Hierarchical Bayesian Inference and Recursive Regularization for Large-Scale Classification

SIDDHARTH GOPAL and YIMING YANG, Carnegie Mellon University

In this article, we address open challenges in large-scale classification, focusing on how to effectively leverage the dependency structures (hierarchical or graphical) among class labels, and how to make the inference scalable in jointly optimizing all model parameters. We propose two main approaches, namely the hierarchical Bayesian inference framework and the recursive regularization scheme. The key idea in both approaches is to reinforce the similarity among parameter across the nodes in a hierarchy or network based on the proximity and connectivity of the nodes. For scalability, we develop hierarchical variational inference algorithms and fast dual coordinate descent training procedures with parallelization. In our experiments for classification problems with hundreds of thousands of classes and millions of training instances with terabytes of parameters, the proposed methods show consistent and statistically significant improvements over other competing approaches, and the best results on multiple benchmark datasets for large-scale classification.

Categories and Subject Descriptors: 1.5.2 [**Pattern Recognition**]: Classifier Design and Evaluation

General Terms: Design, Algorithms, Experimentation

Additional Key Words and Phrases: Large-scale optimization, hierarchical classification, Bayesian methods

ACM Reference Format:
Siddharth Gopal and Yiming Yang. 2015. Hierarchical Bayesian inference and recursive regularization for large-scale classification. ACM Trans. Knowl. Discov. Data 9, 3, Article 18 (April 2015), 23 pages.
DOI: http://dx.doi.org/10.1145/2629585

1. INTRODUCTION

With the advent of big data, there is a growing need to provide structured and organized views of the data for effective search, browsing, and data mining. The large taxonomies of Yahoo! for classifying all of the Web pages on the Web, the International Patent Classification hierarchy for organizing patents, the extensive product hierarchy of Amazon[1] and Google,[2] the Wikipedia graph that connects related Wikipedia categories with each other for easy browsing, and the Medical Subject Heading hierarchy for indexing millions of articles in PubMed are excellent examples of human-generated structures to organize data in important applications.

Such structures present several challenges for large-scale classification with respect to how to leverage the dependencies among classes for more accurate modeling and

[1]http://docs.aws.amazon.com/AWSECommerceService/latest/DG/FindingItemsUsingBrowseNodes.html.
[2]https://support.google.com/merchants/answer/160081?hl=en.

This work is an extension of our previous works [Gopal and Yang 2010, 2013]. Gopal and Yang [2013] and Gopal et al. [2012] it is supported in part by the National Science Foundation (NSF) under grant IIS_1216282. The Open Cloud cluster is supported in part by the NSF under award CCF_1019104 and the Gordon and Betty Moore Foundation in the eScience project.
Authors' address: S. Gopal and Y. Yang, 5000 Forbes Ave, Pittsburgh, PA - 15213.

robust prediction. For example, consider classifying a Web page to the Yahoo! Directory hierarchy, a Web page that belongs to category Disease is also likely to be a member of category Health but unlikely to be a member of Music. One-against-rest classifiers, although most popular due to their simplicity, cannot model such dependencies because classifiers are trained independently and hence lack the necessary expressive power. To address such a limitation, we must go beyond the scope of independent training of classifiers.

Another challenge is the data sparsity issue for a majority of class labels, which is often associated with extremely skewed category distributions in very large taxonomies [Liu et al. 2005; Yang et al. 2003; Bennett and Nguyen 2009]. For example, 76% of the class labels in the Yahoo! Directory have fewer than 5 positive instances [Liu et al. 2005], and 63% of class labels in the English portion of Wikipedia have fewer than 10 positive instances. Joint learning of dependent models would be a better alternative if we could effectively "borrow" the positive training examples from related classes based on the proximity of classes in the provided hierarchies or graphs. Proposing multiple strategies for such structure-based learning and global optimization of classification models is a major contribution of this article.

Last, the sheer sizes of very large hierarchies and graphs present a significant computational challenge for classification algorithms. For example, the Wikipedia dataset[3] consists of approximately 2.4 million Wikipedia articles spanning across 325,000 classes (with graphical dependencies) and a vocabulary size of 1.6 million words. This amounts to approximately 520 billion parameters (325,000 $\times$ 1.6 million), which is roughly 2 terabytes of storage. This is computationally challenging even for training one-against-rest classifier per class and ignoring the dependencies among classification models. Joint learning of all parameters and taking the dependencies among classification models into account is even more challenging. Proposing scalable algorithms for such large-scale structure-based inference with state-of-the-art performance on benchmark datasets for large-scale classification is another major contribution of this paper.

Specifically, the major contributions of our work includethe following:

(1) *Hierarchical Bayesian logistic regression* (HBLR): HBLR is a Bayesian framework for multivariate logistic regression where the prior distribution for the parameters at a node is a Gaussian centered at the parameters of the parent node. Such a recursively propagated prior encourages the parameters of closely located nodes in the hierarchy to be similar to each other. The strength of the Gaussian prior, and hence the amount of information sharing between nodes, is controlled by its covariance parameter, which is also learned from the data. We study multiple ways of modeling the covariance structures and examine how the model flexibility affects the classification performance on various data.
(2) *Recursive regularization* (RR): In this framework, we use the dependencies (hierarchical or graphical) among classes and subclasses to define a joint objective for regularization of model parameters; the model parameters of the siblings nodes who share the same parent are regularized toward the common parent node. Depending on the choice of loss function, this framework is applicable to both large-margin classifiers (support vector machines (SVMs)) as well as probabilistic classifiers (logistic regression).
(3) *Scalable inference*: For our HBLR models, we develop the first hierarchical variational inference method that is orders of magnitude faster than typical MCMC approaches. For RR, we develop fast training procedures based on local dual coordinate descent as well as variants of LBFGS [Liu and Nocedal 1989]-based

[3]http://lshtc.iit.demokritos.gr/.

optimization schemes. In both cases, we localize the dependencies among model parameters to enable efficient parallelization of the training tasks and subtasks. We show for the first time that global hierarchical optimization (with our proposed methods) can be efficiently computed for the largest datasets, such as Wikipedia with 325,000 classes and 2 million training instances in a matter of 37 hours.

(4) *Large-scale evaluation*: We present a thorough evaluation of our proposed methods in comparison with seven representative baseline methods on 10 benchmark datasets, including the large Wikipedia dataset with 478,021 classes. Our methods significantly outperform the baseline methods on multiple datasets, including the largest one.

2. RELATED WORK

There has been more than a decade of work in hierarchical classification. The most popular in the early stage are the pachinko machine models [Dumais and Chen 2000; Yang et al. 2003; Liu et al. 2005; Koller and Sahami 1977], where the classification task is decomposed into subtasks recursively, and each node of the hierarchy has an independently trained classifier. The hierarchy is only used to partition the training data and not used any further in the training. The simplicity makes these methods easy to scale but also makes them limited in effectively using the hierarchical dependencies.

Several approaches have been proposed for making better use of the hierarchical structure. In Bennett and Nguyen [2009], a cascading strategy is employed to add the output of lower-level classifiers as additional features for higher-level classifiers. In DecCoro et al. [2007], a Bayesian aggregation on the results of the individual binary classifiers was proposed. In Xue et al. [2008], a data-driven pruning strategy was proposed for reducing the size of the original hierarchy. Some improvements over the results of the pachinko machine models have been reported; however, these approaches are heuristic by nature.

The more principled methods include the large-margin models by Tsochantaridis et al. [2006], Cai and Hofmann [2004], Rousu et al. [2006], Dekel et al. [2004], and Cesa-Bianchi et al. [2006], where the discriminant functions take the contributions from all nodes along the path to the root, and the model parameters are jointly learned to minimize a global loss over the hierarchy. Similar ideas have been explored in Zou et al. [2011], where orthogonality conditions are imposed between the parent and children classifiers, in Shahbaba and Neal [2007] with Bayesian multinomial logistic models, and in McCallum et al. [1998] using naive Bayes classifiers with hierarchical shrinkage. Although there are a few works that address the scalability of these methods with a large number of training instances [Gornitz et al. 2011; Widmer et al. 2010], there is no work that focuses on scaling with a large number of classes. In fact, empirical improvements of most of these methods over simpler approaches have been shown only on small datasets with typically hundreds (or fewer) class labels. The primary difficulty for *most* of these methods in scaling is due to the high degree of interdependencies among model parameters and that the parameters for all classes cannot be held in memory at the same time.

3. HIERARCHICAL BAYESIAN LOGISTIC REGRESSION

HBLR uses a Bayesian framework for leveraging the hierarchical class structure. The Bayesian framework is a natural fit for this problem, as it can seamlessly capture the idea that the models at the lower levels of the hierarchy are specializations of models at ancestor nodes.

Define a hierarchy as a set of nodes $\mathcal{N} = \{1, 2...\}$ with the parent relationship $\pi : \mathcal{N} \rightarrow \mathcal{N}$ where $\pi(n)$ is the parent of node $n \in \mathcal{N}$. Let $\mathbf{D} = \{(x_i, t_i)\}_{i=1}^{N}$ denote the training data where $x_i \in \mathbb{R}^d$ is an instance and $t_i \in T$ is a label where $T \subset \mathcal{N}$ is the set of leaf nodes in the hierarchy labeled from 1 to $|T|$. We assume that each instance is assigned to one of the leaf nodes in the hierarchy. If there are any instances assigned to an internal node, spawn a leaf node under it and reassign all instances from the internal node to this new leaf node. Let C_n be the set of all children of node n.

For each node $n \in \mathcal{N}$, we associate a parameter vector w_n, which has a Gaussian prior. We set the mean of the prior to the parameter of the parent node, $w_{\pi(n)}$. In what follows, we consider three alternate ways to model the covariance matrix, which we call M1, M2, and M3 variants of HBLR. In the M1 variant, all siblings share the same spherical covariance matrix. Formally, the generative model for M1 is

$$\begin{aligned}
\text{M1}\quad & w_{root} \sim \mathcal{N}(w_0, \Sigma_0), \qquad \alpha_{root} \sim \Gamma(a_0, b_0) \\
& w_n |\, w_{\pi(n)}, \Sigma_{\pi(n)} \sim \mathcal{N}(w_{\pi(n)}, \Sigma_{\pi(n)}) \quad \forall n, \qquad \alpha_n \sim \Gamma(a_n, b_n) \quad \forall n \notin T \\
& t \mid x, \mathbf{W} \sim \quad Categorical(p_1(x), p_2(x), \ldots, p_{|T|}(x)) \quad \forall (x, t) \in \mathbf{D} \\
& p_i(x) = \exp\left(w_i^\top x\right) / \Sigma_{t' \in T} \exp\left(w_{t'}^\top x\right).
\end{aligned} \tag{1}$$

The parameters of the root node are drawn using user-specified parameters w_0, Σ_0, a_0, b_0. Each nonleaf node $n \notin T$ has its own α_n drawn from a Gamma with the shape and inverse-scale parameters specified by a_n and b_n. Each w_n is drawn from the Normal with mean $w_{\pi(n)}$ and covariance matrix $\Sigma_{\pi(n)} = \alpha_{\pi(n)}^{-1} I$. The class labels are drawn from a multinomial whose parameters are a soft-max transformation of the w_ns from the leaf nodes. This model leverages the class hierarchy information by encouraging the parameters of closely related nodes (parents, children, and siblings) to be more similar to each other than those of distant ones in the hierarchy. Moreover, by using different inverse variance parameters α_n for each node, the model has the flexibility to adapt the degree of similarity between the parameters (i.e., parent and children nodes) on a per-family basis. For instance, it can learn that sibling nodes higher in the hierarchy (e.g., *mammals* and *birds*) are generally less similar compared to sibling nodes lower in the hierarchy (e.g., *chimps* and *orangutans*).

Although this model is equivalent to the correlated multinomial logit (corrMNL) proposed in Shahbaba and Neal [2007], the hierarchical logistic regression formulation is different from corrMNL and has a distinct advantage that the parameters can be decoupled. As we shall see in Section 3.1, this enables the use of scalable and parallelizable variational inference algorithms that make our approach 750 times faster than Shshbaba and Neal [2007].

We can further extend M1 by allowing the diagonal elements of the covariance matrix $\Sigma_{\pi(n)}$ to be feature specific instead of uniform. In our previous example with subtopics *mammals* and *birds*, we may want $w_{mammals}$, w_{birds} to be commonly close to their parent in some dimensions (e.g., in some common features like eyes, breath, and claw) but not in other dimensions (e.g., in *bird*-specific features like feathers or beak). We accommodate this by replacing prior α_n using $\alpha_n^{(i)}$ for every feature (i). This form of setting the prior is referred to as automatic relevance determination (ARD) and forms the basis of several works, such as sparse Bayesian learning [Tipping 2001], and relevance vector machines [Bishop and Tipping 2003]. We define the M2 variant of the HBLR approach as

$$\begin{aligned}
\text{M2}\quad & w_n |\, w_{\pi(n)}, \Sigma_{\pi(n)} \sim \mathcal{N}\big(w_{\pi(n)}, \Sigma_{\pi(n)}\big) \qquad \forall n \\
& \alpha_n^{(i)} \sim \Gamma\big(a_n^{(i)}, b_n^{(i)}\big) \quad i = 1..d,\ \forall n \notin T \qquad \text{where } \Sigma_{\pi(n)}^{-1} = \operatorname{diag}\left(\alpha_{\pi(n)}^{(1)}, \alpha_{\pi(n)}^{(2)}, \ldots, \alpha_{\pi(n)}^{(d)}\right).
\end{aligned}$$

Yet another extension of the M1 model would be to allow each node to have its own covariance matrix for the Gaussian prior over w_n, not shared with its siblings. This enables the model to learn how much the individual children nodes differ from the parent node. For example, consider the topic *mammals* and its two subtopics *whales* and *carnivores*; the subtopic *whales* is distinct from a typical *mammal* and is more of an outlier topic. Such mismatches are typical in hierarchies, especially in cases where there is not enough training data and an entire subtree of topics is collapsed as a single node. M3 aims to cope with such differences:

$$\begin{aligned} \text{M3} \quad w_n | \, w_{\pi(n)}, \Sigma_n &\sim \mathcal{N}(w_{\pi(n)}, \Sigma_n) \qquad \forall n \\ \alpha_n &\sim \Gamma(a_n, b_n) \qquad \forall n \notin T. \end{aligned}$$

Note that the only difference between M3 and M1 is that M3 uses $\Sigma_n = \alpha_n^{-1} I$ instead of $\Sigma_{\pi(n)}$ in the prior for w_n. In our experiments, we found that M3 consistently outperformed the other variants, suggesting that such effects are important to model in hierarchies. Although it would be natural to extend M3 by placing ARD priors instead of the uniform α_n, we do not expect to see better performance due to the difficulty in learning a large number of parameters. Preliminary experiments confirmed our suspicions, so we did not explore this direction further.

3.1. Variational Inference

We outline the inference method for M2, but the procedure can be easily extended for M1 and M3. The primary inference procedure in Bayesian methods is to calculate the posterior distribution of the parameters. Specifically, the posterior distribution of parameters in model M2 is given by

$$\begin{aligned} p(\mathbf{W}, \boldsymbol{\alpha} | \mathbf{D}) &\propto p(\mathbf{t} | \mathbf{X}, \mathbf{W}, \boldsymbol{\alpha}) p(\mathbf{W}, \boldsymbol{\alpha}) \\ &\propto \prod_{(x,t) \in D} \frac{\exp\left(w_t^\top x\right)}{\sum\limits_{t' \in T} \exp\left(w_{t'}^\top x\right)} \prod_{n \in \mathcal{N} \setminus T} \prod_{i=1}^{d} p\left(\alpha_n^{(i)} | a_n^{(i)}, b_n^{(i)}\right) \prod_{n \in \mathcal{N}} p\left(w_n | w_{\pi(n)}, \Sigma_{\pi(n)}\right). \end{aligned} \tag{2}$$

Since a closed-form solution for $p(\mathbf{W}, \boldsymbol{\alpha} | \mathbf{D})$ is not possible due to the nonconjugacy between the logistic likelihood and the Gaussian prior, we resort to the variational EM algorithm that approximates this posterior. More specifically, we seek such a distribution q that has a simple factored form and is closest in *KL* divergence to the true posterior $p(\mathbf{W}, \boldsymbol{\alpha} | \mathbf{D})$. We use independent Gaussian $q(w_n)$ and Gamma $q(\alpha_n)$ distributions for w_n and α_n per node as the factored representation:

$$q(\mathbf{W}, \boldsymbol{\alpha}) = \prod_{n \in \mathcal{N} \setminus T} q(\alpha_n) \prod_{n \in \mathcal{N}} q(w_n) \propto \prod_{n \in \mathcal{N} \setminus T} \prod_{i=1}^{d} \Gamma\left(. | \tau_n^{(i)}, \upsilon_n^{(i)}\right) \prod_{n \in \mathcal{N}} \mathcal{N}(. | \mu_n, \Psi_n).$$

Our goal is to estimate the parameters of this new factored posterior $q(\mathbf{W}, \boldsymbol{\alpha})$ such that it maximizes the following variational lower bound (VLB) of the likelihood of the labels,

$$\begin{aligned} \log P(\mathbf{t} | \mathbf{X}) &\geq \int q(\boldsymbol{W}, \boldsymbol{\alpha}) \log p(\boldsymbol{W}, \boldsymbol{\alpha}, \boldsymbol{t} | \mathbf{X}) d\boldsymbol{W} d\boldsymbol{\alpha} - \int q(\boldsymbol{W}, \boldsymbol{\alpha}) \log q(\boldsymbol{W}, \boldsymbol{\alpha}) d\boldsymbol{W} d\boldsymbol{\alpha} \\ \Rightarrow \quad VLB &= E_q \left[\log p(\boldsymbol{W}, \boldsymbol{\alpha}, \boldsymbol{t} | \mathbf{X})\right] + H(q) \\ &= E_q \left[\log p(\boldsymbol{t} | \boldsymbol{W}, \boldsymbol{\alpha}, \mathbf{X})\right] + E_q \left[\log p(\boldsymbol{W}, \boldsymbol{\alpha})\right] + H(q). \end{aligned} \tag{3}$$

To tackle the nonconjugacy inside $p(\mathbf{t} | \mathbf{X}, \mathbf{W}, \boldsymbol{\alpha})$ in (2), we use a suitable lower bound to the soft-max normalization constant proposed by Bouchard [2007], for any $\beta \in \mathcal{R}$,

$\xi_k \in [0, \infty)$:

$$\log\left(\sum_k e^{g_k}\right) \le \beta + \sum_k \left[\frac{g_k - \beta - \xi_k}{2} + \lambda(\xi_k)((g_k - \beta)^2 - \xi_k^2) + \log(1 + e^{\xi_k})\right]$$
$$\text{where} \qquad \lambda(\xi) = \frac{1}{2\xi}\left(\frac{1}{1 + e^{-\xi}} - \frac{1}{2}\right), \tag{4}$$

where β , ξ_k are variational parameters that we can optimize to get the tightest possible bound. For every training example x, we introduce variational parameters β_x and $\xi_{xn} \forall n \in T$. In the E-step, the local variational parameters are fixed, and the posterior for a parameter is computed by optimizing the VLB with respect to the posterior parameters. The parameters are updated as

$$\Psi_n^{-1} = I(n \in T) \sum_{(x,t) \in D} 2\lambda(\xi_{xn}) x x^\top + \operatorname{diag}\left(\frac{\tau_{\pi(n)}}{\upsilon_{\pi(n)}}\right) + |C_n| \operatorname{diag}\left(\frac{\tau_n}{\upsilon_n}\right), \tag{5}$$

$$\mu_n = \Psi_n \left(I(n \in T) \sum_{(x,t) \in D} (I(t = n) - \frac{1}{2} + 2\lambda(\xi_{xn})\beta_x) x + \operatorname{diag}\left(\frac{\tau_{\pi(n)}}{\upsilon_{\pi(n)}}\right) \mu_{\pi(n)} \right.$$
$$\left. + \operatorname{diag}\left(\frac{\tau_n}{\upsilon_n}\right) \sum_{c \in C_n} \mu_c \right)$$
$$\upsilon_n^{(i)} = b_n^{(i)} + \sum_{c \in C_n} \Psi_n^{(i,i)} + \Psi_c^{(i,i)} + \left(\mu_n^{(i)} - \mu_c^{(i)}\right)^2 \qquad \text{and } \tau_n^{(i)} = a_n^{(i)} + \frac{|C_n|}{2}. \tag{6}$$

In the M-step, we keep the parameters of the posterior distribution fixed and optimize the variational parameters ξ_{xn}, β_x to maximize the VLB:

$$\xi_{xn}^2 = x^\top \operatorname{diag}\left(\frac{\tau_n}{\upsilon_n}\right) x + \left(\beta_x - \mu_n^\top x\right)^2 \qquad \beta_x = \frac{\frac{1}{2}(\frac{1}{2}|T| - 1) + \sum_{n \in T} \lambda(\xi_{xn}) \mu_n^\top x}{\sum_{n \in T} \lambda(\xi_{xn})}.$$

3.2. Partial MAP Inference

In most applications, the requirement for a matrix inversion of Ψ_n in step (5) could be demanding. In such scenarios, we split the inference into two stages, first calculating the posterior of w_n (for the leaf nodes) using MAP solution, and second calculating the posterior of α_n. In the first stage, we find the MAP estimate w_n^{map} (for $n \in T$) and then use Laplace approximation to approximate the posterior using a separate Normal distribution for each dimension, thereby leading to a diagonal covariance matrix. Note that due to the Laplace approximation, w_n^{map} and the posterior mean μ_n coincide:

$$w^{map} = \arg\max_{\mathbf{W}} \sum_{n \in T} -\frac{1}{2}\left(w_n - \mu_{\pi(n)}\right)^\top \operatorname{diag}\left(\frac{\tau_{\pi(n)}}{\upsilon_{\pi(n)}}\right)\left(w_n - \mu_{\pi(n)}\right) + \log p(\mathbf{t}|\mathbf{W}, \mathbf{X}, \boldsymbol{\alpha})$$
$$\left(\Psi_n^{(i,i)}\right)^{-1} = \sum_{(x,t) \in D} x^{(i)} p_{xn}(1 - p_{xn}) x^{(i)} + \operatorname{diag}\left(\frac{\tau_{\pi(n)}}{\upsilon_{\pi(n)}}\right), \tag{7}$$

where p_{xn} is the probability that training instance x is labeled as n. The arg max in (7) can be computed for all μ_n at the same time using optimization techniques like LBFGS [Liu and Nocedal 1989]. For the second stage, parameters τ_n and υ_n are updated using (6). Full MAP inference is also possible by performing an alternating maximization between w_n, α_n, but we do not recommend it as there is no gain in scalability compared to partial MAP inference and it loses the posterior distribution of α_n.

3.3. 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 parallel memory-efficient implementation scheme for our partial MAP inference. There are four sets of parameters that are updated: $\{\mu_n, \Psi_n, \tau_n, \nu_n\}$. The Ψ_n, τ_n, ν_n can be updated in parallel for each node using (5) and (6).

For μ, the optimization step in (7) is not easy to parallelize because the w's are coupled together inside the soft-max function. To make it parallelizable, we replace the soft-max function in (1) with multiple binary logistic functions (one for each terminal node), which removes the coupling of parameters inside the log-normalization constant. The optimization can now be done in parallel by making the following observations. First, note that the optimization problem in (7) is concave maximation; therefore, any order of updating the variables reaches the same unique maximum. Second, note that the interactions between the w_n's are only through the parent and child nodes. By fixing the parameters of the parent and children, the parameter w_n of a node can be optimized independently of the rest of the hierarchy. One simple way to parallelize is to traverse the hierarchy level by level, optimize the parameters at each level in parallel, and iterate until convergence. A better way that achieves a larger degree of parallelization is to iteratively optimize the odd and even levels: if we fix the parameters at the odd levels, 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. To aid convergence, we interleave the μ, Ψ updates with the τ, ν updates and warm start with the previous value of μ_n. In practice, for the larger hierarchies, we observed speedups linear in the number of processors. Note that the convergence follows from viewing this procedure as block coordinate ascent on a concave differentiable function [Lou and Tseng 1992].

3.4. Setting Prior Parameters

The w_0, Σ_0 represent the overall mean and covariance structure for the w_n. We set $w_0 = 0$ and $\Sigma_0 = I$ because of their minimal effect on the rest of the parameters. The $a_n^{(i)}, b_n^{(i)}$ are variance components such that $\frac{b_n^{(i)}}{a_n^{(i)}}$ represents the expected variance of the $w_n^{(i)}$. Typically, choosing these parameters is difficult before seeing the data. The traditional way to overcome this is to learn $\{a_n, b_n\}$ from the data using empirical Bayes. Unfortunately, in our proposed model, one cannot do this because each $\{a_n, b_n\}$ is associated with a single α_n. Generally, we need more than one sample value to learn the prior parameters effectively [Casella 1988].

We therefore resort to a data-dependent way of setting these parameters by using an approximation to the observed Fisher information matrix. We first derive on a simpler model and then extend it to a hierarchy. Consider the following binary logistic model with unknown w, and let the Fisher information matrix be I and observed Fisher information $\hat{I}$. Given some training examples D_{binary}.

$$\begin{aligned} Y \mid x &\sim Bernoulli\left(\frac{\exp(w^\top x)}{1+\exp(w^\top x)}\right) \\ I &= E\left[p(x)(1-p(x))xx^\top\right] \\ \hat{I} &= \sum_{(x,t)\in D_{binary}} \hat{p}(x)(1-\hat{p}(x))xx^\top. \end{aligned} \tag{8}$$

It is well known that I^{-1} is the asymptotic covariance of the MLE estimator of w, so a reasonable guess for the covariance of a Gaussian prior over w could be the observed $\hat{I}^{-1}$ from a dataset D_{binary}. The problem with $\hat{I}^{-1}$ is that we do not have a good estimate $\hat{p}(x)$ for a given x, as we have exactly one sample for a given x (i.e., each instance x is labeled exactly once with certainty, and thus $\hat{p}(x)(1 - \hat{p}(x))$ will always be zero). Therefore, we approximate $\hat{p}(x)$ as the sample prior probability independent of x (i.e., $\hat{p}(x) = \hat{p} = \Sigma_{(x,t)\in D} \frac{t}{|D|}$). Now, the prior on the covariance of w can be set such that the expected covariance is $\hat{I}^{-1}$.

To extend this to hierarchies, we need to handle multiple classes, which can be done by estimating $\hat{I}(n)^{-1}$ for each $n \in T$, as well handle multiple levels, which can be done by recursively setting a_n, b_n as follows:

$$\left(a_n^{(i)}, b_n^{(i)}\right) = \begin{cases} \left(\sum\limits_{c\in C_n} a_c^{(i)}, \sum\limits_{c\in C_n} b_c^{(i)}\right) & \text{if } n \notin T \\ \left(1, \hat{I}(n)^{-1(i,i)}\right) & \text{if } n \in T, \end{cases}$$

where $\hat{I}(n)$ is the observed Fisher information matrix for class label n. This way of setting the priors is similar to the method proposed in [Kass and Natarajan 2006]; the key differences are in approximating $p(x)(1 - p(x))$ from the data rather than using $p(x) = \frac{1}{2}$ and extension to handle multiple classes as well as hierarchies.

We also tried other popular strategies, such as setting improper gamma priors $\Gamma(\epsilon, \epsilon)$ $\epsilon \to 0$ widely used in many ARD works (which is equivalent to using type-2 ML for the α's if one uses variational methods [Bishop 2006]) and empirical Bayes using a single a and b (as well as other empirical Bayes variants). Neither worked well, the former being too sensitive to the value of ϵ, which is in agreement with the observations made by [Gelman 2006] and the latter constraining the model by using a single a and b.

4. RECURSIVE REGULARIZATION

In general, not all dependencies are given in the form of hierarchies (e.g., the dependencies among Wikipedia categories are given in the form of a graph). To our knowledge, there has been no work that addresses the problem of classification with graphical dependencies between class labels. The hierarchical Bayesian approach developed previously cannot be generalized to graphs, as there is no notion of parent or child. By resorting to a non-Bayesian risk minimization framework, we develop the first discriminative methods that can leverage both hierarchical and graphical dependencies between the class labels. The key idea is to incorporate the class label dependencies into the regularization structure of the parameters. Depending on the choice of the loss function, the framework can support both probabilistic classifiers such as logistic regression (RR-LR) and large-margin methods (RR-SVM).[4]

To formulate the regularization framework, we resort to the structural risk minimization framework, which prescribes choosing a prediction function to minimize a combination of the empirical risk on the training dataset and a regularization term to penalize the complexity of the function. In the context of hierarchies, the prediction function is parameterized by $\mathbf{W} = \{w_n : n \in \mathcal{N}\}$, and the empirical risk is defined to be the loss incurred by the instances at the leaf nodes of the hierarchy using a loss

[4]In our previous work [Gopal and Young 2013; Gopal et al. 2012], the methods were denoted by HR-{SVM,LR}. In this article, we will use the notation RR-{SVM,LR} for convenience.

function L. The parameters are estimated by minimizing:

$$\arg\min_{\mathbf{W}} \lambda(\mathbf{W}) + C \sum_{n\in T} \sum_{i=1}^{N} L(y_{in}, x_i, w_n) \quad \text{where } y_{in} = 2I(t_i = n) - 1, \tag{9}$$

where $\lambda(\mathbf{W})$ denotes the regularization term and C is a parameter that controls how much to fit to the training data. Note that unlike HBLR, each instance can belong to one or more leaf nodes in the hierarchy. We propose using the hierarchy in the learning process by incorporating a recursive structure into the regularization term for $\mathbf{W}$. More specifically, the regularization term is

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

This recursive form of regularization enforces the parameters of the node to be similar to the parameters of its parent under Euclidean norm. Intuitively, it models the hierarchical dependencies in the sense that it encourages parameters that are nearby in the hierarchy to be similar to each other. This helps classes to leverage information from nearby classes while estimating model parameters and helps share statistical strength across the hierarchy. In the case of graphs, the parameters of each node in the graph is regularized toward each of its neighbors (instead of its parent and children). Specifically, given a graph with a set of edges $E \subseteq \{(i, j) : i, j \in \mathcal{N}\}$, the regularization term is given by

$$\lambda(\mathbf{W}) = \sum_{(i,j)\in E} \frac{1}{2}||w_i - w_j||^2.$$

By defining L to be the hinge loss, we get a large-margin framework (RR-SVM), and by defining L to be the logistic loss, we get the probabilistic framework (RR-LR). The key advantage of RR-{SVM,LR} over other hierarchical models such as those of Tsochantaridis et al. [2006], Cai and Hofmann [2004], and Zou et al. [2011] is that there are no constraints that maximize 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, thereby scaling to very large problems.

4.1. Training RR Models

4.1.1. RR-SVM. The primary optimization problem in RR-SVM is to estimate the parameter $\mathbf{W}$:

$$\min_{\mathbf{W}} \sum_{n\in\mathcal{N}} \frac{1}{2}||w_n - w_{\pi(n)}||^2 + C \sum_{n\in\mathcal{T}} \sum_{i=1}^{N} \left(1 - y_{in} w_n^\top x_i\right)_+. \tag{10}$$

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 nondifferentiability in some of the updates (i.e., the updates at the leaf nodes), we converted these subproblems into their dual form that is differentiable and optimized it using coordinate descent. For each nonleaf node $n \notin T$, differentiating Equation (10) with respect to w_n yields a closed-form update for w_n that is given by

$$w_n = \frac{1}{|C_n| + 1} \left(w_{\pi(n)} + \sum_{c\in C_n} w_c \right). \tag{11}$$

ALGORITHM 1: Optimization of RR-SVM and RR-LR

```
Input : $\mathbf{D}$, $C$, $\pi$,T, $\mathcal{N}$
Result : weight vectors $\mathbf{W}^*$
While Not Converged
    For each $n \in \mathcal{N}$
        If $n \notin T$
            Update $w_n$ of the nonleaf node using Equation (11)
        Else
            If solving RR-SVM
              1. Solve the dual optimization problem (12)
              2. Update the primal parameters $w_n$ using Equation (13)
            Else If solving RR-LR
              1. Solve optimization problem (16) using LBFGS
            End
        End
    End
```

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

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

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

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_{in}y_{jn}x_i^\top x_j - \sum_{i=1}^{N}\alpha_i\big(1 - y_{in}w_{\pi(n)}^\top x_i\big). \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{12}$$

To solve this subproblem, one can easily use any second-order methods (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 $O(N^2)$ to be stored in memory. Typical large-scale hierarchical classification problems have at least hundreds of thousands of instances, and the memory required to store the kernel matrix is in the order of 100GB for each class, thereby rendering it impractical. Instead, we propose a coordinate descent approach that has minimal memory requirements and converges quickly even for large problems. Our work is based on the dual coordinate descent developed in Hsieh et al. [2008].

The core idea in coordinate descent is to iteratively update each dual variable. In the objective function Equation (12), 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:

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

Basically, we substituted α_i by $\alpha_i + d$ in Equation (12) and discarded all 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 preceding 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 $a' = \sum_{i=1}^{N} \alpha_i y_{in} x_i$ is an auxiliary 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 nonzero dimensions in x_i and the memory requirement for solving the entire subproblem is $O(N)$—far more efficient than that $O(N^2)$ compared to the second-order methods. Finally, after solving the dual subproblem, the update for w_n in the primal form can be derived from the KKT conditions for the following subproblem:

$$w_n = w_{\pi(n)} + \sum_{i=1}^{N} \alpha_i y_{in} x_i. \tag{13}$$

Note that the convergence of the preceding optimization method can be derived by viewing the procedure as a block coordinate descent scheme on a convex function where the blocks correspond to parameters at each node of the hierarchy [Lou and Tseng 1992; Tseng 2001].

For graphs, we employ a similar block coordinate descent algorithm by iteratively optimizing each parameter w_n. The optimization subproblem at node n is given by

$$\min_{w_n} \quad \frac{1}{2} \sum_{j:(n,j)\in E} ||w_n - w_j||^2 + C \sum_{i=1}^{N} \left(1 - y_{in} w_n^{\top} x_i\right)_+. \tag{14}$$

To derive the dual of the preceding subproblem, we first rewrite it by expanding the regularization term and discarding all terms that do not depend on w_n:

$$\min_{w_n} \quad \frac{1}{2} S_n ||w_n||^2 - S_n w_n^{\top} m + C \sum_{i=1}^{N} \left(1 - y_{in} w_n^{\top} x_i\right)_+,$$

where S_n denotes the number of neighbors for node n and m denotes the mean of neighbors $m = \frac{1}{S_n} \sum_{j:(n,j)\in E} w_j$. We now add a constant term $\frac{1}{2} S_n ||m||^2$ (constant with respect to this optimization subproblem) and divide by constant S_n and rewrite the optimization problem as

$$\min_{w_n} \quad \frac{1}{2} ||w_n - m||^2 + \frac{C}{S_n} \sum_{i=1}^{N} \left(1 - y_{in} w_n^{\top} x_i\right)_+. \tag{15}$$

Now we solve (15) using the same coordinate descent technique as described earlier and recover the optimal w_n.

4.1.2. RR-LR. We follow a similar iterative strategy for optimizing RR-LR—that is, we update the parameter w_n of each node n iteratively by fixing the rest of the parameters. Unlike RR-SVM, the objective function in RR-LR is convex and differentiable; therefore, we can directly use quasi-Newton methods such as LBFGS for each inner optimization.

The update for each nonleaf node is given in Equation (11). For each leaf node n, isolating the terms that depend on w_n, the objective and the corresponding gradient G can be written as

$$\min_{w_n} \quad \frac{1}{2}||w_n - w_{\pi(n)}||^2 + C\sum_{i=1}^{M} \log\left(1 + \exp\left(-y_{in}w_n^\top x_i\right)\right), \tag{16}$$

$$G = w_n - w_{\pi(n)} - C\sum_{i=1}^{M} \frac{1}{1+\exp(y_{in}w_n^\top x_i)} y_{in}x_i. \tag{17}$$

In the case of graphs, the only change is that instead of a single parent node, we subtract the difference from each neighbor:

$$G = \sum_{j:(n,j)\in E} (w_n - w_j) - C\sum_{i=1}^{M} \frac{1}{1+\exp(y_{in}w_n^\top x_i)} y_{in}x_i.$$

Note that although HBLR and RR-LR seem like two very different models, RR-LR can be viewed a successively simplified version of HBLR. If we replace the multiclass logistic function in HBLR with multiple binary logistic functions and have a single common fixed α for all nodes without any gamma prior, the HBLR model becomes identical to the RR-LR model—the only difference being the inference. Furthermore, if we estimate the parameters using the posterior mode (i.e., MAP estimate) instead of the posterior mean, even the training procedures become identical.

These connections also highlight the fundamental theme in all of our proposed HBLR, RR-SVM, and RR-LR models—that is, to enforce similarity between model parameters based on the hierarchical or graphical dependencies between the class labels. In HBLR models, we enforce it by propagating a Bayesian prior over the hierarchical structure, whereas in RR models, we enforce it via regularization of parameters over the hierarchical or graphical structure.

4.2. Parallelization

On hierarchies, we can parallelize the optimization of the parameters exactly as discussed in Section 3.3. For graphs, however, parallelization is a little tricky. The *ideal* parallelization on graphs involves finding the chromatic number of the graph—which is the smallest K such that each node of the graph is assigned one of K different colors from 1 to K and no two adjacent nodes have the same color. Once the nodes have been assigned colors, we can pick a color and working in parallel optimize the parameters of the nodes that have been assigned that color and repeat. However, finding the chromatic number of the graph is a NP-hard problem; therefore, we can only resort to approximate schemes to find the chromatic number. The degree of parallelization in graphs is given by $\frac{|\mathcal{N}|}{K}$, which is in contrast to $\frac{|\mathcal{N}|}{2}$ for hierarchies. Note that the minimum coloring needs to be solved only to achieve the best possible parallelization; in practice, any *reasonable* coloring achieves good parallelization.

Alternatively, one could also resort to other schemes, such as performing multiple iterations of optimizing all nodes in parallel (although convergence is not guaranteed in theory). Furthermore, to aid in convergence, we also used other tricks such as warm starting with the previously found dual solution and random permutation of subproblems in the coordinate descent method [Hsieh et al. 2008].

Table I. Dataset Statistics

Dataset	Training (#)	Testing (#)	Class Labels (#)	Leaf Labels (#)	Depth	Features (#)	Average Labels per Instance (#)
CLEF	10,000	1,006	87	63	4	89	1
NEWS20	11,260	7,505	27	20	3	53,975	1
RCV1	23,149	784,446	137	101	6	48,734	3.18
IPC	46,324	28,926	552	451	4	541,869	1
LSHTC-small	4,463	1,858	1,563	1,139	6	51,033	1
DMOZ-2010	128,710	34,880	15,358	12,294	6	381,580	1
DMOZ-2012	383,408	103,435	13,347	11,947	6	348,548	1
DMOZ-2011	394,756	104,263	35,448	27,875	6	594,158	1.03
SWIKI-2011	456,886	81,262	50,312	36,504	11	346,299	1.85
LWIKI	2,365,436	452,167	478,020	325,056	—	1,617,899	3.26

5. EXPERIMENTS

5.1. Datasets

We used several large-scale benchmark datasets whose statistics are listed in Table I. To maintain comparability with previously published evaluations, we used the conventional train-test splits wherever available.

(1) CLEF [Dimitrovski et al. 2008]: A hierarchical collection of medical x-ray images with EHD-diagram features. The dataset was normalized to have zero mean and unit variance to improve convergence.
(2) NEWS20:[5] A collection of newsgroup posts across 20 newsgroup categories.
(3) RCV1 [Lewis et al. 2004]: A collection of Reuters news articles from 1996 to 1997. We used the topic-based classification, as it has been the most popular in evaluations.
(4) IPC [IPC WIPO]: A collection of patents organized according to the International Patent Classification hierarchy.
(5) LSHTC-small, DMOZ-2010, DMOZ-2012 and DMOZ-2011: Multiple Web page collections released as a part of the Large-Scale Hierarchical Text Classification (LSHTC) evaluation during 2010–12.[6] It is essentially a subset of the Web pages from the Open Directory Project.[7]
(6) SWIKI-2011, LWIKI: Two subsets (small and large, respectively) of Wikipedia pages with human-generated topic class labels. The dependencies between the class labels in SWIKI-2011 are given as links in a directed acyclic graph, whereas in LWIKI they are given as links in a undirected graph.

Note that RCV1, DMOZ-2011, SWIKI-2011, and LWIKI are multilabel datasets, meaning that an instance may have multiple correct labels; the other datasets only have one correct label per instance. For the graph-based datasets, we introduced a dummy node between any two adjacent leaf-nodes to prevent leaf classes from directly becoming regularized toward each other (the leaf nodes in the case of graphs refers to nodes that have associated positive training examples).

5.2. Methods for Comparison

We include four types of methods for comparison:

—*HBLR models*: Our proposed hierarchical Bayesian model as described in Section 3. For ease of presentation, we mainly report the results of model M3 using partial

[5]http://people.csail.mit.edu/jrennie/20Newsgroups/.
[6]http://lshtc.iit.demokritos.gr/node/3.
[7]http://dmoz.org.

MAP inference (M3-map), as it seemed to perform the best in our experiments. In Section 6.5, we present a conclusive comparison of the different models and inference schemes.

—*RR models*: Our proposed RR methods (i.e., RR-SVM and RR-LR).

—*Flat baselines*: These include methods that cannot leverage class-label dependencies—one-versus-rest binary support vector machines (BSVM), one-versus-rest regularized binary logistic regression (BLR), multiclass support vector machines (MSVM), multiclass logistic regression (MLR).

—*Hierarchical baselines*: We choose four hierarchical methods with competitive performance:

(1) *corrMNL* [Shahbaba and Neal 2007]:[8] This method uses a Bayesian form of the multinomial logit model with a prior that introduces correlations between the parameters for classes that are nearby in the hierarchy. However, the model suffers from two important limitations. First, sensitive hyperparameters of the model need to be tweaked and manually set by the user. This is hard, especially if the user has no prior information. Second, the inference is performed using MCMC sampling and therefore cannot scale to a large number of classes. Due to this limitation, we report the results of this method only on the smallest CLEF dataset.

(2) *Hierarchical SVM* (HSVM) [Tsochantaridis et al. 2006]: This method is one of the most popular large-margin discriminative methods for structured output prediction. We use the specific instantiation of this framework developed for hierarchical classification. The model defines a hierarchical path–dependent discriminant function that minimizes a global hierarchical loss. More specifically, the parameters of the model are estimated by maximizing the margin between every pair of correct and incorrect labels with a penalty proportional to the hierarchical distance between the labels. The resulting optimization problem is solved by using a constraint generation framework that repeatedly adds the most violated constraint to the working set. Note that this method is not applicable on multilabel datasets.

(3) *Hierarchical orthogonal transfer* (OT) [Zou et al. 2011]: OT is large-margin method that encourages the classifiers at each node of the hierarchy to be different from the classifiers at its ancestors. Specifically, the regularization term is modified to enforce orthogonality between the parameters of parent and the children. The resulting optimization problem is solved using a regularized dual averaging method.

(4) *Top-down SVM* (TD) [Liu et al. 2005; Dumais and Chen 2000; Koller and Sahami 1997]: We employed the simplest variant of the top-down method using the SVM classifier. This is a popular baseline in several previous works.

We tuned the regularization parameter for all non-Bayesian methods using cross-validation with a range of values from 10^{-3} to 10^{3}. To make the baselines as competitive as possible on the multilabel datasets, we used an instance-based cut-off strategy as used in Gopal and Yang [2010]. This provided a better performance than using the usual cut-off of zero and other threshold methods like *rcut* or *scut* [Yang 2001]. Note that HSVM, OT, and HBLR (with soft-max logistic function) are inherently multiclass methods and are not applicable in multilabeled scenarios. To enable HBLR models to be applicable on multilabel datasets, we replace the soft-max function with multiple binary logistic functions and use the same instance-based cut-off strategy. The prior parameters for HBLR were set as discussed in Section 3.4 to enable faster convergence

[8]http://www.ics.uci.edu/babaks/Site/Codes.html.

on the larger datasets that we estimated $\hat{p}$ in Equation (8) using the parameters learned by BLR.

To scale up to the larger datasets (e.g., LWIKI, DMOZ-2011), for the HBLR and RR models we used the approximate parallelization as discussed in Sections 3.3 and 4.2. The parallelization for the flat one-versus-rest baselines (BSVM and BLR) is straightforward—that is, simply learn the models for all class labels in parallel. Among the hierarchical baselines, only TD can be easily parallelized, as the class models can be trained independently. It is not known how to parallelize the rest of the methods—MSVM, HSVM, and OT—and hence they cannot be scaled to the larger datasets. Therefore, we only report the results of these methods on the smaller datasets where they scaled.

Our experiments on the smaller datasets—CLEF, RCV1, NEWS20, IPC, and LSHTC-small—were conducted on a 32-core Intel Xeon X7560 at 2.27GHz and 32GB RAM. For the rest of the large-scale datasets, we used a Hadoop with 64 worker nodes having 8 cores and 16GB RAM each. Approximately 300 cores were used as mappers, and 220 cores were used as reducers.

In addition, we also included the best results in benchmark evaluations for comparison according to the numbers available on the LSHTC Web site.

5.3. Evaluation Metrics

We use the following standard evaluation metrics [Yang 1999] to measure the performance of all methods:

—Micro-F_1 is a conventional metric used to evaluate classification decisions [Yang 1999; Lewis et al. 1996]. Let TP_t, FP_t, FN_t denote the true positives, false positives, and false negatives for the class label $t \in T$. The microaveraged F_1 is

$$P = \frac{\Sigma_{t \in T} TP_t}{\Sigma_{t \in T} TP_t + FP_t}$$
$$R = \frac{\Sigma_{t \in T} TP_t}{\Sigma_{t \in T} TP_t + FN_t}$$
$$\textit{Micro-}F_1 = \frac{2PR}{P+R}.$$

—Macro-F_1 is also conventional metric used to evaluate classification decisions; unlike Micro-F_1, which gives equal weight to all instances in the averaging process, Macro-F_1 gives equal weight to each class label:

$$P_t = \frac{TP_t}{TP_t + FP_t}$$
$$R_t = \frac{TP_t}{TP_t + FN_t}$$
$$\textit{Macro-}F_1 = \frac{1}{|T|} \sum_{t \in T} \frac{2P_t R_t}{P_t + R_t}.$$

6. RESULTS

We present six sets of results:

(1) The first set of results compares our proposed methods with the flat baselines: BSVM, BLR, MSVM, and MLR.
(2) The second set of results compares our proposed methods with state-of-art hierarchical baselines: TD, HSVM, and OT.

(3) The third set of results analyzes the efficiency of the proposed methods against the flat and hierarchical baselines.
(4) The fourth set of results compares the performance of our proposed methods with the best results on the datasets used in the benchmark evaluations conducted by LSHTC.[9]
(5) The fifth set of results analyzes the performance of the different Bayesian HBLR models as well as compares the performance against the hierarchical Bayesian baseline: corrMNL.
(6) The sixth set of results analyzes the performance of our approach at various levels of the hierarchy and across classes with different training set sizes.

6.1. Comparison against Flat Baselines

Table II reports the results of our proposed models and several popular flat baselines: BSVM, MSVM, BLR, and MLR. The former two methods are based on hinge loss, whereas the latter two methods are based on the logistic loss. On all datasets except NEWS20, our proposed models perform better. To validate the results, we conducted pairwise significance tests between the RR and non-RR counterparts—that is, between SVM and RR-SVM and LR and RR-LR. We used the sign test for Micro-F_1 and the Wilcoxon rank test for Macro-F_1. We show the results on the CLEF, IPC, LSHTC-small, NEWS20, and RCV1 datasets. We were unable to conduct significance tests on the other datasets because we did not have access to class-wise performance scores and true-test labels—our reported evaluation measures on these datasets are from the output of an online evaluation system that does not reveal class-wise performance measures or true-test labels. The results of the significance tests on the five datasets show that the RR models significantly outperform the non-RR models on three out of the five tested datasets.

Comparing the HBLR and RR models, on the smaller datasets like CLEF, RCV1, NEWS20, and IPC, the HBLR model M3-map offers the best performance, whereas on the larger datasets like DMOZ and Wikipedia, RR-SVM gives the best performance. This observation suggests that the choice of loss function affects the performance; hinge loss seems to work better on the larger datasets (with high dimensions and skewed class label distributions), whereas logistic loss is suited to datasets with balanced class distributions. Between HBLR and RR-LR, HBLR works better because it is a more expressive model than RR-LR. Moreover, we also see some correlation between the performance of HBLR and RR-LR; whenever HBLR performs best, RR-LR is second best.

6.2. Comparison against Hierarchical Baselines

Table III compares the performance of our proposed methods against three hierarchical baselines: TD, HSVM, and OT. On all datasets, the proposed methods are able to leverage the hierarchy better and outperform existing hierarchical methods. In fact, on some datasets like LSHTC-small, there is a 16% relative improvement in performance. We conducted pairwise significance tests between the most scalable hierarchical baseline TD against our proposed methods: HBLR and RR. We used the setup as described in the previous section: the sign test for Micro-F_1 and the Wilcoxon rank test for Macro-F_1. All results are statistically significant.

6.3. Scalability Analysis

Table IV reports the training times taken for all methods. Among the flat baselines, the multiclass classifiers—MLR and MSVM—cannot even be scaled to datasets with

[9]http://lshtc.iit.demokritos.gr/lshtc2_evaluation, excluding our own own submissions to these evaluations.

Table II. Comparison against Flat Baselines: Macro-F_1 and Micro-F_1 on 10 Datasets

	BSVM	BLR	MSVM	MLR	RR-SVM	RR-LR	HBLR (M3-map)
CLEF							
Macro-F_1	48.59	53.26	54.33	54.76	53.92††	55.83†	**59.65**
Micro-F_1	77.53	79.92	80.02	80.52	80.02††	80.12†	**81.41**
RCV1							
Macro-F_1	54.72	53.38	NA	NA	56.56††	55.81†	**56.08**
Micro-F_1	80.82	80.08			81.66††	81.23††	**81.98**
NEWS20							
Macro-F_1	**82.32**	82.17	81.73	81.82	82.00	82.06	81.69
Micro-F_1	**83.10**	82.97	82.47	82.56	82.78	82.86	82.56
IPC							
Macro-F_1	45.71	48.29	NS	NS	47.89†	49.60	**51.06**
Micro-F_1	53.12	55.03			54.26††	55.37†	**56.02**
LSHTC-small							
Macro-F_1	28.62	28.12	28.34	28.38	28.94	28.48	**30.81**
Micro-F_1	45.21	44.94	45.62	45.20	45.31	45.11	**46.03**
	No significance tests reported on the following datasets due to lack of test labels						
DMOZ-2010							
Macro-F_1	32.64	31.58	NS	NS	**33.12**	32.42	31.88
Micro-F_1	45.36	45.40			**46.02**	45.84	45.64
DMOZ-2012							
Macro-F_1	31.59	14.18	NS	NS	**33.05**	20.04	27.19
Micro-F_1	56.44	52.79			**57.17**	53.18	53.15
DMOZ-2011							
Macro-F_1	24.34	21.67	NA	NA	**25.69**	23.90	23.46
Micro-F_1	42.88	41.29			**43.73**	42.27	41.35
SWIKI-2011							
Macro-F_1	26.57	19.51	NA	NA	**28.72**	24.26	NA
Micro-F_1	40.87	37.65			**41.79**	40.99	
LWIKI							
Macro-F_1	19.89	18.65	NA	NA	**22.31**	20.22	NA
Micro-F_1	37.66	36.96			**38.08**	37.67	

Note: Boldfaced numbers indicate the best-performing method. NS denotes that the method is not scalable to the dataset; NA denotes that the method is not applicable since the dataset is multilabeled or has graph-based dependencies. The significance test results between RR-SVM and BSVM and RR-LR and BLR on the first five datasets are denoted with a dagger (†) for a p-value less than 5% and with a double dagger (††) for a p-value less than 1%.

a moderate number of class labels. The one-versus-rest baselines on the other hand are very scalable and faster than our proposed models. BSVM is on average 1.92 times faster than RR-SVM, BLR is on average 2.87 times faster than RR-LR, and BLR is on average 4.89 times faster than M3-map. This is not surprising—the better performance of our models comes at the cost of increased computational time. However, even on the largest dataset LWIKI, RR-SVM takes about 37 hours; although slower than BSVM, the computation time certainly falls within the tractable range.

Among the hierarchical baselines, TD is the only one that can be scaled to the larger datasets, although with a low performance. HSVM and OT could only scale to the smallest datasets (CLEF and LSHTC-small), and both of them are orders of magnitude slower than our proposed methods. On the rest of the datasets, neither of them could be tested successfully either due to scalability issues or modeling (inability to handle multilabel data or graph-based dependencies) issues.

Table III. Comparison against Hierarchical Baselines: Macro-F_1 and Micro-F_1 on 10 Datasets

	TD	HSVM	OT	RR-SVM	RR-LR	HBLR (M3-map)
CLEF						
Macro-F_1	32.32	57.23	37.12	53.92††	55.83††	**59.65**††
Micro-F_1	70.11	79.72	73.84	80.02††	80.12††	**81.41**††
RCV1						
Macro-F_1	34.15	NA	NA	56.56††	55.81††	**56.08**††
Micro-F_1	71.34			81.66††	81.23††	**81.98**††
NEWS20						
Macro-F_1	80.86	80.04	81.20	82.00††	**82.06**††	81.69††
Micro-F_1	81.20	80.79	81.98†	82.78††	**82.86**††	82.56††
IPC						
Macro-F_1	42.62	NS	NS	47.89††	49.60††	**51.06**††
Micro-F_1	50.34			54.26††	55.37††	**56.02**††
LSHTC-small						
Macro-F_1	20.01	21.95	19.45	28.94††	28.48††	**30.81**††
Micro-F_1	38.48	39.66	37.12	45.31††	45.11††	**46.03**††
	No significance tests reported on the following datasets due to lack of test labels					
DMOZ-2010						
Macro-F_1	22.30	NS	NS	**33.12**	32.42	31.88
Micro-F_1	38.64			**46.02**	45.84	45.64
DMOZ-2012						
Macro-F_1	30.01	NS	NS	**33.05**	20.04	27.19
Micro-F_1	55.14			**57.17**	53.18	53.15
DMOZ-2011						
Macro-F_1	21.07	NA	NA	25.69	23.90	23.46
Micro-F_1	35.91			**43.73**	42.27	41.35
SWIKI-2011						
Macro-F_1	17.39	NA	NA	**28.72**	24.26	NA
Micro-F_1	36.65			**41.79**	40.99	

Note: Boldfaced numbers indicate the best-performing method. NS denotes that the method is not scalable to the dataset; NA denotes that the method is not applicable because the dataset is multilabeled or has graph-based dependencies. The significance test results are between RR-SVM, RR-LR, and HBLR against the scalable hierarchical baseline TD on the first five datasets. A double dagger (††) denotes significance at the 1% level.

Table IV. Computational Efficiency: The Training Time (in Minutes) for All Methods

	Flat baselines				Hierarchical Baselines			Proposed methods		
	BSVM	BLR	MSVM	MLR	TD	HSVM	OT	RR-SVM	RR-LR	HBLR (M3-map)
CLEF	.15	.24	.20	1.92	.13	3.19	1.31	.42	1.02	3.05
RCV1	.273	2.89	NA	NA	.213	NA	NA	.55	11.74	12.11
NEWS20	.04	.11	.07	.352	.04	2.31	.98	.14	.52	1.06
IPC	3.12	4.17	NS	NS	2.21	NS	NS	6.81	15.91	31.2
LSHTC-small	.31	1.93	1.65	3.63	.11	289.60	132.34	.52	3.73	5.2
DMOZ-2010	5.12	97.24	NA	NA	3.97	NS	NS	8.23	123.22	151.67
DMOZ-2012	22.31	95.38	NA	NA	12.49	NS	NS	36.66	229.73	331.96
DMOZ-2011	39.12	101.26	NA	NA	16.39	NA	NA	58.31	248.07	224.33
SWIKI-2011	54	99.46	NA	NA	21.34	NA	NA	89.23	296.87	NA
LWIKI	1114.23	2134.46	NA	NA	NA	NA	NA	2230.54	2134.46	NA

Table V. Comparison of Established Benchmark Results Macro-F_1 and Micro-F_1 to Those Established by the LSHTC[7]

	LSHTC Published Results	RR-SVM	RR-LR	HBLR (M3-map)
DMOZ-2010				
Macro-F_1	**34.12**	33.12	32.42	31.88
Micro-F_1	**46.76**	46.02	45.84	45.64
DMOZ-2012				
Macro-F_1	31.36	**33.05**	20.04	27.19
Micro-F_1	51.98	**57.17**	53.18	53.15
DMOZ-2011				
Macro-F_1	**26.48**	25.69	23.90	23.46
Micro-F_1	38.85	**43.73**	42.27	41.35
SWIKI-2011				
Macro-F_1	23.16	**28.72**	24.26	NA
Micro-F_1	37.39	**41.79**	40.99	
LWIKI				
Macro-F_1	18.68	**22.31**	20.22	NA
Micro-F_1	34.67	**38.08**	37.67	

Note: We excluded our own submission of the system. Boldfaced numbers indicate the best-performing method. NA denotes that the method is not applicable on the data set due to graph-based dependencies between class and labels.

Between HBLR and RR-LR, there is an efficiency versus effectiveness trade-off. Although HBLR achieves a better performance than RR-LR with a 7% and .5% improvement in Macro-F_1 and Micro-F_1, it is 1.62 times slower than RR-LR. The choice of the method depends on the user's preference between performance and scalability.

6.4. Comparison Against Established Benchmark Results

Table V compares the results of our proposed models with the well-established results on the large-scale datasets released by the LSHTC community. We focus on only those datasets for which benchmark evaluations were available on the Web site.[7] Table V shows that the our proposed models are able to perform better than the state-of-the-art results reported so far on most of these datasets. In fact, on four out of the five datasets, RR-SVM shows a consistent 10% relative improvement from the currently published results.

6.5. Comparison of HBLR Models and corrMNL

First, we analyze the performance our HBLR models: M1, M2, and M3. Table VI shows the results of the different models on four datasets using partial MAP inference. The performance of M3 seems to be consistently better than M1, followed by M2. Although M2 is more expressive than M1, the benefit of the expressiveness seems to be offset by the difficulty in learning a larger number of parameters.

Next, we analyze the performance of the HBLR models against the MCMC-based Bayesian hierarchical baseline: corrMNL [Shahbaba and Neal 2007]. For corrMNL, we performed sampling for 2,500 iterations with 1,000 for burn-in. We present the result of our models using full variational inference {M1,M2,M3}-var as well as partial MAP inference {M1,M2,M3}-map. With regard to scalability, partial MAP inference is orders of magnitude faster than corrMNL (750 times), followed by full variational inference (20 times). In terms of performance, M3-var offers the best performance, followed by M3-map. MAP inference has only a small drop in performance compared to full variational inference while being 27 times faster than full variational inference.

Table VI. Comparison of HBLR Models using Partial MAP Inference: Macro-F_1 and Micro-F_1 and the Three Models—M1, M2, and M3

	M1-map	M2-map	M3-map
CLEF			
Macro-F_1	55.53	54.76	**59.65**
Micro-F_1	80.88	80.25	**81.41**
NEWS20			
Macro-F_1	81.54	80.91	**81.69**
Micro-F_1	82.24	81.54	**82.56**
LSHTC-small			
Macro-F_1	28.32	24.93	**28.76**
Micro-F_1	43.98	43.11	**44.05**
IPC			
Macro-F_1	50.43	47.45	**51.06**
Micro-F_1	55.80	54.22	**56.02**

Note: Boldfaced numbers indicate the best-performing model.

Table VII. Comparison with corrMNL: Macro-F_1 and Micro-F_1 of the HBLR Models and corrMNL of the CLEF Dataset

		{M1,M2,M3}-var			{M1,M2,M3}-map			{M1,M2,M3}-flat		
	CorrMNL	M1	M2	M3	M1	M2	M3	M1	M2	M3
Macro-f1	55.59	56.67	51.23	**59.67**	55.53	54.76	59.65	52.13	48.78	55.23
Micro-f1	81.10	81.21	79.92	**81.61**	80.88	80.25	81.41	79.82	77.83	80.52
Time (mins)	2279	79	81	80	3	3	3	3	3	3

Table VIII. The Macro-F_1 of BSVM and RR-SVM at Various Levels of the Hierarchy along with the Number of Nodes at Each Level

Level	BSVM	RR-SVM		# Nodes
1	100.0	100.0	(0.00%)	1
2	71.43	**72.50**	(1.50%)	11
3	41.62	**42.93**	(3.12%)	342
4	25.71	**26.73**	(3.93%)	3376
5	26.54	**27.27**	(3.03%)	8858
6	13.06	**16.72**	(28.08%)	549

Note: The improvement in percentage of RR-SVM over BSVM is shown in parentheses.

6.6. Detailed Analysis

We present detailed analysis of the results of our approach on two different fronts: (1) the performance improvement at various levels of the hierarchy and (2) the performance improvement on the classes with fewer training examples. Due to the lack of test set labels (see Section 6.1), we partitioned the training data from one of the datasets (DMOZ-2012) with a 50%–50% train-test split.

Table VIII reports the macro-averaged F_1-score at various levels of the hierarchy. On all levels, our proposed RR-SVM performs better than SVM. The improvement seems to be particularly high at the lower levels (4, 5, and 6) where the leaf nodes are located.

Table IX reports the macro-averaged F_1-score for classes with different training set sizes. The improvement is highest in classes with a moderately small number of training examples (6 to 50 training examples). The improvement seems to be smaller when the number of training examples is too low (1 to 5) or higher (>100).This is expected because in the former case, learning a good classifier is generally very hard,

Table IX. The Macro-F_1 of BSVM and RR-SVM across Classes with Different Training Set Sizes

# Examples	BSVM	RR-SVM		# Classes
1–5	20.24	**20.62**	(1.86%)	7952
6–10	24.84	**26.59**	(6.57%)	1476
11–20	32.16	**34.08**	(5.63%)	1089
21–50	39.94	**41.23**	(3.12%)	680
51–100	49.25	**50.26**	(1.99%)	262
>100	59.81	**60.64**	(1.38%)	278

Note: The number of classes in each range of training set size is shown. The improvement in percentage of RR-SVM over BSVM is shown in parentheses.

and in the latter case, the training examples are already adequate and the hierarchy does not help further.

7. CONCLUSION

In this article, we proposed two frameworks for large-scale classification—HBLR and RR—that can leverage hierarchical and graphical dependencies between class labels for improving classification. The proposed methods rely on enforcing similarity between the model parameters based on the proximity of the classes in the provided structure. For scalability, we developed fast hierarchical variational inference algorithms and efficient training procedures for both frameworks. Our proposed models achieved state-of-the-art results on multiple benchmark datasets and showed a consistent improvement in performance over both flat and other hierarchical methods. For future work, we will explore the possibility of incrementally training the system as new training examples are provided for the rarer classes.

ACKNOWLEDGMENTS

We thank Alexandru Niculescu-Mizil for useful discussions about model design and parallelization, Guy Blelloch for sharing his computational resources, and the Open Cloud cluster for the Hadoop framework.

REFERENCES

P. N. Bennett and N. Nguyen. 2009. Refined experts: improving classification in large taxonomies. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 11–18.

C. M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

C. M. Bishop and M. E. Tipping. 2003. Bayesian regression and classification. In *Advances in Learning Theory: Methods, Models, and Applications*. IOS Press, 267–285.

G. Bouchard. 2007. Efficient bounds for the softmax function and applications to inference in hybrid models. In *Proceedings of the NIPS 2007 Workshop for Approximate Bayesian Inference in Continuous/Hybrid Systems*.

L. Cai and T. Hofmann. 2004. Hierarchical document categorization with support vector machines. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*. 78–87.

G. Casella. 1988. *Empirical Bayes Method—A Tutorial*. Technical Report #88-18. Cornell University and Purdue University.

N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. 2006. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research* 7, 31–54.

C. DeCoro, Z. Barutcuoglu, and R. Fiebrink. 2007. Bayesian aggregation for hierarchical genre classification. In *Proceedings of the International Conference on Music Information Retrieval*. 77–80.

O. Dekel, J. Keshet, and Y. Singer. 2004. Large margin hierarchical classification. In *Proceedings of the 21st International Conference on Machine Learning*. 27.

I. Dimitrovski, D. Kocev, L. Suzana, and S. Džeroski. 2008. Hierchical annotation of medical images. *Pattern Recognition* 44, 10–11, 2436–2449.

S. Dumais and H. Chen. 2000. Hierarchical classification of Web content. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 256–263.

A. Gelman. 2006. Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis* 1, 3, 515–533.

S. Gopal and Y. Yang. 2010. Multilabel classification with meta-level features. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 315–322.

S. Gopal and Y. Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 257–265.

S. Gopal, Y. Yang, B. Bai, and A. Niculescu-Mizil. 2012. Bayesian models for large-scale hierarchical classification. In *Advances in Neural Information Processing Systems* 25, 2420–2428.

S. Gopal, Y. Yang, and A. Niculescu-Mizil. 2012. A regularization framework for large-scale hierarchical classification. In *Proceedings of the Workshop on Large-Scale Hierarchical Text Classification.*

N. Gornitz, C. K. Widmer, G. Zeller, A. Kahles, G. Ratsch, and S. Sonnenburg. 2011. Hierarchical multitask structured output learning for large-scale sequence segmentation. In *Advances in Neural Information Processing Systems*. 2690–2698.

C. J. Hsieh, K. W. Chang, C. J. Lin, S. S. Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th International Conference on Machine Learning*. 408–415.

R. E. Kass and R. Natarajan. 2006. A default conjugate prior for variance components in generalized linear mixed models. *Bayesian Analysis* 1, 3, 535–542.

D. Koller and M. Sahami. 1997. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning*. 170–178.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. 2004. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397.

D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. 1996. Training algorithms for linear text classifiers. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 298–306.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 45, 1, 503–528.

T. Y. Liu, Y. Yang, H. Wan, H. J. Zeng, Z. Chen, and W. Y. Ma. 2005. Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD Explorations Newsletter* 7, 1, 36–43.

Z. Q. Luo and P. Tseng. 1992. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications* 72, 1, 7–35.

A. McCallum, R. Rosenfeld, T. Mitchell, and A. Y. Ng. 1998. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the 15th International Conference on Machine Learning*. 359–367.

J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. 2006. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research* 7, 1601–1626.

B. Shahbaba and R. M. Neal. 2007. Improving classification when a class hierarchy is available using a hierarchy-based prior. *Bayesian Analysis* 2, 1, 221–238.

M. E. Tipping. 2001. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* 1, 211–244.

P. Tseng. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications* 109, 3, 475–494.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2006. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research* 6, 2, 1453.

C. Widmer, J. Leiva, Y. Altun, and G. Rätsch. 2010. Leveraging sequence classification by taxonomy-based multitask learning. In *Research in Computational Molecular Biology*. Springer, 522–534.

IPC WIPO. Download and IT Support Area. Retrieved December 28, 2014, from http://www.wipo.int/classifications/ipc/en/ITsupport.

G. R. Xue, D. Xing, Q. Yang, and Y. Yu. 2008. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st Annual International ACMSIGIR Conference on Research and Development in Information Retrieval*. 619–626.

Y. Yang. 2001. A study of thresholding strategies for text categorization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 137–145.

Y. Yang, J. Zhang, and B. Kisiel. 2003. A scalability analysis of classifiers in text categorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 96–103.

Y. Yang. 1999. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval* 1, 67–88.

D. Zhou, L. Xiao, and M. Wu. 2011. *Hierarchical Classification via Orthogonal Transfer*. Technical Report MSR-TR-2011-54. Microsoft Research, Redmond, WA.

Received October 2013; revised Februrary 2014; accepted April 2014