CS 687
Jana Kosecka

- Nearest Neighbor Methods
- Instance Based Learning
- Application: object instance recognition

- Clustering
- Application: segmentation, visual vocabularies

Instance Based Learning

- Nearest neighbor methods
- Non-parametric learning
- Define similarity measure (Minkovski distance)
- Hamming distance with Boolean attributes

- K-d trees
- Locality sensitive hashing
- Min-hash

Distance Functions

- Minkowski Distance Lp norm

$$L_p(x_i, x_q) = (\sum_j (x_{j,i} - x_{q,i})^p)^{1/p}$$

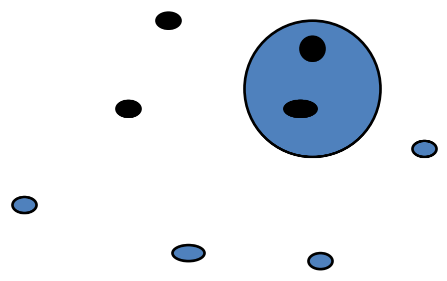
- For p = 1 Manhattan distance (often used for dissimilar attributes)
- For p = 2 Euclidean Distance
- Normalize each dimension (compute mean and standard deviation) and rescale all values to zero mean and unit variance
- Mahalanobis Distance – takes into account covariance between dimensions – where S is a covariance matrix

$$d(x_i, x_q) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}$$

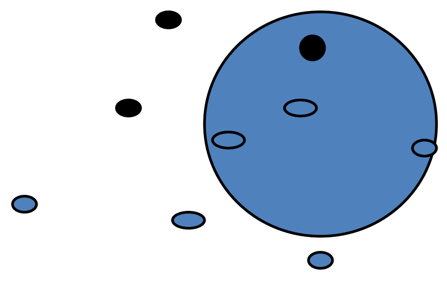
K-Nearest Neighbor

- Features
  - All instances correspond to points in an n-dimensional Euclidean space
  - Classification is delayed till a new instance arrives
  - Classification done by comparing feature vectors of the different points
  - Target function may be discrete or real-valued

1-Nearest Neighbor



3-Nearest Neighbor

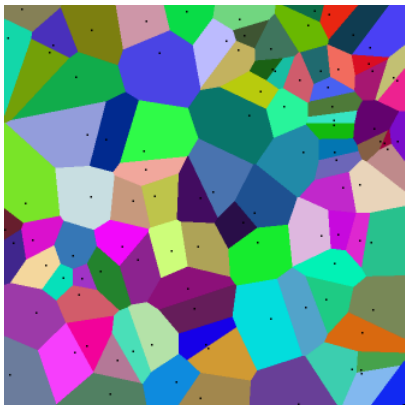


K-Nearest Neighbor

- An arbitrary instance is represented by $(a_1(x), a_2(x), a_3(x),.., a_n(x))$
  - $a_i(x)$ denotes features
- Euclidean distance between two instances
  $d(x_i, x_j)$=sqrt (sum for r=1 to n $(a_r(x_i) - a_r(x_j))^2$)
- Continuous valued target function
  - mean value of the k nearest training examples

Voronoi Diagram

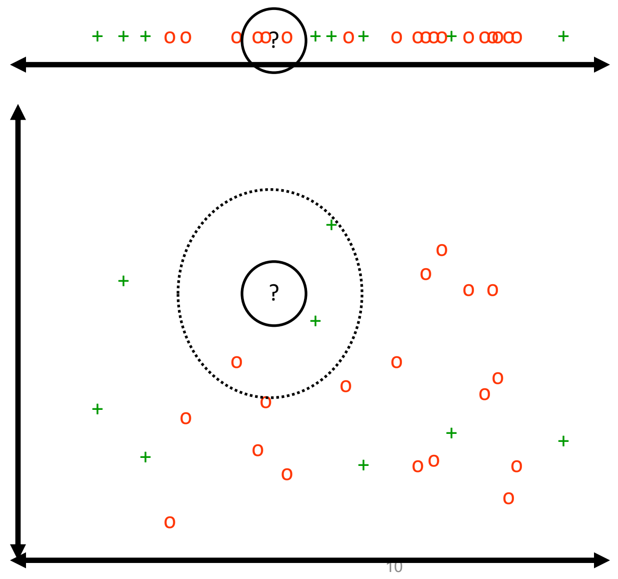
- Decision surface formed by the training examples



Distance-Weighted Nearest Neighbor Algorithm

- Assign weights to the neighbors based on their 'distance' from the query point
  - Weight 'may' be inverse square of the distances
- All training points may influence a particular instance
- Points in the local neighbourhood will influence an instance

K-NN and irrelevant features



Remarks

- Highly effective inductive inference method for noisy training data and complex target functions
- Target function for a whole space may be described as a combination of less complex local approximations
- Learning is very simple
- Classification is time consuming

- Curse of dimensionality

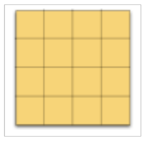
Curse of dimensionality

- Curse of dimensionality – nearest neighbors in high-dimensional spaces are very far – we often do not have enough data
- E.g. for N uniformly distributed points in n dimensional space length of average neighbourhood is $l = (k / N)^{1/n}$
- Average volume containing k points
- Average of the edge length of the cube $l^n = k / N$

- *For n=2 , l = 0.003*
- For *n=17* half edge length
- For *n=200* Probability that two points are less then distance 1 apart is *0.000000 …*

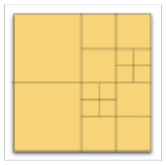
Nearest neighbours

- Instance based learning is simple and effective
- Naively comparing a test example against entire training set is expensive
- Important issue – structure data intelligently in order to avoid comparing against every point
- Structuring multi-dimensional data:
- Option: k-dimensional array of buckets; uniformly partition each dimension (ideal if data is uniformly distributed – which is never the case in multiple dimensions
- 2D example



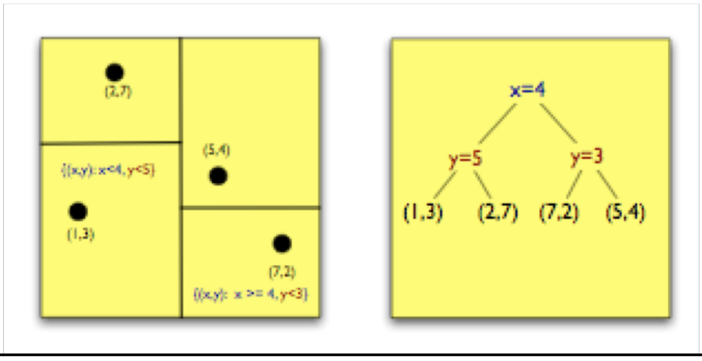
Nearest Neighbours

- Another alternative – quad-tree – each dimension partitioned in two – the cells with lot of data are partitioned further
- Binary tree splits each dimension in half
- Memory intensive, in 2D quad-tree



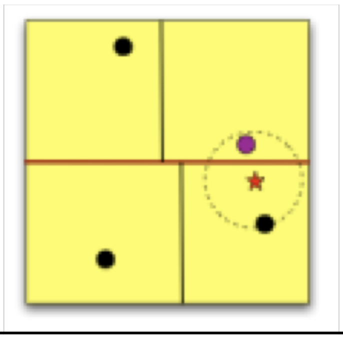
Nearest Neighbours

- k-d Tree - binary tree, which has both dimension number and splitting values at each node
- Splits the space into (hyper) rectangles
- Searching for a point in the tree is accomplished by traversing the tree *O(lg(n))* if there are n data points



Nearest Neighbour Seach

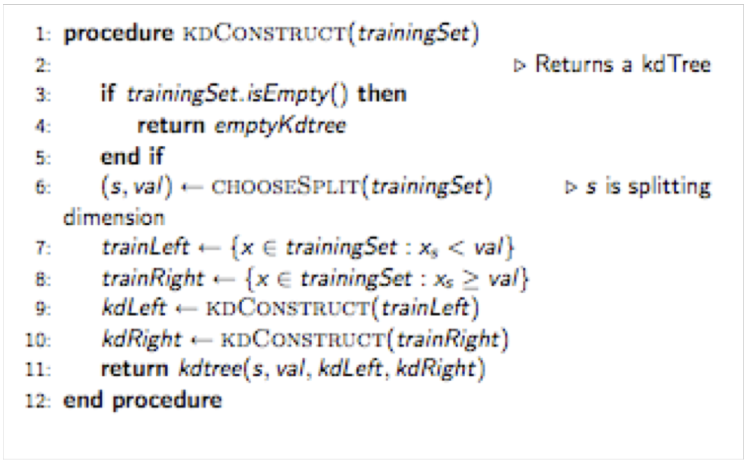
- Depth first search to nearest neighbour
- End up in hyper rectangle, find the nearest point there – not necessarily the nearest neighbour
- There can be another point then leaf which is closer



Constructing a kD-tree

- Select the dimension to split on
- Do this recursively for each child of the split
- Want to split along dimension along which examples are well spread
  1. choose the dimension with greatest variances
  2. choose the dimension with greatest range
- Want to split in the middle – median, mean

Constructing KD tree



```
procedure KDCONSTRUCT(trainingSet)
                                                  ▷ Returns a kdTree
    if trainingSet.isEmpty() then
        return emptyKdtree
    end if
    (s, val) ← CHOOSESPLIT(trainingSet)           ▷ s is splitting
dimension
    trainLeft ← {x ∈ trainingSet : $x_s < val$}
    trainRight ← {x ∈ trainingSet : $x_s \geq val$}
    kdLeft ← KDCONSTRUCT(trainLeft)
    kdRight ← KDCONSTRUCT(trainRight)
    return kdtree(s, val, kdLeft, kdRight)
end procedure
```

References

- Wikipedia on kD-tree
- Introductory tutorial in kD-tree Andrew Moore

Locally Sensitive Hashing

- Application: Large Scale Image Retrieval
- Even faster look up then binary trees, randomized algorithm
- Find approximate Nearest Neighbours
- Find x_j which is within the radius r of the query point, with a high probability
- Need a hash function $g(x)$ such that two points which are near by will hash to the same code
- Assume that each point is a bit string 0 1 0 1 0 0 ….
- Intuition – bin the axes, and the points which fall in the same bin are likely to be close, how about higher dimension ?

Locally Sensitive Hashing

- Idea: create several random projections and combine the results
- Create *l* hash tables $g_1(x), g_2(x) \cdots g_l(x)$
- Each will model one projection of the point
- Enter all the examples in the hash tables
- For query point, get all the examples which are in the same bin for all hash function, take union of these points
- Those will be the candidates

- Application; database of 13 million web images, image descriptor 512 dimensions, LSH examines only thousands examples (thousand fold speedup over exhastive or k-d tree)

Supervised vs. Unsupervised Learning

- So far we have assumed that the training samples used to design the classifier were labeled by their class membership (supervised learning)

- We assume now that all one has is a collection of samples without being told their categories (unsupervised learning)

Clustering

- Goal: Grouping a collection of objects (data points) into subsets or “clusters”, such that those within each cluster are more closely related to one other than objects assigned to different clusters.
- Fundamental to all clustering techniques is the choice of ***distance or dissimilarity measure*** between two objects.

What is Similarity?

The quality or state of being similar; likeness; resemblance; as, a similarity of features.

Webster's Dictionary

Similarity is hard to define, but…
“*We know it when we see it*”

The real meaning of similarity is a philosophical question. We will take a more pragmatic approach.

Slide by E. Keogh

K-means

- One of the most popular iterative descent clustering methods.
- Features: quantitative type.
- Dissimilarity measure: Euclidean distance.

K-means

The "***within cluster point scatter***" becomes:

$$W(C)=\frac{1}{2}\sum_{k=1}^{K}\sum_{i\in C_k}\sum_{j\in C_k}\left\|\boldsymbol{x}_i-\boldsymbol{x}_j\right\|^2$$

$W(C)$ can be rewritten as:

$$W(C)=\sum_{k=1}^{K}\left|C_k\right|\sum_{i\in C_k}\left\|\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k\right\|^2$$

(obtained by rewriting $(\boldsymbol{x}_i-\boldsymbol{x}_j)=(\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k)-(\boldsymbol{x}_j-\bar{\boldsymbol{x}}_k)$)

where

$\bar{\boldsymbol{x}}_k=\frac{1}{\left|C_k\right|}\sum_{i\in C_k}\boldsymbol{x}_i$ is the mean vector of cluster C_k

$\left|C_k\right|$ is the number of points in cluster C_k

K-means

The objective is :

$$\min_{C} \sum_{k=1}^{K} |C_k| \sum_{i \in C_k} \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k\|^2$$

We can solve this problem by noticing :

for any set of data S

$$\bar{\boldsymbol{x}}_S = \arg\min_{\boldsymbol{m}} \sum_{i \in S} \|\boldsymbol{x}_i - \boldsymbol{m}\|^2$$

(this is obtained by setting $\dfrac{\partial \sum_{i \in S} \|\boldsymbol{x}_i - \boldsymbol{m}\|^2}{\partial \boldsymbol{m}} = 0$)

So we can solve the enlarged optimization problem :

$$\min_{C, \boldsymbol{m}_k} \sum_{k=1}^{K} |C_k| \sum_{i \in C_k} \|\boldsymbol{x}_i - \boldsymbol{m}_k\|^2$$

K-means: The Algorithm

1. Given a cluster assignment C, the total within cluster scatter

$\sum_{k=1}^{K} |C_k| \sum_{i \in C_k} \|\boldsymbol{x}_i - \boldsymbol{m}_k\|^2$ is minimized with respect to the $\{\boldsymbol{m}_1, \cdots, \boldsymbol{m}_K\}$

giving the means of the currently assigned clusters;

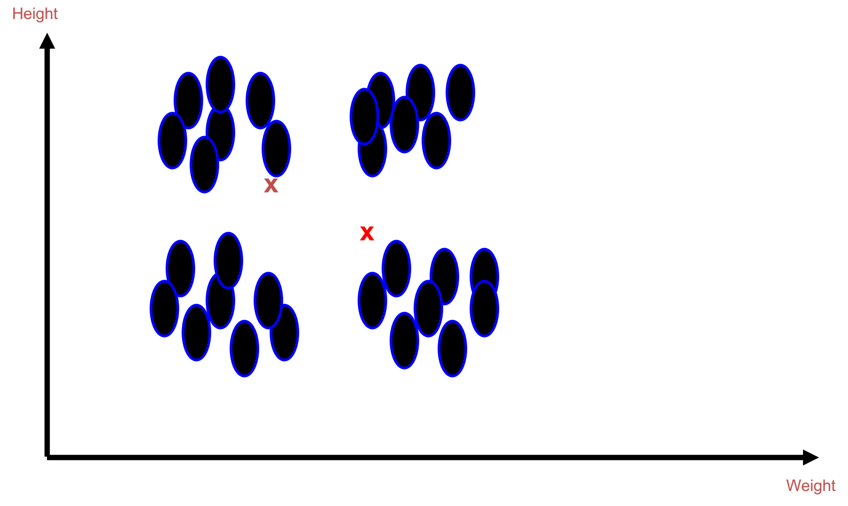
2. Given a current set of means $\{\boldsymbol{m}_1, \cdots, \boldsymbol{m}_K\}$,

$\sum_{k=1}^{K} |C_k| \sum_{i \in C_k} \|\boldsymbol{x}_i - \boldsymbol{m}_k\|^2$ is minimized with respect to C

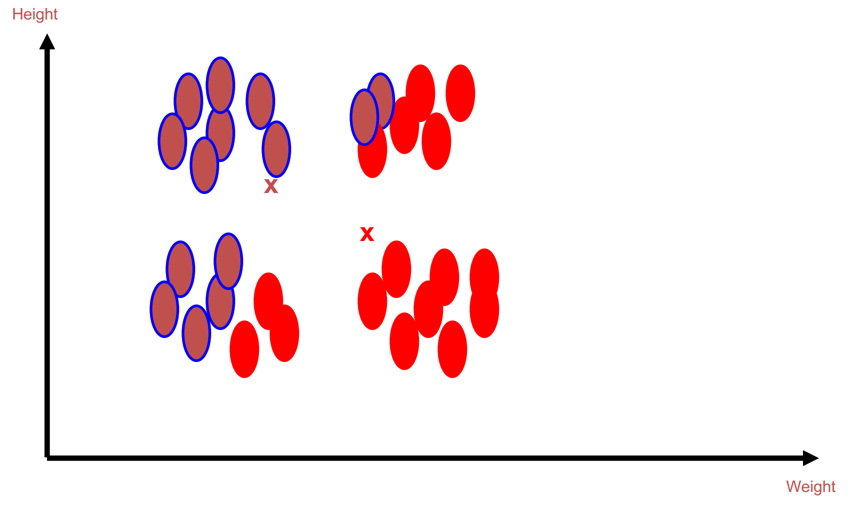
by assigning each point to the closest current cluster mean;

3. Steps 1 and 2 are iterated until the assignments do not change.

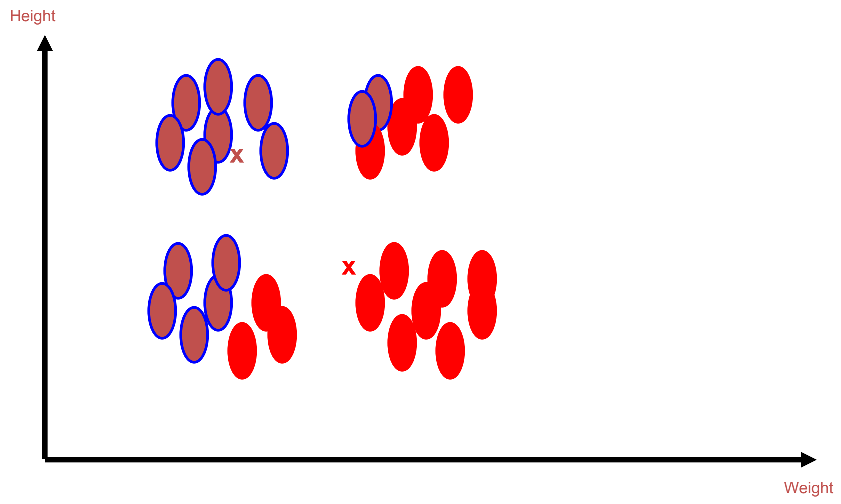
K-means Example (K=2)
Initialize Means



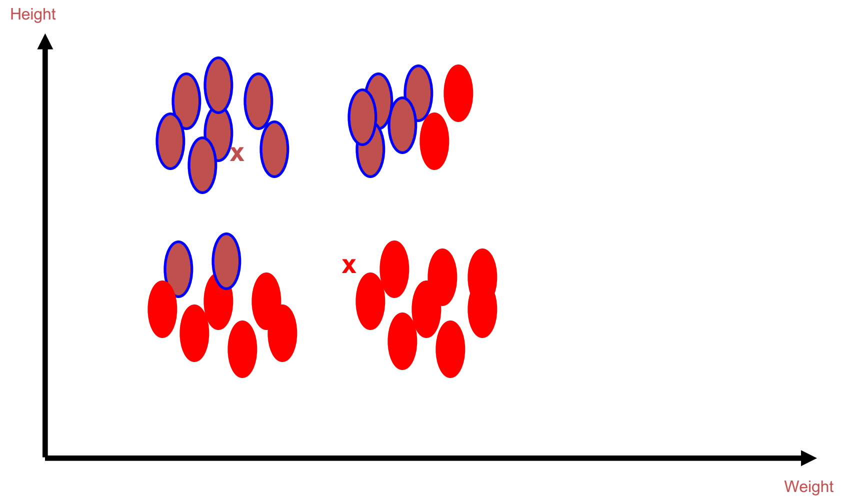
K-means Example
Assign Points to Clusters



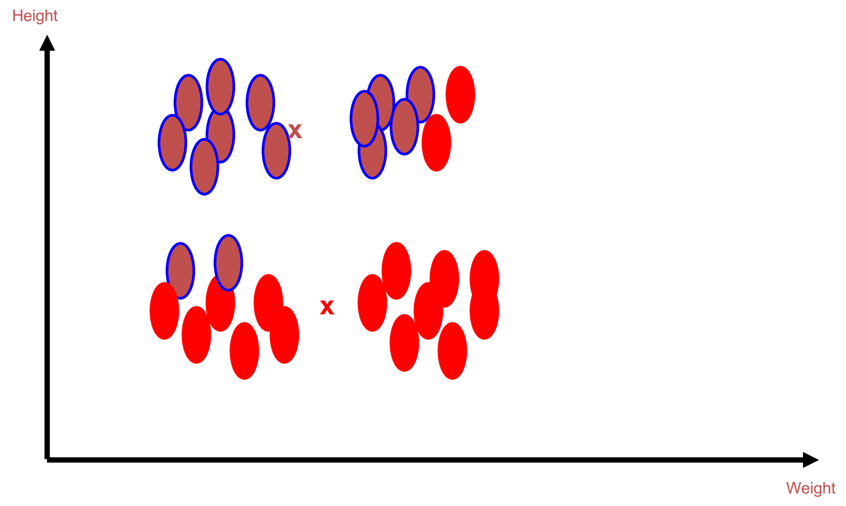
K-means Example
Re-estimate Means



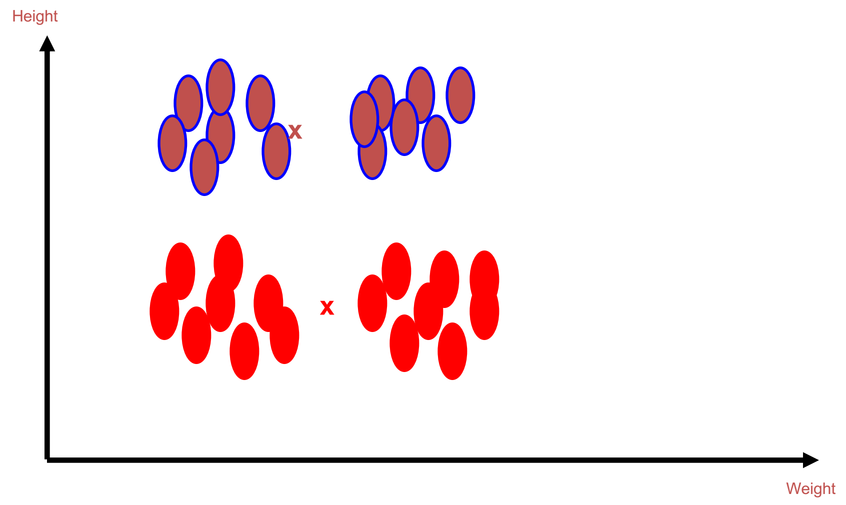
K-means Example
Re-assign Points to Clusters



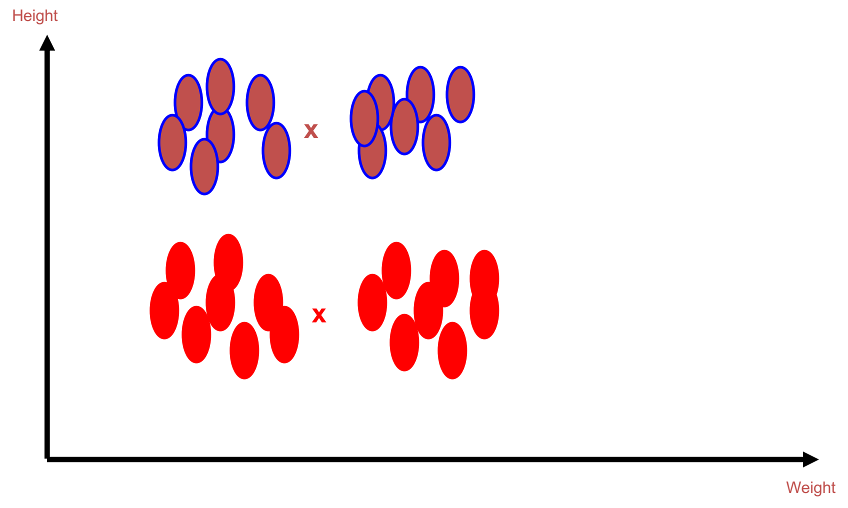
K-means Example
Re-estimate Means



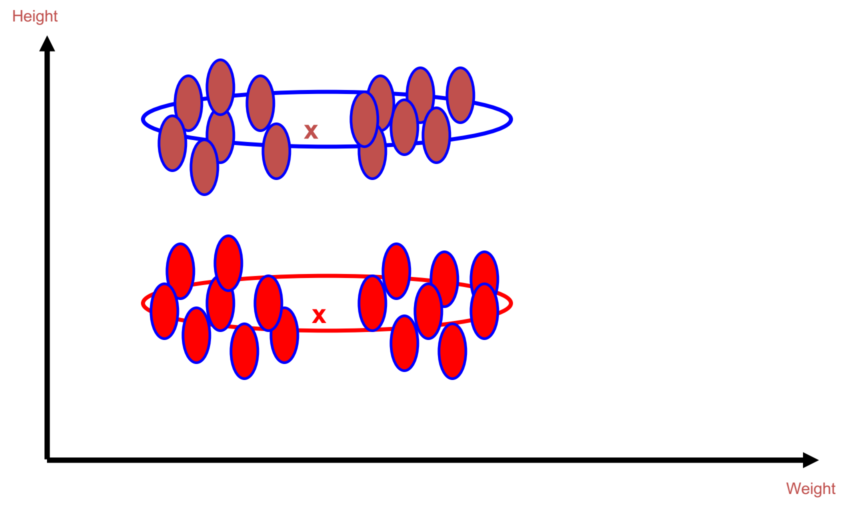
K-means Example
Re-assign Points to Clusters



K-means Example
Re-estimate Means and Converge



K-means Example
Convergence



K-means: Properties and Limitations

- The algorithm converges to a local minimum
- The solution depends on the initial partition
- One should start the algorithm with many different random choices for the initial means, and choose the solution having smallest value of the objective function

K-means: Properties and Limitations

- The algorithm is sensitive to outliers
- A variation of K-means improves upon robustness (K-medoids):
  - Centers for each cluster are restricted to be one of the points assigned to the cluster;
  - The center (*medoid*) is set to be the point that minimizes the total distance to other points in the cluster;
  - K-medoids is more computationally intensive than K-means.

K-means: Properties and Limitations

- The algorithm requires the number of clusters *K*;
- Often *K* is unknown, and must be estimated from the data:

We can test $K \in \{1, 2, \cdots, K_{\max}\}$

Compute $\{W_1, W_2, \cdots, W_{\max}\}$

In general: $W_1 > W_2 > \cdots > W_{\max}$

K^* = actual number of clusters in the data,

when $K < K^*$, we can expect $W_K >> W_{K+1}$

when $K > K^*$, further splits provide smaller decrease of W

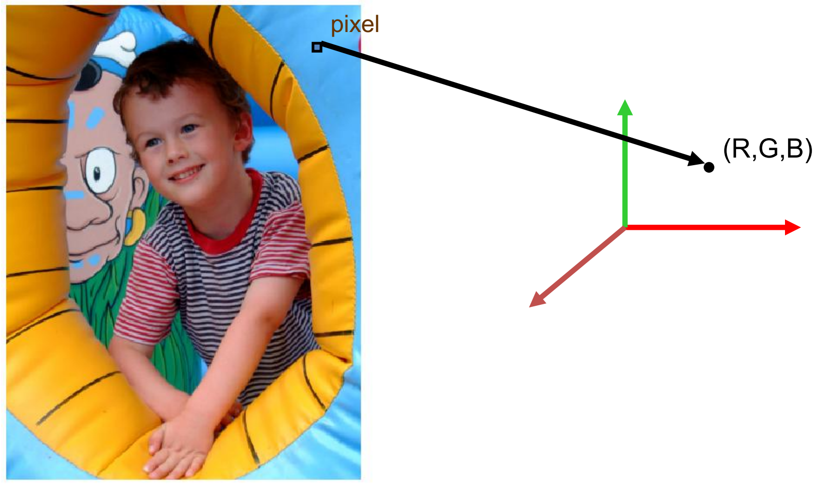
Set $\hat{K}^*$ by identifying an "elbow shape" in the plot of W_k

An Application of K-means: Image segmentation

- **Goal of segmentation**: partition an image into regions with homogeneous visual appearance (which could correspond to objects or parts of objects)
- **Image representation**: each pixel is represented as a three dimensional point in RGB space, where
  - R = intensity of red
  - G = intensity of green
  - B = intensity of blue

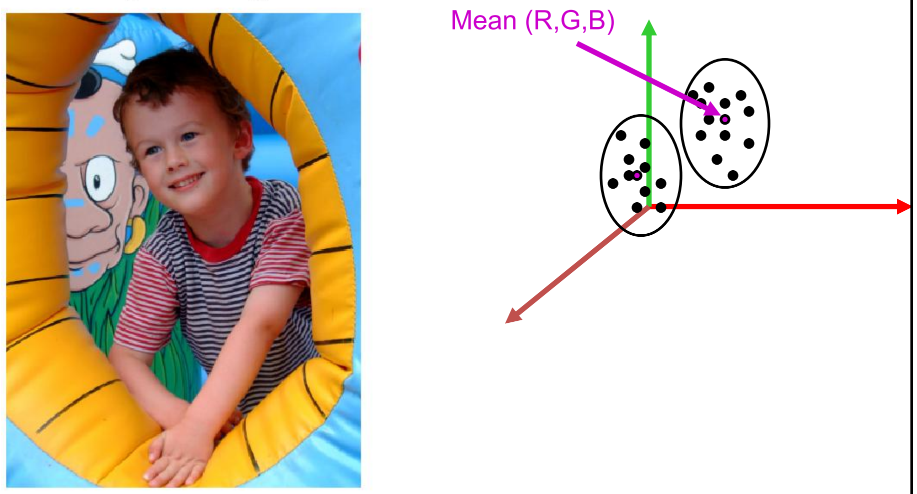
An Application of K-means: Image segmentation

Original image

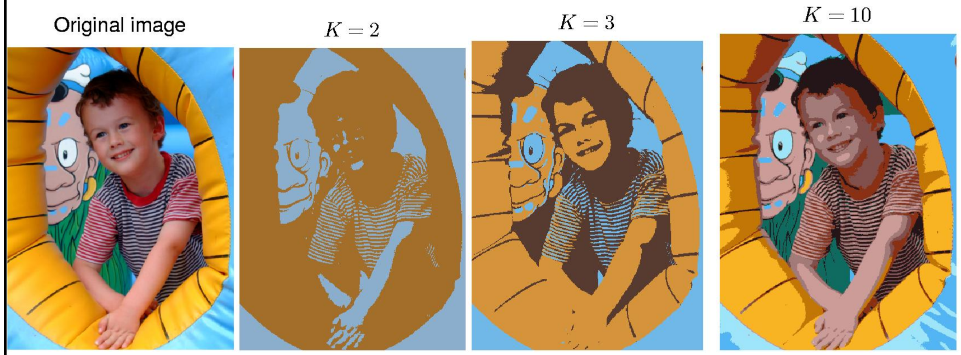


An Application of K-means: Image segmentation

Original image



An Application of K-means: Image segmentation



An Application of K-means: (Lossy) Data compression

- Original image has N pixels
- Each pixel → (R,G,B) values
- Each value is stored with 8 bits of precision
- Transmitting the whole image costs <u>24N bits</u>

<u>Compression achieved by K-means:</u>

- Identify each pixel with the corresponding centroid
- We have K such centroids → we need $\log_2 K$ bits per pixel
- For each centroid we need 24 bits
- Transmitting the whole image costs <u>$24K + N \log 2K$</u> bits
- <u>Original image</u> = 240x180=43,200 pixels → 43,200x24=1,036,800 bits
- <u>Compressed images</u>:

K=2: 43,248 bits K=3: 86,472 K=10: 173,040 bits

Applications Computer Vision Object Recognition, Image Based Retrieval

Applications of NN and clustering techniques in computer vision

Example Problems

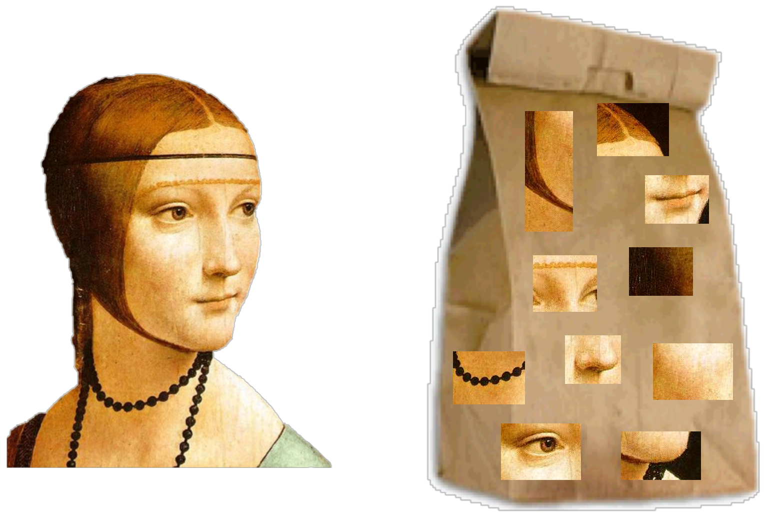
- **Problem1:** Given an image find me similar image/duplicate in the database
- **Problem 2:** Given an image, is this an image of a car ? Given database of images labeled as cars, faces, aeroplanes
- **Problem 3:** Given a document, what is the topic of the document ?
- **Problem 4:** Given an image, find me the closest image (looking most alike) in a large database

- Issues: definition of the similarity between two images, two documents (the key), deployment of clustering, large scale instance based learning

Recognition using local features

- Define a set of local feature templates (we will discuss in more detail how to do this in Computer Vision section)
- Define similarity measure between two images
- Bags of visual features models

Bag-of-features models



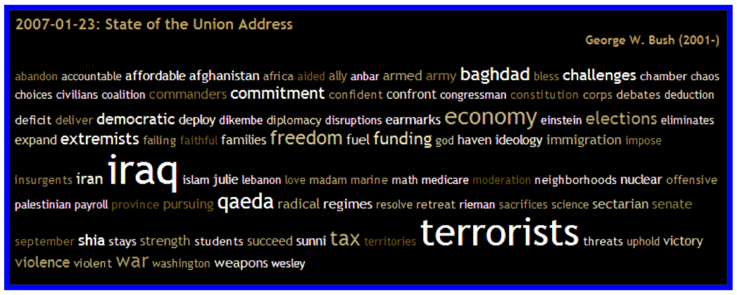
Many slides adapted from S. Lazebnik, Fei-Fei Li, Rob Fergus, and Antonio Torralba and many others

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

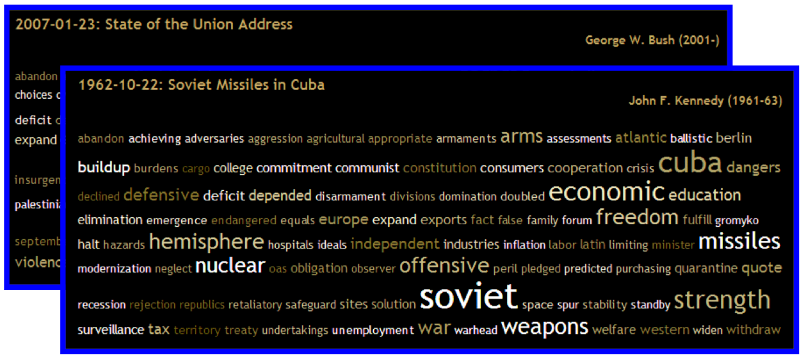
Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)



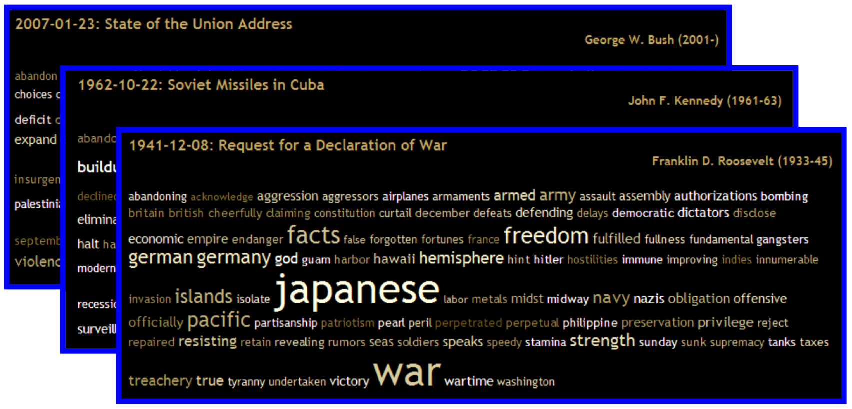
Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

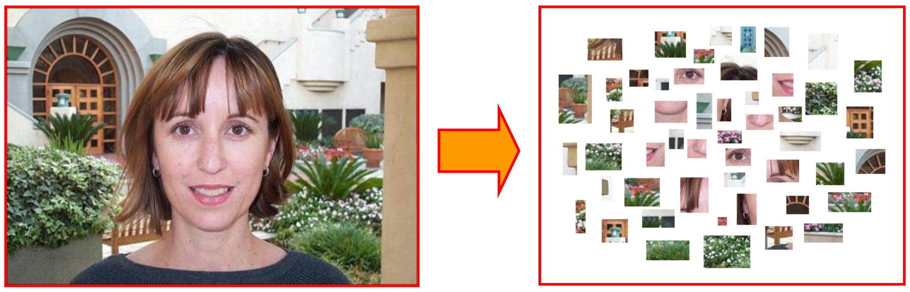

2007-01-23: State of the Union Address
George W. Bush (2001-)
1962-10-22: Soviet Missiles in Cuba
John F. Kennedy (1961-63)
abandon achieving adversaries aggression agricultural appropriate armaments arms assessments atlantic ballistic berlin
buildup burdens cargo college commitment communist constitution consumers cooperation crisis cuba dangers
declined defensive deficit depended disarmament divisions domination doubled economic education
elimination emergence endangered equals europe expand exports fact false family forum freedom fulfill gromyko
halt hazards hemisphere hospitals ideals independent industries inflation labor latin limiting minister missiles
modernization neglect nuclear oas obligation observer offensive peril pledged predicted purchasing quarantine quote
recession rejection republics retaliatory safeguard sites solution soviet space spur stability standby strength
surveillance tax territory treaty undertakings unemployment war warhead weapons welfare western widen withdraw

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)


2007-01-23: State of the Union Address
George W. Bush (2001-)
1962-10-22: Soviet Missiles in Cuba
John F. Kennedy (1961-63)
1941-12-08: Request for a Declaration of War
Franklin D. Roosevelt (1933-45)
abandoning acknowledge aggression aggressors airplanes armaments armed army assault assembly authorizations bombing
britain british cheerfully claiming constitution curtail december defeats defending delays democratic dictators disclose
economic empire endanger facts false forgotten fortunes france freedom fulfilled fullness fundamental gangsters
german germany god guam harbor hawaii hemisphere hint hitler hostilities immune improving indies innumerable
invasion islands isolate japanese labor metals midst midway navy nazis obligation offensive
officially pacific partisanship patriotism pearl peril perpetrated perpetual philippine preservation privilege reject
repaired resisting retain revealing rumors seas soldiers speaks speedy stamina strength sunday sunk supremacy tanks taxes
treachery true tyranny undertaken victory war wartime washington

Bags of features for object recognition



- Works pretty well for image-level classification

Bag of features: outline

1. Extract features

Bag of features: outline

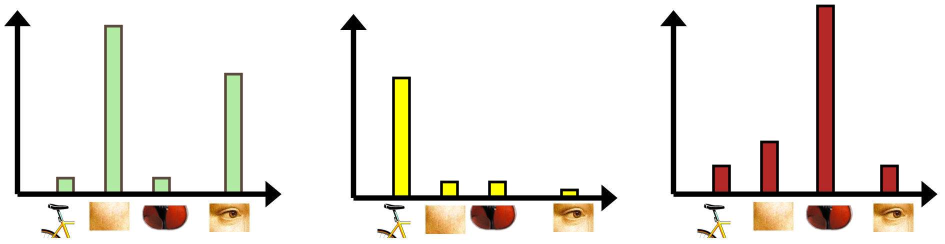
1. Extract features
2. Learn “visual vocabulary”

Bag of features: outline

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary

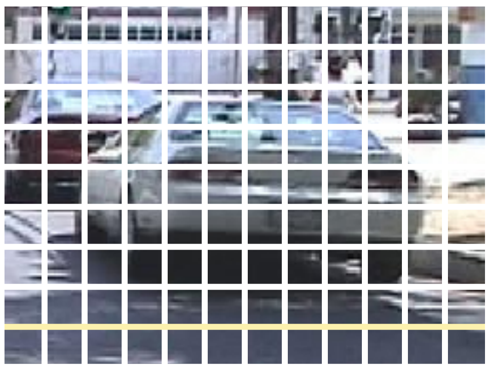
Bag of features: outline

1. Extract features
2. Learn "visual vocabulary"
3. Quantize features using visual vocabulary
4. Represent images by frequencies of "visual words"



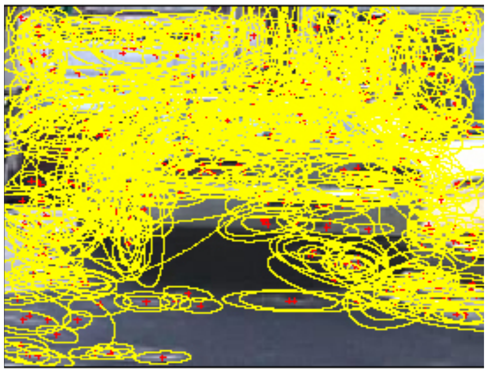
1. Feature extraction

- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005

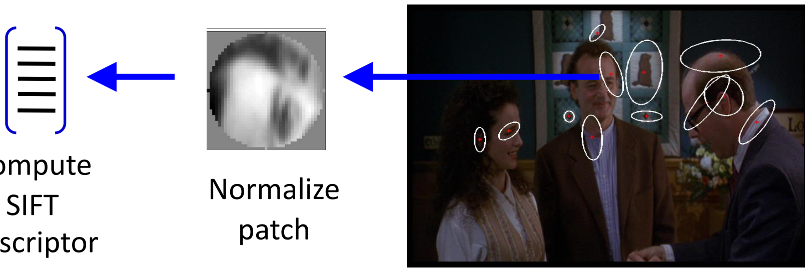


1. Feature extraction

- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005



1. Feature extraction



Detect patches

[Lowe'99]

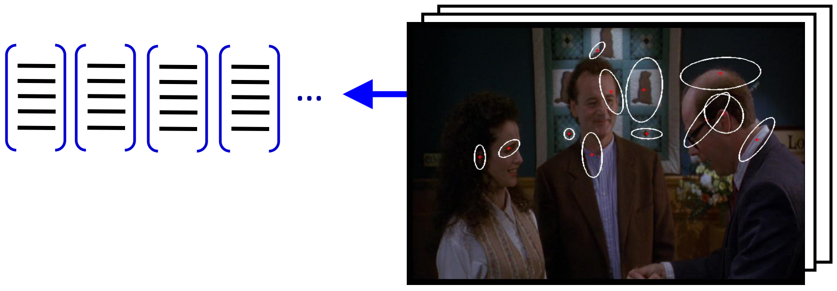
[Mikojaczyk and Schmid '02]

[Mata, Chum, Urban & Pajdla, '02]

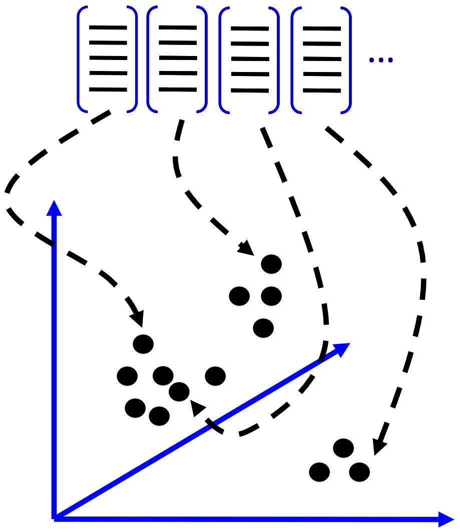
[Sivic & Zisserman, '03]

Slide credit: Josef Sivic

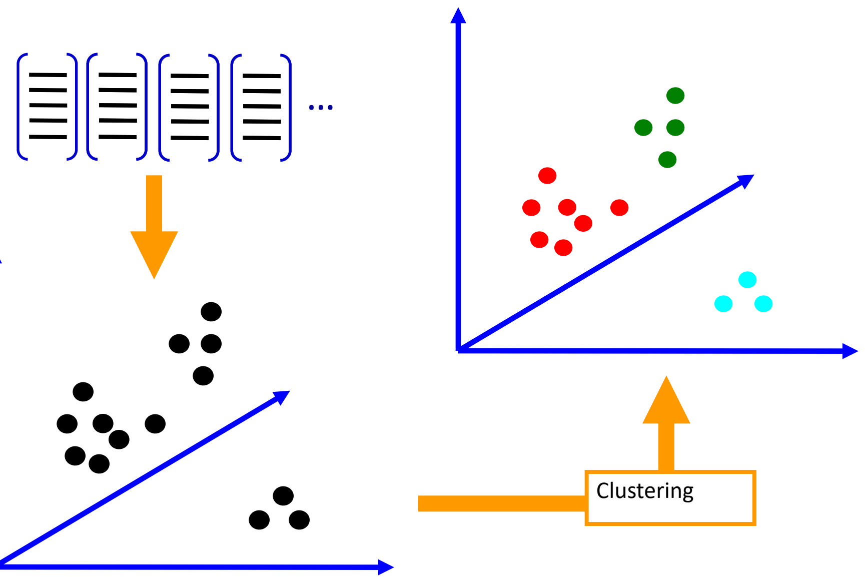
1. Feature extraction



2. Learning the visual vocabulary

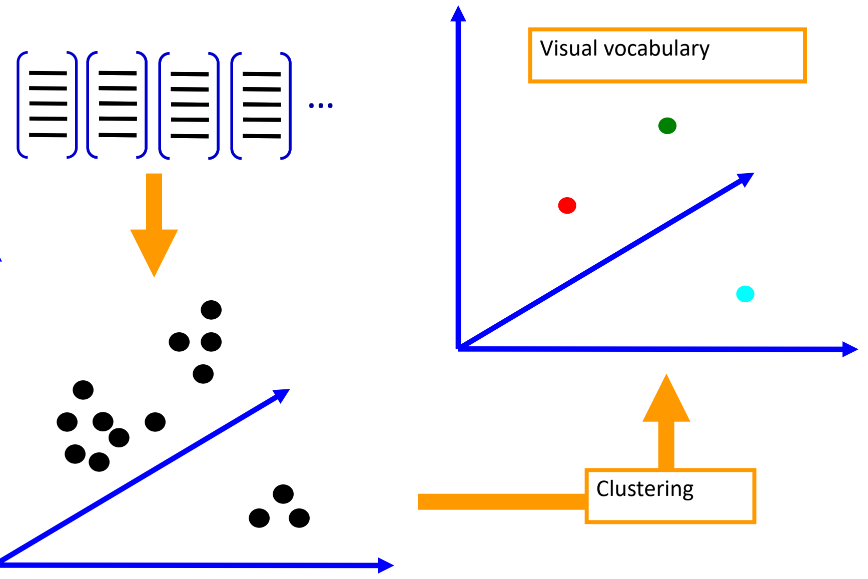


2. Learning the visual vocabulary



Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

K-means clustering

- Want to minimize sum of squared Euclidean distances between points x_i and their nearest cluster centers m_k

$$D(X,M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (x_i - m_k)^2$$

- Algorithm:
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

Expectation Maximization

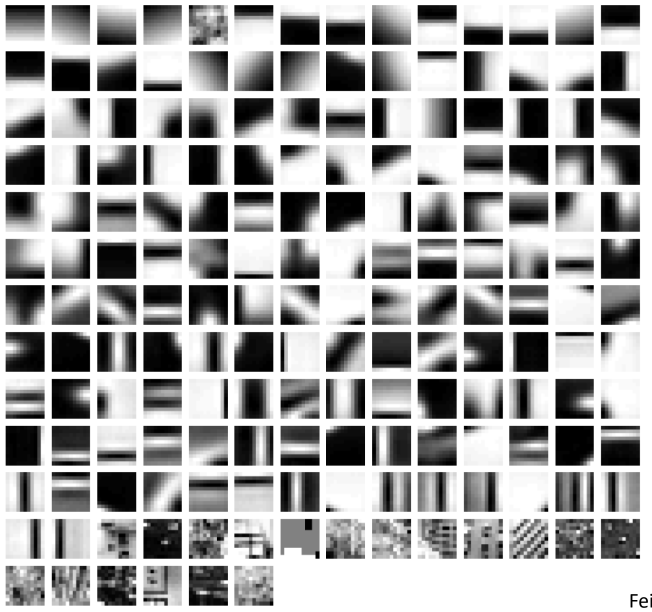
- Probabilistic version of k-means clustering
- Soft assignments of points to clusters
- Weighted recomputation of cluster centers
- Probabilistic formulation (20.3)

- Blackboard

From clustering to vector quantization

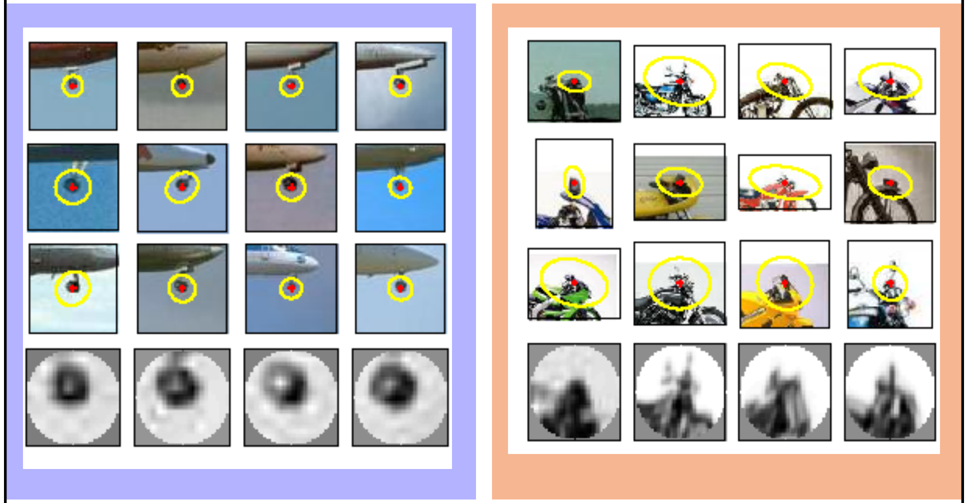
- Clustering is a common method for learning a visual vocabulary or codebook
  – Unsupervised learning process
  – Each cluster center produced by k-means becomes a codevector
  – Codebook can be learned on separate training set
  – Provided the training set is sufficiently representative, the codebook will be "universal"
- The codebook is used for quantizing features
  – A *vector quantizer* takes a feature vector and maps it to the index of the nearest codevector in a codebook
  – Codebook = visual vocabulary
  – Codevector = visual word

Example visual vocabulary



Fei-Fei et al. 2005

Image patch examples of visual words



Sivic et al. 2005

Visual vocabularies: Issues

- How to choose vocabulary size?
  - Too small: visual words not representative of all patches
  - Too large: quantization artifacts, overfitting
- Generative or discriminative learning?
- Computational efficiency
  - Vocabulary trees (Nister & Stewenius, 2006)

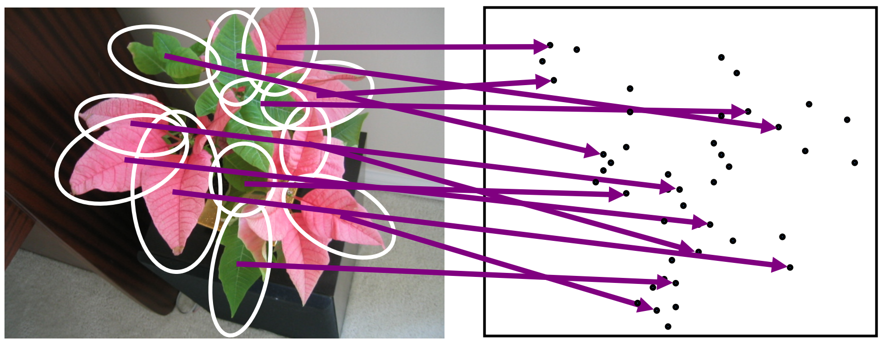
Hierarchical k-means

- We have many, many of these features
- 100000 images ~1000 features per image
- If we can get repeatable, discriminative features,
- then recognition can scale to very large databases
- using the vocabulary tree and indexing approach

- Quantize the feature descriptor space + efficient search
- Hierarchical k-means - Nister&Stewenius [CVPR 2006]
- Visual vocabulary trees

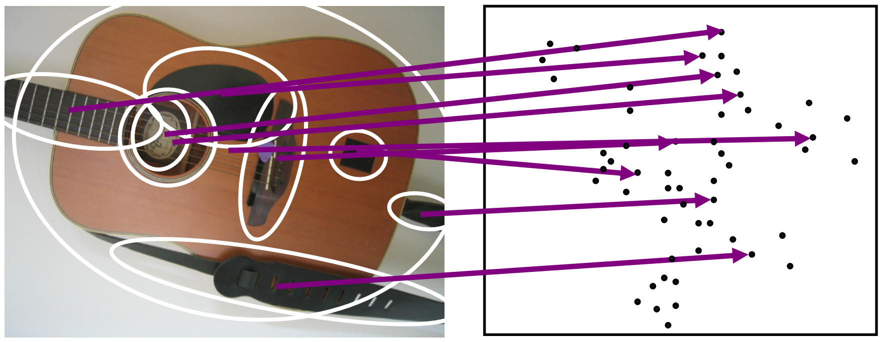
Slides from Nister & Stewenius 06

Building Visual Vocabulary Tree



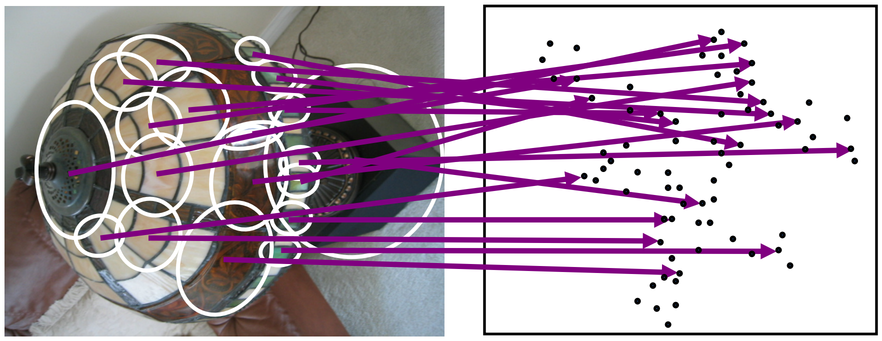
Slides from Nister & Stewenius 06

Building Visual Vocabulary Tree



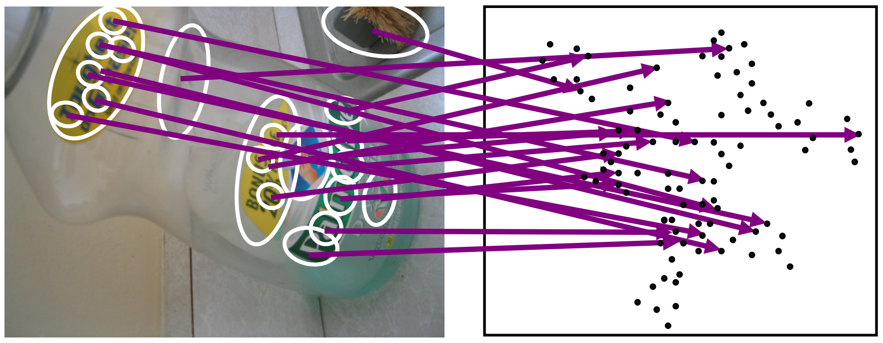
Slides from Nister & Stewenius 06

Building Visual Vocabulary Tree



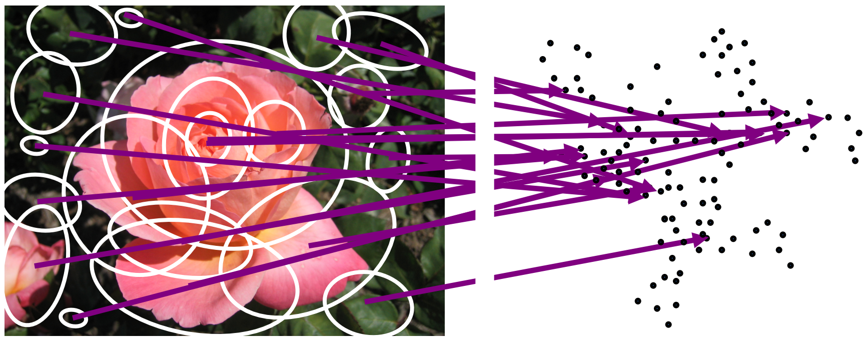
Slides from Nister & Stewenius 06

Building Visual Vocabulary Tree

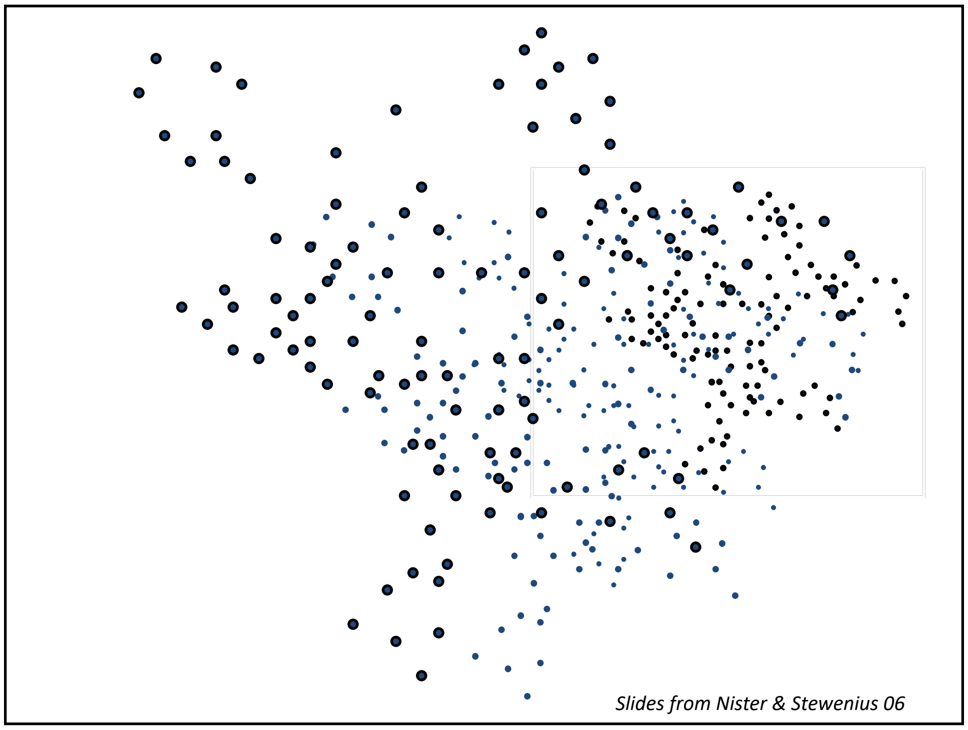


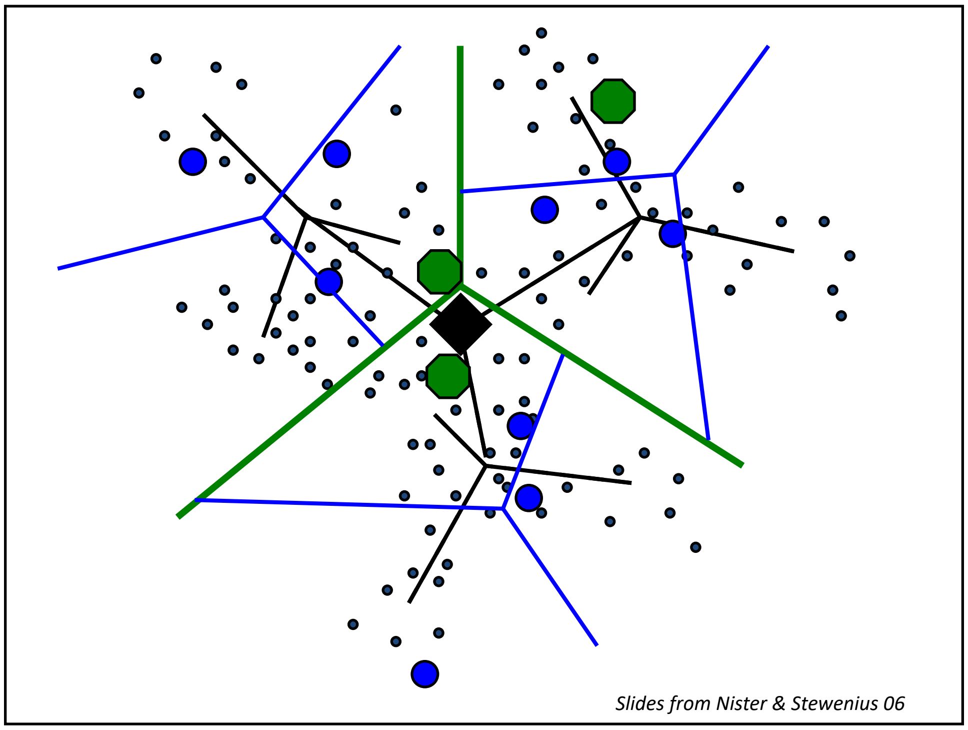
Slides from Nister & Stewenius 06

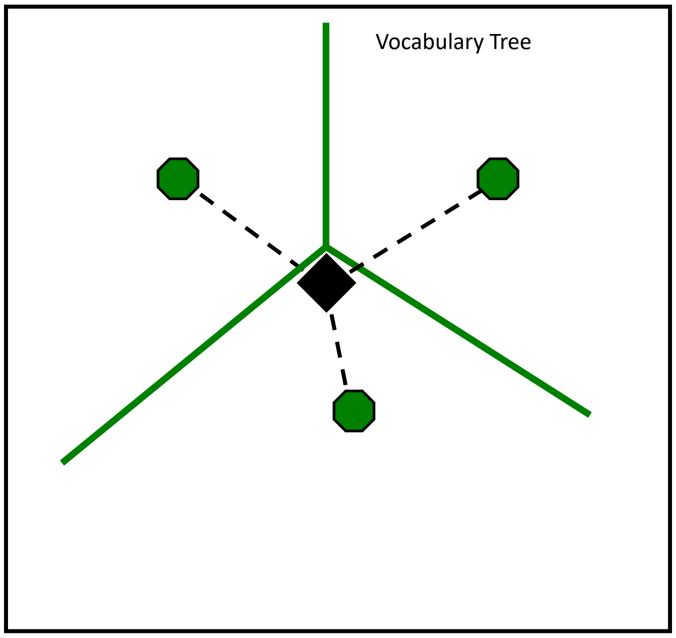
Building Visual Vocabulary Tree



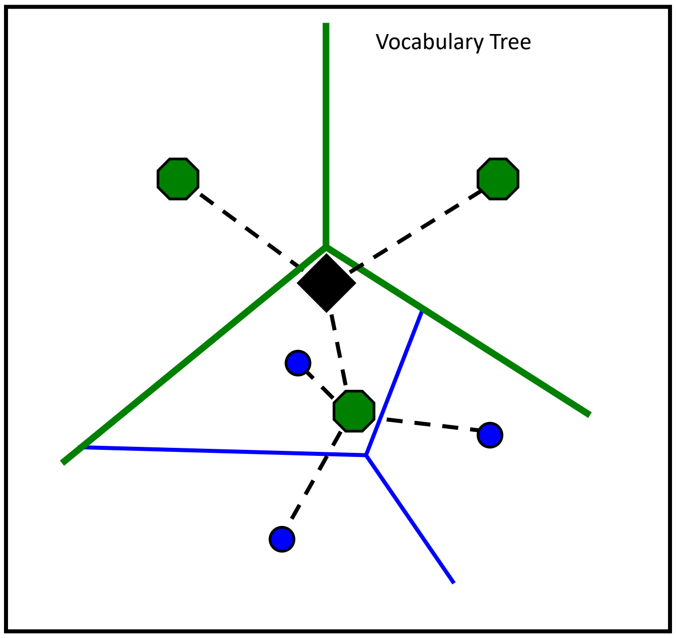
Slides from Nister & Stewenius 06


Slides from Nister & Stewenius 06

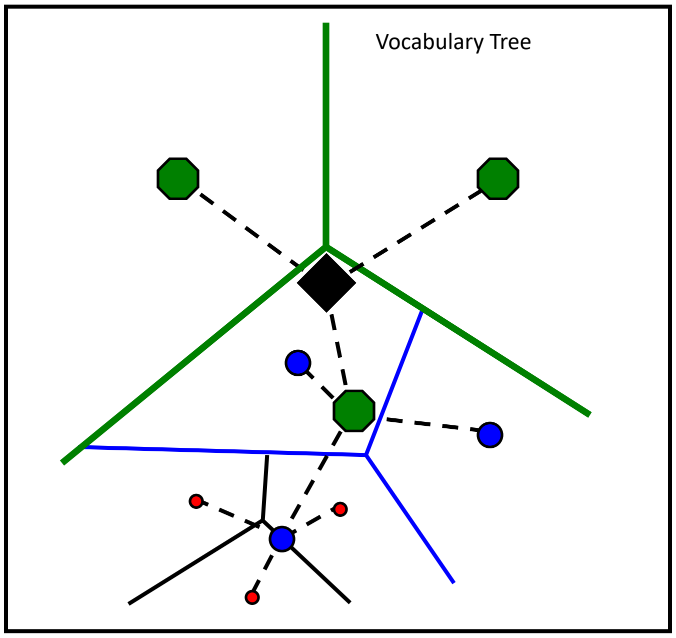

Slides from Nister & Stewenius 06



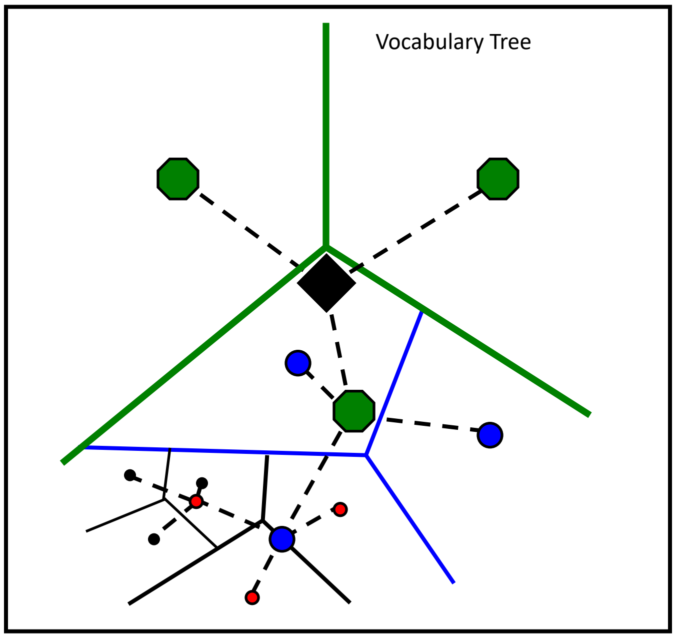
Slides from Nister & Stewenius 06



Slides from Nister & Stewenius 06



Slides from Nister & Stewenius 06



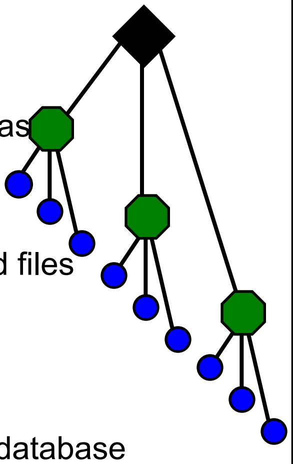
Slides from Nister & Stewenius 06

Vocabulary Trees

- Easy to add/remove images from the database
- Suitable for incremental approach
- Suitable for creating single generic vocabulary
-
- **Approach**
- Extract descriptors from many/many images
- Acquire enough statistics about the descriptor distribution
- Run k-means hierarchically k- is the branching factor of the tree
- E.g. Branching factor of 10 and 6 levels – million leaves

Slides from Nister & Stewenius 06

Vocabulary Trees

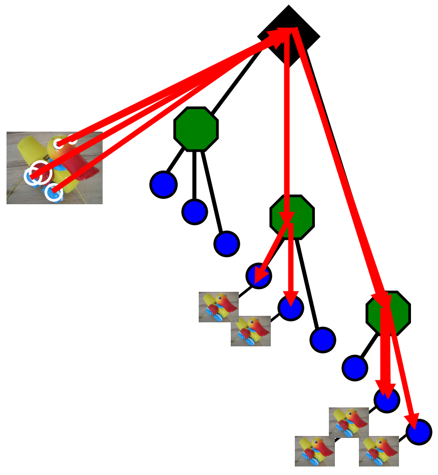


- Training phase – add images to the databas
- Extract descriptors – drop it down the tree
- Each node has an inverted file index
- Index to that image is added to all inverted files

- When we want to query image
- Pass each descriptor down the tree
- Accumulate scores for each image in the database

- At each level do k dot products total of kL dot products
- For k^L leafs and integer descriptors we need Dk^L bytes for 1M leaf
- nodes use 143 MB of memory

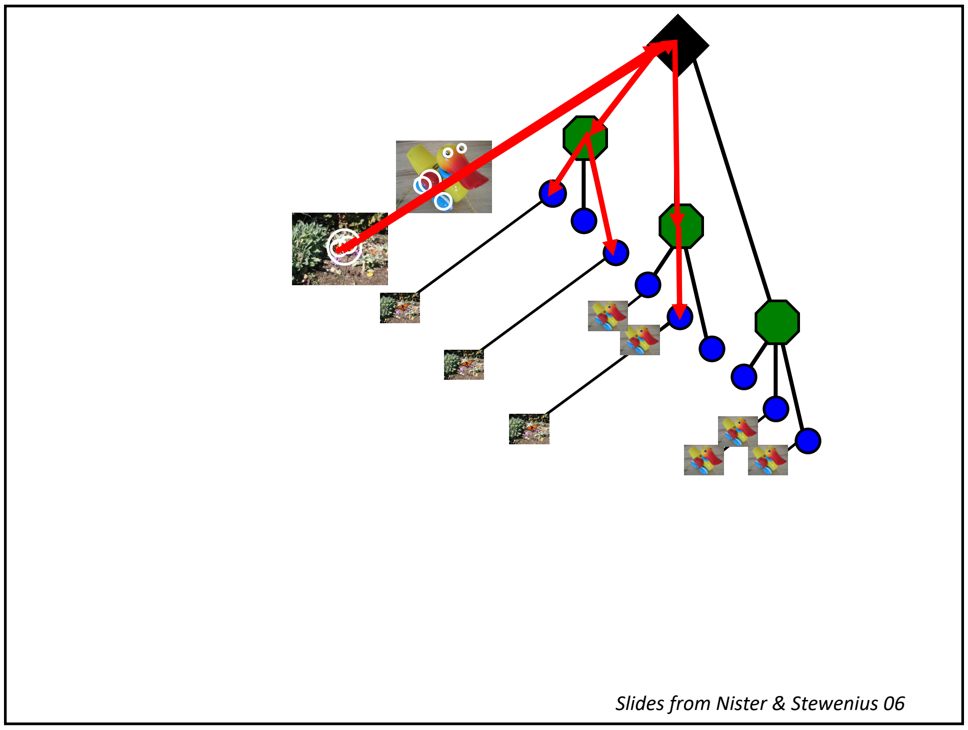
Slides from Nister & Stewenius 06



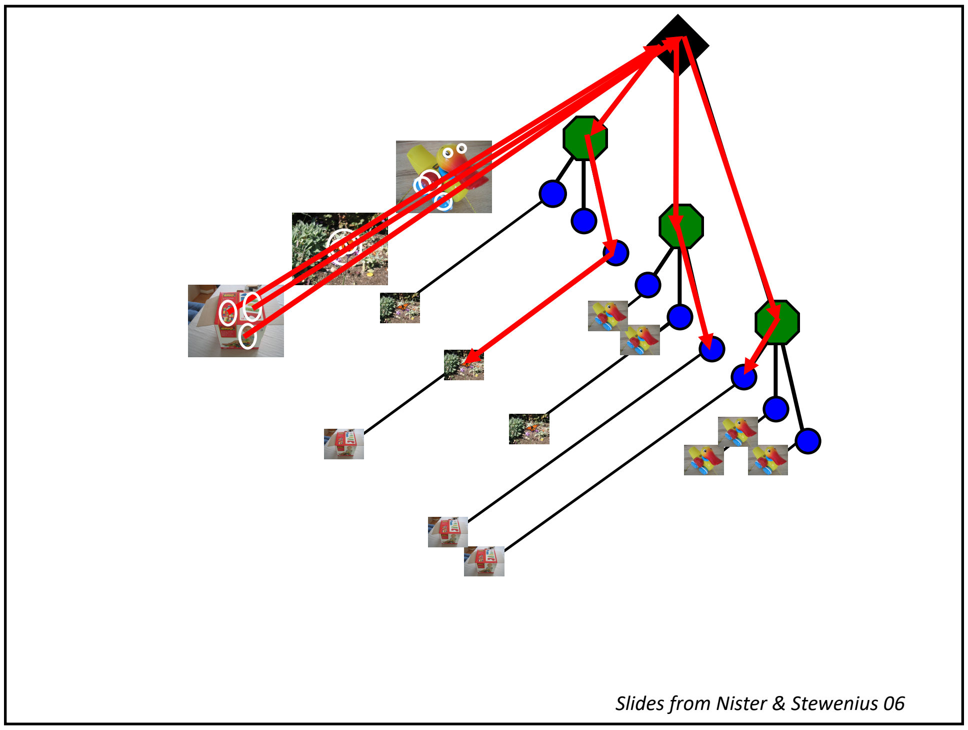
Slides from Nister & Stewenius 06

Application

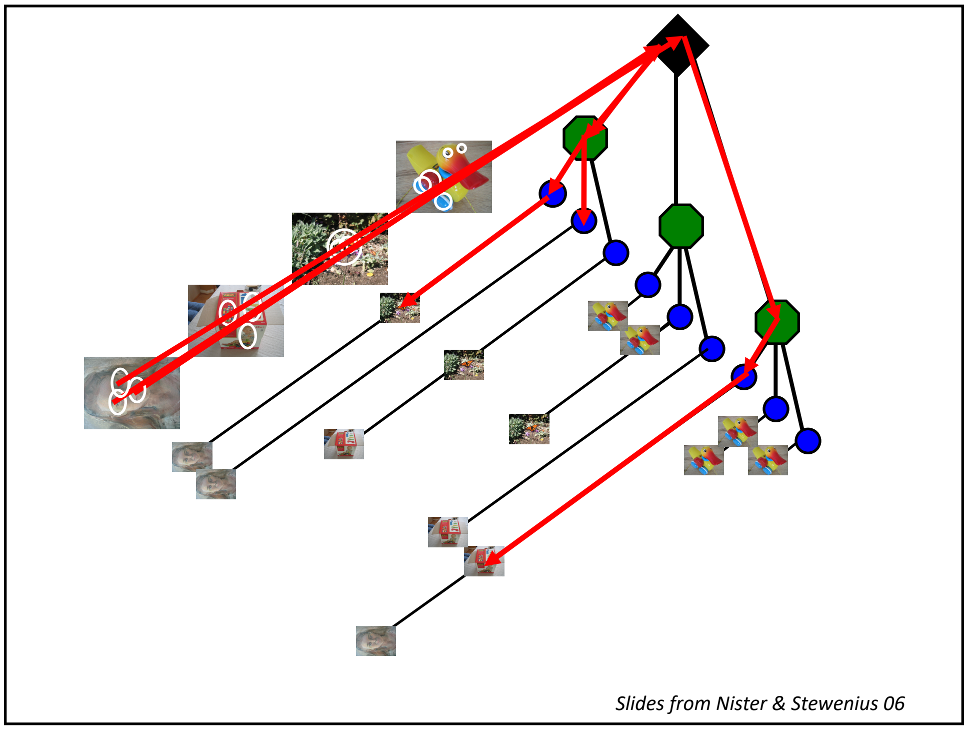
- Vocabulary trees for finding closest image in the database



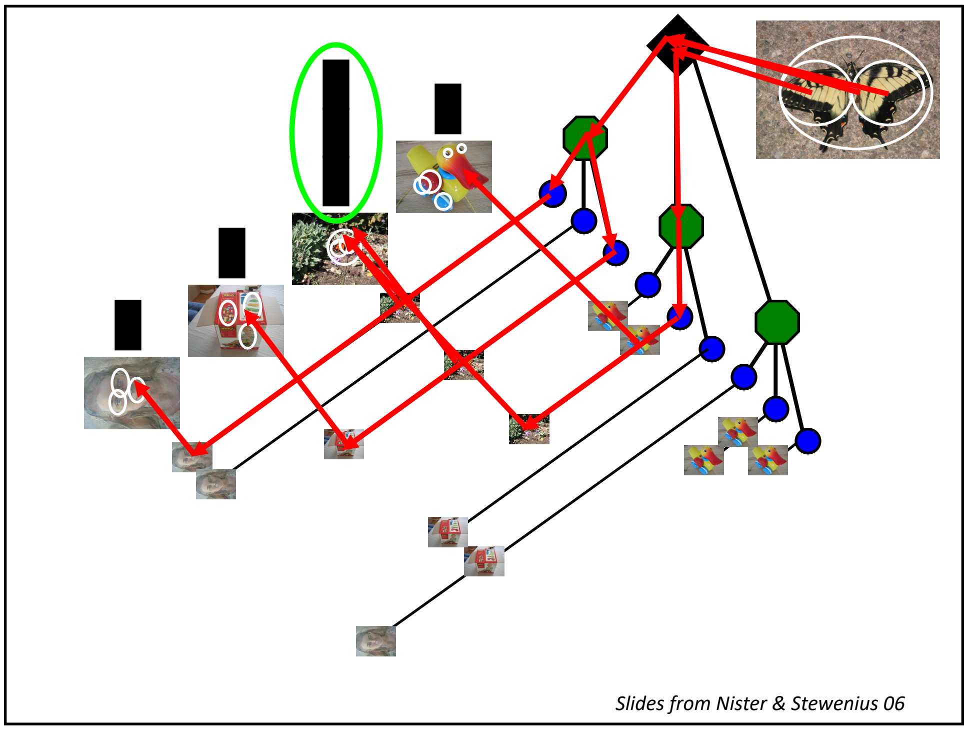
Slides from Nister & Stewenius 06



Slides from Nister & Stewenius 06



Slides from Nister & Stewenius 06



Slides from Nister & Stewenius 06

TF-IDF scoring

- TF-IDF term frequency – inverse document frequency
- Used in the information retrieval and text mining
- To evaluate how important is a word to document

- Importance depends on how many times the word appears in document – offset by number of occurrence of that word in the whole document corpus
- In image based retrieval
- image ~ document analogy
- visual word ~ word analogy

TF-IDF scoring

- TF-IDF term frequency – inverse document frequency
- Number of occurrences of a word in a document / number of occurrences of all words in the document

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \qquad |d : t_i \in d|$$

- Number of documents / number of documents where term appears

$$\mathrm{idf}_{i,j} = \log \frac{|D|}{|\{d : t_i \in d\}|} \qquad |D|$$

- High weight of a word/term is when it has high frequency and low term document frequency

$$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$$

Inverted file index

- Idea – large number of images (documents) – how to find out quickly which image does the word occur ?
- Forward index – list of words per document
- Inverted index – list of documents per word
- Once the word is encountered – we can quickly figure out which image it belongs to

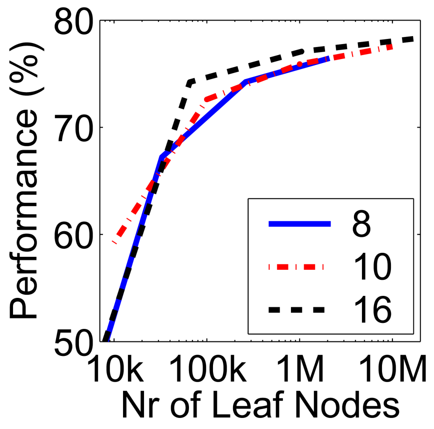
Word 1	{1, 2 , 10, 12}
Word 2	{0, 2, 100, 7, 18}
Word 3	{5, 10, 12}
…	

Size Matters

Performance improves with the
Size of the database

Improves
Retrieval

Improves
Speed

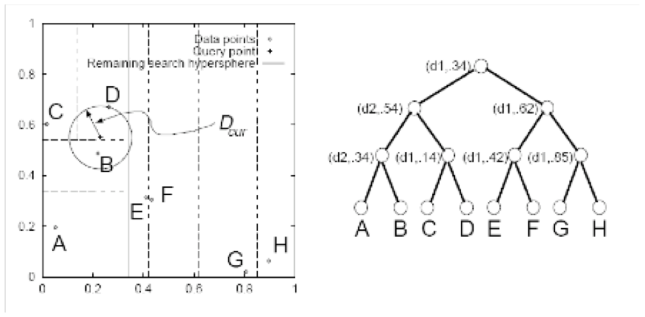


Here the results of particular object instance retrieval, database
Of ~ 40,000 objects, real-time performance

Slides from Nister & Stewenius 06

Alternative NN search

- K-d trees with best bin first
- Split the space along different dimensions recursively
- Each split has dimension and split value
- Balance the tree and have small number of levels
- Query time – locate the bin and and search nearby bins



Application

- Image/Text categorization using bags of (visual) words models

Bags of features for object recognition

Circa 2001: 5 categories, 100s of images per category
Circa 2004: 101 categories
Today: thousands of categories, tens of thousands of images

class	bag of features	bag of features	Parts-and-shape model
	Zhang et al. (2005)	Willamowski et al. (2004)	Fergus et al. (2003)
airplanes	**98.8**	97.1	90.2
cars (rear)	98.3	**98.6**	90.3
cars (side)	**95.0**	87.3	88.5
faces	**100**	99.3	96.4
motorbikes	**98.5**	98.0	92.5
spotted cats	**97.0**	—	90.0