Lecture 13: Bagging, Random Forests; Boosting (1)

Dr. Chengjiang Long
Computer Vision Researcher at Kitware Inc.
Adjunct Professor at RPI.
Email: **longc3@rpi.edu**

Progress

Attendence

2.5% 2.5%

62.5%

Homework

15% 5%

Midterm & Final Exams

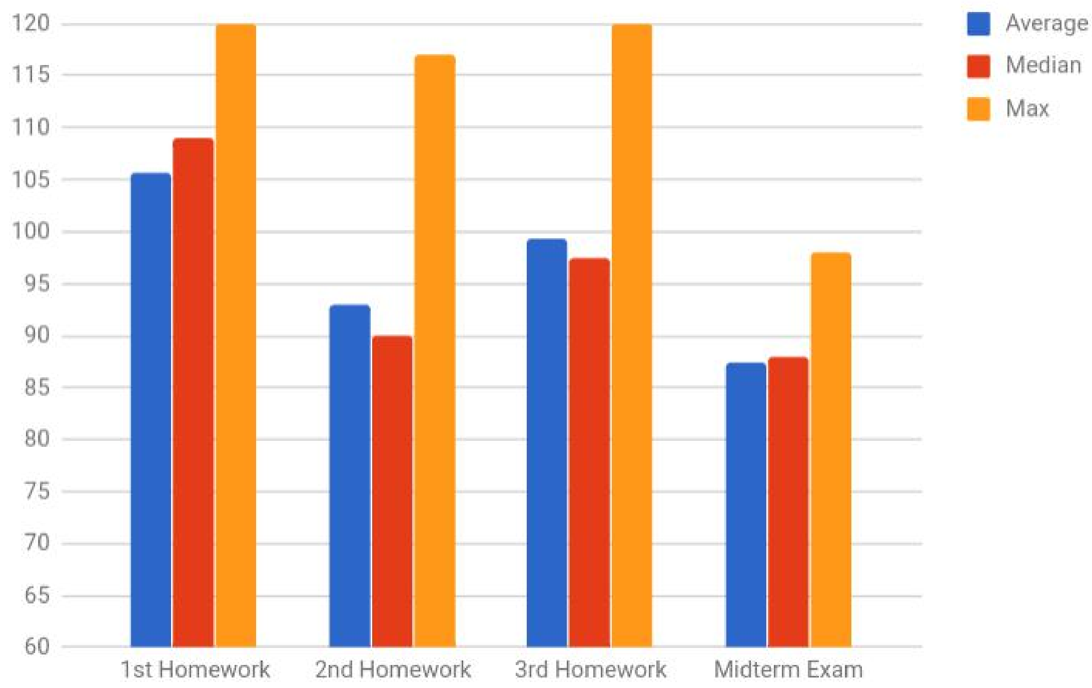
20% 20%

Final Project

35%

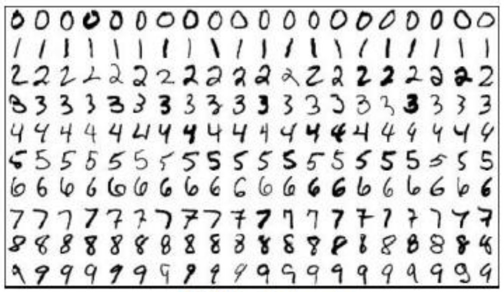
Progress

	Average	Meidan	Max
1st Homework	105.7	109	120
2nd Homework	93	90	117
3rd Homework	99.4	97.5	120
Midterm Exam	87.4	88	98



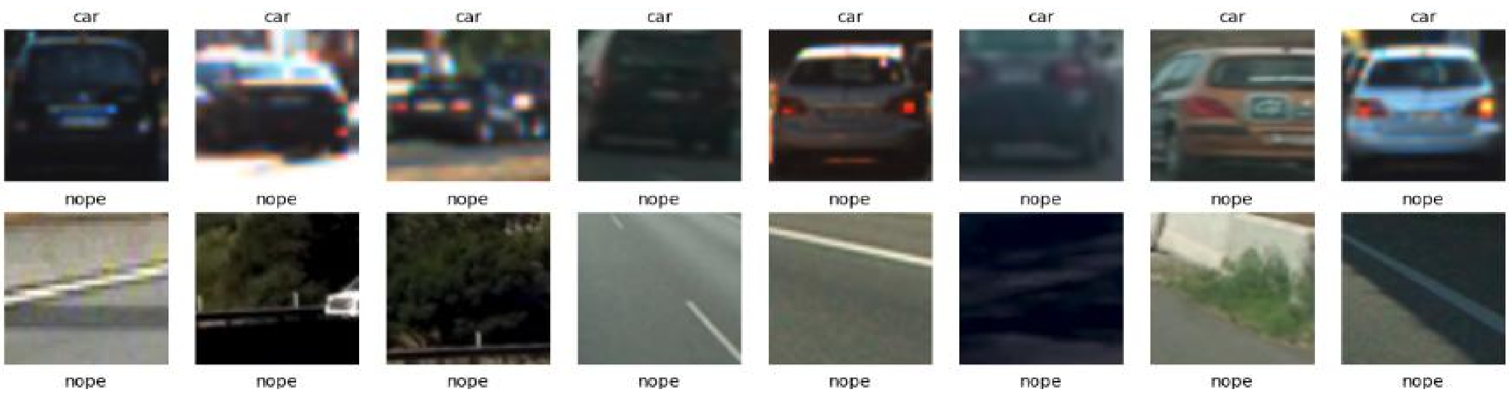
Final Project

- The final project is an open-ended assignment, with the goal of gaining experience applying the techniques presented in class to real-world datasets.



MINST Dataset

LFW dataset



....

Final Project

- Students should work either individually or in groups of 2-3. By 11:59 p.m. on March 23, you will submit a 1–3 paragraph proposal on your project.
- You must do this in order to get full credit for your project, and you must get my approval on it before presentating project proposals to the class on Mar 30, and starting work on your project.
- The proposal presentation should describe the problem you are solving, what data is being used, the proposed technique you are applying in addition to what baseline is used to compare against.
- The final project report should be at least 4 pages and is due on April 30, as well as the data and code.

Final Project - Timeline

- 03/23/2018: 1-3 paragraph proposal for approval.
- 03/30/2018: proposal presentation to class.
- 04/30/2018: submit the data, code and project report.
- 05/01/2018: oral presentation to class.

Outline

- Bagging
- Random Forest
- Boosting

Outline

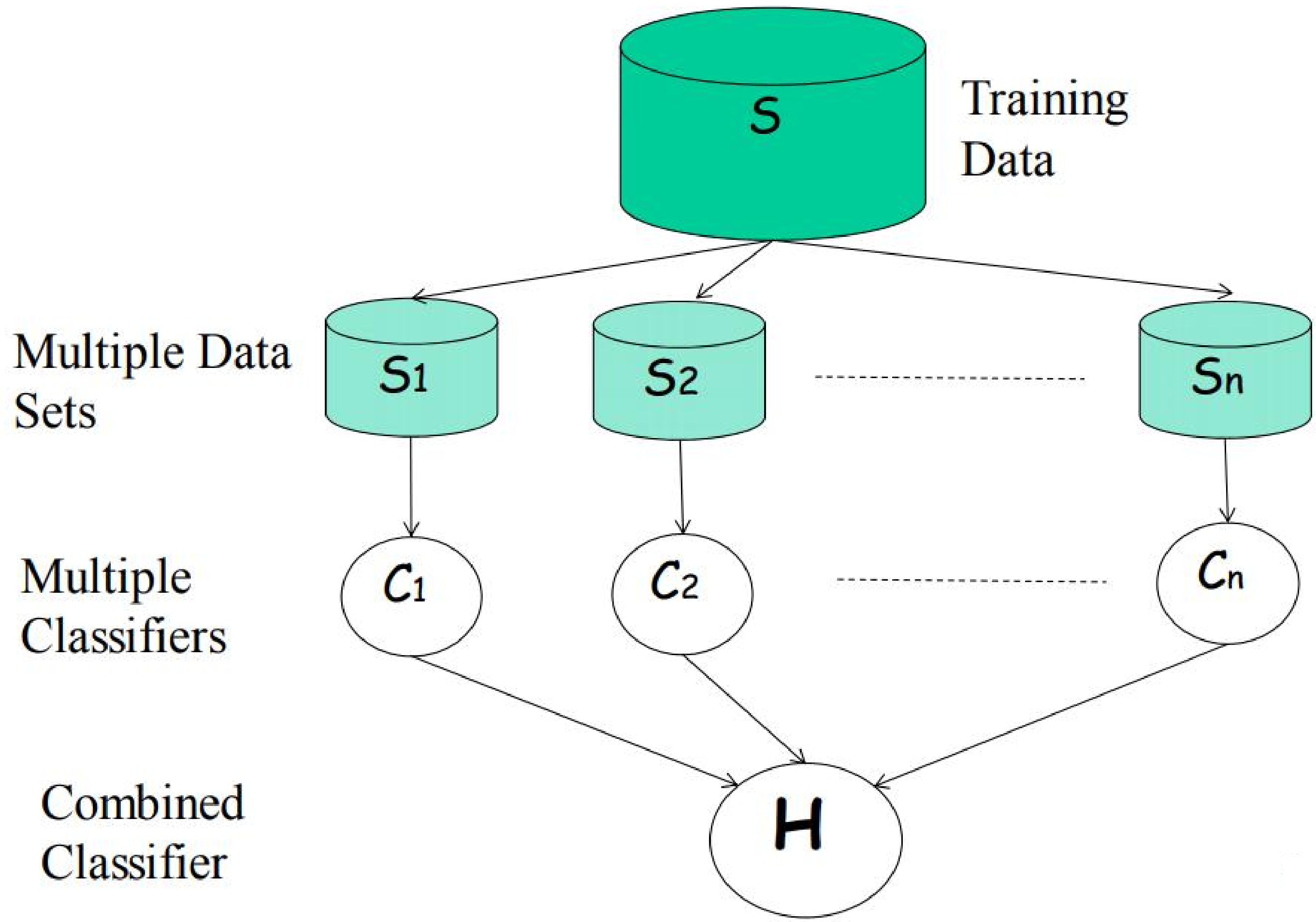
- **Bagging**
- Random Forest
- Boosting

Ensemble Methods

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

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

General Idea



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

Why do they work?

- 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:

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

Bagging

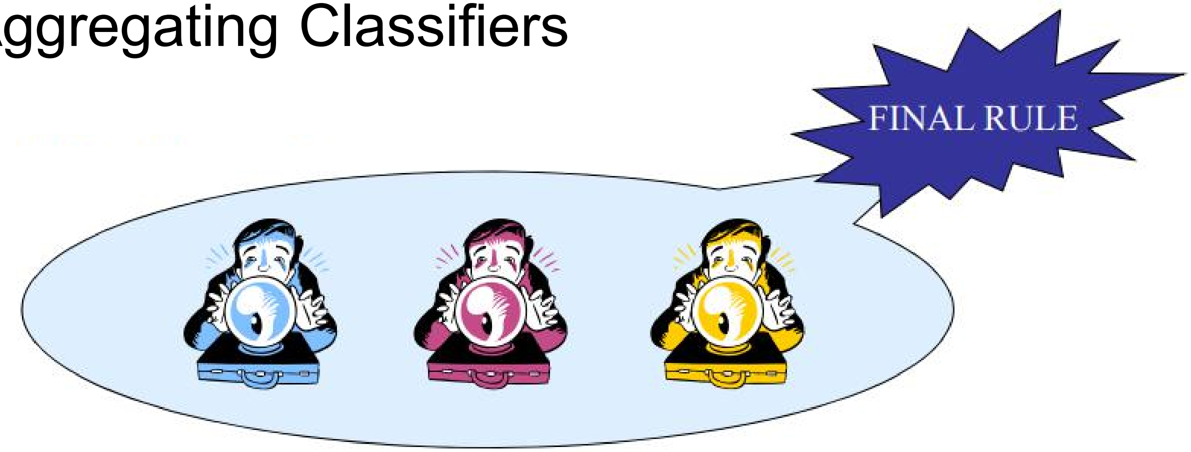
- Training
  - Given a dataset S, at each iteration i, a training set Si is sampled with replacement from S (i.e. bootstraping)
  - A classifier C_i is learned for each S_i
- Classification: given an unseen sample X
  - Each classifier C_i returns its class prediction
  - The bagged classifier H counts the votes and assigns the class with the most votes to X

Bagging

- Bagging works because it reduces variance by voting/averaging
  - In some pathological hypothetical situations the overall error might increase
  - 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

- Bagging and Boosting
  - Aggregating Classifiers



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

Bagging

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

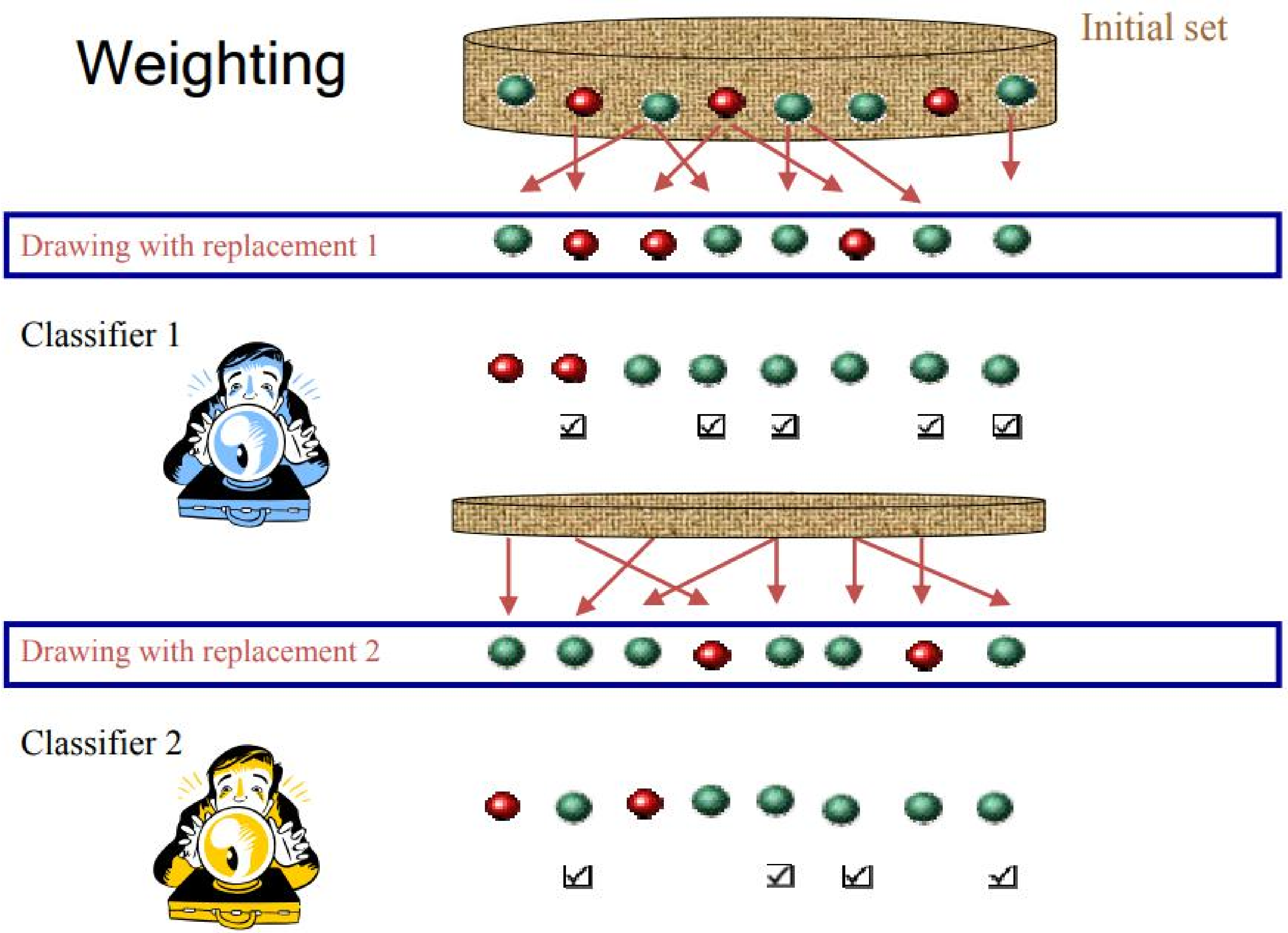
The Bagging Algorithm

- B bootstrap samples
- From which we derive:
  - *B* Classifiers $\in \{-1, 1\}: c^1, c^2, c^3, \ldots, c^B$
  - *B* Estimated probabilities $\in [0, 1]: p^1, p^2, p^3, \ldots, p^B$

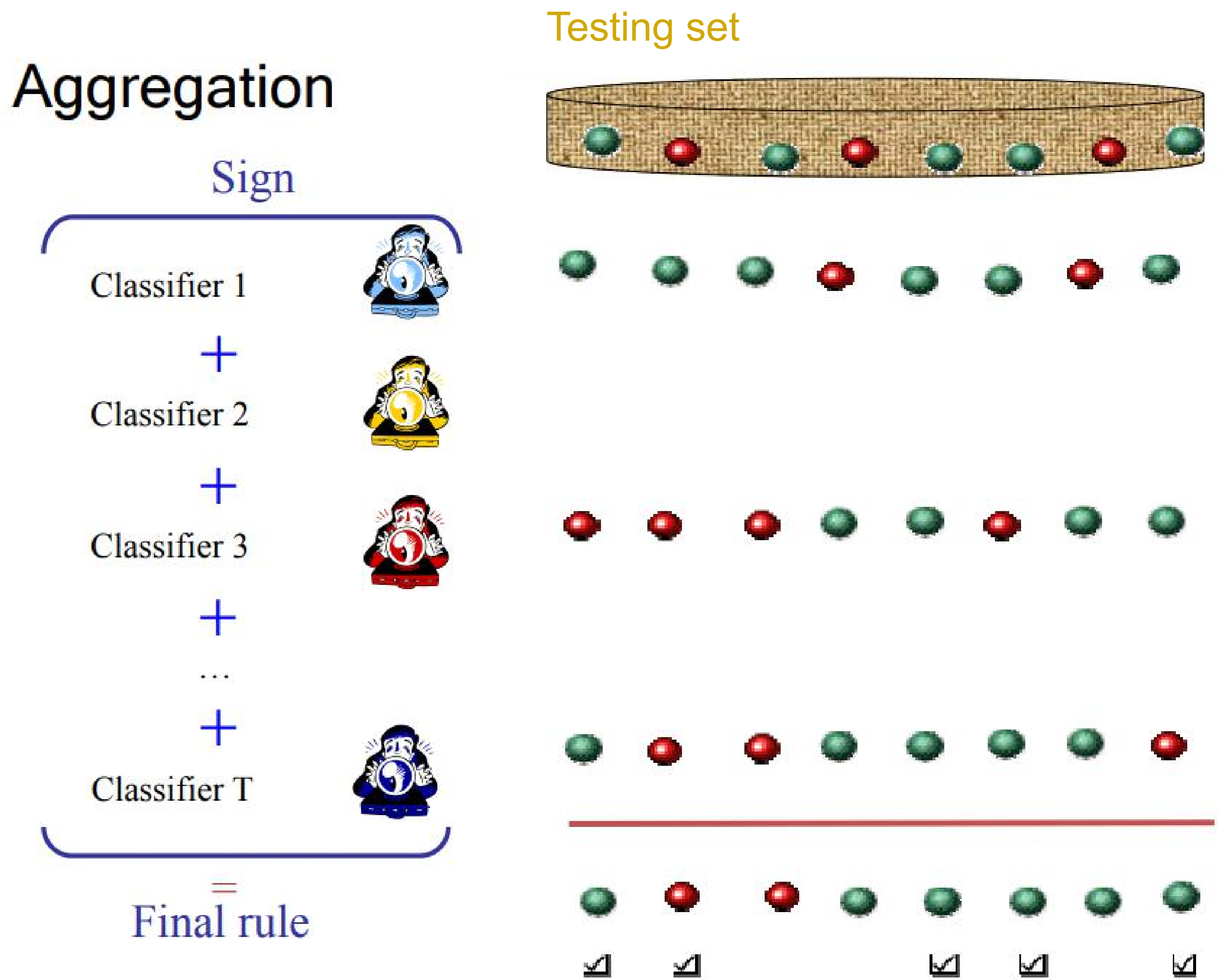
- The aggregate classifier becomes:

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

Example (1)


Weighting
Initial set
Drawing with replacement 1
Classifier 1
Drawing with replacement 2
Classifier 2

Example (2)

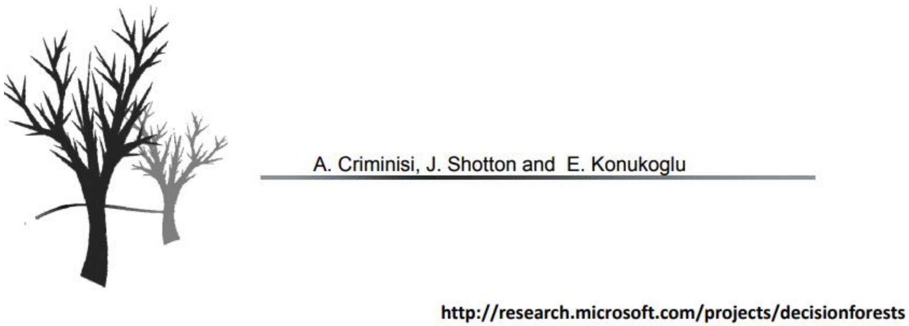

Testing set
Aggregation
Sign
Classifier 1
+
Classifier 2
+
Classifier 3
+
...
+
Classifier T
=
Final rule

Outline

- Bagging
- **Random Forest**
- Boosting

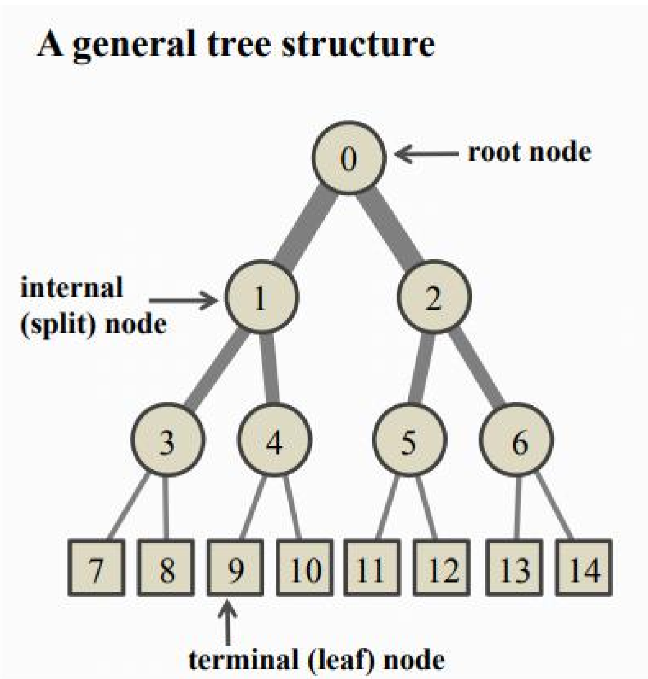
Random Forests

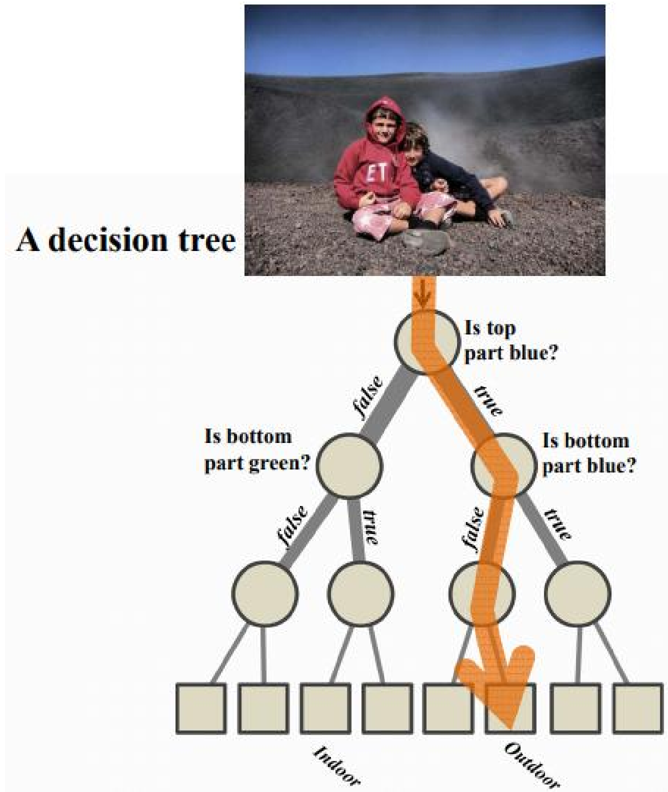
- Breiman L. Random forests. Machine Learning 2001;45(1):5−32
- http://statwww.berkeley.edu/users/breiman/RandomForests/
- For computer vision and medical image analysis



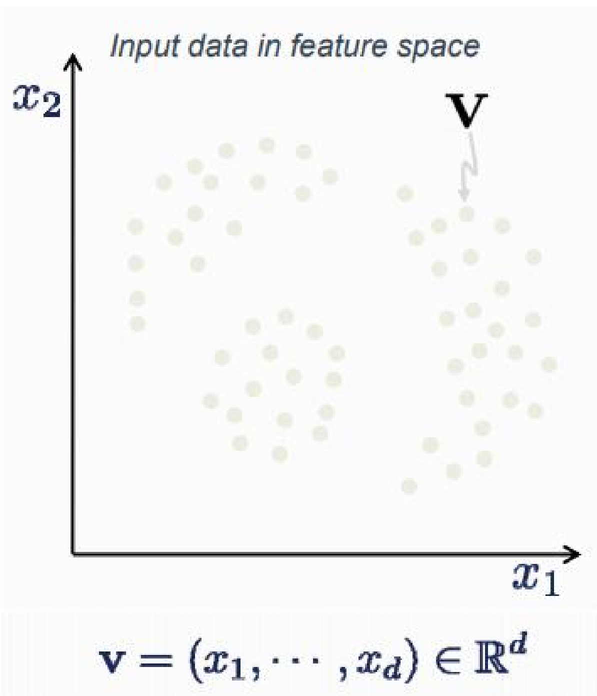
Decision Trees and Decision Forests

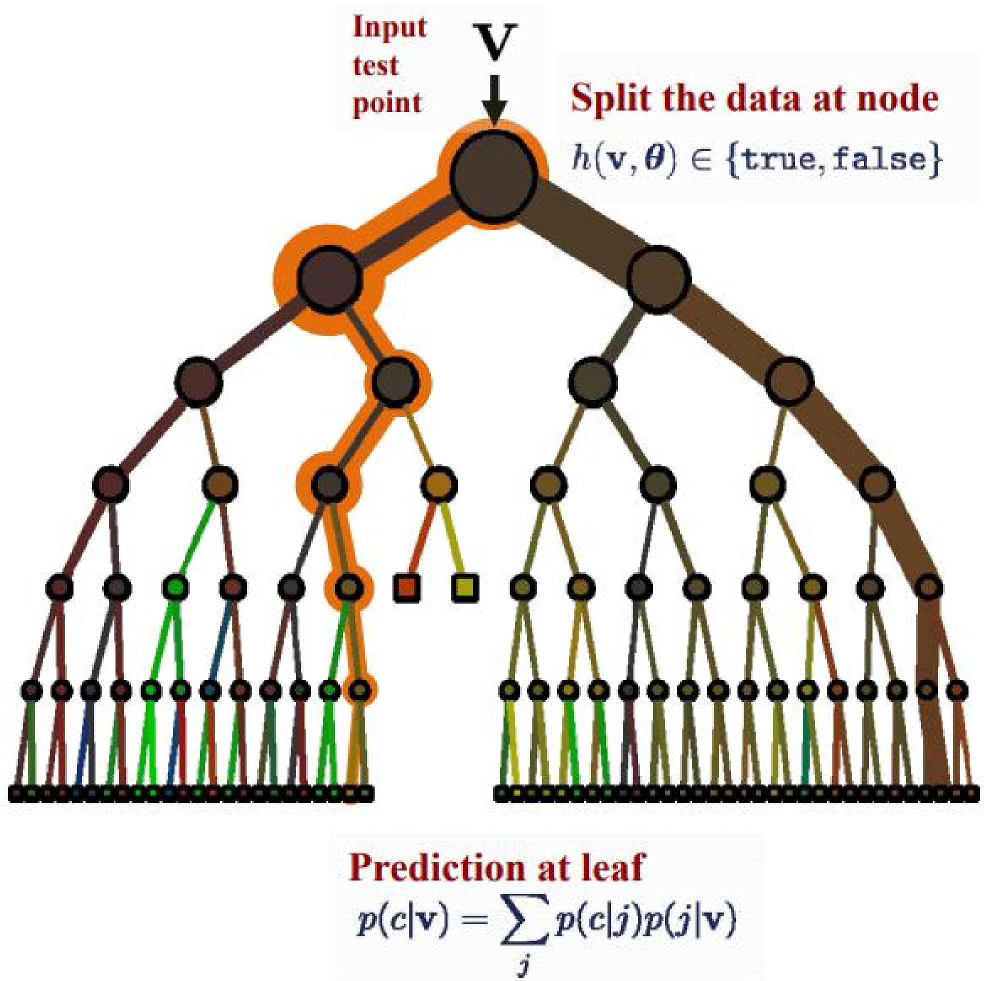
- A forest is an ensemble of trees. The trees are all slightly different from one another



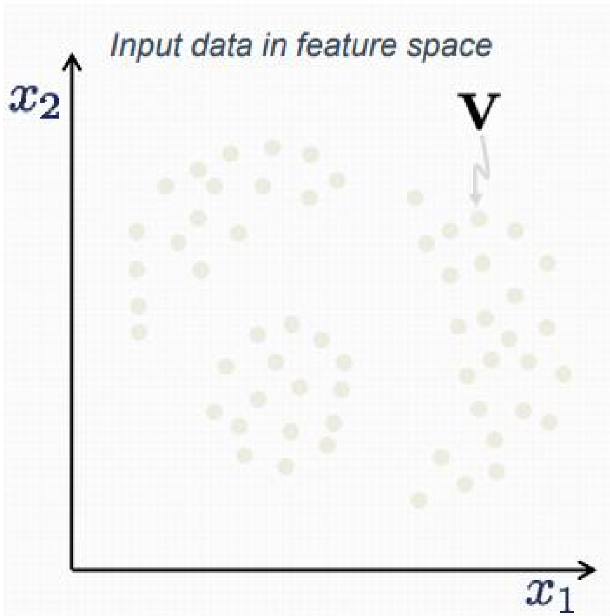


Decision Tree Testing (Runtime)





Decision Tree Training (Offline)



$$\mathbf{v} = (x_1, \cdots, x_d) \in \mathbb{R}^d$$

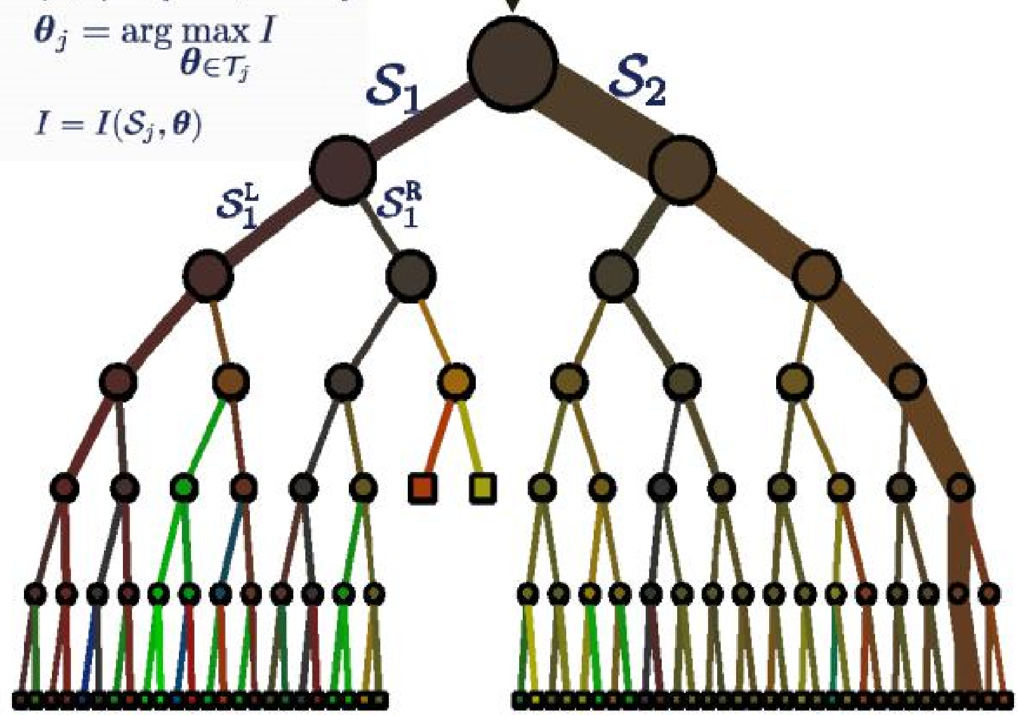
Input training data
$\mathcal{S}_0 = \{\mathbf{v}, c\}$

How to split the data?

$h(\mathbf{v}, \boldsymbol{\theta}) \in \{\texttt{true}, \texttt{false}\}$

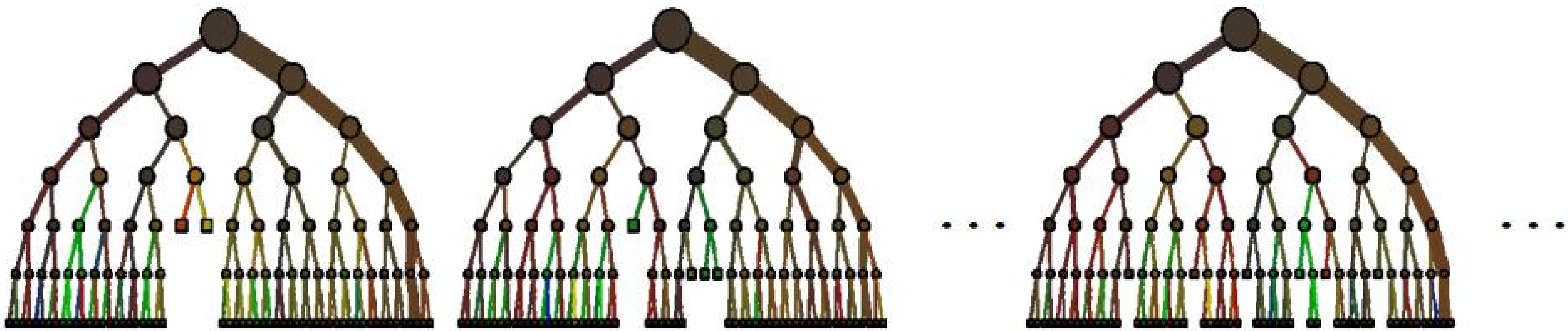
$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} I$

$I = I(\mathcal{S}_j, \boldsymbol{\theta})$



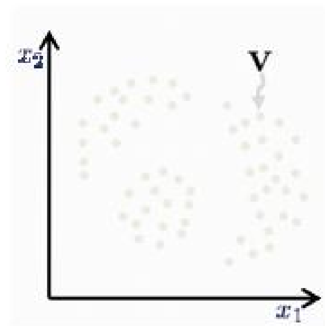
Binary tree? Ternary?
How big a tree?
What tree structure?

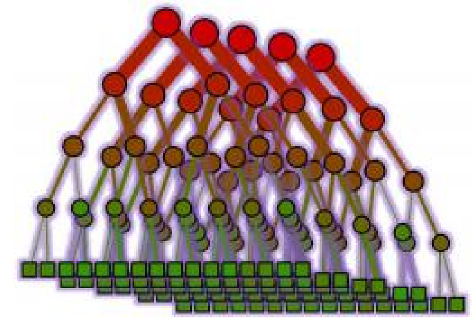
Decision Tree Training (Offline)



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

Decision Forest Model





Basic notation

Input data point	e.g.	$\mathbf{v} = (x_1, \cdots, x_d) \in \mathbb{R}^d$	*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		$\boldsymbol{\theta} \in \mathcal{T}$	*Parameters related to each split node: i) which features, ii) what geometric primitive, iii) thresholds.*
	Node objective function (train.)	e.g.	$I = I(\mathcal{S}_j, \boldsymbol{\theta})$	*The "energy" to be minimized when training the j-th split node*
	Node weak learner	e.g.	$h(\mathbf{v}, \boldsymbol{\theta}_j) \in \{\texttt{true}, \texttt{false}\}$	*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.	$p(c\|\mathbf{v}) = \frac{1}{T}\sum_t^T p_t(c\|\mathbf{v})$	*How to compute the forest output from that of individual trees?*

Random Forest

Algorithm 1 Random Forest

Precondition: A training set $S := (x_1, y_1), \ldots, (x_n, y_n)$, features F, and number of trees in forest B.

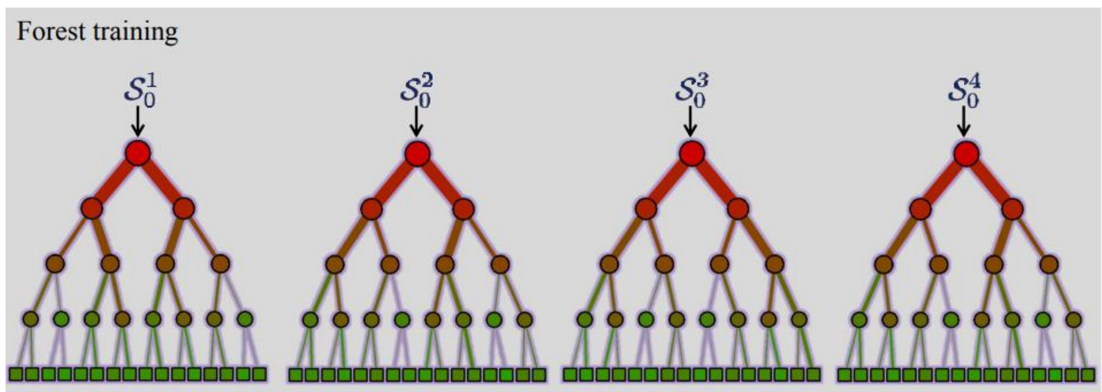
```
function RandomForest(S, F)
    H ← ∅
    for i ∈ 1, ..., B do
        S^(i) ← A bootstrap sample from S
        h_i ← RandomizedTreeLearn(S^(i), F)
        H ← H ∪ {h_i}
    end for
    return H
end function
function RandomizedTreeLearn(S, F)
    At each node:
        f ← very small subset of F
        Split on best feature in f
    return The learned tree
end function
```

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

Efficient training



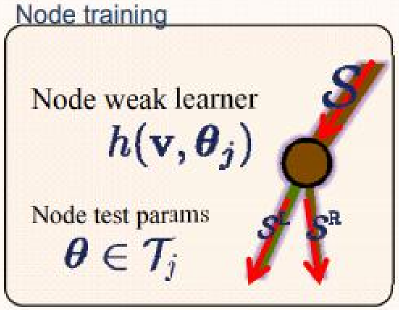
Precondition: A training set $S := (x_1, y_1), \ldots, (x_n, y_n)$, features F, and number of trees in forest B.

```
function RandomForest(S, F)
    H ← ∅
    for i ∈ 1, ..., B do
        S^(i) ← A bootstrap sample from S
        h_i ← RandomizedTreeLearn(S^(i), F)
        H ← H ∪ {h_i}
    end for
    return H
end function
```

Randomness Model

- 2) Randomized node optimization (RNO)

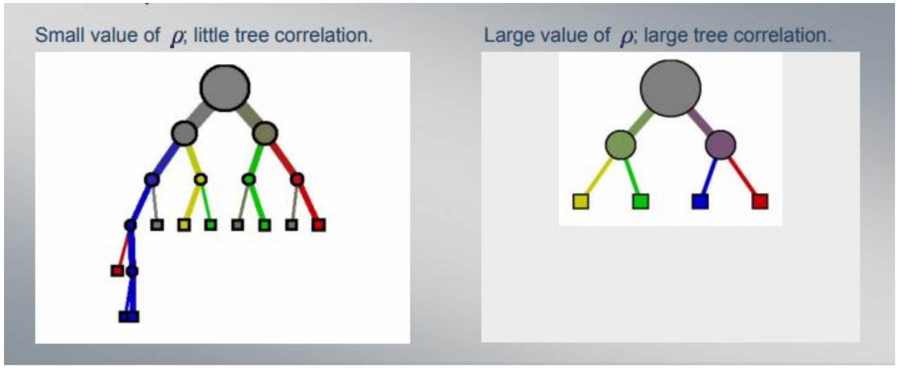
$\mathcal{T}$	The full set of all possible nodes
$\mathcal{T}_j \subset \mathcal{T}$	For each node the set of randomly sampled features
$\rho = \|\mathcal{T}_j\|$	Randomness control parameter. For $\rho = \|\mathcal{T}\|$ no randomness and maximum tree correlation. For $\rho = 1$ minimum tree correlation.



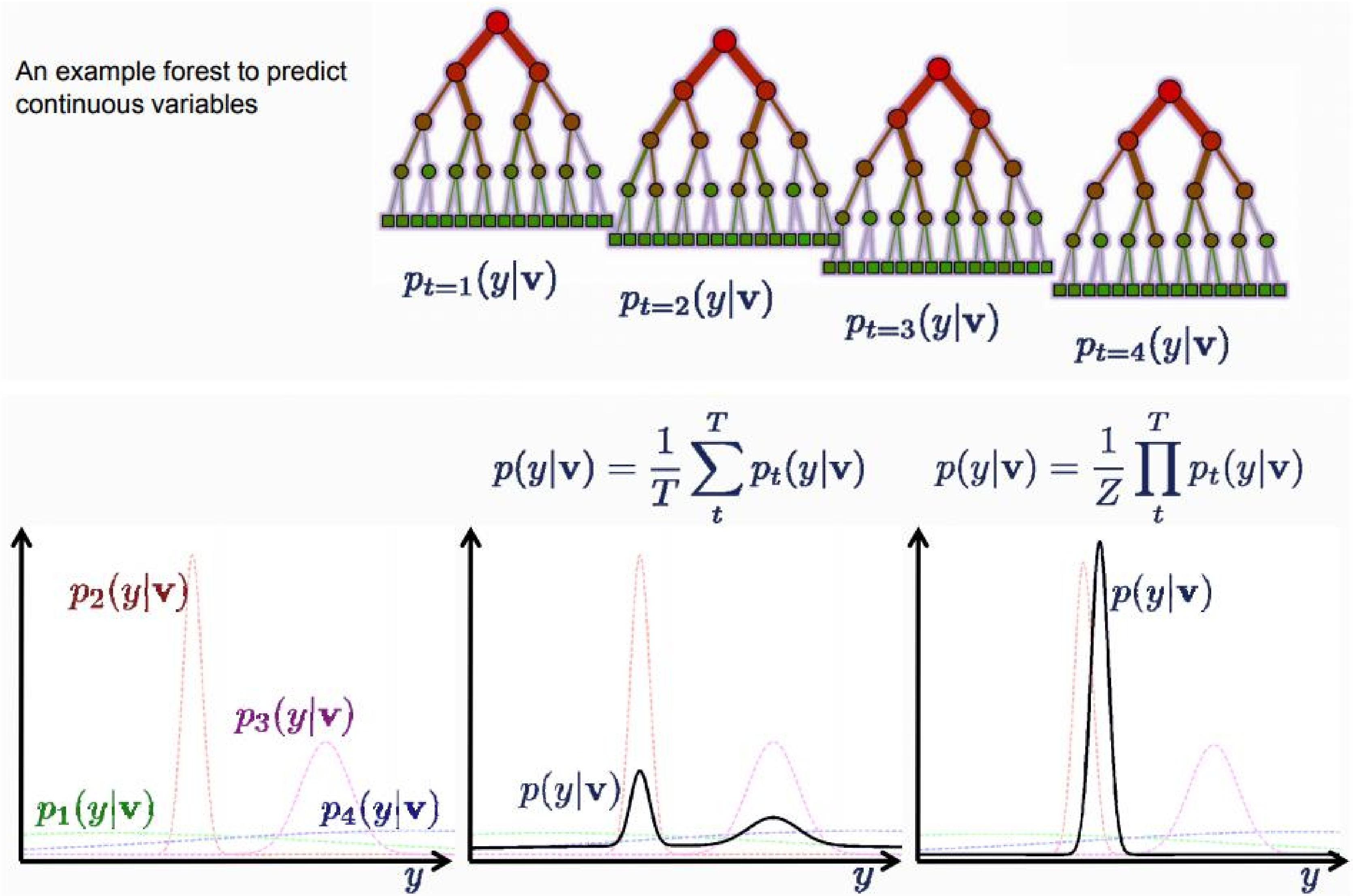
Precondition: A training set $S := (x_1, y_1), \ldots, (x_n, y_n)$, features F, and number of trees in forest B.

```
function RandomForest(S, F)
    H ← ∅
    for i ∈ 1, ..., B do
        S^(i) ← A bootstrap sample from S
        h_i ← RandomizedTreeLearn(S^(i), F)
        H ← H ∪ {h_i}
    end for
    return H
end function
```

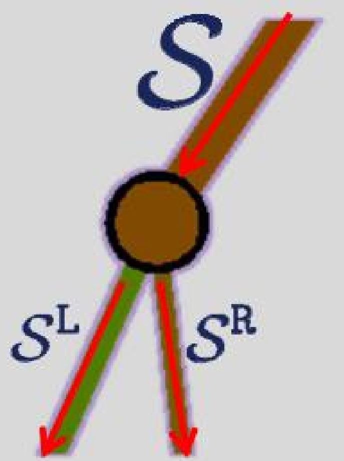
Here B = ρ.



The Ensemble Model



Training and Information Gain



Information gain

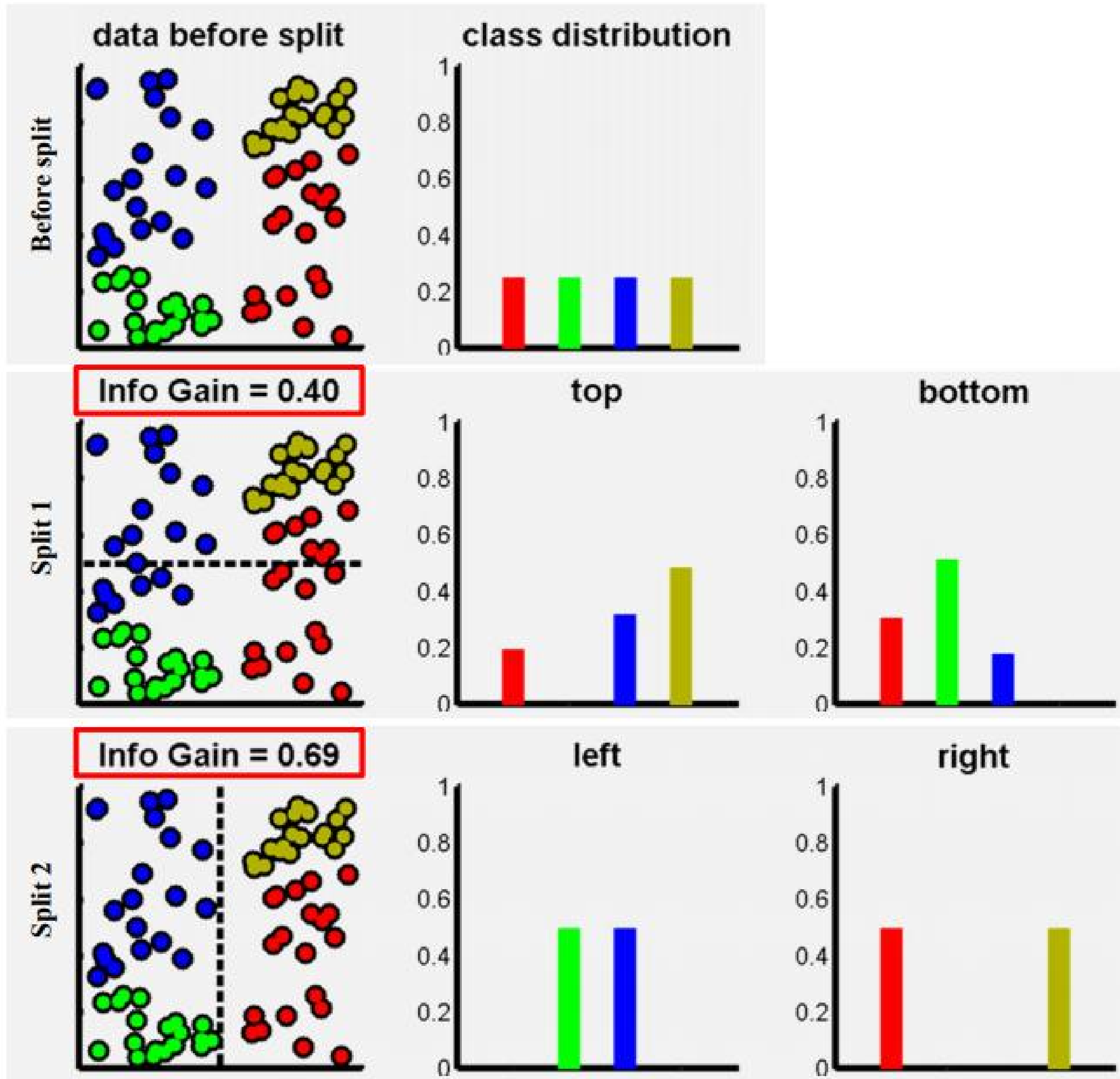
$$I(\mathcal{S},\boldsymbol{\theta}) = H(\mathcal{S}) - \sum_{i\in\{\mathrm{L,R}\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i)$$

Shannon's entropy

$$H(\mathcal{S}) = -\sum_{c\in\mathcal{C}} p(c)\log(p(c))$$

Node training

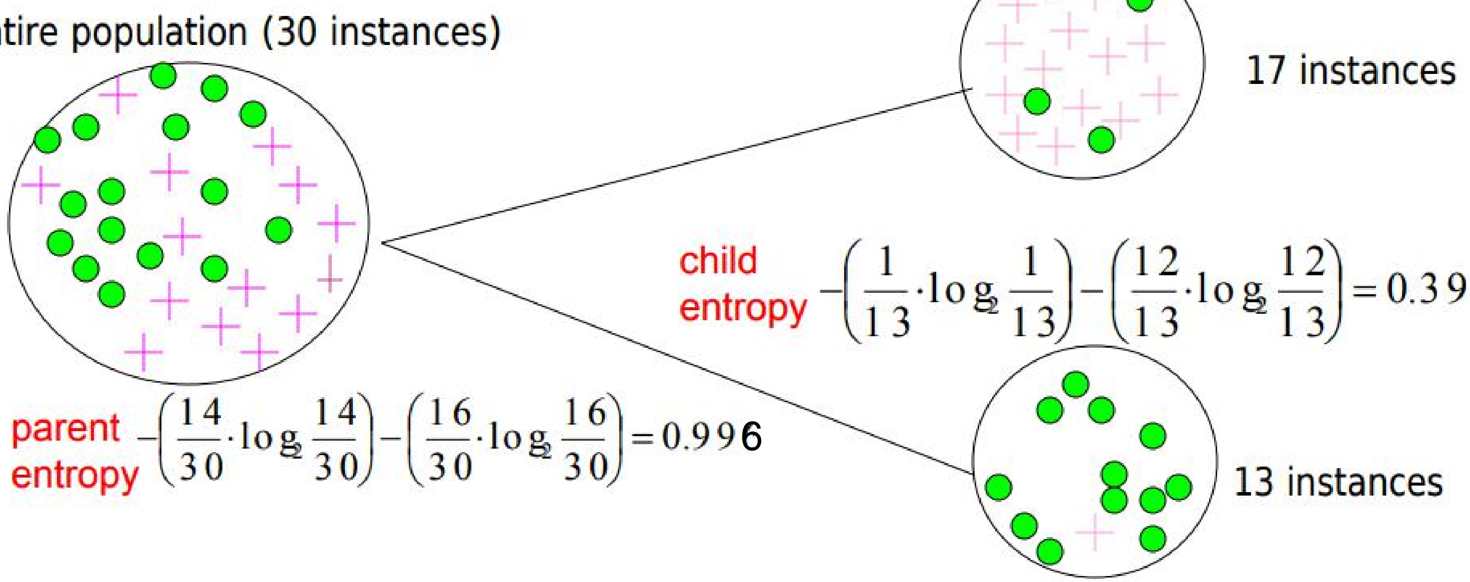
$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}\in\mathcal{T}_j} I(\mathcal{S}_j,\boldsymbol{\theta})$$



Calculating Information Gain

- Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$



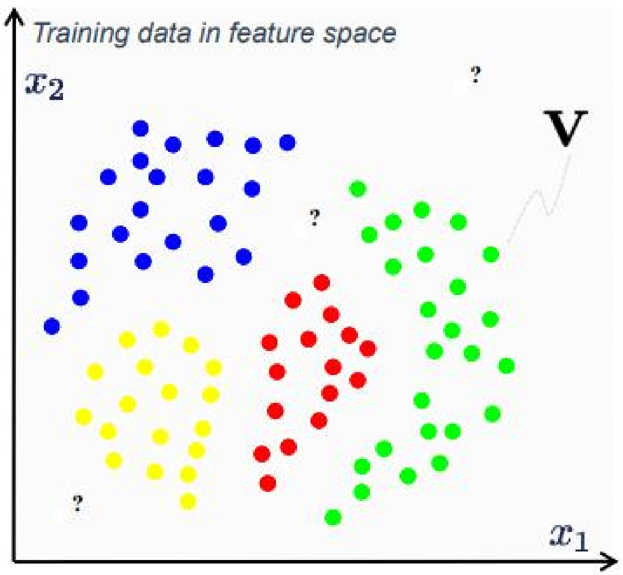
child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

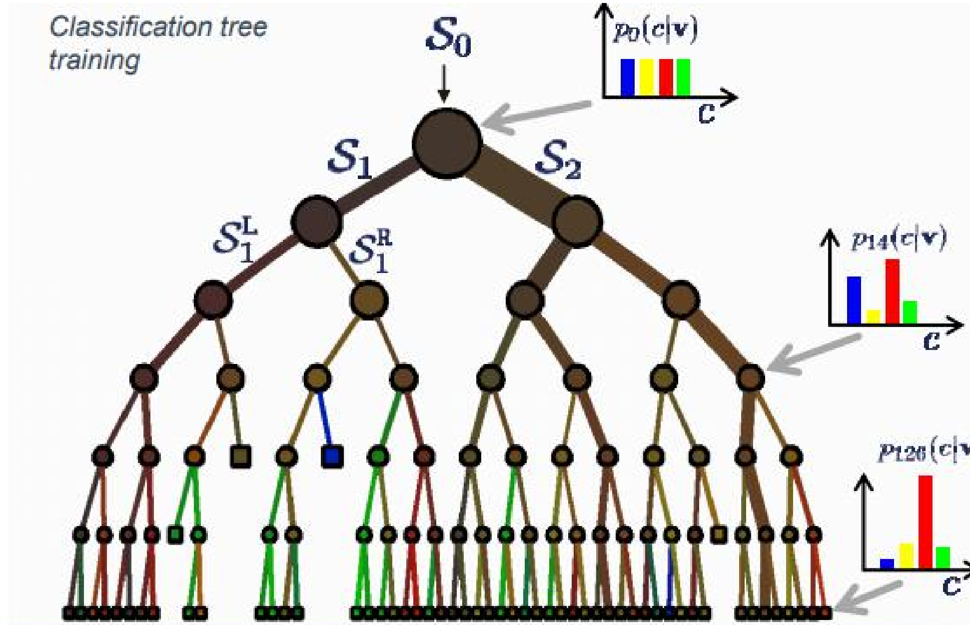
parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38 for this split.

Classification Forest





Model specialization for classification

Input data point	$\mathbf{v} = (x_1, \cdots, x_d) \in \mathbb{R}^d$	(x_i is feature response)
Output is categorical	$c \in \mathcal{C}$ with $\mathcal{C} = \{c_k\}$	(discrete set)
Node weak learner	$h(\mathbf{v}, \boldsymbol{\theta}) \in \{\mathtt{true}, \mathtt{false}\}$	
Obj. funct. for node j	$I = H(\mathcal{S}_j) - \sum_{i=\mathrm{L,R}} \frac{\lvert\mathcal{S}_j^i\rvert}{\lvert\mathcal{S}_j\rvert} H(\mathcal{S}_j^i)$	(information gain)
Training node j	$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} I(\mathcal{S}_j, \boldsymbol{\theta})$	
Predictor model	$p(c\vert\mathbf{v}) = \sum_j p(c\vert j) p(j\vert\mathbf{v})$	(class posterior)

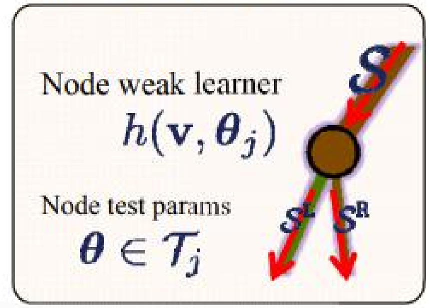
Entropy of a discrete distribution

$$H(\mathcal{S}) = -\sum_{c \in \mathcal{C}} p(c) \log(p(c))$$

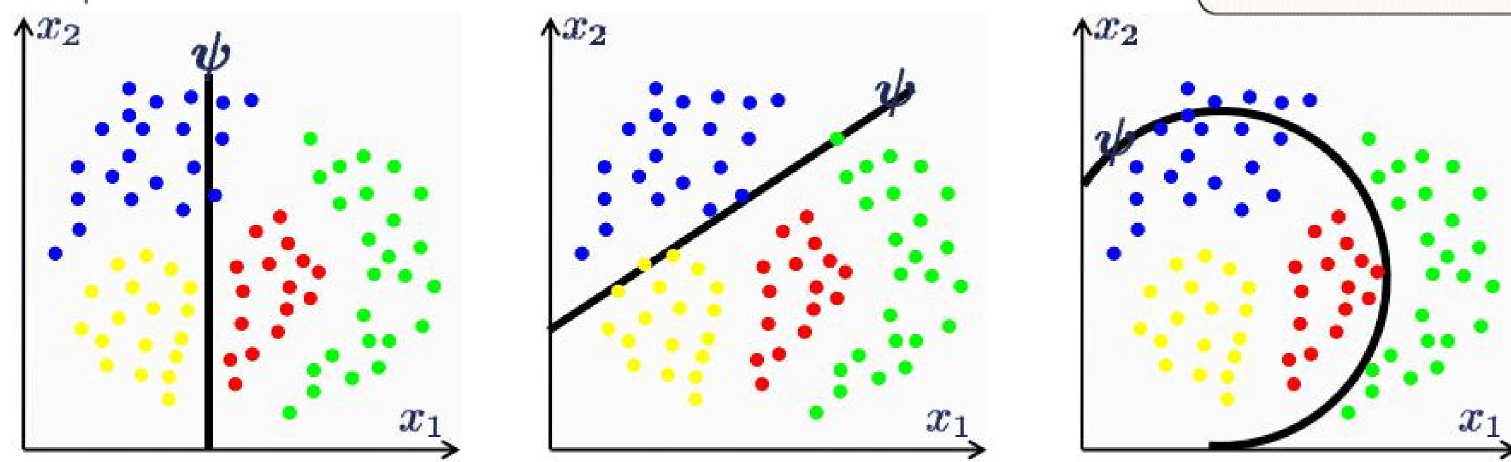
with $c(\mathbf{v}) : \mathbb{R}^d \rightarrow \mathcal{C}$

Weak Learners

Splitting data at node 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

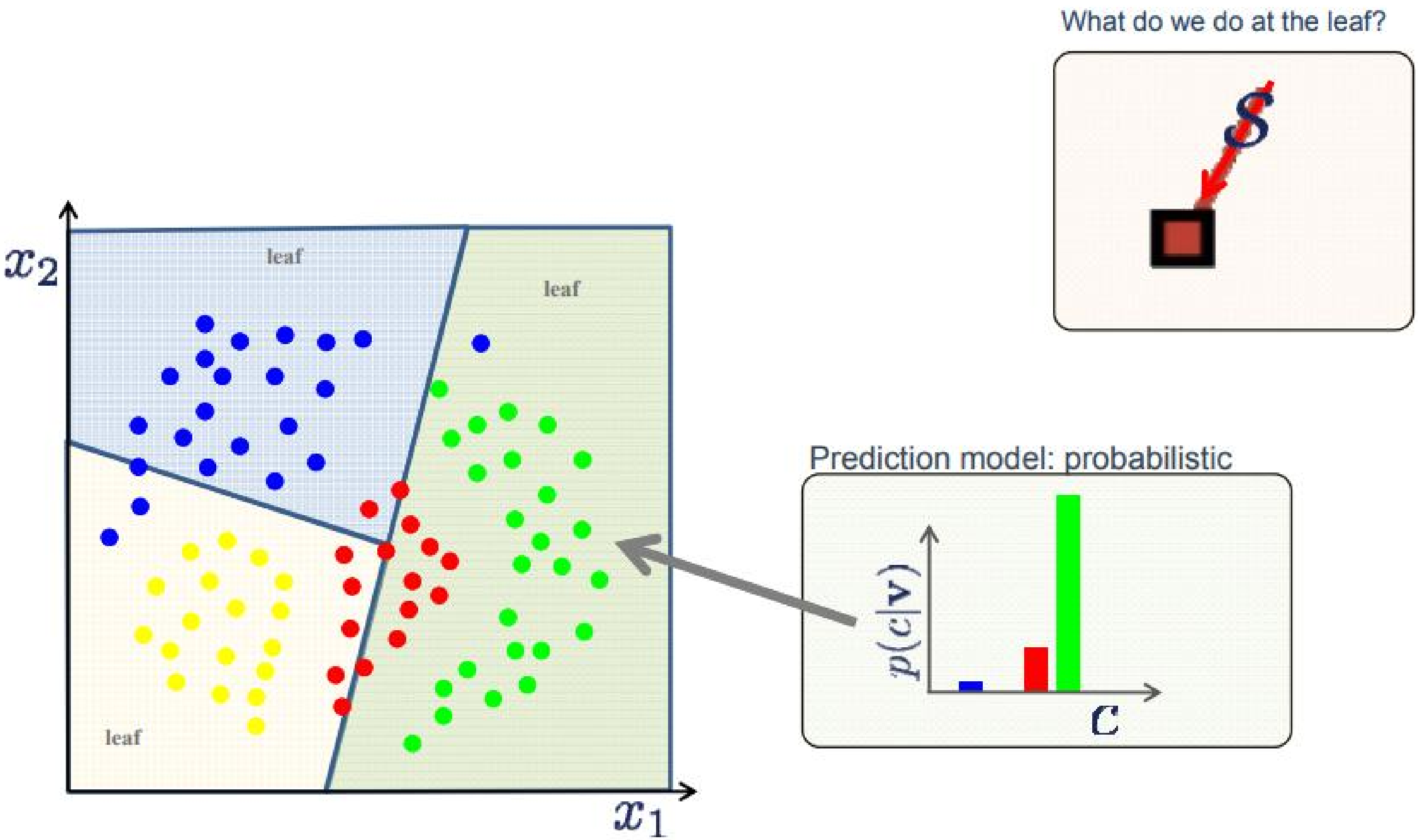
$$h(\mathbf{v}, \boldsymbol{\theta}) = \left[\tau_1 > \boldsymbol{\phi}^\top(\mathbf{v}) \, \boldsymbol{\psi} \, \boldsymbol{\phi}(\mathbf{v}) > \tau_2\right]$$

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

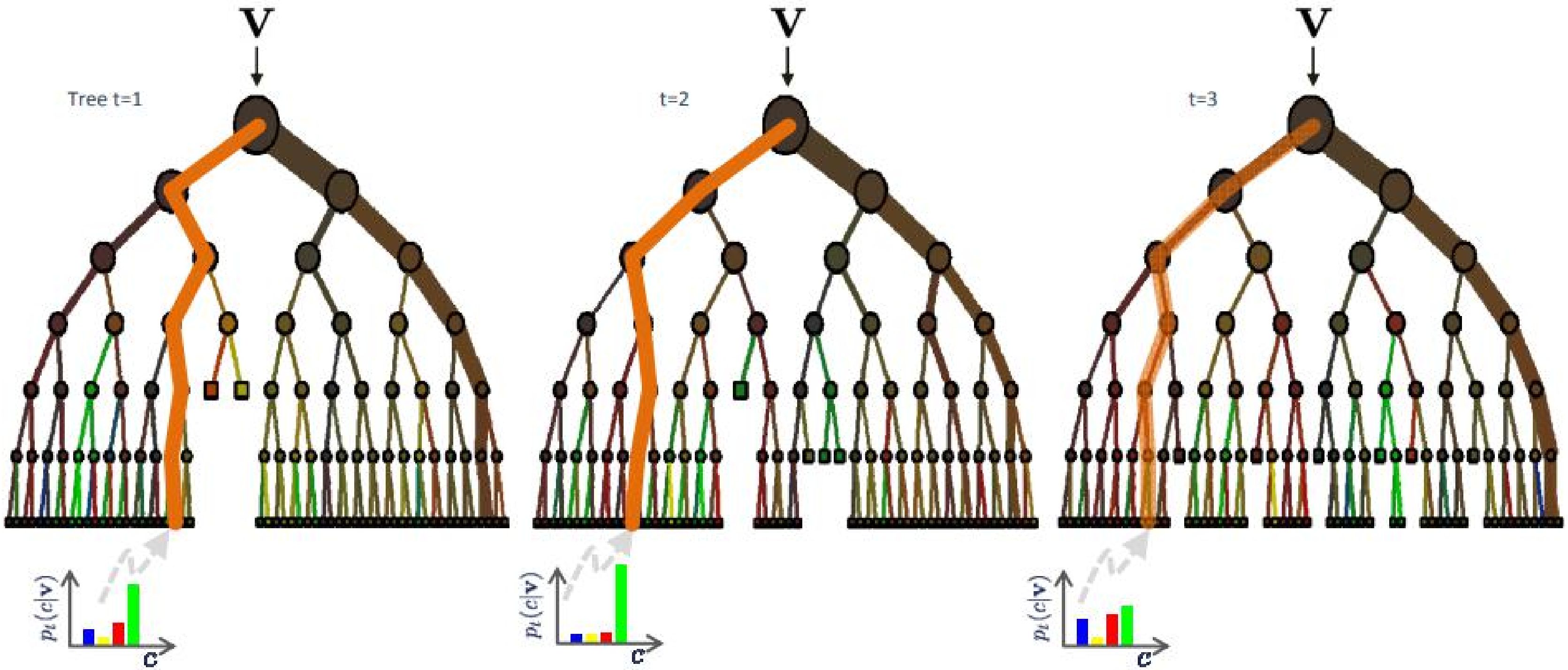
With $\boldsymbol{\psi} \in \mathbb{R}^{3\times 3}$ a matrix representing a conic.

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

Prediction Model

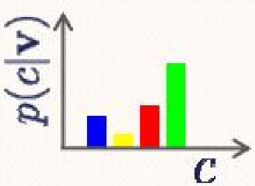


Classification Forest: Ensemble Model

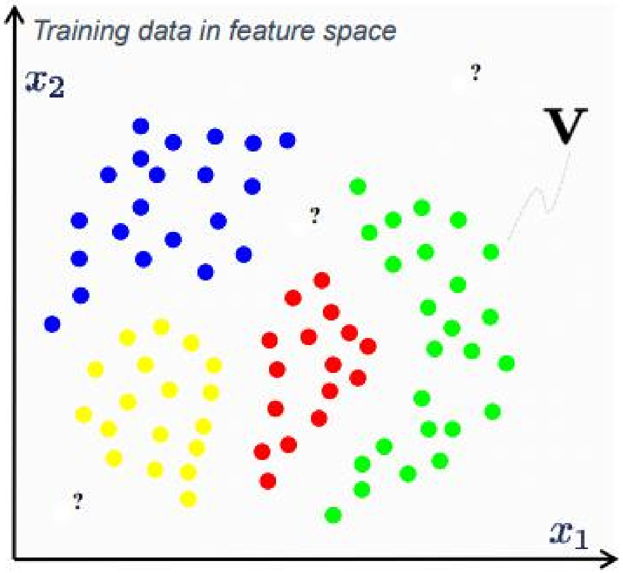


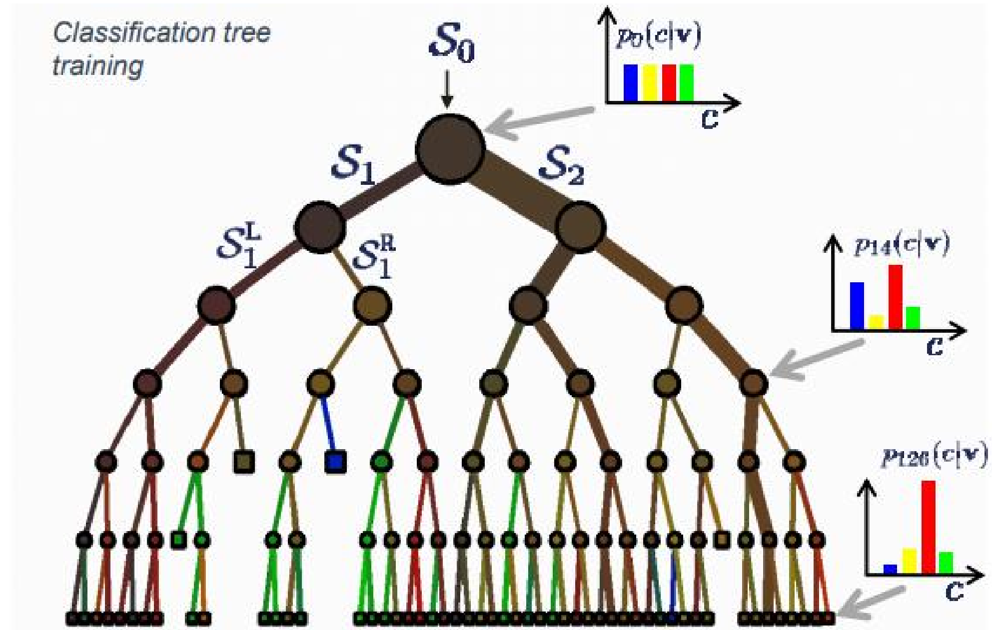
The ensemble model

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



Classification Forest





Model specialization for classification

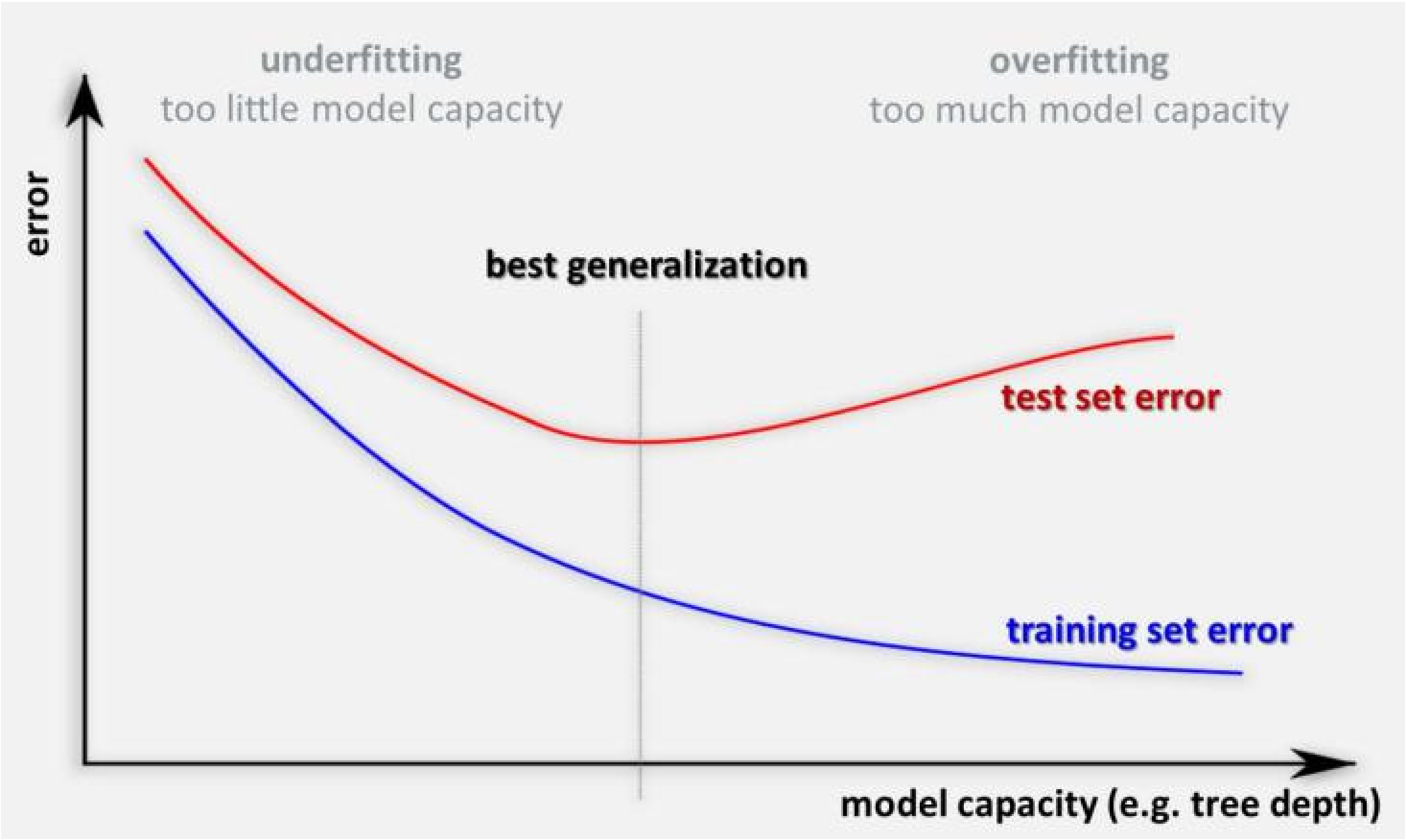
Input data point	$\mathbf{v} = (x_1, \cdots, x_d) \in \mathbb{R}^d$	*(x_i is feature response)*
Output is categorical	$c \in \mathcal{C}$ with $\mathcal{C} = \{c_k\}$	*(discrete set)*
Node weak learner	$h(\mathbf{v}, \boldsymbol{\theta}) \in \{\mathtt{true}, \mathtt{false}\}$	
Obj. funct. for node j	$I = H(\mathcal{S}_j) - \sum_{i=\mathrm{L},\mathrm{R}} \frac{\lvert\mathcal{S}_j^i\rvert}{\lvert\mathcal{S}_j\rvert} H(\mathcal{S}_j^i)$	*(information gain)*
Training node j	$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} I(\mathcal{S}_j, \boldsymbol{\theta})$	
Predictor model	$p(c\lvert\mathbf{v}) = \sum_j p(c\lvert j) p(j\lvert\mathbf{v})$	*(class posterior)*

Entropy of a discrete distribution

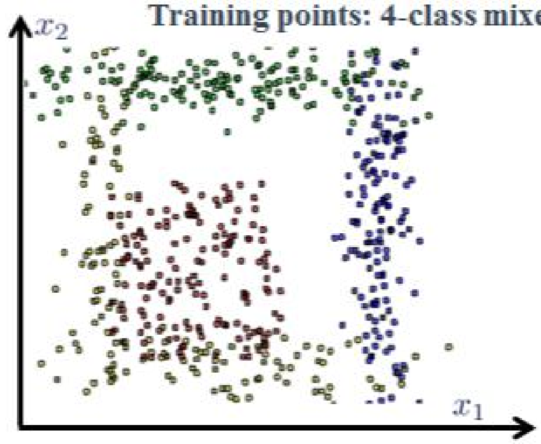
$$H(\mathcal{S}) = -\sum_{c \in \mathcal{C}} p(c) \log(p(c))$$

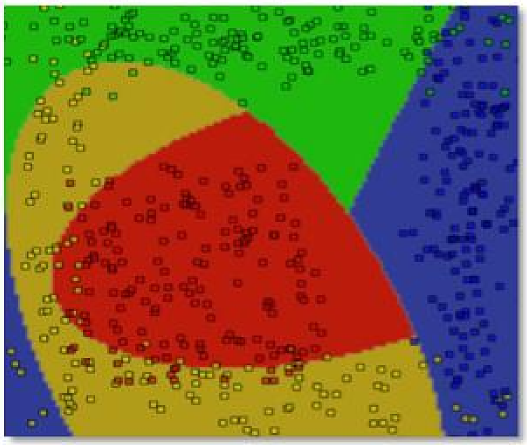
with $c(\mathbf{v}) : \mathbb{R}^d \rightarrow \mathcal{C}$

Overfitting and Underfitting

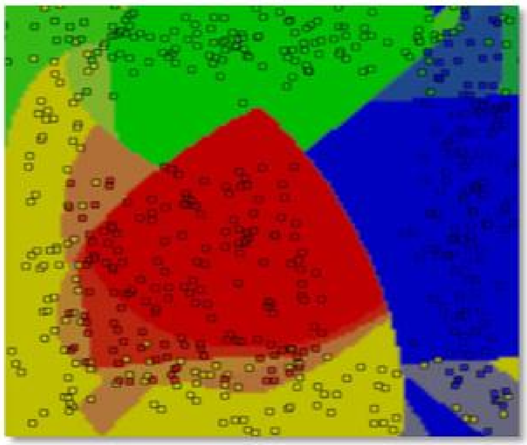


Effect of Tree Depth

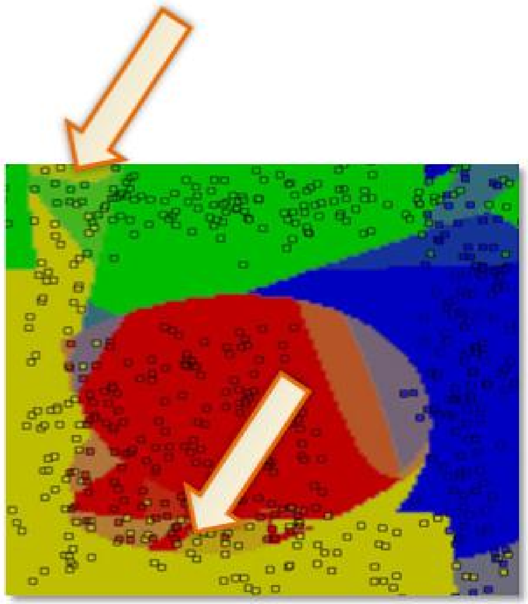




T=200, D=3, w. l. = conic



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



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

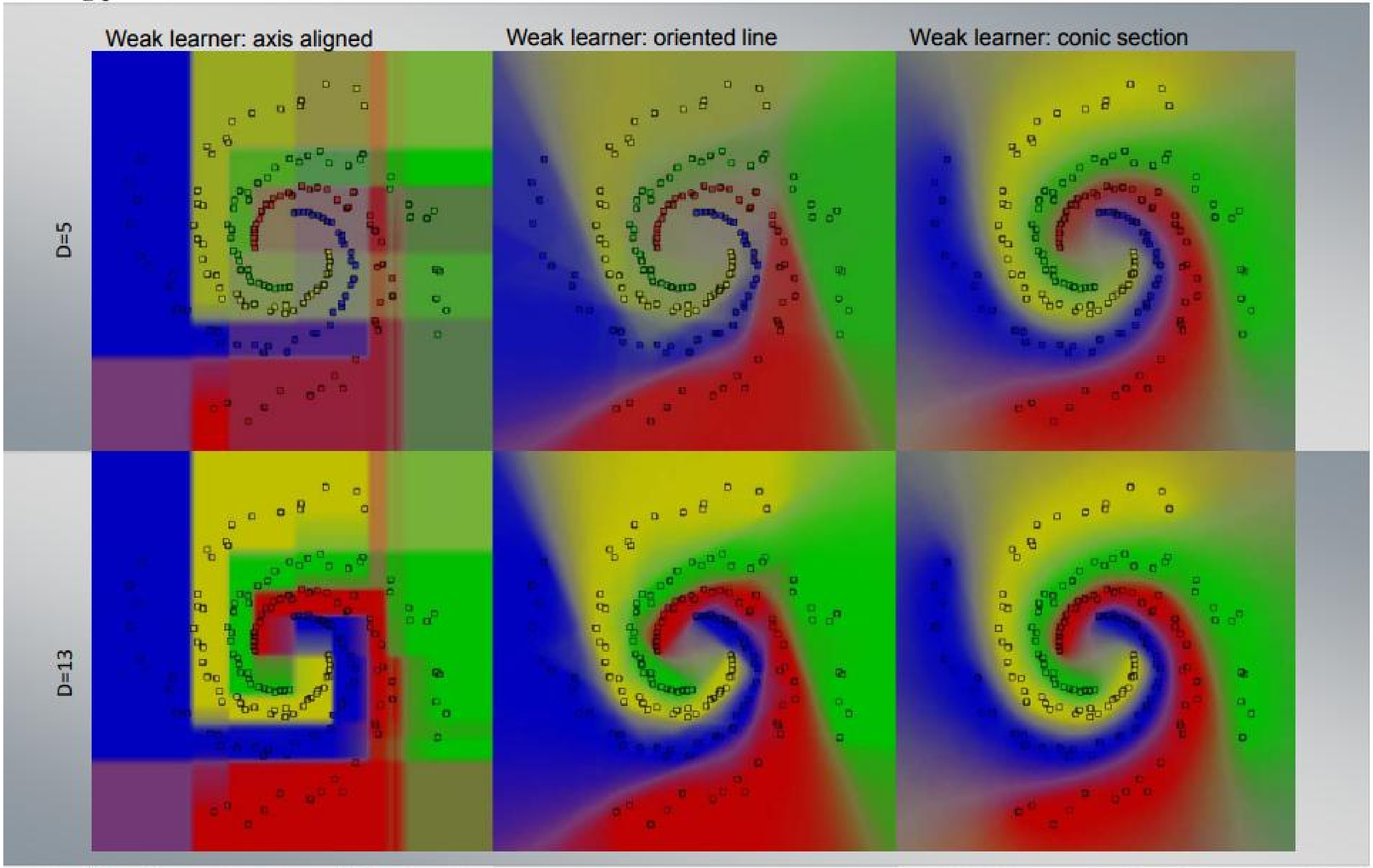
max tree depth, D

underfitting

overfitting

Effect of Weak Learner Model and Randomness

Testing posteriors

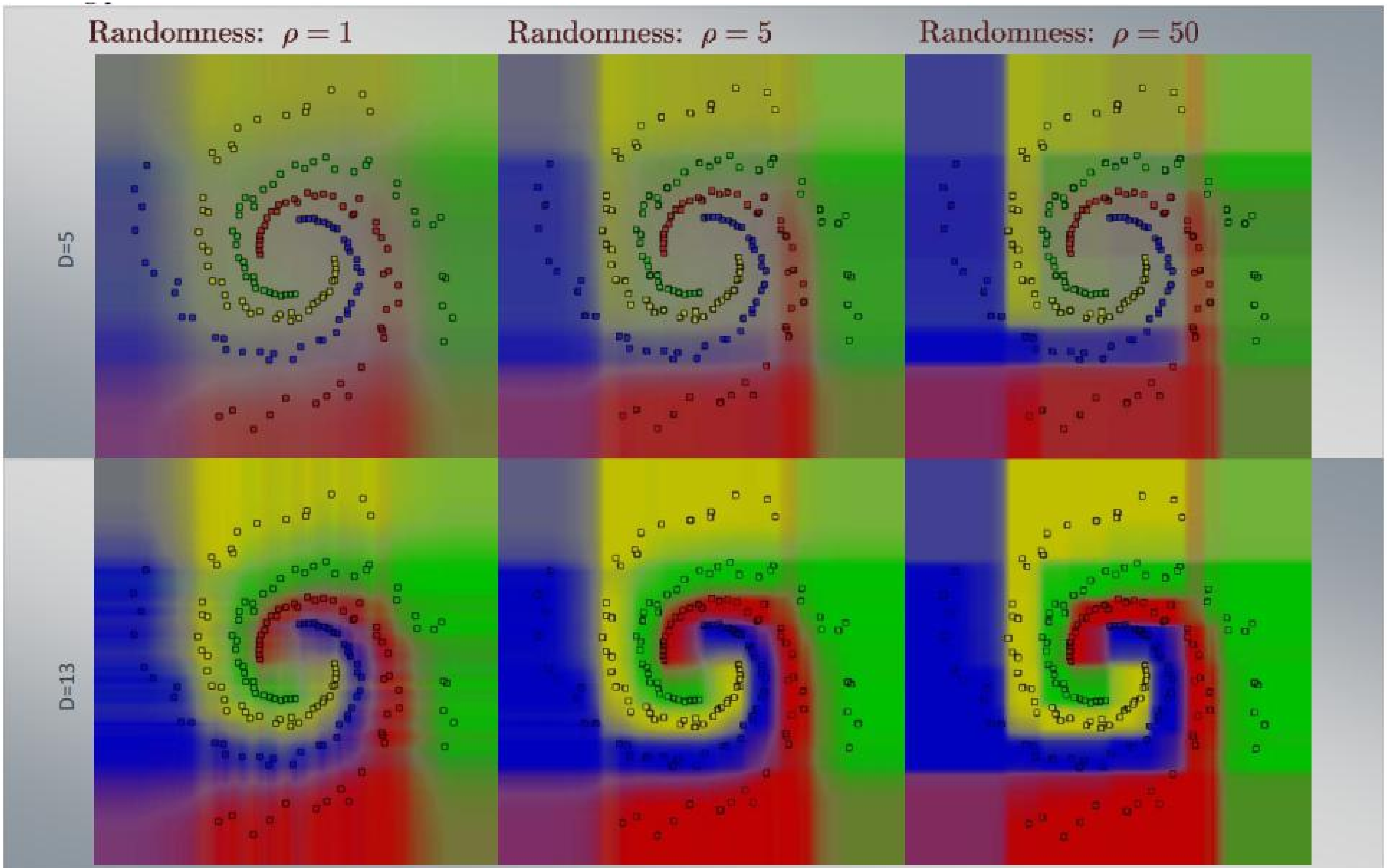


Randomness: $\rho = 500$

Parameters: T=400 predictor model = prob.

Effect of Weak Learner Model and Randomness

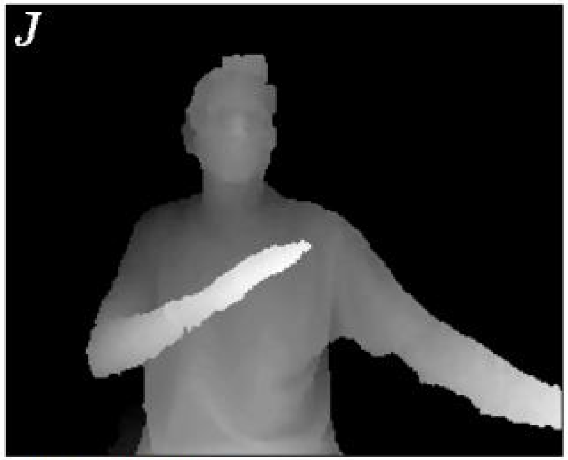
Testing posteriors



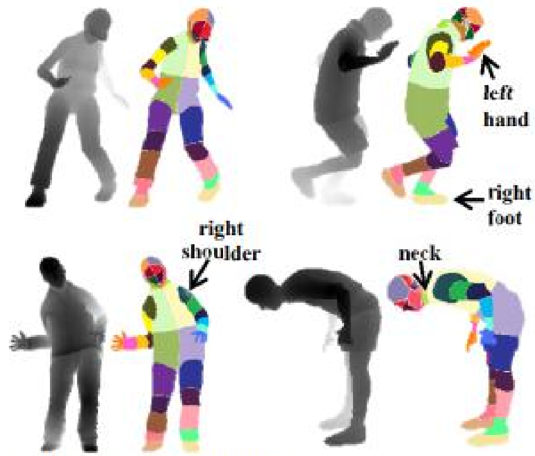
Weak learner: axis aligned

Parameters: T=400 predictor model = prob.

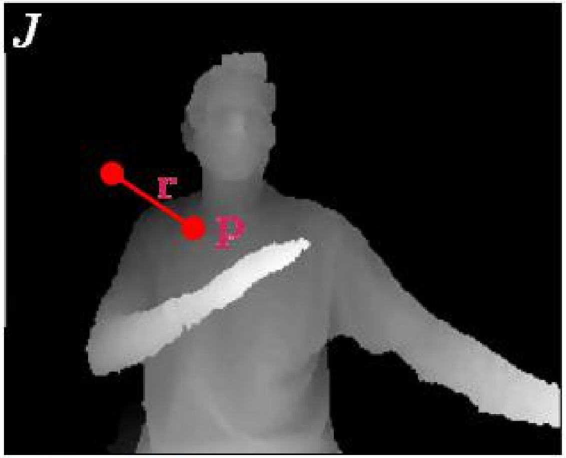
Body tracking in Microsoft Kinect for XBox 360



Input depth image



Training labelled data

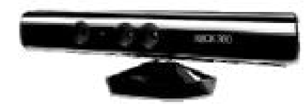


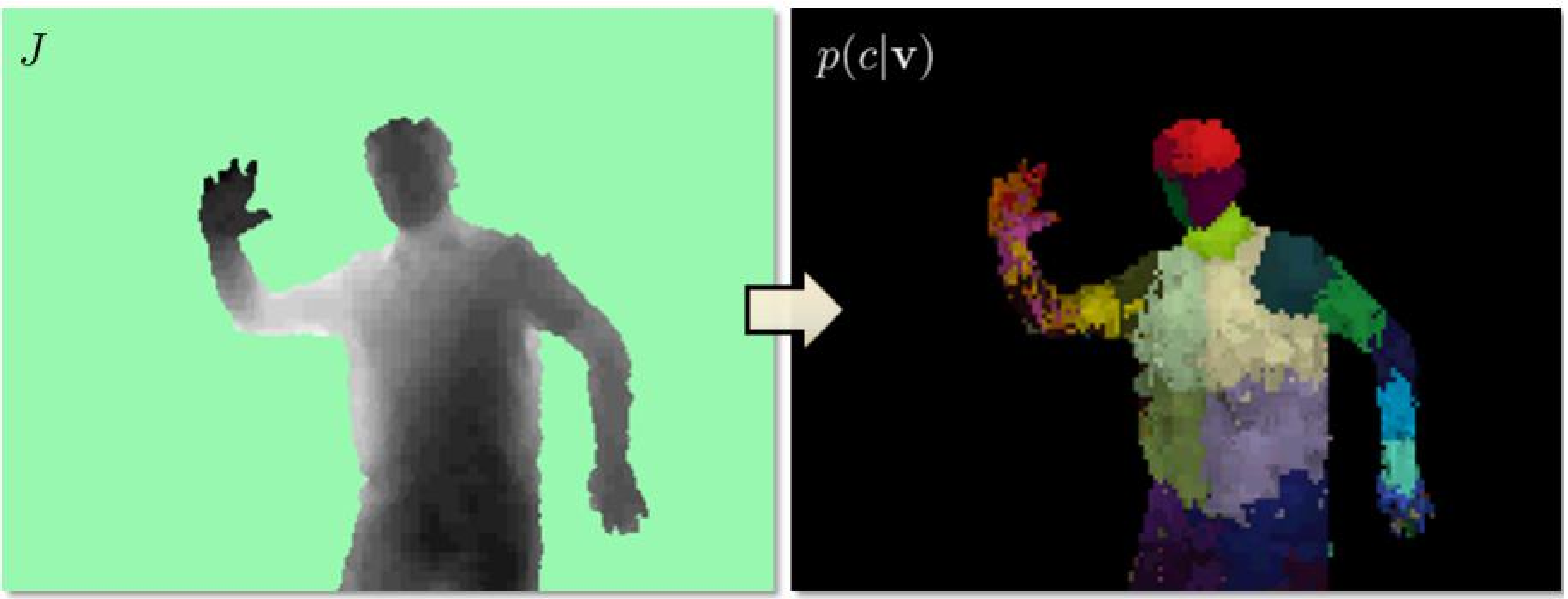
Visual features

Classification forest

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

Body tracking in Microsoft Kinect for XBox 360





Input depth image (bg removed)

Inferred body parts posterior

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
- Can handle categorical variables
- Robust to label noise
- Can be used in clustering, locating outliers and semisupervised learning

Outline

- Bagging
- Random Forest
- **Boosting**

Boosting

- 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"

Definition of Boosting

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

$$h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Output final hypothesis H_{final}

AdaBoost

- Constructing Dt

$$D_1(i) = \frac{1}{m}$$

$\Rightarrow$

$$D_{t+1} = \frac{D_t(i)}{Z_t} c(x)$$

$$c(x) = \begin{cases} e^{-\alpha_t} & : y_i = h_t(x_i) \\ e^{\alpha_t} & : y_i \neq h_t(x_i) \end{cases}$$

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

where Zt is a normalization constant

- Final hypothesis:

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

$$\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t} > 0$$

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train base learner using distribution D_t.
- Get base classifier $h_t : X \to \mathbb{R}$.
- 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: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train base learner using distribution D_t.
- Get base classifier $h_t : X \to \mathbb{R}$. ← with minimum ϵ_t
- 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: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

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

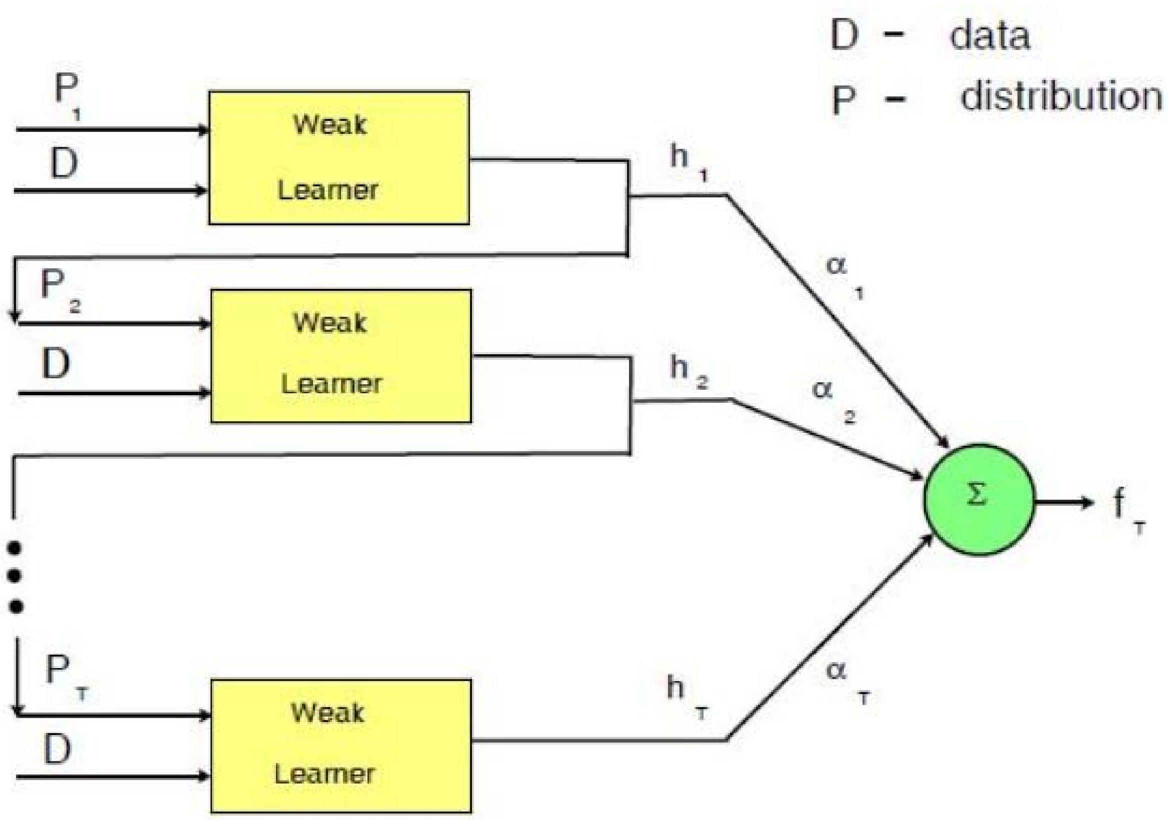
$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

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

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

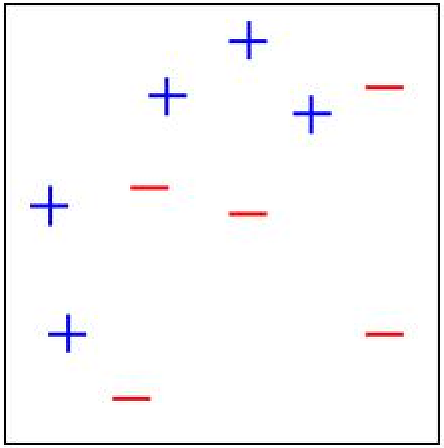
$$\epsilon_t = \frac{1}{\sum_{i=1}^{n} D_t(i)} \sum_{i=1}^{m} D_t(i) \delta(h_t(x_i) \neq y_i)$$

The AdaBoost Algorithm

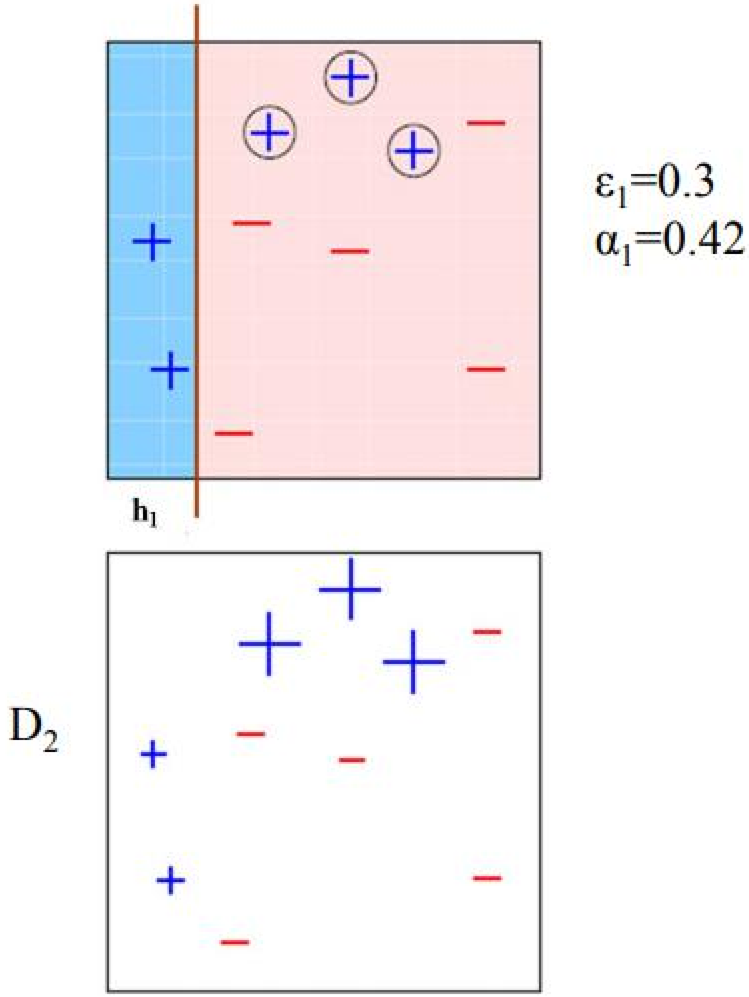


Toy Example

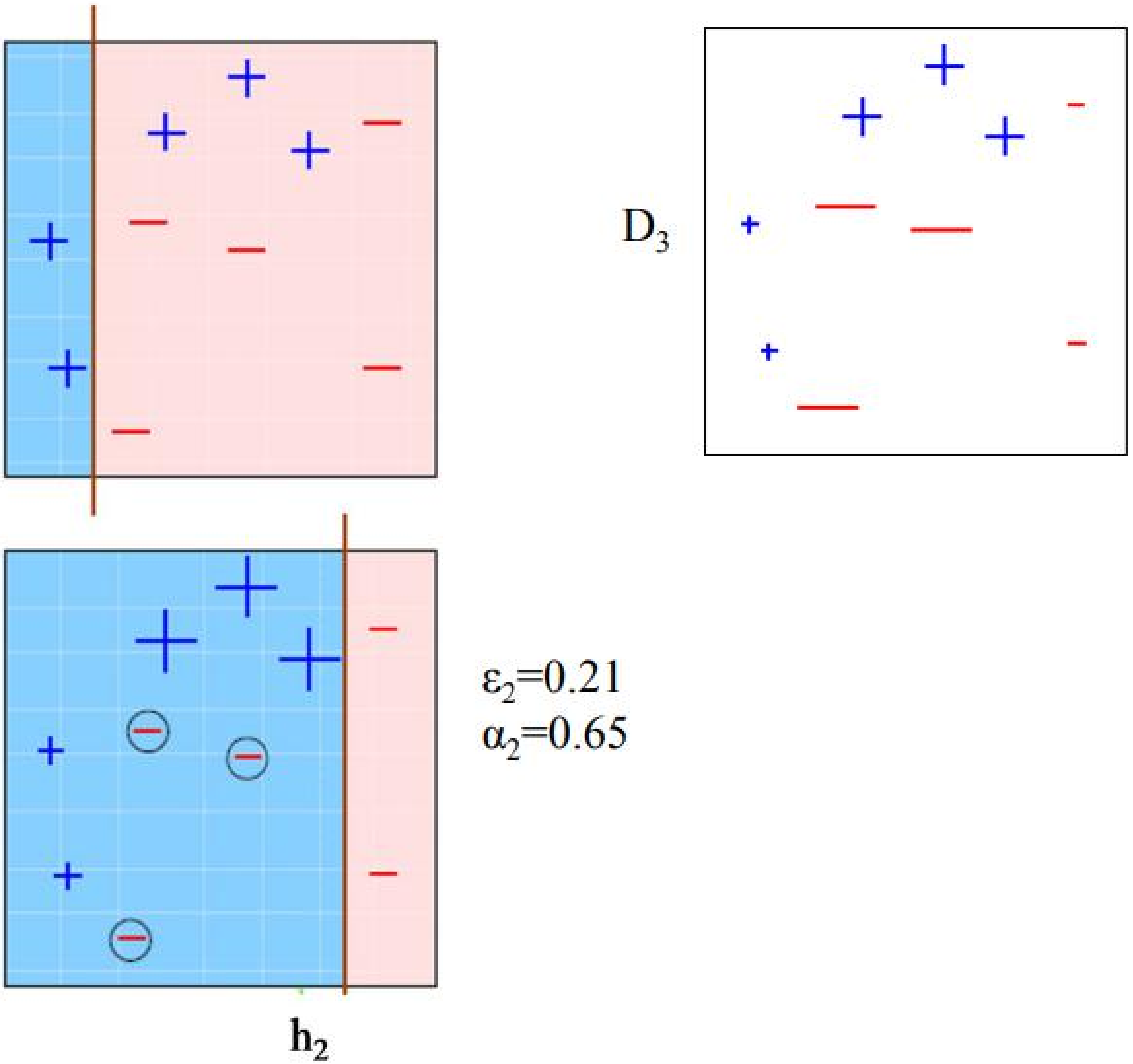
D_1



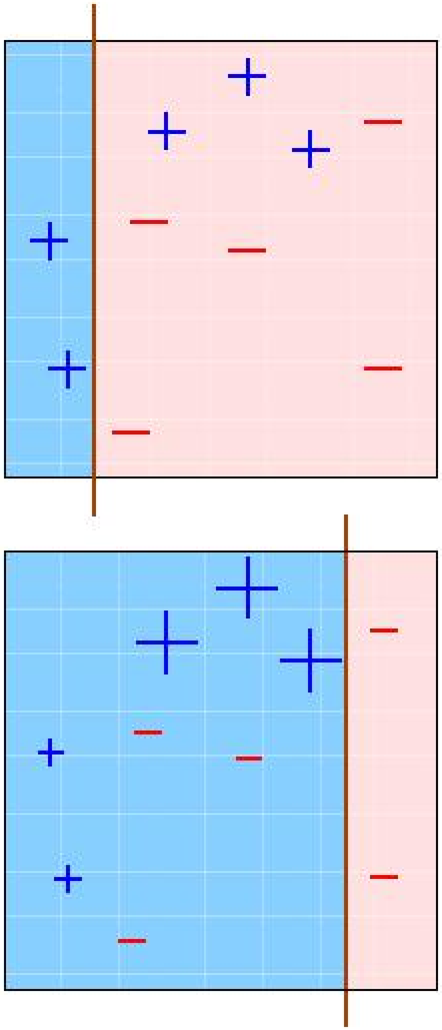
Toy Example: Round 1

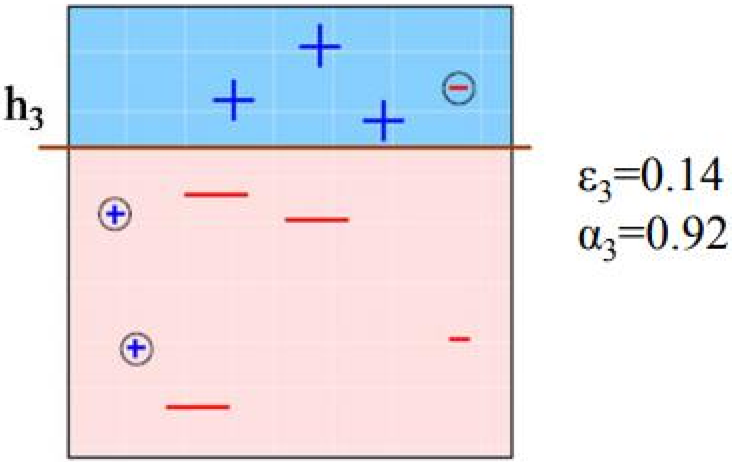


Toy Example: Round 2

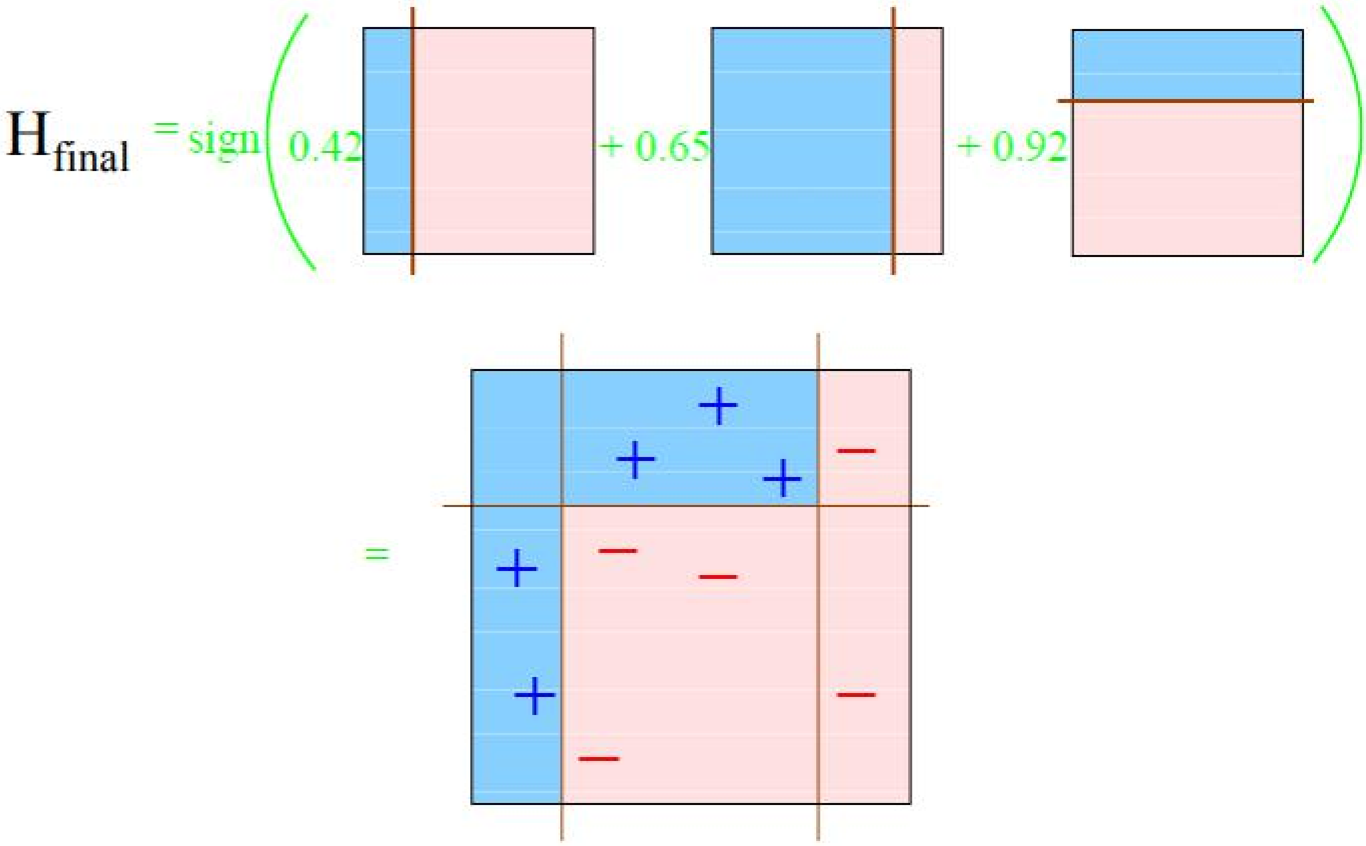


Toy Example: Round 3





Toy Example: Final Hypothesis



Q & A