HierCost: Improving Large Scale Hierarchical Classification with Cost Sensitive Learning

Anveshi Charuvaka(✉) and Huzefa Rangwala

George Mason University, Fairfax, USA
acharuva@gmu.edu, rangwala@cs.gmu.edu

Abstract. Hierarchical Classification (HC) is an important problem with a wide range of application in domains such as music genre classification, protein function classification and document classification. Although several innovative classification methods have been proposed to address HC, most of them are not scalable to web-scale problems. While simple methods such as top-down "pachinko" style classification and flat classification scale well, they either have poor classification performance or do not effectively use the hierarchical information. Current methods that incorporate hierarchical information in a principled manner are often computationally expensive and unable to scale to large datasets. In the current work, we adopt a cost-sensitive classification approach to the hierarchical classification problem by defining misclassification cost based on the hierarchy. This approach effectively decouples the models for various classes, allowing us to efficiently train effective models for large hierarchies in a distributed fashion.

1 Introduction

Categorizing entities according to a hierarchy of general to specific classes is a common practice in many disciplines. It can be seen as an important aspect of various fields such as bioinformatics, music genre classification, image classification and more importantly document classification [18]. Often the data is curated manually, but with exploding sizes of databases, it is becoming increasingly important to develop automated methods for hierarchical classification of entities.

Several classification methods have been developed over the past several years to address the problem of Hierarchical Classification (HC). One straightforward approach is to simply use multi-class or binary classifiers to model the relevant classes and disregard the hierarchical information. This methodology has been called *flat* classification scheme in HC literature [18]. While flat classification can be competitive, an important research directions is to improve the classification performance by incorporating the hierarchical structure of the classes in the learning algorithm. Another simple data decomposition approach trains local classifiers for each of the classes defined according to the hierarchy, such that the trained model can be used in a top-down fashion to take the most relevant path in testing. This top-down approach trains each classifier on a smaller dataset and

A. Appice et al. (Eds.): ECML PKDD 2015, Part I, LNAI 9284, pp. 675–690, 2015.
DOI: 10.1007/978-3-319-23528-8_42

is quite efficient in comparison to flat classification, which generally train one-vs-rest classifiers on the entire dataset. However, a severe drawback of this approach is that if a prediction error is committed at a higher level, then the classifier selects a wrong prediction path, making it impossible to recover from the errors at lower levels. Due to this error propagation, sometimes, sever degradation in performance has been noted for the top-down classifier in comparison to flat classifier [9]. A review of HC in several application domains can be found in a recent survey by Silla Jr. et al. [18].

In recent years, researchers have shown more interest in large scale classification where the number of categories, number of instances, as well as the number of features are large. This has been highlighted by large scale hierarchical text classification competitions such as LSHTC[1] [15] and BioASQ [2], which pose several interesting challenges. Firstly, since these problems deal with several thousands of classes, scalability of the methods is a crucial requirement. Secondly, in spite of having large number of total training examples, many categories have few positive training samples. For example, 76% of the class-labels in the Yahoo! Directory have 5 or fewer positive instances [11], and 72% in the Open Directory Project have fewer than 4 positive instances [9]. This data sparsity brings about two issues: (i) due to the lack of sufficient examples, the learned models tend to be less robust, and (ii) due to the large skew in positive and negative class distributions, the performance of smaller classes tends to deteriorate severely as the mis-predictions tend to favor larger classes.

In this paper, we try to address two main issues of large scale hierarchical classification, class imbalance and training efficiency, by extending the flat classification approach using cost sensitive training examples. Although regularization methods which constrain the learned models to be close to neighboring classes according to the hierarchy have been effective, they induce large scale optimization problems which require specialized solutions [3]. Instead, by re-defining the problem from regularization based approach to a cost sensitive classification approach (based on similar assumptions) tends to decouple the training of the models which can be trained in parallel fashion. We study various methods to incorporate cost-sensitive information into hierarchical classification and empirically evaluate their performance on several datasets. Finally, since any instance based cost sensitive method can be used as a base classifier, the HC problem can benefit from advancements in cost-sensitive classification.

2 Definitions and Notations

In this section, we discuss the notations commonly used in this paper. $\mathcal{N}$ denotes the set of all the nodes in the hierarchy, and $\mathcal{T} \subset \mathcal{N}$ denotes the set of terminal nodes to which examples are assigned. $\mathbf{w}_n$ denotes the model learnt for class $n \in \mathcal{N}$. $(\mathbf{x}_i, l_i)$ denotes the i^{th} example where $\mathbf{x}_i \in \mathbb{R}^D$ and $l_i \in \mathcal{T}$. The number of examples is denote by N. We use y_i^n to denote the binary label used in the

[1] http://lshtc.iit.demokritos.gr
[2] http://bioasq.org

learning algorithm for $\mathbf{w}_n$. For the training example $(\mathbf{x}_i, l_i)$ we set $y_i^n = 1$ iff $l_i = n$ and $y_i^n = -1$ otherwise. $\gamma(a, b)$ denotes the graph distance between classes $a, b \in \mathcal{N}$ in the hierarchy, which is defined as the number of edges in the undirected path between nodes a and b. We use c_i^n to denote the cost of example i in training of the model for class n. To simplify the notation, in some places, we drop the super-script explicitly indicating the class, and use y_i , c_i and $\mathbf{w}$ in place of y_i^n, c_i^n and $\mathbf{w}_n$ where the class is implicitly understood to be n. L is used to denote a generic loss function. In the current work, logistic loss function is used, which is defined as $L(y, f(\mathbf{x})) = \log(1 + \exp(-yf(\mathbf{x})))$.

3 Motivation and Related Work

In this section, we discuss the motivation for the approach taken in this paper and examine various related methods proposed in the literature for addressing the hierarchical classification problem.

A few large margin methods have been proposed as cost sensitive extensions to the multi-class classification problem. Dekel et al. [5] proposed a large margin method where the margin is defined with respect to the tree distance. Although their method shows improvement on tree-error, the performance degrades with respect to misclassification error. The methods proposed by Cai et al. [2] and more recently by Chen et al. [4], also make an argument in favor of modifying the misclassification error by making it dependent on the hierarchy. Both these methods can be seen as special cases of a more general large margin structured output prediction method proposed by Tsochantaridis et al. [20]. Although all these methods try to incorporate cost sensitive losses based on the hierarchy, they formulate a global optimization problem where the models for all the classes are learned jointly and are not scalable to large scale classification problems.

Several methods try to incorporate the bias that categories which are semantically related according to the hierarchy should also be similar with respect to the learned models. McCallum et al. [13] show that for Naive Bayes classifier, smoothing the parameter estimates of the data-sparse children nodes with the parameter estimates of parent nodes, using a technique known as shrinkage, produces more robust models. Other models in this class of methods typically incorporate this assumption using parent child regularization or hierarchy based priors [9,13,17]. In one of the prototypical models in this class of works, which extends Support Vector Machines (SVM) and Logistic Regression (LR) [9], the objective function takes the form given in (1),

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_{|\mathcal{N}|}} \sum_{n\in\mathcal{N}} \frac{1}{2} \left\|\mathbf{w}_n - \mathbf{w}_{\pi(n)}\right\|_2^2 + C \sum_{n\in\mathcal{T}} \sum_{i=1}^{N} L\left(y_i^n, \mathbf{w}_n^T \mathbf{x}_i\right) \tag{1}$$

where, $\pi(n)$ represents the parent of the class n according to the provided hierarchy. The loss function L has been modeled as logistic loss or hinge loss. Note that the loss is defined only on the terminal nodes $\mathcal{T}$, and the non-terminal node $\mathcal{N} - \mathcal{T}$, are introduced only as a means to facilitate regularization. Since the weights associated with different classes are coupled in the optimization problem, Gopal et al. [9] used a distributed implementation of block coordinate descent where each block of variables corresponds to $\mathbf{w}_n$ for a particular class n. The model weights are learned similarly to standard LR or SVM for the leaf nodes $n \in \mathcal{T}$, with the exception that the weights are shrunk towards parents instead of towards the zero vector by the regularizer. For an internal non-leaf node, the weights updates are averages of the other nodes which are connected to it according to the hierarchy, i.e., the parents and children in the hierarchy.

The kind of regularization discussed above can be compared to the formulations proposed in transfer and multi-task learning (MTL) literature [7], where externally provided task relationships can be utilized to constrain the jointly learned model weights to be similar to each other. In the case of HC, the task relationship are explicitly provided as hierarchical relationships. However, one significant difference between the application of this regularization between HC and MTL is that the sets of examples in MTL for different tasks are, in general, disjoint. Whereas, in the case of HC, the examples which are classified as positive for one class are negative for all other classes except those which belong to the ancestors of that class. Therefore, even though these models impose similarity between siblings indirectly through the parent, when their respective models are trained, the negative and positive examples are flipped. Hence, the opposing forces for examples and regularization are acting simultaneously during the learning of these models. However, due to the regularization strength being imposed by the hierarchy, the net effect is that the importance of misclassifying the examples coming for nearby classes is down-weighted. This insight can be directly incorporated into the learning algorithm by defining the loss of nearby negative examples for a class, where "near" is defined with respect to the hierarchy, to be less severe than the examples which are farther. This yields a simple cost sensitive classification method where the misclassification cost is directly proportional to the distance between the nodes of the classes, which is the key contribution of our work. With respect to prediction, there are only two classes for each trained model, but the misclassification costs of negative examples are dependent on the nodes from which they originate.

In this framework for HC, we essentially decouple the learning for each node of the hierarchy and train the model for each one independently. Thus, rendering scalability to this method. Instead of jointly formulating the learning of model parameters for all the classes, we turn the argument around from that of regularizing the model weights towards those of the neighboring models, to the rescaling the loss of example depending on the class relationships. A similar argument was made in the case of multi-task transfer learning by Saha et al. [16], where, in place of joint regularization of model weights, as is typically done in multi-task learning [8], they augment the target tasks with examples from source tasks. However, the losses for the two sets of examples are scaled differently.

4 Methods

As shown in some previous works [1,9], the performance of flat classification has been found to be very competitive, especially for large scale problems. Although, the top-down classification method is efficient in training, it fares poorly with respect to classification performance due to error propagation. Hence, in this work, we extend the flat classification methodology to deal with HC. We use the one-vs-all approach for training, where for each class n, to which examples are assigned, we learn a classification model with weight $\mathbf{w}_n$. Note, that it is unnecessary to train the models for non-terminal classes, as they only serve as virtual labels in the hierarchy. Once the models for each terminal class are trained, we perform prediction for input example $\mathbf{x}$ as per (2)

$$\hat{y} = \mathbf{argmax}_n \ \mathbf{w}_n^T \mathbf{x} \tag{2}$$

The essential idea is to formulate the learning algorithm such that the mis-predictions on negative examples coming from nearby classes are treated as being less severe. We encode this assumption through cost sensitive classification. Standard regularized binary classification models, such as SVMs and Logistic Regression, minimize an objective function consisting of loss and regularization terms as shown in (3).

$$\min_{\mathbf{w}} \underbrace{\sum_{i=1}^{N} L\left(y_i, f\left(\mathbf{x}_i \mid \mathbf{w}\right)\right)}_{loss} + \rho \underbrace{R\left(\mathbf{w}\right)}_{regularizer} \tag{3}$$

where $\mathbf{w}$ denotes the learned model weights. The loss function L, which is generally a convex approximation of zero-one loss, measures how well the model fits the training examples. Here, each example is considered to be equally important. As per the arguments made previously, we modify the importance of correctly classifying examples according to the hierarchy using example based misclassification costs. For models such as logistic regression, incorporating example based costs into the learning algorithm is simply a matter of scaling the loss by a constant positive value. Assuming that the classifier is being learned for class n, we can write the cost sensitive objective function as shown in (4).

$$\min_{\mathbf{w}} \sum_{i=1}^{N} c_i L\left(y_i, \mathbf{w}^T \mathbf{x}_i\right) + \rho R\left(\mathbf{w}\right) \tag{4}$$

Here, c_i denotes the cost associated with misclassification of the i^{th} example. Although, this scaling works for the smooth loss function of Logistic Regression, it is not as straightforward in the case of non-smooth loss functions such as hinge loss [12]. Therefore, using the formulation given in (4), for each of the models, we can formulate the objective function for class n as a cost sensitive logistic regression problem where the cost of the example $\mathbf{x}_i$ for the binary classifier of class n depends on how far the actual label $l_i \in \mathcal{T}$ is from n, according to the hierarchy. Additionally, to deal with the issue of rare categories, we can also increase the cost of the positive examples for data-sparse classes thus mitigating the effects of highly skewed datasets. Since our primary motivation is to argue in favor of using hierarchical cost sensitive learning instead of more expensive regularization models, we only concentrate on logistic loss, which is easier to handle, from optimization perspective, than non-smooth losses such as SVM's hinge loss. The central issue, now, is that of defining the appropriate costs for the positive and negative examples based on the distance of the examples from the true class according to the hierarchy and the number of examples available for training the classifiers. In the following section we discuss the selection of costs for negative and positive examples.

4.1 Cost Calculations

Hierarchical Cost. Hierarchical costs impose the requirement that the misclassification of negative examples that are farther away from the training class according to the hierarchy should be penalized more severely. Encoding this assumption, we define the following instantiations of hierarchical costs. We assume the class under consideration is denoted by n.

Tree Distance (TrD): In (5), we define the cost of negative examples as the undirected graphical distance, $\gamma(n, l_i)$, between the class n and l_i, the class label of example $\mathbf{x}_i$. We call this cost *Tree Distance (TrD)*. We define $\gamma_i \equiv \gamma(n, l_i)$ and $\gamma_{\max} = \max_{j \in \mathcal{T}} \gamma_j$. Since dissimilarity increases with increasing γ_i, the cost is a monotonically increasing function of γ_i.

$$c_i = \begin{cases} \gamma_{\max} & l_i = n \\ \gamma_i & l_i \neq n \end{cases} \tag{5}$$

Number of Common Ancestors (NCA): In some applications, where the depth (distance of a node from the root node) of all terminal labels is not uniform, a better definition of similarity might be the number of common ancestor between two nodes. This is encoded in *NCA* costs, represented in (6). In the definition, α_i is used to denote the number of common ancestors between the pair of nodes l_i and n. Unlike γ_i which is a monotonically increasing function of dissimilarity, α_i is a monotonically increasing function of similarity. $\alpha_{\max} = \max_{j \in \mathcal{T}} \alpha_j$.

$$c_i = \begin{cases} \alpha_{\max} + 1 & l_i = n \\ \alpha_{\max} - \alpha_j + 1 & l_i \neq n \end{cases} \tag{6}$$

Exponentiated Tree Distance (ExTrD): Finally, in some cases, especially for deep hierarchies, the tree distances can be large, and therefore, in order to shrink the values of cost into a smaller range, we define *ExTrD* in (7), where $k > 1$, can be tuned according to the hierarchy. Through tuning we found that on our dataset, the range of values $1.1 \leq k \leq 1.25$ of works well.

$$c_i = \begin{cases} k^{\gamma_{\max}} & l_i \neq n \\ k^{\gamma_i} & l_i \neq n \end{cases} \tag{7}$$

In all these cases, we set the cost of the positive class to the maximum cost of any example.

Imbalance Cost. In certain cases, especially for large scale hierarchical text classification, some classes are extremely small with respect to the number of positive examples available for training. In these cases, the learned decision boundary might favor the larger classes. Therefore, to deal with this imbalance in the class distributions, we increase the cost of misclassifying rare classes. This has the effect of mitigating the influence of skew in the data distributions of abundant and rare classes. We call the cost function incorporating this as *Imbalance Cost (IMB)*, which is given in (8). We noticed that using cost such as inverse of class size diminishes the performance. Therefore, we use a squashing function inspired by logistic function $f(x) = L/\left[1 + \exp -k(x - x_0)\right]$, which would not severely disadvantage very large classes.

$$c_i = 1 + L/\left[1 + \exp\left(|N_i - N_0|_+\right)\right] \tag{8}$$

where $|a|_+ = \max(a, 0)$ and N_i is the number of examples belonging to class denoted by l_i. The value of c_i lies in the range $(1, L/2 + 1)$. We can use a tunable parameter N_0, which can be intuitively interpreted as the number of examples required to build a "good" model, above which increasing the cost does not have a significant effect or might adversely affect the classification performance. In our experiments, we used $N_0 = 10$ and $L = 20$.

In order to combine the Hierarchical Costs with the Imbalance Costs, we simply multiply the contributions of both the costs. We also experimented with several other hierarchical cost variants, which are not discussed here due to space constraints.

4.2 Optimization

Since we are dealing with large scale classification problems, we need an efficient optimization method which relies only on the first order information to solve the learning problem given in (9).

$$\min_{\mathbf{w}} \left[f(\mathbf{w}) = \sum_{i=1}^{N} c_i \log\left(1 + \exp\left(-y_i \mathbf{w}^T \mathbf{x}_i\right)\right) + \rho \|\mathbf{w}\|_2^2 \right] \tag{9}$$

Since the cost values c_i are predefined positive scalars, we can adapt any method used to solve the standard regularized Logistic Regression (LR). We use accelerated gradient descent due to its efficiency and simplicity. The ordinary gradient descent method has a convergence rate of $O\left(1/k\right)$, where k is the number of iterations. Accelerated gradient method improves the convergence rate to $O\left(1/k^2\right)$ by additionally using the gradient information from the previous iteration [14]. The complete algorithm to solve the cost-sensitive binary logistic regression is provided in Line 1. We describe the notations and expressions used in the algorithm below.

N is the number of examples; $X \in \mathbb{R}^{N\times D}$ denotes the data matrix; $\mathbf{y} \in \{\pm 1\}^N$ is the binary label vector for all examples; $\rho \in \mathbb{R}_+$ is the regularization parameter; $\mathbf{c} = (c_1, c_2, \dots, c_N) \in \mathbb{R}_+^N$ denotes the cost vector, where c_i is the cost for example i ; $\mathbf{w} \in \mathbb{R}^D$ denotes the weight vector learned by the classifier; $f\left(\mathbf{w}\right)$ denotes the objective function value, given in (10)

$$f\left(\mathbf{w}\right) = \mathbf{c}^T \left(\log\left[1 + \exp\left(X\mathbf{w}\right)\right]\right) + \rho \left\|\mathbf{w}\right\|_2^2 \tag{10}$$

$\nabla\mathbf{f}$ is the gradient of f w.r.t. $\mathbf{w}$, which is defined in (11), where $(\mathbf{y} \circ \mathbf{c})$ denotes the vector obtained from the element-wise product of $\mathbf{y}$ and $\mathbf{c}$. Similarly $\exp(\cdot)$ and division in (11) are element-wise operators.

$$\nabla\mathbf{f}\left(\mathbf{w}\right) = 2\mathbf{w} + X^T \left(\frac{-\mathbf{y} \circ \mathbf{c}}{1 + \exp\left\{(X\mathbf{w}) \circ \mathbf{y}\right\}}\right) \tag{11}$$

$\hat{f}_\lambda\left(\mathbf{u}, \mathbf{w}\right)$, described in (12), is the quadratic approximation of $f\left(\mathbf{u}\right)$ at $\mathbf{w}$ using approximation constant or step size λ. The appropriate step size in each iteration is found using line search.

$$\hat{f}_\lambda\left(\mathbf{u}, \mathbf{w}\right) = f\left(\mathbf{w}\right) + \left(\mathbf{u} - \mathbf{w}\right)^T \nabla\mathbf{f}\left(\mathbf{w}\right) + 1/2\lambda \left\|\mathbf{u} - \mathbf{w}\right\|_2^2 \tag{12}$$

4.3 Dealing with Hierarchical Multi-label Classification

HC problems are trivially multi-label problems because every example belonging to a class also inherits the labels of the ancestor classes. But in the current context, we call a problem as hierarchical multi-label problem if an example can be assigned multiple labels such that neither is an ancestor nor descendant of the other.

In the case of single label classification, we perform prediction as per (2), which selects only a single label per example. A trivial extension to multi-label classification can be done by choosing a threshold of 0 such that we assign label n to example $\mathbf{x}$ if $\mathbf{w}_n^T\mathbf{x} > 0$ as in the case of binary classification. However, a better strategy is to optimize the threshold t_n for each class using a validation set, such that the label n is assigned to the test example if $\mathbf{w}_n^T\mathbf{x} > t_n$. This strategy is called SCut method [21]. Other strategies such as learning a thresholding function $t\left(\mathbf{w}_1^T\mathbf{x}, \mathbf{w}_2^T\mathbf{x}, \dots, \mathbf{w}_M^T\mathbf{x}\right)$ using the margin scores [9] might improve the results, but they are somewhat more expensive to tune for large scale problems. The SCut method can tune the threshold independently of all other classes. In cases where we do not have sufficient examples to tune the threshold, i.e. the class has a single training example, we set the threshold to $t_n = 0$.

Algorithm 1. Accelerated Gradient Method for Cost Sensitive LR

```
Data: $X, \mathbf{y}, \mathbf{c}, \rho, \beta \in (0,1), max\_iter$
Result: $\mathbf{w}$
Let $\lambda_0 := 1; \mathbf{w}_{-1} = \mathbf{w}_0 = \mathbf{0}$;
for $k = 1 \ldots max\_iter$ do
    $\theta^k = \frac{k-1}{k+2}$
    $\lambda = \lambda_{k-1}$
    while $TRUE$ do
        $\mathbf{w} = \mathbf{u}_k - \lambda \nabla f(\mathbf{u}_{k-1})$
        if $f(\mathbf{w}) \leq \hat{f}_\lambda(\mathbf{u}, \mathbf{w})$ then
            $\lambda_k = \lambda$
            $\mathbf{w}_k = \mathbf{w}$
            break
        else
            $\lambda = \beta\lambda$
        end
    end
    if $converged$ then
        break
    end
end
return $\mathbf{w}_k$
```

The second issue that we must deal with is the definition of cost based on hierarchical distances and class sizes. With respect to the training of a class n, an example $\mathbf{x}_i$ might be associated with multiple labels $l_1, l_2 \ldots, l_K$. In this case the tree distance γ_i is not uniquely defined. Hence, we must aggregate the values of $\gamma(n, l_1), \ldots, \gamma(n, l_K)$. One strategy is to use an average of the values, but we found that the taking the minimum works a little better. Similarly we can use a minimum of of the number of common ancestors to all target labels for *NCA* costs.

Finally, since an example is associated with multiple class labels, the class size N_i of the examples is also not uniquely defined, in this case as well, we use the the effective size as the minimum size out of all the labels associated with $\mathbf{x}_i$ for our *IMB* cost. It also makes intuitive sense, because we are trying to upweight the rare classes, and the rarest class should be given precedence in terms of the cost definition.

5 Experimental Evaluations

5.1 Datasets

The details of the datasets used for our experimental evaluations are provided in Table 1. **CLEF** [6] is a dataset comprising of medical images annotated with hierarchically organized Image Retrieval in Medical Applications (IRMA) codes. The task is to predict the IRMA codes from image features. Images are described with 80 features extracted using a technique called local distribution of edges.

IPC is a collection of patent documents classified according to the International Patent Classification (IPC) System[3]. **DMOZ-small, DMOZ-2010** and **DMOZ-2012** are hierarchical text classification datasets released as part of PASCAL Large Scale Hierarchical Text Classification Challenge (LSHTC)[4] [15]. For LSHTC datasets except DMOZ-small, labels of the test datasets are not available, but certain classification metrics can be obtained through their online evaluation system. **RCV1-v2** [10] is a multi-label text classification dataset extracted from Reuters corpus of manually categorized newswire stories. **RCV1** is multi-label and non-mandatory leaf node predication [18] dataset, while the rest of the datasets are single label datasets with examples assigned only to leaf nodes. All the hierarchies used in the experiments are tree-based. For all the text datasets, raw term frequencies were converted to term weights using Cornell ltc term weighting [10].

Table 1. Dataset Statistics.

	CLEF	DMOZ SMALL	IPC	RCV1	DMOZ 2010	DMOZ 2012
Num. Training Examples	10000	4463	46324	23149	128710	383408
Num. Test Examples	1006	1858	28926	781265	34880	103435
Num. Features	80	51033	345479	48728	381580	348548
Num. Nodes	97	2388	553	117	17222	13963
Num. Terminal Nodes	63	1139	451	101	12294	11947
Max. Depth of Leaf Nodes	4	6	4	6	6	6
Avg. Labels per Example	1	1	1	3.18	1	1

5.2 Evaluation Metrics

We evaluate the prediction using the following standard performance measures used in HC literature. The set based measures Micro-F$_1$ and Macro-F$_1$ are shown below.

$$\text{Micro-F}_1 = (2PR)\,/\,(P+R) \tag{13}$$

$$\text{Macro-F}_1 = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} 2P_tR_t/\,(P_t+R_t) \tag{14}$$

$\mathcal{T}$ denotes is the set of class labels, P_t and R_t are the precision and recall values for class $t \in \mathcal{T}$. P and R are the overall precision and recall values for the all the classes taken together. Micro-F$_1$ gives equal weight to all the examples therefore it favors the classes with more number of examples. In the case of single label classification, Micro-F$_1$ is equivalent to accuracy. Macro-F$_1$ gives equal weight

[3] http://www.wipo.int/classifications/ipc/en/
[4] http://lshtc.iit.demokritos.gr/

to all the classes irrespective of their size. Hence, the performance on the smaller categories is also taken into consideration.

Set based measures do not consider the distance of misclassification with respect to the true label of the example, but in general, it is reasonable to assume in most cases that misclassifications that are closer to the actual class are less severe than misclassifications that are farther from the true class with respect to the hierarchy. Hierarchical measures, therefore, take the distances between the actual and predicted class into consideration. The hierarchical measures, described in eqs. (15) to (17), are Hierarchical Precision (hP), Hierarchical Recall (hR), and their harmonic mean, Hierarchical F$_1$ (hF_1) respectively. These are hierarchical extensions of standard precision and recall scores. Tree-induced Error (TE) [19], given in (18), measures the average hierarchical distance between the actual and predicted labels.

$$hP = \sum_{i=1}^{N} \left| \mathcal{A}\left(l_i\right) \cap \mathcal{A}\left(\hat{l}_i\right) \right| / \sum_{i=1}^{N} \left| \mathcal{A}\left(\hat{l}_i\right) \right| \tag{15}$$

$$hR = \sum_{i=1}^{N} \left| \mathcal{A}\left(l_i\right) \cap \mathcal{A}\left(\hat{l}_i\right) \right| / \sum_{i=1}^{N} \left| \mathcal{A}\left(l_i\right) \right| \tag{16}$$

$$hF_1 = 2 \cdot hP \cdot hR / \left(hP + hR\right) \tag{17}$$

$$TE = \frac{1}{N} \sum_{i=1}^{N} \gamma\left(l_i, \hat{l}_i\right) \tag{18}$$

where, $\hat{l}_i$ and l_i are the predicted label and true labels of example i, respectively. $\gamma\left(a, b\right)$ the graph distance between a and b according to the hierarchy. $\mathcal{A}\left(l\right)$ denotes the set that includes the node l and all its ancestors except the root node. For TE lower values are better, whereas for all other measures higher values are better.

For multi-label classification, where each l_i is a set of micro-labels, we redefine graph distance and ancestors as: $\gamma_{ml}(l_i, \hat{l}_i) = \left|\hat{l}_i\right|^{-1} \sum_{a \in \hat{l}_i} \min_{b \in l_i} \gamma\left(a, b\right)$ and $\mathcal{A}_{ml}\left(l\right) = \cup_{a \in l} \mathcal{A}\left(a\right)$.

5.3 Experimental Details

For all the experiments, the regularization parameter is tuned using a validation set. The model is trained for a range of values 10^k with appropriate values for k selected depending on the dataset. Using the best parameter selected on validation set, we retrained the models on the entire training set and measured the performance on a held out test set. The source code implementing the methods discussed in this paper is available on our website [5]. The experiments were performed on computers with Dell C8220 processors with dual Intel Xeon E5-2670 8 core CPUs and 64 GB memory.

[5] http://cs.gmu.edu/~mlbio/HierCost/

5.4 Methods for Comparison

In our experimental evaluations, we compare our cost sensitive hierarchical classification methods with the following hierarchical and flat classification methods proposed in the literature.

Logistic Regression (LR). One-vs-rest binary logistic regression is used in the conventional flat classification setting. For single label classification, we assign test examples to the class which achieves best classification score.

Hierarchical Regularization for LR (HRLR). This method proposed by Gopal et al. [9], extends flat classification using recursive regularization based on hierarchical relationships. Since we used exactly the same setup as the authors, in terms of training and test datasets, we are reporting their classification scores directly from [9].

Top-Down Logistic Regression (TD). This denotes Top-down logistic regression model, where we train a one-vs-rest multi-class classifier at each internal node. At testing time, the predictions are made starting from the root node. At each internal node, the highest scoring child node is selected until we reach a leaf node.

5.5 Results

In this section, we present experimental comparisons of various cost sensitive learning strategies with other baseline methods. We provide separate comparisons of different cost based improvements on smaller datasets, and finally compare our best method with the competing methods. In the tables, statistically significant results for Micro-F_1 and Macro-F_1 [22] are marked with either † or ‡ which correspond to p-values < 0.05 and < 0.001 respectively.

In Table 2, we compare LR with various *hierarchical costs* defined in Section 4.1. The results show a uniform improvement in all the metrics reported. There was a statistically significant improvement in Micro-F_1, especially for DMOZ Small, IPC and RCV1 datasets. Macro-F_1 scores were also improved, but due to the presence of only a small number of categories in CLEF and RCV1 datasets, statistical significance could not be established, except for ExTrD.

In Table 3 we compare the effect of introducting *imbalance costs*, discussed in Section 4.1, on standard LR and hierarchical costs. In IMB+LR only the imbalance cost is used, in others, we use the product of costs derived from IMB strategy and the corresponding hierarchical costs. We also measured the significance of the improvement over the corresponding results from Table 2. Only for DMOZ Small, which has a large number of classes with few examples, imbalance costs further improve the results significantly for all the methods. On CLEF, IPC and RCV1, where majority of the classes have sufficient number of examples for training, the results did not improve significantly in most cases. Overall, the IMB+ExTrD method provides more robust improvements.

The final comparison of our best method (IMB+ExTrD, which we call *HierCost* in the following) against various baseline methods is presented in Table 4.

Table 2. Performance comparison of hierarchical costs.

		Micro-F_1 (↑)	Macro-F_1 (↑)	hF_1 (↑)	TE (↓)
CLEF	LR	79.82	53.45	85.24	0.994
	TrD	80.02	55.51	**85.39**	0.984
	NCA	80.02	57.48	85.34	0.986
	ExTrD	**80.22**	**57.55**†	85.34	**0.982**
DMOZ SMALL	LR	46.39	30.20	67.00	3.569
	TrD	**47.52**‡	**31.37**‡	**68.26**	**3.449**
	NCA	47.36‡	31.20‡	68.12	3.460
	ExTrD	47.36‡	31.19‡	68.20	3.456
IPC	LR	55.04	48.99	72.82	1.974
	TrD	55.24‡	50.20‡	73.21	1.954
	NCA	**55.33**‡	**50.29**‡	**73.28**	**1.949**
	ExTrD	55.31‡	**50.29**‡	73.26	1.951
RCV1	LR	78.43	60.37	80.16	0.534
	TrD	79.46‡	60.61	82.83	0.451
	NCA	**79.74**‡	60.76	**83.11**	**0.442**
	ExTrD	79.33‡	**61.74**†	82.91	0.466

The evaluations on Dmoz 2010 and Dmoz 2012 datasets are blind and the predictions have to be uploaded to LSHTC website in order to obtain the scores. For Dmoz 2012, Tree Errors are not available and for Dmoz 2010, the hF_1 are not available. For HRLR, we do not have access to the predictions, hence, we could only report the values for Micro-F_1 and Macro-F_1 scores from [9]. Statistical significance tests compare the results of *HierCost* with LR. These tests could not be performed on LSHTC datasets due to non-availability of the true labels on test sets. As seen in Table 4, *HierCost* improves upon the baseline LR results as well as the results reported in [9], in most cases, especially the Macro-F_1 scores. The results of *HierCost* are better on most measures. TD performs worst on average on set-based measures. In fact, only on Dmoz 2012 dataset, TD is competitive, on the rest, the results are much worse than the *flat* LR classifier and its hierarchical extensions. On hierarchical measure, however, TD outperformed *flat* classifiers on some datasets.

In Table 5, we report the run-times comparisons of TD, LR and *HierCost*. We trained the models in parallel for different classes and computed the sum of run-times for each training instance. In theory, the run-times of LR and *HierCost* should be equivalent, because they solve similar optimization problems. However, minor variations in the run-times were observed because of the variations in optimal regularization penalties, which influences the convergence of the optimization algorithm. The runtimes of flat methods were significantly worse than TD, which is efficient in terms of training, but at considerable loss in classification performance. Although, we do not measure the training times of HRLR, based on the experience from a similar problem [3], the recursive model take between 3-10 iterations for convergence. In each iteration, the models for all the terminal labels need to be trained hence each iteration is about as expensive as

Table 3. Peformance comparison with imbalance cost included.

		Micro-F$_1$ (↑)	Macro-F$_1$ (↑)	hF$_1$ (↑)	TE (↓)
CLEF	IMB + LR	79.52	53.11	85.19	1.002
	IMB + TrD	79.92	52.84	85.59	0.978
	IMB + NCA	79.62	51.89	85.34	0.994
	IMB + ExTrD	**80.32**	**58.45**	**85.69**	**0.966**
DMOZ SMALL	IMB + LR	48.55‡	32.72‡	68.62	3.406
	IMB + TrD	**49.03**‡	33.21‡	69.41	3.334
	IMB + NCA	48.87‡	33.27‡	69.37	3.335
	IMB + ExTrD	**49.03**‡	**33.34**‡	**69.54**	**3.322**
IPC	IMB + LR	55.04	49.00	72.82	1.974
	IMB + TrD	55.60‡	**50.45**†	73.56	1.933
	IMB + NCA	55.33	50.29	73.28	1.949
	IMB + ExTrD	**55.67**‡	50.42	**73.58**	**1.931**
RCV1	IMB + LR	78.59‡	60.77	81.27	0.511
	IMB + TrD	**79.63**‡	61.04	**83.13**	**0.435**
	IMB + NCA	79.61	61.04	82.65	0.458
	IMB + ExTrD	79.22	**61.33**	82.89	0.469

Table 4. Performance comparison of HierCost with other baseline methods.

		Micro-F$_1$ (↑)	Macro-F$_1$ (↑)	hF$_1$ (↑)	TE (↓)
	TD	73.06	34.47	79.32	1.366
CLEF	LR	79.82	53.45	85.24	0.994
	HRLR	80.12	55.83	-	-
	HierCost	**80.32**	**58.45**†	**85.69**	**0.966**
	TD	40.90	24.15	**69.99**	**3.147**
DMOZ SMALL	LR	46.39	30.20	67.00	3.569
	HRLR	45.11	28.48	-	-
	HierCost	**49.03**‡	**33.34**‡	69.54	3.322
	TD	50.22	43.87	69.33	2.210
IPC	LR	55.04	48.99	72.82	1.974
	HRLR	55.37	49.60	-	-
	HierCost	**55.67**‡	**50.42**†	**73.58**	**1.931**
	TD	77.85	57.80	**88.78**	0.524
RCV1	LR	78.43	60.37	80.16	0.534
	HRLR	**81.23**	55.81	-	-
	HierCost	79.22‡	**61.33**	82.89	**0.469**
	TD	38.86	26.29	-	3.867
DMOZ 2010	LR	45.17	30.98	-	3.400
	HRLR	45.84	**32.42**	-	-
	HierCost	**45.87**	32.41	-	**3.321**
	TD	51.65	**30.48**	73.33	-
DMOZ 2012	LR	51.72	27.19	72.53	-
	HRLR	53.18	20.04	-	-
	HierCost	**53.36**	28.47	**73.79**	-

Table 5. Total training runtimes (in mins).

	TD-LR	LR	HierCost
CLEF	<1	<1	<1
DMOZ SMALL	4	41	40
IPC	27	643	453
RCV1	20	29	48
DMOZ 2010	196	15191	20174
DMOZ 2012	384	46044	50253

a single run of LR. In addition, the distributed recursive models require communication between the training machines which incurs an additional overhead.

6 Conclusions

In this paper, we have argued that the methods that extend flat classification using hierarchical regularization, can be viewed in a complementary way as weighting the losses on the negative examples depending on dissimilarity between the positive and negative classes. The approach proposed in this paper, incorporates this insight directly into the loss function by scaling the loss function according to the dissimilarity between the classes with respect to the hierarchy, thus obviating the need for recursive regularization and iterative model training. At the same time, this approach also makes parallelization trivial. Our method also mitigates the adverse effects of imbalance in the training data by upweighting the loss for examples from smaller classes, thus, significantly improving their classification results. Our experimental results show that the proposed method is able to efficiently incorporate hierarchical information by transforming the hierarchical classification problem into an example based cost sensitive classification problem. In future work, we would like to evaluate the benefits of cost sensitive classification using large margin classifiers such as support vector machines.

Acknowledgement. This work was supported by NSF IIS-1447489 and NSF career award to Huzefa Rangwala (IIS-1252318). The experiments were run on ARGO (http://orc.gmu.edu).

References

1. Babbar, R., Partalas, I., Gaussier, E., Amini, M.R.: On flat versus hierarchical classification in large-scale taxonomies. In: Advances in Neural Information Processing Systems, pp. 1824–1832 (2013)
2. Cai, L., Hofmann, T.: Hierarchical document categorization with support vector machines. In: ACM International Conf. on Information & Knowledge Management, pp. 78–87 (2004)

3. Charuvaka, A., Rangwala, H.: Approximate block coordinate descent for large scale hierarchical classification. In: ACM SIGAPP Symposium on Applied Computing (2015)
4. Chen, J., Warren, D.: Cost-sensitive learning for large-scale hierarchical classification. In: ACM International Conf. on Information & Knowledge Management, pp. 1351–1360 (2013)
5. Dekel, O., Keshet, J., Singer, Y.: Large margin hierarchical classification. In: International Conf. on Machine Learning, p. 27 (2004)
6. Dimitrovski, I., Kocev, D., Loskovska, S., Džeroski, S.: Hierarchical annotation of medical images. Pattern Recognition **44**(10), 2436–2449 (2011)
7. Evgeniou, T., Micchelli, C., Pontil, M.: Learning multiple tasks with kernel methods. Journal of Machine Learning Research **6**(1), 615–637 (2005)
8. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining, pp. 109–117 (2004)
9. Gopal, S., Yang, Y.: Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In: ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining, pp. 257–265 (2013)
10. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: Rcv1: A new benchmark collection for text categorization research. The Journal of Machine Learning Research **5**, 361–397 (2004)
11. Liu, T.Y., Yang, Y., Wan, H., Zeng, H.J., Chen, Z., Ma, W.Y.: Support vector machines classification with a very large-scale taxonomy. ACM SIGKDD Explorations Newsletter **7**(1), 36–43 (2005)
12. Masnadi-Shirazi, H., Vasconcelos, N.: Risk minimization, probability elicitation, and cost-sensitive svms. In: International Conf. on Machine Learning, pp. 759–766 (2010)
13. McCallum, A., Rosenfeld, R., Mitchell, T.M., Ng, A.Y.: Improving text classification by shrinkage in a hierarchy of classes. In: International Conf. on Machine Learning, vol. 98, pp. 359–367 (1998)
14. Nesterov, Y.: Introductory lectures on convex optimization, vol. 87. Springer Science & Business Media (2004)
15. Partalas, I., Kosmopoulos, A., Baskiotis, N., Artieres, T., Paliouras, G., Gaussier, E., Androutsopoulos, I., Amini, M.R., Galinari, P.: Lshtc: A benchmark for large-scale text classification. arXiv preprint arXiv:1503.08581 (2015)
16. Saha, B., Gupta, S., Phung, D., Venkatesh, S.: Multiple task transfer learning with small sample sizes. Knowledge and Information Systems, 1–28 (2015)
17. Shahbaba, B., Neal, R.M., et al.: Improving classification when a class hierarchy is available using a hierarchy-based prior. Bayesian Analysis **2**(1), 221–237 (2007)
18. Silla Jr., C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery **22**(1–2), 31–72 (2011)
19. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Information Processing & Management **45**(4), 427–437 (2009)
20. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research **6**, 1453–1484 (2005)
21. Yang, Y.: A study of thresholding strategies for text categorization. In: ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 137–145 (2001)
22. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 42–49 (1999)