Recursive Regularization for Large-scale Classification with Hierarchical and Graphical Dependencies*

Siddharth Gopal
Carnegie Mellon University
sgopal1@andrew.cmu.edu

Yiming Yang
Carnegie Mellon University
yiming@cs.cmu.edu

ABSTRACT

The two key challenges in hierarchical classification are to leverage the hierarchical dependencies between the class-labels for improving performance, and, at the same time maintaining *scalability* across large hierarchies. In this paper we propose a regularization framework for large-scale hierarchical classification that addresses both the problems. Specifically, we incorporate the hierarchical dependencies between the class-labels into the regularization structure of the parameters thereby encouraging classes nearby in the hierarchy to share similar model parameters. Furthermore, we extend our approach to scenarios where the dependencies between the class-labels are encoded in the form of a graph rather than a hierarchy. To enable large-scale training, we develop a parallel-iterative optimization scheme that can handle datasets with hundreds of thousands of classes and millions of instances and learning *terabytes* of parameters. Our experiments showed a consistent improvement over other competing approaches and achieved state-of-the-art results on benchmark datasets.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology; Classifier Design and Evaluation; H.4 [**Information Systems**]: General

General Terms

Algorithms,Design,Experimentation

Keywords

Hierarchical Classification, Recursive Regularization, Parallel Optimization, Large-scale Evaluation

*A preliminary version of this paper was presented to the Large-scale Hierarchical Text Classification Workshop, ECML 2012

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD'13, August 11–14, 2013, Chicago, Illinois, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
Copyright 2013 ACM 978-1-4503-2174-7/13/08 ...$15.00.

1. INTRODUCTION

Hierarchies provide a natural way to browse and organize unstructured data at multiple levels of granularity. The large taxonomies for Web Page categorization at Yahoo! Directory and the Open Directory Project, the International Patent Classification Hierarchy for patents are examples of such widely used hierarchies. Firstly, unlike binary classification, we need to address how to use the hierarchy. Secondly and importantly, we need to develop scalable methods than can handle large-scale problems, as most real world Hierarchical Classification (HC) are characterized by large hierarchies and training set sizes. For example, consider the Open Directory Project [1], one of the largest human edited hierarchy of the entire Web containing more than 4.4 million webpages categorized into a hierarchy of 766,930 class-labels with more than 10 levels of depth. Developing techniques that leverage hierarchical dependencies between these hundreds of thousands of class-labels as well as scaling to the millions of training instances is a non-trivial task.

A primary problem in HC is the data-sparsity issue for majority of class-labels [22],[36],[5]. For example, 76% of the class-labels in the Yahoo! Directory have less than 5 positive instances [22] and 72% of the Open Directory Project have less than 4 positive instances [1]. Learning independent models (one per class-label) with such limited training examples might lead to poor performance due to over-fitting. This raises one of the important questions in HC research, i.e., Can we develop solutions that can use the positive training examples from other classes based on the hierarchical dependencies between them? and can we do this both effectively and efficiently for large-scale hierarchical classification (LSHC)?

Previous works have often concentrated on different parts of the problem. For instance, the most popular in the early stage of LSHC research are the "pachinko-machine models" [11], [36] [22], [18] where the classification task is decomposed into sub-tasks 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 [5], a cascading strategy is employed to add the output of lower-level classifiers as additional features for higher-level classifiers. In [8], a Bayesian aggregation on the results of the individual binary

[1] http://www.dmoz.org

classifiers was proposed. In [34], a data-driven pruning strategy is proposed for reducing the size of the original hierarchy. Some improvements over the results of the pachinko-machine models have been reported; however, those approaches are heuristic by nature.

The more principled methods include the large-margin models by [31],[6],[26],[9],[7] 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 [40] where orthogonality conditions are imposed between the parent and children classifiers, in [27] [16] with Bayesian multinomial logistic models, and in [24] using Naive Bayes classifiers with hierarchical shrinkage. Empirical improvements of most of these methods over simpler approaches have been shown on small datasets, typically with hundreds (or less) of class-labels and thousands of instances. The difficulty for *most* of these methods in scaling is due to the high-degree of inter-dependencies among model parameters and the parameters for all the classes cannot be held in memory at the same time.

What we want is an approach that is both principled and at the same time computationally tractable for problems with a very large number of classes, high-dimensional features and large volume of data. Further, we want the approach to be generally applicable to both large-margin classifiers (like Support Vector Machines) as well as probabilistic classifiers (like logistic regression), and finally, to be flexible for leveraging both hierarchical dependencies as well as graph-based dependencies among class labels. We propose such an approach in this paper, namely the recursive regularization framework for Support Vector Machines (SVM) and logistic regression (LR). It uses the dependencies among classes and sub-classes to define a joint objective for regularization of model parameters; the model parameters of the siblings nodes who share the same parent are regularized towards the common parent node. Intuitively, it is based on the assumption that the nearby classes in the hierarchy are semantically close to each other and hence share similar model parameters. Notice that this model is simpler than the fully Bayesian models in [27] and [16] where the dependencies are modeled in richer forms, controlling both the means and covariances in Gaussain models. The simplicity of the proposed approach makes it easier to scale than the fully Bayesian approaches.

For the scalability of our method, we develop a parallel and iterative co-ordinate descent scheme that can easily tackle the large datasets with millions of instances and hundreds of thousands of classes. The key idea here is to formulate the co-ordinate descent objective function in a way that the dependencies between the various parameters can be easily localized, and the computations for all the local regions can be carried out in parallel. More specifically, we formulate the objective such that the parameters at each node is independent of the rest of the hierarchy given the parameters of its parent and children; therefore by fixing the parameters of the parent and children, the node can be optimized independently from the rest of the hierarchy. This aids in achieving a large degree of parallelization and gets a speedup close to linear in the number of processors used. Moreover, the local computations at leaf nodes are dualized in order side-step non-differentiability issues when using loss functions such as Hinge loss. As we shall see in section 4, this combination of using iterative parallelization of local computations and fast dual co-ordinate descent methods for each local computation leads to optimization schemes that can easily scale to real life web-scale data.

We tested our proposed approaches in terms effectiveness as well efficiency by conducting evaluations against other state-of-the-art methods on nine benchmark datasets for LSHC problems, including the large-scale datasets from the Large-scale Hierarchical Text Classification Challenge [2] . With respect to effectiveness, we found that our methods outperformed all the other methods on most tested datasets. With respect to efficiency, 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 300,000 classes and 2 Million training instances in a matter of 37 hours.

To summarize, the contribution of our work is multifold,

1. We propose a regularization framework that goes beyond simple one-versus-rest binary classification schemes such as [18], [11], [36], and uses the hierarchical or graphical dependencies among class labels and that is applicable to both large-margin classifiers (SVM's) and probabilistic classifiers (LR's).
2. We propose a fast iterative co-ordinate descent scheme with appropriate local dualized computations that side-steps non-differentiability issues. This method is highly parallelizable and achieves speed-ups close to linear in the number of processors.
3. Our proposed methods achieve state-of-the-art results on multiple datasets which are orders of magnitudes ($\sim$ 1000x) larger than what the most existing methods have been scaled to [31],[6],[26],[9],[7], [32].

Indirectly related to our paper are a few works in multitask learning [12], [2],[3], [32] where regularization was used as a tool to share information between tasks. However, their focus is not scalability and their techniques cannot be directly applied to *large scale* HC. Other works include regularization on graphs such as [39], [29], but the focus is on graphical dependencies between the instances and not between class-labels.

2. 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 $x_i \in \mathcal{X}$ and $t_i \in T$ is a label, $T \subset \mathcal{N}$ is the set of all leaf nodes in the hierarchy. We assume that each instance is labeled to one or more leaf nodes in the hierarchy. If there are any instances assigned to an internal node, spawn a leaf-node under it and re-assign all the instances from the internal node to this new leaf node. For convenience, let C_n denote the set of all children of node n, and binary variable $y_{in} \in \{+1, -1\}$ denote if x_i belongs to class $n \in T$ i.e. $y_{in} = (2I(t_i = n) - 1)$. The problem of HC is to learn a prediction function $f : \mathcal{X} \to T$ 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 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

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

expected value of a loss function over all possible inputs. The Structural Risk Minimization framework 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

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

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 instances and the complexity of f.

In the problem of HC, the prediction function is parameterized by a set of parameters $\mathbf{W} = \{w_n : n \in \mathcal{N}\}$ i.e each node n in the hierarchy is associated with a parameter vector w_n. First, we define the Empirical Risk in our model as the loss incurred by the instances 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, in principle, be any convex loss function. Second, we propose to use the hierarchy in the learning process by incorporating a recursive structure into the regularization term for $\mathbf{W}$. Specifically, we propose the following form of regularization

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

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 which 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. We hope that this would especially enable classes with very few training instances to pool in information, as well as gain 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,

HR-SVM

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

HR-LR

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

The key advantage of HR-{SVM,LR} over other hierarchical models such as [31], [6], [4], [40] 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 3.3) thereby scaling to very large HC problems.

Input : **D**, C, π,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**
        1. Solve dual in eq (5)
        2. Update w_n using eq (7)
      end
      if *HR-LR* **then**
        1. Solve eq (8) using LBFGS
      end
    end
  end
end

Algorithm 1: Optimization of HR-SVM and HR-LR

3. OPTIMIZATION ALGORITHM

3.1 HR-SVM

Although the objective function in (2) is convex and has a unique maximum, it is not differentiable and straightforward methods such as gradient descent cannot be used. To address the non-differentiability, many works have generally resorted to subgradient techniques such as [40], [28], [1]. However, the general problem with subgradient approaches is that the learning rate of the optimization routine needs to be specified [17]. 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 coordinate descent. This iterative scheme offers several advantages compared to standard approaches,

1. There are no learning rates that need to be 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,

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

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,

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

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

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

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

To solve this subproblem, one can 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 at least 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 [17].

The core idea in co-ordinate descent is to iteratively update each dual variable. In the 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,

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

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 $a' = \sum_{i=1}^{M}\alpha_i y_{in}x_i$ 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. Note that the convergence of the above optimization method can be derived by viewing the procedure as a block co-ordinate descent scheme on a convex function where the blocks corresponds to parameters at each node of the hierarchy [23], [30].

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, they take a much longer time to converge [25] . Even in simpler cases, identifying the rate of convergence is hard and the stopping criteria might not be accurate [23]. However, in the case of HC text classification, the documents are 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 text classification [13], [38].

3.2 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) [21].

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

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

$$G = w_n - w_{\pi(n)} - C\sum_{i=1}^{M}\frac{1}{1 + \exp(y_{in}w_n^\top x_i)}y_{in}x_i \tag{9}$$

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

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 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. Following our previous work [16], we iteratively optimize the odd and even levels in the hierarchy - 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

Table 1: Dataset Statistics

Dataset	#Training	#Testing	#Class-Labels	#Leaf-labels	Depth	#Features	Avg #labels per instance
CLEF	10,000	1,006	87	63	4	89	1
RCV1	23,149	78,446	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	614,428	325,056	-	1,617,899	3.26

parameters. To aid in convergence, we also used other tricks such as warm-starting with the previously found solution and random permutation of subproblems in co-ordinate descent method [17]. For large hierarchies, this method yields a speedup almost linear in the number of processors. Note that for HR-SVM, since only some of the dual variables (α's) are non-zero, we can reduce the storage requirements by representing w_n by the corresponding non-zero dual variables instead of a full vector.

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.

3.4 Extension to Graphs

In several real life scenarios, the dependencies between the class-labels are given in the form of a graph rather than a hierarchy. For instance, in Wikipedia where thousands of people edit pages at the same time, it is common to see editors quickly linking a topic of interest with other related topics. To enable our approach to be applicable in such scenarios, we extend our method to be able to handle graph based dependencies.

The regularization paradigm in the case of graphs can be stated as follows - the parameters of each node in the graph is regularized towards each of its neighbours (instead of its parent and children). Specifically, given a graph with a set of edges $E = \{(i,j) : i,j \in \mathcal{N}\}$, the regularization term is given by ,

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

The optimization algorithms for HR-SVM and HR-LR can be derived similarly. In the interest of space, we just present a brief outline. For those nodes which do not have any training instances, the update is given by,

$$w_n = \frac{1}{S_n} \sum_{j:(n,j)\in E} w_j$$

where S_n denotes the number of neighbours of node n. For the nodes which have associated training instances; in HR-SVM, the optimization objective at node n is given by

$$\frac{1}{2} \sum_{j:(n,j)\in E} ||w_n - w_j||^2 + C \sum_{i=1}^{M} (1 - y_{in} w_n^\top x_i)_+ \qquad (10)$$

To derive the dual, we re-write the above objective using the mean of the neighbouring parameters. Define the mean $m = \frac{1}{S_n} \sum_{j:(n,j)\in E} w_j$. The new objective is given by

$$\frac{1}{2}||w_n - m||^2 + \frac{C}{S_n} \sum_{i=1}^{M} (1 - y_{in} w_n^\top x_i)_+ \qquad (11)$$

It can be easily shown that the optimal solution for objective (11) is the same as the optimal solution for (10) by expanding the regularization term and discarding constant terms in the optimization. Now (11) can be solved using the same co-ordinate descent technique as described earlier.

For HR-LR, we have a similar optimization of the subproblem, the only difference is that the gradient in eq (9) would now have a summation over all the neighbours instead its parent and children.

Parallelization: The parallelization scheme for hierarchies cannot be straight-forwardly extended to graphs as there is no notion of odd or even levels. The **best possible** 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 iterate over the K different colors and parallely optimize the parameters of the nodes which have been assigned that color. However, finding the chromatic number of the graph is a NP-complete 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. 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). A complete discussion of the various approximate parallelization schemes on graphs is however beyond the scope of this paper. For related issues, refer [14] which discuss parallelizing belief propagation on graphs.

4. EXPERIMENTAL DESIGN

4.1 Datasets

We used several large-scale benchmark datasets whose statistics are listed in Table 1. To maintain comparability with previously published evaluations, we used the conventional train-test splits where-ever available.

- **CLEF** [10] A hierarchical collection of medical X-ray images with EHD-diagram features.
- **RCV1** [20] A collection of Reuters News from 1996-1997. We used the topic-based classification as it has been most popular in evaluations.

- **IPC** [33] A collection of patents organized according to the International Patent Classification Hierarchy.
- **LSHTC-small**, **DMOZ-2010**, **DMOZ-2012** and **DMOZ-2011** Multiple web-page collections released as a part of the LSHTC (Large-Scale Hierarchical Text Classification) evaluation during 2010-12 [3]. It is essentially a subset of the web pages from the Open Directory Project.
- **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 while in LWIKI they are given as links in a undirected graph.

Note that RCV1, DMOZ-2011, SWIKI-2011, LWIKI are multi-label datasets, meaning that an instance may have multiple correct labels; the other datasets only have one correct label per instance.

4.2 Methods for Comparison

We include three categories of methods for comparison:

- **HR models**: Our proposed methods, i.e., HR-SVM and HR-LR;
- **Flat baselines**: One-versus-rest binary Support Vector Machines (SVM) and one-versus-rest regularized logistic regression (LR), as conventional flat classifiers;
- **Hierarchical baselines**: We choose 4 hierarchical methods with competitive performance:
  - a **Hierarchical SVM** (HSVM) [31] a large-margin discriminative method with path dependent discriminant function;
  - b **Hierarchical Orthogonal Transfer** (OT) [40], a large-margin method enforcing orthogonality between the parent and the children;
  - c **Top-down SVM** (TD)[22], [11], [18] a top-down pachinko-machine style support vector machine; a popular baseline in several previous works.
  - d **Hierarchical Bayesian Logistic Regression** (HBLR), [16], our recent work using a fully Bayesian hierarchical model, which computationally more costly than HR-LR, and also not applicable on datasets with graph-based dependencies.

We implemented all the above methods, and tuned the regularization parameter using cross-validation with a range of values from 10^{-3} to 10^{3}. On the multi-label datasets, to make the baselines as competitive as possible, we used an instance-based cut-off strategy as used in [15]. This provided a better performance than using the usual cut-off of zero as well as other threshold methods like *rcut* or *scut* [35]. Note that HSVM, OT and HBLR (with soft-max) are inherently multiclass methods and are not applicable in multilabeled scenarios.

To scale up to the larger datasets (e.g., LWIKI), for the HR models we used the approximate parallelization as discussed in section 3.3. The parallelization for the flat baselines (SVM and LR) is straightforward, i.e., simply learning the models for all the class-labels in parallel. Among the hierarchical baselines, TD can be easily parallelized as the class models can be trained independently; HSVM and OT cannot be parallelized and hence cannot be scaled to the larger datasets. Therefore, we only report the results of HSVM and OT on the smaller datasets (CLEF and LSHTC-small) where they scaled. In addition, we also include the best results in benchmark evaluations for comparison, according to the numbers available on the LSHTC website.

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

4.3 Evaluation Metrics

We use the following standard evaluation metrics [37] to measure the performance of all the methods.

- **Micro-$\mathbf{F_1}$** is a conventional metric used to evaluate classification decisions [37], [19]. Let TP_t, FP_t, FN_t denote the true-positives, false-positives and false-negatives for the class-label $t \in T$. The micro-averaged 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-$\mathbf{F_1}$** 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.

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

5. RESULTS

We present three sets of results. The first set of results compares the performance of our proposed HR-models with the best results on the datasets in the benchmark evaluations conducted by LSHTC [4]. The second set of results presents pairwise comparison for SVM/HR-SVM and LR/HR-LR, respectively, to examine the same classifiers with and without recursive regularization. The third set of the results focus on the efficiency in computational time.

5.1 Effectiveness Comparison

Table 2 compares the results of our proposed HR-models (HR-SVM and HR-LR) with some of 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 website[4]. Table 2 shows that the HR-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, HR-SVM shows a consistent 10% relative improvement than the currently published results.

Table 3 summarizes the results of pairwise comparison between HR models against the corresponding non-HR baselines i.e. HR-SVM against SVM and HR-LR against LR. For an informative comparison, we also include the results of the other hierarchical baselines TD, HSVM and OT as well as the results from our work [16] where ever applicable. Note that all our baseline implementations have been

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

Table 2: Macro-F_1 and Micro-F_1 of well established benchmark results (excluding our own submissions to the system) against our proposed models. Bold faced number indicates best performing method.

	LSHTC Published Results	HR-SVM	HR-LR
DMOZ-2010			
Macro-F_1	**34.12**	33.12	32.42
Micro-F_1	**46.76**	46.02	45.84
DMOZ-2012			
Macro-F_1	31.36	**33.05**	20.04
Micro-F_1	51.98	**57.17**	53.18
DMOZ-2011			
Macro-F_1	**26.48**	25.69	23.90
Micro-F_1	38.85	**43.73**	42.27
SWIKI-2011			
Macro-F_1	23.16	**28.72**	24.26
Micro-F_1	37.39	**41.79**	40.99
LWIKI			
Macro-F_1	18.68	**22.31**	20.22
Micro-F_1	34.67	**38.08**	37.67

thoroughly tested and all convergence parameters have been appropriately set to optimize the objective as much as possible. Regularization parameters have been appropriately tuned using cross validation and have NOT been arbitrarily set to heuristic values such as '1' as was done in [5], [40]. In fact we found that setting the regularization parameter to '1' was suboptimal and lowered the performance of the baselines.

The results in Table 3 shows that the HR models consistently outperforms the non-HR counterparts on all tested datasets. They are able to successfully leverage the hierarchical dependencies and further push (especially in Macro-F_1) beyond the performance of the baseline methods.

To further validate our results, we conducted pairwise significance tests between SVM and HR-SVM; LR and HR-LR, on the CLEF, IPC, LSHTC-small and RCV1 datasets. We used the sign test for Micro-F_1 and wilcoxon rank test for Macro-F_1. We are unable to conduct significance tests on the other datasets since 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 [5] which does not reveal classwise performance measures nor true test labels. The results of the significance tests (Table 3) on the four datasets show that the HR-models significantly outperform the non-HR models on three out of the four tested datasets.

Comparing the performance of the HR-models to the other hierarchical baselines (TD, HSVM and OT), we see that the former outperforms the latter on most of the datasets. The unusually low performance of OT seems contrary to the results published in [40], we believe the reason is because the regularization parameter was arbitrarily set to '1' without using cross-validation. In some of the datasets like LSHTC-small, the HR-models show a significant gain in performance, more than 16% relative improvement in both Macro and Micro measures.

5.2 Efficiency Comparison

[5]http://lshtc.iit.demokritos.gr/LSHTC2_ oracleUpload

Table 4 reports the training time taken for all the HR models and the flat baselines. The results on the CLEF, RCV1, IPC and LSHTC-small dataset are from a 32 core Intel Xeon X7560 @ 2.27GHz, 32GB RAM. For all the other datasets, we used the Hadoop cluster as described in section 3.3.

Comparing the training time of the HR-models with corresponding the non-HR counterparts, HR-SVM is on an average about 1.92 slower than SVM and HR-LR about 2.87 slower than LR. This is not surprising - the better performance of the HR models comes at the cost of increased computational time. However, even on the largest dataset LWIKI, SVM takes about 18 hours while the HR-SVM about 37 hours. Although the HR-models which offer better performance are slower, the computation time certainly falls within the tractable range even for the largest datasets.

Comparing the HR-models against other hierarchical baselines; TD is the only one among the baseline methods that scaled to all the data sets but has low performance. HSVM and OT could only scale to the smallest data sets (CLEF and LSHTC-small) and both of them are about 3.5x slower on CLEF and about 86x slower on LSHTC-small compared to HR-SVM. On the rest of the datasets, neither of these could even be trained successfully. HBLR, even with a superior performance, is on an average 7.3x slower than HR-SVM. None of the hierarchical baselines are applicable to graphical dependencies between clases (SWIKI and LWIKI dataset).

6. CONCLUSIONS

In this paper, we proposed a recursive regularization framework along with scalable optimization algorithms for large-scale classification with hierarchical and graphical dependencies between class-labels. We explored 2 different variants of our framework using the logistic loss function and the hinge-loss function. Our proposed approaches achieved state-of-the-art results on multiple benchmark datasets and showed a consistent improvement in performance over flat and hierarchical approaches.

Table 3: 1. Macro-F_1 and Micro-F_1 on the 9 datasets. Bold faced number indicates best performing method. 2. '-' denotes the method cannot scale to the dataset or not applicable (e.g TD, HSVM, OT and HBLR are not applicable on SWIKI/LWIKI datasets due to graphical dependencies between class-labels). 3. The significance-test results are denoted † for a p-value less than 5% and †† for p-value less than 1%. The tests are between HR-SVM and SVM; HR-LR and LR on the first 4 datasets alone as the true test-labels were unavailable on the other datasets (refer section 5)

	Comparison with Flat baselines				Other Hierarchical Baselines			
	SVM	**HR-SVM**	**LR**	**HR-LR**	**TD**	**HSVM**	**OT**	**HBLR**
CLEF								
Macro-F_1	48.59	53.92††	53.26	55.83†	32.32	57.23	37.12	**59.65**
Micro-F_1	77.53	80.02††	79.92	80.12†	70.11	79.72	73.84	**81.41**
RCV1								
Macro-F_1	54.72	**56.56**†	53.39	55.81†	34.15	-	-	-
Micro-F_1	80.82	**81.66**††	80.08	81.23††	71.34	-	-	-
IPC								
Macro-F_1	45.71	47.89†	48.29	49.60	42.62	-	-	**51.06**
Micro-F_1	53.12	54.26††	55.03	55.37†	50.34	-	-	**56.02**
LSHTC-small								
Macro-F_1	28.62	28.94	28.12	28.48	20.01	21.95	19.45	**30.81**
Micro-F_1	45.21	45.31	44.94	45.11	38.48	39.66	37.12	**46.03**
DMOZ-2010								
Macro-F_1	32.64	**33.12**	31.58	32.42	22.30	-	-	-
Micro-F_1	45.36	**46.02**	45.40	45.84	38.64	-	-	-
DMOZ-2012								
Macro-F_1	31.59	**33.05**	14.18	20.04	30.01	-	-	-
Micro-F_1	56.44	**57.17**	52.79	53.18	55.14	-	-	-
DMOZ-2011								
Macro-F_1	24.34	**25.69**	21.67	23.90	21.07	-	-	-
Micro-F_1	42.88	**43.73**	41.29	42.27	35.91	-	-	-
SWIKI-2011								
Macro-F_1	26.57	**28.72**	19.51	24.26	17.39	-	-	-
Micro-F_1	40.87	**41.79**	37.65	40.99	36.65	-	-	-
LWIKI								
Macro-F_1	19.89	**22.31**	18.65	20.22	-	-	-	-
Micro-F_1	37.66	**38.08**	36.96	37.67	-	-	-	-

7. ACKNOWLEDGMENTS

This work is supported, in part, by the National Science Foundation (NSF) under grant IIS_1216282. 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 . The Open Cloud cluster is supported, in part, by NSF, under award CCF_1019104, and the Gordon and Betty Moore Foundation, in the eScience project.

8. REFERENCES

[1] http://leon.bottou.org/projects/sgd.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[3] A. Argyriou, C.A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. 2007.

[4] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. *NIPS*, 23:163–171, 2010.

[5] P.N. Bennett and N. Nguyen. Refined experts: improving classification in large taxonomies. In *SIGIR*, 2009.

[6] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM*, pages 78–87. ACM, 2004.

[7] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Incremental algorithms for hierarchical classification. *JMLR*, 7:31–54, 2006.

[8] 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.

[9] O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *ICML*, page 27. ACM, 2004.

[10] I. Dimitrovski, D. Kocev, L. Suzana, and S. Džeroski. Hierchical annotation of medical images. In *IMIS*, 2008.

[11] S. Dumais and H. Chen. Hierarchical classification of web content. In *ACM SIGIR*, 2000.

[12] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *SIGKDD*, pages 109–117. ACM, 2004.

Table 4: The training time (in mins) for all the methods.

	Comparison with Flat baselines				Other Hierarchical Baselines			
	SVM	**HR-SVM**	**LR**	**HR-LR**	**TD**	**HSVM**	**OT**	**HB-LR**
CLEF	.15	.42	.24	1.02	.13	3.19	1.31	3.05
RCV1	.273	.55	2.89	11.74	.213	-	-	-
IPC	3.12	6.81	4.17	15.91	2.21	-	-	31.2
LSHTC-small	.31	.52	1.93	3.73	.11	289.60	132.34	5.22
DMOZ-2010	5.12	8.23	97.24	123.22	3.97	-	-	-
DMOZ-2012	22.31	36.66	95.38	229.73	12.49	-	-	-
DMOZ-2011	39.12	58.31	101.26	248.07	16.39	-	-	-
SWIKI-2011	54	89.23	99.46	296.87	21.34	-	-	-
LWIKI	1114.23	2230.54	2134.46	7282.09	-	-	-	-

[13] A. Genkin, D.D. Lewis, and D. Madigan. Large-scale bayesian logistic regression for text categorization. 2007.

[14] J. Gonzalez, Y. Low, and C. Guestrin. Residual splash for optimally parallelizing belief propagation. Aistats, 2009.

[15] S. Gopal and Y. Yang. Multilabel classification with meta-level features. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 315–322. ACM, 2010.

[16] Siddharth Gopal, Yiming Yang, Bing Bai, and Alexandru Niculescu-Mizil. Bayesian models for large-scale hierarchical classification. In *Advances in Neural Information Processing Systems 25*, pages 2420–2428, 2012.

[17] 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 *ICML*, pages 408–415. ACM, 2008.

[18] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. 1997.

[19] D.D. Lewis, R.E. Schapire, J.P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *SIGIR*, pages 298–306. ACM, 1996.

[20] D.D. Lewis, Y. Yang, T.G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

[21] D.C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[22] 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.

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

[24] 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.

[25] T.P. Minka. A comparison of numerical optimizers for logistic regression. *Unpublished draft*, 2003.

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

[27] 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.

[28] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*, pages 807–814. ACM, 2007.

[29] A. Smola and R. Kondor. Kernels and regularization on graphs. *Learning theory and kernel machines*, pages 144–158, 2003.

[30] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[31] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6(2):1453, 2006.

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

[33] IPC WIPO. http://www.wipo.int/classifications /ipc/en/support/.

[34] G.R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *SIGIR*, pages 619–626. ACM, 2008.

[35] Y. Yang. A study of thresholding strategies for text categorization. In *SIGIR*, pages 137–145. ACM, 2001.

[36] Y. Yang, J. Zhang, and B. Kisiel. A scalability analysis of classifiers in text categorization. In *SIGIR*, pages 96–103. ACM, 2003.

[37] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1:67–88, 1999.

[38] T. Zhang and F.J. Oles. Text categorization based on regularized linear classification methods. *Information retrieval*, 4(1):5–31, 2001.

[39] T. Zhang, A. Popescul, and B. Dom. Linear prediction models with graph regularization for web-page categorization. In *SIGKDD*, pages 821–826. ACM, 2006.

[40] D. Zhou, L. Xiao, and M. Wu. Hierarchical classification via orthogonal transfer. Technical report, MSR-TR-2011-54, 2011.