

EBU7240

Computer Vision

- Machine learning basics and recognition -

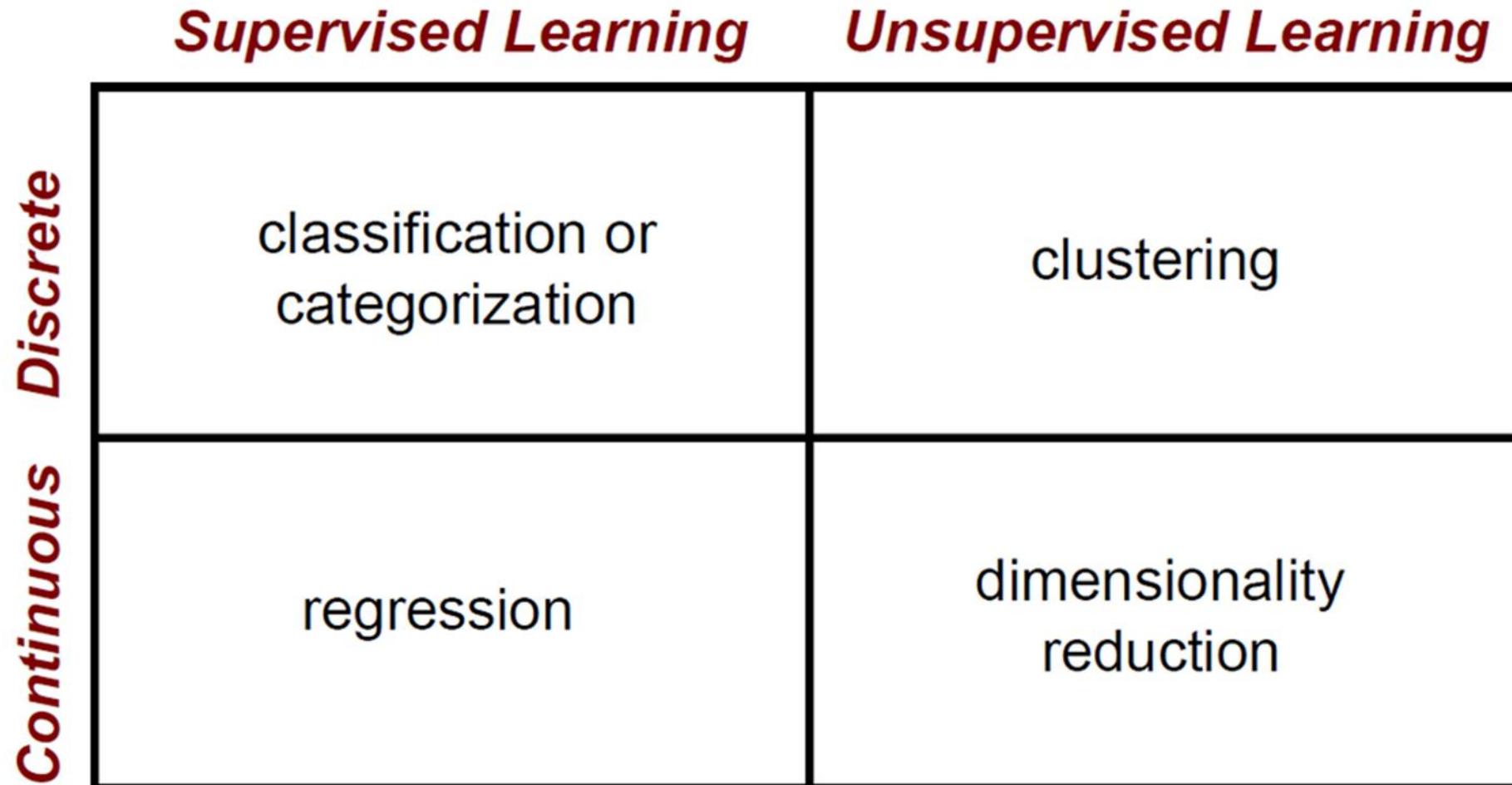
Semester 1, 2021

Changjae Oh

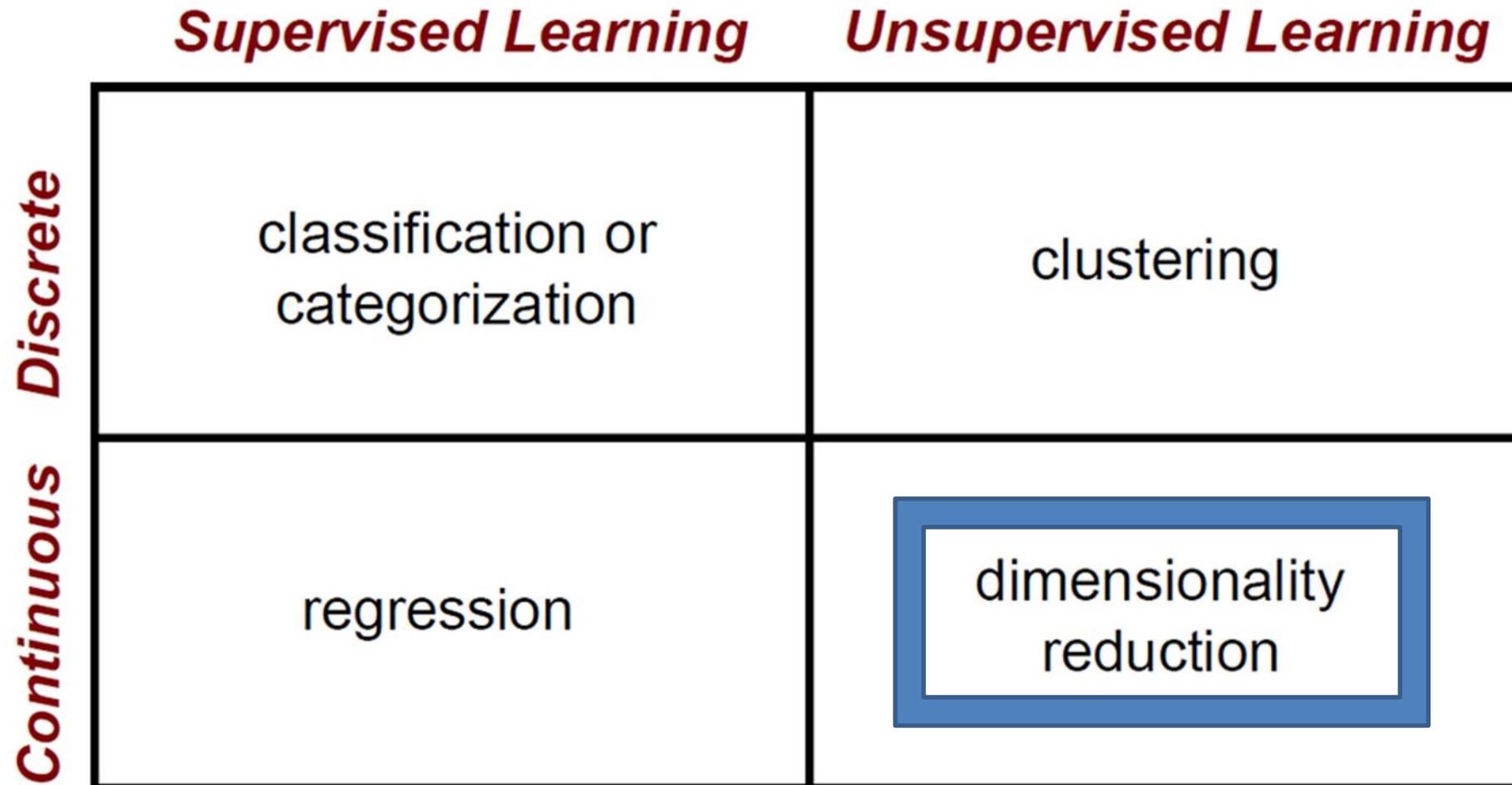
Objectives

- To understand machine learning basics for high-level vision problems

Machine learning problems

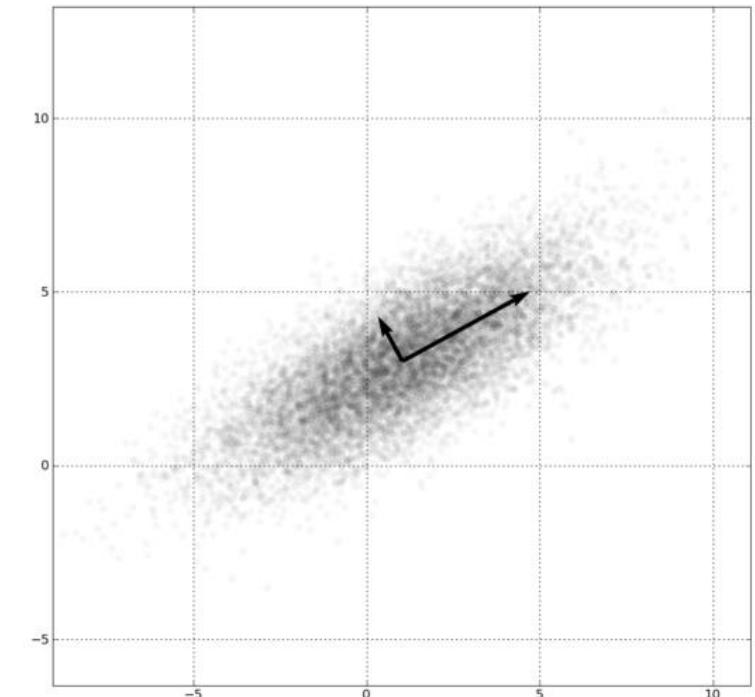


Machine learning problems

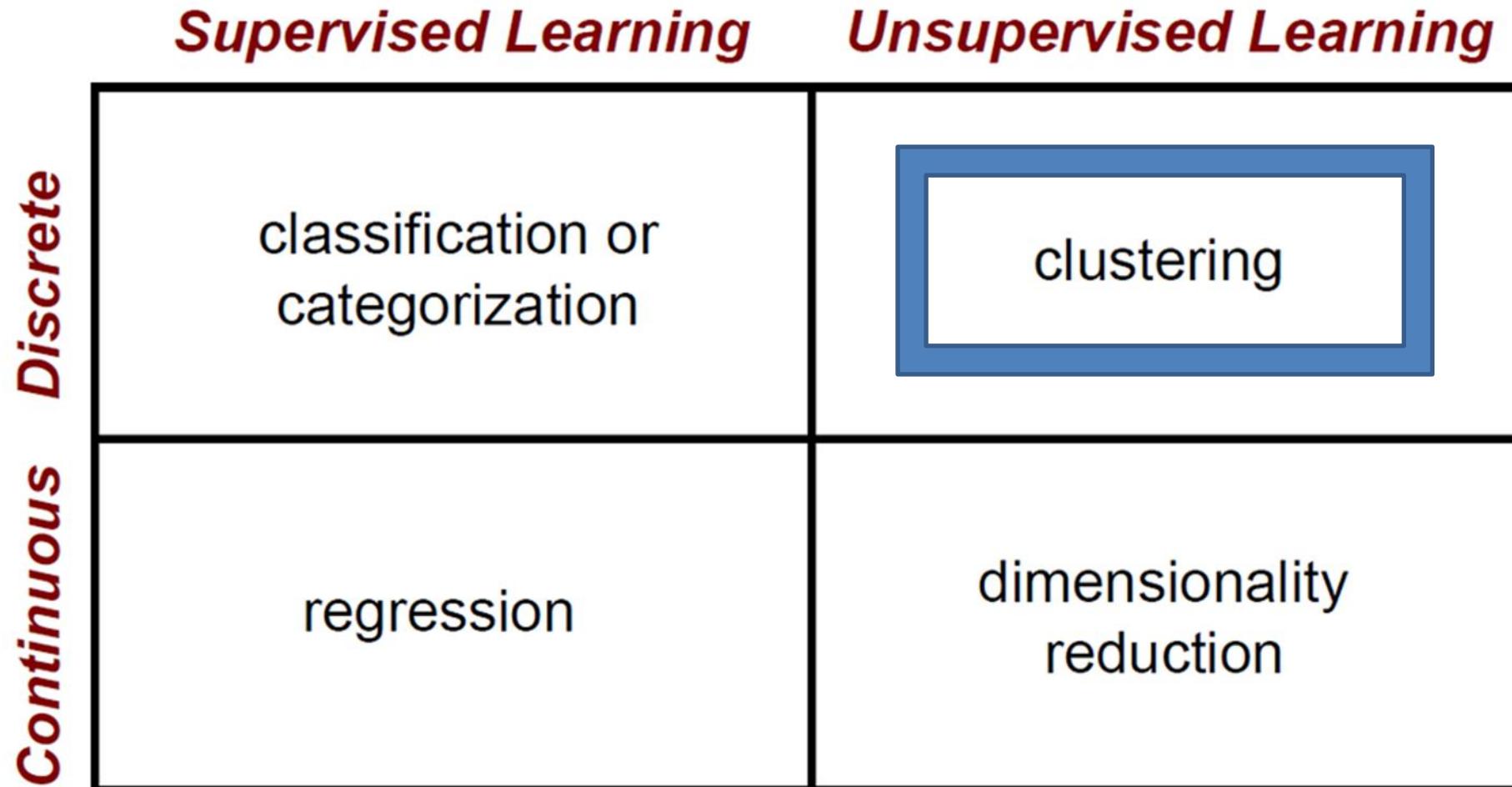


Dimensionality Reduction

- **Principal component analysis (PCA),**
 - PCA takes advantage of correlations in data dimensions to produce the best possible lower dimensional representation based on linear projections (minimizes reconstruction error).
 - PCA should be used for dimensionality reduction, not for discovering patterns or making predictions. Don't try to assign semantic meaning to the bases.
- **Locally Linear Embedding, Isomap, Autoencoder, etc.**



Machine learning problems



K-means clustering

Image



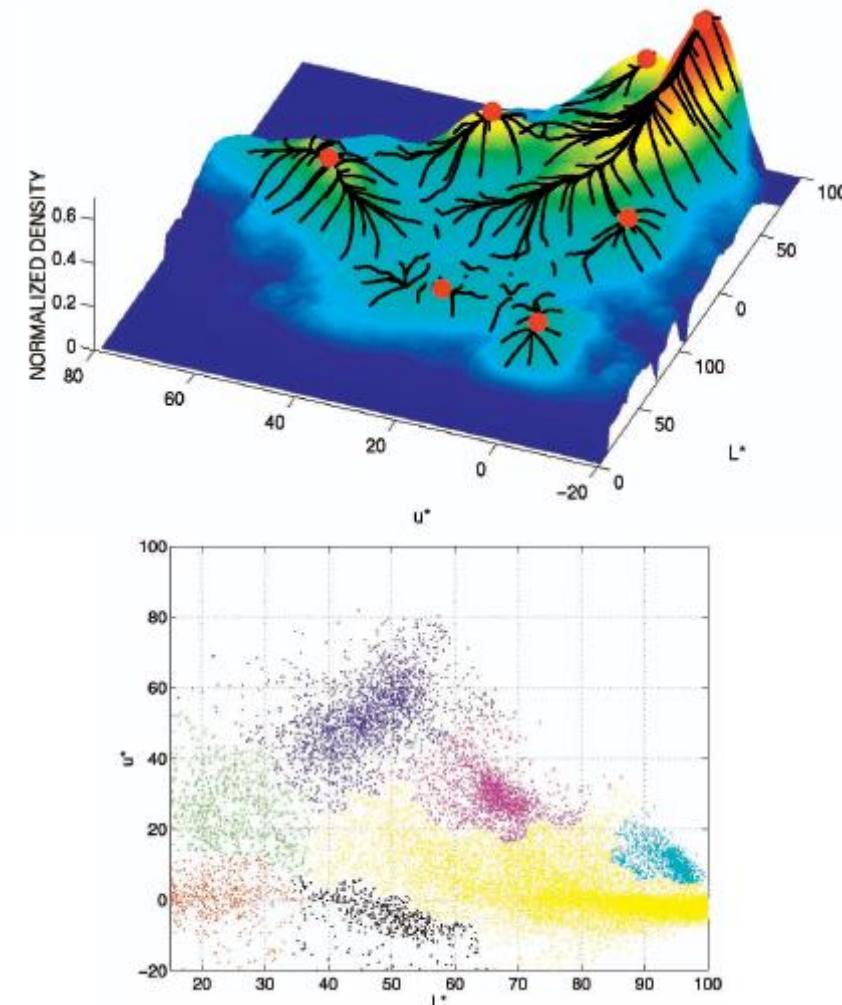
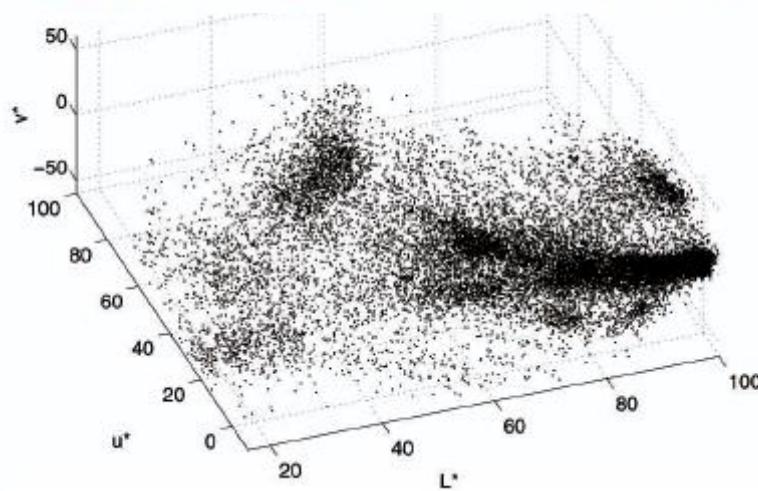
Clusters on intensity



Clusters on color

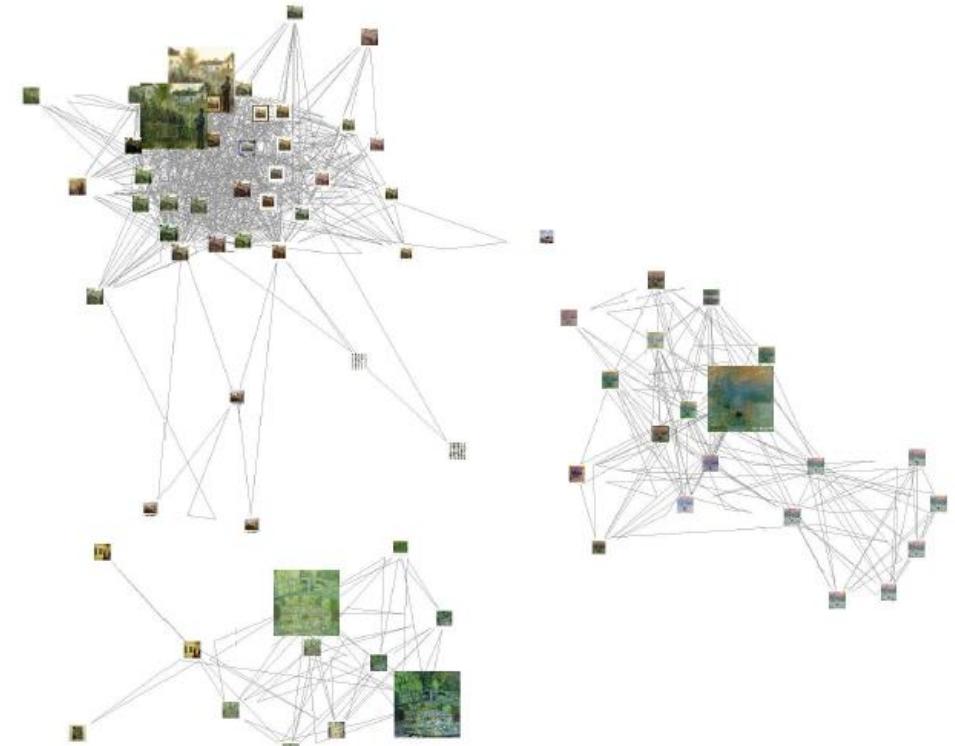
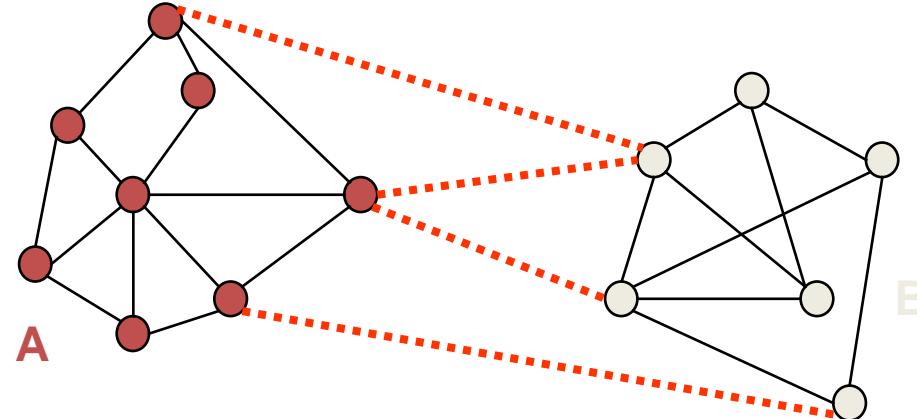


Mean shift algorithm



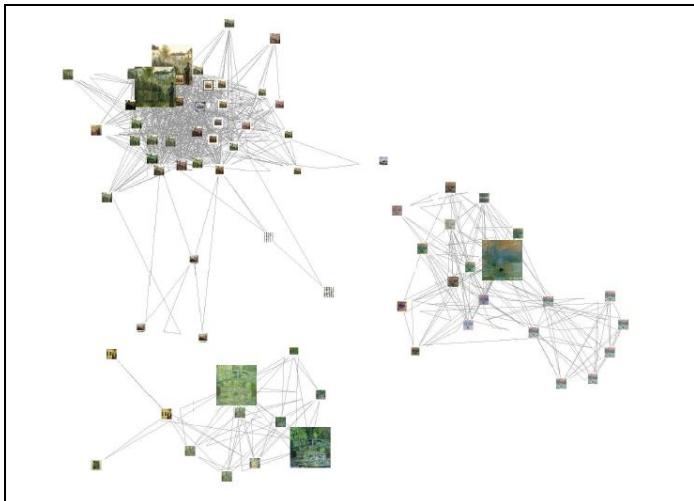
Spectral clustering

Group points based on links in a graph

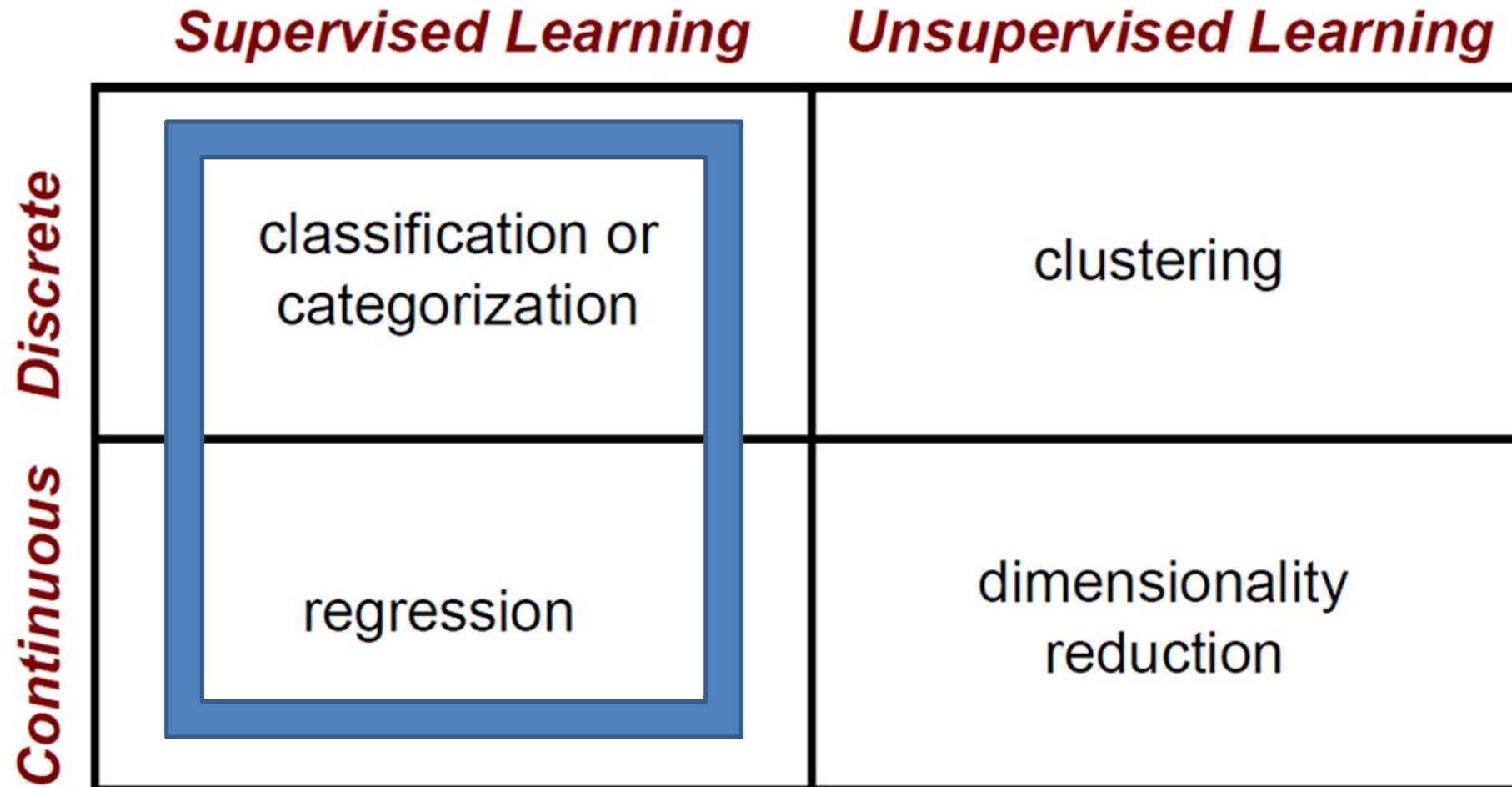


Visual PageRank

- Determining importance by random walk
 - What's the probability that you will randomly walk to a given node?
 - Create adjacency matrix based on visual similarity
 - Edge weights determine probability of transition



Machine learning problems



The machine learning framework

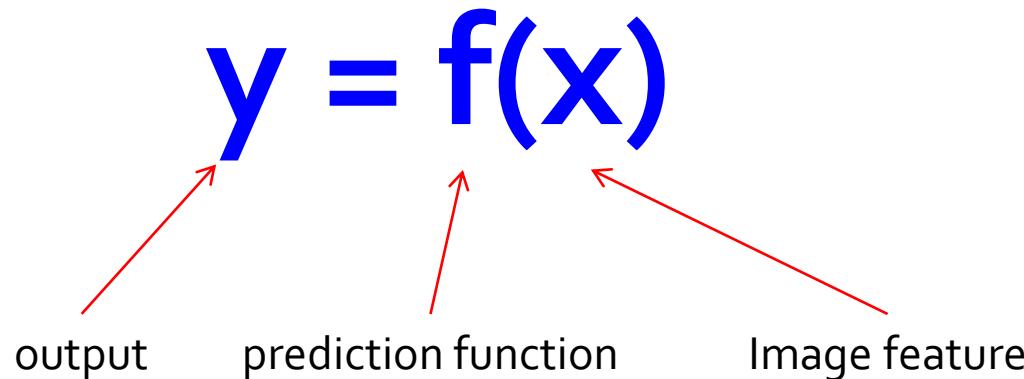
- Apply a **prediction function** to a **feature representation** of the image to get the desired output:

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

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

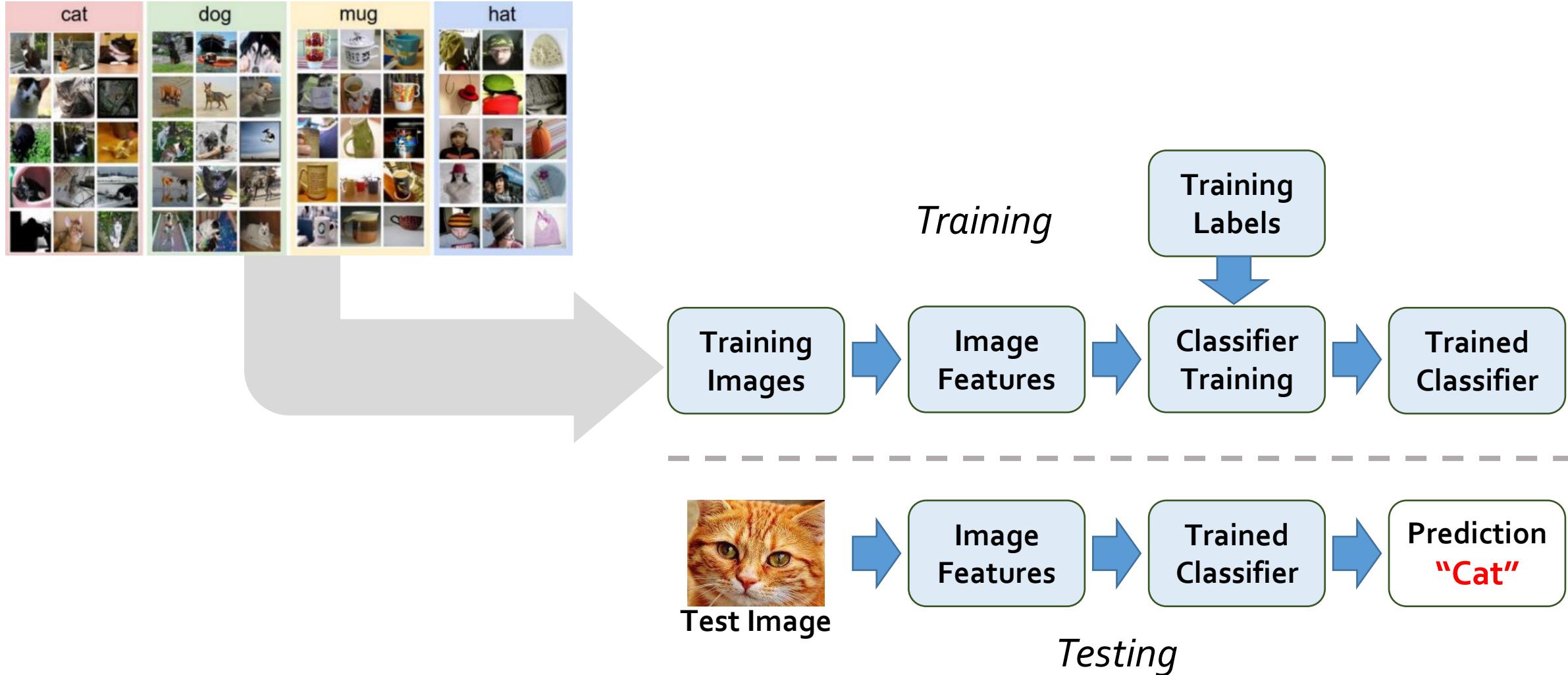
$f(\text{cow}) = \text{"cow"}$

Machine learning framework



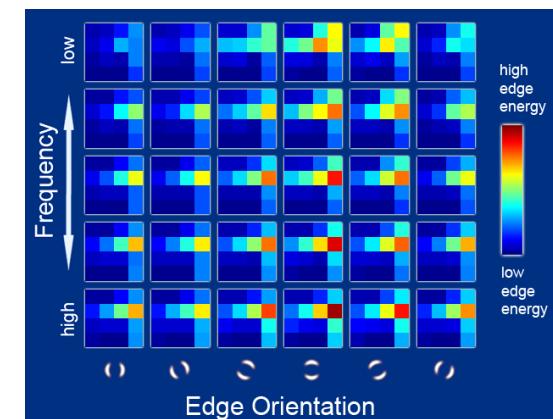
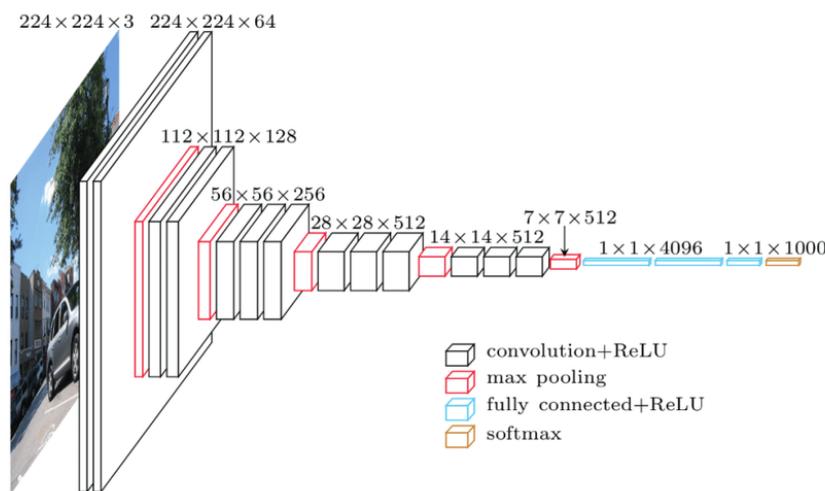
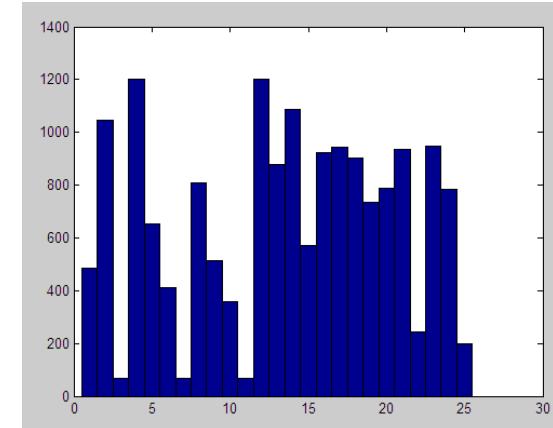
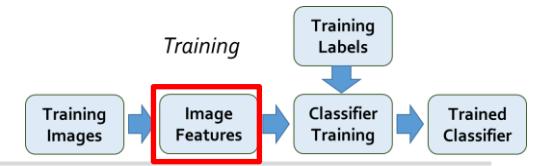
- **Training:** given a *training set* of labeled examples $\{(x_1, y_1), \dots, (x_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* x and output the predicted value $y = f(x)$

Machine learning framework



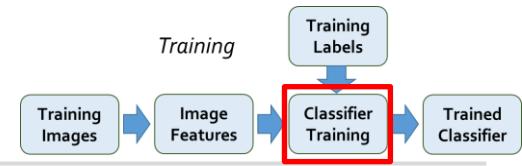
Features

- Raw pixels
- Histograms
- GIST descriptors
- CNNs
- ...

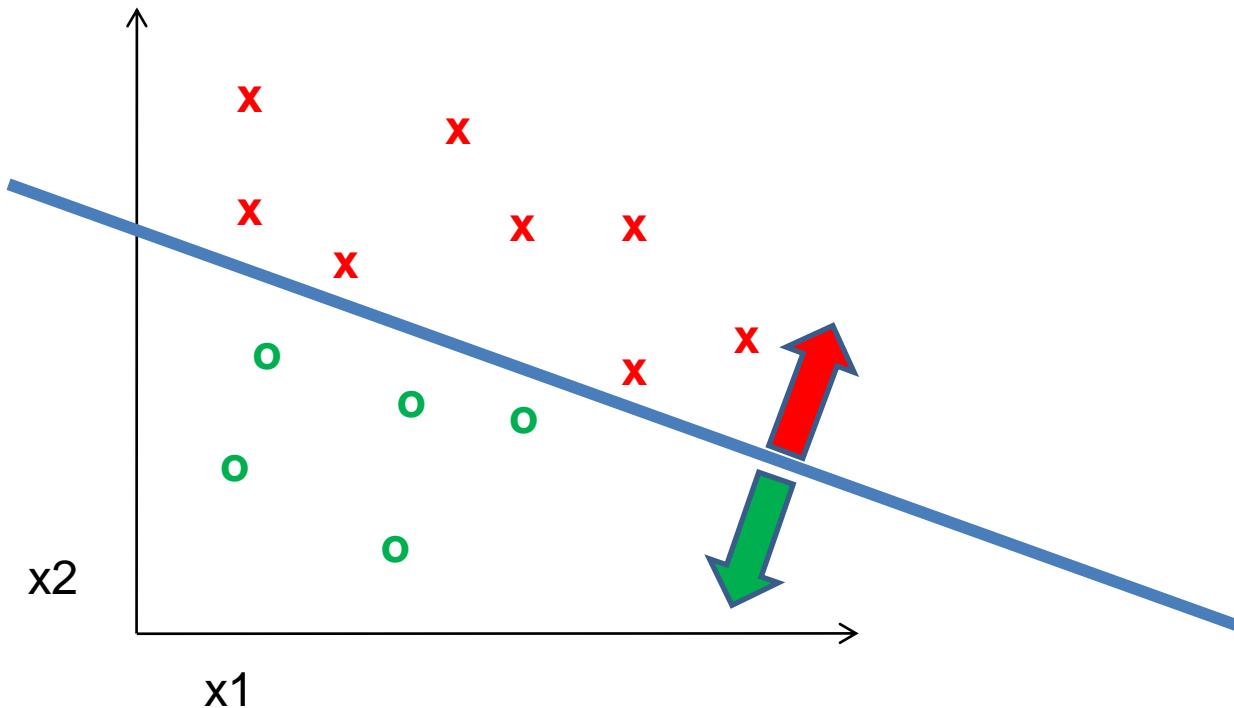


Learning a classifier

Training



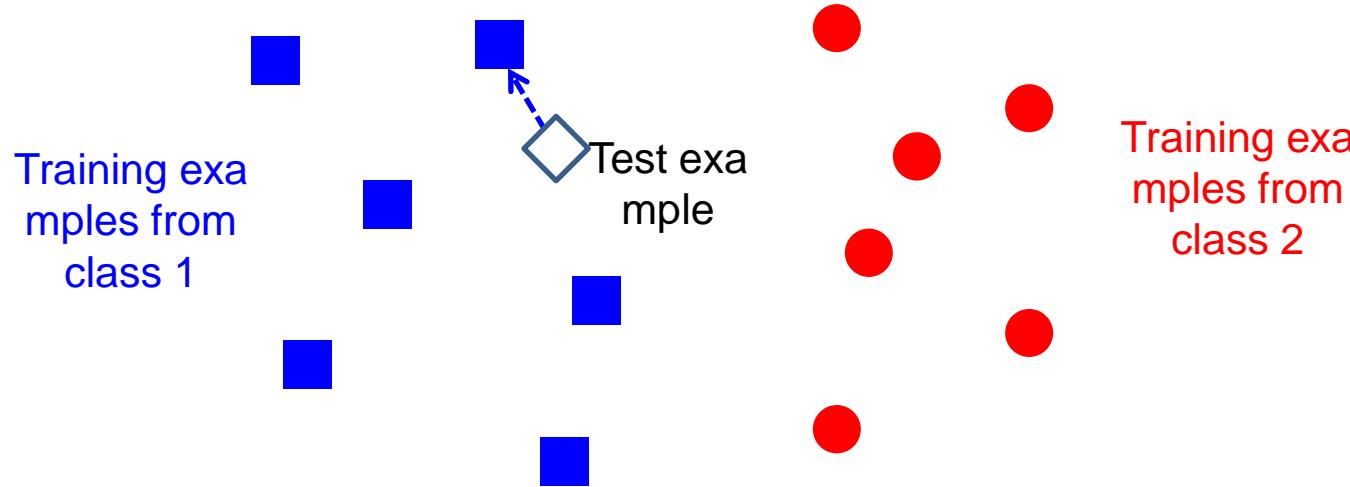
- Given some set of features with corresponding labels, learn **a function to predict the labels from the features**



Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- K-nearest neighbour
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- RBMs
- Deep Convolutional Network
- Etc.

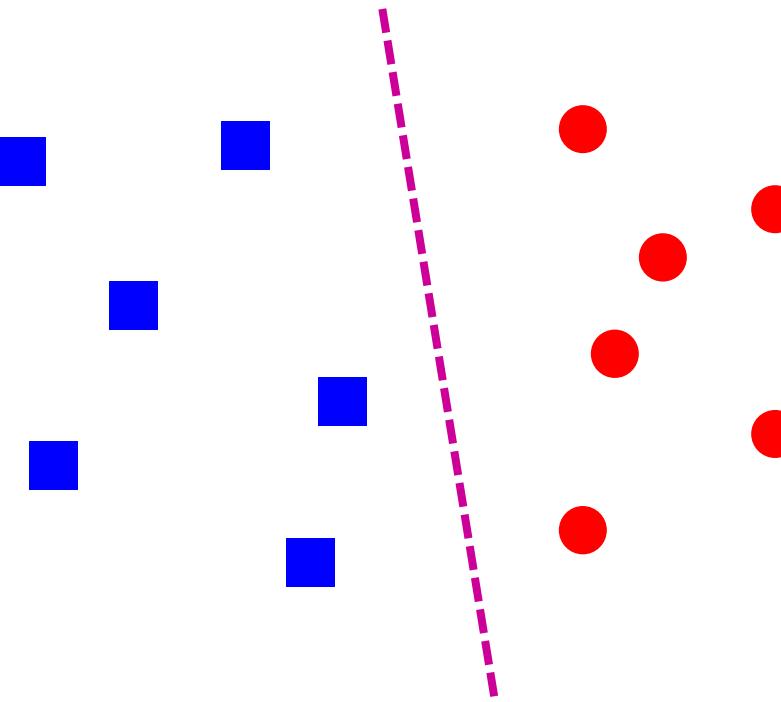
Classifiers: Nearest neighbor



$f(x) = \text{label of the training example nearest to } x$

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

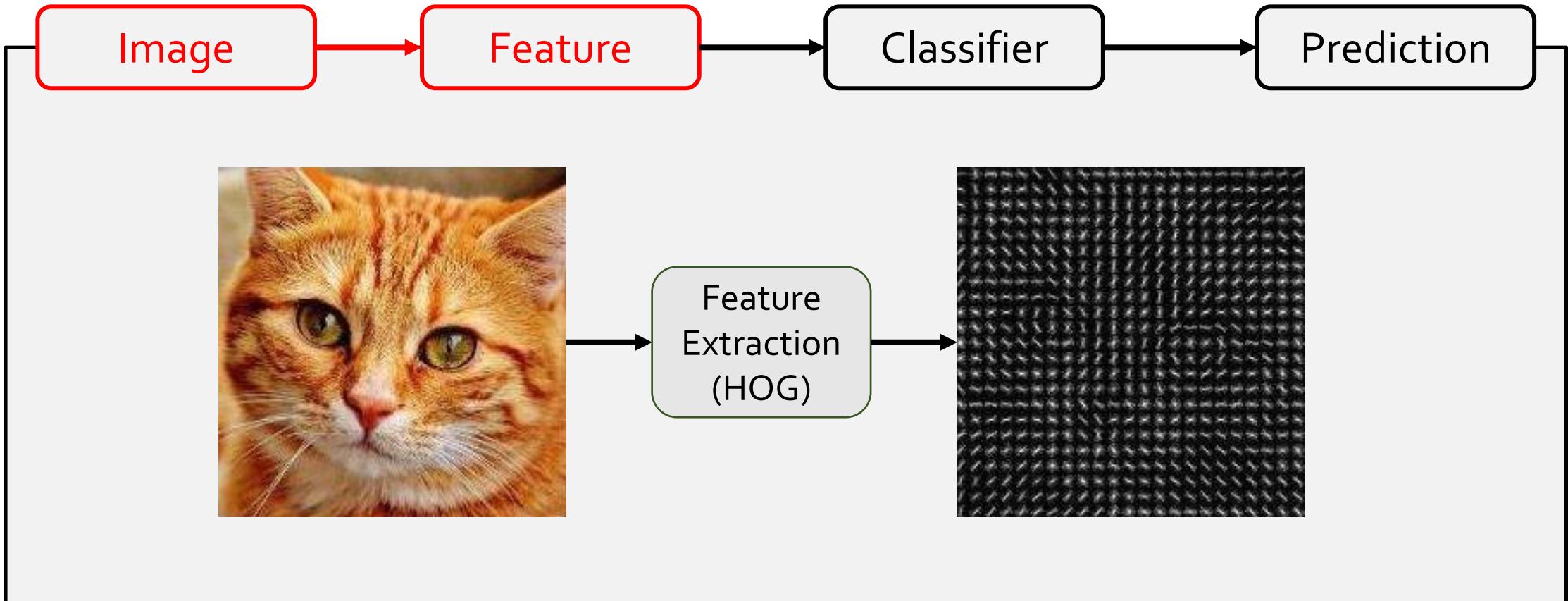
Classifiers: Linear



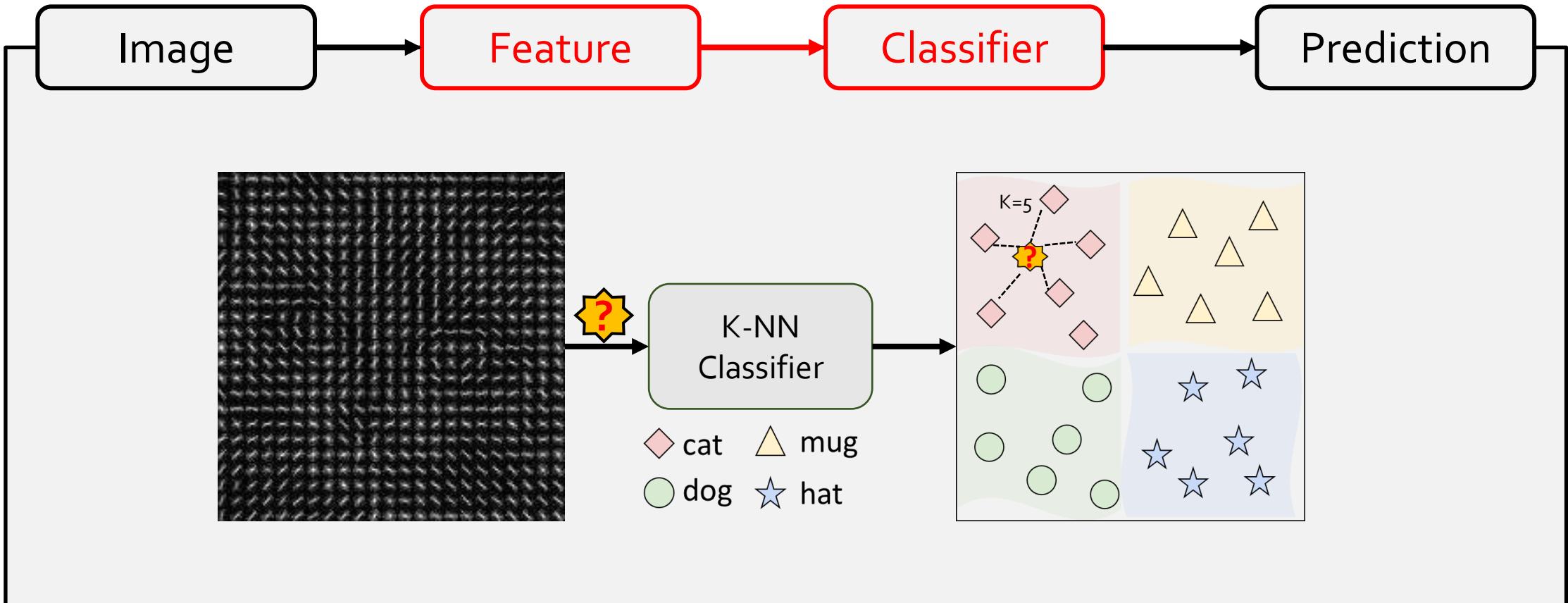
- Find a *linear function* to separate the classes:

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

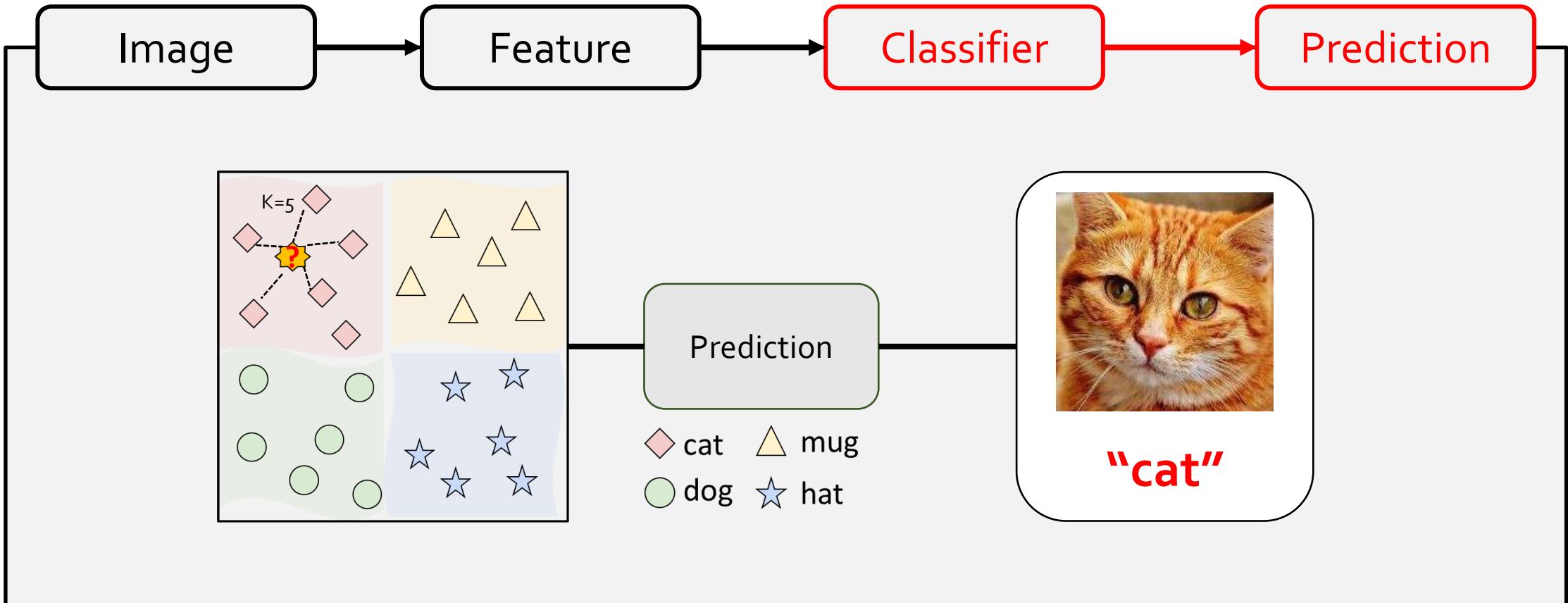
Example: Image Classification by K-NN



Example: Image Classification by K-NN



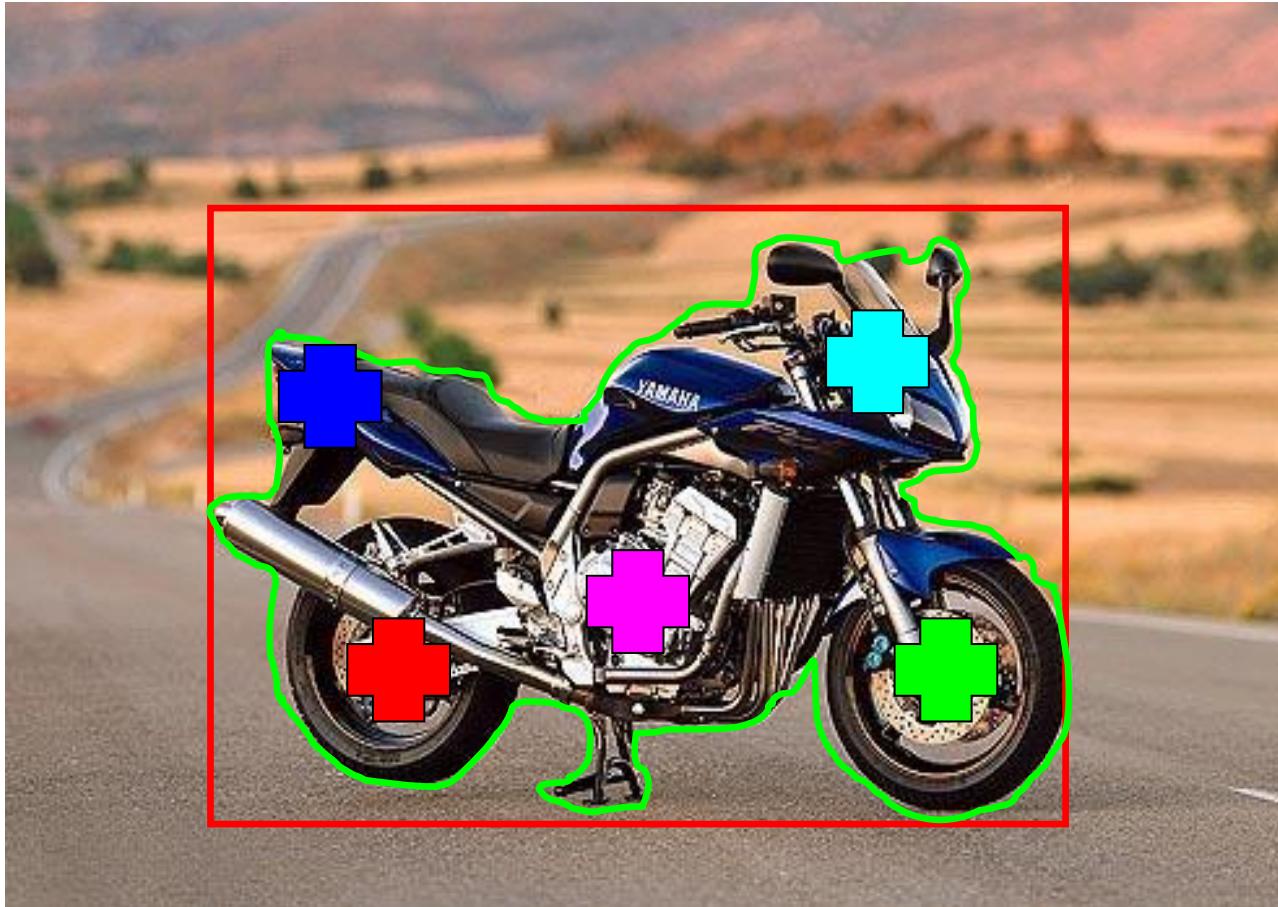
Example: Image Classification by K-NN



Recognition task and supervision

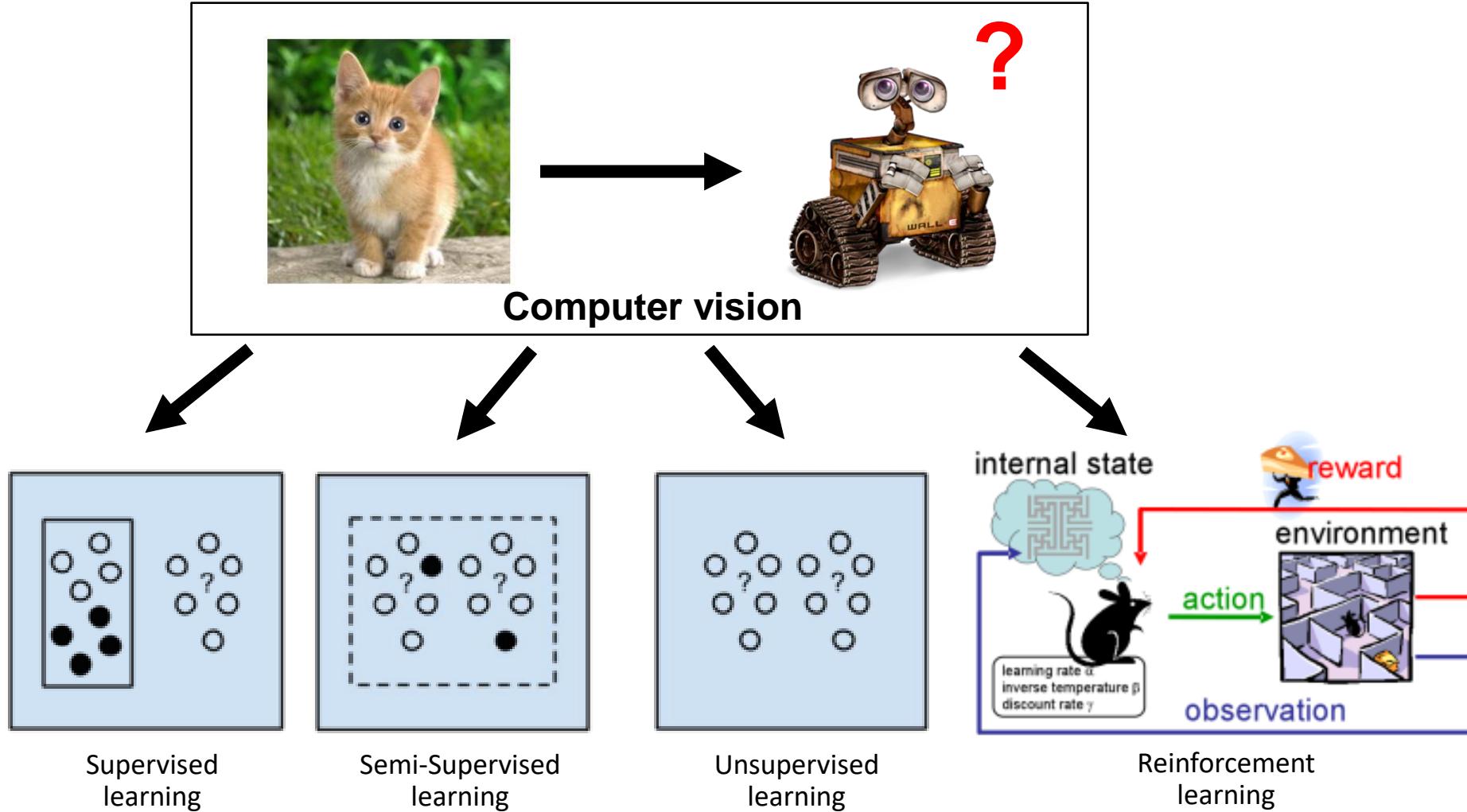
- Images in the training set must be annotated with the “**correct answer**” that the model is expected to produce

Contains a motorbike

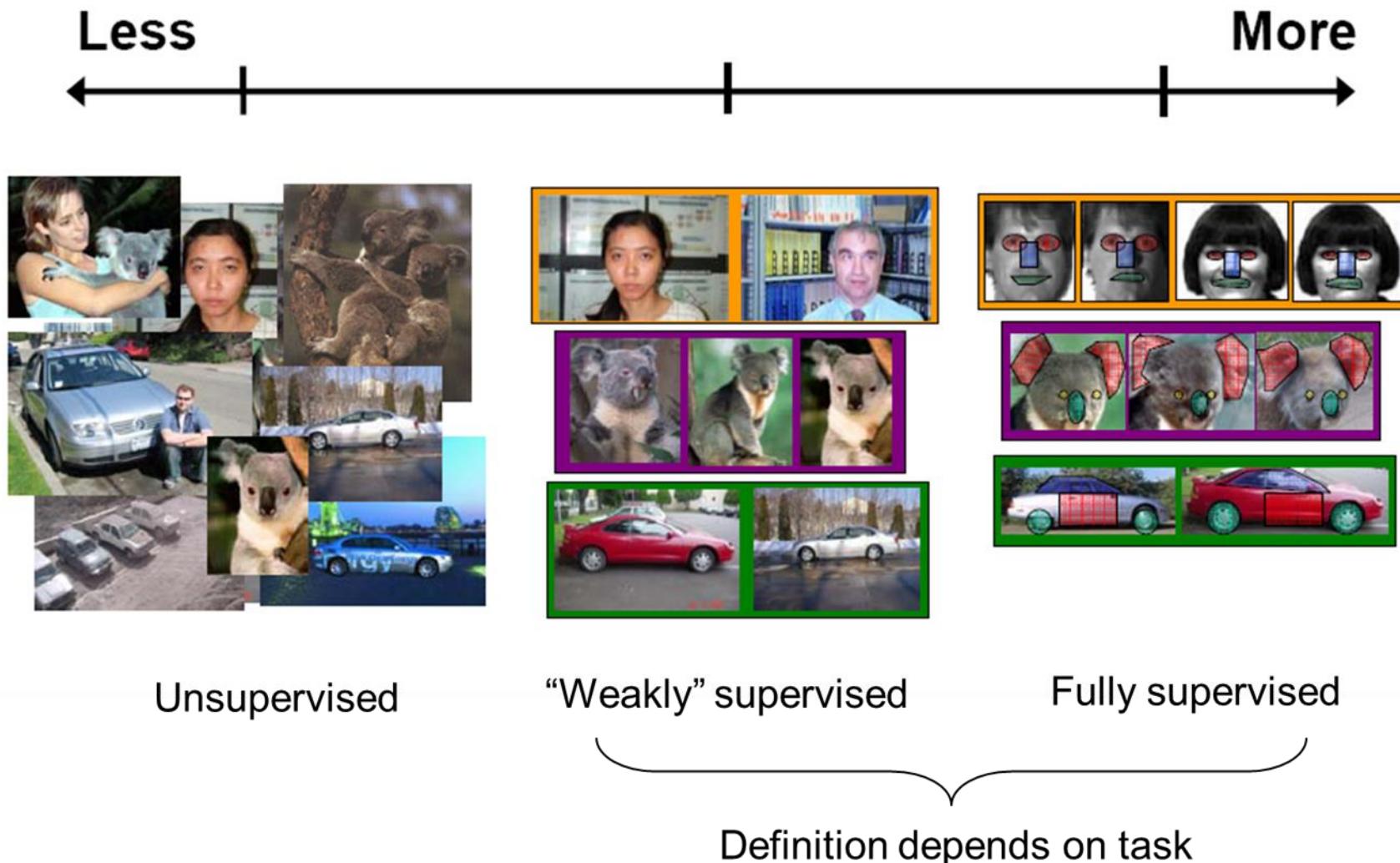


Slide credit: S. Lazebnik

Spectrum of supervision



Spectrum of supervision



Generalisation

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



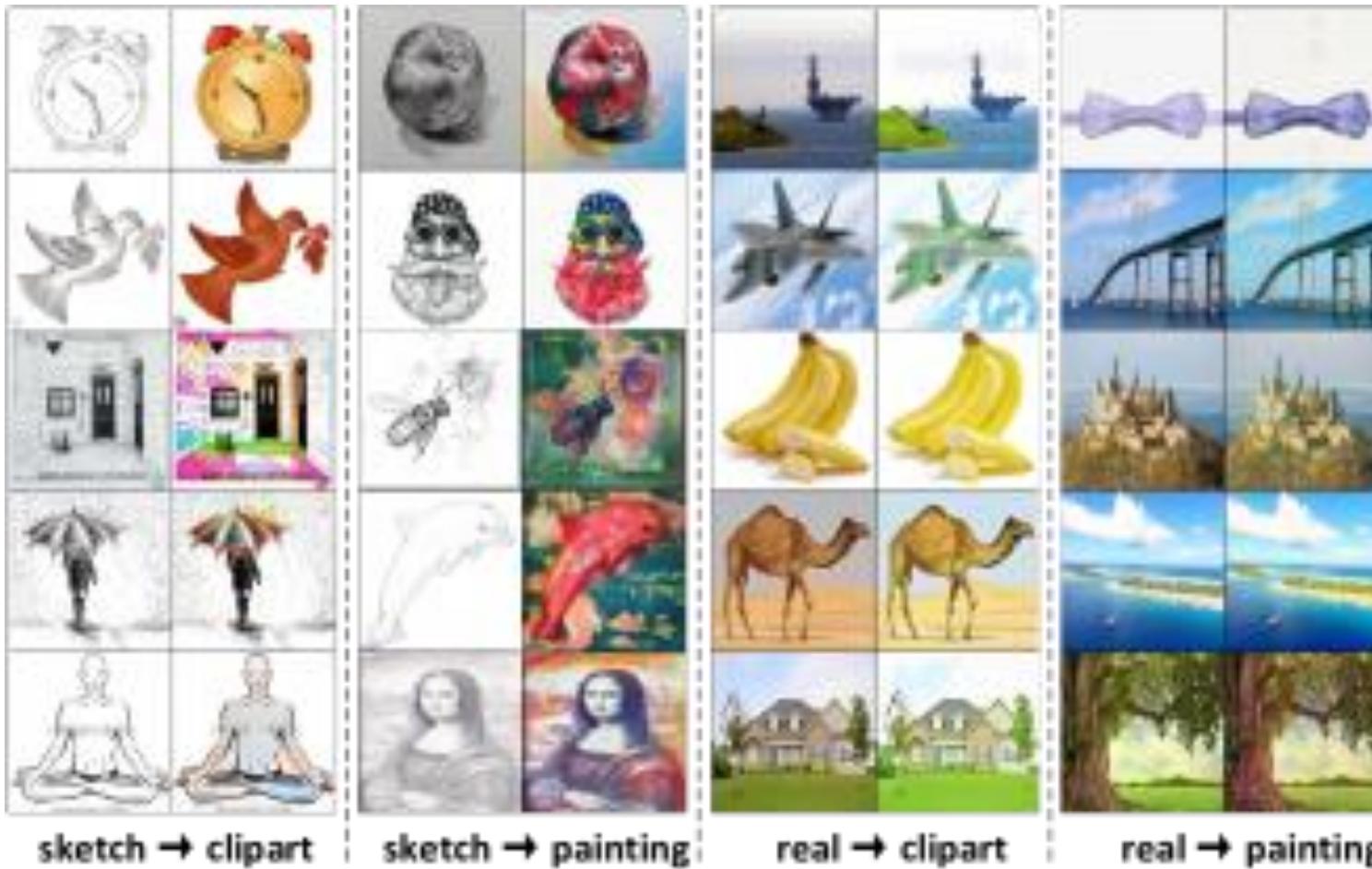
Training set (labels known)



Test set (labels unknown)

Generalisation

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



EBU7240

Computer Vision

- Classification -

Semester 1, 2021

Changjae Oh

Outline

- **Overview of recognition tasks**
- **A statistical learning approach**
- **“Classic” or “shallow” classification pipeline**
 - “Bag of features” representation
 - Classifiers: nearest neighbor, linear, SVM

Verification/Classification

Is this a building?



Adapted from Fei-fei -Li

Detection

Where are the people?



Adapted from Fei-fei -Li

Identification

Is this 天安門?



Adapted from Fei-fei -Li

Semantic Segmentation



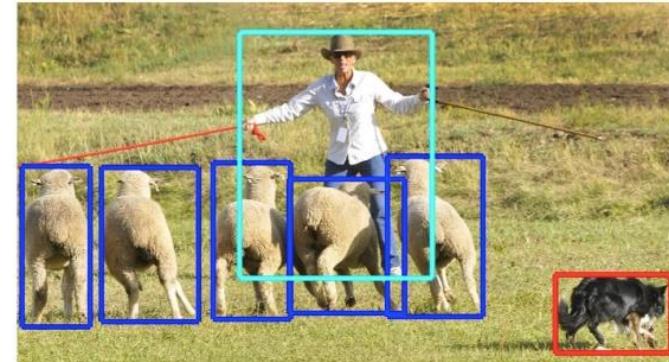
Adapted from Fei-fei -Li

Object recognition

- A collection of related tasks for identifying objects in digital photographs.
- Consists of recognizing, identifying, and locating objects within a picture with a given degree of confidence.



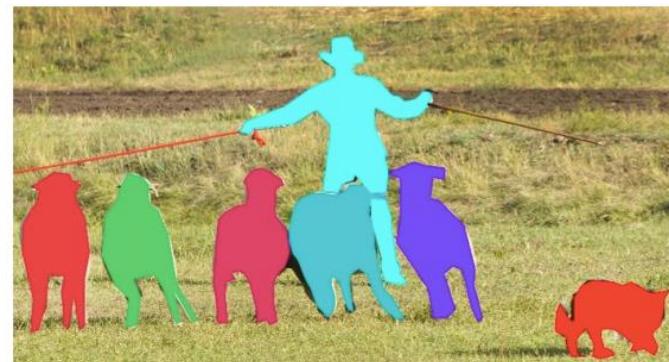
image classification



object detection



semantic segmentation

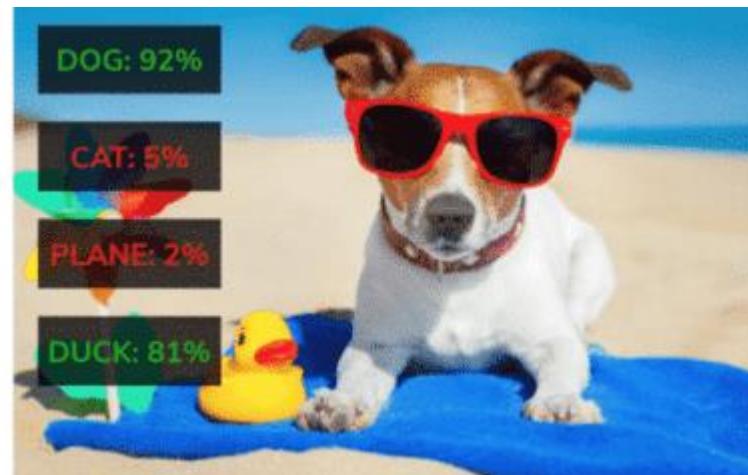


instance segmentation

[Image source](#)

Image classification vs Object detection

- **Image classification**
 - Identifying what is in the image and the associated level of confidence.
 - can be binary label or multi-label classification
- **Object detection**
 - Localising and classifying one or more objects in an image
 - Object localisation and image classification



Semantic segmentation vs Instance segmentation

- **Semantic segmentation**

- Assigning a label to every pixel in the image.
- Treating multiple objects of the same class as a single entity

- **Instance segmentation**

- Similar process as semantic segmentation, but identifies , for each pixel, the object instance it belongs to.
- Treating multiple objects of the same class as distinct individual objects (or instances)
- typically, instance segmentation is harder than semantic segmentation

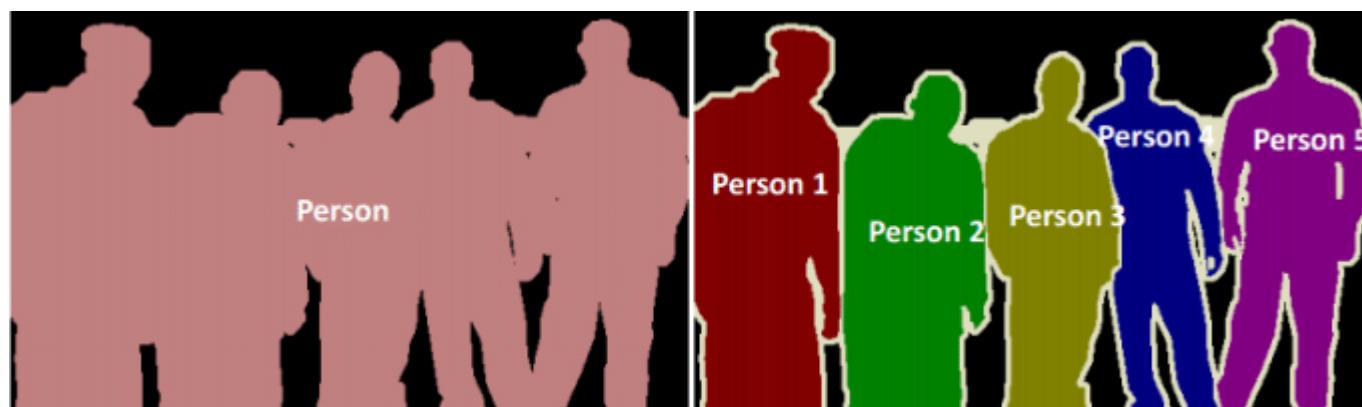


Image classification



The machine learning framework

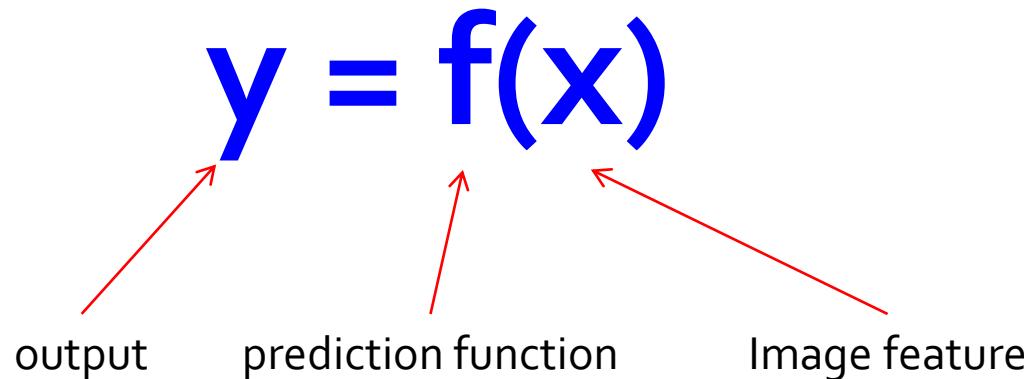
- Apply a **prediction function** to a **feature representation** of the image to get the desired output:

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

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

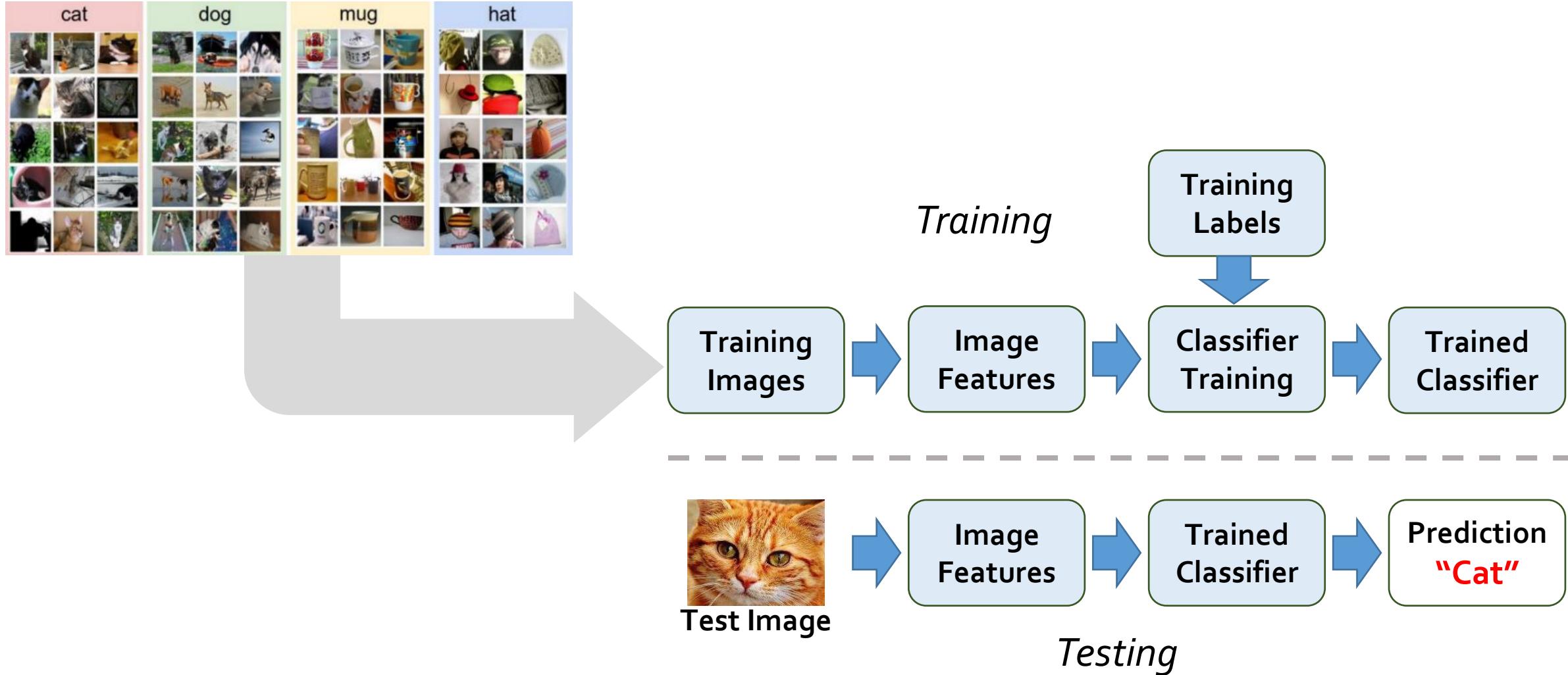
$f(\text{cow}) = \text{"cow"}$

Machine learning framework



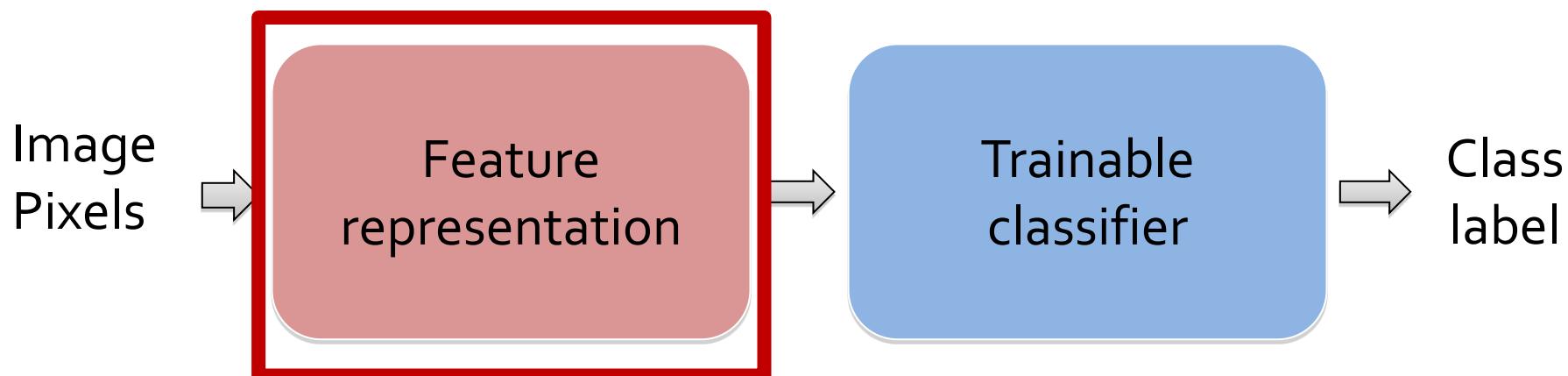
- **Training:** given a *training set* of labeled examples $\{(x_1, y_1), \dots, (x_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* x and output the predicted value $y = f(x)$

Machine learning framework



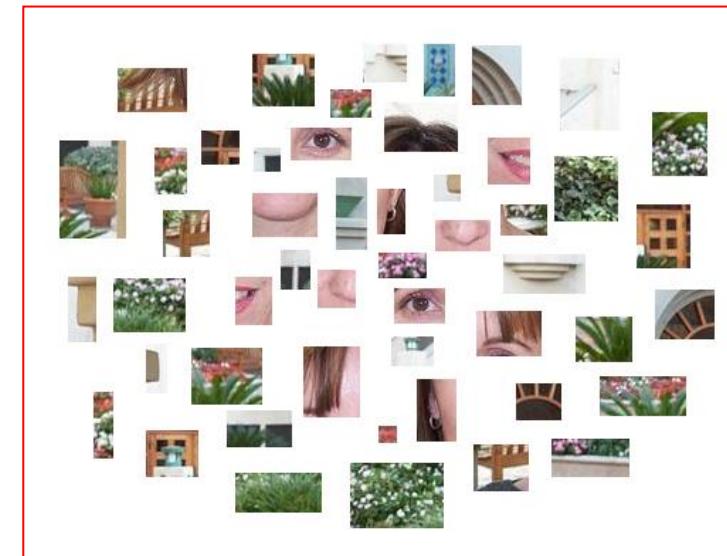
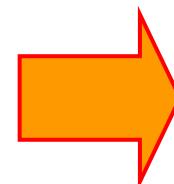
“Classic” recognition pipeline

- Hand-crafted feature representation
- Off-the-shelf trainable classifier



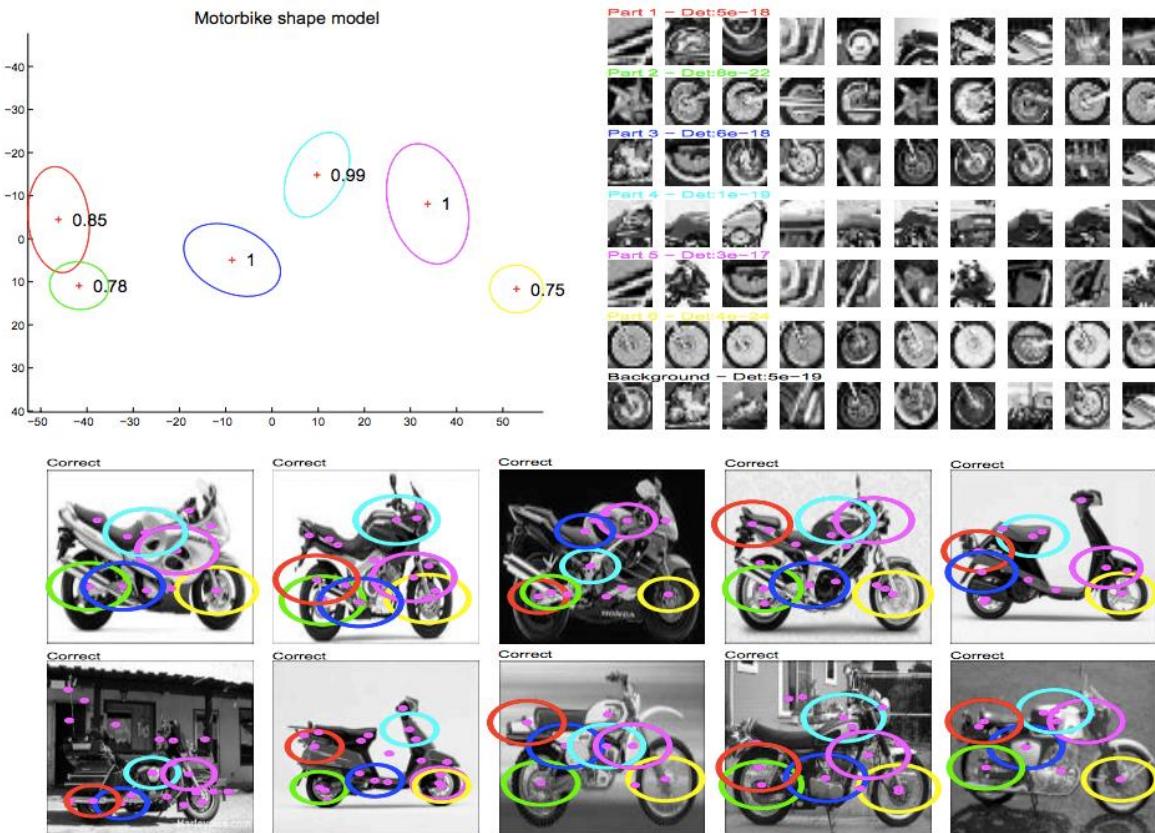
“Classic” representation: Bag of features

- Representing images as orderless collections of local features



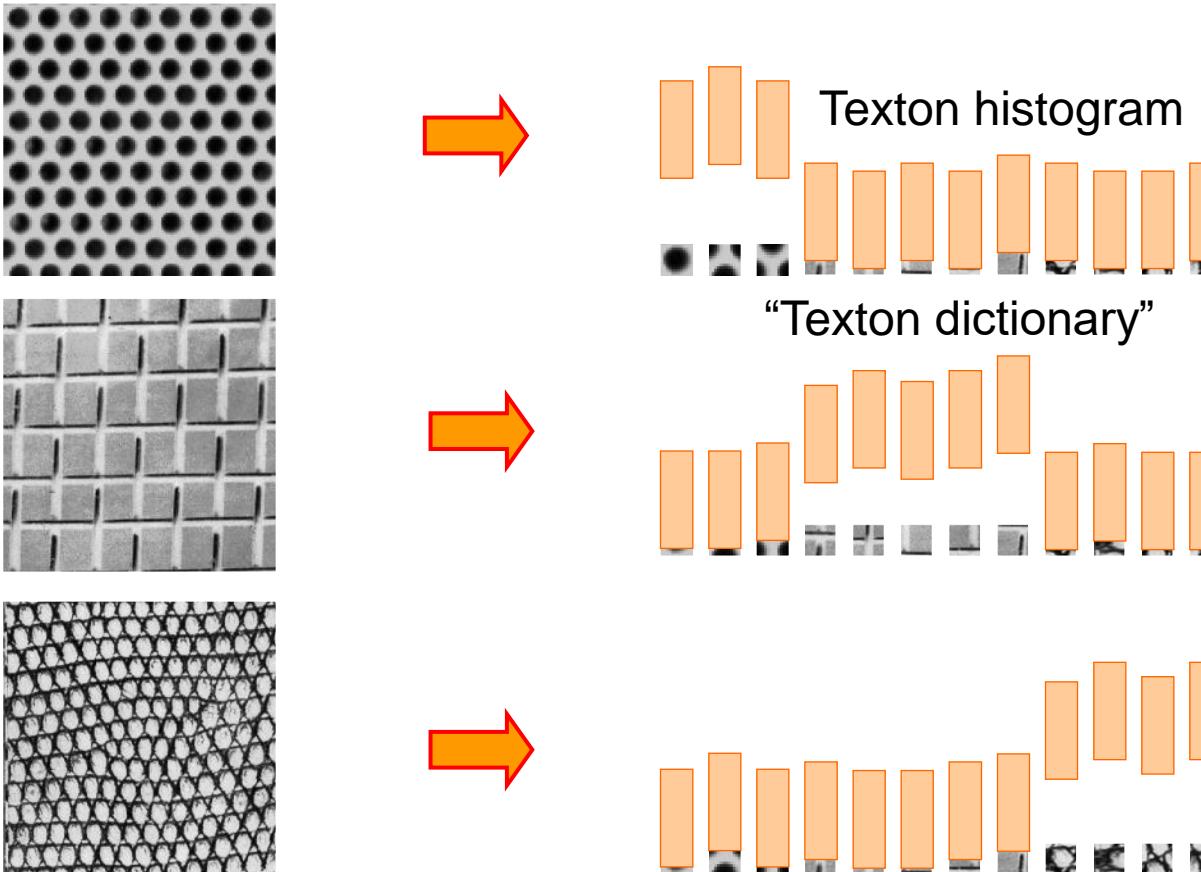
Motivation 1: Part-based models

- Various parts of the image are used separately to determine if and where an object of interest exists



Motivation 2: Texture models

- Texture is characterised by the repetition of basic elements or textons



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

Motivation 3: Bags of words

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



Motivation 3: Bags of words

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



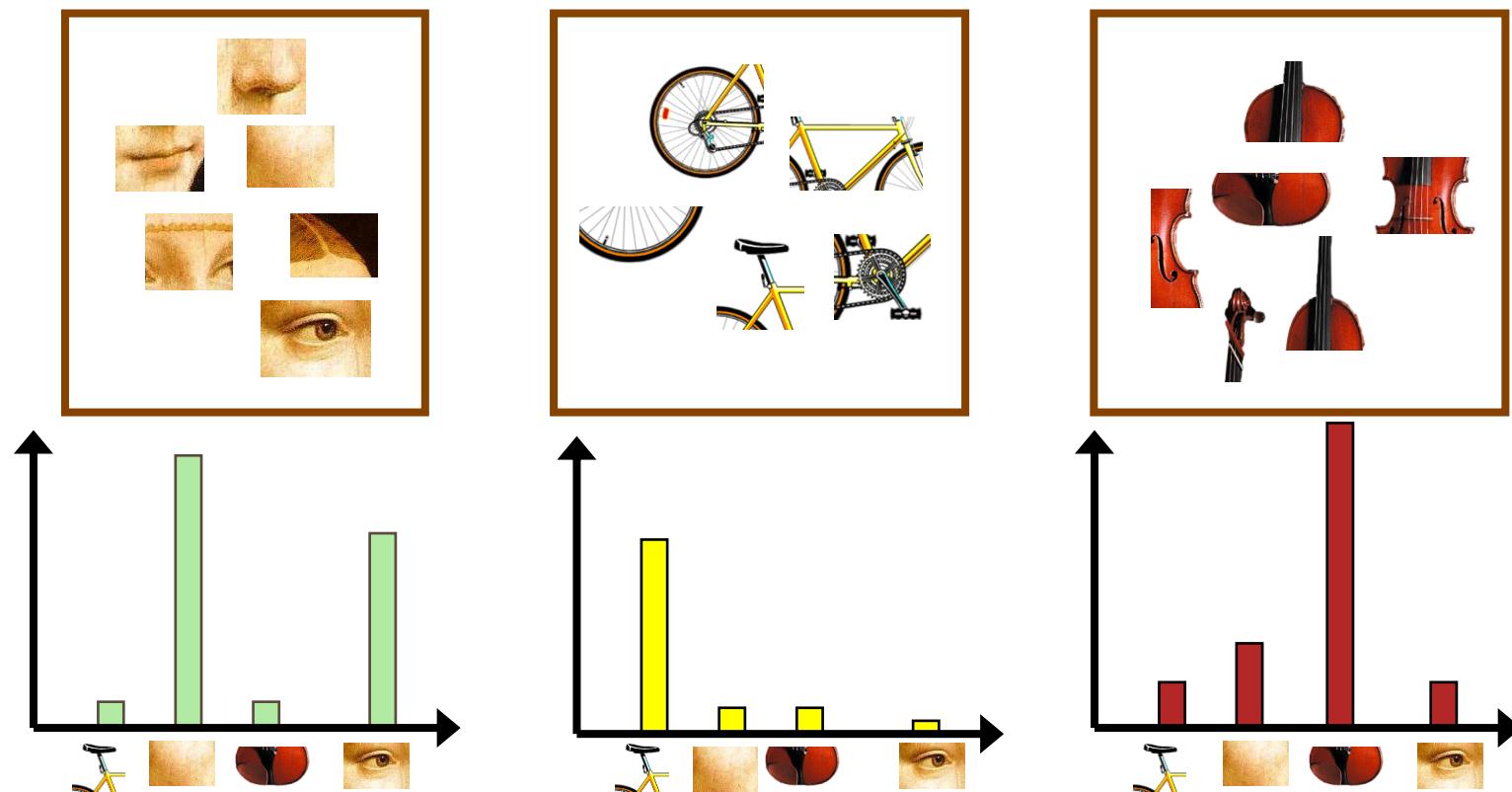
Motivation 3: Bags of words

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



Bag of features: Outline

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

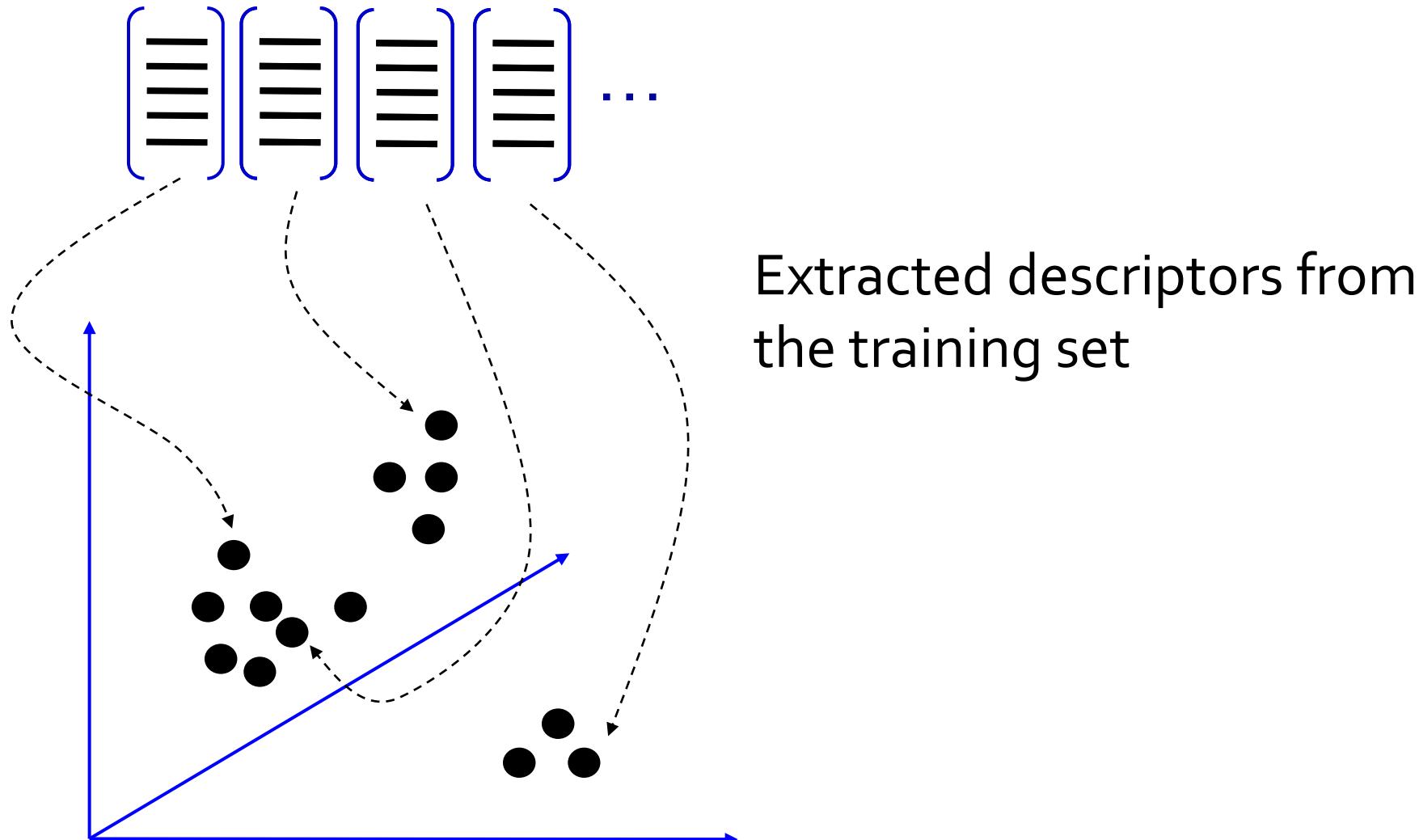


1. Local feature extraction

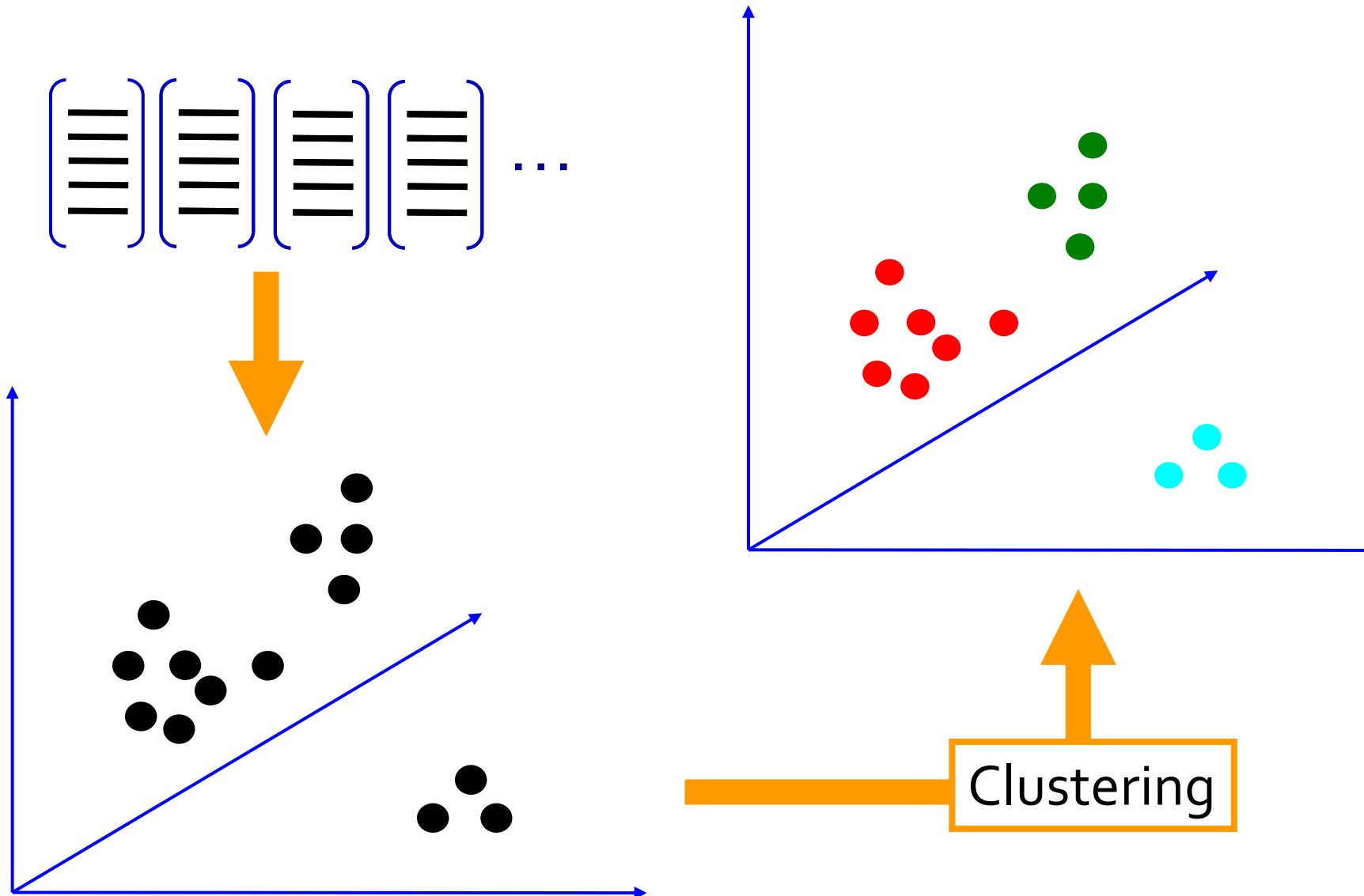
- Sample patches and extract descriptors



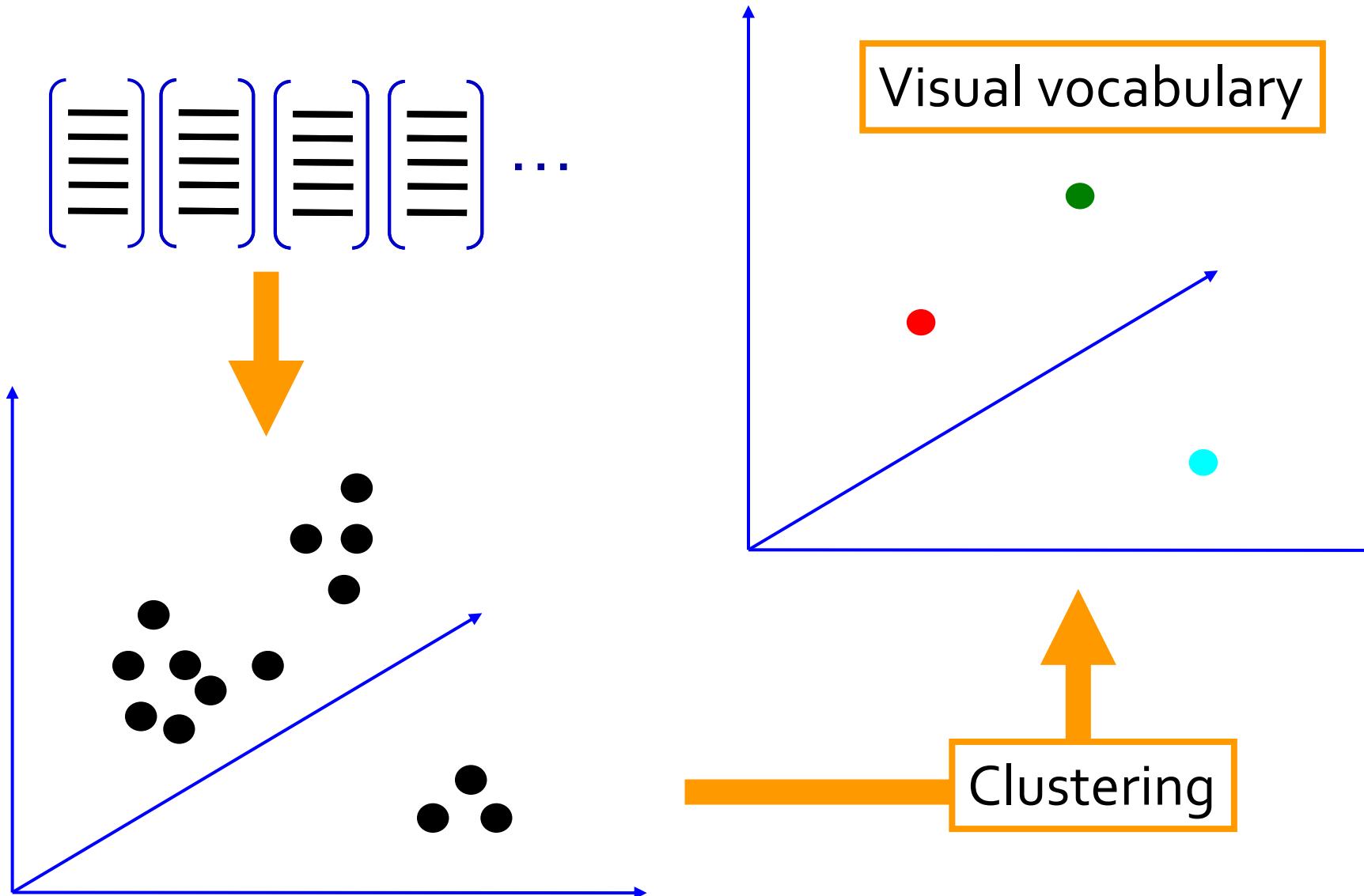
2. Learning the visual vocabulary



2. Learning the visual vocabulary



2. Learning the visual vocabulary



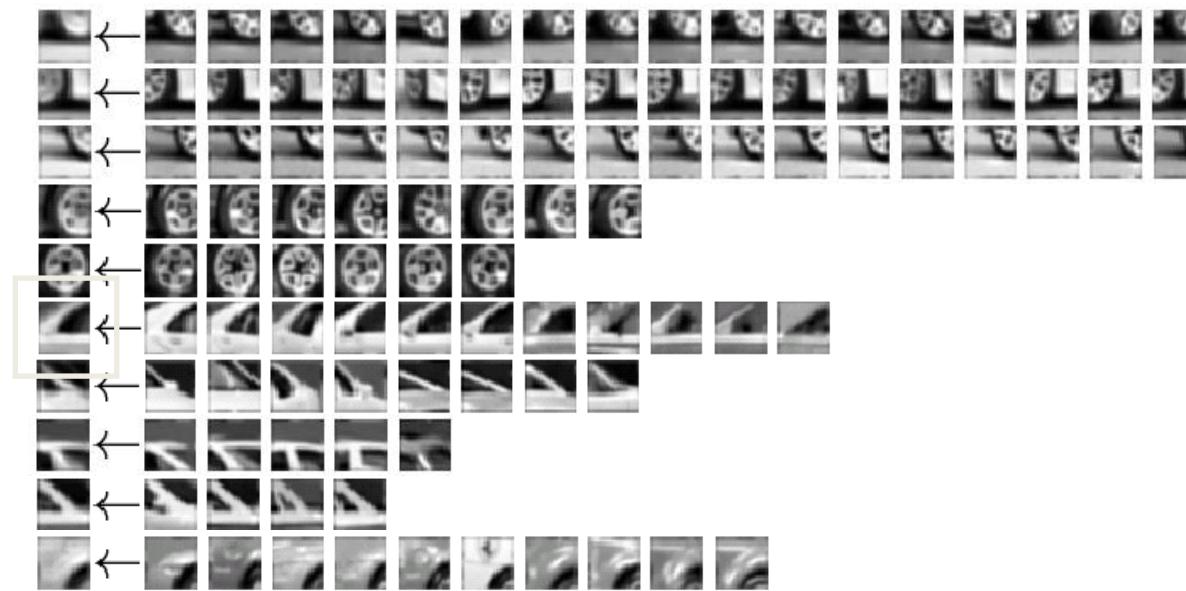
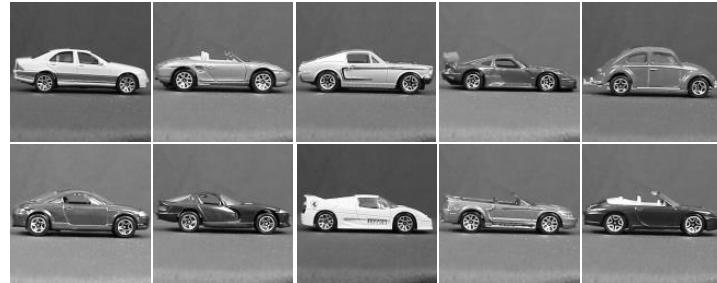
Recall: K-means clustering

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

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

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

Visual vocabularies

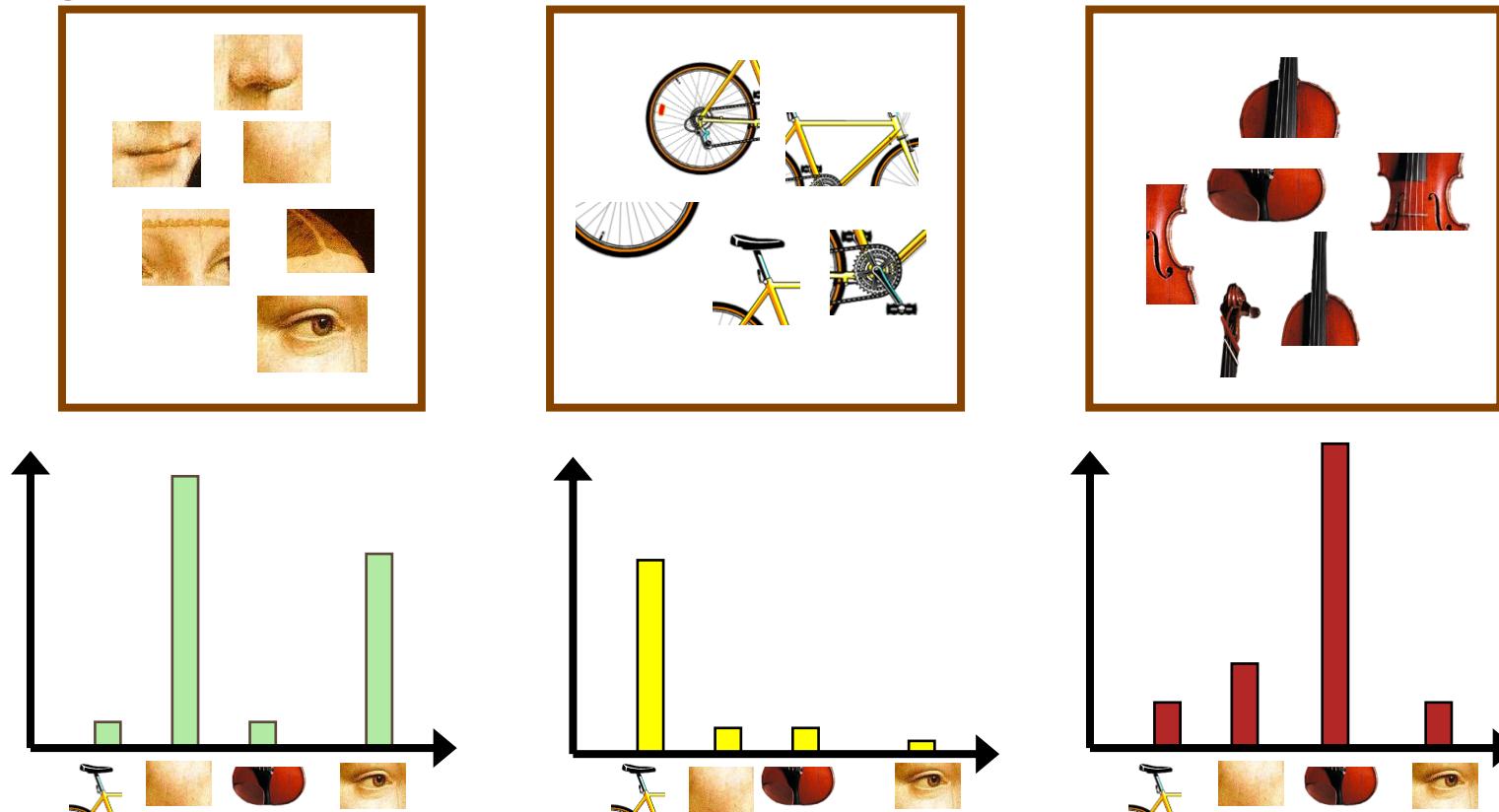


Appearance codebook

Source: B. Leibe

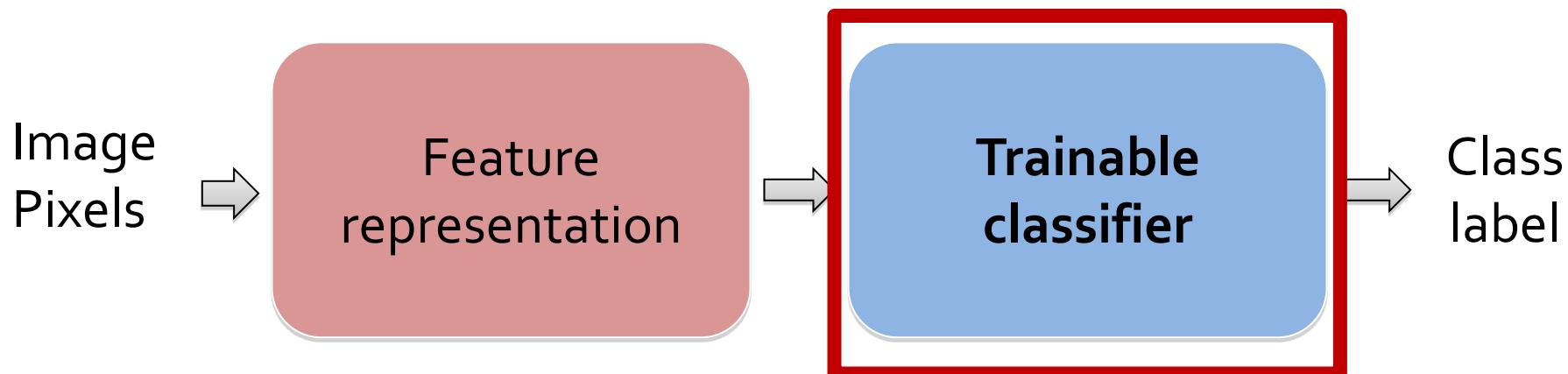
Bag of features: Outline

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

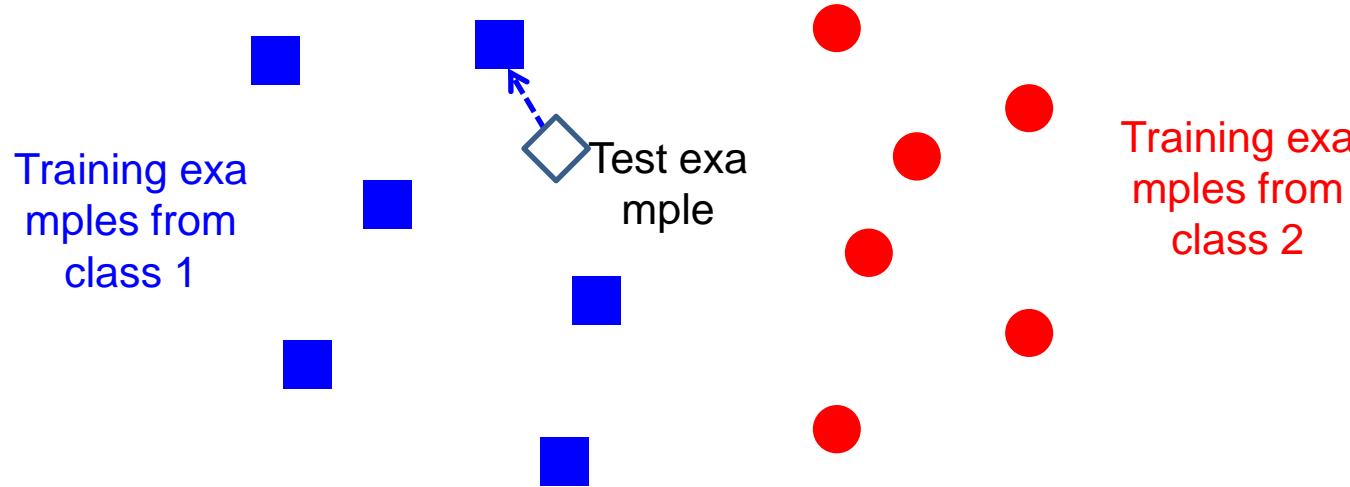


“Classic” recognition pipeline

- Hand-crafted feature representation
- Trainable classifier
 - Nearest Neighbor classifiers
 - Support Vector machines



Classifiers: Nearest neighbor



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

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

Functions for comparing histograms

- **L₁ distance:**
$$D(h_1, h_2) = \sum_{i=1}^N |h_1(i) - h_2(i)|$$
- **χ^2 distance:**
$$D(h_1, h_2) = \sum_{i=1}^N \frac{(h_1(i) - h_2(i))^2}{h_1(i) + h_2(i)}$$

- **Quadratic distance (*cross-bin distance*):**

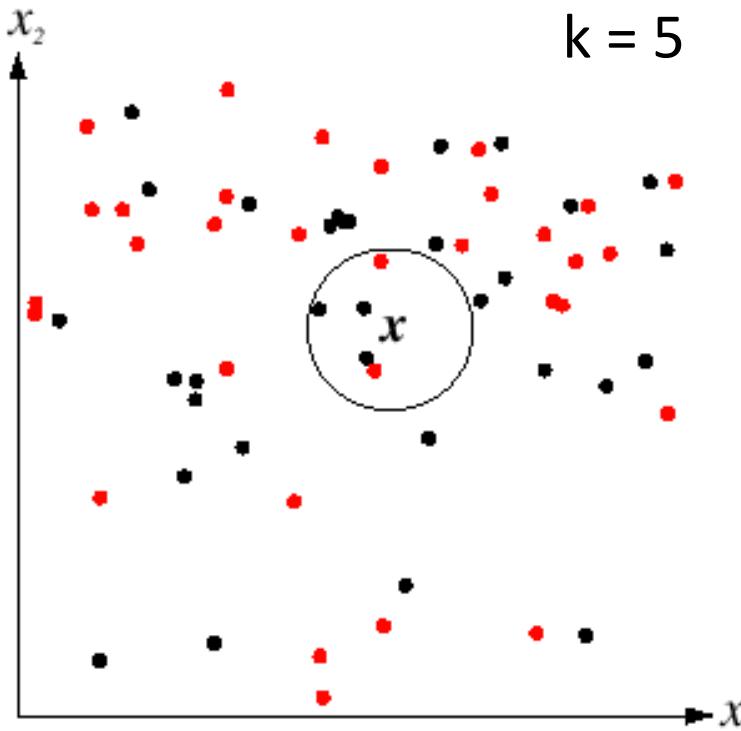
$$D(h_1, h_2) = \sum_{i,j} A_{ij} (h_1(i) - h_2(j))^2$$

- **Histogram intersection (similarity function):**

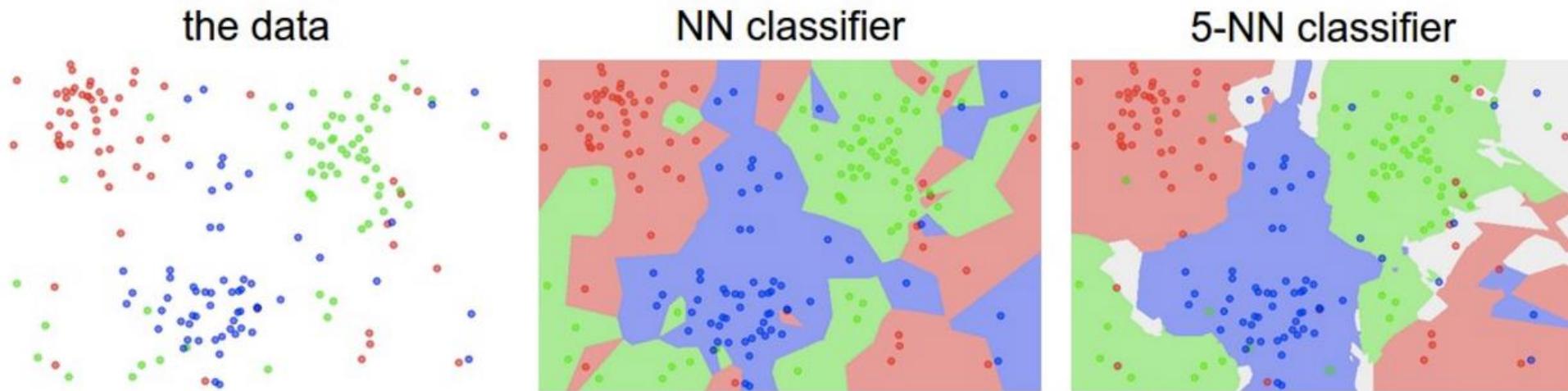
$$I(h_1, h_2) = \sum_{i=1}^N \min(h_1(i), h_2(i))$$

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



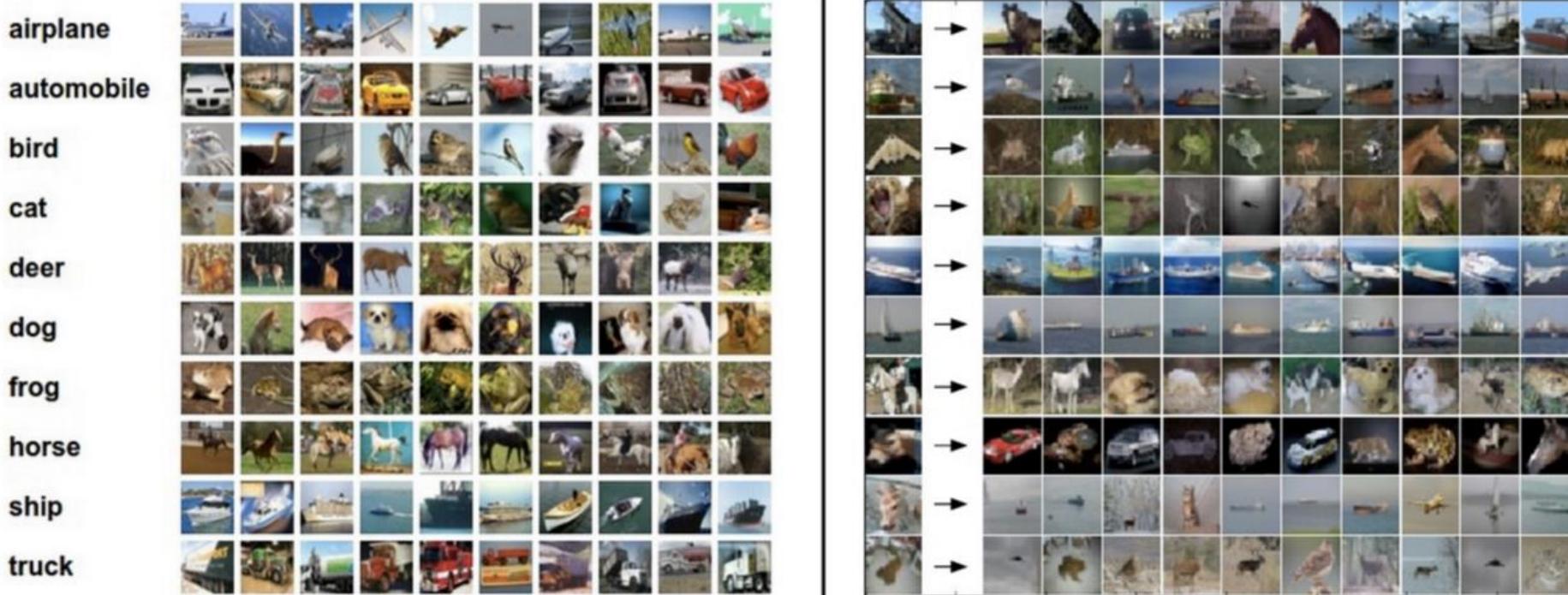
K-nearest neighbor classifier



- Which classifier is more robust to *outliers*?

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

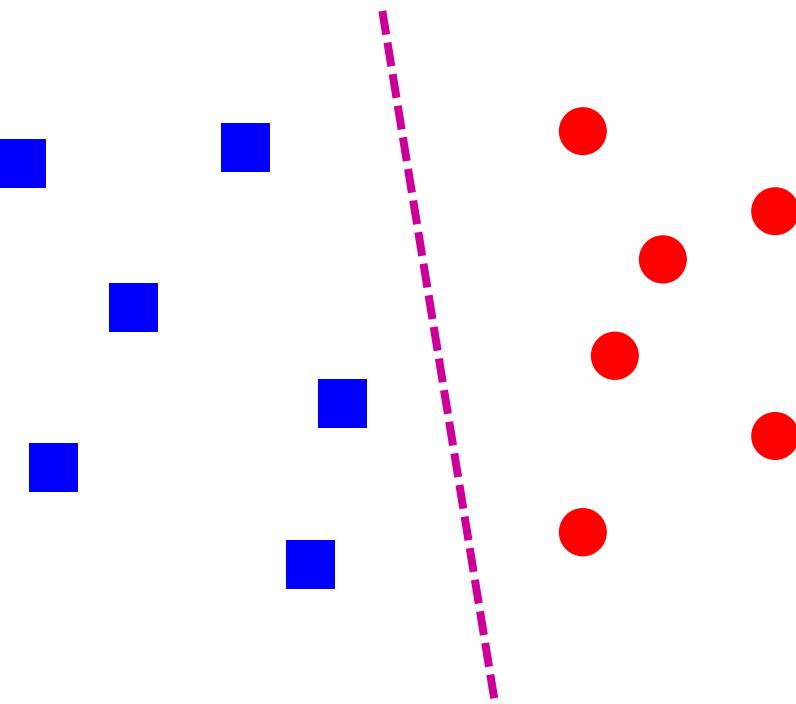
K-nearest neighbor classifier



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/>

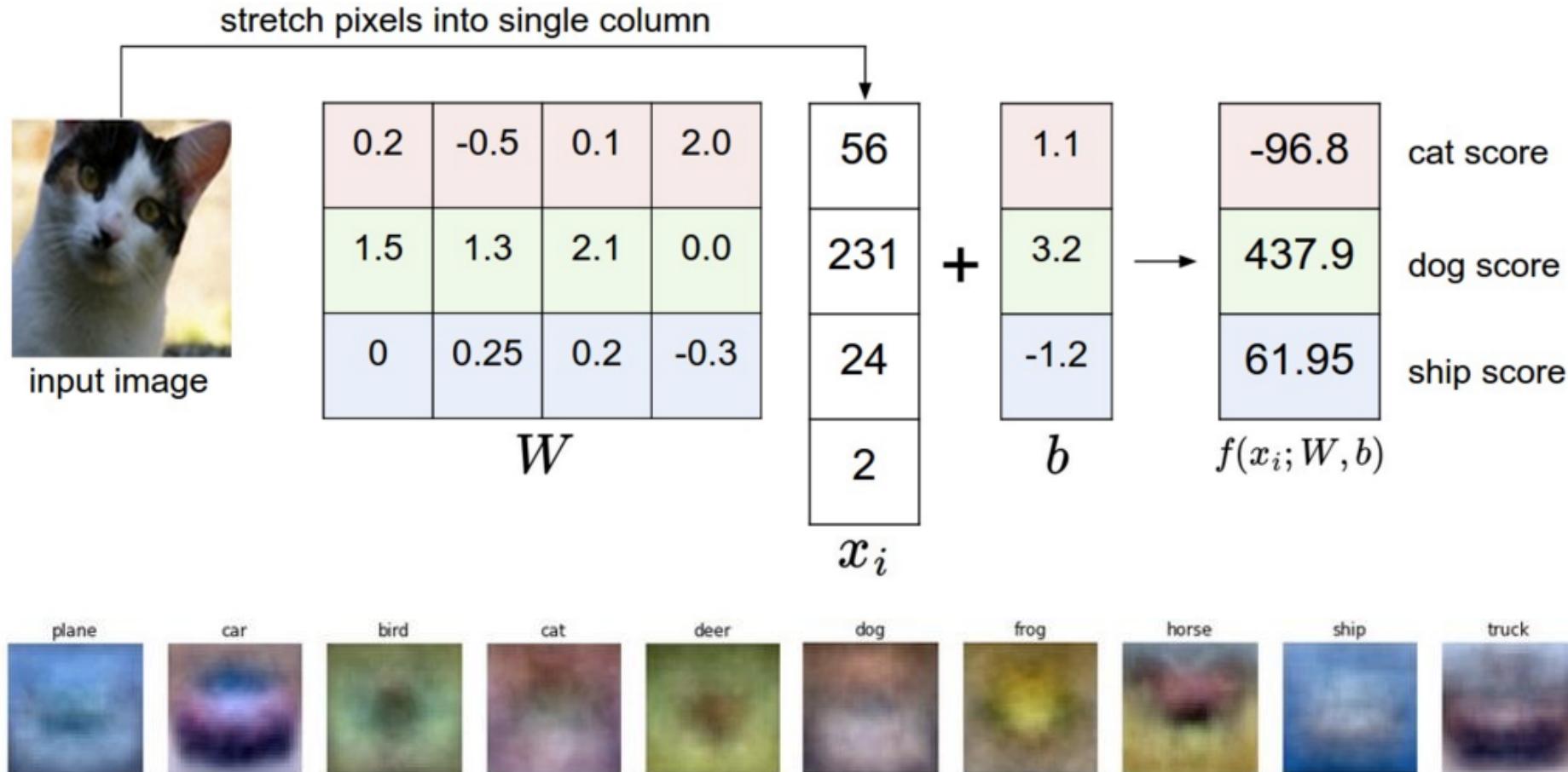
Linear classifiers



- Find a *linear function* to separate the classes:

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

Visualizing linear classifiers



Example learned weights at the end of learning for CIFAR-10.

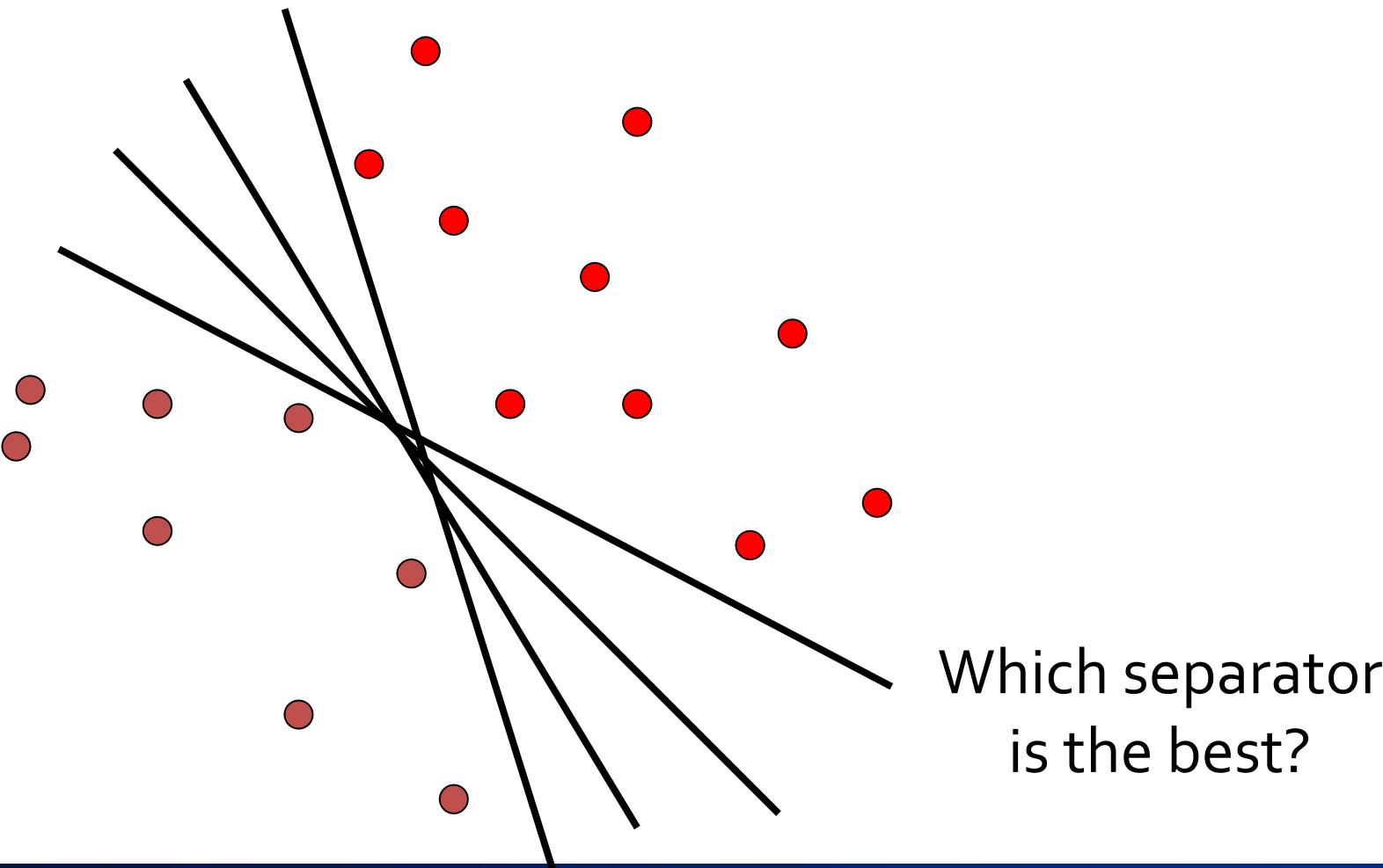
Credit: Andrej Karpathy, <http://cs231n.github.io/classification/>

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?

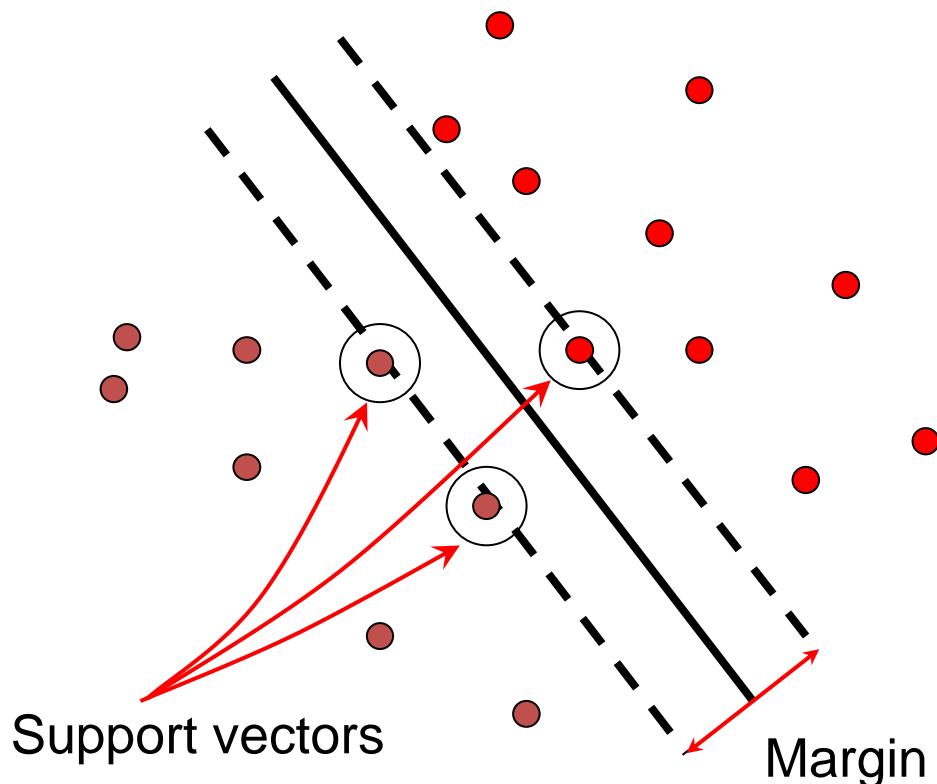
Linear classifiers

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



Support vector machines

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



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

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

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

Distance between point and hyperplane:

$$\frac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$$

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

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:*

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

SVM parameter learning

- Separable data: $\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$ subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$



Maximize margin



Classify training data correctly

- Non-separable data:

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



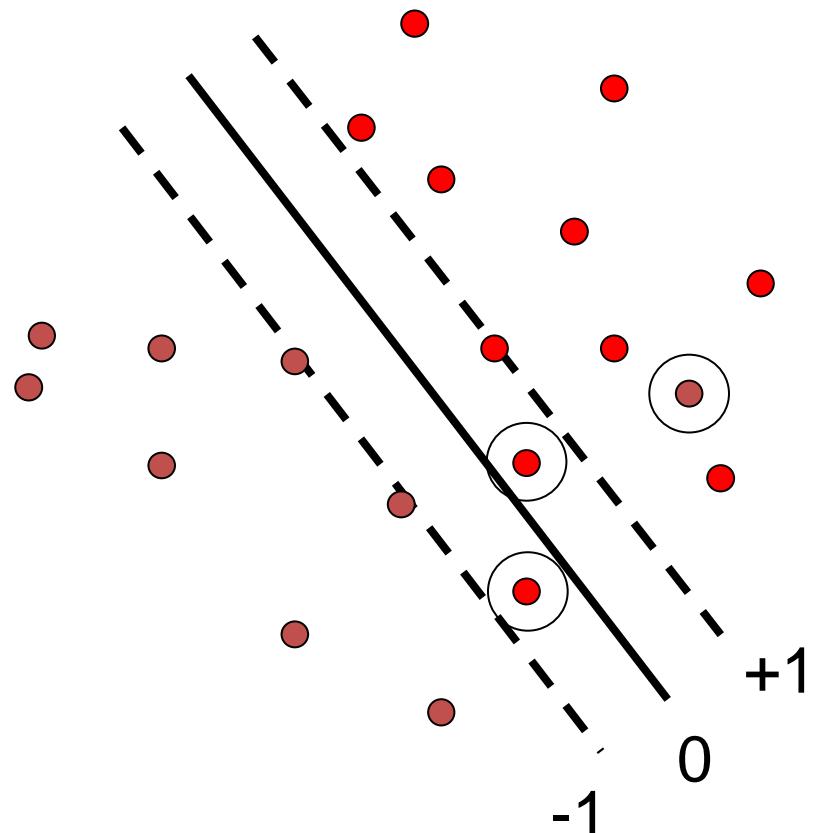
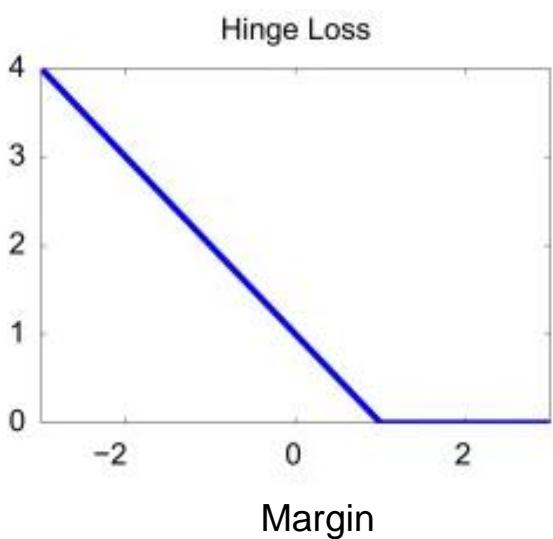
Maximize margin



Minimize classification mistakes

SVM parameter learning

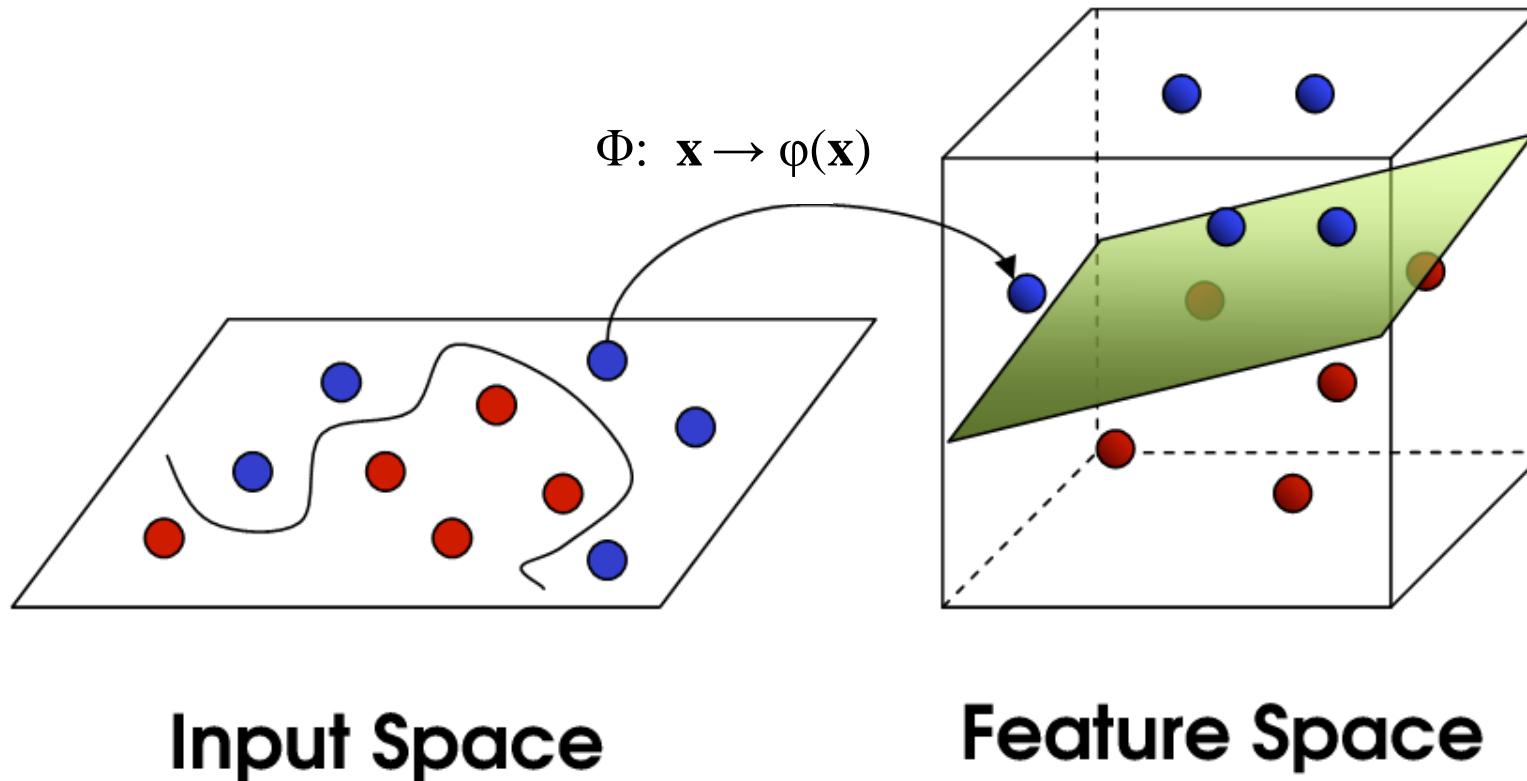
$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b))$$



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

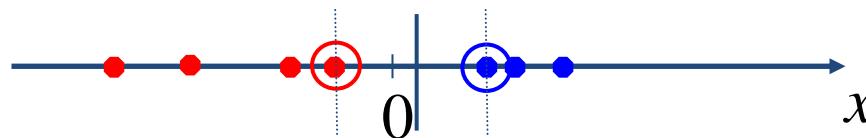
Nonlinear SVMs

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



Nonlinear SVMs

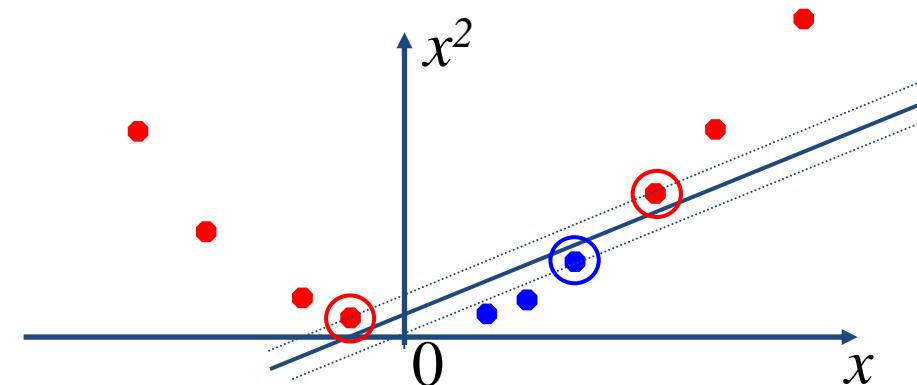
- Linearly separable dataset in 1D:



- Non-separable dataset in 1D:



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



SVMs: Pros and cons

- **Pros**

- Non-linear SVM framework is very powerful, flexible
- Training is convex optimization, globally optimal solution can be found
- 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)