HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# Computer Vision

Ch (part 2): Object recognition

## Contents

# Overview of ‘semantic vision’?



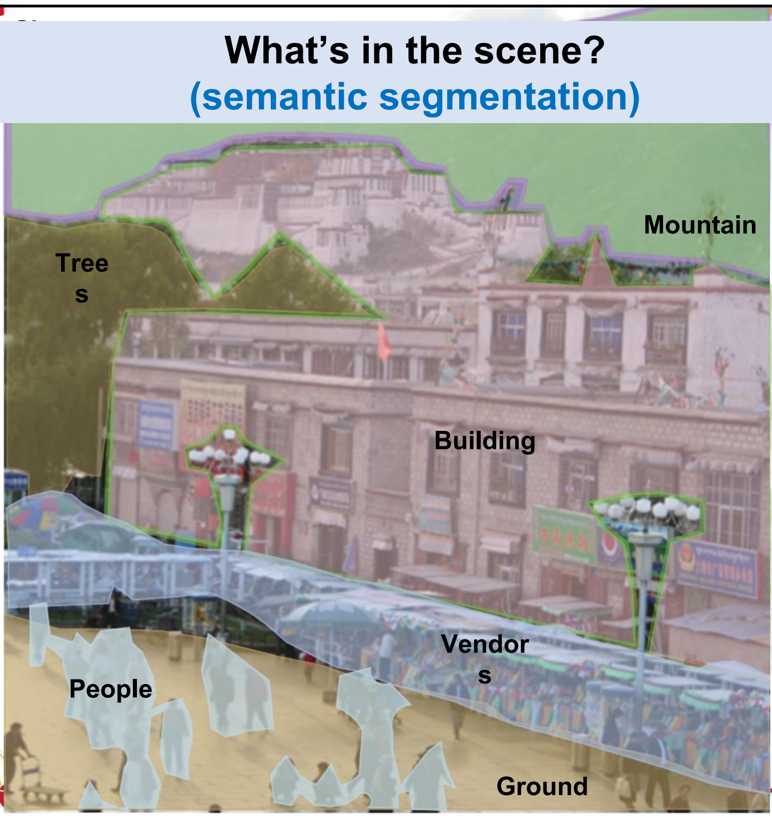
## Is this a street light?
## (Recognition / classification)

## Where are the people?
## (Detection)



## Is that Potala palace?
## (Identification)

## What's in the scene?
## (semantic segmentation)






## What type of scene is it?
## (Scene categorization)

## What are these people doing?
## (Activity / Event Recognition)

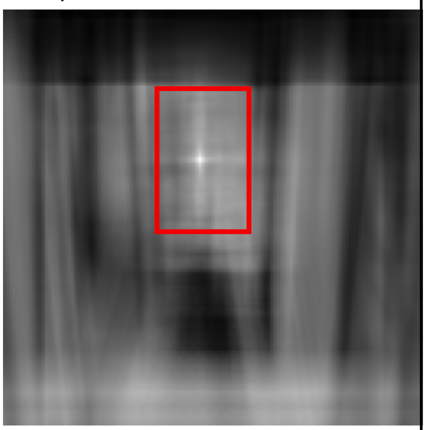


# Object recognition
# Is it really so hard?

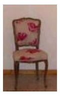
This is a chair

Find the chair in this image

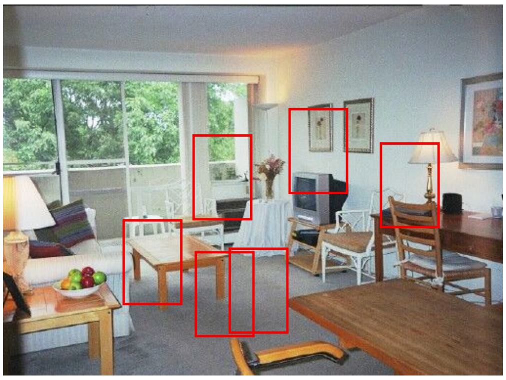
Output of normalized correlation

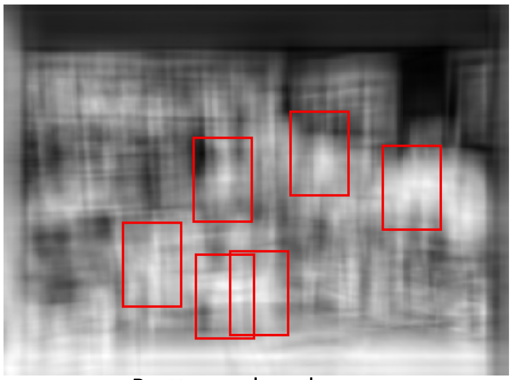


# Object recognition
# Is it really so hard?



Find the chair in this image





Pretty much garbage

Simple template matching is not going to make it

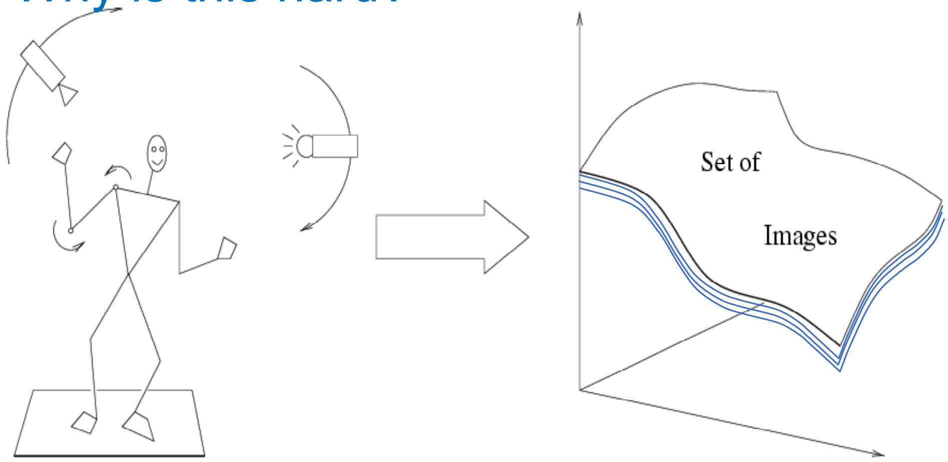
A "popular method is that of template matching, by point to point correlation of a model pattern with the image pattern. These techniques are inadequate for three-dimensional scene analysis for many reasons, such as occlusion, changes in viewing angle, and articulation of parts." Nivatia & Binford, 1977.



# And it can get a lot harder

Brady, M. J., & Kersten, D. (2003). Bootstrapped learning of novel objects. J Vis, 3(6), 413-422

## Why is this hard?




Variability: Camera position
Illumination
Shape parameters



## How many object categories are there?

# Challenge: variable viewpoint

Michelangelo 1475-1564



# Challenge: variable illumination

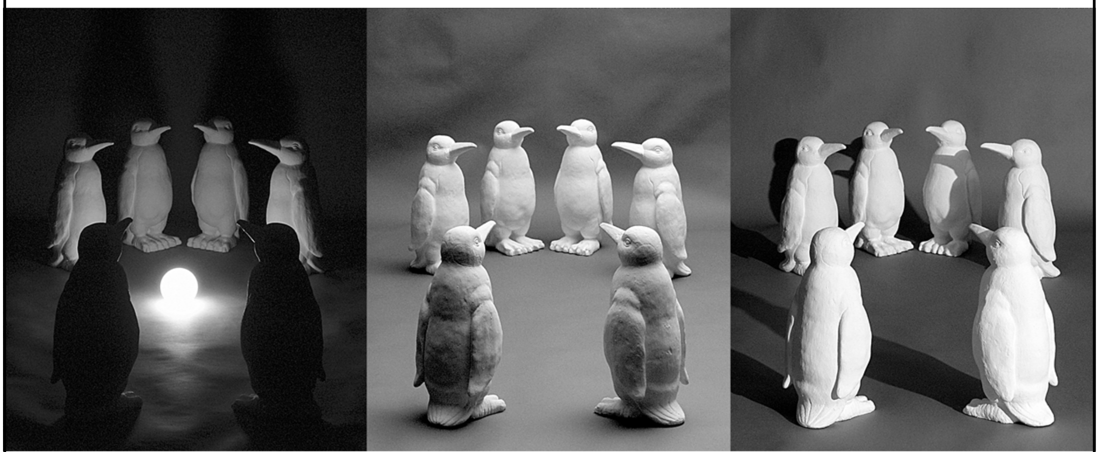


image credit: J. Koenderink

# Challenge: scale



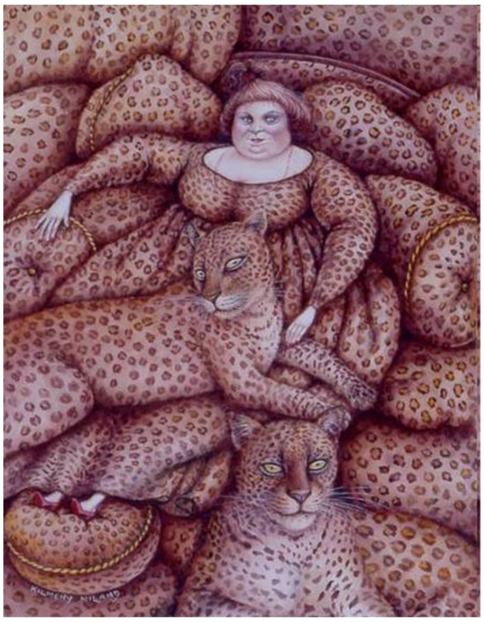

# Challenge: deformation

# Challenge: Occlusion

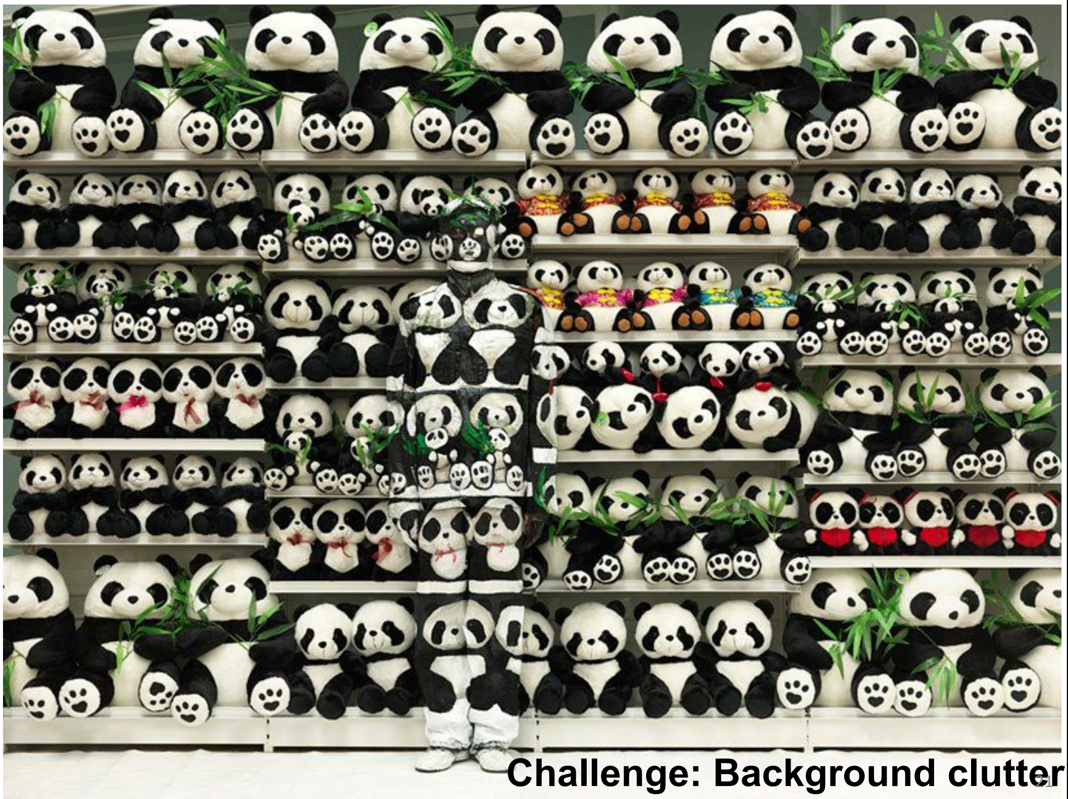
Magritte, 1957



# Challenge: background clutter



Kilmeny Niland. 1995



Challenge: Background clutter

## Challenge: intra-class variations

# Image Classification/ Recognition

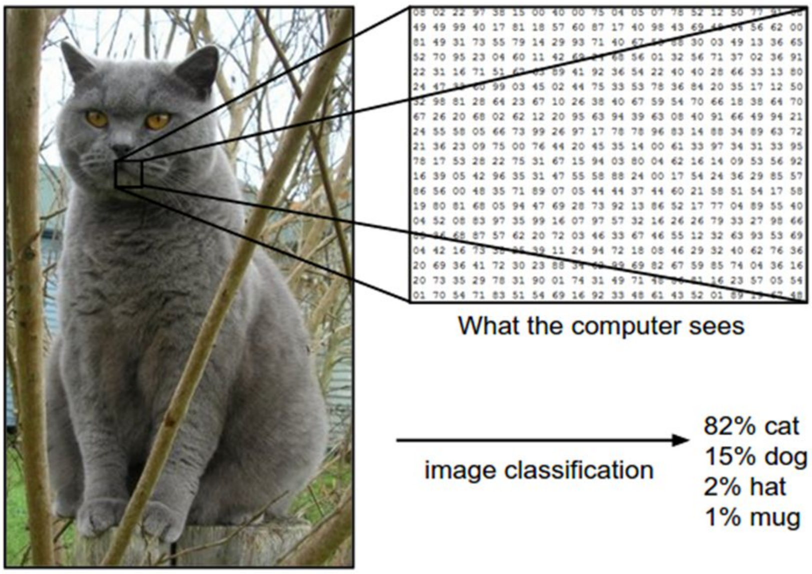


# Image Classification/ Recognition

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

# Image Classification: Problem

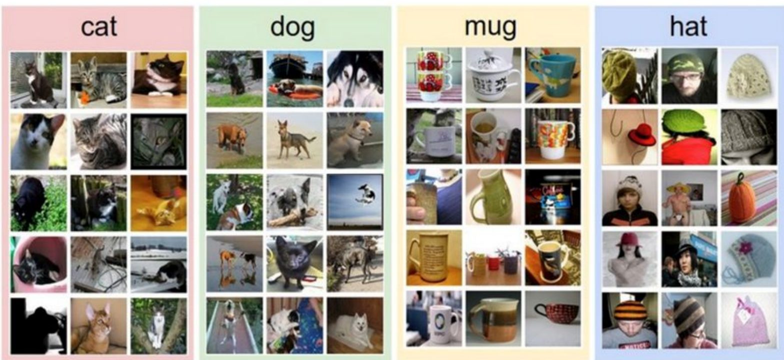





# Data-driven approach

- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images

Example training set



# A simple pipeline - Training

Training Images

Image Features



# A simple pipeline - Training

Training Labels

Training Images

Image Features

Training

# A simple pipeline - Training

Training Images

Image Features → Training → Learned Classifier

Training Labels → Training



# A simple pipeline - Training

Training Images

Image Features → Training → Learned Classifier

Training Labels → Training

Test Image

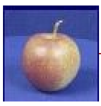
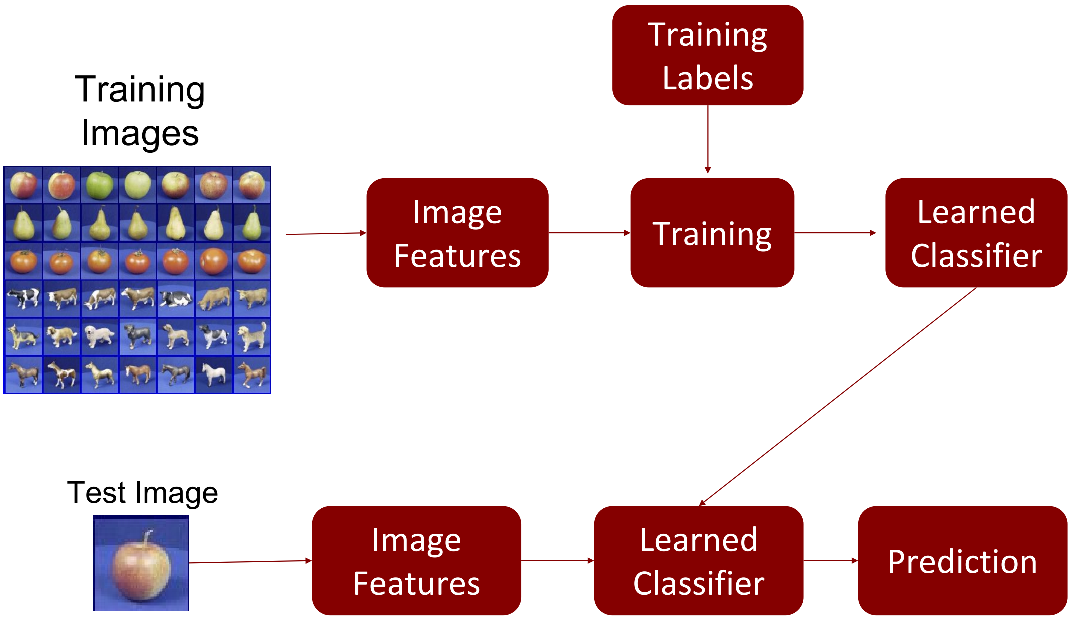


Image Features

# A simple pipeline - Training





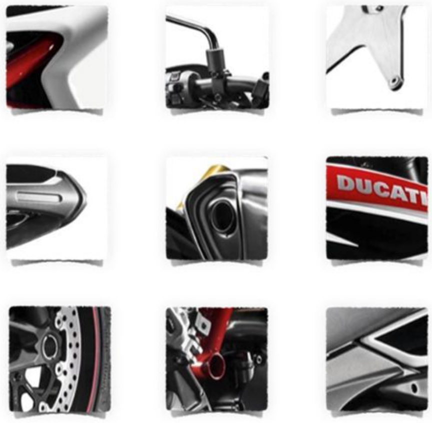

# Bag of words

Basic model: recall

Vocabulary tree

Some local feature are very informative

An object as




a collection of local features

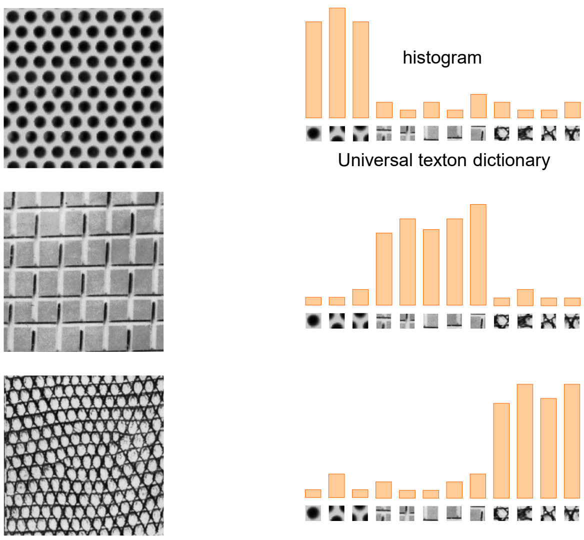
(bag-of-features)

- deals well with occlusion
- scale invariant
- rotation invariant



# Bag-of-features

represent a data item (document, texture, image) as a histogram over features



# Standard BOW pipeline
(for image classification)



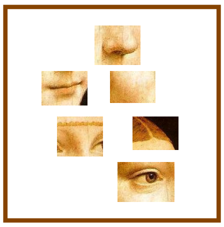
**Dictionary Learning:**
Learn Visual Words using clustering

**Encode:**
build Bags-of-Words (BOW) vectors
for each image

**Classify:**
Train and test data using BOWs

**Dictionary Learning:**
Learn Visual Words using clustering

1. extract features (e.g., SIFT) from images



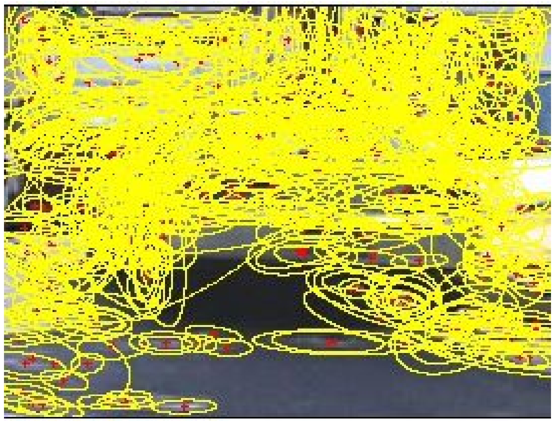


**Dictionary Learning:**
Learn Visual Words using clustering

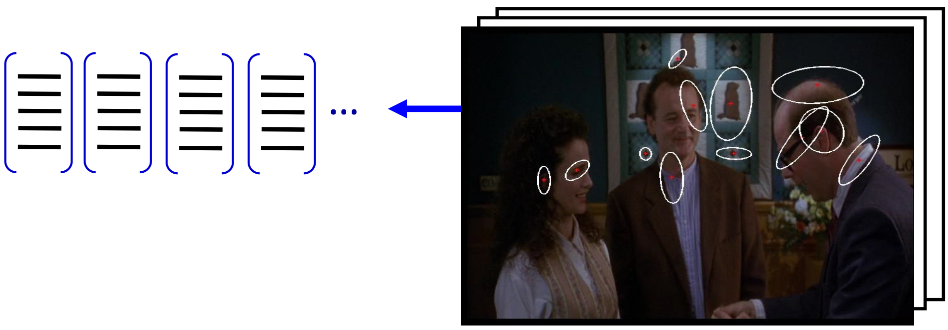
2. Learn visual dictionary (e.g., K-means clustering)

## What kinds of features can we extract?

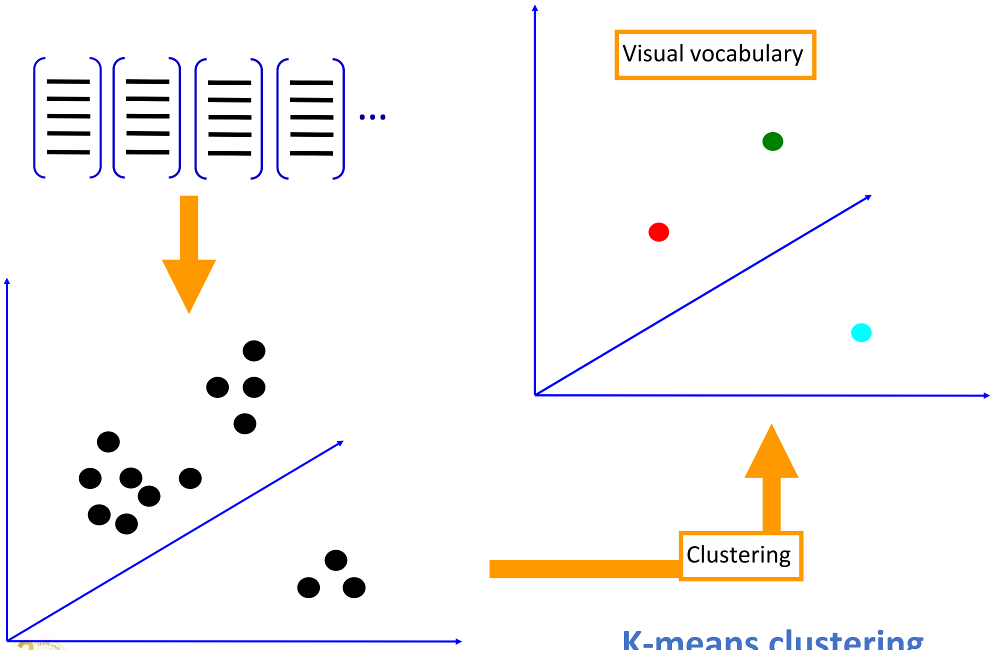
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005
- Other methods
  - Random sampling (Vidal-Naquet & Ullman, 2002)
  - Segmentation-based patches (Barnard et al. 2003)







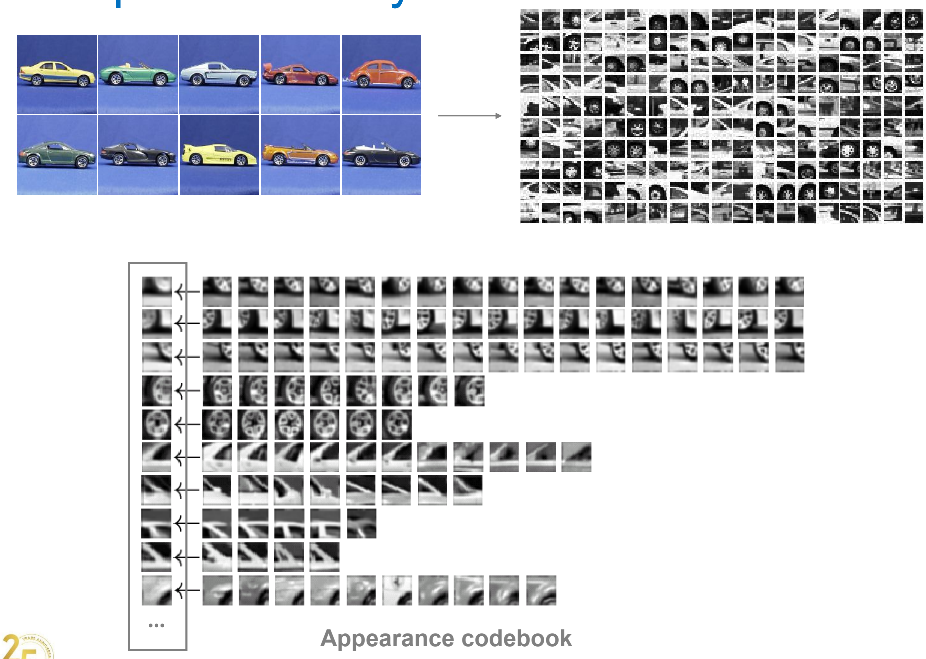
## How do we learn the dictionary?




**K-means clustering**



# Example dictionary



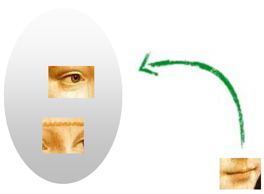

Source: B. Leibe

**Dictionary Learning:**
Learn Visual Words using clustering

**Encode:**
build Bags-of-Words (BOW) vectors
for each image

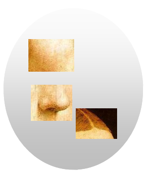
**Classify:**
Train and test data using BOWs





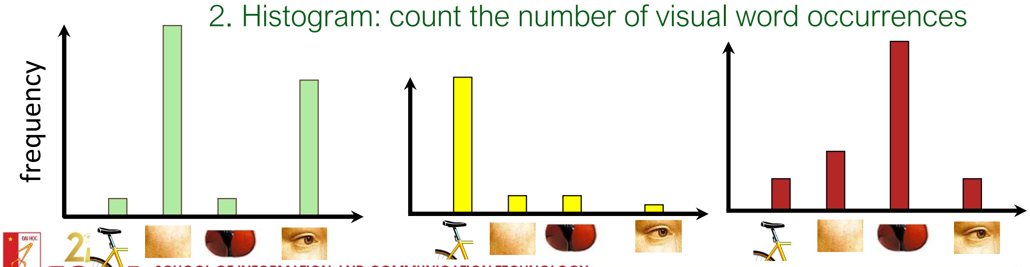
1. Quantization: image features gets associated to a visual word (nearest cluster center)

**Encode:**
build Bags-of-Words (BOW) vectors
for each image

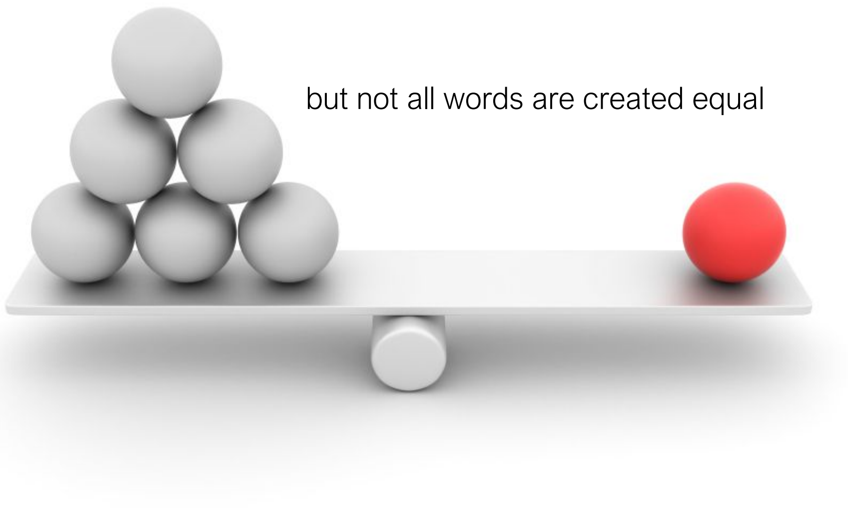


**Encode:**
build Bags-of-Words (BOW) vectors
for each image

2. Histogram: count the number of visual word occurrences





but not all words are created equal

# TF-IDF

**T**erm **F**requency **I**nverse **D**ocument **F**requency

$$\boldsymbol{v}_d = [n(w_{1,d}) \;\; n(w_{2,d}) \;\; \cdots \;\; n(w_{T,d})]$$

weigh each word by a heuristic

$$\boldsymbol{v}_d = [n(w_{1,d})\alpha_1 \;\; n(w_{2,d})\alpha_2 \;\; \cdots \;\; n(w_{T,d})\alpha_T]$$

$$n(w_{i,d})\alpha_i = \underbrace{n(w_{i,d})}_{\text{term frequency}} \underbrace{\log \left\{ \frac{D}{\sum_{d'} \mathbf{1}[w_i \in d']} \right\}}_{\text{inverse document frequency}}$$

(down-weights **common** terms)



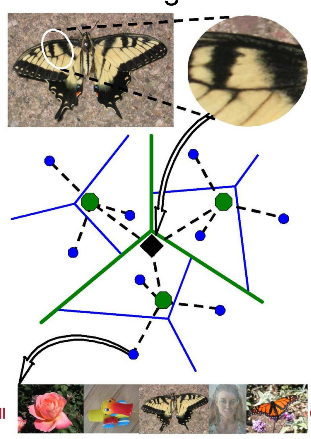
# Scalability: Alignment to large databases

- What if we need to align a test image with thousands or millions of images in a model database?
  - Efficient putative match generation
    - Fast nearest neighbor search, inverted indexes

Test image

Vocabulary tree with inverted index

Database

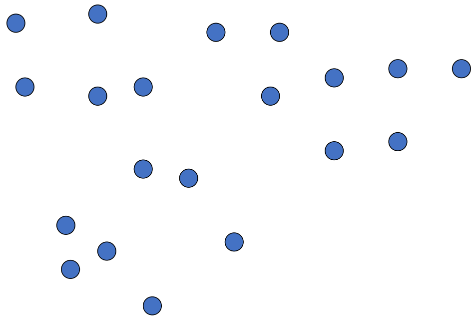


D. Nistér and H. Stewénius, [Scalable Recognition with a Vocabulary Tree](), CVPR 2006

slide: S. Lazebnik

# What is a Vocabulary Tree?

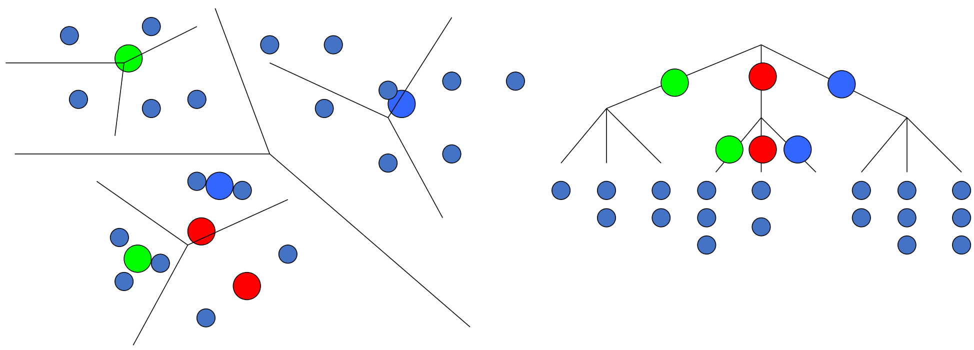
Nister and Stewenius CVPR 2006



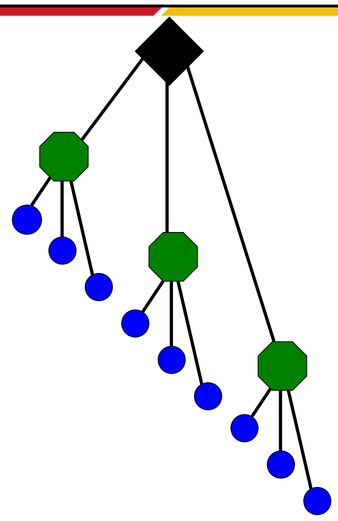


# What is a Vocabulary Tree?

Nister and Stewenius CVPR 2006



- Multiple rounds of K-Means to compute decision tree (offline)
- Fill and query tree online



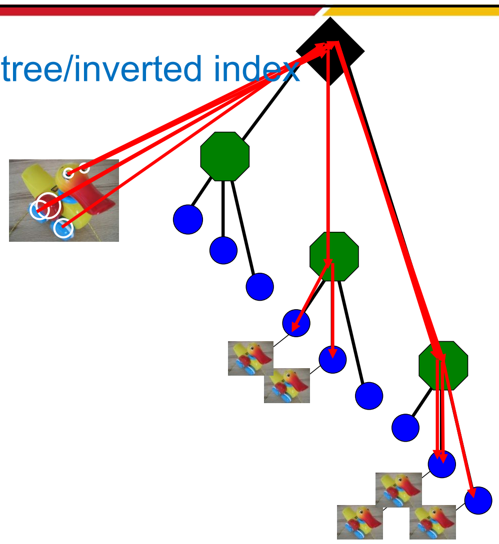


Slide credit: D. Nister



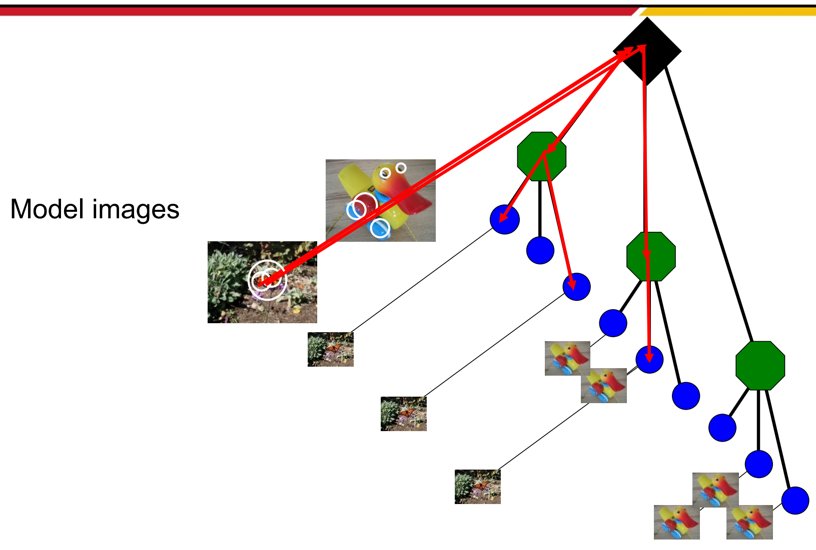
## Populating the vocabulary tree/inverted index

Model images

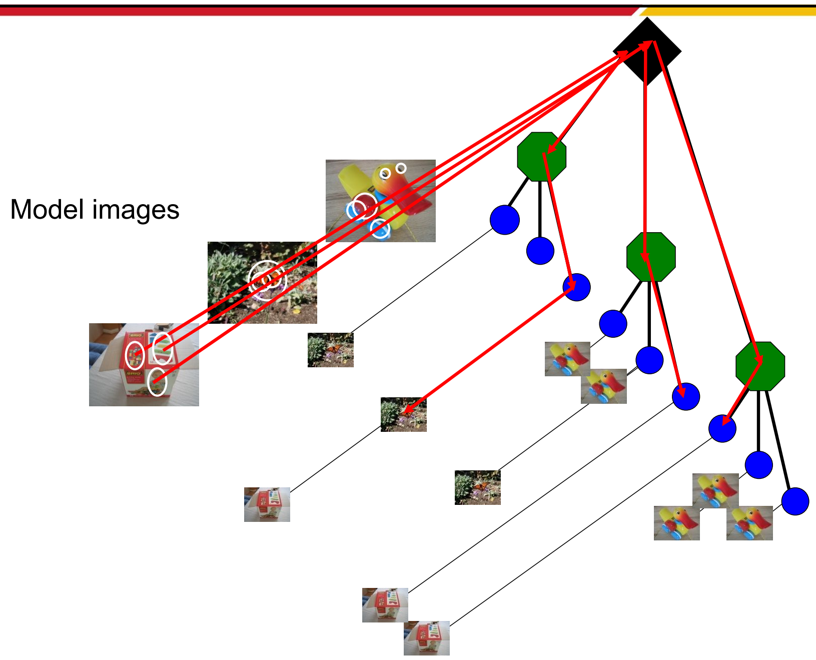


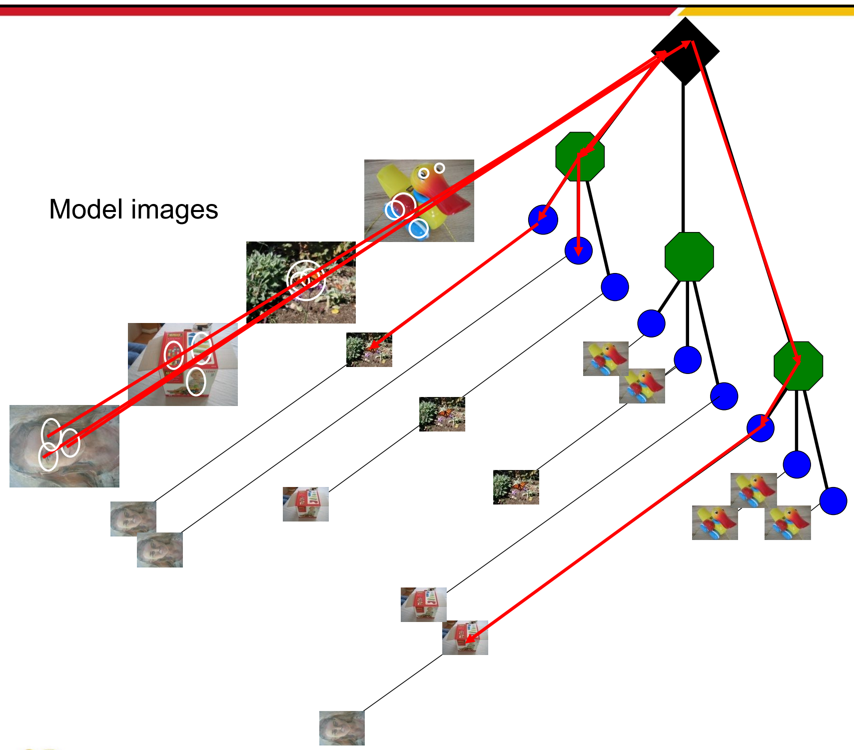


Slide credit: D. Nister


Model images



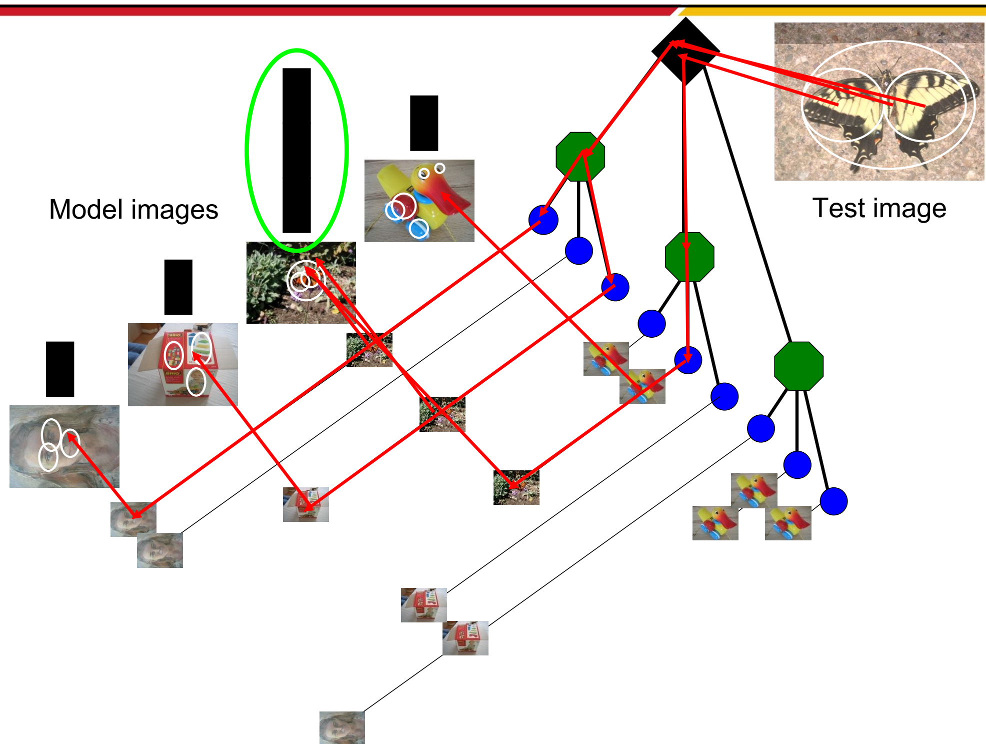

Model images






Slide credit: D. Nister






Looking up a test image



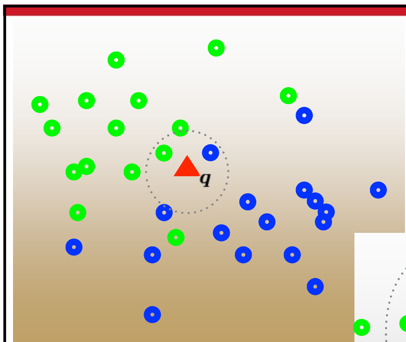
Slide credit: D. Nister

**Dictionary Learning:**
Learn Visual Words using clustering

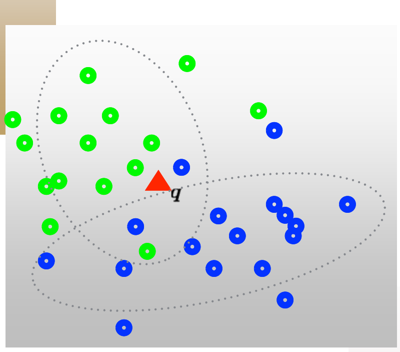
**Encode:**
build Bags-of-Words (BOW) vectors
for each image

**Classify:**
Train and test data using BOWs



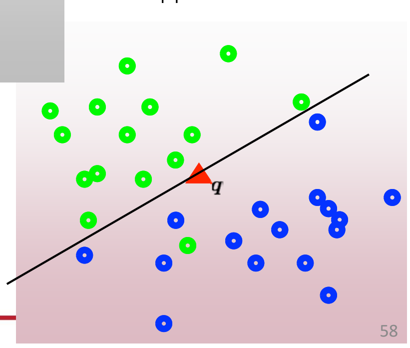



K nearest neighbors




Naïve Bayes

Support Vector Machine

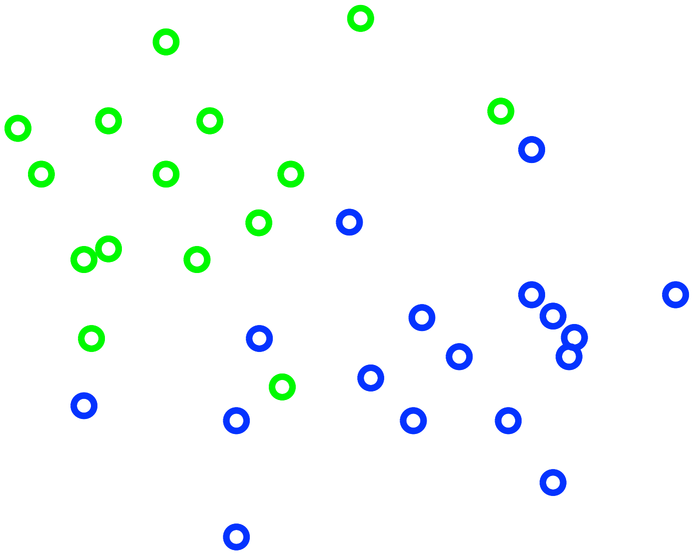


# Classification



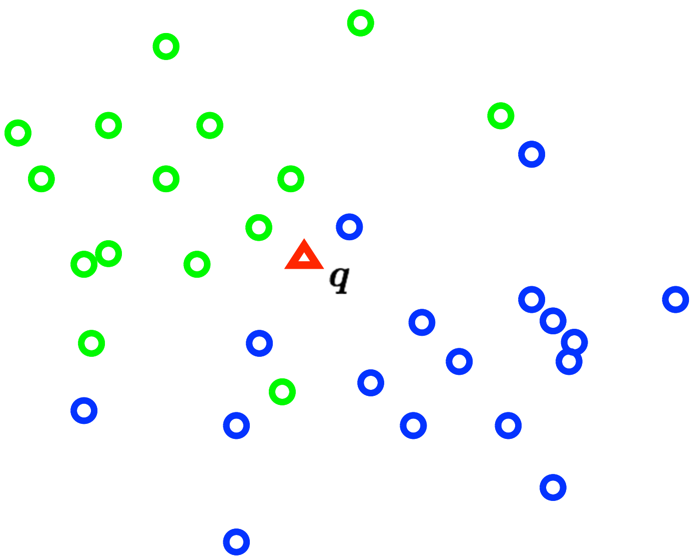
# K nearest neighbors

## Distribution of data from two classes



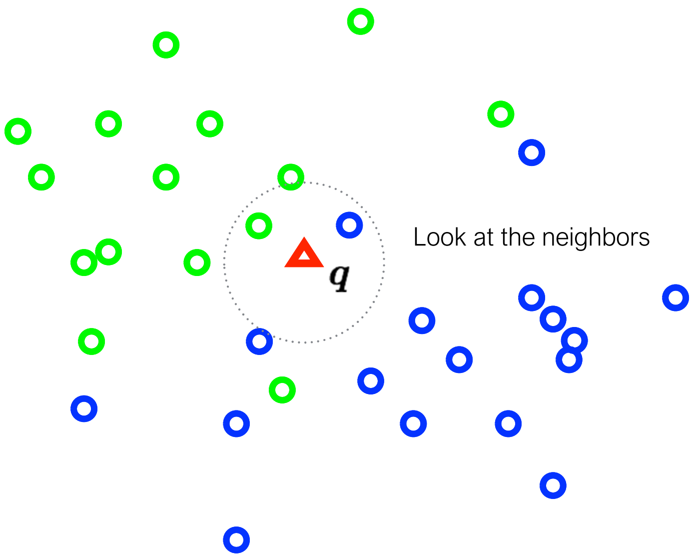


## Distribution of data from two classes



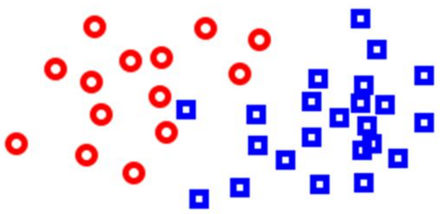

*Which class does q belong too?*

## Distribution of data from two classes






# K-Nearest Neighbor (KNN) Classifier

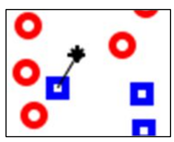


Non-parametric pattern classification approach

Consider a two class problem where each sample consists of two measurements (x,y).

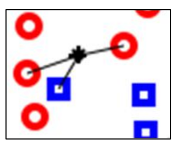
For a given query point q, assign the class of **the nearest neighbor**

k = 1

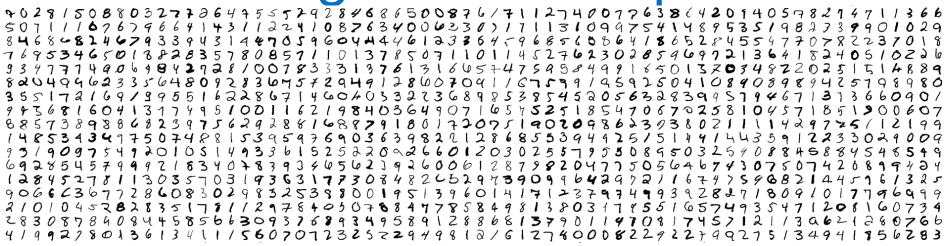


Compute the **k nearest neighbors** and assign the class by majority vote.

k = 3



# Nearest Neighbor is competitive



| | Test Error Rate (%) |
|---|---|
| Linear classifier (1-layer NN) | 12.0 |
| K-nearest-neighbors, Euclidean | 5.0 |
| K-nearest-neighbors, Euclidean, deskewed | 2.4 |
| K-NN, Tangent Distance, 16x16 | 1.1 |
| K-NN, shape context matching | 0.67 |
| 1000 RBF + linear classifier | 3.6 |
| SVM deg 4 polynomial | 1.1 |
| 2-layer NN, 300 hidden units | 4.7 |
| 2-layer NN, 300 HU, [deskewing] | 1.6 |
| LeNet-5, [distortions] | 0.8 |
| Boosted LeNet-4, [distortions] | 0.7 |

**MNIST Digit Recognition**

- Handwritten digits
- 28x28 pixel images: d = 784
- 60,000 training samples
- 10,000 test samples

Yann LeCunn



**What is the best distance metric between data points?**

- Typically Euclidean distance
- Locality sensitive distance metrics
- Important to normalize.
  Dimensions have different scales

**How many K?**

- Typically k=1 is good
- Cross-validation (try different k!)

# Distance metrics

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{(x_1-y_1)^2+\cdots+(x_N-y_N)^2}$$ Euclidean

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|} = \frac{x_1y_1+\cdots+x_Ny_N}{\sqrt{\sum_n x_n^2}\sqrt{\sum_n y_n^2}}$$ Cosine

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2}\sum_n \frac{(x_n-y_n)^2}{(x_n+y_n)}$$ Chi-squared



# Distance metrics

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

L2 (Euclidean) distance

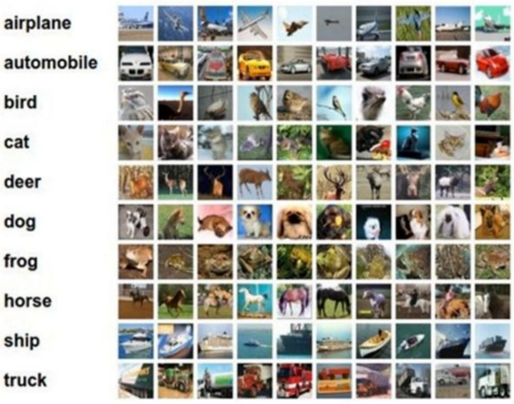
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

- Two most commonly used special cases of p-norm

$$\|x\|_p = \left(|x_1|^p + \cdots + |x_n|^p\right)^{\frac{1}{p}} \qquad p \geq 1, x \in \mathbb{R}^n$$

# CIFAR-10 and NN results

Example dataset: **CIFAR-10**
**10** labels
**50,000** training images
**10,000** test images.

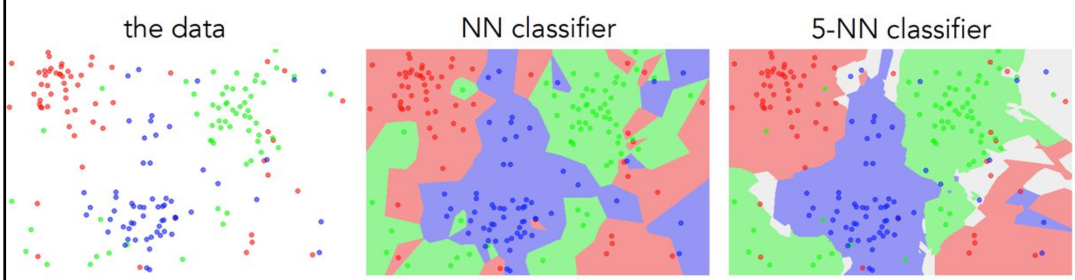



For every test image (first column), examples of nearest neighbors in rows



# k-nearest neighbor

- Find the k closest points from training data
- Labels of the ***k points "vote"*** to classify



# Hyperparameters

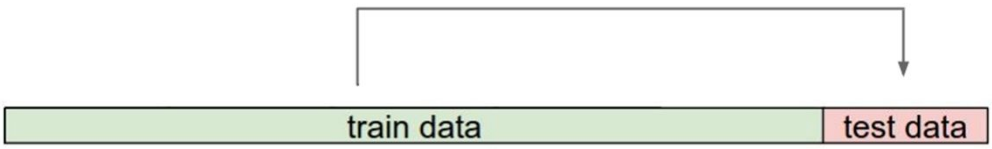
- What is the best distance to use?
- What is the best value of k to use?

- i.e., how do we set the hyperparameters?

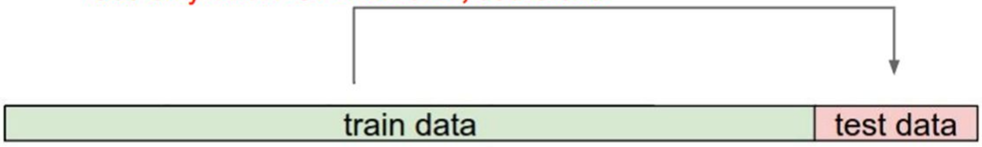
- Very problem-dependent
- Must try them all and see what works best



Try out what hyperparameters work best on test set.

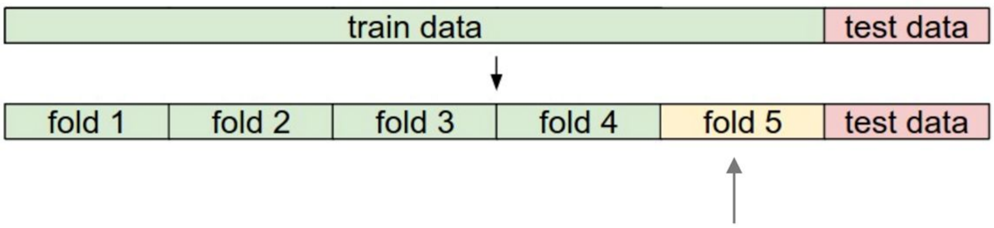


Trying out what hyperparameters work best on test set:
Very bad idea. The test set is a proxy for the generalization performance!
Use only **VERY SPARINGLY,** at the end.



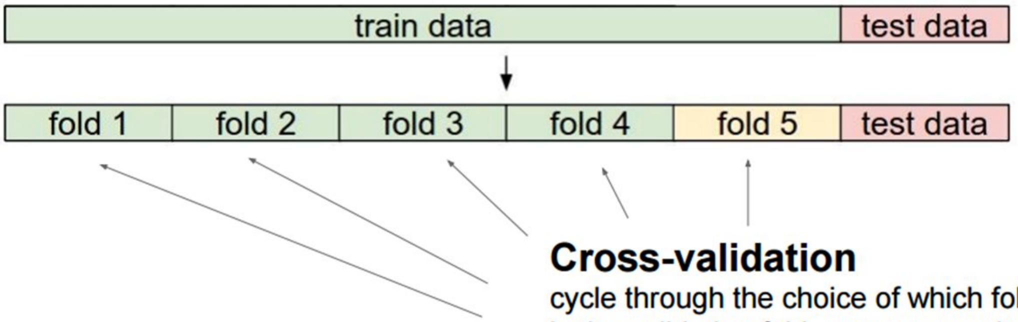



# Validation



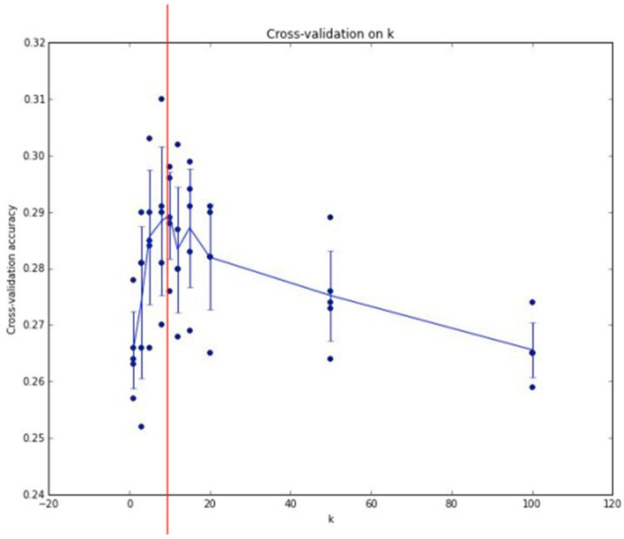

Validation data
use to tune hyperparameters
evaluate on test set ONCE at the end

# Cross-validation




**Cross-validation**
cycle through the choice of which fold is the validation fold, average results.






Example of
5-fold cross-validation
for the value of **k.**

Each point: single outcome.

The line goes through the mean, bars indicated standard deviation

(Seems that k ~= 7 works best for this data)

# How to pick hyperparameters?

- Methodology
  - Train and test
  - Train, validate, test
- Train for original model
- Validate to find hyperparameters
- Test to understand generalizability



# kNN

**Pros**

- simple yet effective

**Cons**

- search is expensive (can be sped-up)
- storage requirements
- difficulties with high-dimensional data

# kNN -- Complexity and Storage

- N training images, M test images

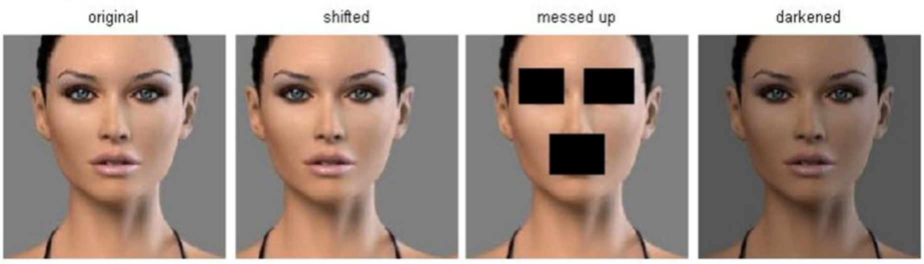
- Training: O(1)
- Testing: O(MN)

- Hmm…
  - Normally need the opposite
  - Slow training (ok), fast testing (necessary)



k-Nearest Neighbor on images **never used.**

- terrible performance at test time
- distance metrics on level of whole images can be very unintuitive

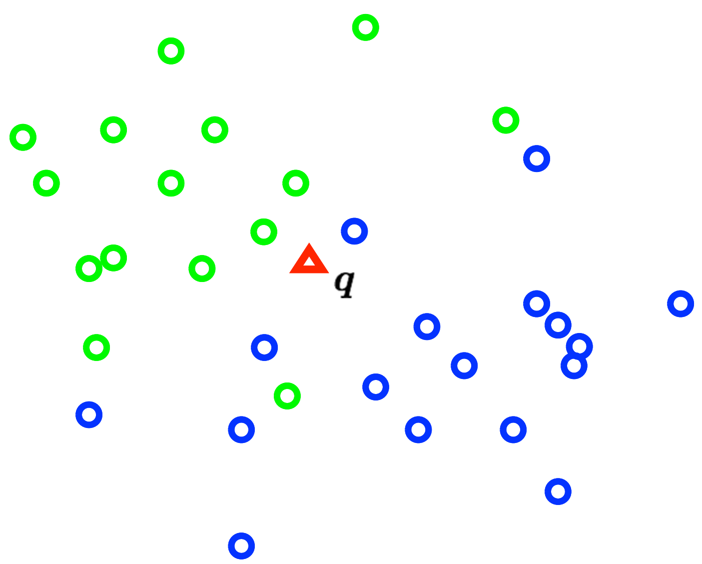



(all 3 images have same L2 distance to the one on the left)

# Naïve Bayes



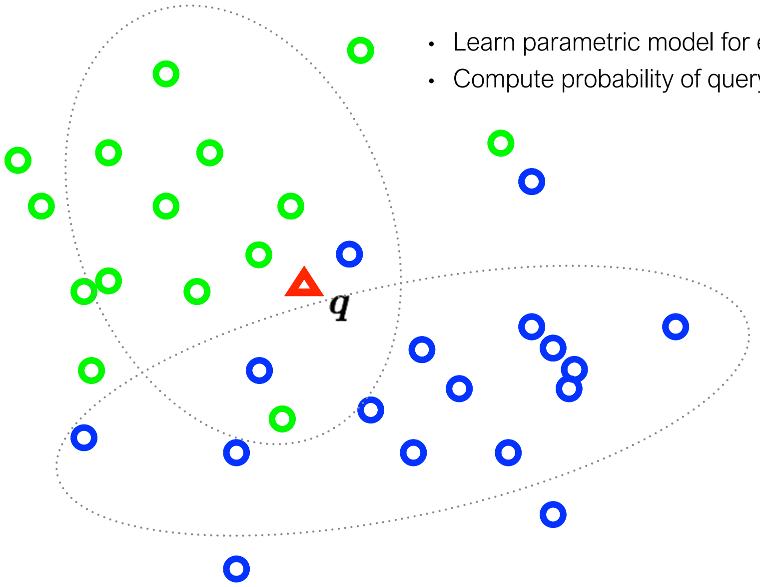
Distribution of data from two classes




*Which class does q belong too?*

## Distribution of data from two classes

- Learn parametric model for each class
- Compute probability of query






This is called the posterior:
the probability of a class $z$ given the observed features $X$

$$p(\boldsymbol{z}|\boldsymbol{X})$$

For classification, z is a discrete random variable (e.g., car, person, building)

X is a set of observed features (e.g., features from a single image)

(it's a function that returns a single probability value)

This is called the posterior:

the probability of a class $z$ given the observed features $X$

$$p(\boldsymbol{z}|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N)$$

For classification, z is a discrete random variable (e.g., car, person, building)

Each x is an observed feature (e.g., visual words)

(it's a function that returns a single probability value)



**Recall:**

The posterior can be decomposed according to **Bayes' Rule**

$$\underbrace{p(A|B)}_{\text{posterior}} = \frac{\overbrace{p(B|A)}^{\text{likelihood}}\overbrace{p(A)}^{\text{prior}}}{p(B)}$$

In our context…

$$p(\boldsymbol{z}|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N) = \frac{p(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N)}$$

The naive Bayes' classifier is solving this optimization

$$\hat{z} = \arg\max_{z \in \boldsymbol{\mathcal{Z}}} p(z|\boldsymbol{X})$$

MAP (maximum a posteriori) estimate

$$\hat{z} = \arg\max_{z \in \boldsymbol{\mathcal{Z}}} \frac{p(\boldsymbol{X}|z)p(z)}{p(\boldsymbol{X})}$$

Bayes' Rule

$$\hat{z} = \arg\max_{z \in \boldsymbol{\mathcal{Z}}} p(\boldsymbol{X}|z)p(z)$$

Remove constants

To optimize this…we need to compute this

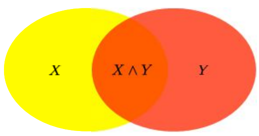
Compute the likelihood…



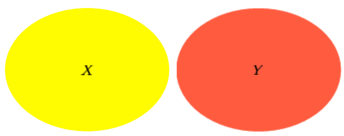
A naive Bayes' classifier assumes all features are
***conditionally independent***

$$\begin{aligned} p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\boldsymbol{z}) &= p(\boldsymbol{x}_1|\boldsymbol{z})p(\boldsymbol{x}_2, \ldots, \boldsymbol{x}_N|\boldsymbol{z}) \\ &= p(\boldsymbol{x}_1|\boldsymbol{z})p(\boldsymbol{x}_2|\boldsymbol{z})p(\boldsymbol{x}_3, \ldots, \boldsymbol{x}_N|\boldsymbol{z}) \\ &= p(\boldsymbol{x}_1|\boldsymbol{z})p(\boldsymbol{x}_2|\boldsymbol{z})\cdots p(\boldsymbol{x}_N|\boldsymbol{z}) \end{aligned}$$

**Recall:**




$$p(x, y) = p(x|y)p(y)$$




$$p(x, y) = p(x)p(y)$$

## To compute the MAP estimate

Given (1) a set of known parameters

$$p(\boldsymbol{z}) \quad p(\boldsymbol{x}|\boldsymbol{z})$$

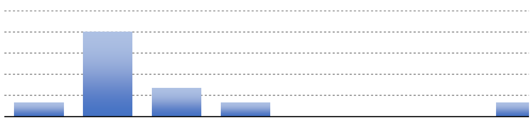
(2) observations

$$\{x_1, x_2, \ldots, x_N\}$$

Compute which z has the largest probability

$$\hat{z} = \underset{z \in \boldsymbol{z}}{\arg\max}\, p(z) \prod_n p(x_n|z)$$





| count | 1 | 6 | 2 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|
| **word** | Tartan | robot | CHIMP | CMU | bio | soft | ankle | sensor |
| **p(x\|z)** | 0.09 | 0.55 | 0.18 | 0.09 | 0.0 | 0.0 | 0.0 | 0.09 |

$$p(X|z) = \prod_v p(x_v|z)^{c(w_v)}$$

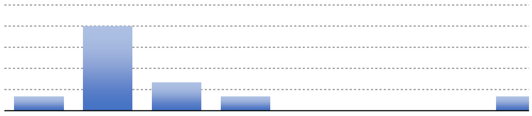
$$= (0.09)^1 (0.55)^6 \cdots (0.09)^1$$

Numbers get really small so use log probabilities

$$\log p(X|z = \text{‘grandchallenge’}) = -2.42 - 3.68 - 3.43 - 2.42 - 0.07 - 0.07 - 0.07 - 2.42 = -14.58$$

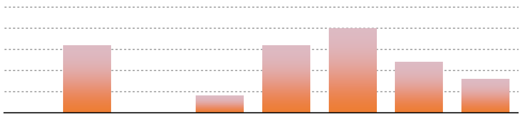
$$\log p(X|z = \text{‘softrobot’}) = -7.63 - 9.37 - 15.18 - 2.97 - 0.02 - 0.01 - 0.02 - 2.27 = -37.48$$

* typically add pseudo-counts (0.001)
** this is an example for computing the likelihood, need to multiply times **prior** to get posterior

| count | 1 | 6 | 2 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|
| **word** | Tartan | robot | CHIMP | CMU | bio | soft | ankle | sensor |
| **p(x\|z)** | 0.09 | 0.55 | 0.18 | 0.09 | 0.0 | 0.0 | 0.0 | 0.09 |

log p(X|z=grand challenge) = **- 14.58**
log p(X|z=bio inspired) = - 37.48

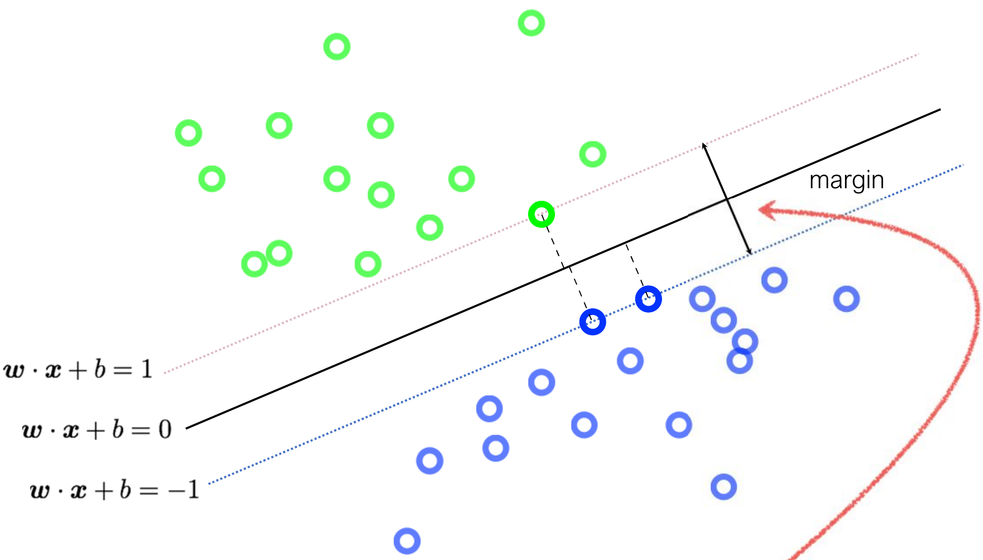
| count | 0 | 4 | 0 | 1 | 4 | 5 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|
| **word** | Tartan | robot | CHIMP | CMU | bio | soft | ankle | sensor |
| **p(x\|z)** | 0.0 | 0.21 | 0.0 | 0.05 | 0.21 | 0.26 | 0.16 | 0.11 |

log p(X|z=grand challenge) = - 94.06
log p(X|z=bio inspired) = **- 32.41**

* typically add pseudo-counts (0.001)
** this is an example for computing the likelihood, need to multiply times prior to get posterior



Find hyperplane **w** such that …




the gap between parallel hyperplanes $\frac{2}{\|w\|}$ is maximized

# References

Most of these slides were adapted from:

1. Ioannis Yannis, Gkioulekas (16-385 Computer Vision, Spring 2020, CMU)

2. Kristen Grauman (CS 376: Computer Vision, Spring 2018, The University of Texas at Austin)

3. Noah Snavely (Cornell University)

4. Fei-Fei Li (Stanford University)



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# Thank you!