Learning in Robotics

Workshop Bidart June 11, 2013

Thomas Hellström

Umeå University

Sweden

SEVENTH FRAMEWORK PROGRAMME

iNTRO

Overview

- Learning in robotics
  - What and why
  - Case study: Image based visual servoing
- Learning in sensing
  - Bayesian decision theory
  - What is a classifier
  - Some classifiers
  - Combining classifiers
  - What makes a classification task hard?

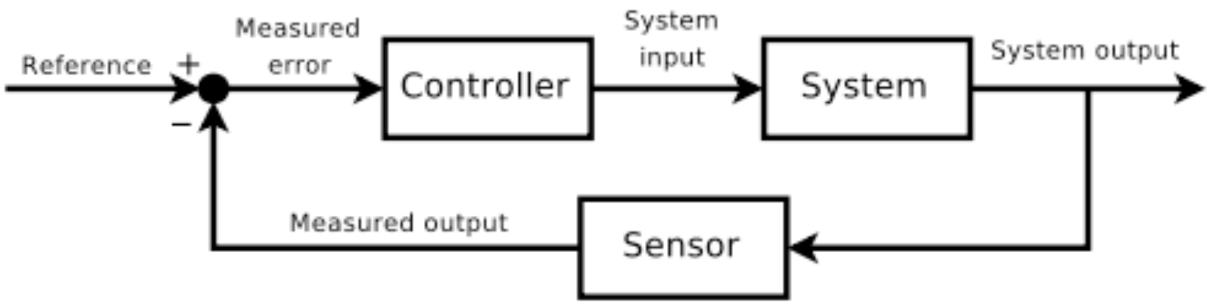
(Bayesian decision theory through classification task hardness: Pattern classification)

What is Learning?

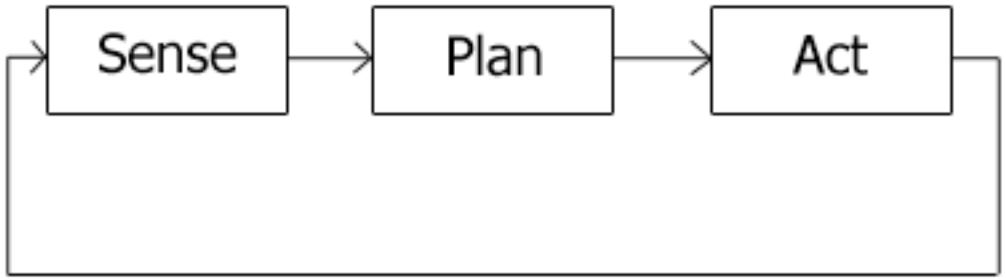
- Webster's Dictionary: "…modification of a behavioral tendency by experience"
- For a robot - and also human:
  - Behavior = Actions
  - Experience = Sensed data

Robot learning

- In all closed loop control, sensing influences actions - but that is not viewed as learning!



- Learning is about modifying the "behavioral tendency" – the mapping from sensing to action



- We have learning opportunities in all three parts

iNTRO

Robot learning in Sensing

- How to interpret sensor data
  - Object localization
  - Object classification
  - Scene understanding
  - Maps of the world
  - …

} Focus for this presentation (eventually…)

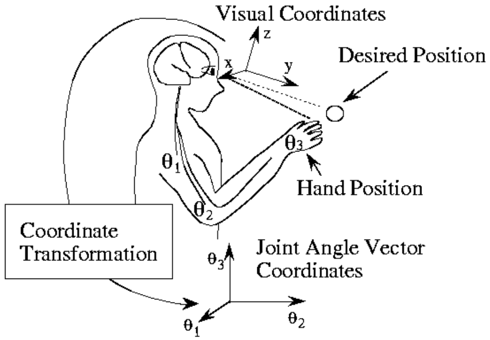
Robot learning in Planning

- How to act to reach a goal
  - Learning from demonstration
    - Sense the actions of a human teacher, and learn to act the same way
  - Reinforcement learning
    - Act and receive feedback, and change the way you act

iNTRO

Robot learning in Acting

- Relations between cause and effect
  - How the robot works
    - Kinematics, Dynamics
  - How the world works
    - How humans act and react to robot actions
    - How objects react to robot actions



Why is learning important in robotics?

- Robots are too expensive to only know pre-programmed skills
  - Entirely new skills have to be learned
- Pre-programmed skills are not sufficient in an open, non-deterministic world
  - Task specifications and environmental conditions vary and have to be learned on-line
- Sometimes we do not know how to pre-program the skills!
  - Learning applied off-line
  - Perhaps the most common case of learning for robotics

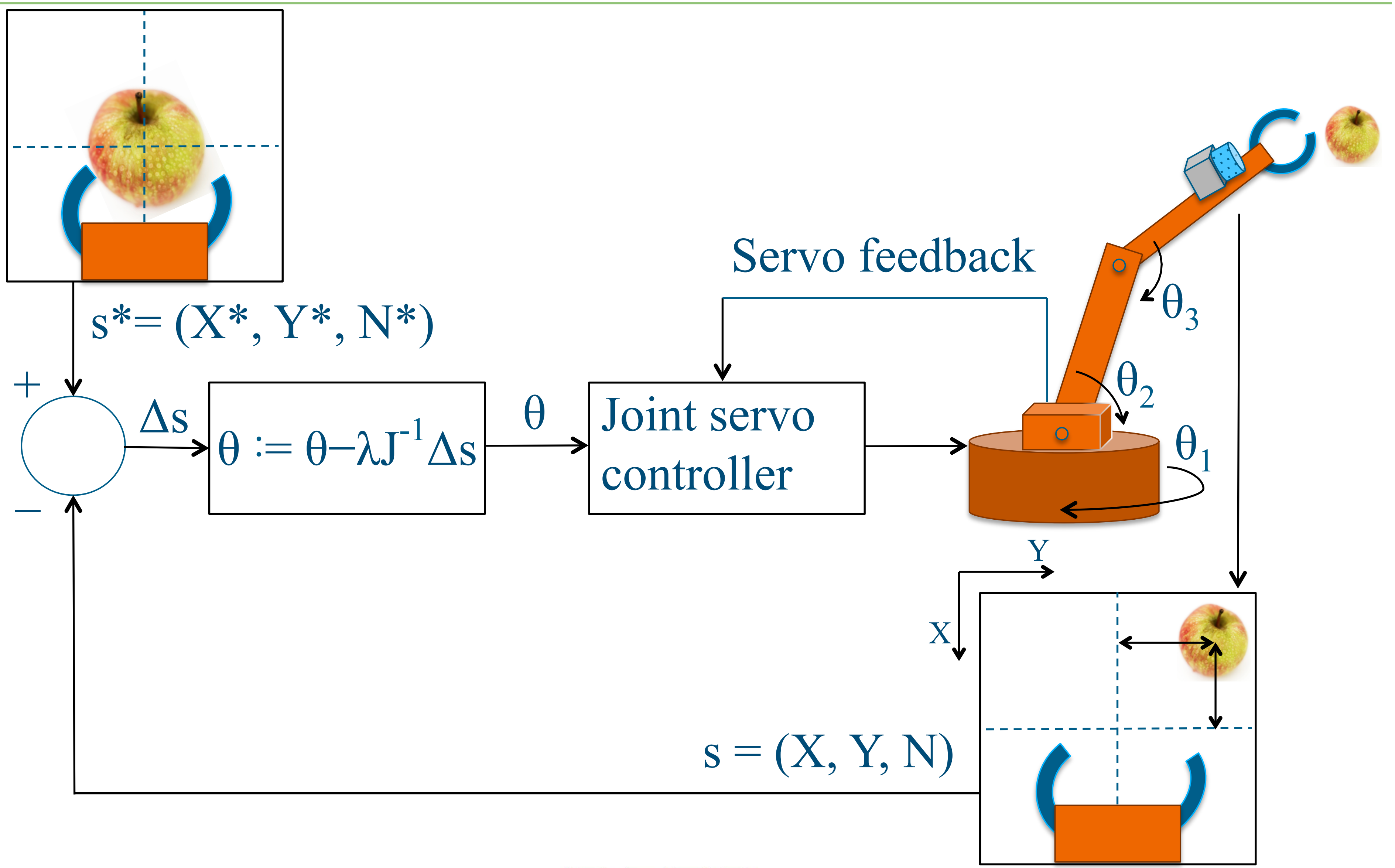
Intertwined learning

- Learning can appear in all parts: Sense, Plan, Act
- However, actions cause changes in sensing, so learning sometimes is intertwined
- One good example is Visual Servoing

Image Based Visual Servoing (IBVS) [13]

- An established technique for vision guided control of robot manipulators and grippers
- Features *s* are extracted from the 2D image (in image coord.system)
  - E.g.: location (X,Y) and area N of centroids for selected image regions
- Target feature values *s** are defined
  - E.g.: (X,Y) at image center, and area N larger than a threshold
- (s-s*) is used as error signal for a controller that drives the joints

Example: Visual servoing for fruit picking



Model-free Image Based Visual Servoing (IBVS)

- An established technique for vision guided control of robot manipulators and grippers
- Features s are extracted from the 2D image (in image coord.system)
  - E.g.: location (X,Y) and area N of centroids of selected image regions
- Target feature values s* are defined
  - E.g.: (X,Y) at image center, and area N larger than a threshold
- (s-s*) is input as error signal to a controller that drives the joints
- The controller maps this error to changes in joint angles θ

$$s = F(\theta)$$

$$\frac{\partial s}{\partial t} = \frac{\partial F}{\partial \theta}\frac{\partial \theta}{\partial t}$$

$$\dot{s} = J(\theta)\dot{\theta}$$

$$\Delta s \approx J(\theta)\Delta\theta$$

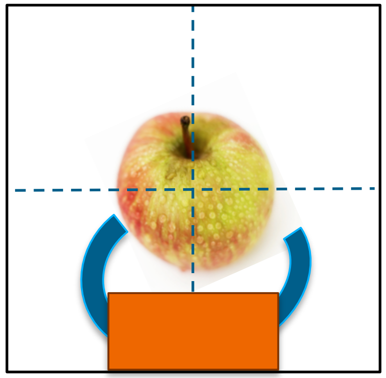
$$\theta := \theta - \lambda \hat{J}^{-1}(s)\Delta s$$

F combines the mapping from *s* to 3D coordinates, and the mapping of 3D coordinates to θ

$J(\theta) = \frac{\partial F}{\partial \theta}$ is the *visual-motor Jacobian*

Learning in model-free IBVS

- Fruit detection (which image area is of interest)
  - For real: refer to Efi, Ehud, Ohad, Yael
  - For the example:
    - Place “fruit” in the gripper
    - Learn the blob colors



$s^* = (X^*, Y^*, N^*)$

- Target feature values
    - Place “fruit” in the gripper
    - Learn X*, Y*, and N* (size) for fruit in gripper
- The visual-motor Jacobian *J*
  - Learned by motor babbling (Broyden's root-finding algorithm)
  - A case of Sensory-motor learning
    - Learning the relation between sensing and acting

$$J = \begin{pmatrix} \frac{\partial X}{\partial \theta_1} & \frac{\partial X}{\partial \theta_2} & \frac{\partial X}{\partial \theta_3} \\ \frac{\partial Y}{\partial \theta_1} & \frac{\partial Y}{\partial \theta_2} & \frac{\partial Y}{\partial \theta_3} \\ \frac{\partial N}{\partial \theta_1} & \frac{\partial N}{\partial \theta_2} & \frac{\partial N}{\partial \theta_3} \end{pmatrix}$$

Demo CRO$^{p}_{b}$S light

Gene modified
Christmas tree

- Features s = (X,Y,N) for detected fruit
- Target features s*=(X*, Y*, N*)
- Δs=(s-s*)
- $J(\theta) = \dfrac{\partial F}{\partial \theta}$

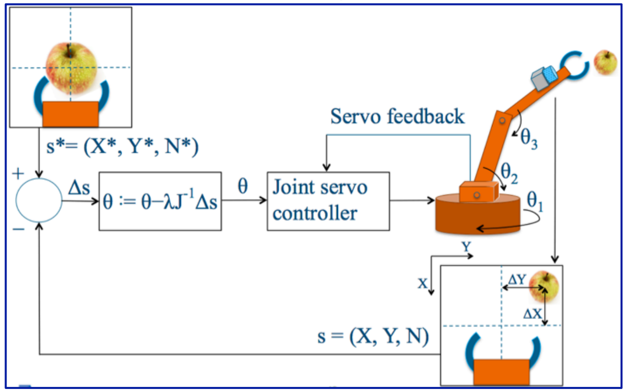
$$s = F(\theta)$$

$$\frac{\partial s}{\partial t} = \frac{\partial F}{\partial \theta}\frac{\partial \theta}{\partial t}$$

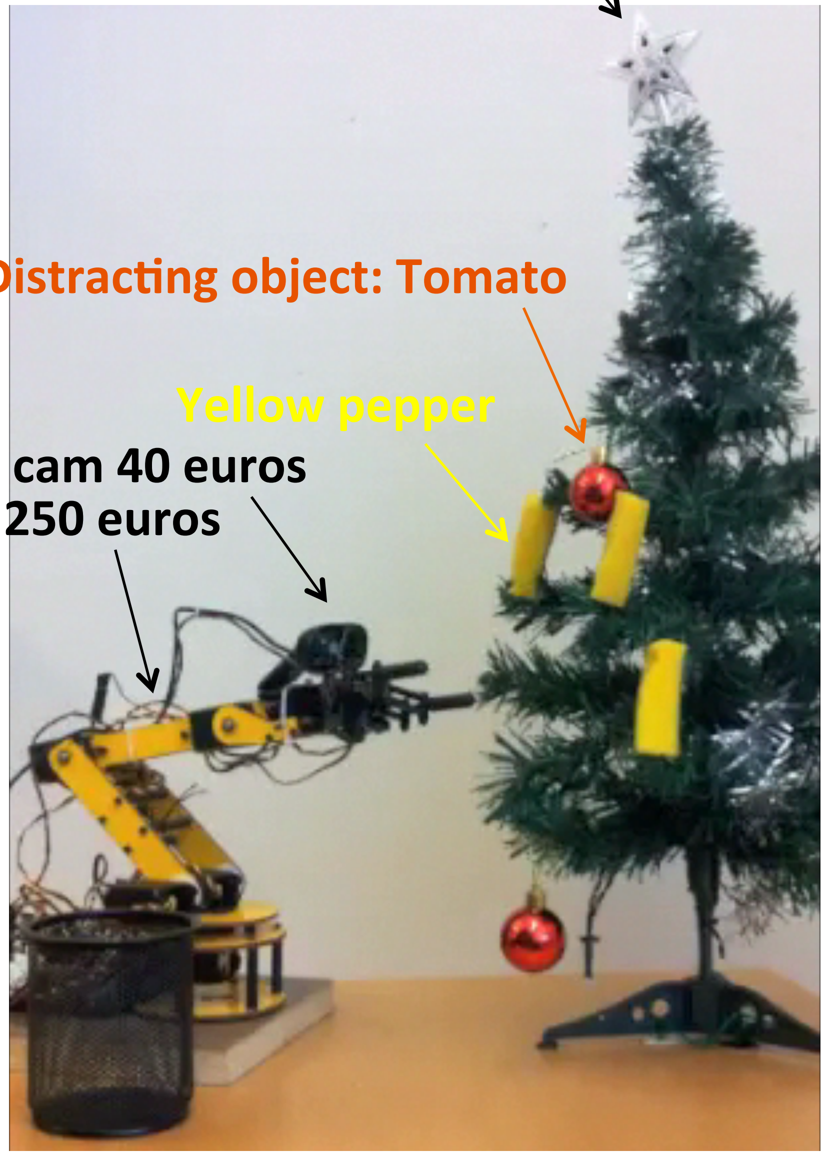
$$\dot{s} = J(\theta)\dot{\theta}$$

$$\Delta s \approx J(\theta)\Delta\theta$$

$$\theta := \theta - \lambda \hat{J}^{-1}(s)\Delta s$$



Logitech web cam 40 euros
Lynx robot arm 250 euros



Distracting object: Tomato

Yellow pepper

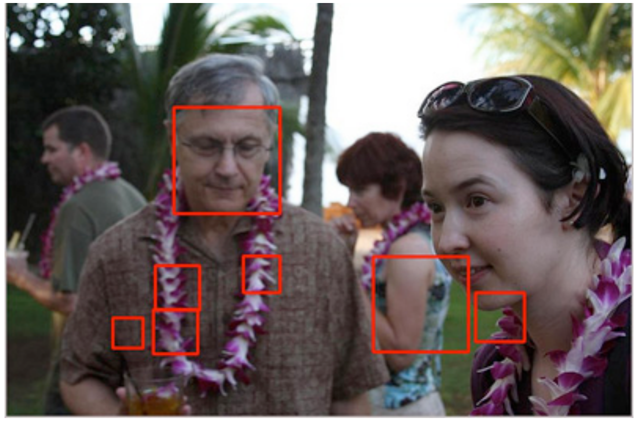
Robot learning in Sensing

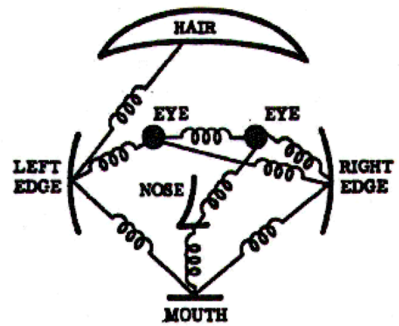
- How to interpret data
  - Object localization
  - Object classification
  - Scene understanding
  - Maps of the world
  - …

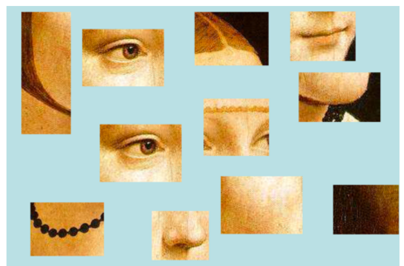
} Focus for the rest of the presentation

Learning for object localization/classification

- Usually based on image data
- Localization by sliding windows
  - Multiple locations and scales
- Feature extraction
  - Edges, Corners, Lines, Circles
  - Histogram of gradients (HOG)
  - Scale-invariant feature transform (SIFT)
- State-of-the-art
  - Part-based methods [1,2]
  - Bag-of-words methods [3,4,5,6]
- In most methods, *Pattern classification* techniques are used to map features to class







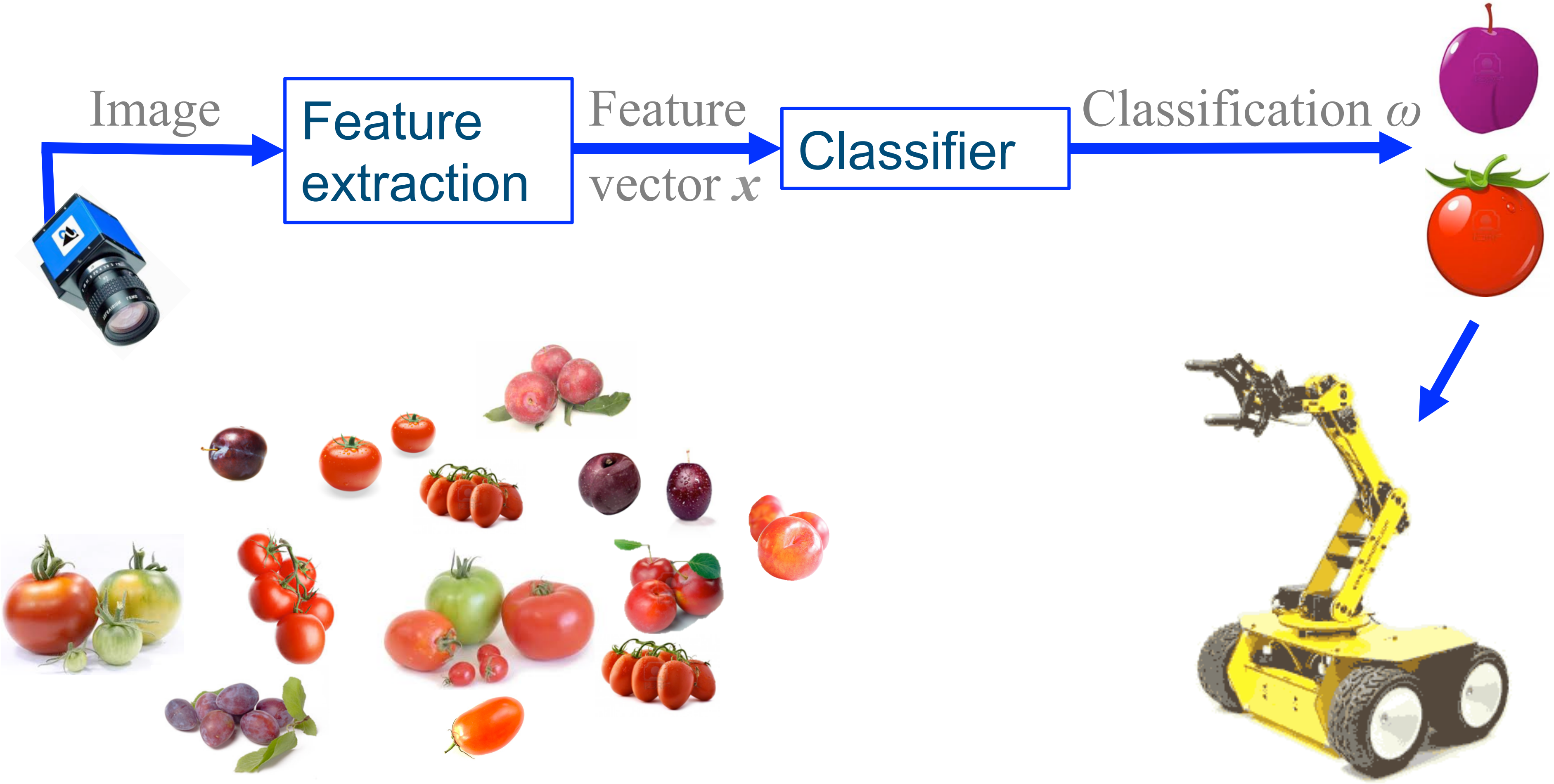
Pattern classification

- One (of a few) central problem statement:

  Given a set of N instances consisting of features $\boldsymbol{x} = (x_1, \ldots, x_d)$ *and a class label* ω *in* $\{\omega_1, \ldots, \omega_c\}$, *construct a classifier that successfully predicts* ω *for new feature vectors*
- Similar to regression, but ω is discrete
- Learning!

Example

Classification of fruit in images



Bayesian decision theory [7]

- A fundamental statistical approach to pattern classification
- ω and $\boldsymbol{x}$ are viewed as stochastic variables (s.v.)
- The unknown class ω is a s.v. with *prior probabilities*

$$P(\omega_1),\ldots, P(\omega_c)$$

- Each feature vector $\boldsymbol{x} = (x_1, \ldots, x_d)$ is a s.v. with *probability density function* $p(\boldsymbol{x}) = (p(x_1), \ldots p(x_d))$
- More useful to look at each class separately: *Class conditional probability density functions* defined as

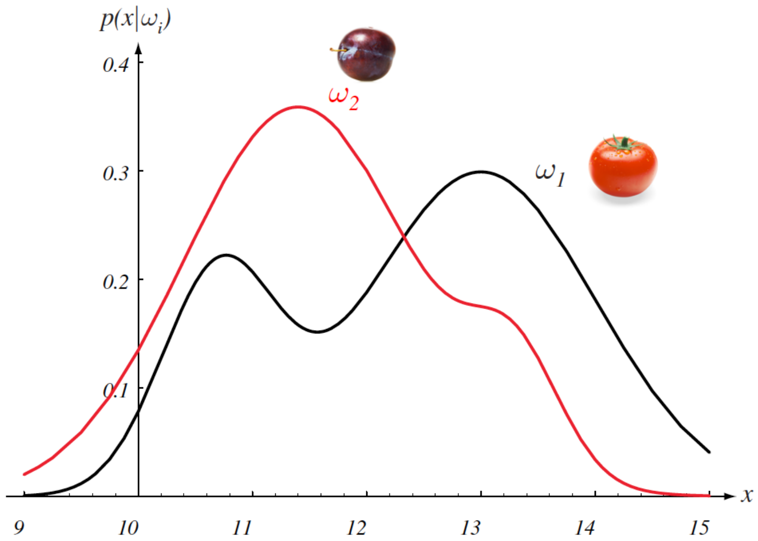
$$p(x_i \mid \omega_j) = p(x_i , \omega_j) / \mathrm{P}(\omega_j)$$

Example:

- The robot
  - should decide if a detected fruit is a plum or tomato
  - measures size (in reality a lot more features: colour, shape, …)
- Two classes: ω_1 (tomatoes) and ω_2 (plums) with priors

$$P(tomato) = 2/3 \quad P(\text{plum}) = 1/3$$

- One feature: $x = size$
- *Class conditional probability density functions* ➔
- Possible interpretation:
  - There are two types of tomatoes; small and big. The size of plums is in between these two types



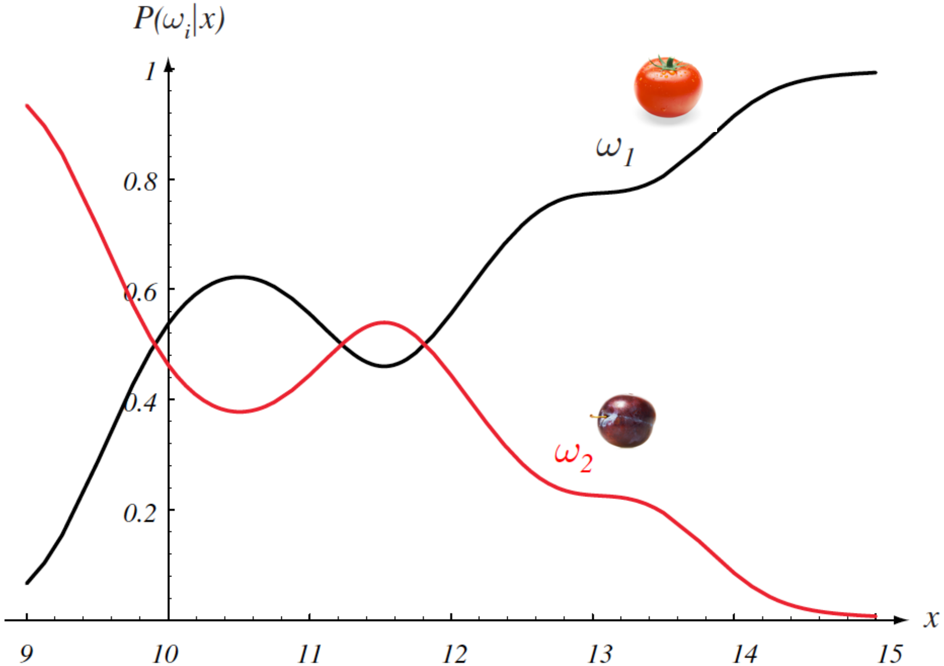
Bayesian decision theory

- We want to predict ω given $\boldsymbol{x}$
- How about this decision rule:
  "Choose the class that is most probable given observation x" ?
- This is expressed by the *posterior probability* $P(\omega_i \mid \boldsymbol{x})$, which is defined by

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x},\omega_i)}{p(\boldsymbol{x})}$$

and also related to the *class conditional probabilities* by Bayes rule:

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}\mid\omega_i)P(\omega_i)}{p(\boldsymbol{x})}$$



Bayesian decision theory

- The decision rule can be written

$$\omega_{MAP} = \arg\max_i P(\omega_i|\boldsymbol{x})$$

  and is called *Maximum a posteriori probabilities* (MAP) or Bayes decision rule

- It can be shown that it minimizes the risk of miss classification: $P(Error \mid x)$
- The idea can easily be extended to take different miss classification costs into account

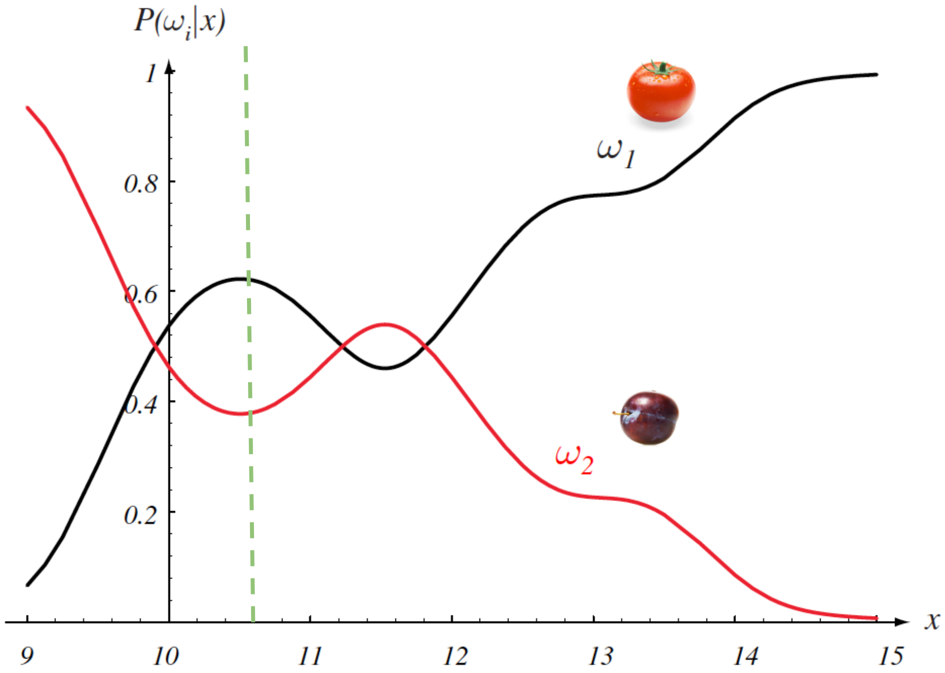
Back to the example

- Given an observed size *x*, we predict the fruit ω_i with maximum $P(\omega_i|x)$
- For observed *size* =10.6:

  $P(tomato|14)=0.62$

  $P(plum|14)=0.38$

  The fruit is a tomato!!



- Simple principle, but we need to know all $P(\omega_i|x)$
- Often they can be estimated from the data (LEARNING!)
- Some classifiers, like KNN and ANN, do exactly that!

Other formulations of MAP

- Reformulation with Bayes rule:

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}\,|\,\omega_i)P(\omega_i)}{p(\boldsymbol{x})}$$

- $p(x)$ is independent of i and does not affect the MAP decision

$$\omega_{MAP} = \arg\max_i p(\boldsymbol{x}|\omega_i)\, P(\omega_i)$$

- Same decision as for $P(\omega_i \,|\, \boldsymbol{x})$, but different methods for the estimation

Classifiers as discriminant functions

- The posterior probabilities can be seen as functions of $\boldsymbol{x}$

$$g_i(\boldsymbol{x}) = P(\omega_i \mid \boldsymbol{x}) \text{ or}$$

$$g_i(\boldsymbol{x}) = p(\boldsymbol{x} \mid \omega_i)\, P(\omega_i)$$

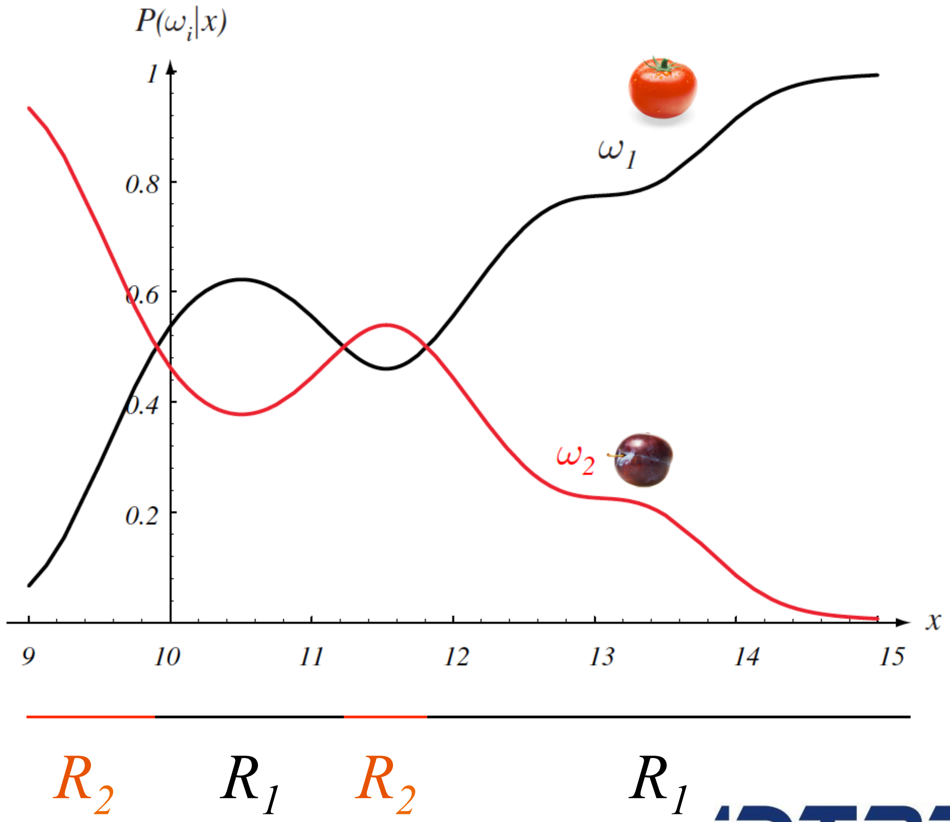
- A classifier can be described as a set of *discriminant functions* $g_i(\boldsymbol{x})$, $i = 1, \ldots, c$.
- The classifier assigns a feature vector $\boldsymbol{x}$ to ω_i iff

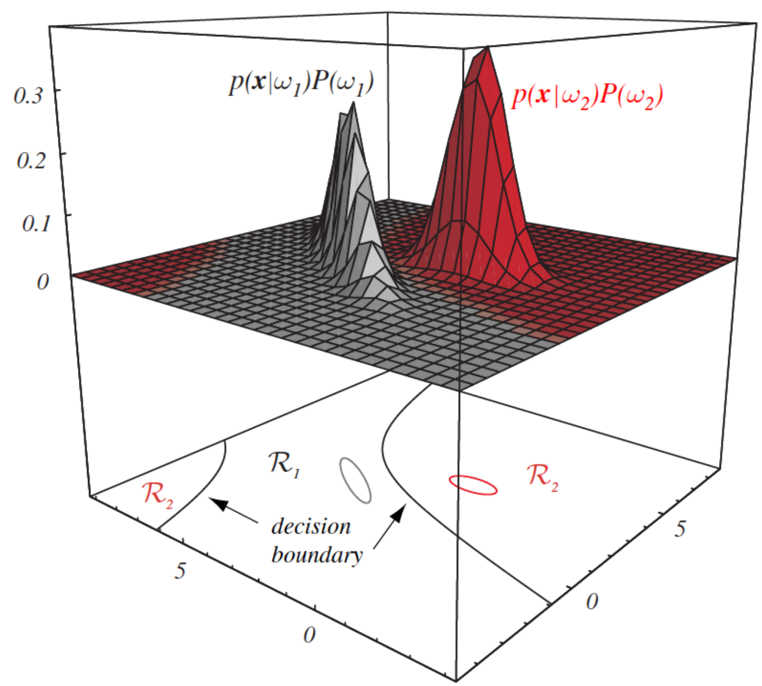
$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \text{ for all } j \neq i$$

- "Lines" along which $g_i(\boldsymbol{x}) = g_j(\boldsymbol{x})$ are the decision boundaries
- Learning a classifier ↔ finding discriminant functions or decision boundaries that optimize some chosen objective

Decision boundaries and decision regions

- Decision boundaries separate input space in *decision regions*
- *Decision regions* R_i, $i=1,\ldots,c$, are the set of points in feature space where we decide ω_i
- The decision regions need not be simply connected





Two ways of constructing classifiers

- Estimating discriminant functions

$$g_{\mathrm{i}}(\boldsymbol{x}) = P(\omega_i \mid \boldsymbol{x}) \text{ or}$$

$$g_i(\boldsymbol{x}) = p(\boldsymbol{x} \mid \omega_i)\, P(\omega_i)$$

- Estimating optimal decision boundaries

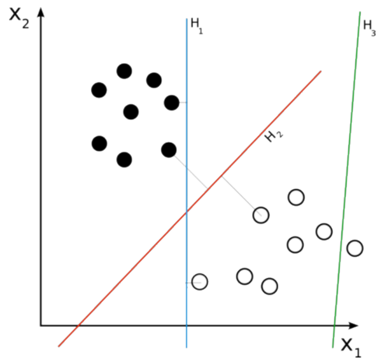
Constructing classifiers by estimating discriminant functions

- Parametric methods
  - Assume a parameterized form of $p(\boldsymbol{x} \mid \omega_i)$ or $P(\omega_i \mid \boldsymbol{x})$ and estimate the parameters from data
- Non-parametric methods
  - Estimate from data
    - Naïve Bayes : $g_i(\boldsymbol{x}) = p(\boldsymbol{x}|\omega_i)\ P(\omega_i)$
    - k-NN : $g_i(\boldsymbol{x}) = P(\omega_i \mid \boldsymbol{x})$

Constructing classifiers

by estimating optimal decision boundaries

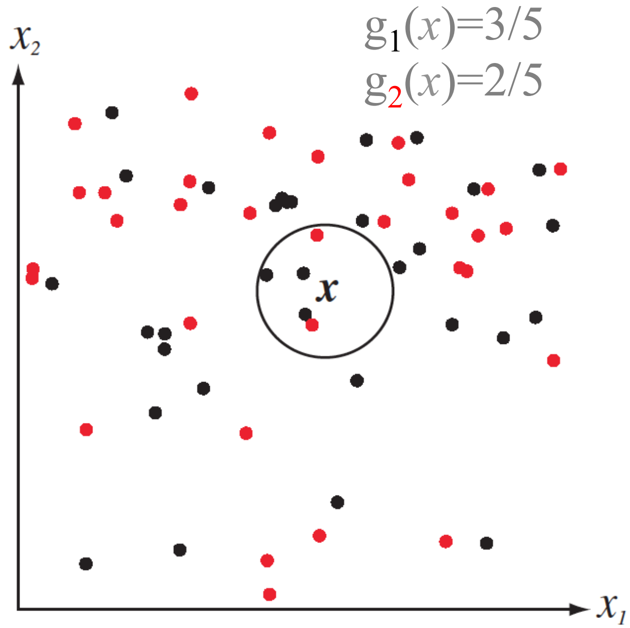
- Parametric methods
  1. Assume a parameterized form of decision boundary
  2. Estimate parameters
  - Perceptrons
  - LDA
  - Support vector machines (SVM)



- Non-parametric methods
  - Neural networks

k-nearest-neighbor classifier (k-NN)

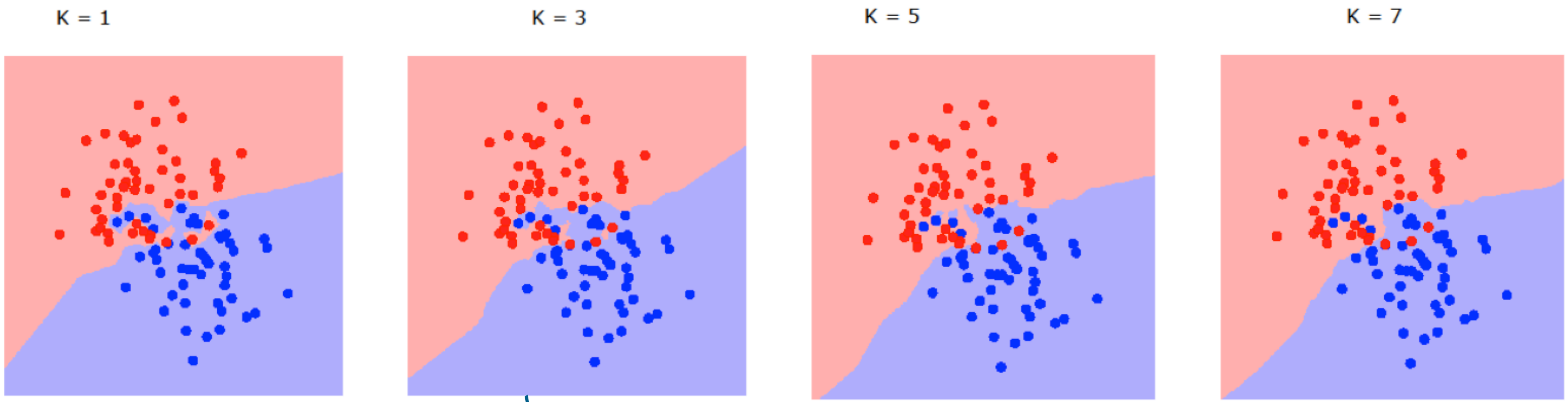
- Estimates discriminant function $g_i(\boldsymbol{x}) = P(\omega_i \mid \boldsymbol{x})$
  - find the k nearest neighbors to $\boldsymbol{x}$
  - $P(\omega_i \mid \boldsymbol{x})$ = #(neighbors with class=i)/k

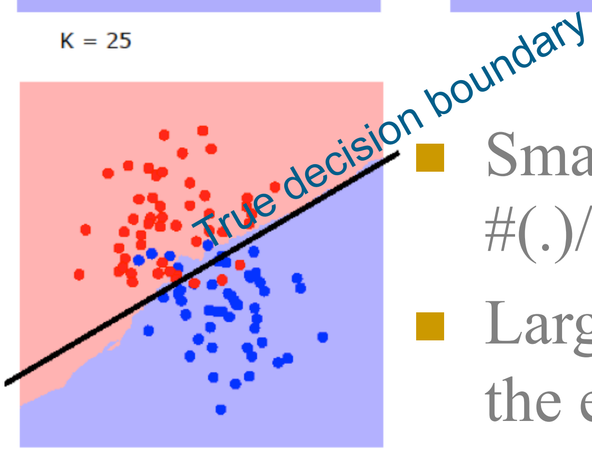


- As before: $\omega = \arg\max_i g_i(\boldsymbol{x})$
  i.e. predict the most frequent class
- Not always good probability estimates, but the decision rule only the ordering of discriminant functions $g_i(\boldsymbol{x})$

k-nearest-neighbor classifier (k-NN)

- Piecewise linear decision boundary
- Larger k gives smoother decision boundary





- Small k : all k neighbors are close to $\boldsymbol{x}$ ➔ #(.)/k is close to $P(\omega_i \mid \boldsymbol{x})$
- Large k : more data in the calculation ➔ the estimate #(.)/k is more reliable
- k can be determined by cross-validation

Pictures from R. H. Lathrop Univ. of California

Combining several classifiers [12]

3 reasons why this may be a good idea

H: All classifier hypotheses that can be learned with the method

f: the correct hypothesis

<u>Statistical argument</u>

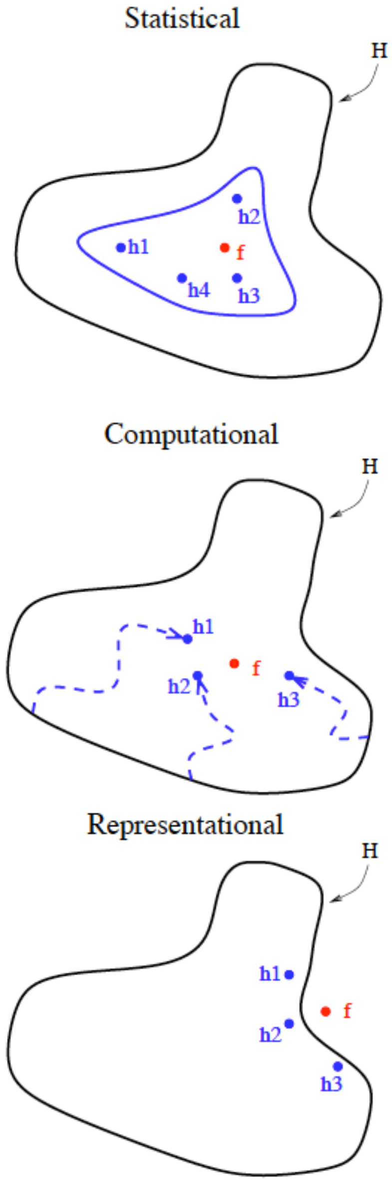
With limited data, we may find different hypotheses h1, h2, h3,… by using different subsets of the data. Averaging increases the chance of being close to f.

<u>Computational argument</u>

Learning, often a search, may get stuck in local minima. Averaging the result from several starting points increases the chance of being close to f.

<u>Representational</u>

F may be outside H. Averaging may expand H.



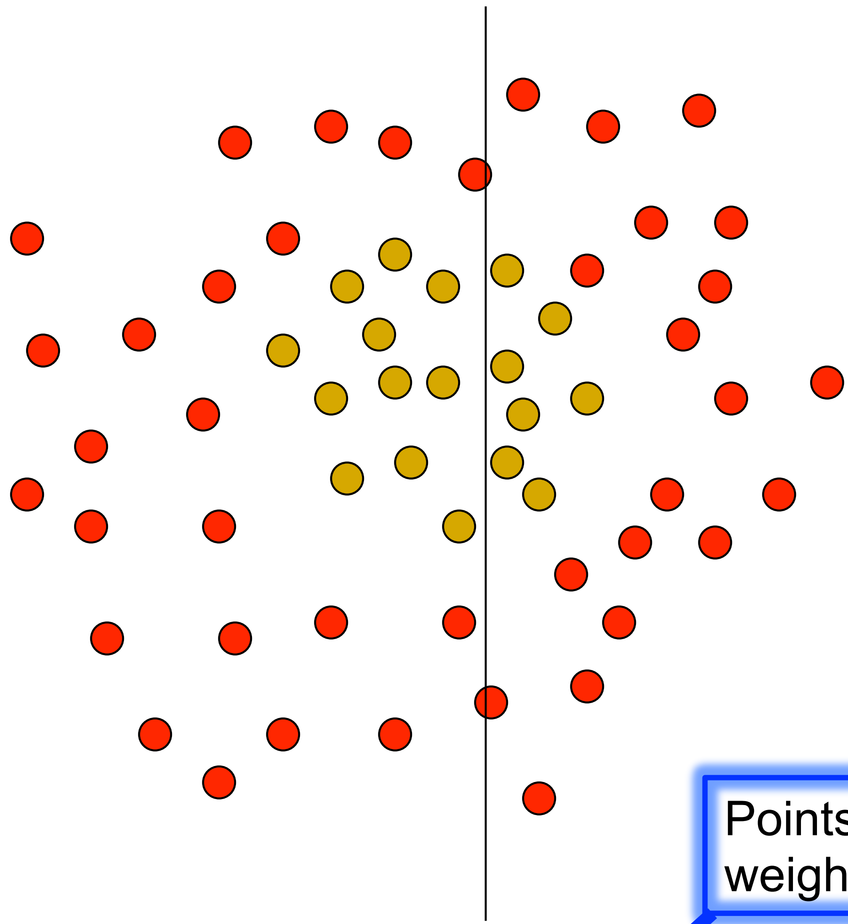
Combining several classifiers

- Several approaches
  - Voting for several classifiers (hypotheses)
  - Modifying the training data
- AdaBoost [8,9]
  - Create a sequence of simple "weak" classifiers
  - Each new classifier
    - is typically fast to train, e.g. a decision tree, "decision stump", or a linear classifier
    - must be a bit better (or worse…) than random guessing
    - focuses on the hard cases by weighting training data
  - Output: a weighted sum of all created classifiers

Example AdaBoost

- figures from Antonio Torralba@MIT

Each data point $\mathbf{x}_i$ has a class label $y_i = \begin{cases} +1 \text{ (red)} \\ -1 \text{ (yellow)} \end{cases}$



Weak classifiers $h_t(\mathbf{x})$: Lines performs (at least) slightly better than chance.

$h_t(\mathbf{x})=+1$ for points on one side of the line and -1 on the other

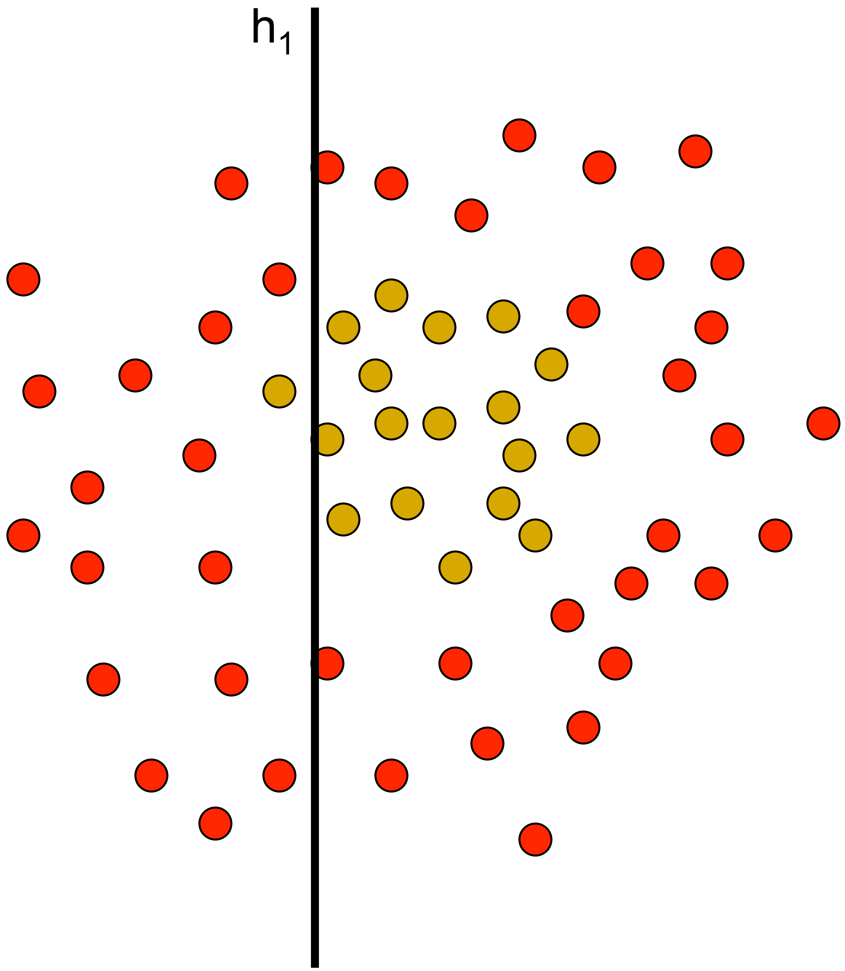
Learning means Finding parameters defining the best h_t

Each data point has an initial weight (size) $w_i=1$

Points with high weights count more

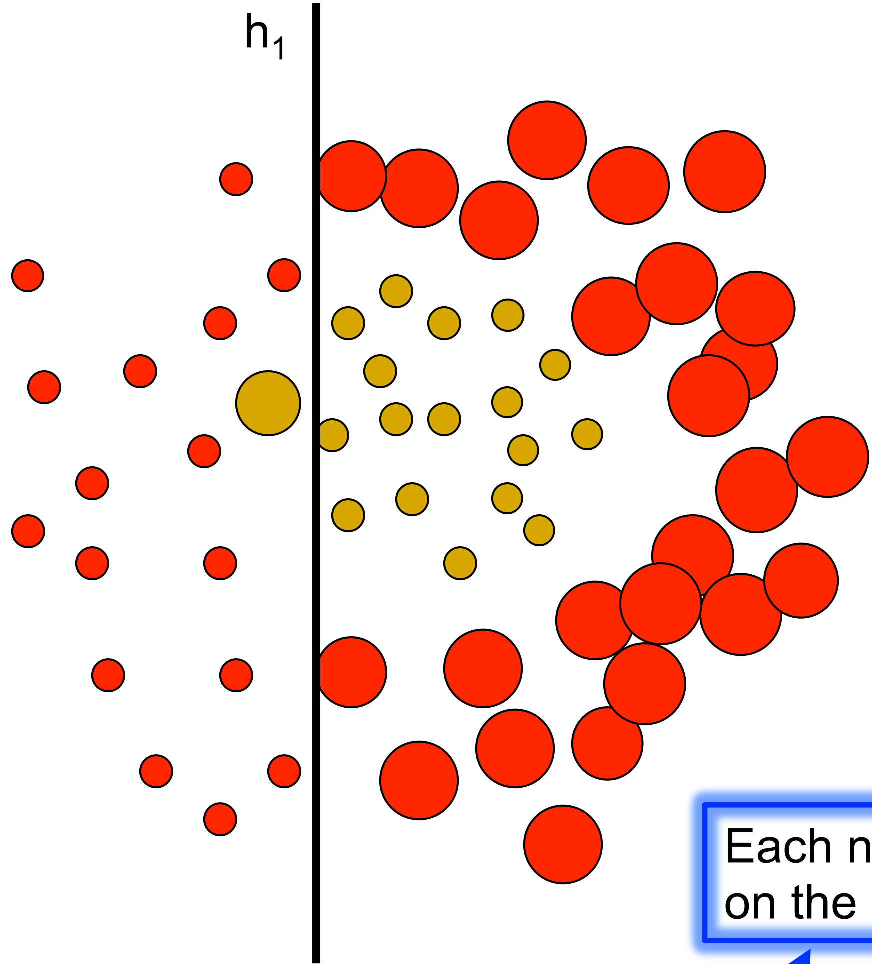
Learn a weak classifier h_1 that maximizes the weighted performance

Example AdaBoost



Update the weights: Increase w_i if $h_t(\mathbf{x}_i) \neq y_i$,decrease if $h_t(\mathbf{x}_i) = y_i$

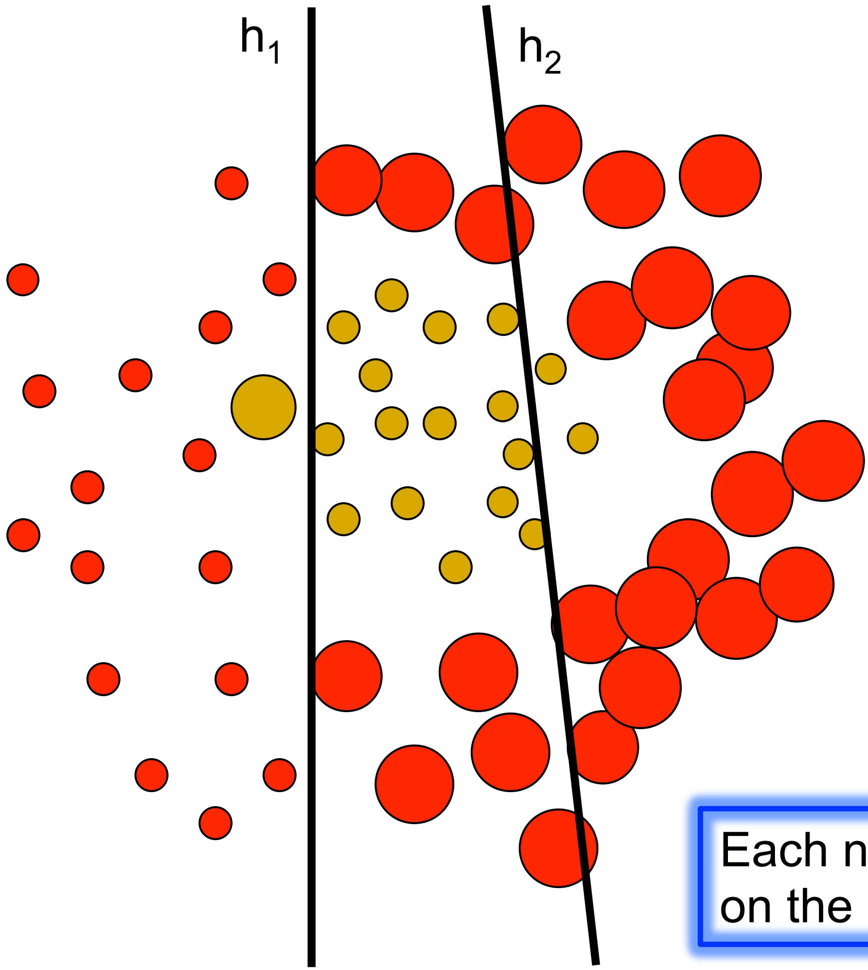
Example AdaBoost



Each new classifier will focus on the “hard” cases

Learn a **new** weak classifier h_2 that maximizes the weighted performance

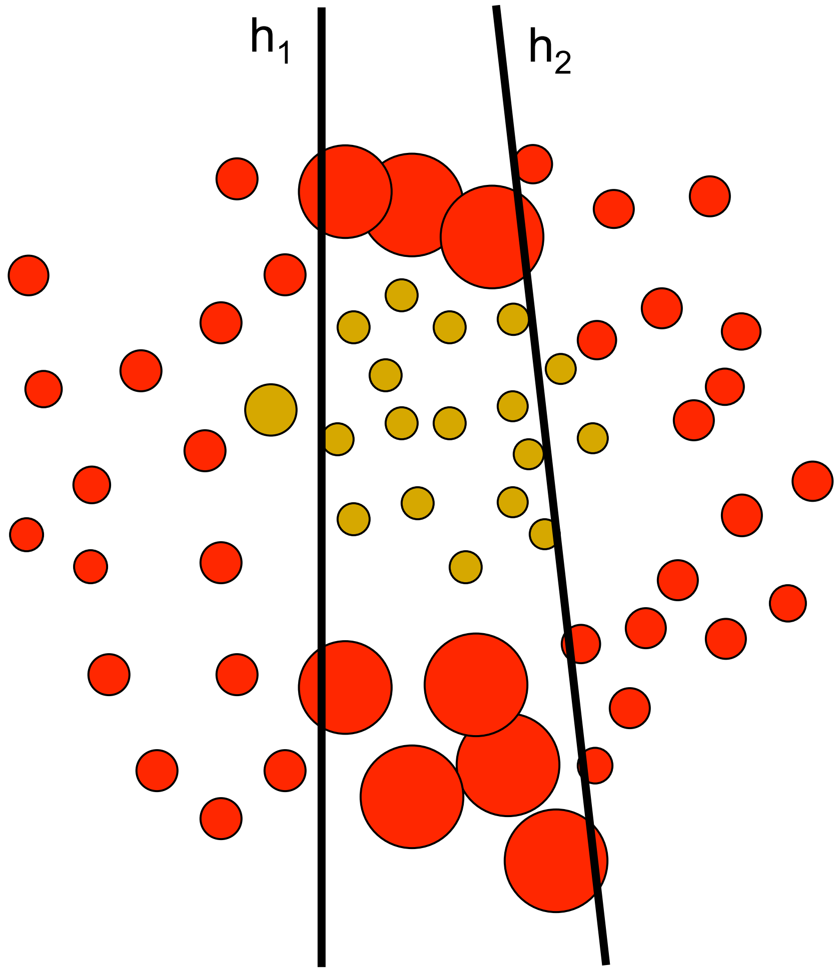
Example AdaBoost



Each new classifier will focus on the “hard” cases

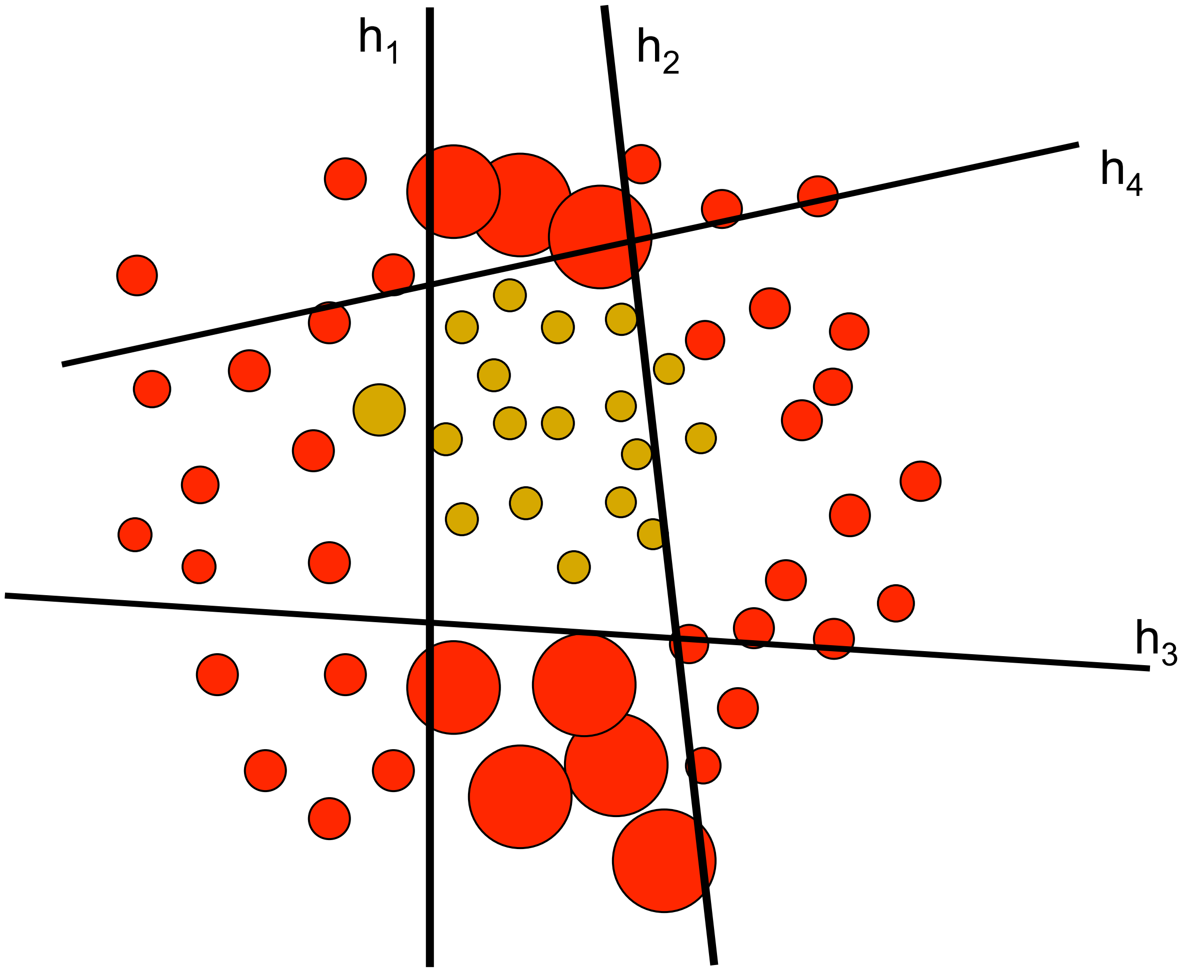
Learn a **new** weak classifier h_2 that maximizes the weighted performance

Example AdaBoost



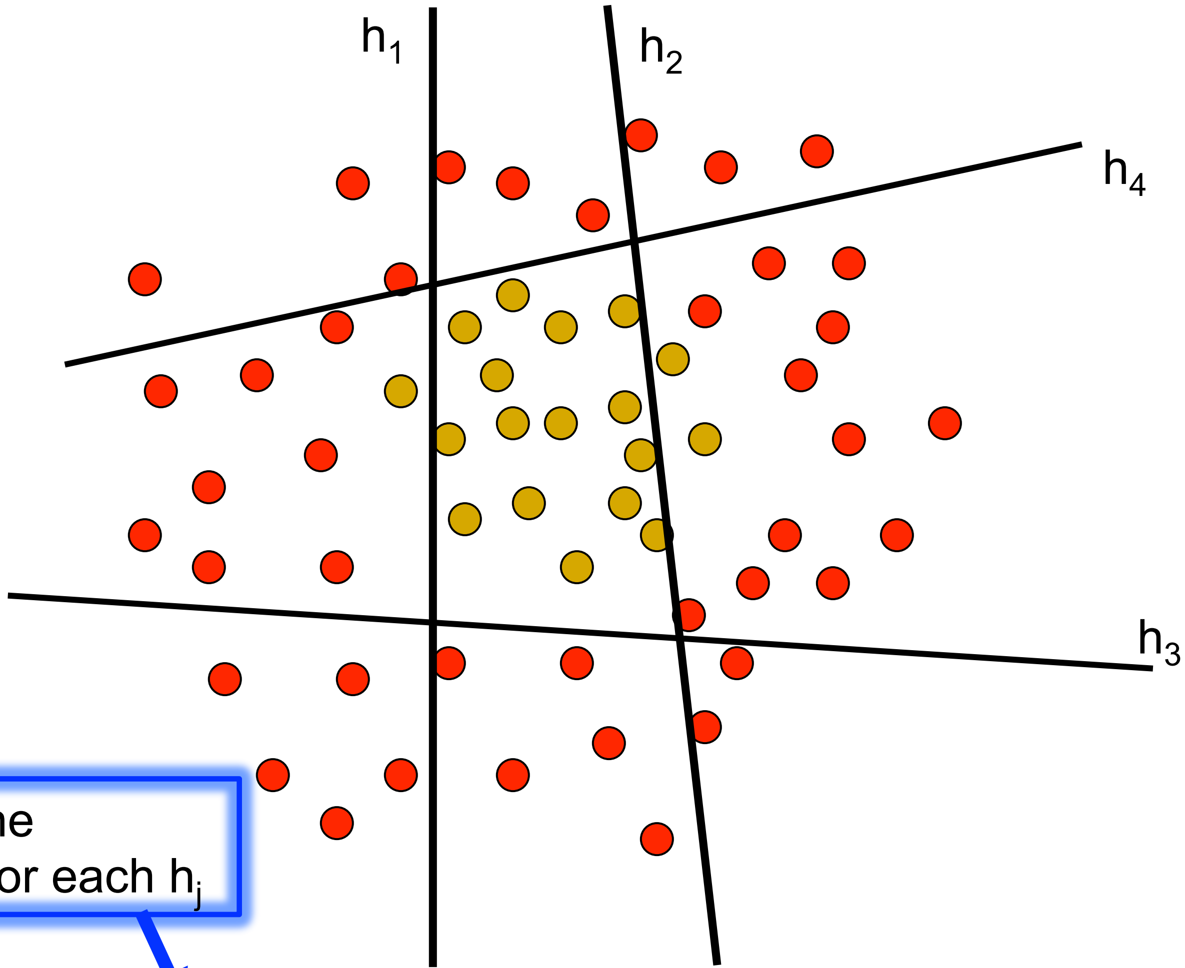
Update the weights: Increase w_i if $h_t(\mathbf{x}_i) \neq y_i$,decrease if $h_t(\mathbf{x}_i) = y_i$

Example AdaBoost



Learn **new** weak classifiers h_3 and h_4 that maximize the weighted performance

Example AdaBoost



Weights are the performance for each h_j

Final classification: A weighted sum of all weak classifiers $h_1 h_{2,} h_{3,} h_4$

AdaBoost

- Very impressive (accurate and fast) performance
  - Used in Viola-Jones face detector
- Simple
  - “just 10 lines of code” [R. Schapire] (actually correct!)
- Solid theoretical foundation
- The inventors R. Schapire and Y. Freund won the 2003 Gödel Prize for the algorithm

INTRO

Just 10 lines of code

$D = \{(x_1,y_1),\dots,(x_n,y_n)\}$, k_{max}, $w_i = 1/n$, $j = 1,\dots,n$

for $k=1$ to k_{max}

  Train weak learner C_k using D sampled according to w_i

  E_k = training error of C_k measured on D using w_i

  $a_k = 1/2 \ln[(1- E_k)/ E_k]$ (performance for classifier C_k)

  if $f_k(x_i) = y_i$ (correct classification)

    $w_i = w_i e^{-a_k}$ (increase weight)

  else

    $w_i = w_i e^{a_k}$ (decrease weight)

end

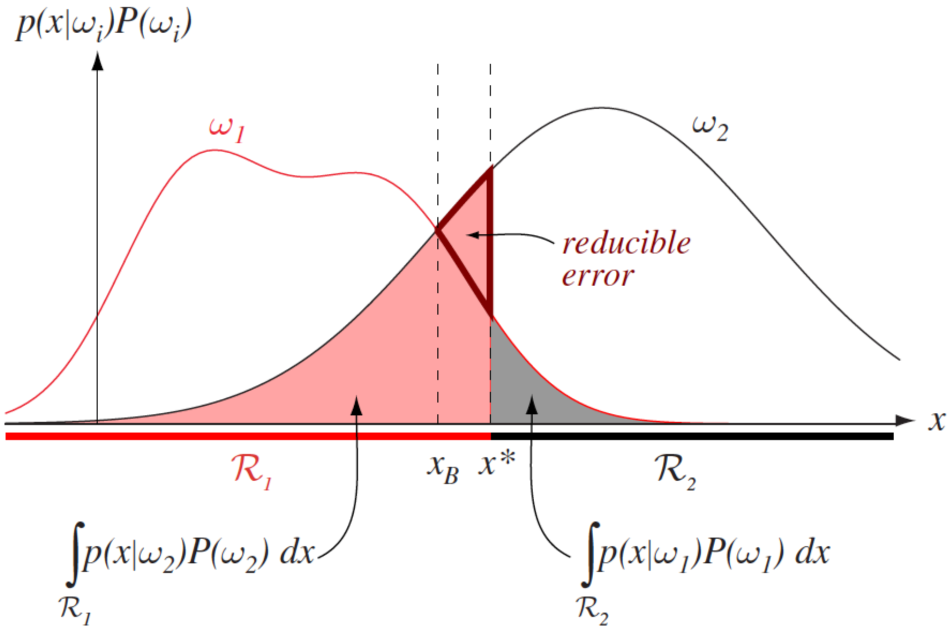
Final classifier: $\sum_{k=1}^{k_{max}} a_k C_k(x)$

What makes a classification task hard?

- Overlapping posterior probabilities $P(\omega_i \mid \boldsymbol{x})$
  - The classes are not uniquely determined by the features
  - Leads to inherent class ambiguity[11]
  - Even with training data covering all combinations of features, classification errors will occur!
- Complexity of the decision boundary
  - The optimal decision boundary needs a long description (Kolmogorov complexity)
  - Complexity typically grows with dimensionality
  - Harder to predict the boundary with little data
  - Extreme example:
    The class labels are assigned randomly
    - No generalization possible - no other way than to use training data as a look-up table
    - Is a problem even with infinite data and no class ambiguity

Class ambiguity

- The Bayes decision assumes full knowledge of all probabilities
- Assume we have a sub optimal decision boundary **x*** separating decision regions R_1 and R_2
- Two possible errors:
  - Predicting ω_2 when real class is ω_1
  - Predicting ω_1 when real class is ω_2



- $$P(error) = P(\mathbf{x} \in \mathcal{R}_2, w_1) + P(\mathbf{x} \in \mathcal{R}_1, w_2) = P(\mathbf{x} \in \mathcal{R}_2 | w_1) P(w_1) + P(\mathbf{x} \in \mathcal{R}_1 | w_2) P(w_2)$$
$$= \int_{\mathcal{R}_2} p(\mathbf{x}|w_1)\, P(w_1)\, d\mathbf{x} + \int_{\mathcal{R}_1} p(\mathbf{x}|w_2)\, P(w_2)\, d\mathbf{x}.$$
- Minimum error achieved for $\mathbf{x}^*=\mathbf{x}_B$ is called the *Bayes error rate*
  - It depends on the class ambiguity
  - It can not be reduced by ANY classifier or data set using the given features

iNTRO

No Free Lunch Theorem [10]

- "Any two algorithms are equivalent when their performance is averaged across all possible problems" (Wolpert)
- Without assumptions on the task, no classifier is superior to any other (including random guesses)!

- Some assumptions – like continuity - are quite valid irl
- Mainly a theoretical result, but gives valuable insight
- There is no "best" classifier (SVM is not "better" than kNN)
- Choose classifier that suits the task – just like you chose your lunch






- But remember - there is always a price to pay

References

1 J. Zhang, K. Huang, Y. Yu, and T. Tan. Boosted local structured hog-lbp for object localization. In CVPR, 2010.

2. P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. PAMI, 32(9):1627–1645, 2010.

3. A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In ICCV, 2009.

4. H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In ICCV, 2009.

5. K. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In ICCV, 2011

6. C. Lampert, M. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In CVPR, 2008.

7. R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, 2nd Edition, Wiley, 2009.

8. Y. Freund, R. Schapire, A short introduction to boosting, Japanese Society for Artificial Intelligence, Vol. 14, No. 5. (1999), pp. 771-780.

9. R. Schapire, Y. Freund, A decision theoretic generalization of on-line learning and an application to Boosting, Journal of Computer and System Sciences, 1997, 55: 119-139.

10. D. Wolpert, The Lack of A Priori Distinctions between Learning Algorithms, Neural Computation, pp. 1341–1390, 1996.

11. T. K. Ho, A data complexity analysis of comparative advantages of decision forest constructors, Pattern Analysis and Applications, 5 (2002) 102-112.

12. T.G. Dietterich, Ensemble Methods in Machine Learning, p.1-15, MCS '00 Proceedings of the First International Workshop on Multiple Classifier Systems, 2000.

13. M. Jagersand, O. Fuentes and R. Nelson, Experimental evaluation of uncalibrated visual servoing for precision manipulation, in Proc. IEEE Int. Conf. Robot. Automat. (ICRA)}, pp, 2874-2880, 1997.