Karlsruhe Institute of Technology

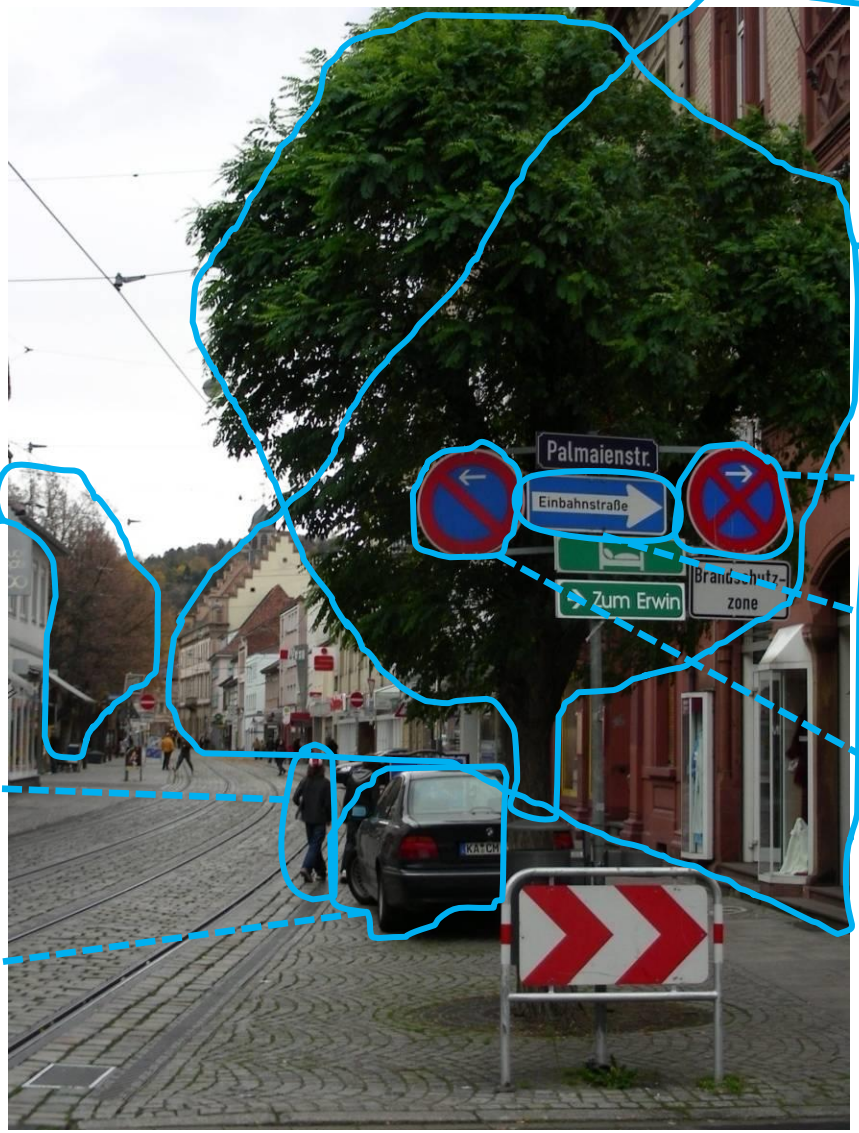
Machine Vision

Chapter 10: Pattern Recognition

Dr. Martin Lauer

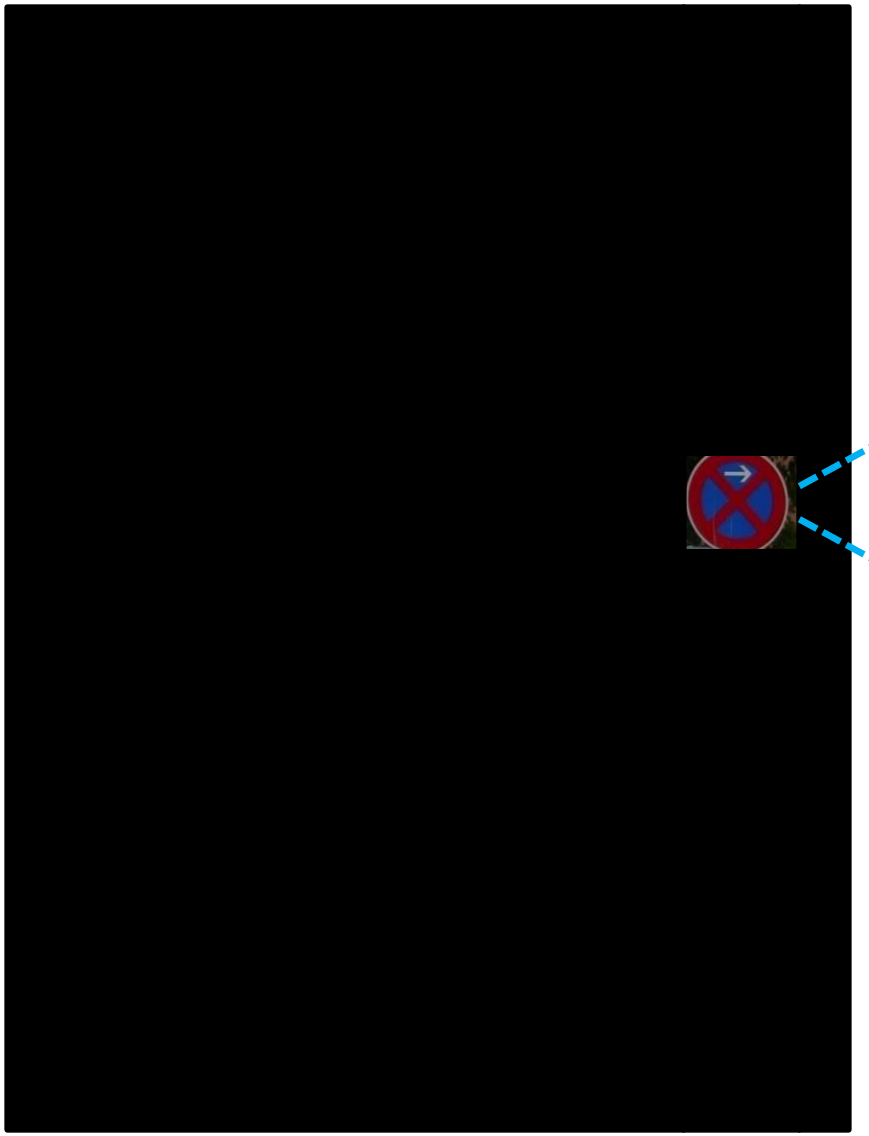
Institut für Mess- und Regelungstechnik

Classification



Classification: assigning objects to categories (classes)

Classification



traffic sign

traffic sign
“no parking”

in machine vision:

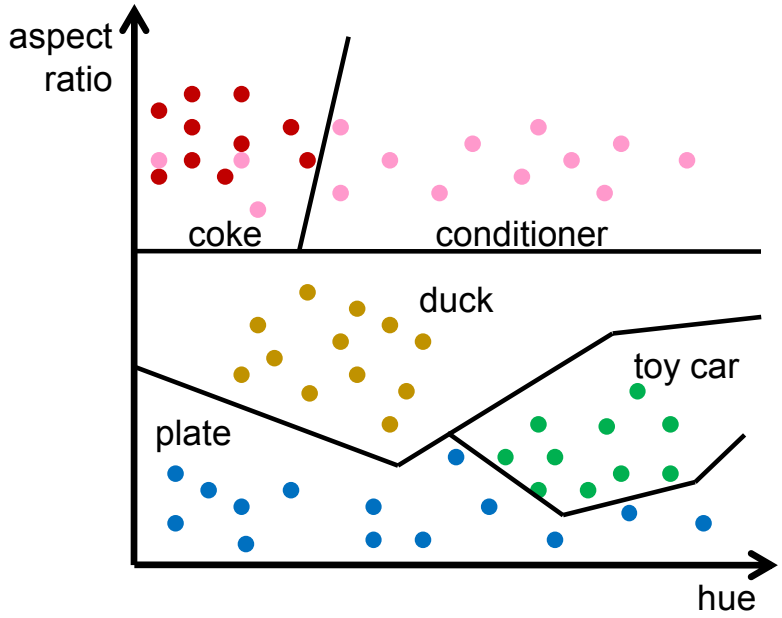
- extract the relevant object from the background (segmentation)
- assign object to a category (classification)
- both steps might depend on each other

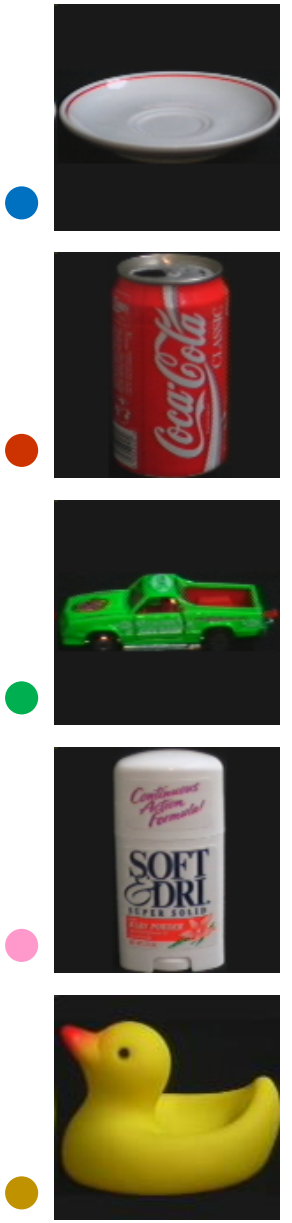
10. 模型比较

- **何时选择 HOG 特征优于 LBP 特征?**
  - HOG 适合对纹理和形状敏感的任务 (如人体检测)。
  - LBP 适合对局部模式变化敏感的任务 (如纹理分类)。
- **如何在提高模型精度的同时控制计算复杂度?**
  - 简化特征提取 (如减少分辨率)。
  - 使用轻量化模型 (如剪枝、蒸馏)。
  - 优化推理过程 (如量化、加速算法)。

Classification cont.

- how can we distinguish these objects?
  - geometric features like aspect ratio, roundness, …
  - color features like dominant hue, average saturation, variance of color, …
  - …
- from a sample of images we get:

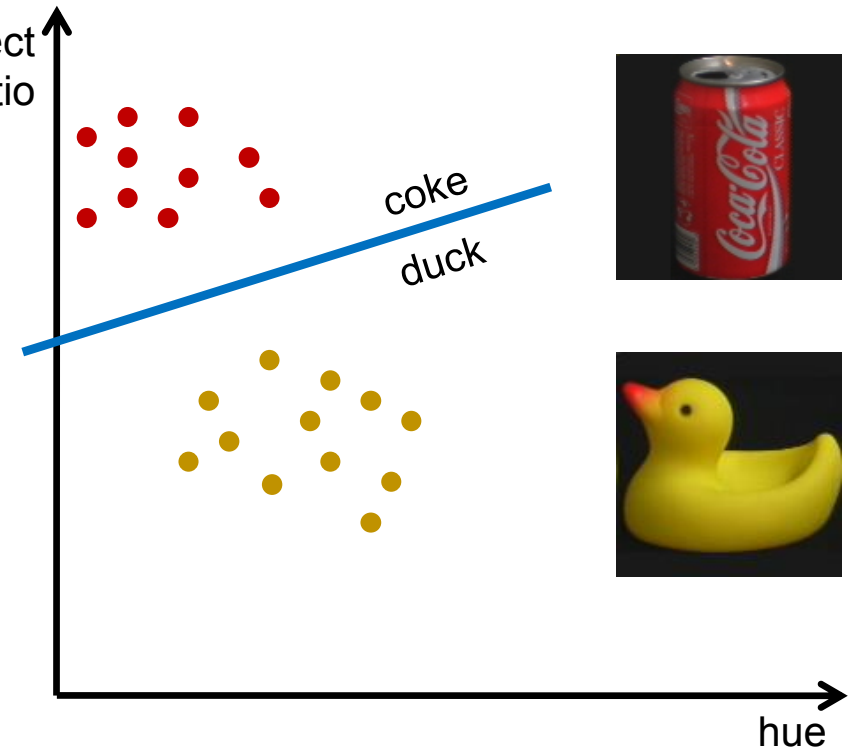




images source: COIL-100 database, http://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php

Classification

- learning from examples
  - collecting images of objects
  - creating a feature vector for each object (“*pattern*”)
  - find a decision rule that distinguishes between feature vectors of different classes
  - the process of creating a decision rule from example patterns is called “*learning*” or “*training*”

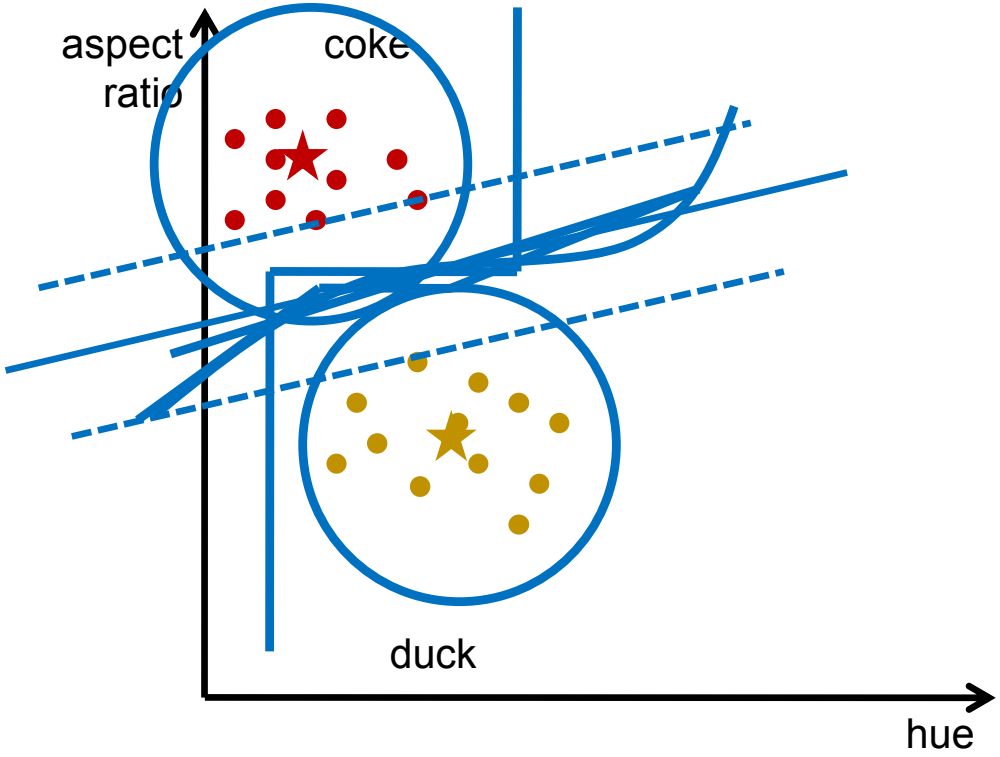


SVM适合处理高维数据，尤其在样本数量少但特征维度高的情况下。

- 决策树适合处理类别间非线性边界，且易于解释，但可能存在过拟合风险。

Classification cont.

- many approaches for decision rules and learning
  - linear classification
  - artificial neural networks/ deep learning
  - prototype-based methods
  - case based reasoning
  - decision trees
  - support vector machines
  - boosting (meta algorithm)
  - …



- in this lecture:

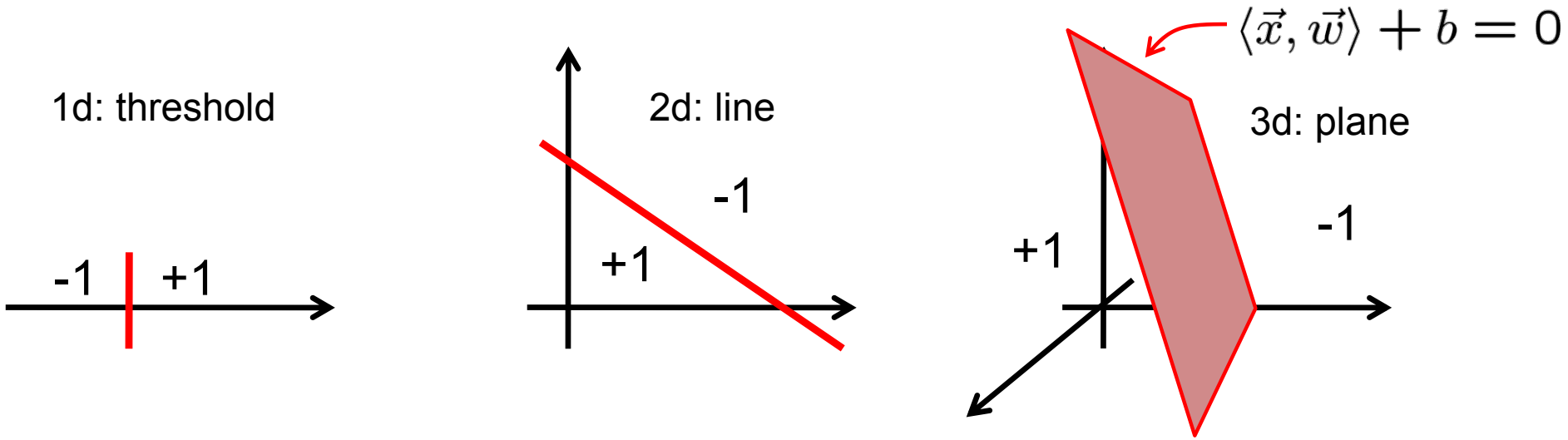
  linear classification, support vector machines, boosting, decision trees, deep learning

Linear Classification

- a linear classifier is a function of the kind:

$$\vec{x} \mapsto \begin{cases} +1 & \text{if } \langle \vec{x}, \vec{w} \rangle + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

  - $\vec{w}$ is the *weight vector* of the linear classifier
  - b is the *bias weight* of the classifier
- a linear classifier subdivides an input space into two half spaces. The decision boundary is a hyperplane



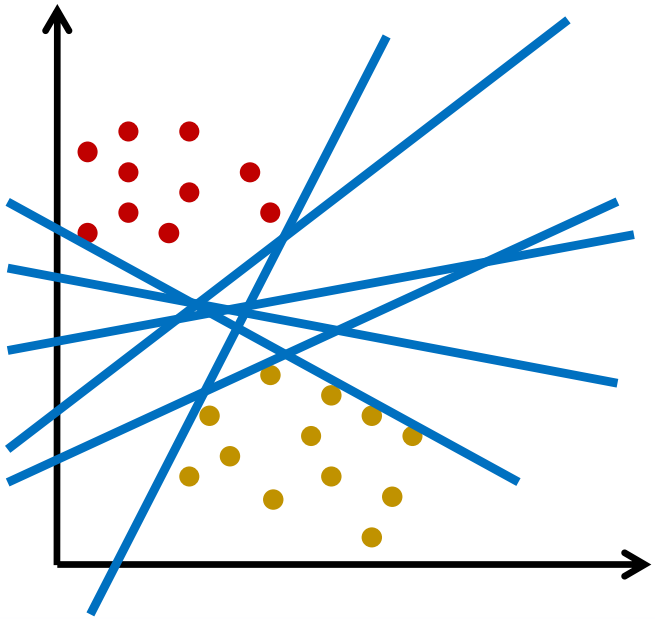
Linear Classification cont.

- learning task:
  - given a set of training examples $\{(\vec{x}^{(1)}, d^{(1)}), \ldots, (\vec{x}^{(p)}, d^{(p)})\}$

    $d^{(i)} = +1$ for examples belonging to the one class (“positive examples”)

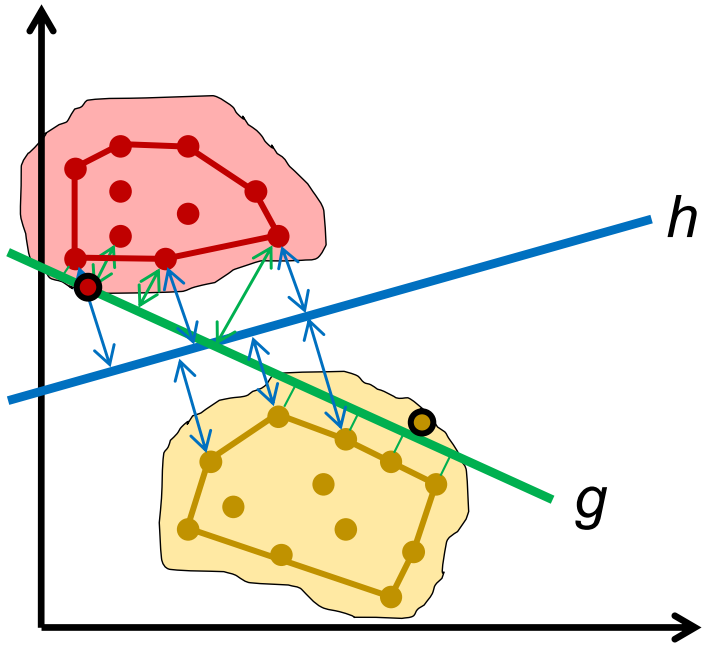
    $d^{(i)} = -1$ for examples belonging to the other class (“negative examples”)
  - find $\vec{w}$ and b such that:

    $$d^{(i)} \cdot (\langle \vec{x}^{(i)}, \vec{w} \rangle + b) > 0 \qquad \text{for all } i \in \{1, \ldots, p\}$$
  - many possible solutions, which one is the best?



Linear Classification cont.

- many possible solutions, which one is the best?
  - *g* and *h*, both don't make classification errors
  - *g* has shorter distance to patterns than *h*
  - risk of misclassification for new pattern is larger for *g* than for *h*
  - (unknown) support of the class probability distributions is similar to convex hull of training examples



Maximising the distance of the separating hyperplane to the convex hull of the training patterns means minimising the risk of misclassification (result from computational learning theory)

Linear Classification cont.

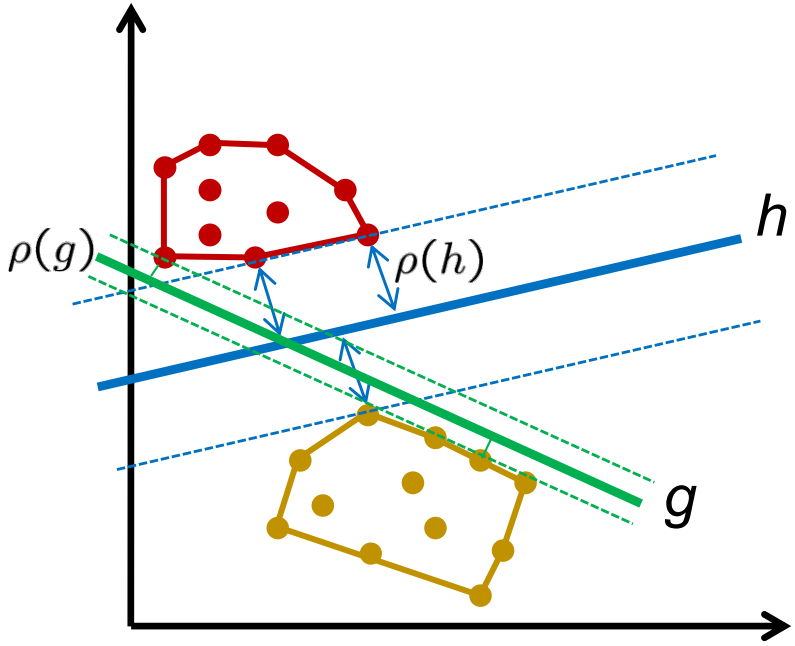
- margin:

  the minimal distance between a hyperplane and the convex hull of the training patterns

$$\rho = \min_i \left(d^{(i)} \cdot \frac{\langle \vec{x}^{(i)}, \vec{w} \rangle + b}{||\vec{w}||} \right)$$

- support vector machine (SVM):

  linear classifier that maximises the margin



SVM

- support vector machine (SVM):

  training a SVM means solving:

$$\underset{\rho,\vec{w},b}{maximise} \quad \rho^2$$

$$subject\ to \quad d^{(i)} \cdot \frac{\langle \vec{x}^{(i)}, \vec{w} \rangle + b}{||\vec{w}||} \geq \rho \qquad \text{for all } i$$

$$\rho > 0$$

  – one degree of freedom: $||\vec{w}||$

  – for convenience set $||\vec{w}|| = \frac{1}{\rho}$ 转换为最小化问题

$$\underset{\vec{w},b}{minimise} \quad \frac{1}{2}||\vec{w}||^2$$

$$subject\ to \quad d^{(i)} \cdot (\langle \vec{x}^{(i)}, \vec{w} \rangle + b) \geq 1 \qquad \text{for all } i$$

SVM cont.

- a simple example:
  - patterns are 1D:
    positive: 5, 10
    negative: -1, 2
  - parameters: w_1, b

为了保证目标函数光滑可导

  - optimization problem:

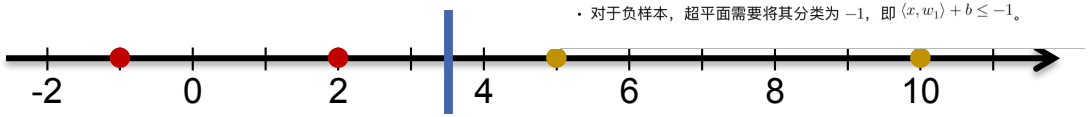
$$\underset{w_1,b}{minimise} \quad \frac{1}{2}w_1^2$$

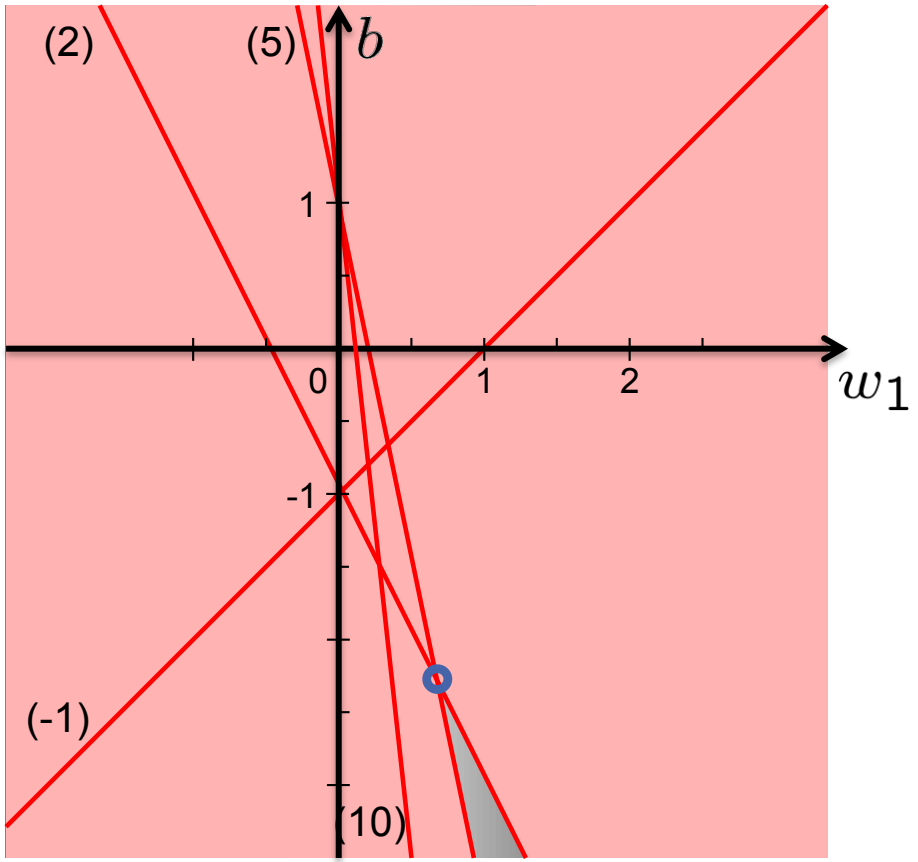
$$subject\ to \quad b \geq 1 - 5w_1$$
$$b \geq 1 - 10w_1$$
$$b \leq -1 + w_1$$
$$b \leq -1 - 2w_1$$

约束条件：

分类条件需要正负样本点正确分类，定义如下：

- **正样本：** $x = 5, 10$
  $b \geq 1 - 5w_1 \qquad b \geq 1 - 10w_1$
- **负样本：** $x = -1, 2$
  $b \leq -1 + w_1 \qquad b \leq -1 - 2w_1$
- **解释：**
  - 对于正样本，超平面需要将其分类为 $+1$，即 $\langle x, w_1 \rangle + b \geq 1$。
  - 对于负样本，超平面需要将其分类为 -1，即 $\langle x, w_1 \rangle + b \leq -1$。





SVM cont.

- how to train a SVM?

$$\underset{\vec{w},b}{minimise} \quad \frac{1}{2}||\vec{w}||^2$$

$$subject\ to \quad d^{(i)} \cdot (\langle \vec{x}^{(i)}, \vec{w} \rangle + b) \geq 1 \qquad \text{for all } i$$

- … skipping all details …
  - theory of Lagrange multipliers applies
  - one Lagrange multiplier α_i per training pattern $\vec{x}^{(i)}$
  - the solution is completely described by the Lagrange multipliers
  - many Lagrange multipliers are zero
  - algorithms exist to calculate the Lagrange multipliers

SVM cont.

- the solution:

$$\vec{w} = \sum_i \alpha_i d^{(i)} \vec{x}^{(i)}$$

only support vectors ($\alpha_i \neq 0$) contribute

$$b = d^{(j)} - \langle \vec{x}^{(j)}, \vec{w} \rangle = d^{(j)} - \sum_i \alpha_i d^{(i)} \langle \vec{x}^{(j)}, \vec{x}^{(i)} \rangle$$

for j with $\alpha_j \neq 0$

- margin: -> 最大間隔距離

$$\rho = \frac{1}{||\vec{w}||} = \frac{1}{\sqrt{\sum_i \sum_j \alpha_i \alpha_j d^{(i)} d^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle}}$$

can be expressed without using $\vec{w}$

- classifying a new pattern $\vec{x}_{new}$:

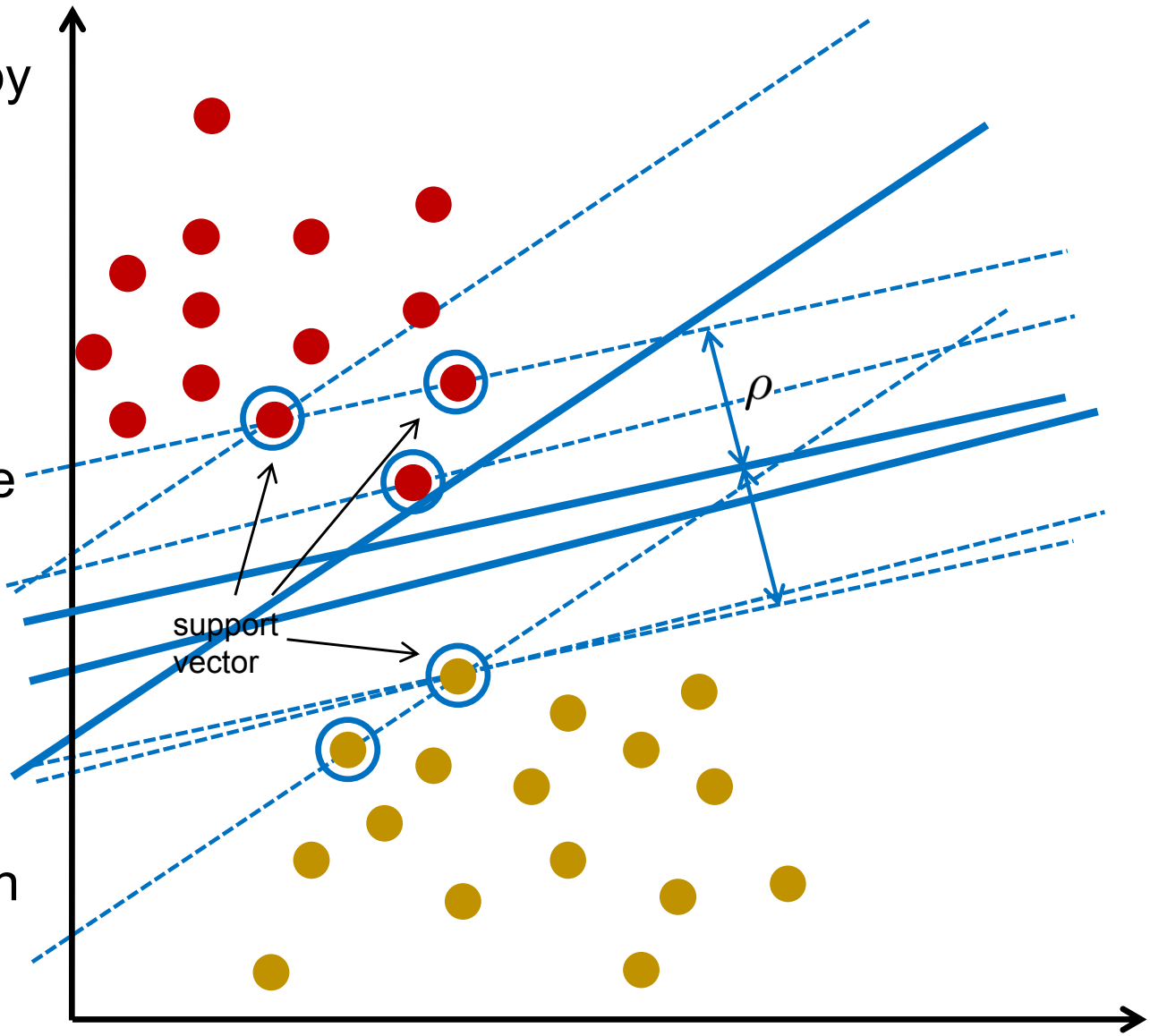
$$sign\Big(\langle \vec{x}_{new}, \vec{w} \rangle + b\Big)$$

$$= sign\Big(\sum_i \alpha_i d^{(i)} \langle \vec{x}_{new}, \vec{x}^{(i)} \rangle + b\Big)$$

patterns $\vec{x}$ occur only inside dot products

SVM cont.

- optimal separating hyperplane determined by support vectors
- removing non-support vectors does not change solution
- adding patterns with distance of more than the margin does not change solution
- removing support vector changes solution
- adding pattern with distance less than margin changes solution

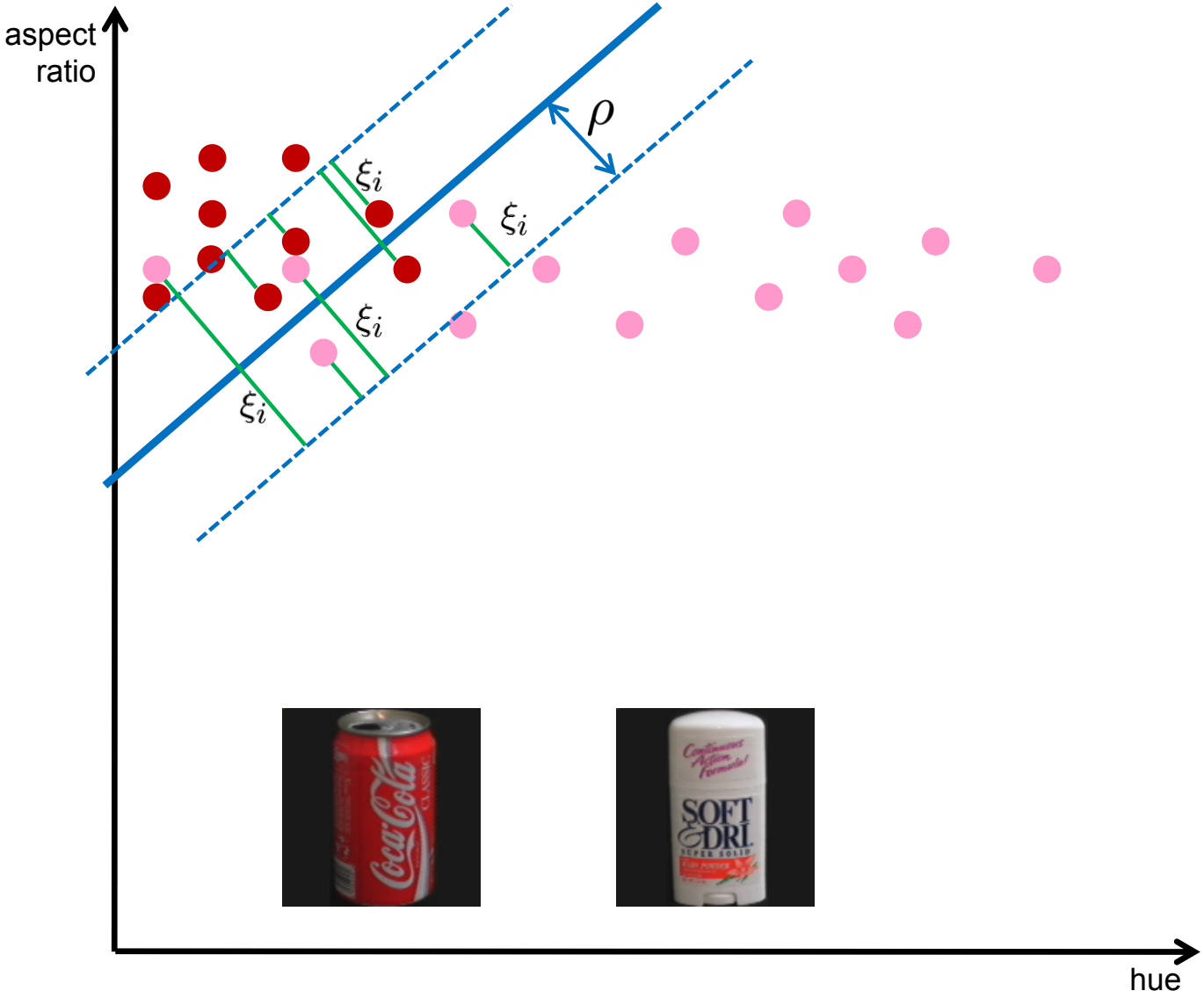


Fault-tolerant SVMs

- overlapping classes force to make errors
- individual errors ξ_i
- conflicting targets:

  maximise ρ

  minimise ξ_i



Fault-tolerant SVMs cont.

- optimization problem:

$$\underset{\vec{w},b,\xi_1,\ldots\xi_p}{minimise} \quad \frac{1}{2}||\vec{w}||^2 + C\sum_i \xi_i$$

$$subject\ to \quad d^{(i)} \cdot (\langle \vec{x}^{(i)}, \vec{w} \rangle + b) \geq 1 - \xi_i \qquad \text{for all } i$$

$$\xi_i \geq 0 \qquad \text{for all } i$$

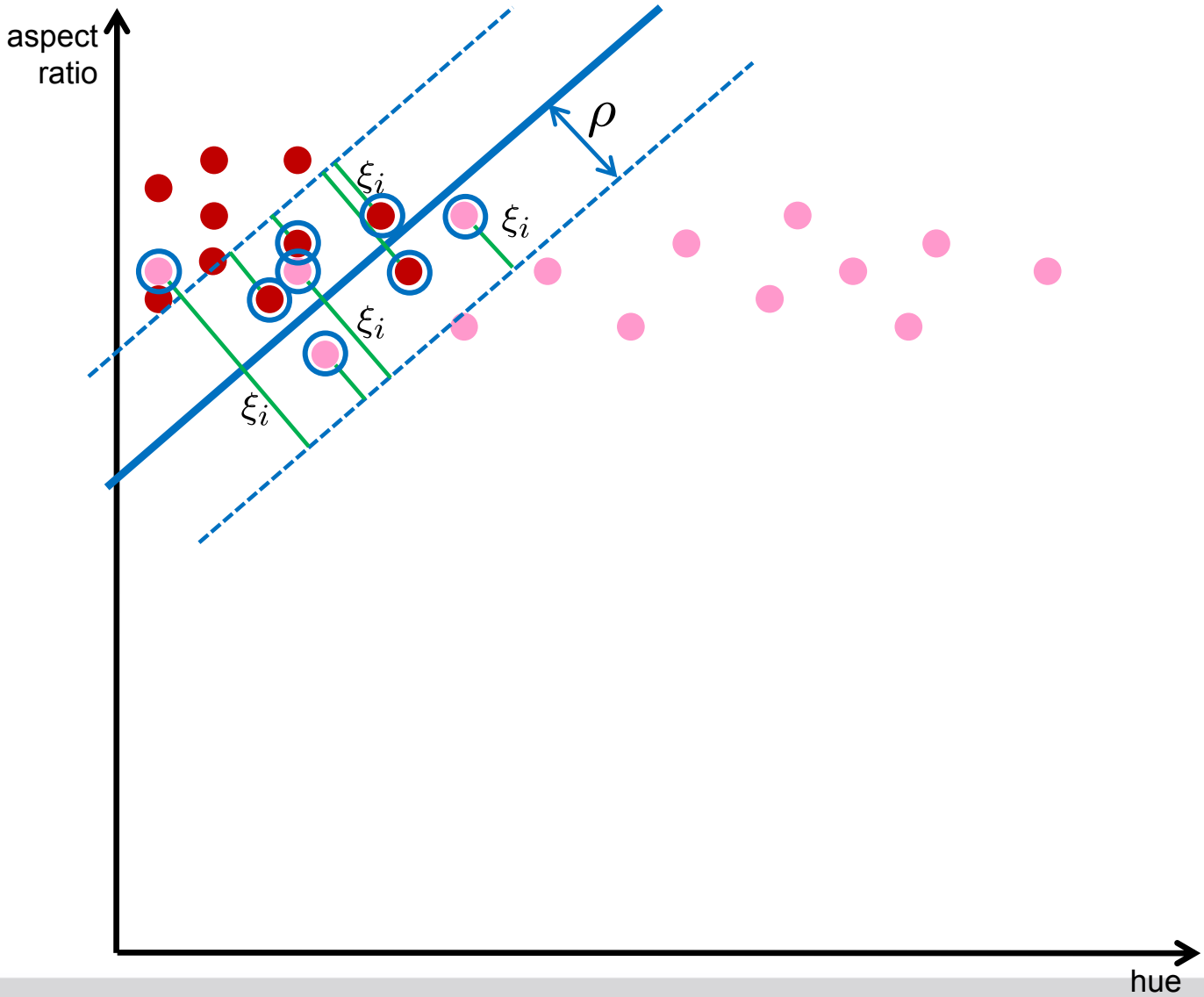
penalize errors

allow errors

- *C>0*: regularization parameter controls balance between small errors and large margin (has to be chosen manually)
- fault-tolerant SVMs are known as *"soft-margin-SVMs"* (in contrast to *"hard-margin-SVMs"*)

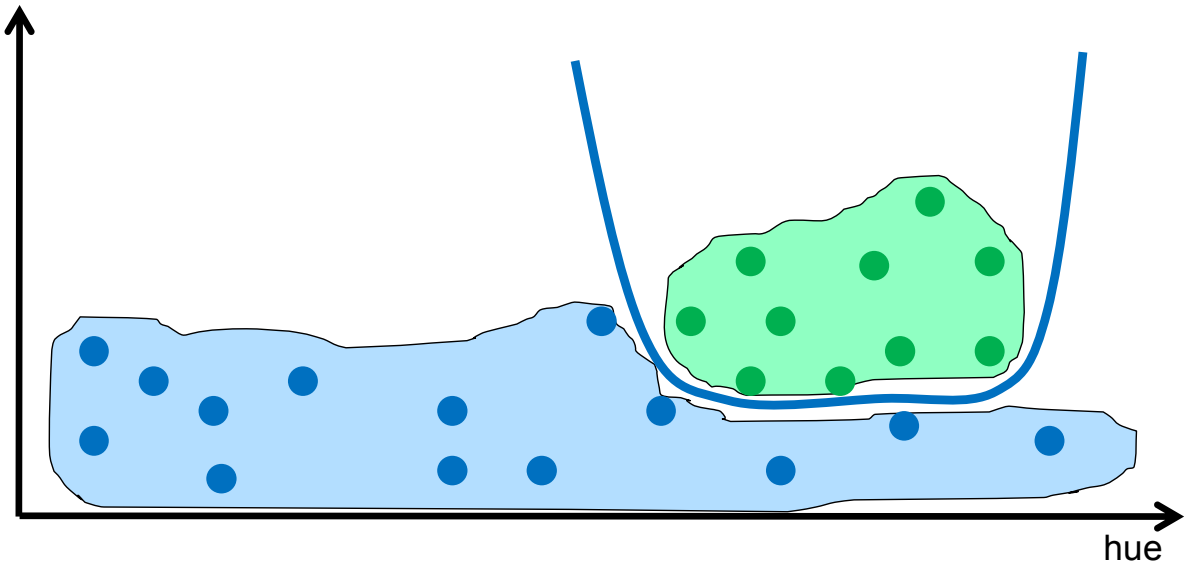
Fault-tolerant SVMs

- similar solution as in the hard-margin case
- support vectors are all patterns that create an individual error or which are on the boundary of the margin area



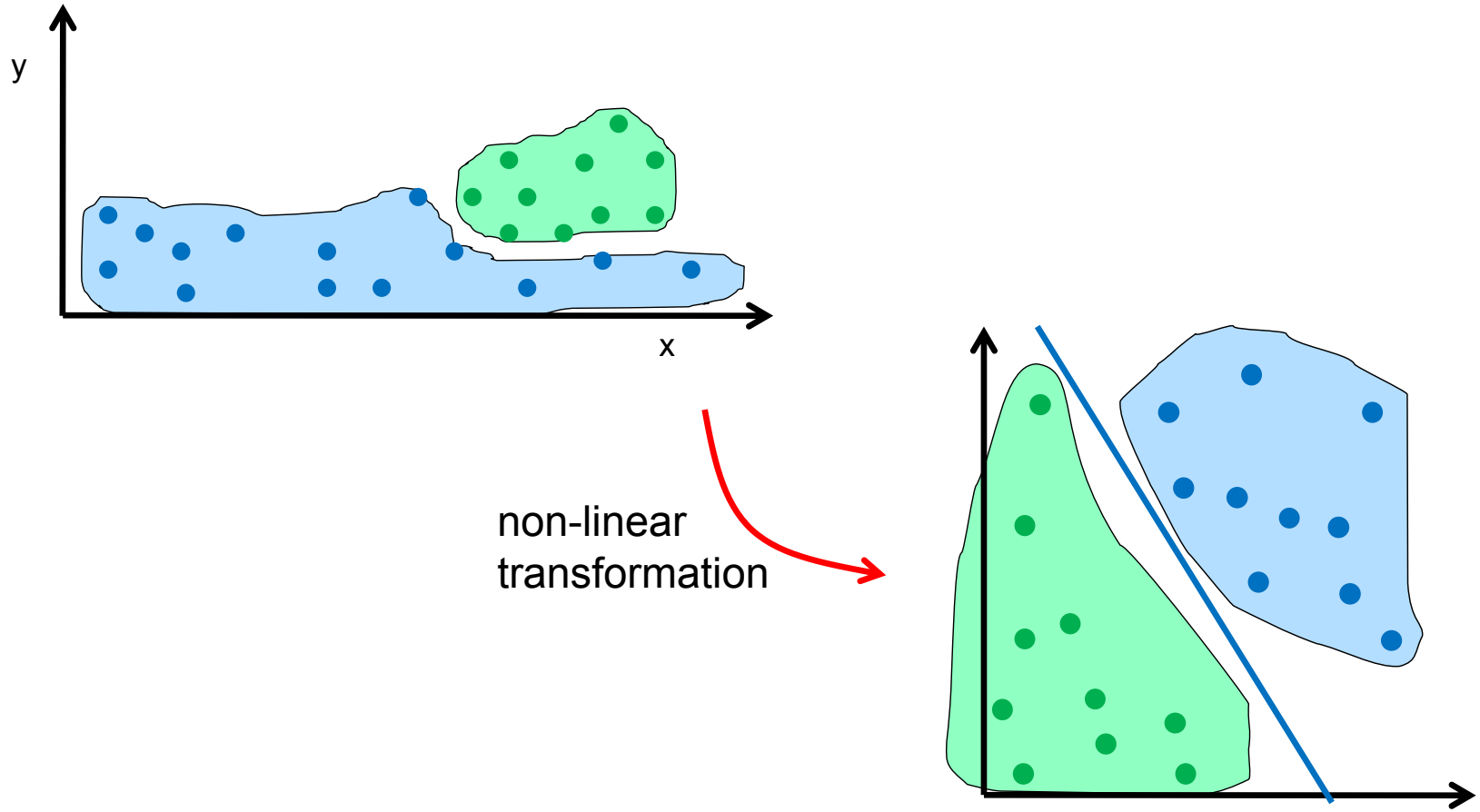
Nonlinear SVMs

- classes with non-overlapping support might not be linearly separable → non-linear classifier
- direct way: use circle/ellipse/non linear curve for classification → difficult to analyse
- indirect way: transform data non-linearly and classify transformed data instead



Nonlinear SVMs cont.

- a non-linear problem might become linear after non-linear transformation



Nonlinear SVMs cont.

- assume nonlinear transformation

$$\Phi : \begin{cases} \mathbb{R}^n \rightarrow \mathbb{R}^m \\ \vec{x} \mapsto \Phi(\vec{x}) = \vec{X} \end{cases}$$

- find SVM that solves:

$$\underset{\vec{W},b}{minimise} \ \frac{1}{2}||\vec{W}||^2$$

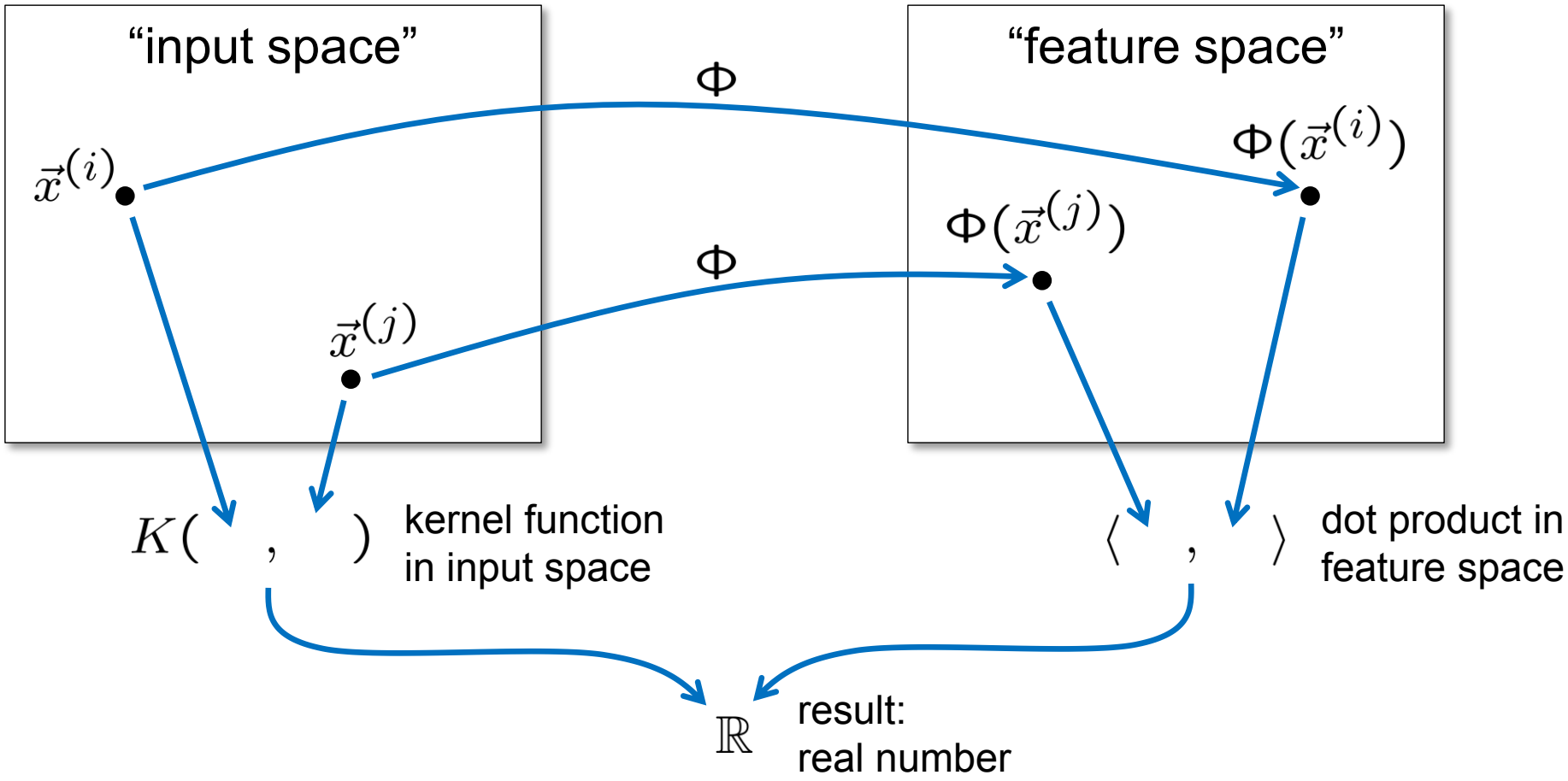
$$subject\ to \ \ d^{(i)} \cdot (\langle \vec{X}^{(i)}, \vec{W} \rangle + b) \geq 1 \qquad \text{for all } i$$

- solution is completely determined knowing the Lagrange multipliers *(cf. slide 18)*
  - *don't need do calculate* $\vec{W}$
  - *patterns only occur pairwise as arguments of dot products*

$$\langle \vec{X}^{(i)}, \vec{X}^{(j)} \rangle = \langle \Phi(\vec{x}^{(i)}), \Phi(\vec{x}^{(j)}) \rangle$$

Nonlinear SVMs cont.

- considering $\langle \Phi(\vec{x}^{(i)}), \Phi(\vec{x}^{(j)}) \rangle$



- shortcut: kernel-function $K(\vec{x}, \vec{y}) = \langle \Phi(\vec{x}), \Phi(\vec{y}) \rangle$
- substituting $\langle \Phi(\vec{x}), \Phi(\vec{y}) \rangle$ by $K(\vec{x}, \vec{y})$ eliminates Φ

Kernel Function

- kernel-function: a nasty trick to hide the complexity?
  - example:

$$\Phi(x) = \begin{pmatrix} x^2 \\ x \end{pmatrix}$$

    - evaluating Φ(x) and Φ(y) needs 2 multiplications
    - evaluating dot product in feature space needs 2 multiplications and 1 addition, in total: 4 multiplications, 1 addition

$$K(x, y) = \langle \Phi(x), \Phi(y) \rangle = (xy)^2 + (xy)$$

    - evaluating the kernel function needs 2 multiplications and 1 addition

  - some kernels are based on Hilbert spaces with infinite dimension

Kernel Function cont.

- useful kernel-functions:
  - dot product

$$K(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle$$

  - polynomial kernels

$$K(\vec{x}, \vec{y}) = (\langle \vec{x}, \vec{y} \rangle)^q \text{ or } (\langle \vec{x}, \vec{y} \rangle + 1)^q$$

  - radial basis function (RBF) kernels

$$K(\vec{x}, \vec{y}) = e^{-\frac{||\vec{x}-\vec{y}||^2}{2\sigma^2}}$$

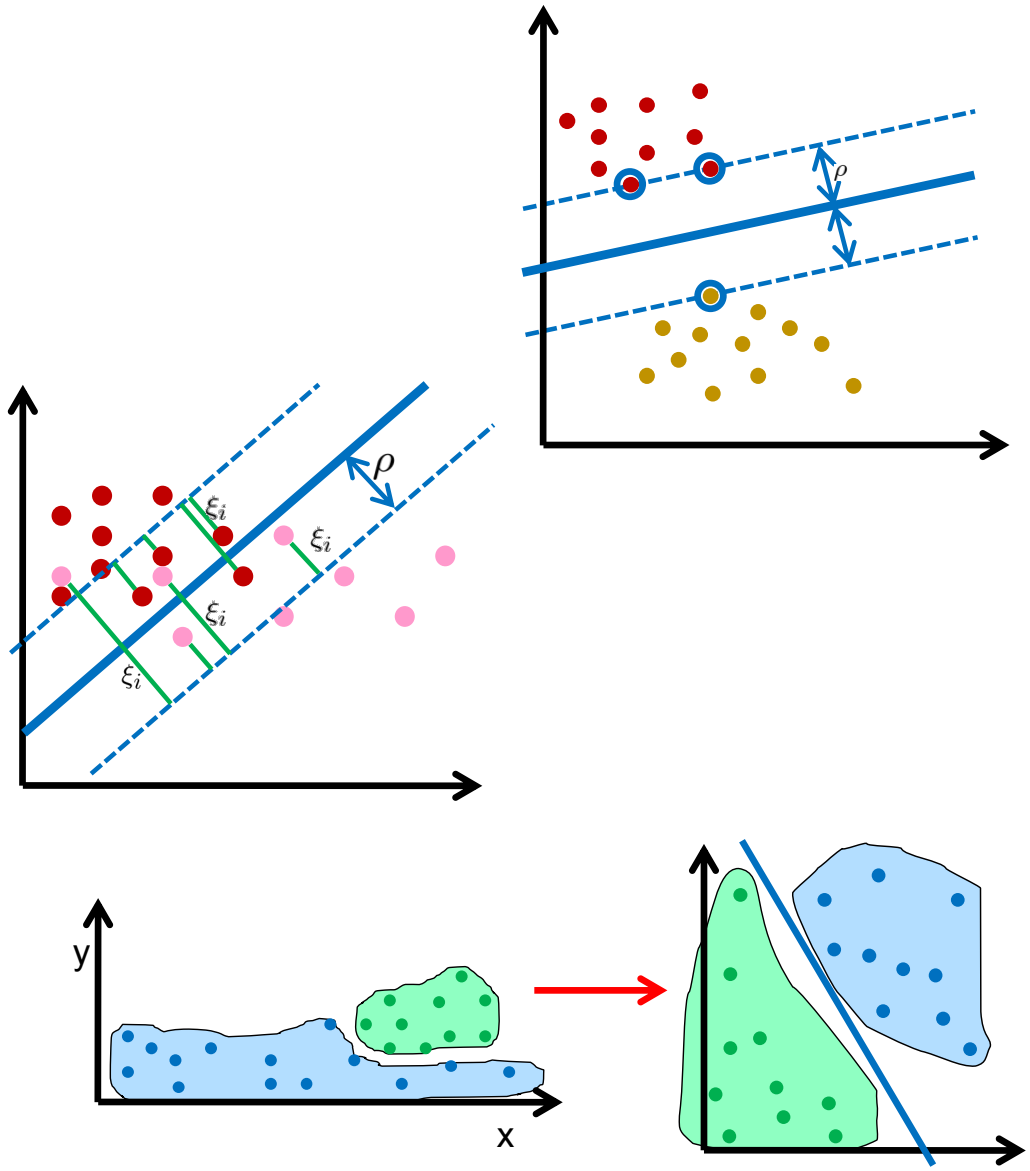
  - Histogram intersection kernel (only for histogram features)

$$K(\vec{x}, \vec{y}) = \sum_i \min\{x_i, y_i\}$$

kernel parameters have to be set manually

SVMs cont.

- combining all ideas:
  - SVMs maximise the margin to minimise the risk of misclassification
  - soft-margin SVM allow individual errors. Balance between margin size and errors controlled by parameter *C*
  - kernel functions allow non-linear classification without changing the theoretical framework. Kernel type and kernel parameters control the degree of non-linearity

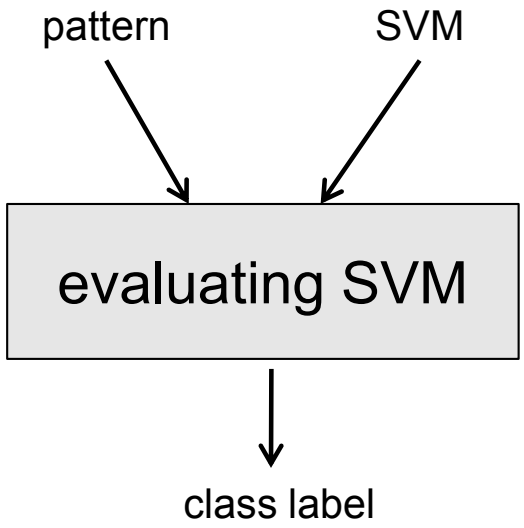


SVMs cont.

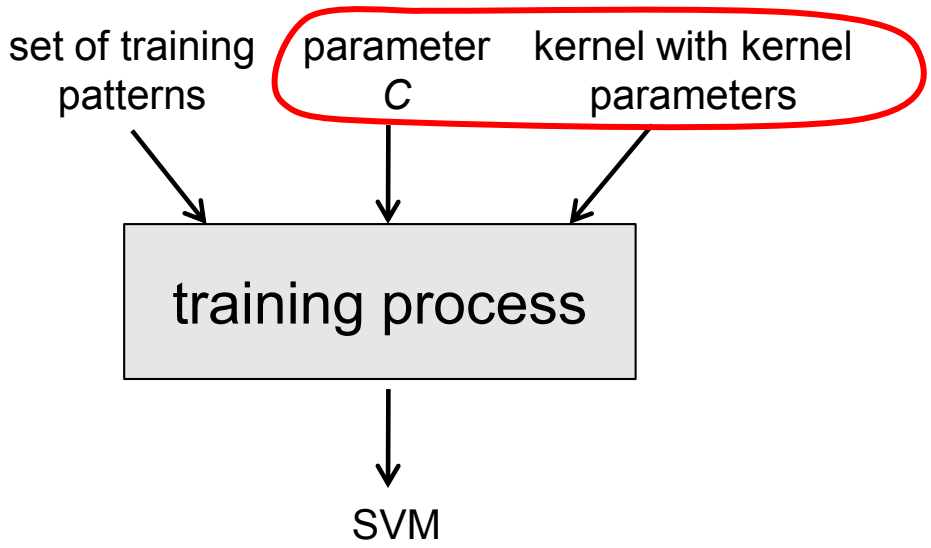
- toy demo

Working with SVMs

- applying a SVM:

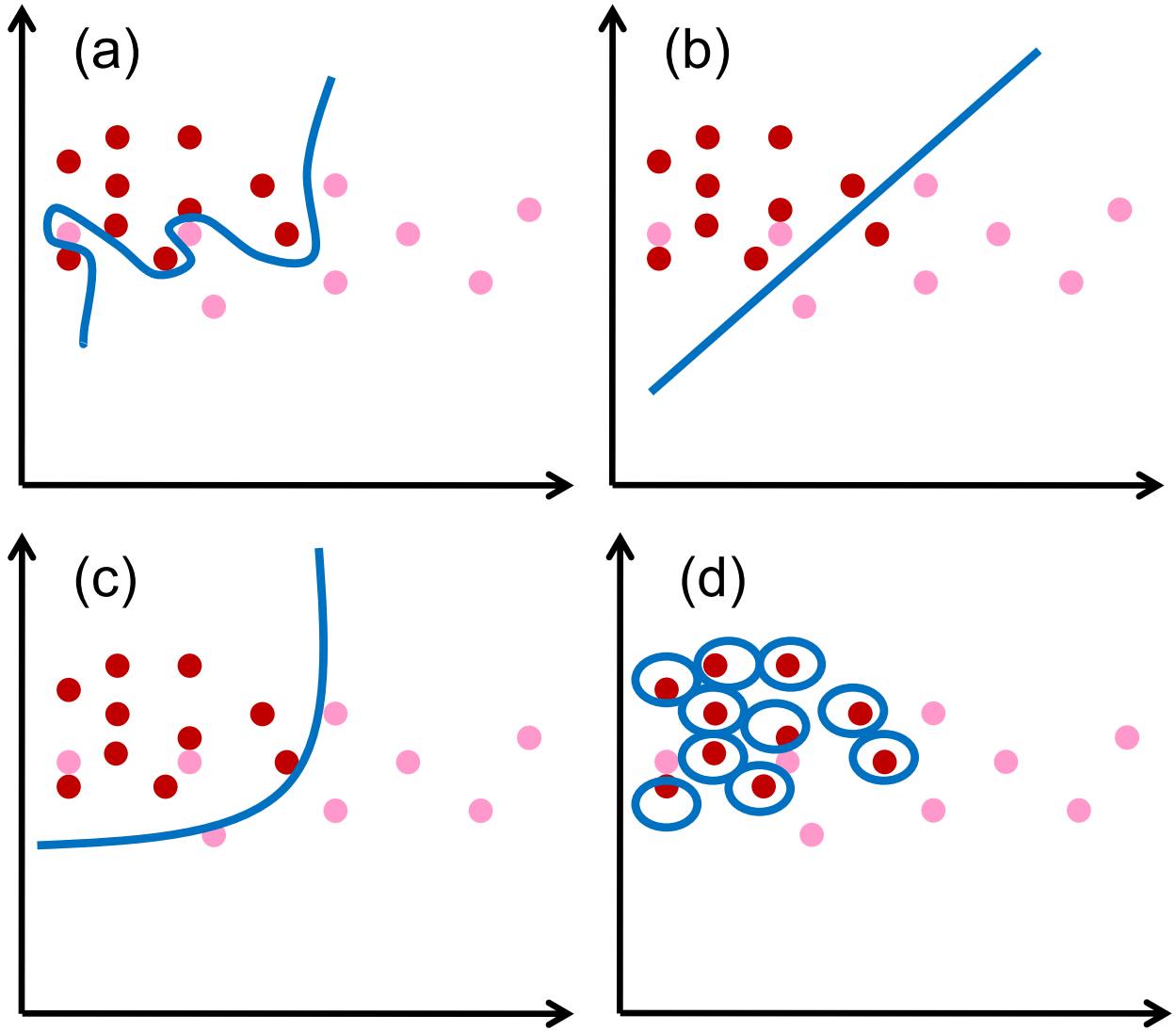


- training a SVM:
  - how can we determine *C* and kernel?



Validation Process

- which SVM is better?



Validation Process cont.

- expected risk of misclassification:

risk of misclassification = risk of "*false negative*" + risk of "*false positive*"

$$E = \int_{A_-} P(+) \cdot p_+(x)dx + \int_{A_+} P(-) \cdot p_-(x)dx$$

negative and positive halfspace given by SVM

prior probabilities of positive and negative class

distribution (pdf) of positive and negative class

概率密度分布

- *E* unknown, but can be approximated from a sample set

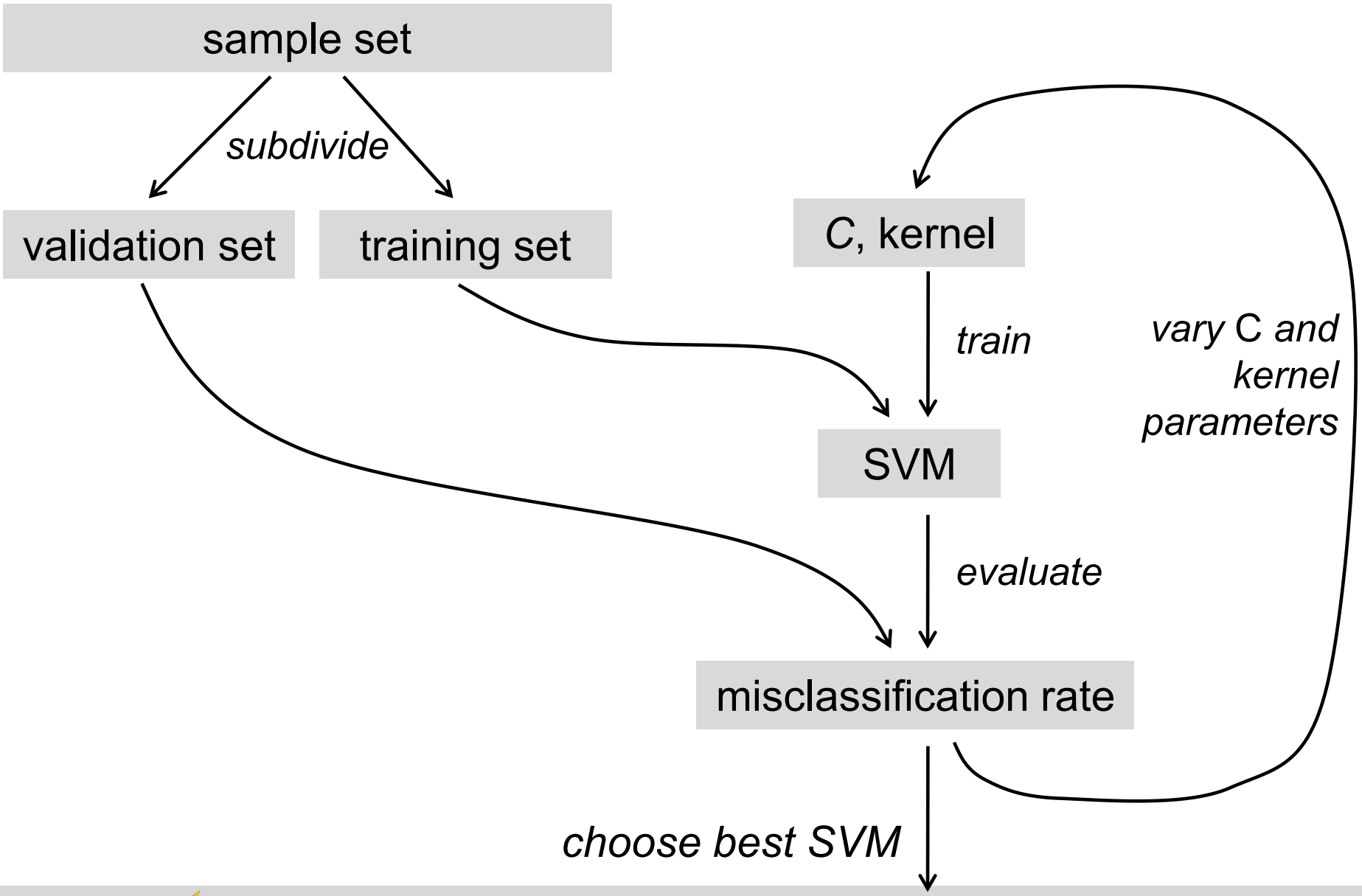
$$E \approx \frac{n_{fn} + n_{fp}}{n}$$

  - n number of elements in sample set
  - n_{fp} number of false positives in sample set
  - n_{fn} number of false negatives in sample set

Validation Process cont.

- validation is a process to test the performance of a classifier on a sample set (“test set”, “validation set”)
- choose the SVM with the smallest misclassification rate on the test set
- test set must be independent of training set!
- validation allows to compare the performance of SVMs trained with different values for *C* and different kernels

Validation Process cont.



Cross-Validation

- disadvantage of validation process:
  - only a part of the data are used for training
  - only a part of the data are used for validation
- *k*-fold cross-validation
  - idea: repeat the training/validation process several times with different training and validation sets
  - *k* is the number of repetitions (between 2 and number of patterns)

Cross-Validation cont.

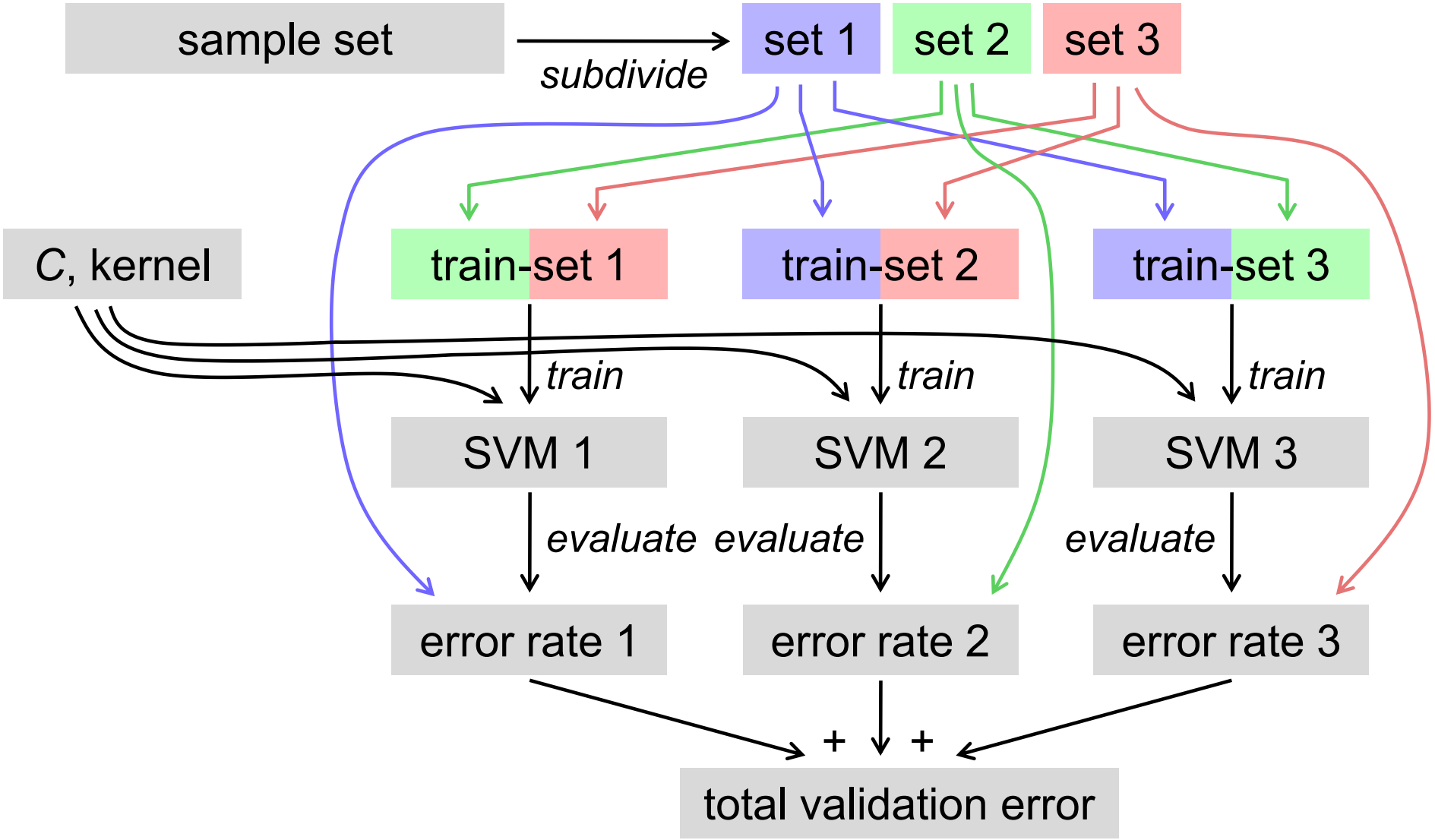
- *k*-fold cross-validation
  1. subdivide pattern set into *k* disjoint subsets of equal size
  2. repeat for every subset *j*:
     2.1. train SVM from subsets *1,…,j-1,j+1,…k*
     2.2. evaluate misclassification rate on subset *j*
  3. average misclassification rates

- advantage:
  - all patterns are used for validation
  - training set contains a rate of $\frac{k-1}{k}$ patterns

- if *k* is equal to the total number of patterns
  → *leave-one-out-error*

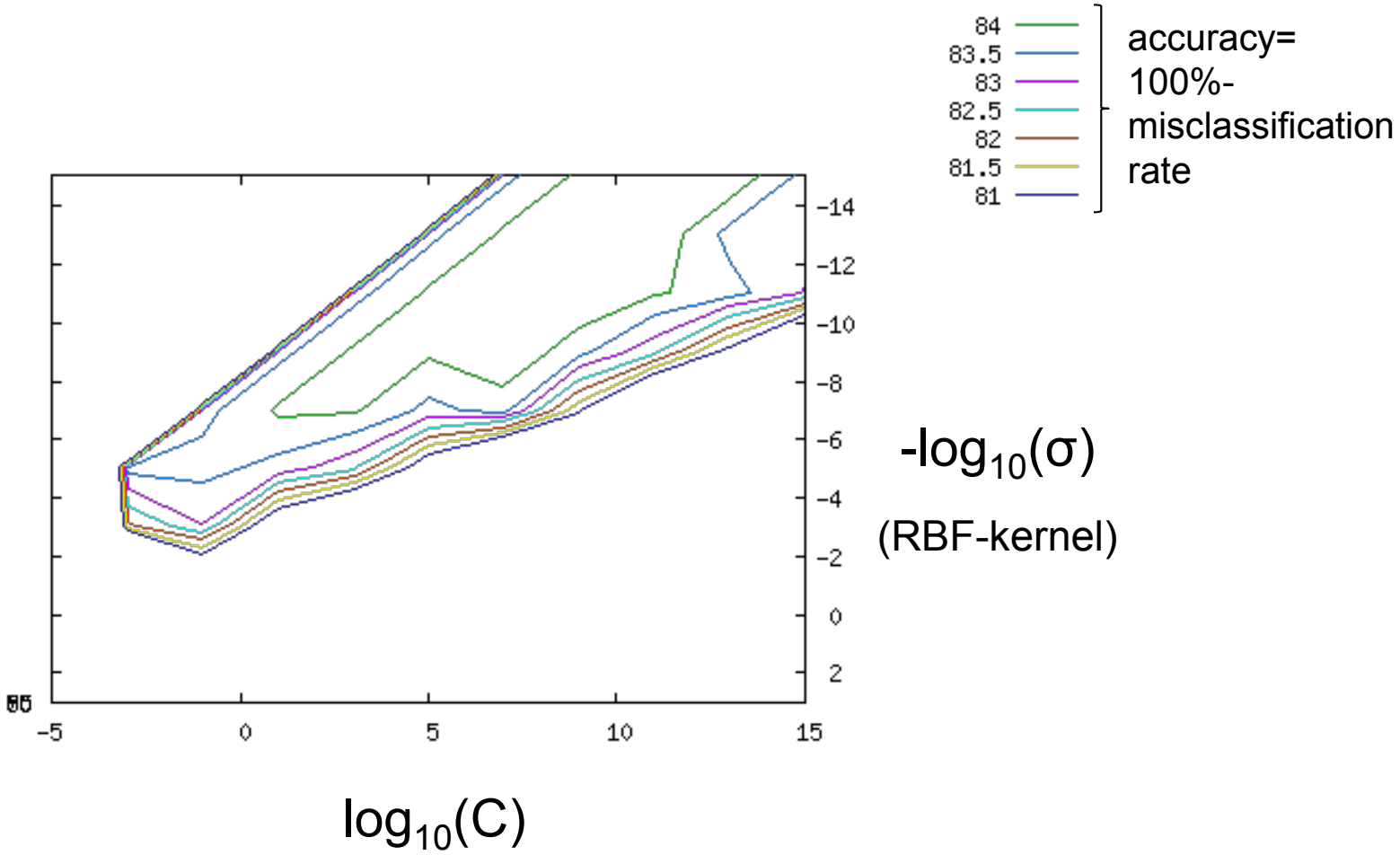
Cross-Validation cont.

- example: 3-fold-cross-validation



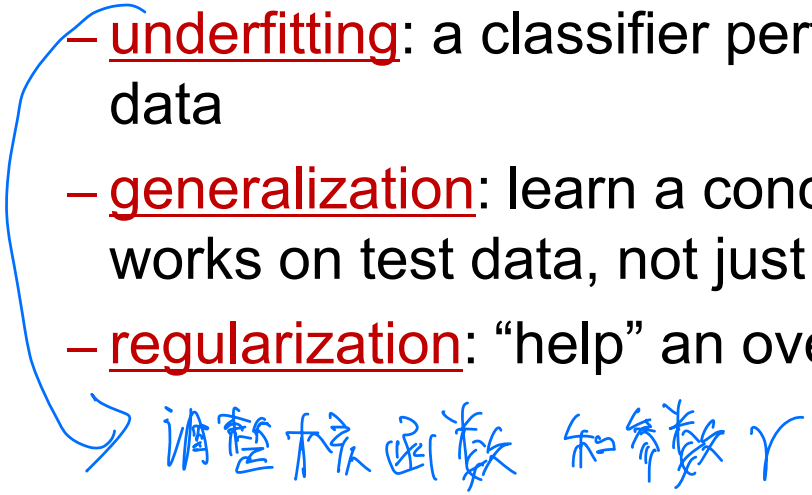
Cross-Validation cont.

- possibility to search the parameter space for optimal parameters, e.g.



Generalization

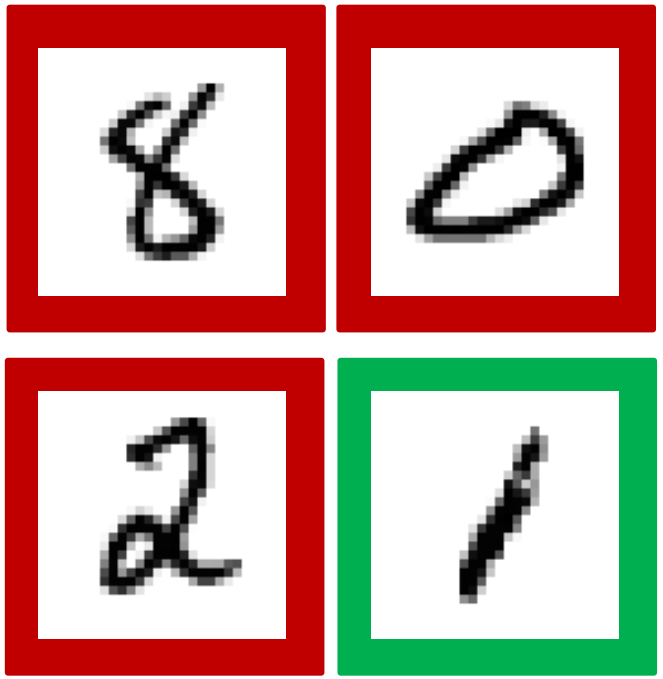
- some concepts you should have heard of:
  - overfitting: a classifier performs well on training data but poor on validation or test data
  - underfitting: a classifier performs poor on both, training and validation data
  - generalization: learn a concept from the training examples that also works on test data, not just memorize the training examples
  - regularization: “help” an overfitted classifier to improve generalization

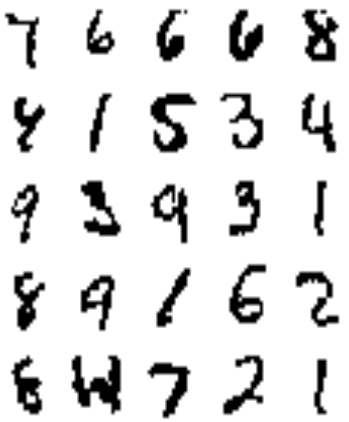


Experiment: Digit Recognition

- classify images of hand-written digits (US postal zip codes)
- simplified task: classify
  - image shows digit “1”
  - image does not show digit “1”

- here:
  - for training and validating: 500 images of “1”, 500 images of “no-1”
  - for testing: 500 images of “1”, 500 images of “not-1”





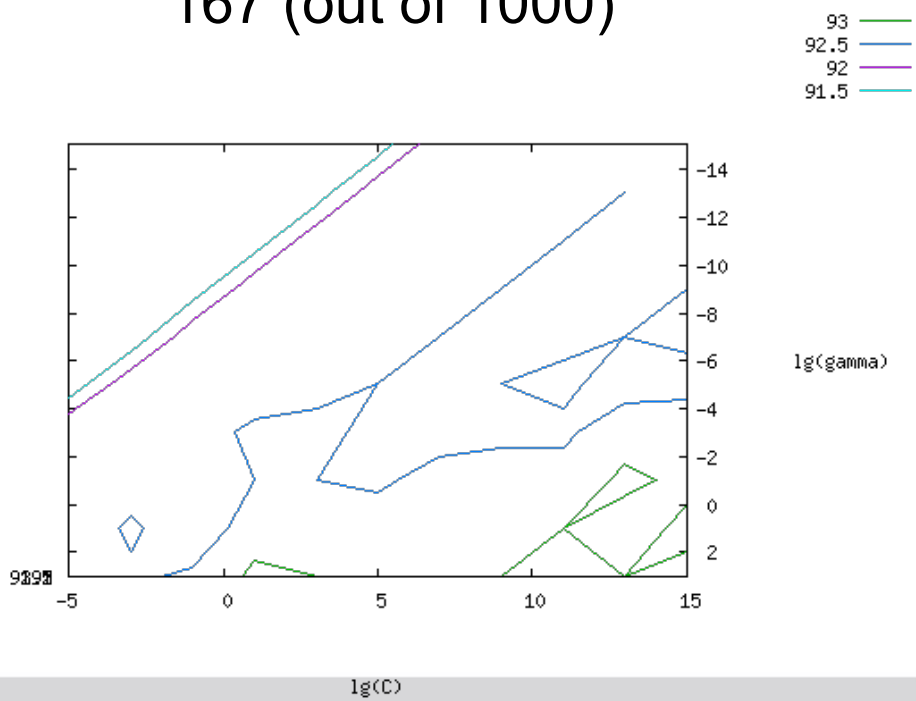
dataset collected by Yann LeCun / http://yann.lecun.com/

Digit Recognition cont.

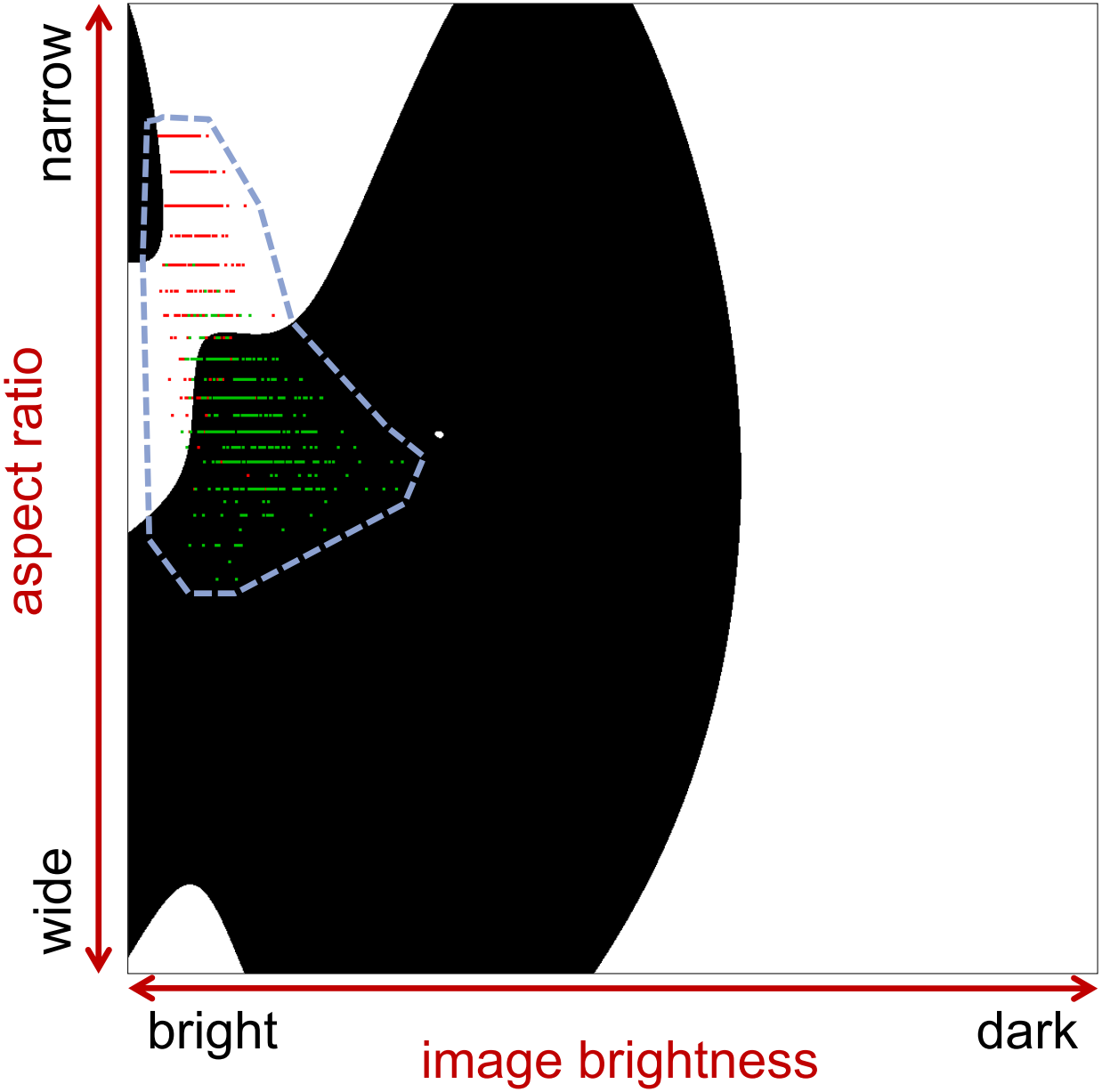
- 1st approach:
  - 2-dimensional patterns
    - average grey value
    - aspect ratio
  - patterns are rescaled to interval [-1, +1]
  - soft-margin SVM with RBF-kernel
  - 5-fold cross validation
  - grid search in parameter space:
    - $10^{-5} \leq C \leq 10^{15}$ (on log scale)
    - $10^{-3} \leq \sigma \leq 10^{15}$ (on log scale)

- accuracy:
  - cross validation: 93.1%
  - test set: 80.3%
- number of support vectors:

  167 (out of 1000)



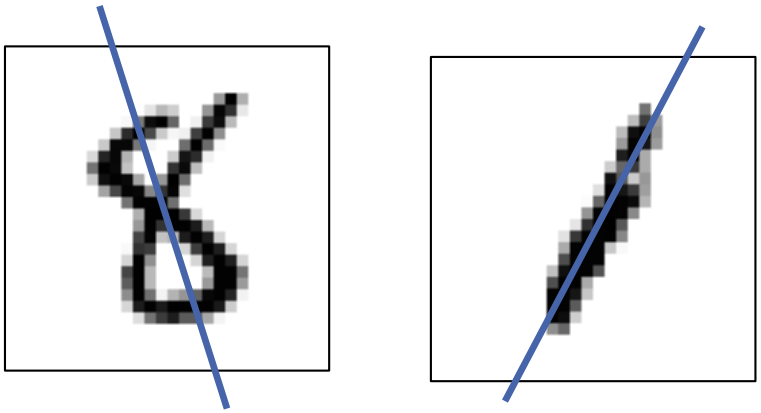
Digit Recognition cont.



- white: “1”
- black: “no-1”

Digit Recognition cont.

- 2nd approach:
  - add a third feature:
    - average distance from line fitted to the dark pixels



第二种方法 (2nd approach)

- **新增特征：**
  - 增加了一个第三个特征：
    - 计算暗像素点到拟合直线的平均距离。
  - **作用：** 这个特征有助于捕捉数字的形状信息（例如“8”和倾斜的数字“1”的差异）。

模型性能

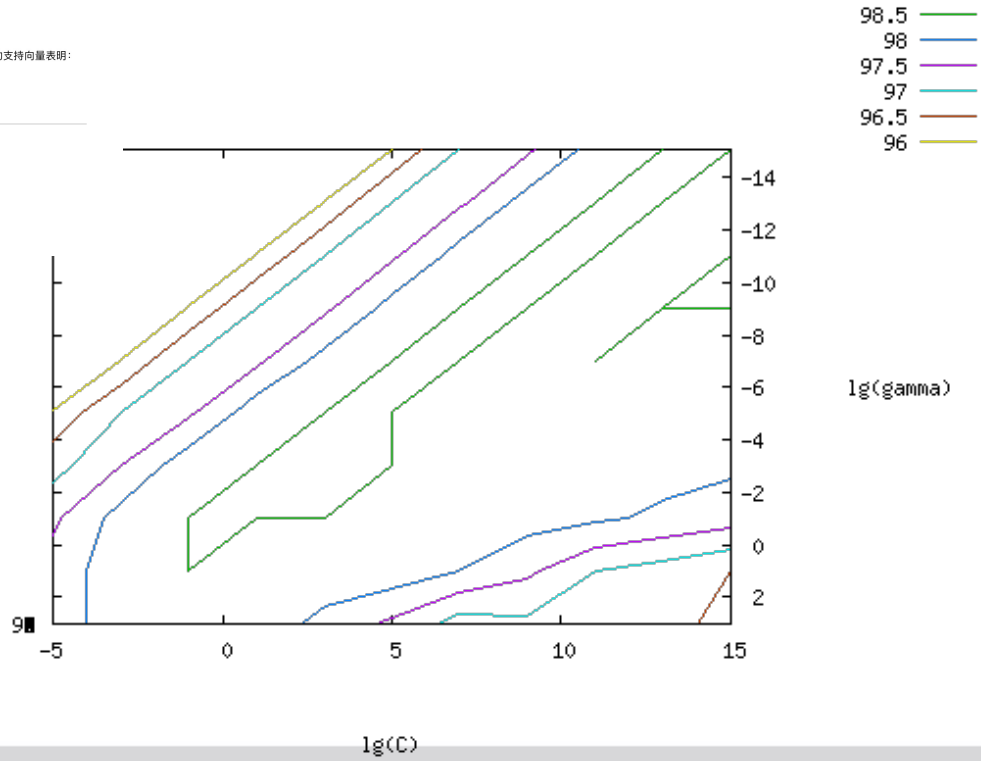
- **准确率 (accuracy)：**
  - **交叉验证 (Cross Validation)：** 98.5%
  - **测试集 (Test Set)：** 98.7%
  - **解释：** 模型在交叉验证和测试集上的高准确率表明其泛化能力较强。
- **支持向量数量 (Number of Support Vectors)：**
  - 支持向量数量为 **95**，而样本总数为 **1000**。
  - **解释：** 支持向量是用于定义分类边界的关键数据点。数量较少的支持向量表明：
    - 决策边界复杂度适中。
    - 模型没有过度依赖特定样本。

右下角图示分析

- **参数调优：**
  - 横轴表示 $\log(C)$，纵轴表示 $\log(\gamma)$。
  - 不同颜色的等高线表示准确率水平：
    - 越靠近右上角，准确率越高（约 ↓ .7%）。

- accuracy:
  - cross validation: 98%
  - test set: 98.7%
- number of support vectors:

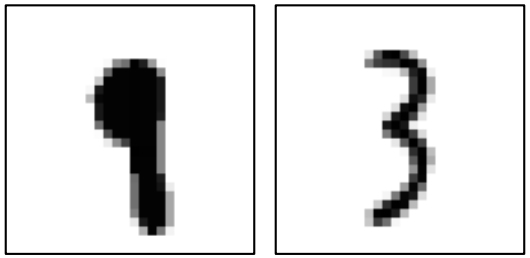
  95 (out of 1000)

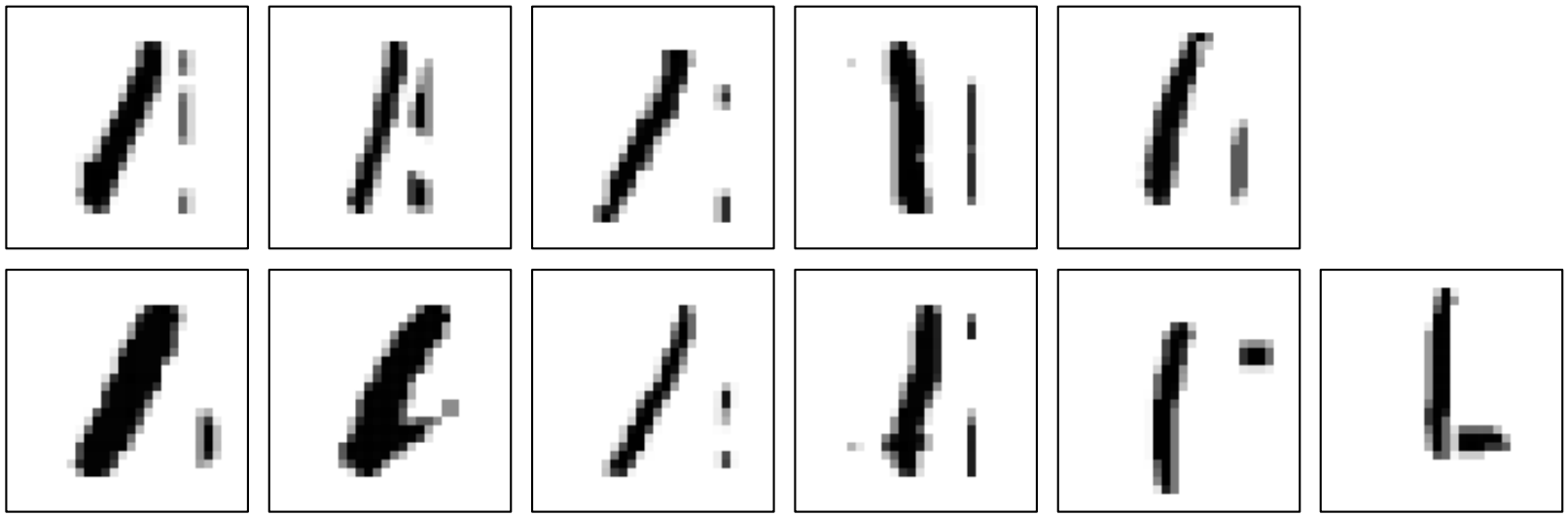


Digit Recognition cont.

- confusion matrix:

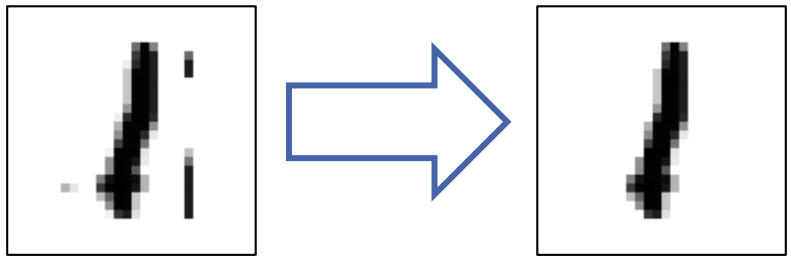
	is 1	is not 1
classified as 1	489	2
classified as not 1	11	498





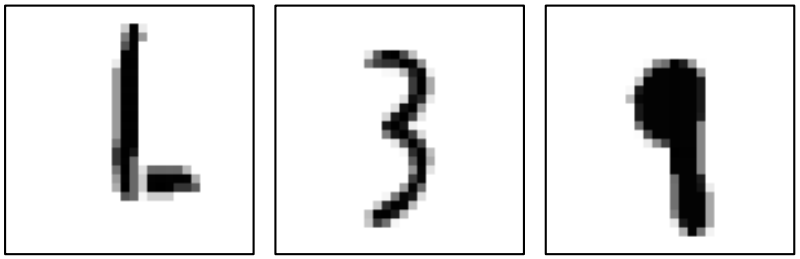
Digit Recognition cont.

- 2nd approach, improvement:
  - find connected components (CCL) and mask out all except the largest segment
  - calculate features from preprocessed image



- accuracy:
  - cross validation: 98.5%
  - test set: 99.7%
- number of support vectors:
  95 (out of 1000)

	is 1	is not 1
classified as 1	499	2
classified as not 1	1	498



Digit Recognition cont.

- 3rd approach:
  - resize all images to 28x28 pixels and use grey values of pixels as features → 784-dimensional patterns

第三种方法

1. **方法描述：**
   - 将所有图像调整为 **28x28 像素**。
   - 使用像素的灰度值作为特征。
   - 形成 **784 维特征向量**（28x28=784）。
2. **核心思想：**
   - 将图像的像素灰度值直接用作特征，完整地保留图像的细节信息，从而帮助模型更精确地分类。

- accuracy:
  - cross validation: 99.0%
  - test set: 98.7%
- number of support vectors:
  220 (out of 1000)

	is 1	is not 1
classified as 1	487	0
classified as not 1	13	500

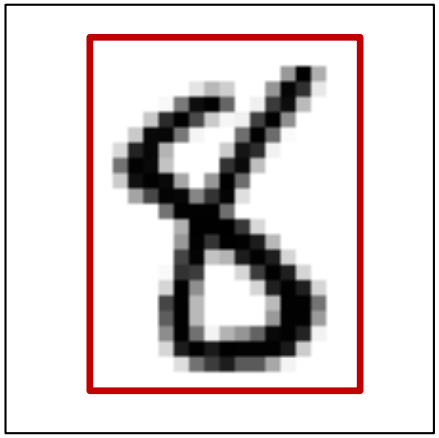
总结与思考

1. **高维特征的优势：**
   - 将图像灰度值作为特征，可以更全面地捕捉数字的特征，尤其适用于复杂分类任务。
2. **支持向量数量增加的意义：**
   - 高维特征使分类边界更加复杂，因此需要更多支持向量来定义边界。
3. **分类性能的稳健性：**
   - 测试集与交叉验证的准确率接近，说明模型泛化性良好。

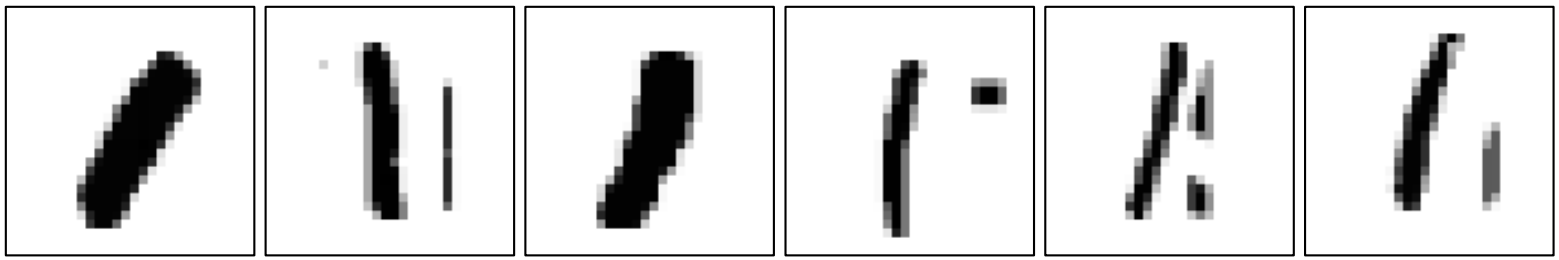
Digit Recognition cont.

- 3rd approach improvement:
  - observation: a lot of pixels do not contribute to the decision, e.g. boundary pixels
  - use only a subset of all pixels, e.g. a 24x18 subarea → 432-dimensional patterns
  - accuracy:
    - cross validation: 99.0%
    - test set: 99.4%
  - number of support vectors:

    219 (out of 1000)



	is 1	is not 1
classified as 1	494	0
classified as not 1	6	500

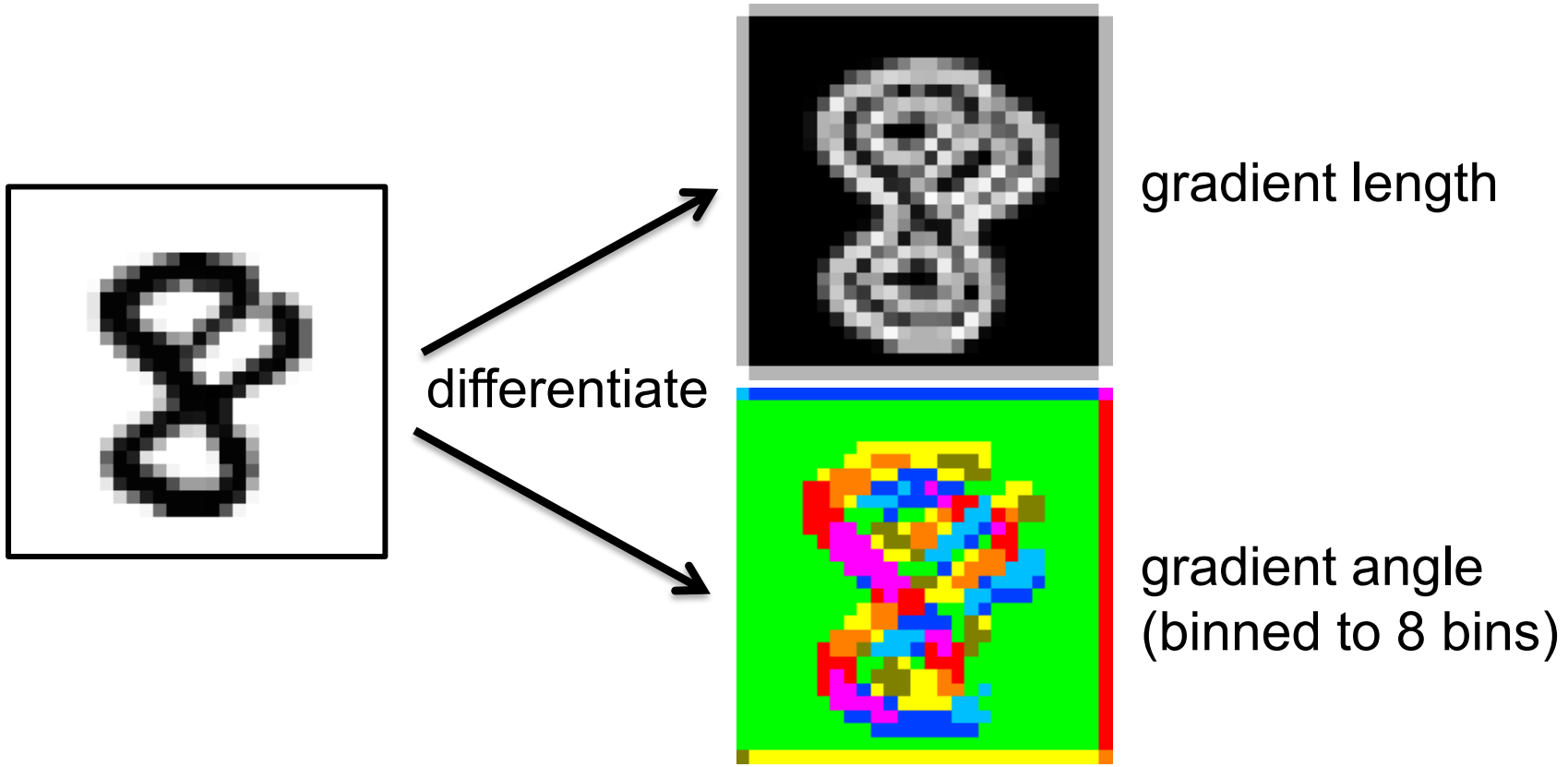


Digit Recognition cont.

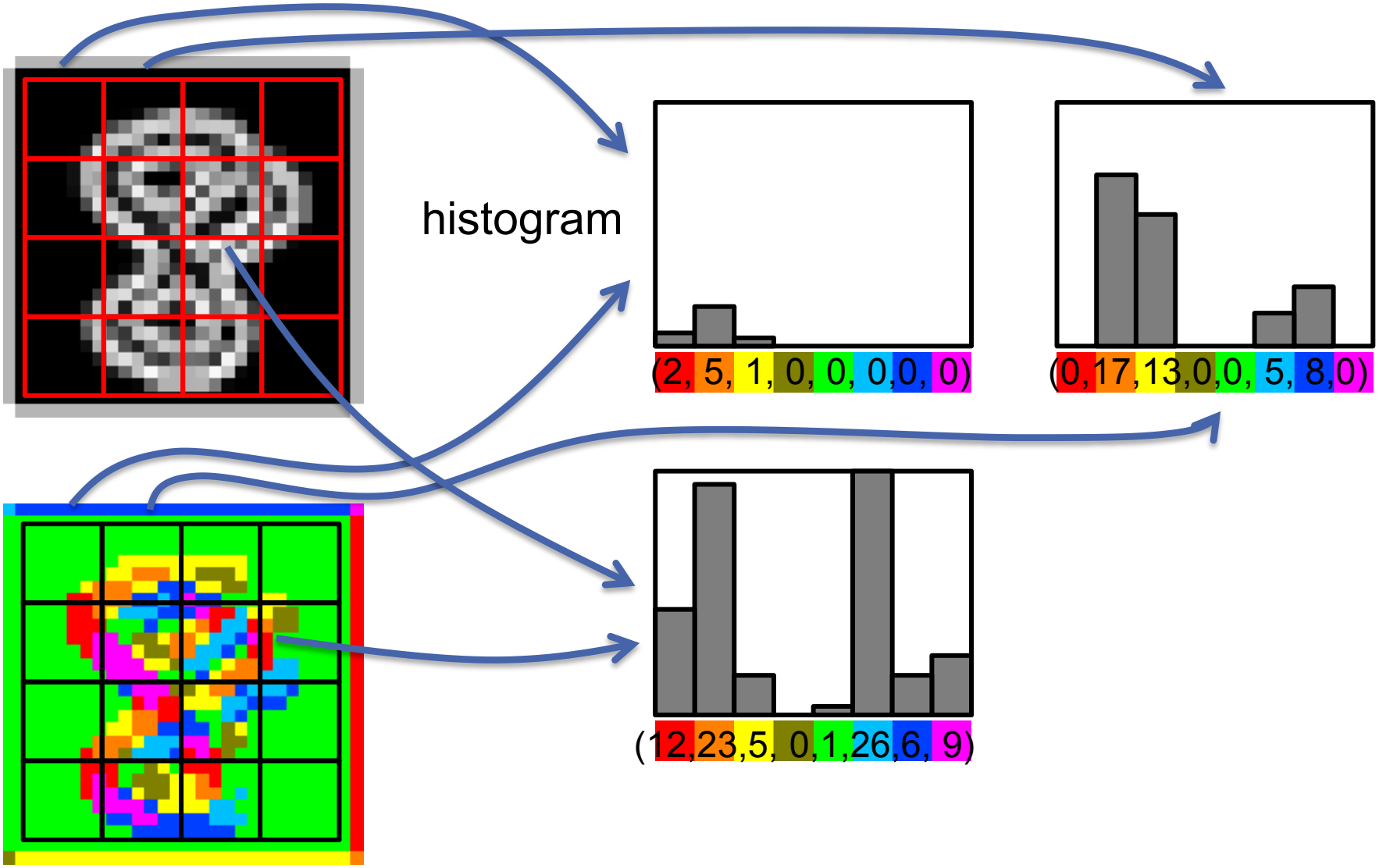
- 4th approach:
  - *HOG*-features

    histogram of oriented gradients (Dalal&Triggs, 2005)

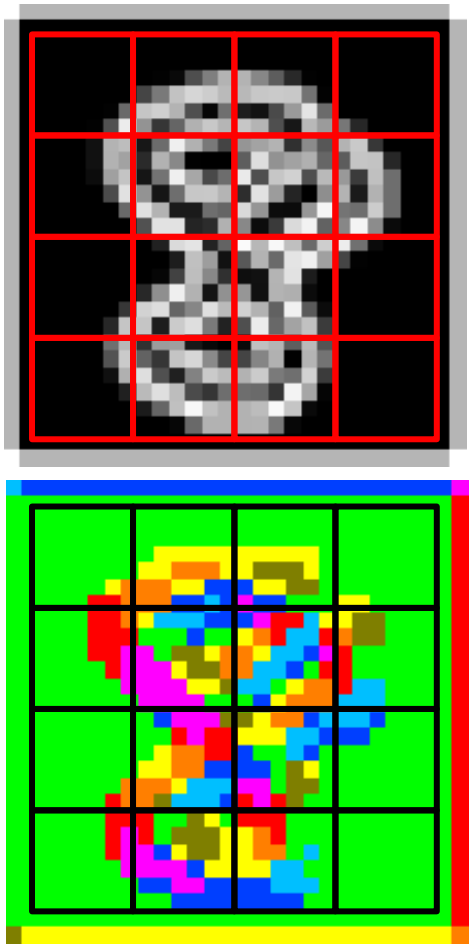
    use gradient information instead of grey levels



Digit Recognition cont.



Digit Recognition cont.



(2, 5, 1, 0, 0, 0,0, 0)

(0,17,13,0,0, 5, 8,0)

⋮

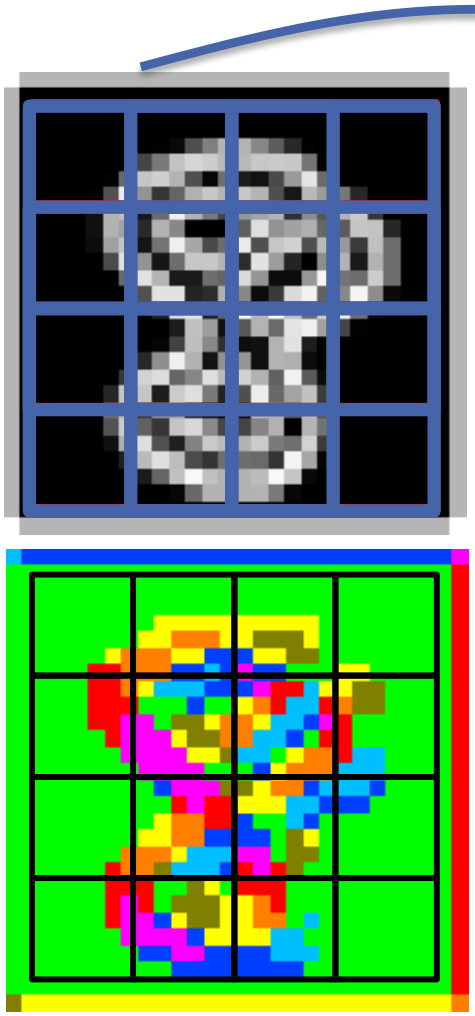
(12,23,5, 0,1,26,6, 9)

⋮

in total: 16 histograms, one for each block

Digit Recognition cont.

– HOG arranges normalized blocks of 4 adjacent cells



- assemble histograms of adjacent cells:

$$\vec{V}_1 = (\underbrace{2,5,1,0,0,0,0,0,}_{\text{from cell 1}} \underbrace{0,17,13,0,0,5,8,0,}_{\text{from cell 2}} \underbrace{15,0,0,0,0,0,0,7,}_{\text{from cell 5}} \underbrace{0,2,4,3,2,3,2,12}_{\text{from cell 6}})$$

- normalize descriptor:

$$\vec{V}_1^{norm} = \frac{\vec{V}_1}{||\vec{V}_1|| + \epsilon}$$

- assemble descriptors of all blocks:

$$\vec{V} = (\vec{V}_1^{norm}, \ldots, \vec{V}_9^{norm})$$

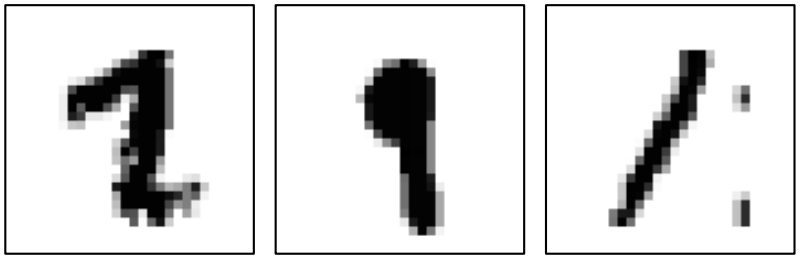
- apply vectors $\vec{V}$ to SVM

Digit Recognition cont.

- 4th approach:
  - use only HOG features
    → 288-dimensional patterns

- accuracy:
  - cross validation: 99.4%
  - test set: 99.7%
- number of support vectors:
  174 (out of 1000)

	is 1	is not 1
classified as 1	499	2
classified as not 1	1	498



4th

方法描述

1. **特征提取**：
   - 仅使用 **HOG（Histogram of Oriented Gradients）** 特征。
   - 每个样本被表示为一个 **288 维特征向量**。
2. **模型训练与测试**：
   - 在提取的 HOG 特征上训练支持向量机（SVM）分类器。
   - 分类任务目标是识别图像是否为数字 “1”。

观察与分析

1. **优势**：
   - 使用 HOG 特征有效地提取了图像的边缘和形状信息，使分类任务更容易。
   - 高准确率表明模型在这个特征表示下表现优秀。
2. **HOG 特征的作用**：
   - HOG 特征捕获了局部梯度方向的分布，是图像分类和目标检测中常用的特征。
   - 在低维空间（288维）中提供了足够的判别力。
3. **支持向量数量**：
   - 支持向量数量适中，表明模型既没有过于复杂，也不是过于简单，具有较好的泛化能力。

5th

1. **Haar 特征的定义**：
   - Haar 特征通过比较矩形区域内的灰度级差值来描述图像的局部特性。
   - 计算公式：红色区域的平均灰度值减去蓝色区域的平均灰度值。
2. **特征种类**：
   - **边缘特征（Edge Features）**：用于捕获图像中的边界。
   - **线条特征（Line Features）**：用于检测图像中的直线。
3. **Haar 特征的数量**：
   - 由于矩形区域可以有不同的位置、大小和方向，可能的 Haar 特征数量非常多。

优点

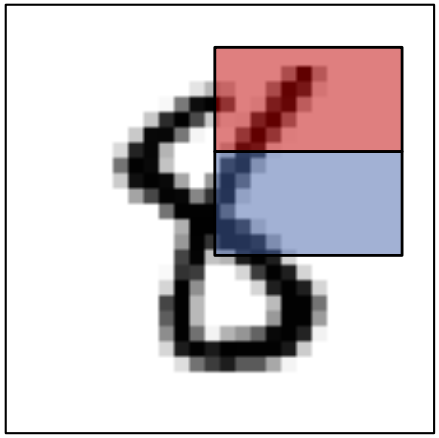
- **鲁棒性**：Haar 特征对灰度变化较敏感，能够有效捕获边缘和线条信息。
- **计算效率**：通过积分图像（Integral Image）技术，可以快速计算 Haar 特征值。

可能的挑战

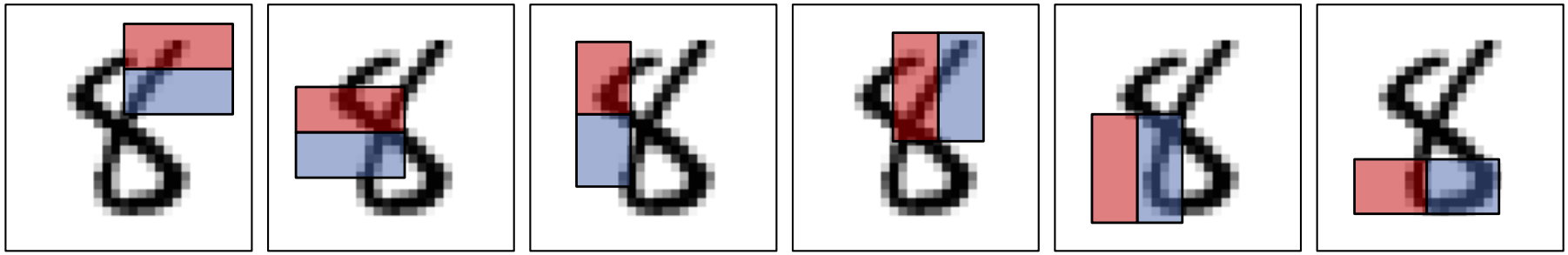
- **特征筛选**：由于可能的 Haar 特征数量非常多，需要设计算法挑选最具判别力的特征。
- **局限性**：在复杂背景或高噪声情况下，Haar 特征可能失去部分判别力。

Digit Recognition cont.

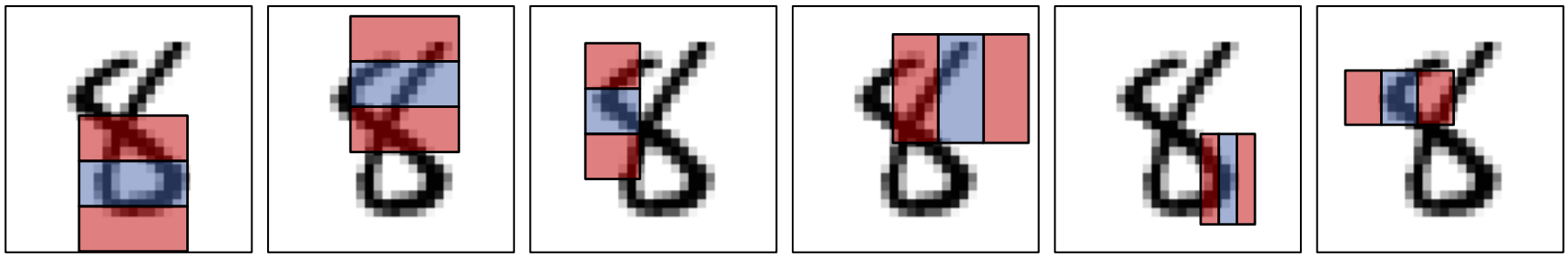
- 5th approach:
  - *Haar features*
    - *compare gray levels of rectangular areas, i.e. average gray level in red area minus average gray level in blue area*
    - *very many possible features*



- edge features

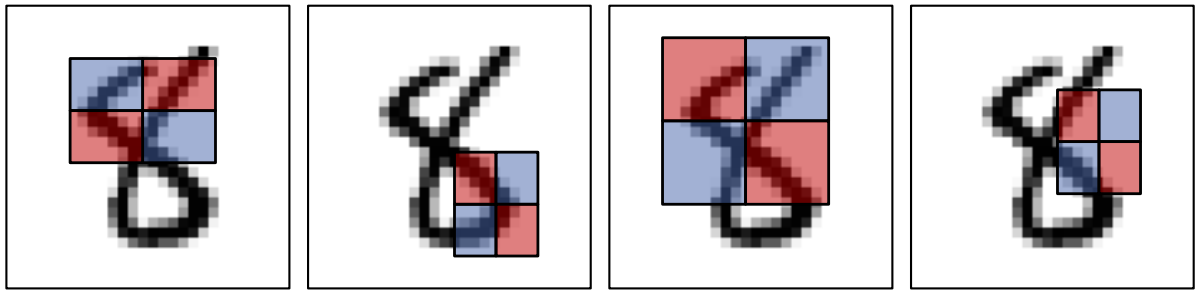


- line features

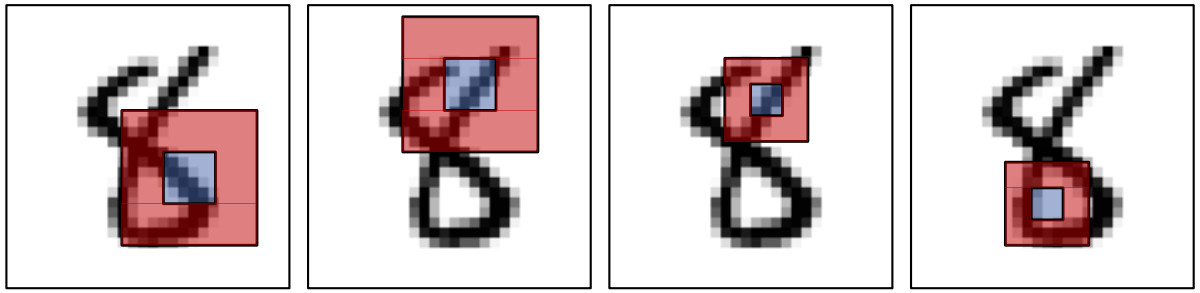


Digit Recognition cont.

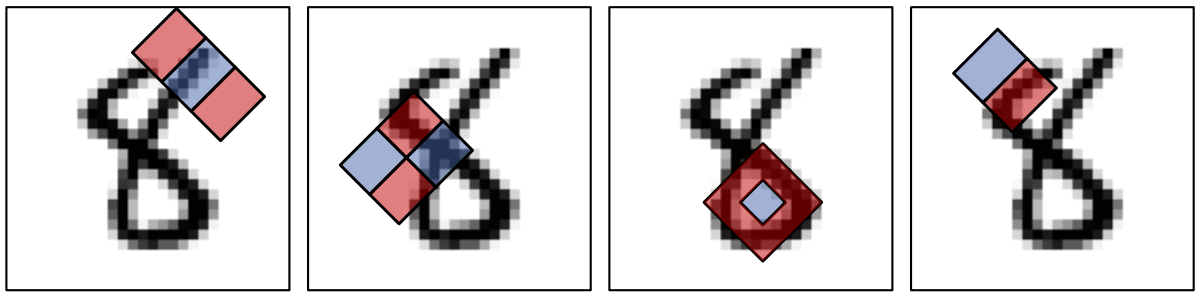
- chessboard features



- center-surround features



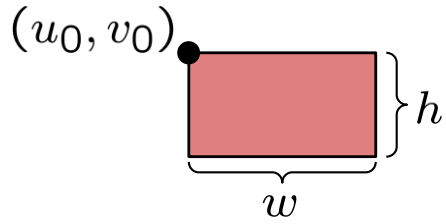
- features with diagonal orientation



Digit Recognition cont.

- Calculating Haar features:
  - the naïve way:

$$s = \sum_{u=u_0}^{u_0+w-1} \sum_{v=v_0}^{v_0+h-1} g(u,v)$$



implementing this with for loops requires $O(w \cdot h)$ operations.

  - the smart way:

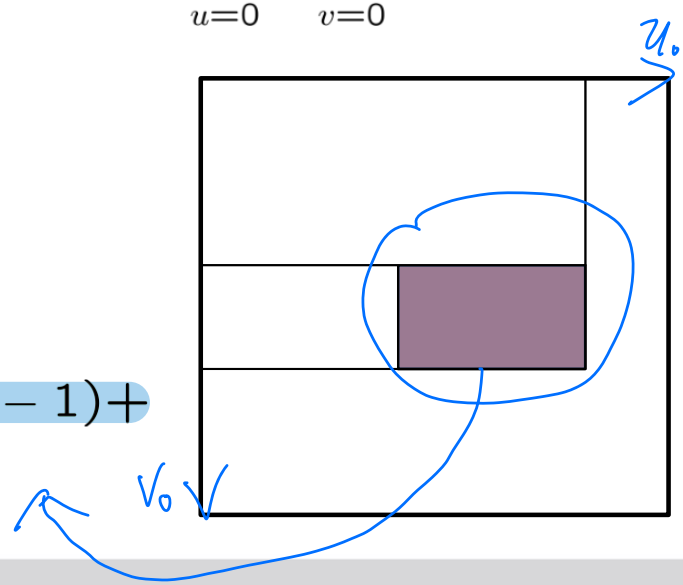
$$s = \sum_{u=0}^{u_0+w-1} \sum_{v=0}^{v_0+h-1} g(u,v) - \sum_{u=0}^{u_0-1} \sum_{v=0}^{v_0+h-1} g(u,v) + \sum_{u=0}^{u_0-1} \sum_{v=0}^{v_0-1} g(u,v) - \sum_{u=0}^{u_0+w-1} \sum_{v=0}^{v_0-1} g(u,v)$$

- the integral image:

$$I(x,y) := \sum_{u=0}^{x} \sum_{v=0}^{y} g(u,v)$$

- calculating *s* requires 4 operations:

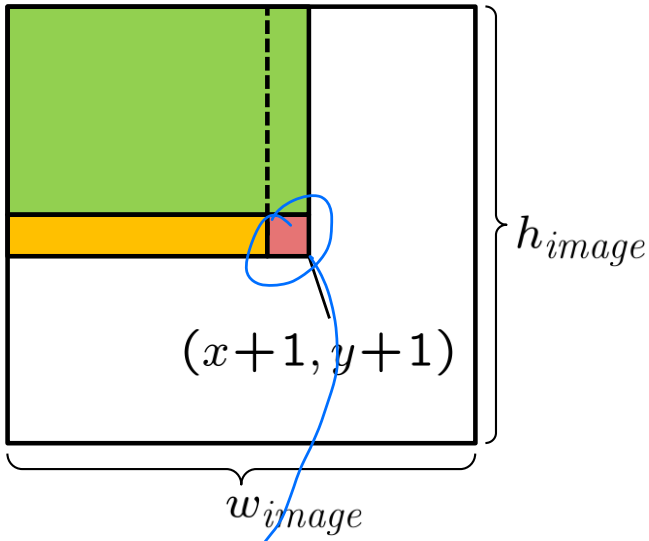
$$s = I(u_0+w-1, v_0+h-1) - I(u_0-1, v_0+h-1) + I(u_0-1, v_0-1) - I(u_0+w-1, v_0-1)$$



Digit Recognition cont.

- calculating the integral image

$$I(x,y) := \sum_{u=0}^{x} \sum_{v=0}^{y} g(u,v)$$



$$I(x+1,y+1) = \sum_{u=0}^{x+1} \sum_{v=0}^{y+1} g(u,v)$$

$$= \sum_{u=0}^{x+1} \sum_{v=0}^{y} g(u,v) + \sum_{u=0}^{x} \sum_{v=0}^{y+1} g(u,v) - \sum_{u=0}^{x} \sum_{v=0}^{y} g(u,v) + g(x+1,y+1)$$

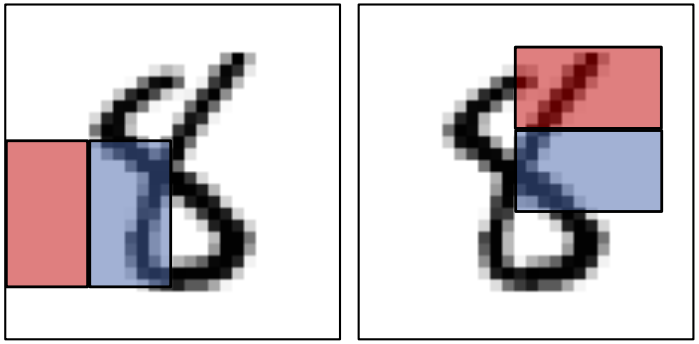
$$= I(x+1,y) + I(x,y+1) - I(x,y) + g(x+1,y+1)$$

⇒ yields an iterative algorithm that calculates the whole integral image with $O(w_{image} \cdot h_{image})$ operations

⇒
- naïve approach is superior if you want to calculate one rectangle
- integral image is superior if you want to calculate many rectangles

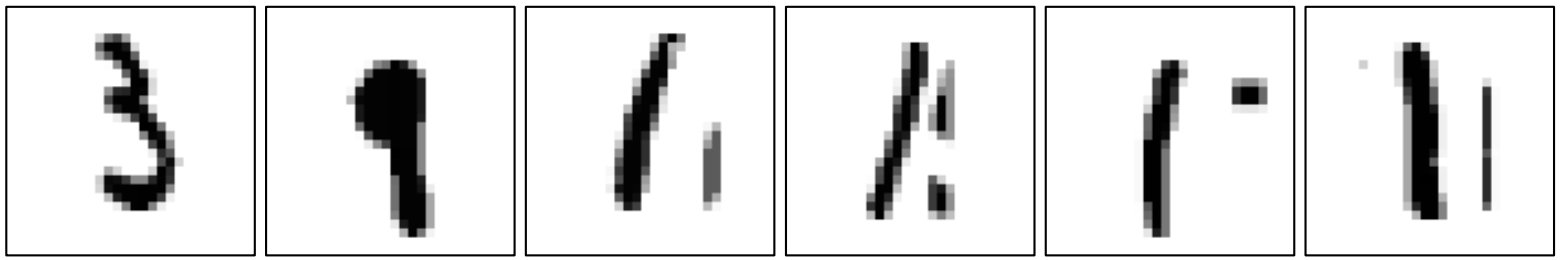
Digit Recognition cont.

- 5th approach:
  - Haar features, use horizontal and vertical edge features at 7x7 positions
    → 98-dimensional patterns



- accuracy:
  - cross validation: 99.1%
  - test set: 99.4%
- number of support vectors:
  109 (out of 1000)

	is 1	is not 1
classified as 1	496	2
classified as not 1	4	498



5.th

1. **特征描述**：
   - 使用 **水平和垂直的边缘特征**，即通过红蓝区域之间的灰度差异捕获图像的局部边界。
   - 特征提取位置：在每个 7×7 的区域中提取。
   - 特征维度：共提取 **98 维特征**。

优势

1. **特征紧凑**：
   - 相较于其他方法，如直接使用像素值或 HOG 特征，Haar 特征的维度较低，仅为 98 维，能有效减少计算量。
2. **高准确率**：
   - 准确率接近 99.4%，说明其特征对数字的边缘信息高度敏感。

不足

1. **特征单一**：
   - 仅依赖水平和垂直边缘特征，可能会对非标准书写的数字（例如弯曲的 “1” 或 “8”）失去一定的判别力。
2. **数据预处理要求高**：
   - Haar 特征通常需要图像对齐，且灰度差异对噪声敏感。

总结

- 这种方法通过有效利用 Haar 特征中的边缘信息，进一步提升了数字识别的性能，同时减少了计算复杂度。
- 然而，其特征表达能力可能不足以覆盖复杂的数字形状或非标准书写形式。

6.th.

方法细节

1. **核心思想**：
   - 分析图像局部区域中的灰度级变化，捕获图像的纹理信息。
   - 对多个局部区域计算直方图，生成纹理特征。
2. **LBP计算过程**：
   - **邻域比较**：
     - 对于中心像素，检查其周围像素是否比中心像素更亮（标记为 1）或更暗（标记为 0）。
   - **二进制编码**：
     - 将这些比较结果按顺序连接形成一个二进制数。
   - **十进制转换**：
     - 最终将二进制数转换为十进制值，例如图示中得到的值为 **22**。
   - **直方图统计**：
     - 统计局部区域中不同 LBP 值的频率分布，用于描述该区域的纹理特征。

优势

1. **纹理识别能力强**：
   - LBP 能够有效捕获局部的边缘、角点等纹理信息。
2. **计算简单高效**：
   - 计算过程仅涉及简单的比较和统计，计算复杂度较低。
3. **对光照变化鲁棒**：
   - 由于 LBP 基于局部的相对亮度变化，对全局光照变化不敏感。

不足

1. **局部信息限制**：
   - LBP 只关注局部纹理，无法有效捕获全局信息或复杂的形状特征。
2. **对噪声敏感**：
   - 边缘噪声或局部灰度级的异常变化可能对 LBP 特征造成较大影响。
3. **维度较高**：
   - 通过多个局部区域的直方图组合可能导致特征维度快速增长。

总结

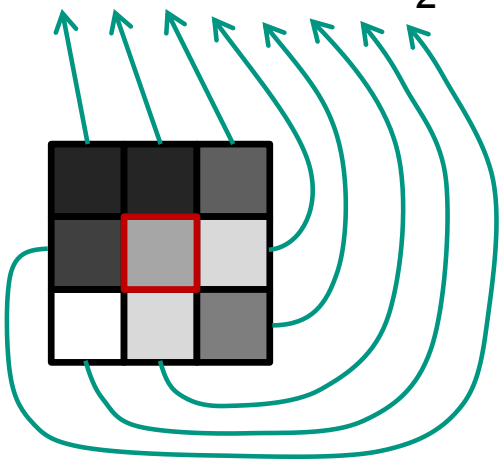
- 局部二值模式（LBP）是一种轻量级的纹理特征提取方法，适用于纹理信息显著的图像分类任务，如数字识别。
- 然而，对于光滑或复杂形状的数字（例如 “0” 和 “8”），可能需要与其他特征结合使用以增强分类性能。

Digit Recognition cont.

- 6th approach:
  - *Local binary patterns (LBP)*
    - *analyze local gray level changes*
    - *perform histograms for multiple areas*

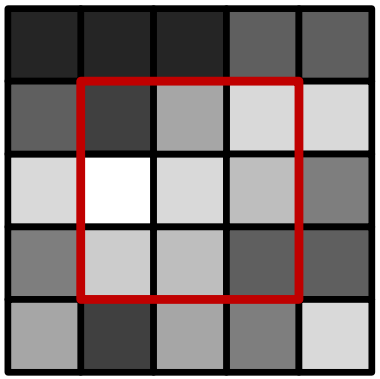
- for each neighboring pixel, check whether neighboring pixel is brighter (1) or darker (0)

$0\ 0\ 0\ 1\ 0\ 1\ 1\ 0_2 = 22_{10}$



Digit Recognition cont.

- 6th approach:
  - *Local binary patterns (LBP)*
    - *analyze local gray level changes*
    - *perform histograms for multiple areas*

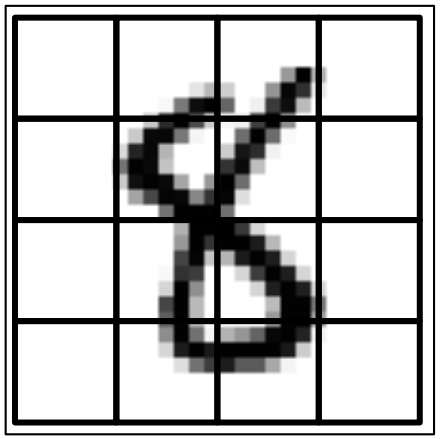


- for each neighboring pixel, check whether neighboring pixel is brighter (1) or darker (0)
- calculate those numbers for all pixels in a block

$00011111_2=31_{10}$
$00010110_2=22_{10}$
$00000000_2=0_{10}$
$00000000_2=0_{10}$
$00100001_2=33_{10}$
$11100001_2=225_{10}$
$11100000_2=224_{10}$
$11100001_2=225_{10}$
$11101111_2=239_{10}$

Digit Recognition cont.

- 6th approach:
  - *Local binary patterns (LBP)*
    - *analyze local gray level changes*
    - *perform histograms for multiple areas*



- for each neighboring pixel, check whether neighboring pixel is brighter (1) or darker (0)
- calculate those numbers for all pixels in a block
- make a histogram
- arrange histograms for all blocks in one vector

31, 22, 0, 0, 33, 225, 224, 225, 239, …

value	0	1	2	3	4	5	6	7	8	9	10	11	…	31	32	33	34	…	255
frequency	2	0	0	0	0	0	0	0	0	0	0	0	…	1	0	1	0	…	0

Digit Recognition cont.

- 6th approach:
  - Local binary patterns
    → 4096-dimensional, sparse patterns

- accuracy:
  - cross validation: 98.6%
  - test set: 99.3%
- number of support vectors:
  264 (out of 1000)

	is 1	is not 1
classified as 1	495	2
classified as not 1	5	498

Digit Recognition cont.

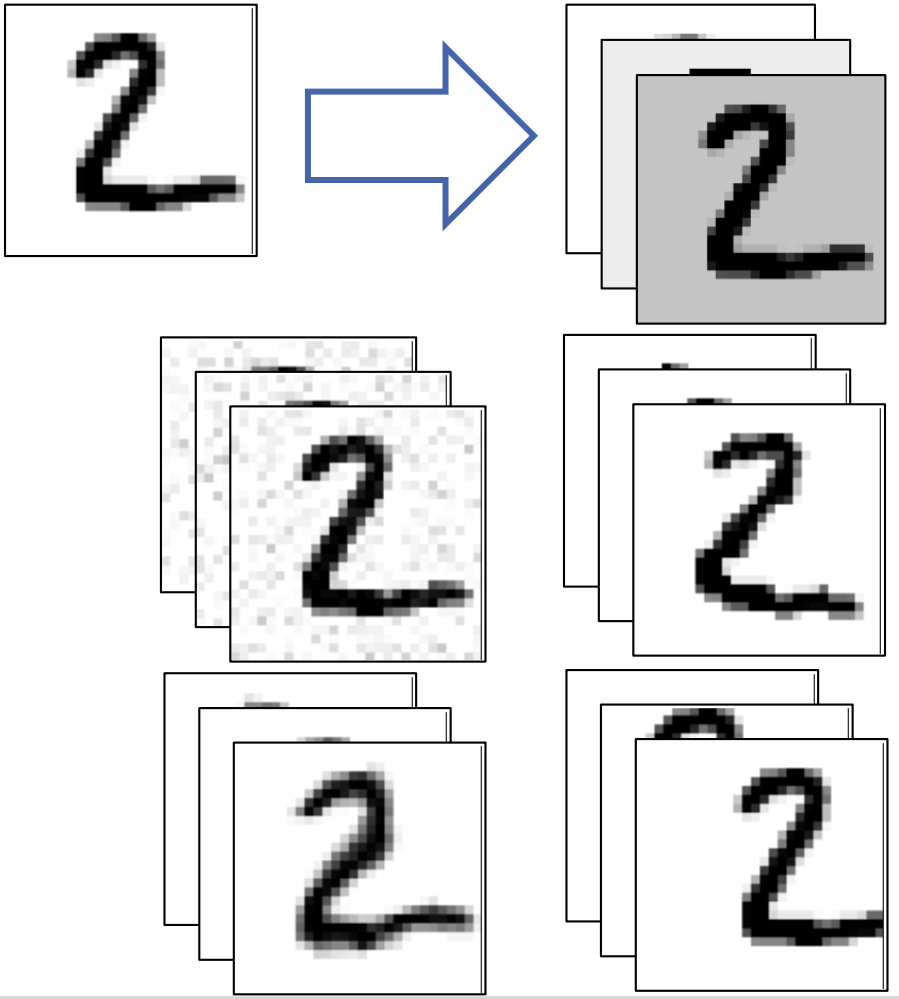
approach	1	2 (line fit)	3 (pixels)	4 (HOG)	5 (Haar)	6 (LBP)
features	2	3	432	288	98	4096
accuracy cross validation	93.1%	98.5%	99.0%	99.4%	99.1%	98.7%
accuracy test set	80.3%	99.7%	99.4%	99.7%	99.4%	99.3%
number of support vectors	167	95	219	174	109	264

- insights:
  - accuracy on training and test do not vary in the same way
  - "smart" features help a lot
  - more features do not mean higher accuracy
  - different approaches:
    - "smart" features including preprocessing
    - "generic" features (pixel values, HOG, Haar)

Data Tuning

Quality and quantity of training data have high impact on classification results

- How can we improve quantity?
  - select and label more images
  - search databases/internet for more training examples (ImageNet, KITTI, CalTech dataset, INRIA dataset, Microsoft COCO, …)
  - vary examples in brightness, contrast, position of object in ROI, rotation
  - add jitter (random noise)
  - mirror examples, if objects are symmetric
  - elastic distortions



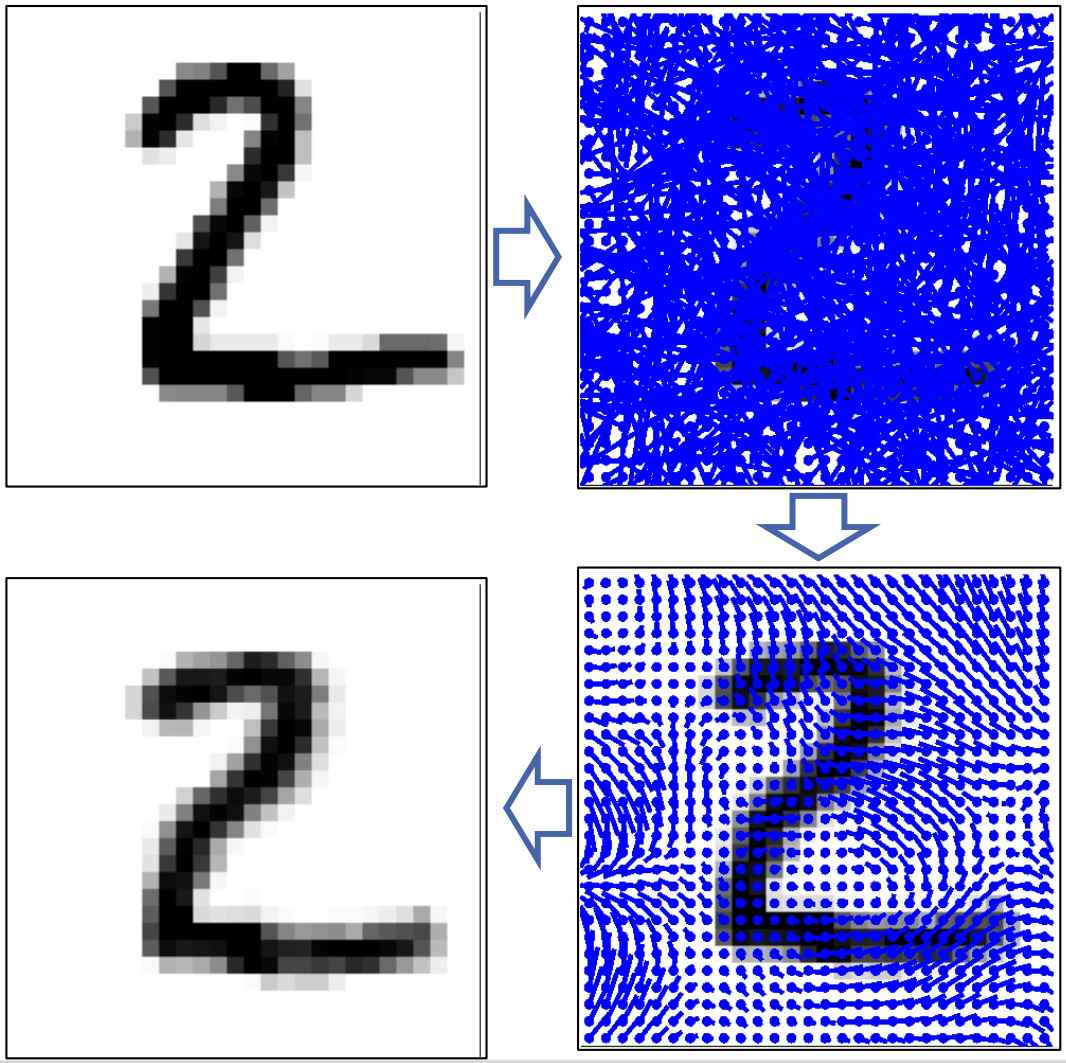
Elastic Distortion

↳ 弹性变形，one of data Augmentation

- Goal: get distorted example images to extend data set

Elastic distortion

1. for each pixel: sample random shift from a Gaussian distribution
2. smooth shift values by convolution with Gaussian filter
3. for each pixel: shift pixel to new position



目标：

生成经过扭曲的样本图像，以扩展数据集，特别适用于手写数字或字符识别任务（如 MNIST 数据集）。

方法：

1. **对每个像素生成随机偏移**：
   - 每个像素的偏移量从 **高斯分布（Gaussian distribution）** 中随机采样，产生随机的扭曲效果。
2. **平滑偏移量**：
   - 使用 **高斯滤波器（Gaussian filter）** 对偏移量进行卷积操作，使偏移平滑化，避免出现过于突兀的扭曲效果。
3. **将像素移动到新的位置**：
   - 按照计算出的平滑偏移量，将每个像素移动到其新的位置。

作用：

1. **增强数据多样性**：
   - 通过添加随机变形，生成大量稍有差异的图像，有助于提高模型对数据的泛化能力。
2. **防止过拟合**：
   - 模型通过学习不同形式的图像变形，能够更好地适应实际数据的变化。
3. **模拟手写特征**：
   - 特别适用于手写识别任务，可以模拟不同书写习惯下的形状变化。

优点：

1. 操作简单。
2. 能够显著提高数据集的多样性。
3. 对小型数据集特别有效。

局限性：

1. 如果扭曲过于严重，可能影响数据的语义信息（如手写数字变得难以辨认）。
2. 执行时间可能较长，特别是在处理大规模数据集时。

Data Tuning cont.

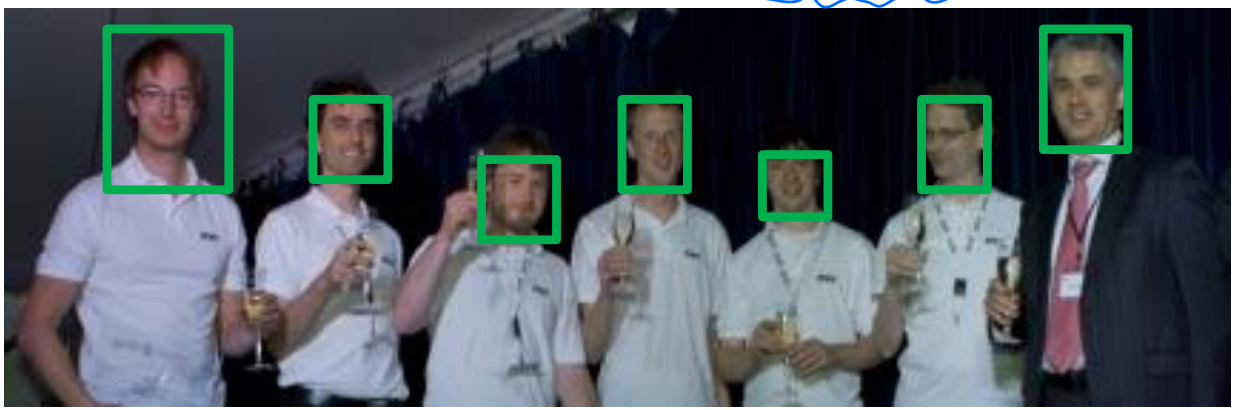
Quality and quantity of training data have high impact on classification results

$$x_i' = \frac{x_i - \bar{x}}{s_x}$$

$$with\ \bar{x} = \frac{1}{n}\sum_i x_i$$

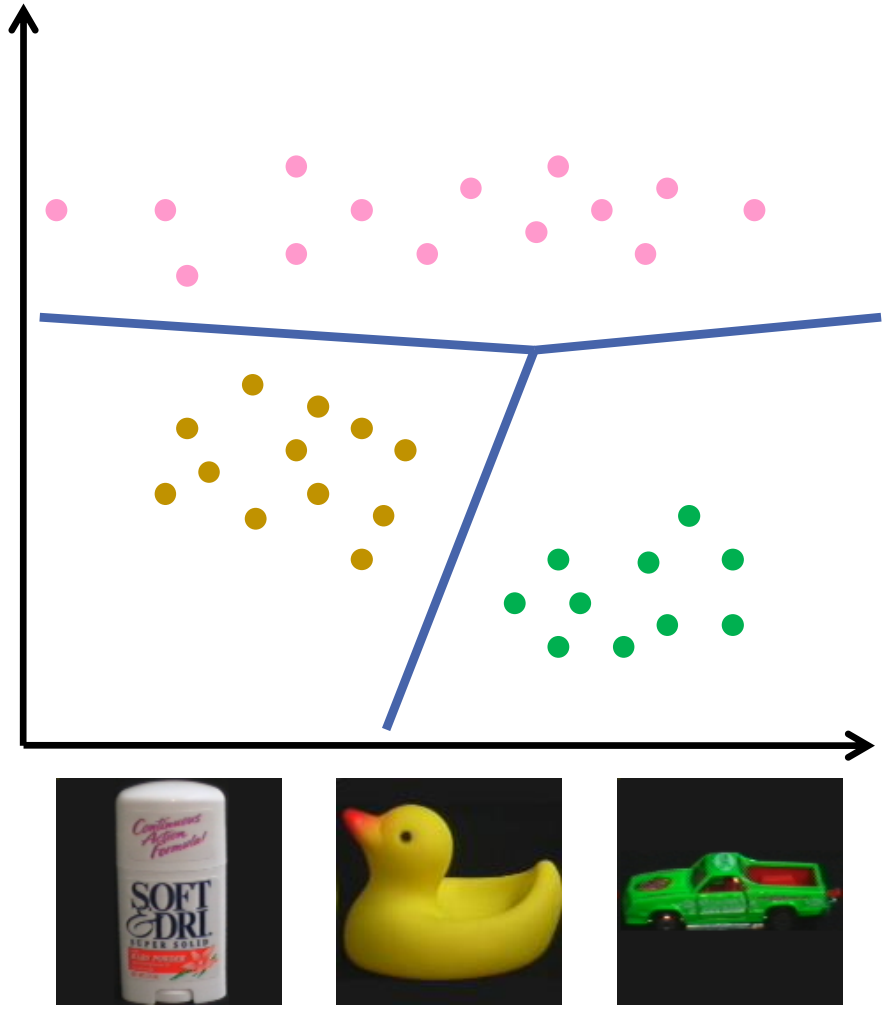
$$and\ s_x = \sqrt{\frac{1}{n}\sum_i (x_i - \bar{x})^2}$$

- How can we improve quality?
  - check consistency of labels
  - standardize/normalize patterns
  - take data from various sources/from various image sequences with varying conditions → increase variation within pattern set
  - check whether ROI is consistent



Multi-Class-Classification

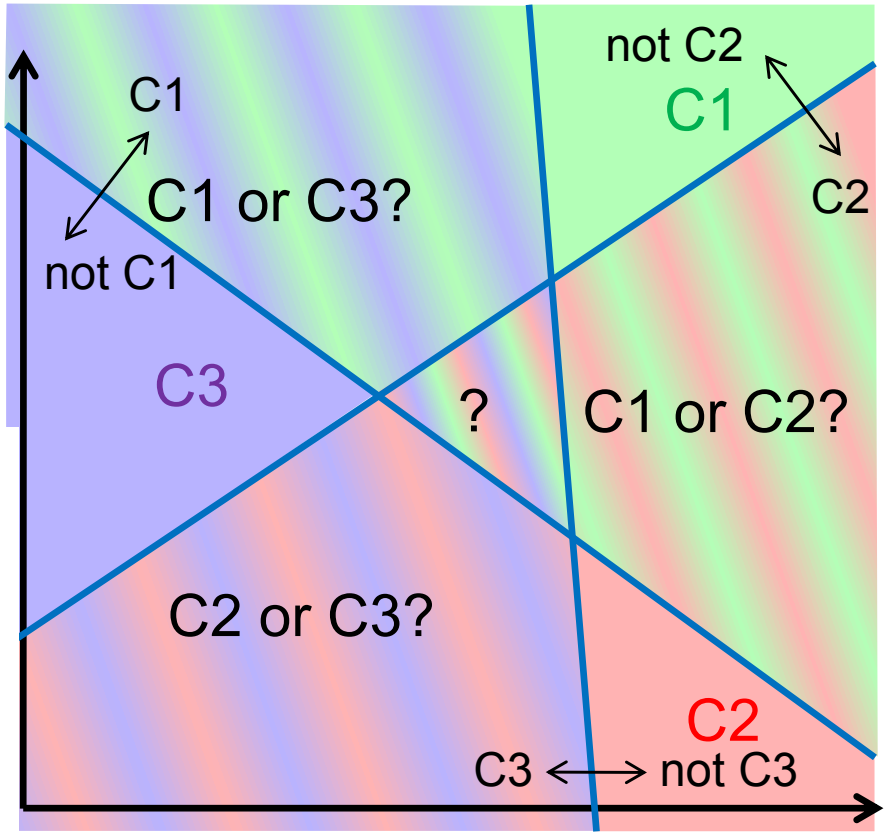
- classification with more than two classes



Multi-Class-Classification cont.

- *one-versus-the-rest* approach:

  build one classifier for each class that classifies class elements versus non-class elements



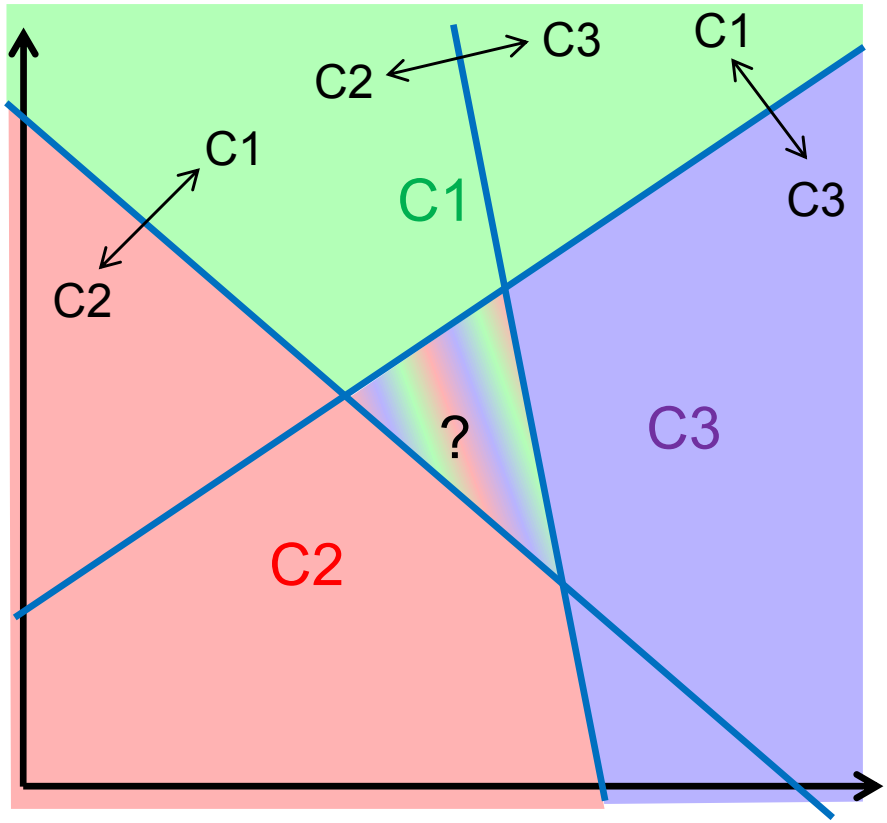
1. 一对多（One-versus-the-Rest）方法

- **定义**：为每个类别构建一个分类器，将该类别的样本与其他所有类别的样本进行区分。
- **方法**：
  - 对于 N 个类别，构建 N 个分类器。
  - 每个分类器将某一个类别与其他类别分开。
  - 如果样本点处于多个分类器的正类区域，可能会出现模糊的决策边界。
- **优点**：
  - 实现简单，适用于任意二分类器。
  - 训练速度较快。
- **缺点**：
  - 决策边界可能重叠，导致对某些样本的分类不确定。
  - 在类别数量较多时，模型可能不够精确。

Multi-Class-Classification cont.

- *one-versus-one* approach:

  build one classifier for each pair of classes and use voting of classifiers – overcoming ambiguities



2. 一对一（One-versus-One）方法

- **定义**：为每对类别构建一个分类器，并通过投票机制解决模糊区域。
- **方法**：
  - 对于 N 个类别，构建 $\binom{N}{2} = \frac{N(N-1)}{2}$ 个分类器。
  - 每个分类器仅区分两类样本。
  - 测试时，根据所有分类器的投票结果确定最终类别。
- **优点**：
  - 通过对两两类别的区分，可以更好地解决类别之间重叠的问题。
  - 决策边界更精确，适合复杂类别分布。
- **缺点**：
  - 类别较多时，分类器数量会急剧增加，计算复杂度高。
  - 训练和测试过程耗时。

使用场景

- **一对多**：适用于类别之间分布相对分散、独立的情况。
- **一对一**：适合类别之间关系复杂且分布有重叠的场景。

ENSEMBLE METHODS

Ensembles

- what do politicians do when they don't know how to decide?
- ask an expert:
  - answer depends on expert's expertise
  - answer depends on expert's point of view

image source: wikipedia

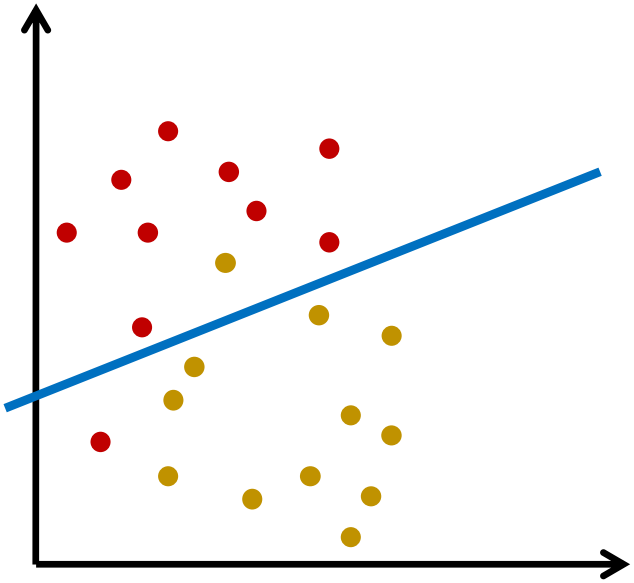
image source: wikipedia

- ask many experts
  - quality: experts must have big expertise
  - fairness & independence: experts must not be selected depending on their point of view
  - decision of committees of experts might be better than individual experts

Ensembles

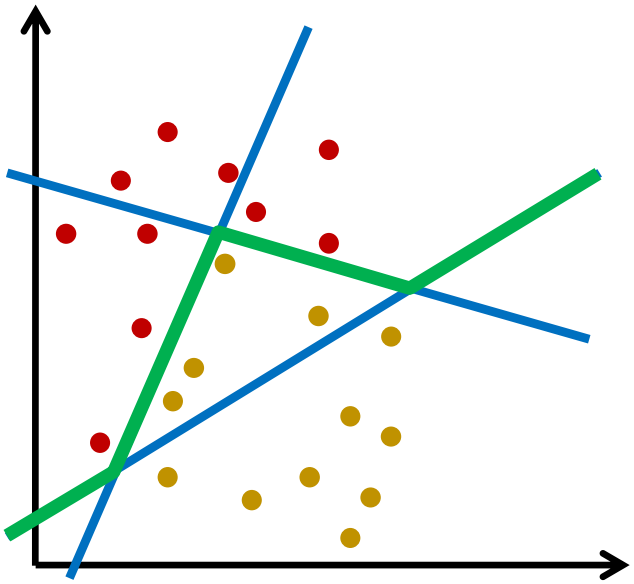
- What do you do if you want to solve a classification problem?

  – *create an expert: train a classifier*



  – *train several classifiers*

    → *and build an ensemble*



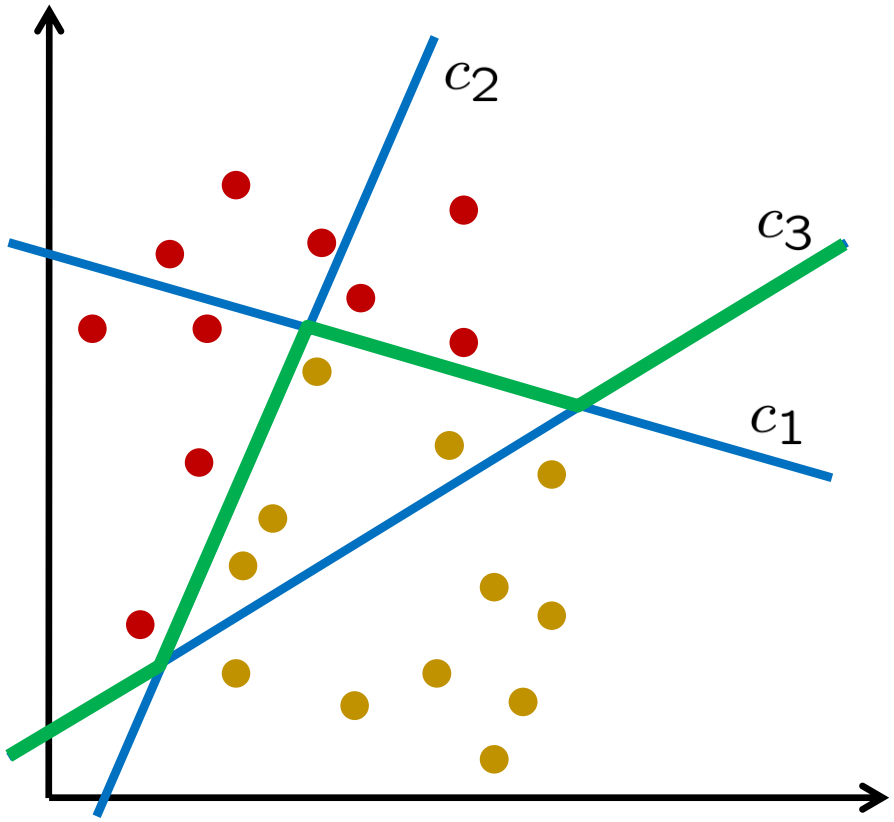
Ensembles cont.

- How do ensembles work?
  - *k* classifiers $c_1, c_2, \ldots, c_k$
  - applying same pattern to all classifiers → *k* predictions

$$c_1(\vec{x}) \in \{-1, +1\},$$
$$c_2(\vec{x}) \in \{-1, +1\},$$
$$\ldots,$$
$$c_k(\vec{x}) \in \{-1, +1\}$$

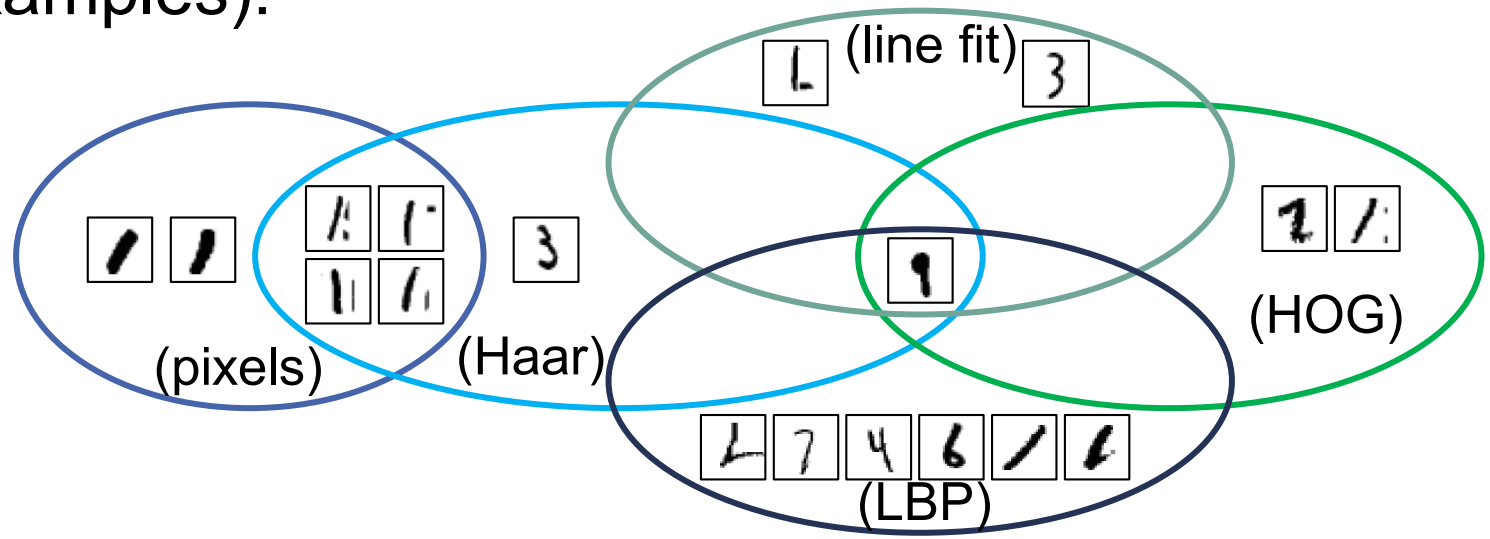
  - sum up all predictions and compare with zero:

$$ensemble(\vec{x}) = sign\Big(\sum_{j=1}^{k} c_j(\vec{x})\Big)$$



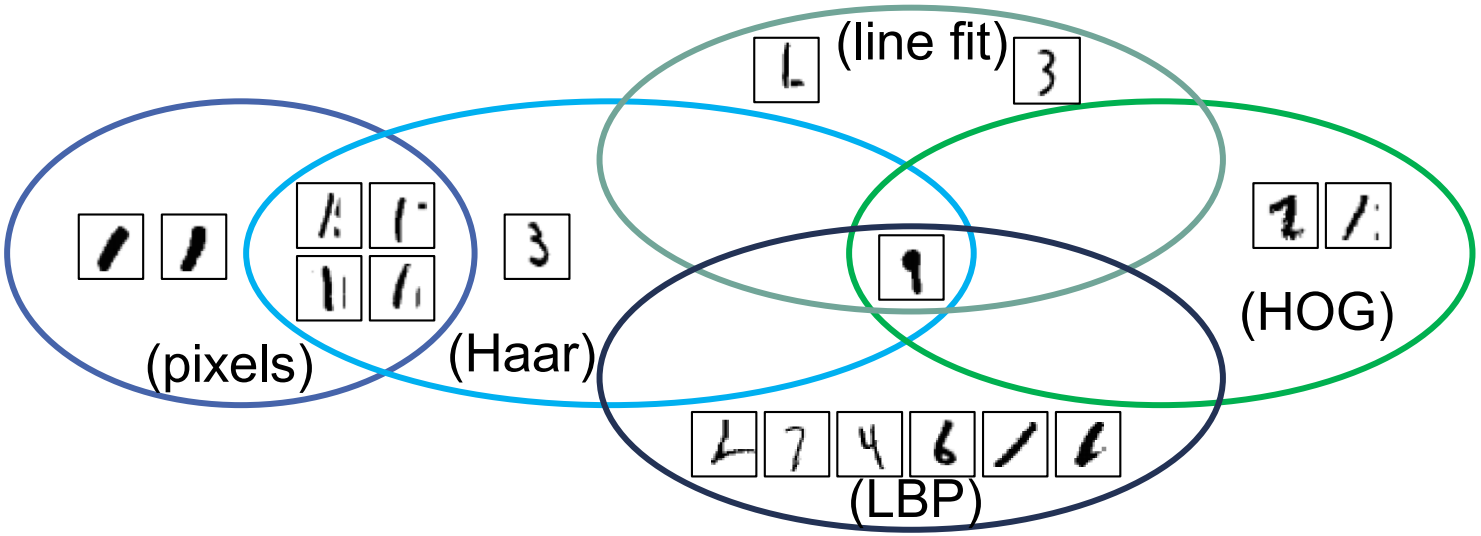
Example: Digit Recognition

- Best four approaches of digit recognition (cf. slide 62)
  - (2) „line fit“ feature: 99.7%
  - (3) pixel values: 99.4%
  - (4) HOG features: 99.7%
  - (5) Haar features: 99.4%
  - (6) LBP features: 99.3%
- Ensemble of these approaches, shared errors (out of 1000 test examples):



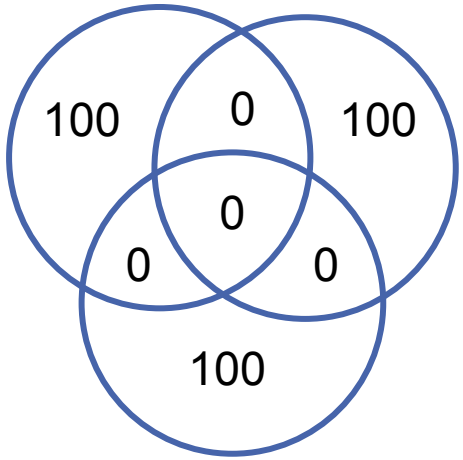
Example: Digit Recognition cont.

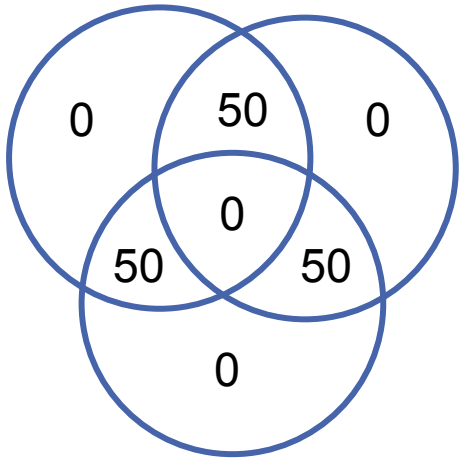
- Ensemble of these approaches:
  - *ensemble: line fit, pixels, Haar*
    - error rate: 5/1000
    - error of members: 3, 6, 6/1000
  - *ensemble: line fit, HOG, Haar*
    - error rate: 1/1000
    - error of members: 3, 3, 6/1000
    - error of SVM with joint features: 2/1000
  - *ensemble: line fit, HOG, LBP*
    - error rate 1/1000
    - error of members: 3, 3, 7/1000
    - error of SVM with joint features: 5/1000
  - *ensemle: all together*
    - error rate 1/1000



Boosting

- when is an ensemble beneficial?
  - *best case: classifiers don't share errors*
    - ensemble errors: 0
    - errors of each classifier: 100
  - *worst case: classifiers share all errors*
    - ensemble errors: 150
    - errors of each classifier: 100





- key idea to improve ensembles:
  - avoid classifiers sharing errors
- Boosting:
  - train classifiers depending on each other
  - classifier No. *n+1* should focus on the examples misclassified by classifiers No. *1,…,n*

Boosting cont.

- implementation ideas:
  - *weighted training patterns*

    introduce weight $\gamma_i \geq 0$ for each *training pattern* to model its importance

    → modification of training algorithms necessary, e.g. soft margin SVM:

$$\begin{aligned} \underset{\vec{w},b}{minimise} \quad & \frac{1}{2}||\vec{w}||^2 + C\sum_i(\gamma_i \cdot \xi_i) \\ subject\ to \quad & d^{(i)} \cdot (\langle \vec{x}^{(i)}, \vec{w} \rangle + b) \geq 1 - \xi_i \qquad \text{for all } i \\ & \xi_i \geq 0 \qquad \text{for all } i \end{aligned}$$

  - *how to determine pattern weights?*

    recalculate weights after training a classifier:
    - increase weight of misclassified patterns
    - decrease weight of well classified patterns

Boosting cont.

- implementation ideas:
  - *weighted voting*

    introduce weight $\beta_k \geq 0$ for each *classifier* to model its reliability

    → modification of voting scheme:

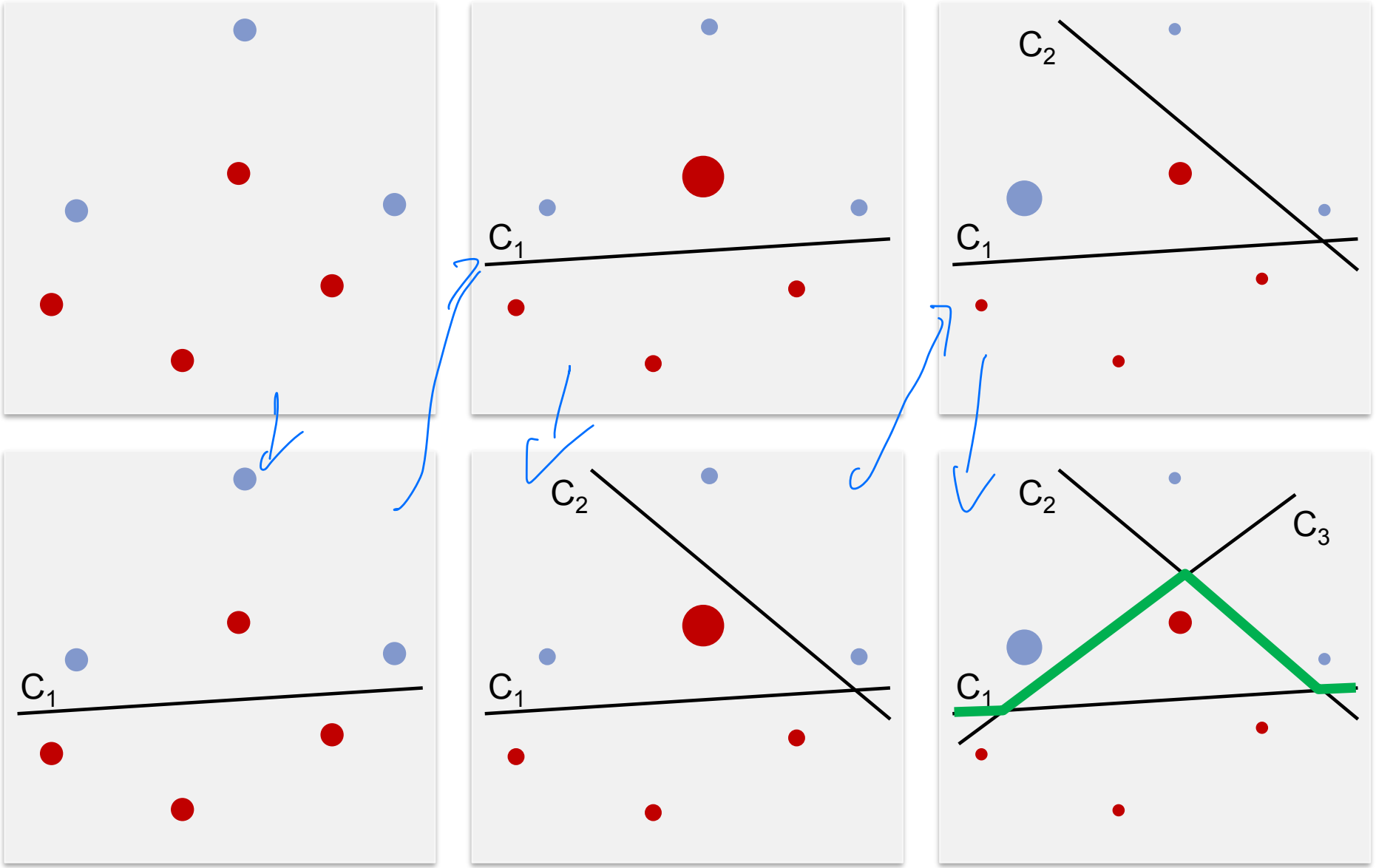
$$ensemble(\vec{x}) = sign\left(\sum_k \beta_k \cdot vote_k\right)$$

  - *how to determine the voting weights?*

    choose weight according to performance of classifier:
    - large weight for classifier with high accuracy
    - small weight for classifier with low accuracy

- Boosting适合数据集复杂且错误代价高的场景。
  - 随机森林更适合处理噪声较大的数据。

Boosting cont.



AdaBoost [Freund & Schapire, 1994]

1: initialize pattern weights $\gamma_i \leftarrow \frac{1}{n} \quad i \in \{1, \ldots, n\}$

2: **for** $k = 1, \ldots, T$ **do**

3: train new classifier c_k from weighted training patterns

4: calculate weighted training error

$$\epsilon_k := \sum_{i=1}^{n} \begin{cases} 0 & \text{if } c_k(\vec{x}^{(i)}) = d^{(i)} \\ \gamma_i & \text{otherwise} \end{cases}$$

5: set $\beta_k \leftarrow \frac{1}{2} \log \frac{1-\epsilon_k}{\epsilon_k}$

6: **for** $i = 1, \ldots, n$ **do**

7: $$\gamma_i \leftarrow \gamma_i \cdot \begin{cases} e^{-\beta_k} & \text{if } c_k(\vec{x}^{(i)}) = d^{(i)} \\ e^{\beta_k} & \text{if } c_k(\vec{x}^{(i)}) \neq d^{(i)} \end{cases}$$

8: **end for**

9: renormalize weights $\gamma_1, \ldots, \gamma_n$

10: **end for**

11: create ensemble $\vec{x} \mapsto \sum_{k=1}^{T} \frac{\beta_k}{\sum_{j=1}^{T} \beta_j} c_k(\vec{x})$

AdaBoost cont.

- properties of AdaBoost:
  - the training error of the ensemble is bounded above by:

$$\prod_{t=1}^{T} \left(2\sqrt{\epsilon_t(1-\epsilon_t)} \right) \leq \exp\left\{ -2 \sum_{t=1}^{T} (\frac{1}{2} - \epsilon_t)^2 \right\}$$

  - if all $\epsilon_t \leq \lambda < \frac{1}{2}$ and $T \to \infty$ AdaBoost yields a perfect classifier

Haar Classifier

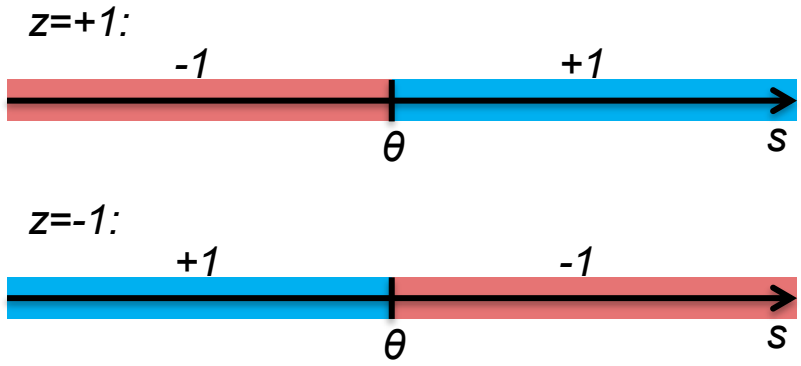
Haar 特征计算

- Haar feature:

$$s = \frac{1}{N_{red}} \sum_{(u,v) \in red\ area} g(u,v) \quad - \frac{1}{N_{blue}} \sum_{(u,v) \in blue\ area} g(u,v)$$

- make it a classifier:

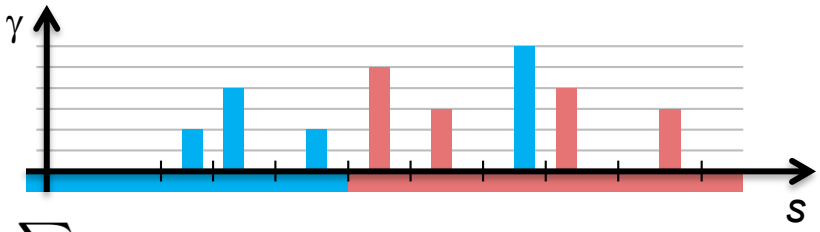
$$c(s) = sign(z \cdot (s - \theta))$$ 分类器构造



  - parameters:

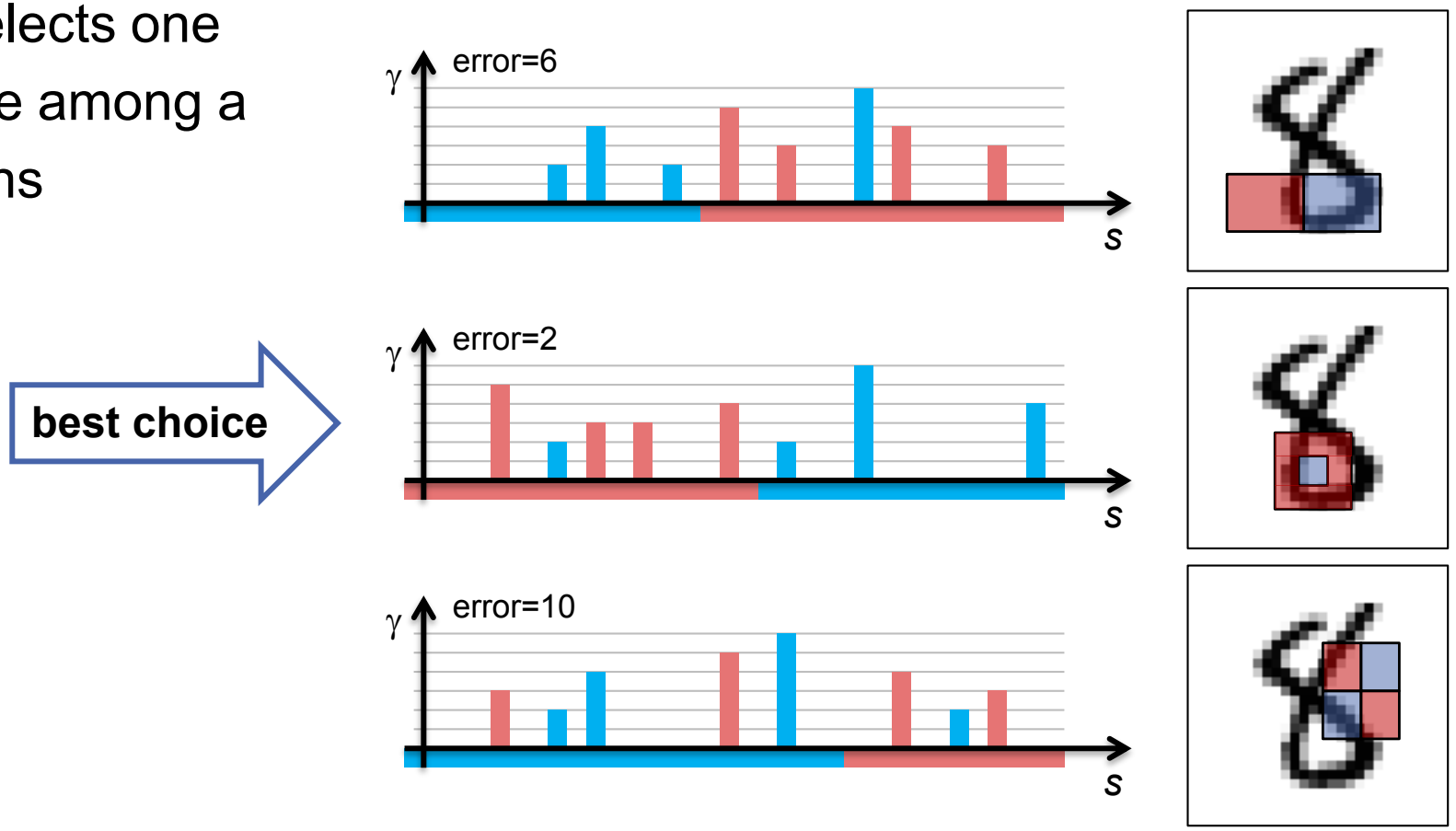
$\theta \in \mathbb{R}$	threshold
$z \in \{+1, -1\}$	orientation

- train the classifier from weighted examples:
  - try all possible values for θ and z
  - select values that minimize weighted error $\sum_{s_i < \theta, d^{(i)} = z} \gamma_i + \sum_{s_i > \theta, d^{(i)} = -z} \gamma_i$



Haar Classifier cont.

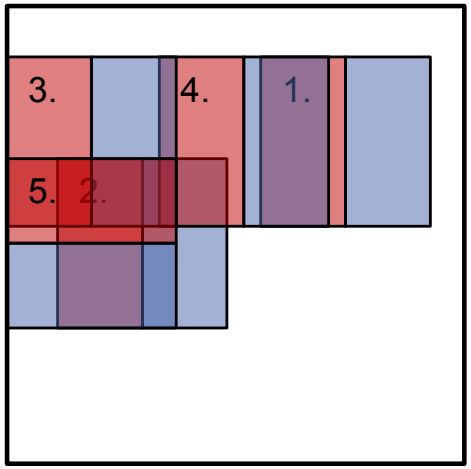
- Haar Classifier with multiple features:
  - classifier selects one Haar feature among a set of options

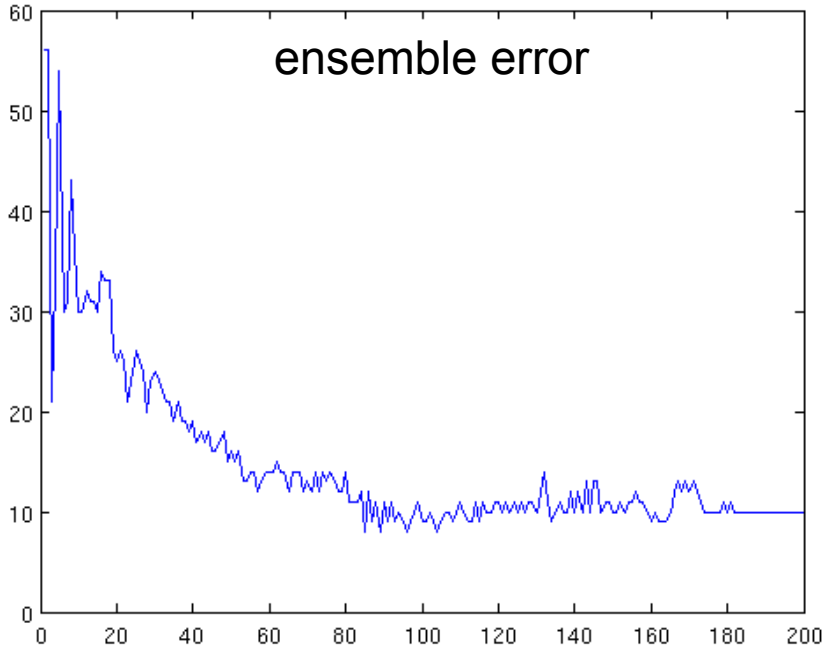


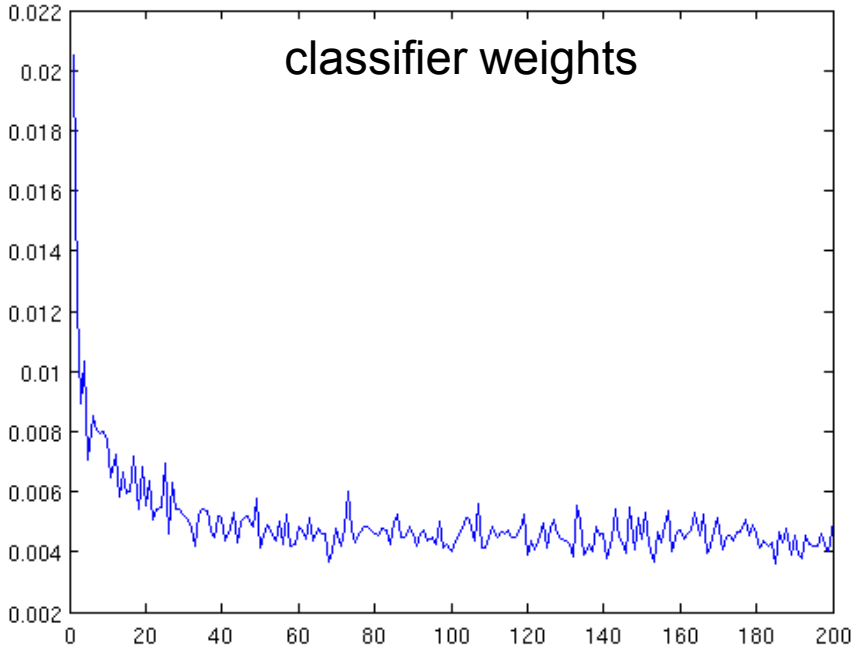
Boosting with Haar Classifiers

- Idea: use AdaBoost with Haar Classifiers
  - test error on digit recognition task:
    - first classifier: 56/1000
    - ensemble size 5: 54/1000
    - ensemble size 50: 16/1000
    - ensemble size 200: 10/1000

一般弱分类器 权重大

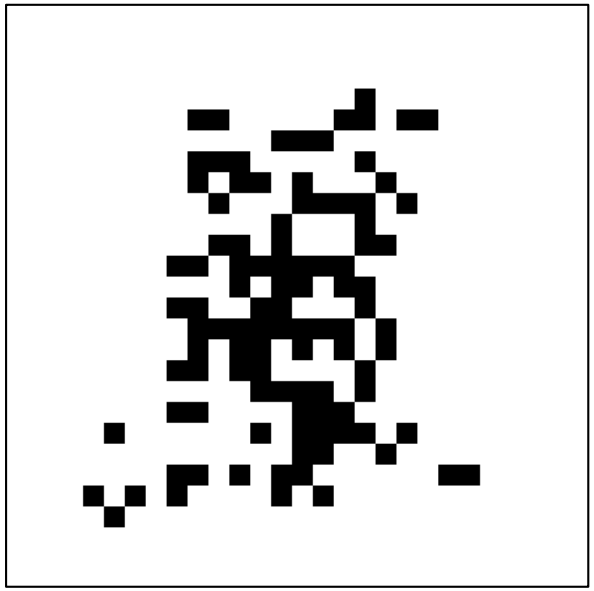


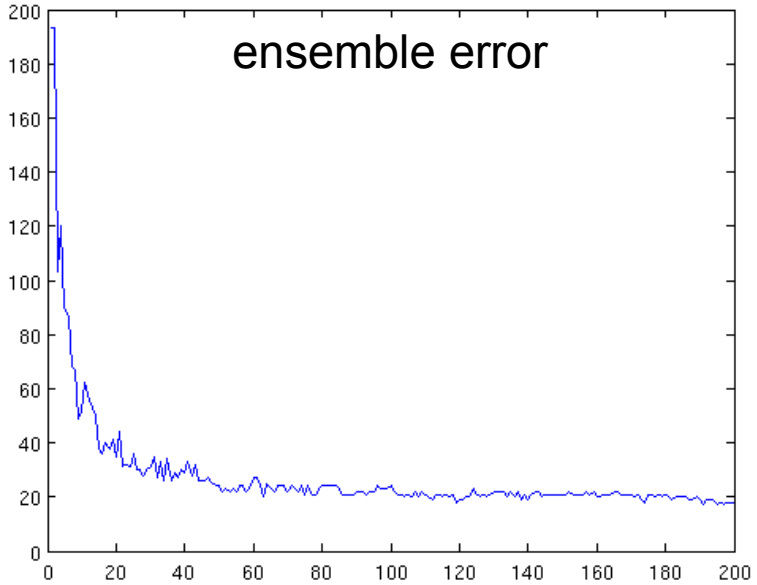


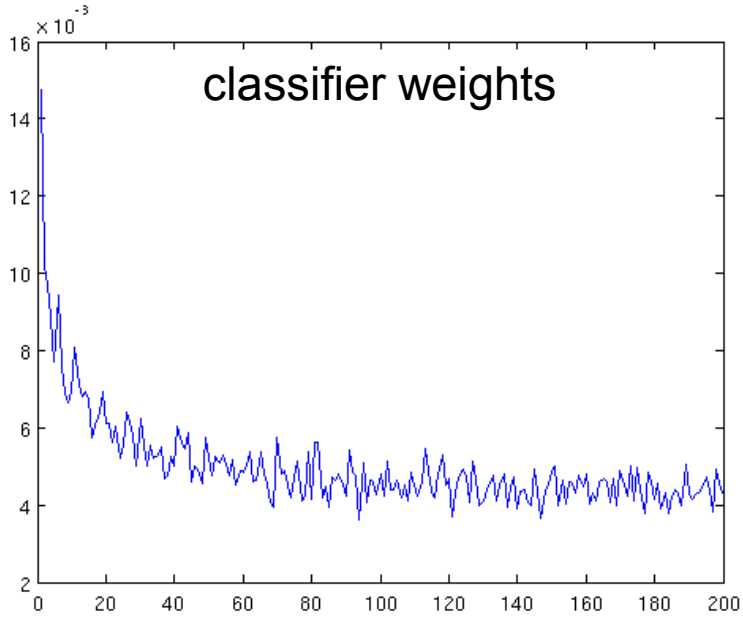


Boosting Other Classifiers

- Every feature induces a classifier
  - e.g. pixel gray levels
  - test error on digit recognition task:
    - first classifier: 193/1000
    - ensemble size 5: 90/1000
    - ensemble size 50: 24/1000
    - ensemble size 200: 18/1000







Balancing Errors

- Ensemble classifies:

$$\sum_k (\beta_k \cdot c_k(\vec{x})) \gtrless 0$$ → 进行 classification

- Extension:

$$\sum_k (\beta_k \cdot c_k(\vec{x})) \gtrless \delta$$

  - $\delta > 0$: classify positive only if you are very sure
  - $\delta < 0$: classify positive even if you are not that sure

Balancing Errors cont.

- Example
  - AdaBoost with Haar features
  - ensemble size 5 (c.f. slide 81)

$$\sum_k (\beta_k \cdot c_k(\vec{x})) \gtrless \delta$$

δ=0	is 1	is not 1
classified as 1	474	28
classified as not 1	26	472

high recall, small precision

δ=-0.5	is 1	is not 1
classified as 1	499	129
classified as not 1	1	371

high precision, small recall

δ=0.5	is 1	is not 1
classified as 1	387	5
classified as not 1	113	495

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

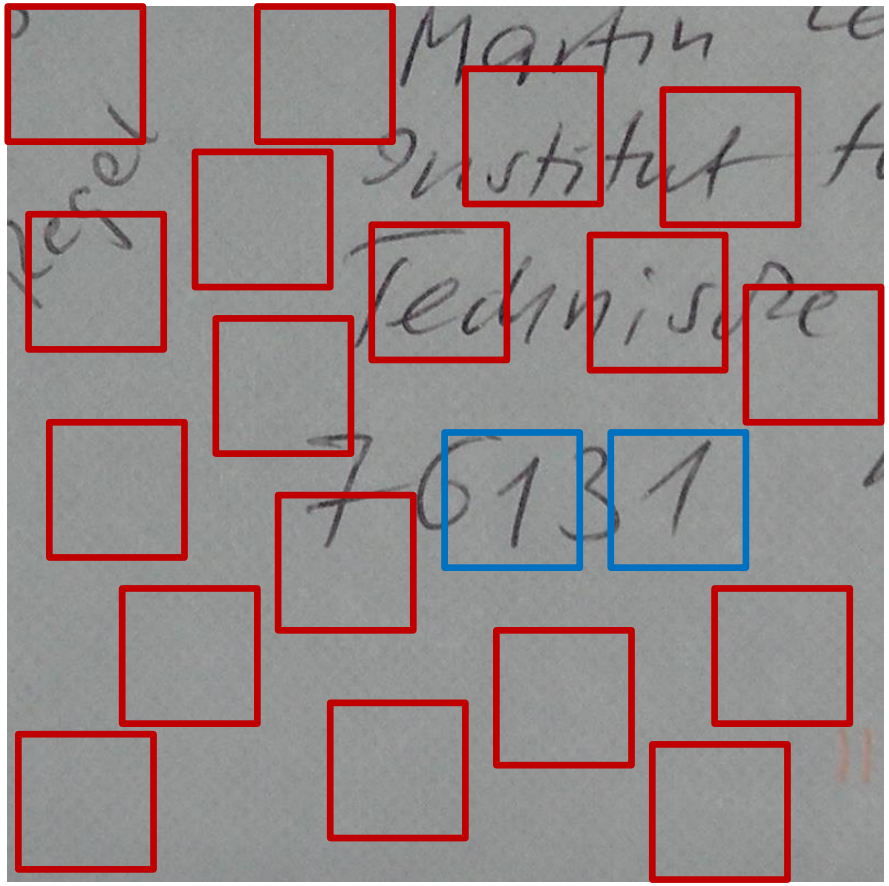
$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

Searching for Objects

- how can we find objects in an image with a classifier?
  - e.g. finding the digit “1” on a letter
  - using a classifier for “1”
- idea:
  - apply classifier to all possible areas in the image
    - vary position of area
    - vary size of area
    - vary orientation of area (optionally)

  → requires millions of trials

  efficiency?



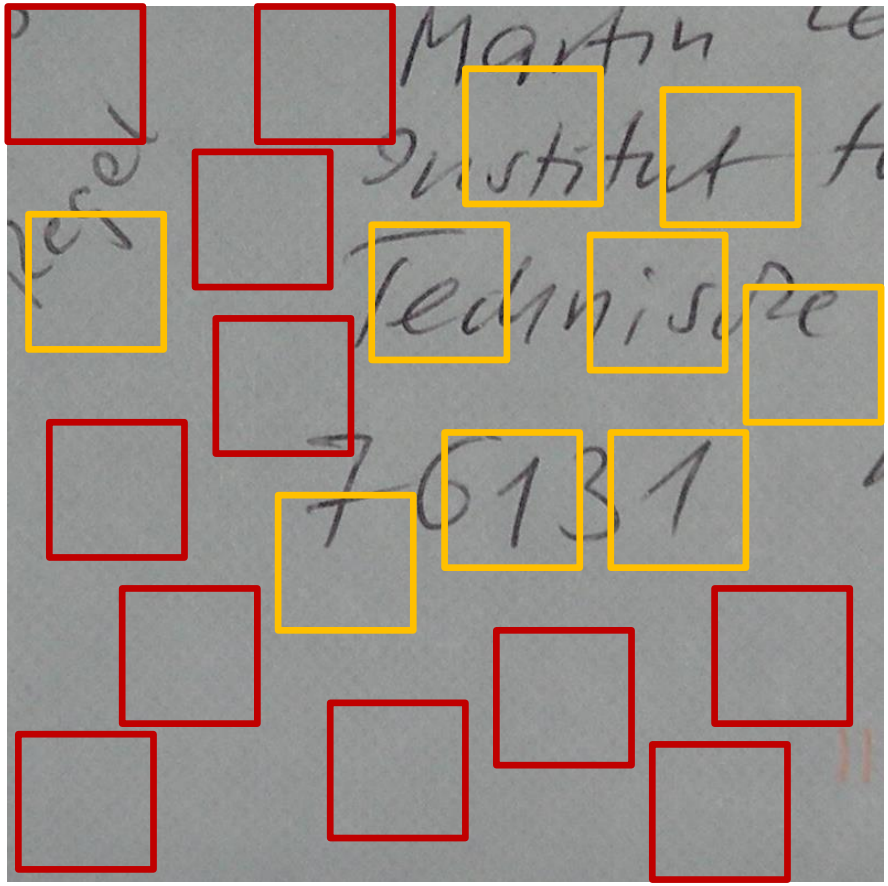
Searching for Objects cont.

- improved idea:

  use two classifiers

  – classifier 1
    - efficient
    - inaccurate
    - high recall
    - low precision
    - applied to all areas

  – classifier 2
    - inefficient
    - accurate
    - high recall
    - high precision
    - applied to areas which are found by classifier 1

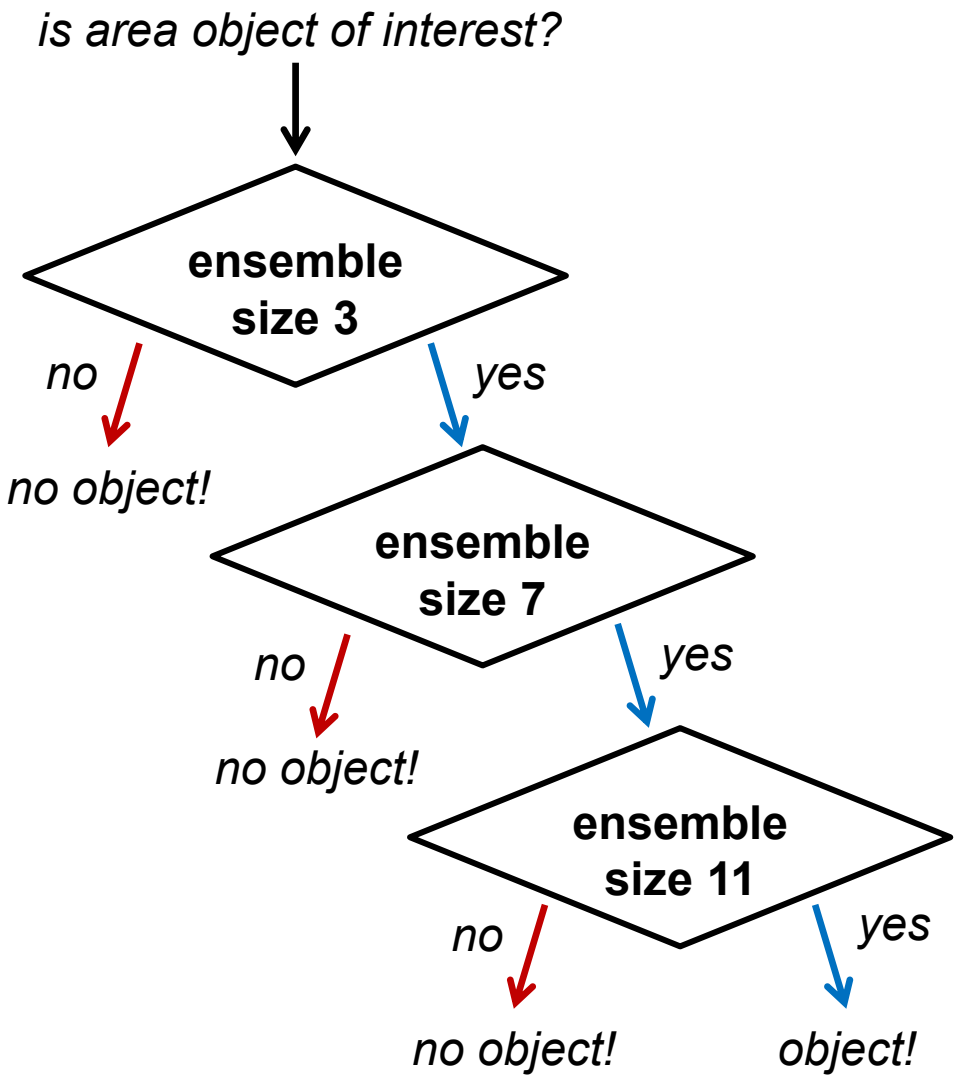


- idea can be extended to a series of many classifiers

  → approach of Viola/Jones

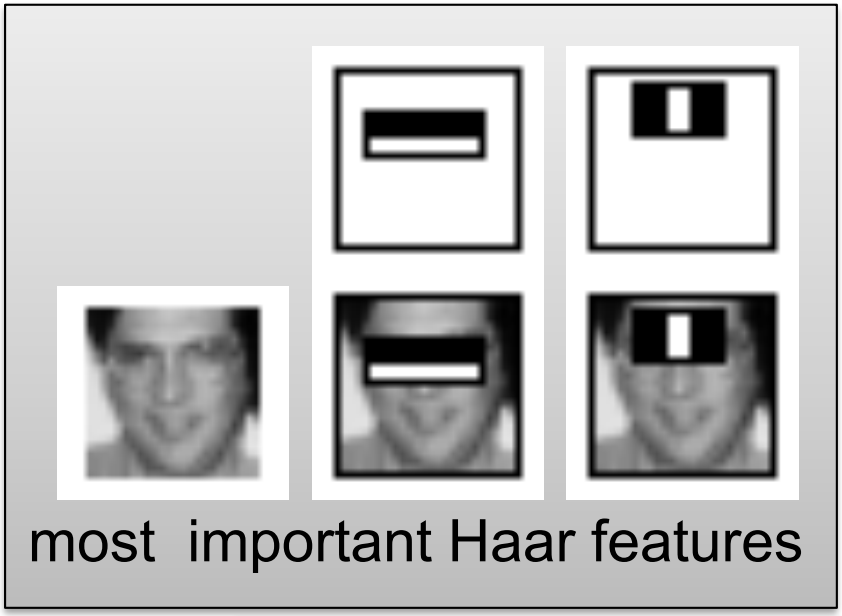
Viola/Jones Approach

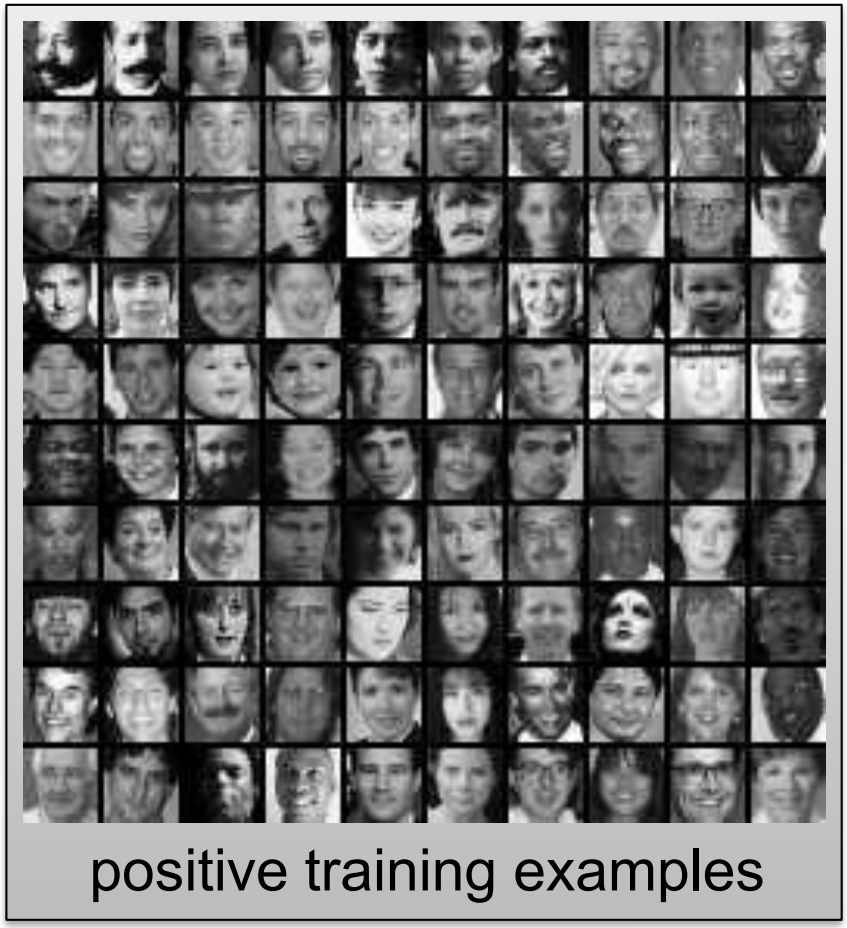
- combines:
  - Haar classifiers
  - AdaBoost
  - series (“cascade”) of classifiers of increasing ensemble size
  - tuning ensembles to maximize recall
  - search over the whole image with varying area position and size

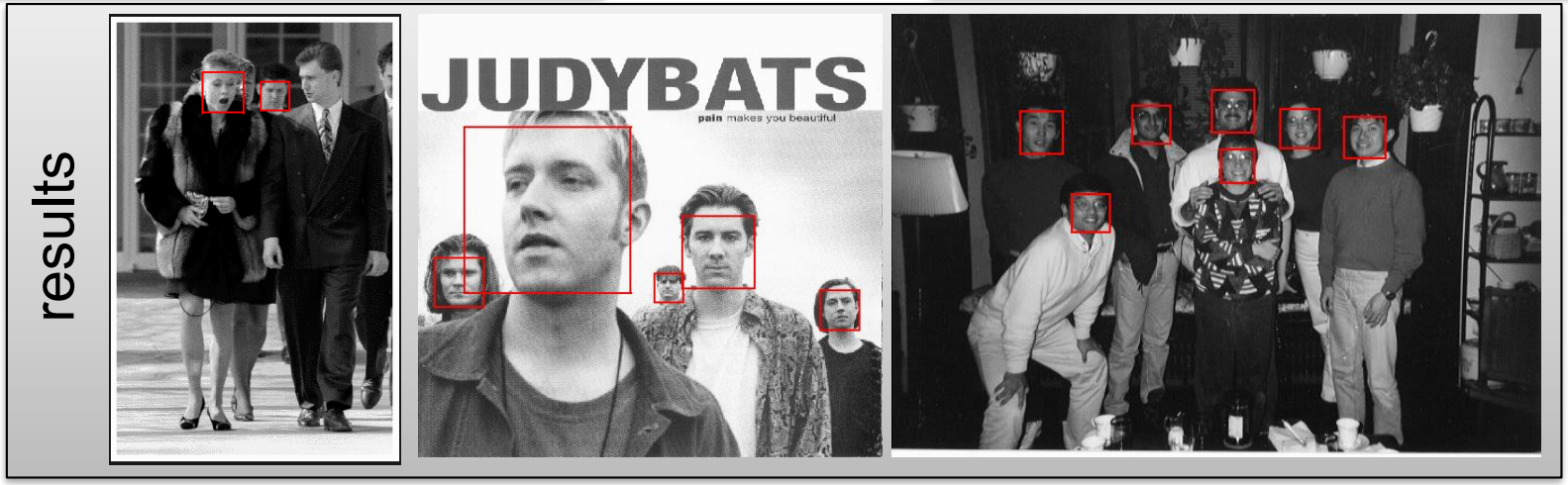


Example by Viola/Jones

- face recognition



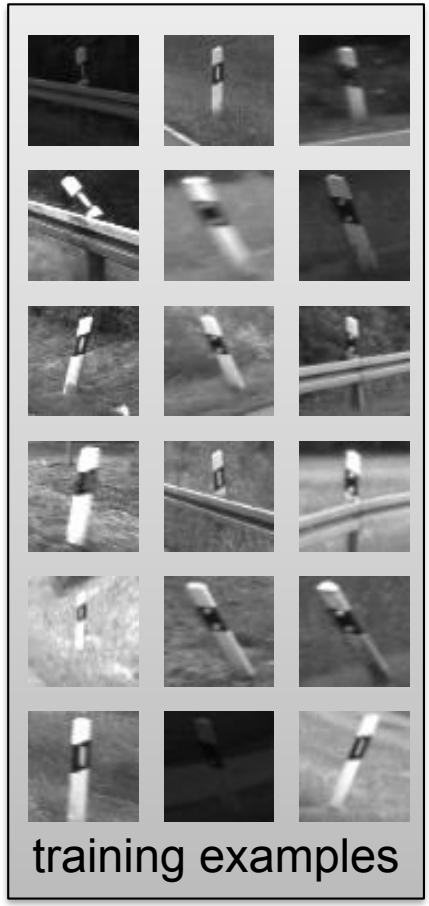


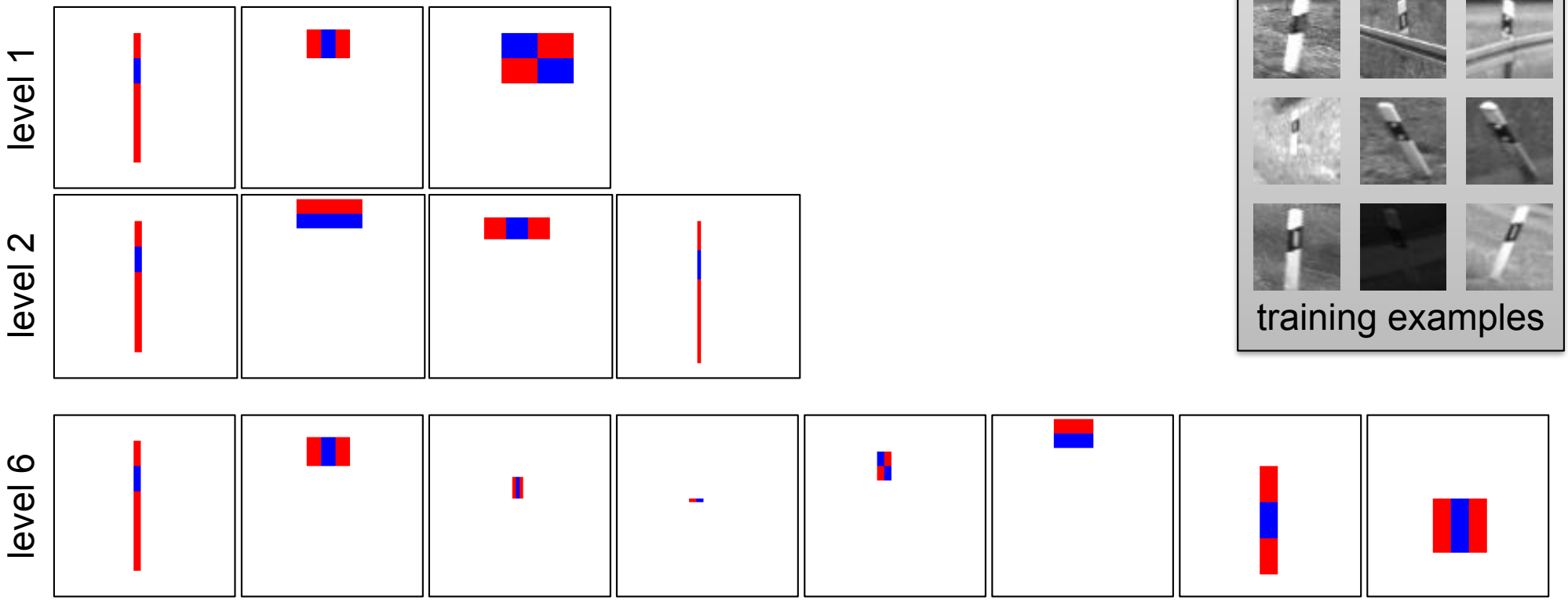


images source: P. Viola, M. Jones, *Rapid Object Detection using a Boosted Cascade of Simple Features*, CVPR 2001

Example: Detection of Delineators

- detection of delineators as visual landmarks
  - using Viola/Jones approach to find regions of interest
  - ensemble sizes: 3, 4, 3, 5, 9, 8, 11, 14, 20, 23
  - classification of regions of interest with SVM and HOG features

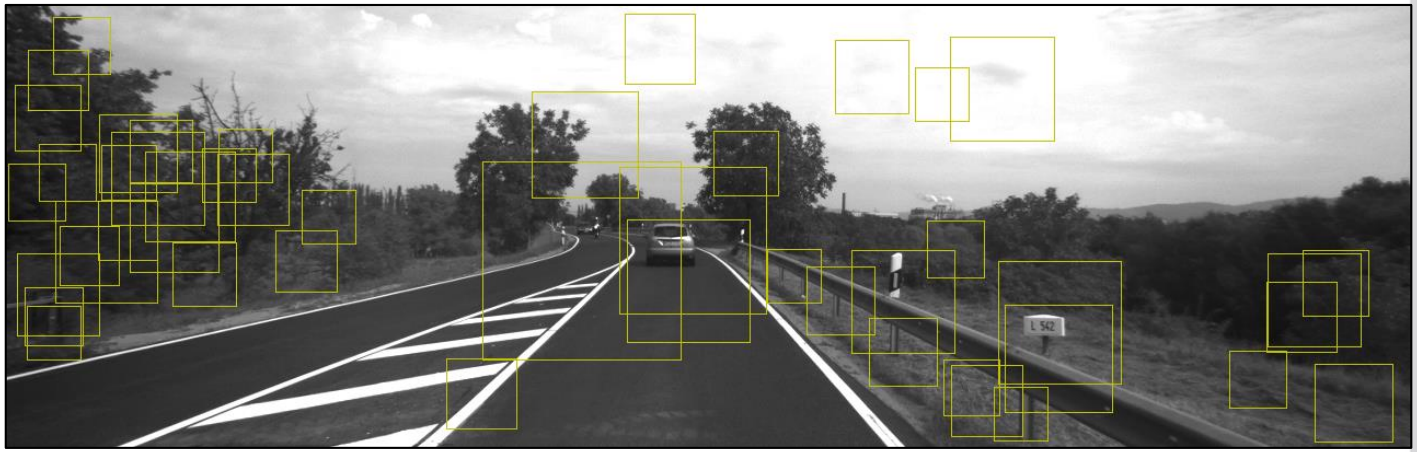




Example: Detection of Delineators cont.

depth of cascade = 4

average: 27 roi/frame



depth of cascade = 7

average: 10 roi/frame

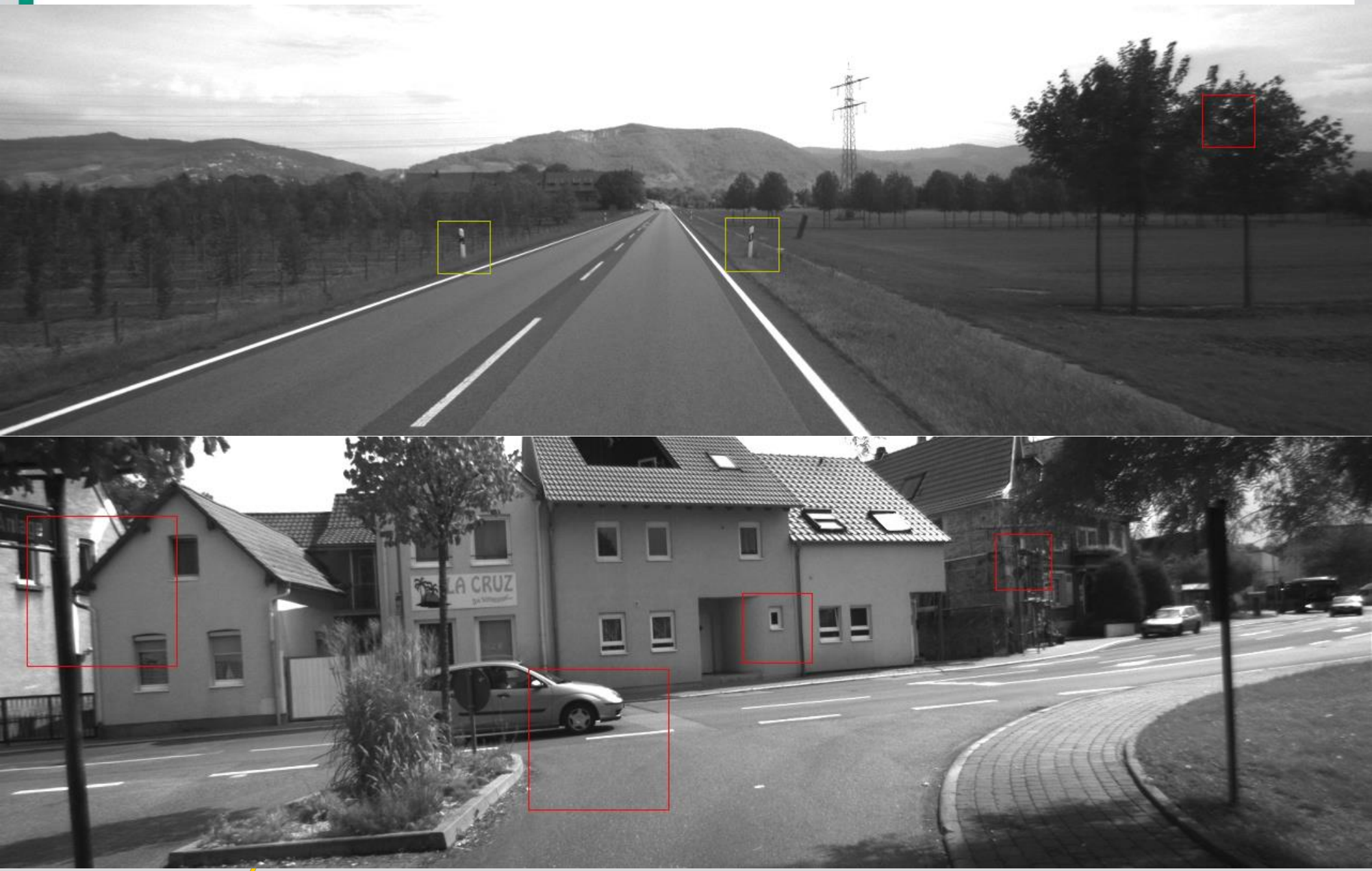
depth of cascade = 10

average: 1.6 roi/frame

Example: Detection of Delineators cont.

- performance on 1283x403 images
  - runtime approx. 55 ms/frame for Haar-classifier/detector
  - 0-10 roi per image found
  - SVM/HOG requires ≈ 1 ms/roi
  - test set accuracy of SVM >99%

Example: Detection of Delineators cont.

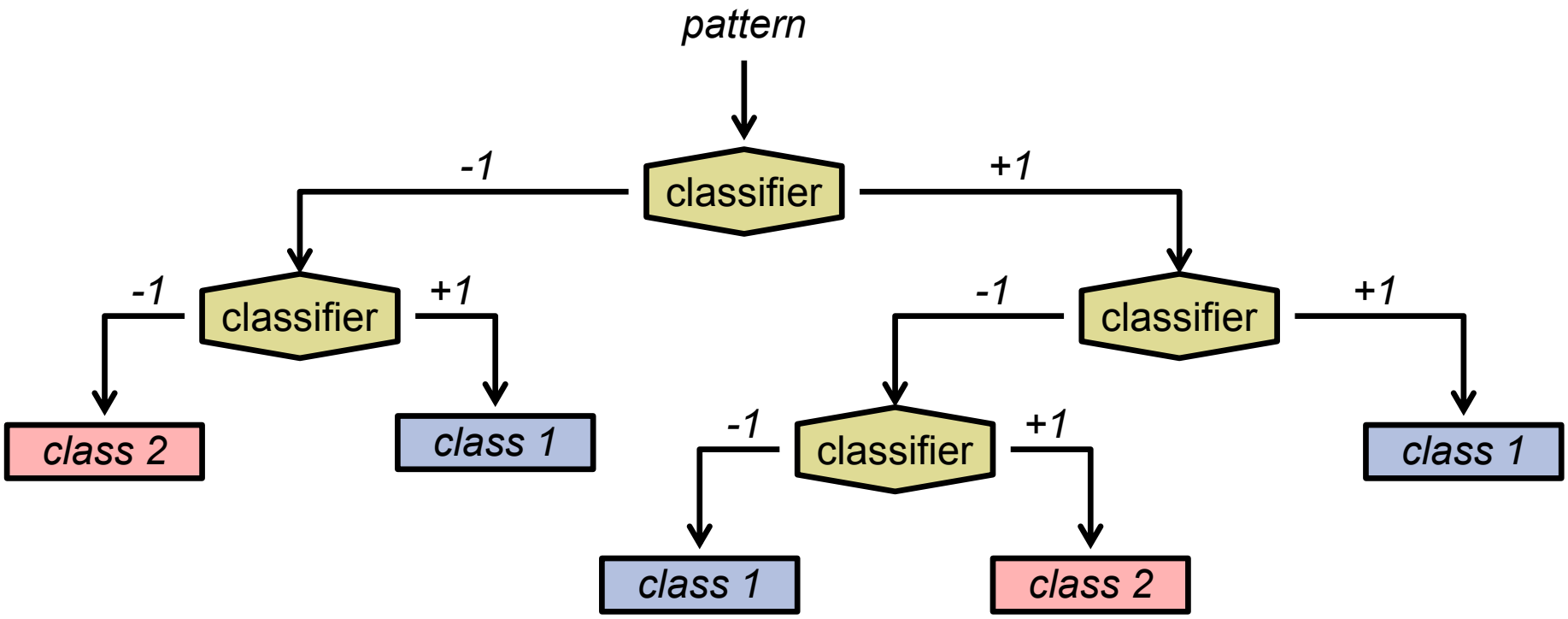


Example: Detection of Delineators cont.

DECISION TREES

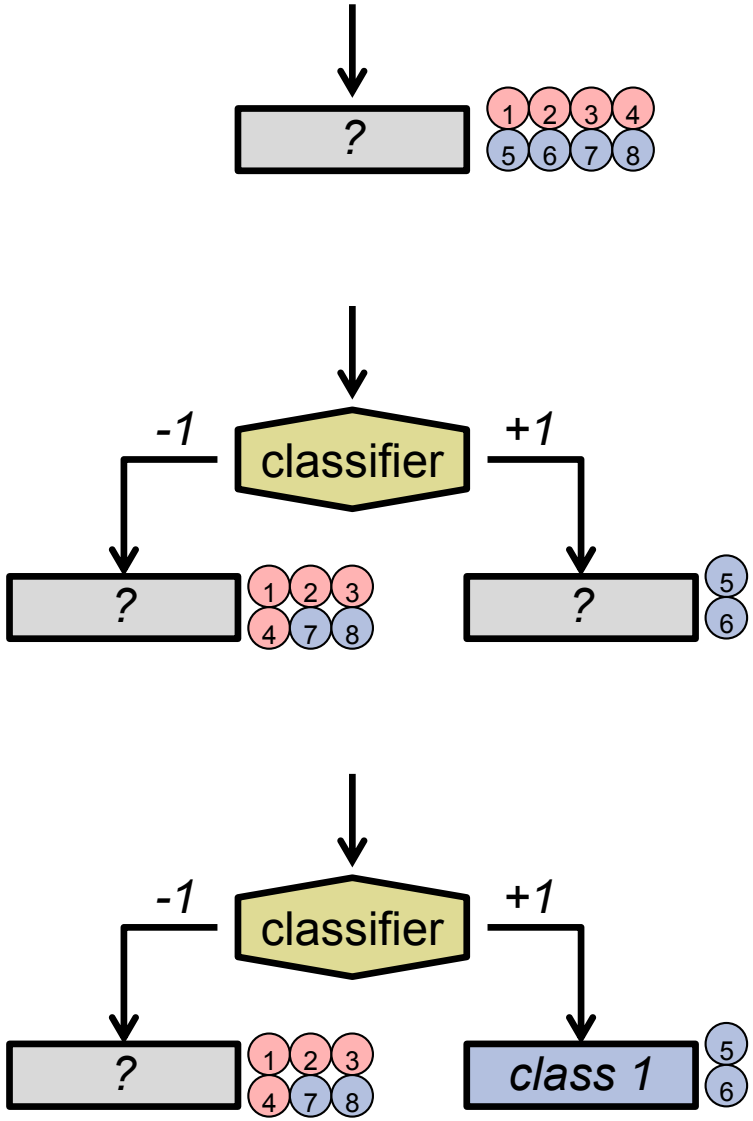
Decision Trees

- Decision Tree:
  - tree structure, branching factor 2
  - interior nodes: binary classifiers
  - leaf nodes: class labels



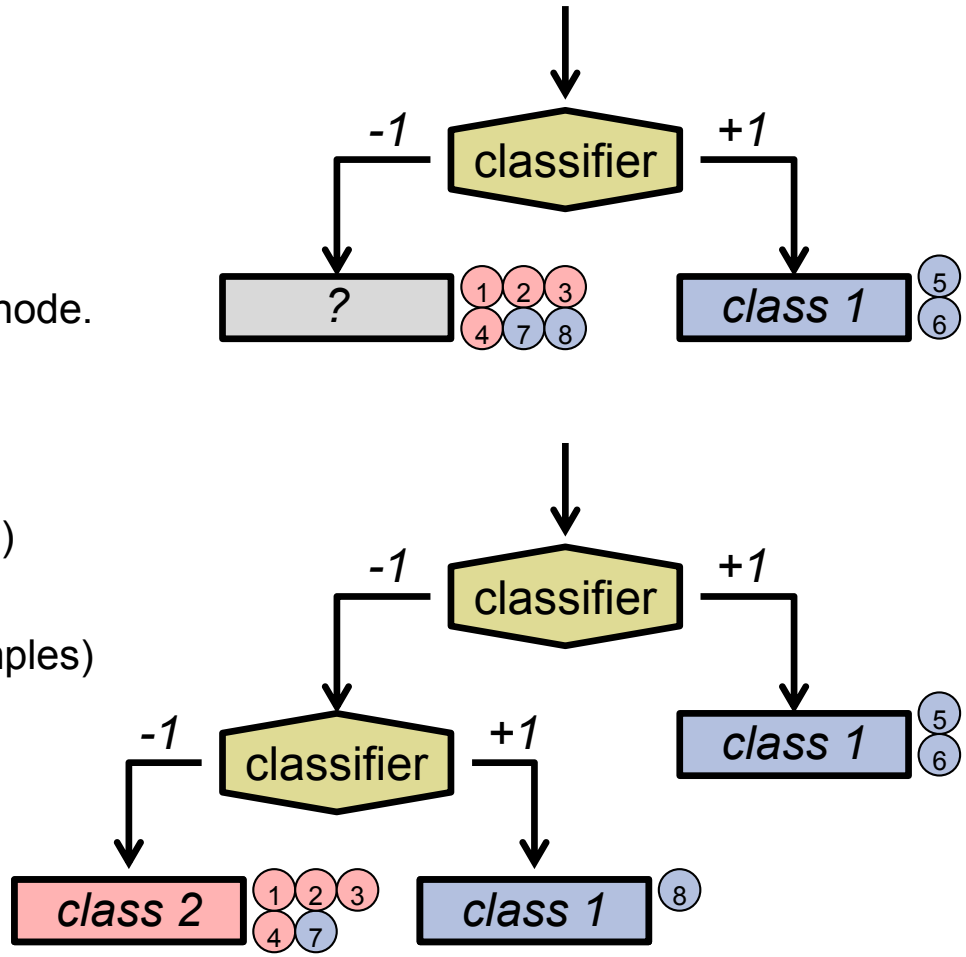
Decision Trees Training

- creating decision trees from training examples
  - create leaf node with unknown class label as root node. Assign all training examples to it
  - **while** (unlabeled leaf nodes exist)
  - select unlabeled leaf node *n*
  - **if** (num. pos. examples >> num. neg. examples)
  - assign pos. label to node *n*
  - **elseif (**num. pos. examples << num. neg. examples)
  - assign neg. label to node *n*
  - **else**
  - train new classifier
  - replace leaf node *n* by classifier node
  - create two unlabeled leaf nodes
  - classify examples and assign them to the leaf nodes
  - **endif**
  - **endwhile**



Decision Trees Training cont.

- creating decision trees from training examples
  - create leaf node with unknown class label as root node. Assign all training examples to it
  - **while** (unlabeled leaf nodes exist)
  - select unlabeled leaf node *n*
  - **if** (num. pos. examples >> num. neg. examples)
  - assign pos. label to node *n*
  - **elseif (**num. pos. examples << num. neg. examples)
  - assign neg. label to node *n*
  - **else**
  - train new classifier
  - replace leaf node *n* by classifier node
  - create two unlabeled leaf nodes
  - classify examples and assign them to the leaf nodes
  - **endif**
  - **endwhile**



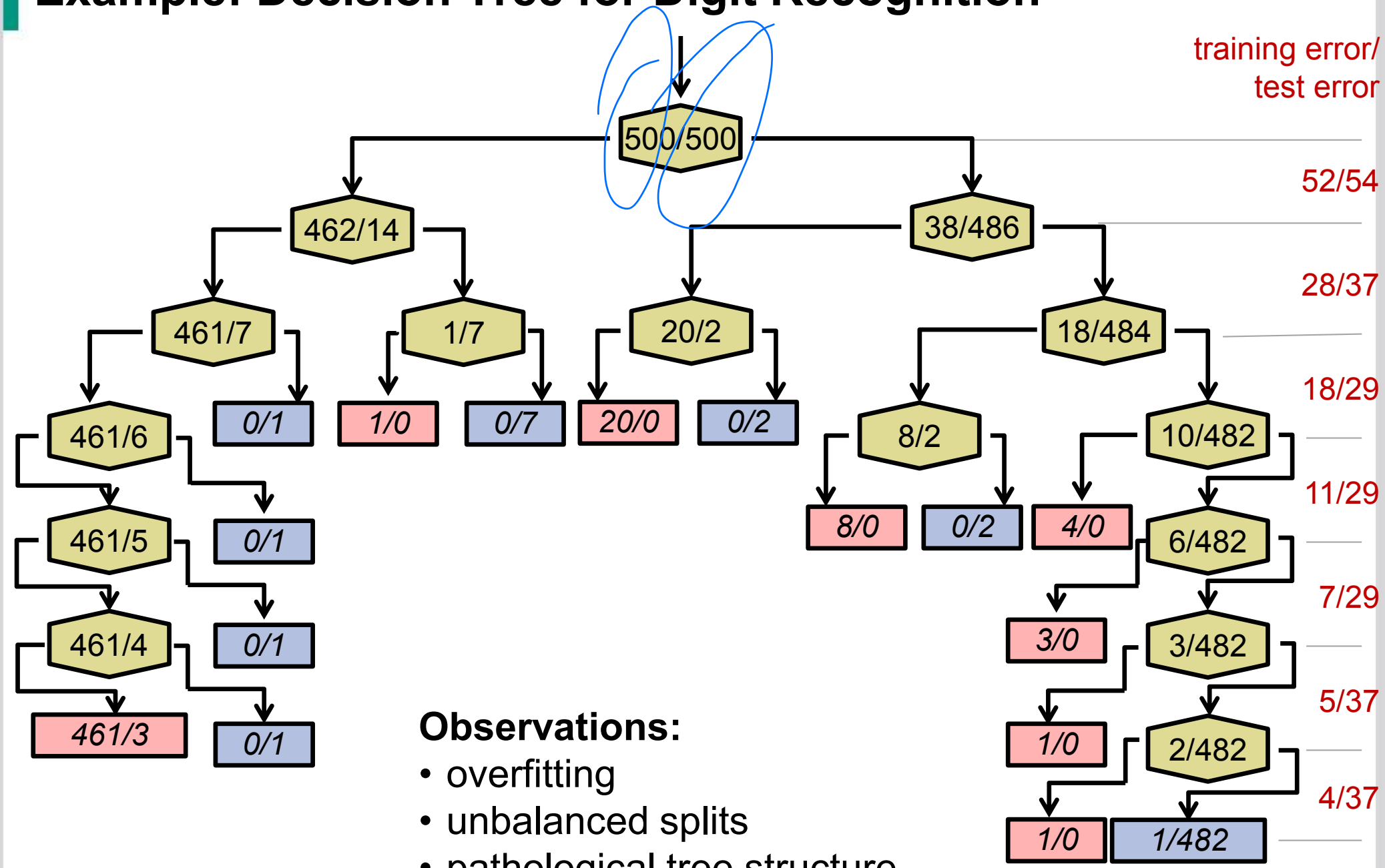
Decision Tree with Threshold Classifier

- Which classifiers are appropriate?
  - in general: all
  - similar idea like boosting:

    create a complex classifier by combining simple classifiers, i.e. threshold classifiers
- Example: digit recognition with Haar classifiers

  (next slide)

Example: Decision Tree for Digit Recognition



Observations:

- overfitting
- unbalanced splits
- pathological tree structure

Techniques to Improve Decision Trees

- Regularization techniques:
  - *Early Stopping*

    use a validation set while building the tree. Stop splitting nodes when you observe non-decreasing error on the validation set.

    Example: the validation error for digit recognition was:

    54 for tree of depth 1

    37 for tree of depth 2

    29 for tree of depth 3

    29 for tree of depth 4

    29 for tree of depth 5

    37 for tree of depth 6

    37 for tree of depth 7

  → take tree with depth 3

Techniques to Improve Decision Trees cont.

– *Pruning*

create complete decision tree first. Remove unbalanced or pathological branches afterwards.

- several pruning criteria
- implemented e.g. in decision tree algorithm C4.5

Techniques to Improve Decision Trees cont.

– *Randomized Decision Trees and Forests*

Randomize decision tree creation by:

- randomly selecting a subset of the training data
- randomly selecting a subset of features which serve as options for the next split
- randomly selecting the threshold for discrimination

Build an ensemble of many randomized trees

→ *Random Decision Forest*

Example: Decision Forest for Digit Recognition

- Building decision forests:
  - using Haar features
  - randomly selecting feature and threshold (best among *k* trials)
  - no variation of training set
  - allowing deep trees
  - varying the ensemble size *n*

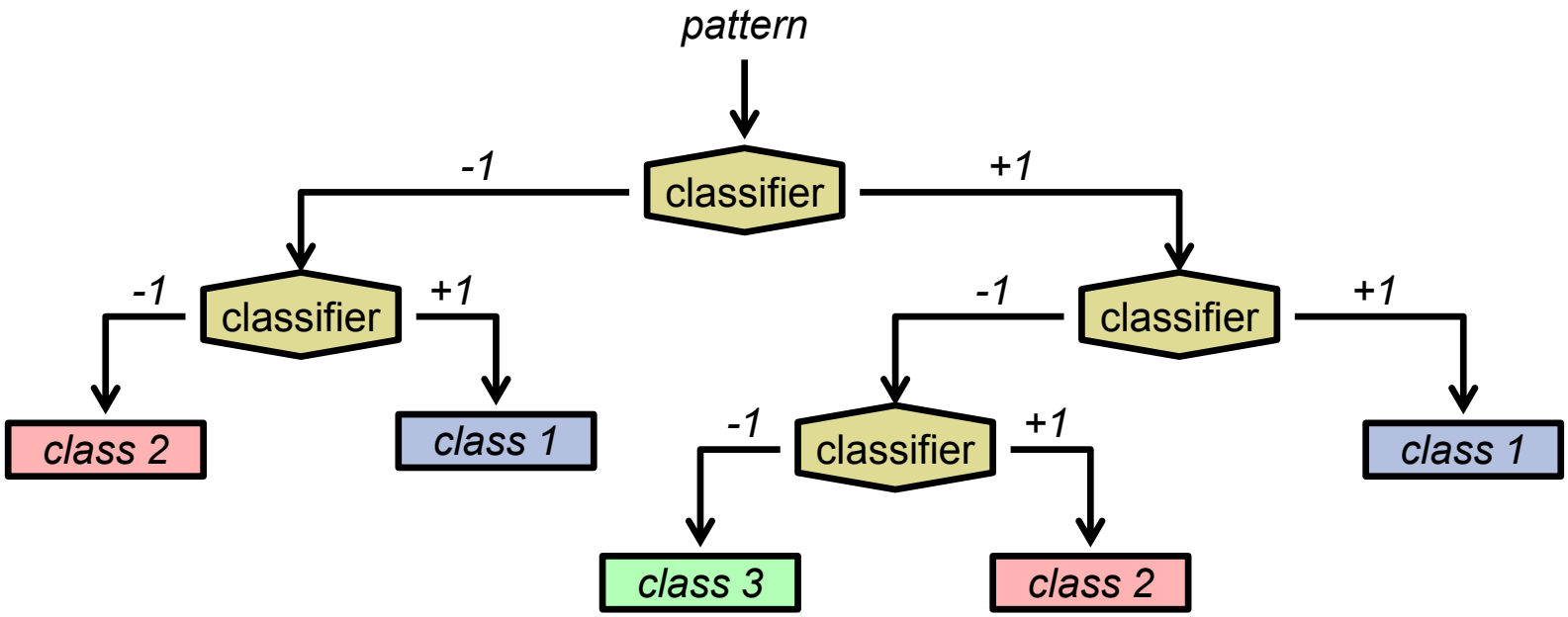
- Error on test set:

	n=1	n=10	n=30	n=80
k=1	336	398	374	392
k=10	90	53	36	32
k=30	43	29	20	17
k=80	41	25	15	11

comparison:
- decision tree trained with early stopping: 29
- AdaBoost ensembles
  - size 5: 54
  - size 50: 16
  - size 200: 10
- SVM: 6

Multi-Class-Classification with Decision Trees

- extension to more than two classes



– classifiers are trained to minimize Shannon entropy in leaf nodes

$$-\sum_{leaf\ node\ n}\left(\frac{N_n}{N}\sum_{class\ c}\left(\frac{N_n^c}{N_n}\log_2\frac{N_n^c}{N_n}\right)\right)$$

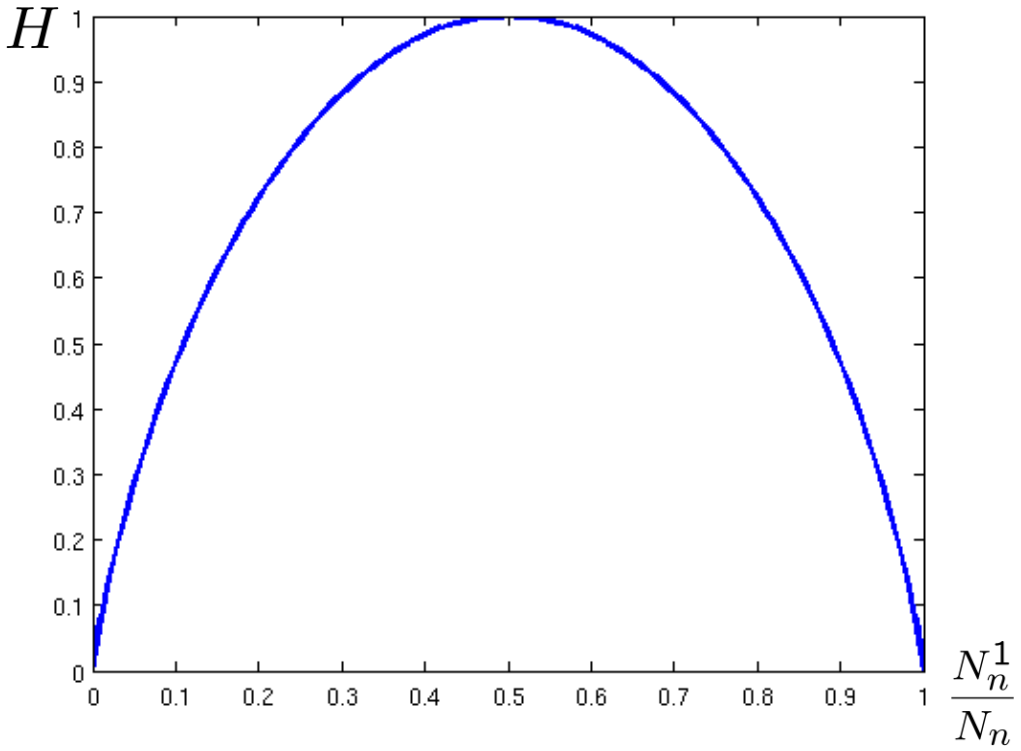
ratio of training patterns assigned to leaf node *n*

ratio of training patterns belonging to class *c* among those assigned to leaf node *n*

Multi-Class-Classification with Decision Trees cont.

- Entropy measures the homogeneity of a pattern set
  - all patterns belong to the same class: entropy minimal (0)
  - same amount of patterns belongs to each class: entropy maximal

entropy in the 2-class case



7. 错误分析

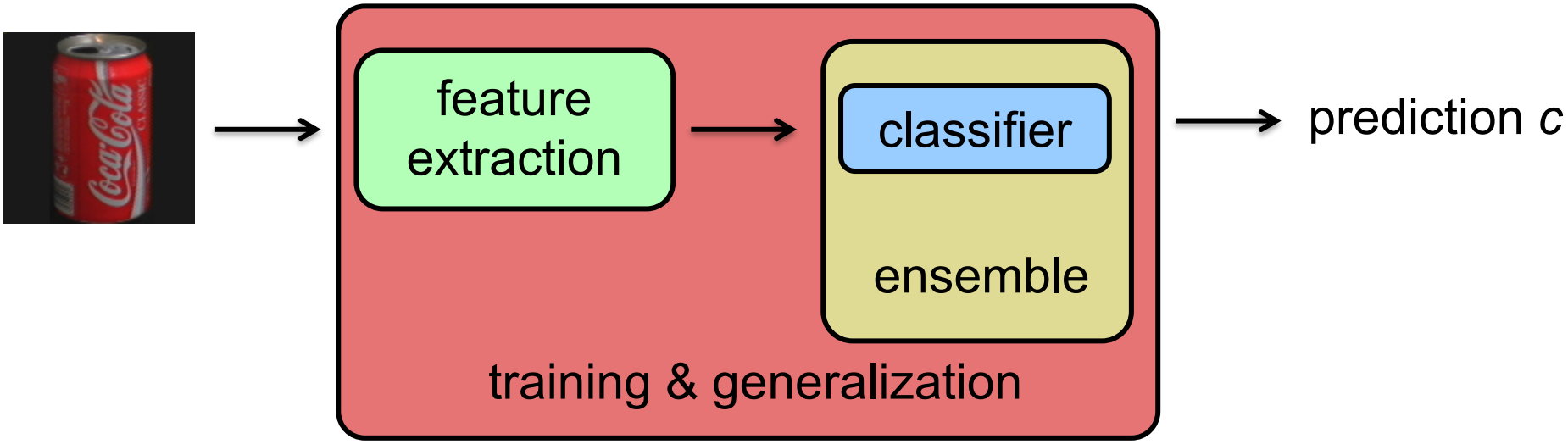
- **常见误分类原因有哪些?**
  - 类别间特征相似度高。
  - 特征提取不充分，未捕捉到关键信息。
  - 数据集中存在噪声或标签错误。
- **如何优化假阳性和假阴性之间的平衡?**
  - 调整分类器的决策阈值。
  - 使用加权损失函数，针对错误代价不同的类别进行优化。

10. 模型比较

- **何时选择 HOG 特征优于 LBP 特征?**
  - HOG 适合对纹理和形状敏感的任务 (如人体检测)。
  - LBP 适合对局部模式变化敏感的任务 (如纹理分类)。
- **如何在提高模型精度的同时控制计算复杂度?**
  - 简化特征提取 (如减少分辨率)。
  - 使用轻量化模型 (如剪枝、蒸馏)。
  - 优化推理过程 (如量化、加速算法)。

SUMMARY: PATTERN RECOGNITION

Pattern Recognition: the Complete Picture



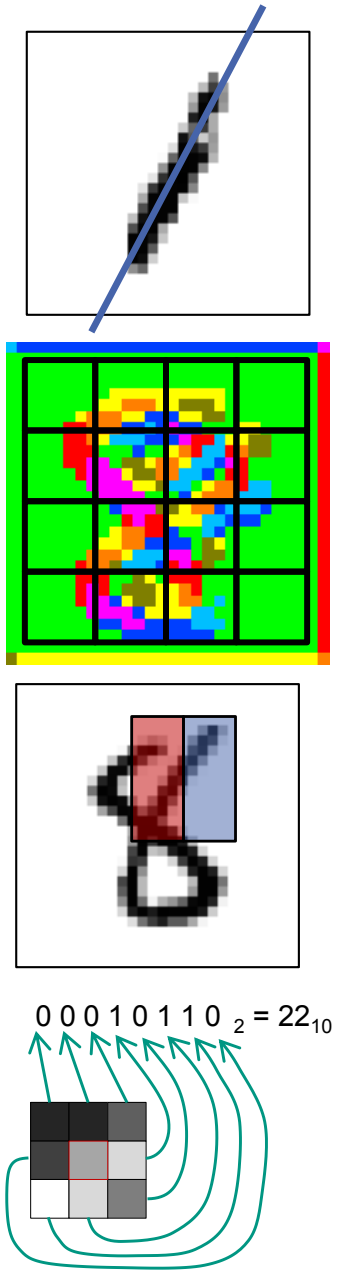
features	classifiers	ensembles	generalization
▪ smart (specific) ▪ HOG ▪ Haar ▪ LBP ▪ neural features ▪ …	▪ SVM ▪ threshold ▪ decision tree ▪ cascade ▪ neural network ▪ …	▪ free ensemble ▪ boosting ▪ decision forest ▪ multi-class ▪ …	▪ validation ▪ cross-validation ▪ data tuning ▪ early stopping ▪ randomization ▪ …

Pattern Recognition: the Complete Picture cont.

features

- smart (specific)
- HOG
- Haar
- LBP
- neural features
- …

- problem-specific features
  - pre processing (segmentation, filtering, …)
  - describe shape, color, template similarity
- gray level features
  - image subareas
- HOG
  - orientation histograms
  - accurate results
- Haar
  - easy and efficient calculation, integral images
  - edge, line, center-surround, chessboard
- LBP
  - local graylevel structure
  - easy and efficient calculation
- neural features
  - cf. chapter 12

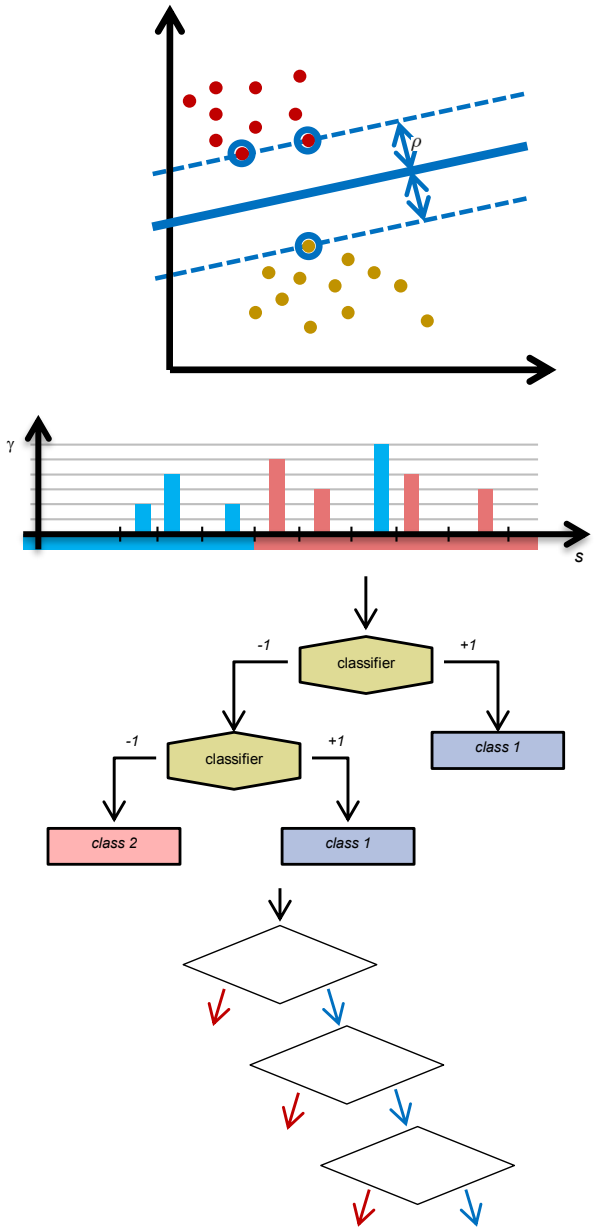


Pattern Recognition: the Complete Picture cont.

classifiers

- SVM
- threshold
- decision tree
- cascade
- neural network
- …

- support vector machines
  - maximal margin classifier
  - soft margin approach to allow errors
  - kernel approach to model non-linearity
- threshold classifier
  - little expressive power, but can be combined by boosting, decision trees
- decision trees
  - powerful classifier combining simple classifiers in tree structure
  - tends to overfit, requires regularization
- cascade
  - set of classifiers with increasing complexity and specivity
- artificial neural network
  - cf. chapter 12

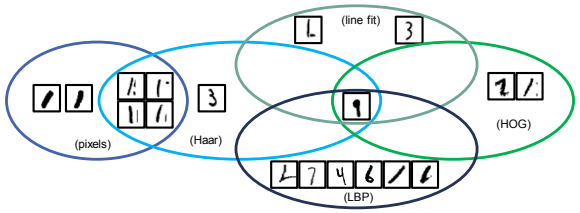


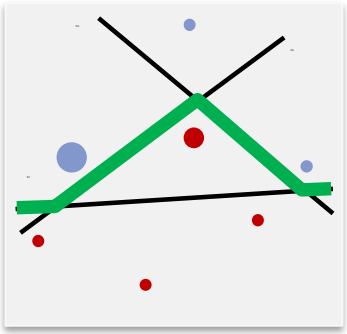
Pattern Recognition: the Complete Picture cont.

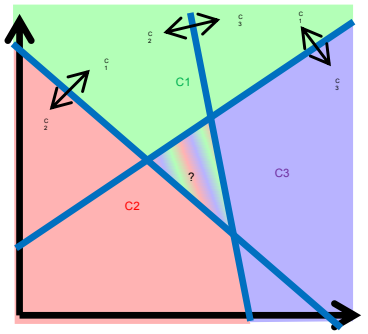
ensembles

- free ensemble
- boosting
- multi-class
- …

- Ensembles
  - combine “weak” classifiers to form a strong classifier by voting
  - independence & accuracy of members
- Boosting
  - train classifiers that compensate errors of other ensemble members
  - weighted training examples
  - weighted voting
  - AdaBoost
- Multi-class classification
  - 1-versus-the rest approach
  - 1-versus-1 approach





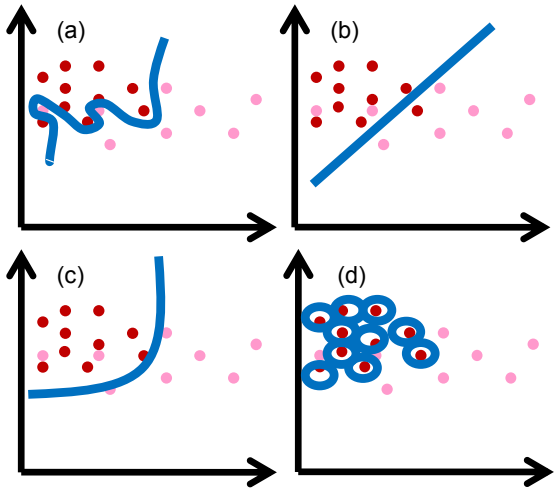


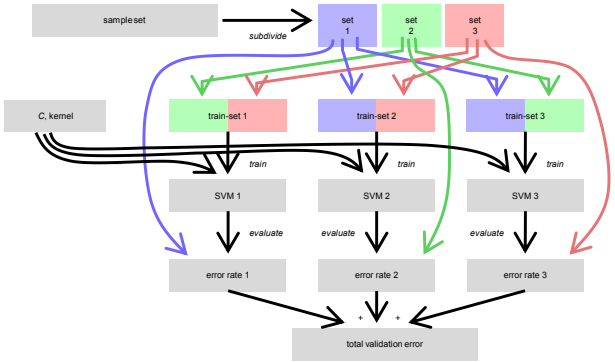
Pattern Recognition: the Complete Picture cont.

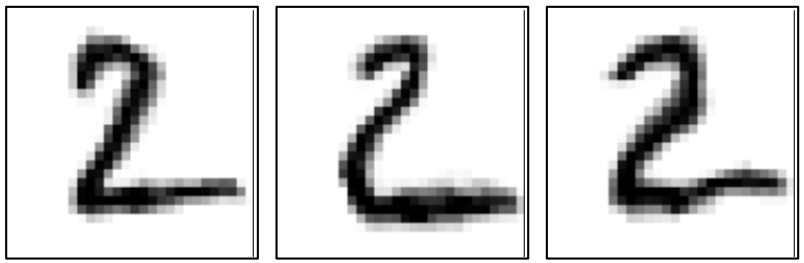
generalization

- validation
- cross-validation
- data tuning
- early stopping
- randomization
- …

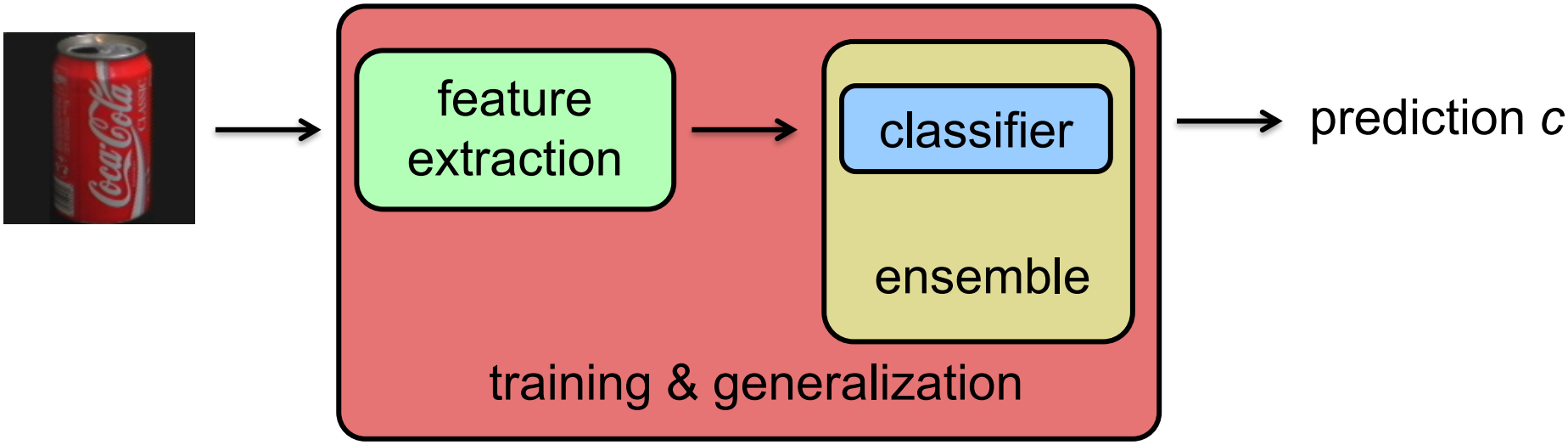
- generalization
  - don't memorize training examples, learn the basic concepts
  - minimization of test error
  - overfitting, underfitting
- validation techniques
  - standard validation
  - cross-validation
- regularization techniques
  - early stopping
  - randomization & ensembles
  - more techniques in chapter 12
- data tuning
  - extend data set
  - modify/distort examples
  - improve quality of data





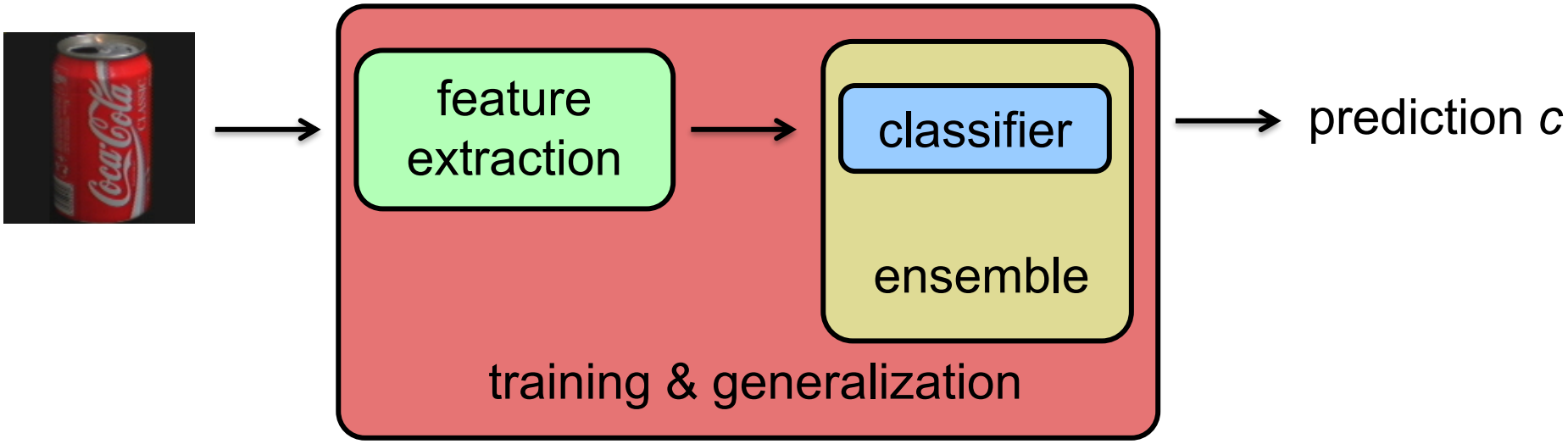


Pattern Recognition: the Complete Picture



- What matters most?
  1. good features
  2. as many training examples as possible
  3. consequent engineering of classification problem
  4. being aware of the generalization pitfalls
  5. type of classifier does not matter that much

Pattern Recognition: the Complete Picture



- Other techniques beyond the scope of the lecture
  - deformable parts models
  - nearest neighbors classifiers
  - learning vector quantization
  - …