Supervised Model Learning with Feature Grouping based on a Discrete Constraint

Jun Suzuki and Masaaki Nagata
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{suzuki.jun, nagata.masaaki}@lab.ntt.co.jp

Abstract

This paper proposes a framework of supervised model learning that realizes feature grouping to obtain lower complexity models. The main idea of our method is to integrate a discrete constraint into model learning with the help of the dual decomposition technique. Experiments on two well-studied NLP tasks, dependency parsing and NER, demonstrate that our method can provide state-of-the-art performance even if the degrees of freedom in trained models are surprisingly small, *i.e.,* 8 or even 2. This significant benefit enables us to provide compact model representation, which is especially useful in actual use.

1 Introduction

This paper focuses on the topic of supervised model learning, which is typically represented as the following form of the optimization problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left\{ \mathcal{O}(\mathbf{w}; \mathcal{D}) \right\}, \quad \mathcal{O}(\mathbf{w}; \mathcal{D}) = \mathcal{L}(\mathbf{w}; \mathcal{D}) + \Omega(\mathbf{w}), \tag{1}$$

where $\mathcal{D}$ is supervised training data that consists of the corresponding input $\mathbf{x}$ and output $\mathbf{y}$ pairs, that is, $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$. $\mathbf{w}$ is an N-dimensional vector representation of a set of optimization variables, which are also interpreted as *feature weights*. $\mathcal{L}(\mathbf{w}; \mathcal{D})$ and $\Omega(\mathbf{w})$ represent a loss function and a regularization term, respectively. Nowadays, we, in most cases, utilize a supervised learning method expressed as the above optimization problem to estimate the feature weights of many natural language processing (NLP) tasks, such as text classification, POS-tagging, named entity recognition, dependency parsing, and semantic role labeling.

In the last decade, the L_1-regularization technique, which incorporates L_1-norm into $\Omega(\mathbf{w})$, has become popular and widely-used in many NLP tasks (Gao et al., 2007; Tsuruoka et al., 2009). The reason is that L_1-regularizers encourage feature weights to be zero as much as possible in model learning, which makes the resultant model a sparse solution (many zero-weights exist). We can discard all features whose weight is zero from the *trained model*[1] without any loss. Therefore, L_1-regularizers have the ability to easily and automatically yield *compact* models without strong concern over feature selection.

Compact models generally have significant and clear advantages in practice: instances are faster loading speed to memory, less memory occupation, and even faster decoding is possible if the model is small enough to be stored in cache memory. Given this background, our aim is to establish a model learning framework that can reduce the model complexity beyond that possible by simply applying L_1-regularizers. To achieve our goal, we focus on the recently developed concept of automatic feature grouping (Tibshirani et al., 2005; Bondell and Reich, 2008). We introduce a model learning framework that achieves feature grouping by incorporating a discrete constraint during model learning.

2 Feature Grouping Concept

Going beyond L_1-regularized sparse modeling, the idea of '*automatic feature grouping*' has recently been developed. Examples are fused lasso (Tibshirani et al., 2005), grouping pursuit (Shen and Huang, 2010), and OSCAR (Bondell and Reich, 2008). The concept of automatic feature grouping is to find accurate models that have fewer degrees of freedom. This is equivalent to enforce every optimization variables to be equal as much as possible. A simple example is that $\hat{\mathbf{w}}_1 = (0.1, 0.5, 0.1, 0.5, 0.1)$ is preferred over $\hat{\mathbf{w}}_2 = (0.1, 0.3, 0.2, 0.5, 0.3)$ since $\hat{\mathbf{w}}_1$ and $\hat{\mathbf{w}}_2$ have two and four unique values, respectively.

There are several merits to reducing the degree

[1] This paper refers to model after completion of (supervised) model learning as "trained model"

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 18–23,
Sofia, Bulgaria, August 4-9 2013.

of freedom. For example, previous studies clarified that it can reduce the chance of over-fitting to the training data (Shen and Huang, 2010). This is an important property for many NLP tasks since they are often modeled with a high-dimensional feature space, and thus, the over-fitting problem is readily triggered. It has also been reported that it can improve the stability of selecting non-zero features beyond that possible with the standard L_1-regularizer given the existence of many highly correlated features (Jörnsten and Yu, 2003; Zou and Hastie, 2005). Moreover, it can dramatically reduce model complexity. This is because we can merge all features whose feature weight values are equivalent in the trained model into a single feature cluster without any loss.

3 Modeling with Feature Grouping

This section describes our proposal for obtaining a feature grouping solution.

3.1 Integration of a Discrete Constraint

Let $\mathcal{S}$ be a finite set of discrete values, *i.e.*, a set integer from -4 to 4, that is, $\mathcal{S}=\{-4,\ldots,-1,0,1,\ldots,4\}$. The detailed discussion how we define $\mathcal{S}$ can be found in our experiments section since it deeply depends on training data. Then, we define the objective that can simultaneously achieve a feature grouping and model learning as follows:

$$\begin{aligned}\mathcal{O}(\mathbf{w};\mathcal{D}) &= \mathcal{L}(\mathbf{w};\mathcal{D})+\Omega(\mathbf{w})\\ \text{s.t.}\quad \mathbf{w} &\in \mathcal{S}^N.\end{aligned} \tag{2}$$

where $\mathcal{S}^N$ is the cartesian power of a set $\mathcal{S}$. The only difference with Eq. 1 is the additional discrete constraint, namely, $\mathbf{w}\in\mathcal{S}^N$. This constraint means that each variable (feature weight) in trained models must take a value in $\mathcal{S}$, that is, $\hat{w}_n\in\mathcal{S}$, where $\hat{w}_n$ is the n-th factor of $\hat{\mathbf{w}}$, and $n\in\{1,\ldots,N\}$. As a result, feature weights in trained models are automatically grouped in terms of the basis of model learning. This is the basic idea of feature grouping proposed in this paper.

However, a concern is how we can efficiently optimize Eq. 2 since it involves a NP-hard combinatorial optimization problem. The time complexity of the direct optimization is exponential against N. Next section introduces a feasible algorithm.

3.2 Dual Decomposition Formulation

Hereafter, we strictly assume that $\mathcal{L}(\mathbf{w};\mathcal{D})$ and $\Omega(\mathbf{w})$ are both convex in $\mathbf{w}$. Then, the properties of our method are unaffected by the selection of $\mathcal{L}(\mathbf{w};\mathcal{D})$ and $\Omega(\mathbf{w})$. Thus, we ignore their specific definition in this section. Typical cases can be found in the experiments section. Then, we reformulate Eq. 2 by using the dual decomposition technique (Everett, 1963):

$$\begin{aligned}\mathcal{O}(\mathbf{w},\mathbf{u};\mathcal{D}) &= \mathcal{L}(\mathbf{w};\mathcal{D})+\Omega(\mathbf{w})+\Upsilon(\mathbf{u})\\ \text{s.t.}\quad \mathbf{w}&=\mathbf{u},\ \text{and}\ \mathbf{u}\in\mathcal{S}^N.\end{aligned} \tag{3}$$

Difference from Eq. 2, Eq. 3 has an additional term $\Upsilon(\mathbf{u})$, which is similar to the regularizer $\Omega(\mathbf{w})$, whose optimization variables $\mathbf{w}$ and $\mathbf{u}$ are tightened with equality constraint $\mathbf{w}=\mathbf{u}$. Here, this paper only considers the case $\Upsilon(\mathbf{u})=\frac{\lambda_2}{2}||\mathbf{u}||_2^2+\lambda_1||\mathbf{u}||_1$, and $\lambda_2\geq 0$ and $\lambda_1\geq 0$[2]. This objective can also be viewed as the decomposition of the standard loss minimization problem shown in Eq. 1 and the additional discrete constraint regularizer by the dual decomposition technique.

To solve the optimization in Eq. 3, we leverage the *alternating direction method of multiplier* (ADMM) (Gabay and Mercier, 1976; Boyd et al., 2011). ADMM provides a very efficient optimization framework for the problem in the dual decomposition form. Here, $\boldsymbol{\alpha}$ represents dual variables for the equivalence constraint $\mathbf{w}=\mathbf{u}$. ADMM introduces the augmented Lagrangian term $\frac{\rho}{2}||\mathbf{w}-\mathbf{u}||_2^2$ with $\rho>0$ which ensures strict convexity and increases robustness[3].

Finally, the optimization problem in Eq. 3 can be converted into a series of iterative optimization problems. Detailed derivation in the general case can be found in (Boyd et al., 2011). Fig. 1 shows the entire model learning framework of our proposed method. The remarkable point is that ADMM works by iteratively computing one of the three optimization variable sets $\mathbf{w}$, $\mathbf{u}$, and $\boldsymbol{\alpha}$ while holding the other variables fixed in the iterations $t=1,2,\ldots$ until convergence.

Step1 ($\mathbf{w}$-update): This part of the optimization problem shown in Eq. 4 is essentially Eq. 1 with a 'biased' L_2-regularizer. 'bias' means here that the direction of regularization is toward point $\mathbf{a}$ instead of the origin. Note that it becomes a standard L_2-regularizer if $\mathbf{a}=\mathbf{0}$. We can select any learning algorithm that can handle the L_2-regularizer for this part of the optimization.

Step2 ($\mathbf{u}$-update): This part of the optimization problem shown in Eq. 5 can be rewritten in the

[2]Note that this setting includes the use of only L_1-, L_2-, or without regularizers (L_1 only: $\lambda_1>0$ and $\lambda_2=0$, L_2 only: $\lambda_1=0$ and $\lambda_2>0$, and without regularizer: $\lambda_1=0$, $\lambda_2=0$).

[3]Standard dual decomposition can be viewed as $\rho=0$

Input: Training data:$\mathcal{D}$, parameters:ρ, ξ, ϵ_{primal}, and ϵ_{dual}
Initialize: $\mathbf{w}^{(1)} = \mathbf{0}$, $\mathbf{u}^{(1)} = \mathbf{0}$, $\boldsymbol{\alpha}^{(1)} = \mathbf{0}$, and $t = 1$.

Step1 $\mathbf{w}$-update:
Solve $\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w}} \{\mathcal{O}(\mathbf{w}; \mathcal{D}, \mathbf{u}^{(t)}, \boldsymbol{\alpha}^{(t)})\}$.
For our case,

$$\mathcal{O}(\mathbf{w}; \mathcal{D}, \mathbf{u}, \boldsymbol{\alpha}) = \mathcal{O}(\mathbf{w}; \mathcal{D}) + \frac{\rho}{2}||\mathbf{w} - \mathbf{a}||_2^2, \quad (4)$$

where $\mathbf{a} = \mathbf{u} - \boldsymbol{\alpha}$.

Step2 $\mathbf{u}$-update:
Solve $\mathbf{u}^{(t+1)} = \arg\min_{\mathbf{u}} \{\mathcal{O}(\mathbf{u}; \mathcal{D}, \mathbf{w}^{(t+1)}, \boldsymbol{\alpha}^{(t)})\}$.
For our case,

$$\mathcal{O}(\mathbf{u}; \mathcal{D}, \mathbf{w}, \boldsymbol{\alpha}) = \frac{\lambda_2}{2}||\mathbf{u}||_2^2 + \lambda_1||\mathbf{u}||_1 + \frac{\rho}{2}||\mathbf{b} - \mathbf{u}||_2^2 \quad \text{s.t.} \quad \mathbf{u} \in \mathcal{S}^N, \quad (5)$$

where $\mathbf{b} = \mathbf{w} + \boldsymbol{\alpha}$

Step3 $\boldsymbol{\alpha}$-update:

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \xi(\mathbf{w}^{(t+1)} - \mathbf{u}^{(t+1)}) \quad (6)$$

Step4 convergence check:

$$||\mathbf{w}^{(t+1)} - \mathbf{u}^{(t+1)}||_2^2/N < \epsilon_{\text{primal}} \quad ||\mathbf{u}^{(t+1)} - \mathbf{u}^{(t)}||_2^2/N < \epsilon_{\text{dual}} \quad (7)$$

Break the loop if the above two conditions are reached, or go back to Step1 with $t = t + 1$.

Output: $\mathbf{u}^{(t+1)}$

Figure 1: Entire learning framework of our method derived from ADMM (Boyd et al., 2011).

following equivalent simple form:

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} \{\tfrac{1}{2}||\mathbf{u} - \mathbf{b}'||_2^2 + \lambda_1'||\mathbf{u}||_1\} \quad \text{s.t.} \quad \mathbf{u} \in \mathcal{S}^N, \quad (8)$$

where $\mathbf{b}' = \frac{\rho}{\lambda_2+\rho}\mathbf{b}$, and $\lambda_1' = \frac{\lambda_1}{\lambda_2+\rho}$. This optimization is still a combinatorial optimization problem. However unlike Eq. 2, this optimization can be efficiently solved.

Fig. 2 shows the procedure to obtain the *exact* solution of Eq. 5, namely $\mathbf{u}^{(t+1)}$. The remarkable point is that the costly combinatorial optimization problem is disappeared, and instead, we are only required to perform two feature-wise calculations whose total time complexities is $O(N \log |\mathcal{S}|)$ and fully parallelizable. The similar technique has been introduced in Zhong and Kwok (2011) for discarding a costly combinatorial problem from the optimization with OSCAR-regularizers with the help of proximal gradient methods, *i.e.*, (Beck and Teboulle, 2009).

We omit to show the detailed derivation of Fig. 2 because of the space reason. However, this is easily understandable. The key properties are the following two folds; (i) The objective shown in Eq. 8 is a convex and also symmetric function with respect to $\hat{\mathbf{u}}'$, where $\hat{\mathbf{u}}'$ is the optimal solution of Eq. 8 without the discrete constraint. Therefore, the optimal solution $\hat{\mathbf{u}}$ is at the point where the nearest valid point given $\mathcal{S}^N$ from $\hat{\mathbf{u}}'$ in terms of the L_2-distance. (ii) The valid points given $\mathcal{S}^N$ are always located at the vertexes of *axis-aligned orthotopes (hyperrectangles)* in the parameter space of feature weights. Thus, the solution $\hat{\mathbf{u}}$, which is the nearest valid point from $\hat{\mathbf{u}}'$, can be obtained by individually taking the nearest value in $\mathcal{S}$ from $\hat{u}_n'$ for all n.

Input: $\mathbf{b}' = (b_n')_{n=1}^N$, λ_1', and $\mathcal{S}$.
1, Find the optimal solution of Eq. 8 without the constraint. The optimization of mixed L_2 and L_1-norms is known to have a closed form solution, *i.e.*, (Beck and Teboulle, 2009), that is;

$$\hat{u}_n' = sgn(b_n') \max(0, |b_n'| - \lambda_1'),$$

where $(\hat{u}_n')_{n=1}^N = \hat{\mathbf{u}}'$.
2, Find the nearest valid point in $\mathcal{S}^N$ from $\hat{\mathbf{u}}'$ in terms of the L_2-distance;

$$\hat{u}_n = \arg\min_{u \in \mathcal{S}} (\hat{u}_n' - u)^2$$

where $(\hat{u}_n)_{n=1}^N = \hat{\mathbf{u}}$. This can be performed by a binary search, whose time complexity is generally $O(\log |\mathcal{S}|)$.
Output: $\hat{\mathbf{u}}$

Figure 2: Procedure for solving Step2

Step3 ($\boldsymbol{\alpha}$-update): We perform gradient ascent on dual variables to tighten the constraint $\mathbf{w} = \mathbf{u}$. Note that ξ is the learning rate; we can simply set it to 1.0 for every iteration (Boyd et al., 2011).

Step4 (convergence check): It can be evaluated both primal and dual residuals as defined in Eq. 7 with suitably small ϵ_{primal} and ϵ_{dual}.

3.3 Online Learning

We can select an online learning algorithm for Step1 since the ADMM framework does not require exact minimization of Eq. 4. In this case, we perform one-pass update through the data in each ADMM iteration (Duh et al., 2011). Note that the total calculation cost of our method does not increase much from original online learning algorithm since the calculation cost of Steps 2 through 4 is relatively much smaller than that of Step1.

4 Experiments

We conducted experiments on two well-studied NLP tasks, namely named entity recognition (**NER**) and dependency parsing (**DEPAR**).

Basic settings: We simply reused the settings of most previous studies. We used CoNLL'03 data (Tjong Kim Sang and De Meulder, 2003) for NER, and the Penn Treebank (PTB) III corpus (Marcus et al., 1994) converted to dependency trees for DEPAR (McDonald et al., 2005).

Our decoding models are the Viterbi algorithm on CRF (Lafferty et al., 2001), and the second-order parsing model proposed by (Carreras, 2007) for NER and DEPAR, respectively. Features are automatically generated according to the predefined feature templates widely-used in the previous studies. We also integrated the cluster features obtained by the method explained in (Koo et al., 2008) as additional features for evaluating our method in the range of the current best systems.

Evaluation measures: The purpose of our experiments is to investigate the effectiveness of our proposed method in terms of both its performance and the complexity of the trained model. Therefore, our evaluation measures consist of two axes. Task performance was mainly evaluated in terms of the complete sentence accuracy (**COMP**) since the objective of all model learning methods evaluated in our experiments is to maximize COMP. We also report the $F_{\beta=1}$ score (**F-sc**) for NER, and the unlabeled attachment score (**UAS**) for DEPAR for comparison with previous studies. Model complexity is evaluated by the number of non-zero active features (**#nzF**) and the degree of freedom (**#DoF**) (Zhong and Kwok, 2011). #nzF is the number of features whose corresponding feature weight is non-zero in the trained model, and #DoF is the number of unique non-zero feature weights.

Baseline methods: Our main baseline is L_1-regularized sparse modeling. To cover both batch and online leaning, we selected L_1-regularized CRF (**L1CRF**) (Lafferty et al., 2001) optimized by OWL-QN (Andrew and Gao, 2007) for the NER experiment, and the L_1-regularized *regularized dual averaging* (**L1RDA**) method (Xiao, 2010)[4] for DEPAR. Additionally, we also evaluated L_2-regularized CRF (**L2CRF**) with L-BFGS (Liu and Nocedal, 1989) for NER, and passive-aggressive algorithm (**L2PA**) (Crammer et al., 2006)[5] for DEPAR since L_2-regularizer often provides better results than L_1-regularizer (Gao et al., 2007).

For a fair comparison, we applied the procedure of Step2 as a simple quantization method to trained models obtained from L_1-regularized model learning, which we refer to as (**QT**).

[4]RDA provided better results at least in our experiments than L_1-regularized FOBOS (Duchi and Singer, 2009), and its variant (Tsuruoka et al., 2009), which are more familiar to the NLP community.

[5]L2PA is also known as a loss augmented variant of one-best MIRA, well-known in DEPAR (McDonald et al., 2005).

4.1 Configurations of Our Method

Base learning algorithm: The settings of our method in our experiments imitate L_1-regularized learning algorithm since the purpose of our experiments is to investigate the effectiveness against standard L_1-regularized learning algorithms. Then, we have the following two possible settings; **DC-ADMM**: we leveraged the baseline L_1-regularized learning algorithm to solve Step1, and set $\lambda_1 = 0$ and $\lambda_2 = 0$ for Step2. **DCwL1-ADMM**: we leveraged the baseline L_2-regularized learning algorithm, but without L_2-regularizer, to solve Step1, and set $\lambda_1 > 0$ and $\lambda_2 = 0$ for Step2. The difference can be found in the objective function $\mathcal{O}(\mathbf{w}, \mathbf{u}; \mathcal{D})$ shown in Eq. 3;

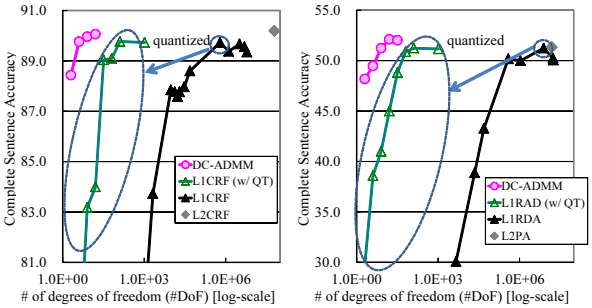
$$\text{(DC-ADMM)}: \ \mathcal{O}(\mathbf{w}, \mathbf{u}; \mathcal{D}) = \mathcal{L}(\mathbf{w}; \mathcal{D}) + \lambda_1 ||\mathbf{w}||_1$$
$$\text{(DCwL1-ADMM)}: \ \mathcal{O}(\mathbf{w}, \mathbf{u}; \mathcal{D}) = \mathcal{L}(\mathbf{w}; \mathcal{D}) + \lambda_1 ||\mathbf{u}||_1$$

In other words, DC-ADMM utilizes L_1-regularizer as a part of base leaning algorithm $\Omega(\mathbf{w}) = \lambda_1 ||\mathbf{w}||_1$, while DCwL1-ADMM discards regularizer of base learning algorithm $\Omega(\mathbf{w})$, but instead introducing $\Upsilon(\mathbf{u}) = \lambda_1 ||\mathbf{u}||_1$. Note that these two configurations are essentially identical since objectives are identical, even though the formulation and algorithm is different. We only report results of DC-ADMM because of the space reason since the results of DCwL1-ADMM were nearly equivalent to those of DC-ADMM.

Definition of $\mathcal{S}$: DC-ADMM can utilize any finite set for $\mathcal{S}$. However, we have to carefully select it since it deeply affects the performance. Actually, this is the most considerable point of our method. We preliminarily investigated the several settings. Here, we introduce an example of template which is suitable for large feature set. Let η, δ, and κ represent non-negative real-value constants, ζ be a positive integer, $\sigma = \{-1, 1\}$, and a function $f_{\eta,\delta,\kappa}(x, y) = y(\eta\kappa^x + \delta)$. Then, we define a finite set of values $\mathcal{S}$ as follows:

$$\mathcal{S}_{\eta,\delta,\kappa,\zeta} = \{f_{\eta,\delta,\kappa}(x, y) | (x, y) \in \mathcal{S}_\zeta \times \sigma\} \cup \{0\},$$

where $\mathcal{S}_\zeta$ is a set of non-negative integers from zero to $\zeta - 1$, that is, $\mathcal{S}_\zeta = \{m\}_{m=0}^{\zeta-1}$. For example, if we set $\eta = 0.1$, $\delta = 0.4$, $\kappa = 4$, and $\zeta = 3$, then $\mathcal{S}_{\eta,\delta,\kappa,\zeta} = \{-2.0, -0.8, -0.5, 0, 0.5, 0.8, 2.0\}$. The intuition of this template is that the distribution of the feature weights in trained model often takes a form a similar to that of the 'power law' in the case of the large feature sets. Therefore, using an exponential function with a scale and bias seems to be appropriate for fitting them.



(a) NER (b) DEPAR

Figure 3: Performance vs. degree of freedom in the trained model for the development data

Note that we can control the upper bound of #DoF in trained model by ζ, namely if $\zeta = 4$ then the upper bound of #DoF is 8 (doubled by positive and negative sides). We fixed $\rho = 1$, $\xi = 1$, $\lambda_2 = 0$, $\kappa = 4$ (or 2 if $\zeta \geq 5$), $\delta = \eta/2$ in all experiments. Thus the only tunable parameter in our experiments is η for each ζ.

4.2 Results and Discussions

Fig. 3 shows the task performance on the development data against the model complexities in terms of the degrees of freedom in the trained models. Plots are given by changing the ζ value for DC-ADMM and L_1-regularized methods with QT. The plots of the standard L_1-regularized methods are given by changing the regularization constants λ_1. Moreover, Table 1 shows the final results of our experiments on the test data. The tunable parameters were fixed at values that provided the best performance on the development data.

According to the figure and table, the most remarkable point is that DC-ADMM successfully maintained the task performance even if #DoF (the degree of freedom) was 8, and the performance drop-offs were surprisingly limited even if #DoF was 2, which is the upper bound of feature grouping. Moreover, it is worth noting that the DC-ADMM performance is sometimes improved. The reason may be that such low degrees of freedom prevent over-fitting to the training data. Surprisingly, the simple quantization method (QT) provided fairly good results. However, we emphasize that the models produced by the QT approach offer no guarantee as to the optimal solution. In contrast, DC-ADMM can truly provide the optimal solution of Eq. 3 since the discrete constraint is also considered during the model learning.

In general, a trained model consists of two parts:

	Test		Model complex.	
NER	COMP	F-sc	#nzF	#DoF
L2CRF	84.88	89.97	61.6M	38.6M
L1CRF	84.85	89.99	614K	321K
(w/ QT $\zeta=4$)	78.39	85.33	568K	8
(w/ QT $\zeta=2$)	73.40	81.45	454K	4
(w/ QT $\zeta=1$)	65.53	75.87	454K	2
DC-ADMM ($\zeta=4$)	84.96	89.92	643K	8
($\zeta=2$)	84.04	89.35	455K	4
($\zeta=1$)	83.06	88.62	364K	2
	Test		Model complex.	
DEPER	COMP	UAS	#nzF	#DoF
L2PA	49.67	93.51	15.5M	5.59M
L1RDA	49.54	93.48	7.76M	3.56M
(w/ QT $\zeta=4$)	38.58	90.85	6.32M	8
(w/ QT $\zeta=2$)	34.19	89.42	3.08M	4
(w/ QT $\zeta=1$)	30.42	88.67	3.08M	2
DC-ADMM ($\zeta=4$)	49.83	93.55	5.81M	8
($\zeta=2$)	48.97	93.18	4.11M	4
($\zeta=1$)	46.56	92.86	6.37M	2

Table 1: Comparison results of the methods on test data (K: thousand, M: million)

feature weights and an indexed structure of feature strings, which are used as the key for obtaining the corresponding feature weight. This paper mainly discussed how to reduce the size of the former part, and described its successful reduction. We note that it is also possible to reduce the latter part especially if the feature string structure is TRIE. We omit the details here since it is not the main topic of this paper, but by merging feature strings that have the same feature weights, the size of entire trained models in our DEPAR case can be reduced to about 10 times smaller than those obtained by standard L_1-regularization, *i.e.*, to 12.2 MB from 124.5 MB.

5 Conclusion

This paper proposed a model learning framework that can simultaneously realize feature grouping by the incorporation of a simple discrete constraint into model learning optimization. This paper also introduced a feasible algorithm, DC-ADMM, which can vanish the infeasible combinatorial optimization part from the entire learning algorithm with the help of the ADMM technique. Experiments showed that DC-ADMM drastically reduced model complexity in terms of the degrees of freedom in trained models while maintaining the performance. There may exist theoretically cleverer approaches to feature grouping, but the performance of DC-ADMM is close to the upper bound. We believe our method, DC-ADMM, to be very useful for actual use.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable Training of L1-regularized Log-linear Models. In Zoubin Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, pages 33–40. Omnipress.

Amir Beck and Marc Teboulle. 2009. A Fast Iterative Shrinkage-thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

Howard D. Bondell and Brian J. Reich. 2008. Simultaneous Regression Shrinkage, Variable Selection and Clustering of Predictors with OSCAR. *Biometrics*, 64(1):115.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Foundations and Trends in Machine Learning.

Xavier Carreras. 2007. Experiments with a Higher-Order Projective Dependency Parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

John Duchi and Yoram Singer. 2009. Efficient Online and Batch Learning Using Forward Backward Splitting. *Journal of Machine Learning Research*, 10:2899–2934.

Kevin Duh, Jun Suzuki, and Masaaki Nagata. 2011. Distributed Learning-to-Rank on Streaming Data using Alternating Direction Method of Multipliers. In *NIPS'11 Big Learning Workshop*.

Hugh Everett. 1963. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Operations Research*, 11(3):399–417.

Daniel Gabay and Bertrand Mercier. 1976. A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximation. *Computers and Mathematics with Applications*, 2(1):17 – 40.

Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 824–831, Prague, Czech Republic, June. Association for Computational Linguistics.

Rebecka Jörnsten and Bin Yu. 2003. Simultaneous Gene Clustering and Subset Selection for Sample Classification Via MDL. *Bioinformatics*, 19(9):1100–1109.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proceedings of ACL-08: HLT*, pages 595–603.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the International Conference on Machine Learning (ICML 2001)*, pages 282–289.

Dong C. Liu and Jorge Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Programming, Ser. B*, 45(3):503–528.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-margin Training of Dependency Parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98.

Xiaotong Shen and Hsin-Cheng Huang. 2010. Grouping Pursuit Through a Regularization Solution Surface. *Journal of the American Statistical Association*, 105(490):727–739.

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*, pages 91–108.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2003*, pages 142–147.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 477–485.

Lin Xiao. 2010. Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization. *Journal of Machine Learning Research*, 11:2543–2596.

Leon Wenliang Zhong and James T. Kwok. 2011. Efficient Sparse Modeling with Automatic Feature Grouping. In *ICML*.

Hui Zou and Trevor Hastie. 2005. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.