The return of AdaBoost.MH: Multi-class Hamming trees and product

Balázs Kégl

University of Paris Sud / CNRS

LEAR

October 9, 2013

Outline

- Binary AdaBoost
  - algorithm
  - convergence
  - binary stumps and trees
- Multi-class / multi-label AdaBoost.MH
  - multi-class / multi-label classification
  - multi-class / multi-label stumps, trees, and products
- Some 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: $f : \mathbb{R}^d \mapsto [-1, 1]$

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

The supervised learning model

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

    $\text{ALGO}(D_n) \to f$

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

AdaBoost (Freund and Schapire, 1997)

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \text{BASE}(\cdot,\cdot), T)$

1 $\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

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

3 $h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ ▷ *calling the base learner*

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

5 $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

6 **for** $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

7 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

8 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

9 **else**

10 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

AdaBoost

ADABOOST($D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, BASE$(\cdot,\cdot)$, T)

$\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

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

  $h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ ▷ *calling the base learner*

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

  $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

  for $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

    if $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

    else

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

AdaBoost

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, BASE$(\cdot,\cdot), T)$

1 $\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

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

3 $h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ ▷ *calling the base learner*

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

5 $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

6 **for** $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

7 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

8 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

9 **else**

10 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

AdaBoost

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, BASE$(\cdot,\cdot)$, $T)$

1 $\quad \mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ $\quad \triangleright$ *initial weights*

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

3 $\quad\quad h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ $\quad \triangleright$ *calling the base learner*

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

5 $\quad\quad \alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ $\quad \triangleright$ *coefficient of* $h^{(t)}$

6 $\quad\quad$ **for** $i \leftarrow 1$ **to** n $\quad \triangleright$ *re-weighting the points*

7 $\quad\quad\quad$ **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

8 $\quad\quad\quad\quad w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

9 $\quad\quad\quad$ **else**

10 $\quad\quad\quad\quad w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

AdaBoost

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \text{BASE}(\cdot,\cdot), T)$

$\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

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

  $h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ ▷ *calling the base learner*

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

  $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

  for $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

    if $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

    else

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

AdaBoost

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \text{BASE}(\cdot,\cdot), T)$

$\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

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

  $h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ ▷ *calling the base learner*

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

  $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

  for $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

    if $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

    else

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

AdaBoost

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \text{BASE}(\cdot,\cdot), T)$

$\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

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

  $h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ ▷ *calling the base learner*

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

  $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

  for $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

    if $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

    else

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

AdaBoost

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \text{BASE}(\cdot,\cdot), T)$

1 $\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

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

3 $h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ ▷ *calling the base learner*

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

5 $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

6 **for** $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

7 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

8 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

9 **else**

10 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

AdaBoost

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \text{BASE}(\cdot,\cdot), T)$

1 $\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

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

3 $h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ ▷ *calling the base learner*

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

5 $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

6 **for** $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

7 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

8 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

9 **else**

10 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

AdaBoost

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \text{BASE}(\cdot,\cdot), T)$

1 $\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

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

3 $h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ ▷ *calling the base learner*

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

5 $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

6 **for** $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

7 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

8 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

9 **else**

10 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

AdaBoost

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \text{BASE}(\cdot,\cdot), T)$

$\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

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

  $h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ ▷ *calling the base learner*

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

  $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

  for $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

    if $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

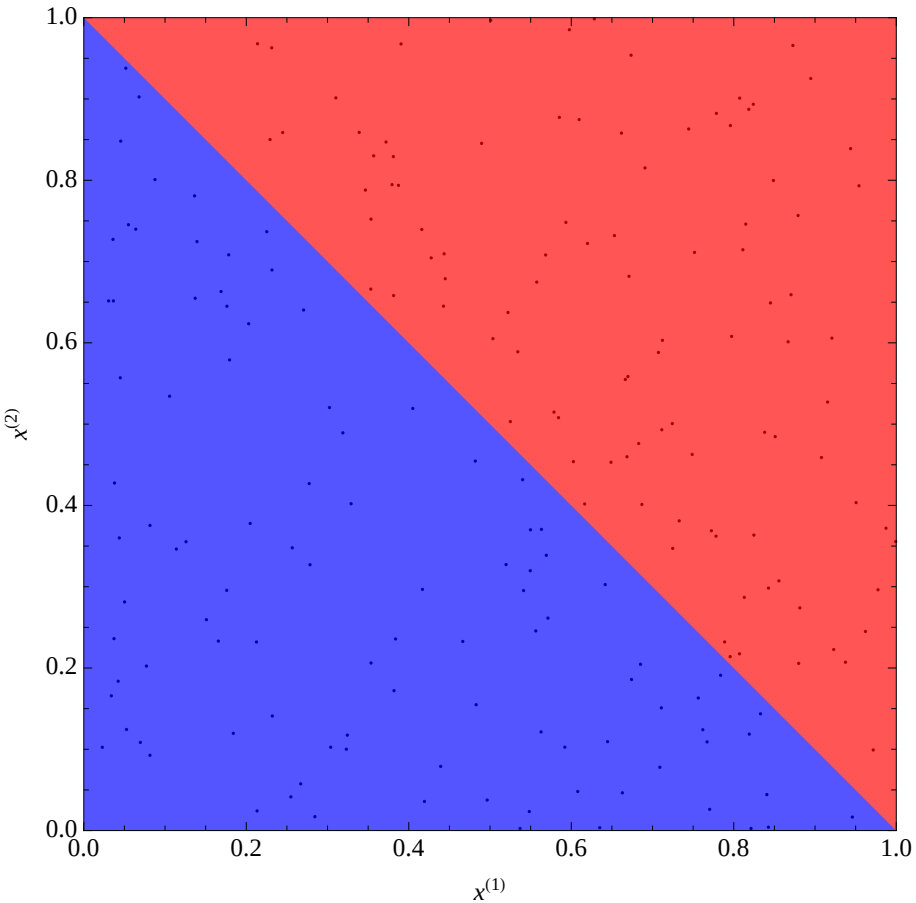
      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

    else

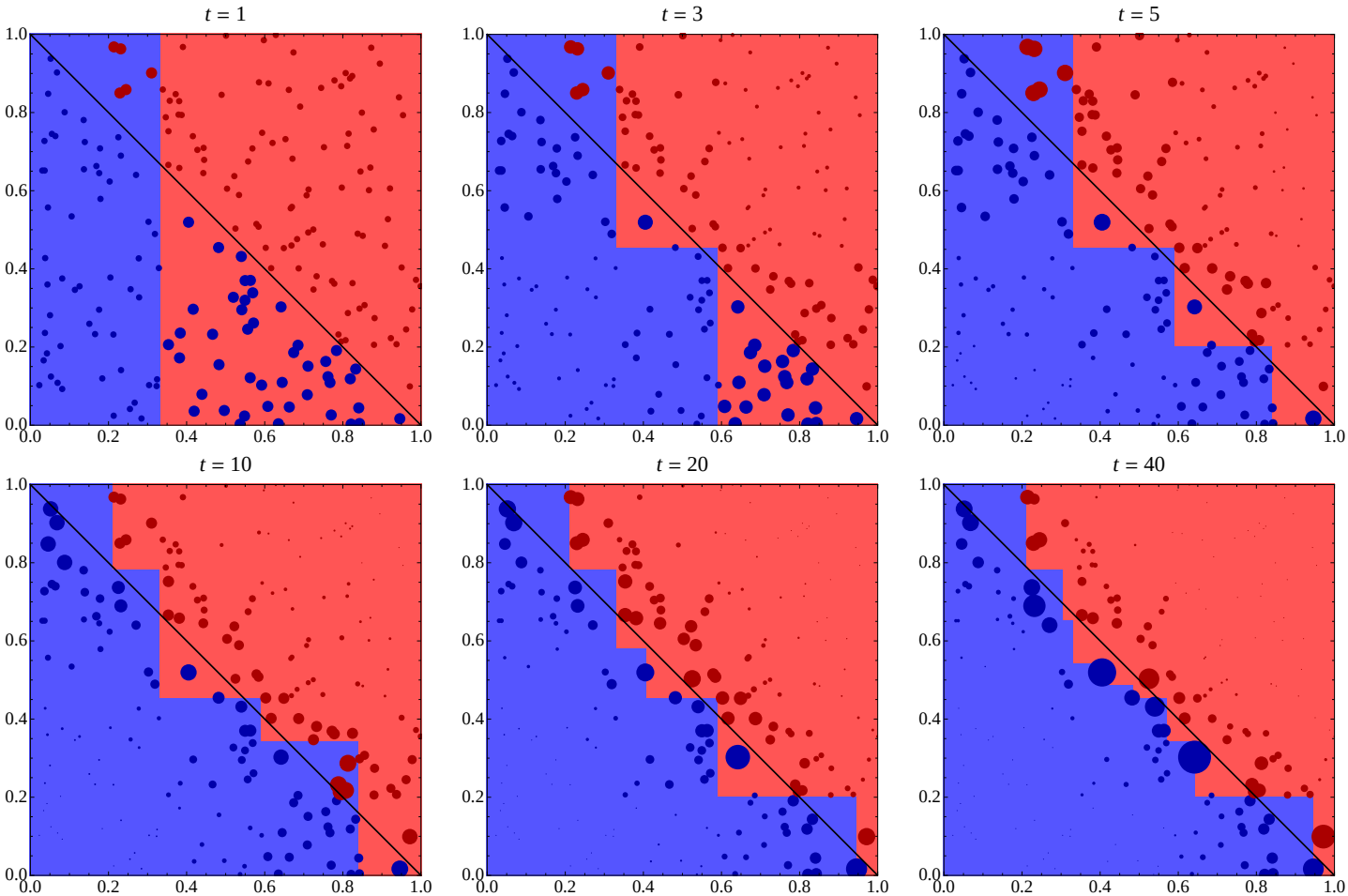
      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

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

Toy data set

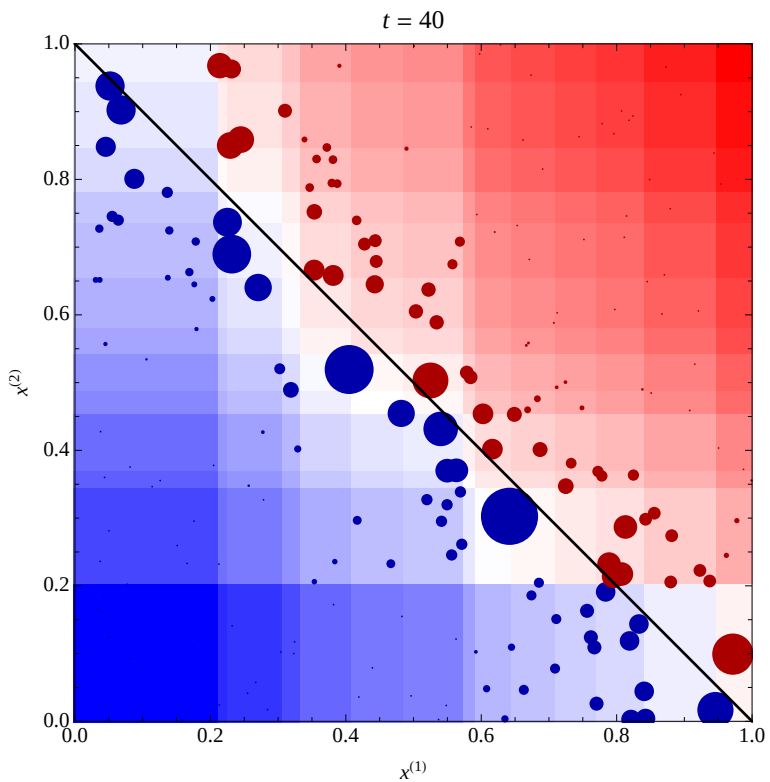


AdaBoost on toy data set



Toy data set

- $w_i^{(t+1)} \sim \exp\left(-\underbrace{f^{(T)}(\mathbf{x}_i)y_i}_{\text{margin}}\right)$



Outline

- Binary AdaBoost
  - algorithm
  - convergence
  - binary stumps and trees
- Multi-class / multi-label AdaBoost.MH
  - multi-class / multi-label classification
  - multi-class / multi-label stumps, trees, and products
- Some results

Convergence of AdaBoost

- If $\gamma^{(t)} \geq \rho > 0$ then the training error is 0 after

$$T = \left\lceil \frac{2 \ln n}{\rho^2} \right\rceil + 1$$

iterations.

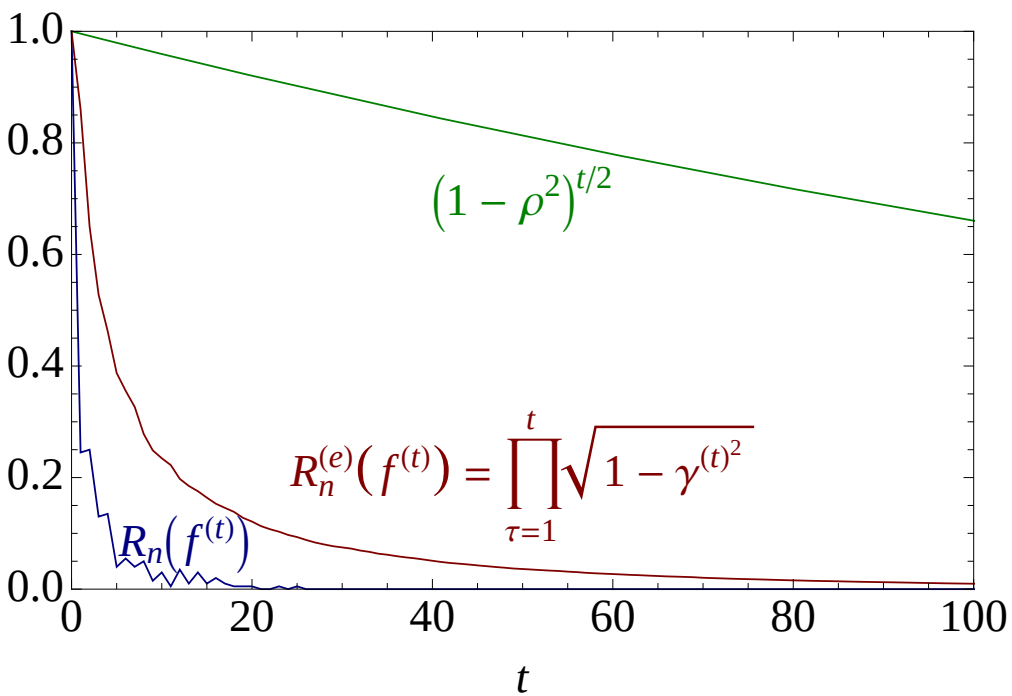
$$\begin{aligned}
R_n\left(f^{(T)}\right) &\triangleq \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left\{\operatorname{sign}\left(f^{(T)}(\mathbf{x}_i)\right) \neq y_i\right\} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left\{f^{(T)}(\mathbf{x}_i) y_i < 0\right\} \\
&\leq \frac{1}{n}\sum_{i=1}^{n} \exp\left(-f^{(T)}(\mathbf{x}_i) y_i\right) \\
&= \prod_{t=1}^{T} \sqrt{1-\gamma^{(t)2}} \\
&\leq \left(\sqrt{1-\rho^2}\right)^T \\
&\leq \exp\left(-T\rho^2/2\right).
\end{aligned}$$

Convergence of AdaBoost

- If $\gamma^{(t)} \geq \rho > 0$ then the training error is 0 after

$$T = \left\lceil \frac{2 \ln n}{\rho^2} \right\rceil + 1$$

iterations.



Convergence of AdaBoost

- Margin maximization
  - Let the optimal normalized minimum margin be

$$\rho^* = \max_{N,(h_1,\ldots,h_N),(\alpha_1,\ldots,\alpha_N)} \min_i \frac{\sum_{j=1}^N \alpha_j h_j(\mathbf{x}_i) y_i}{\sum_{j=1}^N \alpha_j}$$

  - then

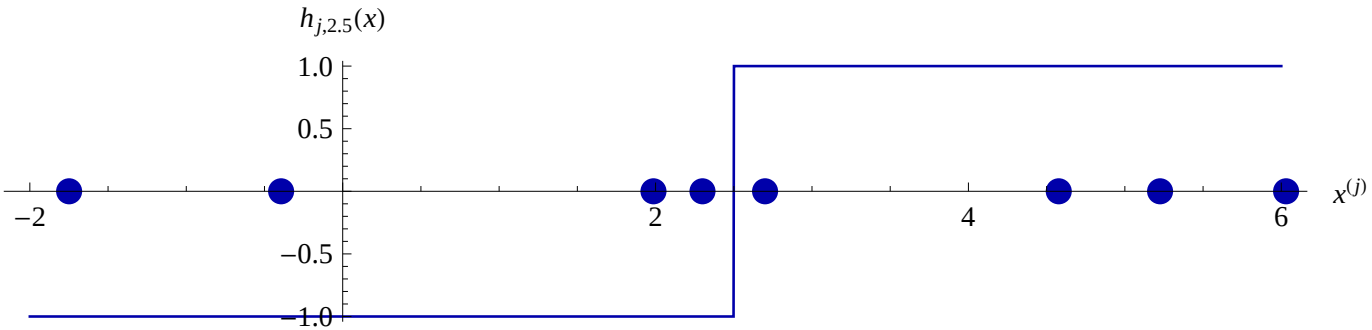
$$\rho^{(T)} = \min_i \frac{\sum_{t=1}^T \alpha^{(t)} h^{(t)}(\mathbf{x}_i) y_i}{\sum_{t=1}^T \alpha^{(t)}} > \frac{\rho^*}{2} - \varepsilon$$

    after $T \sim \ln n$ iterations

Outline

- Binary AdaBoost
  - algorithm
  - convergence
  - binary stumps and trees
- Multi-class / multi-label AdaBoost.MH
  - multi-class / multi-label classification
  - multi-class / multi-label stumps, trees, and products
- Some results

Boosting decision stumps



$$h_{j,b}(\mathbf{x}) = \varphi_{j,b}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^{(j)} \geq b, \\ -1 & \text{otherwise,} \end{cases}$$

- Goal: to maximize the edge $\gamma = \sum_{i=1}^{n} w_i h(\mathbf{x}_i) y_i = 1 - 2 \times \text{error}$
- Can be learned in $\theta(nd)$ time (if features are pre-sorted)
- Often sub-optimal test results in practice
- Can be called recursively to construct decision trees

Outline

- Binary AdaBoost
  - algorithm
  - convergence
  - binary stumps and trees
- Multi-class / multi-label AdaBoost.MH
  - multi-class / multi-label classification
  - multi-class / multi-label stumps, trees, and products
- Some results

Multi-class / multi-label classification

- observation vector: $\mathbf{x} \in \mathbb{R}^d$
- class label index: $\ell \in \{1, \ldots, K\}$
- class label vector: $\mathbf{y} \in \{-1, 1\}^K$
- discriminant function: $\mathbf{f} : \mathbb{R}^d \mapsto [-1, 1]^K$
- multi-class classifier: $\ell_{\mathbf{f}} : \mathbb{R}^d \mapsto \{1, \ldots, K\}$

$$\ell_{\mathbf{f}}(\mathbf{x}) = \arg\max_{\ell} f_\ell(\mathbf{x})$$

- multi-label classifier: $\mathbf{g}_{\mathbf{f}} : \mathbb{R}^d \mapsto \{-1, 1\}^K$

$$\mathbf{g}_{\mathbf{f}}(\mathbf{x}) = \mathbf{sign}\big(\mathbf{f}(\mathbf{x})\big)$$

Multi-class losses

- single-label: $\widehat{R}_{\mathbb{I}}(\mathbf{f}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{\ell_i \neq \ell_{\mathbf{f}}(\mathbf{x}_i)\}$
- Hamming: $\widehat{R}_{\mathsf{H}}(\mathbf{f},\mathbf{W}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{\ell=1}^{K} w_{i,\ell}\mathbb{I}\left\{\mathrm{sign}\left(f_\ell(\mathbf{x}_i)\right) \neq y_{i,\ell}\right\}$
- Exponential Hamming: $\widehat{R}_{\mathsf{EXP}}(\mathbf{f},\mathbf{W}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{\ell=1}^{K} w_{i,\ell}\exp\left(-f_\ell(\mathbf{x}_i)y_{i,\ell}\right)$
- Exponential asymmetric: $\widehat{R}_{\mathsf{EXP.AS}}(\mathbf{f}) = \frac{1}{n(K-1)}\sum_{i=1}^{n}\sum_{\ell=1}^{K}\exp\left(f_\ell(\mathbf{x}_i) - f_{\ell_i}(\mathbf{x}_i)\right)$
- Hinge: $\widehat{R}_{\mathsf{HINGE}}(\mathbf{f}) = \frac{1}{n}\sum_{i=1}^{n}\max_{\ell}\left\{\mathbb{I}\{\ell_i \neq \ell\} + f_\ell(\mathbf{x}_i) - f_{\ell_i}(\mathbf{x}_i)\right\}$

Multi-class / multi-label AdaBoost.MH

(Schapire and Singer, 1999)

ADABOOST.MH$(\mathbf{X}, \mathbf{Y}, \mathbf{W}^{(1)}, \text{BASE}(\cdot,\cdot,\cdot), T)$

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

2 $\quad \mathbf{h}^{(t)}(\cdot) \leftarrow \text{BASE}\left(\mathbf{X}, \mathbf{Y}, \mathbf{W}^{(t)}\right)$

3 $\quad$ **for** $i \leftarrow 1$ **to** n **for** $\ell \leftarrow 1$ **to** K

4 $$w_{i,\ell}^{(t+1)} \leftarrow w_{i,\ell}^{(t)} \frac{\exp\left(-h_\ell^{(t)}(\mathbf{x}_i) y_{i,\ell}\right)}{\sum_{i'=1}^{n} \sum_{\ell'=1}^{K} w_{i',\ell'}^{(t)} \exp\left(-h_{\ell'}^{(t)}(\mathbf{x}_{i'}) y_{i',\ell'}\right)}$$

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

- Usual multi-class initial weights $\mathbf{W}^{(1)}$

$$w_{i,\ell}^{(1)} = \begin{cases} \frac{1}{2n} & \text{if } \ell = \ell_i \text{ (i.e., if } y_{i,\ell} = 1), \\ \frac{1}{2n(K-1)} & \text{otherwise (i.e., if } y_{i,\ell} = -1) \end{cases}$$

Outline

- Binary AdaBoost
  - algorithm
  - convergence
  - binary stumps and trees
- Multi-class / multi-label AdaBoost.MH
  - multi-class / multi-label classification
  - multi-class / multi-label stumps, trees, and products
- Some results

Multi-class / multi-label stumps

- Independent stumps (one-against all)

$$\mathbf{h}(\mathbf{x}) = \left(\alpha_1 \varphi_{j_1,b_1}(\mathbf{x}), \ldots, \alpha_K \varphi_{j_K,b_K}(\mathbf{x})\right)$$

- "Multi-task" or factorized stumps

$$\mathbf{h}(\mathbf{x}) = \alpha\, \mathbf{v}\, \varphi_{j,b}(\mathbf{x})$$

where $\mathbf{v}$ is the vote or code vector

$$\mathbf{v} = \{-1, 1\}^K$$

  - both take $O(nKd)$ time to learn
  - factorized stump is more regularized
  - factorized stump can be used to construct multi-class trees

Decision products (Kégl and Busa-Fekete, 2009)

$$\mathbf{h}(\mathbf{x}) = \bigotimes_{j=1}^{m} \mathbf{h}_j(\mathbf{x})$$

- The edge $\gamma = \sum_{i=1}^{n} \sum_{\ell=1}^{K} w_{i,\ell} \prod_{j=1}^{m} h_{j,\ell}(\mathbf{x}_i) y_{i,\ell}$ is multiplicative
- $\mathbf{h}_j(\mathbf{x})$ can be learned by defining the virtual labels

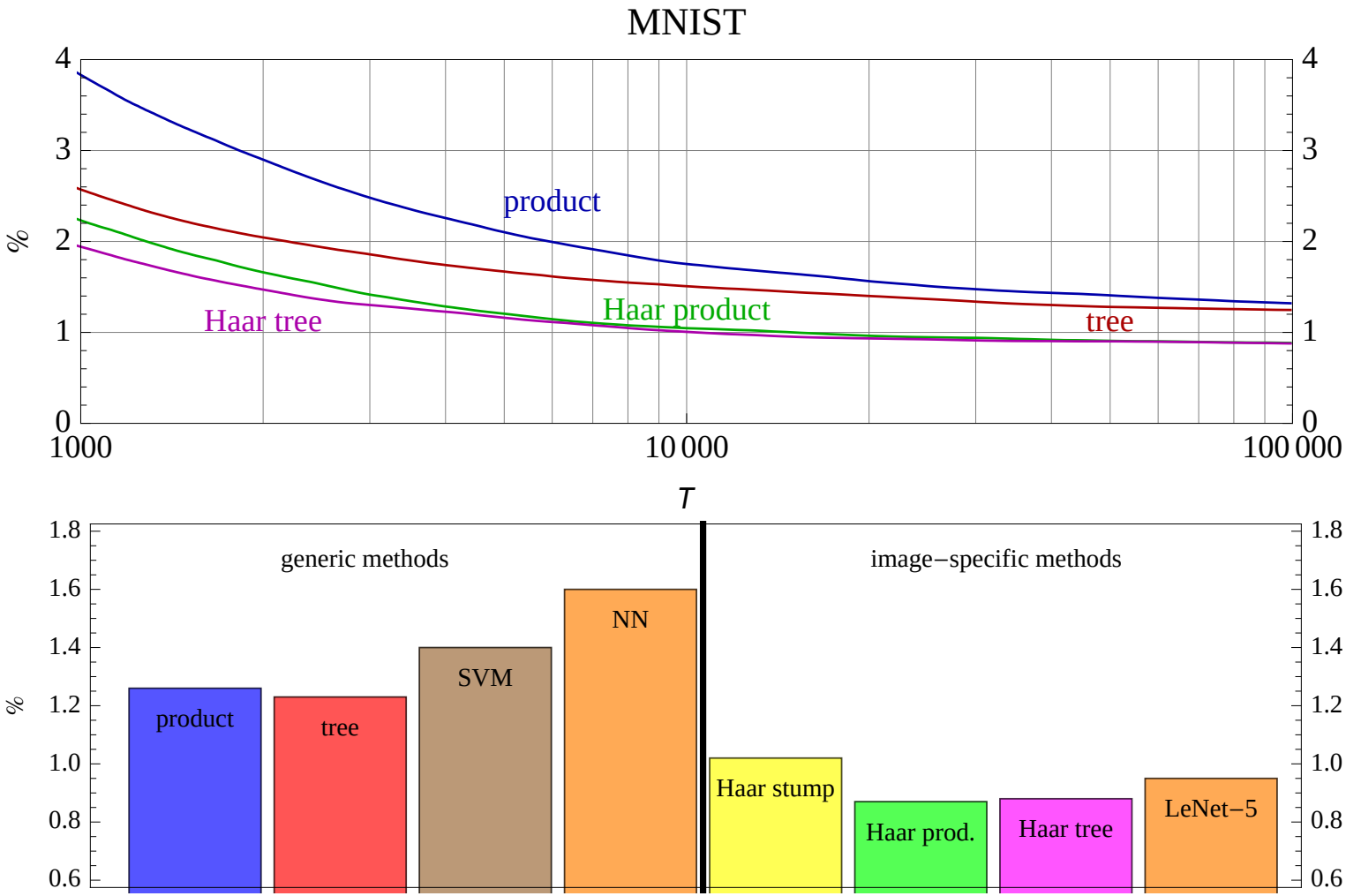
  $$y'_{i,\ell} = h_{1,\ell}(\mathbf{x}_i) \times \ldots \times h_{j-1,\ell}(\mathbf{x}_i) \times h_{j+1,\ell}(\mathbf{x}_i) \times \ldots \times h_{m,\ell}(\mathbf{x}_i) y_{i,\ell}$$

  and calling the base learner
- Initialize $\mathbf{h}_j$ to constant 1, and iterate over j until convergence

Outline

- Binary AdaBoost
  - algorithm
  - convergence
  - binary stumps and trees
- Multi-class / multi-label AdaBoost.MH
  - multi-class / multi-label classification
  - multi-class / multi-label stumps, trees, and products
- Some results

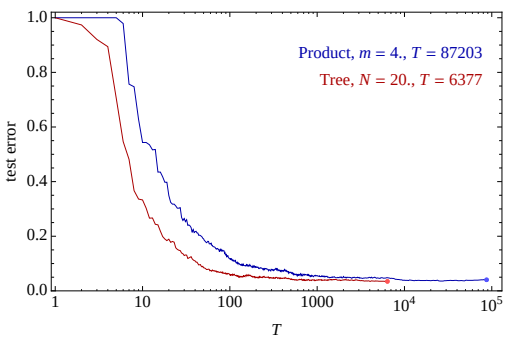
Results on MNIST



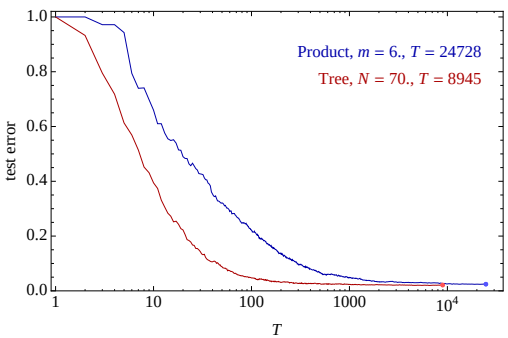
Results on UCI

Method	isolet	letter	optdigits	pendigits	USPS
AdaBoost.MH w Hamming trees	3.5 ± 0.5	2.1 ± 0.2	2.0 ± 0.3	2.1 ± 0.3	4.5 ± 0.5
AdaBoost.MH w Hamming prod. Kégl and Busa-Fekete (2009)	4.2 ± 0.5	2.5 ± 0.2	2.1 ± 0.4	2.1 ± 0.2	4.4 ± 0.5
AOSOLogitBoost $J = 20$, $\nu = 0.1$ Sun *et al.* (2012)	3.5 ± 0.5	2.3 ± 0.2	2.1 ± 0.3	2.4 ± 0.3	4.9 ± 0.5
ABCLogitBoost $J = 20$, $\nu = 0.1$ Li (2009b)	4.2 ± 0.5	2.2 ± 0.2	3.1 ± 0.4	2.9 ± 0.3	4.9 ± 0.5
ABCMart $J = 20$, $\nu = 0.1$ Li (2009a)	5.0 ± 0.6	2.5 ± 0.2	2.6 ± 0.4	3.0 ± 0.3	5.2 ± 0.5
LogitBoost $J = 20$, $\nu = 0.1$ Li (2009b)	4.7 ± 0.5	2.8 ± 0.3	3.6 ± 0.4	3.1 ± 0.3	5.8 ± 0.5
SAMME w single-label trees Zhu *et al.* (2009)		2.3 ± 0.2		2.5 ± 0.3	
AdaBoost.MM Mukherjee and Schapire (2013)		2.5 ± 0.2		2.7 ± 0.3	
AdaBoost.MH w single-label trees Zhu *et al.* (2009)		2.6 ± 0.3		2.8 ± 0.3	
AdaBoost.MH w single-label trees Mukherjee and Schapire (2013)		9.0 ± 0.5		7.0 ± 0.4	

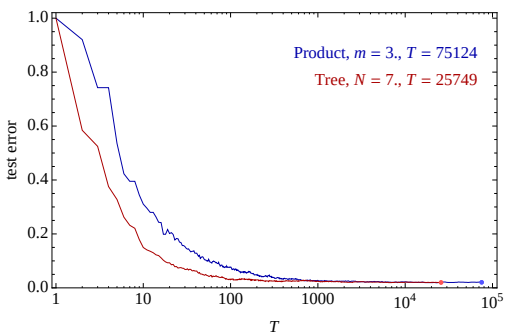
isolet



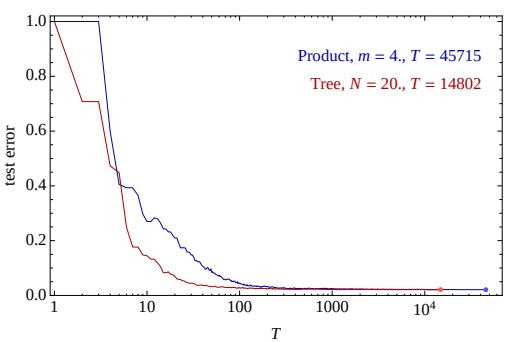
letter



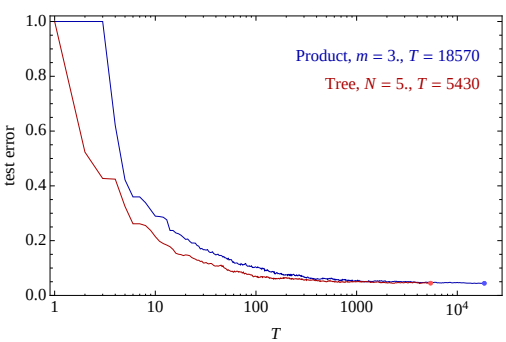
optdigits



pendigits



USPS



Conclusion

- AdaBoost.MH with Hamming trees and products is a state-of-the-art (shallow) multi-class / multi-label learning algorithm

- Everything in this talk is implemented in a turn-key C++ software with a command-line interface (Benbouzid *et al.*, 2012)

`http://multiboost.org`

References

Benbouzid, D., Busa-Fekete, R., Casagrande, N., Collin, F.-D., and Kégl, B. (2012). MultiBoost: a multi-purpose boosting package. *Journal of Machine Learning Research*, **13**, 549–553.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**, 119–139.

Kégl, B. and Busa-Fekete, R. (2009). Boosting products of base classifiers. In *International Conference on Machine Learning*, volume 26, pages 497–504, Montreal, Canada.

Li, P. (2009a). ABC-Boost: Adaptive base class boost for multi-class classification. In *International Conference on Machine Learning*.

Li, P. (2009b). ABC-LogitBoost for multi-class classification. Technical Report arXiv:0908.4144, Arxiv preprint.

Mukherjee, I. and Schapire, R. E. (2013). A theory of multiclass boosting. *Journal of Machine Learning Research*, **14**, 437–497.

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, **37**(3), 297–336.

Sun, P., Reid, M. D., and Zhou, J. (2012). AOSO-LogitBoost: Adaptive one-vs-one LogitBoost for multi-class problem. In *International Conference on Machine Learning (ICML)*.

Zhu, J., Zou, H., Rosset, S., and Hastie, T. (2009). Multi-class AdaBoost. *Statistics and its Interface*, **2**, 349–360.