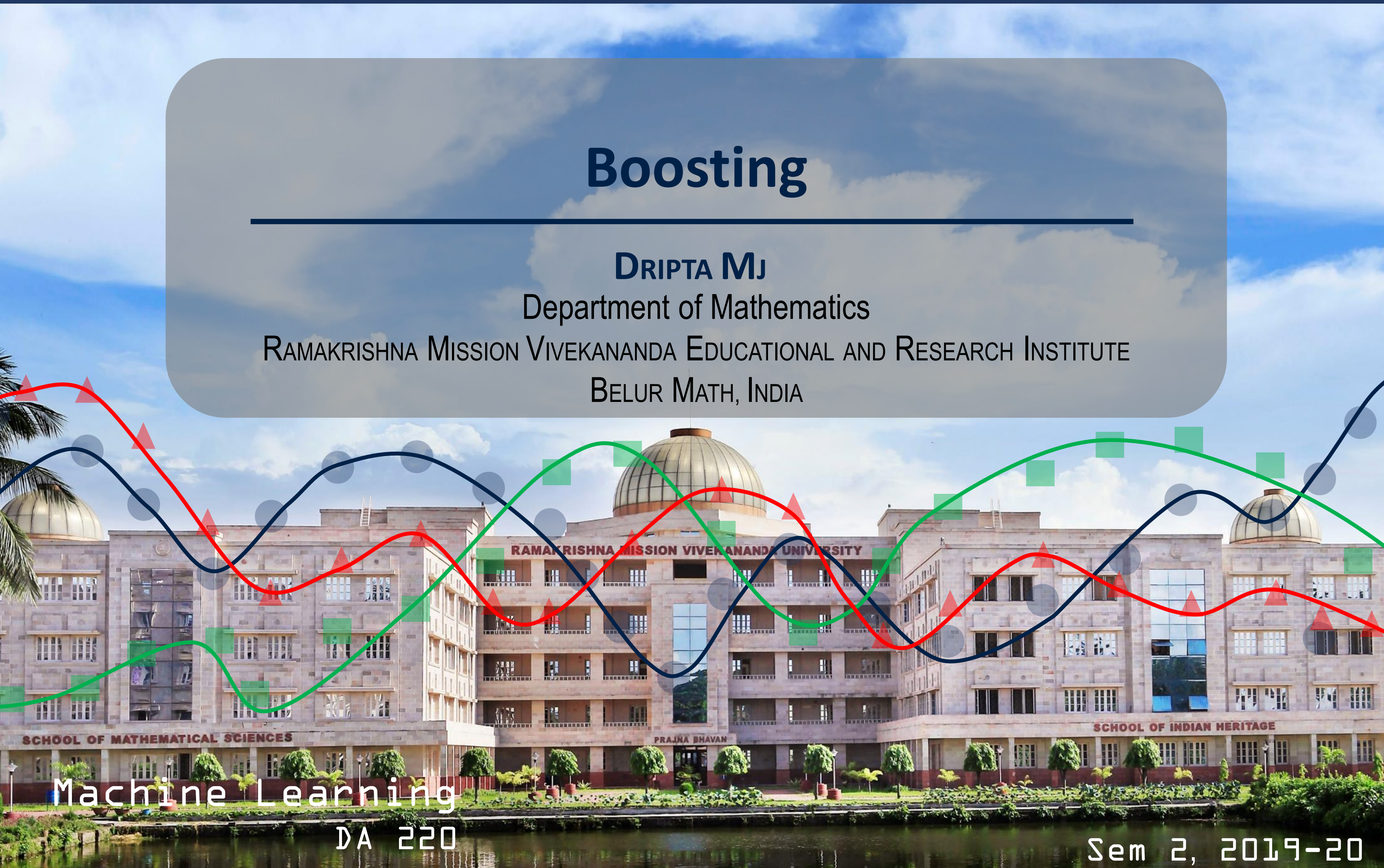
Boosting

Dripta Mj
Department of Mathematics
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, India

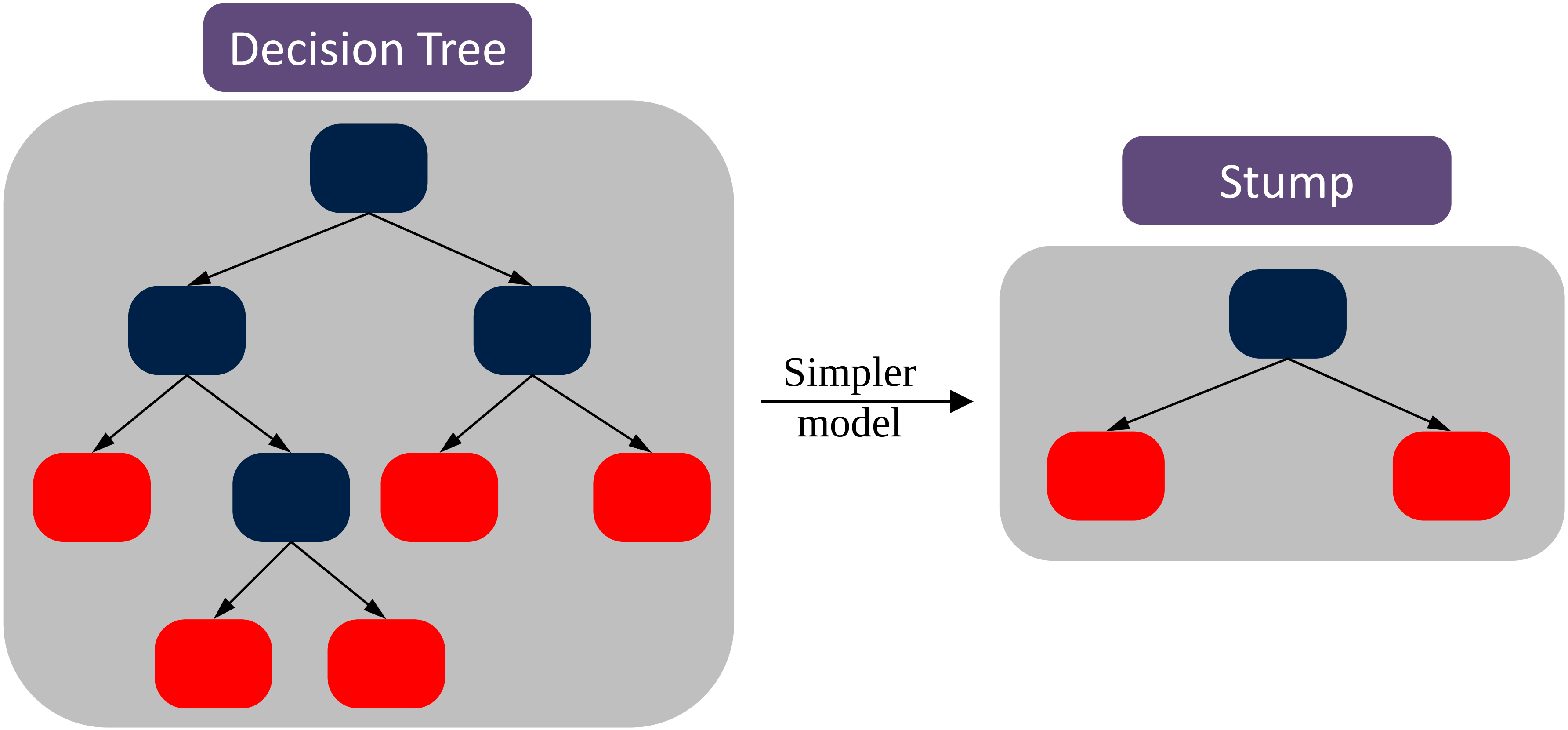


Machine Learning
DA 220

Sem 2, 2019-20

Weak learner

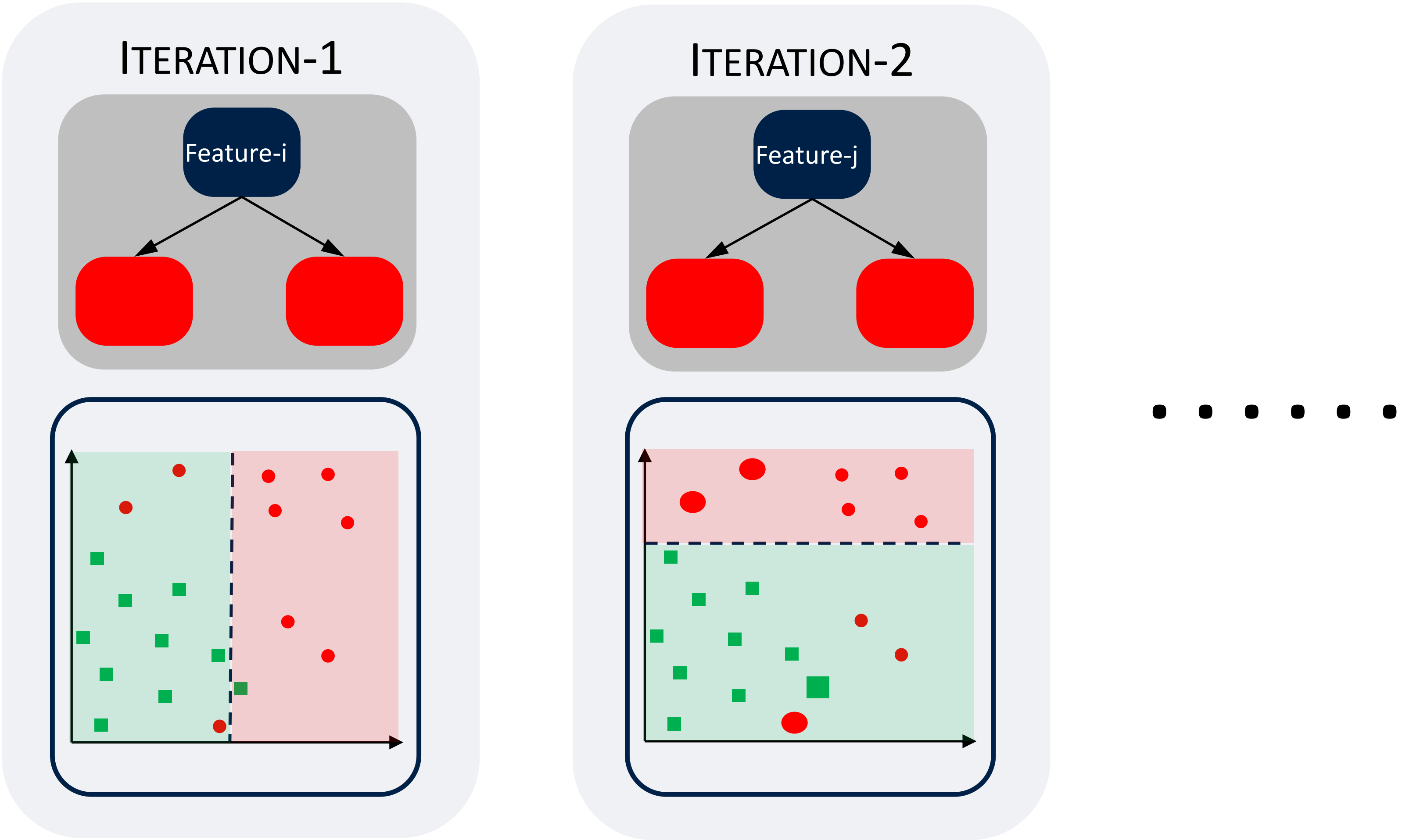
- Transforms a weak learning algorithm into a strong one.



  – A weak learner performs just better than random guessing.

Additive ensemble

- At each stage, a weak learner is introduced to compensate for the shortcomings of existing weak learners.

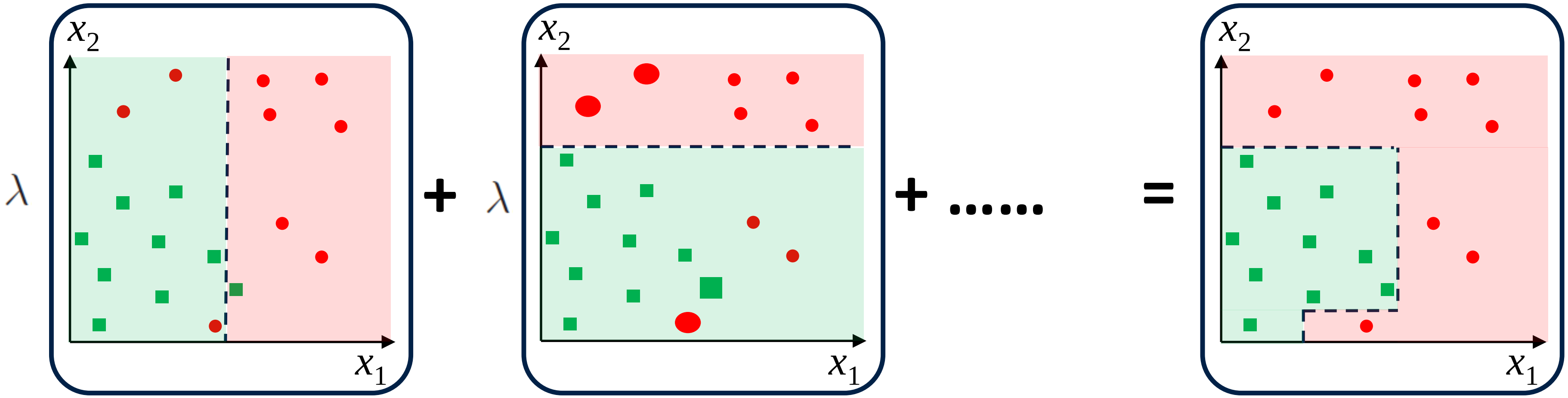


Additive ensemble

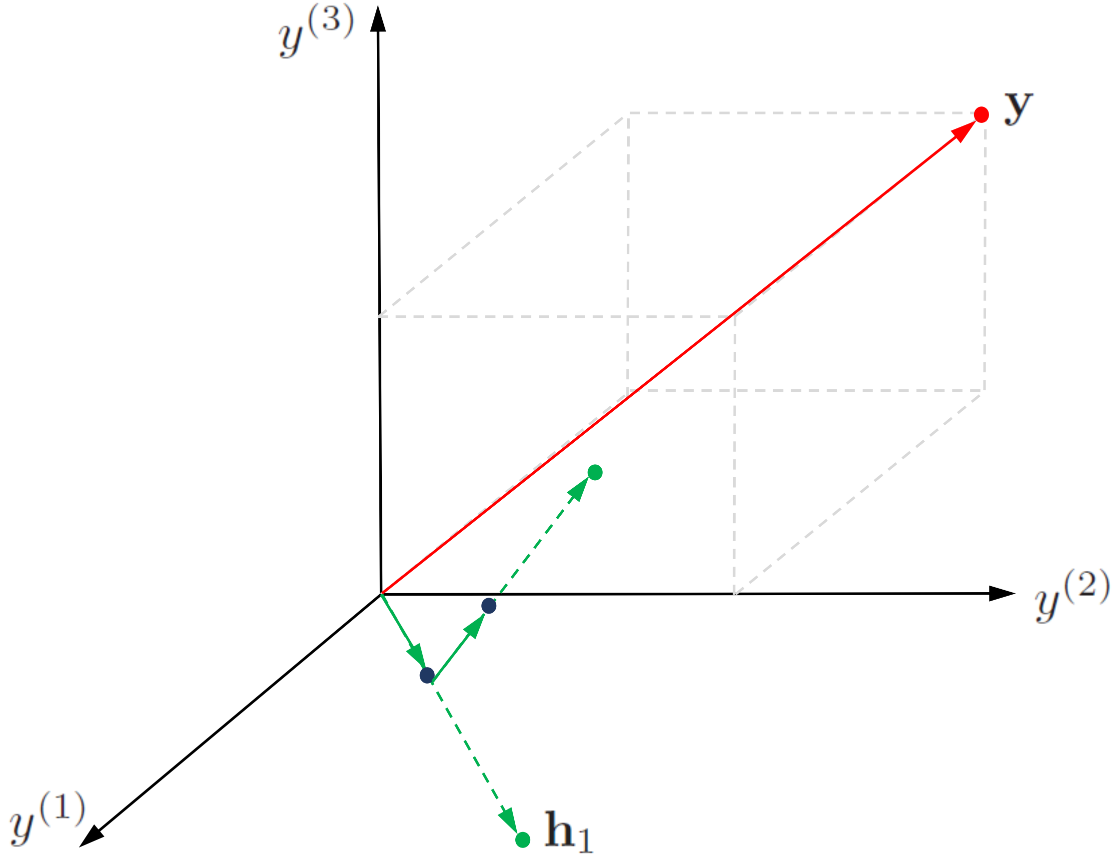
- Define an additive ensemble model at the end of the Tth iteration as

$$H_T(\mathbf{x}) = \sum_{t=1}^{T} \lambda h_t(\mathbf{x})$$

where λ is the step-size and h_t is the classifier that is added to the ensemble at the tth iteration.



Intuition



- Outputs: $\mathbf{y} = \left[y^{(1)}, y^{(2)}, y^{(3)}\right]^{\mathrm{T}}$
- Predictions of 1st weak learner: $\mathbf{h}_1 = \left[h_1(\mathbf{x}^{(1)}), h_1(\mathbf{x}^{(2)}), h_1(\mathbf{x}^{(3)})\right]^{\mathrm{T}}$

Loss function

- Let $\widehat{\mathcal{L}}(H_T(\mathbf{x}^{(n)}), y^{(n)})$ be a convex, differentiable loss function where $H_T(\mathbf{x}^{(n)})$ is the ensemble prediction and $y^{(n)}$ is the observed output for a input $\mathbf{x}^{(n)}$.

- The overall loss can then written as

$$\mathcal{L}(H_T) = \frac{1}{N}\sum_{n=1}^{N} \widehat{\mathcal{L}}(H_T(\mathbf{x}^{(n)}), y^{(n)})$$

- At the $(t+1)$th iteration, a new weak learner is added to the ensemble such that

$$h_{t+1} = \arg\min_{h \in \mathcal{H}} \mathcal{L}(H_t + \lambda h)$$

where λ is the step size.

- Employing Taylor approximation on $\mathcal{L}(H_t + \lambda h)$ gives

$$\begin{aligned}\mathcal{L}(H_t + \lambda h) &\approx \mathcal{L}(H_t) + \lambda < \nabla\mathcal{L}(H_t), \mathbf{h} > \\ &\approx \mathcal{L}(H_t) + \lambda \sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial(H_t(\mathbf{x}^{(n)}))} h(\mathbf{x}^{(n)})\end{aligned}$$

Loss function

- Therefore

$$h_{t+1} = \arg\min_{h \in \mathcal{H}} \mathcal{L}(H_t + \lambda h)$$
$$= \arg\min_{h \in \mathcal{H}} \sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial(H_t(\mathbf{x}^{(n)}))} h(\mathbf{x}^{(n)})$$
$$= \arg\min_{h \in \mathcal{H}} \sum_{n=1}^{N} r_{t,n} h(\mathbf{x}^{(n)})$$

where $r_{t,n} = \dfrac{\partial \mathcal{L}}{\partial(H_t(\mathbf{x}^{(n)}))} h(\mathbf{x}^{(n)})$

- The loss function $\mathcal{L}$ is reduced as long as $\sum_{n=1}^{N} r_{t,n} h(\mathbf{x}^{(n)}) < 0$.

General boosting algorithm

Intialize $H_0 = \mathbf{0}$

for $t = 0$ to $T-1$ **do**

$$h_{t+1} = \arg\min_{h \in \mathcal{H}} \sum_{n=1}^{N} r_{t,n} h(\mathbf{x}^{(n)})$$

if $\sum_{n=1}^{N} r_{t,n} h(\mathbf{x}^{(n)}) < 0$ **then**

$$H_{t+1} = H_t + \lambda h_{t+1}$$

else

return H_t

end if

end for

return H_T

AdaBoost

- Binary classification problem – $y^{(n)} \in \{-1, 1\}$.
- Weak learners $h \in \mathcal{H}$ also have outputs $h(\mathbf{x}^{(n)}) \in \{-1, 1\}$.
- Loss function:

$$\mathcal{L}(H) = \sum_{n=1}^{N} \exp\left[-y^{(n)} H(\mathbf{x}^{(n)})\right]$$

- Gradient of loss function:

$$r_{t,n} = \frac{\partial \mathcal{L}}{\partial\left[H(\mathbf{x}^{(n)})\right]} = -y^{(n)} \exp\left[-y^{(n)} H(\mathbf{x}^{(n)})\right]$$

- Method enables computation of best step-size λ.

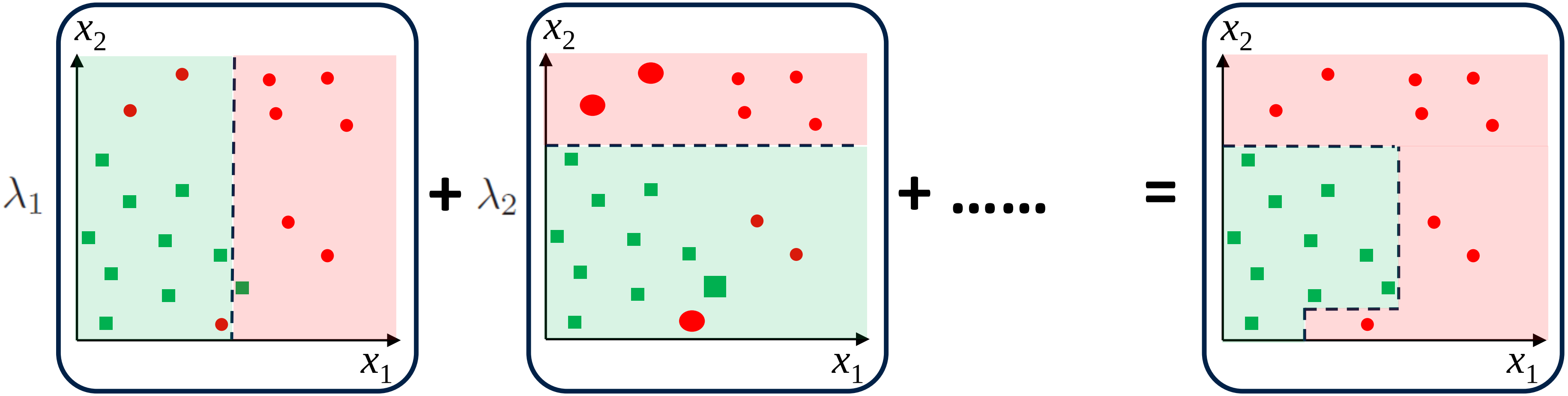
Weights

- Define $w^{(n)}$ as

$$w^{(n)} = \frac{\exp\left[-y^{(n)}H(\mathbf{x}^{(n)})\right]}{\sum_{j=1}^{N}\exp\left[-y^{(j)}H(\mathbf{x}^{(j)})\right]}$$

  - The denominator acts as a normalizing factor to give $\sum_{n=1}^{N} w^{(n)} = 1$.

- Each weight $w^{(n)}$ is the relative contribution of the data point $(\mathbf{x}^{(n)}, y^{(n)})$ to the overall loss.



Optimal weak-learner

- Optimal "weak-learner" at the $(t+1)$th iteration:

$$h_{t+1} = \arg\min_{h \in \mathcal{H}} \sum_{n=1}^{N} r_{t,n} h(\mathbf{x}^{(n)})$$

$$= \arg\min_{h \in \mathcal{H}} - \sum_{n=1}^{N} y^{(n)} \exp\left[-y^{(n)} H(\mathbf{x}^{(n)})\right] h(\mathbf{x}^{(n)})$$

$$\text{as} \quad r_{t,n} = -y^{(n)} \exp\left[-y^{(n)} H(\mathbf{x}^{(n)})\right]$$

$$= \arg\min_{h \in \mathcal{H}} - \sum_{n=1}^{N} y^{(n)} \left(w^{(n)} Z\right) h(\mathbf{x}^{(n)}) \qquad \text{as} \quad w^{(n)} = \frac{\exp\left[-y^{(n)} H(\mathbf{x}^{(n)})\right]}{Z}$$

Now

$$y^{(n)} h(\mathbf{x}^{(n)}) = 1 \qquad \text{if} \quad h(\mathbf{x}^{(n)}) = y^{(n)}$$

$$y^{(n)} h(\mathbf{x}^{(n)}) = -1 \qquad \text{if} \quad h(\mathbf{x}^{(n)}) = -y^{(n)}$$

Optimal weak-learner

therefore

$$h_{t+1} = \arg\min_{h\in\mathcal{H}} -\sum_{n:h(\mathbf{x}^{(n)})=y^{(n)}} w^{(n)} + \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)}$$

$$= \arg\min_{h\in\mathcal{H}} -\left(1 - \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)}\right) + \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)}$$

as $\sum_{n:h(\mathbf{x}^{(n)})=y^{(n)}} w^{(n)} + \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)} = 1$

$$= \arg\min_{h\in\mathcal{H}} -1 + 2\sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)}$$

$$= \arg\min_{h\in\mathcal{H}} 2\sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)}$$

$$= \arg\min_{h\in\mathcal{H}} \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)}$$

$\boxed{h_{t+1} = \arg\min_{h\in\mathcal{H}} \epsilon}$ where $\epsilon = \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)}$ is the weighted classification error

Optimal step-size

- Determining the optimal step-size λ at the $(t+1)$th iteration:

$$\lambda_{t+1} = \arg\min_{\lambda} \mathcal{L}\big(H_t + \lambda h_{t+1}\big)$$
$$= \arg\min_{\lambda} \sum_{n=1}^{N} \exp\Big[-y^{(n)}\big(H_t + \lambda h_{t+1}\big)\Big]$$

- Differentiating the objective function w.r.t. λ and equating to 0:

$$\frac{\partial \sum_{n=1}^{N} \exp\Big[-y^{(n)}\big(H_t + \lambda h_{t+1}\big)\Big]}{\partial \lambda} = 0$$

$$-\sum_{n:h(\mathbf{x}^{(n)})=y^{(n)}} \exp\Big[-y^{(n)}H_t - \lambda y^{(n)} h_{t+1}\Big] + \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} \exp\Big[-y^{(n)}H_t - \lambda\Big] = 0$$

$$-\sum_{n:h(\mathbf{x}^{(n)})=y^{(n)}} \exp\Big[-y^{(n)}H_t\Big]\exp\Big[-\lambda\Big] + \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} \exp\Big[-y^{(n)}H_t\Big]\exp\Big[\lambda\Big] = 0$$

$$-\sum_{n:h(\mathbf{x}^{(n)})=y^{(n)}} w^{(n)} Z \exp\Big[-\lambda\Big] + \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)} Z \exp\Big[\lambda\Big] = 0$$

as $\quad w^{(n)} = \exp[-y^{(n)} H(\mathbf{x}^{(n)})]/Z$

Optimal step-size

$$-\sum_{n:h(\mathbf{x}^{(n)})=y^{(n)}} w^{(n)} \exp\Big[-\lambda\Big] + \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)} \exp\Big[\lambda\Big] = 0$$

$$-\exp\left[-\lambda\right]\Big(1 - \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)}\Big) + \exp\left[\lambda\right] \sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)} = 0$$

$$-\exp\Big[-\lambda\Big](1-\epsilon) + \exp\Big[\lambda\Big]\epsilon = 0$$

$$\exp\Big[\lambda\Big]\epsilon = \exp\Big[-\lambda\Big](1-\epsilon)$$

$$\exp\Big[2\lambda\Big] = \frac{1-\epsilon}{\epsilon}$$

$$\boxed{\lambda = \frac{1}{2}\log\left(\frac{1-\epsilon}{\epsilon}\right)}$$

- The optimal step-size leads to fast convergence of the AdaBoost algorithm.

AdaBoost training

- Optimal "weak-learner" at the $(t+1)$th iteration:

$$h_{t+1} = \arg\min_{h\in\mathcal{H}} \sum_{n=1}^{N} r_{t,n} h(\mathbf{x}^{(n)})$$
$$= \arg\min_{h\in\mathcal{H}} \underbrace{\sum_{n:h(\mathbf{x}^{(n)})\neq y^{(n)}} w^{(n)}}_{\text{weighted classification error}}$$

- Optimal step-size λ at the $(t+1)$th iteration:

$$\lambda_{t+1} = \arg\min_{\lambda} \mathcal{L}(H_t + \lambda h_{t+1})$$

- Differentiating the objective function w.r.t. λ and equating to 0:

$$\frac{\partial \sum_{n=1}^{N} \exp\left[-y^{(n)}(H_t + \lambda h_{t+1})\right]}{\partial \lambda} = 0 \qquad \Rightarrow \lambda_{t+1} = \frac{1}{2}\log\left(\frac{1-\epsilon}{\epsilon}\right)$$

- The optimal step-size leads to fast convergence of the AdaBoost algorithm.

AdaBoost algorithm

Initialize $H_0 = \mathbf{0}$ and $w^{(n)} = 1/N$, $n = 1, 2, \ldots N$

for $t = 0$ to $T - 1$ **do**

$$h_{t+1} = \arg\min_{h \in \mathcal{H}} \sum_{n: h(\mathbf{x}^{(n)}) \neq y^{(n)}} w^{(n)}$$

$$\epsilon = \sum_{n: h(\mathbf{x}^{(n)}) \neq y^{(n)}} w^{(n)}$$

if $\epsilon < 1/2$ **then**

$$\lambda_{t+1} = \frac{1}{2} \log\left(\frac{1-\epsilon}{\epsilon}\right)$$

$$H_{t+1} = H_t + \lambda_{t+1} h_{t+1}$$

$$w^{(n)} \leftarrow \frac{w^{(n)} \exp(-\lambda_{t+1} h(\mathbf{x}^{(n)}) y^{(n)})}{2\sqrt{\epsilon(1-\epsilon)}} \qquad n = 1, 2, \ldots N$$

else

return H_t

end if

end for

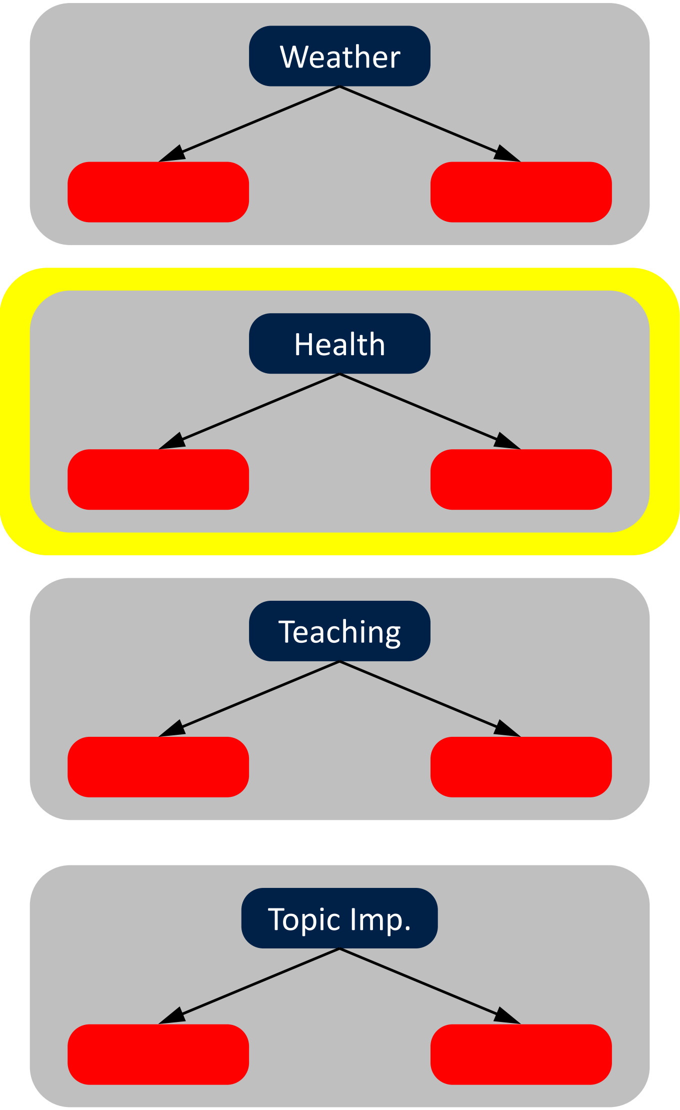
return H_T

AdaBoost (with decision stumps)

Original Dataset

Instance	Weather	Health	Teaching	Topic Importance	Going to class?	$w_1^{(n)}$
1	Hot	Good	Interesting	Medium	Yes	1/8
2	Cold	Average	Boring	High	Yes	1/8
3	Cold	Sick	Mediocre	Medium	No	1/8
4	Mild	Average	Interesting	High	Yes	1/8
5	Rainy	Sick	Mediocre	Low	No	1/8
6	Hot	Good	Boring	High	Yes	1/8
7	Rainy	Good	Mediocre	Medium	No	1/8
8	Mild	Good	Mediocre	Medium	Yes	1/8

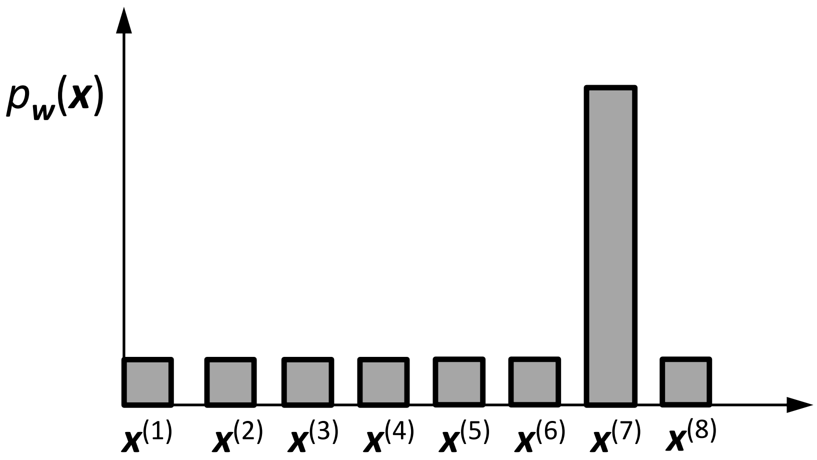
- One example of weak classifier – Decision stumps.
  - A decision stump is a one-level decision tree comprising one root and terminal nodes.
- Can use a separate stump for splitting w.r.t. a particular feature.
- Select the stump giving the least weighted error
  - Suppose the feature Health gives the best result.



AdaBoost (with decision stumps)

Original Dataset

Instance	Weather	Health	Teaching	Topic Importance	Going to class?	$w_1^{(n)}$	$w_2^{(n)}$
1	Hot	Good	Interesting	Medium	Yes	1/8	0.07
2	Cold	Average	Boring	High	Yes	1/8	0.07
3	Cold	Sick	Mediocre	Medium	No	1/8	0.07
4	Mild	Average	Interesting	High	Yes	1/8	0.07
5	Rainy	Sick	Mediocre	Low	No	1/8	0.07
6	Hot	Good	Boring	High	Yes	1/8	0.07
7	Rainy	Good	Mediocre	Medium	No	1/8	0.50
8	Mild	Good	Mediocre	Medium	Yes	1/8	0.07



- Compute λ.
- Update the weights.
- For determining the next weak classifier, algorithms use one of the following two ways:
  - same data with updated weights.
  - samples from the training dataset according to the distribution $p_{\mathbf{w}}(\mathbf{x})$.

Boosting error – example

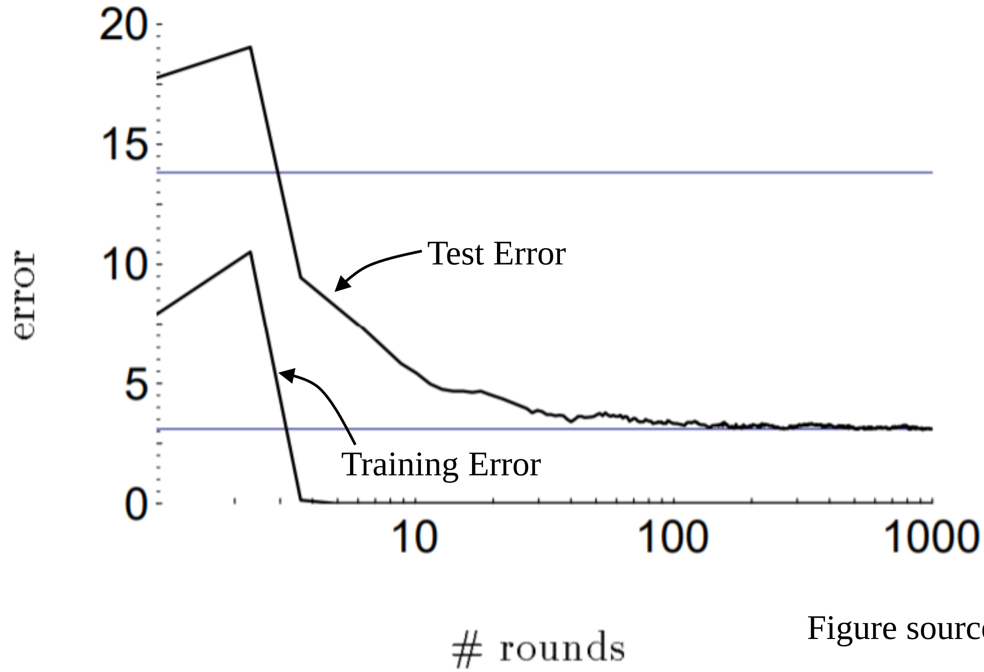


Figure source: Schapire, A Brief Introduction to Boosting, 1999.

- In some cases, boosting is found to be robust to overfitting, i.e. test error decreases even after training error is zero.
- Boosting increases the margin as it focusses on the hardest example.
- Boosting can overfit if
  - margin is small.
  - learners are complex.
  - weak learners perform arbitrarily close to random guessing.

Outlier detection

Optical Character Recognition task

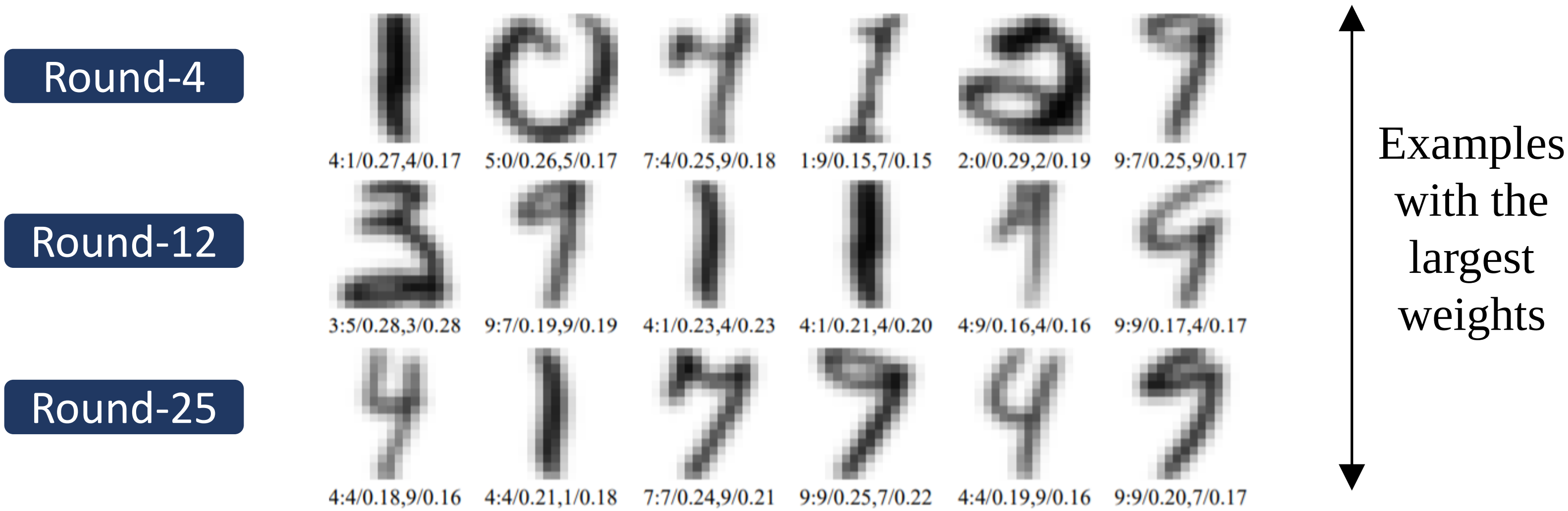


Figure source: Freund and Schapire, Experiments with a New Boosting Algorithm, 1996.

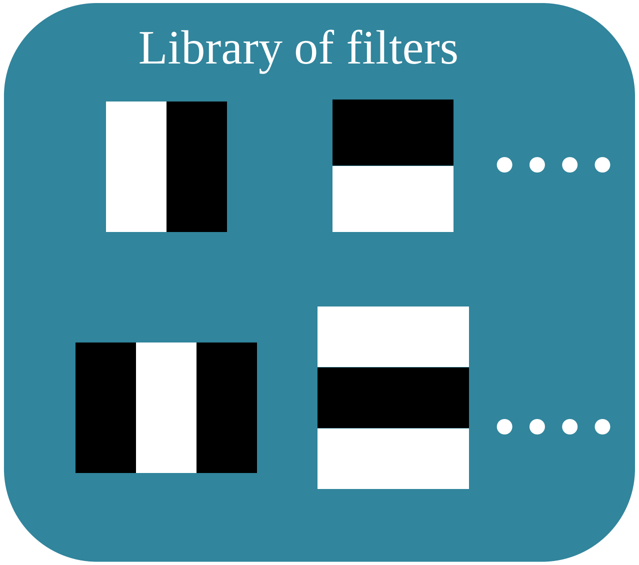
- Adaboost has the ability to identify outliers
- Outliers are those examples which are
  - inherently ambiguous and hard to categorize, or
  - mislabelled
- Adaboost assigns high weights to the hardest examples.
  - These examples often turn to be outliers

Face detection using Adaboost

- Face Detection: Where is the face? Challenge: Real-time detection

Viola-Jones algorithm:

- Feature evaluation
  - Use Haar-like features
  - Slide the filters across the image
  - Value = $\sum$ (Pixels in white region) - $\sum$ (Pixels in black region)
  - If value > threshold, then feature matches



Face detection using Adaboost

- Face Detection: Where is the face?

Viola-Jones algorithm:

- Feature evaluation
- Integral images approach for efficient feature evaluation
- Adaboost for feature selection
  - There are tens of thousands of location and scale combinations of different types of filters.
  - Adaboost is used to select the best features based on weighted error values.
- Cascade of classifiers
  - The cascade quickly discards non-faces and thus gives computational benefits.