Clustering and nearest neighbor search

Yannis Avrithis

University of Athens

Athens, November 2015

Problem

ANN search

- Given query point $\mathbf{q}$, find its nearest neighbor with respect to Euclidean distance within data set $\mathcal{X}$ in a d-dimensional space
- Encode (compress) vectors, speed up distance computations
- Fit underlying distribution with little space & time overhead

Vector quantization

- Given data set $\mathcal{X}$, map it to discrete codebook $\mathcal{C}$ such that distortion is minimized
- Use ANN search to assign points to centroids
- Use vector quantization to improve ANN search

Problem

ANN search

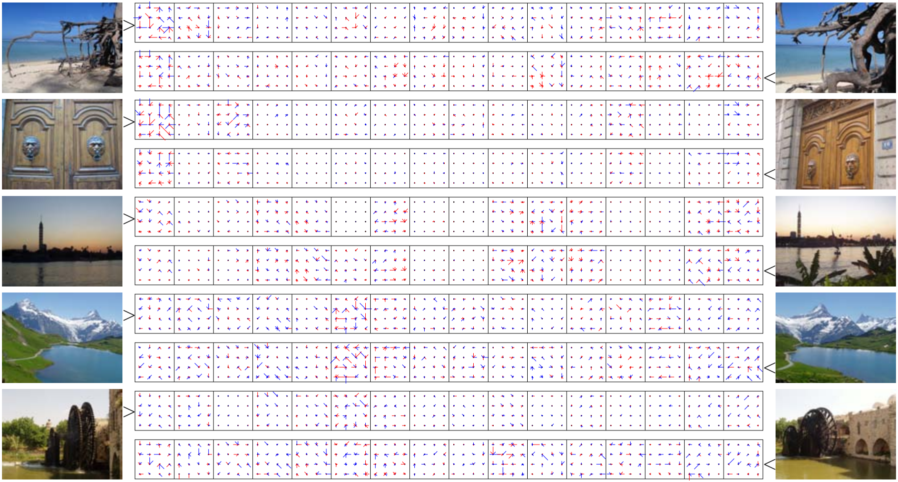
- Given query point $\mathbf{q}$, find its nearest neighbor with respect to Euclidean distance within data set $\mathcal{X}$ in a d-dimensional space
- Encode (compress) vectors, speed up distance computations
- Fit underlying distribution with little space & time overhead

Vector quantization

- Given data set $\mathcal{X}$, map it to discrete codebook $\mathcal{C}$ such that distortion is minimized
- Use ANN search to assign points to centroids
- Use vector quantization to improve ANN search

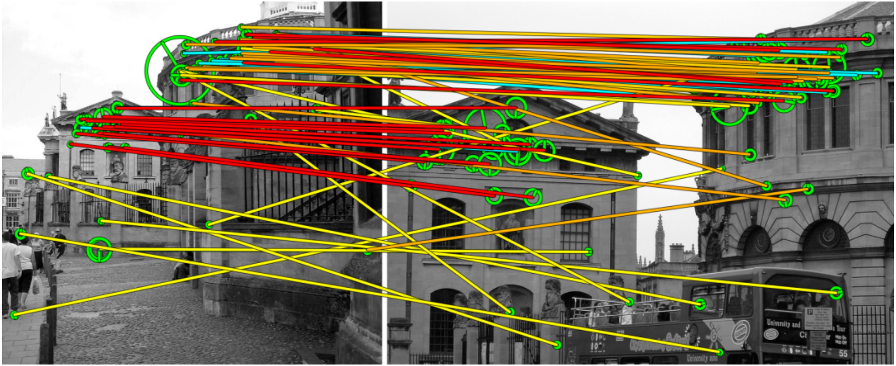
Applications in vision

Retrieval (image as point) [Jégou et al. '10][Perronnin et al. '10]



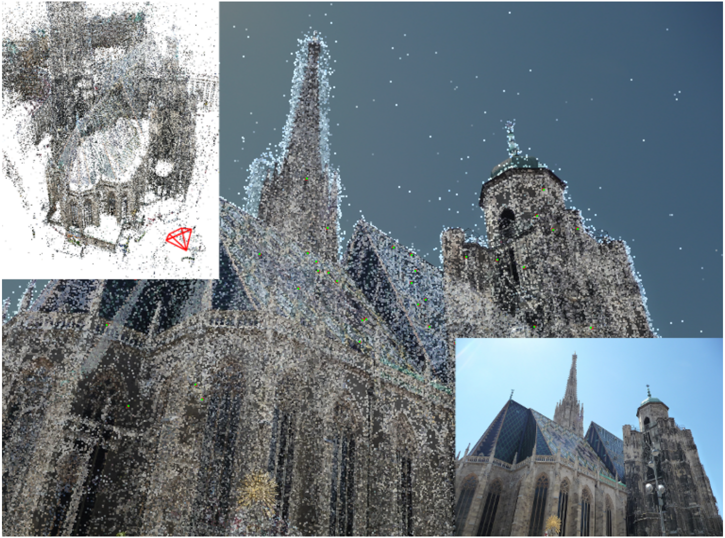
Applications in vision

Retrieval (patch as point) [Tolias et al. '13][Qin et al. '13]



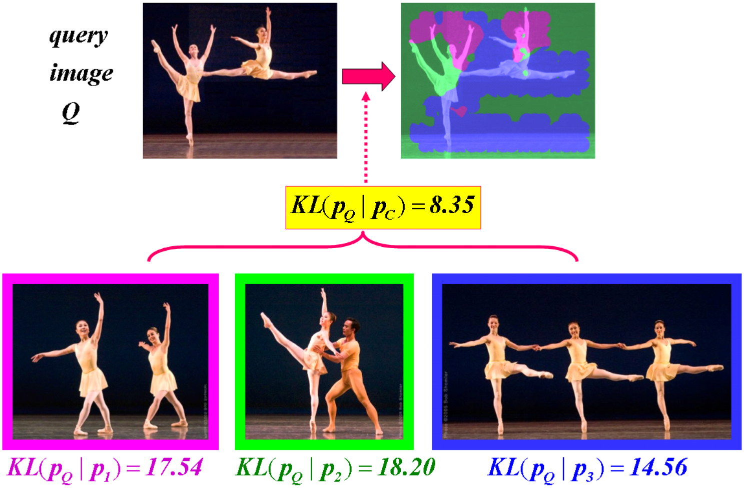
Applications in vision

Localization, pose estimation [Sattler et al. '12][Li et al. '12]



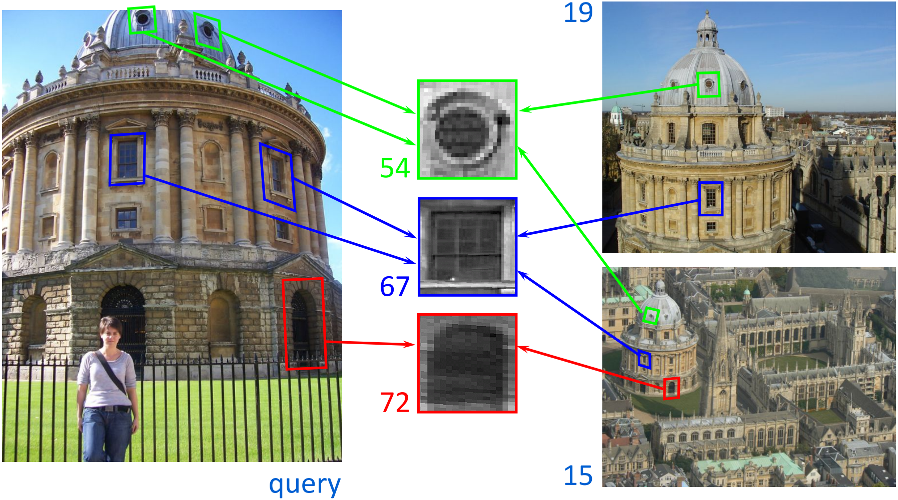
Applications in vision

Classification [Boiman et al. '08][McCann & Lowe '12]



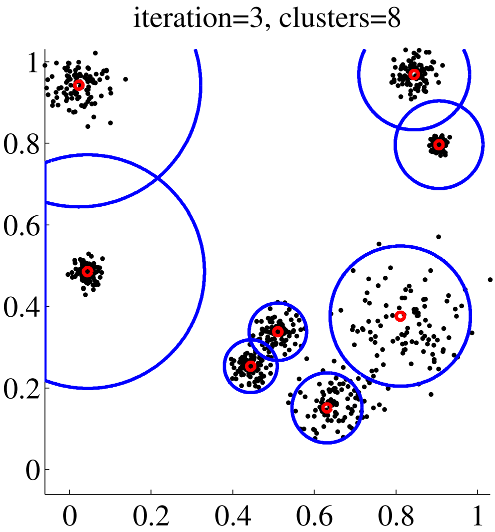
Applications in vision

BoW (patch quantization) [Sivic et al. ’03][Philbin et al. ’07]



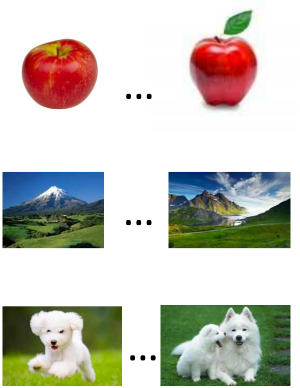
Applications in vision

BoW (codebook construction) [Philbin et al. '07][Avrithis '12]



Applications in vision

Image clustering [Gong et al. '15][Avrithis '15]



1010010101001001001
1011010001001001101
1011010101001101001
1011010101001001011

→ 1010010101001001001

0100101010100000110
0110101110100000110
0100111010100100110

→ 0100101010100000110

0010010000000011101
0010010000000011011
0010000000000001111
0010010000000011010
0010010000110011110

→ 0010010000000011101

Images | binary hashes | binary cluster centers

Overview (1)

Binary codes

- locality sensitive hashing [Charikar '02]
- spectral hashing [Weiss *et al.* '08]
- iterative quantization [Gong and Lazebnik '11]

Quantization

- vector quantization (VQ) [Gray '84]
- product quantization (PQ) [Jégou *et al.* '11]
- optimized product quantization (OPQ) [Ge *et al.* '13]
  Cartesian k-means [Norouzi & Fleet '13]
- locally optimized product quantization (LOPQ) [Kalantidis and Avrithis '14]

Overview (1)

Binary codes

- locality sensitive hashing [Charikar '02]
- spectral hashing [Weiss *et al.* '08]
- iterative quantization [Gong and Lazebnik '11]

Quantization

- vector quantization (VQ) [Gray '84]
- product quantization (PQ) [Jégou *et al.* '11]
- optimized product quantization (OPQ) [Ge *et al.* '13]
  Cartesian k-means [Norouzi & Fleet '13]
- locally optimized product quantization (LOPQ) [Kalantidis and Avrithis '14]

Overview (2)

Non-exhaustive search

- non-exhaustive PQ [Jégou *et al.* '11]
- inverted multi-index [Babenko & Lempitsky '12]
- multi-LOPQ [Kalantidis and Avrithis '14]

Clustering

- hierarchical k-means [Nister & Stewenius '06]
- approximate k-means [Philbin *et al.* '07]
- approximate Gaussian mixtures [Kalantidis & Avrithis '12]
- dimensionality-recursive vector quantization [Avrithis '13]
- ranked retrieval [Broder *et al.* '14]
- inverted-quantized k-means [Avrithis *et al.* '15]

Overview (2)

Non-exhaustive search

- non-exhaustive PQ [Jégou *et al.* '11]
- inverted multi-index [Babenko & Lempitsky '12]
- multi-LOPQ [Kalantidis and Avrithis '14]

Clustering

- hierarchical k-means [Nister & Stewenius '06]
- approximate k-means [Philbin *et al.* '07]
- approximate Gaussian mixtures [Kalantidis & Avrithis '12]
- dimensionality-recursive vector quantization [Avrithis '13]
- ranked retrieval [Broder *et al.* '14]
- inverted-quantized k-means [Avrithis *et al.* '15]

Binary codes

Locality sensitive hashing

random projections [Charikar '02]

- Choose a random vector $\mathbf{a}$ from the d-dimensional Gaussian distribution $\mathcal{N}(0,1)$.
- Define *hash function* $h_{\mathbf{a}} : \mathbb{R}^d \to \{-1, 1\}$ with

$$h_{\mathbf{a}}(\mathbf{x}) = \operatorname{sgn}(\mathbf{a} \cdot \mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{a} \cdot \mathbf{x} \geq 0 \\ -1, & \text{if } \mathbf{a} \cdot \mathbf{x} < 0. \end{cases}$$

- Then, given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$\mathbb{P}[h_{\mathbf{a}}(\mathbf{x}) = h_{\mathbf{a}}(\mathbf{y})] = 1 - \frac{\theta(\mathbf{x}, \mathbf{y})}{\pi}$$

where $\theta(\mathbf{x}, \mathbf{y})$ is the angle between $\mathbf{x}, \mathbf{y}$.

Locality sensitive hashing

random projections [Charikar '02]

- Choose a random vector $\mathbf{a}$ from the d-dimensional Gaussian distribution $\mathcal{N}(0,1)$.
- Define *hash function* $h_{\mathbf{a}} : \mathbb{R}^d \to \{-1, 1\}$ with

$$h_{\mathbf{a}}(\mathbf{x}) = \operatorname{sgn}(\mathbf{a} \cdot \mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{a} \cdot \mathbf{x} \geq 0 \\ -1, & \text{if } \mathbf{a} \cdot \mathbf{x} < 0. \end{cases}$$

- Then, given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$\mathbb{P}[h_{\mathbf{a}}(\mathbf{x}) = h_{\mathbf{a}}(\mathbf{y})] = 1 - \frac{\theta(\mathbf{x}, \mathbf{y})}{\pi}$$

where $\theta(\mathbf{x}, \mathbf{y})$ is the angle between $\mathbf{x}, \mathbf{y}$.

Binary codes and Hamming distance

- Given a set of n data points $\mathbf{x}_i \in \mathbb{R}^d$.
- Define k hash functions $h_j : \mathbb{R}^d \to \{-1, 1\}$, and let $h(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_k(\mathbf{x}))$.
- Encode each data point $\mathbf{x}$ by binary code $\mathbf{y} = h(\mathbf{x})$.
- Now, given a query $\mathbf{q}$, encode it as $h(\mathbf{q})$ and search in Y by Hamming distance.

Binary codes and Hamming distance

- Given a set of n data points $\mathbf{x}_i \in \mathbb{R}^d$.
- Define k hash functions $h_j : \mathbb{R}^d \to \{-1, 1\}$, and let $h(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_k(\mathbf{x}))$.
- Encode each data point $\mathbf{x}$ by binary code $\mathbf{y} = h(\mathbf{x})$.
- Now, given a query $\mathbf{q}$, encode it as $h(\mathbf{q})$ and search in Y by Hamming distance.

Spectral hashing

[Weiss et al. '08]

- Define similarity matrix S with $S_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/t^2)$.
- Require binary codes to be similarity preserving, balanced, and uncorrelated:

$$\begin{array}{rl} \text{minimize} & \sum_{ij} S_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|^2 \\ \text{subject to} & \mathbf{y}_i \in \{-1,1\}^k \\ & \sum_i \mathbf{y}_i = 0 \\ & \frac{1}{n}\sum_i \mathbf{y}_i\mathbf{y}_i^\top = I. \end{array}$$

Spectral hashing

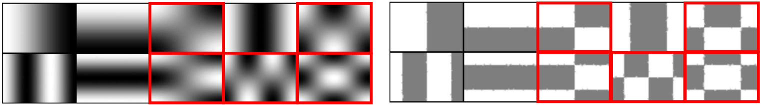
[Weiss et al. '08]

- Define similarity matrix S with $S_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/t^2)$.
- Require binary codes to be similarity preserving, balanced, and uncorrelated:

$$\begin{array}{rl} \text{minimize} & \sum_{ij} S_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|^2 \\ \text{subject to} & \mathbf{y}_i \in \{-1,1\}^k \\ & \sum_i \mathbf{y}_i = 0 \\ & \frac{1}{n}\sum_i \mathbf{y}_i\mathbf{y}_i^\top = I. \end{array}$$

Spectral hashing

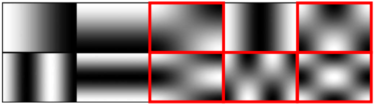
Example

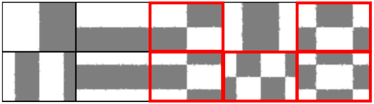


- Red: outer-product eigenfunctions: excluded
- Better to cut long dimension first
- Lower spatial frequencies are better than higher ones

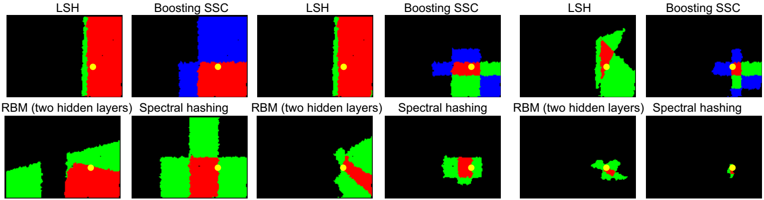
Spectral hashing

Example





- Red: outer-product eigenfunctions: excluded
- Better to cut long dimension first
- Lower spatial frequencies are better than higher ones

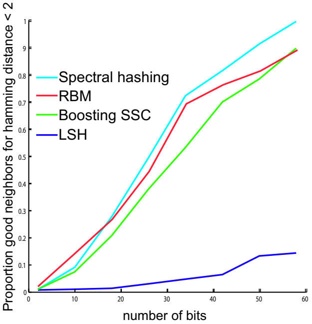


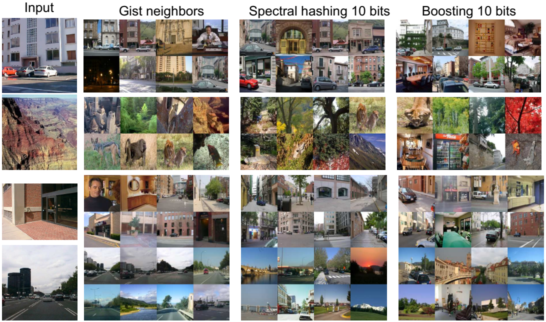
a) 3 bits b) 7 bits c) 15 bits

- Red: radius $= 0$; green: radius $= 1$; blue: radius $= 2$

Spectral hashing

Result on LabelMe

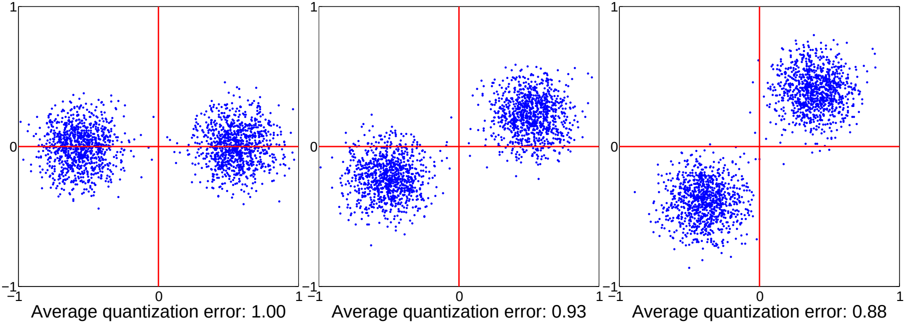




Iterative quantization

[Gong and Lazebnik '11]

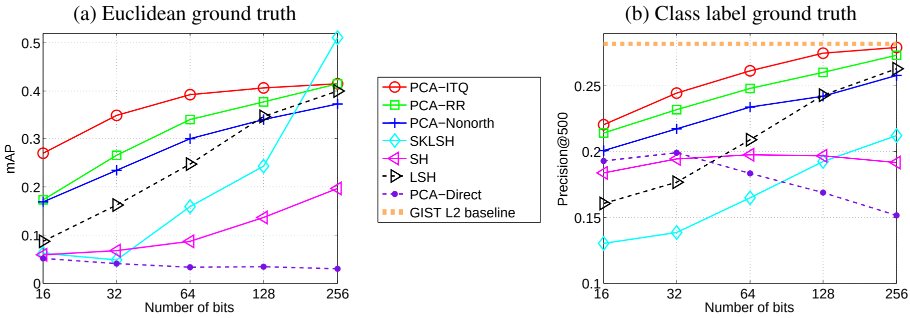
Quantize each data point to the closest vertex of the binary cube, $(\pm 1, \pm 1)$.



(a) PCA aligned. (b) Random Rotation. (c) Optimized Rotation.

Iterative quantization

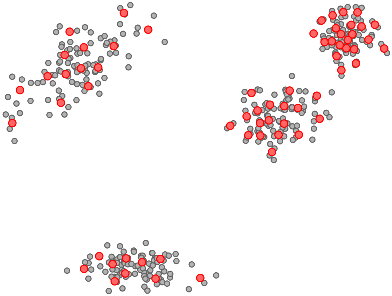
Result on CIFAR



Vector quantization

Vector quantization

[Gray '84]

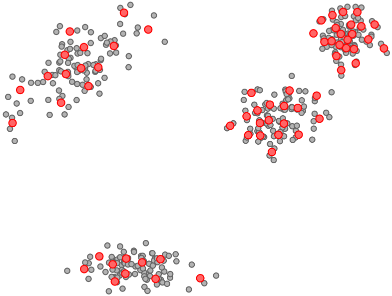


$$\text{minimize } E(\mathcal{C}) = \sum_{\mathbf{x}\in\mathcal{X}} \min_{\mathbf{c}\in\mathcal{C}} \|\mathbf{x} - \mathbf{c}\|^2 = \sum_{\mathbf{x}\in\mathcal{X}} \|\mathbf{x} - q(\mathbf{x})\|^2$$

distortion dataset codebook quantizer

Vector quantization

[Gray '84]



$$\text{minimize } E(\mathcal{C}) = \sum_{\mathbf{x}\in\mathcal{X}} \min_{\mathbf{c}\in\mathcal{C}} \|\mathbf{x}-\mathbf{c}\|^2 = \sum_{\mathbf{x}\in\mathcal{X}} \|\mathbf{x}-q(\mathbf{x})\|^2$$

distortion

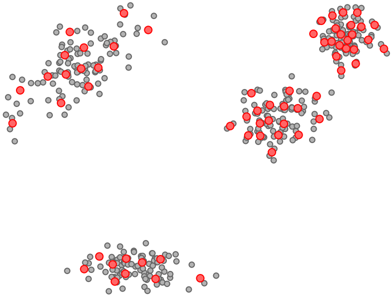
dataset

codebook

quantizer

Vector quantization

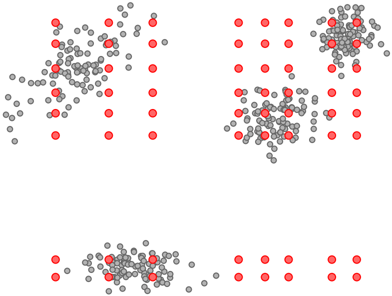
[Gray '84]



- For small distortion $\rightarrow$ large $k = |\mathcal{C}|$:
  - hard to train
  - too large to store
  - too slow to search

Product quantization

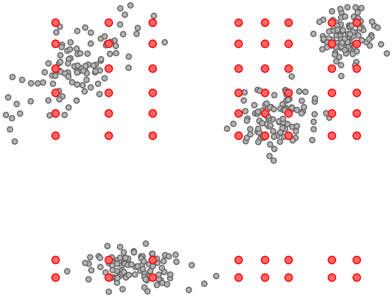
[Jégou et al. '11]



$$\begin{aligned} \text{minimize} \quad & \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x} - \mathbf{c}\|^2 \\ \text{subject to} \quad & \mathcal{C} = \mathcal{C}^1 \times \cdots \times \mathcal{C}^m \end{aligned}$$

Product quantization

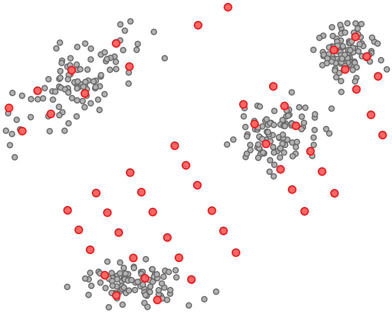
[Jégou et al. '11]



- train: $q = (q^1, \ldots, q^m)$ where $q^1, \ldots, q^m$ obtained by VQ
- store: $|\mathcal{C}| = k^m$ with $|\mathcal{C}^1| = \cdots = |\mathcal{C}^m| = k$
- search: $\|\mathbf{y} - q(\mathbf{x})\|^2 = \sum_{j=1}^{m} \|\mathbf{y}^j - q^j(\mathbf{x}^j)\|^2$ where $q^j(\mathbf{x}^j) \in \mathcal{C}^j$

Optimized product quantization

[Ge et al. '13]



$$\begin{aligned} \text{minimize} \quad & \sum_{\mathbf{x}\in\mathcal{X}} \min_{\hat{\mathbf{c}}\in\hat{\mathcal{C}}} \|\mathbf{x} - R^\top \hat{\mathbf{c}}\|^2 \\ \text{subject to} \quad & \hat{\mathcal{C}} = \mathcal{C}^1 \times \cdots \times \mathcal{C}^m \\ & R^\top R = I \end{aligned}$$

Optimized product quantization

Parametric solution for $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$

- From rate-distortion theory, distortion satisfies

$$E \geq k^{-2/d} d |\Sigma|^{1/d}$$

and practical distortion achieved by k-means is typically within $\sim 5\%$ of the bound. So after rotation $\hat{\Sigma} = R \Sigma R^\top$,

$$E_{\mathsf{PQ}} \geq k^{-2m/d} \frac{d}{m} \sum_{i=1}^{m} |\hat{\Sigma}_{ii}|^{m/d}$$

- But, by *arithmetic-geometric means* and *Fisher's* inequalities,

$$\frac{1}{m} \sum_{i=1}^{m} |\hat{\Sigma}_{ii}|^{m/d} \geq \prod_{i=1}^{m} |\hat{\Sigma}_{ii}|^{1/d} \geq |\hat{\Sigma}|^{1/d} = |\Sigma|^{1/d}$$

with equality implying balanced variance and independence.

Optimized product quantization

Parametric solution for $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$

- From rate-distortion theory, distortion satisfies

$$E \geq k^{-2/d} d |\Sigma|^{1/d}$$

and practical distortion achieved by k-means is typically within $\sim 5\%$ of the bound. So after rotation $\hat{\Sigma} = R \Sigma R^\top$,

$$E_{\mathsf{PQ}} \geq k^{-2m/d} \frac{d}{m} \sum_{i=1}^{m} |\hat{\Sigma}_{ii}|^{m/d}$$

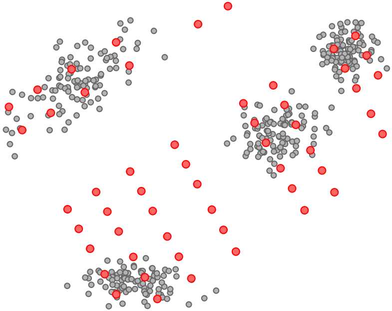
- But, by *arithmetic-geometric means* and *Fisher's* inequalities,

$$\frac{1}{m} \sum_{i=1}^{m} |\hat{\Sigma}_{ii}|^{m/d} \geq \prod_{i=1}^{m} |\hat{\Sigma}_{ii}|^{1/d} \geq |\hat{\Sigma}|^{1/d} = |\Sigma|^{1/d}$$

with equality implying balanced variance and independence.

Optimized product quantization

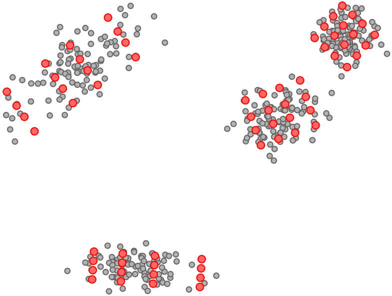
Parametric solution for $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$



- independence: PCA-align by diagonalizing Σ as $U \Lambda U^\top$
- balanced variance: permute Λ by π such that $\prod_i \lambda_i$ is constant in each subspace; $R \leftarrow U P_\pi^\top$
- find $\hat{\mathcal{C}}$ by PQ on rotated data $\hat{X} = RX$

Locally optimized product quantization

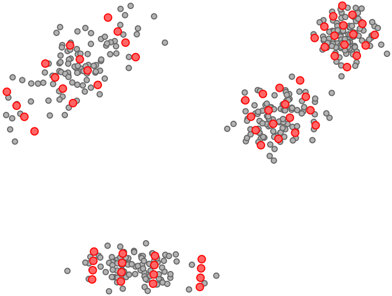
[Kalantidis & Avrithis '14]



- compute residuals $r(\mathbf{x}) = \mathbf{x} - Q(\mathbf{x})$ on coarse quantizer Q
- collect residuals $\mathcal{Z}_i = \{r(\mathbf{x}) : Q(\mathbf{x}) = \mathbf{c}_i\}$ per cell
- train $(R_i, q_i) \leftarrow \text{OPQ}(\mathcal{Z}_i)$ per cell

Locally optimized product quantization

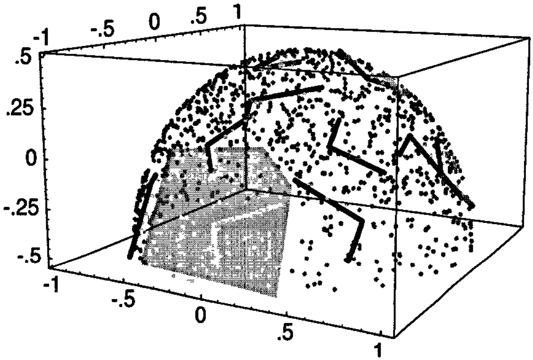
[Kalantidis & Avrithis '14]



- residual distributions closer to Gaussian assumption
- better captures the support of data distribution, like local PCA
  - multimodal (e.g. mixture) distributions
  - distributions on nonlinear manifolds

Local principal component analysis

[Kambhatla & Leen '97]



But, we are not doing dimensionality reduction!

Non-exhaustive search

Inverted index

IVFADC [Jégou et al. '11]

Construction

- train a coarse quantizer Q of K centroids or cells
- quantize each point $\mathbf{x} \in \mathcal{X}$ to $Q(\mathbf{x})$ and compute its residual vector $r(\mathbf{x}) = \mathbf{x} - Q(\mathbf{x})$
- quantize residuals by a product quantizer q
- for each cell, maintain an inverted list of data points and PQ-encoded residuals

Search

- quantize query $\mathbf{y}$ to w nearest cells
- exhaustively search by PQ only within the w inverted lists

Inverted index

IVFADC [Jégou et al. '11]

Construction

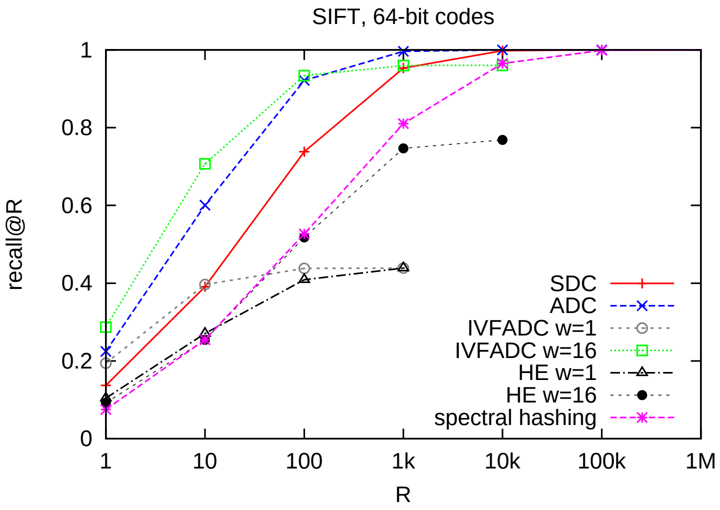
- train a coarse quantizer Q of K centroids or cells
- quantize each point $\mathbf{x} \in \mathcal{X}$ to $Q(\mathbf{x})$ and compute its residual vector $r(\mathbf{x}) = \mathbf{x} - Q(\mathbf{x})$
- quantize residuals by a product quantizer q
- for each cell, maintain an inverted list of data points and PQ-encoded residuals

Search

- quantize query $\mathbf{y}$ to w nearest cells
- exhaustively search by PQ only within the w inverted lists

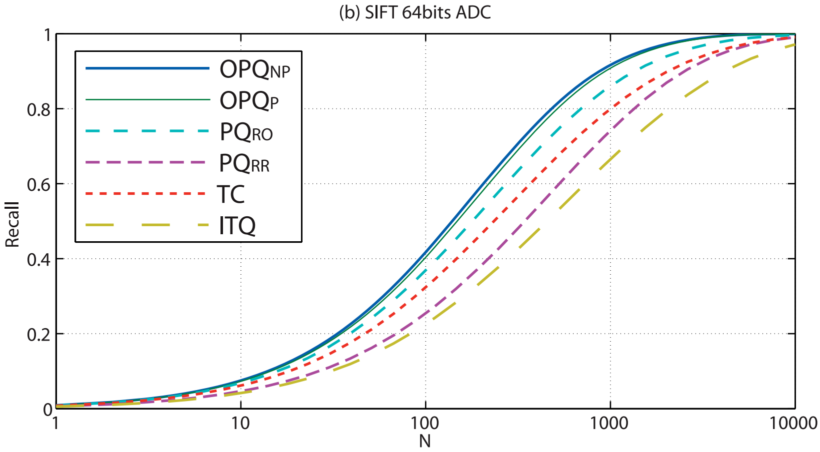
Product quantization

Comparison on SIFT1M



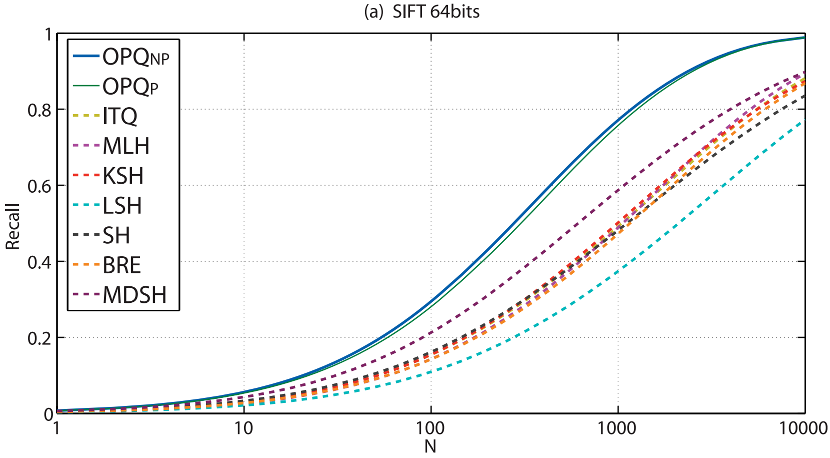
Optimized product quantization

Comparison on SIFT1M



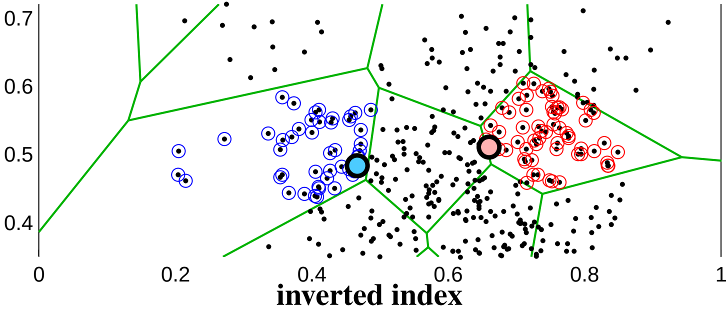
Optimized product quantization

vs. binary codes on SIFT1M



Inverted multi-index

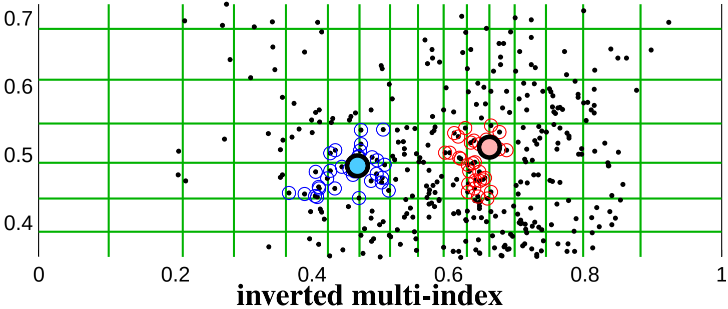
[Babenko & Lempitsky '12]



- train codebook $\mathcal{C}$ from dataset $\{\mathbf{x}_n\}$
- this codebook provides a coarse partition of the space

Inverted multi-index

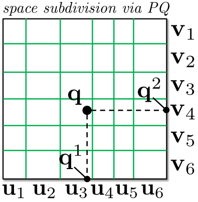
[Babenko & Lempitsky '12]



- decompose vectors as $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2)$
- train codebooks $\mathcal{C}^1, \mathcal{C}^2$ from datasets $\{\mathbf{x}^1_n\}, \{\mathbf{x}^2_n\}$
- induced codebook $\mathcal{C}^1 \times \mathcal{C}^2$ gives a finer partition
- given query $\mathbf{y}$, visit cells $(\mathbf{c}^1, \mathbf{c}^2) \in \mathcal{C}^1 \times \mathcal{C}^2$ in ascending order of distance to $\mathbf{y}$

Inverted multi-index

Multi-sequence algorithm



product quantization

q^1 vs. $\mathcal{U}$

i	$u_{\alpha(i)}$	r
1	u_3	0.5
2	u_4	0.7
3	u_5	4
4	u_2	6
5	u_1	8
6	u_6	9

q^2 vs. $\mathcal{V}$

j	$v_{\beta(j)}$	s
1	v_4	0.1
2	v_3	2
3	v_5	3
4	v_2	6
5	v_6	7
6	v_1	11

multi-sequence algorithm

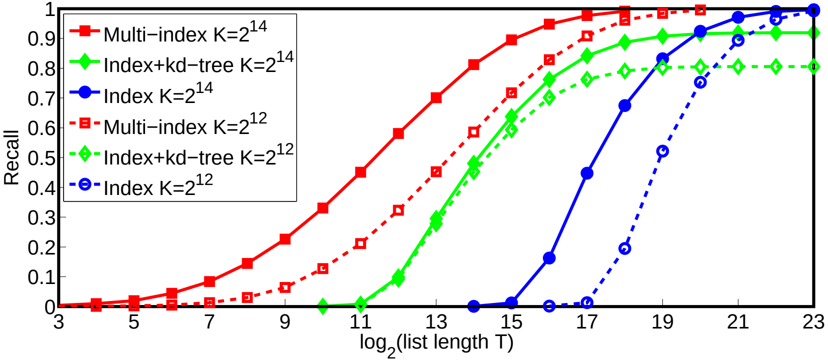
$[u_{\alpha(i)}\ v_{\beta(j)}]$	(i,j)	$r(i)+s(j)$
$[u_3\ v_4]$	(1,1)	0.6 (0.5+0.1)
$[u_4\ v_4]$	(2,1)	0.8 (0.7+0.1)
$[u_3\ v_3]$	(1,2)	2.5 (0.5+2)
$[u_4\ v_3]$	(2,2)	2.7 (0.7+2)
$[u_3\ v_5]$	(1,3)	3.5 (0.5+3)
$[u_4\ v_5]$	(2,3)	3.7 (0.7+3)
$[u_5\ v_4]$	(3,1)	4.1 (4+0.1)
$[u_5\ v_3]$	(3,2)	6 (4+2)
$[u_3\ v_2]$	(1,4)	6.5 (0.5+6)
	...	

	1 (u_3)	2 (u_4)	3 (u_5)	4 (u_2)	5 (u_1)	6 (u_6)	
1	0.6	0.8	4.1	6.1	8.1	9.1	v_4
2	2.5	2.7	6	8	10	11	v_3
3	3.5	3.7	7	9	11	12	v_5
4	6.5	6.7	10	12	14	15	v_2
5	7.5	7.7	11	13	15	16	v_6
6	11.5	11.7	15	17	19	20	v_1

OUTPUT: $(1,1) \rightarrow W_{3,4}$ $(2,1) \rightarrow W_{4,4}$ $(1,2) \rightarrow W_{3,3}$ $(2,2) \rightarrow W_{4,3}$ $(1,3) \rightarrow W_{3,5}$

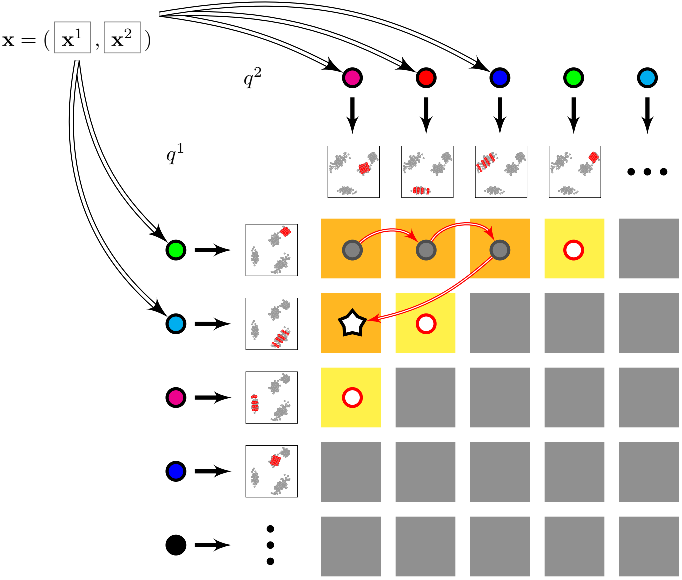
Inverted multi-index

Result on SIFT1B: are NN in candidate lists?



Multi-LOPQ

[Kalantidis & Avrithis '14]



Multi-LOPQ

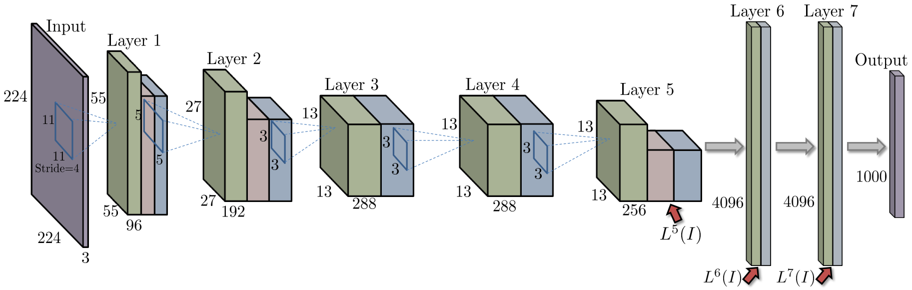
Result on SIFT1B, 128-bit codes

T	Method	$R = 1$	10	100
20K	IVFADC+R [Jégou *et al.* '11]	0.262	0.701	0.962
	LOPQ+R [Kalantidis & Avrithis '14]	0.350	0.820	0.978
10K	Multi-D-ADC [Babenko & Lempitsky '12]	0.304	0.665	0.740
	OMulti-D-OADC [Ge *et al.* '13]	0.345	0.725	0.794
	Multi-LOPQ [Kalantidis & Avrithis '14]	0.430	0.761	0.782
30K	Multi-D-ADC [Babenko & Lempitsky '12]	0.328	0.757	0.885
	OMulti-D-OADC [Ge *et al.* '13]	0.366	0.807	0.913
	Multi-LOPQ [Kalantidis & Avrithis '14]	0.463	0.865	0.905
100K	Multi-D-ADC [Babenko & Lempitsky '12]	0.334	0.793	0.959
	OMulti-D-OADC [Ge *et al.* '13]	0.373	0.841	0.973
	Multi-LOPQ [Kalantidis & Avrithis '14]	0.476	0.919	0.973

Application: image search

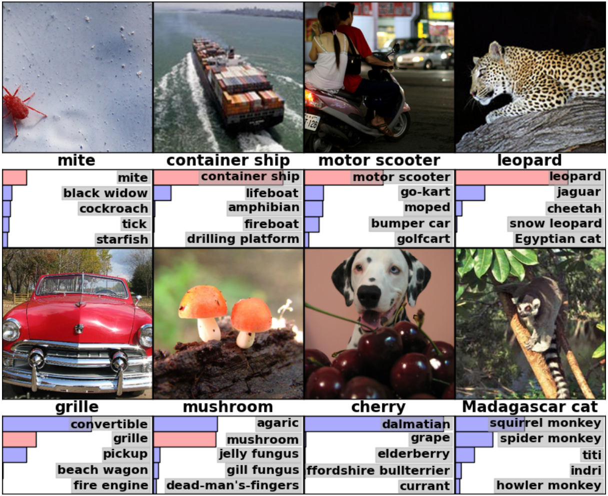
Deep learned image features

[Krizhevsky et al. '12]



Deep learned image features

Classification



Deep learned image features

Search

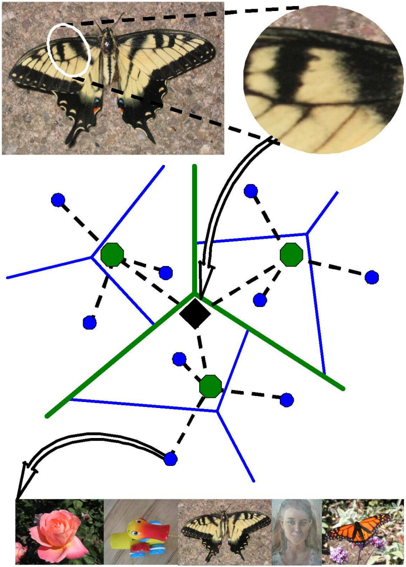
Multi-LOPQ

Image query on Flickr 100M (deep learned features, 4k $\rightarrow$ 128 dimensions)

Clustering

Hierarchical k-means

[Nister & Stewenius '06]



Approximate k-means

[Philbin et al. '07]

- centroids updated as in k-means
- points assigned to centroids by approximate search
- search by randomized k-d trees, even before the latter was published or FLANN was available
- index rebuilt in every k-means iteration

Approximate k-means

vs. Hierarchical k-means

Method	Dataset	mAP	
		Bag-of-words	Spatial
(a) HKM-1	5K	0.439	0.469
(b) HKM-2	5K	0.418	
(c) HKM-3	5K	0.372	
(d) HKM-4	5K	0.353	
(e) AKM	5K	0.618	0.647
(f) AKM	5K+100K	0.490	0.541
(g) AKM	5K+100K+1M	0.393	0.465

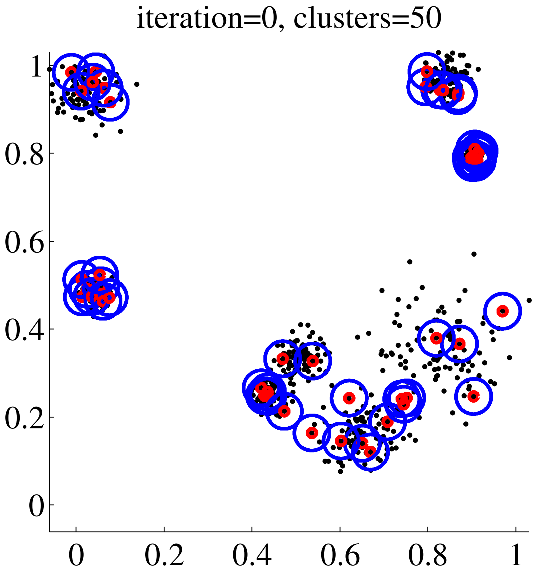
Robust approximate k-means

[Li et al. '10]

- the nearest neighbor in one iteration is re-used in the next
- less effort spent for new neighbor search
- faster convergence at same quality

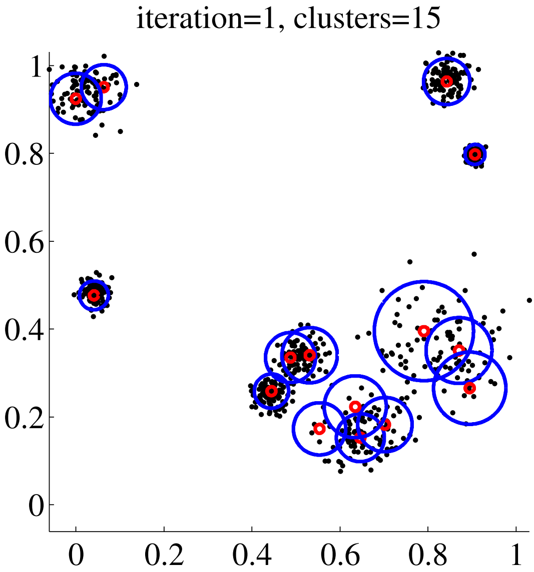
Approximate Gaussian mixtures

[Kalantidis & Avrithis '12]



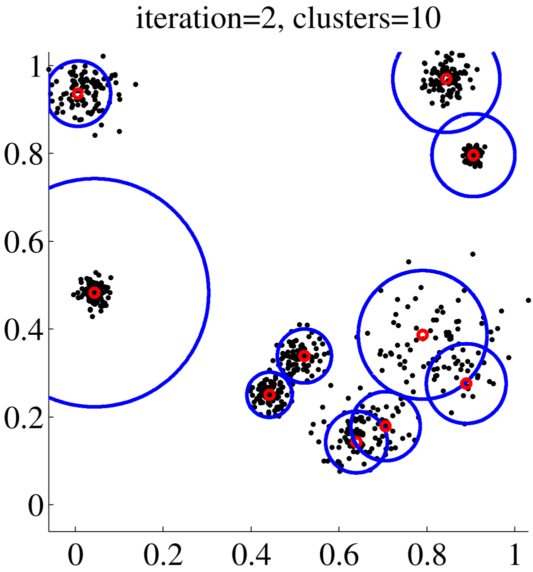
Approximate Gaussian mixtures

[Kalantidis & Avrithis '12]



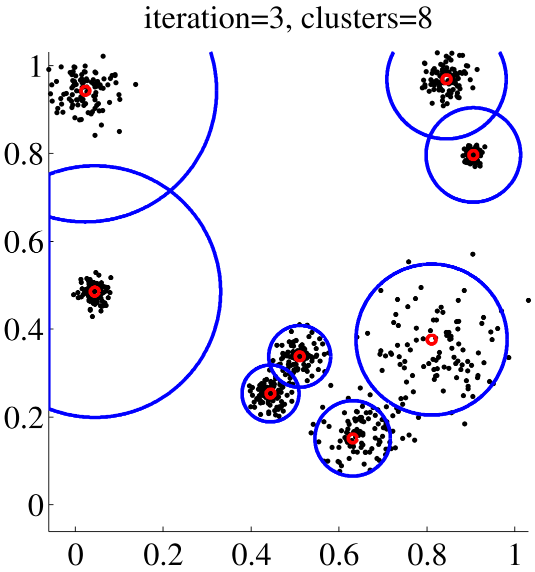
Approximate Gaussian mixtures

[Kalantidis & Avrithis '12]



Approximate Gaussian mixtures

[Kalantidis & Avrithis '12]



Expectation-maximization

[Dempster et al. '77]

- Mixture of K d-dimensional normal densities or components,

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k).$$

- Responsibility of component k for point $\mathbf{x}$:

$$\gamma_k(\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \mathbf{\Sigma}_j)}.$$

- Maximum likelihood solution for π, μ, Σ given N i.i.d. observations:

$$\pi_k = \frac{N_k}{N}$$

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^\top.$$

Expectation-maximization

[Dempster et al. '77]

- Mixture of K d-dimensional normal densities or components,

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k).$$

- Responsibility of component k for point $\mathbf{x}$:

$$\gamma_k(\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \mathbf{\Sigma}_j)}.$$

- Maximum likelihood solution for $\pi, \mu, \mathbf{\Sigma}$ given N i.i.d. observations:

$$\begin{aligned} \pi_k &= \frac{N_k}{N} \\ \mu_k &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n \\ \mathbf{\Sigma}_k &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^\top. \end{aligned}$$

Generalized responsibility and sampling

- Represent component k by function

$$p_k(\mathbf{x}) = \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k).$$

- Responsibility of component k for function q:

$$\hat{\gamma}_k(q) = \frac{\langle q, p_k \rangle}{\sum_{j=1}^{K} \langle q, p_j \rangle},$$

  where $\langle p, q \rangle = \int p(\mathbf{x}) q(\mathbf{x}) \mathrm{d}\mathbf{x}$ is the L^2 inner product.

- 'Sampling' a large component through a smaller one: $\langle p_1, p_2 \rangle \to p_1(\mu_2)$ and $\hat{\gamma}_1(p_2) \to \gamma_1(\mu_2)$ as $p_2(x) \to \delta(x - \mu_2)$.

Generalized responsibility and sampling

- Represent component k by function

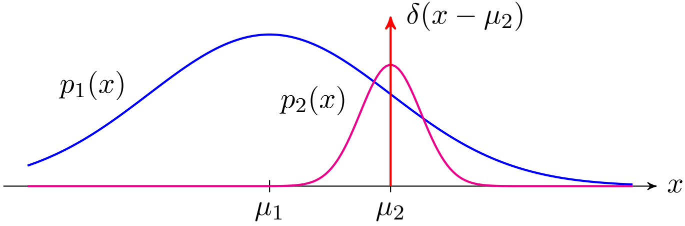
$$p_k(\mathbf{x}) = \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k).$$

- Responsibility of component k for function q:

$$\hat{\gamma}_k(q) = \frac{\langle q, p_k \rangle}{\sum_{j=1}^{K} \langle q, p_j \rangle},$$

where $\langle p, q \rangle = \int p(\mathbf{x}) q(\mathbf{x}) \mathrm{d}\mathbf{x}$ is the L^2 inner product.

- 'Sampling' a large component through a smaller one:
$\langle p_1, p_2 \rangle \to p_1(\mu_2)$ and $\hat{\gamma}_1(p_2) \to \gamma_1(\mu_2)$ as $p_2(x) \to \delta(x - \mu_2)$.



Approximate Gaussian mixtures

Image search—mAP on Oxford 5k

Method	RAKM					AKM	AGM
k	350k	500k	550k	600k	700k	550k	857k
5k	0.471	0.479	**0.486**	0.485	0.476	0.485	**0.492**
5k + 20k	0.439	0.440	**0.448**	0.441	0.437	0.447	**0.459**
5k + 1M	–	–	**0.250**	–	–	–	**0.280**

ANN search - clustering connection

- *hierarchical k-means*: use k-means tree for ANN search
- *approximate k-means*: use ANN search to accelerate assignment step
- *product quantization*: use k-means on subspaces to accelerate ANN search
- *inverted multi-index*: exhaustively search on subspaces before searching on entire space

What is the actual connection? Can we use recursion to solve both problems at the same time?

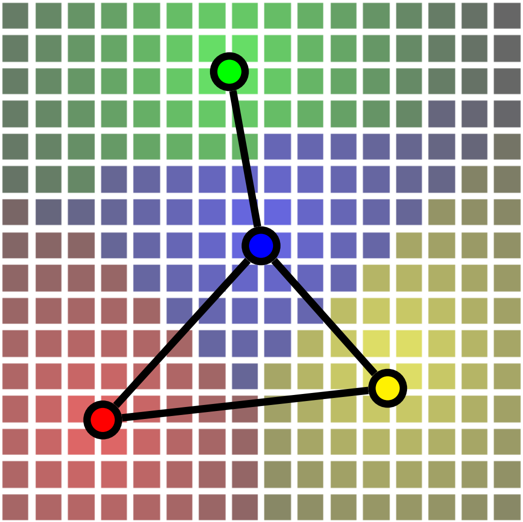
ANN search - clustering connection

- *hierarchical k-means*: use k-means tree for ANN search
- *approximate k-means*: use ANN search to accelerate assignment step
- *product quantization*: use k-means on subspaces to accelerate ANN search
- *inverted multi-index*: exhaustively search on subspaces before searching on entire space

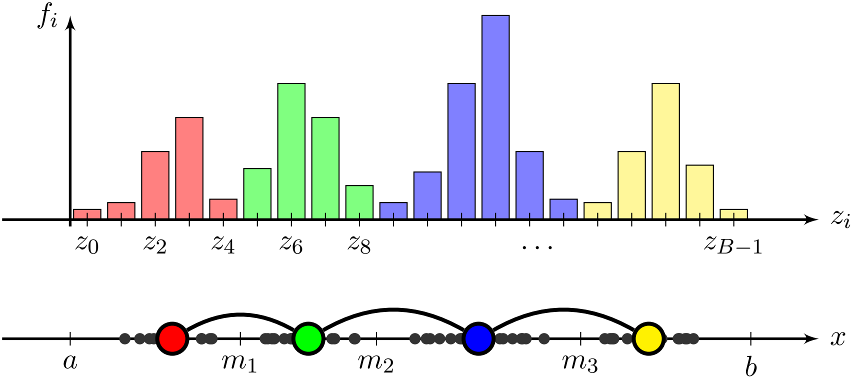
What is the actual connection? Can we use recursion to solve both problems at the same time?

Dimensionality-recursive vector quantization

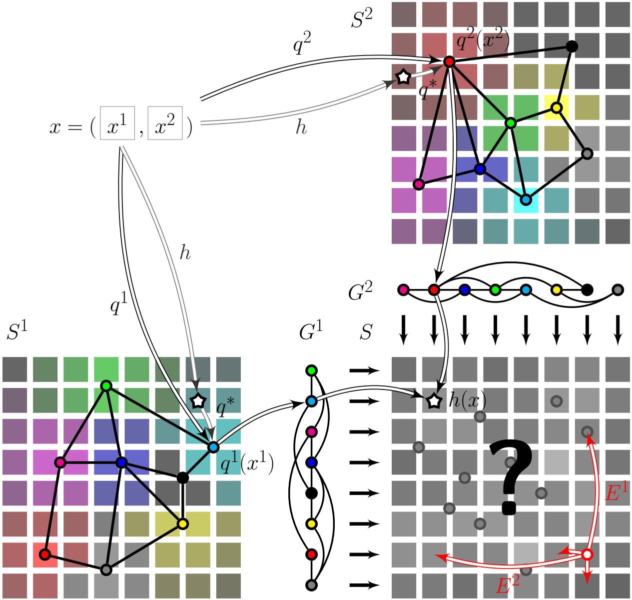
[Avrithis '13]



DRVQ base case: $d = 1$



DRVQ recursion: $d \to 2d$



DRVQ: vector quantization

k	16k	8k	4k	2k	1k	512
Approximate (μs)	0.95	0.83	0.80	0.73	0.80	0.90
Exact (ms)	1.19	0.79	0.51	0.26	0.21	0.11

averaged over the $n = 75$k SIFT descriptors of the 55 cropped query images of *Oxford 5k*

DRVQ: clustering

k	$\log k_p$ $(d = 2^p)$						time (m)
	1	2	4	8	16	32	
16k	6	7	8	9	11	14	129.96
8k	6	7	8	9	11	13	119.43
4k	6	7	8	9	10	12	20.07
2k	5	6	7	8	9	11	2.792
1k	5	6	7	8	9	10	2.608
512	4	5	6	7	8	9	0.866
4k	Approximate k-means						504.2

4 codebooks at $d = 32$ dimensions each on $n = 12.5$M 128-dimensional SIFT descriptors of *Oxford 5k*

Approximate k-means

[Philbin et al. '07]

- centroids updated as in k-means
- points assigned to centroid by approximate search
- index rebuilt in every k-means iteration

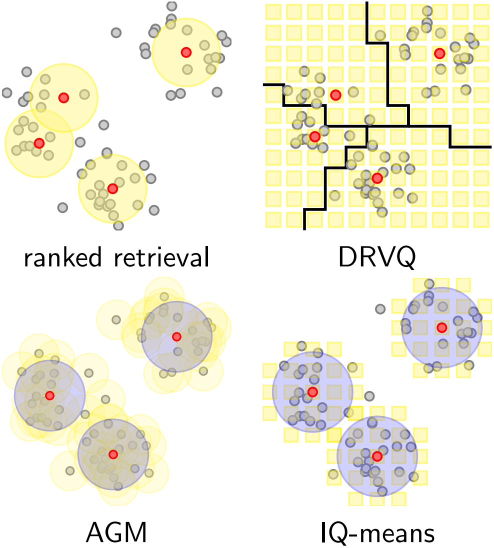
Ranked retrieval

[Broder et al. '14]

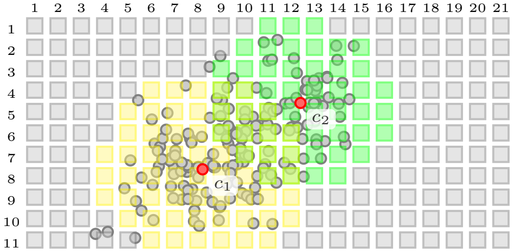
- centroids updated as in k-means
- points assigned by inverse search from centroids to points
- points may remain unassigned
- index built only once

Inverted-quantized k-means

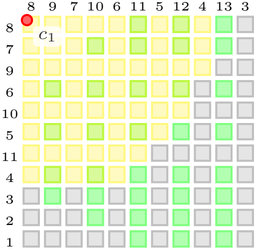
[Avrithis et al. '15]



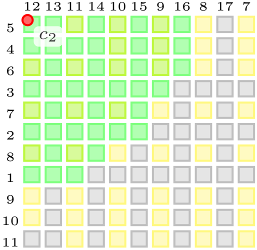
Inverted-quantized k-means



(a) visited cells on original grid



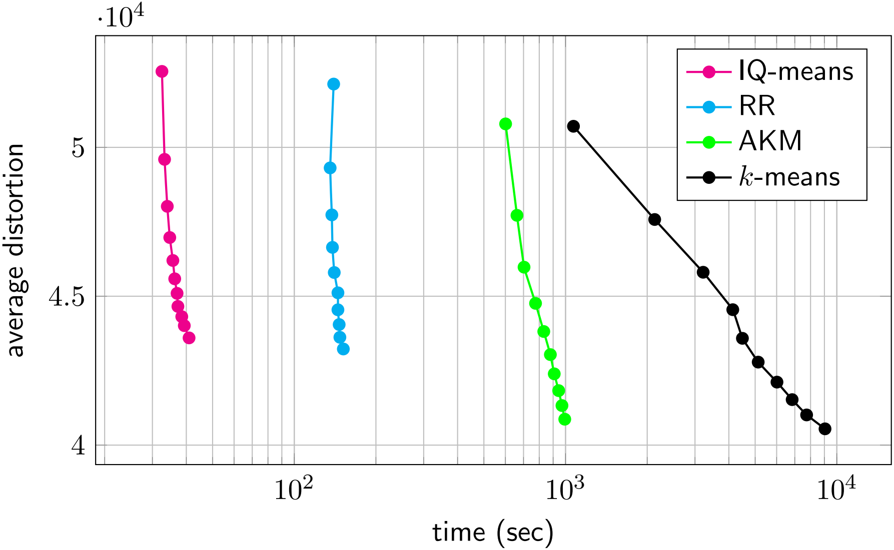
(b) search block of c_1



(c) search block of c_2

Inverted-quantized k-means

Comparison on SIFT1M with $k \in \{10^3, \ldots, 10^4\}$



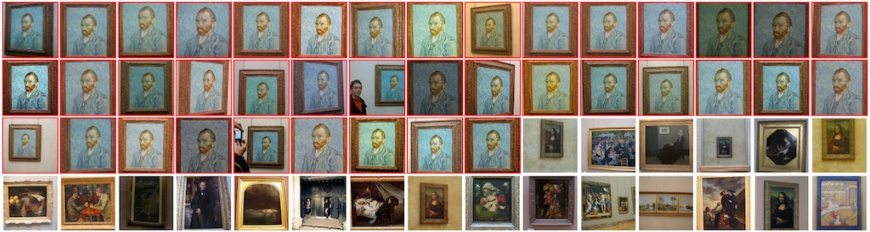
Inverted-quantized k-means

Time / iteration & average precision on YFCC100M, initial $k = 10^5$

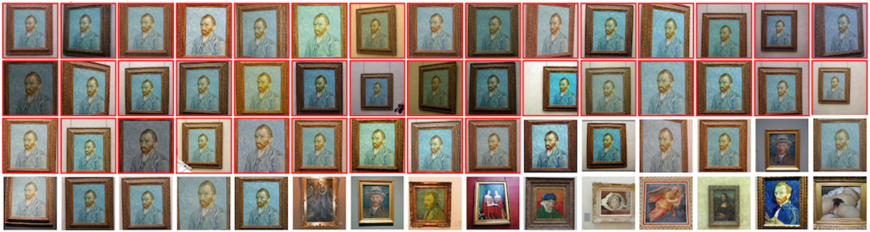
	Cell-KM	DKM ($\times 300$)	D-IQ-Means
k/k'	100000	100000	85742
time (s)	13068.1	7920.0	**140.6**
precision	0.474	**0.616**	0.550

Inverted-quantized k-means

Mining on a 100M image collection



Paris500k



Paris500k + YFCC100M

http://image.ntua.gr/iva/research/

Thank you!