上海交通大学

计算机视觉

教师：赵旭

班级：AI4701

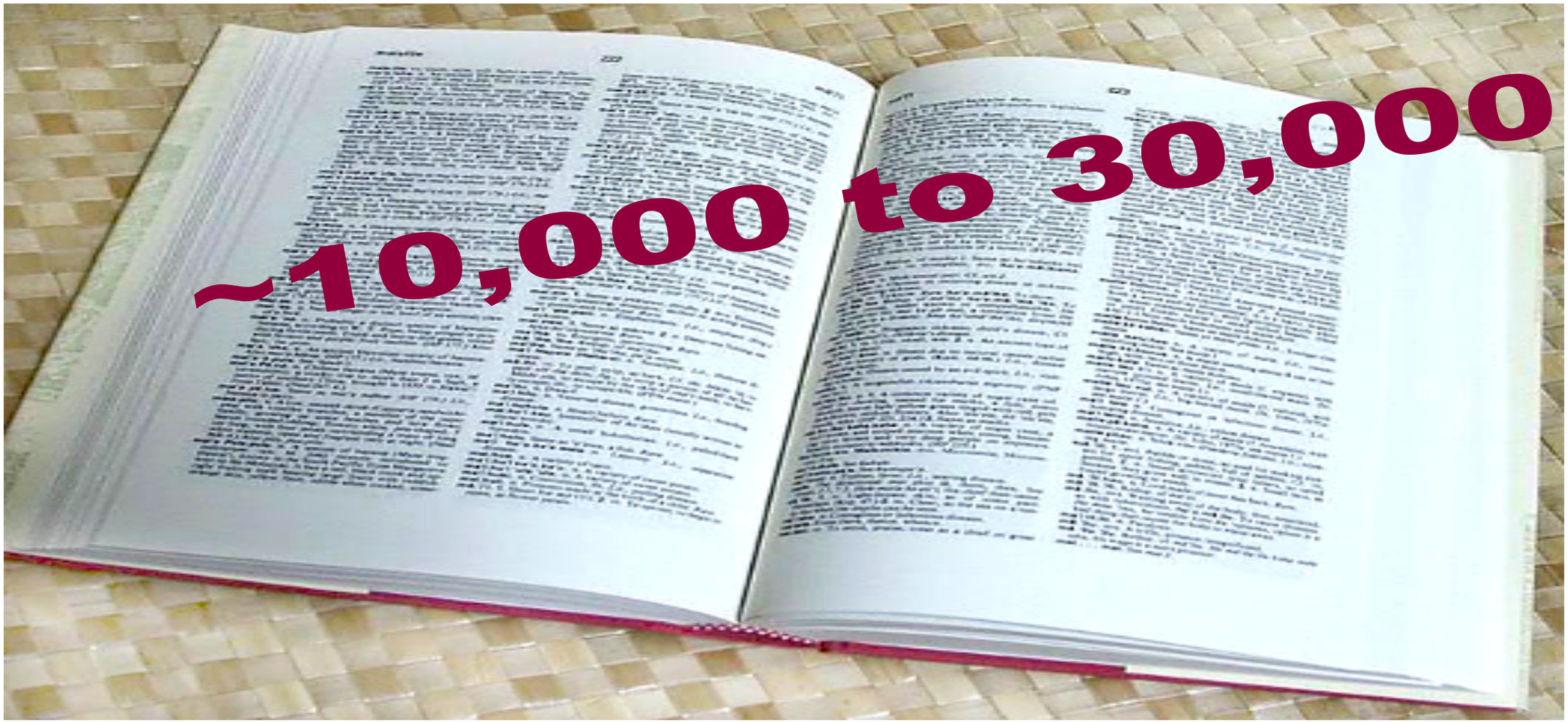
2024 春

16. 分类与识别

Contents

- Recognition: Overview and History
- Machine Learning for Recognition

How many visual object categories are there?



1500-3000 basic-level nouns, ~10 types per basic-level category

Alternative explanation (Perona): ~1000 names per domain (broad scene category), 20-30 domains

~10,000 to 30,000

OBJECTS

ANIMALS

PLANTS

INANIMATE

…..

VERTEBRATE

NATURAL

MAN-MADE

MAMMALS

BIRDS

TAPIR

BOAR

GROUSE

CAMERA

Recognition tasks

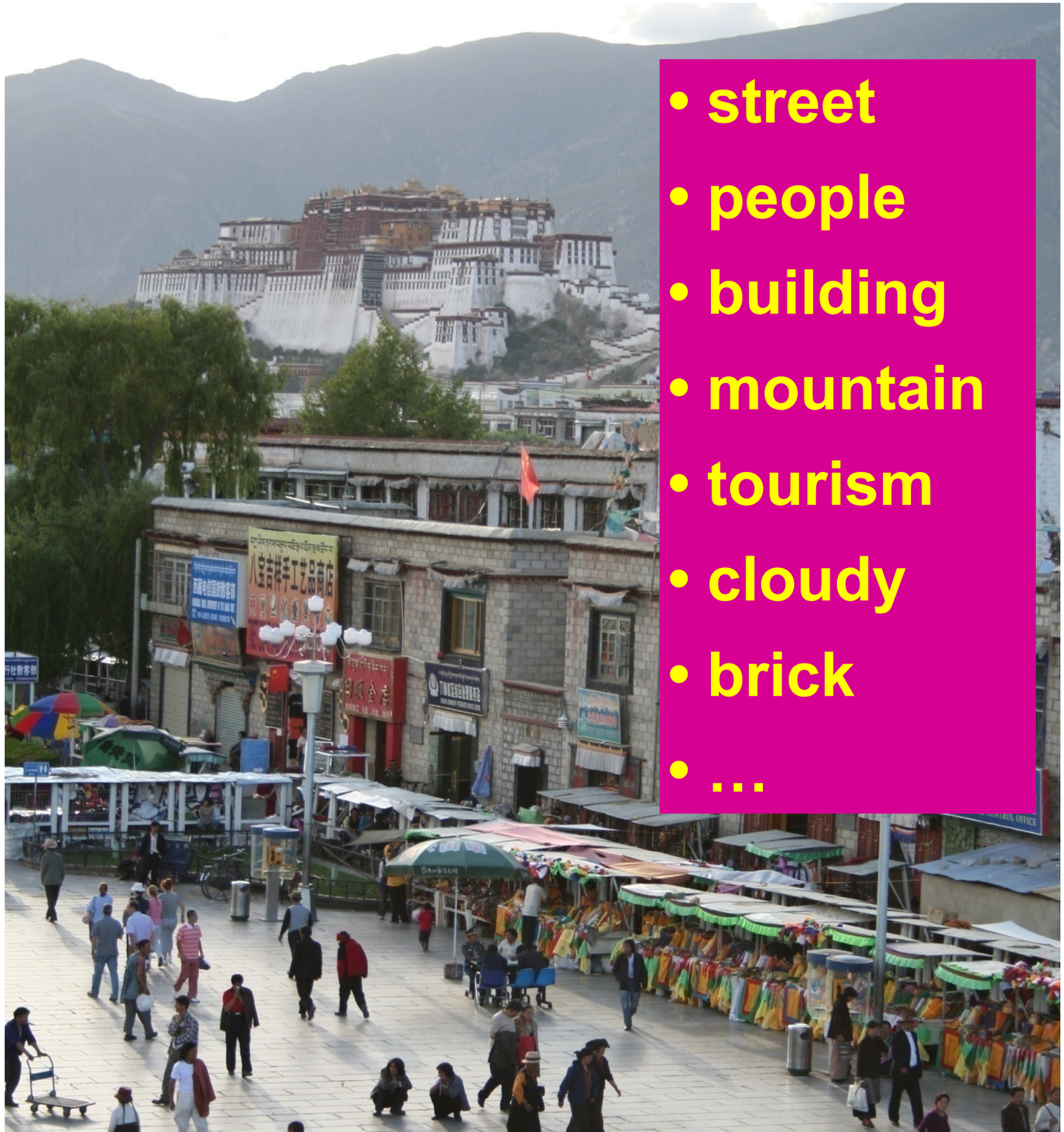
Svetlana Lazebnik

Scene categorization or classification

- outdoor/indoor
- city/forest/factory/etc.

Svetlana Lazebnik

Image annotation / tagging / attributes

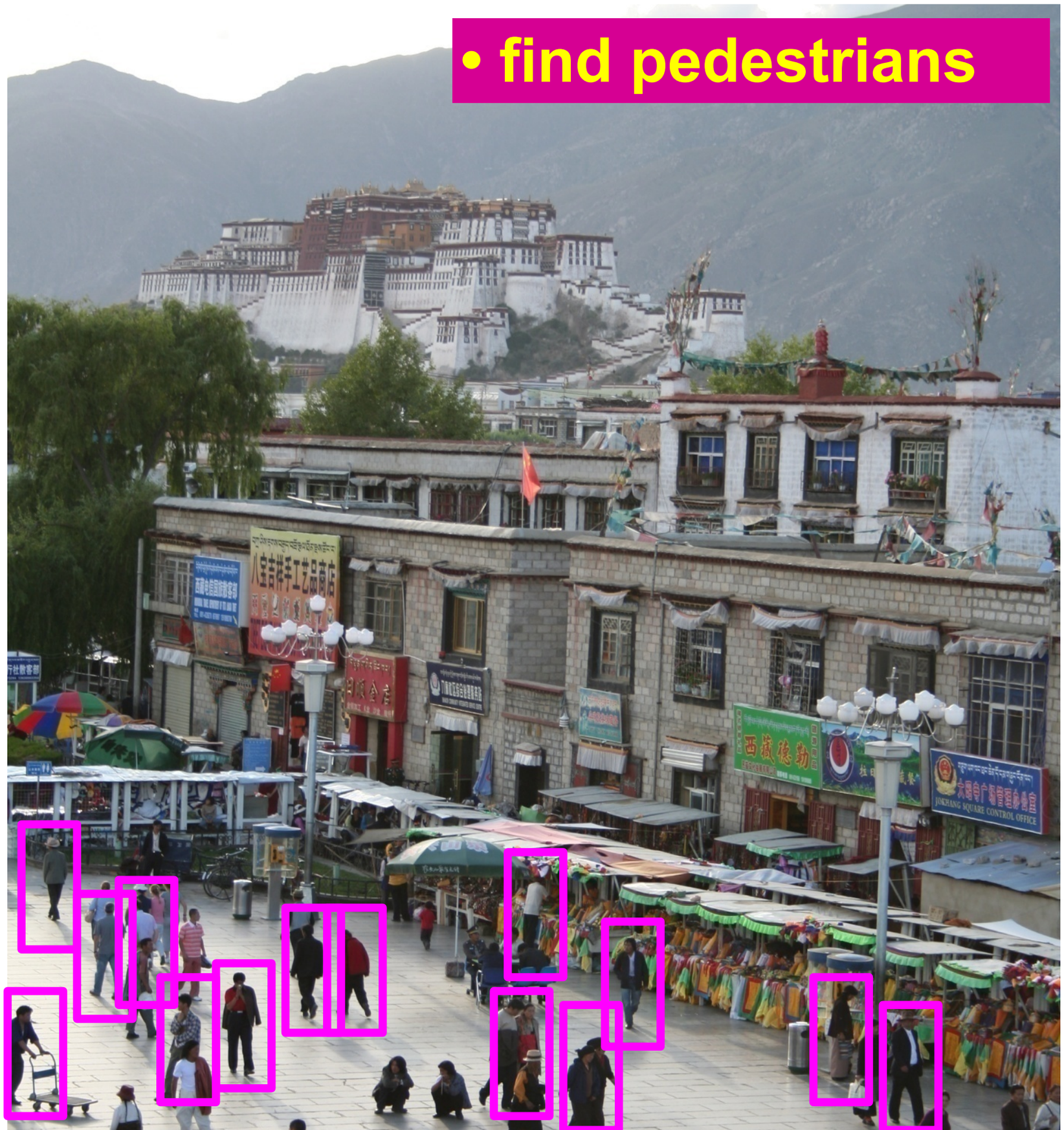


Svetlana Lazebnik

Image parsing / semantic segmentation

Svetlana Lazebnik

Object detection



Svetlana Lazebnik

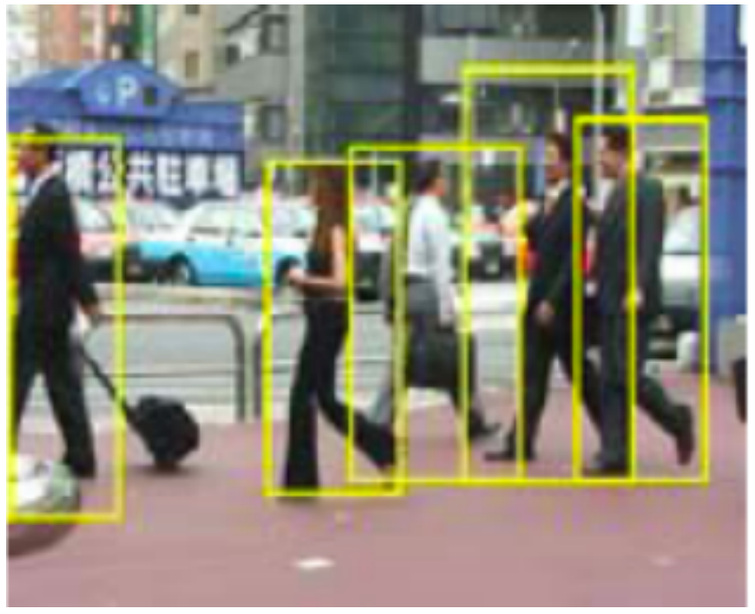
Scene understanding?

Svetlana Lazebnik

Category vs. instance recognition

Category:

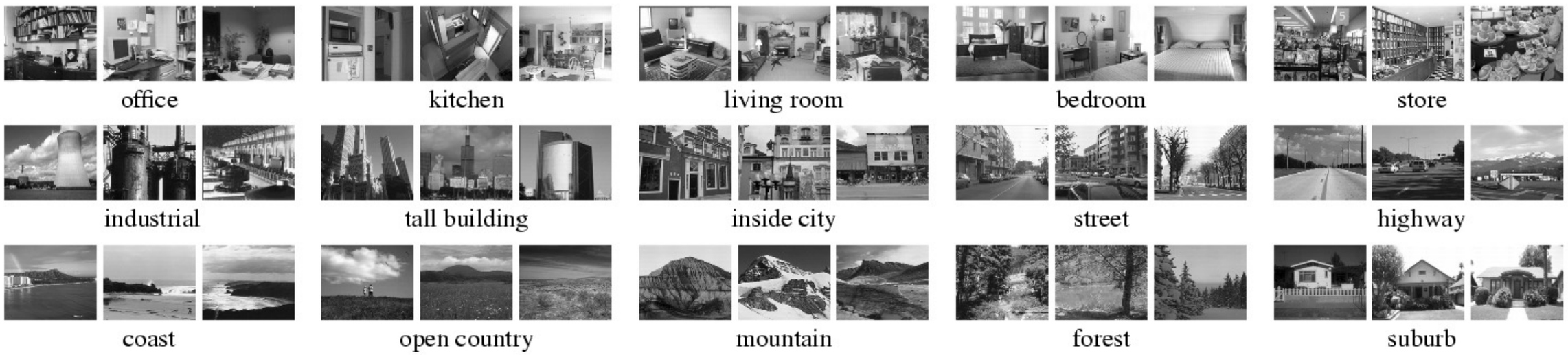
- Find all the people
- Find all the buildings
- Often within a single image
- Often ‘sliding window’



Instance:

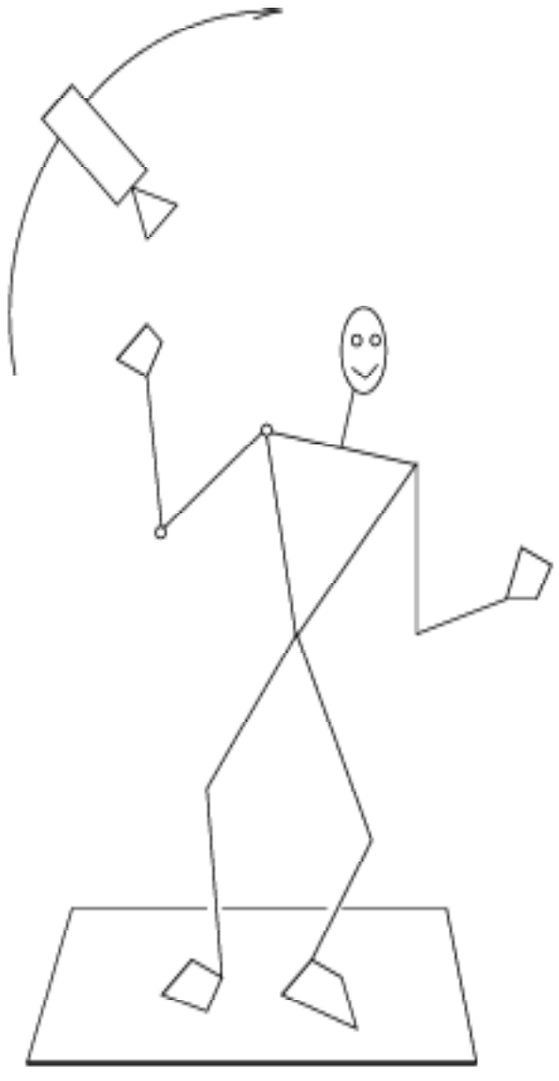
- Is this face Trump?
- Find this specific famous building
- Often within a database of images

Scene recognition dataset



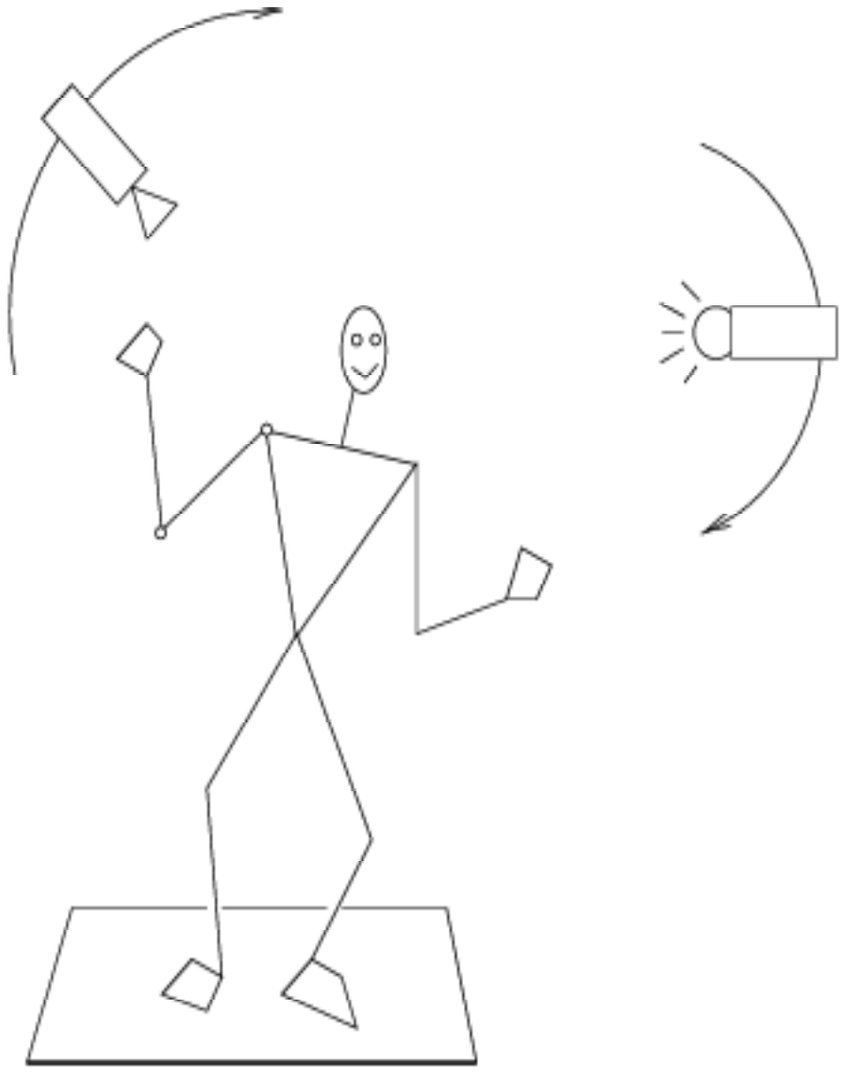
Instance or category?

Recognition is all about modeling variability

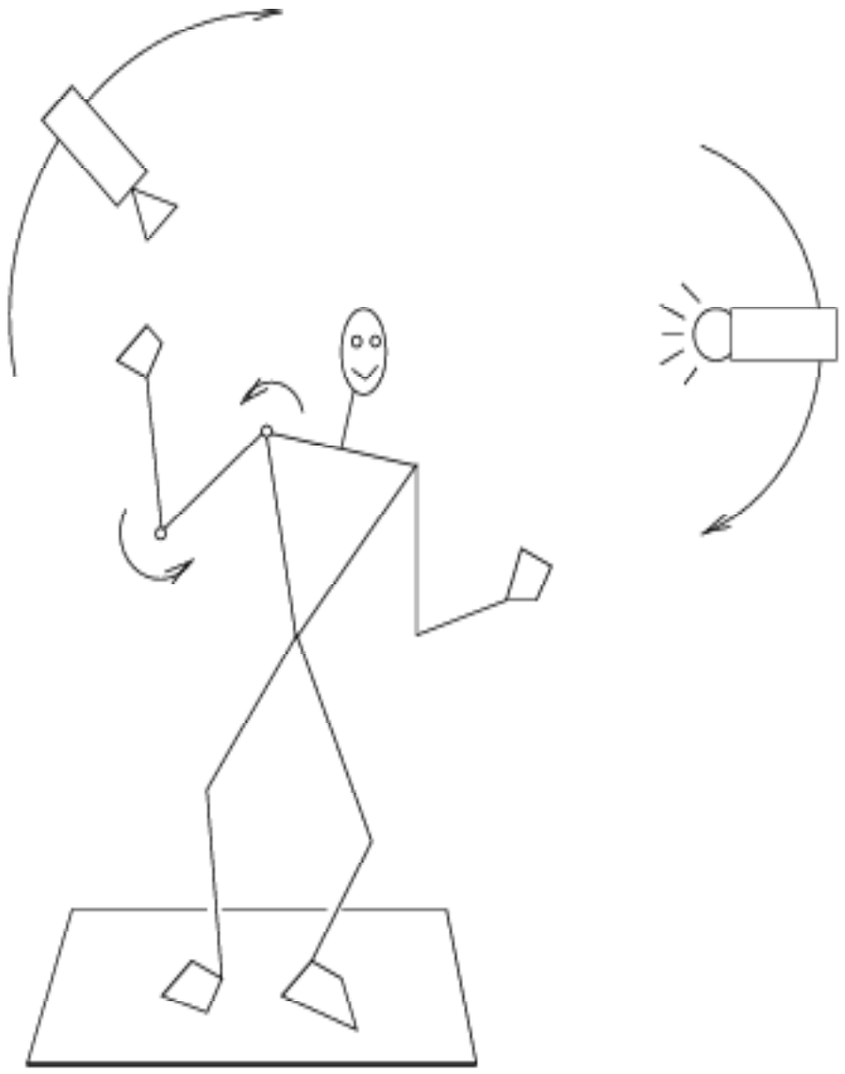


Variability: Camera position

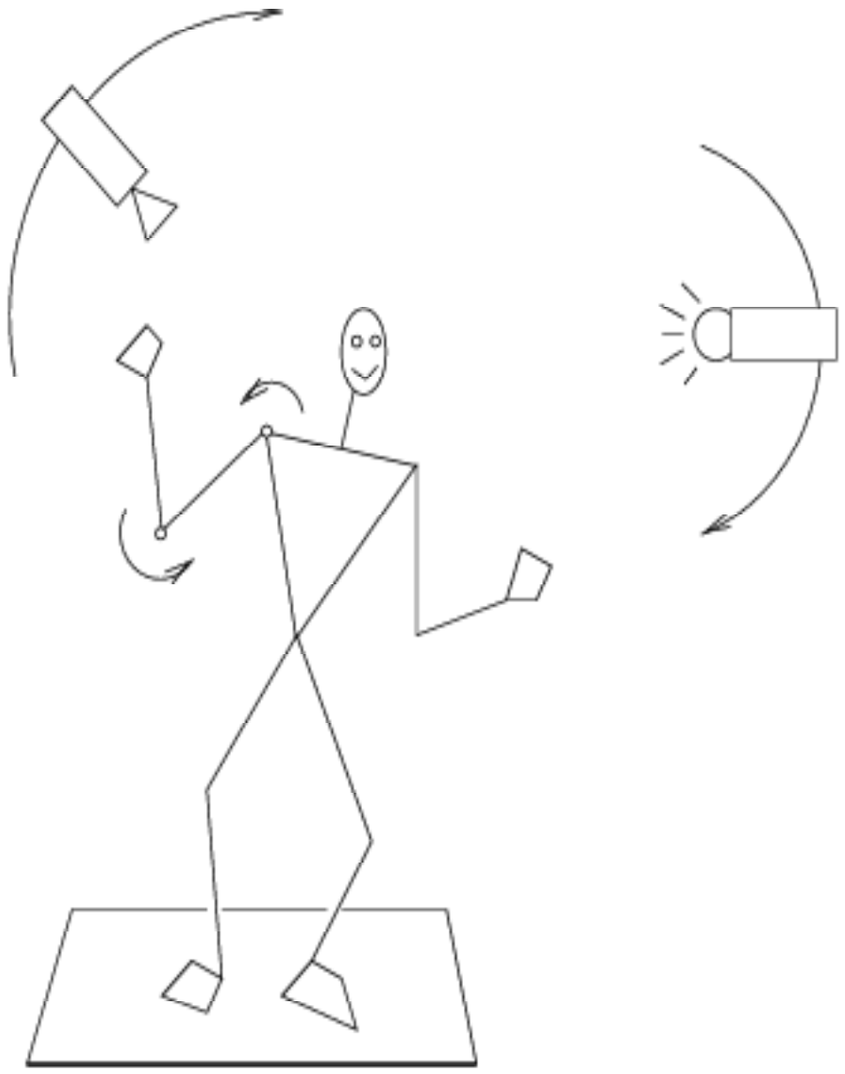
Recognition is all about modeling variability



Variability: Camera position
Illumination



Variability: Camera position
Illumination
Pose/shape parameters



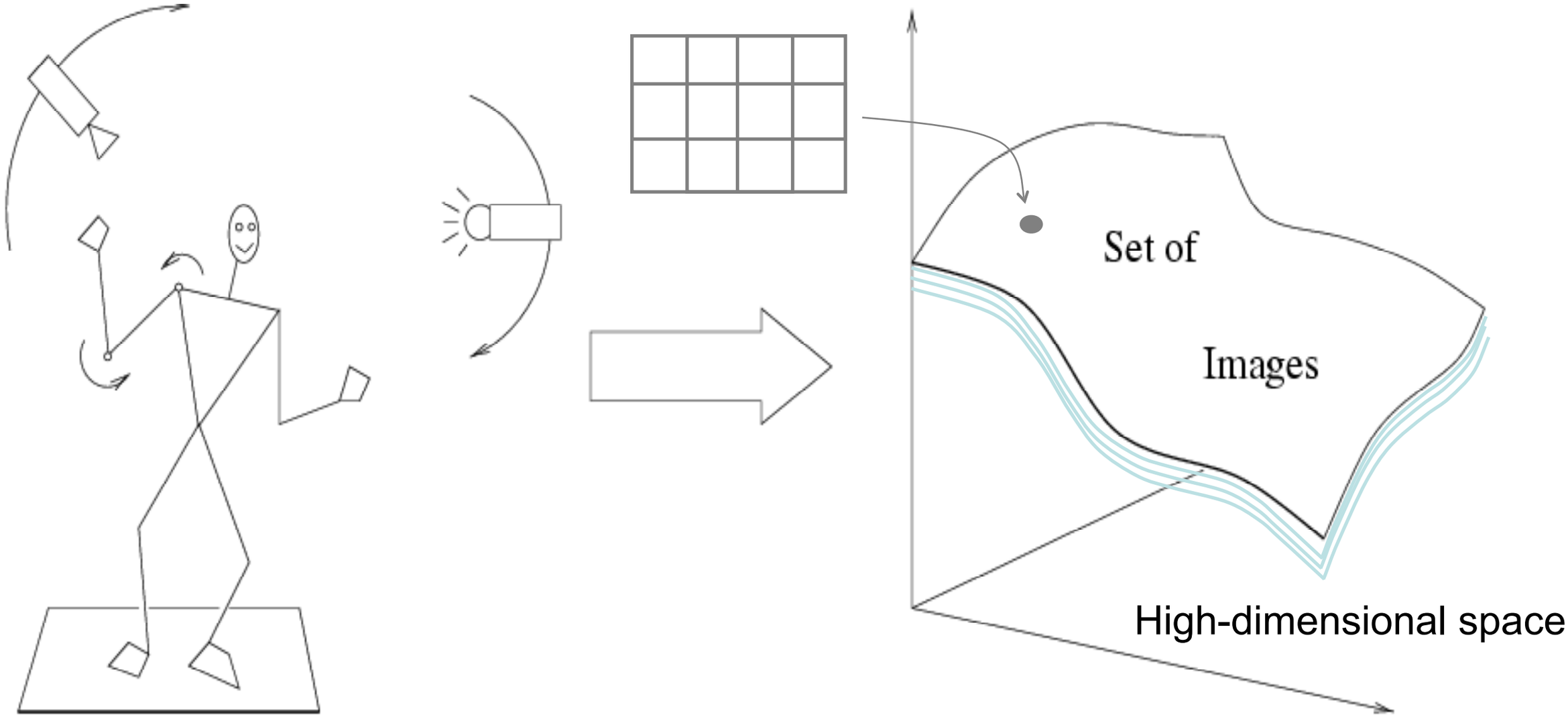
Variability: Camera position

Illumination

Pose/shape parameters

Within-class variations?

Within-class variations



Variability: Camera position

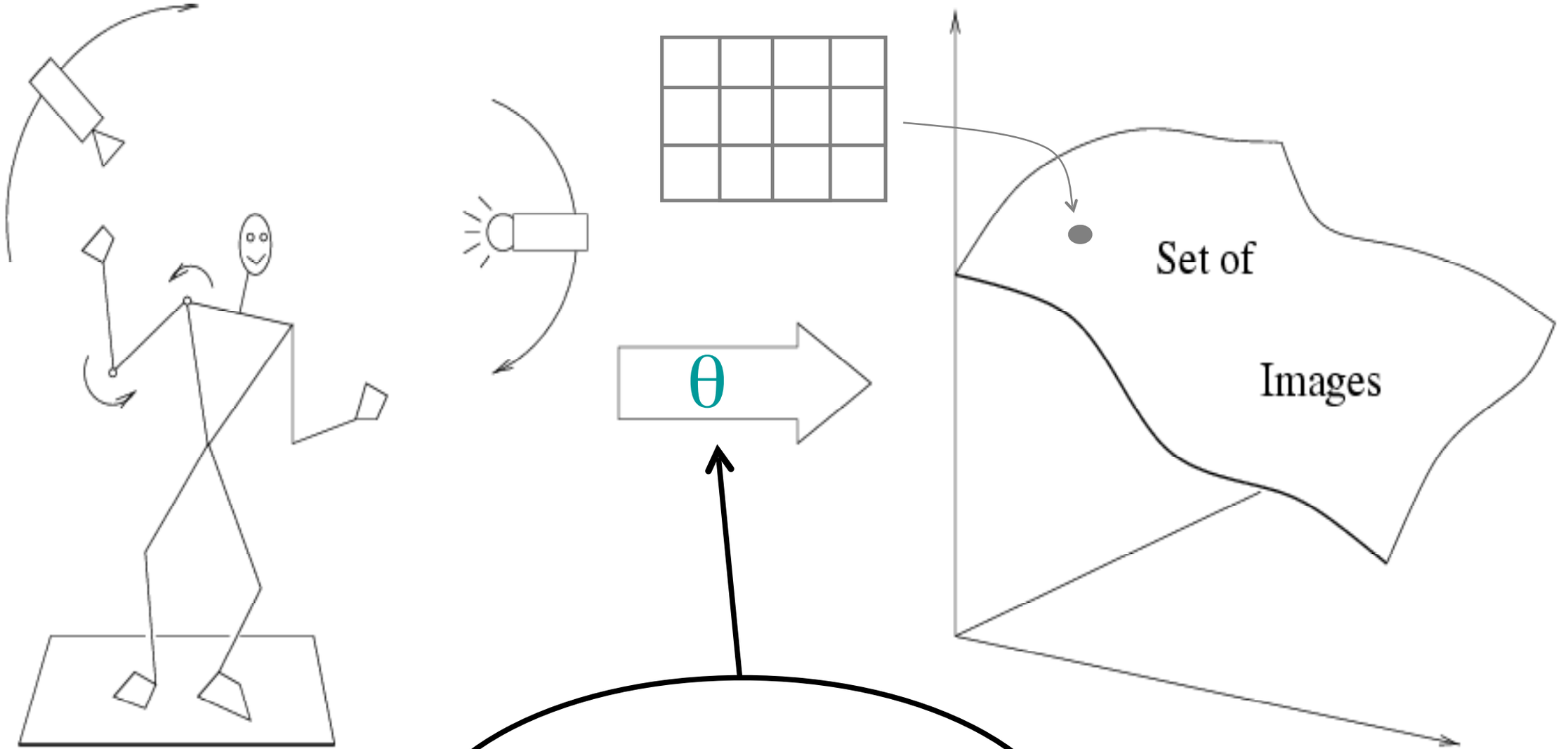
Illumination

Pose/shape parameters
Within-class variation

History of ideas in recognition

- 1960s – early 1990s: the geometric era

No digital cameras!
Slow compute!



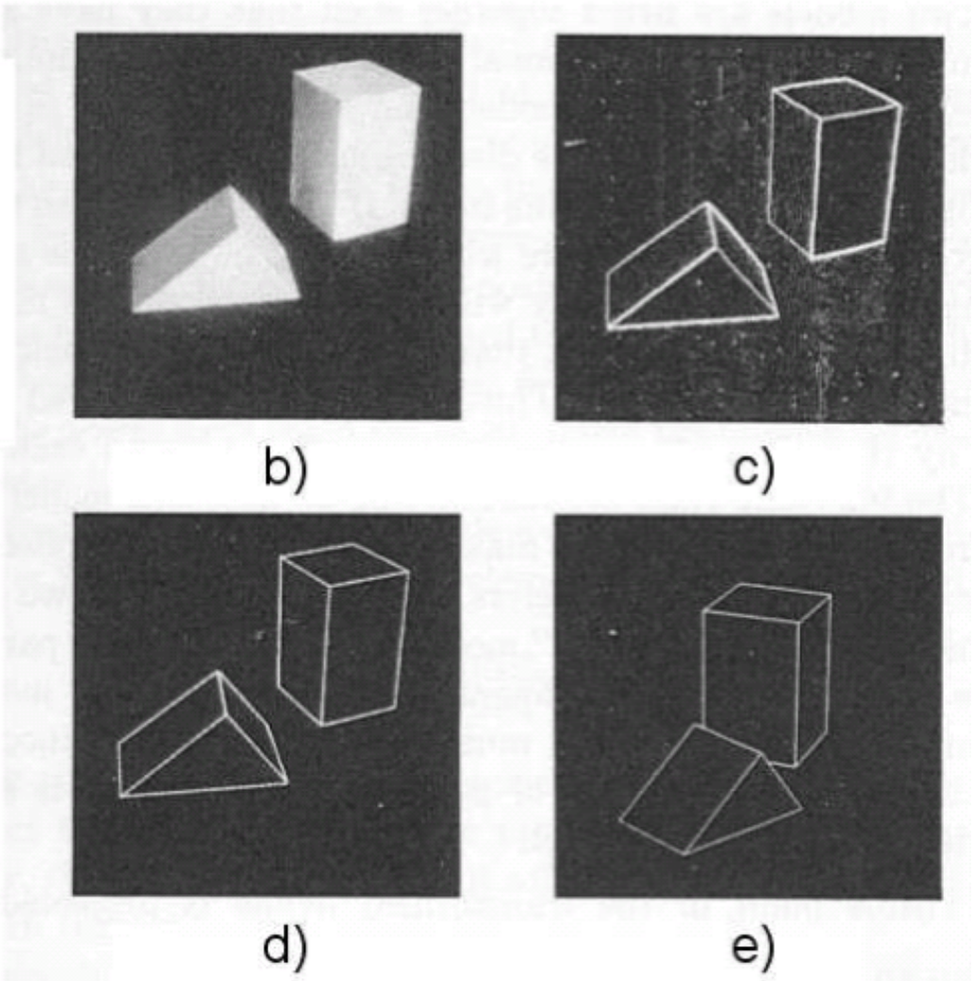
Variability:

Camera position
Illumination

Shape is known

Roberts (1965); Lowe (1987); Faugeras & Hebert (1986); Grimson & Lozano-Perez (1986); Huttenlocher & Ullman (1987)

Recognition as an alignment problem: Block world

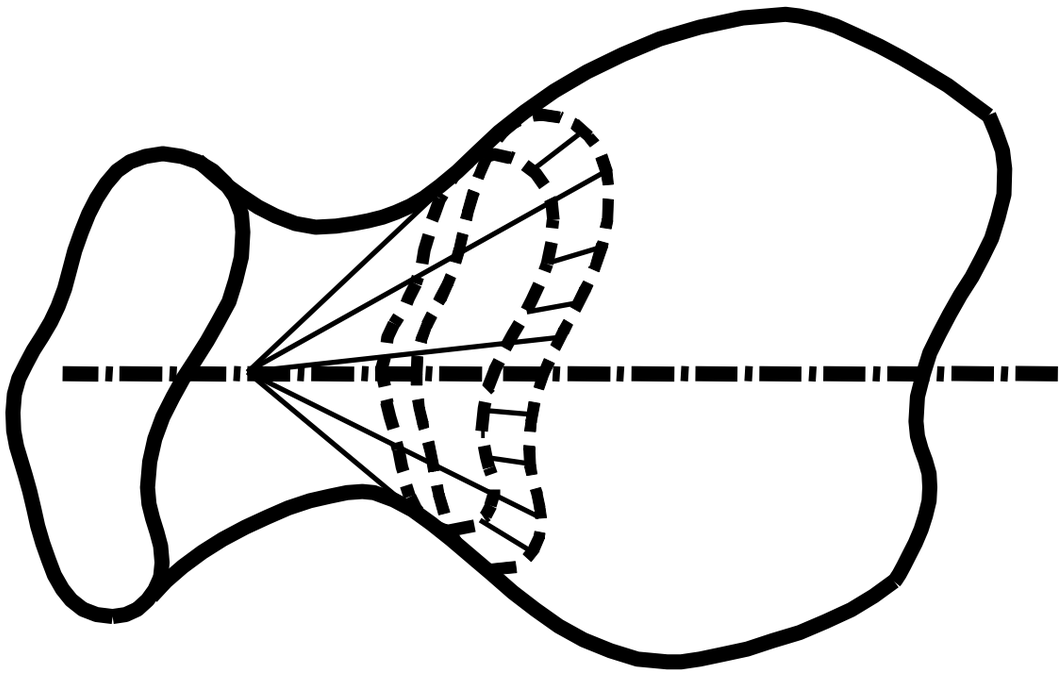


L. G. Roberts
Machine Perception of Three Dimensional Solids,
Ph.D. thesis, MIT
Department of Electrical Engineering, 1963.

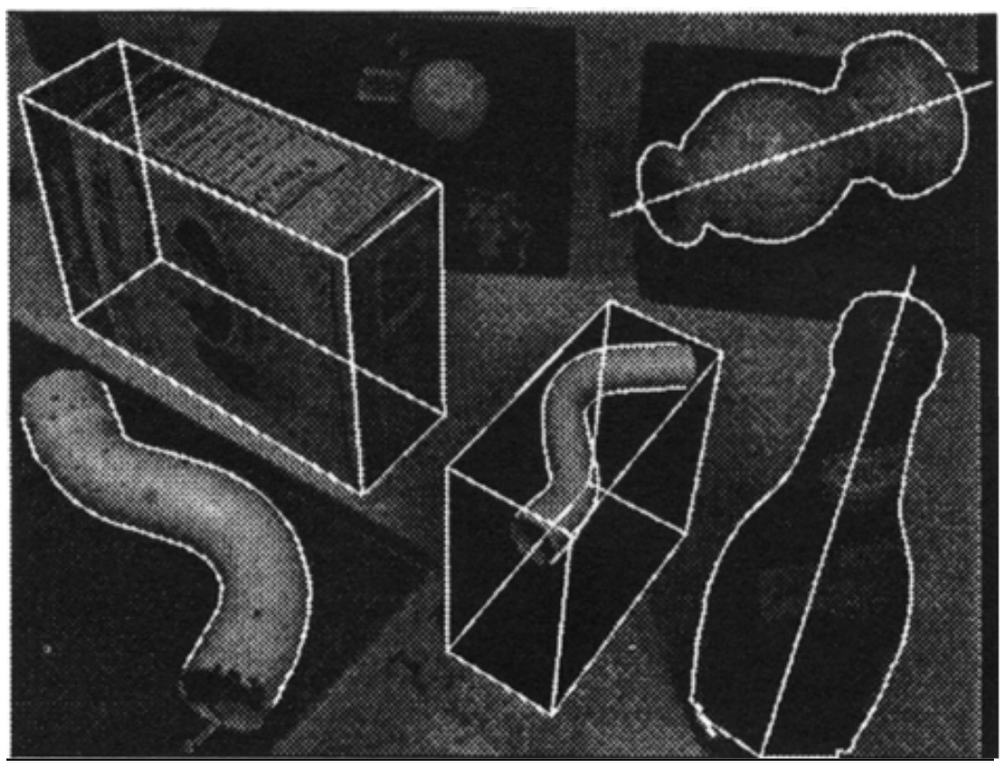
Fig. 1. A system for recognizing 3-d polyhedral scenes. a) L.G. Roberts. b)A blocks world scene. c)Detected edges using a 2x2 gradient operator. d) A 3-d polyhedral description of the scene, formed automatically from the single image. e) The 3-d scene displayed with a viewpoint different from the original image to demonstrate its accuracy and completeness. (b) - e) are taken from [64] with permission MIT Press.)

J. Mundy, Object Recognition in the Geometric Era: a Retrospective, 2006

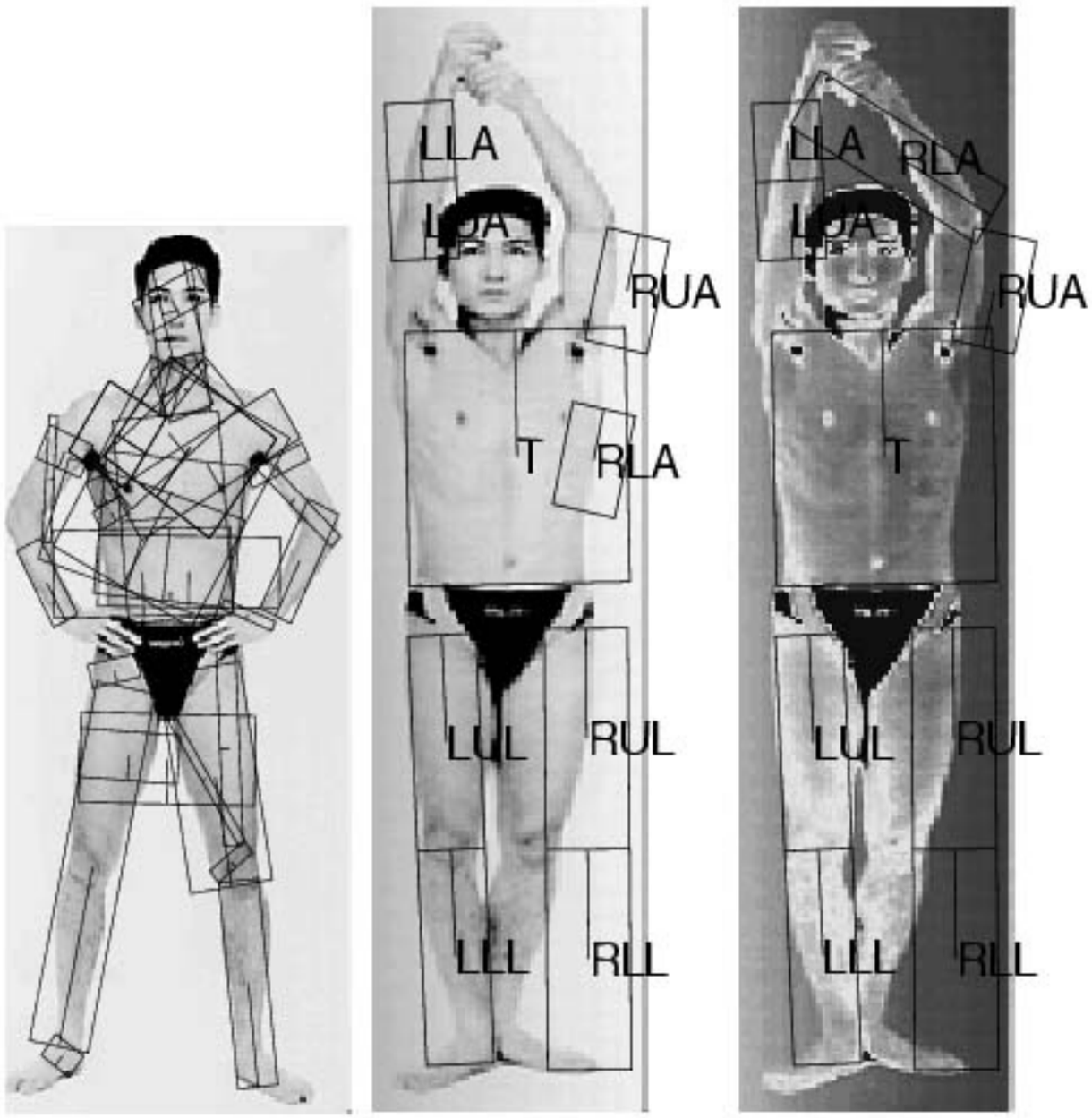
General shape primitives?



Generalized cylinders
Ponce et al. (1989)



Zisserman et al. (1995)

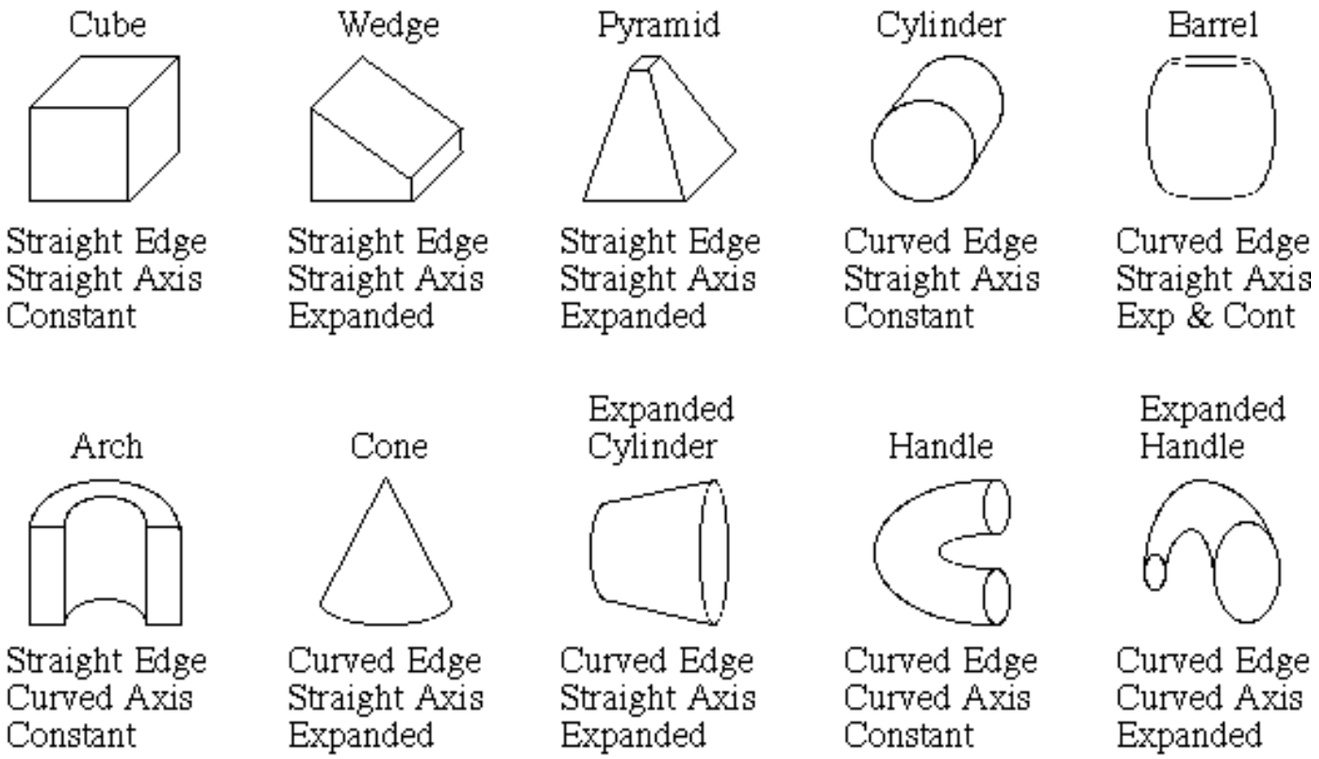


Forsyth (2000)

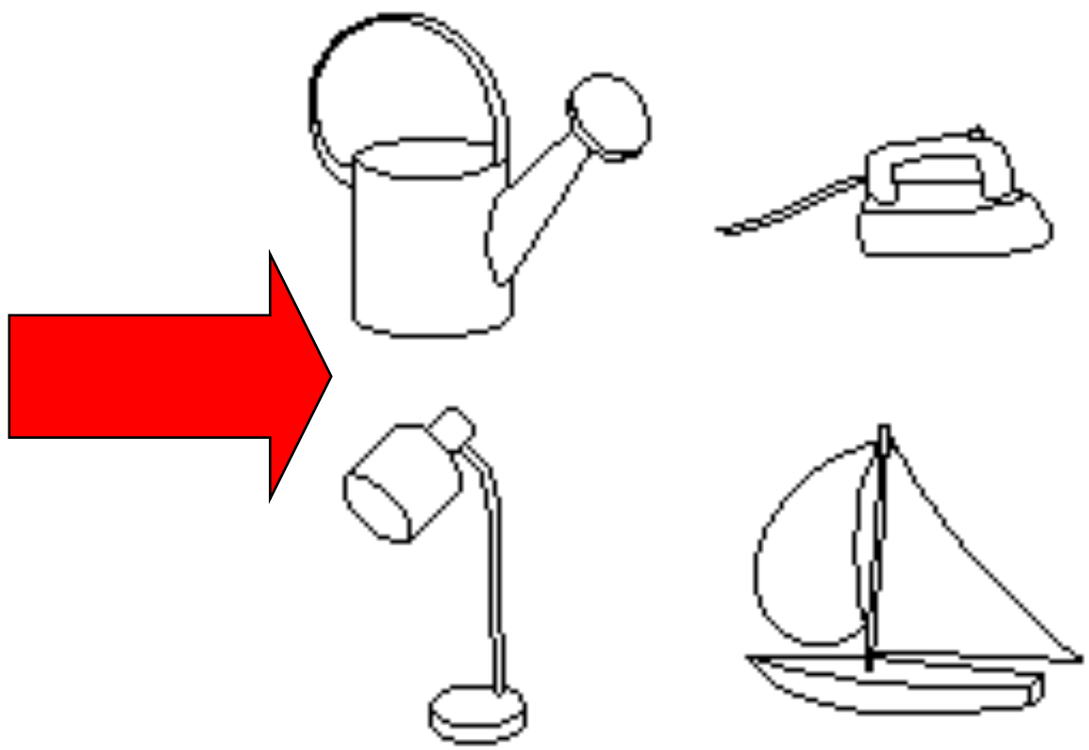
Svetlana Lazebnik

Recognition by components
Biederman (1987)

Primitives (geons)



Objects



http://en.wikipedia.org/wiki/Recognition_by_Components_Theory

Svetlana Lazebnik

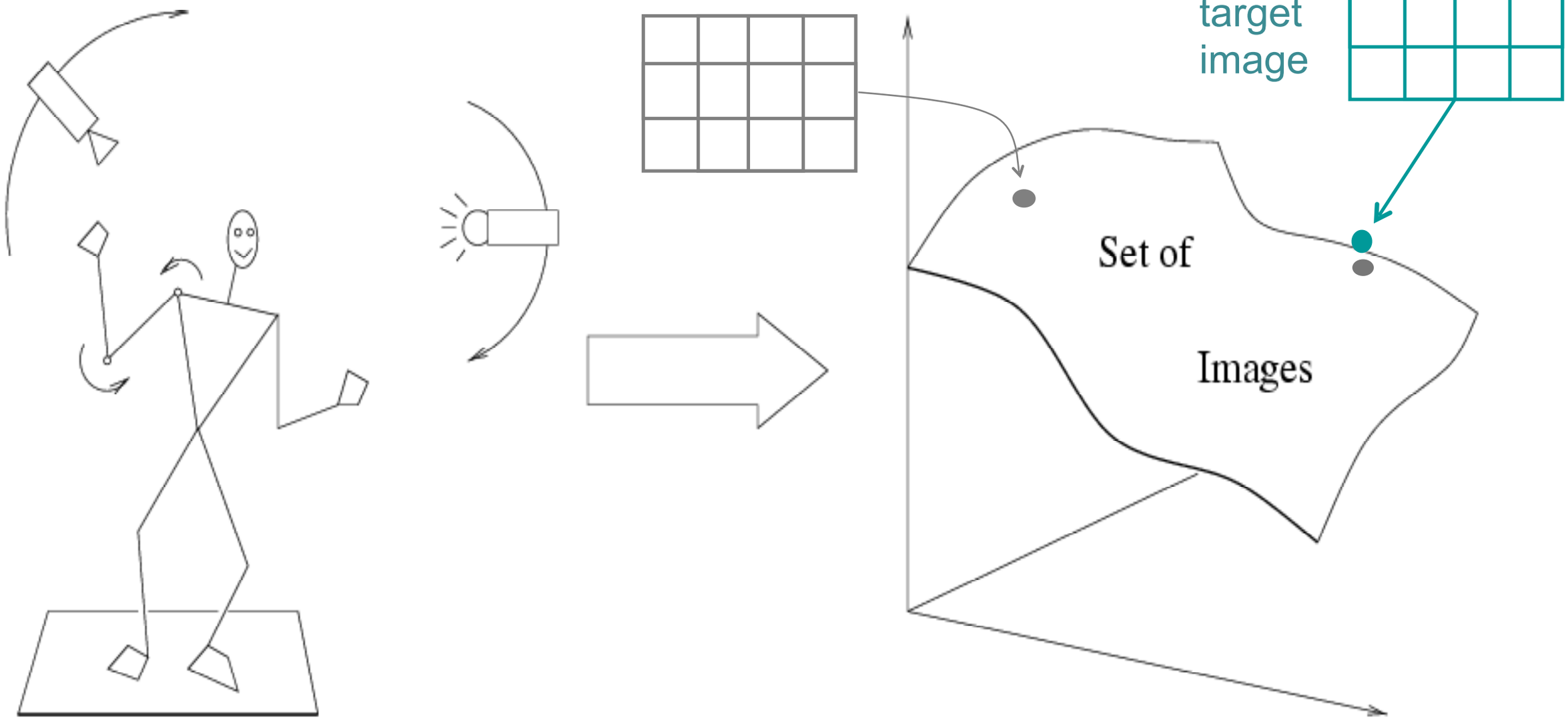
History of ideas in recognition

- 1960s – early 1990s: the geometric era

  No digital cameras!
  Slow compute!

- 1990s: appearance-based models

  Slow compute!

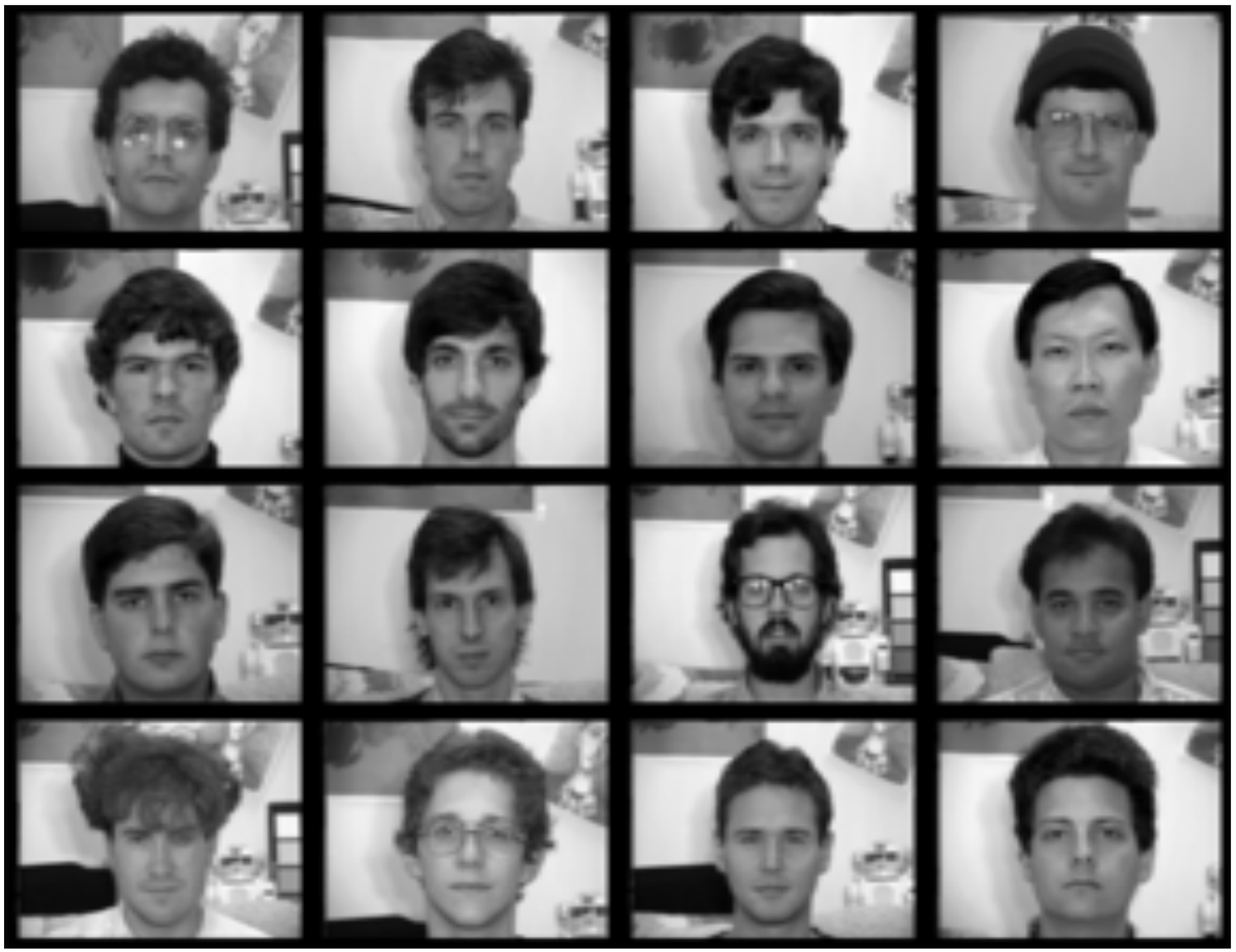


Empirical models of image variability

Appearance-based techniques

Turk & Pentland (1991); Murase & Nayar (1995); etc.

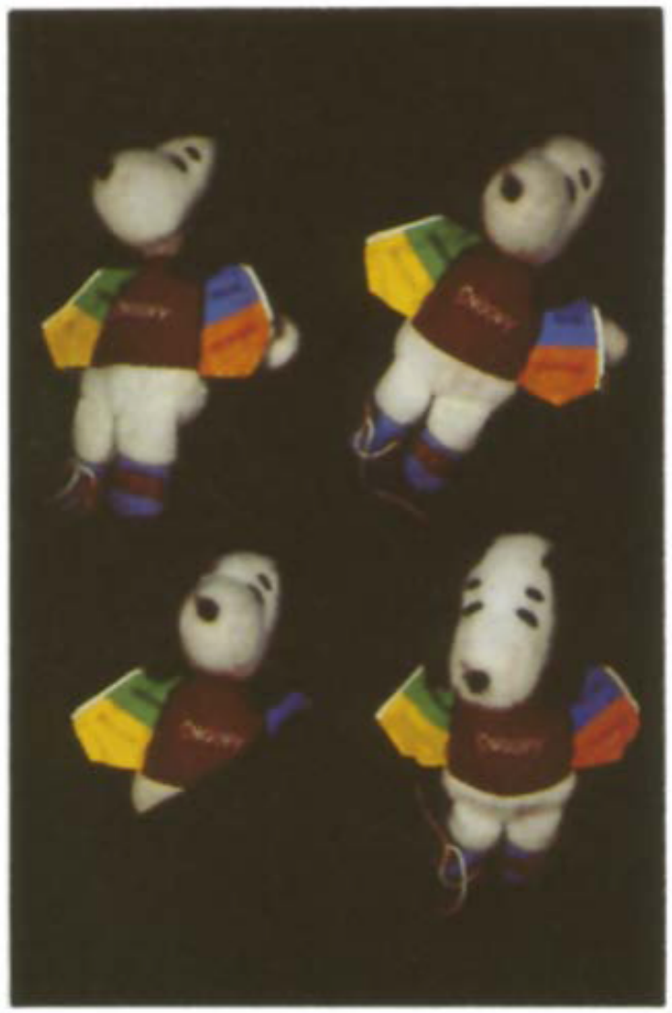
Eigenfaces (Turk & Pentland, 1991)

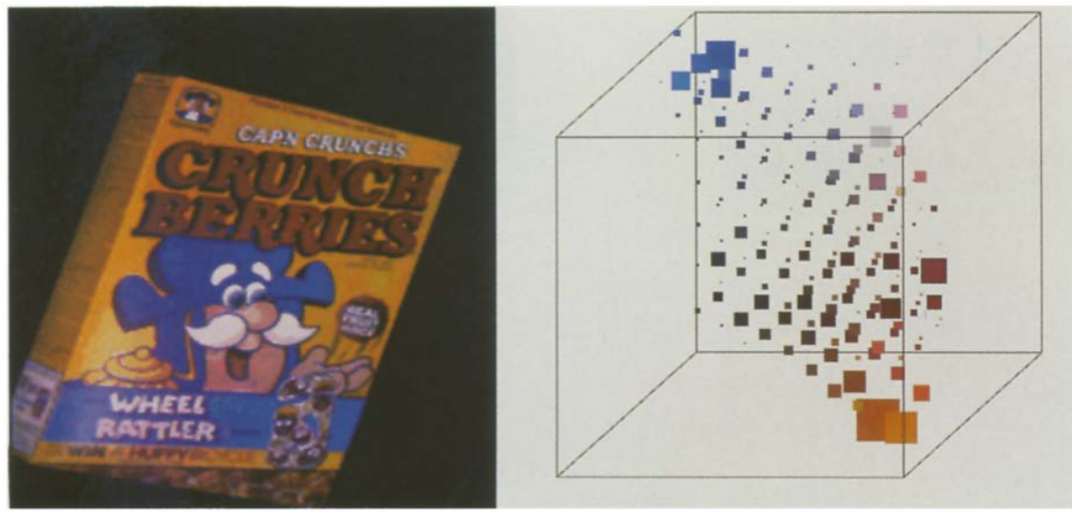


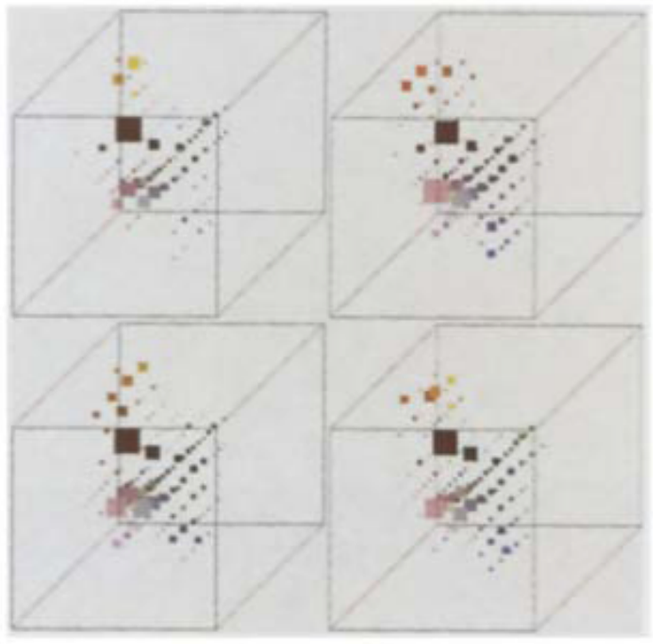
Experimental	Correct/Unknown Recognition Percentage		
Condition	Lighting	Orientation	Scale
Forced classification	96/0	85/0	64/0
Forced 100% accuracy	100/19	100/39	100/60
Forced 20% unknown rate	100/20	94/20	74/20

Svetlana Lazebnik

Color Histograms

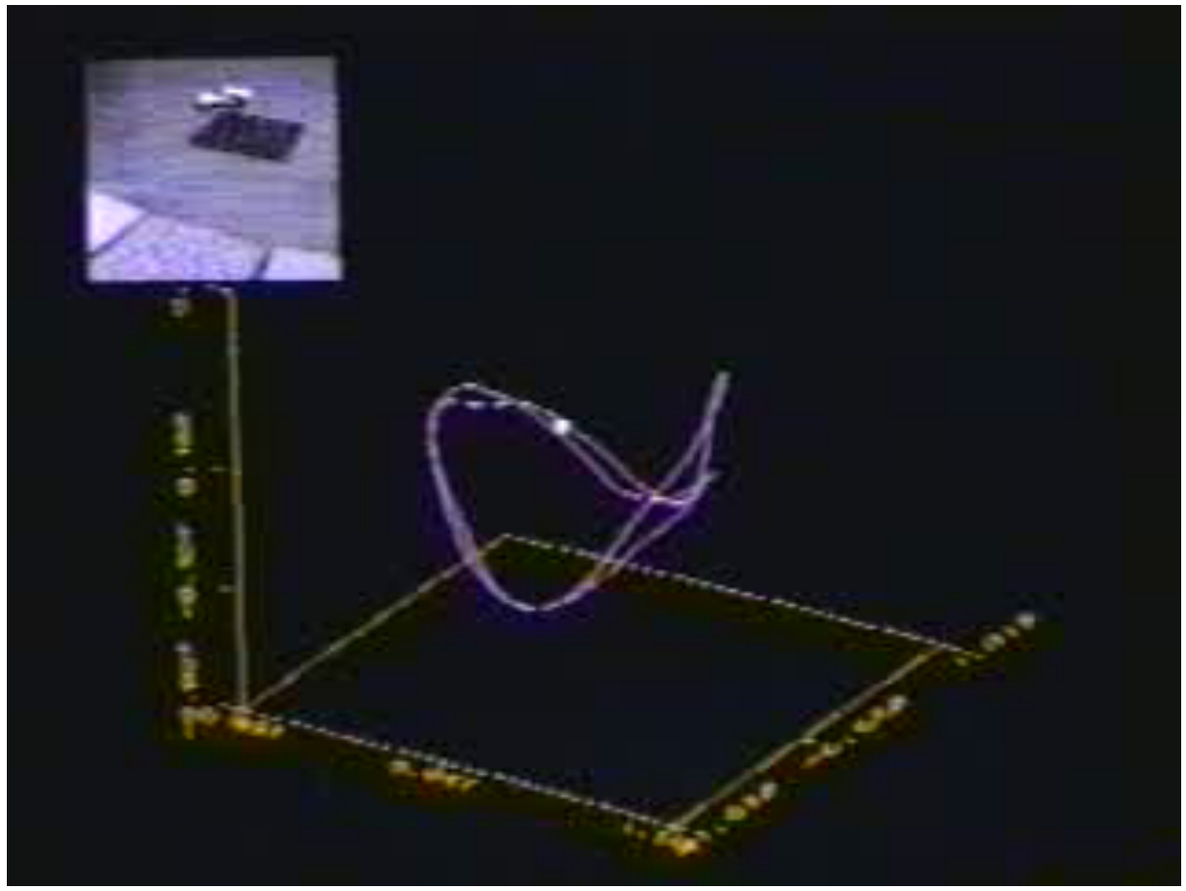






Swain and Ballard, Color Indexing, IJCV 1991.

Appearance manifolds



H. Murase and S. Nayar, Visual learning and recognition of 3-d objects from appearance, IJCV 1995

Limitations of global appearance models

- Requires global registration of patterns
- Not robust to clutter, occlusion, geometric transformations

Svetlana Lazebnik

History of ideas in recognition

- 1960s – early 1990s: the geometric era

  No digital cameras!
  Slow compute!

- 1990s: appearance-based models

  Slow compute!

- 1990s – present: sliding window approaches

Sliding window approaches

Sliding window approaches

- Turk and Pentland, 1991
- Belhumeur, Hespanha, & Kriegman, 1997
- Schneiderman & Kanade 2004
- Viola and Jones, 2000

- Schneiderman & Kanade, 2004
- Argawal and Roth, 2002
- Poggio et al. 1993

History of ideas in recognition

- 1960s – early 1990s: the geometric era

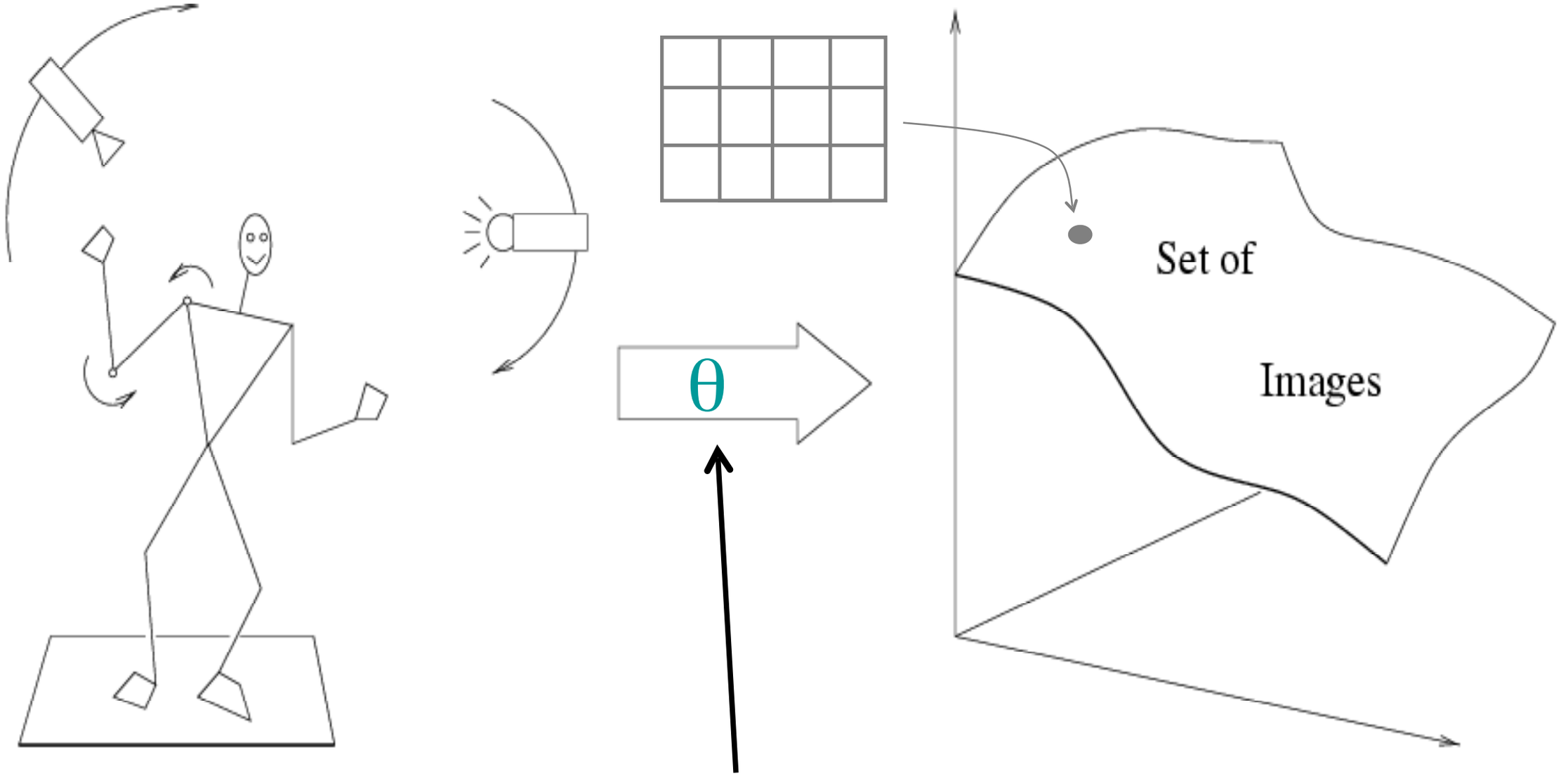
No digital cameras!
Slow compute!

- 1990s: appearance-based models

Slow compute!

- Mid-1990s: sliding window approaches
- Late 1990s: local features

Svetlana Lazebnik



Variability:

Camera position
Illumination

Shape is partially known

Roberts (1965); Lowe (1987); Faugeras & Hebert (1986); Grimson & Lozano-Perez (1986); Huttenlocher & Ullman (1987)

Local features for object instance recognition

D. Lowe (1999, 2004)

Large-scale image search
Combining local features, indexing, and spatial constraints

Philbin et al. ‘07

Large-scale image search
Combining local features, indexing, and spatial constraints

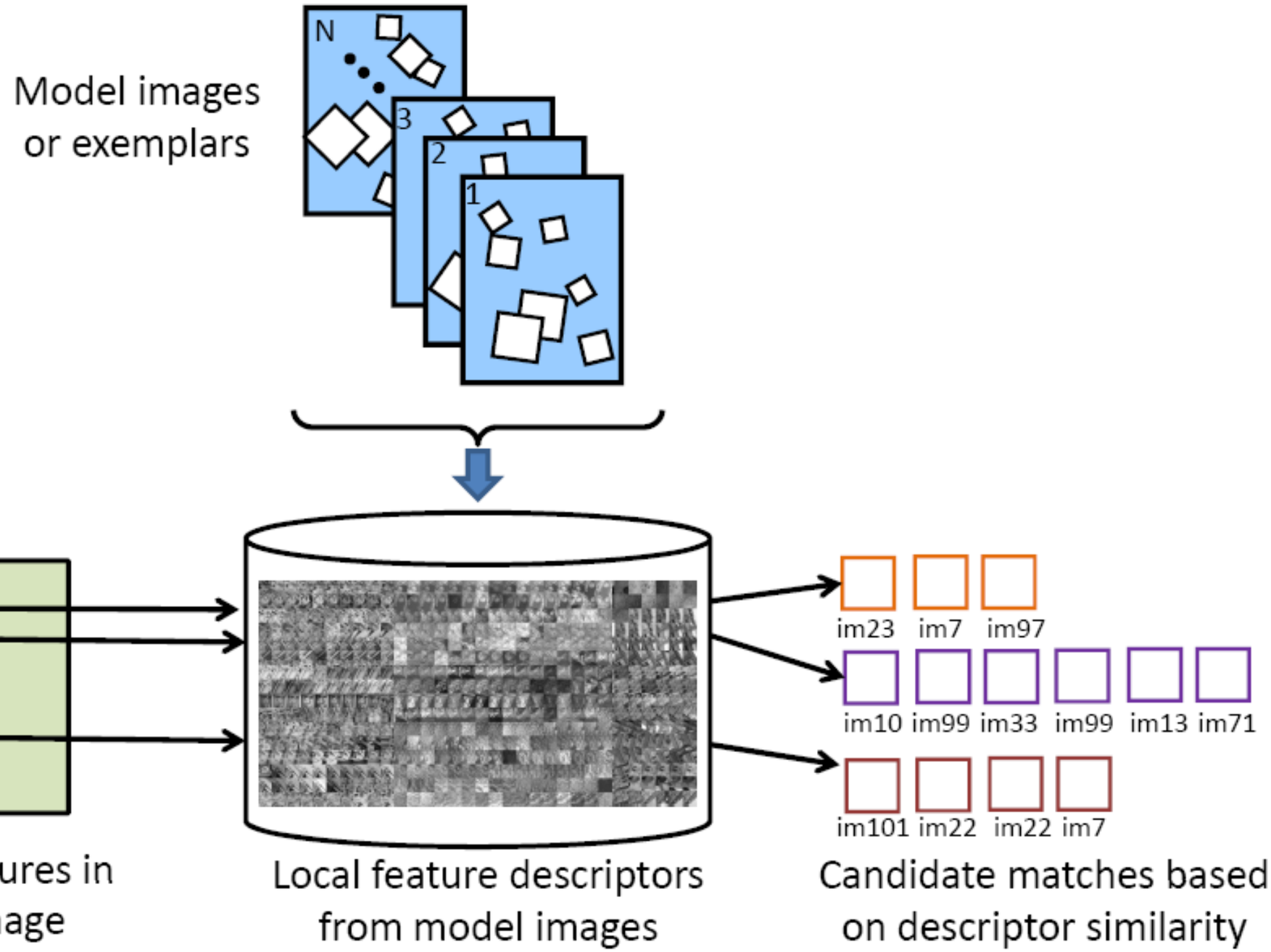


Image credit: K. Grauman and B. Leibe

History of ideas in recognition

- 1960s – early 1990s: the geometric era
- 1990s: appearance-based models
- Mid-1990s: sliding window approaches
- Late 1990s: local features
- Early 2000s: parts-and-shape models

Parts-and-shape models

- Model:
  - Object as a set of parts
  - Relative locations between parts
  - Appearance of part

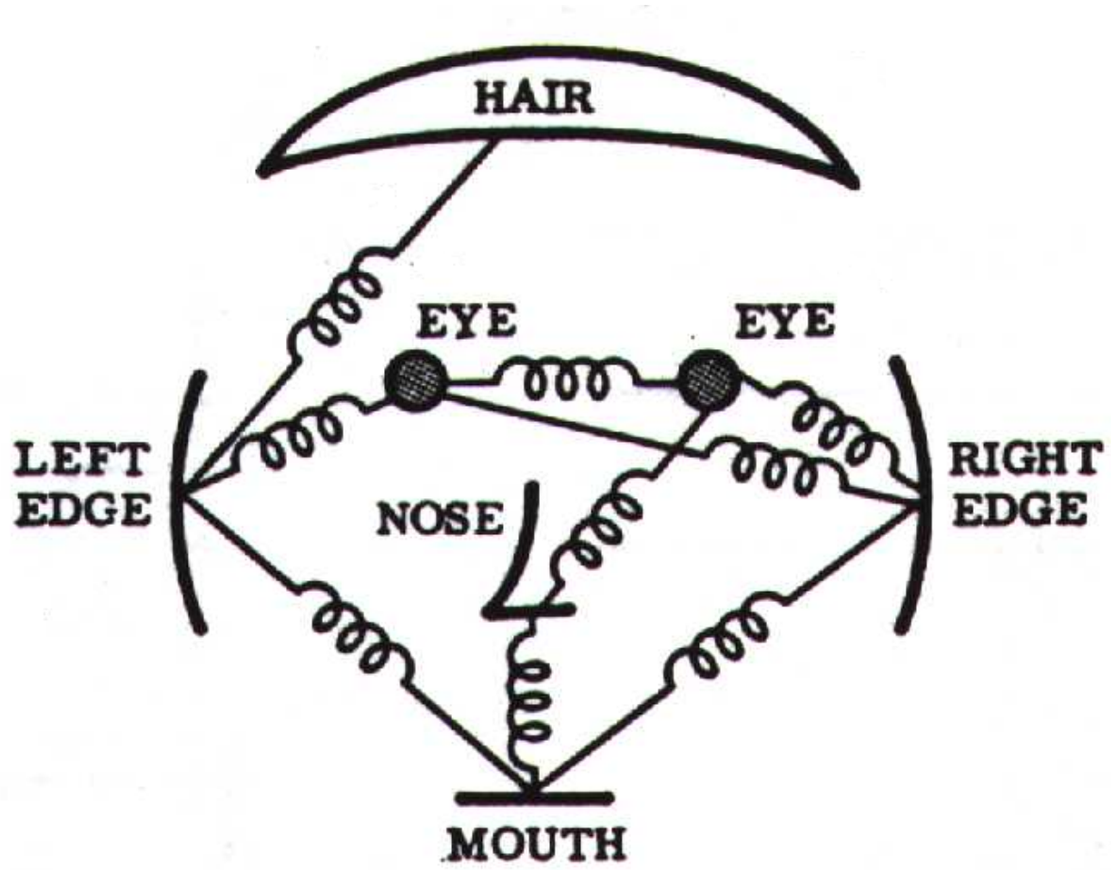
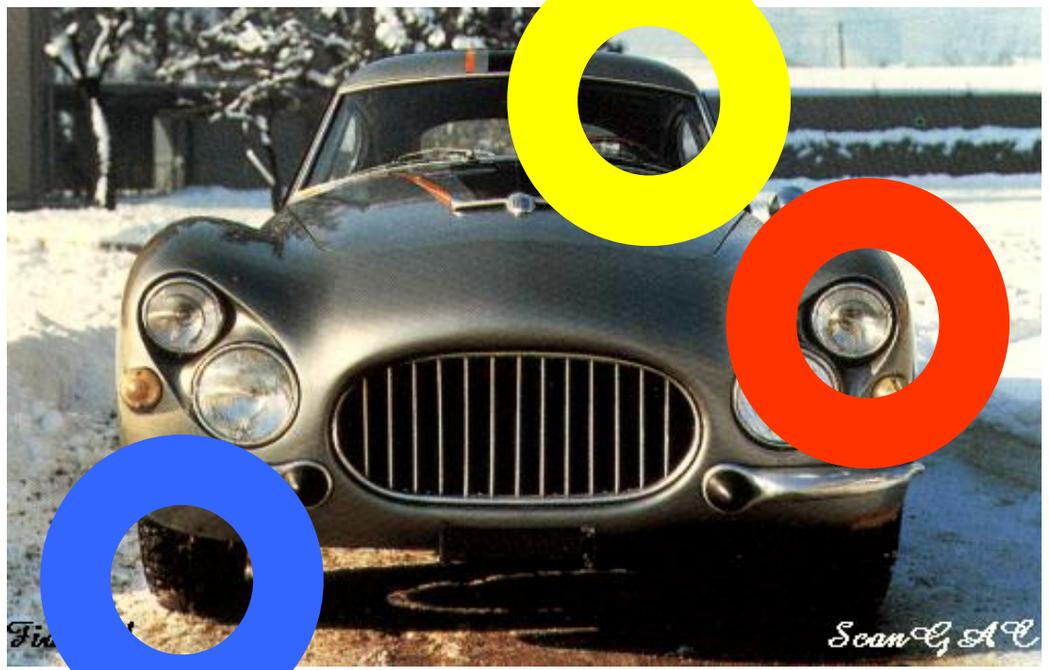


Figure from [Fischler & Elschlager 73]

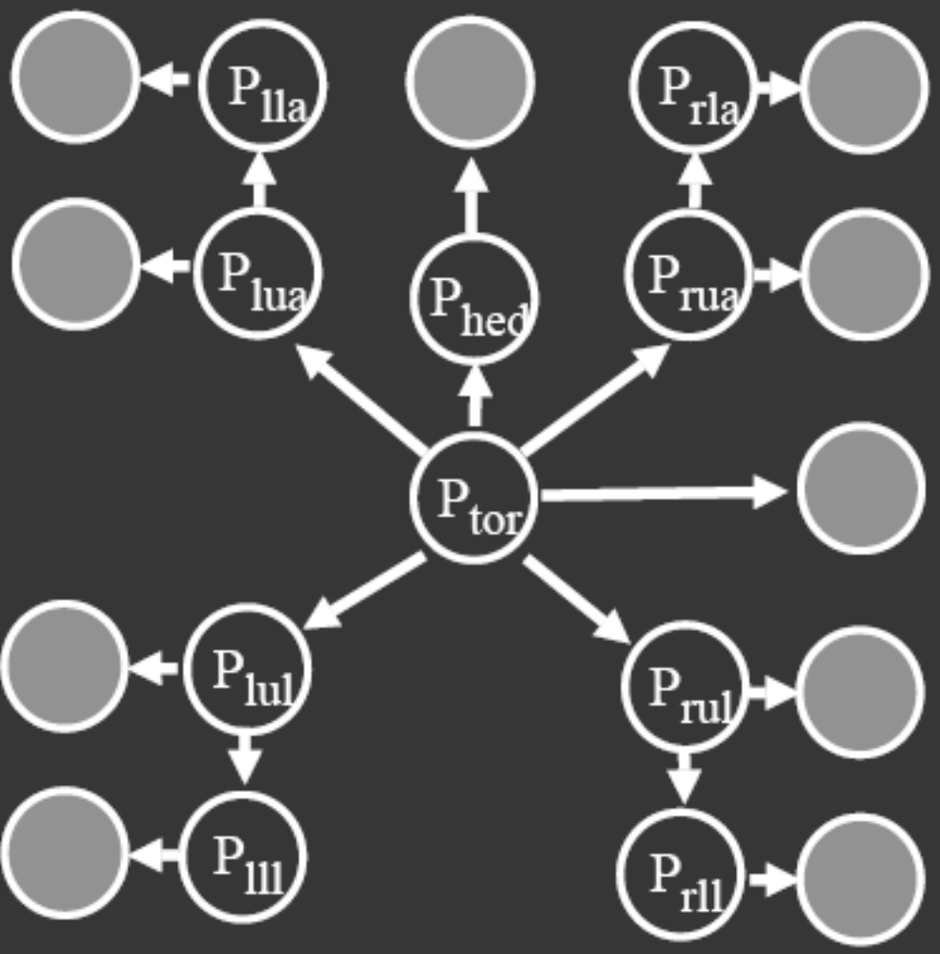
Constellation models

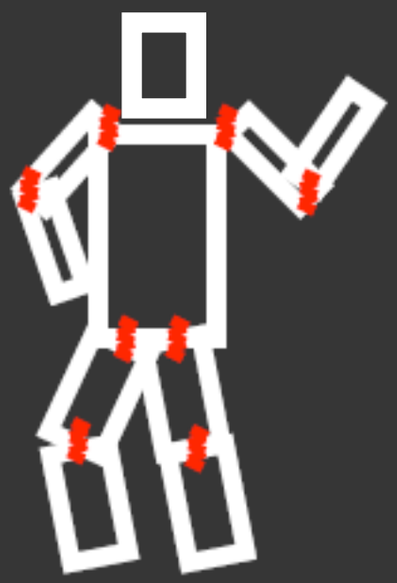


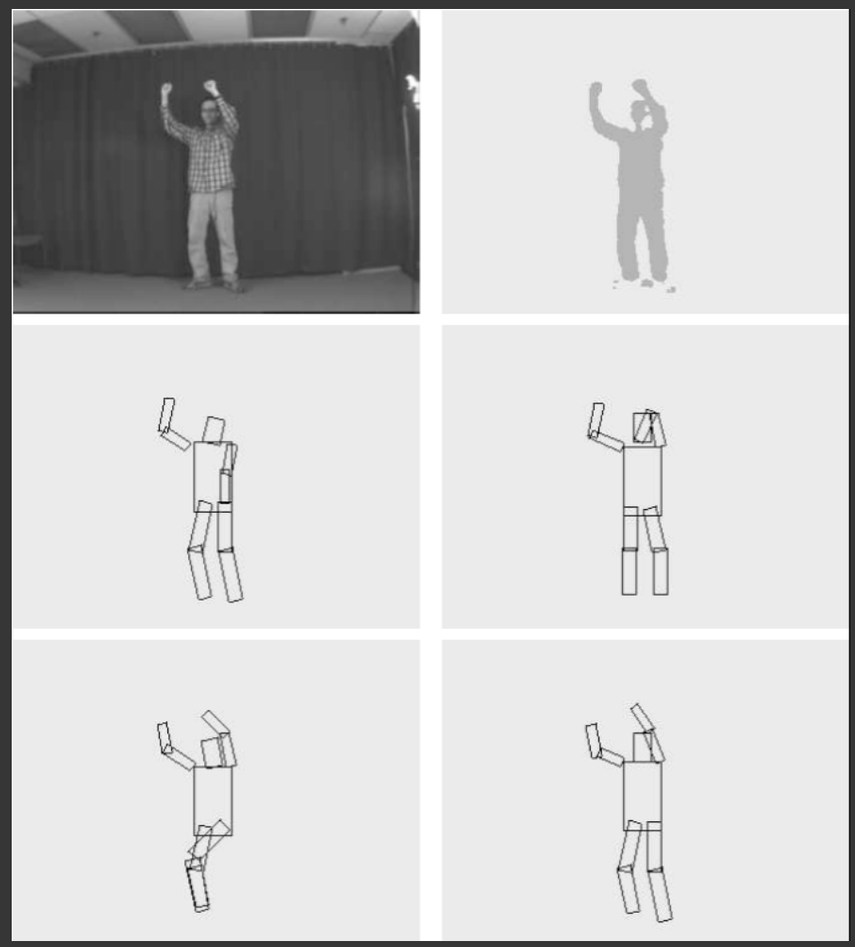
Weber, Welling & Perona (2000), Fergus, Perona & Zisserman (2003)

Pictorial structure model

Fischler and Elschlager(73), Felzenszwalb and Huttenlocher(00)





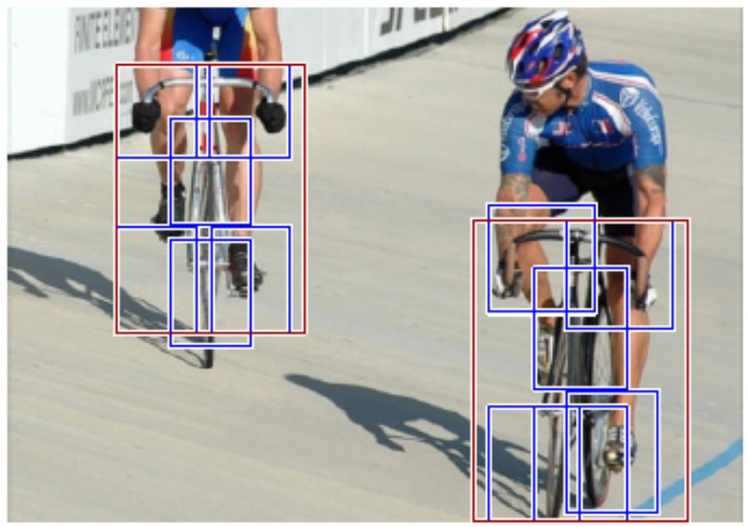


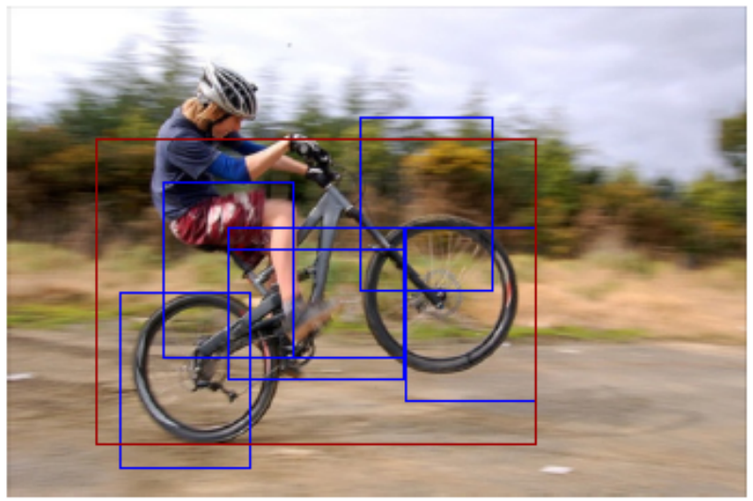
$$\Pr(P_{tor}, P_{arm}, \ldots | Im) \propto \prod_{i,j} \Pr(P_i | P_j) \prod_i \Pr(Im(P_i))$$

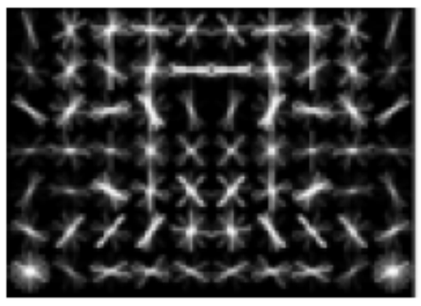
part geometry

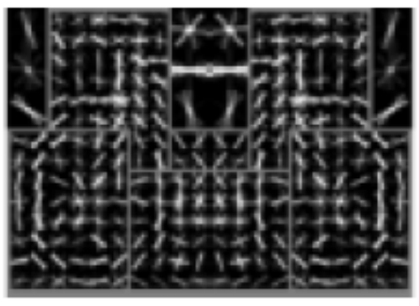
part appearance

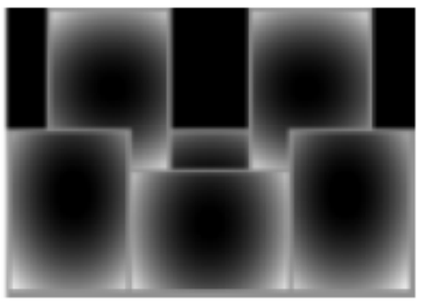
Discriminatively trained part-based models

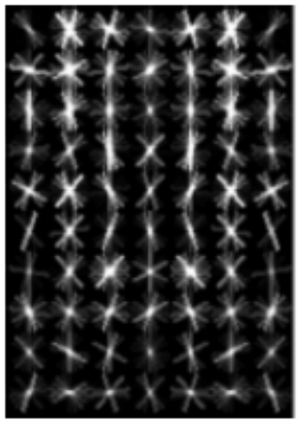


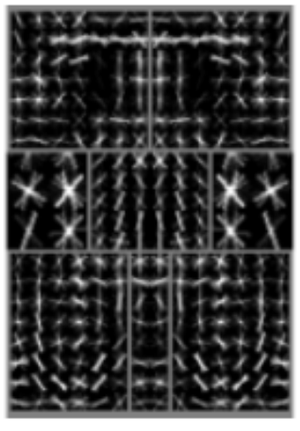


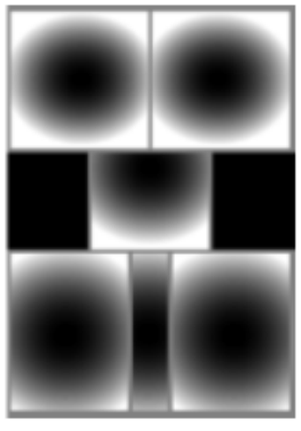












P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, PAMI 2009,

“Object Detection with Discriminatively Trained Part-Based Models”

History of ideas in recognition

- 1960s – early 1990s: the geometric era — No digital cameras! Slow compute!
- 1990s: appearance-based models — Slow compute!
- Mid-1990s: sliding window approaches
- Late 1990s: local features
- Early 2000s: parts-and-shape models
- Mid-2000s: bags of features — Early GPU compute.

Svetlana Lazebnik

History of ideas in recognition

- 1960s – early 1990s: the geometric era — No digital cameras! Slow compute!
- 1990s: appearance-based models — Slow compute!
- Mid-1990s: sliding window approaches
- Late 1990s: local features
- Early 2000s: parts-and-shape models
- Mid-2000s: bags of features (next!) — Early GPU compute.
- *Present trends:*
  Combined local and global methods, context, deep learning — GPU/cloud compute.

The machine learning framework

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

f() = "apple"

f() = "tomato"

f() = "cow"

Slide credit: L. Lazebnik

The machine learning framework

$$f(\mathbf{x}) = y$$

Prediction function (or *classifier*)

Image feature

Output (label)

- **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 unseen *test example* $\mathbf{x}_u$ and output the predicted value $y_u = f(\mathbf{x}_u)$ to *classify* $\mathbf{x}_u$.

Slide credit: L. Lazebnik

Dataset

Training Images

- Train classifier

Validation Images

- Measure error
- Tune model hyperparameters

Testing Images

- Secret labels
- Measure error

Random train/validate splits = cross validation

Training

Images

Labels

Image Features

Training

Trained classifier

Testing

Image *not in training set*

Image Features

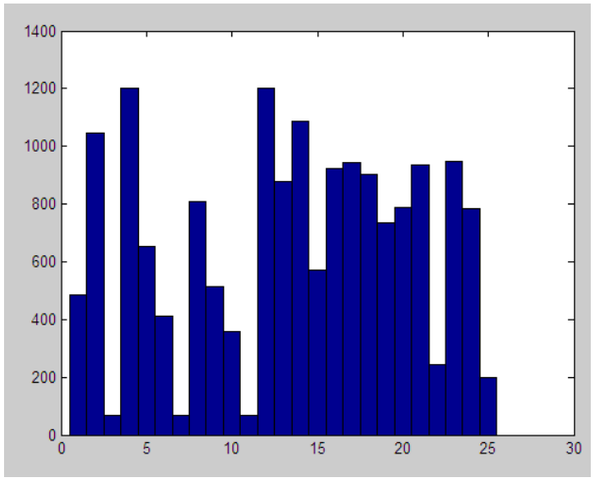
Apply classifier

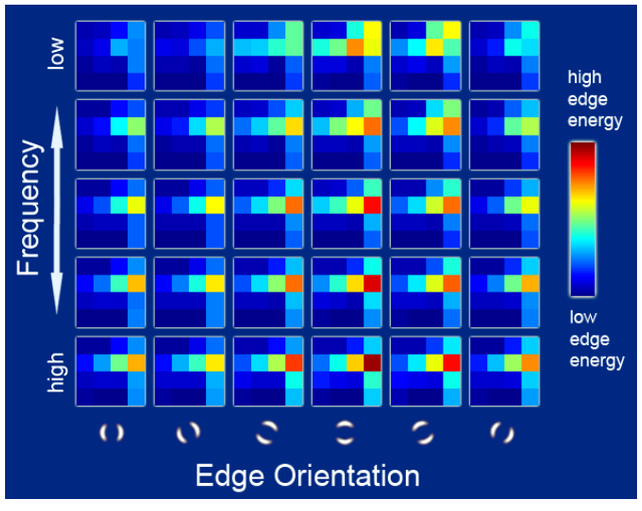
Prediction

Slide credit: D. Hoiem and L. Lazebnik

Features

- Raw pixels
- Histograms
- Templates
- SIFT descriptors
  - GIST
  - ORB
  - HOG….





L. Lazebnik

Training

Images

Labels

Image Features

Training

Trained classifier

Testing

Image *not in training set*

Image Features

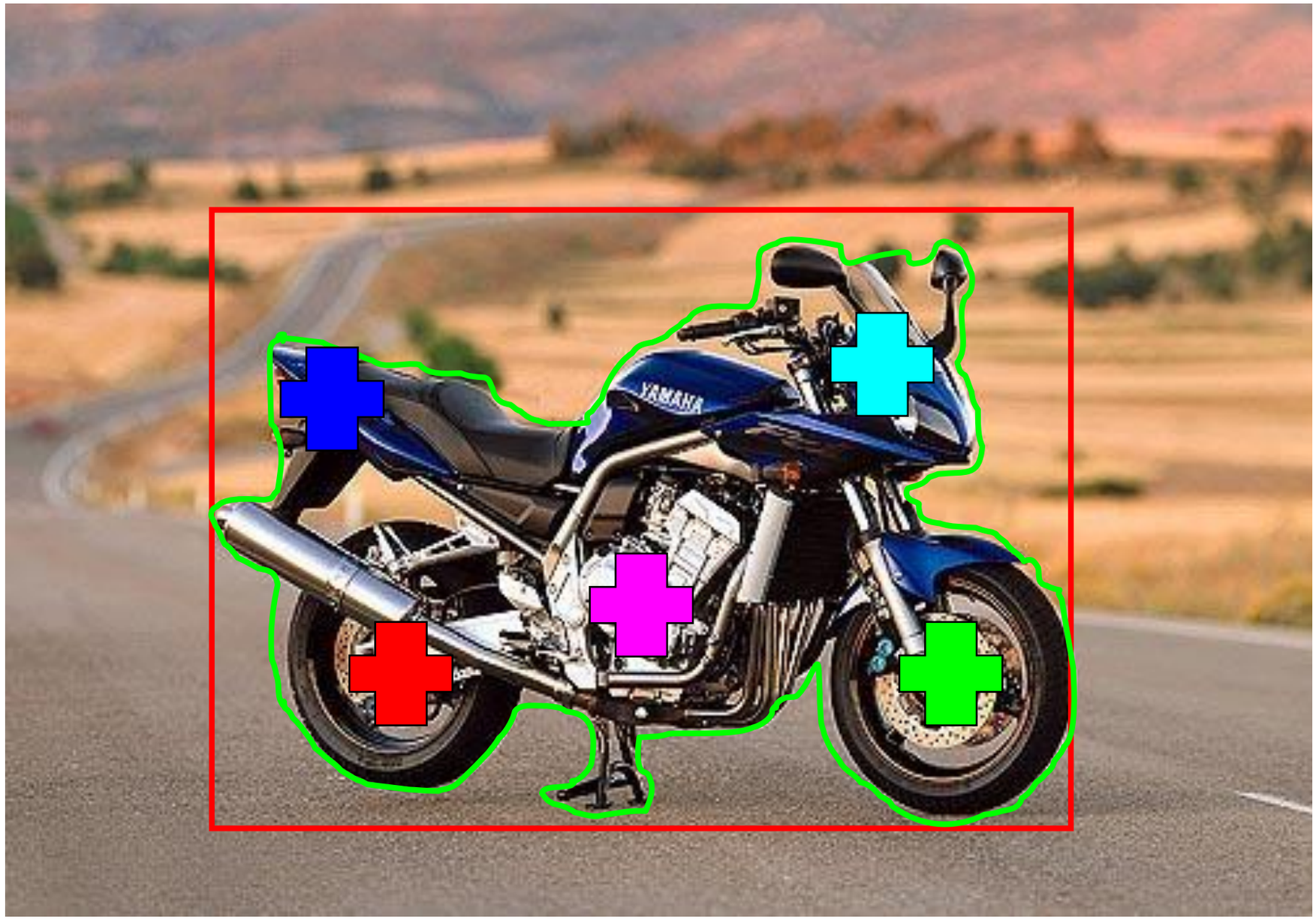
Apply classifier

Prediction

Slide credit: D. Hoiem and L. Lazebnik

Recognition task and supervision

- Images in the training set must be annotated with the "correct answer" that the model is expected to produce



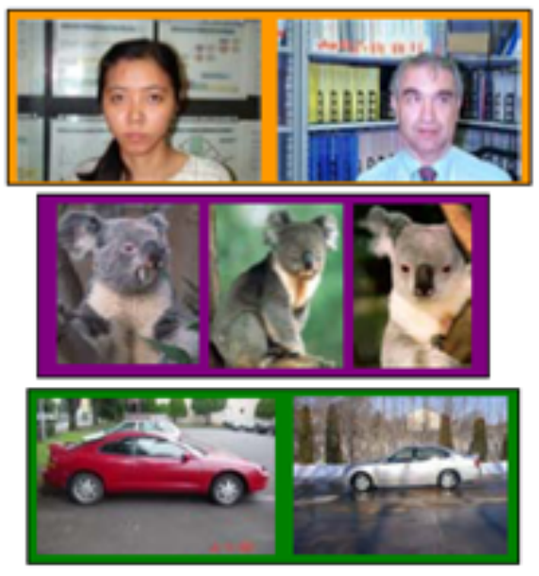
L. Lazebnik

Spectrum of supervision

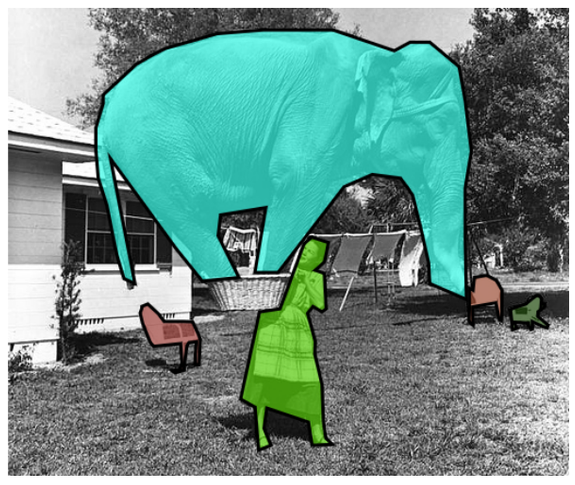
Less

More

E.G., ImageNet



E.G., MS Coco



Unsupervised

“Weakly” supervised

Fully supervised

Fuzzy; definition depends on task

‘Semi-supervised’: small partial labeling

Training

Images

Labels

Image Features

Training

Trained classifier

Testing

Image *not in training set*

Image Features

Apply classifier

Prediction

Slide credit: D. Hoiem and L. Lazebnik

The machine learning framework

$$f(\mathbf{x}) = y$$

Prediction function (or *classifier*)

Image feature

Output (label)

- **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 unseen *test example* $\mathbf{x}_u$ and output the predicted value $y_u = f(\mathbf{x}_u)$ to *classify* $\mathbf{x}_u$.

Slide credit: L. Lazebnik

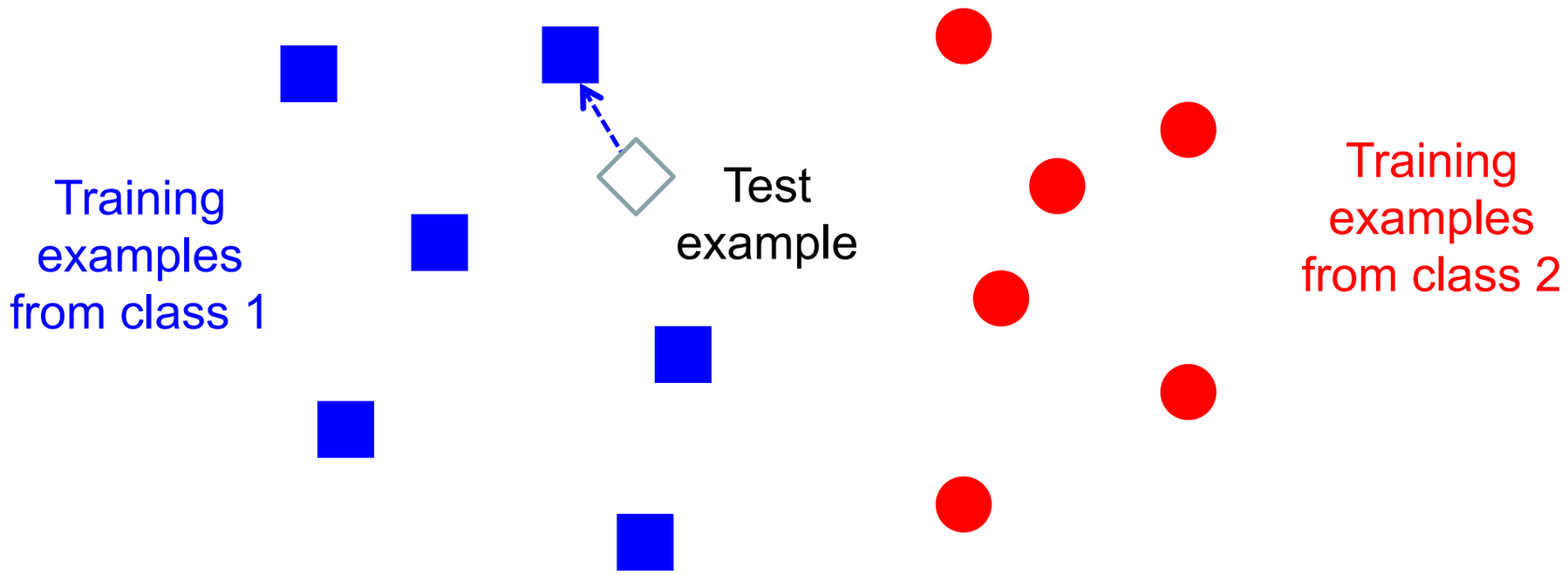
Classification

Assign **x** to one of two (or more) classes.

A decision rule divides input space into *decision regions* separated by *decision boundaries* – literally boundaries in the space of the features.

L. Lazebnik

Classifiers: Nearest neighbor



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

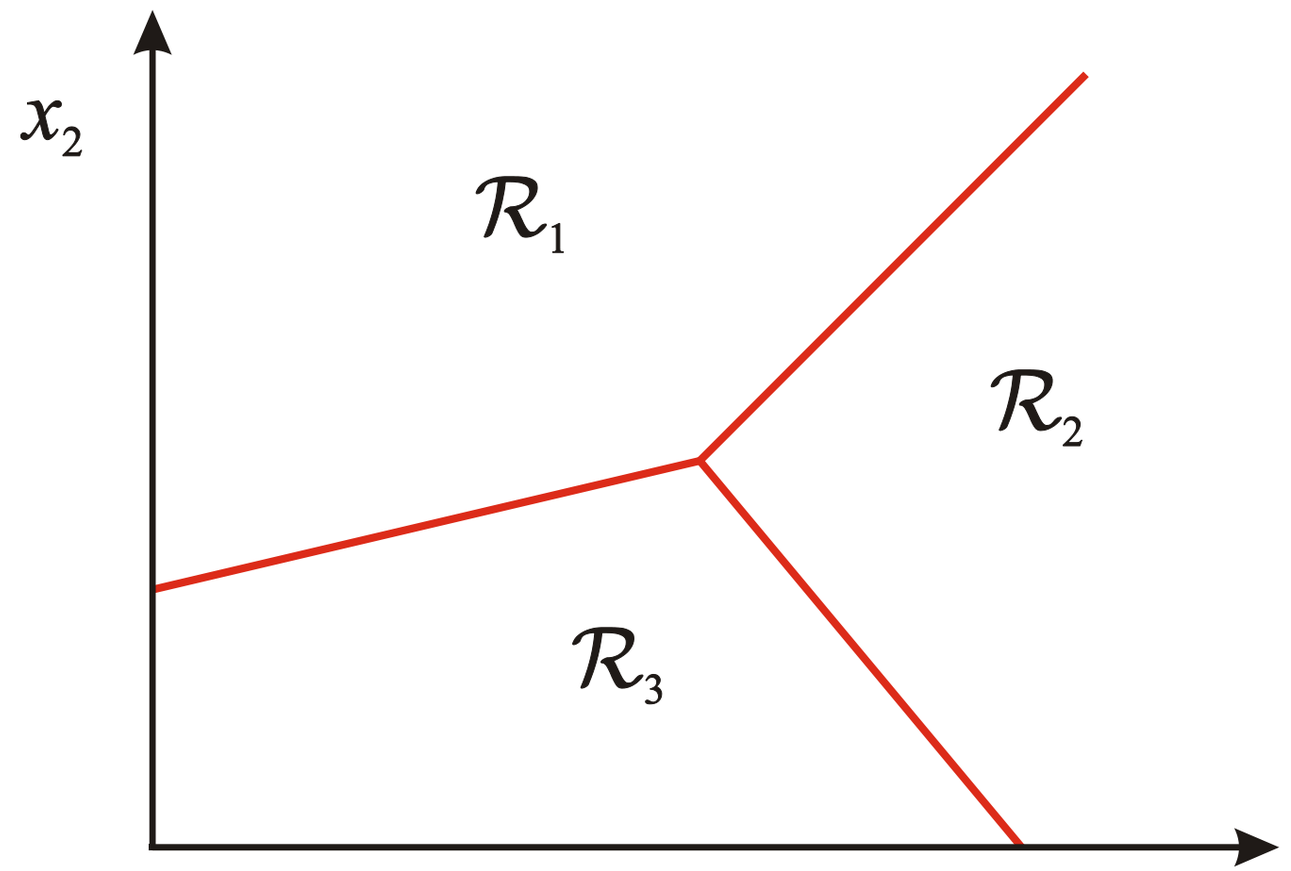
- All we need is a distance function for our inputs
- No training required!

Slide credit: L. Lazebnik

Classification

Assign **x** to one of two (or more) classes.

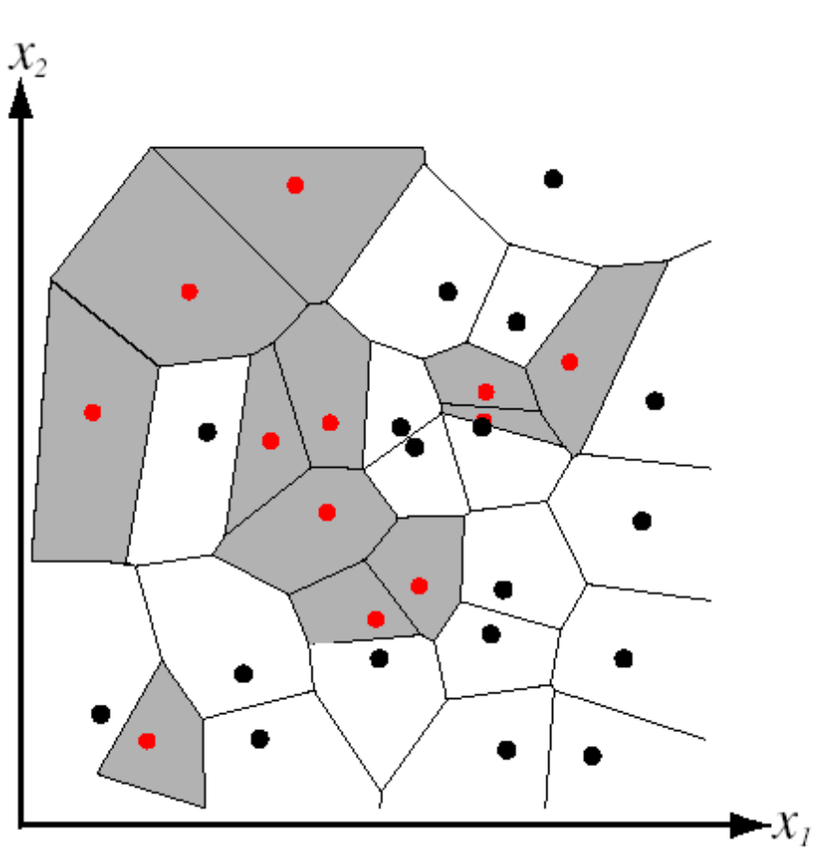
A decision rule divides input space into *decision regions* separated by *decision boundaries* – literally boundaries in the space of the features.



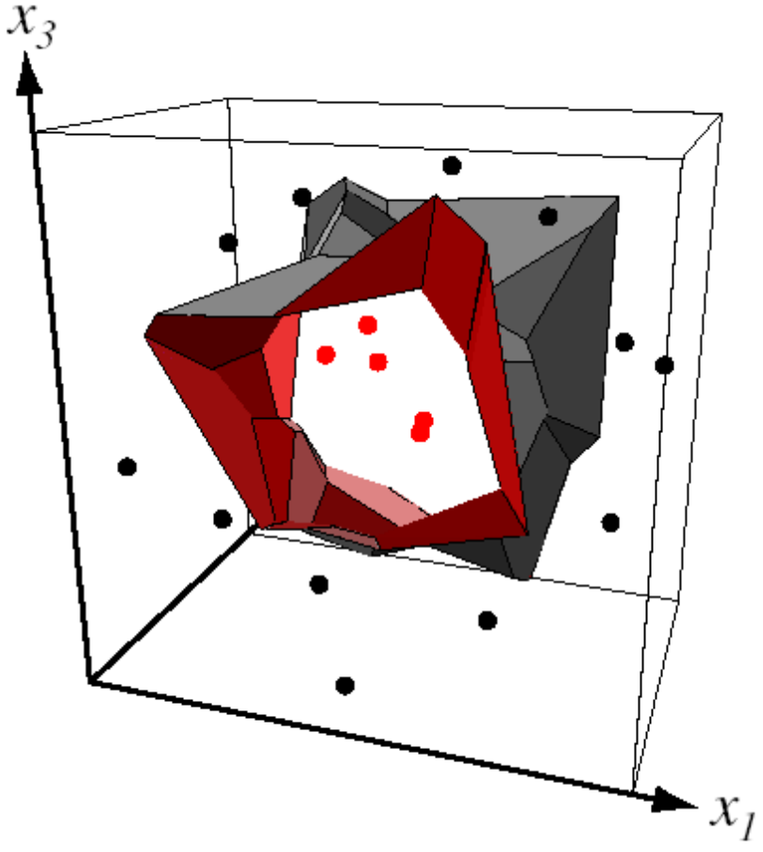
L. Lazebnik

Decision boundary for Nearest Neighbor Classifier

Divides input space into *decision regions* separated by *decision boundaries – Voronoi.*



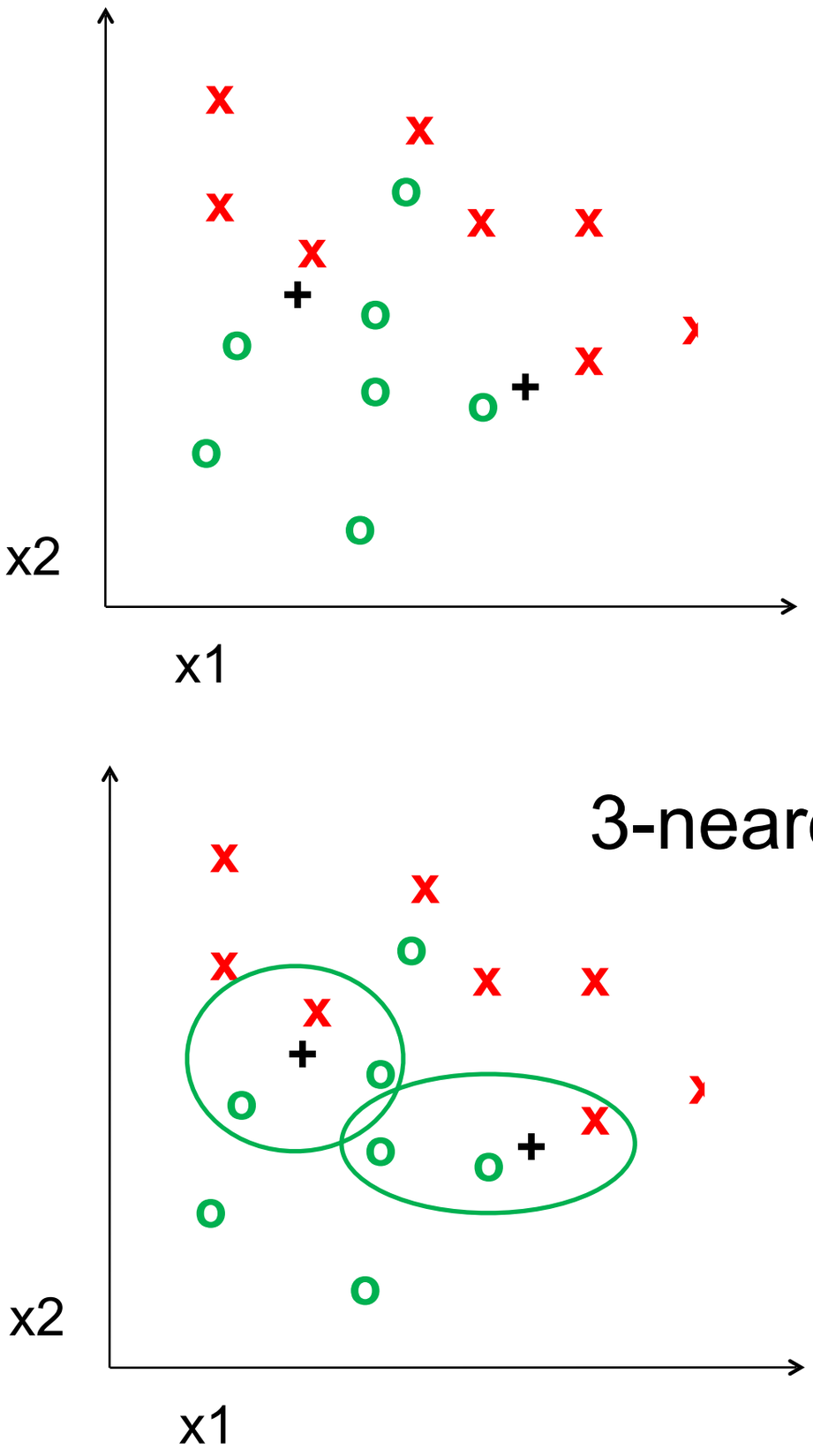
from Duda *et al.*

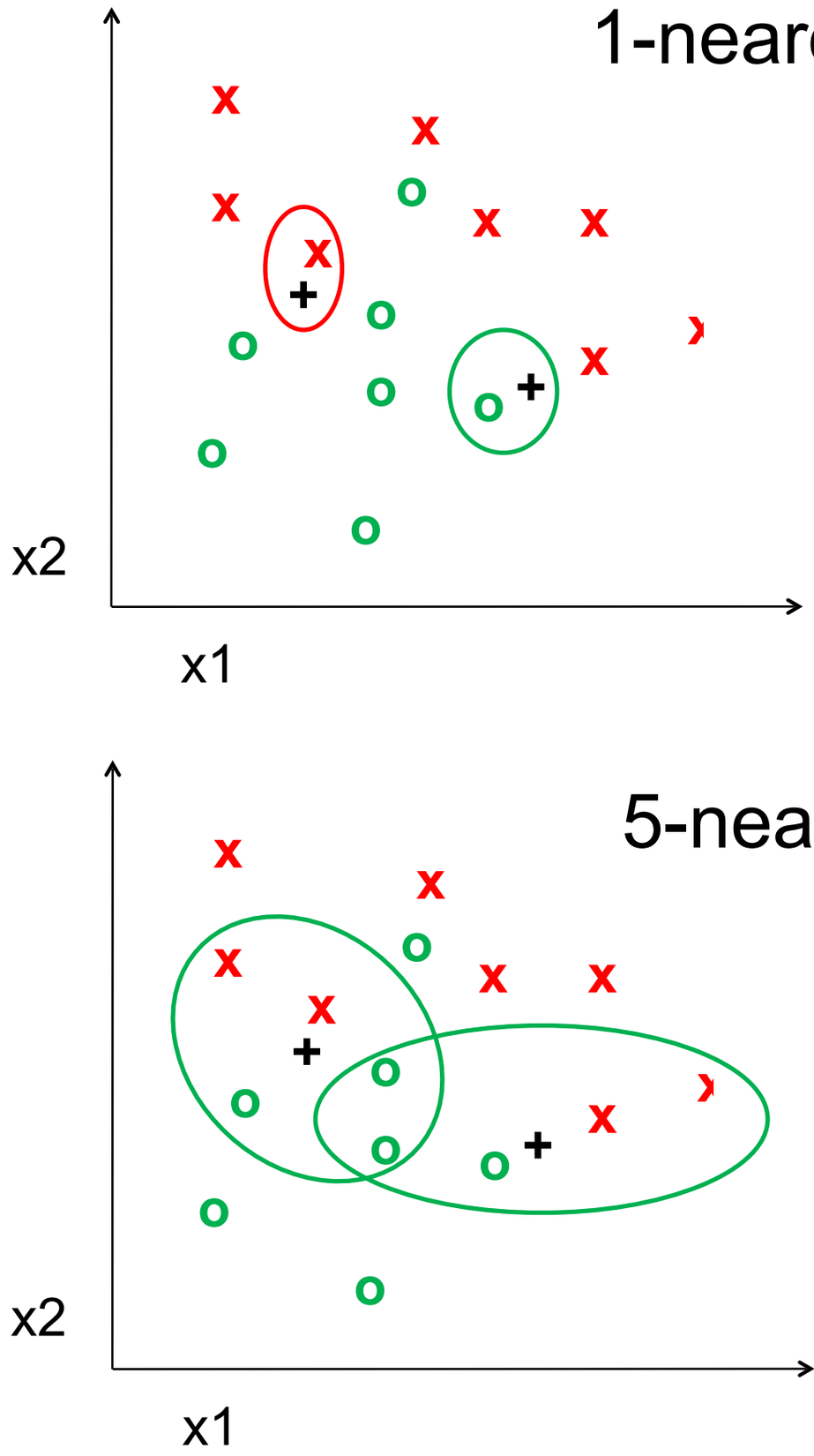


Voronoi partitioning of feature space for two-category 2D and 3D data

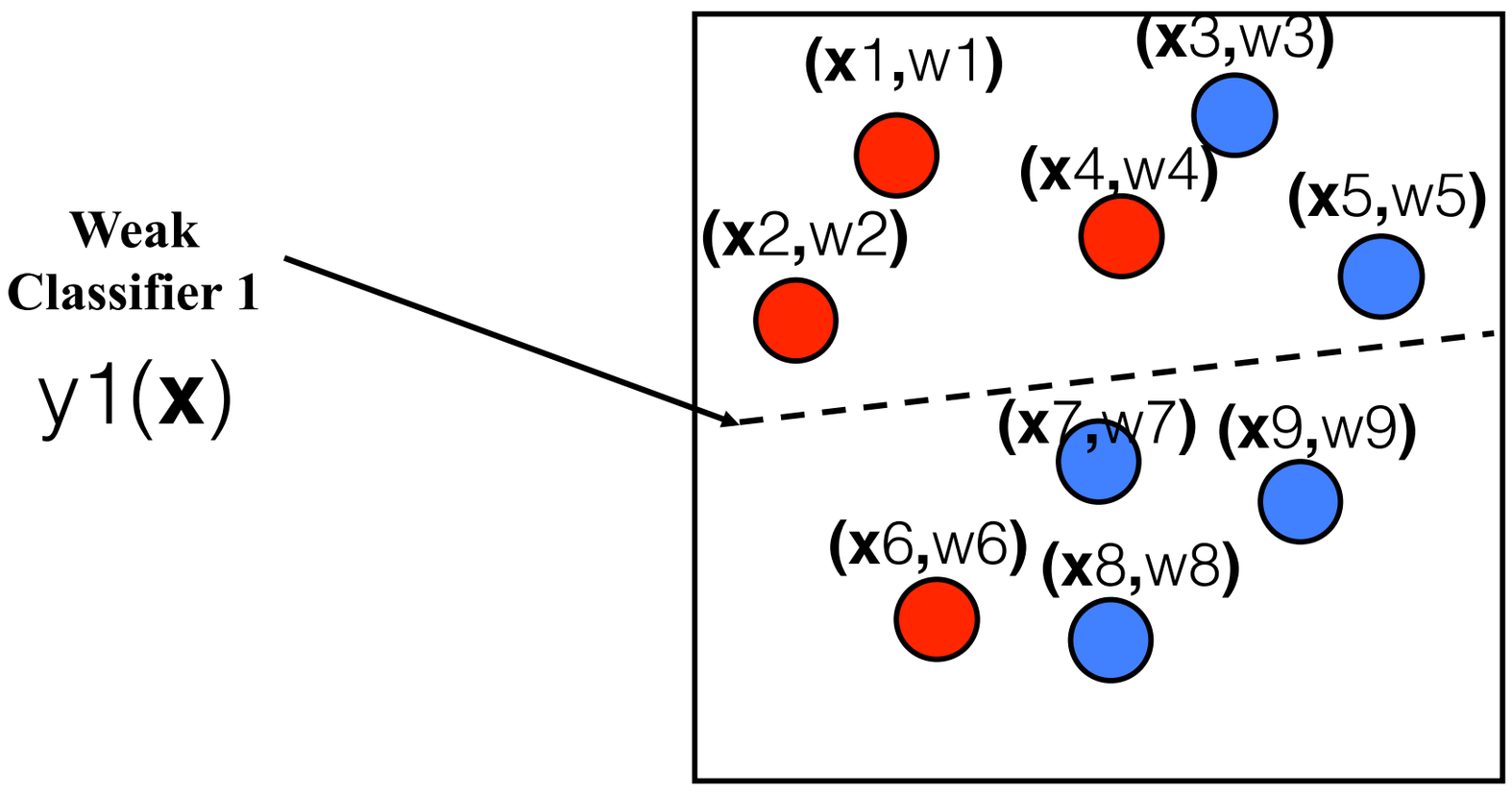
Source: D. Lowe

k-nearest neighbor





Discriminative classifiers - Boosting



$$\varepsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

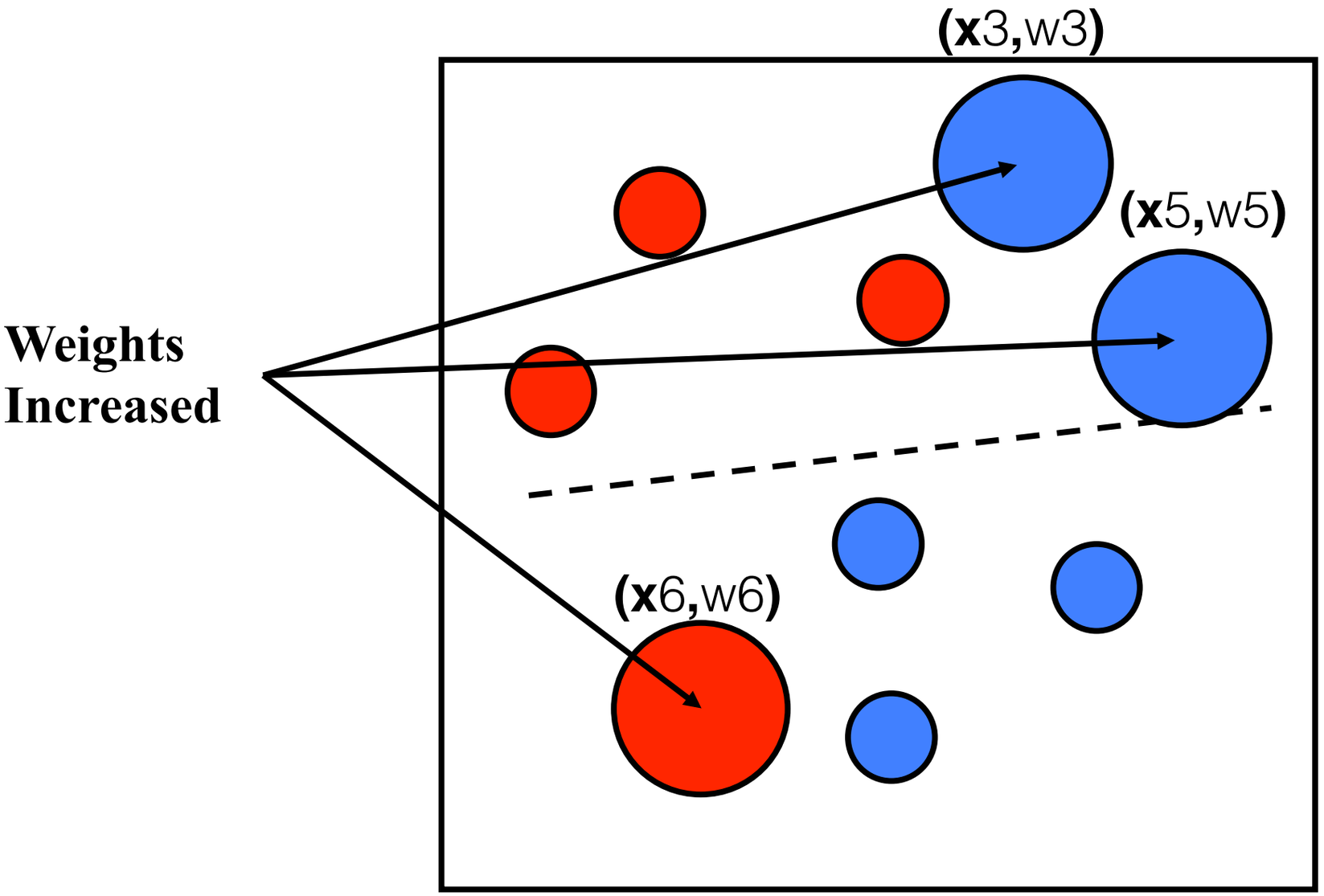
$$\alpha_m = \ln\left\{\frac{1-\varepsilon_m}{\varepsilon_m}\right\}$$

$$\begin{cases} I(y_m(\mathbf{x}_n) \neq t_n) = 1, y_m(\mathbf{x}_n) \neq t_n \\ I(y_m(\mathbf{x}_n) \neq t_n) = 0, y_m(\mathbf{x}_n) = t_n \end{cases}$$

error function: $J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$

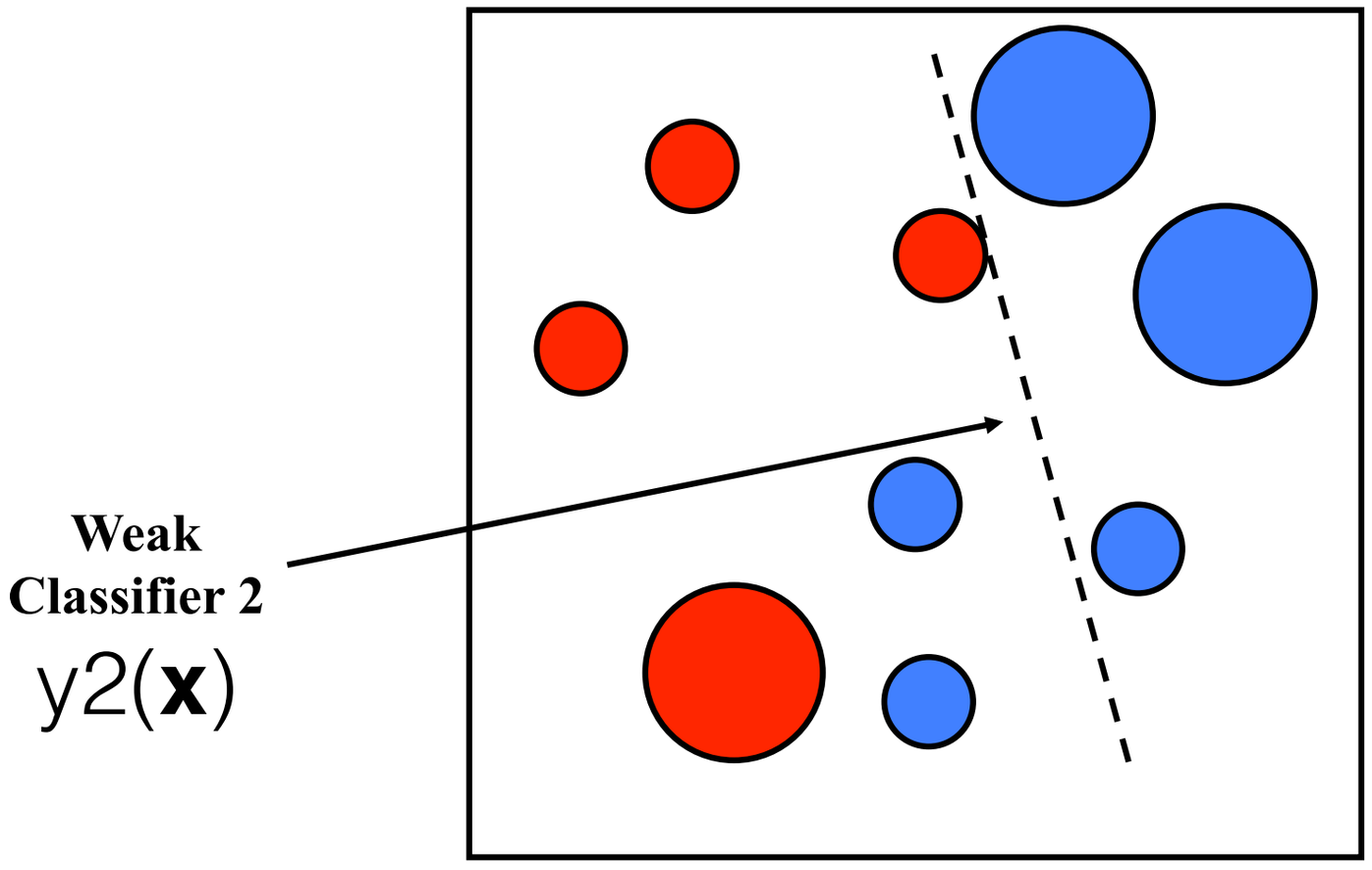
Slide credit: Paul Viola

Discriminative classifiers - Boosting

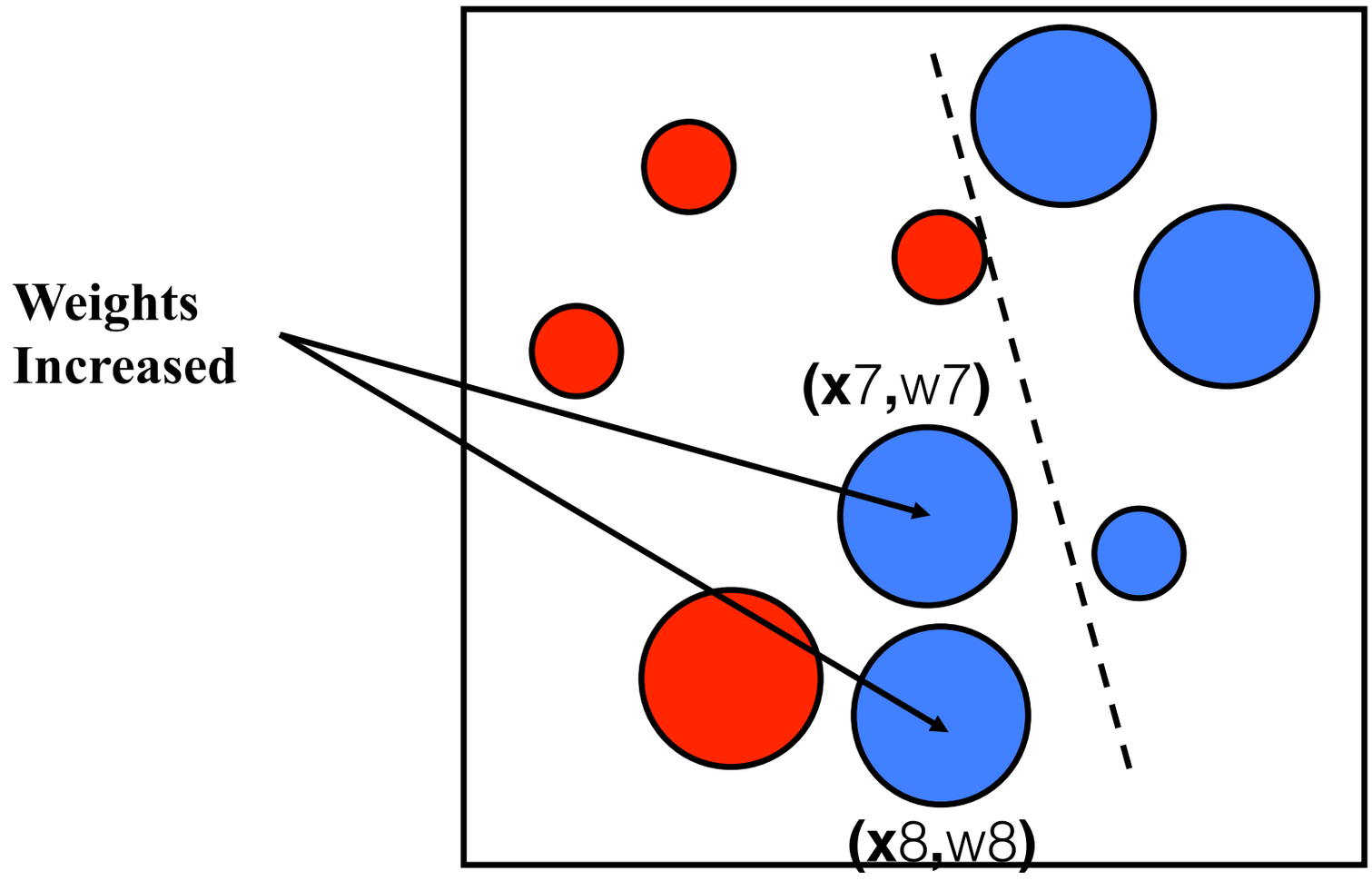


$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\}$$

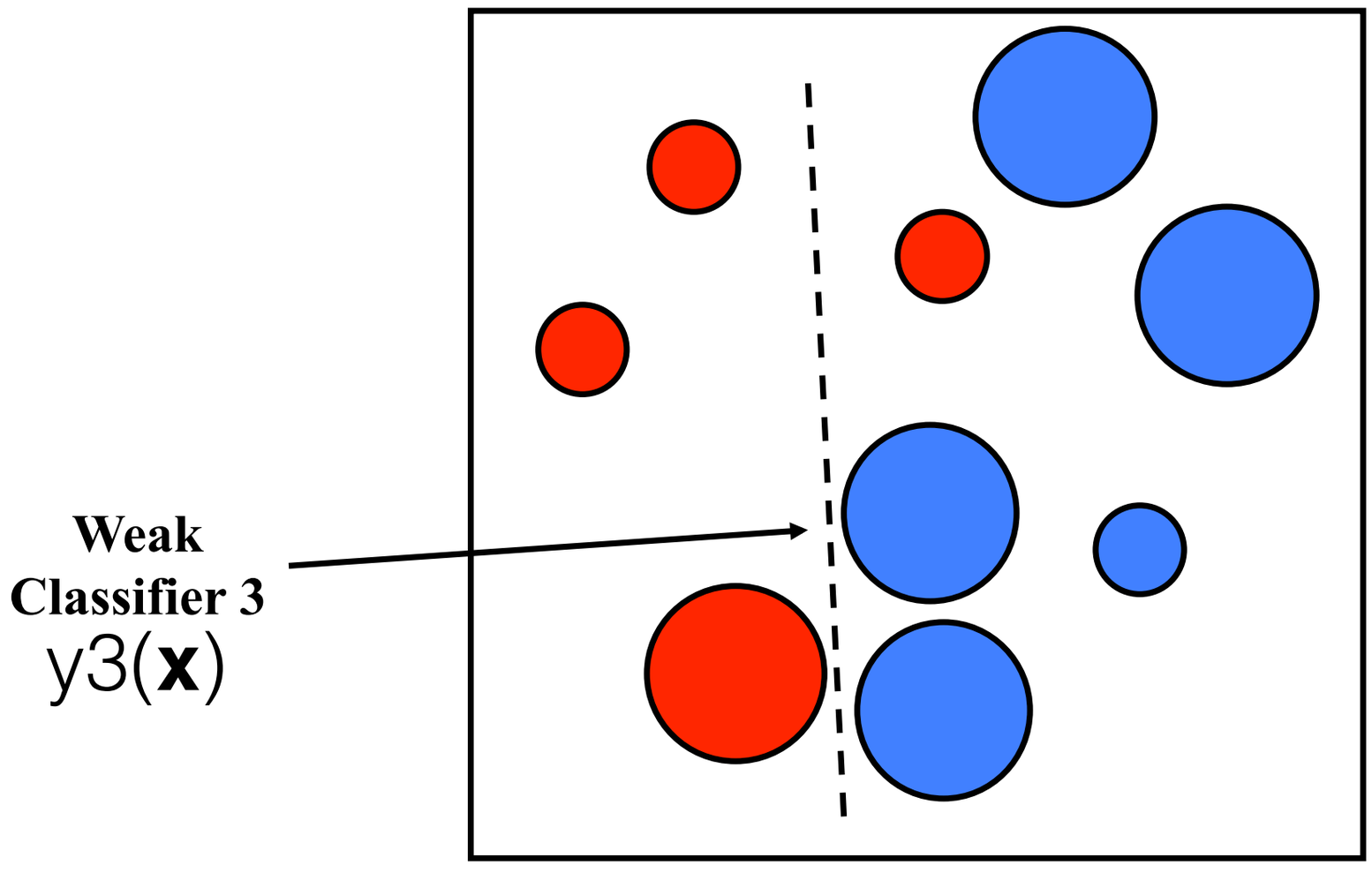
Discriminative classifiers - Boosting



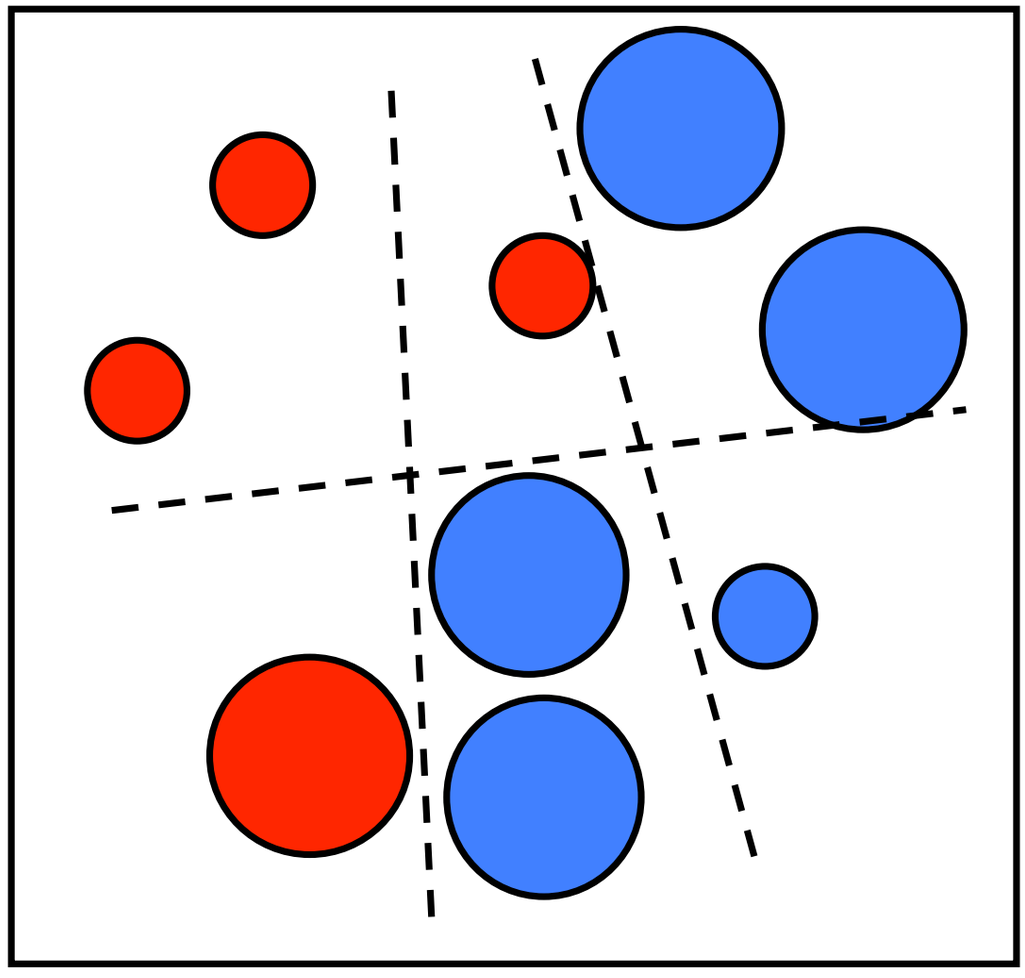
Discriminative classifiers - Boosting



Discriminative classifiers - Boosting



Discriminative classifiers - Boosting



Final classifier is a combination of weak classifiers

$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right)$$

Boosting: training

- Initially, weight each training example equally
- In each boosting round:
  - Find the weak learner that achieves the lowest weighted training error
  - Raise weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
- Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)

Boosting: pros and cons

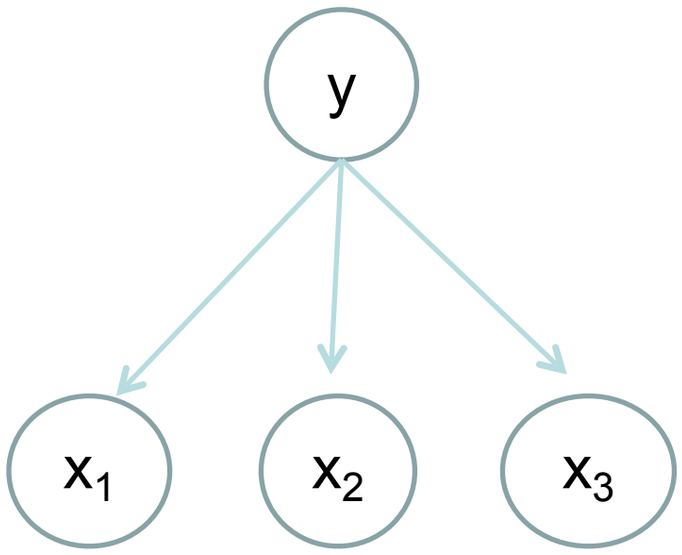
- Advantages of boosting
  - Integrates classification with feature selection
  - Complexity of training is linear in the number of training examples
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is fast
  - Easy to implement

- Disadvantages
  - Needs many training examples
  - Often found not to work as well as an alternative discriminative classifier, support vector machine (SVM)
    - especially for many-class problems

Slide credit: Lana Lazebnik

Classifier: Naïve Bayes

$$p(C_k \mid x_1, \ldots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$



- Conditional probability model over *classes* C_k

Classifier:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

Naïve Bayes model for classification

N patches

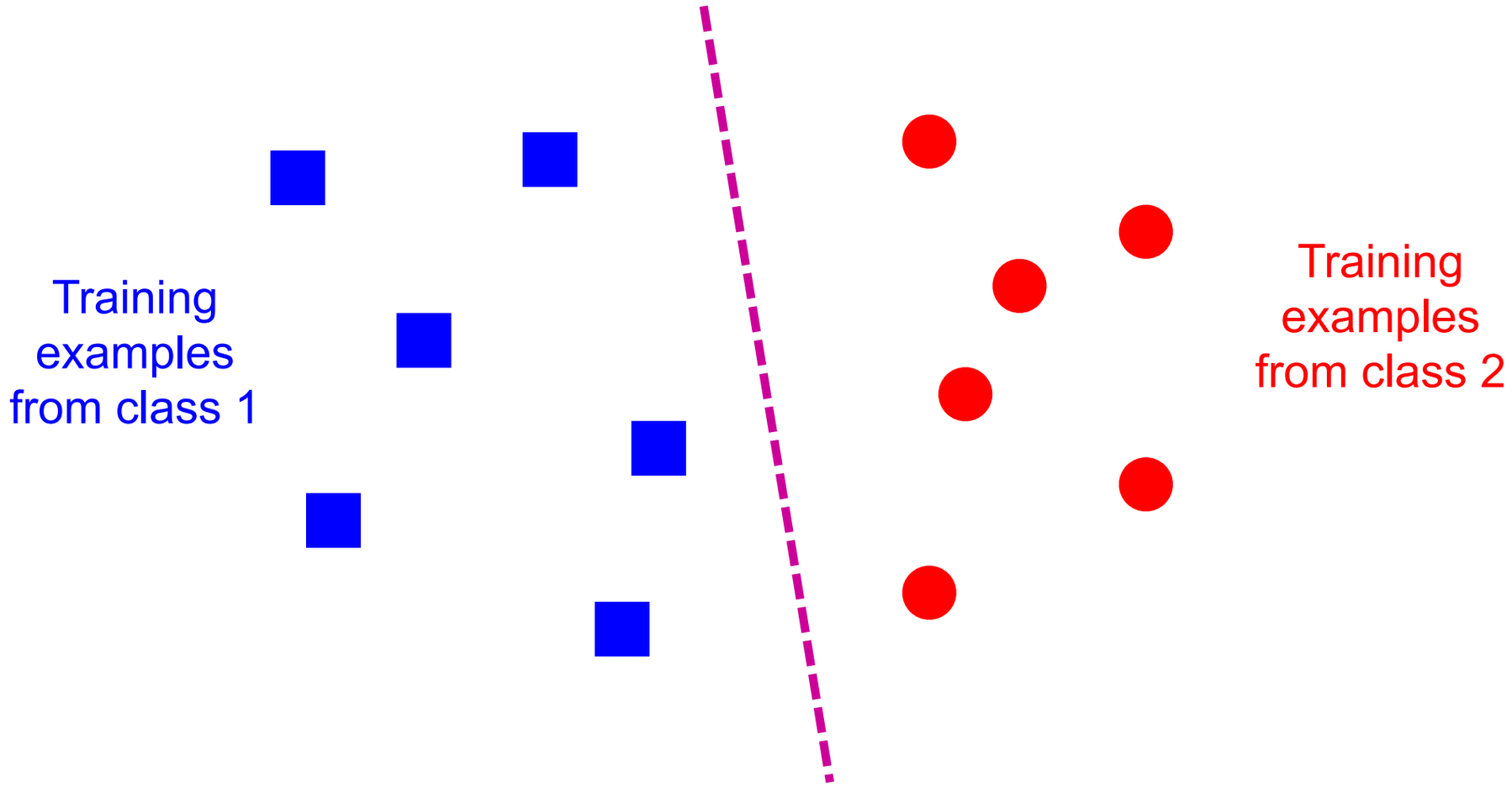
$$c^* = \arg\max_c p(c \mid w) \propto p(c)\,p(w \mid c) = p(c)\prod_{n=1}^{N} p(w_n \mid c)$$

Object class decision

Prior prob. of the object classes

Image likelihood given the class

Classifiers: Linear



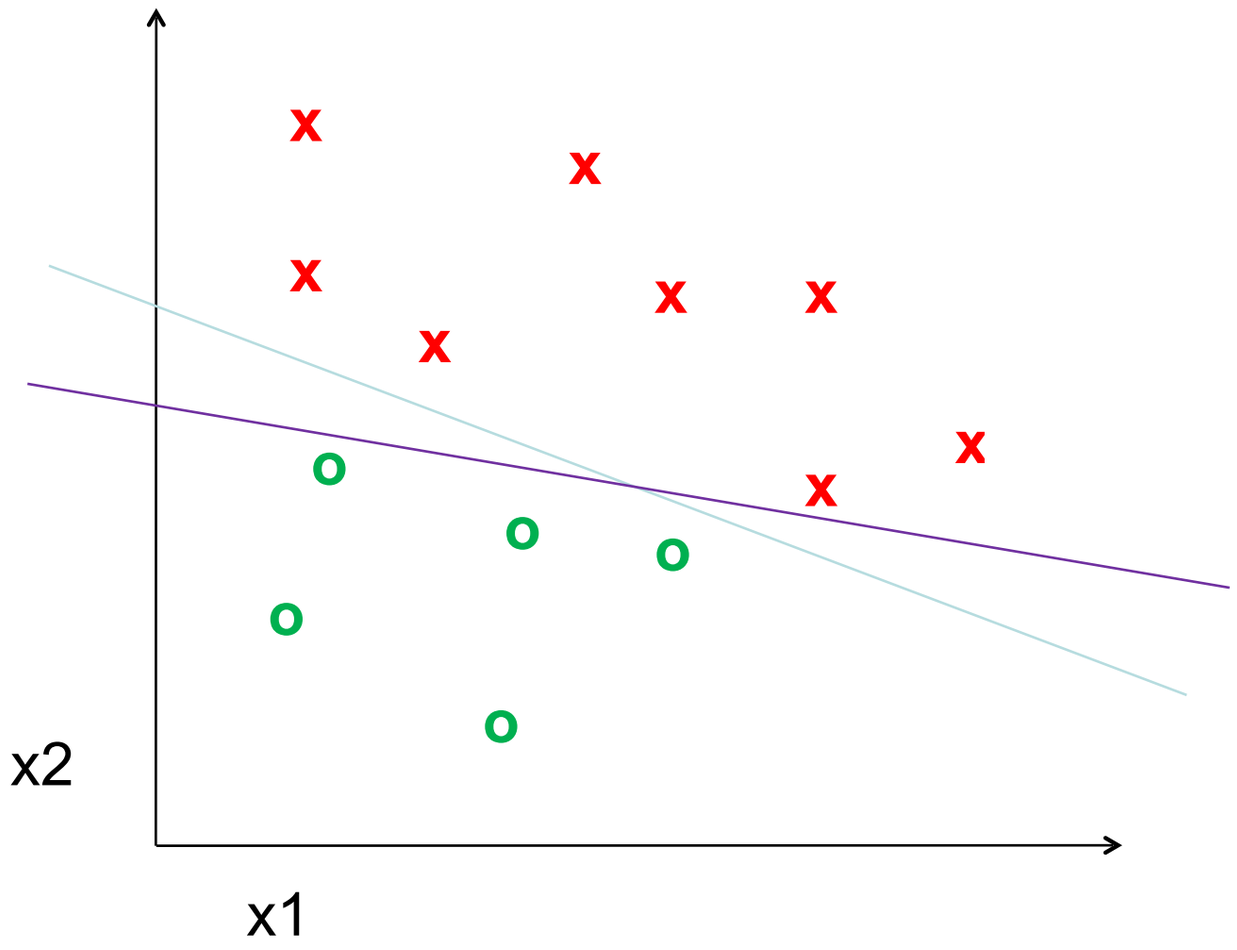
Find a *linear function* to separate the classes

Slide credit: L. Lazebnik

Classifiers: Linear SVM

Find a *linear function* to separate the classes:

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



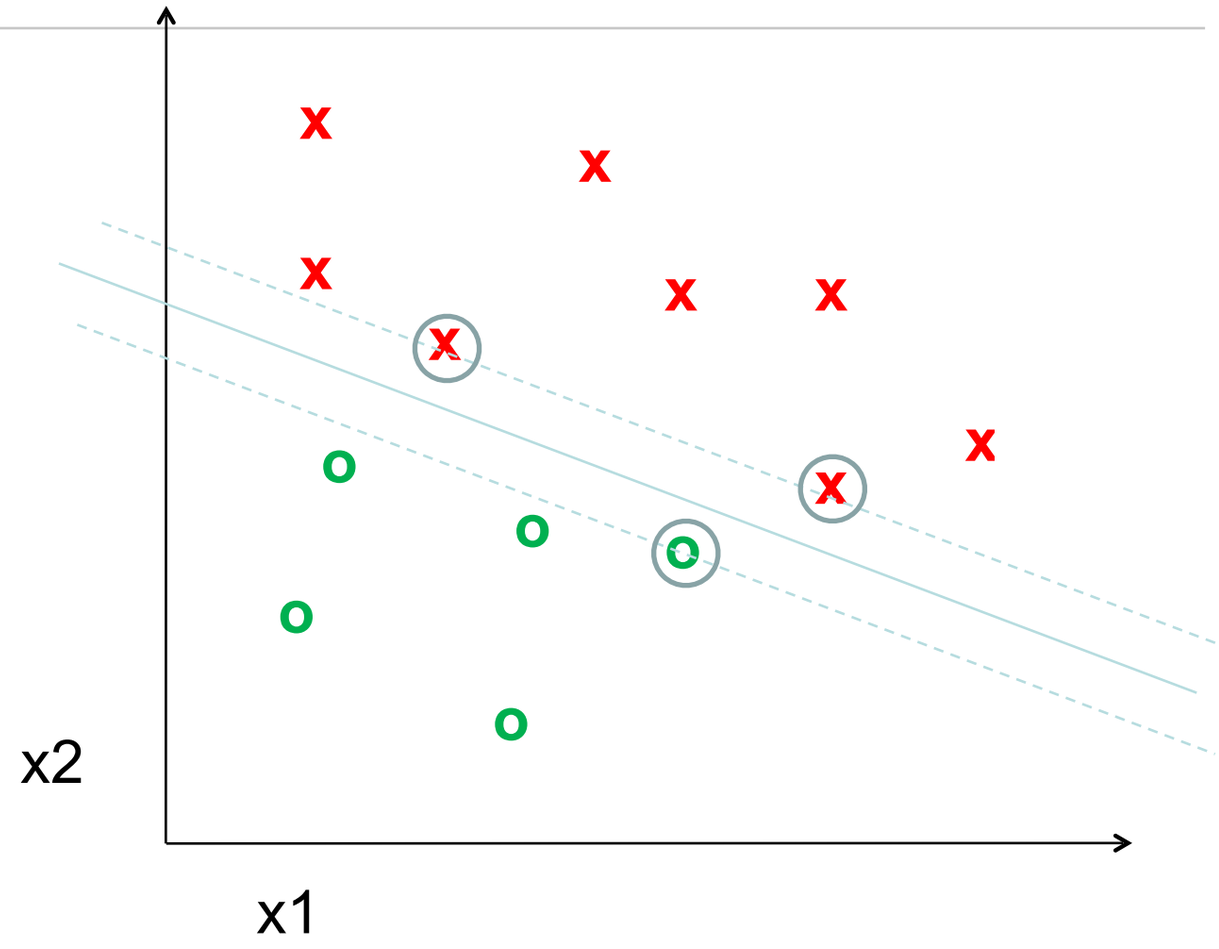
Classifiers: Linear SVM

Find a *linear function* to separate the classes:

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

How?

X = all data points



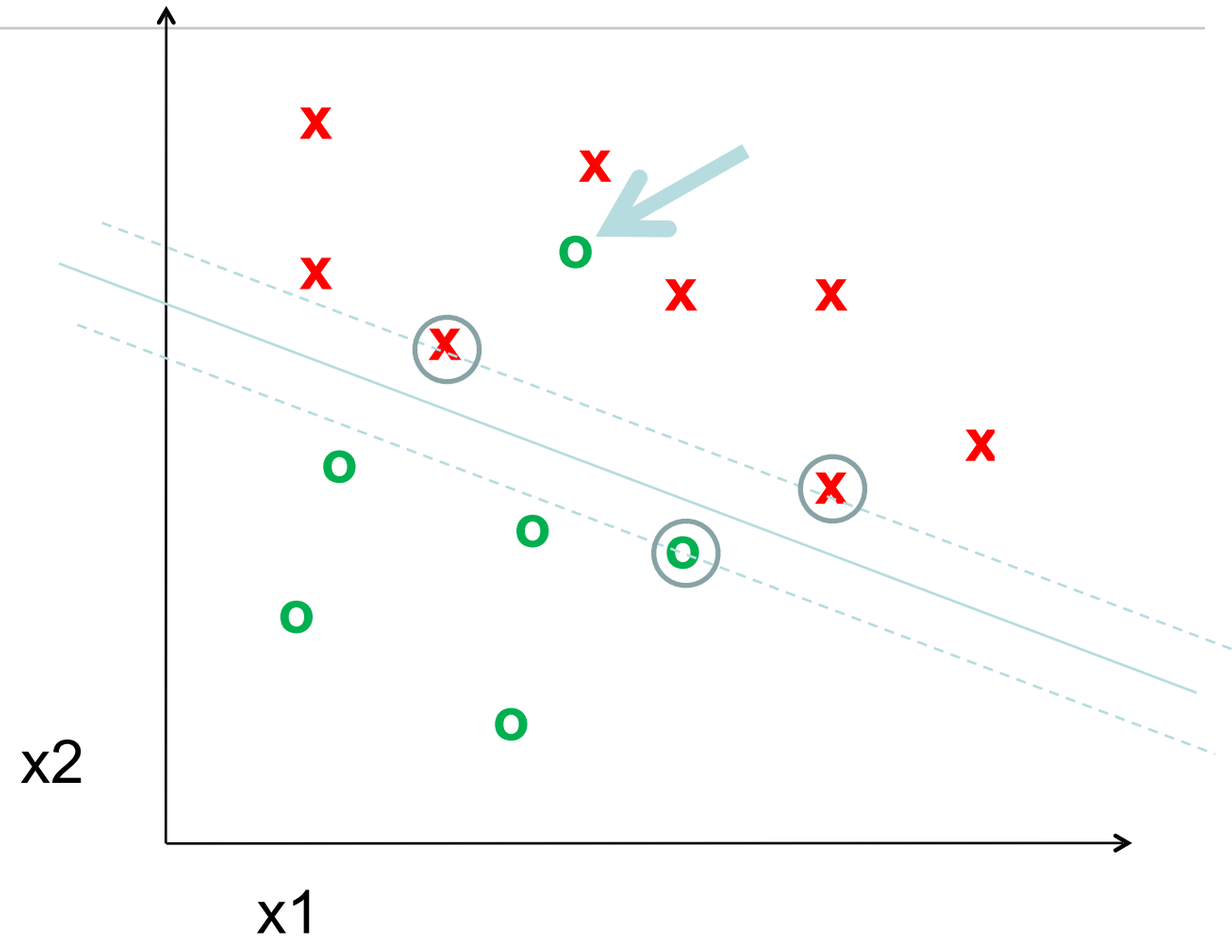
Define *hyperplane* **tX-b = 0**, where **t** is tangent to hyperplane.

Minimize **||t||** s.t. **tX-b** produces correct label for all **X**

Classifiers: Linear SVM

Find a *linear function* to separate the classes:

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

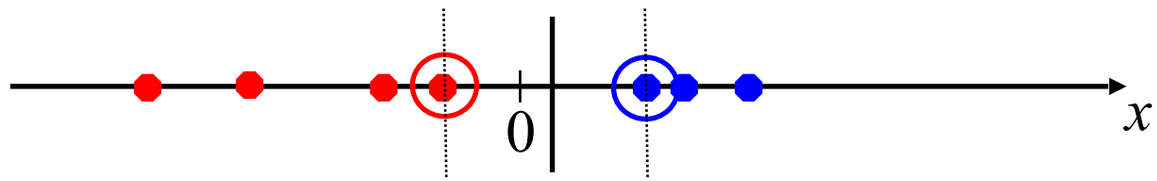


What if my data are not linearly separable?

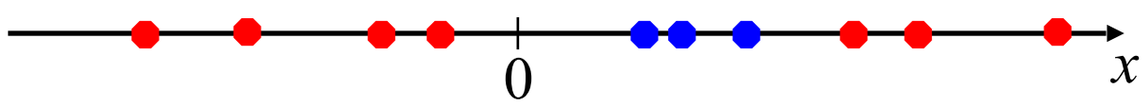
Introduce flexible ‘hinge’ loss (or ‘soft-margin’)

Nonlinear SVMs

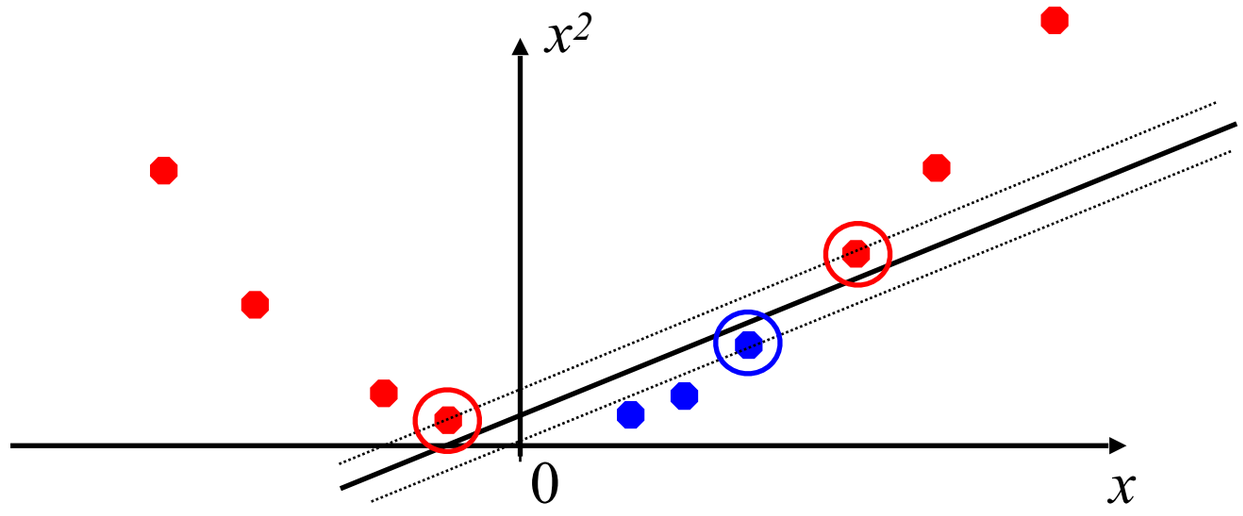
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



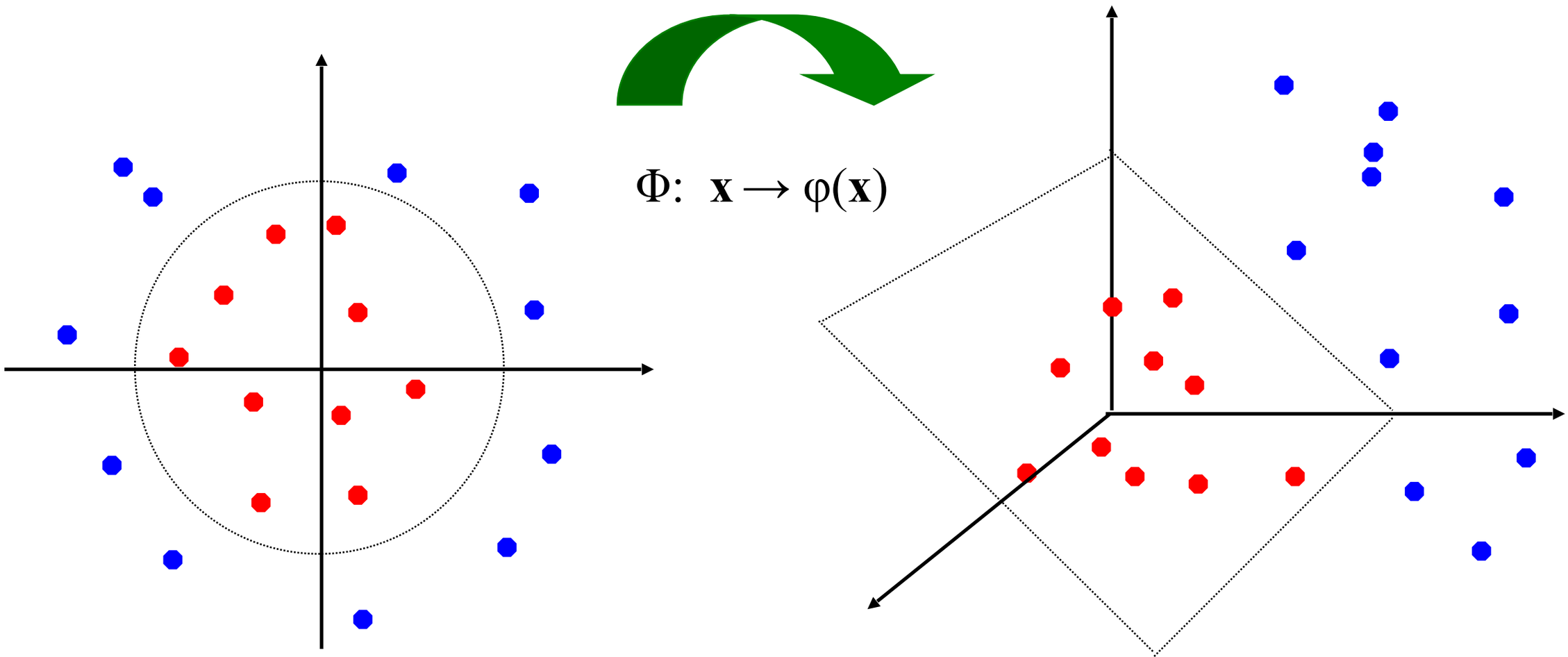
- We can map it to a higher-dimensional space:



Andrew Moore

Nonlinear SVMs

Map the original input space 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 $\varphi(\mathbf{x})$, define a kernel function K such that:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

This gives a *non-linear* decision boundary in the original feature space:

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

But…we only transformed the distance function *K!*

Common kernel function: Radial basis function kernel

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

What about multi-class SVMs?

Unfortunately, there is no "definitive" multi-class SVM.

In practice, we combine multiple two-class SVMs

One vs. others

- Training: learn an SVM for each class vs. the others
- 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

Slide credit: L. Lazebnik

SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages: http://www.kernel-machines.org/software
  - Kernel-based framework is very powerful, flexible
  - SVMs work very well in practice, even with very small training sample sizes

- Cons
  - No "direct" multi-class SVM, must combine two-class SVMs
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems

Ideals for a classification algorithm

- Objective function: encodes the right loss for the problem
- Parameterization: takes advantage of the structure of the problem
- Regularization: good priors on the parameters
- Training algorithm: can find parameters that maximize objective on training set
- Inference algorithm: can solve for labels that maximize objective function for a test example

Two ways to think about classifiers

1. What is the objective?
   What are the parameters?
   How are the parameters learned?
   How is the learning regularized?
   How is inference performed?

2. How is the data modeled?
   How is similarity defined?
   What is the shape of the boundary?

Slide credit: D. Hoiem

Training

Training Images

Training Labels

Image Features

Training

Learned classifier

Testing

Test Image

Image Features

Apply classifier

Prediction

Slide credit: D. Hoiem and L. Lazebnik

Features and **distance measures**

define visual similarity.

Training labels

dictate that examples are the same or different.

Classifiers

learn weights (or parameters) of features and distance measures…

so that visual similarity predicts label similarity.

Generative vs. Discriminative Classifiers

Discriminative Models

- Learn to directly predict the labels from the data
- Often, assume a simple boundary (e.g., linear)
- Examples
  - Logistic regression
  - SVM
  - Boosted decision trees

- Often easier to predict a label from the data than to model the data

Generative Models

- Represent both the data and the labels
- Often, makes use of conditional independence and priors
- Examples
  - Naïve Bayes classifier
  - Bayesian network

- Models of data may apply to future prediction problems

Slide credit: D. Hoiem

Making decisions about data

- 3 important design decisions:
  1) What data do I use?
  2) How do I represent my data (what feature)?
  3) What classifier / regressor / machine learning tool do I use?
- These are in decreasing order of importance
- Deep learning addresses 2 and 3 simultaneously (and blurs the boundary between them).
- You can take the representation from deep learning and use it with any classifier.

Many classifiers to choose from…

- K-nearest neighbor
- SVM
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- Restricted Boltzmann Machines
- Neural networks
- Deep Convolutional Network
- …

Claim:

It is more important to have more or better labeled data than to use a different supervised learning technique.

"The Unreasonable Effectiveness of Data" - Norvig

What to remember about classifiers

- No free lunch: machine learning algorithms are tools, not dogmas
- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)

Slide credit: D. Hoiem