Multi-Label Learning under Feature Extraction Budgets

Pekka Naula[a,*], Antti Airola[a], Tapio Salakoski[a], Tapio Pahikkala[a]

[a] *Department of Information Technology, 20014, University of Turku, Finland*

Abstract

We consider the problem of learning sparse linear models for multi-label prediction tasks under a hard constraint on the number of features. Such budget constraints are important in domains where the acquisition of the feature values is costly. We propose a greedy multi-label regularized least-squares algorithm that solves this problem by combining greedy forward selection search with a cross-validation based selection criterion in order to choose, which features to include in the model. We present a highly efficient algorithm for implementing this procedure with linear time and space complexities. This is achieved through the use of matrix update formulas for speeding up feature addition and cross-validation computations. Experimentally, we demonstrate that the approach allows finding sparse accurate predictors on a wide range of benchmark problems, typically outperforming the multi-task lasso baseline method when the budget is small.

Keywords: Feature selection, Greedy forward selection, Multi-label learning, Regularized least-squares

1. Introduction

Multi-label learning (Tsoumakas et al., 2010) concerns the problem of learning to make predictions about the association between data points and a set of candidate labels. In multi-label classification, one aims to predict which of the available labels are relevant with respect to the data point of interest, and which are not. In label ranking (see e.g. Hüllermeier et al. (2008)) one rather predicts the

*Corresponding author. Tel.: +358 405022708; fax: +358 2 2410154

Email addresses: pekka.naula@utu.fi (Pekka Naula), antti.airola@utu.fi (Antti Airola), tapio.salakoski@utu.fi (Tapio Salakoski), tapio.pahikkala@utu.fi (Tapio Pahikkala)

ordering over the set of labels, where the labels best matching the data point appear at the top of the ordering. The applications of multi-label learning are varied, since in almost any domain of interest there are usually several interesting properties that can be simultaneously used to describe an object. For example, an image often has several objects appearing in it, a piece of music or a movie represents multiple genres, or a newspaper article may belong to several topic categories.

Multi-label methods are often divided into two categories: problem transformation methods and algorithm adaptation methods (Tsoumakas and Katakis, 2007). The former aim at dividing the original problem into one or more single-label classification or regression problems whereas the latter are based on extending existing single-task approaches to multi-label learning. There are a rich family of different approaches for both categories.

Two of the most common problem transformation methods are binary relevance method (BR) and label power-set method (LP). While BR divides the multi-label problem into binary single-task problems, one task per label, LP creates a binary single-label problem for every possible label combination. Compared to BR, LP has the advantage of being able to model the correlation between the labels, but this comes at a steep computational price as the number of possible label combinations grows exponentially with respect to the size of the label set. More advanced transformation methods such as RAKEL (Tsoumakas et al., 2011a) have been developed to overcome this problem. Examples of single-task classifiers adapted to make use of label correlations include the ML-kNN (Zhang and Zhou, 2007) algorithm, that extends the K-nearest neighbors algorithm to multi-label classification, and the ML-C4.5 (Clare and King, 2001) multi-label decision tree method. For a comprehensive overview and experimental comparison of multi-label methods, we refer to Madjarov et al. (2012).

In this work we consider the BR type of setting, where for each label one constructs a linear predictor, that produces scorings from which the classifications or rankings are derived. In many applications *sparsity*, meaning that for a significant number of features the corresponding coefficients in the models are set to zero, is a desirable property. The three most common motivations for learning sparse models are the following. Enforcing sparsity has a regularizing effect which may help to prevent overfitting, models depending only on a few variables are easier to understand and explain by human experts, and sparse models are cheaper to predict with than dense ones. The focus of this paper is especially on the third point of view.

As pointed out by Xu et al. (2012), the prediction cost can, in turn, be divided into the times required for evaluating the models and for extracting the feature values. For linear models, the evaluation time is proportional to the number of nonzero model entries, totaled over all models multi-label learning. In contrast, the feature extraction time is proportional to the set of unique features used for prediction. The feature value is extracted only once for a single data point, while the value can be used to predict several labels. The difference between the two types of sparsity is illustrated in the following example, where two linear models have the same model evaluation cost, but different feature extraction cost. Let

$$\mathbf{W}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 \end{pmatrix}, \qquad \mathbf{W}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 2 & 3 & -1 & 2 \\ 0 & 0 & 0 & 0 \\ 3 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

denote the matrices determining two sparse linear models. The rows and columns of both matrices correspond to features and tasks, respectively. Both matrices have the same number of non-zero coefficients, but $\mathbf{W}_1$ requires all the features for prediction, whereas $\mathbf{W}_2$ requires only two of them.

The feature extraction costs are dominant to the model evaluation costs in many real-world tasks, and hence the focus of this paper is the minimization of the extraction cost. Our problem definition is quite similar to that of *budgeted learning* considered recently by Cesa-Bianchi et al. (2011); Hazan and Koren (2012), the difference being that our work considers multi-label instead of single-task learning, and we do not consider settings where different features may be selected for different data points.

As a motivating example, consider an image recognition system that simultaneously predicts several properties of a given input image in real-time. Since each feature used for prediction is obtained from a possibly computationally expensive feature extractor, one must minimize the number of required features to ensure real-time recognition. A similar setting is commonly encountered in medical testing, where we want to perform as few tests as possible, yet make reliable diagnoses for a patient. To summarize, we consider the setting in which the number of features must be limited even if it decreases the prediction performance, because enforcing sparsity due to the high feature acquisition costs is necessary in numerous practical applications.

Two popular approaches for learning sparse models are the filter methods, that perform feature selection independently of the learning algorithm trained on the selected features, and wrapper or embedded methods where the selection process is optimized for the learning algorithm. The most prominent of the latter type of methods are the method known as lasso or basis pursuit, and the family of greedy search algorithms. There is empirical evidence in the literature favoring lasso over greedy methods (Chen et al., 1998) when the amount of selected features is large. Moreover, it has been shown that if the model underlying the data is truly sparse lasso converges to it (Zhao and Yu, 2006). However, in the setting considered in this work one must select only a small number of features even if the model is not truly sparse. Consequently, since the lasso methods are based on convex regularization, the smaller is the set of selected features, the worse will be the bias caused by the regularization on the learnt model (Zhang, 2011). This phenomenon does not concern the greedy methods, as they are based on a different selection principle.

In the recent years, techniques applicable to learning sparse models in the single-label setting have been extended to the multi-label setting. As a typical example of filter methods, Doquire and Verleysen (2011) proposed a greedy method that combines a mutual information based selection criterion with a variant of the LP transformation method. Zhang et al. (2009) proposed a naive Bayes multi-label method that applies as a first stage principal component analysis in order to reduce the feature set dimensionality followed by a genetic algorithm based feature selection phase. However, the reliance on PCA for dimensionality reduction makes this and similar methods unsuitable for the setting considered in this work, as they still need all the original features during prediction time.

Among the the selection methods optimized for the learning algorithm, sparsity enforcing matrix norm-based regularization approaches, that extend the commonly used l_1-norm to the multi-task setting, have shown to be especially promising (Turlach et al., 2005; Liu et al., 2009; Obozinski et al., 2010; Zhang et al., 2010). As a representative of the state-of-the art in this area, we consider the coordinate descent training approach for the $l_{1,\infty}$-regularization based multi-task lasso (Liu et al., 2009). The optimization criterion for the method directly enforces such sparsity structure that leads to minimal number of features being used in the model (see matrix $\mathbf{W}_2$). Thus, the method provides a natural baseline for comparing our work.

We extend the greedy RLS approach (Pahikkala et al., 2010, 2012), a greedy forward selection method for regularized least-squares proposed by some of the present authors, to multi-label setting. The work continues the work of Naula et al.

(2011b,a), where a high-level description of the idea and some preliminary experimental results were presented. We prove that the resulting training algorithm has linear time and space complexities, making it computationally highly competitive for example with the most efficient known coordinate descent training algorithms proposed for the lasso-type of learning methods. In our experiments, we compare the predictive performance of the multi-label greedy RLS and multi-task lasso approaches over several real-world data sets, in order to determine which approach, if any, leads to higher predictive performance. The results suggest that whenever one wants to strongly enforce sparsity, the greedy approach is preferable, as on small feature subsets multi-label greedy RLS consistently outperforms multi-task lasso.

2. Methods

Here, we present the basic concepts and notations relevant for the following considerations. By $[n]$ we denote the index set $\{1 \ldots n\}$. We use bold lowercase and uppercase letters for denoting vectors and matrices, respectively. Given a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ and index sets $\mathcal{R} \subseteq [m]$ and $\mathcal{S} \subseteq [n]$, we use $\mathbf{M}_{\mathcal{R},\mathcal{S}}$ for denoting the submatrix containing the rows and columns indexed by $\mathcal{R}$ and $\mathcal{S}$, respectively. Further, $\mathbf{M}_{\mathcal{R}}$, $\mathbf{M}_{:,\mathcal{S}}$, and $\mathbf{M}_{i,j}$ are shorthands for, $\mathbf{M}_{\mathcal{R},[n]}$, $\mathbf{M}_{[m],\mathcal{S}}$, and $\mathbf{M}_{\{i\},\{j\}}$, respectively. We use analogous notations also for vectors.

Let

$$D = \{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^n, \mathbf{y}^n)\}$$

be a training set of size n, where $\mathbf{x}^i \in \mathbb{R}^d$ and $\mathbf{y}^i \in \mathbb{R}^t$ are the feature and the label vectors of the ith instance, respectively, and d and t are the numbers of features and labels. The label vectors can be encoded so that $\mathbf{y}^i_j = 1$ if the ith instance is associated with the jth label and $\mathbf{y}^i_j = -1$ otherwise.

Our aim is to learn from D a real valued function

$$f_l : \mathbb{R}^d \to \mathbb{R}.$$

for each label $1 \leq l \leq t$, that is expected to predict a positive value if $\mathbf{x}$ is associated with the label and negative values otherwise.

2.1. Optimization Framework

In the following, we assume that the feature representations of the training instances are stored as row vectors in the data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. Thus the i, jth entry of $\mathbf{X}$ contains the value of the jth feature in the ith training example. Moreover, the labels for the training data points are stored in the matrix $\mathbf{Y} \in \mathbb{R}^{n\times t}$, where the i, jth entry is 1 if the ith training example has the jth label, and -1 otherwise. The predictor is a linear function that can be written as $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{W}$, where $\mathbf{W} \in \mathbb{R}^{d\times t}$ is a matrix of parameters and $\mathbf{x} \in \mathbb{R}^d$ is a column vector containing the feature values of a data point.

Training of multi-label predictors with training data $\mathbf{X}, \mathbf{Y}$ can be expressed as finding a solution to the following problem:

$$\begin{aligned} &\mathrm{argmin}_{\mathbf{W}\in\mathbb{R}^{d\times t}} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 \\ &\text{subject to } C(\mathbf{W}) \end{aligned} \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius matrix norm and C is a constraint function.

One of the most well-known constraint functions is the quadratic one

$$C(\mathbf{W}) = \|\mathbf{W}\|_F^2 < r, \tag{2}$$

where $r \in \mathbb{R}^+$. The feature selection setting, in which the number of feature extractors must not exceed a given limit, can be expressed as the following constraint:

$$C(\mathbf{W}) = |\{i \mid \exists j, \mathbf{W}_{i,j} \neq 0\}| \leq k, \tag{3}$$

where $k \in \mathbb{N}$. The discrete and non-convex nature of the constraint makes its optimization challenging. In the literature, there are two widely used strategies for tackling this problem. The first is to approximate the constraint with continuous and convex functions and the second to use combinatorial optimization techniques.

Multi-task lasso (Turlach et al., 2005) approximates (3) with the following:

$$C(\mathbf{W}) = \sum_{i=1}^{d} \max_j |\mathbf{W}_{i,j}| \leq r.$$

Liu et al. (2009) have shown how the Multi-task lasso optimization problem can be efficiently solved for large data sets using the coordinate descent method. This approach is considered as a baseline method in our experiments. The computational complexity of the training method of Liu et al. (2009) is $O(nd^2 + ndt + htd^2 + hdt\log(t))$, where n, d, and t denote the numbers of data points, features and labels, respectively, and h denotes the number of iterations performed. The number of iterations h depends on the magnitude of the regularization parameter and desired accuracy of solution, in the experiments we noted that the method in practice usually converged within tens of iterations. The memory consumption of the method is $O(nd+nt+d^2)$. The method scales well with respect to the number of instances and labels, but has a quadratic dependency on the dimensionality of the data, limiting its scalability to very high dimensional problems.

Multi-label greedy RLS uses of both (2) and (3) simultaneously. In the next section, we present a novel algorithm for solving the induced optimization problem efficiently by implementing greedy forward selection search using sophisticated matrix algebra shortcuts for speeding up computations.

2.2. Multi-label greedy RLS

Algorithm 1 High-level pseudocode of Multi-label greedy RLS.

```
 1: $\mathcal{S} \leftarrow \emptyset$        ▷ The current set of selected features common for all tasks.
 2: while $|\mathcal{S}| < k$ do        ▷ Select $k$ common features.
 3:     $e \leftarrow \infty$
 4:     $b \leftarrow 0$
 5:     for $i \in \{1, \ldots, d\} \setminus \mathcal{S}$ do        ▷ Test all features before selecting.
 6:         $e_{avg} \leftarrow 0$
 7:         for $j \in \{1, \ldots, t\}$ do
 8:             $e_{i,j} \leftarrow \mathcal{L}(\mathbf{X}_{:,\mathcal{S}\cup\{i\}}, \mathbf{Y}_{:,j})$        ▷ Compute LOO performance for task $j$.
 9:             $e_{avg} \leftarrow e_{avg} + e_{i,j}/t$
10:         if $e_{avg} < e$ then
11:             $e \leftarrow e_{avg}$
12:             $b \leftarrow i$
13:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{b\}$        ▷ Select the feature whose addition leads to lowest
    LOO-error.
14: $\mathbf{W} \leftarrow \mathcal{A}(\mathbf{X}_{:,\mathcal{S}}, \mathbf{Y})$        ▷ Train final models using the selected features.
15: return $\mathbf{W}, \mathcal{S}$
```

Let us next consider solving (1) using only the quadratic constraint (2). With the Lagrange multipliers technique, one can determine such a real-valued multiplier $\lambda > 0$ for which the following unconstrained objective function provides an equivalent solution:

$$\underset{\mathbf{W}\in\mathbb{R}^{d\times t}}{\operatorname{argmin}} \|\mathbf{XW} - \mathbf{Y}\|_F^2 + \lambda\|\mathbf{W}\|_F^2.$$

This multiplier is in the literature often called the regularization parameter, and the above modification of the optimization problem leads to the well-known regularized least-squares (RLS) learning method, also commonly known in the literature as ridge regression (Hoerl and Kennard, 1970), or least-squares support vector machine (Suykens et al., 2002). The RLS induces a convex optimization problem, with a closed-form solution expressible as a solution to a system of linear equations. While the quadratic constraint has a regularizing effect guarding against overfitting, it does not enforce sparsity of the learned model. Hence, we use the additional constraint (3). However, due to the exponential number of different feature combinations, there is no longer a polynomial time algorithm for finding the global optimum for (1) when using both the constraints (2) and (3). Thus, we resort to a greedy search algorithm for traversing through the power set of features.

We apply the greedy forward selection heuristic. By greedy, we indicate that the algorithm starts from an empty set of features and adds one feature at a time to the set but never removes any selected features from the set. At each search step, each non-selected feature is tested by temporarily adding it to the feature set, and computing the mean squared error obtained via leave-one-out (LOO) cross-validation for the resulting feature set (for a description of LOO performance, we refer to e.g. Lachenbruch (1967); Elisseeff and Pontil (2003)). The feature whose addition leads to lowest error is selected, after which the search proceeds to the next step. Once the allocated number of features has been chosen, the search stops resulting in the final model.

In Algorithm 1 we describe the high-level pseudocode of the resulting feature selection algorithm. The outermost loop adds one feature at a time into the set of selected features $\mathcal{S}$ until the size of the set has reached the sparsity budget k. The middle loop goes through every feature that has not yet been added into the set of selected features. By $\mathcal{L}(\mathbf{X}_{:,\mathcal{S}\cup\{i\}}, \mathbf{Y}_{:,j})$ we represent the mean-squared LOO error, computed for a predictor trained for the jth task, using the features indexed by the set $\mathcal{S}\cup\{i\}$. Thus for the ith feature available for addition, the inner loop computes the average LOO performance over the t tasks for RLS predictors trained using the features $\mathcal{S}\cup\{i\}$. After going through all feature candidates, the algorithm then adds the feature with the best average LOO performance into the set of selected features. By $\mathcal{A}(\mathbf{X}_{:,\mathcal{S}}, \mathbf{Y})$, we denote a routine that trains the final predictor, by solving the problem (1) subject to the quadratic constraint (2), using only the features in the set $\mathcal{S}$.

Algorithm 1 can be in principle be straightforwardly implemented as a wrapper algorithm (Kohavi and John, 1997), meaning that a computational wrapper

Table 1: Datasets.

Data sets	domain	labels	features	instances	cardinality	density
Scene	image	6	294	2407	1.074	0.179
Yeast	biology	14	103	2417	4.237	0.303
Emotions	music	6	72	593	1.868	0.311
CAL500*	music	87	68	502	23.010	0.264
Mediamill*	text	9	120	41583	3.059	0.340
Delicious	text	983	500	16105	19.020	0.019
Tmc2007	text	22	49060	28596	2.158	0.098

uses a black-box RLS solver, re-training it for each round of selection process, tested feature, task and round of cross-validation. While the resulting algorithm does have polynomial runtime, it is still highly impractical even for modest sized data sets. In our previous work we have shown that for single-task learning problem, greedy RLS search can be implemented with linear time and memory complexities via matrix algebraic optimization (Pahikkala et al., 2010, 2012). Next, we generalize these results to the multi-label learning setting.

Theorem 1. *On a data set with n data points, d features and t labels, multi-label greedy RLS can select k features in $O(kndt)$ time and with $O(nd + nt)$ memory consumption.*

Proof. For detailed implementation description, computational complexity analysis and proof of correctness, see Appendix B. □

For high-dimensional data multi-label greedy RLS can be expected to be faster than multi-task lasso, due to latters quadratic dependency on the dimensionality of the data. On the other hand, if the dimensionality is not too large, the methods can expected to perform similarly with respect to running times on small budget problems.

3. Experiments

In this section we present an experimental evaluation of the proposed multi-label greedy RLS (ML-gRLS) method, with a comparison to the multi-task lasso (MT-Lasso) baseline. We also show results for a popular multi-label method, ML-kNN (Zhang and Zhou, 2007), in order to provide a baseline on how well a widely applied multi-label method can perform on these problems when not subject to budget constraints. First, we describe the considered data sets. Next, we consider the problem of parameter selection for the methods, proposing suitable selection strategies based on experimental evidence. Finally, we evaluate the methods on varying feature budget sizes on seven real-world data sets representing a variety of different types of application domains.

All the test runs for the ML-gRLS method are carried out using the implementation in RLScore[1], a publicly available open source machine learning library developed by some of the present authors. The software is implemented using the Python programming language, and the NumPy and SciPy libraries. MT-lasso is also implemented in Python according to algorithm presented in (Liu et al., 2009). The experiments for the ML-kNN[2] method are performed using the implementation of the Mulan Java library (Tsoumakas et al., 2011b).

3.1. Datasets

We carry out our experiments using seven publicly available data sets (Scene, Yeast, Emotions, CAL500, Mediamill, Delicious and Tmc2007) that can be found from the web site of the Mulan library. The data sets represent different application domains such as biology, text or music. The properties of the data sets are summarized in Table 1. Two of the data sets are pre-processed by removing some labels from the original ones (denoted by * in the table) to carry out 10-fold cross-validation properly. For CAL500 we select only those labels that include more than 40 instances and for Mediamill only those labels that include more than 5000 instances. The table presents the number of labels, features and instances, and two often used characteristics in multi-label research, *cardinality* and *density* (Tsoumakas et al., 2010).

3.2. Parameter selection

Both the ML-gRLS and MT-Lasso optimization problems incorporate a regularization parameter λ, whose correct selection is crucial for achieving good predictive performance, and in the case of MT-Lasso, the parameter also directly controls the number of selected features. Selecting suitable parameter value is not straightforward, in this section we explore this issue experimentally and propose solutions. All the results presented in this section are based on 10-fold cross-validation (excluding large datasets Delicious and Tmc2007 for which 5-fold cross-validation is implemented), where the feature selection and model construction is performed on nine training folds, test performance computed on the tenth test fold, and final performances computed as averages over the ten cross-validation rounds.

[1]available at `https://github.com/aatapa/RLScore`

[2]available at `http://mulan.sourceforge.net`

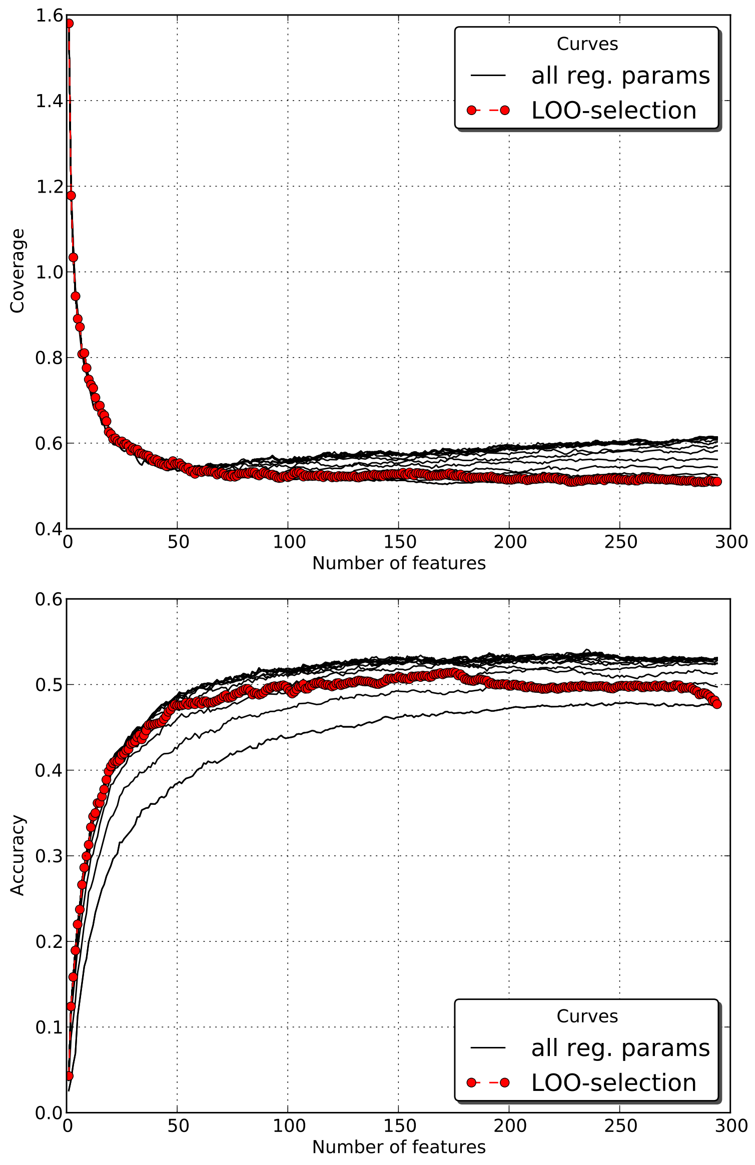

Figure 1: The performance curves of the ML-gRLS method with respect to coverage and accuracy criteria on the Scene data. All reg.params -curves represent prediction performances over different regularization parameter values and LOO-selection -curve represents the prediction performance based on LOO-error for a given budget. Only the curves on the grid $[2^{-15}, 2^{-14}, ..., 2^{5}]$ are shown in the figure.

First, we consider the selection of λ for ML-gRLS. In Figure 1 we present the effects of using different regularization parameter values on the Scene dataset. We test each of the parameters in range [2^{-15}, 2^{-14}, ..., 2^{15}]. Figure 1 plots for different regularization parameters the coverage (top) and accuracy (bottom) against the number of selected features. The curves demonstrate that the correct choice of the regularization parameter is essential for finding a good model. Since ML-gRLS computes for each selected feature the leave-one-out squared error, a natural approach is to use this directly as the selection criterion. In Figure 1 we have plotted the test errors achieved by choosing for each number of selected features from the grid the regularization parameter with lowest leave-one-out error. The approach proves to be reasonable, finding for the coverage measure the near optimal parameters, and works also well for small budget values for the accuracy criterion. However, the suboptimal parameter selection in terms of accuracy for large feature budgets suggests that the selection heuristic may not always perform well, possibly due to overfitting.

For MT-Lasso algorithm, the first practical challenge is how to select λ in order to get a certain number of features with non-zero coefficients. As discussed by Friedman et al. (2010), the coordinate descent optimization based techniques for lasso training do not allow one to directly control the number of selected features, but rather it is necessary to test different regularization parameters and observe the number of selected features. Large enough value a for λ sets all the coefficients exactly equal to zero, whereas small enough value b sets all the coefficients to non-zero, that is, in the former case the model includes zero features and in the latter case it includes all the candidate features. Let $rng = [a, b]$ be a range of the regularization parameters that generates all the possible sizes of the models in terms of the number of the features, where a and b have been selected experimentally. The approach used for example by Friedman et al. (2010) is to simply generate many candidate values for λ on the range rng, then build the models and find out how many features are selected for each value. In Figure 2 we present an experiment on Emotions dataset, where we plot the performance curve (Macro-averaged AUC) and the number of selected features curve over a regularization parameter λ. The scale for performance curve is set on the left vertical axis and the scale for the number of features curve is set on the right vertical axis. The range for the regularization parameter is $rng = [40, 0]$ and a new model is created in every point on equally distributed grid $[a, a - 0.1, a - 0.2, \ldots, b]$. The results demonstrate that the approach allows recovering models for a wide range of feature budgets, though at a quite steep computational price, and it cannot be guaranteed that all feature budget sizes are represented.

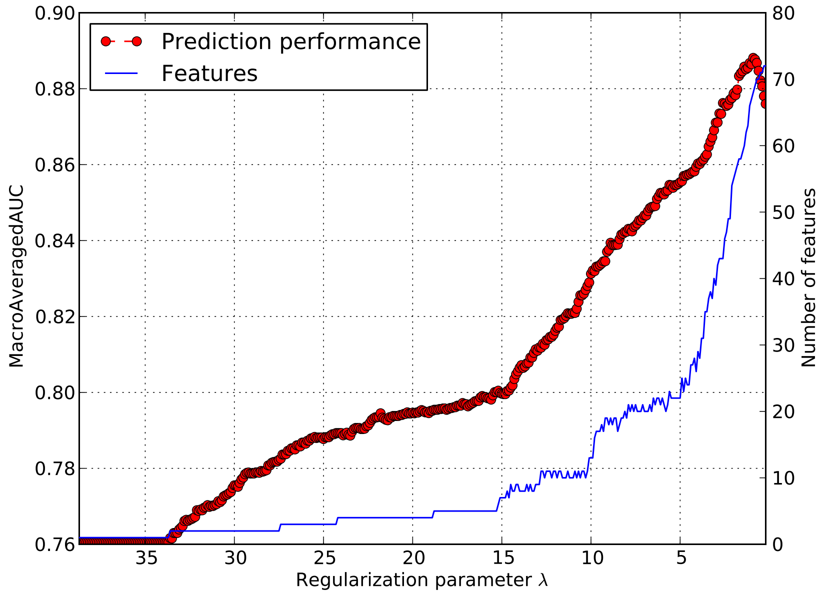

Figure 2: The prediction performance curve and the number of features curve of the MT-Lasso method with respect to different regularization parameters on Emotion data.

The second challenge while using MT-Lasso is finding an optimal value λ for some given budget. This is due to the fact that an infinite number of regularization parameter values on range rng return equal number of features (=same budget) but possibly different prediction performance. Figure 2 presents clearly this phenomenon by showing that in areas where the number of selected features curve levels constant, the predictive performance curve still keeps changing. From the Figure 2 one can be derive visually an observation we made also on other data sets, that the optimal λ value for a given feature budget tends to be the smallest parameter value resulting in the same number of features being selected.

3.3. Experimental comparison

We carry out standard ten-fold cross-validation for all the methods on the five small data sets and five-fold cross-validation on the two large data sets. We calculate the classification performance for seven widely used measures. For a description of measures see Appendix A. We compare ML-gRLS and MT-Lasso methods over different sizes of feature budgets. For each data set, budget size and performance measure we perform the Wilcoxon signed-rank test over the cross-validation results at 0.05 significance level in order to determine, whether the performance differences between the two methods are statistically significant.

We search the grid in range $[2^{-15}, 2^{-14}, ..., 2^{15}]$ for selecting the regularization parameters λ for ML-gRLS, choosing for each feature set size the parameter based on leave-one-out cross-validation error. We train the method starting from zero features, up until all the features have been selected. For MT-Lasso, we first determine the range for regularization parameters that generates the minimum and maximum number of candidate features, and then define a parameter grid between these with a step size of 0.1 (a step size of 5.0 is selected for the large datasets). In cases where the grid does not generate all the possible feature subset sizes, missing performance values are linearly interpolated.

Figure 3 shows the prediction performance curves for MT-Lasso and ML-gRLS methods on all seven datasets. The figure plots the average of the cross-validation results over the number of features in terms of macro-averaged AUC, (left) and Hamming Loss (right). The results for rest of the performance criteria show very similar behavior and are therefore left out from the figures, though some of the selected results are presented in the following tables.

It can be noted, visually, that ML-gRLS outperforms MT-Lasso over a low budgets on all data sets but CAL500 and TMC2007 with respect to macro averaged AUC and Hamming-Loss. The performance differences between these two methods on CAL500 dataset are quite small which might be due the weak learning results, shown in Figure 3, that does not show any improvement over entire budget range. It is also notable that ML-gRLS outperforms MT-Lasso over all budgets on the bigger data sets, such as Mediamill, Delicious and Tmc2007, with the exception that MT-lasso outperforms ML-gRLS over very small budget sizes on Tmc2007 (see Figure 3).

Table 2 summarizes results in terms of all seven performance measures for some selected budgets marked as *low*, *med* and *high*, where budget size is 10, 45 and 80 percents of all the candidate features, respectively. Moreover Table 3 summarizes results for large datasets Delicious and Tmc2007 over small budgets 100 and 50 features, respectively. Each element in the table in this comparison contains mean and standard deviation values denoted by (mean$\pm$ std.dev). The statistically significant differences between the results of ML-gRLS and MT-Lasso according to Wilcoxon test are marked in bold. The "$\uparrow$" indicates that the larger the value is, the better is the result and "$\downarrow$" indicates the lower the better. Tables 2 and 3 present also the results of ML-kNN method with respect to seven performance measures on the seven datasets. ML-kNN derives a model that includes all the available features ignoring feature selection process.

The tables reveal the same findings that could be observed from the figures, the ML-gRLS method is always comparable to MT-Lasso in terms of all evaluation criteria and budget sizes used in this study on all data sets but CAL500 and Tcm2007, and significantly outperforms baseline method on small budgets. The results of ML-kNN are not directly comparable with ML-gRLS and MT-Lasso due to different budget sizes. However, it can be seen that overall performances of feature selection methods in terms of all seven performance measures are slightly worse on small data sets to that of ML-kNN. On the other hand, on the two largest data sets, ML-gRLS clearly outperforms ML-kNN according to most performance measures, which might be due the large number of irrelevant features in the datasets. This may cause a worse prediction performance and results in an expensive dense model. To conclude, while MT-Lasso is competitive with ML-gRLS for unrestricted feature budgets, when the number of features is restricted ML-gRLS clearly outperforms MT-Lasso, making it the preferable method for such settings.

4. Conclusions

In this paper, we considered the problem of multi-label learning under restricted feature extraction budgets. That is, we concentrate on minimizing the number of features required for simultaneous prediction of several labels for a given data point. We proposed a novel greedy multi-label learning algorithm, that achieves high computational efficiency through matrix algebraic optimizations. As a baseline method we tested multi-task lasso based on $l_{1,\infty}$-regularization.

Since the lasso controls the number of features only implicitly, enforcing strict budget constraints requires careful and time-consuming tuning of regularization parameter. In contrast, explicit control is possible with greedy methods. Moreover, small budgets are not the strongest area of lasso methods, because l1-regularization shrinks also the relevant features in addition to the non-relevant ones. This can be observed in our experimental results in which ML-gRLS was competitive on all considered real-world data sets and significantly outperformed MT-Lasso on small budgets.

In this work we have made the assumption of each feature having equal extraction cost and each being independently produced. However, there are many applications for which this is not the case. For example, in visual recognition systems and their applications, features are often extracted in groups rather than individually, that is, the feature extraction cost is common for a whole group of features and it pays to simultaneously select all features belonging to such group instead of single feature at a time. Recently, many of the popular feature selec-

Table 2: Performance on the Scene, Yeast, Emotions, CAL500, and Mediamill.

	ML-gRLS			*MT-lasso*			*ML-kNN*
	Low	*Med*	*High*	*Low*	*Med*	*High*	
SCENE							
Z.-O.Loss ↓	**0.600±0.029**	**0.529±0.023**	0.535±0.031	0.990±0.007	0.724±0.013	0.537±0.024	0.367±0.032
Ham.Loss ↓	**0.118±0.005**	**0.106±0.006**	0.106±0.007	0.178±0.002	0.133±0.003	0.106±0.006	0.085±0.008
Accuracy ↑	**0.431±0.027**	**0.505±0.022**	0.498±0.029	0.010±0.007	0.292±0.015	0.492±0.022	0.674±0.030
One-error ↓	**0.285±0.017**	0.246±0.026	0.246±0.025	0.369±0.033	0.257±0.020	0.239±0.024	0.229±0.028
Coverage ↓	**0.582±0.054**	0.520±0.063	0.515±0.060	0.717±0.046	0.528±0.045	0.502±0.050	0.470±0.051
Rank.Loss ↓	**0.100±0.010**	0.087±0.011	0.086±0.010	0.126±0.010	0.089±0.007	0.083±0.008	0.077±0.008
M.avg.AUC ↑	**0.910±0.009**	**0.923±0.011**	0.924±0.011	0.864±0.007	0.916±0.009	**0.926±0.010**	0.933±0.007
YEAST							
Z.-O.Loss ↓	**0.881±0.017**	**0.851±0.019**	**0.853±0.022**	0.987±0.007	0.931±0.014	0.872±0.019	0.812±0.025
Ham.Loss ↓	**0.210±0.009**	**0.200±0.009**	**0.200±0.009**	0.232±0.009	0.214±0.008	0.203±0.009	0.194±0.013
Accuracy ↑	**0.466±0.016**	**0.494±0.016**	**0.496±0.018**	0.339±0.012	0.420±0.016	0.477±0.017	0.519±0.020
One-error ↓	**0.237±0.023**	**0.221±0.020**	**0.225±0.019**	0.249±0.023	0.248±0.022	0.238±0.017	0.230±0.020
Coverage ↓	6.574±0.244	6.386±0.226	6.374±0.235	6.551±0.206	6.387±0.210	6.349±0.219	6.232±0.278
Rank.Loss ↓	**0.182±0.016**	**0.170±0.015**	0.171±0.015	0.195±0.015	0.181±0.016	0.172±0.015	0.166±0.017
M.avg.AUC ↑	**0.654±0.022**	**0.693±0.016**	**0.699±0.015**	0.630±0.014	0.670±0.012	0.691±0.014	0.688±0.018
EMOTIONS							
Z.-O.Loss ↓	**0.752±0.069**	0.730±0.054	0.740±0.065	0.891±0.060	0.731±0.074	0.732±0.057	0.719±0.045
Ham.Loss ↓	**0.213±0.027**	0.202±0.018	0.203±0.026	0.255±0.027	0.202±0.030	0.192±0.021	0.194±0.018
Accuracy ↑	**0.459±0.067**	**0.512±0.046**	0.493±0.065	0.238±0.062	0.459±0.071	0.499±0.049	0.533±0.043
One-error ↓	0.323±0.067	0.282±0.070	0.268±0.059	0.375±0.067	0.270±0.078	0.256±0.056	0.276±0.068
Coverage ↓	1.915±0.182	1.839±0.214	1.819±0.189	2.064±0.230	1.829±0.207	1.791±0.185	1.826±0.145
Rank.Loss ↓	0.189±0.032	0.173±0.038	0.167±0.030	0.218±0.036	0.168±0.031	0.161±0.031	0.168±0.025
M.avg.AUC ↑	**0.815±0.026**	0.833±0.024	0.832±0.023	0.788±0.023	0.828±0.025	0.839±0.022	0.835±0.028
CAL500							
Z.-O.Loss ↓	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000
Ham.Loss ↓	0.240±0.010	0.239±0.010	0.239±0.010	0.239±0.012	0.238±0.011	0.238±0.009	0.243±0.011
Accuracy ↑	**0.226±0.008**	**0.228±0.010**	0.226±0.012	0.210±0.013	0.219±0.010	0.227±0.008	0.215±0.011
One-error ↓	0.121±0.043	0.118±0.037	0.121±0.035	0.118±0.039	0.121±0.036	0.118±0.037	0.118±0.039
Coverage ↓	**75.40±0.974**	75.50±1.241	75.88±1.273	76.35±0.732	75.48±1.003	**75.18±1.181**	77.58±0.880
Rank.Loss ↓	0.275±0.014	0.274±0.014	0.274±0.012	0.277±0.008	0.269±0.009	**0.269±0.011**	0.288±0.010
M.avg.AUC ↑	0.575±0.034	0.576±0.027	0.568±0.020	0.572±0.016	**0.594±0.019**	**0.590±0.019**	0.520±0.008
MEDIAMILL							
Z.-O.Loss ↓	**0.847±0.010**	**0.812±0.006**	**0.806±0.006**	0.890±0.004	0.854±0.005	0.831±0.006	0.731±0.009
Ham.Loss ↓	**0.190±0.004**	**0.177±0.002**	**0.174±0.002**	0.220±0.003	0.192±0.003	0.184±0.003	0.164±0.003
Accuracy ↑	**0.552±0.007**	**0.580±0.004**	**0.587±0.004**	0.477±0.005	0.548±0.004	0.565±0.004	0.616±0.005
One-error ↓	**0.128±0.008**	**0.111±0.003**	**0.109±0.004**	0.175±0.005	0.125±0.004	0.115±0.005	0.111±0.006
Coverage ↓	**3.619±0.042**	**3.467±0.039**	**3.432±0.037**	3.992±0.034	3.606±0.040	3.528±0.042	3.316±0.040
Rank.Loss ↓	**0.129±0.002**	**0.116±0.002**	**0.113±0.001**	0.163±0.002	0.128±0.002	0.121±0.002	0.107±0.002
M.avg.AUC ↑	**0.779±0.004**	**0.809±0.003**	**0.815±0.003**	0.722±0.004	0.787±0.003	0.799±0.003	0.823±0.004

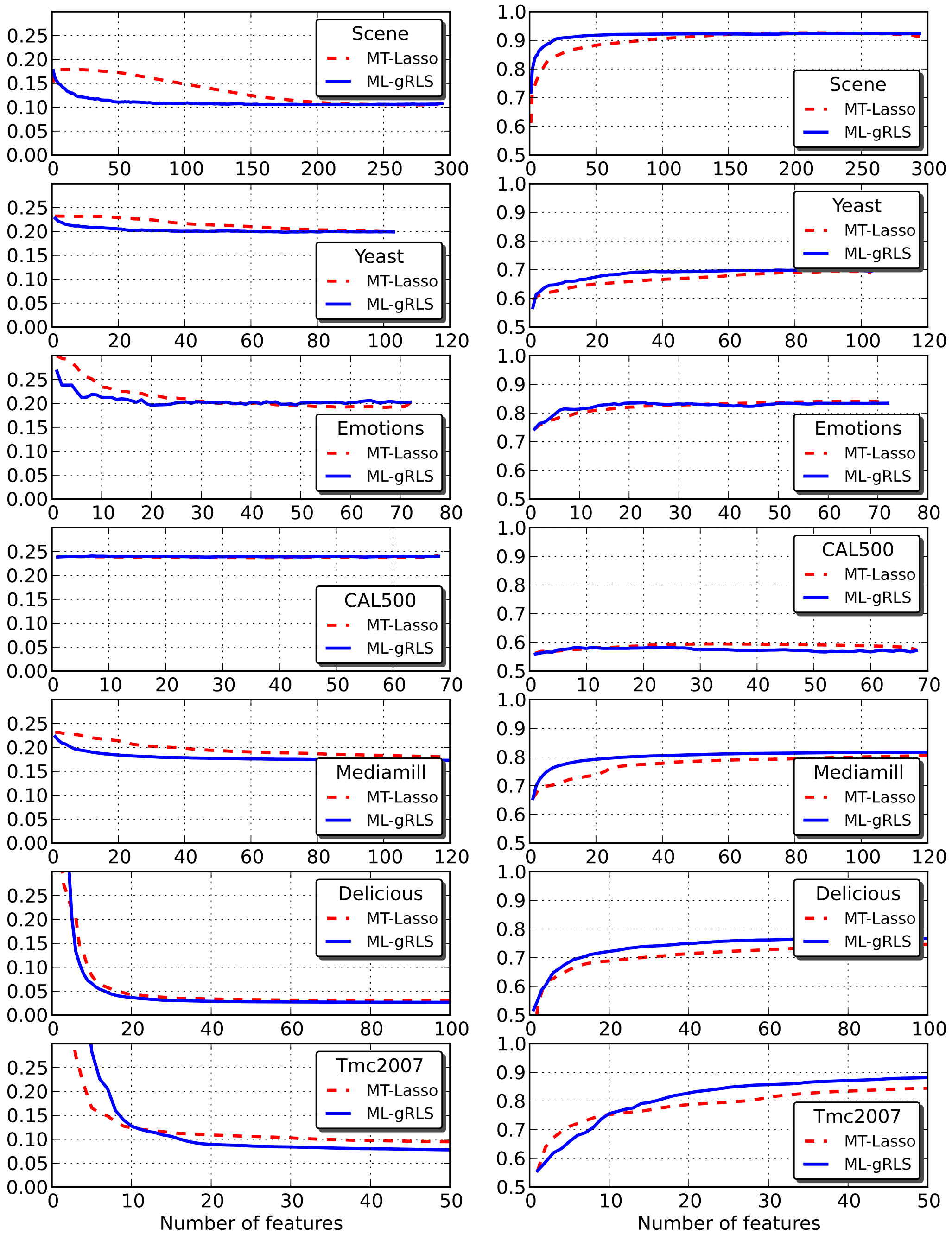

Figure 3: Performance curves (Hamming loss on the left and macro-averaged AUC on the right) on seven data sets.

Table 3: Performance on the two large data sets

	Delicious			*Tmc2007*		
	ML-gRLS	*MT-lasso*	*ML-kNN*	*ML-gRLS*	*MT-lasso*	*ML-kNN*
Z.-O.Loss ↓	1.000±0.000	1.000±0.000	0.998±0.001	**0.793±0.005**	0.869±0.004	0.820±0.005
Ham.Loss ↓	**0.027±0.000**	0.030±0.000	0.018±0.000	**0.078±0.001**	0.095±0.001	0.076±0.001
Accuracy ↑	**0.174±0.002**	0.114±0.001	0.105±0.002	**0.486±0.003**	0.389±0.003	0.430±0.007
One-error ↓	**0.355±0.005**	0.596±0.005	0.391±0.009	**0.288±0.004**	0.422±0.003	0.326±0.007
Coverage ↓	**533.3±5.144**	540.8±1.736	591.7±3.246	**3.526±0.048**	4.637±0.057	4.285±0.046
Rank.Loss ↓	**0.109±0.001**	0.130±0.001	0.128±0.002	**0.074±0.002**	0.115±0.002	0.098±0.001
M.avg.AUC ↑	**0.770±0.003**	0.747±0.002	0.641±0.003	**0.882±0.003**	0.845±0.002	0.778±0.004

tion approaches for single label learning problems have been extended to take account of the group structure, including the group lasso developed by Yuan and Lin (2006) and the grouped orthogonal matching pursuit Lozano et al. (2009), representing the lasso and greedy approaches, respectively. Extending our consideration of multi-label problems towards these concepts is a natural direction for future work.

Appendix A. Evaluation measures

Below, we use the notation $\mathcal{Y}^i = \{l \mid \mathbf{y}_l^i = 1\}$ and $\widehat{\mathcal{Y}}^i = \{l \mid f_l(\mathbf{x}^i) > 0\}$ to denote the sets of labels associated with and predicted for the ith instance, respectively. We also define a function

$$r_f(\mathbf{x}^i, l) = \left| \left\{ j \mid f_j(\mathbf{x}^i) \geq f_l(\mathbf{x}^i), 1 \leq j \leq t \right\} \right|$$

that ranks the labels according to their relevance to $\mathbf{x}$.

Given a test set $T = \{(\mathbf{x}^i, \mathcal{Y}^i) \mid 1 \leq i \leq v\}$, the evaluation metrics are defined as follows:

1. The *0/1 loss* measure

$$1 - \frac{1}{v} \sum_{i=1}^{v} \gamma(\widehat{\mathcal{Y}}^i, \mathcal{Y}^i),$$

where

$$\gamma(\widehat{\mathcal{Y}}^i, \mathcal{Y}^i) = \begin{cases} 1, & \text{if } \widehat{\mathcal{Y}}^i = \mathcal{Y}^i \\ 0, & \text{otherwise} \end{cases}$$

indicates the exact match of the predicted set of labels and actual set of labels.

2. The *Hamming loss* measure

$$\frac{1}{vt}\sum_{i=1}^{v}|\widehat{\mathcal{Y}}^i \Delta \mathcal{Y}^i|$$

evaluates the prediction error and missing error at the same time where the prediction error corresponds to a prediction of an incorrect label and the missing error corresponds to a missed prediction of an actual label. Let a notation $\widehat{\mathcal{Y}}^i \Delta \mathcal{Y}^i$ denote the symmetrical difference (the logical XOR) between the predicted set of labels $\widehat{\mathcal{Y}}^i$ and the actual set of labels $\mathcal{Y}^i$ associated with an instance $\mathbf{x}^i$.

3. The *multi-label accuracy* measure

$$\frac{1}{v}\sum_{i=1}^{v}\frac{|\mathcal{Y}^i \cap \widehat{\mathcal{Y}}^i|}{|\mathcal{Y}^i \cup \widehat{\mathcal{Y}}^i|}$$

is the mean ratio of the intersection and union of the actual and predicted label sets.

4. The *one-error* measure

$$\frac{1}{v}\sum_{i=1}^{v}\delta(\operatorname*{argmin}_{1\le l\le t} r_f(\mathbf{x}^i, l)),$$

where

$$\delta(l) = \begin{cases} 1, & \text{if } l \in \overline{\mathcal{Y}}^i \\ 0, & \text{otherwise} \end{cases}$$

indicates the frequency of the highest ranked label not being an actual label.

5. The *coverage* measure

$$\frac{1}{v}\sum_{i=1}^{v}\max_{l\in\mathcal{Y}^i} r_f(\mathbf{x}^i, l) - 1$$

indicates the distance, on the average, in the ranked list one has to go in order to cover all the actual labels $\mathcal{Y}^i$ assigned to an instance $\mathbf{x}^i$. Thus, the *coverage* extends top-ranked label evaluation used in the *one-error* to all the actual labels.

6. The *ranking loss* measure

$$\frac{1}{v}\sum_{i=1}^{v}\frac{1}{|\mathcal{Y}^i||\overline{\mathcal{Y}}^i|}|\{(l_1,l_2) \mid r_f(\mathbf{x}^i,l_1) \geq r_f(\mathbf{x}^i,l_2),(l_i,l_2) \in \mathcal{Y}^i \times \overline{\mathcal{Y}}^i\}|,$$

where $\overline{\mathcal{Y}}^i = \{1,\ldots,t\} \setminus \mathcal{Y}^i$ indicates how often the actual label $l_1 \in \mathcal{Y}^i$ receives lower or equal rank than the label $l_2 \notin \mathcal{Y}^i$.

7. The *macro-averaged AUC* measure

$$\frac{1}{t}\sum_{i=1}^{t} AUC_i$$

is defined as an averaged area under ROC curve (AUC) (Hanley and McNeil, 1982; Huang and Ling, 2005) over all the labels, where AUC is first calculated separately for each label. In the following, we denote by AUC_i the AUC computed for the ith label.

Appendix B. Pseudocode

Detailed pseudocode for multi-label greedy RLS is presented in Algorithm 2.

Proof of Theorem 1. We start by finding a solution for the multi-label problem with the quadratic regularizer for a fixed set of features $\mathcal{S}$:

$$\underset{\mathbf{W}\in\mathbb{R}^{|\mathcal{S}|\times t}}{\operatorname{argmin}} \left\{\|\mathbf{X}_{:,\mathcal{S}}\mathbf{W}-\mathbf{Y}\|_F^2 + \lambda\|\mathbf{W}\|_F^2\right\}$$

Using standard linear algebra and matrix inversion identities (see e.g. Henderson and Searle (1981)), a solution to the above problem can be expressed as

$$\mathbf{W} = (\mathbf{X}_{:,\mathcal{S}})^{\mathrm{T}}\mathbf{G}\mathbf{Y},$$

where

$$\mathbf{G} = (\mathbf{X}_{:,\mathcal{S}}(\mathbf{X}_{:,\mathcal{S}})^{\mathrm{T}} + \lambda\mathbf{I})^{-1}$$

and $\mathbf{I}$ is the identity matrix of size $n \times n$.

In order to perform feature selection computationally efficiently, the algorithm maintains, in addition to the set $\mathcal{S}$ of selected features, the following data structures in memory:

$$\begin{aligned}
\mathbf{A} &= \mathbf{G}\mathbf{Y}, \\
\mathbf{g} &= \mathrm{diag}(\mathbf{G}), \\
\mathbf{C} &= \mathbf{G}\mathbf{X},
\end{aligned}$$

where $\mathrm{diag}(\mathbf{G})$ denotes a vector that consists of the diagonal entries of $\mathbf{G}$. The greedy RLS algorithm starts from an empty set of selected features, and hence the values of $\mathbf{A}$, $\mathbf{g}$, and $\mathbf{C}$ are initialized to $\lambda^{-1}\mathbf{Y}$, $\lambda^{-1}\mathbf{1}$, and $\lambda^{-1}\mathbf{X}$, respectively, where $\mathbf{1} \in \mathbb{R}^n$ is a vector having every entry equal to 1. This initialization requires $O(nt + nd)$ time and memory.

The greedy RLS algorithm uses LOO-CV as a selection criterion, and we next recollect how this can be computed efficiently for RLS models. This computational short-cut is a multi-label modification of a classical result for RLS (see e.g. Elisseeff and Pontil (2003) and references therein). Provided that we have the matrix $\mathbf{A}$ and the vector $\mathbf{g}$ available, the squared LOO error for the jth training example and the hth task is

$$(\mathbf{g}_j)^{-2}(\mathbf{A}_{j,h})^2.$$

This involves only a constant number of standard floating point operations, and hence the average squared LOO error over the whole data set and all tasks can be computed in $O(nt)$ time.

Assume that we have computed the matrix $\mathbf{A}$ and the vector $\mathbf{g}$ corresponding to the current set of selected features $\mathcal{S}$. Then, to find out how much the LOO error would change if we would also select the ith feature, we have to update $\mathbf{A}$ and $\mathbf{g}$ so that they corresponding to the updated set $\mathcal{S} \cup \{i\}$.

$$\begin{aligned}
\widetilde{\mathbf{A}} &= (\mathbf{X}_{:,\mathcal{S}}(\mathbf{X}_{:,\mathcal{S}})^{\mathrm{T}} + \mathbf{X}_{:,i}(\mathbf{X}_{:,i})^{\mathrm{T}} + \lambda\mathbf{I})^{-1}\mathbf{Y}, \\
&= (\mathbf{G}^{-1} + \mathbf{X}_{:,i}(\mathbf{X}_{:,i})^{\mathrm{T}})^{-1}\mathbf{Y}, \\
&= (\mathbf{G} - \mathbf{u}(\mathbf{X}_{:,i})^{\mathrm{T}}\mathbf{G})\mathbf{Y} \\
&= \mathbf{A} - \mathbf{u}(\mathbf{X}_{:,i})^{\mathrm{T}}\mathbf{G}\mathbf{Y},
\end{aligned}$$

where the second last equation follows from the Woodbury matrix inversion formula (see e.g. Henderson and Searle (1981)) and

$$\mathbf{u} = \mathbf{C}_{:,i}(1 + (\mathbf{X}_{:,i})^{\mathrm{T}}\mathbf{C}_{:,i})^{-1}.$$

The vector g can be updated in an analogous way, that is, the jth entry of $\tilde{\mathbf{g}}$ is obtained from

$$\begin{aligned}\tilde{\mathbf{g}}_j &= (\mathbf{G} - \mathbf{u}(\mathbf{X}_{:,i})^{\mathrm{T}}\mathbf{G})_{j,j} \\ &= (\mathbf{G} - \mathbf{u}(\mathbf{C}_{:,i})^{\mathrm{T}})_{j,j} \\ &= \mathbf{g}_j - \mathbf{u}_j\mathbf{C}_{j,i},\end{aligned}$$

From these, we observe that the matrix $\mathbf{A}$ can be updated in $O(nt)$ time, the same which is spent for computing the LOO error for all tasks, and the vector g in $O(n)$ time. Thus, given that we have the above mentioned cache memories, the computation of LOO error for the updated feature sets is not more expensive than computing it for the current set. If the LOO computation is performed for every feature that has not yet been selected, the complexity of a single selection step is $O(ndt)$.

After the feature that decreases the LOO error the most is found, its index is added to the set of selected features and the cache memories have to be updated accordingly. The matrix $\mathbf{A}$ and the vector g can be updated similarly as in the LOO computation. The matrix $\mathbf{C}$ is updated again in an analogous way

$$\widetilde{\mathbf{C}} = \mathbf{C} - \mathbf{u}((\mathbf{X}_{:,i})^{\mathrm{T}}\mathbf{C}).$$

This update operation requires $O(nd)$ time but this is dominated by the time spent for searching the best feature.

The algorithm selects altogether k features and every time it performs the search for the best feature to be added. Thus, the overall time complexity of the whole selection process is $O(kndt)$. As a final step, the algorithm returns $\mathbf{W} = (\mathbf{X}_{:,\mathcal{S}})^{\mathrm{T}}\mathbf{A}$, whose computation requires $O(knt)$ time. The space complexity of the algorithm is dominated by the matrices $\mathbf{C}$ and $\mathbf{A}$ that require $O(nd)$ and $O(nt)$ space, respectively. □

Algorithm 2 Multi-label greedy RLS

```
1: $\mathbf{A} \leftarrow \lambda^{-1}\mathbf{Y}$
2: $\mathbf{g} \leftarrow \lambda^{-1}\mathbf{1}$
3: $\mathbf{C} \leftarrow \lambda^{-1}\mathbf{X}$
4: $\mathcal{S} \leftarrow \emptyset$
5: while $|\mathcal{S}| < k$ do
6:     $e \leftarrow \infty$
7:     $b \leftarrow 0$
8:     for $i \in \{1, \dots, d\} \setminus \mathcal{S}$ do
9:         $\mathbf{u} \leftarrow \mathbf{C}_{:,i}(1 + (\mathbf{X}_{:,i})^{\mathrm{T}}\mathbf{C}_{:,i})^{-1}$
10:        $e_i \leftarrow 0$
11:        $\widetilde{\mathbf{A}} \leftarrow \mathbf{A} - \mathbf{u}((\mathbf{X}_{:,i})^{\mathrm{T}}\mathbf{A})$
12:        for $h \in \{1, \dots, t\}$ do
13:            for $j \in \{1, \dots, n\}$ do
14:                $\tilde{\mathbf{g}}_j \leftarrow \mathbf{g}_j - \mathbf{u}_j\mathbf{C}_{j,i}$
15:                $e_i \leftarrow e_i + (\tilde{\mathbf{g}}_j)^{-2}(\widetilde{\mathbf{A}}_{j,h})^2$
16:        if $e_i < e$ then
17:            $e \leftarrow e_i$
18:            $b \leftarrow i$
19:    $\mathbf{u} \leftarrow \mathbf{C}_{:,b}(1 + (\mathbf{X}_{:,b})^{\mathrm{T}}\mathbf{C}_{:,b})^{-1}$
20:    $\mathbf{A} \leftarrow \mathbf{A} - \mathbf{u}((\mathbf{X}_{:,b})^{\mathrm{T}}\mathbf{A})$
21:    for $j \in \{1, \dots, n\}$ do
22:        $\mathbf{g}_j \leftarrow \mathbf{g}_j - \mathbf{u}_j\mathbf{C}_{j,b}$
23:    $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{u}((\mathbf{X}_{:,b})^{\mathrm{T}}\mathbf{C})$
24:    $\mathcal{S} \leftarrow \mathcal{S} \cup \{b\}$
25: $\mathbf{W} \leftarrow (\mathbf{X}_{:,\mathcal{S}})^{\mathrm{T}}\mathbf{A}$
```

Acknowledgements

This work has been supported by the Academy of Finland (grant 134020).

References

Cesa-Bianchi, N., Shalev-Shwartz, S., Shamir, O., 2011. Efficient learning with partially observed attributes. Journal of Machine Learning Research 12, 2857–2878.

Chen, S. S., Donoho, D. L., Saunders, M. A., 1998. Atomic decomposition by basis pursuit. SIAM Journal on Scientific Computing 20 (1), 33–61.

Clare, A., King, R. D., 2001. Knowledge discovery in multi-label phenotype data. In: Raedt, L. D., Siebes, A. (Eds.), PKDD. Vol. 2168 of Lecture Notes in Computer Science. Springer, pp. 42–53.

Doquire, G., Verleysen, M., 2011. Feature selection for multi-label classification problems. In: Cabestany, J., Rojas, I., Joya, G. (Eds.), Advances in Computational Intelligence: Proceedings of the 11th international work conference on Artificial neural networks, Part I. Vol. 6691 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, pp. 9–16.

Elisseeff, A., Pontil, M., 2003. Leave-one-out error and stability of learning algorithms with applications. In: Suykens, J., Horvath, G., Basu, S., Micchelli, C., Vandewalle, J. (Eds.), Advances in Learning Theory: Methods, Models and Applications. Vol. 190 of NATO Science Series III: Computer and Systems Sciences. IOS Press, Amsterdam, Netherlands, Ch. 6, pp. 111–130.

Friedman, J. H., Hastie, T., Tibshirani, R., 2 2010. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33 (1), 1–22.

Hanley, J. A., McNeil, B. J., 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143 (1), 29–36.

Hazan, E., Koren, T., July 2012. Linear regression with limited observation. In: Langford, J., Pineau, J. (Eds.), Proceedings of the 29th International Conference on Machine Learning (ICML '12). ICML '12. icml.cc / Omnipress, pp. 807–814.

Henderson, H. V., Searle, S. R., 1981. On deriving the inverse of a sum of matrices. SIAM Review 23 (1), 53–60.

Hoerl, A. E., Kennard, R. W., 1970. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics 12, 55–67.

Huang, J., Ling, C. X., 2005. Using AUC and accuracy in evaluating learning algorithms. IEEE Transactions on Knowledge and Data Engineering 17 (3), 299–310.

Hüllermeier, E., Fürnkranz, J., Cheng, W., Brinker, K., 2008. Label ranking by learning pairwise preferences. Artificial Intelligence 172 (16–17), 1897 – 1916.

Kohavi, R., John, G. H., 1997. Wrappers for feature subset selection. Artificial Intelligence 97, 273–324.

Lachenbruch, P. A., 1967. An almost unbiased method of obtaining confidence intervals for the probability of misclassification in discriminant analysis. Biometrics 23 (4), 639–645.

Liu, H., Palatucci, M., Zhang, J., 2009. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In: Proceedings of the 26th Annual International Conference on Machine Learning. ACM, New York, NY, USA, pp. 649–656.

Lozano, A., Swirszcz, G., Abe, N., 2009. Grouped orthogonal matching pursuit for variable selection and prediction. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., Culotta, A. (Eds.), Advances in Neural Information Processing Systems 22. pp. 1150–1158.

Madjarov, G., Kocev, D., Gjorgjevikj, D., Deroski, S., September 2012. An extensive experimental comparison of methods for multi-label learning. Pattern Recognition 45 (9), 3084–3104.

Naula, P., Pahikkala, T., Airola, A., Salakoski, T., December 2011a. Greedy regularized least-squares for multi-task learning. In: Spiliopoulou, M., Wang, H., Cook, D., Pei, J., Wang, W., Zaïane, O., Wu, X. (Eds.), 11th IEEE International Conference on Data Mining Workshops (ICDMW'11). IEEE Computer Society, pp. 527–533.

Naula, P., Pahikkala, T., Airola, A., Salakoski, T., May 2011b. Learning multi-label predictors under sparsity budget. In: Kofod-Petersen, A., Heintz, F., Langseth, H. (Eds.), Eleventh Scandinavian Conference on Artificial Intelligence, SCAI 2011. Vol. 227 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 30–39.

Obozinski, G., Taskar, B., Jordan, M. I., 2010. Joint covariate selection and joint subspace selection for multiple classification problems. Statistics and Computing 20, 231–252.

Pahikkala, T., Airola, A., Salakoski, T., 2010. Speeding up greedy forward selection for regularized least-squares. In: Draghici, S., Khoshgoftaar, T. M., Palade, V., Pedrycz, W., Wani, M. A., Zhu, X. (Eds.), Proceedings of The Ninth International Conference on Machine Learning and Applications (ICMLA 2010). IEEE, pp. 325–330.

Pahikkala, T., Okser, S., Airola, A., Salakoski, T., Aittokallio, T., 2012. Wrapper-based selection of genetic features in genome-wide association studies through fast matrix operations. Algorithms for Molecular Biology 7 (1), 11.

Suykens, J., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J., 2002. Least Squares Support Vector Machines. World Scientific Pub. Co., Singapore.

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

Tsoumakas, G., Katakis, I., Vlahavas, I., 2010. Mining multi-label data. In: Maimon, O., Rokach, L. (Eds.), Data Mining and Knowledge Discovery Handbook. Springer US, pp. 667–685.

Tsoumakas, G., Katakis, I., Vlahavas, I., 2011a. Random k-labelsets for multi-label classification. IEEE Transactions on Knowledge and Data Engineering 23 (7), 1079–1089.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., Vlahavas, I., 2011b. Mulan: A java library for multi-label learning. Journal of Machine Learning Research 12, 2411–2414.

Turlach, B. A., Venables, W. N., Wright, S. J., 2005. Simultaneous variable selection. Technometrics 47 (3), 349–363.

Xu, Z., Weinberger, K. Q., Chapelle, O., July 2012. The greedy miser: Learning under test-time budgets. In: Langford, J., Pineau, J. (Eds.), Proceedings of the 29th International Conference on Machine Learning (ICML-12). ICML '12. Omnipress, pp. 1175–1182.

Yuan, M., Lin, Y., 2006. Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68 (1), 49–67.

Zhang, M.-L., na José M., P., Robles, V., 2009. Feature selection for multi-label naive bayes classification. Information Sciences 179 (19), 3218–3229.

Zhang, M.-L., Zhou, Z.-H., July 2007. Ml-knn: A lazy learning approach to multi-label learning. Pattern Recognition 40 (7), 2038–2048.

Zhang, T., 2011. Adaptive forward-backward greedy algorithm for learning sparse representations. IEEE Transactions on Information Theory 57, 4689–4708.

Zhang, Y., Yeung, D.-Y., Xu, Q., 2010. Probabilistic multi-task feature selection. In: Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., Culotta, A. (Eds.), Advances in Neural Information Processing Systems 23. MIT Press, pp. 2559–2567.

Zhao, P., Yu, B., 2006. On model selection consistency of lasso. Journal of Machine Learning Research 7, 2541–2563.