

Principles of Safe Autonomy

Lecture 4: Object recognition

Sayan Mitra

~~Jan 30~~, Feb 13 2019

slides from Svetlana Lazebnik



Overview

- Recognition tasks
- A statistical learning approach
- “Classic” recognition pipeline
 - Bags of features
 - Spatial pyramids
- Classifiers: SVM

Announcement:

- Uber talk Feb 20th, 4-5 pm, NCSA



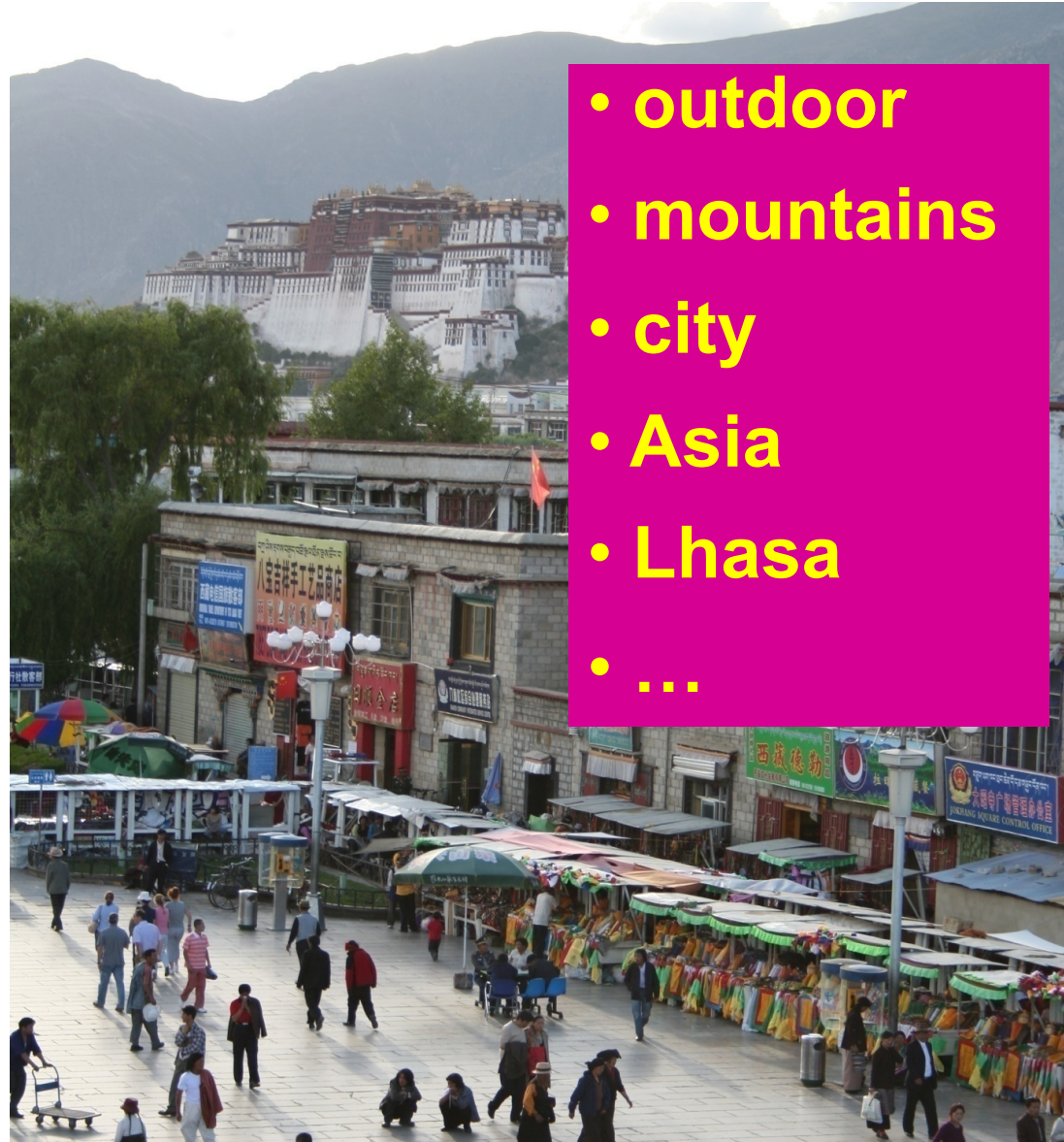
Common recognition tasks



Adapted from
Fei-Fei Li



Image classification and tagging



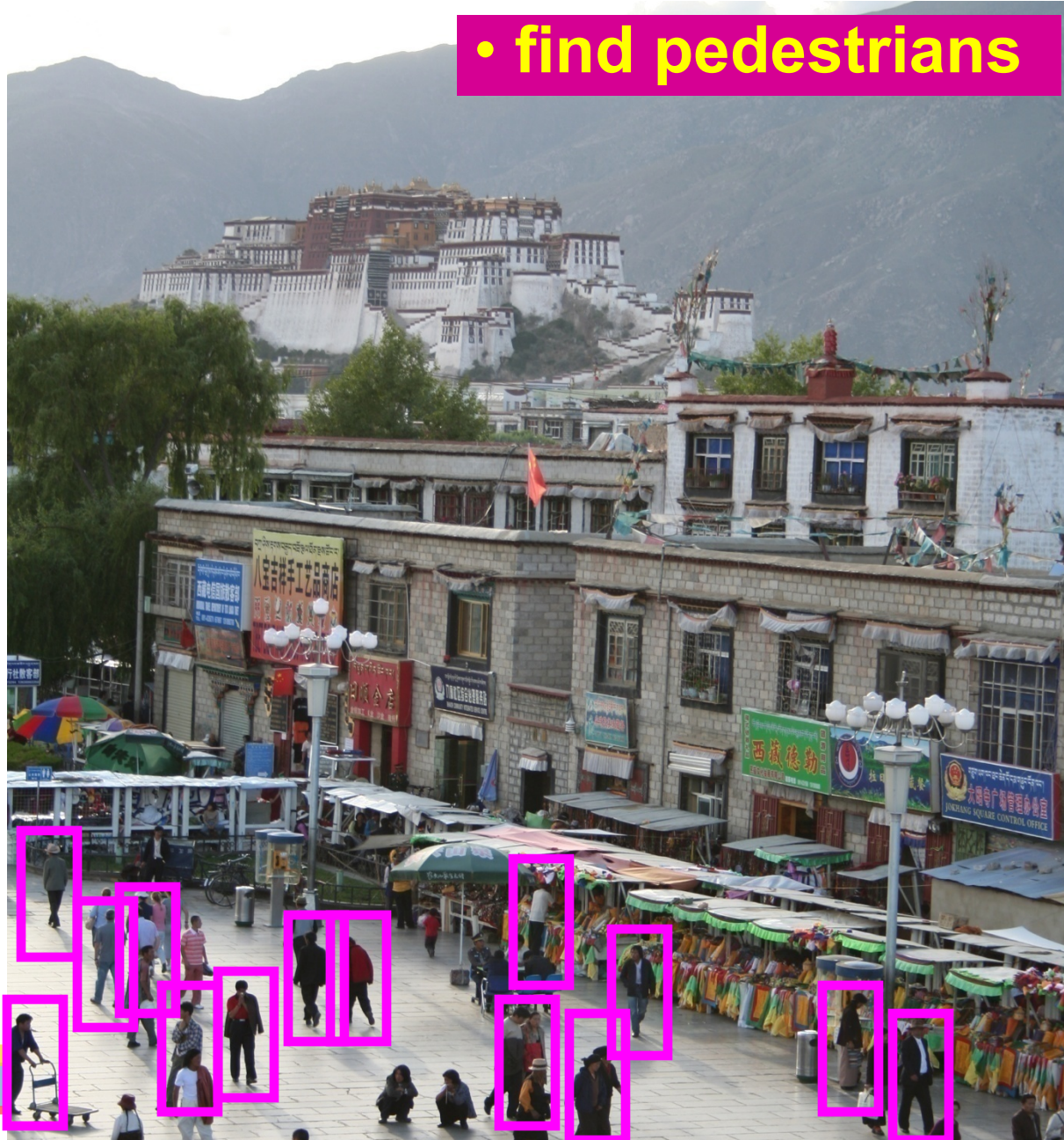
- outdoor
- mountains
- city
- Asia
- Lhasa
- ...

Adapted from
Fei-Fei Li



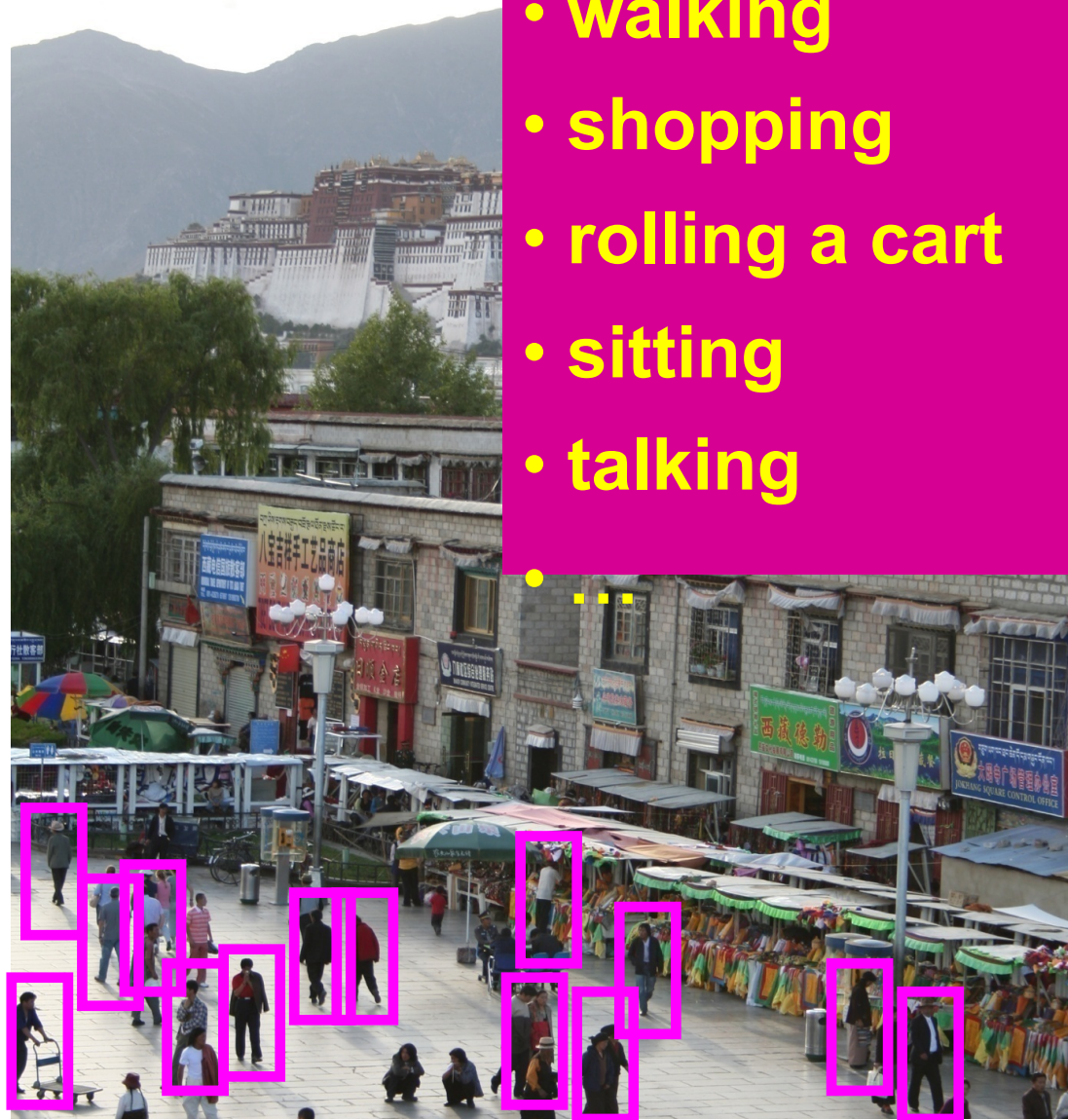
Object detection

- find pedestrians



Adapted from
Fei-Fei Li

Activity recognition



- walking
- shopping
- rolling a cart
- sitting
- talking

Adapted from
Fei-Fei Li



Semantic segmentation



Adapted from
Fei-Fei Li



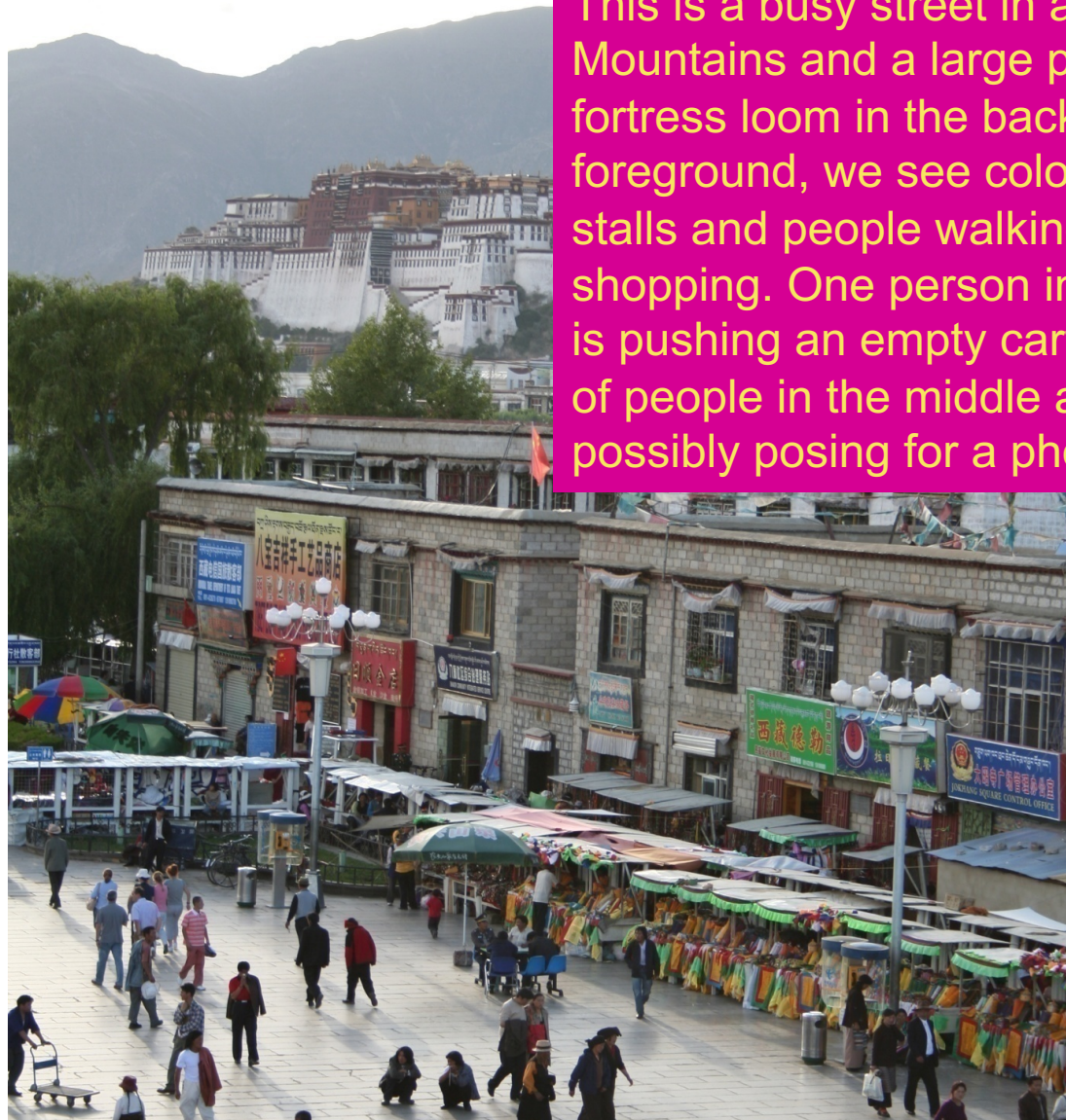
Semantic segmentation



Adapted from
Fei-Fei Li



Image description



This is a busy street in an Asian city. Mountains and a large palace or fortress loom in the background. In the foreground, we see colorful souvenir stalls and people walking around and shopping. One person in the lower left is pushing an empty cart, and a couple of people in the middle are sitting, possibly posing for a photograph.

Adapted from
Fei-Fei Li



Image classification



The statistical learning framework

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

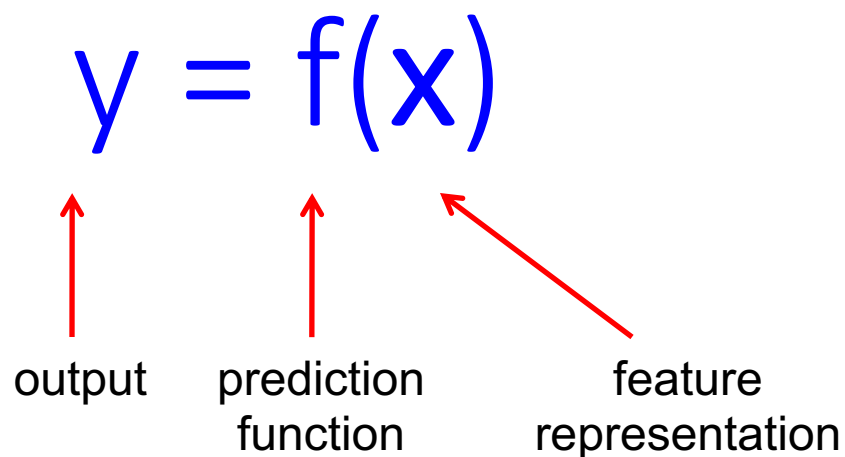
$$f(\text{apple image}) = \text{"apple"}$$

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

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



The statistical 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 $y = f(x)$



Training

Training Images



Image Features



Training Labels



Training



Learned model

Testing



Test Image



Image Features

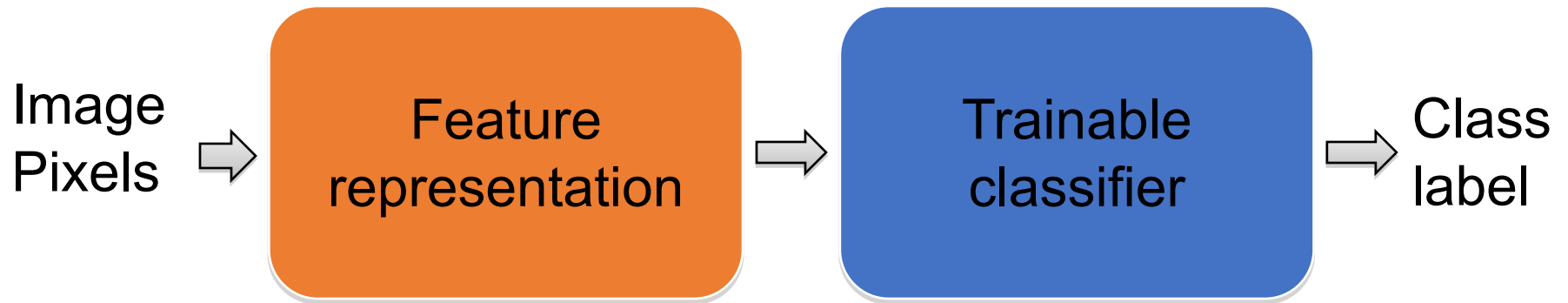


Prediction

Learned model



Creating a “Classic” recognition pipeline



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



Starting with a simpler problem

Document classification: Given a big collection of documents, quickly classify them into a set of categories

How would you do it in an afternoon ?



Motivation 1: Bags of words

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



Motivation 1: Bags of words

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

2007-01-23: State of the Union Address

George W. Bush (2001-)

abandon accountable affordable afghanistan africa aided ally anbar armed army **baghdad** bless **challenges** chamber chaos
choices civilians coalition commanders **commitment** confident confront congressman constitution corps debates deduction
deficit deliver **democratic** deploy dikembe diplomacy disruptions earmarks **economy** einstein **elections** eliminates
expand **extremists** failing faithful families **freedom** fuel **funding** god haven ideology immigration impose
insurgents iran **iraq** islam julie lebanon love madam marine math medicare moderation neighborhoods nuclear offensive
palestinian payroll province pursuing **qaeda** radical regimes resolve retreat rieman sacrifices science sectarian senate
september shia stays strength students succeed sunni **tax** territories **terrorists** threats uphold victory
violence violent **war** washington weapons wesley

US Presidential Speeches Tag Cloud
<http://chir.ag/projects/preztags/>



Motivation 1: Bags of words

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



US Presidential Speeches Tag Cloud
<http://chir.ag/projects/preztags/>



Motivation 1: Bags of words

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



US Presidential Speeches Tag Cloud
<http://chir.ag/projects/preztags/>

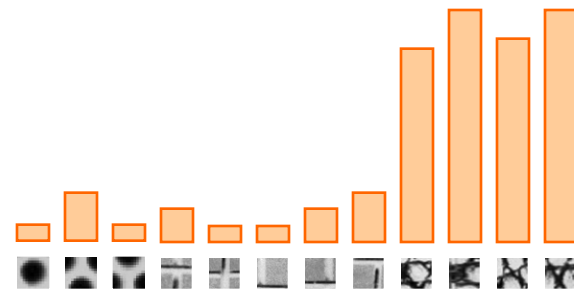
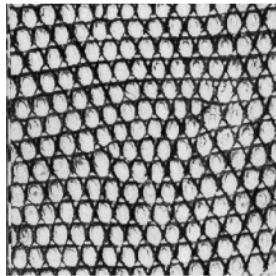
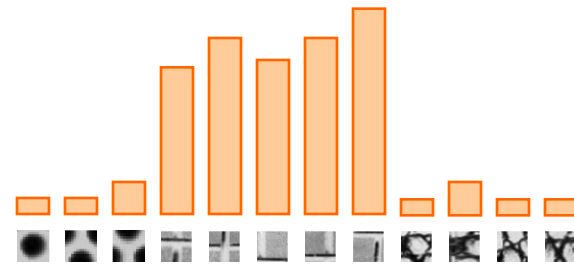
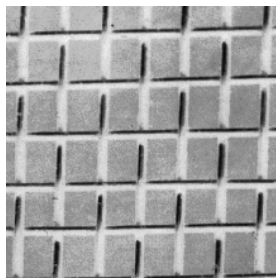
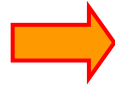
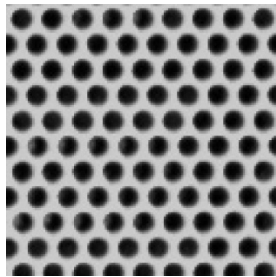


How to apply the same idea to images?

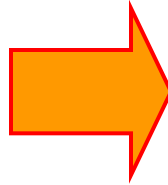
What are “visual words” ?



Motivation 2: Texture models

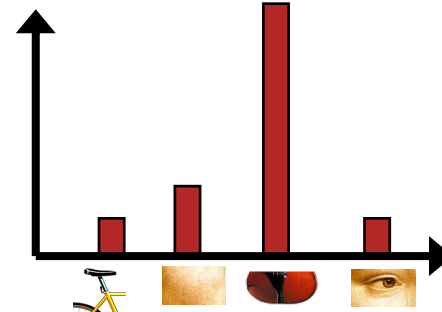
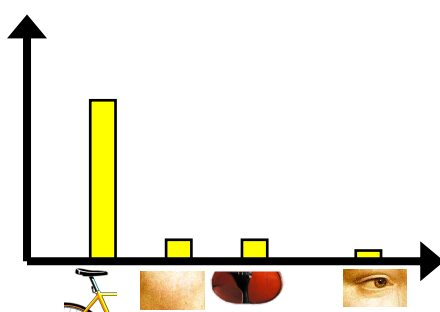
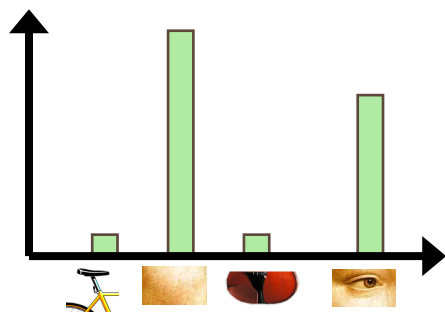
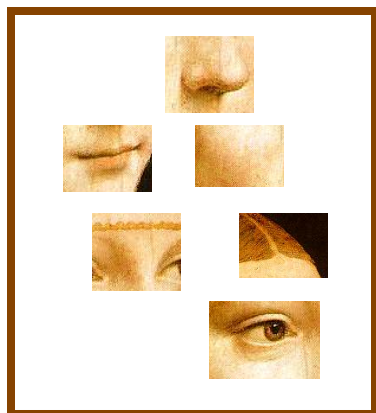


“Classic” representation: Bag of features



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

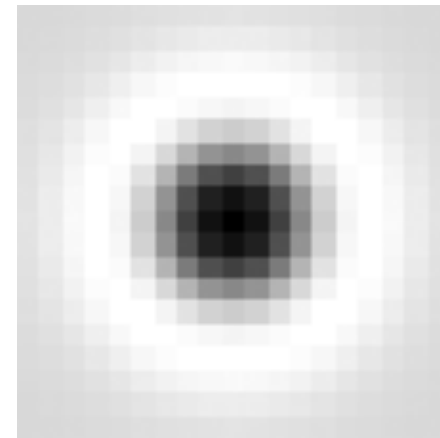
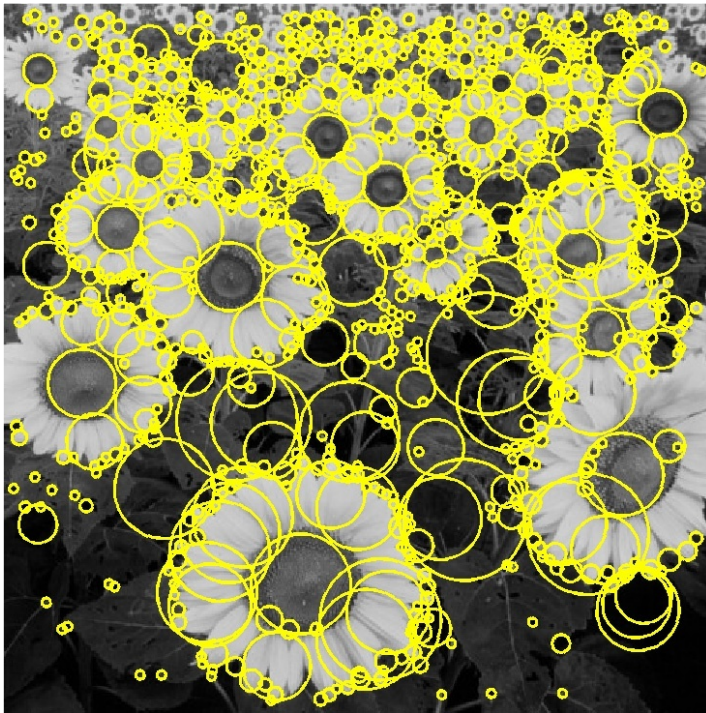


- We want to extract keypoints with *characteristic scales* that are *covariant* w.r.t. the image transformation



One idea: scale-invariant feature transform (SIFT)

- Convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting *scale space*



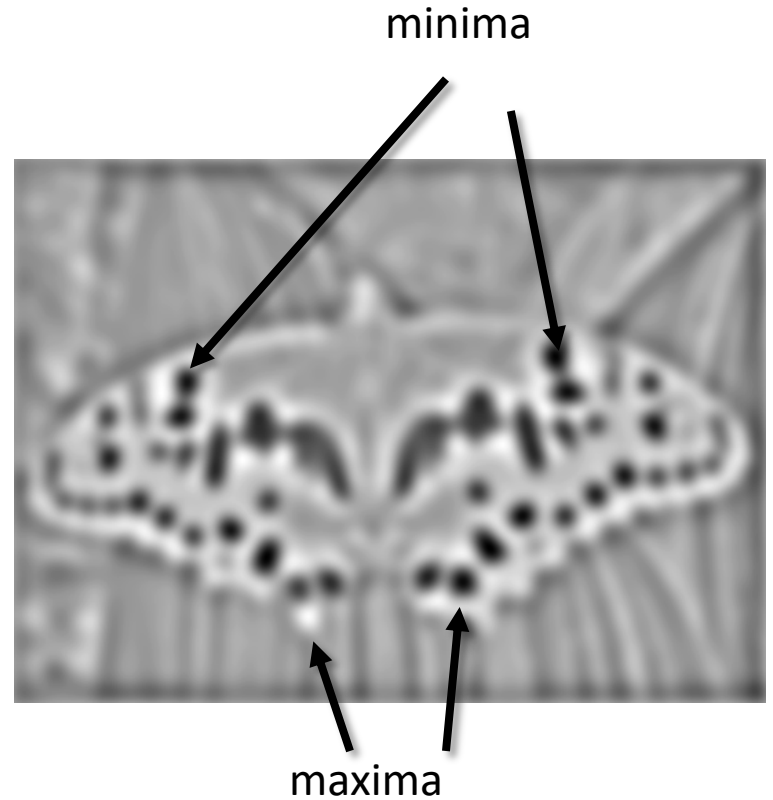
T. Lindeberg, [Feature detection with automatic scale selection](#),
IJCV 30(2), pp 77-116, 1998



Blob detection



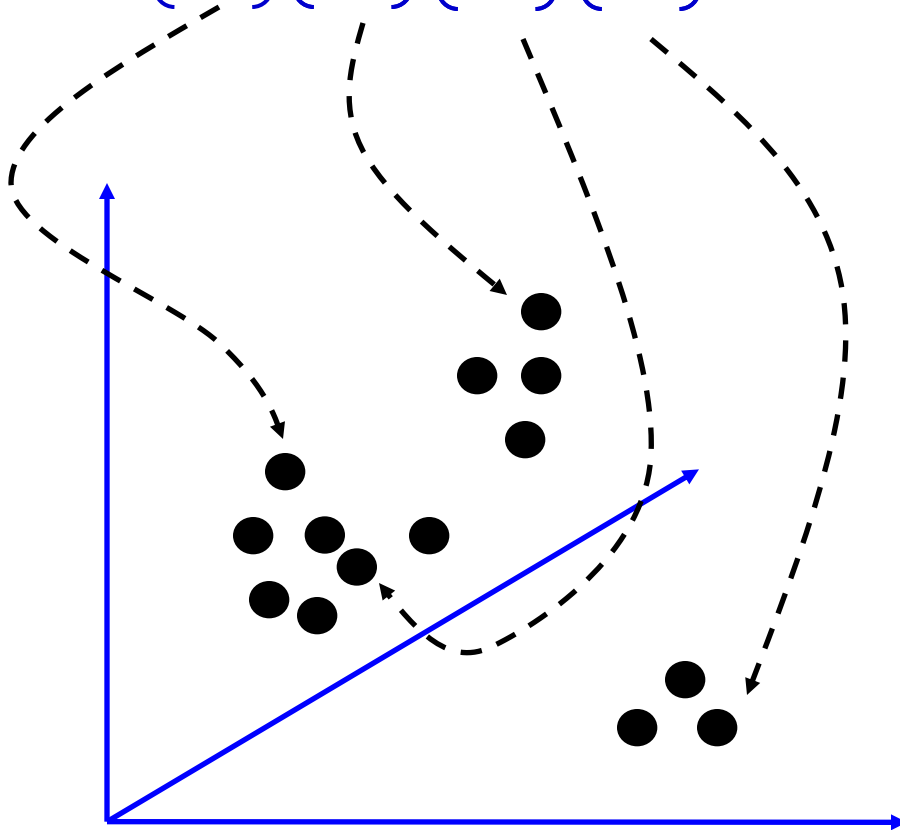
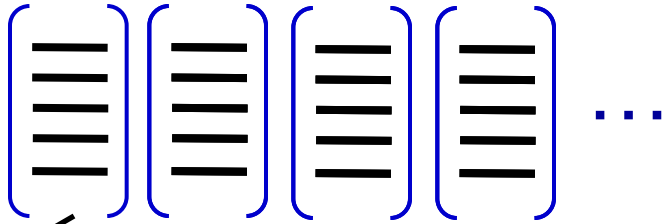
$$* \quad \text{[blob filter kernel]} \quad =$$



Find maxima *and minima* of blob filter response in space *and scale*



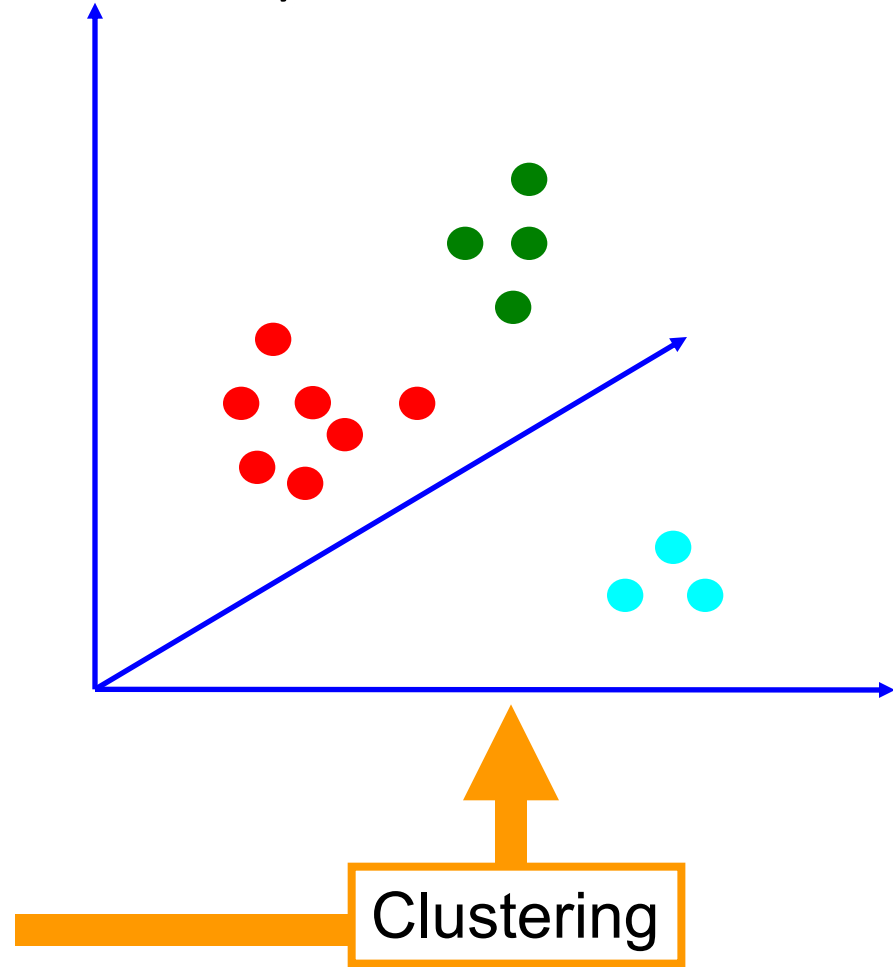
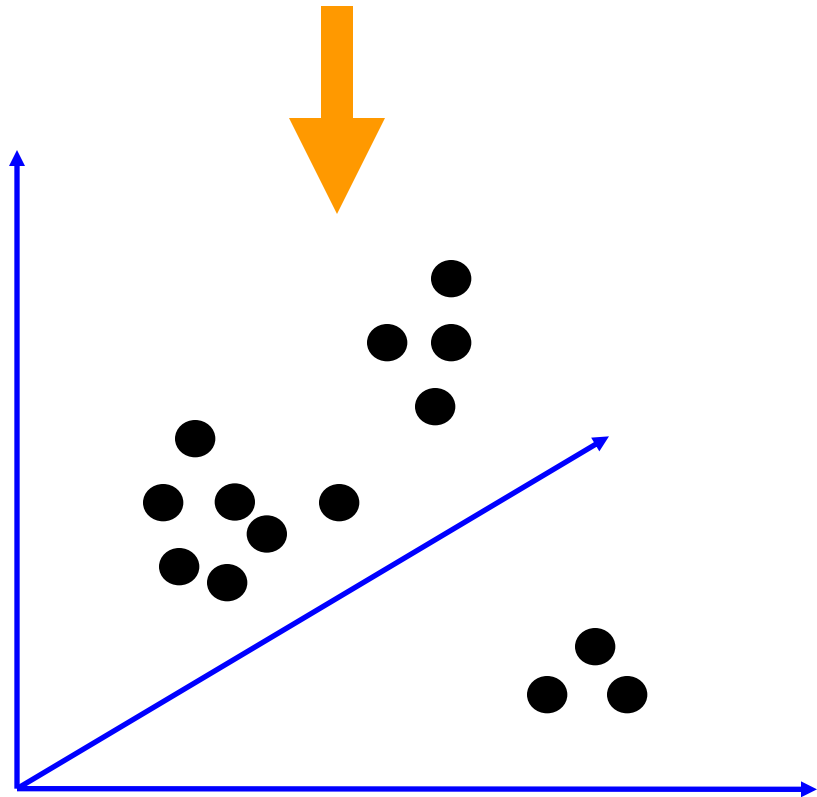
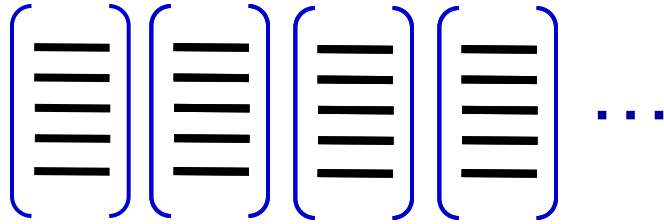
2. Learning the visual vocabulary



From each image patch in the training set, extract descriptors (e.g., HOG, SIFT). This gives a vector in \mathbb{R}^n which is used for clustering



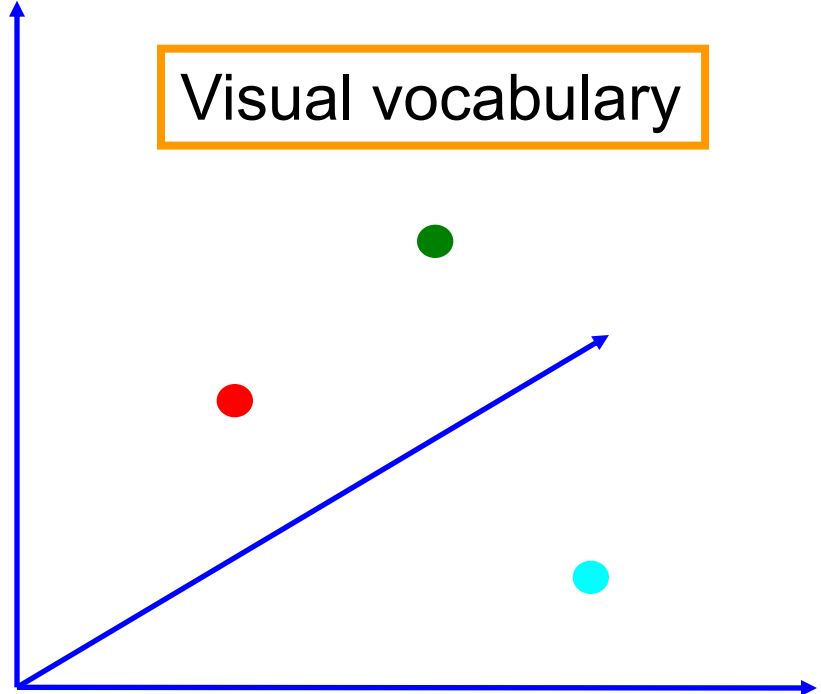
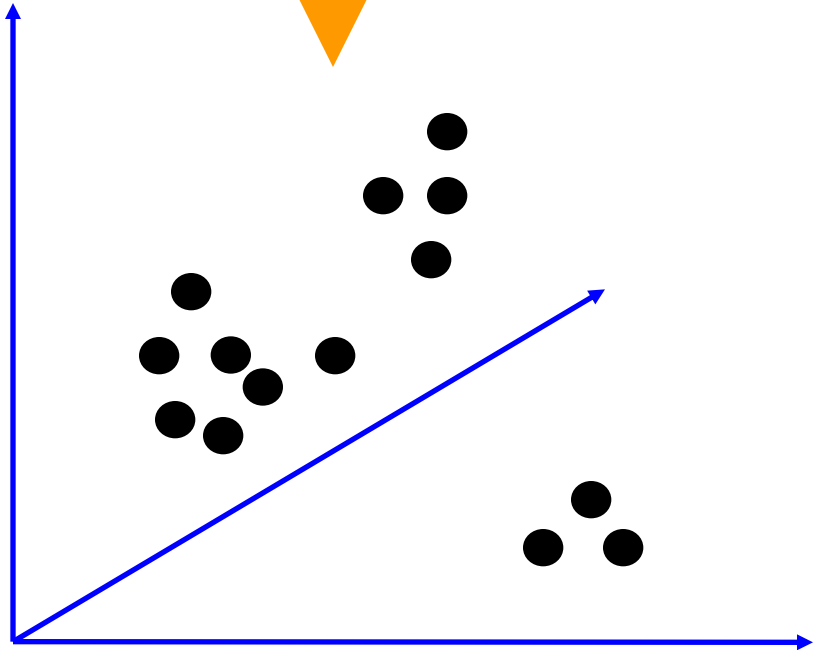
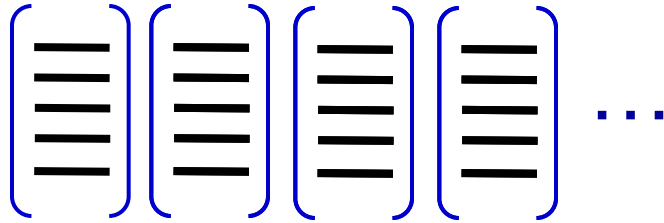
2. Learning the visual vocabulary



Slide credit: Josef Sivic



2. Learning the visual vocabulary



Slide credit: Josef Sivic



Recall: K-means clustering (Lecture 1)

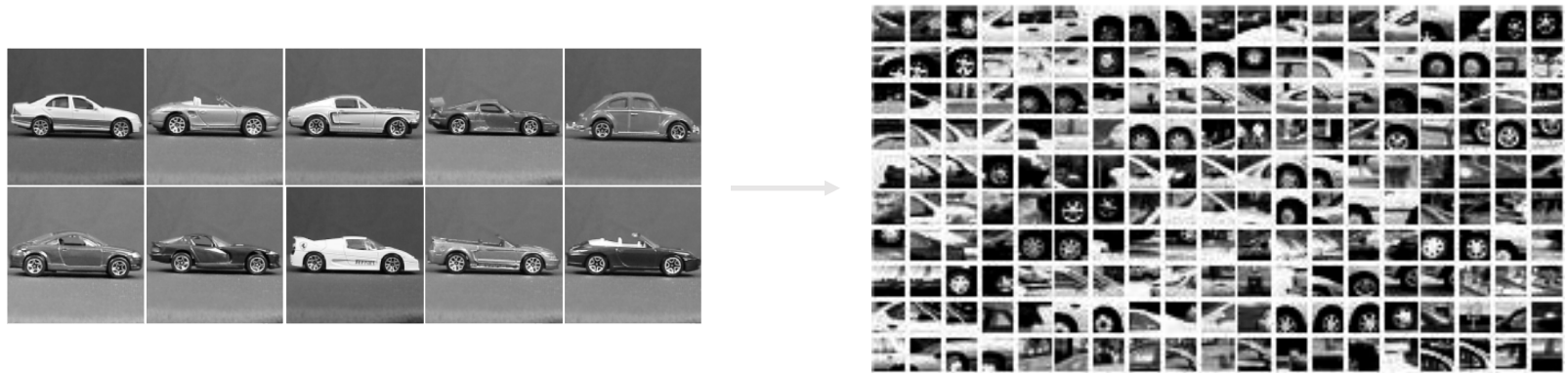
- Want to minimize sum of squared Euclidean distances between features \mathbf{x}_i and their nearest cluster centers \mathbf{m}_k

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

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



Recall: Visual vocabularies from cluster centers

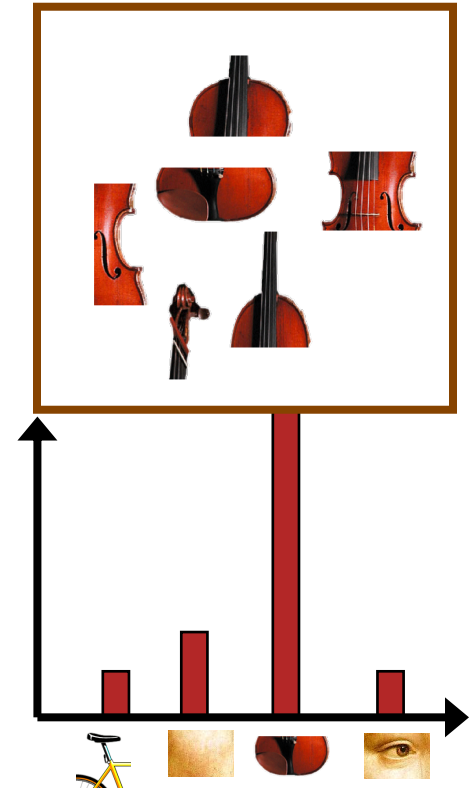
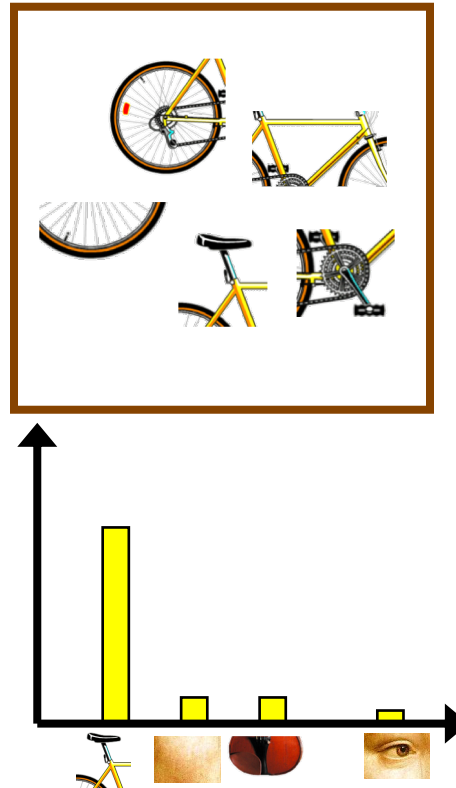
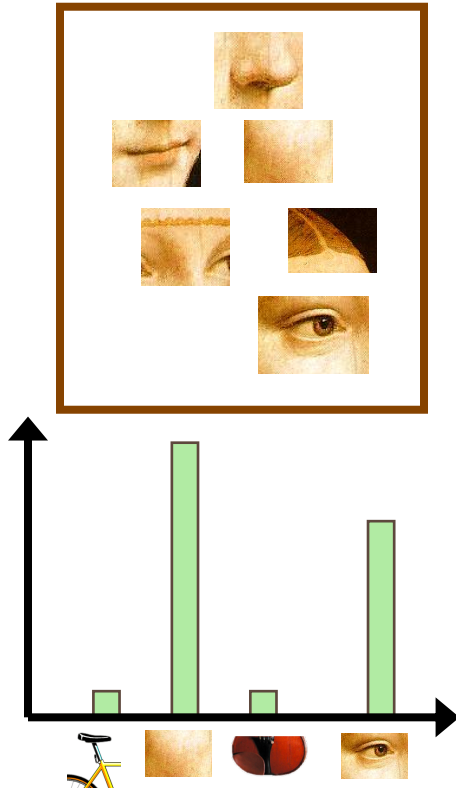


Appearance codebook



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”



Example visual vocabulary

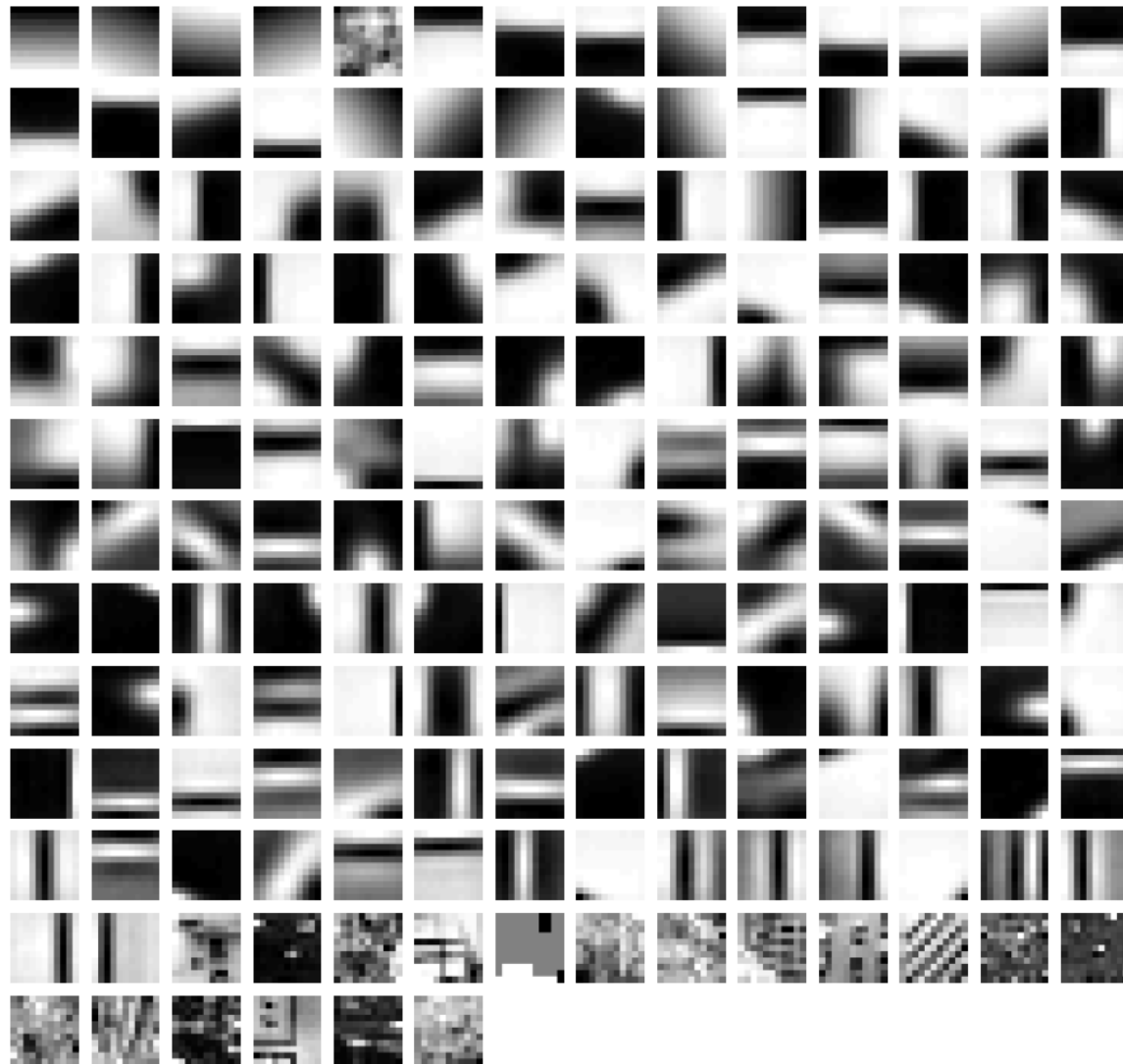
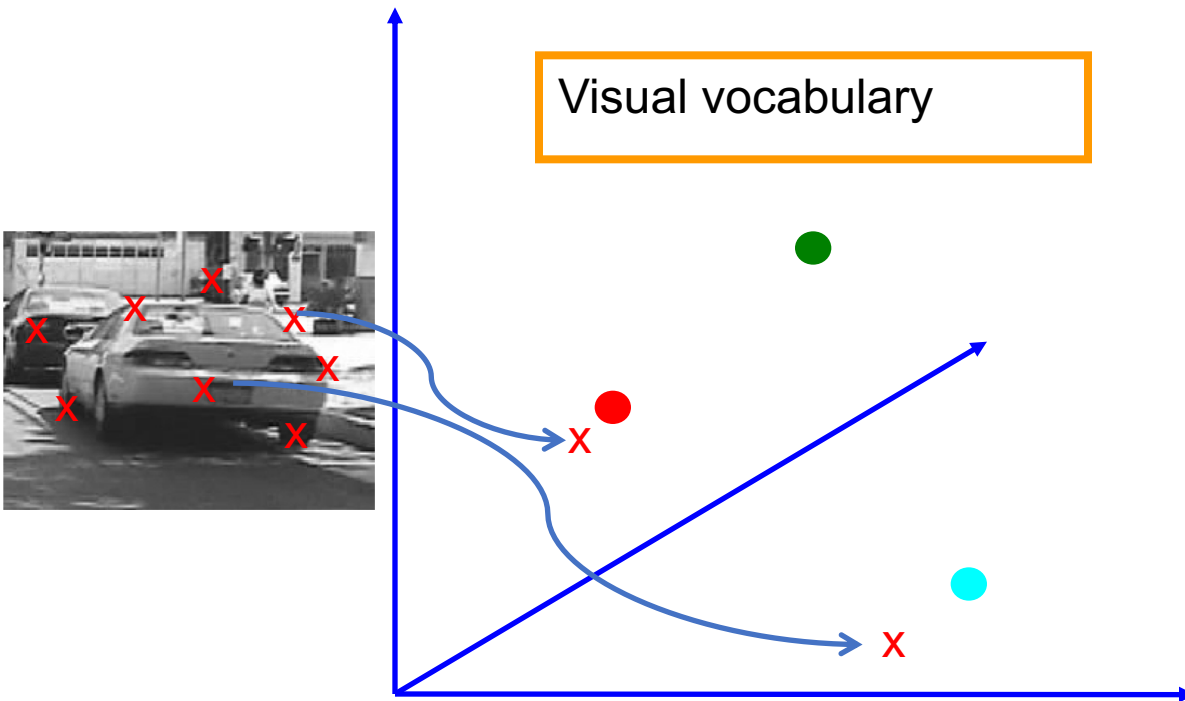


Image Representation

- For a query image



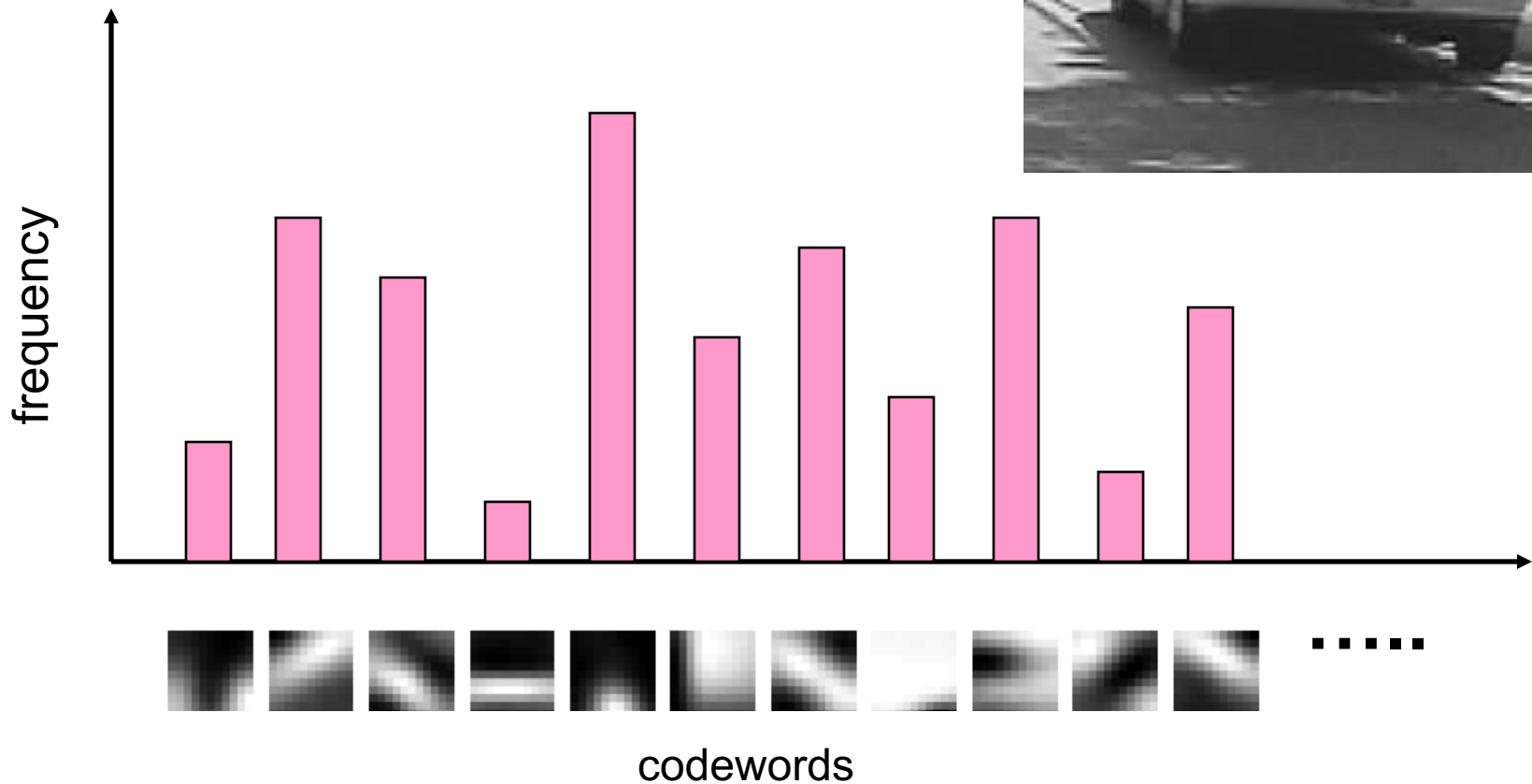
Extract features

Associate each feature with the nearest cluster center (visual word)

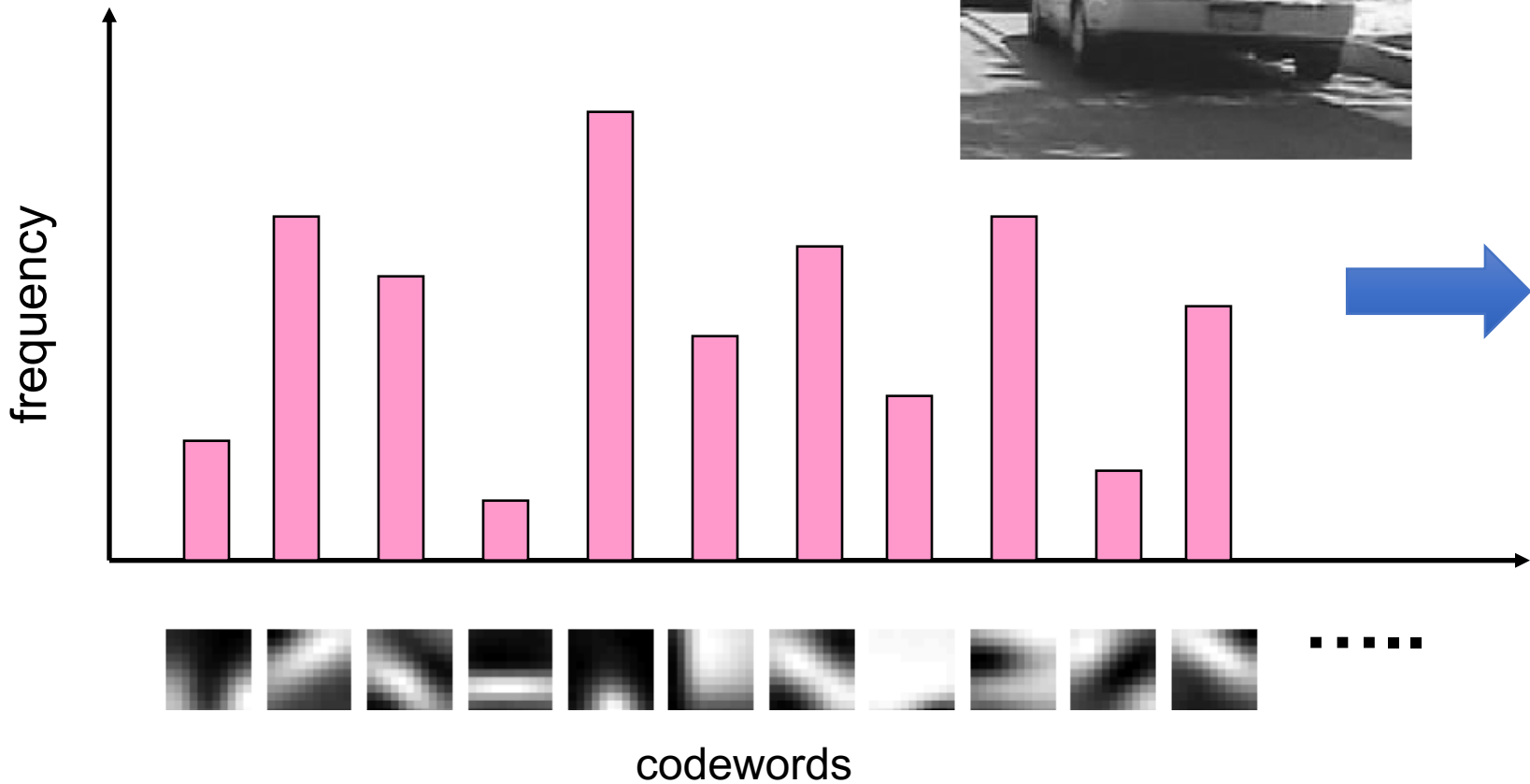
Accumulate visual word frequencies over the image



3. Image representation



4. Image classification



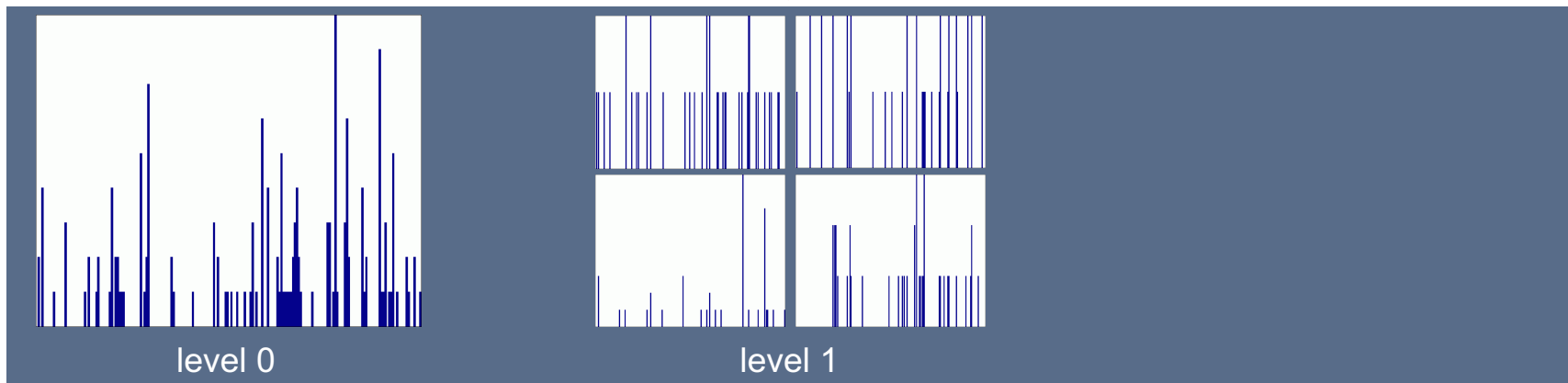
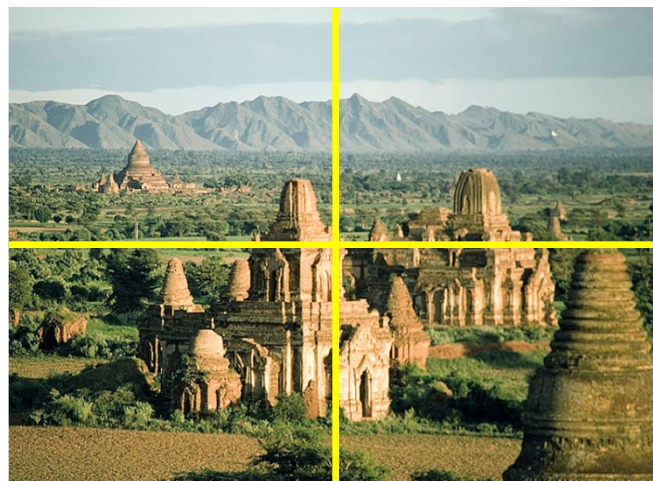
Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?



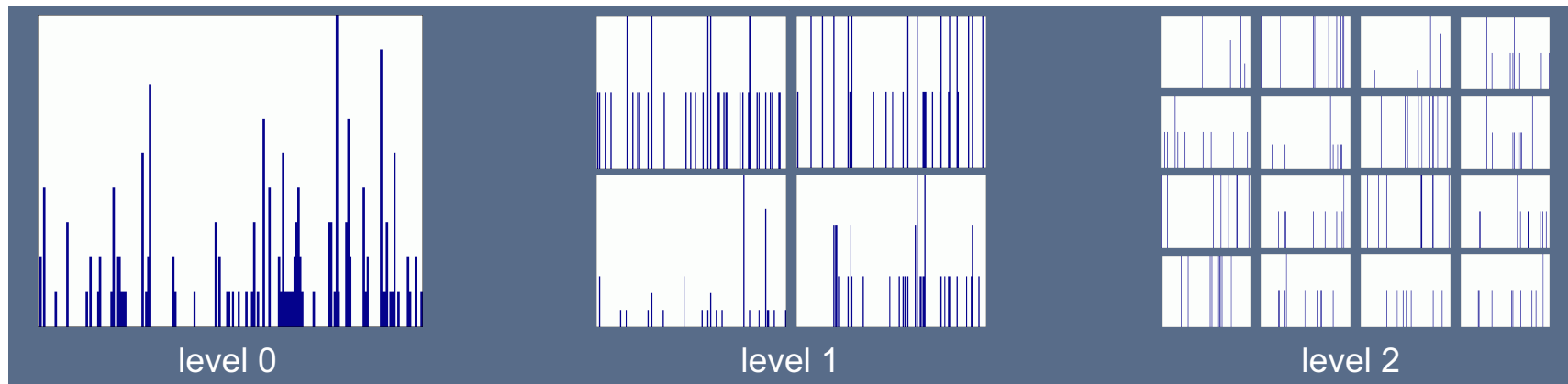
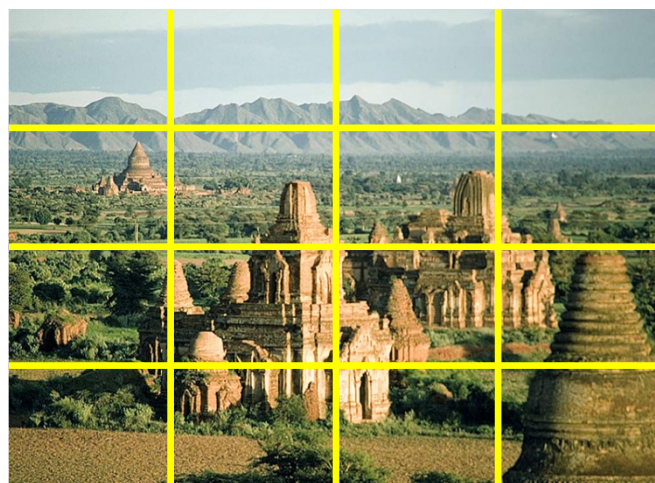
Spatial pyramids (orderless -> locally orderless)



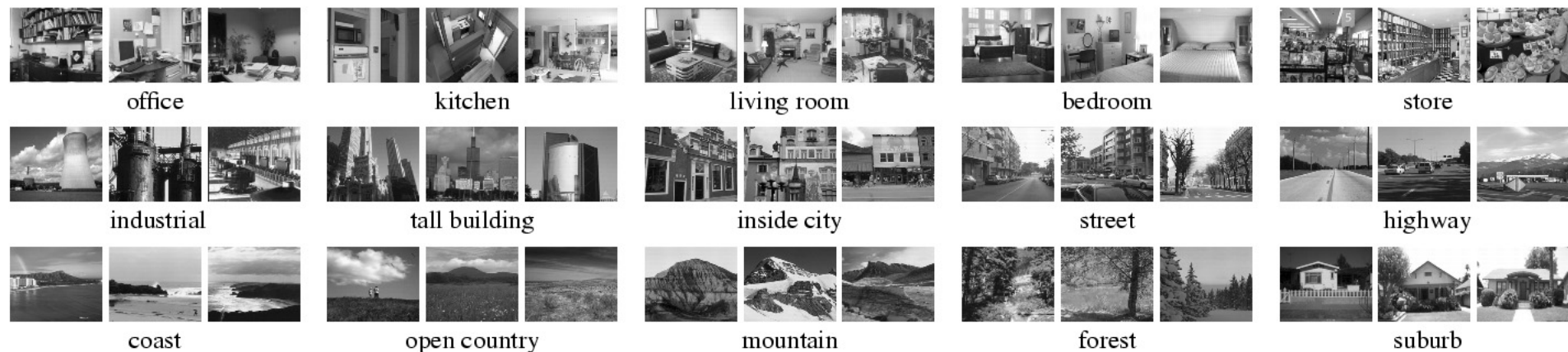
Spatial pyramids



Spatial pyramids



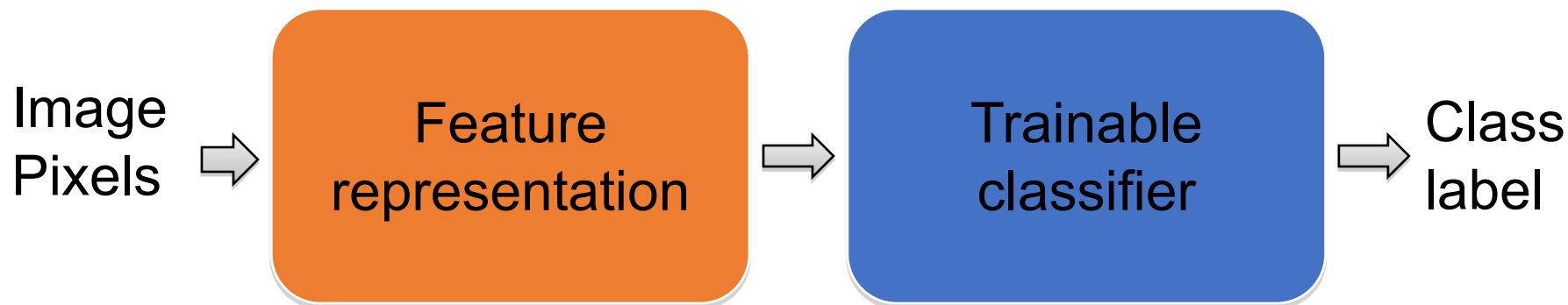
Spatial pyramids



Level	Weak features (vocabulary size: 16)		Strong features (vocabulary size: 200)	
	Single-level	Pyramid	Single-level	Pyramid
0 (1 × 1)	45.3 ±0.5		72.2 ±0.6	
1 (2 × 2)	53.6 ±0.3	56.2 ±0.6	77.9 ±0.6	79.0 ±0.5
2 (4 × 4)	61.7 ±0.6	64.7 ±0.7	79.4 ±0.3	81.1 ±0.3
3 (8 × 8)	63.3 ±0.8	66.8 ±0.6	77.2 ±0.4	80.7 ±0.3



“Classic” recognition pipeline



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

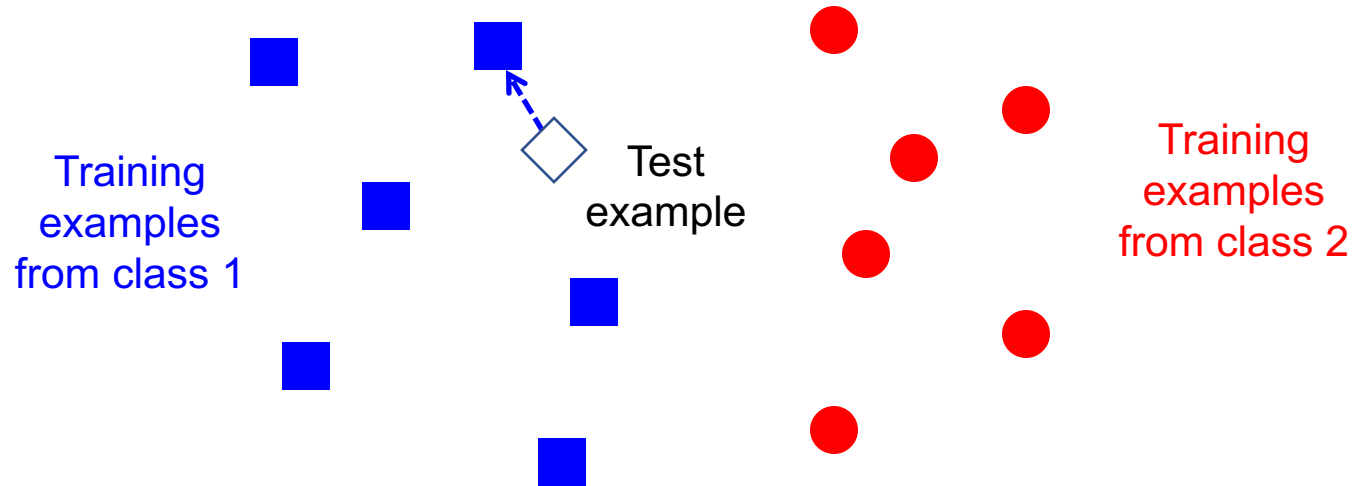


Classification

Given a new image, and the vector of trained visual word histograms, how to classify the new image?



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*):

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

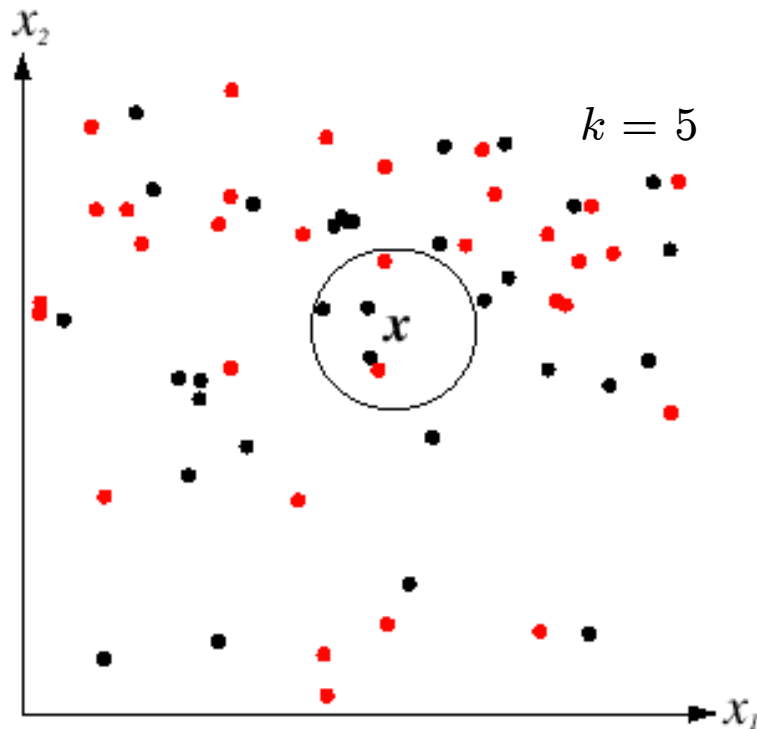
- Histogram intersection (similarity function):

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



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



Effects of scaling

- Some components of the vector with large values may influence the classification more than others

- Normalize vectors

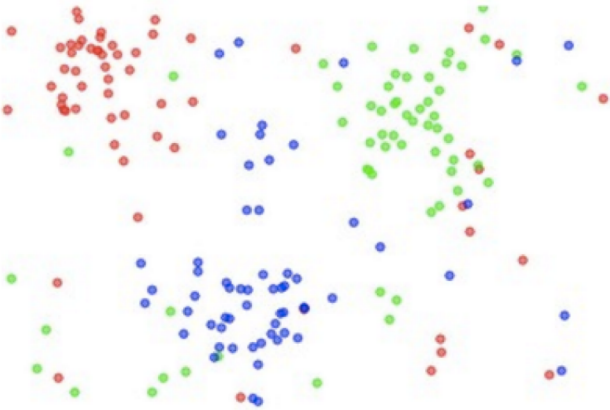
- x_{ij} value for the i^{th} sample and j^{th} feature
- μ_j mean of all x_{ij} for feature j
- σ_j standard deviation of all x_{ij} over all input samples

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

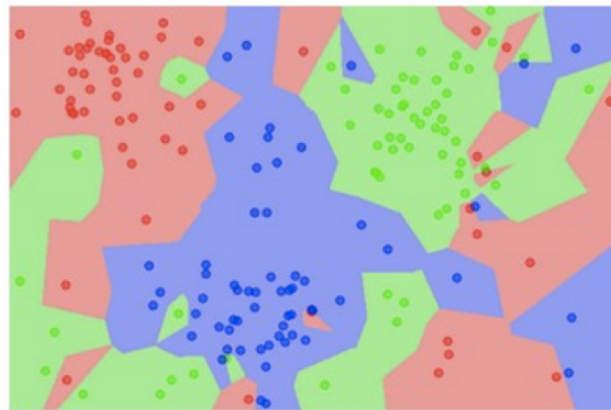


K-nearest neighbor classifier

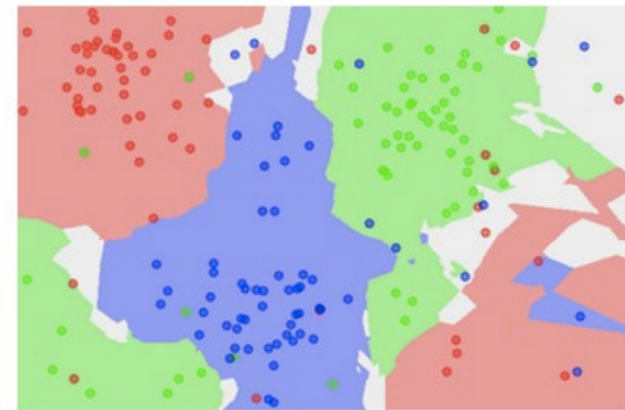
the data



NN classifier



5-NN classifier



- 2d points, and 3 classes. White regions are “ambiguous”
- Which classifier is more robust to *outliers*?

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



Hyperparameters for K-NN



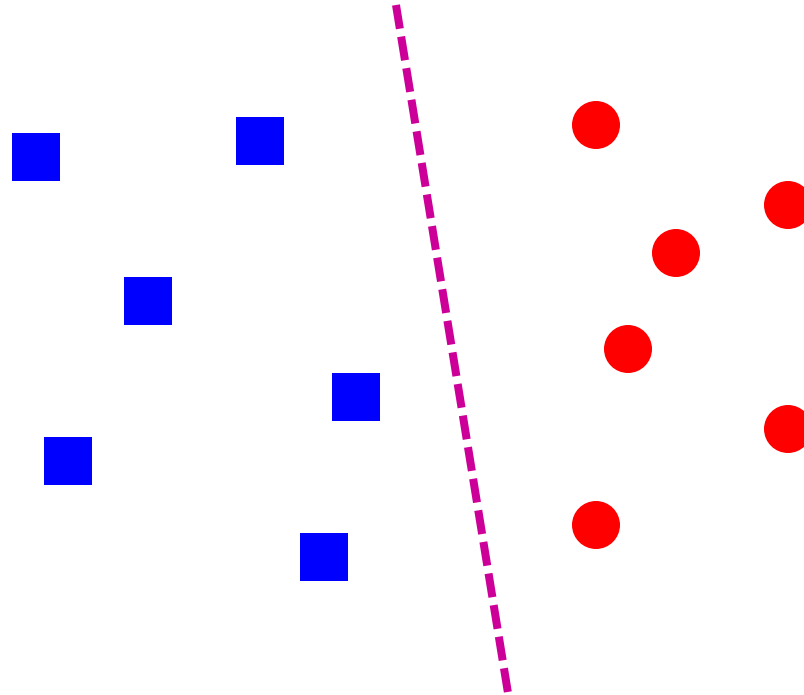
K-nearest neighbor classifier



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.



Linear classifiers

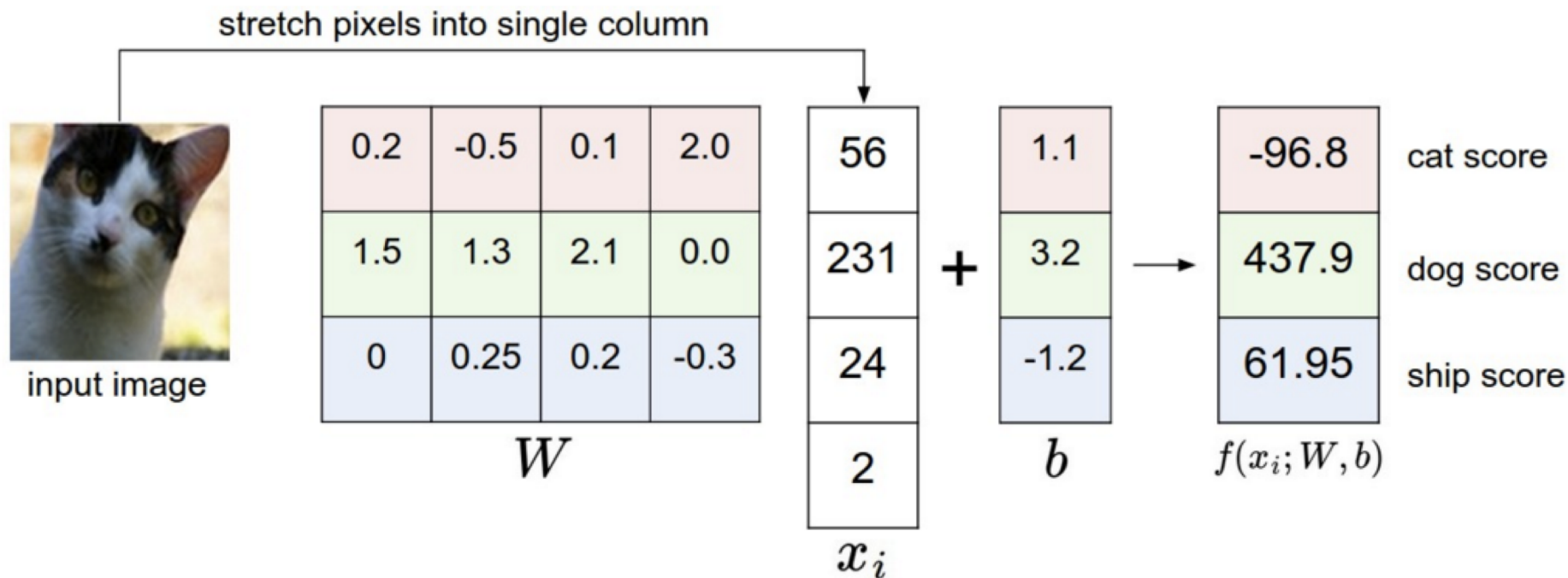


- Find a *linear function* to separate the classes:

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



Visualizing linear classifiers



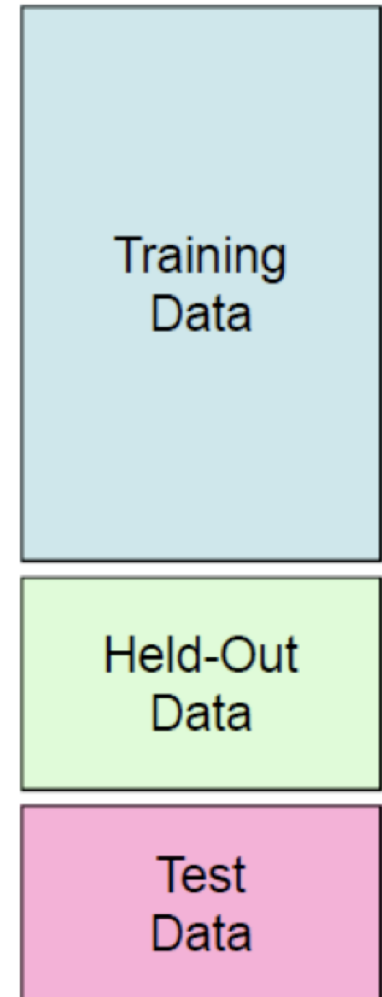
Nearest neighbor vs. linear classifiers

- **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?



Best practices for training classifiers

- Goal: obtain a classifier with **good generalization** or performance on never before seen data
 1. Learn *parameters* on the **training set**
 2. Tune *hyperparameters* (implementation choices) on the *held out validation set*
 3. Evaluate performance on the **test set**
 - Crucial: do not peek at the test set when iterating steps 1 and 2!




What's the big deal?

Baidu admits cheating in international supercomputer competition



Baidu recently apologised for violating the rules of an international supercomputer test in May, when the Chinese search engine giant claimed to beat both Google and Microsoft on the ImageNet image-recognition test.

 By [Cyrus Lee](#) | June 10, 2015 -- 00:15 GMT (17:15 PDT) | Topic: [China](#)

TECHNOLOGY

The New York Times

Computer Scientists Are Astir After Baidu Team Is Barred From A.I. Competition

By [JOHN MARKOFF](#) JUNE 3, 2015

Baidu caught gaming recent
supercomputer performance test



 by [Andrew Tarantola](#) | [@terrortola](#) | June 3rd 2015 At 11:09pm



IMAGENET Large Scale Visual Recognition Challenge (ILSVRC)

Date: June 2, 2015

Dear ILSVRC community,

This is a follow up to the announcement on [May 19, 2015](#) with some more details and the status of the test server.

During the period of November 28th, 2014 to May 13th, 2015, there were at least 30 accounts used by a team from Baidu to submit to the test server at least 200 times, far exceeding the specified limit of two submissions per week. This includes short periods of very high usage, for example with more than 40 submissions over 5 days from March 15th, 2015 to March 19th, 2015. Figure A below shows submissions from ImageNet accounts known to be associated with the team in question. Figure B shows a comparison to the activity from all other accounts.

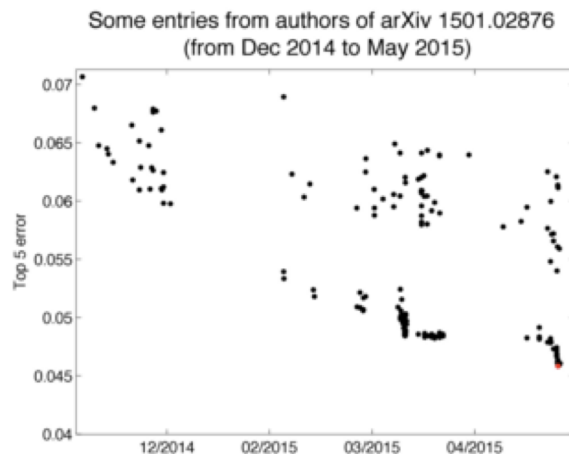


Figure A

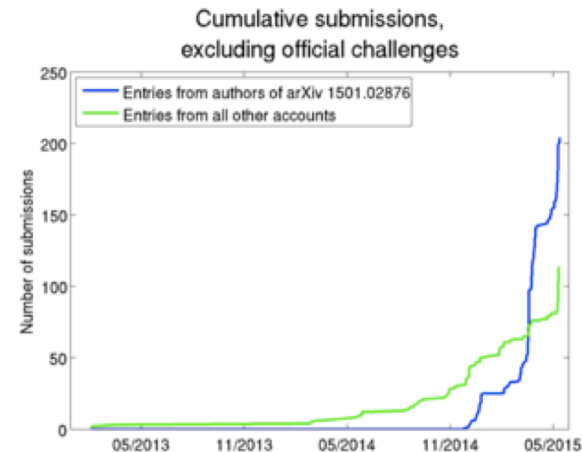


Figure B

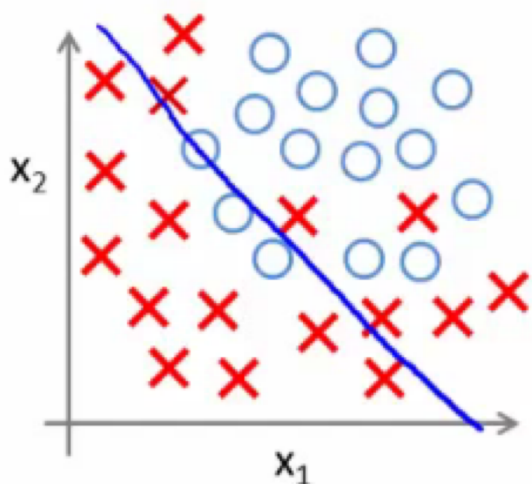
The results obtained during this period are reported in a [recent arXiv paper](#). Because of the violation of the regulations of the test server, these results may not be directly comparable to results obtained and reported by other teams. To make this clear, by exploiting the ability to test many slightly different solutions on the test server it is possible to 1) select the best out of a set of very similar solutions based on test performance and achieve a small but potentially significant advantage and 2) choose methods for further research and development based directly on the test data instead of using only the training and validation data for such choices.



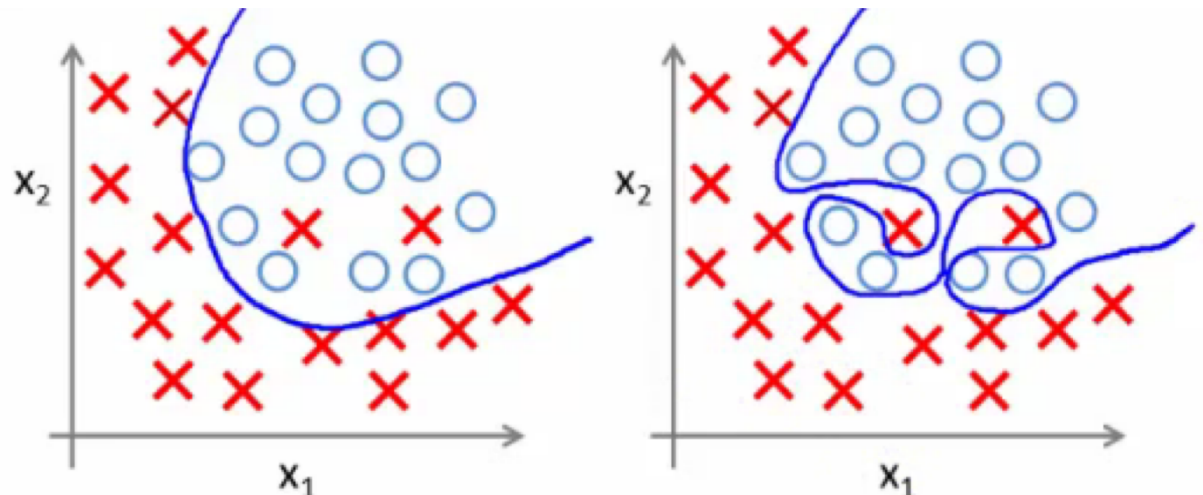
Bias-variance tradeoff

- Prediction error of learning algorithms has two main components:
 - **Bias**: error due to simplifying model assumptions
 - **Variance**: error due to randomness of training set
- **Bias-variance tradeoff** can be controlled by turning “knobs” that determine model complexity

High bias, low variance



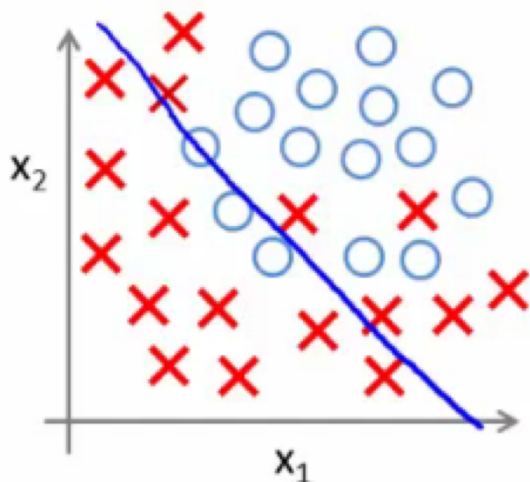
Low bias, high variance



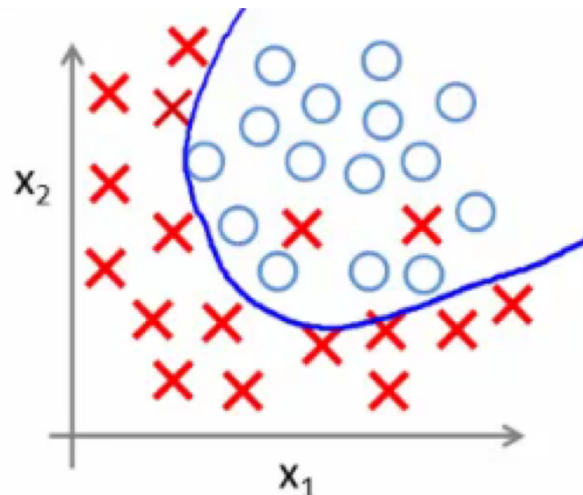
Underfitting and overfitting

- **Underfitting:** training and test error are both *high*
 - Model does an equally poor job on the training and the test set
 - The model is too “simple” to represent the data or the model is not trained well
- **Overfitting:** Training error is *low* but test error is *high*
 - Model fits irrelevant characteristics (noise) in the training data
 - Model is too complex or amount of training data is insufficient

Underfitting



Good tradeoff



Overfitting

