Feature Indexing

COMP 388-002/488-002 Biometrics

Daniel Moreira

Fall 2023

LOYOLA
UNIVERSITY CHICAGO

Today you will…

Get to know
Methods of feature indexing for biometric identification.

Today’s attendance

Please fill out the form

https://forms.gle/sjQKGKyMH7ogmPNa7

What is Biometrics?

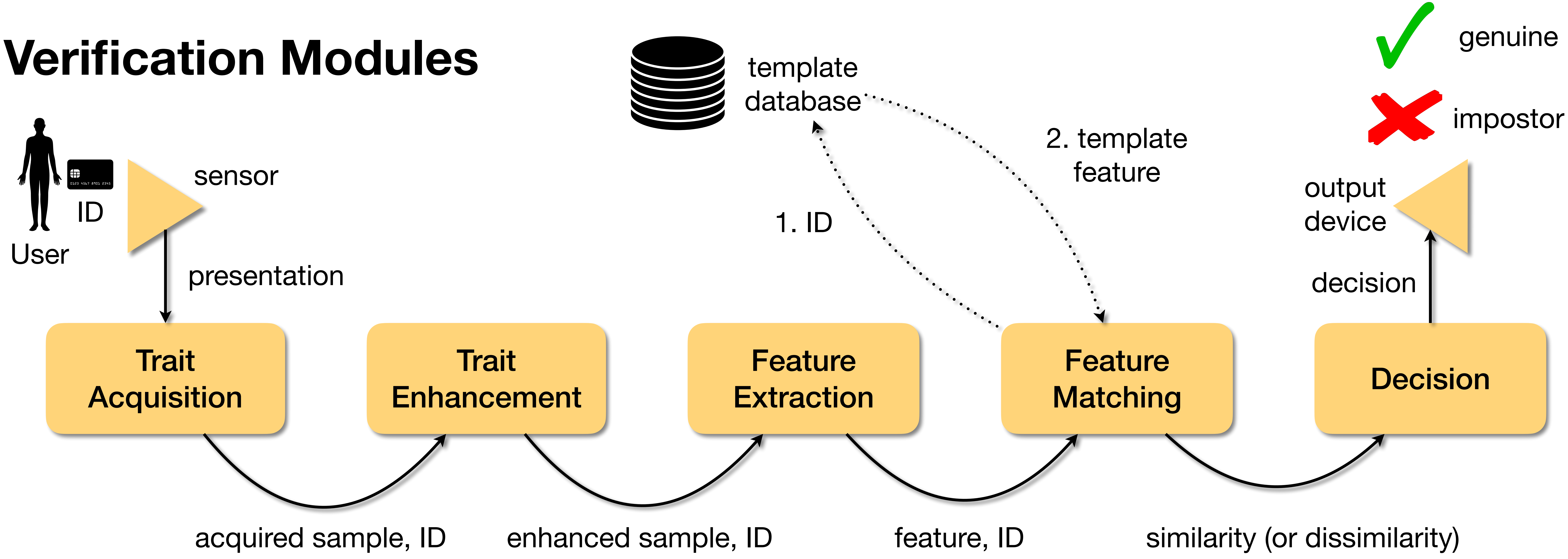
7 billion people
Who is this person? *(Identification)*
Is this person Jane Doe? *(Verification)*

Biometrics aims at ***identifying*** or ***verifying*** the claimed or denied identity of an individual based on their *physical*, *chemical* or *behavioral* traits.

LOYOLA UNIVERSITY CHICAGO

Biometric Systems

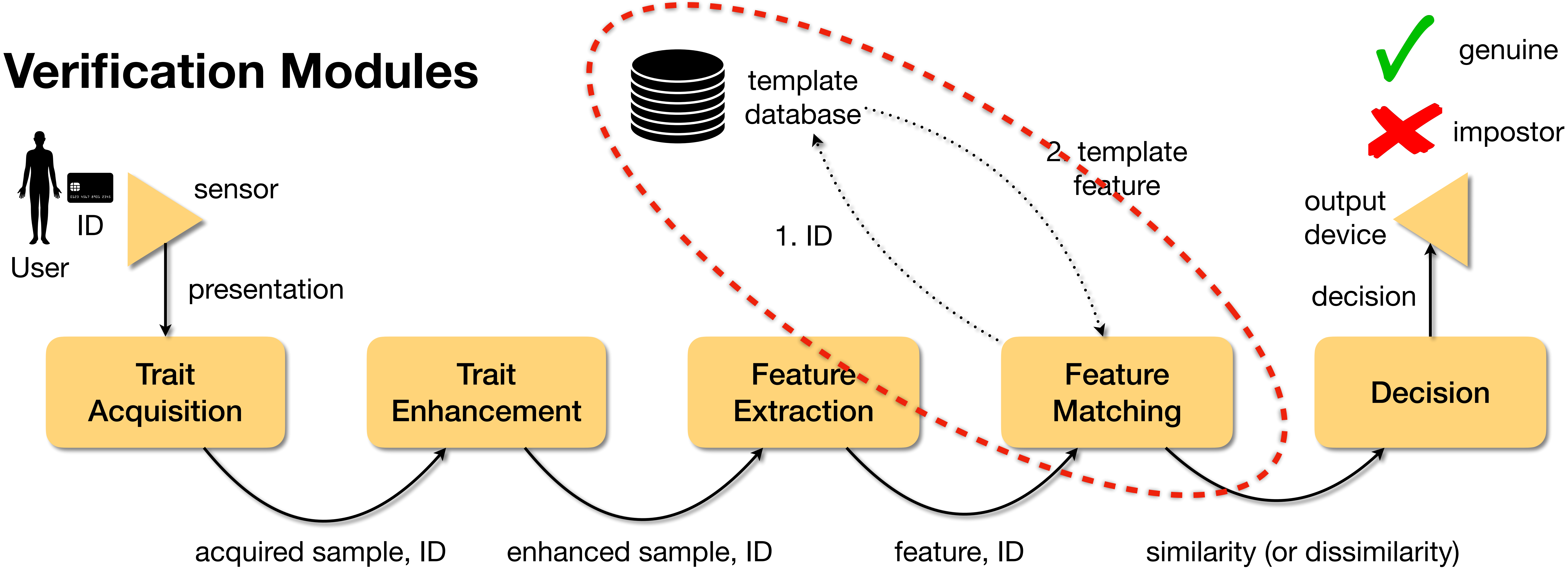
Verification Modules



LOYOLA UNIVERSITY CHICAGO

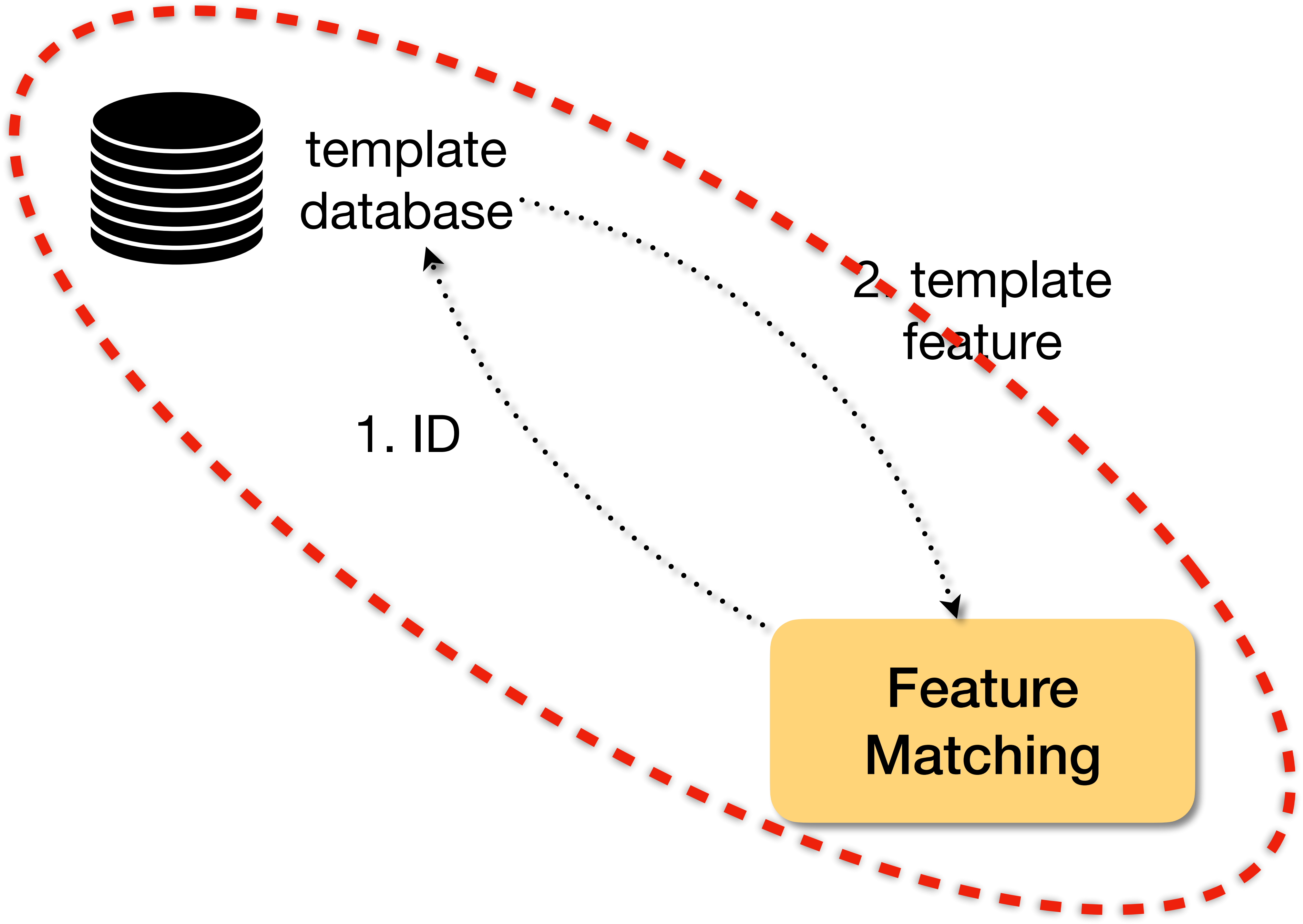
Biometric Systems

Verification Modules



Biometric Verification

No need for complex feature indexing.

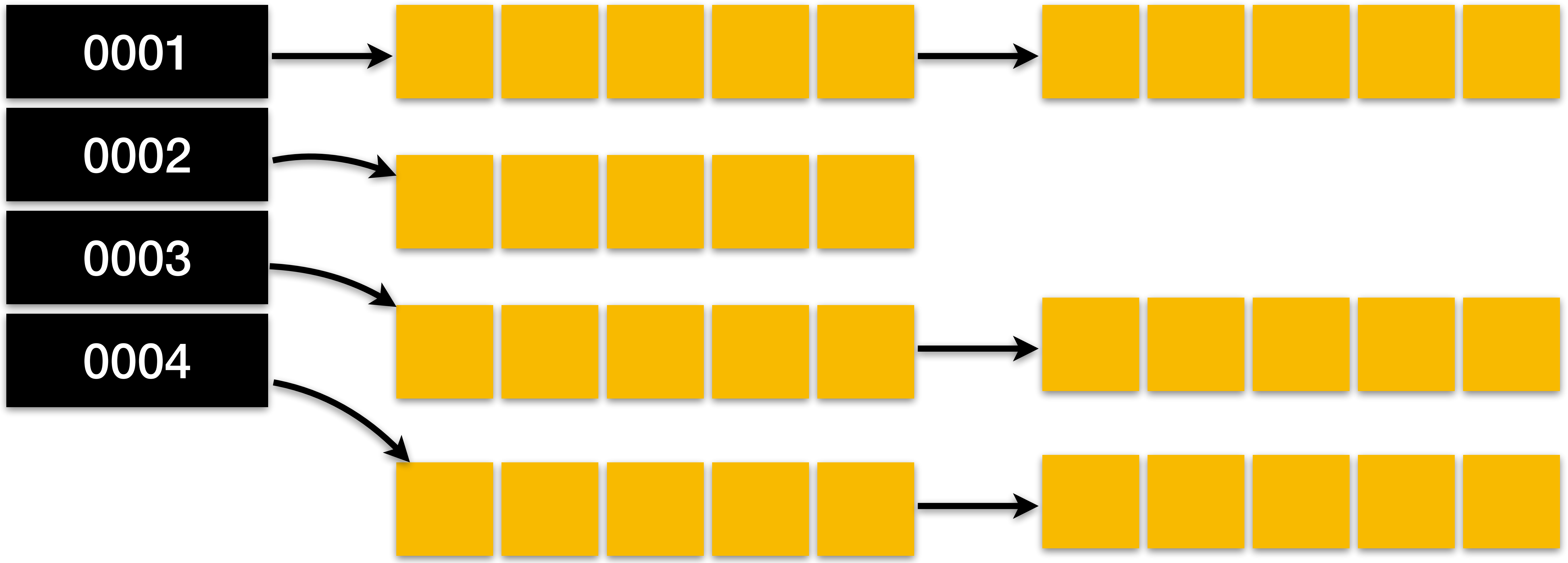


Biometric Verification

No need for complex feature indexing.

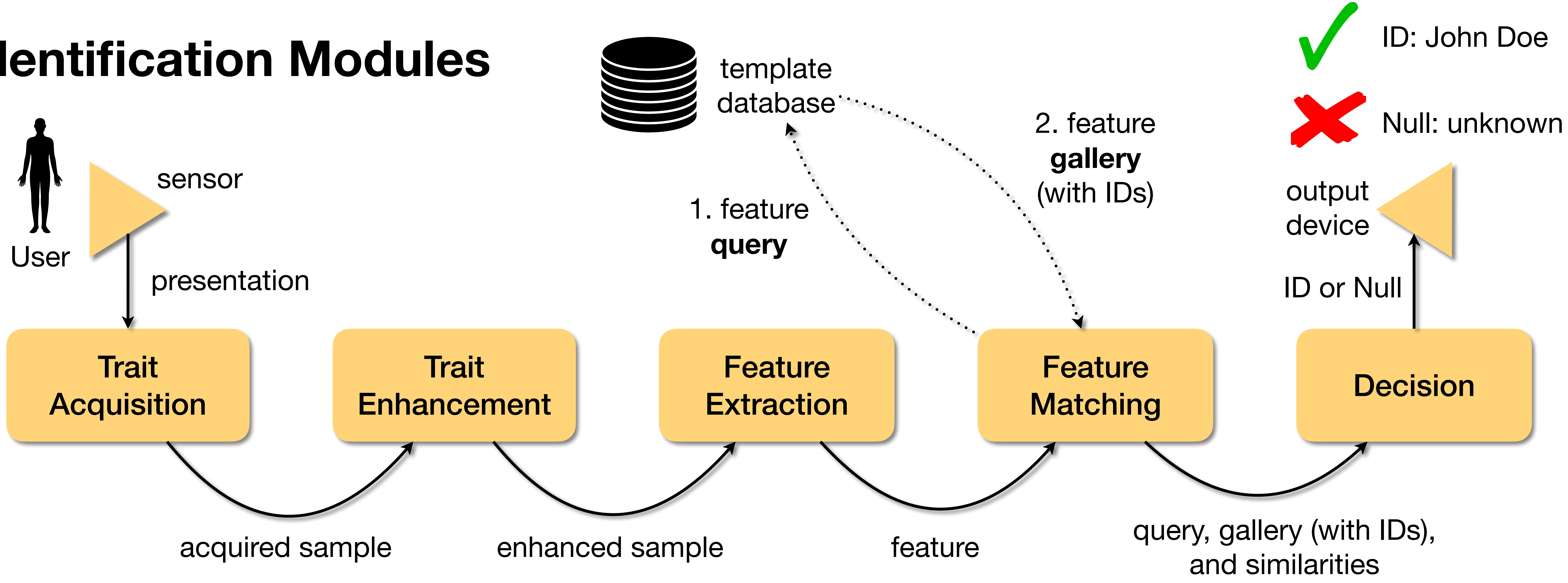
Use unique person's ID as index (or hash function input).

Retrieval of features in constant time.



Biometric Systems

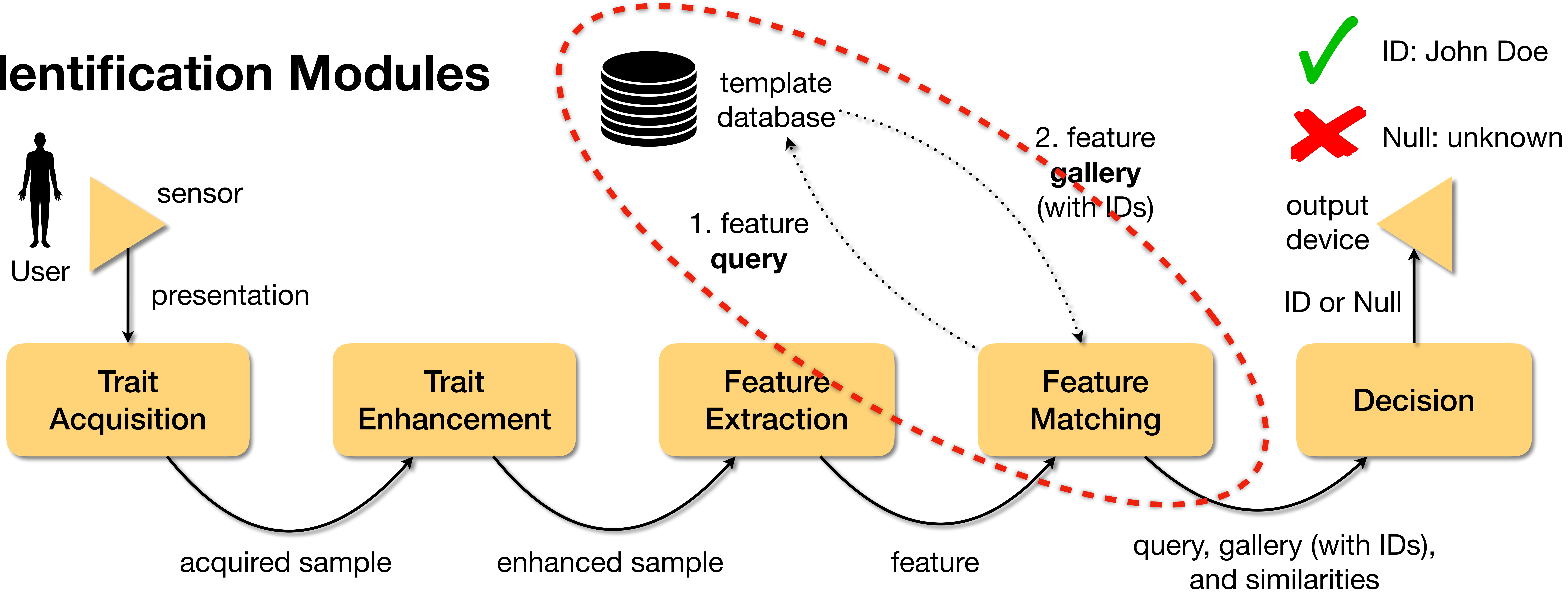
Identification Modules



LOYOLA UNIVERSITY CHICAGO

Biometric Systems

Identification Modules

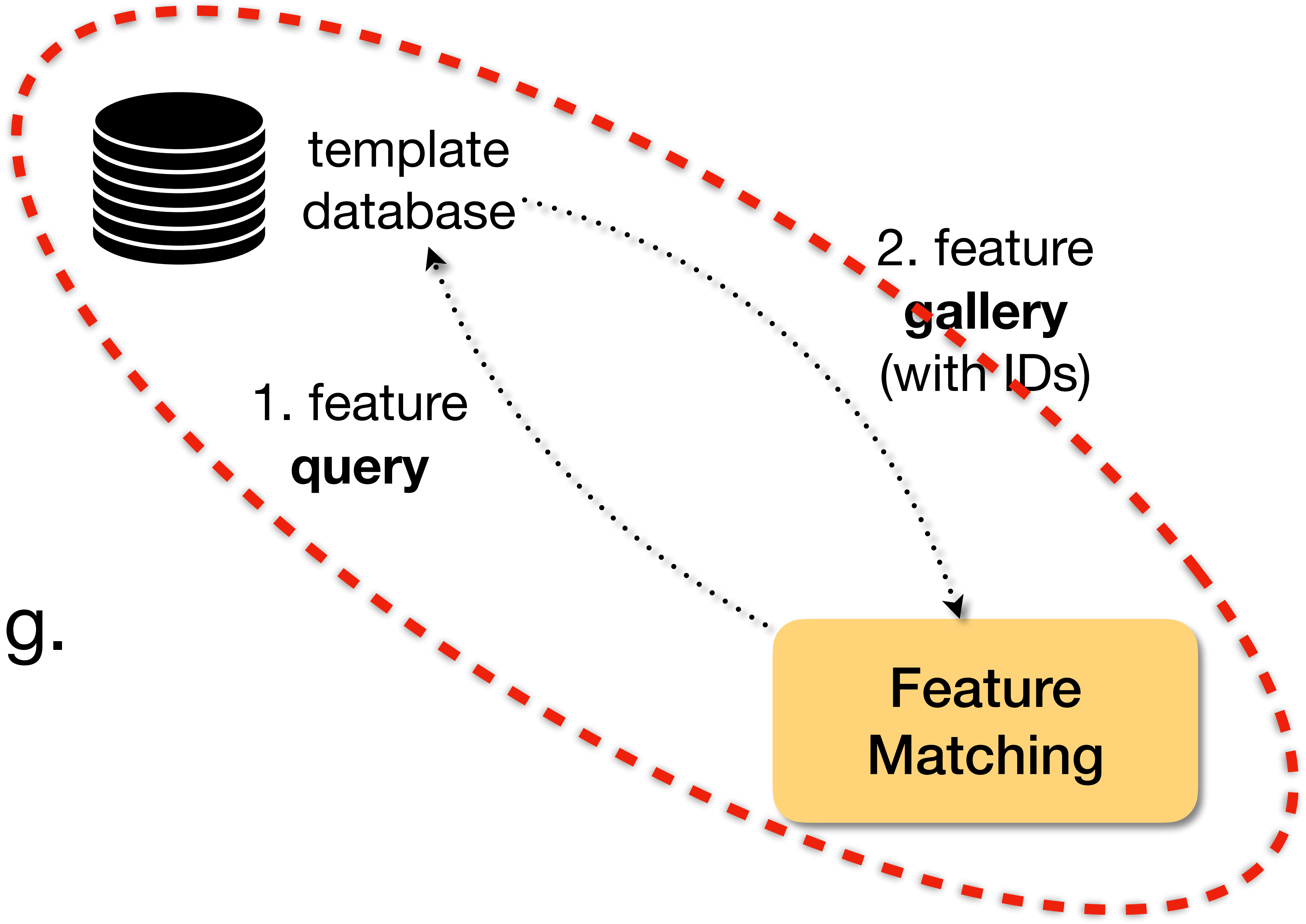


Biometric Identification

How to retrieve *k*-nearest features to compose gallery?

Need for more complex indexing.

Retrieval of features as quick as possible.



Biometric Identification

How to retrieve *k*-nearest features to compose gallery?

Need for more complex indexing.

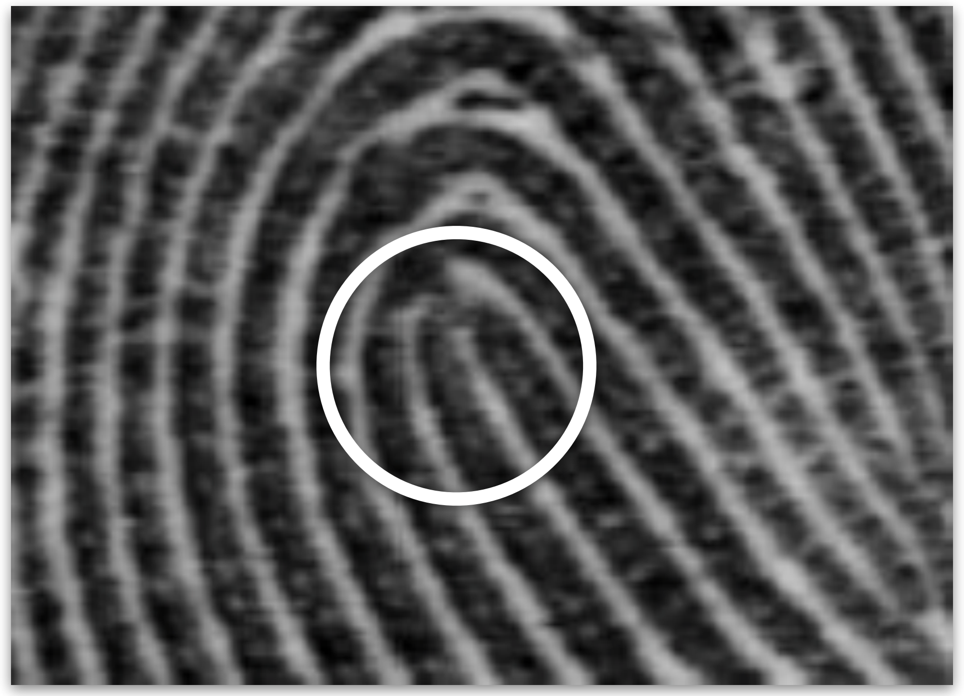
Retrieval of features as quick as possible.

LOYOLA UNIVERSITY CHICAGO

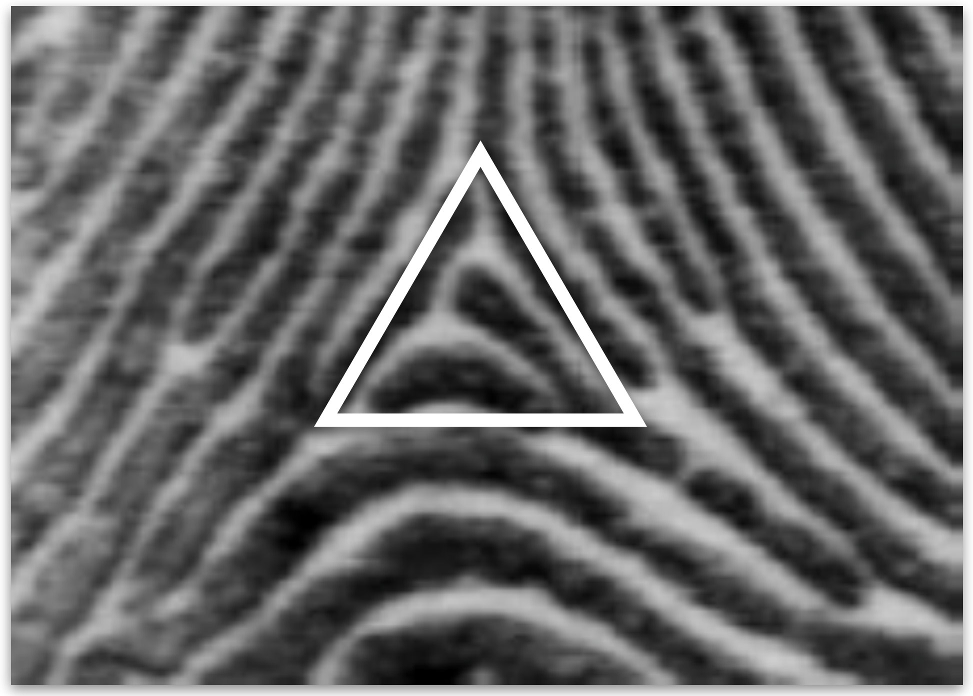
Fingerprint Indexing

Level-1 Features

Usage of Singular Points and Core



loop



delta

Jain, Ross, and Nadakumar
Introduction to Biometrics
Springer Books, 2011

LOYOLA UNIVERSITY CHICAGO

Fingerprint Indexing

Level-1 Features

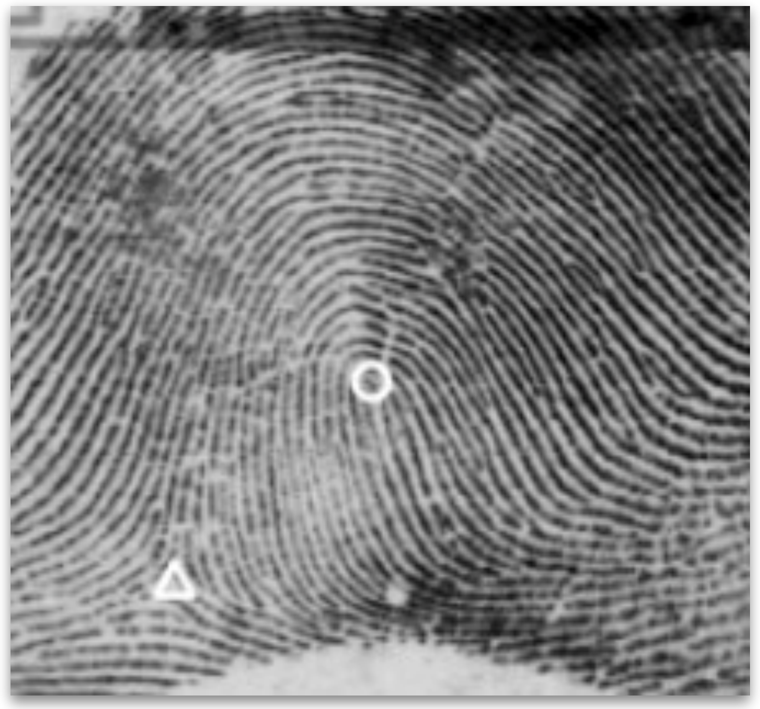
Usage of Singular Points and Core

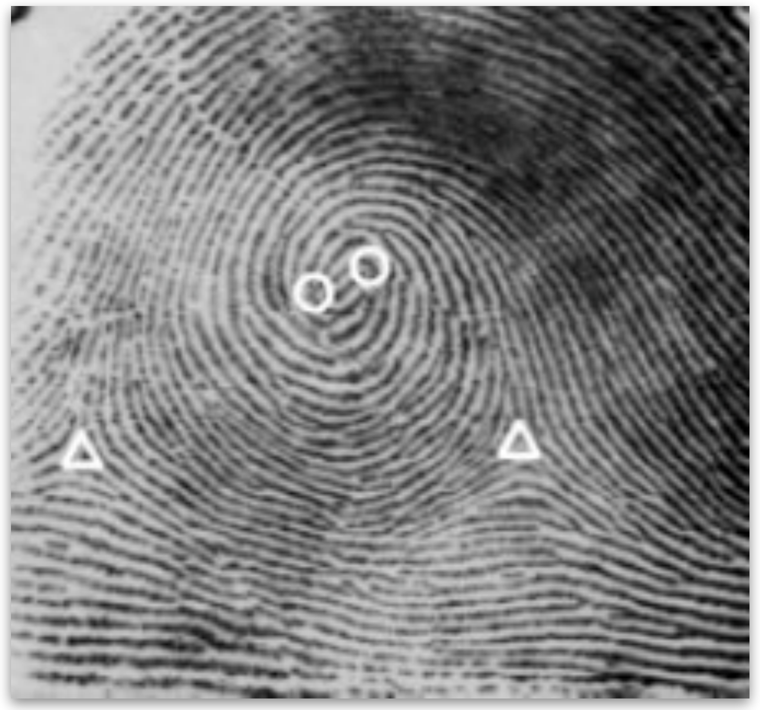
Jain, Ross, and Nadakumar
Introduction to Biometrics
Springer Books, 2011

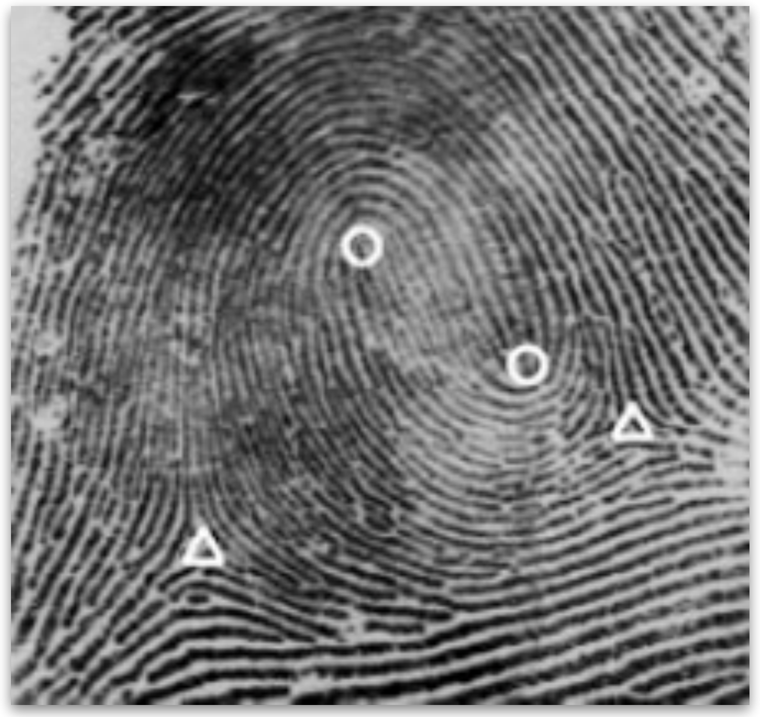

plain arch


tented arch


left loop


right loop


whorl


twin loop

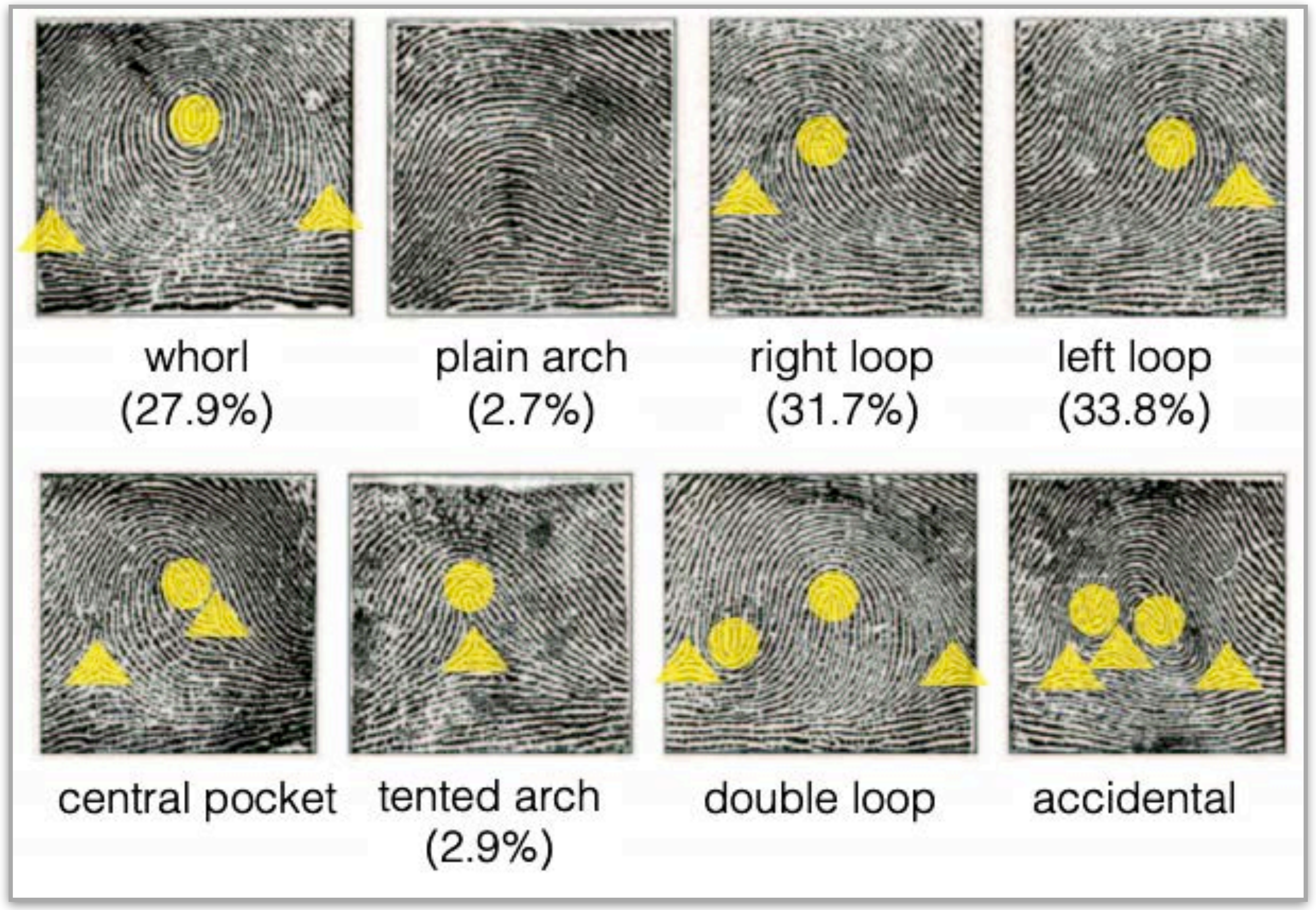
LOYOLA
UNIVERSITY CHICAGO

Fingerprint Indexing

Level-1 Features

FBI Automated Fingerprint Identification system (AFIS)
More than 200 million dactyloscopy cards.
Varied quality of samples.

Thanks to fingerprint classification through level-1 features, this time is reduced to **20 min**.



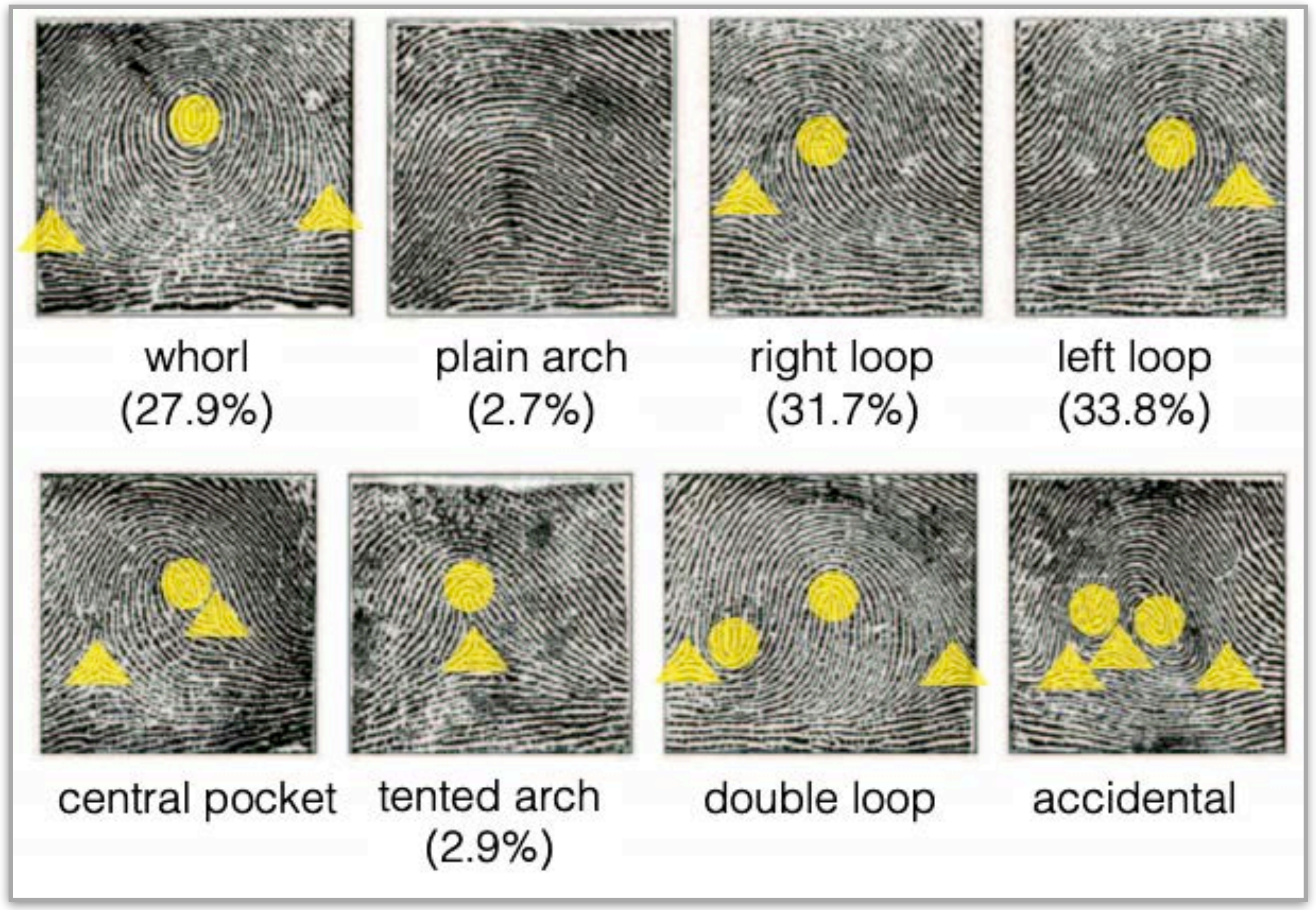
Henry's features, an alternative classification of level-1 features with 8 classes.

Fingerprint Indexing

Level-1 Features

FBI Automated Fingerprint Identification system (AFIS)
More than 200 million dactyloscopy cards.
Varied quality of samples.

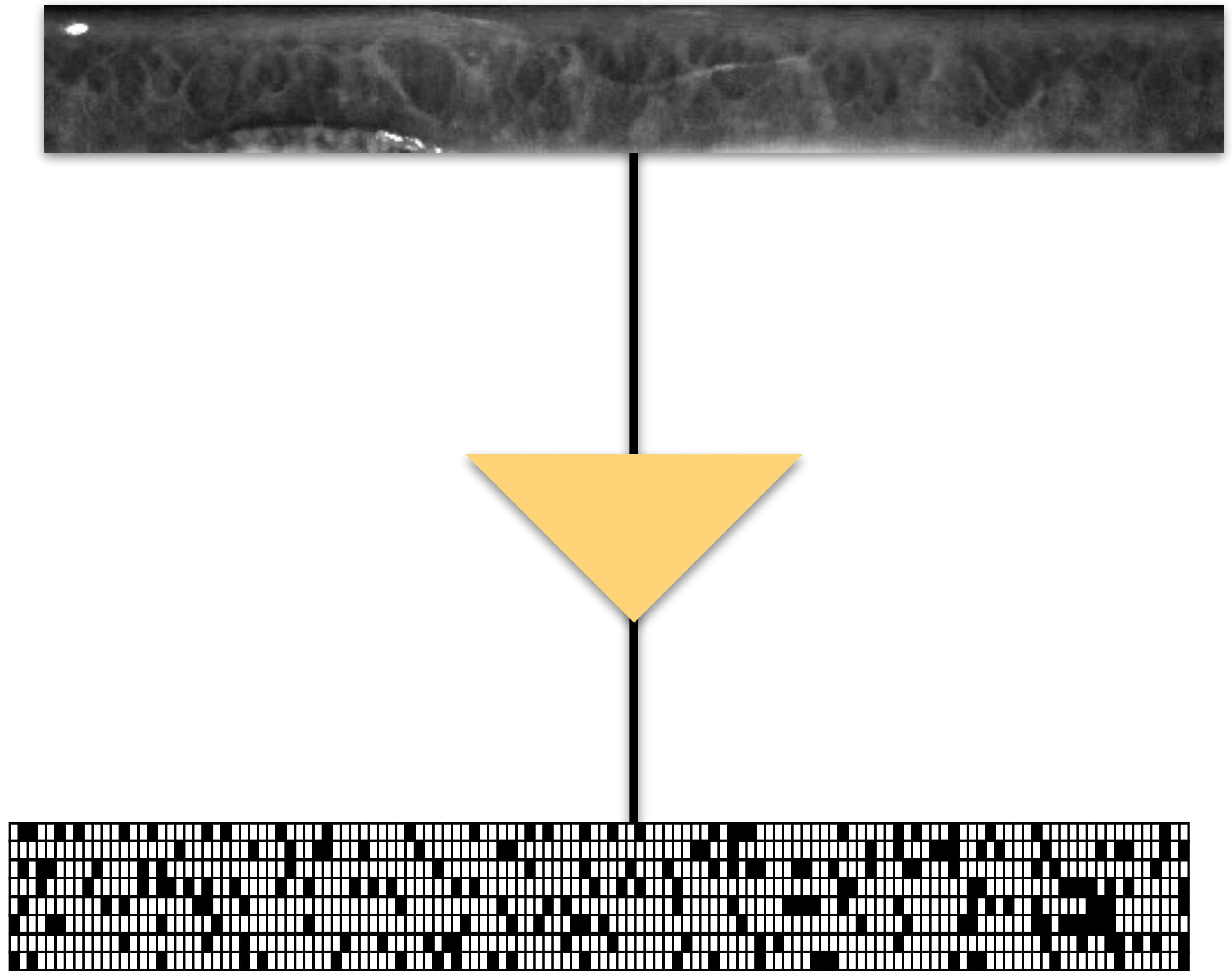
And a computer-based solution can do it in seconds, benefitting from the same features.



Henry's features, an alternative classification of level-1 features with 8 classes.

Feature Indexing

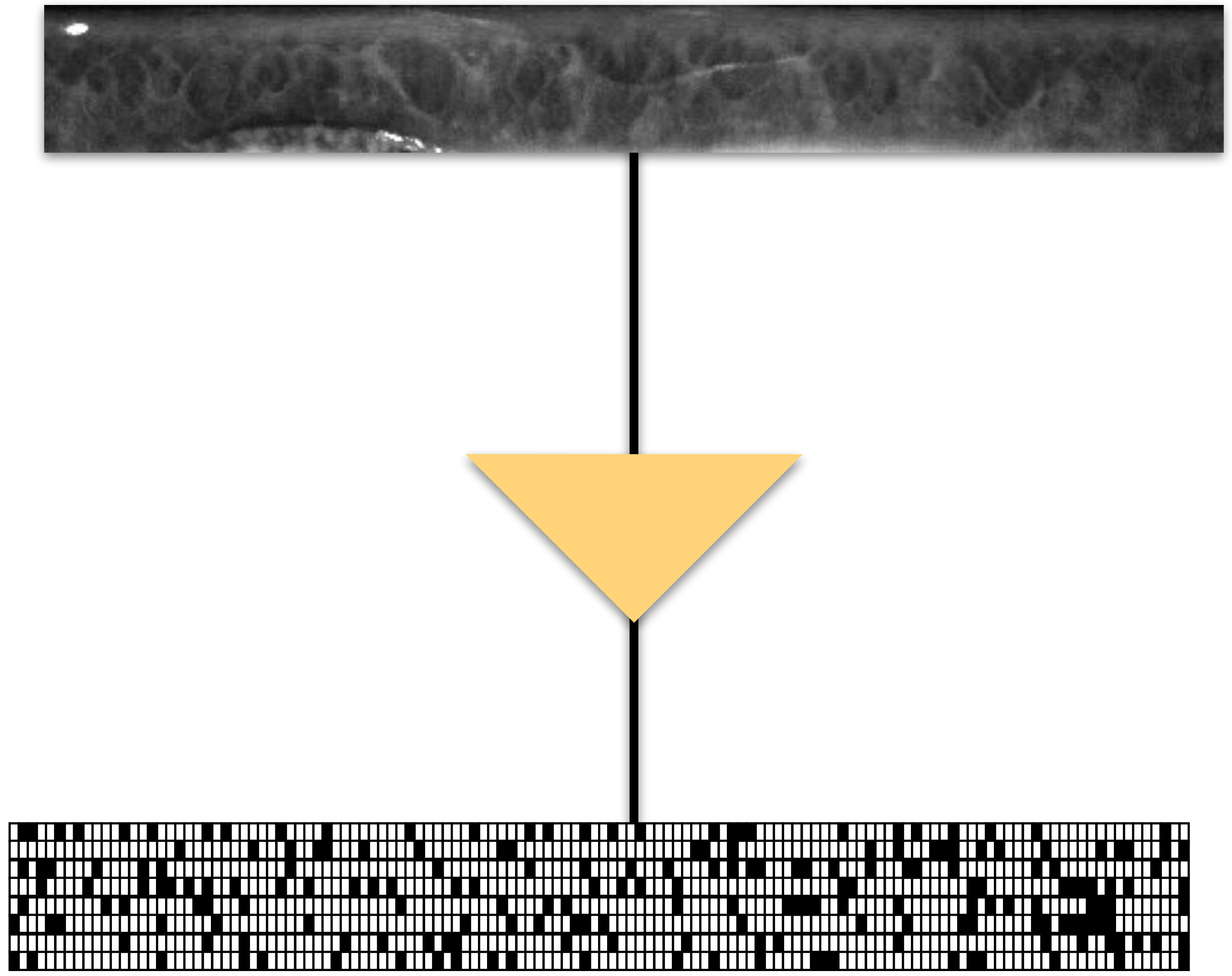
Iris Identification



2048 bits IrisCode

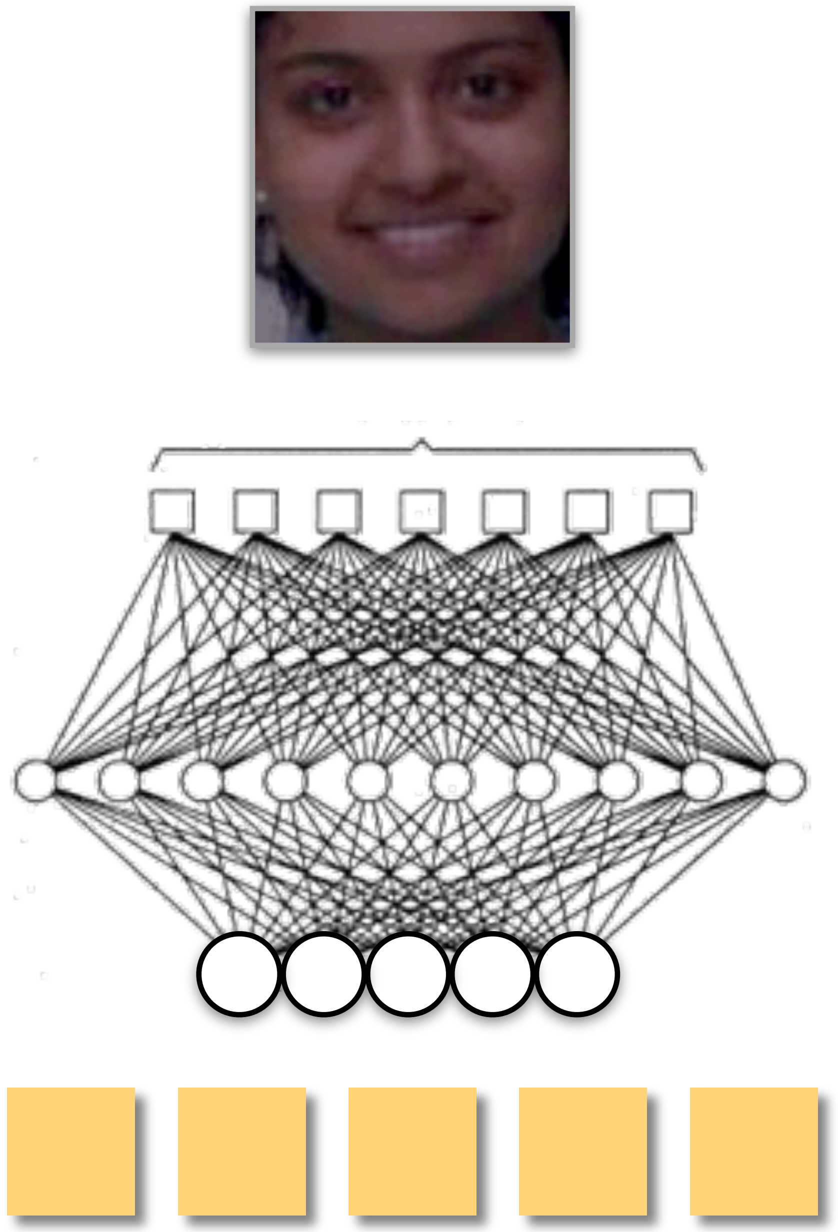
Feature Indexing

Iris Identification



2048 bits IrisCode

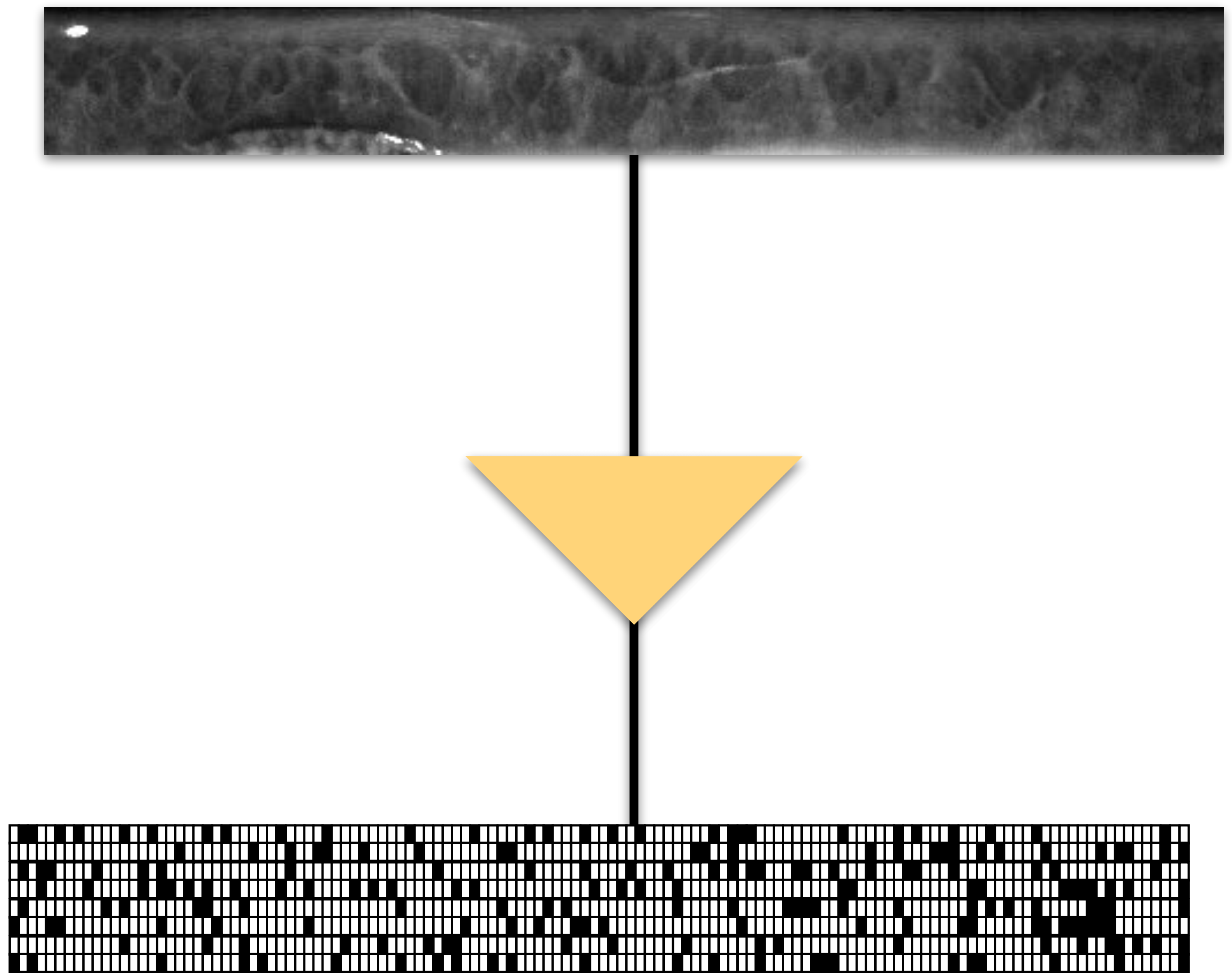
Face Identification



512D ArcFace embedding

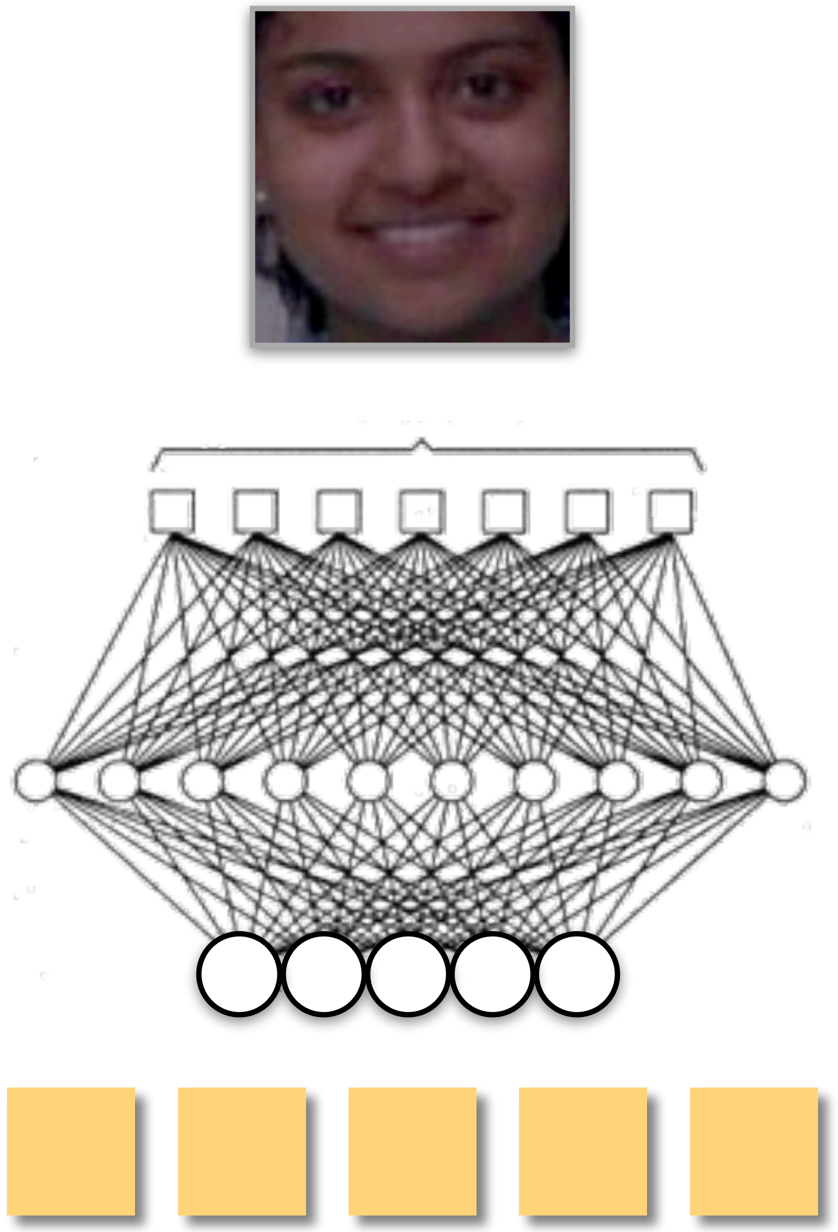
Feature Indexing

Iris Identification



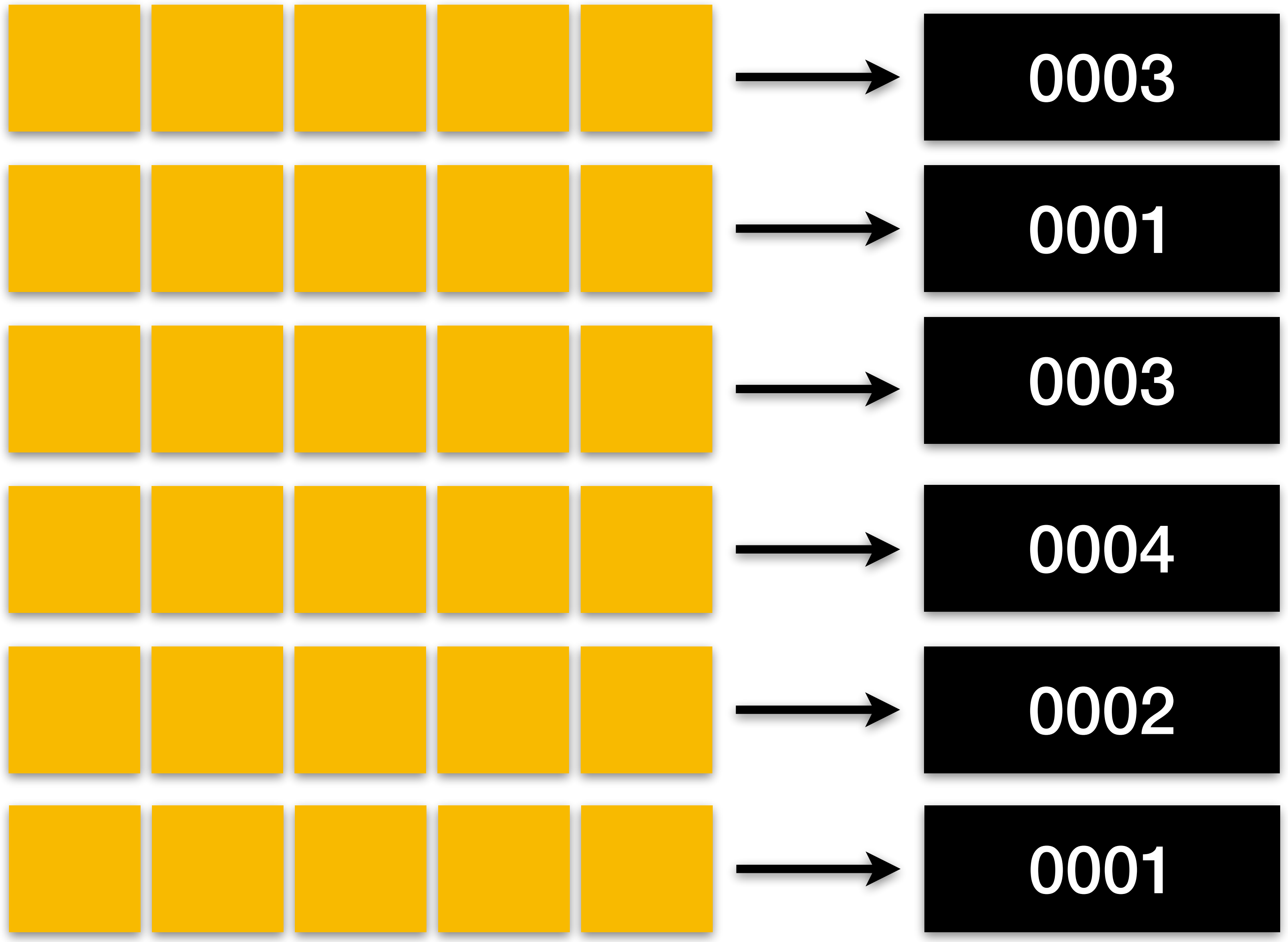
2048 bits IrisCode

Face Identification



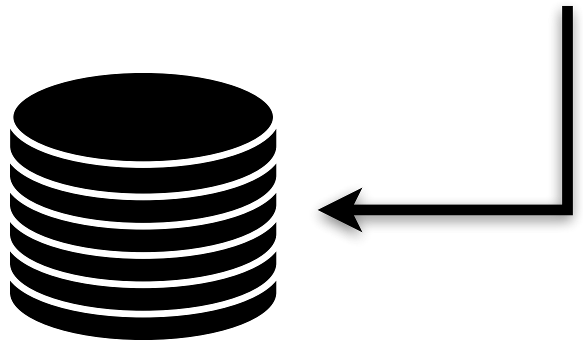
512D ArcFace embedding

Inverted Index



Feature space

Person’s IDs



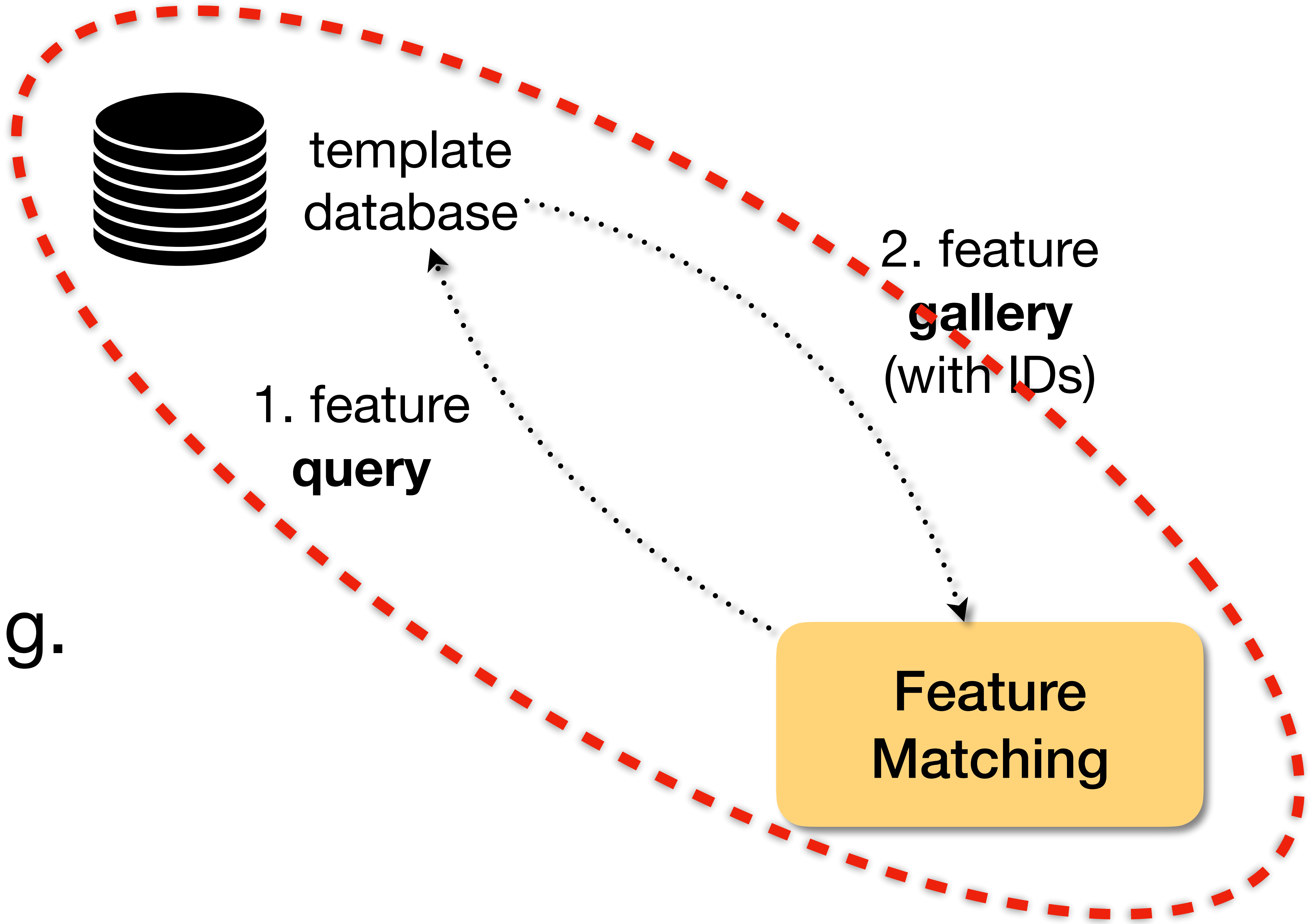
LOYOLA UNIVERSITY CHICAGO

Feature Indexing

How to retrieve *k*-nearest features to compose gallery?

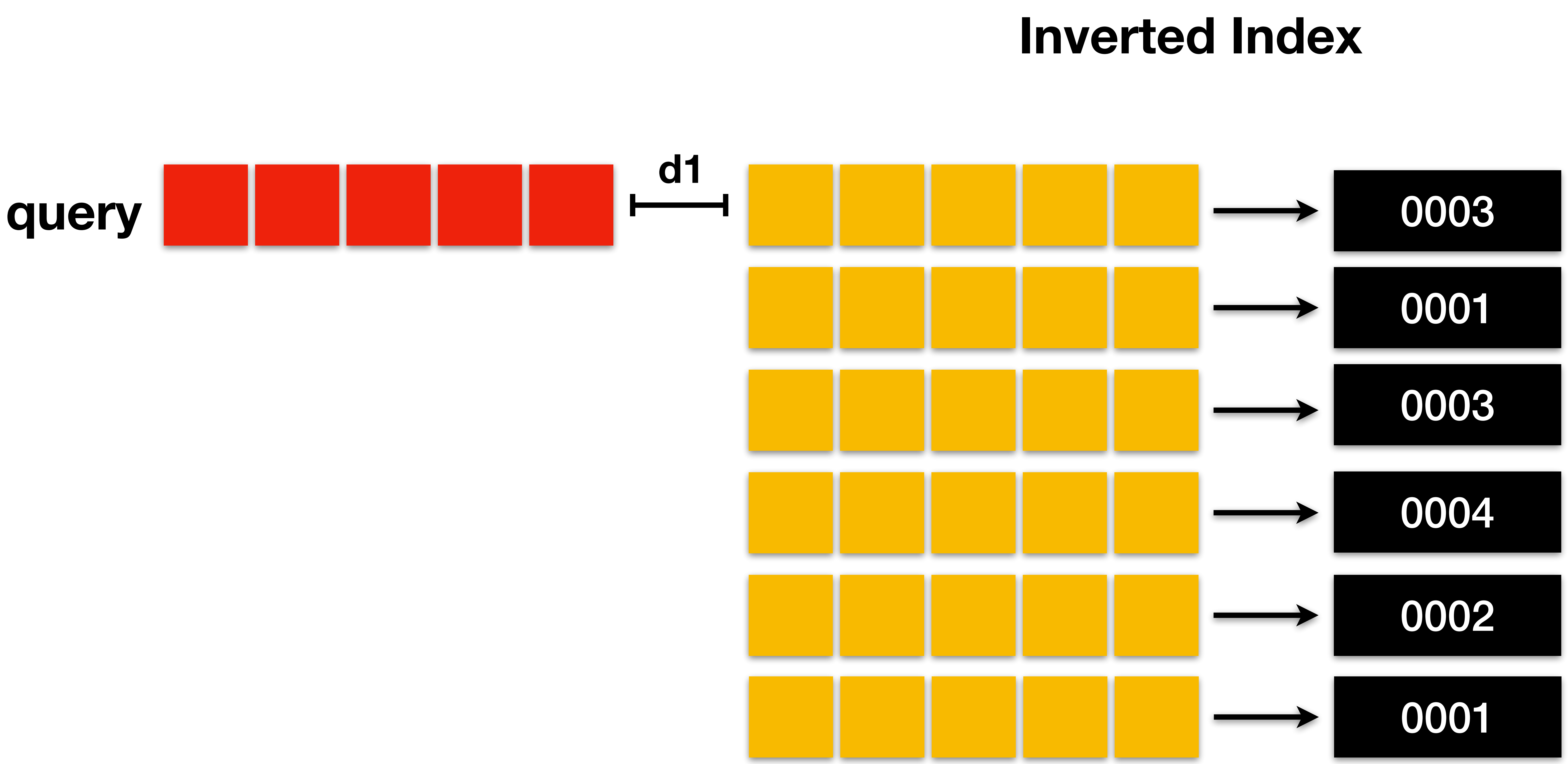
Need for more complex indexing.

Retrieval of features as quick as possible.



LOYOLA UNIVERSITY CHICAGO

Brute Force Search

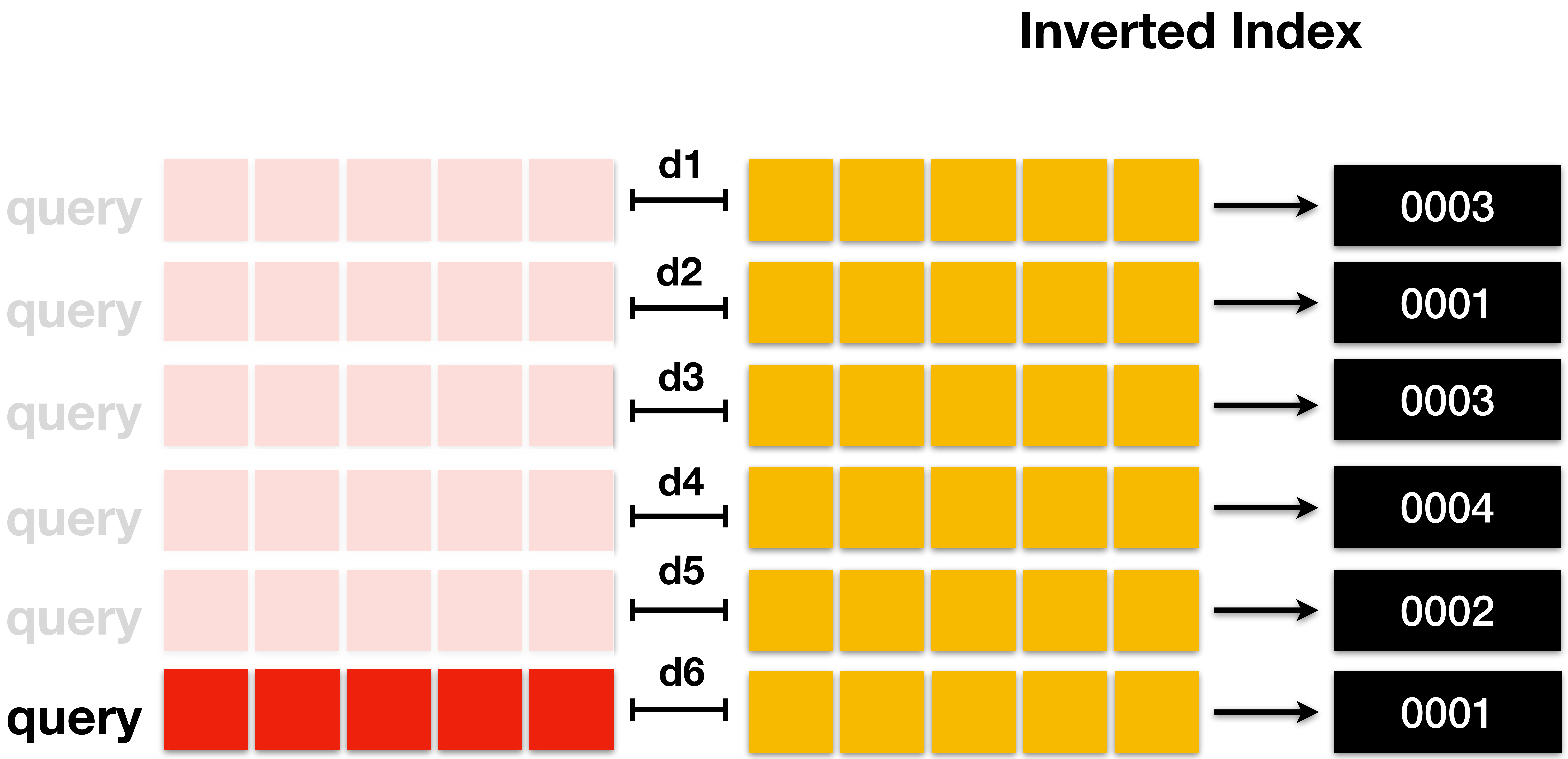


Brute Force Search

What is the computational complexity?

Linear: *O(n)*, where *n* is the number of features.

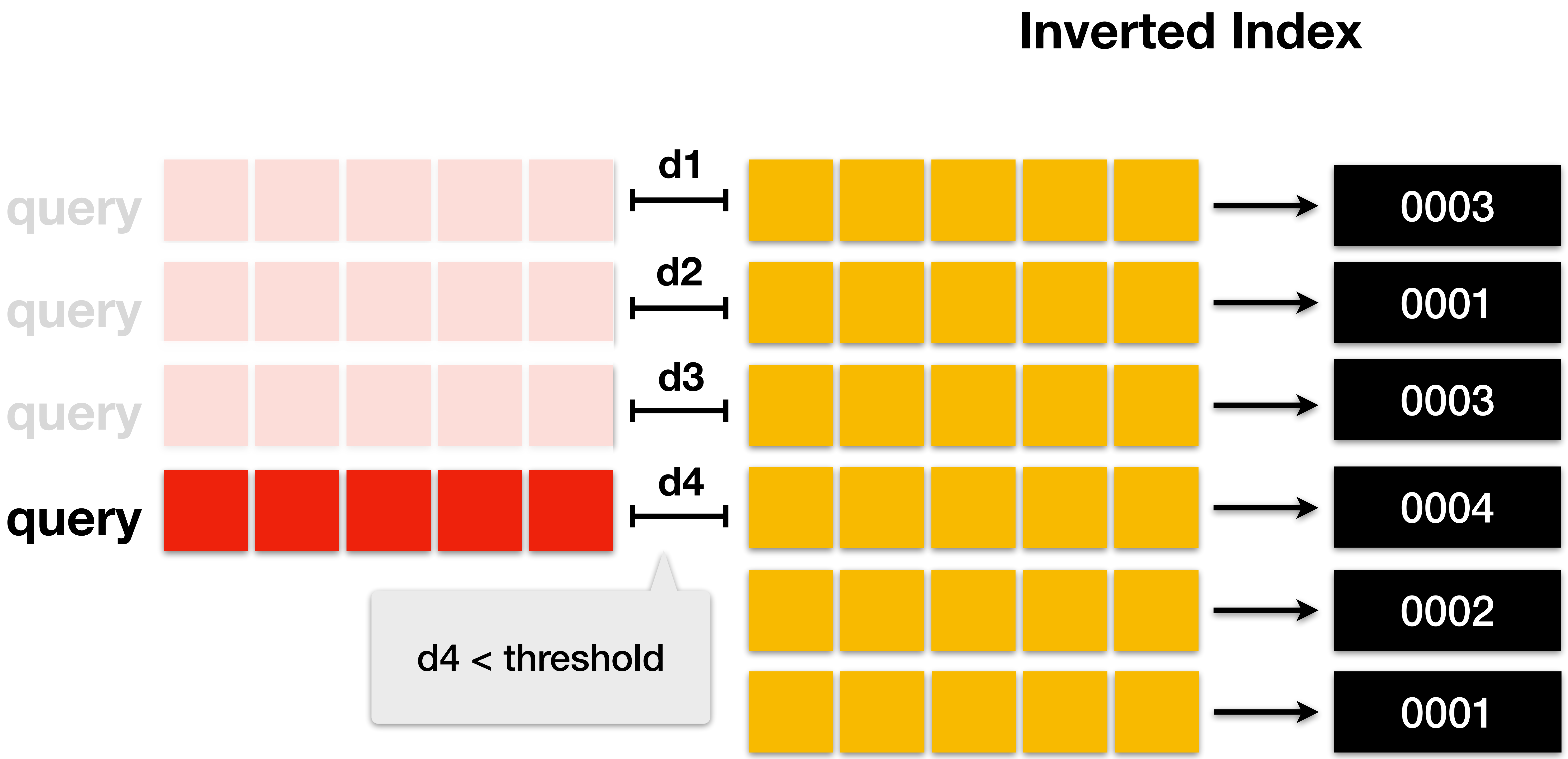
How to reduce it?



Early Stop Search

How to reduce complexity?

Stop when you find a feature that is close enough.



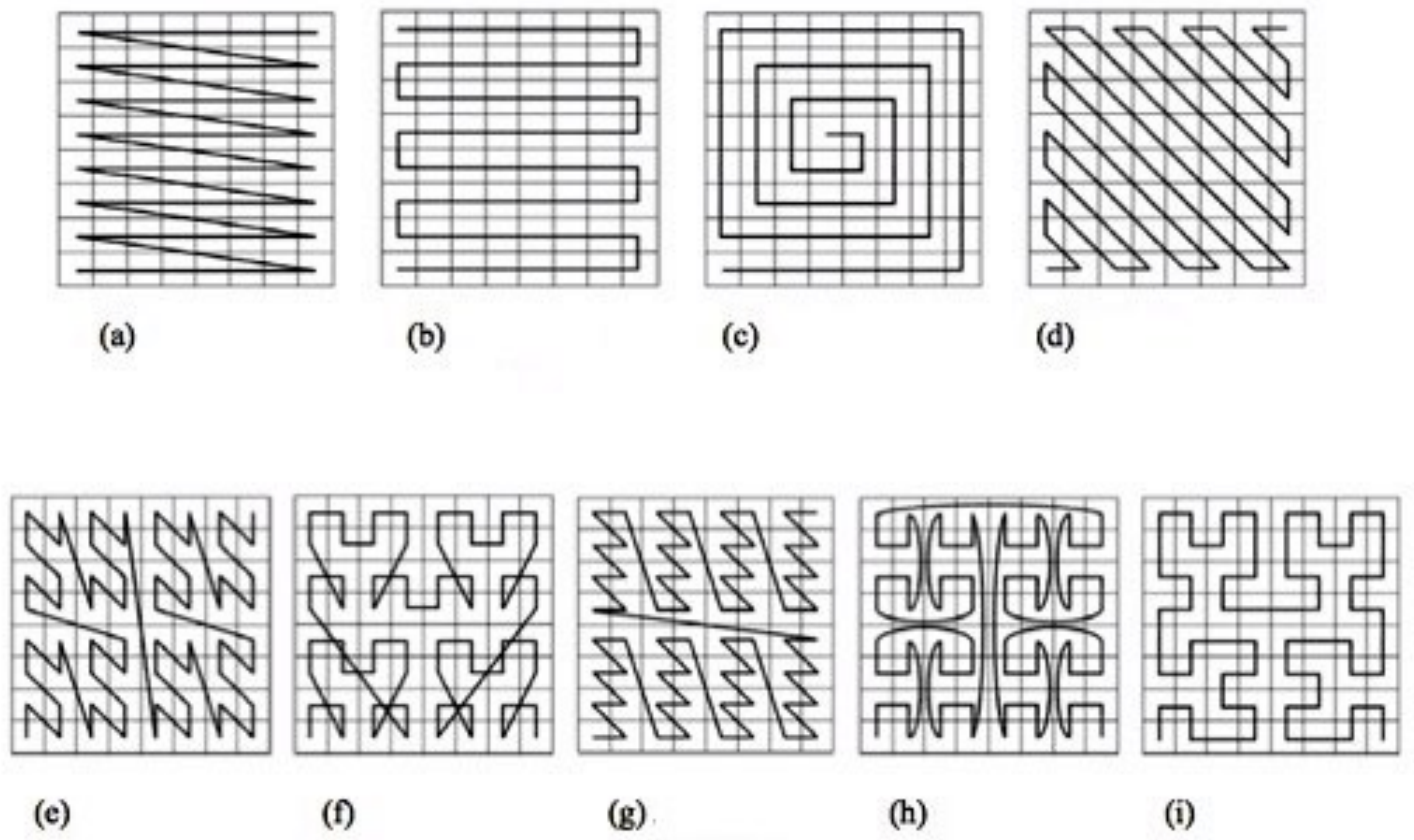
LOYOLA UNIVERSITY CHICAGO

Space Filling Curves

2D space examples

How to reduce complexity?

Curves determined by index mapping functions that pass once through every point of an *N*-dimensional space.

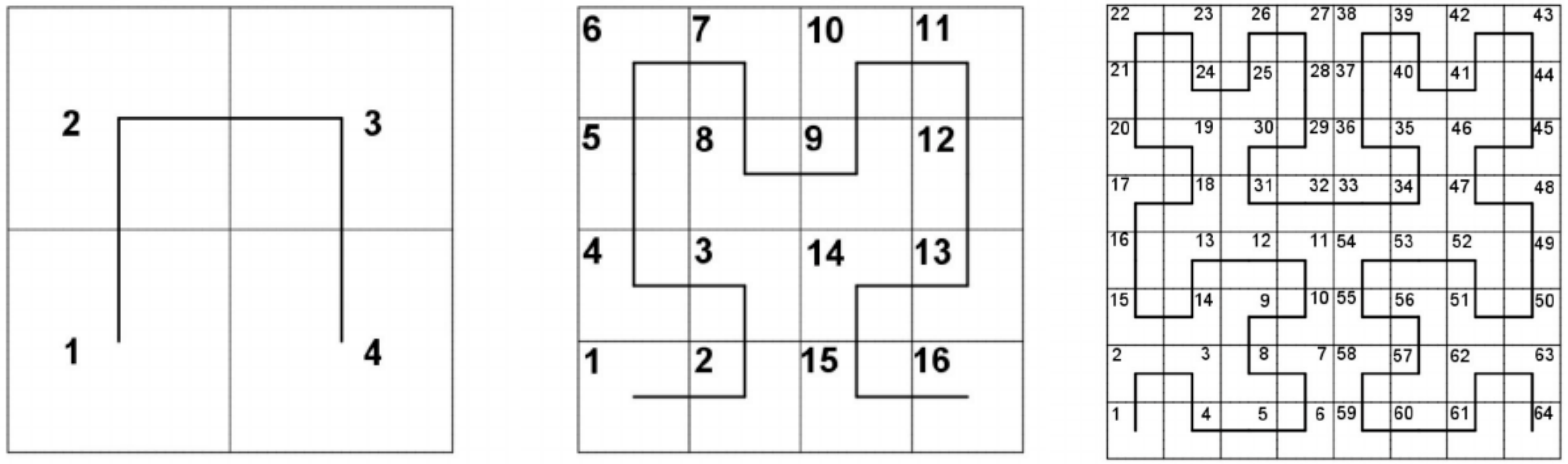


Space Filling Curves

How to reduce complexity?

Curves determined by index mapping functions that pass once through every point of an *N*-dimensional space.

2D space examples



Hilbert curves

LOYOLA UNIVERSITY CHICAGO

Space Filling Curves

How to reduce complexity?

Curves determined by index mapping functions that pass once through every point of an *N*-dimensional space.

The mapping functions are executed in constant time, w.r.t. the number of features.

3D space examples

Hilbert curves

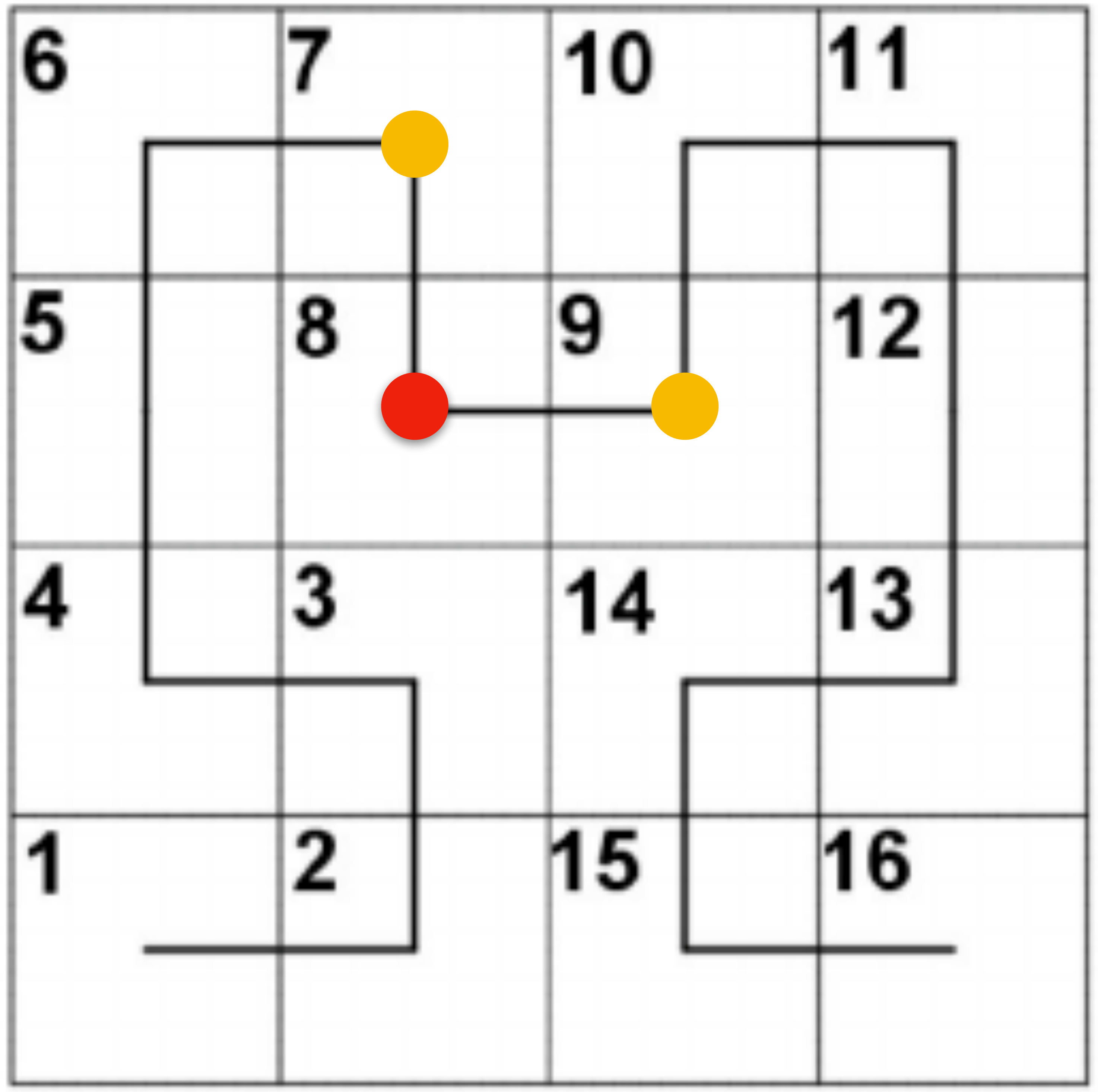
Space Filling Curves

How to reduce complexity?

The curves are 1D and the elements indexed by them are “sorted” in an *approximation* of their distances in the original space.

If the curve is used as a binary tree, an approximation of the k-nearest elements can be obtained in *O(log(n))*, where *n* is the number of features.

Example: 2-nearest elements



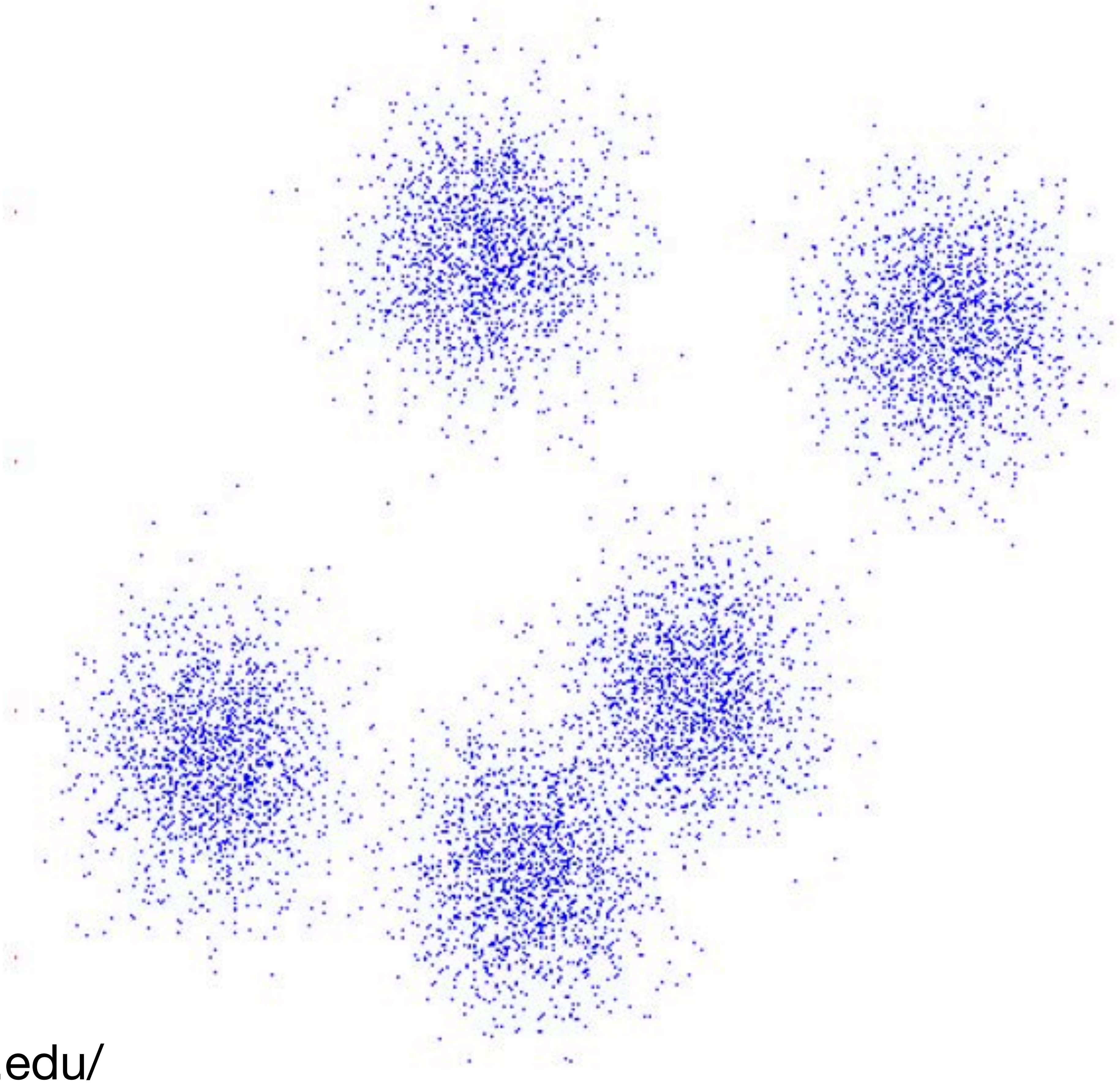
LOYOLA UNIVERSITY CHICAGO

Clustering

How to reduce complexity?

Cluster the features and limit the k-nearest search to one or a couple of clusters.

There will be less elements to consider



Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

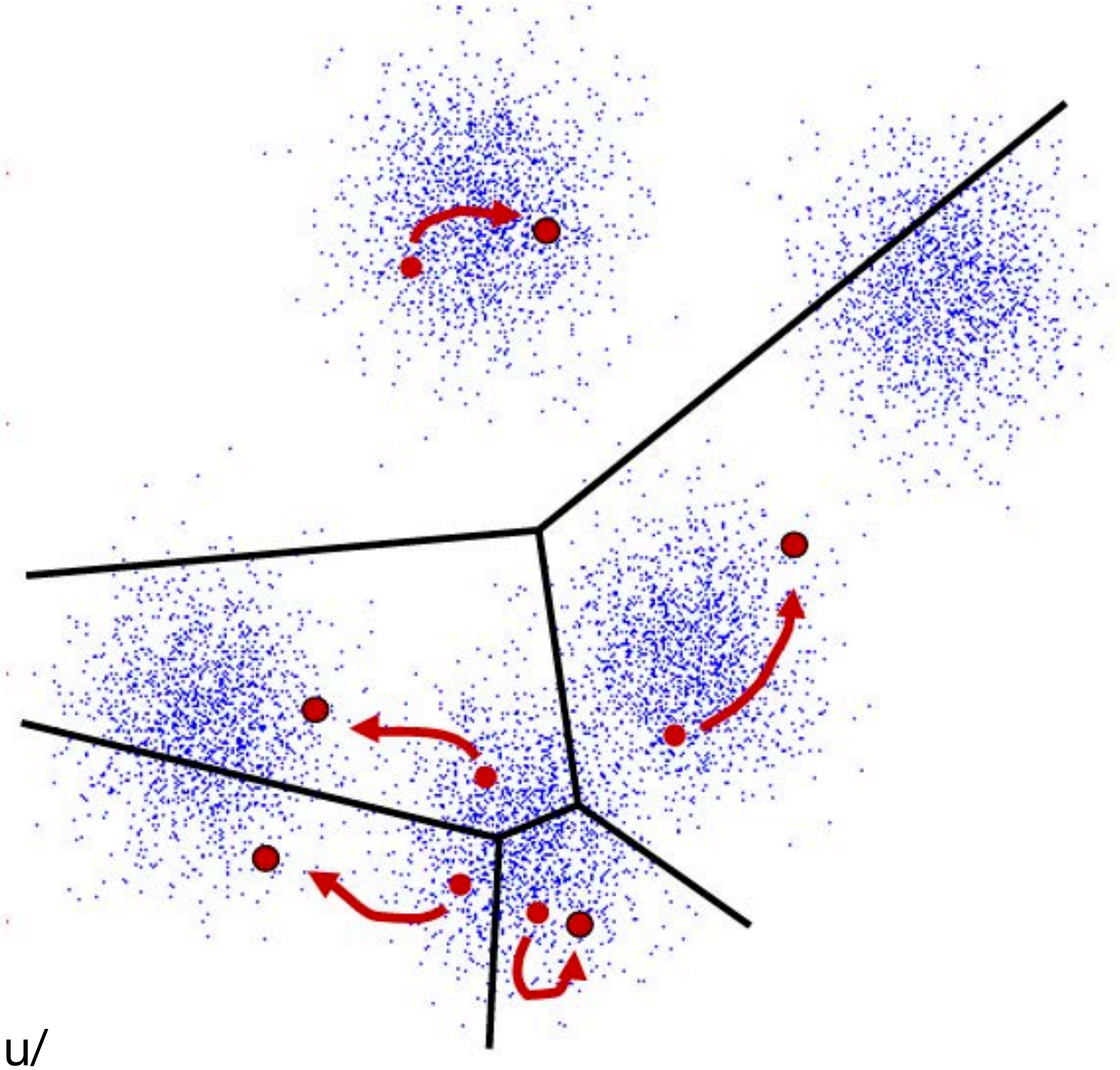
Clustering

How to reduce complexity?

Cluster the features and limit the k-nearest search to one or a couple of clusters.

There will be less elements to consider

K-Means



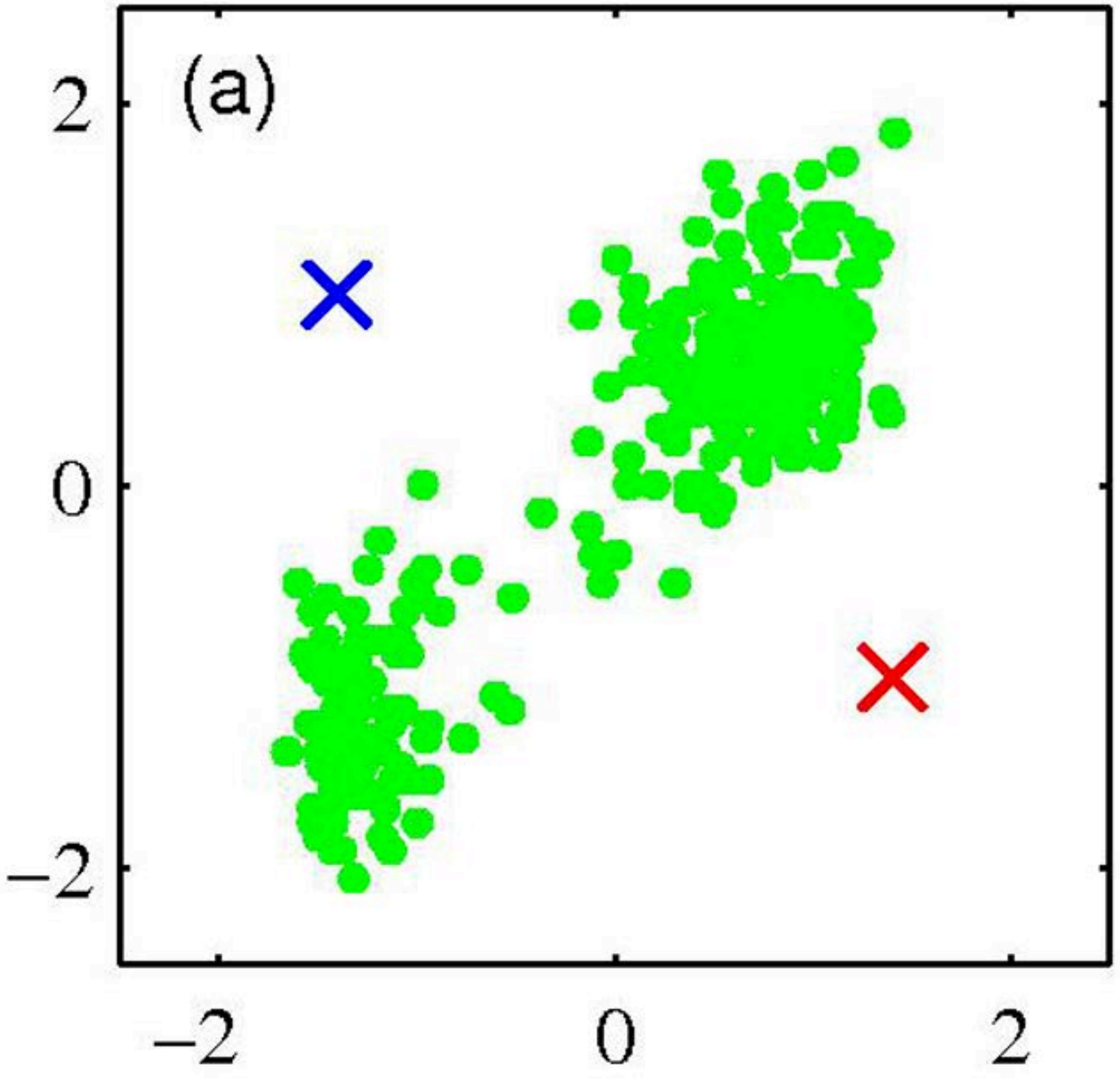
Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

LOYOLA UNIVERSITY CHICAGO

Clustering

K-Means

Select K random features as cluster centers.



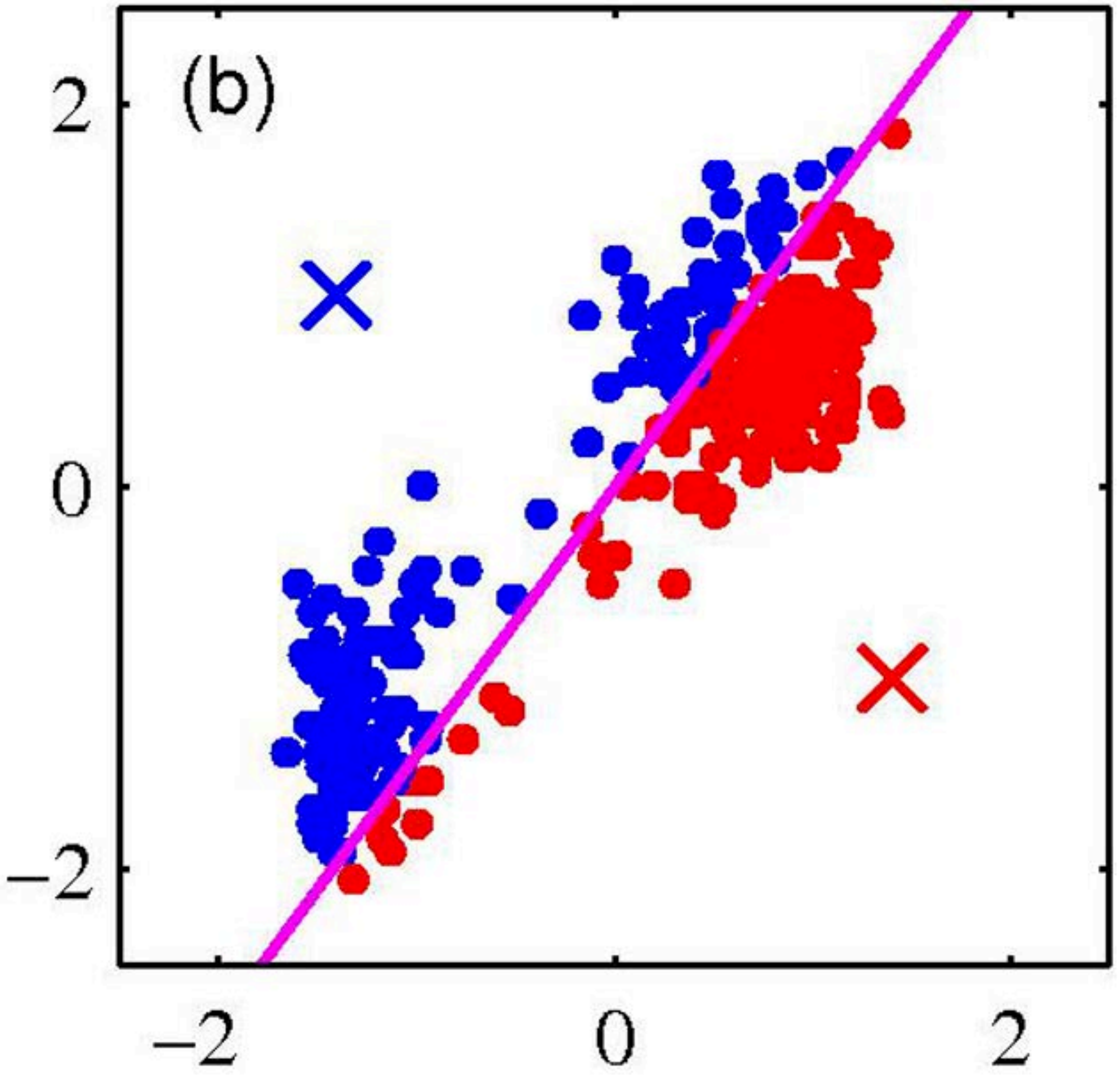
Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

LOYOLA UNIVERSITY CHICAGO

Clustering

K-Means

Assign features to closes cluster centers.



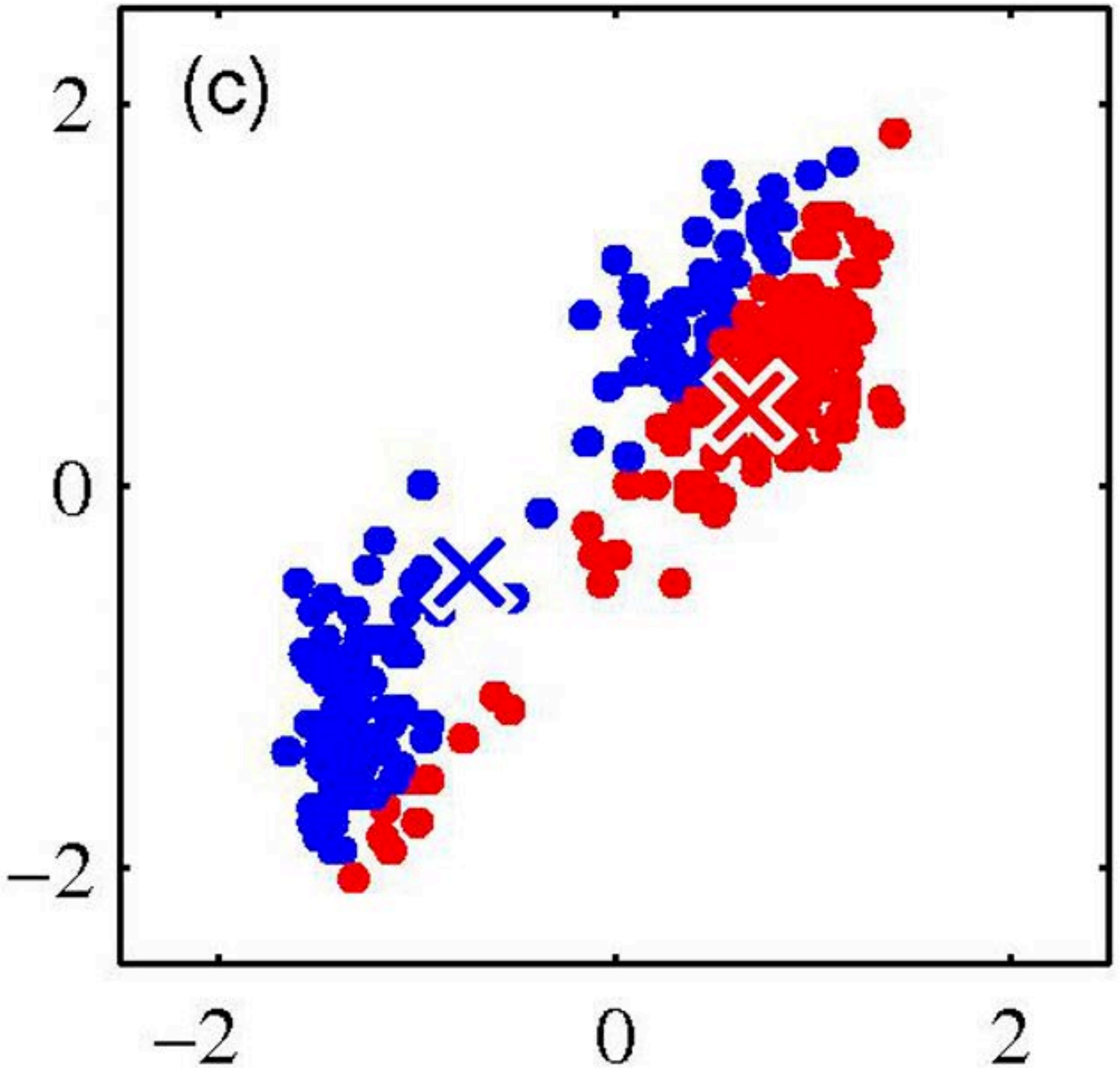
Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

LOYOLA UNIVERSITY CHICAGO

Clustering

K-Means

Update the cluster centers by taking the **means** of each cluster.

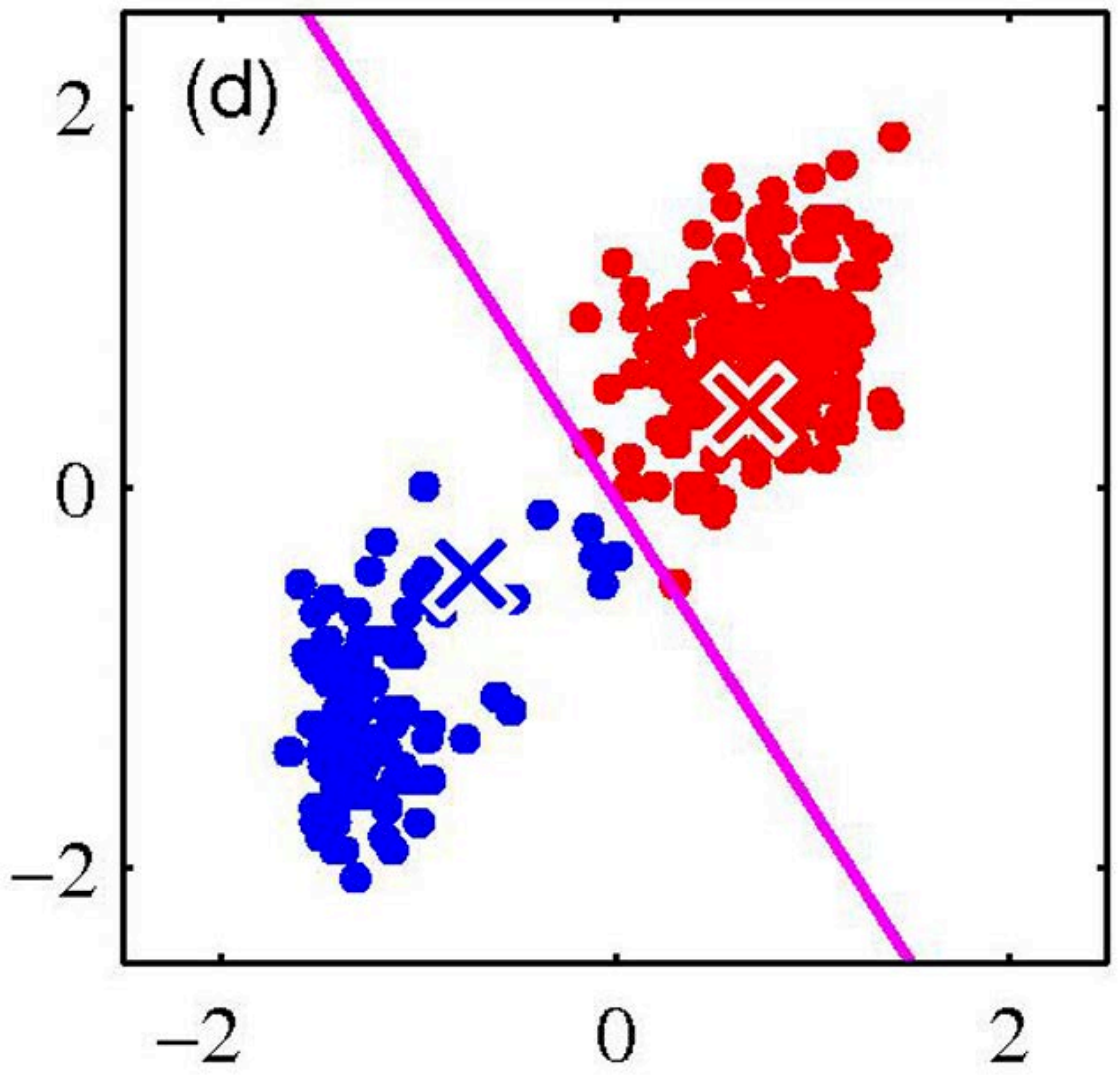


Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

Clustering

K-Means

Repeat until convergence.



Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

LOYOLA UNIVERSITY CHICAGO

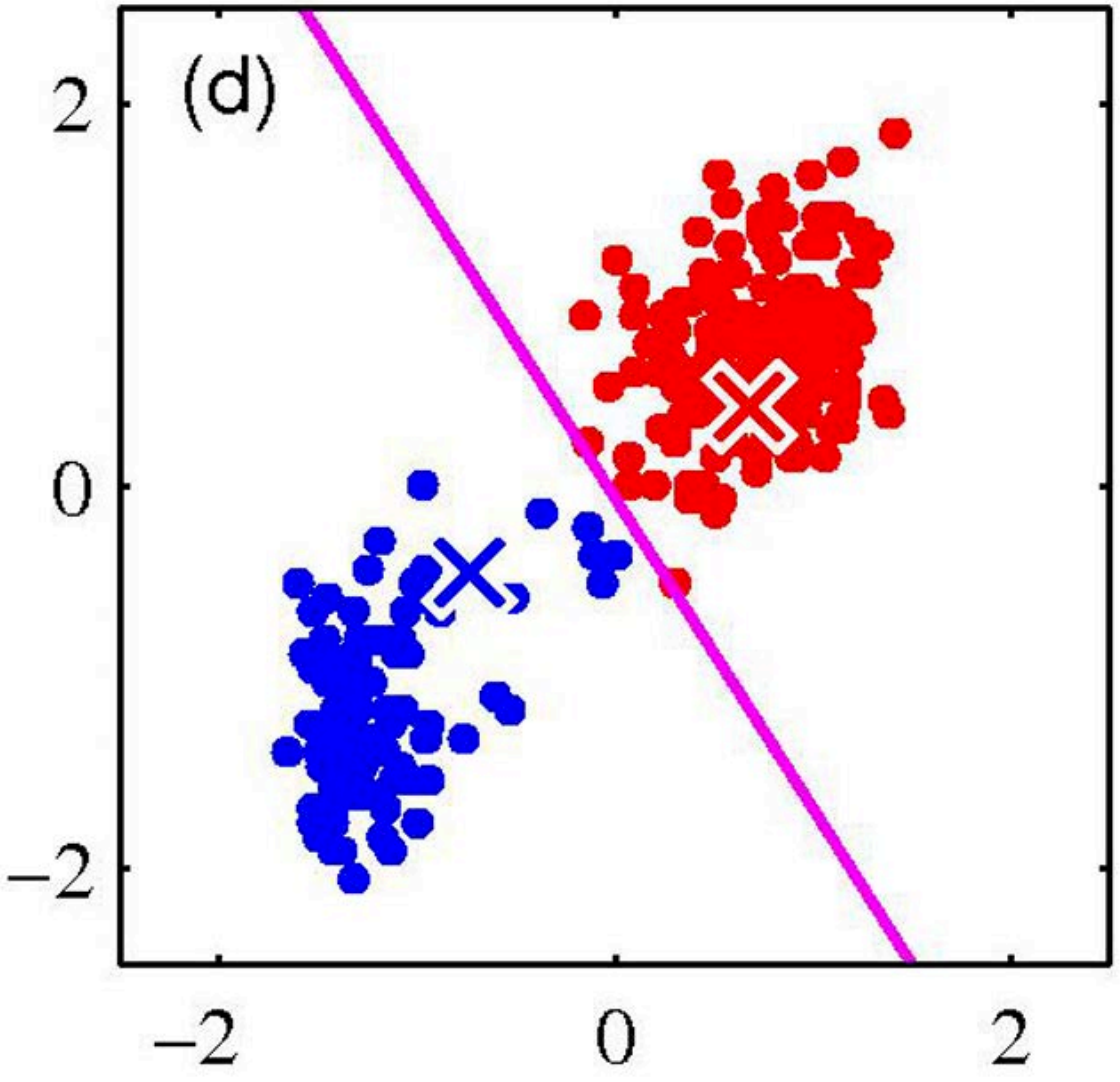
Clustering

K-Means

What are the limitations of this approach?

- What is the ideal number of clusters?
- Complexity of building clusters: *O(Kn)* in each step until convergence.
- Clustering is *offline*: i.e., it does not happen at feature querying time.

K: #clusters
n: #features

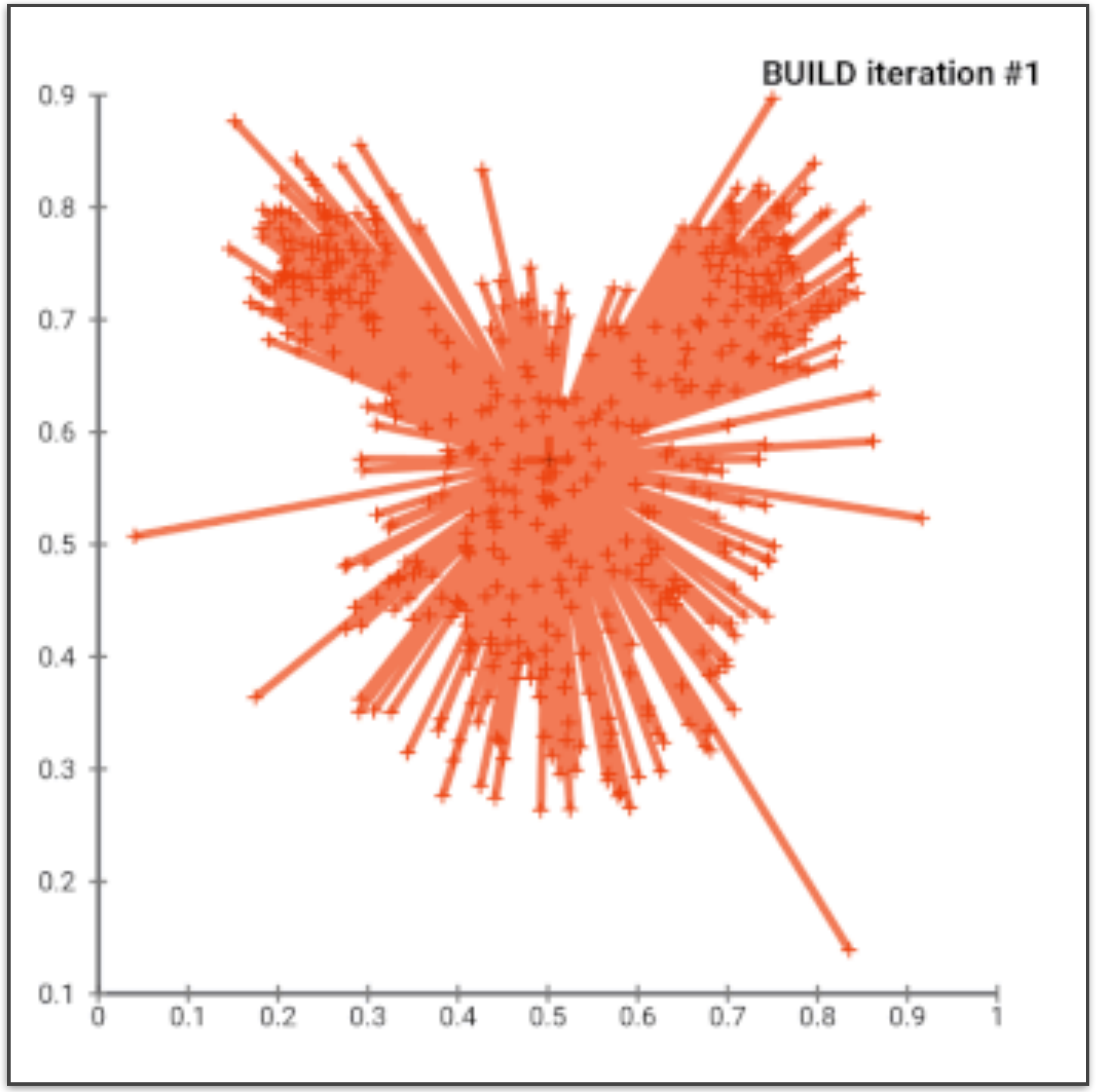


LOYOLA UNIVERSITY CHICAGO

Clustering

Variation: K-medoids

Instead of using *means* as the cluster centers, use *medians*, which are actual existing features.

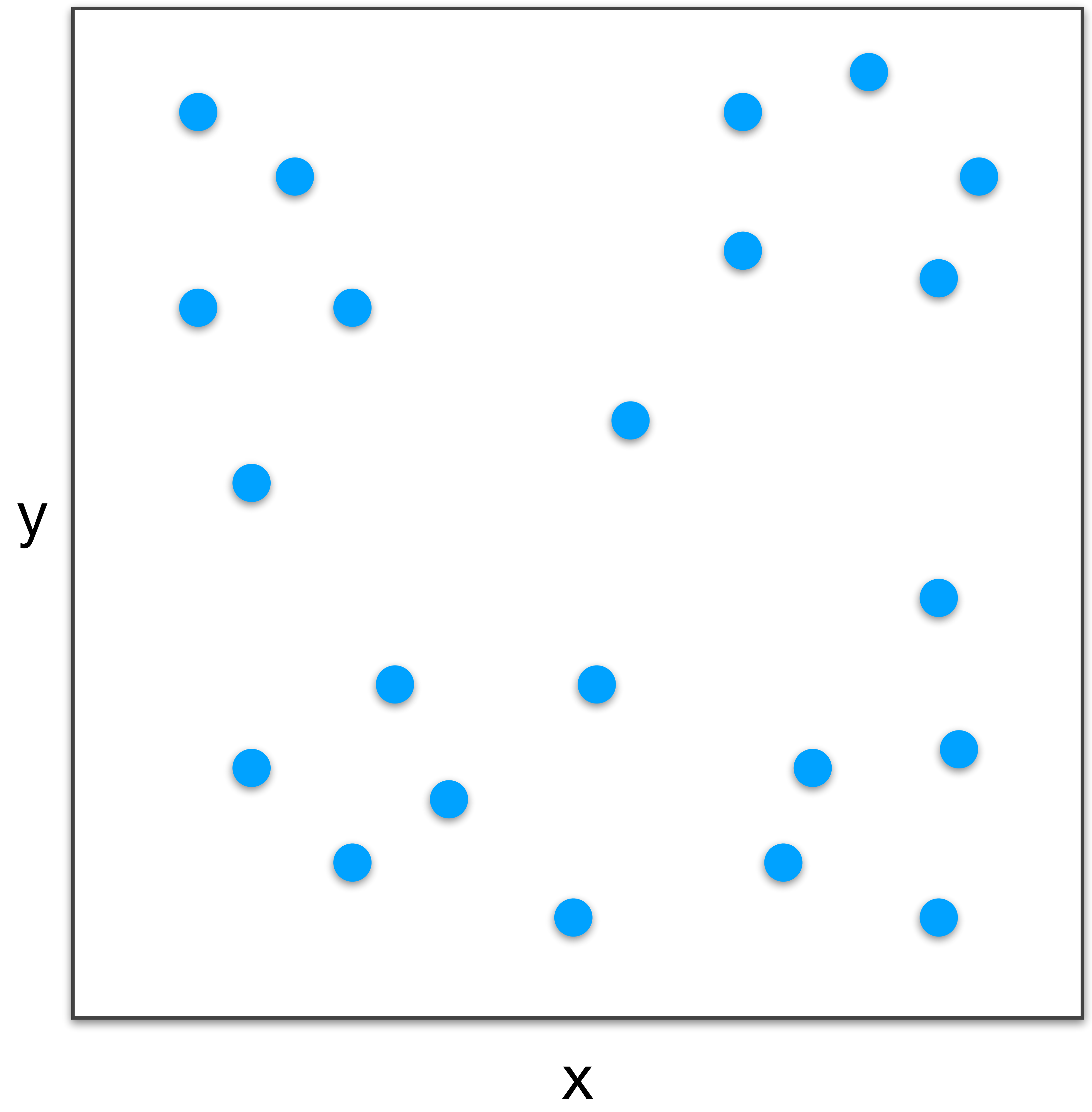


KD Trees

How to reduce complexity?

K-dimensional trees:
For K times
Split one feature dimension into two halves.

2D-features toy case



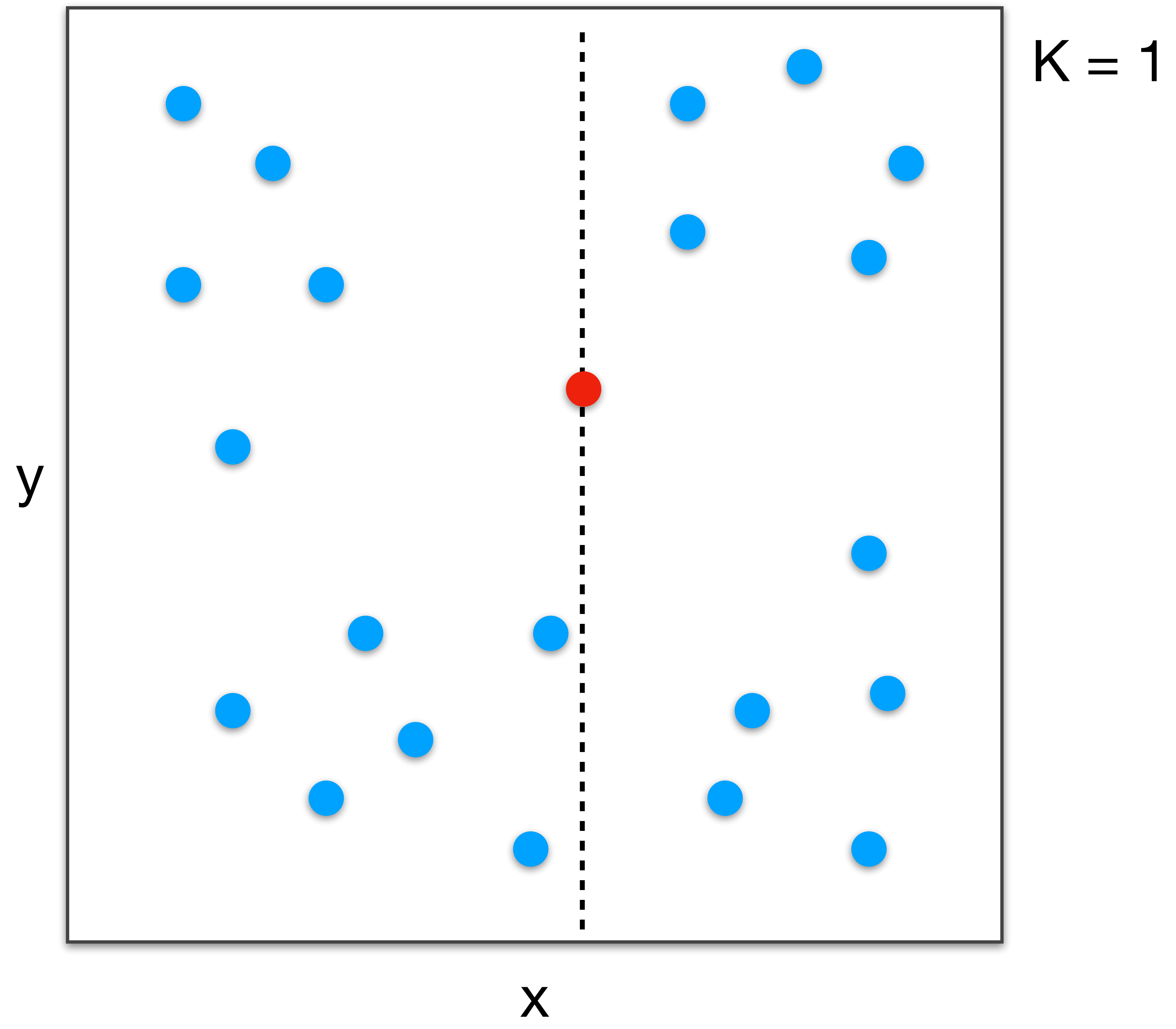
KD Trees

How to reduce complexity?

K-dimensional trees:
For K times
   Split one feature dimension into two partitions using medians.

indices

2D-features toy case



LOYOLA UNIVERSITY CHICAGO

KD Trees

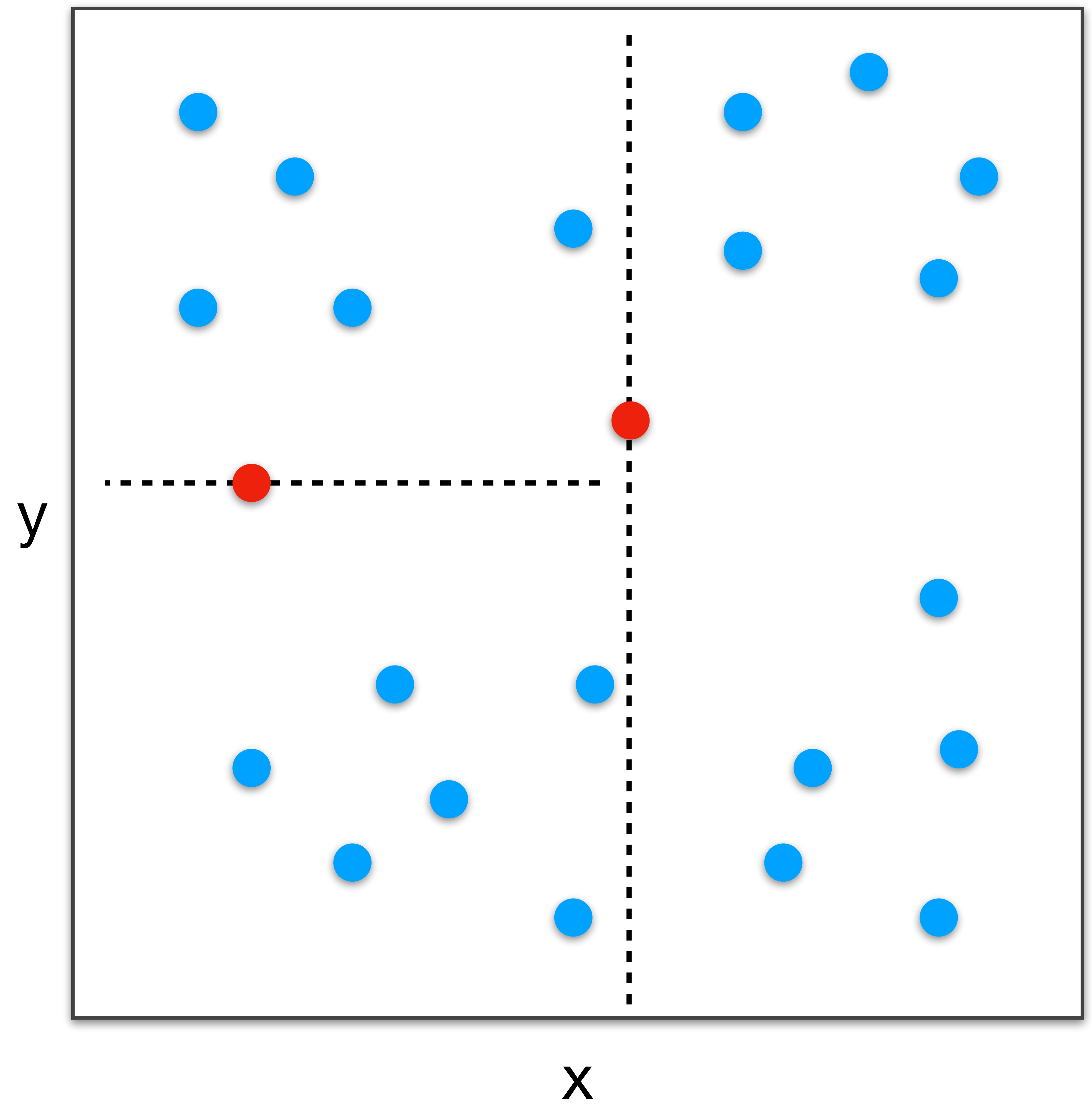
How to reduce complexity?

K-dimensional trees:
For K times
Split one feature dimension into two partitions using medians.

indices

2D-features toy case

K = 2



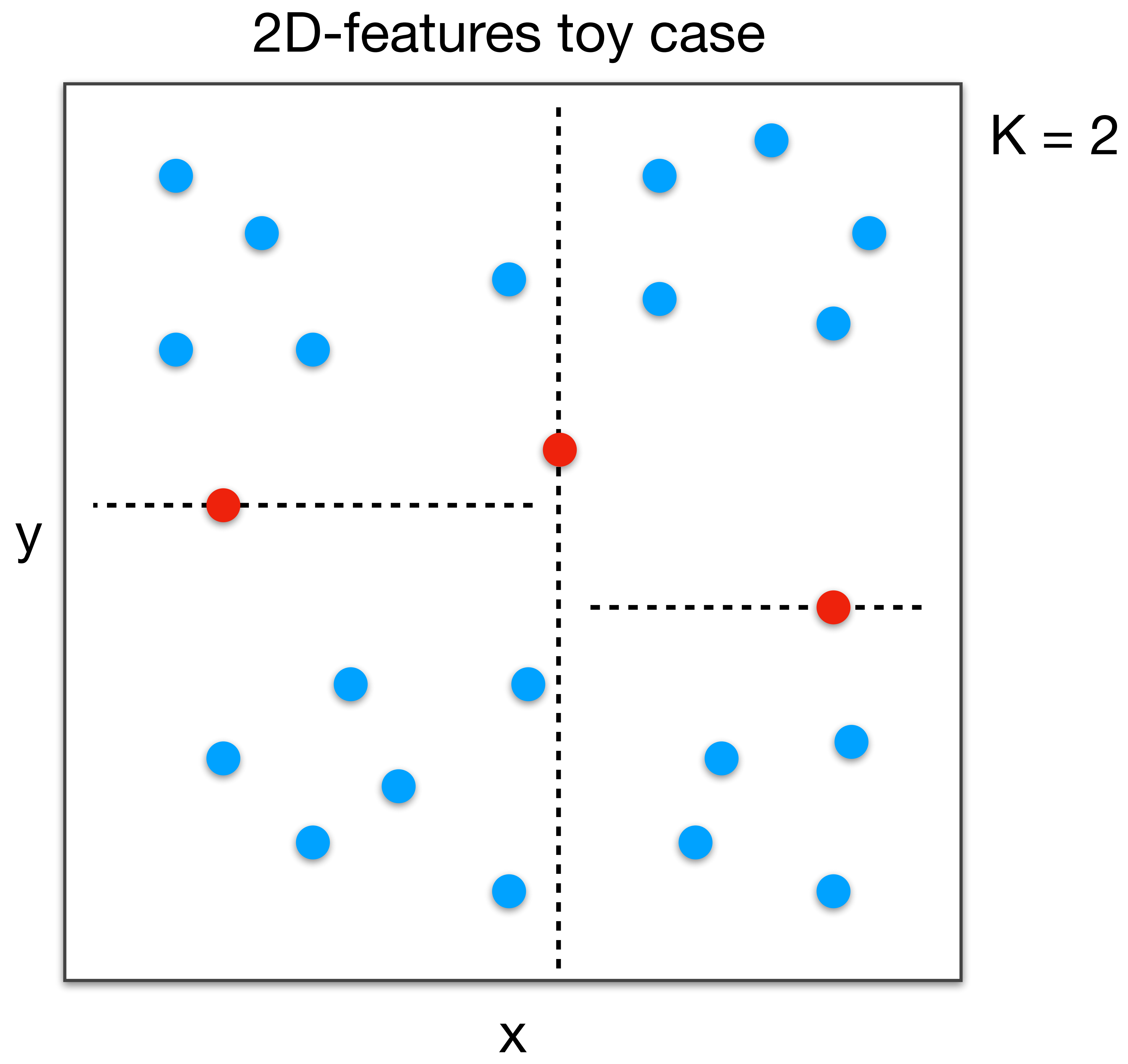
LOYOLA UNIVERSITY CHICAGO

KD Trees

How to reduce complexity?

K-dimensional trees:
For K times
Split one feature dimension into two partitions using medians.

indices



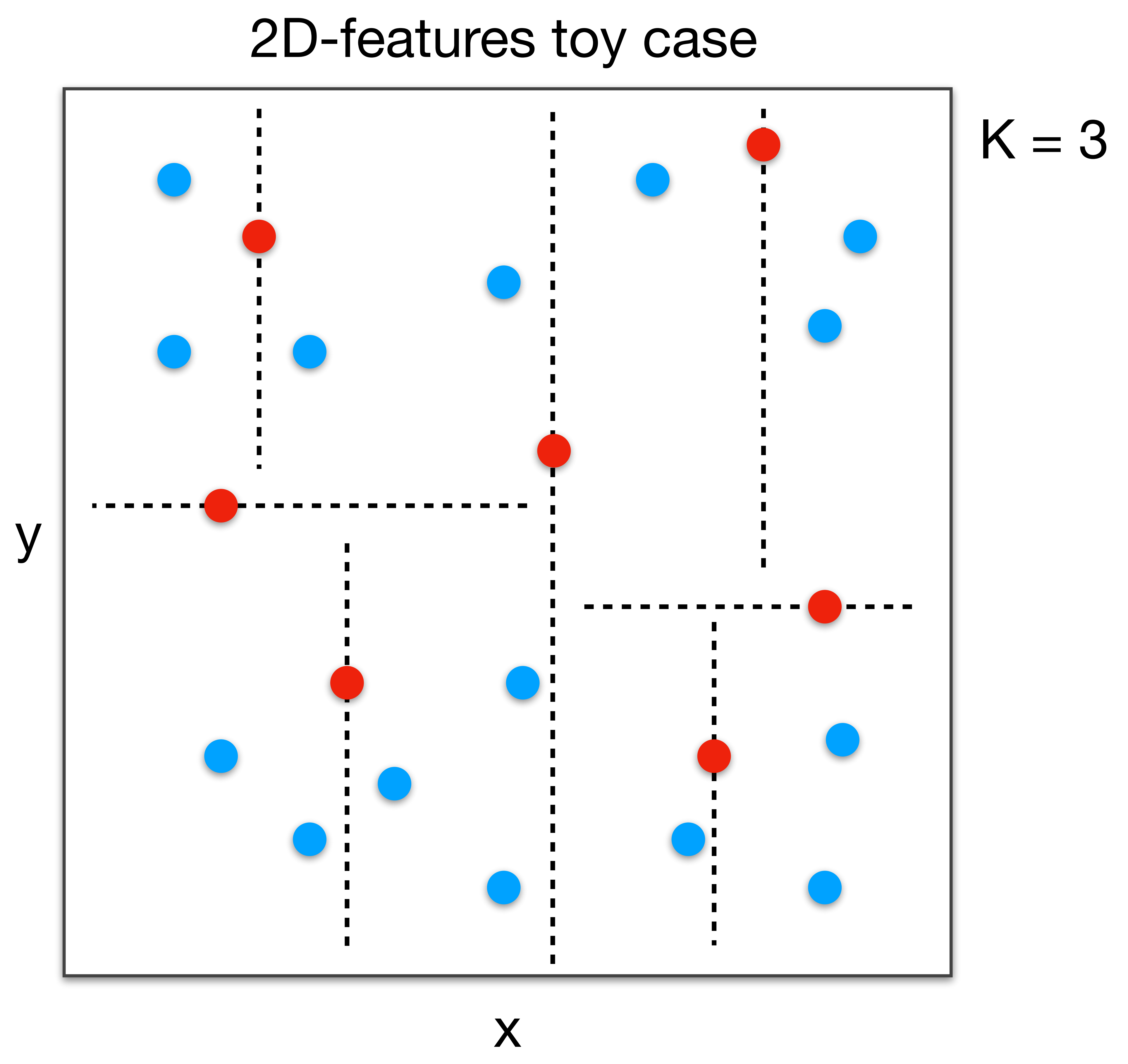
LOYOLA UNIVERSITY CHICAGO

KD Trees

How to reduce complexity?

K-dimensional trees:
For K times
Split one feature dimension into two partitions using medians.

indices



KD Trees

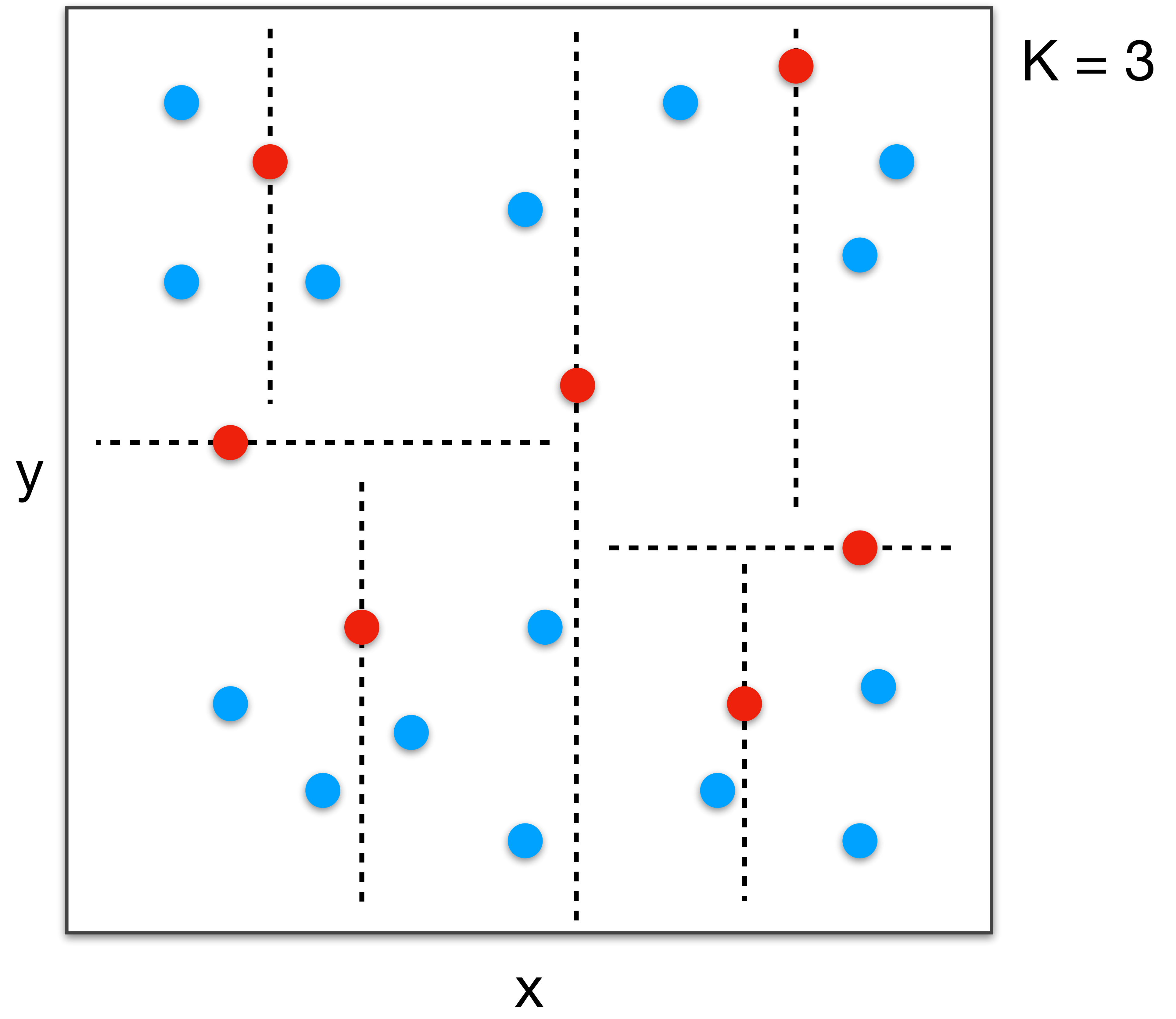
How to reduce complexity?

K-dimensional trees:
For K times
  Split one feature dimension into two partitions using medians.

Complexity to build: *O(n log(n))*

Building is *offline*: i.e., it does not happen at feature querying time.

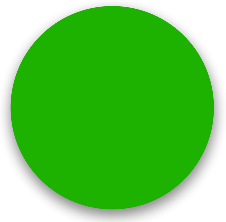
2D-features toy case



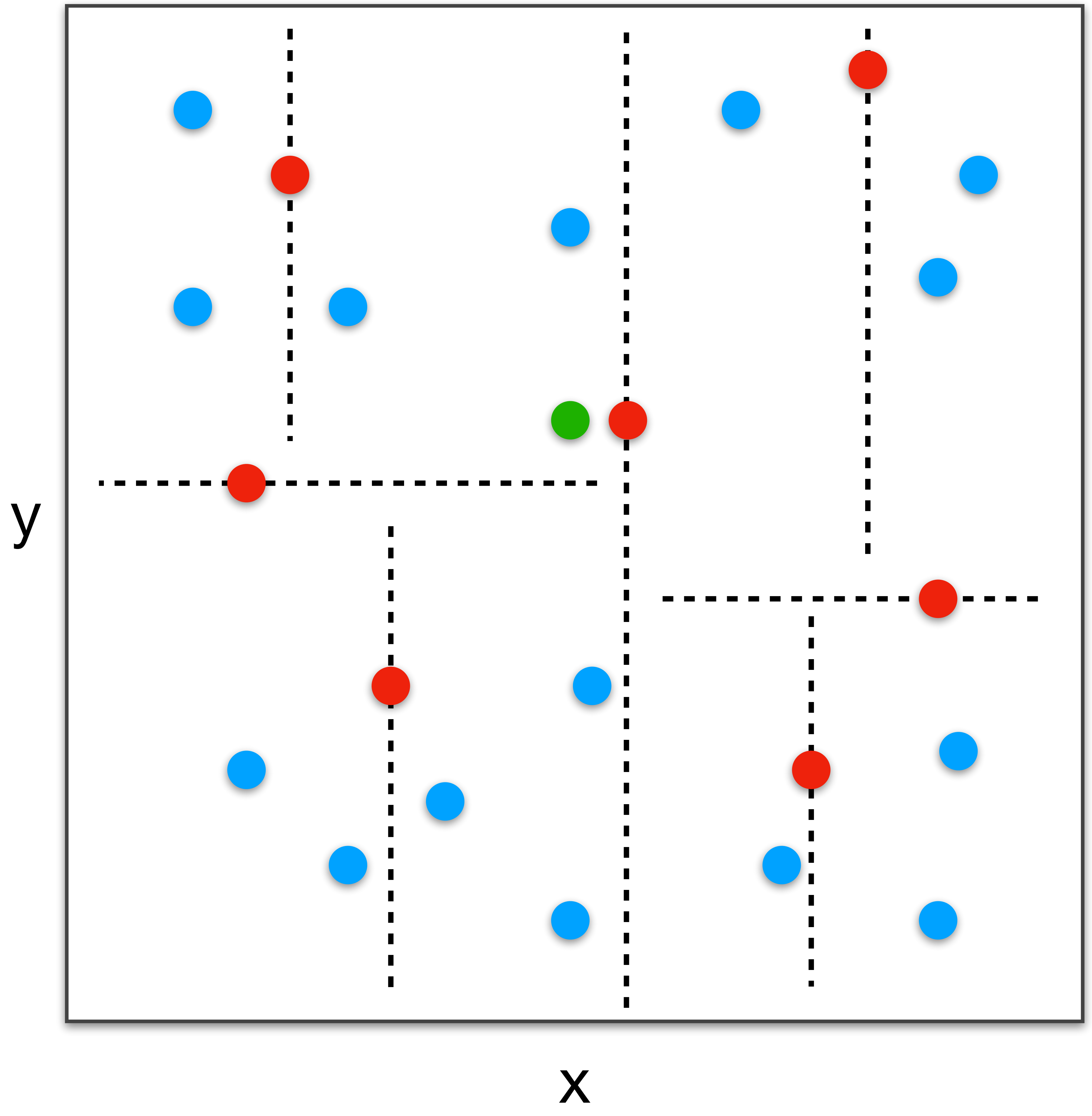
KD Trees

How to reduce complexity?

How to obtain 3-nearest neighbors?



2D-features toy case



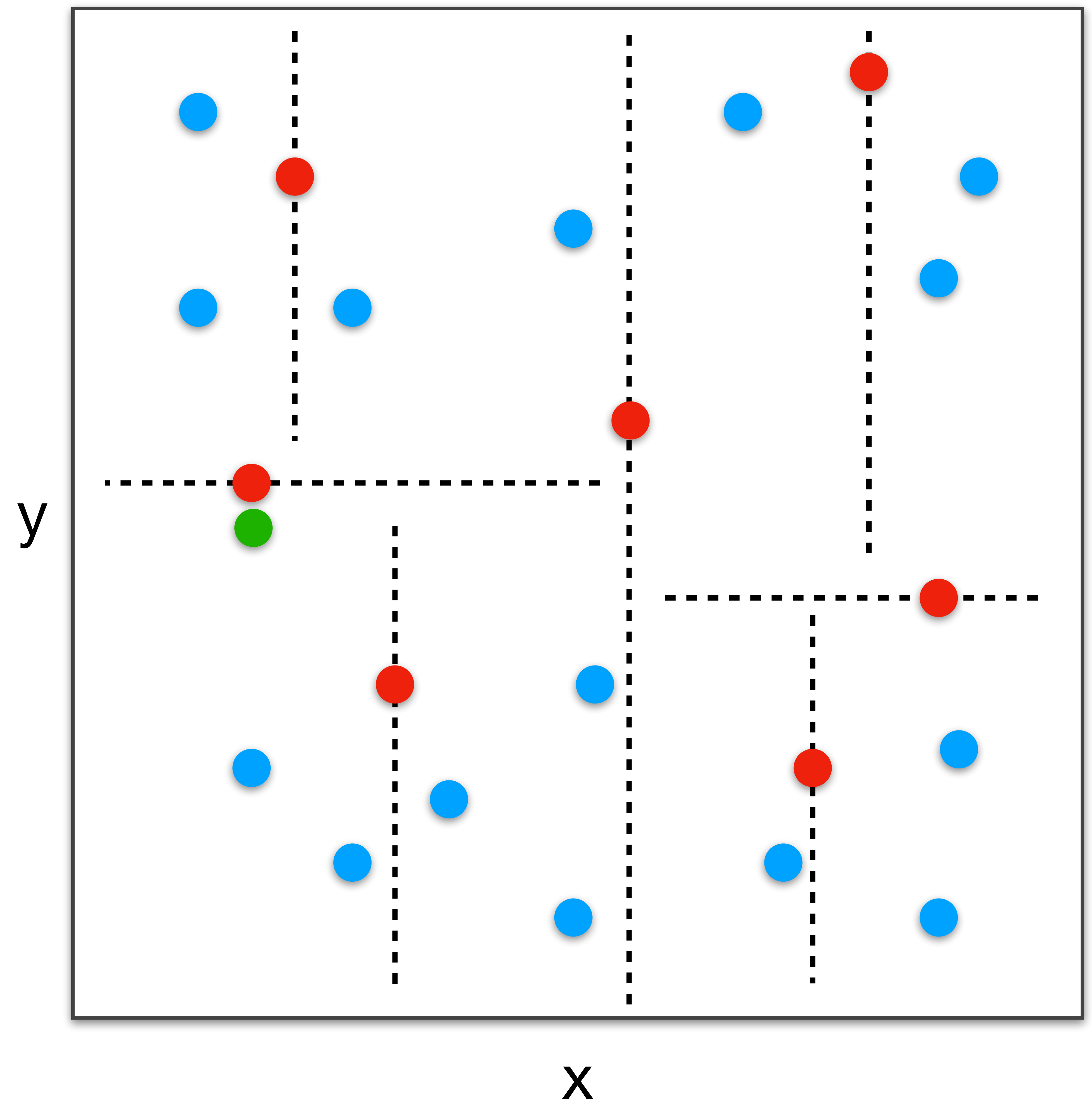
KD Trees

How to reduce complexity?

How to obtain 3-nearest neighbors?

query

2D-features toy case



K = 3

LOYOLA UNIVERSITY CHICAGO

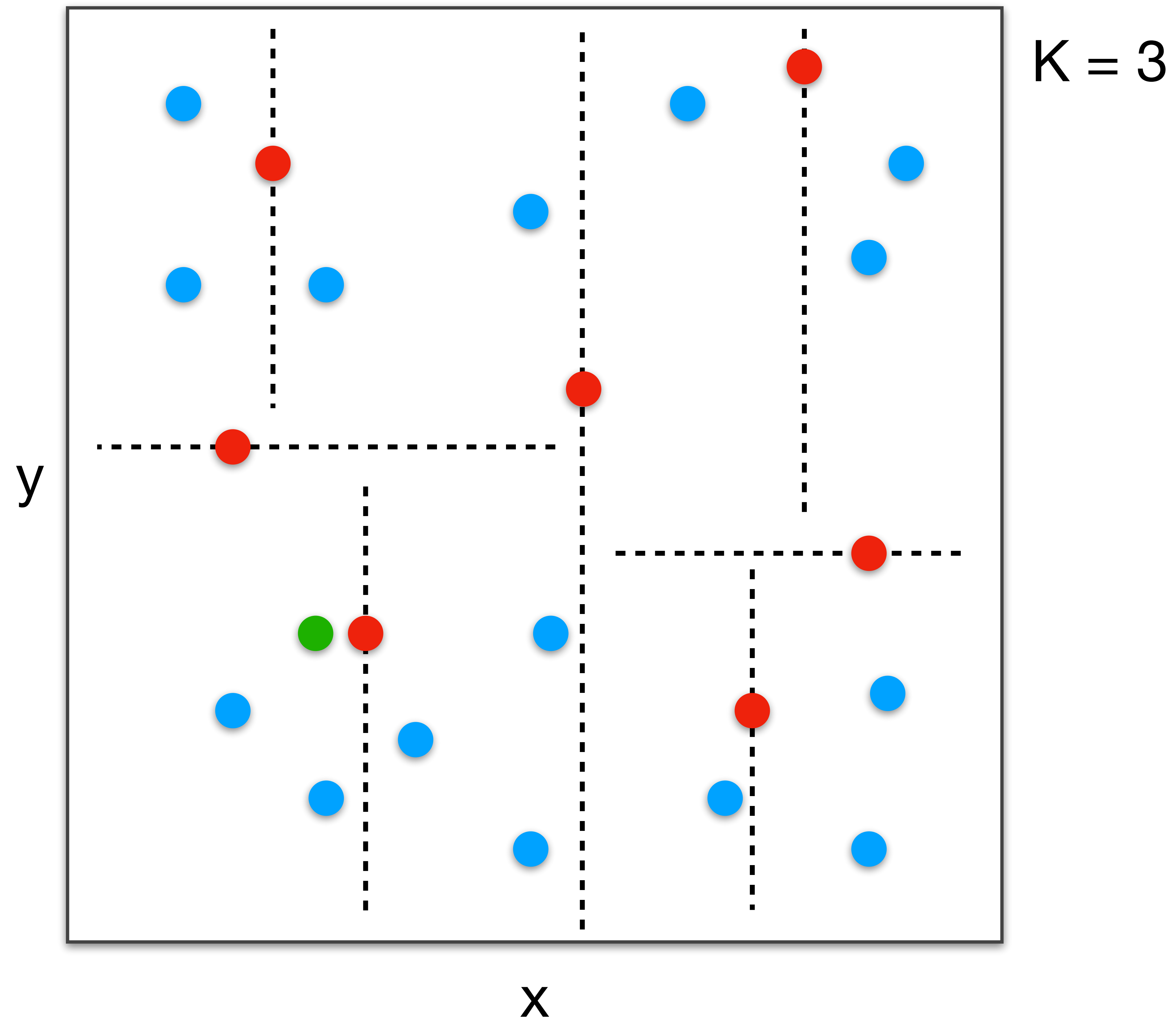
KD Trees

How to reduce complexity?

How to obtain 3-nearest neighbors?

query

2D-features toy case



LOYOLA UNIVERSITY CHICAGO

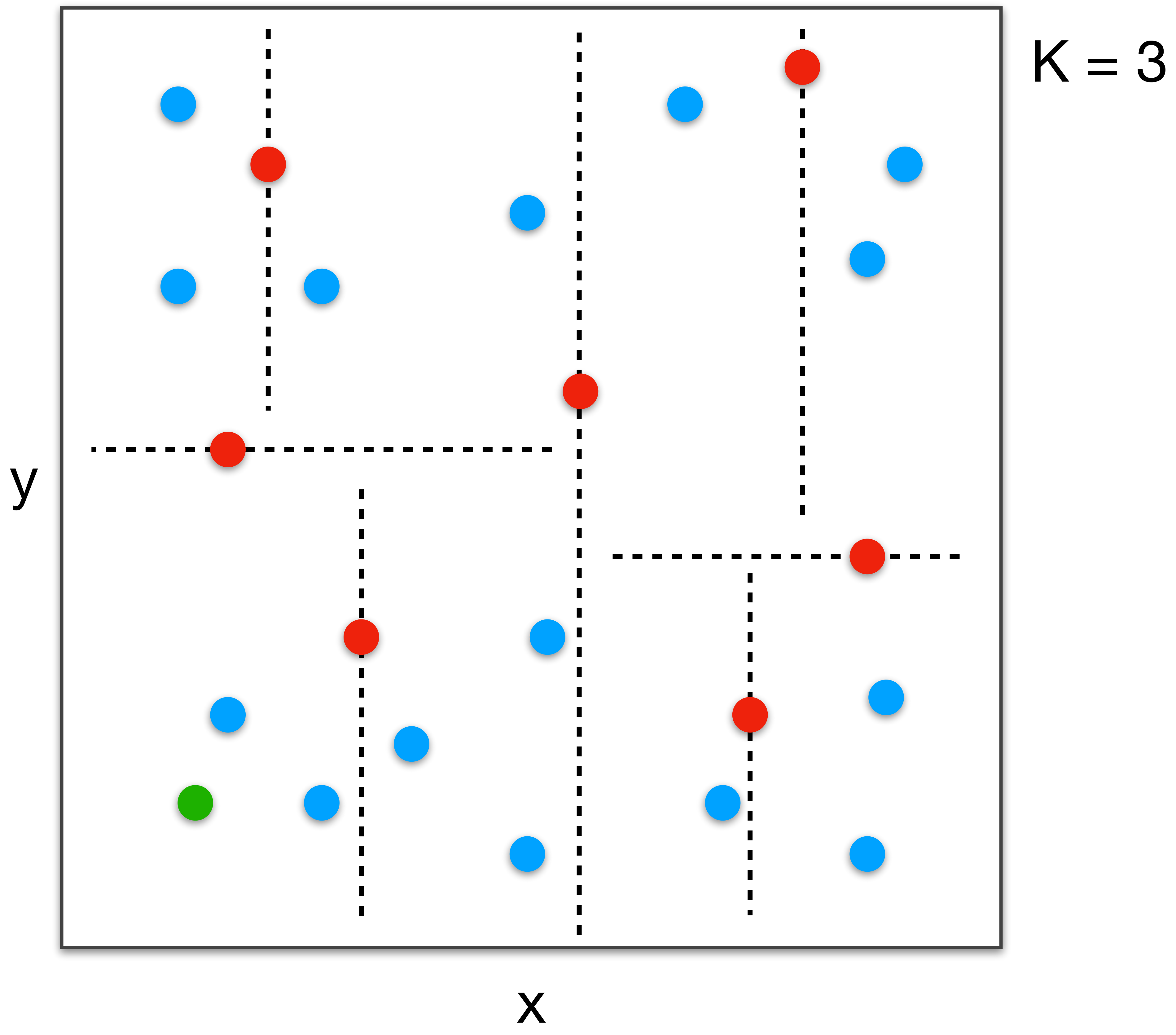
KD Trees

How to reduce complexity?

How to obtain 3-nearest neighbors?

query

2D-features toy case



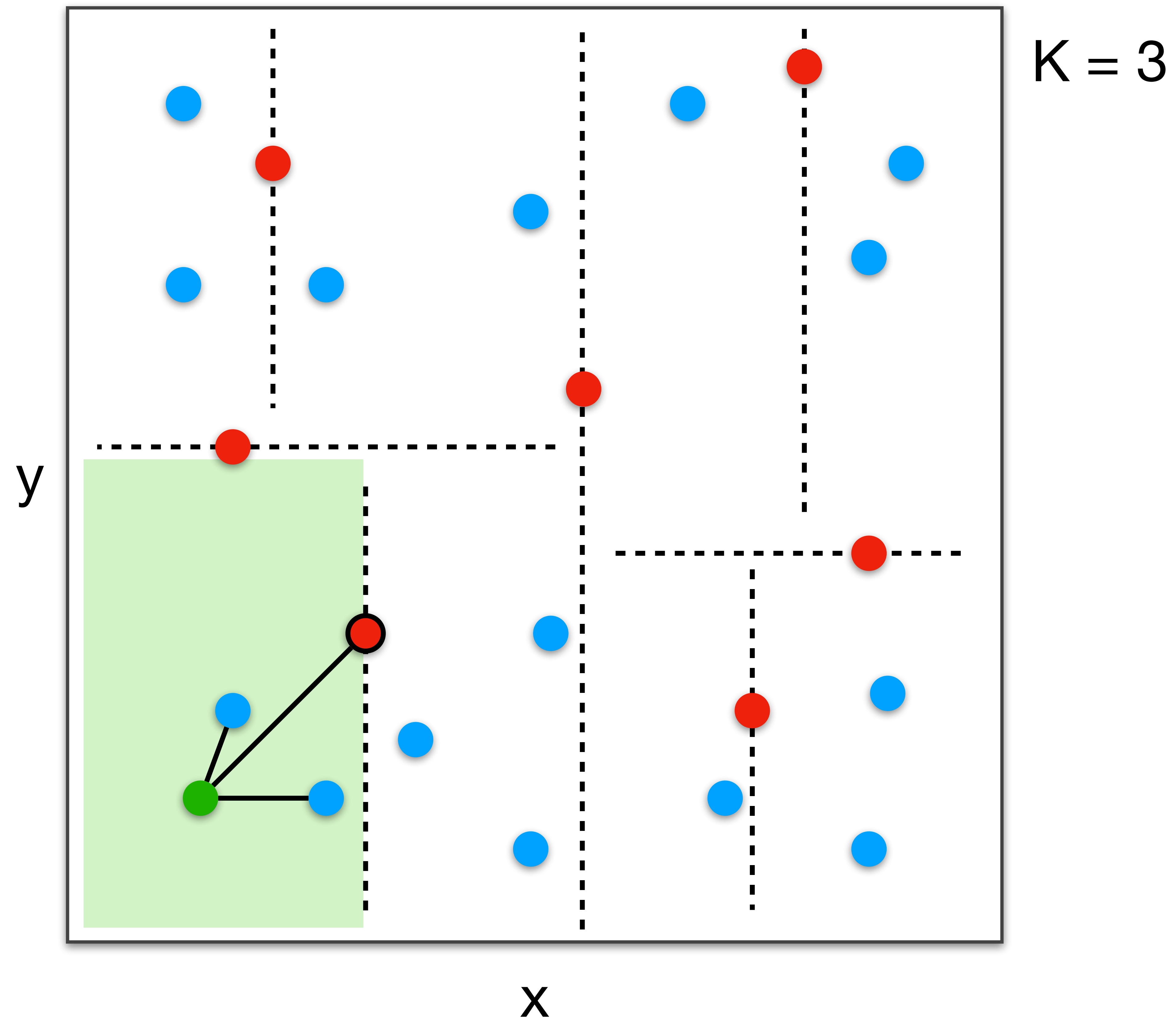
KD Trees

How to reduce complexity?

How to obtain 3-nearest neighbors?

query

2D-features toy case

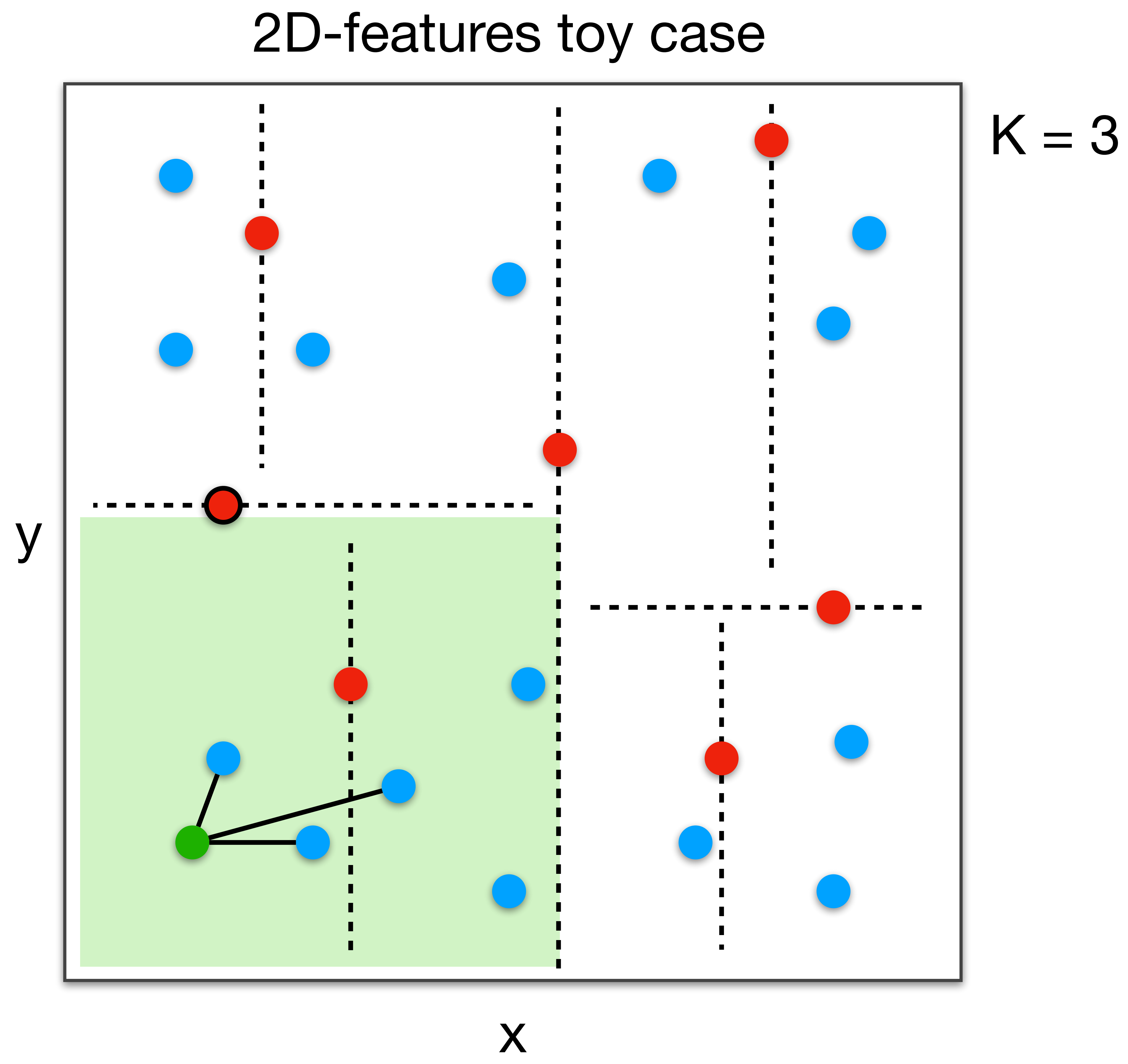


KD Trees

How to reduce complexity?

How to obtain 3-nearest neighbors?

query



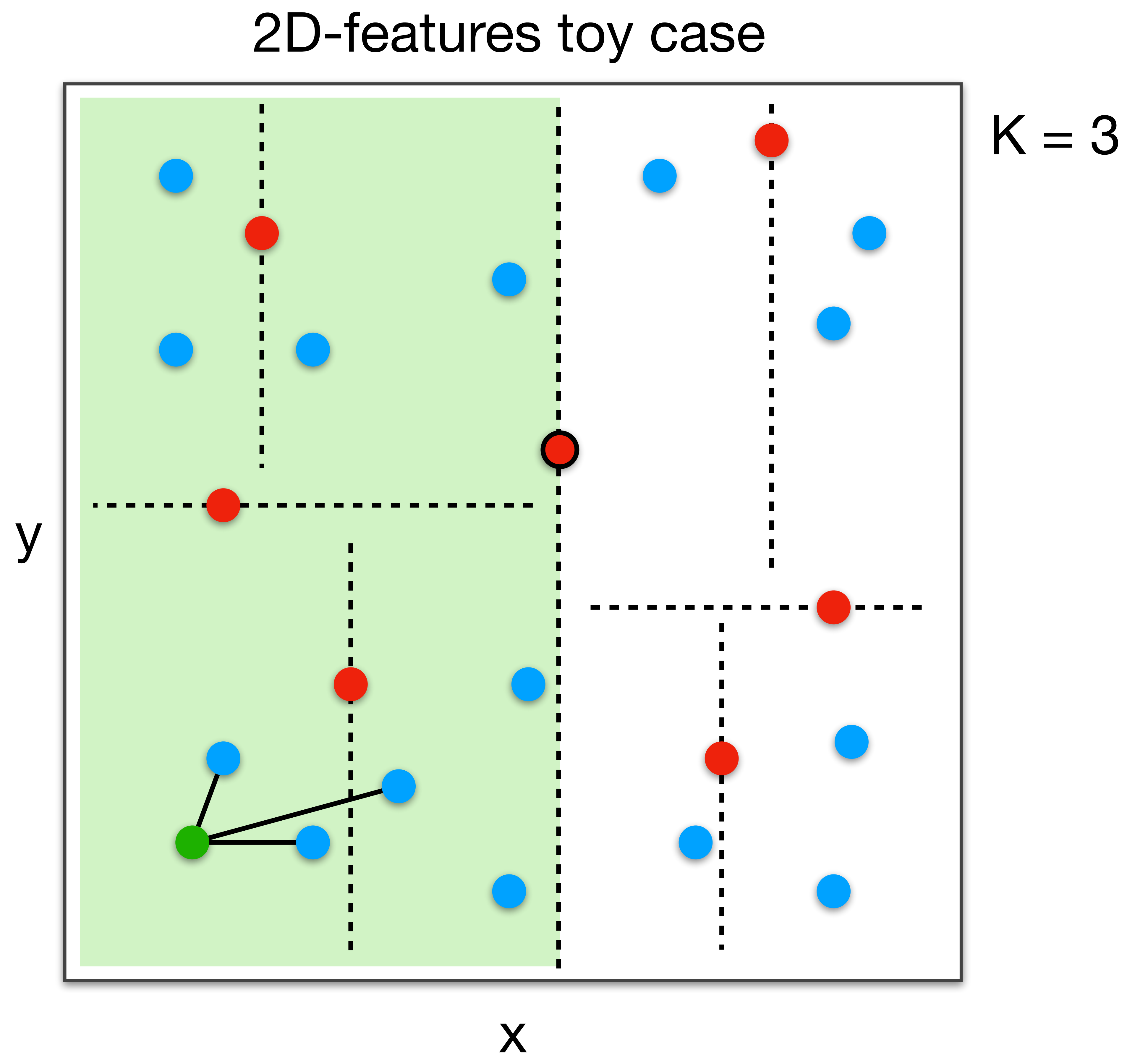
KD Trees

How to reduce complexity?

How to obtain 3-nearest neighbors?

query

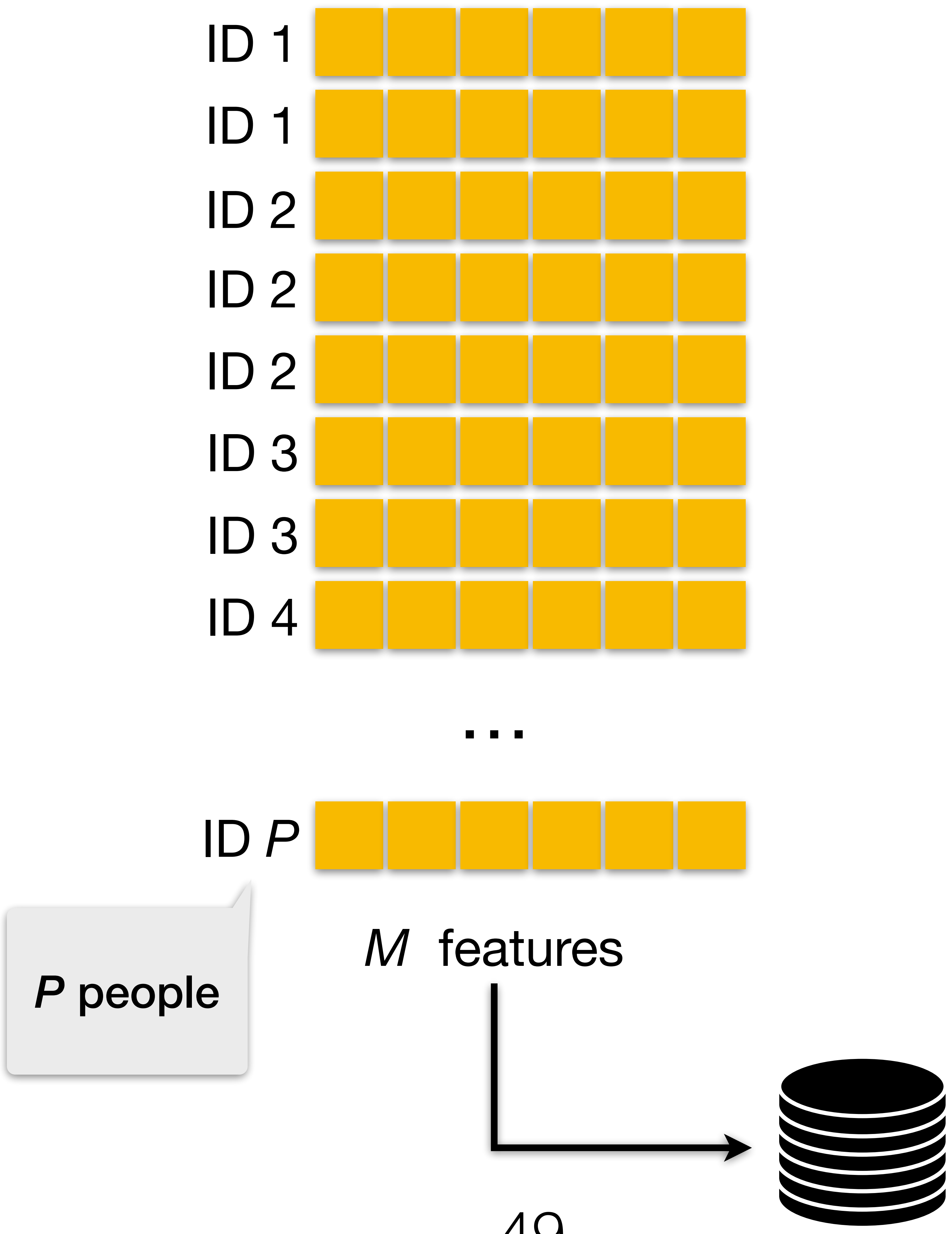
No changes in 3-nearest, so stop.



Product Quantization

How to reduce size?

Toy Case (6D features, reality: 512D for faces)



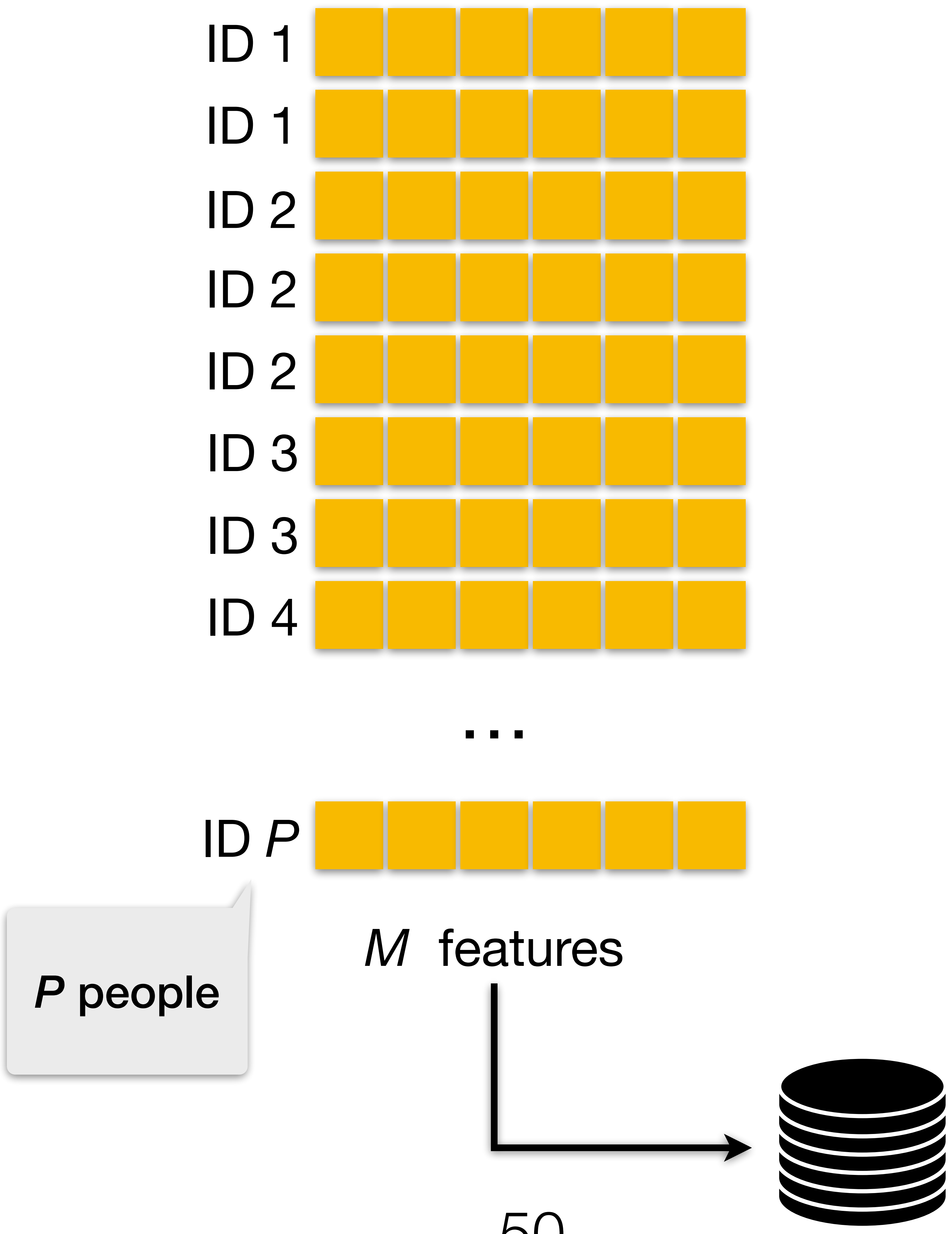
Product Quantization

How to reduce size?

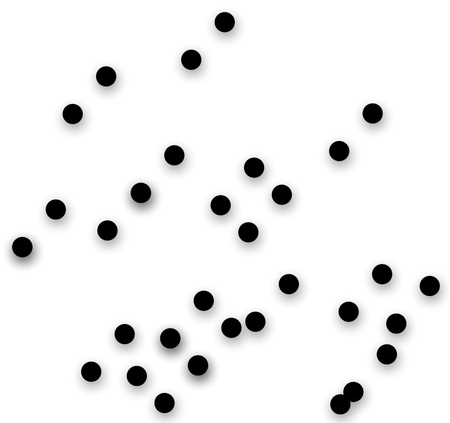
State-of-the-art feature indexing.

1. Start with a **coarse quantizer**.

Toy Case (6D features, reality: 512D for faces)



coarse quantizer



LOYOLA UNIVERSITY CHICAGO

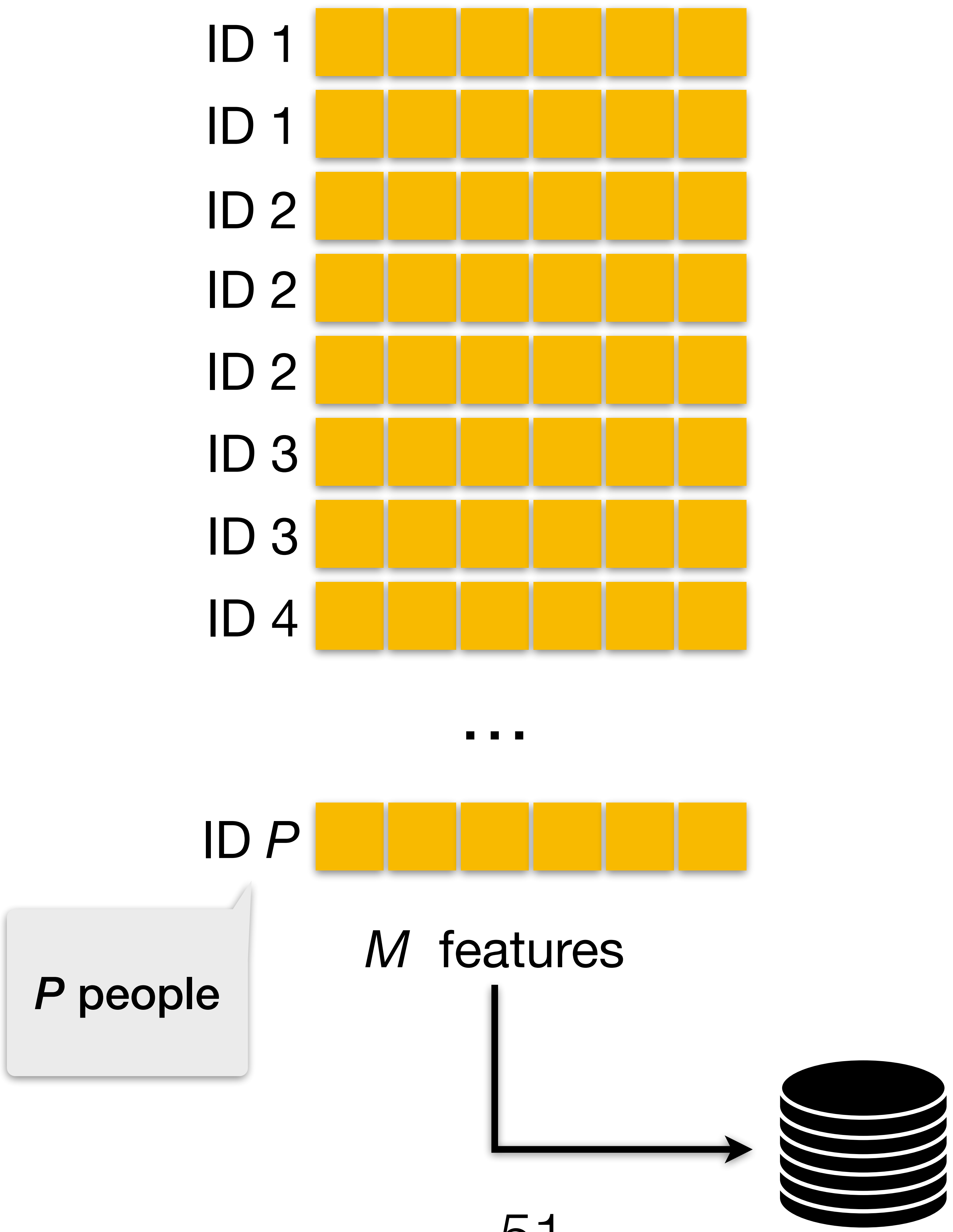
Product Quantization

How to reduce size?

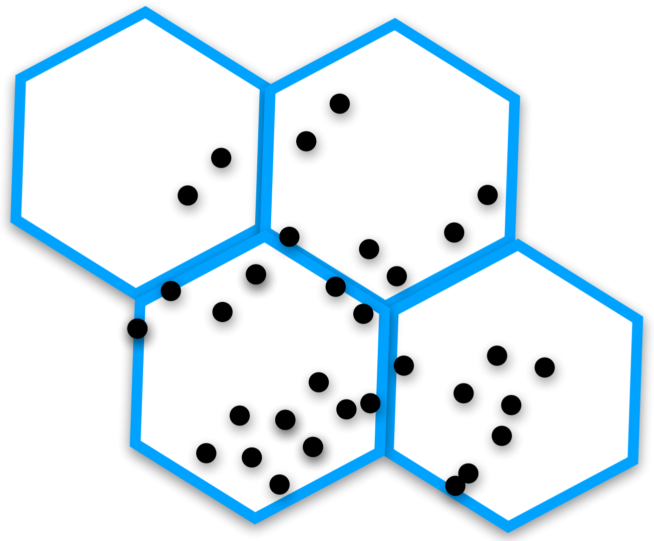
State-of-the-art feature indexing.

1. Start with a **coarse quantizer**.

Toy Case (6D features, reality: 512D for faces)



coarse quantizer

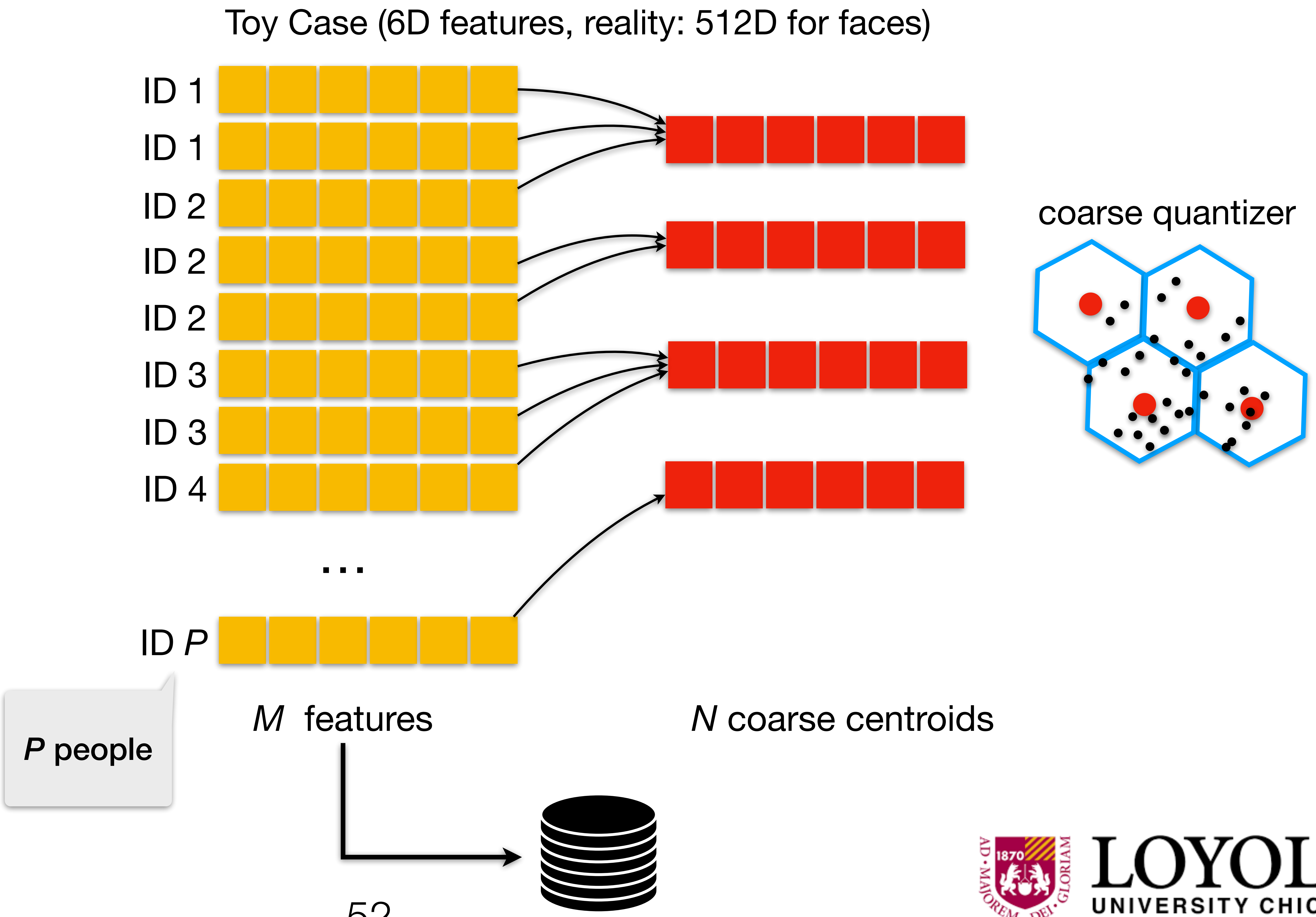


Product Quantization

How to reduce size?

State-of-the-art feature indexing.

1. Start with a **coarse quantizer**.

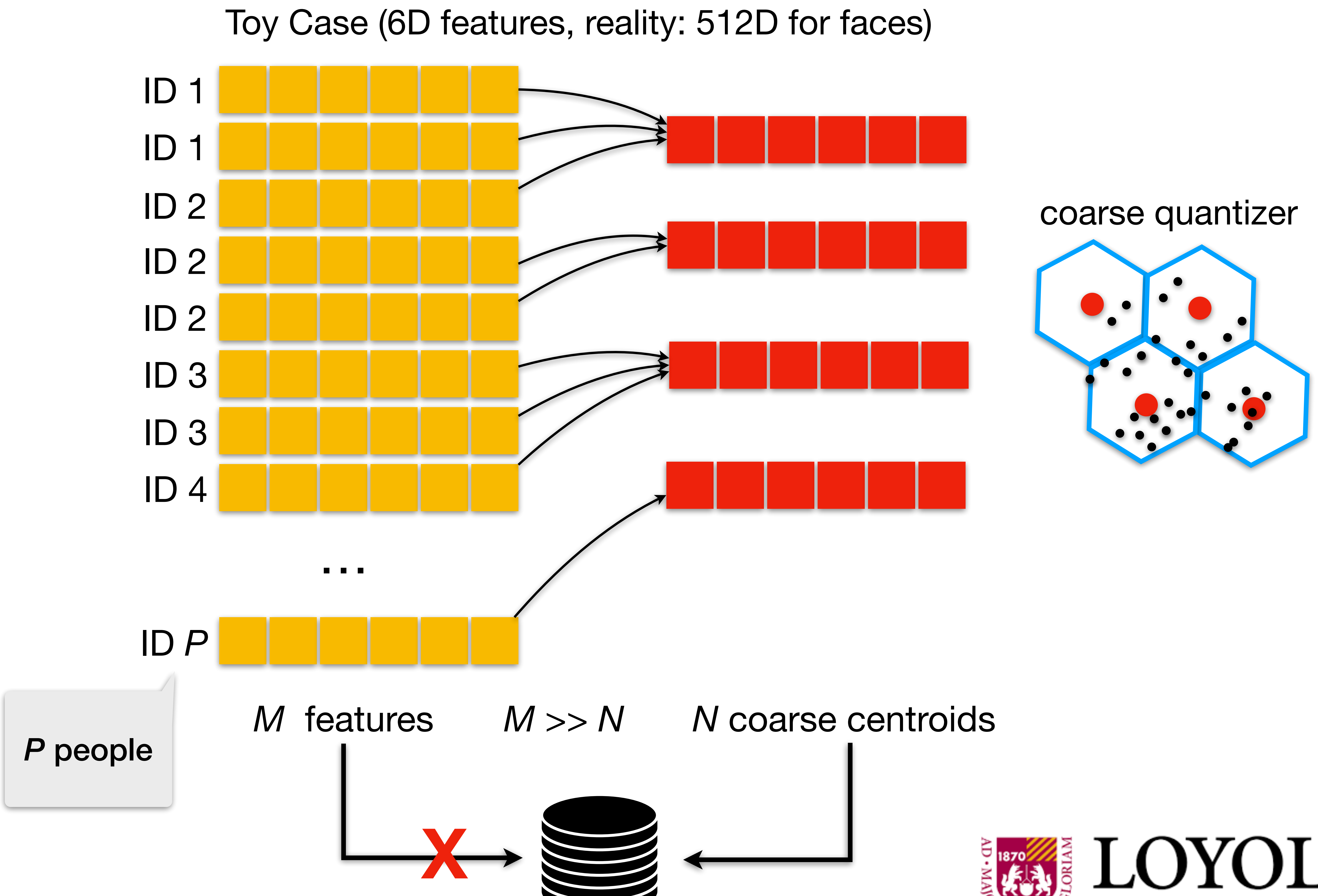


Product Quantization

How to reduce size?

State-of-the-art feature indexing.

1. Start with a **coarse quantizer**.

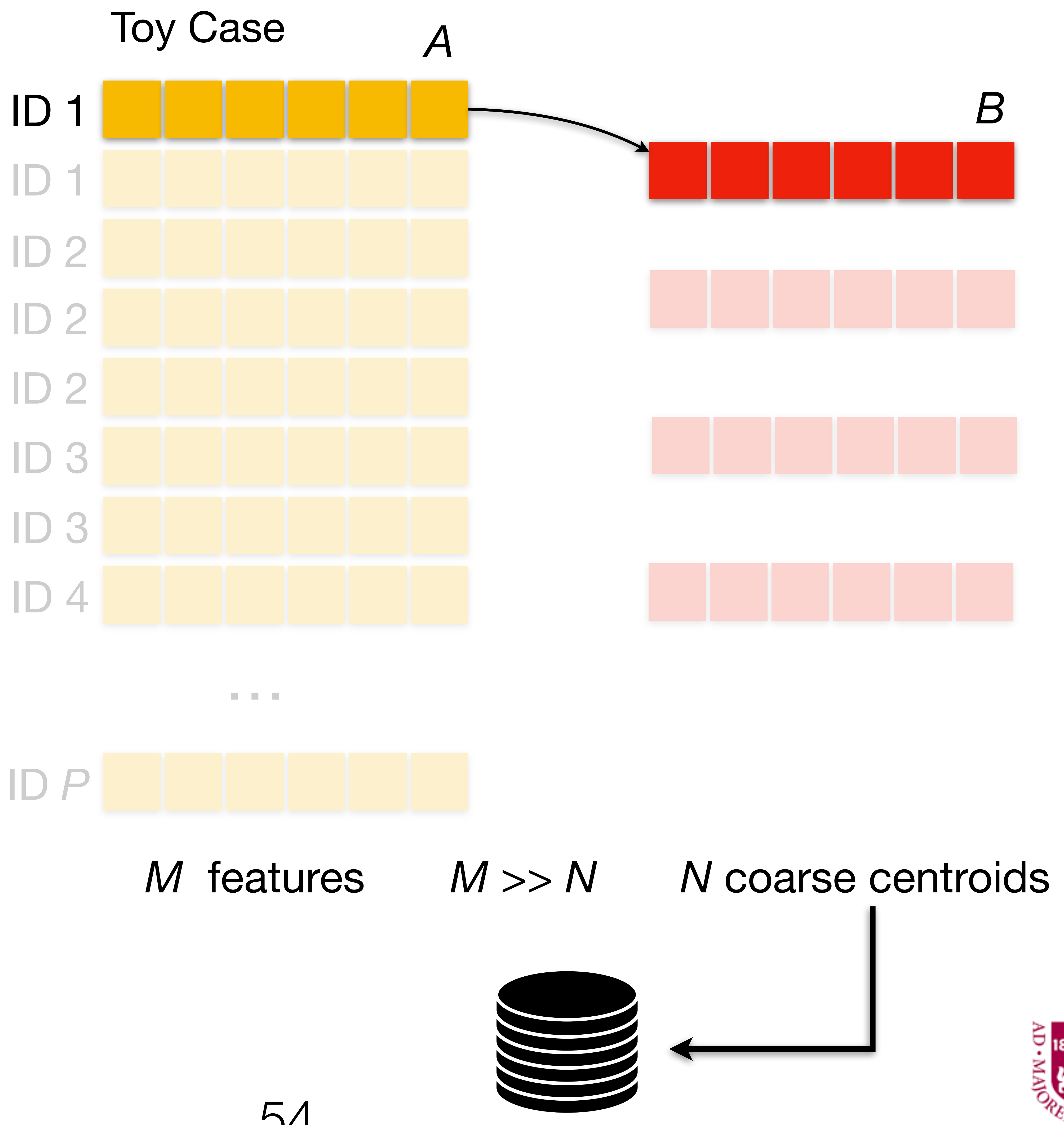


Product Quantization

How to reduce size?

State-of-the-art feature indexing.

2. Compute **residuals** (differences) between features and their respective coarse centroids.

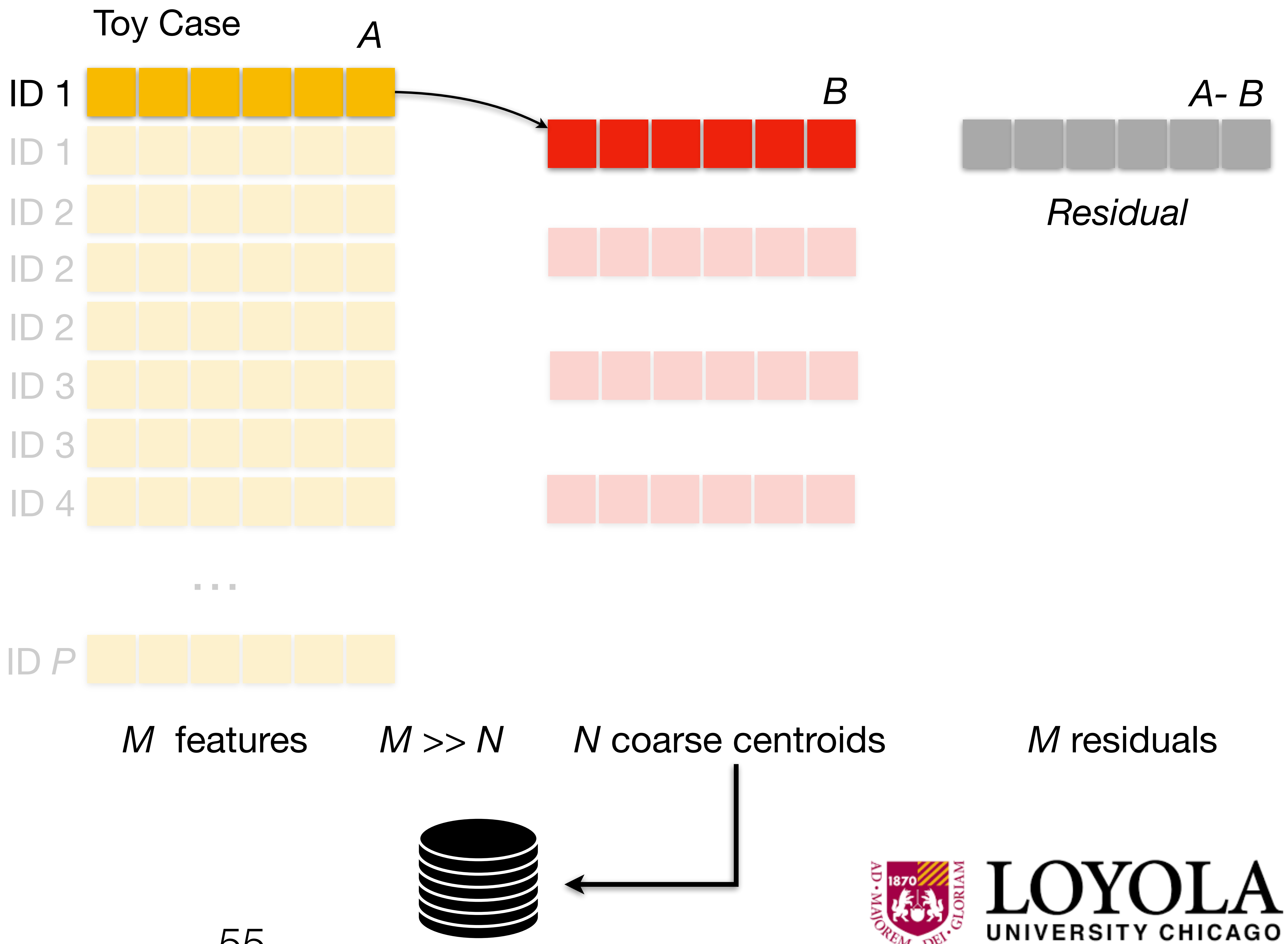


Product Quantization

How to reduce size?

State-of-the-art feature indexing.

2. Compute **residuals** (differences) between features and their respective coarse centroids.



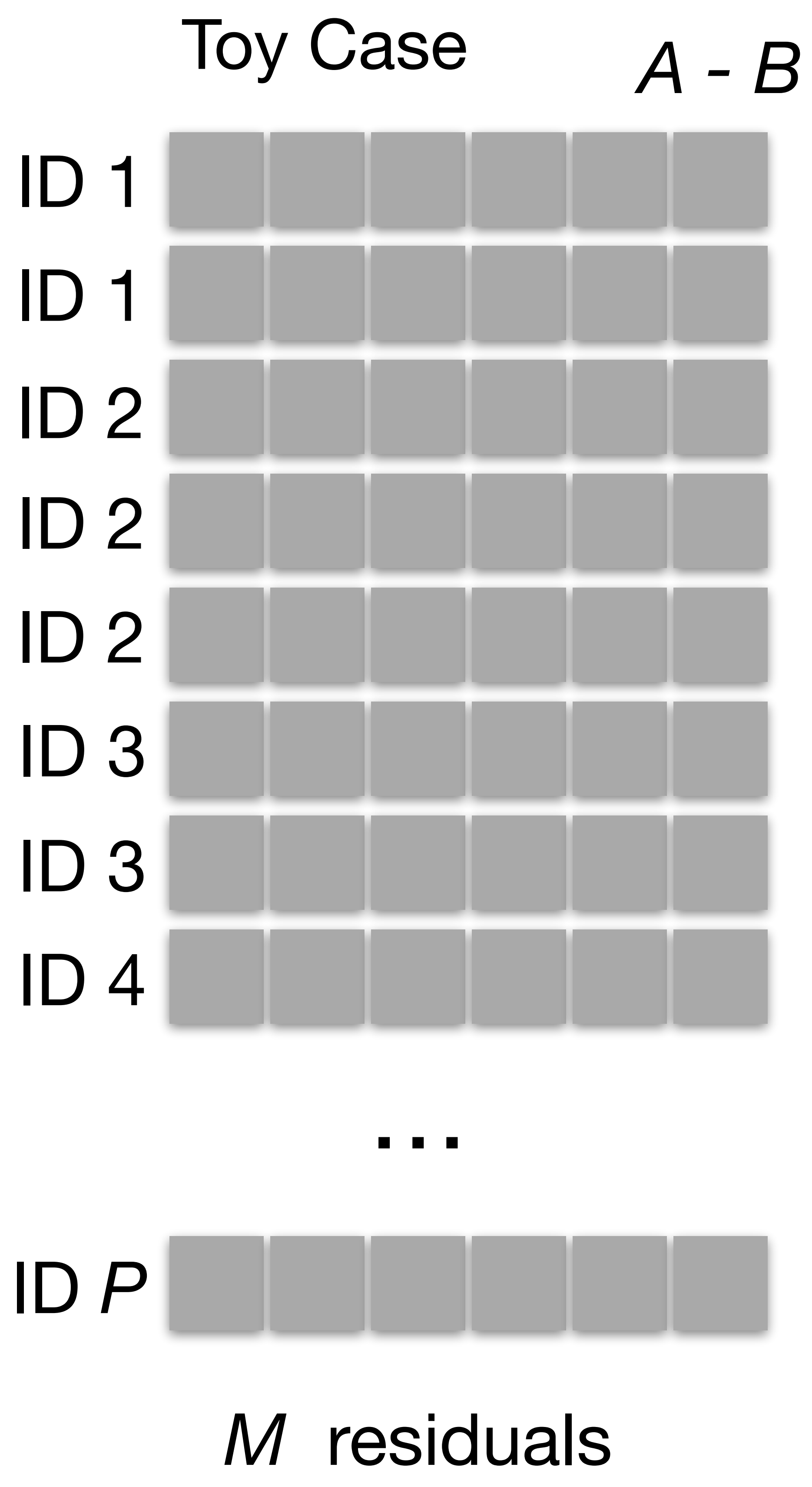
LOYOLA UNIVERSITY CHICAGO

Product Quantization

How to reduce size?

State-of-the-art feature indexing.

3. Reduce the dimensionality of residuals with **Product Quantization**.

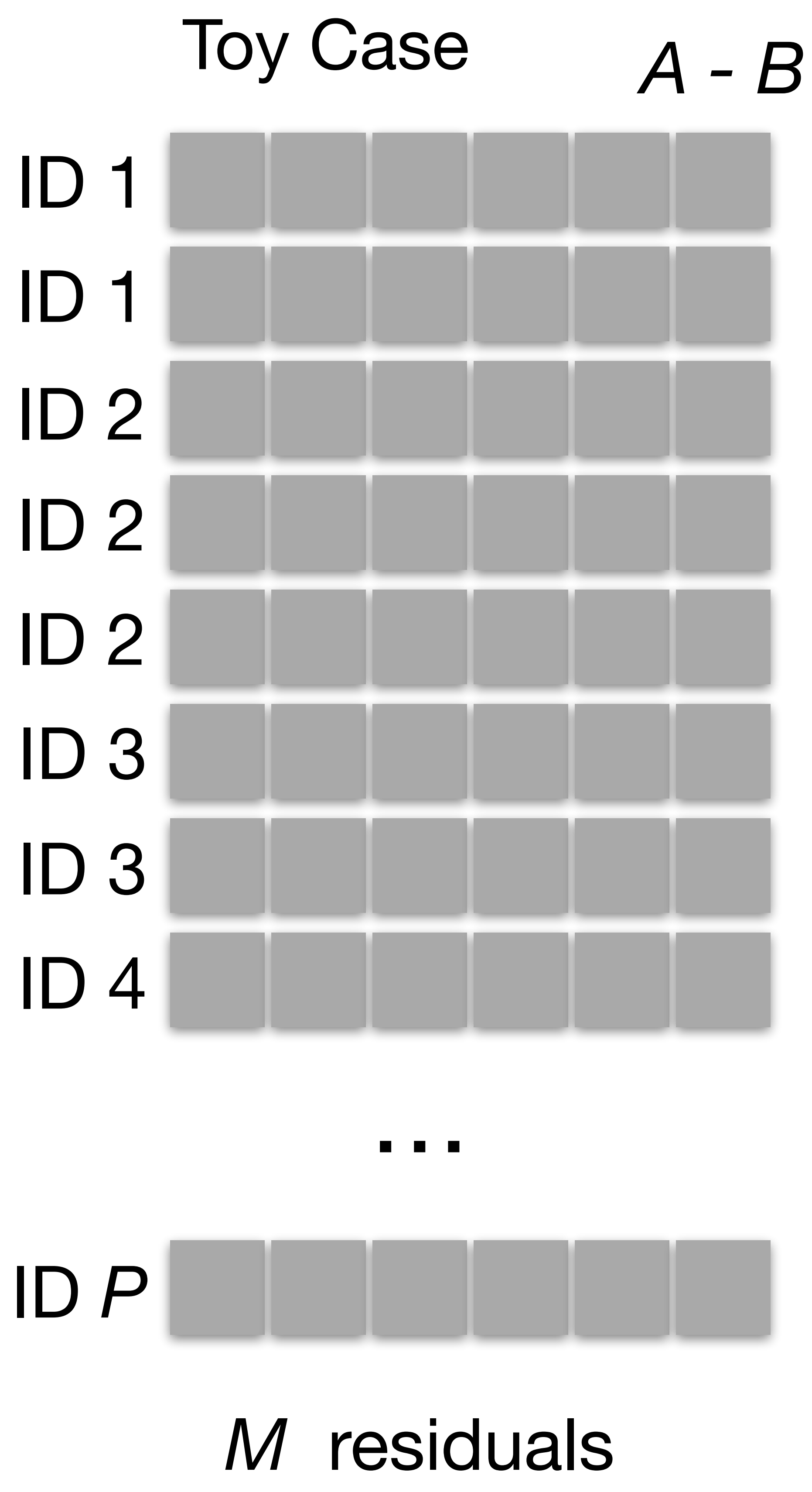


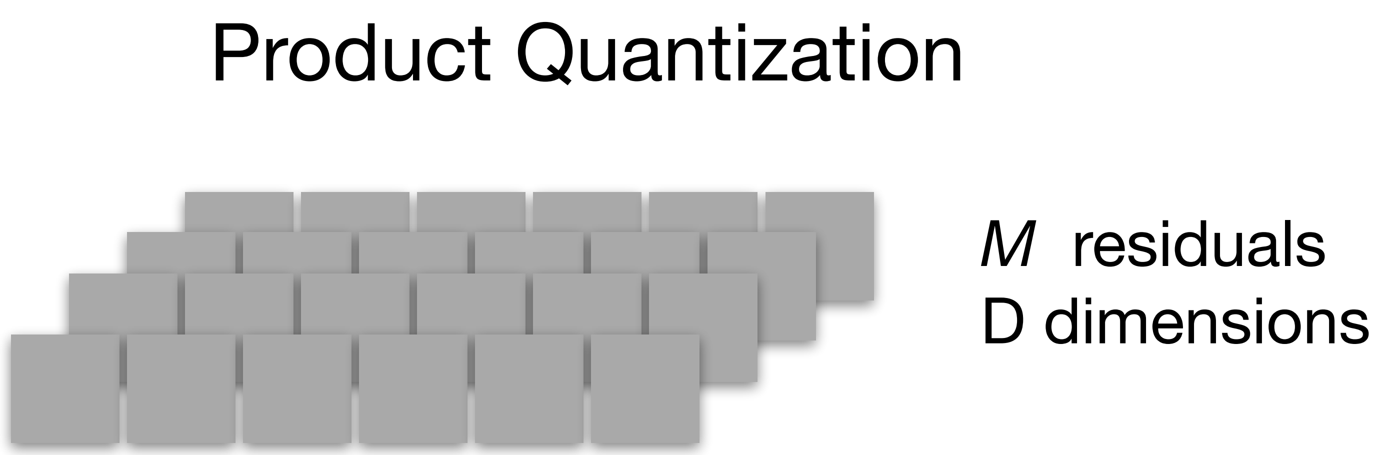
Product Quantization

How to reduce size?

State-of-the-art feature indexing.

3. Reduce the dimensionality of residuals with **Product Quantization**.





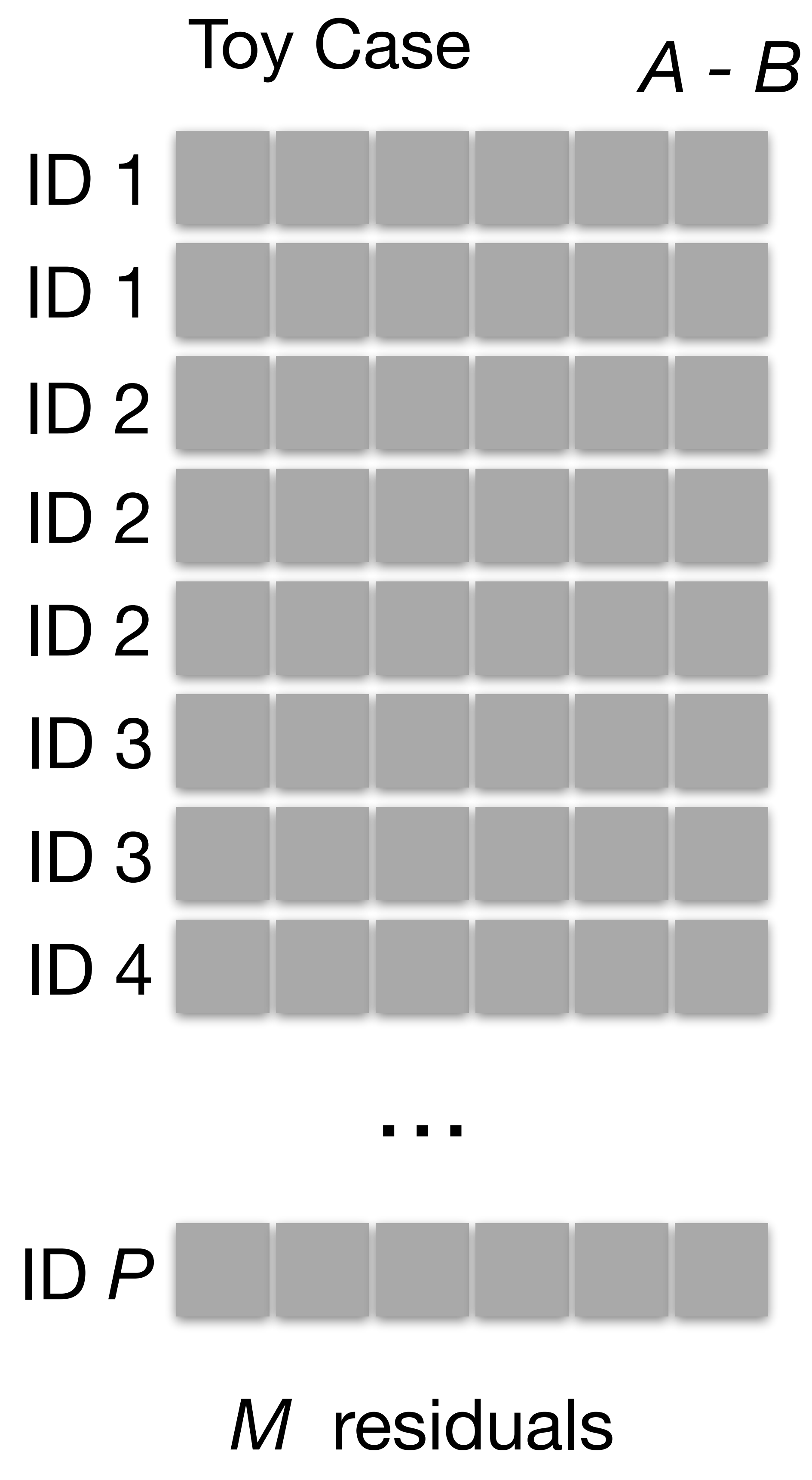
LOYOLA UNIVERSITY CHICAGO

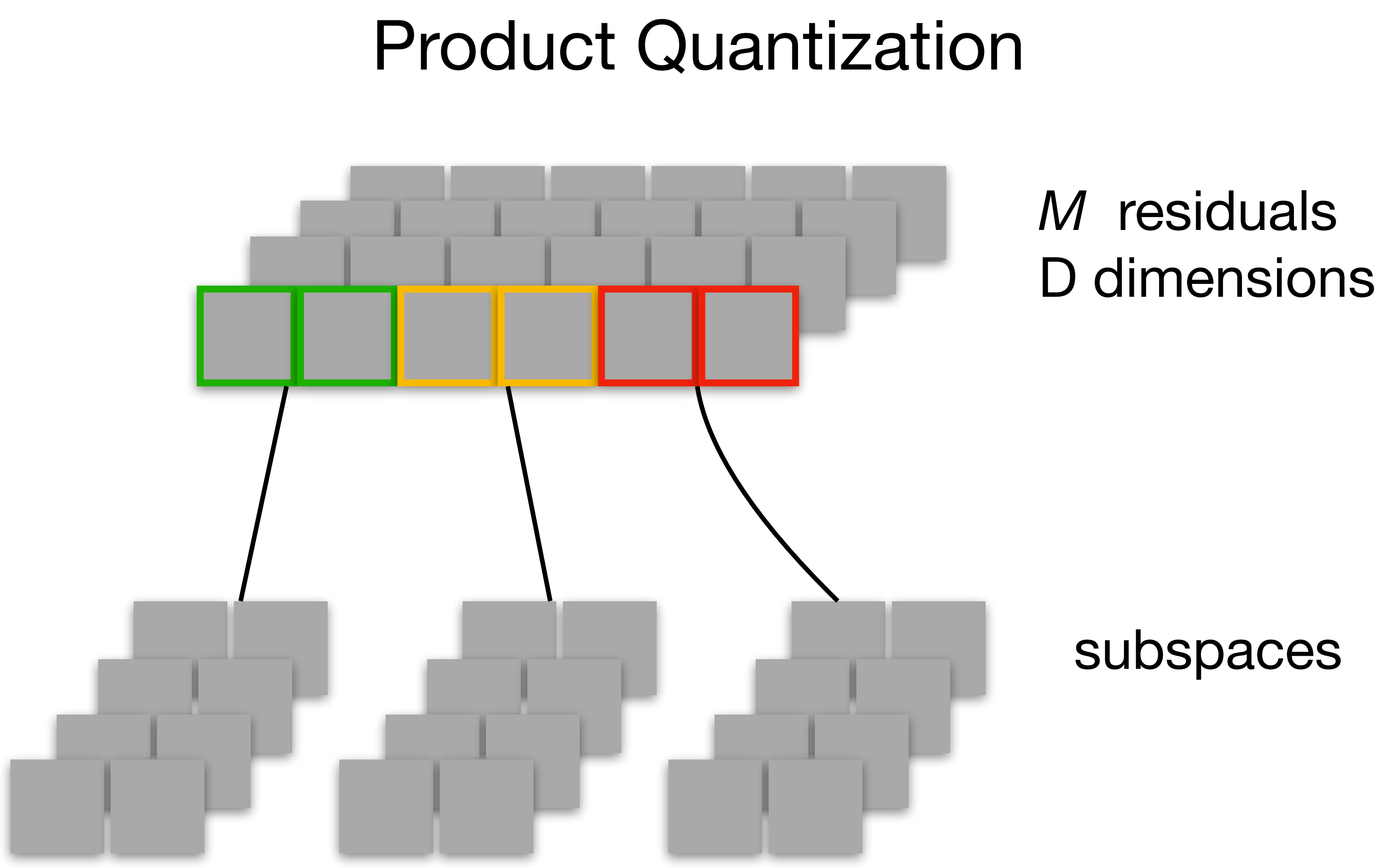
Product Quantization

How to reduce size?

State-of-the-art feature indexing.

3. Reduce the dimensionality of residuals with **Product Quantization**.





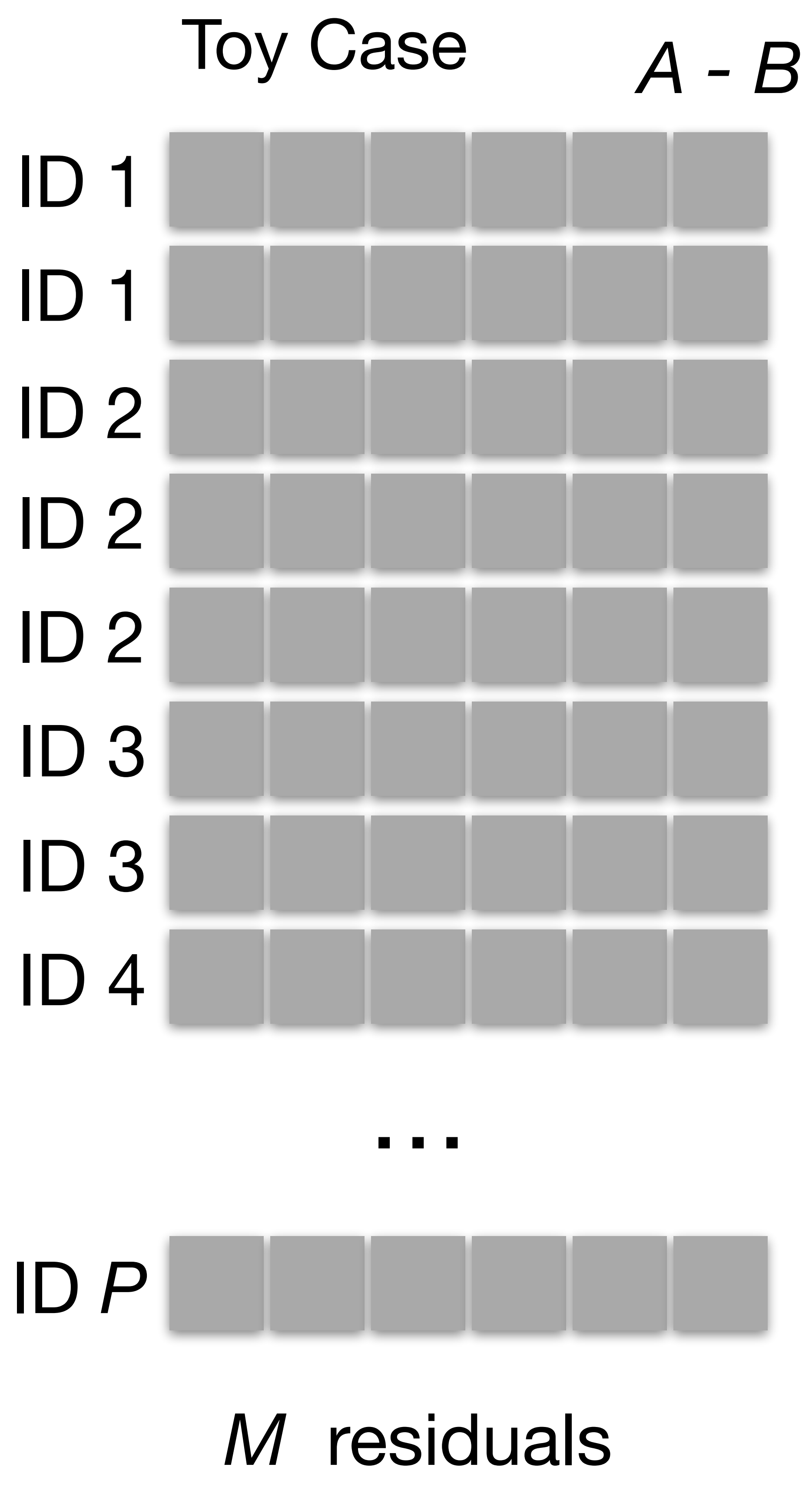
LOYOLA UNIVERSITY CHICAGO

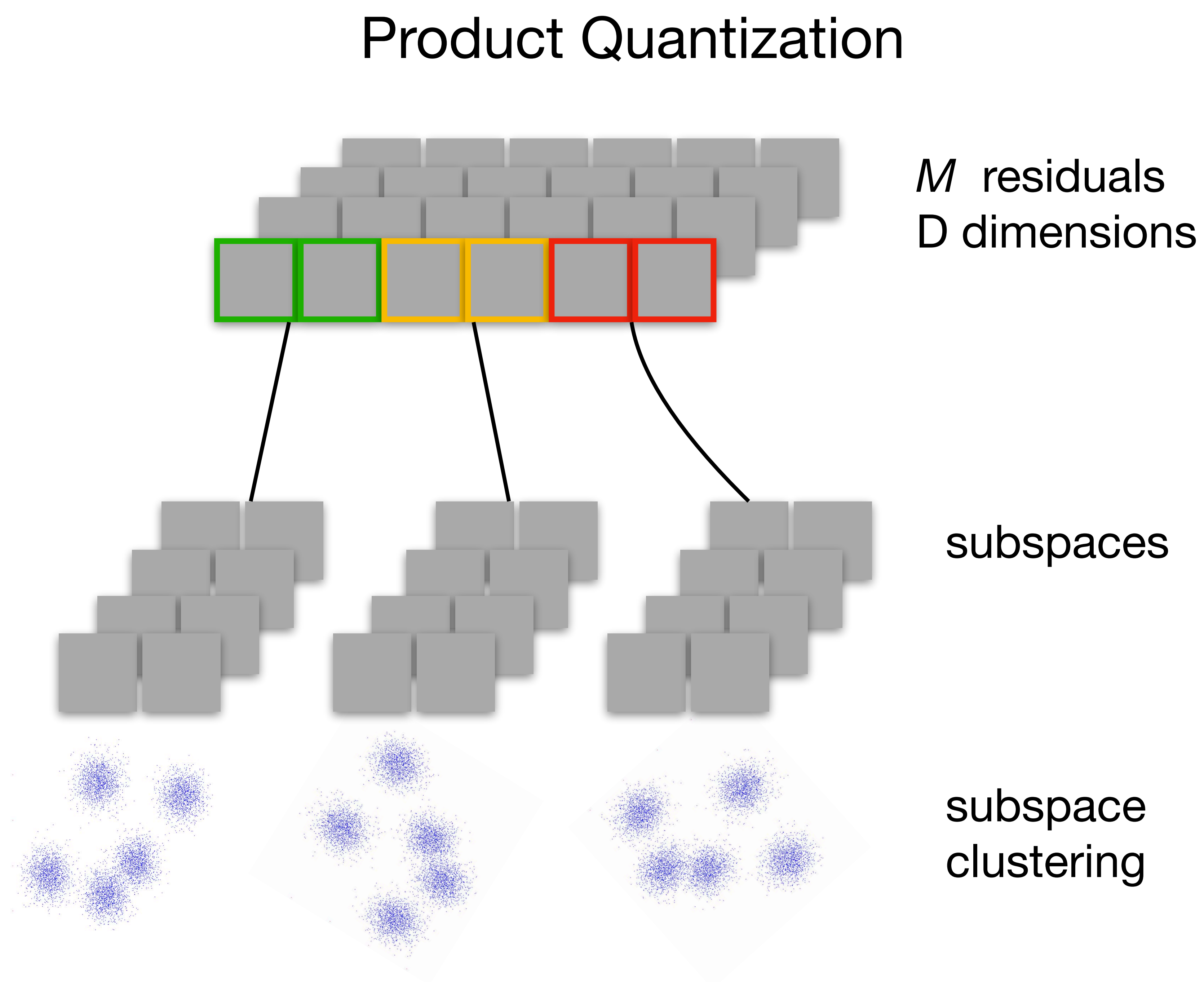
Product Quantization

How to reduce size?

State-of-the-art feature indexing.

3. Reduce the dimensionality of residuals with **Product Quantization**.





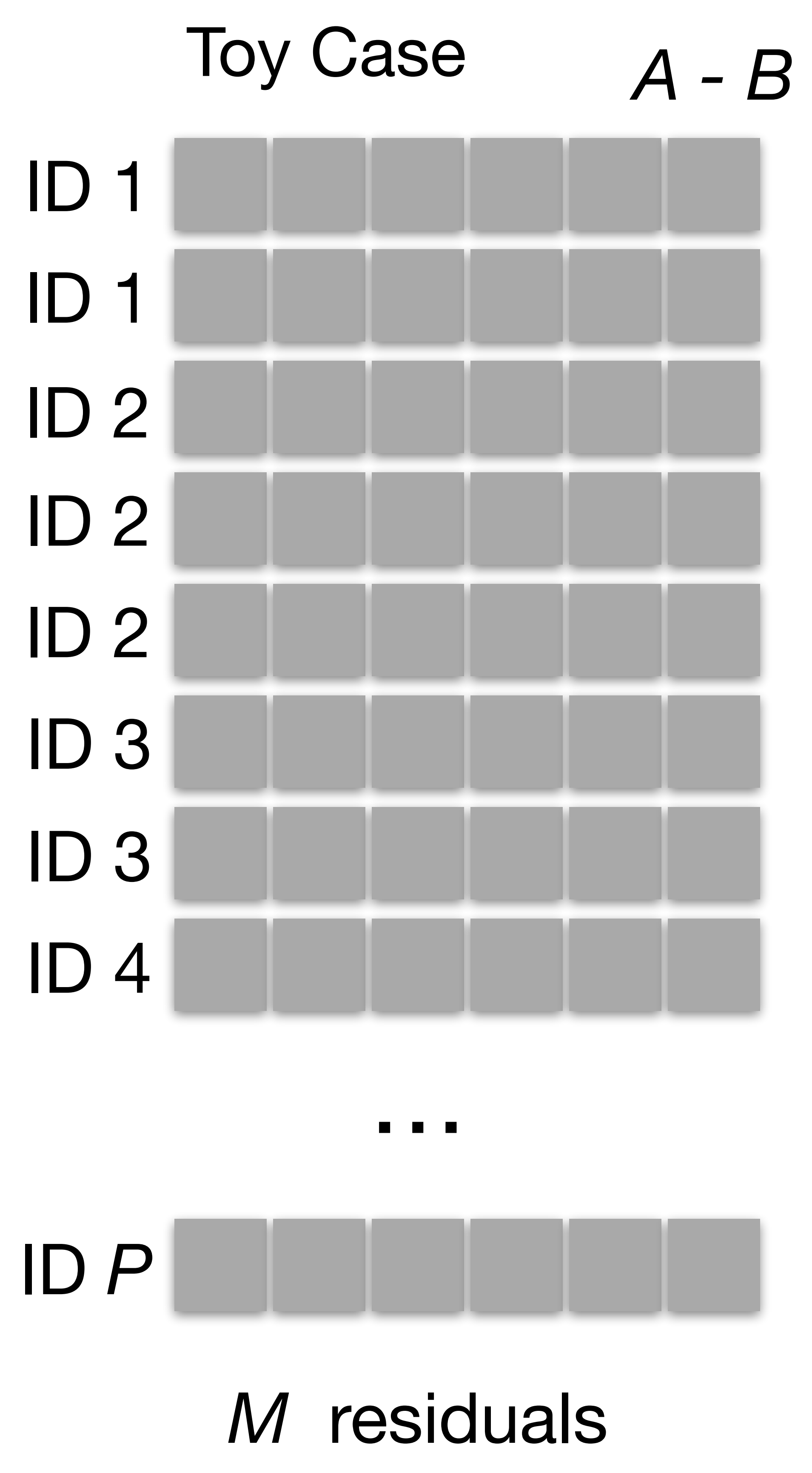
LOYOLA UNIVERSITY CHICAGO

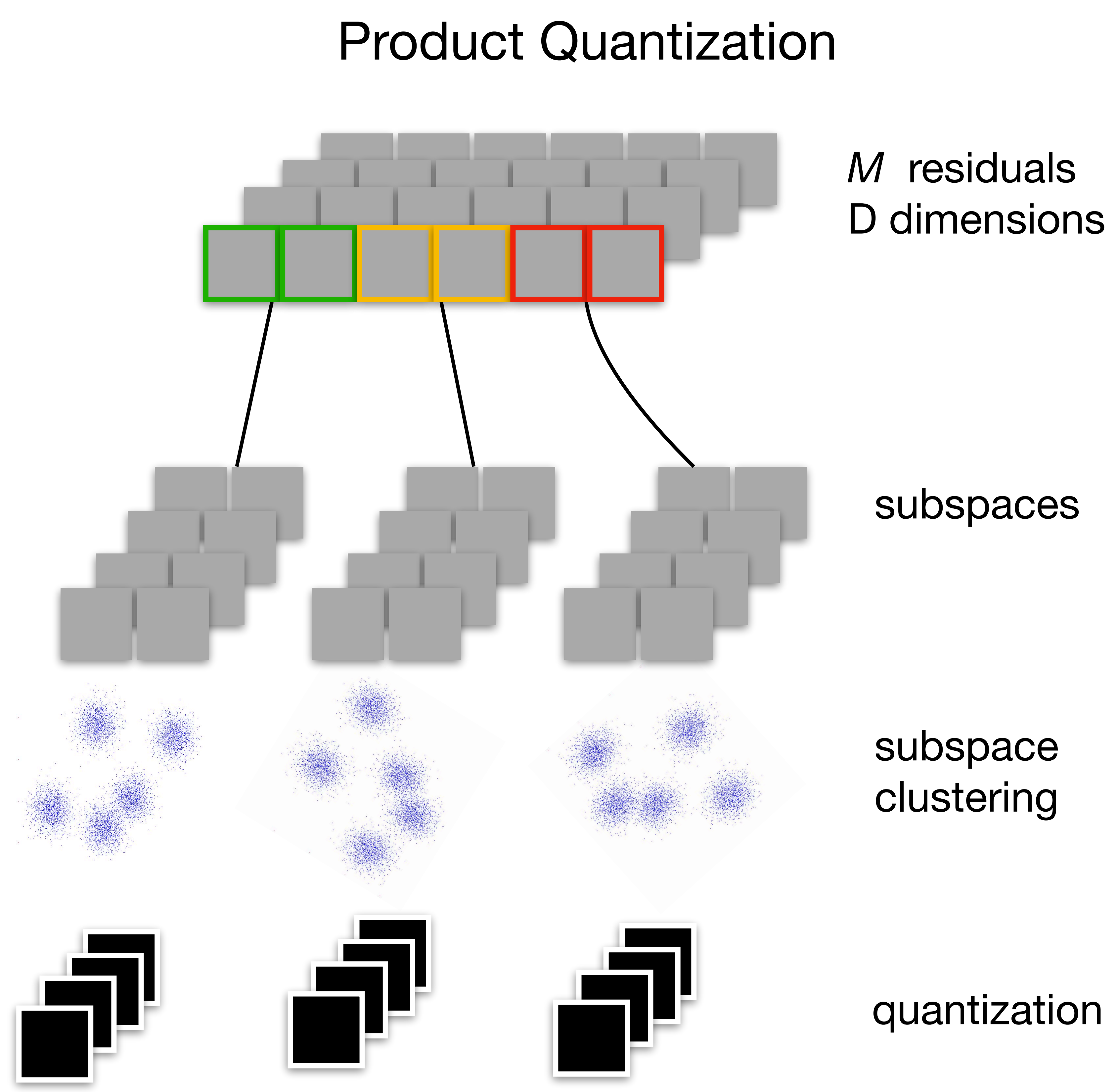
Product Quantization

How to reduce size?

State-of-the-art feature indexing.

3. Reduce the dimensionality of residuals with **Product Quantization**.



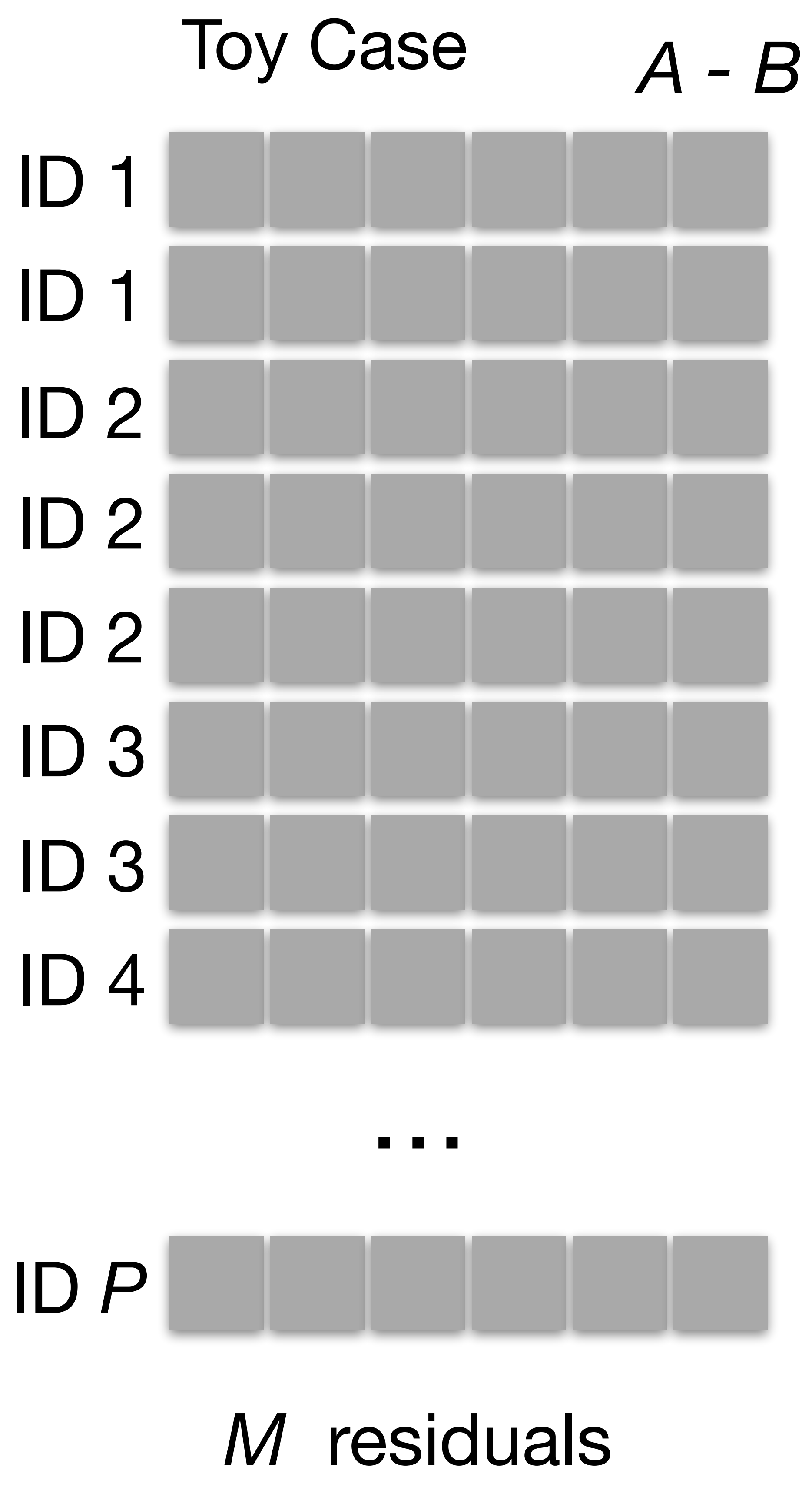


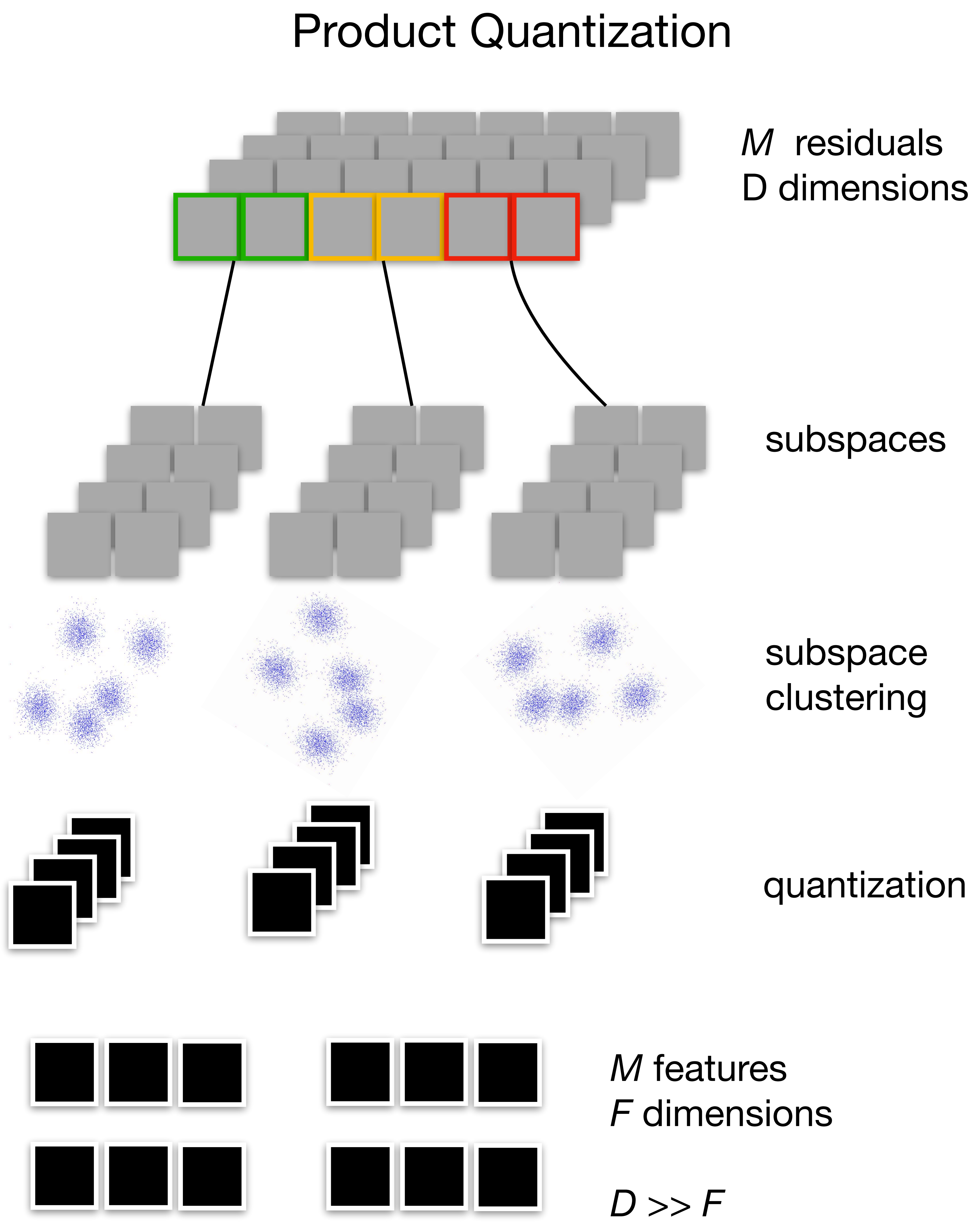
Product Quantization

How to reduce size?

State-of-the-art feature indexing.

3. Reduce the dimensionality of residuals with **Product Quantization**.



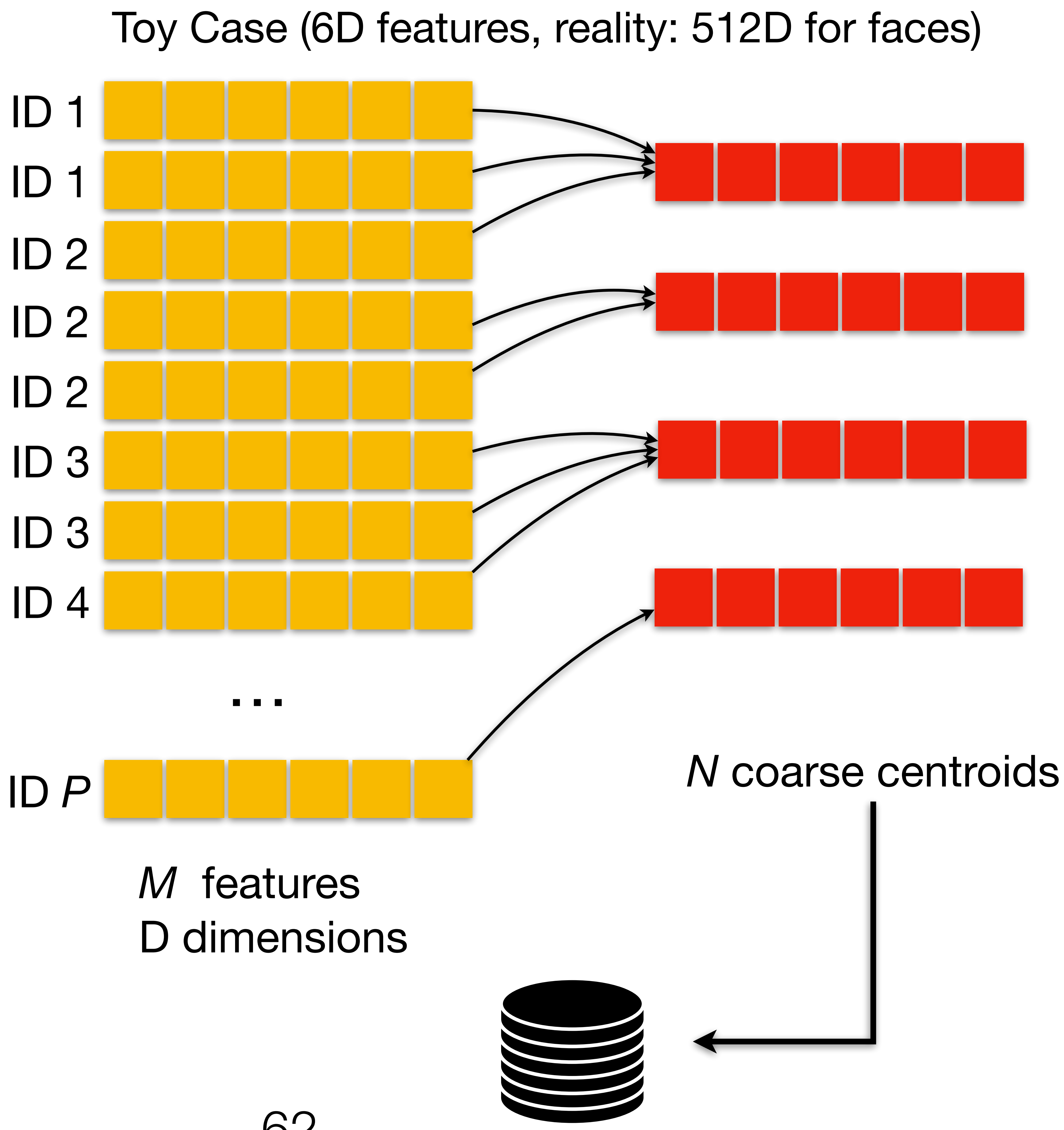


Product Quantization

How to reduce size?

State-of-the-art feature indexing.

4. Append the product quantized residuals to an **inverted file index.**



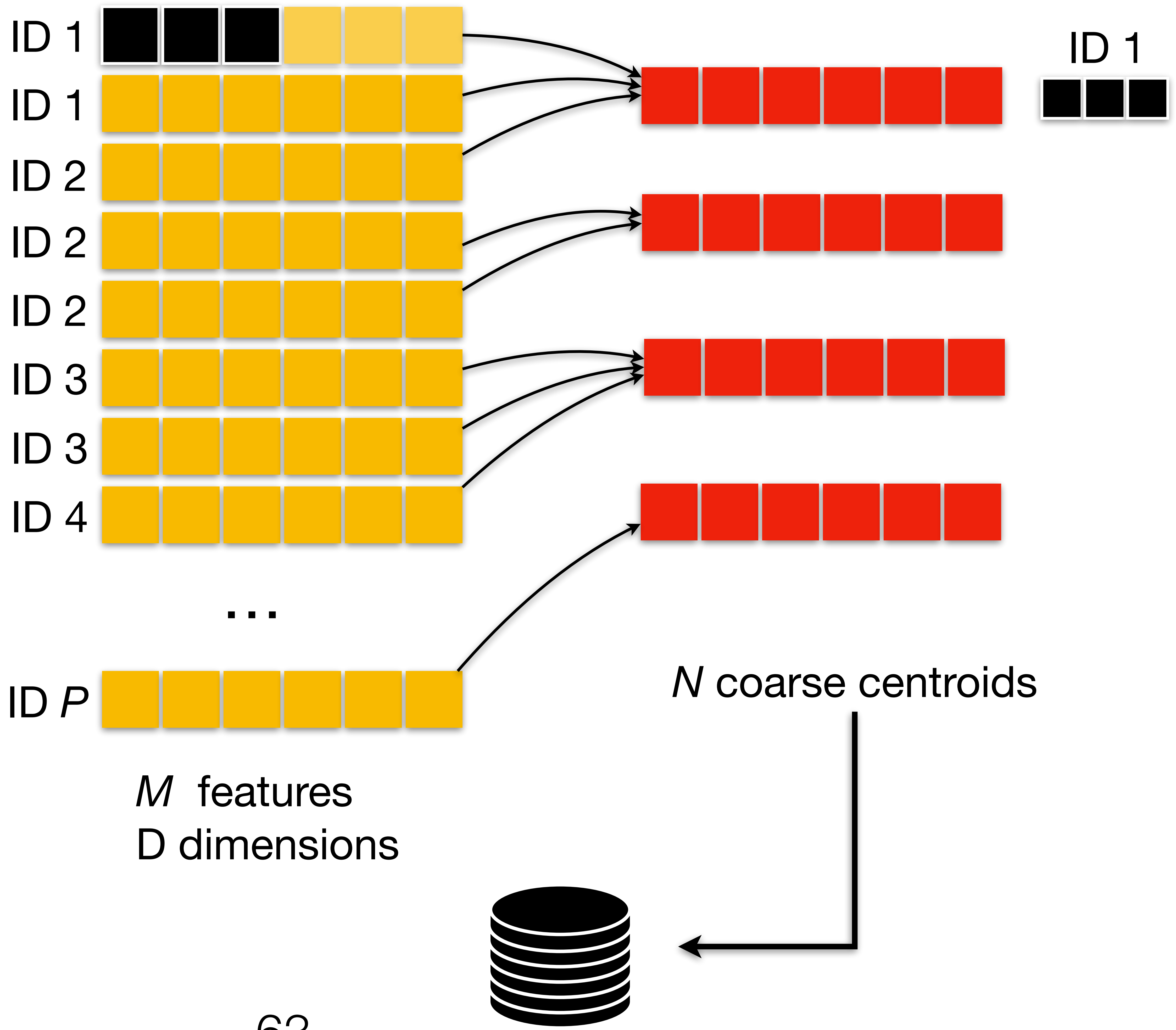
Product Quantization

How to reduce size?

State-of-the-art feature indexing.

4. Append the product quantized residuals to an **inverted file index**.

Toy Case (6D features, reality: 512D for faces)

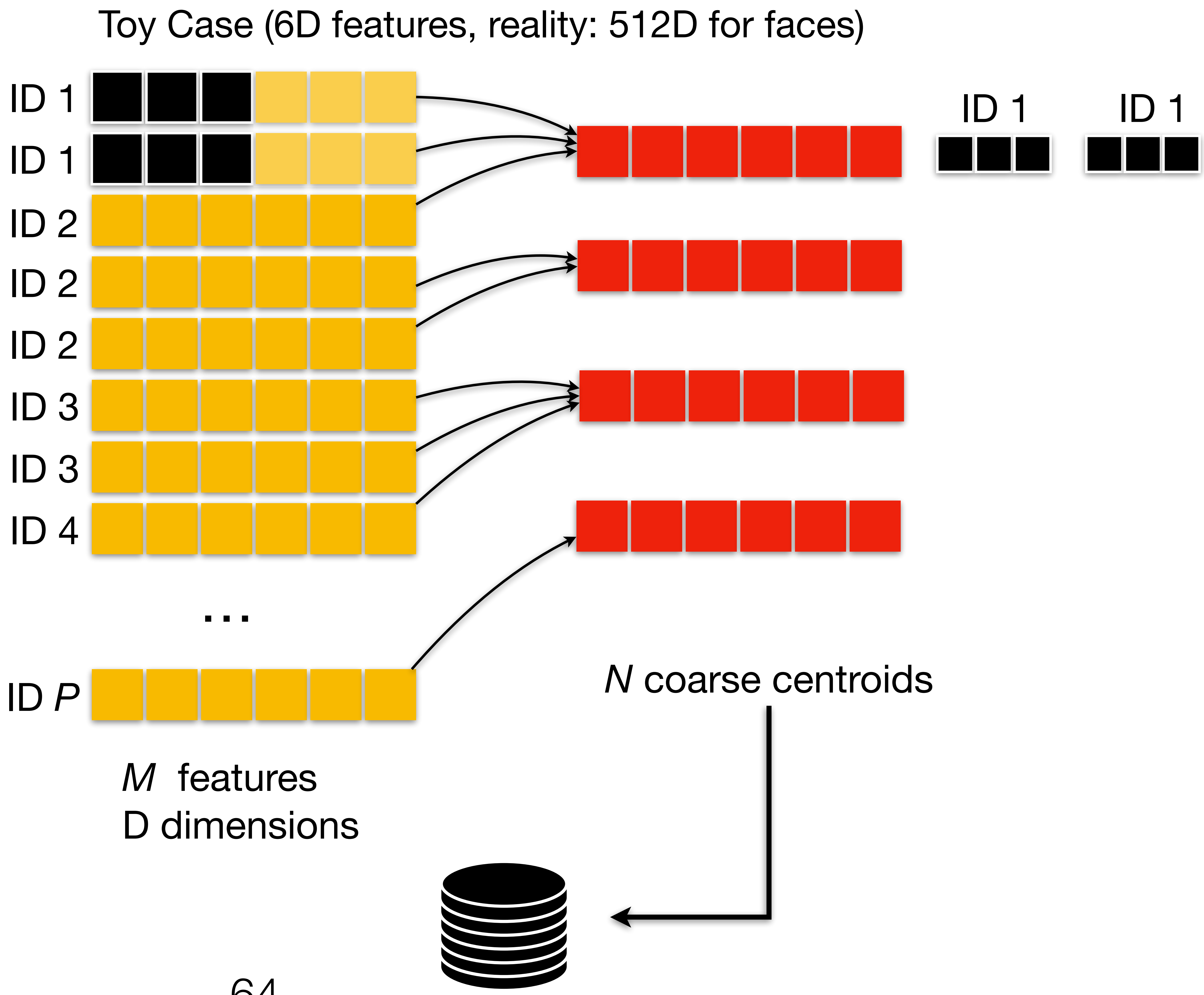


Product Quantization

How to reduce size?

State-of-the-art feature indexing.

4. Append the product quantized residuals to an **inverted file index.**

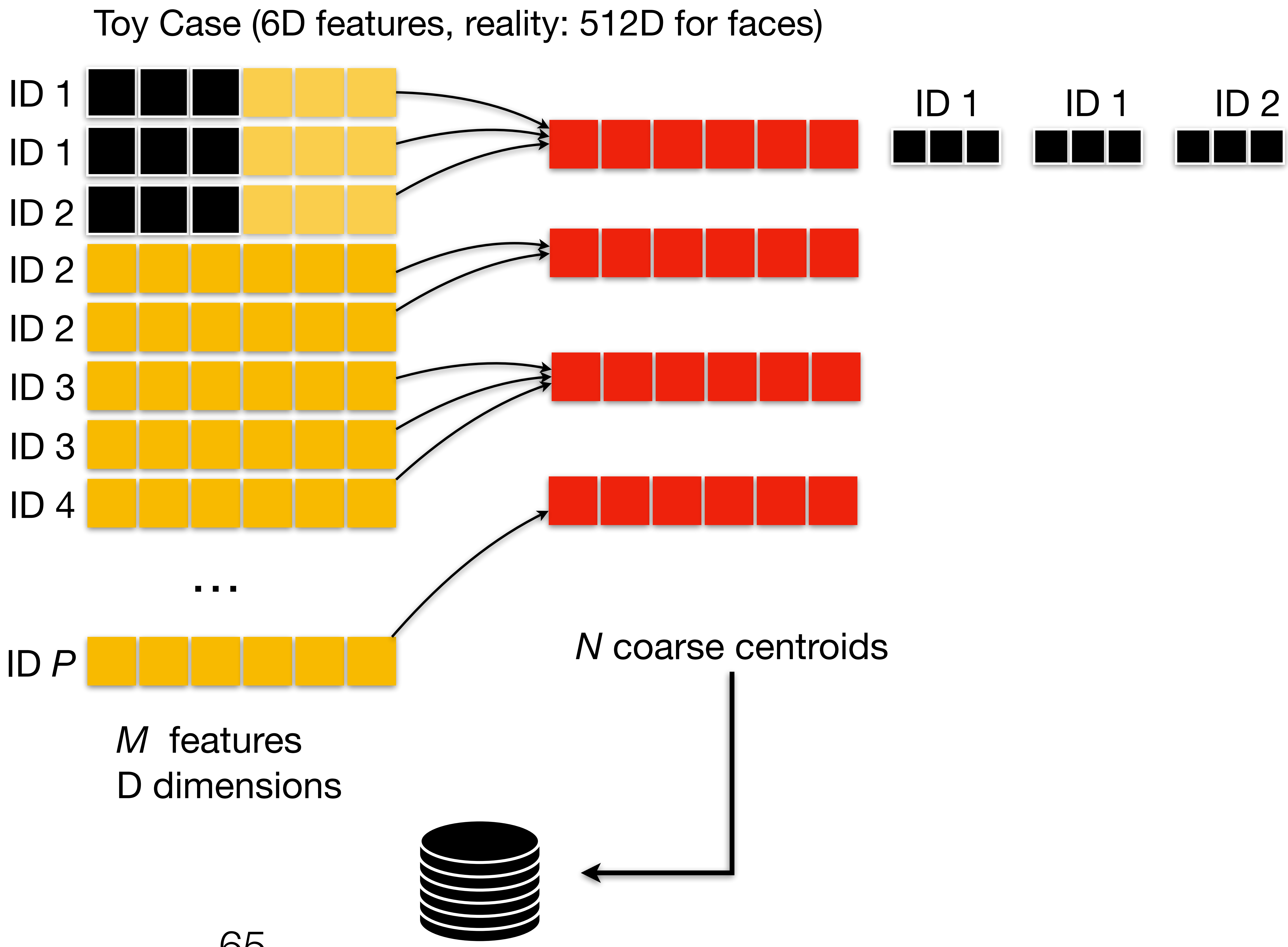


Product Quantization

How to reduce size?

State-of-the-art feature indexing.

4. Append the product quantized residuals to an **inverted file index.**

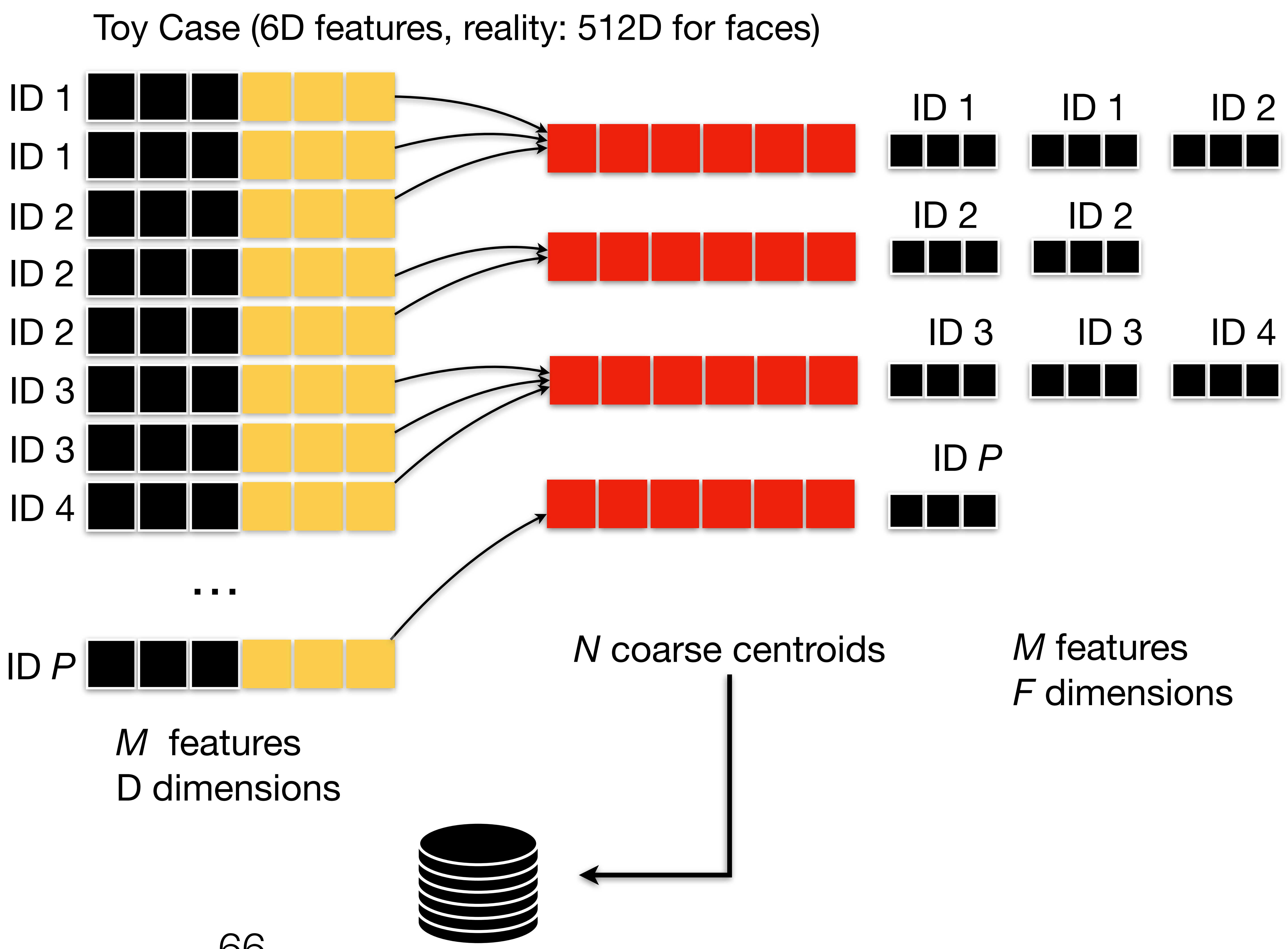


Product Quantization

How to reduce size?

State-of-the-art feature indexing.

4. Append the product quantized residuals to an **inverted file index**.

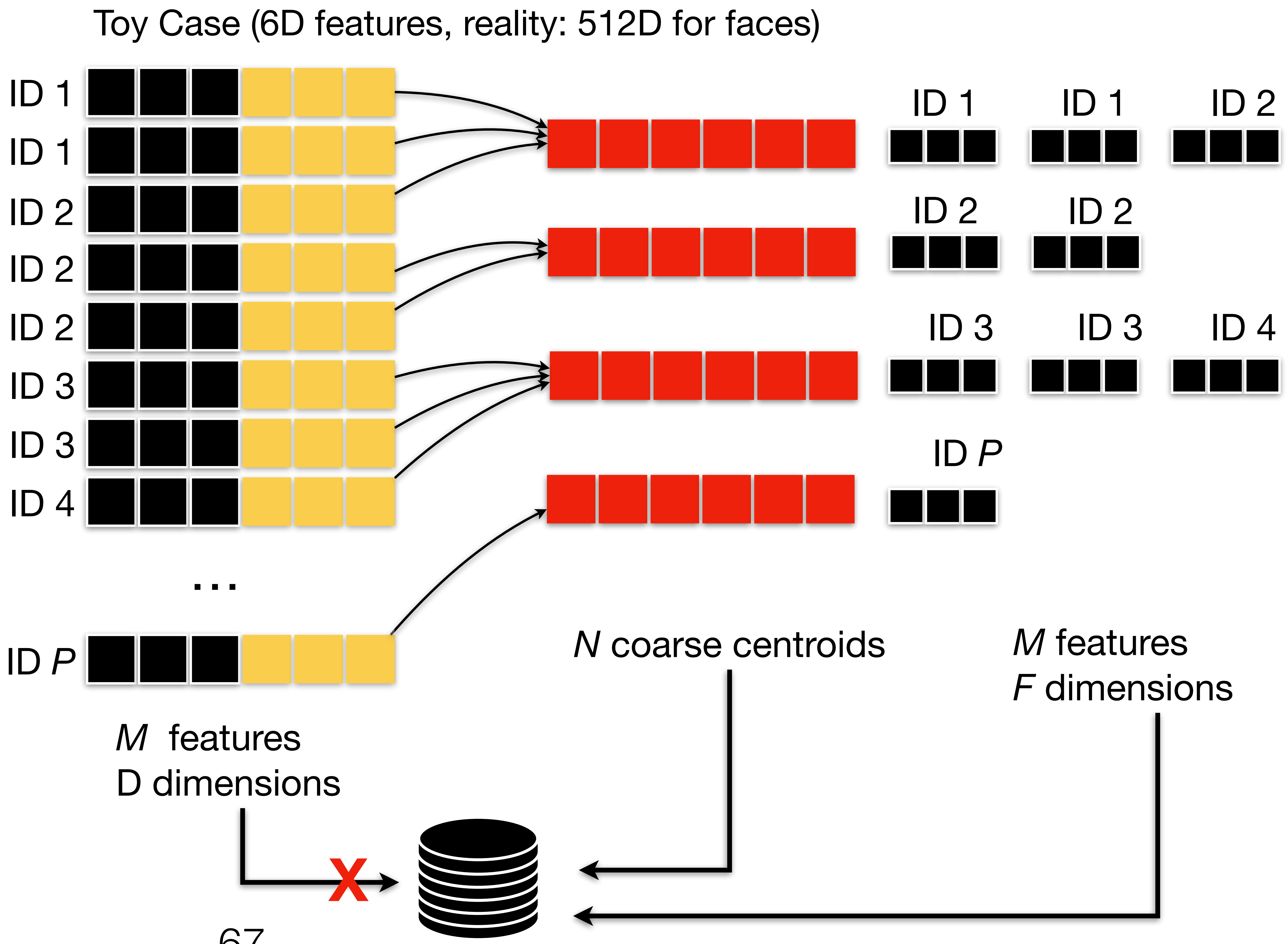


Product Quantization

How to reduce size?

State-of-the-art feature indexing.

4. Append the product quantized residuals to an **inverted file index**.



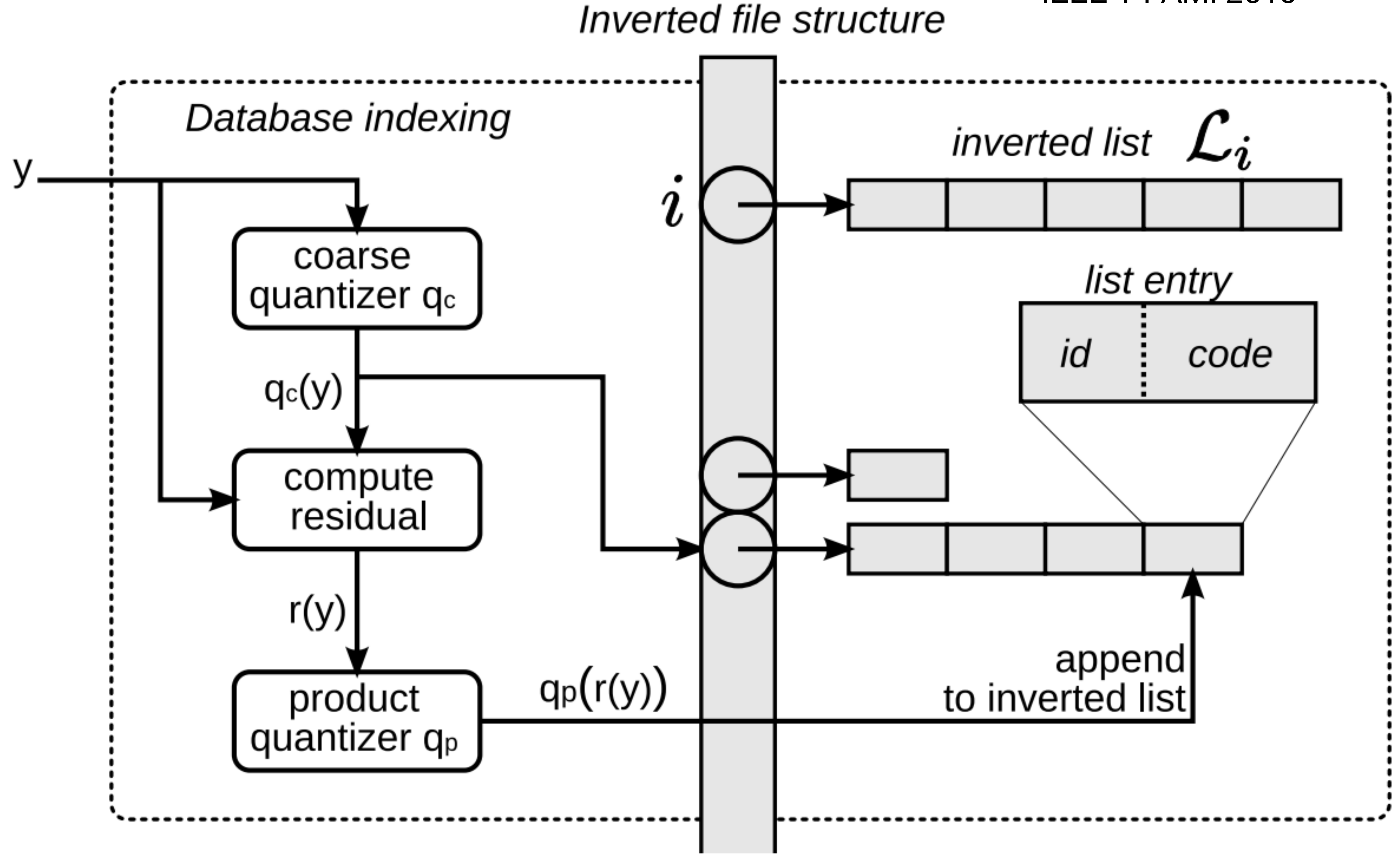
Product Quantization

Source: Jegou et al.
Product quantization for nearest neighbor search
IEEE T-PAMI 2010

How to reduce size?

State-of-the-art feature indexing.

Usage example:
Indexing.



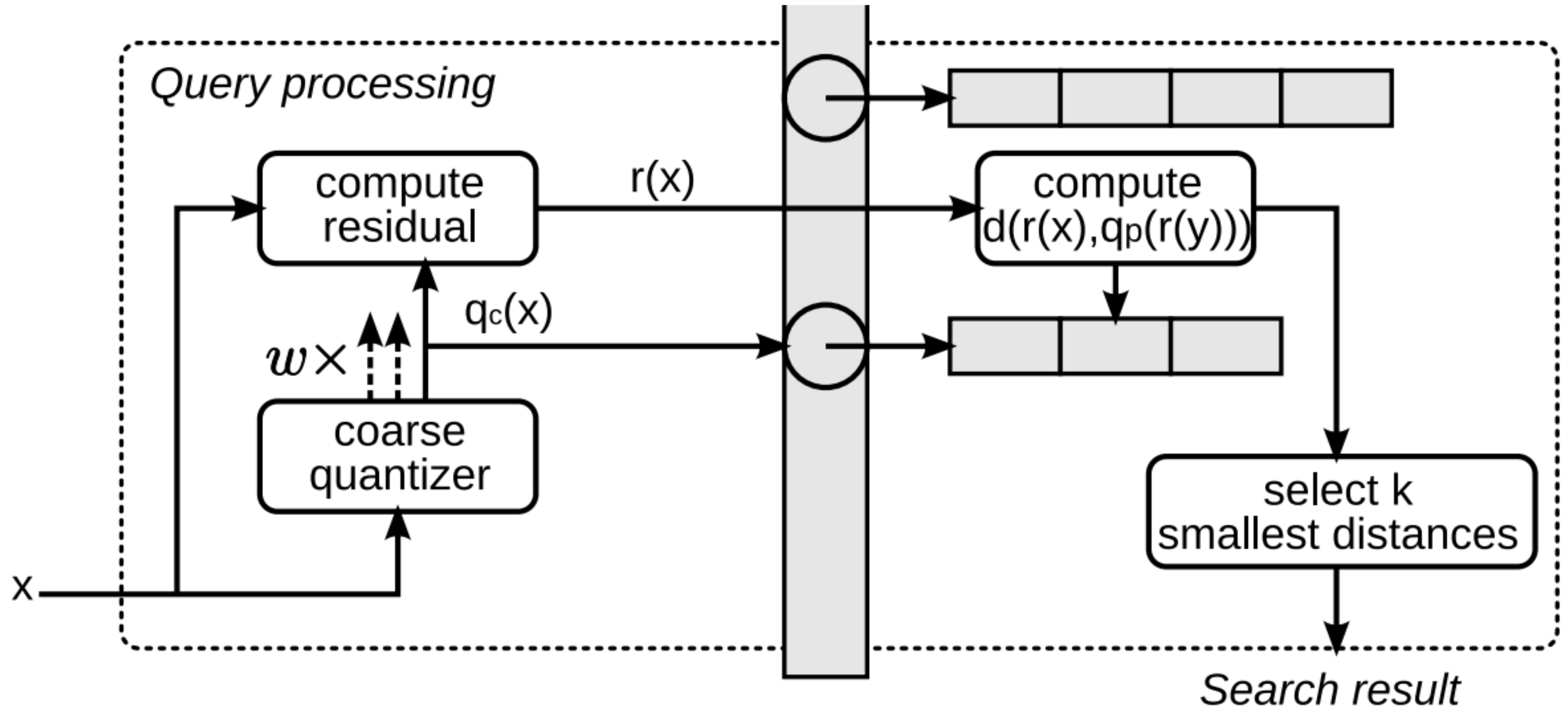
Product Quantization

How to reduce
size?

State-of-the-art feature
indexing.

Usage example:
Retrieving k-nearest.

Source: Jegou et al.
Product quantization for nearest neighbor search
IEEE T-PAMI 2010



LOYOLA
UNIVERSITY CHICAGO

Product Quantization

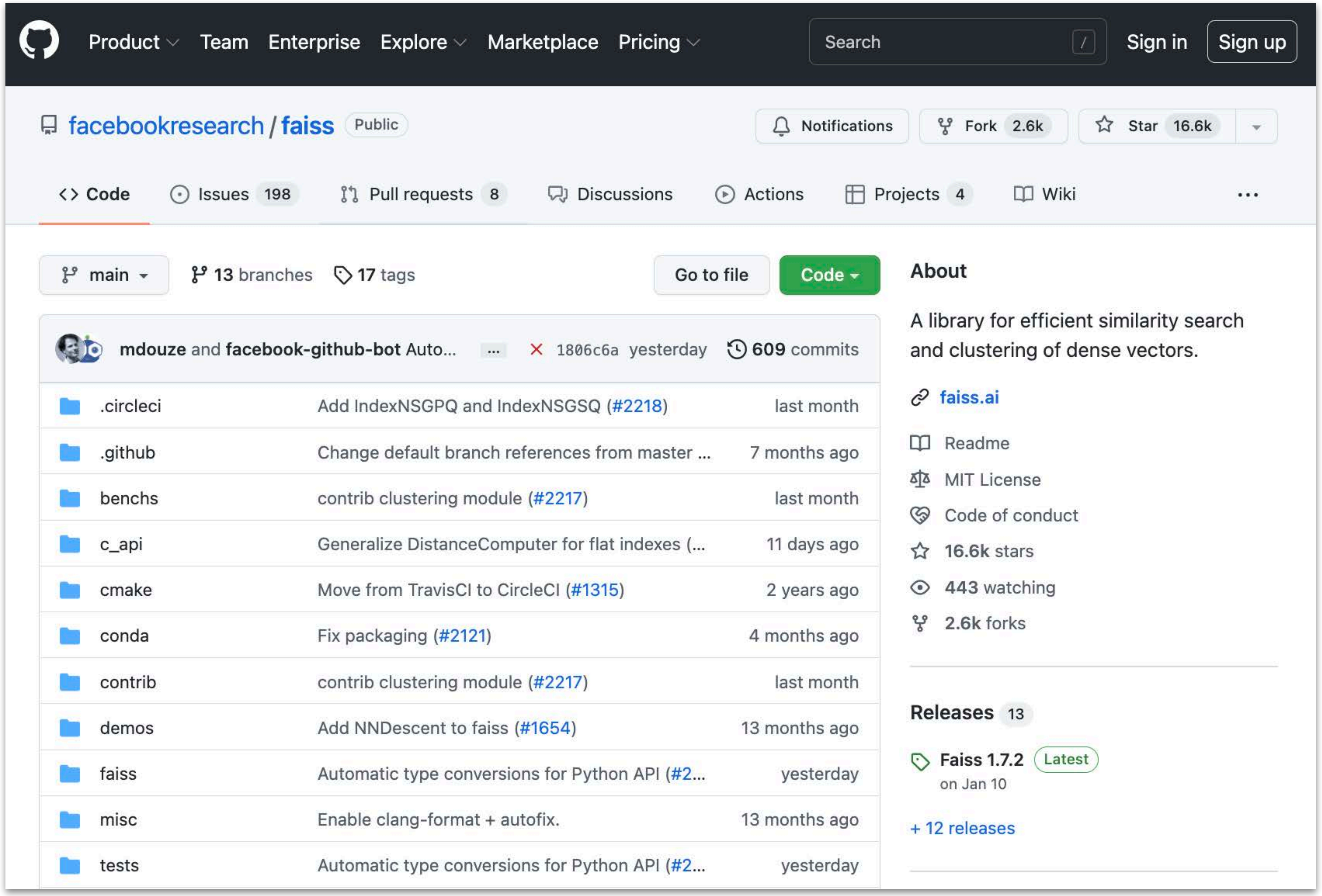
How to reduce size?

State-of-the-art feature indexing.

Available implementation.

Faiss

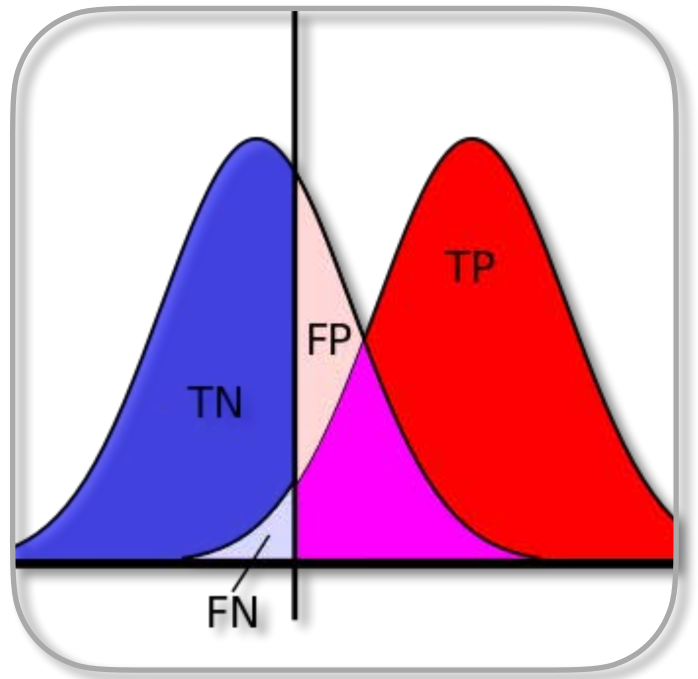
Faiss is a library for efficient similarity search and clustering of dense vectors. It contains algorithms that search in sets of vectors of any size, up to ones that possibly do not fit in RAM. It also contains supporting code for evaluation and parameter tuning. Faiss is written in C++ with complete wrappers for Python/numpy. Some of the most useful algorithms are implemented on the GPU. It is developed primarily at Facebook AI Research.



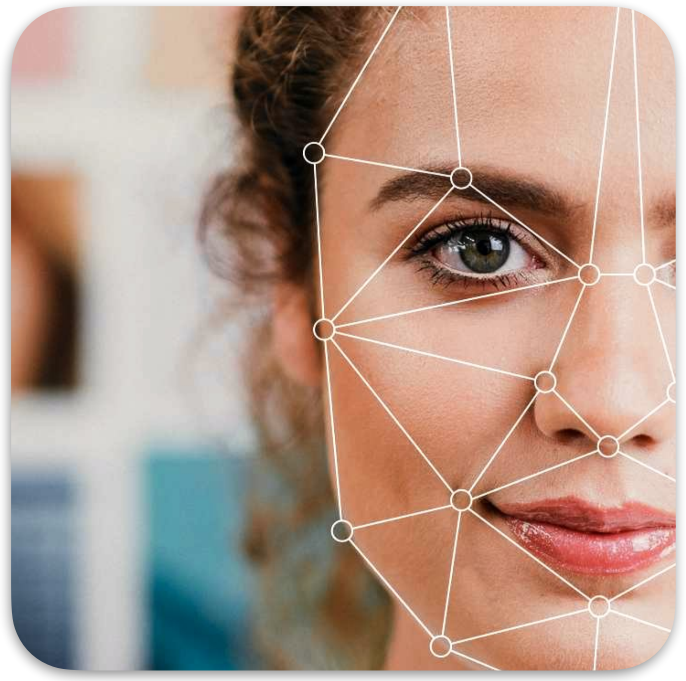
https://github.com/facebookresearch/faiss

What's Next?

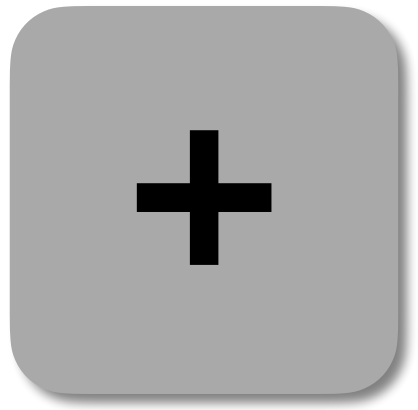
Content

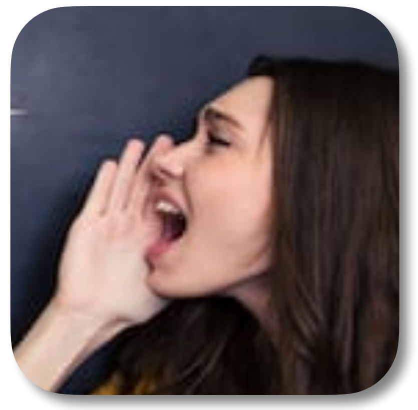


Basics
Concepts
Metrics
Metric implementation



Core Traits (3)
Concepts
Baseline implementation
Data collection
Evaluation
Attacks
Assignments





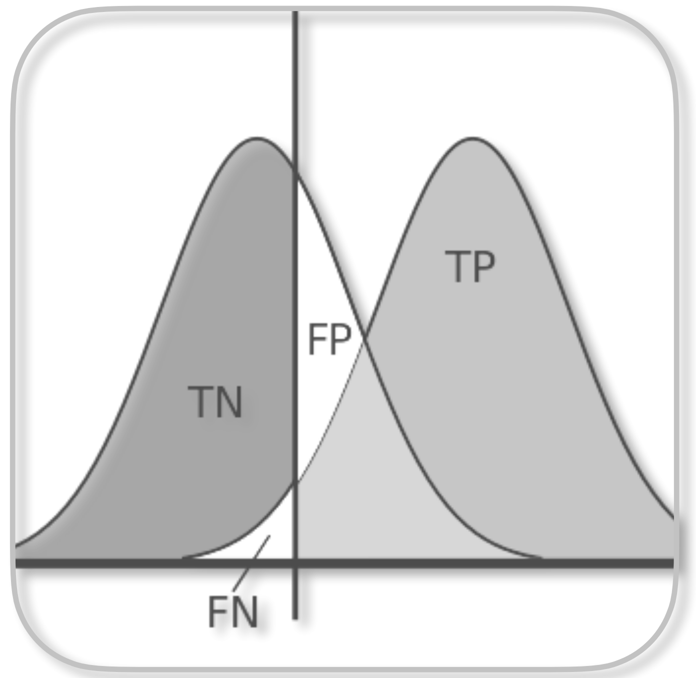
Alternative Traits and Fusion
Concepts

Invited Talks (2)
State of the art
Future work

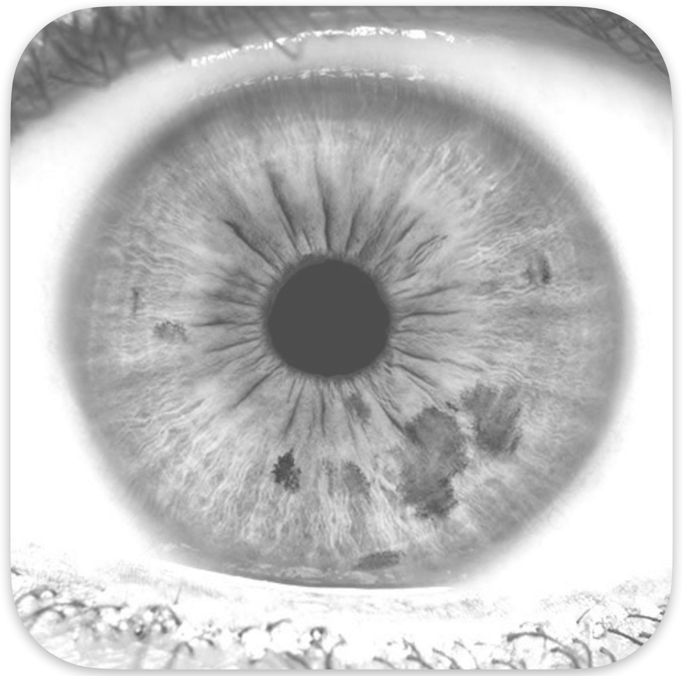
LOYOLA UNIVERSITY CHICAGO

What’s Next?

Content



Basics
Concepts
Metrics
Metric implementation



Core Traits (3)
Concepts
Baseline implementation
Data collection
Evaluation
Attacks
Assignments

Alternative Traits and Fusion
Concepts

Invited Talks (2)
State of the art
Future work

LOYOLA UNIVERSITY CHICAGO