

## COMPUTER VISION LECTURE 12 – INTRO TO OBJECT RECOGNITION

Prof. Dr. Francesco Maurelli  
2018-10-12

# What we will learn today?

- Introduction to object recognition
- K-nearest neighbor algorithm
- A simple Object Recognition pipeline

# What are the different visual recognition tasks?





# Classification:

Does this image contain a building? [yes/no]



**Yes!**



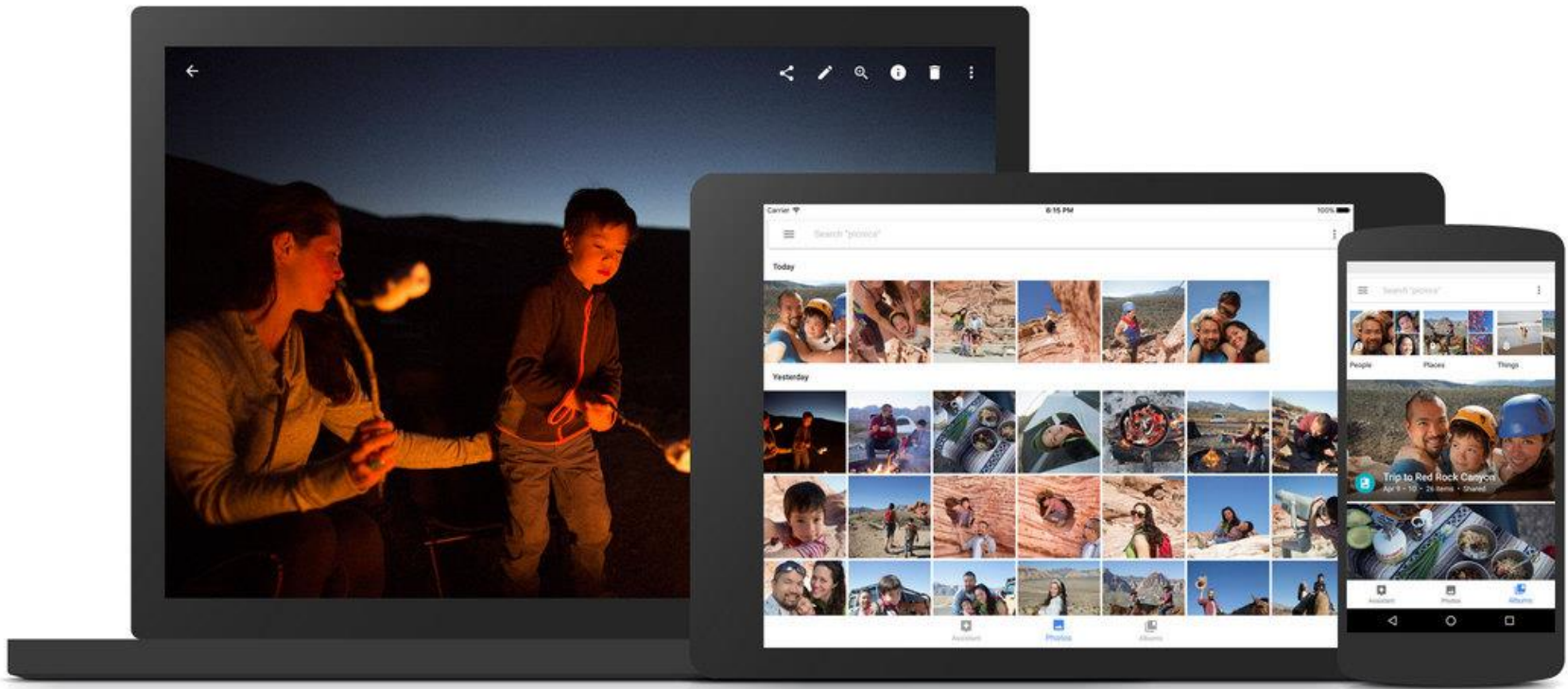
# Classification:

Is this a beach?



# Image Search

## Organizing photo collections





# Detection:

Does this image contain a car? [where?]



# Detection:

Which object does this image contain? [where?]

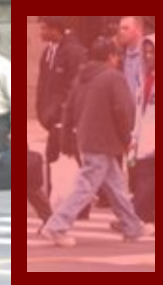
Building



clock



person



car



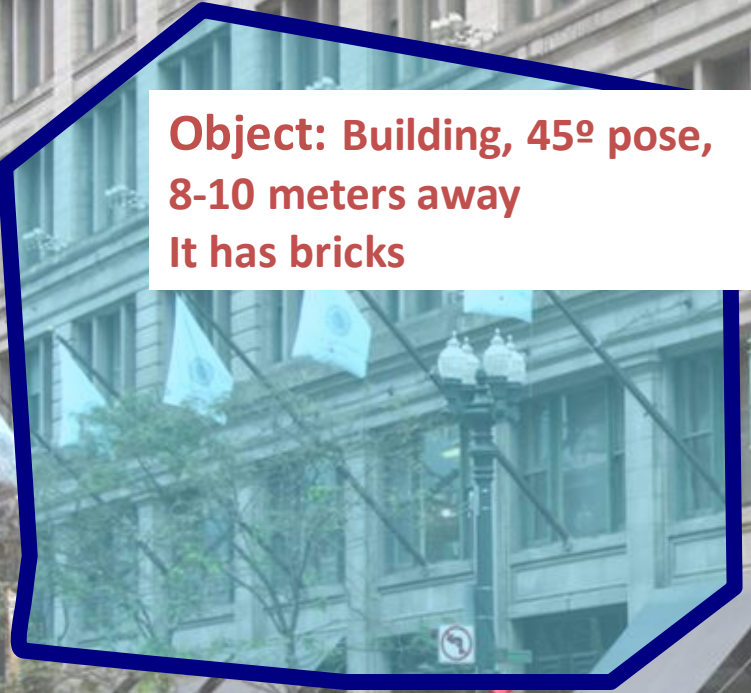


# Detection:

Accurate localization (segmentation)



# Detection: Estimating object semantic & geometric attributes



**Object: Building, 45° pose,  
8-10 meters away  
It has bricks**



**Object: Person, back;  
1-2 meters away**

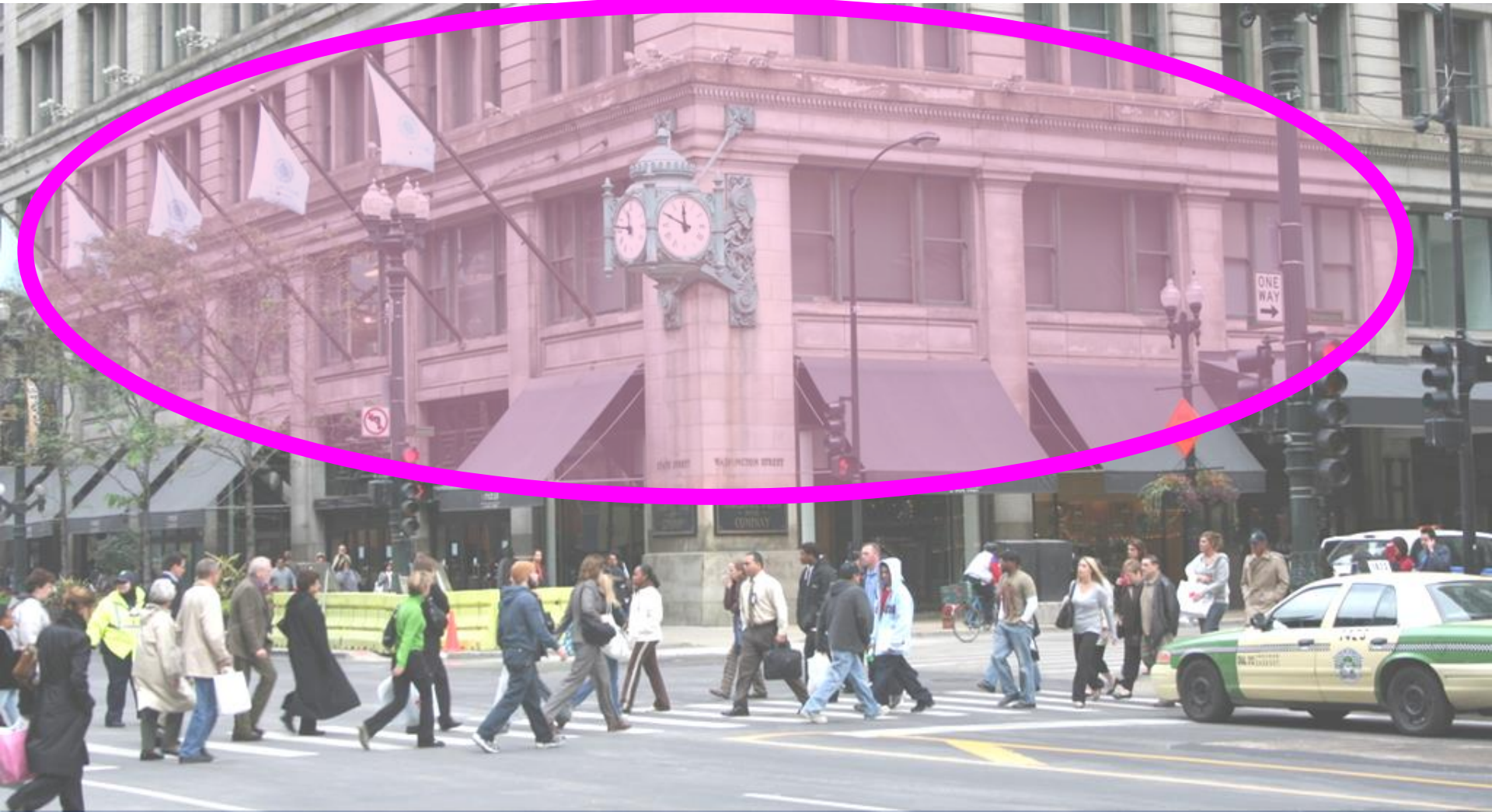


**Object: Police car, side view, 4-5 m away**



# Categorization vs Single instance recognition

Does this image contain the Chicago Macy's building?



# Categorization vs Single instance recognition

Where is the crunchy nut?

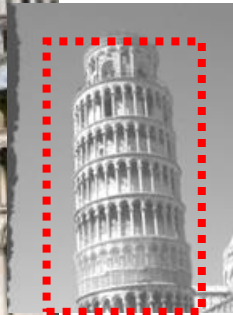




# Applications of computer vision



- Recognizing landmarks in mobile platforms



+ GPS

# Activity or Event recognition

What are these people doing?





# Visual Recognition

- Design algorithms that have the capability to:
  - Classify images or videos
  - Detect and localize objects
  - Estimate semantic and geometrical attributes
  - Classify human activities and events

## Why is this challenging?

How many object categories are there?

~10,000 to 30,000

**~10,000 to 30,000**





# Challenges: viewpoint variation



Michelangelo 1475-1564

# Challenges: illumination

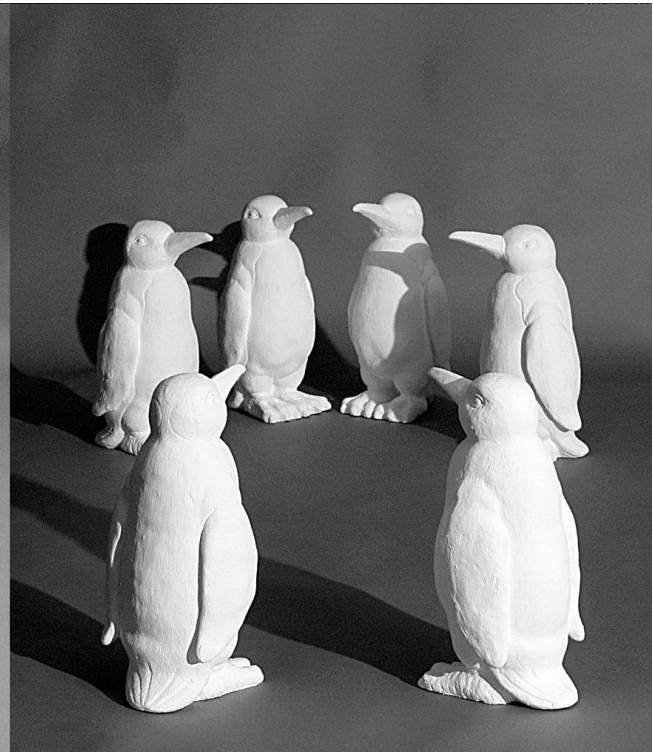
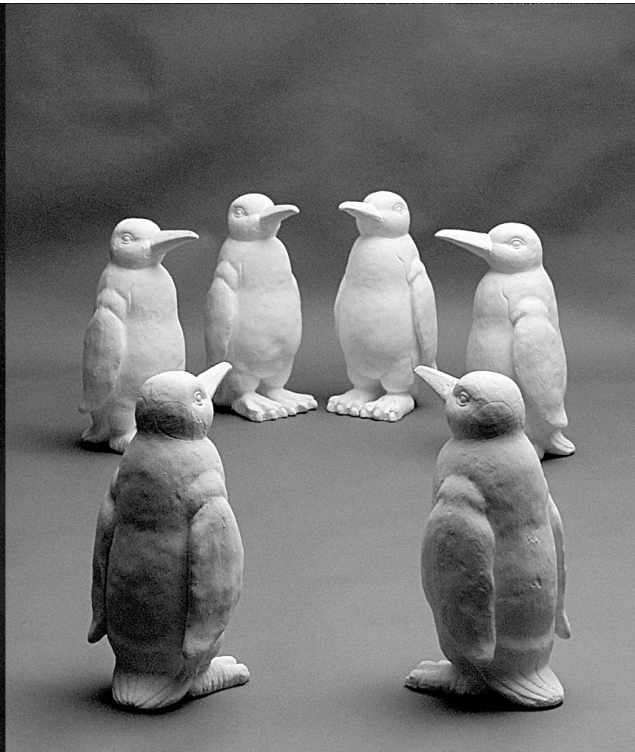


image credit: J. Koenderink



# Challenges: scale



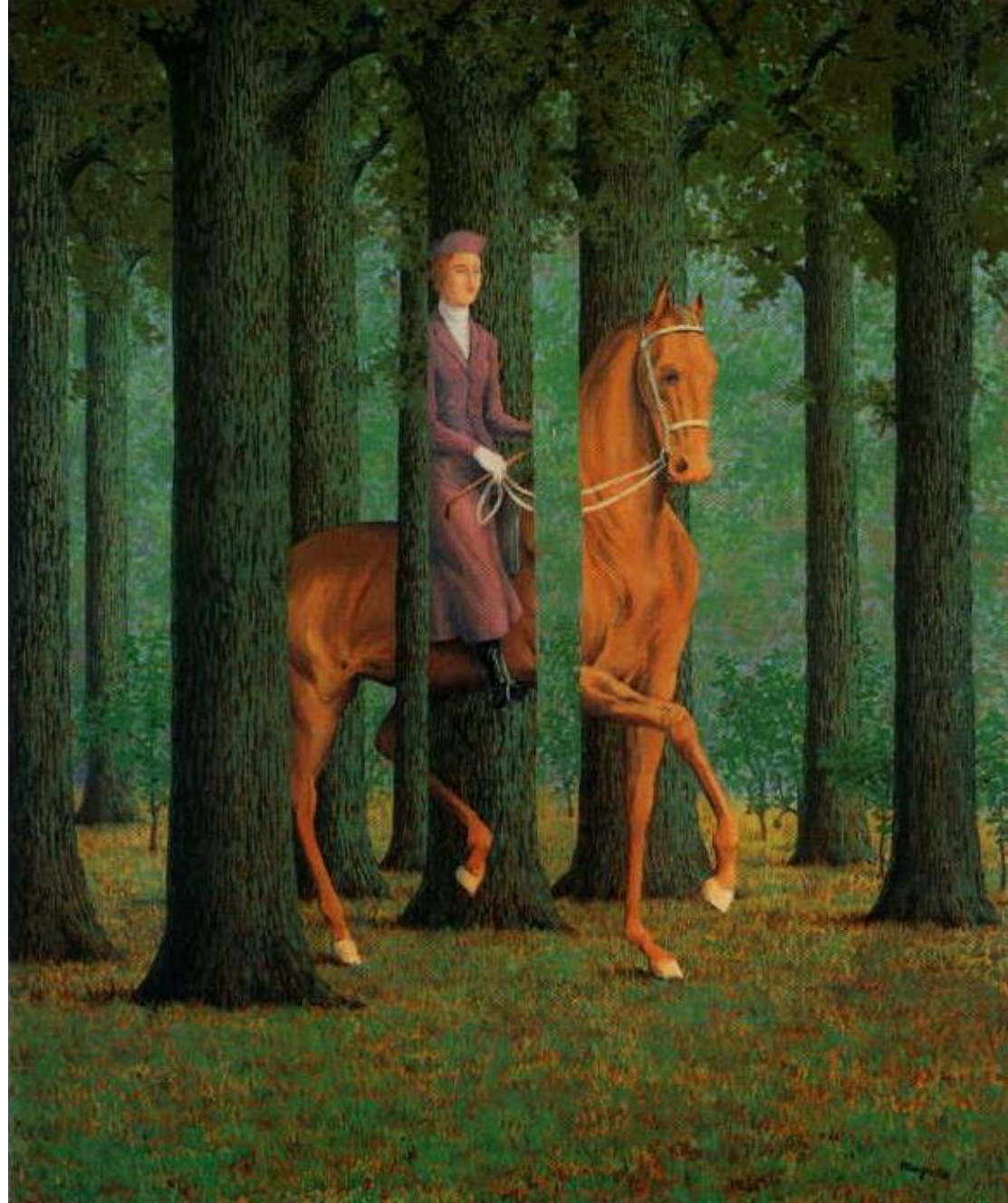
# Challenges: deformation





# Challenges: occlusion

Magritte, 1957



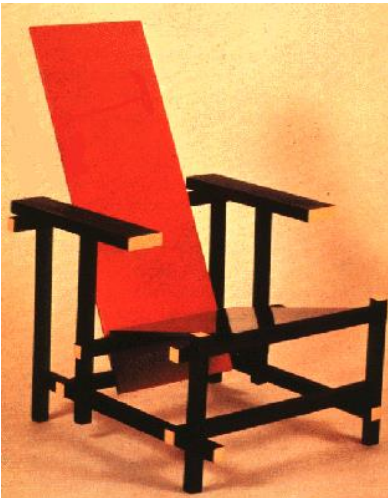
# Challenges: background clutter



Kilmeny Niland. 1995



# Challenges: intra-class variation



# What we will learn today?

- Introduction
- K-nearest neighbor algorithm
- A simple Object Recognition pipeline



# The machine learning framework

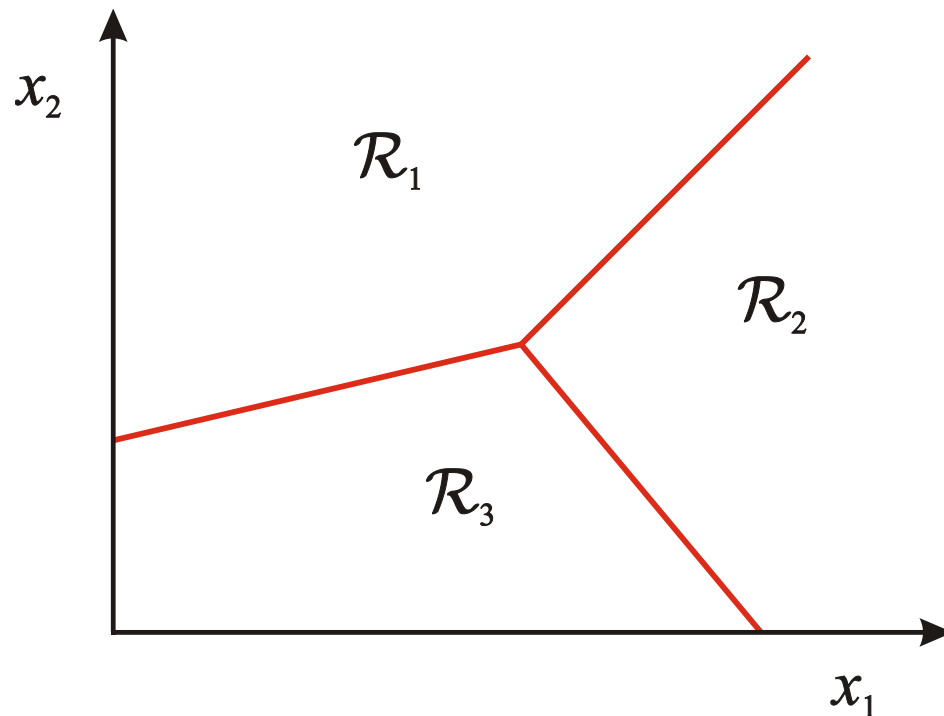
$$y = f(\mathbf{x})$$

output      prediction function      Image feature

- **Training:** given a *training* set of labeled examples  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ , 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  $y = f(\mathbf{x})$

# Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*

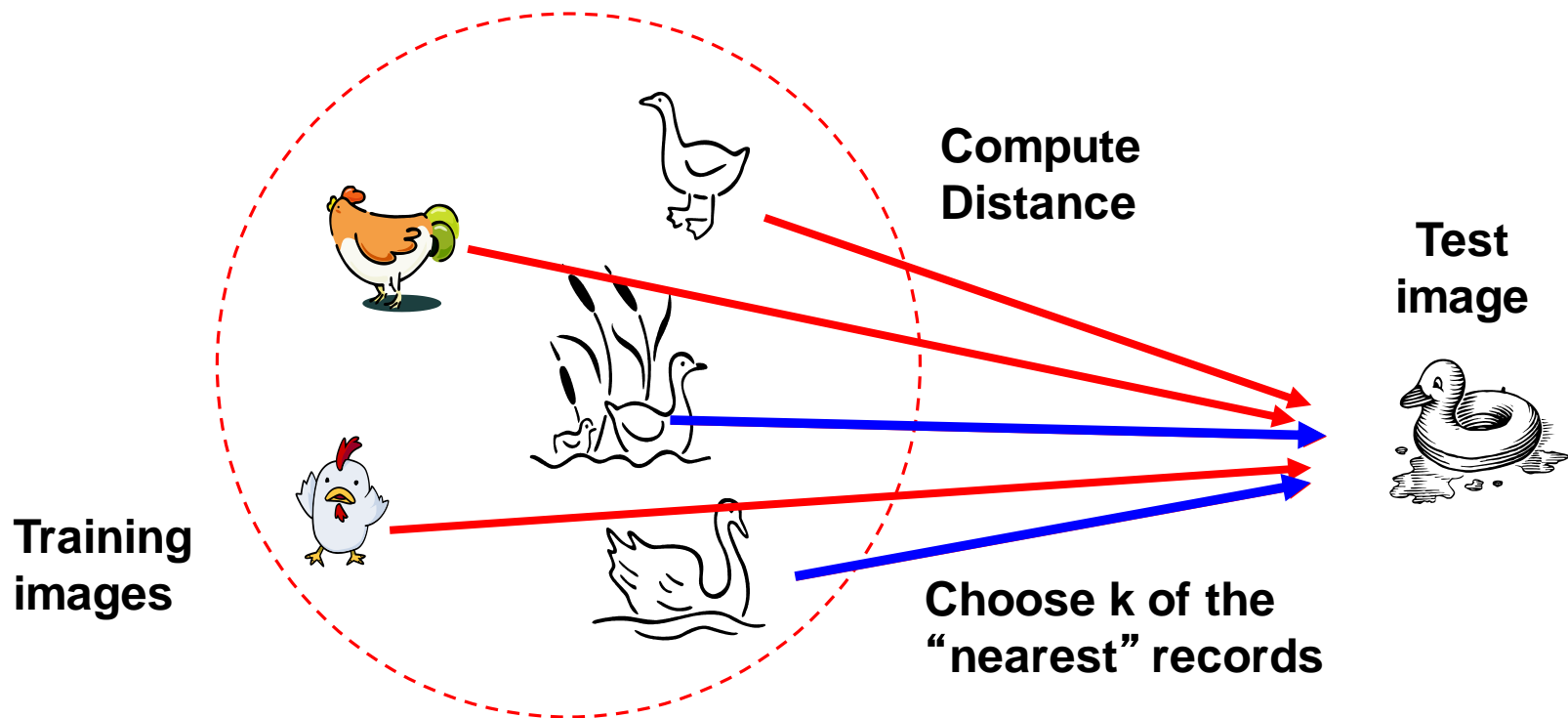


Slide credit: L. Lazebnik



# Nearest Neighbor Classifier

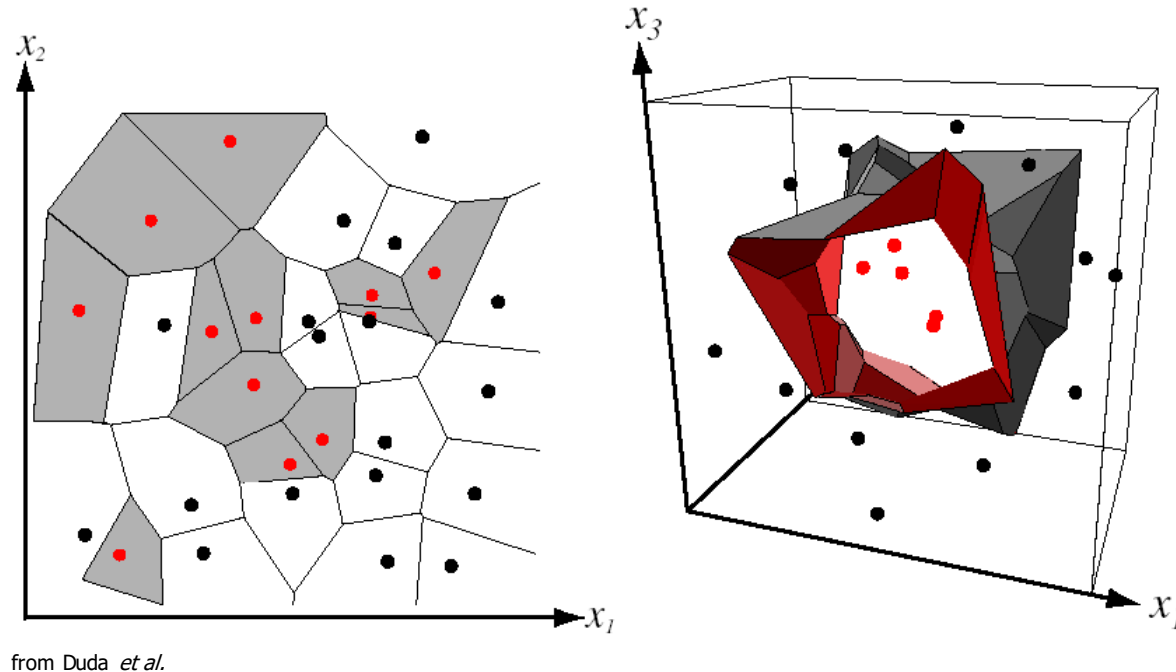
- Assign label of nearest training data point to each test data point



Source: N. Goyal

# Nearest Neighbor Classifier

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



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

Source: D. Lowe

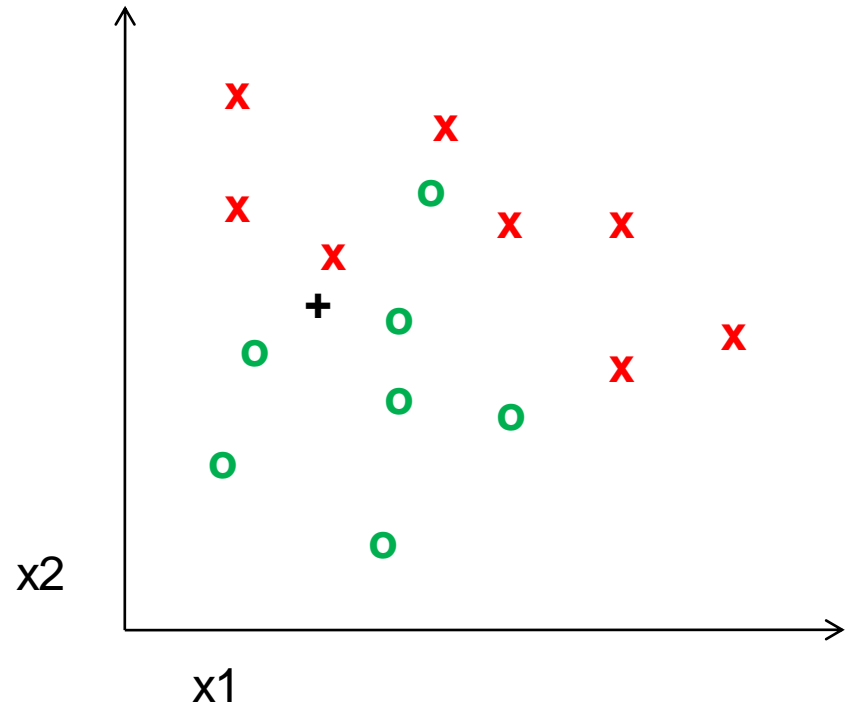


# K-nearest neighbor

Distance measure - Euclidean

$$Dist(X^n, X^m) = \sqrt{\sum_{i=1}^D (X_i^n - X_i^m)^2}$$

Where  $X^n$  and  $X^m$  are the n-th and m-th data points

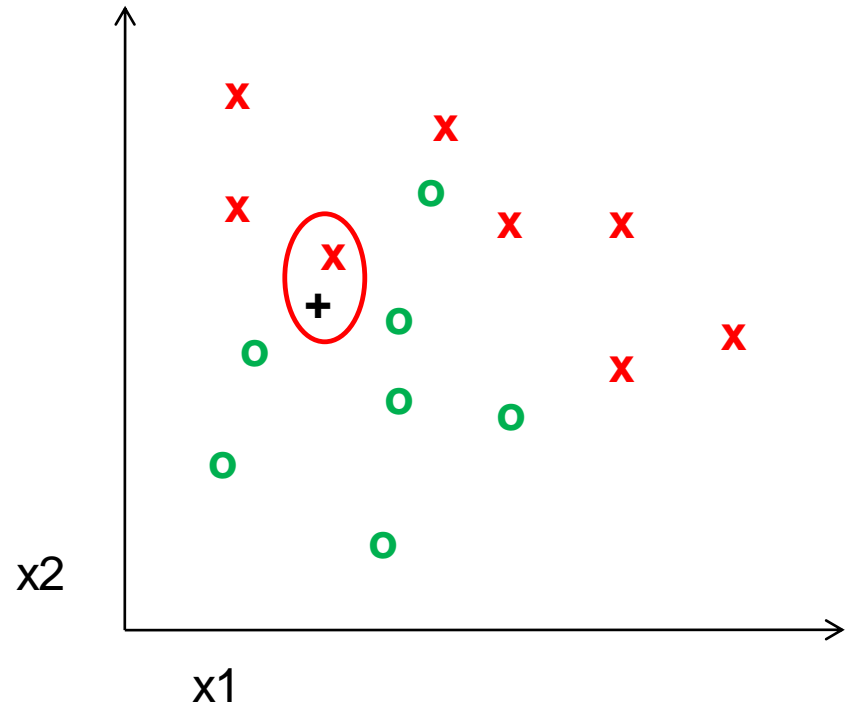


# 1-nearest neighbor

Distance measure - Euclidean

$$Dist(X^n, X^m) = \sqrt{\sum_{i=1}^D (X_i^n - X_i^m)^2}$$

Where  $X^n$  and  $X^m$  are the n-th and m-th data points

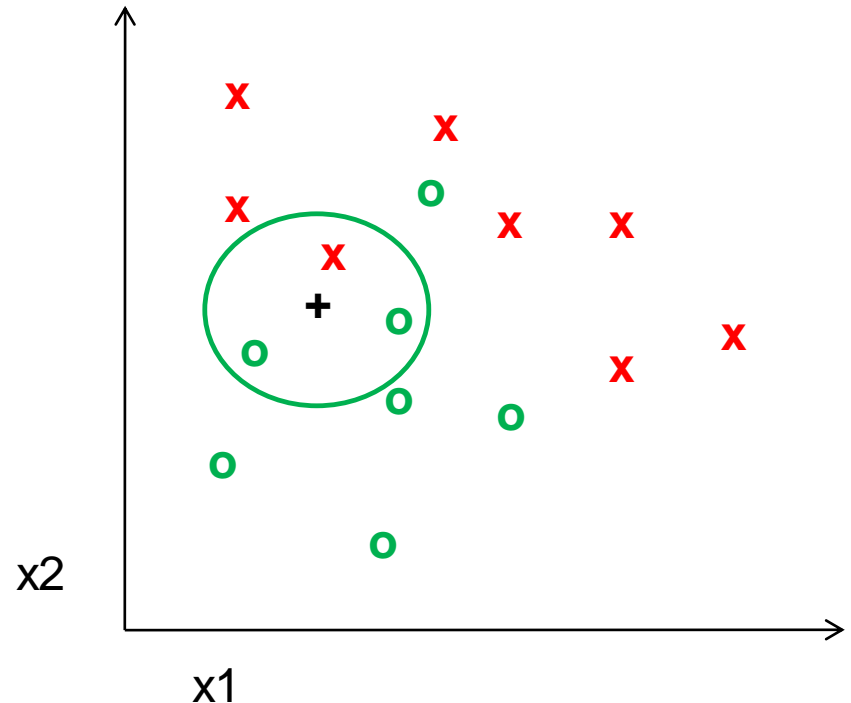


# 3-nearest neighbor

Distance measure - Euclidean

$$Dist(X^n, X^m) = \sqrt{\sum_{i=1}^D (X_i^n - X_i^m)^2}$$

Where  $X^n$  and  $X^m$  are the n-th and m-th data points



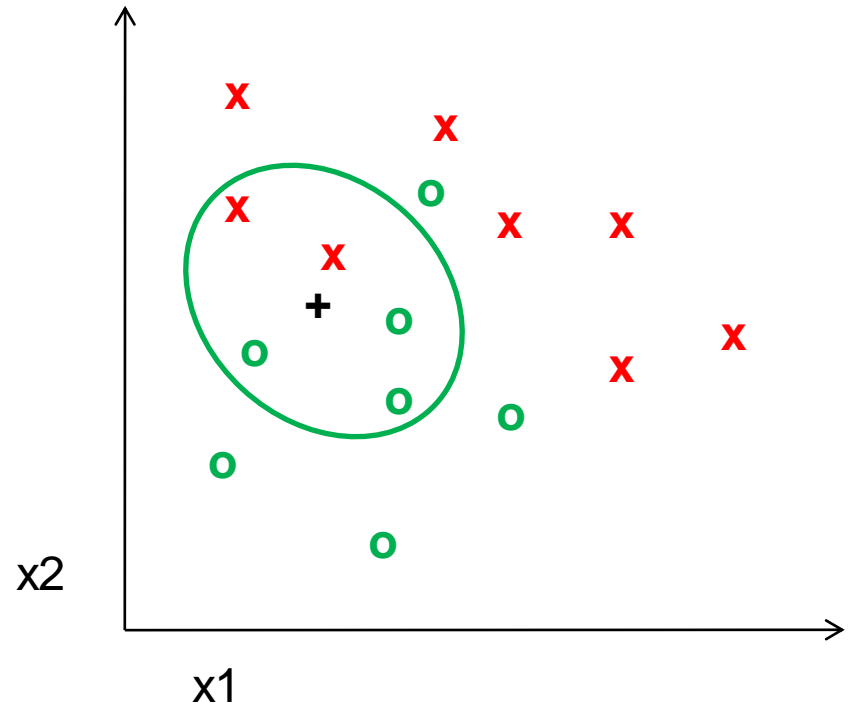


# 5-nearest neighbor

Distance measure - Euclidean

$$Dist(X^n, X^m) = \sqrt{\sum_{i=1}^D (X_i^n - X_i^m)^2}$$

Where  $X^n$  and  $X^m$  are the n-th and m-th data points

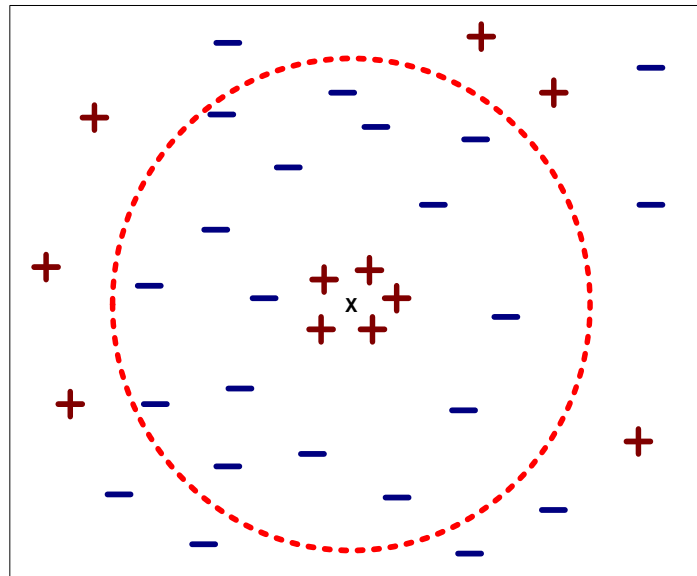


# K-NN: a very useful algorithm

- Simple, a good one to try first
- Very flexible decision boundaries

# K-NN: issues to keep in mind

- Choosing the value of k:
  - If too small, sensitive to noise points
  - If too large, neighborhood may include points from other classes

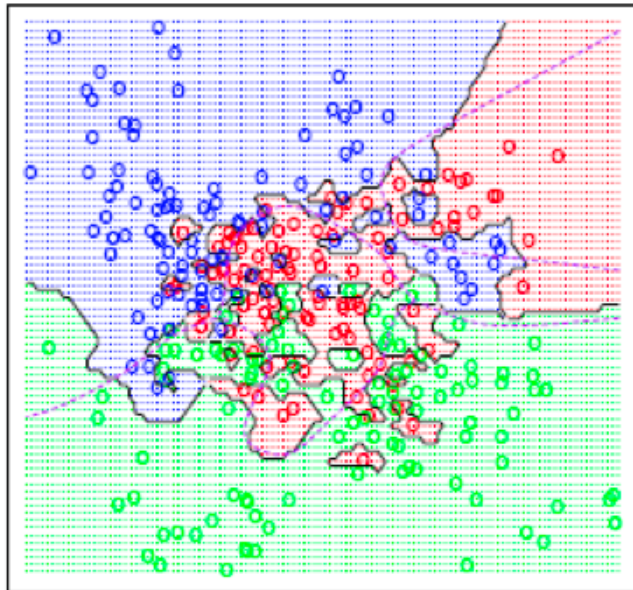




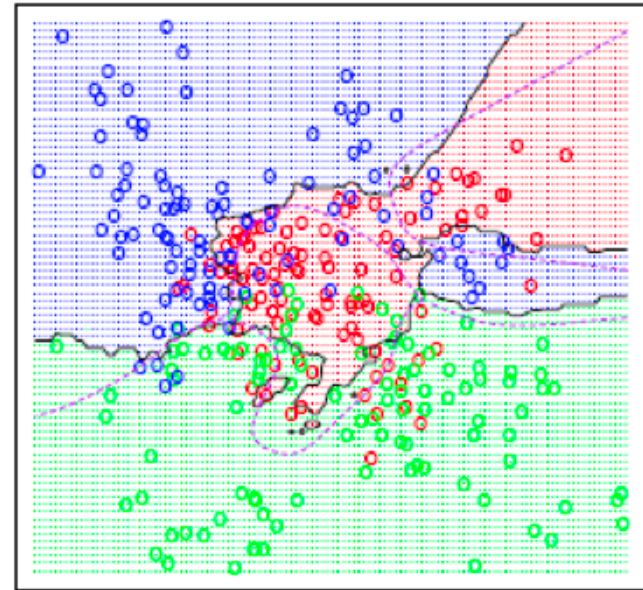
# K-NN: issues to keep in mind

- Choosing the value of  $k$ :
  - If too small, sensitive to noise points
  - If too large, neighborhood may include points from other classes

$K=1$

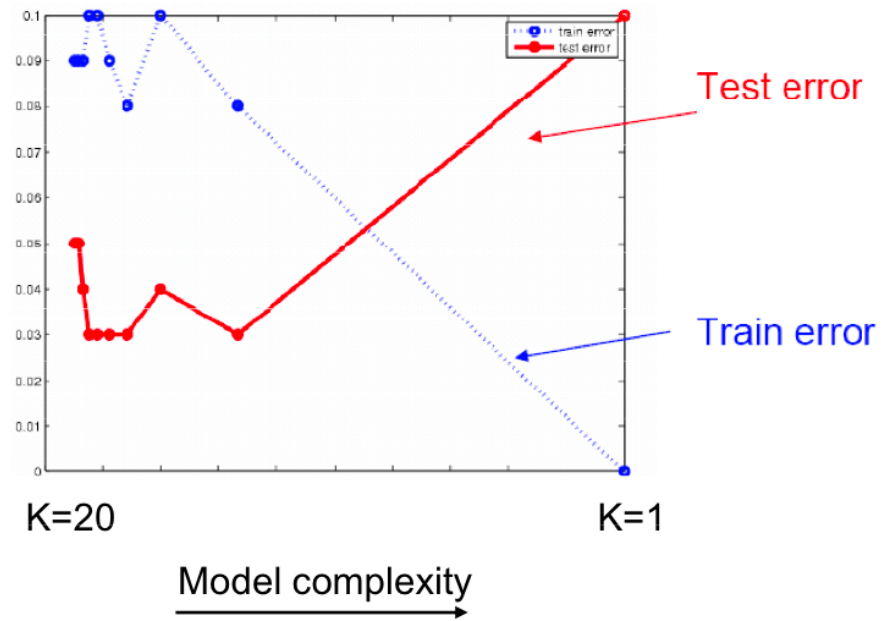


$K=15$

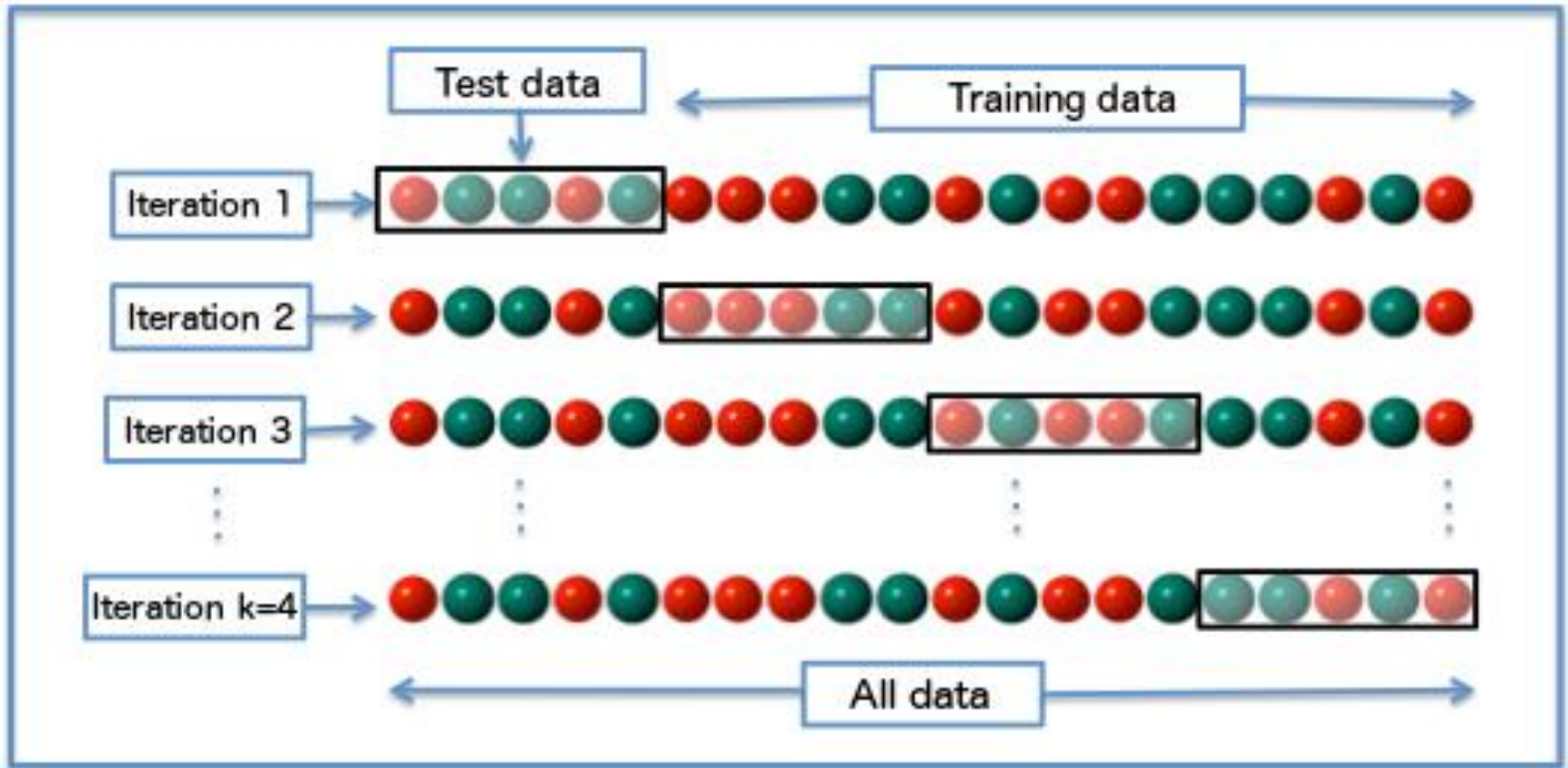


# K-NN: issues to keep in mind

- Choosing the value of  $k$ :
  - If too small, sensitive to noise points
  - If too large, neighborhood may include points from other classes
  - **Solution**: cross validate!



# Cross validation





# K-NN: issues to keep in mind

- Choosing the value of  $k$ :
  - If too small, sensitive to noise points
  - If too large, neighborhood may include points from other classes
  - **Solution**: cross validate!
- Can produce counter-intuitive results (using Euclidean measure)

# Euclidean measure

1 1 1 1 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 1 1 1 1

$d = 1.4142$

vs

1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1

$d = 1.4142$

# K-NN: issues to keep in mind

- Choosing the value of  $k$ :
  - If too small, sensitive to noise points
  - If too large, neighborhood may include points from other classes
  - **Solution**: cross validate!
- Can produce counter-intuitive results (using Euclidean measure)
  - **Solution**: normalize the vectors to unit length



# Many classifiers to choose from

- **K-nearest neighbor**
- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- RBMs
- Etc.

Which is the best one?

Slide credit: D. Hoiem

# Generalization



Training set (labels known)

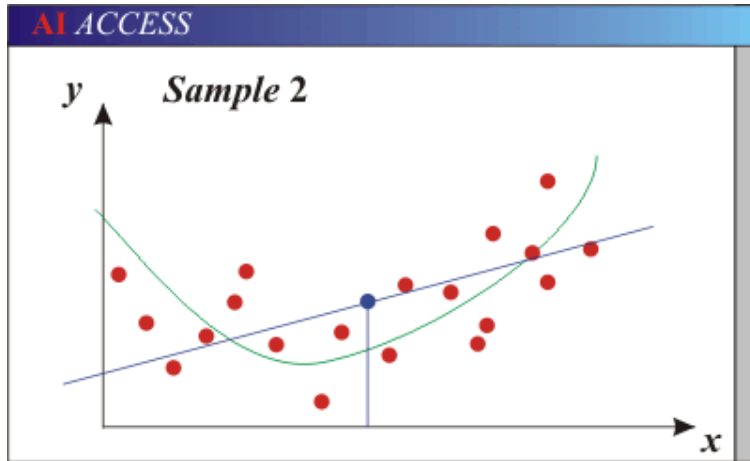


Test set (labels unknown)

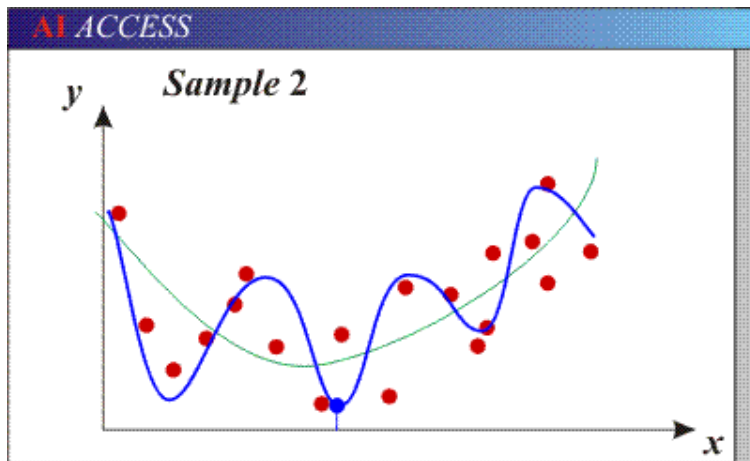
- How well does a learned model generalize from the data it was trained on to a new test set?

Slide credit: L. Lazebnik

# Bias-Variance Trade-off



- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).



- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Slide credit: D. Hoiem

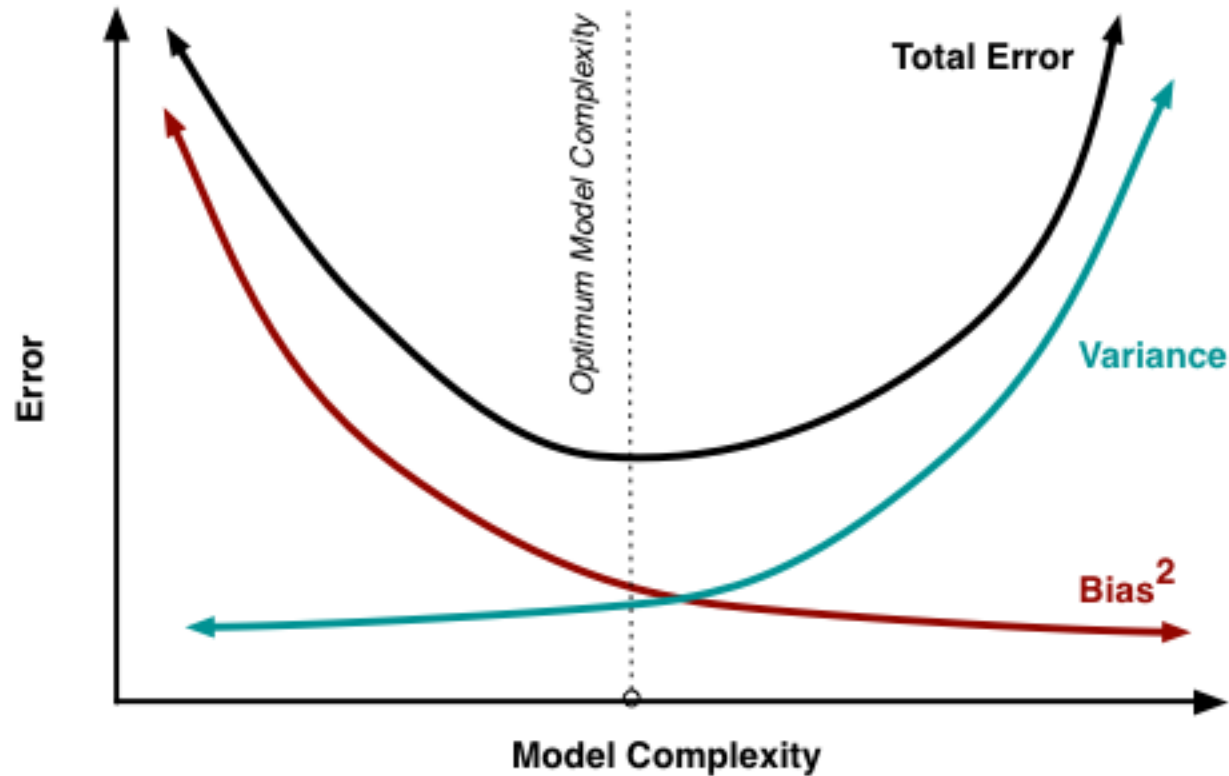
# Bias versus variance

- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

Slide credit: L. Lazebnik



# Bias versus variance trade off



# No Free Lunch Theorem



In a supervised learning setting, we can't tell which classifier will have best generalization

Slide credit: D. Hoiem

# Remember...

- No classifier is inherently better than any other: you need to make assumptions to generalize
- Three kinds of error
  - Inherent: unavoidable
  - Bias: due to over-simplifications
  - Variance: due to inability to perfectly estimate parameters from limited data



Slide credit: D. Hoiem

# How to reduce variance?

- Choose a simpler classifier
- Regularize the parameters
- Get more training data

Slide credit: D. Hoiem



# Last remarks about applying machine learning methods to object recognition

- There are machine learning algorithms to choose from
- Know your data:
  - How much supervision do you have?
  - How many training examples can you afford?
  - How noisy?
- Know your goal (i.e. task):
  - Affects your choices of representation
  - Affects your choices of learning algorithms
  - Affects your choices of evaluation metrics
- Understand the math behind each machine learning algorithm under consideration!

# What we will learn today?

- Introduction
- K-nearest neighbor algorithm
- A simple Object Recognition pipeline

# Object recognition: a classification framework

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

$$f(\text{apple image}) = \text{"apple"}$$

$$f(\text{tomato image}) = \text{"tomato"}$$

$$f(\text{cow image}) = \text{"cow"}$$

# A simple pipeline - Training

Training  
Images

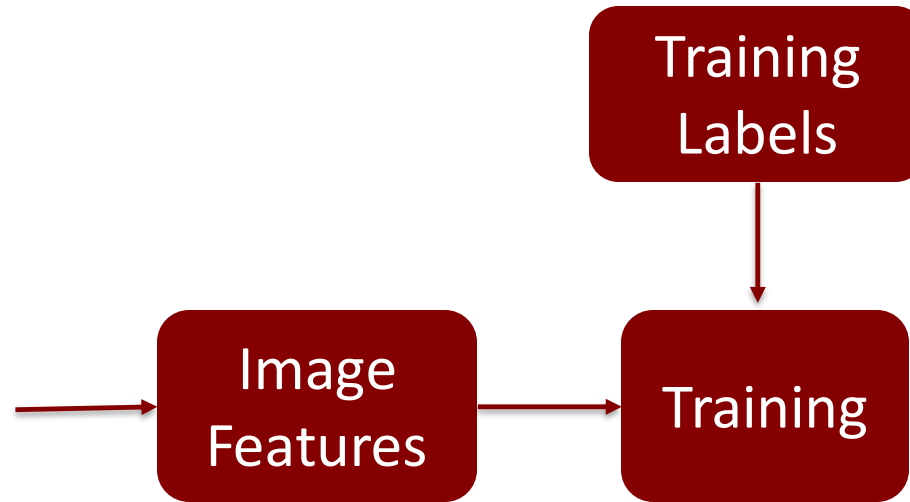


Image  
Features



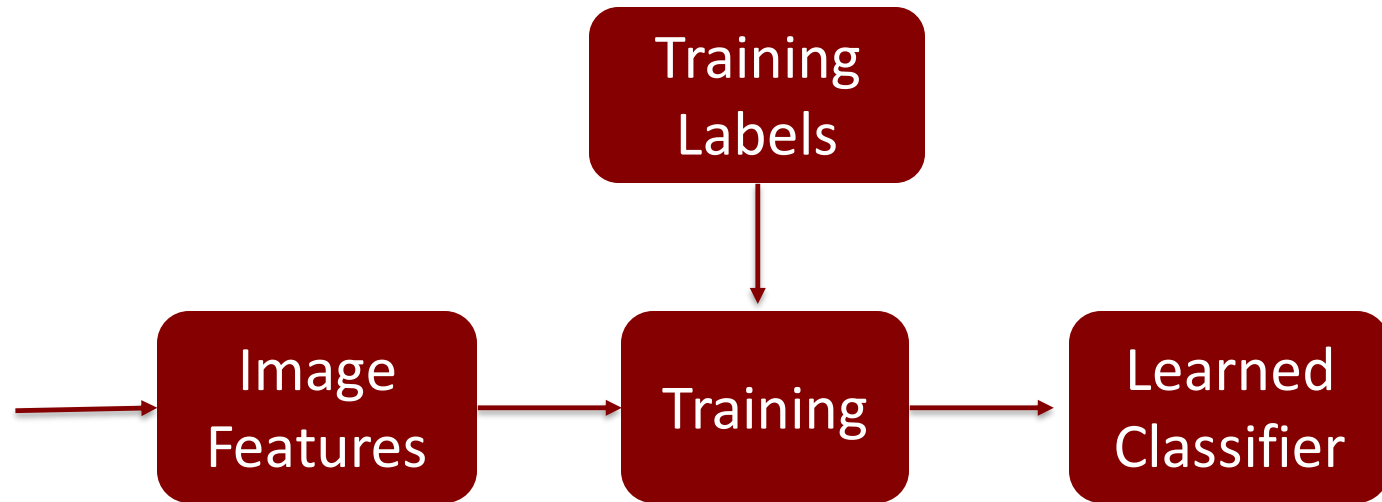
# A simple pipeline - Training

Training  
Images



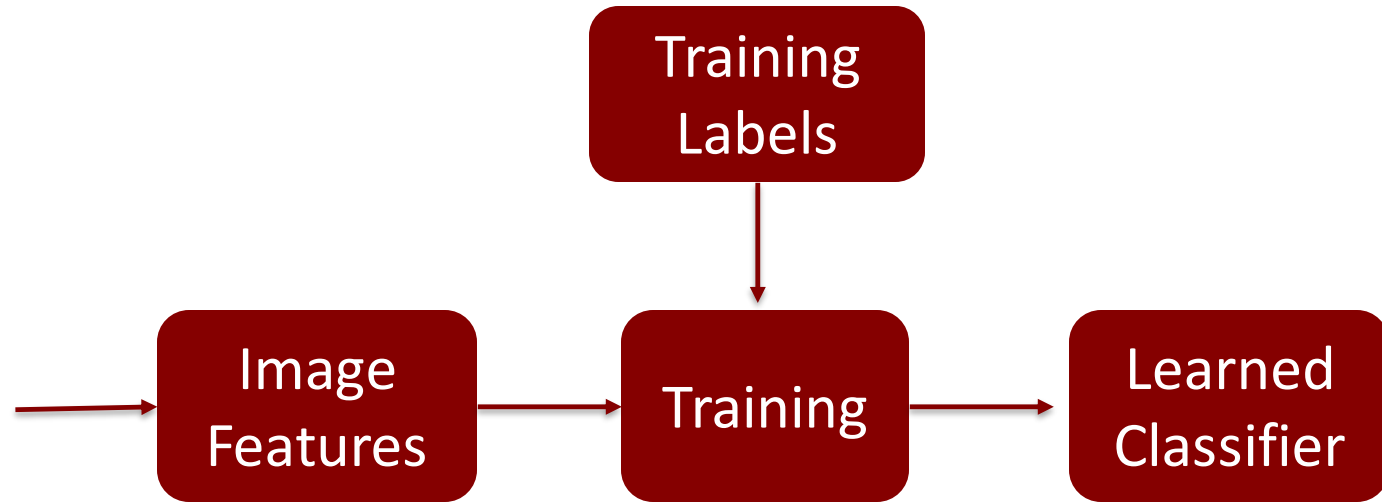
# A simple pipeline - Training

Training  
Images



# A simple pipeline - Training

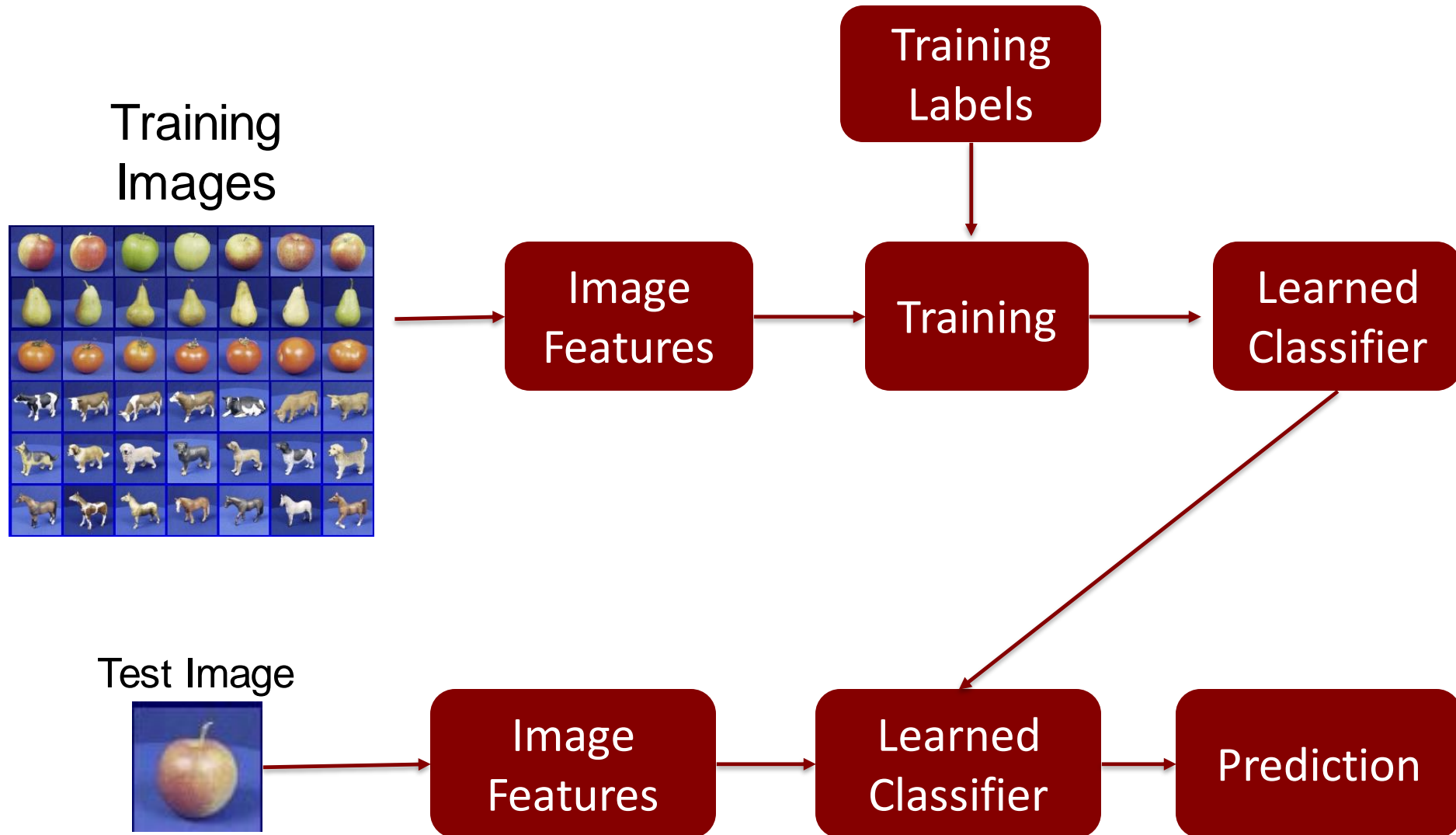
Training  
Images



Test Image

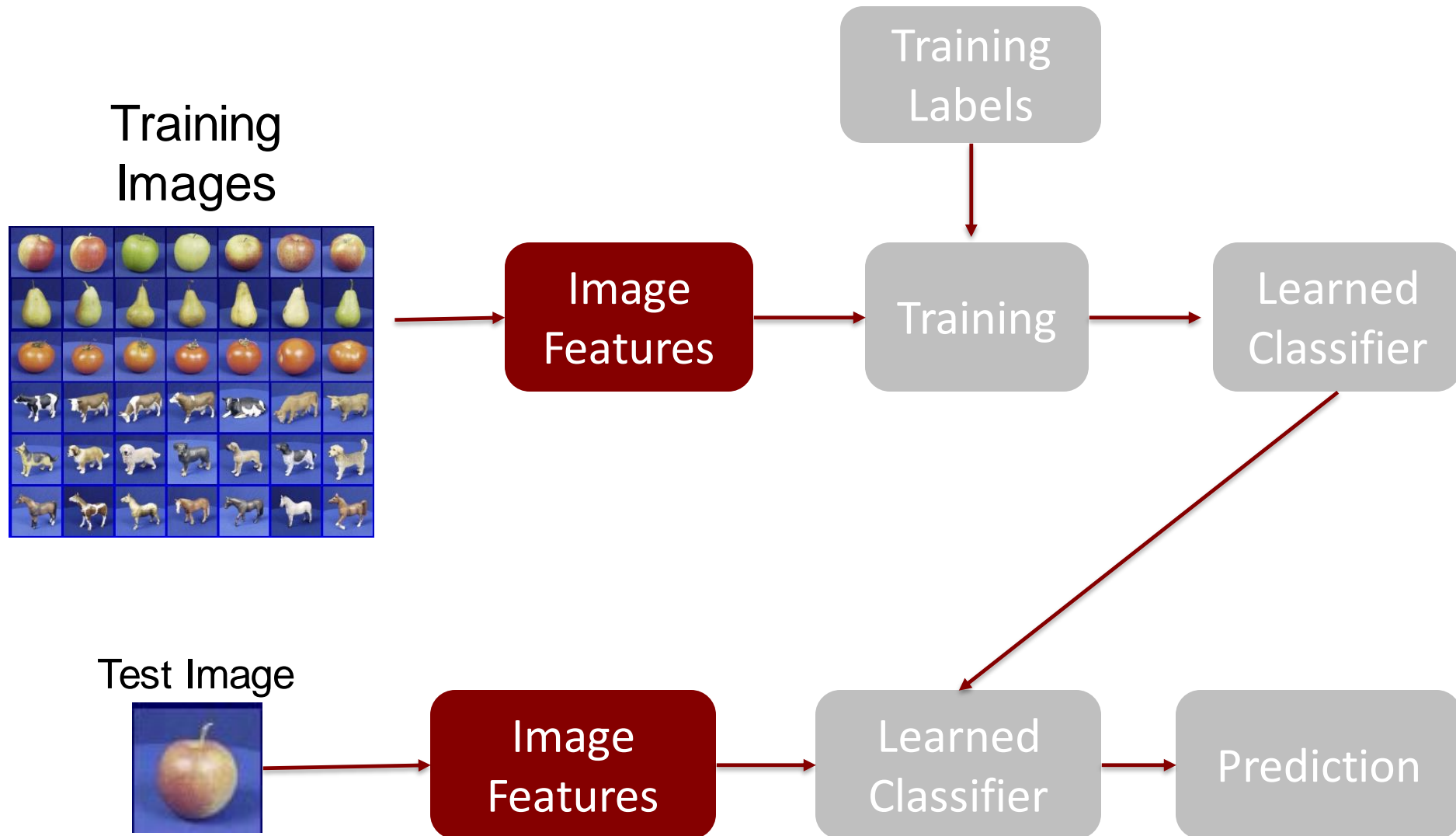


# A simple pipeline - Training





# A simple pipeline - Training

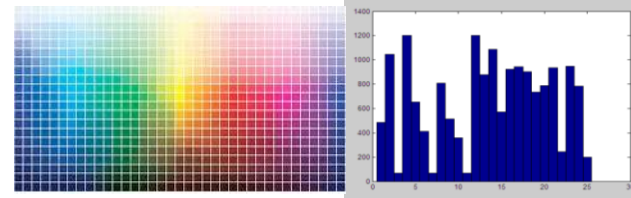


# Image features

Input image



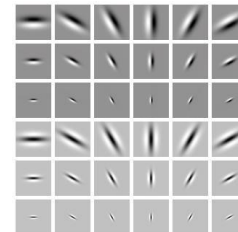
Color: Quantize RGB values



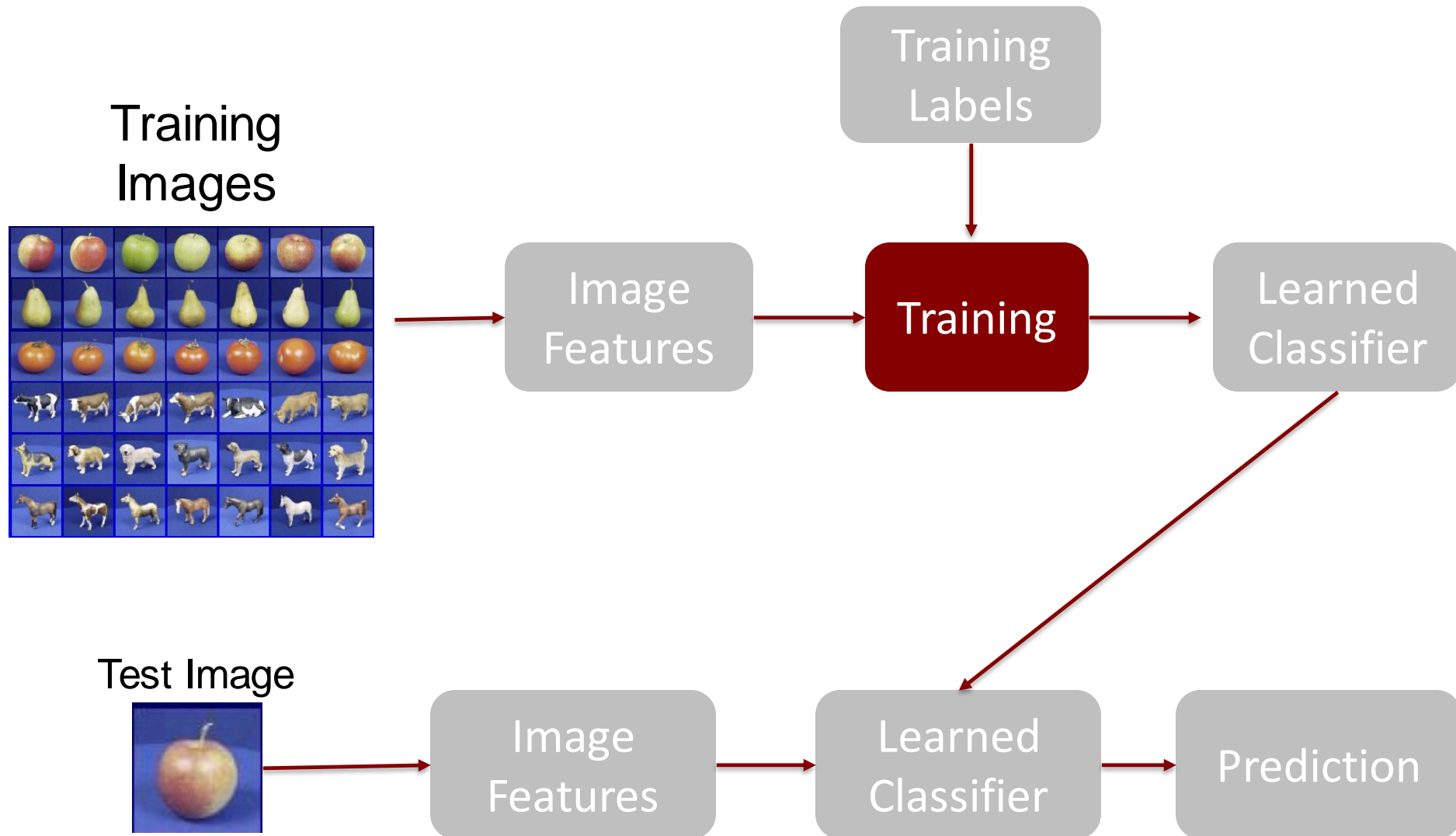
Global shape: PCA



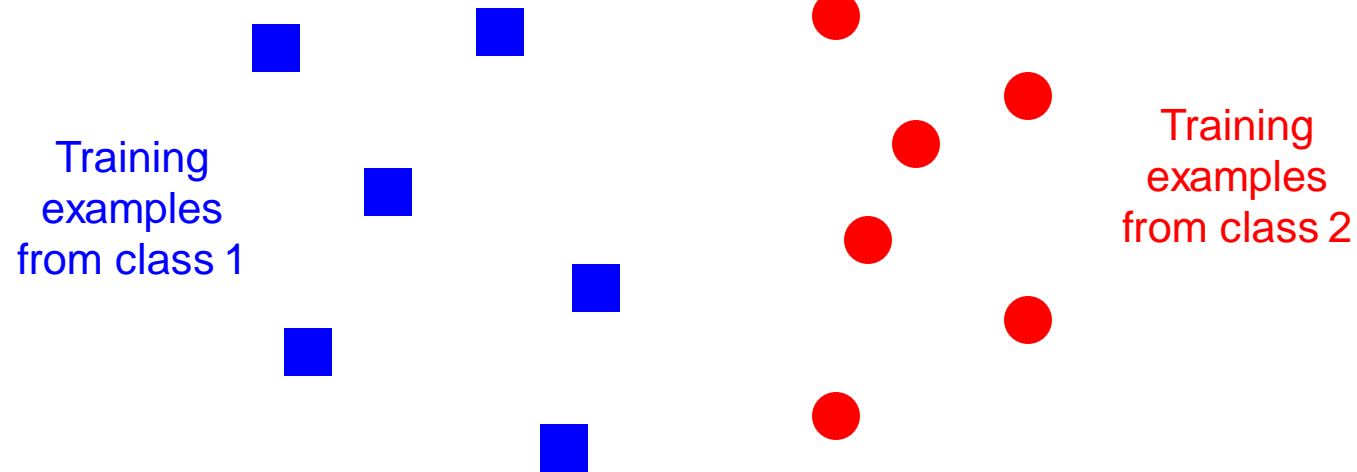
Texture: Filter banks



# A simple pipeline - Training



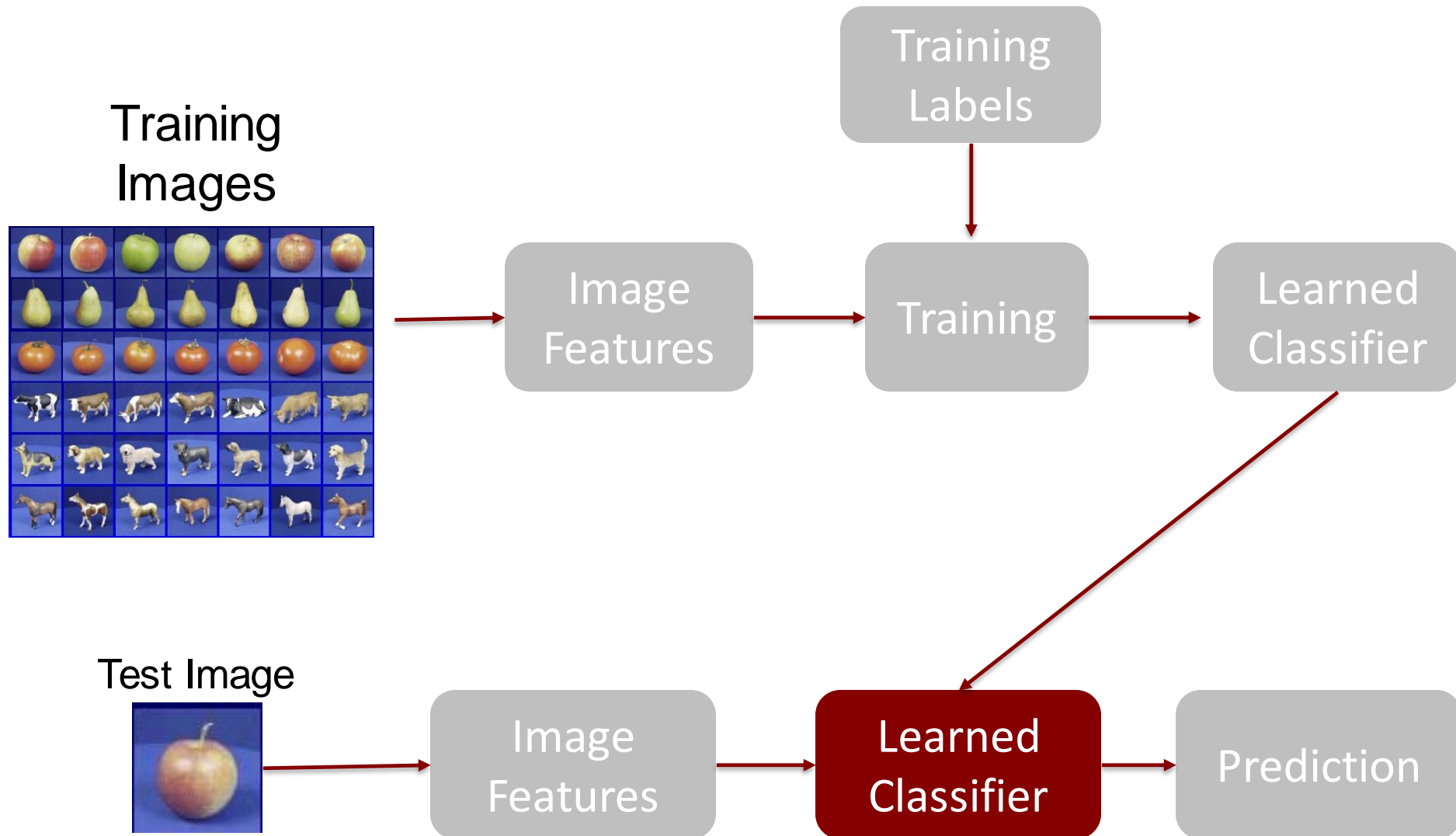
# Classifiers: Nearest neighbor



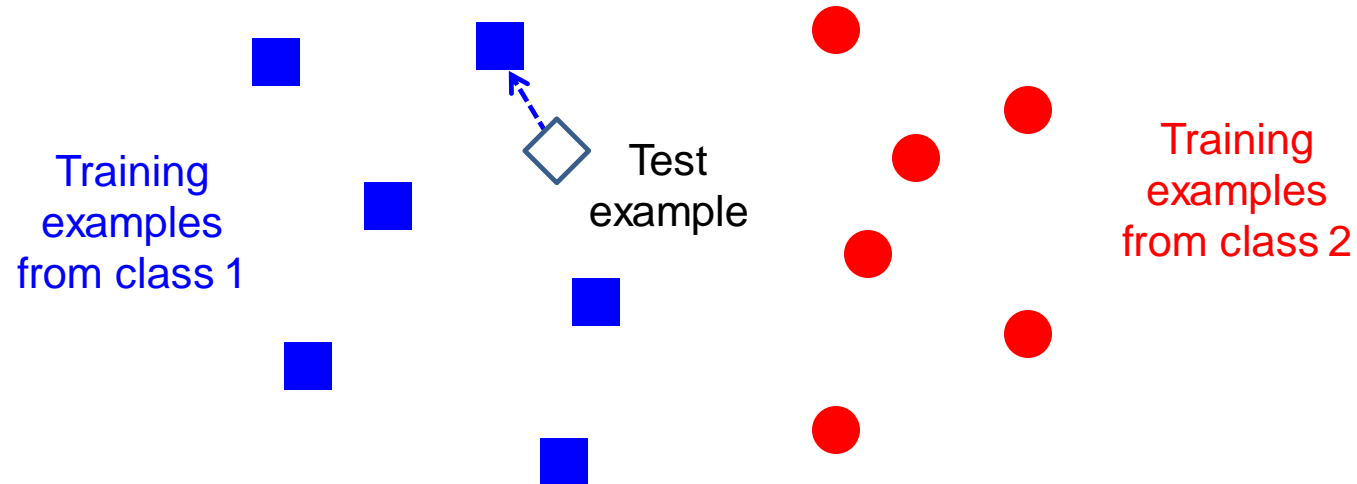
Slide credit: L. Lazebnik



# A simple pipeline - Training



# Classifiers: Nearest neighbor



Slide credit: L. Lazebnik

# Results



	Color	$D_x D_y$	Mag-Lap	PCA Masks	PCA Gray	Cont. Greedy	Cont. DynProg	Avg.
apple	57.56%	<b>85.37%</b>	80.24%	78.78%	<b>88.29%</b>	77.07%	76.34%	77.66%
pear	66.10%	90.00%	85.37%	<b>99.51%</b>	<b>99.76%</b>	90.73%	91.71%	89.03%
tomato	<b>98.54%</b>	94.63%	<b>97.07%</b>	67.80%	76.59%	70.73%	70.24%	82.23%
cow	86.59%	82.68%	<b>94.39%</b>	75.12%	62.44%	86.83%	86.34%	82.06%
dog	34.63%	62.44%	74.39%	72.20%	66.34%	<b>81.95%</b>	<b>82.93%</b>	67.84%
horse	32.68%	58.78%	70.98%	77.80%	77.32%	<b>84.63%</b>	<b>84.63%</b>	69.55%
cup	79.76%	66.10%	77.80%	<b>96.10%</b>	<b>96.10%</b>	<b>99.76%</b>	<b>99.02%</b>	87.81%
car	62.93%	<b>98.29%</b>	77.56%	<b>100.0%</b>	<b>97.07%</b>	<b>99.51%</b>	<b>100.0%</b>	90.77%
total	64.85%	79.79%	82.23%	83.41%	82.99%	86.40%	86.40%	80.87%

Dataset: ETH-80, by B. Leibe, 2003

# What we have learned today?

- Introduction
- K-nearest neighbor algorithm
- A simple Object Recognition pipeline