Boosting on manifolds: adaptive regularization of base classifiers

Balázs Kégl

(joint work with Ligen Wang)

Université de Montréal

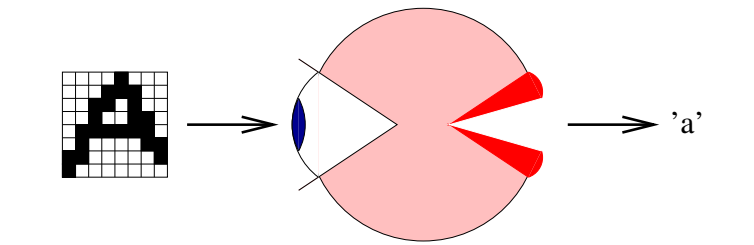
IRIS Machine Learning Workshop

June 9, 2004

Outline

- The supervised learning model
- AdaBoost
- RegBoost
- The graph Laplacian regularizer
- Experimental results

The supervised learning model



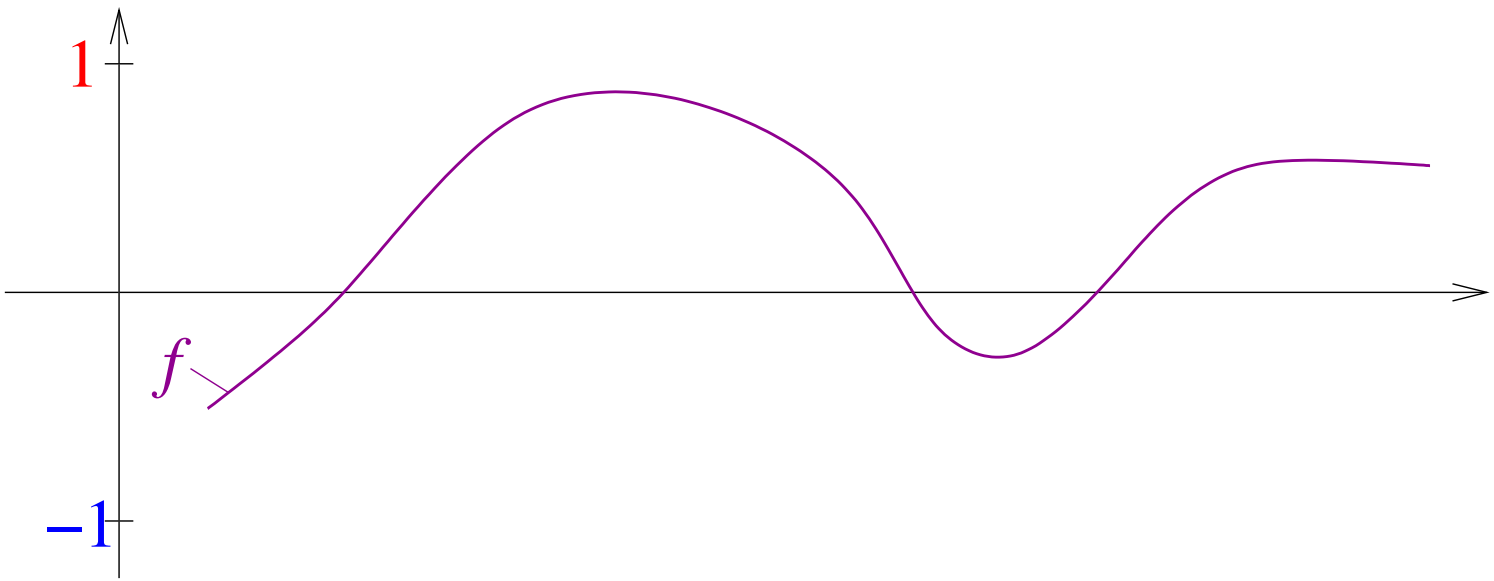
- observation vector: $\mathbf{x} \in \mathbb{R}^d$
- class label: $y \in \{-1, 1\}$ – binary classification
- classifier: $g : \mathbb{R}^d \mapsto \{-1, 1\}$
- discriminant function: $f : \mathbb{R}^d \mapsto [-1, 1]$

$$g(\mathbf{x}) = \begin{cases} 1, & \text{if } f(\mathbf{x}) \geq 0, \\ -1, & \text{if } f(\mathbf{x}) < 0 \end{cases}$$

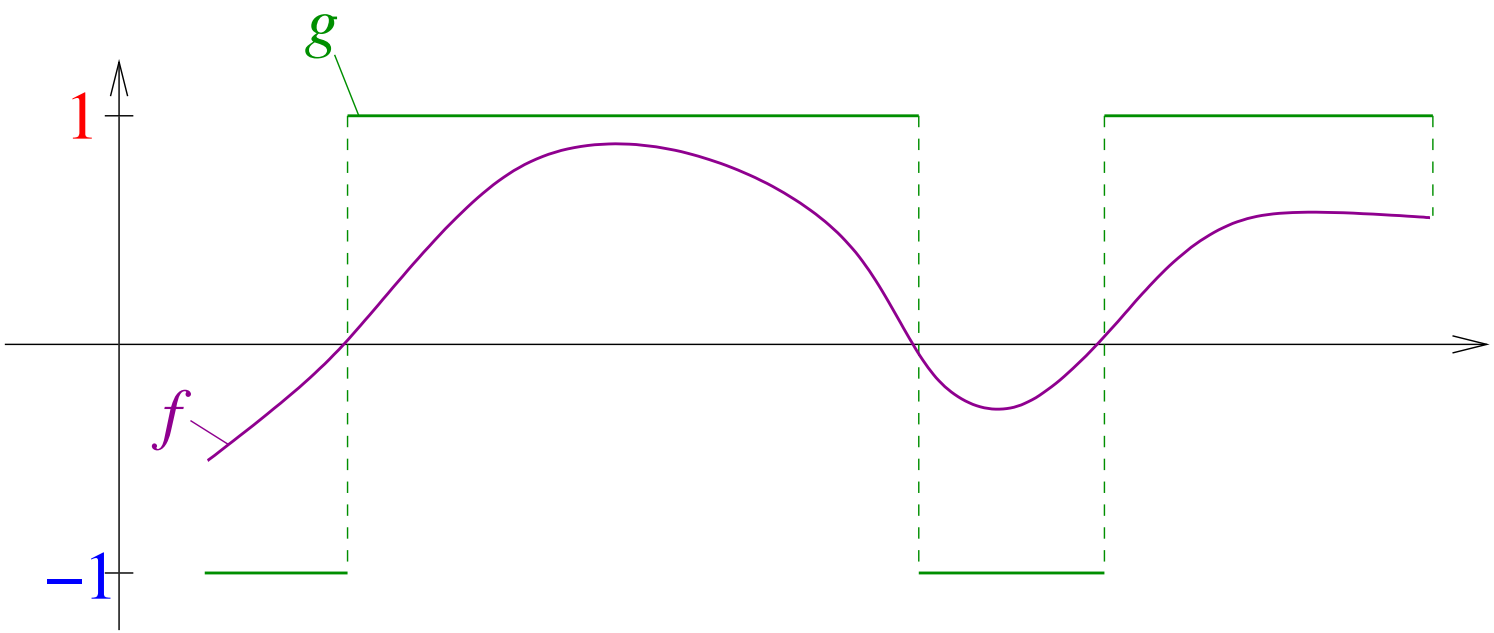
- decision border: $\{\mathbf{x} : f(\mathbf{x}) = 0\}$

The supervised learning model

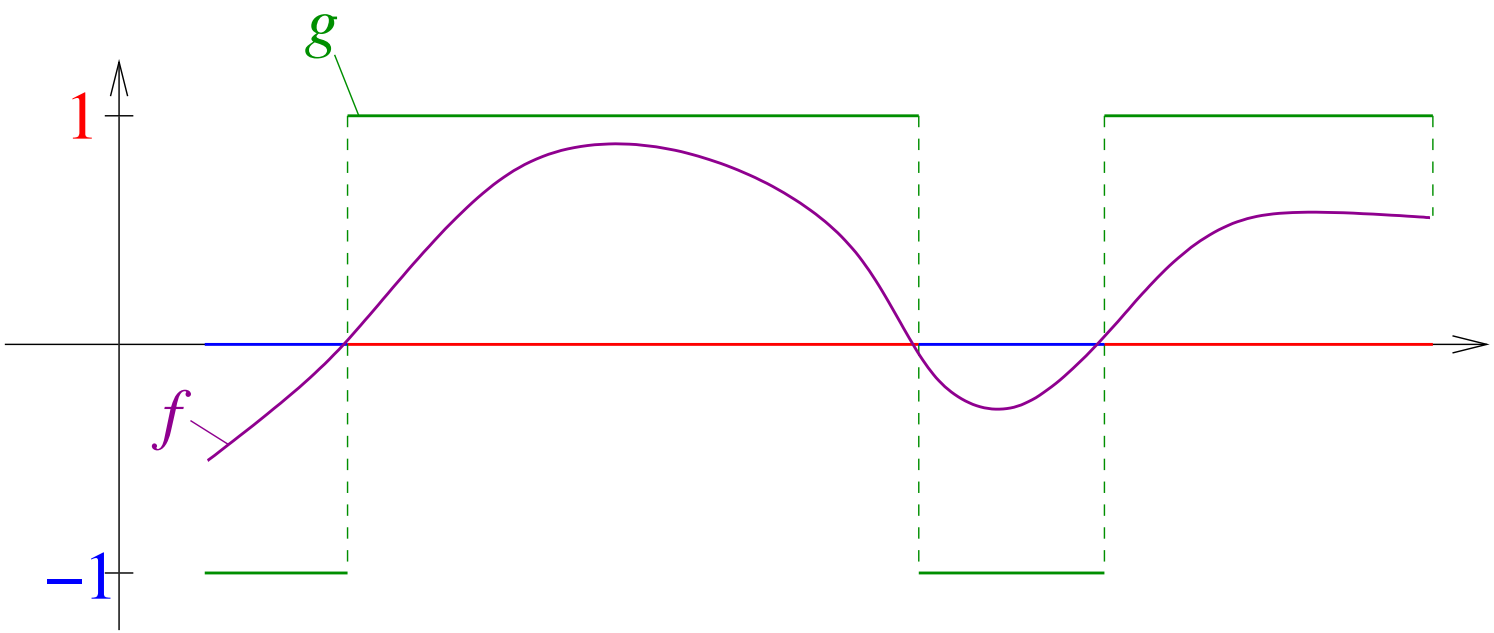
The supervised learning model



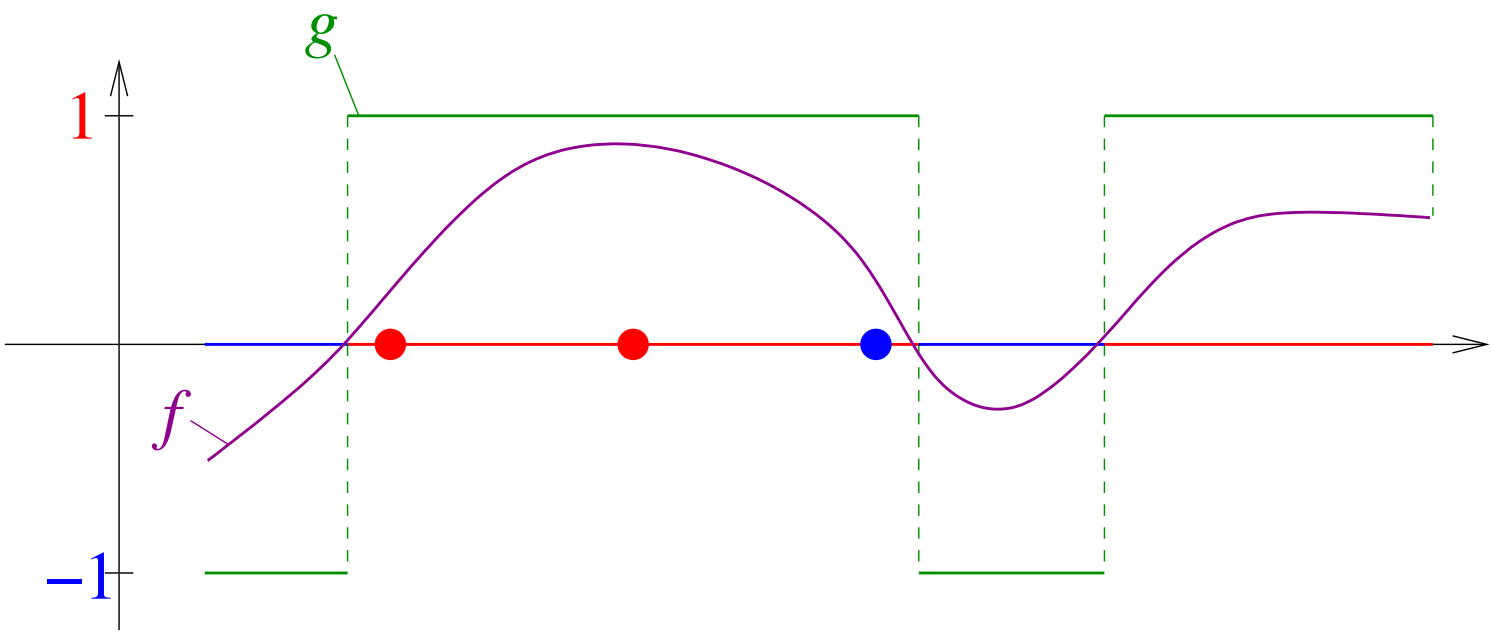
The supervised learning model



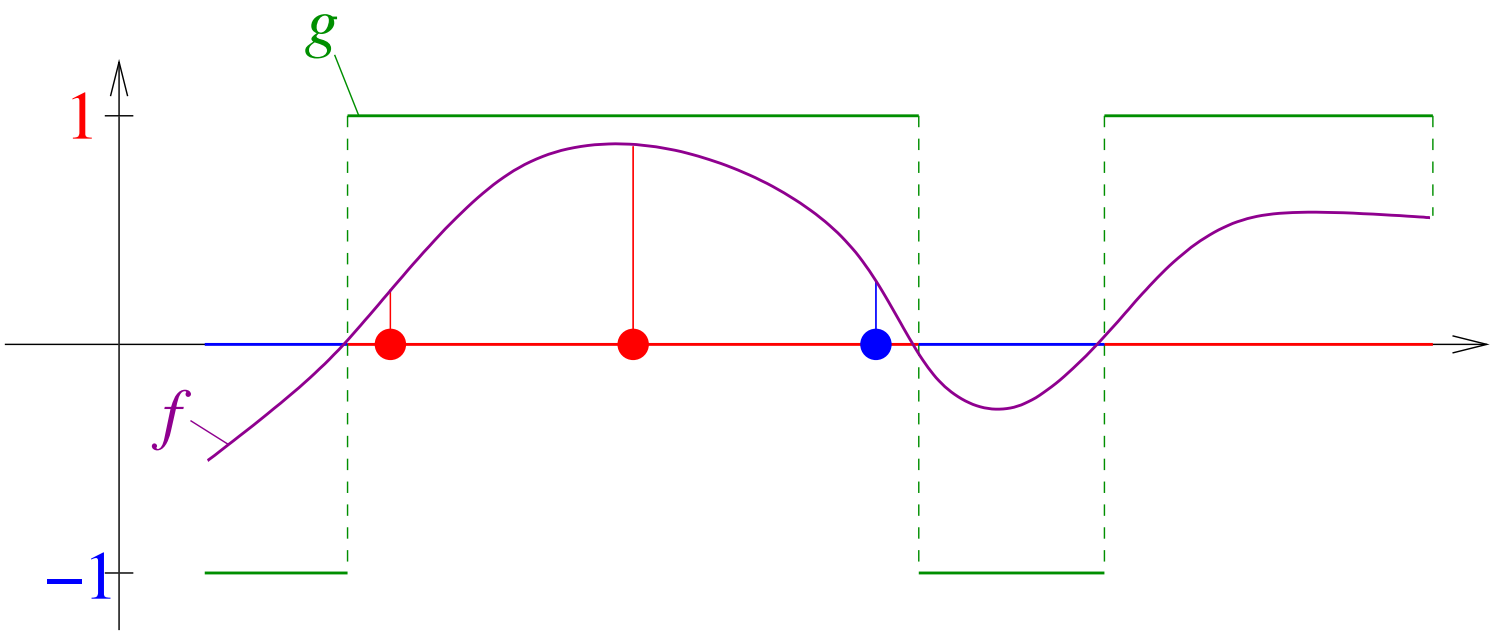
The supervised learning model



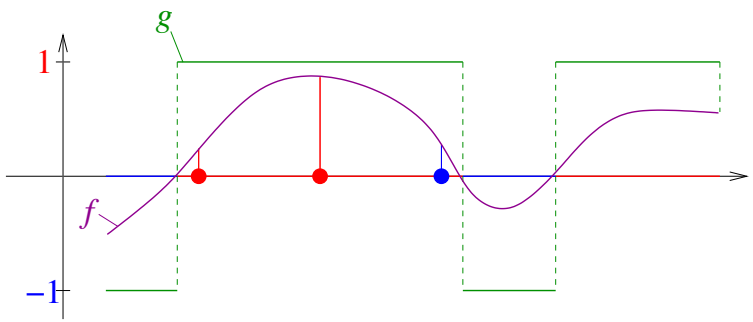
The supervised learning model



The supervised learning model

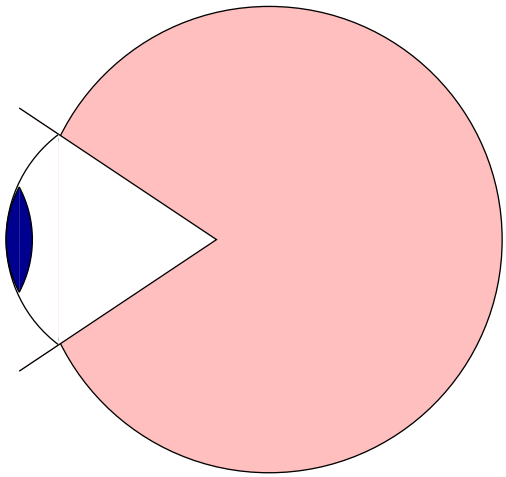


The supervised learning model

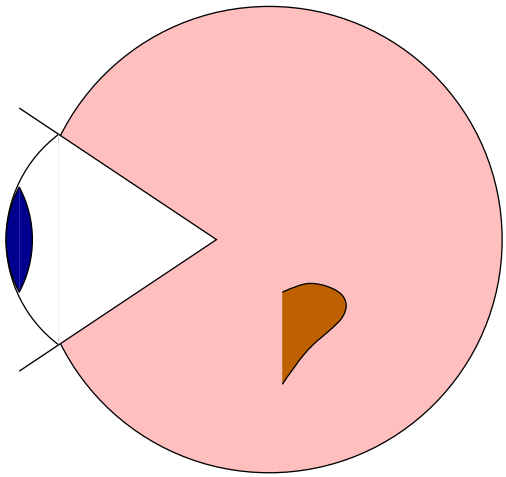


- Margin: $\rho = y \cdot f(\mathbf{x})$
  - classification error $=$ negative margin
  - magnitude of a positive margin quantifies confidence
  - plays an important role in analyzing the generalization performance

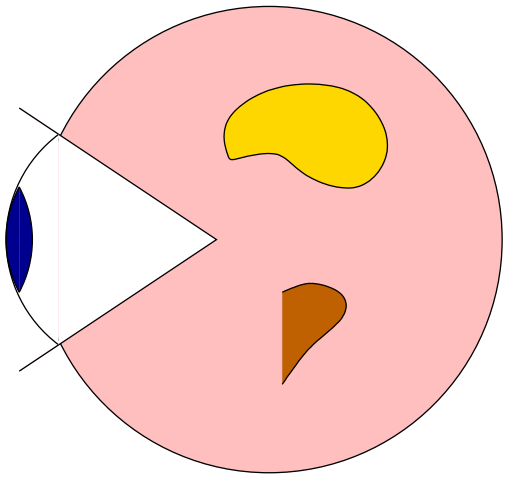
The supervised learning model



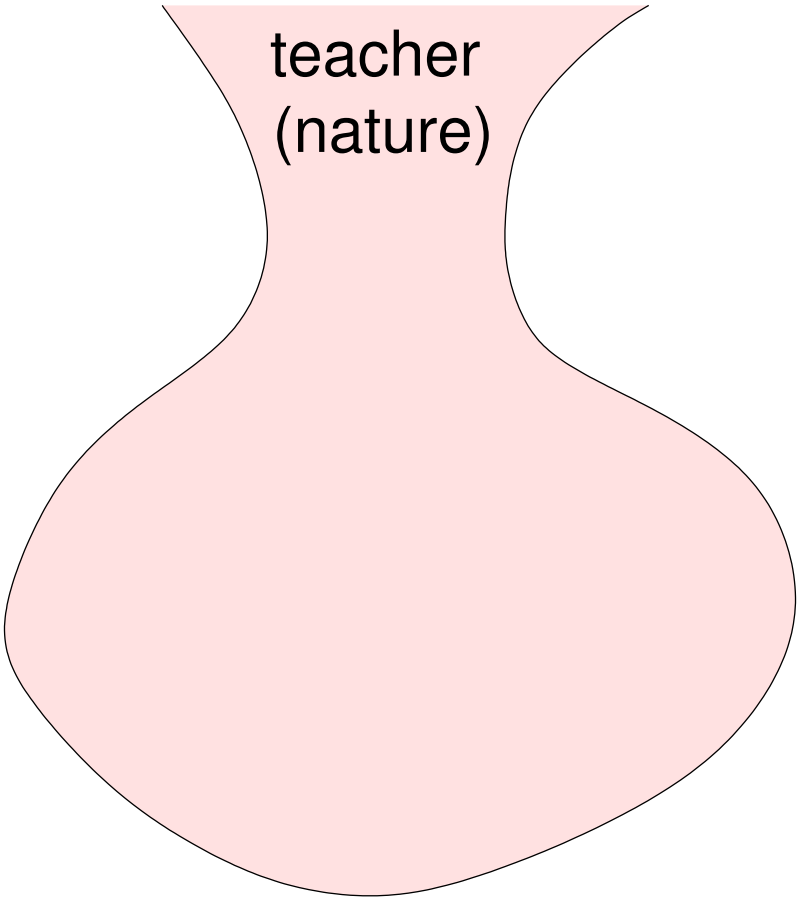
The supervised learning model

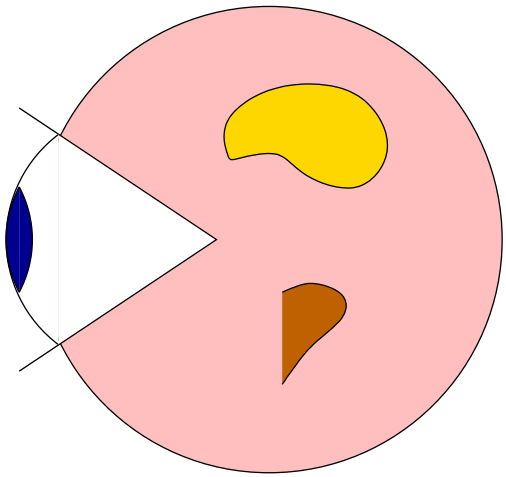


The supervised learning model

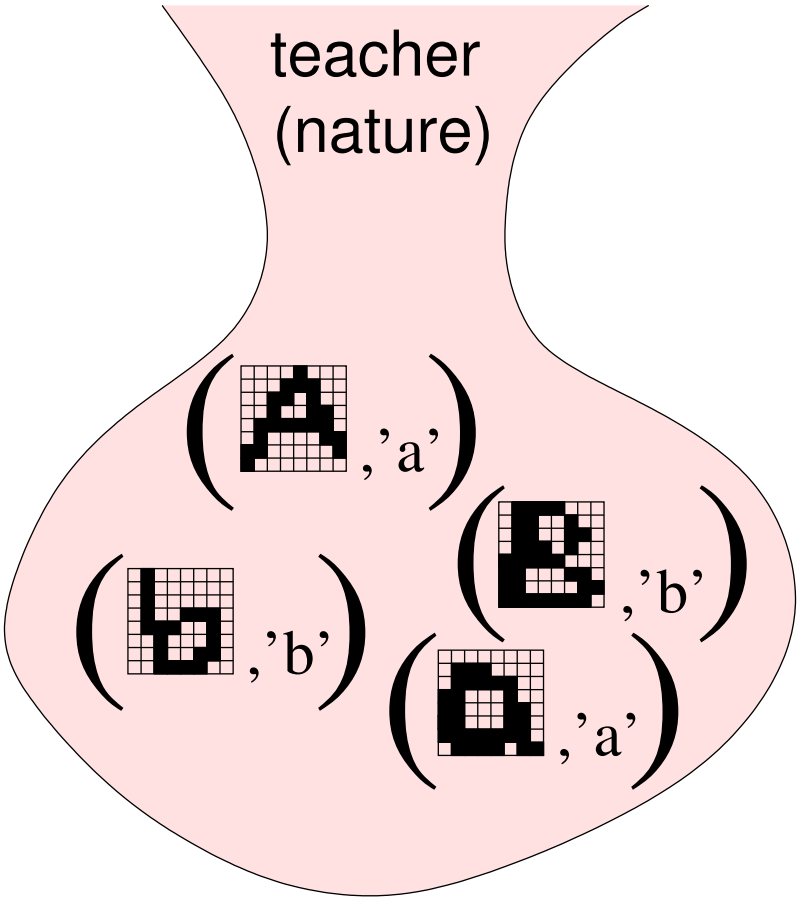


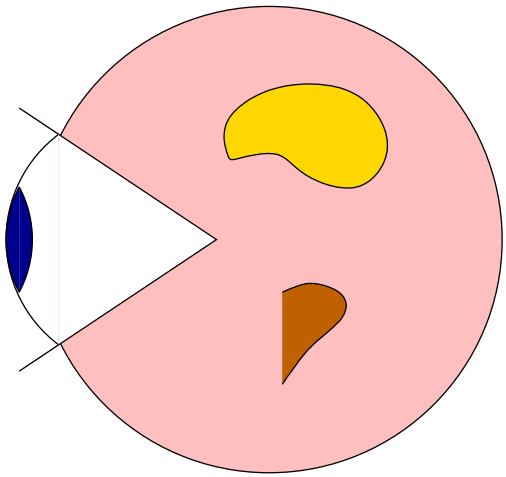
The supervised learning model



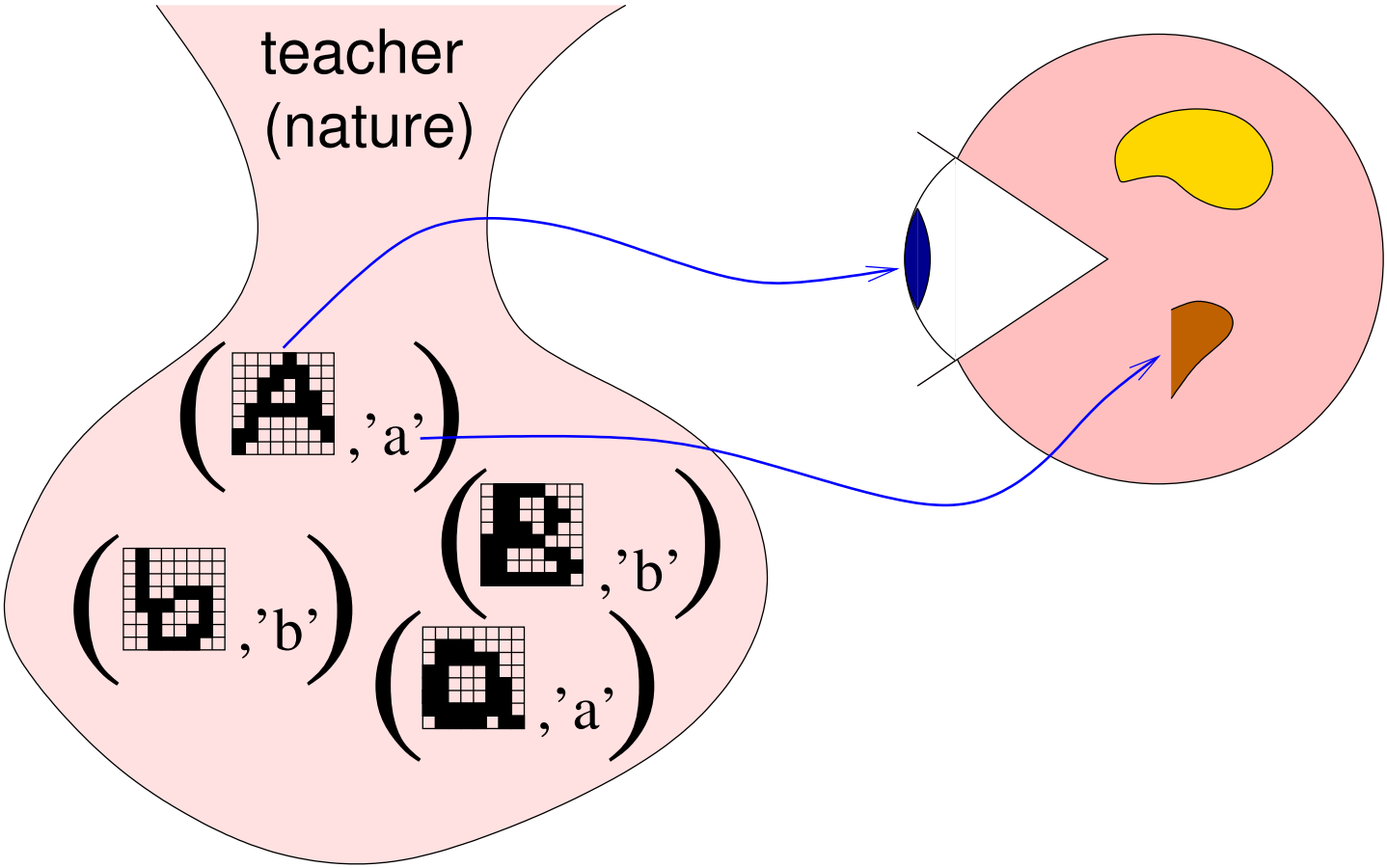


The supervised learning model

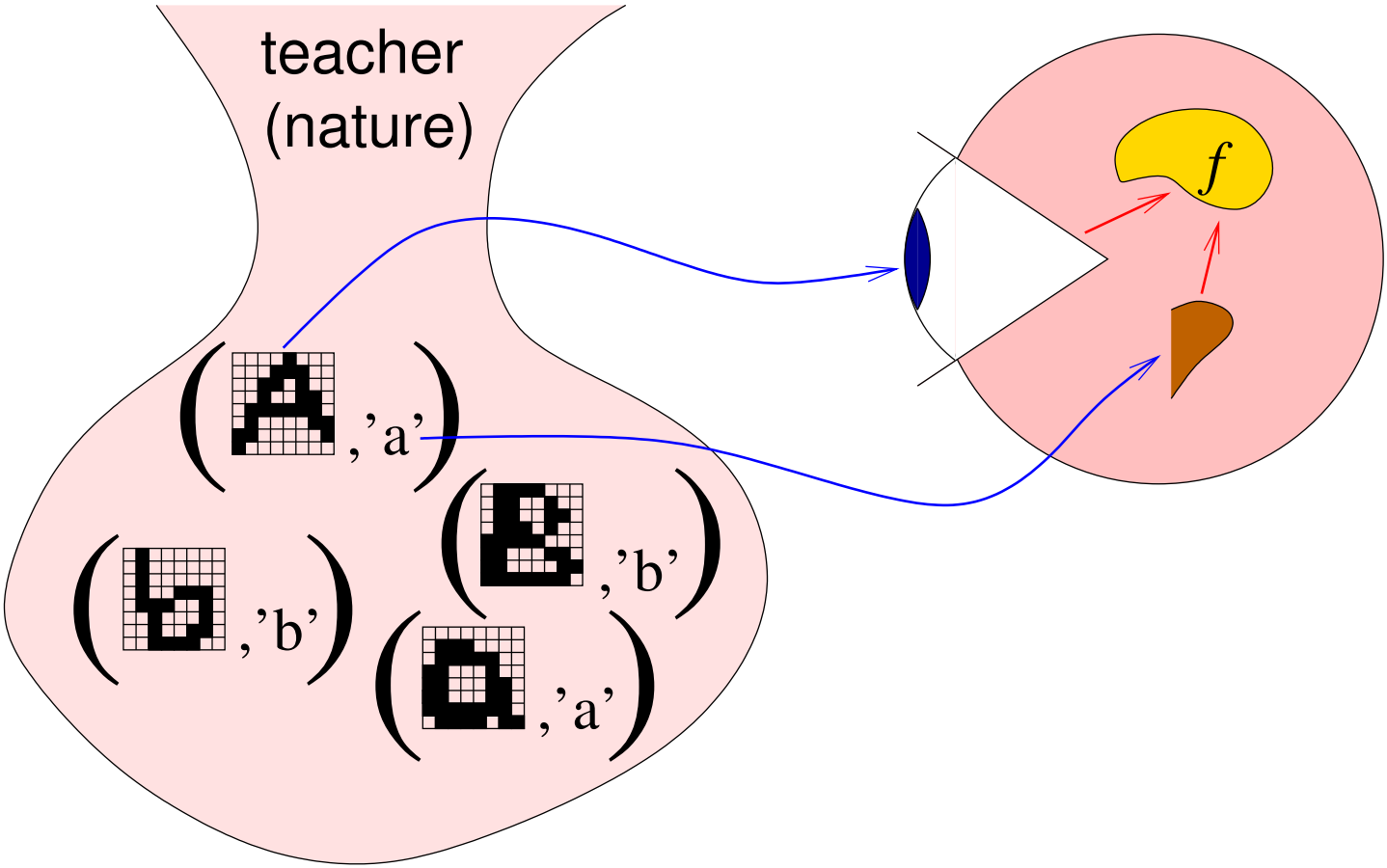




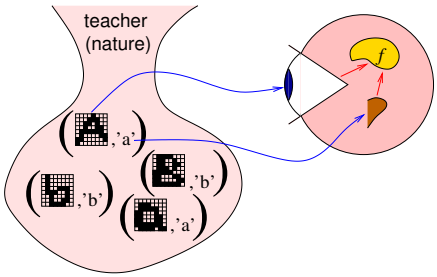
The supervised learning model



The supervised learning model



The supervised learning model



- Learning
  - training sample: $D_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
  - function set: $\mathcal{F}$
  - learning algorithm: $\textsc{Algo} : \left(\mathbb{R}^d \times \{-1, 1\}\right)^n \mapsto \mathcal{F}$

    $\textsc{Algo}(D_n) \rightarrow f$

  - goal: small generalization error $P\left[\textsc{Algo}(D_n, \mathbf{X}) \neq Y\right]$

Outline

- The supervised learning model
- AdaBoost
- RegBoost
- The graph Laplacian regularizer
- Experimental results

AdaBoost

- Model: $f(\mathbf{x}) = \sum_{j=1}^{T} \alpha^{(j)} h^{(j)}(\mathbf{x})$
  - $h^{(j)}$ can be any "weak" classifier: it has to be only slightly better than a random guess
  - interpretation: weighted vote of "weak" experts
  - [Freund - Schapire '97]

AdaBoost

- Algorithm
  - simple, no gradient descent, no quadratic optimization
  - add base functions one at a time
  - set their weights proportionally to their correctness
  - train the next expert on data points that were missed by previous experts
  - automatically concentrates on training points that are hard to learn

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST($D_n = \big((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\big)$, BASE$(D_n, \mathbf{w})$, T)

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $\quad h^{(t)} \leftarrow$ BASE$(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\quad \varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\quad \alpha^{(t)} \leftarrow \frac{1}{2} \log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 $\quad$ **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 $\quad\quad$ **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 $\quad$ **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 $\quad\quad$ **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 $\quad\quad$ **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

$\textsc{AdaBoost}(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \textsc{Base}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \textsc{Base}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2} \log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

AdaBoost$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{Base}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{Base}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2\varepsilon}$ ↑

10 **else** $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{2(1-\varepsilon)}$ ↓

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\{h^{(t)}(\mathbf{x}_i) \neq y_i\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2} \log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 $$w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{\exp\left(-\alpha^{(t)} h^{(t)}(\mathbf{x}_i) y_i\right)}{\sum_{j=1}^{t} w_j^{(t)} \exp\left(-\alpha^{(t)} h^{(t)}(\mathbf{x}_j) y_j\right)}$$

10 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

ADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\varepsilon \leftarrow \sum_{i=1}^{n} w_i I\left\{h^{(t)}(\mathbf{x}_i) \neq y_i\right\}$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\varepsilon \geq 1/2$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 $w_i^{(t+1)} \leftarrow w_i^{(t)} \dfrac{\exp\left(-\alpha^{(t)} h^{(t)}(\mathbf{x}_i) y_i\right)}{\sum_{j=1}^{t} w_j^{(t)} \exp\left(-\alpha^{(t)} h^{(t)}(\mathbf{x}_j) y_j\right)}$

10 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

$\textsc{AdaBoost}(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \textsc{Base}(D_n, \mathbf{w}), T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \textsc{Base}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\gamma \leftarrow \sum_{i=1}^{n} w_i h^{(t)}(\mathbf{x}_i) y_i = 1 - 2\varepsilon$ ▷ *edge*

5 $\alpha^{(t)} \leftarrow \frac{1}{2} \log\left(\frac{1+\gamma}{1-\gamma}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\gamma \geq 0$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{\exp\left(-\alpha^{(t)} h^{(t)}(\mathbf{x}_i) y_i\right)}{\sum_{j=1}^{t} w_j^{(t)} \exp\left(-\alpha^{(t)} h^{(t)}(\mathbf{x}_j) y_j\right)}$

10 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

- Analysis
  - margin: $\rho_i = f(\mathbf{x}_i) y_i$
  - normalized margin: $\widetilde{\rho}_i = \dfrac{\rho_i}{\|\alpha\|_1} = \dfrac{\sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\mathbf{x}_i) y_i}{\sum_{t=1}^{T} \alpha^{(t)}}$
  - AdaBoost directly minimizes the exponential risk

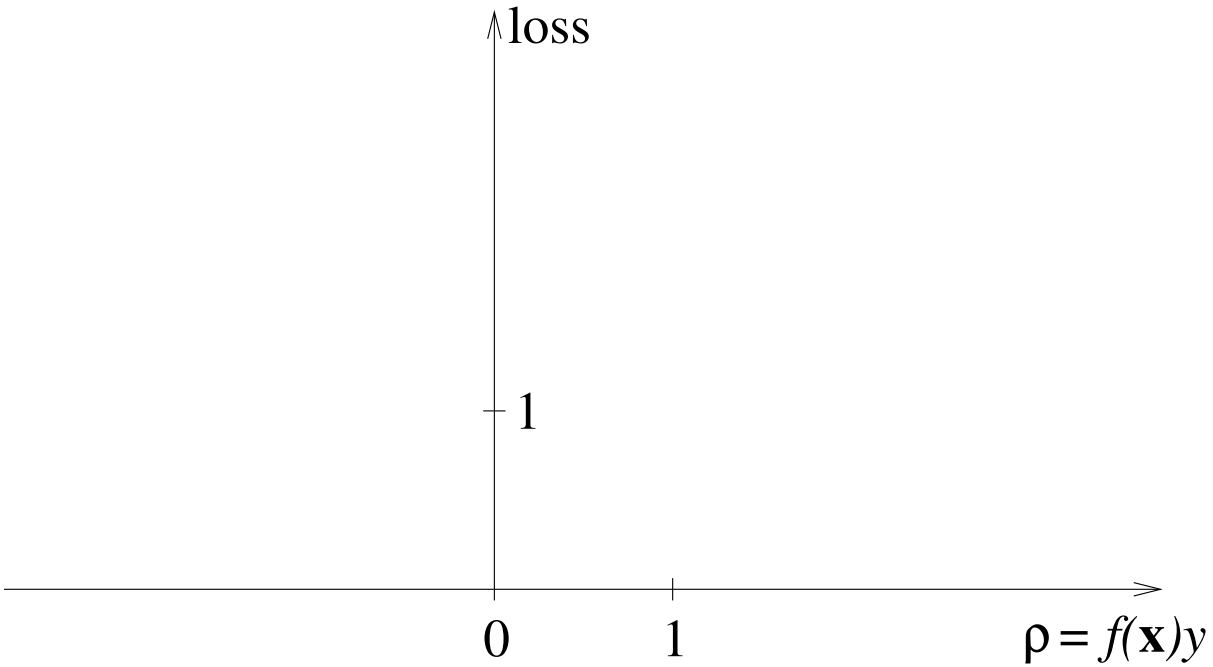
$$\widehat{R}^{(e)}(f) = \frac{1}{n} \sum_{i=1}^{n} e^{-\rho_i}$$

  - AdaBoost indirectly minimizes the training error

$$\widehat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} I\{\rho_i < 0\}$$

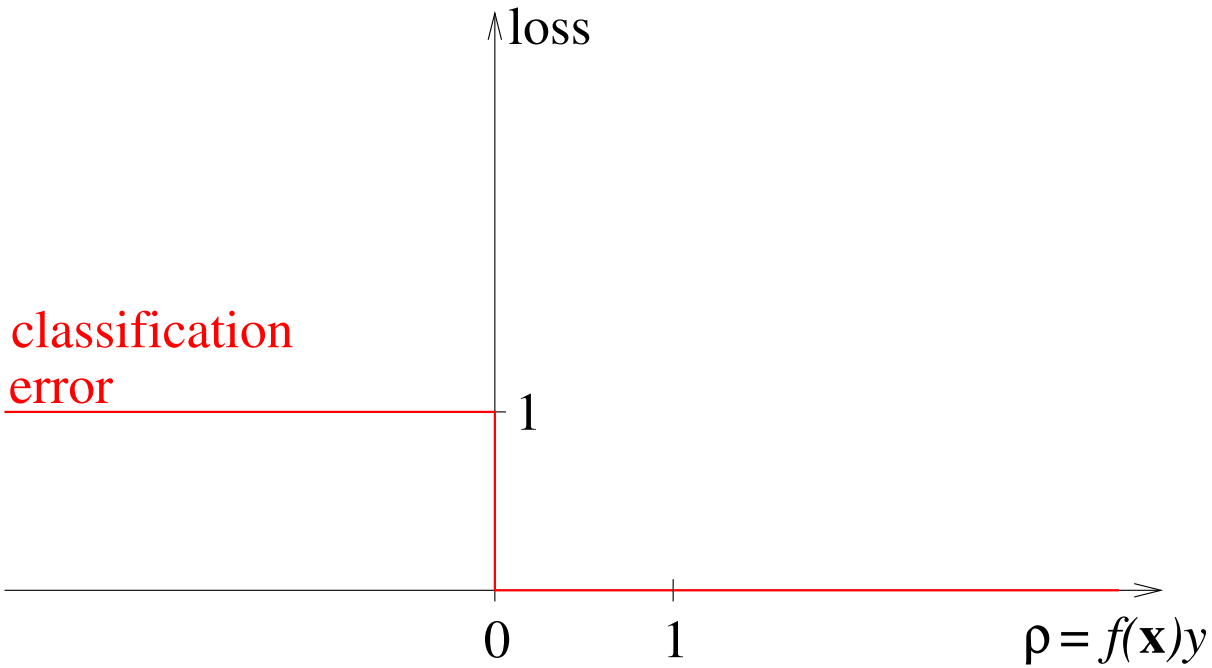
AdaBoost

- Loss functions



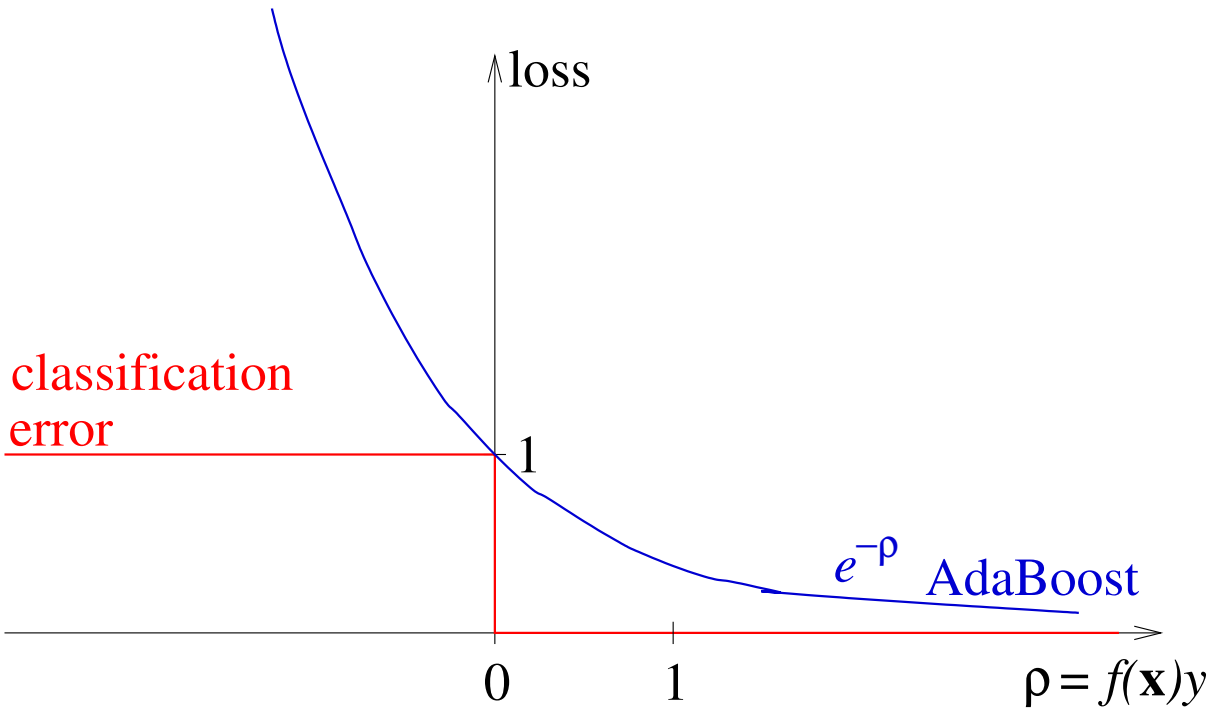
AdaBoost

- Loss functions



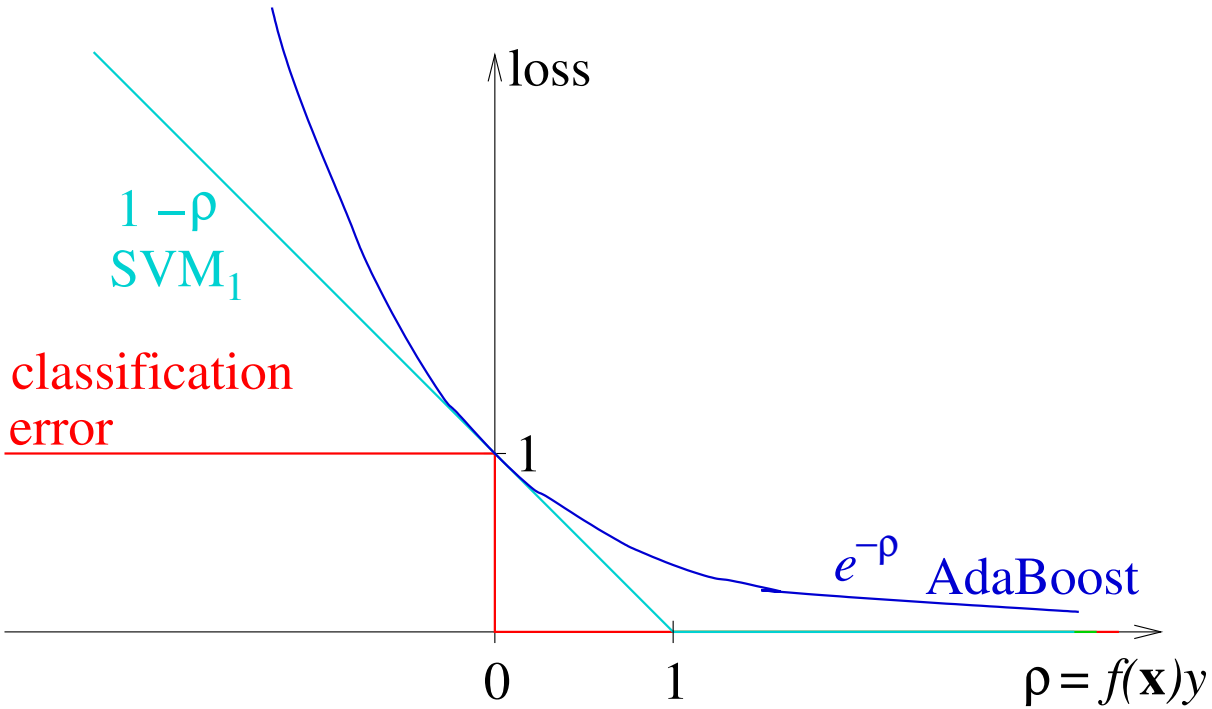
AdaBoost

- Loss functions



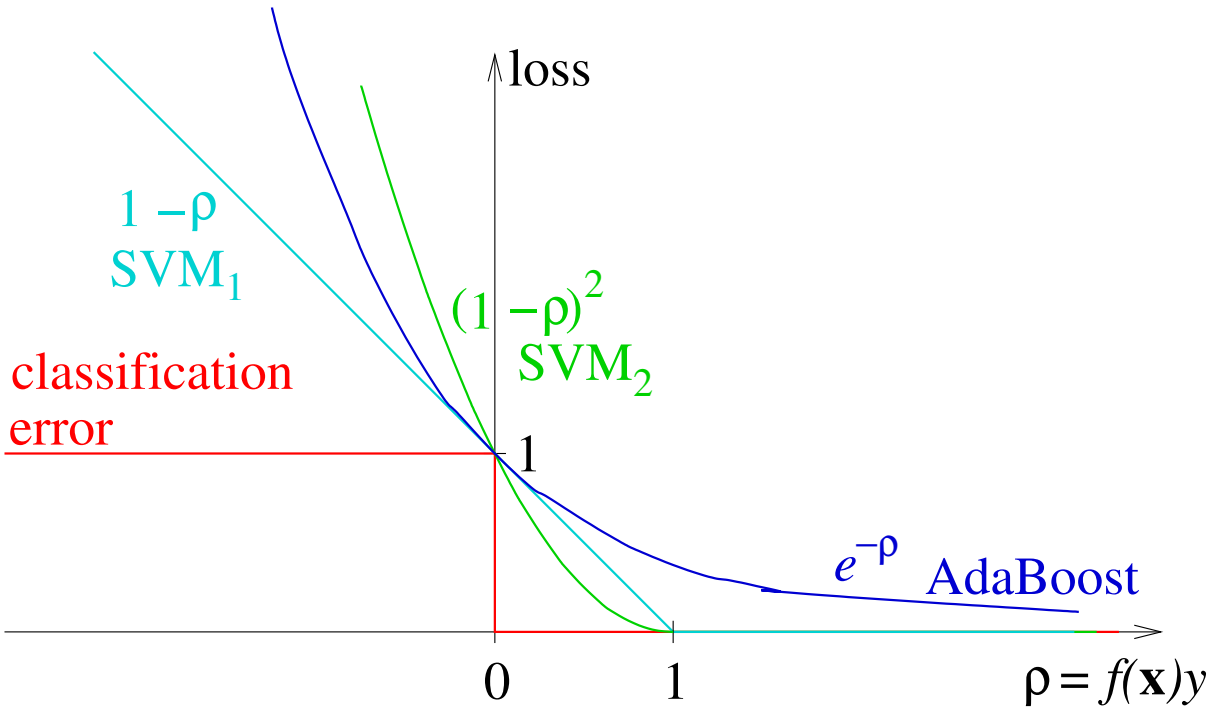
AdaBoost

- Loss functions



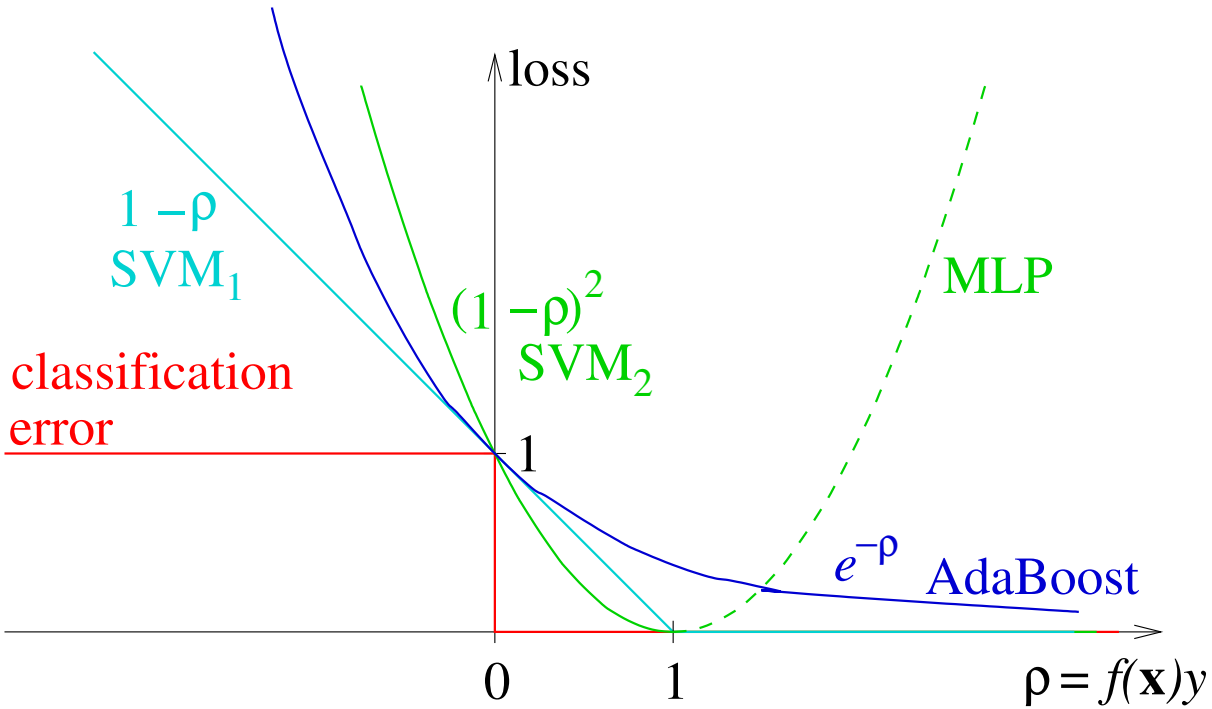
AdaBoost

- Loss functions



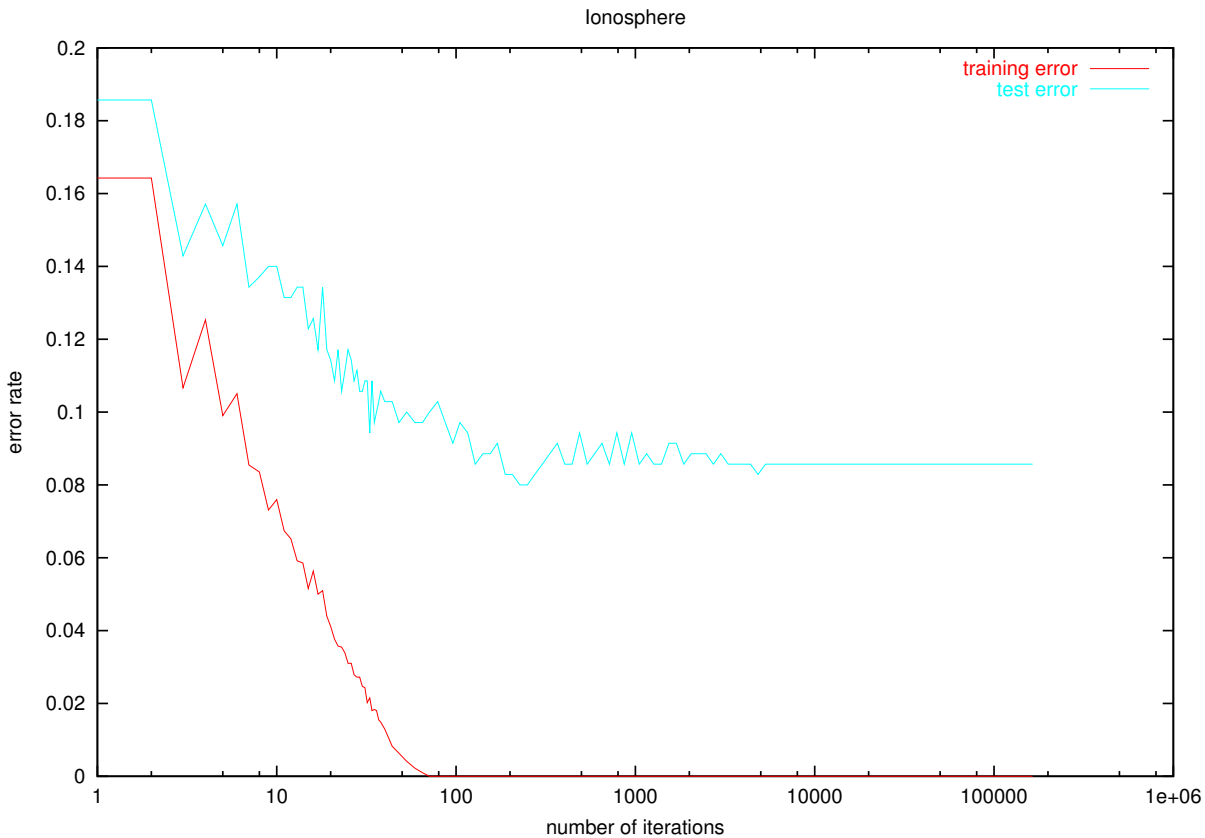
AdaBoost

- Loss functions



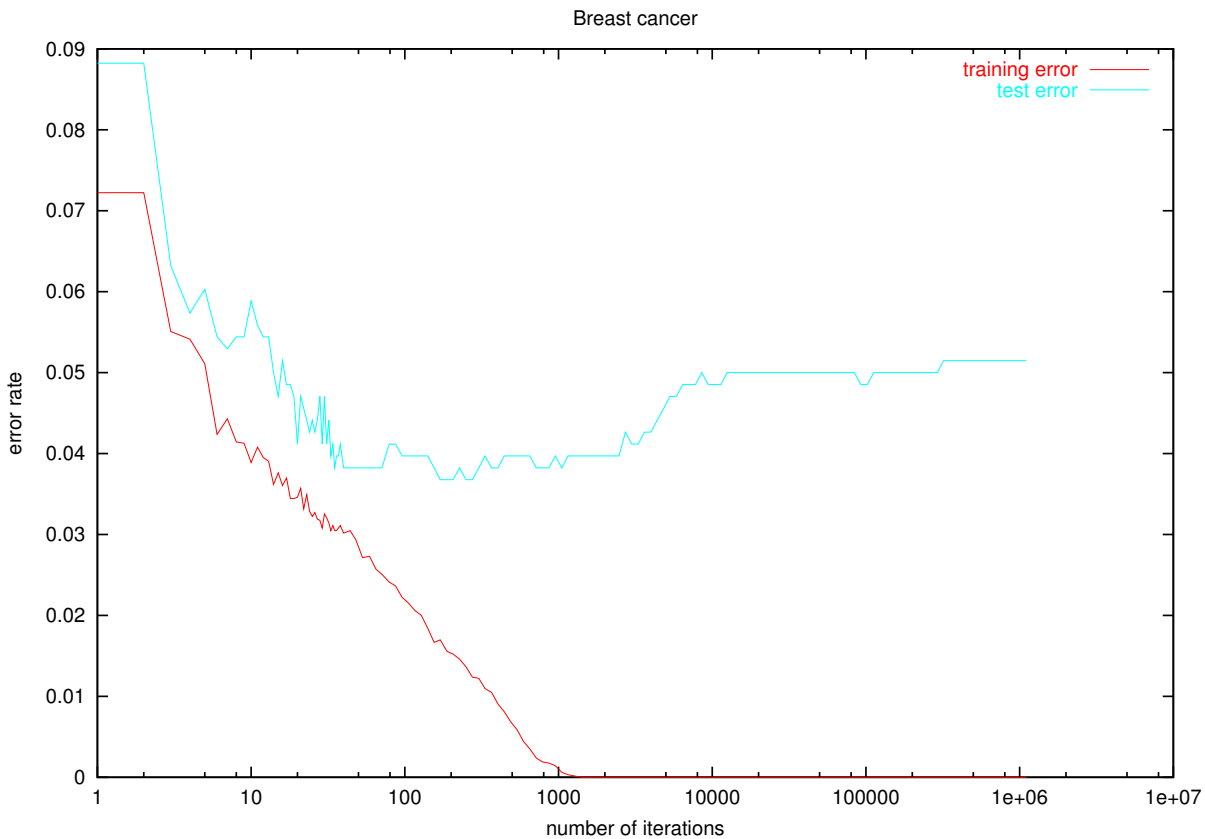
AdaBoost

- Learning curves



AdaBoost

- Learning curves



Marginal AdaBoost

- Theory suggests:
  - minimization of the robust (marginal) training error

$$\widehat{R}_\theta(f) = \frac{1}{n}\sum_{i=1}^{n} I\{\widetilde{\rho}_i < \theta\}$$

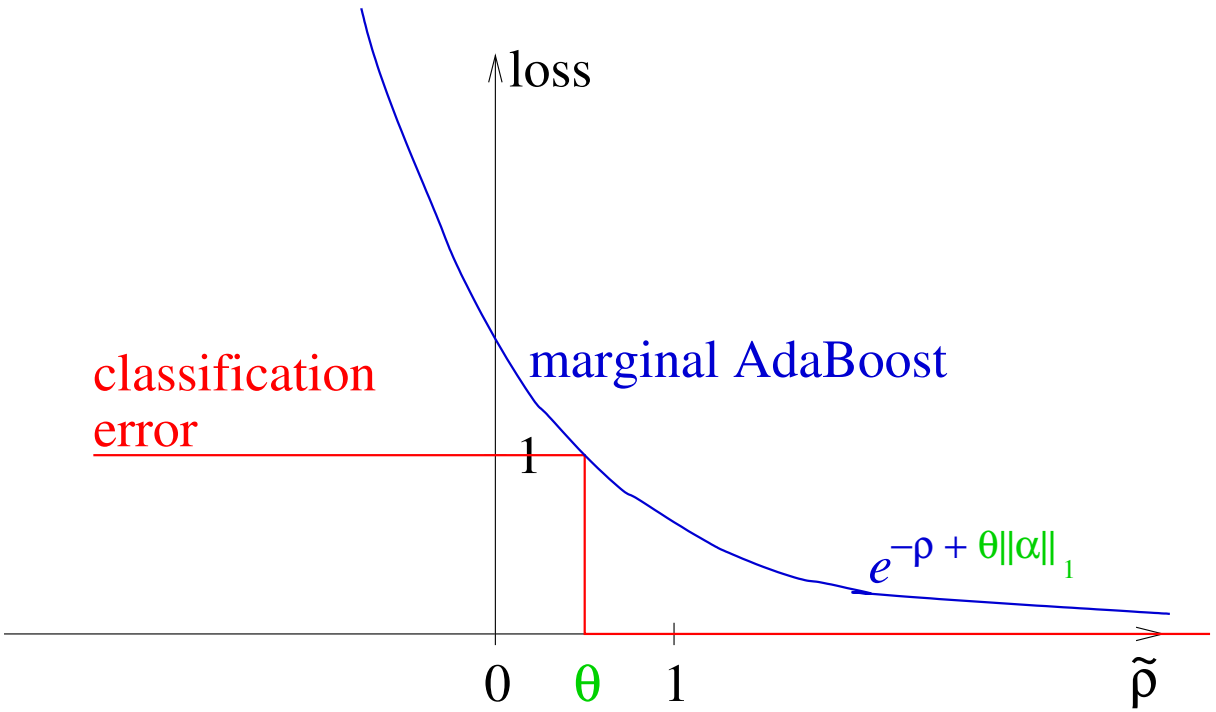
  improves generalization.

- Idea [Breiman '99, Rätsch - Warmuth '02]
  - directly minimize the exponential marginal risk

$$\widehat{R}_\theta^{(e)}(f) = \frac{1}{n}\sum_{i=1}^{n} e^{-\rho_i + \theta\|\alpha\|_1}$$

Marginal AdaBoost

- Loss functions



Marginal AdaBoost

MARGINALADABOOST$(D_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)), \text{BASE}(D_n, \mathbf{w}), \theta, T)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base classifier*

4 $\gamma \leftarrow \sum_{i=1}^{n} w_i h^{(t)}(\mathbf{x}_i) y_i$ ▷ *edge*

5 $\alpha^{(t)} \leftarrow \frac{1}{2} \log\left(\frac{1+\gamma}{1-\gamma}\right) - \frac{1}{2} \log\left(\frac{1+\theta}{1-\theta}\right)$ ▷ *weight of base classifier*

6 **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\gamma \geq \theta$

7 **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

8 **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

9 $w_i^{(t+1)} \leftarrow w_i^{(t)} \dfrac{\exp\left(-\alpha^{(t)} h^{(t)}(\mathbf{x}_i) y_i\right)}{\sum_{j=1}^{t} w_j^{(t)} \exp\left(-\alpha^{(t)} h^{(t)}(\mathbf{x}_j) y_j\right)}$

10 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

Outline

- The supervised learning model
- AdaBoost
- RegBoost
- The graph Laplacian regularizer
- Experimental results

RegBoost

- Idea: let θ depend on the complexity of the base classifier

$$\theta = 2\lambda P(h)$$

  - $P(h)$ is an arbitrary penalty functional
  - λ is a penalty coefficient

- Base objective
  - minimize

$$R_1(h) = \varepsilon + \lambda P(h)$$

  - standard regularized error minimization

RegBoost

$\textsc{RegBoost}\left(D_n, \textsc{Base}(D_n, \mathbf{w}, P), P(h), \lambda, T\right)$

1 $\mathbf{w} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weight distribution on the training data*

2 **for** $t \leftarrow 1$ **to** T

3 $\quad h^{(t)} \leftarrow \textsc{Base}(D_n, \mathbf{w}, P)$ ▷ *base classifier*

4 $\quad \gamma \leftarrow \sum_{i=1}^{n} w_i h^{(t)}(\mathbf{x}_i) y_i$ ▷ *edge*

5 $\quad \theta \leftarrow 2\lambda P(h^{(t)})$ ▷ *edge offset*

6 $\quad \alpha^{(t)} \leftarrow \frac{1}{2}\log\left(\frac{1+\gamma}{1-\gamma}\right) - \frac{1}{2}\log\left(\frac{1+\theta}{1-\theta}\right)$ ▷ *weight of base classifier*

7 $\quad$ **if** $\alpha^{(t)} \leq 0$ ▷ *equivalent to* $\gamma \geq \theta$

8 $\quad\quad$ **return** $f^{(t-1)}(\cdot) = \sum_{j=1}^{t-1} \alpha^{(j)} h^{(j)}(\cdot)$

9 $\quad$ **for** $i \leftarrow 1$ **to** n ▷ *weight distribution update*

10 $\quad\quad w_i^{(t+1)} \leftarrow w_i^{(t)} \dfrac{\exp\left(-\alpha^{(t)} h^{(t)}(\mathbf{x}_i) y_i\right)}{\sum_{j=1}^{t} w_j^{(t)} \exp\left(-\alpha^{(t)} h^{(t)}(\mathbf{x}_j) y_j\right)}$

11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

RegBoost

- Interpretation 1: L_1 weight decay
  - average edge offset ($\sim$ average penalty)

$$\bar{\theta} = \frac{\sum_{t=1}^{T} \alpha^{(t)} \theta^{(t)}}{\sum_{t=1}^{T} \alpha^{(t)}}$$

  - RegBoost minimizes

$$\frac{1}{n} \sum_{i=1}^{n} \exp\left(-\rho_i + \bar{\theta} \|\alpha\|_1\right)$$

  - weight decay coefficient $\bar{\theta}$ is adaptive

RegBoost

- Interpretation 2: adaptive marginal error minimization
  - REGBOOST minimizes
$$\frac{1}{n}\sum_{i=1}^{n} I\{\widetilde{\rho}_i < \bar{\theta}\}$$
  - margin parameter $\bar{\theta}$ is adaptive

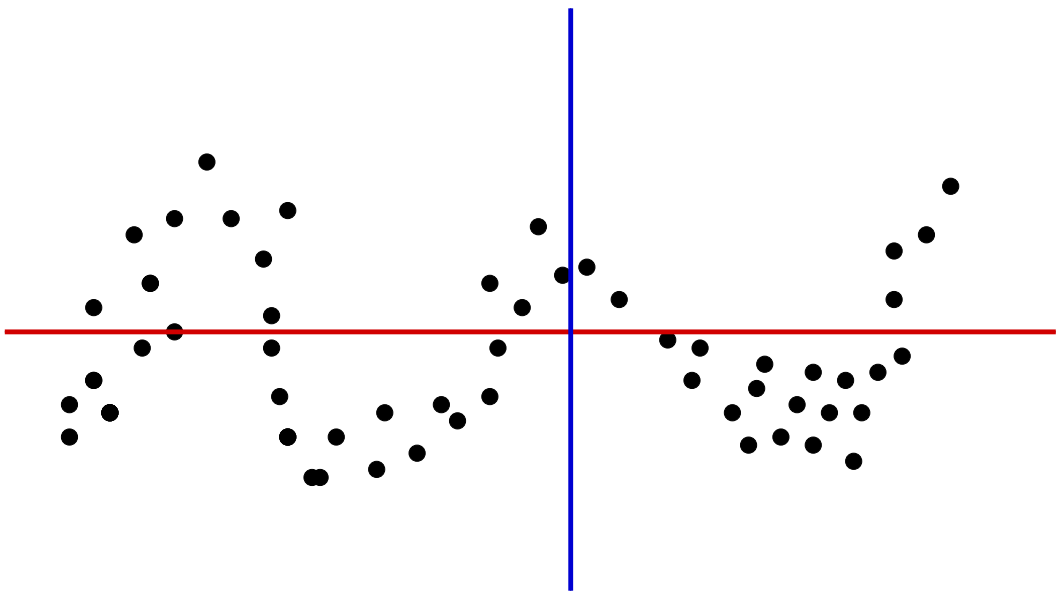
- Interpretation 3: pool reduction
  - decreased number (complexity) of base classifiers:
$$\#\{h : \gamma > \theta\} \;<\; \#\{h : \gamma > 0\}$$

Outline

- The supervised learning model
- AdaBoost
- RegBoost
- The graph Laplacian regularizer
- Experimental results

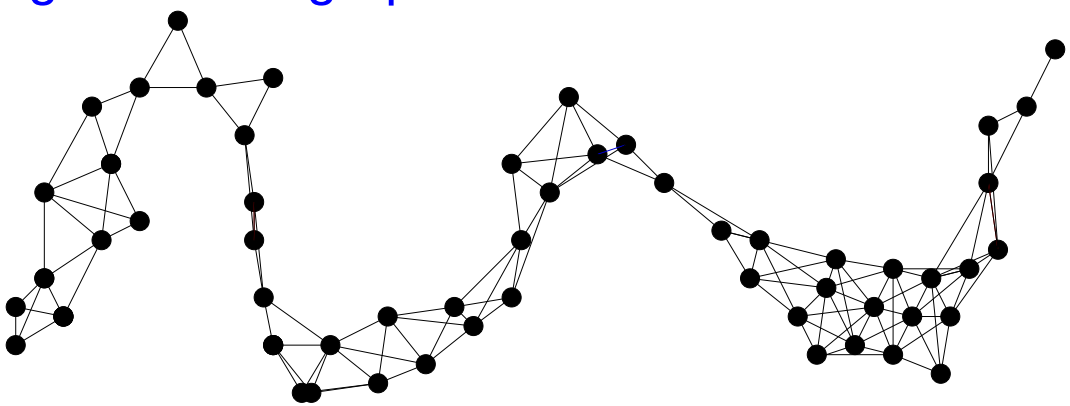
The graph Laplacian regularizer

- Motivation:



The graph Laplacian regularizer

- The neighborhood graph:



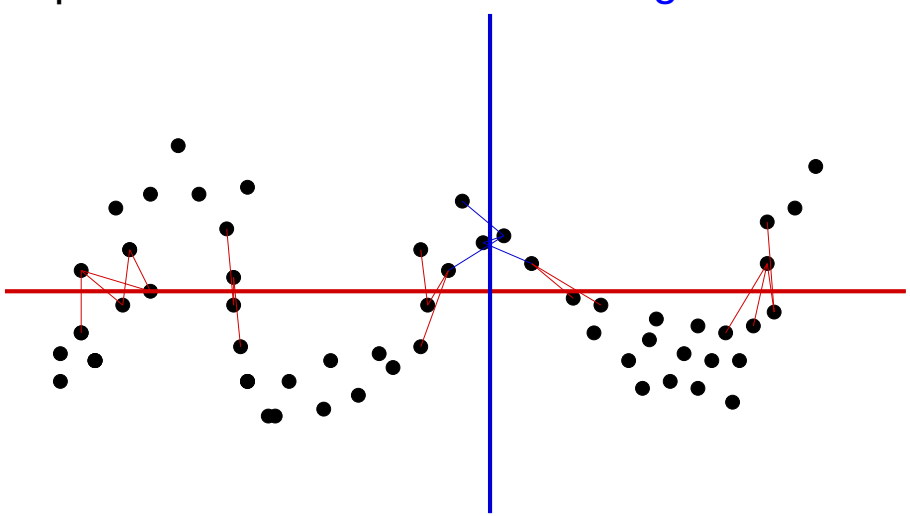
  - $G = (\mathcal{V}, \mathcal{E})$
  - $\mathcal{V} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
  - $\mathcal{E} = \{(\mathbf{x}_i, \mathbf{x}_j) : \|\mathbf{x}_i - \mathbf{x}_j\| < r\}$ (or k nearest neighbors)
  - adjacency matrix: $W_{ij} = I\{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{E}\}$

The graph Laplacian regularizer

- The penalty:

$$P_{\mathcal{L}}(h) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \big(h(\mathbf{x}_i) - h(\mathbf{x}_j)\big)^2 W_{ij},$$

  - $P_{\mathcal{L}}(h)$ is proportional to the number of neighbors that h separates



  - related to dimensionality reduction [Belkin - Niyogi '03] and spectral clustering [Shi - Malik '00]

Outline

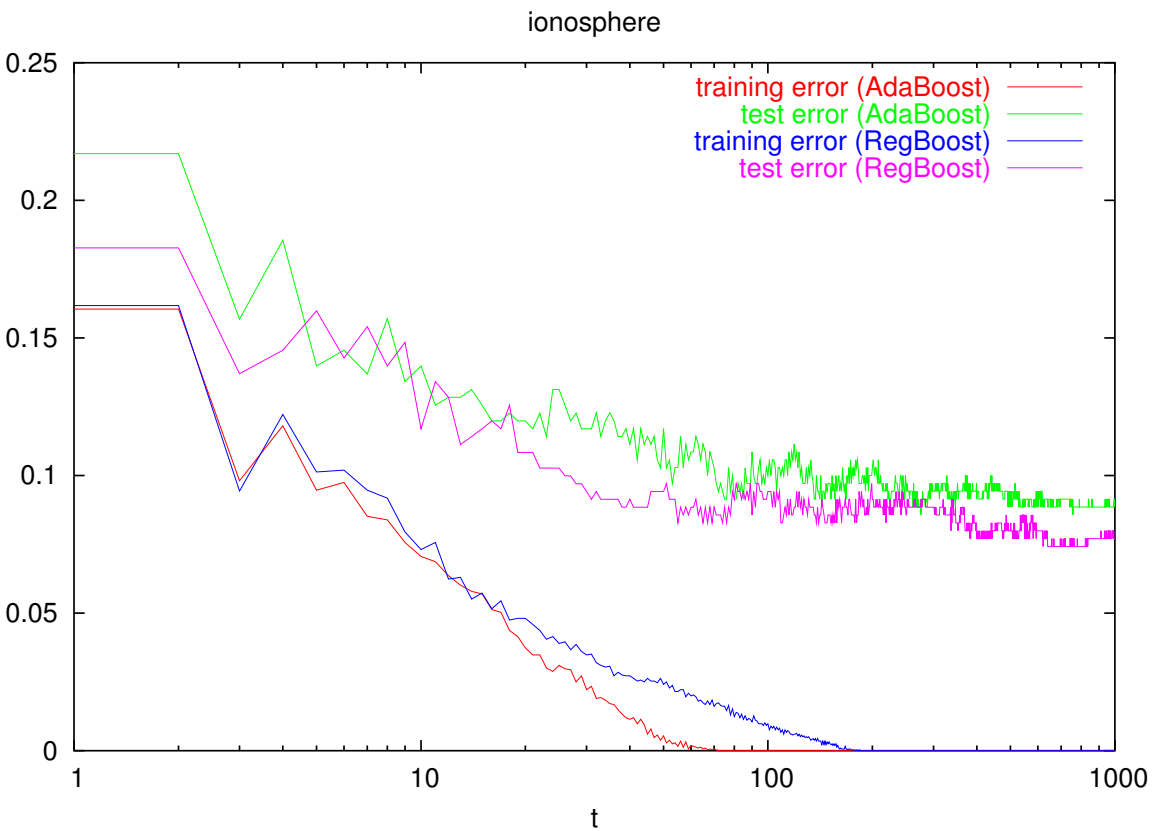
- The supervised learning model
- AdaBoost
- RegBoost
- The graph Laplacian regularizer
- Experimental results

Experimental results

- Setup
  - decision stumps
  - 8 nearest neighbors
  - $5\times$ CV for λ
  - $10\times$ CV for test
  - UCI datasets

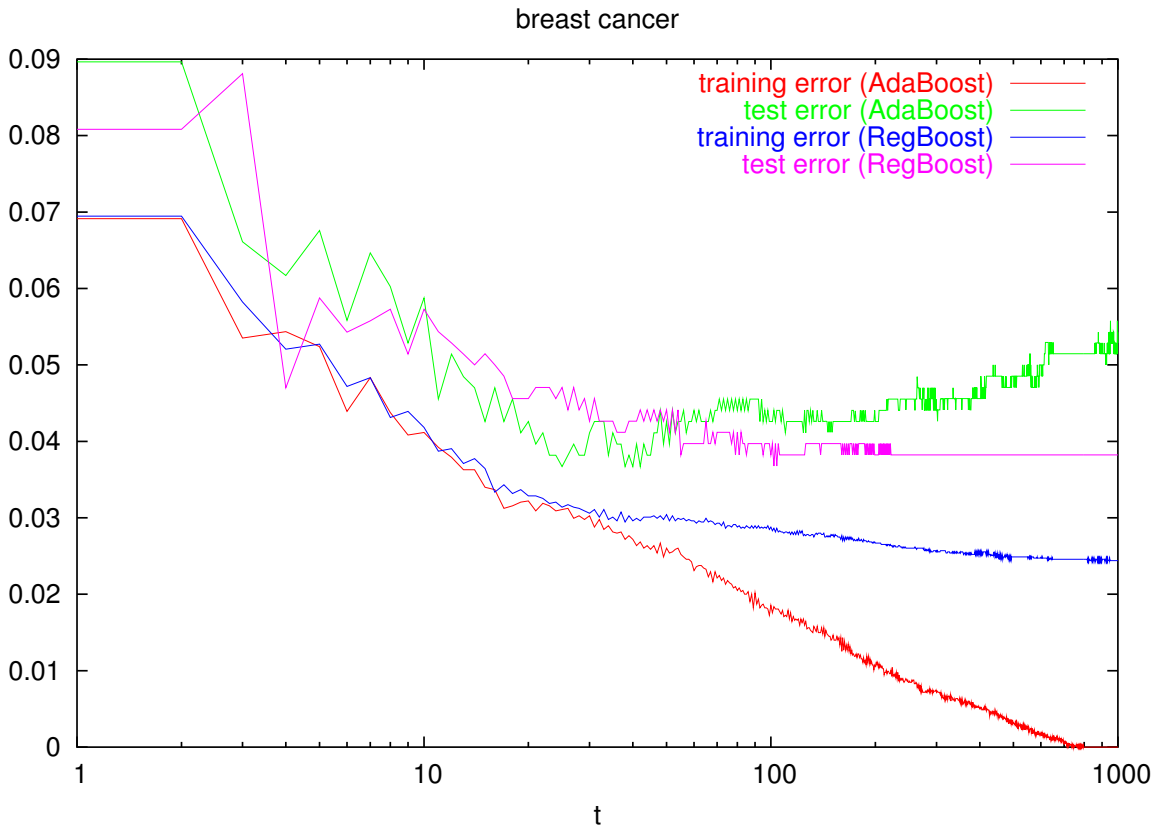
Experimental results

- Ionosphere



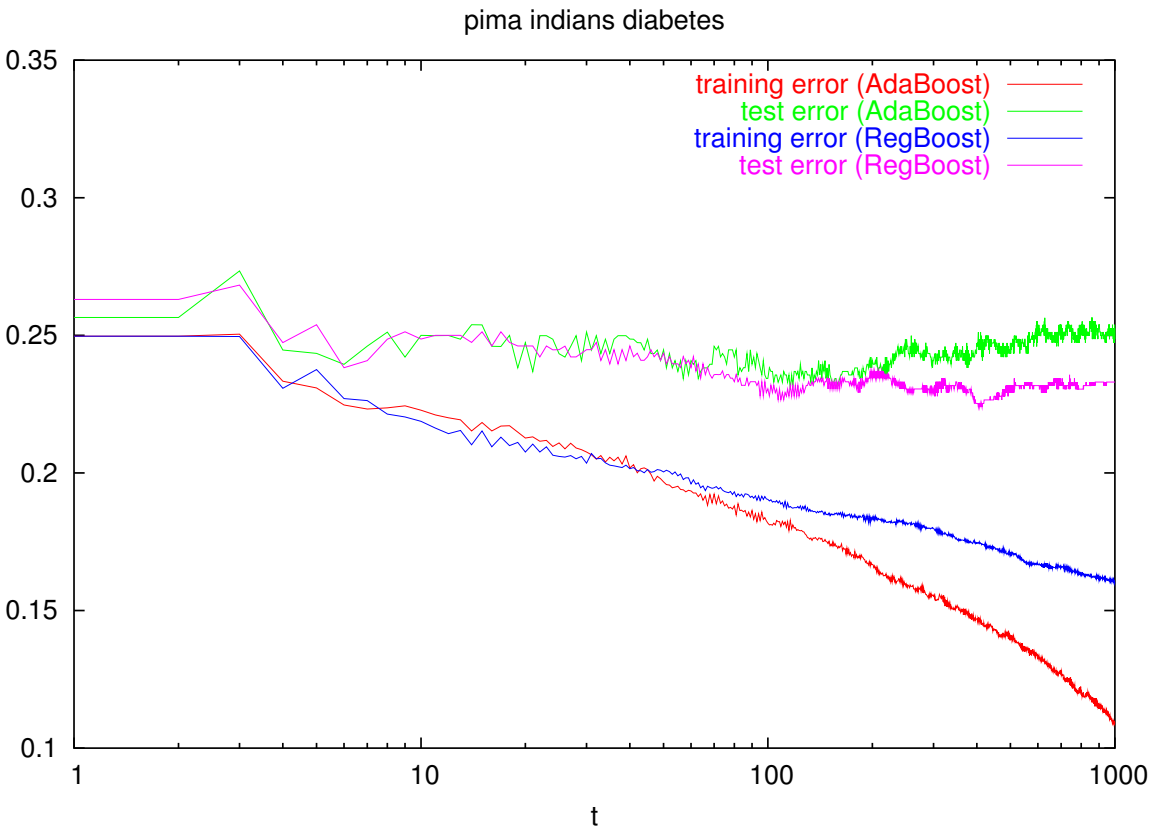
Experimental results

- Breast cancer



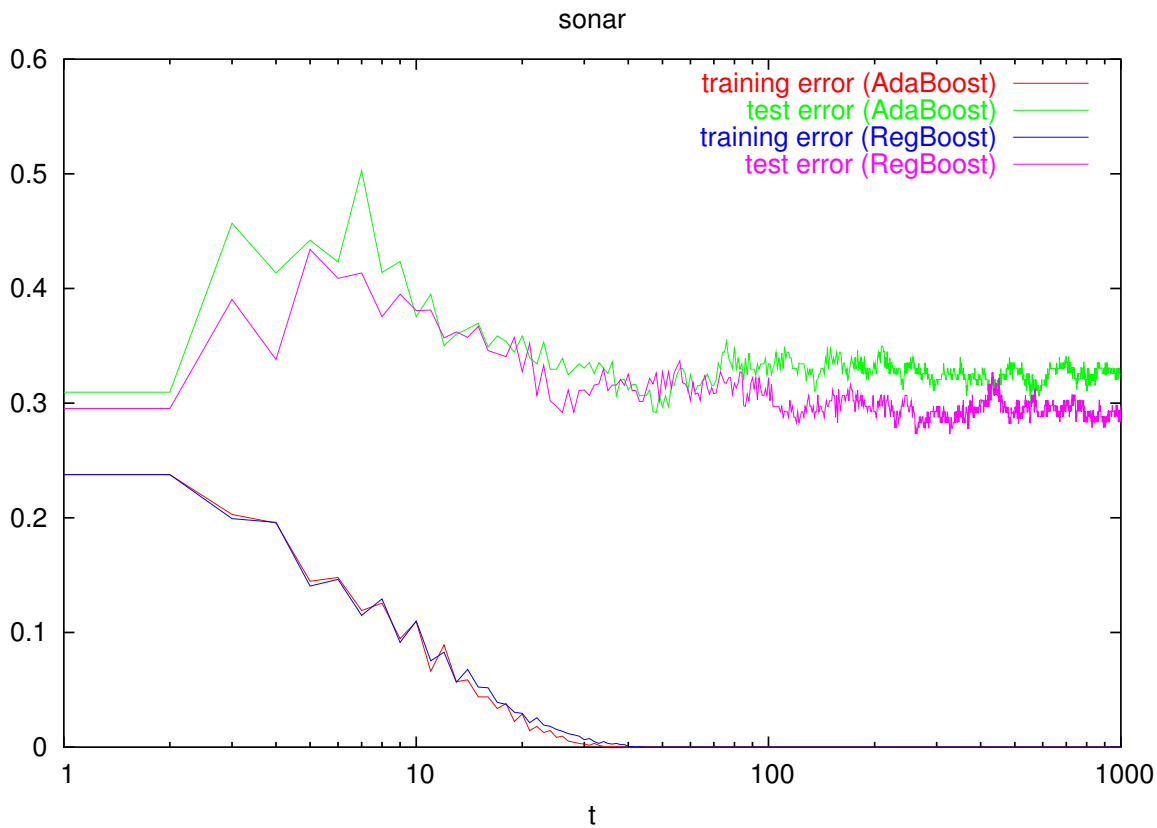
Experimental results

- Pima Indians diabetes



Experimental results

- Sonar



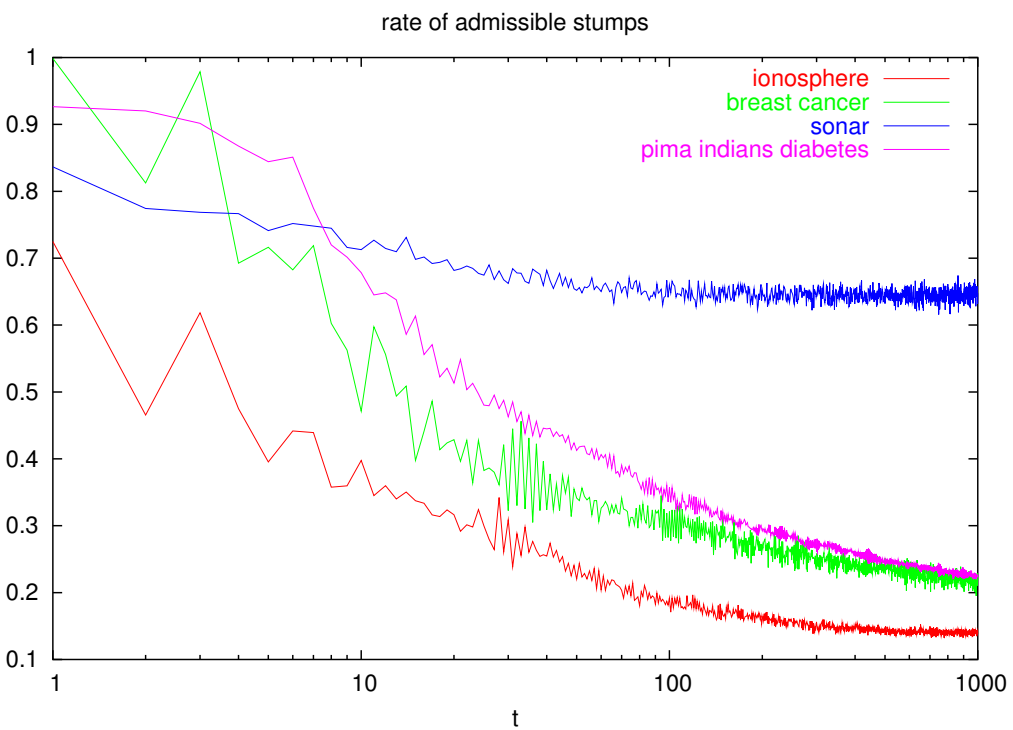
Experimental results

- Discussion
  - test error is always better, although not significantly
  - the difference between test and training errors decreases significantly
  - final classifier is significantly sparser

data set	# of stumps ($T = 1000$)	
	ADABOOST	REGBOOST
ionosphere	182	114
breast cancer	58	30
pima Indians diabetes	175	91
sonar	233	199

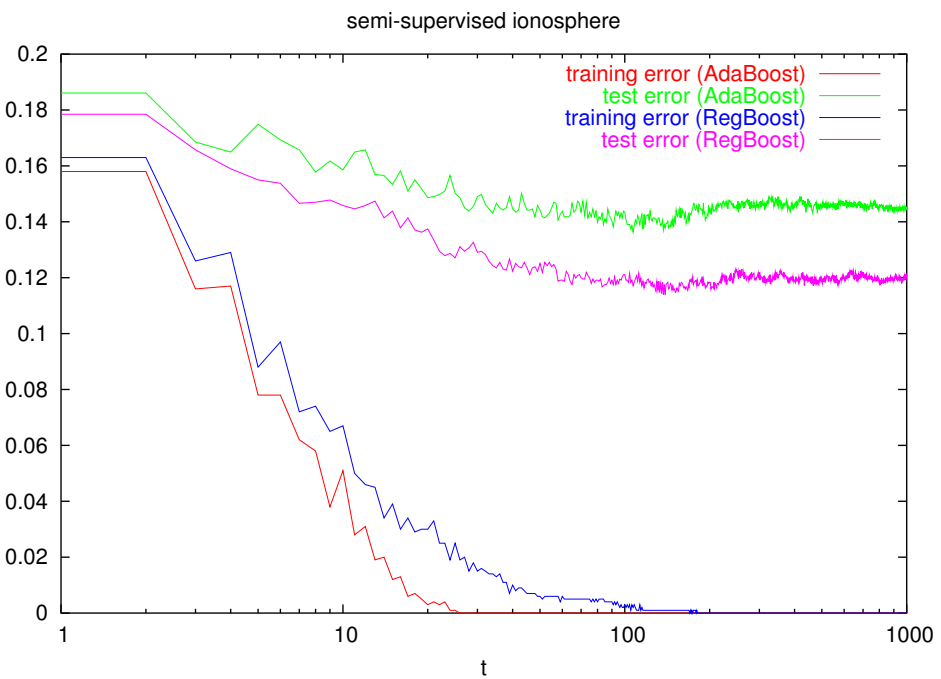
Experimental results

- Pool reduction
  - the number of admissible stumps drops significantly



Experimental results

- Semi-supervised learning
  - the penalty can be calculated on unlabeled points
  - ionosphere, 100 labeled, 251 unlabeled



Conclusion

- Combination of two powerful ideas, boosting and manifold learning
- Simple modification to AdaBoost
- Improves generalization, sparsity
- Semi-supervised approach: improves on AdaBoost and beats other manifold-based semi-supervised algorithms by far