מבוא לראייה ממוחשבת – 22928
2016א

מנחה: אמיר אגוזי
egozi5@gmail.com

מפגש מס' 3

מה ראינו בפעם שעברה?

- Model fitting
  - Least squares /total least squares
- Voting
  - Hough transform/generalized Hough transform
- RANSAC

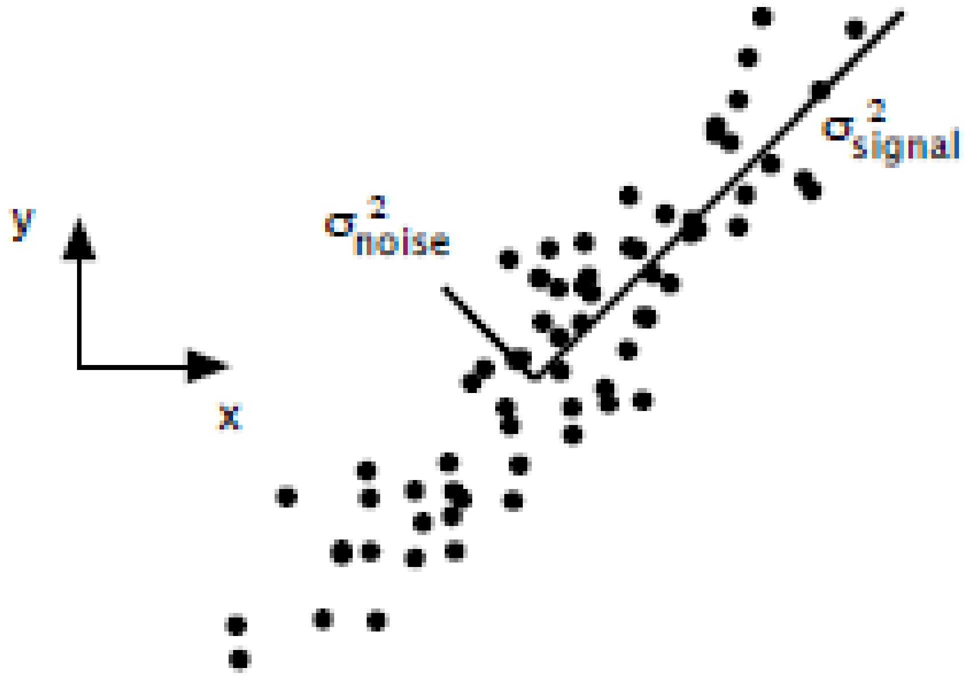
מה היום?

- Data analysis
  - הורדת מימד לינארית - PCA
  - סיווג – Classification

Dimensionality reduction

- Reducing the number of random variables under consideration (curse of dimensionality).
- Projection of high-dimensional data to a low-dimensional space that preserves the "important" characteristics of the data.
- Visualize high-dimensional data in low-dimensional space.

Principle component analysis (PCA)



Principle component analysis (PCA)

- Given $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^d$
  - Compute sample mean: $\hat{\mu} = \frac{1}{n} \sum_i x_i$
  - Compute sample covariance.

$$\hat{\Sigma} = \frac{1}{n-1} \sum_i (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T$$

  - Compute eigen-decomposition of $\hat{\Sigma}$

$$\hat{\Sigma} = U \Lambda U^T, \quad \Lambda = (\sigma_1^2, \ldots, \sigma_n^2), U^T U = I$$

  - Create projection matrix *P* from *k* “top” eigenvector

$$P = \begin{bmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_k^T \end{bmatrix}$$

Principle component analysis (2)

- Given new dataset $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}, z \in \mathbb{R}^d$
- Project according to:

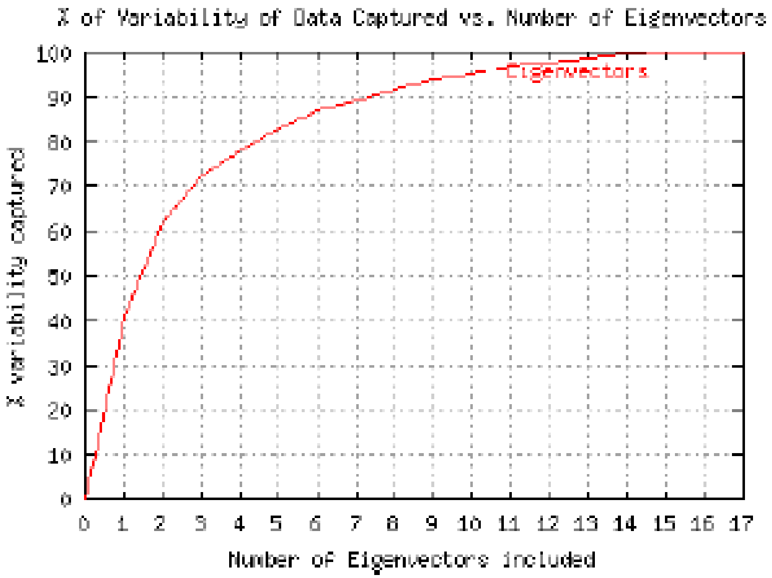
$$\tilde{\mathbf{z}} = P \cdot (\mathbf{z} - \hat{\mu})^T$$

- Use $\tilde{\mathcal{Z}} = \{\tilde{\mathbf{z}}_1, \ldots, \tilde{\mathbf{z}}_m\}, z \in \mathbb{R}^k$ to all further processings.

Note the new dimension, usually: k << d

Principal component analysis

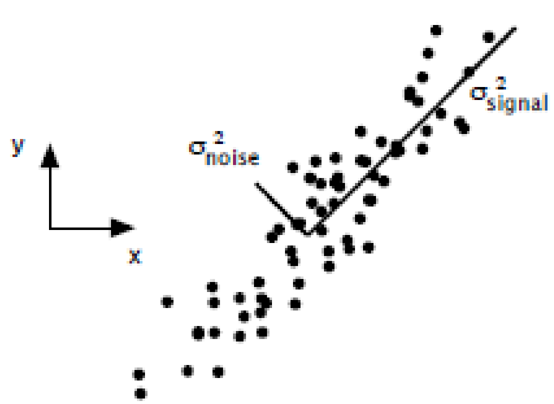
- a natural measure is to pick the eigenvectors that explain X% of the data variability
  - can be done by plotting the ratio r_k as a function of k

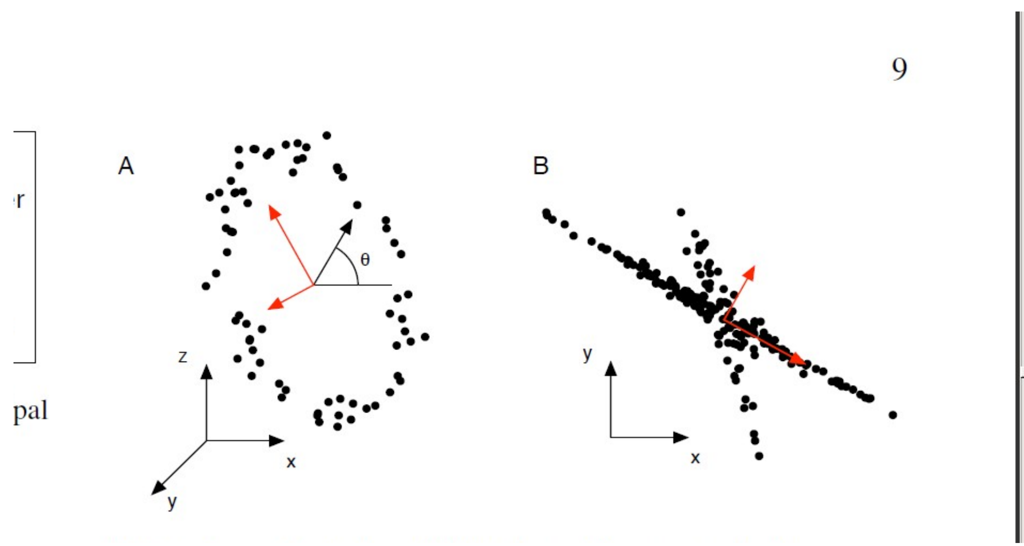


$$r_k = \frac{\sum_{i=1}^{k} \lambda_i^2}{\sum_{i=1}^{n} \lambda_i^2}$$

PCA - conclusions

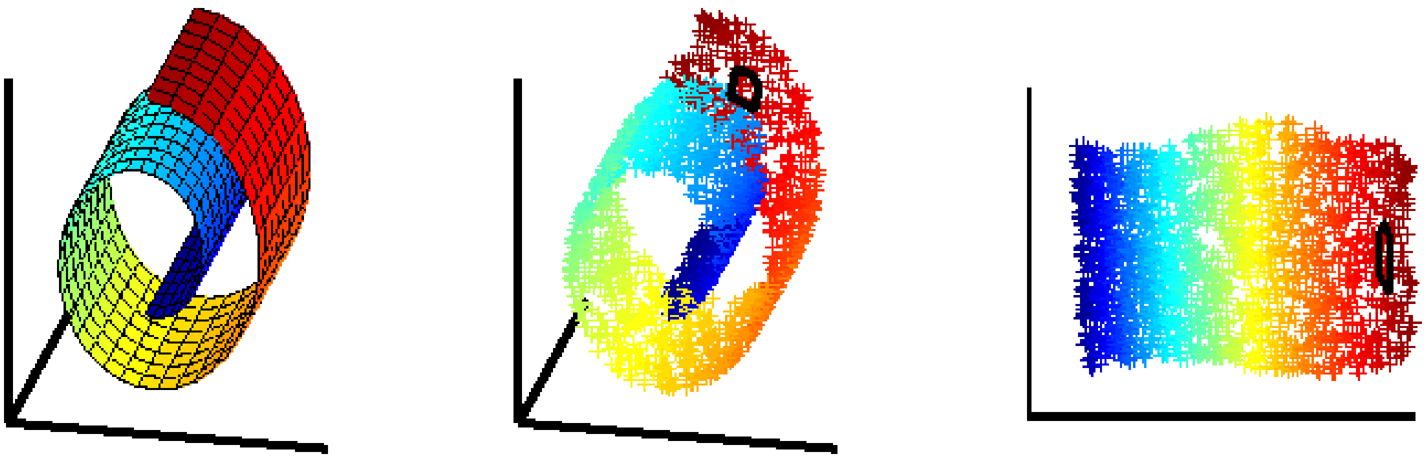
- Simple and intuitive dimensionality reduction approach
- Not always keeping large variance is a good idea.
- Not all datasets exist in a **linear** subspace.
- But, practically...





Issues: dimensionality

What if your space isn't *flat*?



Nonlinear methods
LLE, MDS, etc.

Feature extraction

- PCA is a feature extraction technique.
- There are many many more.
- For a review on state-of-the-art techniques:
  http://en.wikipedia.org/wiki/Feature_extraction

Classification – Pattern recognition

Subproblems of Pattern classification

- Feature extraction / representation
- Noise
- Over-fitting
- Model selection
- Prior knowledge
- Missing features
- Segmentation
- Context
- Invariances
- Cost and risks
- Computational complexity

From Duda & Hart sec. 1.3

Machine learning

- **Supervised learning** - Learn from examples, ((input, output) pairs)
  - **Regression**: predict a continuous value as a function of the input.
  - **Classification**: predict the class label of the input object
- **Unsupervised learning** - fit a model to observations without a priori output.
  - Clustering
  - Identifying patterns in the data.

Unsupervised Learning

- Input – unlabeled data samples $\{\mathbf{x}_i\}, i = 1, \ldots, n$
- Why study unlabeled data?
  - Labeled data can be costly
  - Cluster first, label later
  - Identify features that will be useful for categorization
  - Exploratory data analysis

Supervised learning

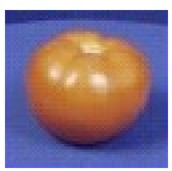
Recognition: A machine learning approach

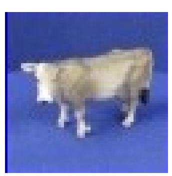
Slides adapted from Fei-Fei Li, Rob Fergus, Antonio Torralba, Kristen Grauman, and Derek Hoiem

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”

The machine learning framework

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

output

prediction function

Image feature

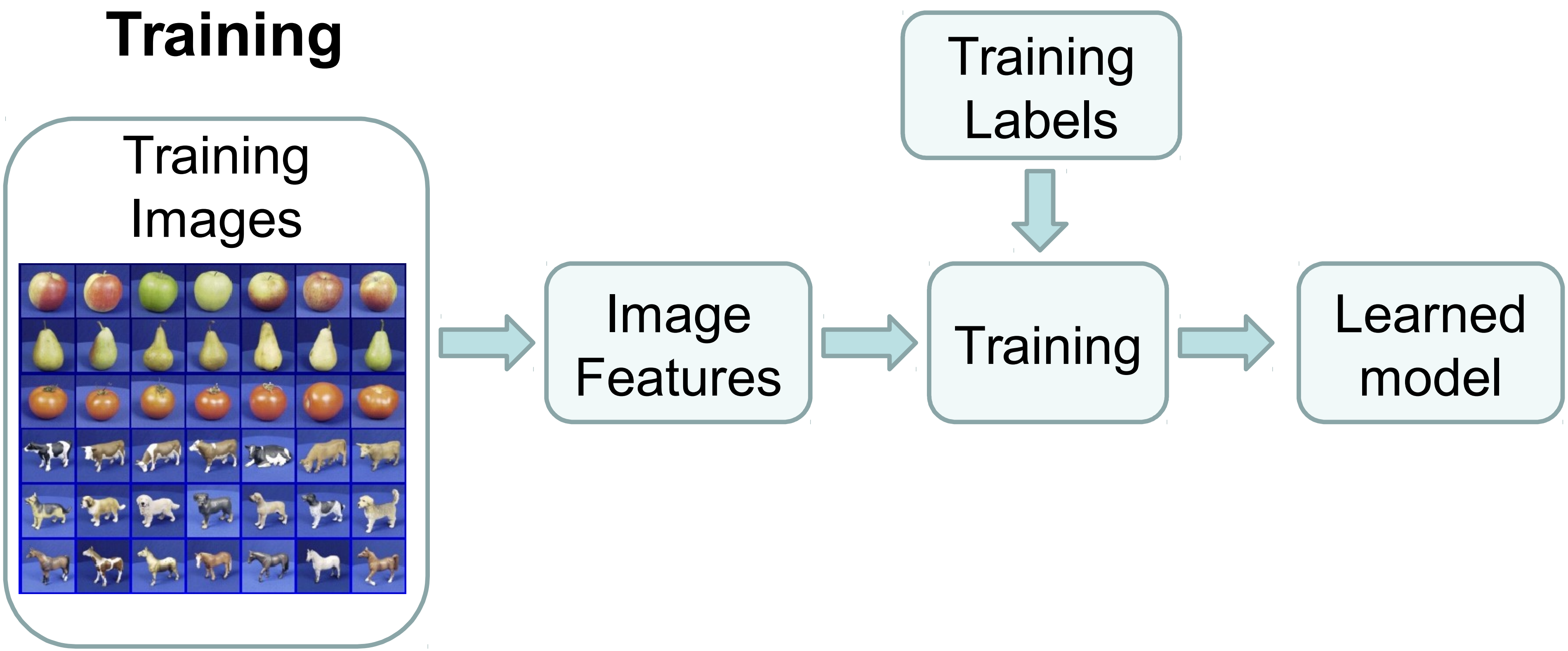
Training: given a *training set* of labeled examples

$$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$$

estimate the prediction function $f(\mathbf{x})$ 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 $y = f(\mathbf{x})$

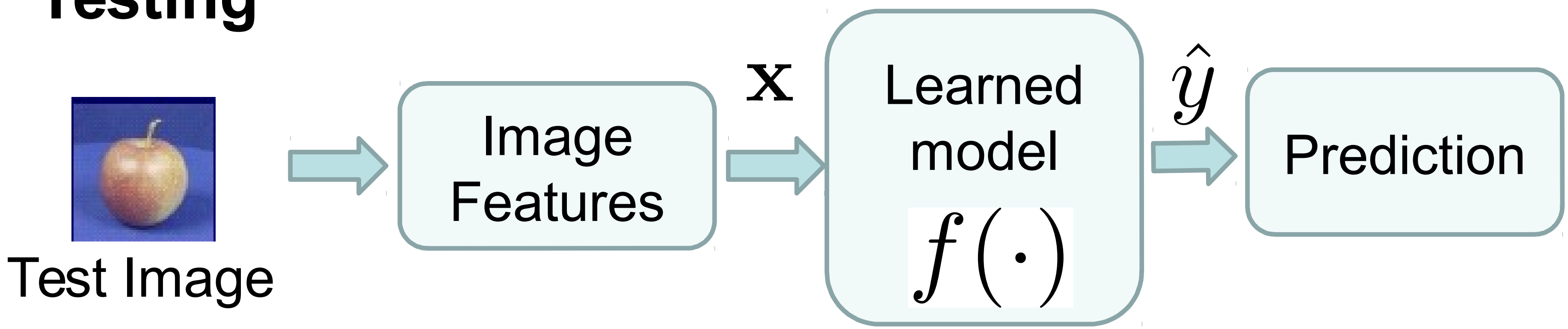
Step I - Learning



Slide credit: D. Hoiem

Step II - Test

Testing

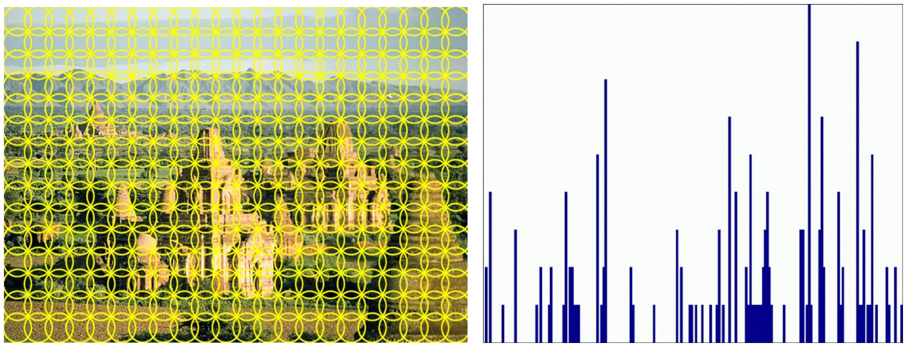


Slide credit: D. Hoiem

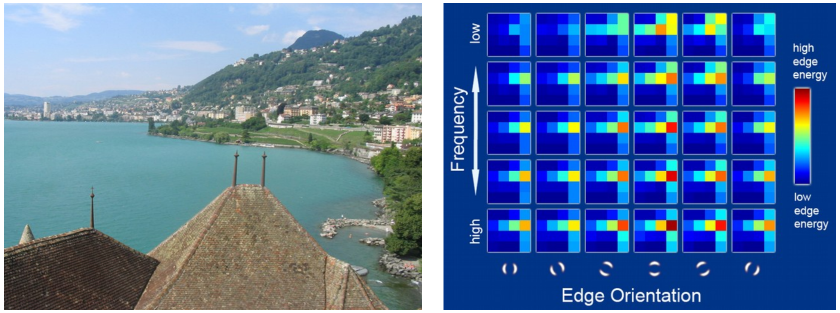
Features (examples)

- Raw pixels (and simple functions of raw pixels)

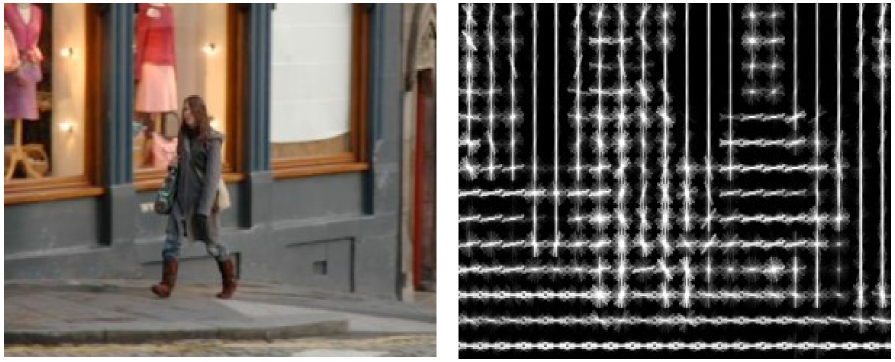
- Histograms, bags of features



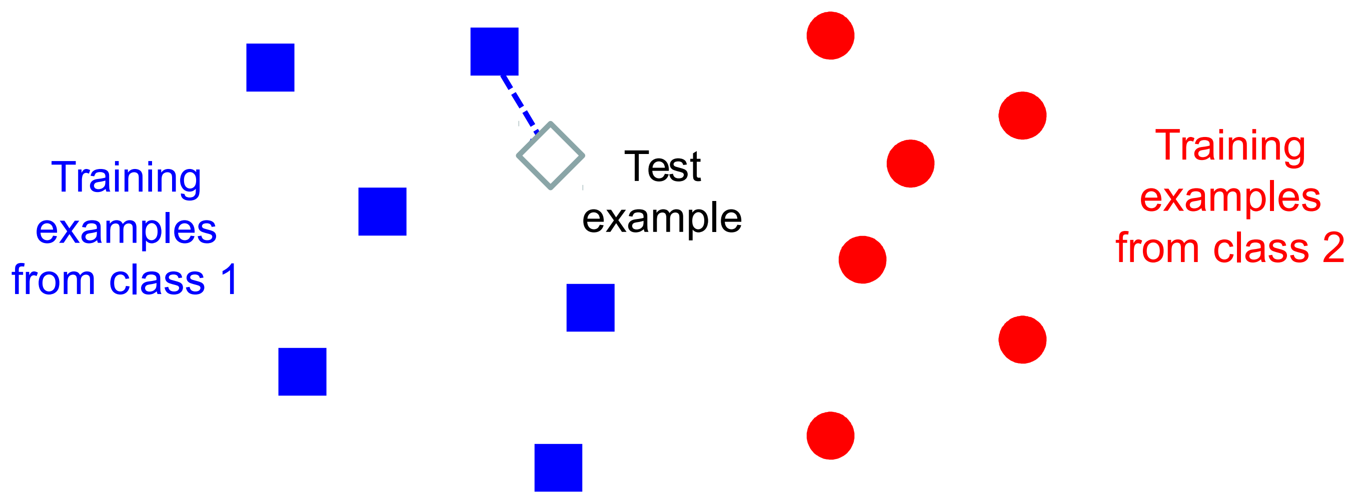
- GIST descriptors



- Histograms of oriented gradients (HOG)



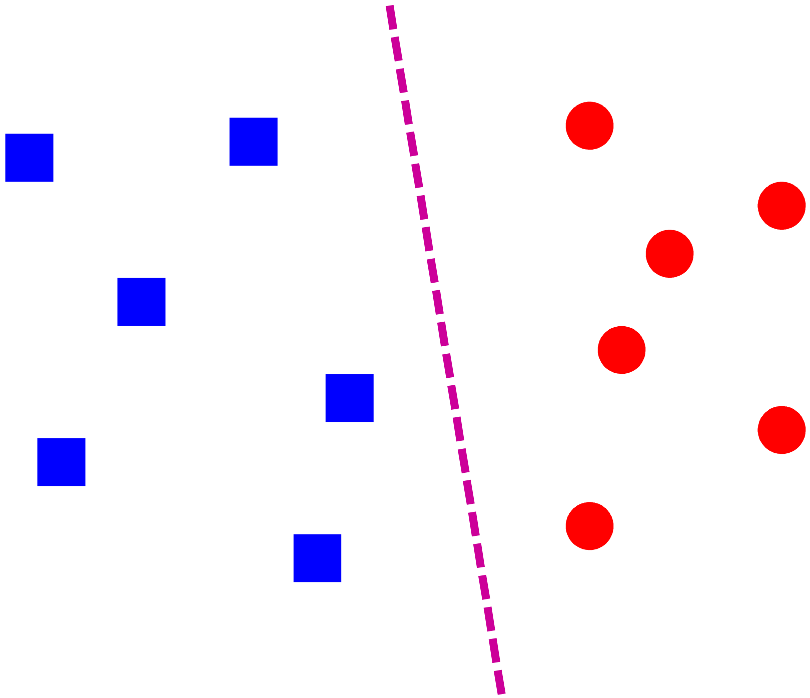
Classifiers: Nearest neighbor



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

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

Classifiers: Linear

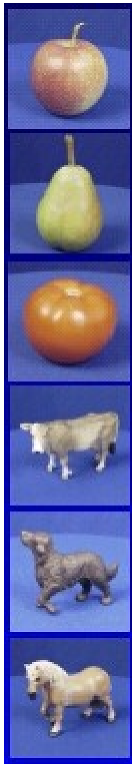


- Find a *linear function* to separate the classes:

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

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?

Diagnosing generalization ability

- **Training error:** how well does the model perform at prediction on the data on which it was trained?
- **Test error:** how well does it perform on a never before seen test set?
- Training and test error are both *high*: **underfitting**
  - Model does an equally poor job on the training and the test set
  - Either the training procedure is ineffective or the model is too "simple" to represent the data
- Training error is *low* but test error is *high*: **overfitting**
  - Model has fit irrelevant characteristics (noise) in the training data
  - Model is too complex or amount of training data is insufficient

Underfitting and overfitting

Underfitting | Good generalization | Overfitting

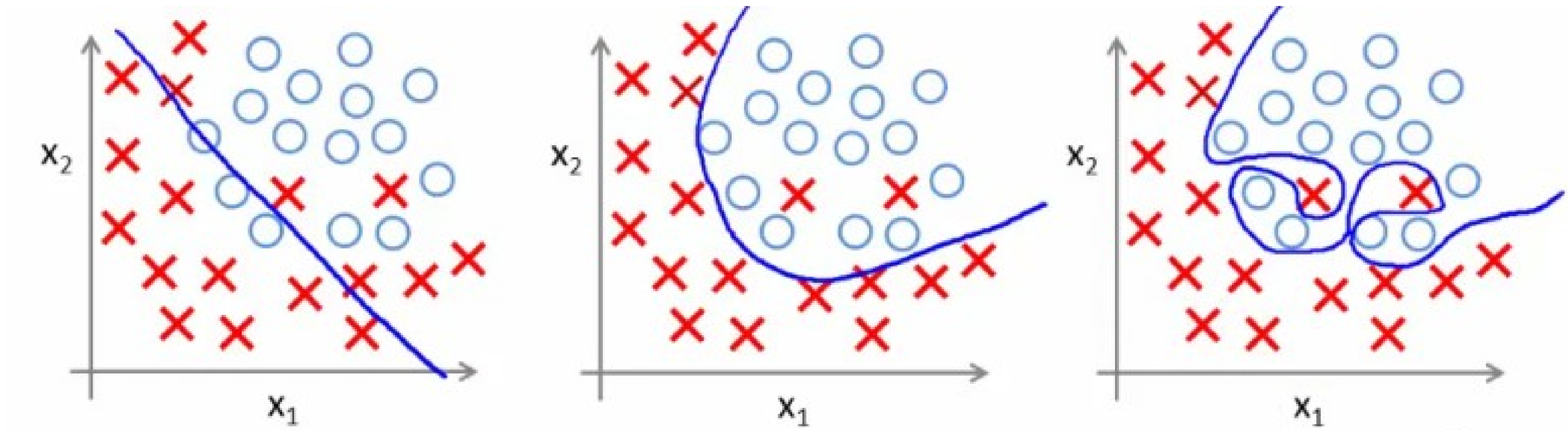
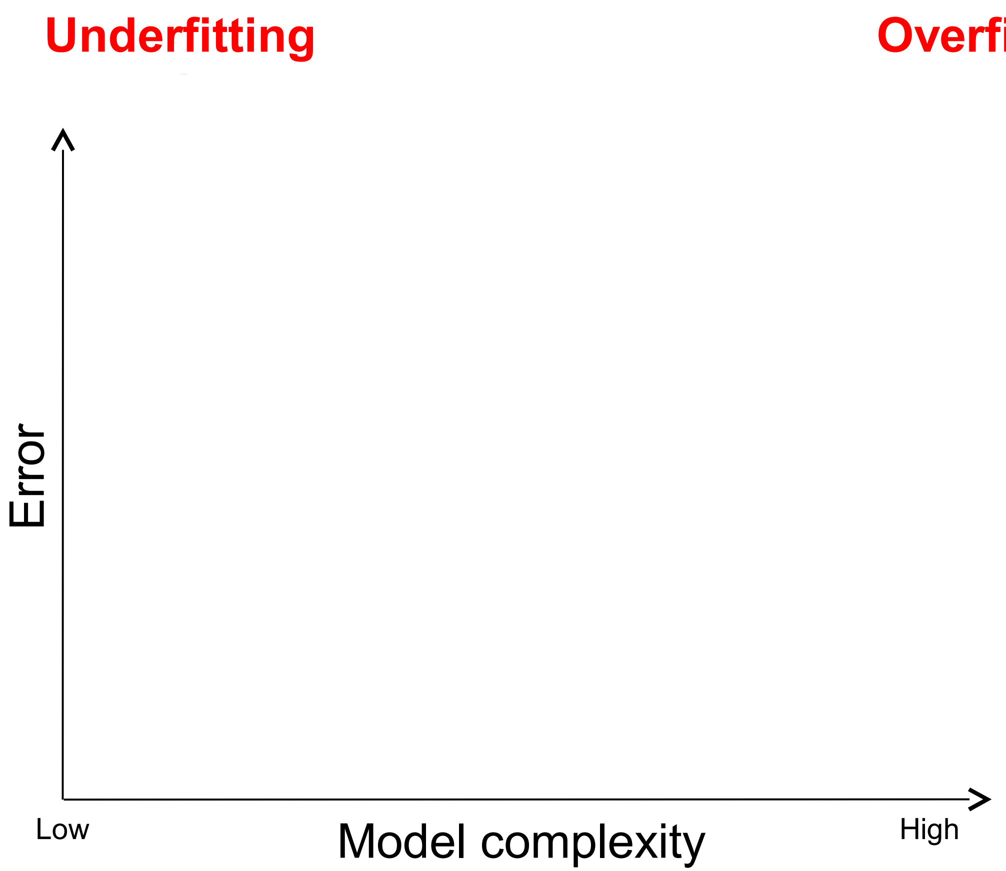


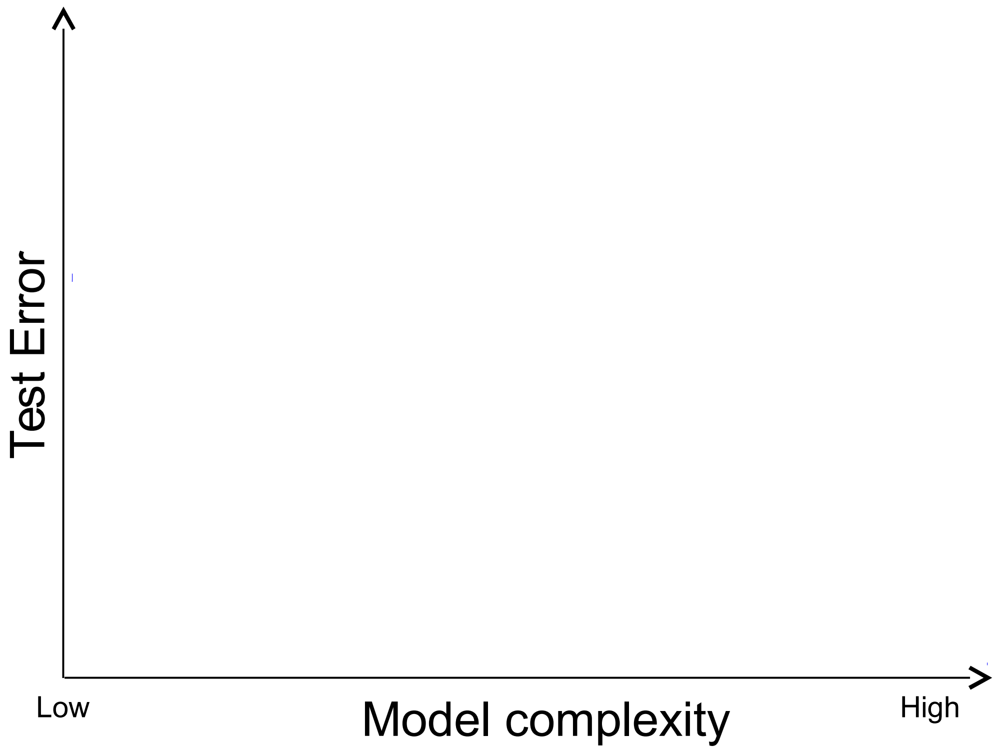
Figure source

Effect of model complexity



Slide credit: D. Hoiem

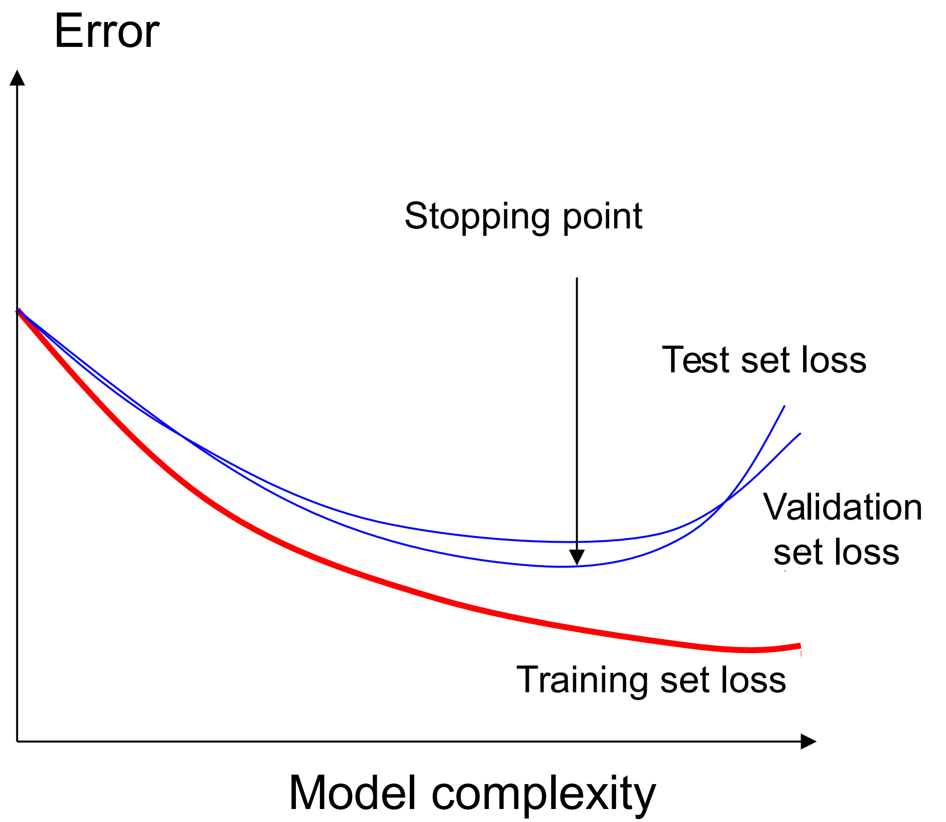
Effect of training set size



Slide credit: D. Hoiem

Validation

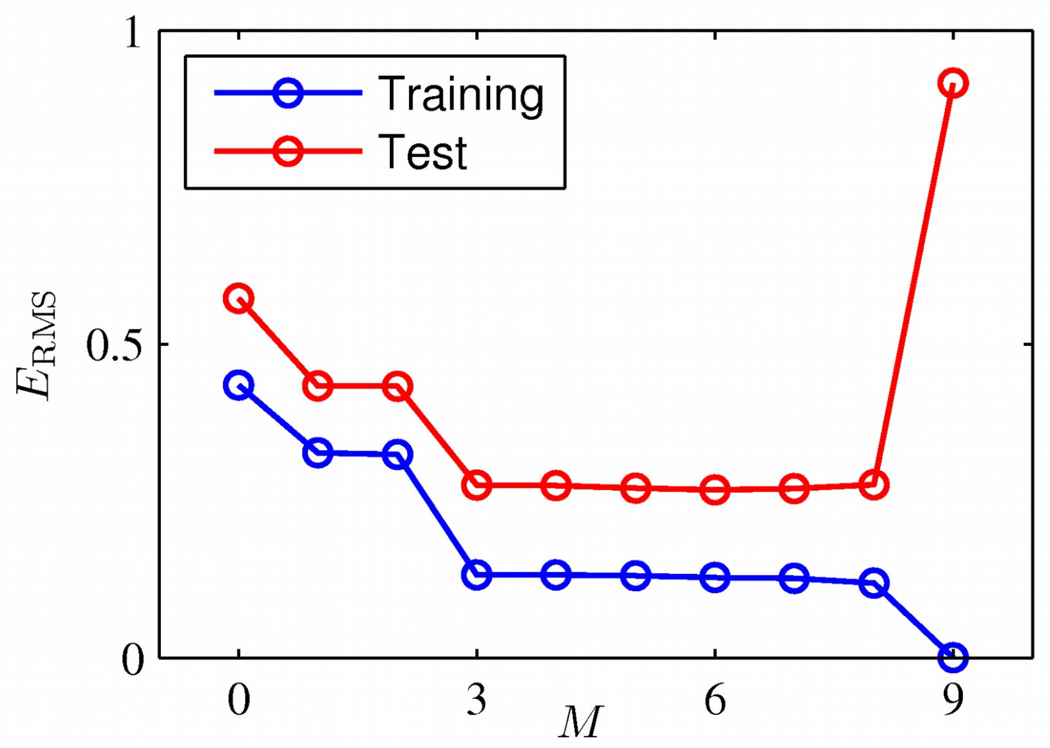
- Split the dataset into **training**, **validation**, and **test** sets
- Use training set to **optimize model parameters**
- Use validation test to **choose the best model**
- Use test set only to **evaluate performance**



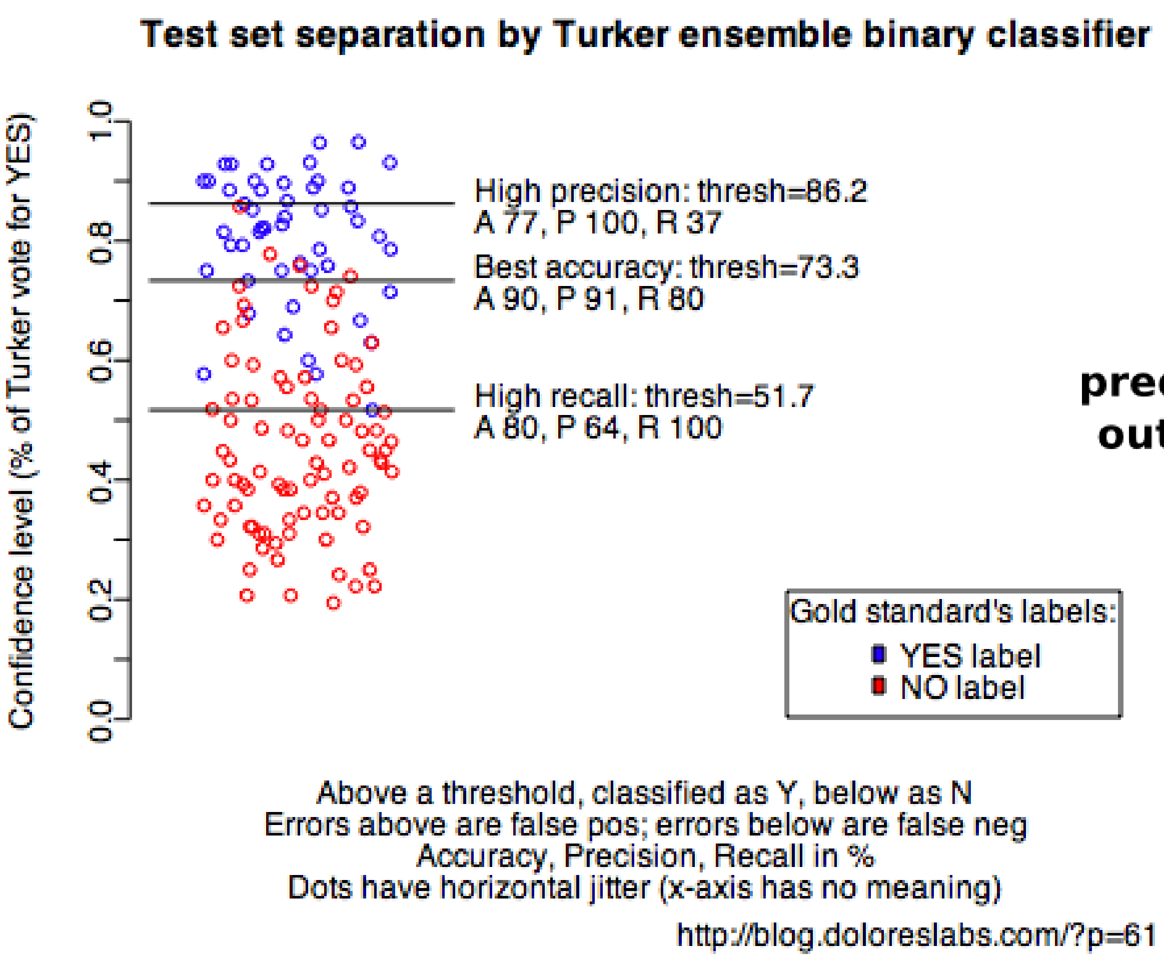
Cross validation

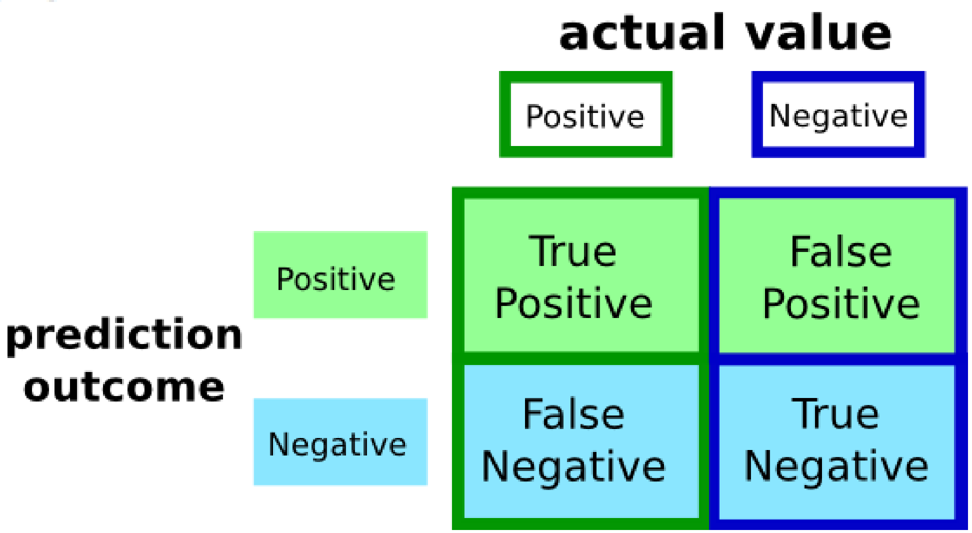
	Downside	Upside
Test-set	Variance: unreliable estimate of future performance	Cheap
Leave-one-out	Expensive. Has some weird behavior	Doesn't waste data
10-fold	Wastes 10% of the data. 10 times more expensive than test set	Only wastes 10%. Only 10 times more expensive instead of R times.
3-fold	Wastier than 10-fold. Expensivier than test set	Slightly better than test-set
R-fold	Identical to Leave-one-out	

Generalization

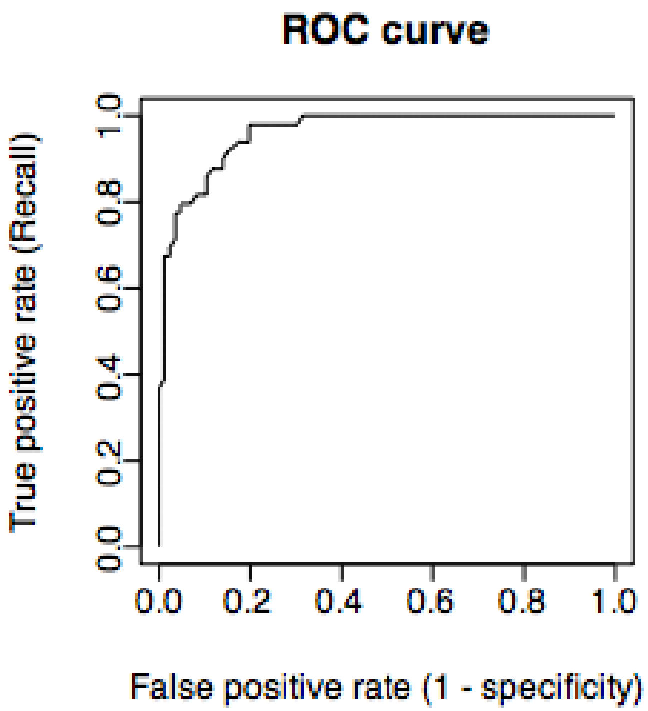


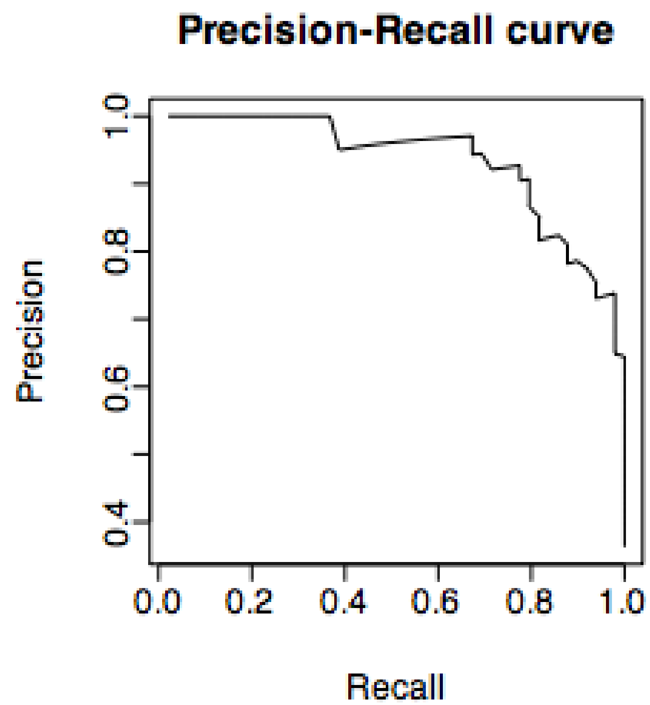
Classifier performance

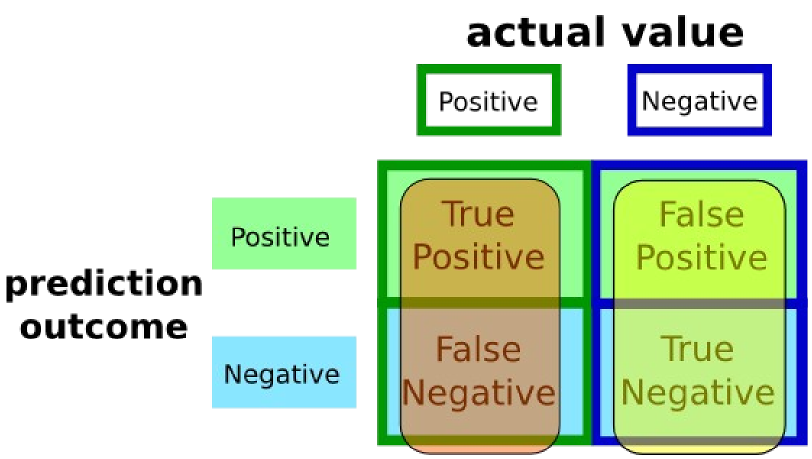


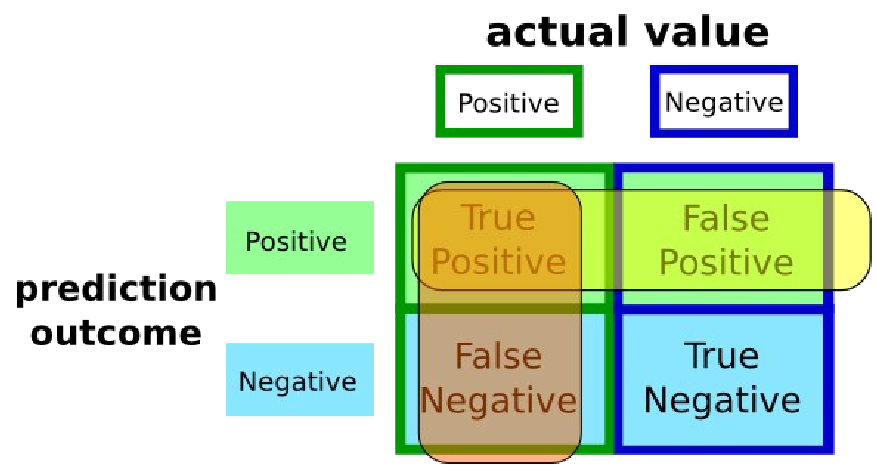


Classifier performance









Object categorization: the statistical viewpoint

- MAP decision:

$$p(zebra \mid image)$$

vs.

$$p(no\ zebra \mid image)$$

Object categorization: the statistical viewpoint

- MAP decision:

$$p(zebra \mid image)$$

vs.

$$p(no\ zebra \mid image)$$

- Bayes rule:

$$\underbrace{p(zebra \mid image)}_{\text{posterior}} \propto \underbrace{p(image \mid zebra)}_{\text{likelihood}} \underbrace{p(zebra)}_{\text{prior}}$$

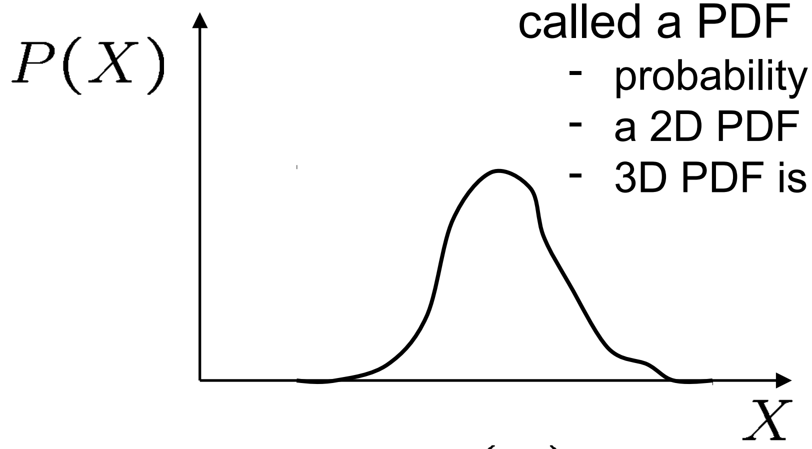
Object categorization: the statistical viewpoint

$$\underbrace{p(zebra \mid image)}_{\text{posterior}} \propto \underbrace{p(image \mid zebra)}_{\text{likelihood}} \underbrace{p(zebra)}_{\text{prior}}$$

- **Discriminative methods:** **model posterior**
- **Generative methods:** **model likelihood and prior**

Probability

– X is a random variable.
– P(X) is the probability that X achieves a certain value.



called a PDF

- probability distribution/density function
- a 2D PDF is a surface
- 3D PDF is a volume

$$0 \leq P(X) \leq 1$$

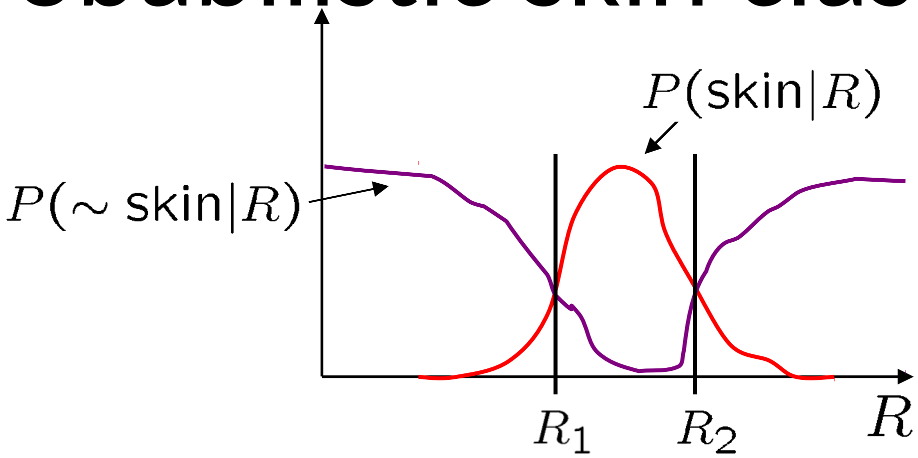
$$\int_{-\infty}^{\infty} P(X)dX = 1$$

continuous X

$$\sum P(X) = 1$$

discrete X

Probabilistic skin classification



- Model PDF / uncertainty
  - Each pixel has a probability of being skin or not skin

$$P(\sim \text{skin}|R) = 1 - P(\text{skin}|R)$$

- Skin classifier
  - Given X = (R,G,B): how to determine if it is skin or not?
  - Choose interpretation of highest probability

- Where do we get $P(\text{skin}|R)$ $P(\sim \text{skin}|R)$

התפלגות גאוסיאנית רב-ממדית

- בהנתן:

(מטריצת שונות משותפת) – Σ

(ממוצע רב-ממדי) – μ

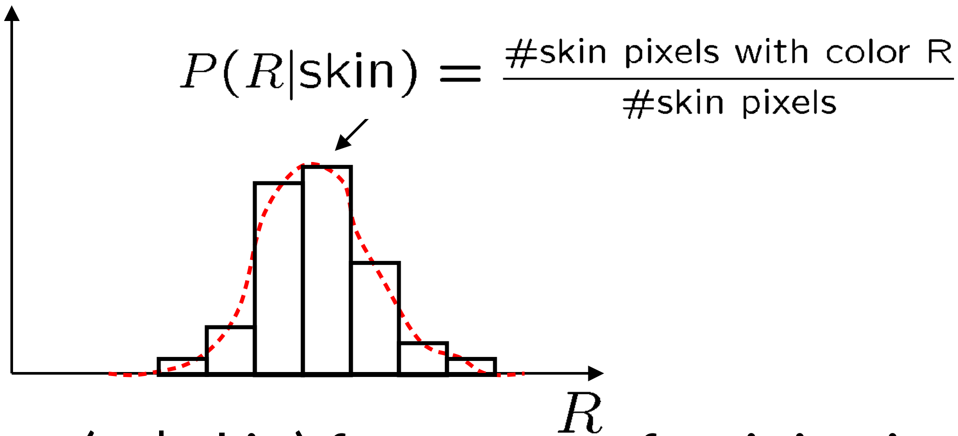
- חישוב הסתברות לנקודה x ממד k:

$$P(x)=\left(\frac{1}{2\pi}\right)^{k/2}\left|\Sigma\right|^{-1/2}\exp\left(-\frac{1}{2}\left(x-\mu\right)^{T}\Sigma^{-1}\left(x-\mu\right)\right)$$

- או ב- MATLAB

```
y = mvnpdf(X,MU,SIGMA);
```

Learning conditional PDF's



- We can calculate P(R | skin) from a set of training images
- But this isn't quite what we want
  - We want P(skin | R) not P(R | skin)
  - How can we get it?

Bayes rule

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

- In terms of our problem:

what we measure
(**likelihood**)

domain knowledge
(**prior**)

$$P(\text{skin}|R) = \frac{P(R|\text{skin})\ P(\text{skin})}{P(R)}$$

what we want
(**posterior**)

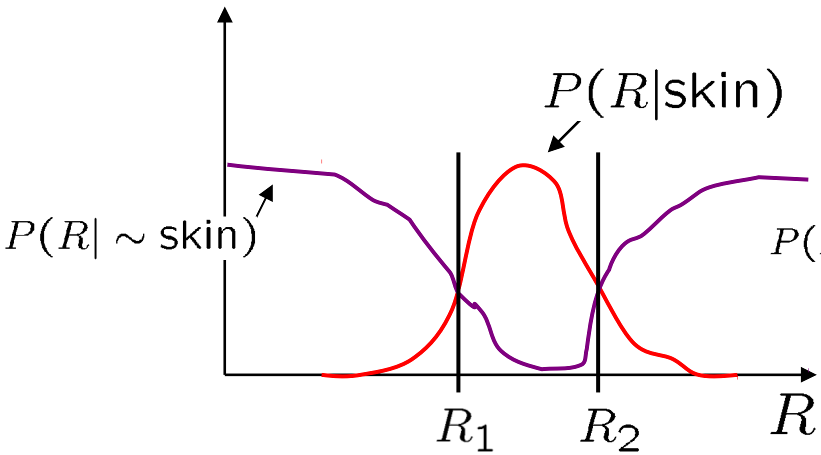
normalization term

$$P(R) = P(R|\text{skin})P(\text{skin}) + P(R|\sim\text{skin})P(\sim\text{skin})$$

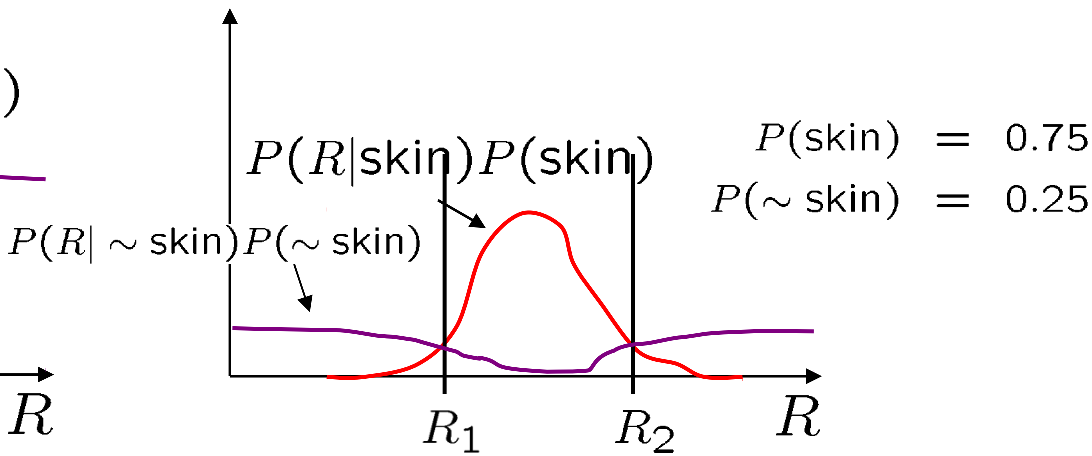
What can we use for the prior P(skin)?

- Domain knowledge:
  - P(skin) may be larger if we know the image contains a person
  - For a portrait, P(skin) may be higher for pixels in the center
- Learn the prior from the training set. How?
  - P(skin) is proportion of skin pixels in training set

Bayesian estimation



likelihood

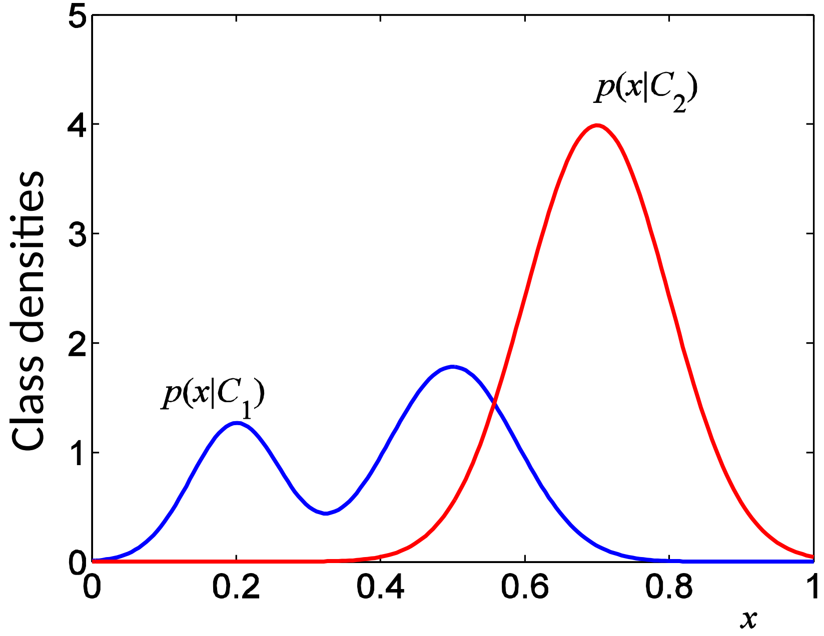


posterior (unnormalized)

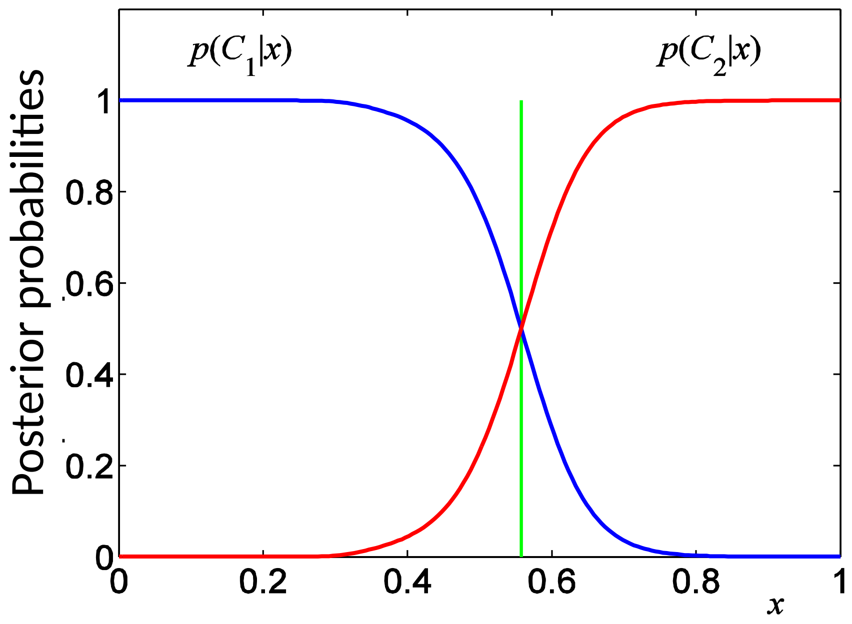
- Bayesian estimation
  - Goal is to choose the label (skin or ~skin) that maximizes the posterior $\leftrightarrow$ minimizes probability of misclassification
    - this is called **Maximum A Posteriori (MAP) estimation**

Generative vs. discriminative learning

Generative



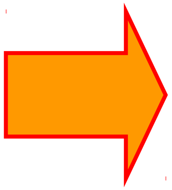
Discriminative



Steps for statistical recognition

- Representation
  - Specify the model for an object category
  - Bag of features, part-based, global, etc.
- Learning
  - Given a *training set*, find the parameters of the model
  - Generative vs. discriminative
- Recognition
  - Apply the model to a new *test image*

Bag-of-features models



Origin: Bag-of-words models

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

US Presidential Speeches Tag Cloud

`http://chir.ag/projects/preztags/`

Origin: Bag-of-words models

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

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

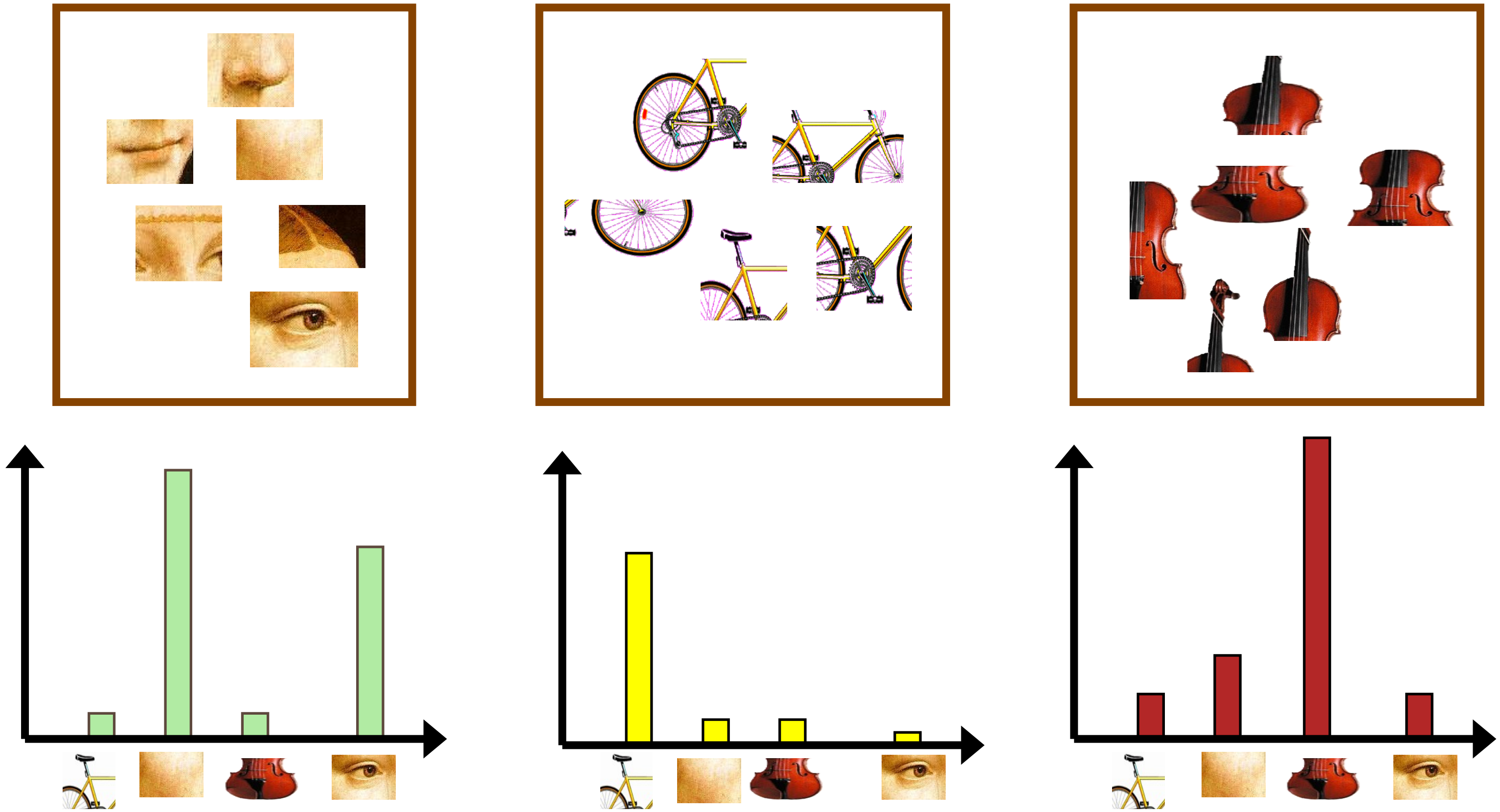
Origin: Bag-of-words models

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

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

Bag-of-features steps

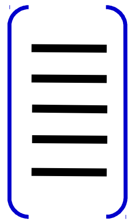
1. Extract features
2. Learn "visual vocabulary"
3. Quantize features using visual vocabulary
4. Represent images by frequencies of "visual words"



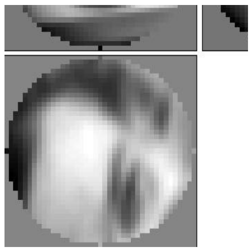
1. Feature extraction

- Regular grid or interest regions

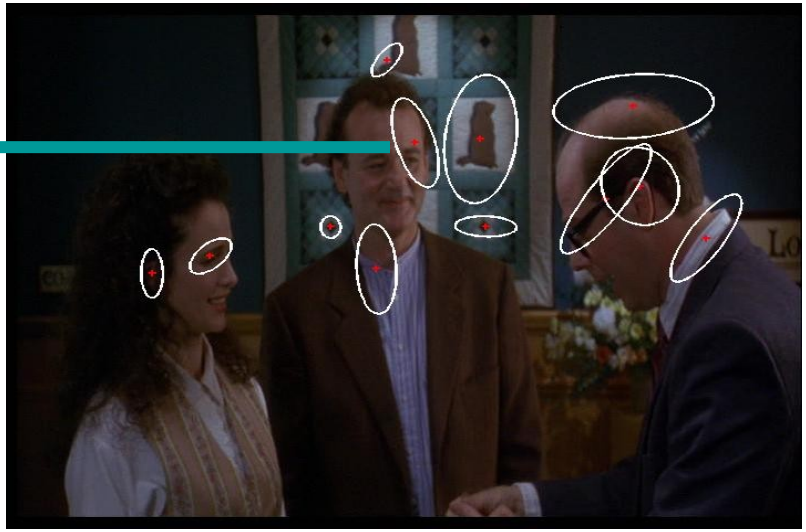
1. Feature extraction



Compute descriptor



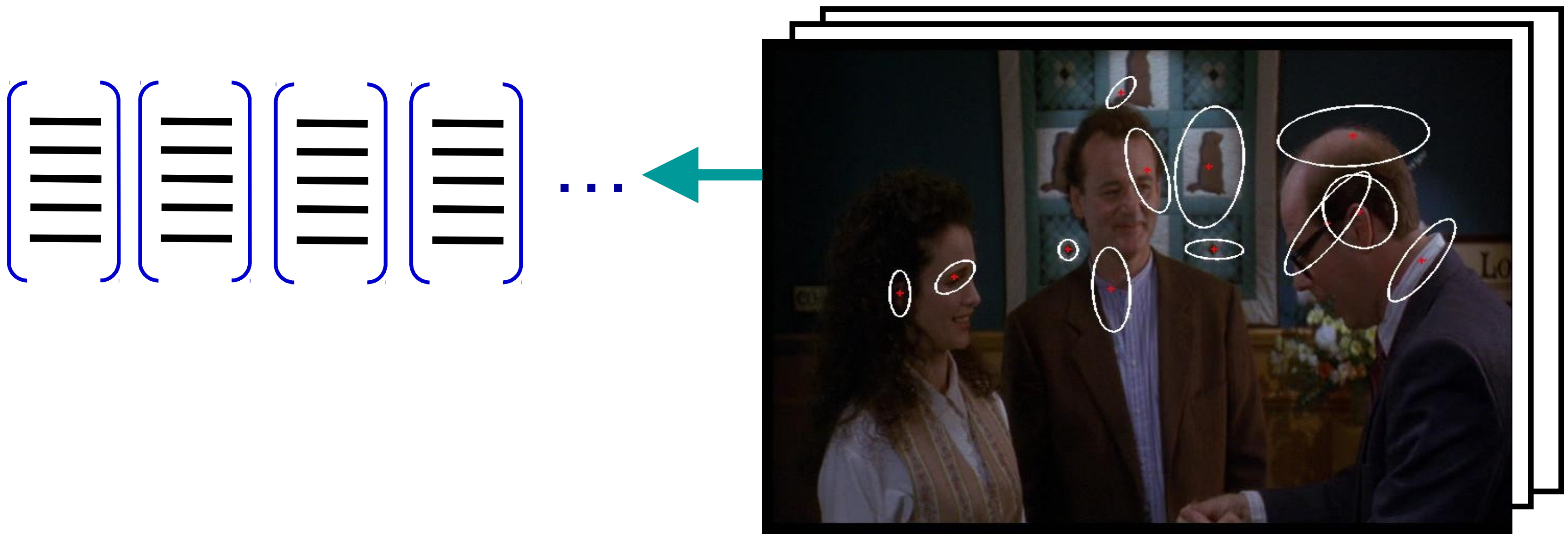
Normalize patch



Detect patches

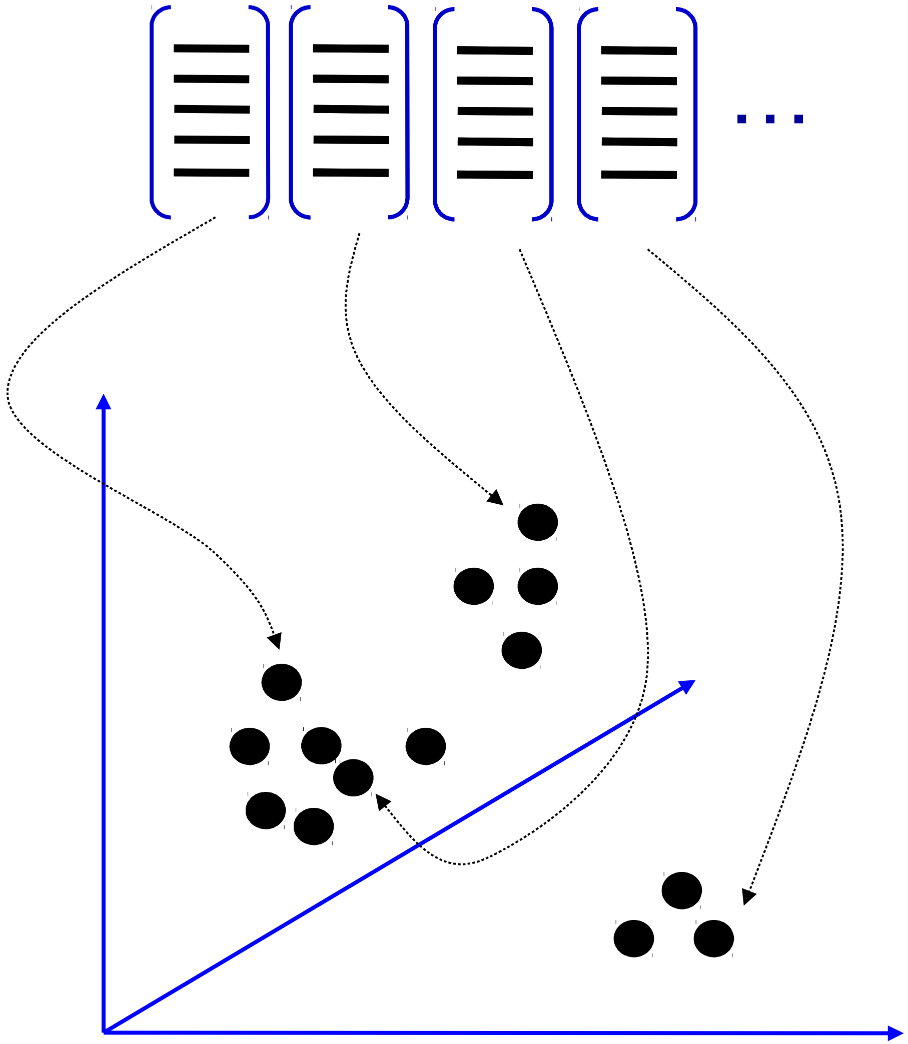
Slide credit: Josef Sivic

1. Feature extraction



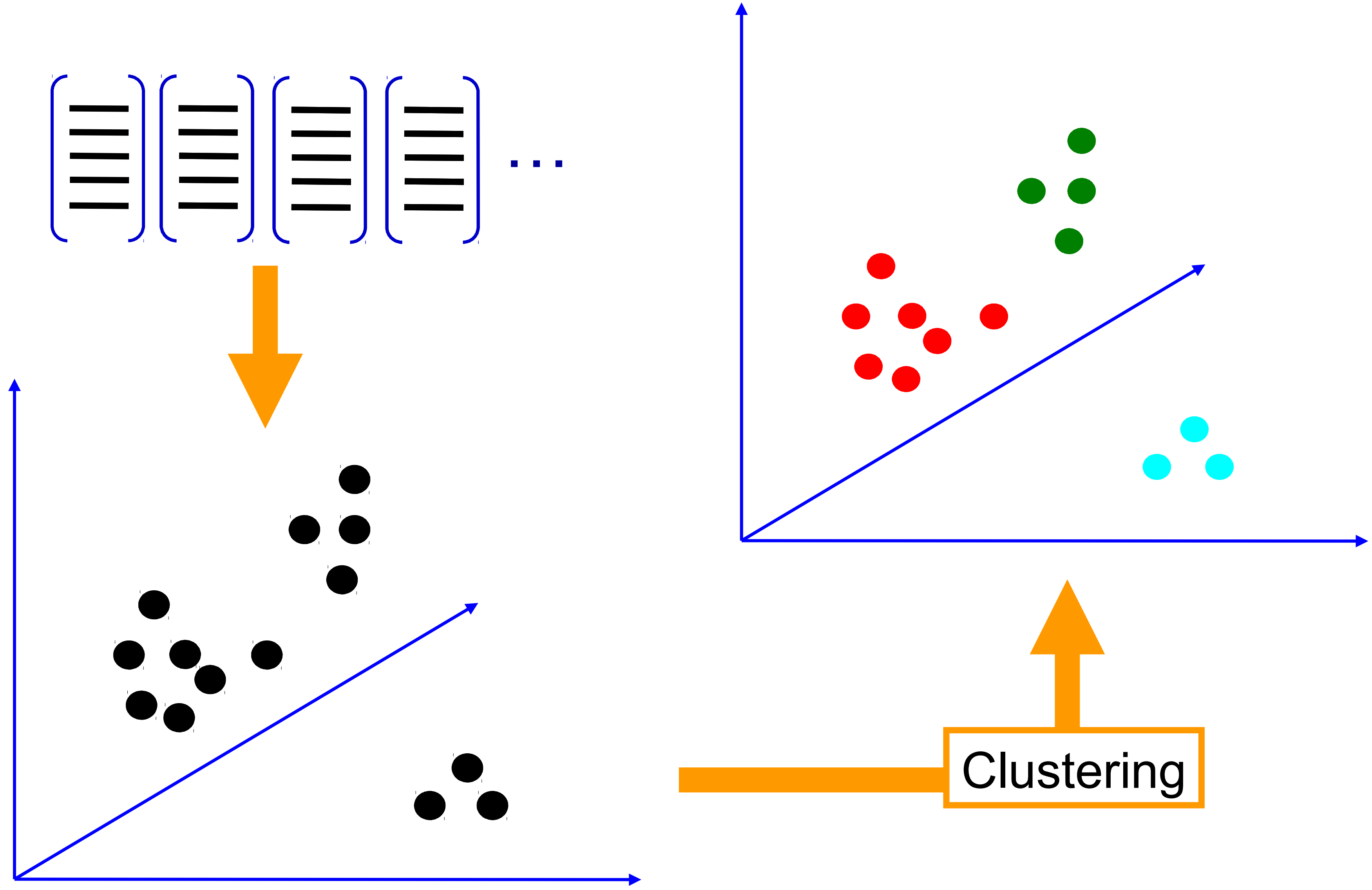
Slide credit: Josef Sivic

2. Learning the visual vocabulary



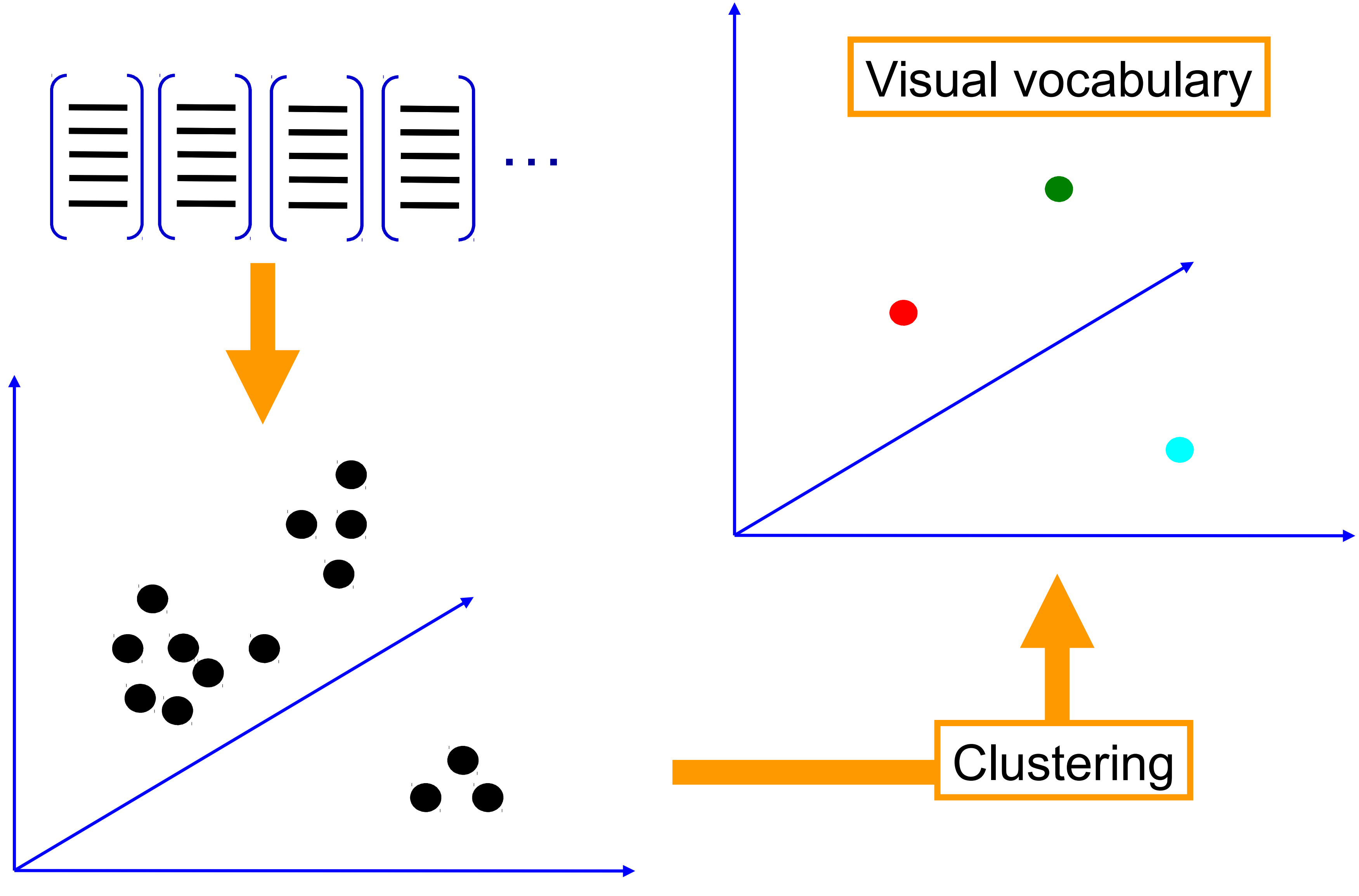
Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

K-means clustering

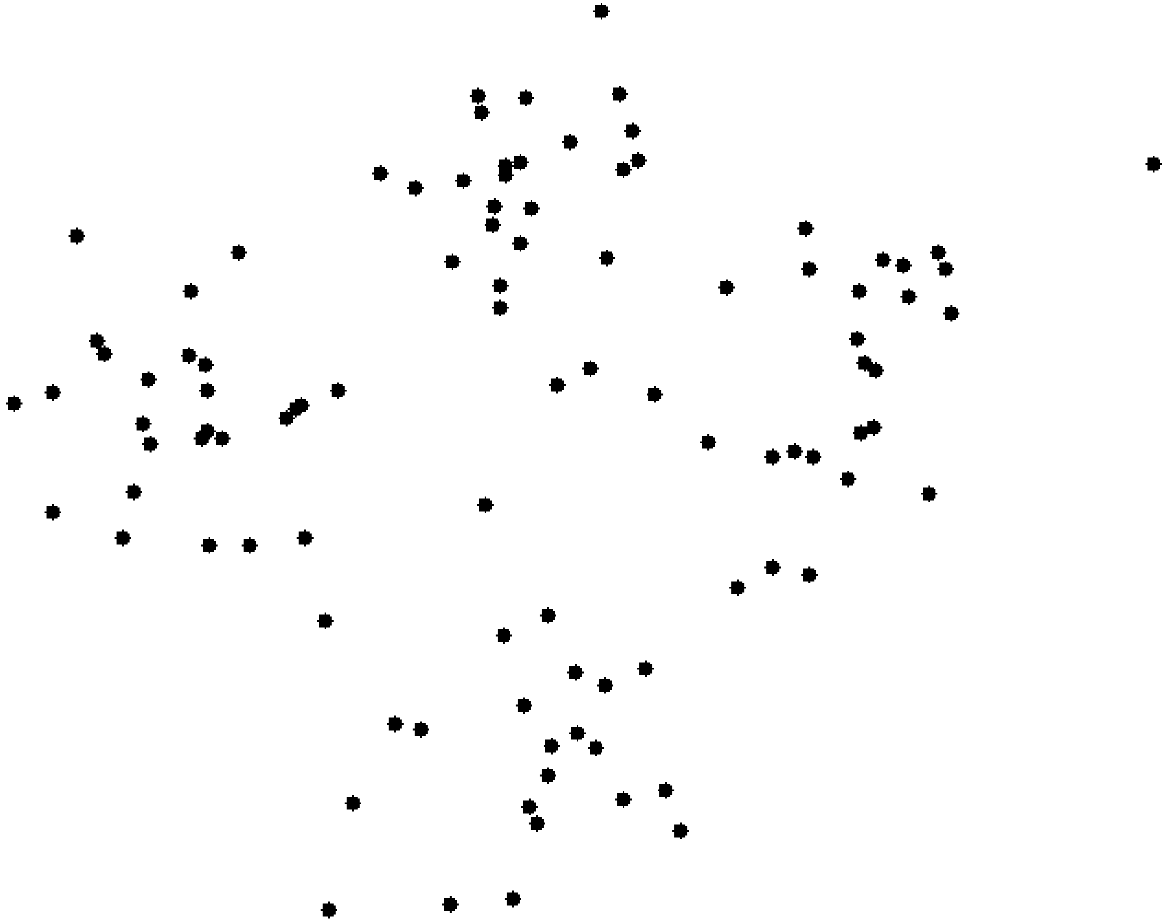
- Want to minimize sum of squared Euclidean distances between points **x**i and their nearest cluster centers **m**k

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

Algorithm:

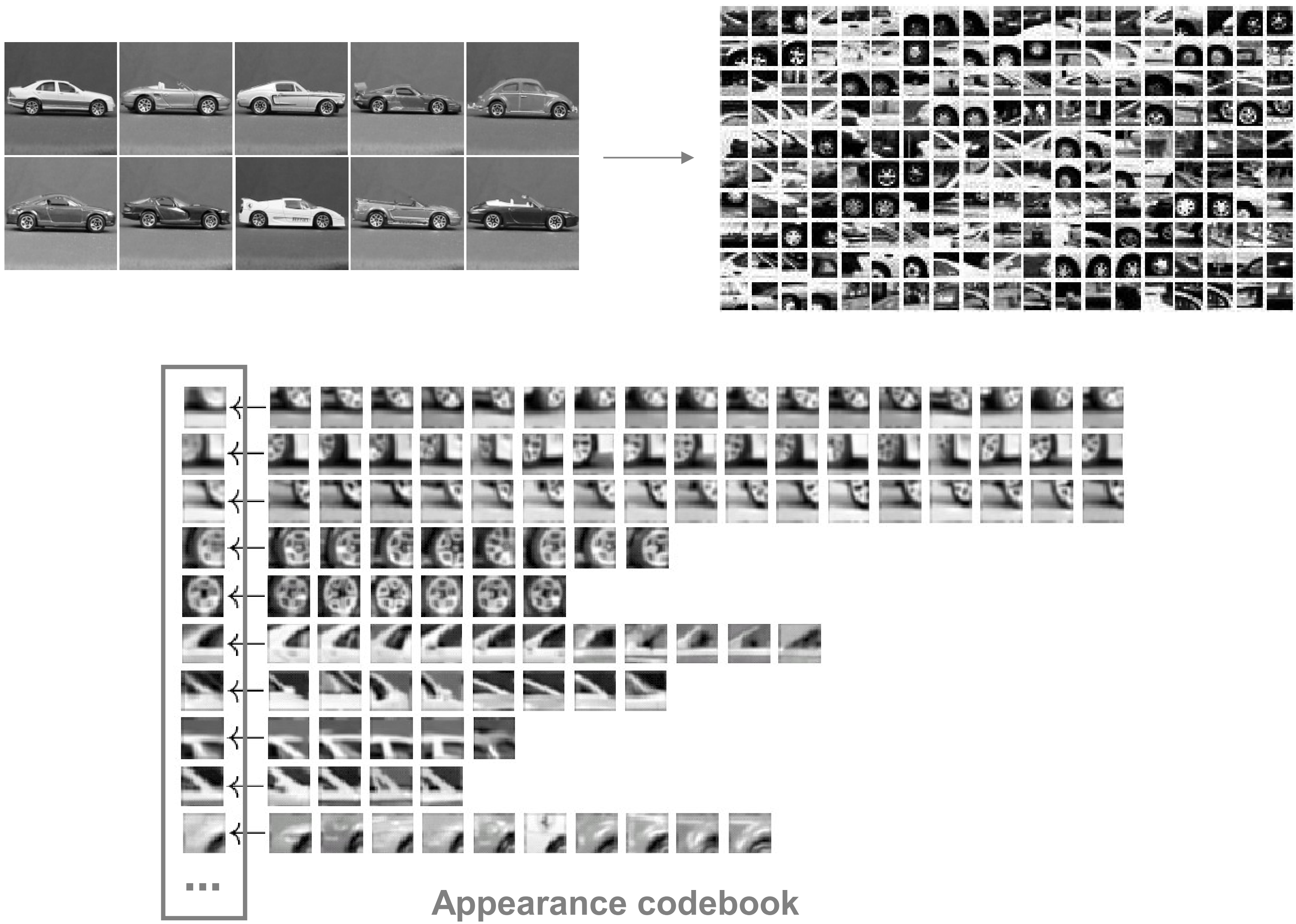
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

K-means demo



Source: http://shabal.in/visuals/kmeans/1.html
Another demo: http://www.kovan.ceng.metu.edu.tr/~maya/kmeans/

Example codebook



Source: B. Leibe

Clustering and vector quantization

- Clustering is a common method for learning a visual vocabulary or codebook
  - Unsupervised learning process
  - Each cluster center produced by k-means becomes a codevector
  - Codebook can be learned on separate training set
  - Provided the training set is sufficiently representative, the codebook will be “universal”

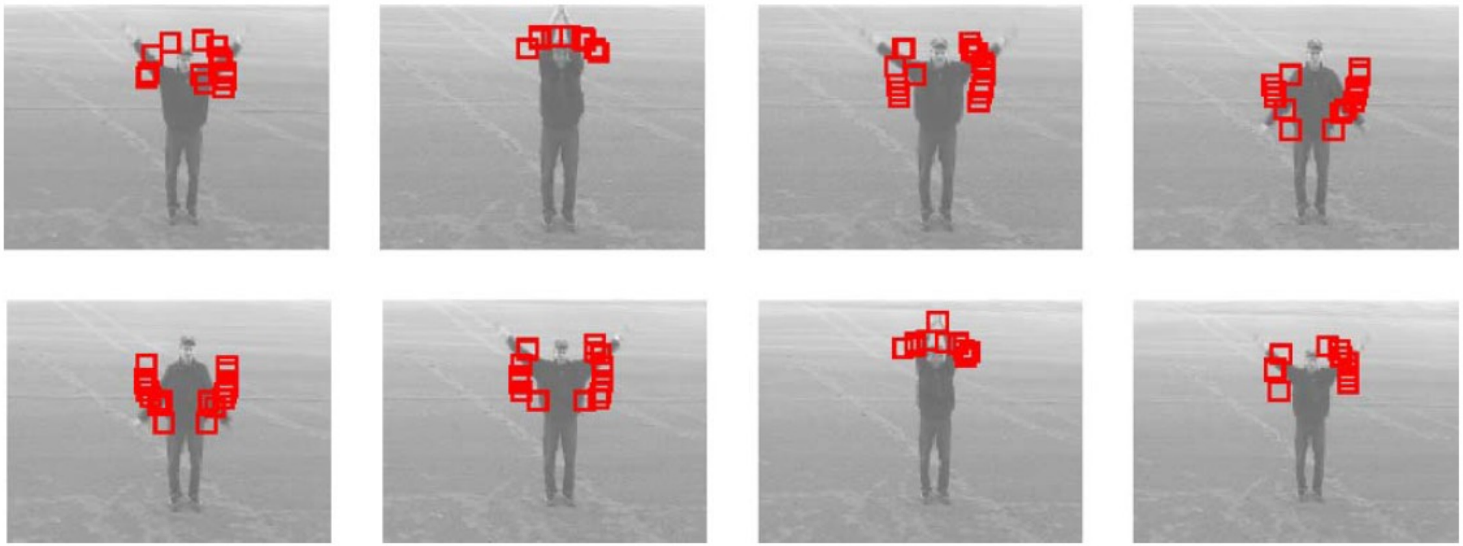
- The codebook is used for quantizing features
  - A *vector quantizer* takes a feature vector and maps it to the index of the nearest codevector in a codebook
  - Codebook = visual vocabulary
  - Codevector = visual word

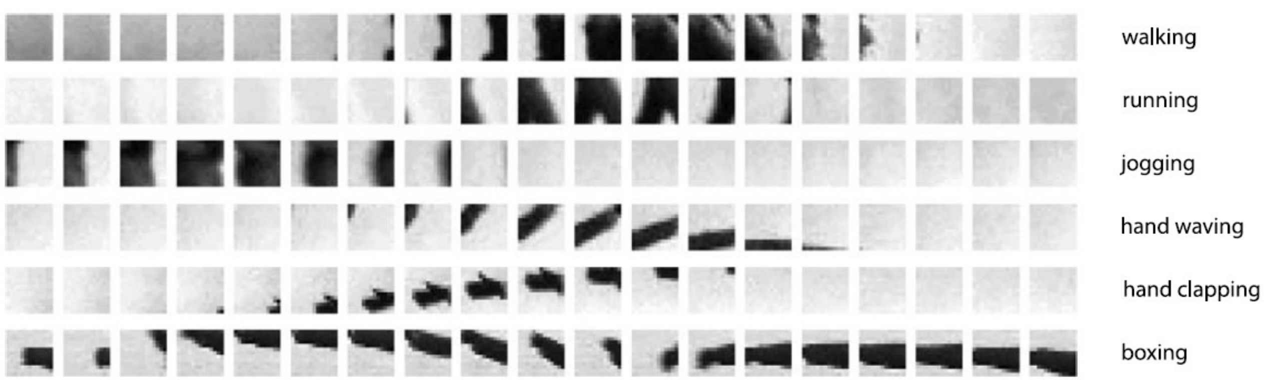
Visual vocabularies: Issues

- How to choose vocabulary size?
  - Too small: visual words not representative of all patches
  - Too large: quantization artifacts, overfitting
  - Right size is application-dependent
- Improving efficiency of quantization
  - Vocabulary trees (Nister and Stewenius, 2005)
- Improving vocabulary quality
  - Discriminative/supervised training of codebooks
  - Sparse coding
- More informative bag-of-words representations
  - Fisher Vectors (Perronnin et al., 2007), VLAD (Jegou et al., 2010)
- Incorporating spatial information

Bags of features for action recognition

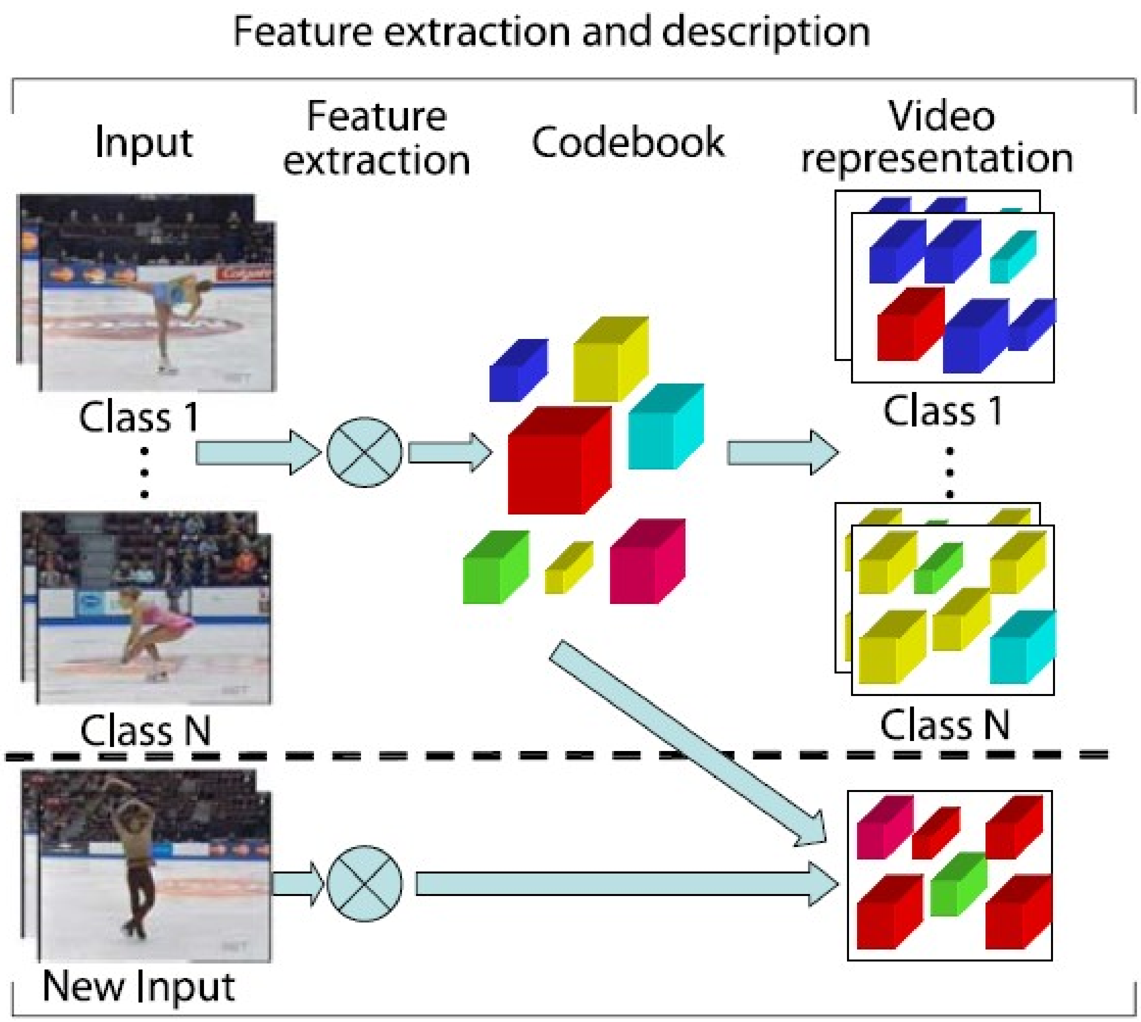
Space-time interest points





Juan Carlos Niebles, Hongcheng Wang and Li Fei-Fei,
Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words,
IJCV 2008.

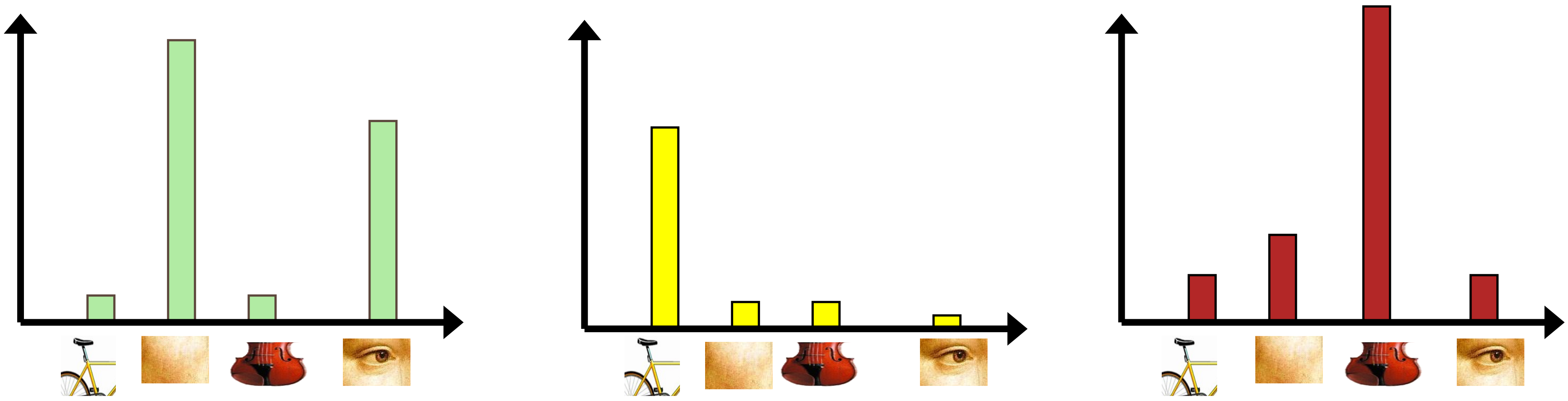
Bags of features for action recognition



Juan Carlos Niebles, Hongcheng Wang and Li Fei-Fei,
Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words,
IJCV 2008.

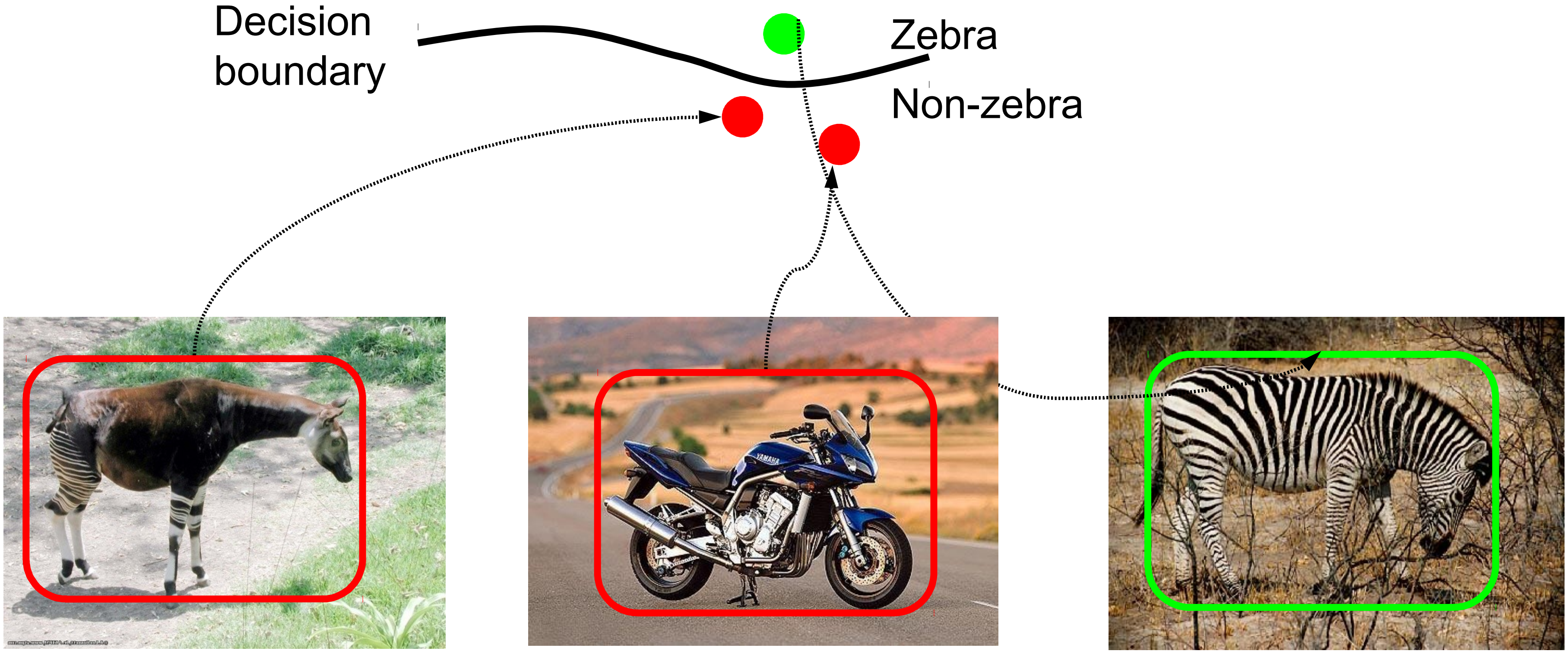
3. classification

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



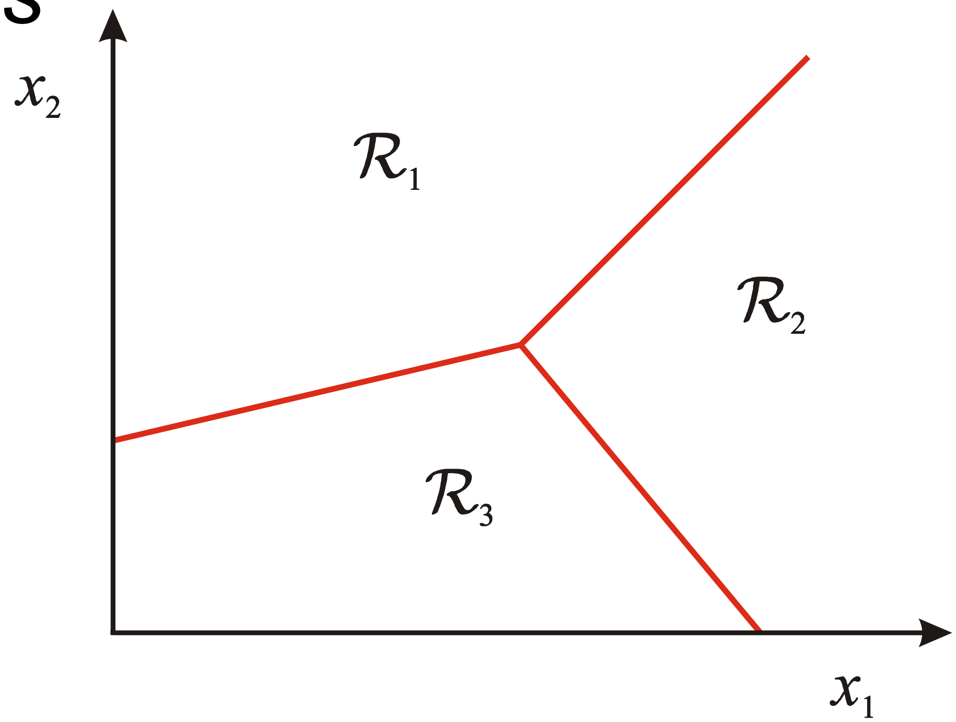
Classifiers

- Learn a decision rule assigning bag-of-features representations of images to different classes



Classification

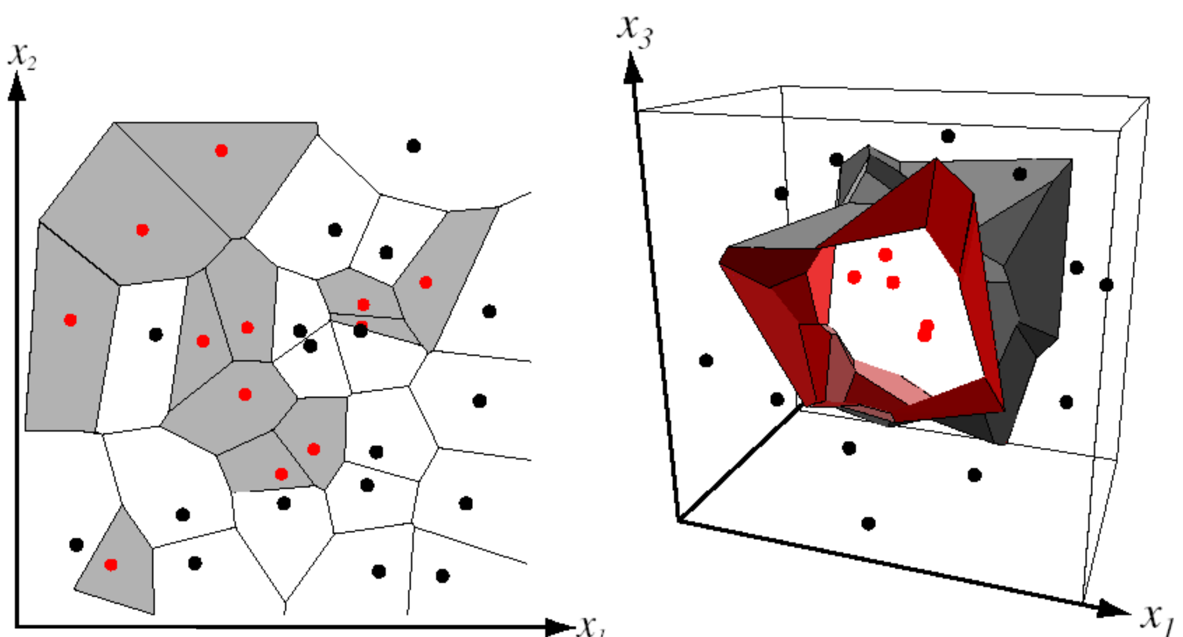
- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



Nearest Neighbor Classifier

Nearest Neighbor Classifier

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



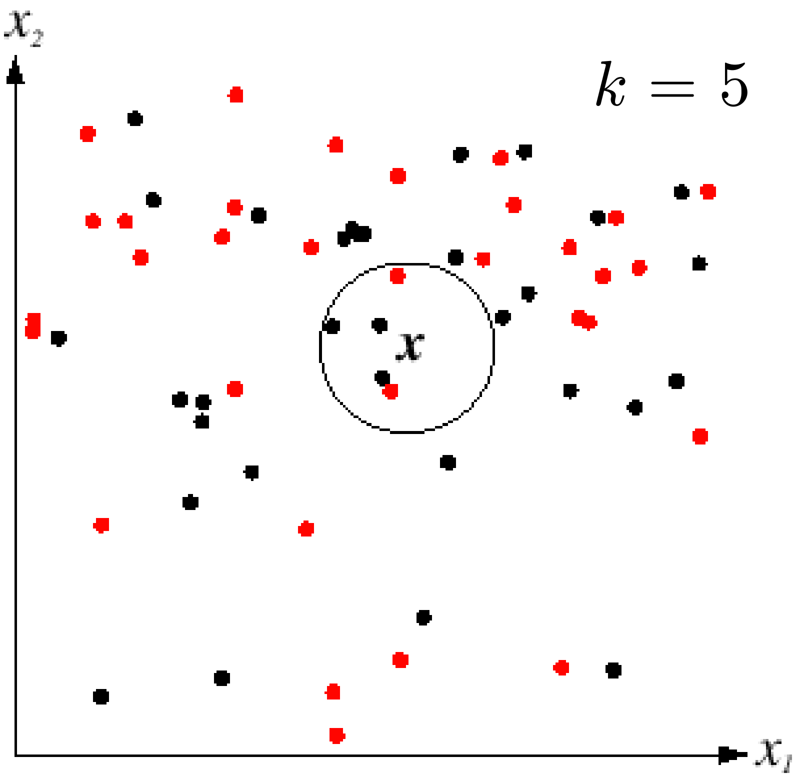
from Duda *et al.*

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

Source: D. Lowe

K-Nearest Neighbors

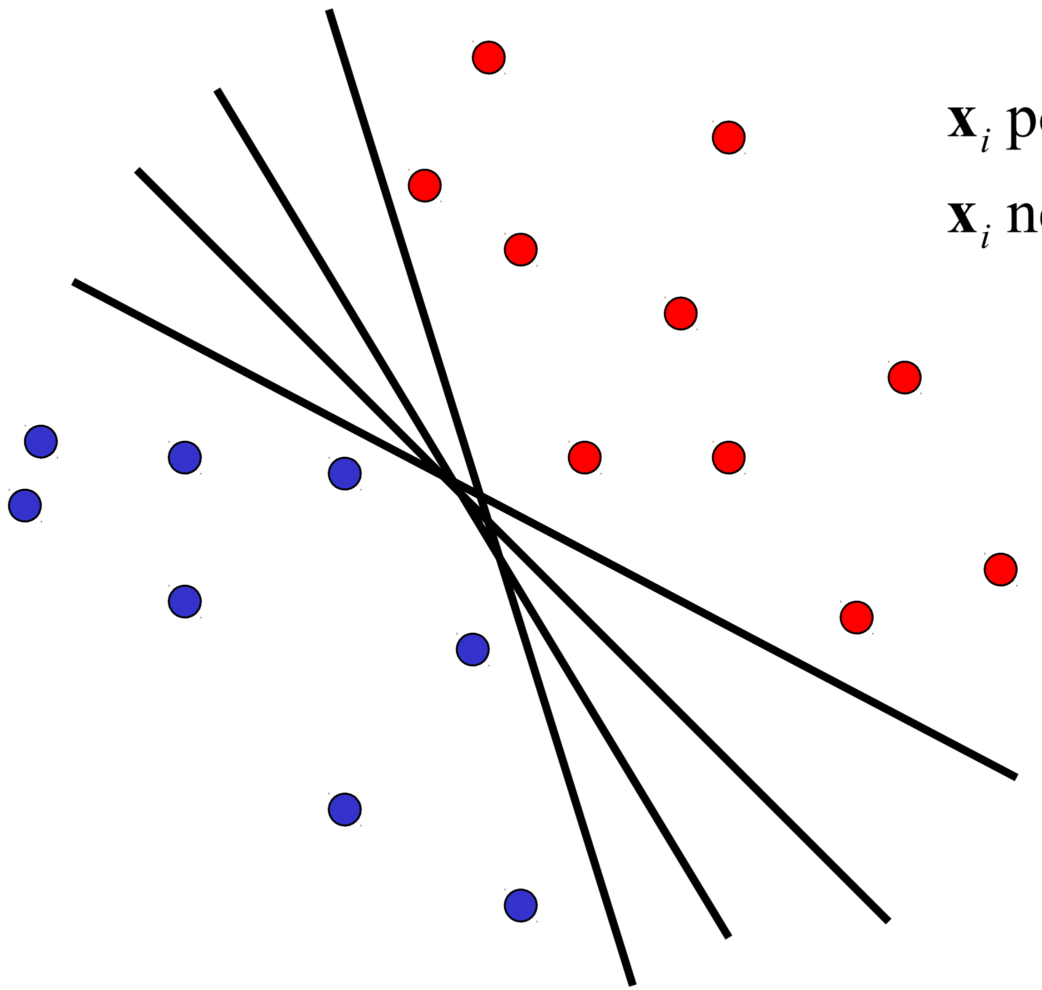
- For a new point, find the k closest points from training data
- Labels of the k points "vote" to classify
- Works well provided there is lots of data and the distance function is good



Source: D. Lowe

Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

Which hyperplane
is best?

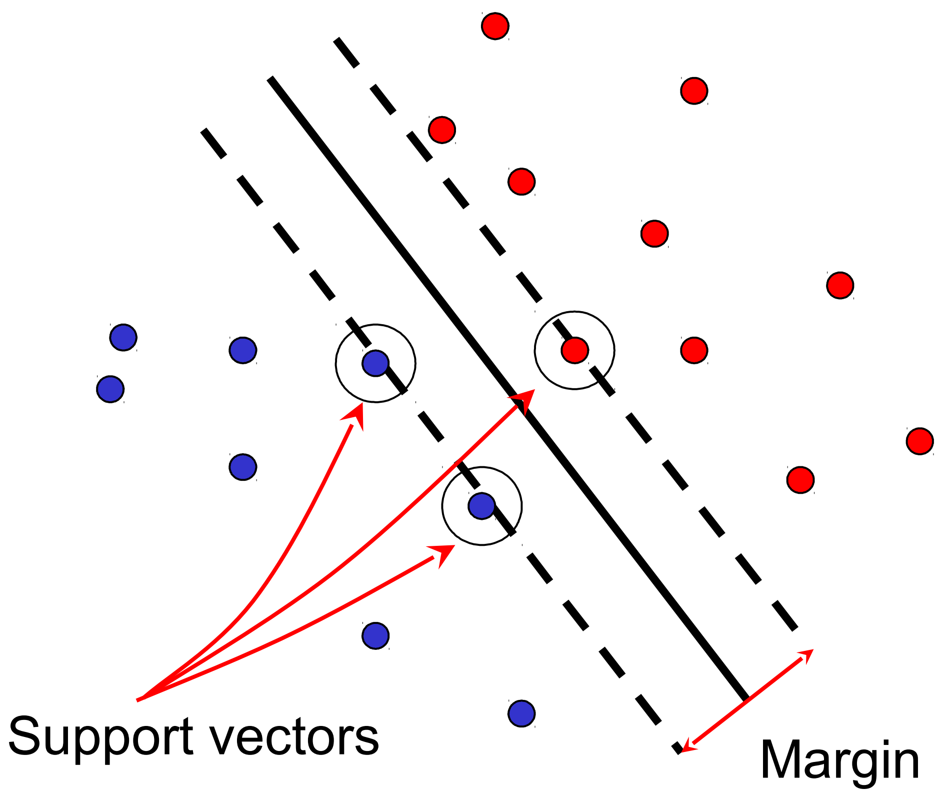
Support vector machines

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

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

Support vector machines

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



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

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

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

Distance between point and hyperplane: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

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

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

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

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
  - Unbalanced classes may be problematic

SVM: Practical Advice

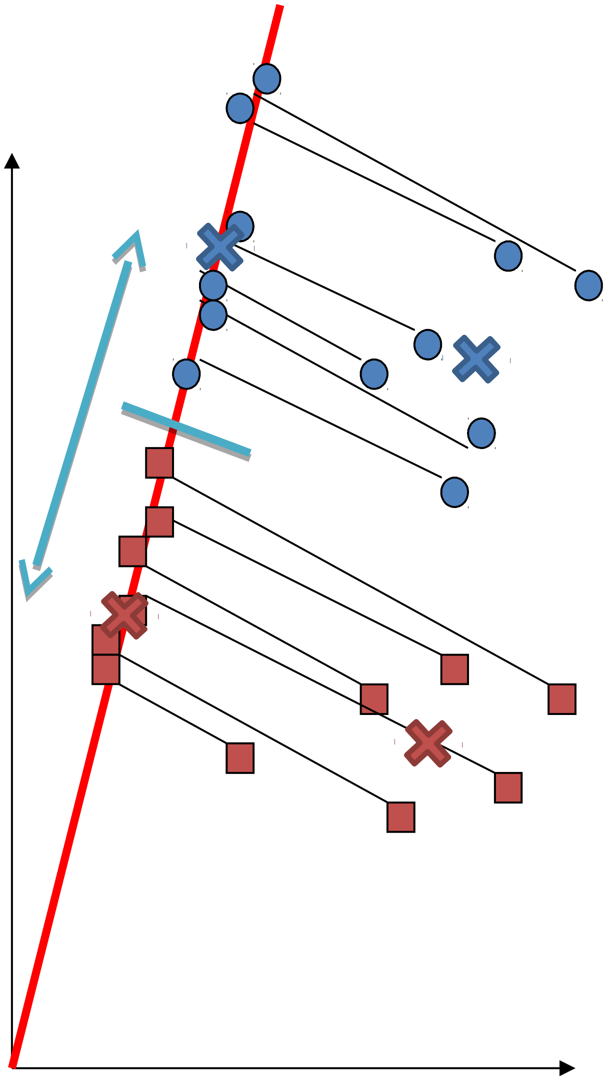
- Avoid MATLAB's implementation
- Use
  http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Or
  http://www.csie.ntu.edu.tw/~cjlin/liblinear/
  when you know you want linear SVM
- Play with C
- Avoid unbalanced classes

LDA

Linear Discriminant Analysis (LDA)

- הרעיון:

1. שימוש ב- LDA למציאת הטלה לממד נמוך
2. בחירת גבול הסיווג על ידי בחירה של היפר מישור מפריד (במקרים רבים, נק. האמצע בין הטלות מרכזי הקבוצות)



LDAs: Pros and cons

- Pros
  - Easy to implement
  - Efficient
  - Works well with unbalanced classes
  - Is often all that’s needed!

- Cons
  - No “direct” multi-class LDA, must combine two-class LDAs
  - Linear separator
  - Makes assumptions on the distributions (Gaussians with similar cov.)…but may work well even when violated

Knn classification revisited

Boiman, Shechtman, Irani; “In Defense of Nearest-Neighbor Based Image Classification”, CVPR’08

Non-parametric classifiers

- The most common non-parametric method is kNN
- Base their classification decision directly on the data, and require (almost) **no learning/training of parameters**.

Advantages over Learning-Based Approaches

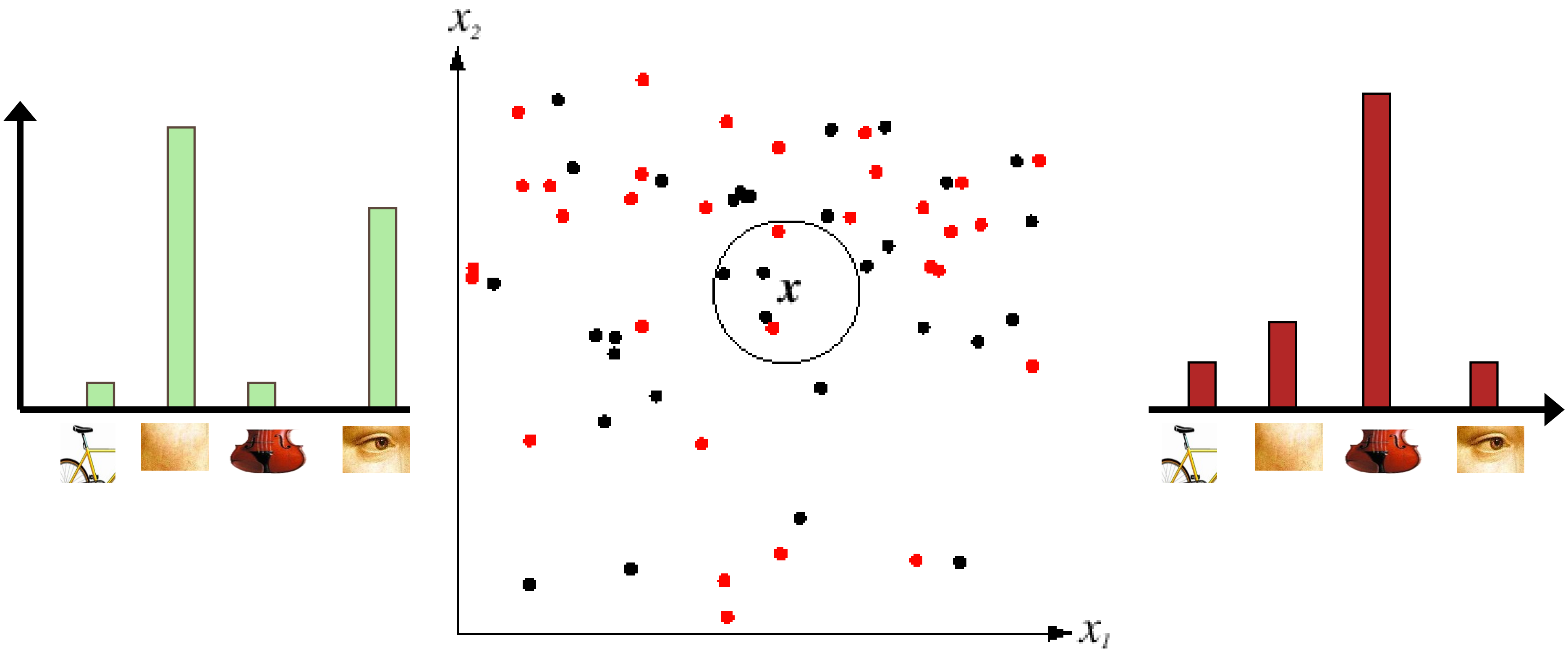
- Naturally handle multiple classes
- No parameters, and so no overfitting of parameters
- Require no training / learning phase

BUT

- Inferior performance compared to learning based approaches (LDA, SVM, AdaBoost, …)

Why?

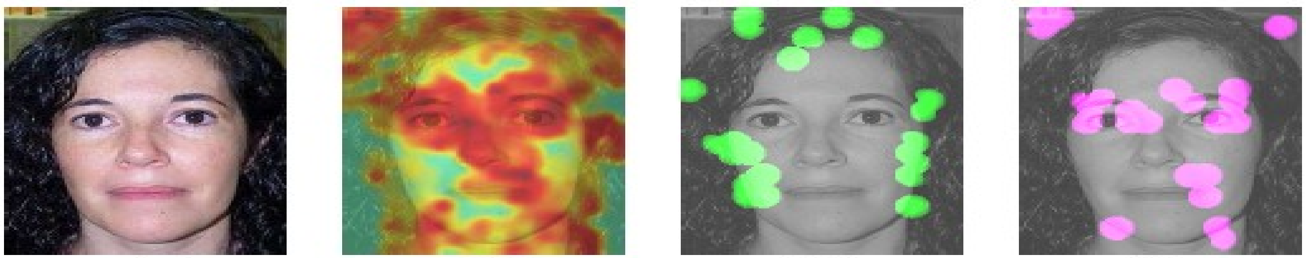
- Reason 1: Descriptor quantization
- Reason 2: image-to-image distance



Reason 1: Intuition

- Highly frequent descriptors have low quantization error, while rare descriptors have high quantization error.
- The most frequent descriptors in a large database of images (e.g., Caltech-101) comprise of simple edges and corners that appear abundantly in all the classes within the database, and therefore are least informative for classification.
- In contrast, the most informative descriptors for classification are the ones found in one (or few) class, but are rare in other classes.

Reason 1: Example



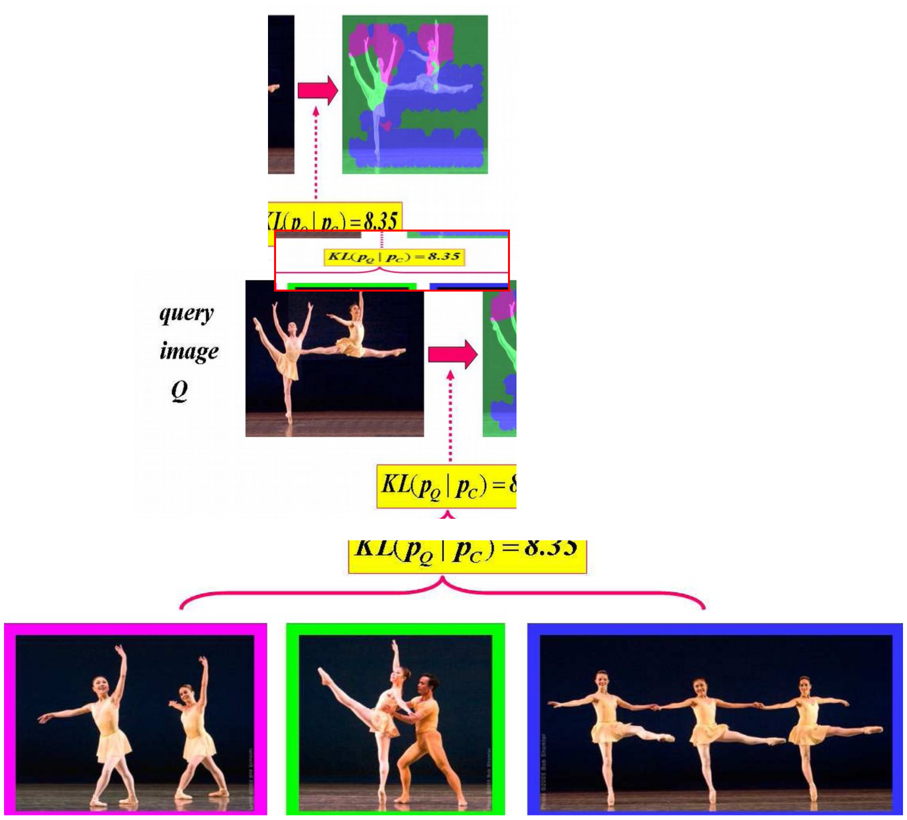
(a) (b) (c) (d)

Figure 1. **Effects of descriptor quantization – Informative descriptors have low database frequency, leading to high quan-**

Reason 2: Image-2-Image

- NN-image classifiers provide good image classification when the query image is similar to one of the labeled images in its class.
- Few example images from a class with large variability □ bad classification!

Reason 2: Example



Conclusion

- Data analysis
- PCA
- Classification problem
- Performance evaluation