

Learning Embeddings for Indexing, Retrieval, and Classification, With Applications to Object and Shape Recognition in Image Databases

Vassilis Athitsos

Computer Science Department

Boston University

Problem Definition

database
(n objects)

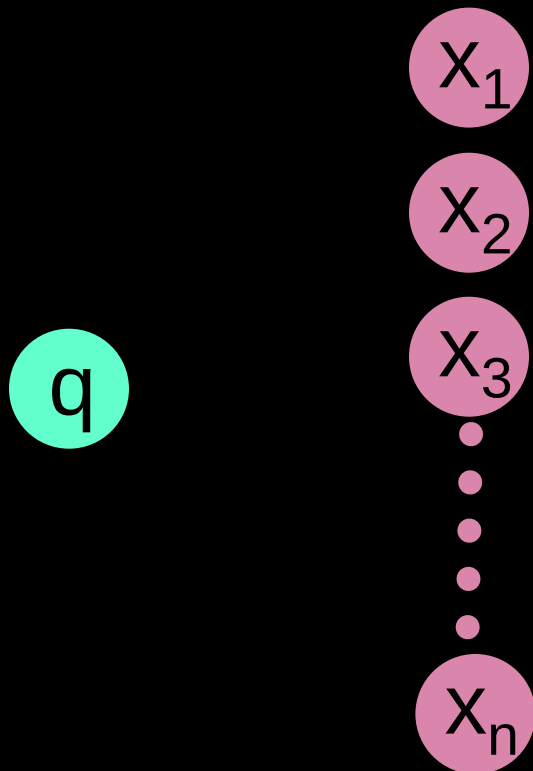


Problem Definition

database
(n objects)

■ Goals:

- find the k nearest neighbors of query q .

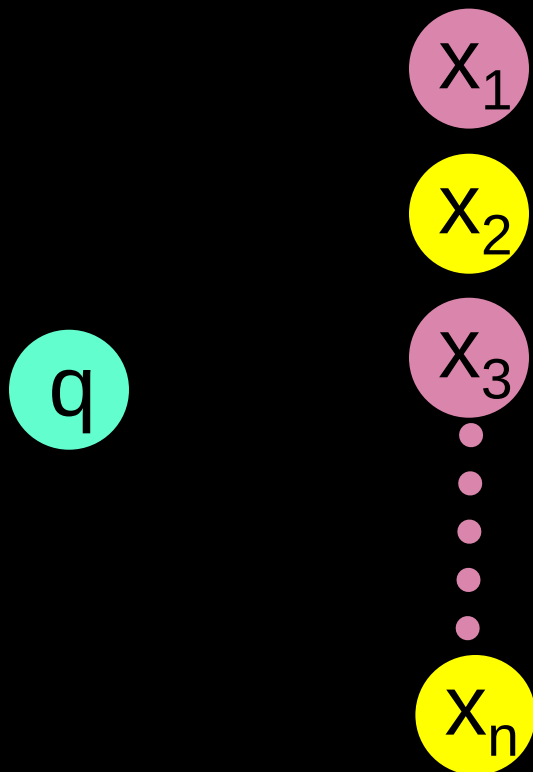


Problem Definition

database
(n objects)

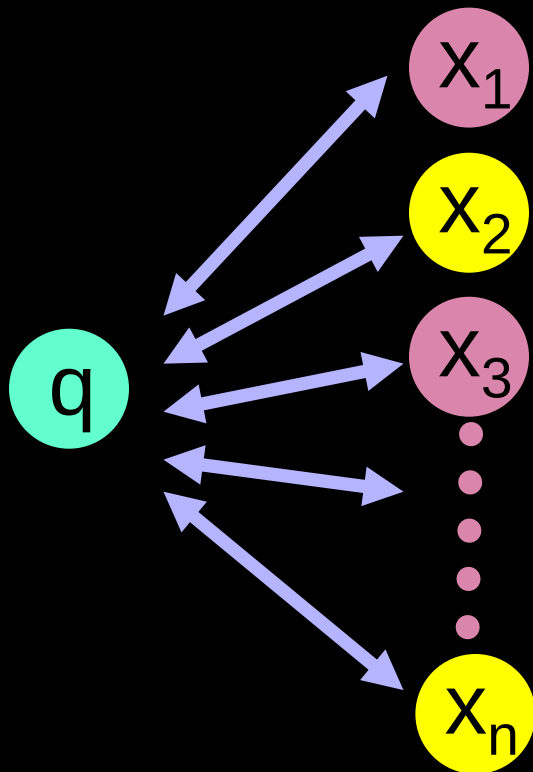
■ Goals:

- find the k nearest neighbors of query q.
- classify query q.



Problem Definition

database
(n objects)



■ Goals:

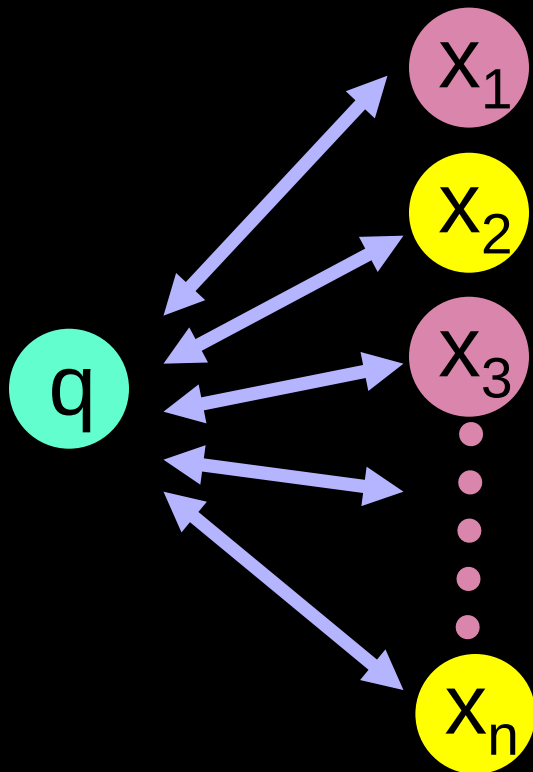
- find the k nearest neighbors of query q .
- classify query q .

■ Brute force time is linear to:

- n (size of database).
- time it takes to measure a *single distance*.

Problem Definition

database
(n objects)



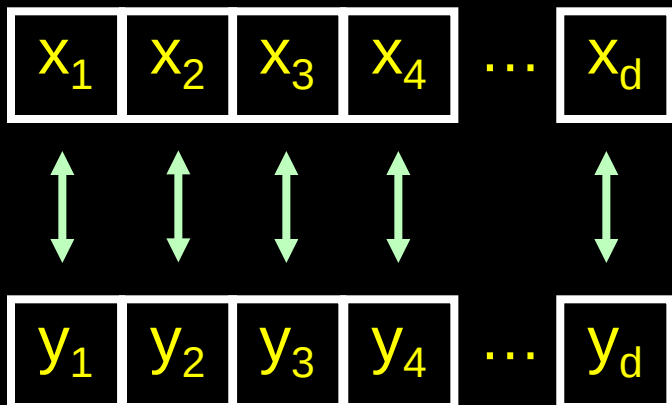
- Goals:
 - find the k nearest neighbors of query q .
 - classify query q .
- Brute force time is linear to:
 - n (size of database).
 - time it takes to measure a *single distance*.

Overview

- In many domains we use computationally expensive (slow) distance measures.
 - Matching shapes, strings, proteins.
- Embeddings map objects to a space where distances can be measured fast.
- BoostMap is a novel embedding method.
 - Main novelty: machine learning formulation.
 - Main advantage: principled optimization.
 - No assumptions of Euclidean or metric properties.
 - Experiments: better efficiency/accuracy trade-offs.

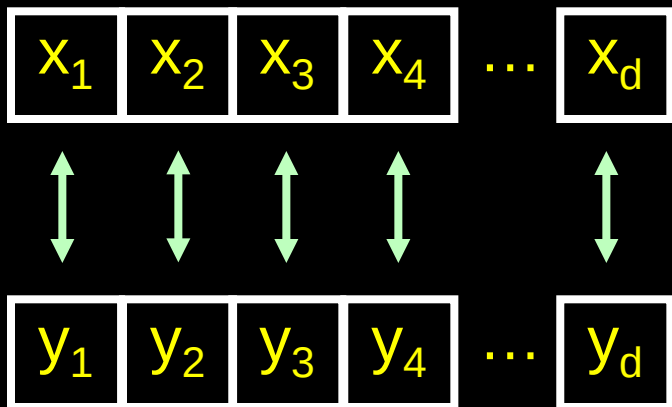
Expensive Distance Measures

- Comparing d -dimensional vectors is efficient:
 - $O(d)$ time.

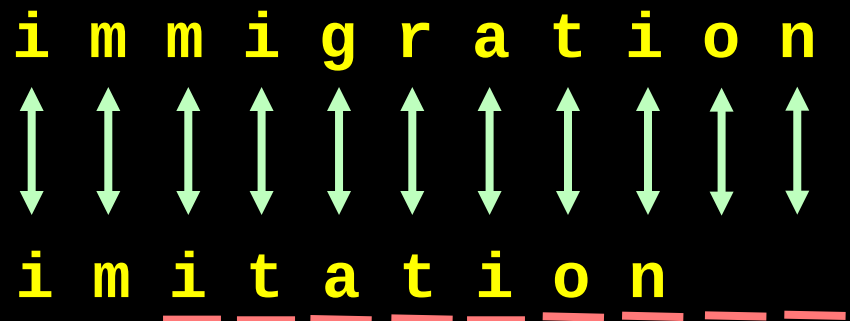


Expensive Distance Measures

- Comparing d-dimensional vectors is efficient:
 - $O(d)$ time.

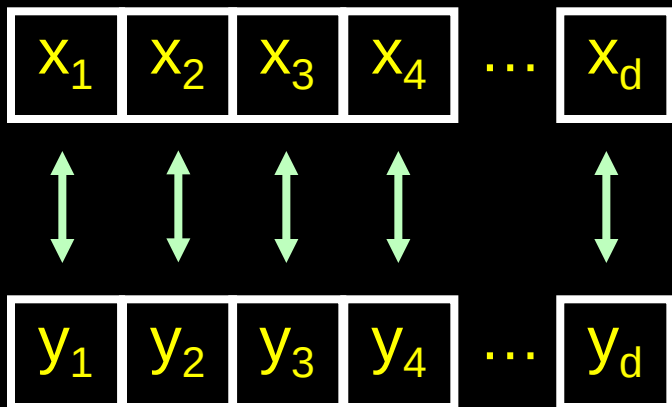


- Comparing strings of length d with the edit distance is more expensive:
 - $O(d^2)$ time.
- Reason: alignment.

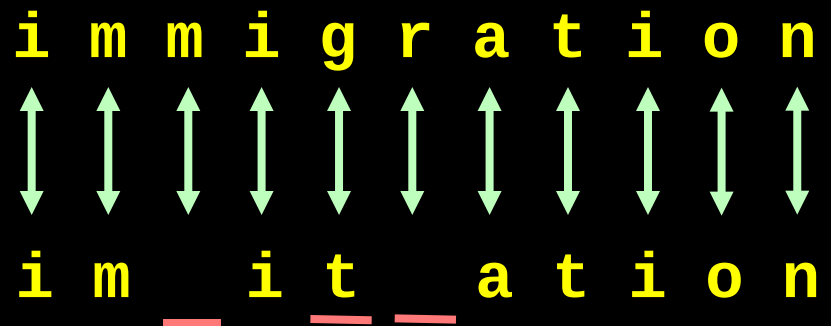


Expensive Distance Measures

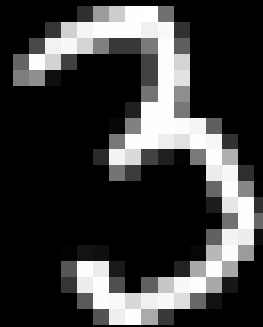
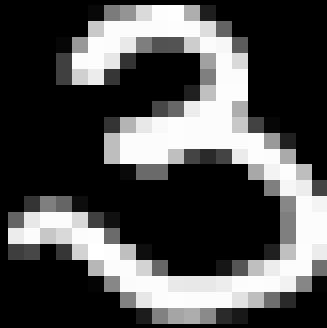
- Comparing d-dimensional vectors is efficient:
 - $O(d)$ time.



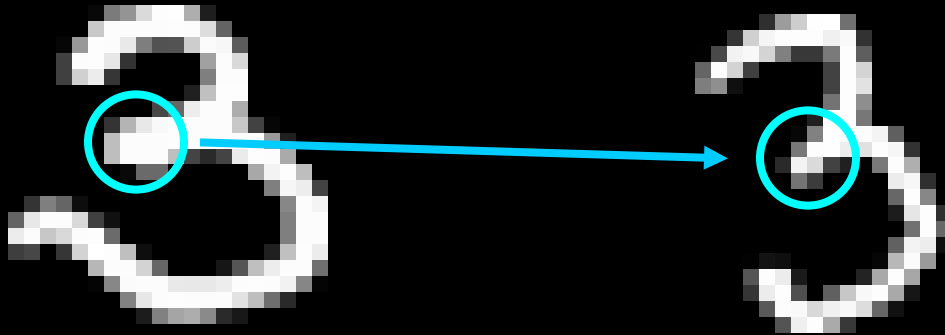
- Comparing strings of length d with the edit distance is more expensive:
 - $O(d^2)$ time.
- Reason: alignment.



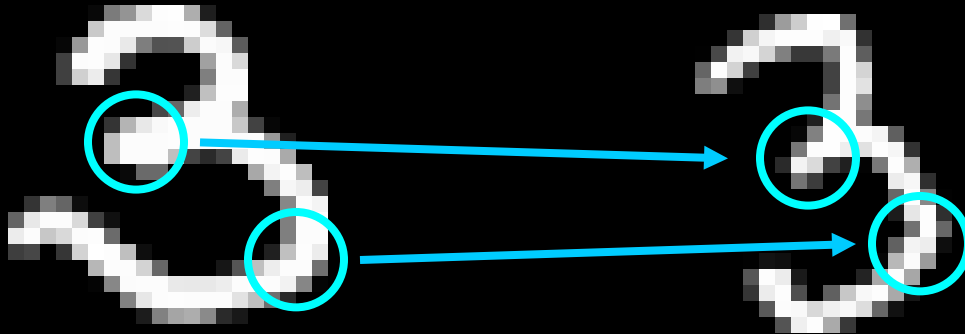
Matching Handwritten Digits



Matching Handwritten Digits



Matching Handwritten Digits



Shape Context Distance



- Proposed by Belongie et al. (2001).
- Database size: 20,000.
- 3-nearest-neighbor error rate: 0.63%.
- Uses bipartite matching.
- 22 minutes/object, heavily optimized.

Hand Shape Classification



query

Hand Shape Classification

Database (80,640 images)



query



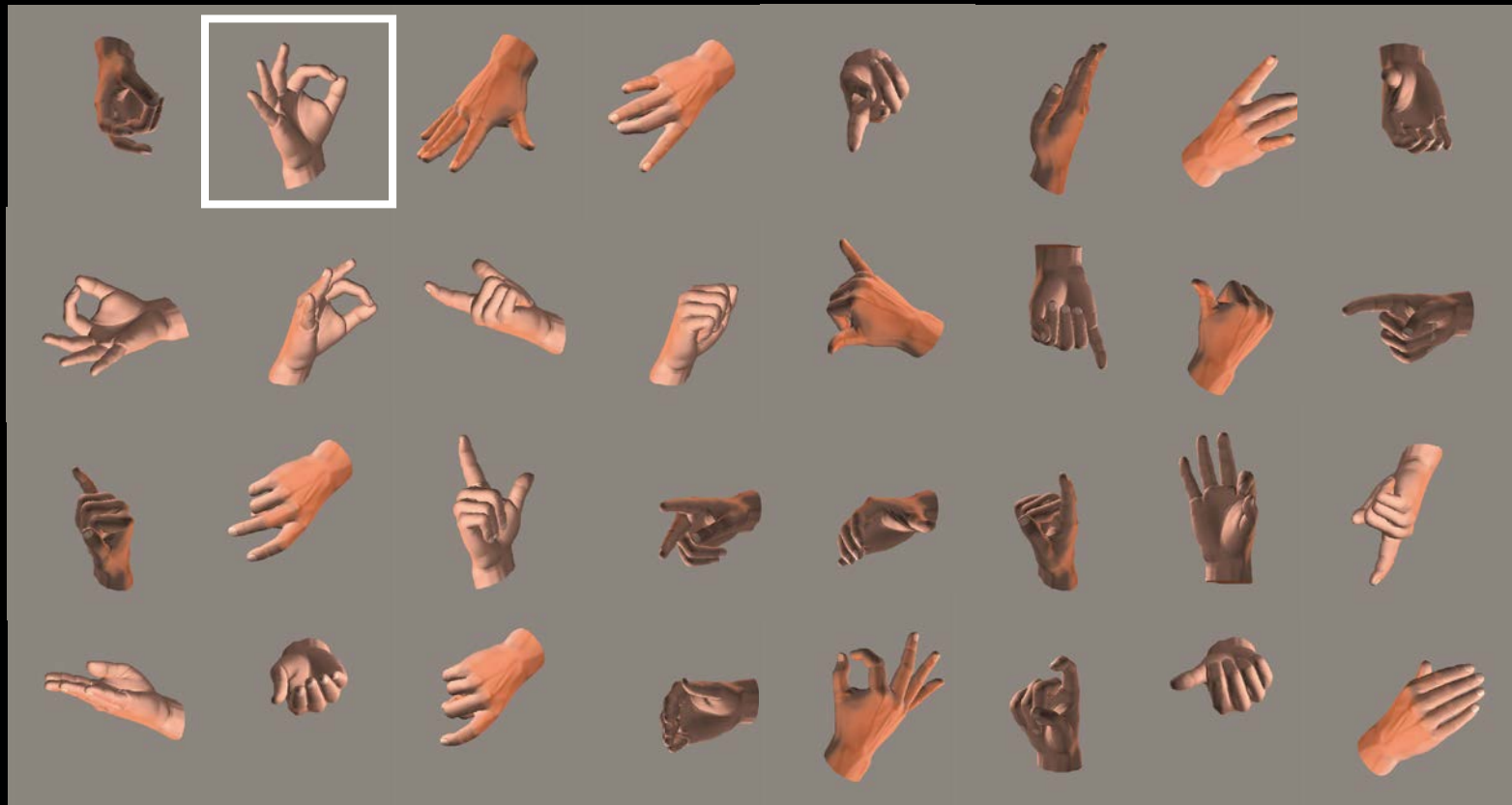
Hand Shape Classification

nearest
neighbor

Database (80,640 images)

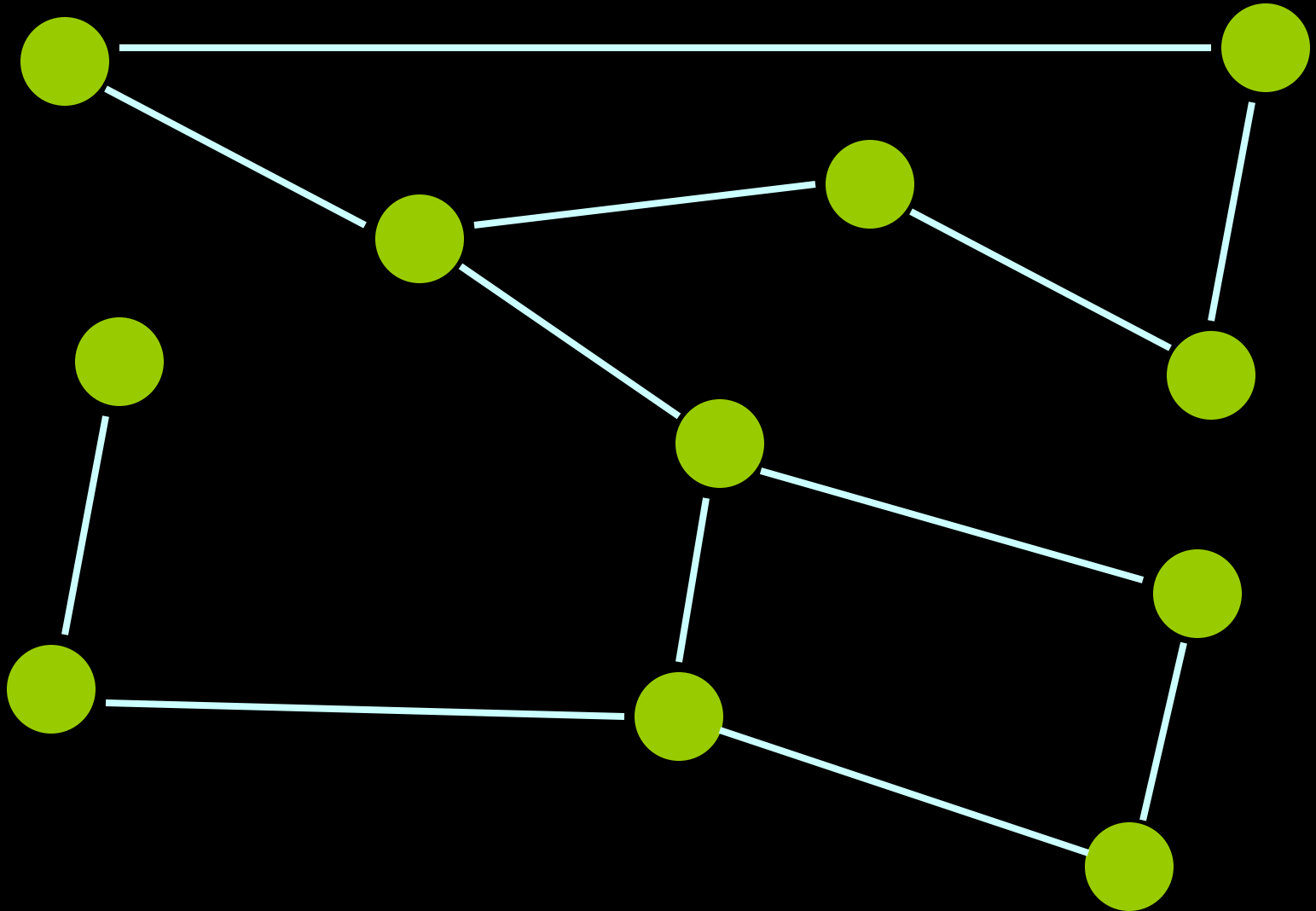


query

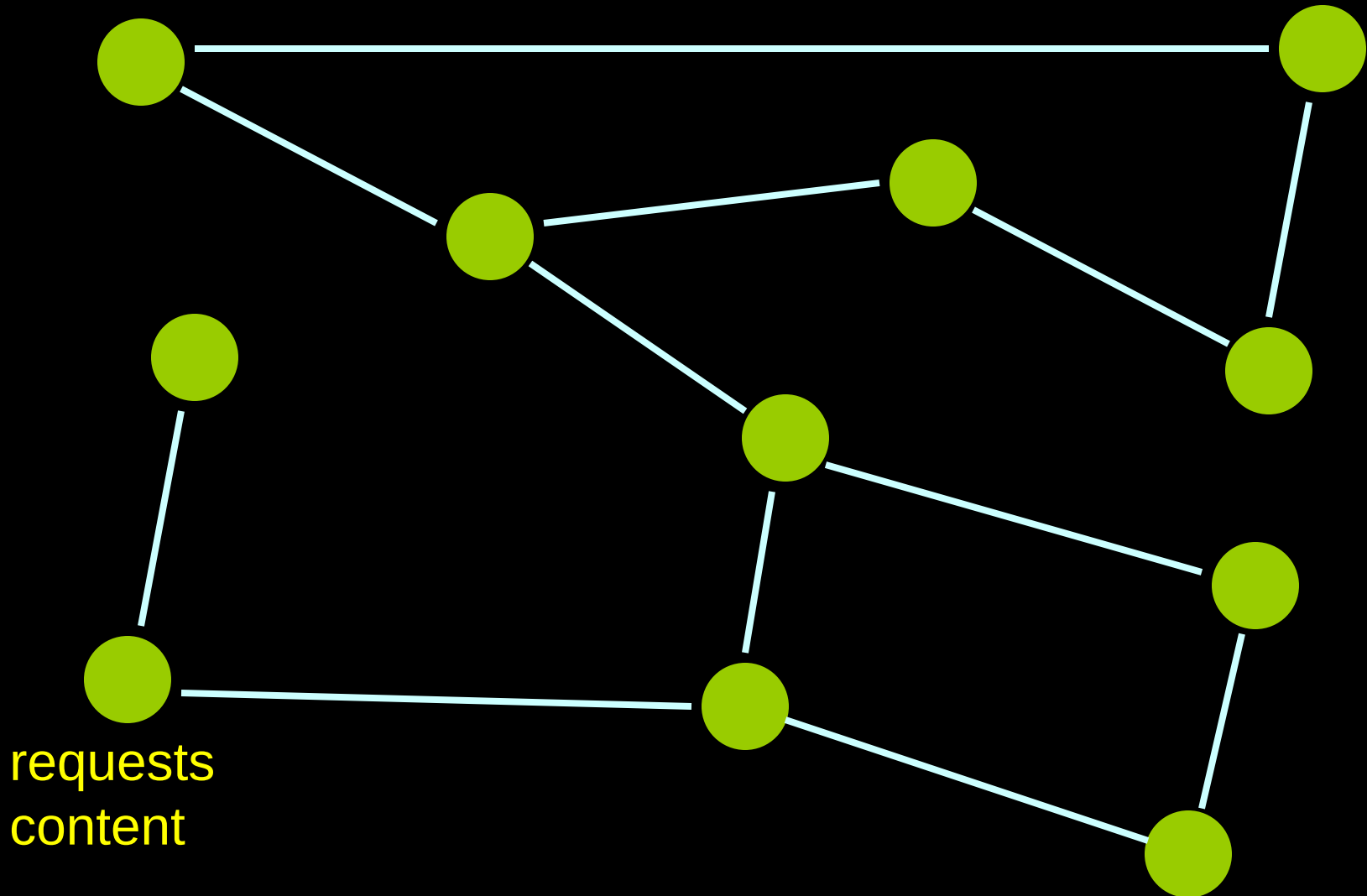


Chamfer distance: 112 seconds per query

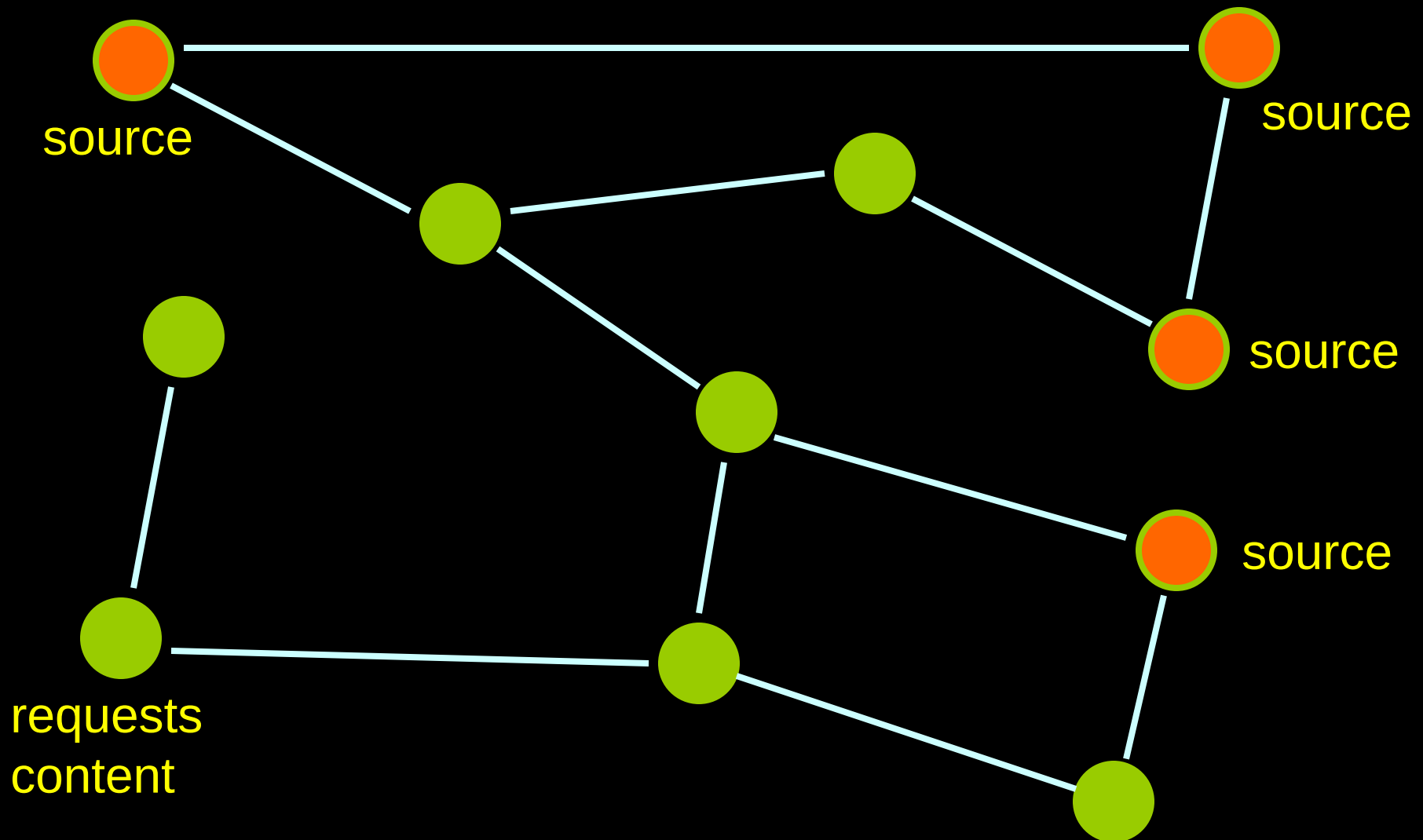
Peer-To-Peer Computer Networks



Peer-To-Peer Computer Networks



Peer-To-Peer Computer Networks



More Examples

- DNA and protein sequences:
 - Smith-Waterman.
- Time series:
 - Dynamic Time Warping.
- Probability distributions:
 - Kullback-Leibler Distance.
- These measures are non-Euclidean, sometimes non-metric.

Indexing Problem

- Vector indexing methods NOT applicable.
 - PCA.
 - R-trees, X-trees, SS-trees.
 - VA-files.
 - Locality Sensitive Hashing.

Distance-Based Indexing

■ Assumptions:

- We have a database and a distance measure.
- We can precompute distances between database objects.

■ Goal, given a query:

- Find (approximate) nearest neighbors of the query, while computing relatively few distances.

Metric Methods

- Pruning-based methods.
 - VP-trees, MVP-trees, M-trees, Slim-trees,...
 - Use triangle inequality for tree-based search.
 - Suffer from curse of dimensionality.
- Filtering methods.
 - AESA, LAESA...
 - Use the triangle inequality to compute upper/lower bounds of distances.
- Inexact and heuristic in non-metric spaces.

Embeddings

database

x_1

x_2

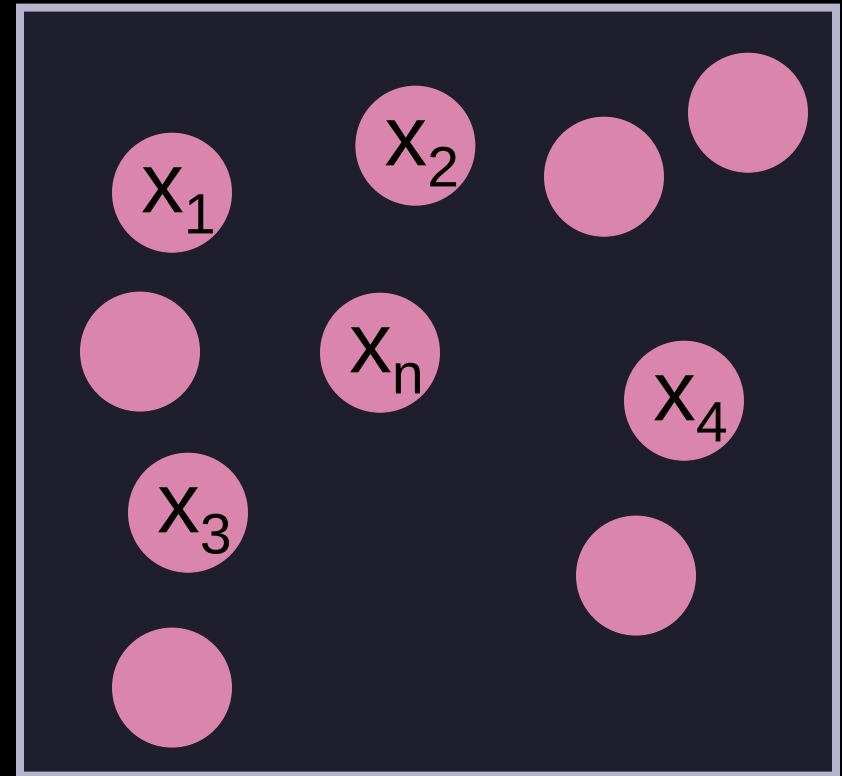
x_3

⋮

x_n

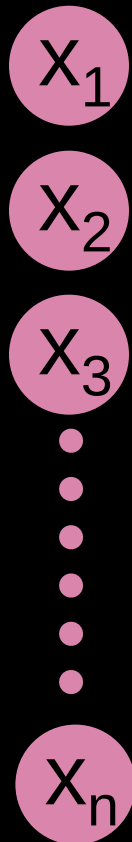
embedding
 F

$\mathbb{R}^d$



Embeddings

database



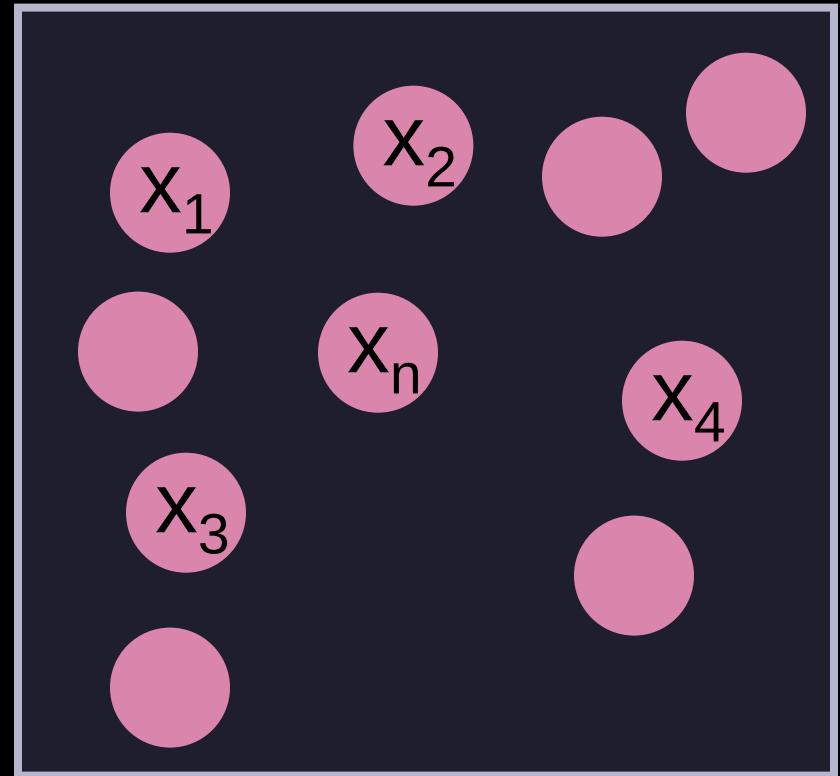
embedding
 F



query

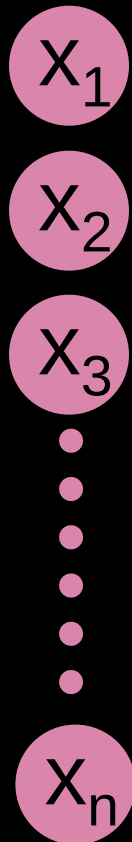


$\mathbb{R}^d$



Embeddings

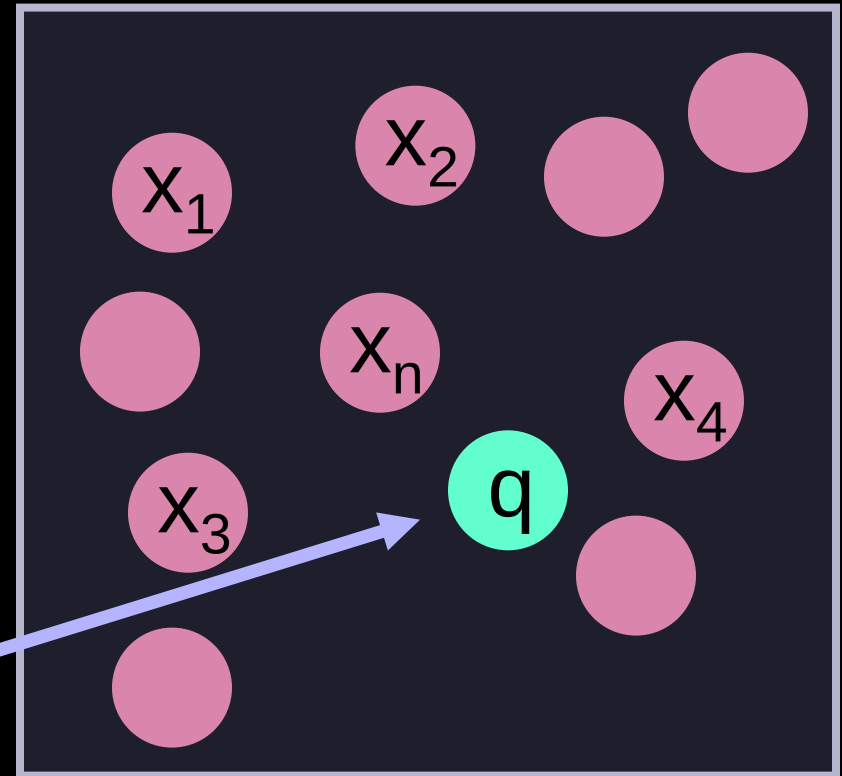
database



embedding
 F



$\mathbb{R}^d$



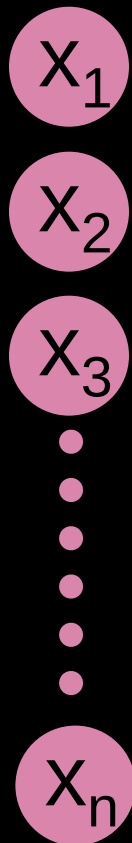
query



Embeddings

- Measure distances between vectors (typically much faster).

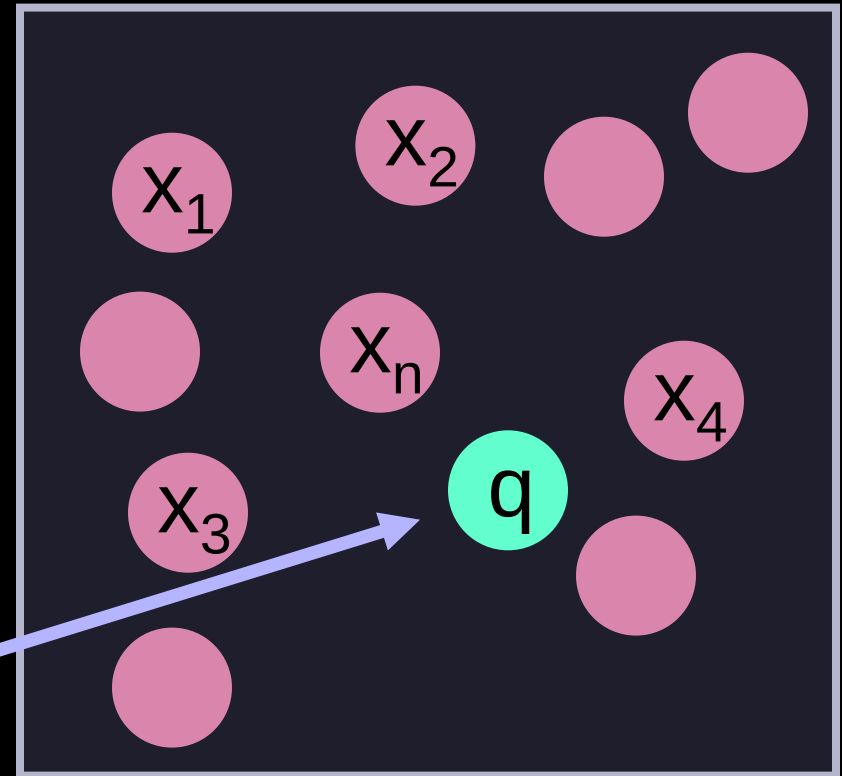
database



embedding
 F



$\mathbb{R}^d$



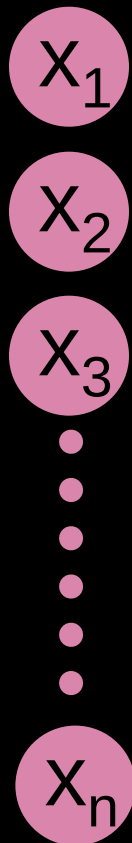
query



Embeddings

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

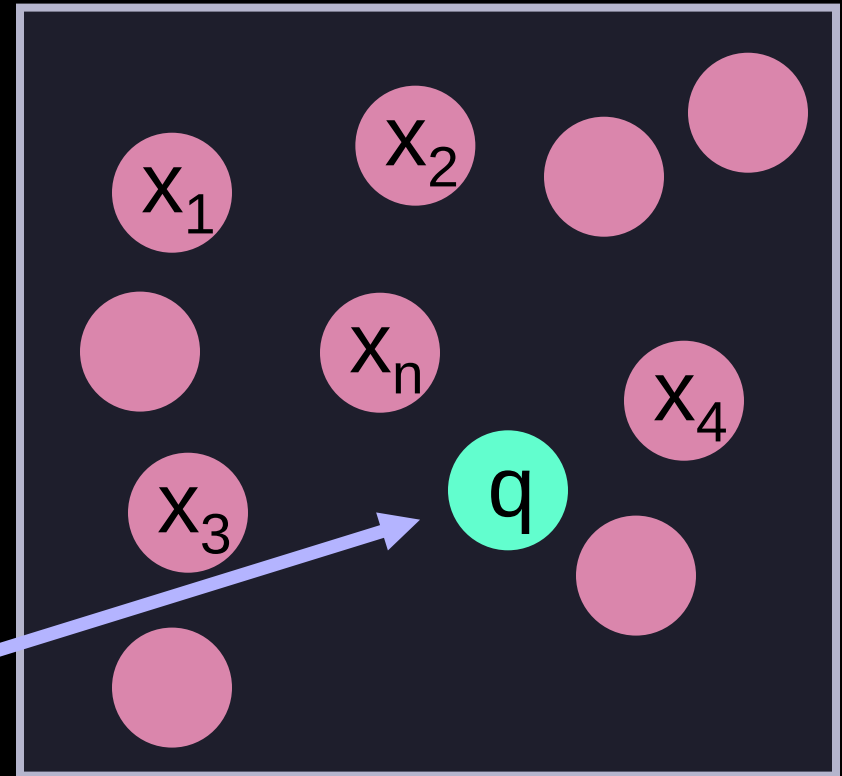
database



embedding
 F



$\mathbb{R}^d$

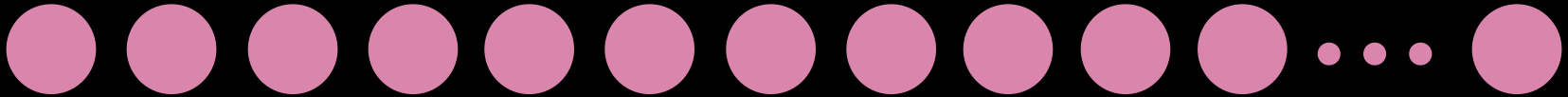


query



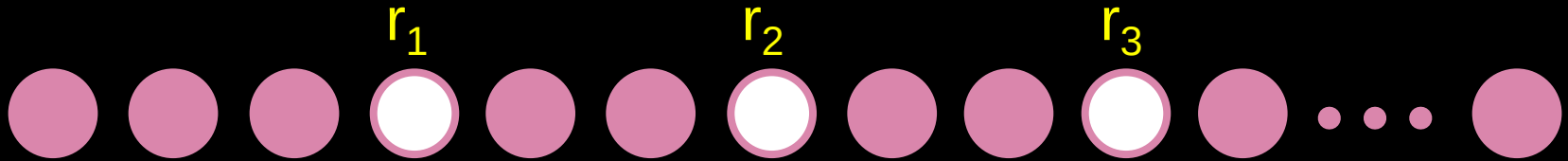
Reference Object Embeddings

database



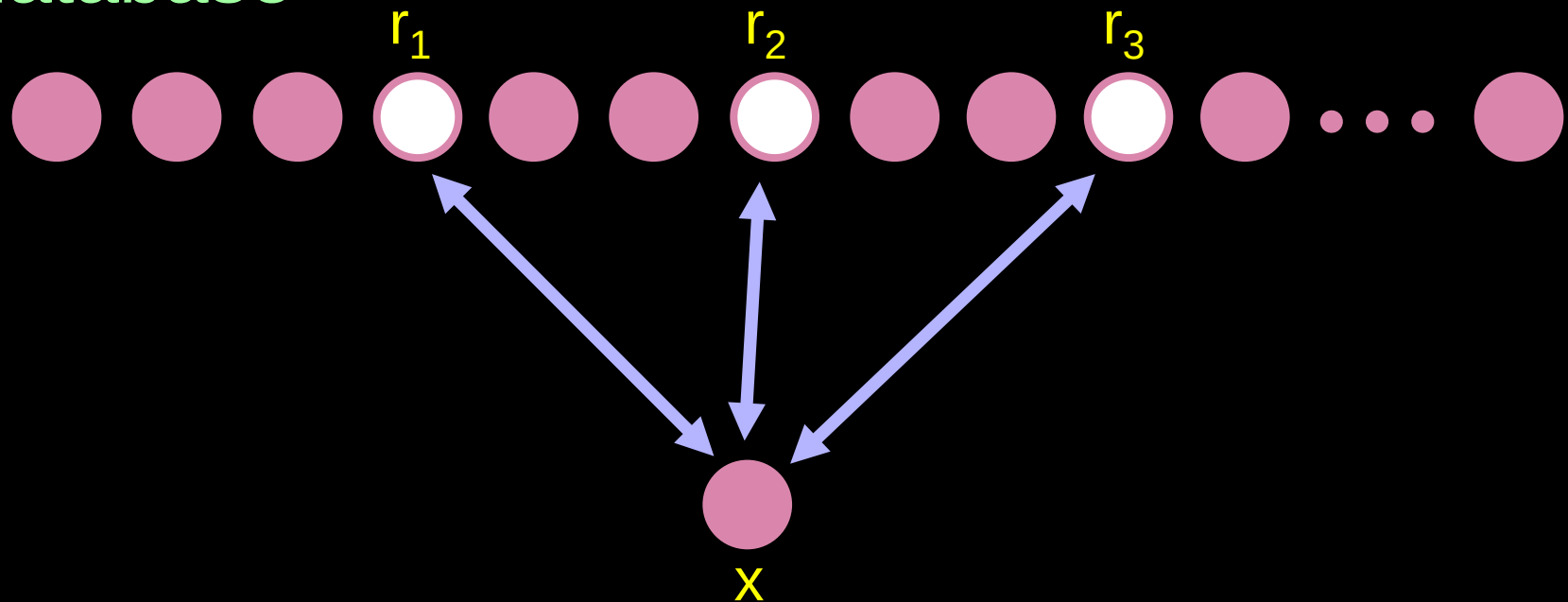
Reference Object Embeddings

database



Reference Object Embeddings

database



$$F(x) = (D(x, r_1), D(x, r_2), D(x, r_3))$$

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



$F(\text{Sacramento}) \dots = (386, 1543, 2920)$

$F(\text{Las Vegas}) \dots = (262, 1232, 2405)$

$F(\text{Oklahoma City}) = (1345, 437, 1291)$

$F(\text{Washington DC}) = (2657, 1207, 853)$

$F(\text{Jacksonville}) = (2422, 1344, 141)$

Existing Embedding Methods

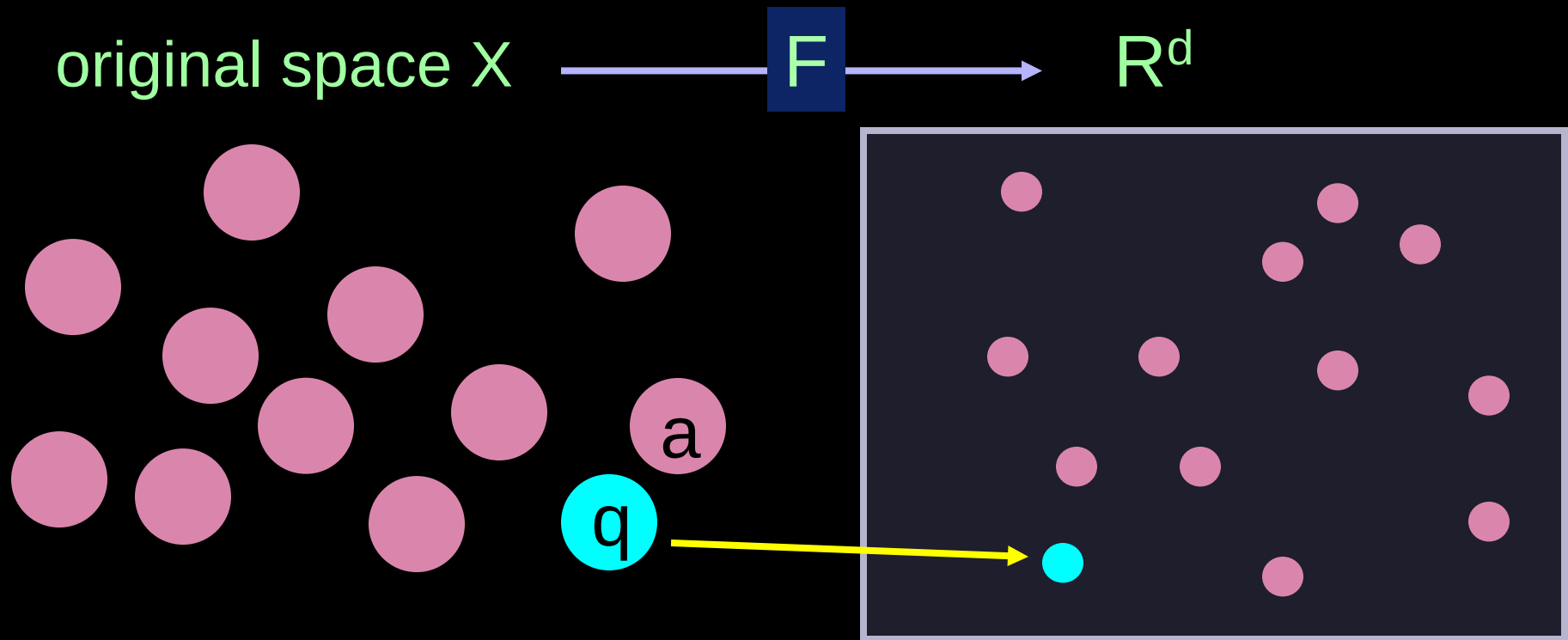
- FastMap, MetricMap, SparseMap, Lipschitz embeddings.
 - Use distances to reference objects (prototypes).
- Question: how do we directly optimize an embedding for nearest neighbor retrieval?
 - FastMap & MetricMap assume Euclidean properties.
 - SparseMap optimizes stress.
 - Large stress may be inevitable when embedding non-metric spaces into a metric space.

Key Features of Our Method

■ BoostMap:

- Maximizes amount of nearest neighbor structure preserved by the embedding.
- Based on machine learning, not on geometric assumptions.
- Combines efficiency of measuring distances in vector spaces with ability to capture non-metric structure.
 - Query-sensitive version of BoostMap.

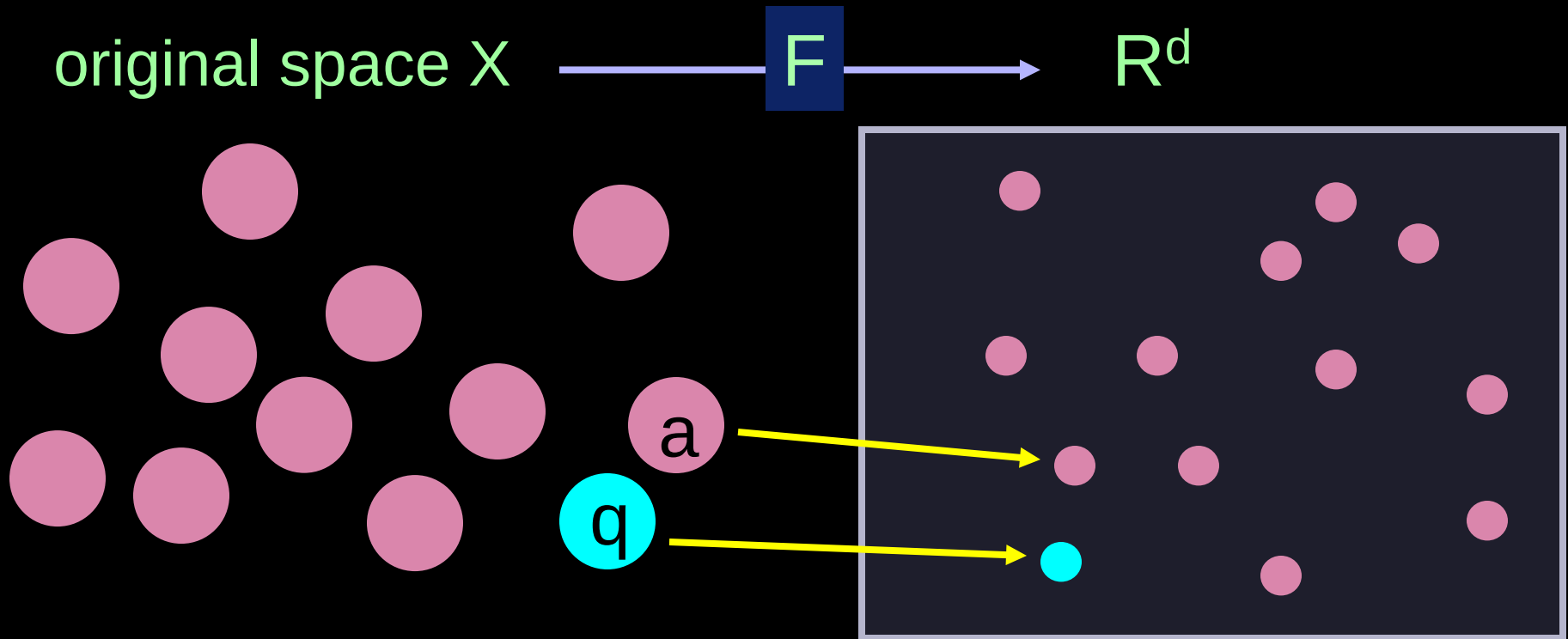
Ideal Embedding Behavior



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

For any q : we want $F(NN(q)) = NN(F(q))$.

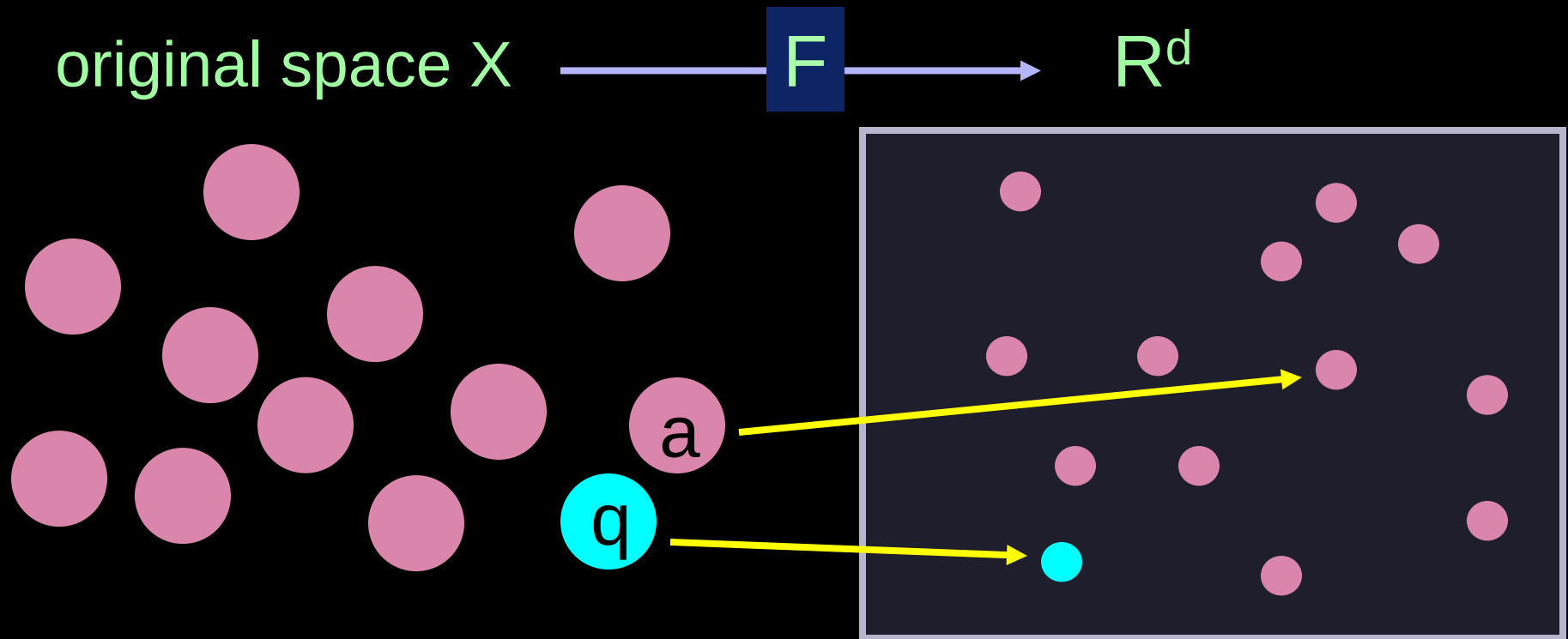
Ideal Embedding Behavior



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

For any q : we want $F(NN(q)) = NN(F(q))$.

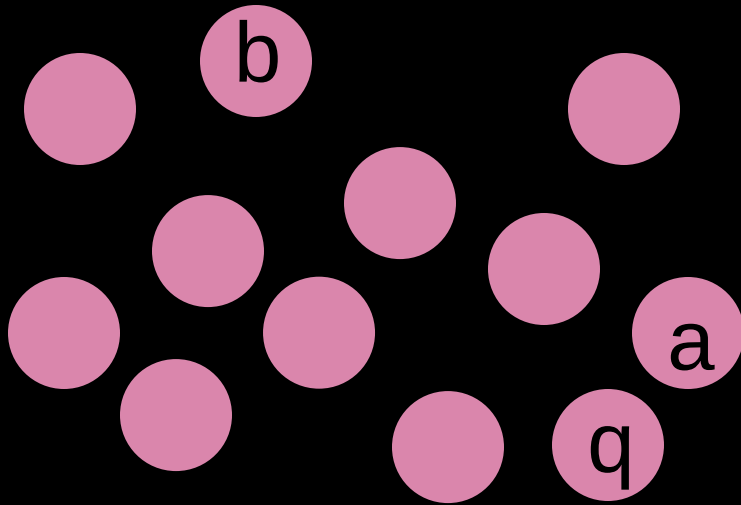
Ideal Embedding Behavior



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

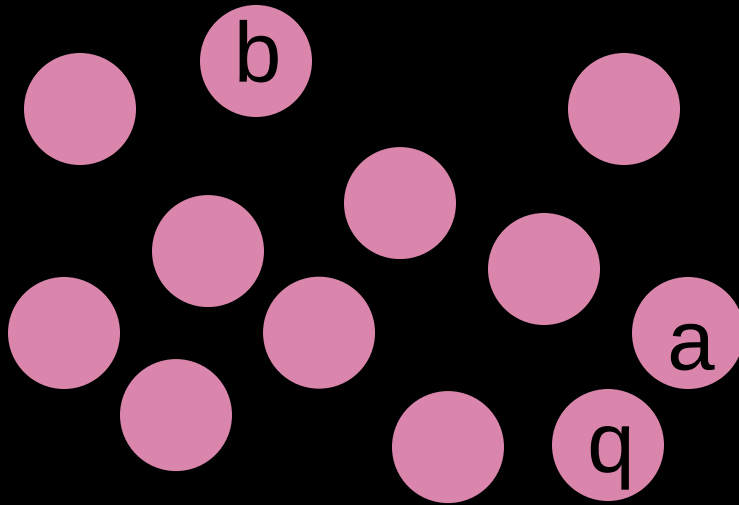
For any q : we want $F(NN(q)) = NN(F(q))$.

Embeddings Seen As Classifiers



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

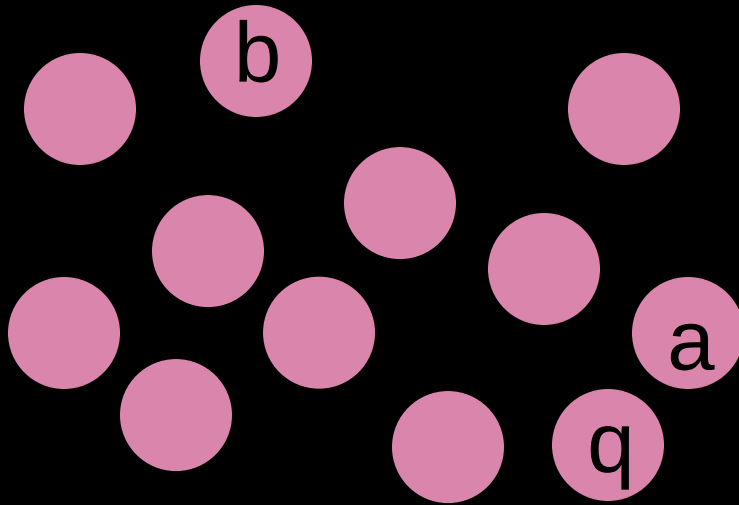
Embeddings Seen As Classifiers



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 $F(b)$.

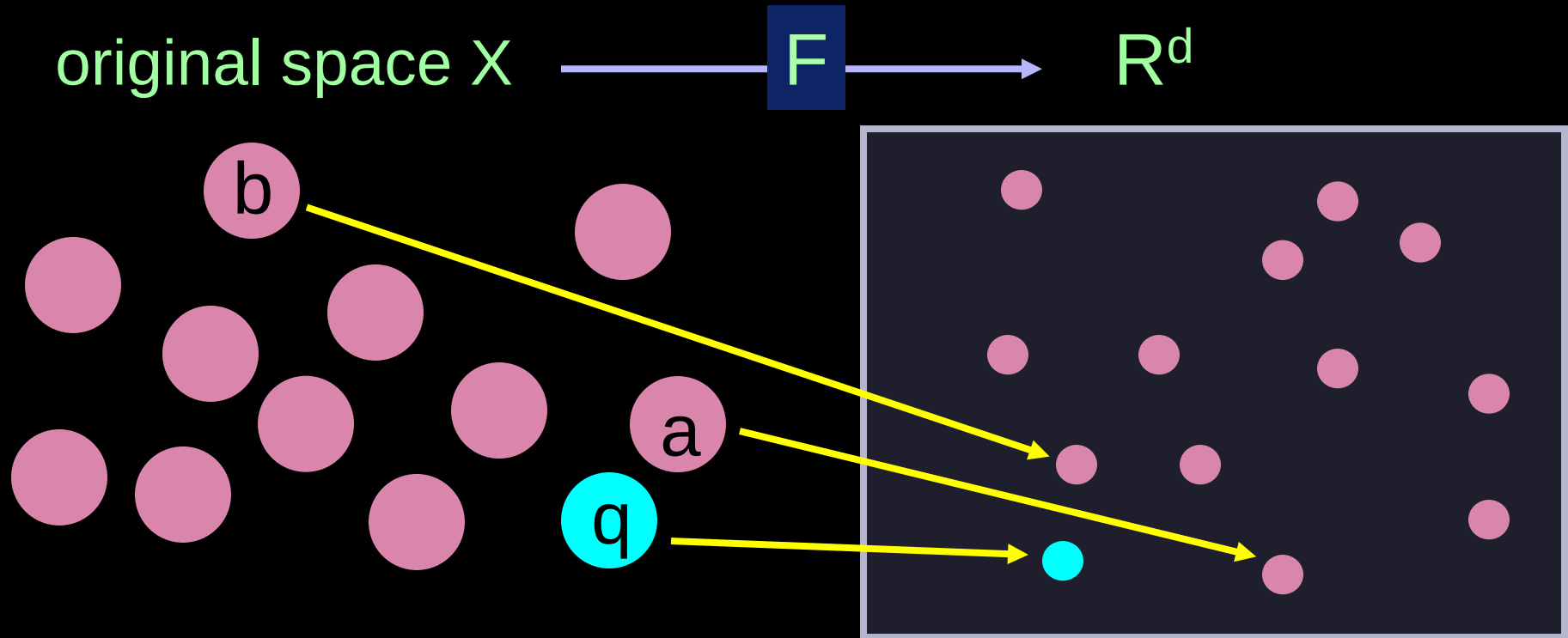
Classifier Definition



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

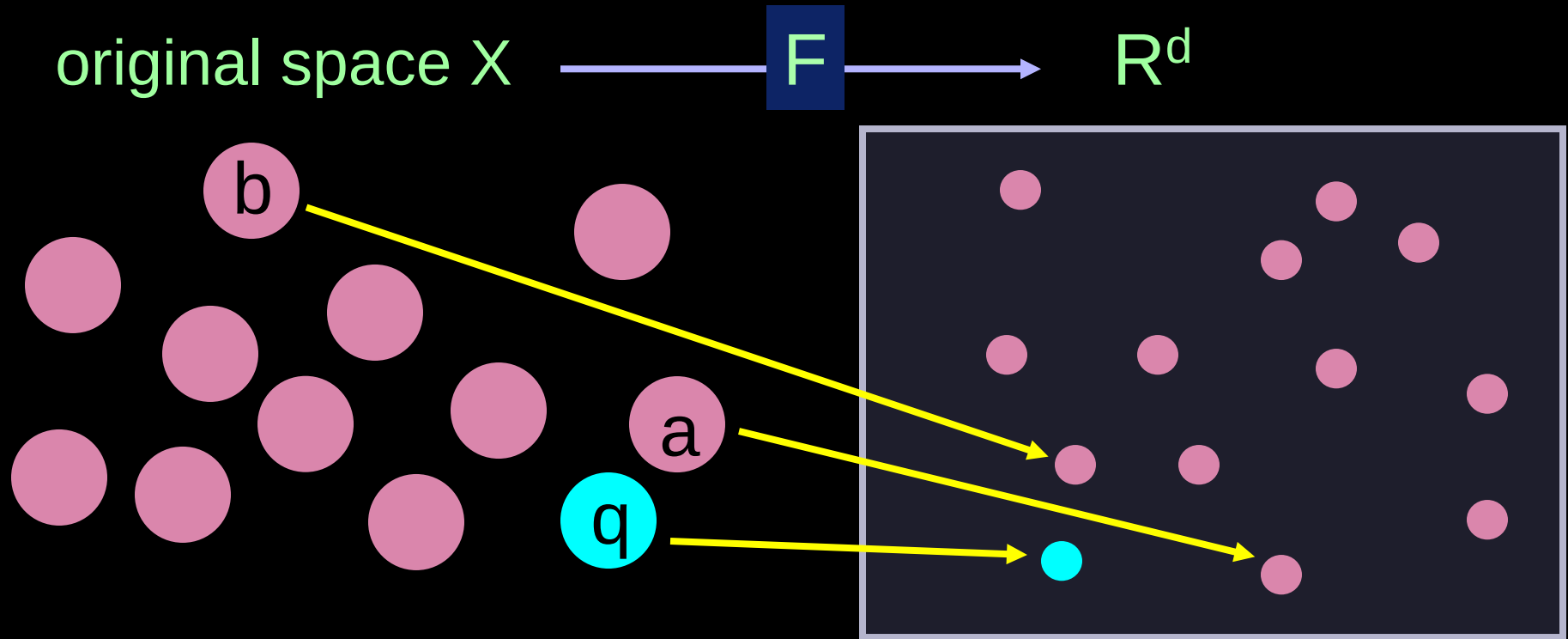
- Given embedding $F: X \rightarrow \mathbb{R}^d$:
 - $F'(q, a, b) = \|F(q) - F(b)\| - \|F(q) - F(a)\|$.
- $F'(q, a, b) > 0$ means “ q is closer to a .”
- $F'(q, a, b) < 0$ means “ q is closer to b .”

Key Observation



- If classifier F' is perfect, then for every q , $F(\text{NN}(q)) = \text{NN}(F(q))$.
 - If $F(q)$ is closer to $F(b)$ than to $F(\text{NN}(q))$, then triple (q, a, b) is misclassified.

Key Observation



- Classification error on triples $(q, \text{NN}(q), b)$ measures how well F preserves nearest neighbor structure.

Optimization Criterion

- Goal: construct an embedding F optimized for k -nearest neighbor retrieval.
- Method: maximize accuracy of F' on triples (q, a, b) of the following type:
 - q is any object.
 - a is a k -nearest neighbor of q in the database.
 - b is in database, but NOT a k -nearest neighbor of q .
- If F' is perfect on those triples, then F perfectly preserves k -nearest neighbors.

Overview of Strategy

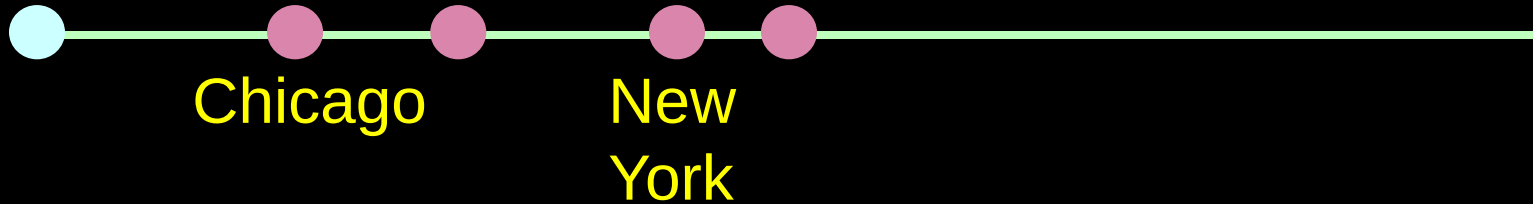
- Start with simple 1D embeddings.
- Convert 1D embeddings to classifiers.
- Combine those classifiers into a single, optimized classifier.
- Convert optimized classifier into a multidimensional embedding.

1D Embeddings as Weak Classifiers

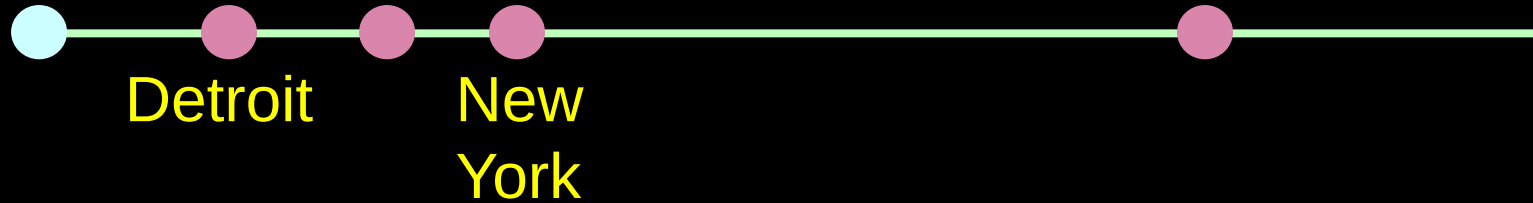
- 1D embeddings define *weak classifiers*.
 - Better than a random classifier (50% error rate).



Lincoln Detroit LA



Cleveland Chicago LA



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.

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

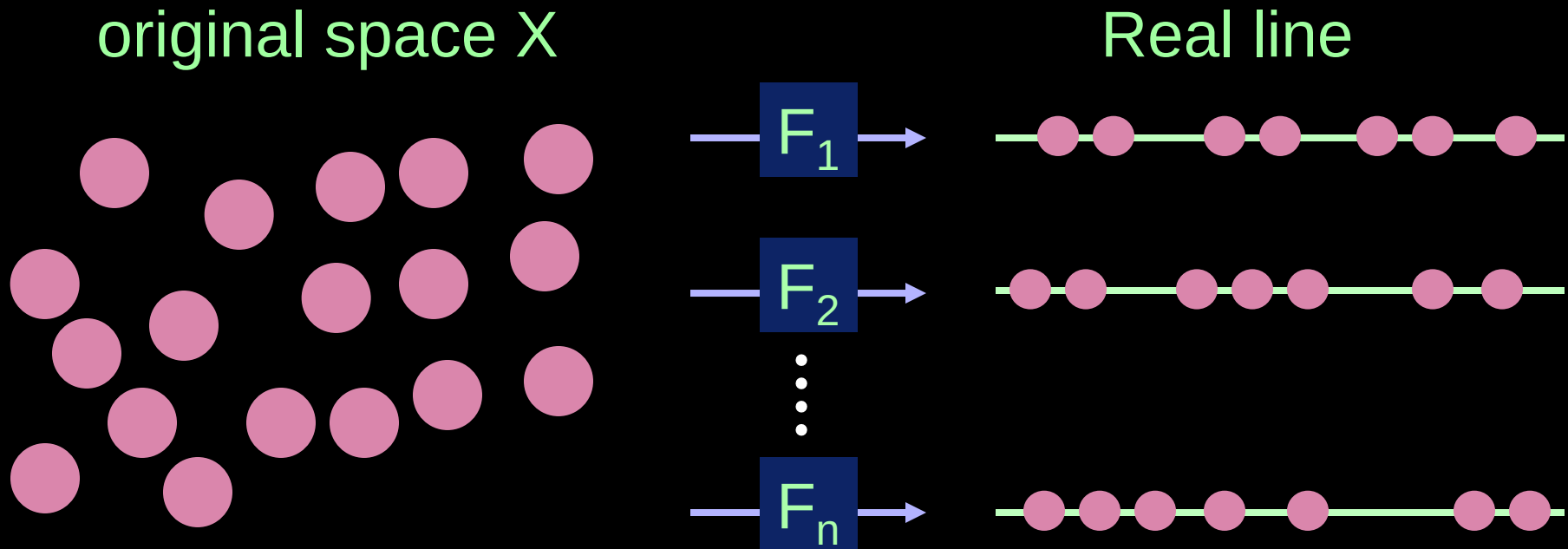
- 1D embeddings define *weak classifiers*.
 - Better than a random classifier (50% error rate).
- We can define lots of different classifiers.
 - Every object in the database can be a reference object.

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: $H = w_1 F'_1 + w_2 F'_2 + \dots + w_d F'_d$.
 - 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

$$H = w_1 F'_1 + w_2 F'_2 + \dots + w_d F'_d$$

What embedding should we use?

What distance measure should we use?

From Classifier to Embedding

AdaBoost output $H = w_1 F'_1 + w_2 F'_2 + \dots + w_d F'_d$

BoostMap
embedding $F(x) = (F_1(x), \dots, F_d(x)).$

From Classifier to Embedding

AdaBoost output $H = w_1 F'_1 + w_2 F'_2 + \dots + w_d F'_d$

BoostMap
embedding $F(x) = (F_1(x), \dots, F_d(x)).$

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

From Classifier to Embedding

AdaBoost output $H = w_1 F'_1 + w_2 F'_2 + \dots + w_d F'_d$

BoostMap
embedding $F(x) = (F_1(x), \dots, F_d(x)).$

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

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.
- More general proof in thesis:
 - If $h_1, \dots, h_m$ are classifiers corresponding to embeddings, any linear combination of h_i 's corresponds to an embedding.

How Do We Use It?

Filter-and-refine retrieval:

- Offline step: compute embedding F of entire database.
- Given a query object q :
 - Embedding step:
 - Compute distances from query to reference objects $\rightarrow F(q)$.
 - 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 $\rightarrow F(q)$.
- 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 $\rightarrow F(q)$.
- 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 $\rightarrow F(q)$.
- 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 $\rightarrow F(q)$.
- 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 $\rightarrow F(q)$.

- 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

What is the nearest neighbor classification error?

How many exact distance computations do we need?

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

Methods

- Brute Force.
- BoostMap (BM).
- FastMap (FM).
- VP-trees (VP).
- Random Reference Objects (RRO).
- Random Line Projections (RLP).

Results on Hand Dataset



Chamfer distance: 112 seconds per query

Results on Hand Dataset

Database: 80,640 synthetic images of hands.

Query set: 710 real images of hands.

	Brute Force
Accuracy	100%
Distances	80640
Seconds	112
Speed-up	1

Results on Hand Dataset

Database: 80,640 synthetic images of hands.

Query set: 710 real images of hands.

	Brute Force	BM	RLP	FM	VP
Accuracy	100%	95%	95%	95%	95%
Distances	80640	450	1444	2647	5471
Seconds	112	0.6	2.0	3.7	7.6
Speed-up	1	179	56	30	15

Results on Hand Dataset

Database: 80,640 synthetic images of hands.

Query set: 710 real images of hands.

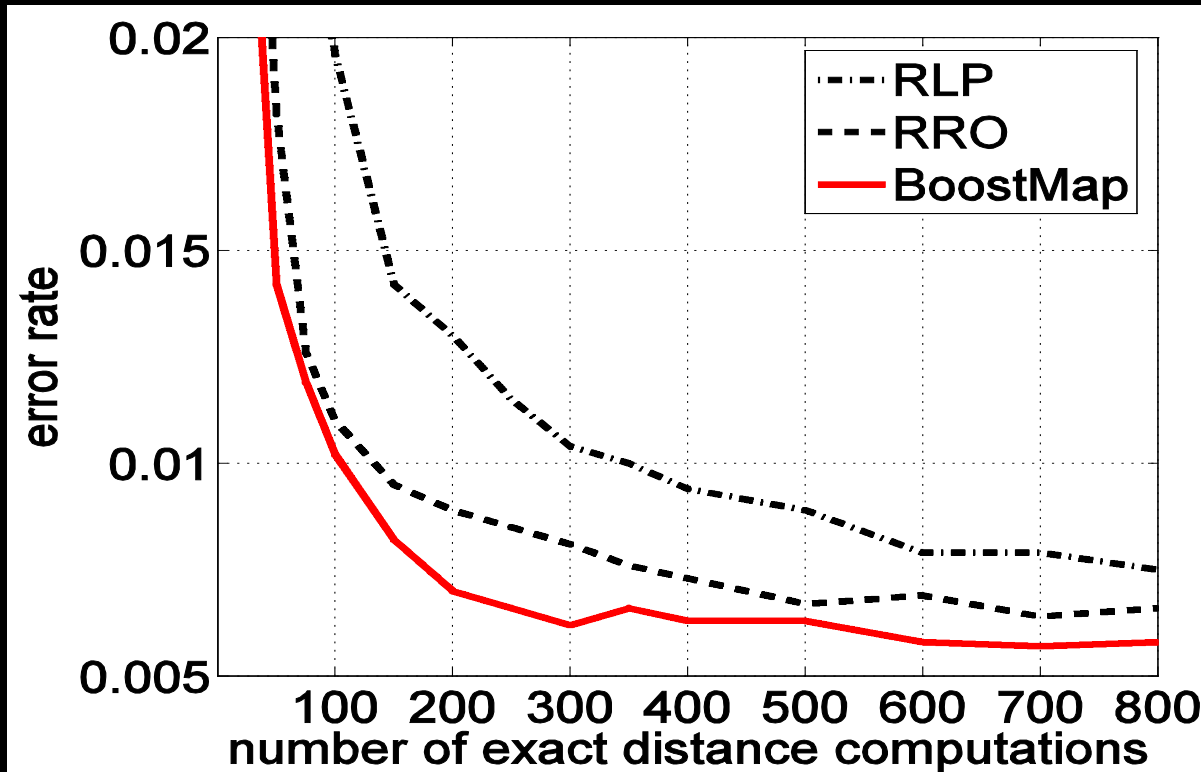
	Brute Force	BM	RLP	FM	VP
Accuracy	100%	100%	100%	100%	100%
Distances	80640	5995	13235	16242	12012
Seconds	112	8.3	18.4	22.6	16.7
Speed-up	1	13.5	6.1	5.0	6.7

Results on MNIST Dataset



- MNIST: 60,000 database objects, 10,000 queries.
- Shape context (Belongie 2001):
 - 0.63% error, 20,000 distances, 22 minutes.
 - 0.54% error, 60,000 distances, 66 minutes

Results on MNIST Dataset



Distances	50	100	200	300	400	600	800
BoostMap	1.42	1.02	0.70	0.62	0.63	0.58	0.58
RRO	1.82	1.10	0.89	0.81	0.73	0.69	0.66

Results on MNIST Dataset

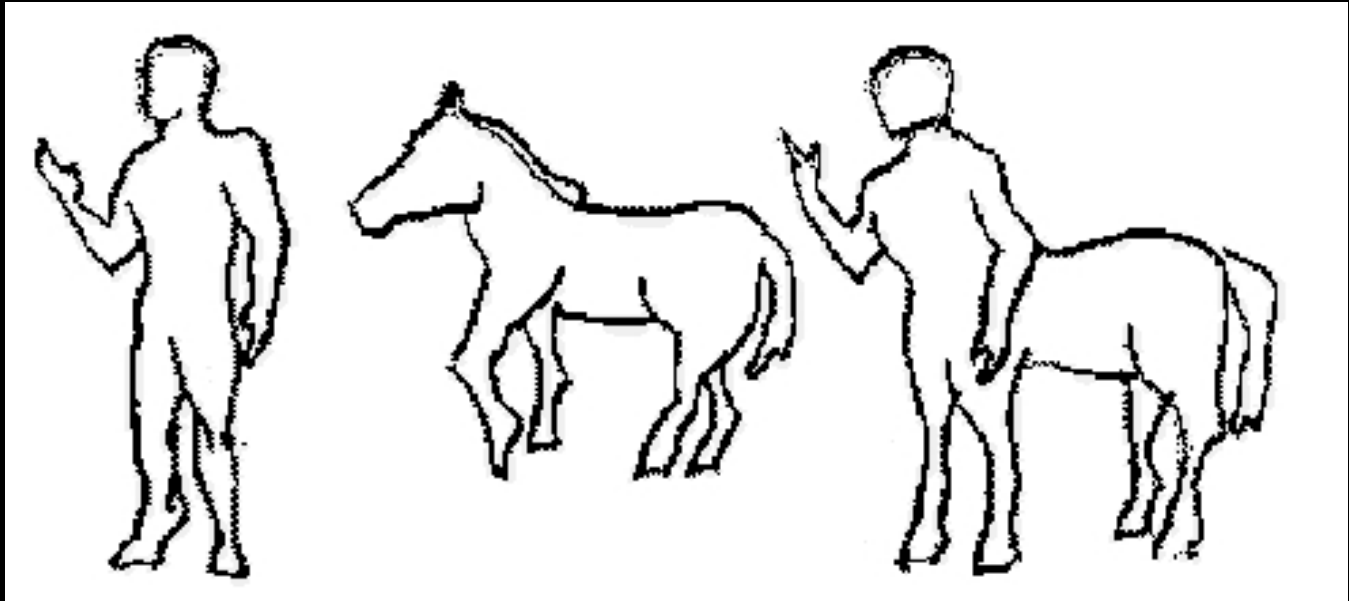
- Zhang et al. (2003): custom-made method for speeding up shape context classification.
 - Only 50 distances evaluated.
 - Time: 3.5 seconds per object.
 - Error rate: 2.55%.
- BoostMap, with 50 distance evaluations:
 - Error rate: 1.42%.

Key Features of Our Method

■ BoostMap:

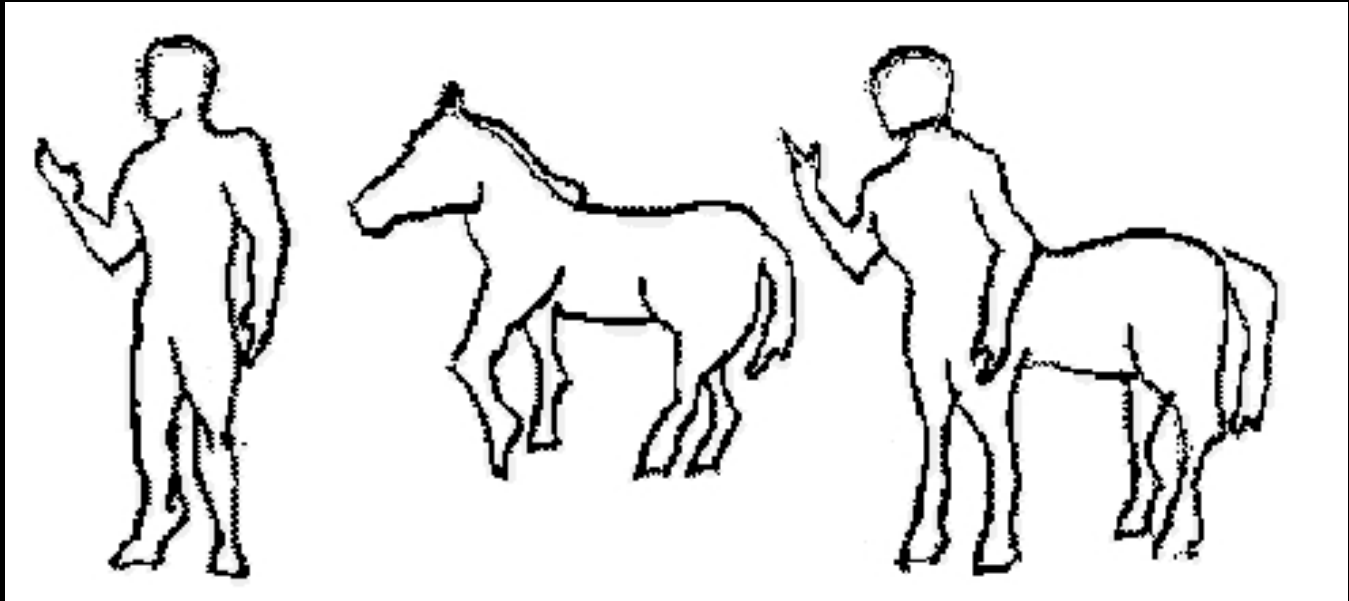
- ✓ Maximizes amount of nearest neighbor structure preserved by the embedding.
- Based on machine learning, not on geometric assumptions.
- Combines efficiency of measuring distances in vector spaces with ability to capture non-metric structure.

Capturing Non-Metric Structure



- A human is *not* similar to a horse.
- A centaur is similar *both* to a human and a horse.
- Triangle inequality is violated:
 - Using human ratings of similarity (Tversky, 1982).
 - Using k-median Hausdorff distance.

Capturing Non-Metric Structure



- Mapping to a metric space presents dilemma:
 - If $D(F(\text{centaur}), F(\text{human})) = D(F(\text{centaur}), F(\text{horse})) = C$, then $D(F(\text{human}), F(\text{horse})) \leq 2C$.
- Query-sensitive embeddings:
 - Have the modeling power to preserve non-metric structure.

Query-Sensitive Embeddings

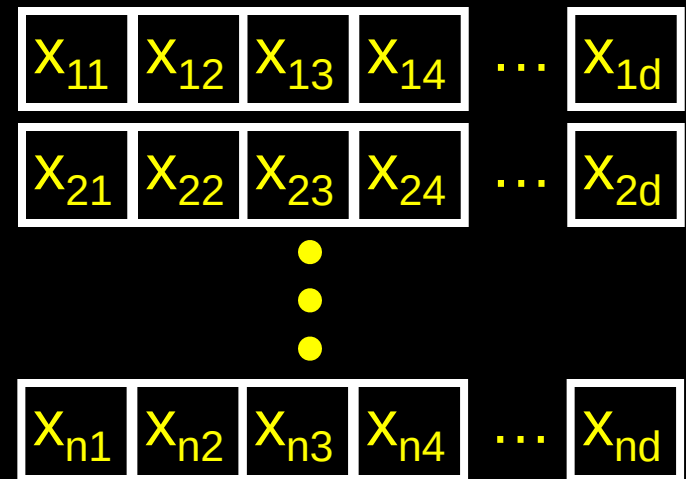
- How important is each coordinate in comparing embeddings?

database

x_1 ●
 x_2 ●
⋮
 x_n ●

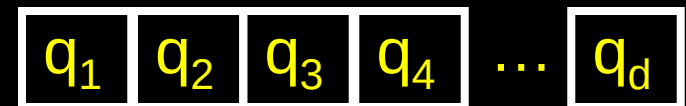
embedding
 F

$\mathbb{R}^d$



query

q ●



Query-Sensitive Embeddings

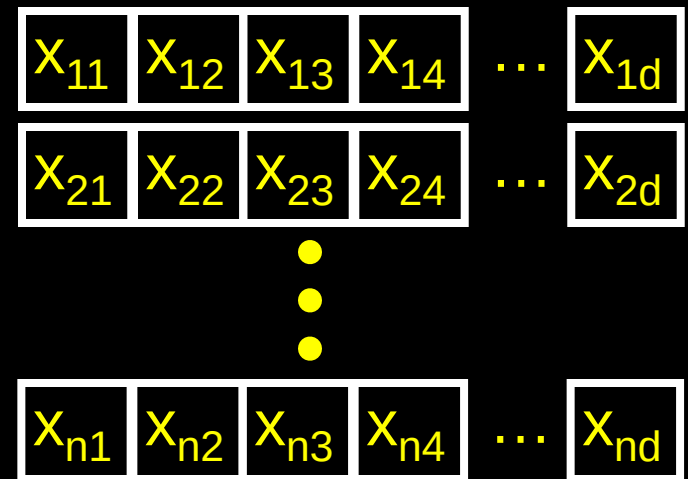
- Given a query, automatically choose the weight of each coordinate.

database

x_1 ●
 x_2 ●
⋮
 x_n ●

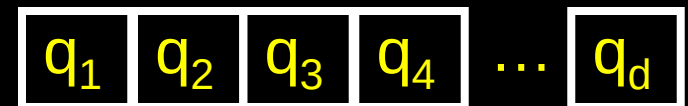
embedding
 F

$\mathbb{R}^d$



query

q ●



Query-Sensitive Embeddings

- Given a query, automatically choose the weight of each coordinate.

database

x_1 ●

x_2 ●

⋮

x_n ●

query

q ●

embedding
 F

$\mathbb{R}^d$

x_{11}	x_{12}	x_{13}	x_{14}	...	x_{1d}
x_{21}	x_{22}	x_{23}	x_{24}	...	x_{2d}
		⋮			
x_{n1}	x_{n2}	x_{n3}	x_{n4}	...	x_{nd}
q_1	q_2	q_3	q_4	...	q_d

Query-Sensitive Embeddings

- Given a query, automatically choose the weight of each coordinate.

database

x_1 ●

x_2 ●

⋮

x_n ●

query

q ●

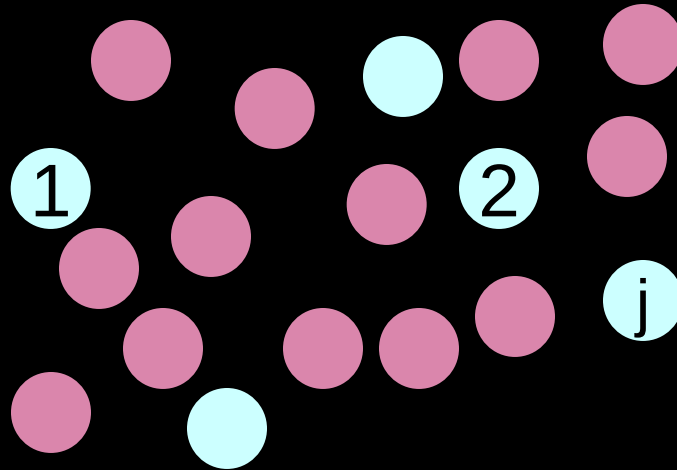
embedding
 F

$\mathbb{R}^d$

x_{11}	x_{12}	x_{13}	x_{14}	...	x_{1d}
x_{21}	x_{22}	x_{23}	x_{24}	...	x_{2d}
		⋮			
x_{n1}	x_{n2}	x_{n3}	x_{n4}	...	x_{nd}
q_1	q_2	q_3	q_4	...	q_d

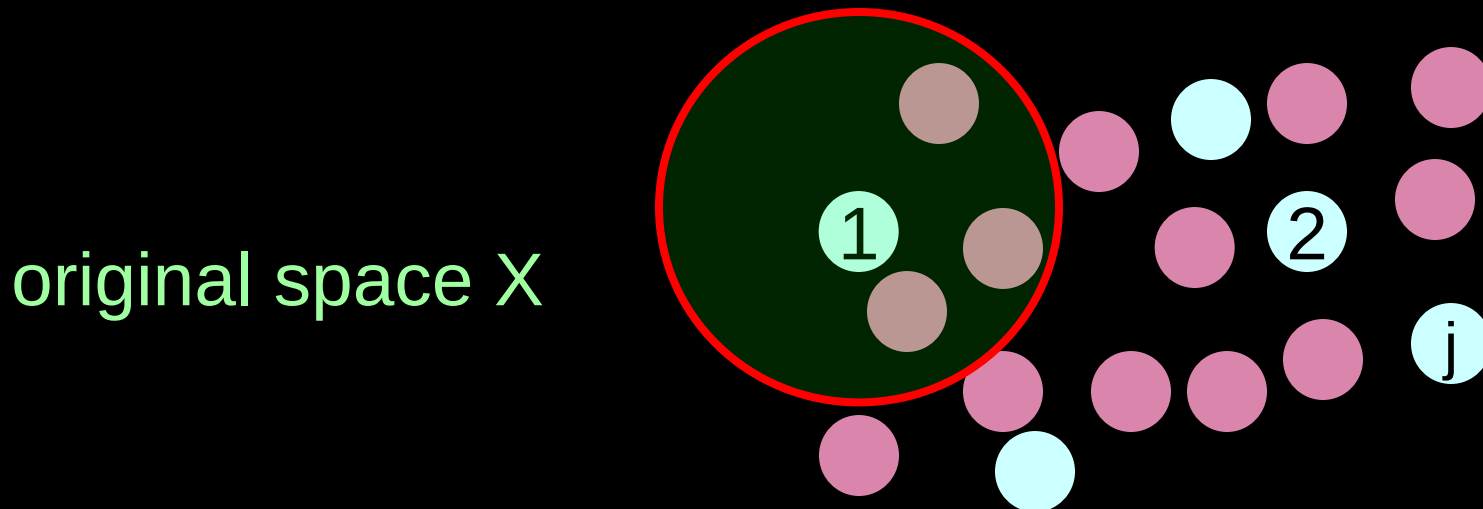
Motivation

original space X



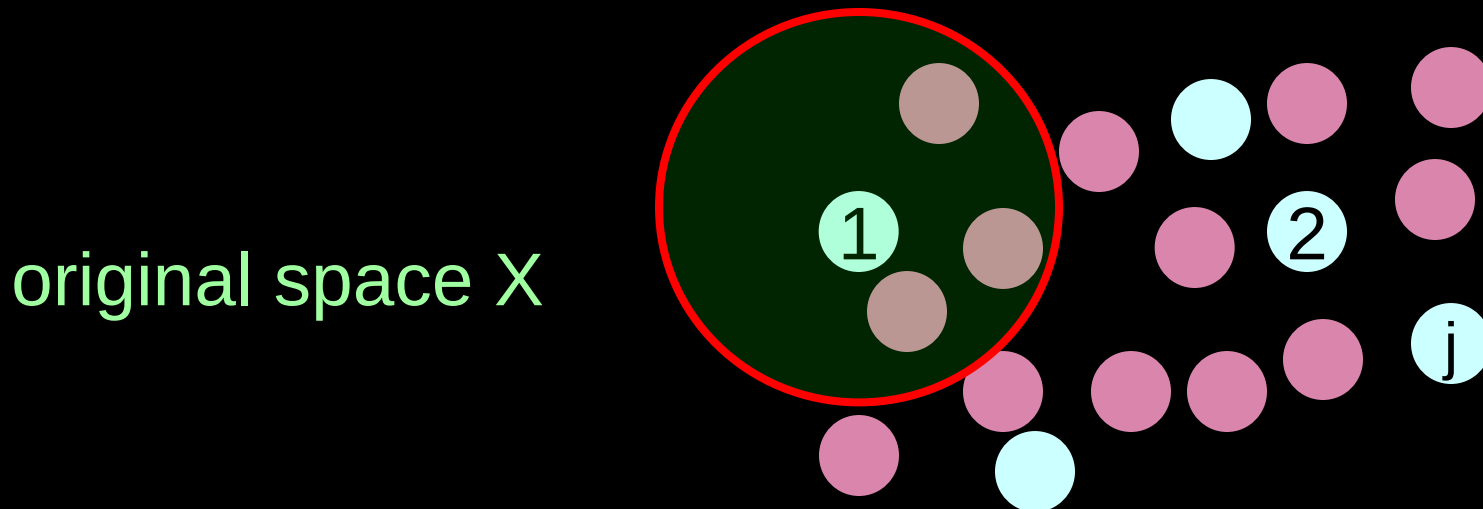
- Classifier: $H = a_1 F'_1 + a_2 F'_2 + \dots + a_j F'_j$.
- Observation: accuracy of weak classifiers depends on query.
 - F'_1 is perfect for (q, a, b) where q = reference object 1.
 - F'_1 is good for queries close to reference object 1.
- Question: how can we capture that?

Query-Sensitive Weak Classifiers



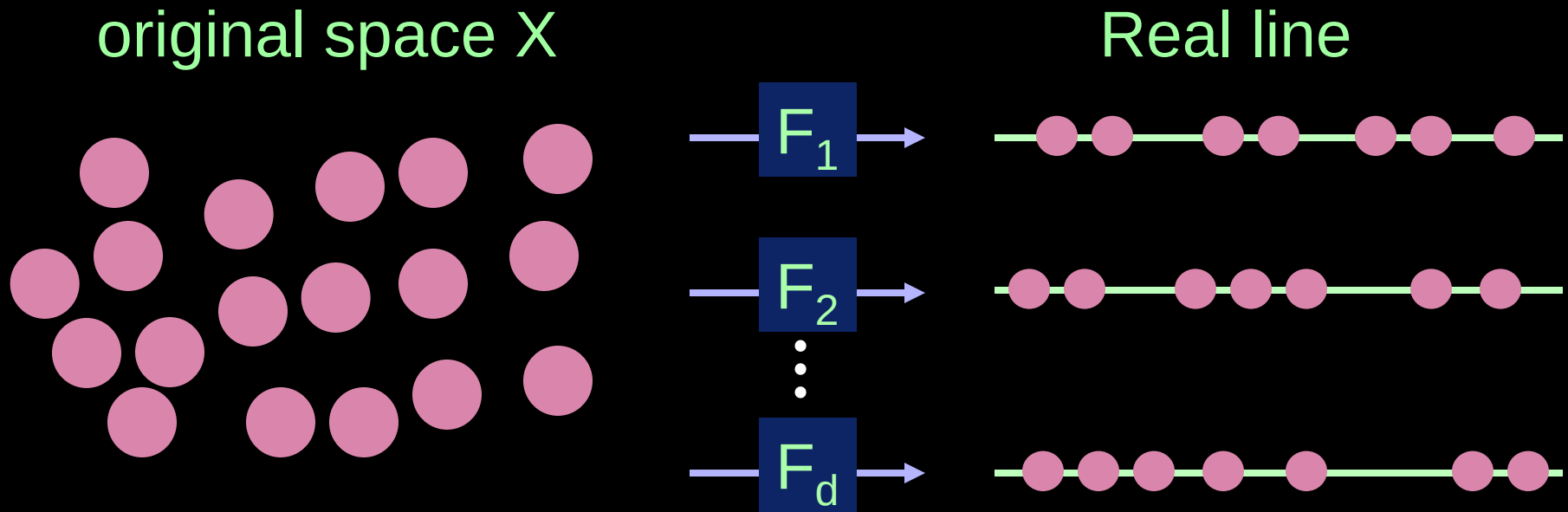
- V : area of influence (interval of real numbers).
- $Q_{F,V}(q, a, b) = \begin{cases} F'(q, a, b) & \text{if } F(q) \text{ is in } V \\ \text{"I don't know"} & \text{if } F(q) \text{ not in } V \end{cases}$

Query-Sensitive Weak Classifiers



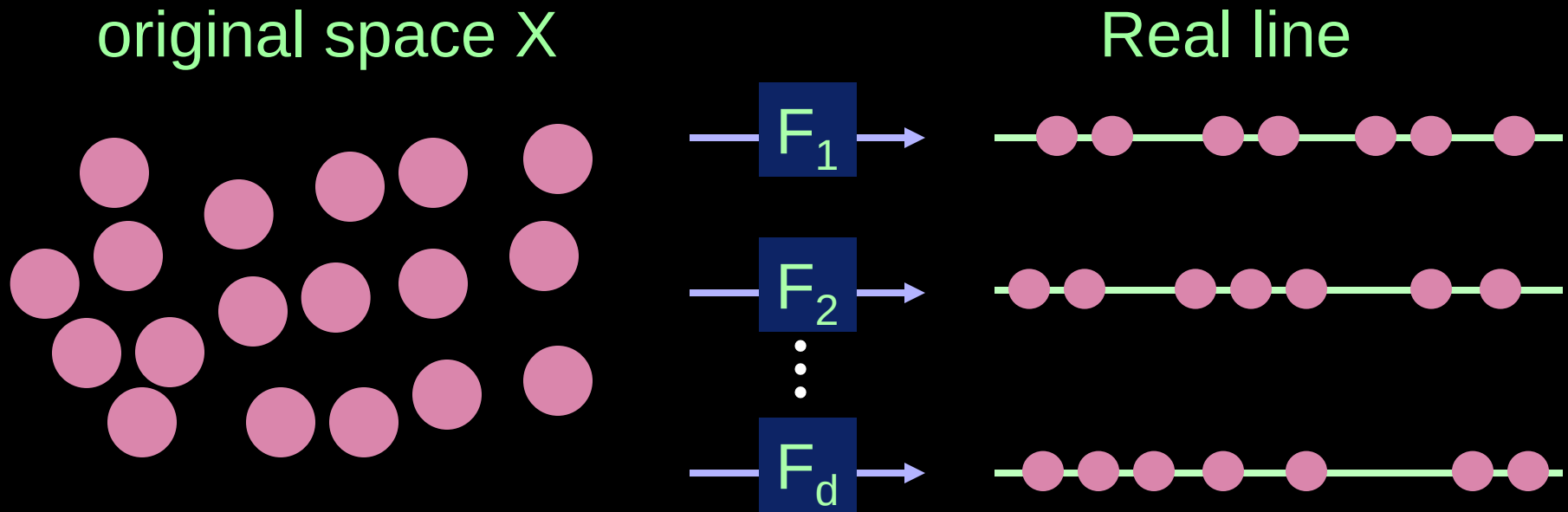
- V : area of influence (interval of real numbers).
- $Q_{F,V}(q, a, b) = \begin{cases} F'(q, a, b) & \text{if } F(q) \text{ is in } V \\ \text{"I don't know"} & \text{if } F(q) \text{ not in } V \end{cases}$
- If V includes all real numbers, $Q_{F,V} = F'$.

Applying AdaBoost



- AdaBoost forms classifiers Q_{F_i, V_i} .
 - F_i : 1D embedding.
 - V_i : area of influence for F_i .
- Output: $H = w_1 Q_{F_1, V_1} + w_2 Q_{F_2, V_2} + \dots + w_d Q_{F_d, V_d}$.

Applying AdaBoost



■ Empirical observation:

- At late stages of the training, query-sensitive weak classifiers are still useful, whereas query-insensitive classifiers are not.

From Classifier to Embedding

AdaBoost
output

$$H(q, a, b) = \sum_{i=1}^d w_i Q_{F_i, v_i}(q, a, b)$$

What embedding should we use?

What distance measure should we use?

From Classifier to Embedding

AdaBoost
output

$$H(q, a, b) = \sum_{i=1}^d w_i Q_{F_i, V_i}(q, a, b)$$

BoostMap
embedding

$$F(x) = (F_1(x), \dots, F_d(x))$$

Distance
measure

$$D(F(q), F(x)) = \sum_{i=1}^d w_i S_{F_i, V_i}(q) |F_i(q) - F_i(x)|$$

From Classifier to Embedding

AdaBoost
output

$$H(q, a, b) = \sum_{i=1}^d w_i Q_{F_i, V_i}(q, a, b)$$

BoostMap
embedding

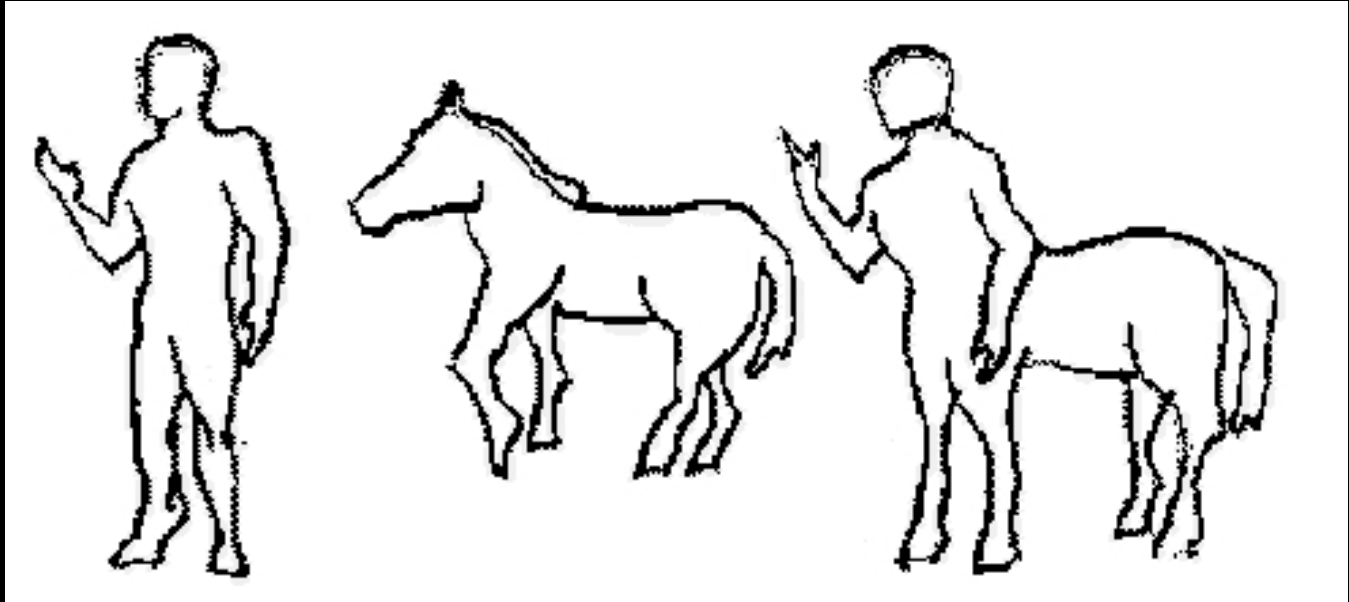
$$F(x) = (F_1(x), \dots, F_d(x))$$

Distance
measure

$$D(F(q), F(x)) = \sum_{i=1}^d w_i S_{F_i, V_i}(q) |F_i(q) - F_i(x)|$$

- Distance measure is *query-sensitive*.
 - Weighted L_1 distance, weights depend on q .
 - $S_{F, V}(q) = 1$ if $F(q)$ is in V , 0 otherwise.

Centaur Revisited



- Reference objects: human, horse, centaur.
 - For centaur queries, use weights $(0,0,1)$.
 - For human queries, use weights $(1,0,0)$.
- Query-sensitive distances are non-metric.
 - Combine efficiency of L_1 distance and ability to capture non-metric structure.

Additional Modeling Power

- For any *finite* space X obeying reflexivity, there is an isometric query-sensitive embedding to the $|X|$ -dimensional vector space.
 - Reflexivity: $D(x,x) = 0$.
- For query-insensitive embeddings:
 - Any finite *metric* space X can be isometrically embedded to an L -infinity space.
 - Any finite *metric* space X can be embedded to a Euclidean space with distortion $\leq \log|X|$.

Dynamic Time Warping on Time Series

Database: 31818 time series.

Query set: 1000 time series.

	Query-Sensitive	Query-Insensitive	RRO
Accuracy	95%	95%	95%
# of distances	1995	5691	5229
Sec. per query	33	95	87
Speed-up factor	16	5.6	6.1

Dynamic Time Warping on Time Series

Database: 31818 time series.

Query set: 1000 time series.

	Query-Sensitive	Query-Insensitive	RRO
Accuracy	100%	100%	100%
# of distances	6965	9562	10134
Sec. per query	116	159	169
Speed-up factor	4.6	3.3	3.1

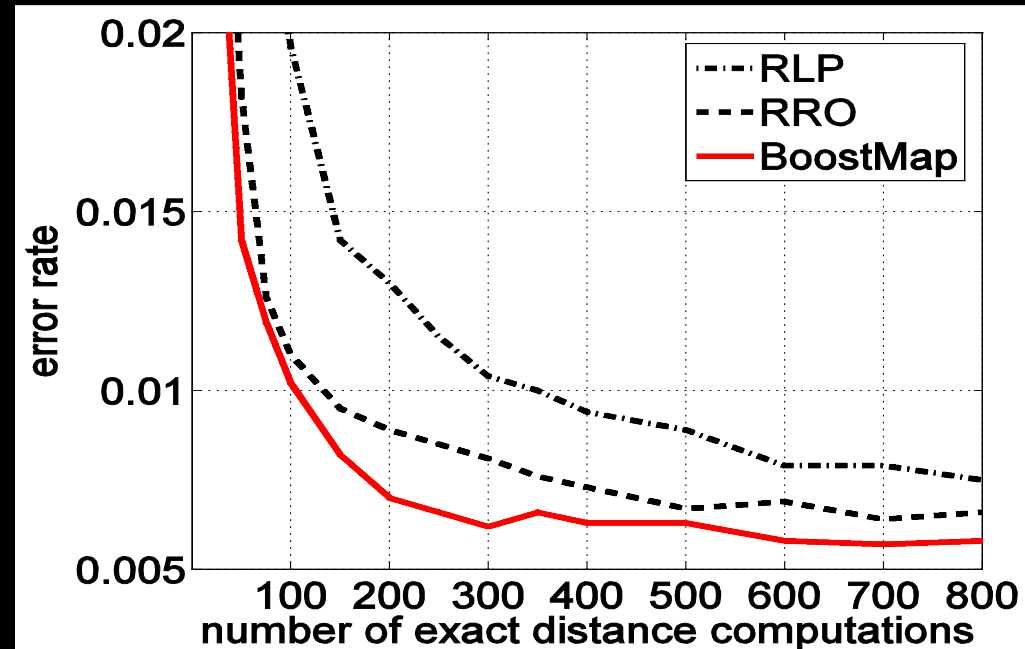
Dynamic Time Warping on Time Series

Database: 32768 time series.

Query set: 50 time series.

	Query-Sensitive	Vlachos 2003
Accuracy	100%	100%
# of distances	640	over 6500
Sec. per query	10.7	over 110
Speed-up factor	51.2	under 5

Speeding Up Classification

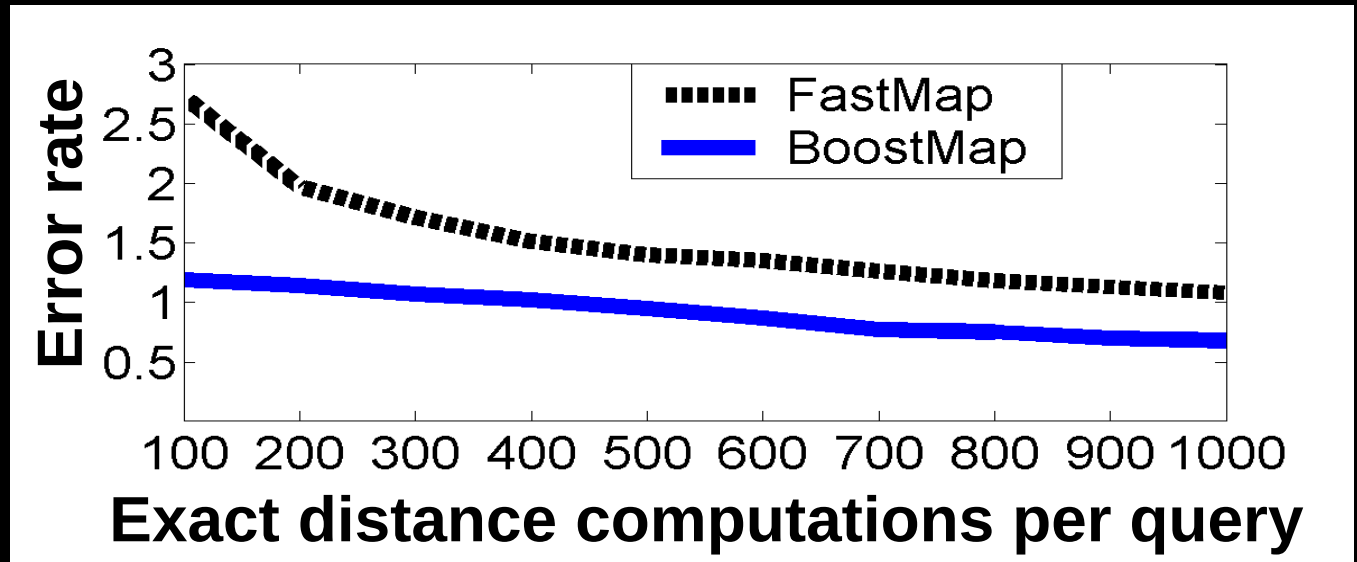


■ For each test object:

- Measure distance to 100 prototypes.
- Find 700 nearest neighbors using the embedding.
- Find 3 nearest neighbors among the 700 candidates.

Is this work always necessary?

Speeding Up Classification



- Suppose that, for some test object:
 - We measure distance to 10 prototypes.
 - Find 50 nearest neighbors using the embedding.
 - All 50 objects are twos.

It is a two!

Using a Cascade

- 10 dimensions, 50 nearest neighbors.
- 20 dimensions, 26 nearest neighbors.
- 30 dimensions, 43 nearest neighbors.
- 40 dimensions, 32 nearest neighbors.
- ...
- Filter-and-refine, 1000 distances.
 - Easy objects take less work to recognize.
 - Thresholds can be learned.

Cascade Results on MNIST

	Brute force	BoostMap	Cascade
Distances per query	20000	1000	93
Average time	22 min	67 sec	6.2 sec
Error rate	0.63%	0.68%	0.74%

Cascade Results on MNIST

	Brute force	BoostMap	Cascade	Cascade (60000)
Distances per query	20000	1000	93	77
Average time	22 min	67 sec	6.2 sec	5.2 sec
Error rate	0.63%	0.68%	0.74%	0.61%

Cascade Results on UNIPEN

	Brute force	Bahlmann 2004	BM	BM	Cascade
Distances per query	10630	150	300	60	30
Seconds/ query	12	0.16	0.33	0.07	0.03
Error rate	1.90%	2.90%	1.90%	2.14%	2.10%

BoostMap Recap - Theory

- Machine-learning method for optimizing embeddings.
 - Explicitly maximizes amount of nearest neighbor structure preserved by embedding.
 - Optimization method is independent of underlying geometry.
 - Query-sensitive version can capture non-metric structure.
 - Additional savings can be gained using cascades.
 - Sacrificing retrieval accuracy, while preserving classification accuracy.

BoostMap Recap - Practice

- BoostMap can significantly speed up nearest neighbor retrieval and classification.
 - Useful in real computer vision datasets:
 - Hand shape classification.
 - Optical character recognition.
 - In our four datasets, does better than other methods.
 - In three datasets, better than custom-made methods.
 - Domain-independent formulation.
 - Distance measures are used as a black box.
 - May be of interest to other domains/communities.

Future Work

- Applying BoostMap to other domains.
 - Networks.
 - Biological sequences.
- Using domain-specific information.
 - The work presented here only uses distances.
 - In ICDAR 2005, we improve performance by using *alignment* information.
 - Domain-specific features can be used as weak classifiers to improve embedding accuracy.

Future Work

- Indexing as a machine learning problem.
 - Potentially applicable to other indexing paradigms:
 - Tree-based structures.
 - Hash tables.
 - Replacing geometric formulations with learning-based formulations may help in non-metric spaces.
- Sublinear retrieval in non-metric spaces.
 - LSH only applicable to vector spaces.

Future Work

- Minimizing need for supervision.
 - For datasets like UNIPEN, MNIST, we need manual labeling of each object.
 - Impractical for huge databases.
 - Can we create very large databases for classification with minimal manual labeling?

Future Work

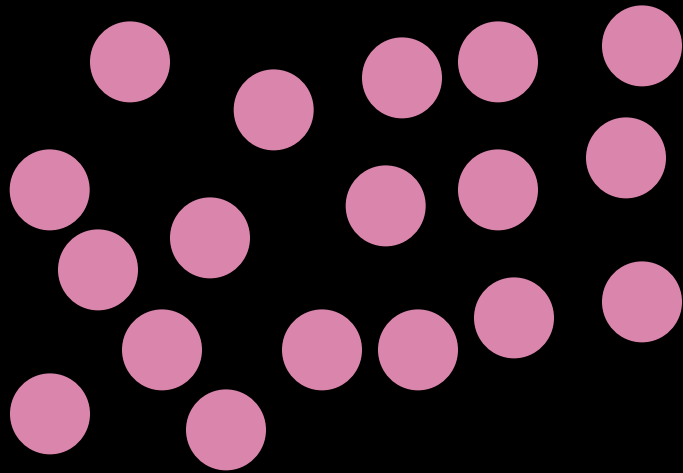
- Unsegmented nearest neighbor problem:
 - Query may be unsegmented.



- Database objects may be unsegmented.
 - Example 1: continuous footage of sign language, where we want to spot a sign.
 - Example 2: Query-by-humming. Database of songs, query is a small part of a song.

END

original space X



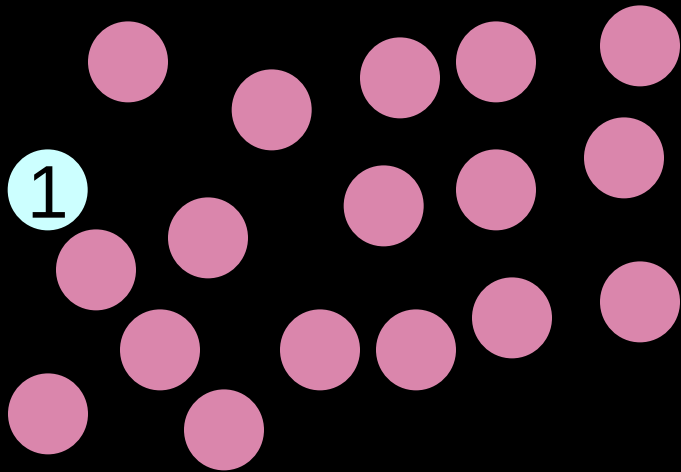
Training triples

Weights

(q_1, a_1, b_1)		$1/m$
(q_2, a_2, b_2)		$1/m$
(q_3, a_3, b_3)		$1/m$
$\vdots$		$\vdots$
(q_m, a_m, b_m)		$1/m$

- Training round 0.
- Classifier: $H = \text{"I don't know"}$.
- Embedding: $F = 0$
- Distance: $D(F(x), F(y)) = 0$.
- Weights: all equal to $1/m$ (example: $m = 100,000$).

original space X



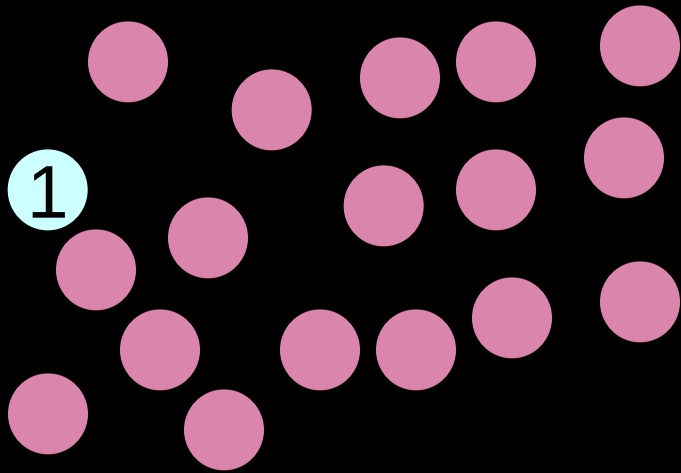
Training triples

Weights

(q_1, a_1, b_1)		$1/m$
(q_2, a_2, b_2)		$1/m$
(q_3, a_3, b_3)		$1/m$
$\vdots$		$\vdots$
(q_m, a_m, b_m)		$1/m$

- Training round 1.
- Classifier: $H = a_1 F'_1$.
- Embedding: $F = (F_1)$.
- Distance: $D(F(x), F(y)) = a_1 |F_1(x) - F_1(y)|$.

original space X



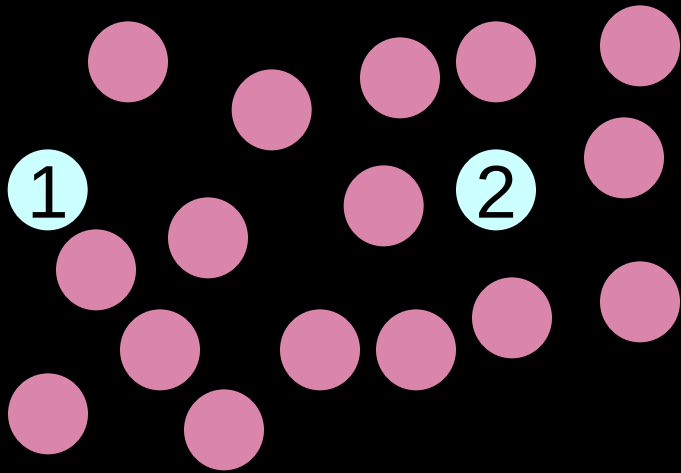
Training triples

Weights

(q_1, a_1, b_1)		W_{11}
(q_2, a_2, b_2)		W_{12}
(q_3, a_3, b_3)		W_{13}
$\vdots$		$\vdots$
(q_m, a_m, b_m)		W_{1m}

- Training round 1.
- Classifier: $H = a_1 F'_1$.
- Embedding: $F = (F_1)$.
- Distance: $D(F(x), F(y)) = a_1 |F_1(x) - F_1(y)|$.
- Weights: higher for incorrectly classified triples.

original space X



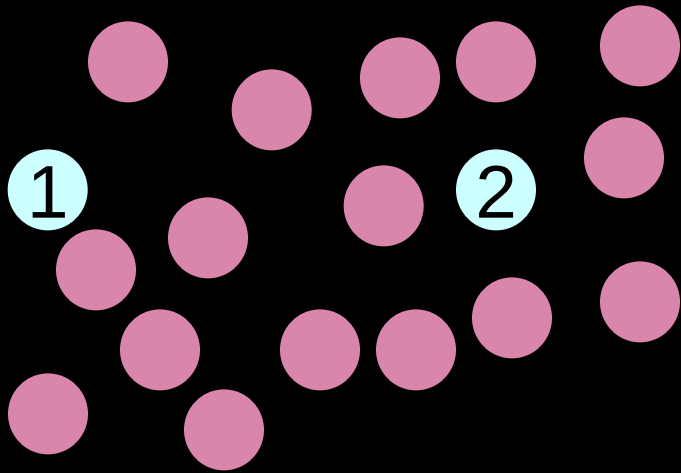
Training triples

Weights

$$\begin{array}{ll} (q_1, a_1, b_1) & \dots\dots\dots w_{11} \\ (q_2, a_2, b_2) & \dots\dots\dots w_{12} \\ (q_3, a_3, b_3) & \dots\dots\dots w_{13} \\ & \vdots \\ (q_m, a_m, b_m) & \dots\dots\dots w_{1m} \end{array}$$

- Training round 2.
- Classifier: $H = a_1 F'_1 + a_2 F'_2$.
- Embedding: $F = (F_1, F_2)$.
- Distance: $D(F(x), F(y)) = a_1 |F_1(x) - F_1(y)| + a_2 |F_2(x) - F_2(y)|$.

original space X



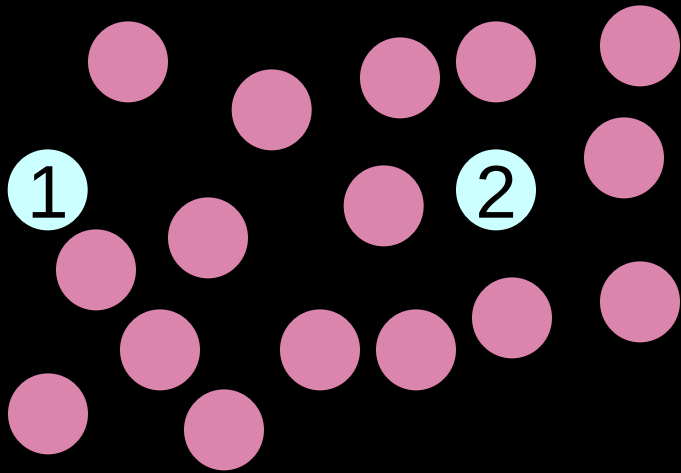
Training triples

Weights

(q_1, a_1, b_1)		W_{21}
(q_2, a_2, b_2)		W_{22}
(q_3, a_3, b_3)		W_{23}
$\vdots$		$\vdots$
(q_m, a_m, b_m)		W_{2m}

- Training round 2.
- Classifier: $H = a_1 F'_1 + a_2 F'_2$.
- Embedding: $F = (F_1, F_2)$.
- Distance: $D(F(x), F(y)) = a_1 |F_1(x) - F_1(y)| + a_2 |F_2(x) - F_2(y)|$.
- Weights: higher for incorrectly classified triples.

original space X



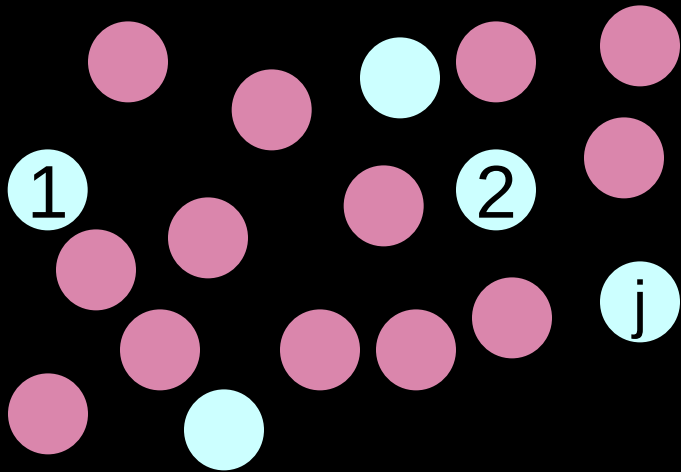
Training triples

Weights

$$\begin{aligned} (q_1, a_1, b_1) &\dots\dots\dots w_{j1} \\ (q_2, a_2, b_2) &\dots\dots\dots w_{j2} \\ (q_3, a_3, b_3) &\dots\dots\dots w_{j3} \\ &\vdots \\ (q_m, a_m, b_m) &\dots\dots\dots w_{jm} \end{aligned}$$

- Training round j.
- Classifier: $H = a_1 F'_1 + a_2 F'_2 + \dots + a_j F'_j$.
- Embedding: $F = (F_1, F_2, \dots, F_n)$.
- Distance: $D(F(x), F(y)) =$
 $a_1 |F_1(x) - F_1(y)| + a_2 |F_2(x) - F_2(y)| + \dots + a_j |F_j(x) - F_j(y)|.$

original space X



Training triples

Weights

$$\begin{aligned} (q_1, a_1, b_1) &\dots\dots\dots W_{j1} \\ (q_2, a_2, b_2) &\dots\dots\dots W_{j2} \\ (q_3, a_3, b_3) &\dots\dots\dots W_{j3} \\ &\vdots \\ (q_m, a_m, b_m) &\dots\dots\dots W_{jm} \end{aligned}$$

- Training round j.
- Classifier: $H = a_1 F'_1 + a_2 F'_2 + \dots + a_j F'_j$.
- Embedding: $F = (F_1, F_2, \dots, F_n)$.
- Distance: $D(F(x), F(y)) =$
 $a_1 |F_1(x) - F_1(y)| + a_2 |F_2(x) - F_2(y)| + \dots + a_j |F_j(x) - F_j(y)|$.
- Stop when accuracy stops improving ($a_j = 0$).