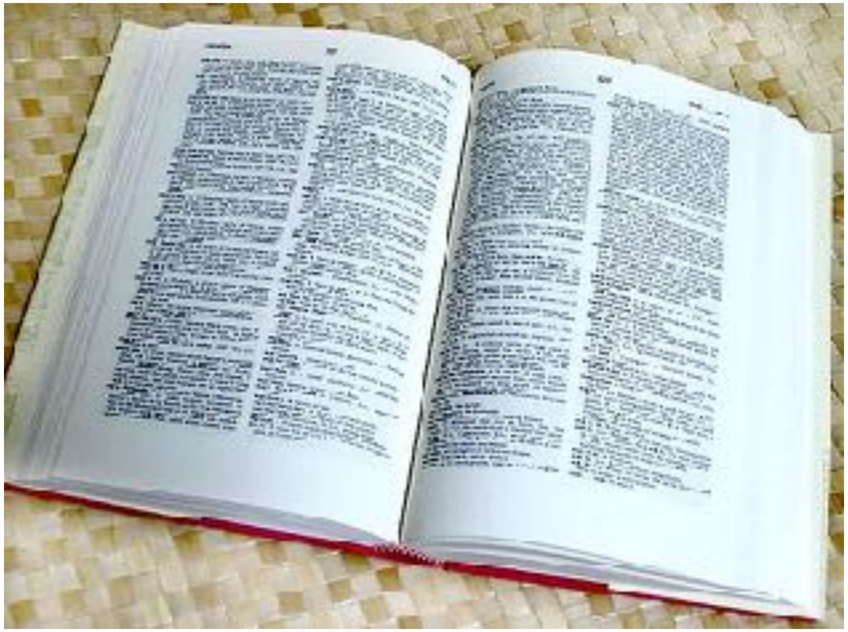
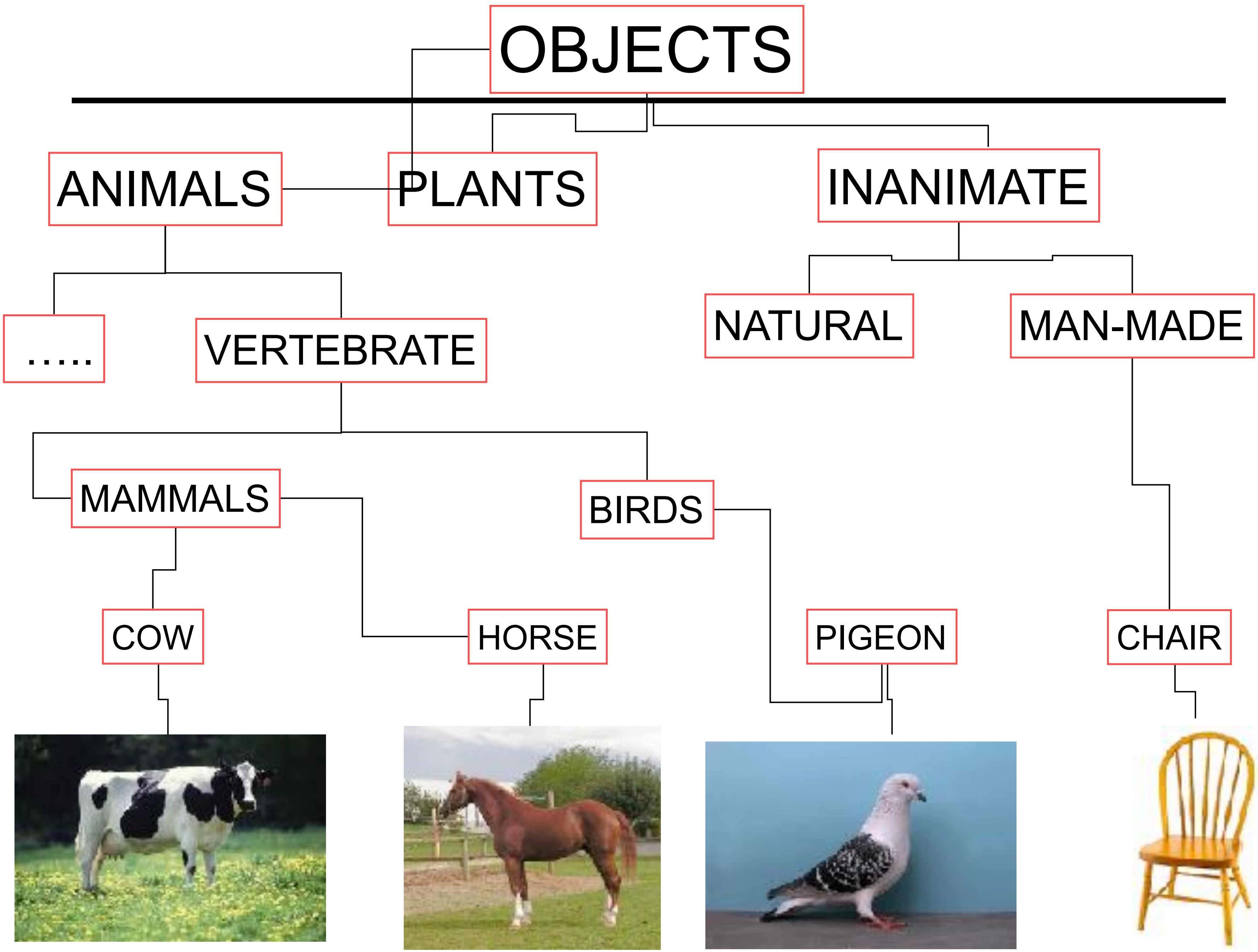
RECOGNITION

Thanks to Svetlana Lazebnik and Andrew Zisserman for the use of some slides

How many categories?



10,000-30,000


OBJECTS
ANIMALS
PLANTS
INANIMATE
.....
VERTEBRATE
NATURAL
MAN-MADE
MAMMALS
BIRDS
COW
HORSE
PIGEON
CHAIR

Variability makes recognition hard

Camera position
Illumination
Shape parameters

Within-class variations?

Variations within the same class






History

1960s – early 1990s: geometry

1990s: appearance

Mid-1990s: sliding window

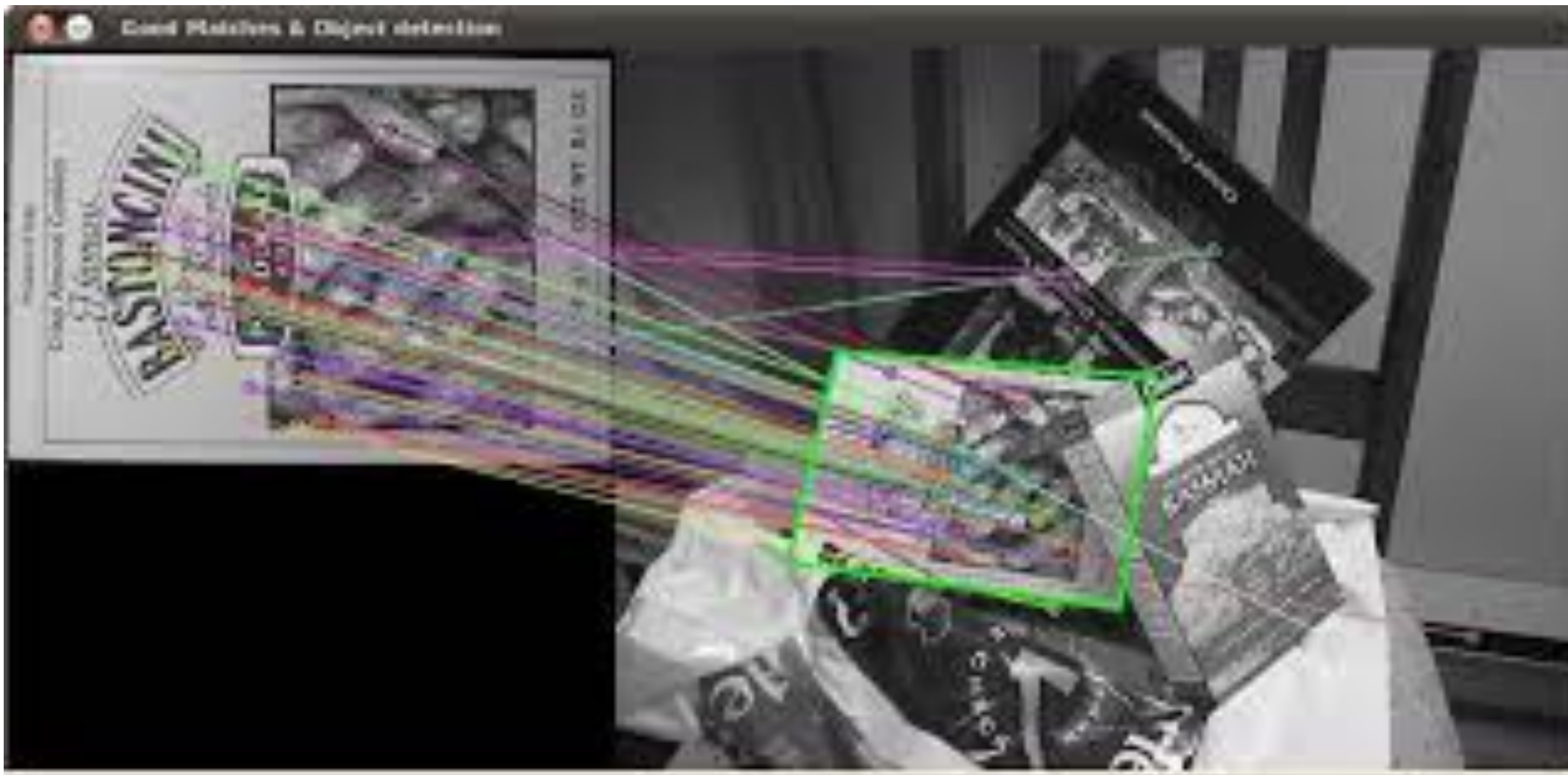
Late 1990s: local features

Early 2000s: parts-and-shape models

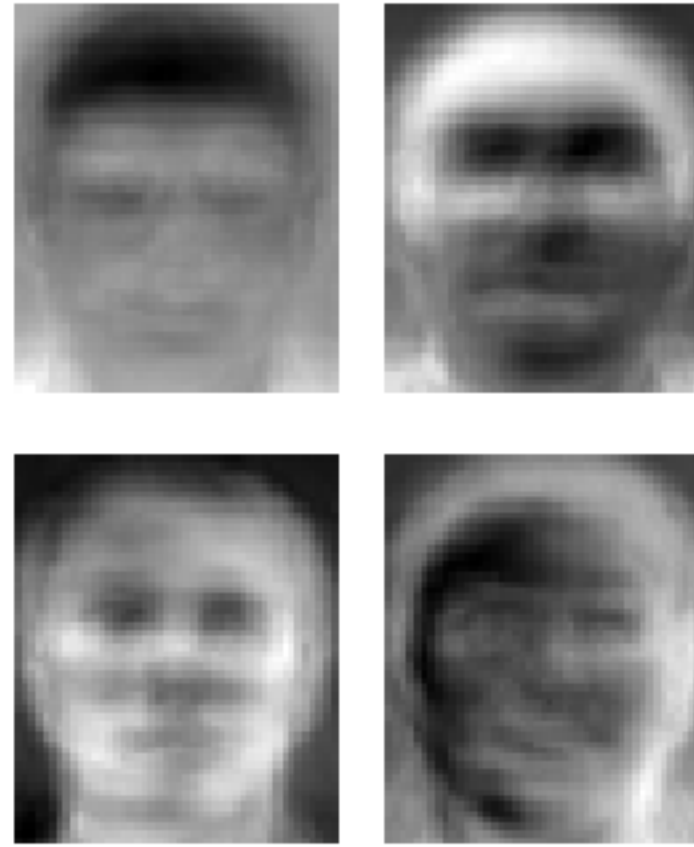
Mid-2000s: bags of features

Present trends: data-driven methods, context

2D objects



Eigenfaces (Turk & Pentland, 1991)



Experimental Condition	Correct/Unknown Recognition Percentage		
	Lighting	Orientation	Scale
Forced classification	96/0	85/0	64/0
Forced 100% accuracy	100/19	100/39	100/60
Forced 20% unknown rate	100/20	94/20	74/20

Local features

D. Lowe (1999, 2004)

Image classification

The statistical learning framework

- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

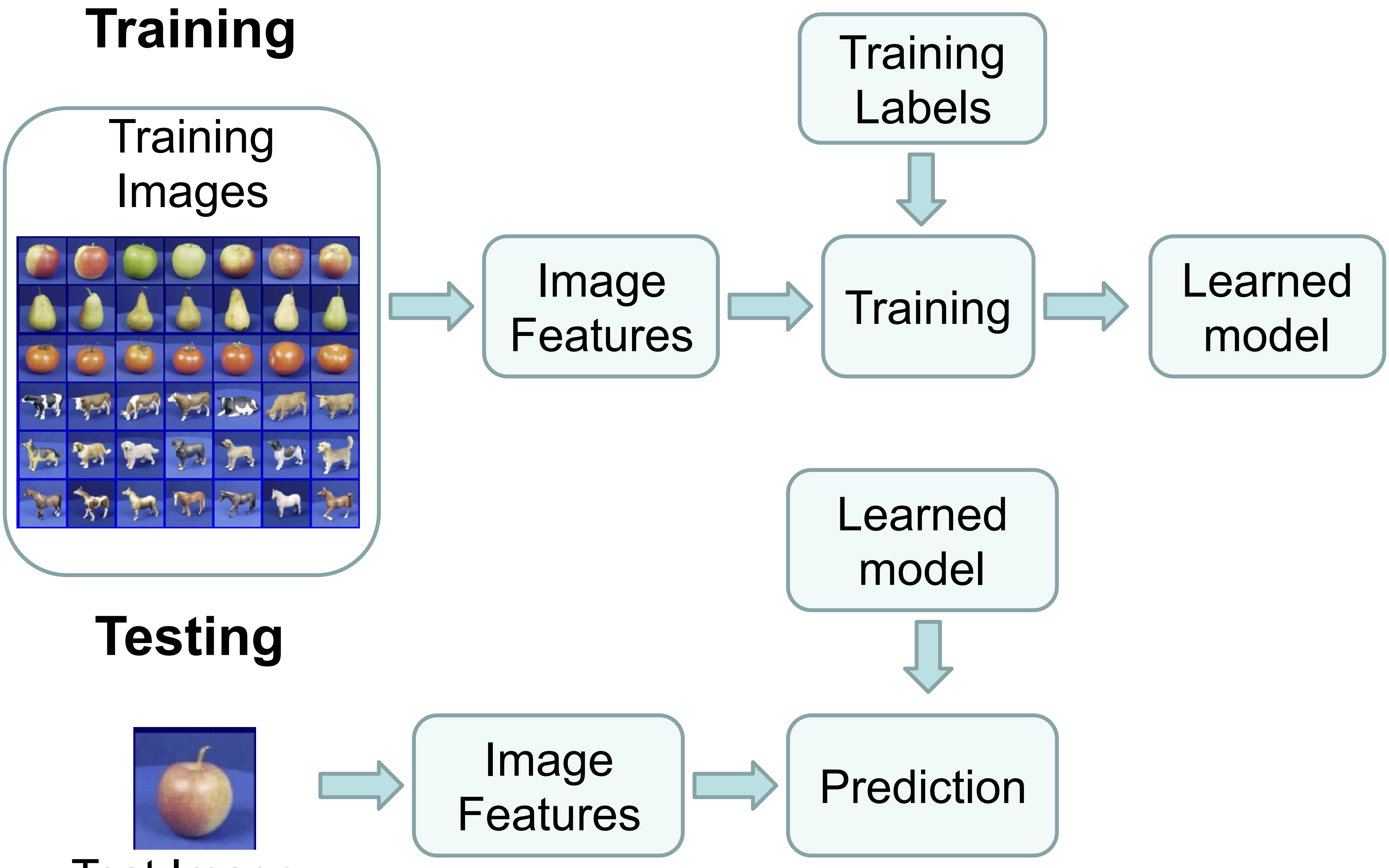
f() = “cow”

The statistical learning framework

$$\underbrace{y}_{\text{output}} = \underbrace{f}_{\text{prediction function}}(\underbrace{\mathbf{x}}_{\text{Image feature}})$$

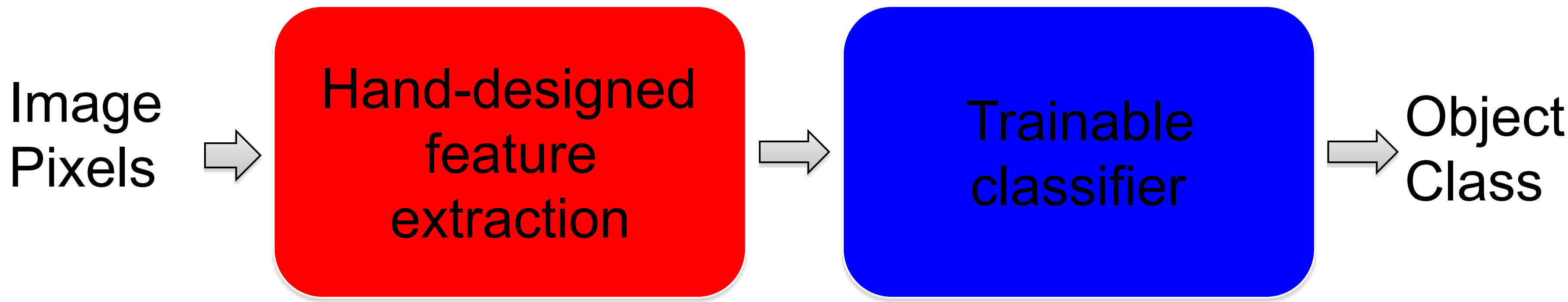
- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Steps



Slide credit: D. Hoiem

Traditional recognition pipeline

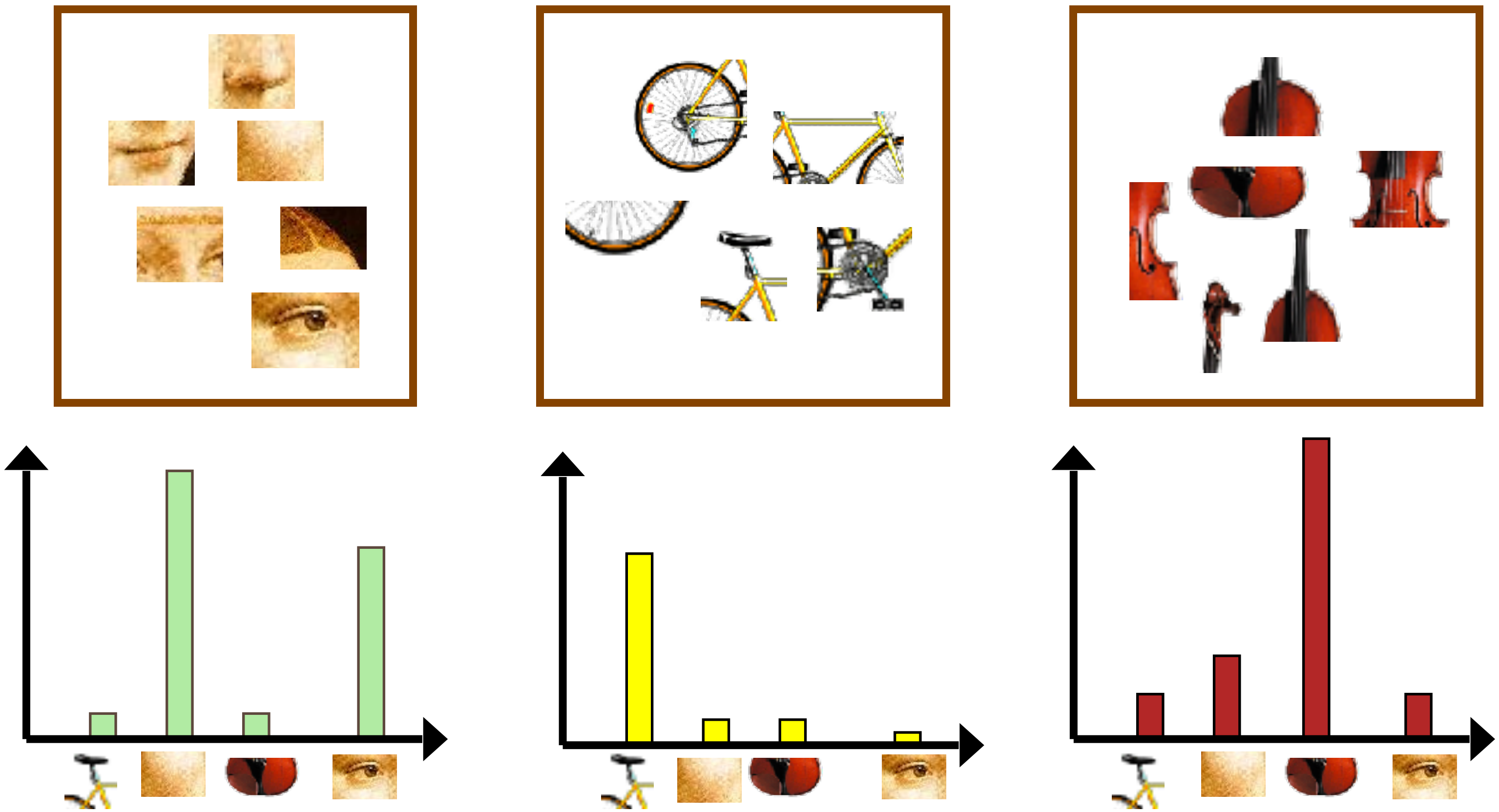


- Features are not learned
- Trainable classifier is often generic (e.g. SVM)

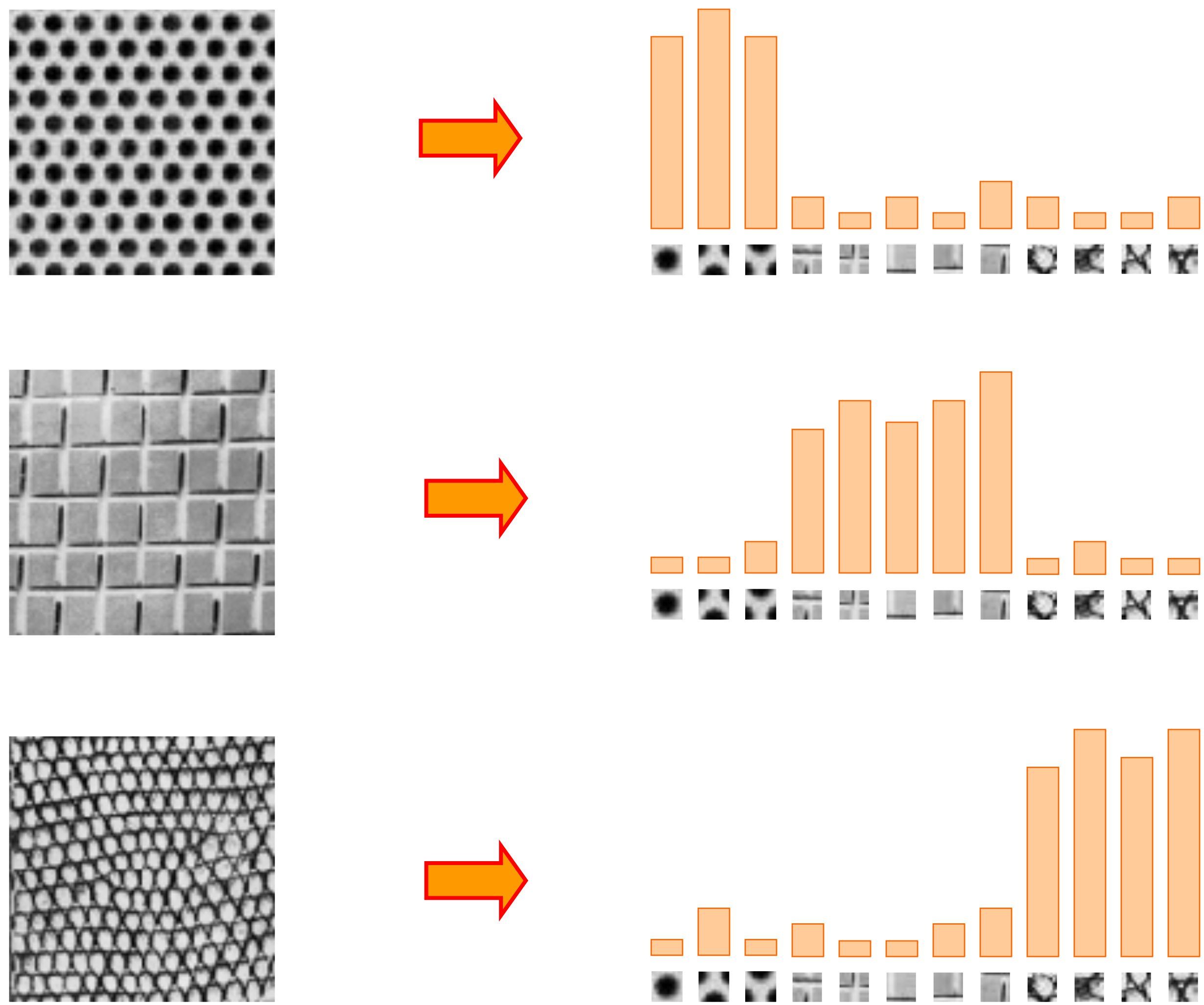
Bags of features

Traditional features: Bags-of-features

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



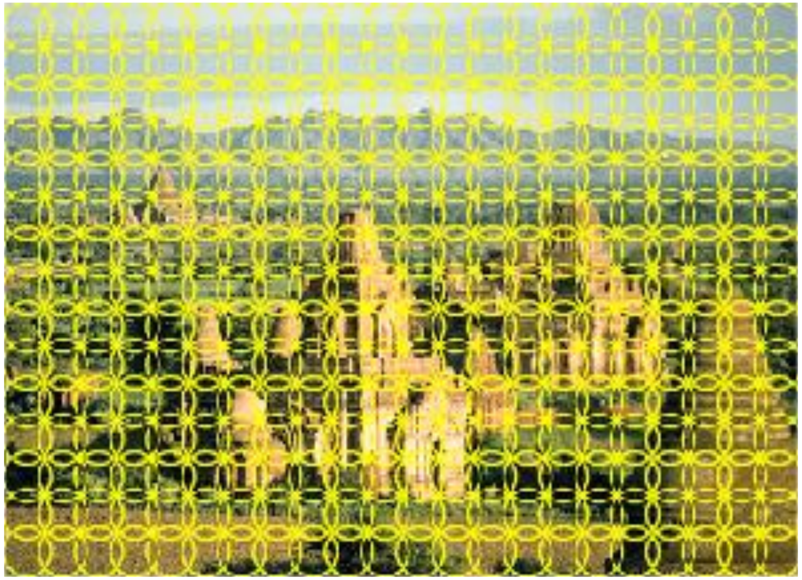
Texture recognition



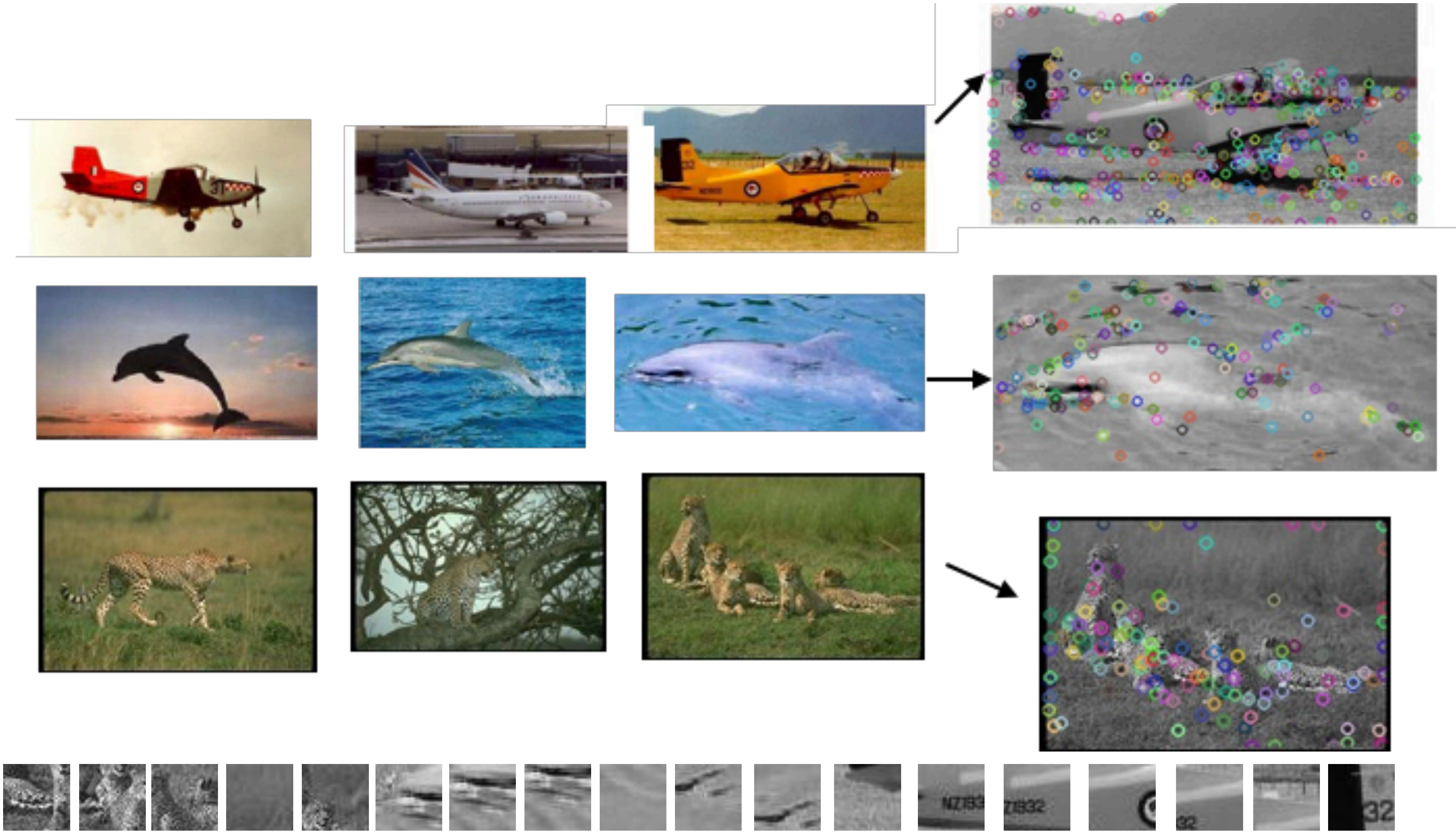
Julesz 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

1. Local feature extraction

- Sample patches and extract descriptors

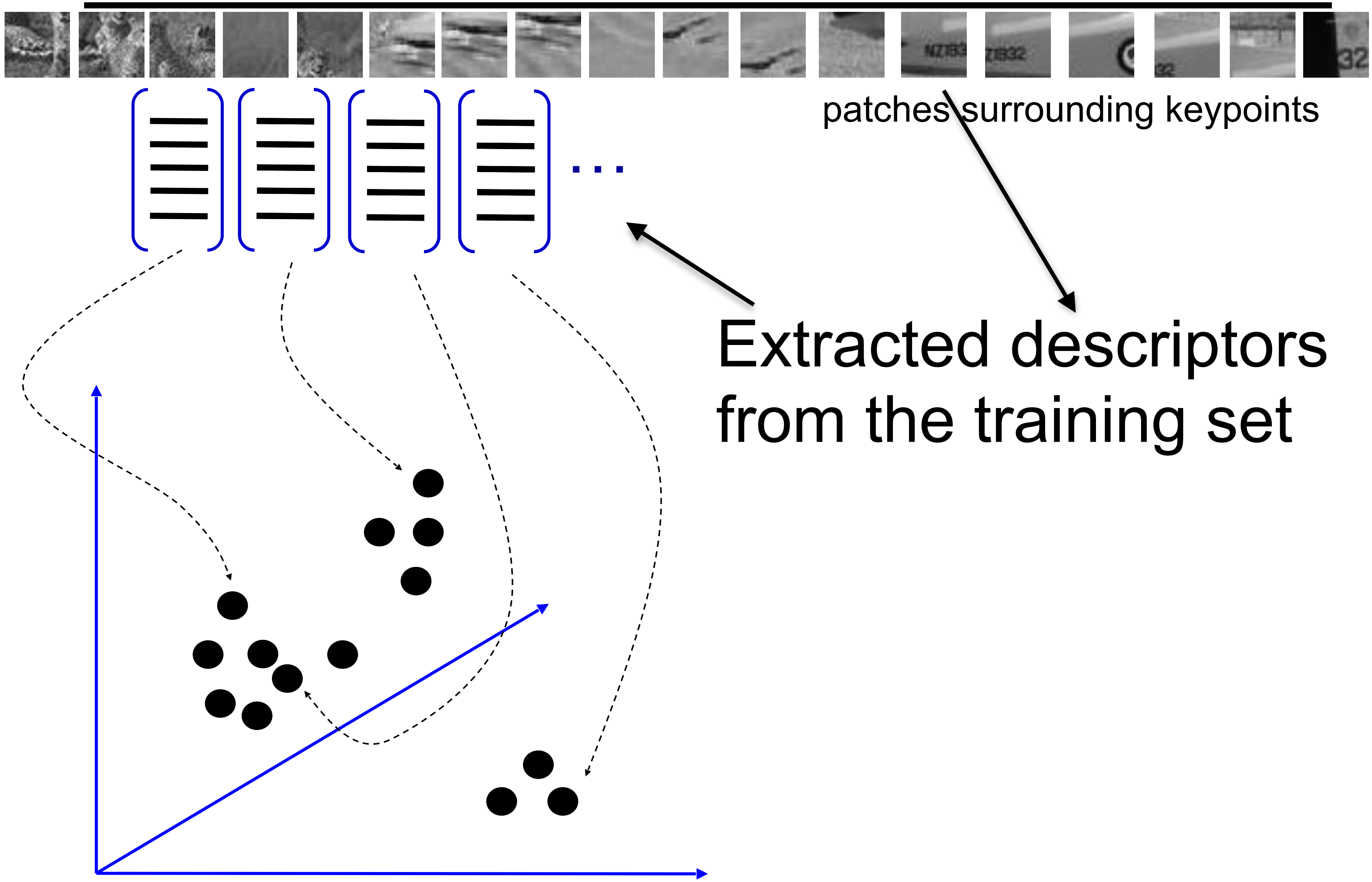


Keypoints



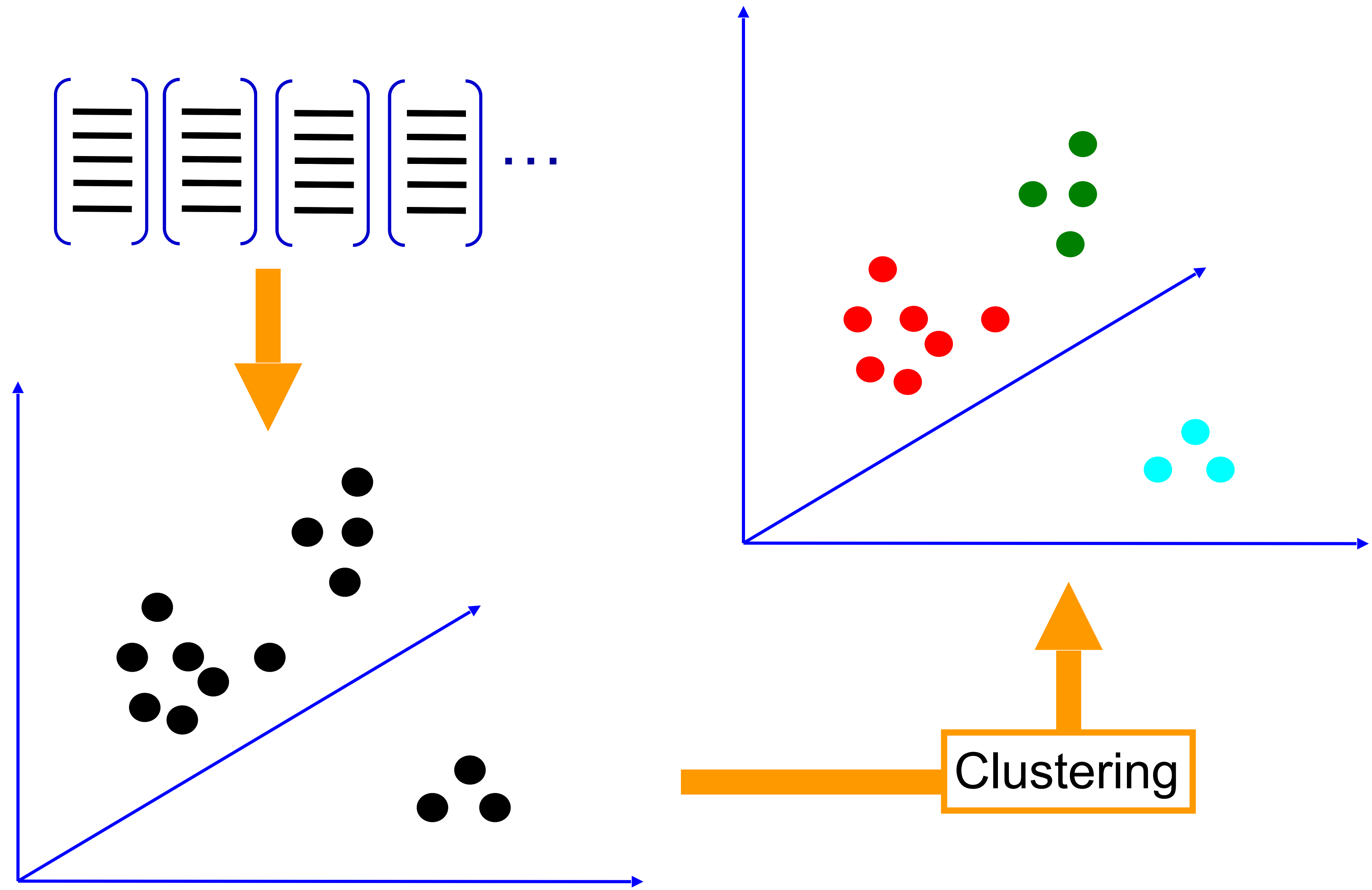
patches surrounding keypoints

2. Learning the visual vocabulary



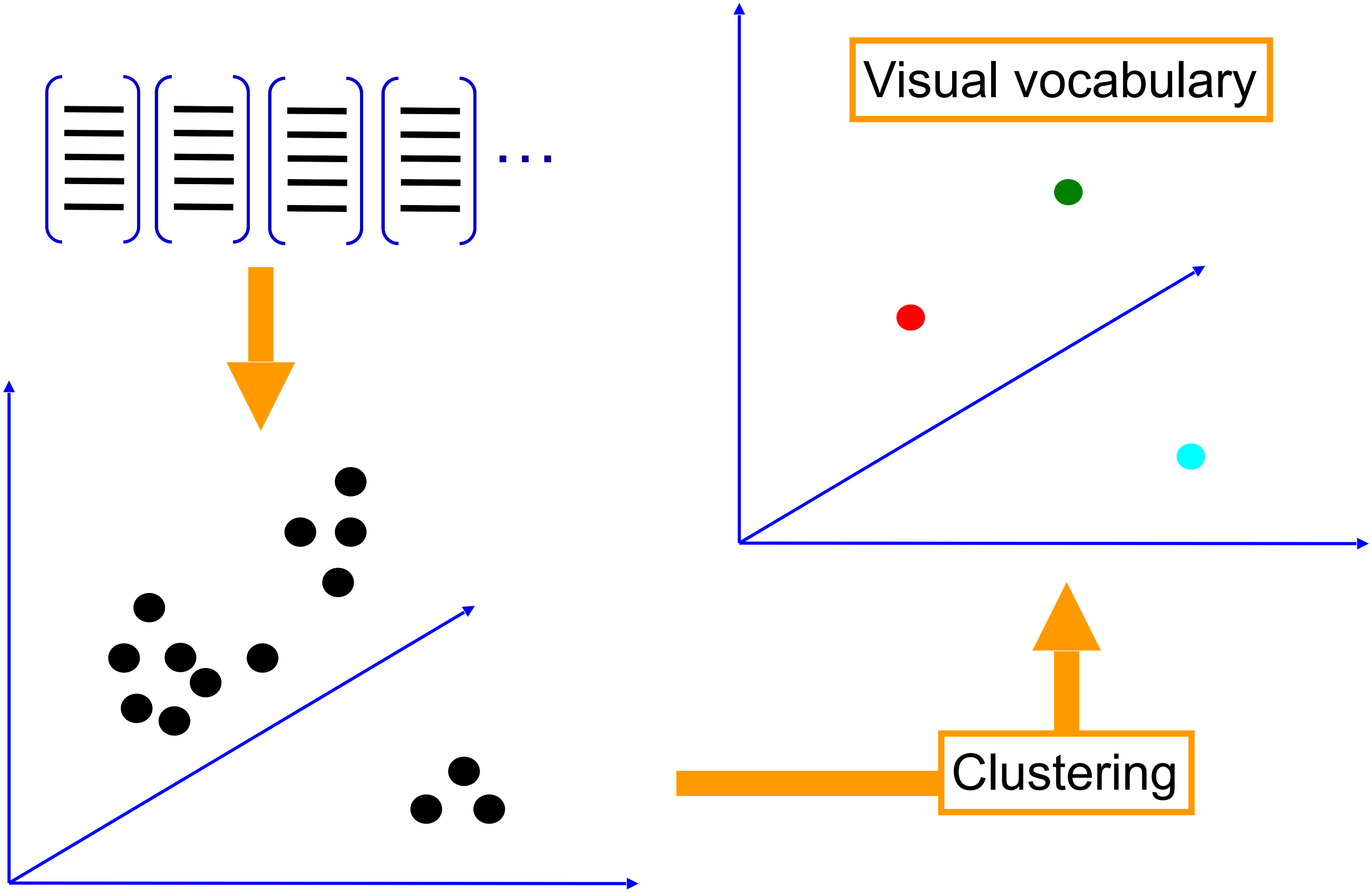
Slide credit: Josef Sivic except for the image patches

2. Learning the visual vocabulary



Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

Review: K-means clustering

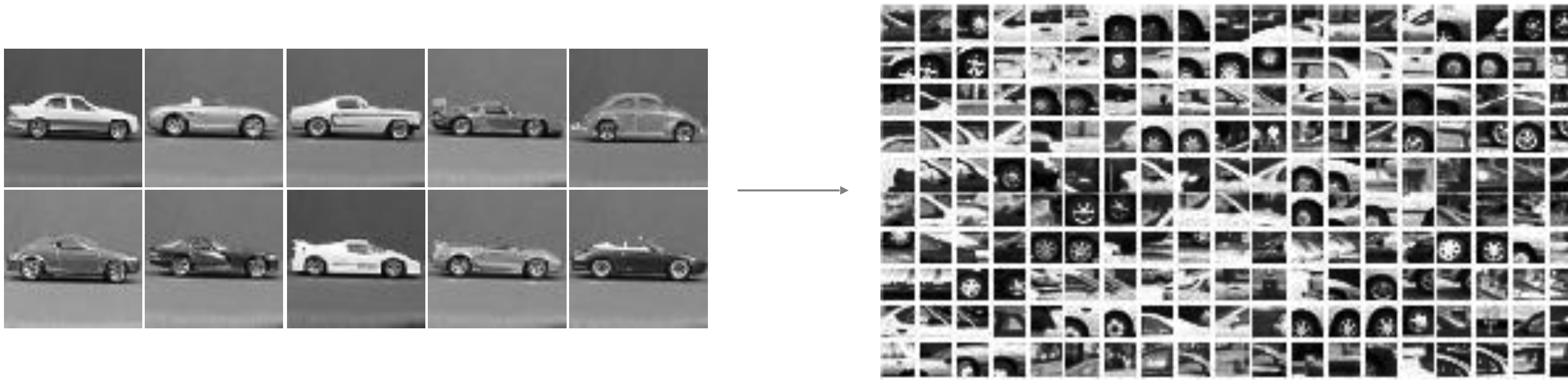
- Want to minimize sum of squared Euclidean distances between features $\mathbf{x}_i$ and their nearest cluster centers $\mathbf{m}_k$

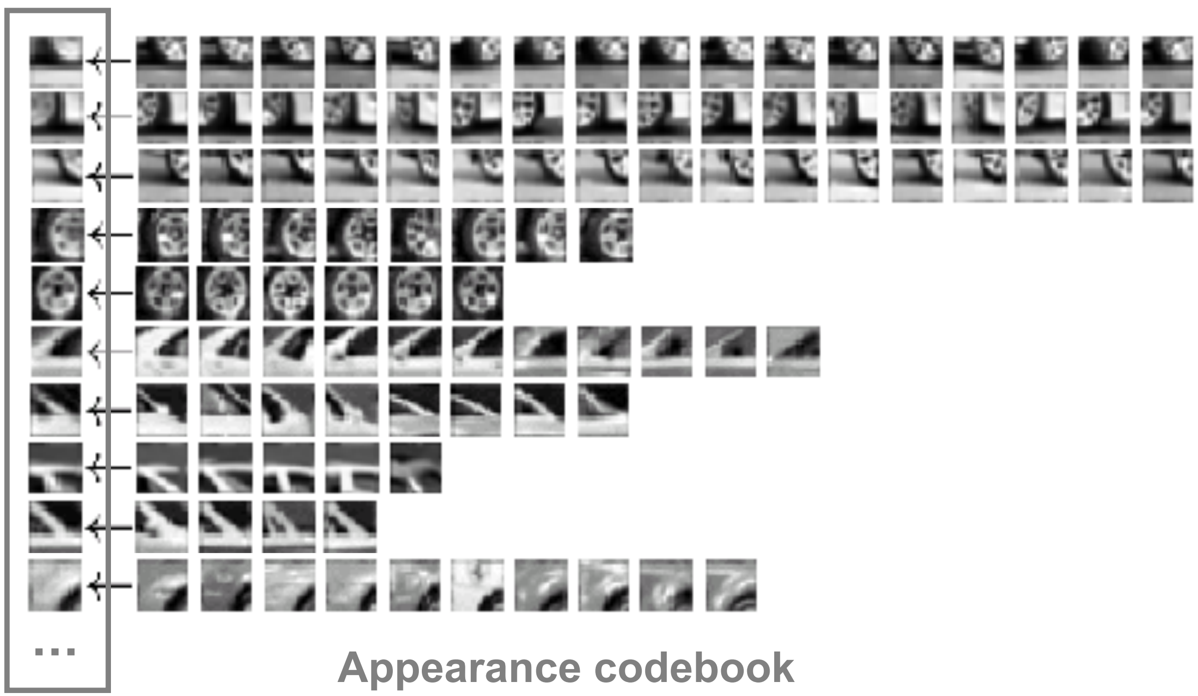
$$D(X,M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (\mathbf{x}_i - \mathbf{m}_k)^2$$

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 visual vocabulary

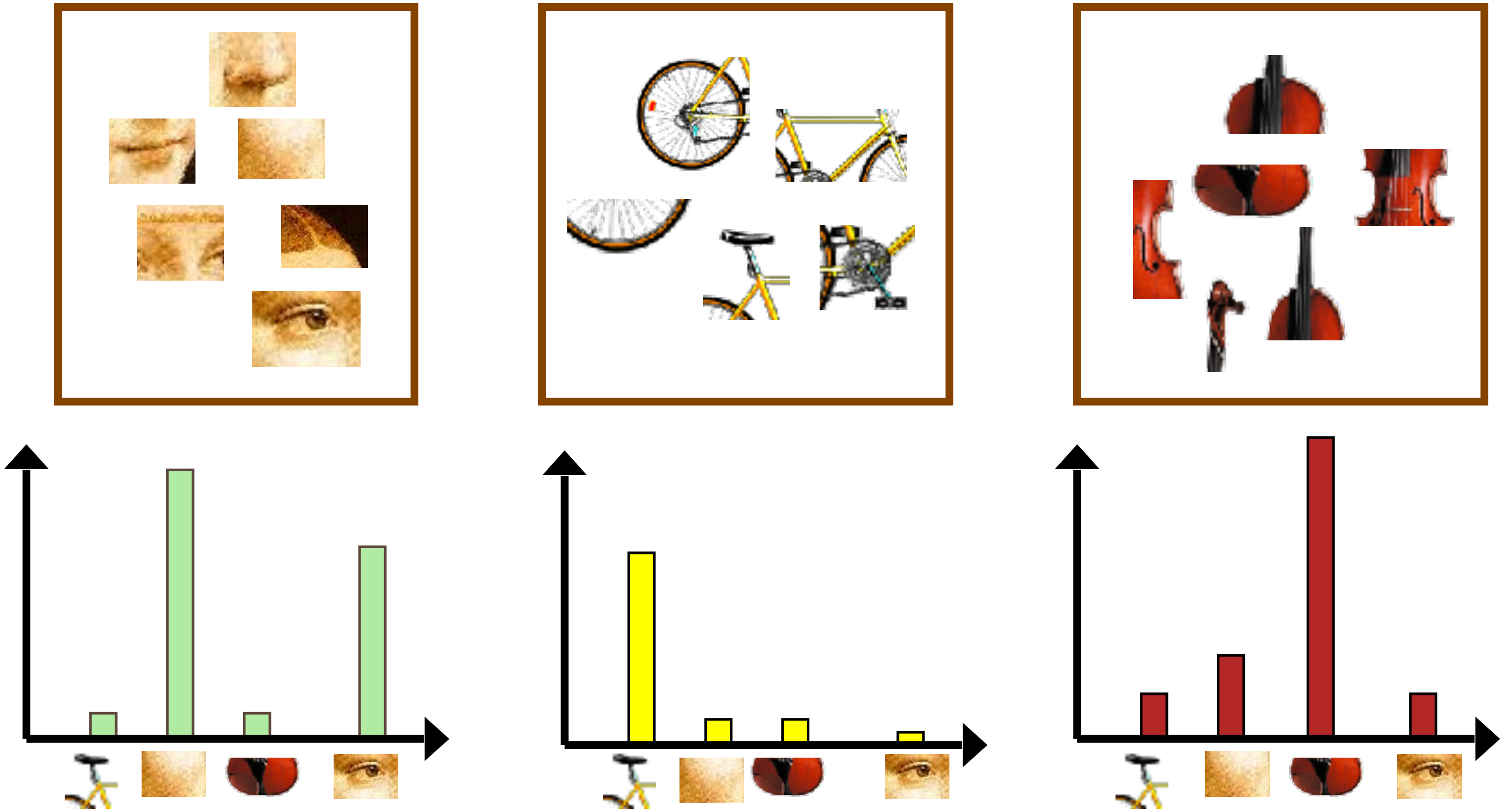




Source: B. Leibe

Bag-of-features steps

1. Extract local features
2. Learn “visual vocabulary”
3. **Quantize local features using visual vocabulary**
4. **Represent images by frequencies of “visual words”**



Bags of features: Motivation

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

Bags of features: Motivation

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

Bags of features: Motivation

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

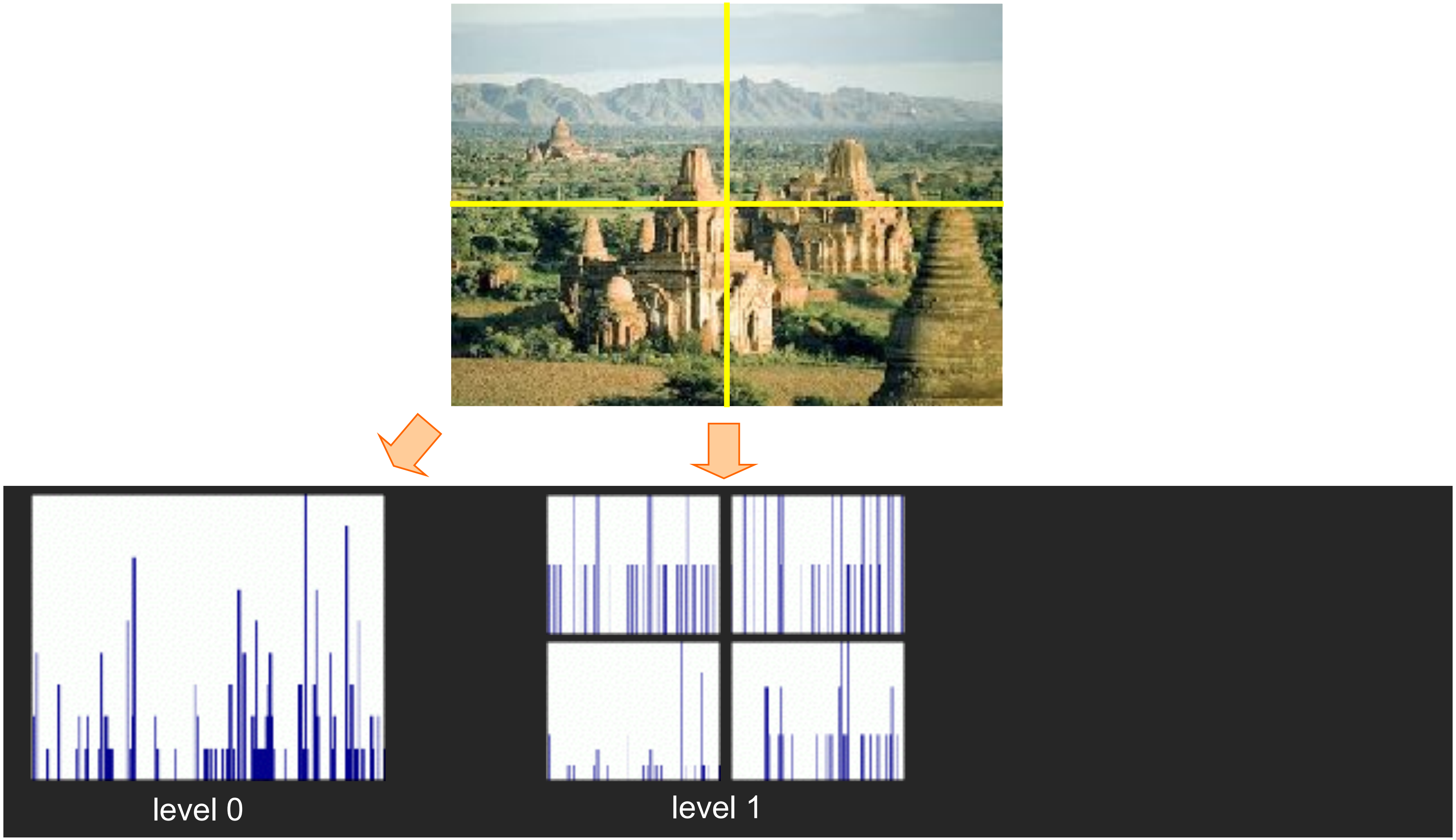
Bags of features: Motivation

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

Spatial pyramids

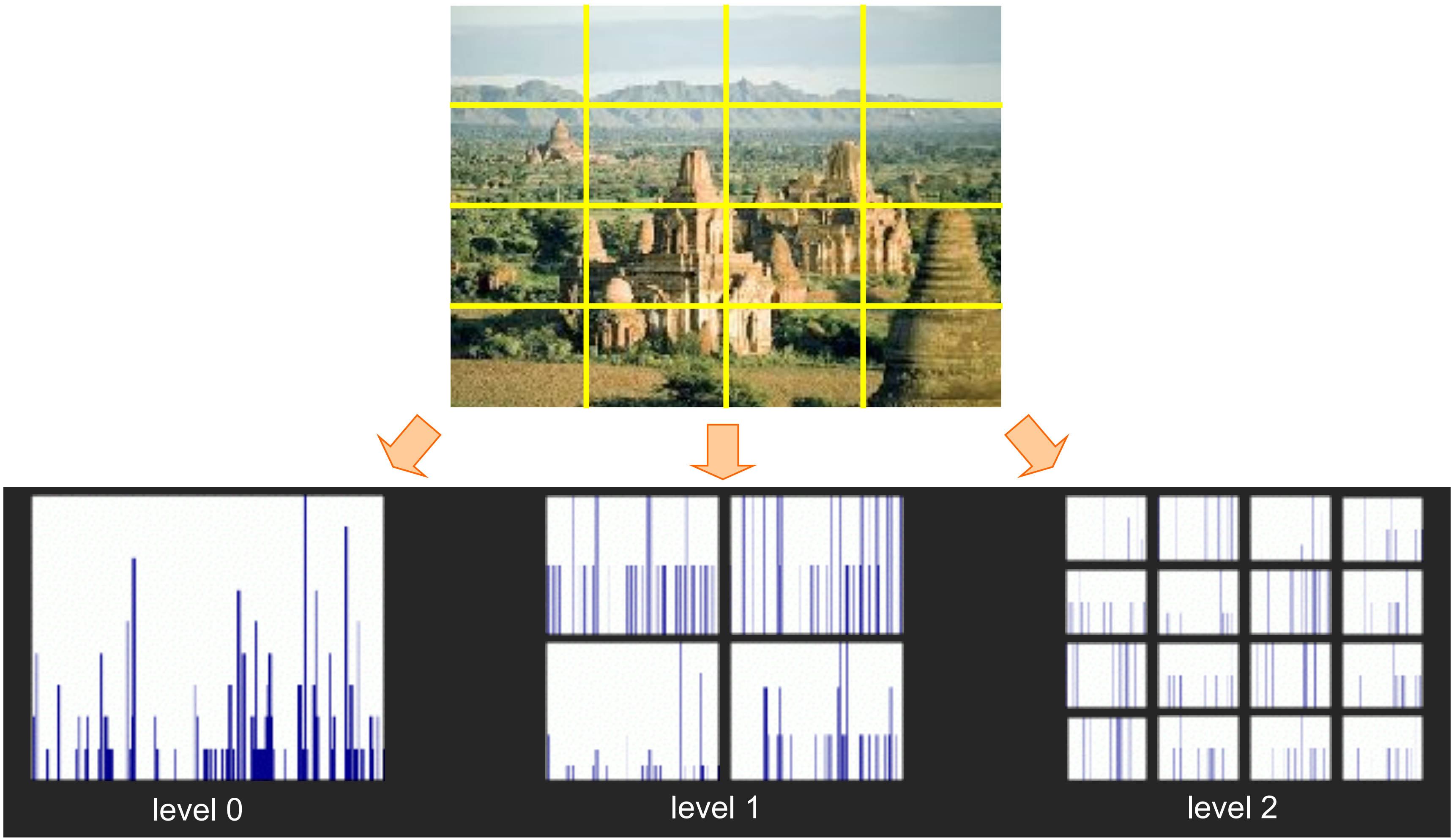
Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramids



Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramids



Lazebnik, Schmid & Ponce (CVPR 2006)

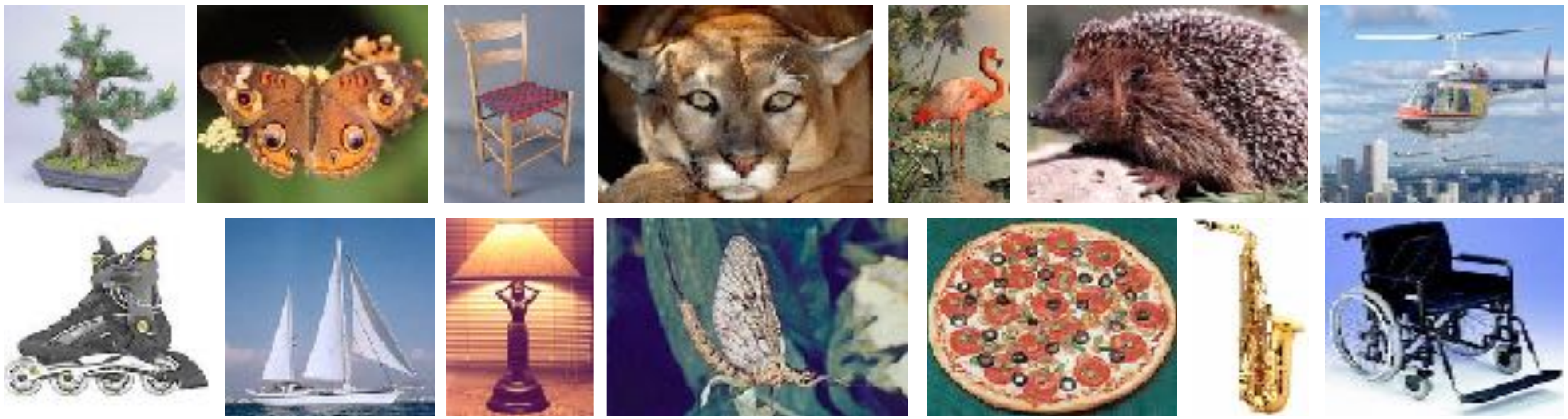
Spatial pyramids

- Scene classification results

	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

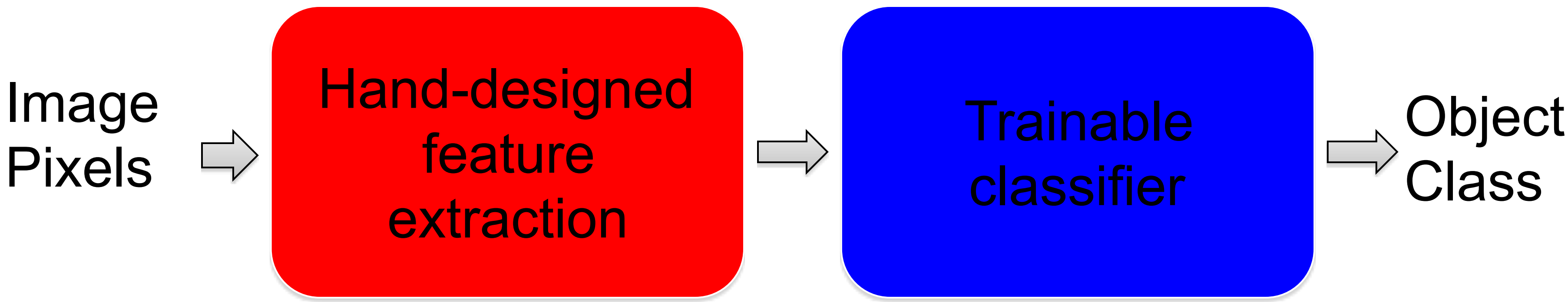
Spatial pyramids

- Caltech101 classification results

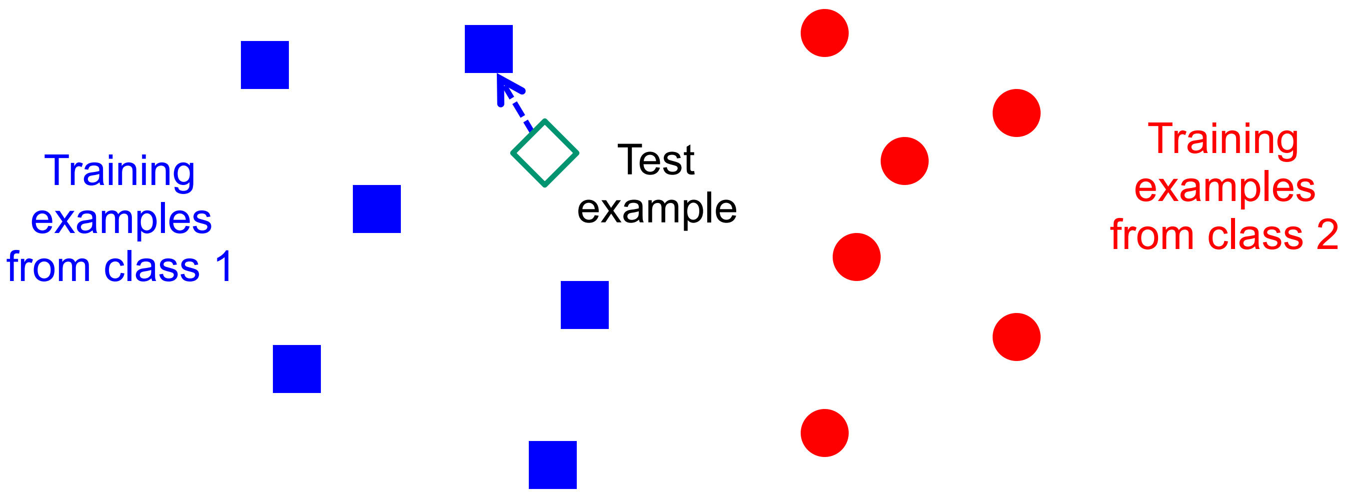


	Weak features (16)		Strong features (200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0	15.5 ±0.9		41.2 ±1.2	
1	31.4 ±1.2	32.8 ±1.3	55.9 ±0.9	57.0 ±0.8
2	47.2 ±1.1	49.3 ±1.4	63.6 ±0.9	**64.6** ±0.8
3	52.2 ±0.8	**54.0** ±1.1	60.3 ±0.9	64.6 ±0.7

Traditional recognition pipeline



Classifiers: Nearest neighbor



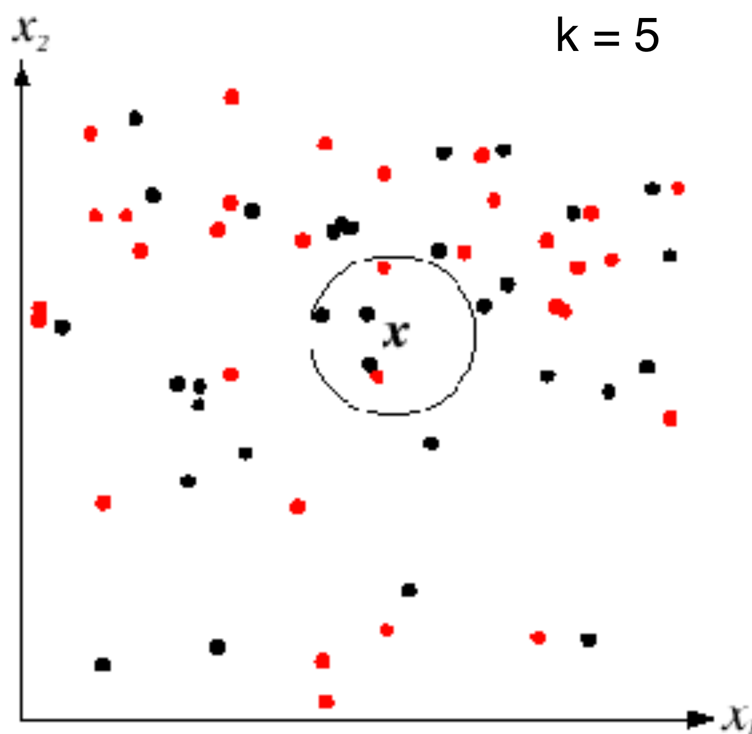
f(**x**) = label of the training example nearest to **x**

All we need is a distance function for our inputs

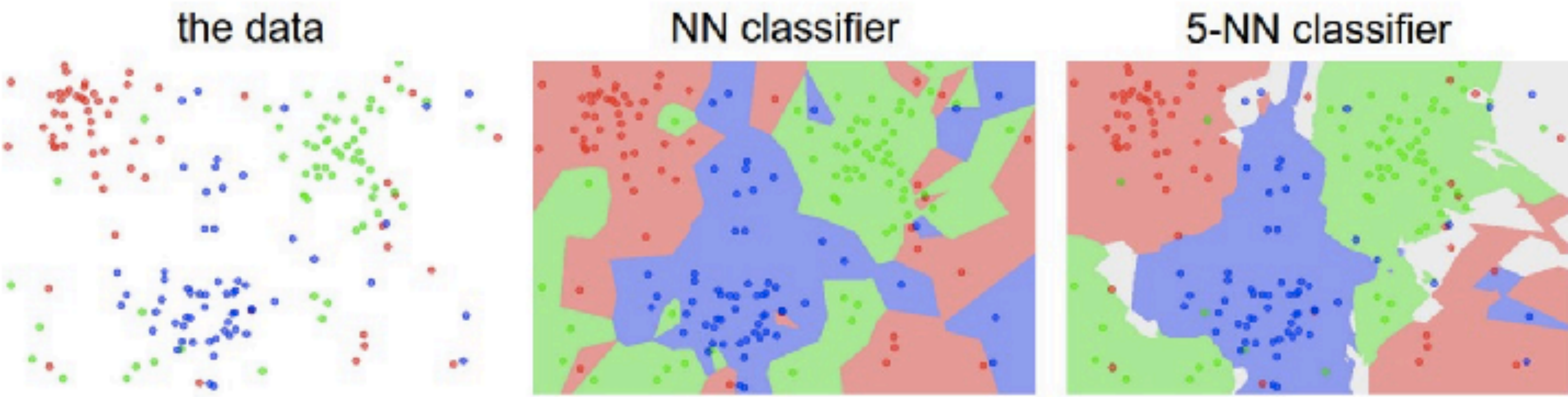
No training required!

K-nearest neighbor classifier

- For a new point, find the k closest points from training data
- Vote for class label with labels of the k points



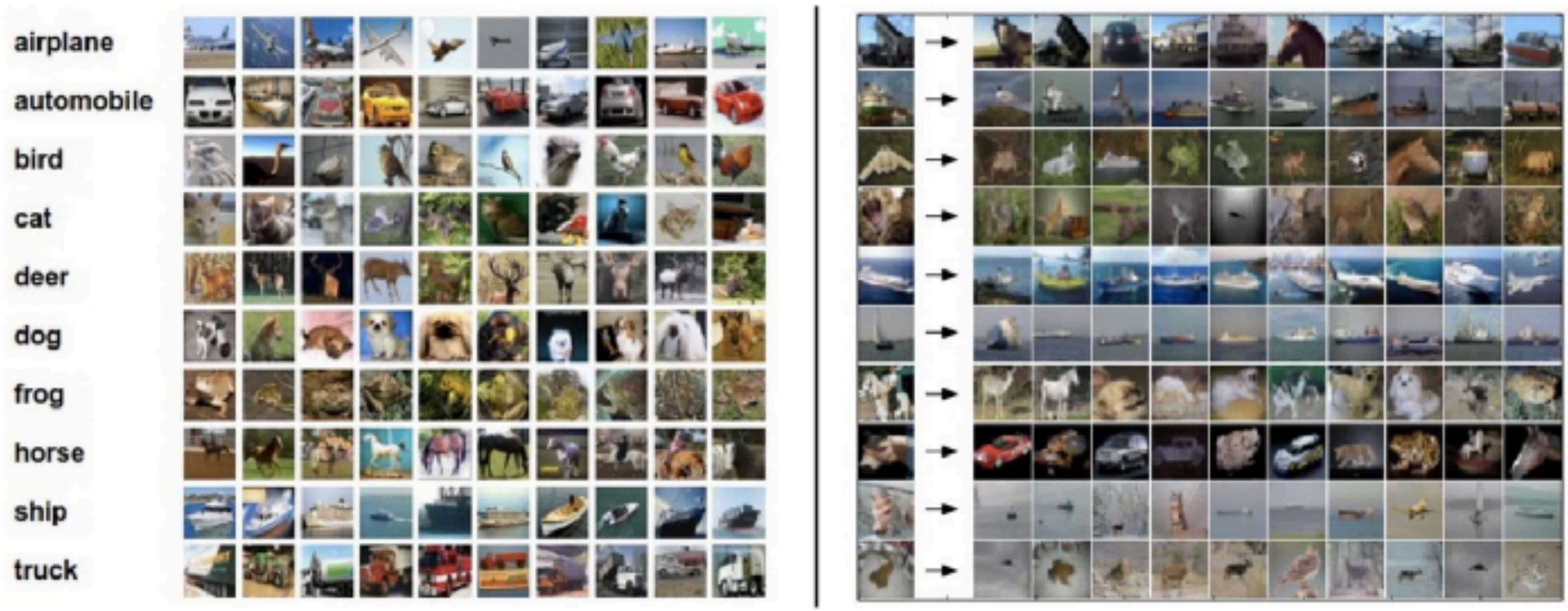
K-nearest neighbor classifier



Which classifier is more robust to *outliers*?

Credit: Andrej Karpathy, http://cs231n.github.io/classification/

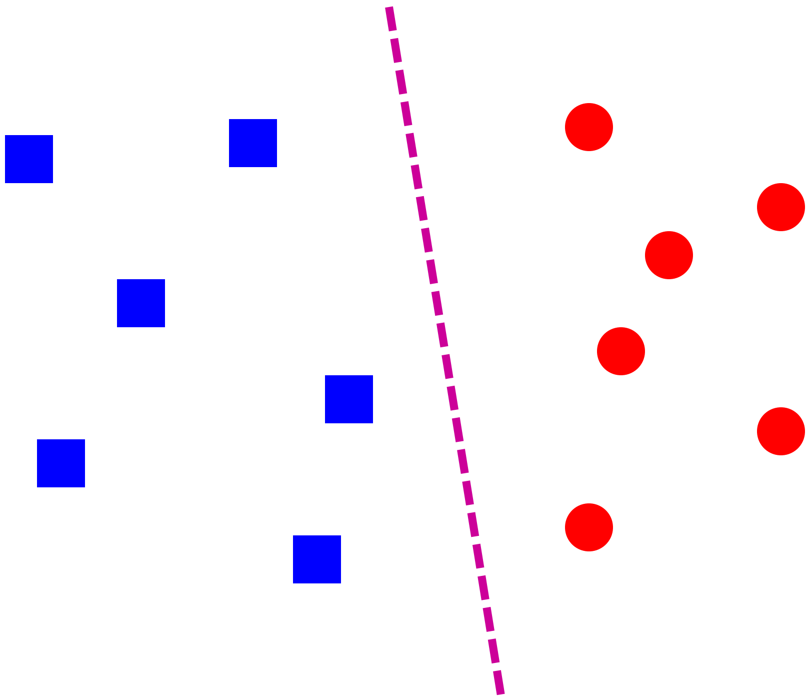
K-nearest neighbor classifier



Left: Example images from the CIFAR-10 dataset. Right: first column shows a few test images and next to each we show the top 10 nearest neighbors in the training set according to pixel-wise difference.

Credit: Andrej Karpathy, http://cs231n.github.io/classification/

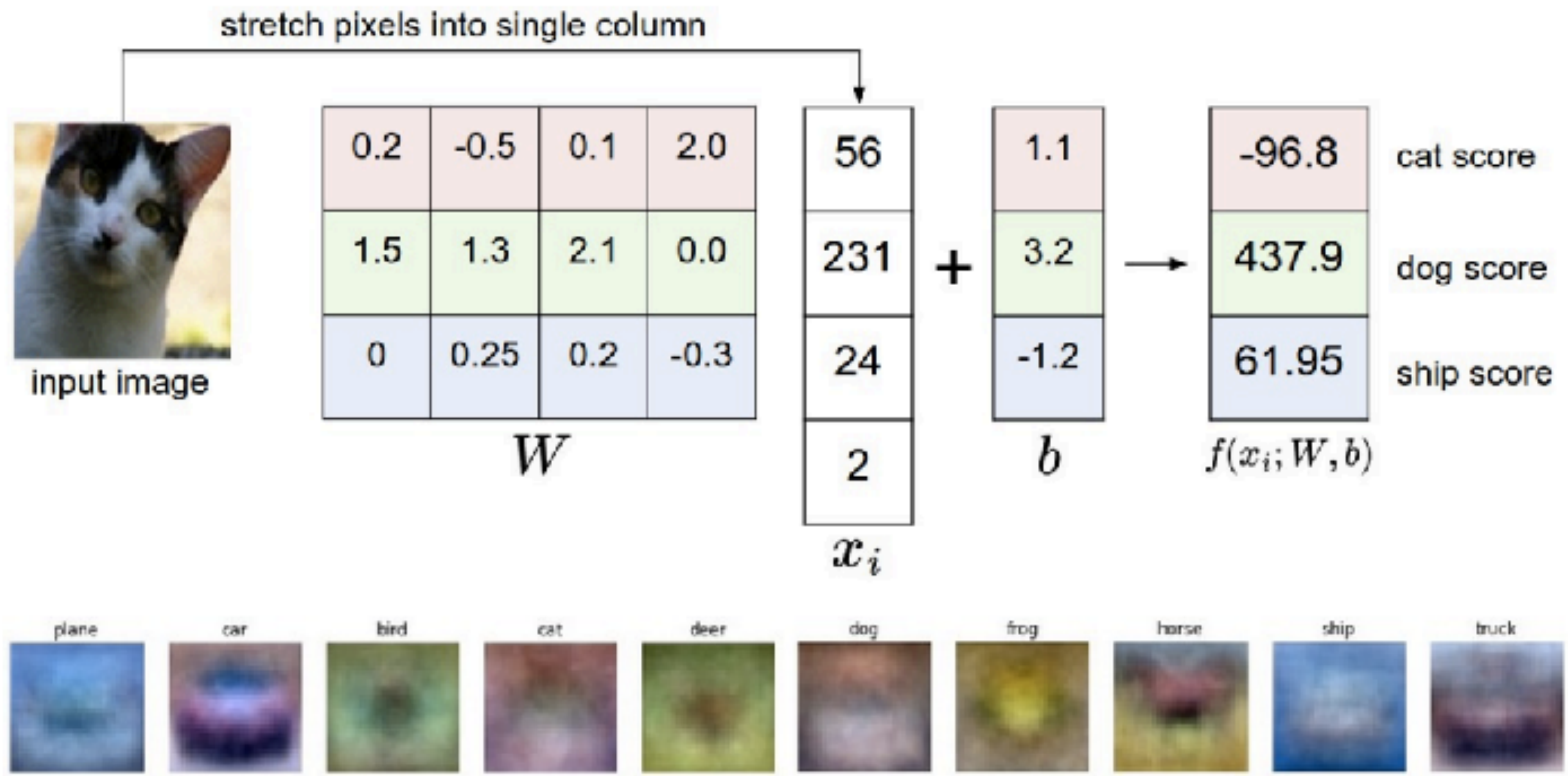
Linear classifiers



Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Visualizing linear classifiers



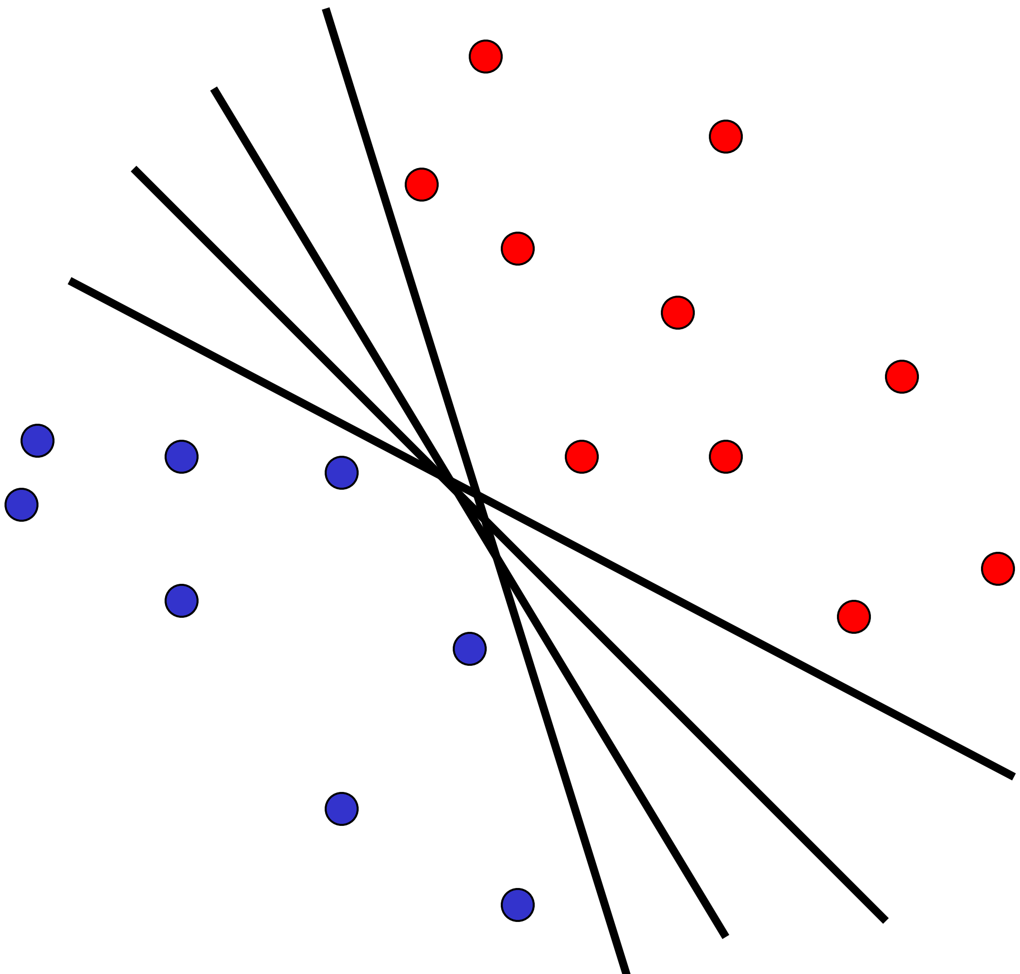
Source: Andrej Karpathy, http://cs231n.github.io/linear-classify/

Nearest neighbor vs. linear classifiers

- NN pros:
  - Simple to implement
  - Decision boundaries not necessarily linear
  - Works for any number of classes
  - *Nonparametric* method
- NN cons:
  - Need good distance function
  - Slow at test time
- Linear pros:
  - Low-dimensional *parametric* representation
  - Very fast at test time
- Linear cons:
  - Works for two classes
  - How to train the linear function?
  - What if data is not linearly separable?

Support vector machines

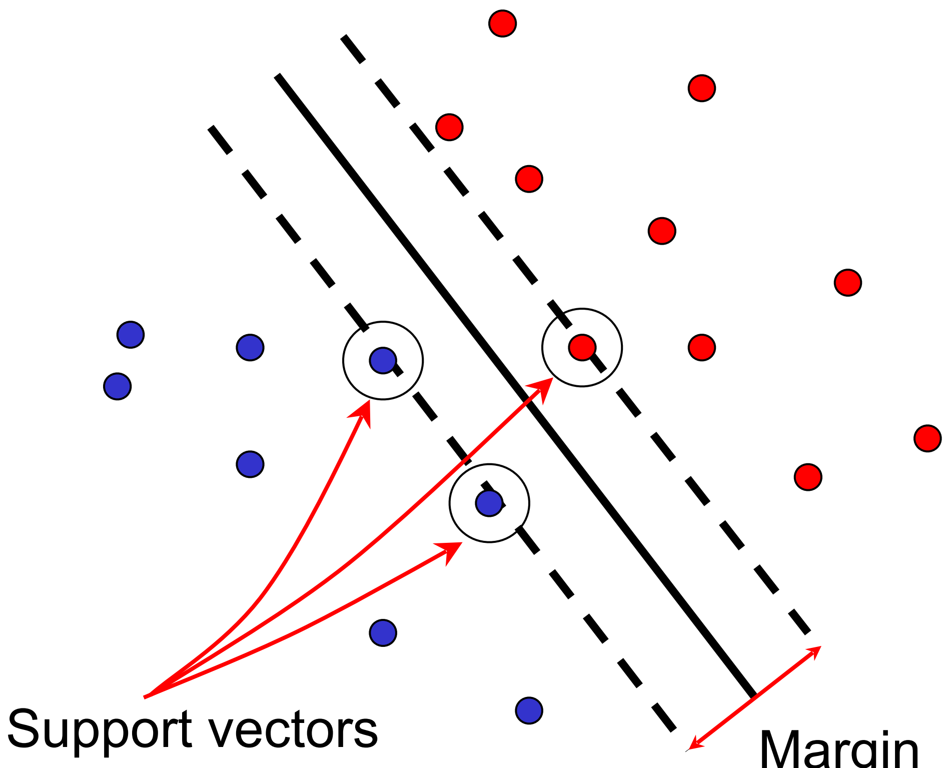
- When the data is linearly separable, there may be more than one separator (hyperplane)



Which separator is best?

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

SVM parameter learning

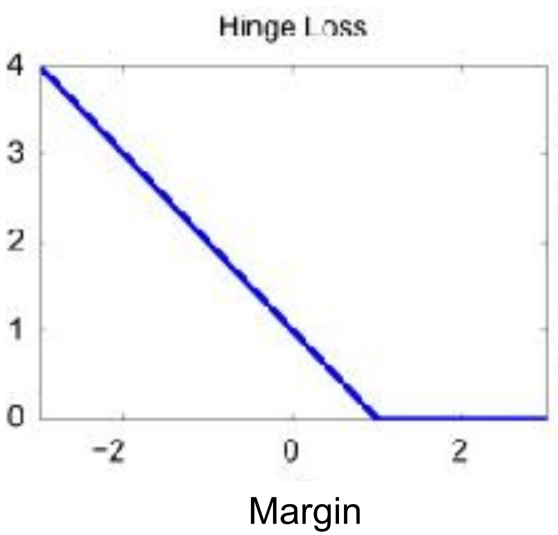
- Separable data: $$\underbrace{\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2}_{\boxed{\text{Maximize margin}}} \quad \underbrace{\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1}_{\boxed{\text{Classify training data correctly}}}$$

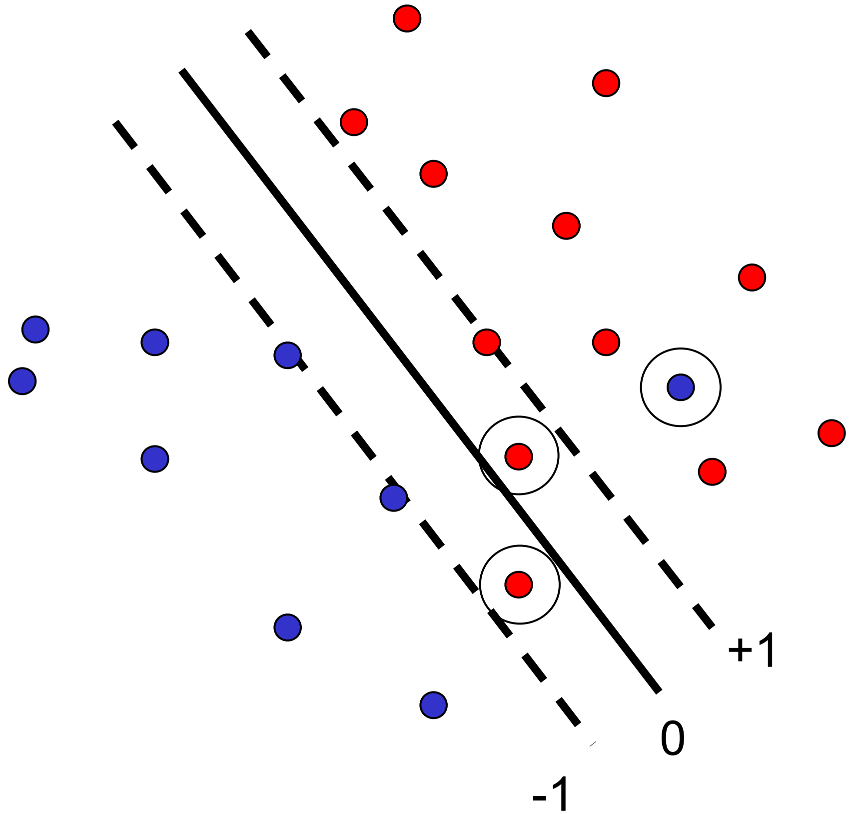
- Non-separable data:

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

SVM parameter learning

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





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

Nonlinear SVMs

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

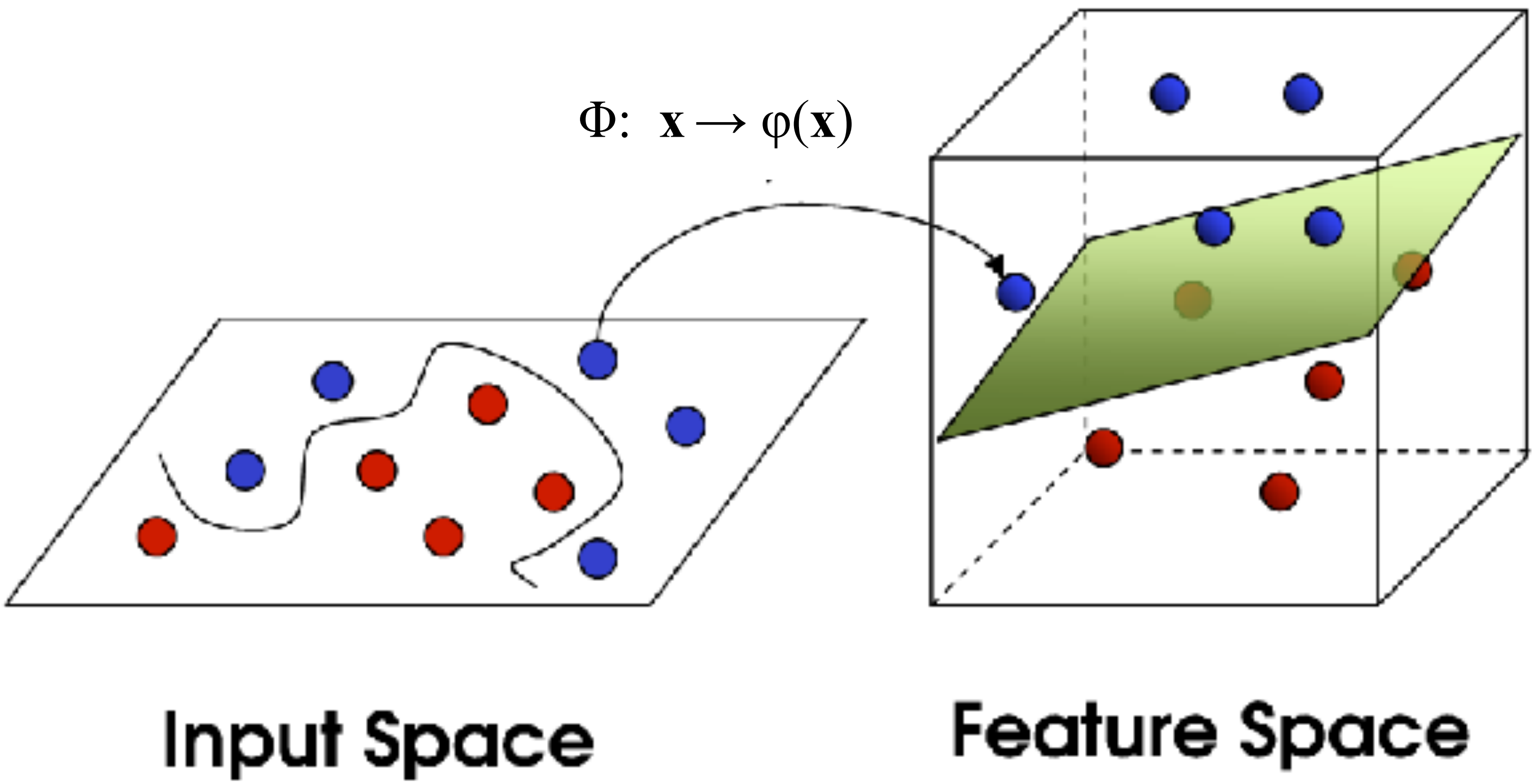
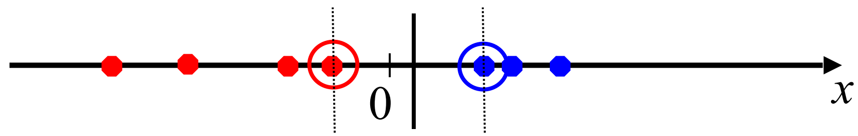


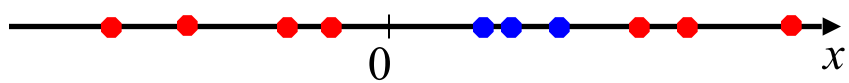
Image source

Nonlinear SVMs

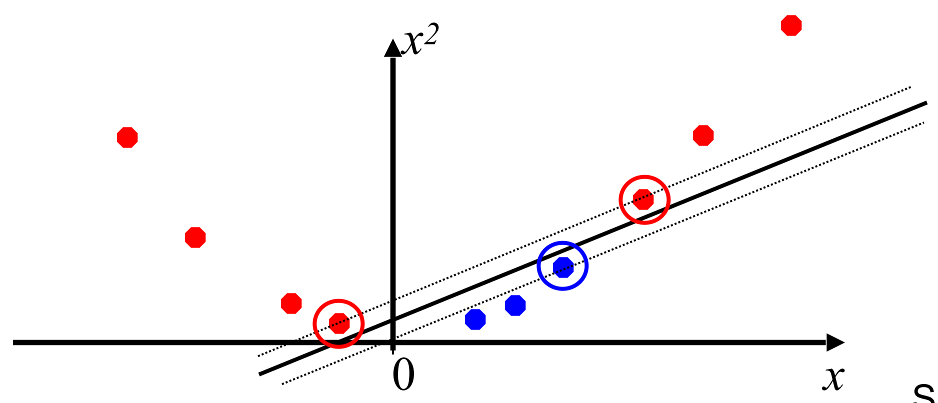
- Linearly separable dataset in 1D:



- Non-separable dataset in 1D:



- We can map the data to a *higher-dimensional space*:



Slide credit: Andrew Moore

The kernel trick

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

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

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

(to be valid, the kernel function must satisfy *Mercer's condition*)

The kernel trick

- Linear SVM decision function:

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} \;+ b$$

learned weight

Support vector

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

The kernel trick

- Linear SVM decision function:

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

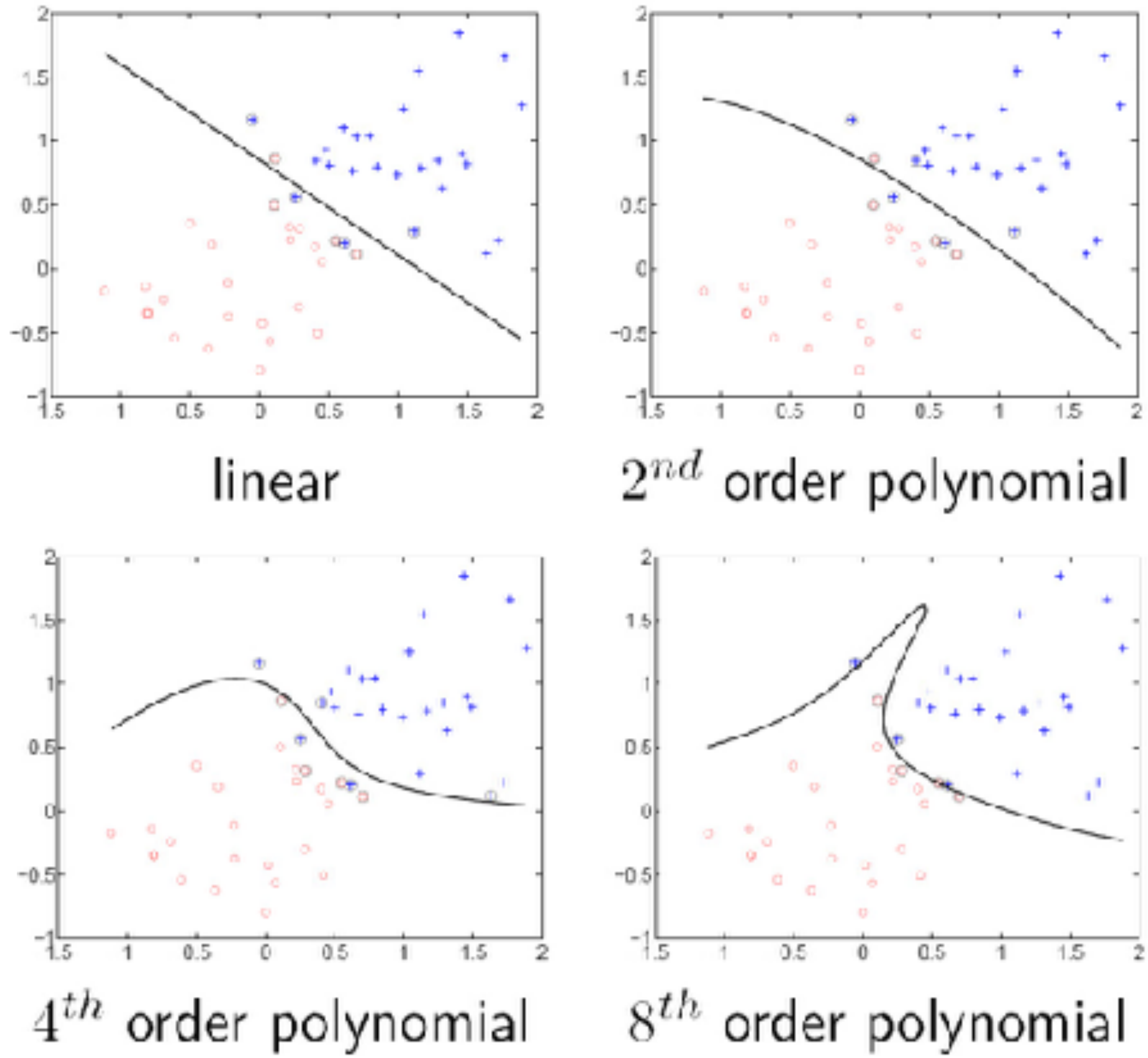
- Kernel SVM decision function:

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

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

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

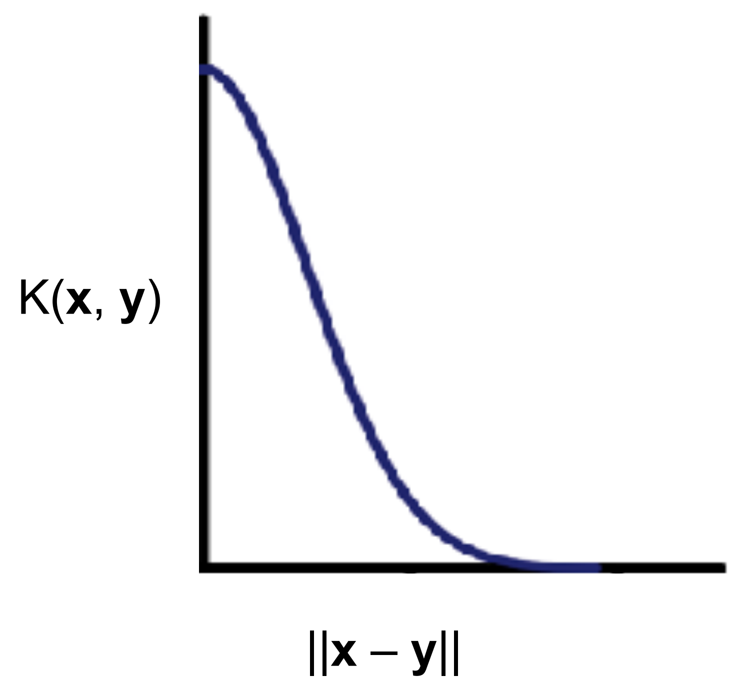
Polynomial kernel: $K(\mathbf{x},\mathbf{y}) = (c + \mathbf{x}\cdot\mathbf{y})^d$



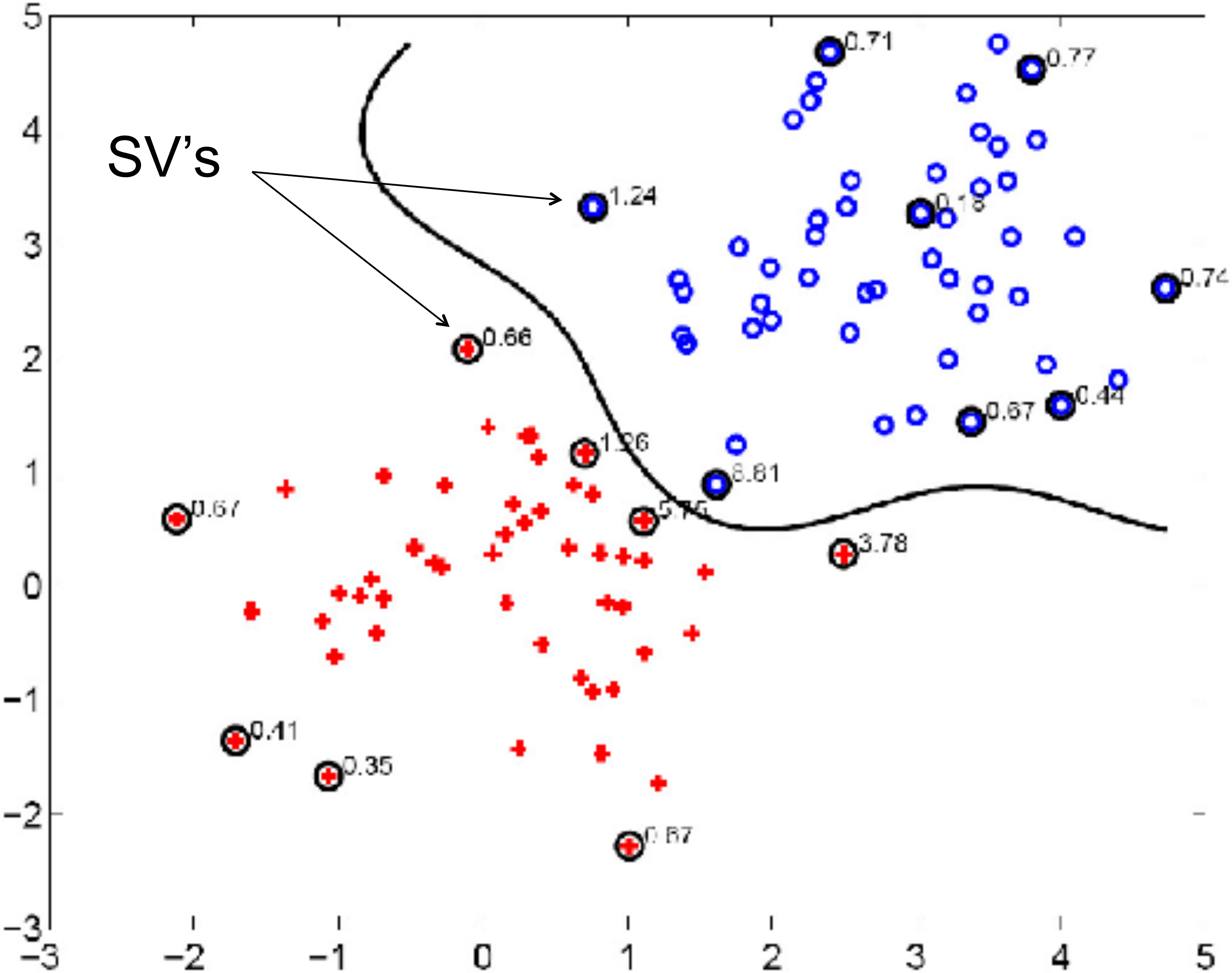
Gaussian kernel

- Also known as the radial basis function (RBF) kernel:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{\sigma^2} \|\mathbf{x} - \mathbf{y}\|^2 \right)$$



Gaussian kernel



Kernels for histograms

- Histogram intersection:

$$K(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i))$$

- Square root (Bhattacharyya kernel):

$$K(h_1, h_2) = \sum_{i=1}^{N} \sqrt{h_1(i)\, h_2(i)}$$

SVMs: Pros and cons

- Pros
  - Kernel-based framework is very powerful, flexible
  - Training is convex optimization, globally optimal solution can be found
  - Amenable to theoretical analysis
  - SVMs work very well in practice, even with very small training sample sizes
- Cons
  - No “direct” multi-class SVM, must combine two-class SVMs (e.g., with one-vs-others)
  - Computation, memory (esp. for nonlinear SVMs)

Generalization

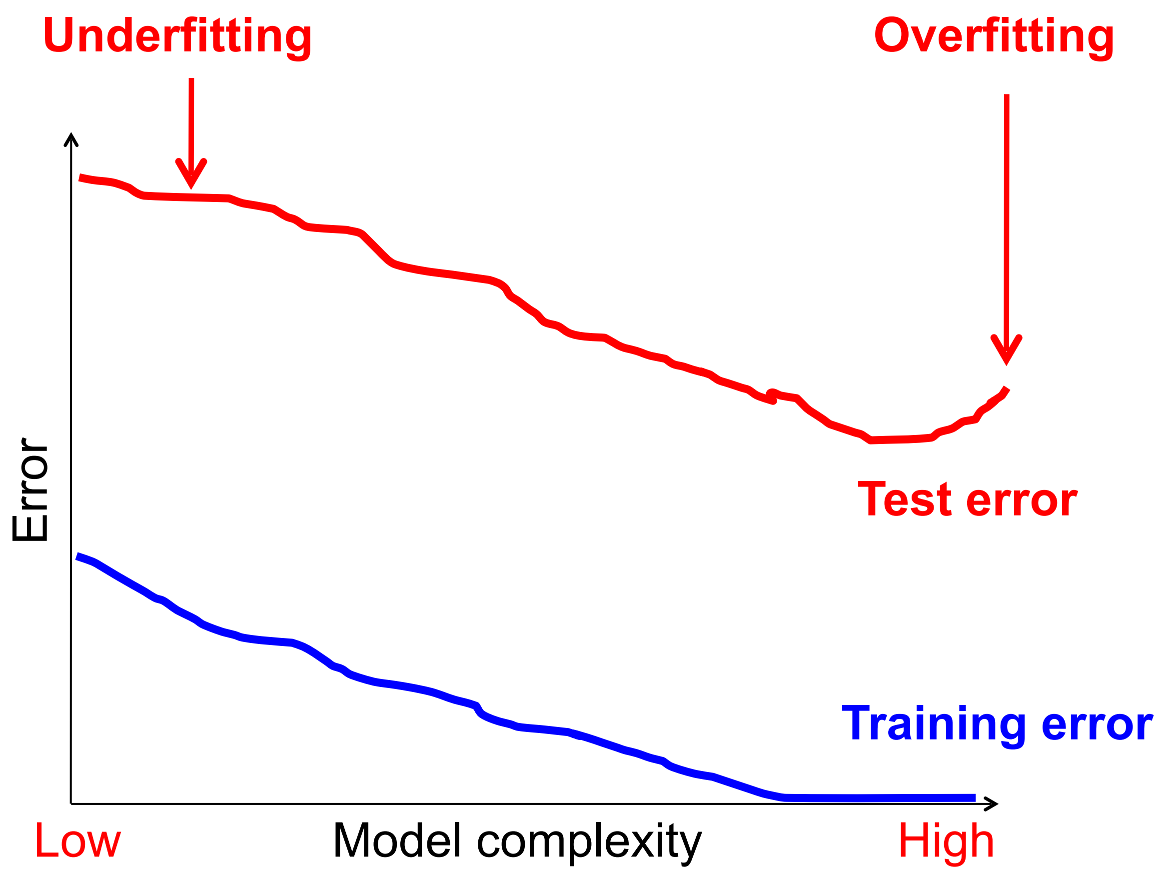
- Generalization refers to the ability to correctly classify never before seen examples
- Can be controlled by turning “knobs” that affect the complexity of the model

Training set (labels known)

Test set (labels unknown)

Diagnosing generalization ability

- **Training error:** how does the model perform on the data on which it was trained?
- **Test error:** how does it perform on never before seen data?



Source: D. Hoiem

Underfitting and overfitting

- **Underfitting:** training and test error are both *high*
  - Model does an equally poor job on the training and the test set
  - Either the training procedure is ineffective or the model is too "simple" to represent the data
- **Overfitting:** Training error is *low* but test error is *high*
  - Model fits irrelevant characteristics (noise) in the training data
  - Model is too complex or amount of training data is insufficient

Underfitting | Good generalization | Overfitting

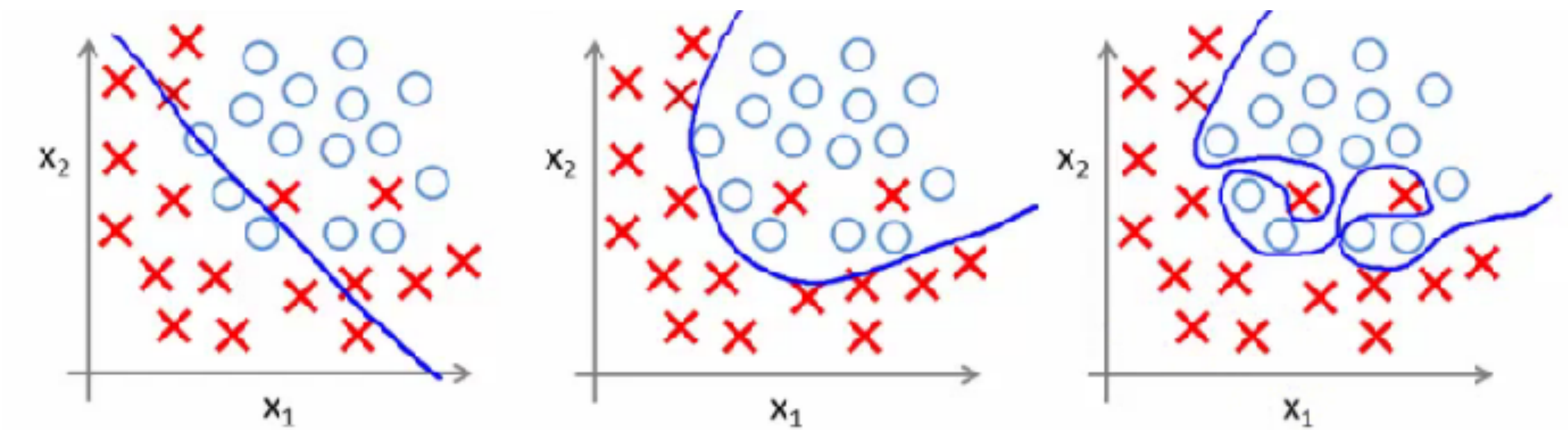
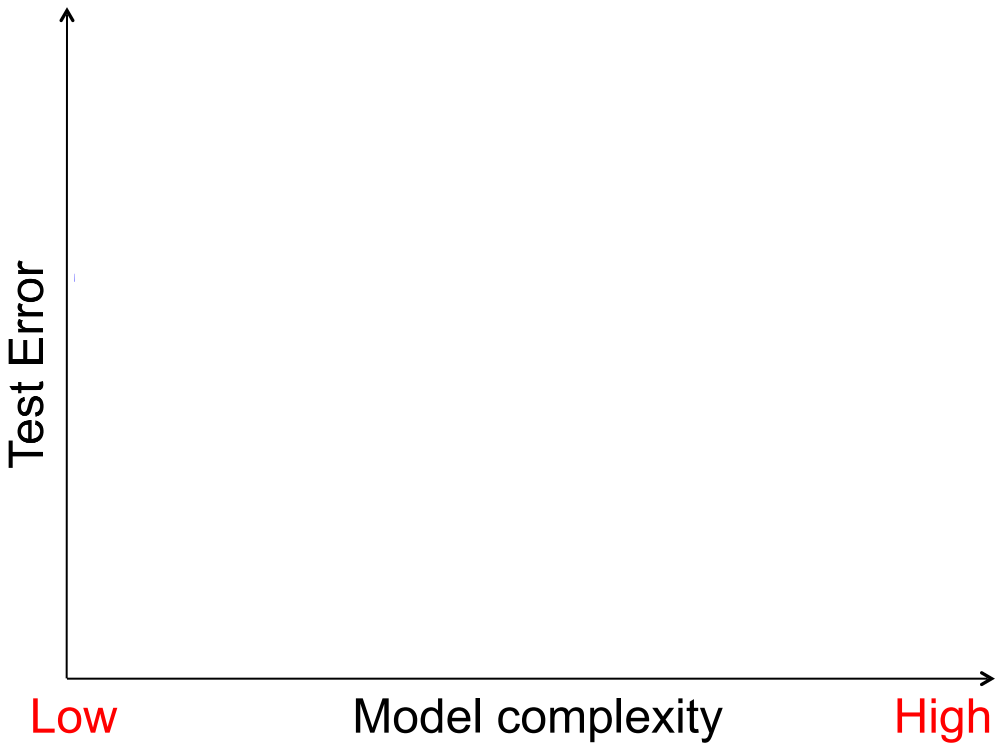


Figure source

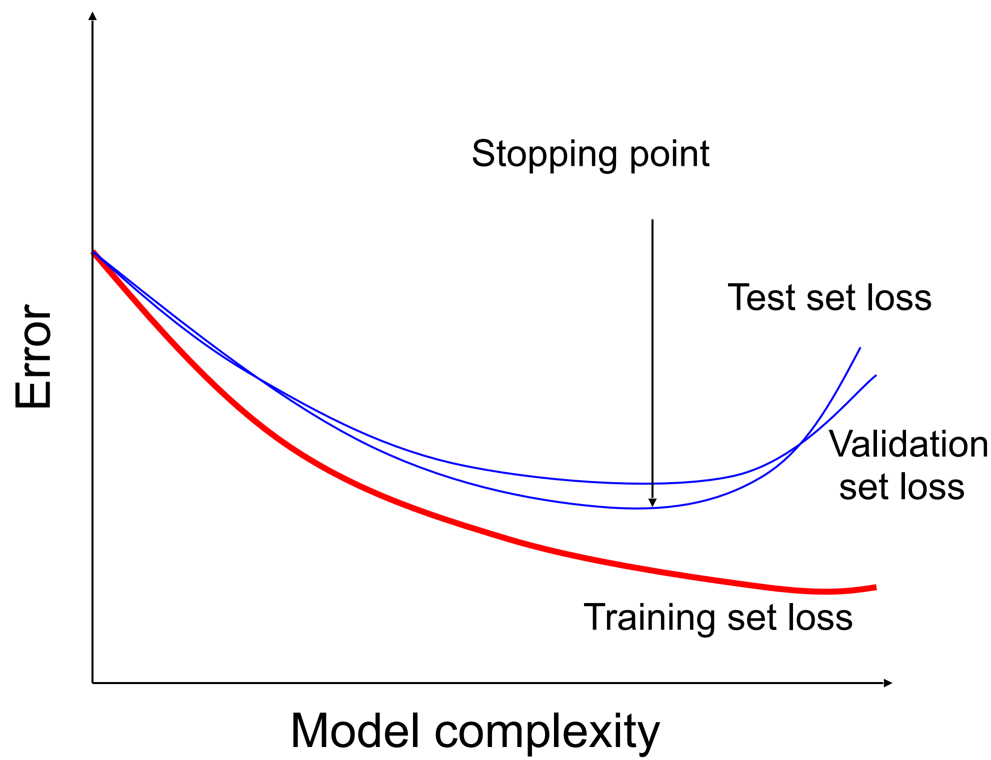
Effect of training set size



Source: D. Hoiem

Validation

- Split the data into **training**, **validation**, and **test** subsets
- Use training set to **optimize model parameters**
- Use validation test to **choose the best model**
- Use test set only to **evaluate performance**



Summary

The different steps