CMPSCI 670: Computer Vision

Image representation

University of Massachusetts, Amherst
November 3, 2014

Instructor: Subhransu Maji

Slides credit S. Lazebnik, J. Civic and others

Administrivia

- This week's office hours are today after class
  - Canceling Wednesday's office hours because …

Distinguished Lecturer Series

Maneesh Agrawala
University of California, Berkeley
EECS Department

Wednesday, November 5, 2014
4:00pm - 5:00pm
Computer Science Building, Room 151
Faculty Host: Evangelos Kalogerakis

"Storytelling Tools"

Storytelling is essential for communicating ideas. When they are well told, stories help us make sense of information, appreciate cultural or societal differences, and imagine living in entirely different worlds. Audio/visual stories in the form of radio programs, books-on-tape, podcasts, television, movies and animations, are especially powerful because they provide a rich multisensory experience. Technological advances have made it easy to capture stories using the microphones and cameras that are readily available in our mobile devices, But, the raw media rarely tells a compelling story.

- Homework 4 due on Wednesday

Lecture outline

- Origin and motivation of the “bag of words” model
- Algorithm pipeline
  - Extracting local features
  - Learning a dictionary — clustering using k-means
  - Encoding methods — hard vs. soft assignment
  - Spatial pooling — pyramid representations
  - Similarity functions and classifiers

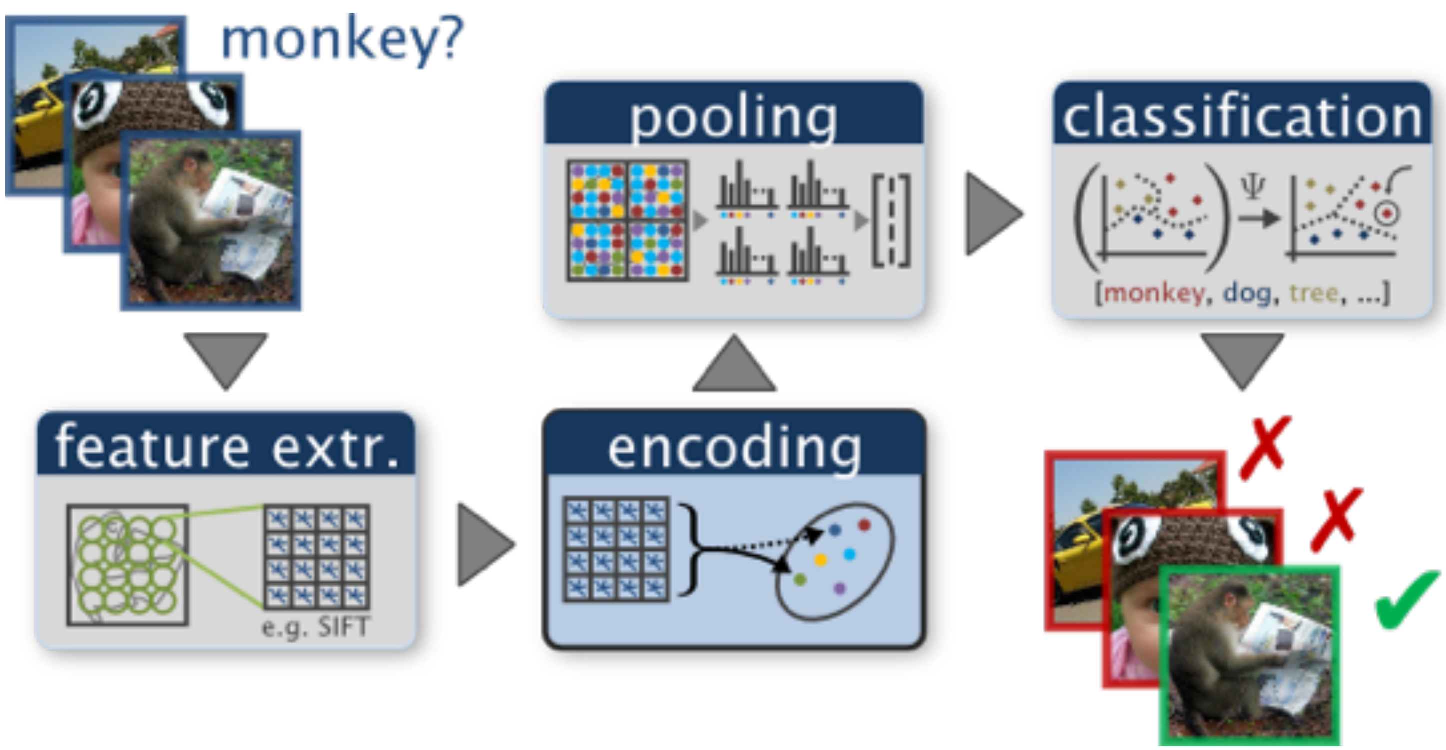
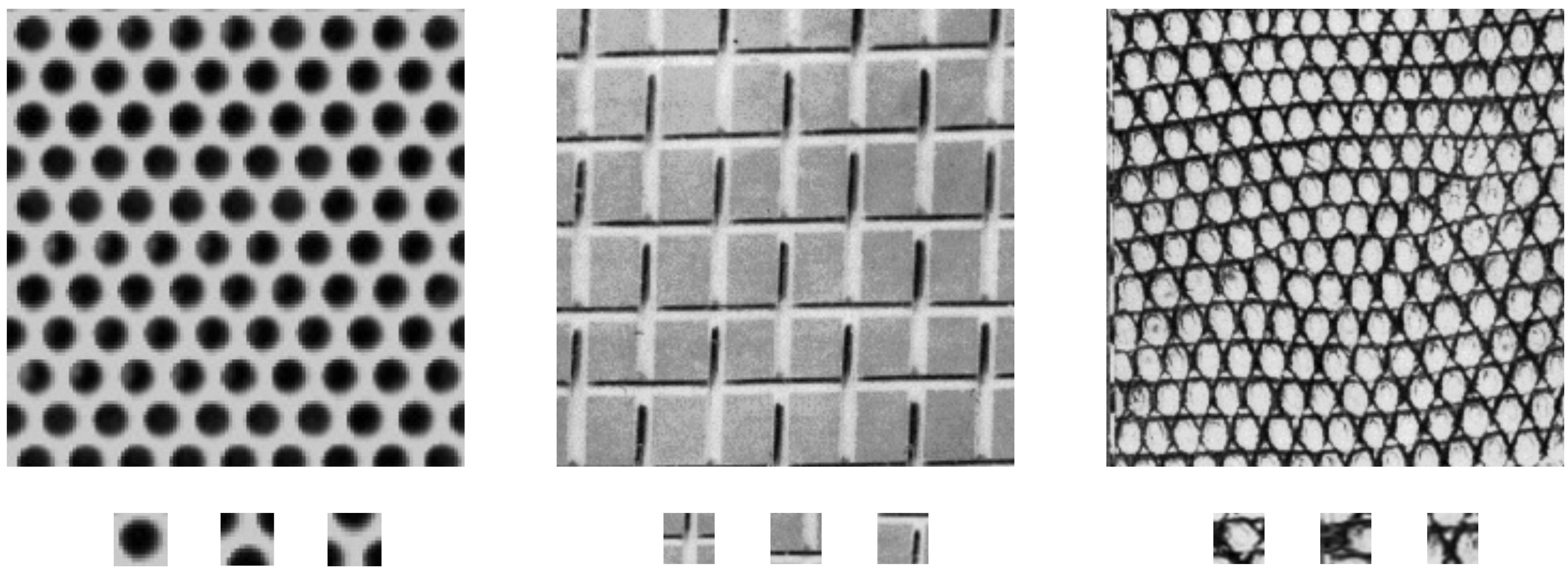


Figure from *Chatfield et al.,2011*

Bag of features

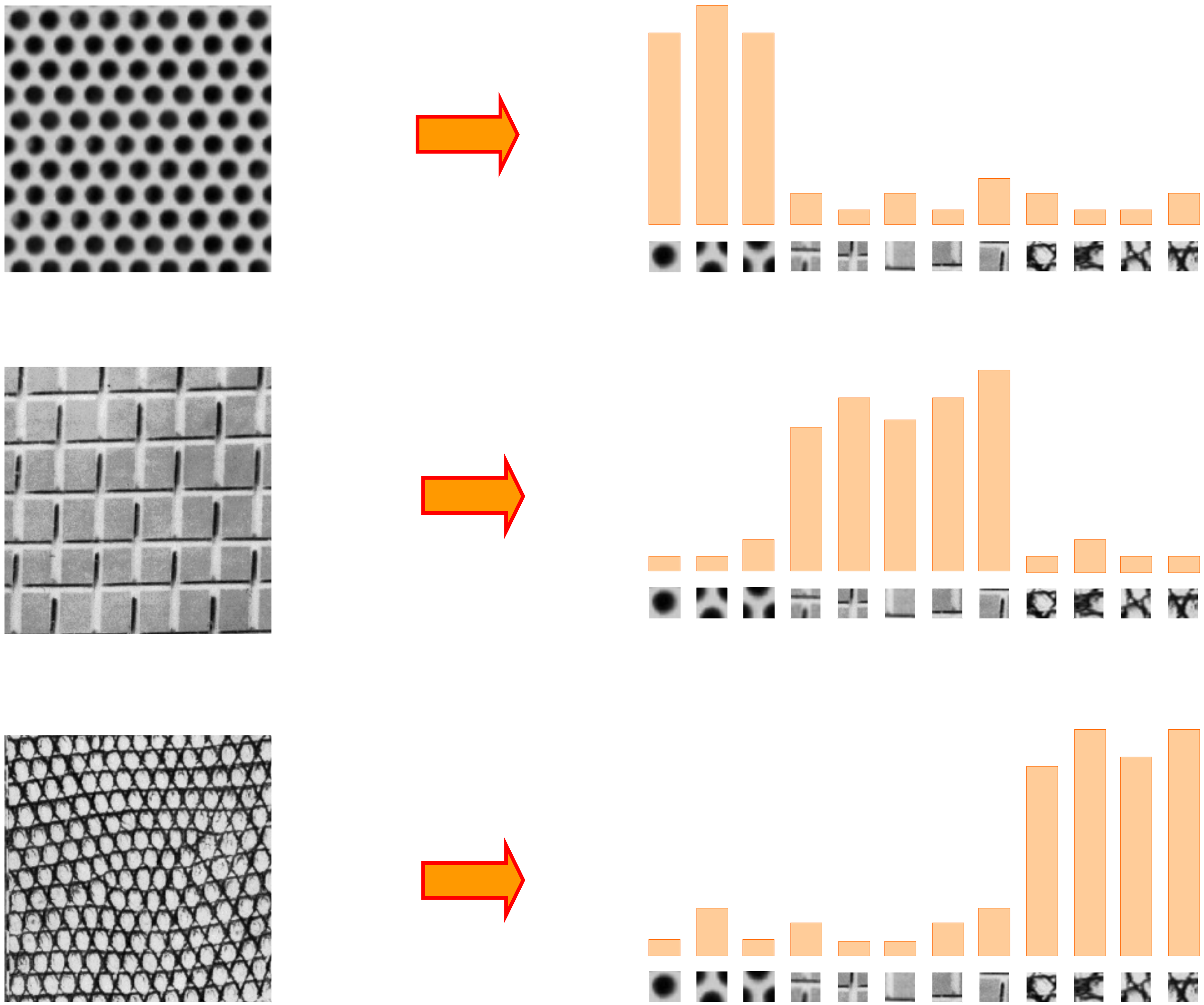
Origin 1: Texture recognition

- Texture is characterized by the repetition of basic elements or *textons*
- For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Origin 1: Texture recognition



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Origin 2: Bag-of-words models

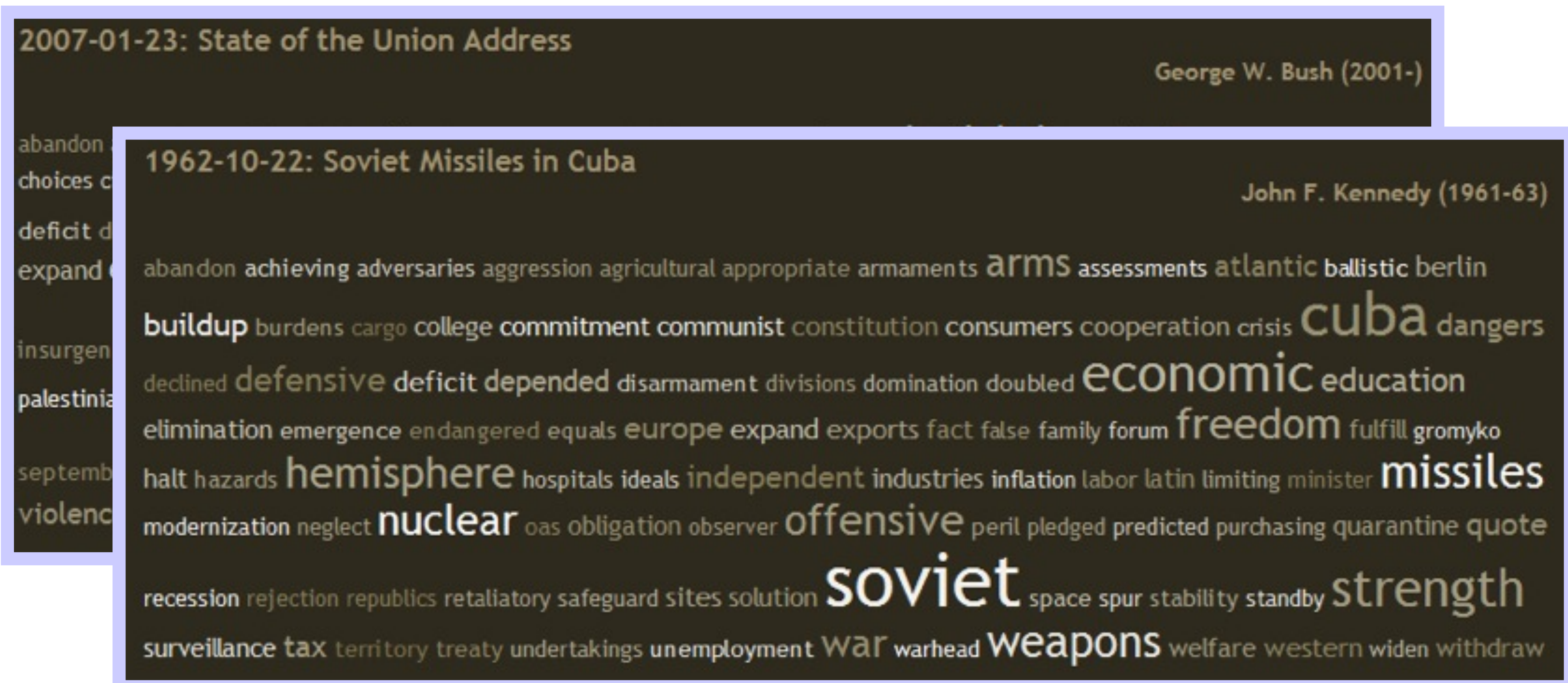
- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Origin 2: Bag-of-words models

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

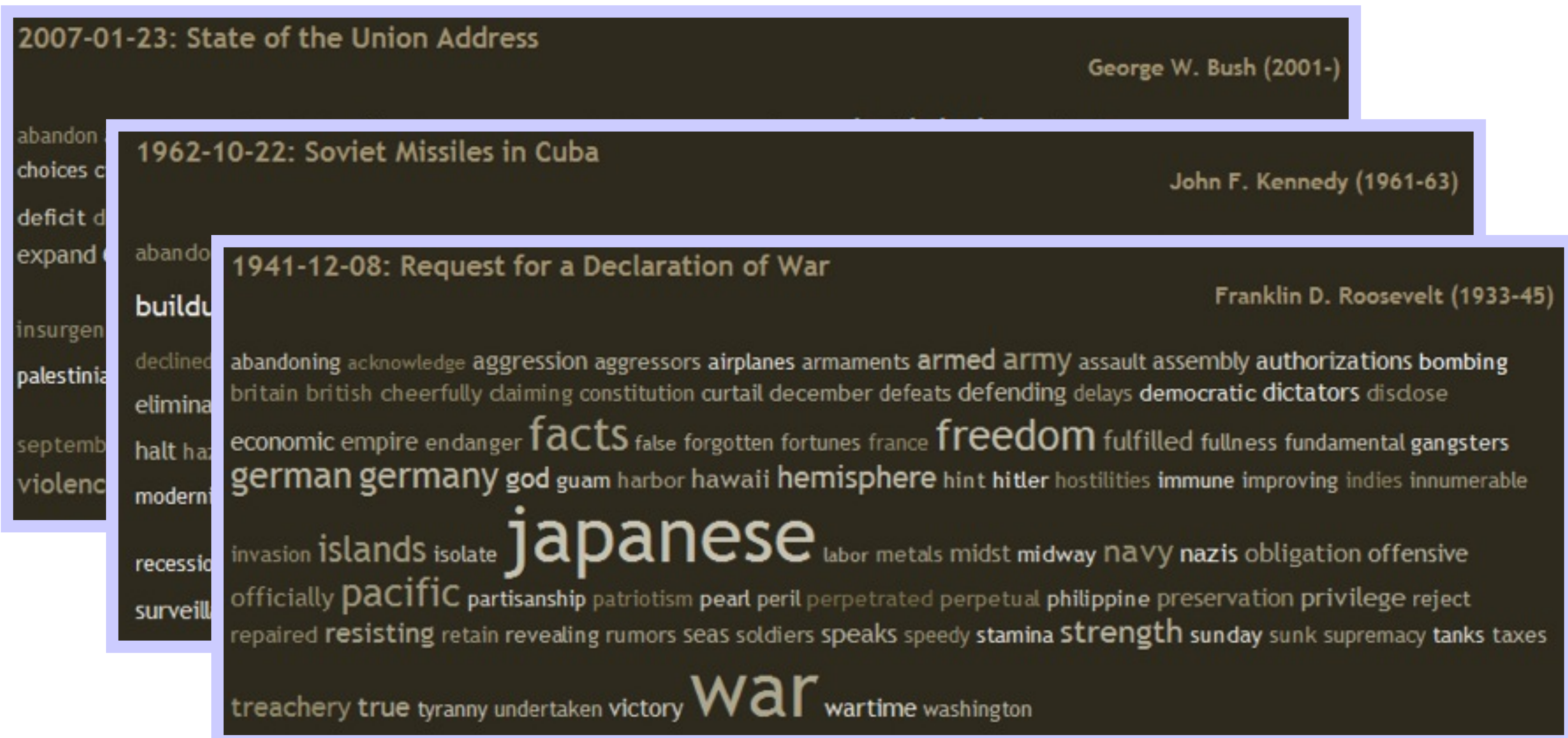
Origin 2: Bag-of-words models

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



Origin 2: Bag-of-words models

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



Lecture outline

- Origin and motivation of the “bag of words” model
- Algorithm pipeline
  - Extracting local features
  - Learning a dictionary — clustering using k-means
  - Encoding methods — hard vs. soft assignment
  - Spatial pooling — pyramid representations
  - Similarity functions and classifiers

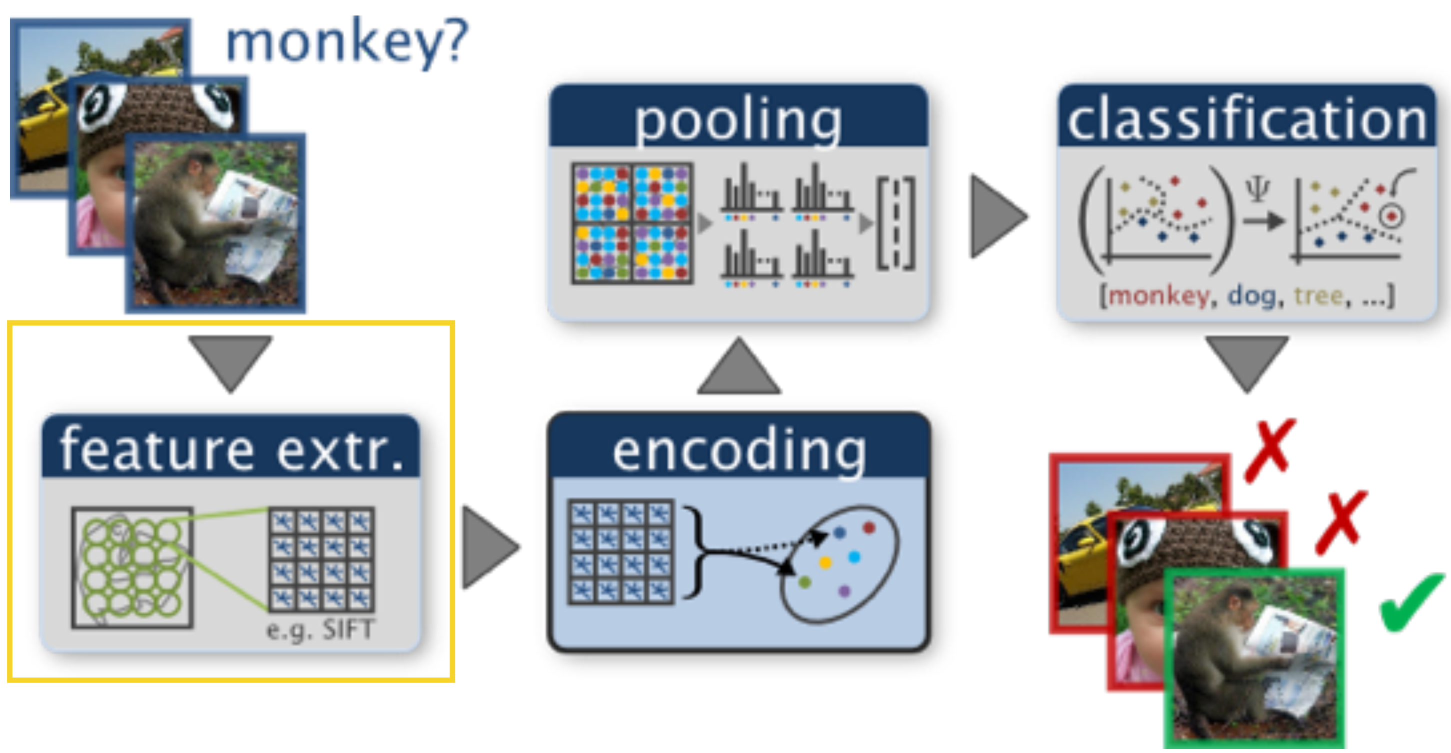
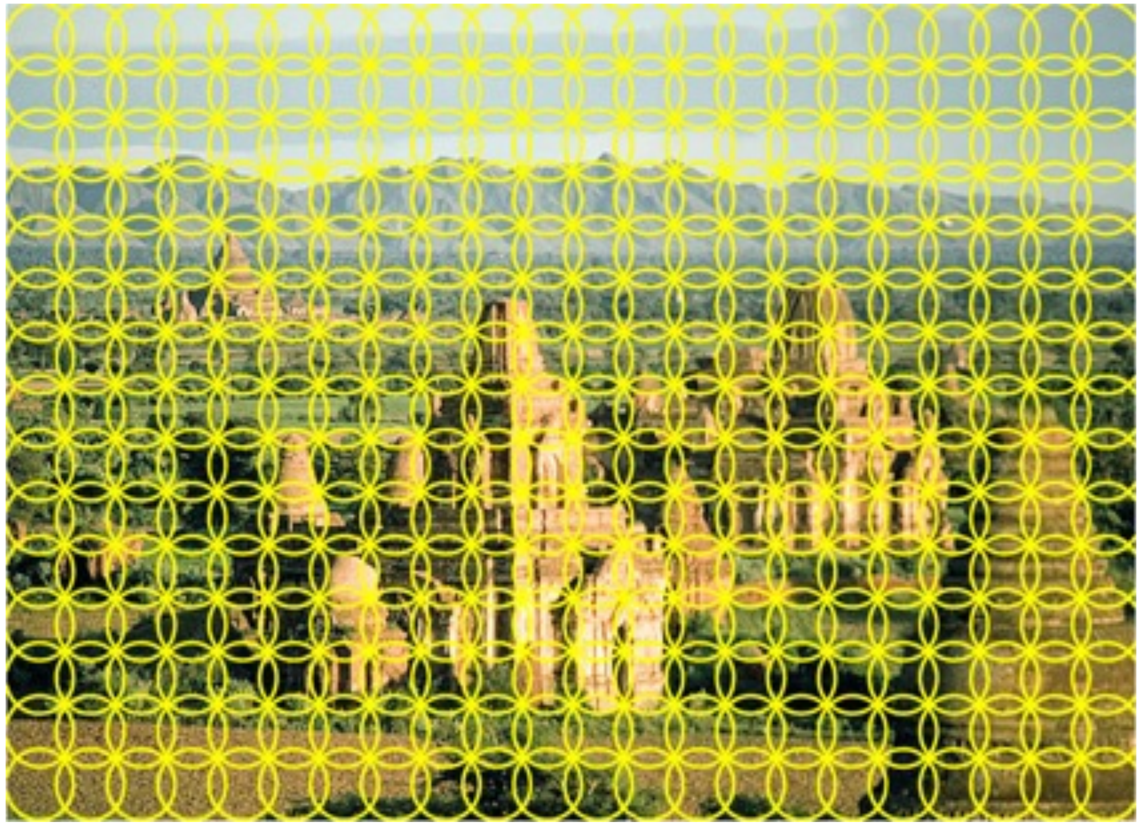


Figure from *Chatfield et al.,2011*

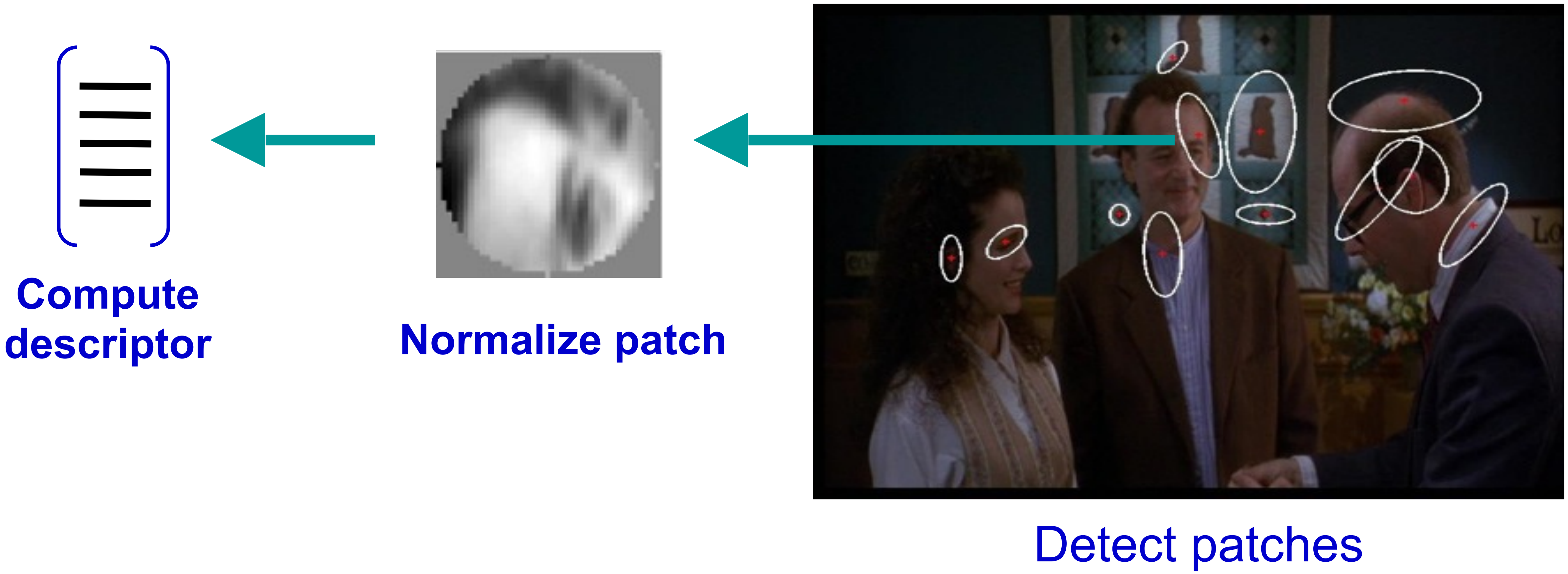
Local feature extraction

- Regular grid or interest regions



blob detector

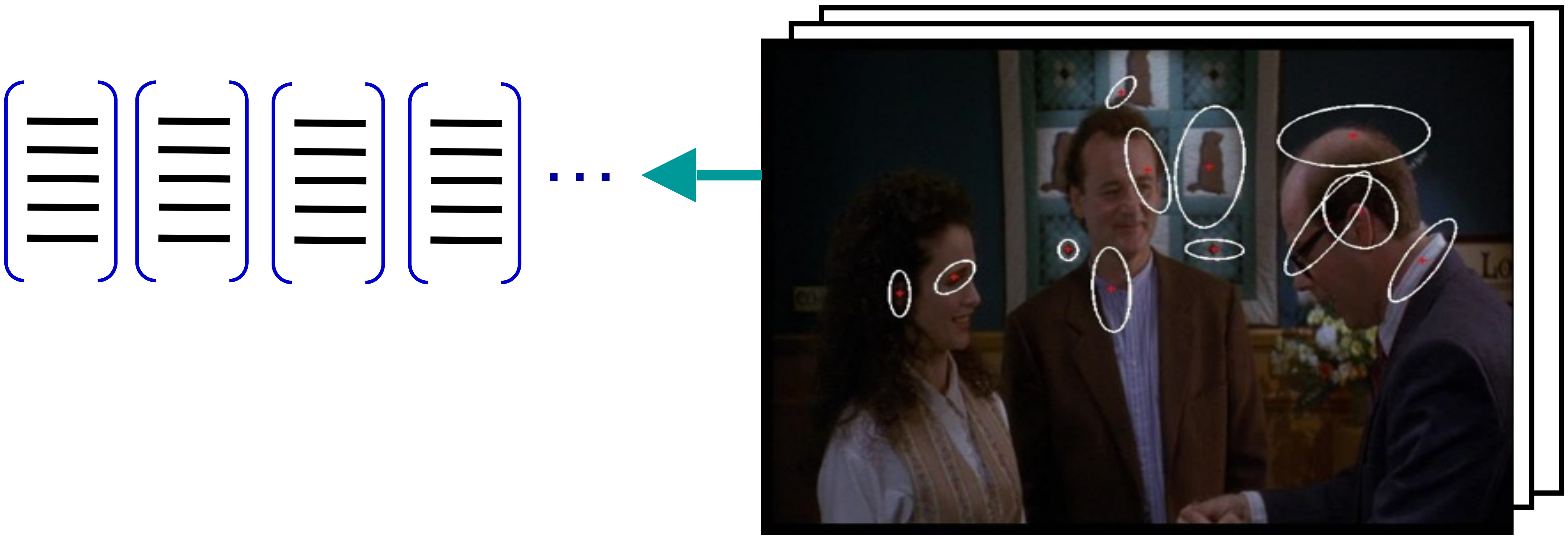
Local feature extraction



Choices of descriptor:

- SIFT
- Filterbank histograms
- The patch itself

Local feature extraction



Extract features from many images

Slide credit: Josef Sivic

Lecture outline

- Origin and motivation of the "bag of words" model
- Algorithm pipeline
  - Extracting local features
  - Learning a dictionary — clustering using k-means
  - Encoding methods — hard vs. soft assignment
  - Spatial pooling — pyramid representations
  - Similarity functions and classifiers

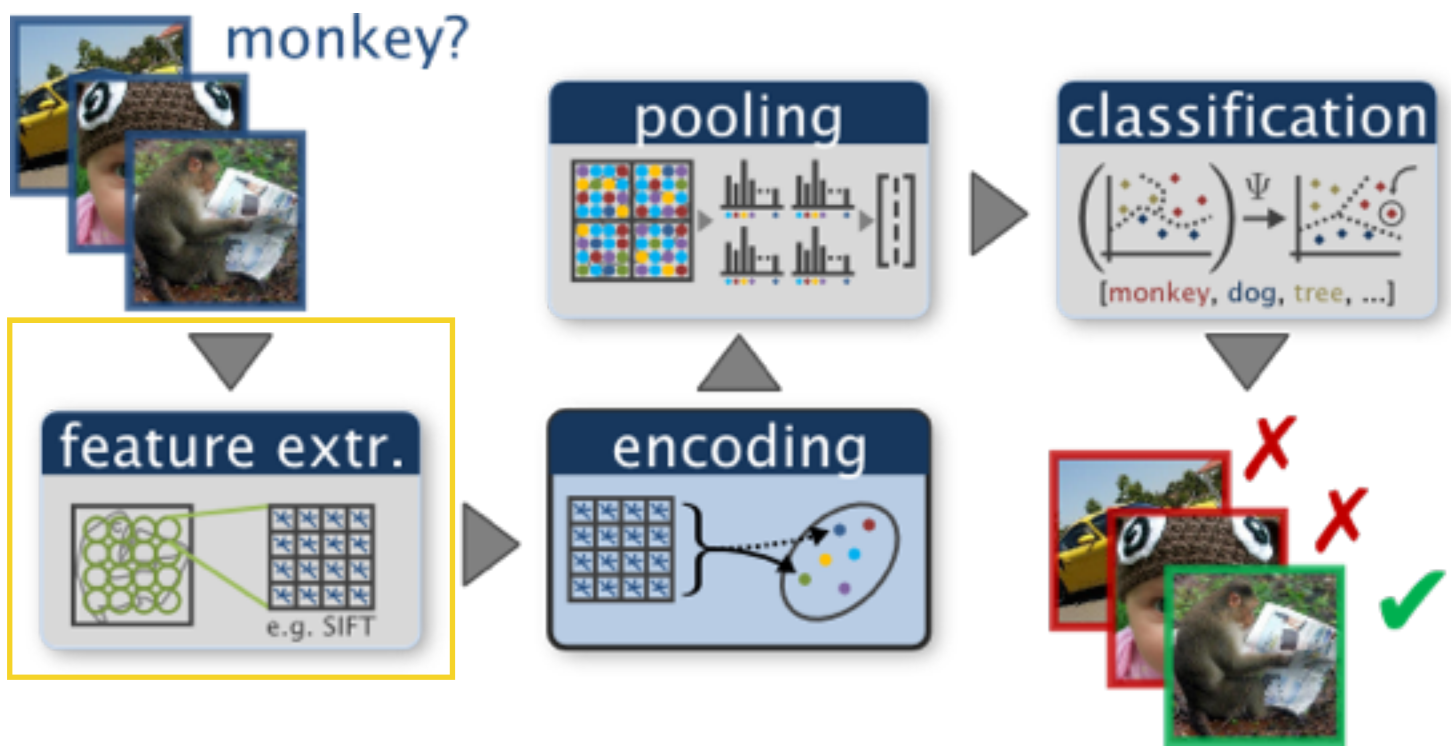
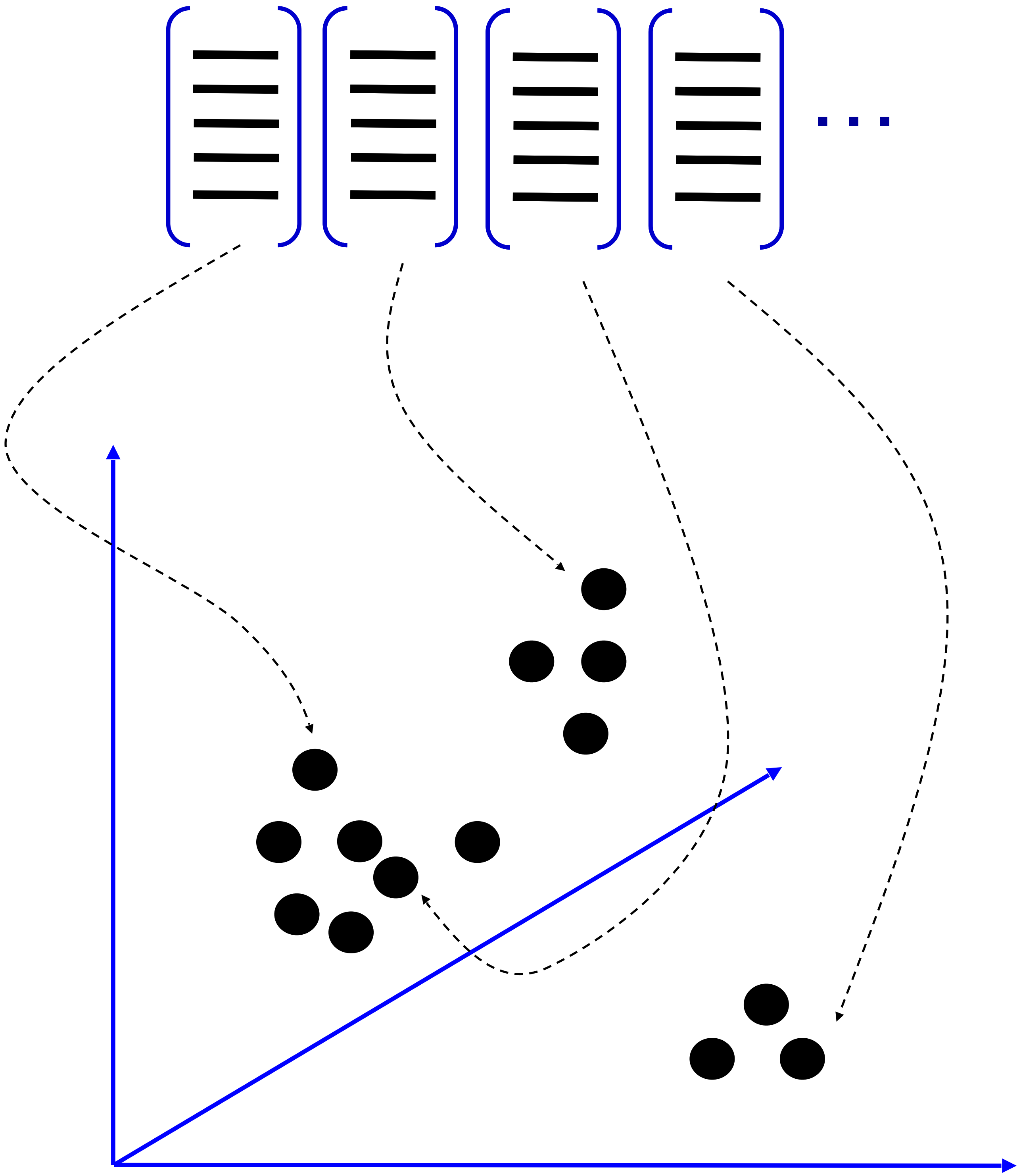


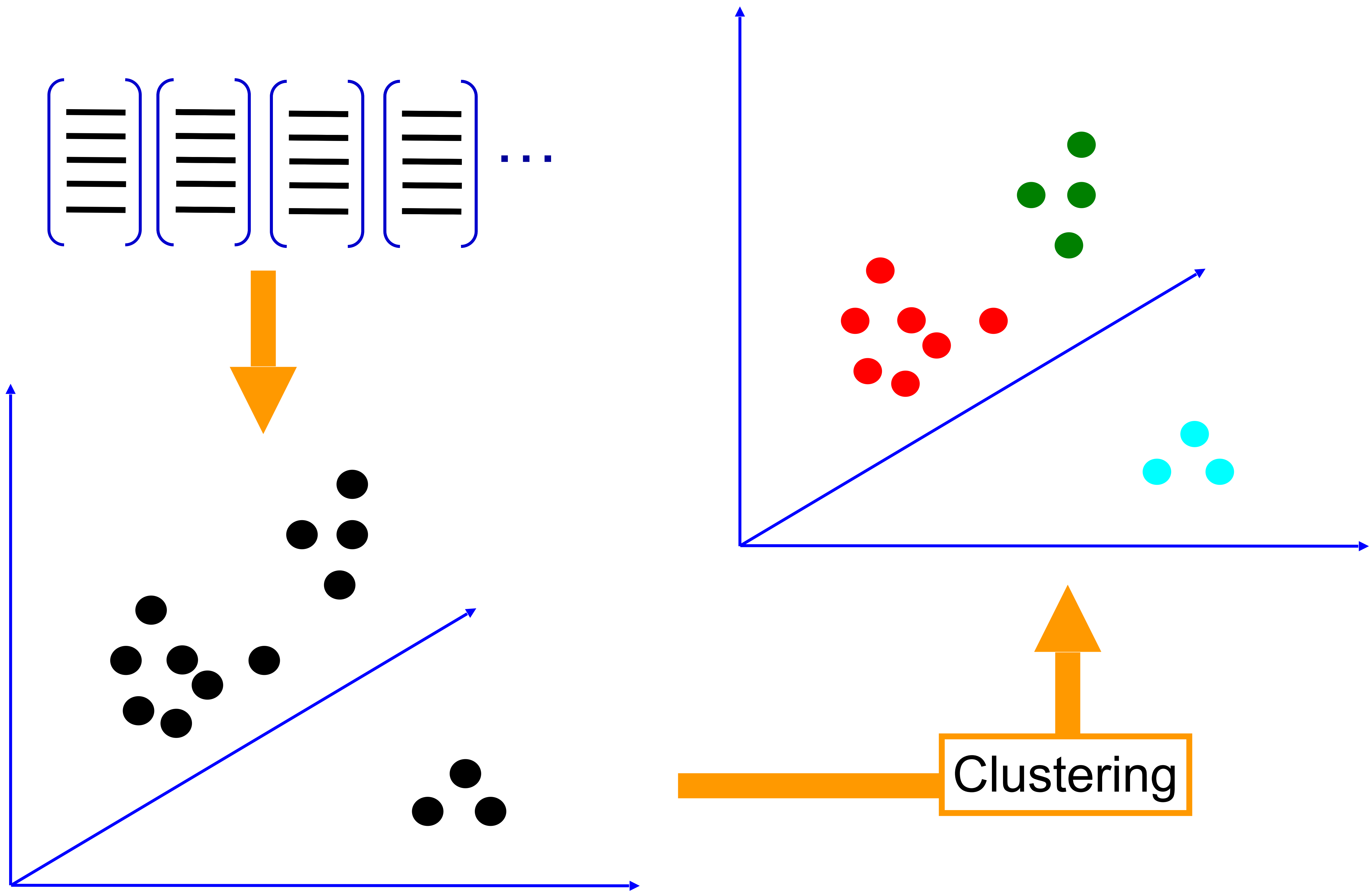
Figure from *Chatfield et al.,2011*

Learning a dictionary



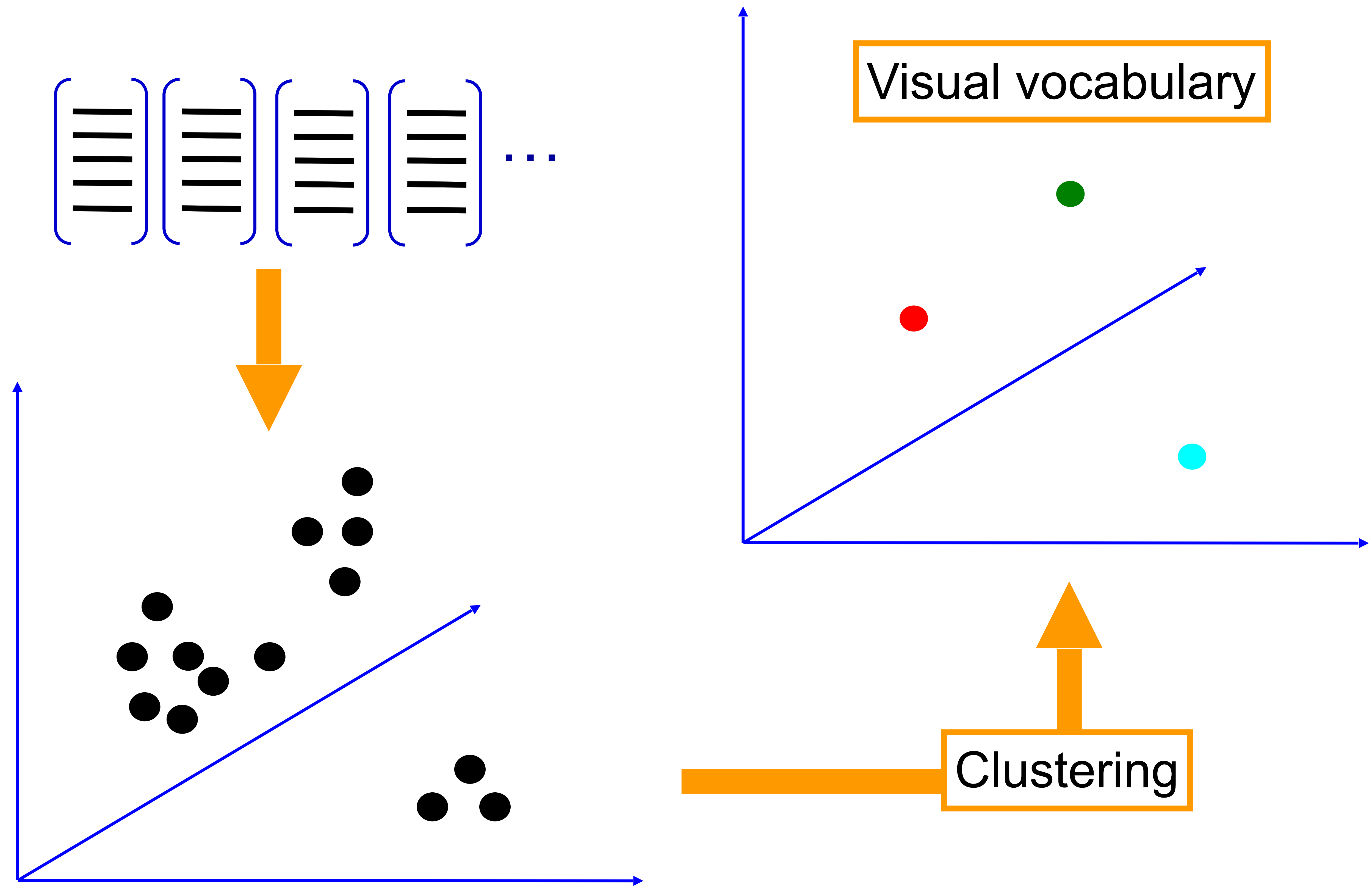
Slide credit: Josef Sivic

Learning a dictionary



Slide credit: Josef Sivic

Learning a dictionary



Slide credit: Josef Sivic

Review: K-means clustering

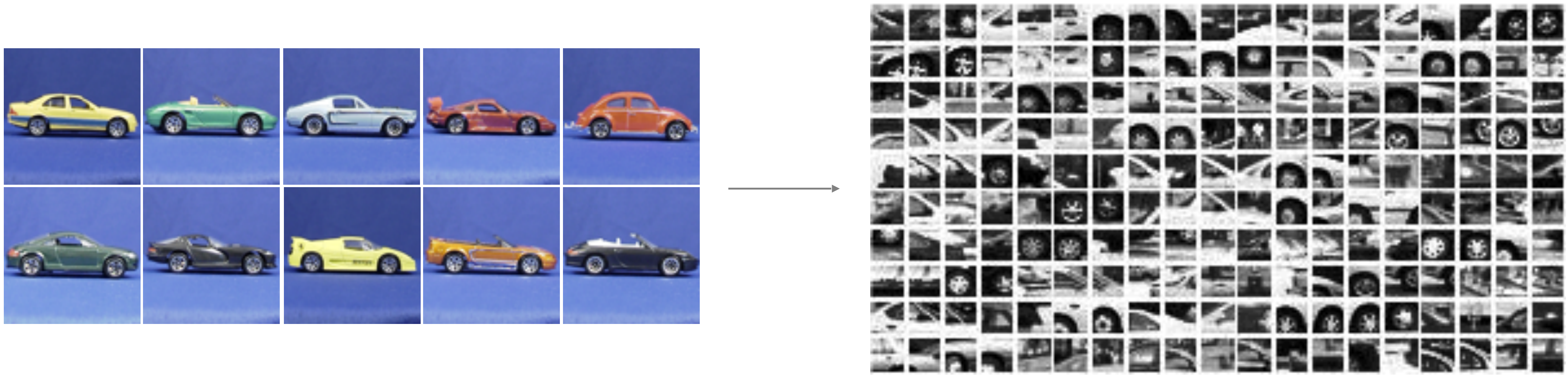
- Want to minimize sum of squared Euclidean distances between features $\mathbf{x}_i$ and their nearest cluster centers $\mathbf{m}_k$

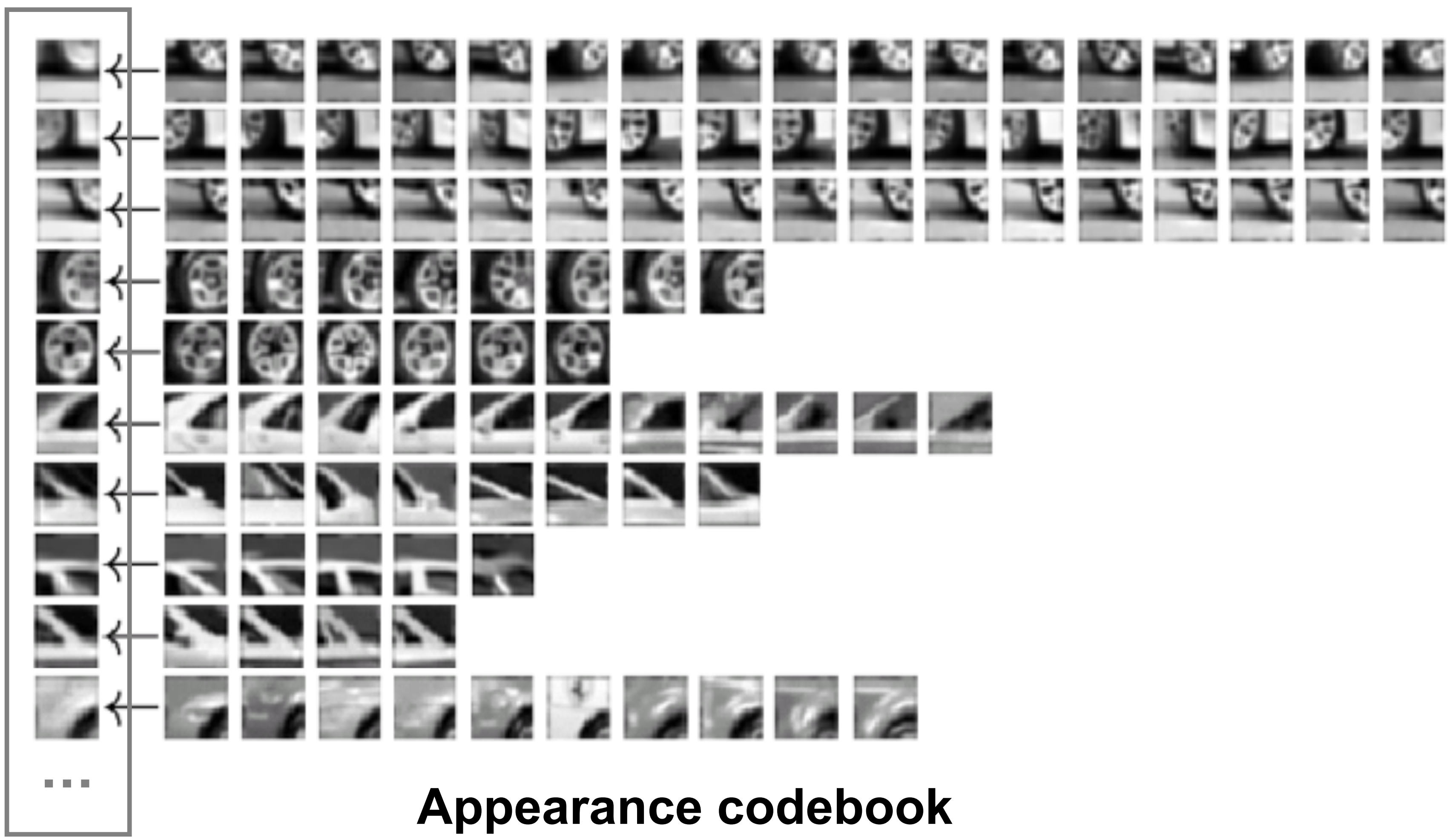
$$D(X,M) = \sum_{\text{cluster } k} \; \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (\mathbf{x}_i - \mathbf{m}_k)^2$$

Algorithm:

- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each feature to the nearest center
  - Recompute each cluster center as the mean of all features assigned to it

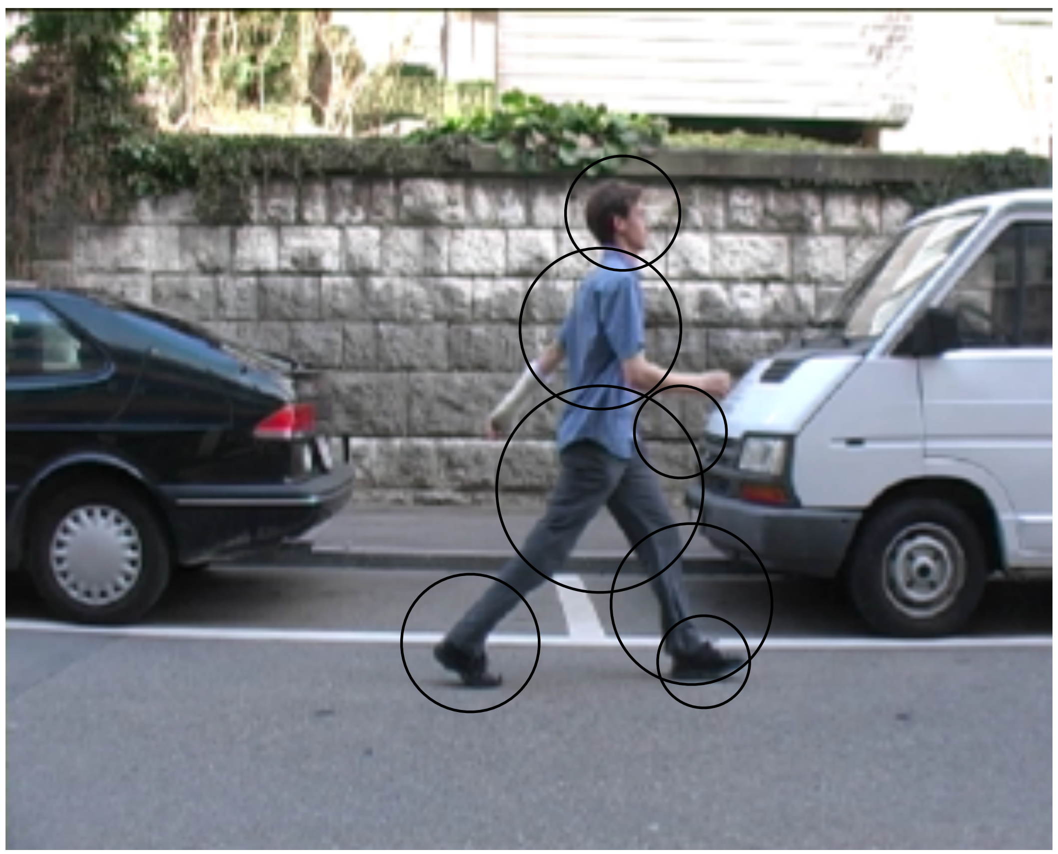
Example codebook

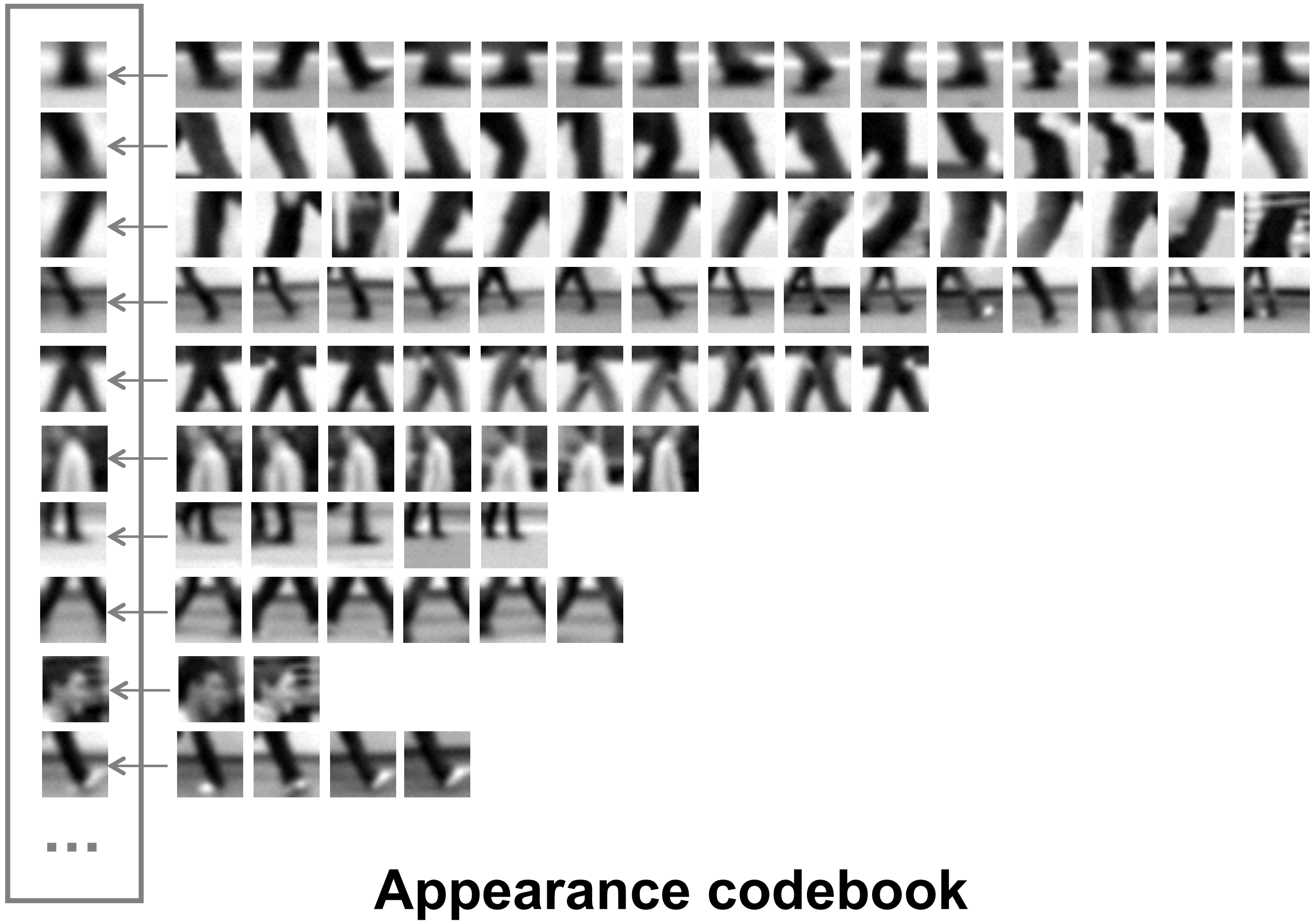




Appearance codebook

Another codebook





Appearance codebook

Lecture outline

- Origin and motivation of the "bag of words" model
- Algorithm pipeline
  - Extracting local features
  - Learning a dictionary — clustering using k-means
  - Encoding methods — hard vs. soft assignment
  - Spatial pooling — pyramid representations
  - Similarity functions and classifiers

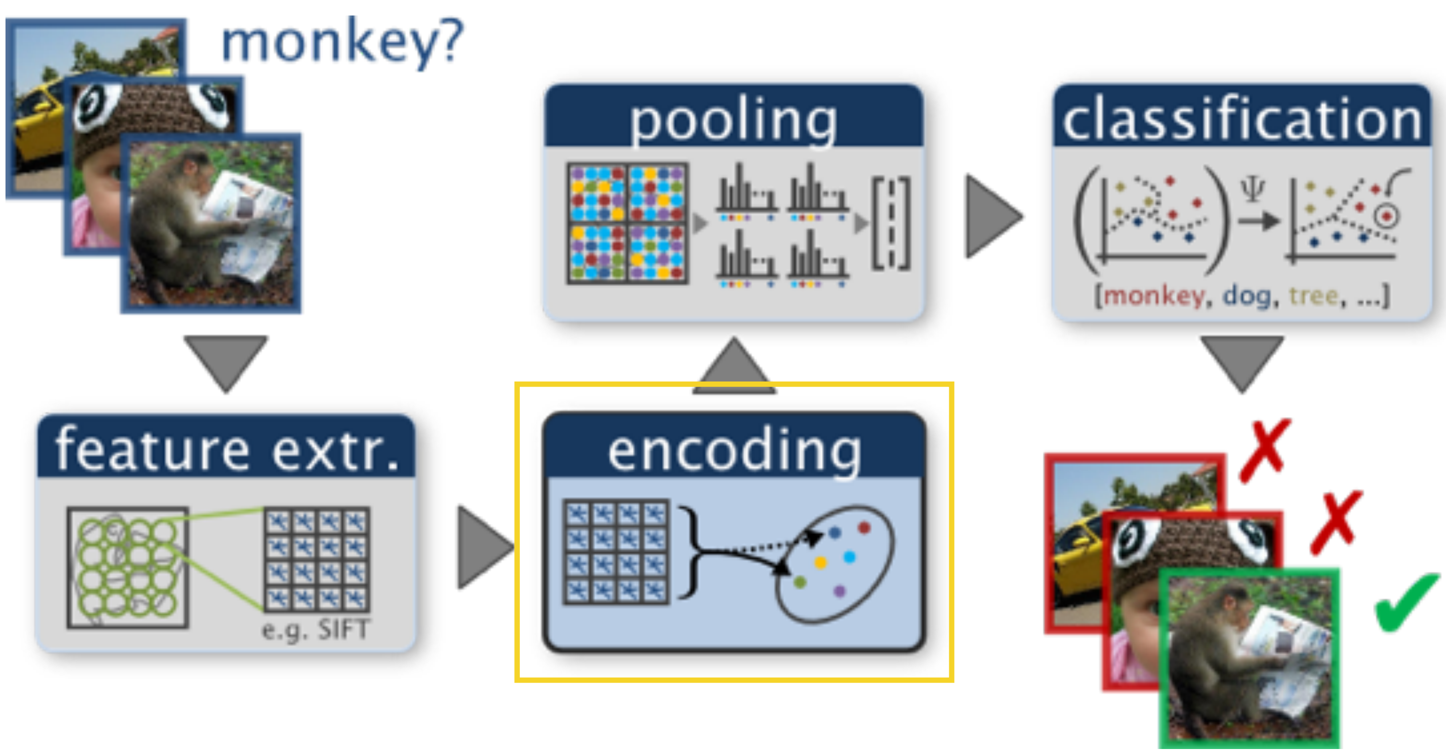
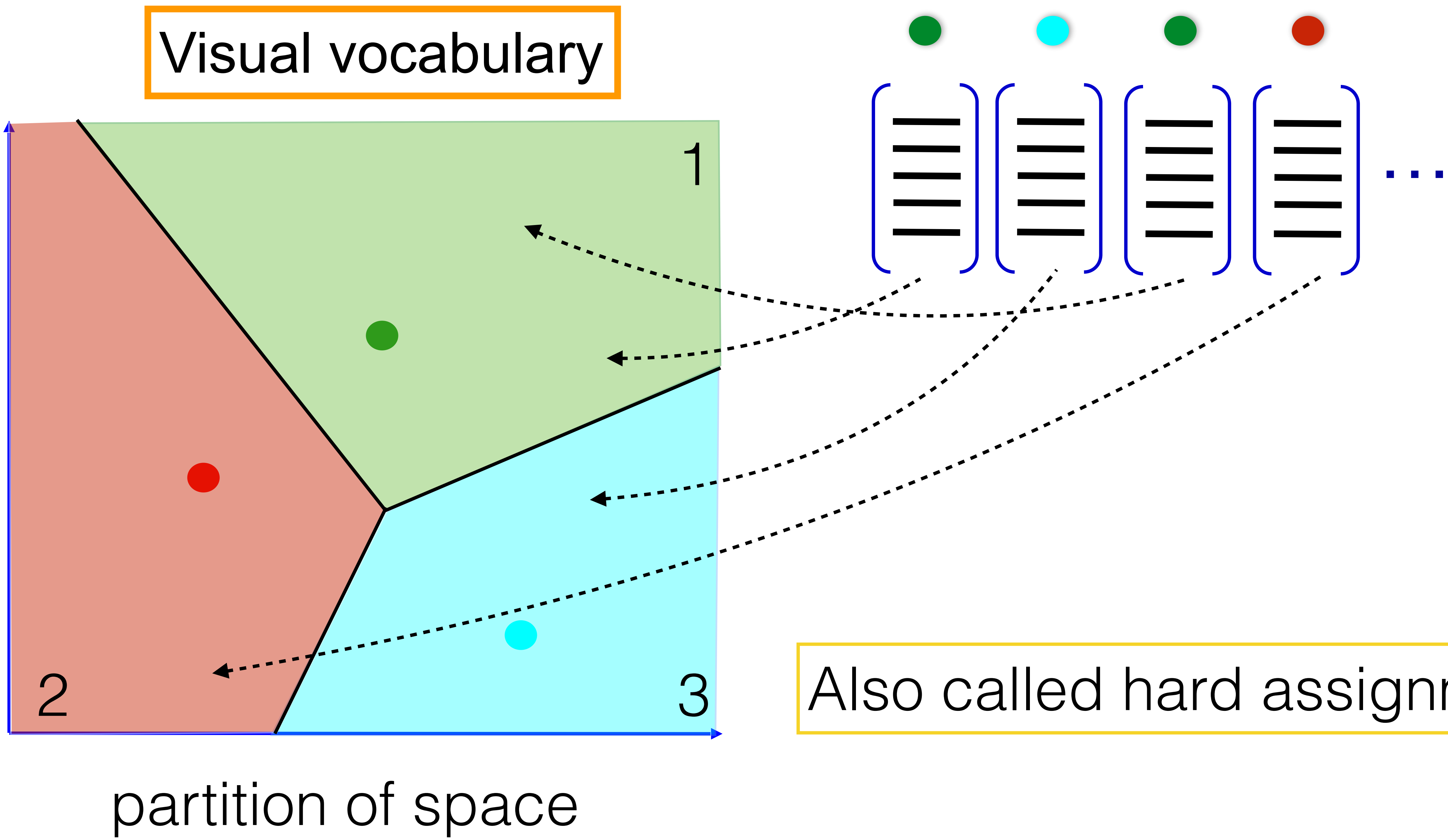


Figure from *Chatfield et al.,2011*

Encoding methods

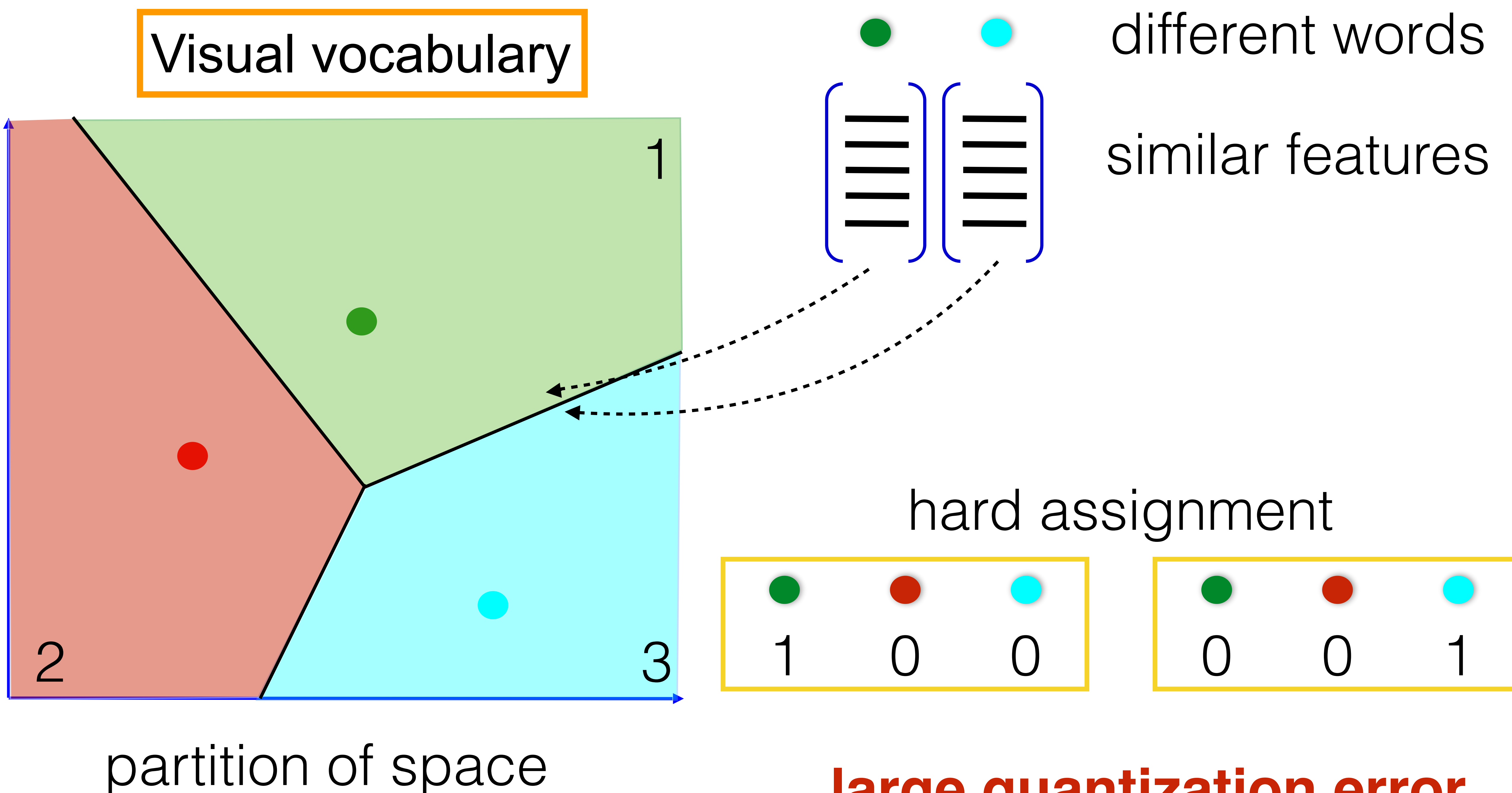
- Assigning words to features



Also called hard assignment

Encoding methods

- Assigning words to features

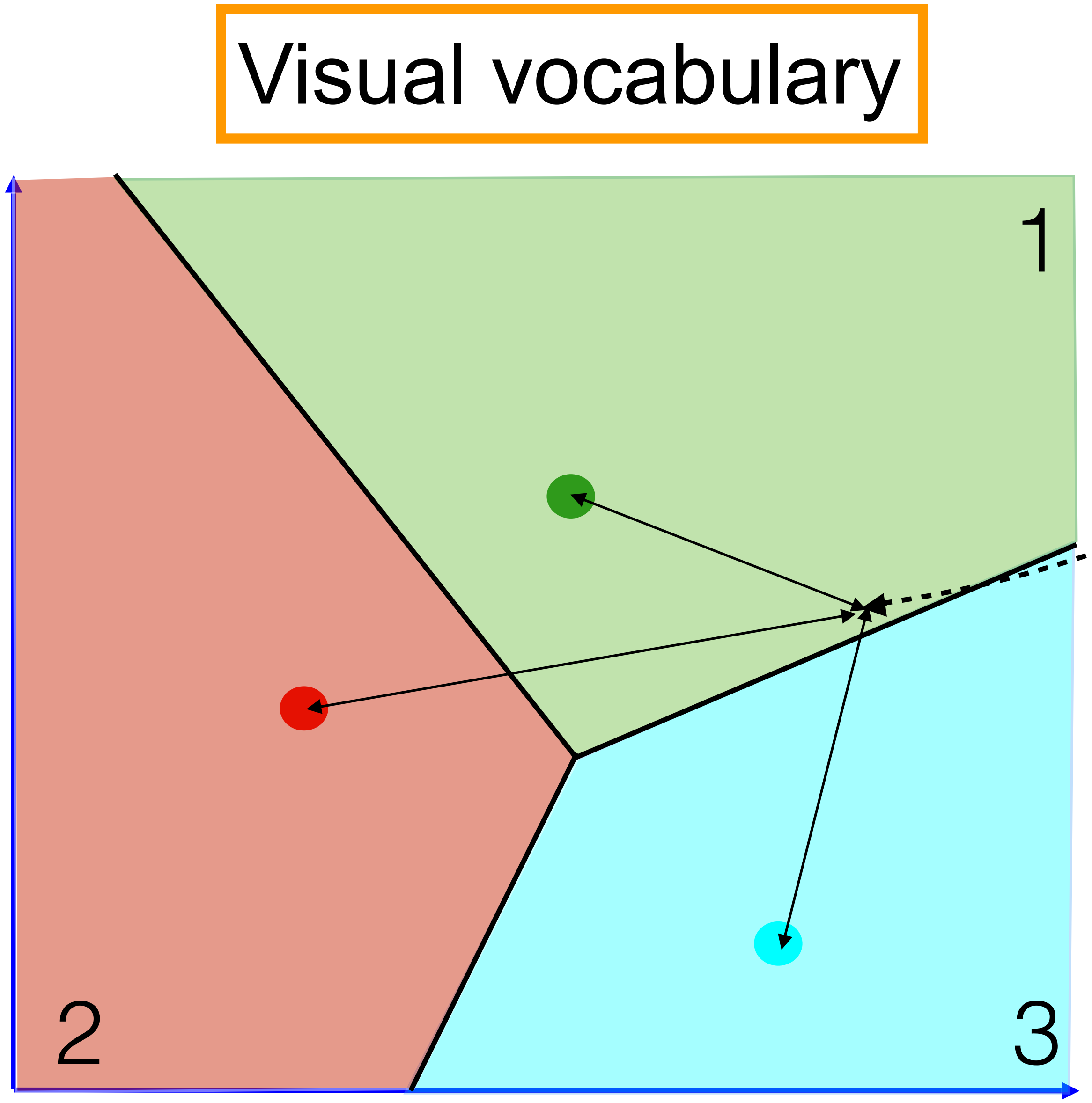


large quantization error

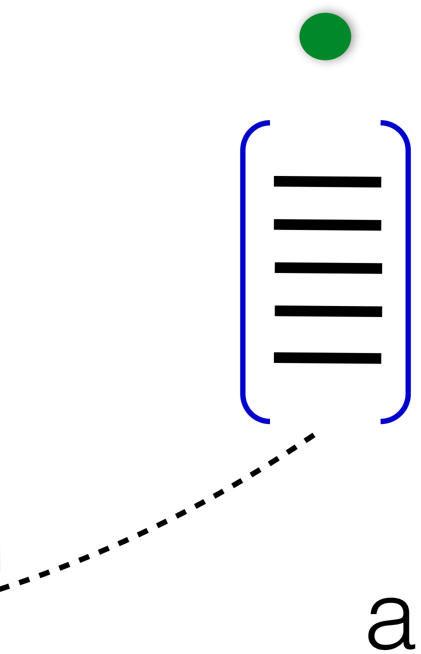
Encoding methods

- Assigning words to features

soft assignment



partition of space



$$\alpha_i \propto e^{-f(d(\mathbf{x},\mathbf{c_i}))}$$

assign high weights to centers that are close

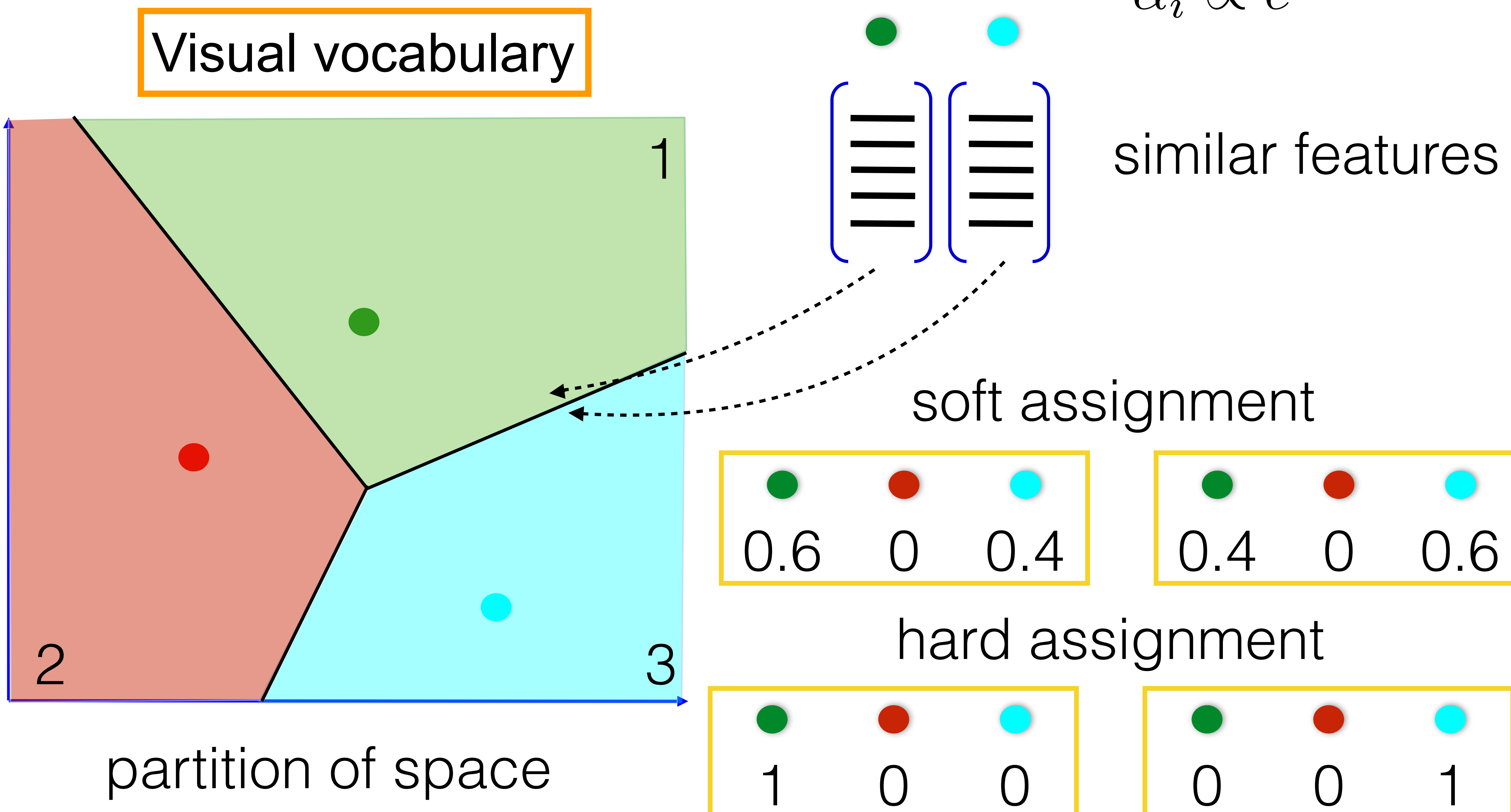
in practice non-zero to only k-nearest neighbors

Encoding methods

- Assigning words to features

soft assignment

$$\alpha_i \propto e^{-f(d(\mathbf{x},\mathbf{c_i}))}$$



Encoding considerations

- What should be the size of the dictionary?
  - Too small: don't capture the variability of the dataset
  - Too large: have too few points per cluster
  - The right size depends on the task and amount of data
    - e.g. instance retrieval (e.g. Nister) uses a vocabulary of 1 million, whereas recognition (e.g., texture) uses a vocabulary of about a hundred.
- Speed of embedding
  - Tree structured vocabulary (e.g. Nister)
  - Hashing, product quantization
- More accurate embeddings
  - Generalizations of soft embedding: LLC coding, sparse coding
  - Higher order statistics: Fisher vectors, VLAD, etc.

Lecture outline

- Origin and motivation of the “bag of words” model
- Algorithm pipeline
  - Extracting local features
  - Learning a dictionary — clustering using k-means
  - Encoding methods — hard vs. soft assignment
  - Spatial pooling — pyramid representations
  - Similarity functions and classifiers

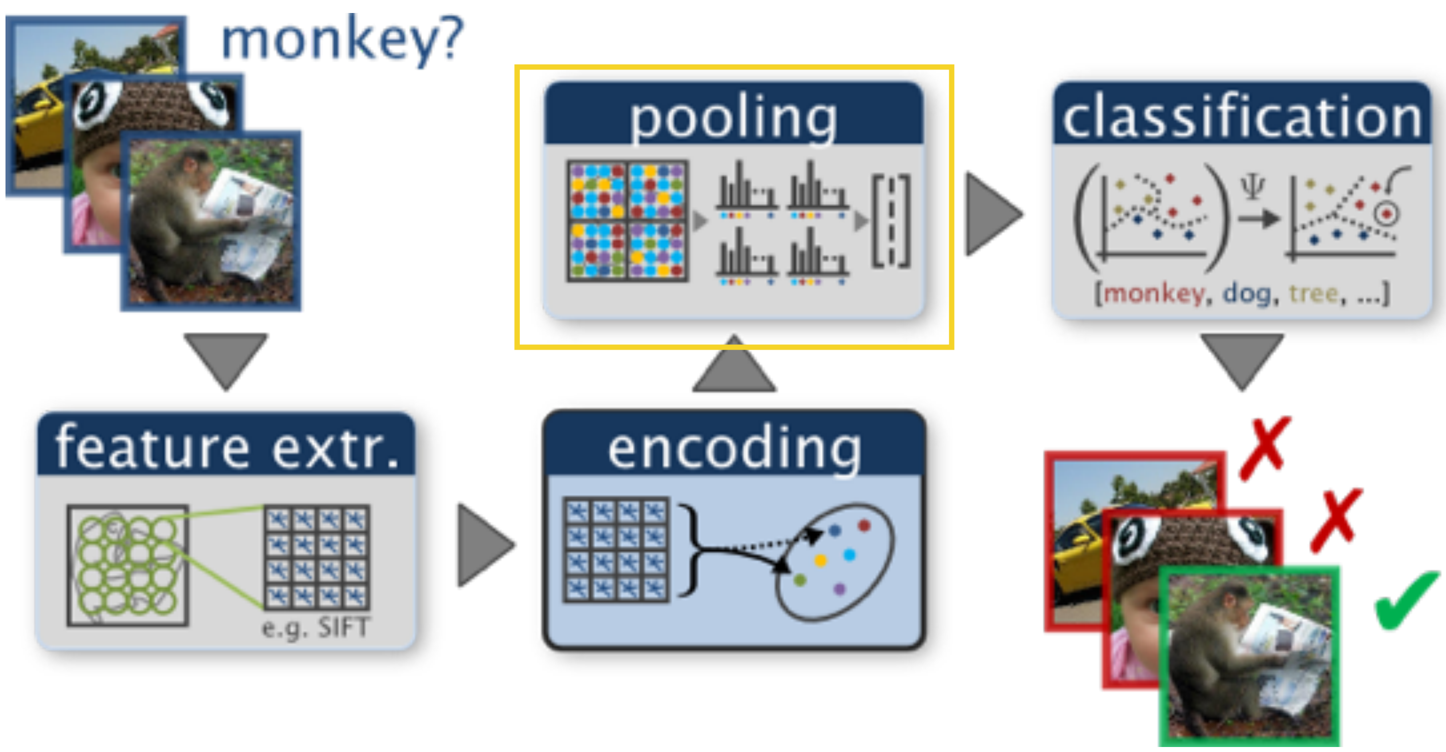
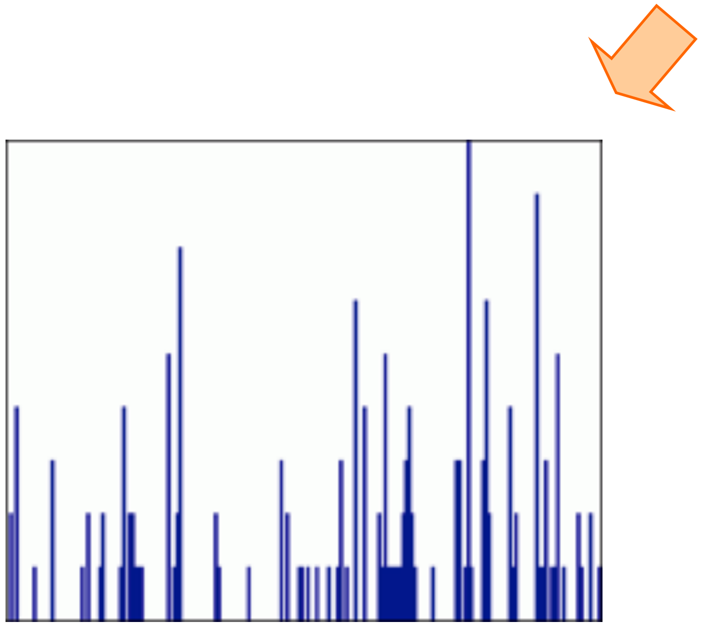


Figure from *Chatfield et al.,2011*

Spatial pyramids

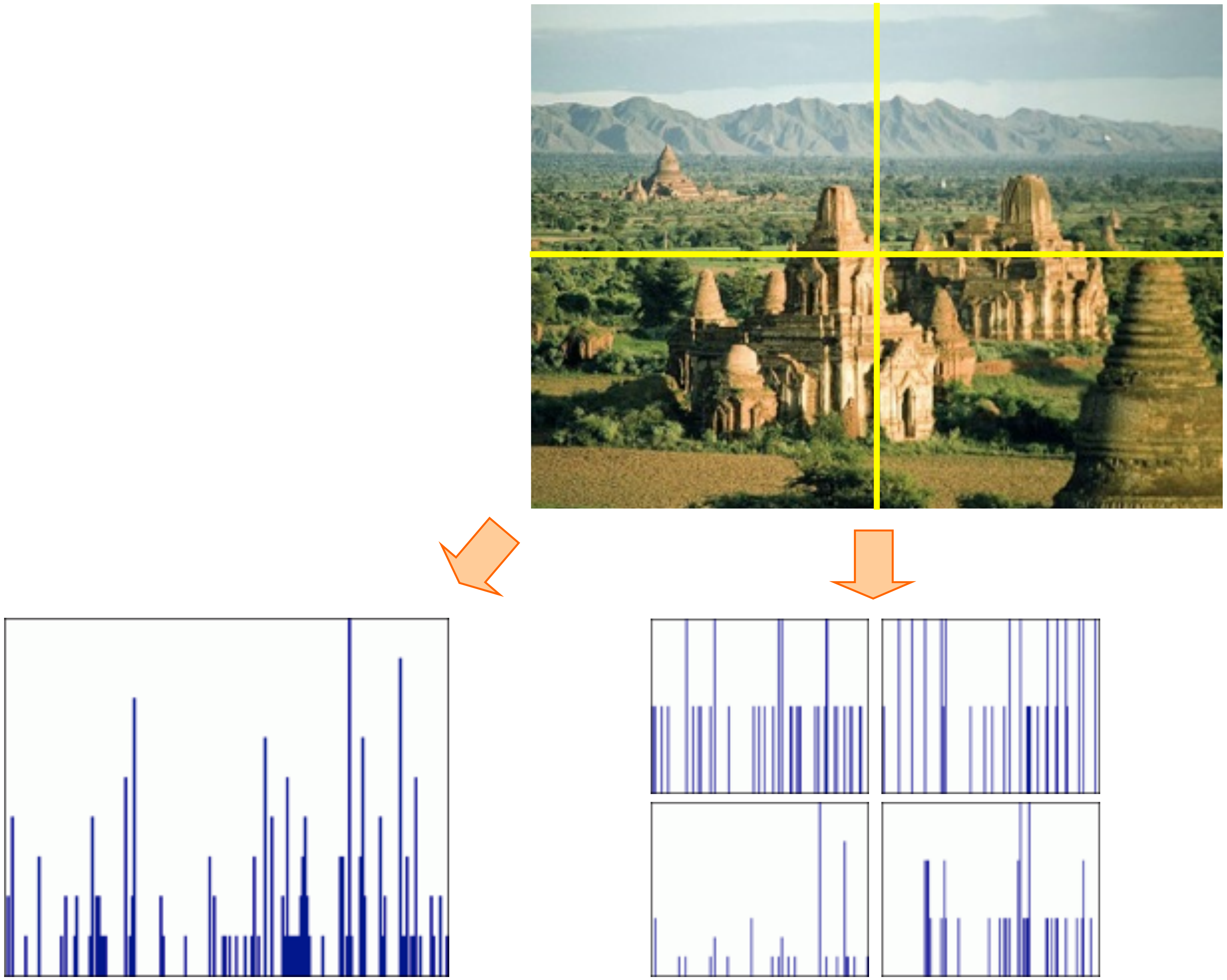
pooling: sum embeddings of local features within a region



Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramids

pooling: sum embeddings of local features within a region

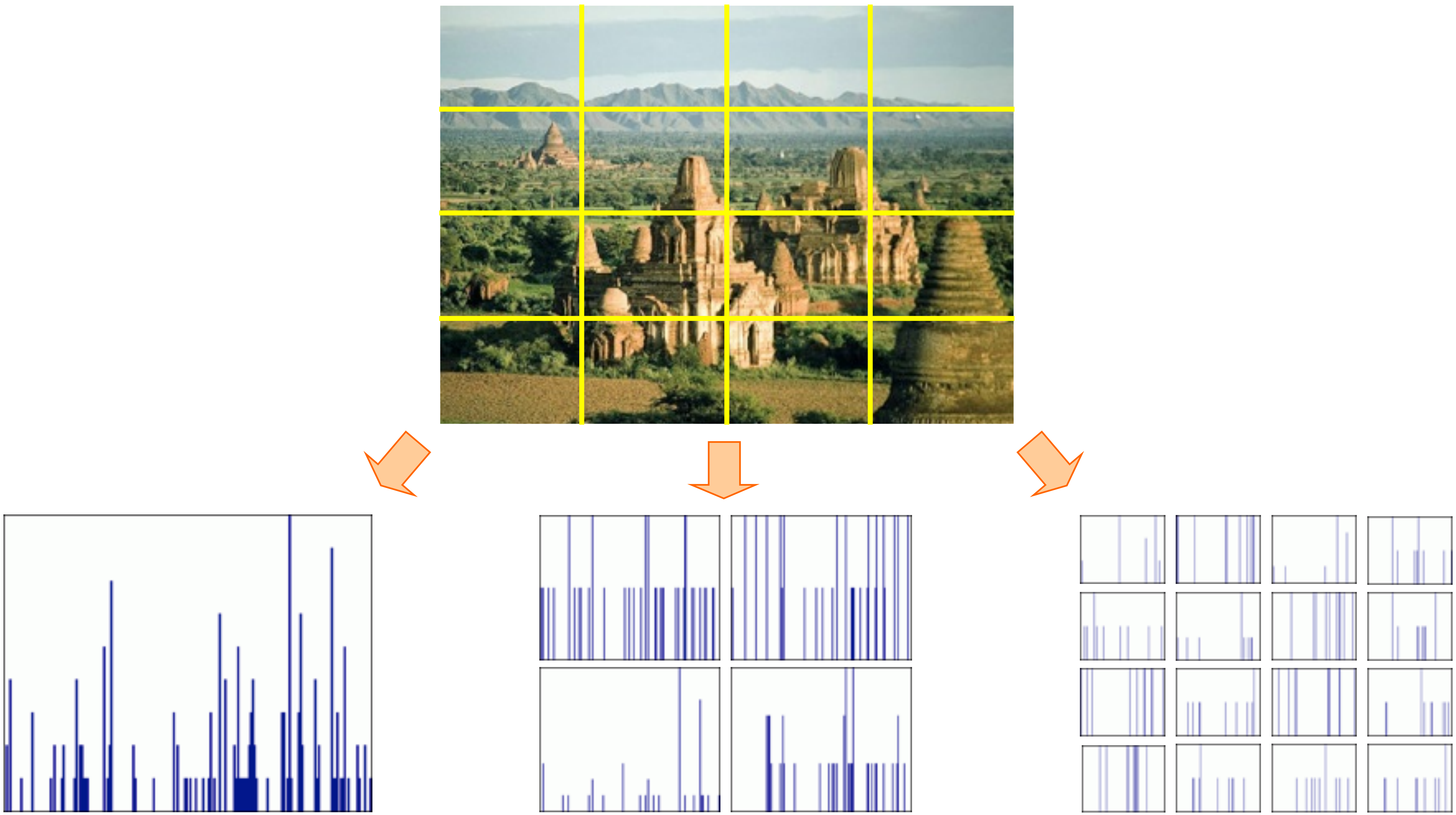


Same motivation as **SIFT** — keep coarse layout information

Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramids

pooling: sum embeddings of local features within a region



Same motivation as **SIFT** — keep coarse layout information

Lazebnik, Schmid & Ponce (CVPR 2006)

Lecture outline

- Origin and motivation of the “bag of words” model
- Algorithm pipeline
  - Extracting local features
  - Learning a dictionary — clustering using k-means
  - Encoding methods — hard vs. soft assignment
  - Spatial pooling — pyramid representations
  - Similarity functions and classifiers

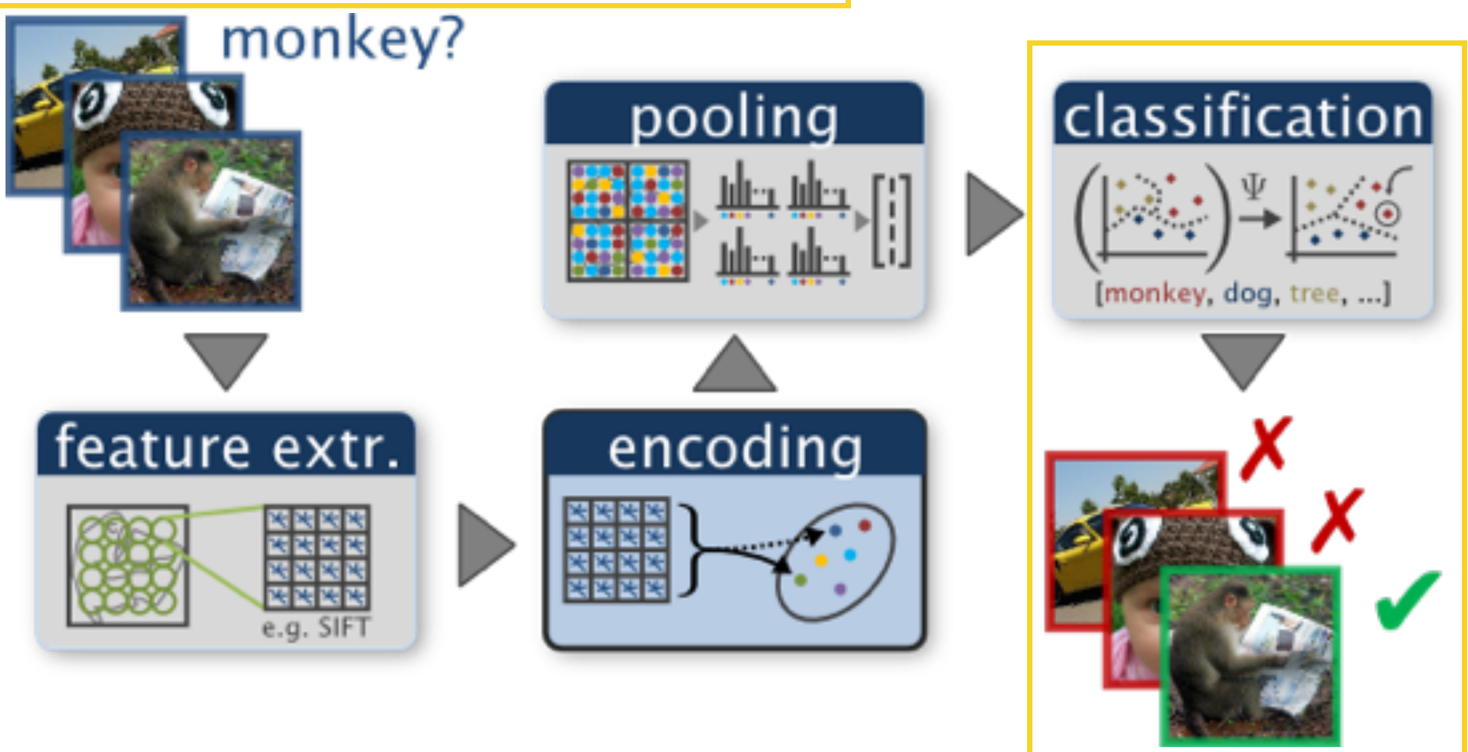


Figure from *Chatfield et al.,2011*

Bags of features representation

I

$\mathbf{h} = \Phi(I)$

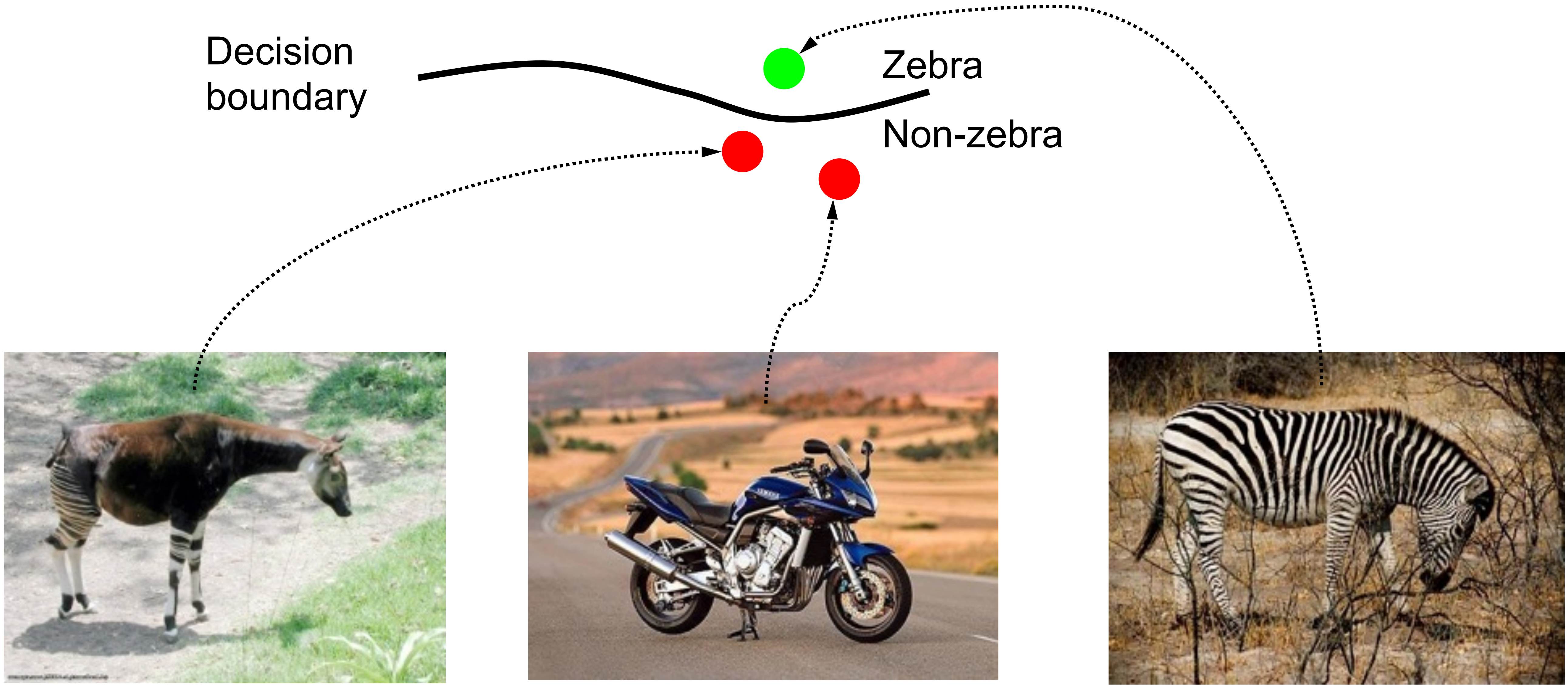
image similarity = feature similarity

Comparing features

- Euclidean distance: $$D(\mathbf{h}_1,\mathbf{h}_2) = \sqrt{\sum_{i=1}^{N} (\mathbf{h}_1(i) - \mathbf{h}_2(i))^2}$$
- L1 distance: $$D(\mathbf{h}_1,\mathbf{h}_2) = \sum_{i=1}^{N} | \mathbf{h}_1(i) - \mathbf{h}_2(i) |$$
- χ^2 distance: $$D(\mathbf{h}_1,\mathbf{h}_2) = \sum_{i=1}^{N} \frac{\left(\mathbf{h}_1(i) - \mathbf{h}_2(i)\right)^2}{\mathbf{h}_1(i) + \mathbf{h}_2(i)}$$
- Histogram intersection (similarity): $$I(\mathbf{h}_1,\mathbf{h}_2) = \sum_{i=1}^{N} \min(\mathbf{h}_1(i), \mathbf{h}_2(i))$$
- Hellinger kernel (similarity): $$K(\mathbf{h}_1,\mathbf{h}_2) = \sum_{i=1}^{N} \sqrt{\mathbf{h}_1(i)\,\mathbf{h}_2(i)}$$

Classifiers

- Given a feature representation for images, how do we learn a model for distinguishing features from different classes?

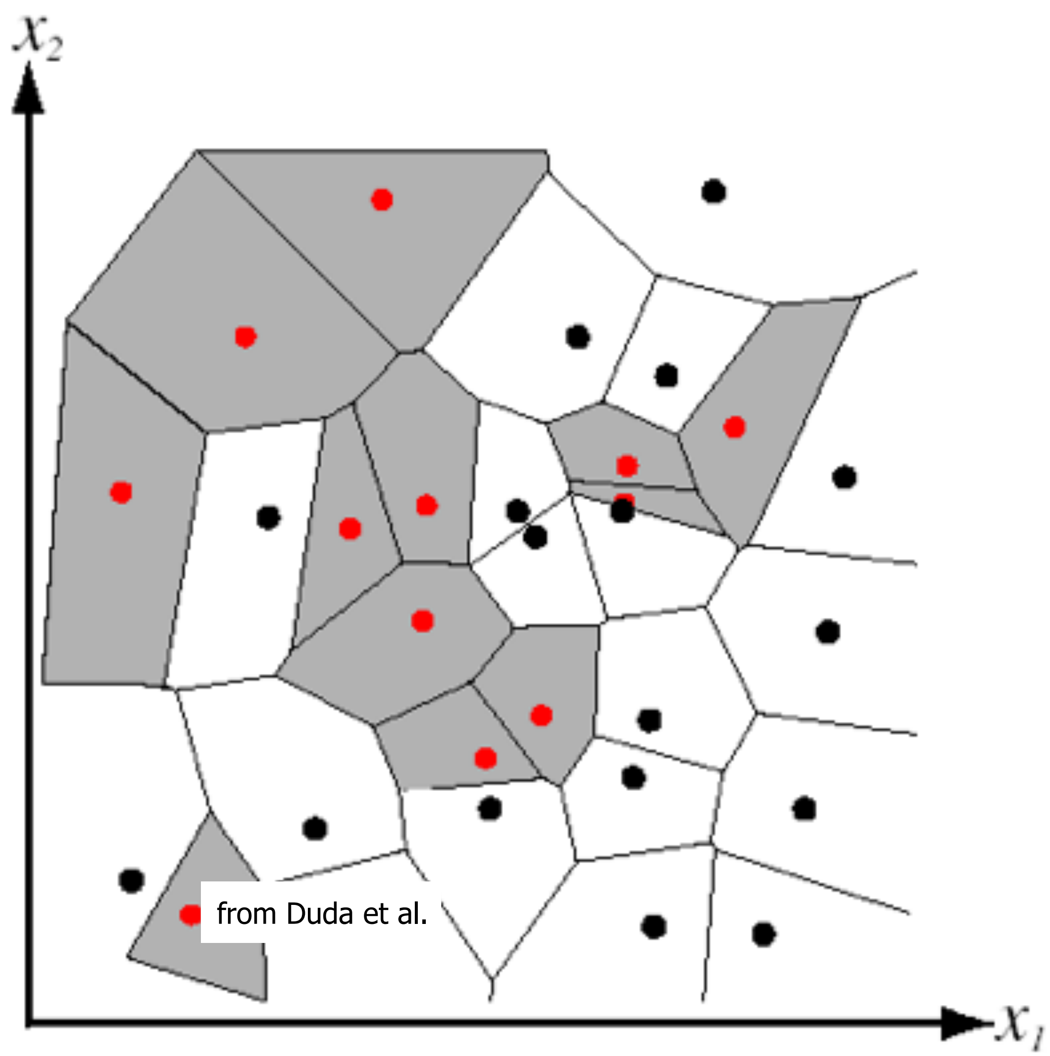


Classifiers

- Given a feature representation for images, how do we learn a model for distinguishing features from different classes?
- Examples of commonly used classifiers
  - Nearest neighbor classifiers
  - Linear classifiers: support vector machines

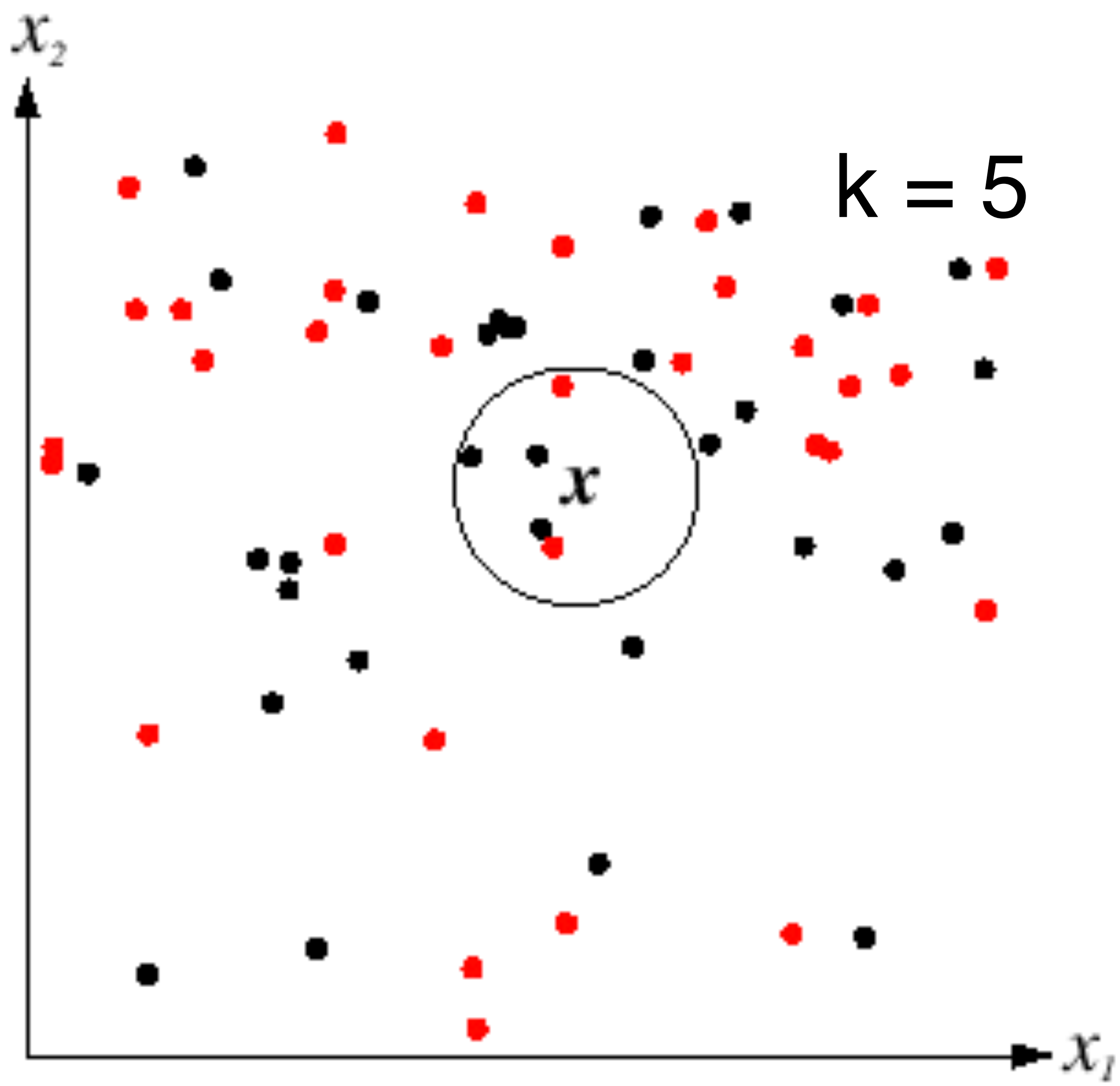
Nearest neighbor classifier

- Assign label of nearest training data point to each test data point



k-Nearest neighbor classifier

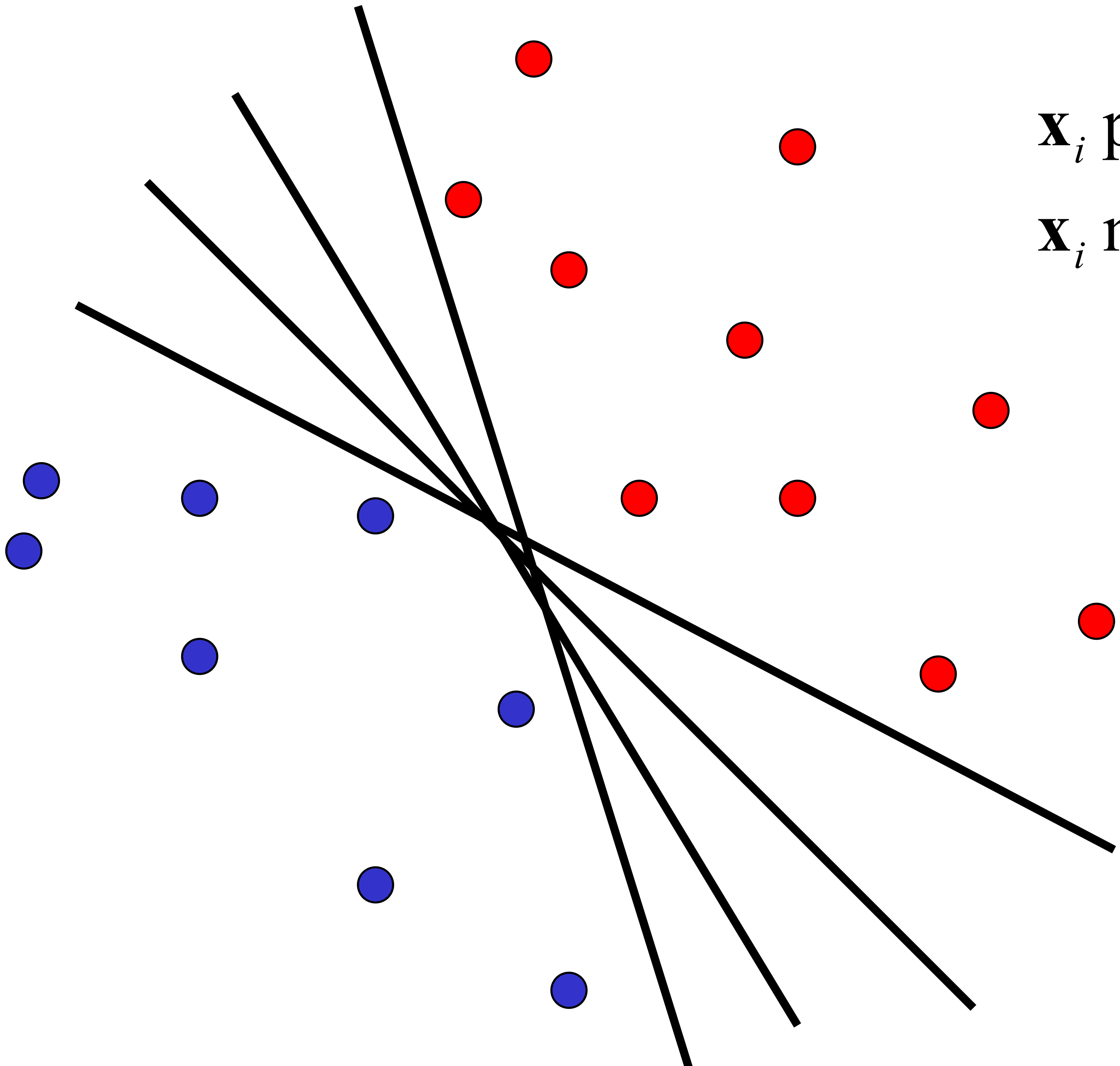
- For a new point, find the k closest points from training data
- Labels of the k points “vote” to classify



Linear classifiers

Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

Which hyperplane is best?

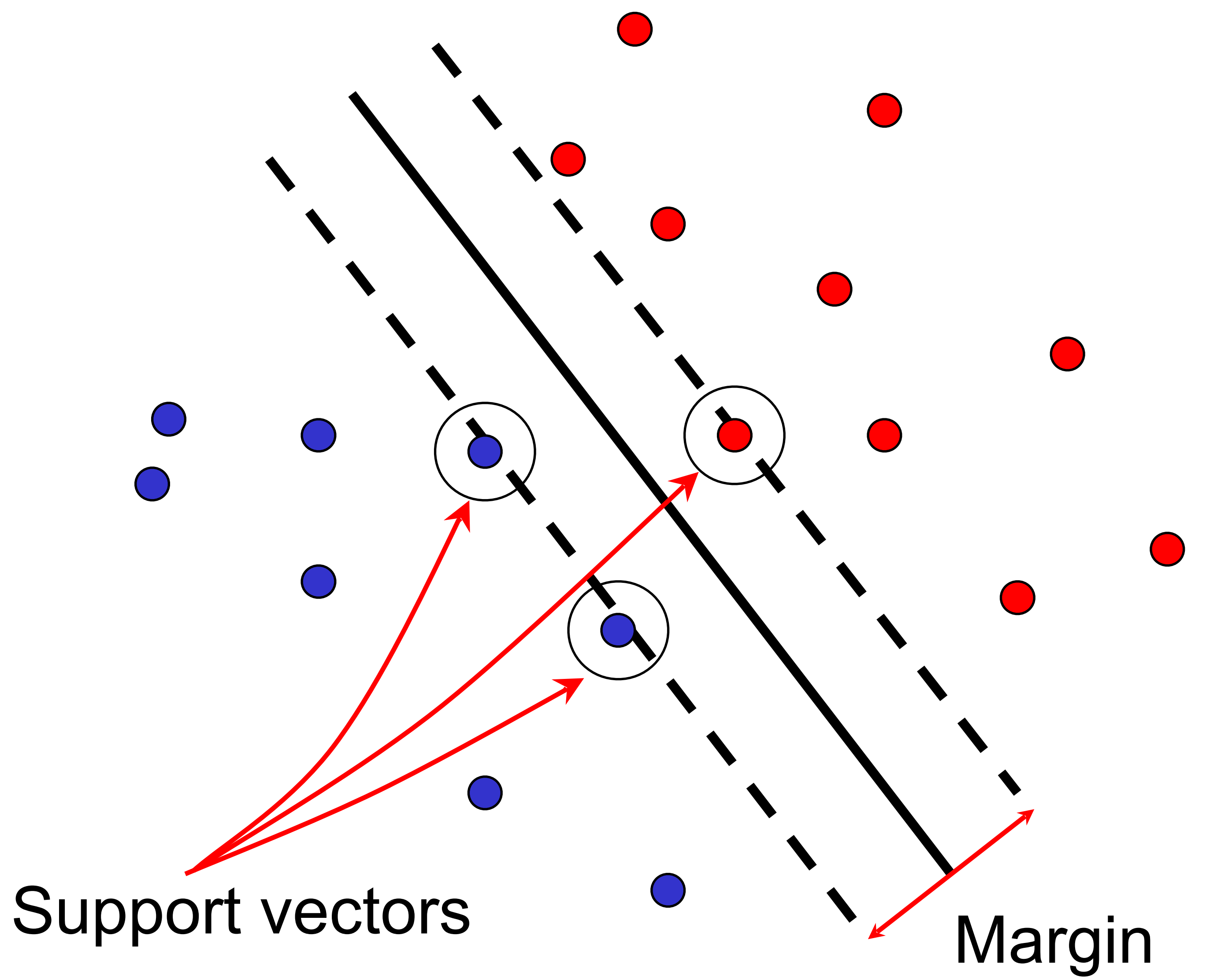
Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples

C. Burges, **A Tutorial on Support Vector Machines for Pattern Recognition**, Data Mining and Knowledge Discovery, 1998

Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

C. Burges, **A Tutorial on Support Vector Machines for Pattern Recognition**, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

1. Maximize margin $2 / \|\mathbf{w}\|$
2. Correctly classify all training data:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

Quadratic optimization problem:

$$\boxed{\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1}$$

C. Burges, **A Tutorial on Support Vector Machines for Pattern Recognition**, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

- Solution: $$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

Learned weight
(nonzero only for support vectors)

C. Burges, **A Tutorial on Support Vector Machines for Pattern Recognition**, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

- Solution: $$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

  $\mathbf{w} \cdot \mathbf{x}_i + b = y_i$, for any support vector

- Classification function (decision boundary):
$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x}_i$

- Solving the optimization problem also involves computing the inner products $\mathbf{x}_i \cdot \mathbf{x}_j$ between all pairs of training points

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

What if the data is not linearly separable?

- Separable: $$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$$
- Non-separable: $$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$ $$\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

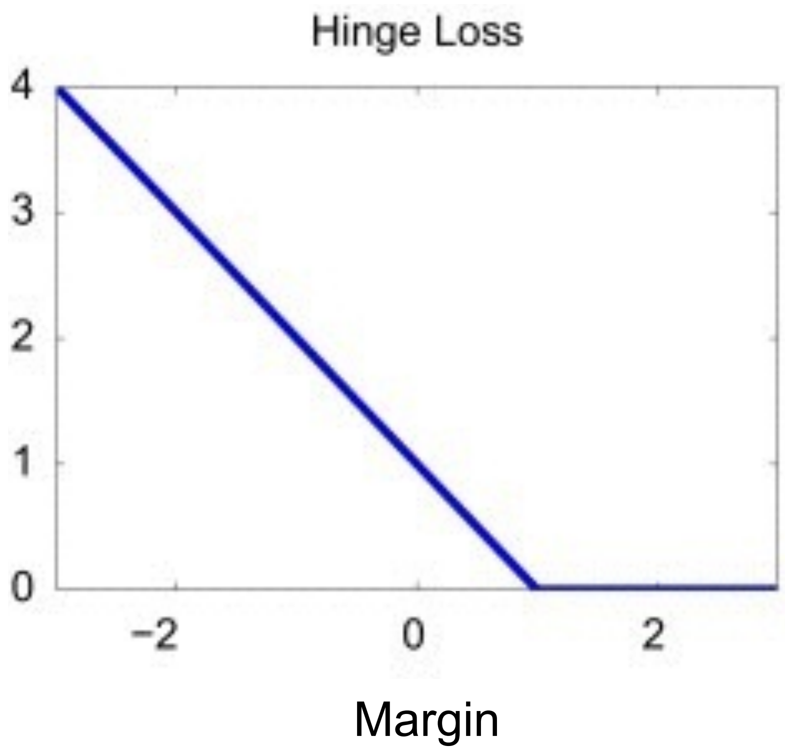
- *C*: tradeoff constant, ξ_i : *slack variable* (positive)
- Whenever margin is ≥ 1, $\xi_i = 0$ $\xi_i = 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)$
- Whenever margin is < 1,

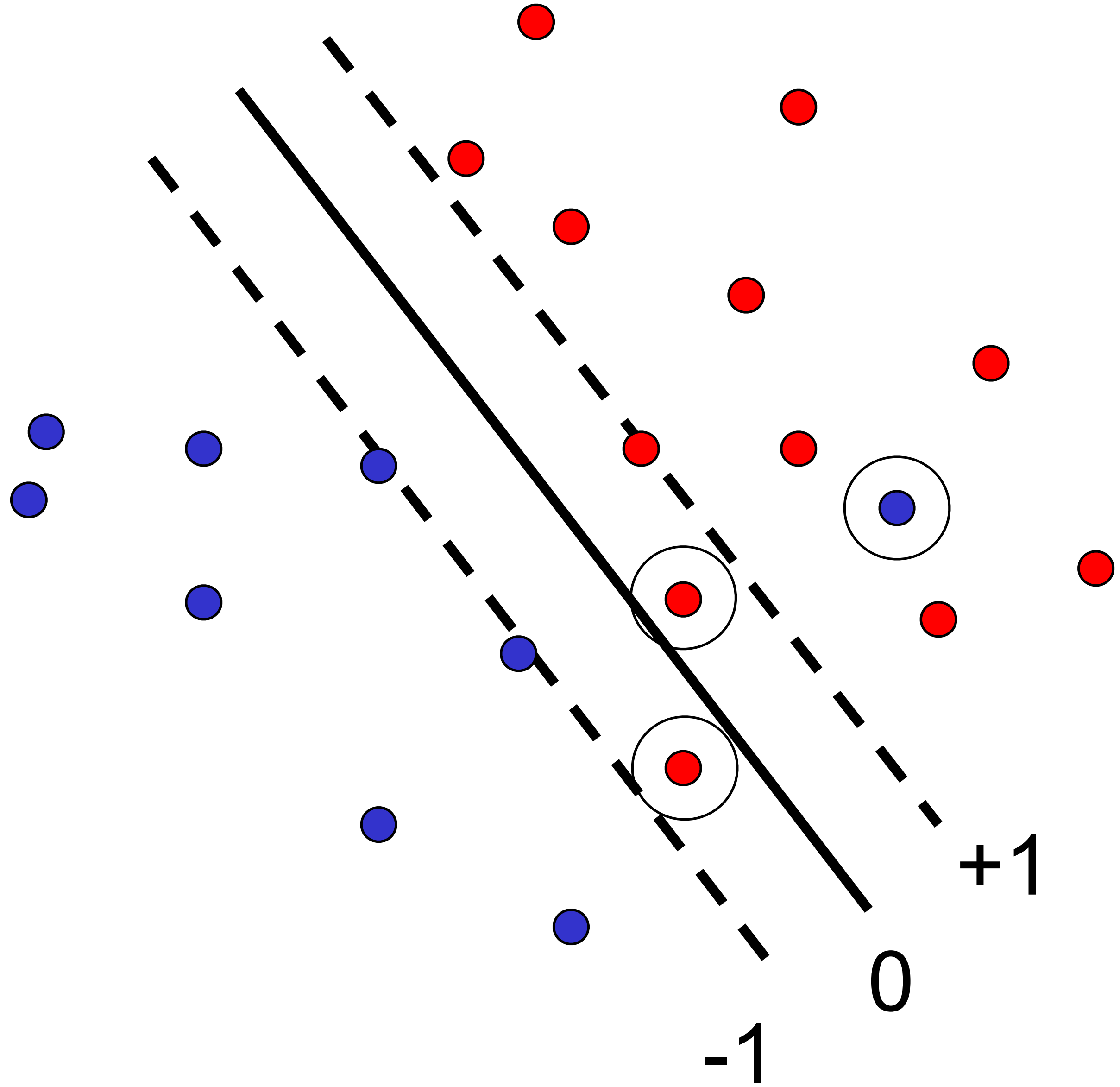
What if the data is not linearly separable?

$$\min_{\mathbf{w},b} \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\boxed{\text{Maximize margin}}} + C \underbrace{\sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)\right)}_{\boxed{\text{Minimize classification mistakes}}}$$

What if the data is not linearly separable?

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\max\big(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)\big)$$

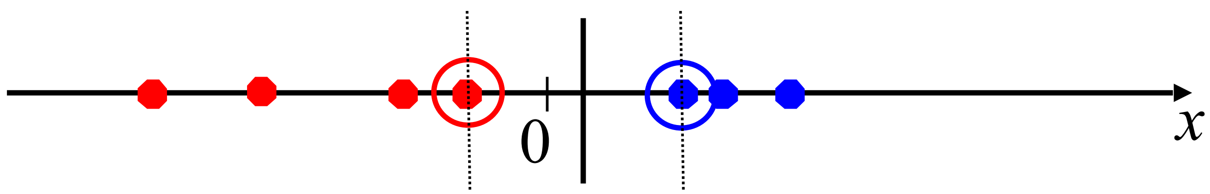




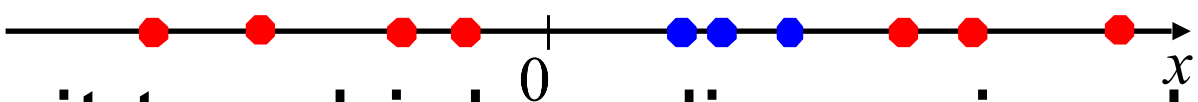
Demo: http://cs.stanford.edu/people/karpathy/svmjs/demo

Nonlinear SVMs

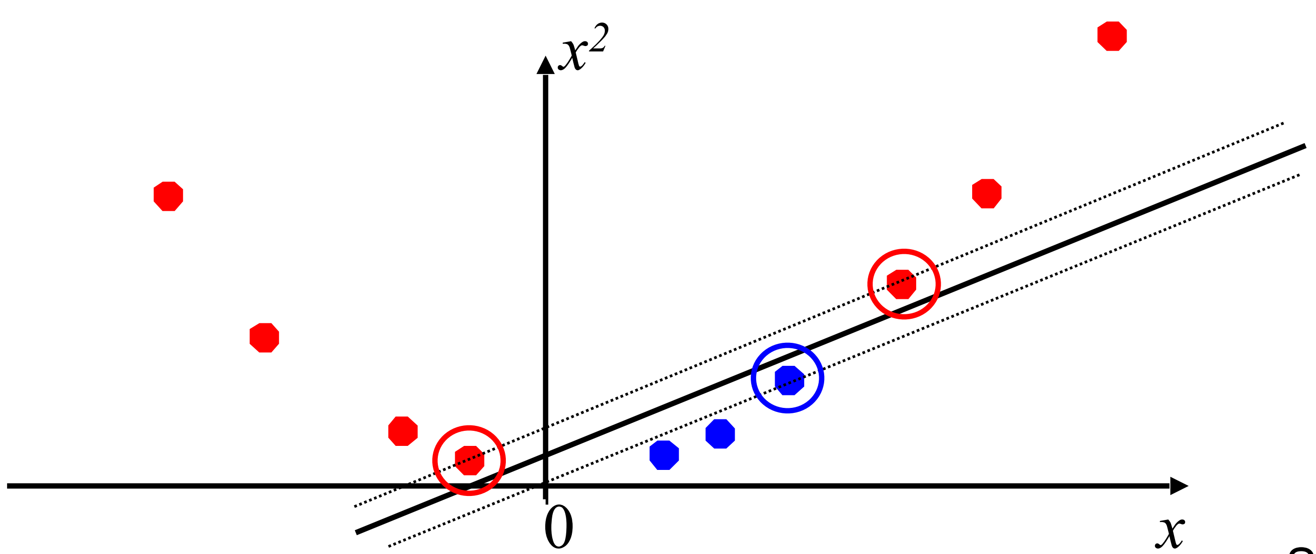
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



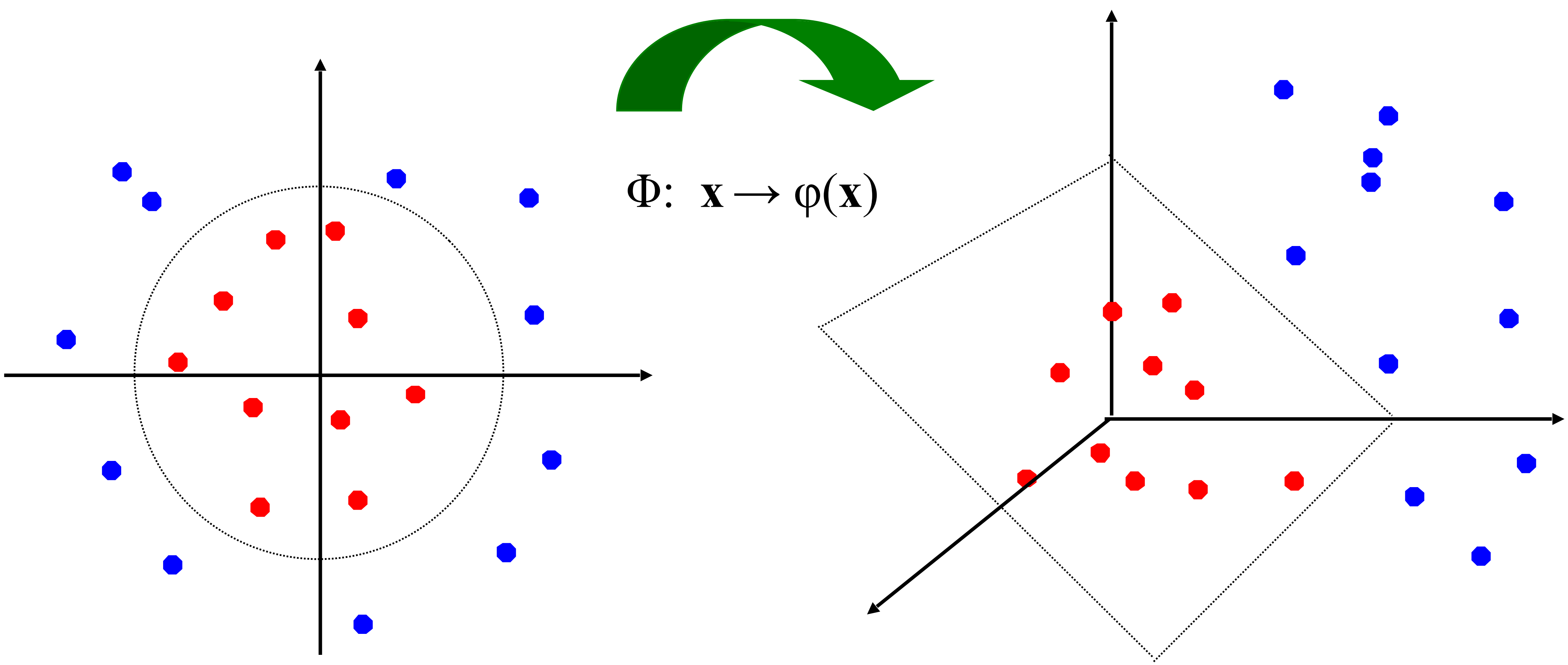
- We can map it to a higher-dimensional space:



Slide credit: Andrew Moore

Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

Nonlinear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\boldsymbol{\varphi}(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x},\mathbf{y}) = \boldsymbol{\varphi}(\mathbf{x}) \cdot \boldsymbol{\varphi}(\mathbf{y})$$

(the kernel function must satisfy *Mercer's condition*)

- This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

C. Burges, [A Tutorial on Support Vector Machines for Pattern Recognition](), Data Mining and Knowledge Discovery, 1998

Non-linear kernels for histograms

- Histogram intersection kernel:

$$I(\mathbf{h}_1, \mathbf{h}_2) = \sum_{i=1}^{N} \min(\mathbf{h}_1(i), \mathbf{h}_2(i))$$

- Hellinger kernel: $$K(\mathbf{h}_1, \mathbf{h}_2) = \sum_{i=1}^{N} \sqrt{\mathbf{h}_1(i)\,\mathbf{h}_2(i)}$$

- Generalized Gaussian kernel:

$$K(\mathbf{h}_1, \mathbf{h}_2) = \exp\left(-\frac{1}{A} D(\mathbf{h}_1, \mathbf{h}_2)^2\right)$$

  - *D* can be L1, Euclidean, χ^2 distance, etc.

J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, **Local Features and Kernels for Classifcation of Texture and Object Categories: A Comprehensive Study**, IJCV 2007

Summary: SVMs for image classification

1. Pick an image representation (in our case, bag of features)
2. Pick a kernel function for that representation
3. Feed the kernel and features into your favorite SVM solver to obtain support vectors and weights
4. At test time: compute kernel values for your test example and each support vector, and combine them with the learned weights to get the value of the decision function

$$\sum_i \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

Lots of software available! LIBSVM, LIBLINEAR, SVMLight

What about multi-class SVMs?

- Many options!
- For example, we have to obtain a multi-class SVM by combining multiple two-class SVMs
- One vs. rest
  - **Training**: learn an SVM for each class vs. the rest
  - **Testing**: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- One vs. one
  - **Training**: learn an SVM for each pair of classes
  - **Testing**: each learned SVM "votes" for a class to assign to the test example
  - **http://www.kernel-machines.org/software**

Lecture outline

- Origin and motivation of the “bag of words” model
- Algorithm pipeline
  - Extracting local features
  - Learning a dictionary — clustering using k-means
  - Encoding methods — hard vs. soft assignment
  - Spatial pooling — pyramid representations
  - Similarity functions and classifiers

Putting it all together

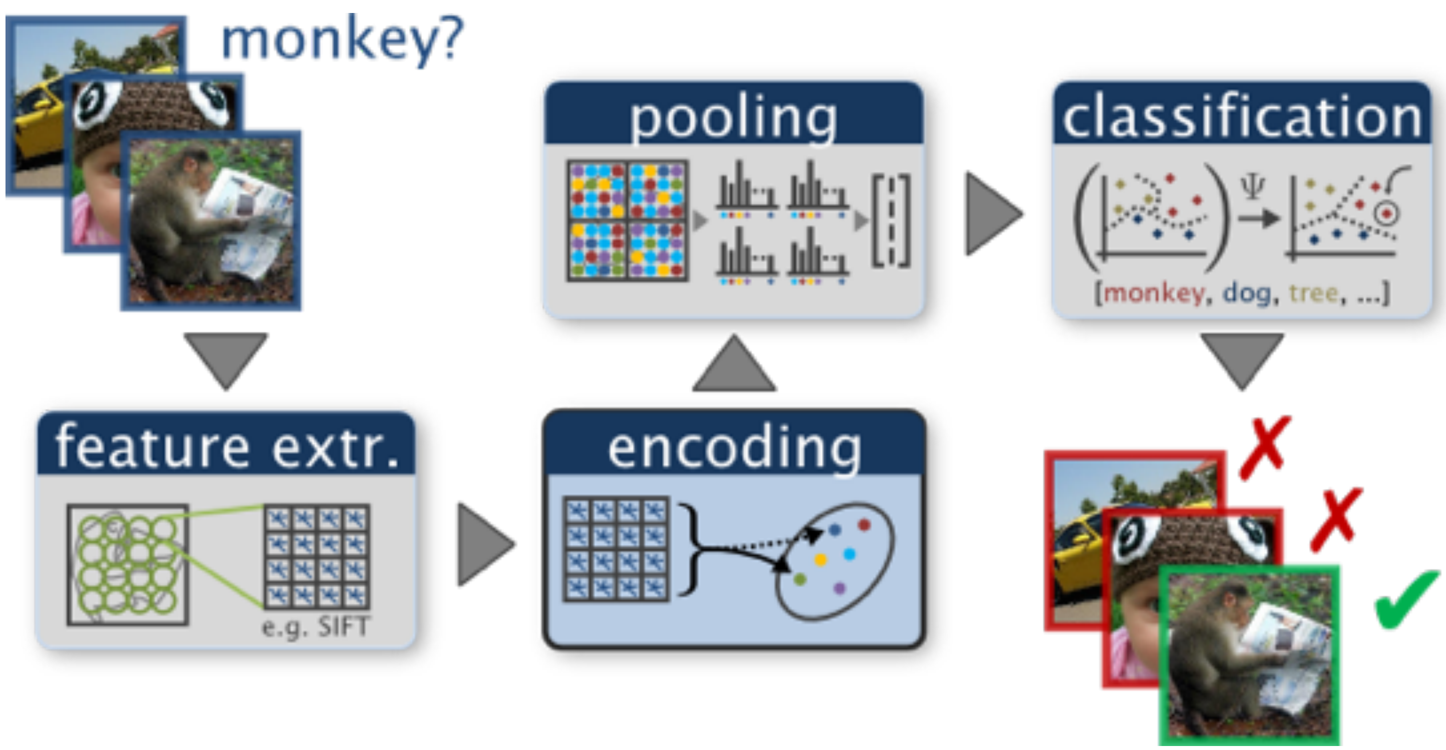


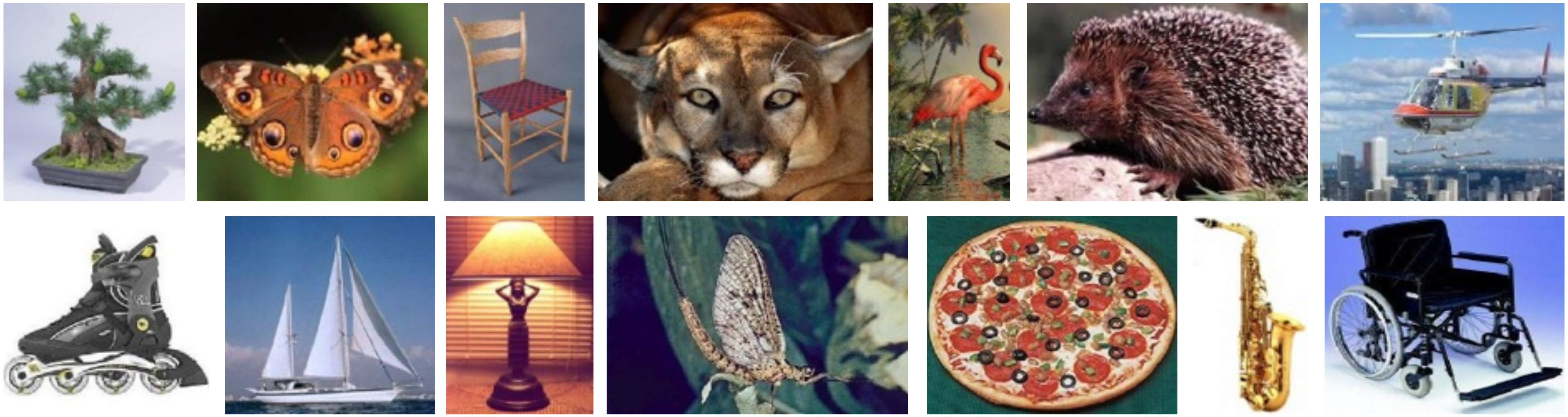
Figure from *Chatfield et al.,2011*

Results: scene category dataset

Multi-class classification results
(100 training images per class)

	Weak features (vocabulary size: 16)		Strong features (vocabulary size: 200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0 (1×1)	45.3 ±0.5		72.2 ±0.6	
1 (2×2)	53.6 ±0.3	56.2 ±0.6	77.9 ±0.6	79.0 ±0.5
2 (4×4)	61.7 ±0.6	64.7 ±0.7	79.4 ±0.3	**81.1** ±0.3
3 (8×8)	63.3 ±0.8	**66.8** ±0.6	77.2 ±0.4	80.7 ±0.3

Results: Caltech-101 dataset



Multi-class classification results (30 training images per class)

	Weak features (16)		Strong features (200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0	15.5 ±0.9		41.2 ±1.2	
1	31.4 ±1.2	32.8 ±1.3	55.9 ±0.9	57.0 ±0.8
2	47.2 ±1.1	49.3 ±1.4	63.6 ±0.9	**64.6** ±0.8
3	52.2 ±0.8	**54.0** ±1.1	60.3 ±0.9	64.6 ±0.7

Further thoughts and readings ...

- All about embeddings (detailed experiments and code)
  - K. Chatfield et al., The devil is in the details: an evaluation of recent feature encoding methods, BMVC 2011
  - **http://www.robots.ox.ac.uk/~vgg/research/encoding_eval/**
  - Includes discussion of advanced embeddings such as Fisher vector representations and locally linear coding (LLC)
- All about SVMs — **http://research.microsoft.com/pubs/67119/svmtutorial.pdf**
- Fast non-linear SVM evaluation (scales linearly with #SVs)
  - Classification using Intersection kernel SVMs is efficient, Maji et al., CVPR 2008 — O(1) evaluation ~ **1000x** faster on on large datasets! (Also see the PAMI 2013 paper on my webpage)
  - Approximate embeddings for kernels (Maji and Berg, Vedaldi and Zisserman) — O(n) training ~ **100x** faster on large datasets!