Support Vector Machines *(SVM):* *an introduction*

Linear discriminant with margin

Support Vector Machines

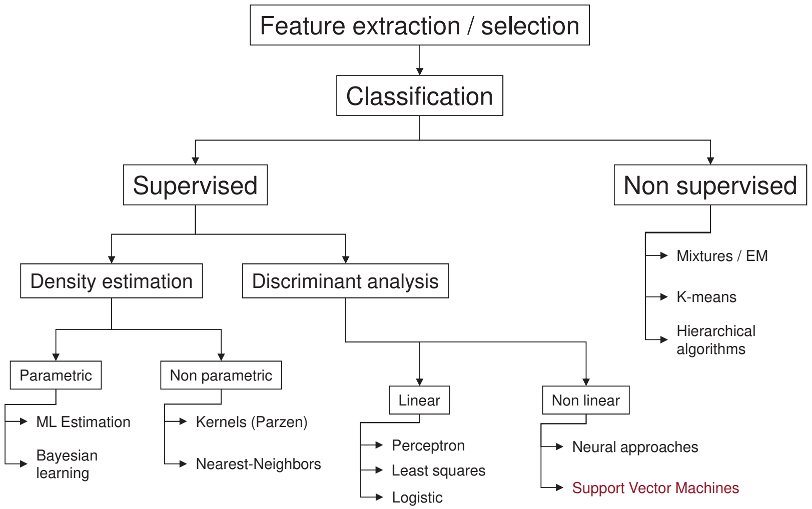
Kernels and non linear Support Vector Machines

Special thanks to J.P. Tarel (Université Gustave Eiffel)

Summary of previous episodes

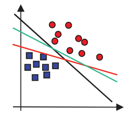
- **In this course, we studied several ways of devising linear discriminant functions**
  - Parametric Gaussian models with identical variances in the Bayesian framework,
  - Direct estimation of the linear discriminant function by criterion minimization (Perceptron, Least Squares or Logistic regression, for example).
- **We also studied 2 ways of generating discriminant functions in a neuro-mimetic (or neural) approach**
  - Namely, multilayer Perceptrons and Radial basis Functions (or RBF's)
- **We now introduce an alternative technique: Support Vector Machines or SVMs (in French: "Séparateurs à Vaste Marge").**
  - Proposed in 1995, based on much older ideas (1963, 1964…)

A hierarchy of methods

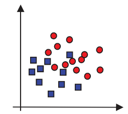


Introduction – Basis ideas

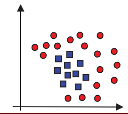
- **Support Vector Machines (SVM)**
  - Essentially, a special case of linear discriminant.
  - Consider N samples drawn from two linearly separable classes.
  - Each sample $\mathbf{x}_k$ has a label, $t_k \in \{-1,1\}$
  - Find a separating hyper-plane by a learning procedure



- **Multiple possible choices**
  - Solution: Margin maximization methods [Vapnik1963]



- **Classes may not be linearly separable**
  - Class overlap
    Solution: Slack variables [Cortes1995] / hinge loss
  - Nonlinearity
    Solution: Kernel methods [Aizerman1964]

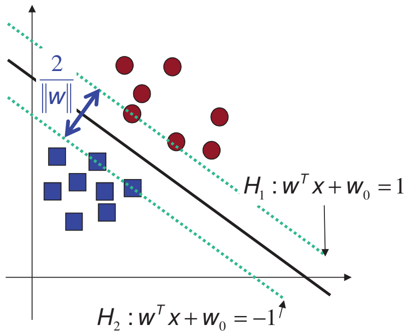


Maximum margin linear classifiers

- **Perceptron with margin, where $t_k \in \{-1,1\}$**

  $$t_k(\mathbf{w}^T\mathbf{x}_k + w_0) \geq b$$

  - Identical up to a scaling factor…
  - Canonical Hyper-plane : $|b| = 1$ for samples lying on margin hyper-planes H_1 and H_2
  - The distance from these points to the separating surface (the margin) is $1/\|\mathbf{w}\|$



- **Maximizing the margin = constrained optimization problem**
  - Minimize $\frac{1}{2}\|\mathbf{w}\|^2$ s.t. $t_k(\mathbf{w}^T\mathbf{x}_k + w_0) \geq 1$, for $k = 1, \ldots, N$

Learning: the dual optimization problem

- **Tool for constrained optimization: the Lagrangian**

  $$L_P = \frac{1}{2}w^T w - \sum_{k=1}^{N} \alpha_k \left(t_k\left(w^T x_k + w_0\right) - 1\right)$$

  - Minimize w.r.t. w, w_0 and maximize w.r.t. $\alpha_k \geq 0$ (Lagrange multipliers)

  $$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{k=1}^{N} \alpha_k t_k x_k \qquad \frac{\partial L_P}{\partial w_0} = 0 \Rightarrow \sum_{k=1}^{N} \alpha_k t_k = 0$$

- **Plugging these expressions into L_P, one obtains the following dual problem: maximize w.r.t. α**

  $$\boxed{L_D = \sum_{k=1}^{N} \alpha_k - \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k t_j t_k \left(x_j^T x_k\right)}$$

- **Subject to the (simpler) constraints** $\alpha_k \geq 0$ and $\sum_{k=1}^{N} \alpha_k t_k = 0$

Learning: solving the dual problem

- **Numerical methods are available to solve this quadratic programming problem and estimate α^{opt},**
  - e.g. Matlab's `quadprog` routine (optimization toolbox)
  - or Python's `cvxopt` or `quadprog` packages
- **A constrained optimization problem of this sort satisfies the Kuhn, Karush and Tucker conditions**

  $$\alpha_k \geq 0$$
  $$t_k\left(w^T x_k + w_0\right) - 1 \geq 0$$
  $$\alpha_k\left(t_k\left(w^T x_k + w_0\right) - 1\right) = 0$$

  - At least one factor must be zero. For most samples, it will be α_k

➔ **Points lying on H_1 and H_2 are the only ones for which $\alpha_k = 0$ may not be satisfied. They are called Support Vectors (SV)**

Discriminant function

- **Once the learning stage is performed, the discrimination rule may be simplified by keeping only the $n \leq N$ SV: the solution is *sparse***
- **Using the expression of w as a function of α opt :**

  $$w = \sum_{k=1}^{N} \alpha_k t_k x_k$$

  $$\boxed{g(x) = \sum_{k=1}^{n} \alpha_k^{opt} t_k \left(x_k^T x\right) + w_0^{opt}}$$

- **For w_0, one must come back to the primal problem,**
  - $t_i . g(\mathbf{x}_i) = 1$ for all support vectors, i.e. $t_i^2 . g(\mathbf{x}_i) = g(\mathbf{x}_i) = t_i$ (since $t_i^2 = 1$)
  - Replacing $g(\mathbf{x}_i)$ by its expression and taking the mean

  $$t_i - \sum_{k=1}^{n} \alpha_k^{opt} t_k (\mathbf{x}_k^T \mathbf{x}_i) = w_0^{opt}$$

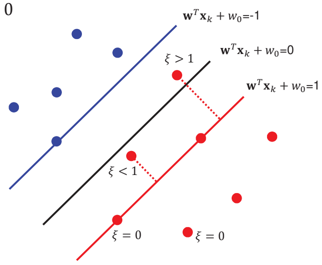
  $$w_0^{opt} = \frac{1}{n} \sum_{i=1}^{n} \left(t_i - \sum_{k=1}^{n} \alpha_k^{opt} t_k \left(x_k^T x_i\right)\right)$$

Learning with overlapping class distributions

- **The *soft margin* model [Cortes1995]**
  - Introduces *slack* variables, $\xi_k \geq 0, k = 1, \dots, N$ (one per sample)
  - Relaxes classification condition: $t_k(\mathbf{w}^T\mathbf{x}_k + w_0) \geq 1 - \xi_k$
    - ✓ $\xi_k = 0$: usual condition $t_k(\mathbf{w}^T\mathbf{x}_k + w_0) \geq 1$
    - ✓ $0 < \xi_k \leq 1$: allows margin violation, $t_k(\mathbf{w}^T\mathbf{x}_k + w_0) \geq 0$
    - ✓ $\xi_k > 1$: allows misclassification, $t_k(\mathbf{w}^T\mathbf{x}_k + w_0) < 0$
  - Penalizes relaxations: $\sum_{k=1}^{N} \xi_k$
- **The optimization problem becomes**
  - Minimize $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{k=1}^{N} \xi_k$

    s.t. $t_k(\mathbf{w}^T\mathbf{x}_k + w_0) \geq 1 - \xi_k$,

    $\xi_k \geq 0,\ k = 1, \dots, N$



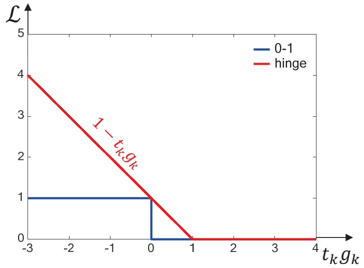
Following [Bishop]

- **Mathematical developments are similar**
  - Support Vectors satisfy : $t_k(\mathbf{w}^T\mathbf{x}_k + w_0) = 1 - \xi_k$

Learning: solving the primal problem

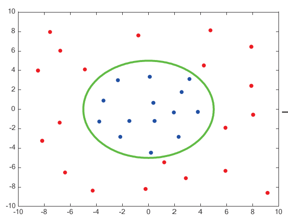
- **Dual problem: size N (one Lagrange multiplier per sample)**
- **Primal: size $D + 1$ (one coefficient per dimension + bias)**
- **Reformulate primal criterion as unconstrained optimization**
  - Introducing $g_k \stackrel{\text{def}}{=} \mathbf{w}^T\mathbf{x}_k + w_0$, the constraint $t_k g_k \geq 1 - \xi_k$ may be written as $\xi_k \geq 1 - t_k g_k$ which, with $\xi_k \geq 0$, is equivalent to $\xi_k^* = \max(0{,}1 - t_k g_k)$
  - The learning problem is formulated as the minimization, with respect to $\mathbf{w}$, of

$$\underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\text{regularization}} + C\sum_{k=1}^{N} \underbrace{\max(0{,}1 - t_k g_k)}_{\textit{hinge}\text{ loss } \mathcal{L}}$$

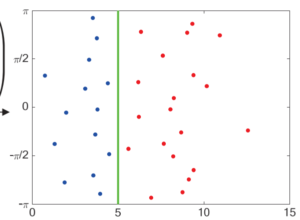


- **Algorithm**
  - The hinge loss is not differentiable, but is convex and has a *sub-gradient*
  - The primal criterion can be optimized using (stochastic) gradient descent

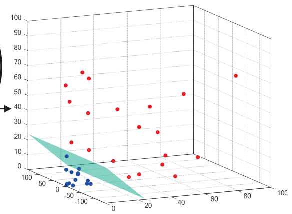
Managing non linearly separable data



$$\Phi: \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r = \sqrt{x_1^2 + x_2^2} \\ \theta = \operatorname{atan}\left(\frac{x_2}{x_1}\right) \end{pmatrix}$$



$$\Phi: \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$



- **Map $\mathbf{x}$ to $\Phi(\mathbf{x})$ to make data separable**
  - Use polar coordinates

    $\mathbb{R}^2 \rightarrow \mathbb{R}^2$
  - Or map data to higher dimension

    $\mathbb{R}^2 \rightarrow \mathbb{R}^3$

SVM in transformed feature space

- **Map coordinates to transformed feature space**

$$\Phi: \mathbb{R}^d \mapsto \mathbb{R}^D$$
$$\mathbf{x} \rightarrow \Phi(\mathbf{x})$$

- **Learn a linear classifier for w ∈ $\mathbb{R}^D$**

$$g(\mathbf{x}) = \mathbf{w}^T\Phi(\mathbf{x}) + w_0$$

e.g. primal classifier formulation

$$\min_{\mathbf{w}\in\mathbb{R}^D} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{k=1}^{N} \max\big(0{,}1 - t_k g(\mathbf{x}_k)\big)$$

- **If $D \gg d$, there are many more parameters to learn for w**

➔ **Can we avoid this?**

Dual SVM in transformed feature space

- **Expression of the discriminant function (dual)**

$$g(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k t_k \, \Phi(\mathbf{x}_k)^T \Phi(\mathbf{x}) + w_0$$

- **Learning: the dual optimization problem**

$$\text{Maximize } L_D = \sum_{k=1}^{N} \alpha_k - \sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k t_j t_k \, \Phi(\mathbf{x}_j)^{\mathrm{T}} \Phi(\mathbf{x}_k)$$

subject to $\alpha_k \geq 0$ and $\sum_{k=1}^{N} \alpha_k t_k = 0$

- **Remarks**
  - Note that, Φ appears in scalar products $\Phi(\mathbf{x})^T \Phi(\mathbf{x}')$
  - Once the scalar products are computed, only the N –dimensional vector $\boldsymbol{\alpha}$ needs to be learnt (instead of $D+1$ weights in the primal problem)

The "Kernel Trick" [Aizerman 1964]

- **Write $\Phi(\mathbf{x})^T \Phi(\mathbf{x}') = K(\mathbf{x}, \mathbf{x}')$, where K is known as a kernel**
- **Expression of the discriminant function**

$$g(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k t_k \, K(\mathbf{x}_k, \mathbf{x}) + w_0$$

- **Learning: the dual optimization problem**

$$\text{Maximize } L_D = \sum_{k=1}^{N} \alpha_k - \sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k t_j t_k \, K(\mathbf{x}_j, \mathbf{x}_k)$$

subject to $\alpha_k \geq 0$ and $\sum_{k=1}^{N} \alpha_k t_k = 0$

- **The Kernel Trick**
  - Nor learning neither classification require explicitly computing the mapping, $\Phi(\mathbf{x})$
  - The mapping remains implicit: all that is needed is the kernel function, $K(\mathbf{x}, \mathbf{x}')$
  - Computing kernels is most often cheaper than computing scalar products in transformed feature space

Example: quadratic kernels

- **Consider the squared scalar product of $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$**

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^2 = \left(\begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix} \right)^2 = (x_1 x'_1 + x_2 x'_2)^2$$

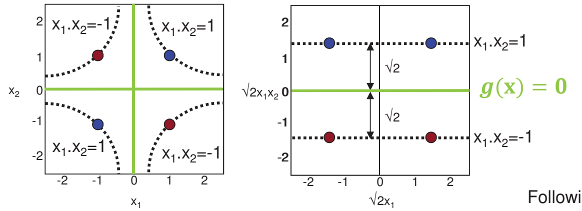
$$= (x_1 x'_1)^2 + 2 x_1 x'_1 x_2 x'_2 + (x_2 x'_2)^2 = \begin{pmatrix} x_1^2 & \sqrt{2} x_1 x_2 & x_2^2 \end{pmatrix} \begin{pmatrix} x'^2_1 \\ \sqrt{2} x'_1 x'_2 \\ x'^2_2 \end{pmatrix}$$

so $K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$ with $\Phi(\mathbf{x}) = \begin{pmatrix} x_1^2 & \sqrt{2} x_1 x_2 & x_2^2 \end{pmatrix}^{\mathrm{T}}$

- **Note that, this decomposition is not unique. It also works for**
  - $\Phi(\mathbf{x}) = (1/\sqrt{2}) \begin{pmatrix} x_1^2 - x_2^2 & 2 x_1 x_2 & x_1^2 + x_2^2 \end{pmatrix}^{\mathrm{T}}$ in $\mathbb{R}^3$
  - $\Phi(\mathbf{x}) = \begin{pmatrix} x_1^2 & x_1 x_2 & x_1 x_2 & x_2^2 \end{pmatrix}^{\mathrm{T}}$ in $\mathbb{R}^4$
- **The simple polynomial kernel contains terms of degree 2 only**
  - $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^2$ with $c > 0$ also contains linear and constant terms

Dealing with the XOR with polynomial kernels

- **Consider the kernel $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^2$**
  - After some manipulations (exercise): $\Phi(\mathbf{x}) = (1, \sqrt{2}.x_1, \sqrt{2}.x_2, \sqrt{2}.x_1 x_2, x_1^2, x_2^2)^{\mathrm{T}}$, $\mathcal{H} = \mathbb{R}^6$
- **In the XOR case, (centered coordinates)**
  - (-1,-1) and (1,1) belong to class ($t = 1$), (1,-1) and (-1,1) to class ($t = -1$)
  - "Manual" optimization leads to $\alpha_k = 1/8 \ \forall k$
  - The 4 points are support vectors (unusual, due to the symmetry of the XOR)
  - The resulting discriminant is: $g(\mathbf{x}) = x_1 . x_2$



Following [Duda]

Mercer kernels

- **Any function K that fulfills Mercer's condition may be used as a Kernel (no need to know the lifting function, Φ, explicitly)**
  - If $K(\mathbf{u},\mathbf{v})$ is such that, for all square-integrable function f (i.e. $\iint f(\mathrm{u})^2 du < \infty$) $\iint K(\mathbf{u},\mathbf{v}) f(\mathbf{u}) f(\mathbf{v}) d\mathbf{u} d\mathbf{v} \geq 0$ then there exists a mapping:

$$\Phi\colon \mathcal{L} \mapsto \mathcal{H}$$
$$\mathbf{x} \to \Phi(\mathbf{x})$$

with an expansion $K(\mathbf{u},\mathbf{v}) = \langle \Phi(\mathbf{u}), \Phi(\mathbf{v}) \rangle$ for all $(\mathbf{u},\mathbf{v}) \in \mathcal{L}^2$

- **Kernel arithmetic** [Bishop, 2006]
  - The following kernels: $ck(\mathbf{x},\mathbf{x}')$, $f(\mathbf{x})k(\mathbf{x},\mathbf{x}')f(\mathbf{x}')$, $q\big(k(\mathbf{x},\mathbf{x}')\big)$, $\exp\big(k(\mathbf{x},\mathbf{x}')\big)$, $k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}')$, $k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}')$, … are valid kernels

  where $k(\mathbf{x},\mathbf{x}')$, $k_1(\mathbf{x},\mathbf{x}')$, $k_2(\mathbf{x},\mathbf{x}')$ are valid kernels, $c > 0$ is a constant, f is any function, q is a polynomial with nonnegative coefficients

Well-known kernels

- **Linear kernel**

$$K(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\mathbf{x}'$$

- **Polynomial kernels**
  - Parameter M: the degree of the polynomial

$$K(\mathbf{x},\mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}'\right)^M$$

  ✓ Contains all monomials of order M

$$K(\mathbf{x},\mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + c\right)^M \text{ with } c > 0$$

  ✓ Contains all polynomial terms up to order M

- **Gaussian kernel**
  - Parameter $\sigma > 0$: standard deviation

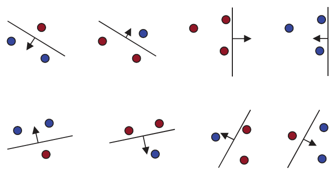
$$K(\mathbf{x},\mathbf{x}') = \exp\left(-\|\mathbf{x}-\mathbf{x}'\|^2/2\sigma^2\right)$$

  ✓ No need to consider normalization constants in this context

  ✓ Dimension of transformed feature space = infinite ➔ cannot work in transformed space

The Vapnik-Chervonenkis dimension

- **The VC-dimension of a function is the <u>maximum</u> number of points that can be separated (*shattered*) using this function.**
  - E.g. oriented hyper-planes in $\mathbb{R}^D$
    VC-dimension = $D + 1$



From [Burges,1998]

- **Polynomial kernels of order M have a VC-dimension of**

$$\binom{d_{\mathcal{L}} + M - 1}{M} + 1$$

where $d_{\mathcal{L}}$is the dimension of $\mathcal{L}$ (VC grows fast with M!)

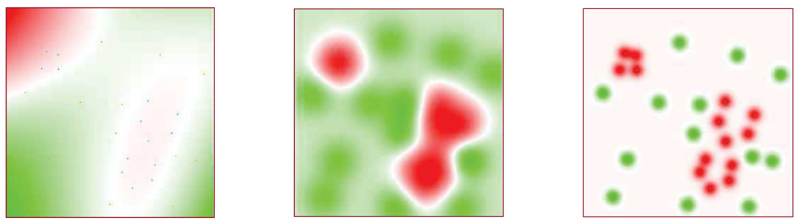
- **The Gaussian kernel has an infinite VC-dimension**
  - Highly flexible, but beware of overfitting!

Setting up hyper-parameters

- **Care must be taken to the choice of the spread parameter of the Gaussian kernel**

$$K(x_1, x_2) = \exp\left(-\frac{1}{2\sigma^2}\|x_1 - x_2\|^2\right)$$

- **In practice: cross-validation, for example**



$\sigma = 1.0$, 10 SV — $\sigma = 0.01$, 26 SV — $\sigma = 0.001$, 27 SV

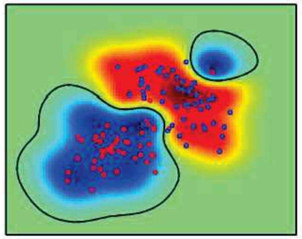
JP Tarel (UGE)

Graphical interpretation

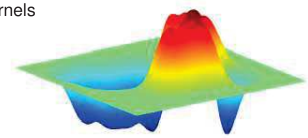
- **Constructing the discriminant function $g(\mathbf{x})$ amounts to positioning a kernel on each support vector, which defines a hyper-surface**

$$\sum_{k=1}^{N} \alpha_k t_k\, K(\mathbf{x}_k, \mathbf{x})$$

- **The decision boundary, $g(\mathbf{x}) = 0$, is obtained by "cutting" the hyper-surface at the altitude $-w_0$**



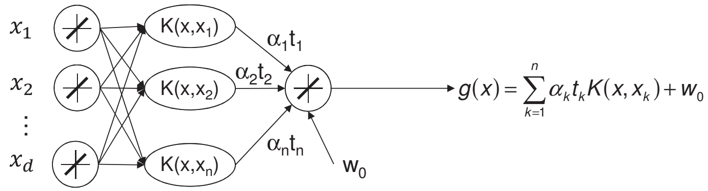
e.g. with Gaussian kernels



[Davy2003]

Link with neural networks

- **Dual SVMs may be seen as neural networks:**



- **Difference with 2-layer Perceptrons**
  - The number of hidden neurons is determined automatically by the support vectors $x_1 \ldots x_n$. Their output weights are the Lagrange multipliers, α_k.
- **Difference with RBF's**
  - The number of centers and their positions are automatically given by the learning stage of the SVM, as well as their weights (Lagrange multipliers) and threshold w_0.

Take-away remarks

- **Just as multilayer Perceptron's and RBF's,**
  - SVM's can deal with non-linearly separable classes, thanks to the "kernel trick" and imperfectly separable classes, thanks to the "soft-margin model"
  - However, the number of parameters is much less, thanks to the maximal margin criterion
    - ✓ Spread parameter, σ, plus regularization parameter, C, for soft margins.
    - ✓ The other ones are automatically given by the learning procedure
- **Learning = solve quadratic programming (optimization) problem**
  - May be performed in a "reasonable" (polynomial) amount of time
  - Thanks to duality, the dimensionality does not matter…
  - Optimizing the primal problem is also feasible
- **The kernel trick may be applied to any algorithm were the input vector enters only in the form of scalar products**
  - Arises naturally in the dual form of SVMs
  - Perceptrons also may be *kernelized* (Aizerman,1964), see Computer Exercise 16
  - Other *kernelized* algorithms include kernel PCA, Kernel Nearest-Neighbors classifier, Kernel Fisher discriminant, Kernel logistic regression…
- **Extensions: 1-class SVM, multi-class SVM, regression SVM**

Ensemble learning

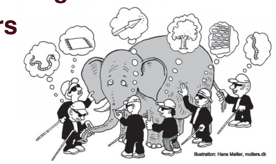
Ensemble learning

Bagging - decision trees and random forests

Boosting - AdaBoost

Introduction: ensemble learning

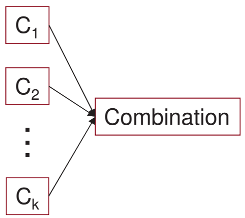
- **Methods that learn a target function by combining the predictions of a number of individual learners**
  - Models combinations are also called committees



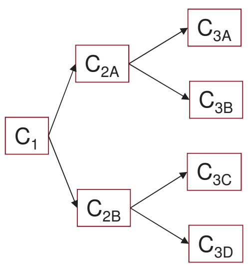
- **Why ensemble learning ?**
  - Decompose complex problems into multiple easier sub-problems
  - Improve performance of individual (weak) learners
  - No single model can solve all pattern recognition problems
- **Possible strategies to generate classifiers**
  - Subsample the learning data set, then use subsets to train classifiers e.g. random sampling with replacement = bootstrap
  - Train individual classifiers on different feature representations
  - Use different training parameters (e.g. k in kNN) to generate classifiers

Possible structures of ensemble classifiers

- **Parallel (majority of approaches)**
  - Independent classifications
  - Combination (average, majority vote, L-bit code word + error-correcting codes)

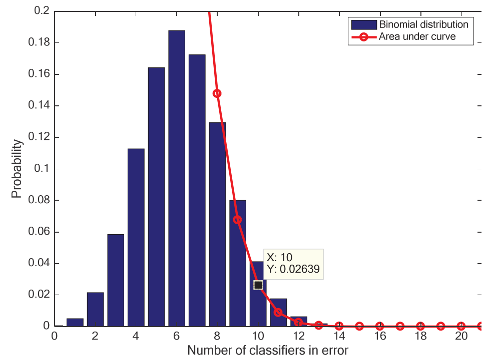


- **Hierarchical (cascade)**
  - Sequential or tree-like combination
  - First, inaccurate but fast classifiers
  - Then, more accurate but more computationally intensive methods



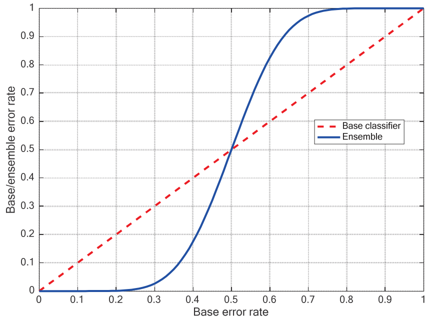
Motivating example [Dietterich, 1997]

- **Averaging can eliminate uncorrelated errors of classifiers**
  - e.g. combining 21 classifiers by majority vote
  - Error rate of individual classifiers = 0.3 (30 %)
  - Misclassification ⇒ 11 classifiers in error over 21 (at least)
  - Probability: 0.026 (2,6 %)



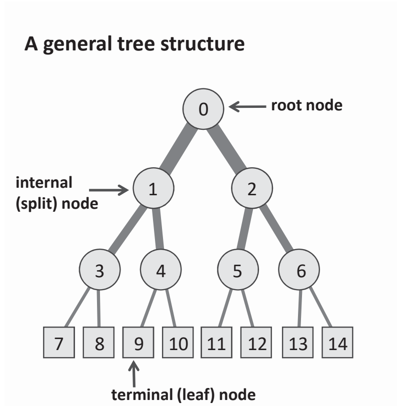
Motivating example

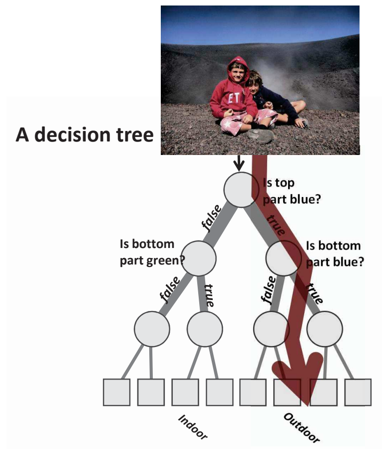
- **Plot of the ensemble error as a function of individual errors**
  - Base classifiers must be **better than random** (less than 50 % individual error)
  - Another condition for success of the committee is **diversity** (i.e. uncorrelated errors)



- **We will introduce two well-known strategies**
  - Train individual classifiers of the same kind on bootstrap samples of the training set and combine them by majority voting: bagging
  - Train classifiers in sequence, adapting the training function to the performance of the previous model: boosting.

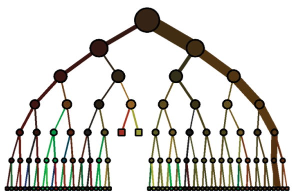
Decision trees (From [Criminisi2012])

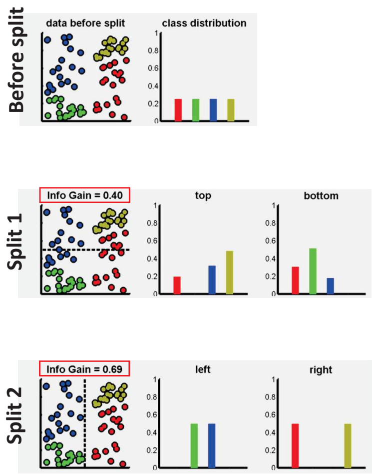




Building trees (From [Criminisi2012])

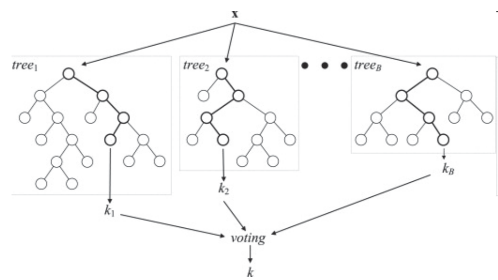
- **Each node corresponds to a binary decision (threshold)**
  - Weak learner
- **Decision based on an information - theoretic measure (entropy)**
  - Promotes "peakness" or "purity"
- **Random tree**
  - Random selection of a subset of features at each node





Random forests [Breiman2001]

- **Bootstrap aggregating = bagging**
- **Learning**
  - Create sub-samples of learning data set by random selection with replacement
  - Build one random tree per sub-sample ➔ hence the name, random forest
- **Decision**
  - Go down all trees
  - Select class by voting
- **Remarks**
  - Fast (parallel)
  - Improves accuracy of unstable, uncorrelated classifiers



[Verikas2011]

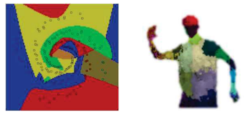
Remarks

- **Out-of-bag (OOB) error**
  - Measures classification error on the data not used for training
  - Similar to (cross-)validation error
- **Variable importance**
  - Provides an ordered importance measure of the features used
  - Might be sensitive to the number of levels for categorical (discrete) features

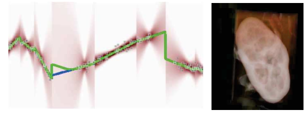
- **Resources**
  - Decision Forests in Computer Vision and Medical Image Analysis. A. Criminisi and J. Shotton. Springer. 2013.
  - http://research.microsoft.com/projects/decisionforests/
  - Code
    - ✓ The Microsoft Research Cambridge Sherwood Software Library
    - ✓ Matlab (classification Toolbox), sklearn.ensemble.RandomForestClassifier
  - MOOCS (Nando de Freitas) : http://www.youtube.com/watch?v=-dCtJjlEEgM

Decision forests (From [Criminisi2012])

- **The concept of classification and regression trees can be extended to other applications**

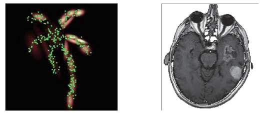


Classification / segmentation

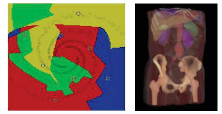


Regression / object localization

Manifold learning / dimensionality reduction



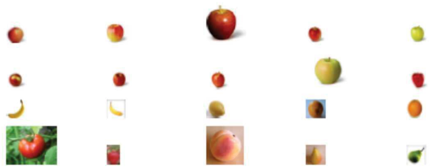
PDF estimation / novelty detection



Semi-supervised classification / segmentation

AdaBoost [Freund&Schapire1995]

- **Stands for <u>ada</u>ptive <u>boost</u>ing**
- **Combines weak classifiers**
  - Misclassified samples are given more weight in successive learning step
- **Intuitive example: learning kids how to recognize apples**

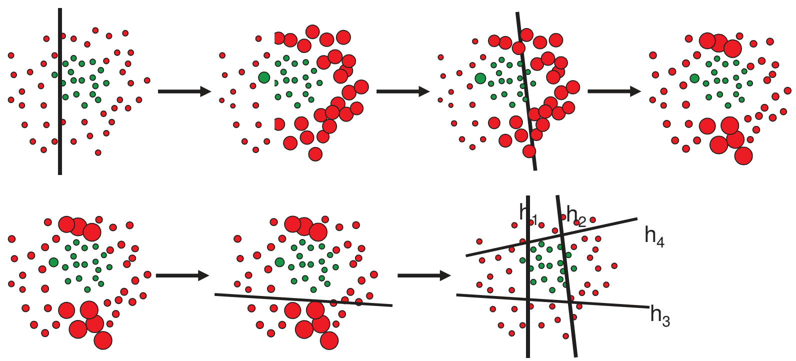


“Apples are somewhat circular and somewhat red and possibly green”

From [Hsuan-Tien Lin,2009]

AdaBoost [Freund&Schapire1995]

- **Example (weak learner = linear discriminant)**



From [Antonio Torralba @MIT]

AdaBoost algorithm *[Freund&Schapire1995]*

- **Input: training examples $D=\{(x_n,t_n)\}_{n=1\ldots N}$**
- **For t = 1, 2 … T,**
  - Learn a simple rule h_t from emphasized training examples.
    - ✓ How? Choose a $h_t \in H$ with minimum emphasized error.
  - Get the confidence α_t of such rule
    - ✓ How? An h_t with lower error should get higher α_t
  - Emphasize the training examples that do not agree with h_t
    - ✓ How? Maintain an emphasis value (weight) u_n per example
- **Output: combined function**

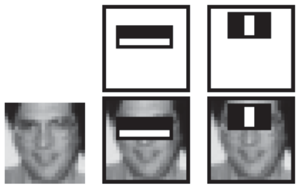
$$Y(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Application to face detection *[Viola&Jones2001]*

- **Widely-used method for face (and other objects) detection**
  - Detection, not recognition…i.e. binary classification
- **Very fast classification (real time), but slow training**
- **Viola&Jones method:**
  - Introduces Haar-like, fast-to-compute image features, that are well suited to face detection
  - Uses AdaBoost algorithm where each stage of the boosting process selects a single feature
  - Exploits a cascade of classifiers with increasing complexity and decreasing false alarm rate

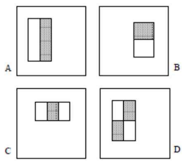
Features

- **Faces share specific properties**
  - Eye region are darker than cheeks
  - Nose bridge is brighter than eyes
  - == Domain-based knowledge
  - → Can be captured by rectangle filters



[Viola&Jones2001]

- **Haar-like feature detector**
  - Feature value = Σ (pixels in white rectangles) - Σ (pixels in black rectangles)
  - Consider two- three- four-rectangle patterns
  - All possible positions and sizes are considered
  - ~160 000 possibilities in a 24x24 window



- **Fast computation using integral images**

[Viola&Jones2001]

Integral images

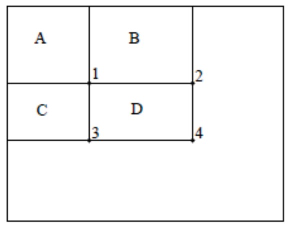
- **Sum of pixels above and to the left of the current pixel**
  - With i=original image, ii = integral image

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y')$$

- **Recursive implementation**

$$s(x,-1) = 0 \text{ and } ii(-1,y) = 0$$
$$s(x,y) = s(x,y-1) + i(x,y)$$
$$ii(x,y) = ii(x-1,y) + s(x,y)$$

  - Where s is the cumulative row sum
  - The integral image is computed only once, and in a single pass



[Viola&Jones2001]

- **Any rectangular sum (feature part) computed in constant time**
  - e.g. 1=A, 2=A+B, 3=A+C, 4=A+B+C+D → **D = 4+1-(2+3)**

AdaBoost-feature selection

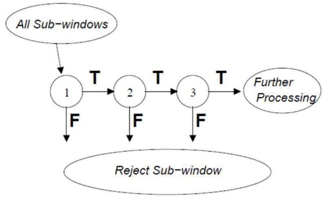
- **On each round, many possibilities of weak classifiers**
- **1 feature → 1 weak classifier**
- **At each stage of boosting**
  - Given reweighted data from previous stage
  - Train all K (160,000) single-feature classifiers (threshold=*decision stump*)
  - Select the single best classifier (lowest weighted classification error)
  - Combine it with the other previously selected classifiers
  - Reweight the data
- **Repeat until T classifiers selected**

- **Very computationally intensive**
  - Learning K decision stumps T times
  - E.g., K = 160,000 and T = 1000

- **Performs training + feature selection**

Cascade training

- **A positive decision from the first classifier triggers the evaluation of the second one and so on…**



[Viola&Jones2001]

- **The classifiers are arranged by increasing complexity**
  - 1 feature classifier → 100% detection rate, ~50% false positive rate
  - 5 features → 100% detection rate, ~40% FP rate (20% cumulative)
  - 20 features → 100% detection rate, 10% FP (2% cumulative)

- **Viola&Jones: 38 layers in the cascade, 6060 features**
  - 2, 10, 25, 25, 50, 50, 50 … features.

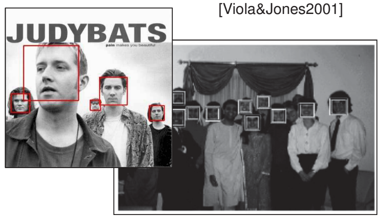
Results (1/2)

- **Learning**
  - 5000 faces x2 (vertical symmetry)
  - 10000 non-faces

- **Learning takes "weeks"…**

- **Detection**
  - 24x24 sliding window
  - Multi-scale: scale the detector, not the image (features can be computed at any scale)
  - In average, 10 features/window on test set
  - 10x faster than single-stage AdaBoost with 200 features
  - Due to analyzing window overlap, multiple nearby detections may occur → post-processing

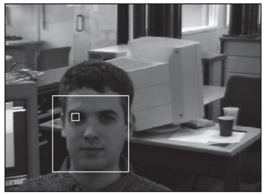
[Viola&Jones2001]



Results (2/2)

Pros and cons

- **Pros**
  - Extremely fast feature computation
  - Scale and location invariant
  - Very generic: can be trained to detect other facial features: nose, eyes, or body parts or many objects: cars, licence plates…
  - Public implementations (and pre-trained cascades, also) available e.g. OpenCV, Matlab



- **Cons**
  - Can hardly cope with face rotations (needs special training for profiles)
  - Sensitive to lighting conditions

Take-away

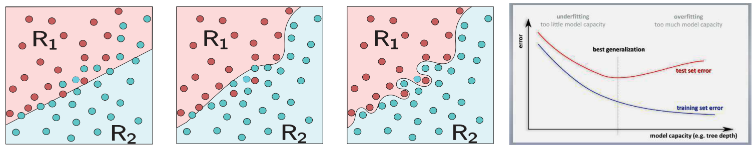
- **Ensemble learning**
  - Multiple learners are trained to solve the same problem
  - They try to build a set of diverse hypotheses and combine them for use
- **Diversity**
  - Necessary to improve the performance of basic (weak) classifiers
  - Can be introduced in different ways
    - ✓ Subsampling the training examples (e.g. bagging)
    - ✓ Manipulating the attributes, i.e. using subsets of a common feature representation
    - ✓ Using different training parameters
    - => Injecting randomness into learning algorithms
- **Strategies**
  - Bagging: use bootstrap (sampling with replacement), e.g. Random Forests
    - ✓ A "Swiss knife" for machine learning?
  - Boosting: focusing on misclassified samples, e.g. AdaBoost

Evaluation

Methodology
Confusion matrices
Scoring metrics
ROC / PR curves

Evaluation: basis ideas

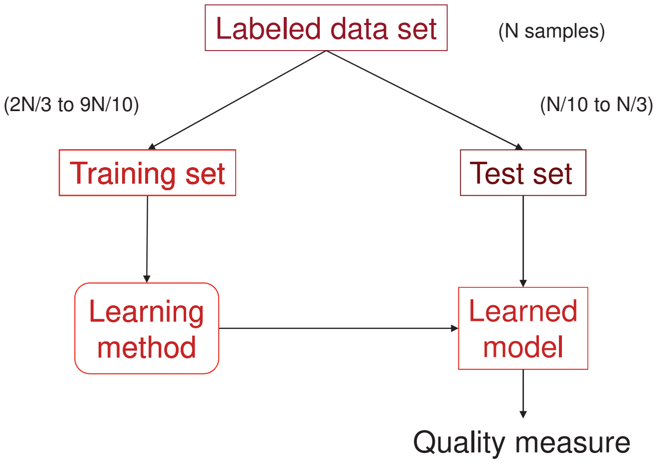
- **Why?**
  - Setting-up classifiers: compare methods/ optimize classifier parameters
  - Assess performance (and limitations) of classifiers
    - ✓ No classifier is perfect !
    - ✓ Generalization capacities



Underfitting | Correct learning | Overfitting | (From [Dietterich2011])

- **How?**
  - Needs a test data set different from training
    - ✓ Supervised: "ground-truth"
  - What if not enough data? Generate sub-sets (cross-validation)
  - Define a quality measure

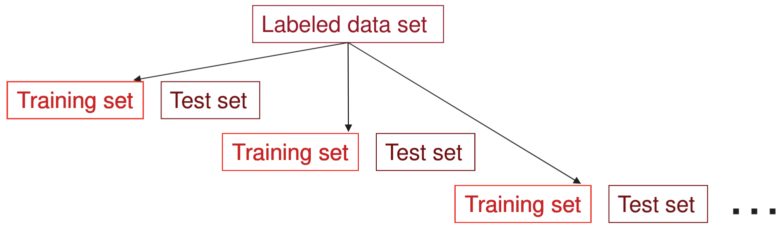
Test set definition (holdout method)



- Stratification: each class represented in both sets with same proportion

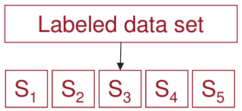
Random resampling

- **Using a single training/test partition may be limited**
  - E.g. not enough data to make sufficiently large training and test sets
    - ✓ Large test set → more reliable estimate of performance
    - ✓ Large training set →more representative
- **Performance may be sensitive to the choice of the training set**
- **Solution: repeated random partitioning of the available data into training/test sets (repeated holdout) + average performance**



Cross-validation (CV)

- **Partition data into n subsamples**
- **Iteratively leave one subsample out for test, train on the rest**
  - e.g. N=100, 5-fold cross validation



Iteration	Train on	Test on	Correct
1	$S_2S_3S_4S_5$	S_1	16/20
2	$S_1S_3S_4S_5$	S_2	15/20
3	$S_1S_2S_4S_5$	S_3	17/20
4	$S_1S_2S_3S_5$	S_4	14/20
5	$S_1S_2S_3S_4$	S_5	18/20

Accuracy 80 %

Cross-validation

- **Widely used approach for estimating test error**
- **Typical value: n=10**
- **Leave-one-out Cross Validation (LOOCV): n=N**
- **Stratified cross-validation: class proportions are maintained in each selected set**
- **Cross-validation vs. bootstrap: bootstrap tends to underestimate test error**
  - Can be corrected: the ".632+" rule [Efron1997]

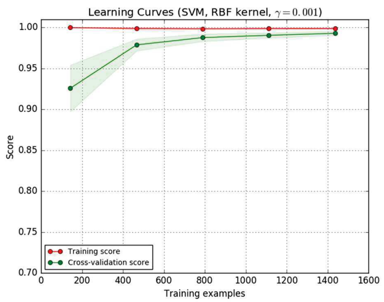
Learning curves

- How does the performance of a learning method change as a function of the training-set size?

- Randomly select n instances from training set
- Learn model
- Evaluate model on test set

➔ One point on the curve

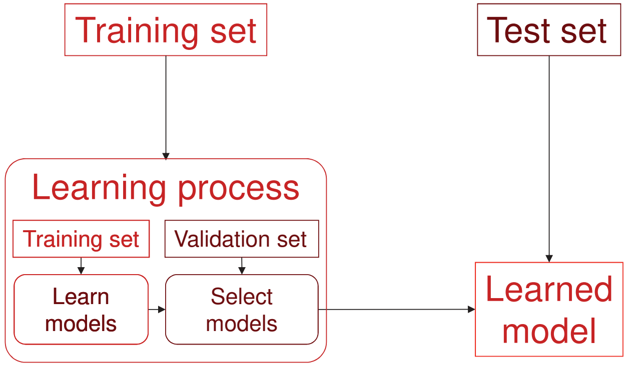
- Can be repeated several times (➔ mean + error bars)



Source : http://scikit-learn.org

Validation sets

- Used for tuning parameters during the learning process



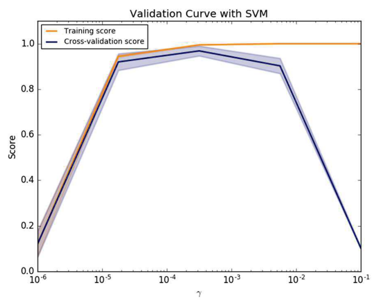
- CV can be used inside learning step: nested cross-validation

Validation score

- Plots the performance of the learned model as a function of a parameter, e.g. kernel width parameter for SVM

- Helps tuning the optimal value

- In general, three cases may be observed
  - Low training score, low validation score: underfitting
  - High scores: good performance
  - High training score, low validation score: overfitting



Source : http://scikit-learn.org

Confusion matrix

- 2 classes
- ω_- negative

  ω_+ positive

		Prediction (result) ω_-	Prediction (result) ω_+	
Truth	ω_-	TN – True negative	FP – False positive	N
Truth	ω_+	FN – False negative	TP – True positive	P

Classification error
- $ERR = \frac{FP+FN}{Total}$

Accuracy
- $ACC = \frac{TP+TN}{Total}$

Recall
- $REC = \frac{TP}{FN+TP}$
- What percentage of objects of interest are detected?

Precision
- $PRE = \frac{TP}{FP+TP}$
- What percentage of what we detect is correct?

$$F_1 = 2\frac{PRE \times REC}{PRE + REC}$$

Dealing with imbalanced classes

- **Accuracy is not a good metric in case of imbalanced classes**
  - E.g. breast cancer detection, 98 % negative, 2 % positive samples
  - Dummy classifier (no learning !) = always predict "negative"
    - ➔ 98% accuracy... but 0 % recall
- **Example remedies to deal with imbalanced classes**
  - Change performance metric (precision, recall, F1-score…)
  - Try to collect more data
  - Oversample minority class, or generate synthetic example (SMOTE)
  - Subsample majority class (at random, or by clustering)
  - Change the algorithm
    - ✓ Penalize wrong classification of the rare class (e.g. use weighted cross-entropy loss)
    - ✓ Tree based algorithms are considered to be less sensitive to the problem
    - ✓ Consider one-class classification + outlier detection

Multi-class confusion matrix

- **e.g. Optical Character (digits) Recognition with rejection class**
- **Observe confusions between, e.g. '4' and '9'**

	class j predicted by a classifier										
true class i	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'R'
'0'	97	0	0	0	0	0	1	0	0	1	1
'1'	0	98	0	0	1	0	0	1	0	0	0
'2'	0	0	96	1	0	1	0	1	0	0	1
'3'	0	0	2	95	0	1	0	0	1	0	1
'4'	0	0	0	0	98	0	0	0	0	2	0
'5'	0	0	0	1	0	97	0	0	0	0	2
'6'	1	0	0	0	0	1	98	0	0	0	0
'7'	0	0	1	0	0	0	0	98	0	0	1
'8'	0	0	0	1	0	0	1	0	96	1	1
'9'	1	0	0	0	3	1	0	0	0	95	0

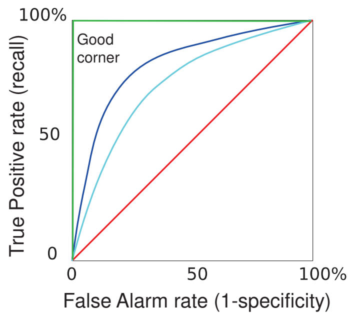
[Hlavác]

Scoring metrics for multiclass classification

- **Accuracy : sum of diagonal elements /number of test samples**
- **Consider *one-versus-all* confusion matrices**
  - $TN_c = \sum_{\mathbf{x} \notin \omega_c} (pred(\mathbf{x}_k) \neq \omega_c)$ $FP_c = \sum_{\mathbf{x} \notin \omega_c} (pred(\mathbf{x}_k) = \omega_c)$
  - $FN_c = \sum_{\mathbf{x} \in \omega_c} (pred(\mathbf{x}_k) \neq \omega_c)$ $TP_c = \sum_{\mathbf{x} \in \omega_c} (pred(\mathbf{x}_k) = \omega_c)$
- **Micro-averaged metrics**
  - e.g. $PRE_{Micro} = \frac{TP_1 + \cdots + TP_C}{TP_1 + \cdots + TP_C + FP_1 + \cdots + FP_C}$
  - Note that $PRE_{Micro} = REC_{Micro} = ACC$
- **Macro-averaged metrics (beware of class imbalance)**
  - $PRE_{Macro} = \frac{PRE_1 + \cdots + PRE_C}{C}$

ROC curves

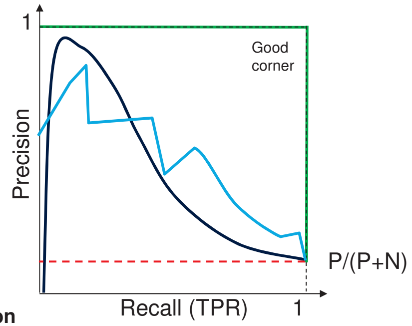
- **ROC stand for Receiver Operational Characteristics**
  - Terminology from RADAR detection
- **Suited to binary classification (C=2)**
  - Helps comparing methods, or tuning parameters
- **ROC curve = parametric curve**
  - Most of the time, parameter = detection threshold
  - For each parameter value,
    - ✓ Perform classification
    - ✓ Count TP / FA
    - ✓ Draw a point on the curve
  - Example ROC curves
    - ✓ In green, ideal curve
    - ✓ In red, pure chance curve
    - ✓ In blue, 2 different classifiers
    - ✓ Performance metric: **Area under curve (AUC)**



Precision-recall curves

- **Plots precision (relevance rate) vs. recall (True Positive Rate) as a function of a given parameter**
  - Most of the time, detection threshold
  - Suited to the C=2 (dichotomy) case

- **Properties**
  - Not always monotonic
  - End point (all samples selected) precision = P/(P+N)
  - Random classifier: constant precision P/(P+N)
  - The higher the curve, the better the classifier.
  - Area under curve = **Average Precision**



To summarize…

- **Evaluation is necessary for:**
  - Assessing the overall performance of a classifier (or a classification method)
    - ✓ Comparison with other classifiers
    - ✓ Tuning parameters
  - Evaluating generalization capabilities

- **Take care to separate training from test data**
  - For tuning parameters, training data may be separated into training / validation

- **Cross validation is popular, especially when few data are available**

- **Accuracy is not always the best evaluation criterion**
  - Sensitive to class imbalance
  - Positive and negative errors may have different meanings
  - Other criteria (precision, recall, specificity) may be envisioned

- **ROC curves and Precision-Recall curves are other common comparison tools for binary classification**

Conclusion

Hierarchy of methods

Current trends in Pattern Recognition

Towards *deep learning*

Conclusion

- **Classification: "The act of taking in raw data and making an action based on the "category" of the pattern" [Duda]**

- **This implies two steps**
  - Machine learning (modeling classes)
  - Classification (making decisions)

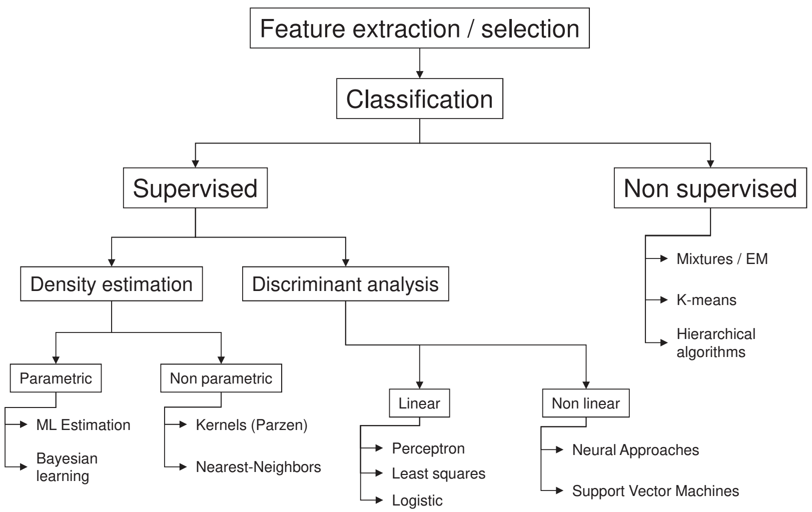
- **We studied the main "traditional" statistical & connectionist pattern recognition approaches**
  - Feature extraction and selection
  - Bayesian decision and optimal classifiers
  - Supervised learning of probability density functions / priors
  - Supervised learning of linear / non-linear discriminant functions
  - Non-supervised learning (probabilistic, deterministic, hierarchical)

Remark: classification and image processing

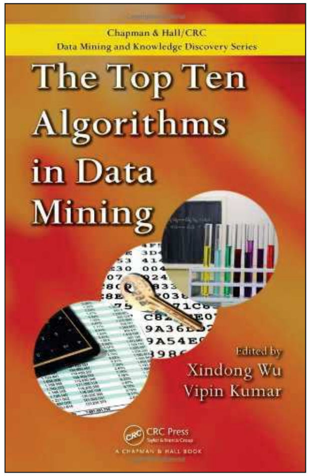
- **Both domains are closely related:**
  - Image processing is required to generate features from the images
    - ✓ Noise removal, contrast enhancement, primitive extraction, segmentation
  - Classification problems are many in I.P. & Computer Vision
    - ✓ e.g. thresholding, detection, segmentation

- **Classification for segmentation**
  - Pixel-wise decision
  - Accounting for pixel interaction: Markov Random Fields (OATI course)
    - ✓ Functional with 2 terms
      - – Distance feature – class model
      - – Regularization: label homogeneity on pixel's neighborhood.
  - Deformable regions (see course on Deformable Models)

Hierarchy of methods



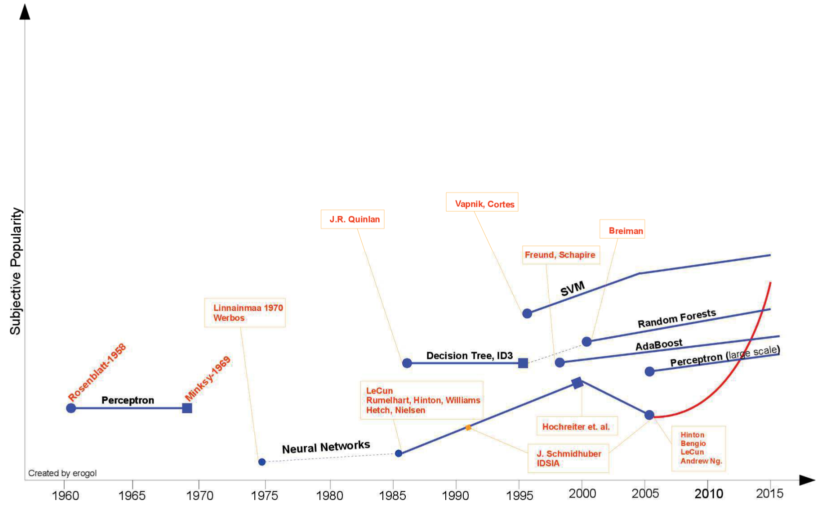
Top 10 algorithms in data mining (IEEE, 2006)



- **kNN**
- **k-Means**
- Apriori
- **EM**
- **Naive Bayes**
- **Decision trees (and forests)**
  - C4.5,
  - CART
- **SVM**
- **AdaBoost**
- PageRank (Google)

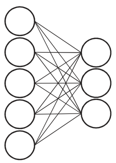
…+ deep learning (2012: ImageNet)

Trends in machine learning

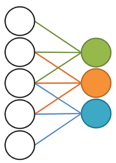


1985 - 2007: towards deep learning

- **NN are high-performance discrimination algorithms**
  - Boom in the 1990s, despite their "black box" side
  - But, still challenged by other algorithms during the early 2000's (*SVM's, Random Forests, AdaBoost and cascades*)
- **Convolutional neural networks (CNN)**
  - Neocognitron (Fukushima,1980), LeNet (LeCun,1989)
  - Reduce the number of connections and share weights
  - This mimics the behavior of *visual cortex*



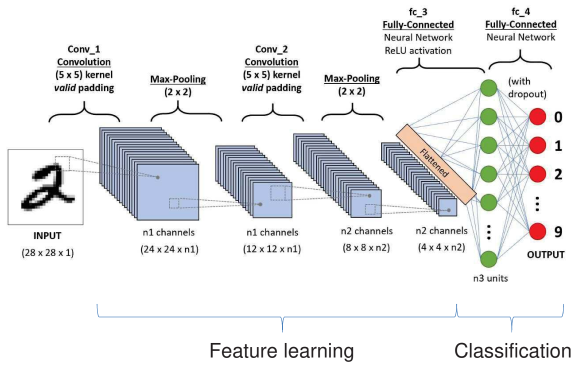
Fully connected layer



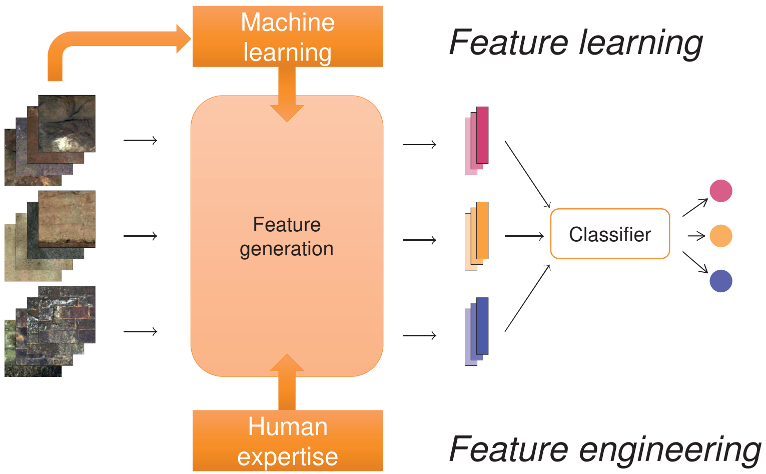
Convolutional layer

Convolutional neural networks (CNN)

- **Input = 1 image, 1 neuron per image region (receptive field)**
- **Weight sharing, local connection → convolution**
- **Down-sampling → *pooling***
- **Learning: back-propagation**



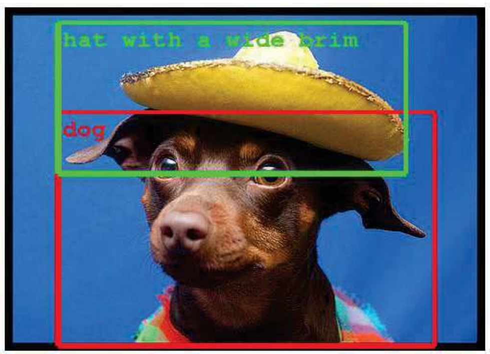
Feature learning vs. feature engineering



The Deep Learning's boom

- **Other progresses in the late 2000's**
  - Bigger datasets
  - Faster computers (GPU's > 2007)
  - Better training algorithms and techniques (>2005)
- **The role of public competitions**
  - Public annotated dataset + competition + workshop
  - 2012: Deep Learning (AlexNet) wins the ImageNet contest
- **New research dissemination habits**
  - Publications (arXiv) and publicly available code (pre-trained NNs)
- **A flourishing research**
  - More and more applications, deeper networks, new architectures

Thank you for your attention…



Credit

Many of the slides in this course were borrowed or adapted from Ricardo Gutierrez Osuna's Pattern Recognition course (Texas A&M University).

Machine Learning and Pattern Recognition Bibliography

Books and Theses

[Biernat and Lutz, 2015] Biernat, E. and Lutz, M. (2015). *Data science : fondamentaux et études de cas*. Eyrolles, Paris.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[Criminisi and Shotton, 2013] Criminisi, A. and Shotton, J. (2013). *Decision forests for Computer Vision and Medical Image Analysis*. Springer, London.

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An introduction to support vector machines (and other kernel-based learning methods)*. Cambridge University Press, Cambridge.

[Duda et al., 2001] Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. John Wiley and Sons, Inc., New York, second edition.

[Fukunaga, 1990] Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Academic Press, New York, second edition.

[Geron, 2019] Geron, A. (2019). *Machine Learning avec Scikit-Learn - Mise en œuvre et cas concrets*. Dunod, Malakoff.

[Guyon et al., 2006] Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. A., editors (2006). *Feature Extraction - Foundations and applications*. Springer, Berlin Heidelberg.

[Hartigan, 1975] Hartigan, J. A. (1975). *Clustering Algorithms*. Wiley, New York.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, second edition.

[Hebb, 1949] Hebb, D. (1949). *The organization of behavior: a neuropsychological theory*. Wiley, New York.

[Jurafsky and Martin, 2000] Jurafsky, D. and Martin, J. (2000). *Speech and Language Processing*. Pearson International Edition, first edition.

[Minsky and Papert, 1969] Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA.

[Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

[Raschka and Mirjalili, 2017] Raschka, S. and Mirjalili, V. (2017). *Python Machine Learning*. Packt Publishing Ltd, Birmingham, second edition.

[Salton and McGill, 1983] Salton, G. and McGill, M. (1983). *Introduction to modern information retrieval*. McGraw-Hill, New York.

[Schalkoff, 1992] Schalkoff, R. (1992). *Pattern recognition: statistical, structural, and neural approaches*. Wiley, New York.

[Theodoridis and Koutroumbas, 1999] Theodoridis, S. and Koutroumbas, K. (1999). *Pattern Recognition*. Academic Press, San Diego, first edition.

[Webb and Copsey, 2011] Webb, A. R. and Copsey, K. D. (2011). *Statistical Pattern Recognition*. Wiley, Chichester, third edition.

[Werbos, 1974] Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University.

[Wu and Kumar, 2009] Wu, X. and Kumar, V. (2009). *The Top Ten Algorithms in Data Mining*. Chapman & Hall/CRC, Boca Raton, first edition.

Papers, Book chapters and Technical reports

[Aizerman et al., 1964] Aizerman, M., Braverman, E., and Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25(6):917–936. Translated from Russian.

[Ball and Hall, 1965] Ball, G. and Hall, D. (1965). Isodata: a method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park, United States. Prepared for Office of Naval Research. Information Sciences Branch.

[Bilmes, 1998] Bilmes, J. A. (1998). A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report TR-97-021, University of Berkeley, Berkeley.

[Boser et al., 1992] Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[Burges, 1998] Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

[Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis Machine Intell.*, 24(5):603–619.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[Cover and Hart, 1967] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

[Csurka et al., 2004] Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. *Workshop on Statistical Learning in Computer Vision, ECCV*, Vol. 1:1–22.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314.

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

[Dietterich, 1997] Dietterich, T. (1997). Machine-learning research - four current directions. *AI Magazine*, 18(4):97–136.

[Dunn, 1973] Dunn, J. C. (1973). A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57.

[Efron and Tibshirani, 1997] Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–560.

[Ester et al., 1996] Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

[Fisher, 1936] Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

[Fix and J.L. Hodges, 1951] Fix, E. and J.L. Hodges, J. (1951). Discriminatory analysis, nonparametric discrimination. Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, USA. Project 21-49-004, Contract AF41(128)-31.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

[Jain et al., 2000] Jain, A. K., Duin, R. P. W., and Mao, J. (2000). Statistical pattern recognition: a review. *IEEE Trans. Pattern Analysis Machine Intell.*, 22(1):4–37.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323.

[Lange et al., 2000] Lange, K., Hunter, D., and Yang, I. (2000). Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–20.

[Le Cun, 1986] Le Cun, Y. (1986). Learning processes in an asymmetric threshold network. In Bienenstock, E., Fogelman-Soulié, F., and Weisbuch, G., editors, *Disordered systems and biological organization*, pages 233–240, Les Houches, France. Springer-Verlag.

[Lee et al., 2011] Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Commun. ACM*, 54(10):95–103.

[Linde et al., 1980] Linde, Y., Buzo, A., and Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95.

[Lloyd, 1982] Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297. University of California Press.

[McCulloch and Pitts, 1943] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

[Moghaddam and Pentland, 2000] Moghaddam, B. and Pentland, A. (2000). Probabilistic visual learning for object representation. *IEEE Trans. Pattern Analysis Machine Intell.*, 19(7):696–710.

[Parzen, 1962] Parzen, E. (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33(3):1065–1076.

[Powell, 1987] Powell, J. (1987). Radial basis functions for multivariable interpolation: a review. In Mason, J. and Cox, M., editors, *Algorithms for Approximation*, pages 143–167. Oxford University Press, Oxford.

[Rosenblatt, 1957] Rosenblatt, F. (1957). The Perceptron - a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell aeronautical laboratory, Inc., Buffalo.

[Roweis, 1997] Roweis, S. (1997). EM algorithms for PCA and SPCA. In *NIPS*, pages 626–632.

[Rumelhart et al., 1986] Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

[Tenenbaum et al., 2000] Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

[Tipping and Bishop, 1999] Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(3):611–622.

[Trunk, 1979] Trunk, G. V. (1979). A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(3):306–307.

[Vapnik, 1963] Vapnik, V. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780.

[Verikas et al., 2011] Verikas, A., Gelzinis, A., and Bacauskiene, M. (2011). Mining data with random forests: A survey and results of new tests. *Pattern Recognition*, 44(2):330–349.

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–511–I–518.

[Widrow and Hoff, 1960] Widrow, B. and Hoff, M. (1960). Adaptive switching circuits. *IRE WESCON Convention Record*, 4:96–104.

[Wu et al., 2008] Wu, X., Kumar, V., Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z. H., Steinbach, M., Hand, D., and Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37.