Flatten Hierarchies for Large-scale Hierarchical Text Categorization*

Xiao-Lin Wang[1], Bao-Liang Lu[1,2]
[1]Center for Brain-Like Computing and Machine Intelligence
Department of Computer Science and Engineering
[2]MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems
Shanghai Jiao Tong University, China
bllu@sjtu.edu.cn

Abstract

Hierarchies are very popular in organizing documents and web pages, hence automated hierarchical classification techniques are desired. However, the current dominant hierarchical approach of top-down method suffers accuracy decrease compared with flat classification approaches, because of error propagation and bottom nodes' data sparsity. In this paper we flatten hierarchies to relieve such accuracy decrease in top-down method, which aims to make hierarchies both effective enough to make large-scale classification tasks feasible, and simple enough to ensure high classification accuracy. We propose two flattening strategies based on these two causes of the accuracy decrease, and experimental results show that the flattening strategy designed for error propagation is more effective, which suggests that hierarchies with lots of branches at top layers can provide high classification accuracy. Besides, we analyze the computational complexity before and after flattening, which approximately agree with the experimental results.

1 Introduction

Hierarchies are widely used in most real-world datasets, as they are natural ways to organize and classify objects. The newswire article corpus of Reuters Corpus Volume 1 (RCV1) has a hierarchy of about 110 classes [12]. Open Directory Project (ODP, *http://dmoz.org*) and Yahoo! category (*http://dir.yahoo.com*) are two hierarchical ontology schemes for organizing web pages. International Patent Classification is an official category for maintaining patent documents [7].

The widespread use of hierarchies makes hierarchical classification a hot topic in research communities. Classification refers to the task of learning a model from classified instances that can predict the classes of previously unseen instances; hierarchical classification differs from normal classification by that the classes are organized in a hierarchy. In hierarchical classification, an example that belongs to some class automatically belongs to all its super classes.

Hierarchial classfication approaches fall into two categories according to their objectives. The first category of approaches aims to raise classification accuracy via hierarchies, which use hierarchies as additional information in deciding the classes of an instance, instead of using the content of the instance only. Barutcuoglu et al. model hierarchy constraints by a Bayesian Network, which combines individual binary predicts on each classes [1]. Cai and Hofmann develop a hierarchical SVM [3]; Labrou and Finin develop hierarchical Rochhio-like classifiers [11]. Lu et al. incorporate hierarchy into the task decomposition of a Min-Max modular network [14].

The second category of approaches aims to reduce computational complexity via hierarchies. Mostly, an isomorphism hierarchical network of classifiers is built. In the training phrase, base classifiers are trained with respect to internal nodes or parent-children branches, by the instances belonging to those nodes; in the classifying phrase, instances are filtered through the network of classifiers, predicting one or several topics at each layer, until they reach the bottom layer [4, 13]. This category of approaches is called top-down method [17, 19].

Flat classification approaches can also be used to handle hierarchical classifications besides the above two categories of hierarchical approaches [4, 13]. Flat classification approaches ignore hierarchies and treat the tasks as normal multiclass classifications.

Among all these methods, top-down methods are rela-

*This work was partially supported by the National Natural Science Foundation of China (Grant No. 60773090 and Grant No.90820018), the National Basic Research Program of China (Grant No.2009CB320901), the National High-Tech Research Program of China (Grant No. 2008AA02Z315), and the Science and Technology Commission of Shanghai Municipality (Grant No. 09511502400).

tively more valuable in real world applications, as most real-world datasets are large-scale ones with hundreds of thousands of classes. For example, the ODP has over 130 000 classes and the IPC has about 650 000 classes. The flat methods and accuracy-aimed methods are not feasible on these datasets.

However, top-down method suffers accuracy decrease, for which there are two explanations [2, 4, 19]. The first explanation is the data sparsity in bottom nodes, which is caused by the simultaneous increase in the number of classes and their organization. This data sparsity hinders the training of classifiers for these bottom nodes in the training phrase. The second explanation is the error propagation in the test phrase. The prediction errors occurs at high levels are not recoverable at lower layers, thus errors are amplified through each layer and eventually greatly reduce the accuracy.

In this paper, we flatten the hierarchies to reduce the accuracy decrease of top-down methods. Two flattening strategies, top-flattening and bottom-flattening, are proposed based on the two explanations of accuracy decrease, error propagation and data sparsity, in order to get higher accuracies via the flattened hierarchies. Besides the computational complexity before and after flattening are analyzed. We do experiments on a sub-hierarchy of ODP, which shows the flattening strategy based on error propagation overperforms the other one.

D'Alessio et al study the effects of modifying the hiearchies in the context of ACTION algorithm, and they also achieve positive results [6]. Malik flattens the hierarchies in the context of the top-down methods, while he conducts flattening only in the bottom-up manner [15]. This corresponds to the bottom-flattening in our paper, which is proved to be less effective than the top-flattening.

The rest of this paper is organized as follows: we first formally describe the top-down method and analyze its computational complexity in Sec.2; then we present our flattening method and two flattening strategies , analyze the computational complexity after flattening in Sec.3; after that we present the experiments where three flattened hierarchies are tested on two datasets in Sec.4; in the end we give our conclusions and future works in Sec.5.

2 Top-down method

In this section we formally present the top-down method and its related issues, which are the foundations of this paper. We first define the task of hierarchical classification; then we present the algorithm of top-down method; in the end we analyze the computational complexity of top-down method, where we explain how this method makes large-scale classifications feasible.

There are two issues about our presentation compared with previous researches. First, we use natural multi-class classifiers, and train one classifier for each node to distinguish its children nodes; while previously most researchers use binary classifiers, and train one classifier for each parent-child branch. Second, we handle the problem of single-label hierarchical classification. For the counterpart methods of multi-label hierarchical classification, please see [4].

2.1 Hierarchical classification

The task of single-label hierarchical classification can be defined as follows(similar with [18]):

Given:

- an instance space X
- a class hierarchy(C, $>_h$), where C is a set of classes and $>_h$ is a partial order representing the parent-child relationship (for all C_1,C_2 $\in$ C, $C_1 >_h C_2$ if and only if C_1 is the parent of C_2).
- a label space Y = $\{(y^1, y^2, \ldots, y^k) \mid y^j >_h y^{j+1}, y^k$ is a leaf$\}$, which denotes all the paths from the root to leafs
- a set T of examples$\{(x_i,y_i) \mid i=1,2,\ldots,n\}$

Find:
find a function f:X $\rightarrow$ Y that maximizes accuracy f(x)=y.

2.2 Top-down method

The algorithm of top-down includes training and classifying, which are given in the form of pseudo-code in algorithm 1. The process of training is to make local training sets and training base classifiers on them. The process of classifying is to filter an instance through the hierarchy, from the root node to a certain leaf node.

2.3 Computational complexity

In this subsection, we first discuss the complexity of multiclass SVMs, which are used as the base classifiers in this paper. Then we will discuss the complexity of hierarchical classification.

2.3.1 Complexity of multiclass SVMs

To handle the multiclass problems at internal nodes in hierarchical classification, multiclass SVMs are used in this paper. They can achieve the same accuracy with the stat-of-the-art method of one-vs-rest binary SVMs, and with

Algorithm 1 Top-down method

```
Training
Input: a set of samples $\{(x_i,y_i)|\ i=1,2,\ldots,n\}$
    a class hierarchy H = (C, $>_h$)
Output: a network of classifiers $\{f_C \mid$ C is a internal node
    in H$\}$
 1: for C is an internal node of H do
 2:     $T_C = \{(x_i,y_i^p) \mid y_i^{p-1}=C\}$
 3:     $f_C \leftarrow$ train a classifier by $T_C$
 4: end for
 5: return $\{f_C \mid$ C is a internal node in H$\}$
Classifying
Input: a instance x
    a class hierarchy H = (C, $>_h$)
    a network of classifiers $\{f_C \mid$ C is a internal node in H$\}$
Output: a hierarchical label $y=(y^1,y^2,\ldots,y^k)$
 1: y $\leftarrow$ ()
 2: C $\leftarrow$ the root node of H
 3: while true do
 4:     C $\leftarrow f_C(x)$
 5:     y $\leftarrow (y^1,y^2,\ldots,y^k, C)$
 6:     if C is a leaf of H then
 7:         return y
 8:     end if
 9: end while
10: return y
```

much less time cost (http:// www.cs.cornell.edu/ People/ tj/ svm_light/ svm_multiclass.html).

Multiclass SVM is based on SVM^{struct} and uses cut-plane algorithm for training [8, 9]. The cut-plane solution is an iterative algorithm, terminating when a pre-defined error tolerance is satisfied. Joachims et al. prove that the number of iterations is linear to the number of samples, and not directly related to the size of label space [9]. While in multiclass SVMs each iterations' complexity is linear to the number of classes. So the complexity of training a multiclass SVM is

$$O\left(nm\right) \tag{1}$$

where n is the number of samples and m is the number of classes.

Empirically, a multiclass SVM runs many times faster than a batch of binary SVM under one-vs-rest framework in solving a multiclass classification problem. The underlying reason is that a multiclass SVM handles all the classes at the same time so it can save much common computation.

2.3.2 Complexity of hierarchical classification

Hierarchical classification reduces the computational complexity of a large classification problem, which can be seen as an effect of dividing it into many smaller ones. In hierarchical classification, training the root classifier is dealing with all the samples but fewer target classes, and training the rest nodes are dealing with both fewer samples and fewer classes.

The complexity of top-down method presented here is similar to the case discussed in Yang's paper [20]. By applying (1), the complexity of training in the top-down method can be expressed as

$$\begin{aligned} t_{hierarchy} &= \sum_{i=1}^{h}\sum_{j=1}^{l_i} O\left(n_{ij}m_{ij}\right) \\ &= \sum_{i=1}^{h} O\left(n\right) \sum_{j=1}^{l_i} O\left(\pi_{ij}m_{ij}\right) \end{aligned} \tag{2}$$

where h is the max depth of internal nodes, that is, the height of hierarchy minus one; l_i is the number of classes at the i^{th} level; i = 0, 1, …, h, and i = 0 corresponds to the root level; j = 1, 2,…,l_i are the classes at the i^{th} level; n_{ij} is the number of local training samples; m_{ij} is the number of local classes; $\pi_{ij} = \frac{n_{ij}}{n}$ and $\sum_{j=1}^{l_i} \pi_{ij}$=1.

The computational complexity of hierarchical classification depends on the hierarchies according to (2). To illustrate the effectiveness of hierarchy actually reducing the complexity, suppose the hierarchy is a standard m_0-ary tree, this is, all the internal nodes have m_0 children. Then we can get

$$\begin{aligned} t_{hierarchy} &= hnm_0 \\ &= hnm^{\frac{1}{h}} \end{aligned} \tag{3}$$

where n and m is the total number of samples and classes, h is the depth of the tree minus one. (3) shows that the complexity with respect to the number of classes is greatly reduced, which explains how hierarchy make large-scale classifications feasible.

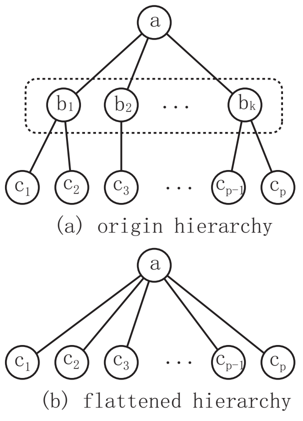
3 Flattening hierarchies

In this section we first present the process of flattening hierarchies; then we introduce two flattening strategies, top-flattening and bottom-flattening; at last we analyze the computation complexity of the top-down methods on flattened hierarchies.

3.1 Flattening method

Hierarchies are flattened by removing internal nodes. In Fig. 1, suppose nodes b_1, b_2, …, b_k be the nodes to be removed, a is the parent node, c_1, c_2 , c_3 ,…,c_p are the children nodes of b's. To remove nodes b's, we just connect nodes c's to parent node a. After flattening c's become the direct child nodes of a.

Figure 1. Flatten hierarchies by removing internal nodes



3.2 Flattening strategies

Error propagation and data sparsity in bottom nodes are the two possible causes of accuracy decrease in top-down methods, as mentioned in Sec. 1. We design two flattening strategy based on these two causes, named top-flattening strategy and bottom-flattening strategy.

Top-flattening strategy is based on the cause of error propagation. It removes the root node's child nodes and attaches the grandchild nodes directly to the root node. This results in enlarging the number of classes at the root node, and converting the top part of the original hierarchy from a two-layered hierarchical classification into a flat multiclass classification. Hence the chances that classification mistakes happen at top level are reduced, thus error propagation is relieved too.

Bottom-flattening strategy is based on the cause of data sparsity in bottom nodes. It removes the bottom internal nodes, and attaches their child nodes (or classes precisely) to high-layered parent nodes. Hence the bottom internal nodes in the new hierarchy are larger classification tasks with more training instances as well as more classes. So the data sparsity can be relieved.

3.3 Complexity of flattened hierarchies

Let's consider the changes of complexity for the top-down method on flattened hierarchy in figure 1. The complexities before and after flattening are

$$t_{origin} = O\left(nk + \sum_{i=1}^{k} n_i m_i\right)$$

$$t_{flattened} = O\left(n \sum_{i=1}^{k} m_i\right).$$

Hence the complexity increases after flattening. To illustrate the extent of increasing, suppose the hierarchy is a standard m-ary tree, that is,

$$k = n_1 = n_2 = \ldots = n_k = m.$$

Then the complexity formula can be simplified into ,

$$t_{origin} = 2O\left(nm\right)$$

$$t_{flattened} = O\left(nm^2\right),$$

which means the local computational complexity is $\frac{m}{2}$ times larger after flattening in a standard m-ary tree.

4 Experiments

In this section we do experiments on the datasets of ODP web pages to test the method of flattening hierarchies. We first introduce the experimental settings; then describe how we apply the two flattening strategies and get three flattened hierarchies; in the end we present and discuss the experiments' results.

4.1 Experimental settings

We use the datasets of ODP web pages in our experiments, and use normalized TFIDF as features. An empirical formula is established by us to estimate the optimal parameter C's of multiclass SVMs.

4.1.1 Datasets and preprocessing

The datasets of large-scale hierarchical text categorization (LSHTC, http://lshtc.iit.demokritos.gr) are used in our experiments, which are single-label hierarchical classification tasks obtained from ODP web pages, including over 120 000 classes and 155 000 instances. The datasets of the basic track and the cheap track are used here. The dataset of each track consists of a textual description of the hierarchy, a training set of about 90 000 labeled instances, a development set of about 30 000 labeled instances, and a test set of about 35 000 instances. These two track's hierarchies are the same, and the difference is feature vectors. In the basic task, features of all instances are indexing of web pages' content; while in the cheap task, features of training instances and development instances are indexing of web pages' ODP descriptions, and features of test instances are indexing of web pages' content.

The prediction labels of the test set can be submitted to the LSHTC web site, which returns the classification results including accuracy, macro-F_1, macro-p, macro-r, and tree induced-error. In the challenge participants can use both training set and development set in training classifiers, while here we use the training set only.

The instances of the LSHTC datasets are presented in the form of term frequencies, that is, IDs of words and their times of occurrences; the real words are unknown. The feature related preprocess that we perform includes TFIDF and normalization [12, 16].

4.1.2 Parameter C of SVM^{multiclass}

The key parameter of a multiclass SVM is the trade off between training errors and margins, usually noted as C, which is the same as a binary SVM. However, the parameter C of SVM^{multiclass} implemented by Joachims et al. is scaled differently from conventional binary SVM such as SVM^{light}.

We find that the following empirical formula estimates the optimal C of SVM^{multiclass} well[1],

$$C_{opt} \quad = \quad \max(4nm, 40\,000), \qquad (4)$$

where n and m are numbers of samples and classes.

4.1.3 Software and hardware

Our system is coded with the computer languages of C and Python. In order to use Joachims' multiclass SVM in an efficient manner as the base classifiers of top-down method, its IO parts have been rewritten. The training of classifiers can be run parallel.

The experiments are done on a 64-bit computer with an AMD 1.4Ghz CPU and 64G memory. Large memory is required by the base classifier of multiclass SVM, which uses up to 20G memories in learning the largest classification task (the root node in flattened hierarchies).

4.2 Flatten hierarchy of ODP

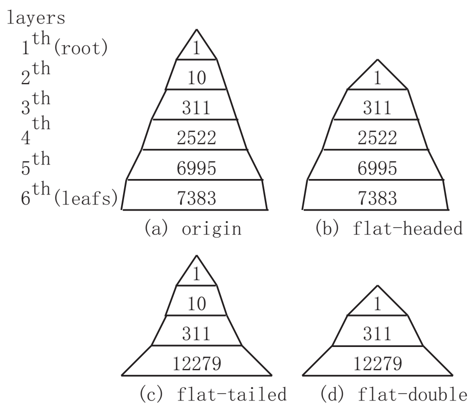
The ODP hierarchy of LSHTC dataset is 6-layered in height, with 4 928 internal nodes and 12 994 leafs. By applying the flattening strategies in Sec. 3.2, three hierarchies are made(see fig. 2). Note that leaf nodes can locate at high layers other than the bottom layer.

First, by the top-flattening strategy, the internal nodes at the second layers are removed. The new root node is connected to two parts of nodes: the original leaf nodes at the second layer, and the removed nodes' child nodes at the third layer. As the new hierarchy has a large number of parent-child branches (311 branches) at the first layer, we name it flat-headed hierarchy.

Second, by the bottom-flattening strategy, the internal nodes at the fourth and fifth layers are removed. In the new hierarchy, the original leaf nodes at the fifth and sixth layer are attached to their ancestor nodes at the third layer. As the new hierarchy has a large number of parent-child branches (311 branches) just above the bottom layer, we name it flat-dilate hierarchy.

[1]There is an exception that we set the C of the root node in flattened hierarchy to be 6×10^6

Figure 2. Number of nodes at each layers of experiments' ODP Hierarchies



Third, by the both two strategies, the internal nodes at the original second, fourth and fifth layers are all removed. The new hierarchy is a heavily flattened 3-layered hierarchy, which we name flat-double hierarchy.

4.3 Experimental results and discussion

The experimental results of the four hierarchies at two tasks are shown in Tab. 1, with measurements of effectiveness and time cost. Here the measurement of accuracy is taken as a main criterion, since LSHTC datasets are single-labeled ones. The following three points can be got from these results.

First, flattening hierarchies can raise classification accuracies, as all the flattened hierarchies overperform the original one, and the heavily flattened hierarchy of flat-double achieves the highest accuracy.

Second, the top-flattening strategy overperforms the bottom-flattening strategy, as the flat-headed hierarchies accuracies are obviously higher that the flat-tailed ones', though the previous one has the larger height and costs less training time. This implies that error propagation has a greater effect on the accuracy decrease of top-down methods than bottom nodes' data sparsity does.

Third, flattening hierarchies does raise the computational complexity as the training times are lengthened several times, which roughly agrees with the complexity analysis in Sec. 3.3.

Table 1. Results on LSHTC datasets

Task	Hierarchy	Acc.	Ma-F_1	Training
basic	origin	0.409	0.278	8h21m
	flat-headed	0.420	0.284	25h35m
	flat-tailed	0.411	0.286	37h53m
	flat-double	0.423	0.293	53h57m
cheap	origin	0.322	0.206	54m
	flat-headed	0.341	0.218	1h20m
	flat-tailed	0.331	0.217	1h24m
	flat-double	0.349	0.230	2h20m

5 Conclusions and future work

Through this research we demonstrate that flattening hierarchies can raise the accuracies of hierarchical classification. Moreover, by testing two opposite flattening strategies by experiments, we know that error propagation should be blamed more than data sparsity for the accuracy decrease of top-down method. From another viewpoint, flat-headed hierarchies can provides higher classification accuracy than flat-tailed hierarchies .

Besides, theoretical analysis shows that the computation complexity of training classifiers via flattened hierarchies will be enlarged several times, linear to the number of branches per node, which is roughly confirmed by the experiments.

In terms of practical applications, our method of flattening hierarchies can serve as a trade-off between standard top-methods and flat methods, with respect to time cost and accuracy. The top-flattening strategy shows a direction of flattening hierarchies effectively. Typically, for a 6-layered sub-hierarchy of ODP, a 2-layered hierarchy of classifiers is proper, which both makes the task feasible and ensures high accuracy.

In terms of researches, the good performance of top-flattening strategy suggests that error propagation is the main cause of the accuracy decrease of top-down method in hierarchical classifcation. Researchers might take it into consideration when doing researches related to hierarchical classification.

In future we will first test our method on larger datasets, such as IPC, in order to confirm the results in this paper and to test the scalability of our method. Then we will try to develop more effective hierarchical methods based on top-flattening strategy.

References

[1] Z. Barutcuoglu, R. E.Schapire, and O. G. Troyanskaya. Hierarchical multi-label prediction of gene function. In Journal of *Bioinformatics*, 22(7), pp.830-836, 2006.

[2] P. N. Bennett, and N.Nguyen. Refined experts: improving classification in large taxonomies In Proc. of *SIGIR*, pp.11-18, 2009.

[3] L. Cai,and T. Hofmann. Hierarchical document categorization with support vector machines. In Proc. of *CIKM*, pp.78-87, 2004.

[4] M. Ceci, and D. Malerba. Classifying web documents in a hierarchy of categories: a comprehensive study. In Journal of *Intelligent Information Systems*, 28, pp.37-38, 2007.

[5] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Hierarchical classification: combining Bayes with SVM In Proc. of *ICML*, pp.177-184, 2006

[6] S. D'Alessio, K. Murray, R. Schiaffino, and A. Kershenbaum. Category levels in hierarchical text categorization. In Proc. of *EMNLP*, 1998.

[7] C. J. Fall, A. Torcsvari, K. Benzineb, and G. Karetka. Automated categorization in the international patent classification In Journal of *ACM SIGIR*, 37(1), pp.10-25, 2003.

[8] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In Proc. of *ICML*, 2004.

[9] T. Joachims, T. Finley, and C. N. Yu. Cutting-plane training of structural SVMs. In Journal of *Machine Learning*, 77(1), pp.27-59, 2009.

[10] D. Koller, and M. Sahami. Hierarchically classifying documents using very few words. In Proc. of *ICML*, pp.170-178, 1997.

[11] Y. Labrou, and T. W. Finin. Yahoo! as an ontology: using Yahoo! categories to describe documents. In Proc. of *the 8th ACM CIKM*, pp.180-187, 1999.

[12] D. D. Lewis, Y. M. Yang, T. G. Rose, and F. Li. RCV1: a new benchmark collection for text categorization research. In Journal of *Machine Learning Research*, Vol.5, pp.361-397, 2004.

[13] T. Y. Liu, Y. M. Yang, H. Wan, H. J. Zeng, Z. Chen, and W. Y. Ma. Support vector machines classification with a very large-scale taxonomy. In *SIGKDD Explorations*, 7(1), pp.36-43,2005.

[14] B. L. Lu, and X. L. Wang. A parallel and modular pattern classification framework for large-scale problems. In *Handbook of pattern recognition and computer vision (4th edition , Chen, C. H.)*, pp.725-746, World Scientific, 2009.

[15] H. H. Malik. Improving hierarchical SVMs by hierarchy flattening and lazy classification. In *http://lshtc.iit.demokritos.gr/short_papers*, 2009.

[16] F. Sebastiani. Machine learning in automated text categorization, In Journal of *ACM Computing Surveys*, 34, pp.1-47, 2002.

[17] A. Sun, and E. P. Lim. Hierarchical text classification and evaluation. In Proc. of *IEEE ICDM*, pp.521-528, 2001.

[18] C. Vens, J. Struyf, L. Schietgat, S. Dzeroski, and H. Blockeel. Decision trees for hierarchical multi-label classification. In Journal of *Machine Learning*, pp.185C214, 2008.

[19] G. R. Xue, D. K. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In Proc. of *ACM SIGIR*, pp.619-626, 2008.

[20] Y. M.Yang, J. Zhang,and B. Kisiel. A scalability analysis of classifiers in text categorization. In Proc. of *ACM SIGIR*, pp.96-103, 2003.