Regularization Framework for Large Scale Hierarchical Classification

Siddharth Gopal[1], Yiming Yang[1], and Alexandru Niculescu-Mizil[2]

[1] Carnegie Mellon University, Pittsburgh, USA
[2] NEC Labs, Princeton, USA

Abstract. In this paper, we propose a hierarchical regularization framework for large-scale hierarchical classification. In our framework, we use the regularization structure to share information across the hierarchy and enforce similarity between class-parameters that are located nearby in the hierarchy. To address the computational issues that arise, we propose a parallel-iterative optimization scheme that can handle large-scale problems with tens of thousands of classes and hundreds of thousands of instances. Experiments on multiple benchmark datasets showed significant performance improvements of our proposed approach over other competing approaches.

Keywords: Hierarchical Regularization, Parallelization, Hierarchical Classification

1 Introduction

With the tremendous growth of data, there is an increasing need to provide a hierarchically structured view of the data at multiple granularity levels. The large taxonomies for Web Page categorization at Yahoo! Directory and the Open Directory Project, the Image-Net Hierarchy for organizing images, the International Patent Classification Hierarchy for patents, the Reuters hierarchy for news-stories are examples of such widely used hierarchies.

Hierarchical Classification (HC) goes beyond traditional classification schemes like Binary or Multiclass in the sense that we need to address joint training and inference based on the hierarchical dependencies between class labels. Moreover, this has to be done a computationally efficient and scalable manner as many real world hierarchies are characterized by large taxonomies and high dimensionalities.

In this paper, we propose two methods for HC that incorporate the hierarchical dependencies between the class labels into the regularization structure of the parameters. The key idea is to shrink the parameters of sibling nodes in the hierarchy towards the parameters of their common parent node. This enforces the parameters of a node be similar to the parameters of its children and its parent and thereby captures the idea that classes which are nearby in the hierarchy share similar model parameters. To enable large-scale implementation and training, we develop a parallel iterative optimization scheme for our proposed approaches. In our specific way of formulating the objective, the parameters of each node depends only on the parameters of its parent and children. This keeps the dependencies between the parameters minimal and aids in achieving a large degree of parallelization. We tested our proposed approaches on 4 large-scale benchmark datasets (3 of them released as a part of LSHTC challenge) and found that they not only consistently outperform other competing approaches, but can also be trained in a matter of several minutes.

2 Related Work

There is a large body of existing work on Hierarchical Classification (HC). Some of the early works in HC such as [8], [13], [10] use the hierarchical structure to break the problem down into sub-problems recursively along the hierarchy by allocating a classifier at each individual node. The hierarchy is used to partition the training datasets into node-specific subsets after which all the classifiers are trained independently without using the hierarchy any further. The use of the hierarchy in such works is fairly limited.

Several heuristic approaches have been proposed to better utilize the hierarchical structures. In [3], a cross-validation strategy is employed to add the output of the top-level classifiers as features for the second-level classifiers. A bayesian aggregation on the result of the individual binary classifiers was proposed in [6]. [22] proposed a pruning technique to the prune the hierarchy based on the test-instance and re-training on the smaller hierarchy.

Large-margin discriminative methods proposed by [20], [4] use discriminant functions that takes the contributions from all nodes along the path to the root node, and the parameters are jointly learned to minimize a global loss over the hierarchy. Similar Online variants of such methods have been proposed in [7], [5].

In [24], the authors enforce orthogonality constraints between the parent and the children in additional to minimizing the loss at individual nodes. Hierarchical shrinkage approaches such as isotonic smoothing [16], smoothing the estimated parameters in Naive Bayes classifiers along the path from root to the leaf node [14] have also been tried. [17] proposed Multinomial logistic models that takes sum of contributions along the path with bayesian priors on the variances. For a more thorough survey and comparison of hierarchical classification methods refer [19].

3 Proposed Method

Let the hierarchy be a tree defined over a set of nodes $\mathcal{N}$ by the parent child relationships given by $\pi : \mathcal{N} \to \mathcal{N}$ where $\pi(n)$ is the parent of node n. Let $\mathbf{D} = \{x_i, t_i\}_{i=1}^{M}$ denote the training dataset of M instances where each $x_i \in \mathbb{R}^d$ and $t_i \in T$ is a label, $T \subset \mathcal{N}$ is the set of all leaf nodes in the hierarchy. In our task, we assume that each instance is labeled exactly to one of the leaf nodes in the hierarchy. For convenience, let C_n denote the set of all children of node n, and binary variable y_{in} denote if x_i belongs to class $n \in T$ i.e.

$$C_n = \{c : \pi(c) = n\}$$
$$y_{in} = \begin{cases} +1 & t_i = n \\ -1 & t_i \neq n \end{cases}$$

The problem of HC is to learn a prediction function $f : \mathcal{X} \rightarrow T$ that predicts the target class-label of a given input instance with smallest possible error. More over, the hierarchical dependencies between the classes are encoded in the form of a hierarchy which can be used in the learning process.

Following the principle of statistical decision theory the risk (or error) of a prediction function f is defined as the expected value of a loss function over all possible inputs. Since in principle we do not know the distribution over (x, y), it is not possible to find the optimal f with lowest achievable risk. We therefore resort to the Structural Risk Minimization framework which prescribes choosing f to minimize a combination of the Empirical Risk based on the training dataset and a regularization term to penalize the complexity of f. Typically the prediction function f is parameterized by an unknown set of parameters w which are then estimated in the learning process. The estimated parameters $\hat{w}$ is given by

$$\hat{w} = \arg\min_{w} \lambda(w) + C \times R_{emp} \qquad (1)$$

where R_{emp} denotes the Empirical Risk or Loss on the training dataset, $\lambda(w)$ denotes the regularization term and C is a parameter that controls the trade-off between fitting to the given training examples and the complexity of f.

In our problem of HC, the prediction function is parameterized by a set of weight vectors $\mathbf{W} = \{w_n : n \in \mathcal{N}\}$ i.e each node n in the hierarchy is associated with a weight-vector w_n. Following the principles of Structural risk minimization the parameters are estimated by minimizing a weighted combination of the Empirical Risk and a regularization term. First, we define the Empirical Risk in our model as the loss incurred by the examples at the leaf-nodes of the hierarchy

$$R_{emp} = \sum_{n \in T} \sum_{i=1}^{M} L(y_{in}, x_i, w_n)$$

where L could be any convex loss function. Second, we propose to use the hierarchy in the learning process by incorporating the hierarchical structure into the regularization term for $\mathbf{W}$. We propose the following hierarchical form of regularization

$$\lambda(\mathbf{W}) = \sum_{n \in \mathcal{N}} \frac{1}{2} ||w_n - w_{\pi(n)}||^2$$

This form of regularization models the hierarchical dependencies in the sense that it encourages parameters which are nearby in the hierarchy to be similar to each other in euclidean norm. This helps classes to leverage information from nearby classes while estimating model parameters and helps share statistical strength across the hierarchy. We hope that this would especially enable classes with very few training instances to pool in information, as well as leverage information from classes with a larger number of training examples, to yield better classification models despite the limited training examples.

We explore two choices for the loss function L and define two different variants of our approach - Hierarchically Regularized Support Vector Machines (HR-SVM) using the hinge-loss function and Hierarchically Regularized Logistic Regression (HR-LR) using the logistic loss function gives by,

$$\textbf{HR-SVM} \quad \min_{\mathbf{W}} \sum_{n \in \mathcal{N}} \frac{1}{2} ||w_n - w_{\pi(n)}||^2 + C \sum_{n \in T} \sum_{i=1}^{M} (1 - y_{in} w_n^\top x_i)_+ \tag{2}$$

$$\textbf{HR-LR} \quad \min_{\mathbf{W}} \sum_{n \in \mathcal{N}} \frac{1}{2} ||w_n - w_{\pi(n)}||^2 + C \sum_{n \in T} \sum_{i=1}^{M} \log(1 + \exp(-y_{in} w_n^\top x_i)) \tag{3}$$

The key advantage of HR-{SVM,LR} over other hierarchical models such as [20], [4], [2], [24] is that there are no constraints that maximizes the margin between correct and incorrect predictions. This keeps the dependencies between the parameters minimal and in turn enables us to develop a parallel-iterative method to optimize the objective (details in 4.3) thereby scaling to very large HC problems with tens of thousands of classes and hundreds of thousands of instances.

4 Optimization

4.1 HR-SVM

Although the objective function in (2) is convex and has a unique maximum, it is not differentiable and straight-forward methods such as gradient descent cannot be used. To address the non-differentiability, many works have generally resorted to subgradient techniques such as [24], [18], [1]. However, the general problem with subgradient approaches is that the learning rate of the optimization routine needs to be specified [9]. In our initial experiments using subgradient approaches, we found that the optimization was highly sensitive to the learning rate used and tweaking this parameter for each dataset was a difficult task involving several trials and adhoc heuristics. In order to overcome such issues, we resorted to an iterative approach where we update the parameters associated with each node n iteratively by fixing the rest of the parameters. To tackle the non-differentiability in some of the updates (i.e. the updates at the leaf-nodes), we converted these sub-problems into their dual form which is differentiable and optimized it using co-ordinate descent. This iterative scheme offers several advantages compared to standard approaches like subgradient descent,

1. There are no learning rates that need to be set or tweaked.
2. The non-differentiability of the objective at the leaf nodes is side-stepped by solving a differentiable dual subproblem.
3. It can easily incorporate closed form updates for the non-leaf nodes, thereby accelerating the convergence.

For each non-leaf node $n \notin T$, differentiating eq (2) w.r.t w_n yields a closed-form update for w_n given by,

$$w_n = \frac{1}{|C_n| + 1} \left(w_{\pi(n)} + \sum_{c \in C_n} w_c \right) \tag{4}$$

For each leaf node $n \in T$, the objective cannot be differentiated due to a discontinuous Hinge-loss function. Isolating the terms that depend on w_n and introducing slack variables ξ_{in}, the primal objective of the subproblem for w_n is given by,

$$\begin{aligned} \min_{w_n} \quad & \frac{1}{2}||w_n - w_{\pi(n)}||^2 + C\sum_{i=1}^{M} \xi_{in} & \\ \text{subject to} \quad & \xi_{in} \geq 0 & \forall i = 1..M \\ & \xi_{in} \geq 1 - y_{in} w_n^\top x_i & \forall i = 1..M \end{aligned}$$

The dual of the above subproblem by introducing appropriate dual variables α_i , $i = 1..M$ is

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_{in} y_{jn} x_i^\top x_j - \sum_{i=1}^{M} \alpha_i (1 - y_{in} w_{\pi(n)}^\top x_i) \tag{5}$$

$$0 \leq \alpha_i \leq C \tag{6}$$

To solve this subproblem, one case easily use any second order methods such as interior-point methods etc. The downside of such solvers is that it takes a long time even for a single iteration and requires the entire kernel matrix of size $O(M^2)$ to be stored in memory. Typical large-scale HC problems have atleast hundreds of thousands of instances and the memory required to store the kernel matrix is in the order of 100 GB for each class, thereby rendering it impractical. Instead we propose a co-ordinate descent approach which has minimal memory requirements and converges quickly even for large problems. Our work is based on the dual co-ordinate descent developed in [9].

The core idea in co-ordinate descent is to iteratively update each dual variable. In our objective function eq (5), the update for each dual variable has a simple closed form solution. To derive the update for the i^{th} dual variable α_i given by $\alpha_i + d$, we solve the following one-variable problem,

$$\begin{aligned} \min_{d} \quad & \frac{1}{2} d^2 (x_i^\top x_i) + d \left(\sum_{i=1}^{M} \alpha_i y_{in} x_i \right)^\top x_i - d(1 - y_{in} w_{\pi(n)}^\top x_i) \\ \text{subject to} \quad & 0 \leq \alpha_i + d \leq C \end{aligned}$$

Basically, we substituted α_i by $\alpha_i + d$ in eq (5) and discarded all the terms that do not depend on d. This one-variable problem can be solved in closed form by considering the gradient of the objective and appropriately updating α_i to obey the constraints. The gradient G for the above objective and the corresponding update for α_i is given by,

$$G = a'^{\top} x_i - 1 + y_{in} w_{\pi(n)}^{\top} x_i$$
$$\alpha_i^{new} = \min\left(\max\left(\alpha_i^{old} - \frac{G}{x_i^{\top} x_i}, 0\right), C\right)$$

where $a' = \sum\limits_{i=1}^{M} \alpha_i y_{in} x_i$ is an auxilary variable maintained and updated throughout the optimization of the subproblem. The time complexity for each α_i update is *O(#nnz in x_i)* - the number of non-zero dimensions in x_i and the memory requirement for solving the entire subproblem is $O(M)$ - far more efficient than that $O(M^2)$ compared to the second order methods. Finally, after solving the dual subproblem, the update for w_n in the primal form is given by the K.K.T conditions for the subproblem,

$$w_n = w_{\pi(n)} + \sum_{i=1}^{N} \alpha_i y_{in} x_i \tag{7}$$

The pseudocode for the entire optimization routine is shown in Algorithm 1 . Although co-ordinate descent methods are easy to implement, have low memory requirements and reach an acceptable solution quickly, they have their own pitfalls. If the data is highly correlated or dense in the dimensions, co-ordinate descent methods tend to take a much longer time to converge [15] . Even in simpler cases, identifying the rate of convergence is hard and the stopping criteria might not be accurate. However, in the case of HC text classification, the documents are very sparse and there is not much correlation between the dimensions. Moreover, most of the classes are linearly separable and finding a decision boundary is an 'easy' problem - this makes co-ordinate descent a natural choice for optimization in the context of HC text classification.

4.2 HR-LR

We follow a similar iterative strategy for optimizing HR-LR, i.e. we update the parameter w_n of each node n iteratively by fixing the rest of the parameters. Unlike HR-SVM, the objective function in HR-LR is convex and differentiable and therefore second order methods such exact newton's methods as well as quasi-newton methods are applicable. Typically, exact newton methods require the computation of the inverse of the Hessian of the objective function. For large-scale problems with high dimensions, it might not even be feasible to store the Hessian in memory. Therefore, we resort to a limited memory variant of a quasi newton method - Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) [12].

```
Input : Dataset $\mathbf{D}$, parameter $C$, hierarchy in terms of $\pi$,T, $\mathcal{N}$
Result : weight vectors $\mathbf{W}^*$
while Not Converged do
    foreach $n \in \mathcal{N}$ do
        if $n \notin T$ then
            Update $w_n$ using eq (4)
        else
            if HR-SVM then
                Create the dual subproblem in terms of $\alpha$ and solve eq (5)
                Update the primal parameters $w_n$ using eq (7)
            end
            if HR-LR then
                Solve eq (8) using LBFGS and set $w_n$ to the optimal value.
            end
        end
    end
end
```

Algorithm 1: Optimization Routine for HR-SVM and HR-LR

The update for each non-leaf node is exactly the same as in HR-SVM eq (4). For each leaf node n, isolating the terms that depend on w_n, the objective and the corresponding gradient G can be written as

$$\min_{w_n} \quad \frac{1}{2}||w_n - w_{\pi(n)}||^2 + C\sum_{i=1}^{M} \log(1 + \exp(-y_{in} w_n^\top x_i)) \tag{8}$$

$$G = w_n - w_{\pi(n)} - \sum_{i=1}^{M} \frac{1}{1 + \exp(y_{in} w_n^\top x_i)} y_{in} x_i$$

Since the gradient can be computed in closed-form it is possible to directly apply quasi newton methods such as LBFGS to solve the above optimization problem. The pseudocode for the optimization routine is shown in Algorithm 1

4.3 Parallelization

For large hierarchies, it might be impractical to learn the parameters of all classes, or even store them in memory, on a single machine. We therefore, devise a parallelization scheme for our optimization algorithm. The key idea is to note that the interactions between the different w_n's are only through the parent and child nodes. By fixing the parameters of the parent and children for a node n, the parameter w_n associated with node n can be optimized independently of the rest of the parameters. This leads to a simple parallelization strategy - traverse the hierarchy level by level, optimize the parameters at each level in parallel, and iterate until convergence.

A better way that yields a larger degree of parallelization is to iteratively optimize the odd and even levels - if we fix the parameters at the odd levels,

Table 1. Dataset Statistics

Dataset	#Training	#Testing	#Class-Labels	#Leaf-labels	Depth	#Features
RCV1	13732	468355	97	75	5	48734
IPC	46324	28926	552	451	4	541869
LSHTC-2010	128710	34880	15358	12294	6	381580
DMOZ-2012	177580	177581	13137	11737	6	348548
DMOZ-2011	384161	104263	33591	27840	6	594158

the parameters of parents and the children of all nodes at even levels are fixed, and the w_n's at all even levels can be optimized in parallel. The same goes for optimizing the odd level parameters. For large hierarchies, this method yielded a speedup close to linear in the number of processors.

We tested our parallelization framework on a cluster running map-reduce based Hadoop 20.2 with 64 worker nodes having 8 cores and 16GB RAM each. Around 300 cores were used as Mappers and 220 cores were used as Reducers. We used Accumulo 1.4 key-value store for fast retrieve-update of the w_ns.

5 Experimental Design

5.1 Datasets

We used the following popular large-scale benchmark datasets (Table 1) with the recommended train-test splits where-ever available.

- **RCV1** [11] A set of News stories from 1996-1997 available in 2 different hierarchies. We used the topic-hierarchy and retained only documents which belonged to exactly one of the leaf-nodes.
- **IPC** [21] A collection of patents organized according to the International Patent Classification Hierarchy.
- **LSHTC-2010** [3] A subset of webpages from the ODP released as a part of LSHTC-2010.
- **DMOZ-2012** [4] A subset of webpages from ODP released as a part of Track 2, LSHTC-2012. Since the challenge was officially closed at the time of writing this paper and we did not have access to the true test labels, we used our own 50-50 split of the released training dataset.
- **DMOZ-2012** [5] A subset of webpages from ODP released as a part of LSHTC-2011. Since our focus is on multiclass classification, we removed all training instances that had multiple labels. The test-set however, is intact and therefore results on the test-set can be comparable to other published results on this dataset. During testing, each instance is predicted with exactly one label.

[3] http://lshtc.iit.demokritos.gr/node/3
[4] http://lshtc.iit.demokritos.gr/LSHTC3_DATASETS
[5] http://lshtc.iit.demokritos.gr/LSHTC2_DATASETS

Table 2. Macro-F_1 and Micro-F_1 on the 5 datasets. Bold faced number indicate best performing method. The results of the significance tests are denoted * for a p-value less than 5% and † for p-value less than 1%.

	SVM	HR-SVM
RCV1		
Macro-F_1	55.29	**56.61**†
Micro-F_1	85.00	**85.32**†
IPC		
Macro-F_1	45.71	**47.89**†
Micro-F_1	53.12	**54.26**†
LSHTC-2010		
Macro-F_1	32.64	**33.12**
Micro-F_1	45.36	**46.02**
DMOZ-2012		
Macro-F_1	24.62	**25.59**†
Micro-F_1	54.87	**55.67**†
DMOZ-2011		
Macro-F_1	42.61	**43.82**
Micro-F_1	42.56	**43.77**

	LR	HR-LR
RCV1		
Macro-F_1	54.89	**55.67**†
Micro-F_1	85.16	**85.32**†
IPC		
Macro-F_1	48.29	**49.60***
Micro-F_1	55.03	**55.37**†
LSHTC-2010		
Macro-F_1	45.40	**45.84**
Micro-F_1	31.58	**32.42**
DMOZ-2012		
Macro-F_1	19.11	**21.62**†
Micro-F_1	53.29	**54.09**†
DMOZ-2011		
Macro-F_1	41.35	**42.33**
Micro-F_1	41.29	**42.27**

5.2 Methods for Comparison

We give a brief overview of some of the methods that we compared

- **HR-SVM** Our proposed hierarchically regularized Support vector machine.
- **HR-LR**: Our proposed hierarchically regularized logistic regression.
- **SVM**: One-versus rest Binary Support vector machines.
- **LR**: One-versus rest regularized logistic regression.

To evaluate on such large-scale datasets, for HR-LR and HR-SVM we used the parallelization strategy discussed in section 4.3. SVM and LR can be trivially parallelized.

Although there are several other competing hierarchical approaches such as [20], [4], [24] we are unable to evaluate as they cannot scale to size of the datasets that we test on. For all approaches, we tune the regularization parameter using cross-validation with a range of values from 10^{-3} to 10^3.

6 Results

6.1 Performance Comparison

For all the evaluations, we used the standard Micro-F_1 and Macro-F_1 evaluation measures [23]. Table 2 compares the results of using the hierarchical regularization framework against the corresponding non-hierarchical counterparts as baselines.

In our experiments, even some of our baseline results achieve state-of-the-art (or close to) performance. For e.g., on the DMOZ-2011 dataset, our baseline

Table 3. The training time taken (in minutes) for all the methods across all datasets.

Dataset	**SVM**	**HR-SVM**	**LR**	**HR-LR**
RCV1	2.73	10.54	2.89	11.74
IPC	3.12	13.54	4.17	15.91
LSHTC-2010	5.12	27.55	97.24	123.22
DMOZ-2012	21.34	126.83	98.38	223.60
DMOZ-2011	38.47	221.23	102.90	242.45

SVM has a 3.7% higher Micro-F_1 and Macro-F_1 than the best reported results on this dataset.

Despite such strong baselines, we can clearly see that on all the tested datasets, both HR-SVM and HR-LR outperform SVM and LR respectively. The average gain in Micro-F_1 and Macro-F_1 are .716% and 1.80% respectively. The gain in Macro-F_1 is particularly large indicating that the hierarchical regularization framework improved the performance for majority of the rare categories.

We further conducted significant tests using sign-test for Micro-F_1 and a wilcoxon rank test on the Macro-F_1 scores. On each dataset, HR-SVM and HR-LR are compared against the SVM and LR respectively. The null hypothesis is that there is no significant difference between the two systems being compared, the alternative is that the better-performing-method is better. (Note that we are unable to perform significance tests on LSHTC-2010 and DMOZ-2012 as the we did not receive class-wise performance measures from the submission system). The results show that our proposed approach performed significantly better on all tested datasets establishing the importance of modeling hierarchical relationships through our framework.

Comparing the performance of hinge-loss (SVM, HR-SVM) vs logistic (HR-LR,LR), we find that the logistic loss function performs better on smaller datasets while the hinge-loss function performs better on the large-scale datasets. This might be because in large-scale datasets majority of the class-labels have only a few training examples and the hinge-loss is able to generalize better than logistic models in such scenarios.

6.2 Timing Comparison

Comparing the time taken, HR-SVM is on an average 4x times more expensive than SVM and HR-LR is 2.5x times more expensive than LR. On the smaller datasets RCV1 and IPC, the difference between the hierarchical regularization framework and the non-hierarchical counterparts is higher. This is because most of the time taken is devoted to starting the solver in the cluster and for HR-SVM and HR-LR, the solver needs to be started multiple times as they are iterative methods. On the larger dataset, the difference is lesser since time taken to start the solver is masked by the actual optimization time.

Between hinge-loss and logistic models, the former is faster than the latter on all the datasets. In our experience, co-ordinate descent methods seem to be faster than second order methods especially when there is not much correlation between the different dimensions.

7 Conclusion

In this paper, we proposed a hierarchical regularization framework along with scalable optimization algorithms for large-scale hierarchical classification. We explored 2 different variants of our framework using the logistic loss function and the hinge-loss function. Our results on benchmark datasets showed consistent and significant improvement in performance.

In our future work, we plan to extend this approach along two directions - firstly, to tackle problems where the hierarchical dependencies between class-labels are given in the form a graph rather than a tree and secondly, handle cases where instances can have multiple correct labels instead of a single one.

References

1. http://leon.bottou.org/projects/sgd.
2. S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. *NIPS*, 23:163–171, 2010.
3. P.N. Bennett and N. Nguyen. Refined experts: improving classification in large taxonomies. In *SIGIR*, 2009.
4. L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM*, pages 78–87. ACM, 2004.
5. N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Incremental algorithms for hierarchical classification. *The Journal of Machine Learning Research*, 7:31–54, 2006.
6. C. DeCoro, Z. Barutcuoglu, and R. Fiebrink. Bayesian aggregation for hierarchical genre classification. In *Proceedings of the International Conference on Music Information Retrieval*, pages 77–80, 2007.
7. O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *ICML*, page 27. ACM, 2004.
8. S. Dumais and H. Chen. Hierarchical classification of web content. In *ACM SIGIR*, 2000.
9. C.J. Hsieh, K.W. Chang, C.J. Lin, S.S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.
10. D. Koller and M. Sahami. Hierarchically classifying documents using very few words. 1997.
11. D.D. Lewis, Y. Yang, T.G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.
12. D.C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.
13. T.Y. Liu, Y. Yang, H. Wan, H.J. Zeng, Z. Chen, and W.Y. Ma. Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD*, pages 36–43, 2005.
14. A. McCallum, R. Rosenfeld, T. Mitchell, and A.Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML*, pages 359–367, 1998.

15. T.P. Minka. A comparison of numerical optimizers for logistic regression. *Unpublished draft*, 2003.
16. K. Punera and J. Ghosh. Enhanced hierarchical classification via isotonic smoothing. In *Proceeding of the 17th international conference on World Wide Web*, pages 151–160. ACM, 2008.
17. B. Shahbaba and R.M. Neal. Improving classification when a class hierarchy is available using a hierarchy-based prior. *Bayesian Analysis*, 2(1):221–238, 2007.
18. S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated subgradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM, 2007.
19. C.N. Silla and A.A. Freitas. A survey of hierarchical classification across different application domains. *KDD*, 22(1):31–72, 2011.
20. I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6(2):1453, 2006.
21. IPC WIPO. http://www.wipo.int/classifications /ipc/en/support/.
22. G.R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *SIGIR*, pages 619–626. ACM, 2008.
23. Yiming Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1:67–88, 1999.
24. D. Zhou, L. Xiao, and M. Wu. Hierarchical classification via orthogonal transfer. Technical report, MSR-TR-2011-54, 2011.