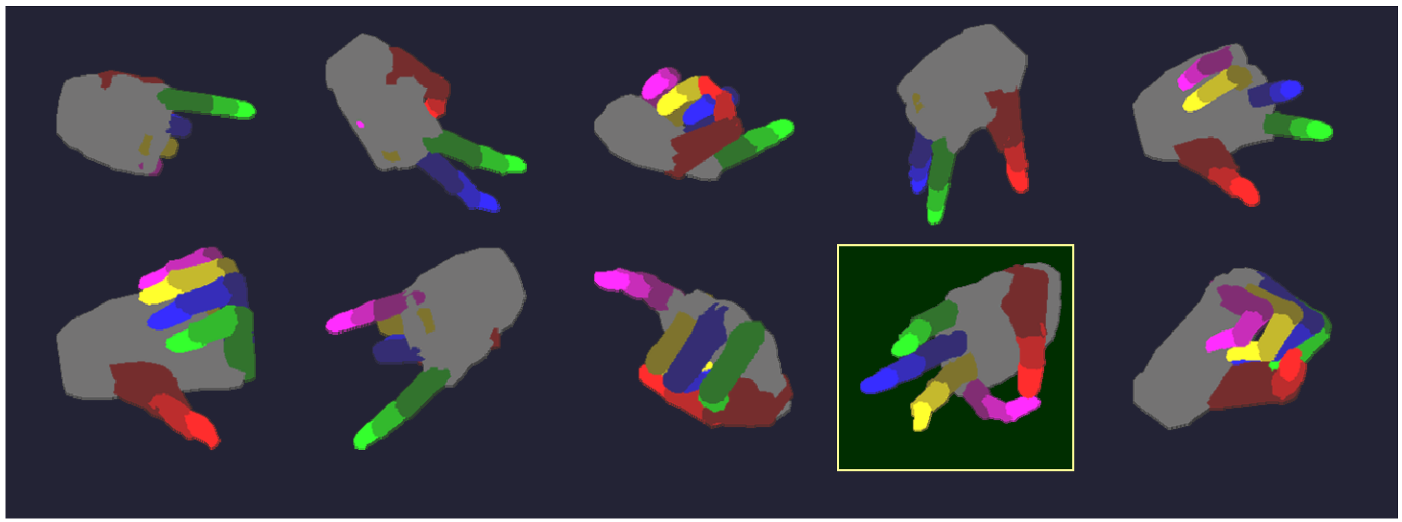
Significance of Proof

- AdaBoost optimizes a direct measure of embedding quality.
- We have converted a database indexing problem into a machine learning problem.

Results: Chamfer Distance on Hand Images

Database (80,640 images)

query



nearest neighbor

Brute force retrieval time: 112 seconds.

Results: Chamfer Distance on Hand Images

Database: 80,640 synthetic images of hands.

Query set: 710 real images of hands.

	Brute Force	Random Reference Objects	BoostMap
Accuracy	**100%**	**95%**	**95%**
# of distances	**80640**	**1866**	**450**
Sec. per query	**112**	**2.6**	**0.63**
Speed-up factor	**1**	**43**	**179**

Results: Chamfer Distance on Hand Images

Database: 80,640 synthetic images of hands.

Query set: 710 real images of hands.

	Brute Force	Random Reference Objects	BoostMap
Accuracy	**100%**	**100%**	**100%**
# of distances	**80640**	**24950**	**5995**
Sec. per query	**112**	**34**	**13.5**
Speed-up factor	**1**	**3.23**	**8.3**