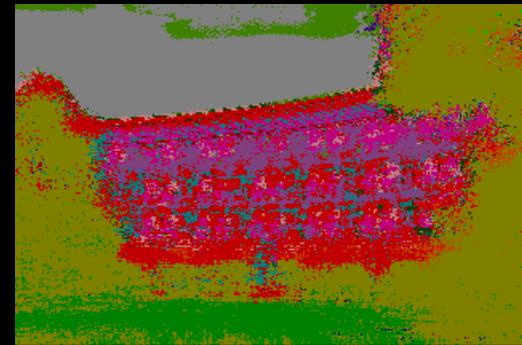
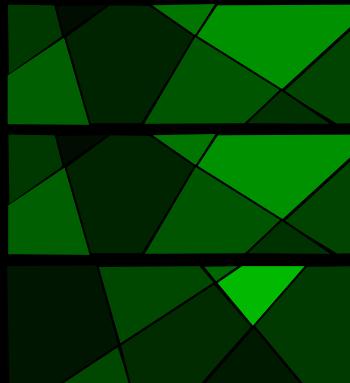


Machine Learning for Computer Vision

19 October, 2012
MVA – ENS Cachan



Lecture 4: Adaboost and Ensemble Classifiers

Iasonas Kokkinos

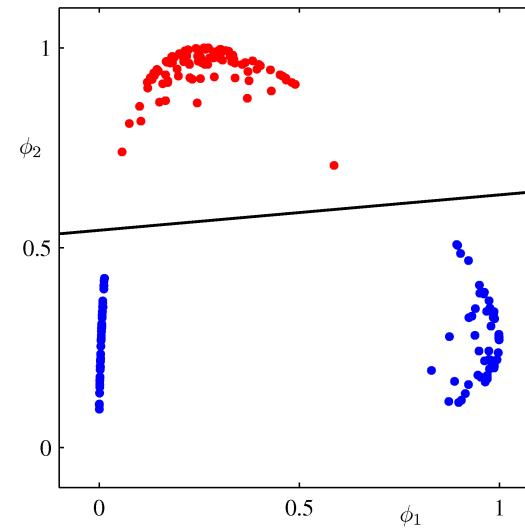
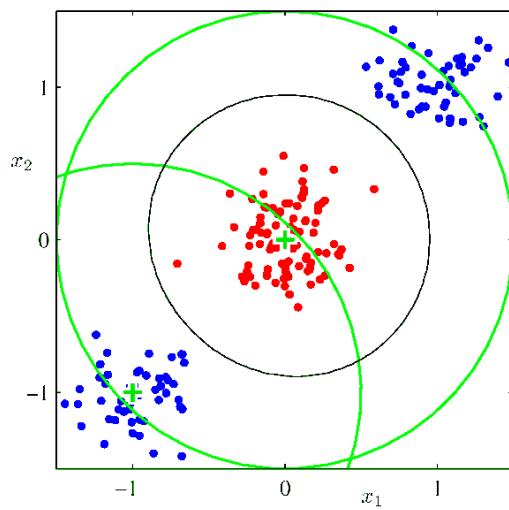
iasonas.kokkinos@ecp.fr

Center for Visual Computing
Ecole Centrale Paris

Galen Group
INRIA-Saclay

Beyond linear boundaries

- Straightforward extension: More features



- Which features?
- How can we control complexity?

Last lecture: Kernel trick

Define Kernel: $K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$

Rewrite everything with kernels: same optimization in the dual!

$$\max_{\mu} \quad \sum_{i=1}^M \mu_i - \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M y^i y^j \mu_i \mu_j K(\mathbf{x}^i, \mathbf{x}^j)$$

$$s.t. \quad 0 \leq \mu_i \leq C$$

$$\sum_{i=1}^M \mu_i y^i = 0$$

$$\text{Classifier: } y = \text{sign}\left(\sum_{i=1}^M \mu_i y^i K(\mathbf{x}^i, \mathbf{x}) + b\right)$$

Complexity of evaluation: $O(\#\text{Support Vectors})$



Lecture outline

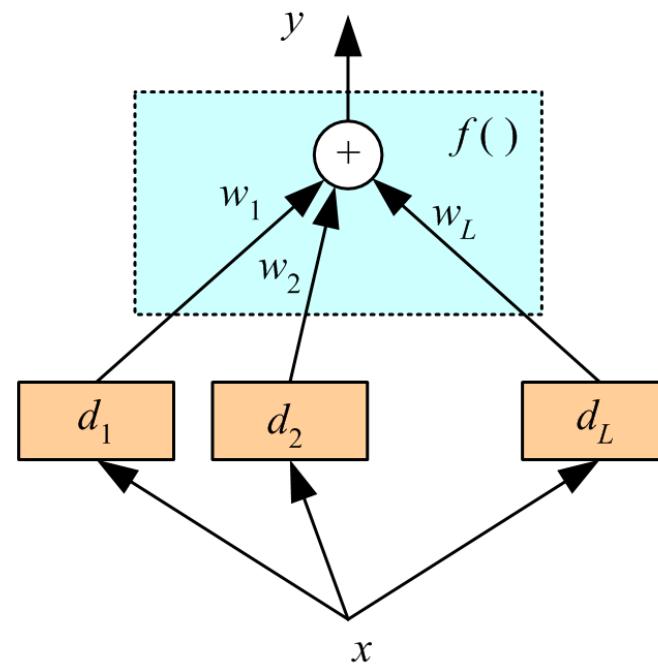
Adaboost

Decision Trees

Random Forests

Voting Methods

- Give up idea of building ‘the’ classifier
- Generate a group of **base-learners** which has higher accuracy when combined
- Main tasks
 - Generating the learners
 - Combining them



Why should this work?

- Committee of M predictors for target output

$$y_{COM}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x})$$

- Output: true value + error

$$y(\mathbf{x}) = h(\mathbf{x}) + \epsilon(\mathbf{x})$$

- Average sum of squares error for m-th expert:

$$\mathbb{E}_{\mathbf{x}} = \left[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2 \right] = \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2]$$

- Average error of individual members: $\mathbb{E}_{AV} = \frac{1}{M} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2]$
- Average error of committee:

$$\mathbb{E}_{COM} = \mathbb{E}_{\mathbf{x}} \left[\left\{ \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x}) - h(\mathbf{x}) \right\}^2 \right] = \mathbb{E}_{\mathbf{x}} \left[\left\{ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right\}^2 \right]$$

- If errors have zero mean and are uncorrelated: $\mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})] = 0$

then $\mathbb{E}_{COM} = \frac{1}{M} \mathbb{E}_{AV}$

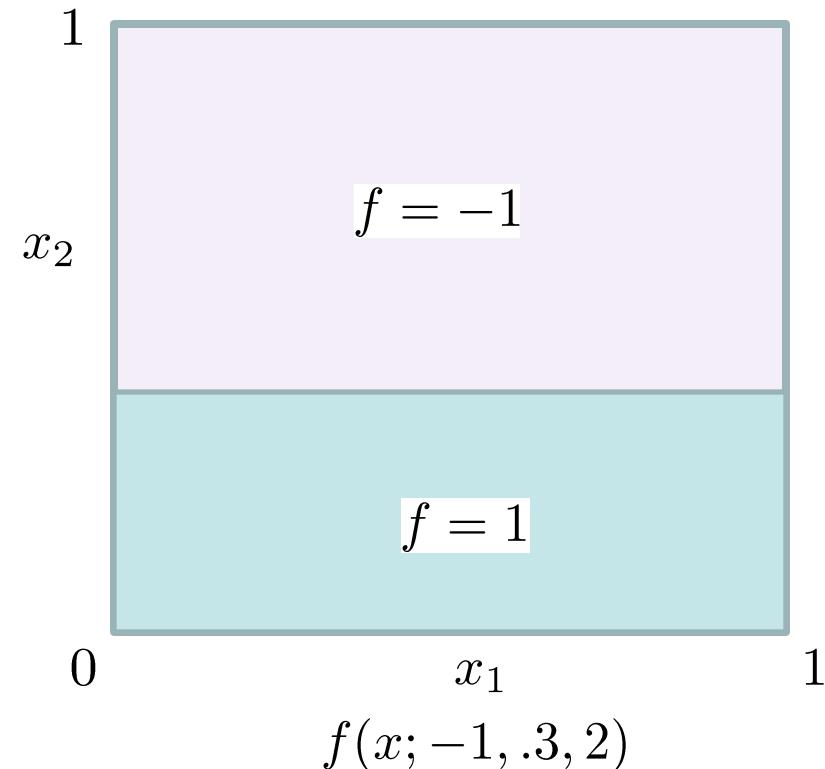
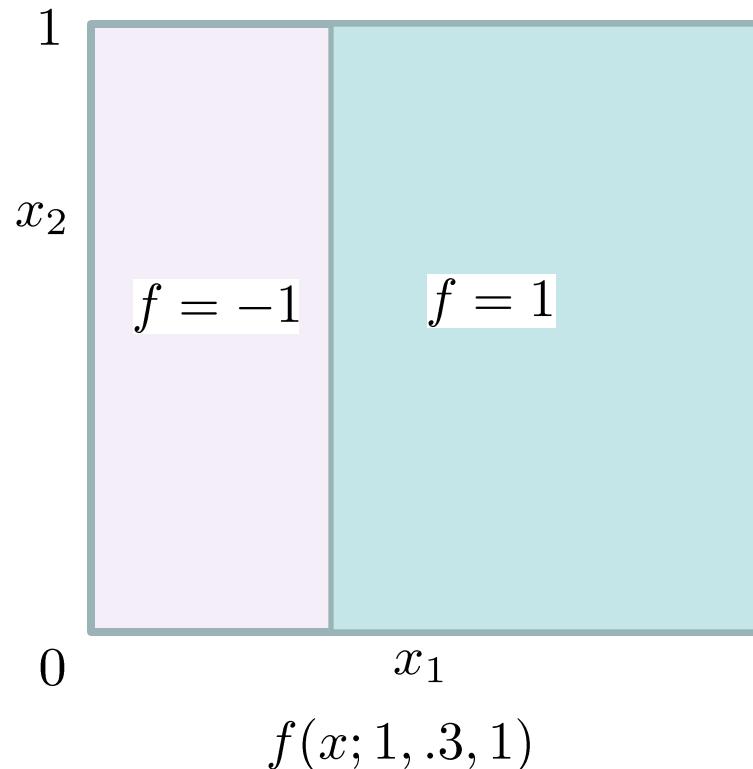
$$\mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x}) \epsilon_j(\mathbf{x})] = 0$$

Boosting Idea

- General algorithm
 - Iterate
 - Pick subset of training data
 - Obtain **weak learner**
 - Easy to train and evaluate
 - Output final classifier by adding weak learner responses (majority voting)

Weak Learner Example

- Decision stump $f(x; s, \theta, i) = 2s[x_i > \theta] - s, \quad s \in \{-1, 1\}$

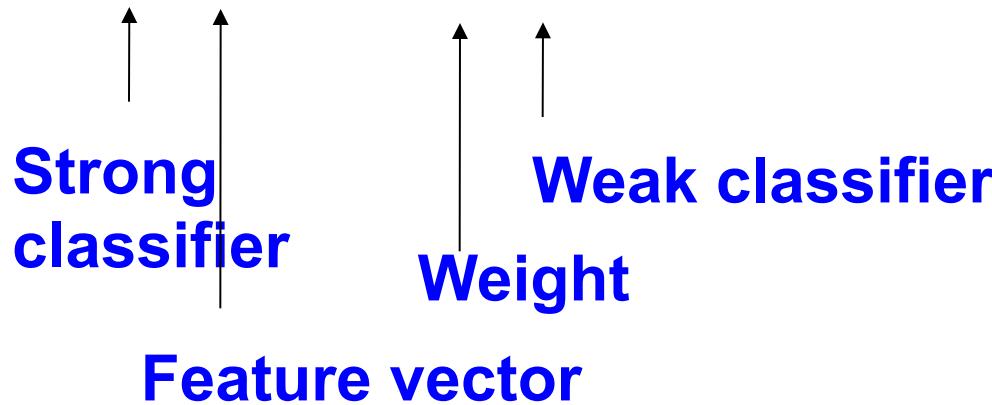


- Extremely easy to train and evaluate

Adaboost

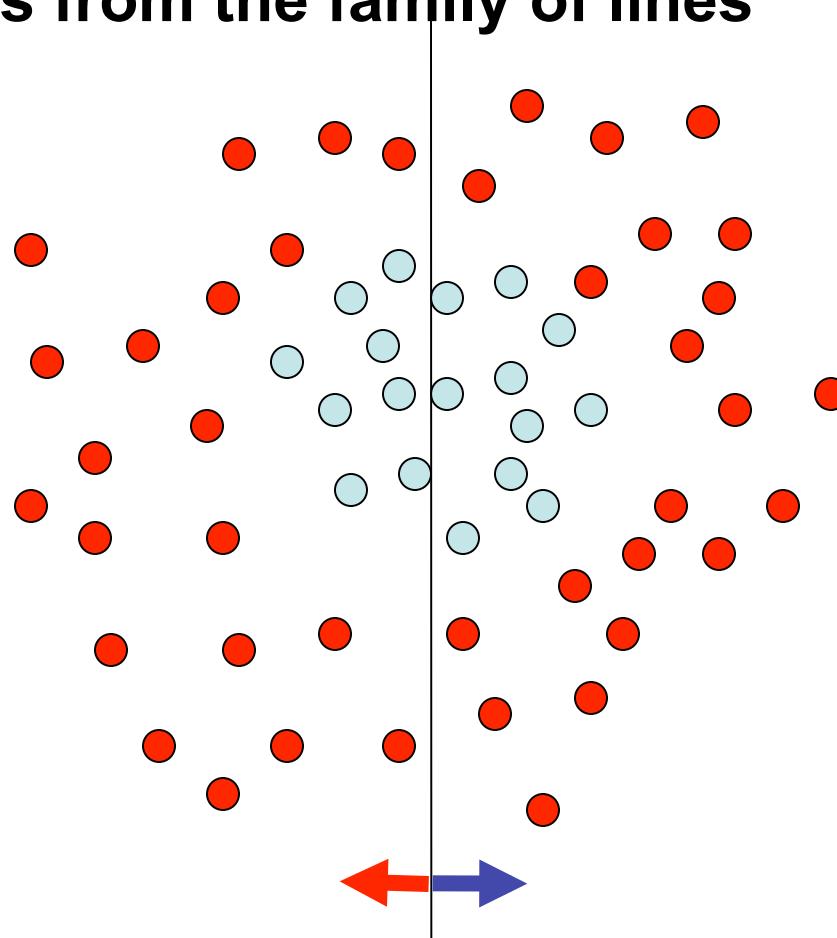
- Adaptive Boosting
 - ‘A decision theoretic generalization of on-line learning and an application to boosting’, Freund and Schapire ’95
 - ~5500 Citations
 - Versatile (Discrete data, arbitrary distributions, multi-class, regression)
 - Very easy to program
 - Excellent results
- Defines a classifier using an additive model (weighted voting):

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



Toy example

Weak learners from the family of lines



Each data point has
a class label:

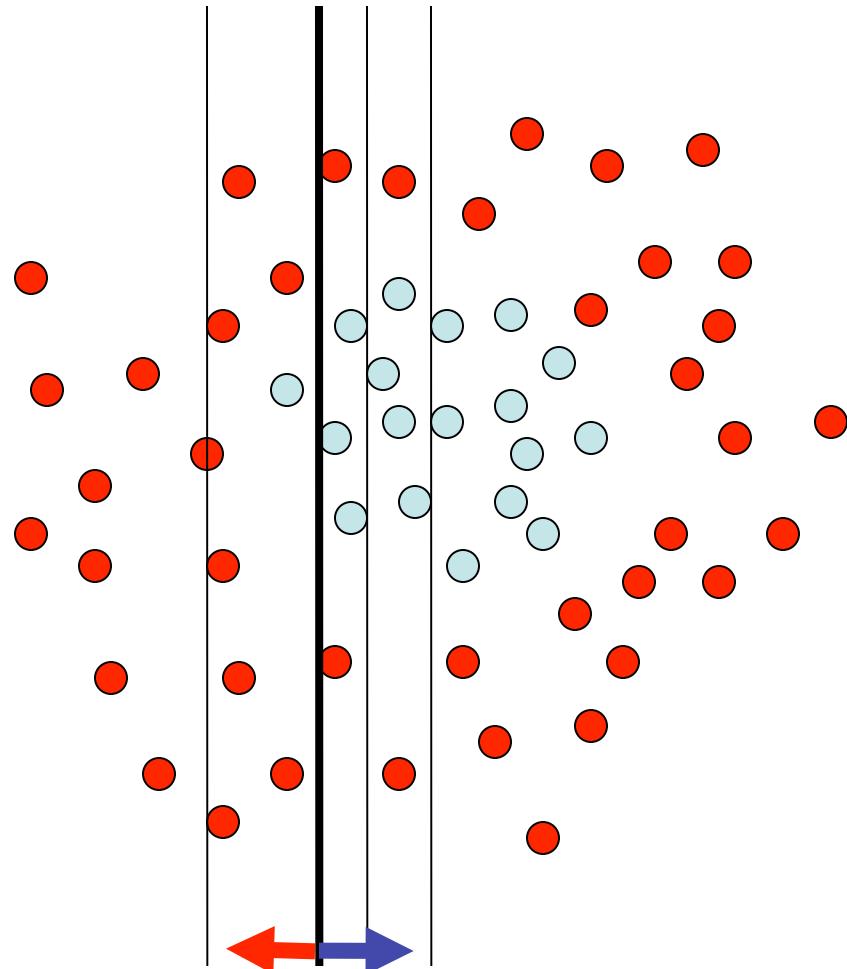
$$y_t = \begin{cases} +1 & (\text{Red}) \\ -1 & (\text{Blue}) \end{cases}$$

and a weight:

$$w_t = 1$$

$h \Rightarrow p(\text{error}) = 0.5$ it is at chance

Toy example



This one seems to be the best

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

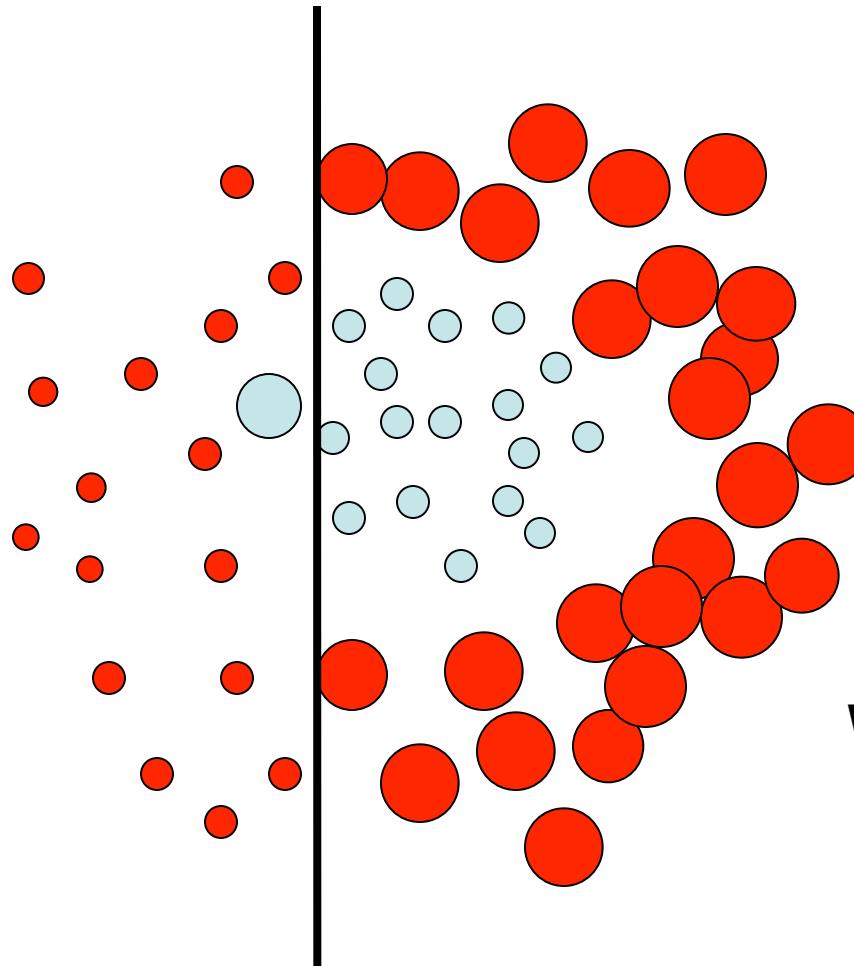
Each data point has
a class label:

$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{light blue}) \end{cases}$$

and a weight:

$$w_t = 1$$

Toy example



Each data point has
a class label:

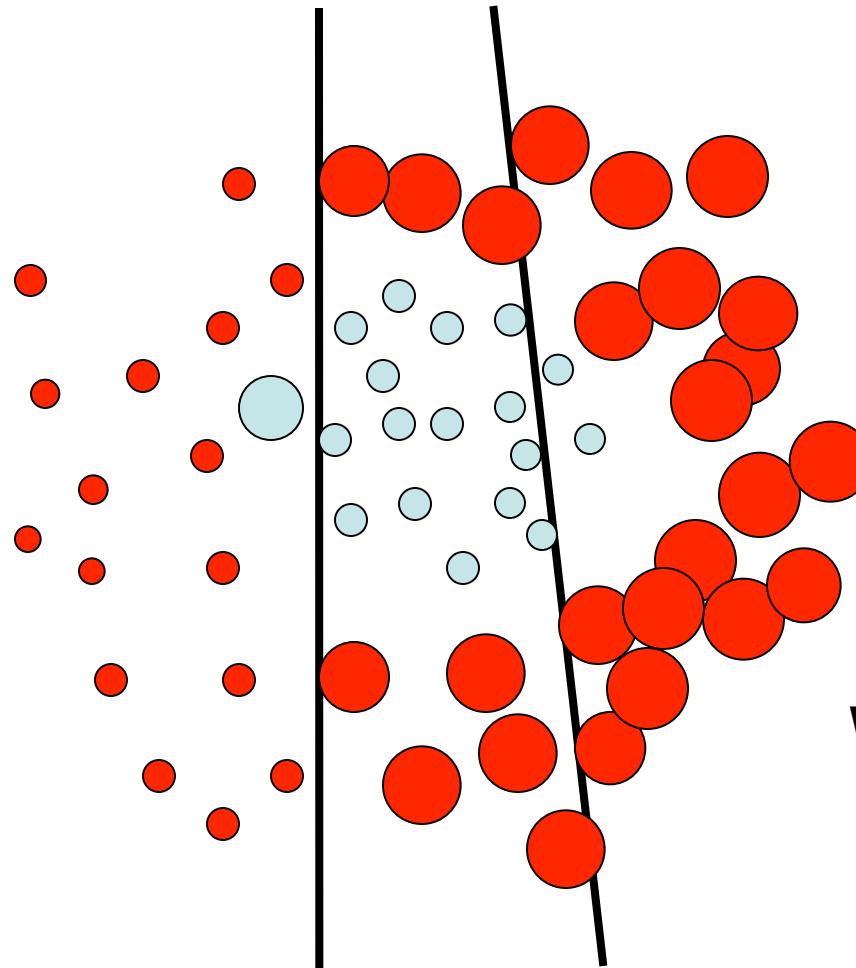
$$y_t = \begin{cases} +1 (\textcolor{red}{\bullet}) \\ -1 (\textcolor{lightblue}{\circ}) \end{cases}$$

We update the weights:

$$\mathbf{w}_t \leftarrow \mathbf{w}_t \exp\{-y_t \mathbf{H}_t\}$$

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

Toy example



Each data point has
a class label:

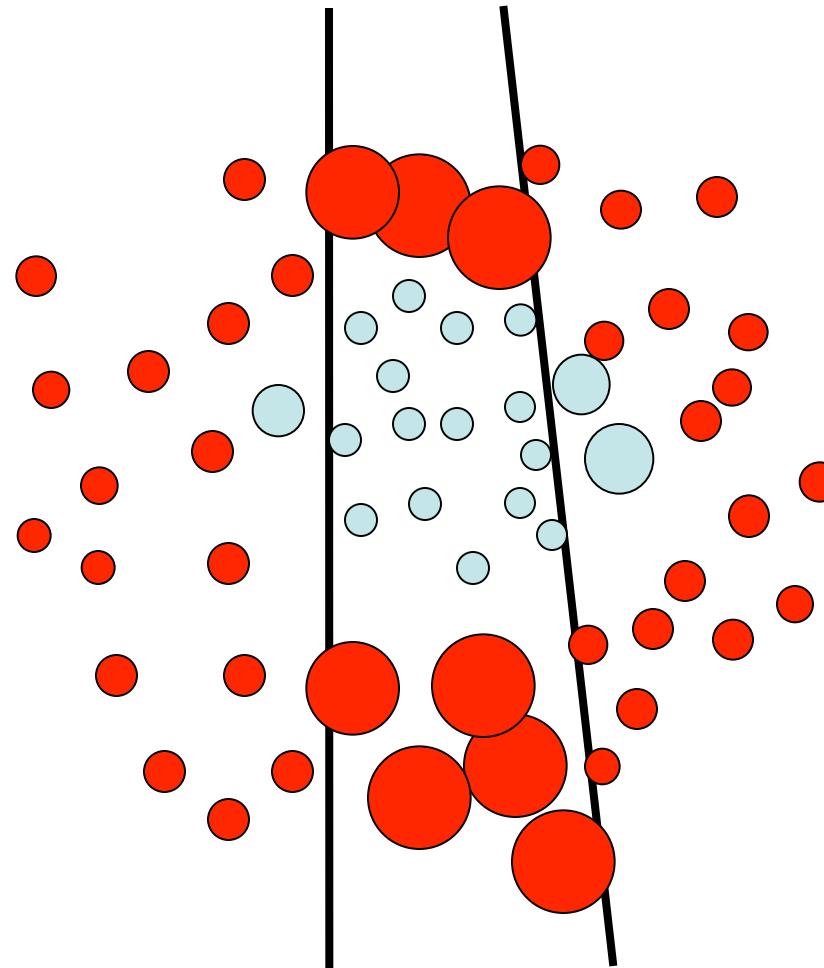
$$y_t = \begin{cases} +1 (\text{red}) \\ -1 (\text{light blue}) \end{cases}$$

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



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

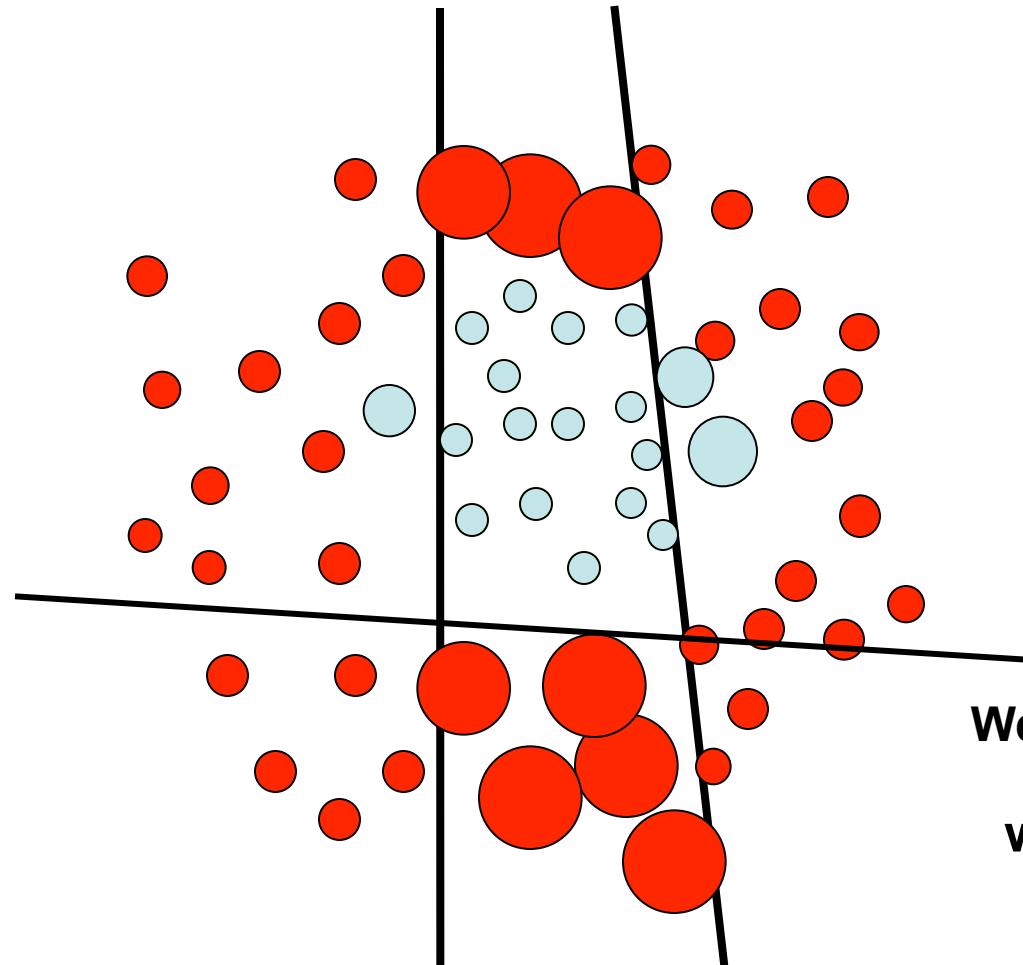
Each data point has a class label:

$$y_t = \begin{cases} +1 (\text{red}) \\ -1 (\text{light blue}) \end{cases}$$

We update the weights:

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

Toy example



Each data point has
a class label:

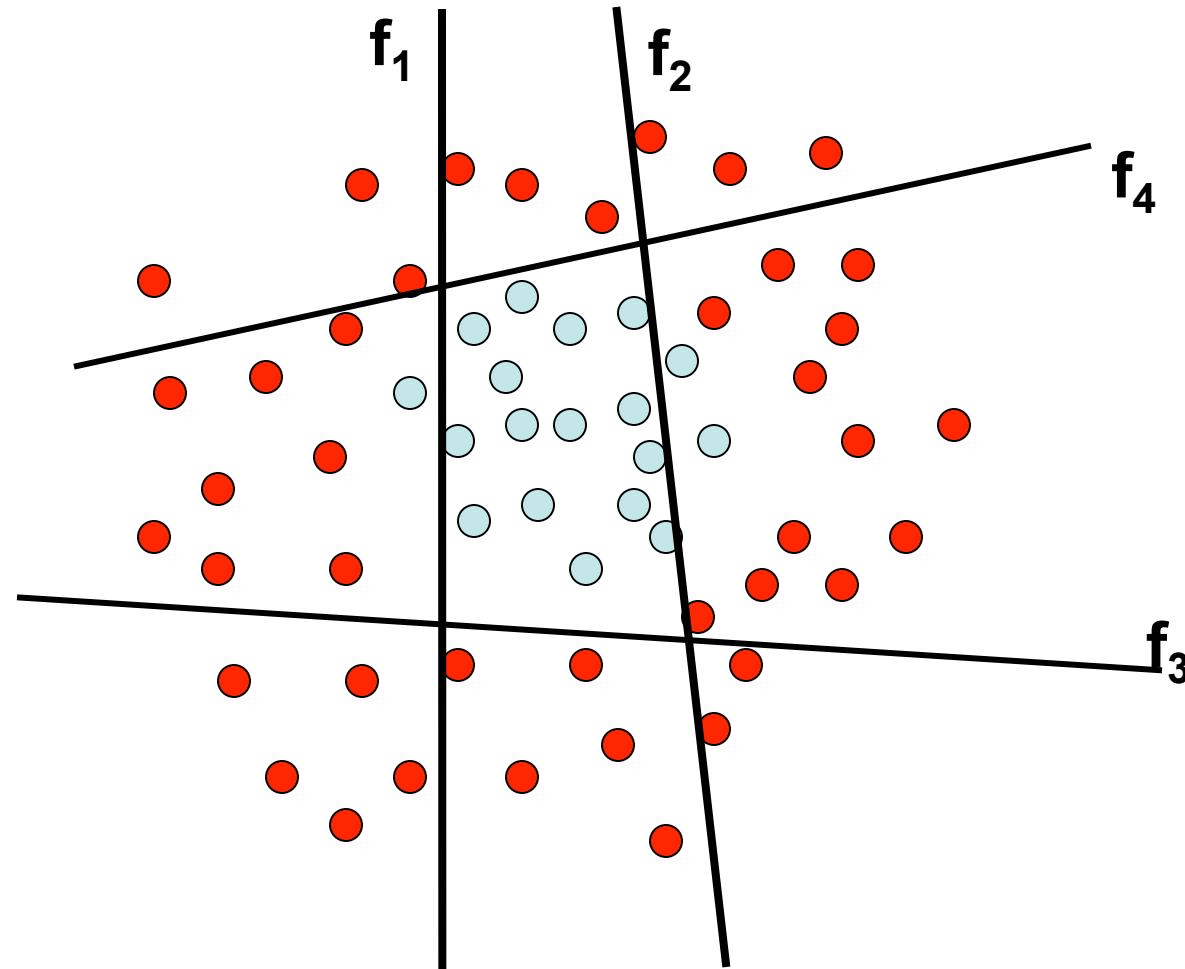
$$y_t = \begin{cases} +1 (\text{red}) \\ -1 (\text{light blue}) \end{cases}$$

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: $(x^i, y^i), x^i \in \mathcal{X}, y^i \in \{-1, 1\}, i = 1, \dots, N$
- Initialize: $D_1(i) = \frac{1}{N}$
- For $t = 1 \dots T$
 - Find classifier $h_t : \mathcal{X} \rightarrow \{-1, 1\}$ with smallest weighted error

$$\epsilon_t = \frac{\sum_{i=1}^N D_t^i [y^i \neq h_t(x^i)]}{\sum_i D_t^i}$$

- Update distribution

$$D_{t+1}^i = \frac{D_t^i}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } y^i = h_t(x^i) \\ \exp(\alpha_t), & \text{if } y^i \neq h_t(x^i) \end{cases}$$

$$a_t = \frac{1}{2} \log \frac{(1-\epsilon_t)}{\epsilon_t} = \frac{D_t^i}{Z_t} \exp(-\alpha_t y^i h_t(x^i))$$

$$Z_t = \sum_i D_t^i \exp(-\alpha_t y^i h_t(x^i))$$

- Final classifier $H(x) = \operatorname{sign} \left(\sum_t a_t h_t(x) \right)$

When and why does this work?

- Introduce ‘edge’: $\epsilon_t = \frac{1}{2} - \underbrace{\gamma_t}_{\text{Edge}}$

- Theorem:

$$\underbrace{\sum_{i=1}^N \frac{1}{N} [y^i \neq H(x^i)]}_{\text{Final Training Error}} \leq \prod_{t=1}^T \left[2\sqrt{\epsilon_t(1-\epsilon_t)} \right]$$

$$= \prod_{t=1}^T \sqrt{1 - 4\gamma_t^2}$$

$$\leq \exp \left(-2 \sum_{t=1}^T \gamma_t^2 \right)$$

- Assumption $\gamma_t \geq \gamma > 0, \quad \forall t$

$$\sum_{i=1}^N \frac{1}{N} [y^i \neq H(x^i)] \leq \exp(-2T\gamma^2)$$

Step I: Relate weights with vote

- Recursion for D

$$\begin{aligned}
 D_{t+1}^i &= \frac{D_t^i}{Z_t} \exp(-\alpha_t y^i h_t(x^i)) \\
 &= \frac{\frac{D_{t-1}^i}{Z_{t-1}} \exp(-\alpha_{t-1} y^i h_{t-1}(x^i))}{Z_t} \exp(-\alpha_t y^i h_T(x^i)) \\
 &= \frac{D_{t-1}}{Z_t Z_{t-1}} \exp \left(- [\alpha_t y^i h_t(x^i) + \alpha_{t-1} y^i h_{t-1}(x^i)] \right) \\
 &= \frac{1}{N} \frac{\exp \left(-y^i \sum_{k=1}^t \alpha_k h_k(x^i) \right)}{\prod_{k=1}^t Z_k}
 \end{aligned}$$
- So

$$D_{T+1}^i = \frac{1}{N} \frac{\exp \left(-y^i f(x^i) \right)}{\prod_{k=1}^T Z_k}$$

Step II: Bound training error in terms of Z

- Lower bound for error

$$\begin{aligned} \sum_{i=1}^N \frac{1}{N} [y^i \neq H(x^i)] &\stackrel{H(x)=\text{sign } f(x)}{=} \frac{1}{N} \sum_i \begin{cases} 1 & y^i f(x^i) \leq 0 \\ 0 & y^i f(x^i) > 0 \end{cases} \\ &\leq \sum_{i=1}^N \frac{1}{N} \exp(-y^i f(x^i)) \\ &= \sum_{i=1}^N D_{T+1}^i \prod_{k=1}^T Z_k \\ &= 1 \prod_{k=1}^T Z_k \end{aligned}$$

Step III: Relate Z with ϵ

- We set out to show
- So far we have shown

$$\sum_{i=1}^N \frac{1}{N} [y^i \neq H(x^i)] \leq \prod_{t=1}^T [2\sqrt{\epsilon_t(1-\epsilon_t)}]$$

$$\sum_{i=1}^N \frac{1}{N} [y^i \neq H(x^i)] \leq \prod_{k=1}^T Z_k$$

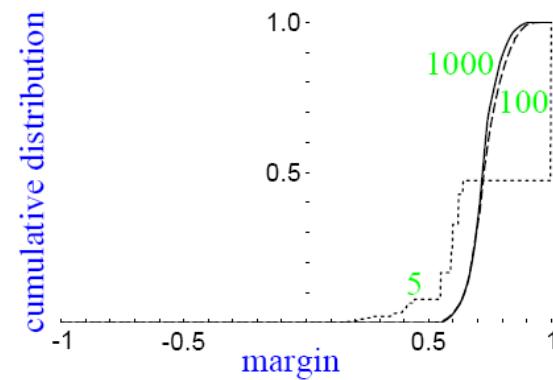
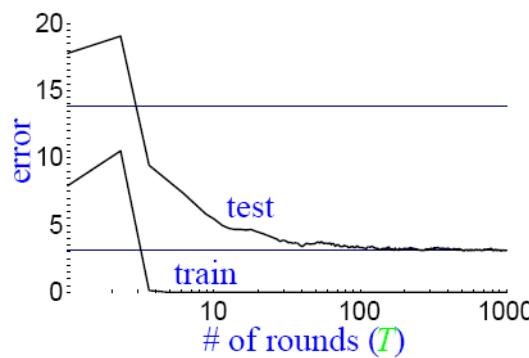
- But

$$\begin{aligned}
 Z_t &= \sum_{i=1}^N D_t^i \exp(-\alpha_t y^i h_t(x^i)) \\
 &= \sum_{i:y^i \neq h_t(x^i)} D_t^i \exp(-\alpha_t y^i h_t(x^i)) + \sum_{i:y^i = h_t(x^i)} D_t^i \exp(-\alpha_t y^i h_t(x^i)) \\
 &= \sum_{i:y^i \neq h_t(x^i)} D_t^i \exp(\alpha_t) + \sum_{i:y^i = h_t(x^i)} D_t^i \exp(-\alpha_t) \\
 &= (\epsilon_t) \exp(\alpha_t) + (1 - \epsilon_t) \exp(-\alpha_t) \\
 \alpha = \frac{1}{2} \log\left(\frac{1-\epsilon}{\epsilon}\right) &\equiv \epsilon_t \frac{\sqrt{1-\epsilon_t}}{\sqrt{\epsilon_t}} + (1 - \epsilon_t) \frac{\sqrt{\epsilon_t}}{\sqrt{1-\epsilon_t}} \\
 &= 2\sqrt{\epsilon_t(1-\epsilon_t)}
 \end{aligned}$$

- Done

Generalization Error for Adaboost

□ Empirical Evidence



□ Theoretical Justification:

- Margin of example i: $y_i f(x_i)$
- Adaboost is increasing the margins of the training set
- Large margin classifiers lead to good generalization (last lecture)

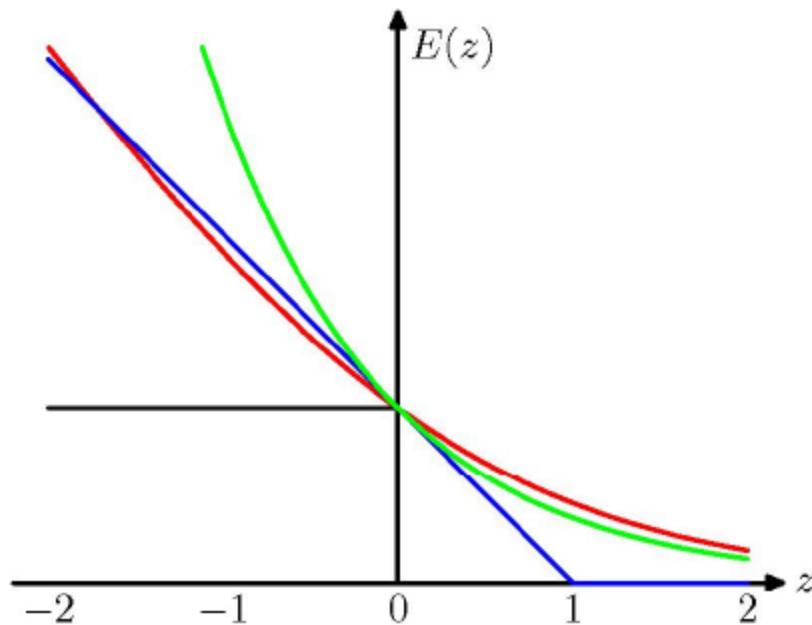
Optimization perspective of Adaboost

- Each round minimizes $\sum_{i=1}^N \exp(-y^i f(x^i))$ by fitting an additive model

$$f'(x) = f(x) + \alpha h(x)$$
- Proof:

$$\begin{aligned} \sum_{i=1}^N \exp(-y^i f'(x^i)) &= \sum_{i=1}^M \exp(-y^i [f(x^i) + \alpha h(x^i)]) \\ &\simeq \sum_{i=1}^N \exp(-y^i f(x^i)) [1 - \alpha y^i h(x^i) + \frac{\alpha^2}{2} (y^i)^2 h^2(x^i)] \\ &= \sum_{i=1}^N \exp(-y^i f(x^i)) [1 - \alpha y^i h(x^i) + \frac{\alpha^2}{2}] \\ &= Z \sum_{i=1}^N \underbrace{\frac{\exp(-y^i f(x^i))}{Z}}_{D^i} [1 - \alpha y^i h(x^i) + \frac{\alpha^2}{2}] \end{aligned}$$
- Choice of h : minimize $\epsilon = \sum D^i y^i h(x^i)$
- Choice of a : minimize $\sum_{i=1}^N D^i \exp(-\alpha y^i h(x^i))$

Loss functions



- $z: y^*f(x)$
- Ideal misclassification cost $H(-z)$ (# training errors)

Exponential Error

$$\exp(-z)$$

(Adaboost)

Cross Entropy error

$$\ln(1 + \exp(-z))$$

(Logistic regression)

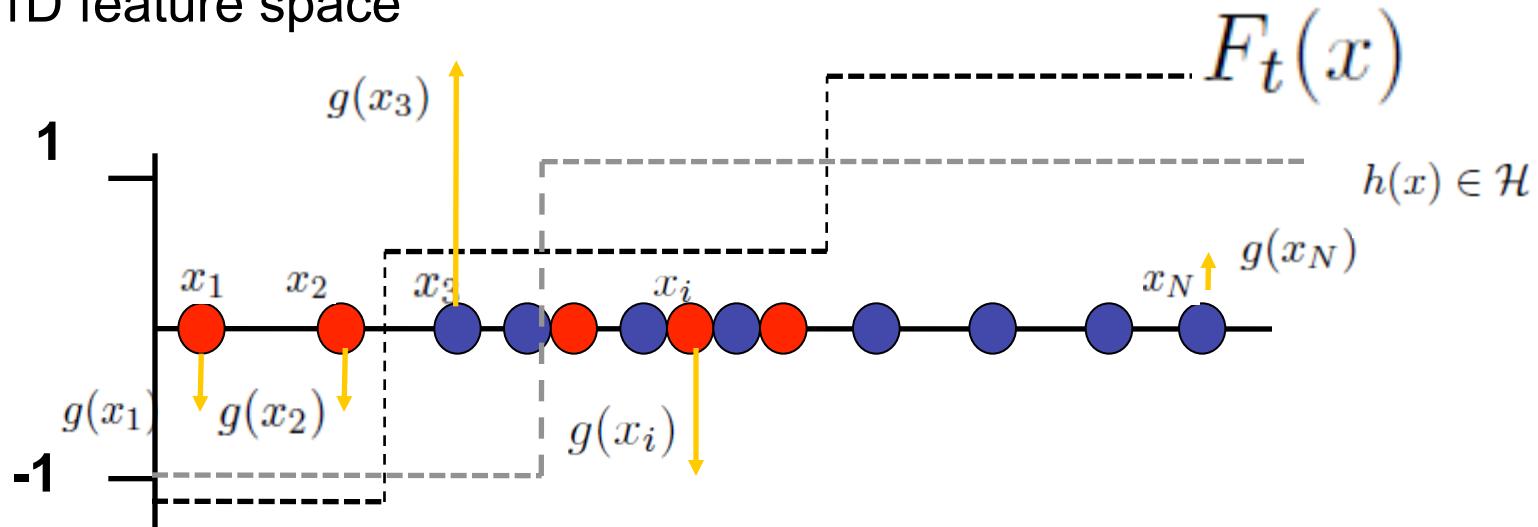
Hinge loss

$$\max(0, 1-z)$$

(SVMs) \

Anyboost- I

1D feature space



Anyboost -II

Additive form: $F_t(x) = a_1 f_1(x) + \dots + a_t f_t(x)$ $a_t \in R, f_t \in \mathcal{H}$

At each round, add optimal pair $(a_{t+1}, f_{t+1}(x))$

See training cost as function of $\mathbf{F} = (F_t(x^1), \dots, F_t(x^N))^T$

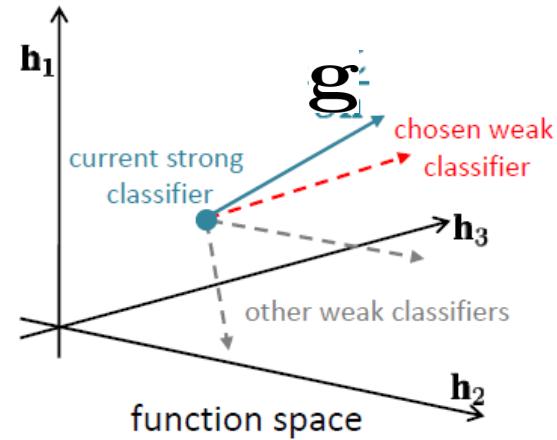
Steepest descent direction:

$$\mathbf{g} = \left(-\frac{\partial L}{\partial \mathbf{F}_1}, \dots, -\frac{\partial L}{\partial \mathbf{F}_N} \right)^T$$

Find $f \in \mathcal{H}$ ‘closest’ to \mathbf{g}

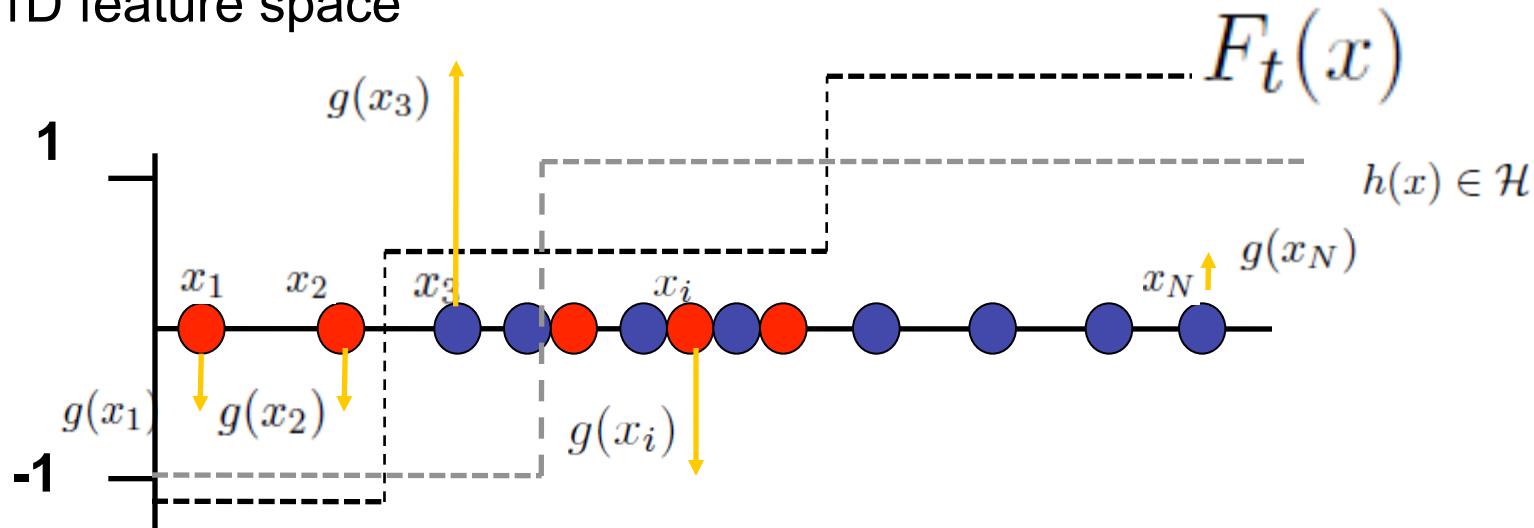
$$f_{t+1} = \operatorname{argmax}_{f \in \mathcal{H}} \mathbf{f}^T \mathbf{g}$$

$$\mathbf{f} = (f(x^1), \dots, f(x^N))^T$$



Anyboost- III

1D feature space



Adaboost: Anyboost for exponential loss

$$L(S, F_t) = \sum_{i=1}^N \exp(-y^i F_t(x^i)) \quad \mathbf{g}_i = y^i \exp(-y^i F_t(x^i))$$

$y^i \in \{-1, 1\}$

sign	weight
------	--------

Cost function for training

Training set $S = \{(x^i, y^i)\} \quad i = 1, \dots, N \quad x^i \in \mathcal{X}, \quad y^i \in \{0, 1\}$

Classifier $F : \mathcal{X} \rightarrow \{0, 1\}$

Loss $L(S, F)$

additive $L(\{(x^i, y^i)\}, F) = \sum_{i=1}^N l(F(x^i), y^i)$

non-additive: F-measure, Area Under Curve (AUC),...

- potentially better suited for the problem
- but also potentially non-convex (local optimality)

T. Joachims, 'A Support Vector Method for Multivariate Performance Measures', ICML, 2005

M. Jansche, 'Maximum Expected F-Measure Training Of Logistic Regression Models', EMNLP, 2005

M. Ranjbar, G. Mori and Y. Wang 'Optimizing Complex Loss Functions in Structured Prediction' ECCV, 2010

I. Kokkinos, Boundary Detection using F-Measure-, Filter- and Feature boost, ECCV, 2010

F-measure

Goal: deal with unbalanced datasets (many negative)

no reward for true negative decisions

$$\text{Predicted label } \hat{y}^i = \left[F(x^i) > \frac{1}{2} \right]$$

$$t = \sum_{i=1}^N [y^i = 1][\hat{y}^i = 1] \quad m = \sum_{i=1}^N [y^i = 1][\hat{y}^i = 0] \quad f = \sum_{i=1}^N [y^i = 0][\hat{y}^i = 1]$$

true positives

misses

false alarms

$$p = \frac{\#\text{true positives}}{\#\text{detected}} = \frac{t}{t+f} \qquad r = \frac{\#\text{true positives}}{\#\text{true}} = \frac{t}{t+m}$$

precision

recall

F-measure: geometric mean of **precision** and **recall**

$$\mathcal{F} = \frac{2}{\frac{1}{p} + \frac{1}{r}} = \frac{pr}{p+r} = 2 \frac{t}{t + \frac{1}{2}(f+m)}$$

F-measure approximation

predicted label

$$\hat{y}^i = \left[F(x^i) > \frac{1}{2} \right]$$

differentiable approximation

$$\tilde{y}^i = \frac{1}{1 + \exp(-F(x^i))} = \sigma(F(x^i))$$

approximate F-measure

$$\tilde{\mathcal{F}} = \frac{\tilde{t}}{\tilde{t} + \frac{1}{2}(\tilde{f} + \tilde{m})}$$

$$\tilde{t} = \sum_{i=1}^N [y^i = 1] \tilde{y}^i \quad \tilde{m} = \sum_{i=1}^N [y^i = 1] (1 - \tilde{y}^i) \quad \tilde{f} = \sum_{i=1}^N [y^i = 0] \tilde{y}^i$$

F-measure optimization via Anyboost

Previous iteration $F_t(x) = a_1 f_1(x) + \dots + a_t f_t(x)$

Loss $L(\{x^i, y^i\}, F) = -\tilde{\mathcal{F}}$

function of responses $\mathbf{F} = (F_t(x^1), \dots, F_t(x^N))^T$

Anyboost $f_{t+1} = \operatorname{argmax}_{f \in \mathcal{H}} \mathbf{f}^T \mathbf{g}$

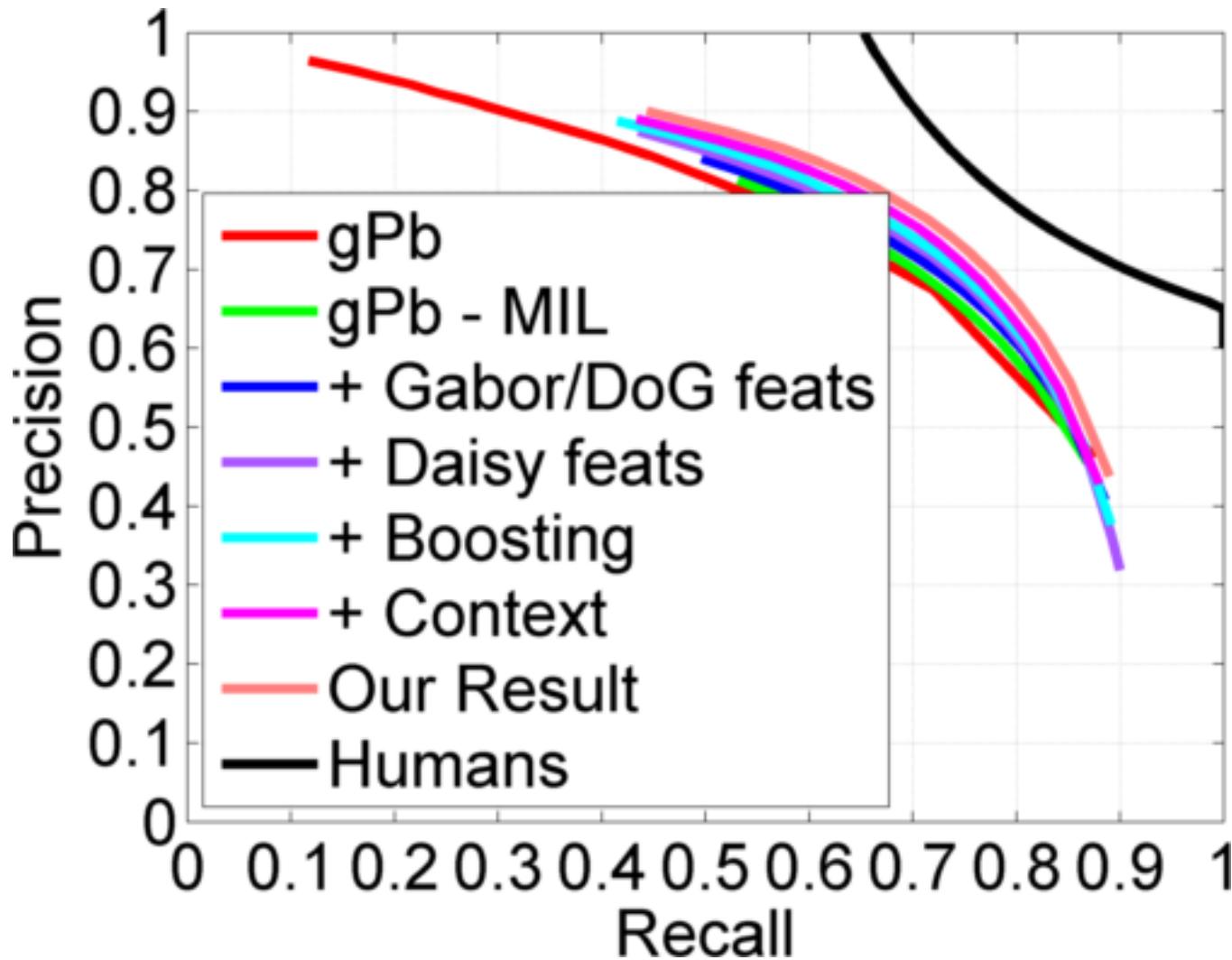
$$\mathbf{f} = (f(x^1), \dots, f(x^N))^T$$

$$\mathbf{g} = \left(-\frac{\partial L}{\partial \mathbf{F}_1}, \dots, -\frac{\partial L}{\partial \mathbf{F}_N} \right)^T = \left(\frac{\partial \tilde{\mathcal{F}}}{\partial \mathbf{F}_1}, \dots, \frac{\partial \tilde{\mathcal{F}}}{\partial \mathbf{F}_N} \right)^T$$

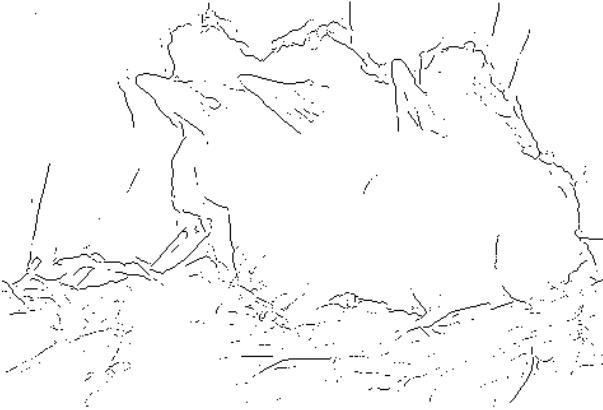
$$\mathbf{g}_i = \left[H[y^i = 1] - \frac{1}{2} H^2 \tilde{t} \right] \sigma'_{y_i}(\mathbf{F}_i)$$

$$H^{-1} = \sum_{i=1}^N [y^i = 1] + \frac{1}{2} (\tilde{t} + \tilde{f}) \quad \sigma_{y^i}(\mathbf{F}_i) = \frac{1}{1 + \exp(-y^i F(x^i))}$$

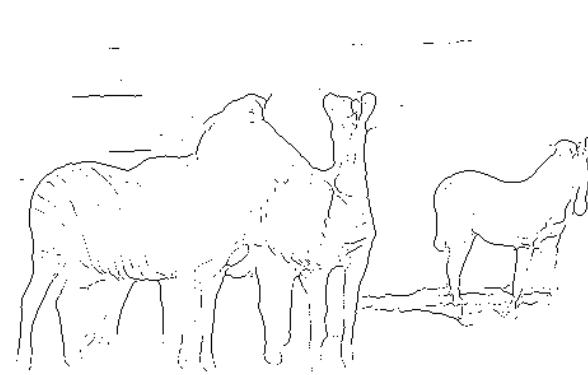
F-measure boosting for boundary detection



Comparisons with gPb @ optimal threshold



Comparisons with gPb @ optimal threshold



Comparisons with gPb @ optimal threshold



Using Adaboost to compute posteriors

- Consider ‘Empirical distribution’ of data

$$\begin{aligned} E_{X,Y} \exp(-yf(x)) &= \int_x \sum_y \exp(-yf(x)) P(x,y) dx dy \\ &= \int_x \sum_y \exp(-yf(x)) P(y|x) dy P(x) dx \\ &= \int_x [P(y=1|x) \exp(-f(x)) + P(y=-1|x) \exp(f(x))] dy dx \end{aligned}$$

- Optimize w.r.t $f(x)$:
$$f(x) = \frac{1}{2} \log \frac{P(y=1|x)}{P(y=-1|x)}$$
- We can therefore use $f(x)$ for a probabilistic classifier

Side-by-side

Logistic regression

- Linear

$$F_a(x) = a_1x_1 + \dots + a_Kx_K$$

- Summands: features

 - fixed

- a : Newton-Raphson

- Cost: minus label log likelihood

Anyboost

- Additive

$$F_t(x) = a_1f_1(x) + \dots + a_tf_t(x)$$

- Summands: weak learners

 - added ‘on the fly’

- a : Coordinate descent

- Cost: exponential loss (AdaBoost)

Connections: M. Collins, R. Schapire, Y. Singer ‘Logistic Regression, AdaBoost and Bregman Distances’ COLT (2000)

Application to Vision

- Viola & Jones, 'Rapid Object Detection using a Boosted Cascade of Simple Features', CVPR 01
 - First reliable, real-time face detection system
 - Used in commercial products, e.g. digital cameras
 - 4500 Citations
- Sample detections (back in 2001)

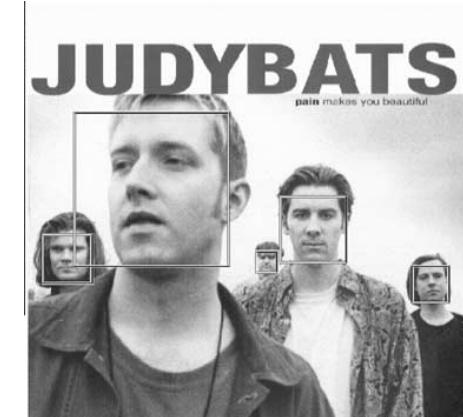
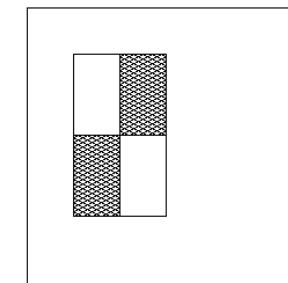
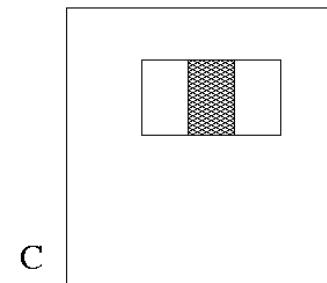
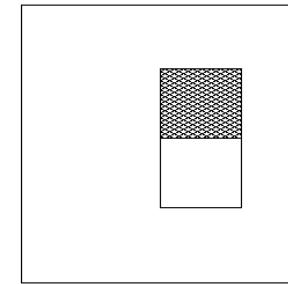
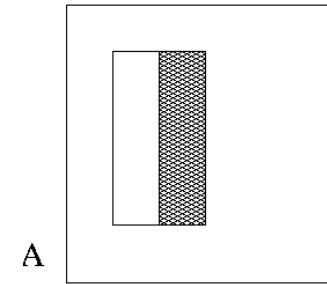


Image Features (Weak learners)

“Rectangle filters”



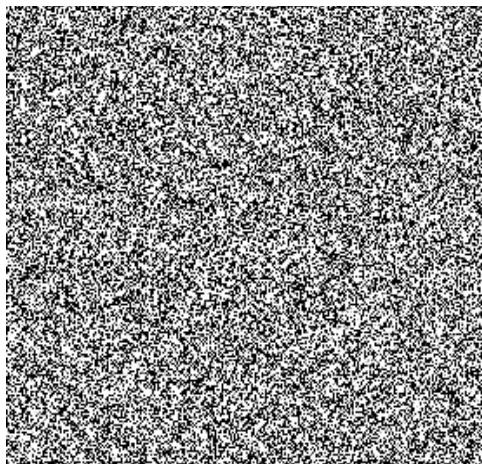
$$\text{Value} = \sum (\text{pixels in white area}) - \sum (\text{pixels in black area})$$

Decision stump: Threshold difference

Why these features?

Extremely fast to compute (4 pixel operations per box)

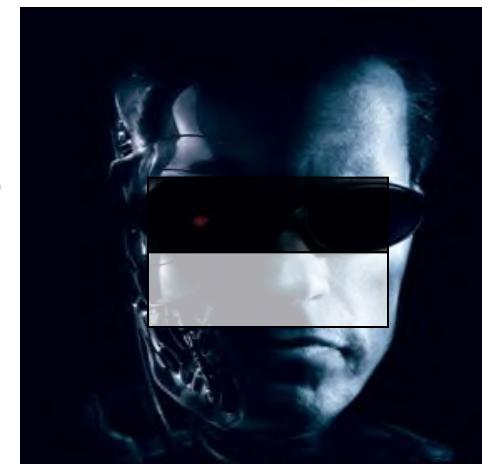
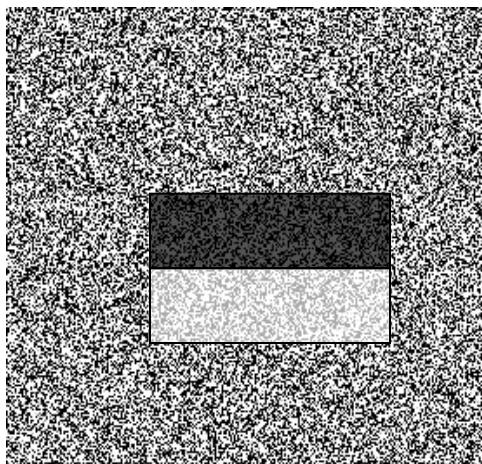
Example



Source

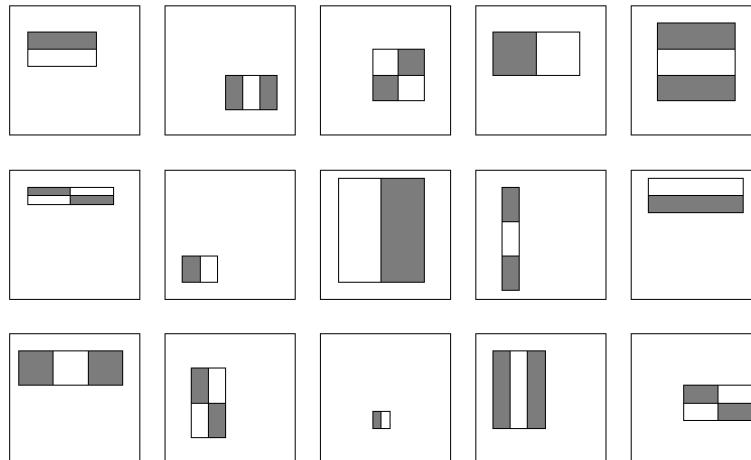


Result



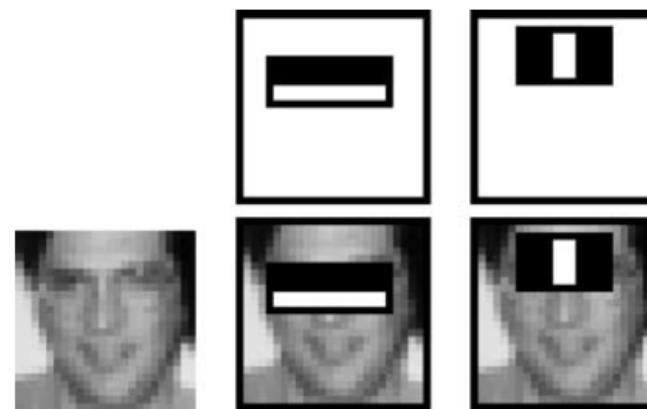
Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~180,000!



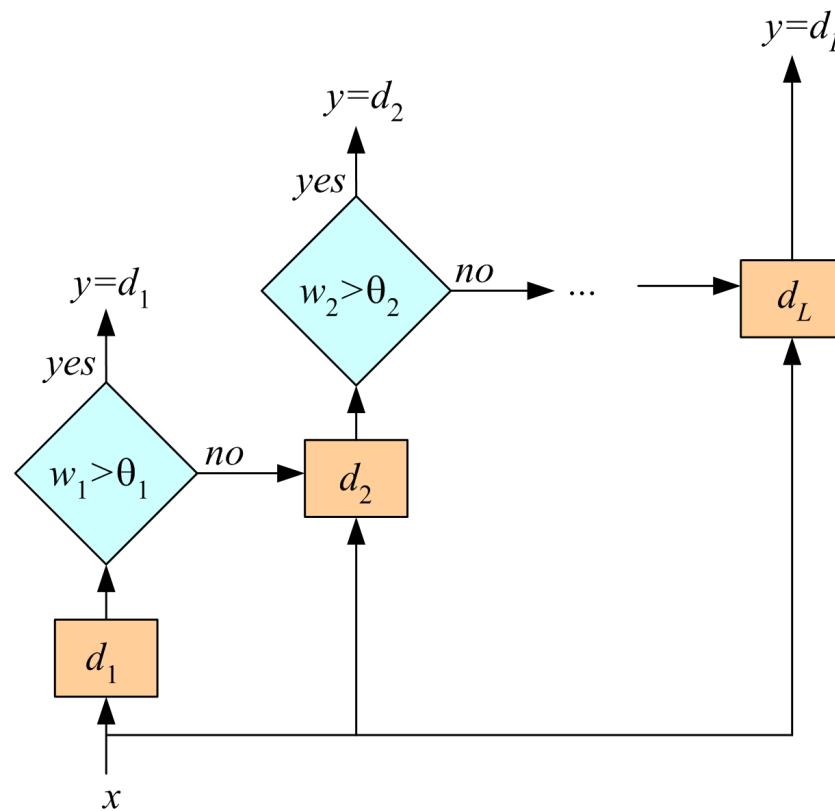
- Chosen automatically by Adaboost

1st WL 2nd WL



Viola & Jones: Cascade

- Reject large portions of the image based on thresholding first few weak learners



Example: screen detection

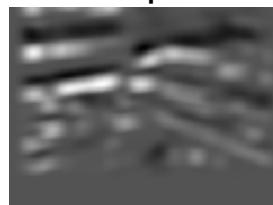
Feature
output



Example: screen detection



Feature
output



Thresholded
output



Example: screen detection



Feature
output



Thresholded
output



Strong classifier
at iteration 1



Example: screen detection



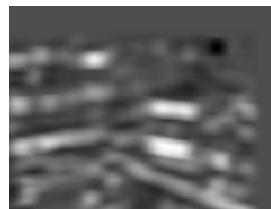
Feature
output



Thresholded
output



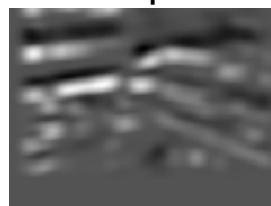
Strong
classifier



Example: screen detection



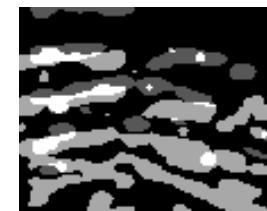
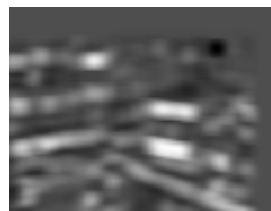
Feature
output



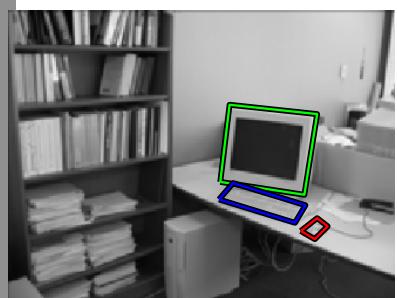
Thresholded
output



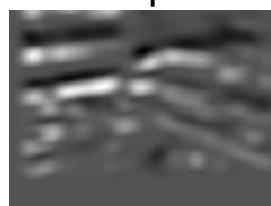
Strong
classifier



Example: screen detection



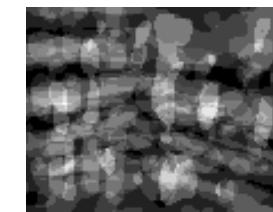
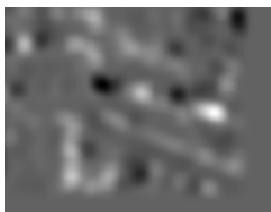
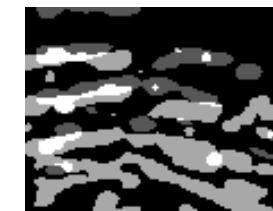
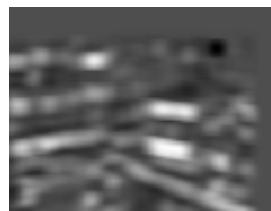
Feature
output



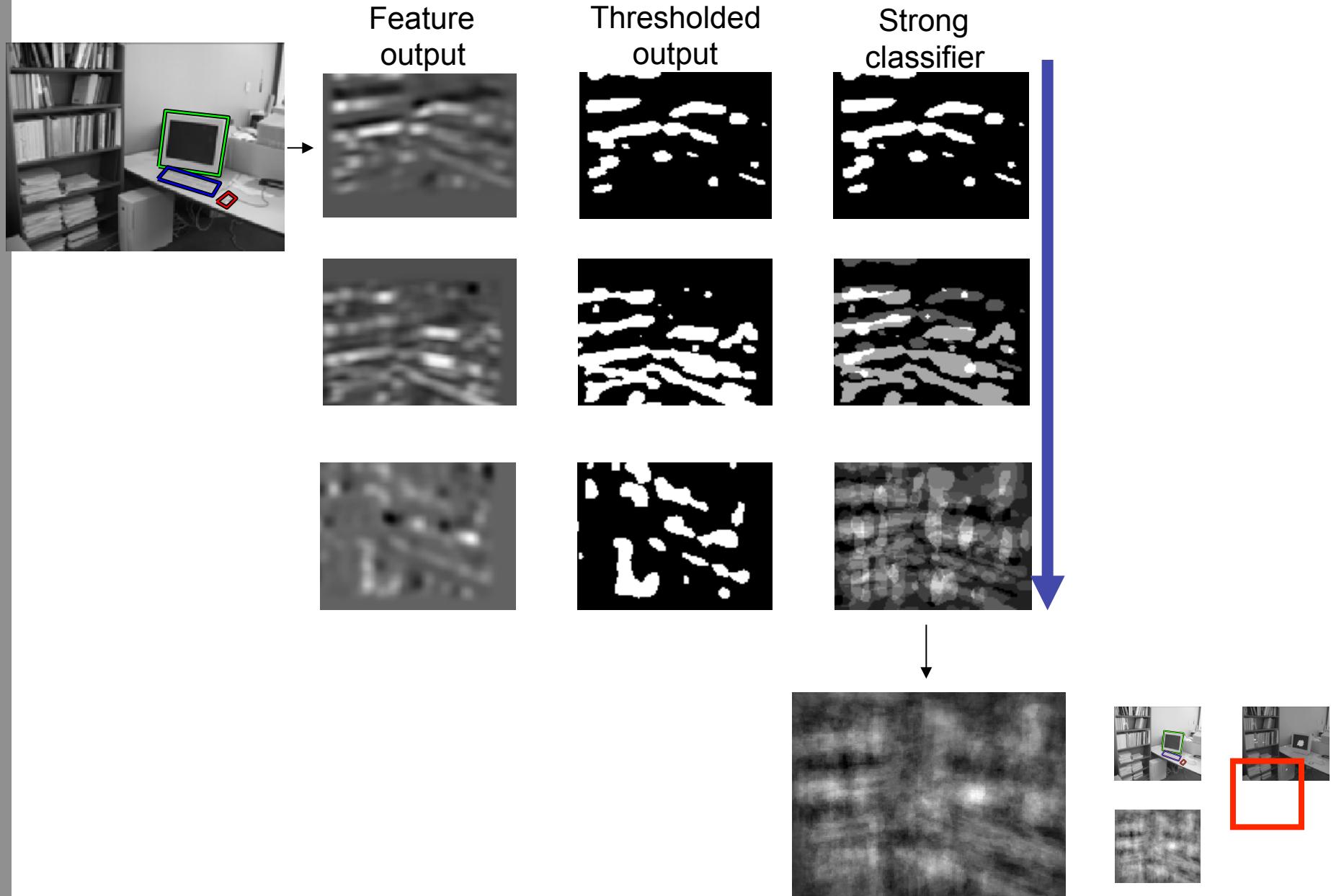
Thresholded
output



Strong
classifier



Example: screen detection





Lecture outline

Recap

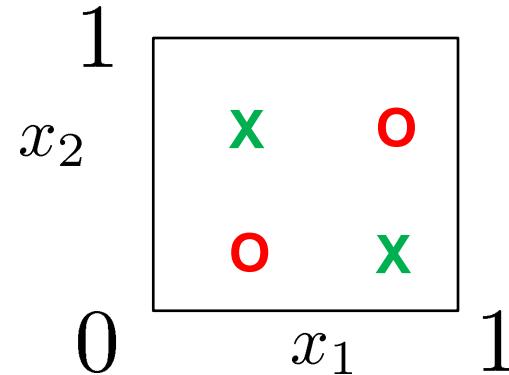
Adaboost

Decision Trees

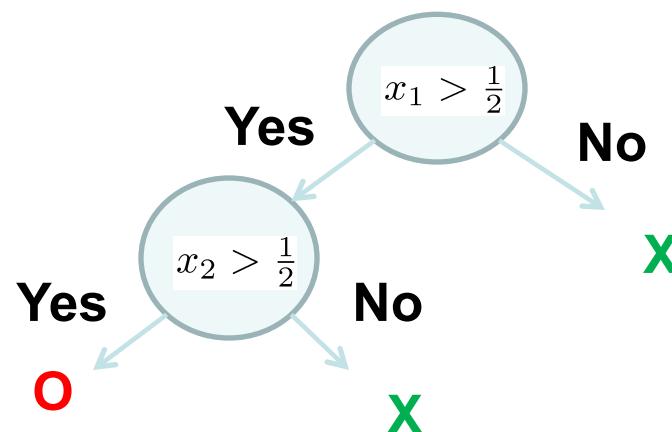
Random Forests & Ferns

Will AdaBoost always work?

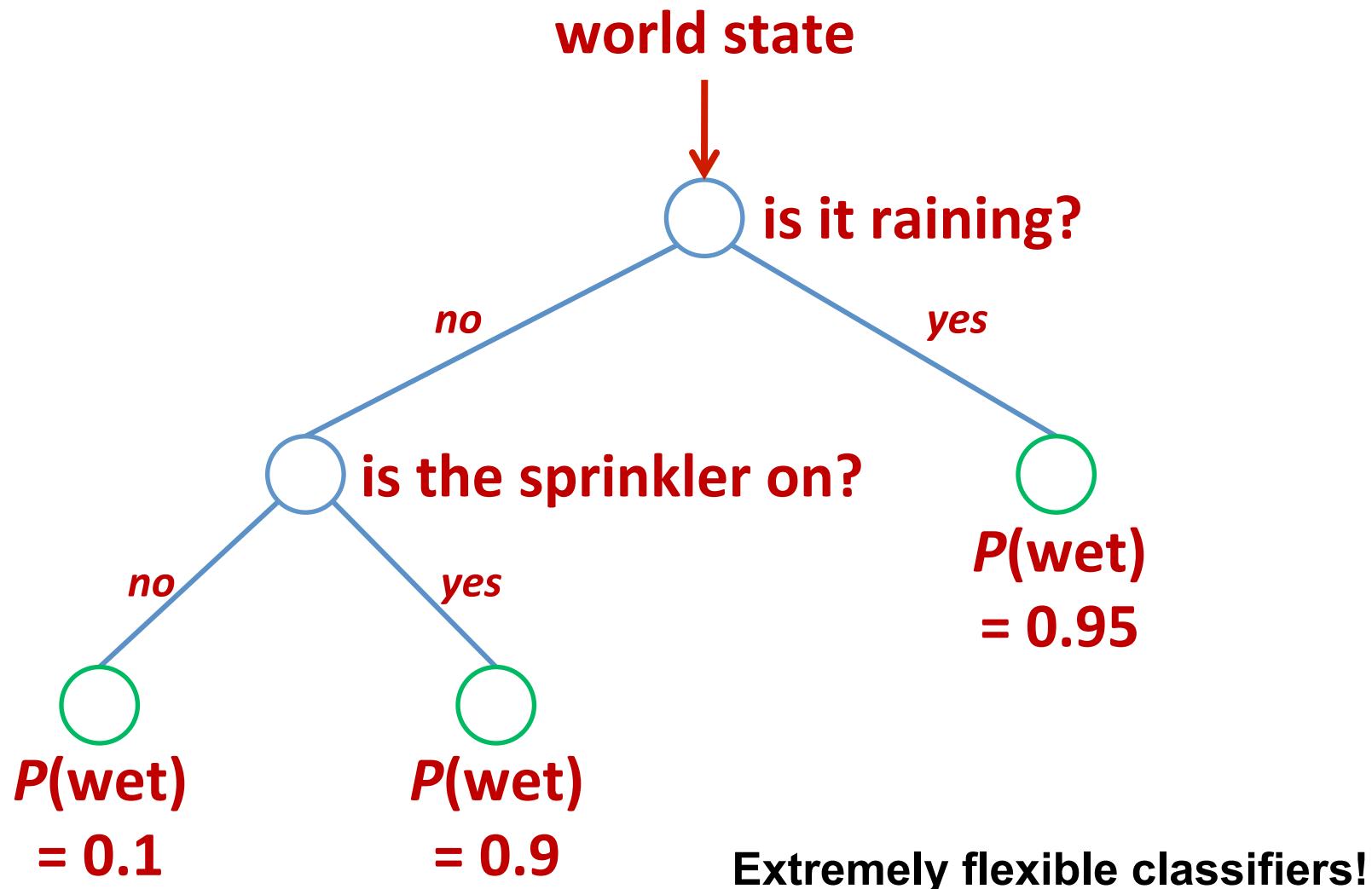
- No decision stump can separate these data with error less than 1/2



- Use classification trees (error 1/4)

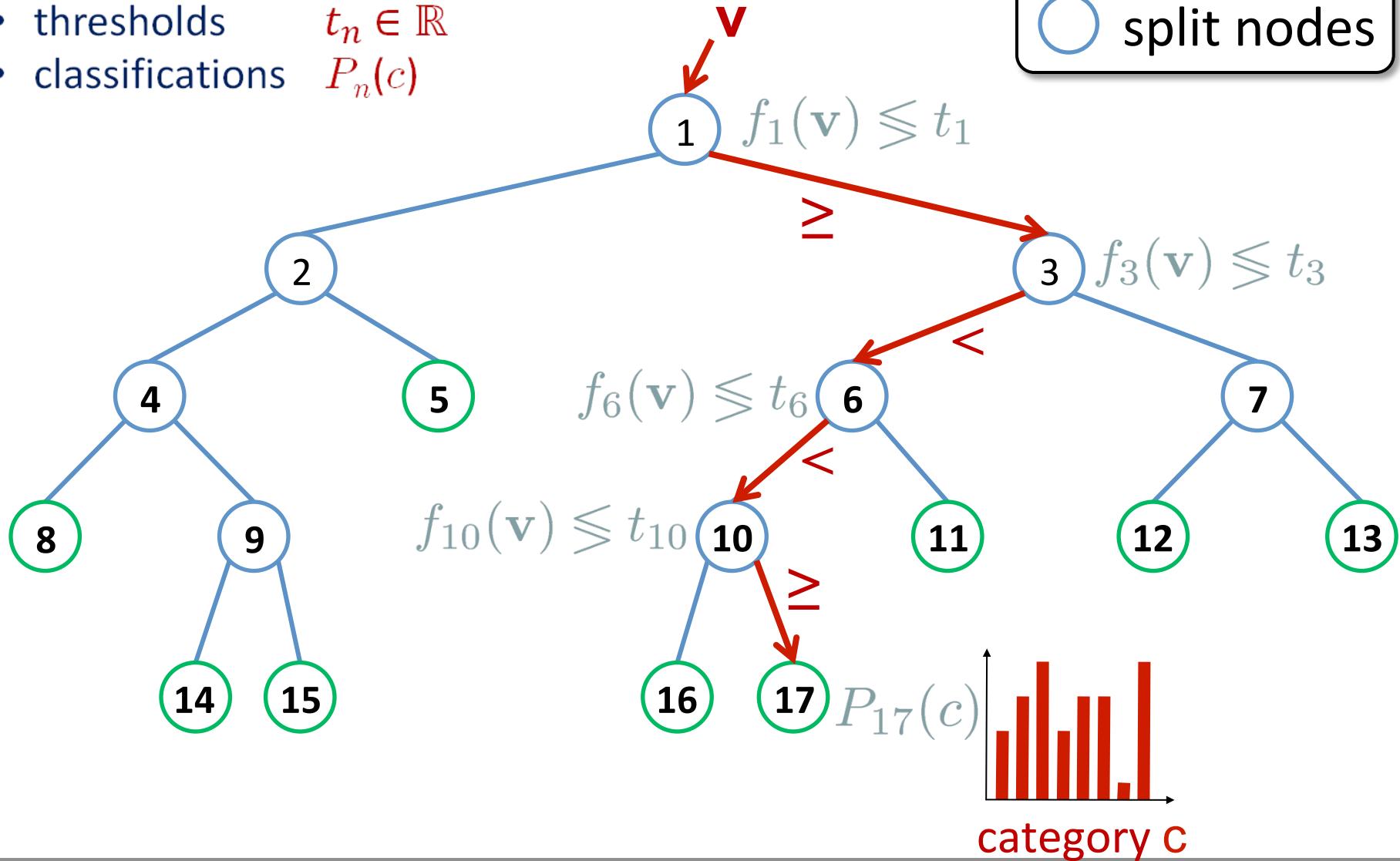
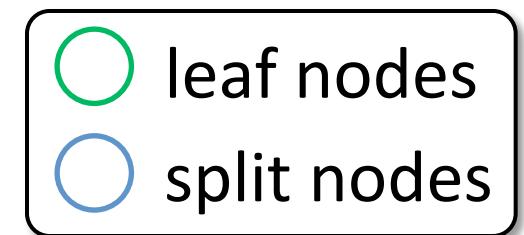


Is The Grass Wet?



Binary Decision Trees

- feature vector $v \in \mathbb{R}^N$
- split functions $f_n(v) : \mathbb{R}^N \rightarrow \mathbb{R}$
- thresholds $t_n \in \mathbb{R}$
- classifications $P_n(c)$



Randomized Learning

- Recursively split examples at node n
 - set I_n indexes labeled training examples (\mathbf{v}_i, l_i):

$$\begin{aligned} \text{left split} \rightarrow I_l &= \{i \in I_n \mid f(\mathbf{v}_i) < t\} \\ \text{right split} \rightarrow I_r &= I_n \setminus I_l \end{aligned}$$

↑ *↑*
**function of
example i's
feature vector** **threshold**

- At node n, $P_n(c)$ is histogram of example labels l_i

More Randomized Learning

$$\begin{aligned}\text{left split } I_l &= \{i \in I_n \mid f(\mathbf{v}_i) < t\} \\ \text{right split } I_r &= I_n \setminus I_l\end{aligned}$$

- Features $f(v)$ chosen at random from feature pool $f \in \mathcal{F}$
- Thresholds t chosen in range $t \in (\min_i f(\mathbf{v}_i), \max_i f(\mathbf{v}_i))$
- Choose f and t to maximize gain in information

$$\Delta E = -\frac{|I_l|}{|I_n|}E(I_l) - \frac{|I_r|}{|I_n|}E(I_r)$$

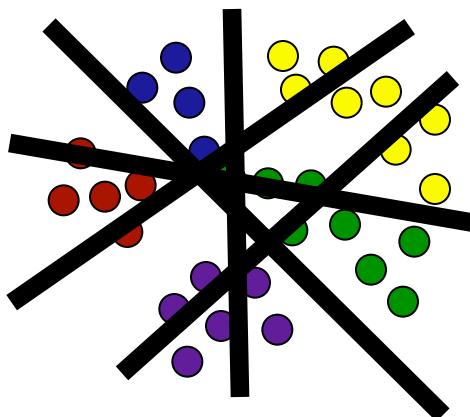
Entropy E calculated from histogram of labels in I

- Design choices: how many thresholds? Which features? When should we stop growing?

Why do random tests work?

For a small number of classes

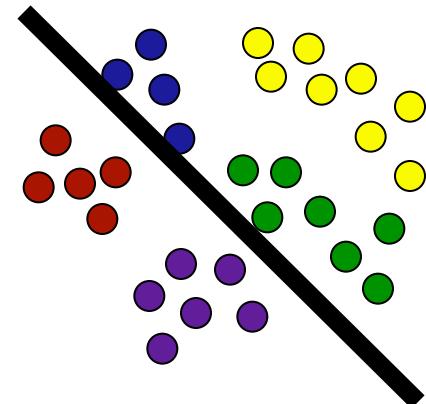
- we can try several tests, and
- retain the best one according to some criterion.



Why do random tests work?

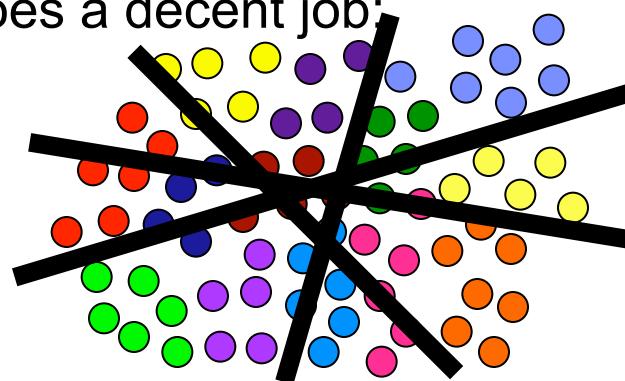
For a small number of classes

- we can try several tests, and
- retain the best one according to some criterion.



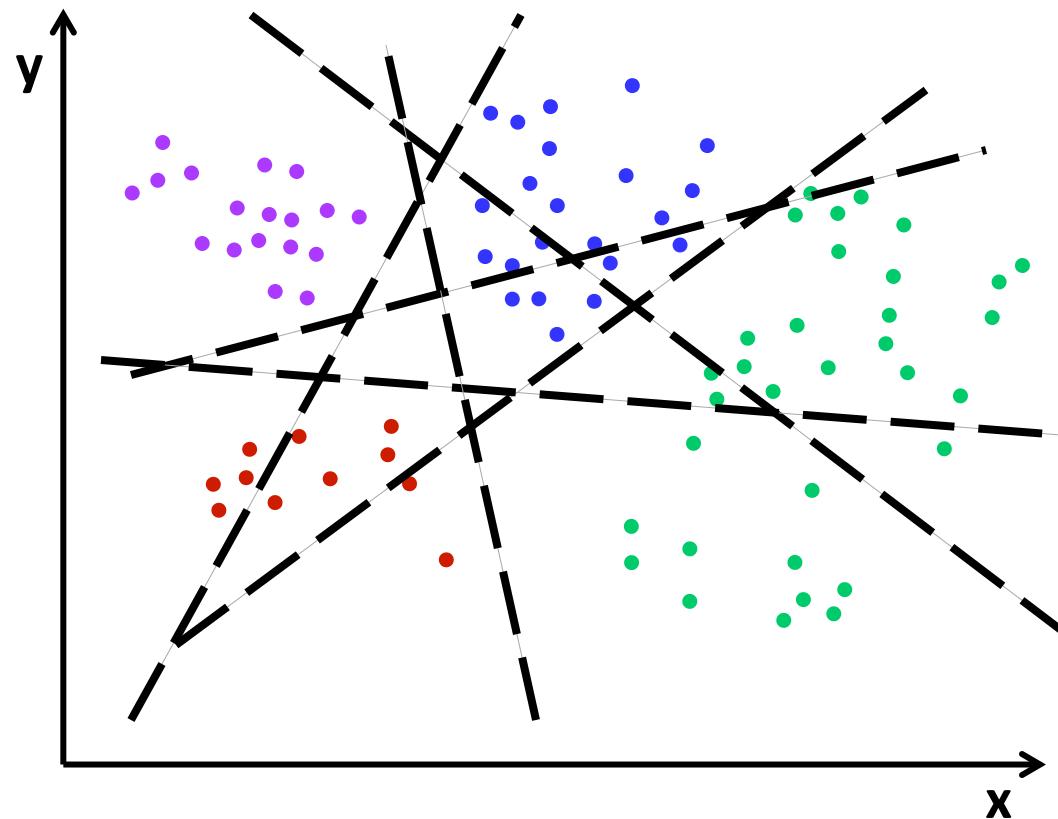
When the number of classes is large

- any test does a decent job:



Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
 - information gain
- Recurse

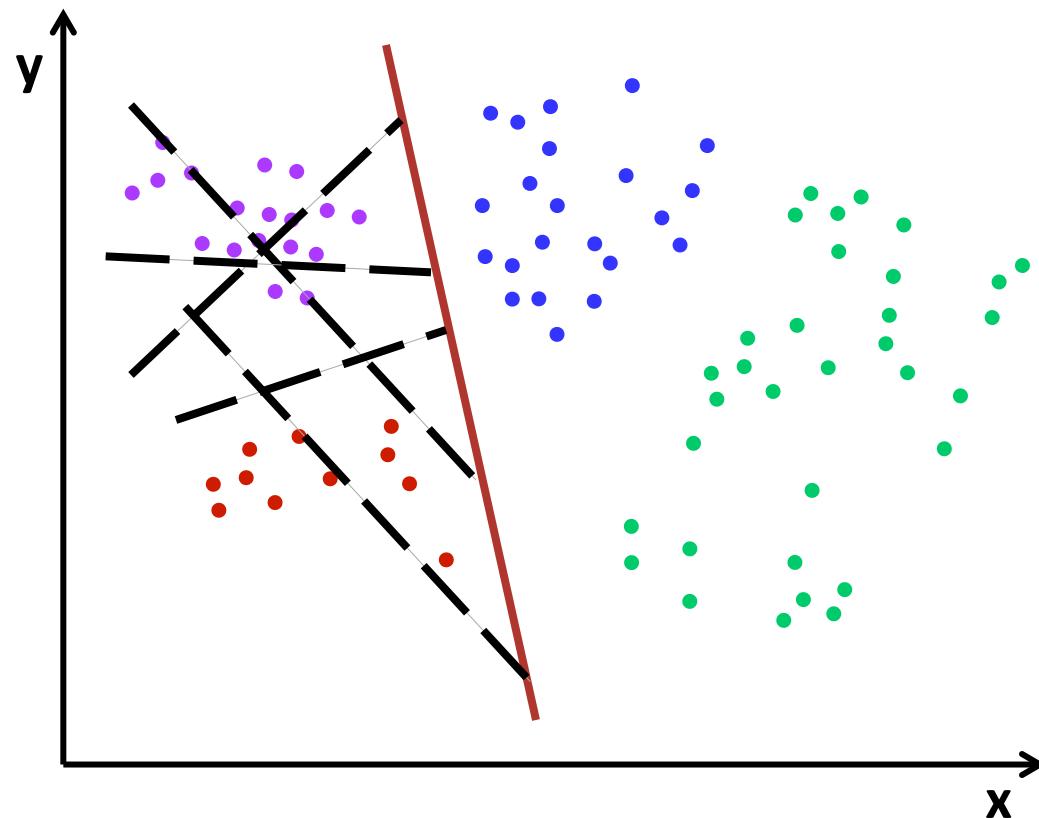


- feature vectors are x, y coordinates:
- split functions are lines with parameters a, b :
- threshold determines intercepts:
- four classes: purple, blue, red, green

$$\begin{aligned} \mathbf{v} &= [x, y]^T \\ f_n(\mathbf{v}) &= ax + by \\ t_n \end{aligned}$$

Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
 - information gain
- Recurse

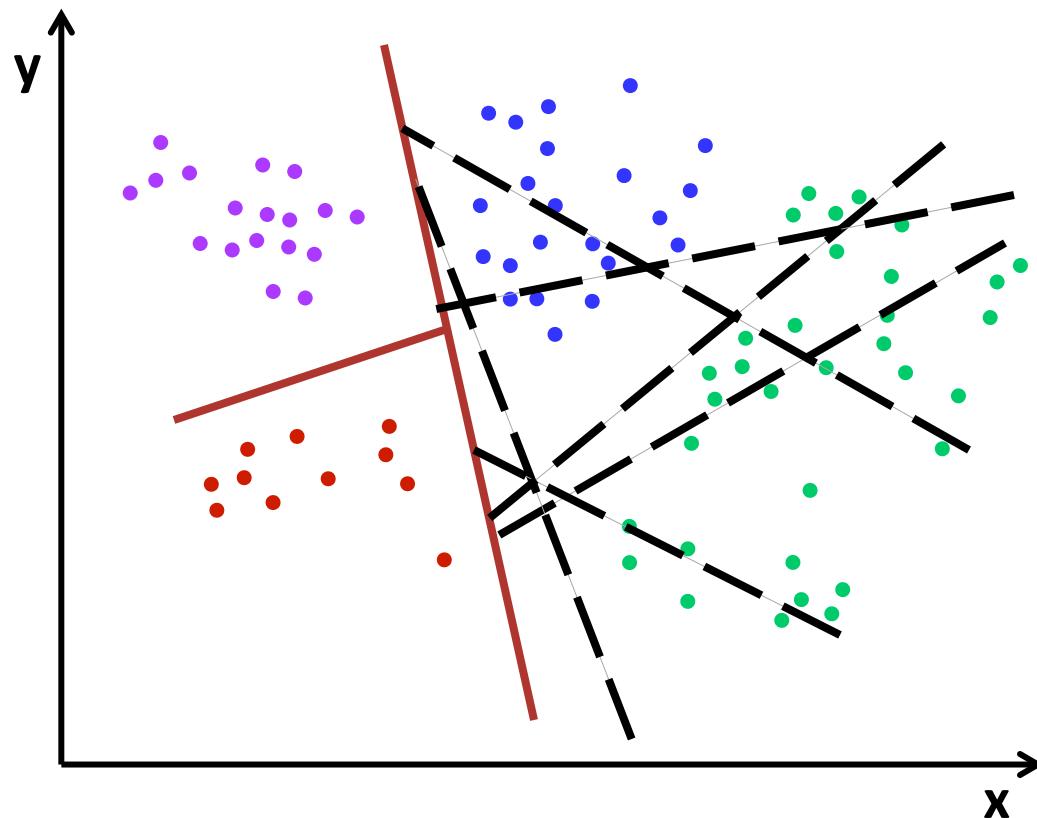


- feature vectors are x, y coordinates:
- split functions are lines with parameters a, b :
- threshold determines intercepts:
- four classes: purple, blue, red, green

$$\begin{aligned} \mathbf{v} &= [x, y]^T \\ f_n(\mathbf{v}) &= ax + by \\ t_n \end{aligned}$$

Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
 - information gain
- Recurse

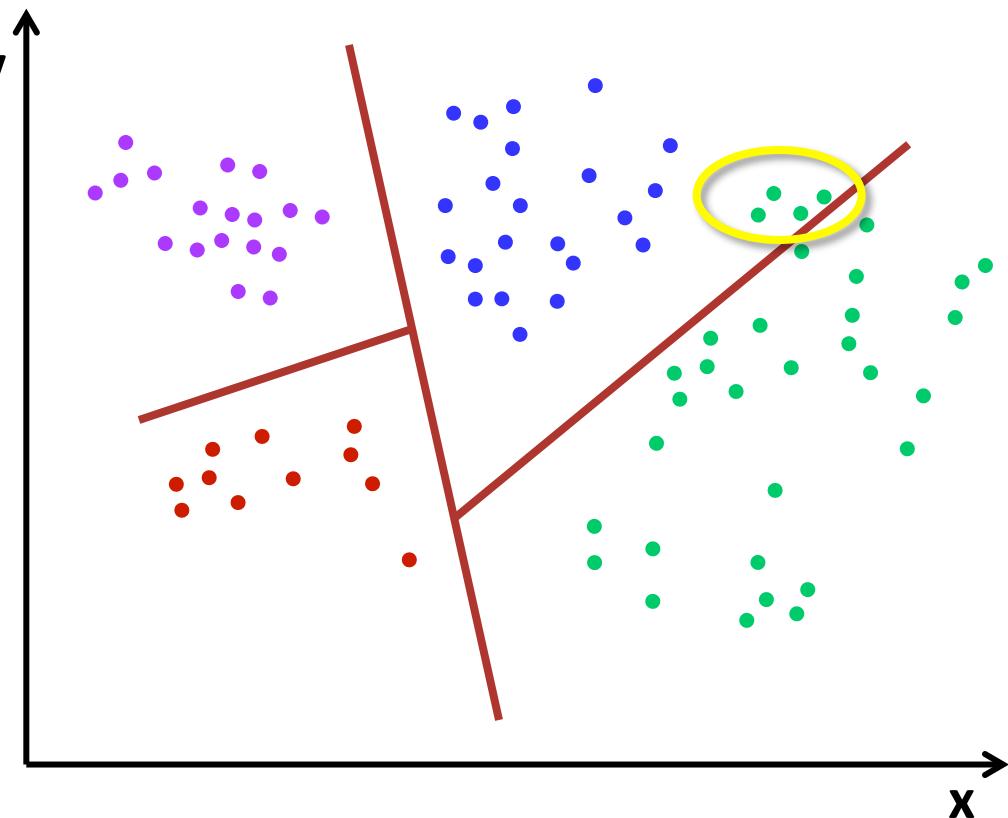


- feature vectors are x, y coordinates:
- split functions are lines with parameters a, b :
- threshold determines intercepts:
- four classes: purple, blue, red, green

$$\begin{aligned} \mathbf{v} &= [x, y]^T \\ f_n(\mathbf{v}) &= ax + by \\ t_n \end{aligned}$$

Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
 - information gain
- Recurse

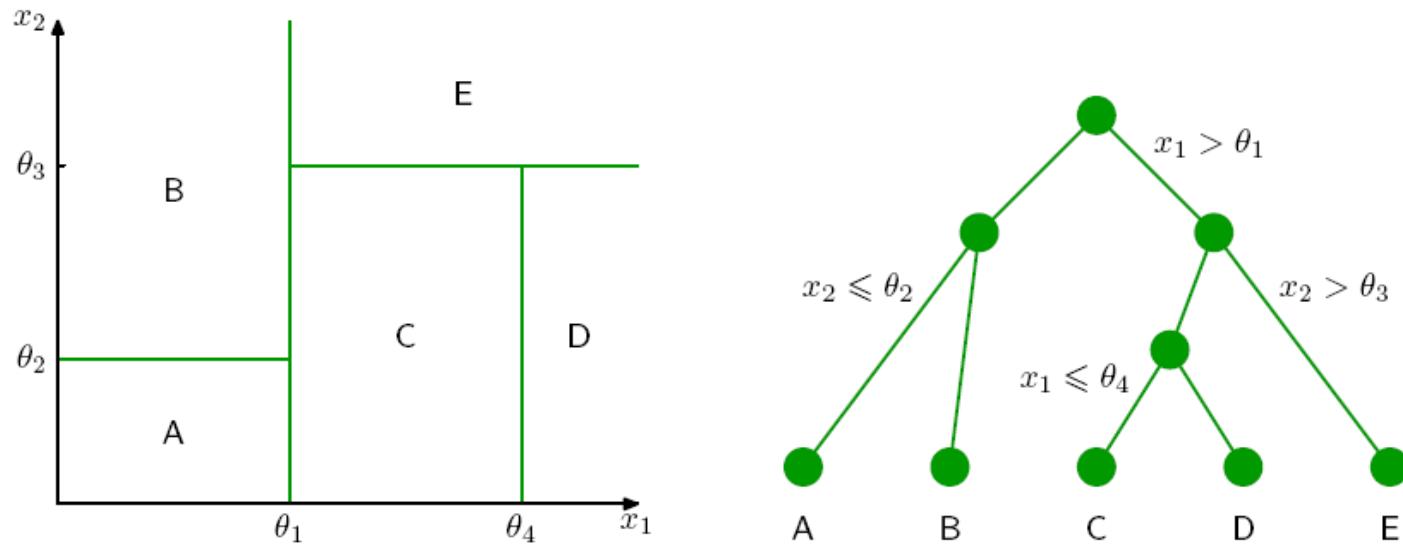


- feature vectors are x, y coordinates:
- split functions are lines with parameters a, b :
- threshold determines intercepts:
- four classes: purple, blue, red, green

$$\begin{aligned} \mathbf{v} &= [x, y]^T \\ f_n(\mathbf{v}) &= ax + by \\ t_n \end{aligned}$$

Which level of classification tree?

- Deeper trees result in more complex classifiers



- After some point overfitting occurs

Binary Decision Trees Summary

- Fast greedy training algorithms
 - can search infinite pool of features
 - heterogeneous pool of features
- Fast testing algorithm
- Needs careful choice of hyper-parameters
 - maximum depth
 - number of features and thresholds to try
- Prone to over-fitting



Lecture outline

Recap

Adaboost

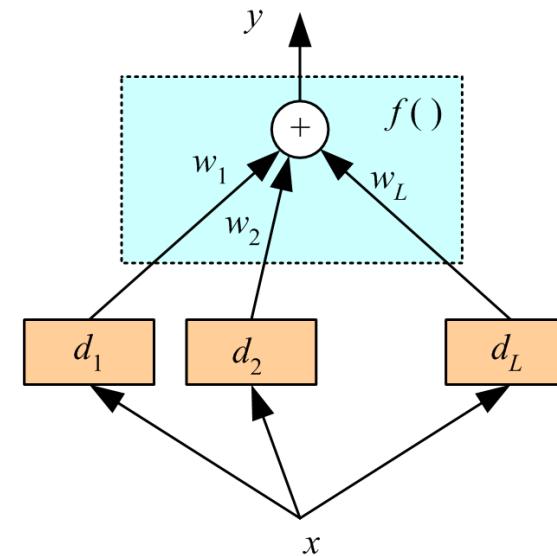
Decision Trees

Random Forests & Ferns

Voting Methods

- Give up idea of building ‘the’ classifier
- Generate a group of **base-learners** which has higher accuracy when combined
- Main tasks
 - Generating the learners
 - Combining them

$$y_{COM}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x})$$

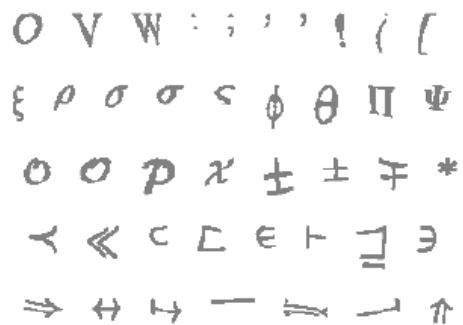


If errors have zero mean and are uncorrelated: $\mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})] = 0$

then $\mathbb{E}_{COM} = \frac{1}{M} \mathbb{E}_{AV}$

$$\mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x}) \epsilon_j(\mathbf{x})] = 0$$

Randomized Forests in Vision



[Amit & Geman, 97]
digit recognition



[Lepetit *et al.*, 06]
keypoint recognition



[Moosmann *et al.*, 06]
visual word clustering

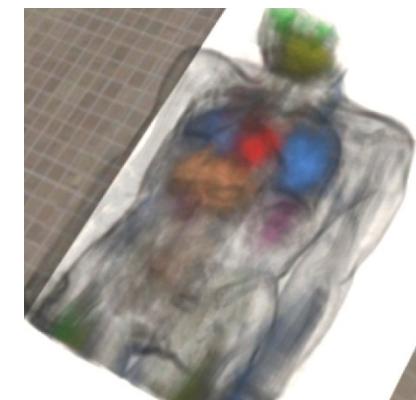


water
boat
tree
chair
road

[Shotton *et al.*, 08]
object segmentation



[Rogez *et al.*, 08]
pose estimation

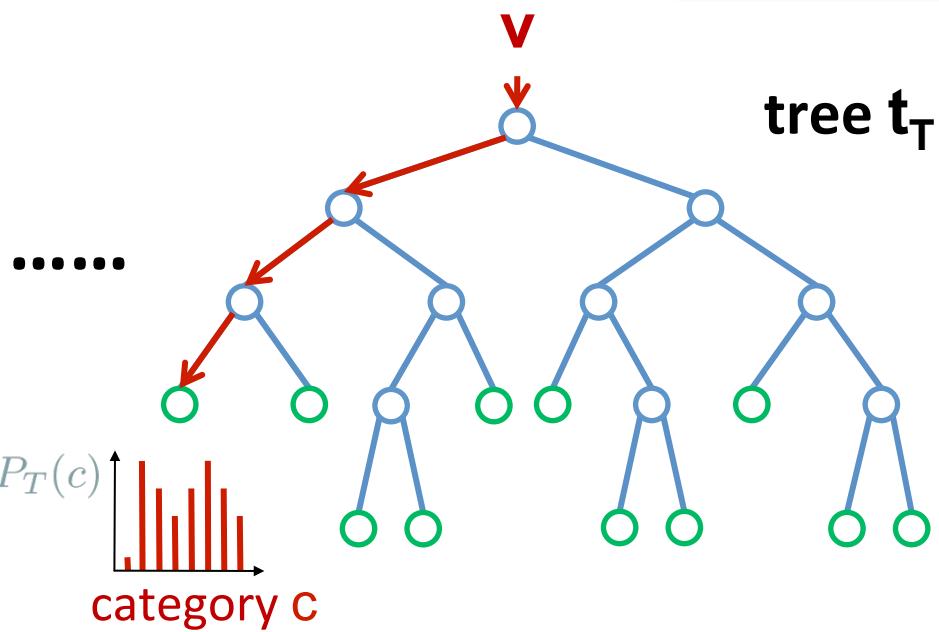
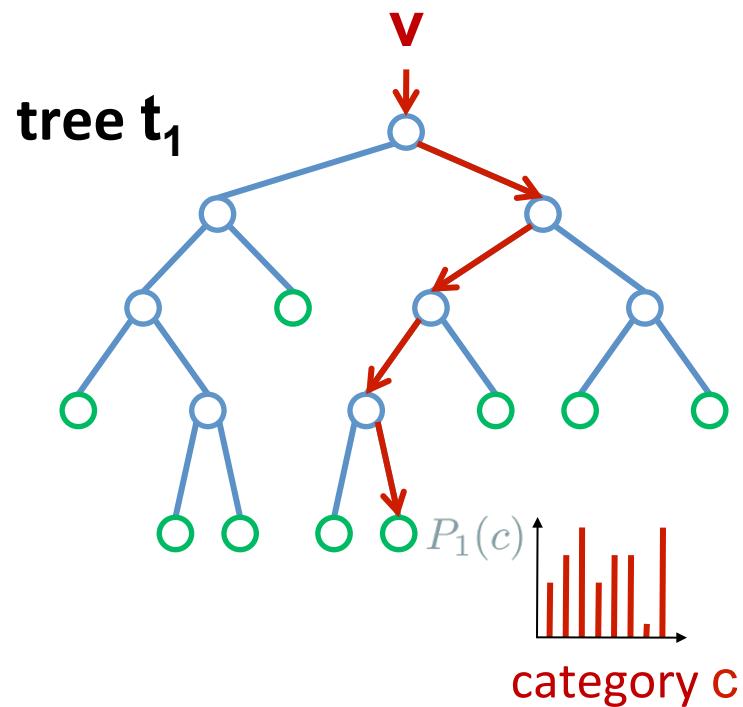
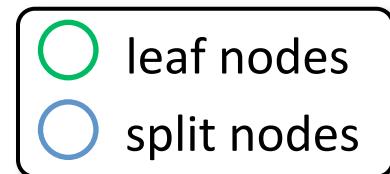


[Criminisi *et al.*, 09]
organ detection

(Among many others...)

A Forest of Trees

- Forest is ensemble of several decision trees



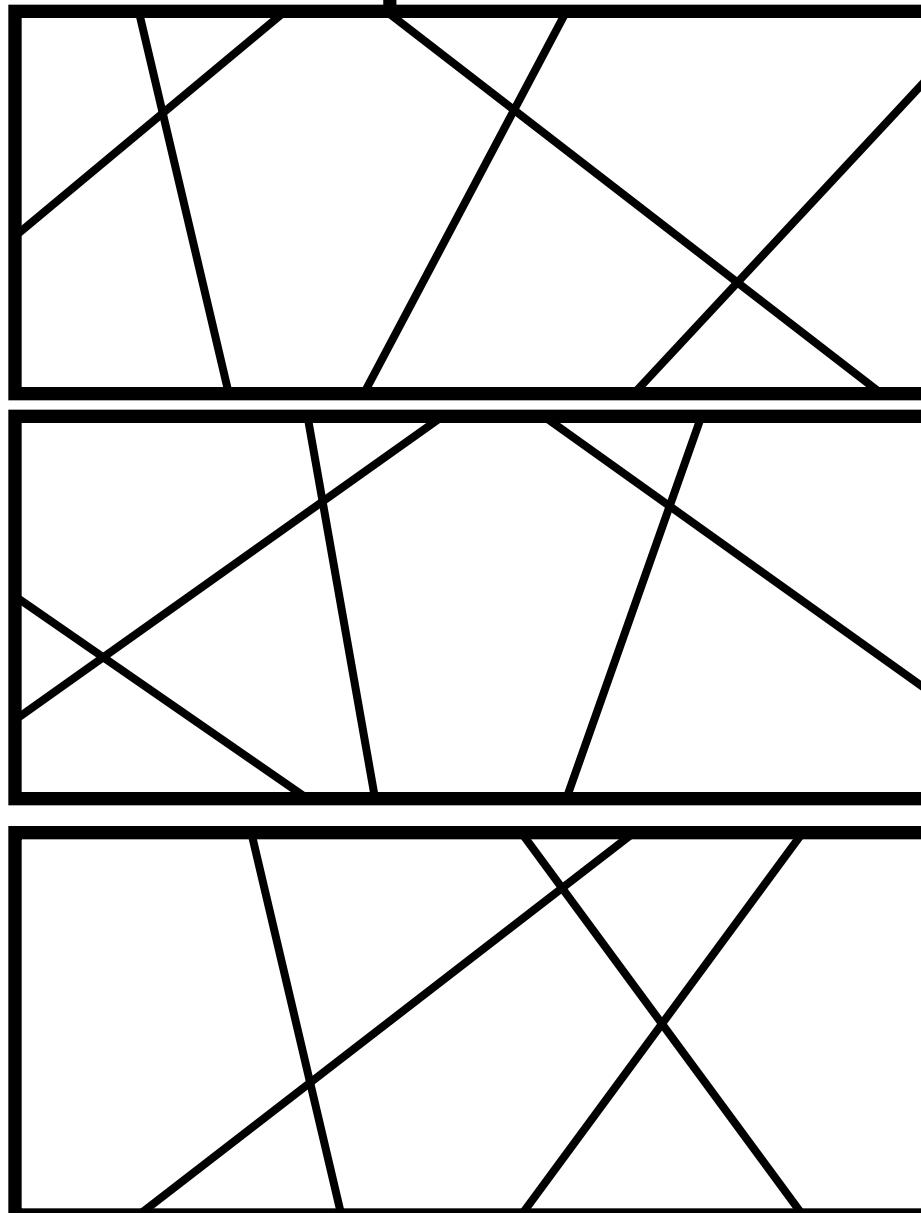
– classification is
$$P(c|\mathbf{v}) = \frac{1}{T} \sum_{t=1}^T P_t(c|\mathbf{v})$$

[Amit & Geman 97]
 [Breiman 01]
 [Lepetit *et al.* 06]

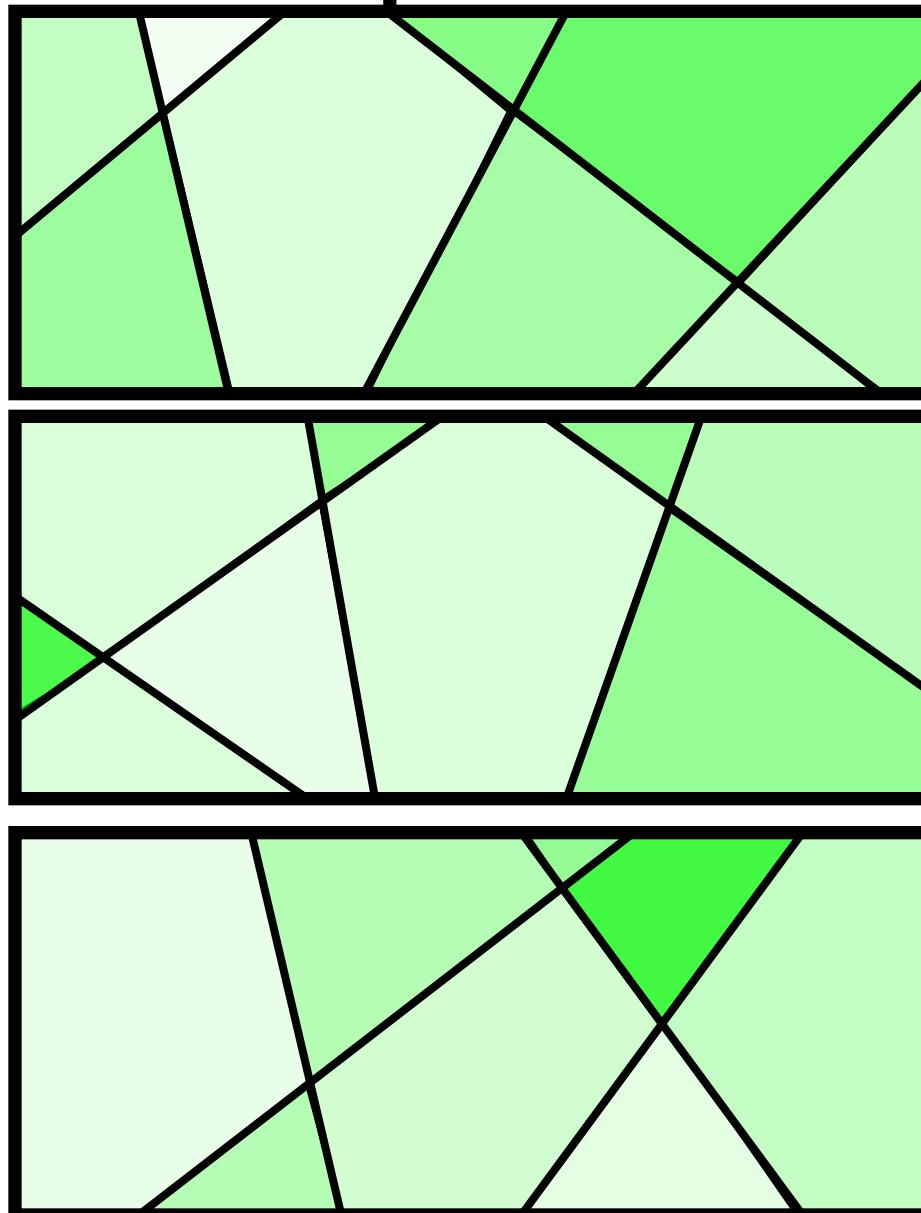
Learning a Forest

- Divide training examples into T subsets I_t
 - **improves generalization**
 - reduces **memory requirements & training time**
- Train each decision tree t on subset I_t
 - same decision tree learning as before

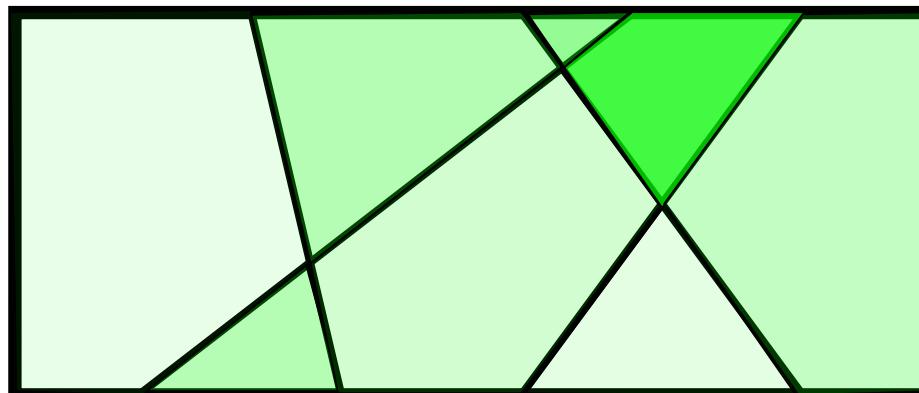
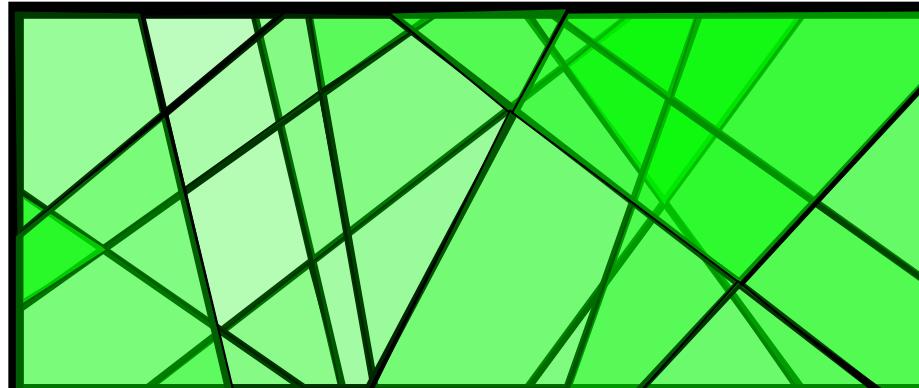
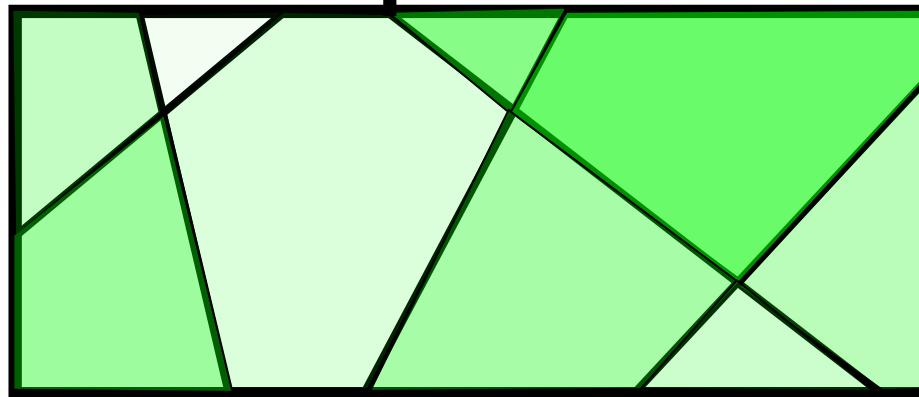
A Graphical Interpretation



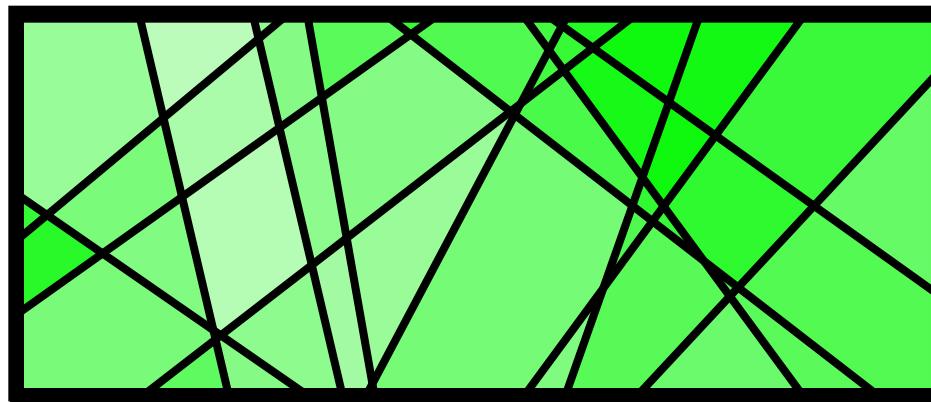
A Graphical Interpretation



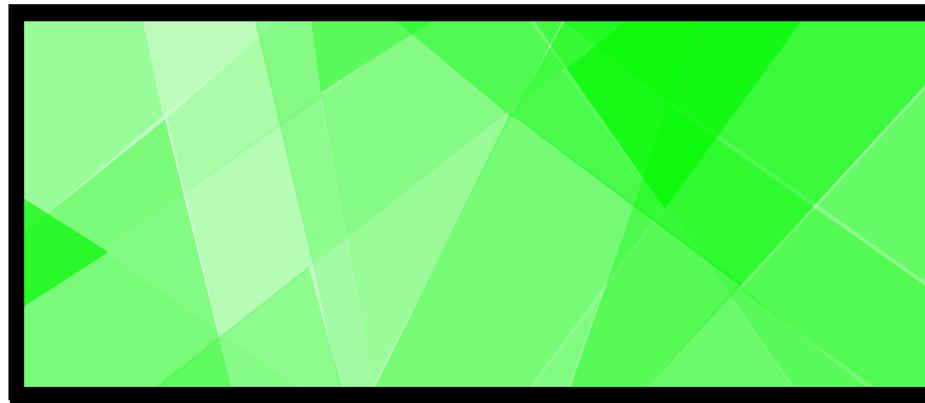
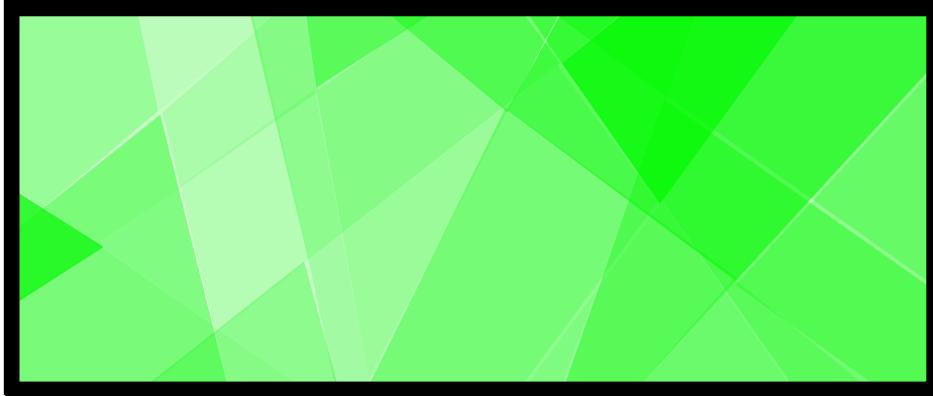
A Graphical Interpretation



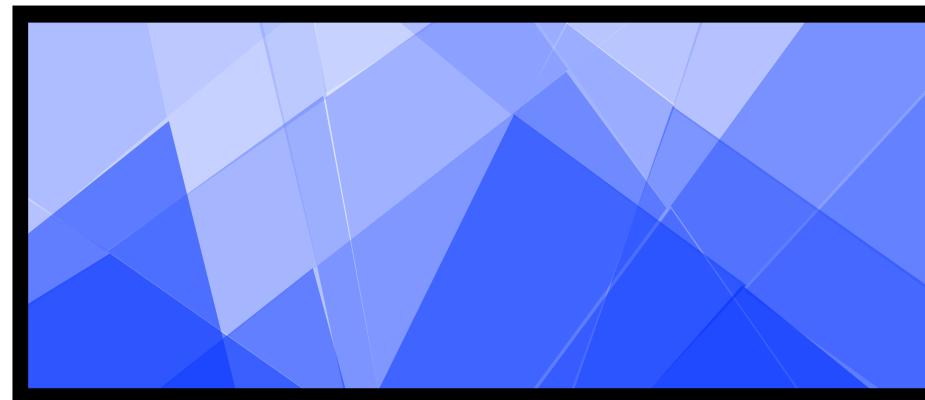
A Graphical Interpretation



A Graphical Interpretation



A Graphical Interpretation



Relation to Cascades

[Viola & Jones 04]

- Boosted Cascades
 - very unbalanced tree
 - good for unbalanced binary problems
e.g. sliding window object detection
- Randomized forests
 - less deep, fairly balanced
 - ensemble of trees gives robustness
 - good for multi-class problems



Real-Time Object Segmentation

[Shotton *et al.* 2008]

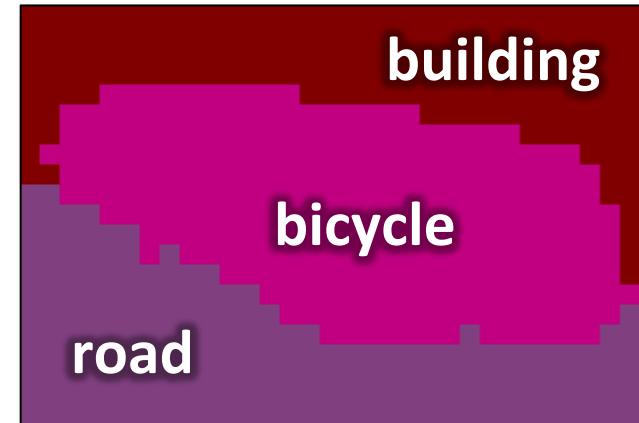
- Segment image and label segments in real-time



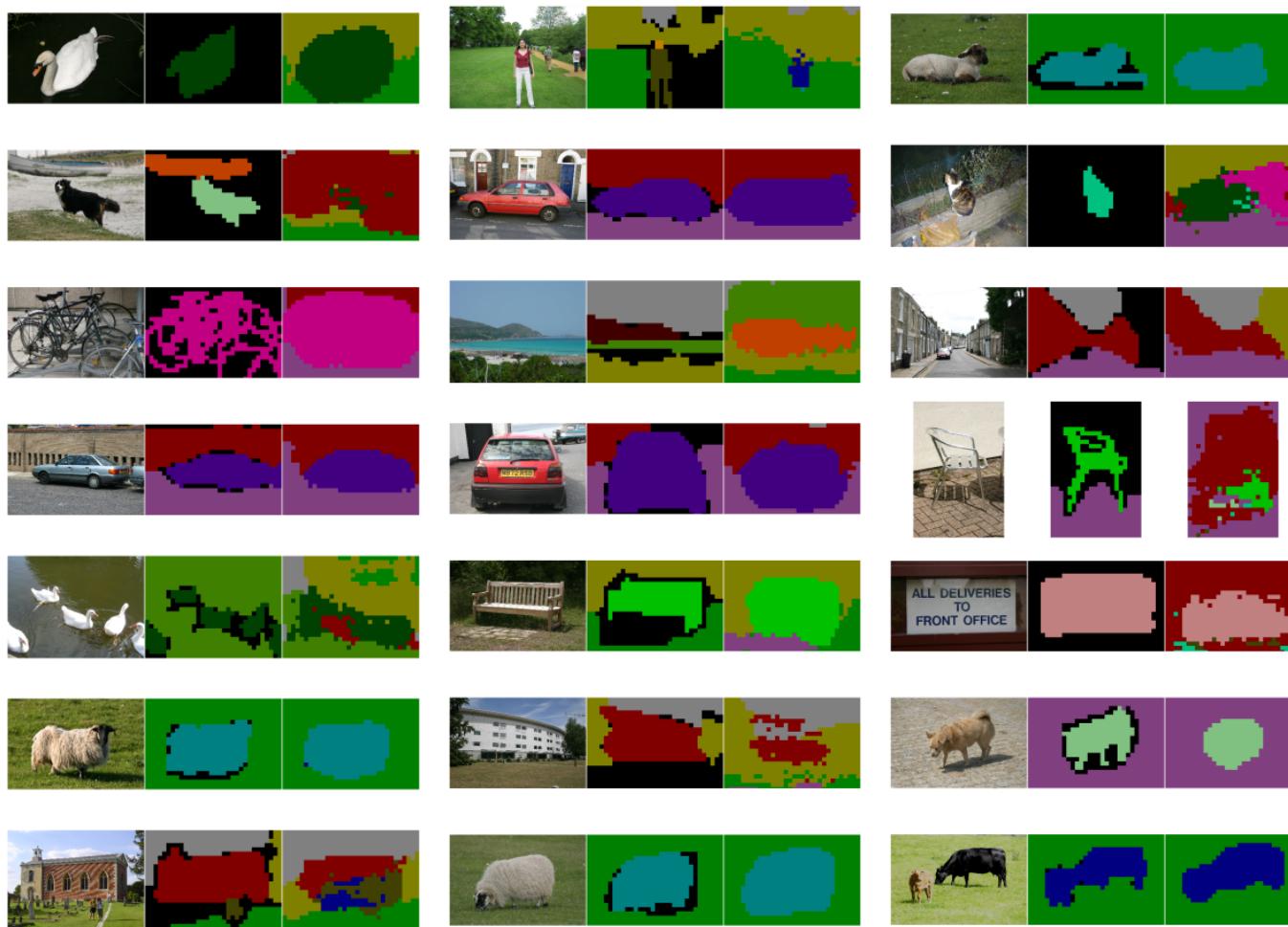
CVPR 2008 Best Demo Award

Segmentation Forest

- Object segmentation



MSRC Dataset Results



building	grass	tree	cow	sheep	sky	airplane	water	face	car	bicycle
	flower	sign	bird	book	chair	road	cat	dog	body	

\$UCCE\$\$ STORY: Kinect



12.1 million sold by end 2011



Kinect's pipeline

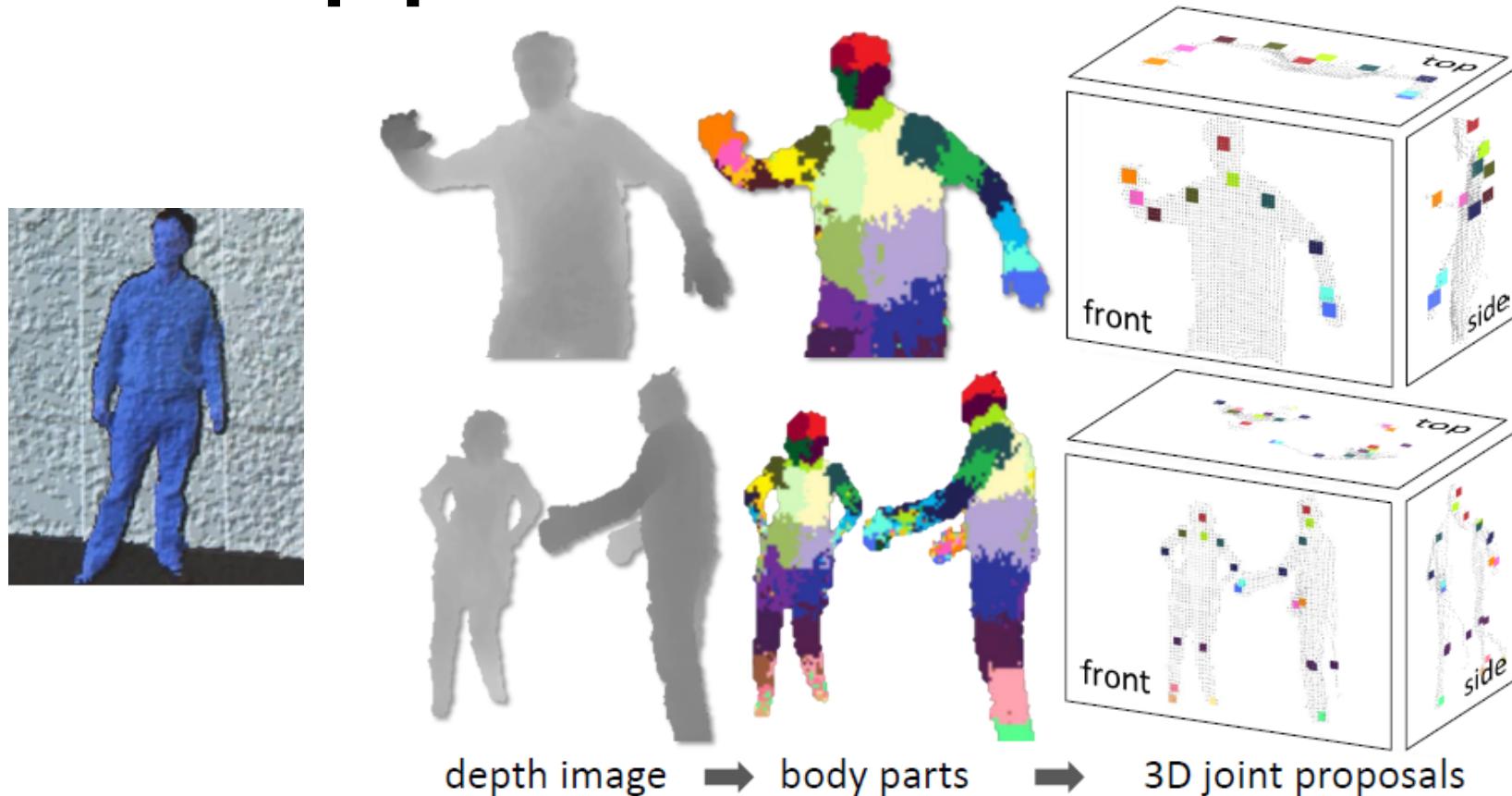
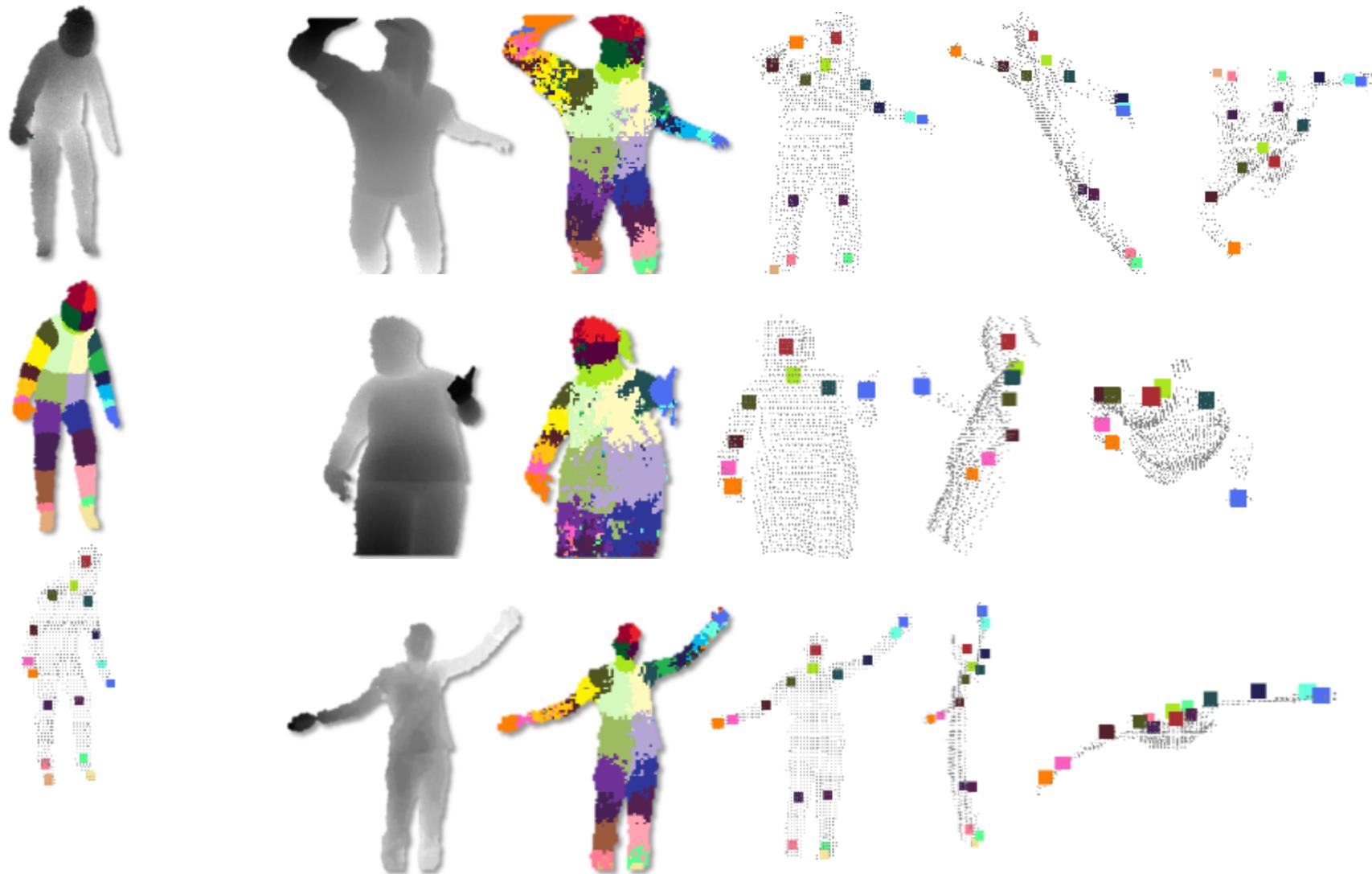


Figure 1. Overview. From a single input depth image, a per-pixel body part distribution is inferred. (Colors indicate the most likely part labels at each pixel, and correspond in the joint proposals). Local modes of this signal are estimated to give high-quality proposals for the 3D locations of body joints, even for multiple users.



- Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake,
[Real-Time Human Pose Recognition in Parts from a Single Depth Image](#), in CVPR, IEEE, June 2011
- **Abstract:** We propose a new method to quickly and accurately predict 3D positions of body joints from a single depth image, using no temporal information. We take an object recognition approach, designing an intermediate body parts representation that **maps the difficult pose estimation problem into a simpler per-pixel classification problem**. Our **large and highly varied training dataset** allows the classifier to estimate body parts invariant to pose, body shape, clothing, etc. Finally we generate confidence-scored 3D proposals of several body joints by reprojecting the classification result and finding local modes.
- **The system runs at 200 frames per second on consumer hardware.** Our evaluation shows high accuracy on both synthetic and real test sets, and investigates the effect of several training parameters. We achieve state of the art accuracy in our comparison with related work and demonstrate improved generalization over exact whole-skeleton nearest neighbor matching.



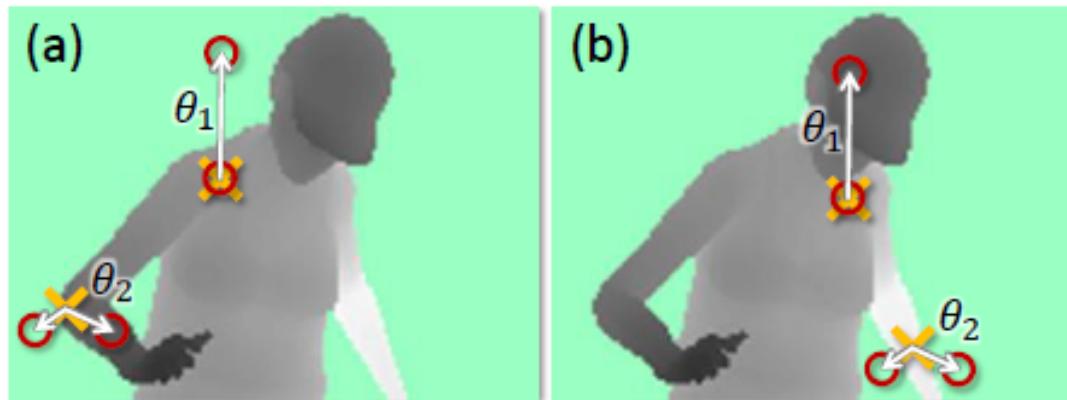


Figure 3. Depth image features. The yellow crosses indicates the pixel x being classified. The red circles indicate the offset pixels as defined in Eq. 1. In (a), the two example features give a large depth difference response. In (b), the same two features at new image locations give a much smaller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

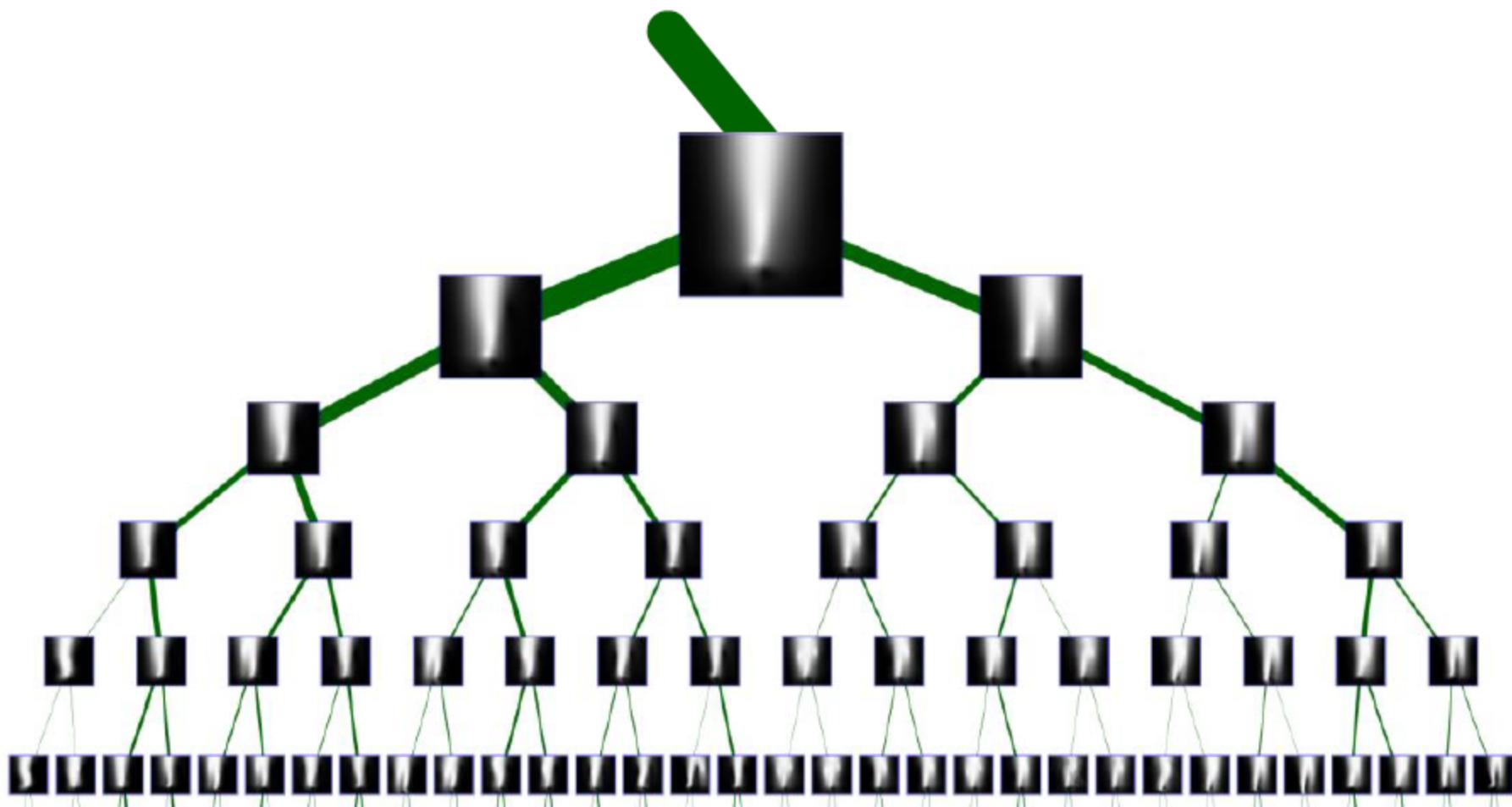
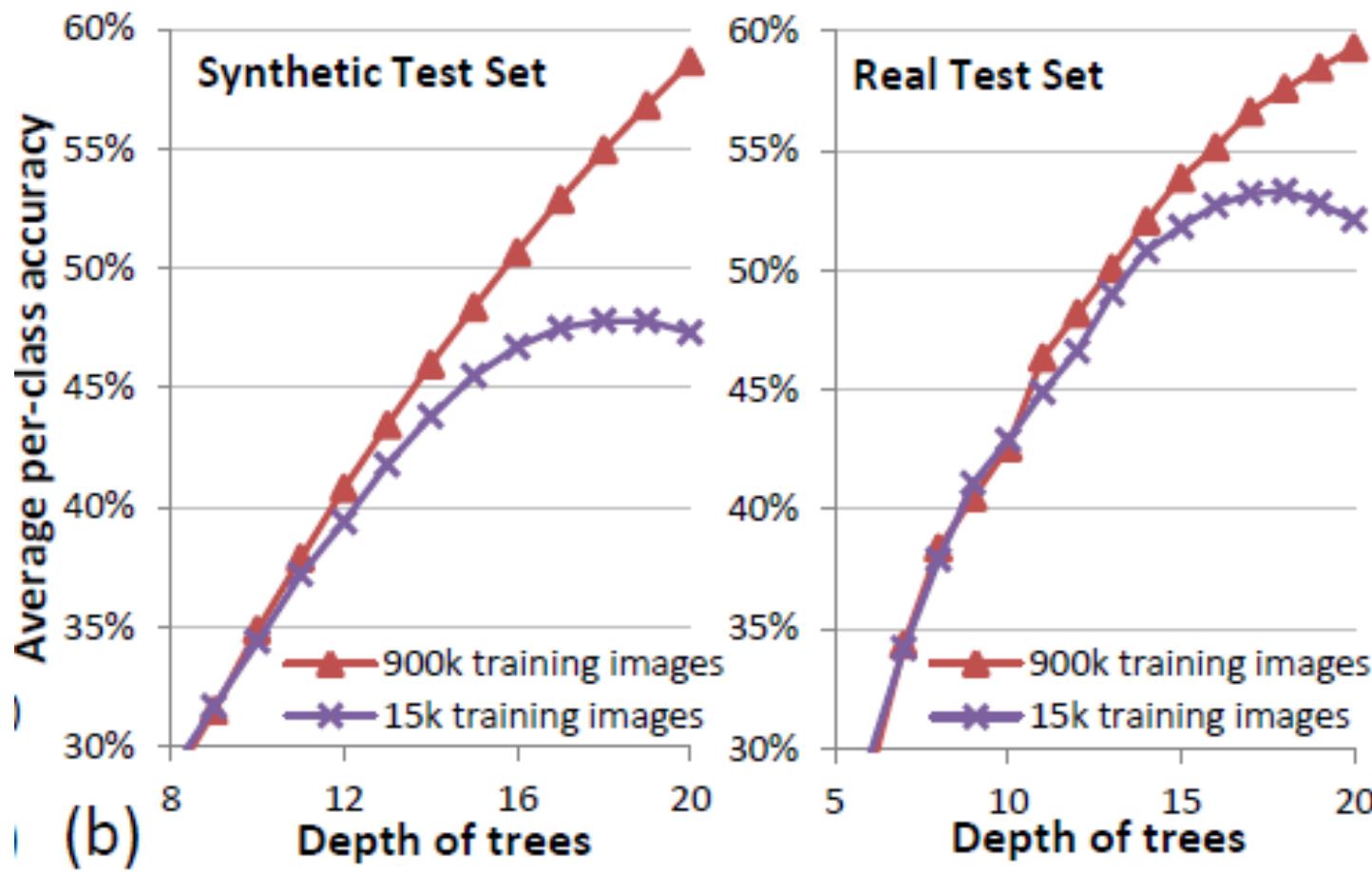


Figure 11. Visualization of a trained decision tree. Two separate subtrees are shown. A depth image patch centered on each pixel is taken, depth normalized, and binarized to a foreground/background silhouette. The patches are averaged across all pixels that reached any given tree node. The thickness of the edges joining the tree nodes is proportional to the number of pixels, and here shows fairly balanced trees. All pixels from 15k images are used to build the visualization shown.



Why the class is not over yet

