Relational Lasso
—An Improved Method Using the Relations among Features—

Kotaro Kitagawa **Kumiko Tanaka-Ishii**
Graduate School of Information Science and Technology,
The University of Tokyo
kitagawa@cl.ci.i.u-tokyo.ac.jp kumiko@i.u-tokyo.ac.jp

Abstract

Relational lasso is a method that incorporates feature relations within machine learning. By using automatically obtained noisy relations among features, relational lasso learns an additional penalty parameter per feature, which is then incorporated in terms of a regularizer within the target optimization function.

Relational lasso has been tested on three different tasks: text categorization, polarity estimation, and parsing, where it was compared with conventional lasso and adaptive lasso (Zou, 2006) when using a multi-class logistic regression optimization method. Relational lasso outperformed these other lasso methods in the tests.

1 Introduction

As machine learning methods scale up and now deal with millions of features, we ideally want to add all possible features without having to manually verify or consider the effectiveness of each feature with respect to the performance. In other words, we need an automatic way to exploit any usable information that can be obtained from features. However, with current machine learning methods, adding noisy features could lower performance, so the user still has to decide which features are worth adding.

Regularization methods have recently received greater interest because of this need. Regularization is expressed as a constraint term within an optimization function, where the term is given as a function regarding the importance weight of each feature. Regularization provides a means of importance control embedded within the target optimization problem. Among the various regularizers, *lasso*, proposed by (Tibshirani, 1996) is the most widely used because of its mathematical comprehensiveness.

This paper describes a method, which we call *relational lasso*, that improves upon the conventional lasso method. We show that relational lasso improves the overall performance of classification compared with that of other lasso methods. This study was motivated through a limitation we observed in conventional lasso: that features could be inter-related, but such dependences are not incorporated within the current regularization. Therefore, the conventional method tends to favor correlated features, which can lead to the importance of non-correlated features being neglected.

Relational lasso overcomes this limitation of conventional lasso by introducing an additional penalty parameter for each feature; this parameter is estimated automatically given the noisy relations among features, where the relations are also automatically generated. While the proposed method does not add to the computational complexity of the conventional regularization method, it improves the quality of classification. We explain the method and show empirical results from tests based on three different text classification and parsing tasks.

Regularization was originally proposed as a way to avoid over-fitting by favoring some small number of features. Our method presented in this article exploits this approach further and attempts to estimate the importance of each feature within the relations it has with other features. As explained in more detail in the following section, attempts along the same line have been made to incorporate underlying relations among features, such as by *fused lasso* through the ordering among features (Tibshirani et al., 2005), or by *group lasso* through groups among features (Yuan and Lin, 2006). However, fused lasso assumes problems with features that can be ordered in some meaningful way, and group lasso requires underlying group information to be configured before the method is applied. The closely related work to ours is *adaptive lasso* (Zou, 2006), which introduces an additional parameter per feature. However, since adaptive lasso does not explicitly process relations among features, the estimation of this additional parameter is completely different from our method. Moreover, as will be empirically shown, our method outperforms adaptive lasso, which in fact performed worse than even the conventional method.

Related work on regularization techniques has shown the potential of regularization not only to prevent over-fitting, but also to serve as a kind of feature selection. Relational lasso provides another step along this line. Here, we show that our method outperforms other lasso methods in different classification tasks. Moreover, it works well even when noisy features are added, something that degrades the performance of other lasso methods.

2 Related Work

As explained, feature selection is the key to our method's effectiveness, and here we summarize the related work done along this line. A substantial number of studies have been done on feature selection techniques, and these techniques can be classified into three categories according to (Guyon and Elisseeff , 2003): wrapper methods, filter methods and embedded methods.

The wrapper is the most naïve way of selecting a subset of features through predictive accuracy (Kohavi and John, 1997). The user searches the possible feature space greedily using an induction algorithm and selects the best subset with the best predictive accuracy. Wrappers with greedy algorithms can be computationally expensive and the stepwise selection is often trapped into a local optimal solution. Since the possible number of subsets for a set is exponential to the number of features in the original set, in practice the user typically defines the subset arbitrarily depending on the category of features, as was tested by (Scott and Matwin , 1999). This makes impossible any fine adjustment as to which individual features to use.

Filter methods, on the other hand, select good features according to some criteria, thus providing the means for selecting individual features. These methods have been extensively studied, and a good overview is available in (Manning, 1999). The representatives of the evaluation function for choosing good/bad features are chi-squares and mutual information, and features having higher scores for these functions are considered good features. While filtering methods are effective and therefore often used, these methods are independent of the learning method that they are used with. Moreover, the performance is not guaranteed to improve even though feature selection is used.

The last category is embedded methods, where the feature selection is embedded within the overall classification problem. The decision tree is an example of an embedded method, and machine learning techniques using pruning steps have been studied (Perkins et al., 2003). *Lasso*, an acronym for "least absolute shrinkage and selection operator", using the L_1 norm (Tibshirani, 1996) is a computationally efficient method for simultaneously achieving the estimation and feature selection.

Although lasso helps achieve an effective model, the L_1 norm could cause biased estimation among features (Knight and Fu, 2000) by not being able to distinguish between truly significant and noisy features.

To cope with this problem, (Zou and Hastie, 2005) proposed the *elastic net* method, which is expressed as the conjunct of L_1 and L_2 norms of the feature weights. They show that this method works when the number of features substantially exceeds the number of learning data, and also when there is strong correlation between some features. Another proposal is *fused lasso*, which incorporates the order of features (as found in their numbers, such as found in the case when each image pixel value forms a feature) (Tibshirani et al., 2005). Here, the target application is protein mass spectroscopy and gene expression data, and the method is only applicable to a target where the order among features is explicit, as in the case of gene or image pixels. (Yuan and Lin, 2006) proposed *grouped lasso*, which incorporates underlying groups among features. The fused and grouped lasso methods require configuration of the structure among features.

Recently, a new approach called *weighted lasso* is proposed which calculates the L_1 norm on features, each of which is weighted. As one method, (Zou, 2006) proposed a two-step approach called *adaptive lasso*. This paper proposes an alternative weighted lasso method, in which the estimation of the weights is processed differently from that of adaptive lasso. Although the procedure of adaptive lasso is closest to relational lasso, the learning of adaptive lasso does not explicitly handle the relations among features.

All these methods are attempts to incorporate the relations, or structure, among features —such as dependence, ordering and groups— into machine learning through the framework of regularization. Although such dependence is not always given or tractable, we believe this information can be learned from some automatically generated noisy relation among features.

3 L1-Regularization of Multi-Class Logistic Regression

Before going on to the main points of relational lasso, let us summarize the regularization framework that we adopt. Regularization is the general method used in classification. The target function has two terms, one for fitting and another for regularization. This second term penalizes the weights acquired by each feature, typically by incorporating the addition of their norms into a target function for the classifier. This prevents the target function becoming too over-fitted by favoring some specific sets of features. In this sense, the regularization term can be considered as serving for feature selection.

Of the various ways to define the target function, in this paper we focus on the multi-class logistic regression model and L1-regularization; namely, the lasso method.

The fitting function adopted in this paper is a multi-class logistic regression model, denoted as LR in the following. LR is used to model the relationship between the input vectors $\boldsymbol{x} = \mathbb{R}^n$ and labels $y \in \boldsymbol{Y}$. The conditional probability for a label y given x is defined as

$$\begin{aligned} p(y|x;\boldsymbol{w}) &= \frac{1}{Z(\boldsymbol{x})} \exp\left(\boldsymbol{w}^T \phi(\boldsymbol{x}, y)\right) \\ Z(x) &= \sum_{y'} \exp\left(\boldsymbol{w}^T \phi(\boldsymbol{x}, y')\right), \end{aligned}$$

where $\phi(\boldsymbol{x}, y) \in \mathbb{R}^m$ is the feature vector and $\boldsymbol{w} \in \mathbb{R}^m$ is the weight vector. When the training examples $\{(x_i, y_i)\}(i = 1, \cdots, l)$ are given, minimization of the loss function

$$L(\boldsymbol{w}) = -\sum_i \log p(y_i|\boldsymbol{x}_i;\boldsymbol{w}),$$

is equivalent to the maximum likelihood estimation.

Regularization makes it possible to obtain a good model for LR, without restricting the number of features, by imposing appropriate restrictions on weight $\boldsymbol{w}$. Of the different ways of regularization, in this paper we adopt (Tibshirani, 1996)'s method of imposing an L_1 norm on parameters because of its mathematical simplicity, especially when applied with LR. This method is called lasso and facilitates both estimation and automatic variable selection. When applying this lasso to logistic regression, the MAP estimation of weights for each feature is given by the following formula, the target function to be optimized:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} L(\boldsymbol{w}) + \lambda \sum_i |w_i|, \qquad (1)$$

where λ is the parameter defining the strength of the regularization term's influence on the optimization.

Although our proposal applies in general to various types of target function, in this paper we examine its effectiveness within this particular target function. This target function was chosen because the target function of LR-lasso is mathematically comprehensive, so multi-class logistic regression and lasso are widely applied. Further investigation to determine whether our method works well for other target functions will be part of our future work.

4 Relational Lasso —The Proposed Method

The limitation of conventional lasso is that relations among features cannot be incorporated. For example, highly correlated features which lead towards a higher performance could all acquire relatively large weights. This would lead to favoring a single aspect that counts for the classification and neglecting other minor but still important aspects which would enable better classification. This happens when a high correlation is found among features. Therefore, when one feature is favored, the other correlated features must be heavily penalized, so that features which count for classification from a different aspect are more favored.

To express this, we adopt the weighted lasso approach, so an additional penalizing parameter α_i for each feature i is introduced in the second term of formula (1):

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} L(\boldsymbol{w}) + \lambda \sum_i \alpha_i |w_i|. \qquad (2)$$

The solution found here, subject to the L_1 regularizer, is equivalent to the solution obtained from the constrained optimization problem:

$$\begin{aligned} &\underset{\boldsymbol{w}}{\text{minimize}} && L(\boldsymbol{w}), \\ &\text{s.t.} && \sum_i \alpha_i |w_i| \leq \gamma. \end{aligned}$$

The parameter γ corresponds to λ of formula (2). Each parameter α_i determines the penalty for w_i and directly affects the importance of the ith feature.

Previous work on adaptive lasso (Zou, 2006) also introduces an additional parameter, such as α_i, in addition to the weight parameter. Adaptive lasso focuses on the presence of oracle properties[1], and works in two stages of optimization. First weight w_i is learned with conventional lasso. Second, L_1 norm is re-weighted with the parameters α_i as $\alpha_i = 1/|\hat{w}_i|^\delta$ being set from initial lasso estimator $\hat{\boldsymbol{w}}$ using a parameter δ, and the optimization problem is processed using $\boldsymbol{\alpha}$. Therefore, their way of learning this additional parameter does not explicitly concern the exploitation of additional information different from the original weight w_i. On the other hand, our α is estimated given a noisy relation among features, thus it plays a different role from w. In this sense, the way to handle this additional parameter for relational lasso is completely different from adaptive lasso. In other words, the originality of our method lies in using α to express the relations between underlying features.

The relations between features are denoted as R, which is provided to the proposed algorithm and used to estimate α. R denotes a pairwise dependence relation between features. That is, if there are m features in total, $R \subset (1, \cdots, m) \times (1, \cdots, m)$. Unlike previous work such as fused (Tibshirani et al., 2005) and grouped lasso (Yuan and Lin, 2006), R in our work is a noisy relation which can be automatically obtained by scanning through features.

Although there are various possibilities for obtaining R, one way is through the *inclusion* relation among features. Given a pair of features p and q, the feature q *includes* p, if in every data of the learning data, when the value of feature q is non-zero, the value of feature p is always non-zero. For example, for the case of the adjective "economic" and its stem "econom", the latter includes the former while also being a stem for other terms such as "economy" and "economist". In the final classification, it is unknown which of "econom" and "economic" counts. For part-of-speech tagging, "econom" would not provide much information since it does not have a complete form, but for topic estimation, "econom" might provide sufficient information by representing the terms "economic", "economy" and "economist". In both cases, when the two words appear as features, they share a tight relation and when one is given high importance, the other will as well in conventional lasso. In relational lasso, if one representative is selected, then other similar features in the same group will acquire less importance by having a larger penalty.

The overall procedure is shown in Procedure 1. The procedure obtains three kinds of learning data input, a relation among features, and parameter values. Before optimizing the target function, denoted in the second line from the bottom, the procedure calculates α depending on the given relation R among features. This procedure is expressed in terms of a while-structure, which enables the adjustment of penalty parameters for highly correlated features. The processed feature is held in set F to avoid any duplicate processing of features.

In the while-structure, features are selected one at a time in the order of larger values of $\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}}$, with $\boldsymbol{w}$ being the zero vector[2]. The number of the selected feature is denoted as k^*. Then, for all ks which are related to k^* in R, the α is enlarged by a

[1]Oracle property (Fan and Li , 2001) is satisfied if the optimization problem can correctly select the nonzero weights with probability converging to one and the estimators of the nonzero weights are asymptotically normal with the same means and covariance that they would have if the zero coefficients were known in advance.

[2]There are other possibilities for this order of processing features, such as randomizing the order. In the Grafting method (Perkins et al., 2003), the processed feature is selected by calculating $\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}}$ every time in the while-structure, which is also possible with relational lasso. This however remains as future work.

Procedure 1 Relational Lasso

```
**Input:**
  • $(x_i, y_i)(i = 1, \cdots, n)$
  • Parameters $\lambda, a_1, a_2$
  • Relation among $m$ features
    $R \subset (1, \cdots, m) \times (1, \cdots, m)$
$\boldsymbol{\alpha} = \mathbf{1}, \boldsymbol{w} = \mathbf{0}, F = \{\}$
$\boldsymbol{v} = \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}}$
while $|F| < m$ do
  $k^* = \arg\max_{k \notin F} |v_k|$
  for all $k \notin F$ and $(k^*, k) \in R$ do
    $\alpha_k = \alpha_k + a_1$
  end for
  for all $k \notin F$ and $(k, k^*) \in R$ do
    $\alpha_k = \alpha_k + a_2$
  end for
  $F = F \cup k^*$
end while
$\boldsymbol{w} = \arg\min_{\boldsymbol{w}} L(\boldsymbol{w}) + \lambda \sum_k \alpha_k |w_k|$
return $\boldsymbol{w}$
```

certain constant a_1 and a_2, depending on whether the feature k is included or k^* is included. When the while procedure ends, F includes all the features. Finally, $\boldsymbol{w}$ is estimated in terms of LR-lasso, where the second term is weighted further with the thus estimated α. When some specific w_k becomes zero, this means that the weight is considered as not selected for the classification task.

One further improvement that might be possible for the above procedure is to repeat the while-structure and the estimation of w, so that α and w perform co-training; this also remains for our future work. Moreover, the procedure presented here remains an ad hoc modification of conventional lasso based on our motivation. A more proper mathematical reformulation of this method will be part of our future work.

5 Evaluation

5.1 Experimental Settings

We will consider the following two feature sets.

- Standard features
- Standard and additional features

Here, an additional feature set is introduced so that it can be noisy with respect to the classification. Therefore, the interest lies in whether the performance is better when we have the additional features than it is when we have only the standard features.

We consider three methods:

- Conventional lasso (Tibshirani, 1996)
- Adaptive lasso (Zou, 2006)
- Relational lasso (proposed method)

We are interested in whether relational lasso performs better than the other methods. In practice we can select the best λ parameter used for lasso methods, using cross-validation, although we examined multiple λs, which determine the strength of the regularizers' influence as shown in formulas (1) and (2) of Sections 3 and 4.

The other parameters introduced in Section 4 are set as follows. Adaptive lasso has the parameter $\delta = 1$, which is set as the common choice in (Krämer et al., 2009) and the parameter α_i is defined from initial estimator $\hat{\boldsymbol{w}}$ as follows:

$$\alpha_i = \max\left\{\frac{1}{|\hat{w_i}|}, 1\right\}.$$

For relational lasso, parameters a_1 and a_2 were each set to 1. For L_1 regularized LR, a coordinate descent method is implemented by modifying LIBLINEAR[3]. Coordinate descent methods have been widely applied elsewhere because of their suitability for application to higher-order problems (Yuam et al., 2010).

For each pair of a feature and a method, we considered the following three problems of text classification, polarity estimation, and statistical parsing. The next three sections explain the standard and additional feature sets, the relation among features R, and the evaluation scores.

Task 1: Text Classification

Twenty Newsgroups (20NG)[4] were used as the dataset for text classification. This collection contains 18,846 English documents partitioned across 20 different news groups.

The data was sorted by date, with the first 60% used as a training set and the remaining 40% used as a test set. A simple bag of words was used

[3]http://www.csie.ntu.edu.tw/~cjlin/liblinear/

[4]provided by Jason Rennie, http://people.csail.mit.edu/jrennie/20Newsgroups/

Table 1: Features used for dependency parsing

unigram	for w in $w_{i-1}, w_i, w_j, w_{j+1}, w_{j+2}, w_{j+2}$	pos(w), lex(w)
	for w in $w_{i-1}, w_i, w_j,$	pos(w^{left}), lex(w^{left})
	for w in $w_{i-1}, w_i,$	pos(w_i^{right}), lex(w_i^{right}), pos(w_i^{head}),lex(w_i^{head})
bigram	for (v, w) in $(w_i, w_j), (w_{i-1}, w_j)$	pos(v)pos(w),pos(v)lex(w),lex(v)pos(w),lex(v)lex(w)
add bigram	for (v, w) in $(w_i, w_{j+1}), (w_j, w_{j+1})$	pos(v)pos(w),pos(v)lex(w),lex(v)pos(w),lex(v)lex(w)
add preposition	for w in $w_{j+1}, w_{j+2}, w_{j+3}$ (if w_j is a preposition)	lex(w_i)lex(w_j)pos(w), pos(w_i)lex(w_j)lex(w)

as the standard feature sets, whereas all stems of all words were used as additional features. R was defined as the relation between each word and its stems.

A multi-class classification task is typically evaluated by macro and micro F1 values, so we also provided these values.

Task 2: Polarity Estimation

Polarity dataset v2.0[5] was used as the second data set. The content of each data was a movie review in text, tagged with the sentiment of positive or negative. The data consisted of 1,000 positive and 1,000 negative reviews. Since the data set was small, the average accuracy was obtained through 10-fold cross validation.

Feature sets were basically the same as for Task 1, where the standard was a bag of words, the additional set consisted of word stems, and relation R was the relation among words and their stems.

The evaluation was based on the accuracy of the binary classification of positive/negative.

Task 3: Parsing

We also tested the methods on a parsing task, which was a task drastically different from tasks 1 and 2. We used CoNLL-X formatted sentences from the Wall Street Journal section of the Penn Tree-bank. Sections 2-21 were used as training data (39,832 sentences), and section 23 was used as test data (2,416 sentences).

The parsing algorithm we tested is the standard shift-reduce parsing proposed by (Nivre, 2003), where the parsing proceeds by successive determination of the relation between two words (denoted as w_i and w_j). Such a determination is considered a 4-class classification problem that is modeled and learned by LR, augmented by the three lasso methods being evaluated.

The standard features used are listed in Table 1. Here, pos(w) indicates the part of speech of the word w, where w_i indicates the ith word of a given sentence, and w_i^{left} indicates the already parsed dependent word of w_i placed to its farthest left side. The additional feature set included all dependent words involving w_i and w_j, and all bigrams concerning words used as features in the standard set. In our dependency parsing task, we measured the word accuracy which was defined as the ratio of words assigned correct heads divided by the total of all words.

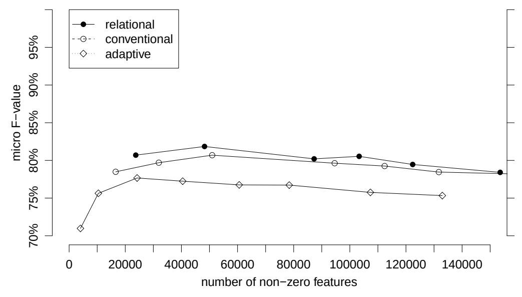

Figure 1: Task 1: Micro F1 values for the number of non-zero features

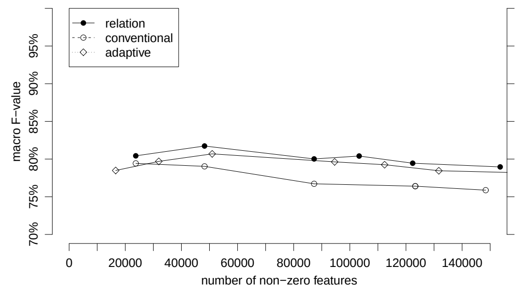

Figure 2: Task 1: Macro F1 values for the number of non-zero features

[5]provided by Bo Pang, `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

5.2 Results

Figures 1 and 2 show the results for Task 1, Figure 3 those for Task 2, and Figure 4 those for Task 3. Horizontal axes show the number of features and vertical axes show accuracy. Each graph has three lines, indicating the conventional, adaptive and relational lasso methods applied to the standard and additional features all together. Each line has five points, each corresponding to a different value of λ. The horizontal coordinate was determined by counting how many features remained non-zero for each value of λ.

Overall, all figures, except for Figure 3 show that relational lasso outperformed the adaptive and conventional lasso methods. This was to be expected, since relational lasso has the relation R as input, unlike the conventional lasso method. This confirms that information from the underlying R does improve lasso performance. Curiously, the performance of adaptive lasso for some figures was lower than that of the conventional method. The reason for this will be given later in this section.

As Figure 3 and Figure 4 show, the performance was competitive among the three lasso methods, when the number of features were small. However, with a large number of features, relational lasso generally outperforms the other lassos.

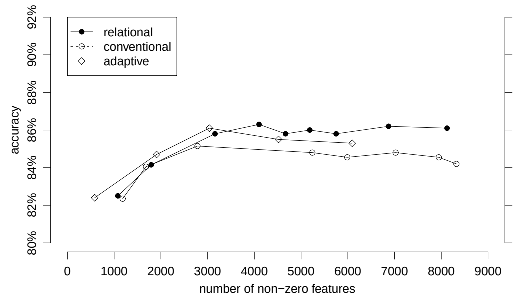

Figure 3: Task 2: Accuracy for the number of non-zero features

It is difficult to compare the performance since different λ values lead to different levels of performance, so the maximum performance obtained by changing the λ value is shown in Table 2. Columns are for different lasso methods with standard and additional feature sets, whereas rows represent different tasks. Note that the best values of λ differ depending on the pairs of methods and features.

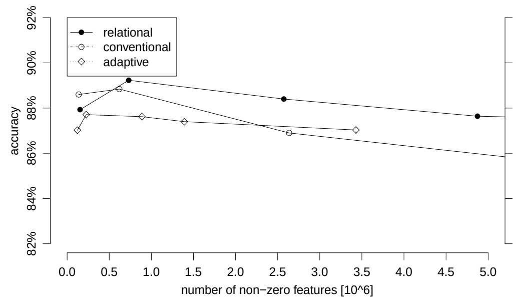

Figure 4: Task 3: Word accuracy for the number of non-zero features

Overall, the last column presents the highest performance in each row, thus suggesting the effectiveness of relational lasso.

For Task 2, when features of standard and additional sets were used, the performance of the conventional method decreased compared to that when only the standard set was used. This could happen if the additional feature set is noisy and the regularizer cannot exploit the useful information from the additional set of features. On the other hand, the performance of relational and adaptive lasso for the same task was improved by extracting the useful information; that is, the performance was higher than when using only the standard features. This shows that the use of underlying information among features enhances the overall performance.

For Tasks 1 and 3, adding features led to better performance than when using only the standard set. Note, though, that the performance increase was greatest for relational lasso. Thus, relational lasso is the best among the three lasso methods at exploiting information, and thus performs better in terms of accuracy.

In this table, too, we see that for Task 2, the performance of adaptive lasso is below that of the conventional lasso. We consider the reason for this to be as follows. Since the optimization for adaptive lasso is done in two stages, some of the features are dropped within the first stage. In the second stage, these features will never re-acquire any importance. In other words, the feature selection must be done at the very end in order to preserve the possibility of some features to re-acqure the importance through learning.

Before ending, we must note the impact of relational lasso on the speed of overall processing.

Table 2: Maximum Performance among Various Values of λ for Three Lasso Methods

Lasso Methods	Conventional		Adaptive		Relational	
Feature Sets	Std	Std+Add	Std	Std+Add	Std	Std+Add
Task 1 (micro)	79.67%	81.03%	76.78%	77.16%	78.87%	**81.81**%
Task 1 (macro)	79.43%	80.69%	76.53%	77.67%	78.52%	**81.72**%
Task 2	85.81%	84.95%	85.56%	86.1%	84.82%	**86.45**%
Task 3	75.87%	88.81%	74.64%	87.71%	75.97%	**89.23**%

The pre-processing to obtain α is very fast, since it only scans the number of features once. The bottleneck of the procedure lies in the estimation of w since this requires convergence through a repetitive procedure. Therefore, the computational complexity of relational lasso will not change even with α within the regularizer, and the overall speed of relational lasso is almost the same as that of the conventional method. In contrast, adaptive lasso requires twice as much time since the bottleneck part is done twice. In this sense, our method outperforms adaptive lasso in speed and is not significantly slower than conventional, at least for the settings we have examined in this section.

6 Conclusion

Relational lasso utilizes relations among features to better exploit information through regularization, especially through lasso methods. The conventional lasso method is not designed to incorporate relations among features, and this leads to biased weighting of a group of features having similar behavior. Relational lasso controls such relations underlying features by introducing a penalty parameter for each feature. The penalty increases when a feature is related to some other feature having less of a penalty. This parameter score is incorporated as the regularization term of the target machine learning function for the optimization objective.

We compared relational lasso to the conventional method and the adaptive lasso proposed by (Zou, 2006), which also uses an additional parameter per feature. We evaluated the methods based on how well they performed three tasks of text categorization, polarity estimation, and parsing. Relational lasso outperformed the other lasso methods in these tasks. Moreover, the performance of the conventional lasso methods deteriorated when noisy features were added, while relational lasso successfully extracted useful information from these features and its performance improved.

As part of our future work, we plan to investigate whether our method works for other tasks such as tagging, and with other target functions. Moreover, there are many directions we can take to further improve the method, such as through co-training. Last, it will be interesting to see how our method can be mathematically reformulated.

References

Jianqing Fan, Runze Li. 2001. Variable selection via non-concave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, vol.96, pp.1348–1360.

Isabelle Guyon, André Elisseeff. 2003. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, vol.3, pp.1157–1182.

G. V. Kass. 1980. An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society. Series C*, vol.29, pp.119–127.

Keith Knight, Wenjiang Fu. 2000. Asymptotics for lasso-type estimators. *The Annals of Statistics*, vol.28, pp.1356–1378.

Ron Kohavi, George H. John. 1997. Wrappers for feature subset selection, *Artificial intelligence*, vol.97, pp.273–324.

Nicole Krämer, Juliane Schäfer, Anne-Laure Boulesteix. 2009. Regularized estimation of large-scale gene association networks using graphical Gaussian models. *BMC Bioinformatics*, vo.10.

Christopher Manning, Hinrich Schuetze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. *Proceedings of the 8th International Workshop on Parsing Technologies*, pp. 149-160.

Simon Perkins, Kervin Lacker, and James Theiler. 2003. Grafting: Fast, Incremental Feature Selection by Gradient Descent in Function Space. *The Journal of Machine Learning Research*, vol.3, pp.1333–1356.

Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. 2007. A review of feature selection techniques in bioinformatics. *Bioinformatics*, vol.23, num.19, pp.2507–2517.

Sam Scott, Stan Matwin. 1999. Feature engineering for text classification. *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pp.379–388.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, Series B, vol.58, pp.267–288.

Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight 2005. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society*, Series B, vol.67, pp.91–108.

Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, Chih-Jen Lin. 2010. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *The Journal of Machine Learning Research*, vol.11, pp.3183–3234.

Ming Yuan, Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society*, Series B, vol.68, pp.49–67.

Ying Yang, Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. *Proceedings of the Fourteenth International Conference on Machine Learning*, pp.412–420.

Hui Zou. 2006. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, vol.101, pp.1418–1429.

Hui Zou, Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, Series B, vol.67, pp.301–320.