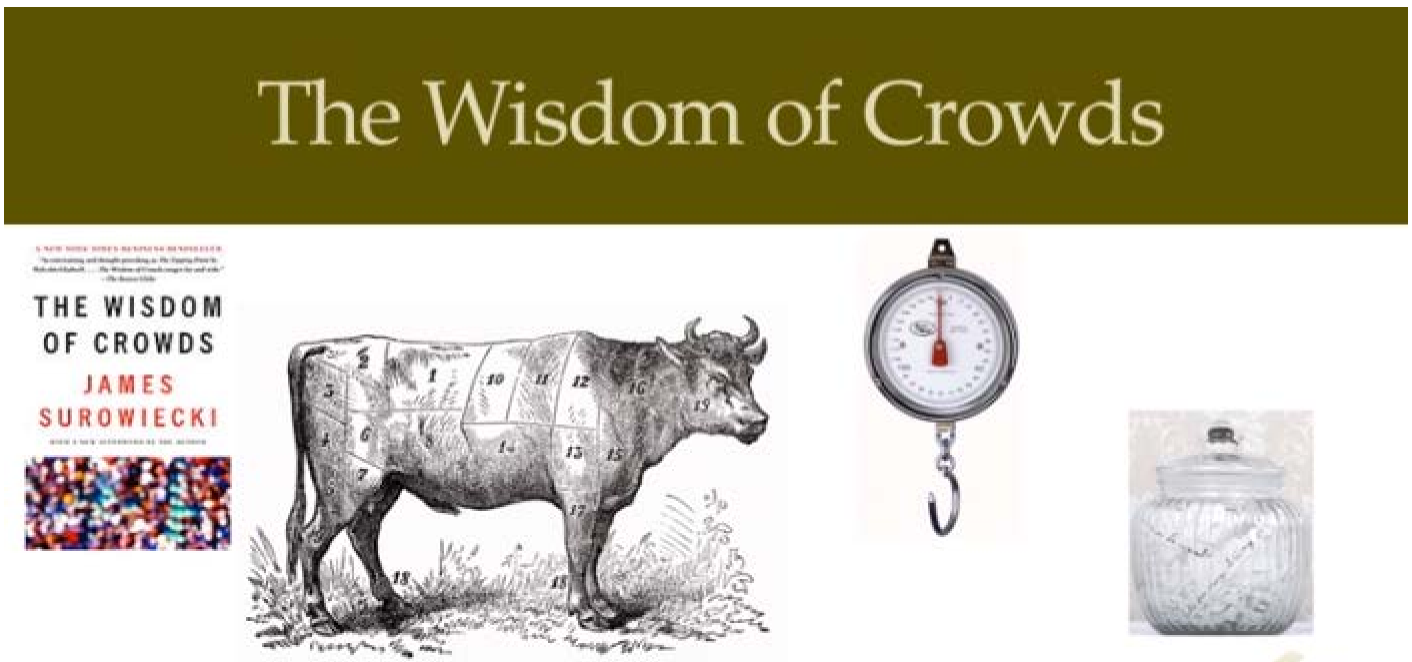
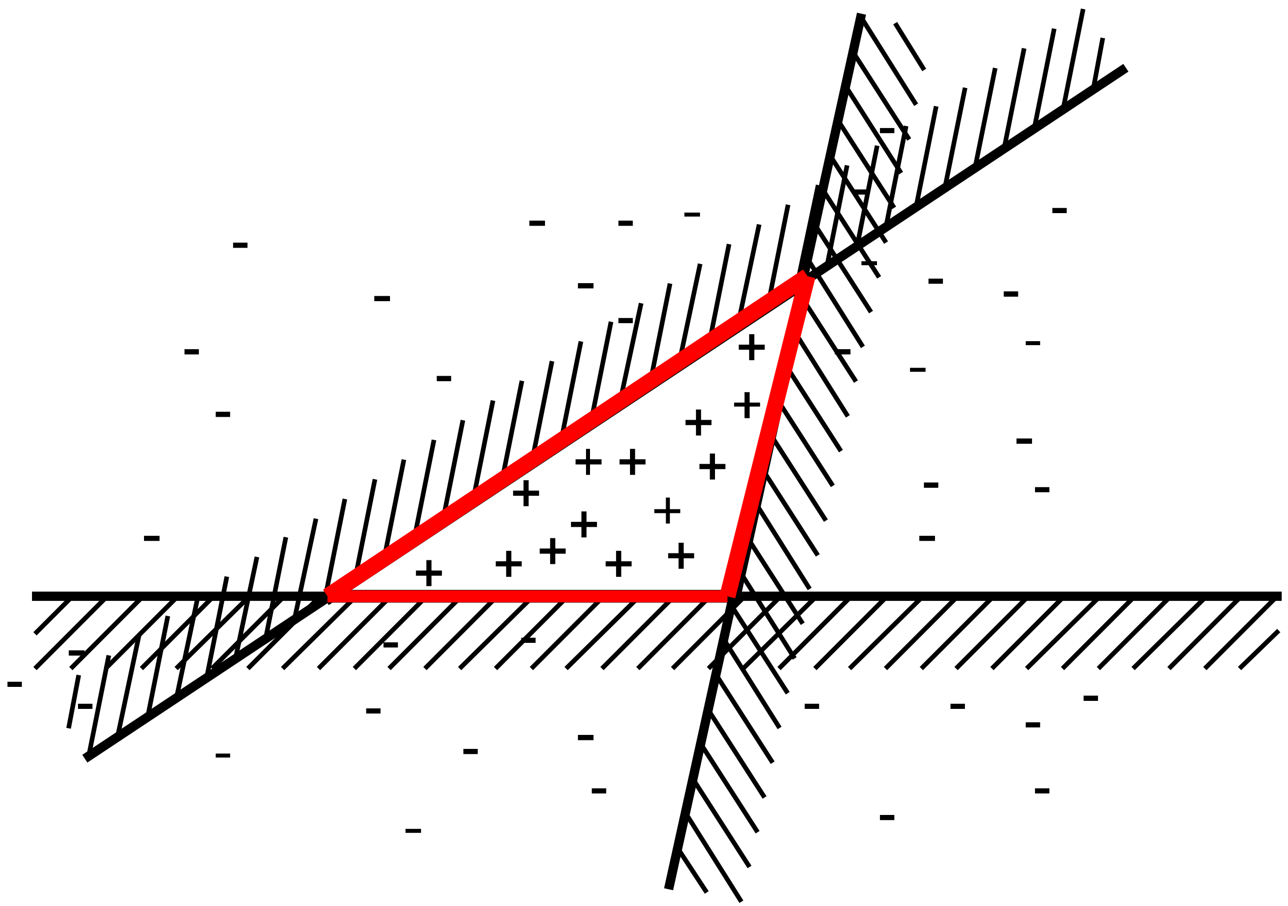
“Wisdom of Crowds” (Francis Galton)

- Many idiots (“weak learners”) are often better than one expert



Combination of Several “decision stumps”

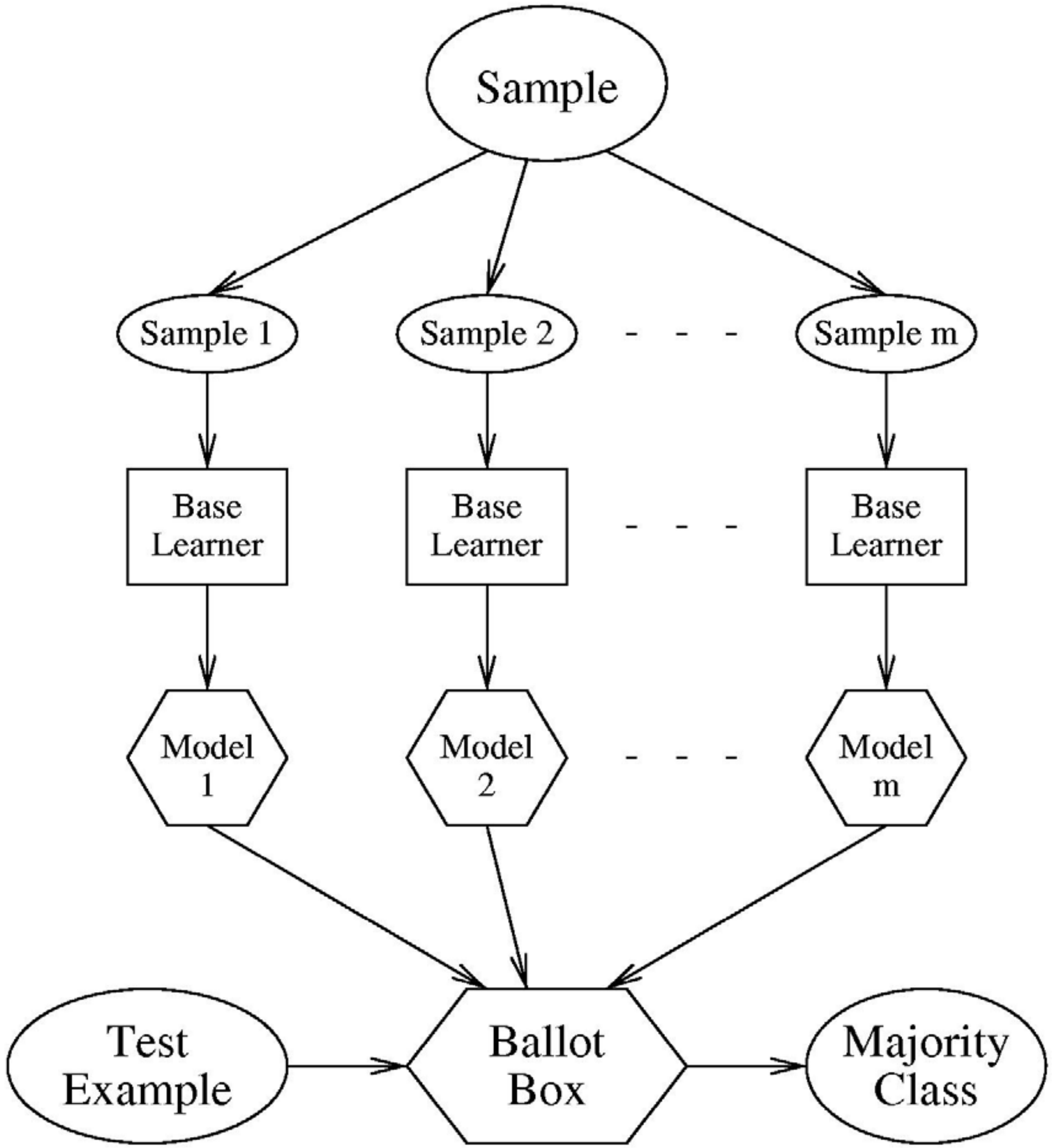


Ensemble Methods

- Instead of learning one model, learn several and combine, e.g.
  - Averaging
  - Bagging
  - Random Forests
  - Boosting
- All can be applied on top of any “weak learner”, but particularly popular with decision trees/stumps

Bagging

- Generate "bootstrap" replicates of training set by sampling with replacement
- Learn one model on each replicate
- Combine by uniform voting



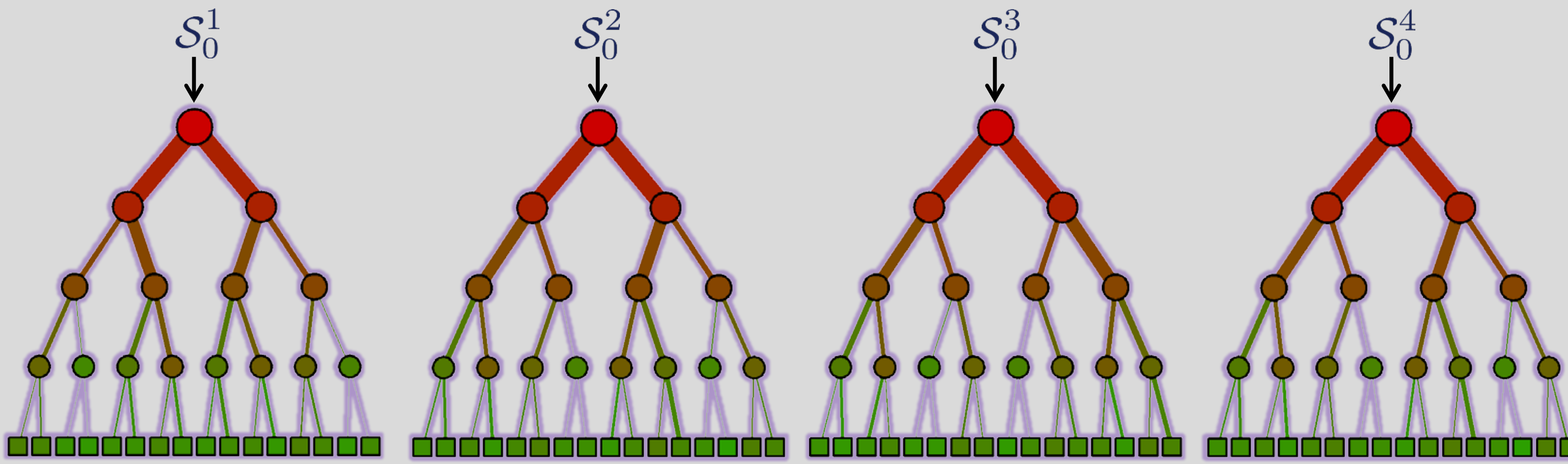
Sample
Sample 1
Sample 2
Sample m
Base Learner
Base Learner
Base Learner
Model 1
Model 2
Model m
Test Example
Ballot Box
Majority Class

Bagging on Trees

1) Bagging (randomizing the training set)

$\mathcal{S}_0$	The full training set
$\mathcal{S}_0^t \subset \mathcal{S}_0$	The randomly sampled subset of training data made available for the tree *t*

Forest training



Efficient training

Random Forests

- With bagging, often the trees look very correlated. Why?
- All trees pick the same very good splits
  - The trees become correlated, so averaging doesn't by as much
- What can we do?
  - Add more randomness:
  - at each node, allow a random subset of *k* splits
  - Typically $k = \sqrt{n}$

Decision forest model: the randomness model

2) Randomized node optimization (RNO)

$\mathcal{T}$	The full set of all possible node test parameters
$\mathcal{T}_j \subset \mathcal{T}$	For each node the set of randomly sampled features
$\rho = \lvert\mathcal{T}_j\rvert$	Randomness control parameter. For $\rho = \lvert\mathcal{T}\rvert$ no randomness and maximum tree correlation. For $\rho = 1$ max randomness and minimum tree correlation.

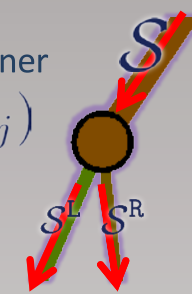
Node training

Node weak learner

$$h(\mathbf{v}, \boldsymbol{\theta}_j)$$

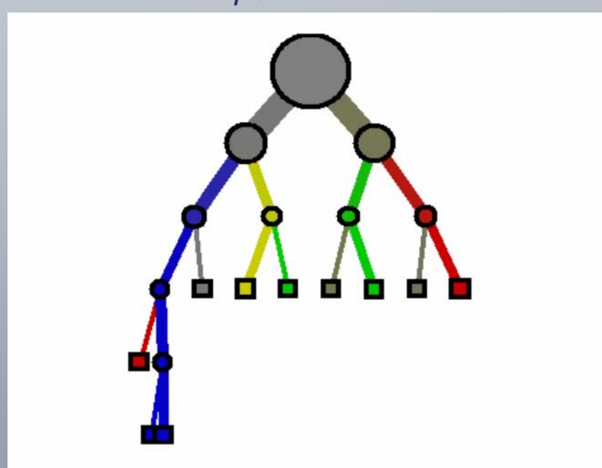
Node test params

$$\boldsymbol{\theta} \in \mathcal{T}_j$$

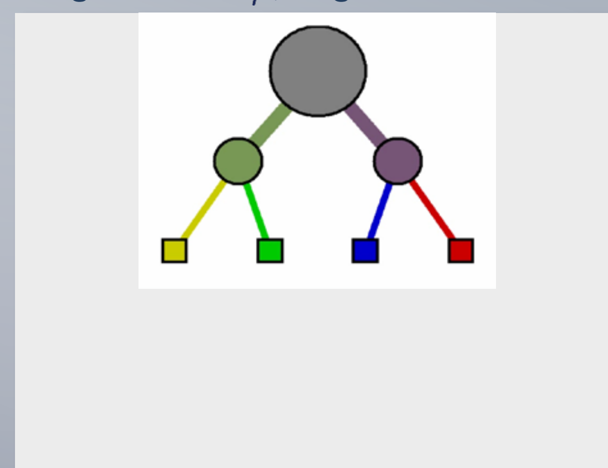


The effect of ρ

Small value of ρ; little tree correlation.



Large value of ρ; large tree correlation.



Classification forest: the weak learner model

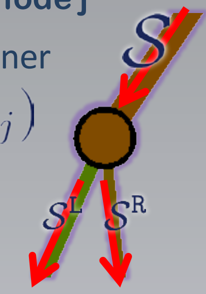
Splitting data at node j

Node weak learner

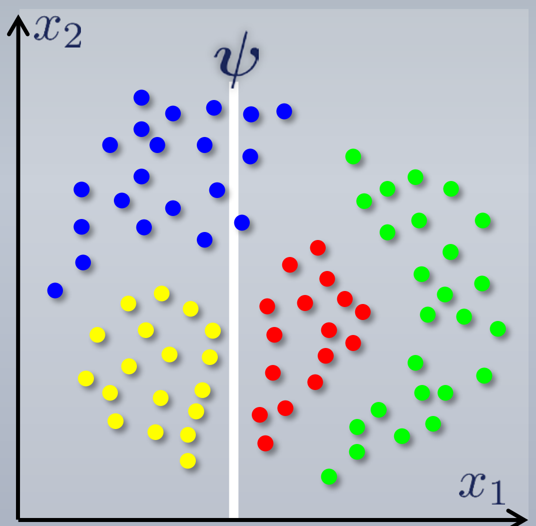
$$h(\mathbf{v}, \boldsymbol{\theta}_j)$$

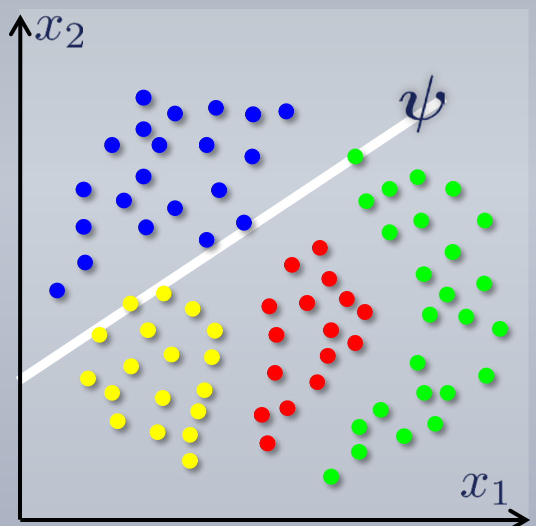
Node test params

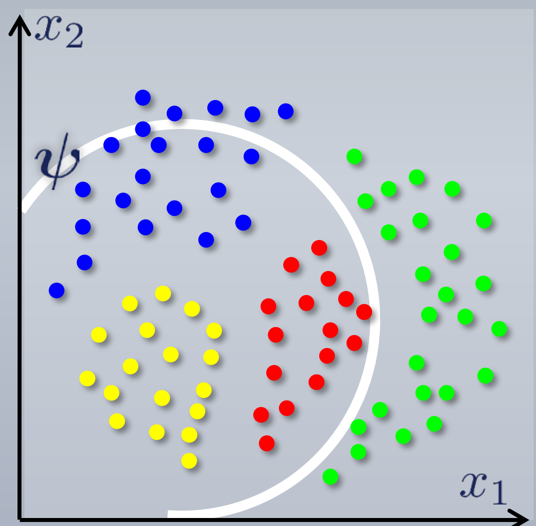
$$\boldsymbol{\theta} \in \mathcal{T}_j$$



Examples of weak learners







Weak learner: axis aligned

$$h(\mathbf{v}, \boldsymbol{\theta}) = [\tau_1 > \boldsymbol{\phi}(\mathbf{v}) \cdot \boldsymbol{\psi} > \tau_2]$$

Feature response for 2D example. $\boldsymbol{\phi}(\mathbf{v}) = (x_1 \ x_2 \ 1)^\top$

With $\boldsymbol{\psi} = (1 \ 0 \ \psi_3)$ or $\boldsymbol{\psi} = (0 \ 1 \ \psi_3)$

Weak learner: oriented line

$$h(\mathbf{v}, \boldsymbol{\theta}) = [\tau_1 > \boldsymbol{\phi}(\mathbf{v}) \cdot \boldsymbol{\psi} > \tau_2]$$

Feature response for 2D example. $\boldsymbol{\phi}(\mathbf{v}) = (x_1 \ x_2 \ 1)^\top$

With $\boldsymbol{\psi} \in \mathbb{R}^3$ a generic line in homog. coordinates.

Weak learner: conic section

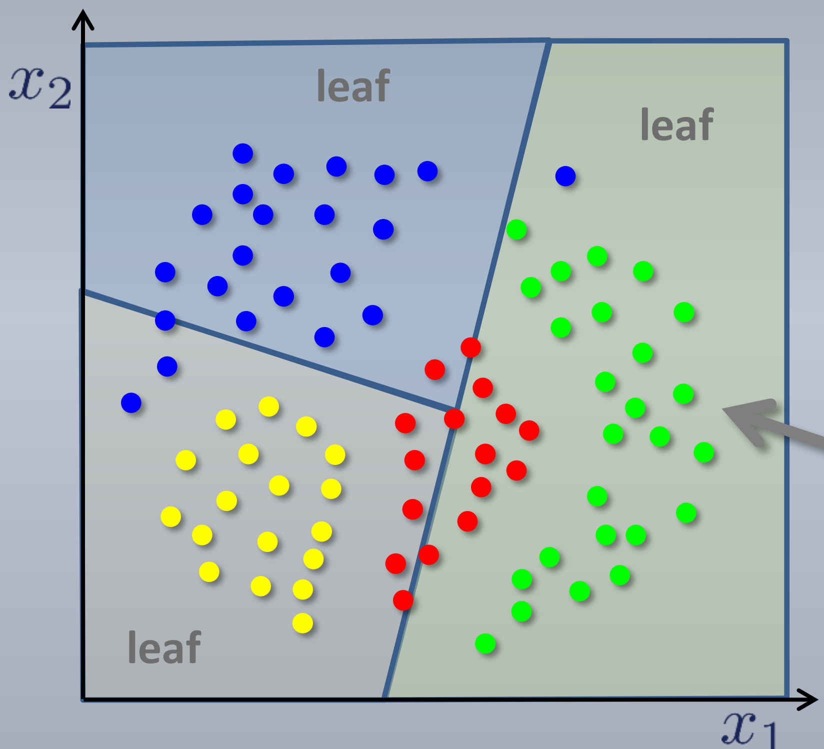
$$h(\mathbf{v}, \boldsymbol{\theta}) = \left[\tau_1 > \boldsymbol{\phi}^\top(\mathbf{v}) \ \boldsymbol{\psi} \ \boldsymbol{\phi}(\mathbf{v}) > \tau_2\right]$$

Feature response for 2D example. $\boldsymbol{\phi}(\mathbf{v}) = (x_1 \ x_2 \ 1)^\top$

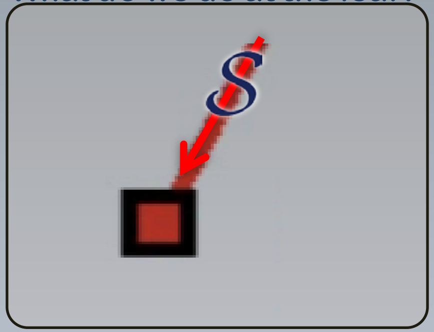
With $\boldsymbol{\psi} \in \mathbb{R}^{3\times 3}$ a matrix representing a conic.

In general $\boldsymbol{\phi}$ may select only a very small subset of features $\boldsymbol{\phi}(\mathbf{v}) : \mathbb{R}^d \rightarrow \mathbb{R}^{d'+1}, \ \ d' << d$

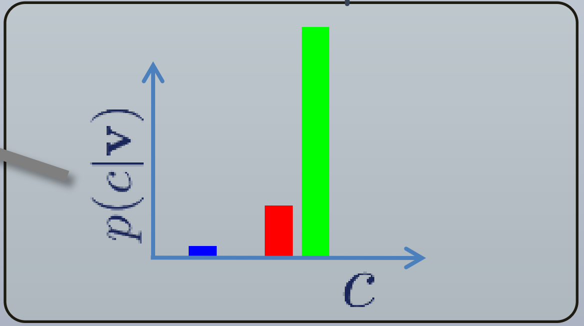
Classification forest: the prediction model



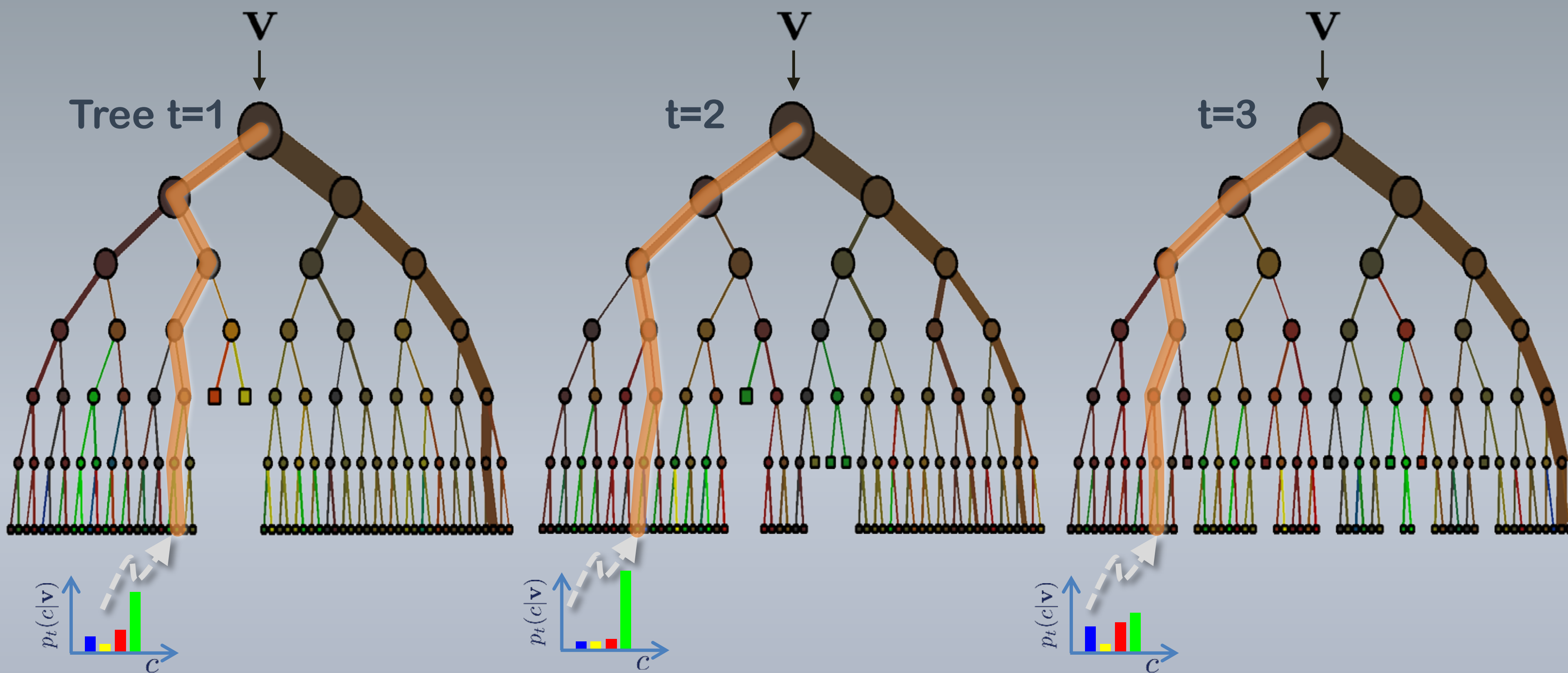
What do we do at the leaf?



Prediction model: probabilistic

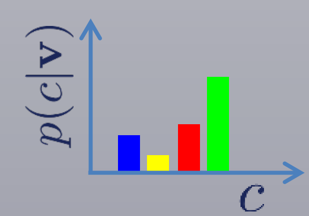


Classification forest: the ensemble model

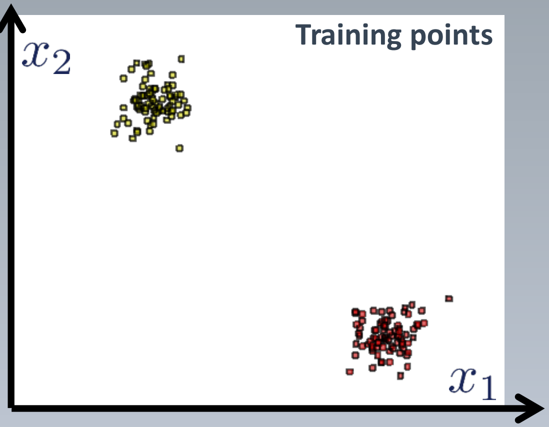


The ensemble model

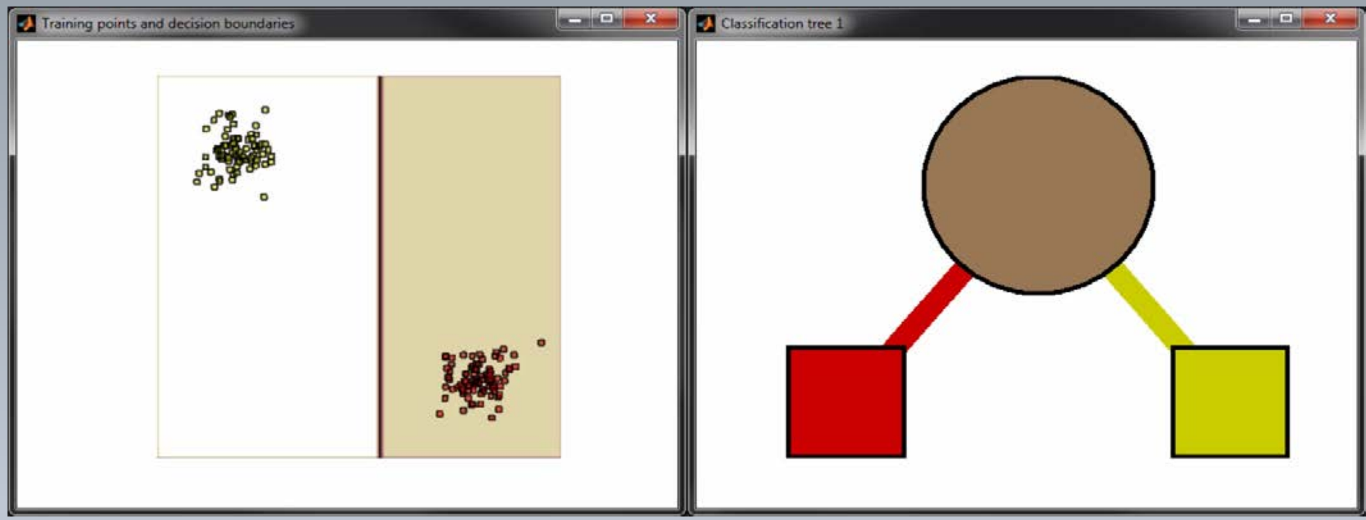
Forest output probability $$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t}^{T} p_t(c|\mathbf{v})$$



Classification forest: effect of the weak learner model



Training different trees in the forest



Testing different trees in the forest

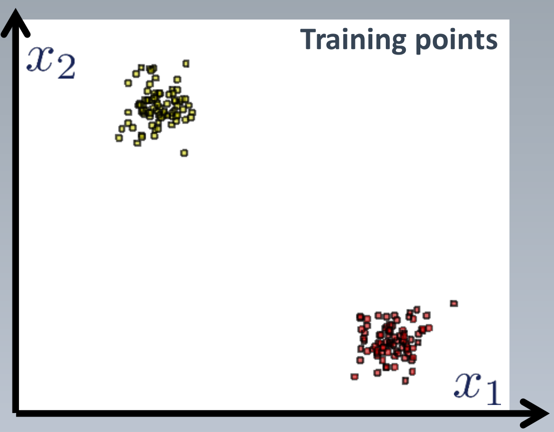
Three concepts to keep in mind:

- "Accuracy of prediction"
- "Quality of confidence"
- "Generalization"

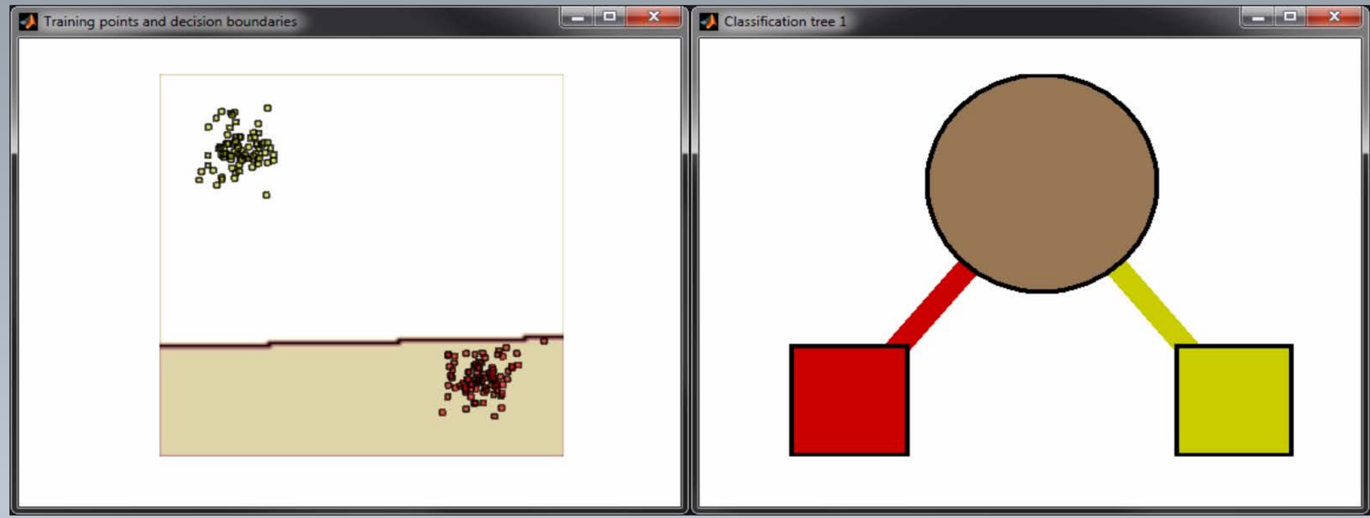
(2 videos in this page)

Parameters: T=200, D=2, weak learner = aligned, leaf model = probabilistic

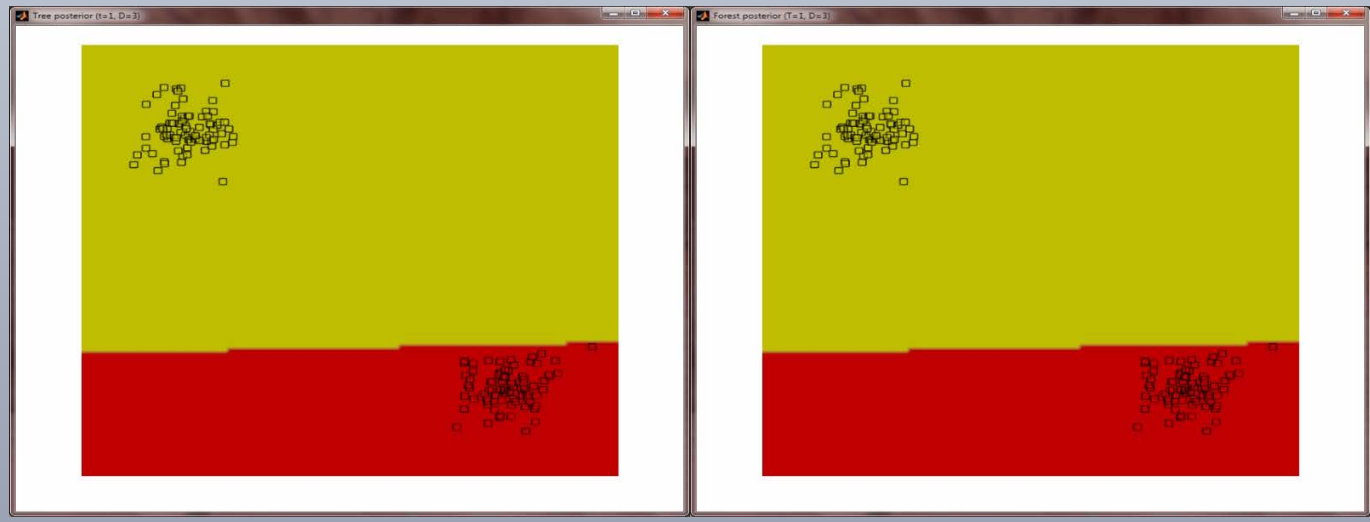
Classification forest: effect of the weak learner model



Training different trees in the forest



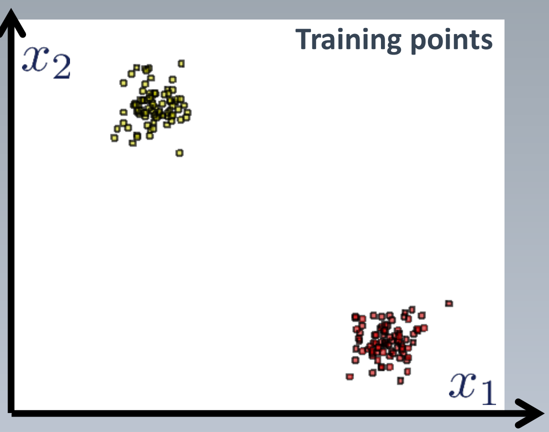
Testing different trees in the forest



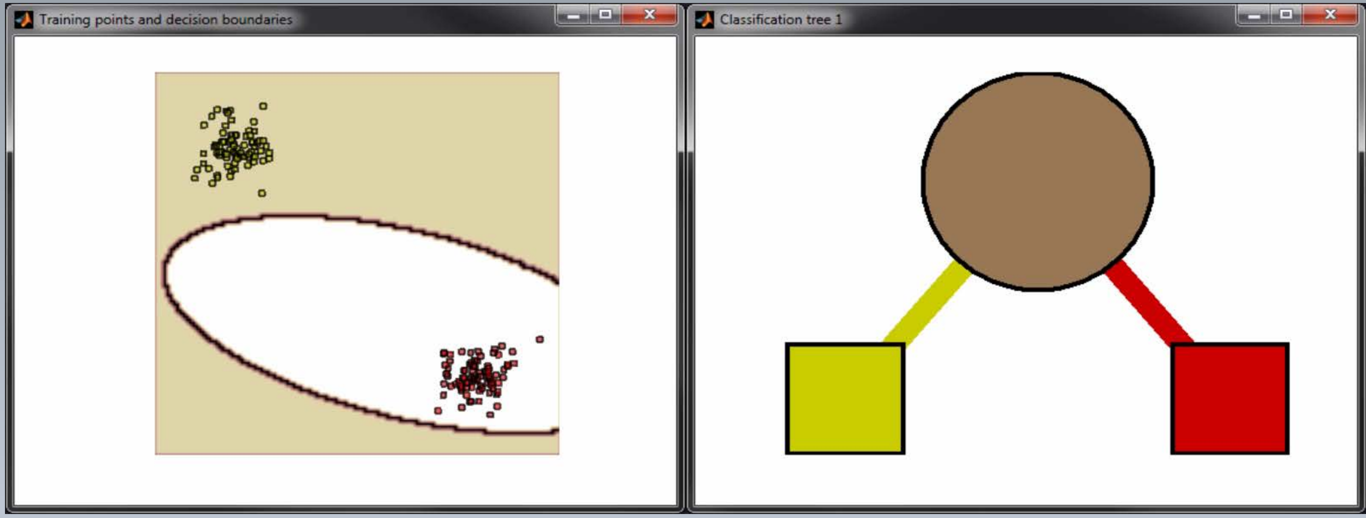
(2 videos in this page)

Parameters: T=200, D=2, weak learner = linear, leaf model = probabilistic

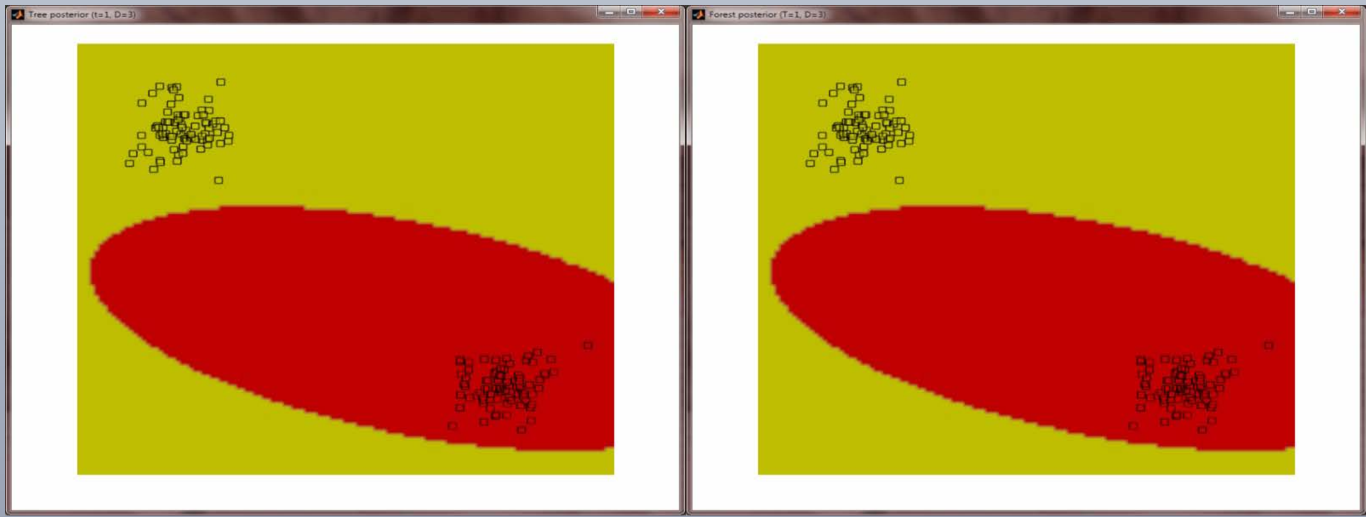
Classification forest: effect of the weak learner model



Training different trees in the forest



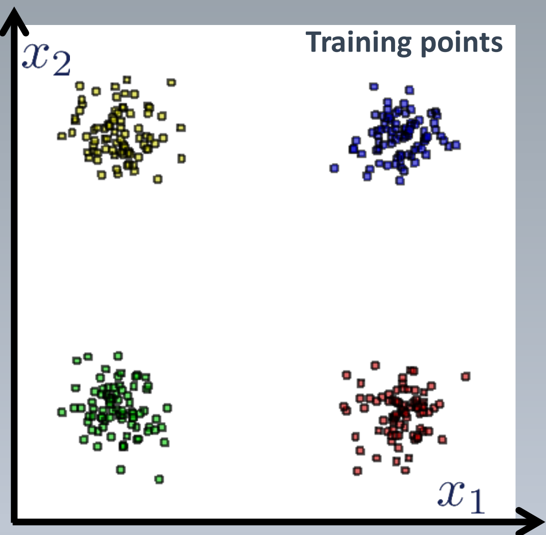
Testing different trees in the forest



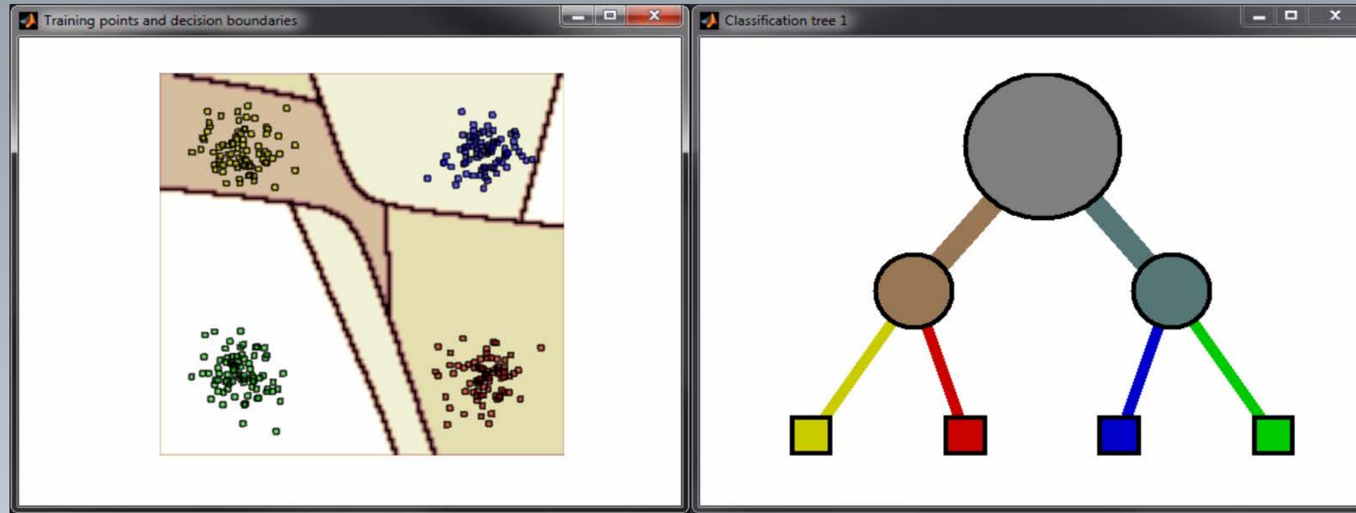
(2 videos in this page)

Parameters: T=200, D=2, weak learner = conic, leaf model = probabilistic

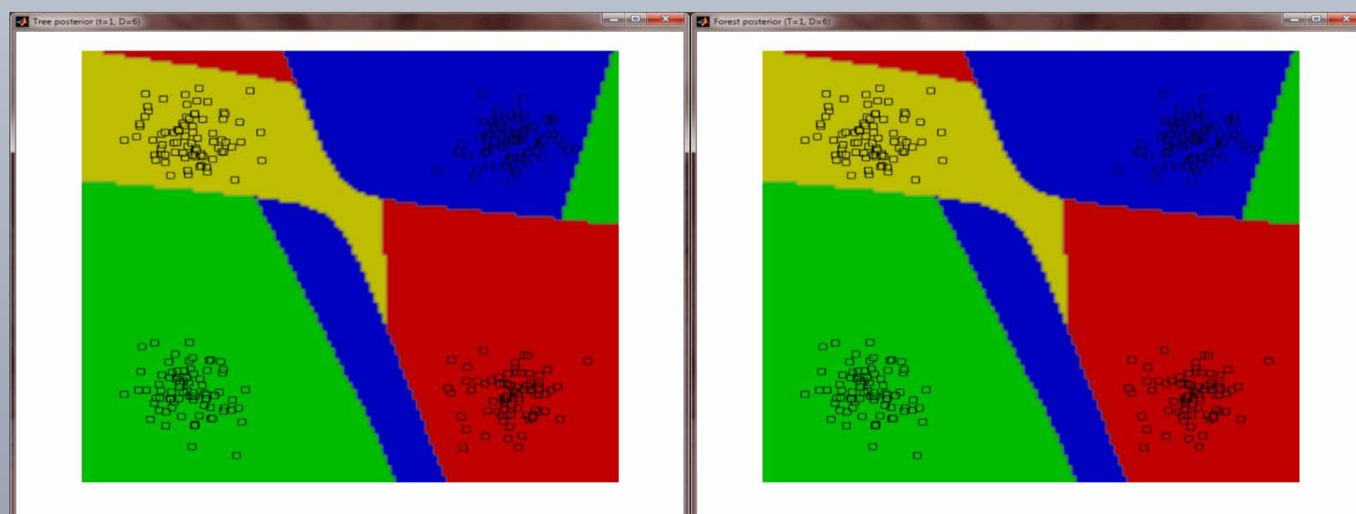
Classification forest: with >2 classes



Training different trees in the forest



Testing different trees in the forest

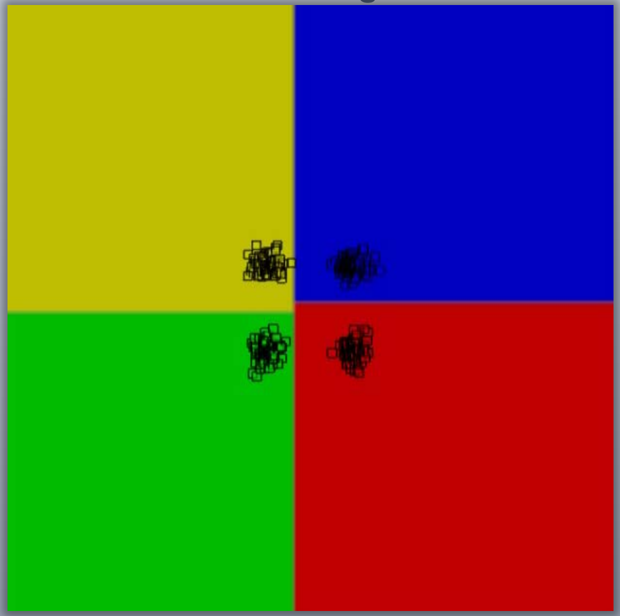


(2 videos in this page)

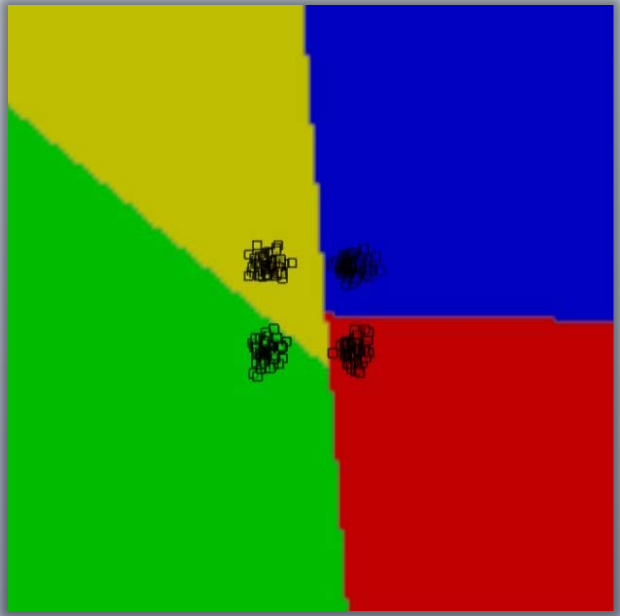
Parameters: T=200, D=3, weak learner = conic, leaf model = probabilistic

Classification forest: analysing generalization

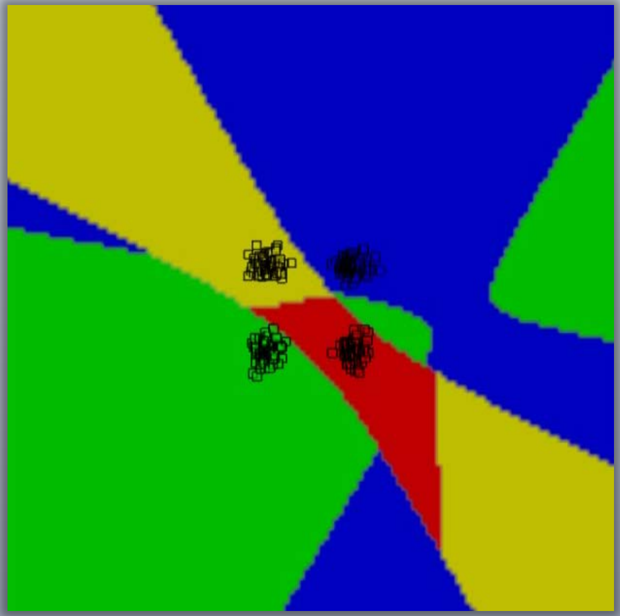
Weak learner: axis aligned



Weak learner: oriented line



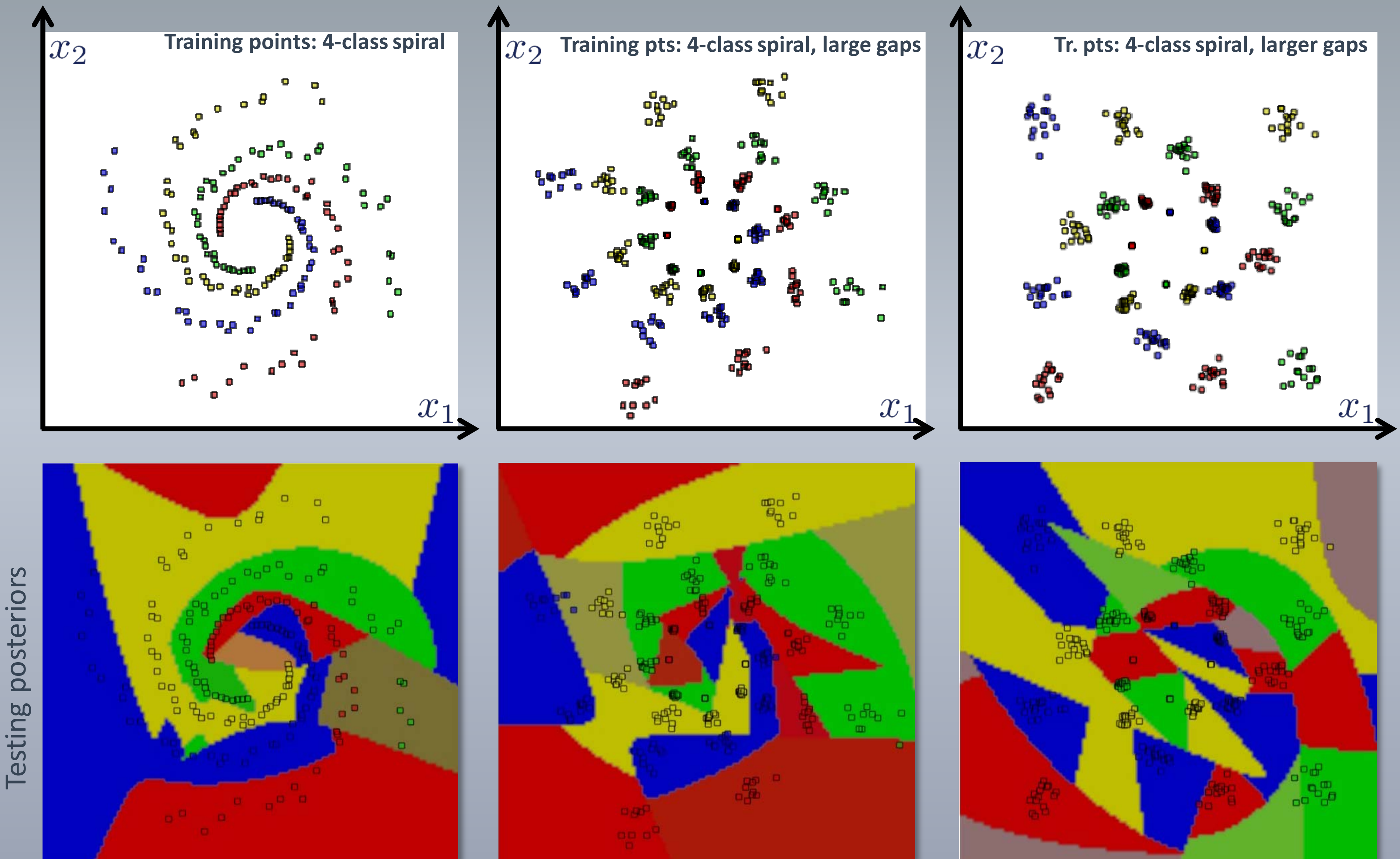
Weak learner: conic section



(3 videos in this page. Increasing T)

Parameters: T=200, D=3, leaf model = probabilistic

Classification forest: analysing generalization

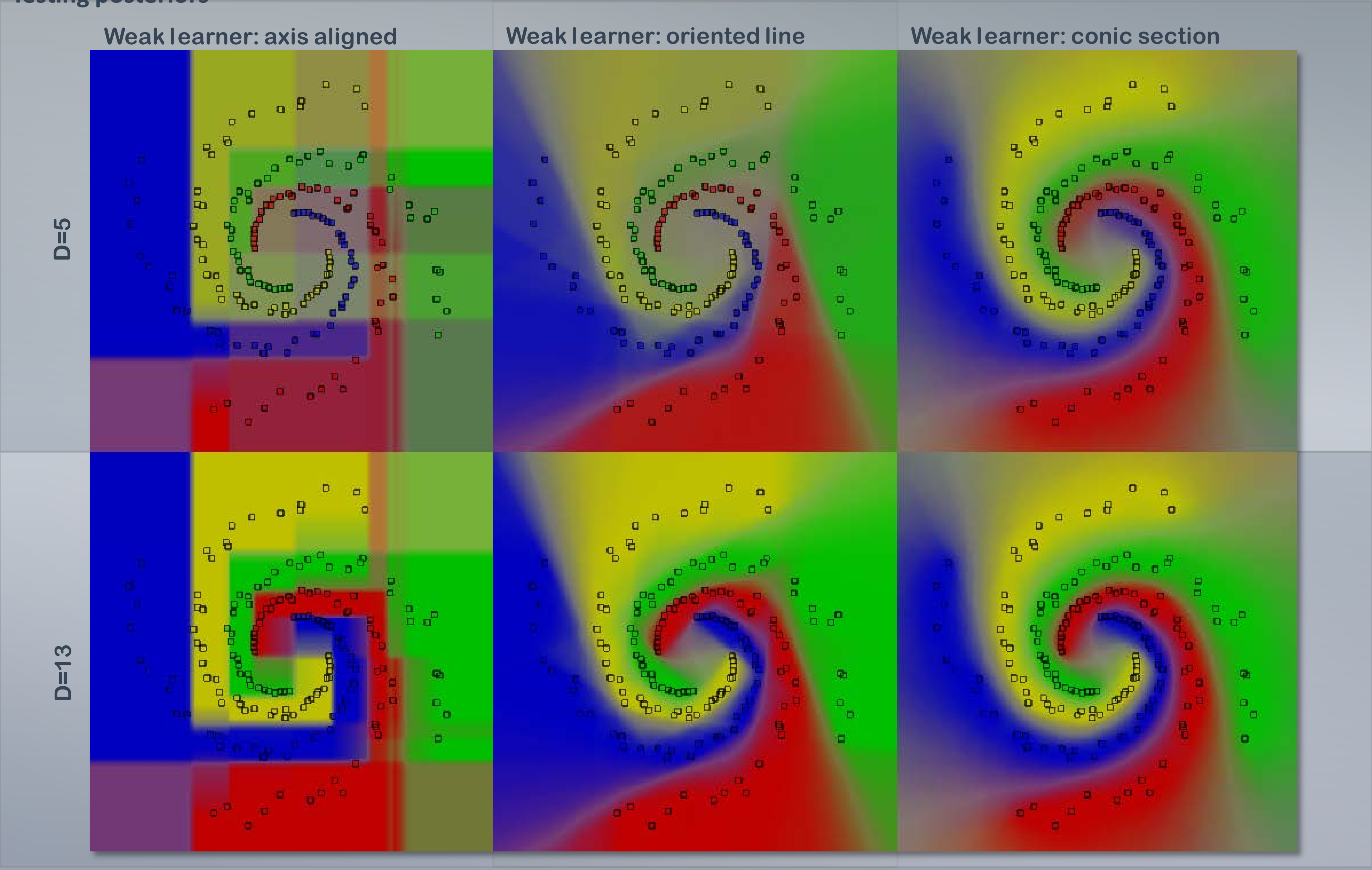


(3 videos in this page)

Parameters: T=200, D=13, w. l. = conic, predictor = prob.

Classification forest: effect of weak learner model and randomness

Testing posteriors

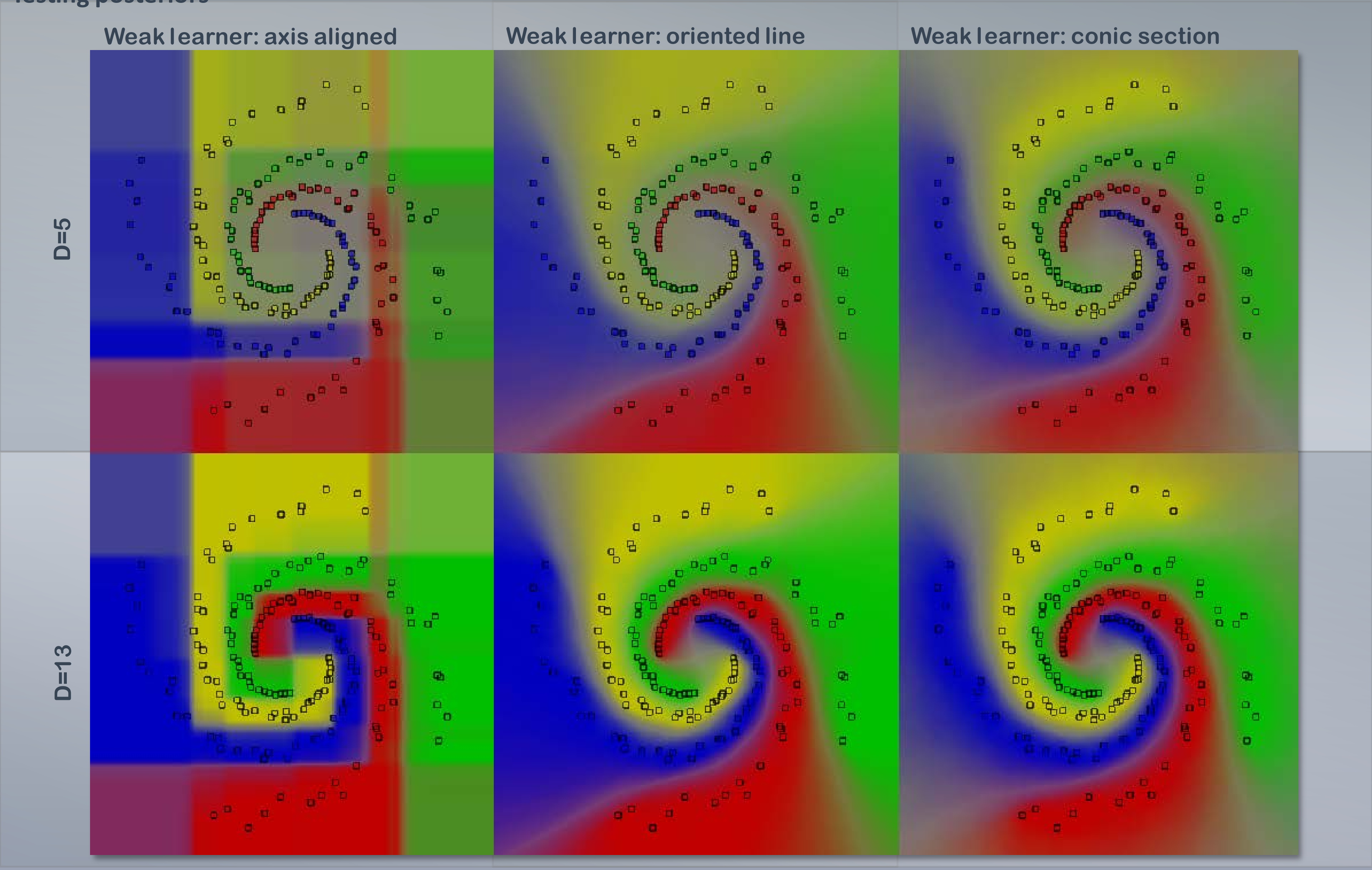


Randomness: $\rho = 500$

Parameters: T=400 predictor model = prob.

Classification forest: effect of weak learner model and randomness

Testing posteriors

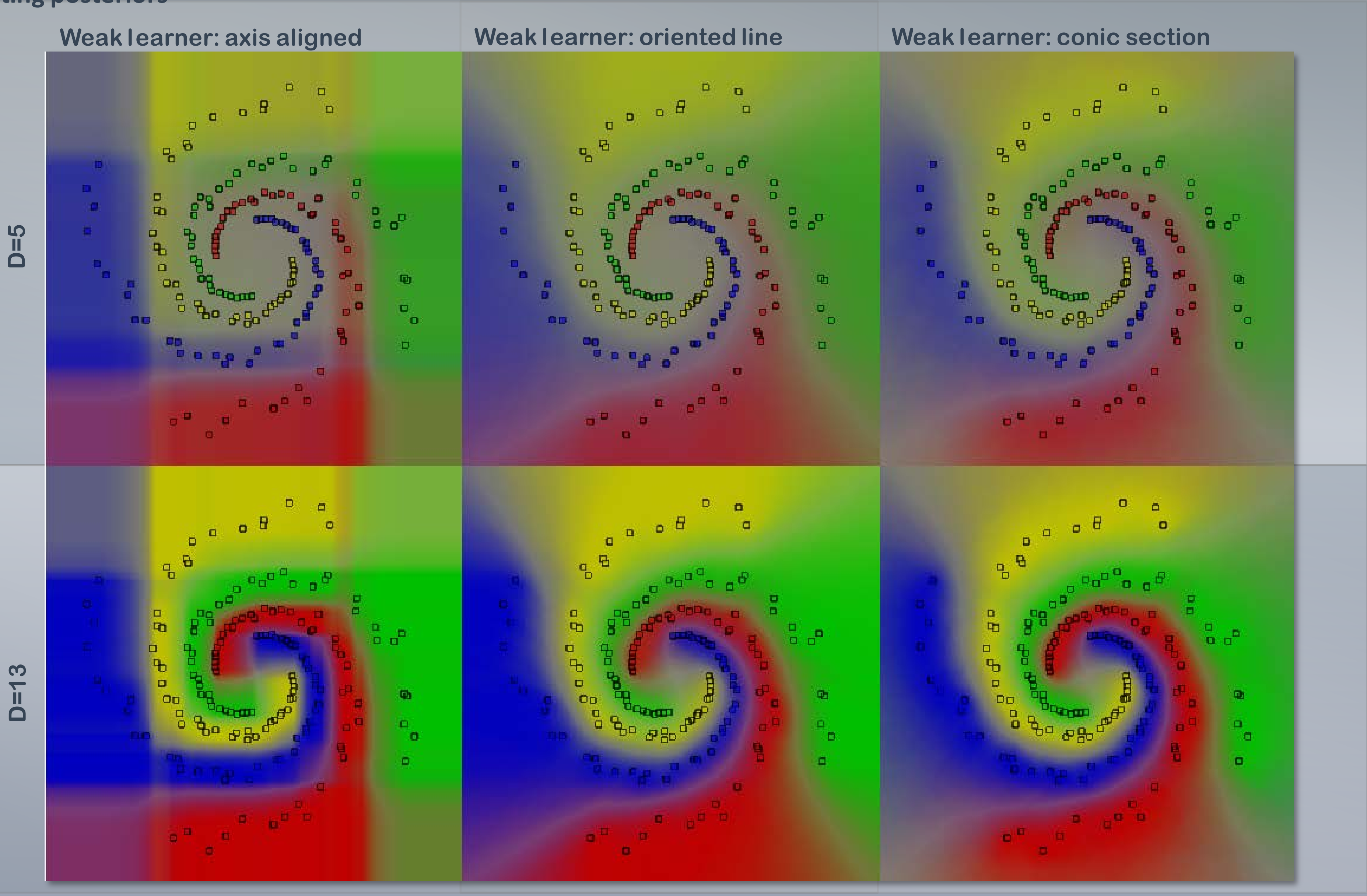


Randomness: $\rho = 50$

Parameters: T=400 predictor model = prob.

Classification forest: effect of weak learner model and randomness

Testing posteriors

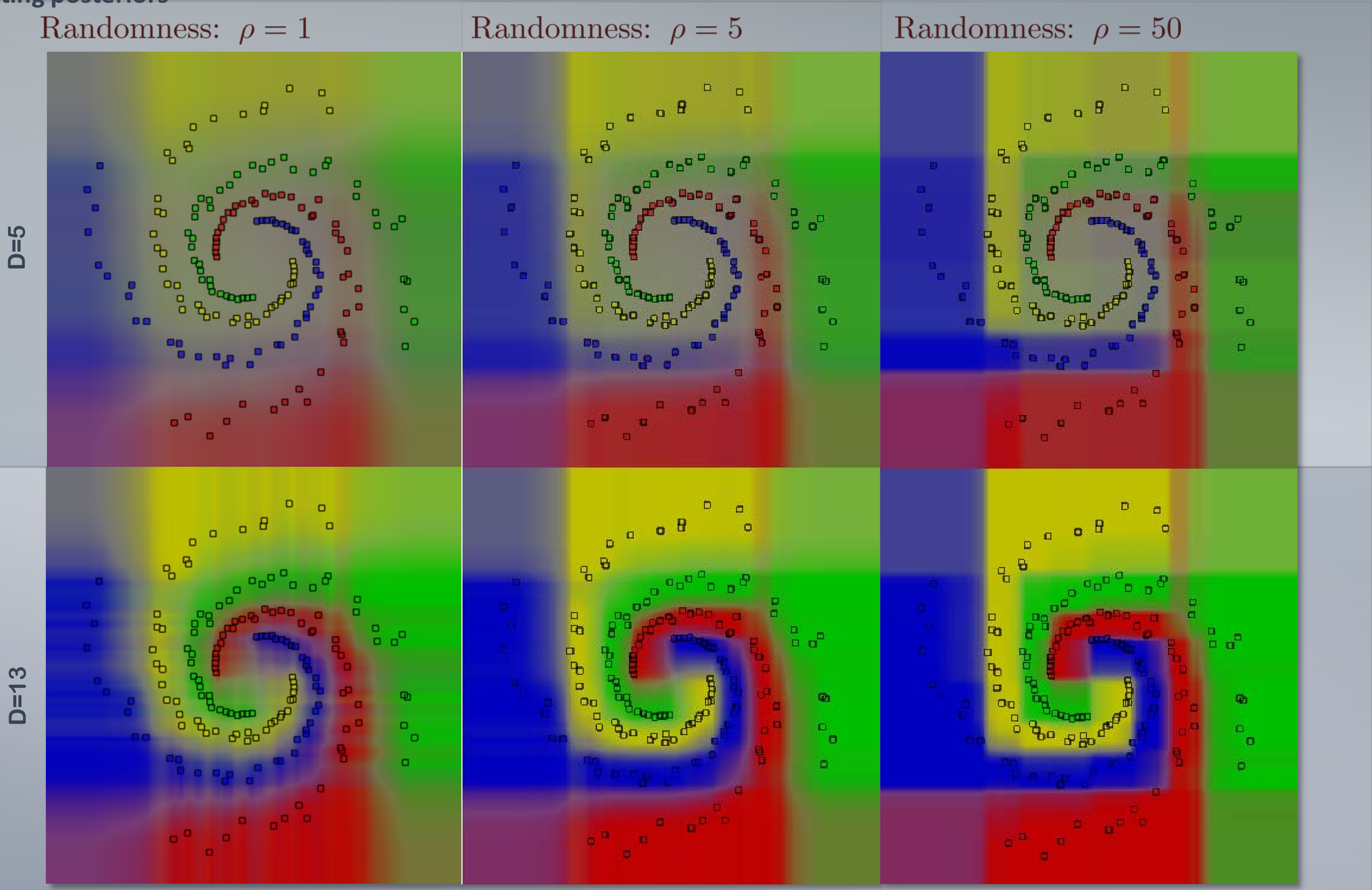


Randomness: $\rho = 5$

Parameters: T=400 predictor model = prob.

Classification forest: effect of randomness

Testing posteriors



Weak learner: axis aligned

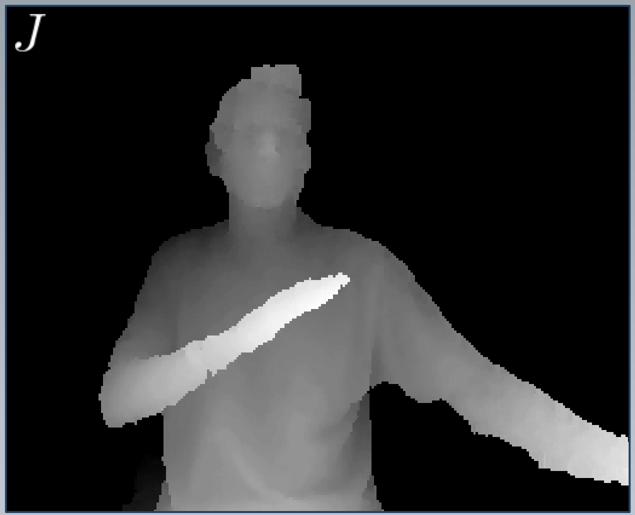
Parameters: T=400 predictor model = prob.

Classification forests in practice

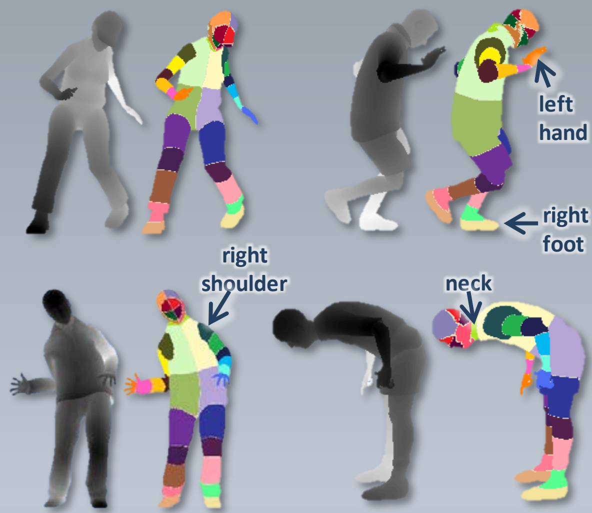
Microsoft Kinect for Xbox 360

[J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake.
Real-Time Human Pose Recognition in Parts from a Single Depth Image. *In Proc. IEEE CVPR*, June 2011.]

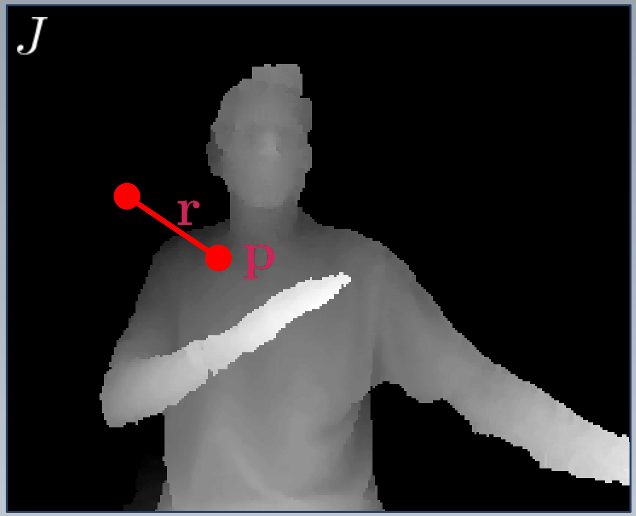
Body tracking in Microsoft Kinect for XBox 360



Input depth image



Training labelled data



Visual features

Classification forest

Labels are categorical $c \in \{\texttt{l.hand}, \texttt{r.hand}, \texttt{head}, \ldots\}$

Input data point $\mathbf{p} \in \mathbb{R}^2$

Visual features $\mathbf{v}(\mathbf{p}) = (x_1, \ldots, x_i, \ldots, x_d) \in \mathbb{R}^d$

Feature response $x_i = J(\mathbf{p}) - J\left(\mathbf{p} + \frac{\mathbf{r}_i}{J(\mathbf{p})}\right)$

Predictor model $p(c|\mathbf{v})$

Objective function $I = H(\mathcal{S}_j) - \sum_{i=\mathrm{L,R}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} H(\mathcal{S}_j^i)$

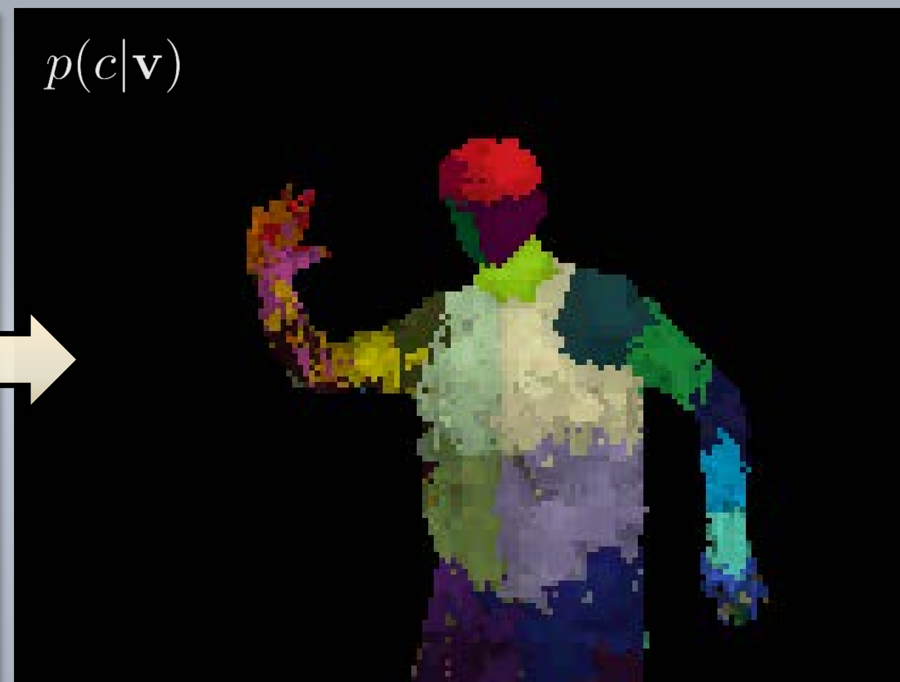
Node parameters $\boldsymbol{\theta} = (\mathbf{r}, \tau)$

Node training $\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} I(\mathcal{S}_j, \boldsymbol{\theta})$

Weak learner $h(\mathbf{v}, \boldsymbol{\theta}) = [\phi(\mathbf{v}, \mathbf{r}) > \tau]$

Body tracking in Microsoft Kinect for XBox 360

Input depth image (bg removed)



Inferred body parts posterior

(2 videos here)

Boosting

- Defines a classifier using an additive model:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier

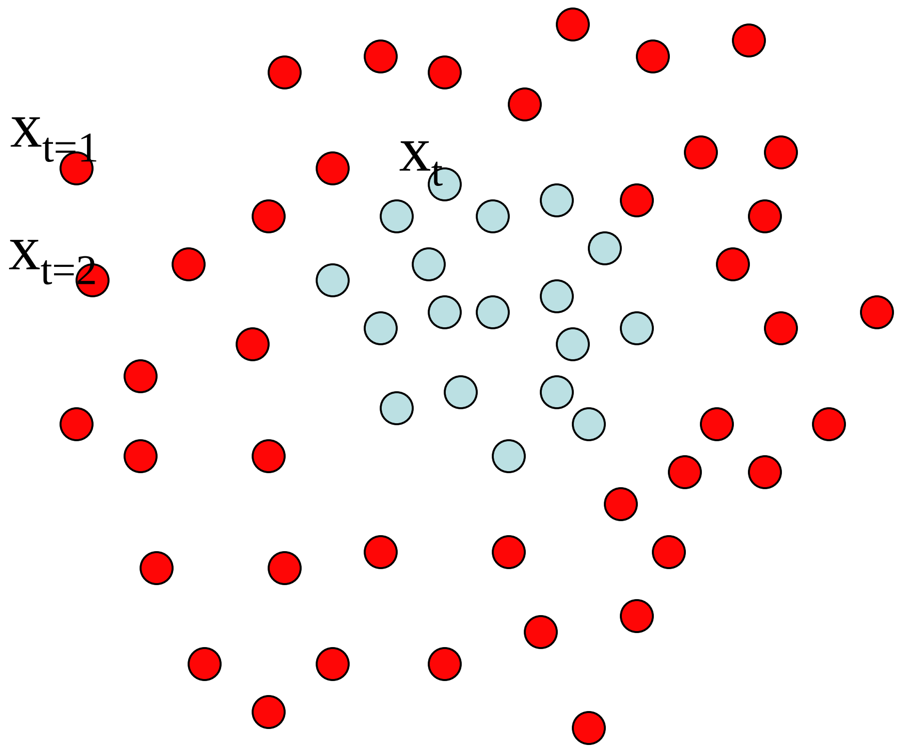
Features vector

Weight

Weak classifier

Boosting

- It is a sequential procedure:



Each data point has a class label:

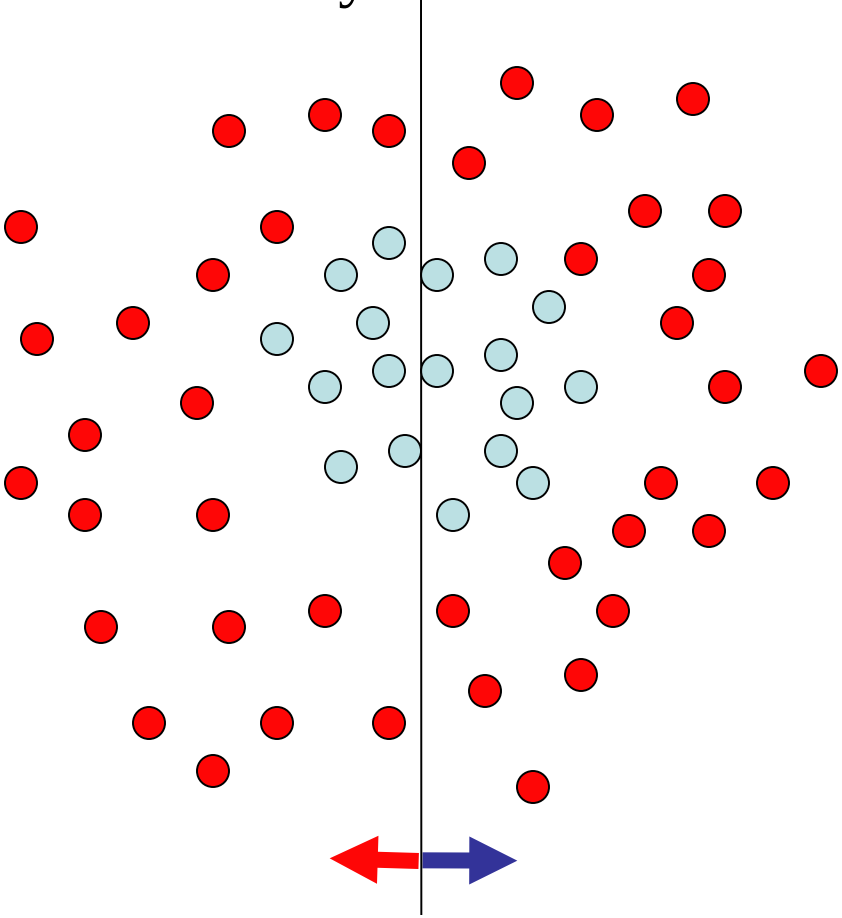
$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

and a weight:

$w_t = 1$

Toy example

Weak learners from the family of lines



Each data point has a class label:

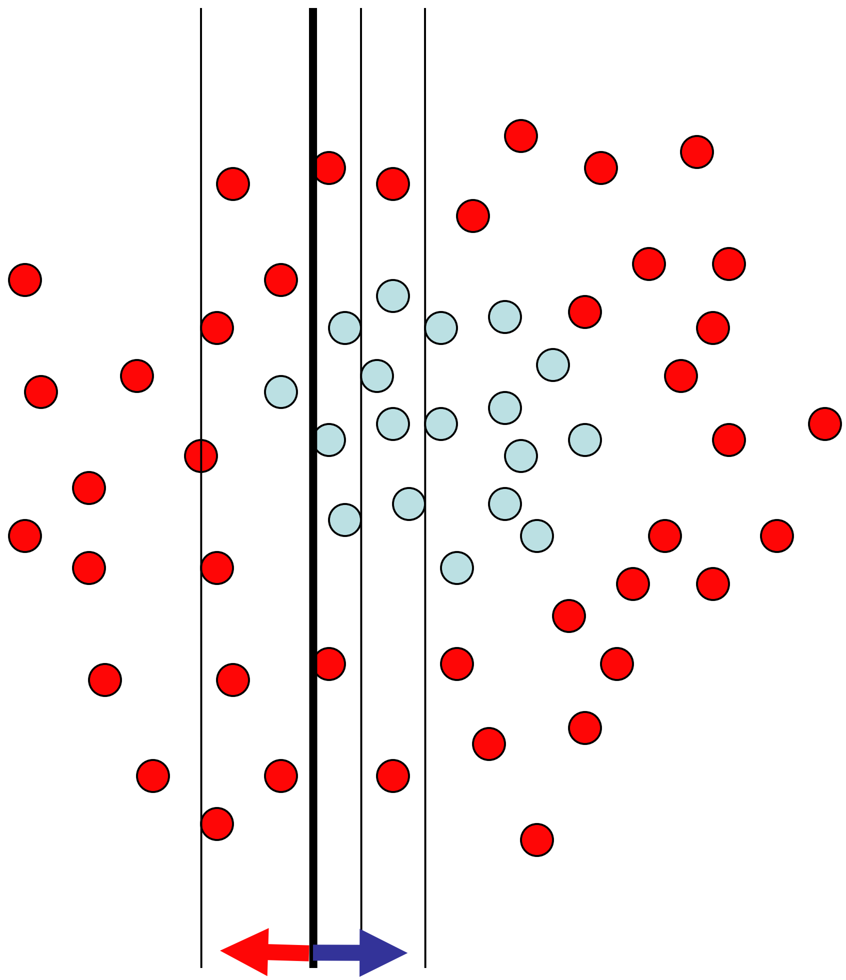
$$y_t = \begin{cases} +1 \; (\quad) \\ -1 \; (\quad) \end{cases}$$

and a weight:

$w_t = 1$

h => p(error) = 0.5 it is at chance

Toy example



Each data point has a class label:

$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

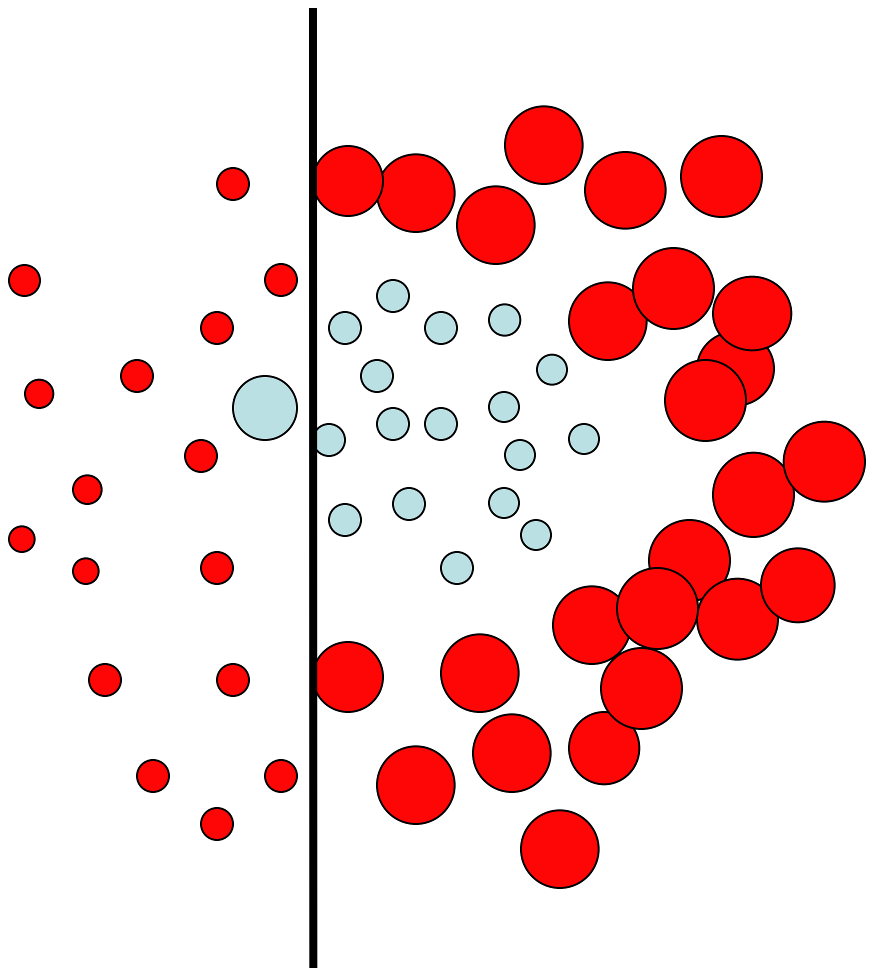
and a weight:

$w_t = 1$

This one seems to be the best

This is a ‘**weak classifier**’: It performs slightly better than chance.

Toy example



Each data point has a class label:

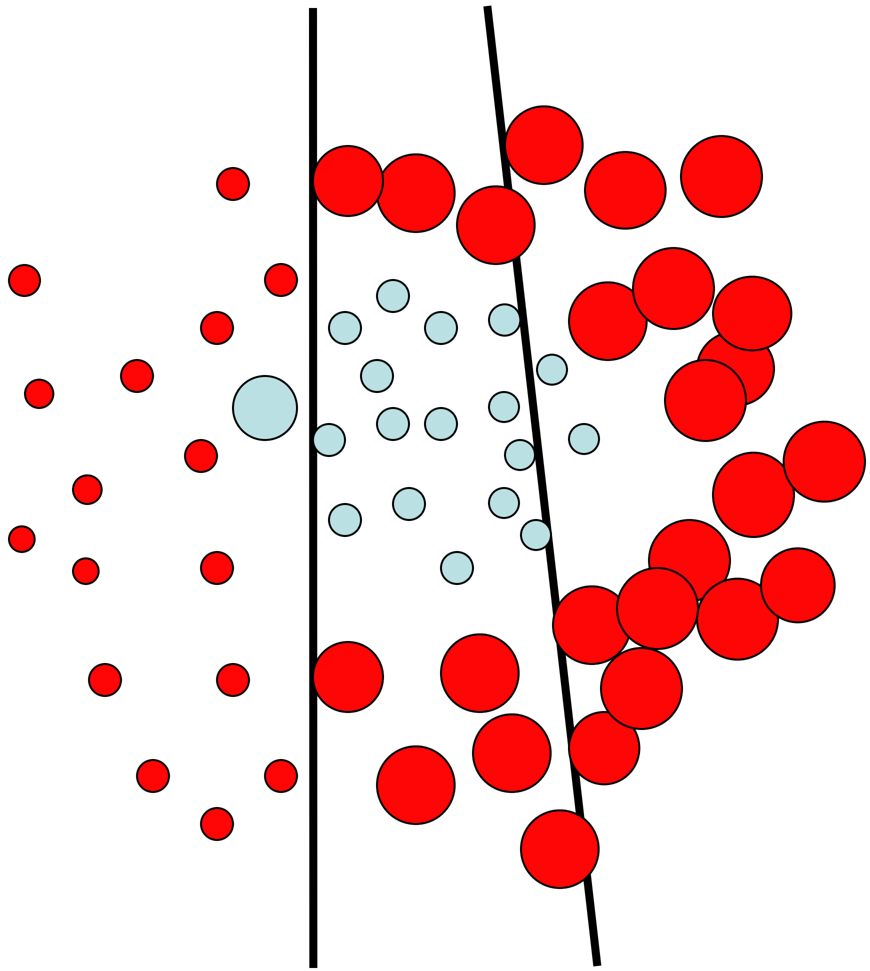
$$y_t = \begin{cases} +1 \ (\quad) \\ -1 \ (\quad) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has a class label:

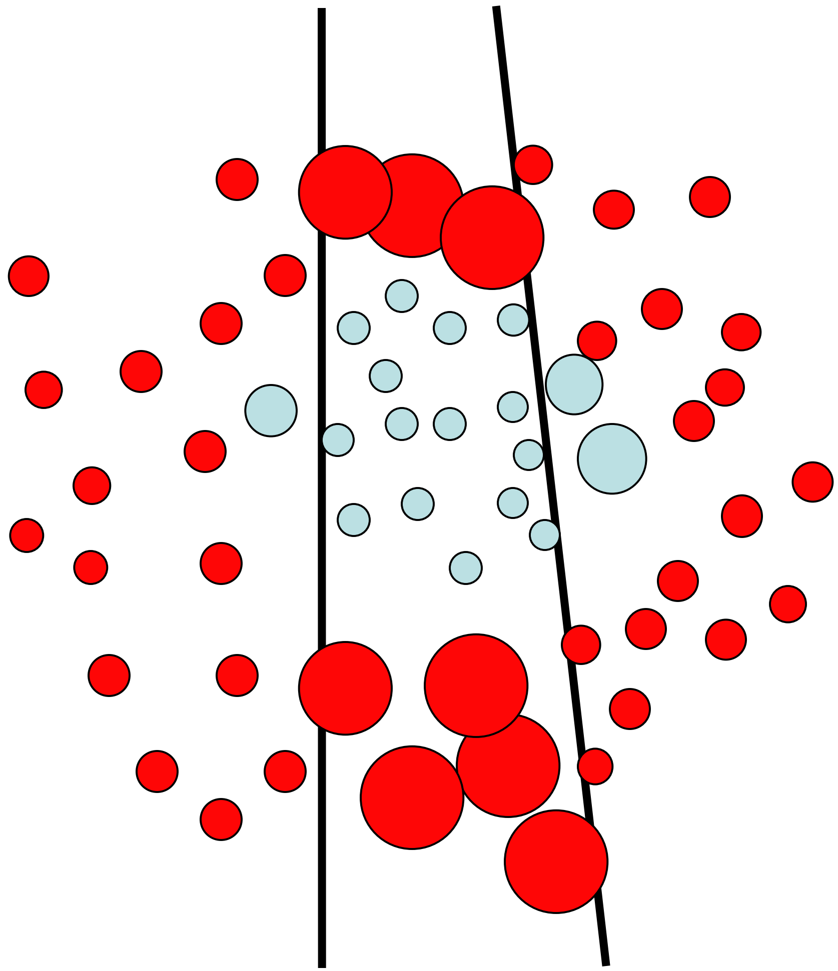
$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has a class label:

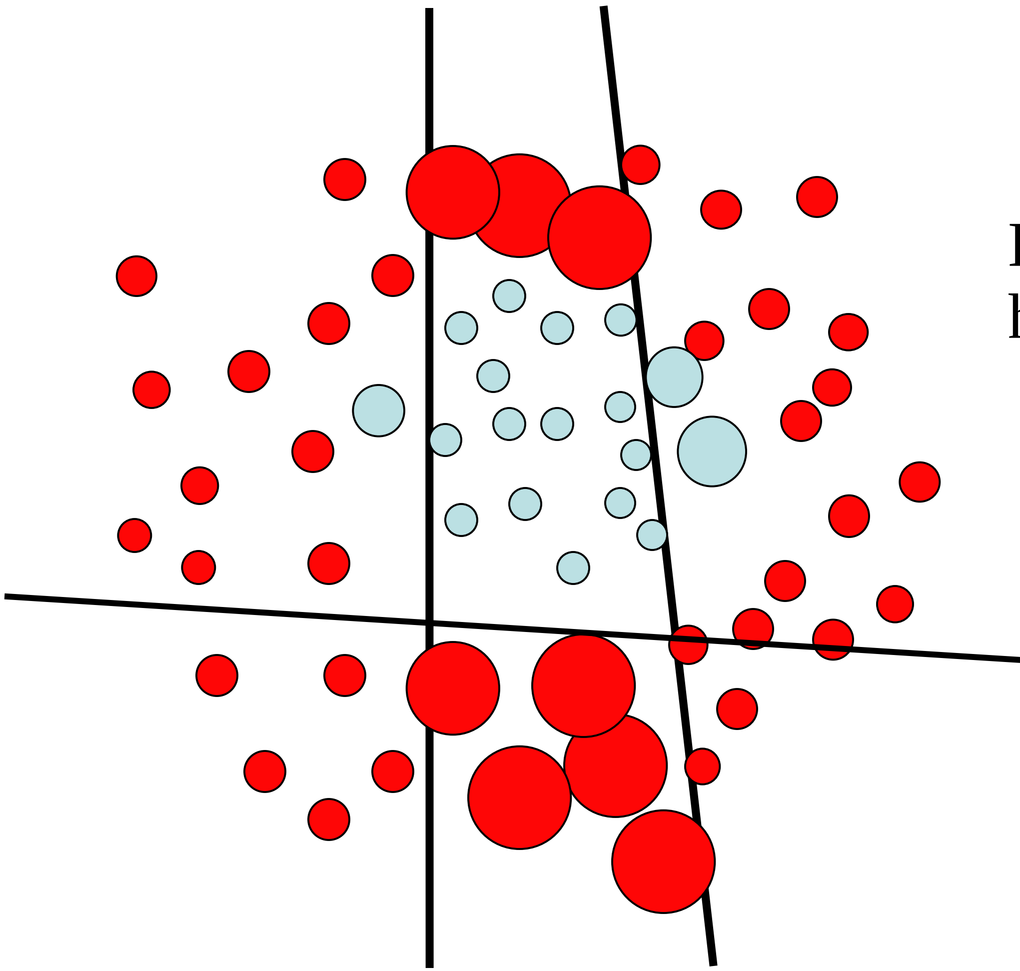
$$y_t = \begin{cases} +1 \; (\quad) \\ -1 \; (\quad) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has a class label:

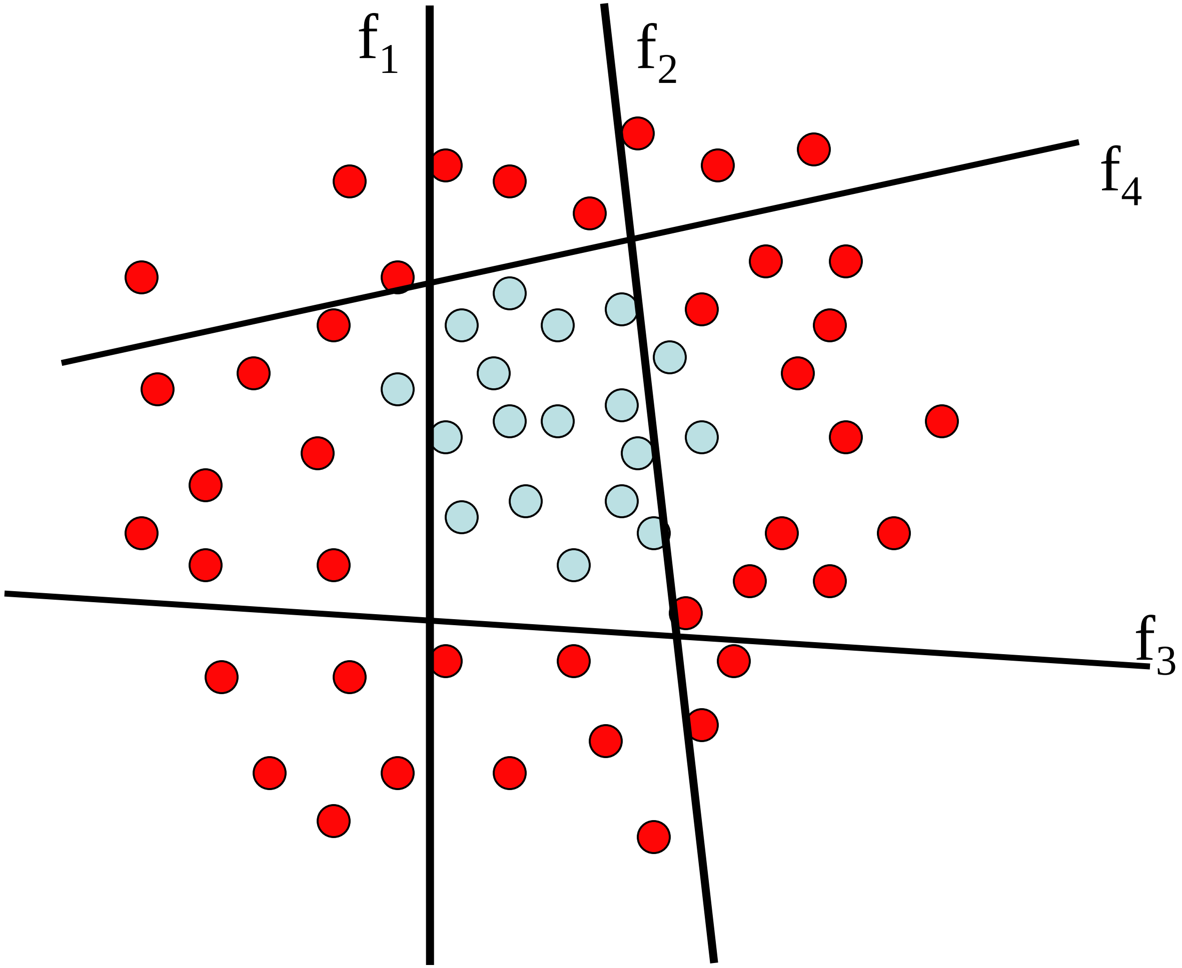
$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

AdaBoost Algorithm

Given: m examples $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$

For t = 1 to T

1. Train learner $\boldsymbol{h_t}$ with min error $\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$
2. Compute the hypothesis weight $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
3. For each example *i* = 1 to m

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

Output

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

The goodness of h_t is calculated over D_t and the bad guesses.

The weight **Ada**pts. The bigger ε_t becomes the smaller α_t becomes.

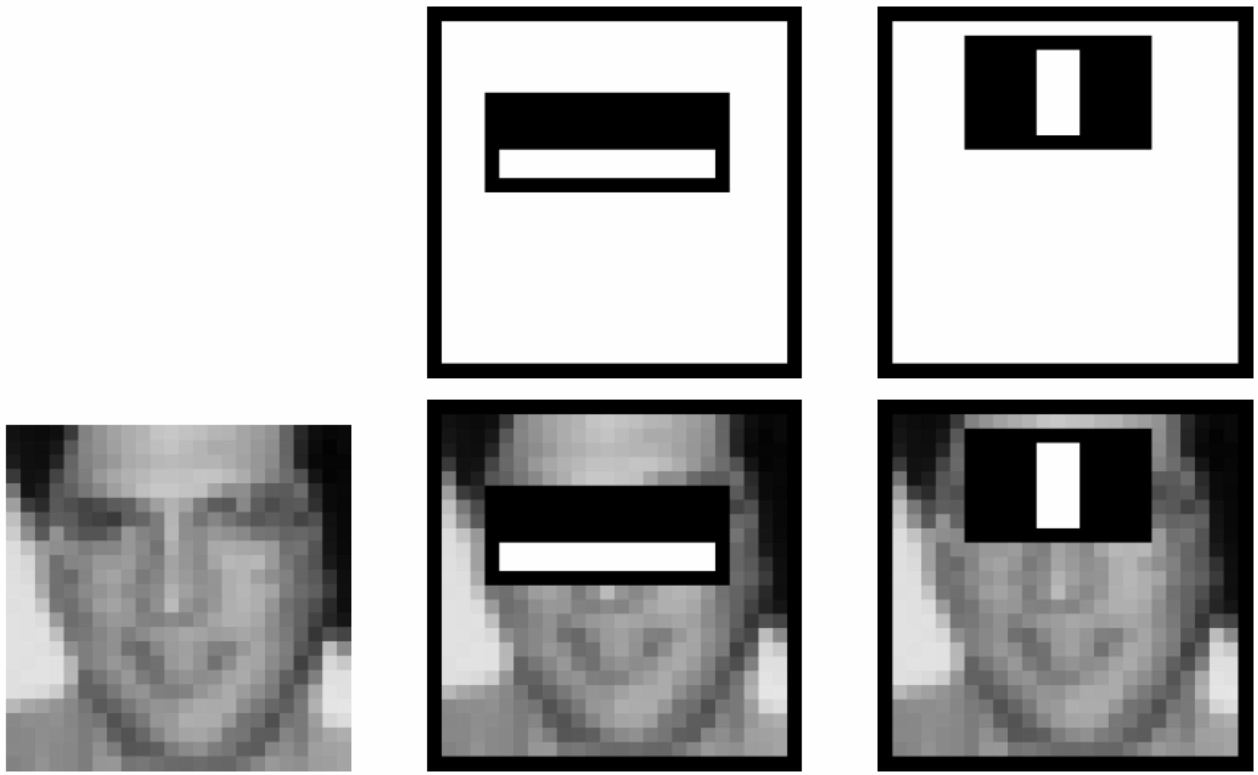
Boost example if incorrectly predicted.

Z_t is a normalization factor.

Linear combination of models.

Boosting for face detection

- First two features selected by boosting:



This feature combination can yield 100% detection rate and 50% false positive rate

Random Forest vs. Boosting

What are the pros and cons?