COMPUTATIONAL VISION: Object recognition by bag-of-words

Master in Artificial Intelligence

Department of Mathematics and Computer Science

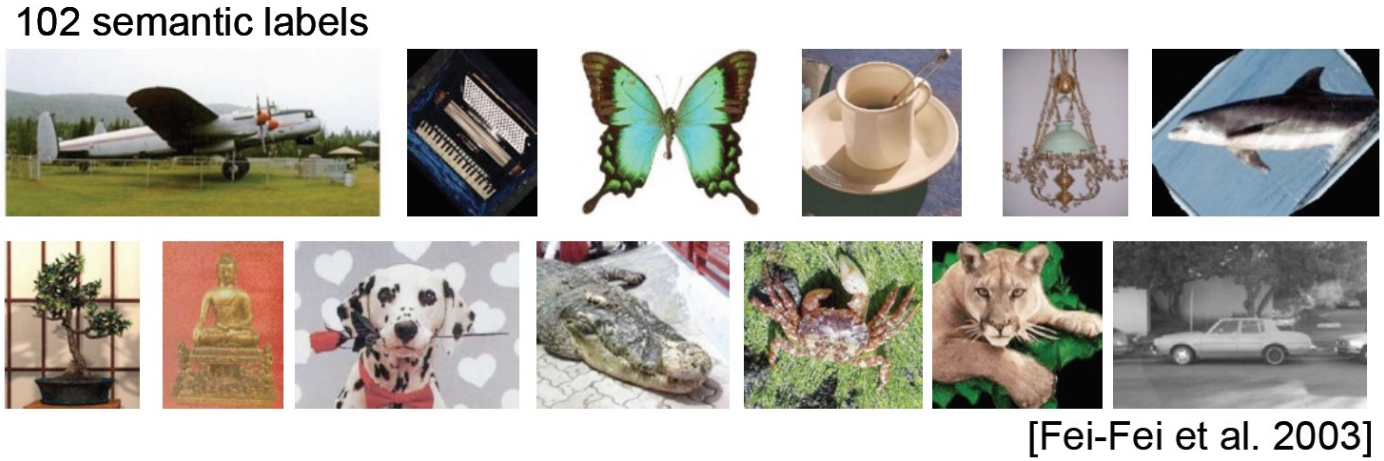
2023-2024

UNIVERSITAT DE BARCELONA

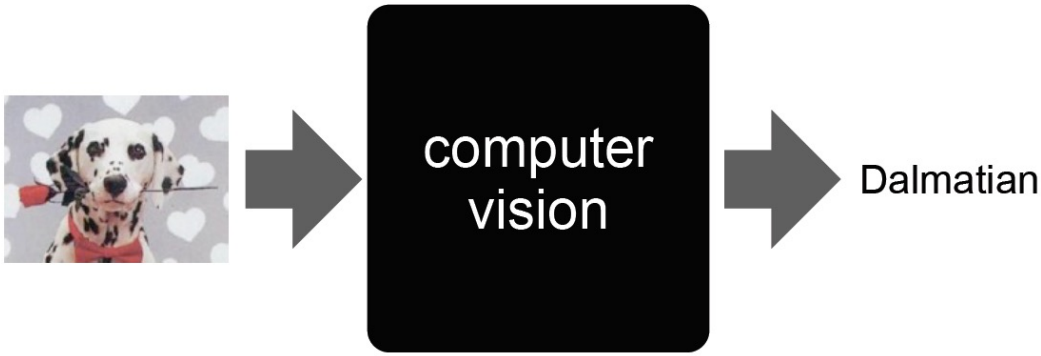
Overview: Bag-of-features models

- Motivation
- Bag of Words
  - Image representation
    - Feature extraction
    - Visual vocabularies
- Discriminative methods for classification
  - Support Vector Machines (SVMs)

Object recognition in Caltech-101

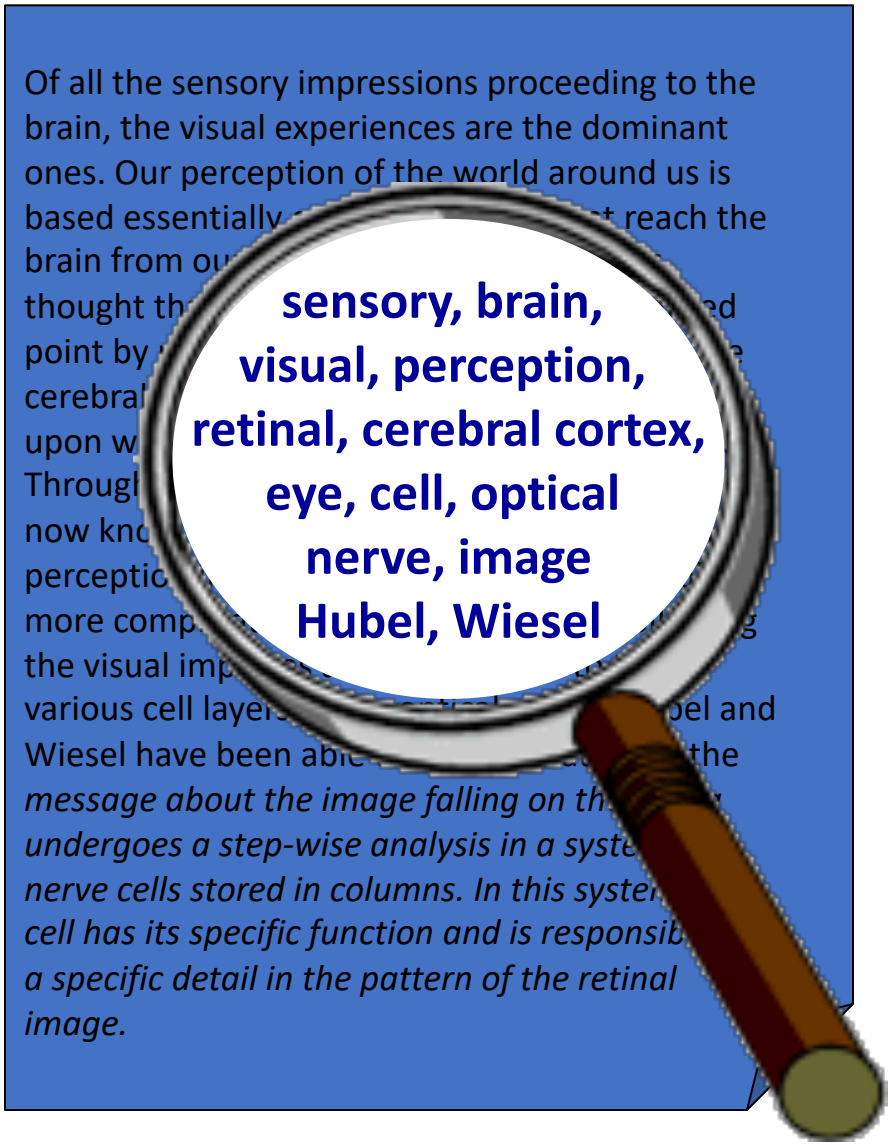


[Fei-Fei et al. 2003]



About 40 to 800 images per category, 9K in total. Most categories have about 50 images. Images are of variable sizes, with typical edge lengths of 200-300 pixels.

Documents classification



Origin 1: Bag-of-words models

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

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

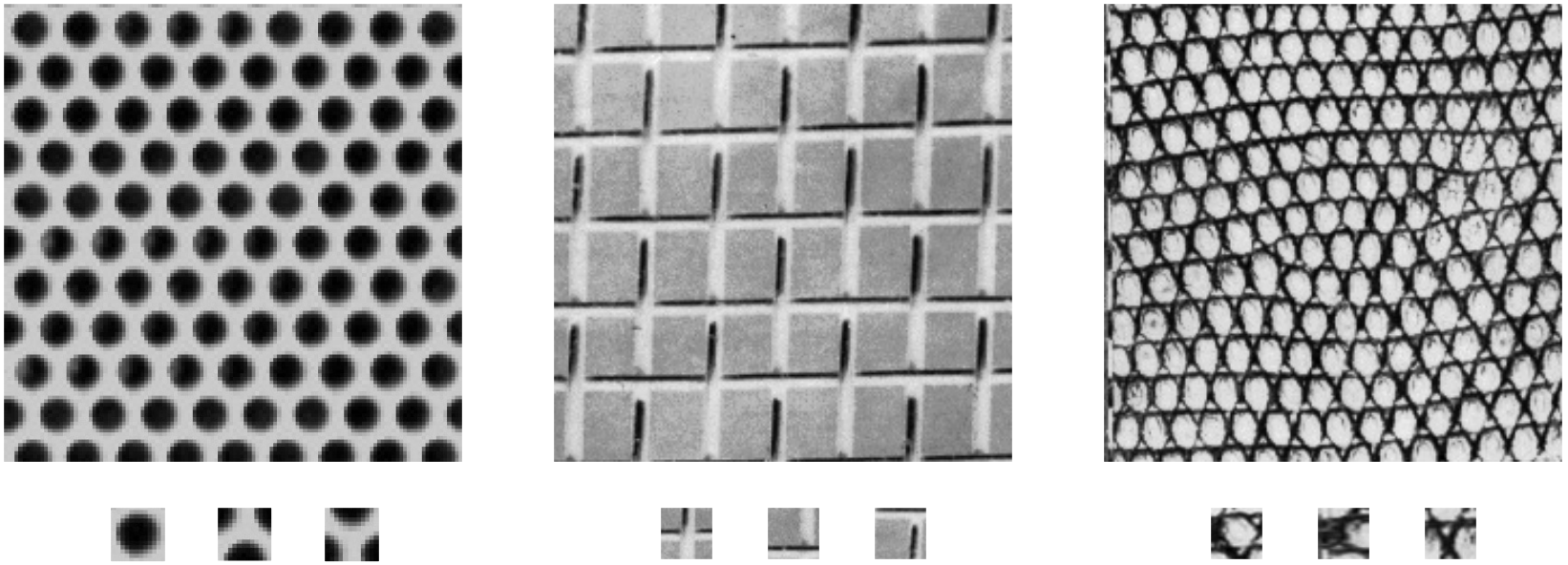
Origin 1: Bag-of-words models

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

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

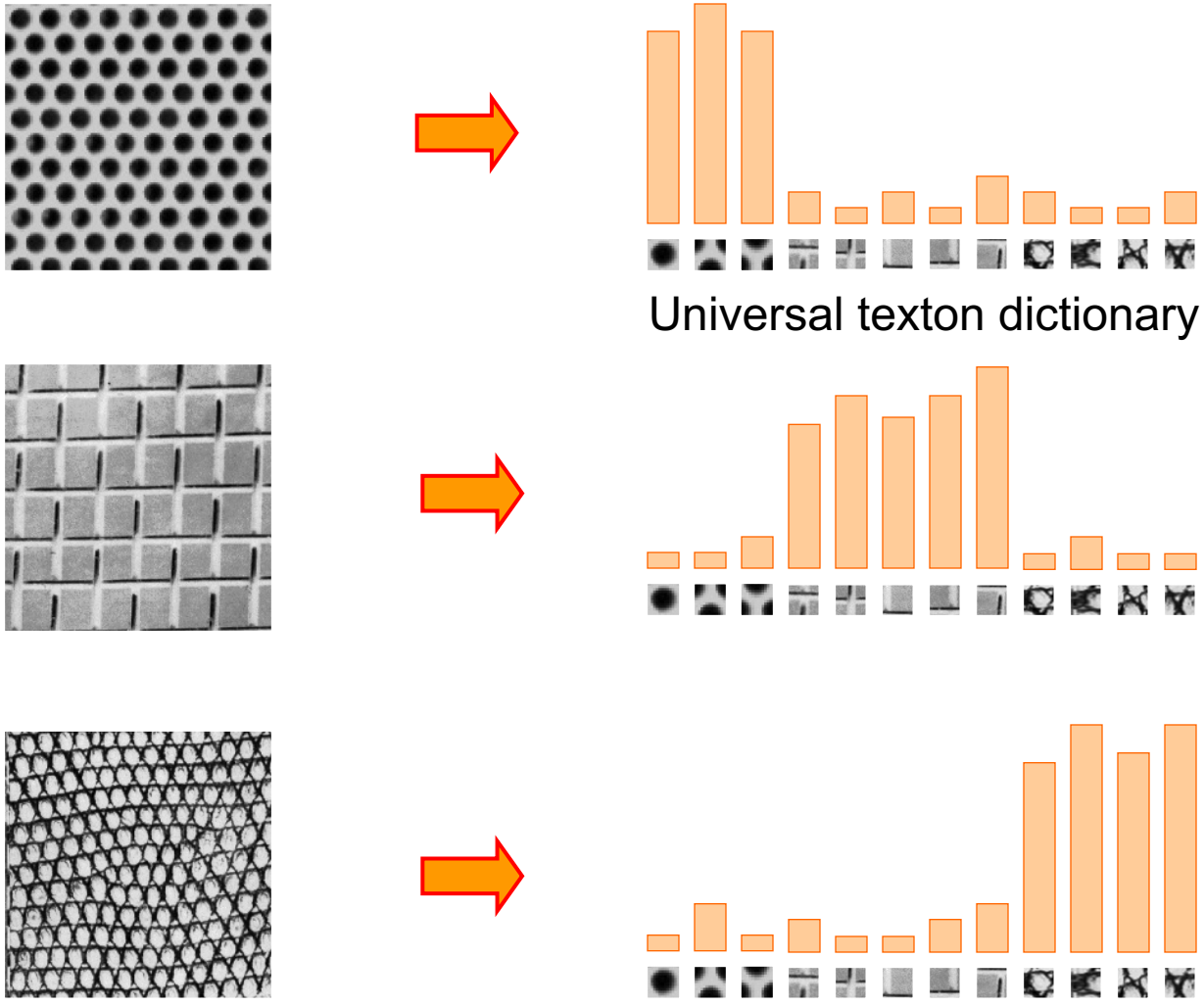
Origin 2: Texture recognition

- Texture is characterized by the repetition of basic elements or *textons*
- For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters



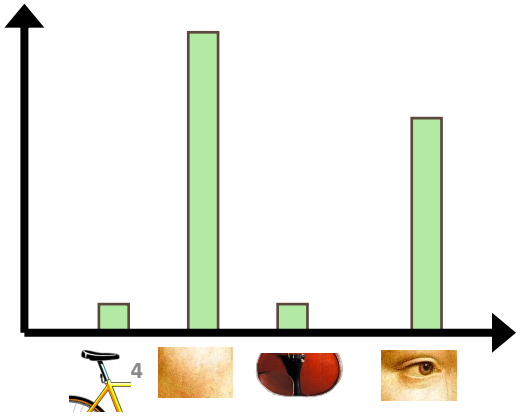
Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

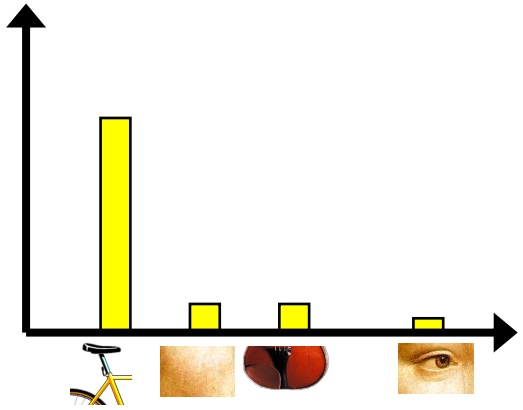
Origin 2: Texture recognition

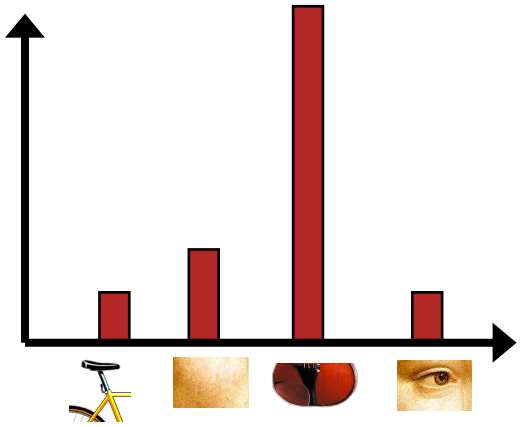


Can we describe objects as bag-of-words?

Can we describe objects as bag-of-words?

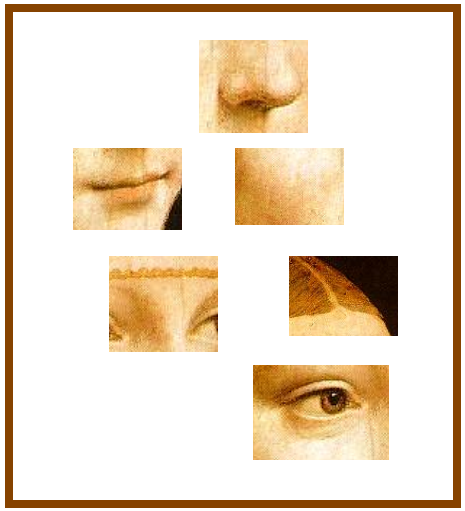






Bag of features: outline

1. Extract features

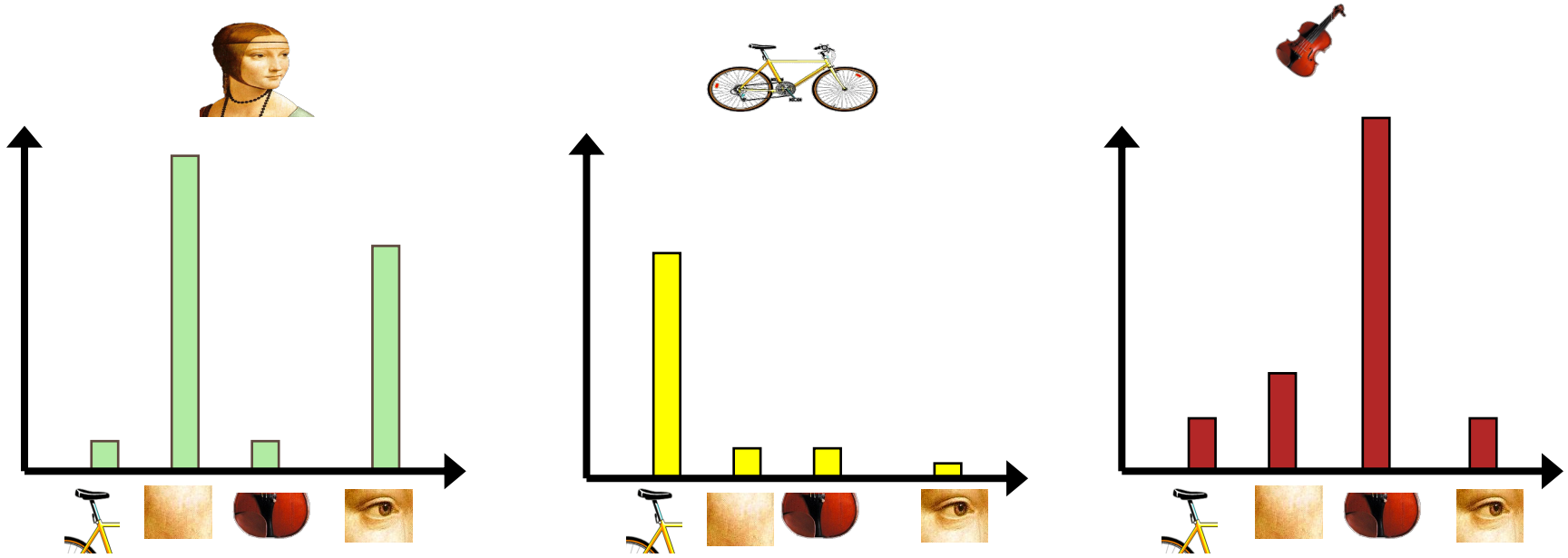


Bag of features: outline

1. Extract features
2. Learn “visual vocabulary”

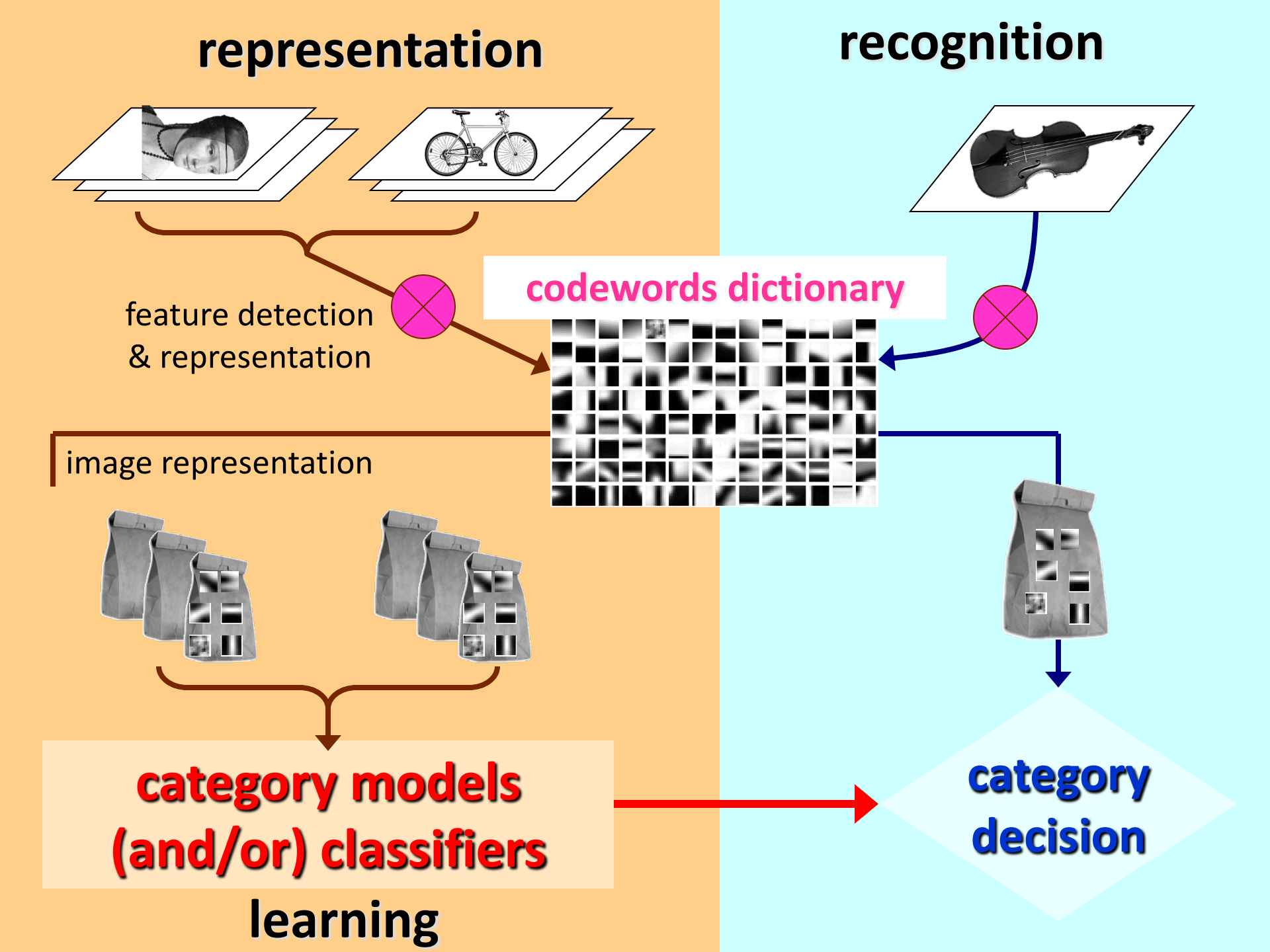
Bag of features: outline

1. Extract features
2. Learn “visual vocabulary”
3. Image representation:
   - Quantize features using visual vocabulary and represent images by frequencies of “visual words”



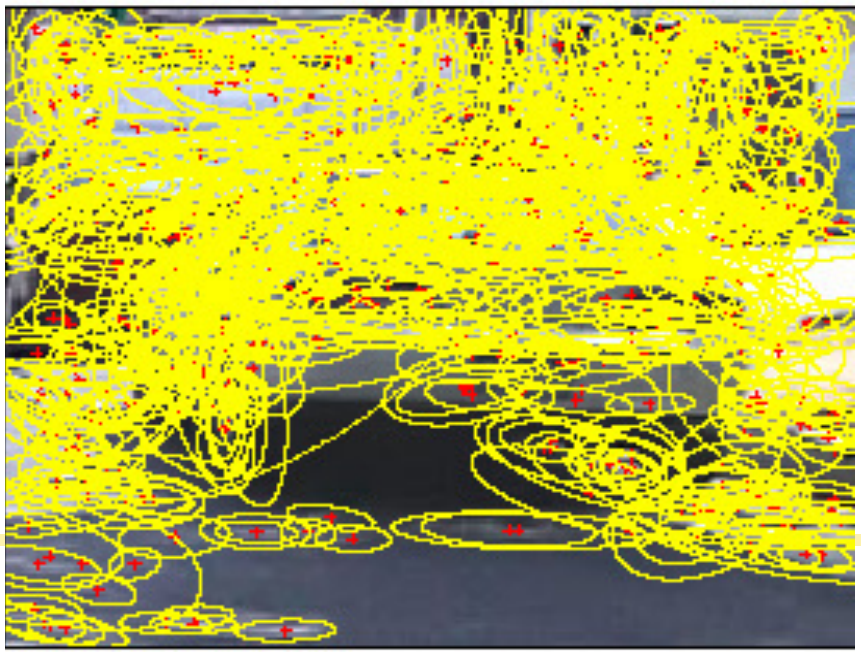
Bag of features: outline

1. Extract features
2. Learn “visual vocabulary”
3. Image representation:
   - Quantize features using visual vocabulary and represent images by frequencies of “visual words”
4. Classification


representation
recognition
codewords dictionary
feature detection
& representation
image representation
category models
(and/or) classifiers
learning
category
decision

1. Feature extraction:
Feature detection and description

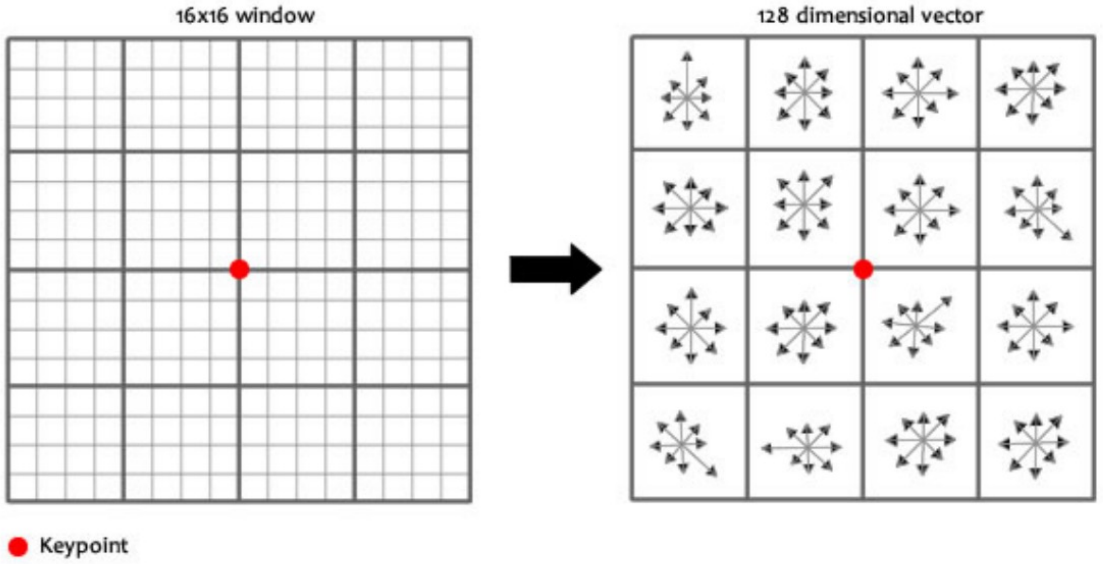
- Regular grid
- Interest point detector



SIFT descriptor

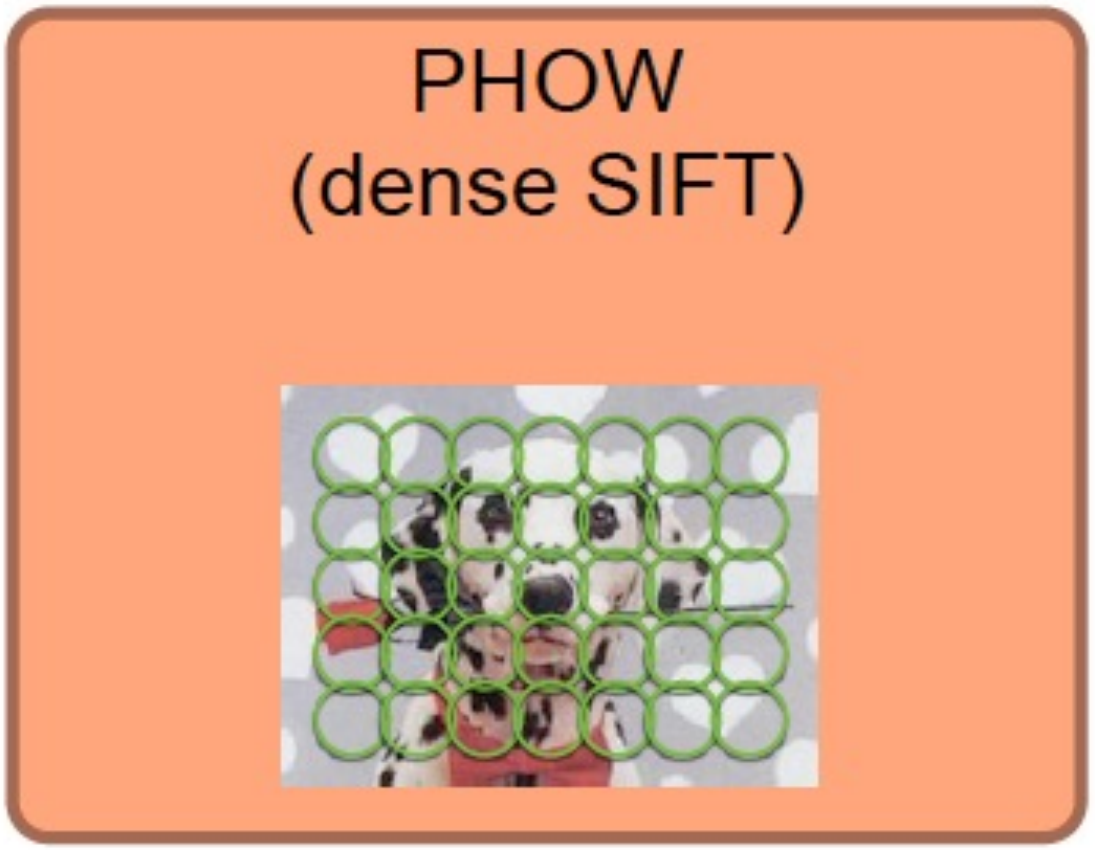
- SIFT **keypoint** [Lowe 2004]
  - extrema of the DoG scale space
  - blob-like image structure
  - oriented

- SIFT **descriptor**
  - 4 x 4 spatial histogram of gradient orientations
  - linear interpolation
  - Gaussian weighting

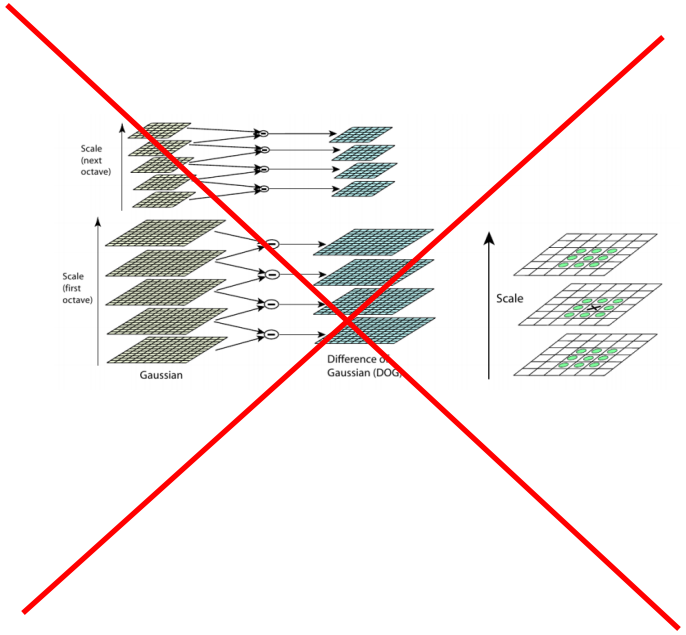


What is the size of the Sift descriptor?

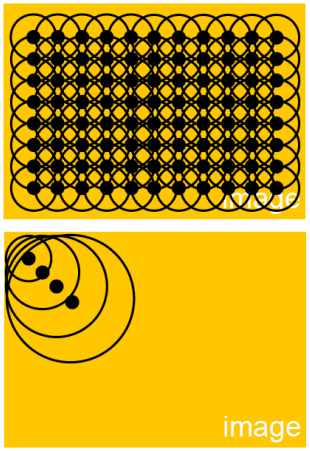
Pyramid Histogram of Visual Words (PHOW)



Dense SIFT: PHOW features

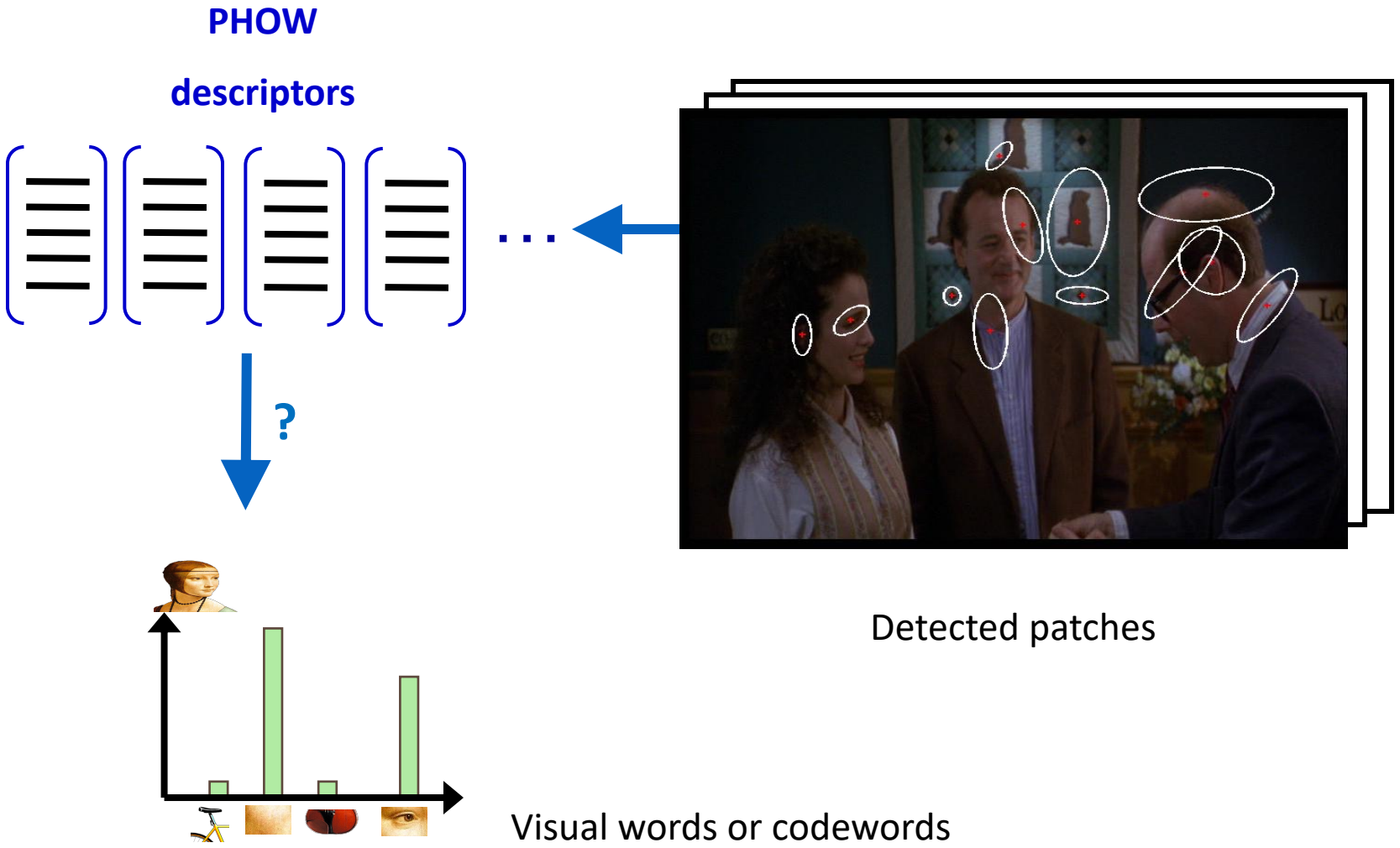


- **PHOW features**
  - [Bosch et al. 2006], [Bosch et al. 2007]
  - dense multiscale SIFT
  - uniform spacing (e.g. 5 pixels)
  - four scales (e.g. 5, 7, 10, 12 pixels)
- **Direct implementation**
  - for each scale
    - create a list of keypoints



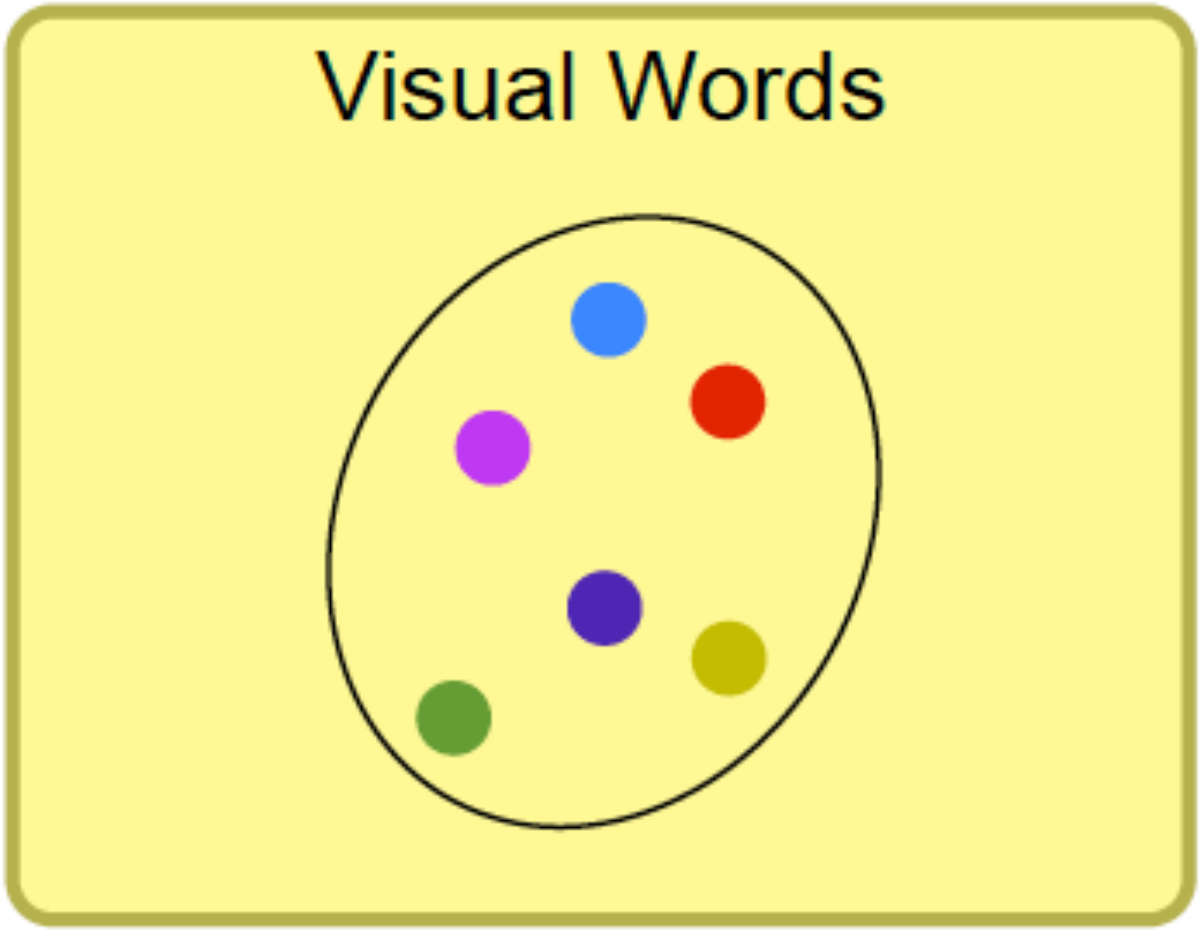
Large speed-up due to the uniform scale and sampling!

Feature extraction

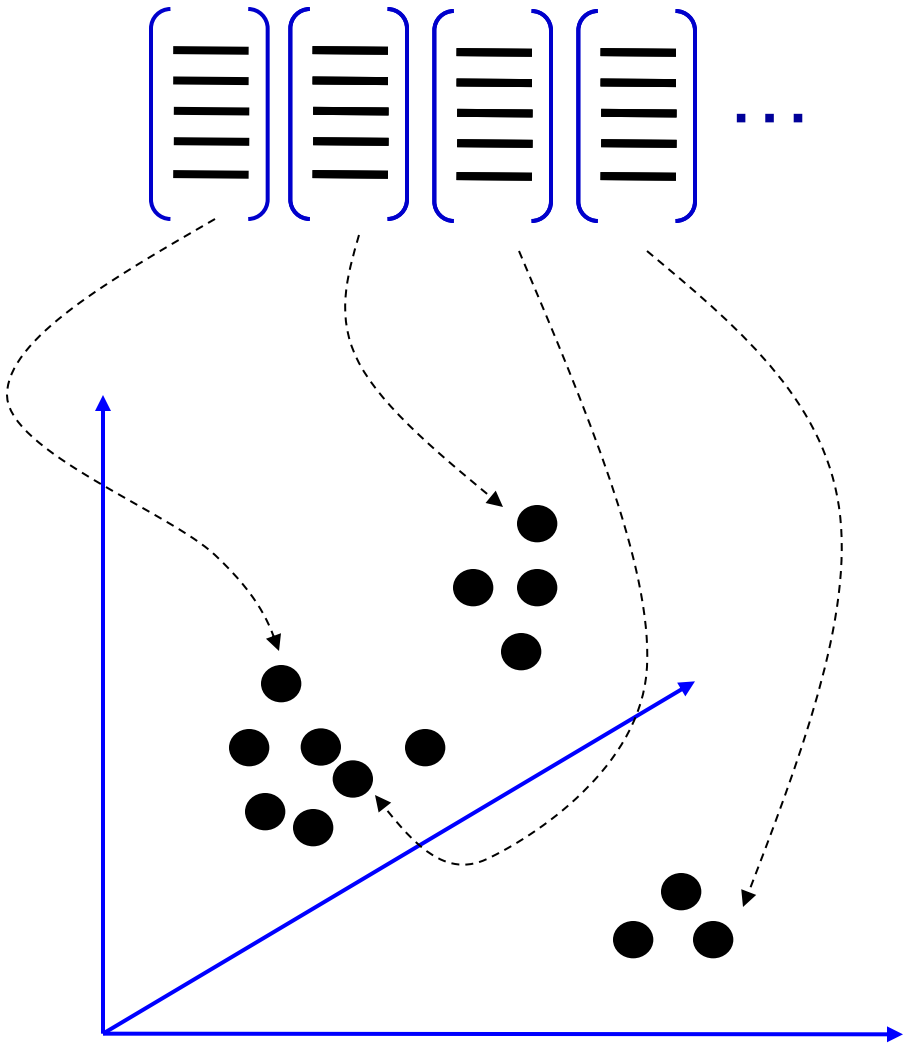


2. Learning the visual vocabulary

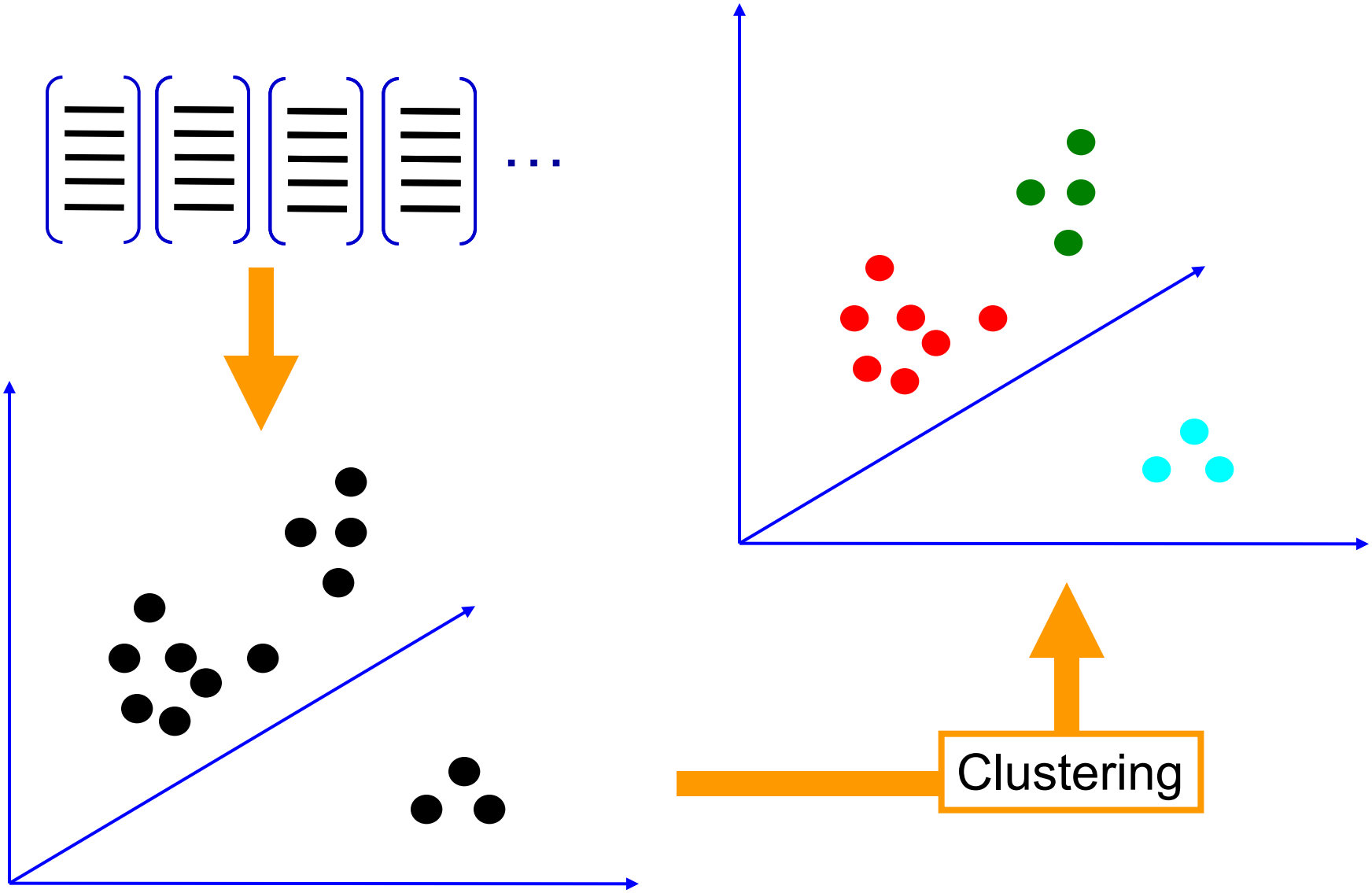
Let’s see how the visual vocabulary is built



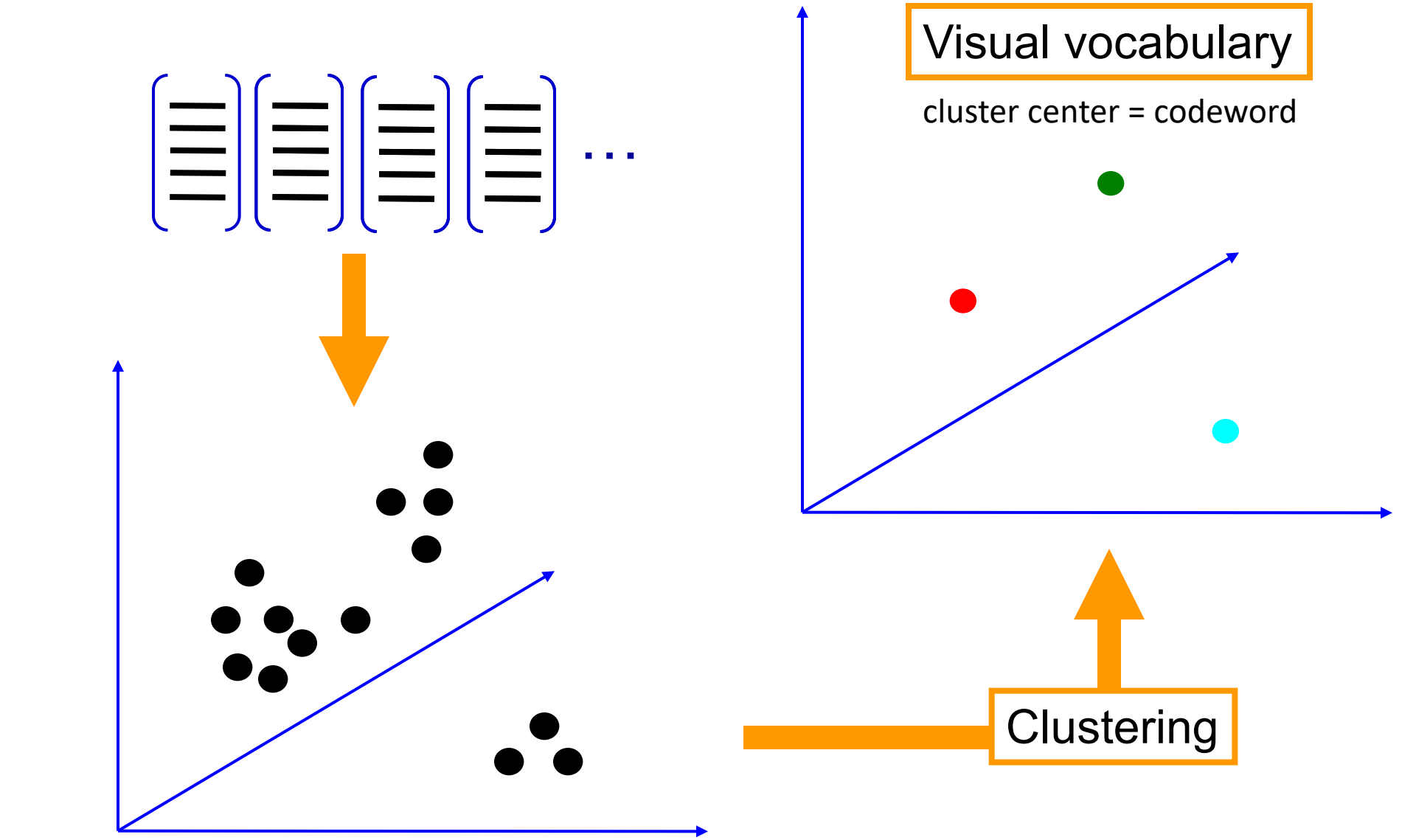
2. Learning the visual vocabulary



2. Learning the visual vocabulary



2. Learning the visual vocabulary



K-means clustering

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

$$D(X, M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (x_i - m_k)^2$$

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

From clustering to 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
  - If provided 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 words

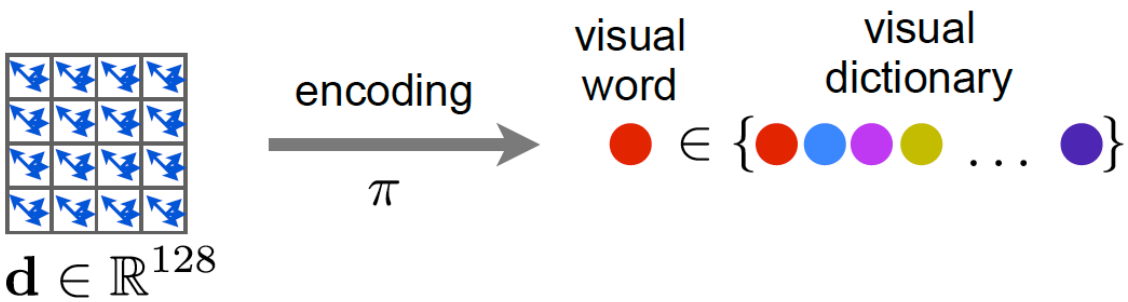
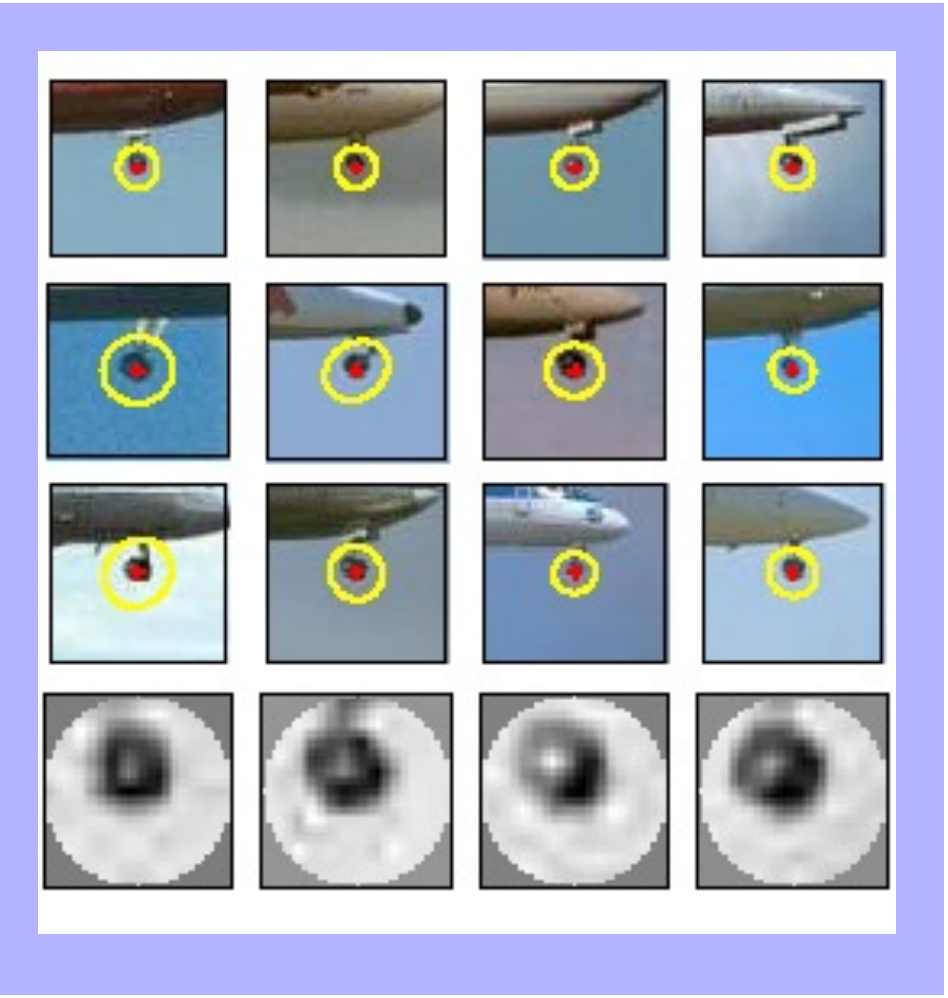
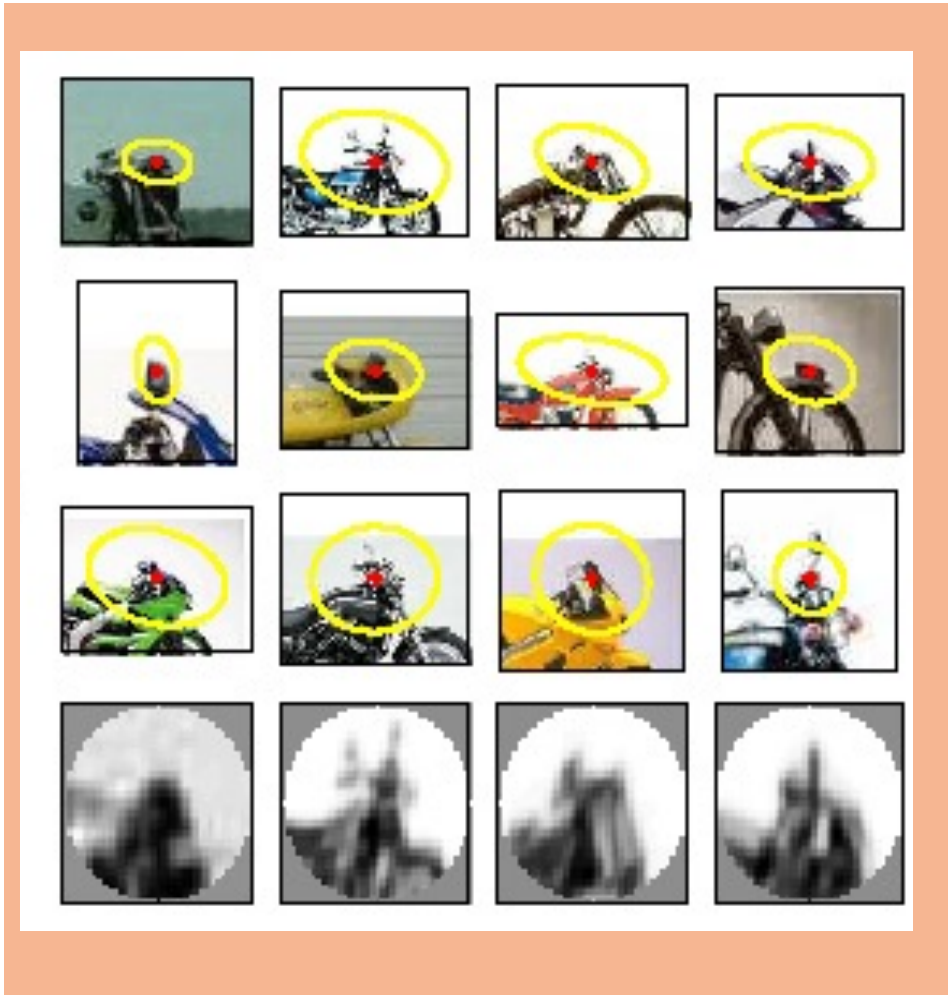
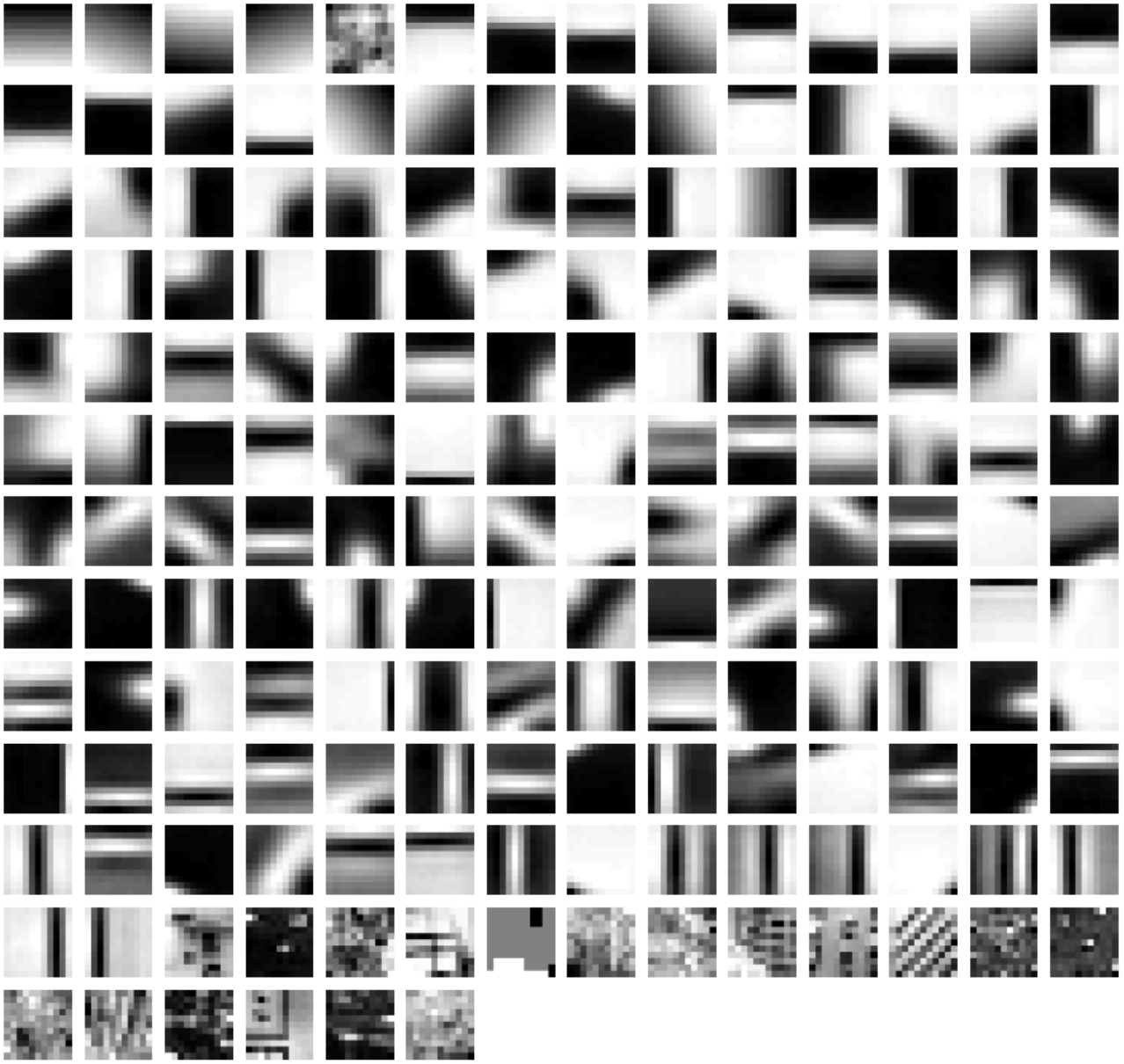


Image patch examples of visual words





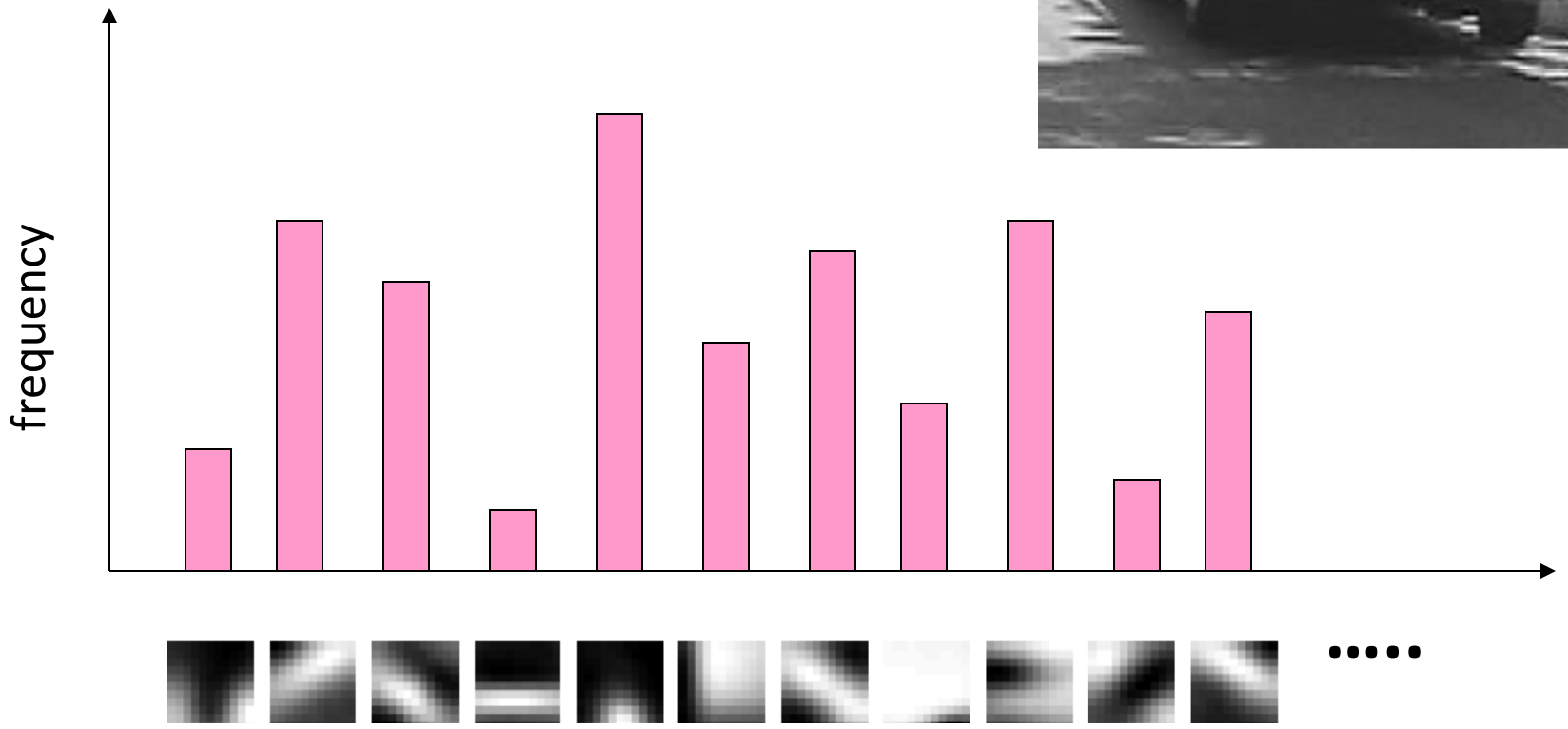
Example visual vocabulary



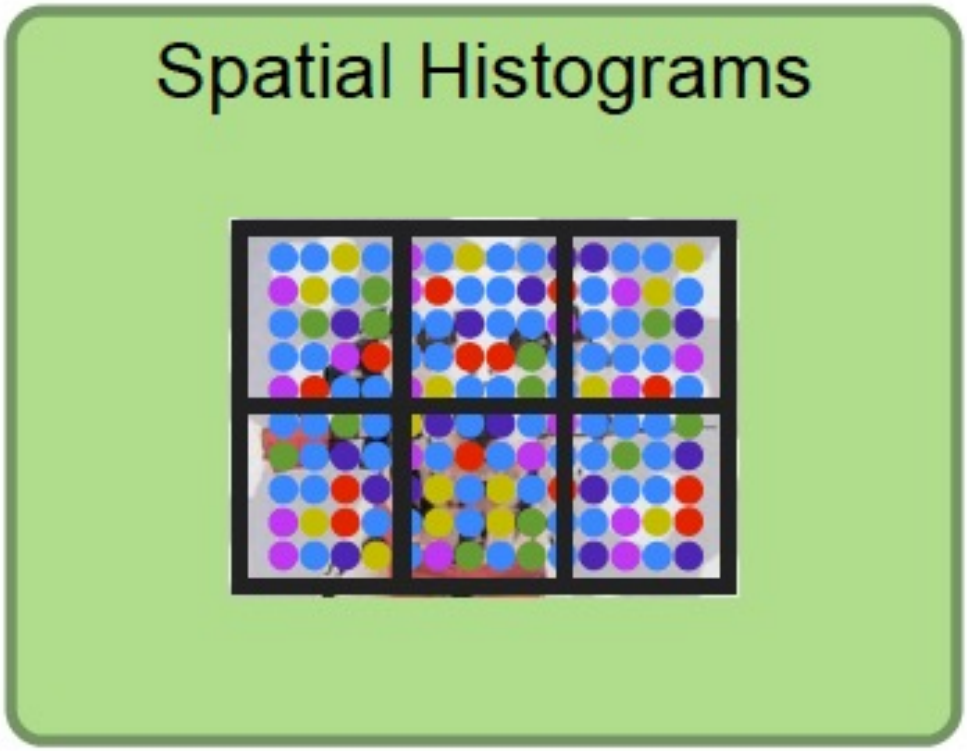
Visual vocabularies: Issues

- How to choose vocabulary size?
  - Too small: visual words not representative of all patches
  - Too large: quantization artifacts, overfitting
- Computational efficiency

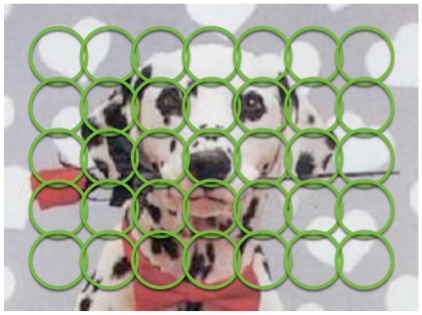
3. Image representation

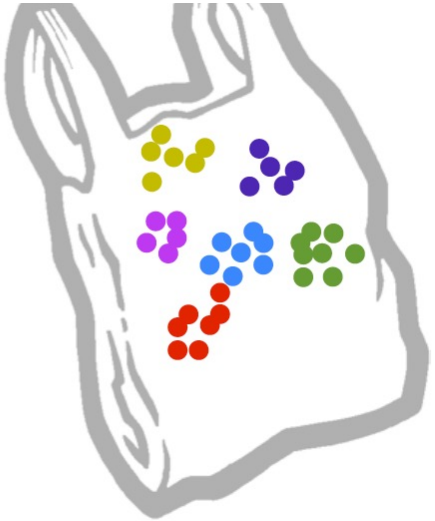


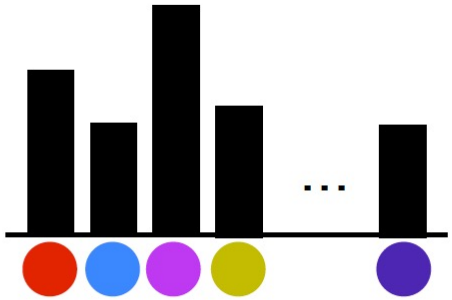
Spatial histograms



Bag-of-words



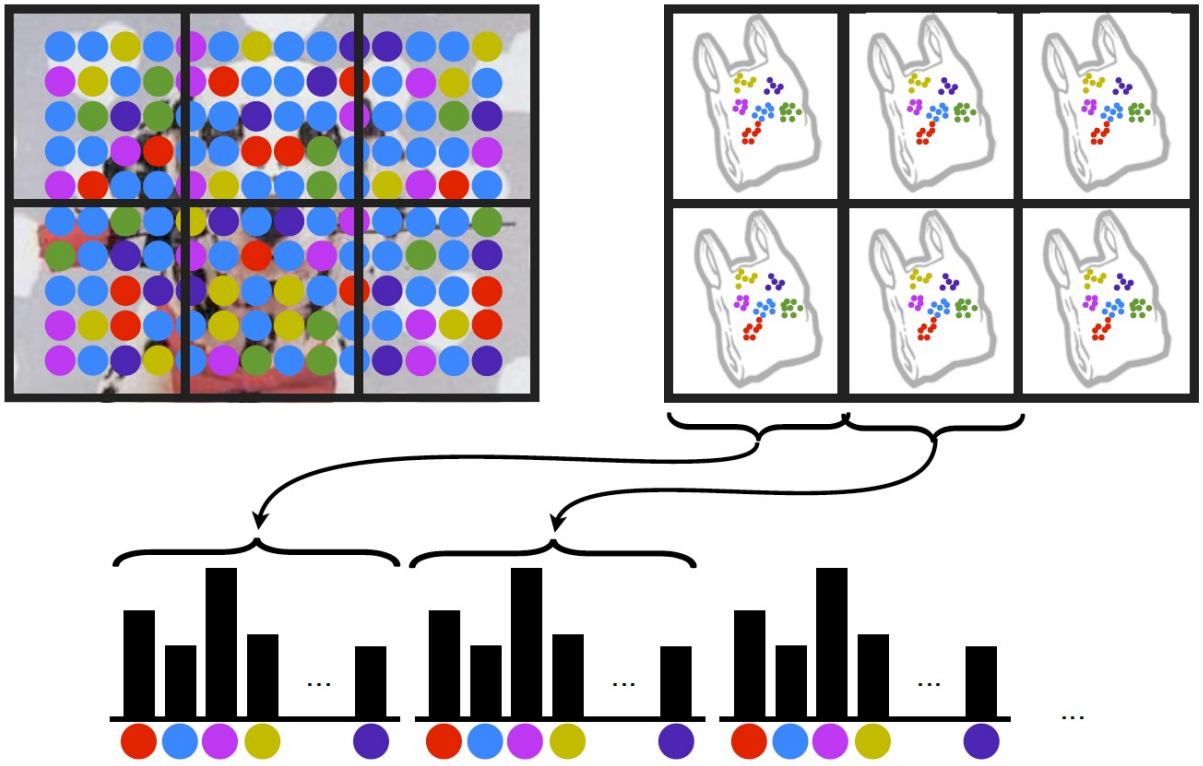




[Csurka et al. 2004]

histogram (bag) of visual words

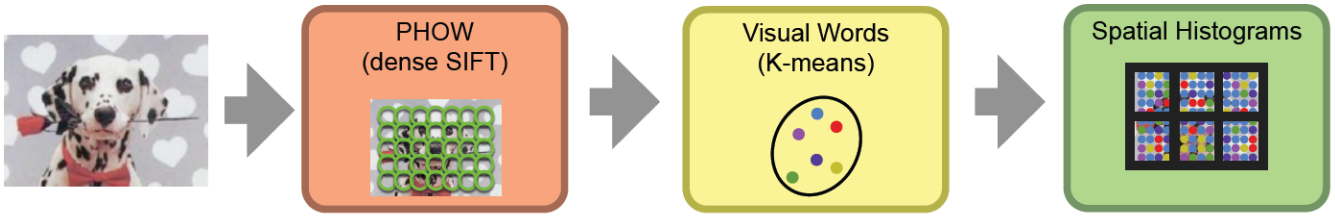
Spatial histograms

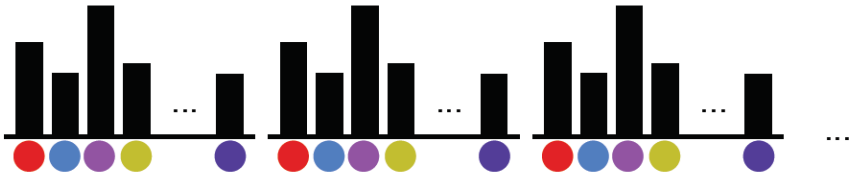


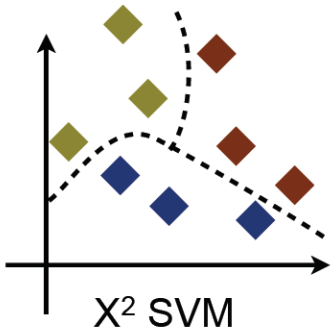
spatial histogram

[Lazebnik et al. 2004]

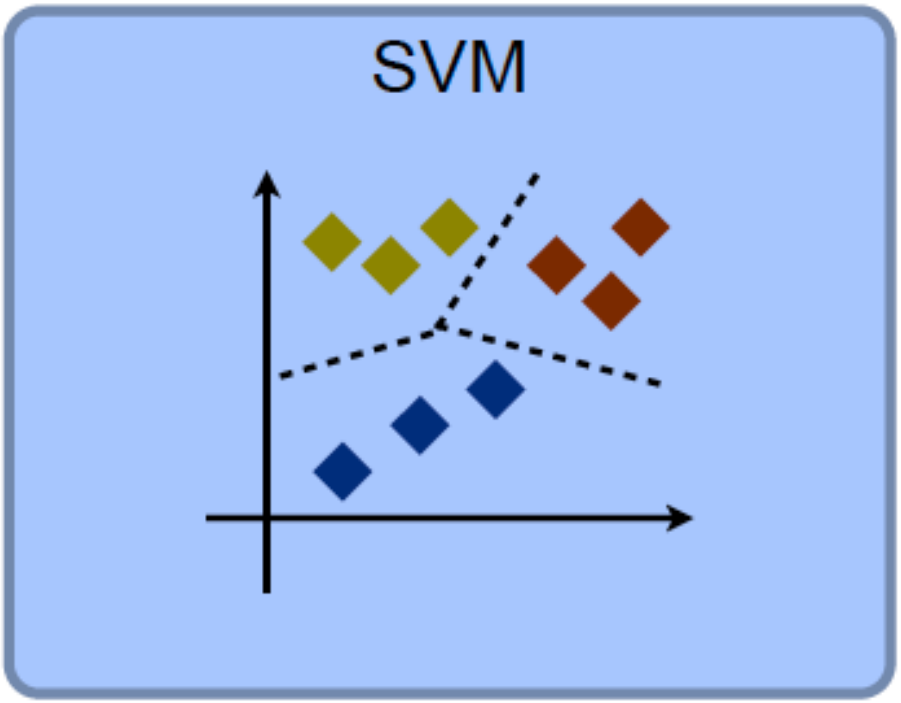
Summary: image descriptors







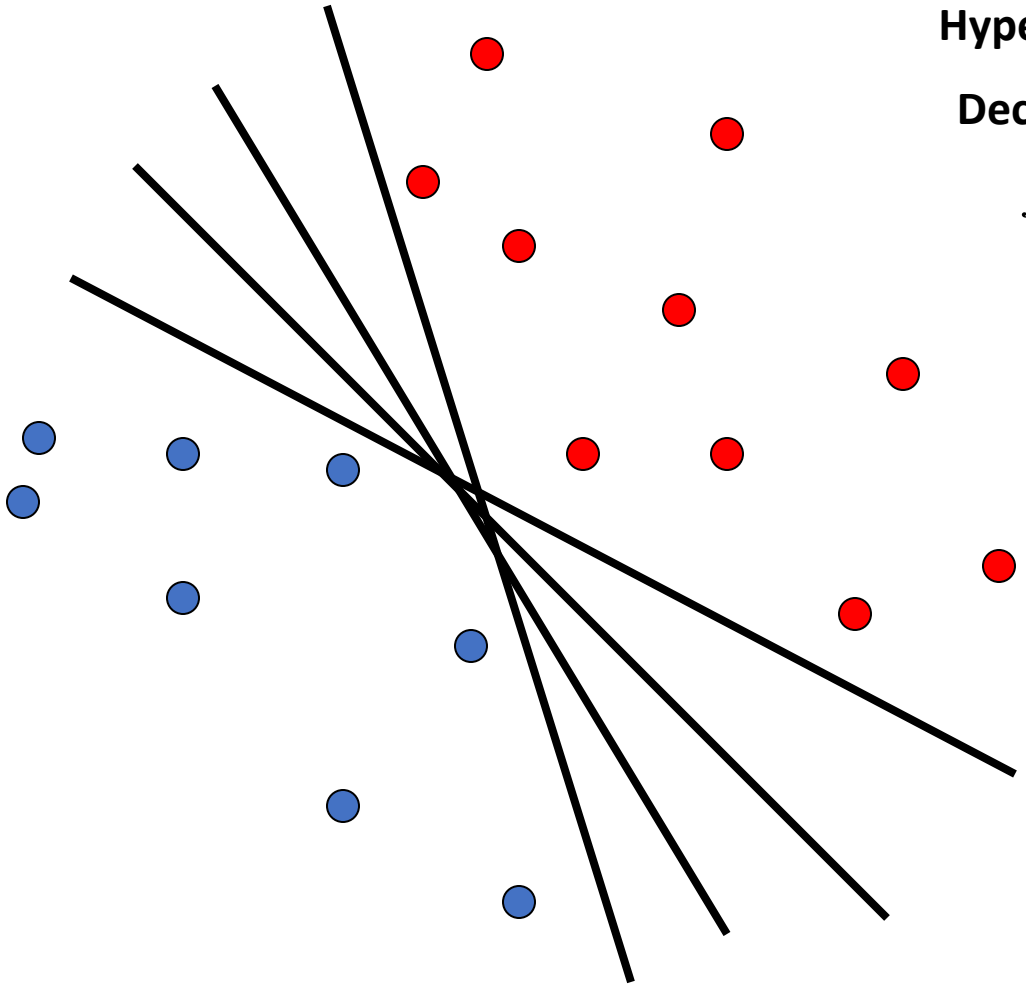
4. Classification



- "*A training algorithm for optimal margin classifiers*", Bernhard E. Boser, Isabelle M. Guyon, Vladimir N. Vapnik. Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory (COLT)
- *"The Nature of Statistical Learning Theory"*, Vladimir N. Vapnik. Springer-Verlag,1995. 2000.

Linear classifiers

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



Hyperplane: $(\mathbf{w} \cdot \mathbf{x}) + b = 0$

Decision function:

$$f_{\mathbf{w},b}(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b$$

bias

weight vector

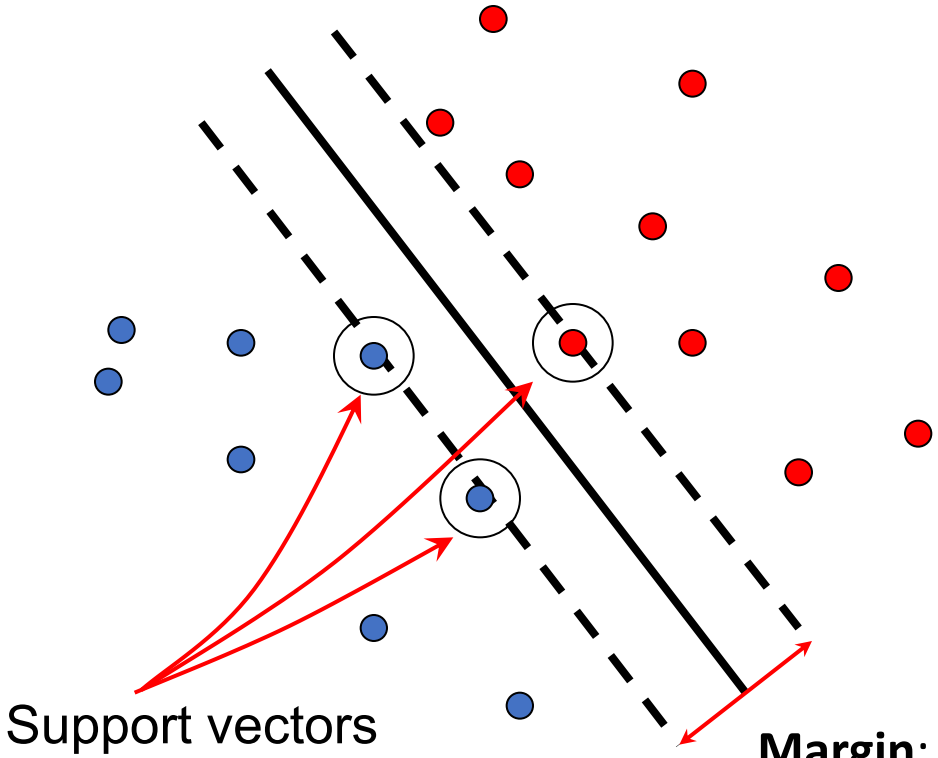
$\mathbf{x}_i$ positive: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

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

Which hyperplane is best?

Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples.
- Instead of fitting all points, focus on boundary points.



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

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

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

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

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

Margin:
perpendicular distance from the hyperplane to the closest samples

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

Finding the maximum margin hyperplane

1. Maximize margin 2/||**w**||
2. Correctly classify all training data:

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

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

Both cases are summarized as follows: $y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \forall i = 1, \ldots, N.$

Quadratic optimization problem:

Minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

Subject to $y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \forall i = 1, \ldots, N.$

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

Finding the optimal hyperplane

Instead of solving this QP directly

Solve a **dual formulation** of the SVM optimization problem (using Lagrange multipliers)

Dual formulation

- Has **simpler constraints**
- Allow us to use the **kernel trick.**

Reminder: Method of Lagrange Multipliers

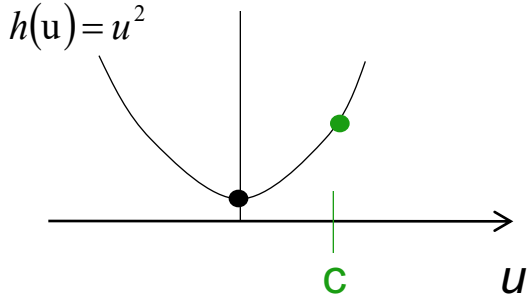
How to find the minimum of a convex function with linear constraints?

- Suppose that we want the minimum of $h(\mathrm{u})$ under the constraints:

$$g_i(\mathrm{u}) \geq 0, i = 1, \ldots, N$$

where each function $g_i(\mathrm{u})$ is affine.

Ex.: min *h(u)* s.t. u≥c



- We introduce one variable $\alpha_i \geq 0$ for each constraint and consider the **Lagrangian**:

$$L_P(\mathrm{u}, \alpha) = h(\mathrm{u}) - \sum_{i=1}^{N} \alpha_i g_i(\mathrm{u}).$$

Reminder: Method of Lagrange Multipliers

Method of Lagrange Multipliers

- For each α we can look for u_α which minimize $L_p(\mathrm{u},\alpha)$ (with no constraint), and note the dual function:

$$L_D(\alpha) = \min_{\mathrm{u}} L_P(\mathrm{u},\alpha)$$

- The dual variable α^* which maximizes $L_D(\alpha)$ gives the solution of the primal minimization problem with constraint: $u^* = u_{\alpha^*}$

- So, we first compute $\alpha^* = \arg\max_{\mathrm{u}} L_D(\alpha)$

  and then, compute the solution u^* using α^* .

Application to optimal hyperplane

In order to minimize: $\frac{1}{2}\|\mathbf{w}\|^2$

under the constraints: $\forall i = 1,\ldots,N, y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 \geq 0$

We introduce one dual variable α_i for each constraint (for each training point $\mathbf{x}_i$).

The Lagrangian is:

$$L_P(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1).$$

Solving the dual problem

We optimize the function $L_P(\mathbf{w}, b, \alpha)$

For that we use the equations:

$$\partial \mathrm{L_p} / \partial \mathrm{w} = 0$$

$$\partial \mathrm{L_p} / \partial b = 0$$

Giving,

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i .$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0.$$

Substitute in $L_P(\mathbf{w}, b, \alpha)$ $\longrightarrow$

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_i \mathbf{x}_i \cdot \mathbf{x}_j ,$$

Solving the dual problem

The dual problem is:

To find α^* that maximize $L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_i \mathbf{x}_i \cdot \mathbf{x}_j$,

under the (simple) constraints $\alpha_i \geq 0, \forall i = 1, \ldots, N$ and

$$\sum_{i=1}^{N} \alpha_i y_i = 0.$$

- α^* *can be easily found using classical optimization softwares.*
- The constraints are replaced by constraints on the Lagrangian multipliers → much easier to handle.

Recovering the optimal hyperplane

Once α^* is found, we recover $(\mathbf{w}^*, b^*)$ corresponding to the optimal hyperplane. $\mathbf{w}^*$ is given by:

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha^*_i y_i \mathbf{x}_i,$$

and b^* is computed using the equation: $y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b) - 1 = 0$

The decision function is therefore:

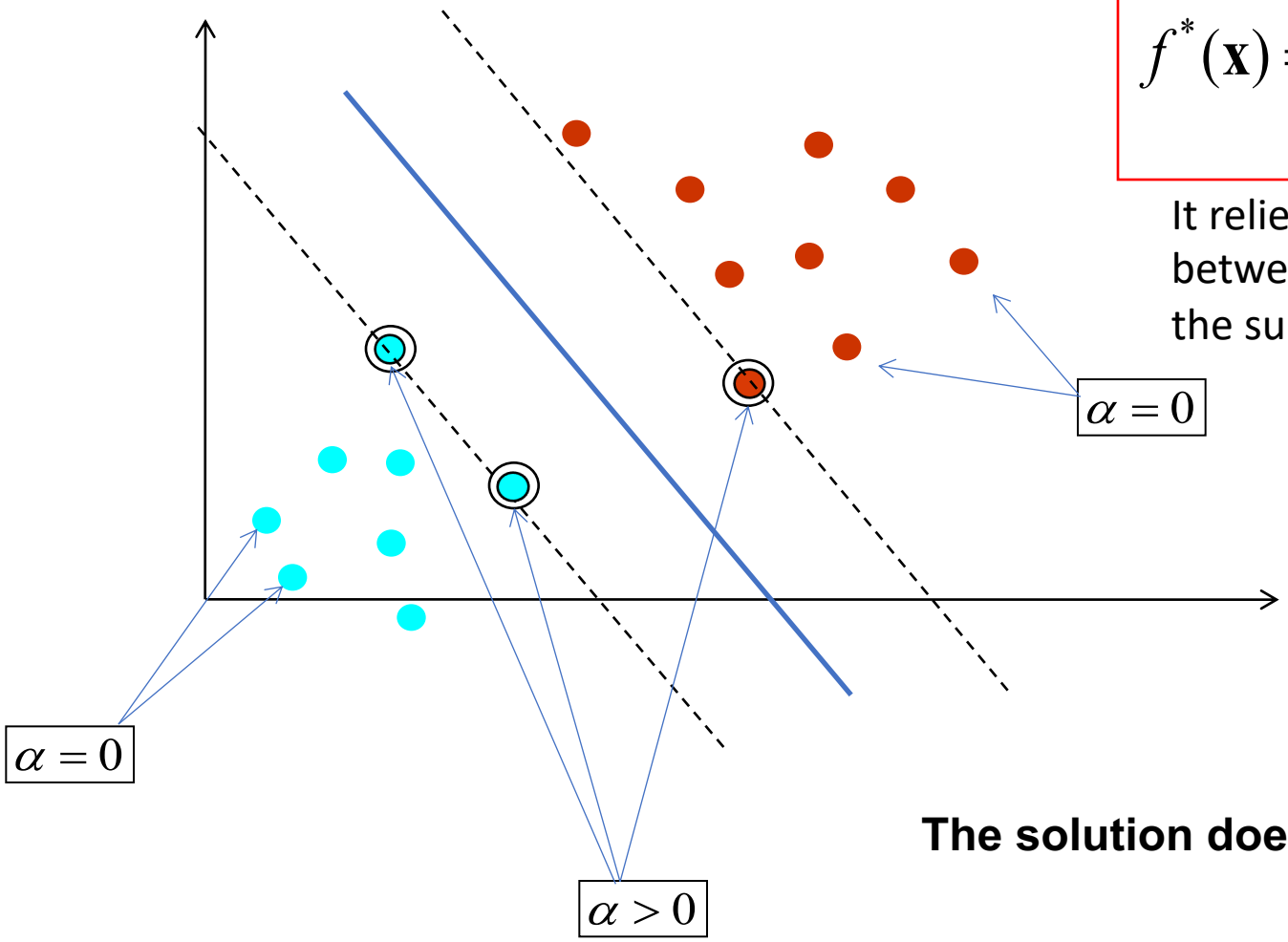
$$f^*(\mathbf{x}) = \mathbf{w}^* \cdot \mathbf{x} + b^*$$

$$= \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b^*.$$

← To evaluate a new sample **x** we need to evaluate this function.

Is this too much computational work?

Interpretation: support vectors



support vectors

Decision function:

$$f^*(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b^*.$$

It relies on an *inner product* between the test point ***x*** and the support vectors $\boldsymbol{x}_i$

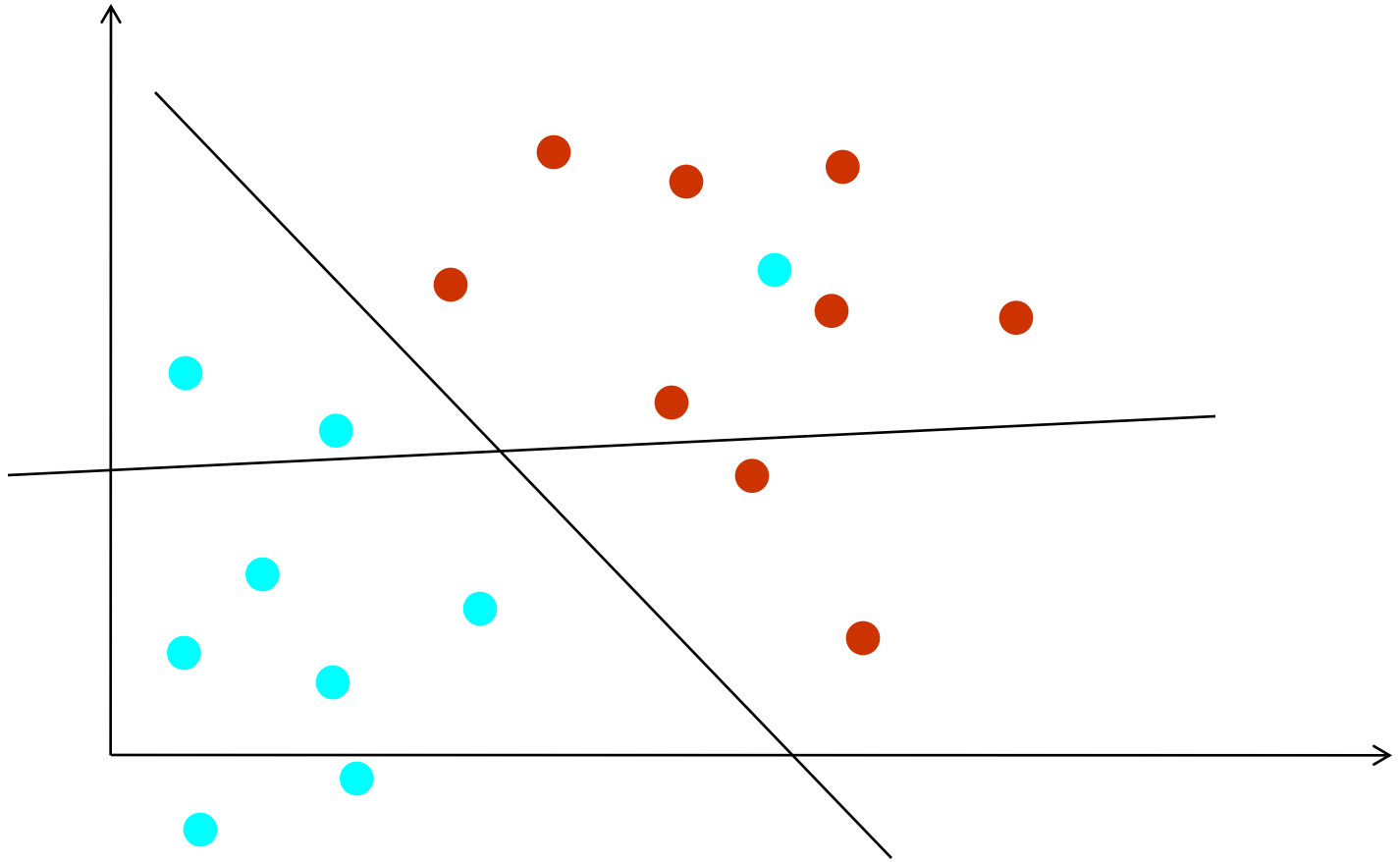
The solution does not change!

Simplest SVM: remarks

- Find the optimal hyperplane, which corresponds to the largest margin
- Can be solved easily using a dual formulation
- The solution is sparse: the number of support vectors can be very small compared to the size of the training set
- Only support vectors are important for prediction of future points. All other points can be forgotten.

In general, training sets are non linearly separable

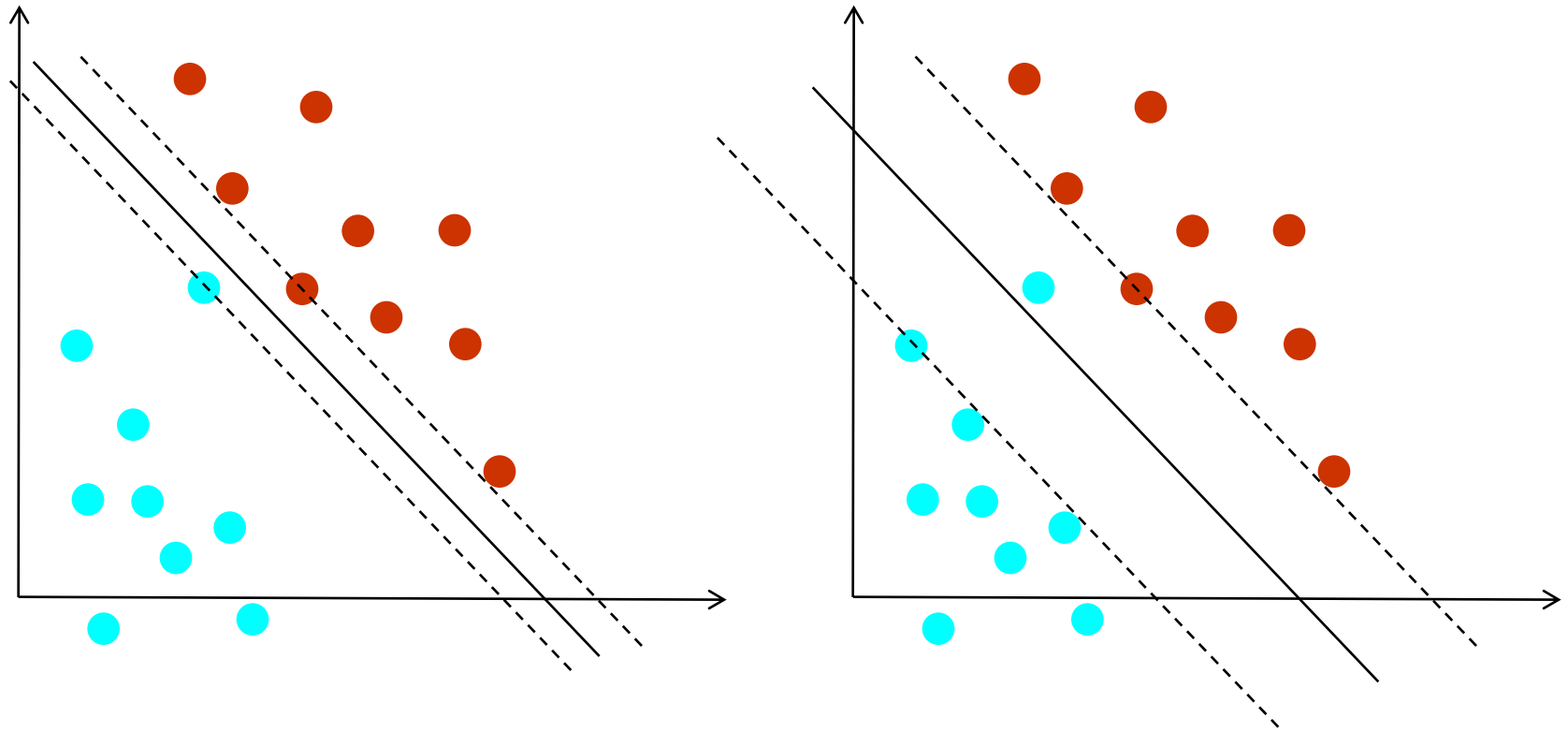
There is no **"correct"** linear classifier for the training set.



Constraints can not be satisfied.

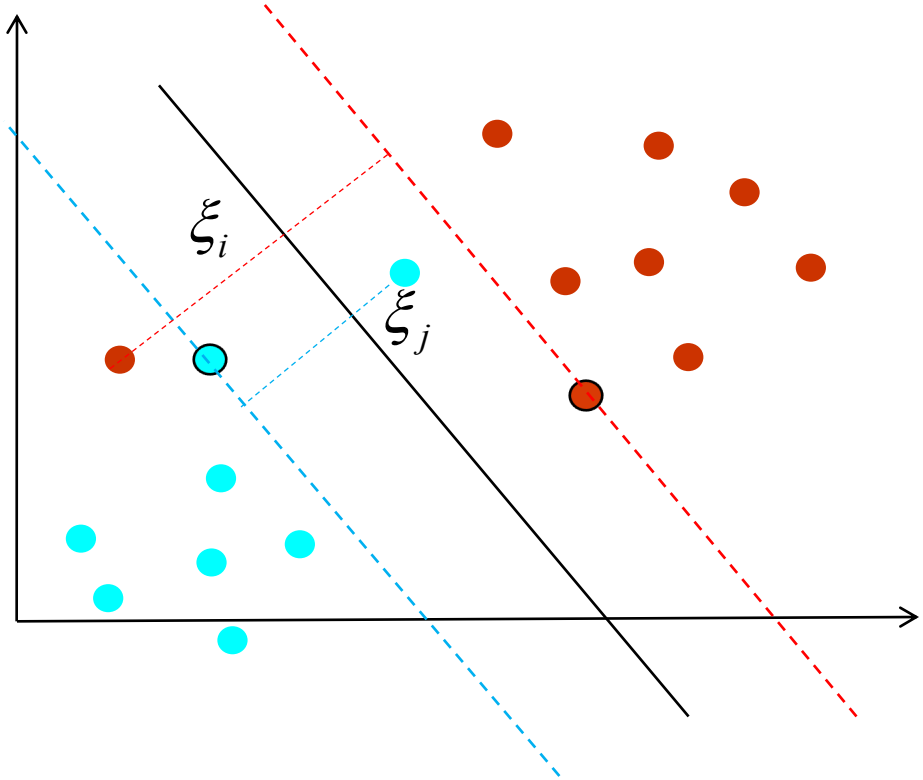
Proposal: Permit the algorithm to misclassify some of the data points without affecting the final result.

Even if *correct* linear classifier exists…



- Even if a decision boundary is possible exactly separating the data,
- It is probably not desirable:
  - If the data has **noise** and **outliers**, a **soft margin** decision boundary that ignores a few data points is better.

Forcing a solution



- Minimize the distance between these points and their correct plane.
- A soft margin is generated.

Change constraints: $\forall i = 1,\ldots,N, y_i\left(\mathbf{w}\cdot\mathbf{x}_i + b\right) \geq 1$

to:

$$\forall i = 1,\ldots,N, y_i\left(\mathbf{w}\cdot\mathbf{x}_i + b\right) \geq 1 - \xi_i$$

where ξ_i are called *slack variables*

Optimization for non linearly separable case

So, the problem becomes:

Add cost term to penalize large slack variables (violations)

Minimize $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$

under the constraints $\forall i = 1,\ldots,N,\ y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i$

and $\xi_i \geq 0$

- The soft margin parameter C allows us to trade off between **correctness** and **robustness** of the classifier.

Optimization for non linearly separable case

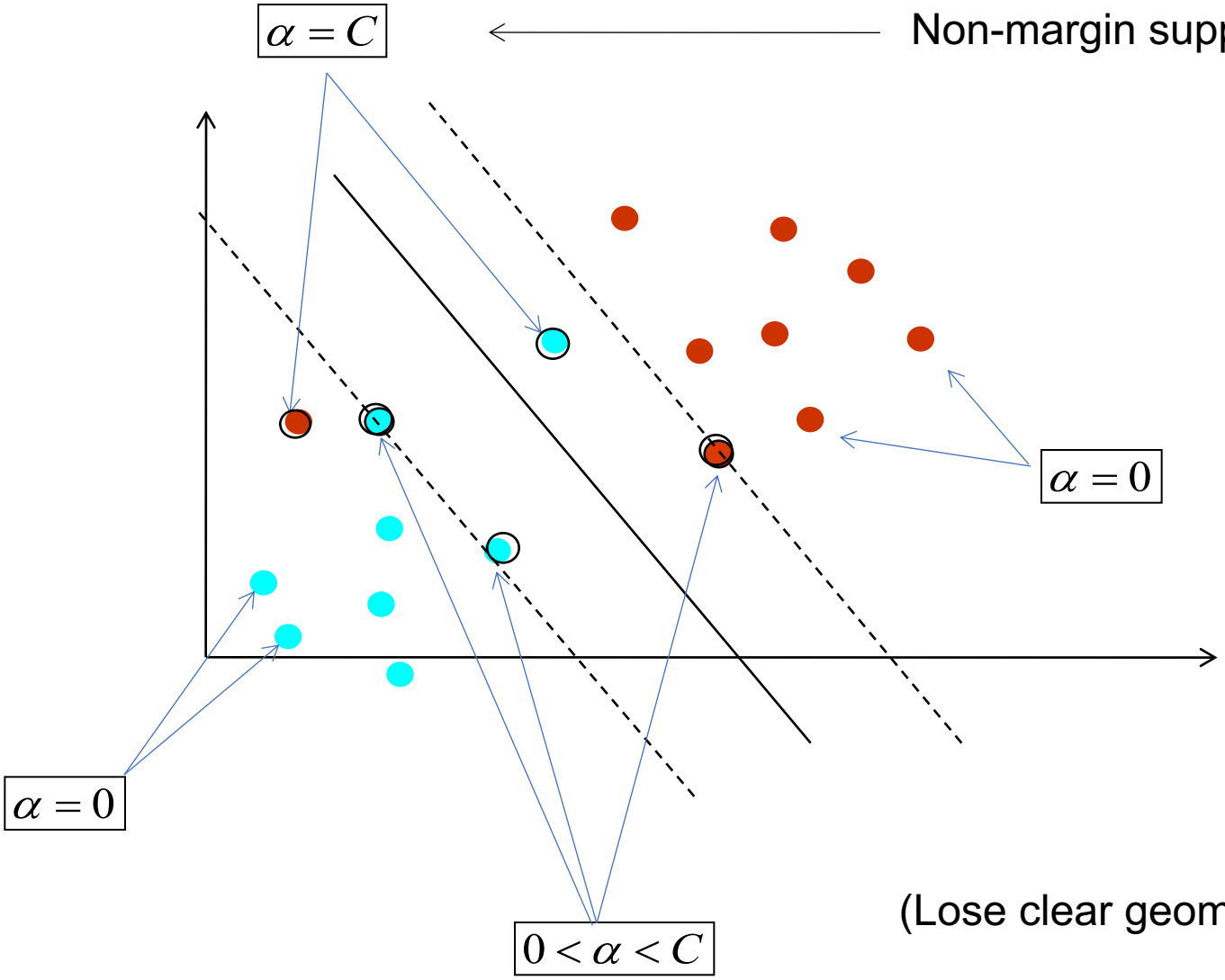
So, the dual problem becomes:

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_i \mathbf{x}_i \cdot \mathbf{x}_j,$$

under the constraints

$$\begin{cases} 0 \le \alpha_i \le C, i = 1, \ldots, N \\ \sum_{i=1}^{N} \alpha_i y_i = 0. \end{cases}$$

Interpretation



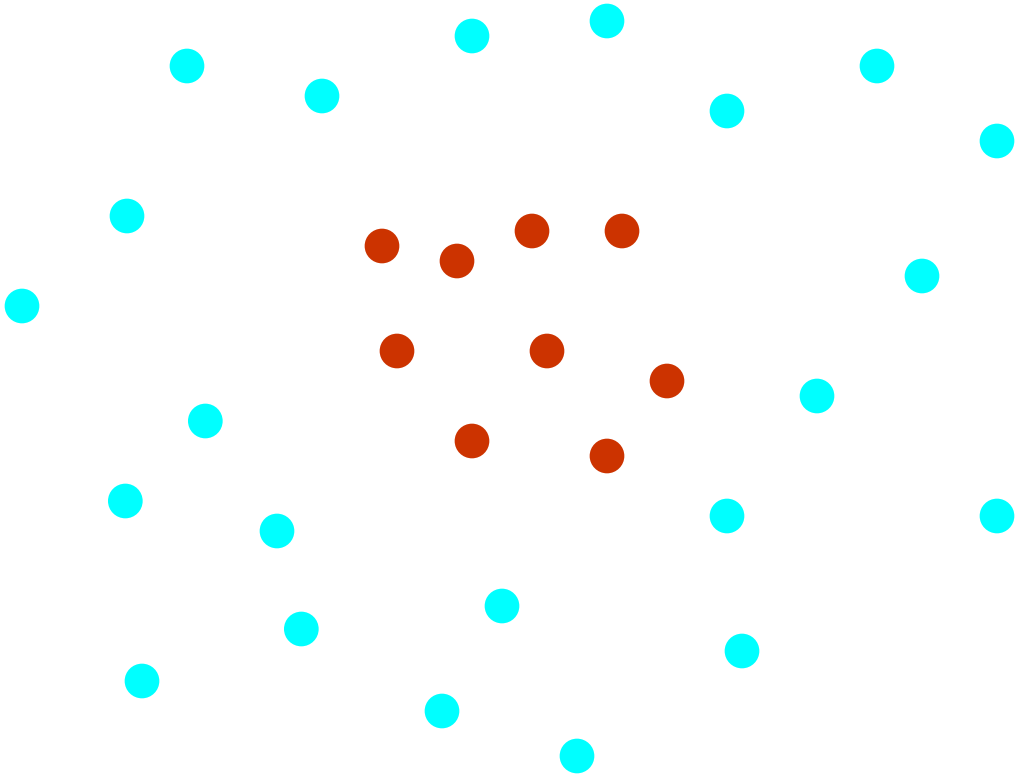
(Lose clear geometric intuition)

Remarks

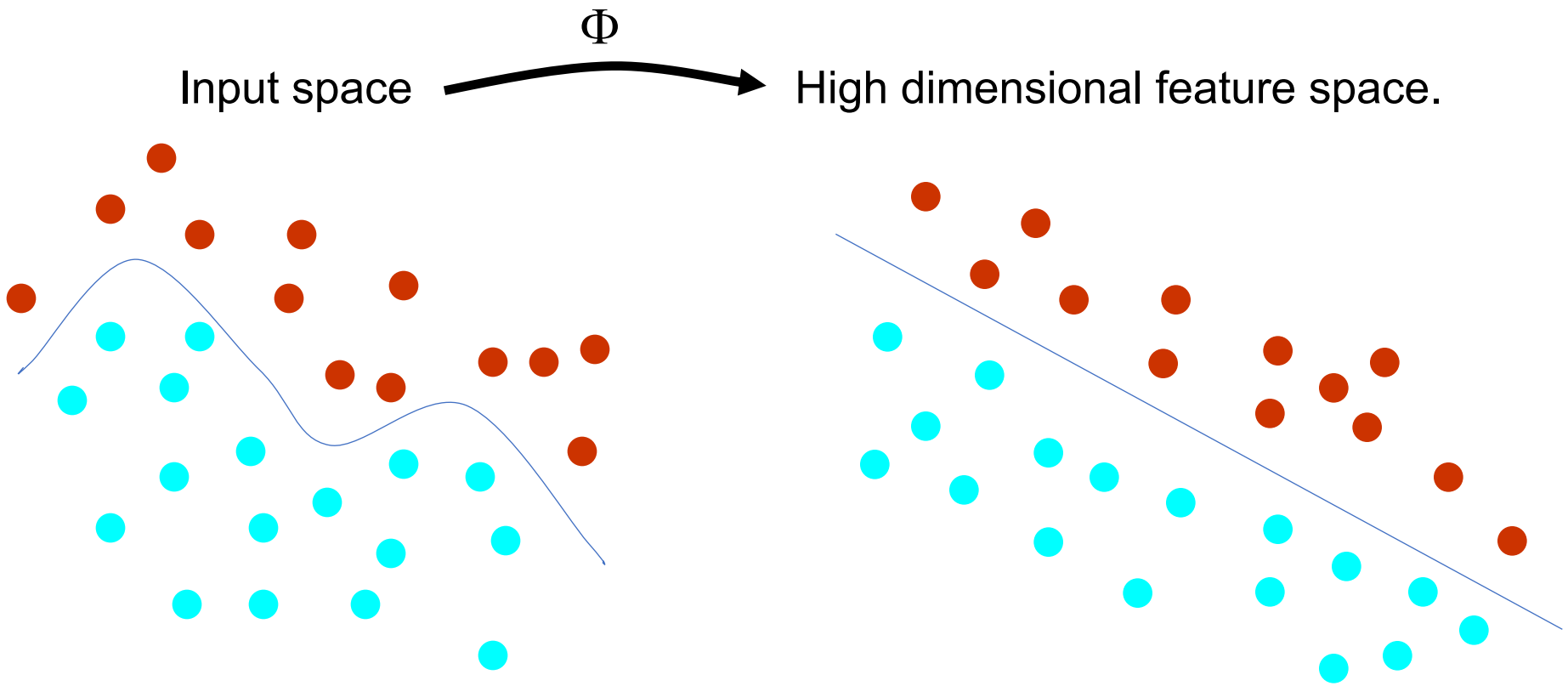
- This formulation finds a trade-off between:
  - minimizing the training error
  - maximizing the margin
- Soft margin parameter specifies this trade-off.
- All properties of the separable case are conserved (support vectors, sparseness, computation efficiency,...)

General SVM: Non-linear classifiers for general training sets

Sometimes linear classifiers are not interesting



Solution: non-linear mapping to a richer feature space

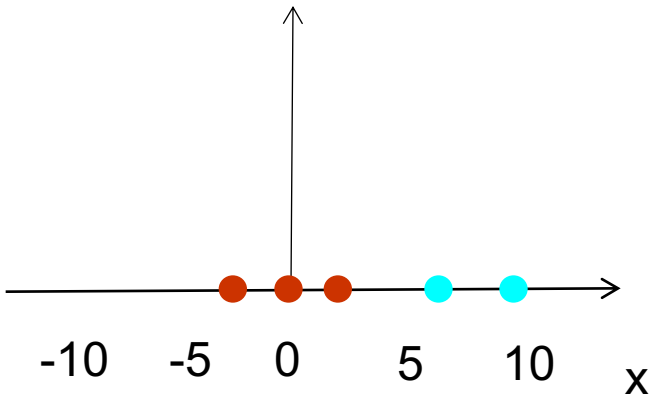


- To define a non-linear classifier, we use a **non-linear mapping** to map the input space to a high dimensional feature space.

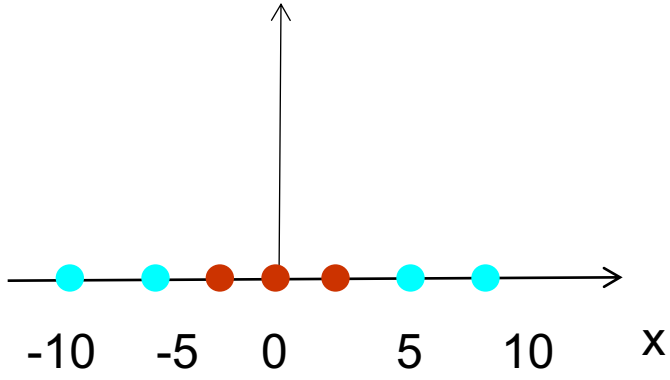
- If one is lucky (or smart) and chooses a good function, the data will become separable in the resulting higher dimensional space.

Examples in 1-D

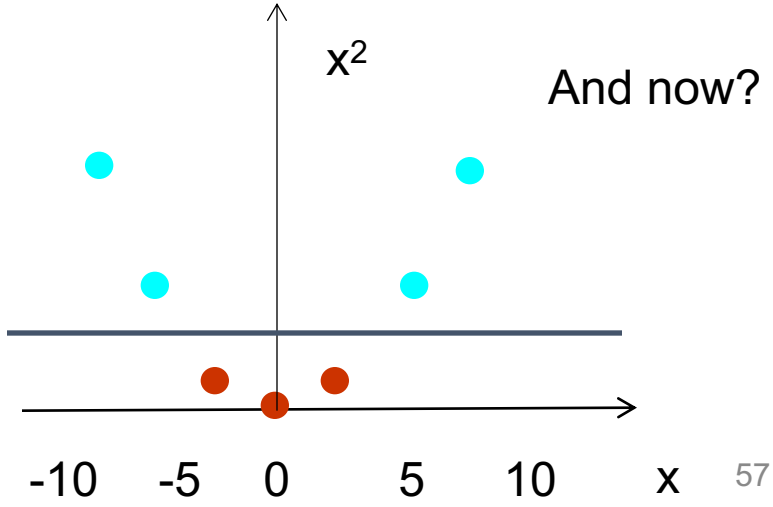
Can an SVM correctly classify this data?



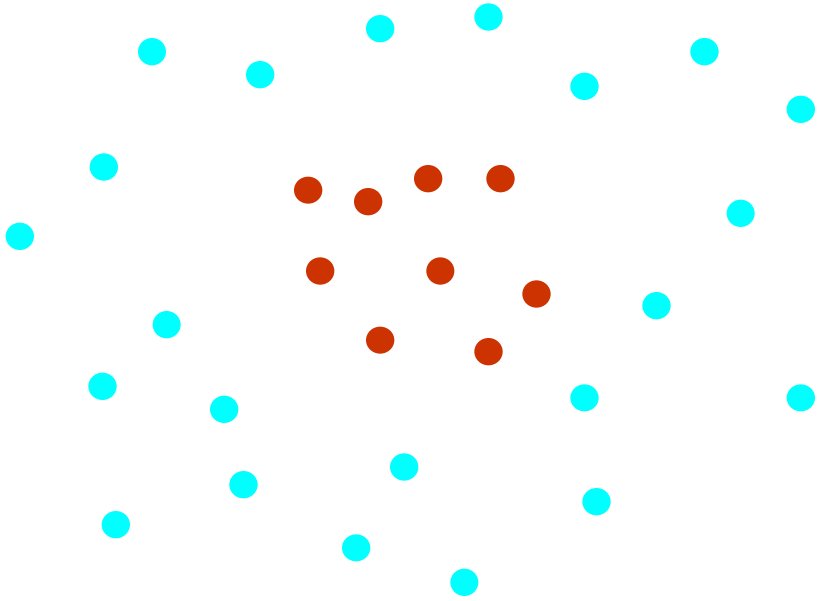
What about this?



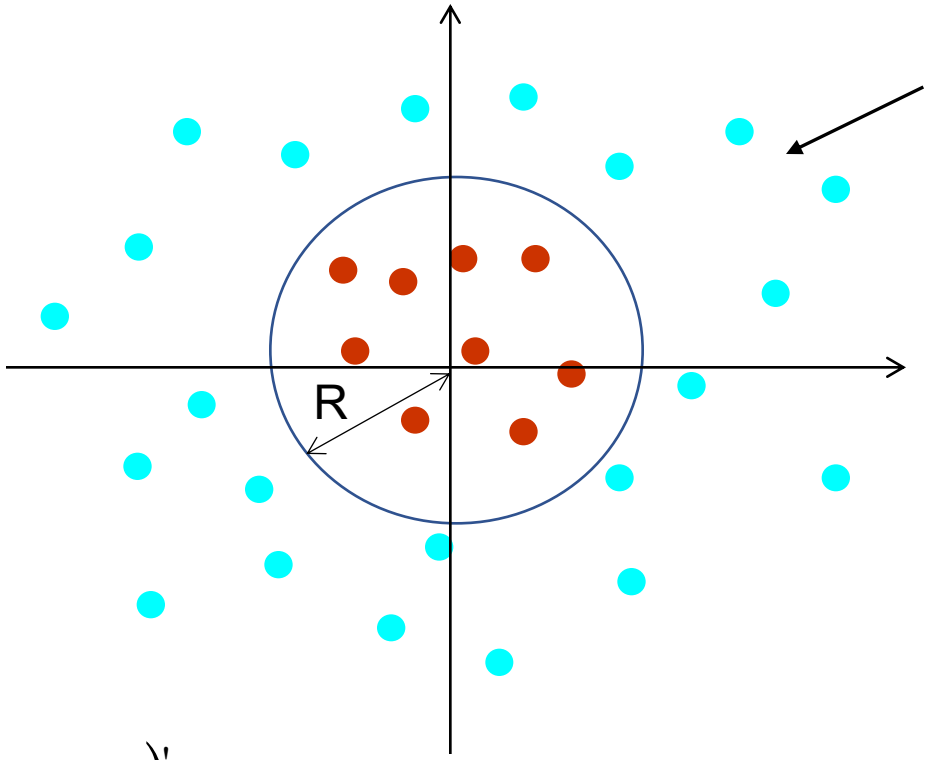
And now?



Example



Example



Non-linear funtion in the original feature space

Let $\Phi(\mathbf{x}) = (x_1^2, x_2^2)'$, $\mathbf{w} = (1,1)'$ and $b = -1$. Then the decision function is:

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

← **Linear funtion** in the new feature space

$$= x_1^2 + x_2^2 - 1,$$
$$= x_1^2 + x_2^2 - R^2,$$

Training a SVM in the feature space

The dual problem is to maximize

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j),$$

under the constraints

$$\begin{cases} 0 \le \alpha_i \le C, i = 1, \ldots, N \\ \sum_{i=1}^{N} \alpha_i y_i = 0. \end{cases}$$

The decision function is: $f^*(\mathbf{x}) = \mathbf{w}^* \cdot \Phi(\mathbf{x}) + b^*$

Kernel definition

For a given mapping Φ from the space of objects X to some feature space, the kernel of two objects $\mathbf{x}$ and $\mathbf{x}'$ is the inner product of their images in the features space:

$$\forall \mathbf{x}, \mathbf{x}' \in X, K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}').$$

Example: if $\Phi(\mathbf{x}) = \left(x_1^2, x_2^2\right)'$, then

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = (x_1)^2 \left(x_1'\right)^2 + (x_2)^2 \left(x_2'\right)^2.$$

Kernel example

For any vector $\mathbf{x} = (x_1, x_2)^T$, consider the mapping:

$$\Phi : \Re^2 \to \Re^6, \quad \Phi(\mathbf{x}) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1\right)^T.$$

The associated kernel is:

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') = \left(x_1x_1', x_2x_2' + 1\right)^2 = \left(\mathbf{x} \cdot \mathbf{x}' + 1\right)^2$$

Useful property of this mapping: allows the computation of the inner products of feature vectors Φ(**x**) and Φ(**x'**) in R^6 by just squaring the inner product of the data vectors **x** and **x'** in R^2.

Training a SVM in the feature space

Replace each $\mathbf{x} \cdot \mathbf{x}'$ in the SVM algorithm by $K(\mathbf{x}, \mathbf{x}')$

The dual problem is to maximize

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j),$$

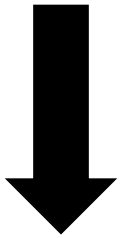
under the constraints

$$\begin{cases} 0 \le \alpha_i \le C, i = 1, \ldots, N \\ \sum_{i=1}^{N} \alpha_i y_i = 0. \end{cases}$$

Predicting with a SVM in the feature space

The decision function becomes:

$$f^*(\mathbf{x}) = \mathbf{w}^* \cdot \mathbf{x} + b^* = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b^*.$$



$$f^*(\mathbf{x}) = \mathbf{w}^* \cdot \Phi(\mathbf{x}) + b^* = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b^*.$$

The kernel trick

- The explicit computation of $\Phi(\mathbf{x}_i)$ is not necessary.
  - The kernel $K(\mathbf{x}, \mathbf{x}')$ is enough.
- SVM work implicitly in the feature space.
- It is sometimes possible to easily compute kernels which correspond to complex large-dimensional feature spaces.

→All the considerations of the previous sections holds since we are still doing a linear separation, but in a different space.

Common kernels

Polynomial

$$K(\mathbf{x},\mathbf{x}') = \left((\mathbf{x}\cdot\mathbf{x}')+1\right)^d, d = 1,2,\ldots$$

Radial Basis Functions

$$K(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\sigma^2}\right), \sigma > 0$$

Sigmoid

$$K(\mathbf{x},\mathbf{x}') = \tanh\left(k\mathbf{x}\cdot\mathbf{x}' + \theta\right)$$

Kernel benefits

- Allow SVMs to handle nonlinearly separable data sets
- Incorporate prior knowledge
- Can be defined on inputs that are not vectors (such as strings, graphs...)
- Provide a mathematical formalism for combining different types of data (critical in biological applications).

Kernel dangers

Why not always project the data into a very high-dimensional space, to be sure of finding a separating hyperplane?

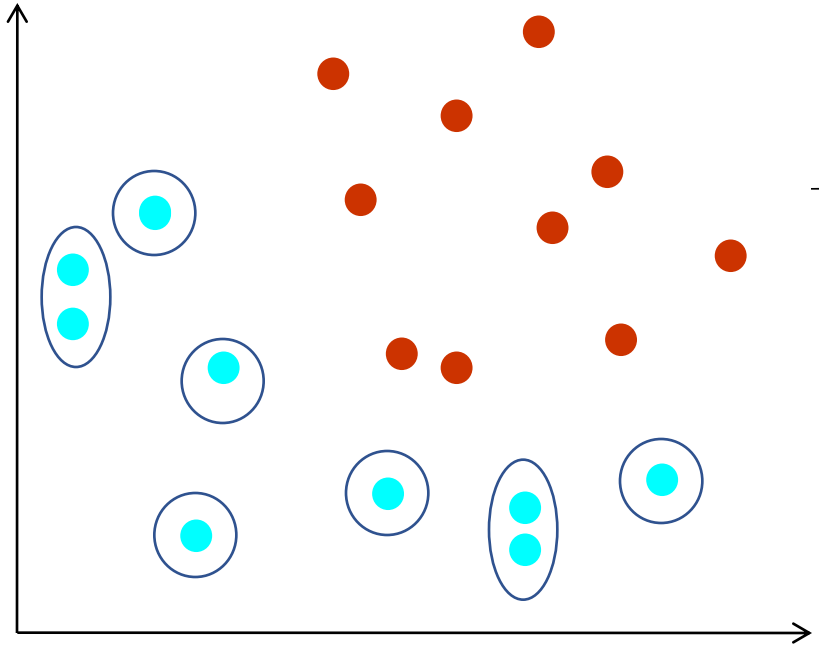
- Projecting into very high-dimensional spaces can be problematic, due to the:

  curse of dimensionality:

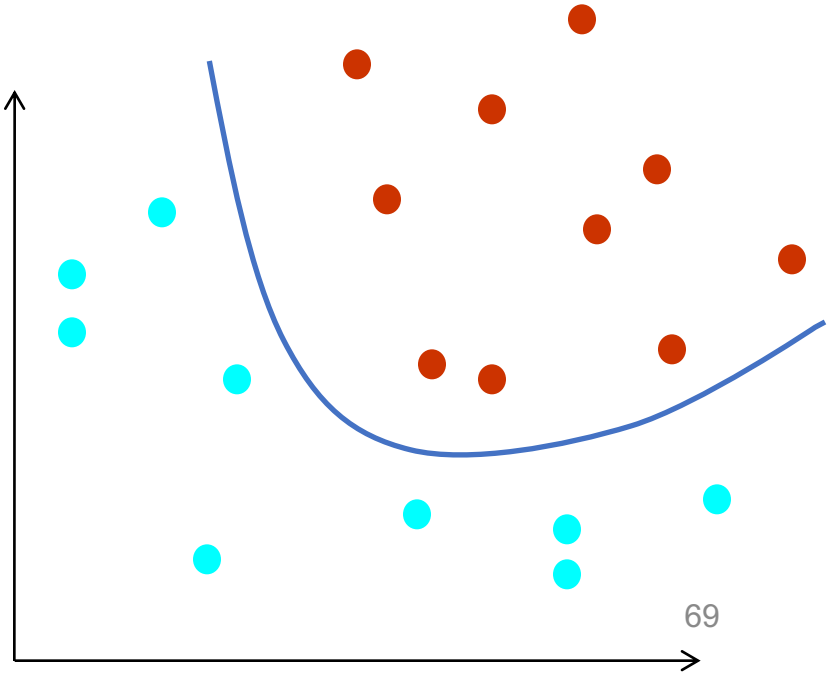
  – As the number of variables under consideration increases, the number of possible solutions also increases, but exponentially.

  – Consequently, it becomes harder for any algorithm to select a correct solution.

Curse of dimensionality

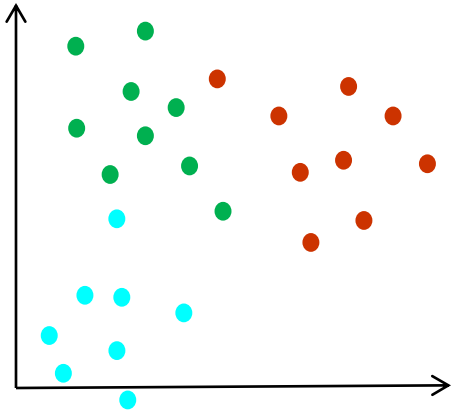


- The SVM is said to have **overfit** the data
- Clearly, such a SVM will not generalize well when presented with new data



Multi-class classification

- World is inherently multi-class.
- Discriminative classifiers are binary-defined by default.

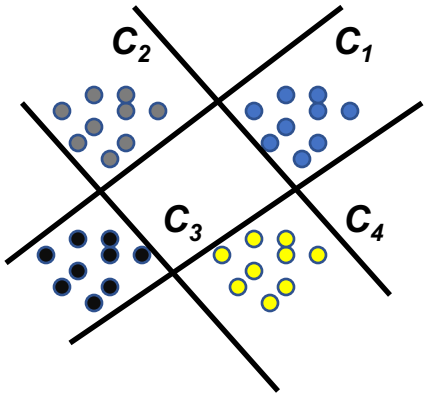


- What if we have data from more than two classes?
- Most common solution: Describe multi-class problem as **combination of binary problems.**

Multi-class classification

- Two common methods to build binary classifiers:

One-versus-all



c_1 vs $c_2 c_3 c_4$

c_2 vs $c_1 c_3 c_4$

c_3 vs $c_1 c_2 c_4$

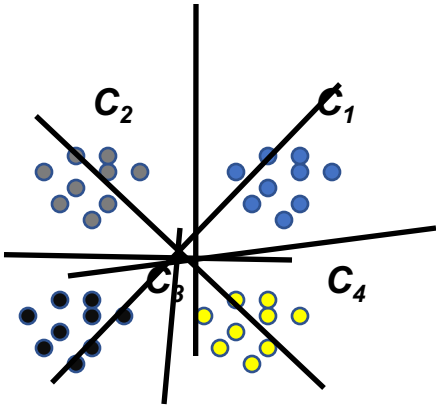
c_4 vs $c_1 c_2 c_3$

One-versus-one (***pairwise***)

c_1 vs c_2 c_2 vs c_3

c_1 vs c_3 c_2 vs c_4

c_1 vs c_4 c_3 vs c_4



Number of one-versus-one classifiers:

$$K = \frac{n(n-1)}{2}$$

Classification of new instances

- One-versus-all:
  - ***Winner-takes-all strategy***: the classifier with the highest output function ($f_{\mathbf{w},b}(\mathbf{x})$) assigns the class (continuous decision values).
  - It is important that output functions are calibrated to produce comparable scores
- One-versus-one:
  - ***Max-wins voting strategy***: every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and the class with most votes determines the instance classification.
- Multi-class SVM often works well in practice!

Algorithm: 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

In practice

- For data sets of thousands of examples, solving the SVM optimization problem is quite **fast**.

- Empirically, running times of state-of-the-art SVM learning algorithms **scale approximately quadratically** (when you give the SVM twice as much data, it requires four times as long to run)**.**

Practical difficulty

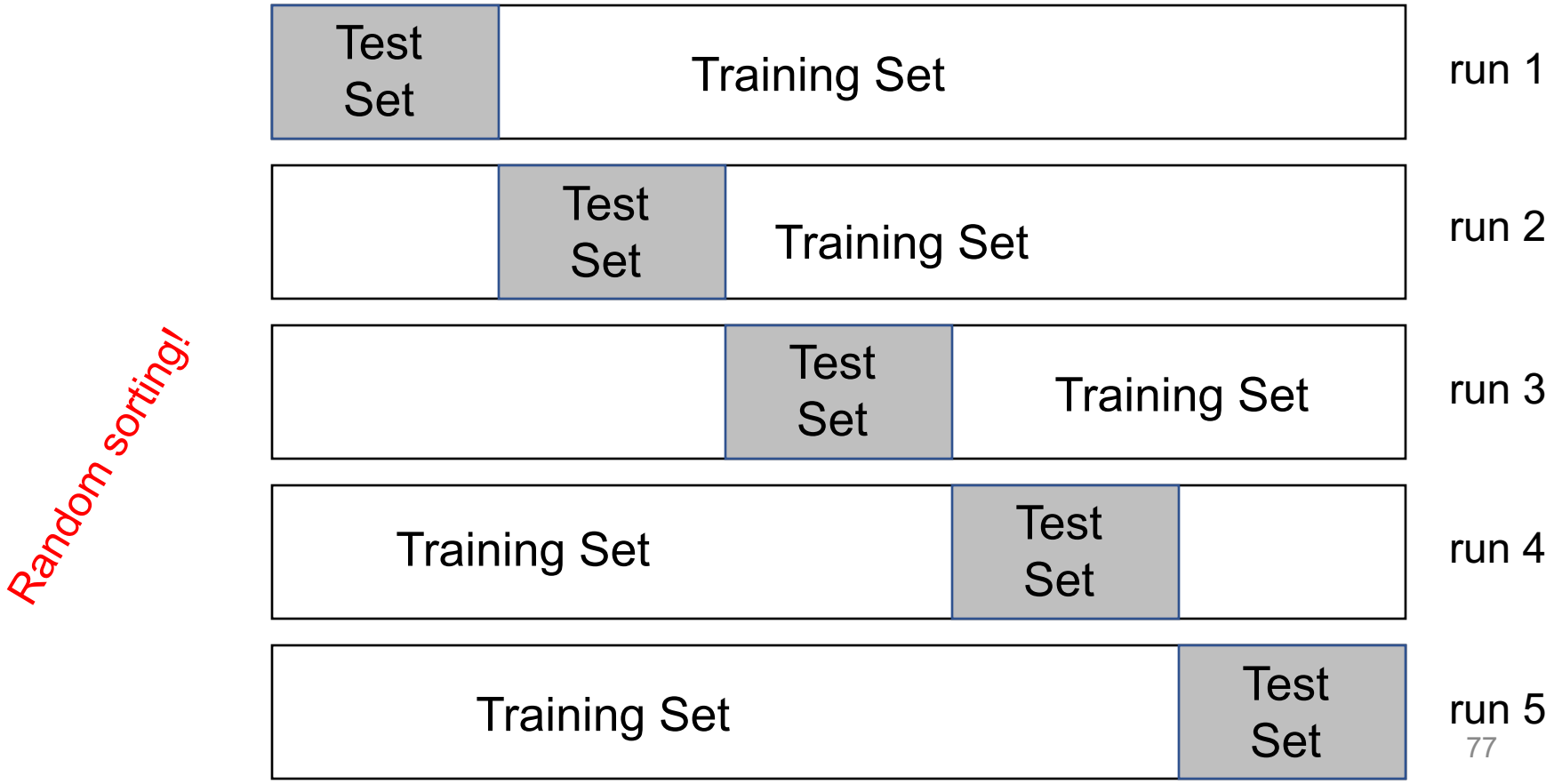
- One would like to use a kernel function that
  - Is likely to allow the data to be separated
  - But without introducing too many irrelevant dimensions.
- How is such a function best chosen?
  - The only realistic answer is **trial and error**.
  - Typically, begin with a simple SVM, and then experiment with a variety of ‘standard’ kernel functions.

Cross-Validation

- Split dataset into two groups:
  - Training set: used to train the classifier
  - Test set: used to estimate the error rate of the trained classifier
- Each split **randomly** selects a % of examples without replacement

Example: K-Fold Cross-validation

K=5. Test set = 20% of the data set



Unranked retrieval evaluation: Precision and Recall

- **Precision**: fraction of retrieved objects that are relevant = P(relevant|retrieved)

- **Recall**: fraction of relevant objects that are retrieved

  = P(retrieved|relevant)

	Relevant	Nonrelevant
Retrieved	tp	fp
Not Retrieved	fn	tn

- Precision P = tp/(tp + fp)
- Recall R = tp/(tp + fn)

Should we instead use the accuracy measure for evaluation?

- Given a query, an engine classifies each object as “Relevant” or “Nonrelevant”
- The **accuracy** of an engine: the fraction of these classifications that are correct
    - (tp + tn) / (tp + fp + fn + tn)
- **Accuracy** is a commonly used evaluation measure in machine learning classification work
- Why is this not always a very useful evaluation measure?

Precision/Recall

- You can get high recall (but low precision) by retrieving all objects for all queries!
- Recall is a non-decreasing function of the number of objects retrieved
- In a good system, precision decreases as either the number of objects retrieved or recall increases
  - This is not a theorem, but a result with strong empirical confirmation

A combined measure: F

- Combined measure that assesses precision/recall tradeoff is **F measure** (weighted harmonic mean):

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- People usually use balanced F_1 measure
  - i.e., with $\beta = 1$ or $\alpha = ½$

Results of the Bags of words for object recognition






class	bag of features	bag of features	Parts-and-shape model
	Zhang et al. (2005)	Willamowski et al. (2004)	Fergus et al. (2003)
airplanes	**98.8**	97.1	90.2
cars (rear)	98.3	**98.6**	90.3
cars (side)	**95.0**	87.3	88.5
faces	**100**	99.3	96.4
motorbikes	**98.5**	98.0	92.5
spotted cats	**97.0**	—	90.0

Implementation of the object recognition by Bag-of-Words in VL-Feat.

Advantages and disadvantages of SVM

- The **advantages** of support vector machines are:
  - Effective in high dimensional spaces.
  - Still effective in cases where number of dimensions is greater than the number of samples.
  - Use a subset of training points in the decision function (called support vectors), so it is also memory efficient.
  - Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.
  - 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

http://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation

Advantages and disadvantages of SVM

The **disadvantages** of support vector machines include:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using a five-fold cross-validation.
- 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

http://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation

Laboratory

Use BoW with SVM for object recognition on a small dataset (Caltech101)

http://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation

References

- Some slides borrowed from:
- Li Fei-Fei and A. Vedaldi
- Josef Sivic

Lecture videos

SVM + BoW

- From 773 – Intro. Until 800 – End.

From *Introduction to Computer Vision*:

https://www.udacity.com/course/introduction-to-computer-vision--ud810

COMPUTATIONAL VISION: Image Features (SIFT)

Master in Artificial Intelligence

Department of Mathematics and Computer Science

UNIVERSITAT DE BARCELONA