Learning Hierarchical Multi-Category Text Classification Models

Juho Rousu JUHO AT CS.RHUL.AC.UK
Department of Computer Science, Royal Holloway University of London, TW20 0EX, United Kingdom

Craig Saunders CJS AT ECS.SOTON.AC.UK
Sandor Szedmak SS03V AT ECS.SOTON.AC.UK
John Shawe-Taylor JST AT ECS.SOTON.AC.UK
Electronics and Computer Science, University of Southampton, SO17 1BJ, United Kingdom

Abstract

We present a kernel-based algorithm for hierarchical text classification where the documents are allowed to belong to more than one category at a time. The classification model is a variant of the Maximum Margin Markov Network framework, where the classification hierarchy is represented as a Markov tree equipped with an exponential family defined on the edges. We present an efficient optimization algorithm based on incremental conditional gradient ascent in single-example subspaces spanned by the marginal dual variables. Experiments show that the algorithm can feasibly optimize training sets of thousands of examples and classification hierarchies consisting of hundreds of nodes. The algorithm's predictive accuracy is competitive with other recently introduced hierarchical multi-category or multilabel classification learning algorithms.

1. Introduction

In many application fields, taxonomies and hierarchies are natural ways to organize and classify objects, hence they are widely used for tasks such as text classification. In contrast, machine learning research has largely been focused on flat target prediction, where the output is a single binary or multivalued scalar variable. Naively encoding a large hierarchy either into a series of binary problems or a single multiclass problem with many possible class values suffers from the fact that dependencies between the classes cannot be represented well. For example, if a news article belongs to category MUSIC, it is very likely that the article belongs to category ENTERTAINMENT. The failure to represent these relationships leads to a steep decline of the predictive accuracy in the number of possible categories. In recent years, methods that utilize the hierarchy in learning the classification have been proposed by several authors (Koller & Sahami, 1997; McCallum et al., 1998; Dumais & Chen, 2000). Very recently, new hierarchical classification approaches utilizing kernel methods have been introduced (Hofmann et al., 2003; Cai & Hofmann, 2004; Dekel et al., 2004). The main idea behind these methods is to map the documents (or document–labeling pairs) into a potentially high-dimensional feature space where linear maximum margin separation of the documents becomes possible.

Most of the above mentioned methods assume that the object to be classified is assumed to belong to exactly one (leaf) node in the hierarchy. In this paper we consider the more general case where a single object can be classified into several categories in the hierarchy, to be specific, the multilabel is a *union of partial paths* in the hierarchy. For example, a news article about David and Victoria Beckham could belong to partial paths SPORT, FOOTBALL and ENTERTAINMENT, MUSIC but might not belong to any leaf categories such as CHAMPIONS LEAGUE. The problem of multiple partial paths was also considered in Cesa-Bianchi et al. (2004).

Recently Taskar et al. (2003) introduced a maximum margin technique which optimised an SVM-style objective function over structured outputs. This technique used a marginalisation trick to obtain a polynomial sized quadratic program using marginal dual variables. This was an improvement over the exponentially-sized problem resulting from the dualization of the primal margin maximization prob-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

lem, which only can be approximated with polynomial number of support vectors using a working set method (Altun et al., 2003; Tsochantaridis et al., 2004).

Even using marginal variables, however, the problem becomes infeasible for even medium sized data sets. Therefore, efficient optimization algorithms are needed. In this paper we present an algorithm for working with the marginal variables that is in the spirit of Taskar et al. (2003), however a reformulation of the objective allows a conditional-gradient method to be used which gains efficiency and also enables us to work with a richer class of loss functions.

The structure of this article is the following. In Section 2 we present the classification framework, review loss functions and derive a quadratic optimization problem for finding the maximum margin model parameters. In Section 3 we present an efficient learning algorithm relying a decomposition of the problem into single training example subproblems and conducting iterative conditional gradient ascent in marginal dual variable subspaces corresponding to single training examples. We compare the new algorithm in Section 4 to flat and hierarchical SVM learning approaches and the hierarchical regularized least squares algorithm recently proposed by Cesa-Bianchi et al. (2004). We conclude the article with discussion in Section 5.

2. Maximum Margin Hierarchical Multilabel Classification

When dealing with structured outputs, it is common to assume an exponential family over the labelings. Our setting therefore is as follows. The training data $((x_i, \mathbf{y}_i))_{i=1}^m$ consists of pairs $(x, \mathbf{y})$ of document x and a multilabel $\mathbf{y} \in \{+1,-1\}^k$ consisting of k microlabels. As the model class we use the exponential family

$$P(\mathbf{y}|x) \propto \prod_{e \in E} \exp\left(\mathbf{w}_e^T \boldsymbol{\phi}_e(x, \mathbf{y}_e)\right) = \exp\left(\mathbf{w}^T \boldsymbol{\phi}(x, \mathbf{y})\right) \tag{1}$$

defined on the edges of a Markov tree $T = (V, E)$, where node $j \in V$ corresponds to the j'th component of the multilabel and the edges $e = (j, j') \in E$ correspond to the classification hierarchy given as input. By $\mathbf{y}_e = (y_j, y_{j'})$ we denote the restriction of the multilabel $\mathbf{y} = (y_1, \ldots, y_k)$ to the edge $e = (j, j')$.

We use a similar feature vector structure to Altun et al. (2003). The *edge-feature* vector $\boldsymbol{\phi}_e$ is a concatenation of 'class-sensitive' feature vectors $\boldsymbol{\phi}_e^{\mathbf{u}}(x, \mathbf{y}_e) = [\![\mathbf{y}_e = \mathbf{u}_e]\!]\boldsymbol{\phi}(x)$, where $[\![\cdot]\!]$ denotes an indicator function. The vector $\boldsymbol{\phi}(x)$ could be a bag of words—as in the experiments reported here—or any other feature representation of the document x. Note that although the same feature vector $\boldsymbol{\phi}(x)$ is duplicated for each edge and edge-labeling, in the weight vector $\mathbf{w} = (\mathbf{w}_e^{\mathbf{u}_e})_{e \in E, \mathbf{u}_e}$ we still have separate weights to represent differences in the importance of a given feature in different contexts.

2.1. Loss Functions for Hierarchical Multilabel Classification

There are many ways to define loss functions for multilabel classification setting, and it depends on the application which loss function is the most suitable. A few general guidelines can be set, though. The loss function should obviously fulfil some basic conditions: $\ell(\hat{\mathbf{y}}, \mathbf{y}) = 0$ if and only if $\hat{\mathbf{y}} = \mathbf{y}$, $\ell(\hat{\mathbf{y}}, \mathbf{y})$ is maximum when $\hat{\mathbf{y}}_j \neq \mathbf{y}_j$ for every $1 \leq j \leq k$, and ℓ should be monotonically non-decreasing with respect to the sets of incorrect microlabels. These conditions are satisfied by, for example, *zero-one loss* $\ell_{0/1}(\mathbf{y}, \mathbf{u}) = [\![\mathbf{y} \neq \mathbf{u}]\!]$. However, it gives loss of 1 if the complete hierarchy is not labelled correctly, even if only a single microlabel was predicted incorrectly. In multilabel classification, we would like the loss to increase smoothly so that we can make a difference between 'nearly correct' and 'clearly incorrect' multilabel predictions. *Symmetric difference loss* $\ell_\Delta(\mathbf{y}, \mathbf{u}) = \sum_j [\![y_j \neq u_j]\!]$, has this property and is an obvious first choice as the loss function in structured classification tasks. However, the classification hierarchy is not reflected in any way in the loss. For *uni-category* hierarchical classification (Hofmann et al., 2003; Cai & Hofmann, 2004; Dekel et al., 2004), where exactly one of the microlabels has value 1, Dekel et al. (2004) use as a loss function the length of the path $(i_1, \cdots, i_k)$ between the the true and predicted nodes with positive microlabels $\ell_{PATH}(\mathbf{y}, \mathbf{u}) = |path(i : y_i = 1, j : u_j = 1)|$. Cai and Hofmann (2004) defined a weighted version of the loss that can take into account factors such as subscription loads of nodes.

In the union of partial paths model, where essentially we need to compare a predicted tree to the true one the concept of a path distance is not very natural. We would like to account for the incorrectly predicted subtrees—in the spirit of ℓ_Δ—but taking the hierarchy into account. Predicting the parent microlabel correctly is more important than predicting the child correctly, as the child may deal with some detailed concept that the user may not be interested in; for example whether a document was about CHAMPIONS LEAGUE football or not may not relevant to a person that is interested in FOOTBALL in general. Also, for the learners point of view, if the parent class was already predicted incorrectly, we don't want to penalize the mistake in the child. A loss function that has these properties was given by Cesa-Bianchi et al. (2004). It penalizes the first mistake along a path from root to a node

$$\ell_H(\mathbf{y},\mathbf{u}) = \sum_j c_j [\![y_j \neq u_j \ \ \& \ \ y_h = u_h \forall h \in anc(j)]\!],$$

where $anc(j)$ denotes the set of ancestors of node j. The coefficients $0 \leq c_j \leq 1$ are used for down-scaling the loss when going deeper in the tree. These can be chosen in many ways. One can divide the maximum loss among the subtrees met along the path. This is done by defining

$$c_{root} = 1, c_j = c_{pa(j)}/|sibl(j)|,$$

where we denoted by $pa(j)$ the immediate parent and by $sibl(j)$ the set of siblings of node j (including j itself). Another possibility is to scale the loss by the proportion of the hierarchy that is in the subtree $T(j)$ rooted by j, that is, to define

$$c_j = |T(j)|/|T(root)|.$$

In our experiments we use both the sibling and subtree scaling to re-weight prediction errors on individual nodes, these are referred to as *ℓ-sibl* and *ℓ-subtree* respectively. If we just use a uniform weighting ($c_j = 1$) in conjunction with the hierarchical loss above this is denoted as *ℓ-unif*.

Using ℓ_H for learning a model has the drawback that it does not decompose very well: the labelings of the complete path are needed to compute the loss. Therefore, in this paper we consider a simplified version of ℓ_H, namely

$$\ell_{\tilde{H}}(\mathbf{y},\mathbf{u}) = \sum_j c_j [\![y_j \neq u_j \ \ \& \ \ y_{pa(j)} = u_{pa(j)}]\!],$$

that penalizes a mistake in a child only if the label of the parent was correct. This choice leads the loss function to capture some of the hierarchical dependencies (between the parent and the child) but allows us define the loss in terms of edges, which is crucial for the efficiency of our learning algorithm.

Using the above, the per-microlabel loss is divided among the edges adjacent to the node. This is achieved by defining an *edge-loss* $\ell_e(\mathbf{y}_e,\mathbf{u}_e) = \ell_j(y_j,u_j)/\mathcal{N}(j) + \ell_{j'}(y_{j'},u_{j'})/\mathcal{N}(j')$ for each $e = (j,j')$, where ℓ_j is the term regarding microlabel j, $\mathbf{y}_e = (y_j, y_{j'})$ is a labeling of the edge e and $\mathcal{N}(j)$ denotes the neighbours of node j in the hierarchy (i.e. the children of a nodes and it's parent). Intuitively, the edges adjacent to node j 'share the blame' of the microlabel loss ℓ_j. The multilabel loss (ℓ_Δ or $\ell_{\tilde{H}}$) is then written as a sum over the edges: $\ell(\mathbf{y},\mathbf{u}) = \sum_{e \in E} \ell_e(\mathbf{y}_e,\mathbf{u}_e)$.

2.2. Maximum margin learning

As in Taskar et al. (2003) and Tsochantaridis et al. (2004), our goal is to learn a weight vector $\mathbf{w}$ that maximizes the minimum margin on training data the between the correct multilabel $\mathbf{y}_i$ and the incorrect multilabels $\mathbf{y} \neq \mathbf{y}_i$. Also, we would like the margin to scale as a function of the loss. Allotting a single slack variable for each training example results in the following soft-margin optimization problem:

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^m \xi_i$$
$$\text{s.t. } \mathbf{w}^T \Delta\boldsymbol{\phi}(x_i,\mathbf{y}) \geq \boldsymbol{\ell}(\mathbf{y}_i,\mathbf{y}) - \xi_i, \forall i, \mathbf{y} \quad (2)$$

where $\Delta\boldsymbol{\phi}(x_i,\mathbf{y}) = \boldsymbol{\phi}(x_i,\mathbf{y}_i) - \boldsymbol{\phi}(x_i,\mathbf{y})$. This optimization problem suffers from the possible high-dimensionality of the feature vectors. A dual problem

$$\max_{\boldsymbol{\alpha}>0} \boldsymbol{\alpha}^T\boldsymbol{\ell} - \frac{1}{2}\boldsymbol{\alpha}^T K \boldsymbol{\alpha}, \text{ s.t.} \sum_{\mathbf{y}} \alpha(i,\mathbf{y}) \leq C, \forall i, \mathbf{y} \quad (3)$$

where $K = \Delta\Phi^T\Delta\Phi$ is the *joint* kernel matrix for *pseudo-examples* $(x_i,\mathbf{y})$ and $\boldsymbol{\ell} = (\ell(\mathbf{y}_i,\mathbf{y}))_{i,\mathbf{y}}$ is the loss vector, allows us to circumvent the problem with feature vectors. However, in the dual problem there are exponentially many dual variables $\alpha(i,\mathbf{y})$, one for each pseudo-example. One can guarantee an approximate solution with a polynomial number of support vectors, though (Tsochantaridis et al., 2004).

For the loss functions ℓ_Δ and $\ell_{\tilde{H}}$ we can use the marginalization trick of Taskar et al. (2003) to obtain a polynomial-sized optimization problem with dual variables

$$\mu_e(i,\mathbf{y}_e) = \sum_{\{\mathbf{u}|\mathbf{u}_e=\mathbf{y}_e\}} \alpha(i,\mathbf{u}). \quad (4)$$

These variables can be seen as edge-marginals of the original dual variables. Applying (4) to (3) requires us to write the kernel and the loss in terms of the edges. The required loss vector is $\boldsymbol{\ell} = (\ell_e(\mathbf{y}_{e,i},u_e))_{i,e,\mathbf{u}_e}$ and the kernel decomposition is

$$K(i,\mathbf{y};i',\mathbf{y}') = \sum_{e\in E} \Delta\boldsymbol{\phi}_e(x_i,\mathbf{y}_e)^T \Delta\boldsymbol{\phi}_e(x'_i,\mathbf{y}'_e)$$
$$= \sum_{e\in E} K_e(i,\mathbf{y}_e;i',\mathbf{y}'_e),$$

where K_e is the joint kernel for edge e. With the vector $\boldsymbol{\mu} = (\mu_e(i,\mathbf{u}_e))_{i,e,\mathbf{u}_e}$ of marginal dual variables, loss vector $\boldsymbol{\ell}$ and the edge-kernels K_e the objective in (3) can be expressed as

$$\sum_{e\in E} \boldsymbol{\mu}_e^T\boldsymbol{\ell}_e - \frac{1}{2}\boldsymbol{\mu}_e^T K_e \boldsymbol{\mu}_e,$$

with the constraints

$$\sum_{\mathbf{u}_e} \mu_e(i, \mathbf{u}_e) \leq C, \forall i, e \in E. \tag{5}$$

However, in order to ensure that the marginal dual variables $\mu_e(i, \mathbf{y}_e)$ correspond to a valid $\alpha(i, \mathbf{y})$, additional constraints need to be inserted. To ensure overall consistency, for tree-shaped models it is sufficient to make sure that adjacent edges have consistent marginals (Taskar et al., 2003; Wainwright & Jordan, 2003). If two edges share a node j, they need to have equal node-marginals μ_j:

$$\sum_{y'} \mu_e(i, y, y') = \mu_j(i, y) = \sum_{y'} \mu_{e'}(i, y, y'). \tag{6}$$

To enforce this constraint, it suffices to pair up each edge with its parent which results in the set of edge pairs $E_2 = \{(e, e') \in E \times E | e = (p, i), e' = (i, j)\}$. By introduction of these marginal consistency constraints the optimization problem gets the form

$$\begin{aligned} &\max_{\boldsymbol{\mu}>0} \sum_{e \in E} \boldsymbol{\mu}_e^T \boldsymbol{\ell}_e - \frac{1}{2} \sum_{e \in E} \boldsymbol{\mu}_e^T K_e \boldsymbol{\mu}_e \qquad (7) \\ &\quad \text{s.t} \sum_{y,y'} \mu_e(i, y, y') \leq C, \forall i, e \in E, \\ &\quad \sum_{y'} \mu_e(i, y', y) = \sum_{y'} \mu_{e'}(i, y, y'),\ \forall i, y, (e, e') \in E_2, \end{aligned}$$

While the above formulation is closely related to that described in Taskar et al. (2003), there are a few differences to be pointed out. Firstly, as we assign the loss to the edges rather than the microlabels, we are able to use richer loss functions than the simple ℓ_Δ. Secondly, single-node marginal dual variables (the μ_j's in 6) become redundant when the constraints are given in terms of the edges. Thirdly, we have utilized the fact that in our feature representation the 'cross-edge' values $\Delta\boldsymbol{\phi}_e(x, \mathbf{y})^T \Delta\boldsymbol{\phi}_{e'}(x', \mathbf{y}')$ do not contribute to the kernel, hence we have a block-diagonal kernel $K_E = diag(K_{e_1}, \ldots, K_{e_{|E|}}), K_E(i, e, \mathbf{u}_e; j, e, \mathbf{v}_e) = K_e(i, \mathbf{u}_e; j, \mathbf{v}_e)$ with the number of non-zero entries thus scaling linearly rather than quadratically in the number of edges. Finally, we write the box constraint (5) as an inequality as we want the algorithm to be able to inactivate training examples (see Section 3.2).

Like that of Taskar et al. (2003), our approach can be generalized to non-tree structures. However, the feasible region in (7) will in general only approximate that of (3), which will give rise to a approximate solution to the primal. Also, finding the maximum likelihood multilabel can only be approximated tractably.

3. Efficient Optimization using Conditional Subspace Gradient Ascent

While the above quadratic program is polynomial-sized—and considerably smaller than that described in Taskar et al. (2003)—it is still easily too large in practise to fit in main memory or to solve by off-the-shelf QP solvers. To arrive at a more tractable problem, we notice that both the box constraints (5) and the marginal consistency constraints (6) are defined for each x separately, and they only depend on the edge set E, not on the training example in question. Thus, the constraints not only decompose by the training examples but the constraints are also identical for each example. However, the kernel matrix only composes by the edges. Thus there does not seem to be a straightforward way to decompose the quadratic programme.

A decomposition becomes possible when considering gradient-based approaches. Let us consider optimizing the dual variables $\boldsymbol{\mu}_i = (\mu_e(i, \mathbf{y}_e))_{e, \mathbf{y}_e}$ of example x_i. Let us denote by $K_{ij} = (K_e(i, \mathbf{u}_e; j, \mathbf{v}_e)_{e, \mathbf{u}_e, \mathbf{v}_e}$ the block of kernel values between examples i and j, and by $K_{i\cdot} = (K_{ij})_j$ the columns of the kernel matrix K_E referring to example i.

Obtaining the gradient for the x_i-subspace requires computing $\mathbf{g}_i = \boldsymbol{\ell}_i - K_{i\cdot}\boldsymbol{\mu}$ where $\boldsymbol{\ell}_i = (\ell_e(i, \mathbf{u}_e))_{e, \mathbf{u}_e}$ is the loss vector for x_i. However, when updating $\boldsymbol{\mu}_i$ only, evaluating the change in objective and updating the gradient can be done more cheaply: $\Delta\mathbf{g}_i = -K_{ii}\Delta\boldsymbol{\mu}_i$ and $\Delta obj = \boldsymbol{\ell}_i^T \Delta\boldsymbol{\mu}_i - 1/2\Delta\boldsymbol{\mu}_i K_{ii} \Delta\boldsymbol{\mu}_i$. Thus local optimization in a subspace of a single training example can be done without consulting the other training examples. On the other hand, we do not want to spend too much time in optimizing a single example: When the dual variables of the other examples are non-optimal, so is the initial gradient $\mathbf{g}_i$. Thus the optimum we would arrive at would not be the global optimum of the quadratic objective. It makes more sense to optimize all examples more or less in tandem so that the full gradient approaches its optimum as quickly as possible.

In our approach, we have chosen to conduct a few optimization steps for each training example using a conditional gradient ascent (see Section 3.1) before moving on to the next example. The iteration limit for each example is set by using the KKT conditions as a guideline (see Section 3.2).

The pseudocode of our algorithm is given in Algorithm 1. It takes as input the training data, the edge set of the hierarchy, the loss vector $\boldsymbol{\ell} = (\boldsymbol{\ell}_i)_{i=1}^m$ and the constraints defining the feasible region. The algorithm chooses a chunk of examples as the working set, computes the x-kernel and makes an optimization pass over the chunk. After one pass, the gradient, slacks and the duality gap are computed and a new chunk is picked. The process is iterated until the duality gap gets below given threshold.

Note in particular, that the joint kernel is not explicitly computed, although evaluating the gradient requires computing the product $K\boldsymbol{\mu}$. However, we are able to take advantage of the special structure of the feature vectors–repeating the same feature vector in different contexts–to facilitate the computation using the x-kernel $K_X(i,j) = \boldsymbol{\phi}(x_i)^T\boldsymbol{\phi}(x_j)$ and the dual variables only.

Algorithm 1 Maximum margin optimization algorithm for the H-M^3 hierarchical classification model.

H-M$^3(S, E, \boldsymbol{\ell}, \mathcal{F})$

Inputs: Training data $S = ((x_i, \mathbf{y}_i))_{i=1}^m$, edge set E of the hierarchy, a loss vector $\boldsymbol{\ell}$, constraint matrix A and vector b.

Outputs: Dual variable vector $\boldsymbol{\mu}$ and objective value obj.

Initialize $\mathbf{g} = \boldsymbol{\ell}$, $\boldsymbol{\xi} = \boldsymbol{\ell}$,$dg = \infty$ and $OBJ = 0$.
while $dg > dg_{min}$ & $iter < max\_iter$ **do**
  $[WS, Freq] =$ UpdateWorkingSet$(\boldsymbol{\mu}, \mathbf{g}, \boldsymbol{\xi})$;
  Compute x-kernel values $K_{X,WS} = K_X(\cdot,\cdot; WS, \cdot)$;
  for $i \in WS$ **do**
    Compute joint kernel block K_{ii} and subspace gradient $\mathbf{g}_i$;
    $[\boldsymbol{\mu}_i, \Delta obj] =$ CSGA$(\boldsymbol{\mu}_i, \mathbf{g}_i, K_i, \mathcal{F}, Freq_i)$;
  end for
  Compute gradient $\mathbf{g}$, slacks $\boldsymbol{\xi}$ and duality gap dg;
end while

3.1. Optimization in single example subspaces

The optimization algorithm used for a single example is a variant of conditional gradient ascent (or descent) algorithms (Bertsekas, 1999). The algorithms in this family solve a constrained quadratic problem by iteratively stepping to the best feasible direction with respect to the current gradient.

The pseudocode of our variant CSGA is given in Algorithm 2. The algorithm takes as input the current dual variables, gradient, constraints and the kernel block for the example, and an iteration limit. It outputs new values for the dual variables and the change in objective value. As discussed above, the iteration limit is set very tight so that only a few iterations will be typically conducted.

For choosing the step length, c we take the optimal solution, namely we look for the saddle point along the ray $\boldsymbol{\mu}_i(c) = \boldsymbol{\mu}_i + c\Delta\boldsymbol{\mu}, c > 0$, where $\Delta\boldsymbol{\mu} = \boldsymbol{\mu}'_i - \boldsymbol{\mu}_i$ is the line segment between the highest feasible point along the gradient and the current point. The saddle point is found by solving

$$\frac{\mathbf{d}}{\mathbf{d}c}\boldsymbol{\ell}_i^T\boldsymbol{\mu}_i(c) - 1/2\boldsymbol{\mu}_i(c)^T K_i \boldsymbol{\mu}_i(c) = 0.$$

If $c > 1$, the saddle point is infeasible and the feasible maximum is obtained at $c = 1$. In our experience, the time taken to compute the saddle point was negligible compared to finding $\boldsymbol{\mu}'_i$, which in implementation was done by Matlab's linear interior point solver LIPSOL.

Algorithm 2 Conditional subspace gradient ascent optimization step.

CSGA$(\boldsymbol{\mu}_0, \mathbf{g}, K, \mathcal{F}, maxiter)$

Inputs: Initial dual variable vector $\boldsymbol{\mu}_0$, gradient $\mathbf{g}$, constraints of the feasible region $\mathcal{F}$, a joint kernel block K for the subspace, and an iteration limit $maxiter$.

Outputs: New values for dual variables $\boldsymbol{\mu}$ and change in objective Δobj.

$\boldsymbol{\mu} = \boldsymbol{\mu}_0; \Delta obj = 0; iter = 0;$
while $iter < maxiter$ **do**
  % find highest feasible point given $\mathbf{g}$
  $\boldsymbol{\mu} = \mathrm{argmax}_{\mathbf{v}\in\mathcal{F}}\ \mathbf{g}^T\mathbf{v}$;
  $\Delta\boldsymbol{\mu} = \boldsymbol{\mu} - \boldsymbol{\mu}_0$;
  $l = \mathbf{g}_x^T\Delta\boldsymbol{\mu}; q = \Delta\boldsymbol{\mu}^T K \Delta\boldsymbol{\mu}$; % saddle point
  $c = \min(l/q, 1)$; % clip to remain feasible
  if $c \le 0$ **then**
    break; % no progress, stop
  else
    $\boldsymbol{\mu}_0 = \boldsymbol{\mu}; \boldsymbol{\mu} = \boldsymbol{\mu} + c\Delta\boldsymbol{\mu}$; % update
    $\mathbf{g}_x = \mathbf{g}_x - cK\Delta\boldsymbol{\mu}$;
    $\Delta obj = \Delta obj + cl - c^2q/2$;
  end if
  $iter = iter + 1$;
end while

3.2. Working set maintenance

We wish to maintain the working set so that the most promising examples to be updated are contained there at all times to minimize the amount of computation used for unsuccessful updates. Our working set update is based on the KKT conditions which at optimum hold for all x_i:

1. $(C - \sum_{e,\mathbf{y}_e}\mu_e(i,\mathbf{y}_e))\xi_i = 0$, and
2. $\alpha(i,\mathbf{y})(\mathbf{w}^T\phi(x_i,\mathbf{y}) - \ell(x_i,y) + \xi_i) = 0$.

The first condition states that, at optimum, only examples that saturate the box constraint can have positive slack, and consequently a pseudo-example that has a negative margin. The second condition states that pseudo-examples with non-zero dual variables are those that have the minimum margin, that is, need the full slack ξ_i. Consequently, if all pseudo-examples of x_i have positive margin, all dual variables satisfy $\alpha(i, \mathbf{y}) = 0$. This observation leads to the following heuristics for the working set update:

- Non-saturated ($\sum_{e,\mathbf{y}_e} \mu_e(i, \mathbf{y}_e) < C$) examples are given priority as they certainly will need to be updated to reach the optimum.
- Saturated examples ($\sum_{e,\mathbf{y}_e} \mu_e(i, \mathbf{y}_e) = C$) are added if there are not enough non-saturated ones. The rationale is that the even though an example is saturated, the individual dual variable values may still be suboptimal.
- Inactive ($\sum_{e,\mathbf{y}_e} \mu_e(i, \mathbf{y}_e) = 0$) non-violators ($\xi_i = 0$) are removed from the working set, as they do not constrain the objective.

Another heuristic technique to concentrate computational effort to most promising examples is to favour examples with a large duality gap

$$\Delta obj(\boldsymbol{\mu}, \boldsymbol{\xi}) = \sum_i C\xi_i + \boldsymbol{\mu}_{x_i}^T \mathbf{g}_{x_i}.$$

As feasible primal solutions always are least as large as feasible dual solutions, the duality gap gives an upper bound to the distance from the dual solution to the optimum. We use the quantity $\Delta_i = C\xi + \boldsymbol{\mu}_{x_i}^T \mathbf{g}_{x_i}$ as a heuristic measure of the work needed for that particular example in order to reach the optimum. Examples are then chosen to the chunk to be updated with probability proportional to $p_i \propto \Delta_i - \min_j \Delta_j$. An example that is drawn more than once will be set a higher iteration limit for the next optimization step.

4. Experiments

We tested the presented learning approach on two publicly available document collection that have an associated classification hierarchy:

- Reuters Corpus Volume 1, RCV1 (Lewis et al., 2004). 2500 documents were used for training and 5000 for testing. As the label hierarchy we used the 'CCAT' family of categories, which had a total of 34 nodes, organized in a tree with maximum depth 3. The tree is quite unbalanced, half of the nodes residing in depth 1.
- WIPO-alpha patent dataset (WIPO, 2001). The dataset consisted of the 1372 training and 358 testing document comprising the D section of the hierarchy. The number of nodes in the hierarchy was 188, with maximum depth 3. Each document in this dataset belongs to exactly one leaf category, hence it contains no multiple or partial paths.

Both datasets were processed into bag-of-words representation with TFIDF weighting. No word stemming or stop-word removal was performed.

We compared the performance of the presented learning approach—below denoted by H-M^3—to three algorithms: SVM denotes an SVM trained for each microlabel separately, H-SVM denotes the case where the SVM for a microlabel is trained only with examples for which the ancestor labels are positive. H-RLS is the hierarchical least squares algorithm described in Cesa-Bianchi et al. (2004). It essentially solves for each node i a least squares style problem $\mathbf{w}_i = (I + S_i S_i^T + xx^T)^{-1} S_i \mathbf{y}_i$, where S_i is a matrix consisting of all training examples for which the parent of node i was classified as positive, $\mathbf{y}_i$ is a microlabel vector for node i of those examples and I is an identity matrix. Predictions for a node i for a new example x is -1 if the parent of the node was classified negatively and $\mathrm{sign}(\mathbf{w}_\mathrm{i}^\mathrm{T}\mathrm{x})$ otherwise.

The algorithms were implemented in MATLAB and the tests were run on a high-end PC. For SVM,H-SVM and H-M^3, the regularization parameter value $C = 1$ was used in all experiments.

Obtaining consistent labelings. As the learning algorithms compared here all decompose the hierarchy for learning, the multilabel composed of naively combining the microlabel predictions may be inconsistent, that is, they may predict a document as part of the child but not as part of the parent. For SVM and H-SVM consistent labelings were produced by post-processing the predicted labelings as follows: start at the root and traverse the tree in a breadth-first fashion. If the label of a node is predicted as -1 then all descendants of that node are also labelled negatively. This post-processing turned out to be crucial to obtain good accuracy, thus we only report results with the postprocessed labelings. Note that H-RLS performs essentially the same procedure (see above). For H-M^3 models, we computed by dynamic programming the consistent multilabel with maximum likelihood

$$\hat{\mathbf{y}}(x) = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}_T} P(\mathbf{y}|x) = \mathrm{argmax}_{\mathbf{y}} \mathbf{w}^T \boldsymbol{\phi}(x, \mathbf{y}),$$

where $\mathcal{Y}_T$ is the set of multilabels that correspond to unions of partial paths in T.

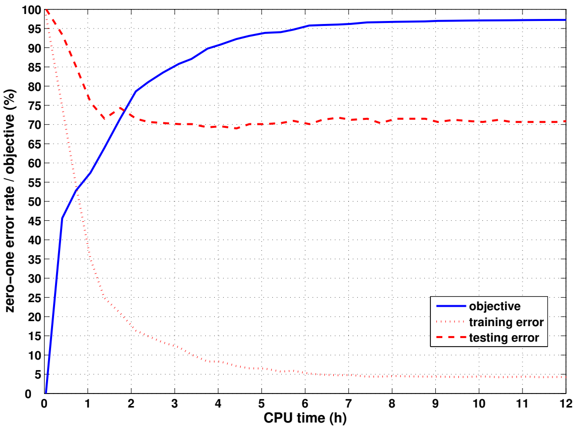

Figure 1. The objective function (% of optimum) and zero-one error rates for H-M^3 on training and test sets (WIPO-alpha). $\ell_{\tilde{H}}$ loss with no scaling was used for training.

Table 1. Prediction losses obtained using different training losses on Reuter's (top) and WIPO-alpha data (bottom). The loss $\ell_{0/1}$ is given as a percentage, the other losses as averages per-example.

	Test loss				
	$\ell_{0/1}$	ℓ_Δ	$\ell_{\tilde{H}}$ +scaling		
Tr. loss	%		unif	sibl.	subtree
ℓ_Δ	27.1	0.574	0.344	0.114	0.118
$\ell_{\tilde{H}}$-unif	26.8	0.590	0.338	0.118	0.122
$\ell_{\tilde{H}}$-sibl.	28.2	0.608	0.381	0.109	0.114
$\ell_{\tilde{H}}$-subtree	27.9	0.588	0.373	0.109	0.109
	$\ell_{0/1}$	ℓ_Δ	$\ell_{\tilde{H}}$ +scaling		
Tr. loss	%		unif	sibl.	subtree
ℓ_Δ	70.9	1.670	0.891	0.050	0.070
$\ell_{\tilde{H}}$-unif.	70.1	1.721	0.888	0.052	0.074
$\ell_{\tilde{H}}$-sibl.	64.8	1.729	0.927	0.048	0.071
$\ell_{\tilde{H}}$-subtree	65.0	1.709	0.919	0.048	0.072

Efficiency of optimization. To give an indication of the efficiency of the H-M^3 algorithm, Figure 1 shows an example learning curve on WIPO-alpha dataset. The number of dual variables for this training set is just over one million with a joint kernel matrix with approx 5 billion entries. Note that the solutions for this optimisation are not sparse, typically less than 25% of the marginal dual variables are zero. The training error obtains its minimum after ca. 6 hours of CPU time, when the objective was around 95% of the maximum. Testing error ($\ell_{0/1}$ loss) reaches close to its minimum after 2 hours, suggesting the possibility of early stopping. In our experiments, the running time of H-RLS, although faster than H-M^3, was in the same order of magnitude whereas SVM and H-SVM are expectedly much faster than the other two algorithms.

Table 2. Prediction loss, precision, recall and F1 values obtained using different learning algorithms on Reuter's (top) and WIPO-alpha data (bottom). The loss $\ell_{0/1}$ is given as a percentage, the other losses as averages per-example. Precision and recall are computed in terms of totals of microlabel predictions in the test set.

Alg.	$\ell_{0/1}$	ℓ_Δ	ℓ_H	P	R	F1
SVM	32.9	0.61	0.099	94.6	58.4	72.2
H-SVM	29.8	0.57	0.097	92.3	63.4	75.1
H-RLS	28.1	0.55	0.095	91.5	65.4	76.3
H-M^3-ℓ_Δ	27.1	0.58	0.114	91.0	64.1	75.2
H-M^3-$\ell_{\tilde{H}}$	27.9	0.59	0.109	85.4	68.3	75.9
Alg.	$\ell_{0/1}$	ℓ_Δ	ℓ_H	P	R	F1
SVM	87.2	1.84	0.053	93.1	58.2	71.6
H-SVM	76.2	1.74	0.051	90.3	63.3	74.4
H-RLS	72.1	1.69	0.050	88.5	66.4	75.9
H-M^3-ℓ_Δ	70.9	1.67	0.050	90.3	65.3	75.8
H-M^3-$\ell_{\tilde{H}}$	65.0	1.73	0.048	84.1	70.6	76.7

Effect of choice of the loss function. In order to show the effect of training the H-M^3 algorithm using the different loss functions described in Section 2.1, we compared the performance of the algorithm on both the Reuters and WIPO data sets. The results can be seen in Table 1. The WIPO dataset would suggest that using a hierarchical loss function during training (e.g. either $\ell_{\tilde{H}}$-sibl. or $\ell_{\tilde{H}}$-subtree) leads to a reduced 0/1 loss on the test set. On Reuters dataset this effect is not observed, however this is due to the fact that the label tree of the Reuters data set is very shallow.

Comparison to other learning methods. In our final test we compare the predictive accuracy of H-M^3 to other learning methods. For H-M^3 we include the results for training with ℓ_Δ and $\ell_{\tilde{H}}$-*subtree* losses. For training SVM and H-SVM, these losses produce the same learned model.

Table 2 depicts the different test losses, as well as the standard information retrieval statistics precision (P), recall (R) and F1 statistic ($F1 = 2PR/(P+R)$). Precision and recall were computed over all microlabel predictions in the test set. Flat SVM is expectedly inferior to the competing algorithms with respect to most statistics, as it cannot utilize the dependencies between the microlabels in any way. The two variants of H-M^3 are the most efficient in getting the complete tree correct as shown by the low zero-one loss. With respect to other statistics, the hierarchical methods are quite evenly matched overall.

Finally, to highlight the differences between the predicted labelings, we computed level-wise precision and recall values, that is, the set of predictions contained all test instances and microlabels on a given level of the

Table 3. Precision/Recall statistics for each level of the hierarchy for different algorithms on Reuters RCV1 (top) and WIPO-alpha (bottom) datasets.

Alg.	Level 0	Level 1	Level 2	Level 3
SVM	92.4/89.4	96.8/38.7	98.1/49.3	81.8/46.2
H-SVM	92.4/89.4	93.7/43.6	91.1/61.5	72.0/46.2
H-RLS	93.2/89.1	90.9/46.8	89.7/64.8	76.0/48.7
H-M^3-ℓ_Δ	94.1/83.0	87.3/48.9	91.1/63.2	79.4/69.2
H-M^3-$\ell_{\tilde{H}}$	91.1/87.8	79.2/53.1	85.4/66.6	77.9/76.9

Alg.	Level 0	Level 1	Level 2	Level 3
SVM	100/100	92.1/77.7	84.4/42.5	82.1/12.8
H-SVM	100/100	92.1/77.7	79.6/51.1	77.0/24.3
H-RLS	100/100	91.3/79.1	78.2/57.0	72.6/29.6
H-M^3-ℓ_Δ	100/100	90.8/80.2	86.1/50.0	72.1/31.0
H-M^3-$\ell_{\tilde{H}}$	100/100	90.9/80.4	76.4/62.3	60.4/39.7

tree (Table 3). On both datasets, recall of all methods, especially with SVM and H-SVM, diminishes when going farther from the root. H-M^3 is the most efficient method in fighting the recall decline, and is still able to obtain reasonable precision.

5. Conclusions and Future Work

In this paper we have proposed a new method for training variants of the Maximum Margin Markov Network framework for hierarchical multi-category text classification models.

Our method relies on a decomposition of the problem into single-example subproblems and conditional gradient ascent for optimisation of the subproblems. The method scales well to medium-sized datasets with label matrix (examples $\times$ microlabels) size upto hundreds of thousands, and via kernelization, very large feature vectors for the examples can be used. Initial experimental results on two text classification tasks show that using the hierarchical structure of multi-category labelings leads to improved performance over the more traditional approach of combining individual binary classifiers.

Our future work includes generalization of the approach to general graph structures and looking for ways to scale up the method further.

Acknowledgements

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. Juho Rousu has been supported by the European Union Marie Curie Fellowship grant HPMF-CT-2002-02110.

References

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden markov support vector machines. *ICML'03* (pp. 3–10).

Bertsekas, D. (1999). *Nonlinear programming*. Athena Scientific.

Cai, L., & Hofmann, T. (2004). Hierarchical document categorization with support vector machines. *13 ACM CIKM*.

Cesa-Bianchi, N., Gentile, C., Tironi, A., & Zaniboni, L. (2004). Incremental algorithms for hierarchical classification. *Neural Information Processing Systems*.

Dekel, O., Keshet, J., & Singer, Y. (2004). Large margin hierarchical classification. *ICML'04* (pp. 209–216).

Dumais, S. T., & Chen, H. (2000). Hierarchical classification of web content. *SIGIR'00* (pp. 256–263).

Hofmann, T., Cai., L., & Ciaramita, M. (2003). Learning with taxonomies: Classifying documents and words. *NIPS Workshop on Syntax, Semantics, and Statistics*.

Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. *ICML'97* (pp. 170–178).

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

McCallum, A., Rosenfeld, R., Mitchell, T., & Ng, A. Y. (1998). Improving text classification by shrinkage in a hierarchy of classes. *ICML'98* (pp. 359–367).

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. *Neural Information Processing Systems*.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. *ICML'04* (pp. 823–830).

Wainwright, M., & Jordan, M. (2003). *Graphical models, exponential families, and variational inference* (Technical Report 649). Department of Statistics, University of California, Berkeley.

WIPO (2001). *World intellectual property organization. http://www.wipo.int/classifications/en.*