Mach Learn (2012) 88:127–155
DOI 10.1007/s10994-012-5291-x

Efficient max-margin multi-label classification with applications to zero-shot learning

Bharath Hariharan · S.V.N. Vishwanathan · Manik Varma

Received: 20 October 2010 / Accepted: 11 April 2012 / Published online: 3 May 2012

Abstract The goal in multi-label classification is to tag a data point with the subset of relevant labels from a pre-specified set. Given a set of L labels, a data point can be tagged with any of the 2^L possible subsets. The main challenge therefore lies in optimising over this exponentially large label space subject to label correlations.

Our objective, in this paper, is to design efficient algorithms for multi-label classification when the labels are densely correlated. In particular, we are interested in the zero-shot learning scenario where the label correlations on the training set might be significantly different from those on the test set.

We propose a max-margin formulation where we model prior label correlations but do not incorporate pairwise label interaction terms in the prediction function. We show that the problem complexity can be reduced from exponential to linear while modelling dense pairwise prior label correlations. By incorporating relevant correlation priors we can handle mismatches between the training and test set statistics. Our proposed formulation generalises the effective 1-vs-All method and we provide a principled interpretation of the 1-vs-All technique.

We develop efficient optimisation algorithms for our proposed formulation. We adapt the Sequential Minimal Optimisation (SMO) algorithm to multi-label classification and show that, with some book-keeping, we can reduce the training time from being super-quadratic to almost linear in the number of labels. Furthermore, by effectively re-utilizing the kernel cache and jointly optimising over all variables, we can be orders of magnitude faster than

Editors: Grigorios Tsoumakas, Min-Ling Zhang, and Zhi-Hua Zhou.

B. Hariharan
University of California at Berkeley, Berkeley, USA
e-mail: bharath2@eecs.berkeley.edu

S.V.N. Vishwanathan
Purdue University, West Lafayette, USA
e-mail: vishy@stat.purdue.edu

M. Varma (✉)
Microsoft Research India, Bangalore, India
e-mail: manik@microsoft.com

the competing state-of-the-art algorithms. We also design a specialised algorithm for linear kernels based on dual co-ordinate ascent with shrinkage that lets us effortlessly train on a million points with a hundred labels.

Keywords Multi-label classification · Zero-shot learning · Max-margin methods · SMO optimization · 1-vs-all classification

1 Introduction

Our objective, in this paper, is to develop efficient algorithms for max-margin, multi-label classification. Given a set of pre-specified labels and a data point, (binary) multi-label classification deals with the problem of predicting the subset of labels most relevant to the data point. This is in contrast to multi-class classification where one has to predict just the single, most probable label. For instance, rather than simply saying that Fig. 1 is an image of a Babirusa we might prefer to describe it as containing a brown, hairless, herbivorous, medium sized quadruped with tusks growing out of its snout.

There are many advantages in generating such a description and multi-label classification has found applications in areas ranging from computer vision to natural language processing to bio-informatics. We are specifically interested in the problem of image search on the web and in personal photo collections. In such applications, it is very difficult to get training data for every possible object out there in the world that someone might conceivably search for. In fact, we might not have any training images whatsoever for many object categories such as the obscure Babirusa. Nevertheless, we can not preclude the possibility of someone searching for one of these objects. A similar problem is encountered when trying to search videos on the basis of human body pose and motion and many other applications such as neural activity decoding (Palatucci et al. 2009).

One way of recognising object instances from previously unseen test categories (the zero-shot learning problem) is by leveraging knowledge about common attributes and shared parts. For instance, given adequately labelled training data, one can learn classifiers for the attributes occurring in the training object categories. These classifiers can then be used to recognise the same attributes in object instances from the novel test categories. Recognition can then proceed on the basis of these learnt attributes (Farhadi et al. 2009, 2010; Lampert et al. 2009).

The learning problem can therefore be posed as multi-label classification where there is a significant difference between attribute (label) correlations in the training categories and the previously unseen test categories. What adds to the complexity of the problem is the fact that these attributes are often densely correlated as they are shared across most categories. This makes optimising over the exponentially large output space, given by the power set of all labels, very difficult. The problem is acute not just during prediction but also during training as the number of training images might grow to be quite large over time in some applications.

Previously proposed solutions to the multi-label problem take one of two approaches—neither of which can be applied straight forwardly in our scenario. In the first, labels are *a priori* assumed not to be correlated so that a predictor can be trained for each label independently. This reduces training and prediction complexity from exponential in the number of labels to linear. Such methods can therefore scale efficiently to large problems but at the cost of not being able to model label correlations. Furthermore, these methods typically tend not to minimise a multi-label loss. In the second, label correlations are explicitly taken into

Fig. 1 Having never seen a Babirusa before we can still describe it as a brown, hairless, herbivorous, medium sized quadruped with tusks growing out of its snout

account by incorporating pairwise, or higher order, label interactions. However, exact inference is mostly intractable for densely correlated labels and in situations where the label correlation graph has loops. Most approaches therefore assume sparsely correlated labels such as those arranged in a hierarchical tree structure.

In this paper, we follow a middle approach. We develop a max-margin multi-label classification formulation, referred to as M3L, where we do model prior label correlations but do not incorporate pairwise, or higher order, label interaction terms in the prediction function. This lets us generalise to the case where the training label correlations might differ significantly from the test label correlations. We can also efficiently handle densely correlated labels. In particular, we show that under fairly general assumptions of linearity, the M3L primal formulation can be reduced from having an exponential number of constraints to linear in the number of labels. Furthermore, if no prior information about label correlations is provided, M3L reduces directly to the 1-vs-All method. This lets us provide a principled interpretation of the 1-vs-All multi-label approach which has enjoyed the reputation of being a popular, effective but nevertheless, heuristic technique.

Much of the focus of this paper is on optimising the M3L formulation. It turns out that it is not good enough to just reduce the primal to have only a linear number of constraints. A straight forward application of state-of-the-art decompositional optimisation methods, such as Sequential Minimal Optimisation (SMO), would lead to an algorithm that is super-quadratic in the number of labels. We therefore develop specialised optimisation algorithms that can be orders of magnitude faster than competing methods. In particular, for kernelised M3L, we show that by simple book keeping and delaying gradient updates, SMO can be adapted to yield a linear time algorithm. Furthermore, due to efficient kernel caching and jointly optimising all variables, we can sometimes be an order of magnitude faster than the 1-vs-All method. Thus our code, available from Hariharan et al. (2010a), should also be very useful for learning independent 1-vs-All classifiers. For linear M3L, we adopt a dual co-ordinate ascent strategy with shrinkage which lets us efficiently tackle large scale training data sets. In terms of prediction accuracy, we show that incorporating prior knowledge about label correlations using the M3L formulation can substantially boost performance over independent methods.

The rest of the paper is organised as follows. Related work is reviewed in Sect. 2. Section 3 develops the M3L primal formulation and shows how to reduce the number of primal constraints from exponential to linear. The 1-vs-All formulation is also shown to be a special case of the M3L formulation. The M3L dual is developed in Sect. 4 and optimised in

Sect. 5. We develop algorithms tuned to both the kernelised and the linear case. Experiments are carried out in Sect. 7 and it is demonstrated that the M3L formulation can lead to significant gains in terms of both optimisation and prediction accuracy. An earlier version of the paper appeared in Hariharan et al. (2010b).

2 Related Work

The multi-label problem has many facets including binary (Tsoumakas and Katakis 2007; Ueda and Saito 2003), multi-class (Dekel and Shamir 2010) and ordinal (Cheng et al. 2010) multi-label classification as well as semi-supervised learning, feature selection (Zhang and Wang 2009b), active learning (Li et al. 2004), multi-instance learning (Zhang and Wang 2009a), *etc*. Our focus, in this paper, is on binary multi-label classification where most of the previous work can be categorised into one of two approaches depending on whether labels are assumed to be independent or not. We first review approaches that do assume label independence. Most of these methods try and reduce the multi-label problem to a more "canonical" one such as regression, ranking, multi-class or binary classification.

In regression methods (Hsu et al. 2009; Ji et al. 2008; Tsoumakas and Katakis 2007), the label space is mapped onto a vector space (which might sometimes be a shared subspace of the feature space) where regression techniques can be applied straightforwardly. The primary advantage of such methods is that they can be extremely efficient if the mapped label space has significantly lower dimensionality than the original label space (Hsu et al. 2009). The disadvantage of such approaches is that the choice of an appropriate mapping might be unclear. As a result, minimising regression loss functions, such as square loss, in this space might be very efficient but might not be strongly correlated with minimising the desired multi-label loss. Furthermore, classification involves inverting the map which might not be straightforward, result in multiple solutions and might involve heuristics.

A multi-label problem with L labels can be viewed as a classification problem with 2^L classes (McCallum 1999; Boutell et al. 2004) and standard multi-class techniques can be brought to bear. Such an approach was shown to give the best empirical results in the survey by Tsoumakas and Katakis (2007). However, such approaches have three major drawbacks. First, since not all 2^L label combinations can be present in the training data, many of the classes will have no positive examples. Thus, predictors can not be learnt for these classes implying that these label combinations can not be recognised at run time. Second, the 0/1 multi-class loss optimised by such methods forms a poor approximation to most multi-label losses. For instance, the 0/1 loss would charge the same penalty for predicting all but one of the labels correctly as it would for predicting all of the labels incorrectly. Finally, learning and predicting with such a large number of classifiers might be very computationally expensive.

Binary classification can be leveraged by replicating the feature vector for each data point L times. For copy number l, an extra dimension is added to the feature vector with value l and the training label is $+1$ if label l is present in the label set of the original point and -1 otherwise. A binary classifier can be learnt from this expanded training set and a novel point classified by first replicating it as described above and then applying the binary classifier L times to determine which labels are selected. Due to the data replication, applying a binary classifier naively would be computationally costly and would require that complex decision boundaries be learnt. However, Schapire and Singer (2000) show that the problem can be solved efficiently using Boosting. A somewhat related technique is 1-vs-All (Rifkin and Khautau 2004) which independently learns a binary classifier for each label. As we'll show in Sect. 3, our formulation generalises 1-vs-All to handle prior label correlations.

A ranking based solution was proposed in Elisseeff and Weston (2001). The objective was to ensure that, for every data point, all the relevant labels were ranked higher than any of the irrelevant ones. This approach has been influential but suffers from the drawback of not being able to easily determine the number of labels to select in the ranking. The solution proposed in Elisseeff and Weston (2001) was to find a threshold so that all labels scoring above the threshold were selected. The threshold was determined using a regressor trained subsequently on the ranker output on the training set. Many variations have been proposed, such as using dummy labels to determine the threshold, but each has its own limitations and no clear choice has emerged. Furthermore, posing the problem as ranking induces a quadratic number of constraints per example which leads to a harder optimisation problem. This is ameliorated in Crammer and Singer (2003) who reduced the space complexity to linear and time complexity to sub-quadratic.

Most of the approaches mentioned above do not explicitly model label correlations—McCallum (1999) has a generative model which can, in principle, handle correlations but greedy heuristics are used to search over the exponential label space. In terms of discriminative methods, most work has focused on hierarchical tree, or forest, structured labels. Methods such as Cai and Hofmann (2007), Cesa-Bianchi et al. (2006) optimise a hierarchical loss over the tree structure but do not incorporate pairwise, or higher order, label interaction terms. In both these methods, a label is predicted only if its parent has also been predicted in the hierarchy. For instance, Cesa-Bianchi et al. (2006) train a classifier for each node of the tree. The positive training data for the classifier is the set of data points marked with the node label while the negative training points are selected from the sibling nodes. Classification starts at the root and all the children classifiers are tested to determine which path to take. This leads to a very efficient algorithm during both training and prediction as each classifier is trained on only a subset of the data. Alternatively, Cai and Hofmann (2007) classify at only the leaf nodes and use them as a proxy for the entire path starting from the root. A hierarchical loss is defined and optimised using the ranking method of Elisseeff and Weston (2001).

The M3N formulation of Taskar et al. (2003) was the first to suggest max-margin learning of label interactions. The original formulation starts off having an exponential number of constraints. These can be reduced to quadratic if the label interactions formed a tree or forest. Approximate algorithms are also developed for sparse, loopy graph structures. While the M3N formulation dealt with the Hamming loss, a more suitable hierarchical loss was introduced and efficiently optimised in Rousu et al. (2006) for the case of hierarchies. Note that even though these methods take label correlations explicitly into account, they are unsuitable for our purposes as they cannot handle densely correlated labels and learn training set label correlations which are not useful at test time since the statistics might have changed significantly.

Finally, Tsochantaridis et al. (2005) propose an iterative, cutting plane algorithm for learning in general structured output spaces. The algorithm adds the worst violating constraint to the active set in each iteration and is proved to take a maximum number of iterations independent of the size of the output space. While this algorithm can be used to learn pairwise label interactions it too can't handle a fully connected graph as the worst violating constraint cannot be generally found in polynomial time. However, it can be used to learn our proposed M3L formulation but is an order of magnitude slower than the specialised optimisation algorithms we develop.

Zero shot learning deals with the problem of recognising instances from novel categories that were not present during training. It is a nascent research problem and most approaches tackle it by building an intermediate representation leveraging attributes, features or classifier outputs which can be learnt from the available training data (Farhadi et al. 2009, 2010;

Lampert et al. 2009; Palatucci et al. 2009). Novel instances are classified by first generating their intermediate representation and then mapping it onto the novel category representation (which can be generated using meta-data alone). The focus of research has mainly been on what is a good intermediate level representation and how should the mapping be carried out.

A popular choice of the intermediate level representation have been parts and attributes—whether they be semantic or discriminative. Since not all features are relevant to all attributes, Farhadi et al. (2009) explore feature selection so as to better predict a novel instance's attributes. Probabilistic techniques for mapping the list of predicted attributes to a novel category's list of attributes (known *a priori*) are developed in Lampert et al. (2009) while Palatucci et al. (2009) carry out a theoretical analysis and use the one nearest neighbour rule. An alternative approach to zero-shot learning is not to name the novel object, or explicitly recognise its attributes, but simply say that it is "like" an object seen during training (Wang et al. 2010). For instance, the Babirusa in Fig. 1 could be declared to be like a pig. This is sufficient for some applications and works well if the training set has good category level coverage.

3 M3L: the max-margin multi-label classification primal formulation

The objective in multi-label classification is to learn a function f which can be used to assign a *set* of labels to a point $\mathbf{x}$. We assume that N training data points have been provided of the form $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^D \times \{\pm 1\}^L$ with y_{il} being $+1$ if label l has been assigned to point i and -1 otherwise. Note that such an encoding allows us to learn from both the presence and absence of labels, since both can be informative when predicting test categories.

A principled way of formulating the problem would be to take the loss function Δ that one truly cares about and minimise it over the training set subject to regularisation or prior knowledge. Of course, since direct minimisation of most discrete loss functions is hard, we might end up minimising an upper bound on the loss, such as the hinge. The learning problem can then be formulated as the following primal

$$P_1 = \min_f \frac{1}{2}\|f\|^2 + C\sum_{i=1}^{N} \xi_i \tag{1}$$

$$\text{s.t.} \quad f(\mathbf{x}_i, \mathbf{y}_i) \geq f(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \quad \forall i, \mathbf{y} \in \{\pm 1\}^L \setminus \{\mathbf{y}_i\} \tag{2}$$

$$\xi_i \geq 0 \quad \forall i \tag{3}$$

with a new point $\mathbf{x}$ being assigned labels according to $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$. The drawback of such a formulation is that there are $N2^L$ constraints which make direct optimisation very slow. Furthermore, classification of novel points might require 2^L function evaluations (one for each possible value of $\mathbf{y}$), which can be prohibitive at run time. In this Section, we demonstrate that, under general assumptions of linearity, (P_1) can be reformulated as the minimisation of L densely correlated sub-problems each having only N constraints. At the same time, prediction cost is reduced to a single function evaluation with complexity linear in the number of labels. The ideas underlying this decomposition were also used in Evgeniou et al. (2005) in a multi-task learning scenario. However, their objective is to combine multiple tasks into a single learning problem, while we are interested in decomposing (3) into multiple subproblems.

We start by making the standard assumption that

$$f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^t\big(\boldsymbol{\phi}(\mathbf{x}) \otimes \boldsymbol{\psi}(\mathbf{y})\big) \tag{4}$$

where $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are the feature and label space mappings respectively, $\otimes$ is the Kronecker product and $\mathbf{w}^t$ denotes $\mathbf{w}$ transpose. Note that, for zero shot learning, it is possible to theoretically show that , in the limit of infinite data, one does not need to model label correlations when training and test distributions are the same (Palatucci et al. 2009). In practice, however, training sets are finite, often relatively small, and have label distributions that are significantly different from the test set. Therefore to incorporate prior knowledge and correlate classifiers efficiently, we assume that labels have at most linear, possibly dense, correlation so that it is sufficient to choose $\boldsymbol{\psi}(\mathbf{y}) = \mathbf{P}\mathbf{y}$ where $\mathbf{P}$ is an invertible matrix encoding all our prior knowledge about the labels. If we assume f to be quadratic (or higher order) in $\mathbf{y}$, as is done in structured output prediction, then it would not be possible to reduce the number of constraints from exponential to linear while still modelling dense, possibly negative, label correlations. Furthermore, learning label correlation on the training set by incorporating quadratic terms in $\mathbf{y}$ might not be fruitful as the test categories will have very different correlation statistics. Thus, by sacrificing some expressive power, we hope to build much more efficient algorithms that can still give improved prediction accuracy in the zero-shot learning scenario.

We make another standard assumption that the chosen loss function should decompose over the individual labels (Taskar et al. 2003). Hence, we require that

$$\Delta(\mathbf{y}_i, \mathbf{y}) = \sum_{l=1}^{L} \Delta_l(\mathbf{y}_i, y_l) \tag{5}$$

where $y_l \in \{\pm 1\}$ corresponds to label l in the set of labels represented by $\mathbf{y}$. For instance, the popular Hamming loss, amongst others, satisfies this condition. We define the Hamming loss $\Delta_H(\mathbf{y}_i, \mathbf{y})$, between a ground truth label $\mathbf{y}_i$ and a prediction $\mathbf{y}$ as

$$\Delta_H(\mathbf{y}_i, \mathbf{y}) = \mathbf{y}_i^t(\mathbf{y}_i - \mathbf{y}) \tag{6}$$

which is a count of twice the total number of individual labels mispredicted in $\mathbf{y}$. Note that the Hamming loss can be decomposed over the labels as $\Delta_H(\mathbf{y}_i, \mathbf{y}) = \sum_l 1 - y_l y_{il}$. Of course, for Δ to represent a sensible loss we also require that $\Delta(\mathbf{y}_i, \mathbf{y}) \geq \Delta(\mathbf{y}_i, \mathbf{y}_i) = 0$.

Under these assumptions, (P_1) can be expressed as

$$P_1 \equiv \min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^t\mathbf{w} + C\sum_{i=1}^{N} \max_{\mathbf{y}\in\{\pm 1\}^L} \left[\Delta(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^t\boldsymbol{\phi}(\mathbf{x}_i) \otimes \mathbf{P}(\mathbf{y} - \mathbf{y}_i)\right] \tag{7}$$

where the constraints have been moved into the objective and $\xi_i \geq 0$ eliminated by including $\mathbf{y} = \mathbf{y}_i$ in the maximisation. To simplify notation, we express the vector $\mathbf{w}$ as a $D \times L$ matrix $\mathbf{W}$ so that

$$P_1 \equiv \min_{\mathbf{W}} \frac{1}{2}\operatorname{Trace}\left(\mathbf{W}^t\mathbf{W}\right) + C\sum_{i=1}^{N} \max_{\mathbf{y}\in\{\pm 1\}^L} \left[\Delta(\mathbf{y}_i, \mathbf{y}) + (\mathbf{y} - \mathbf{y}_i)^t\mathbf{P}^t\mathbf{W}^t\boldsymbol{\phi}(\mathbf{x}_i)\right] \tag{8}$$

Substituting $\mathbf{Z} = \mathbf{W}\mathbf{P}$, $\mathbf{R} = \mathbf{P}^t\mathbf{P} \succ 0$ and using the identity $\operatorname{Trace}(\mathbf{ABC}) = \operatorname{Trace}(\mathbf{CAB})$ results in

$$P_1 \equiv \min_{\mathbf{Z}} \frac{1}{2}\sum_{l=1}^{L}\sum_{k=1}^{L} R_{lk}^{-1}\mathbf{z}_l^t\mathbf{z}_k$$

$$+ C \sum_{i=1}^{N} \max_{\mathbf{y} \in \{\pm 1\}^L} \left[\sum_{l=1}^{L} \left[\Delta_l(\mathbf{y}_i, y_l) + (y_l - y_{il}) \mathbf{z}_l^t \boldsymbol{\phi}(\mathbf{x}_i) \right] \right] \tag{9}$$

where $\mathbf{z}_l$ is the lth column of $\mathbf{Z}$. Note that the terms inside the maximisation break up independently over the L components of $\mathbf{y}$. It is therefore possible to interchange the maximisation and summation to get

$$\begin{aligned} P_1 \equiv \min_{\mathbf{Z}} & \frac{1}{2} \sum_{l=1}^{L} \sum_{k=1}^{L} R_{lk}^{-1} \mathbf{z}_l^t \mathbf{z}_k \\ & + C \sum_{i=1}^{N} \sum_{l=1}^{L} \left[\max_{y_l \in \{\pm 1\}} \left[\Delta_l(\mathbf{y}_i, y_l) + (y_l - y_{il}) \mathbf{z}_l^t \boldsymbol{\phi}(\mathbf{x}_i) \right] \right] \end{aligned} \tag{10}$$

This leads to an equivalent primal formulation (P_2) as the summation of L correlated problems, each having N constraints which is significantly easier to optimise.

$$P_2 = \sum_{l=1}^{L} S_l \tag{11}$$

$$S_l = \min_{\mathbf{Z}, \boldsymbol{\xi}} \frac{1}{2} \mathbf{z}_l^t \sum_{k=1}^{L} R_{lk}^{-1} \mathbf{z}_k + C \sum_{i=1}^{N} \xi_{il} \tag{12}$$

$$\text{s.t.} \quad 2 y_{il} \mathbf{z}_l^t \boldsymbol{\phi}(\mathbf{x}_i) \geq \Delta_l(\mathbf{y}_i, -y_{il}) - \xi_{il} \tag{13}$$

$$\xi_{il} \geq \Delta_l(\mathbf{y}_i, y_{il}) \tag{14}$$

Furthermore, a novel point $\mathbf{x}$ can be assigned the set of labels for which the entries of $\text{sign}(\mathbf{Z}^t \boldsymbol{\phi}(\mathbf{x}))$ are $+1$. This corresponds to a single evaluation of f taking time linear in the number of labels.

The L classifiers in $\mathbf{Z}$ are not independent but correlated by $\mathbf{R}$—a positive definite matrix encoding our prior knowledge about label correlations. One might typically have thought of learning $\mathbf{R}$ from training data. For instance, one could learn $\mathbf{R}$ directly or express $\mathbf{R}^{-1}$ as a linear combination of predefined positive definite matrices with learnt coefficients. Such formulations have been developed in the Multiple Kernel Learning literature and we could leverage some of the proposed MKL optimization techniques (Vishwanathan et al. 2010). However, in the zero-shot learning scenario, learning $\mathbf{R}$ from training data is not helpful as the correlations between labels during training might be significantly different from those during testing.

Instead, we rely on the standard zero-shot learning assumption, that the test category attributes are known a priori (Farhadi et al. 2009, 2010; Lampert et al. 2009; Palatucci et al. 2009). Furthermore, if the prior distribution of test categories was known, then R could be set to approximate the average pairwise test label correlation (see Sect. 7.2.1 for details).

Note that, in the zero-shot learning scenario, $\mathbf{R}$ can be dense, as almost all the attributes might be shared across categories and correlated with each other, and can also have negative entries representing negative label correlations. We propose to improve prediction accuracy on the novel test categories by encoding prior knowledge about their label correlations in $\mathbf{R}$.

Note that we deliberately chose not to include bias terms $\mathbf{b}$ in f even though the reduction from (P_1) to (P_2) would still have gone through and the resulting kernelised optimisation been more or less the same (see Sect. 7.1). However, we would then have had to regularise $\mathbf{b}$

and correlate it using $\mathbf{R}$. Otherwise $\mathbf{b}$ would have been a free parameter capable of undoing the effects of $\mathbf{R}$ on $\mathbf{Z}$. Therefore, rather than explicitly have $\mathbf{b}$ and regularise it, we implicitly simulate $\mathbf{b}$ by adding an extra dimension to the feature vector. This has the same effect while keeping optimisation simple.

We briefly discuss two special cases before turning to the dual and its optimisation.

3.1 The special case of 1-vs-all

If label correlation information is not included, i.e. $\mathbf{R} = \mathbf{I}$, then (P_2) decouples into L completely independent sub-problems each of which can be tackled in isolation. In particular, for the Hamming loss we get

$$P_3 = \sum_{l=1}^{L} S_l \tag{15}$$

$$S_l = \min_{\mathbf{z}_l, \boldsymbol{\xi}} \frac{1}{2}\mathbf{z}_l^t \mathbf{z}_l + 2C \sum_{i=1}^{N} \xi_i \tag{16}$$

$$\text{s.t.} \quad y_{il}\mathbf{z}_l^t \boldsymbol{\phi}(\mathbf{x}_i) \geq 1 - \xi_i \tag{17}$$

$$\xi_i \geq 0 \tag{18}$$

Thus, S_l reduces to an independent binary classification sub-problem where the positive class contains all training points tagged with label l and the negative class contains all other points. This is exactly the strategy used in the popular and effective 1-vs-All method and we can therefore now make explicit the assumptions underlying this technique. The only difference is that one should charge a misclassification penalty of $2C$ to be consistent with the original primal formulation.

3.2 Relating the kernel to the loss

In general, the kernel is chosen so as to ensure that the training data points become well separated in the feature space. This is true for both K_x, the kernel on $\mathbf{x}$, as well as K_y, the kernel on $\mathbf{y}$. However, one might also take the view that since the loss Δ induces a measure of dissimilarity in the label space it must be related to the kernel on $\mathbf{y}$ which is a measure of similarity in the label space. This heavily constrains the choice of K_y and therefore the label space mapping $\boldsymbol{\psi}$. For instance, if a linear relationship is assumed, we might choose $\Delta(\mathbf{y}_i, \mathbf{y}) = K_y(\mathbf{y}_i, \mathbf{y}_i) - K_y(\mathbf{y}_i, \mathbf{y})$. Note that this allows Δ to be asymmetric even though K_y is not and ensures the linearity of Δ if ψ, the label space mapping, is linear.

In this case, label correlation information should be encoded directly into the loss. For example, the Hamming loss could be transformed to $\Delta_H(\mathbf{y}_i, \mathbf{y}) = \mathbf{y}_i^t \mathbf{R}(\mathbf{y}_i - \mathbf{y})$. $\mathbf{R}$ is the same matrix as before except the interpretation now is that the entries of $\mathbf{R}$ encode label correlations by specifying the penalties to be charged if a label is misclassified in the set. Of course, for Δ to be a valid loss, not only must $\mathbf{R}$ be positive definite as before but it now must also be diagonally dominant. As such, it can only encode "weak" correlations. Given the choice of Δ and the linear relationship with K_y , the label space mapping gets fixed to $\psi(\mathbf{y}) = \mathbf{P}\mathbf{y}$ where $\mathbf{R} = \mathbf{P}^t\mathbf{P}$.

Under these assumptions once can still go from (P_1) to (P_2) using the same steps as before. The main differences are that $\mathbf{R}$ is now more restricted and that $\Delta_l(\mathbf{y}_i, y_l) = (1/L)\mathbf{y}_i^t \mathbf{R}\mathbf{y}_i - y_l \mathbf{y}_i^t \mathbf{R}_l$ where $\mathbf{R}_l$ is the lth column of $\mathbf{R}$. While this result is theoretically interesting, we do not explore it further in this paper.

4 The M3L dual formulation

The dual of (P_2) has similar properties in that it can be viewed as the maximisation of L related problems which decouple into independent binary SVM classification problems when $\mathbf{R} = \mathbf{I}$. The dual is easily derived if we rewrite (P_2) in vector notation. Defining

$$\mathbf{Y}_l = \operatorname{diag}\big([y_{1l}, \ldots, y_{Nl}]\big) \tag{19}$$

$$\mathbf{K}_{\mathbf{x}} = \boldsymbol{\phi}^t(\mathbf{X})\boldsymbol{\phi}(\mathbf{X}) \tag{20}$$

$$\boldsymbol{\Delta}_l^{\pm} = \big[\Delta_l(\mathbf{y}_1, \pm y_{1l}), \ldots, \Delta_l(\mathbf{y}_N, \pm y_{Nl})\big]^t \tag{21}$$

we get the following Lagrangian

$$L = \sum_{l=1}^{L}\Bigg(\frac{1}{2}\sum_{k=1}^{L} R_{lk}^{-1}\mathbf{z}_l^t\mathbf{z}_k + C\mathbf{1}^t\boldsymbol{\xi}_l - \boldsymbol{\beta}_l^t\big(\boldsymbol{\xi}_l - \boldsymbol{\Delta}_l^+\big) - \boldsymbol{\alpha}_l^t\big(2\mathbf{Y}_l\boldsymbol{\phi}^t(\mathbf{X})\mathbf{z}_l - \boldsymbol{\Delta}_l^- + \boldsymbol{\xi}_l\big)\Bigg) \tag{22}$$

with the optimality conditions being

$$\nabla_{\mathbf{z}_l} L = 0 \Rightarrow \sum_{k=1}^{L} R_{lk}^{-1}\mathbf{z}_k = 2\boldsymbol{\phi}(\mathbf{X})\mathbf{Y}_l\boldsymbol{\alpha}_l \tag{23}$$

$$\nabla_{\boldsymbol{\xi}_l} L = 0 \Rightarrow C\mathbf{1} - \boldsymbol{\alpha}_l - \boldsymbol{\beta}_l = \mathbf{0} \tag{24}$$

Substituting these back into the Lagrangian leads to the following dual

$$D_2 = \max_{\mathbf{0}\le\boldsymbol{\alpha}\le C\mathbf{1}} \sum_{l=1}^{L}\boldsymbol{\alpha}_l^t\big(\boldsymbol{\Delta}_l^- - \boldsymbol{\Delta}_l^+\big) - 2\sum_{l=1}^{L}\sum_{k=1}^{L} R_{lk}\boldsymbol{\alpha}_l^t\mathbf{Y}_l\mathbf{K}_{\mathbf{x}}\mathbf{Y}_k\boldsymbol{\alpha}_k \tag{25}$$

Henceforth we will drop the subscript on the kernel matrix and write $\mathbf{K}_{\mathbf{x}}$ as $\mathbf{K}$.

5 Optimisation

The M3L dual is similar to the standard SVM dual. Existing optimisation techniques can therefore be brought to bear. However, the dense structure of $\mathbf{R}$ couples all NL dual variables and simply porting existing solutions leads to very inefficient code. We show that, with book keeping, we can easily go from an $O(L^2)$ algorithm to an $O(L)$ algorithm. Furthermore, by re-utilising the kernel cache, our algorithms can be very efficient even for non-linear problems. We treat the kernelised and linear M3L cases separately.

5.1 Kernelised M3L

The Dual (D_2) is a convex quadratic programme with very simple box constraints. We can therefore use co-ordinate ascent algorithms (Platt 1999; Fan et al. 2005; Lin et al. 2009) to maximise the dual. The algorithms start by picking a feasible point—typically $\boldsymbol{\alpha} = \mathbf{0}$. Next,two variables are selected and optimised analytically. This step is repeated until the

projected gradient magnitude falls below a threshold and the algorithm can be shown to have converged to the global optimum (Lin et al. 2009). The three key components are therefore: (a) reduced variable optimisation; (b) working set selection and (c) stopping criterion and kernel caching. We now discuss each of these components. The pseudo-code of the algorithm and proof of convergence are given in the Appendix.

5.1.1 Reduced Variable Optimisation

If all but two of the dual variables were fixed, say α_{pl} and α_{ql} along the label l, then the dual optimisation problem reduces to

$$D_{2pql} = \max_{\delta_{pl},\delta_{ql}} -2\left(\delta_{pl}^2 K_{pp} R_{ll} + \delta_{ql}^2 K_{qq} R_{ll} + 2\delta_{pl}\delta_{ql} y_{pl} y_{ql} K_{pq} R_{ll}\right) + \delta_{pl} g_{pl} + \delta_{ql} g_{ql} \tag{26}$$

$$\text{s.t.} \quad -\alpha_{pl} \leq \delta_{pl} \leq C - \alpha_{pl} \tag{27}$$

$$-\alpha_{ql} \leq \delta_{ql} \leq C - \alpha_{ql} \tag{28}$$

where $\delta_{pl} = \alpha_{pl}^{\text{new}} - \alpha_{pl}^{\text{old}}$ and $\delta_{ql} = \alpha_{ql}^{\text{new}} - \alpha_{ql}^{\text{old}}$ and

$$g_{pl} = \nabla_{\alpha_{pl}} D_2 = \Delta_{pl}^- - \Delta_{pl}^+ - 4\sum_{i=1}^{N}\sum_{k=1}^{L} R_{kl} K_{ip} y_{ik} y_{pl} \alpha_{ik} \tag{29}$$

Note that D_{2pql} has a quadratic objective in two variables which can be maximised analytically due to the simple box constraints. We do not give the expressions for α_{pl}^{new} and α_{ql}^{new} which maximise D_{2pql} as many special cases are involved for when the variables are at bound but they can be found in Algorithm 2 of the pseudo-code in Appendix A.

5.1.2 Working set selection

Since the M3L formulation does not have a bias term, it can be optimized by picking a single variable at each iteration rather than a pair of variables. This leads to a low cost per iteration but a large number of iterations. Selecting two variables per iteration increases the cost per iteration but significantly reduces the number of iterations as second order information can be incorporated into the variable selection policy.

If we were to choose two variables to optimise along the same label l, say α_{pl} and α_{ql}, then the maximum change that we could affect in the dual is given by

$$\delta D_2(\alpha_{pl}, \alpha_{ql}) = \frac{g_{pl}^2 K_{qq} + g_{ql}^2 K_{pp} - 2 g_{pl} g_{ql} y_{pl} y_{ql} K_{pq}}{8 R_{ll}(K_{pp} K_{qq} - K_{pq}^2)} \tag{30}$$

In terms of working set selection, it would have been ideal to have chosen the two variables α_{pl} and α_{ql} which would have maximised the increase in the dual objective. However, this turns out to be too expensive in practice. A good approximation is to choose the first point α_{pl} to be the one having the maximum projected gradient magnitude. The projected gradient is defined as

$$\tilde{g}_{pl} = \begin{cases} g_{pl} & \text{if } \alpha_{pl} \in (0, C) \\ \min(0, g_{pl}) & \text{if } \alpha_{pl} = C \\ \max(0, g_{pl}) & \text{if } \alpha_{pl} = 0 \end{cases} \tag{31}$$

and hence the first point is chosen as

$$\left(p^*, l^*\right) = \operatorname*{argmax}_{p,l} |\tilde{g}_{pl}| \tag{32}$$

Having chosen the first point, the second point is chosen to be the one that maximises (30).

Working set selection can be made efficient by maintaining the set of gradients g. Every time a variable, say α_{pl} is changed, the gradients need to be updated as

$$g_{jk}^{\text{new}} = g_{jk}^{\text{old}} - 4y_{pl}y_{jk}R_{kl}K_{pj}\left(\alpha_{pl}^{\text{new}} - \alpha_{pl}^{\text{old}}\right) \tag{33}$$

Note that because of the dense structure of $\mathbf{R}$ all NL gradients have to be updated even if a single variable is changed. Since there are NL variables and each has to be updated presumably at least once we end up with an algorithm that takes time at least N^2L^2.

The algorithm can be made much more efficient if, with some book keeping, not all gradients had to be updated every time a variable was changed. For instance, if we were to fix a label l and modify L variables along the chosen label, then the gradient update equations could be written as

$$g_{jk}^{\text{new}} = g_{jk}^{\text{old}} - 4y_{jk}\sum_{i=1}^{N} y_{il}R_{kl}K_{ij}\left(\alpha_{il}^{\text{new}} - \alpha_{il}^{\text{old}}\right) \tag{34}$$

$$= g_{jk}^{\text{old}} - 4R_{kl}y_{jk}u_{jl} \tag{35}$$

$$\text{where} \quad u_{jl} = \sum_{i=1}^{N} K_{ij}y_{il}\left(\alpha_{il}^{\text{new}} - \alpha_{il}^{\text{old}}\right) \tag{36}$$

As long as we are changing variables along a particular label, the gradient updates can be accumulated in u and only when we switch to a new variable do all the gradients have to be updated. We therefore end up doing $O(NL)$ work after changing L variables resulting in an algorithm which takes time $O(N^2L)$ rather than $O(N^2L^2)$.

5.1.3 Stopping criterion and kernel caching

We use the standard stopping criterion that the projected gradient magnitude for all NL dual variables should be less than a predetermined threshold.

We employ a standard Least Recently Used (LRU) kernel cache strategy implemented as a circular queue. Since we are optimising over all labels jointly, the kernel cache gets effectively re-utilised, particularly as compared to independent methods that optimise one label at a time. In the extreme case, independent methods will have to rebuild the cache for each label which can slow them down significantly.

6 Linear M3L

We build on top of the stochastic dual coordinate ascent with shrinkage algorithm of Hsieh et al. (2008). At each iteration, a single dual variable is chosen uniformly at random from the active set, and optimised analytically. The variable update equation is given by

$$\alpha_{pl}^{\text{new}} = \max\left(0, \min\left(C, \alpha_{pl}^{\text{old}} + \delta\alpha_{pl}\right)\right) \tag{37}$$

$$\text{where} \quad \delta\alpha_{pl} = \frac{\Delta_{pl}^{-} - \Delta_{pl}^{+} - 4\sum_q \sum_k K_{pq} R_{lk} y_{pl} y_{qk} \alpha_{qk}}{4K_{pp} R_{ll}} \tag{38}$$

$$= \frac{\Delta_{pl}^{-} - \Delta_{pl}^{+} - 2y_{pl} \mathbf{z}_l^t \mathbf{x}_p}{4R_{ll} \mathbf{x}_p^t \mathbf{x}_p} \tag{39}$$

As can be seen, the dual variable update can be computed more efficiently in terms of the primal variables $\mathbf{Z}$ which then need to be maintained every time a dual variable is modified. The update equation for $\mathbf{Z}$ every time α_{pl} is modified is

$$\mathbf{z}_l^{\text{new}} = \mathbf{z}_l^{\text{old}} + 2R_{kl} y_{pl} \left(\alpha_{pl}^{\text{new}} - \alpha_{pl}^{\text{old}}\right) \mathbf{x}_p \tag{40}$$

Thus, all the primal variables $\mathbf{Z}$ need to be updated every time a single dual variable is modified. Again, as in the kernelised case, the algorithm can be made much more efficient by fixing a label l and modifying L dual variables along it while delaying the gradient updates as

$$\mathbf{z}_k^{\text{new}} = \mathbf{z}_k^{\text{old}} + 2R_{kl} \sum_{j=1}^{N} y_{jl} \left(\alpha_{jl}^{\text{new}} - \alpha_{jl}^{\text{old}}\right) \mathbf{x}_j \tag{41}$$

$$= \mathbf{z}_k^{\text{old}} + 2R_{kl} \mathbf{v}_l \tag{42}$$

$$\text{where} \quad \mathbf{v}_l = \sum_{j=1}^{N} y_{jl} \left(\alpha_{jl}^{\text{new}} - \alpha_{jl}^{\text{old}}\right) \mathbf{x}_j \tag{43}$$

In practice, it was observed that performing L stochastic updates along a chosen label right from the start could slow down convergence in some cases. Therefore, we initially use the more expensive strategy of choosing dual variables uniformly at random and only after the projected gradient magnitudes are below a pre-specified threshold do we switch to the strategy of optimising L dual variables along a particular label before picking a new label uniformly at random.

The active set is initialised to contain all the training data points. Points at bound having gradient magnitude outside the range of currently maintained extremal gradients are discarded from the active set. Extremal gradients are re-estimated at the end of each pass and if they are too close to each other the active set is expanded to include all training points once again.

A straightforward implementation with globally maintained extremal gradients leads to inefficient code. Essentially, if the classifier for a particular label has not yet converged, then it can force a large active set even though most points would not be considered by the other classifiers. We therefore implemented separate active sets for each label but coupled the maintained extremal gradients via $\mathbf{R}$. The extremal gradients lb_l and ub_l, for label l. are initially set to $-\infty$ and $+\infty$ respectively. After each pass through the active set, they are updated as

$$lb_l = \min_k \left(|R_{kl}| \min_{i \in A_k} \tilde{g}_{ik} \right) \tag{44}$$

$$ub_l = \max_k \left(|R_{kl}| \max_{i \in A_k} \tilde{g}_{ik} \right) \tag{45}$$

where A_k is the set of indices in the active set of label k. This choice was empirically found to decrease training time.

Once all the projected gradients in all the active sets have magnitude less than a threshold τ, we expand the active sets to include all the variables, and re-estimate the projected gradients. The algorithm stops when all projected gradients have magnitude less than τ.

7 Experiments

In this section we first compare the performance of our optimisation algorithms and then evaluate how prediction accuracy can be improved by incorporating prior knowledge about label correlations.

7.1 Optimisation experiments

The cutting plane algorithm in SVMStruct (Tsochantaridis et al. 2005) is a general purpose algorithm that can be used to optimise the original M3L formulation (P_1). In each iteration, the approximately worst violating constraint is added to the active set and the algorithm is proved to take a maximum number of iterations independent of the size of the output space. The algorithm has a user defined parameter ϵ for the amount of error that can be tolerated in finding the worst violating constraint.

We compared the SVMStruct algorithm to our M3L implementation on an Intel Xeon 2.67 GHz machine with 8 GB RAM. It was observed that even on medium scale problems with linear kernels, our M3L implementation was nearly a hundred times faster than SVMStruct. For example, on the Media Mill data set (Snoek et al. 2006) with a hundred and one labels and ten, fifteen and twenty thousand training points, our M3L code took 19, 37 and 55 seconds while SVMStruct took 1995, 2998 and 7198 seconds respectively. On other data sets SVMStruct ran out of RAM or failed to converge in a reasonable amount of time (even after tuning ϵ). This demonstrates that explicitly reducing the number of constraints from exponential to linear and implementing a specialised solver can lead to a dramatic reduction in training time.

As the next best thing, we benchmark our performance against the 1-vs-All method, even though it can't incorporate prior label correlations. In the linear case, we compare our linear M3L implementation to 1-vs-All trained by running LibLinear (Fan et al. 2008) and LibSVM (Chang and Lin 2001) independently over each label. For the non-linear case, we compare our kernelised M3L implementation to 1-vs-All trained using LibSVM. In each case, we set $\mathbf{R} = \mathbf{I}$, so that M3L reaches exactly the same solution as LibSVM and LibLinear. Also, we avoided repeated disk I/O by reading the data into RAM and using LibLinear and LibSVM's API's.

Table 1 lists the variation in training time with the number of training examples on the Animals with Attributes (Lampert et al. 2009), Media Mill (Snoek et al. 2006), Siam (SIAM 2007) and RCV1 (Lewis et al. 2004) data sets. The training times of linear M3L (LM3L) and LibLinear are comparable, with LibLinear being slightly faster. The training time of kernelised M3L (KM3L) are significantly lower than LibSVM, with KM3L sometimes being as much as 30 times faster. This is primarily because KM3L can efficiently leverage the kernel cache across all labels while LibSVM has to build the cache from scratch each time. Furthermore, leaving aside caching issues, it would appear that by optimising over all variables jointly, M3L reaches the vicinity of the global optimum much more quickly than 1-vs-All. Figure 2 plots dual progress against the number of iterations for all four data sets with ten thousand training points. As can be seen, kernelised M3L gets to within the vicinity of the global optimum much faster than 1-vs-All implemented using LibSVM. Figure 3

Table 1 Comparison of training times for the linear M3L (LM3L) and kernelised M3L (KM3L) optimisation algorithms with 1-vs-All techniques implemented using LibLinear and LibSVM. Each data set has N training points, D features and L labels. See text for details

(a) Animals with Attributes: $D = 252$, $L = 85$

N	Linear Kernel (s)				RBF Kernel (s)	
	1-vs-All LibLinear	LM3L	1-vs-All LibSVM	KM3L	1-vs-All LibSVM	KM3L
2,000	3	7	234	15	250	20
10,000	48	51	5438	245	6208	501
15,000	68	74	11990	500	13875	922
24,292	102	104	29328	1087	34770	3016

(b) RCV1: $D = 47{,}236$(sparse), $L = 103$

N	Linear Kernel (s)				RBF Kernel (s)	
	1-vs-All LibLinear	LM3L	1-vs-All LibSVM	KM3L	1-vs-All LibSVM	KM3L
2,000	7	4	54	6	139	11
10,000	23	27	743	110	1589	177
15,000	33	43	1407	230	2893	369
23,149	45	57	2839	513	5600	817

(c) Siam: $D = 30{,}438$(sparse), $L = 22$

N	Linear Kernel (s)				RBF Kernel (s)	
	1-vs-All LibLinear	LM3L	1-vs-All LibSVM	KM3L	1-vs-All LibSVM	KM3L
2,000	1	1	27	5	43	7
10,000	2	2	527	126	775	185
15,000	3	3	1118	288	1610	422
21,519	5	5	2191	598	3095	878

(d) Media Mill: $D = 120$, $L = 101$

N	Linear Kernel (s)				RBF Kernel (s)	
	1-vs-All LibLinear	LM3L	1-vs-All LibSVM	KM3L	1-vs-All LibSVM	KM3L
2,000	2	2	11	2	15	6
10,000	18	19	456	57	505	123
15,000	35	37	1014	124	1107	275
25,000	62	75	2662	337	2902	761
30,993	84	97	4168	527	4484	1162

shows similar plots with respect to time. The difference is even more significant due to kernel caching effects. In conclusion, even though M3L generalises 1-vs-All, its training time can be comparable, and sometimes, even significantly lower.

Finally, to demonstrate that our code scales to large problems, we train linear M3L on RCV1 with 781,265 points, 47,236 dimensional sparse features and 103 labels. Table 2 charts dual progress and train and test error with time. As can be seen, the model is nearly fully trained in under six minutes and converges in eighteen.

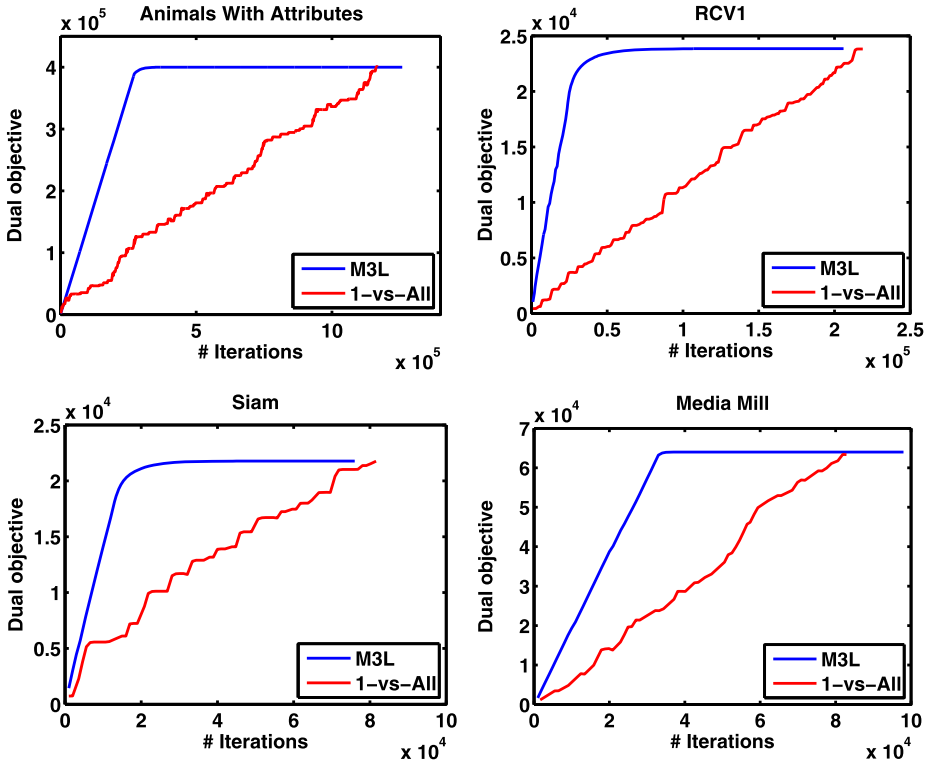

Fig. 2 Dual progress versus number of iterations for the kernelised M3L algorithm and 1-vs-All implemented using LibSVM for an RBF kernel and ten thousand training points. M3L appears to get close to the vicinity of the global optimum much more quickly than 1-vs-All. The results are independent of kernel caching effects

Table 2 Linear M3L training on RCV1 with 781,265 points, 47,236 dimensional sparse features and 103 labels

Time (s)	Dual	Train Error (%)	Test Error (%)
60	1197842	0.86	0.98
183	1473565	0.74	0.84
300	1492664	0.72	0.83
338	1494012	0.72	0.82
345	1494050	0.72	0.82
353	1494057	0.72	0.82
1080	1494057	0.72	0.82

7.2 Incorporating prior knowledge for zero-shot learning

In this section, we investigate whether the proposed M3L formulation can improve label prediction accuracy in a zero-shot learning scenario. Zero-shot learning has two major components as mentioned earlier. The first component deals with generating an intermediate level representation, generally based on attributes for each data point. The second concerns itself with how to map test points in the intermediate representation to points representing

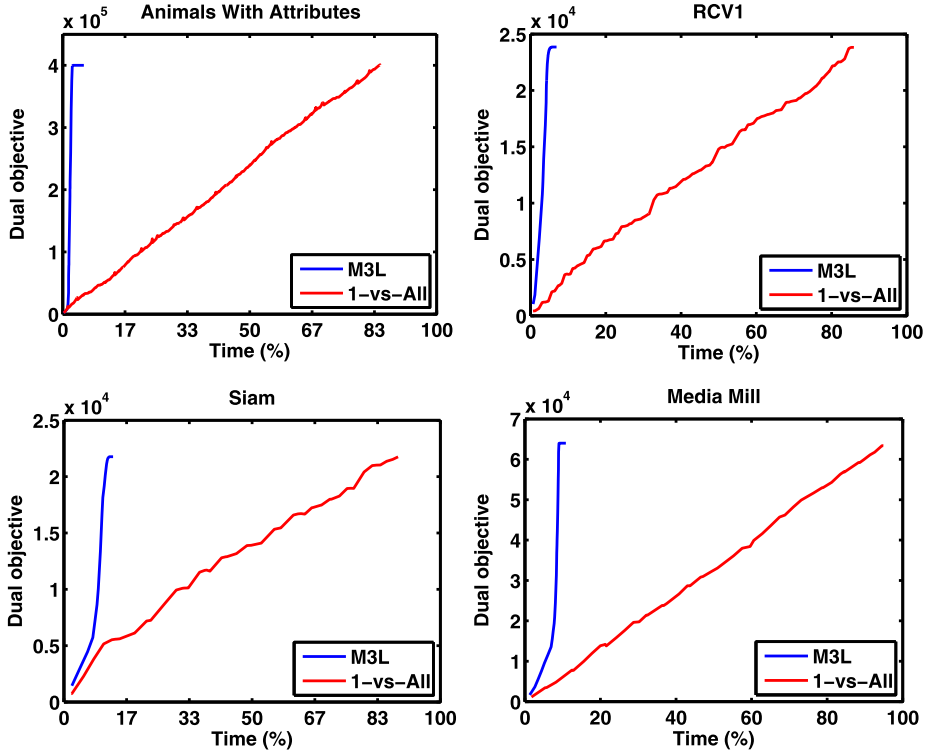

Fig. 3 Dual progress versus normalised time for the kernelised M3L algorithm and 1-vs-All implemented using LibSVM for an RBF kernel and ten thousand training points. The difference between M3L and 1-vs-All is even starker than in Fig. 2 due to kernel caching effects

novel categories. Our focus is on the former and the more accurate prediction of multiple, intermediate attributes (labels) when their correlation statistics on the training and test sets are significantly different.

7.2.1 Animals with attributes

The Animals with Attributes data set (Lampert et al. 2009) has forty training animal categories, such as Dalmatian, Skunk, Tiger, Giraffe, Dolphin, *etc*. and the following ten disjoint test animal categories: Humpback Whale, Leopard, Chimpanzee, Hippopotamus, Raccoon, Persian Cat, Rat, Seal, Pig and Giant Panda. All categories share a common set of 85 attributes such as has yellow, has spots, is hairless, is big, has flippers, has buckteeth, *etc*. The attributes are densely correlated and form a fully connected graph. Each image in the database contains a dominant animal and is labelled with its 85 attributes. There are 24,292 training images and 6,180 test images. Some example images are shown in Fig. 4. We use 252 dimensional PHOG features that are provided by the authors. M3L training times for this data set are reported in Table 1(a).

We start by visualising the influence of $\mathbf{R}$. We randomly sample 200 points from the training set and discard all but two of the attributes—"has black" and "is weak". These two attributes were selected as they are very weakly correlated on our training set, with a correlation coefficient of 0.2, but have a strong negative correlation of -0.76 on the test

Fig. 4 Sample training (*top*) and test (*bottom*) images from the Animals with Attributes data set

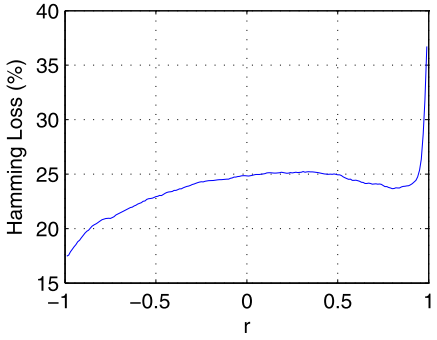

Fig. 5 Test Hamming loss versus classifier correlation

animals (Leopards, Giant Pandas, Humpback Whales and Chimpanzees all have black but are not weak). Figure 5 plots the Hamming loss on the test set as we set $\mathbf{R} = [1\ r; r\ 1]$, plug it into the M3L formulation, and vary r from -1 to $+1$. Learning independent classifiers for the two attributes ($r = 0$) can lead to a Hamming loss of 25 % because of the mismatch between training and test sets. This can be made even worse by incorrectly choosing, or learning using structured output prediction techniques, a prior that forces the two labels to be positively correlated. However, if our priors are generally correct, then negatively correlating the classifiers lowers prediction error.

We now evaluate performance quantitatively on the same training set but with all 85 labels. We stress that in the zero shot learning scenario no training samples from any of the test categories are provided. As is commonly assumed (Farhadi et al. 2009, 2010; Lampert et al. 2009; Palatucci et al. 2009), we only have access to $\mathbf{y}_c$ which is the set of attributes for a given test category. Furthermore we require, as additional information, the prior distribution over test categories $p(c)$. For the M3L formulation we set $\mathbf{R} = \sum_{c=1}^{10} p(c)\mathbf{y}_c\mathbf{y}_c^t$. Under this setup, learning independent classifiers using 1-vs-All yields a Hamming loss of 29.38 %. The Hamming loss for M3L, with the specific choice of $\mathbf{R}$, is 26.35 %. This decrease in error is very significant given that 1-vs-All, trained on all 24,292 training points, only manages to reduce error to 28.64 %. Thus M3L, with extra knowledge, in the form of just test category distributions, can dramatically reduce test error. The results also compare favourably to other independent methods such as BoostTexter (Schapire and Singer 2000) (30.28 %), power set multi-class classification (32.70 %), 5 nearest neighbours (31.79 %), regression (Hsu et al. 2009) (29.38 %) and ranking (Crammer and Singer 2003) (34.84 %).

Table 3 Test Hamming loss (%) on benchmark data sets

Method	fMRI-Words	SIAM	Media Mill	RCV1	Yeast	a-Yahoo
M3L	**47.29**	**8.41**	**3.78**	**3.45**	**24.99**	**9.897**
1-vs-All	53.97	11.15	4.69	4.25	26.93	11.35
BoostTexter	49.89	12.91	4.91	4.12	31.82	13.17
Power Set	48.69	14.01	6.27	3.71	32.32	17.81
Regression	53.76	11.19	4.69	4.26	26.70	11.36
Ranking	52.38	9.41	9.06	5.67	28.02	10.27
5-NN	50.81	12.51	4.74	4.47	28.82	13.04

7.2.2 Benchmark data sets

We also present results on the fMRI-Words zero-shot learning data set of Mitchell et al. (2008). The data set has 60 categories out of which we use 48 for training and 12 for testing. Each category is described by 25 real valued attributes which we convert to binary labels by thresholding against the median attribute value. Prior information about which attributes occur in which novel test categories is provided in terms of a knowledge base. The experimental protocol is kept identical to the one used in Animals with Attributes. $\mathbf{R}$ is set to $\sum_{c=1}^{10} p(c)\mathbf{y}_c\mathbf{y}_c^t$ where $\mathbf{y}_c$ comes from the knowledge base and $p(c)$ is required as additional prior information. We use 400 points for training and 648 points for testing. The test Hamming loss for M3L and various independent methods is given in Table 3. The M3L results are much better than 1-vs-All with the test Hamming loss being reduced by nearly 7 %. This is noteworthy since even if 1-vs-All were trained on the full training set of 2592 points, it would decrease the Hamming loss by just over 5 % to 48.79 %.

Table 3 also presents results on some other data sets. Unfortunately, most of them have not been designed for zero-shot learning. Siam (SIAM 2007), Media Mill (Snoek et al. 2006), RCV1 (Lewis et al. 2004) and Yeast (Elisseeff and Weston 2001) are traditional multi-label data sets with matching training and test set statistics. The a-PASCAL+a-Yahoo (Farhadi et al. 2009) data set has different training and test categories but does not include prior information about which attributes are relevant to which test categories. Thus, for all these data sets, we sample the original training set to create a new training subset which has different label correlations than the provided test set. The remainder of the original training points are used only to estimate the $\mathbf{R}$ matrix. As Table 3 indicates, by incorporating prior knowledge M3L can do better than all the other methods which assume independence.

8 Conclusions

We developed the M3L formulation for learning a max-margin multi-label classifier with prior knowledge about densely correlated labels. We showed that the number of constraints could be reduced from exponential to linear and, in the process, generalised 1-vs-All multi-label classification. We also developed efficient optimisation algorithms that were orders of magnitude faster than the standard cutting plane method. Our kernelised algorithm was significantly faster than even the 1-vs-All technique implemented using LibSVM and hence our code, available from Hariharan et al. (2010a), can also be used for efficient independent learning. Finally, we demonstrated on multiple data sets that incorporating prior knowledge

using M3L could improve prediction accuracy over independent methods. In particular, in zero-shot learning scenarios, M3L trained on 200 points could outperform 1-vs-All trained on nearly 25,000 points on the Animals with Attributes data set and the M3L test Hamming loss on the fMRI-Words data set was nearly 7 % lower than that of 1-vs-All.

Acknowledgements We would like to thank Alekh Agarwal, Brendan Frey, Sunita Sarawagi, Alex Smola and Lihi Zelnik-Manor for helpful discussions and feedback.

Appendix A: Pseudo code of the kernelised M3L algorithm

The dual that we are trying to solve is:

$$\max_{\boldsymbol{\alpha}} \quad \sum_{l=1}^{L} \boldsymbol{\alpha}_l^t \left(\boldsymbol{\Delta}_l^- - \boldsymbol{\Delta}_l^+\right) - 2 \sum_{l=1}^{L} \sum_{k=1}^{L} R_{lk} \boldsymbol{\alpha}_l^t \mathbf{Y}_l \mathbf{K} \mathbf{Y}_k \boldsymbol{\alpha}_k \tag{46}$$

s.t.

$$0 \leq \boldsymbol{\alpha} \leq C\mathbf{1}$$

where $\boldsymbol{\alpha}_l = [\alpha_{1l}, \ldots \alpha_{Nl}]$, $\mathbf{Y}_l = \text{diag}([y_{1l} \ldots y_{Nl}])$ and $\mathbf{K} = \phi(\mathbf{X})^t \phi(\mathbf{X})$. Algorithm 1 describes the training algorithm. The algorithm relies on picking two variables at each step and optimising over them keeping all the others constant. If the two variables are α_{pl} and α_{ql} (note that we choose two variables corresponding to the same label l), then at each step we maximise $h(\delta_{pl}, \delta_{ql}) = D_2(\boldsymbol{\alpha} + [\delta_{pl}, \delta_{ql}, \mathbf{0}^t]^t) - D_2(\boldsymbol{\alpha})$ subject to $-\alpha_{pl} \leq \delta_{pl} \leq C - \alpha_{pl}$ and $-\alpha_{ql} \leq \delta_{ql} \leq C - \alpha_{ql}$. Here, the indices have been reordered so that α_{pl}, α_{ql} occupy the first two indices. D_2 is the dual objective function. It can be seen that $h(\delta_{pl}, \delta_{ql})$ comes out to be:

$$\begin{aligned} h(\delta_{pl}, \delta_{ql}) = &-2\left(\delta_{pl}^2 K_{pp} R_{ll} + \delta_{ql}^2 K_{qq} R_{ll} + 2\delta_{pl}\delta_{ql} y_{pl} y_{ql} K_{pq} R_{ll}\right) \\ &+ \delta_{pl} g_{pl} + \delta_{ql} g_{ql} \end{aligned} \tag{47}$$

Here $g_{pl} = \nabla_{pl} D_2$ and similarly $g_{ql} = \nabla_{ql} D_2$. Since h is basically a quadratic function, it can be written as:

$$h(\boldsymbol{\delta}_{pq}) = -\frac{1}{2} \boldsymbol{\delta}_{pq} \mathbf{Q}_{pq} \boldsymbol{\delta}_{pq} + \mathbf{g}_{pq}^t \boldsymbol{\delta}_{pq} \tag{48}$$

where

$$\boldsymbol{\delta}_{pq} = \begin{bmatrix} \delta_{pl} \\ \delta_{ql} \end{bmatrix} \tag{49}$$

$$\mathbf{Q}_{pq} = \begin{bmatrix} 4K_{pp}R_{ll} & 4K_{pq}R_{ll}y_{pl}y_{ql} \\ 4K_{pq}R_{ll}y_{pl}y_{ql} & 4K_{qq}R_{ll} \end{bmatrix} \tag{50}$$

$$\mathbf{g}_{pq} = \begin{bmatrix} g_{pl} \\ g_{ql} \end{bmatrix} \tag{51}$$

The constraints too can be written in vector form as:

$$\mathbf{m}_{pq} \leq \boldsymbol{\delta}_{pq} \leq \mathbf{M}_{pq} \tag{52}$$

Algorithm 1 Kernelised M3L

$\theta_{ik} \leftarrow 0 \qquad \forall i,k$
$g_{ik} \leftarrow \Delta_{ik}^{-} - \Delta_{ik}^{+}$
repeat
  for $i = 1$ to N **do**
    $u_i \leftarrow 0$
  end for
  $l \leftarrow \arg\max_k(\max_i \mid \tilde{g}_{ik} \mid)$
  for *iteration* $= 1$ to L **do**
    $p \leftarrow \arg\max_i \mid \tilde{g}_{il} \mid$
    $S_p \leftarrow \{j : K_{pj} < \sqrt{K_{pp}K_{jj}}$ and $\tilde{g}_{jk} \neq 0\}$
    if $S_p \neq \phi$ **then**
      $q \leftarrow \arg\max_{j \in S_p} h_{pj}^{max}$
      $(\delta_{pl}, \delta_{ql}) \leftarrow$Solve2DQP$(\mathbf{Q}_{pq}, \mathbf{g}_{pq}, \mathbf{m}_{pq}, \mathbf{M}_{pq})$
      $\alpha_{pl} \leftarrow \alpha_{pl} + \delta_{pl}$
      $\alpha_{ql} \leftarrow \alpha_{ql} + \delta_{ql}$
      for $i = 1$ to N **do**
        $g_{il} \leftarrow g_{il} - 4R_{ll}y_{il}(K_{ip}y_{pl}\delta_{pl} + K_{iq}y_{ql}\delta_{ql})$
        $u_i \leftarrow u_i + (K_{ip}y_{pl}\delta_{pl} + K_{iq}y_{ql}\delta_{ql})$
      end for
    else
      $\delta_{pl} \leftarrow$Solve1DQP$(4K_{pp}R_{ll}, g_{pl}, -\alpha_{pl}, C - \alpha_{pl})$
      $\alpha_{pl} \leftarrow \alpha_{pl} + \delta_{pl}$
      for $i = 1$ to N **do**
        $g_{il} \leftarrow g_{il} - 4R_{ll}y_{il}K_{ip}y_{pl}\delta_{pl}$
        $u_i \leftarrow u_i + K_{ip}y_{pl}\delta_{pl}$
      end for
    end if.
    for $k \in \{1, \ldots, L\} \backslash \{l\}$ **do**
      for $i = 1$ to N **do**
        $g_{ik} \leftarrow g_{ik} - 4R_{kl}y_{ik}u_i$
      end for
    end for
  end for
until $\mid \tilde{g}_{ik} \mid < \tau \qquad \forall i,k$

where

$$\mathbf{m}_{pq} = \begin{bmatrix} -\alpha_{pl} \\ -\alpha_{ql} \end{bmatrix} \tag{53}$$

$$\mathbf{M}_{pq} = \begin{bmatrix} C - \alpha_{pl} \\ C - \alpha_{ql} \end{bmatrix} \tag{54}$$

Therefore at each step we solve a 2-variable quadratic program with box constraints. The algorithm to do so is described later.

The variables α_{pl} and α_{ql} being optimised over need to be chosen carefully. In particular we need to ensure that the matrix $\mathbf{Q}_{pq}$ is positive definite so that it can be maximised easily. We also need to make sure that none of α_{pl} and α_{ql} has projected gradient 0. The projected

gradient of α_{pl}, denoted here as $\tilde{g}_{pl}$, is given by:

$$\tilde{g}_{pl} = \begin{cases} g_{pl} & \text{if } \alpha_{pl} \in (0, C) \\ \min(0, g_{pl}) & \text{if } \alpha_{pl} = C \\ \max(0, g_{pl}) & \text{if } \alpha_{pl} = 0 \end{cases} \tag{55}$$

We also use some heuristics when choosing p and q. It can be seen that the unconstrained maximum of h is given by:

$$h_{pq}^{max} = \frac{g_{pl}^2 K_{qq} + g_{ql}^2 K_{pp} - 2 g_{pl} g_{ql} y_{pl} y_{ql} K_{pq}}{8 R_{ll} (K_{pp} K_{qq} - K_{pq}^2)} \tag{56}$$

This is an upper bound on the dual progress that we can achieve in an iteration and we pick p and q such that h_{pq}^{max} is as big as possible.

Algorithm 2 Solve2DQP($\mathbf{Q}, \mathbf{g}, \mathbf{m}, \mathbf{M}$)

$\mathbf{x}^* = \mathbf{Q}^{-1}\mathbf{g} = \frac{1}{Q_{11}Q_{22} - Q_{12}^2} \begin{bmatrix} Q_{22}g_1 - Q_{12}g_2 \\ Q_{11}g_2 - Q_{12}g_1 \end{bmatrix}$
$\mathbf{x}^0 = \min(\mathbf{M}, \max(\mathbf{m}, \mathbf{x}^*))$;
if $\mathbf{x}^* \in [\mathbf{m}, \mathbf{M}]$ **then**
  return $\mathbf{x}^*$
else if $x_1^* \in [l_1, u_1]$ **then**
  $x_1 \leftarrow Solve1DQP(Q_{11}, g_1 - Q_{12}x_2^0, m_1, M_1)$
  return (x_1, x_2^0)
else if $x_2^* \in [l_2, u_2]$ **then**
  $x_2 \leftarrow Solve1DQP(Q_{22}, g_2 - Q_{12}x_1^0, m_2, M_2)$
  return (x_1^0, x_2)
else
  $\mathbf{x}^1 \leftarrow [x_1^0, Solve1DQP(Q_{22}, g_2 - Q_{12}x_1^0, m_2, M_2)]$
  $\mathbf{x}^2 \leftarrow [Solve1DQP(Q_{11}, g_1 - Q_{12}x_2^0, m_1, M_1), x_2^0]$
  $d_1 \leftarrow -\frac{1}{2}\mathbf{x}^{1t}\mathbf{Q}\mathbf{x}^1 + \mathbf{g}^t\mathbf{x}^1$
  $d_2 \leftarrow -\frac{1}{2}\mathbf{x}^{2t}\mathbf{Q}\mathbf{x}^2 + \mathbf{g}^t\mathbf{x}^2$
  if $d_1 > d_2$ **then**
    return $\mathbf{x}^1$
  else
    return $\mathbf{x}^2$
  end if
end if

Algorithm 2 solves the problem:

$$\max_{\mathbf{m} \le \mathbf{x} \le \mathbf{M}} \quad -\frac{1}{2}\mathbf{x}^t\mathbf{Q}\mathbf{x} + \mathbf{g}^t\mathbf{x} \tag{57}$$

where $\mathbf{x}$ is 2-dimensional. Setting the gradient = 0, we get that the unconstrained maximum is at $\mathbf{Q}^{-1}\mathbf{g}$. If this point satisfies the box constraints, then we are done. If not, then we need to look at the boundaries of the feasible set. This can be done by clamping one variable to the boundary and maximising along the other, which becomes a 1-dimensional quadratic

problem. Solve1DQP(a, b, m, M), referenced in lines 21 of Algorithm 1 and lines 6, 9, 12 and 13 solves a 1-dimensional QP with box constraints:

$$\max_{m\le x\le M} -\frac{1}{2}ax^2 + bx \tag{58}$$

The solution to this is merely $\min(M, \max(m, \frac{b}{a}))$.

Appendix B: Proof of convergence of the kernelised M3L algorithm

We now give a proof of convergence of the kernelised M3L algorithm. The proof closely follows the one in Keerthi and Gilbert (2002) and is provided for the sake of completeness.

B.1 Notation

We denote vectors in bold small letters, for example $\mathbf{v}$. If $\mathbf{v}$ is a vector of dimension d, then $v_k, k \in \{1,\ldots,d\}$ is the k-th component of $\mathbf{v}$, and $\mathbf{v}_I$, $I \subseteq \{1,\ldots,d\}$ denotes the vector with components $v_k, k \in I$ (with the v_k's arranged in the same order as in $\mathbf{v}$). Similarly, matrices will be written in bold capital letters, for example $\mathbf{A}$. If $\mathbf{A}$ is an $m \times n$ matrix, then A_{ij} represents the ij-th entry of $\mathbf{A}$, and $\mathbf{A}_{IJ}$ represents the matrix with entries $A_{ij}, i \in I, j \in J$.

A sequence is denoted as $\{a^n\}$, and a^n is the n-th element of this sequence. If $\hat{a}$ is a limit point of the sequence, we write $a^n \to \hat{a}$.

B.2 The optimisation problem

The dual that we are trying to solve is:

$$\max_{\boldsymbol{\alpha}} \quad \sum_{l=1}^{L} \boldsymbol{\alpha}_l^t\left(\boldsymbol{\Delta}_l^- - \boldsymbol{\Delta}_l^+\right) - 2\sum_{l=1}^{L}\sum_{k=1}^{L} R_{lk}\boldsymbol{\alpha}_l^t\mathbf{Y}_l\mathbf{K}\mathbf{Y}_k\boldsymbol{\alpha}_k \tag{59}$$

s.t.

$$\mathbf{0} \le \boldsymbol{\alpha} \le C\mathbf{1}$$

where $\boldsymbol{\alpha}_l = [\alpha_{1l},\ldots,\alpha_{Nl}]$, $\mathbf{Y}_l = \mathrm{diag}([y_{1l}\ldots y_{Nl}])$, $\mathbf{K} = \phi(\mathbf{X})^t\phi(\mathbf{X})$ and $\boldsymbol{\Delta}_l^{\pm} = (\Delta_l(\mathbf{y}_1, \pm y_{1l}), \ldots, \Delta_l(\mathbf{y}_N, \pm y_{Nl}))$. This can be written as the following optimisation problem:

Problem:

$$\max_{\boldsymbol{\alpha}} \quad f(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}^t\mathbf{Q}\boldsymbol{\alpha} + \mathbf{p}^t\boldsymbol{\alpha} \tag{60}$$

s.t.

$$\mathbf{l} \le \boldsymbol{\alpha} \le \mathbf{u}$$

Here the vector $\boldsymbol{\alpha} = [\alpha_{11}\ldots\alpha_{1L}, \alpha_{21}, \ldots\alpha_{NL}]^t$ and $\mathbf{Q} = 4\mathbf{Y}\mathbf{K}\otimes\mathbf{R}\mathbf{Y}$ where $\otimes$ is the Kronecker product. $\mathbf{Y} = \mathrm{diag}([y_{11}\ldots y_{1L}, y_{21}, \ldots y_{NL}])$. $\mathbf{p} = \boldsymbol{\Delta}_l^- - \boldsymbol{\Delta}_l^+$, $\mathbf{l} = \mathbf{0}$ and $\mathbf{u} = C\mathbf{1}$. We assume that $\mathbf{R}$ and $\mathbf{K}$ are both positive definite matrices. The eigenvalues of $\mathbf{K}\otimes\mathbf{R}$ are then $\lambda_i\mu_j$ (see, for example, Bernstein 2005), where λ_i are the eigenvalues of $\mathbf{K}$ and μ_j are the

eigenvalues of $\mathbf{R}$. Because all eigenvalues of both $\mathbf{R}$ and $\mathbf{K}$ are positive, so are the eigenvalues of $\mathbf{K} \otimes \mathbf{R}$ and thus $\mathbf{Q}$ is positive definite. Thus the dual we are trying to solve is a strictly convex quadratic program.

Our algorithm will produce a sequence of vectors $\{\boldsymbol{\alpha}^n\}$ where $\{\boldsymbol{\alpha}^i\}$ is the vector before the i-th iteration. For brevity, we denote the gradient $\nabla f(\boldsymbol{\alpha}^n)$ as $\mathbf{g}^n$ and the projected gradient $\nabla^P f(\boldsymbol{\alpha}^n)$ as $\tilde{\mathbf{g}}^n$. The algorithm stops when all the projected gradients have magnitude less than τ. It can be easily seen that by reducing τ, we can get arbitrarily close to the optimum.

Hence, in the following, we only need to prove that the algorithm will terminate in a finite number of steps.

B.3 Convergence

In this section we prove that the sequence of vectors $\boldsymbol{\alpha}^n$ converges.

Note the following:

- In each iteration of the algorithm, we optimise over a set of variables, which may either be a single variable α_{pl} or a pair of variables $\{\alpha_{pl}, \alpha_{ql}\}$.
- The projected gradient of all the chosen variables is non zero at the start of the iteration.
- At least one of the chosen variables has projected gradient with magnitude greater than τ.

Consider the n-th iteration. Denote by B the set of indices of the variables chosen: $B = \{(p,l)\}$ or $B = \{(p,l),(q,l)\}$. Without loss of generality, reorder variables so that the variables in B occupy the first $|B|$ indices. In the n-th iteration, we optimise f over the variables in B keeping the rest of the variables constant. Thus we have to maximise $h(\boldsymbol{\delta}_B) = f(\boldsymbol{\alpha}^n + [\boldsymbol{\delta}_B^t,\ \mathbf{0}^t]^t) - f(\boldsymbol{\alpha}^n)$. This amounts to solving the optimisation problem:

$$\max_{\boldsymbol{\delta}_B} \quad h(\boldsymbol{\delta}_B) = -\frac{1}{2}\boldsymbol{\delta}_B^t \mathbf{Q}_{BB}\boldsymbol{\delta}_B - \boldsymbol{\delta}_B^t\left(\mathbf{Q}\boldsymbol{\alpha}^n\right)_B + \mathbf{p}_B^t\boldsymbol{\delta}_B \tag{61}$$

s.t.

$$\mathbf{l}_B - \boldsymbol{\alpha}_B \leq \boldsymbol{\delta}_B \leq \mathbf{u}_B - \boldsymbol{\alpha}_B$$

Note that since $\mathbf{g}_B^n = -(\mathbf{Q}\boldsymbol{\alpha}^n)_B + \mathbf{p}_B$

$$h(\boldsymbol{\delta}_B) = -\frac{1}{2}\boldsymbol{\delta}_B^t \mathbf{Q}_{BB}\boldsymbol{\delta}_B + \boldsymbol{\delta}_B^t\mathbf{g}_B^n \tag{62}$$

$\mathbf{Q}_{BB}$ is positive definite since $\mathbf{Q}$ is positive definite, so this QP is convex. Hence standard theorems (see Nocedal and Wright 2006) tell us that $\boldsymbol{\delta}_B^*$ optimises (61) iff it is feasible and

$$\nabla^P h\left(\boldsymbol{\delta}_B^*\right) = \mathbf{0} \tag{63}$$

Then we have that $\boldsymbol{\alpha}^{n+1} = \boldsymbol{\alpha}^n + \boldsymbol{\delta}^*$, where $\boldsymbol{\delta}^* = [\boldsymbol{\delta}_B^{*t},\ \mathbf{0}^t]^t$. Now

$$\nabla h\left(\boldsymbol{\delta}_B^*\right) = -\mathbf{Q}_{BB}\boldsymbol{\delta}_B^* + \mathbf{g}_B^n \tag{64}$$

Also,

$$\begin{aligned}\mathbf{g}_B^{n+1} &= -\left(\mathbf{Q}\boldsymbol{\alpha}^{n+1}\right)_B + \mathbf{p}_B \\ &= -\left(\mathbf{Q}\left(\boldsymbol{\alpha}^n + \left[\boldsymbol{\delta}_B^{*t},\ \mathbf{0}^t\right]^t\right)\right)_B + \mathbf{p}_B\end{aligned} \tag{65}$$

$$
\begin{aligned}
&= \left(-\left(\mathbf{Q}\boldsymbol{\alpha}^n\right)_B + \mathbf{p}_B\right) - \mathbf{Q}_{BB}\boldsymbol{\delta}_B^* \\
&= \mathbf{g}_B^n - \mathbf{Q}_{BB}\boldsymbol{\delta}_B^* \\
&= \nabla h\left(\boldsymbol{\delta}_B^*\right)
\end{aligned}
$$

Then (65) means that:

$$
\tilde{\mathbf{g}}_B^{n+1} = \nabla^P h\left(\boldsymbol{\delta}_B^*\right) \tag{66}
$$

Using (63)

$$
\tilde{\mathbf{g}}_B^{n+1} = \nabla^P h\left(\boldsymbol{\delta}_B^*\right) = \mathbf{0} \tag{67}
$$

This leads us to the following lemma:

Lemma 1 *Let $\boldsymbol{\alpha}^n$ be the solution at the start of the n-th iteration. Let B be the set of indices of the variables over which we optimise. Let the updated solution be $\boldsymbol{\alpha}^{n+1}$. Then*

1. $\tilde{\mathbf{g}}_B^{n+1} = \mathbf{0}$
2. $\boldsymbol{\alpha}^{n+1} \neq \boldsymbol{\alpha}^n$
3. *If* $l_{jk} < \alpha_{jk}^{n+1} < u_{jk}$ *then* $g_{jk}^{n+1} = 0\ \forall (j,k) \in B$

Proof 1. This follows directly from (67).

2. If $\boldsymbol{\alpha}^{n+1} = \boldsymbol{\alpha}^n$, then $\boldsymbol{\delta}_B^* = \mathbf{0}$ and so, from (65), $\mathbf{g}_B^{n+1} = \nabla h(\mathbf{0}) = \mathbf{g}_B^n$. This means that from (67) $\tilde{\mathbf{g}}_B^n = \tilde{\mathbf{g}}_B^{n+1} = \mathbf{0}$. But this is a contradiction since we required that all variables in the chosen set have non zero projected gradient before the start of the iteration.

3. Since the final projected gradients are 0 for all variables in the chosen set (from (67)), if $l_{jk} < \alpha_{jk}^{n+1} < u_{jk}$ then $g_{jk}^{n+1} = 0\ \forall (j,k) \in B$. □

Lemma 2 *In the same setup as the previous lemma, $f(\boldsymbol{\alpha}^{n+1}) - f(\boldsymbol{\alpha}^n) \geq \sigma \|\boldsymbol{\alpha}^{n+1} - \boldsymbol{\alpha}^n\|^2$, for some fixed $\sigma > 0$.*

Proof

$$
\begin{aligned}
f\left(\boldsymbol{\alpha}^{n+1}\right) - f\left(\boldsymbol{\alpha}^n\right) &= h\left(\boldsymbol{\delta}_B^*\right) \\
&= -\frac{1}{2}\boldsymbol{\delta}_B^{*t}\mathbf{Q}_{BB}\boldsymbol{\delta}_B^* + \boldsymbol{\delta}_B^{*t}\mathbf{g}_B^n
\end{aligned} \tag{68}
$$

where $\boldsymbol{\delta}_B^*$ is the optimum solution of Problem (61). Now, note that since $\boldsymbol{\delta}_B^*$ is feasible and $\mathbf{0}$ is feasible and h is concave, we have that (see Nocedal and Wright 2006):

$$
\left(\mathbf{0} - \boldsymbol{\delta}_B^*\right)^t \nabla h\left(\boldsymbol{\delta}_B^*\right) \leq 0 \tag{69}
$$

$$
\Rightarrow \boldsymbol{\delta}_B^{*t}\mathbf{Q}_{BB}\boldsymbol{\delta}_B^* - \boldsymbol{\delta}_B^{*t}\mathbf{g}_B^n \leq 0 \tag{70}
$$

$$
\Rightarrow \boldsymbol{\delta}_B^{*t}\mathbf{Q}_{BB}\boldsymbol{\delta}_B^* \leq \boldsymbol{\delta}_B^{*t}\mathbf{g}_B^n \tag{71}
$$

This gives us that

$$
-\frac{1}{2}\boldsymbol{\delta}_B^{*t}\mathbf{Q}_{BB}\boldsymbol{\delta}_B^* + \mathbf{g}_B^{nt}\boldsymbol{\delta}_B^* \geq \frac{1}{2}\boldsymbol{\delta}_B^{*t}\mathbf{Q}_{BB}\boldsymbol{\delta}_B^* \tag{72}
$$

$$\Rightarrow f\left(\boldsymbol{\alpha}^{n+1}\right) - f\left(\boldsymbol{\alpha}^{n}\right) \geq \frac{1}{2}\boldsymbol{\delta}_B^{*t}\mathbf{Q}_{BB}\boldsymbol{\delta}_B^{*} \tag{73}$$

$$\Rightarrow f\left(\boldsymbol{\alpha}^{n+1}\right) - f\left(\boldsymbol{\alpha}^{n}\right) \geq \nu_B \frac{1}{2}\boldsymbol{\delta}_B^{*t}\boldsymbol{\delta}_B^{*} \tag{74}$$

where ν_B is the minimum eigenvalue of the matrix $\mathbf{Q}_{BB}$. Since $\mathbf{Q}_{BB}$ is positive definite always, this value is always greater than zero, and bounded below by the minimum eigenvalue among all 2×2 positive definite sub matrices of $\mathbf{Q}$. Thus

$$\begin{aligned} f\left(\boldsymbol{\alpha}^{n+1}\right) - f\left(\boldsymbol{\alpha}^{n}\right) &\geq \sigma \boldsymbol{\delta}_B^{*t}\boldsymbol{\delta}_B^{*} \\ &= \sigma \left\|\boldsymbol{\alpha}^{n+1} - \boldsymbol{\alpha}^{n}\right\|^2 \end{aligned} \tag{75}$$

for some fixed $\sigma \geq 0$. □

Theorem 1 *The sequence* $\{\boldsymbol{\alpha}^n\}$ *generated by our algorithm converges.*

Proof From Lemma 2, we have that $f(\boldsymbol{\alpha}^{n+1}) - f(\boldsymbol{\alpha}^n) \geq 0$. Thus the sequence $\{f(\boldsymbol{\alpha}^n)\}$ is monotonically increasing. Since it is bounded from above (by the optimum value) it must converge. Since convergent sequences are Cauchy, this sequence is also Cauchy. Thus for every ϵ, $\exists n_0$ s.t. $f(\boldsymbol{\alpha}^{n+1}) - f(\boldsymbol{\alpha}^n) \leq \sigma\epsilon^2\ \forall n \geq n_0$. Again using Lemma 2, we get that

$$\left\|\boldsymbol{\alpha}^{n+1} - \boldsymbol{\alpha}^{n}\right\|^2 \leq \epsilon^2 \tag{76}$$

for every $n \geq n_0$. Hence the sequence $\{\boldsymbol{\alpha}^n\}$ is Cauchy. The feasible set of $\boldsymbol{\alpha}$ is closed and compact, so Cauchy sequences are also convergent. Hence $\{\boldsymbol{\alpha}^n\}$ converges. □

B.4 Finite termination

We have shown that $\{\boldsymbol{\alpha}^n\}$ converges. Let $\hat{\boldsymbol{\alpha}}$ be a limit point of $\{\boldsymbol{\alpha}^n\}$. We will start from the assumption that the algorithm runs for an infinite number of iterations and then prove a contradiction.

Call the variable α_{ik} as τ-violating if the magnitude of the projected gradient $\tilde{g}_{ik}$ is greater than τ. Note that at every iteration, the chosen set of variables contains at least one that is τ-violating. Now suppose the algorithm runs for an infinite number of iterations. Then it means that the sequence of iterates $\boldsymbol{\alpha}^k$ contains an infinite number of τ-violating variables. Since there are only a finite number of distinct variables, we have that at least one variable figures as a τ-violating variable in the chosen set B an infinite number of times. Suppose that α_{il} is one such variable, and let $\{k_{il}\}$ be the sub-sequence in which this variable is chosen as a τ-violating variable.

Lemma 3 *For every* $\epsilon\ \exists k_{il}^0$ *s.t* $|\alpha_{il}^{k_{il}+1} - \alpha_{il}^{k_{il}}| \leq \epsilon\ \forall k_{il} > k_{il}^0$.

Proof We have that since $\boldsymbol{\alpha}^k \to \hat{\boldsymbol{\alpha}}$, $\boldsymbol{\alpha}^{k_{il}} \to \hat{\boldsymbol{\alpha}}$, and $\boldsymbol{\alpha}^{k_{il}+1} \to \hat{\boldsymbol{\alpha}}$. Thus, for any given $\epsilon\ \exists\ k_{il}^0$ such that

$$\left|\alpha_{il}^{k_{il}} - \hat{\alpha}_{il}\right| \leq \epsilon/2 \quad \forall k_{il} > k_{il}^0 \tag{77}$$

$$\left|\alpha_{il}^{k_{il}+1} - \hat{\alpha}_{il}\right| \leq \epsilon/2 \quad \forall k_{il} + 1 > k_{il}^0 \tag{78}$$

This gives, by triangle inequality,

$$\left|\alpha_{il}^{k_{il}+1} - \alpha_{il}^{k_{il}}\right| \leq \epsilon \quad \forall k_{il} > k_{il}^0 \tag{79}$$

□

Lemma 4 *$|\hat{g_{il}}| \geq \tau$, where $\hat{g_{il}}$ is the derivative of f w.r.t. α_{il} at $\hat{\boldsymbol{\alpha}}$.*

Proof This is simply because of the fact that $|g_{il}^{k_{il}}| \geq \tau$ for every k_{il}, and the absolute value of the derivative w.r.t. α_{il} is a continuous function of $\boldsymbol{\alpha}$, and $\boldsymbol{\alpha}^{k_{il}} \to \hat{\boldsymbol{\alpha}}$. □

We use some notation. If $\alpha_{il}^{k_{il}} \in (l_{il}, u_{il})$ and if $\alpha_{il}^{k_{il}+1} = l_{il}$ or $\alpha_{il}^{k_{il}+1} = u_{il}$, then we say that "$k_{il}$ is int → bd", where "int" stands for interior and "bd" stands for boundary. Similar interpretations are assumed for "bd → bd" and "int → int". Thus each iteration k_{il} can be of one of only four possible kinds: int → int, int → bd, bd → int and bd → bd. We will prove that each of these kinds of iterations can only occur a finite number of times.

Lemma 5 *There can be only a finite number of int → int and bd → int transitions.*

Proof Suppose not. Then we can construct an infinite sub-sequence $\{s_{il}\}$ of the sequence $\{k_{il}\}$ that consists of these transitions. Then we have that $g_{il}^{s_{il}+1} = 0$, using Lemma 1. Hence $g_{il}^{s_{il}+1} \to 0$. Since the gradient is a continuous function of $\boldsymbol{\alpha}$, and since $\boldsymbol{\alpha}^{s_{il}+1} \to \hat{\boldsymbol{\alpha}}$, we have that $g_{il}^{s_{il}+1} \to \hat{g}_{il}$. But this means $\hat{g}_{il} = 0$, which contradicts Lemma 4. □

Lemma 6 *There can be only a finite number of int → bd transitions.*

Proof Suppose that we have completed sufficient number of iterations so that all int → int and bd → int transitions have completed. The next int → bd transition will place α_{il} on the boundary. Since there are no bd → int transitions anymore, α_{il} will stay on the boundary henceforth. Hence there can be no more int → bd transitions. □

Lemma 7 *There can only be a finite number of bd → bd transitions.*

Proof Suppose not, i.e. there are an infinite number of bd → bd transitions. Let t_{il} be the sub-sequence of k_{il} consisting of bd → bd transitions. Now, the sequence $\alpha_{il}^{t_{il}} \to \hat{\alpha}_{il}$ and is therefore Cauchy. Hence $\exists n_1$ s.t.

$$\left|\alpha_{il}^{t_{il}} - \alpha_{il}^{t_{il}+1}\right| \leq \epsilon \ll u_{il} - l_{il} \quad \forall t_{il} \geq n_1 \tag{80}$$

Similarly, because the gradient is a continuous function of α, the sequence $\{g_{il}^{t_{il}}\}$ is convergent and therefore Cauchy. Hence $\exists n_2$ s.t.

$$\left|g_{il}^{t_{il}} - g_{il}^{t_{il}+1}\right| \leq \frac{\tau}{2} \quad \forall k_{il} \geq n_2 \tag{81}$$

Also, from the previous lemmas, $\exists n_3$ s.t. t_{il} is not int → int, bd → int or int → bd $\forall t_{il} \geq n_3$.

Take $n_0 = \max(n_1, n_2, n_3)$. Now, consider $t_{il} \geq n_0$. Without loss of generality, assume that $\alpha_{il}^{t_{il}} = l_{il}$. Then, since $|\tilde{g}_{il}^{t_{il}}| \geq \tau$, we must have that $g_{il}^{t_{il}} \geq \tau$. From (80), and using the fact that this is a bd → bd transition, we must have that

$$\alpha_{il}^{t_{il}+1} = l_{il} \tag{82}$$

From (81), we have that

$$g_{il}^{t_{il}+1} \geq \frac{\tau}{2} \tag{83}$$

From (82) and (83), we have that $\tilde{g}_{il}^{t_{il}+1} \geq \frac{\tau}{2}$, which contradicts Lemma 1. □

But if all int → int, int → bd, bd → int and bd → bd transitions are finite, then α_{il} cannot be τ-violating an infinite number of times and hence we have a contradiction. This gives us the following theorem:

Theorem 2 *Our algorithm terminates in finite number of steps.*

References

Bernstein, D. S. (2005). *Matrix mathematics*. Princeton: Princeton University Press.

Boutell, M., Luo, J., Shen, X., & Brown, C. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*(9), 1757–1771.

Cai, L., & Hofmann, T. (2007). Exploiting known taxonomies in learning overlapping concepts. In *Proceedings of the international joint conference on artificial intelligence* (pp. 714–719).

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

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM*: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Cheng, W., Dembczynski, K., & Huellermeier, E. (2010). Graded multilabel classification: the ordinal case. In *Proceedings of the international conference on machine learning*.

Crammer, K., & Singer, Y. (2003). A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, *3*, 1025–1058.

Dekel, O., & Shamir, O. (2010). Multiclass-multilabel classification with more classes than examples. In *Proceedings of the international conference on artificial intelligence and statistics*.

Elisseeff, A., & Weston, J. (2001). A kernel method for multi-labelled classification. In *Advances in neural information processing systems* (pp. 681–687).

Evgeniou, T., Micchelli, C. A., & Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*.

Fan, R. E., Chen, P. H., & Lin, C. J. (2005). Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, *6*, 1889–1918.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). Liblinear: a library for large linear classification. *Journal of Machine Learning Research*, *9*, 1871–1874.

Farhadi, A., Endres, I., Hoeim, D., & Forsyth, D. A. (2009). Describing objects by their attributes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Farhadi, A., Endres, I., & Hoeim, D. (2010). Attribute-centric recognition for cross-category generalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Hariharan, B., Vishwanathan, S. V. N., & Varma, M. (2010a). M3L code. http://research.microsoft.com/~manik/code/M3L/download.html.

Hariharan, B., Zelnik-Manor, L., Vishwanathan, S. V. N., & Varma, M. (2010b). Large scale max-margin multi-label classification with priors. In *Proceedings of the international conference on machine learning*.

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., & Sundarajan, S. (2008). A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the international conference on machine learning*.

Hsu, D., Kakade, S., Langford, J., & Zhang, T. (2009). Multi-label prediction via compressed sensing. In *Advances in neural information processing systems*.

Ji, S., Sun, L., Jin, R., & Ye, J. (2008). Multi-label multiple kernel learning. In *Advances in neural information processing systems* (pp. 777–784).

Keerthi, S. S., & Gilbert, E. G. (2002). Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, *46*(1–3), 351–360.

Lampert, C. H., Nickisch, H., & Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Lewis, D., Yang, Y., Rose, T., & Li, F. (2004). RCV1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*, 361–397.

Li, X., Wang, L., & Sung, E. (2004). Multi-label SVM active learning for image classification. In *Proceedings of the IEEE international conference on image processing* (pp. 2207–2210).

Lin, C. J., Lucidi, S., Palagi, L., Risi, A., & Sciandrone, M. (2009). Decomposition algorithm model for singly linearly-constrained problems subject to lower and upper bounds. *Journal of Optimization Theory and Applications*, *141*(1), 107–126.

McCallum, A. (1999). Multi-label text classification with a mixture model trained by EM. In *AAAI 99 workshop on text learning*.

Mitchell, T., Shinkareva, S., Carlson, A., Chang, K.-M., Malave, V., Mason, R., & Just, A. (2008). Predicting human brain activity associated with the meanings of nouns. *Science*, *320*, 1191–1195.

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2nd ed.). Berlin: Springer.

Palatucci, M., Pomerleau, D., Hinton, G., & Mitchell, T. (2009). Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods—support vector learning* (pp. 185–208).

Rifkin, R., & Khautau, A. (2004). In defense of one-vs-all classification. *Journal of Machine Learning Research*, *5*, 101–141.

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

Schapire, R. E., & Singer, Y. (2000). Boostexter: a boosting-based system for text categorization. *Machine Learning*, *39*(2/3), 135–168.

Snoek, C., Worring, M., van Gemert, J., Geusebroek, J.-M., & Smeulders, A. (2006). The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of ACM multimedia* (pp. 421–430).

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin Markov networks. In *Advances in neural information processing systems*.

The SIAM Text Mining Competition (2007). http://www.cs.utk.edu/tmw07/.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, *6*, 1453–1484.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: an overview. *International Journal of Data Warehousing and Mining*, *3*(3), 1–13.

Ueda, N., & Saito, K. (2003). Parametric mixture models for multi-labeled text. In *Advances in neural information processing systems*.

Vishwanathan, S. V. N., Sun, Z., Ampornpunt, N., & Varma, M. (2010). Multiple kernel learning and the SMO algorithm. In *Advances in neural information processing systems 23*.

Wang, G., Forsyth, D. A., & Hoeim, D. (2010). Comparative object similarity for improved recognition with few or no examples. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Zhang, M.-L., & Wang, Z.-J. (2009a). Mimlrbf: Rbf neural networks for multi-instance multi-label learning. *Neural Computing*, *72*(16–18), 3951–3956.

Zhang, M.-L., & Wang, Z.-J. (2009b). Feature selection for multi-label naive Bayes classification. *Information Sciences*, *179*(19), 3218–3229.