Recursive NMF: Efficient Label Tree Learning for Large Multi-Class Problems

Lei Liu$^+$, Prakash Mandayam Comar$^+$, Sabyasachi Saha*, Pang-Ning Tan$^+$, Antonio Nucci*
$^+$*Dept. of Computer Science, Michigan State University, East Lansing, MI 48824, USA*
$^+\{$*liulei1, mandayam, ptan*$\}$*@msu.edu*

**Narus Inc. 570 Maude Court, Sunnyvale, CA 94085, USA*
$^*\{$*ssaha, anucci*$\}$*@narus.com*

Abstract

Many object recognition or concept identification tasks require accurate detection of large number of classes. These applications present enormous challenges to traditional classification methods, which are mostly designed for solving problems with small number of classes. In this paper, we develop a method called recursive non-negative matrix factorization (RNMF) for building a hierarchical label tree over set of classes. The internal nodes of the tree employ linear classifiers to propagate a data instance to its corresponding leaf node, where one or more one class support vector machine (SVM) classifiers is applied to accurately predict its class. Our experiment results show that the proposed method achieves significant gain in test efficiency and comparable accuracy to some of the more expensive label tree learning methods.

1 Introduction

Classification with large number of classes is an important but challenging problem with many potential applications including malware detection [8], visual object recognition [3], and text categorization [5]. Existing approaches for multi-class classification employ a collection of binary classifiers, organized either in a non-hierarchical (flat) structure or a hierarchical graph architecture. Non-hierarchical approaches, including one-versus-rest [10], one-versus-one [6], and error-correcting output coding (ECOC) [7], require invocation of all the binary classifiers during test time in order to make their multi-class predictions. Thus, the efficiency of such methods during testing is bounded by the number of binary classifiers used to uniquely identify the k classes. Clearly, the one-versus-one method is not a feasible solution when the number of classes is large due to the quadratic number of binary classifiers that must be constructed, unlike the one-versus-rest method that requires only k binary classifiers.

Hierarchical approaches [9, 11, 2, 1, 3] help to further reduce the number of classifiers that need to be invoked during testing by organizing the classifiers in a tree or DAG-like structure. Their runtime complexity during testing depends on the depth of the structure. For example, the Decision Directed Acyclic Graph (DDAG) [9] approach arranges the one-versus-one classifiers in a rooted binary DAG in order to reduce the runtime complexity for testing from $O(k(k-1)/2)$ to $O(k)$. In fact, it is possible to achieve sublinear complexity (e.g., $O(\log k)$) by assigning different subsets of the classes to the nodes of the hierarchical structure. Such a strategy was adopted by label tree learning methods such as [1, 3], which induces a linear classifier at each node to assign the data instances to their corresponding child nodes. With this approach, only the binary classifiers located along the path from the root node to one of the leaf nodes in the hierarchical structure will be invoked to determine the class label of a test instance.

The hierarchical approaches vary from one another in terms of how their tree or DAG structure and the associated classes of each node are learned from training data. For example, Bengio et al. [1] developed their label embedding tree approach by training k one-vs-rest classifiers to obtain an initial confusion matrix, which is then used as the affinity matrix for applying spectral clustering to iteratively partition the classes into smaller subgroups. Limitations of this method, which include the additional training time needed for constructing the initial confusion matrix and the hard partitions generated by the nodes of the tree, have been well-documented [3]. Deng et al. [3] presented an alternative strategy that simultaneously learns the class partitioning and the weights of the associated linear classifiers by optimizing their joint objective function. However, the approach requires solving an integer programming problem, which is NP-hard. Instead the authors proposed relaxing the problem to a linear program in order to find a polynomial time solution.

This paper presents a novel label tree learning approach to speed up model testing in large multi-class problems. Unlike previous work, the leaf nodes of the tree may contain multiple one-class SVM models to distinguish instances from different classes. This helps to shorten the depth of the tree without significantly degrading its accuracy. Our label tree learning algorithm partitions the classes and learns the weights of their classifiers simultaneously. It allows for soft partitioning of the classes by minimizing an information-theoretic loss function, which can be cast into a regularized non-negative matrix factorization problem. The regularization term introduced by our framework is unique in that it minimizes the entropy of the node partitions. Our experimental results suggest that the induced tree achieves high accuracy comparable to more complex tree learning methods but is significantly shorter.

2 Tree Learning with Recursive NMF

Let $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ denote the training instances sampled from a distribution P over $\mathcal{X}\times\mathcal{Y}$, where $\mathcal{X} = \mathcal{R}^d$ is the d-dimensional feature space and $\mathcal{Y} = \{1, 2, \ldots k\}$ is the set of k distinct classes. The goal of multi-class classification is to learn a target function f that minimizes the classification error of any instance drawn from P, i.e., $\mathbf{Pr}_{(x,y)\sim P}[f(x) \neq y]$. A typical approach for multi-class learning is to reduce the problem into multiple binary classification tasks. However, when the number of classes is large, applying all the binary classifiers to determine the predicted class becomes computationally infeasible.

This paper presents a novel and efficient label-tree learning algorithm for solving large multi-class problems. A label tree [1] is a structure $T =< V, E, \Phi, \Lambda >$, where V is the set of nodes, E is the set of directed links connecting each node to its children, Φ is the set of classification functions and Λ is the set of class labels associated with the nodes. Let $v_0 \in V$ denote the root node, which has no incoming links, and $V_l \subset V$ denote the leaf nodes, which have no outgoing links. The label tree framework proposed in this study is shown in Figure 1. The framework recursively applies a regularized non-negative matrix factorization approach for soft partitioning of the class labels in the tree. It also employs one or more 1-class SVM classifiers at the leaf nodes of the tree.

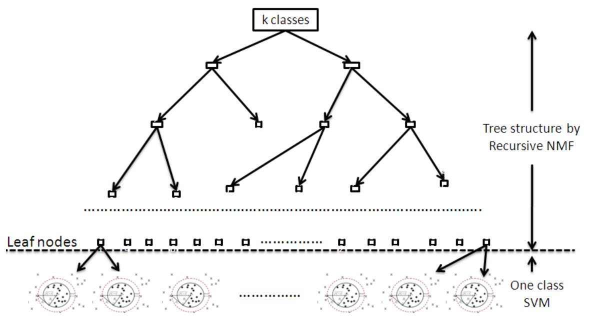

Figure 1. Proposed RNMF framework

Let $\mathbf{X}$ be an $n \times d$ data matrix containing the feature vector of the training instances and $\mathbf{Y}$ be an $n \times k$ matrix that represents the class label information, i.e., $\mathbf{Y}_{ij} = 1$ if the i^{th} instance belongs to the j^{th} class, and zero otherwise. Let p be the branching factor of the tree. At the root node v_0, our goal is to learn a partition membership matrix $\mathbf{L}$ of size $n \times p$ where n is the number of training instances assigned to the node and the associated feature weight matrix $\mathbf{W}$, which is a $d \times p$ matrix that indicates the importance of each feature in defining the given partition. This is accomplished by minimizing the following objective function:

$$\begin{aligned} \min_{\mathbf{L},\mathbf{W}} \quad & D(\mathbf{X} \parallel \mathbf{L}\mathbf{W}^T) + \lambda H(\mathbf{L}^T\mathbf{Y}) \\ s.t. \quad & \mathbf{L}^T\mathbf{1}_p = \mathbf{1}_n \end{aligned} \tag{1}$$

where $D(X||LW^T)$ refers to the Kullback-Leilber divergence between the input data matrix $\mathbf{X}$ and the product of its latent factors $\mathbf{L}\mathbf{W}^T$. The regularizer term $H(\mathbf{L}^T\mathbf{Y})$ corresponds to the entropy of the class distribution of each partition. The regularizer is needed to avoid splitting instances from the same class into different partitions. Since the entropy definition of the partitions requires the rows of matrix $\mathbf{L}^T\mathbf{Y}$ to be properly normalized (i.e., sum up to 1), i.e.,

$$H(\mathbf{M}) = -\sum_i \frac{M_{i+}}{M_{++}} \sum_j \frac{M_{ij}}{M_{i+}} \log \frac{M_{ij}}{M_{i+}},$$

where $M_{i+} = \sum_j M_{ij}$ and $M_{++} = \sum_{ij} M_{ij}$, the normalization is achieved by the constraint $\mathbf{L}^T\mathbf{Y}^T\mathbf{1}_p = \mathbf{L}^T\mathbf{1}_p = \mathbf{1}_n$ since $\mathbf{Y}^T\mathbf{1}_p = \mathbf{1}_p$. λ is a parameter provided by the user that controls the trade off between optimal partition of X and class purity of each partition[1]. Thus, the Lagrange formulation of (1) is given by:

$$\begin{aligned} \mathcal{L} \;=\; & \sum_{i,j}[X_{ij} \log \frac{X_{ij}}{(LW^T)_{ij}} - X_{ij} + (LW^T)_{ij}] \\ & - \; \lambda \sum_{ij}(L^TY)_{ij} \log(L^TY)_{ij} - \sum_s \mu_s(\sum_m L^T_{sm} - 1) \end{aligned}$$

[1] λ is set to the default value 1 in our experiments

The objective function is minimized by taking its partial derivative with respect to the parameters W and L and setting them to zero:

$$\frac{\partial \mathcal{L}}{\partial W_{pq}} = -\sum_i \frac{X_{ip}L_{iq}}{(LW^T)_{ip}} + \sum_i L_{iq} = 0 \quad (2)$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial L_{pq}} &= -\sum_j \frac{X_{pj}W_{jq}}{(LW^T)_{pj}} + \sum_j W_{jq} \\ &- \lambda \sum_j Y_{pj}\log(L^TY)_{qj} \\ &- \lambda \sum_j Y_{pj} - \mu_q = 0 \quad (3)\end{aligned}$$

By summing up over p in Equation (3), we have

$$\begin{aligned}\mu_q = &- \frac{1}{n}\sum_{jp} \frac{X_{pj}W_{jq}}{(LW^T)_{pj}} + \sum_j W_{jq} \\ &- \lambda \sum_{jp} \frac{Y_{pj}}{n}\log(L^TY)_{qj} - \lambda \quad (4)\end{aligned}$$

Replacing (4) back into (3), the partial derivative can be rewritten as follow:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial L_{pq}} &= -\sum_j \frac{X_{pj}W_{jq}}{(LW^T)_{pj}} + \frac{1}{n}\sum_{jp} \frac{X_{pj}W_{jq}}{(LW^T)_{pj}} \\ &- \lambda \sum_j Y_{pj}\log(L^TY)_{qj} \\ &+ \lambda \sum_{jp} \frac{Y_{pj}}{n}\log(L^TY)_{qj} \quad (5)\end{aligned}$$

The objective function can be solved using a gradient descent approach. Following the approach used in [4], the gradient descent formula can be transformed into a multiplicative update formula as follows:

$$W_{pq} = W_{pq}\frac{(X^TL)_{pq}}{(WL^TL)_{pq}},\ L_{pq} = L_{pq}\frac{\alpha}{\beta} \quad (6)$$

in which,

$$\begin{aligned}\alpha &= \sum_j \frac{X_{pj}W_{jq}}{(LW^T)_{pj}} + \lambda \sum_j Y_{pj}\log(L^TY)_{qj} \\ \beta &= \frac{1}{n}\sum_{jp} \frac{X_{pj}W_{jq}}{(LW^T)_{pj}} + \lambda \sum_{jp} \frac{Y_{pj}}{n}\log(L^TY)_{qj}\end{aligned}$$

Algorithm 1 summarizes the key steps of our proposed recursive Non-Negative Matrix Factorization (RNMF) framework for finding the partitions at each node in the tree. We recursively apply Algorithm 1 to partition the training instances until the stopping criteria is met. The stopping criteria for splitting a node in the tree are as follows: (1) stop if all instances in a partition belong to the same class, or (2) stop if the node contains less than $minleaf$ training instances, or (3) stop if there is only one class containing more than $minclass$ instances[2].

Algorithm 1 Recursive NMF at a node v

```
1: Input: Matrices $\mathbf{X}_v$, $\mathbf{Y}_v$, and MaxIter
2: Output: Matrices $\mathbf{W}_v$ and $\mathbf{L}_v$
3: Initialize: $\mathbf{W}_v^{old}$ and $\mathbf{L}_v^{old}$ to random matrices
4: for j=1 to MaxIter do
5:     update $\mathbf{W}_v^{new}$ using (6)
6:     update $\mathbf{L}_v^{new}$ using (6)
7:     set $\mathbf{W}_v^{old} \leftarrow \mathbf{W}_v^{new}$; $\mathbf{L}_v^{old} \leftarrow \mathbf{L}_v^{new}$;
8: end for
```

Table 1. Classification results

	F1	Test time (s)	depth
CM [1]	0.4826 ± 0.0017	195.5± 9.2	10
DDAG [9]	0.5309 ± 0.002	942.9 ± 17.6	214
RNMF(p=5)	0.4960 ± 0.0015	115.7 ± 3.9	7

Instead of learning a separate linear classifier at each node, the classifiers' weights are obtained from $\mathbf{W}_v$ as follows. Given a test instance $\mathbf{x}_{\text{test}}$, we determine which partition of node v it should go to by computing the following $1 \times p$ vector: $\pi = \mathbf{x}_{\text{test}}^T\mathbf{W}_v(\mathbf{W}_v^T\mathbf{W}_v)^{-1}$ and choose the partition j that has the largest magnitude, i.e., $j = \arg\max_k \pi_k$. The assignment procedure is repeated until the test instances reach the leaf node, where we apply all the 1-class SVM models associated with the leaf node to predict the label for the test instances. The multi-class prediction using multiple 1-class SVM is described in our previous work [8].

3 Experimental Evaluation

Our experiments were performed using Wikipedia data from October 9, 2009. After preprocessing, there are about 3.6 million articles categorized into 336K categories. We choose the largest 214 categories which correspond to 24,378 articles. We use the F1 score to evaluate our classifier's performance. All the results reported in this study are based on 5-fold cross validation. We compared the performance of RNMF against two state-of-the art algorithms proposed in the literature: DDAG [9] and the confusion matrix approach (CM) [1]. The results are summarized in Table 1. The results show that RNMF outperforms the confusion matrix based label embedding tree approach both in terms of their F1 score and test efficiency. Furthermore, the depth of our tree is also shorter. When compared against DAGSVM,

[2]We set $minleaf = 15$ and $minclass = 5$ for our experiments.

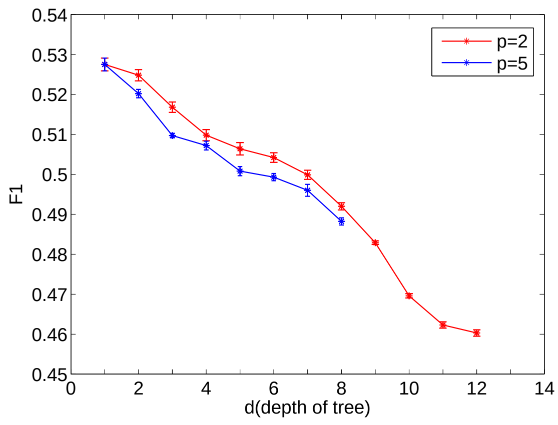

Figure 2. F1 score for RNMF with different branching factor (p) and depths (d).

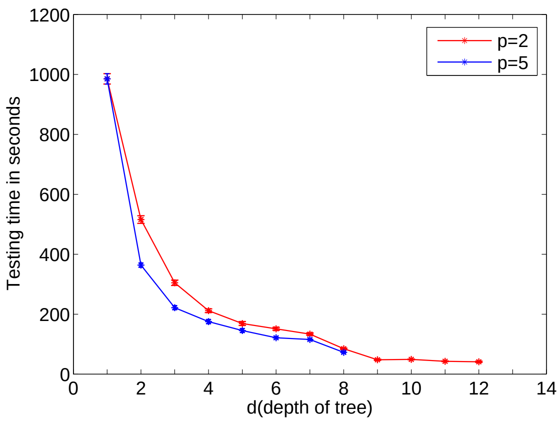

Figure 3. Test time for RNMF with different branching factor (p) and depths (d).

which requires building a quadratic number of 1-vs-1 classifiers, our F1-score is slightly worse but our test time is at least 8 times faster.

Note that the performance of RNMF depends on the branching factor p and the depth of the tree. If the tree has only a single node (i.e., a flat model consisting of k one-class SVM models), our F1 score and test times are comparable to that of DAGSVM (as shown in Figures 2 and 3). As the depth of the tree increases, the test time improves significantly at the expense of decreasing F1 values. In fact, if we terminate the tree growing procedure at depth equals to 3, the F1 score is around 0.51 but the speedup is almost 3 times faster.

4 Conclusion

This paper presents a novel learning algorithm for large multi-class problems called Recursive NMF. The algorithm creates soft partitions of the classes by solving a regularized non-negative matrix factorization problem. It uses a set of 1-class SVM models to predict the classes at the leaf nodes of the tree. Our experimental results on Wikipedia data suggests that RNMF outperforms a state-of-the-art label embedding tree algorithm [1] both in terms of its F1 score and test time. It also achieves comparable F1-score but higher testing efficiency compared to DDAG [9].

References

[1] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *Neural Information Processing Systems*, pages 163–171, 2010.

[2] A. Beygelzimer, J. Langford, and P. Ravikumar. Multiclass classification with filter trees. In *Information Theory and Application Workshop*, 2007.

[3] J. Deng, S. Satheesh, A. C. Berg, and F. F. F. Li. Fast and balanced: Efficient label tree learning for large scale object recognition. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 567–575. 2011.

[4] D.Lee and H.Seung. Algorithm for non-negative matrix factorization. In *Neural Information Processing Systems*, pages 556–562, 2001.

[5] R. Ghani. Using error-correcting codes for efficient text classification with a large number of categories. Masters thesis, Carnegie Mellon University, 2001.

[6] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471, 2001.

[7] J. Langford and A. Beygelzimer. Sensitive error correcting output codes. In *Conference on Learning Theory*, pages 158–172, 2005.

[8] P. MandayamComar, L. Liu, S. Saha, P.-N. Tan, and A. Nucci. One-class svm with tree transformation and profiling for zero-day malware detection. Technical Report MSU-CSE-12-4, Michigan State University, 2012.

[9] J. C. Platt, N. Cristianini, and J. Shawe-taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, pages 547–553. MIT Press, 2000.

[10] R. Ryan and K. Aldebaro. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, December 2004.

[11] V. Vural and J. G. Dy. A hierarchical method for multiclass support vector machines. In *Proceedings of the twenty-first international conference on Machine learning*, pages 105–112, 2004.