# Learning Multi-Label Predictors under Sparsity Budget

Pekka NAULA [a],Tapio PAHIKKALA [a,b], Antti AIROLA [a,b]
and Tapio SALAKOSKI [a,b]

[a] *Department of Information Technology, University of Turku, Finland*
[b] *Turku Centre for Computer Science (TUCS), Finland*

**Abstract.** In real-life machine learning applications, there are often costs associated with the features needed in prediction. This is the case for example when deploying learned models in mass produced products, where the manufacturing costs or space limitations may restrict the number of feature extracting sensors that can be included in each device. In such situations, the training process involves a sparsity budget restricting the number of features the learned predictor can use. In this paper, we consider the problem of learning multi-label predictors under a sparsity budget. For this purpose, we consider three different wrapper-based greedy forward selection approaches for constructing sparse multi-label learning models. In our experiments, we show that the method selecting a common set of features shared by multiple tasks by greedily maximizing the prediction performance averaged over all the tasks provides a better prediction performance than the approaches selecting the features separately for each task.

**Keywords.** Feature subset selection, multi-label learning, budgeted learning, regularized least-squares

## 1. Introduction

Given a set of training examples with associated labels, a machine learning algorithm builds a mathematical model that predicts the label(s) for a new unseen example. Examples are high-dimensional data inputs that are often considered to be acquired with no costs. However, in real-life, there are always costs associated with producing or measuring high-dimensional data. Many such different cost scenarios have been studied in the machine learning literature (see [1] for overview). For example in active feature acquisition [2] or budgeted learning [4], the learning algorithm is assumed to operate during training on a limited budget, and must pay for the acquisition of the features of the training examples. In this study we assume that we have an unlimited budget during training, but that there is a cost associated with each chosen feature when the learned models are deployed. For example, in industrial machine learning applications, the costs might be due to the manufacturing costs of physical sensors which are used to measure the feature values of new examples. Alternatively, in medical decision making the features might correspond to the outcomes of medical tests, and costs to the monetary price of administering the tests.

As a motivating example, let us consider a smart handheld device which is able to solve several tasks simultaneously, based on data obtained from an array of sensors integrated to the device. For example, the purpose of the device could be to predict a predefined set of health related properties of the user, such as current stress level, calorie consumption, whether the user is fit to drive etc. There are several candidate data suppliers such as sensors (accelerometer, barometer, GPS, temperature sensor, heart rate sensor etc.) or questionnaire data (sex, age, weight, etc.) embedded into the prototype devices, which are used to collect the training data. The tasks are similar in nature, suggesting that some of the sensors could be used in solving several of the tasks. All of the possible software and hardware components come at a cost, which might be money or space in a circuit board, and we have restricted budget available. How should the data suppliers be selected for the final devices to allow constructing a predictor with acceptable level of accuracy for all the tasks, given a restricted budget?

In this paper, we consider approaches that combine the ideas of multi-label learning and wrapper-based feature subset selection to solve multiple learning tasks with a limited budget. In multi-label learning [5], each data instance contains a feature representation and multiple labels, that describe the properties we wish to learn to predict from the features. For example, in text classification the features can encode word frequencies, and the labels different topics that a document can have. Multi-label learning is an instance of the more general setting of multi-task learning [6]. Feature subset selection [7] is a standard approach to reduce the dimensionality of the feature space and to achieve a useful set of features. This can allow reduced costs in deploying the learned models, and may in some cases also lead to improved accuracy for the model.

Feature selection for multi-label classification has been previously addressed, for example, with filter methods (see e.g. [8,9]). In the filter approach, the selection is done as a pre-processing step before learning, by computing univariate statistics such as information gain, mutual information or $\chi^2$ on feature-by-feature basis. The main advantages of the approach are efficiency and ease of implementation and the main disadvantages the inability to take account of the dependencies between the features, or the properties of the learning algorithm which is subsequently trained on the features. The problem of multi-label feature selection has also been addressed through Bayesian [10,**?**] or regularization-based [11] learning algorithms that encourage sparsity in the learned solutions.

Wrapper methods [12] have been proposed as a way to overcome the limitations of the filter approach. In the wrapper model, features are selected through interaction with a learner training method. Here, the power set of features is searched over. A new predictor is trained during each search step using the corresponding feature subset and an estimate of its predictive performance is used to measure the quality of the subset. The number of possible feature sets grows exponentially with the number of available features. Therefore, the wrapper type of feature selection methods require a search heuristic when traversing through the power set. The most commonly used search heuristic is greedy forward selection that adds one feature at a time to the set of selected features, but never removes any features from the set. In addition to the search strategy, the wrapper methods require a method for assessing how good the feature subsets under consideration are. Measuring the prediction performance on the training set is known to be unreliable. Therefore, [13] proposed to measure the goodness with leave-one-out (LOO) cross-validation. In LOO, each example in turn is left out of the training set and

used for testing. The wrapper approach has been empirically shown to outperform filter based selection (see e.g. [14]), but in straightforward implementations this improvement comes with a considerable computational cost, especially when used with LOO selection criterion.

In [15], we proposed greedy RLS, a wrapper-based feature selection method for regularized least-squares, which employs a greedy search heuristic and a LOO selection criterion. The computational complexity of greedy RLS was shown to be $O(kmn)$, where $m$ is the number of data points in the training set, $n$ is the overall number of features among which the selection is made, and $k$ is the number of features selected by greedy RLS. That is, the training time is linear with respect to $k$, $m$, and $n$, despite the fact that the learned predictors and selected features are exactly equivalent with those that would be obtained if RLS would be used as a black-box method, around which the incremental feature selection and LOO criterion are wrapped.

This paper is organized as follows. In Section 2 we briefly review formally the standard framework for regularized least-squares for single-label learning, and present a high level description of the greedy RLS feature selection method. Then, we extend this method to multi-label learning, where multiple tasks are learned simultaneously. Next, in Section 3, we present experiments on three real-world data sets, comparing our proposed multi-label method to a baseline method where tasks are learned in isolation. Finally, in Section 4, we summarize our findings, and suggest future research possibilities.

## 2. Methods

Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ be a matrix containing the feature representation of the examples in the training set, where $n$ is the number of features and $m$ is the number of training examples. The $i, j$th entry of $\mathbf{X}$ contains the value of the $j$th feature in the $i$th training example. Moreover, let $\mathbf{y}^j \in \mathbb{R}^m$ for $j = 1, \ldots, t$ be vectors consisting of the labels of the training examples for $t$ tasks. In multi-class or multi-label classification, the labels can be restricted to be either $-1$ or $1$ depending whether the data points belong to the class, while they can be any real numbers in multi-label regression tasks.

For each of the $t$ tasks, we construct a linear prediction function. The function for the $j$th task can be expressed as

$$f^j(\mathbf{x}) = (\mathbf{x}_{\mathcal{S}^j})^{\mathrm{T}} \mathbf{w}^j,$$

where $\mathbf{w}^j \in \mathbb{R}^{|\mathcal{S}^j|}$ is a vector representation of the learned predictor for the $j$th task, $\mathbf{x}$ is a data point for which the prediction of the $j$th label is to be made, and $\mathbf{x}_{\mathcal{S}^j}$ a vector representation of $\mathbf{x}$ that only contains the features indexed by the set $\mathcal{S}^j$.

We assume that the sets $\mathcal{S}^j$ contain feature indices selected by a training algorithm that performs feature selection while constructing the predictors. Moreover, we assume that the number of features we can extract from a data point at prediction time is constrained by a a given sparsity budget, that is, we have

$$\left| \bigcup_{j=1}^{t} \mathcal{S}^j \right| \leq k \tag{1}$$

for some $k \in \mathbb{N}$. Note that since the sparsity budget constraint is given for the number of features, it is assumed that all features have an equal extraction cost at the prediction time. This can be easily generalized to a setting in which each feature would have a different extraction cost. However, this is not considered further in this paper, because of limited space and the lack of available data for practical experiments.

The most straightforward approach for constructing the predictors under a common budget constraint is to simply train them separately and ensure that each predictor uses only $k/t$ features. For doing this, we can use numerous off-the-shelf feature selection methods available for single-task learning. It may of course happen that the size of the union feature set in (1) is strictly smaller than $k$, if the same features are selected for several tasks. In such cases the selection process can be continued until the sparsity budget is completely fulfilled. An alternative approach is to select the features jointly for all tasks, favoring such features that increase the performance averaged over all tasks.

Before considering the actual training algorithms, we present some of the building blocks. As a base learner, we use regularized least-squares (RLS) [16], also known as the least-squares support vector machine [17] and ridge regression [18], which is a state-of-the art machine learning method suitable for several types of machine learning tasks. RLS can be expressed for a single task with training data $\mathbf{X} \in \mathbb{R}^{m \times n}, \mathbf{y} \in \mathbb{R}^m$ as the following problem:

$$\mathcal{A}(\mathbf{X}, \mathbf{y}) = \underset{\mathbf{w} \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ (\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathrm{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathrm{T}} \mathbf{w} \right\}, \tag{2}$$

where $\lambda > 0$ is a regularization parameter. The first term in (2), called the empirical risk, measures how well the prediction function fits to the training data. The second term is called the regularizer and it controls the tradeoff between the empirical error on the training set and the complexity of the prediction function.

As a selection criterion, we use the mean squared error obtained via leave-one-out (LOO) cross-validation. Given a training data $\mathbf{X} \in \mathbb{R}^{m \times n}, \mathbf{y} \in \mathbb{R}^m$, this can be expressed as follows:

$$\mathcal{L}(\mathbf{X}, \mathbf{y}) = \sum_{h=1}^{m} (f_{\overline{h}}(\mathbf{X}_h) - \mathbf{y}_h)^2, \tag{3}$$

where $f_{\overline{h}}$ is a predictor which is trained using the whole training set except the $h$th training example, and $\mathbf{X}_h$ and $\mathbf{y}_h$ contain, respectively, the features and the label of the $h$th example.

As a training algorithm for learning predictors with a restricted sparsity budget, we present a greedy forward feature selection method for RLS with LOO criterion. 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 features from the set. A high level pseudo code of the algorithm for a single task is given in Algorithm 1. In the pseudo code, the function call $\mathcal{L}(\mathbf{X}^{\mathcal{S} \cup \{i\}}, \mathbf{y})$ returns a real value, which is equivalent to the LOO performance of a RLS predictor trained with a feature set $\mathcal{S} \cup \{i\}$. The feature added in each iteration of the outer loop is the one providing maximal decrease of LOO classification error. To select the features separately for all of the $t$ tasks with a common sparsity budget $k$, we simply call Algorithm 1 with a constraint $k/t$ consecutively for each task.

**Algorithm 1** Pseudo code for selecting $k$ features for a single task.

$\mathcal{S} \leftarrow \emptyset$ ▷ The current set of selected features.
**while** $|\mathcal{S}| < k$ **do** ▷ Select $k$ features.
  $e \leftarrow \infty$
  $b \leftarrow 0$
  **for** $i \in \{1, \ldots, n\} \setminus \mathcal{S}$ **do** ▷ Test all features before selecting.
    $e_i \leftarrow \mathcal{L}(\mathbf{X}^{\mathcal{S} \cup \{i\}}, \mathbf{y})$ ▷ Compute LOO performance.
    **if** $e_i < e$ **then**
      $e \leftarrow e_i$
      $b \leftarrow i$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{b\}$ ▷ Select the feature that increases the performance the most.
$\mathbf{w} \leftarrow \mathcal{A}(\mathbf{X}^{\mathcal{S}}, \mathbf{y})$
**return** $\mathbf{w}, \mathcal{S}$

**Algorithm 2** Pseudo code for selecting $k$ features common for all tasks.

$\mathcal{S} \leftarrow \emptyset$ ▷ The current set of selected features common for all tasks.
**while** $|\mathcal{S}| < k$ **do** ▷ Select $k$ common features.
  $e \leftarrow \infty$
  $b \leftarrow 0$
  **for** $i \in \{1, \ldots, n\} \setminus \mathcal{S}$ **do** ▷ Test all features before selecting.
    $e_{avg} \leftarrow 0$
    **for** $j \in \{1, \ldots, t\}$ **do**
      $e_{i,j} \leftarrow \mathcal{L}(\mathbf{X}^{\mathcal{S} \cup \{i\}}, \mathbf{y}^j)$ ▷ Compute LOO performance for task $j$.
      $e_{avg} \leftarrow e_{avg} + e_{i,j}/t$
    **if** $e_{avg} < e$ **then**
      $e \leftarrow e_{avg}$
      $b \leftarrow i$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{b\}$ ▷ Select the feature that increases the average performance the most.
**for** $j \in \{1, \ldots, t\}$ **do**
  $\mathbf{w}^j \leftarrow \mathcal{A}(\mathbf{X}^{\mathcal{S}}, \mathbf{y}^j)$
**return** $\mathbf{w}^1, \ldots, \mathbf{w}^t, \mathcal{S}$

Next, we consider an algorithm that selects a common set of features for the tasks using the performance averaged over all tasks as a selection criterion. A high level pseudo code of this algorithm is presented in Algorithm 2. 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. For the $i$th 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.

In [15], we proposed an implementation of Algorithm 1, which we named greedy RLS, whose computational complexity is $O(kmn)$, where $m$ is the number of data points in the training set, $n$ is the overall number of features among which the selection is made, and $k$ is the number of features selected by greedy RLS. The modification of greedy RLS for multiple labels per data point is quite straightforward, since only the selection criterion has to be modified. The technical description of the modified algorithm is left out from this paper due to the limited space. By extending the computational complexity analysis presented in [15], it can be shown that the time complexity of the multi-task greedy RLS is $O(kmnt)$.

**Table 1.** Datasets.

| Data sets | features | labels | train | test |
|---|---|---|---|---|
| scene | 294 | 6 | 1211 | 1196 |
| yeast | 103 | 14 | 1500 | 917 |
| siam | 30438 | 22 | 21519 | 7077 |

## 3. Experiments

### *3.1. Datasets and setup*

In the experiments we compare three methods for budgeted multi-label learning. We assume that all the features have equal cost and we are given a budget constraint $k$ on the number of features, and the number of labels is $t$. Method $1$ learns the $t$ tasks separately, using Algorithm 1, so that the individual training processes do not share any knowledge between each other. The budget is divided evenly between the tasks, meaning that each individual predictor uses $k/t$ features. Method $1$ can be considered wasteful in the sense that none of the tasks benefit from the features selected for the other tasks. Method $2$ aims to fix this problem as follows. All the features are again selected as in Method $1$, but after the selection is finished, the predictors for each task are re-trained using the union of all the selected feature sets. Finally, Method $3$ (Algorithm 2) aims to improve on Method $2$ by performing the selection process itself jointly over all the tasks, in each step selecting such features that give the best average performance over all the tasks. The methods are compared over a varying range of budgets. In the comparisons, the budget size is always computed as the size of the final set of selected features. Thus, if some of the features selected by Method $1$ are shared by multiple tasks, each such feature incurs only unit cost in the budget. This means that the comparison between the three methods are always made so that they all use exactly the same number of features.

We perform our experiments on three publicly available multi-label data sets [1]. The data sets are selected in order to cover different application domains, in our case image, biology and text. The first data set, *scene* (see [19]), is a scene classification problem where images are categorized into semantic classes such as beaches, sunsets or mountains. The second data set, *yeast* (see [20]), is a biological data set about gene function classification. Finally, we have one large text classification data set, *siam* (see [21]). We use the standard training/test splits provided on the web page. The characteristics of the data sets are summarized in Table 1.

All the test runs are carried out using the greedy RLS implementation in RLScore[2], 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. The regularization parameter $\lambda$ is set to 1 in this study. In preliminary experiments we also tested a large range of other values, the observed trends were similar as those presented in the paper. We use all the available tasks with scene data and yeast data, but only the first 9 out of the 22 possible tasks on siam data, because of large processing time.

To evaluate the overall behavior of both of the algorithms, we calculate the average AUC (the area under the ROC curve) [22,23] over all the tasks. The AUC is a popular

[1] available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`
[2] available at `http://www.tucs.fi/RLScore`

ranking based performance measure for binary classification. It is invariant to relative class distributions and class specific error costs, allowing us to assign equal importance to each label regardless of how commonly they appear in the data.

### *3.2. Results*

On the left sides of the Figures 1, 2 and 3, are the performance curves for scene data, yeast data and siam data, respectively. In each figure we plot the performance curves for the three considered methods. The average AUC is plotted as a function of the available budget. Method 1 shows on all of the data sets much worse performance than either Method 2 or 3. Thus, in our experiments sharing features between the tasks proves to be always beneficial. For the scene (Figure 1), and for the yeast (Figure 2) data, Method 3 outperforms Method 2. Somewhat surprisingly, on the SIAM data (see Figure 3), Method 2 outperforms Method 3. Further examination of this issue revealed that on siam data Method 2 had higher AUC than Method 3 even on the training set. This suggests that the mean-squared error based selection criterion (equation (3)) used in the experiments fails to choose the best features in terms of AUC for Algorithm 2. Earlier results in the literature (see e.g. [24]), have shown that for imbalanced data sets optimizing mean-squared error can yield sub-optimal performance in terms of AUC. The AUC measure is slower to compute than the mean-squared error ($O(m \log(m))$ instead of $O(m)$), and hence we used the latter as a selection criterion in order to scale the methods to the large data set sizes considered in the experiments.

On the right sides of the Figures 1, 2 and 3, we plot the cost curves measuring the overlap between the feature sets selected for each task by Method 1. *Realized cost* represents the size of the union of feature sets (see Eq. (1)) over all of the tasks as a function of the given budget constraint. *Max cost* represents the upper bound on the realized cost for a given budget constraint. This situation occurs, if the feature sets selected for all of the tasks are mutually exclusive. *Min cost* represents the lower bound on the realized cost. This situation occurs, if the feature sets selected for all of the tasks are exactly the same. The realized cost determines the size of the budget in the comparisons. For example, with a budget constraint 84 on the scene data, the Method 1 selected 74 distinct features (see Figure 1), and hence it is compared to the other methods under budget 74.

Table 2 shows the classification performance (AUC) for Methods 1, 2 and 3, for each of the tasks on scene data over some of the selected budgets. The numbers in brackets for Method 1 results (top), indicate the size of the feature sets per each task on selected budgets (note that the sum of the sizes of the feature sets over all tasks on budgets 53 and 74 for the other methods are 54 and 84 due to the redundant features). Among some tasks, such as 3, 4 and 5 for Method 3 (bottom), the performance does not improve after a small number of selected features (53 out of 294). This may indicate that some of the common selected features are not representative for those tasks. One might think that enforcing a common feature subset might be harmful for some specific task due to irrelevant or redundant features, but this is not observed to occur in our test runs. Methods 2 and 3 give as good as or better prediction performances than to Method 1 for all of the tasks, under all given budgets. On smallest budget sizes Method 3 outperforms Method 2 on all of the tasks, for the largest budgets the differences disappear.

**Table 2.** Performance for tasks on scene data with Method 1 (top), Method 2 (middle), and Method 3 (bottom).

| Budget | avg | task 1 | task 2 | task 3 | task 4 | task 5 | task 6 |
|---|---|---|---|---|---|---|---|
| 6 (1) | 0.796 | 0.789 | 0.898 | 0.820 | 0.880 | 0.662 | 0.725 |
| 18 (3) | 0.859 | 0.862 | 0.939 | 0.865 | 0.937 | 0.740 | 0.810 |
| 53 (9) | 0.896 | 0.908 | 0.962 | 0.902 | 0.950 | 0.794 | 0.858 |
| 74 (14) | 0.899 | 0.918 | 0.973 | 0.920 | 0.952 | 0.799 | 0.831 |

| Budget | avg | task 1 | task 2 | task 3 | task 4 | task 5 | task 6 |
|---|---|---|---|---|---|---|---|
| 6 | 0.834 | 0.854 | 0.914 | 0.827 | 0.904 | 0.715 | 0.790 |
| 18 | 0.878 | 0.903 | 0.954 | 0.870 | 0.951 | 0.756 | 0.833 |
| 53 | 0.906 | 0.921 | 0.977 | 0.922 | 0.953 | 0.800 | 0.861 |
| 74 | 0.905 | 0.923 | 0.978 | 0.927 | 0.954 | 0.803 | 0.846 |

| Budget | avg | task 1 | task 2 | task 3 | task 4 | task 5 | task 6 |
|---|---|---|---|---|---|---|---|
| 6 | 0.852 | 0.880 | 0.942 | 0.835 | 0.935 | 0.744 | 0.779 |
| 18 | 0.890 | 0.906 | 0.968 | 0.898 | 0.958 | 0.777 | 0.832 |
| 53 | 0.906 | 0.915 | 0.974 | 0.922 | 0.957 | 0.804 | 0.862 |
| 74 | 0.907 | 0.922 | 0.977 | 0.920 | 0.954 | 0.803 | 0.868 |

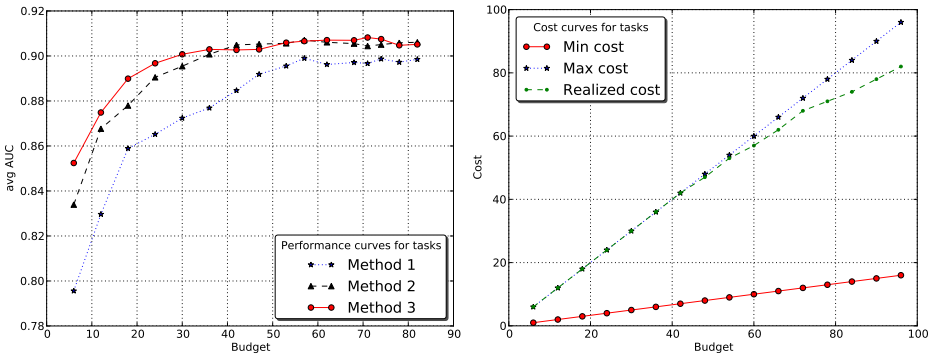


**Figure 1.** Performance and cost curves on SCENE data.

## 4. Conclusions and Future Work

In this paper, we propose a method that combines the ideas of multi-label learning and feature subset selection under a sparsity budget by selecting a minimal set of common features simultaneously for several tasks. The goal of the method is to achieve the best possible predictive performance for multiple tasks under a restricted budget where the cost is associated with high-dimensional data. The method itself can be integrated into any supervised learning algorithm, but we select RLS because of its efficiency in performance due to mathematical short-cuts used in matrix calculations. We tested the method using real data sets from three different domains. The experimental results show that the proposed multi-label method leads to improved prediction performance under a given budget. One can see the benefits of our method from two different perspective. Given equal budgets for our method and the baseline method, the proposed approach provides better performance than the baseline (quality perspective). Given equal performance requirements, the proposed method has smaller costs than the baseline (cost perspective).

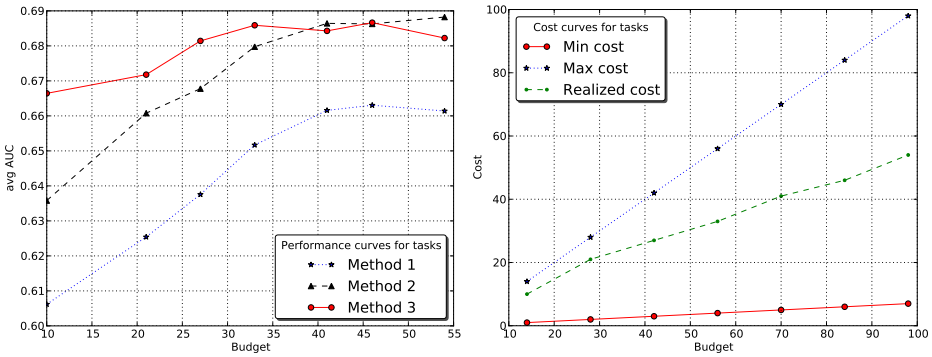


**Figure 2.** Performance and cost curves on YEAST data.

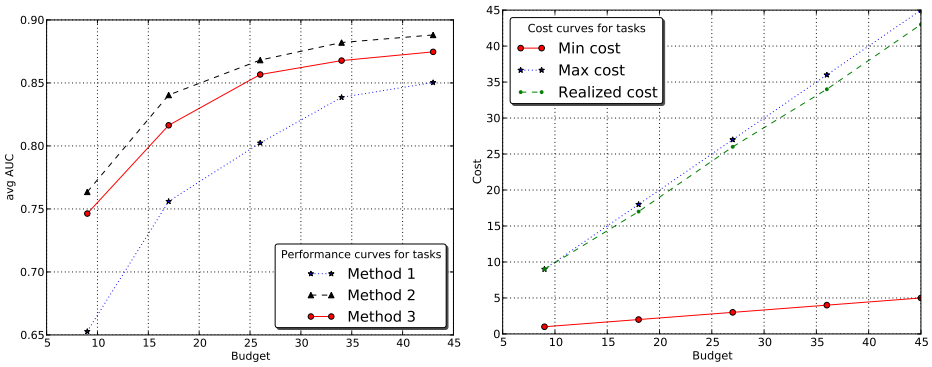


**Figure 3.** Performance and cost curves on SIAM data.

In our future work we plan to extend the introduced approach to more general settings where for example the features may be associated with variable costs. Other search strategies than the greedy forward selection might also be worth while to study. For example, genetic algorithms [25] have been shown to be effective in multi-criteria optimization problems.

## Acknowledgements

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

## References

[1] P. D. Turney. Types of cost in inductive concept learning. In T. Dietterich, D. Margineantu, F. Provost, and P. D. Turney, editors, *Proceedings of the ICML 2000 Workshop on Cost-Sensitive Learning*, 2000.

[2] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. Active feature-value acquisition for classifier induction. In R. Rastogi, K. Morik, M. Bramer, and X. Wu, editors, *Proceedings of the 4th IEEE International Conference on Data Mining*, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[3] D. J. Lizotte, O. Madani, and R. Greiner. Budgeted learning of naïve-Bayes classifiers. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI'03)*, pages 378–385, 2003.

[4] A. Kapoor and R. Greiner. Learning and classifying under hard budgets. In J. Gama, R. Camacho, P. Brazdil, A. Jorge, and L. Torgo, editors, *Proceedings of the 16th European Conference on Machine Learning (ECML'05)*, volume 3720 of *Lecture Notes in Computer Science*. Springer, 2005.

[5] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer US, 2010.

[6] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[7] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

[8] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997)*, pages 412–420. Morgan Kaufmann, 1997.

[9] W. Chen, J. Yan, B. Zhang, Z. Chen, and Q. Yang. Document transformation for multi-label feature selection in text categorization. In N. Ramakrishnan, O.R. Zaïane, Y. Shi, C.W. Clifton, and X. Wu, editors, *Proceedings of the 7th IEEE International Conference on Data Mining*, pages 451–456, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[10] P. J. Brown, M. Vannucci, and T. Fearn. Multivariate Bayesian variable selection and prediction. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60:627–641, 1998.

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

[12] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

[13] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In William W. Cohen and Haym Hirsch, editors, *Proceedings of the 11th International Conference on Machine Learning*, pages 121–129, San Fransisco, CA, 1994. Morgan Kaufmann Publishers.

[14] I. Inza, P. Larrañaga, R. Blanco, and A. J. Cerrolaza. Filter versus wrapper gene selection approaches in DNA microarray domains. *Artificial Intelligence in Medicine*, 31(2):91–103, 2004.

[15] T. Pahikkala, A. Airola, and T. Salakoski. Speeding up greedy forward selection for regularized least-squares. In Sorin Draghici, Taghi M. Khoshgoftaar, Vasile Palade, Witold Pedrycz, M. Arif Wani, and Xingquan Zhu, editors, *Proceedings of The Ninth International Conference on Machine Learning and Applications (ICMLA'10)*, pages 325–330. IEEE, 2010.

[16] R. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2002.

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

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

[19] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[20] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in neural information processing systems 14*. MIT Press, Cambridge, MA, 2002.

[21] A. Srivastava and B. Zane-Ulman. Discovering recurring anomalies in text reports regarding complex space systems. In *2005 IEEE Aerospace conference*, pages 3853–3862. IEEE, 2005.

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

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

[24] T. Pahikkala, A. Airola, H. Suominen, J. Boberg, and T. Salakoski. Efficient AUC maximization with regularized least-squares. In A. Holst, P. Kreuger, and P. Funk, editors, *Proceedings of the 10th Scandinavian Conference on Artificial Intelligence (SCAI 2008)*, volume 173 of *Frontiers in Artificial Intelligence and Applications*, pages 12–19. IOS Press, Amsterdam, Netherlands, 2008.

[25] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems and their Applications*, 13(2):44–49, 1998.