Photogrammetry & Robotics Lab

Ensemble Classification: Bagging & Boosting

Cyrill Stachniss

Reminder: Classification

- Given a set of K classes

$$\Omega = \{\omega_1, \ldots, \omega_K\}$$

- and features $\boldsymbol{e}$
- learn a function f that assigns a class given feature

$$c = f(\boldsymbol{e}) \text{ with } c \in \Omega$$

Ensemble Methods

- The key idea of ensemble methods for classification is to use **multiple classifiers** and to **combine** them to a stronger classifier

Two prominent approaches

- Bagging
- Boosting

Bagging (Boostrap Aggregating)

- Idea of combining multiple classifiers trained on subsampled training data to obtain a better overall classifier
- Emphasis on multiple training sets

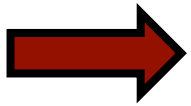
$$\{(\boldsymbol{e},\omega)\}_{i=1}^{N} \Rightarrow \{(\boldsymbol{e},\omega)\}_{i=1}^{N_1} \ldots \{(\boldsymbol{e},\omega)\}_{i=1}^{N_B}$$

$$c = f(\boldsymbol{e}) \Rightarrow \text{majority vote } f_1(\boldsymbol{e}), \ldots, f_B(\boldsymbol{e})$$

Boosting

- Incrementally ensemble building
- Train new model instances to emphasize the training data instances that previous models misclassified
- Emphasis on multiple classifiers

$$c = f(\boldsymbol{e})$$



weighted majority vote

$$w_1 f_1(\boldsymbol{e}), \ldots, w_B f_B(\boldsymbol{e})$$

incrementally learned

Random Forests
(Bagging with Decision Trees)

Bagging

- Combining multiple classifiers trained from subsampled training sets to obtain a better classifier

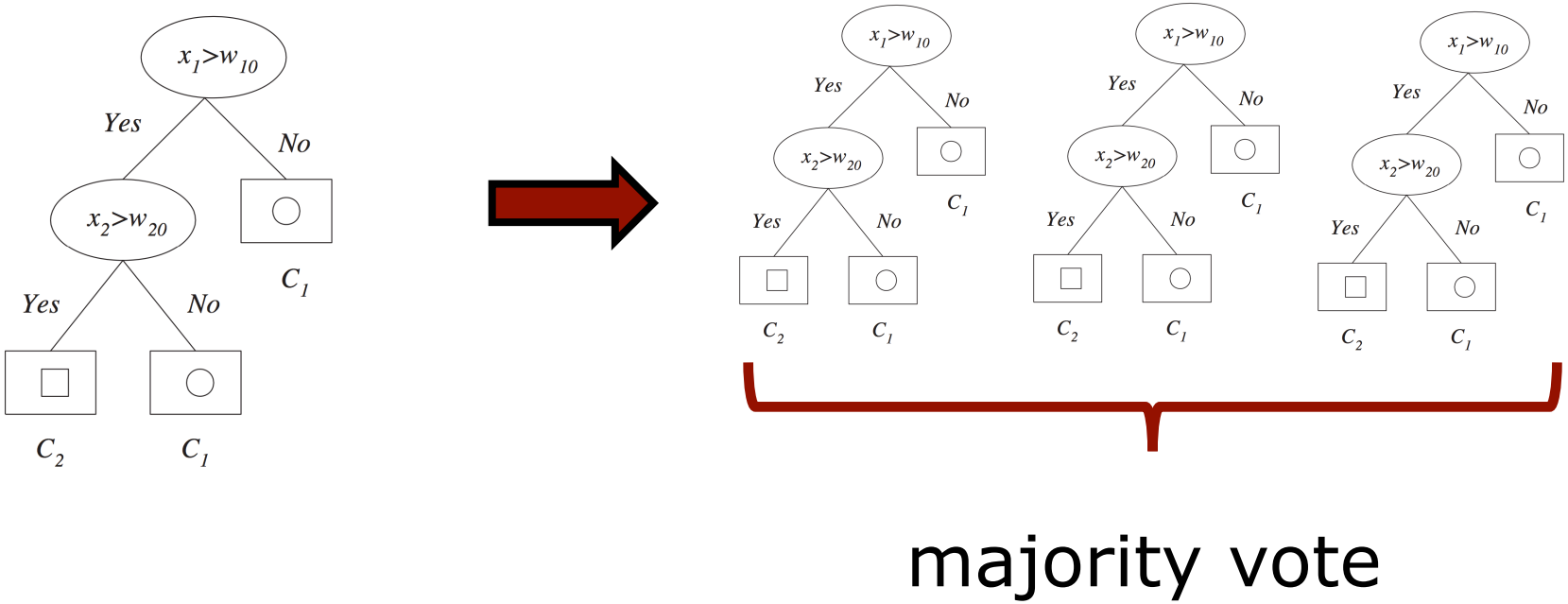
$$\{(\boldsymbol{e},\omega)\}_{i=1}^{N} \Rightarrow \{(\boldsymbol{e},\omega)\}_{i=1}^{N_1} \ldots \{(\boldsymbol{e},\omega)\}_{i=1}^{N_B}$$

majority vote

$$f_1(\boldsymbol{e}), \ldots, f_B(\boldsymbol{e})$$

Random Forests = Bagging with Decision Trees

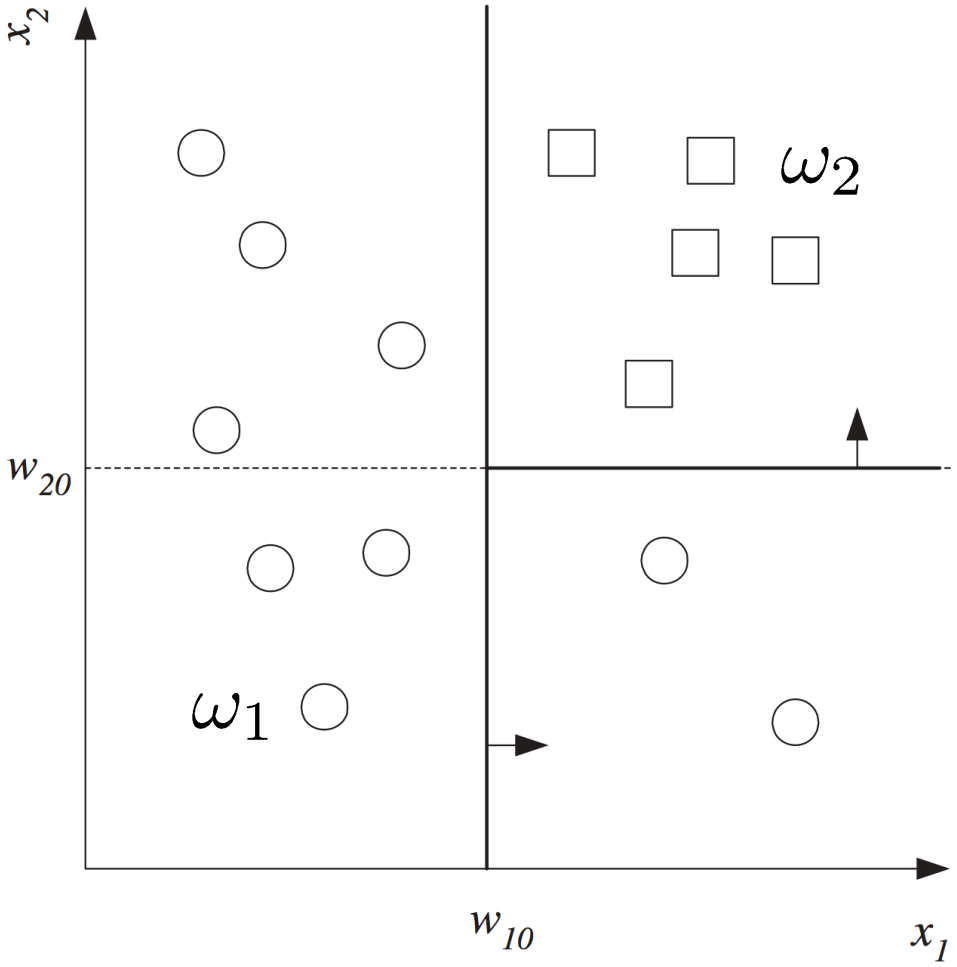
- Combine multiple decision trees into a forests of decision trees



Previous Lecture: Decision Trees for Classification

- **Idea: sequences of splits** of the input space define **regions that correspond to classes**
- Tree is built in a divide-and-conquer approach based on the training data
- Each node realizes a split in feature space (split node) or represents a classification output (leaf node)

Previous Lecture: Decision Tree Example



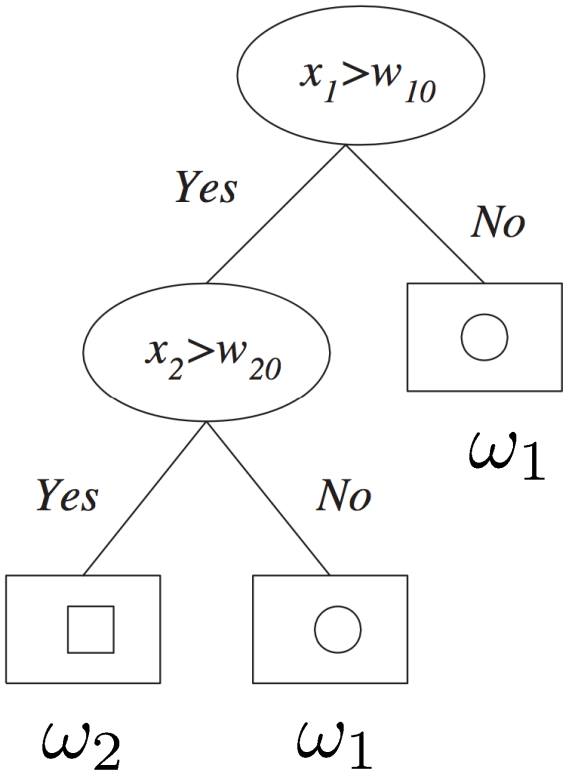


Image courtesy: Aplaydin

Random Forest Learning

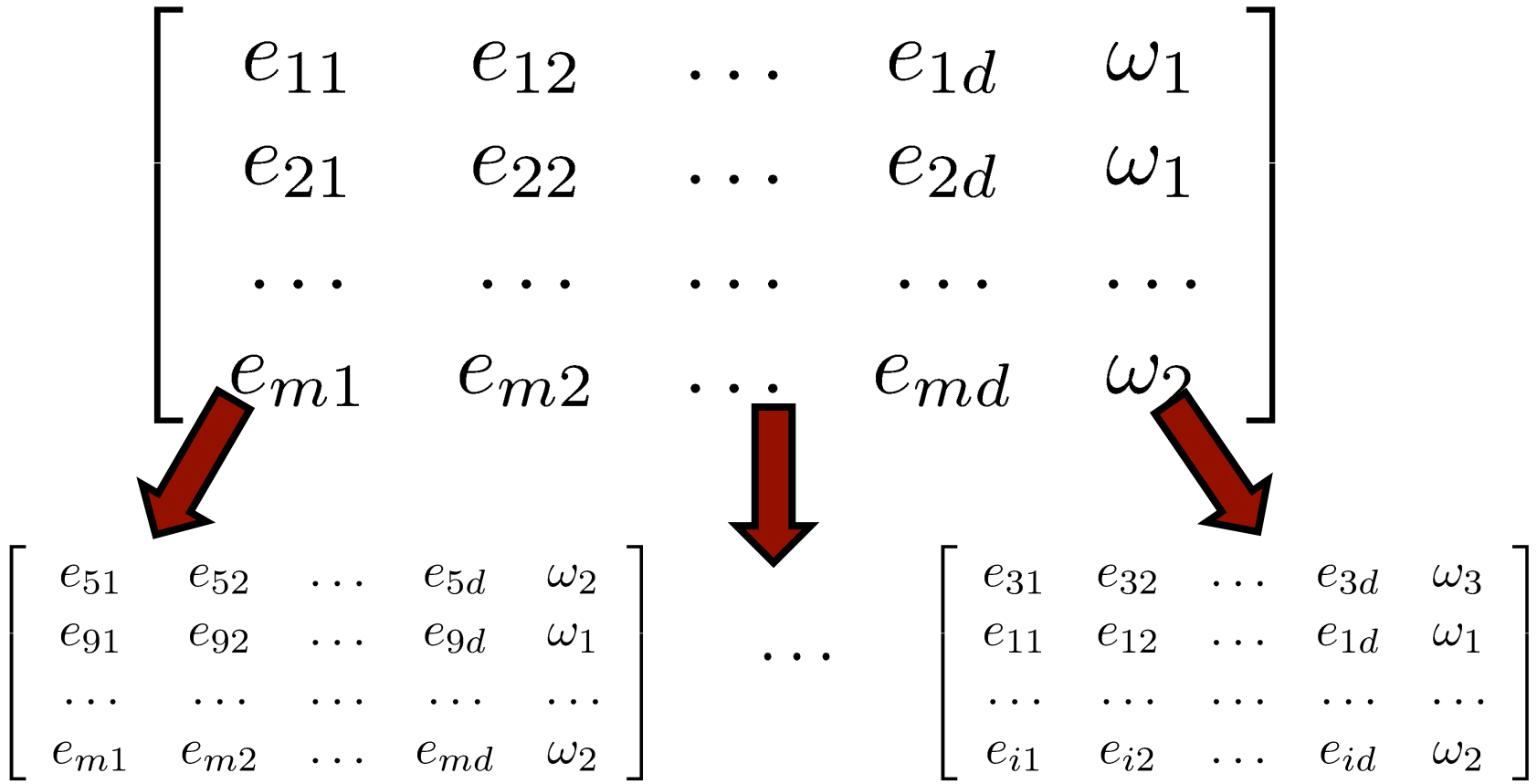
1. Randomly split the training data

$$\begin{bmatrix} e_{11} & e_{12} & \dots & e_{1d} & \omega_1 \\ e_{21} & e_{22} & \dots & e_{2d} & \omega_1 \\ \dots & \dots & \dots & \dots & \dots \\ e_{m1} & e_{m2} & \dots & e_{md} & \omega_2 \end{bmatrix}$$

"Bootstrapping" step

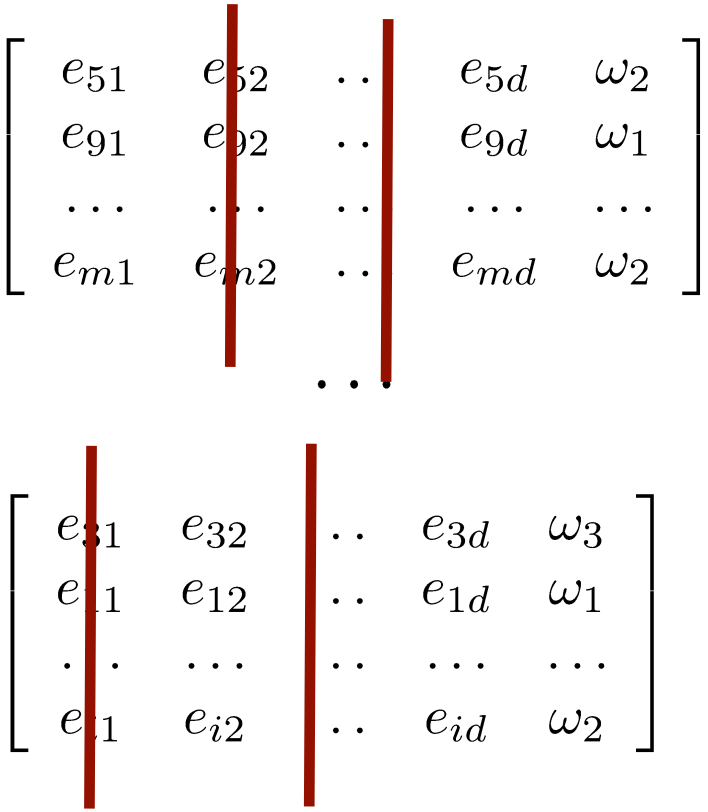
Random Forest Learning

1. Randomly split the training data



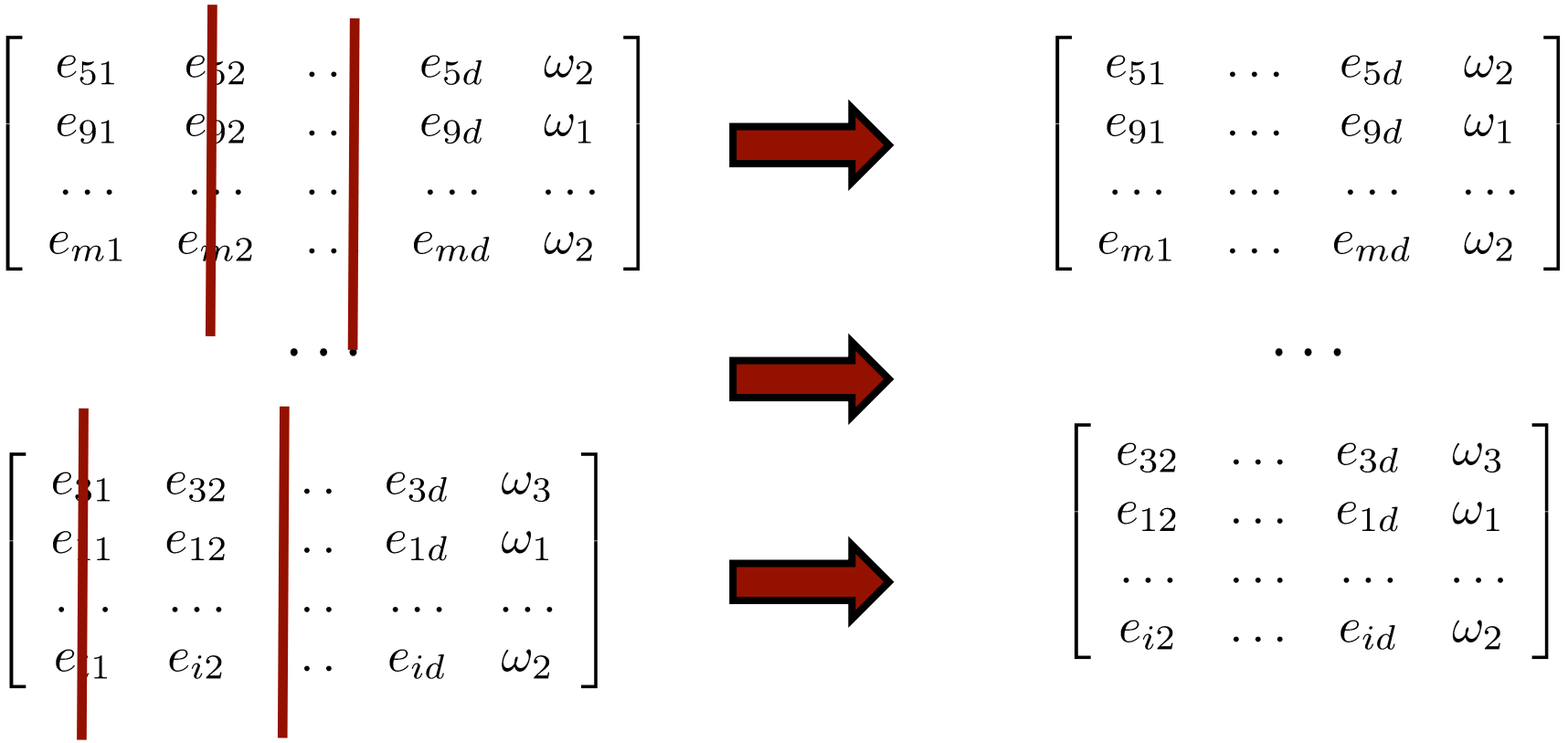
Random Forest Learning

2. Randomly subsample the dimensions in the training data subsets



Random Forest Learning

2. Randomly subsample the dimensions in the training data subsets



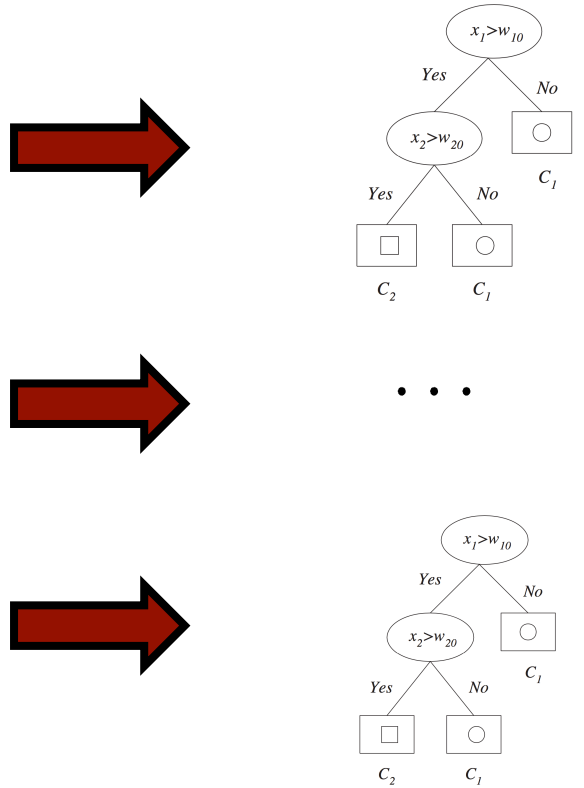
Random Forest Learning

3. Learn standard decision trees based on sampled training data subsets

$$\begin{bmatrix} e_{51} & \dots & e_{5d} & \omega_2 \\ e_{91} & \dots & e_{9d} & \omega_1 \\ \dots & \dots & \dots & \dots \\ e_{m1} & \dots & e_{md} & \omega_2 \end{bmatrix}$$

$$\dots$$

$$\begin{bmatrix} e_{32} & \dots & e_{3d} & \omega_3 \\ e_{12} & \dots & e_{1d} & \omega_1 \\ \dots & \dots & \dots & \dots \\ e_{i2} & \dots & e_{id} & \omega_2 \end{bmatrix}$$

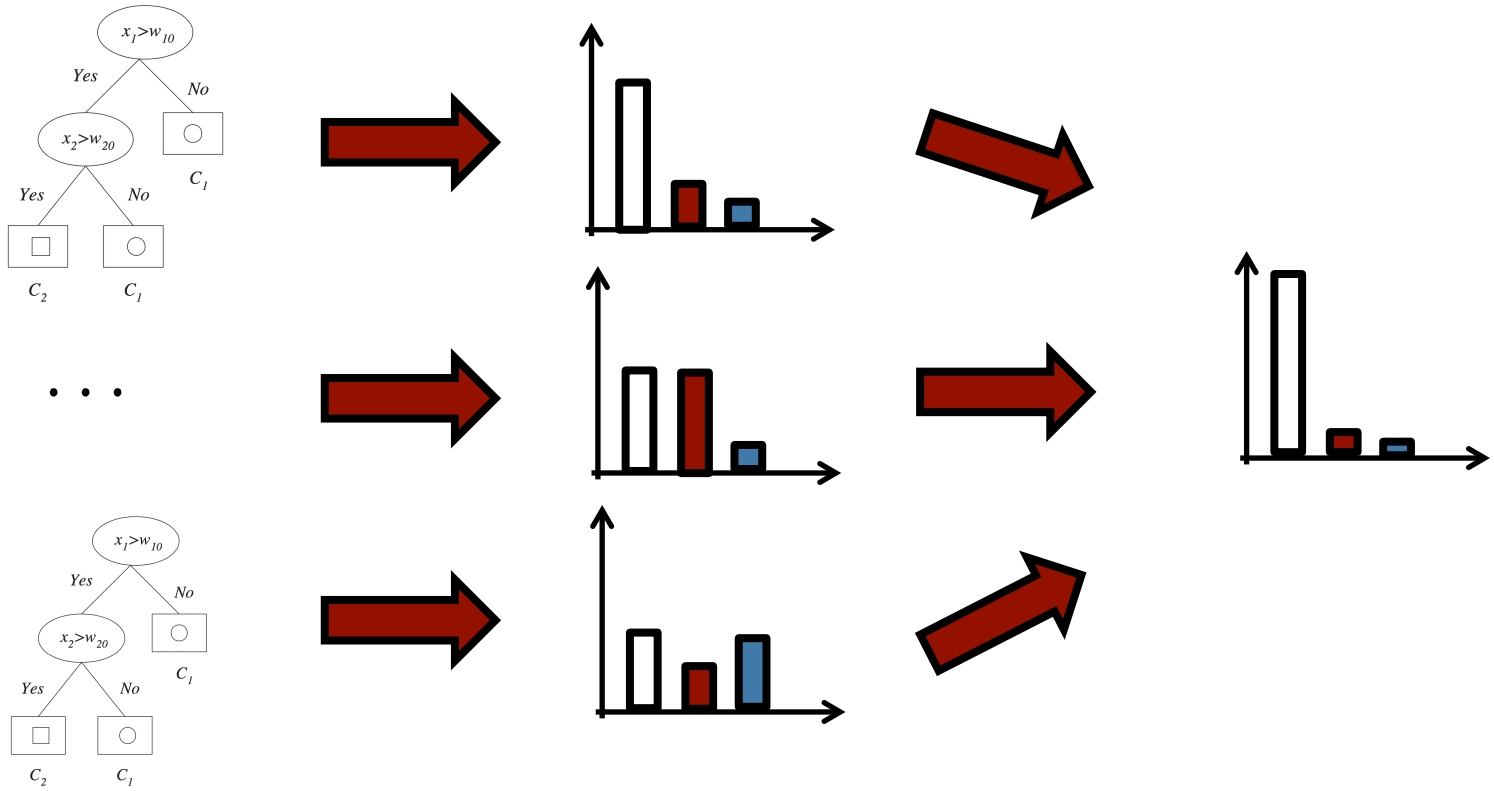


$\dots$

forest

Random Forest Classification

- Classify the test sample with all trees

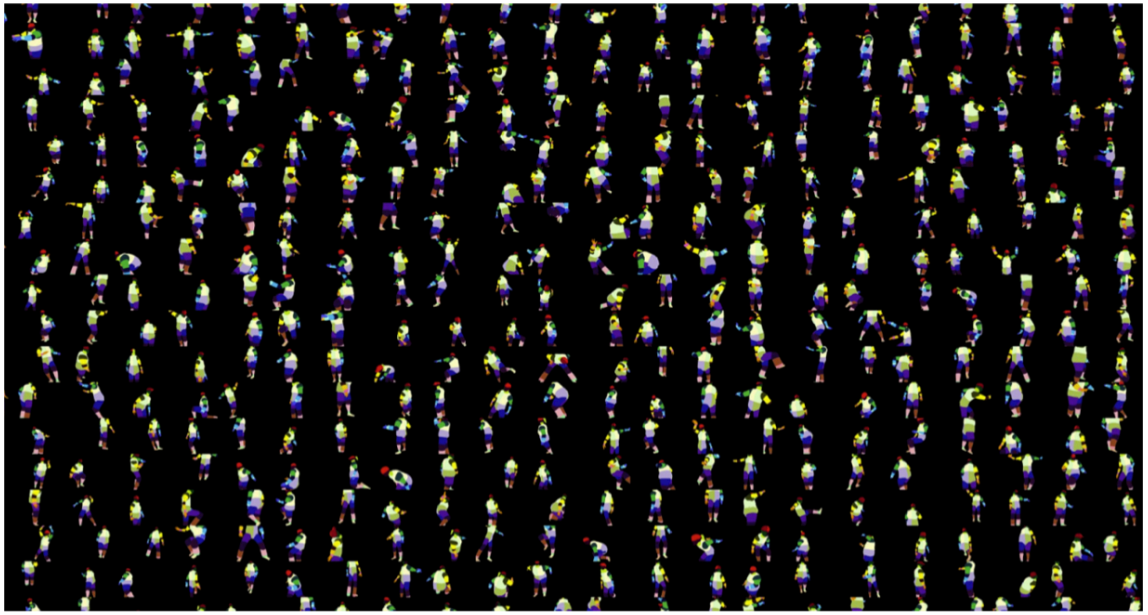


Properties

- Naturally reduces the risk of overfitting
- Provide comparably accurate classification results
- Easy to parallelize
- Efficient GPU implementations exist
- As easy to implement as decision trees

Random Forest Showcase

- Body part classification in the Kinect



1 million test images, 1 day using a 1000 core cluster

Random Forest Summary

- Combines a set of decision trees into a decision forest
- Each tree is learned using a subsampled training set
- Subsampling of training examples and feature dimensions
- Bagging with decision trees
- Frequently used in classification today

AdaBoost

Boosting

- Incremental ensemble building
- Train new model instances to emphasize the training data instances that previous models misclassified

$$c = f(\boldsymbol{e}) \quad \Rightarrow \quad c = \frac{1}{B}\sum_{i=1}^{B} w_i \, f_i(\boldsymbol{e})$$

incrementally learned

Boosting with AdaBoost

- Learn an accurate **strong classifier** by combining an ensemble of inaccurate "rules of thumb"
- **Inaccurate rule** $h(x)$: "weak" classifier, weak learner, classifier
- **Accurate rule** $H(x)$: "strong" classifier, final classifier

AdaBoost

- Most popular algorithm for this type of problem [Freund et al. 95], [Schapire et al. 99]
- Given an ensemble of weak classifiers $h(\boldsymbol{x})$, the combined strong classifier $H(\boldsymbol{x})$ is obtained by a **weighted majority voting scheme**

$$f(\boldsymbol{x}) = \sum_{l=1}^{L} \alpha_l \, h_l(\boldsymbol{x}) \qquad H(\boldsymbol{x}) = \operatorname{sign}\left(f(\boldsymbol{x})\right)$$

Why is AdaBoost Interesting?

It tells us

- What the "best features" are
- What the best thresholds are, and
- How to combine them to a classifier

AdaBoost can be seen as a feature selection strategy

AdaBoost

- AdaBoost is a **non-linear** classifier
- **Generalizes** well: tends to maximize the margin
- Easy to implement

Prerequisite

- Weak classifiers must be better than random guessing
- Error < 0.5 in a binary classification problem

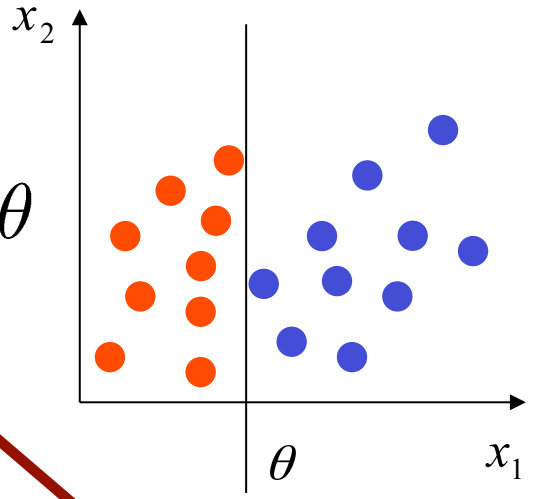
Possible Weak Classifiers

- **Decision stump**:
  Single axis-parallel partition of space (popular choice)
- **Decision tree**:
  Hierarchical partition of space
- **Support Vector Machines** (SVM):
  Best separating hyperplane
- …

Decision Stump

- Trivial decision tree
- Equivalent to linear classifier defined by a hyperplane
- Plane is orthogonal to the j-th axis (with which it intersects in threshold θ)
- Formally,

$$h(\boldsymbol{x}, j, \theta) = \begin{cases} +1 & \text{if } \; p_j \, \boldsymbol{x}_j > p_j \, \theta \\ -1 & \text{otherwise} \end{cases}$$



-1/+1: inequality direction

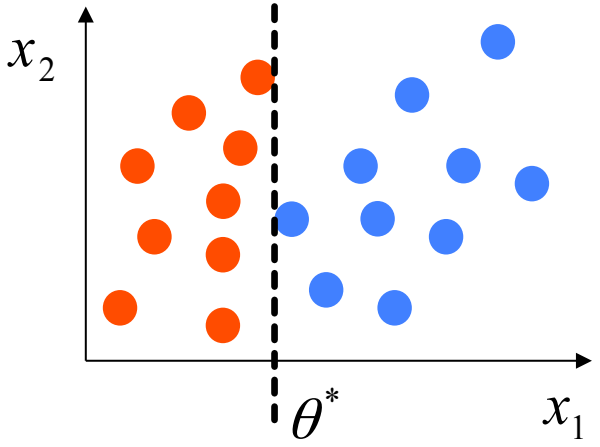
j-th dimension

Decision Stump Training

- Train a decision stump on weighted data

$$(j^*, \theta^*) = \arg\min_{j,\theta} \sum_{t=1}^{T} w(t) I(\omega^t \neq h(\boldsymbol{x}^t))$$

- Finding an optimum parameter θ* for each dimension j =1…d and then select the j* for which the weighted error is minimal

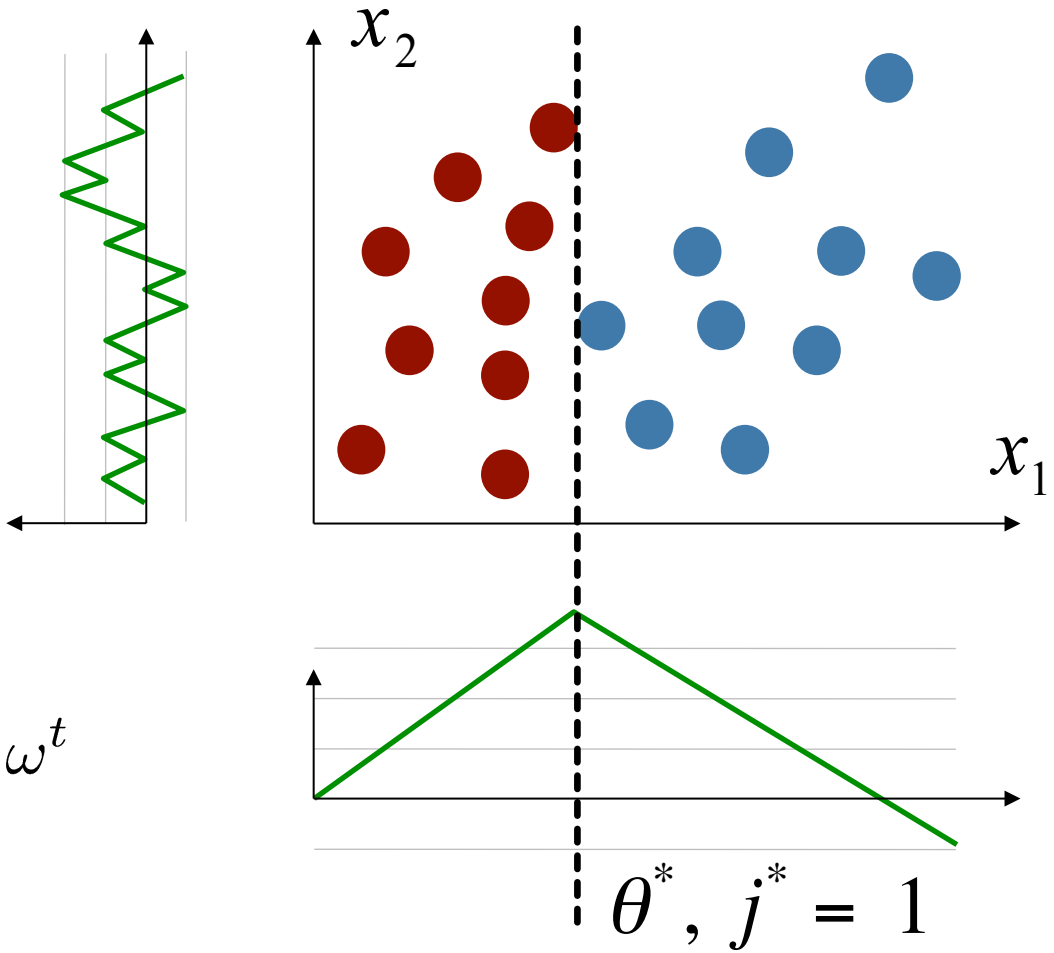


Decision Stump Training

Training algorithm for stumps: Intuition

- **Label:**
  red: +
  blue: –

- Assuming all weights = 1



$$W_j(i) = \sum_{t=1}^{i} w(t)\omega^t$$

Decision Stump Training Algo.

$\forall\ j\ = 1 \ldots d$

Sort samples $\boldsymbol{x}^i$ in ascending order along dimension j

$\forall\ t\ = 1 \ldots T$

Compute T cumulative sums $W_j(i) = \sum_{t=1}^{i} w(t)\omega^t$

end

Threshold θ_j is at extremum of $W_j(i)$

Sign of extremum gives direction p_j of inequality

end

Global extremum in all D sums $W_j(i), j = 1..D$ gives **threshold** θ^* and **dimension** j^*

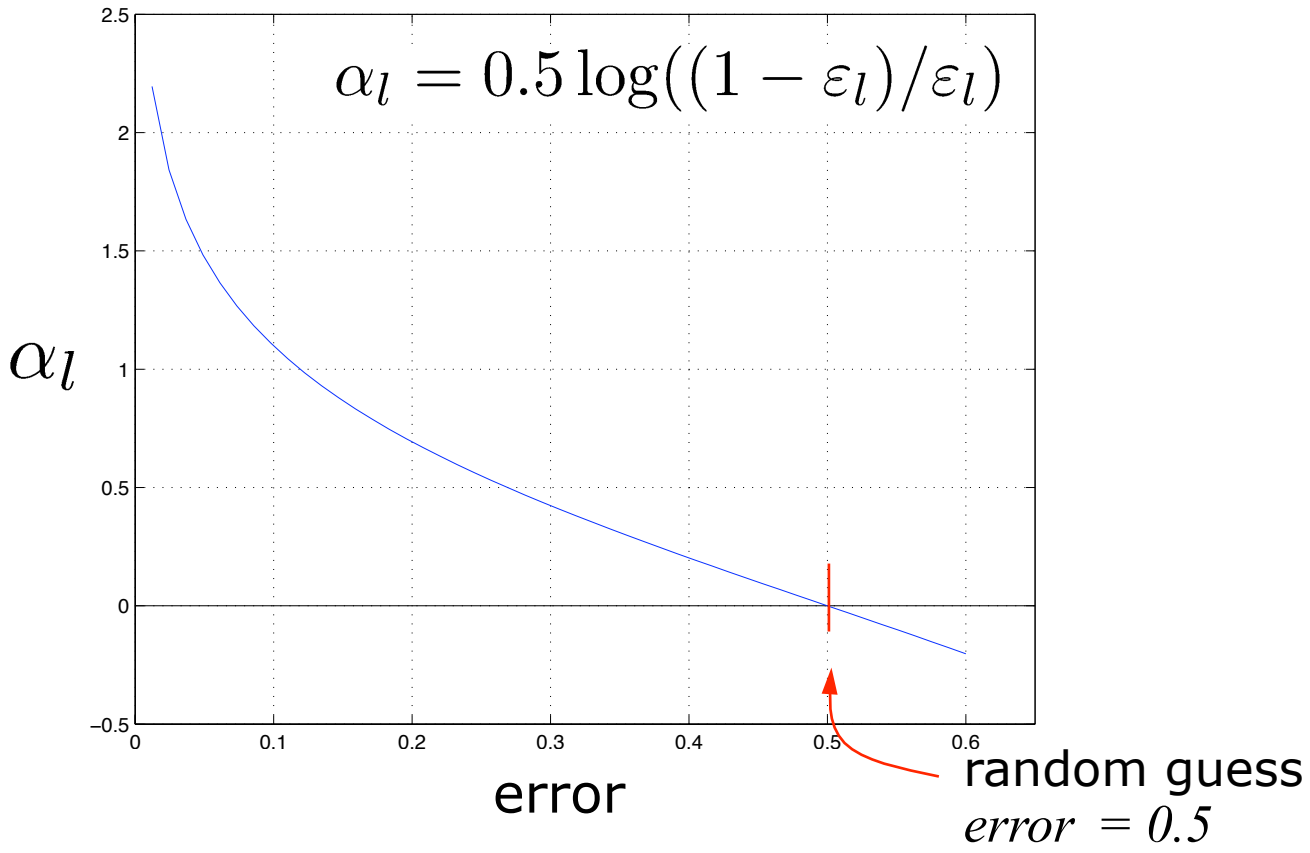
AdaBoost Algorithm

Given the **training data** $\{(\boldsymbol{x}^1, \omega^1), ..., (\boldsymbol{x}^T, \omega^T)\}$

1. Initialize weights $w(t) = 1/T$
2. For $l = 1,...,L$
   - Train **weak classifiers** $h(\boldsymbol{x})$ on weighted training data
   - Select classifier minimizing $\varepsilon_l = \sum_{t=1}^{T} w(t) I(\omega^t \neq h_l(\boldsymbol{x}^t))$
   - Compute **voting weight** of $h_l(\boldsymbol{x})$: $\alpha_l = 0.5 \log((1 - \varepsilon_l)/\varepsilon_l)$
   - Recompute weights: $w(t) = w(t) \exp(-\alpha_l \omega^t h_l(\boldsymbol{x}^t))/Z_l$
3. Combine results to a **strong classifier**

AdaBoost: Voting Weight

- The **voting weight** α_l of a weak classifier $h_l(\boldsymbol{x})$ quantifies its importance



AdaBoost: Training Data Weight Update

- Update step:

$$w(t) = w(t)\frac{\exp(\overset{\begin{cases}<1 & \text{if } \omega^t = h_l(\boldsymbol{x}^t) \\ >1 & \text{otherwise}\end{cases}}{-}\alpha_l \omega^t h_l(\boldsymbol{x}^t))}{Z_l}$$

previous weight

normalizer so that all weights sum up to 1

- Weights of misclassified training samples are increased
- Weights of correctly classified samples are decreased

AdaBoost: Training Data Weight Update

- Weights of misclassified training samples are increased
- Weights of correctly classified samples are decreased
- In each iteration, AdaBoost puts **more weight on the misclassified examples** of the previous weak classifier

AdaBoost: Strong Classifier

- The resulting strong classifier is

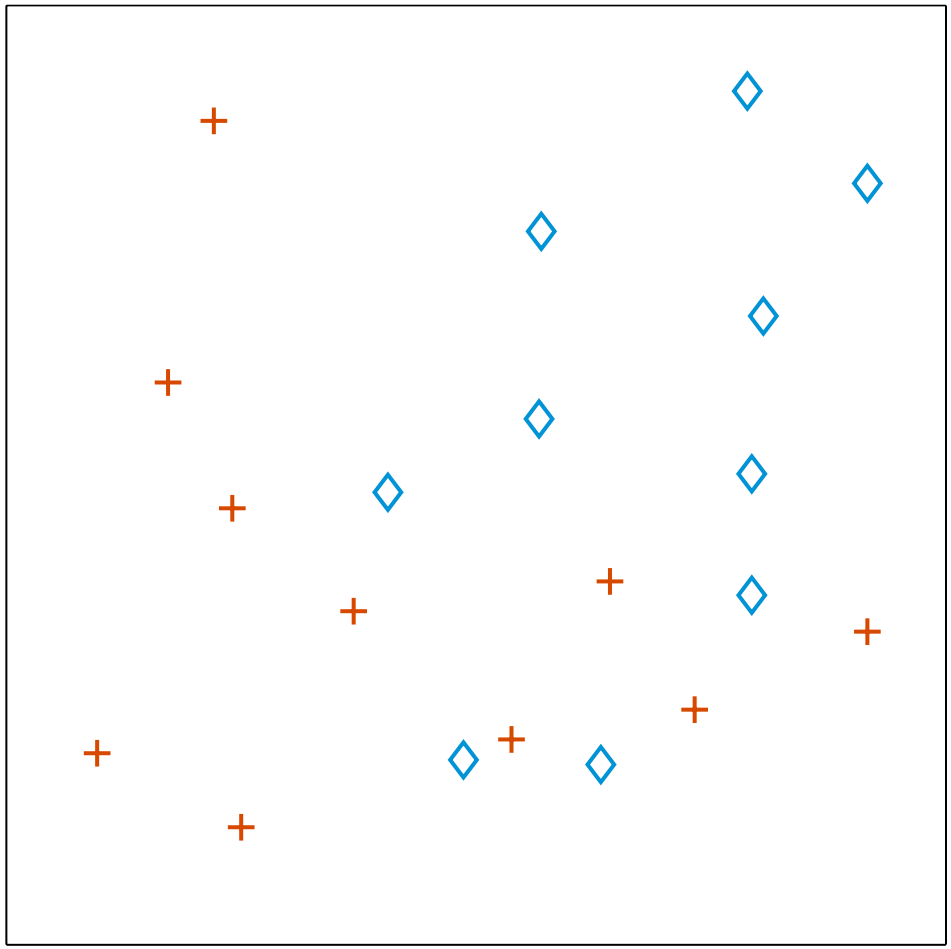
$$H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{l=1}^{L} \alpha_l \, h_l(\boldsymbol{x})\right) \longrightarrow \text{Class Result } \{+1, -1\}$$

Put your data here

- AdaBoost implements a weighted majority voting scheme
- Classification is efficient to implement (sum over simple weak classifiers)

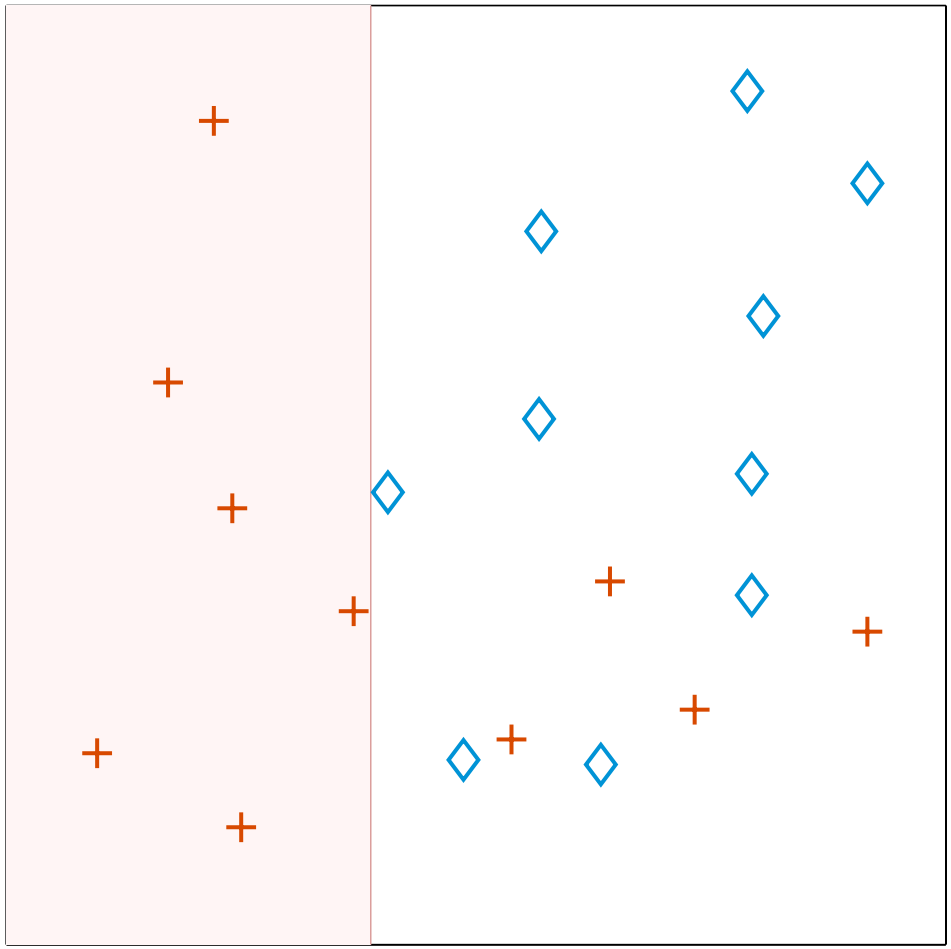
AdaBoost: Step-By-Step

- **Training data**



AdaBoost: Step-By-Step

- **Iteration 1, train weak classifier 1**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

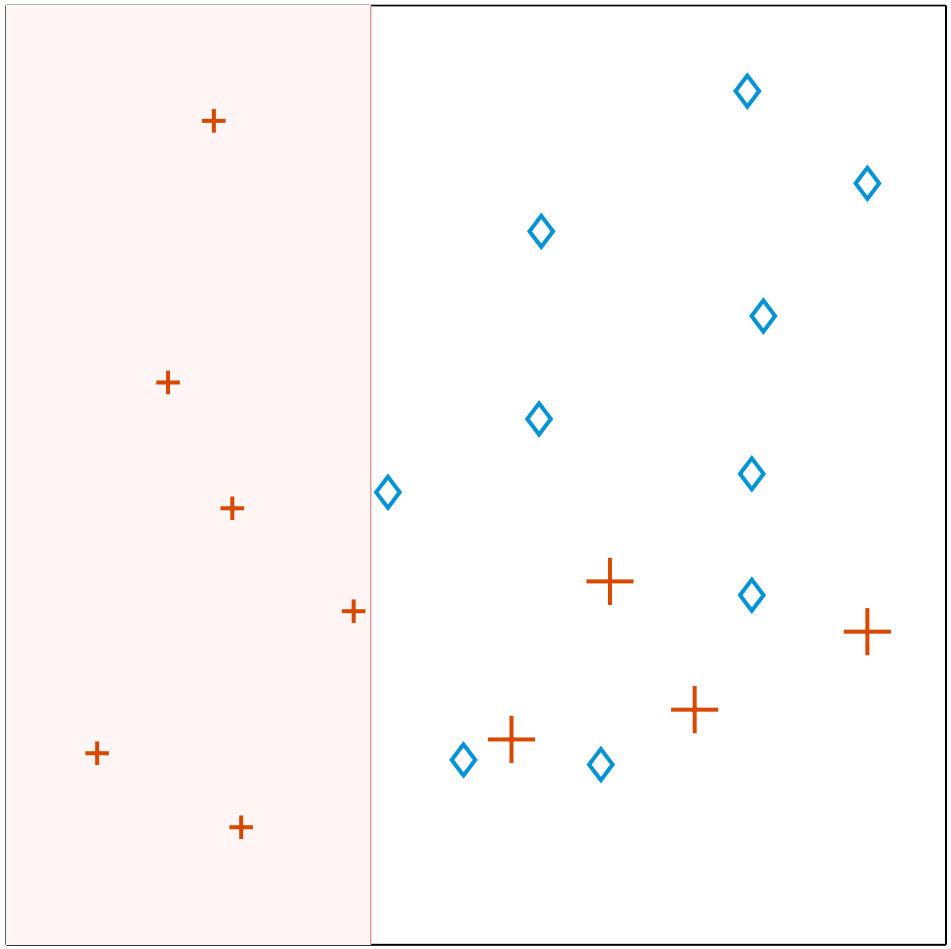
Weighted error
$e_l = 0.2$

Voting weight
$\alpha_l = 1.39$

Total error = 4

AdaBoost: Step-By-Step

- **Iteration 1, recompute weights**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

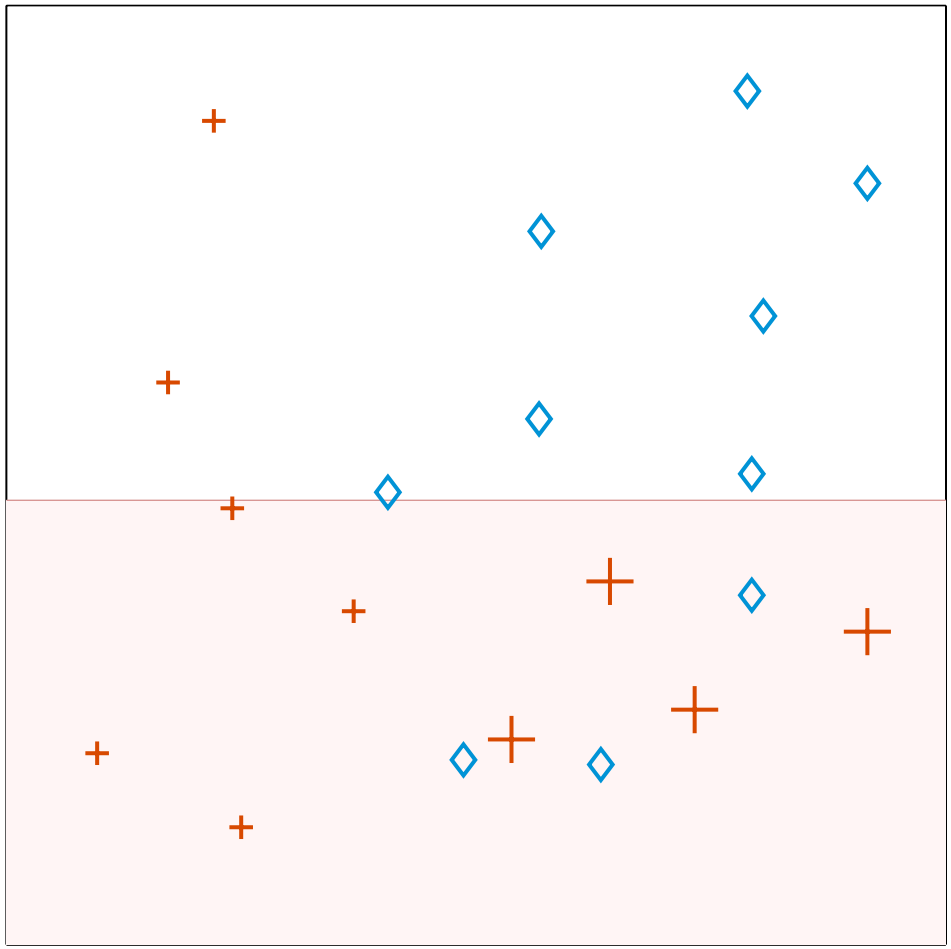
Weighted error
$e_l = 0.2$

Voting weight
$\alpha_l = 1.39$

Total error = 4

AdaBoost: Step-By-Step

- **Iteration 2, train weak classifier 2**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

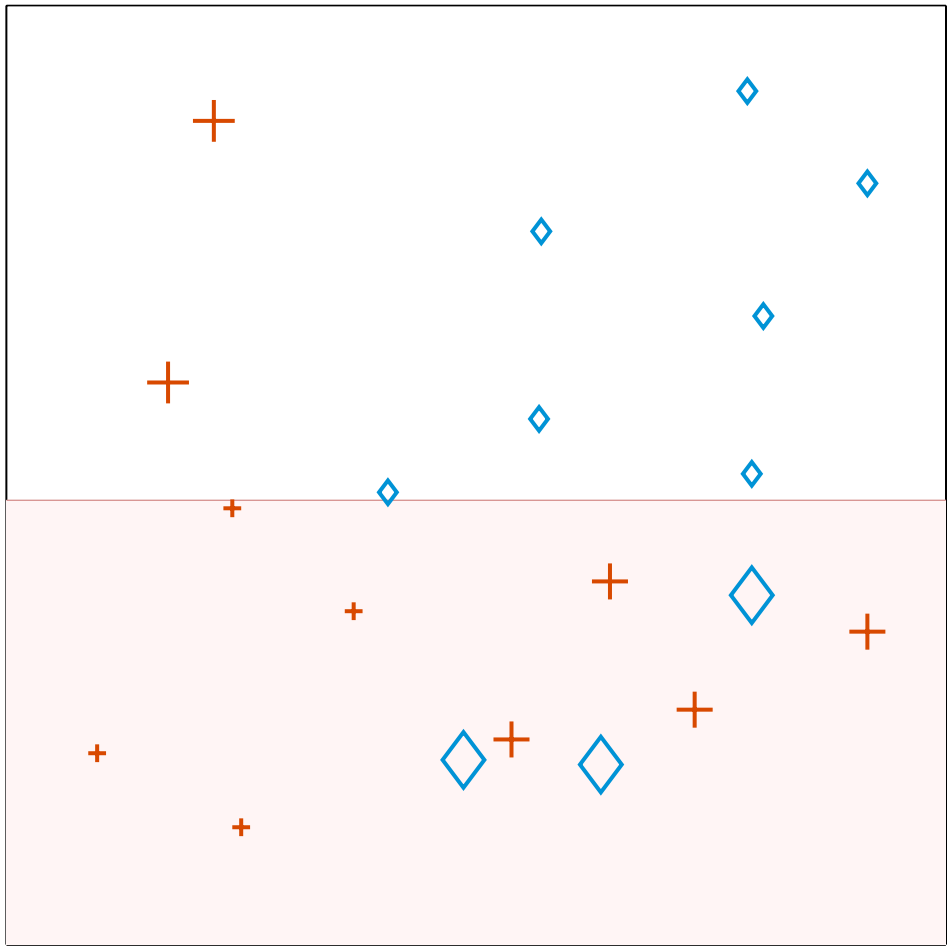
Weighted error
$e_l = 0.16$

Voting weight
$\alpha_l = 1.69$

Total error = 5

AdaBoost: Step-By-Step

- **Iteration 2, recompute weights**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

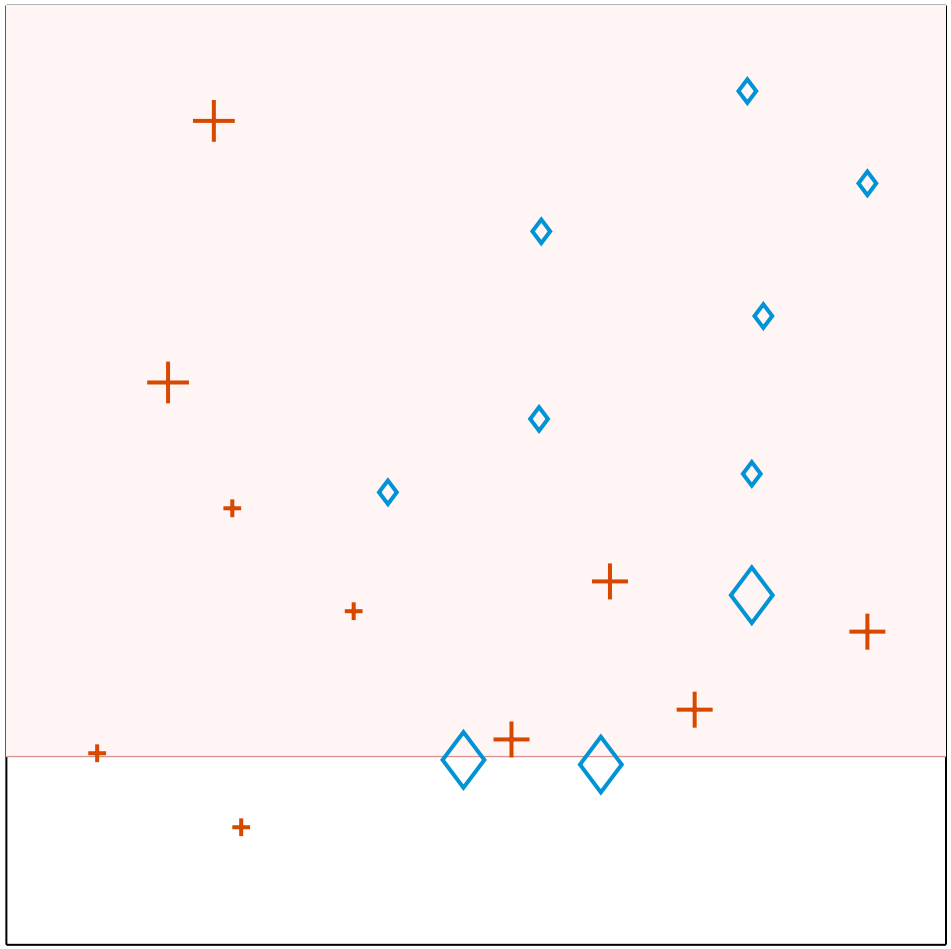
Weighted error
$e_l = 0.16$

Voting weight
$\alpha_l = 1.69$

Total error = 5

AdaBoost: Step-By-Step

- **Iteration 3, train weak classifier 3**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

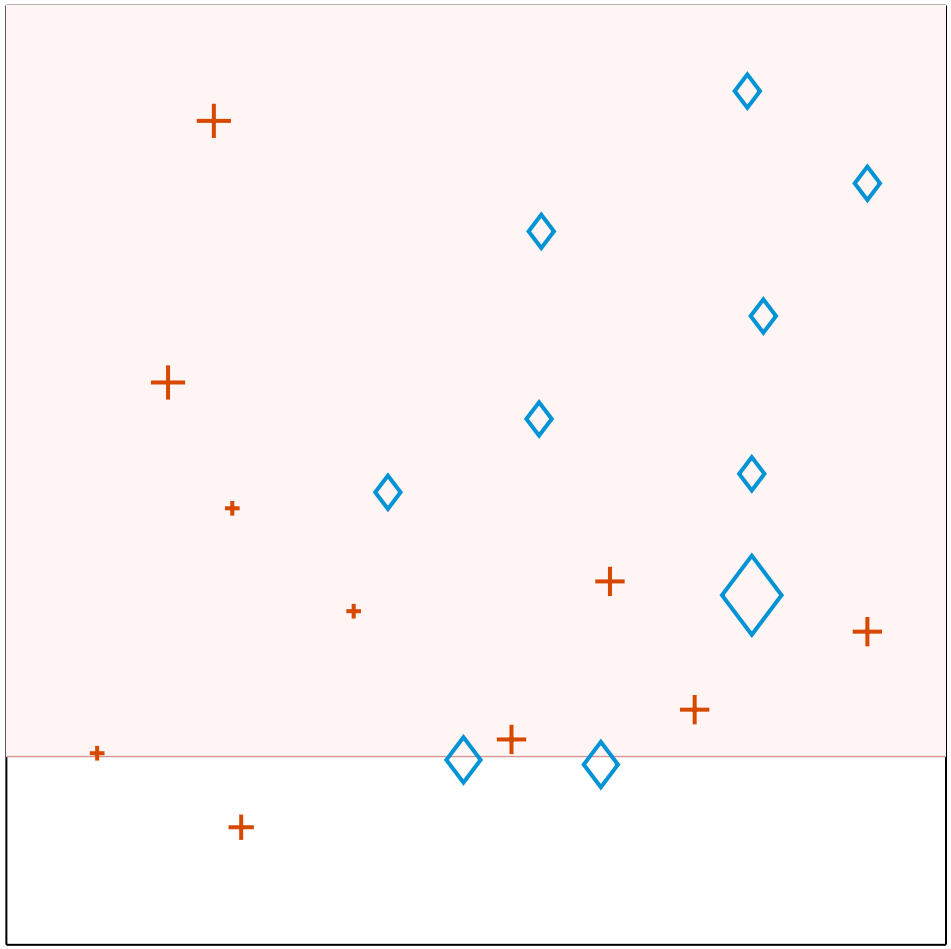
Weighted error
$e_l = 0.25$

Voting weight
$\alpha_l = 1.11$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 3, recompute weights**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

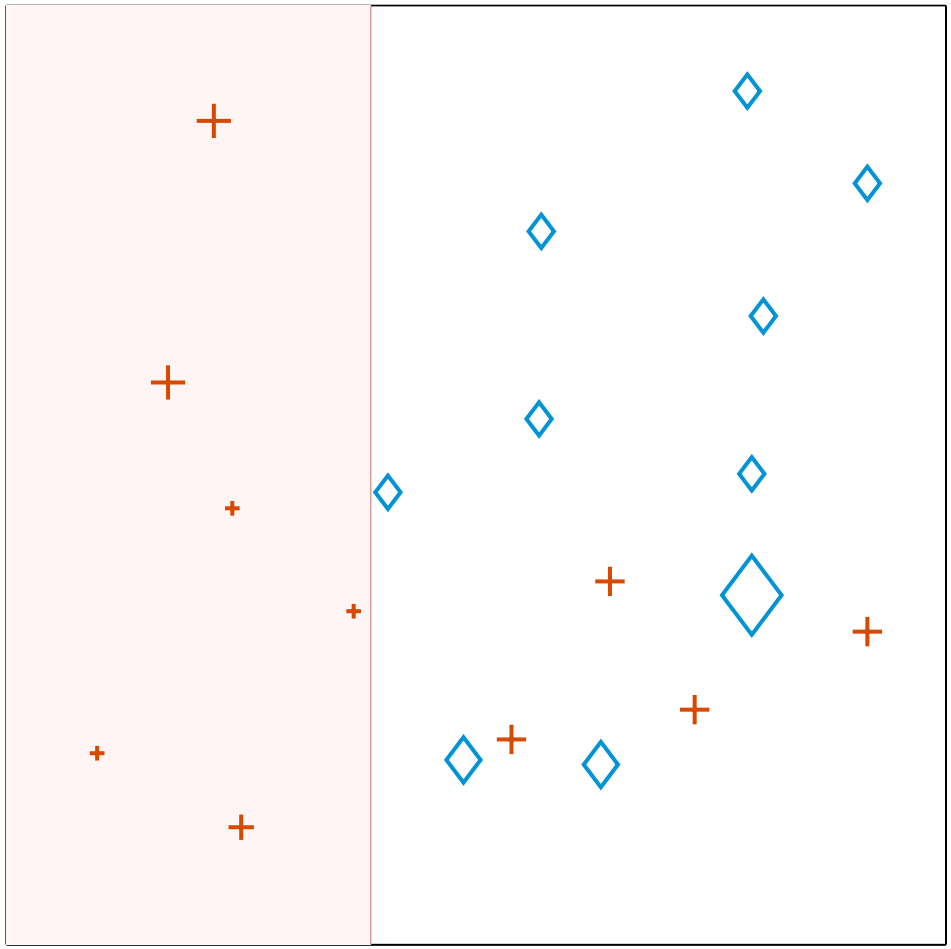
Weighted error
$e_l = 0.25$

Voting weight
$\alpha_l = 1.11$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 4, train weak classifier 4**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

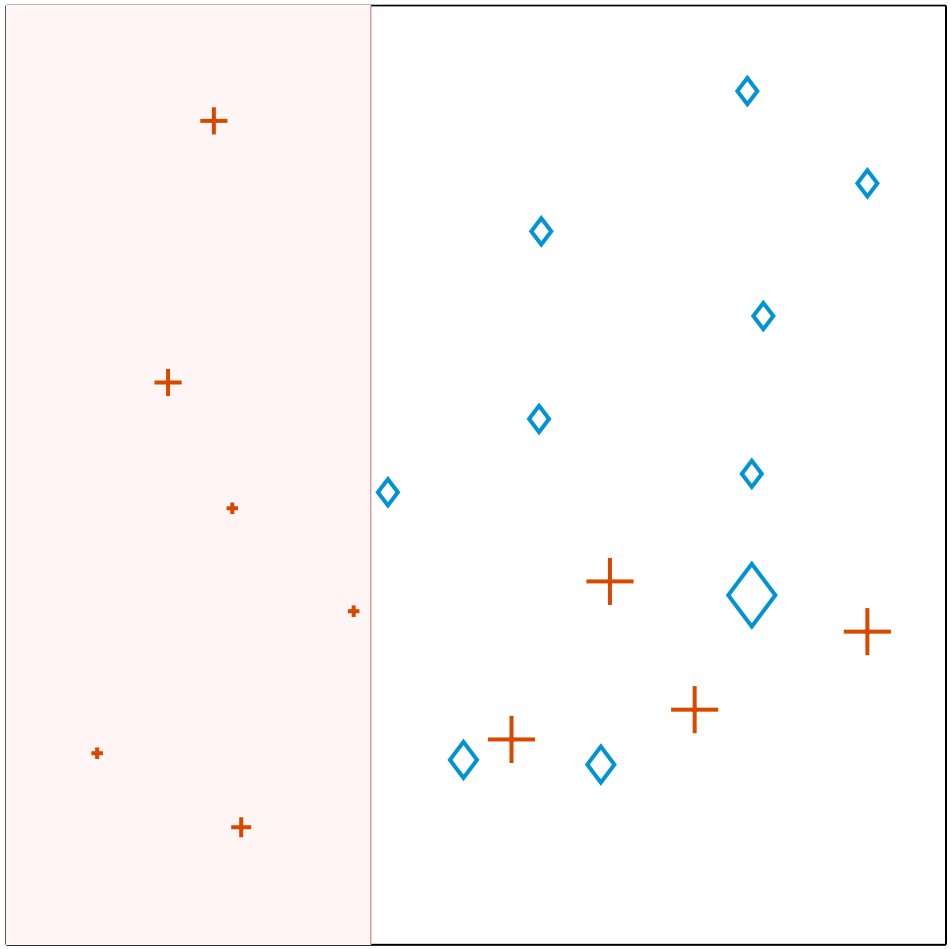
Weighted error
$e_l = 0.20$

Voting weight
$\alpha_l = 1.40$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 4, recompute weights**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

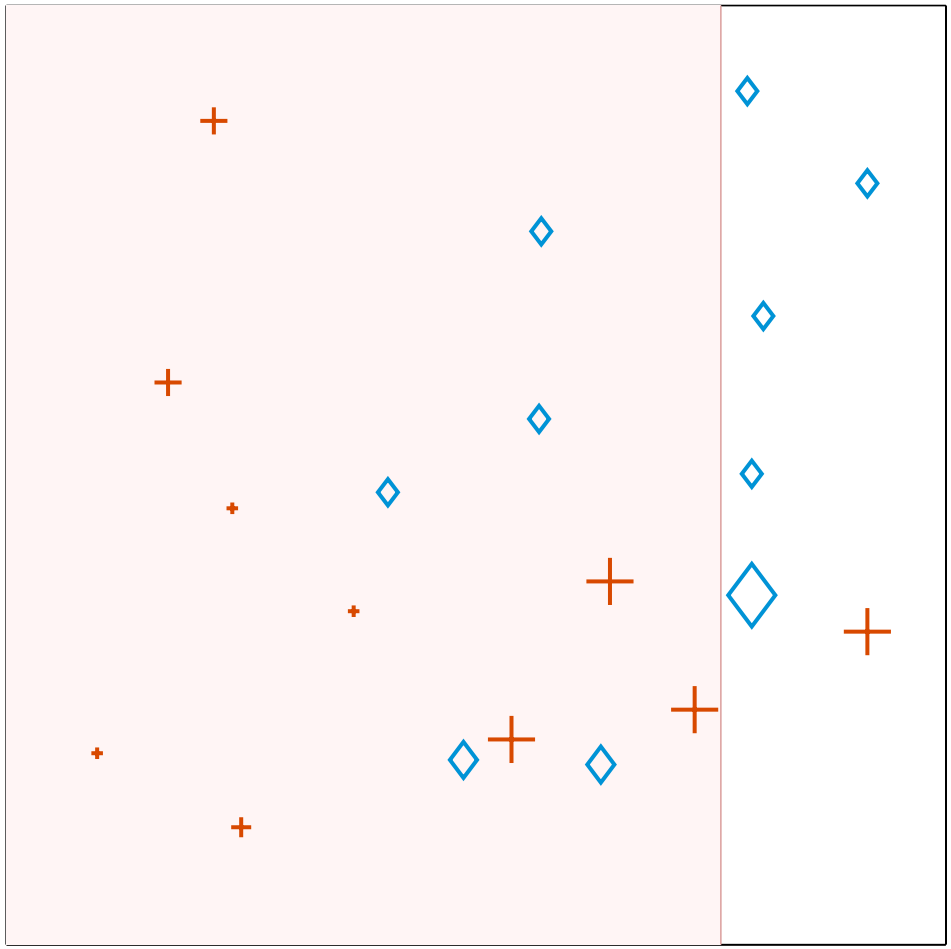
Weighted error
$e_l = 0.20$

Voting weight
$\alpha_l = 1.40$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 5, train weak classifier 5**



Threshold
$\theta^* = 0.81$

Dimension
$j^* = 1$

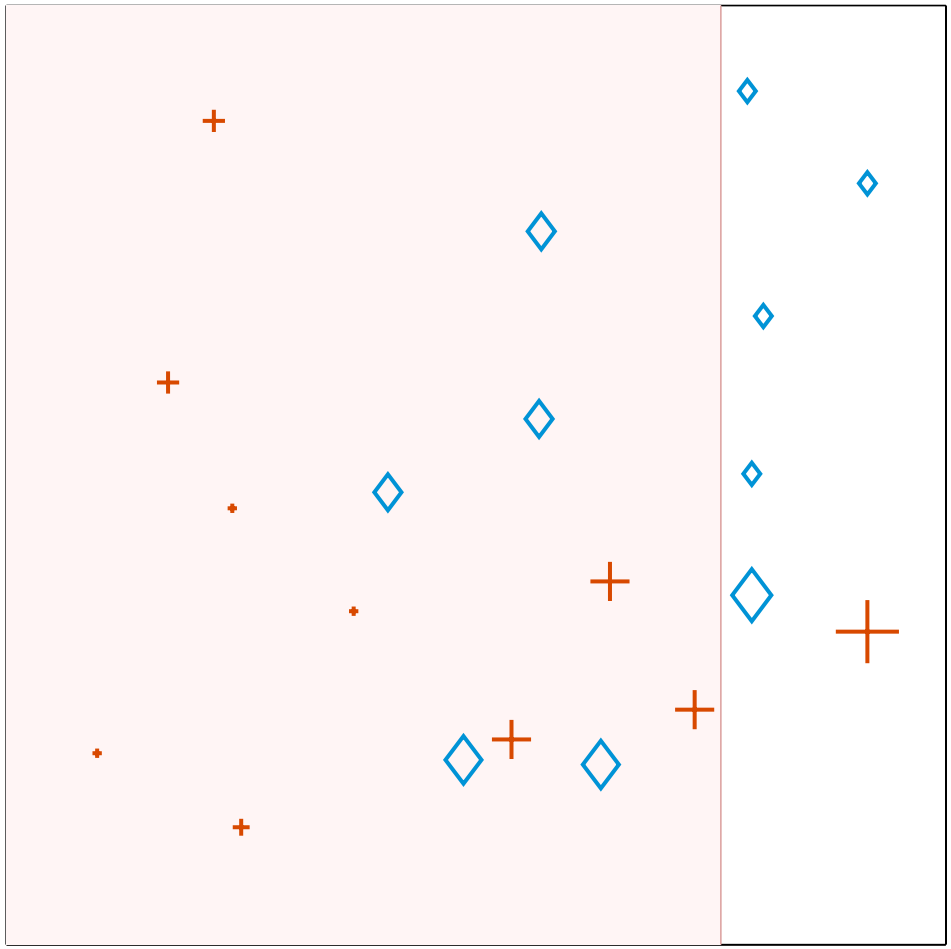
Weighted error
$e_l = 0.28$

Voting weight
$\alpha_l = 0.96$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 5, recompute weights**



Threshold
$\theta^* = 0.81$

Dimension
$j^* = 1$

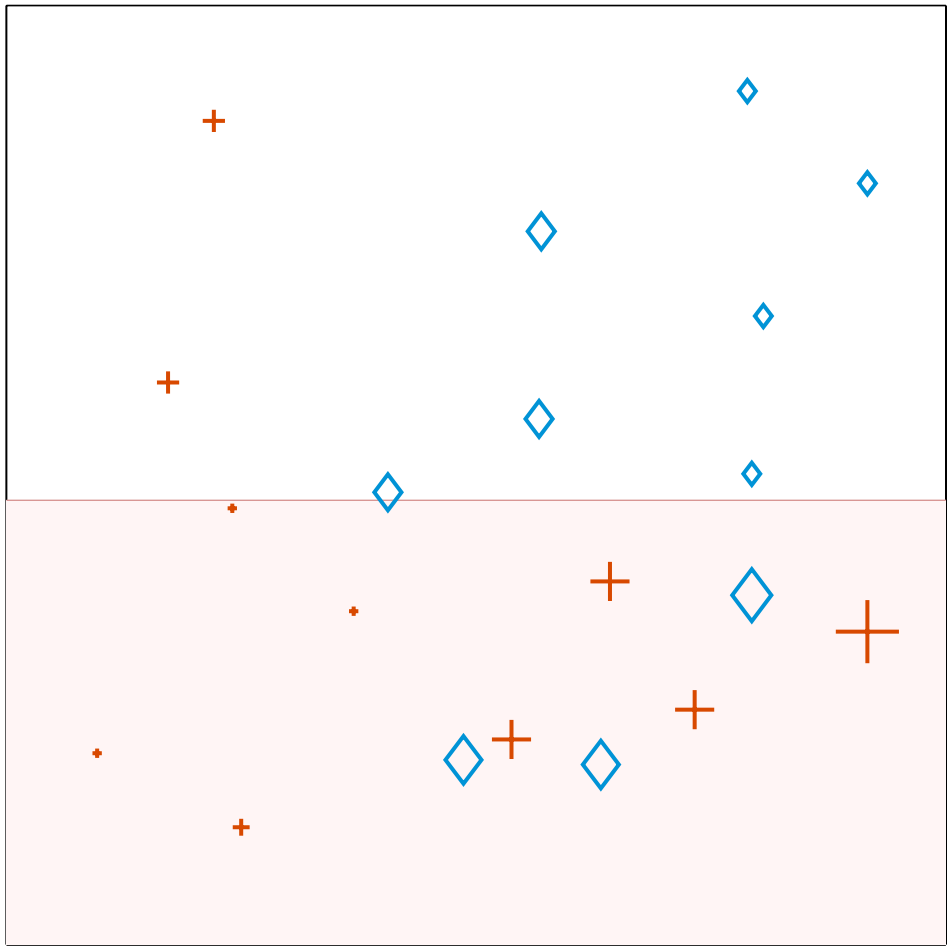
Weighted error
$e_l = 0.28$

Voting weight
$\alpha_l = 0.96$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 6, train weak classifier 6**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

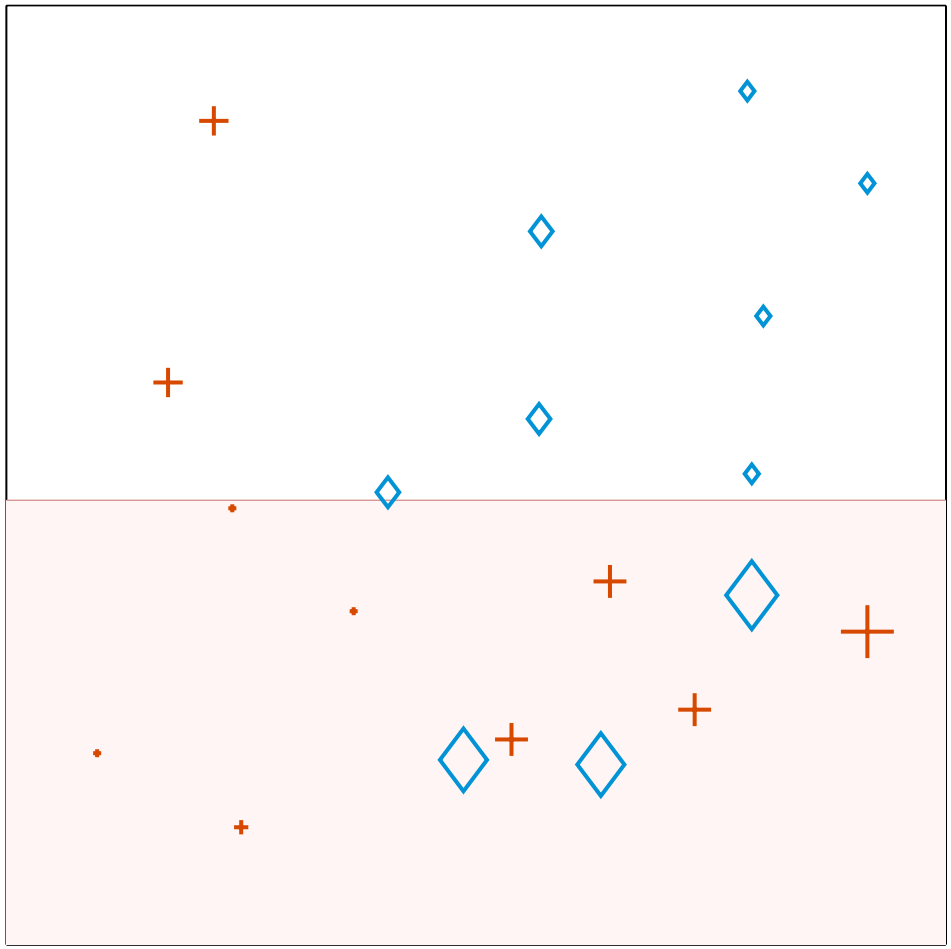
Weighted error
$e_l = 0.29$

Voting weight
$\alpha_l = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 6, recompute weights**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

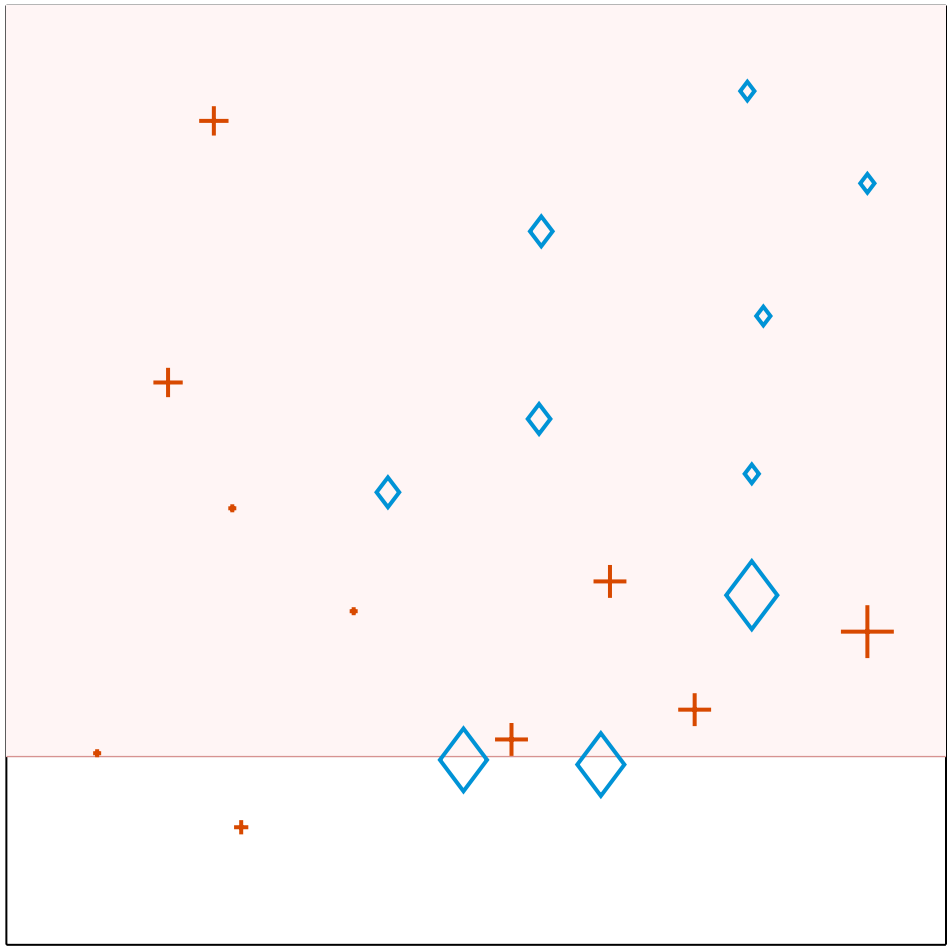
Weighted error
$e_l = 0.29$

Voting weight
$\alpha_l = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 7, train weak classifier 7**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

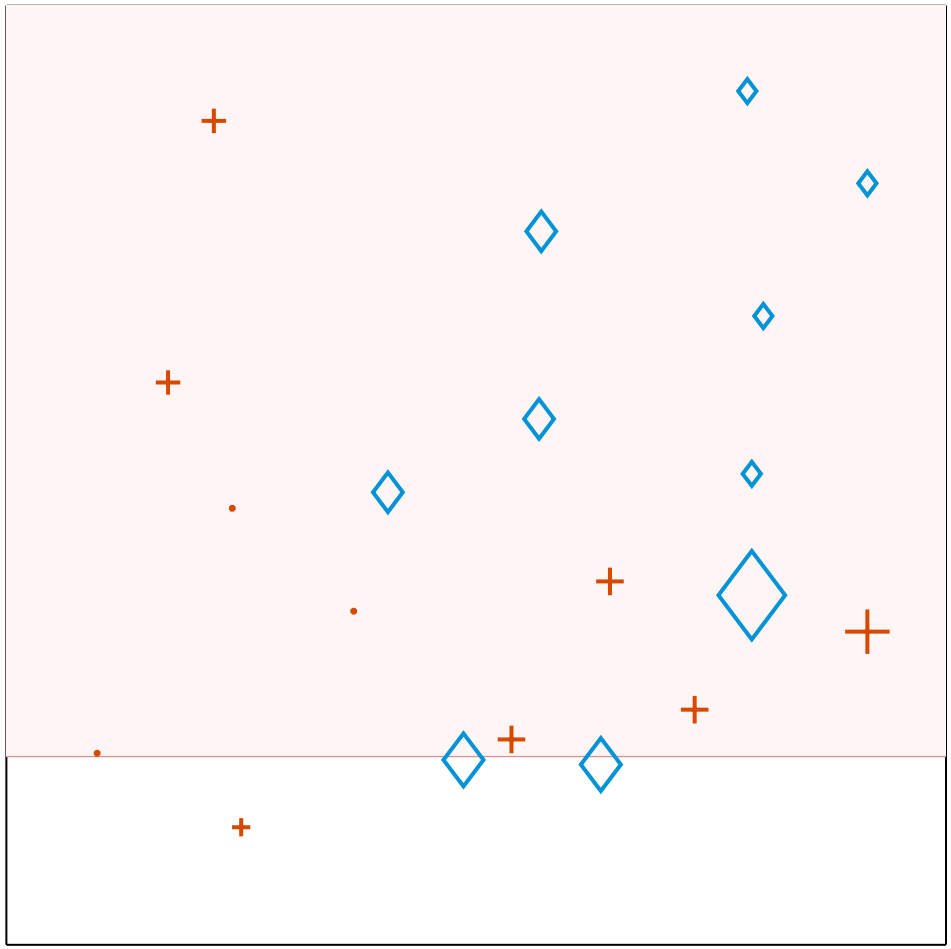
Weighted error
$e_l = 0.29$

Voting weight
$\alpha_l = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 7, recompute weights**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

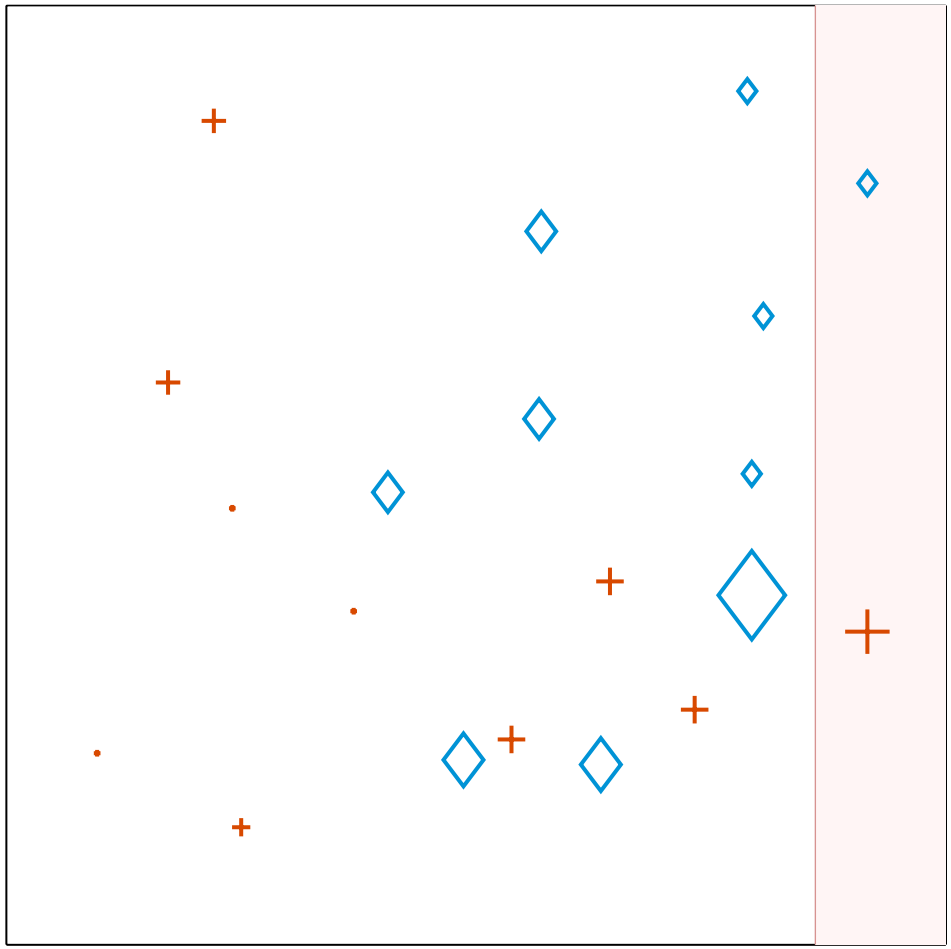
Weighted error
$e_l = 0.29$

Voting weight
$\alpha_l = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 8, train weak classifier 8**



Threshold
$\theta^* = 0.93$

Dimension, sign
$j^* = 1$, neg

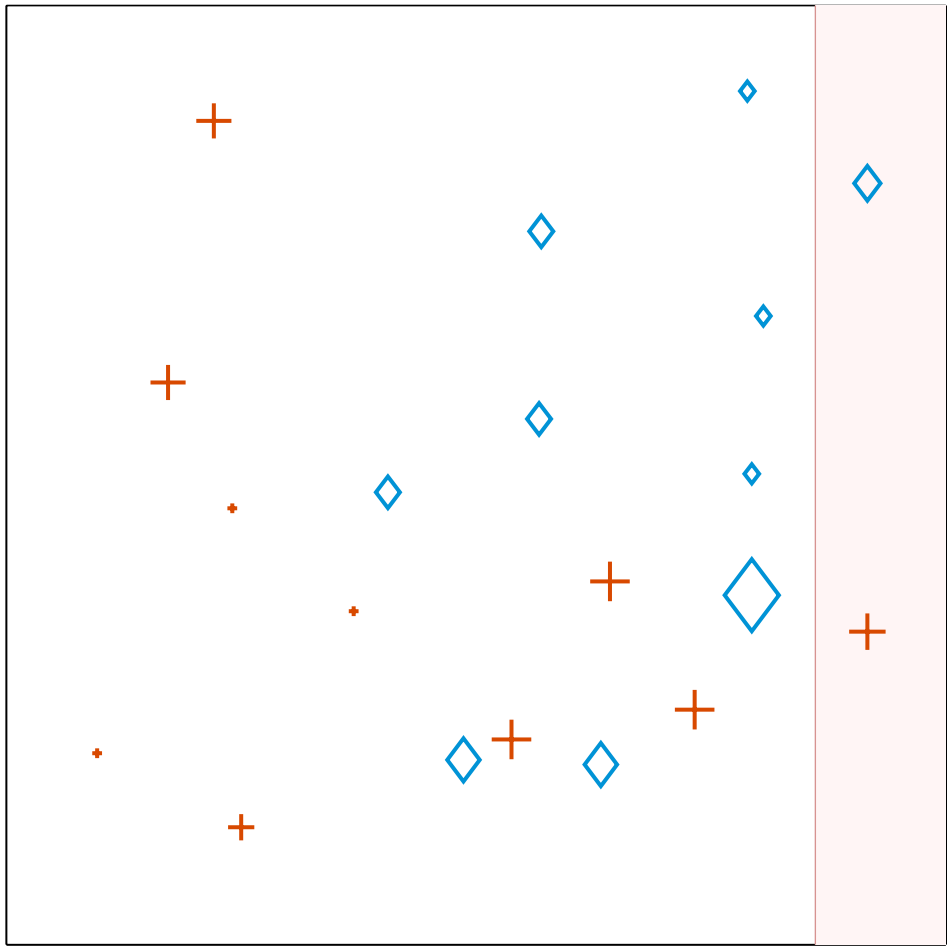
Weighted error
$e_l = 0.25$

Voting weight
$\alpha_l = 1.12$

Total error = 0

AdaBoost: Step-By-Step

- **Iteration 8, recompute weights**



Threshold
$\theta^* = 0.93$

Dimension, sign
$j^* = 1$, neg

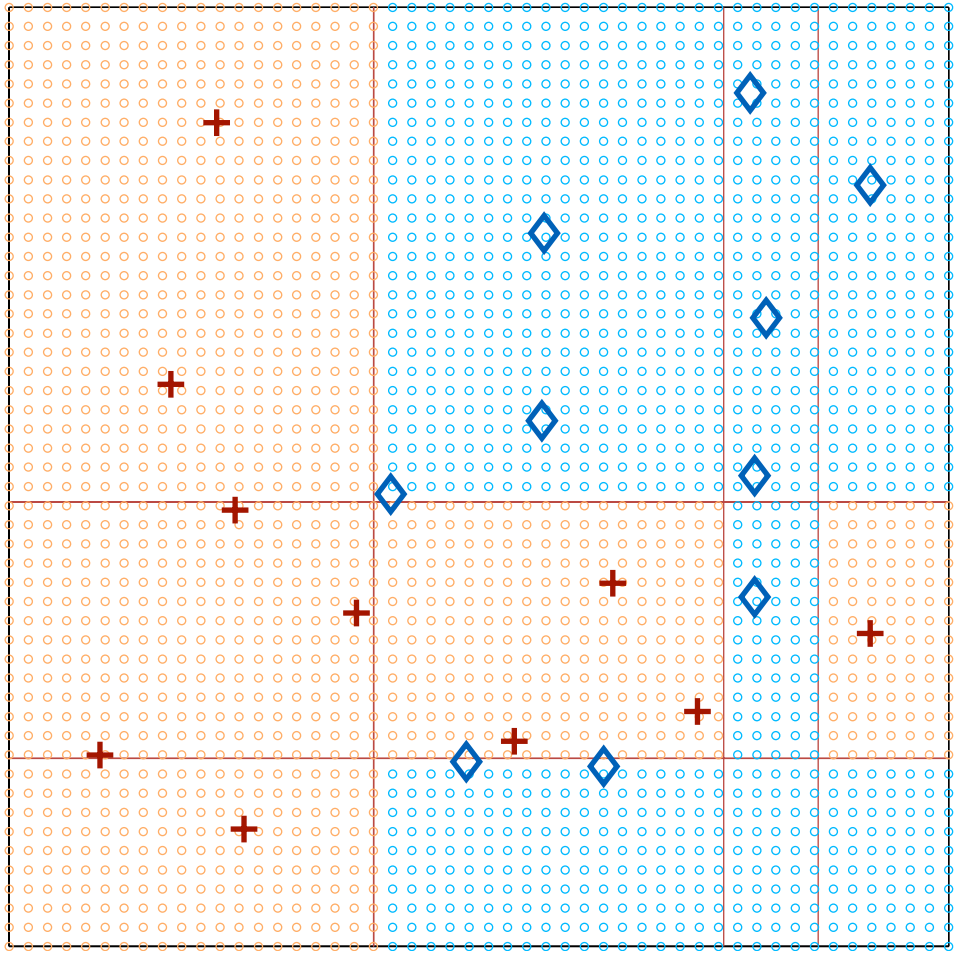
Weighted error
$e_l = 0.25$

Voting weight
$\alpha_l = 1.12$

Total error = 0

AdaBoost: Step-By-Step

- **Final Strong Classifier**

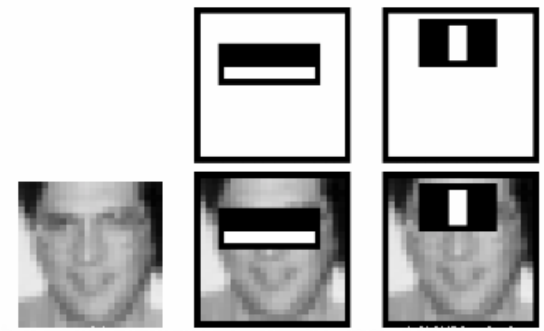


Total training error = 0

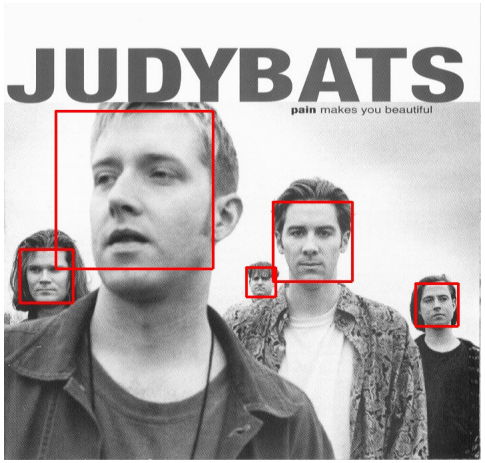
(rare in practice)

AdaBoost Showcase

- Face detection by Viola & Jones



+ AdaBoost



AdaBoost Summary

- Combines multiple weak classifiers into a strong one
- Incrementally adds weak classifiers
- Reweights the training data in each step to focus in wrongly classified examples
- AdaBoost for face detection became **very popular** (around 2001)

Summary

- Evaluation classifiers
- Generalization
- Ideas of Boosting and Bagging
- Random forest classification
- AdaBoost

Literature

- Breitman, Random Forests, 2011
- Alpaydin, Introduction to Machine Learning, Chapter 17