TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni
http://www.isti.cnr.it/People/T.Fagni/

Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
Via G Moruzzi, 1 – 56124 Pisa, Italy
E-mail: tiziano.fagni@isti.cnr.it

11/12/2006
Yahoo! Research Barcelona

Joint work with Andrea Esuli and Fabrizio Sebastiani

Outline

What is Hierarchical Text Categorization (HTC)?

- HTC is the task of classifying text documents on a set of topics organized in a tree/DAG-shaped hierarchy.
- Given a set of textual documents T and a predefined set of categories $C = \{c_1, \ldots, c_m\}$, n-of-m text classification is the task of estimating an unknown target function $\Phi : T \times C \rightarrow \{-1, +1\}$, that describes how documents ought to be classified, by means of a classifier $\hat{\Phi} : T \times C \rightarrow \{-1, +1\}$ such that Φ and $\hat{\Phi}$ "coincide as much as possible".

Why is HTC important?

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Introduction

An introduction to boosting and AdaBoost.MH

MP-Boost, an improved boosting algorithm with multiple pivot terms

TreeBoost, our novel hierarchical algorithm

Experiments

Conclusions

- Many text collections are naturally organized as hierarchies of documents (Web repositories, digital libraries, patent repositories, etc.).
- The topological relations between categories can be useful to build more accurate and more efficient classifiers.
- In many classification contexts it is necessary to deal with huge data collections: the hierarchical model is simply a necessity.
- The attention of text classification researchers has mostly been concentrated on "flat" classification.

Our proposal

- We propose TreeBoost[2], a simple and elegant multi-label hierarchical algorithm which recursively uses AdaBoost.MH [4] or MP-Boost [1] as base learner.
- We analyze TreeBoost in terms of both
  - computational cost
  - effectiveness and efficiency as measured on two standard benchmarks.

Introduction to AdaBoost.MH

- Boosting algorithms attempt to generate a highly accurate classifier (final hypothesis) by combining a set of moderately accurate classifiers (weak hypotheses – WHs).
- AdaBoost.MH works by iteratively calling a weak learner to generate a sequence $\hat{\Phi}_1, \ldots, \hat{\Phi}_S$ of WHs; at the end of the iteration the final hypothesis is obtained as $\hat{\Phi} = \sum_{s=1}^{S} \hat{\Phi}_s$.
- The input to AdaBoost.MH is $Tr = \{\langle d_1, C_1 \rangle, \ldots, \langle d_g, C_g \rangle\}$
- A WH is a function $\hat{\Phi}_s : D \times C \rightarrow \mathbf{R}$. We interpret
  - the sign of $\hat{\Phi}_s(d_i, c_j)$ as the prediction of $\hat{\Phi}_s$ on whether d_i belongs to c_j
  - the absolute value of $\hat{\Phi}_s(d_i, c_j)$ as the strength of this belief.

Introduction to AdaBoost.MH (2)

- At each iteration s AdaBoost.MH tests the effectiveness of $\hat{\Phi}_s$ on Tr and uses the results to update a distribution $D_s(d_i, c_j)$ on the training data.
- $D_{s+1}(d_i, c_j)$ is meant to capture how (in)effective $\hat{\Phi}_1, \ldots, \hat{\Phi}_s$ have been in correctly predicting whether the training document d_i belongs to c_j or not. By passing D_s to the weak learner, AdaBoost.MH forces it to generate a new WH $\hat{\Phi}_{s+1}$ that concentrates on the pairs with the highest weight.
- D_1 is uniform. At each iteration s the weights $D_s(d_i, c_j)$ are updated to $D_{s+1}(d_i, c_j)$ according to the rule

$$D_{s+1}(d_i, c_j) = \frac{D_s(d_i, c_j)\exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}_s(d_i, c_j))}{Z_s} \quad (1)$$

The AdaBoost.MH algorithm

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Introduction

An introduction to boosting and AdaBoost.MH

Description

Choosing the WHs

MP-Boost, an improved boosting algorithm with multiple pivot terms

TreeBoost, our novel hierarchical algorithm

Experiments

Conclusions

I: A training set $Tr = \{\langle d_1, C_1\rangle, \ldots, \langle d_g, C_g\rangle\}$ where $C_i \subseteq C = \{c_1, \ldots, c_m\}$ for all $i = 1, \ldots, g$.

B: Let $D_1(d_i, c_j) = \frac{1}{gm}$ for all $i = 1, \ldots, g$ and for all $j = 1, \ldots, m$

For $s = 1, \ldots, S$ do:

- pass distribution $D_s(d_i, c_j)$ to the weak learner;
- get the WH $\hat{\Phi}_s$ from the weak learner;
- set $D_{s+1}(d_i, c_j) = \frac{D_s(d_i, c_j)\exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}_s(d_i, c_j))}{Z_s}$

  where $Z_s = \sum_{i=1}^{g}\sum_{j=1}^{m} D_s(d_i, c_j)\exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}_s(d_i, c_j))$

  is a normalization factor chosen so that $\sum_{i=1}^{g}\sum_{j=1}^{m} D_{s+1}(d_i, c_j) = 1$

O: A final hypothesis $\hat{\Phi}(d, c) = \sum_{s=1}^{S} \hat{\Phi}_s(d, c)$

AdaBoost.MH: the form used to represent WHs

- Each document d_i is represented as a vector $\langle w_{1i}, \ldots, w_{ri} \rangle$ of r binary weights.
- The WHs generated at iteration s are decision stumps of the form

$$\hat{\Phi}_s(d_i, c_j) = \begin{cases} a_{0j} & \text{if } w_{ki} = 0 \\ a_{1j} & \text{if } w_{ki} = 1 \end{cases} \quad (2)$$

  where t_k (called the pivot term of $\hat{\Phi}_s$) belongs to $\{t_1, \ldots, t_r\}$, and a_{0j} and a_{1j} are real-valued constants.
- A WK thus consists in one pivot term (t_k) and $2m$ constants (the a_{0j}'s and a_{1j}'s), different for each iteration s.

AdaBoost.MH: how to choose WHs

- It may be proven that a reasonable (although suboptimal) way to maximize the effectiveness of $\hat{\Phi}$ is to choose each $\hat{\Phi}_s$ in such a way as to minimize Z_s.
- AdaBoost.MH chooses WHs of the form described in Equation 2 by the following algorithm.
  1. For each term $t_k \in \{t_1, \ldots, t_r\}$, select, among all the WHs $\hat{\Phi}$ that have t_k as the "pivot term", the one (indicated by $\hat{\Phi}_{best(k)}$) for which Z_s is minimum.
  2. Among all the WHs $\hat{\Phi}_{best(1)}, \ldots, \hat{\Phi}_{best(r)}$ selected for the r different terms in Step 1, select the one (indicated by $\hat{\Phi}_s$) for which Z_s is minimum.

AdaBoost.MH: how to choose WHs (2)

- Step 1 is clearly the key step, since there are a non-enumerable set of WHs with t_k as pivot. It has been proven that, given t_k and c_j,

$$\hat{\Phi}_{best(k)}(d_i, c_j) = \begin{cases} \frac{1}{2}\ln\frac{W_{+1}^{0jk}}{W_{-1}^{0jk}} & \text{if } w_{ki} = 0 \\ \frac{1}{2}\ln\frac{W_{+1}^{1jk}}{W_{-1}^{1jk}} & \text{if } w_{ki} = 1 \end{cases} \quad (3)$$

where

$$W_b^{xjk} = \sum_{i=1}^{g} D_s(d_i, c_j) \cdot [\![w_{ki} = x]\!] \cdot [\![\Phi(d_i, c_j) = b]\!] \quad (4)$$

for $b \in \{-1, +1\}$, $x \in \{0, 1\}$, $j \in \{1, \ldots, m\}$ and $k \in \{1, \ldots, r\}$, and where $[\![\pi]\!]$ indicates the characteristic function of predicate π (i.e. the function that returns 1 if π is true and 0 otherwise). For term t_k and for these values of a_{xj} we obtain

$$Z_s = 2\sum_{j=1}^{m}\sum_{x=0}^{1}(W_{+1}^{xjk} W_{-1}^{xjk})^{\frac{1}{2}} \quad (5)$$

MP-Boost: the motivations

- In AdaBoost.MH at each iteration s a pivot term t_k, the same for all categories, is chosen. This is suboptimal, since t_k may be a good discriminator for c' but not for c''.
- We claim that choosing, at each iteration s, a different pivot $t_{\langle s,j \rangle}$ for each c_j allows the WH to provide customized treatment to each c_j.
- AdaBoost.MH with multiple pivot terms (MP-Boost) basically consists in modifying the form of WHs, which become

$$\hat{\Phi}_s(d_i, c_j) = \begin{cases} a_{0j} & \text{if } w_{\langle s,j \rangle i} = 0 \\ a_{1j} & \text{if } w_{\langle s,j \rangle i} = 1 \end{cases} \quad (6)$$

where $t_{\langle s,j \rangle}$ is the pivot term chosen for c_j at iteration s.

MP-Boost: how to choose WHs

- Let us define a weak c_j-hypothesis as a function

$$\hat{\Phi}^j(d_i) = \begin{cases} a_{0j} & \text{if } w_{ki} = 0 \\ a_{1j} & \text{if } w_{ki} = 1 \end{cases} \quad (7)$$

  that is only concerned with c_j; a WH is thus the union of weak c_j-hypotheses, one for each $c_j \in C$.
- At each iteration s MP-Boost chooses a WH $\hat{\Phi}_s$ as follows:
  1. For each c_j and for each t_k select, among all weak c_j-hypothesis $\hat{\Phi}^j$ with t_k as the pivot, the one ($\hat{\Phi}^j_{best(k)}$) which minimizes

$$Z^j_s = \sum_{i=1}^{g} D_s(d_i, c_j) \exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}^j(d_i)) \quad (8)$$

  2. For each c_j, among all the hypotheses $\hat{\Phi}^j_{best(1)}, \ldots, \hat{\Phi}^j_{best(r)}$ selected in Step 1 for the r different terms, select the one ($\hat{\Phi}^j_s$) for which Z^j_s is minimum;
  3. Choose, as the WH $\hat{\Phi}_s$, the "union", across all $c_j \in C$, of the weak c_j-hypotheses selected in Step 2, i.e. the function such that $\hat{\Phi}_s(d_i, c_j) = \hat{\Phi}^j_s(d_i)$.

MP-Boost: observations

- Step 2 is such that weak c_j-hypotheses based on different pivots may be chosen for different c_j's.
- $\hat{\Phi}^j_{best(k)}$ has still the form described in Equation 3, since the WH generated by Equation 6 is the same that Equation 2 generates when $m = 1$.
- The "outer" AdaBoost.MH algorithm is untouched, except that a Z^j_s local to each c_j is used.

The notation

Let:

- H be a tree-structured set of categories
- r be its root category
- $L = \langle\langle l_1, Tr^+(l_1)\rangle, \ldots, \langle l_m, Tr^+(l_m)\rangle\rangle$ be the set of leaf categories together with their set of positive training documents

Symbol	Meaning
$\uparrow(c_j)$	the parent category of c_j
$\downarrow(c_j)$	the set of children categories of c_j
$\Uparrow(c_j)$	the set of ancestor categories of c_j
$\Downarrow(c_j)$	the set of descendant categories of c_j
$\leftrightarrow(c_j)$	the set of sibling categories of c_j

Definitions used in the algorithm

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Introduction

An introduction to boosting and AdaBoost.MH

MP-Boost, an improved boosting algorithm with multiple pivot terms

TreeBoost, our novel hierarchical algorithm

Assumptions

Description of the algorithm

The computational cost of the algorithm

Experiments

Assumptions:

- Each document can belong to zero, one or several categories (n-of-m classification).
- Each document can only belong to leaf categories.

$Tr^+(c_j)$ Given a non-leaf category c_j, its set of positive training examples $Tr^+(c_j)$ is defined as

$$Tr^+(c_j) = \bigcup_{c \in \Downarrow(c_j)} Tr^+(c) \tag{9}$$

$Tr^-(c_j)$ Given a non-leaf category c_j, its set of negative training examples $Tr^-(c_j)$ is defined as

$$Tr^-(c_j) = \left(\bigcup_{c \in \leftrightarrow(c_j)} Tr^+(c) \right) - Tr^+(c_j) \tag{10}$$

The intuitions behind the algorithm

There are several intuitions which motivate TreeBoost, some of them already present in the HTC literature:

- The classification of a document d_i is to be seen as a descent through the hierarchy, from the root to the leaf categories where d_i is deemed to belong.
- The training of $\hat{\Phi}_j$ should be performed "locally" by applying the "quasi-positive" criterion in the choice of negative training examples.
- Feature selection should be performed locally.
- Distribution update, a step typical of the boosting algorithm, should be performed locally.

TreeBoost: the algorithm

Input parameters

- H is a tree-structured set of categories.
- r is the root category of H.
- L is the set of leaf categories together with their set of positive training documents.

procedure TreeBoost(H, r, L)
begin
    if *not (r is a leaf category)* **then**
        $Tr_{\downarrow(r)}$ = identifyPositivesAndNegatives($\downarrow(r), L$);
        applyFeatureReduction($Tr_{\downarrow(r)}$);
        runBoostingAlgorithm($Tr_{\downarrow(r)}$);
        foreach *child in* $\downarrow(r)$ **do**
            H_{child} =subtree(H,*child*);
            L_{child} = selectLeafsAndPositives(H_{child});
            TreeBoost(H_{child}, child, L_{child});
        end
    else
        do nothing;
    end
end

TreeBoost: the algorithm

Input parameters

- H is a tree-structured set of categories.
- r is the root category of H.
- L is the set of leaf categories together with their set of positive training documents.

procedure TreeBoost(H, r, L)
begin
    if *not (r is a leaf category)* **then**
        $Tr_{\downarrow(r)}$ = `identifyPositivesAndNegatives`($\downarrow(r), L$);
        `applyFeatureReduction`($Tr_{\downarrow(r)}$);
        `runBoostingAlgorithm`($Tr_{\downarrow(r)}$);
        foreach *child in* $\downarrow(r)$ **do**
            H_{child} =`subtree`(H,*child*);
            L_{child} = `selectLeafsAndPositives`(H_{child});
            TreeBoost(H_{child}, child, L_{child});
        end
    else
        do nothing;
    end
end

TreeBoost: the algorithm

Input parameters

- H is a tree-structured set of categories.
- r is the root category of H.
- L is the set of leaf categories together with their set of positive training documents.

procedure TreeBoost(H, r, L)
begin
    if *not (r is a leaf category)* **then**
        $Tr_{\downarrow(r)}$ = `identifyPositivesAndNegatives`($\downarrow(r), L$);
        `applyFeatureReduction`($Tr_{\downarrow(r)}$);
        `runBoostingAlgorithm`($Tr_{\downarrow(r)}$);
        foreach *child in* $\downarrow(r)$ **do**
            H_{child} =`subtree`(H,*child*);
            L_{child} = `selectLeafsAndPositives`(H_{child});
            TreeBoost(H_{child}, child, L_{child});
        end
    else
        do nothing;
    end
end

TreeBoost: the algorithm

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Input parameters

- H is a tree-structured set of categories.
- r is the root category of H.
- L is the set of leaf categories together with their set of positive training documents.

procedure TreeBoost(H, r, L)
begin
  if *not (r is a leaf category)* **then**
    $Tr_{\downarrow(r)}$ = identifyPositivesAndNegatives($\downarrow(r)$, L);
    applyFeatureReduction($Tr_{\downarrow(r)}$);
    runBoostingAlgorithm($Tr_{\downarrow(r)}$);
    foreach *child in* $\downarrow(r)$ **do**
      H_{child} =subtree(H,$child$);
      L_{child} = selectLeafsAndPositives(H_{child});
      TreeBoost(H_{child}, child, L_{child});
    end
  else
    do nothing;
  end
end

TreeBoost: the algorithm

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Introduction
An introduction to boosting and AdaBoost.MH
MP-Boost, an improved boosting algorithm with multiple pivot terms
TreeBoost, our novel hierarchical algorithm
Assumptions
Description of the algorithm
The computational cost of the algorithm
Experiments

Input parameters

- H is a tree-structured set of categories.
- r is the root category of H.
- L is the set of leaf categories together with their set of positive training documents.

procedure TreeBoost(H, r, L)
begin
  if *not (r is a leaf category)* **then**
    $Tr_{\downarrow(r)}$ = identifyPositivesAndNegatives($\downarrow(r)$, L);
    applyFeatureReduction($Tr_{\downarrow(r)}$);
    runBoostingAlgorithm($Tr_{\downarrow(r)}$);
    foreach *child in* $\downarrow(r)$ **do**
      H_{child} =subtree(H,$child$);
      L_{child} = selectLeafsAndPositives(H_{child});
      TreeBoost(H_{child}, child, L_{child});
    end
  else
    do nothing;
  end
end

The computational cost of TreeBoost: training time

Notation

- g The number of training documents.
- m The number of categories.
- r The dimension of the vector space.
- a The ariety of a perfectly balanced category tree.
- h The height of a perfectly balanced category tree.

- The computational cost of AdaBoost.MH and MP-Boost is $O(gmr)$.
- The computational cost of TreeBoost is $O(gmr)$ in the worst case and $O(gahr)$ in the best case.
- Given $m = a^h$, this means TreeBoost is cheaper than baselines by an exponential factor.

The computational cost of TreeBoost: testing time

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Introduction

An introduction to boosting and AdaBoost.MH

MP-Boost, an improved boosting algorithm with multiple pivot terms

TreeBoost, our novel hierarchical algorithm

Assumptions

Description of the algorithm

The computational cost of the algorithm

Experiments

Notation

- S The number of boosting iterations.
- m The number of categories.
- a The ariety of a perfectly balanced category tree.
- h The height of a perfectly balanced category tree.

- The computational cost of AdaBoost.MH and MP-Boost is $O(Sm)$.
- The computational cost of TreeBoost is $O(Sm)$ in the worst case and $O(Sah)$ in the best case.
- Recalling that $m = a^h$, we can see that TreeBoost is cheaper than baselines by an exponential factor.

Experimental setting

- Preprocessing: stop word removal + punctuation removal + downcasing + number removal + (Porter) stemming.
- Feature selection (*aka* term space reduction, TSR)
  - It is performed by scoring features by means of

$$IG(t_k, c_i) = \sum_{c \in \{c_i, \bar{c}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t, c) \cdot \log \frac{P(t, c)}{P(t) \cdot P(c)}$$

  - The final set of features has been chosen according to Forman's *round robin* technique (i.e. picking, for each c_i, the v features with the highest $IG(t_k, c_i)$, and pooling them together).
- Tests are run (i) with a 0% reduction factor, and (ii) with values of v bringing about a 90% reduction factor.
- We use 2 types of reduced feature sets: "globalTSR" and "glocalTSR".

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Introduction

An introduction to boosting and AdaBoost.MH

MP-Boost, an improved boosting algorithm with multiple pivot terms

TreeBoost, our novel hierarchical algorithm

Experiments

Experimental setting

Results

Conclusions

Evaluation measures

We define precision (π) as $\pi = \frac{TP}{TP+FP}$ and recall (ρ) as $\rho = \frac{TP}{TP+FN}$. As measure which combines π and ρ we use F_1, defined as

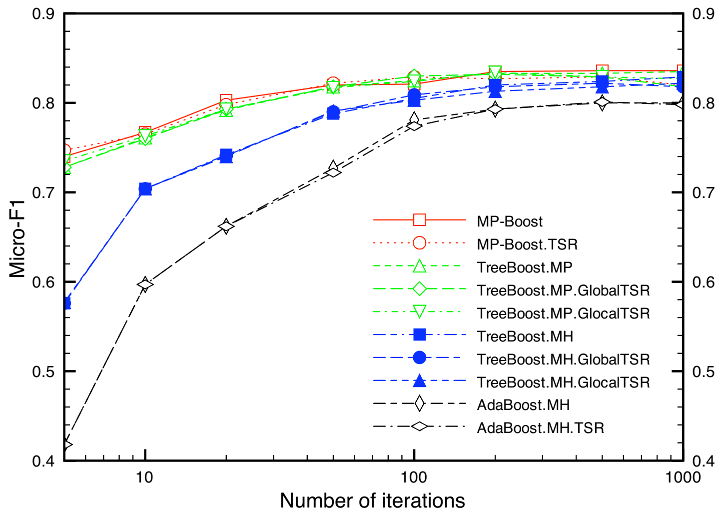
$$F_1 = \frac{2\pi\rho}{\pi + \rho} = \frac{2\,TP}{2\,TP + FP + FN} \quad (11)$$

- microaveraged F_1 (F_1^{μ}), obtained by (i) computing the category-specific values TP_i, (ii) obtaining TP as the sum of the TP_i's (same for FP and FN), and then (iii) applying Equation 11.
- macroaveraged F_1 (F_1^{M}), obtained by first computing the F_1 values specific to the individual categories, and then averaging them across the c_i's.

We have used two corpora of preclassified documents:

Corpus	# of training docs	# of test docs	# of cats	# of unique terms
Hier. Reuters-21578[5]	7,770	3,299	94	17,804
RCV1-v2[3]	23,149	781,265	101	55,051

TreeBoost: microaveraged-F1 on Reuters21578



TreeBoost: macroaveraged-F1 on Reuters21578

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Introduction

An introduction to boosting and AdaBoost.MH

MP-Boost, an improved boosting algorithm with multiple pivot terms

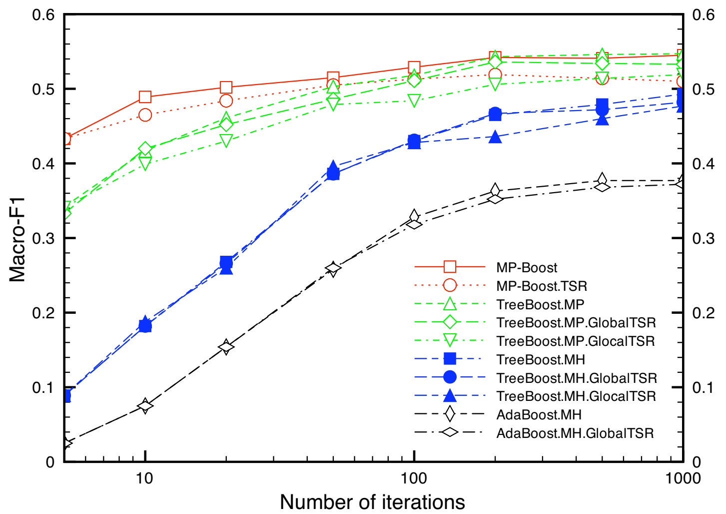
TreeBoost, our novel hierarchical algorithm

Experiments

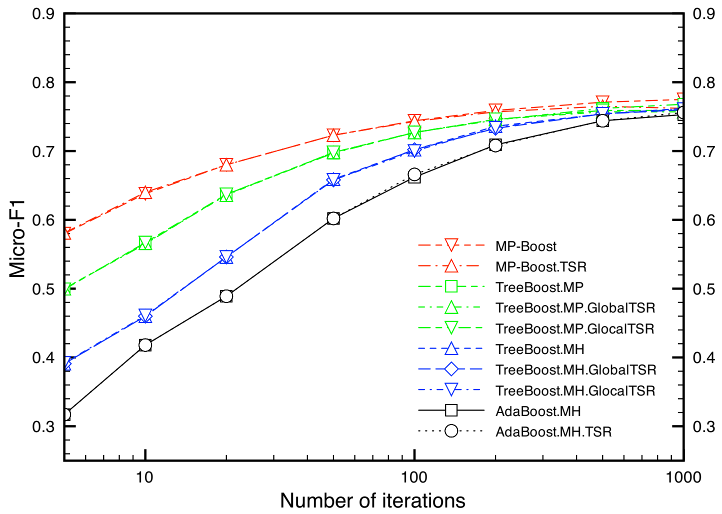
Experimental setting

Results

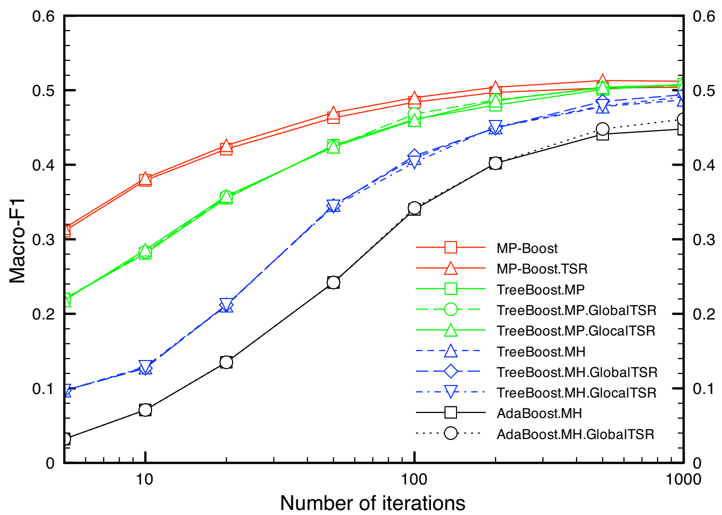
Conclusions



Microaveraged-$F1$ on RCV1-v2



Macroaveraged-$F1$ on RCV1-v2



TreeBoost.MH: effectiveness considerations

	TreeBoost.MH.GlocalTSR		TreeBoost.MH.GlobalTSR	
S	F_1^μ	F_1^M	F_1^μ	F_1^M
5	+38.00	+288.00	+37.80	+256.00
10	+18.00	+150.00	+18.00	+142.00
20	+11.80	+68.80	+11.90	+72.70
50	+9.60	+52.30	+9.40	+48.50
100	+3.70	+34.60	+4.60	+35.50
200	+2.50	+23.90	+3.20	+32.70
500	+2.10	+25.00	+2.60	+28.30
1000	+3.00	+28.20	+2.50	+29.60

- Switching from AdaBoost.MH to TreeBoost.MH brings about substantial improvements for both F_1^μ and F_1^M.
- F_1^M improves much more than F_1^μ, meaning that TreeBoost.MH is suitable to classification problem with skewed category distributions.
- TreeBoost.MH converges to optimum performance more rapidly than AdaBoost.MH.
- There is no relevant performance degradation when using reduced feature sets compared to full feature set.

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Introduction

An introduction to boosting and AdaBoost.MH

MP-Boost, an improved boosting algorithm with multiple pivot terms

TreeBoost, our novel hierarchical algorithm

Experiments

Experimental setting

Results

Conclusions

TreeBoost.MP: effectiveness considerations

	TreeBoost.MP.GlocalTSR		TreeBoost.MP.GlobalTSR	
S	F_1^μ	F_1^M	F_1^μ	F_1^M
5	-1.60	-21.00	-2.50	-23.10
10	-0.30	-14.20	-0.70	-9.70
20	-0.60	-11.20	-0.60	-6.60
50	-0.60	-5.10	-0.50	-3.80
100	-0.60	-5.70	+0.10	-0.40
200	+0.80	-2.50	+0.60	+3.30
500	+0.00	+0.00	+0.00	+3.90
1000	+0.90	+1.80	+0.10	+4.50

- TreeBoost has practically no effectiveness degradation even when comparing results with a strong baseline like MP-Boost.
- There is no relevant performance degradation when using reduced feature sets compared to full feature set.

TreeBoost: efficiency considerations

	TreeBoost.MH.GlobalTSR		TreeBoost.MP.GlobalTSR	
S	$\tau(Tr)$	$\tau(Te)$	$\tau(Tr)$	$\tau(Te)$
5	-60.70	+10.20	- 69.80	+11.40
10	-60.70	+3.50	-69.80	+5.30
20	-60.70	-10.40	-69.80	-6.30
50	-60.70	-33.70	-69.80	-19.80
100	-60.70	-42.70	-69.80	-28.40
200	-60.70	-54.80	-69.80	-32.50
500	-60.70	-64.60	-69.80	-34.30
1000	-60.70	-66.00	-69.80	-34.50

- The training time is smaller than with the baselines, thus confirming the computational cost analysis.
- TreeBoost diminishes significantly the classification time. This improvement is more marked when the number of iterations increase.

Comparison with Hoffman et al.[5] results

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Introduction

An introduction to boosting and AdaBoost.MH

MP-Boost, an improved boosting algorithm with multiple pivot terms

TreeBoost, our novel hierarchical algorithm

Experiments

Experimental setting

Results

Conclusions

Method	π^μ	ρ^μ	F_1^μ	π^M	ρ^M	F_1^M
Hierarchical Mixture*	.872	.771	.818	.815	.408	.459
Hierarchical Shrinkage*	.849	.780	.813	.765	.392	.432
TreeBoost.MH	.798	.898	.845	.419	.890	.478
TreeBoost.MP	.840	.886	**.862**	.499	.795	**.525**

Table: Results for hierarchical algorithms

Method	π^μ	ρ^μ	F_1^μ	π^M	ρ^M	F_1^M
SVM*	.881	.868	**.875**	.837	.524	**.554**
Knn*	.790	.813	.802	.759	.503	.491
NaiveBayes*	.825	.713	.765	.763	.283	.337
MP-Boost	.884	.843	.863	.796	.537	.548

Table: Results for flat algorithms

* Original paper results

In conclusion

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

- We have proposed TreeBoost, a multi-label HTC algorithm based on a hierarchical variant of AdaBoost.MH and MP-Boost.
- We have analyzed the computational cost of the algorithm showing that TreeBoost is cheaper to train and test by an exponential factor than the baselines.
- The results show that:
  - TreeBoost.MH outperforms AdaBoost.MH in terms of effectiveness, especially for F_1^M;
  - TreeBoost.MP has very similar effectiveness performance respect to MP-Boost;
  - TreeBoost.MP is better than TreeBoost.MH;
  - TreeBoost outperforms baselines in terms of efficiency by heavily diminishing training and testing time;
  - We can use a highly reduced feature set without bringing about any effectiveness degradation.

Bibliography

TreeBoost: A Boosting Algorithm for Multi-label Hierarchical Text Categorization

Tiziano Fagni

Introduction
An introduction to boosting and AdaBoost.MH
MP-Boost, an improved boosting algorithm with multiple pivot terms
TreeBoost, our novel hierarchical algorithm
Experiments
Conclusions
Bibliography

A. Esuli, T. Fagni, and F. Sebastiani.
Mp-boost: A multiple-pivot boosting algorithm and its application to text categorization.
In *Proceedings of the 13th International Symposium on String Processing and Information Retrieval (SPIRE'06), Glasgow, UK*, pages 1–12, 2006.

A. Esuli, T. Fagni, and F. Sebastiani.
Treeboost.mh: A boosting algorithm for multi-label hierarchical text categorization.
In *Proceedings of the 13th International Symposium on String Processing and Information Retrieval (SPIRE'06), Glasgow, UK*, pages 13–24, 2006.

D. D. Lewis, F. Li, T. Rose, and Y. Yang.
RCV1: A new benchmark collection for text categorization research.
Journal of Machine Learning Research, 5:361–397, April 2004.

R. E. Schapire and Y. Singer.
BoosTexter: a boosting-based system for text categorization.
Machine Learning, 39(2/3):135–168, 2000.

K. Toutanova, F. Chen, K. Popat, and T. Hofmann.
Text classification in a hierarchical mixture model for small training sets.
In *Proceedings of the 10th ACM International Conference on Information and Knowledge Management (CIKM'01)*, pages 105–113, Atlanta, US, 2001.

ANY QUESTIONS?

Appendix

7. Optimizing AdaBoost.MH implementation
8. Details about computational cost of TreeBoost.MH

Optimized AdaBoost.MH implementation

- Note that if $\{\hat{\Phi}_1, \dots, \hat{\Phi}_S\}$ contains a subset $\{\hat{\Phi}_1^{(k)}, \dots, \hat{\Phi}_{q(k)}^{(k)}\}$ of WHs that all hinge on t_k and are of the form

$$\hat{\Phi}_r^{(k)}(d_i, c_j) = \begin{cases} a_{0j}^r & \text{if } w_{ki} = 0 \\ a_{1j}^r & \text{if } w_{ki} = 1 \end{cases} \quad (12)$$

  for $r = 1, \dots, q(k)$, the collective contribution of $\hat{\Phi}_1^{(k)}, \dots, \hat{\Phi}_{q(k)}^{(k)}$ to the final hypothesis is the same as that of a "combined hypothesis"

$$\hat{\Phi}^{(k)}(d_i, c_j) = \begin{cases} \sum_{r=1}^{q(k)} a_{0j}^r & \text{if } w_{ki} = 0 \\ \sum_{r=1}^{q(k)} a_{1j}^r & \text{if } w_{ki} = 1 \end{cases} \quad (13)$$

- In the implementation we have thus replaced $\sum_{s=1}^{S} \hat{\Phi}_s(d_i, c_j)$ with $\sum_{k=1}^{\Delta} \hat{\Phi}^{(k)}(d_i, c_j)$, where Δ is the number of different terms that act as pivot for the WHs in $\{\hat{\Phi}_1, \dots, \hat{\Phi}_S\}$.

The computational cost of the training time

Notation

- g The number of training documents.
- m The number of categories.
- r The number of unique terms in training documents.

AdaBoost.MH

The key steps in computing weak learner require $O(gmr)$:

1. computing, for each $t_k \in T$, the $Z_s(\hat{\Phi}_{best(k)})$ factor (requires $O(gm)$);
2. computing the minimum, over all t_k, of such $Z_s(\hat{\Phi}_{best(k)})$ factors (requires $O(r)$).

The computational cost of the training time (2)

TreeBoost.MH

The computation cost heavily depends on the topology of the categories tree and on the distribution of positive training examples across leaves of the tree. We can thus identify "worst case" and "best case".

worst case: The categories tree has height 1, only leaf categories and no internal category nodes. The cost is equivalent to AdaBoost.MH algorithm.

best case: The categories tree is a perfectly balanced tree of ariety a and height $h = \log_a m$. At each level $l = 1, \ldots, h$, TreeBoost.MH calls AdaBoost.MH a^{l-1} times. This implies:

- each call to AdaBoost.MH requires $O(\frac{g}{a^{l-1}} ar)$ instead of $O(gmr)$;
- the number of operations required by TreeBoost.MH is $O(\sum_{l=1}^{h} a^{l-1} \cdot \frac{g}{a^{l-1}} ar) = O(\sum_{l=1}^{h} gar) = O(garh)$;
- Given $m = a^h$, this means TreeBoost.MH is cheaper than AdaBoost.MH by an exponential factor.

The computational cost of the testing time

AdaBoost.MH

The cost is $O(Sm)$. Each test document is given as input to $O(S)$ weak hypothesis, each of which performs 1 test and m additions.

TreeBoost.MH

worst case: The cost is $O(Sm)$, the same computational cost as AdaBoost.MH.

best case: The cost is $O(Sah)$. Each test document is input to h classifiers, each of which performs 1 test and a additions. Recalling that $m = a^h$, we can see that TreeBoost.MH is cheaper than AdaBoost.MH by an exponential factor.

MP-Boost vs. SVM

At the moment SVM has probably slightly better effectiveness than MP-Boost. The main limits/problems of MP-Boost are:

- it uses a document binary representation;
- it is sensible to outliers;
- it has no internal mechanisms to tune relative importance of errors (FPs vs. FNs).

MP-Boost is less expensive than SVM in terms of computational cost:

- Training time: MP-Boost is $O(gr)$ while SVM is $O(gn_{sv} + n_{sv}^3)$. Recently Steinwart has shown that n_{sv} grows as a linear function of g.
- Testing time: MP-Boost is $O(S)$ while SVM is $O(n_{sv})$.