



Patten Recognition

Lecture 9

Lecturer: Xinchao Wang

xinchao@nus.edu.sg



What we have learned so far...

- Classification using single models
 - Bayesian
 - LDA
 - SVM
- Can we do better than single models?
 - Yes!
 - Using multiple models when possible!

ensemble

Ensemble Methods

Supervised.

- 1. Bagging (Breiman 1994,...)
 - Random forests (Breiman 2001,...)
- 2. Boosting (Freund and Schapire 1995, Friedman et al. 1998,...)



orderless model

A B
B A

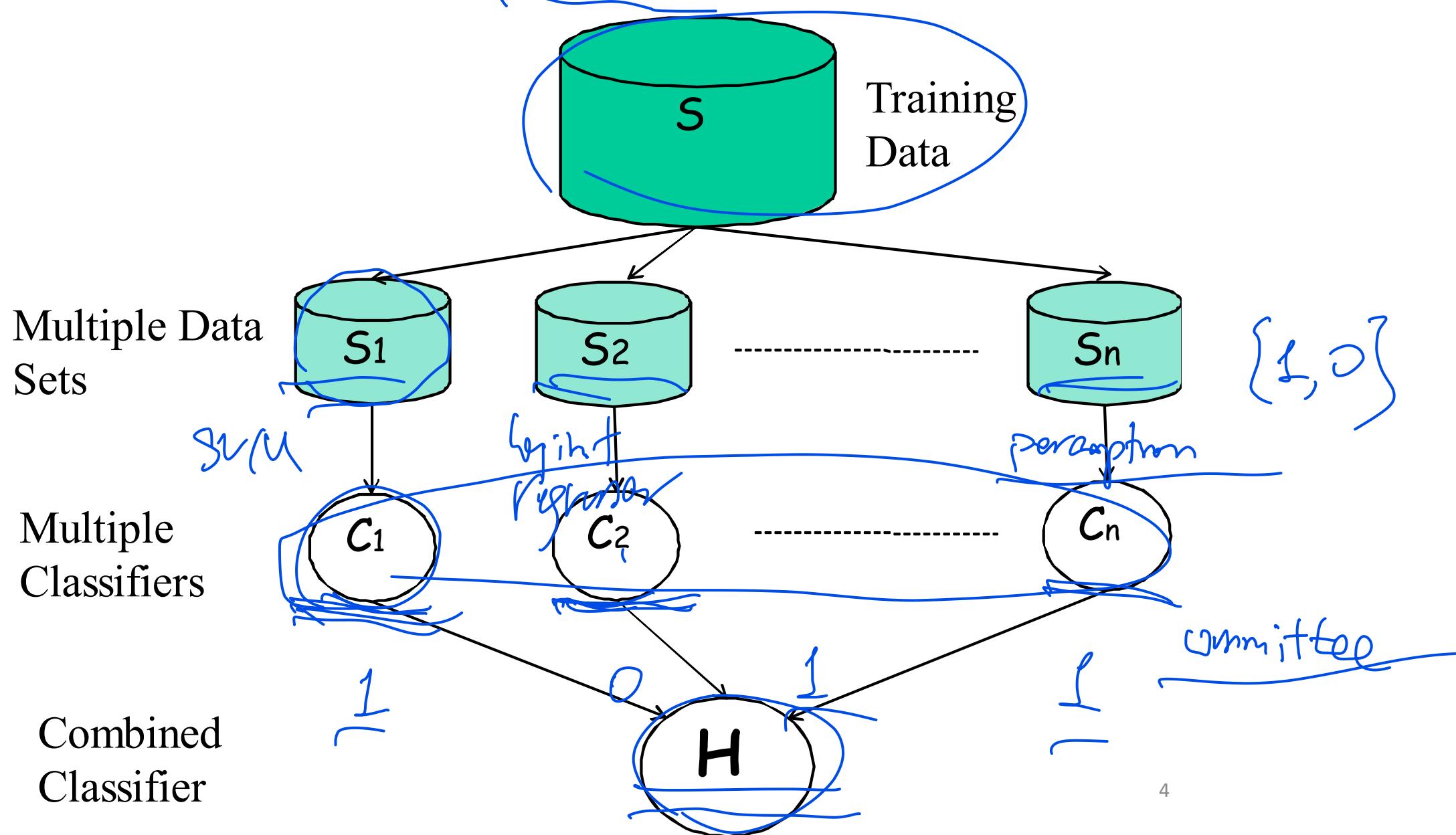


ordered model

A_1, A_2, A_3
 $\downarrow \cup \downarrow$

Predict class label for unseen data by aggregating a set of predictions (classifiers learned from the training data)

General Idea of Bagging



Ensemble Classifiers

- Basic idea: Build different “experts” and let them vote
- Advantages:
 - Improve predictive performance
 - Different types of classifiers can be directly included
 - Easy to implement
 - Not too much parameter tuning
- Disadvantages:
 - The combined classifier is not transparent (black box)
 - Not a compact representation

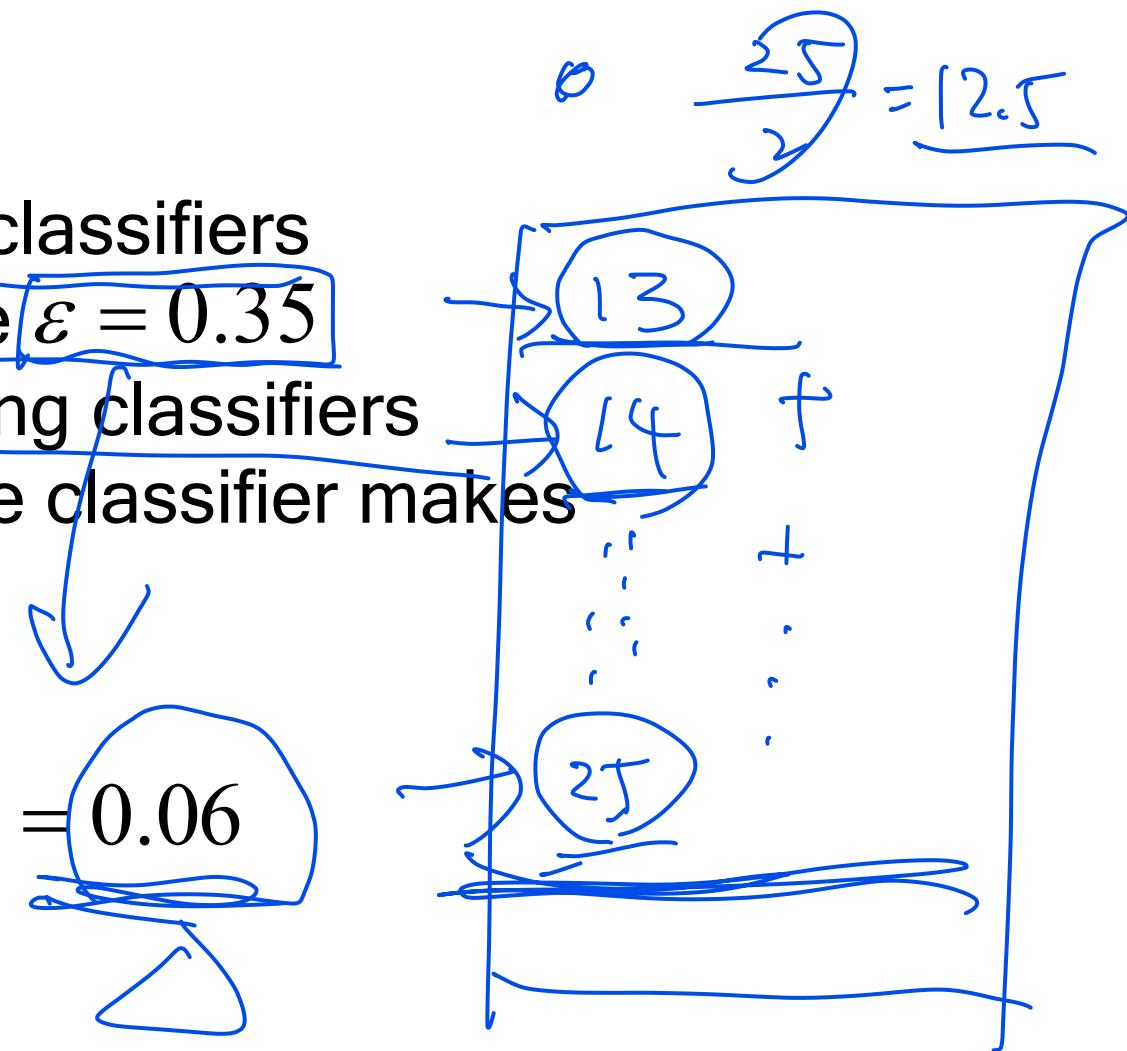
majority
vote

Why do they work?

$C_1 C_2 \dots C_{25}$

- Suppose there are 25 base classifiers
- Each classifier has error rate $\varepsilon = 0.35$
- Assume independence among classifiers
- Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$



25 Bagging = Bootstrap Aggregating

1
2
3
4

- Training

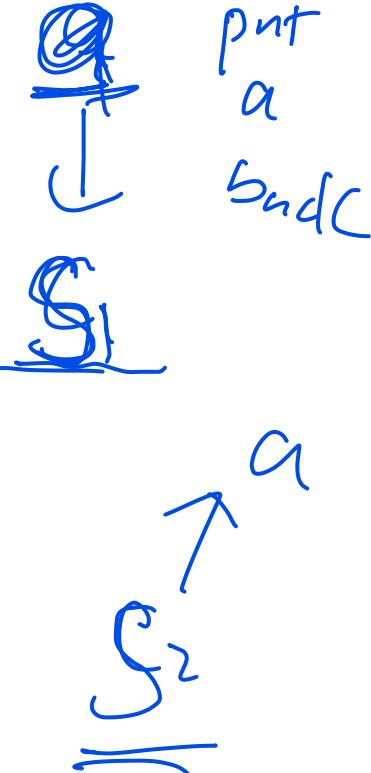
- Given a dataset S , at each iteration i , a training set S_i is sampled with replacement from S (i.e. bootstrapping)

- A classifier C_i is learned for each S_i

- Classification: given an unseen sample X

- Each classifier C_i returns its class prediction

- The bagged classifier H counts the votes and assigns the class with the most votes to X

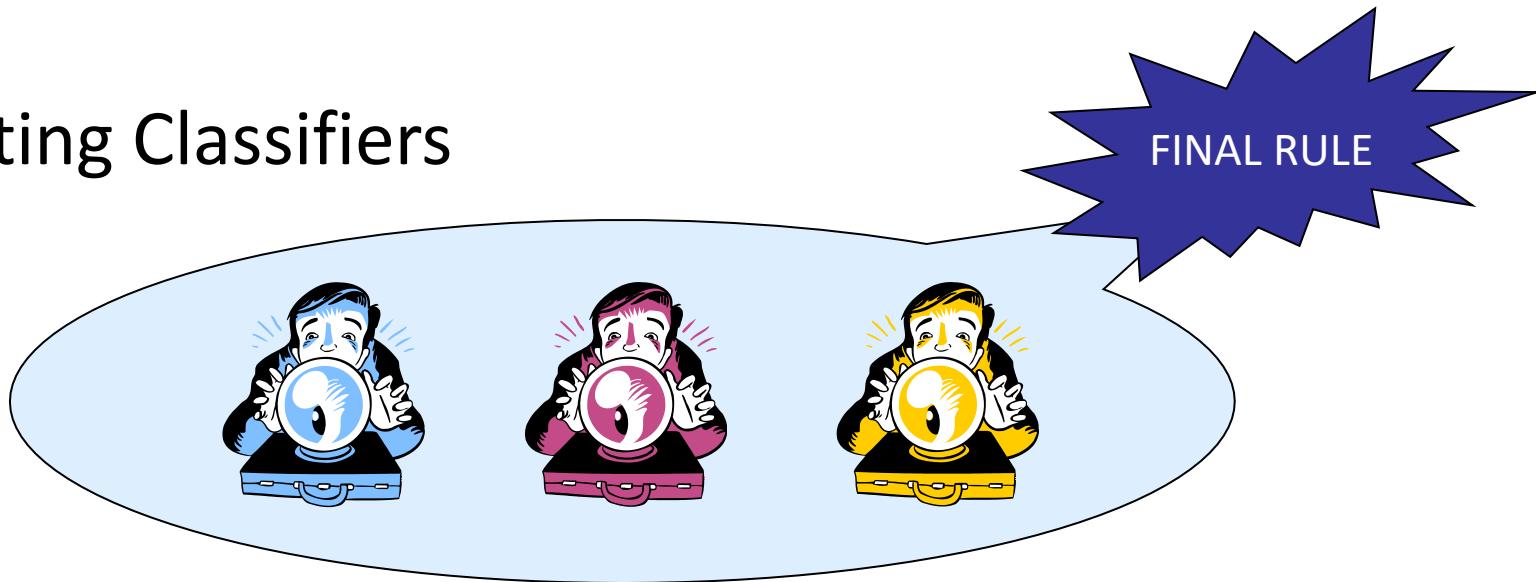


Bagging

- Bagging works because it reduces variance by voting/averaging
 - Usually, the more classifiers the better
- Problem: we only have one dataset
- Solution: generate new ones of size n by bootstrapping, i.e. sampling with replacement
- Can help a lot if data is noisy

Aggregating Classifiers

→ Aggregating Classifiers



- Breiman (1996) found gains in accuracy by aggregating predictors built from reweighted versions of the learning set

Bagging

- *Bagging* = Bootstrap Aggregating
- Reweighting of the learning sets is done by drawing at random with replacement from the learning sets
- Predictors are aggregated by voting

The Bagging Algorithm

- B bootstrap samples
- From which we derive:

- | | | | +

- B Classifiers $\in \{-1, 1\}$: $c^1, c^2, c^3, \dots, c^B$

$x - |$

- B Estimated probabilities $\in [0, 1]$: $p^1, p^2, p^3, \dots, p^B$

- The aggregate classifier becomes:

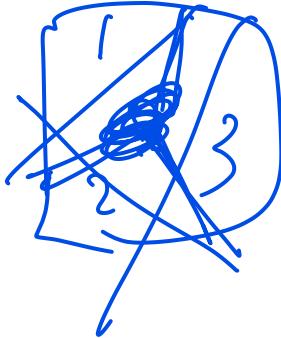
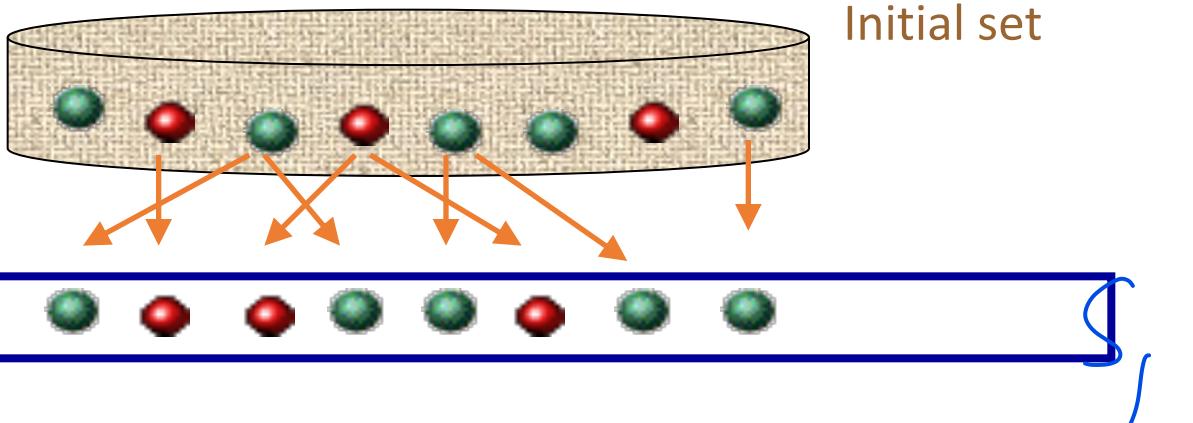
$$c_{bag}(x) = sign\left(\frac{1}{B} \sum_{b=1}^B c^b(x)\right)$$

or

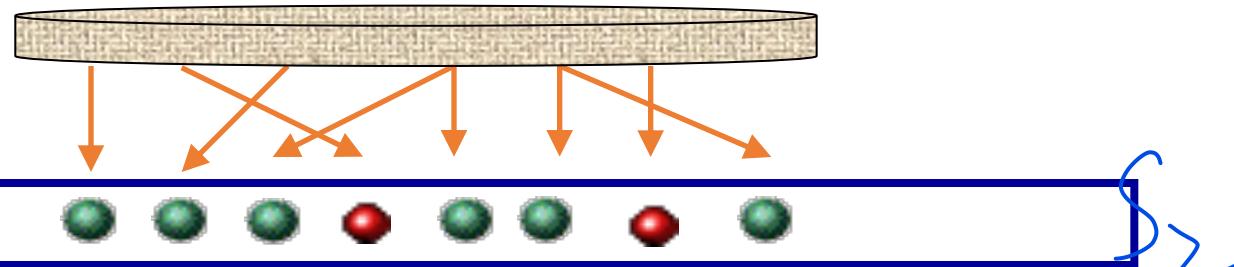
$$p_{bag}(x) = \frac{1}{B} \sum_{b=1}^B p^b(x)$$

$$\begin{aligned} p^1(x) \\ p^2(x) \end{aligned}$$

Weighting



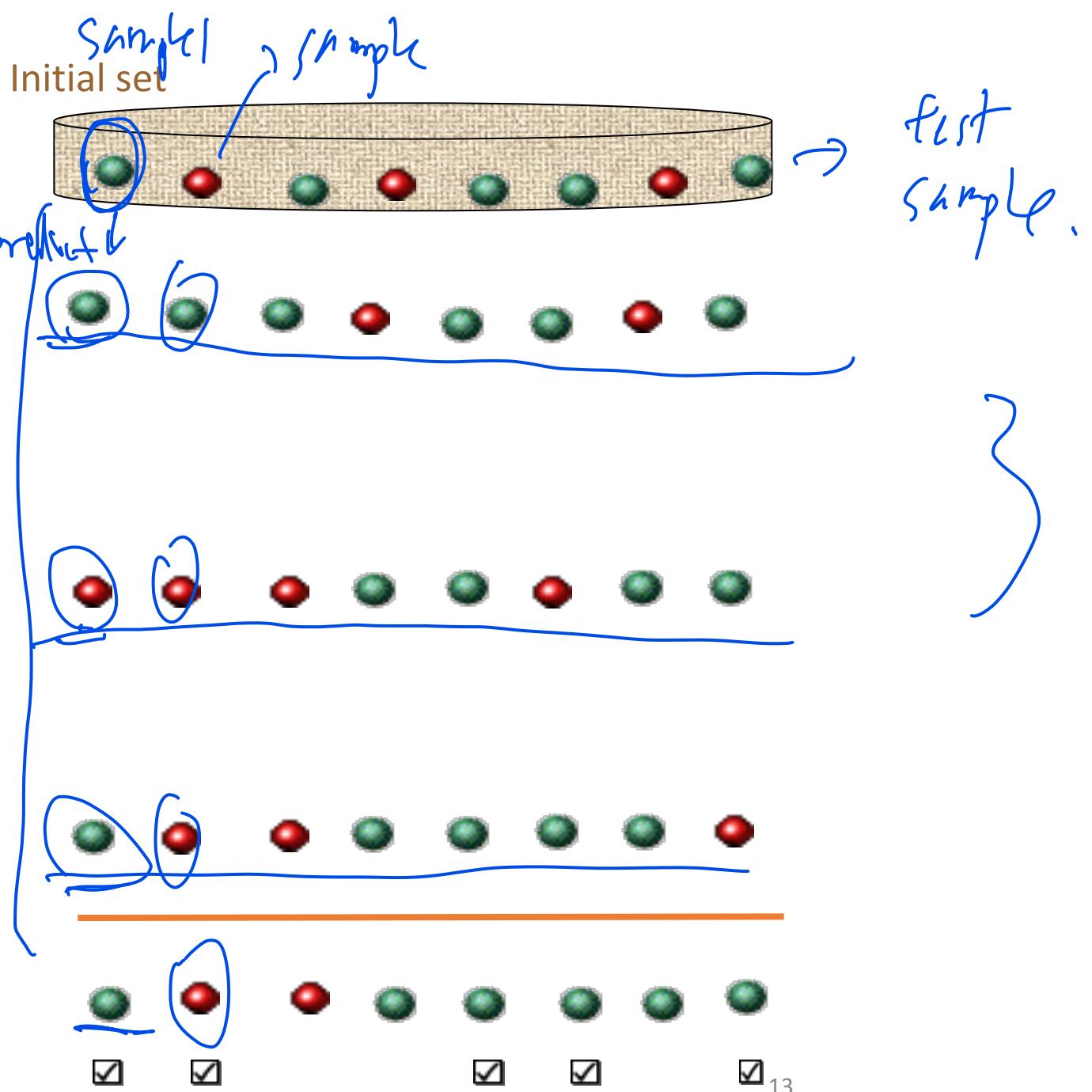
Classifier 1



Classifier 2



Aggregation

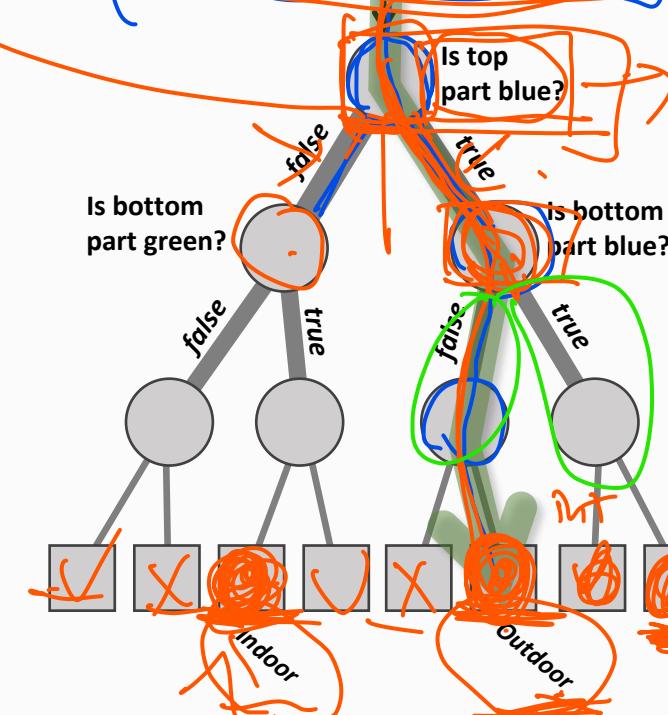
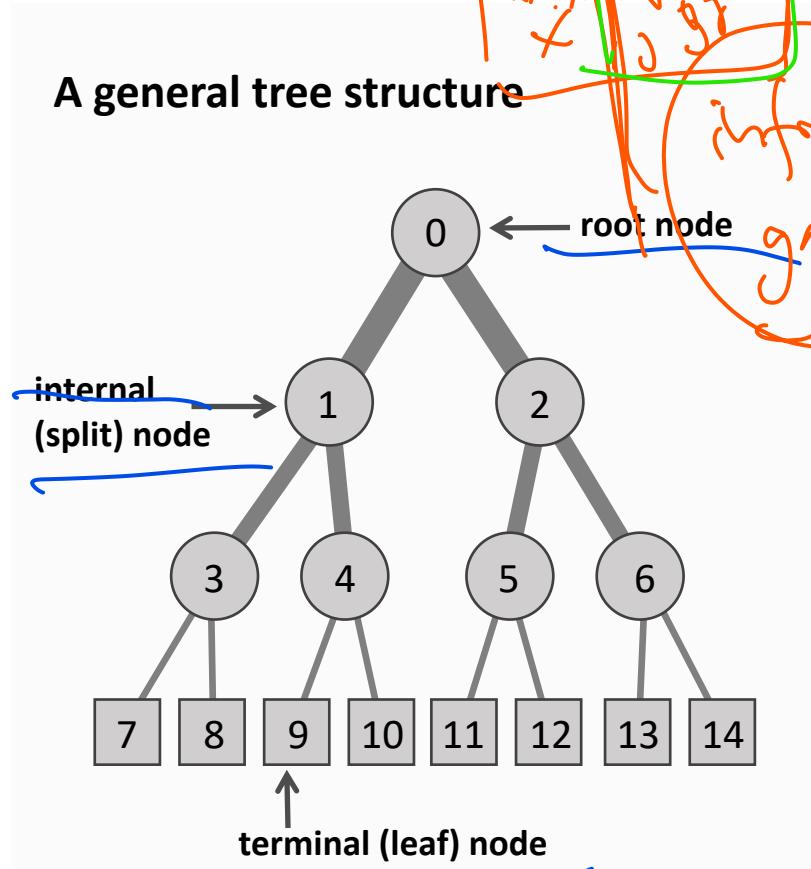


Random Forests

Breiman L. Random forests. Machine Learning 2001;45(1):5-32
<http://stat-www.berkeley.edu/users/breiman/RandomForests/>



Decision Trees and Decision Forests

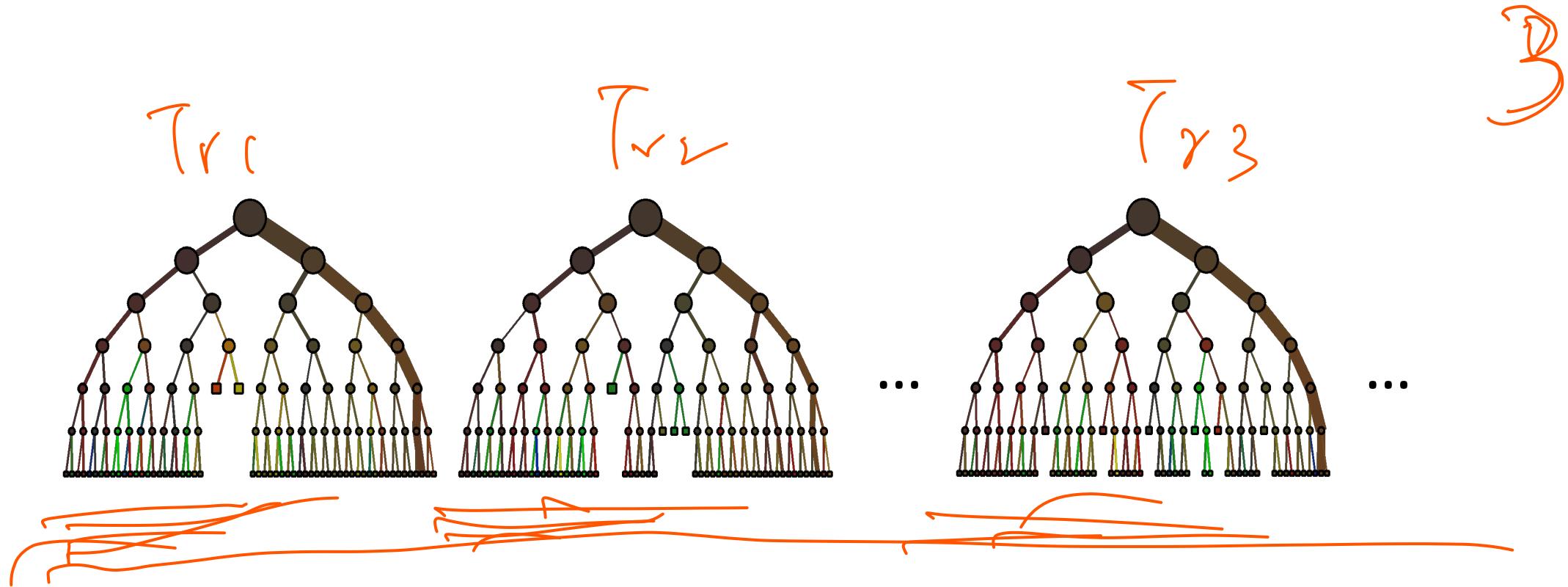


A forest is an ensemble of trees. The trees are all slightly different from one another

ensemble

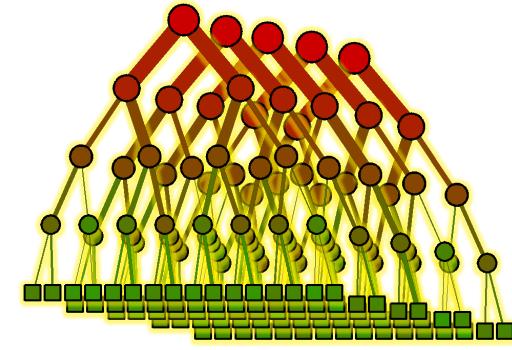
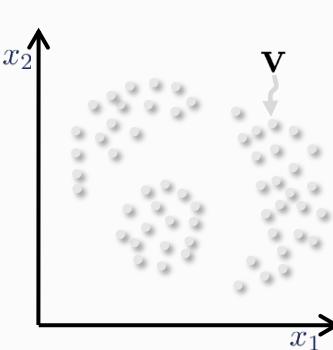
trained

Decision Tree Training (Offline)



**How many trees?
How different?
How to fuse their outputs?**

Random Forest Model



Basic notation

Input data point

e.g. $\mathbf{v} = (x_1, \dots, x_d) \in \mathbb{R}^d$

Collection of feature responses x_i . $d=?$

Output/label space

e.g. $\in \{c_k\}$? \mathbb{R} ?

Categorical, continuous?

Feature response selector

ϕ

Features can be e.g. wavelets? Pixel intensities? Context?

Forest model

tree

Node test parameters

$\theta \in \mathcal{T}$

*Parameters related to each split node:
i) which features, ii) what geometric primitive, iii) thresholds.*

Node objective function (train.) e.g. $I = I(\mathcal{S}_j, \theta)$

The “energy” to be minimized when training the j -th split node

Node weak learner

e.g. $h(\mathbf{v}, \theta_j) \in \{\text{true, false}\}$

The test function for splitting data at a node j .

Leaf predictor model

e.g. $p(c|\mathbf{v})$

Point estimate? Full distribution?

Randomness model (train.)

e.g. 1. Bagging,
2. Randomized node optimization

How is randomness injected during training? How much?

Stopping criteria (train.)

e.g. max tree depth = D

When to stop growing a tree during training

ensemble

Forest size

T

Total number of trees in the forest

The ensemble model

e.g. $p(c|\mathbf{v}) = \frac{1}{T} \sum_t p_t(c|\mathbf{v})$

How to compute the forest output from that of individual trees?

Randomness Model

1) Bagging (randomizing the training set)

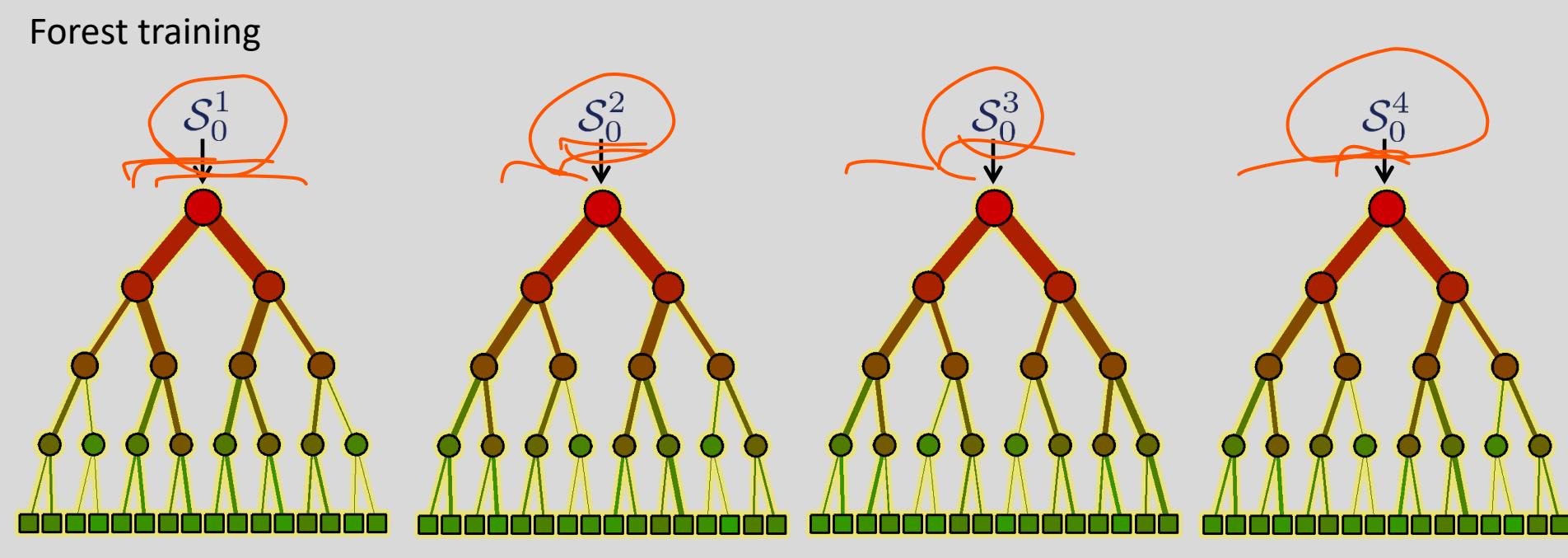
\mathcal{S}_0

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

data



Randomness Model

Features

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

 $\rho = |\mathcal{T}_j|$

Randomness control parameter.

For $\rho = |\mathcal{T}|$ no randomness and maximum tree correlation.

For $\rho = 1$ max randomness and minimum tree correlation.

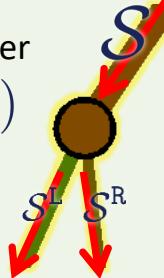
Node training

Node weak learner

$$h(\mathbf{v}, \theta_j)$$

Node test params

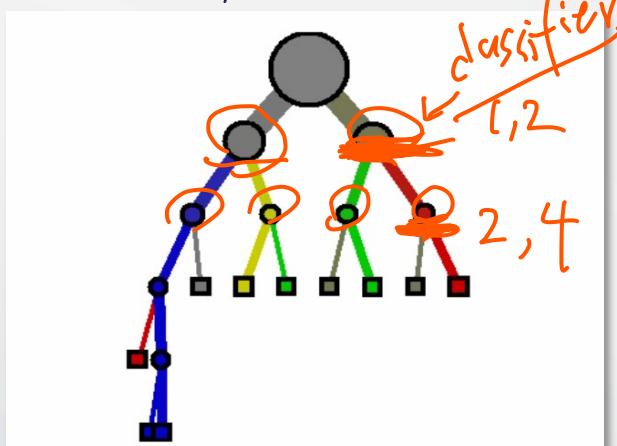
$$\theta \in \mathcal{T}_j$$



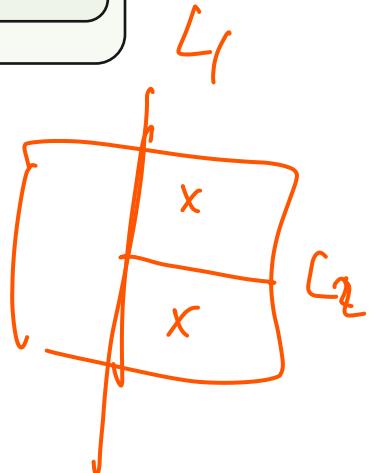
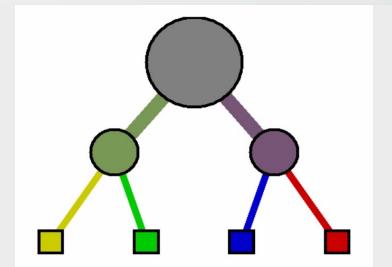
- 1 { color
- 2 size
- 3 feature
- 4 object type

The effect of ρ

Small value of ρ ; little tree correlation.

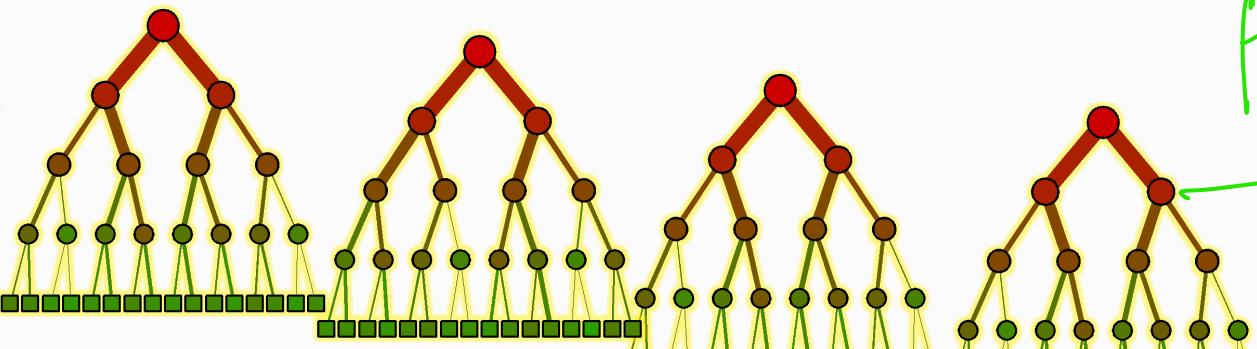


Large value of ρ ; large tree correlation.



The Ensemble Model

An example forest to predict continuous variables



$$P(y|X)$$

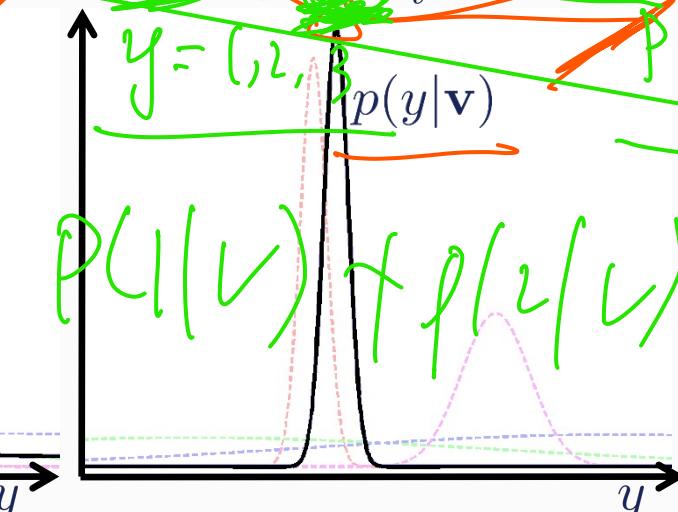
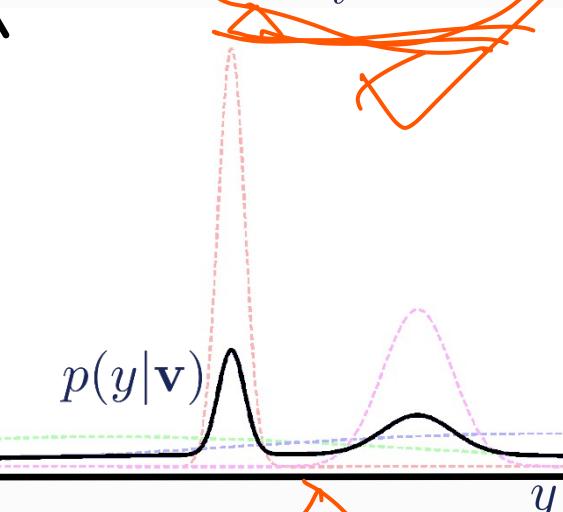
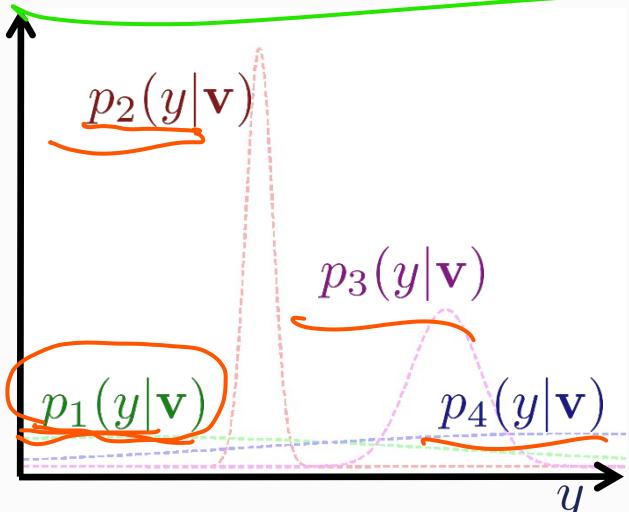
$$\sum P(j|X)$$

$$y = 1$$

$$p_1(y|v) + p_2(y|v) + \dots$$

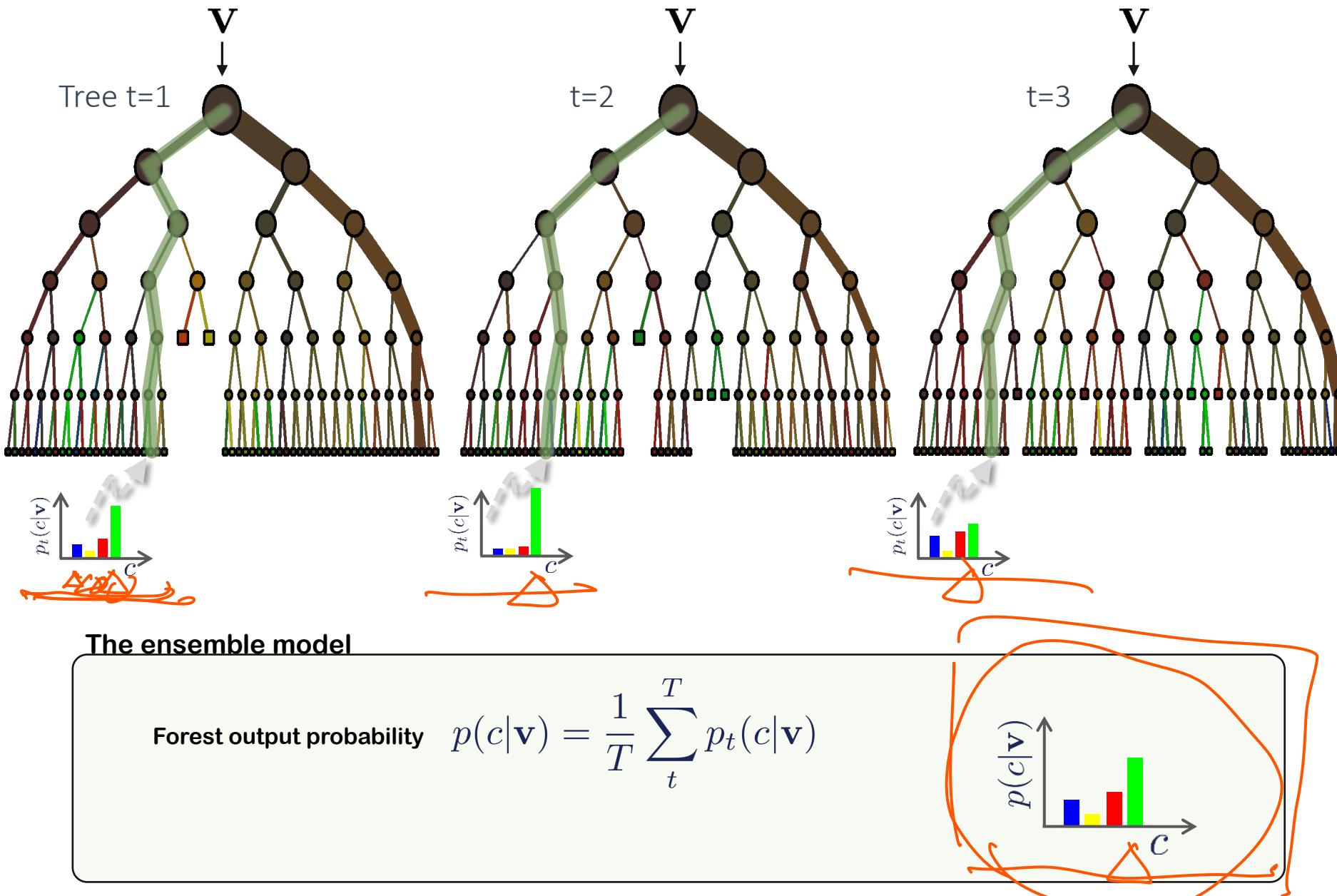
$$p(y|v) = \frac{1}{T} \sum_t^T p_t(y|v)$$

$$p(y|v) = \frac{1}{Z} \prod_t^T p_t(y|v)$$

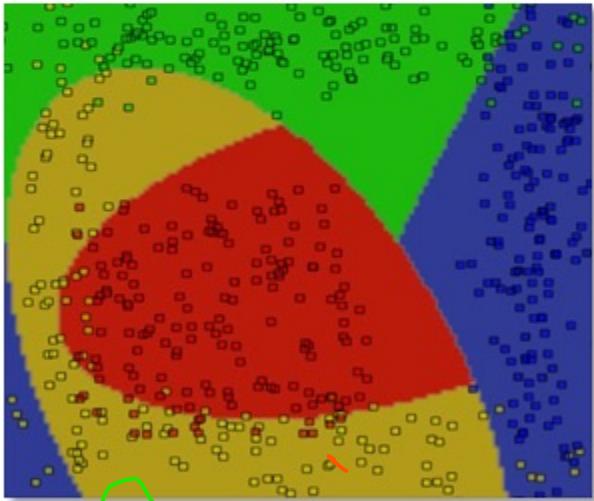
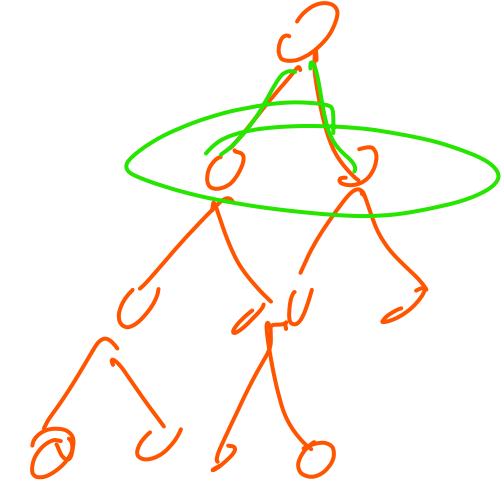
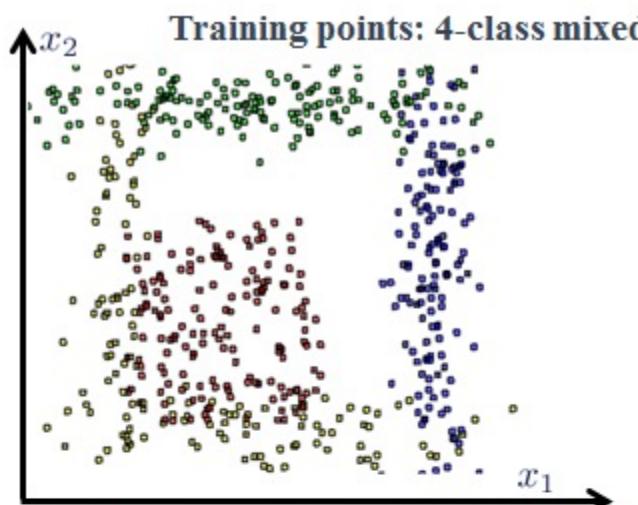
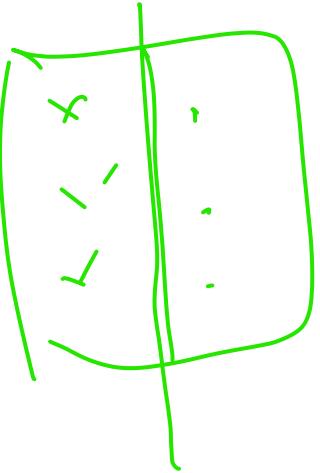


$$p(y|v) = p_1(y|v) + p_2(y|v) + \dots + p_4(y|v)$$

Classification Forest: Ensemble Model

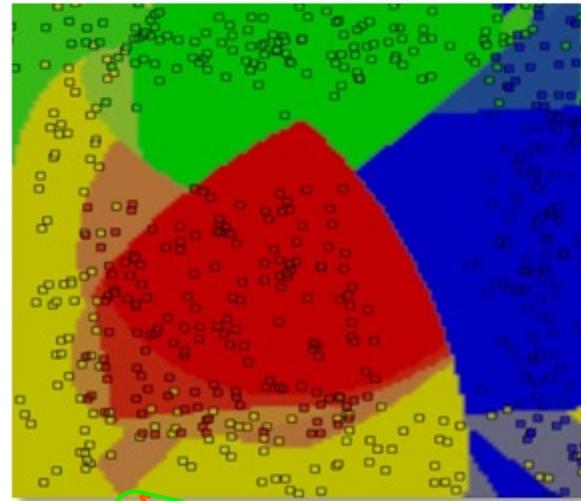


Effect of Tree Depth

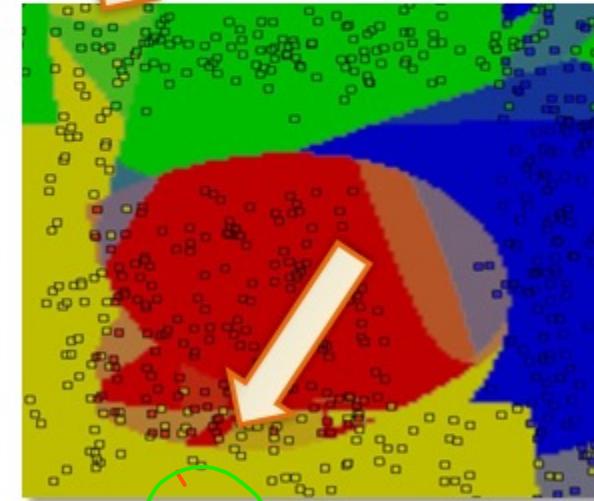


$T=200, D=3, w. l. =$
conic

underfitting



$T=200, D=6, w. l. =$
conic

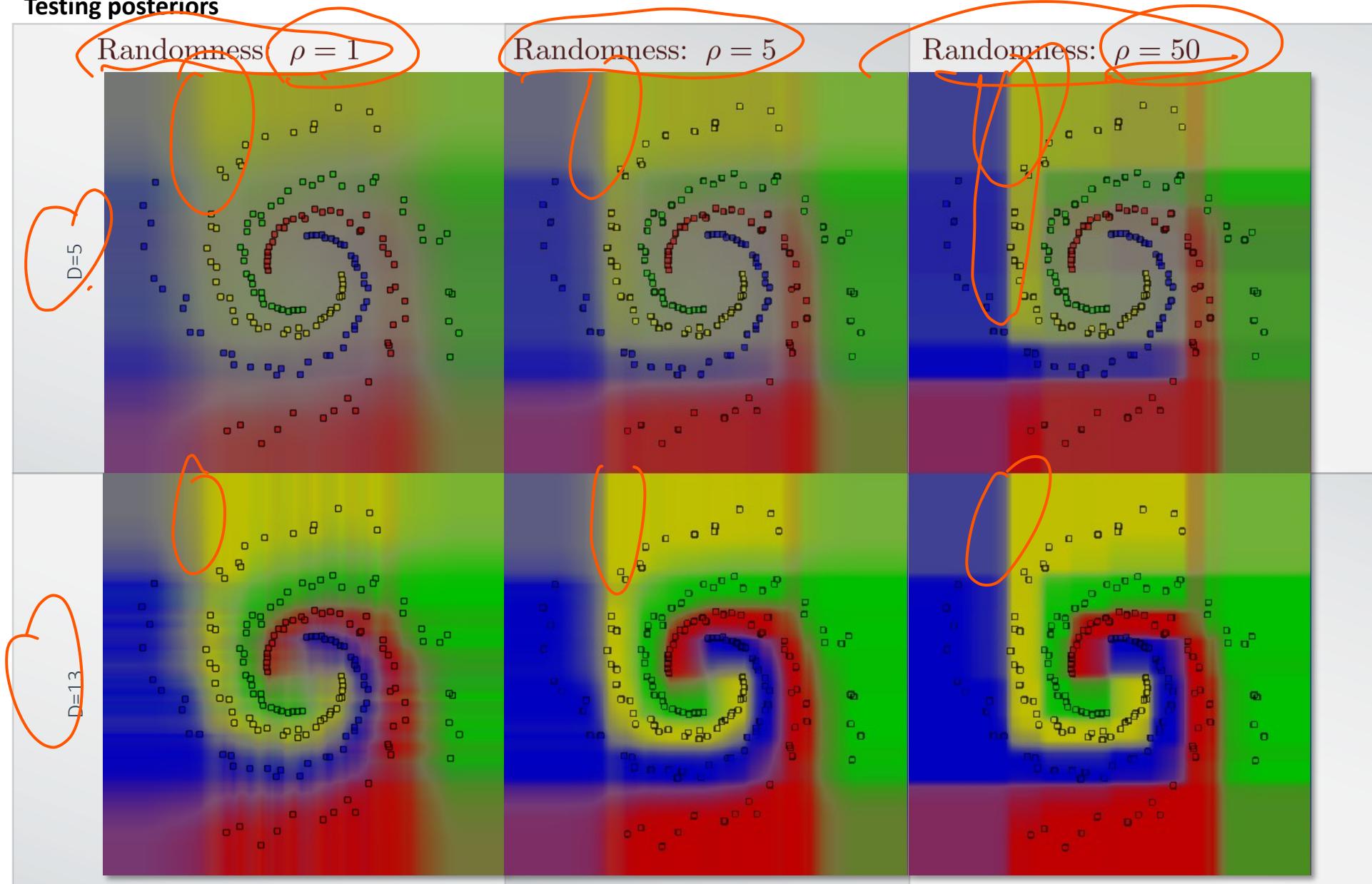


$T=200, D=15, w. l. =$ conic

max tree depth, D
overfitting

Effect of Weak Learner Model and Randomness

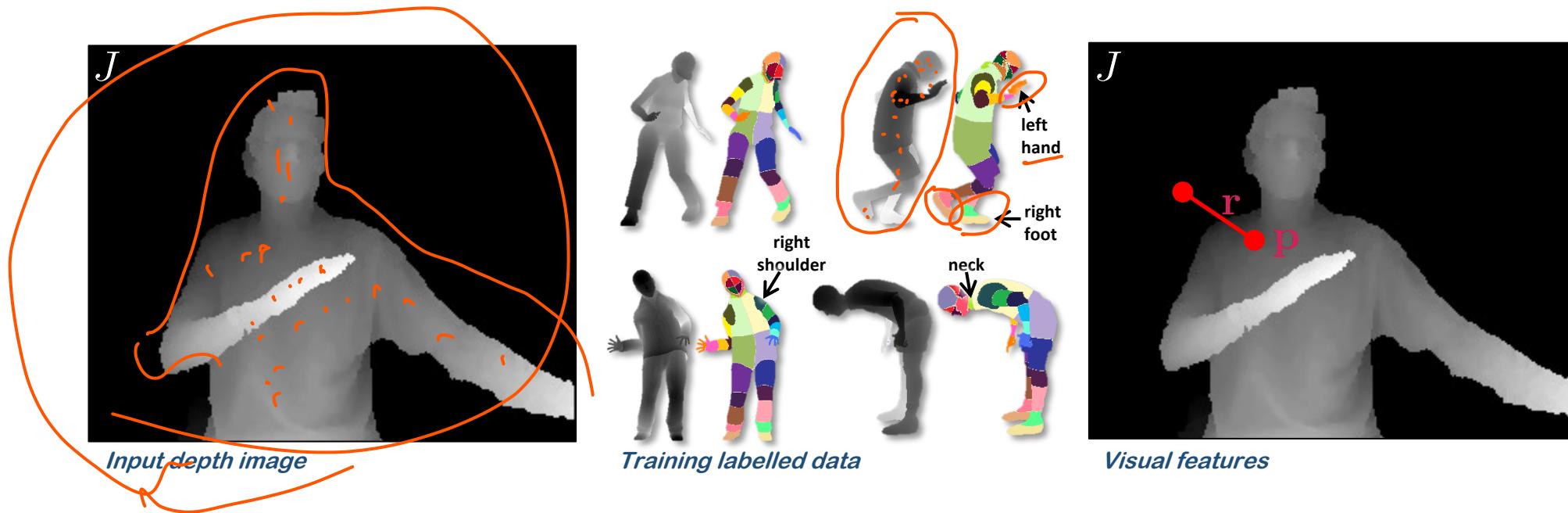
Testing posteriors



Weak learner: axis aligned

Parameters: T=400 predictor model = prob.

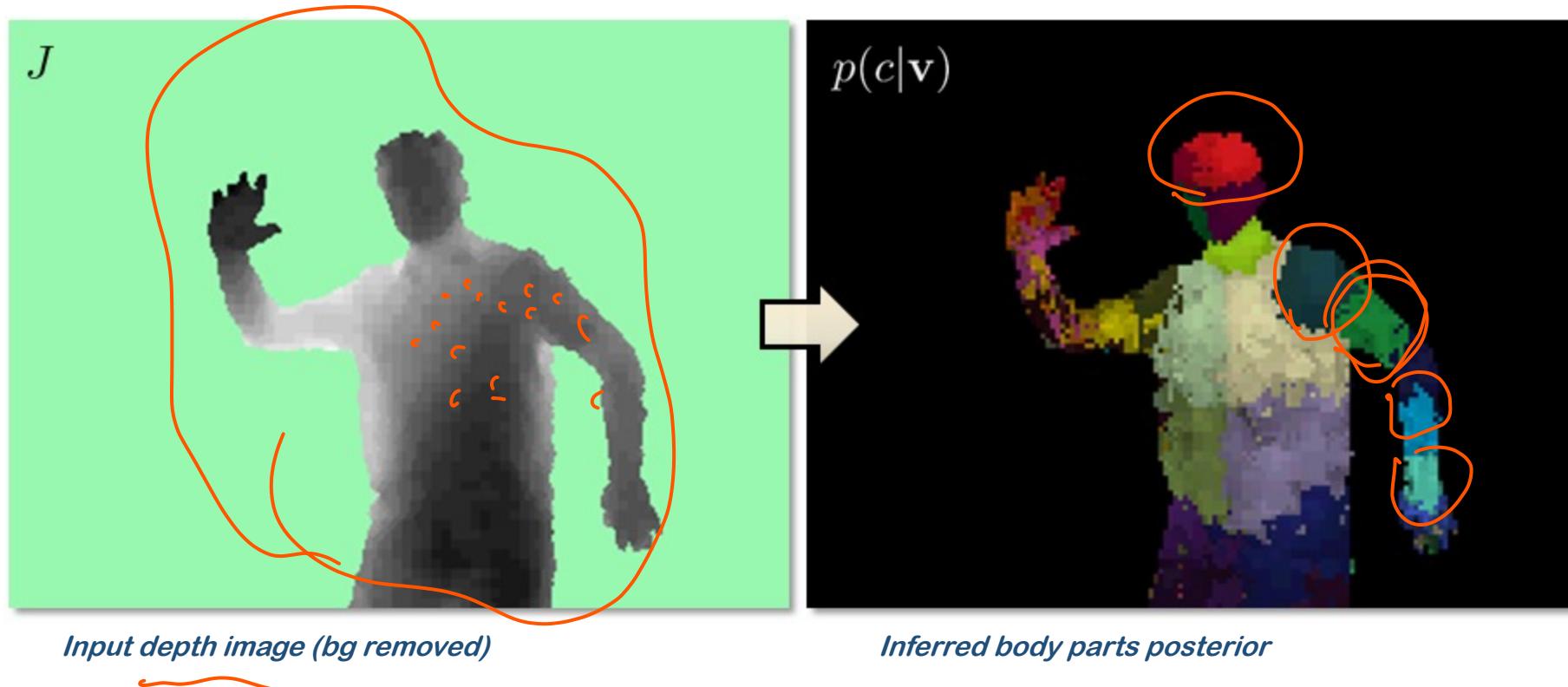
Body tracking in Microsoft Kinect for XBox 360



Classification forest

| | | | |
|------------------------|---|--------------------|---|
| Labels are categorical | $c \in \{l.\text{hand}, r.\text{hand}, \text{head}, \dots\}$ | Objective function | $I = H(\mathcal{S}_j) - \sum_{i=\text{L,R}} \frac{ \mathcal{S}_j^i }{ \mathcal{S}_j } H(\mathcal{S}_j^i)$ |
| Input data point | $\mathbf{p} \in \mathbb{R}^2$ | Node parameters | $\theta = (\mathbf{r}, \tau)$ |
| Visual features | $\mathbf{v}(\mathbf{p}) = (x_1, \dots, x_i, \dots, x_d) \in \mathbb{R}^d$ | Node training | $\theta_j = \arg \max_{\theta \in \mathcal{T}_j} I(\mathcal{S}_j, \theta)$ |
| Feature response | $x_i = J(\mathbf{p}) - J\left(\mathbf{p} + \frac{\mathbf{r}_i}{J(\mathbf{p})}\right)$ | Weak learner | $h(\mathbf{v}, \theta) = [\phi(\mathbf{v}, \mathbf{r}) > \tau]$ |
| Predictor model | $p(c \mathbf{v})$ | | |

Body tracking in Microsoft Kinect for XBox 360



Advantages of Random Forests

- Very high accuracy – not easily surpassed by other algorithms
- Efficient on large datasets
- Can handle thousands of input variables without variable deletion
- Effective method for estimating missing data, also maintains accuracy when a large proportion of the data are missing
- Robust to label noise
- Can be used in clustering, locating outliers and semi-supervised learning

Boosting

Boosting Resources

- Slides Credits:
 - Tutorial by Rob Schapire
 - Tutorial by Yoav Freund
 - Slides by Carlos Guestrin
 - Tutorial by Paul Viola
 - Tutorial by Ron Meir
 - Slides by Aurélie Lemmens
 - Slides by Zhuowen Tu

Code

- Antonio Torralba (Object detection)
 - <http://people.csail.mit.edu/torralba/shortCourseRLOC/boosting/boosting.html>
- GML AdaBoost
 - <http://graphics.cs.msu.ru/ru/science/research/machinelearning/adaboosttoolbox>

Boosting

- Invented independently by Schapire (1989) and Freund (1990)
 - Later joined forces
- Main idea: train a strong classifier by combining weak classifiers
 - Practically useful
 - Theoretically interesting

Boosting

weak classifier
hypothesis

- Given a set of weak learners, run them multiple times on (reweighted) training data, then let learned classifiers vote
- At each iteration t :
 - Weight each training example by how incorrectly it was classified
 - Learn a hypothesis – h_t
 - The one with the smallest error
 - Choose a strength for this hypothesis – α_t
- Final classifier: weighted combination of weak learners

Learning from Weighted Data

- Sometimes not all data points are equal
 - Some data points are more equal than others
- Consider a weighted dataset
 - $D(i)$ – weight of i^{th} training example (x_i, y_i)
 - Interpretations:
 - i^{th} training example counts as $D(i)$ examples
 - If I were to “resample” data, I would get more samples of “heavier” data points
- Now, in all calculations the i^{th} training example counts as $D(i)$ “examples”

$$D(i) = 2$$

Definition of Boosting

- Given training set $(\underline{x_1}, \underline{y_1}), \dots, (\underline{x_m}, \underline{y_m})$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- For $t=1, \dots, T$
 - construct distribution D_t on $\{1, \dots, m\}$
 - find weak hypothesis
 - $h_t: X \rightarrow \{-1, +1\}$ with small error ϵ_t on D_t
- Output final hypothesis H_{final}

2 weight
1) how large \pm
2) $\alpha_i h_i$

AdaBoost

- Constructing D_{t+1}
 - $D_1 = 1/m$
 - Given D_t and h_t :

where Z_t is a normalization constant

$$f \quad 1.2$$

- Final hypothesis:

$$H_{\text{final}}(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right)$$

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

$$Z_t = \sum_{i=1}^m D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) > 0$$

$$\begin{array}{c}
 \epsilon_t \downarrow \\
 \alpha_t \uparrow
 \end{array}$$

$$\begin{array}{c}
 \alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3 \\
 \downarrow \quad \downarrow \quad \downarrow \\
 34
 \end{array}$$

The AdaBoost Algorithm

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

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

- Train base learner using distribution D_t .
- Get base classifier $h_t : X \rightarrow \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

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

The AdaBoost Algorithm

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

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

- Train base learner using distribution D_t .
- Get base classifier $h_t : X \rightarrow \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

with minimum ϵ_t

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

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

The AdaBoost Algorithm

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

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

- Train base learner using distribution D_t .
- Get base classifier $h_t : X \rightarrow \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

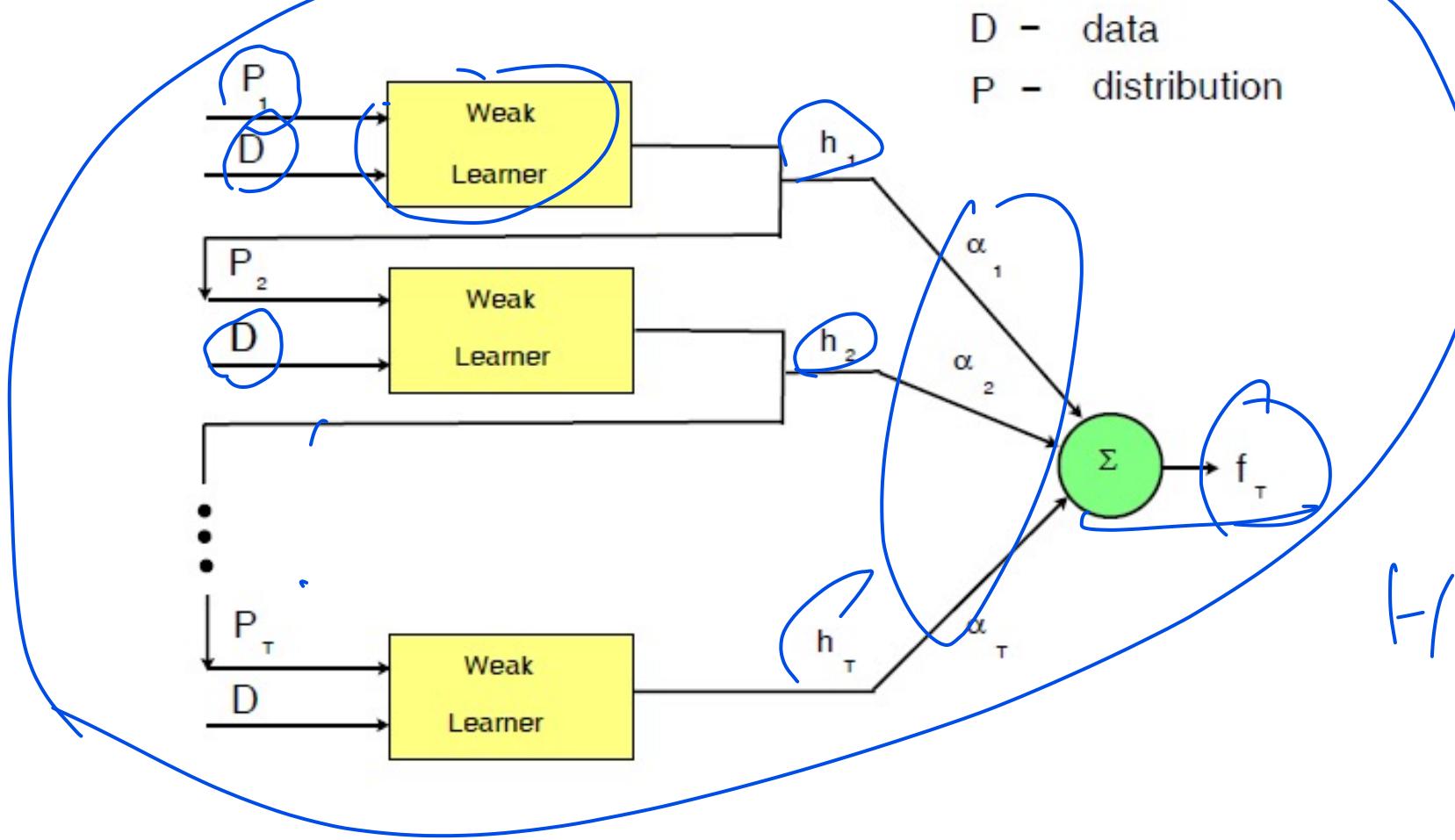
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

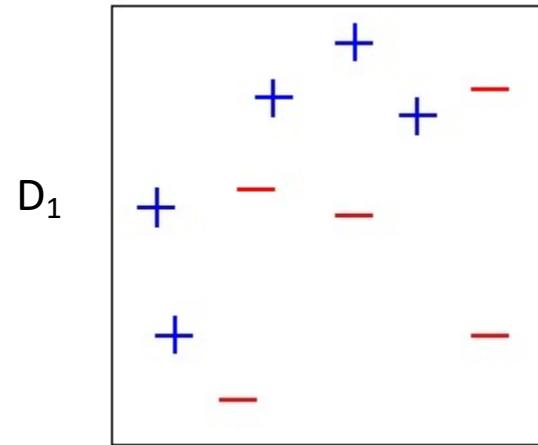
$$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i]$$

$$\epsilon_t = \frac{1}{\sum_{i=1}^n D_t(i)} \sum_{i=1}^m D_t(i) \delta(h_t(x_i) \neq y_i)$$

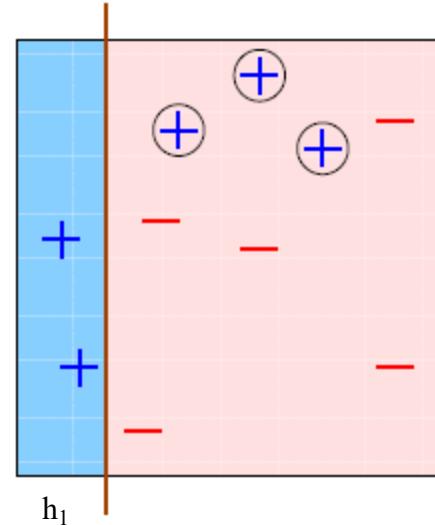
The AdaBoost Algorithm



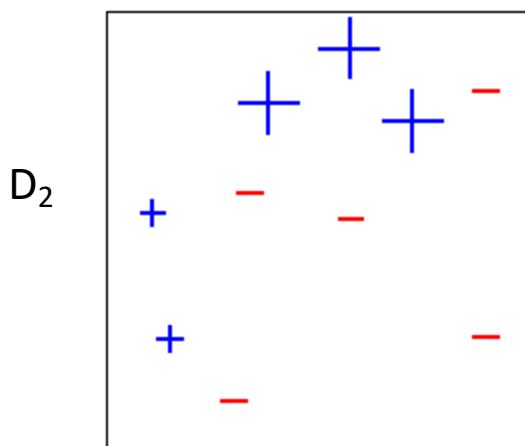
Toy Example



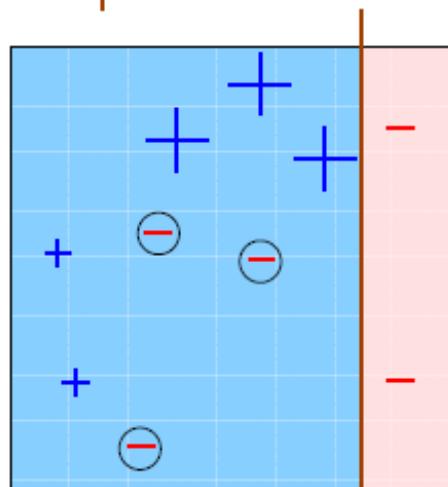
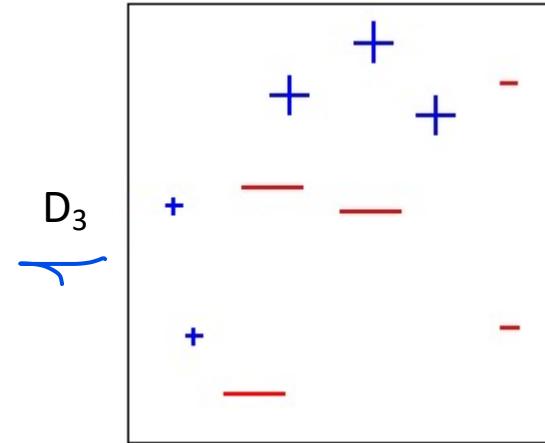
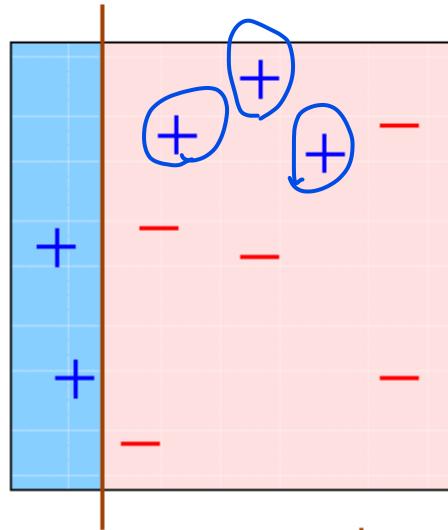
Toy Example: Round 1



$$\begin{aligned}\varepsilon_1 &= 0.3 \\ \alpha_1 &= 0.42\end{aligned}$$



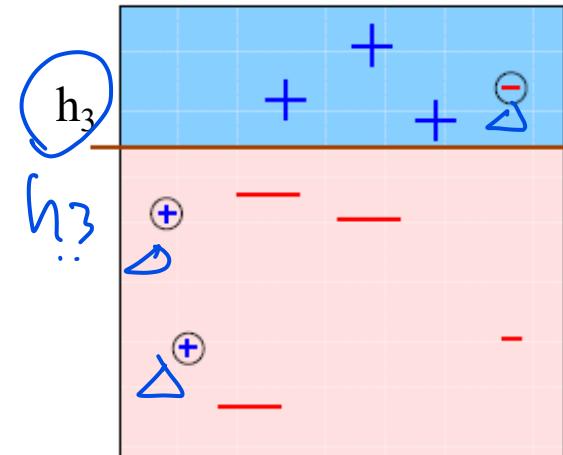
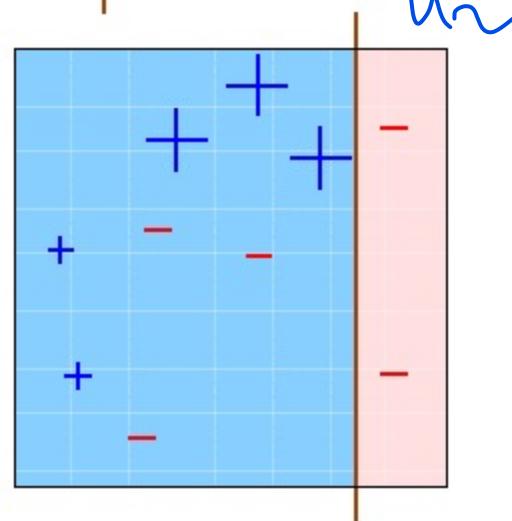
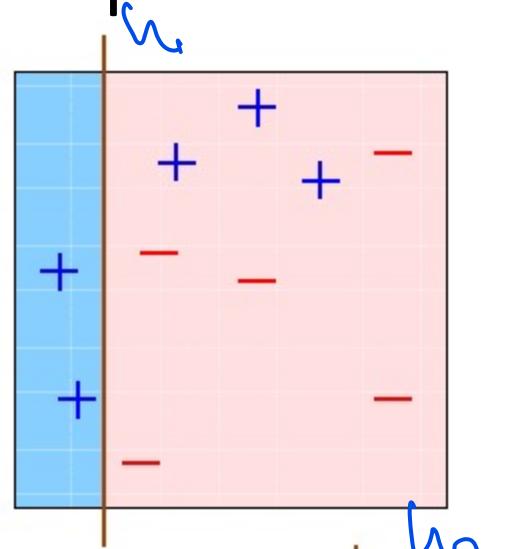
Toy Example: Round 2



$$\begin{aligned}\varepsilon_2 &= 0.21 \\ \alpha_2 &= 0.65\end{aligned}$$

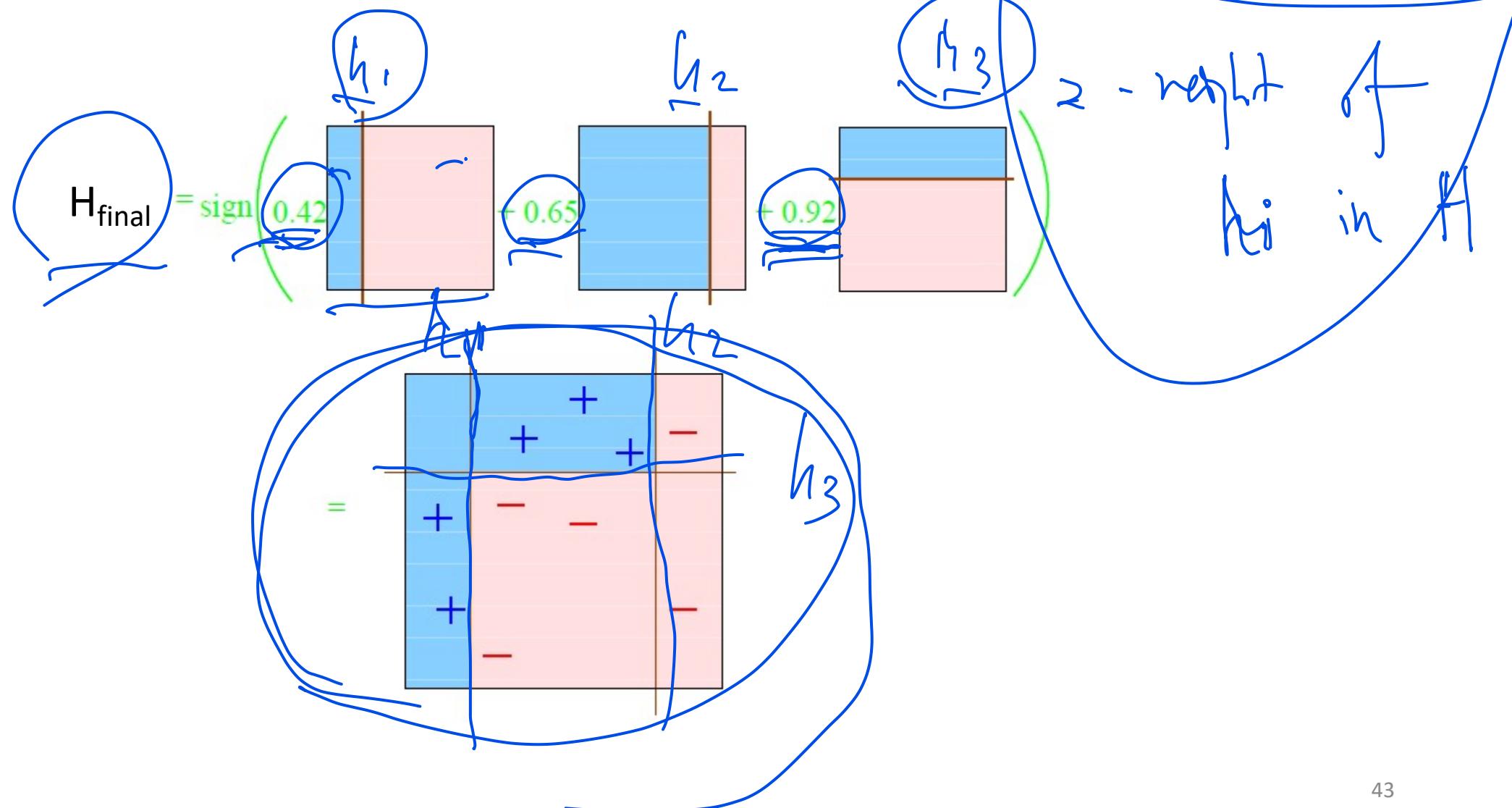
h_2

Toy Example: Round 3



h_1
 h_2
set
iteration = 3

Toy Example: Final Hypothesis



Applications of Boosting

Viola-Jones Face Detector

Real time face detection using a classifier cascade [Viola and Jones, 2001 and 2004]

The Classical Face Detection Process



Classifier is Trained on Labeled Data

- Training Data
 - 5000 faces
 - All frontal
 - 10^8 non faces
 - Faces are normalized
 - Scale, translation
- Many variations
 - Across individuals
 - Illumination
 - Pose (rotation both in plane and out)

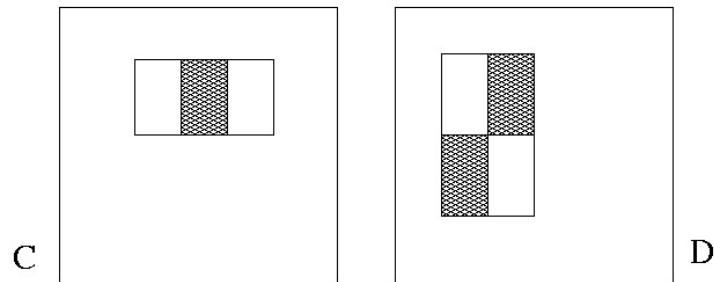
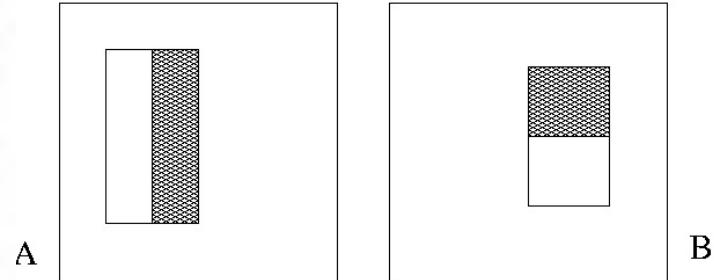


Classifier Cascade (Viola-Jones)

- For real problems results are only as good as the features used...
 - This is the main piece of ad-hoc (or domain) knowledge
- Rather than the pixels, use a very large set of simple functions
 - Sensitive to edges and other critical features of the image
 - Computed at multiple scales
- Introduce a threshold to yield binary features
 - Binary features seem to work better in practice
 - In general, convert continuous features to binary by quantizing

Boosted Face Detection: Image Features

“Rectangle filters”



$$h_t(x_i) = \begin{cases} \alpha_t & \text{if } f_t(x_i) > \theta_t \\ \beta_t & \text{otherwise} \end{cases}$$

$$C(x) = \theta \left(\sum_t h_t(x) + b \right)$$

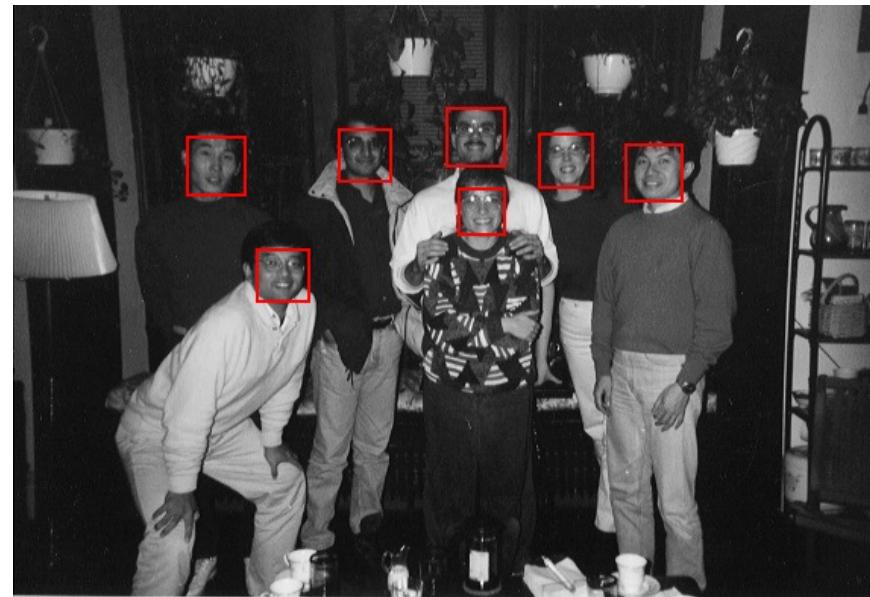
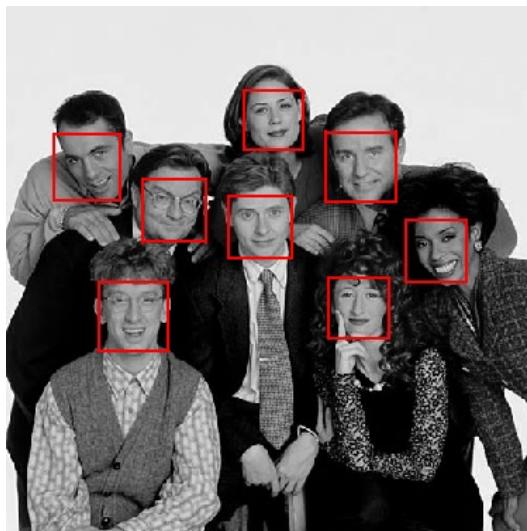
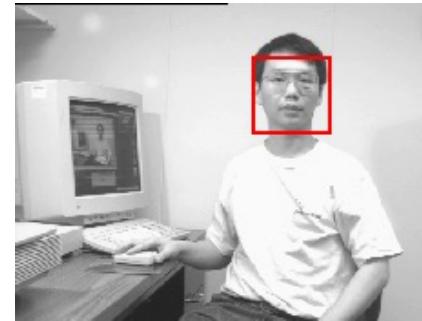
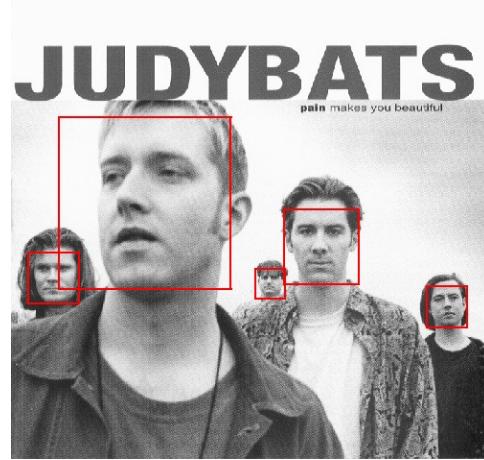
Feature Selection

- For each round of boosting:
 - Evaluate each rectangle filter on each example
 - Sort examples by filter values
 - Select best threshold for each filter
 - Select best filter/threshold (= Feature)
 - Reweight examples

Building Fast Classifiers

- In general, simple classifiers are more efficient, but they are also weaker
- We could define a computational risk hierarchy
 - A nested set of classifier classes
- The training process is reminiscent of boosting...
 - Previous classifiers reweight the examples used to train subsequent classifiers

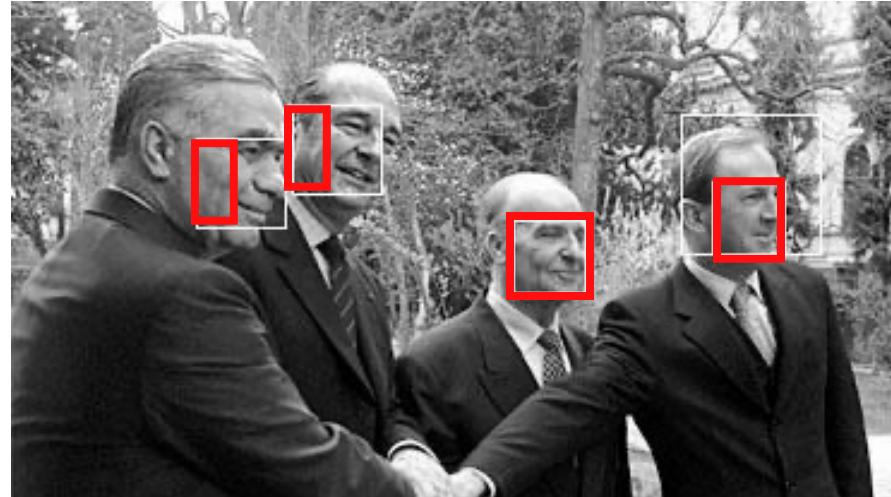
Output of Face Detector on Test Images



Solving other “Face” Tasks

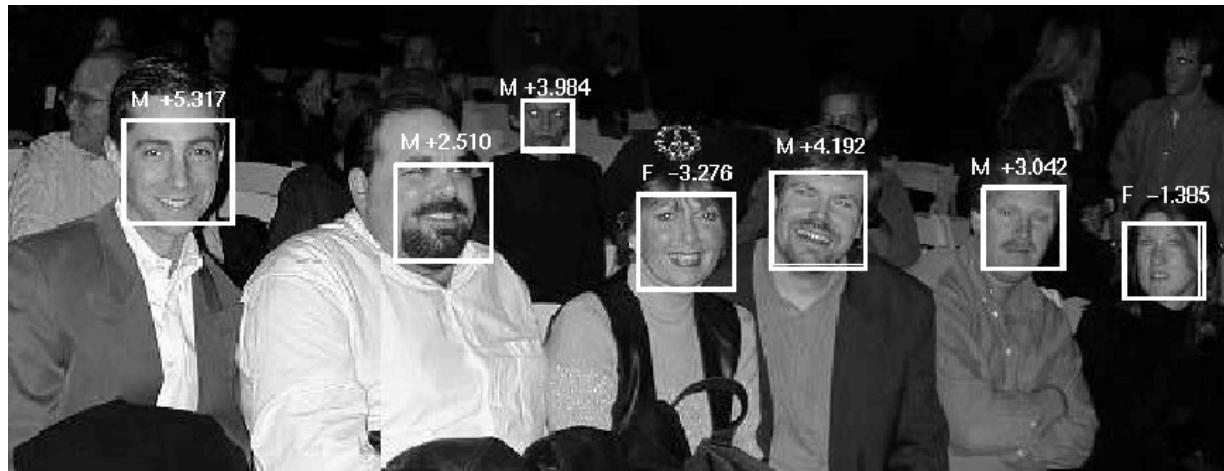


Facial Feature Localization



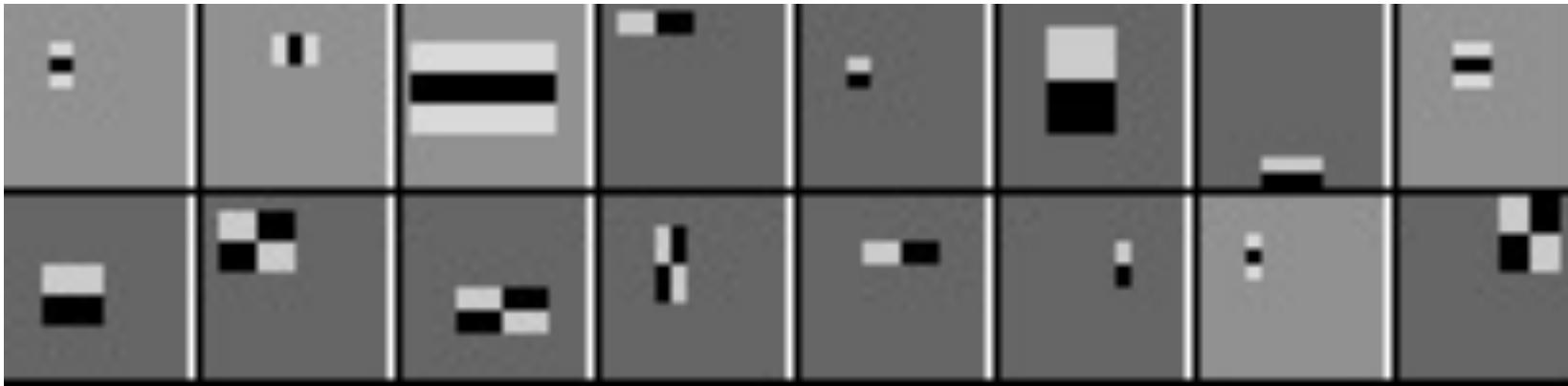
Profile Detection

Demographic Analysis

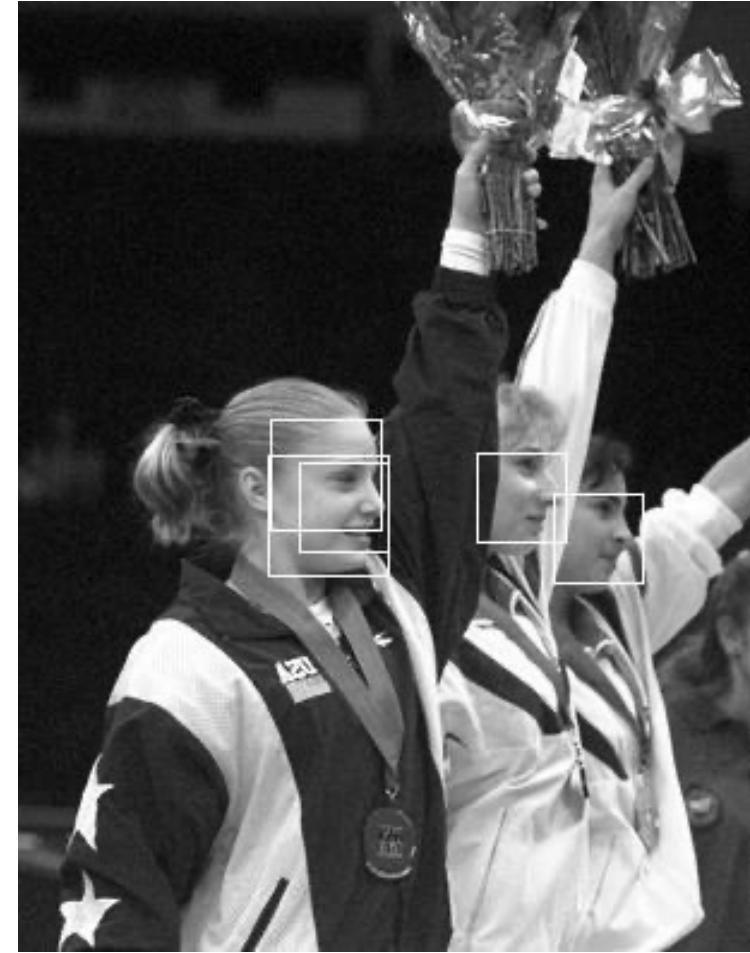


Feature Localization

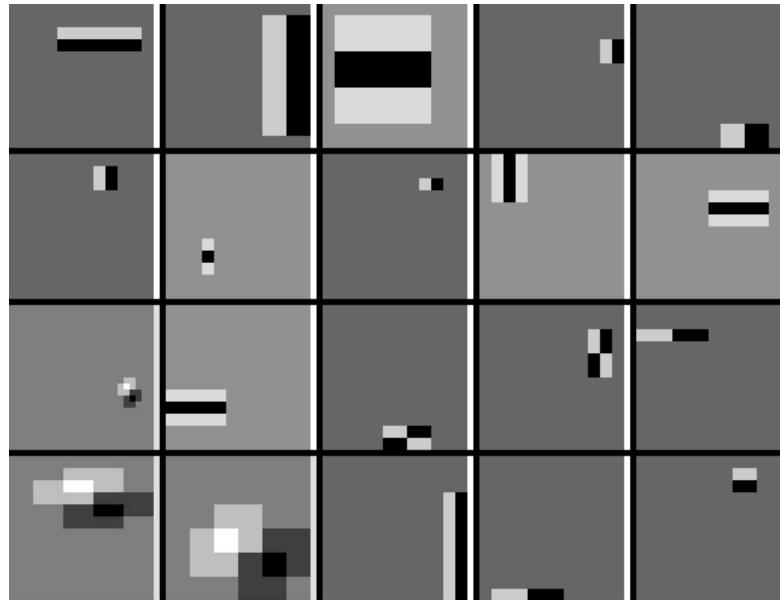
- Learned features reflect the task



Profile Detection



Profile Features



Summary...

- Bagging
 - Random Forest/Decision Forest
- Boosting
 - Viola-Jones Face Detector