Boosting products of base classifiers

Balázs Kégl and Róbert Busa-Fekete

University of Paris Sud / CNRS

ICML
June 17, 2009

Outline

- Boosting products of base classifiers
- Products of stumps
- INDICATORBASE: a new base learner for nominal features
- Products of indicators and relation to maximum margin matrix factorization

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

$\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *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$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \text{BASE}(\cdot,\cdot), T)$

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

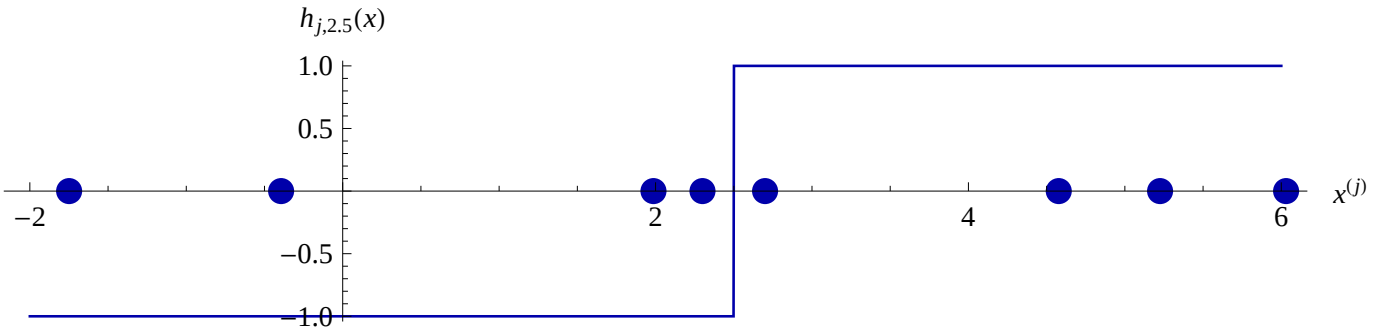
8 $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

9 **else**

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

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

Boosting decision stumps



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

- 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

Products of base learners

$$h(\mathbf{x}) = \prod_{j=1}^{m} h_j(\mathbf{x})$$

- The edge $\gamma = \sum_{i=1}^{n} w_i y_i \prod_{j=1}^{m} h_j(\mathbf{x}_i)$ is multiplicative

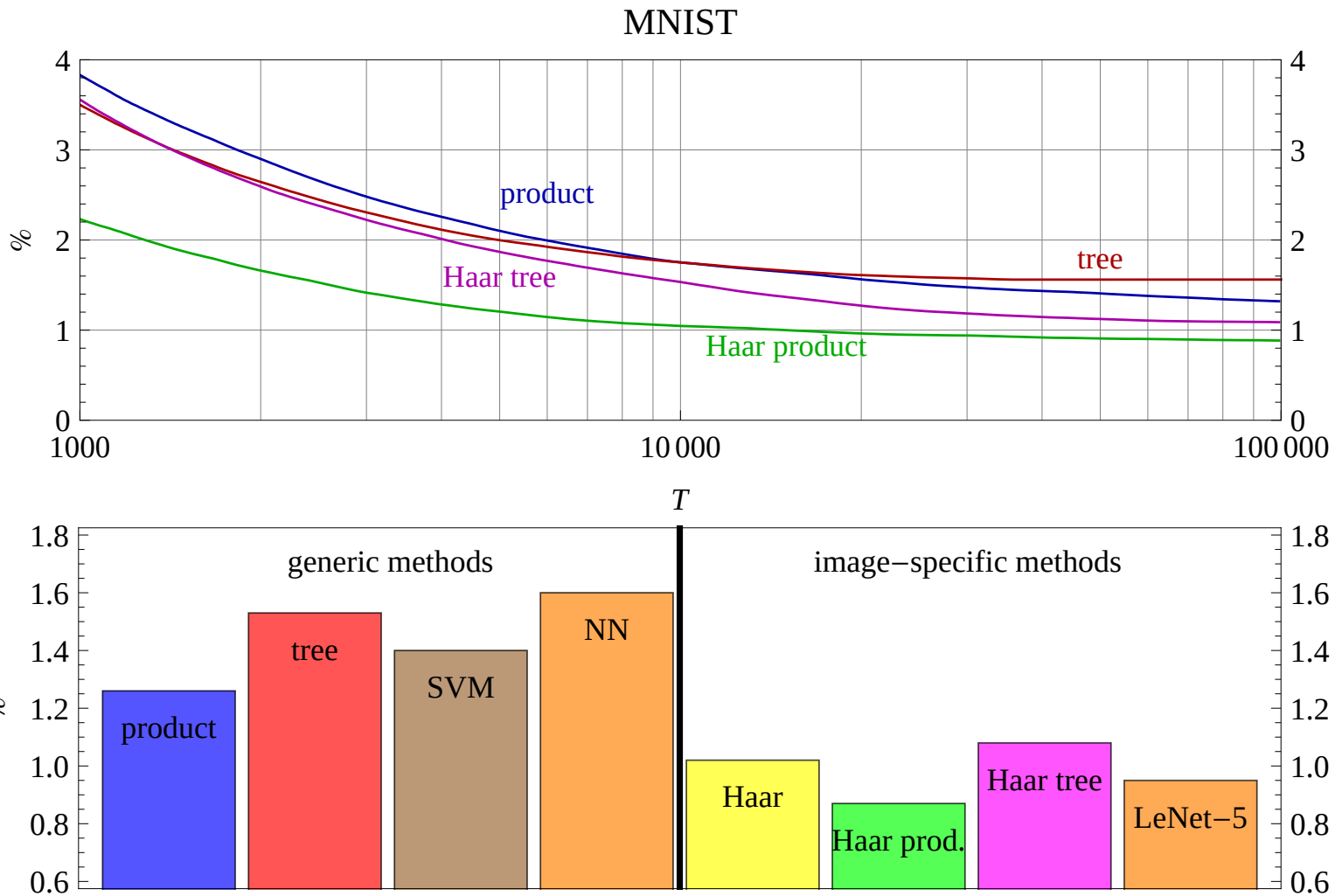
- $h_j(\mathbf{x})$ can be learned by defining the virtual labels

  $$y'_i = h_1(\mathbf{x}_i) \times \ldots \times h_{j-1}(\mathbf{x}_i) \times h_{j+1}(\mathbf{x}_i) \times \ldots \times h_m(\mathbf{x}_i) y_i$$

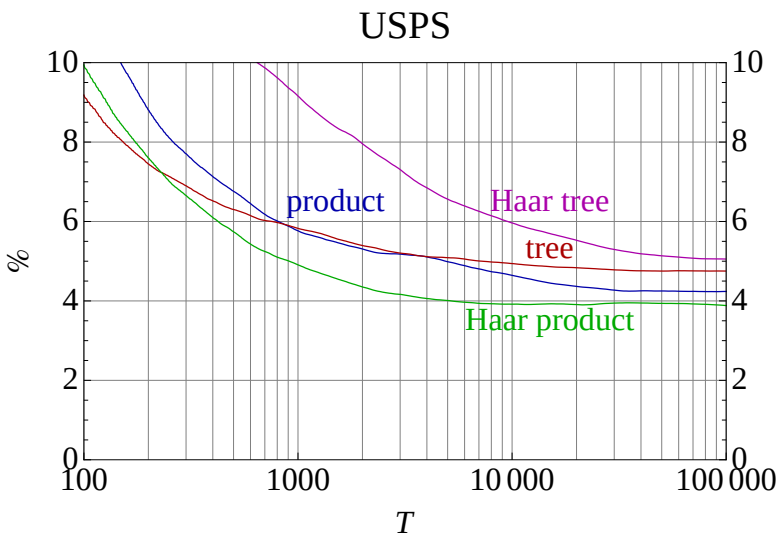
  and calling the base learner

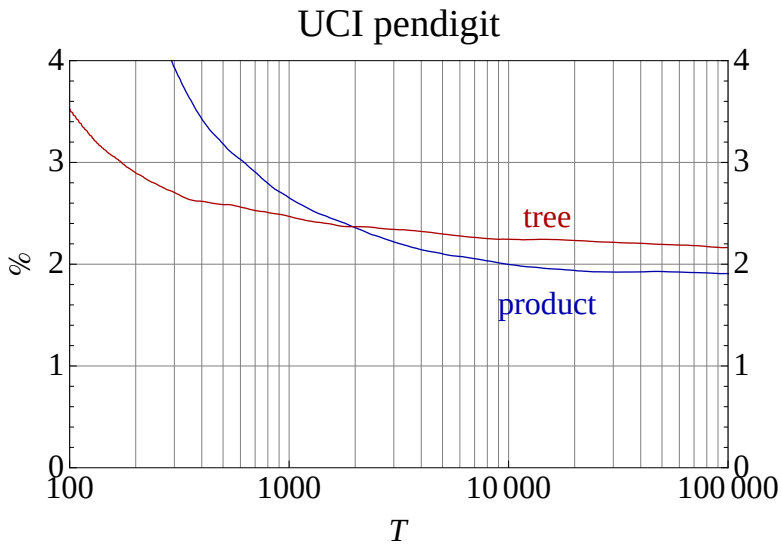
- Initialize h_j to constant 1, and iterate over j until convergence

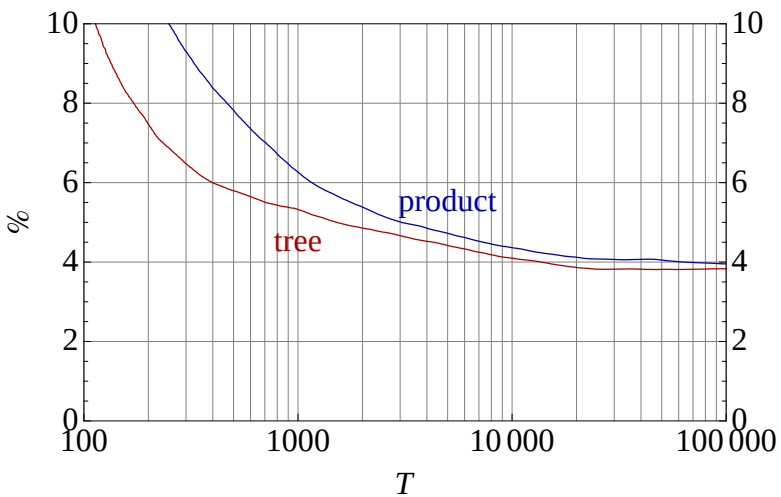
Results with boosting products of stumps

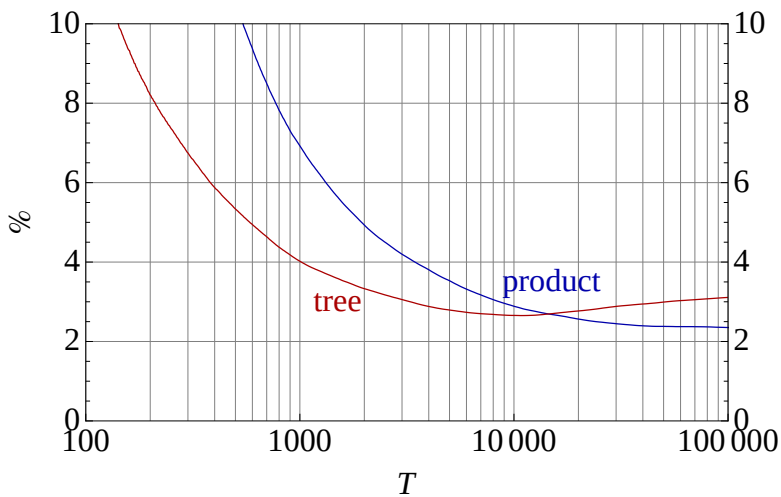


Results with boosting products of stumps









Products of stumps

- Extended to multi-class ADABOOST.MH (see paper)
- Products of stumps are not strong learners (at least, not with using this greedy optimization)
- Usually the (validated) nominal complexity m is much smaller than the number of tree leaves
- The class of linear combination of products of m stumps is a universal approximator if and only if $m \geq d$

Indicator base learner for nominal features

- Nominal features: $x_i^{(j)} \in I^{(j)} = \{1, \ldots, M^{(j)}\} = \{$, , , , $\}$

- SELECTORLEARNER trains classifiers of the form

$$h_{j,\iota}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^{(j)} = \iota, \\ -1 & \text{otherwise.} \end{cases} \qquad h_{j,\square} : \{-, +, -, -, -\}$$

  - Takes $\Theta\left(nd + \sum_j M^{(j)}\right)$ to learn

- INDICATORLEARNER trains classifiers of the form

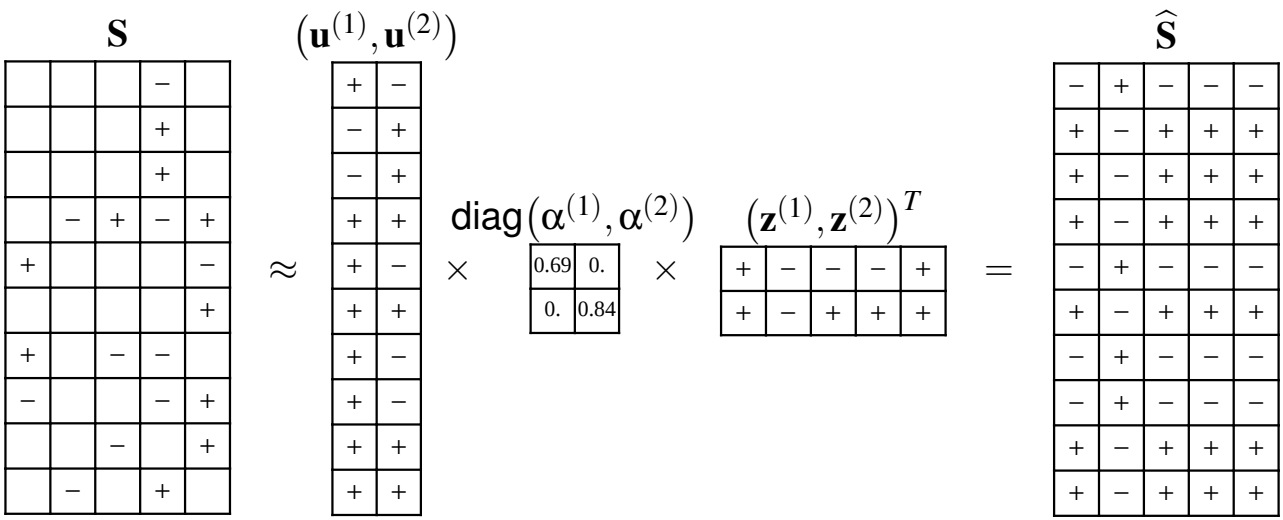
$$h_{j,\mathbf{u}}(\mathbf{x}) = \mathbf{u}_{x^{(j)}} \qquad h_{j,\{+1,-1,-1,+1,+1\}} : \{+, -, -, +, +\}$$

where $\mathbf{u}$ is a $\{-1, 1\}$-vector over $I^{(j)}$

  - Takes $\Theta\left(nd + \sum_j M^{(j)}\right)$ to learn

Products of subset indicators

- Example
  - $d = 2$, $I^{(1)}$: movie IDs, $I^{(2)}$: user IDs
  - products are of the form $\mathbf{h}^{(t)}_{\mathbf{u}^{(t)},\mathbf{z}^{(t)}}\left(x^{(1)},x^{(2)}\right) = \alpha^{(t)} u^{(t)}_{x^{(1)}} z^{(t)}_{x^{(2)}}$



Products of subset indicators

- ADABOOST does maximum margin matrix factorization

SVM	SDP SVM [Srebro et al. 2005]
ADABOOST	ADABOOST with products of subset indicators

- Results on a small subset of MovieLens is comparable to [Srebro et al. 2005] but weak learner must be improved for scaling up to larger sets

Conclusion

- ADABOOST with products of stumps:
  - outperforms boosted trees
  - less prone to overfitting
  - able to improve stumps even in high-dimensional feature spaces

- ADABOOST with products of subset indicators:
  - solves the the MMMF problem with L_1 margin
  - weak learner must be improved for practical performance