Computer Vision Course
Lecture 09

Recognition 01

Ceyhun Burak Akgül, PhD
cba-research.com

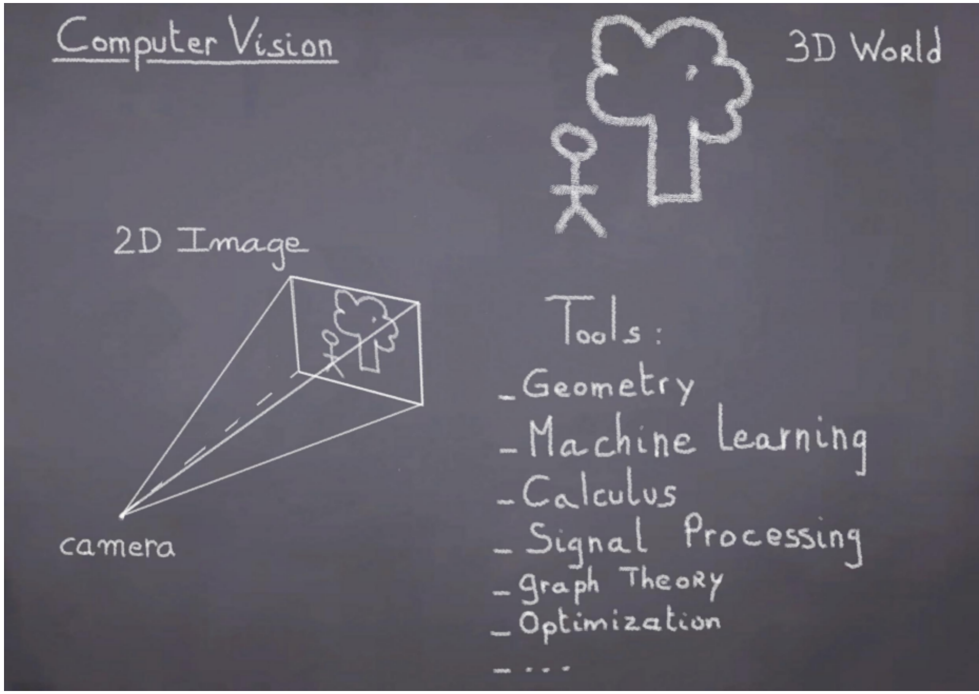
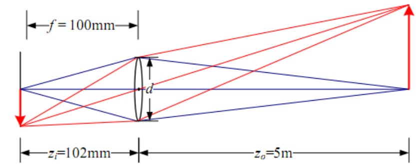


Photo credit: Olivier Teboul
vision.mas.ecp.fr/Personnel/teboul

Spring 2015
Last updated 29/04/2015

These slides have been adapted from James Hays's 2014 Computer Vision course slides at Brown University.

Course Outline



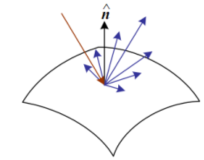
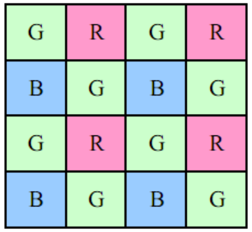


Image Formation and Processing

Light, Shape and Color

The Pin-hole Camera Model, The Digital Camera

Linear filtering, Template Matching, Image Pyramids



Feature Detection and Matching

Edge Detection, Interest Points: Corners and Blobs

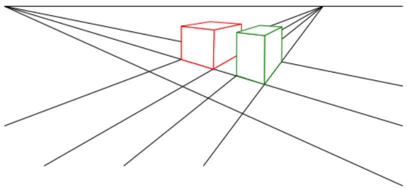
Local Image Descriptors

Feature Matching and Hough Transform

Multiple Views and Motion

Geometric Transformations, Camera Calibration

Feature Tracking , Stereo Vision



Segmentation and Grouping

Segmentation by Clustering, Region Merging and Growing

Advanced Methods Overview: Active Contours, Level-Sets, Graph-Theoretic Methods

Detection and Recognition

Problems and Architectures Overview

Statistical Classifiers, Bag-of-Words Model, Detection by Sliding Windows

Visual Recognition Problems

Object Instance Recognition

Recognize different instances of the same object (e.g., a product package, a face, a specific mug) given an image that tightly contains a single object

Object Category Recognition

Recognize different examples of the same object category (e.g., car, airplane, flower) given an image that tightly contains a single object

Object Detection and Localization

Do the above (instance or category) on an image containing the object at arbitrary position and scale

Image Classification

Classify an image based on its content (indoor/outdoor, nature/urban, sunny/cloudy/rainy, Paris/Istanbul/…, etc.)

Scene Understanding

Tell what is going in the image, e.g., “a car running on the high way at sunset, it’s summer time, …”

Instance vs. Category

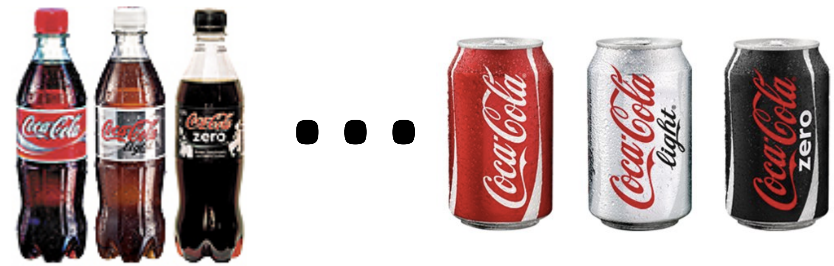
Instance

Coca Cola 1lt Pet Bottle

A Specific Mug

Category

Coca Cola Products



Mugs

Instance vs. Category

Instance

A Specific Ferrari

Ceyhun’s Face



Category

Sports Cars



• • •

Human Faces

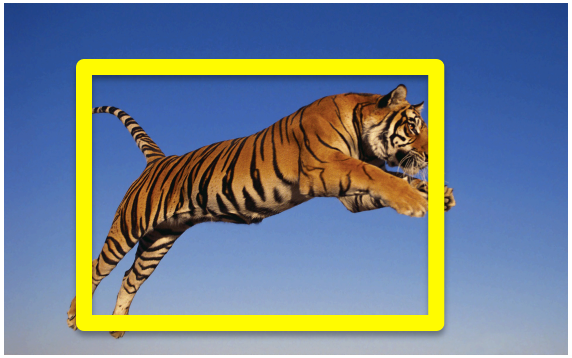

• • •

Recognition vs. Detection/Localization

Recognition

"There is a tiger in the image"

Detection/Localization



"There is a tiger in the image
at that particular position in the image"

Architectures

- Aligned Representations
- Voting Schemes: Generalized Hough Transform
- Bag-of-Words Model
- Detection by Sliding Windows
- Parts-based Models

Architectures

- **Aligned Representations**
- Voting Schemes: Generalized Hough Transform – *seen*
- Bag-of-Words Model
- Detection by Sliding Windows
- Parts-based Models – *not in this class*

Aligned Representations

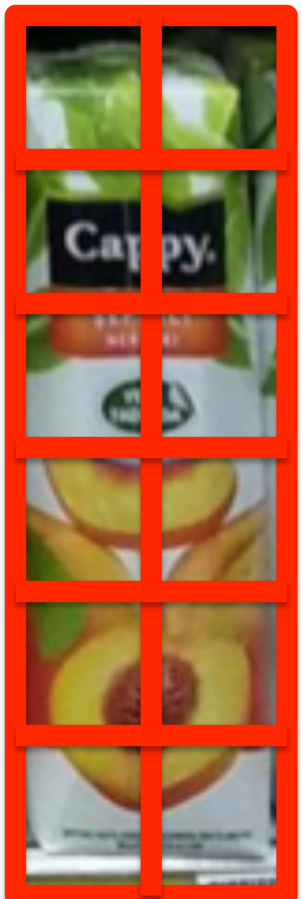
Images of objects of interests are roughly aligned. No detection necessary!

Place a grid on the image, extract a visual descriptor from each cell

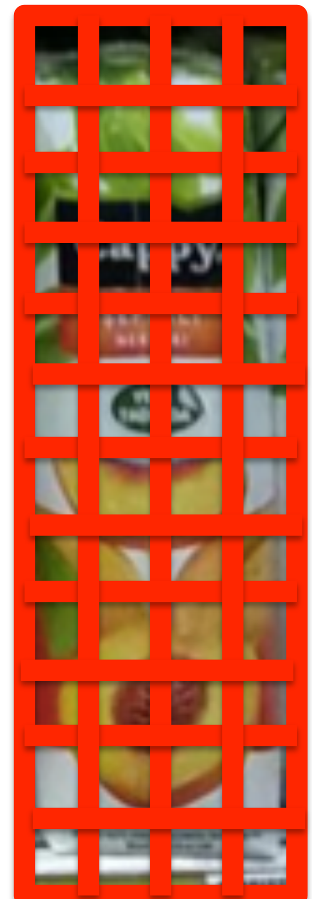
Each cell is indexed and can be compared directly with its corresponding cell in another image.

Or all cell descriptors can be compared into one global descriptor

Aligned Representations

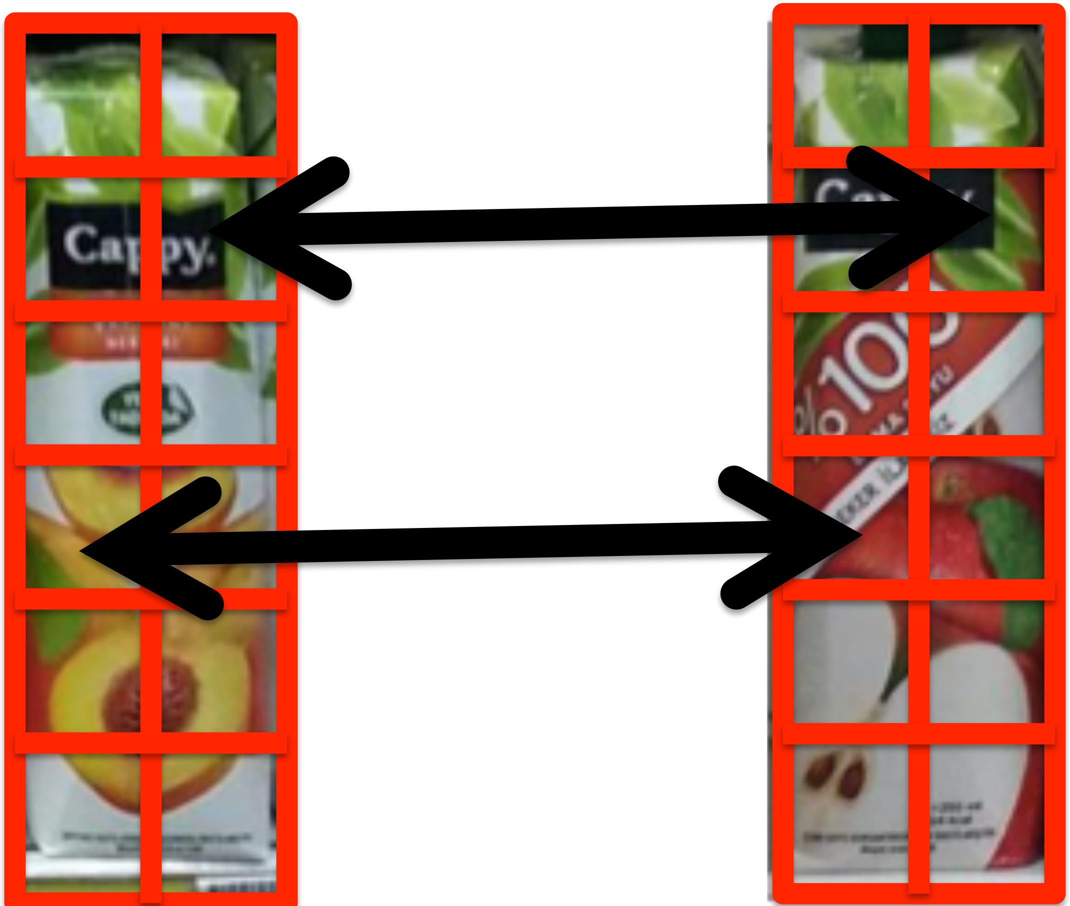


6x2 grid



12x4 grid

Aligned Representations



Cells are already in correspondence!

A Simple Instance Recognizer

Cappy Peach Bottle

Coca Cola Can

…

Cappy Peach Tetra

Extract Descriptors

$$\begin{pmatrix} f_1^{A(1)} \\ f_2^{A(1)} \\ \vdots \\ f_n^{A(1)} \end{pmatrix} \cdots \begin{pmatrix} f_1^{A(N_A)} \\ f_2^{A(N_A)} \\ \vdots \\ f_n^{A(N_A)} \end{pmatrix}$$

$$\cdots$$

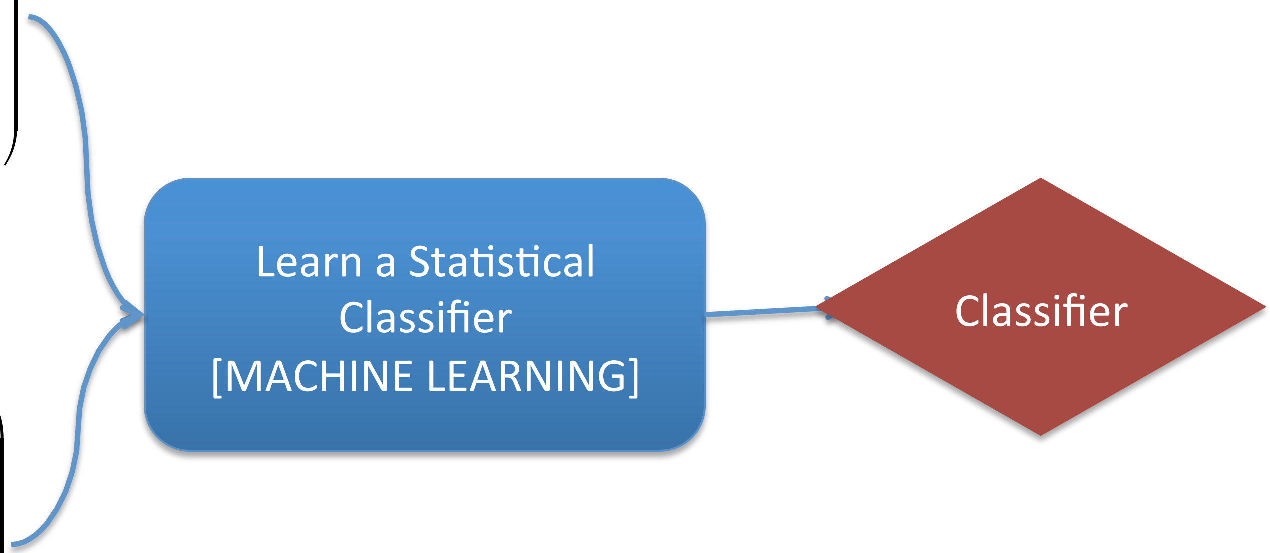
$$\begin{pmatrix} f_1^{Z(1)} \\ f_2^{Z(1)} \\ \vdots \\ f_n^{Z(1)} \end{pmatrix} \cdots \begin{pmatrix} f_1^{Z(N_Z)} \\ f_2^{Z(N_Z)} \\ \vdots \\ f_n^{Z(N_Z)} \end{pmatrix}$$

A Simple Instance Recognizer

$$\begin{pmatrix} f_1^{A(1)} \\ f_2^{A(1)} \\ \vdots \\ f_n^{A(1)} \end{pmatrix} \cdots \begin{pmatrix} f_1^{A(N_A)} \\ f_2^{A(N_A)} \\ \vdots \\ f_n^{A(N_A)} \end{pmatrix}$$

$$\cdots$$

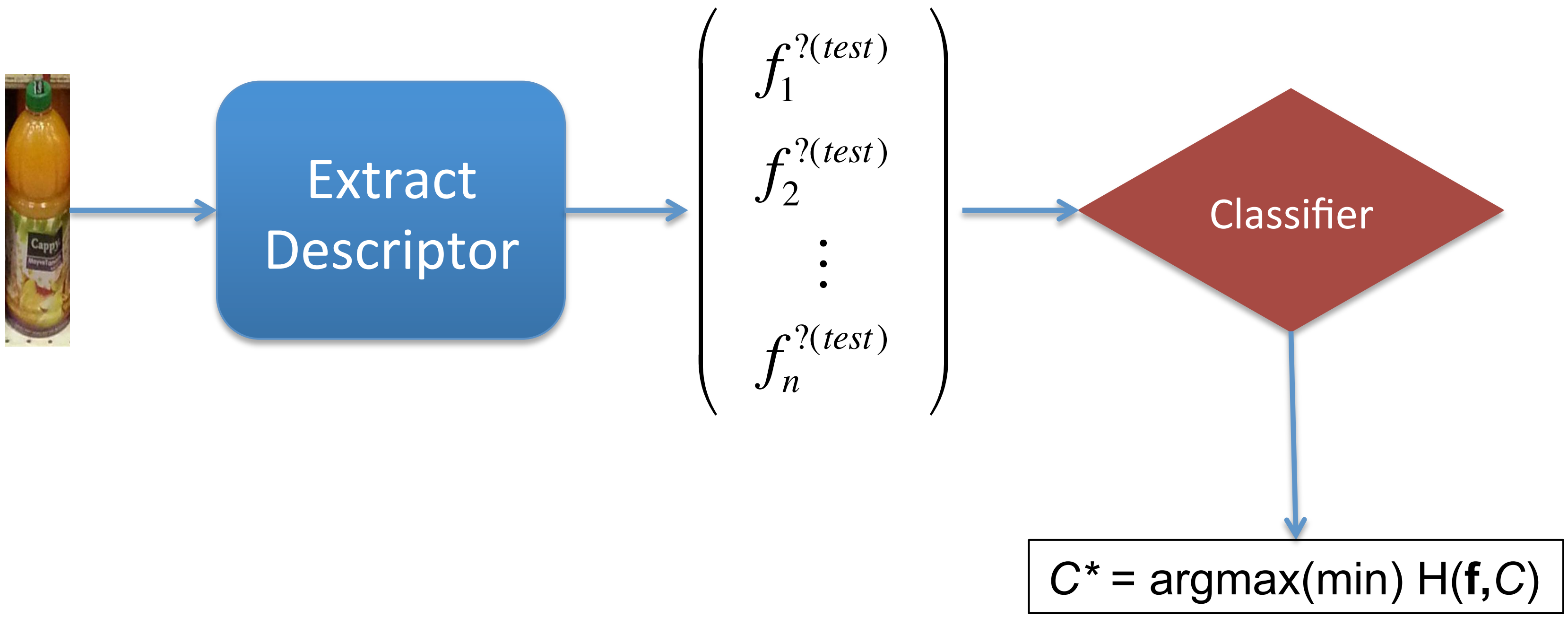
$$\begin{pmatrix} f_1^{Z(1)} \\ f_2^{Z(1)} \\ \vdots \\ f_n^{Z(1)} \end{pmatrix} \cdots \begin{pmatrix} f_1^{Z(N_Z)} \\ f_2^{Z(N_Z)} \\ \vdots \\ f_n^{Z(N_Z)} \end{pmatrix}$$



The learned classifier is a mathematical function H(**f**,*C*) that takes a descriptor vector as input.

The function tests the input and depending on the function value, it assigns the descriptor into one of the trained classes.

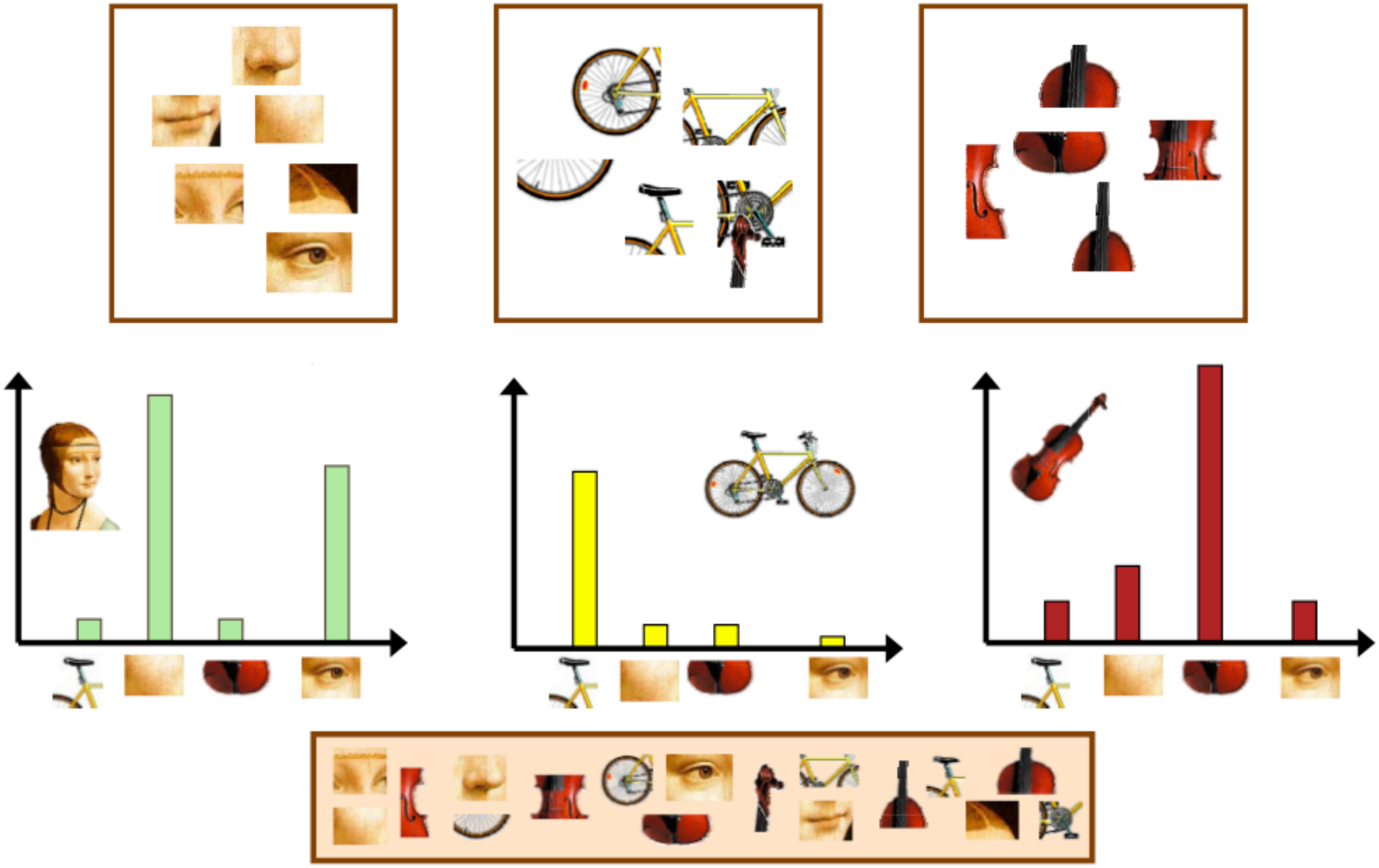
A Simple Instance Recognizer



Bag-of-Words Model

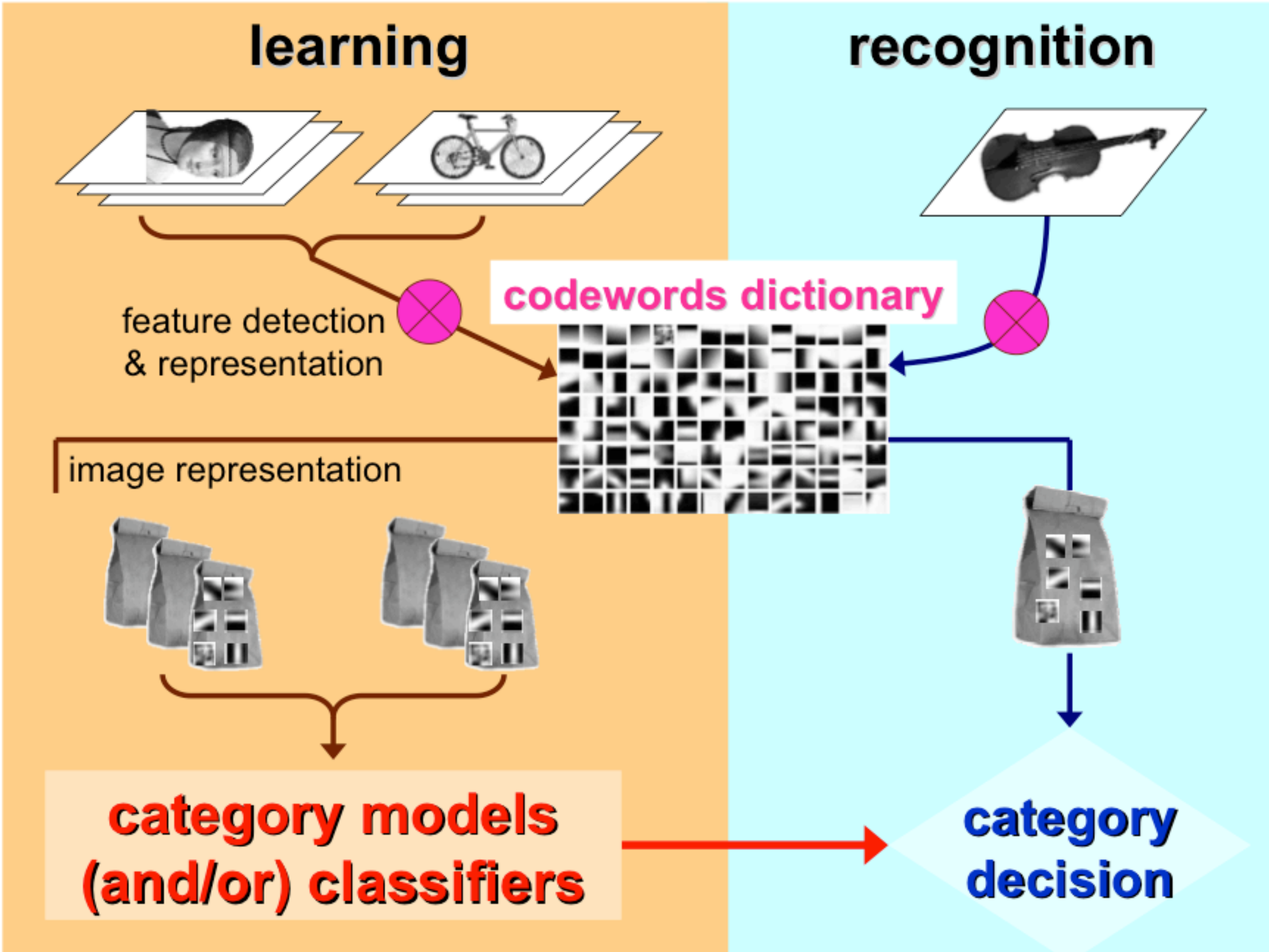
*Picture credits: Li Fei-Fei, Princeton University

Bag-of-Words Model



*Picture credits: Li Fei-Fei, Princeton University

Bag-of-Words Model

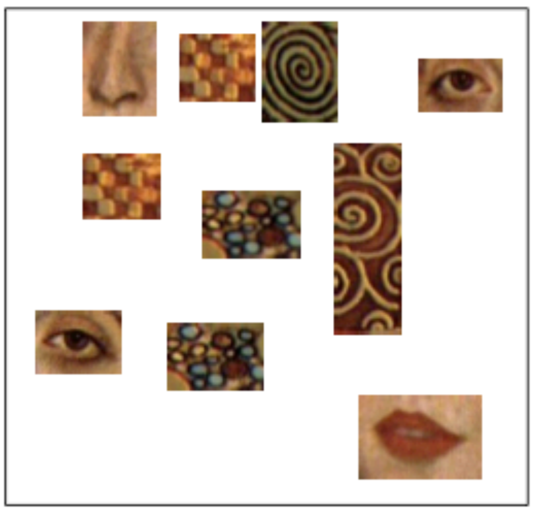


*Slide credits: Li Fei-Fei, Princeton University

Bag-of-Words Model

Limitations

- All the images below have the same representation
- BoW not good when location matters



*Picture credits: Li Fei-Fei, Princeton University

Machine Learning Problems

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Clustering: group together similar points and represent them with a single token

Key Challenges:

1) What makes two points/images/patches similar?

2) How do we compute an overall grouping from pairwise similarities?

How do we cluster?

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Mean-shift clustering
  - Estimate modes of pdf
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights

Clustering for Summarization

Goal: cluster to minimize variance in data given clusters

– Preserve information

Cluster center

Data

$$\mathbf{c}^*, \boldsymbol{\delta}^* = \underset{\mathbf{c},\boldsymbol{\delta}}{\operatorname{argmin}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} \left(\mathbf{c}_i - \mathbf{x}_j\right)^2$$

Whether x_j is assigned to c_i

Slide: Derek Hoiem

K-means algorithm

1. Randomly select K centers

2. Assign each point to nearest center

3. Compute new center (mean) for each cluster

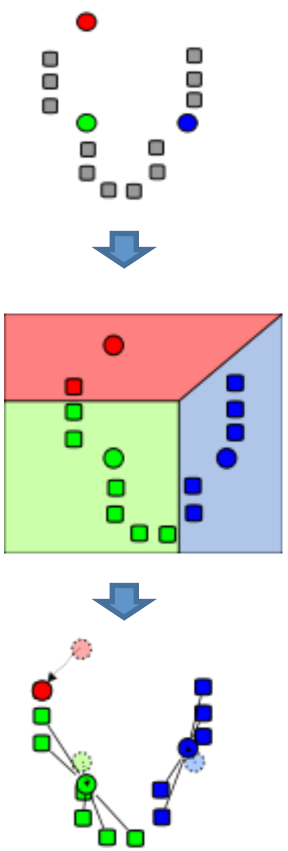
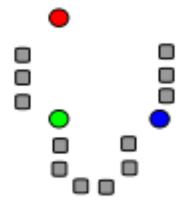


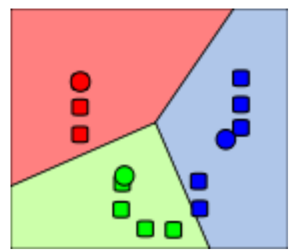
Illustration: http://en.wikipedia.org/wiki/K-means_clustering

K-means algorithm

1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster

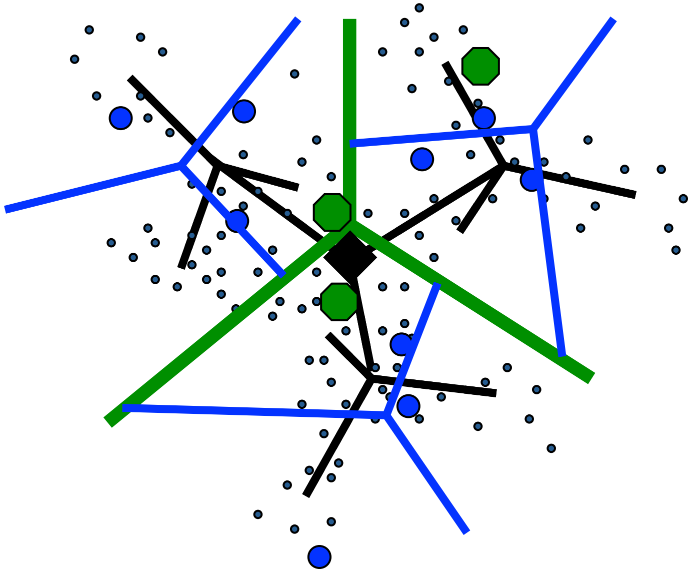
Back to 2

Illustration: http://en.wikipedia.org/wiki/K-means_clustering

Building Visual Dictionaries

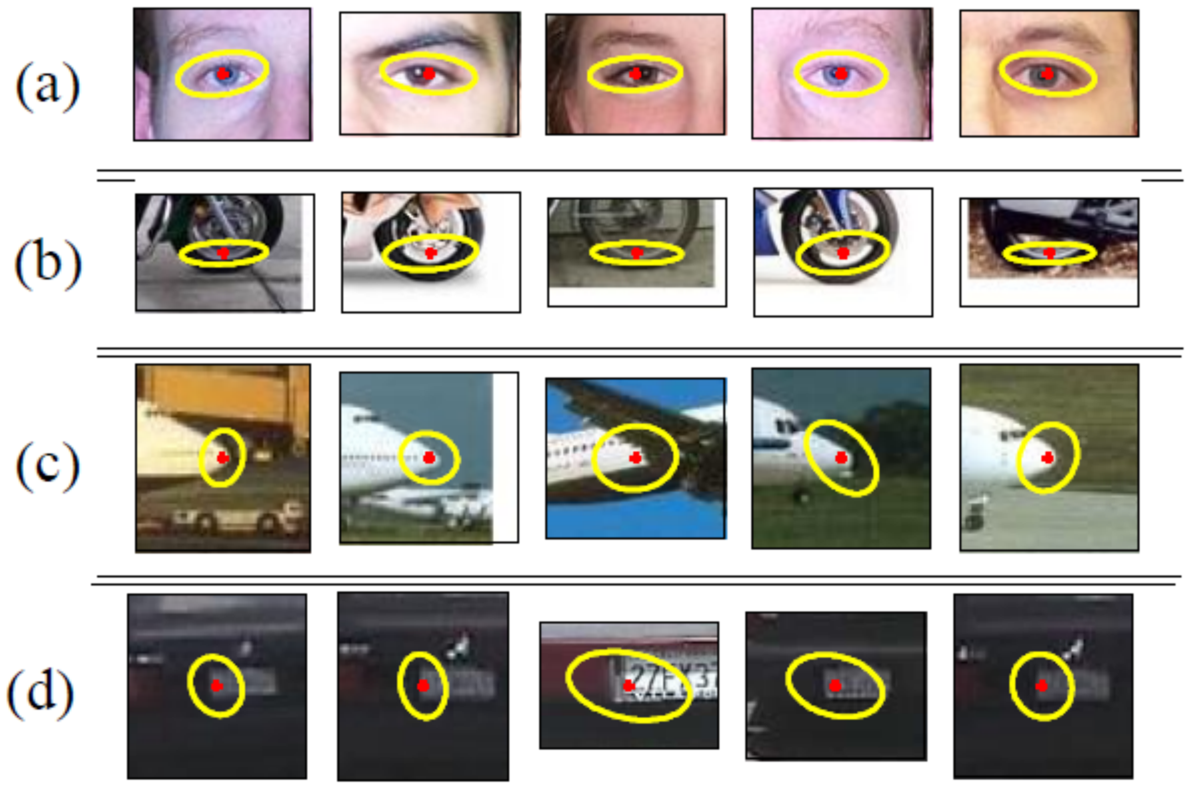
1. Sample patches from a database
   – E.g., 128 dimensional SIFT vectors

2. Cluster the patches
   – Cluster centers are the dictionary



3. Assign a codeword (number) to each new patch, according to the nearest cluster

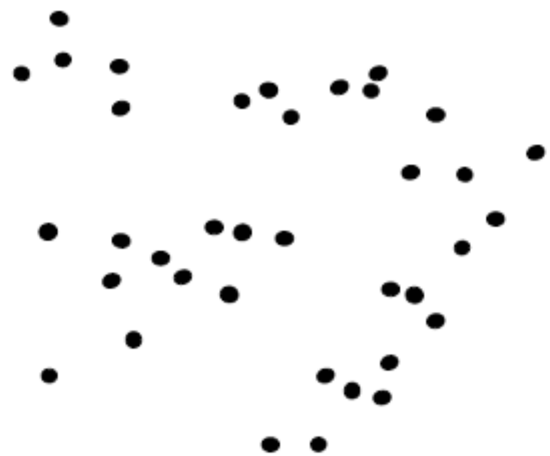
Examples of learned codewords



Most likely codewords for 4 learned “topics”
EM with multinomial (problem 3) to get topics

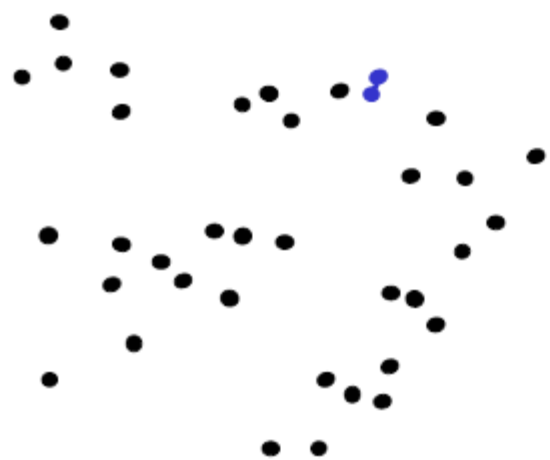
http://www.robots.ox.ac.uk/~vgg/publications/papers/sivic05b.pdf Sivic et al. ICCV 2005

Agglomerative clustering



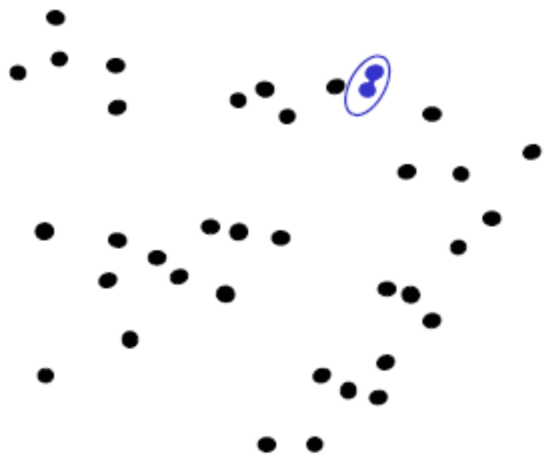
1. Say "Every point is its own cluster"

Agglomerative clustering



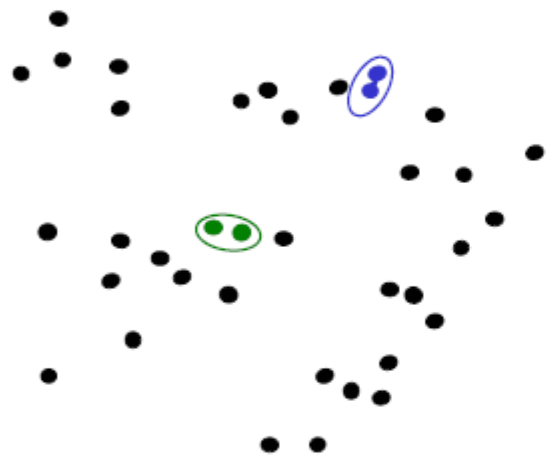
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters

Agglomerative clustering



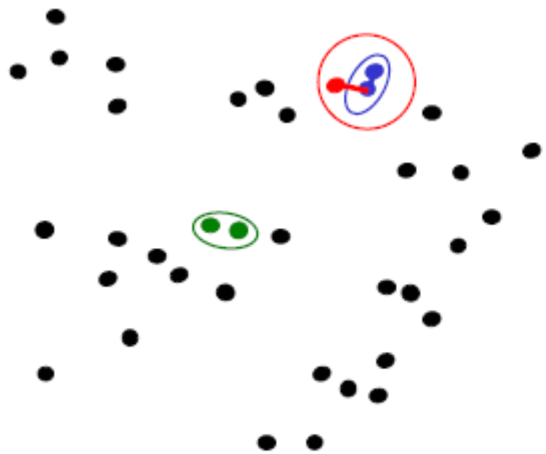
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster

Agglomerative clustering



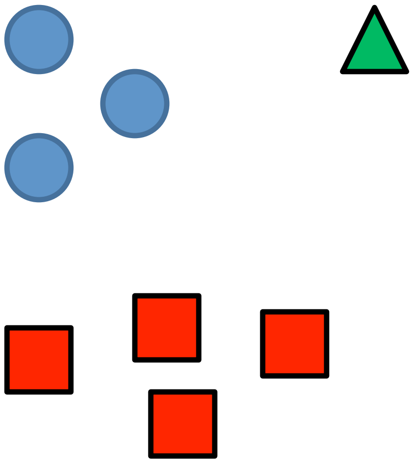
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Agglomerative clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Agglomerative clustering

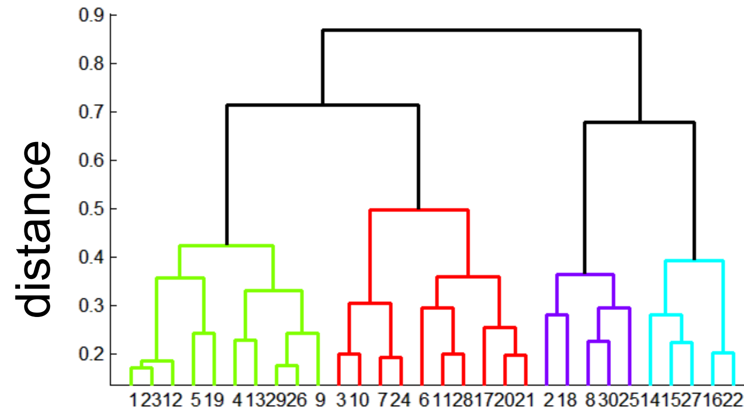


How to define cluster similarity?

- Average distance between points, maximum distance, minimum distance
- Distance between means or medoids

How many clusters?

- Clustering creates a dendrogram (a tree)
- Threshold based on max number of clusters or based on distance between merges



Machine Learning Problems

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

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

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

output

prediction function

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

Slide credit: L. Lazebnik

Steps

Training

Training Images

Image Features → Training → Learned model

Training Labels → Training

Testing

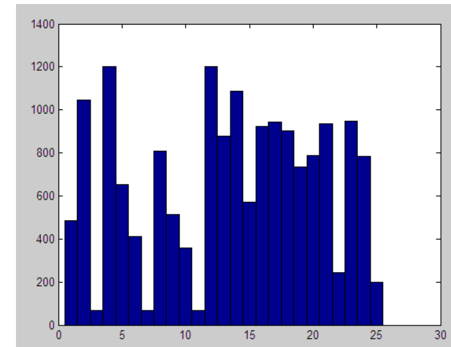
Test Image

Image Features → Learned model → Prediction

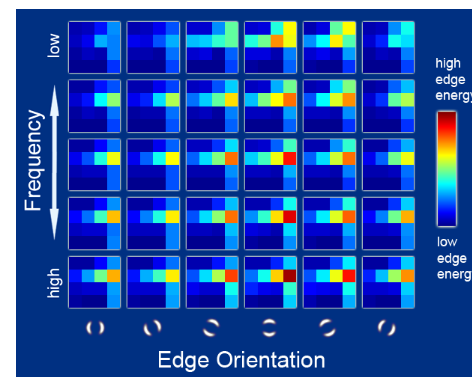
Slide credit: D. Hoiem and L. Lazebnik

Features

- Raw pixels
- Histograms



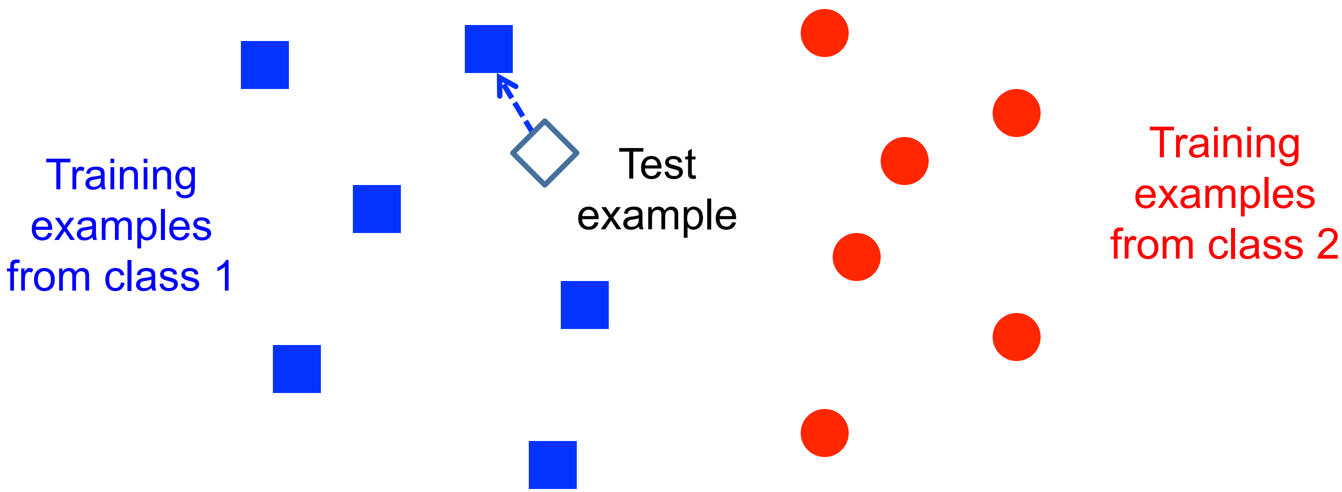
- SIFT descriptors



- …

Slide credit: L. Lazebnik

Classifiers: Nearest neighbor

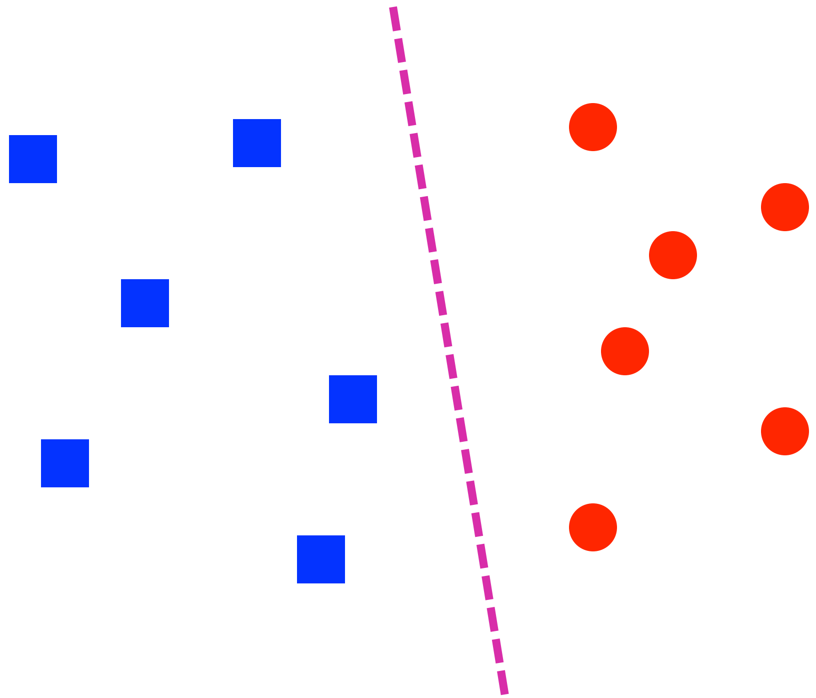


f(**x**) = label of the training example nearest to **x**

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

Slide credit: L. Lazebnik

Classifiers: Linear



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

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

Slide credit: L. Lazebnik

Many classifiers to choose from

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

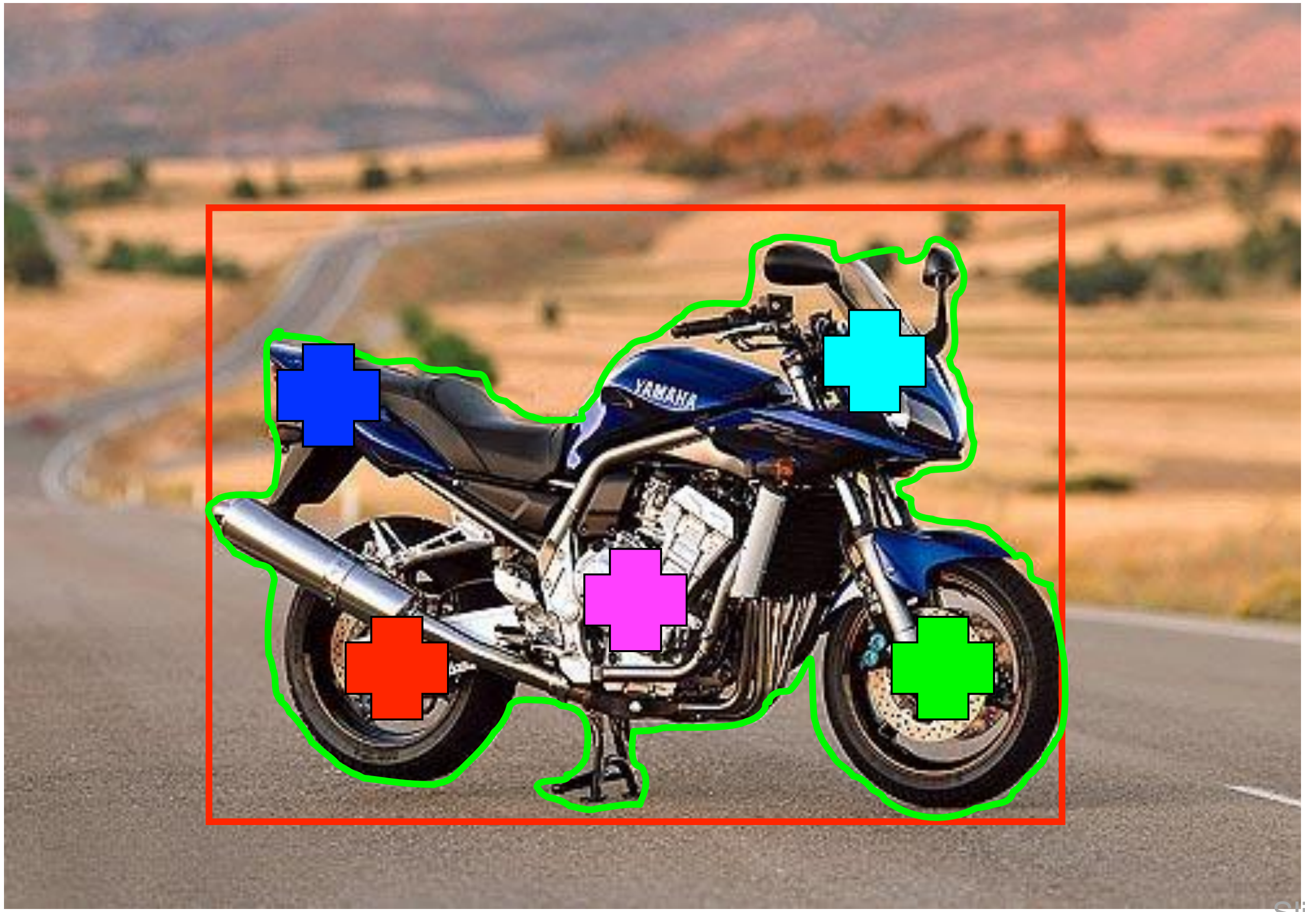
Which is the best one?

Slide credit: D. Hoiem

Recognition task and supervision

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

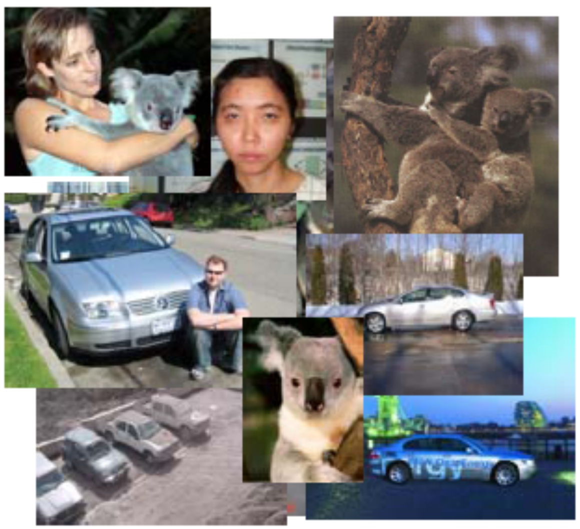
Contains a motorbike



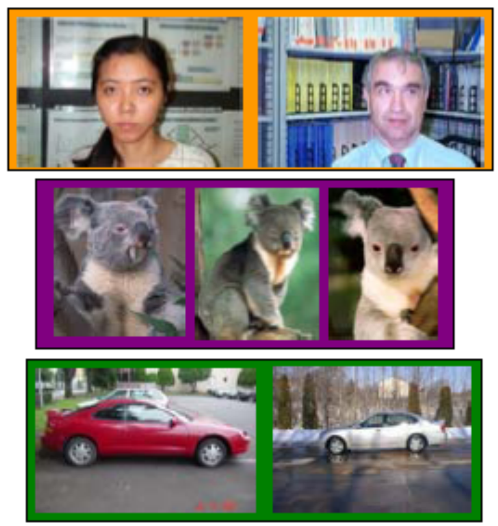
Slide credit: L. Lazebnik

Spectrum of supervision

Less **More**



Unsupervised



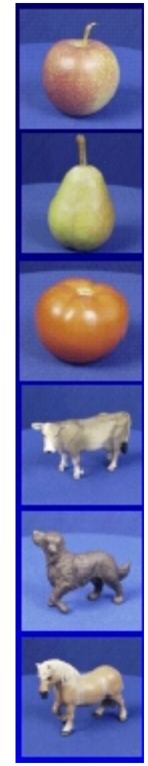
“Weakly” supervised

Fully supervised

Definition depends on task

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?

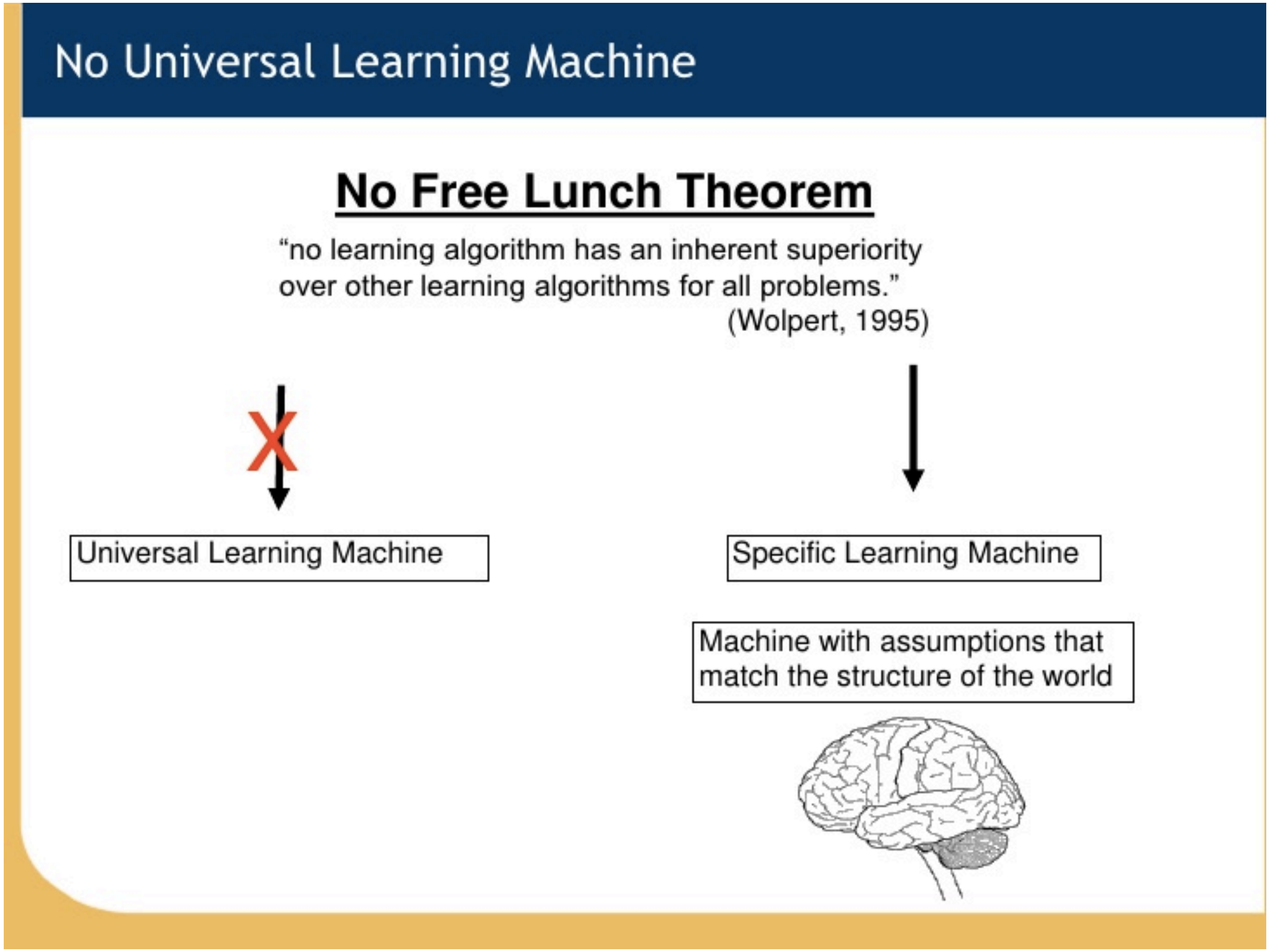
Slide credit: L. Lazebnik

Generalization

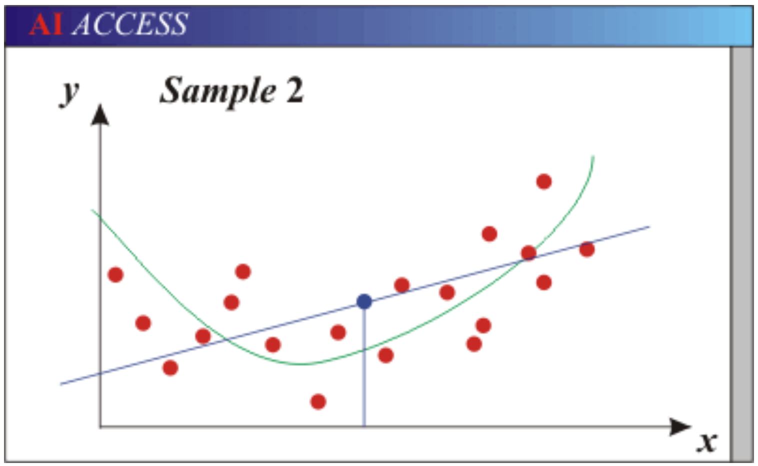
- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - **Variance:** how much models estimated from different training sets differ from each other
- **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

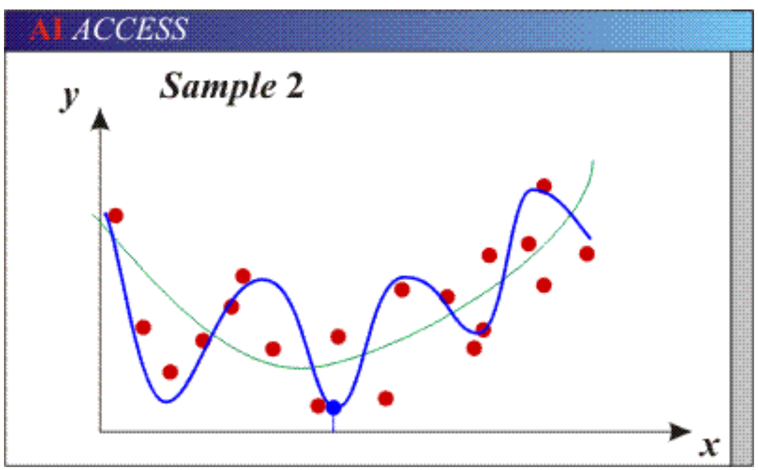
Slide credit: L. Lazebnik

No Free Lunch Theorem



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-Variance Trade-off

$$E(MSE) = noise^2 + bias^2 + variance$$

Unavoidable error

Error due to incorrect assumptions

Error due to variance of training samples

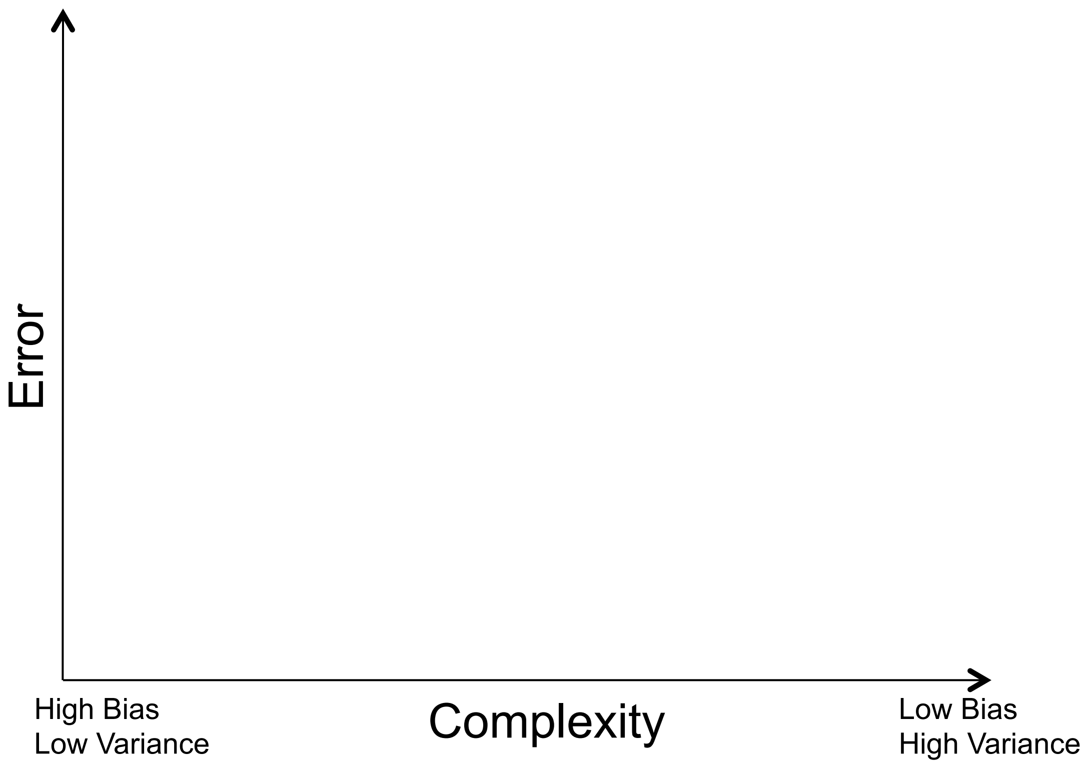
See the following for explanations of bias-variance (also Bishop's "Neural Networks" book):

- http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

Slide credit: D. Hoiem

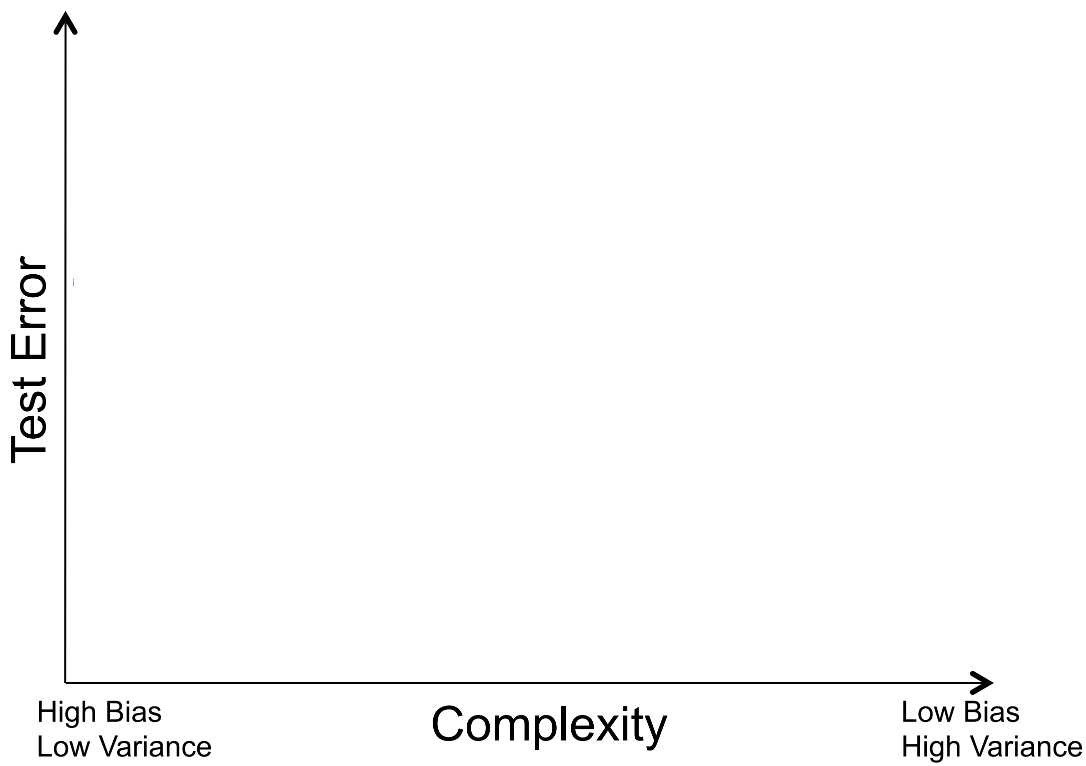
Bias-variance tradeoff

Underfitting **Overfitting**



Slide credit: D. Hoiem

Bias-variance tradeoff



Slide credit: D. Hoiem

The perfect classification algorithm

- Objective function: encodes the right loss for the problem
- Parameterization: makes assumptions that fit the problem
- Regularization: right level of regularization for amount of training data
- Training algorithm: can find parameters that maximize objective on training set
- Inference algorithm: can solve for objective function in evaluation

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

Very brief tour of some classifiers

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

Generative vs. Discriminative Classifiers

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

Discriminative Models

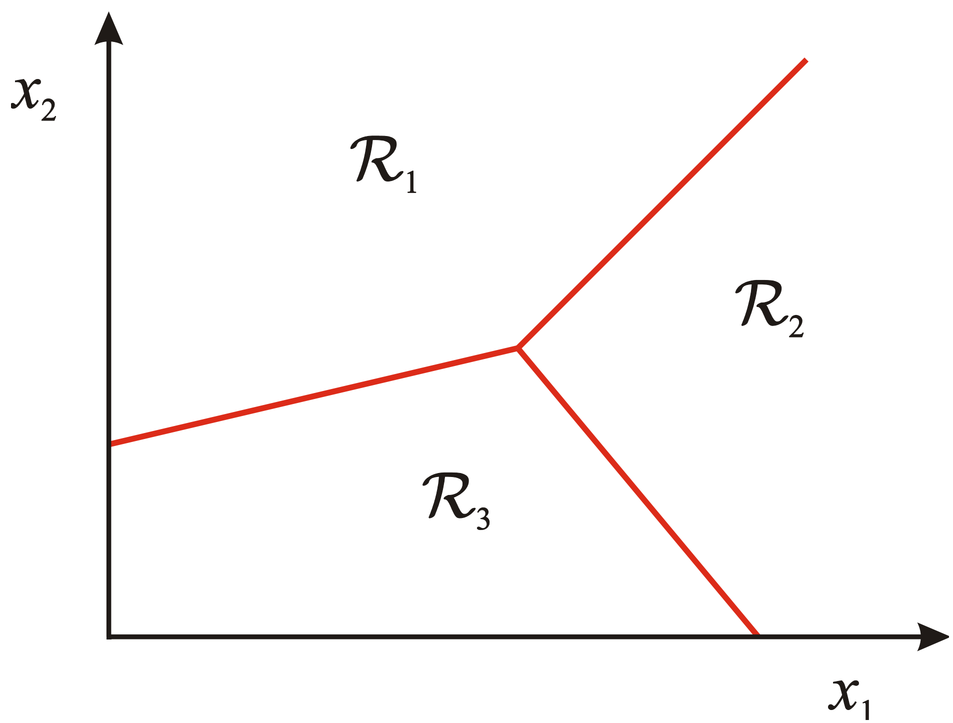
- Learn to directly predict the labels from the data
- Often, assume a simple boundary (e.g., linear)
- Examples
  - Logistic regression
  - SVM
  - Boosted decision trees

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

Slide credit: D. Hoiem

Classification

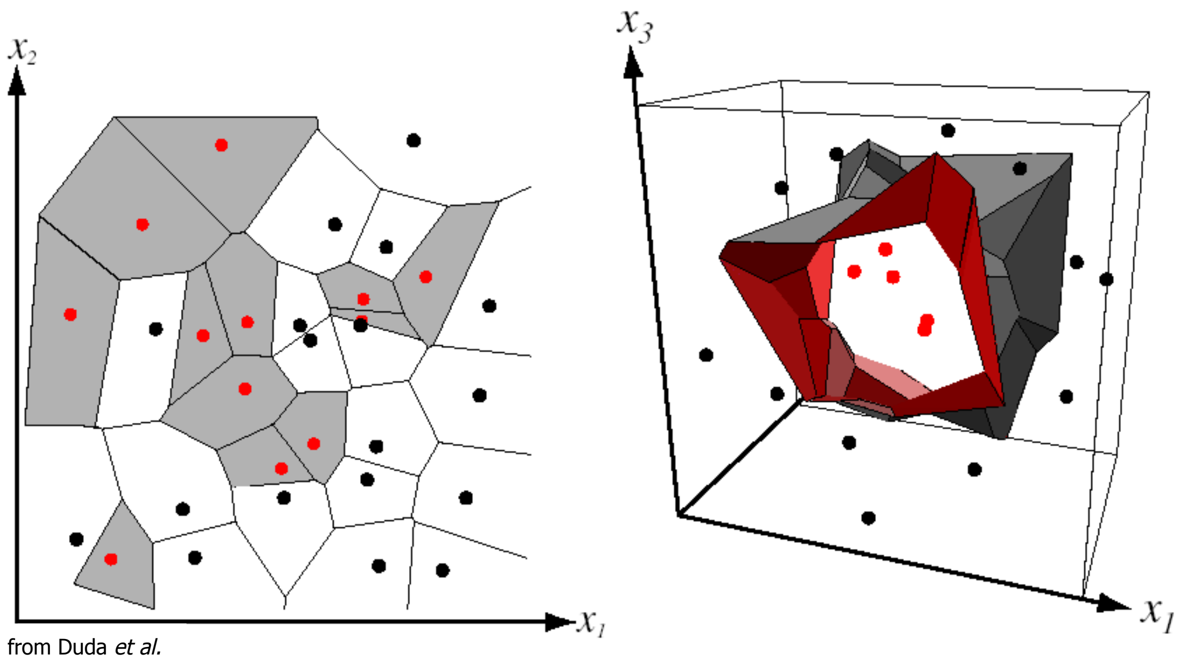
- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



Slide credit: L. Lazebnik

Nearest Neighbor Classifier

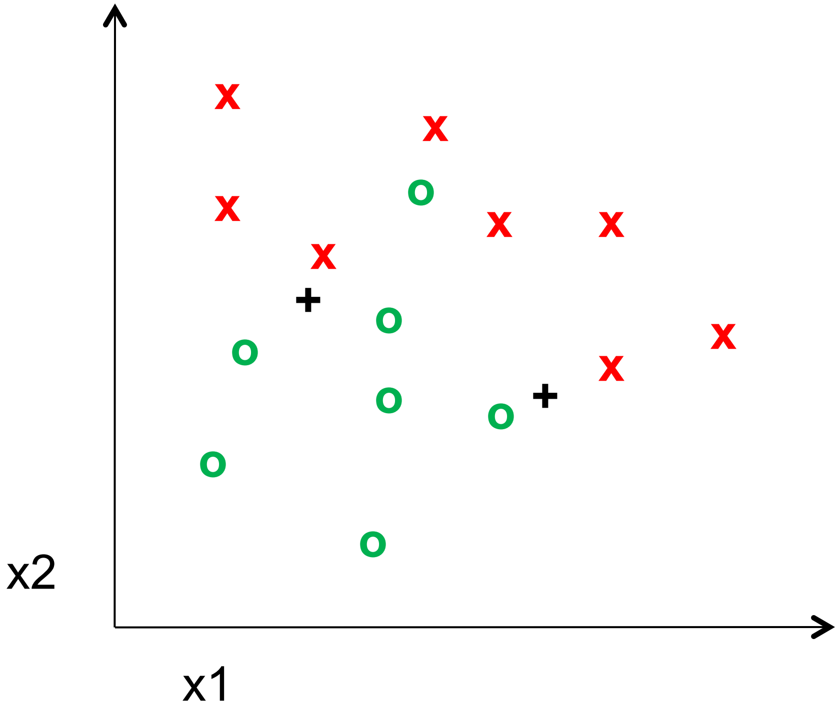
- Assign label of nearest training data point to each test data point



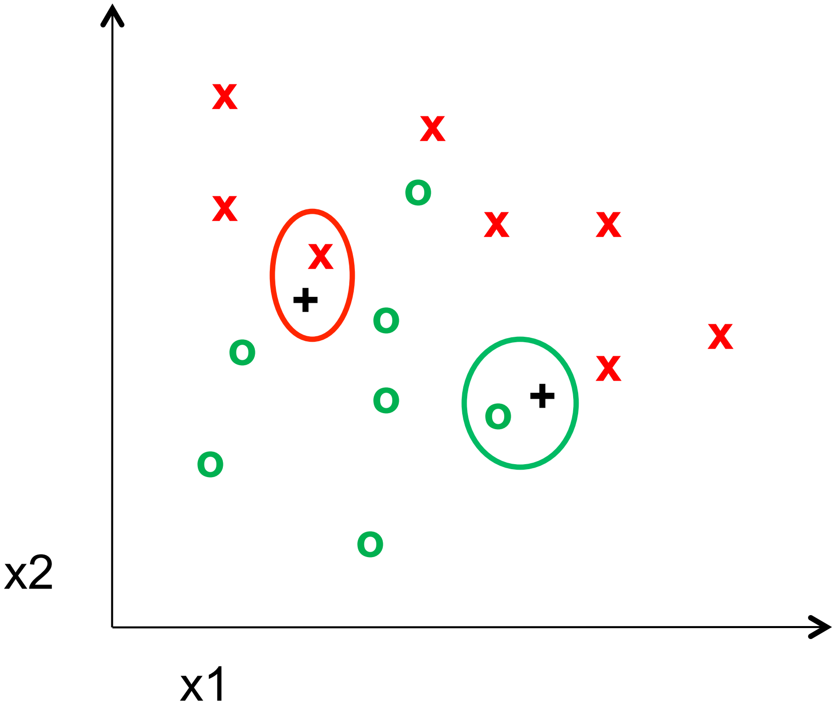
Voronoi partitioning of feature space
for two-category 2D and 3D data

Source: D. Lowe

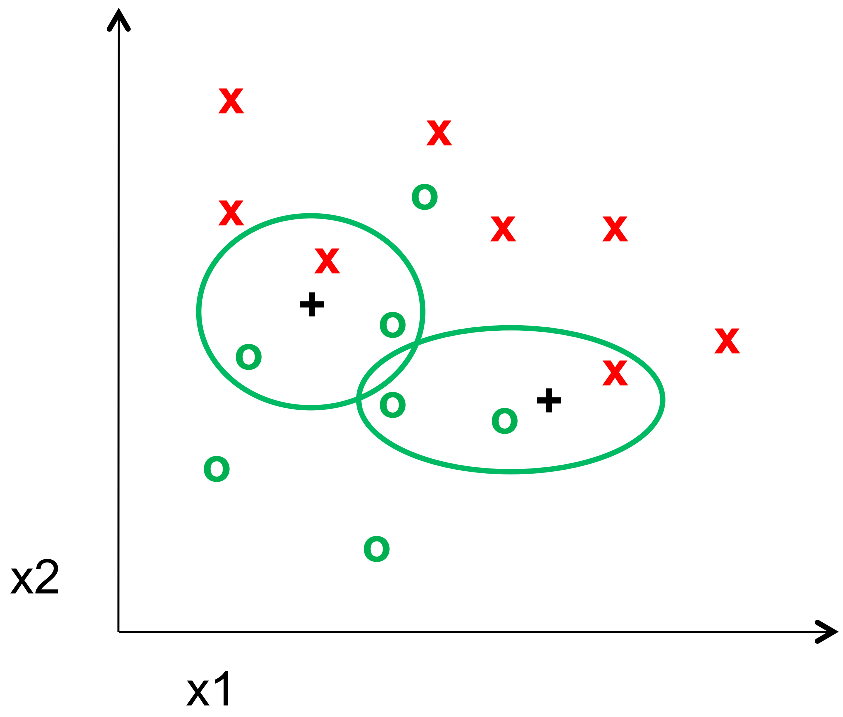
K-nearest neighbor



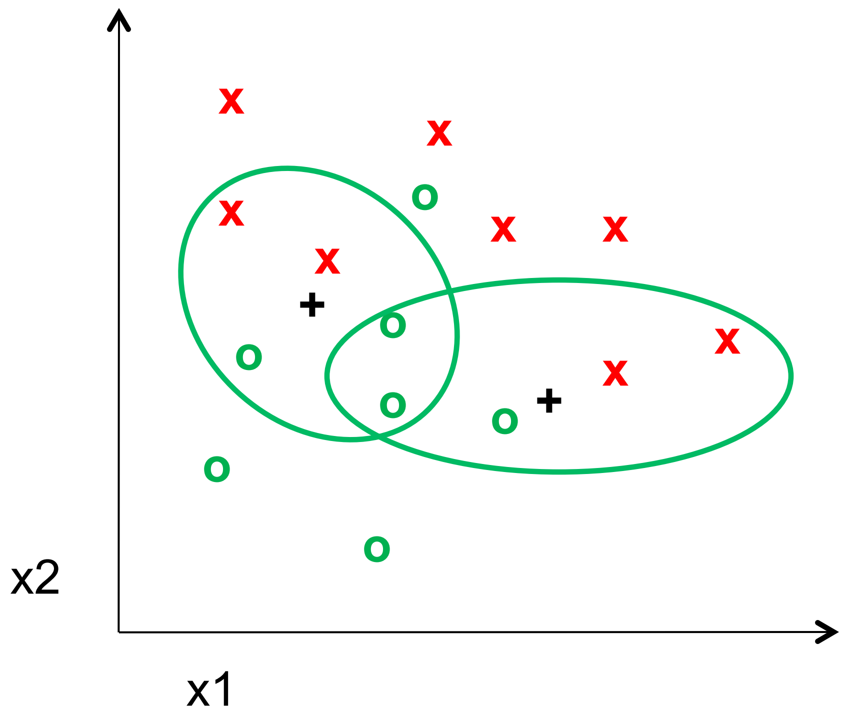
1-nearest neighbor



3-nearest neighbor



5-nearest neighbor

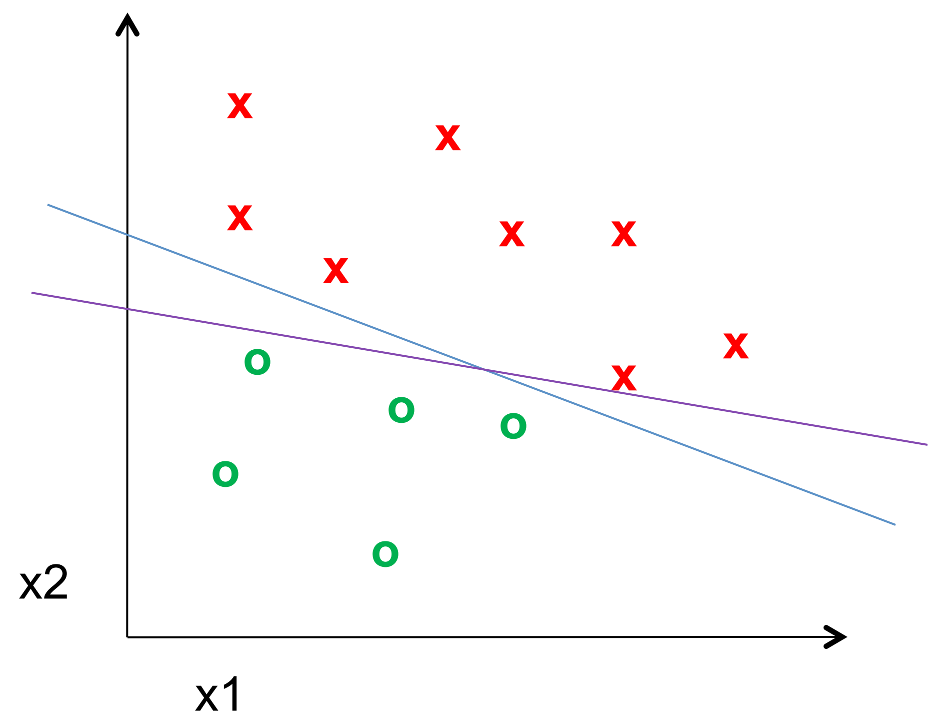


Using K-NN

- Simple, a good one to try first

- With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error

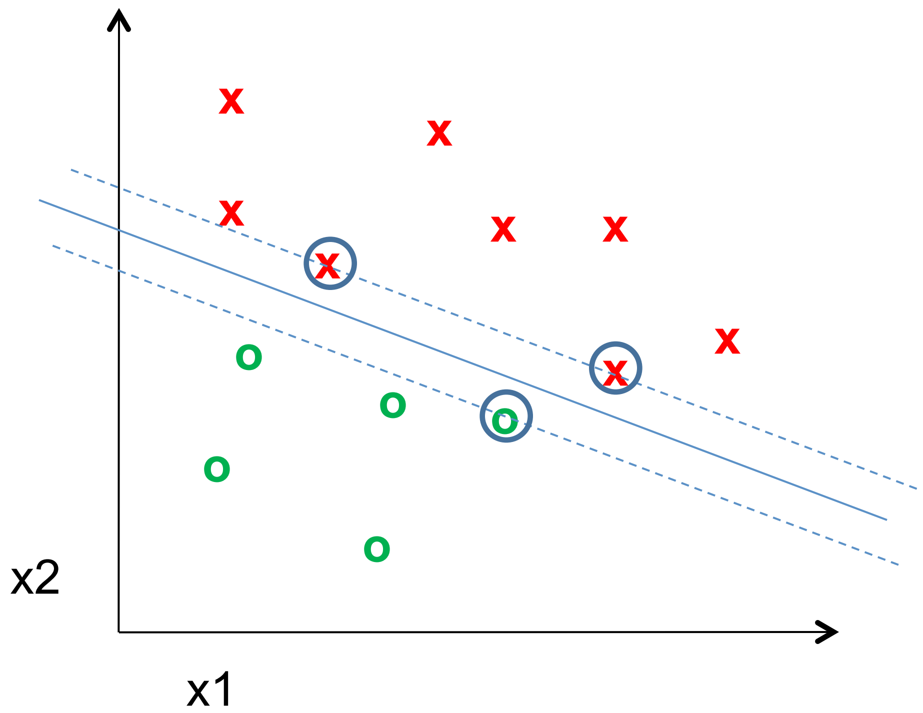
Classifiers: Linear SVM



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

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

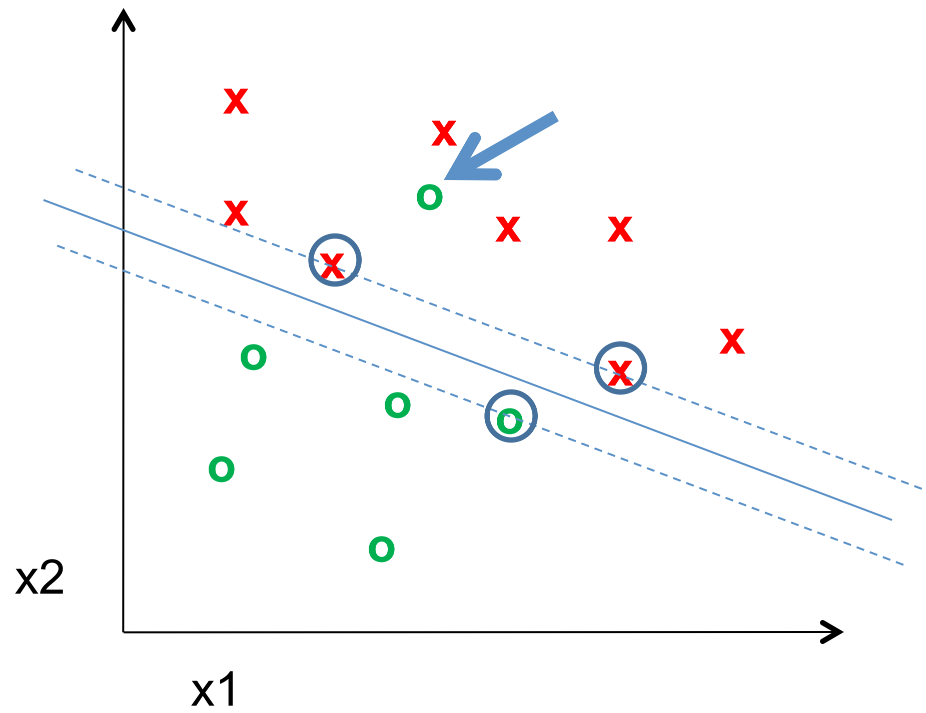
Classifiers: Linear SVM



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

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

Classifiers: Linear SVM

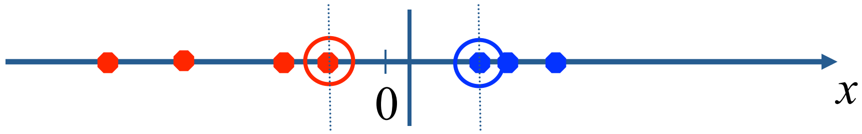


- Find a *linear function* to separate the classes:

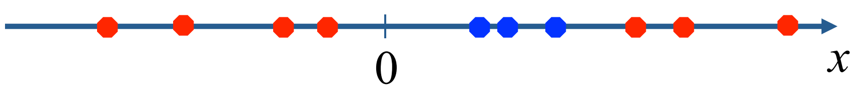
$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Nonlinear SVMs

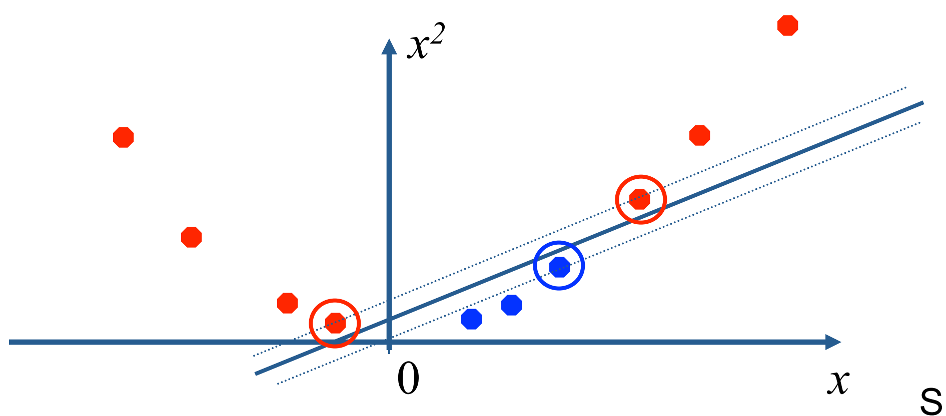
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



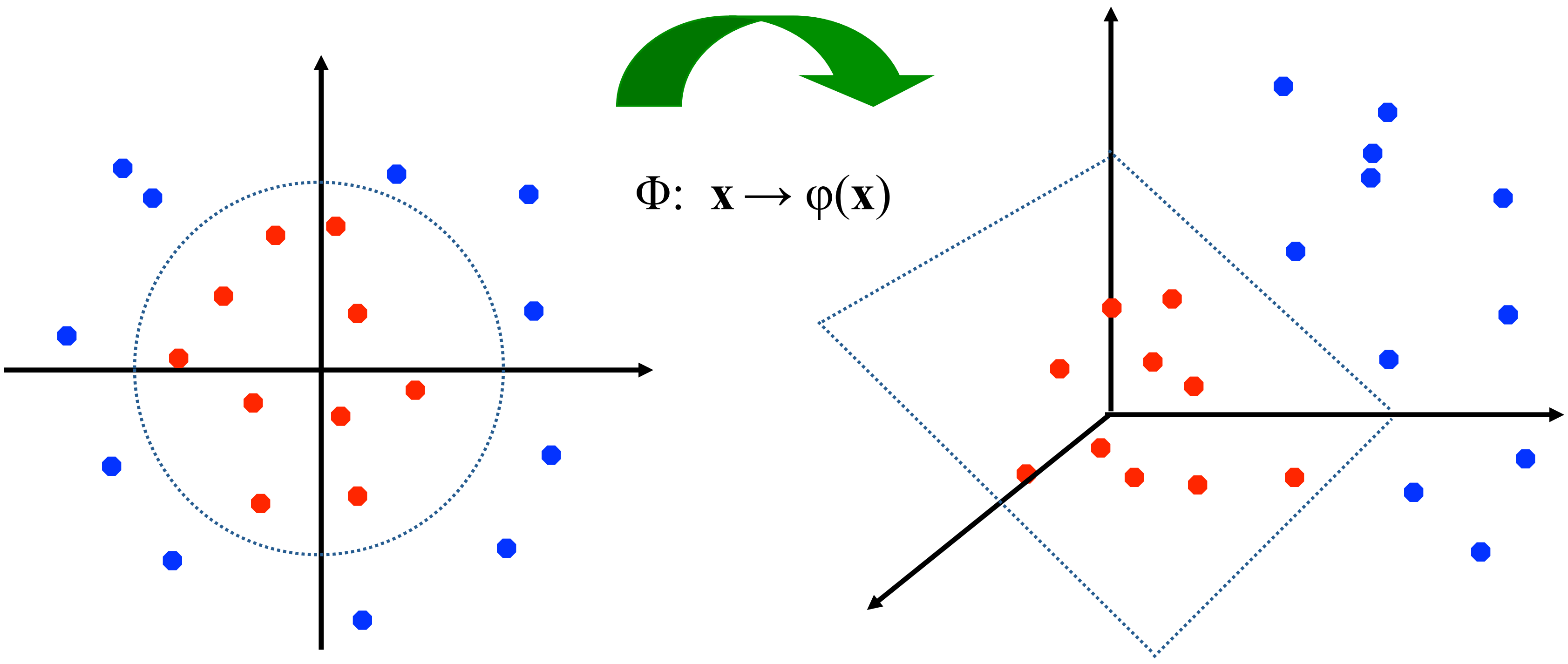
- We can map it to a higher-dimensional space:



Slide credit: Andrew Moore

Nonlinear SVMs

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



$\Phi\text{:}\ \mathbf{x} \rightarrow \varphi(\mathbf{x})$

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

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

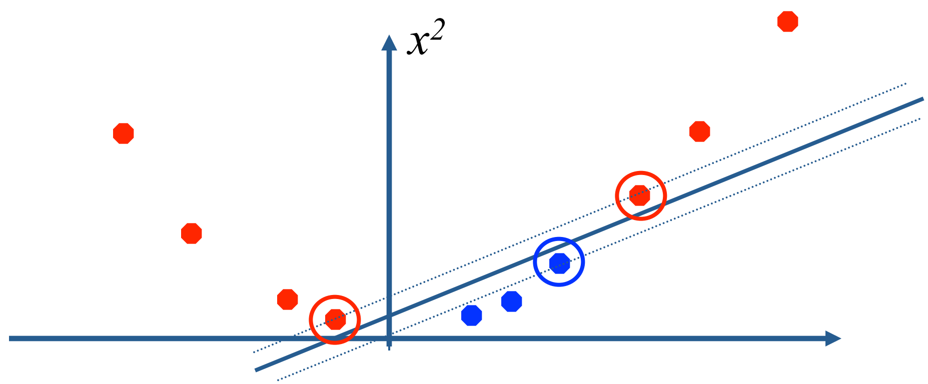
- (to be valid, the kernel function must satisfy *Mercer's condition*)
- This gives a nonlinear decision boundary in the original feature space:

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

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

Nonlinear kernel: Example

- Consider the mapping $\varphi(x) = (x, x^2)$



$$\varphi(x) \cdot \varphi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2$$

$$K(x, y) = xy + x^2 y^2$$

Kernels for bags of features

- Histogram intersection kernel:

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

- Generalized Gaussian kernel:

$$K(h_1, h_2) = \exp\left(-\frac{1}{A} D(h_1, h_2)^2\right)$$

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

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

Summary: SVMs for image classification

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

Slide credit: L. Lazebnik

What about multi-class SVMs?

- Unfortunately, there is no “definitive” multi-class SVM formulation
- In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs
- One vs. others
  - Traning: 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

Summary: Classifiers

- Nearest-neighbor and k-nearest-neighbor classifiers
  - L1 distance, χ^2 distance, quadratic distance, histogram intersection
- Support vector machines
  - Linear classifiers
  - Margin maximization
  - The kernel trick, Kernel functions: histogram intersection, generalized Gaussian, pyramid match
  - Multi-class
- Of course, there are many other classifiers out there: Neural networks, boosting, decision trees

Slide credit: L. Lazebnik

Comparison

assuming x in {0 1}

	Learning Objective	Training	Inference
Naïve Bayes	$\text{maximize} \sum_i \left[\sum_j \log P(x_{ij} \mid y_i; \theta_j) + \log P(y_i; \theta_0) \right]$	$\theta_{kj} = \frac{\sum_i \delta(x_{ij} = 1 \wedge y_i = k) + r}{\sum_i \delta(y_i = k) + Kr}$	$\boldsymbol{\theta}_1^T \mathbf{x} + \boldsymbol{\theta}_0^T (1 - \mathbf{x}) > 0$ where $\theta_{1j} = \log \frac{P(x_j = 1 \mid y = 1)}{P(x_j = 1 \mid y = 0)}$, $\theta_{0j} = \log \frac{P(x_j = 0 \mid y = 1)}{P(x_j = 0 \mid y = 0)}$
Logistic Regression	$\text{maximize} \sum_i \log(P(y_i \mid \mathbf{x}, \boldsymbol{\theta})) + \lambda \lVert \boldsymbol{\theta} \rVert$ where $P(y_i \mid \mathbf{x}, \boldsymbol{\theta}) = 1/(1 + \exp(-y_i \boldsymbol{\theta}^T \mathbf{x}))$	Gradient ascent	$\boldsymbol{\theta}^T \mathbf{x} > 0$
Linear SVM	$\text{minimize } \lambda \sum_i \xi_i + \frac{1}{2} \lVert \boldsymbol{\theta} \rVert$ such that $y_i \boldsymbol{\theta}^T \mathbf{x} \geq 1 - \xi_i \quad \forall i$	Linear programming	$\boldsymbol{\theta}^T \mathbf{x} > 0$
Kernelized SVM	complicated to write	Quadratic programming	$\sum_i y_i \alpha_i K(\hat{\mathbf{x}}_i, \mathbf{x}) > 0$
Nearest Neighbor	most similar features → same label	Record data	y_i where $i = \underset{i}{\text{argmin}}\ K(\hat{\mathbf{x}}_i, \mathbf{x})$

Slide credit: D. Hoiem

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)

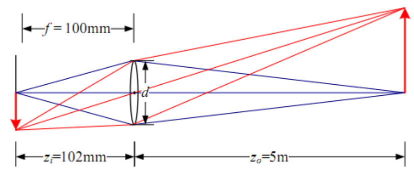
Slide credit: D. Hoiem

Some Machine Learning References

- General
  - Tom Mitchell, *Machine Learning*, McGraw Hill, 1997
  - Christopher Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995
- SVMs
  - http://www.support-vector.net/icml-tutorial.pdf

Slide credit: D. Hoiem

Course Outline



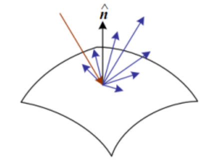
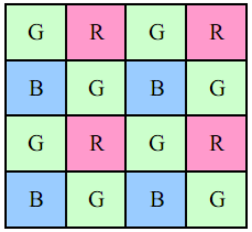


Image Formation and Processing

Light, Shape and Color

The Pin-hole Camera Model, The Digital Camera

Linear filtering, Template Matching, Image Pyramids



Feature Detection and Matching

Edge Detection, Interest Points: Corners and Blobs

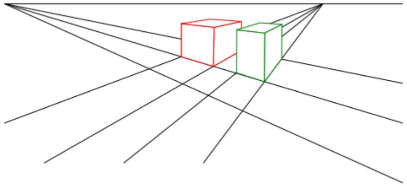
Local Image Descriptors

Feature Matching and Hough Transform

Multiple Views and Motion

Geometric Transformations, Camera Calibration

Feature Tracking , Stereo Vision



Segmentation and Grouping

Segmentation by Clustering, Region Merging and Growing

Advanced Methods Overview: Active Contours, Level-Sets, Graph-Theoretic Methods

Detection and Recognition

Problems and Architectures Overview

Statistical Classifiers, Bag-of-Words Model, Detection by Sliding Windows

Resources

Books

R. Szeliski, Computer Vision: Algorithms and Applications, 2010 – *available online*

D. A. Forsyth and J. Ponce, Computer Vision: A Modern Approach, 2003

L. G. Shapiro and G. C. Stockman, Computer Vision, 2001

Web

CVonline: The Evolving, Distributed, Non-Proprietary, On-Line Compendium of Computer Vision

http://homepages.inf.ed.ac.uk/rbf/CVonline/

Dictionary of Computer Vision and Image Processing

http://homepages.inf.ed.ac.uk/rbf/CVDICT/

Computer Vision Online

http://www.computervisiononline.com/

Programming

Development environments/languages: Matlab, Python and C/C++

Toolboxes and APIs: OpenCV, VLFeat Matlab Toolbox, Piotr's Computer Vision Matlab Toolbox, EasyCamCalib Software, FLANN, Point Cloud Library PCL, LibSVM, Camera Calibration Toolbox for Matlab