Last decade success stories of Machine Learning

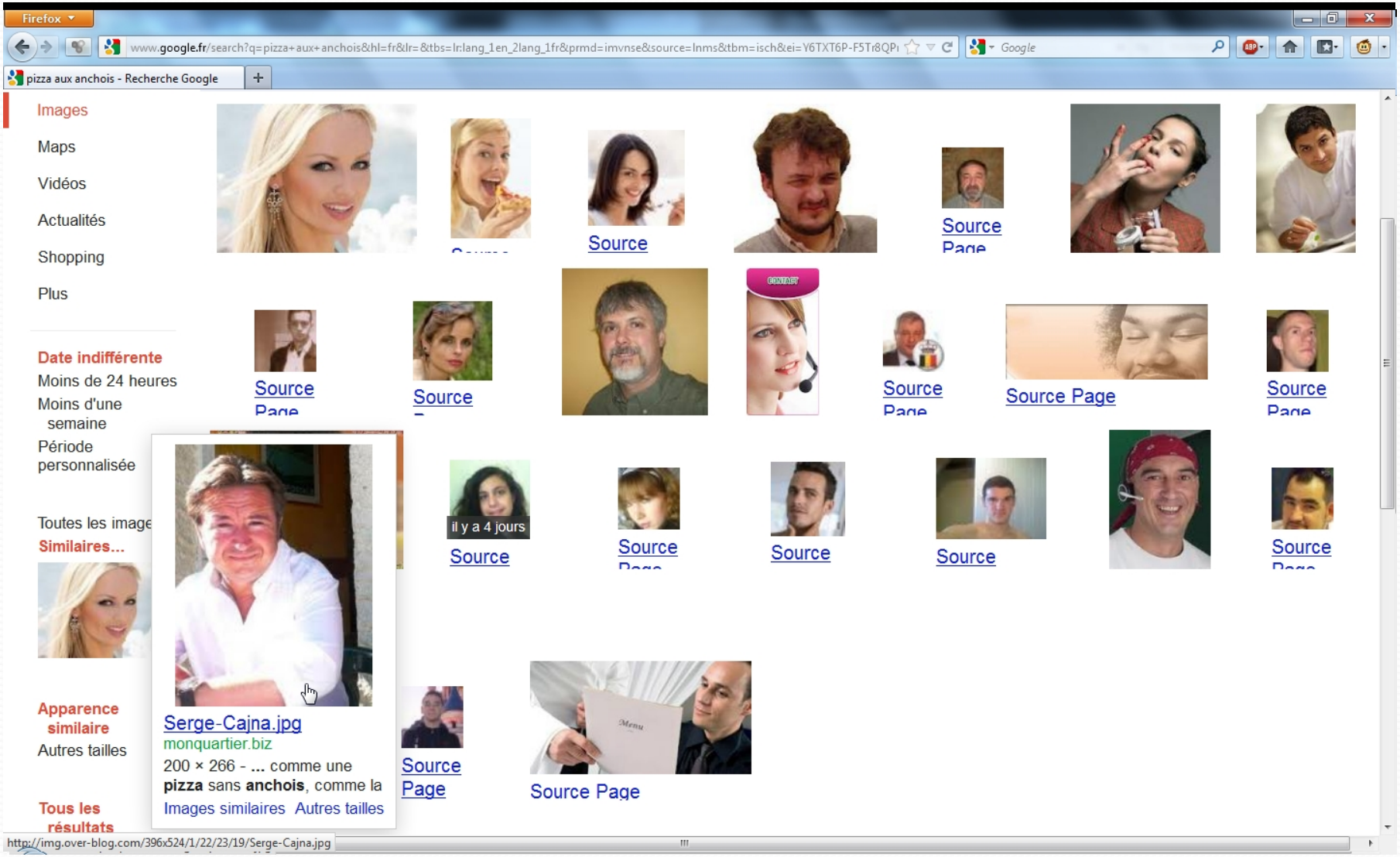
14/10/2014

Frédéric Precioso
Laboratoire I3S – UMR UNS-CNRS 7271
Pôle GLC – Equipe MinD

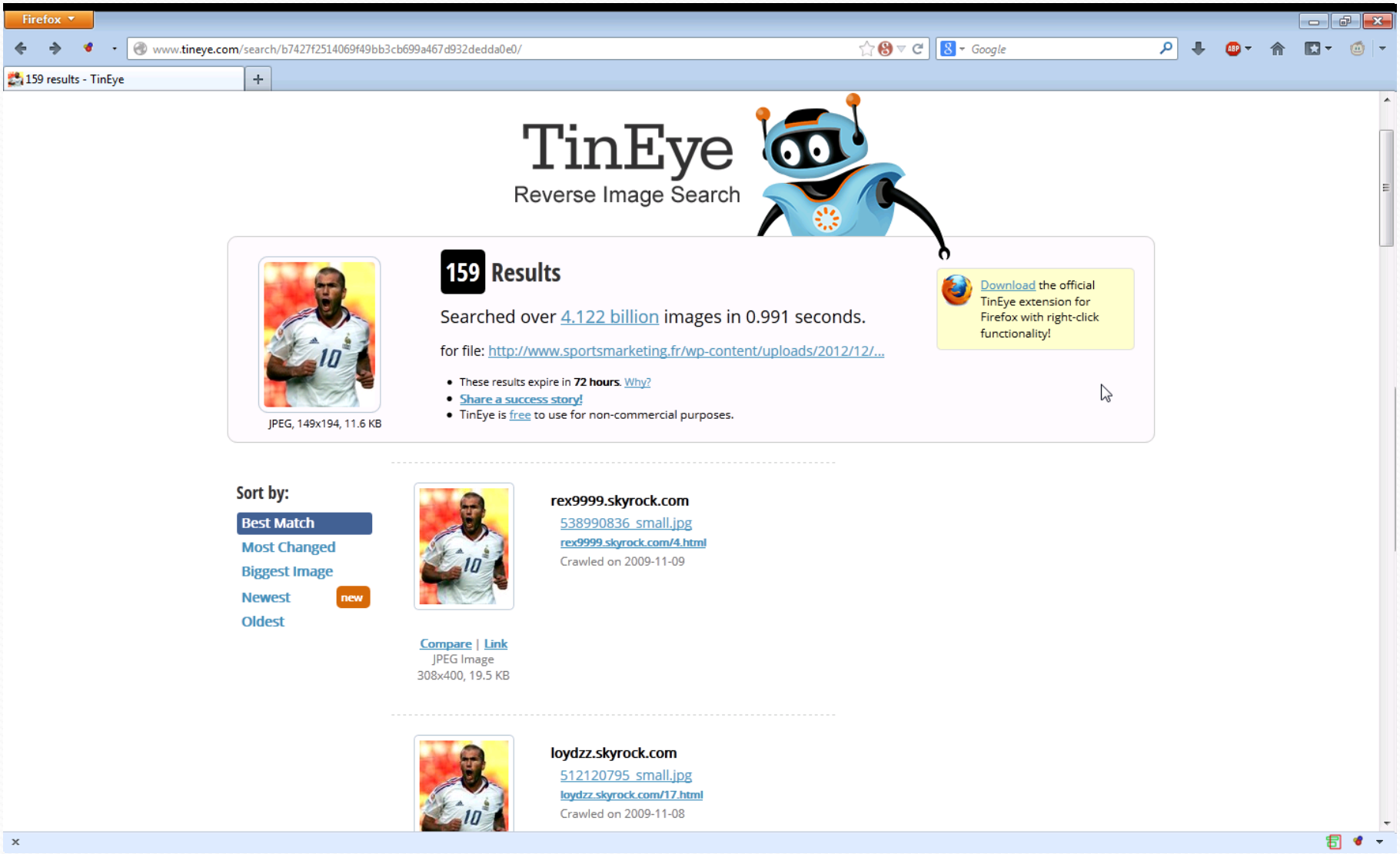
Outline

- The reality of the problem
- A classification problem
- First sucess-story:
  - Algorithm: Boosting
  - Application: Face detection
- Second success-story:
  - Algorithm: Support Vector Machines
  - Application: Pedestrian detection
- Third success-story:
  - Algorithm: Random Forests
  - Application: Microsoft Kinect
- Some other success-stories

Google Image results



TinEye results



Data mining in « The Big Data »

- The ***Digital Universe*** was about 281 exabytes (281 Billions of gigabytes) in 2007; it increased about 6 times, cracking the zettabyte barrier in 2010 (i.e. more than 1000 exabytes) **to pass 1.8 zettabyte in 2011** [1]
- **Images and videos**, captured by several billions of cameras, webcams, phones are the **biggest part** of this incredibly big amount of data [1]
  - **YouTube** = May 2010, more videos uploaded in 60 days than filmed by the 3 biggest US majors in 60 years [1]; 20 h of video uploaded per minute in May 2009, 24 h per minute in March 2010, to reach **1 h per second** few months ago [1,2]
- **IDC experts believe that by 2020, a third of the data in the digital universe (more than 13,000 exabytes) will have Big Data value, but only if it is tagged and analyzed.**
- **Medical and healthcare sensors are "obvious examples of new segments of big data growth"**

[1] International Data Corporation white papers

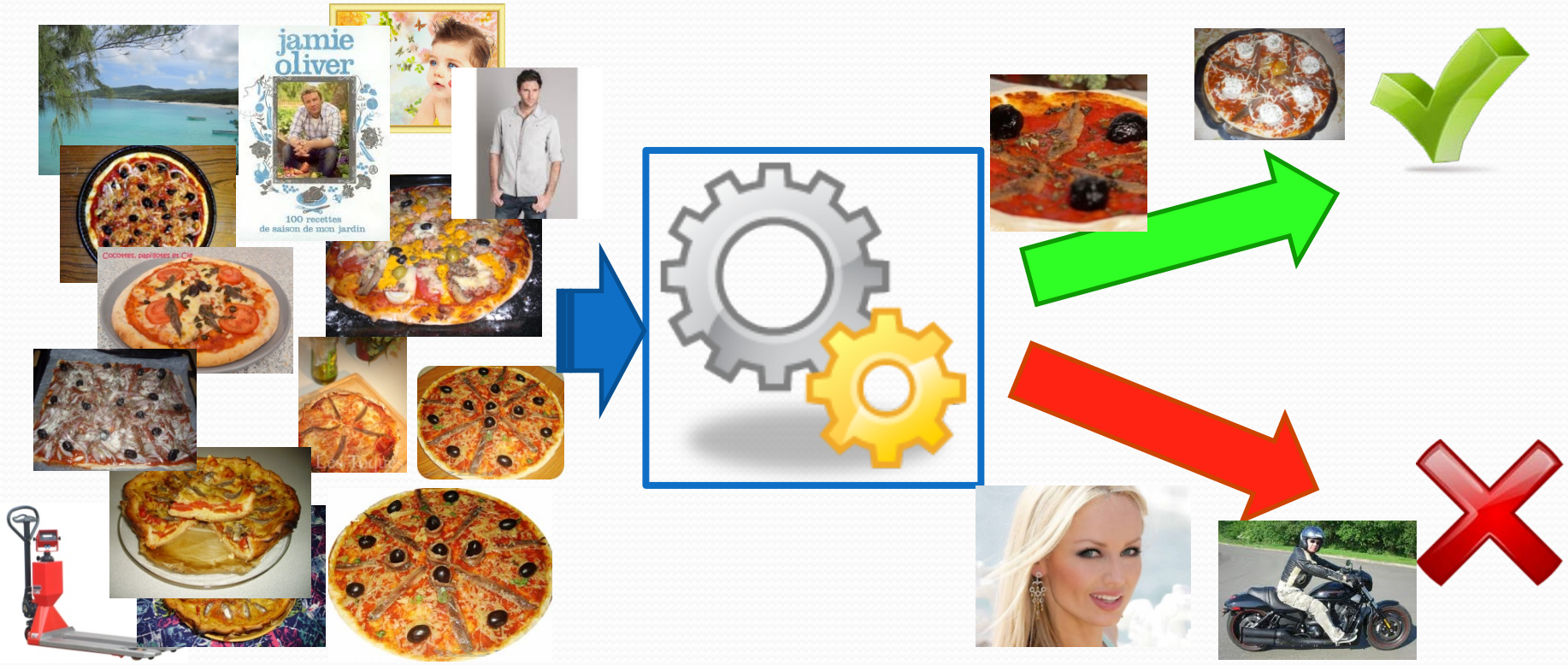
[2] http://www.onehourpersecond.com/

Outline

- The reality of the problem
- A classification problem
- First sucess-story:
  - Algorithm: Boosting
  - Application: Face detection
- Second success-story:
  - Algorithm: Support Vector Machines
  - Application: Pedestrian detection
- Third success-story:
  - Algorithm: Random Forests
  - Application: Microsoft Kinect
- Some other success-stories

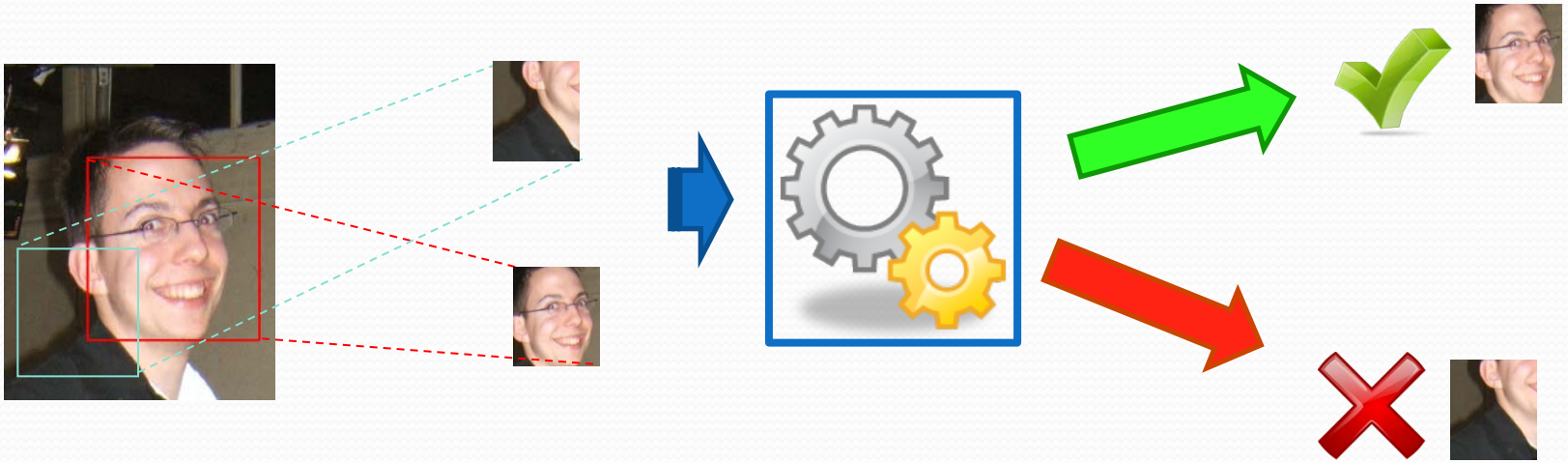
A classification problem

- All the questions we have mentioned, can be seen as a **binary classification** problem: separate the data **relevant *vs* irrelevant**.



A classification problem

- More specifically, in the applications we will develop, the objective is to detect regions which correspond to a certain category among data.
- This defines a binary classification:
  regions corresponding vs regions not corresponding to the target category.



A classification problem

- How to seperate the data ?

Algorithm

$$\text{Error}_{Real}(\text{Algorithm}) \leq \text{Error}_{Empirical}(\text{Algorithm}) + \text{Capacity}(\text{Algorithm})$$

Outline

- The reality of the problem
- A classification problem
- First sucess-story:
  - Algorithm: Boosting
  - Application: Face detection
- Second success-story:
  - Algorithm: Support Vector Machines
  - Application: Pedestrian detection
- Third success-story:
  - Algorithm: Random Forests
  - Application: Microsoft Kinect
- Some other success-stories

Ideas of boosting: Football Bets

If Ribéry and Ménez are happy together, French Football team wins.

If Diaby is not injured, French Football team wins.

If Benze
French

If Nasri is happy, French Football team wins.

How to win?

- Ask to professional gamblers
- Lets assume:
  - That professional gamblers can provide one single decision rule simple and relevant
  - But that face to several games, they can always provide decision rules a little bit better than random
- Can we become rich?

Idea

- Ask heuristics to the expert
- Gather a set of cases for which these heuristics fail (difficult cases)
- Ask again the expert to provide heuristics for the difficult cases
- And so one…

- Combine these heuristics
- expert stands for weak learner

Questions

- How to choose games (i.e. learning examples) at each step?
  - Focus on games (examples) the most “difficult” (the ones on which previous heuristics are the less relevant)
- How to merge heuristics (decision rules) into one single decision rule?
  - Take a weighted vote of all decision rules

Boosting

- boosting = general method to convert several poor decision rules into one very powerful decision rule
- More precisely:
  - Let have a weak learner which can always provide a decision rule with an error $\leq \frac{1}{2} - \gamma$
  - A boosting algorithm can build (theoretically) a global decision rule with an error rate $\leq \varepsilon$
- A theorem of Schapire on « weak learning power » proves that **H** gets a higher relevance than a global decision rule which would have been learnt directly on all training examples.

Probabilistic boosting

3 main ideas to generalize towards ***probabilistic boosting***:

1. A set of specialized experts and ask them to vote to take a decision.

2. Adaptive weighting of votes by multiplicative update.

3. Modifying example distribution to train each expert, increasing the weights iteratively of examples misclassified at previous iteration.

Probabilistic boosting: AdaBoost

The standard algorithm is **AdaBoost** (*Adaptive Boosting*).

The main idea is to define at each iteration $1< t < T$, a new distribution of *a priori probabilities* on training samples with respect to classification result at previous iteration.

The weight at iteration t of an example $(\boldsymbol{x}_i, y_i)$ of index i is denoted $D_t(i)$.

Initially, all the examples get a similar weight, at each iteration, weights of misclassified examples are increased

Thus the learner focuses on difficult examples of training set

Boosting: the algorithm

- A training set: $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$
- $y_i \in \{-1, +1\}$ label (annotation) of example $\mathbf{x}_i \in S$
- A set of weak learners $\{h_t\}$
- For $t = 0, \ldots, T$:
  - Build the distribution D_t on $\{1, \ldots, m\}$
  - Find the weak decision ("heuristic"): $h_t : S \rightarrow \{-1, +1\}$ with the smallest error ε_t on D_t :

$$\varepsilon_t = \Pr_{D_t}[h_t(\mathbf{x}_i) \neq y_i] = \sum_{i : h_t(\mathbf{x}_i) \neq y_i} D_t(i)$$

  - Compute α_t the influence/impact of h_t

- Final decision $H_{\text{final}} = \text{sgn}(\Sigma\, \alpha_t h_t)$

Boosting: the algorithm

- A training set: $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$
- $y_i \in \{-1,+1\}$ label (annotation)
- A set of weak learners
- For $t = 0,\ldots,T$:
  - Build the distribution D_t on $\{1,\ldots,m\}$
  - Find the weak decision ("heuristic"): $h_t : S \rightarrow \{-1,+1\}$ with the smallest error ε_t on D_t :

$$\varepsilon_t = \Pr_{D_t}[h_t(\mathbf{x}_i) \neq y_i] = \sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_t(i)$$

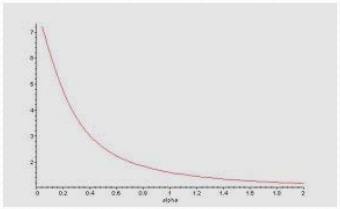
  - Compute α_t the influence/impact of h_t
- Final decision $H_{\text{final}} = \text{sgn}(\Sigma\, \alpha_t h_t)$

They are goal dependent.

Error of generalization for AdaBoost

- Error of generalization of H can be bounded by:

$$E_{\text{Real}}(H_T) = E_{HingeLoss}(H_T) + \text{O}\left(\sqrt{\frac{T.d}{m}}\right)$$



- where
  - ***T*** is the number of boosting iterations
  - ***m*** the number of training examples
  - ***d*** a dimension (VC-dimension) of H_T space (« weaks learner complexity »)

Advantages of AdaBoost

- (very) fast
- simple + easy to code
- One unique parameter to set: the number of boosting iterations (T)
- Can be use with any machine learning algorithm
- No-overfitting (margin theory)
- Is adapted to multi-class problems where $y_i \in \{1,..,c\}$ and to multi-label problems
- Allow to find out outliers

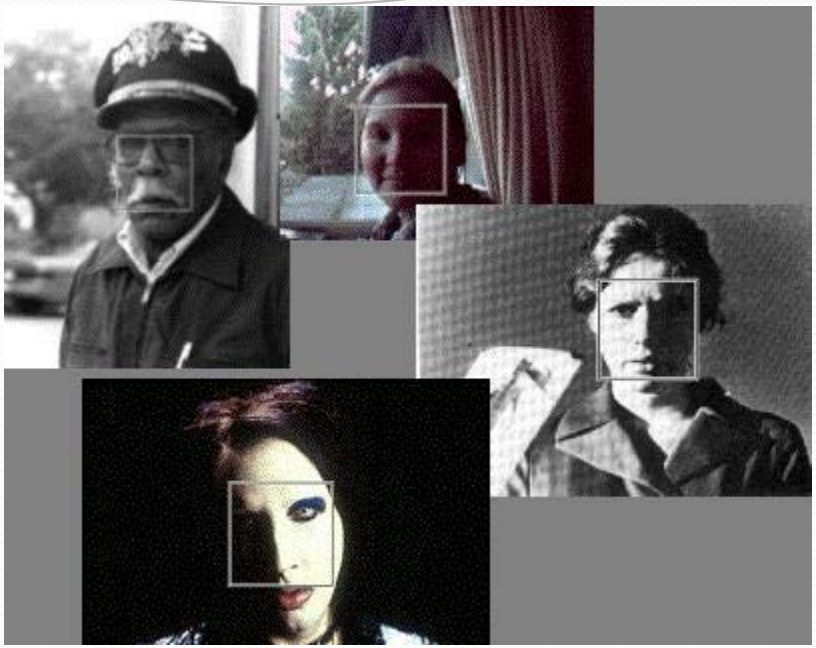
Drawbacks of AdaBoost

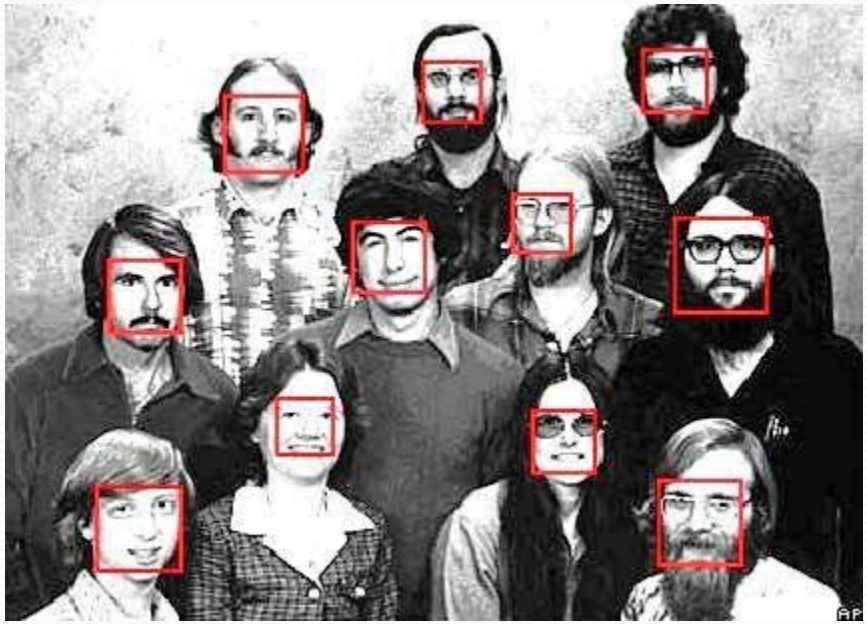
- Performances depend on training data & weak learners
- AdaBoost can fail if
  - The weak learner is to complex
    - Low margins → overfitting
  - The weak learner is too weak
    - $\gamma t \rightarrow 0$ too fast → underfitting
- AdaBoost seems specifically sensitive to noise

Outline

- The reality of the problem
- A classification problem
- First sucess-story:
  - Algorithm: Boosting
  - Application: Face detection
- Second success-story:
  - Algorithm: Support Vector Machines
  - Application: Pedestrian detection
- Third success-story:
  - Algorithm: Random Forests
  - Application: Microsoft Kinect
- Some other success-stories

The Task of Face Detection





Many slides adapted from P. Viola

Basic Idea

- Slide a window across image and evaluate a face model at every location.

Challenges

- Slide a window across image and evaluate a face model at every location.
- Sliding window detector must evaluate tens of thousands of location/scale combinations.
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A megapixel image has ~10^6 pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image image, our false positive rate has to be less than 10^{-6}

The Viola/Jones Face Detector

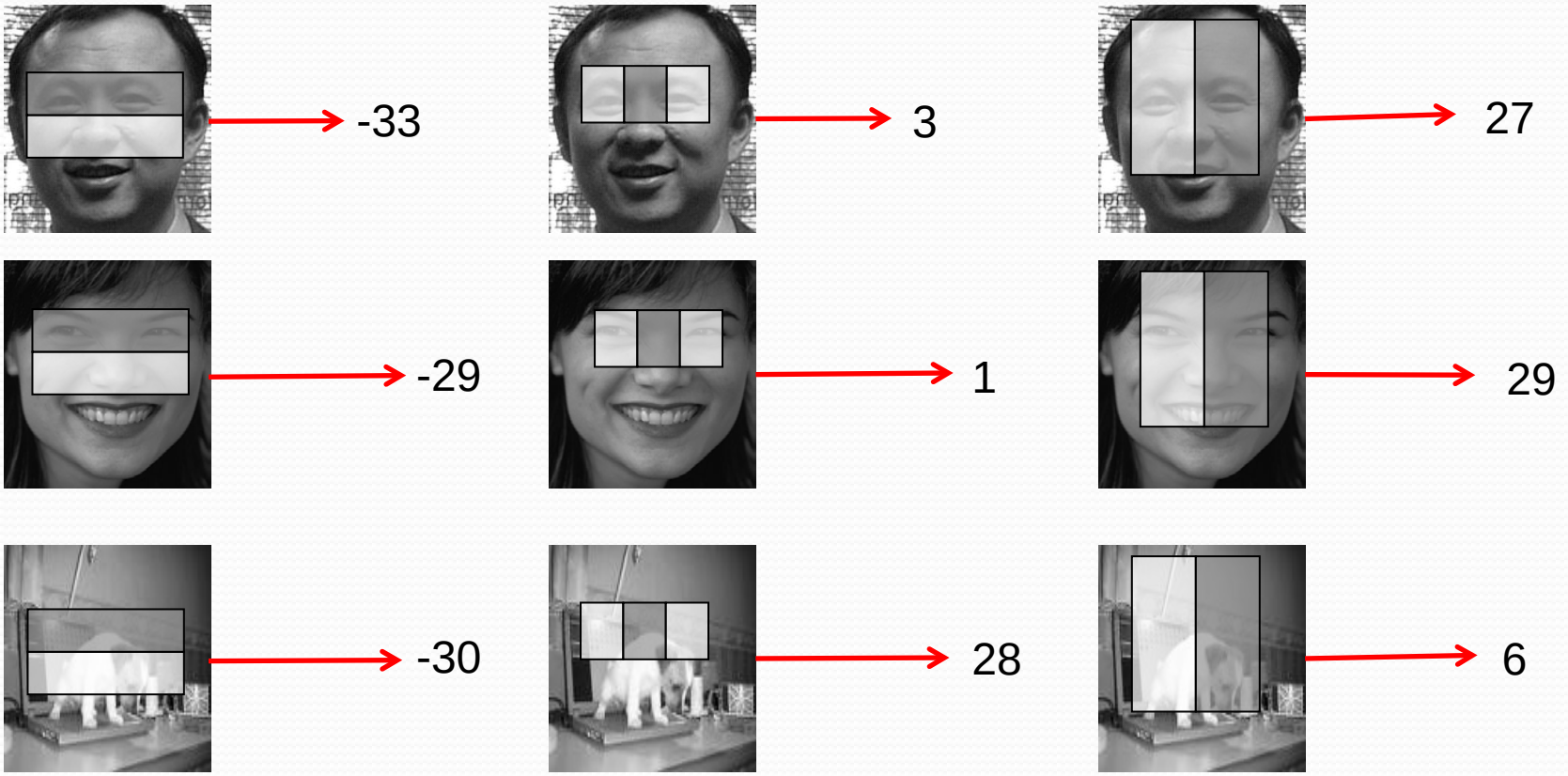
- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - Integral images for fast feature evaluation
  - Boosting for feature selection
  - Attentional cascade for fast rejection of non-face windows

P. Viola and M. Jones. ***Rapid object detection using a boosted cascade of simple features.*** CVPR 2001.

P. Viola and M. Jones. ***Robust real-time face detection.*** IJCV 57(2), 2004.

Image Features

Feature Value = $\sum$(Pixel in white area) − $\sum$(Pixel in black area)



Weak Learners for Face Detection

value of rectangle feature

parity

threshold

window

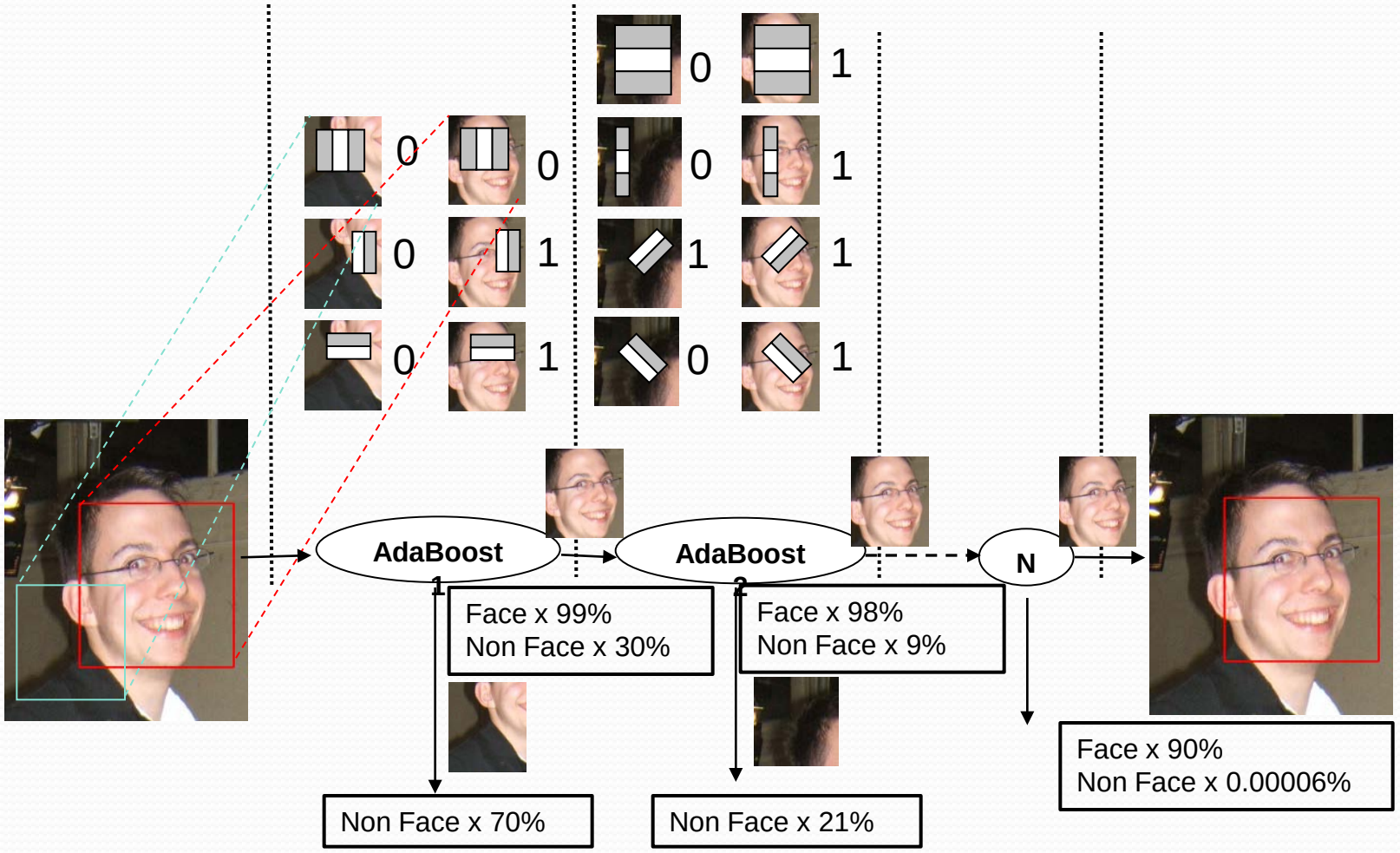
$$h_t(W) = \begin{cases} 1 & \text{if } p_t f_t(W) > p_t \theta_t \\ 0 & \text{otherwise} \end{cases}$$

$h_1($  $) = \begin{cases} 1 & \text{if } (-1)$  $> (-1)(-29) \\ 0 & \text{otherwise} \end{cases}$

$h_2($  $) = \begin{cases} 1 & \text{if } (-1)$  $> (-1)26 \\ 0 & \text{otherwise} \end{cases}$

$h_3($  $) = \begin{cases} 1 & \text{if } (+1)$  $> (+1)11 \\ 0 & \text{otherwise} \end{cases}$

AdaBoost
Cascade Principle



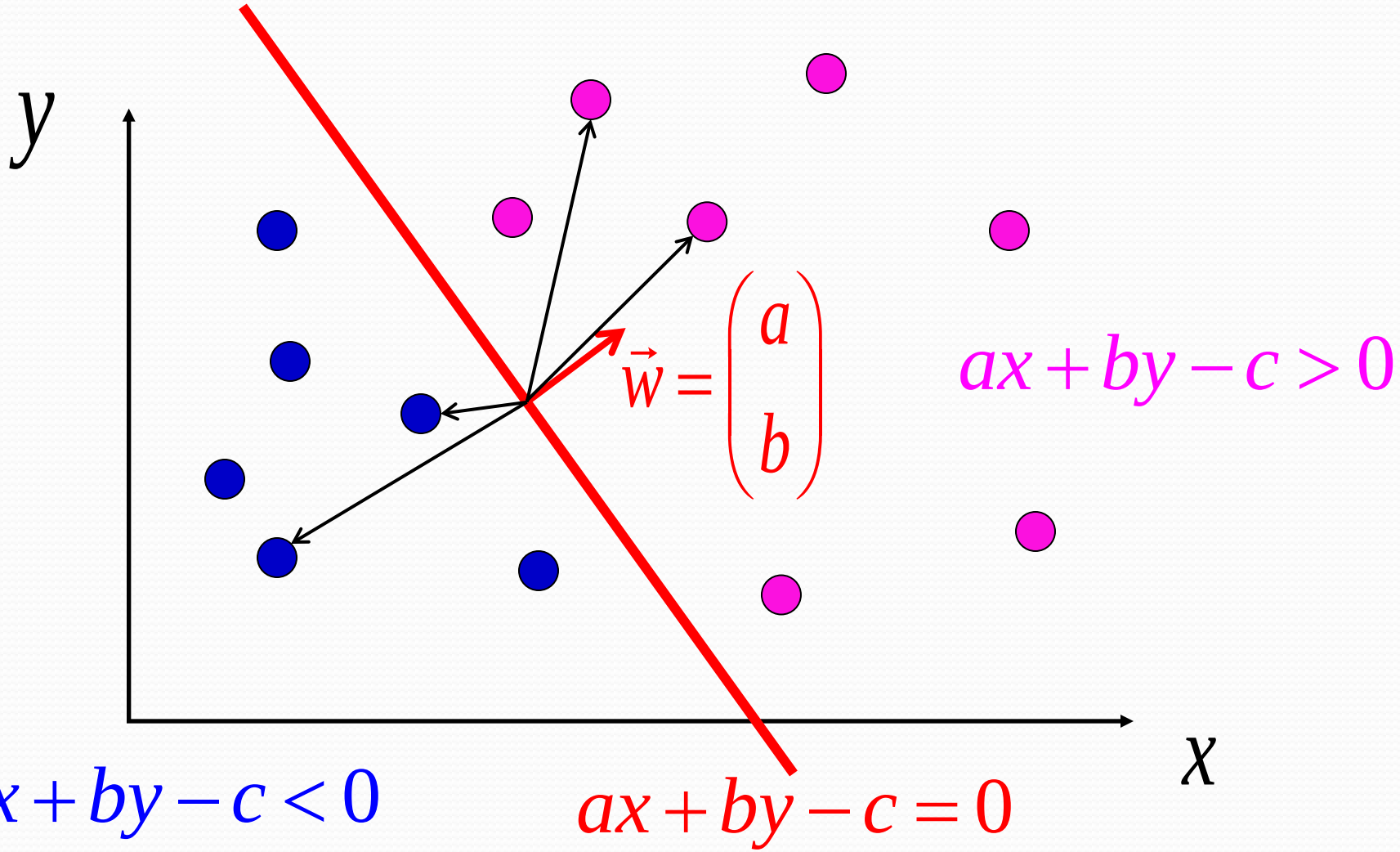
The Implemented System

- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces sub-windows
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

Outline

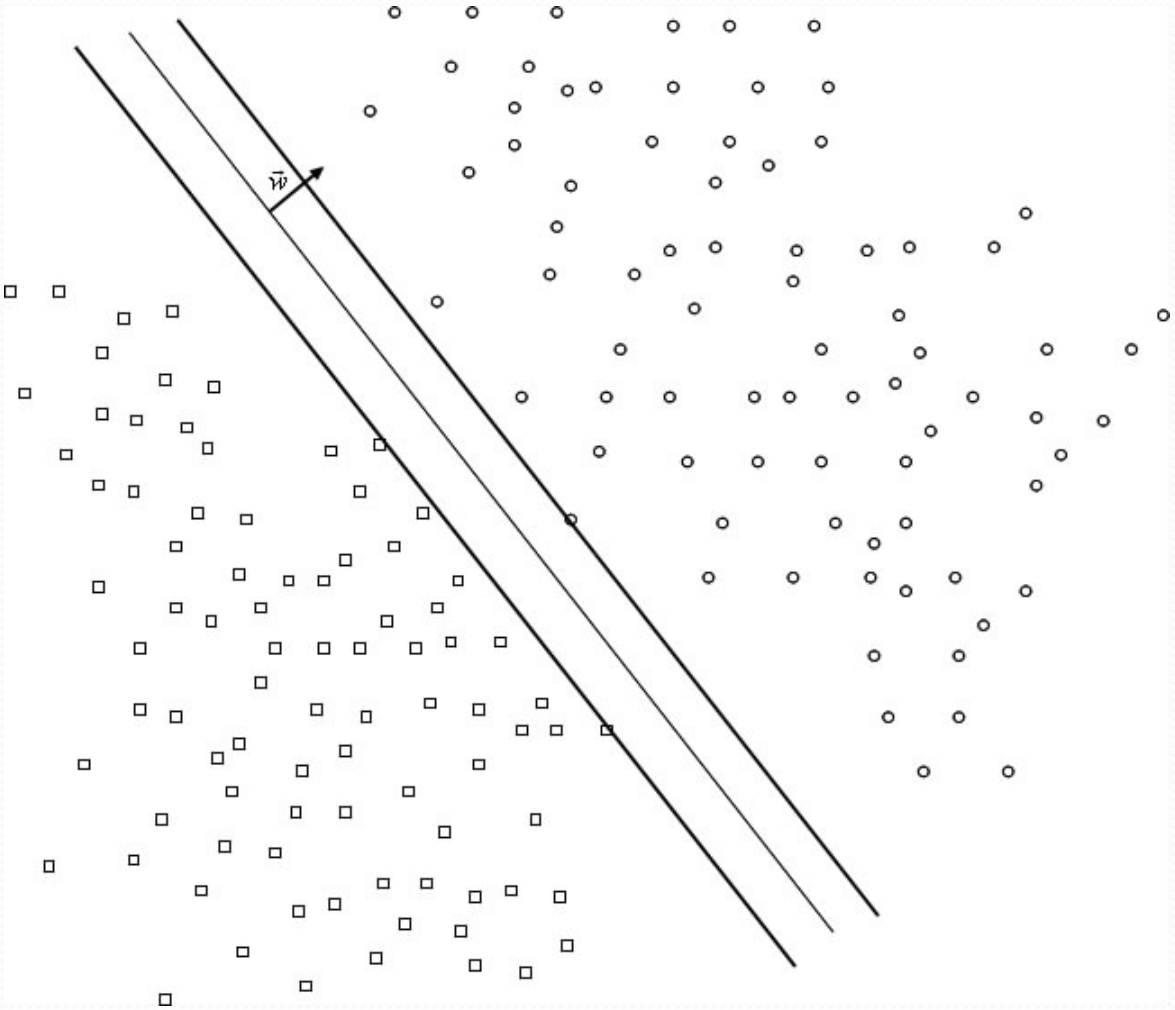
- The reality of the problem
- A classification problem
- First sucess-story:
  - Algorithm: Boosting
  - Application: Face detection
- Second success-story:
  - Algorithm: Support Vector Machines
  - Application: Pedestrian detection
- Third success-story:
  - Algorithm: Random Forests
  - Application: Microsoft Kinect
- Some other success-stories

Geometric approach

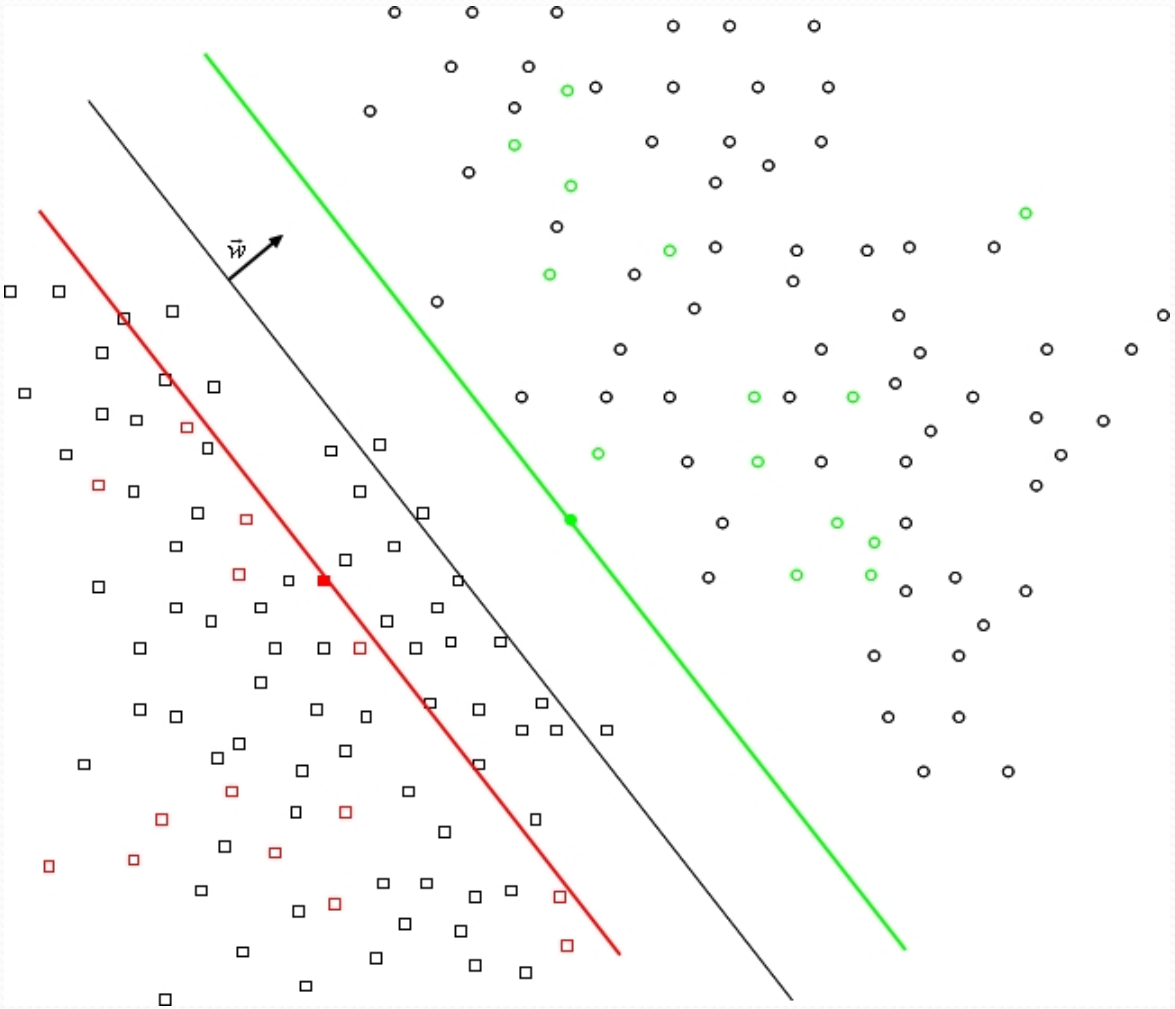


SVM: margin maximization

- Maximization the «margin» or corridor radius

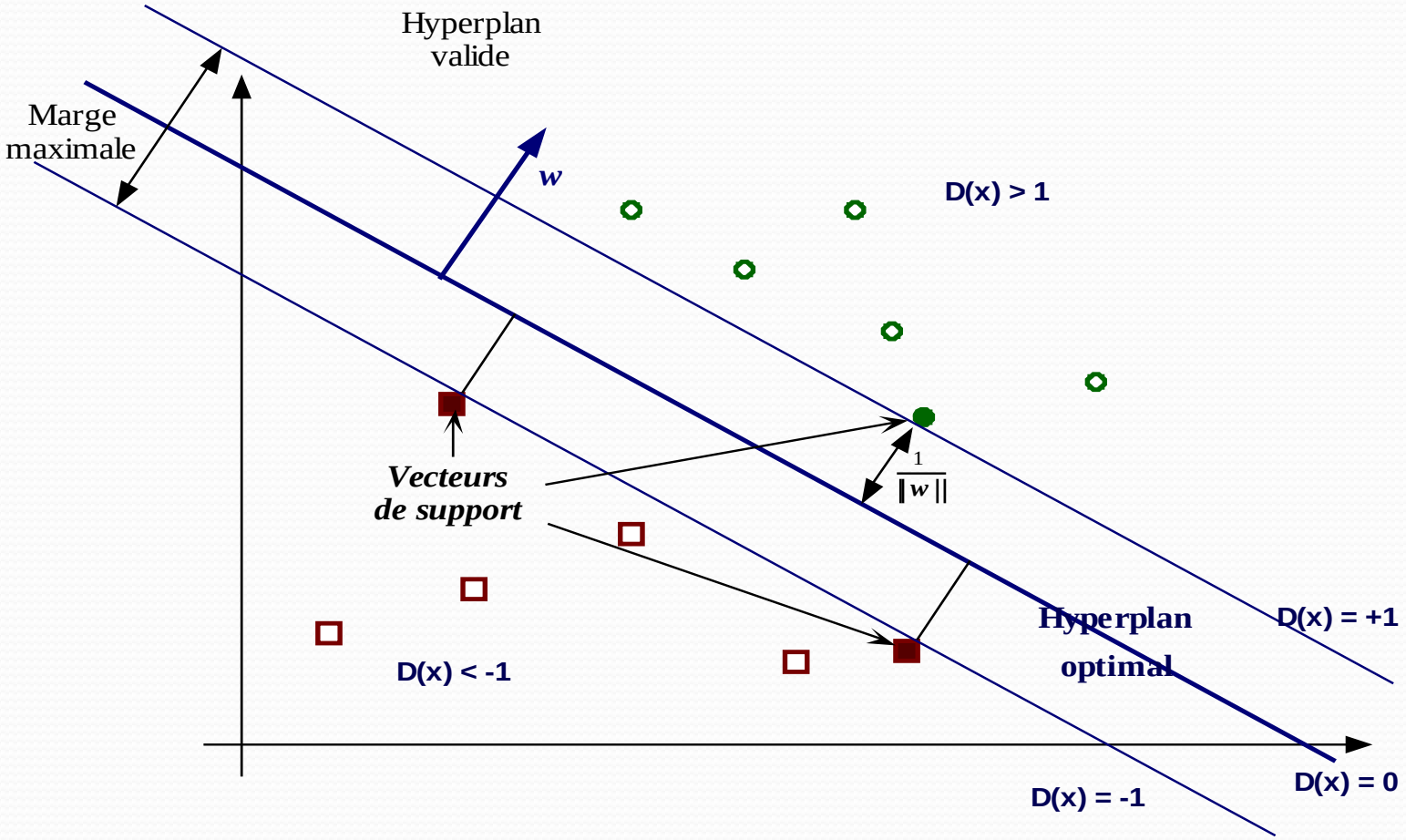


SVM: margin maximization



SVM: margin maximization

- Maximization of the «margin» or corridor radius: distance between the closest point and the hyperplan is 1/||w||



SVM: margin maximization

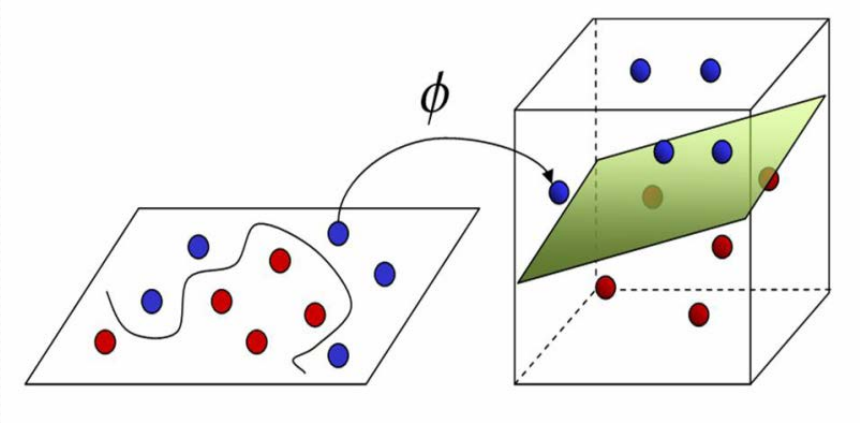
- Maximization of the «margin» or corridor radius: distance between the closest point and the hyperplan is 1/||w||
  - Under the constraints that all the training examples are well classified

$$\begin{cases} \min \quad \frac{1}{2}\| \mathbf{w} \|^2 \\ \forall i \quad y_i(\mathbf{w}.\mathbf{x_i} + w_0) \geq 1 \end{cases}$$

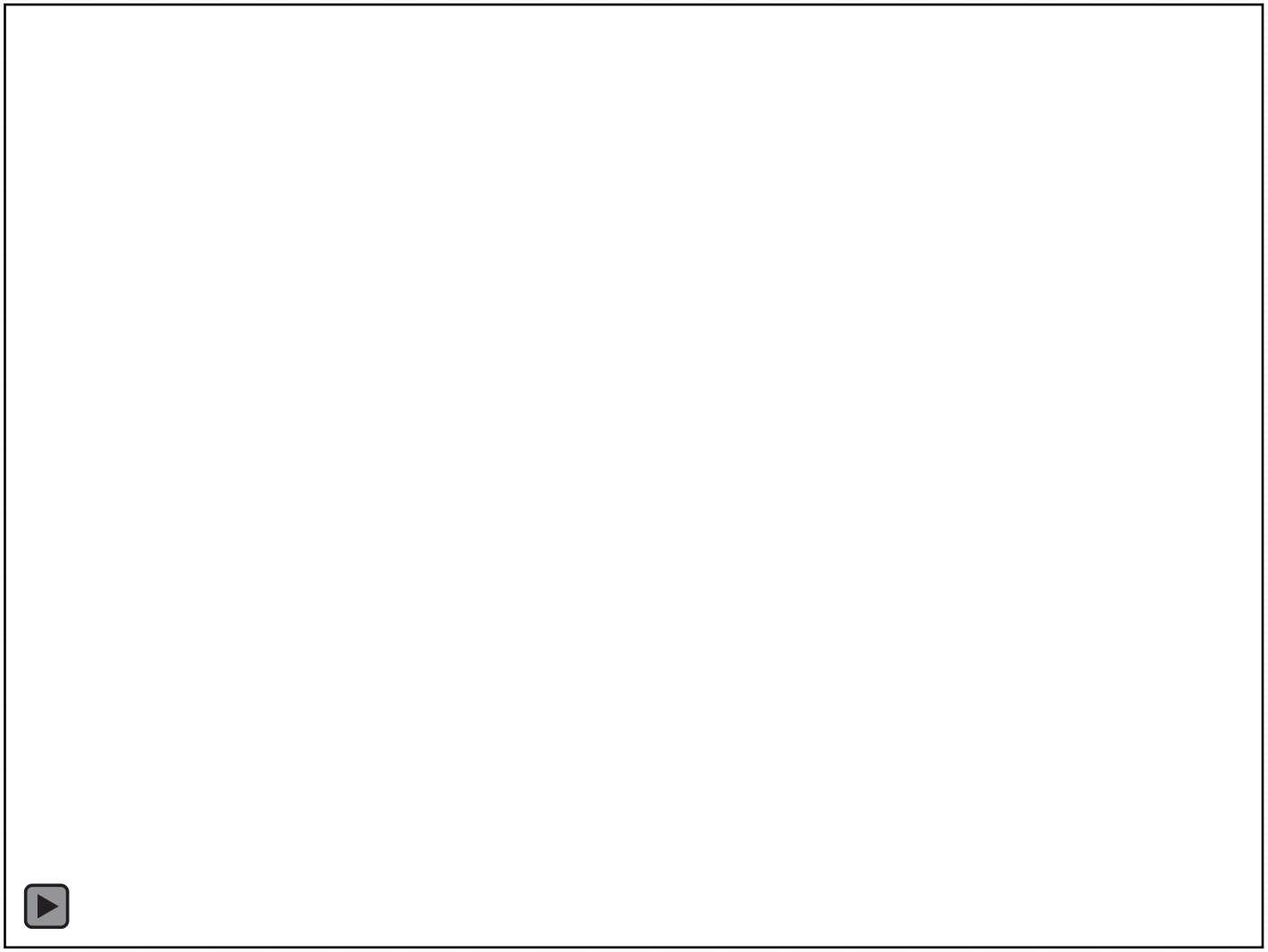
SVM: introduction to kernel

- Another possibility to overcome the problem of non-linearly seperable data in the *input space E*: change to a new space Φ, *the feature space*, of higher dimensions.

- A linear classifier in $\Phi(E)$ provides a non-linear classifier in E.



SVM: Kernel trick



SVM “conclusions”

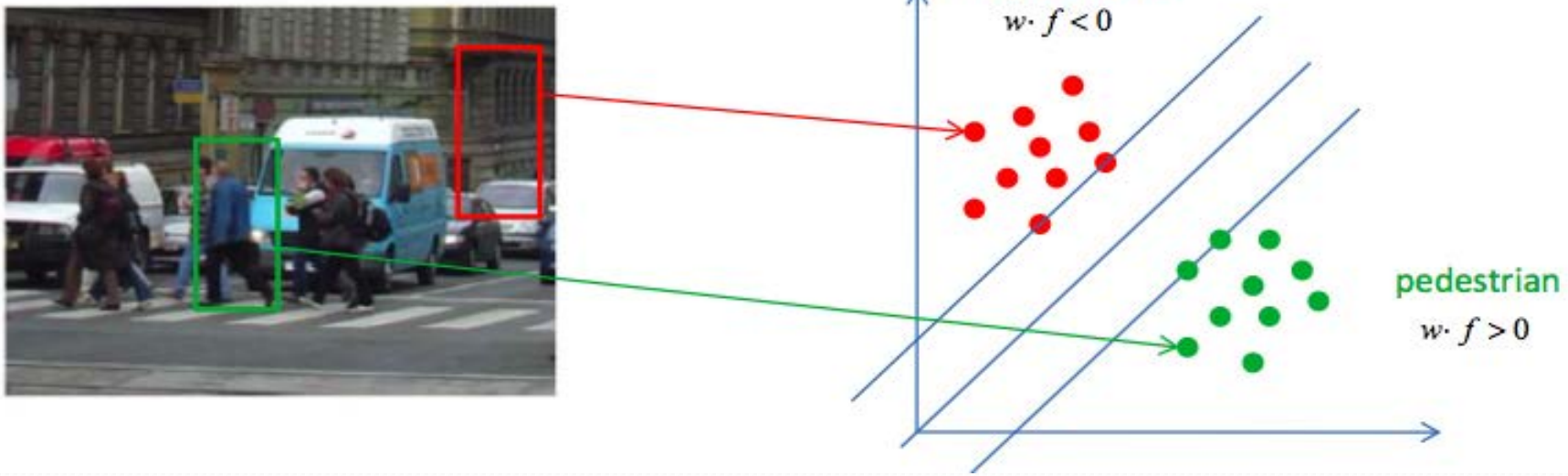
- Advantages
  - Only support vectors are required to take the decision
  - Convex problem => Global optimum
  - Easily non linear with the kernel trick

- Drawbacks
  - Most of the time, no heuristic to set up the parameters and the kernel
  - Intrinsically binary classification oriented => multi-class decision is doable but may not be optimal
  - Do not handle missing values

Outline

- The reality of the problem
- A classification problem
- First sucess-story:
  - Algorithm: Boosting
  - Application: Face detection
- Second success-story:
  - Algorithm: Support Vector Machines
  - Application: Pedestrian detection
- Third success-story:
  - Algorithm: Random Forests
  - Application: Microsoft Kinect
- Some other success-stories

Pedestrian detection



Typical form of a body model

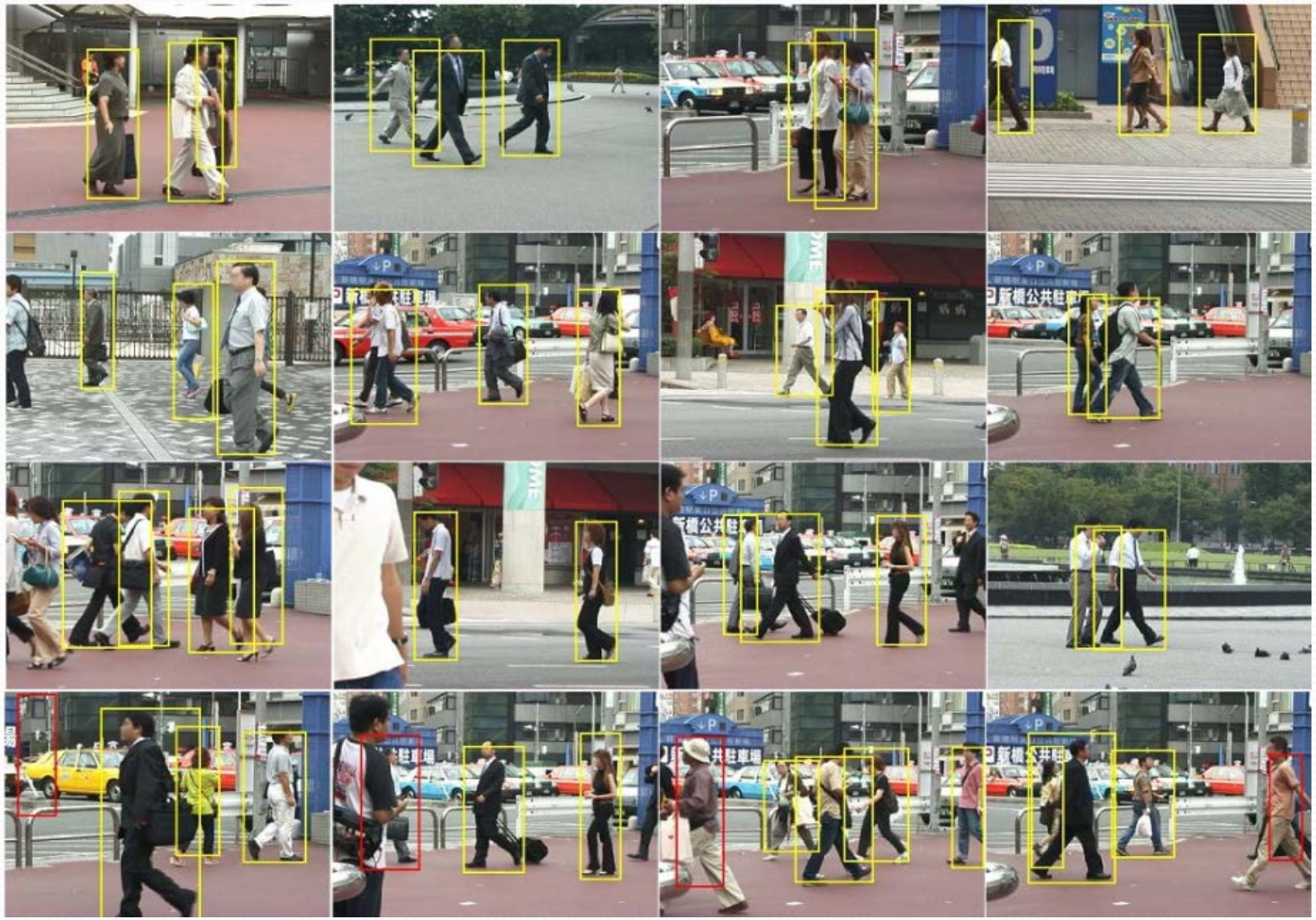
There is much more background than objects

Start with random negatives and repeat:

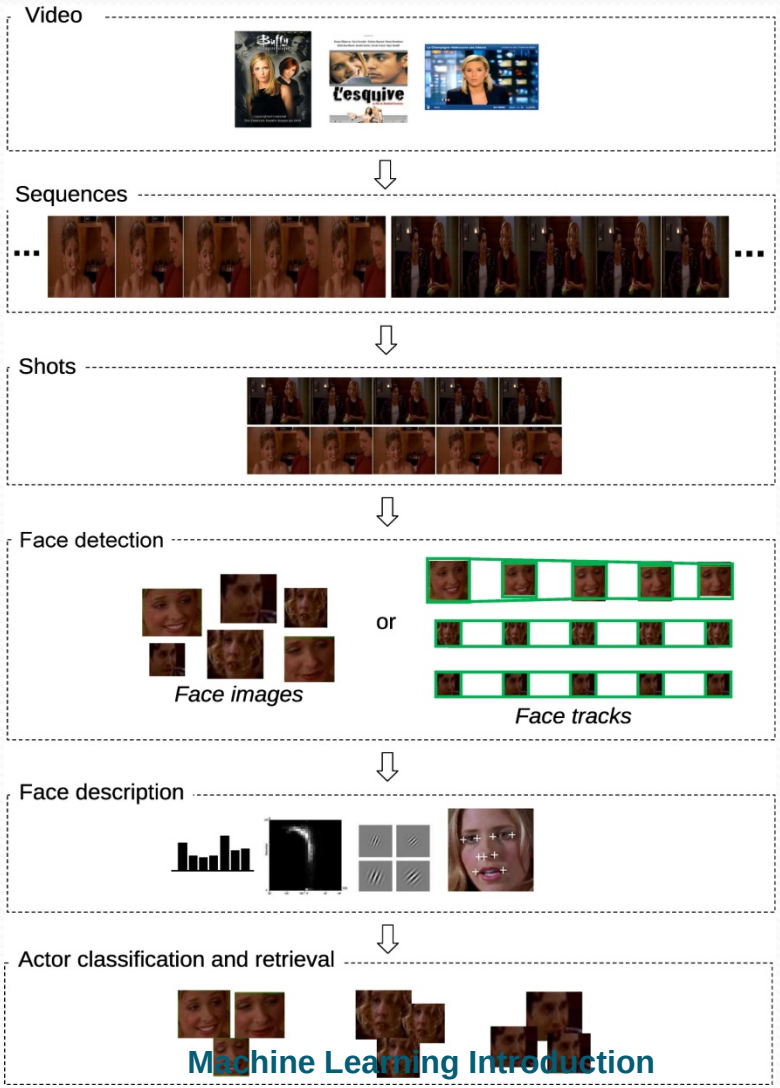
1) Train a model

2) Gather false positives to define “hard negatives”

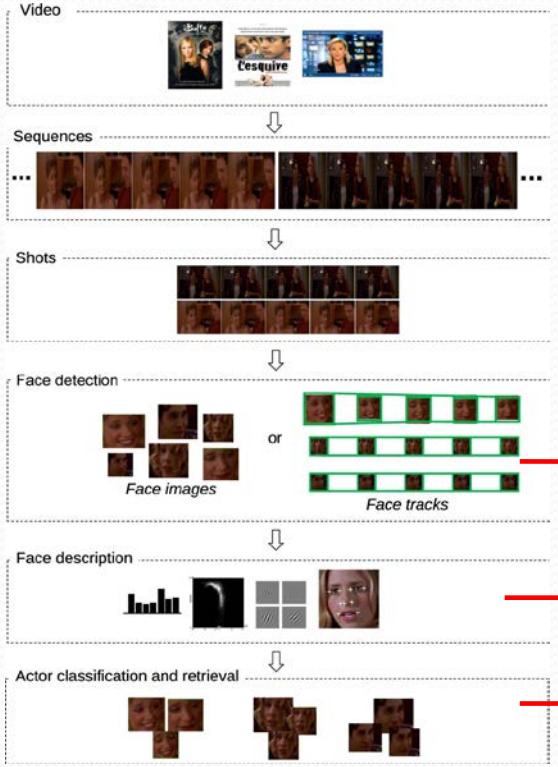
Pedestrian detection

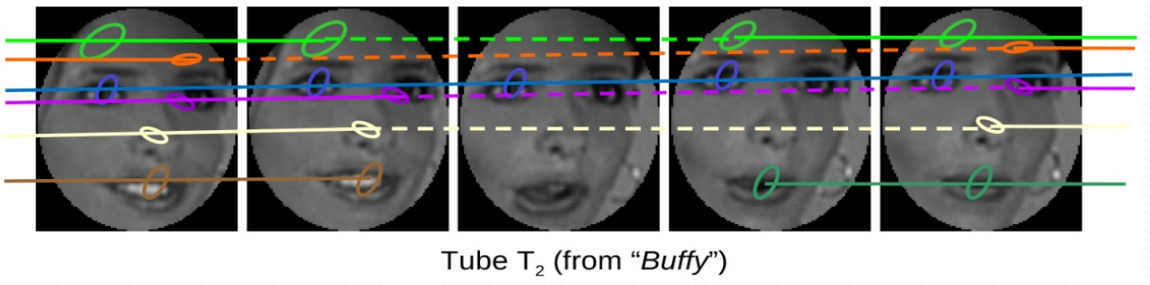


K-Videoscan


Video
Sequences
Shots
Face detection
Face images
or
Face tracks
Face description
Actor classification and retrieval

SVM and kernels

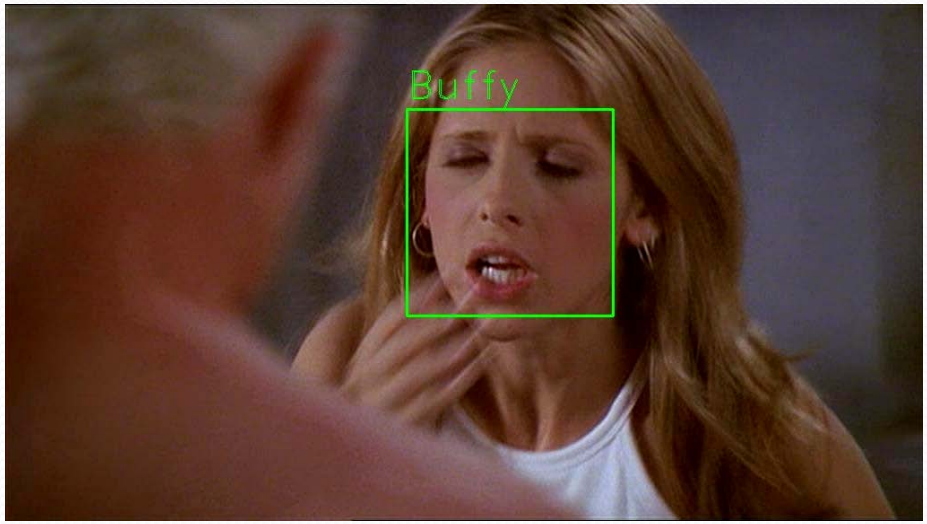




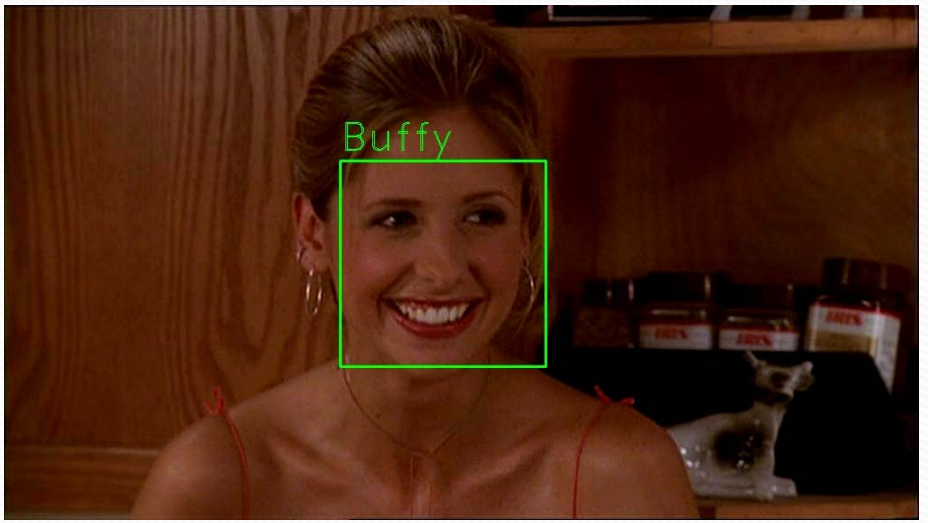
Tube T_2 (from "*Buffy*")

$$K(T_i,T_j) = \sum_r \sum_s \frac{|C_{ri}|}{\sqrt{|T_i|}} \frac{|C_{sj}|}{\sqrt{|T_j|}} e^{-\frac{q}{2\sigma_1^2}\frac{(\bar{C}_{ri}-\bar{C}_{sj})^2}{\bar{C}_{ri}+\bar{C}_{sj}}} e^{-\frac{(\bar{x}_{ri}-\bar{x}_{sj})^2+(\bar{y}_{ri}-\bar{y}_{sj})^2}{2\sigma_2^2}}$$

Actor recognition



Intra-episode



Inter-episode

Outline

- The reality of the problem
- A classification problem
- First sucess-story:
  - Algorithm: Boosting
  - Application: Face detection
- Second success-story:
  - Algorithm: Support Vector Machines
  - Application: Pedestrian detection
- Third success-story:
  - Algorithm: Random Forests
  - Application: Microsoft Kinect
- Some other success-stories

Decision tree for Tennis games

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Objective

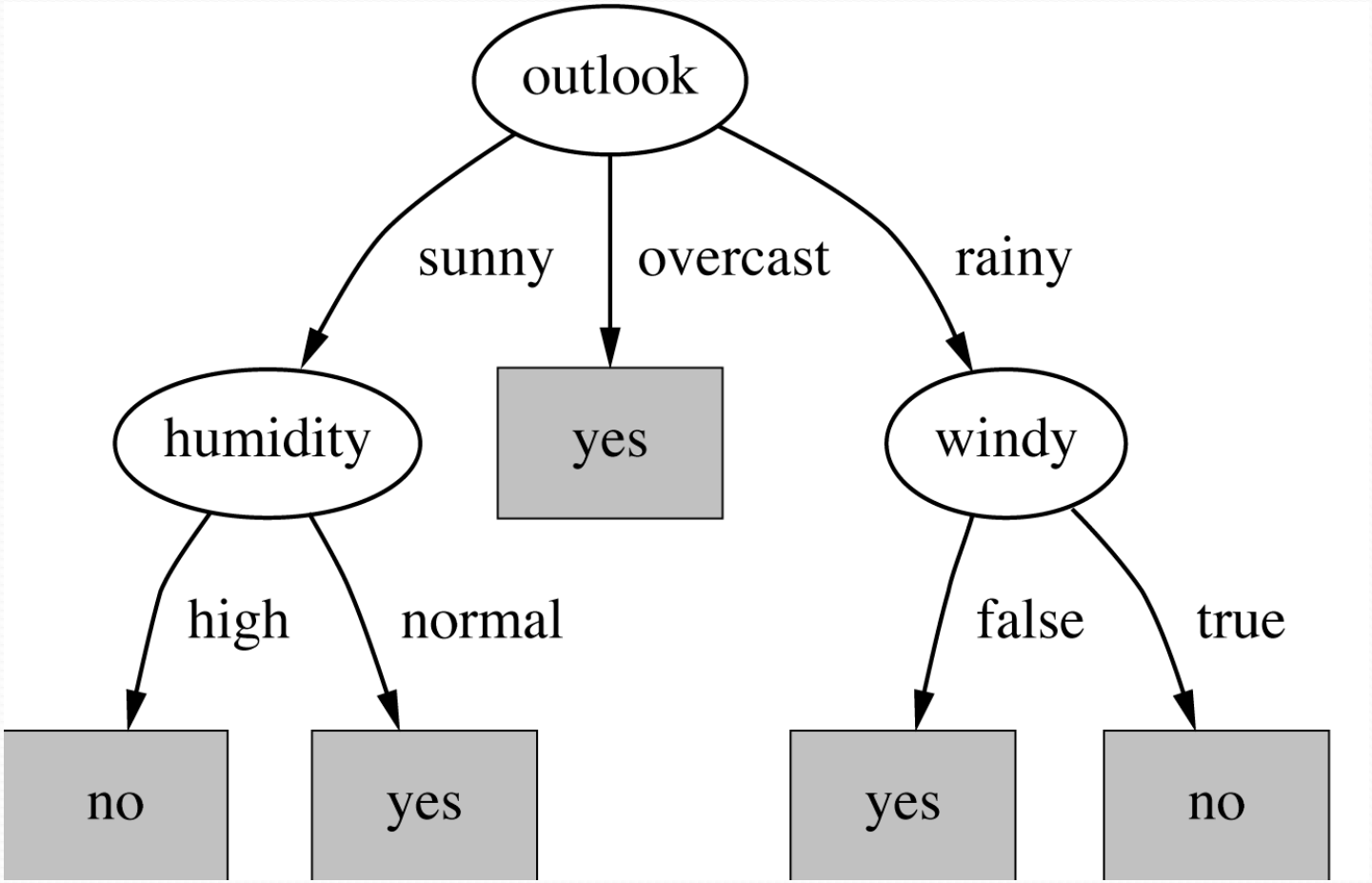
2 classes: yes & no

Prediction if a game will be played or not

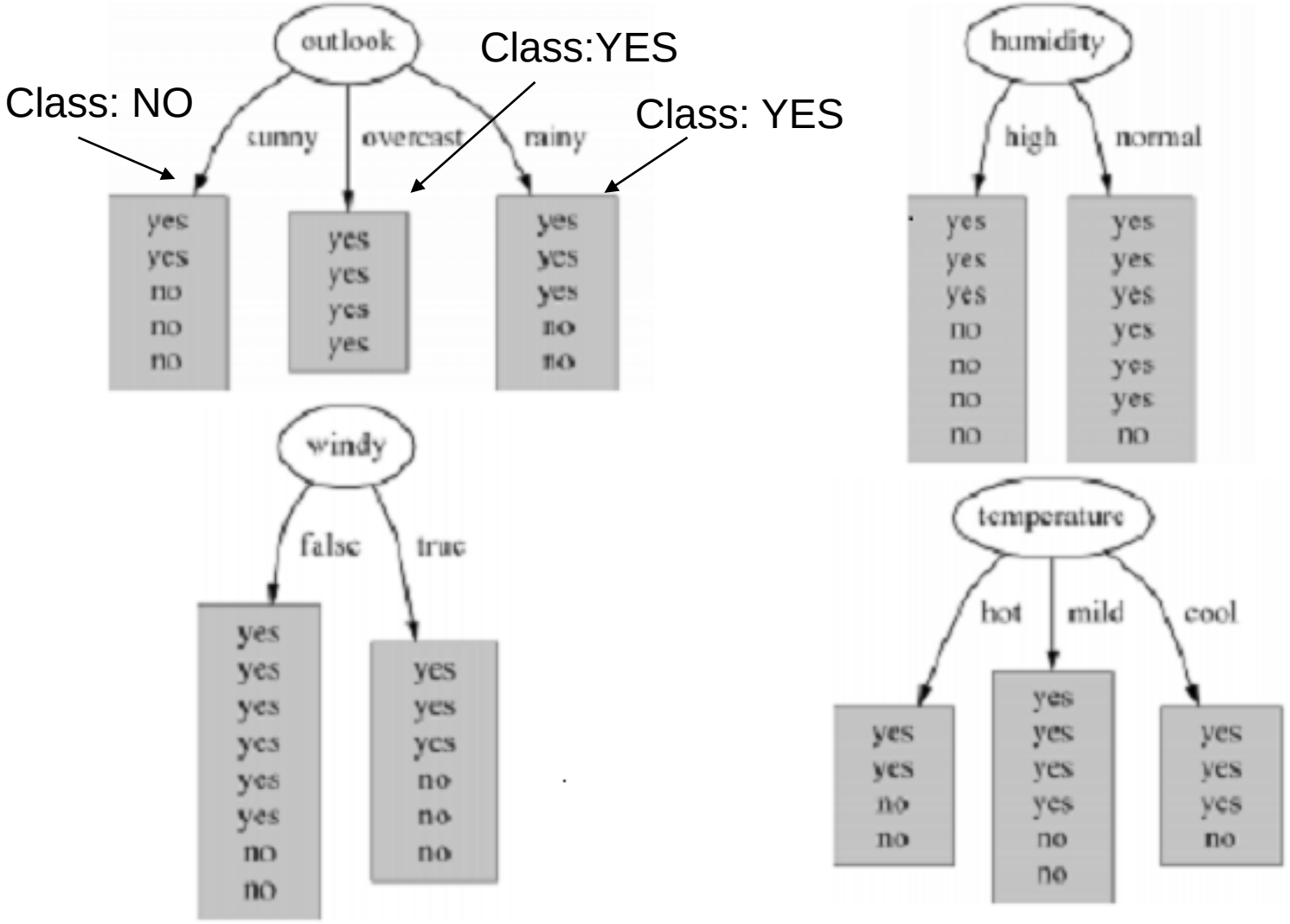
Temperature will be easily converted into numerical

I.H. Witten and E. Frank, “Data Mining”, Morgan Kaufmann Pub., 2000.

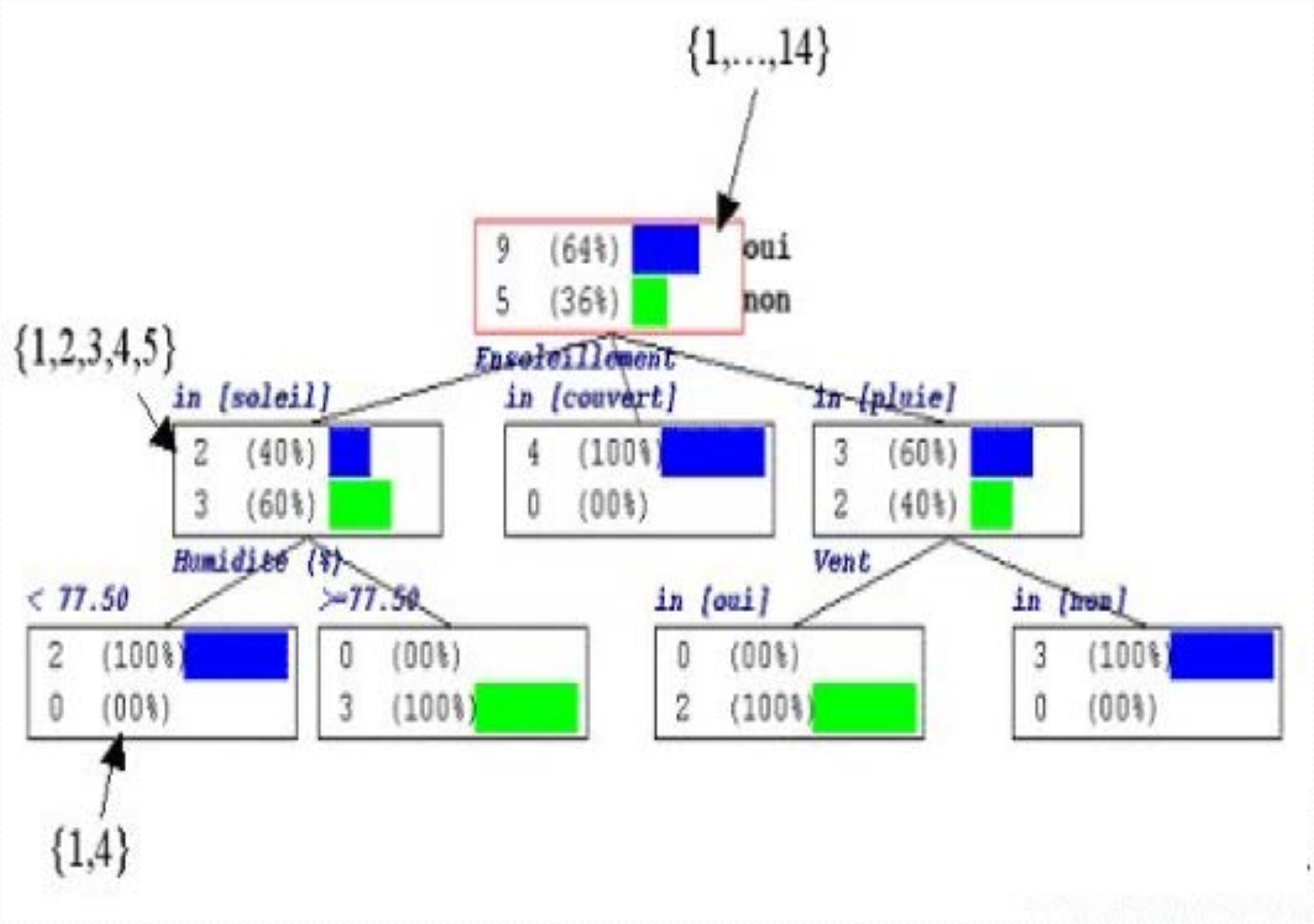
Final decision tree ?



Decision tree for Tennis games



Example


{1,...,14}
9 (64%)
5 (36%)
oui
non
Ensoleillement
{1,2,3,4,5}
in [soleil]
in [couvert]
in [pluie]
2 (40%)
3 (60%)
4 (100%)
0 (00%)
3 (60%)
2 (40%)
Humidité (%)
Vent
< 77.50
>=77.50
in [oui]
in [non]
2 (100%)
0 (00%)
0 (00%)
3 (100%)
0 (00%)
2 (100%)
3 (100%)
0 (00%)
{1,4}

Entropy in a Nutshell (from Andrew W. Moore)

Low Entropy

..the values (locations of soup) sampled entirely from within the soup bowl

High Entropy

..the values (locations of soup) unpredictable... almost uniformly sampled throughout the dining room

First Step: Information *Outlook*

Outlook = “Sunny”

$$\mathbf{Info}([2,3]) = \mathbf{Entropy}\left(\frac{2}{5},\frac{3}{5}\right)$$

$$\mathbf{Info}([2,3]) = \mathbf{Entropy}(0.4,0.6)$$

$$\mathbf{Info}([2,3]) = \frac{1}{\log 2}\left(-0.4\log(0.4) - 0.6\log(0.6)\right)$$

$$\mathbf{Info}([2,3]) = 0.971\,\mathbf{bits}$$

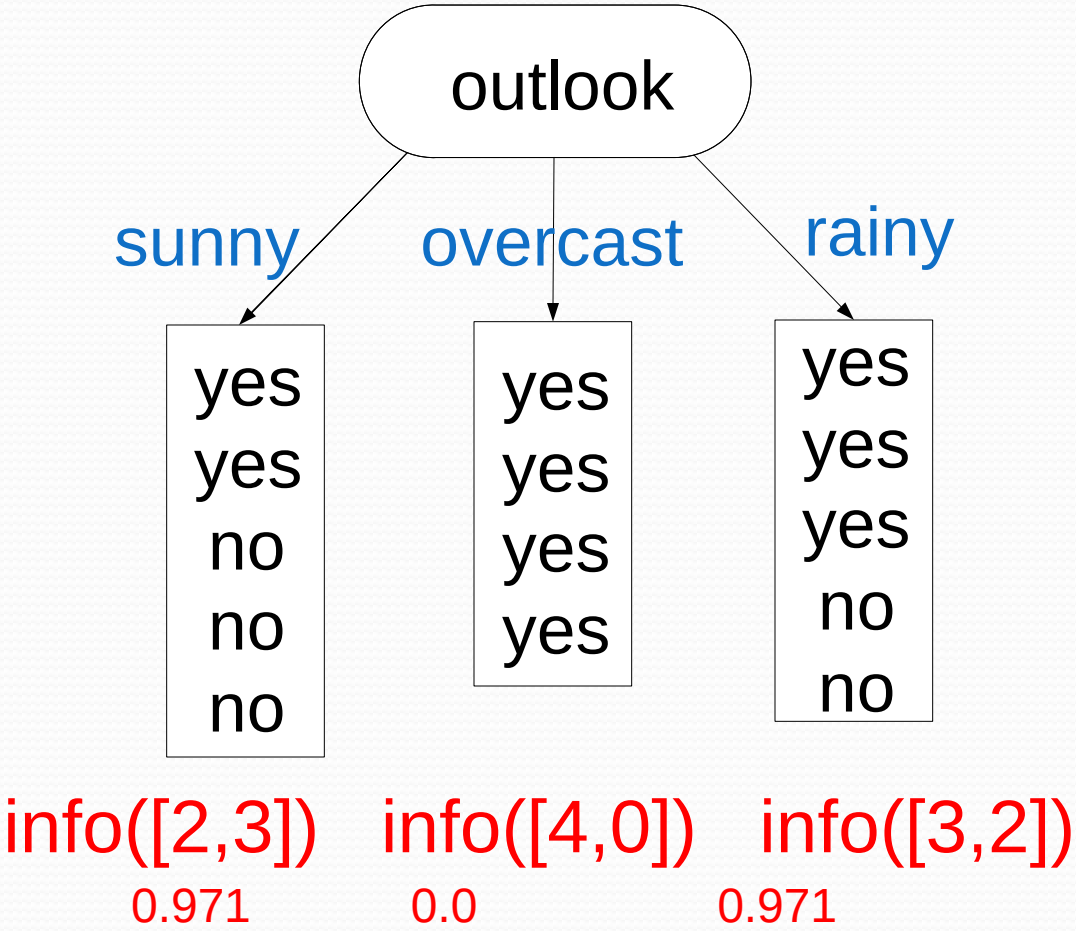
Similarly:

Outlook = “Overcast”

$$\mathbf{Info}([4,0]) = 0.0\,\mathbf{bits}$$

Outlook = “Rainy”

$$\mathbf{Info}([3,2]) = 0.971\,\mathbf{bits}$$


outlook
sunny
overcast
rainy
yes
yes
no
no
no
yes
yes
yes
yes
yes
yes
yes
no
no
info([2,3])
0.971
info([4,0])
0.0
info([3,2])
0.971

Information for the tree

Information for the full tree after choosing *Outlook*:

$$\textbf{Info}([2,3],[4,0],[3,2]) = \frac{5}{14}\textbf{Info}([2,3]) + \frac{4}{14}\textbf{Info}([4,0]) + \frac{5}{14}\textbf{Info}([3,2])$$

Info([2,3],[4,0],[3,2]) = 0.693

Information without considering tree structure

Outlook

$$\mathbf{Info}([9,5]) = \mathbf{Entropy}\left(\frac{9}{14}, \frac{5}{14}\right)$$

$$\mathbf{Info}([9,5]) = 0.940\ \mathbf{bits}$$

Information Gain for *Outlook*

$$\mathbf{gain(outlook)} = \mathbf{Info}([9,5]) - \mathbf{Info}([2,3],[4,0],[3,2])$$

$$\mathbf{gain(outlook)} = 0.940 - 0.693$$

$$\mathbf{gain(outlook)} = 0.247\ \mathbf{bits}$$

As well:

$$\mathbf{gain(temperature)} = 0.029\ \mathbf{bits}$$

$$\mathbf{gain(humidity)} = 0.152\ \mathbf{bits}$$

$$\mathbf{gain(windy)} = 0.048\ \mathbf{bits}$$

Outlook is chosen

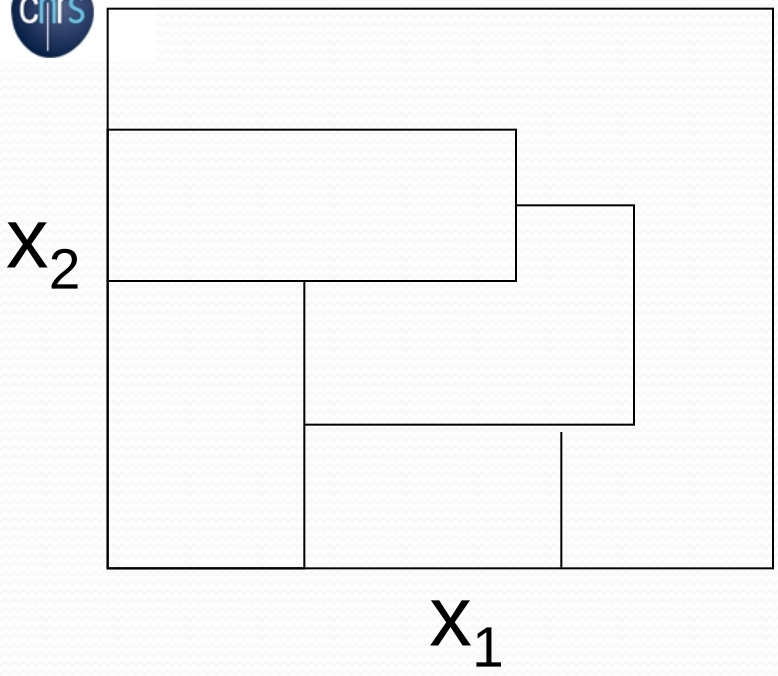
Avantages des arbres de décisions

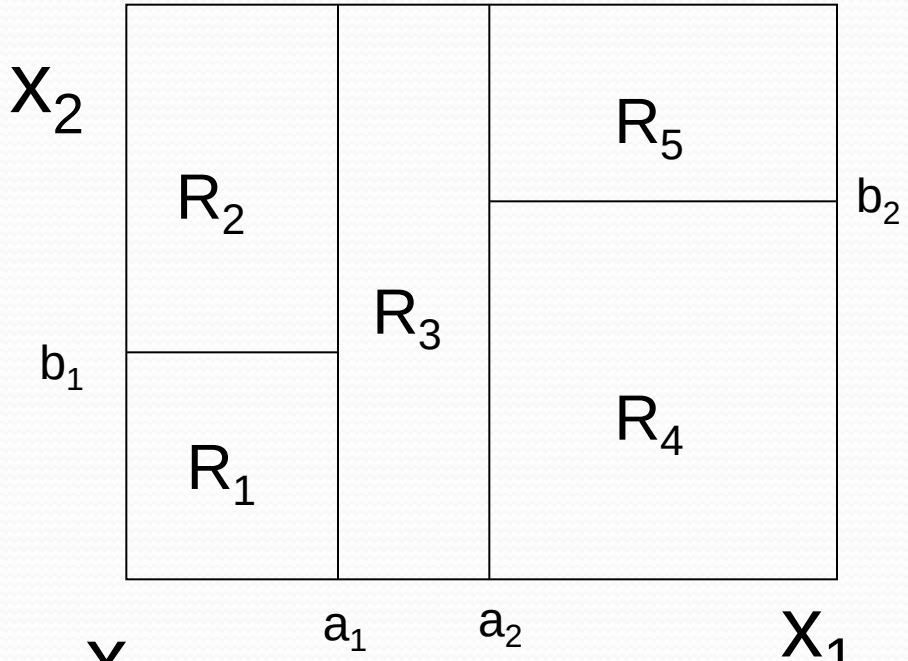
- Apprentissage supervisé
- Le résultat est lisible
  - Outils de navigation dans l'arbre
- Les valeurs manquantes peuvent être traitées
- Tous les types d'attributs peuvent être pris en compte
- Ils peuvent être utilisés comme prétraitement
- La classification d'un exemple est très efficace
- Moins efficace pour un nombre important de classes
- Ce n'est pas une méthode incrémentale

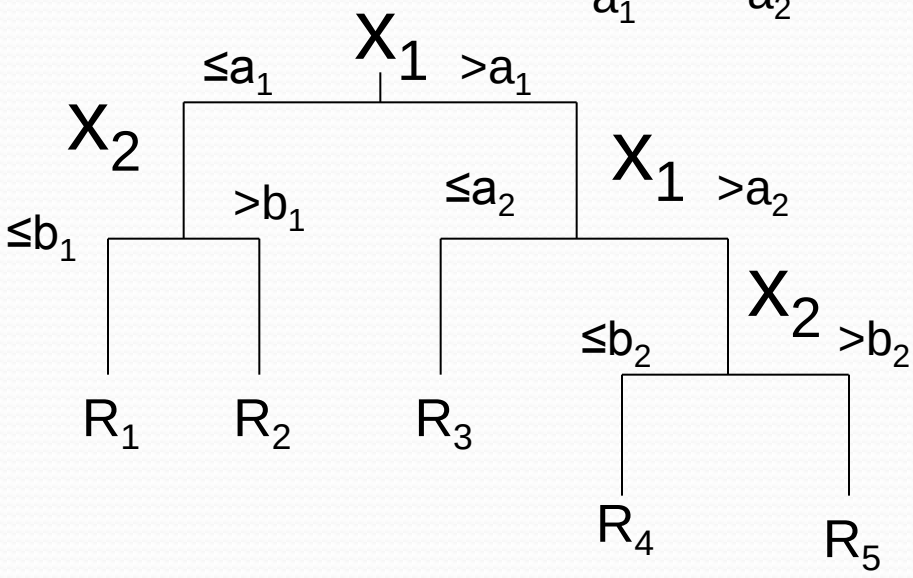
Limites des arbres de décision

- Test d'un seul attribut à la fois : coupes parallèles aux axes
- Non incrémental: recommencer la construction de l'arbre si on veut intégrer de nouvelles données
- Sensible à de petites variations dans les données (instable)
- Résultat fortement dépendant de l'ordre des attributs choisis pour créer l'arbre → la généralisation est donc difficile
- Trouver un arbre de décision d'erreur apparente minimale est, en général, un problème NPcomplet

==> méthodes pour améliorer l'apprentissage


X2
X1


X2
R2
R5
b2
R3
b1
R1
R4
a1
a2
X1


X1
≤a1
>a1
X2
X1
>b1
≤a2
>a2
≤b1
X2
≤b2
>b2
R1
R2
R3
R4
R5

Problème lié aux arbres de décision

- Les arbres ont connu un net regain d'intérêt lorsque les méthodes d'agrégation des classiffieurs tels que le boosting, le bagging et les Random Forests ont été développées pour l'apprentissage automatique
- Permettent de tirer parti des avantages de la combinaison des prédicteurs
- Idée principale: **on utilise le hasard pour améliorer les performances d'algorithmes de plus faibles performances**

Bootstrap Aggregating

- Autrement connu sous le nom de BAGGING
  - Principe
    - Étant donné un ensemble d'apprentissage D de taille n, on génère m nouveaux ensembles D_i de taille $n' \leq n$ en échantillonnant uniformément les exemples de D avec remise.
    - Les m modèles sont entraînés en utilisant les m ensembles les réponses des modèles sont combinées (moyenne ou vote)
  - Technique d'apprentissage visant à
    - Améliorer la stabilité
    - Réduire la variance
    - Éviter le surapprentissage (overfitting)
  - Utilisable pour n'importe quel type de modèle
  - Utilisé surtout pour les arbres de décision

Définition

- Définition de Léo Breiman (2001)
  - Une Forêt Aléatoire est un classifieur constitué d'un ensemble de classifieurs élémentaires de type arbres de décision, noté $\{ h(x, \Theta_k), \quad k = 1, ...L \}$ où $\{\Theta_k\}$ est une famille de vecteurs aléatoires indépendants et identiquement distribués, et au sein duquel chaque arbre participe au vote de la classe la plus populaire pour une donnée d'entrée inconnue ***x***

Error of generalization for Random Forest

- Error of generalization of H can be bounded by:

$$E_{\text{Real}}(RF) \leq \frac{\rho(1-s^2)}{s^2}$$

- where
  - ρ is the mean correlation between two decision trees
  - s is the value of prediction of the set of decision trees

Données importantes

- Les données à paramétrer
  - Nombre d'arbres de la forêt
  - Profondeur des arbres
  - Nombre de paramètres considérés pour construire chaque arbre
  - Choix des paramètres parmi l'ensemble des paramètres initiaux

Principe des forêts aléatoires

- Avantages
  - Réduction de la variance (influence des données)
  - Simple à mettre en œuvre
- Inconvénients
  - Temps de calcul plus important
  - Interprétabilité diminuée
- Introduction du caractère aléatoire : objectif de rendre les modèles (arbres) plus indépendant entre eux

==> vote des experts (les CART) plus efficace

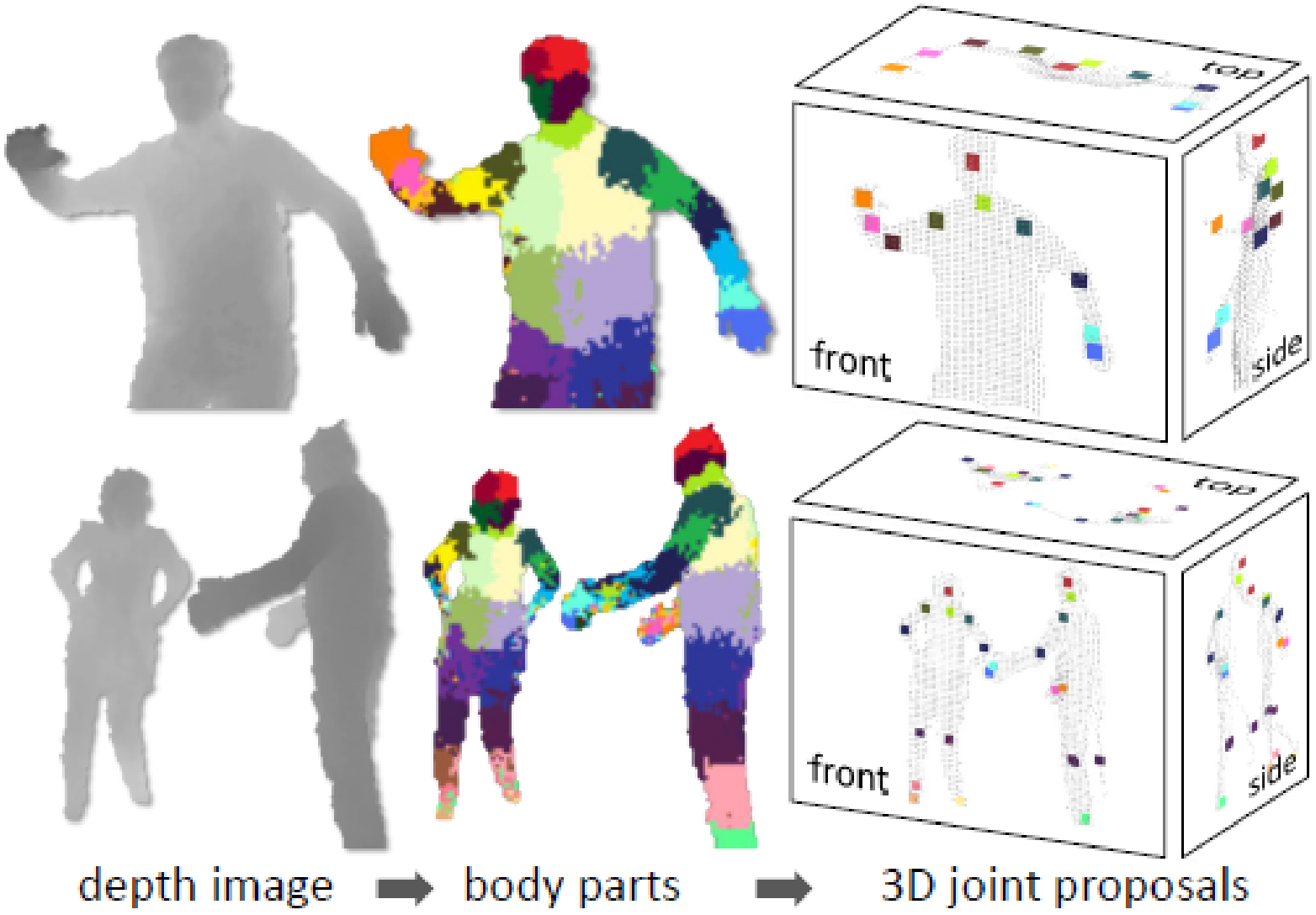
Outline

- The reality of the problem
- A classification problem
- First sucess-story:
  - Algorithm: Boosting
  - Application: Face detection
- Second success-story:
  - Algorithm: Support Vector Machines
  - Application: Pedestrian detection
- Third success-story:
  - Algorithm: Random Forests
  - Application: Microsoft Kinect
- Some other success-stories

Success story: Kinect

Kinect Movie

Success story: Kinect



Success story: Kinect

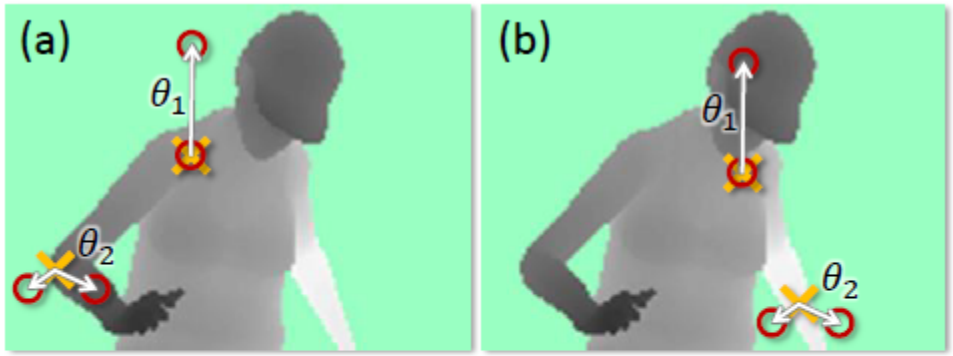


Figure 3. **Depth image features.** The yellow crosses indicates the pixel x being classified. The red circles indicate the offset pixels as defined in Eq. 1. In (a), the two example features give a large depth difference response. In (b), the same two features at new image locations give a much smaller response

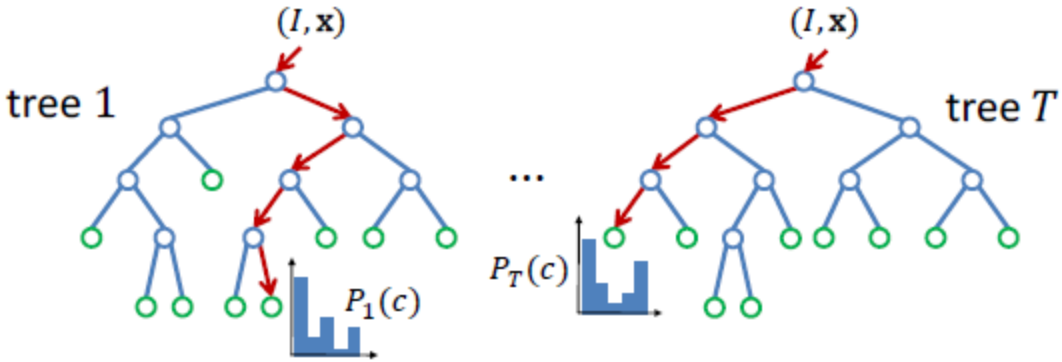


Figure 4. **Randomized Decision Forests.** A forest is an ensemble of trees. Each tree consists of split nodes (blue) and leaf nodes (green). The red arrows indicate the different paths that might be taken by different trees for a particular input.

Success story: Kinect

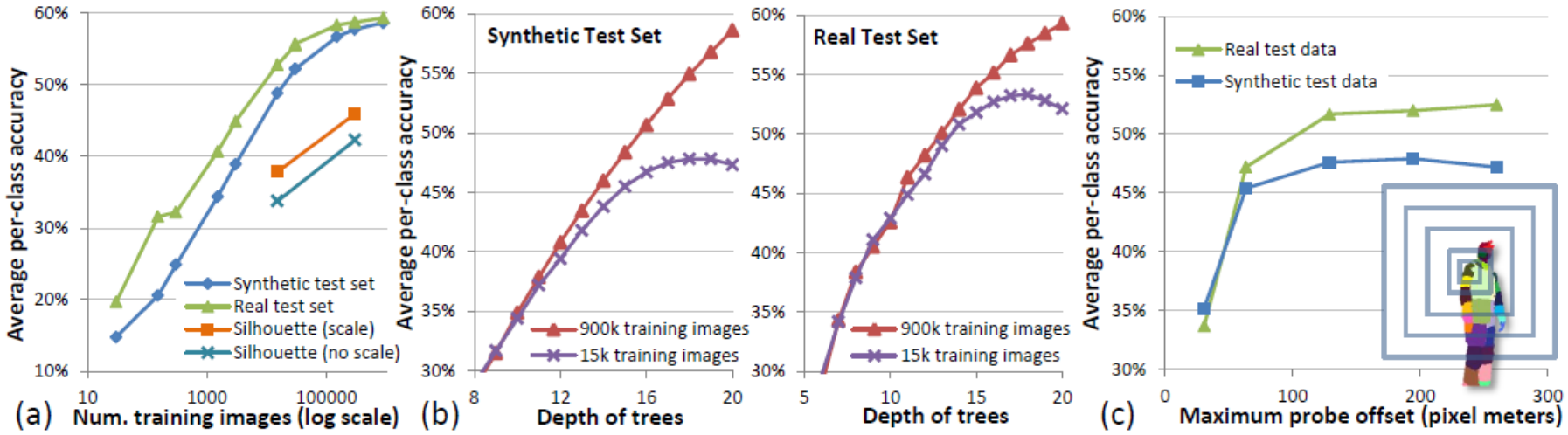


Figure 6. **Training parameters *vs.* classification accuracy.** (a) Number of training images. (b) Depth of trees. (c) Maximum probe offset.

Outline

- The reality of the problem
- A classification problem
- First sucess-story:
  - Algorithm: Boosting
  - Application: Face detection
- Second success-story:
  - Algorithm: Support Vector Machines
  - Application: Pedestrian detection
- Third success-story:
  - Algorithm: Random Forests
  - Application: Microsoft Kinect
- Some other success-stories

Some other success-stories

- **Boosting + SVM: Object models**

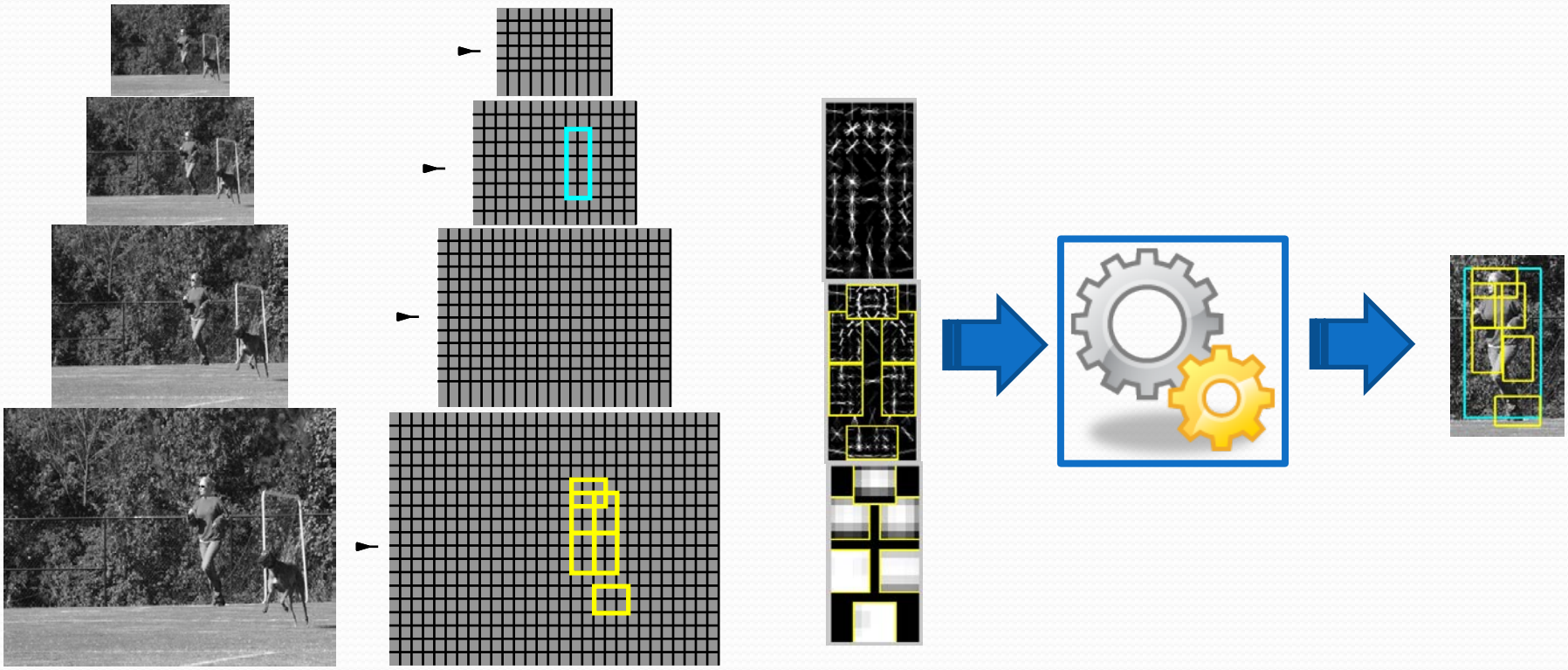
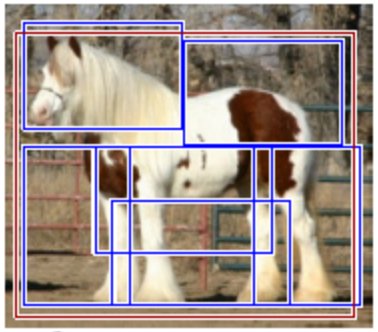
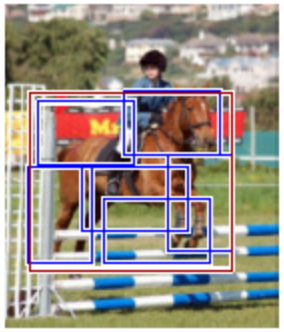
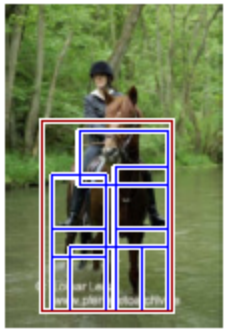
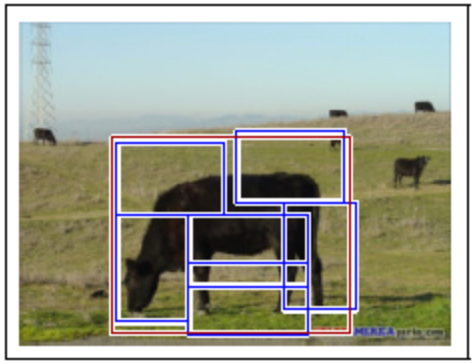
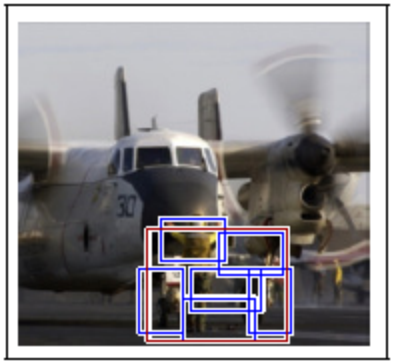


Image pyramid

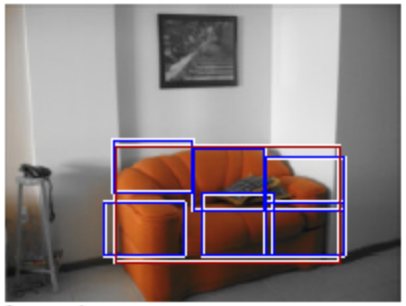
HOG feature pyramid

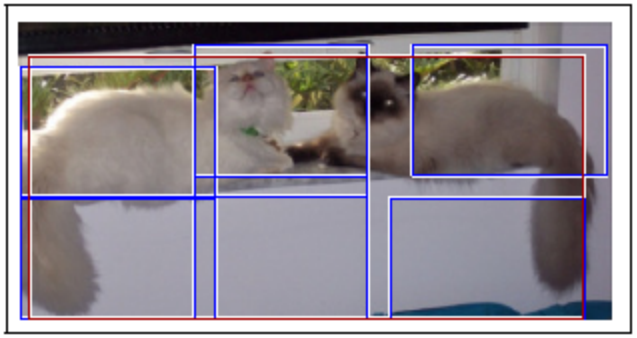
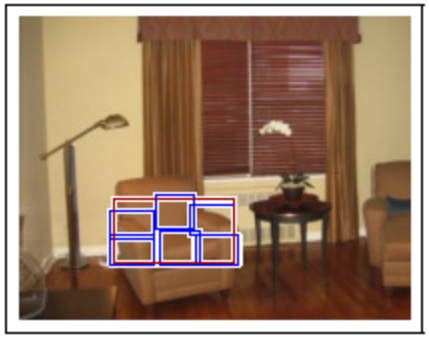
Some other success-stories

horse

sofa



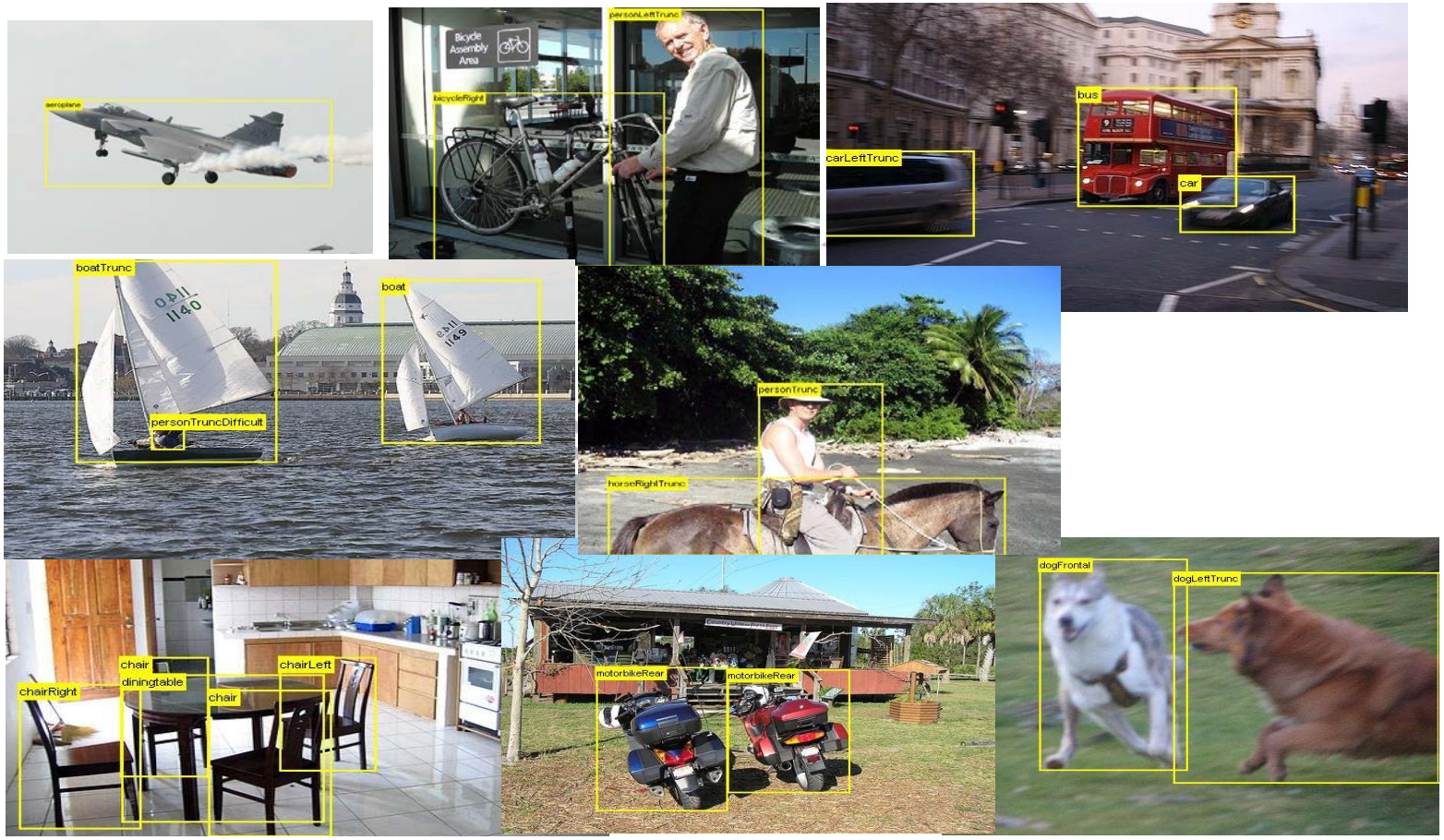



bottle



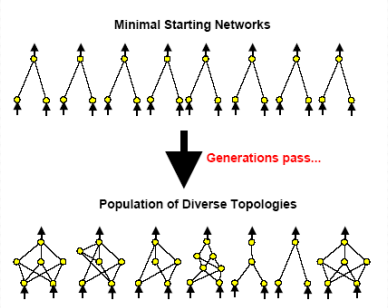

Some other success-stories

- **Boosting + SVM: Object models**



Some other success-stories

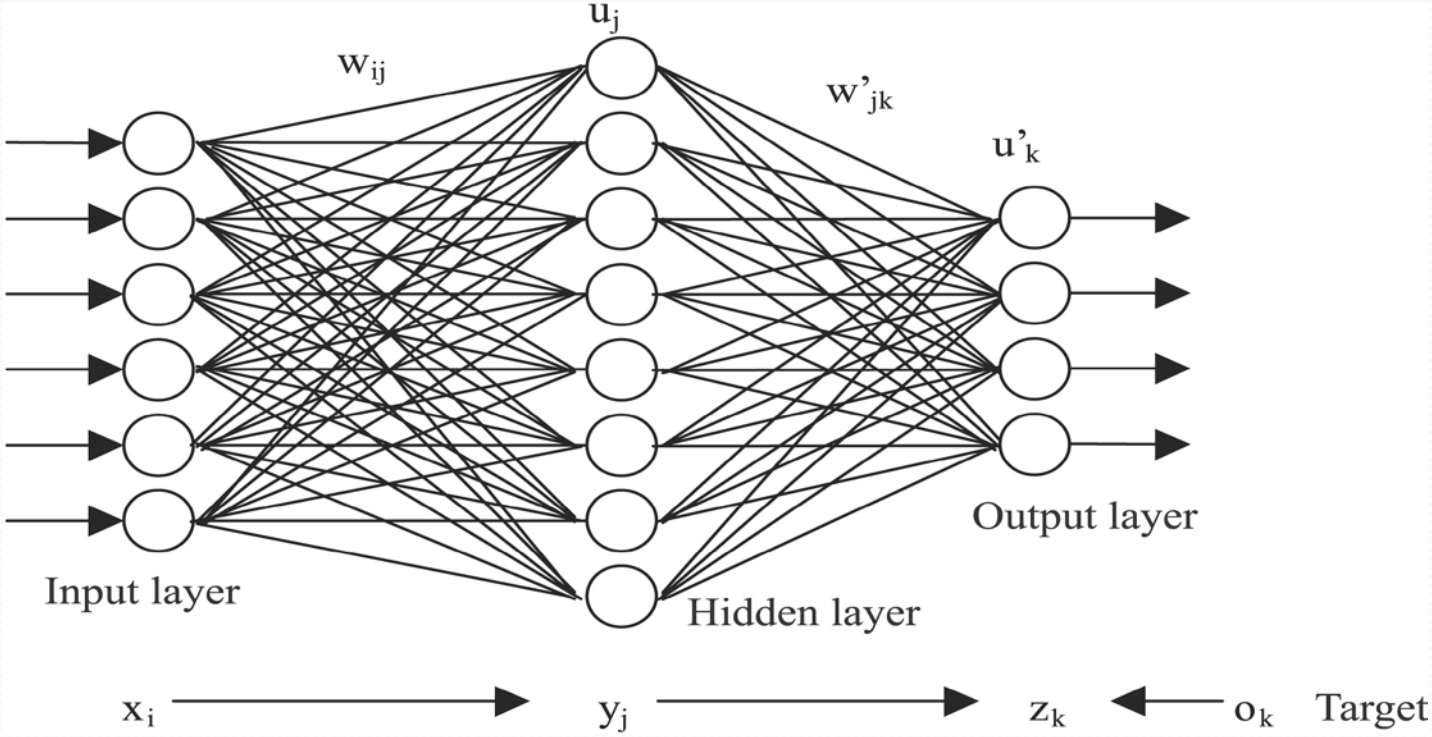
- **Neuro-Evolution: IA Games**



- **Deep neural architectures…**

Some other success-stories

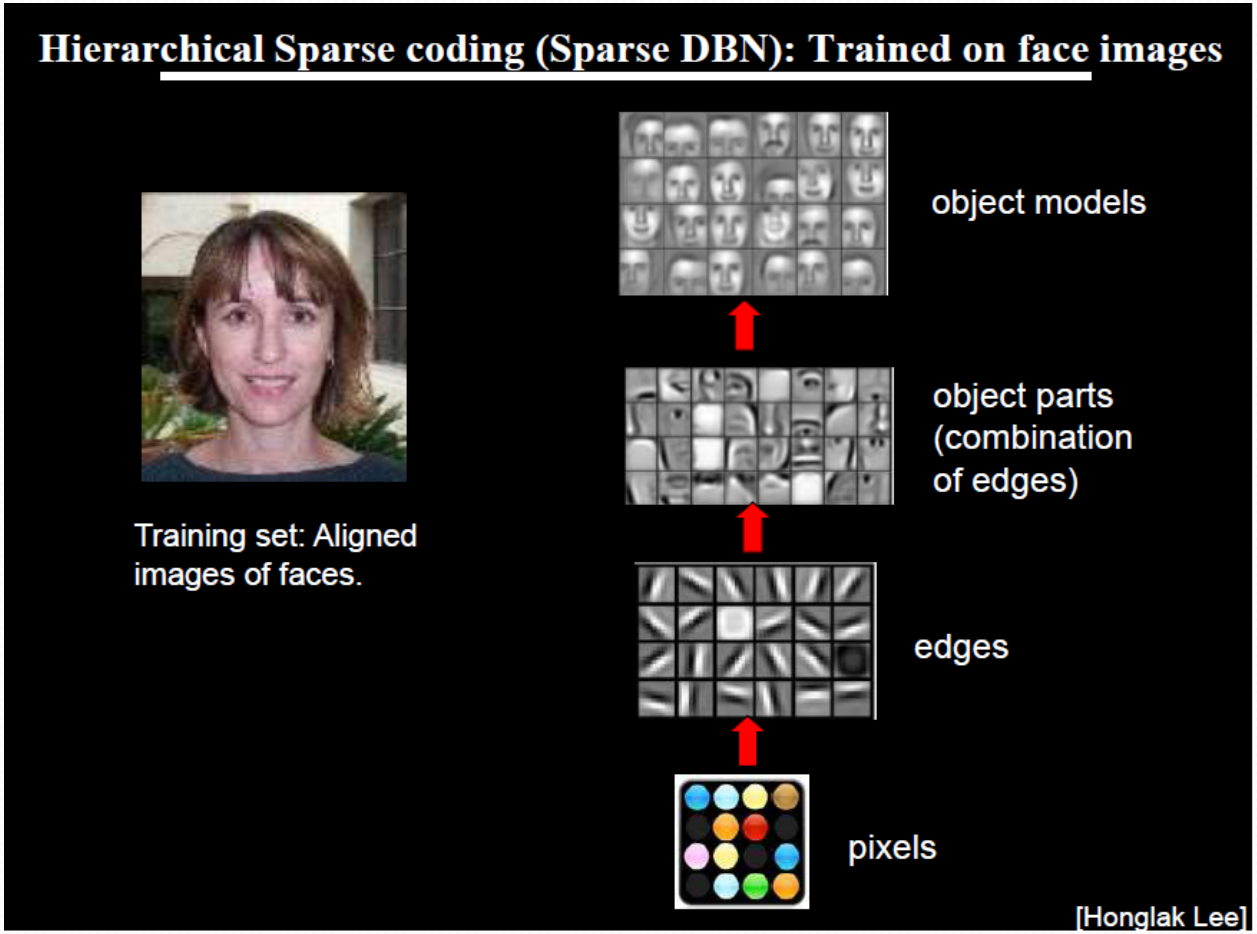
- **Deep neural architectures…first backpropagation**



Errors

Some other success-stories

- **Deep neural architectures…then deep**



Last decade success stories of Machine Learning

Avez-vous des questions ?

Frédéric Precioso
Laboratoire I3S – Equipe MinD